Package 'rtemis'

May 19, 2022

Version 0.91

Title Machine Learning and Visualization

Date 2022-04-11

Maintainer Efstathios D. Gennatas <gennatas@lambdamd.org>

Description Advanced Machine Learning and Visualization. Unsupervised Learning (Clustering, Decomposition), Supervised Learning (Classification, Regression), Cross-Decomposition, Bagging, Boosting, Meta-models. Static and interactive graphics.

License GPL (>=3)

URL https://rtemis.lambdamd.org

ByteCompile yes

Depends R (>= 4.1.0)

Imports grDevices, graphics, stats, methods, utils, data.table, R6, future, crayon, htmltools

Suggests ada, arm, bartMachine, base64enc, dbscan, car, cluster, C50, data.tree, deepnet, dendextend (>= 0.18.0), DiagrammeR, DiagrammeRsvg, doParallel, dplyr, earth, EMCluster, extraTrees, evtree, e1071, fansi, fastICA, flexclust, FNN,

fpc,
fs,
future.apply,
gamsel,
gamsel2,
gbm,
gbm3,
geojsonio,
geosphere,
ggplot2,
glmnet,
GPArotation,
grid,
gsubfn,
heatmaply,
hopach,
htmlwidgets,
h2o,
ica,
igraph,
imager,
inTrees,
iRF,
JuliaCall,
keras,
kernlab,
knitr,
lars,
lavaan,
leaflet,
leaps,
lle,
matrixStats,
MASS,
mda,
meanShiftR,
mgcv,
mlbench,
mice,
missForest,
missRanger,
networkD3,
nlme,
nnet,
NMF,
np,
nsprcomp,
oro.nifti,
partykit,
pbapply,
plotly,
polspline,

PPtree,
pROC,
progress,
progressr,
psych,
PMA,
png,
plotrix,
plyr,
pmml,
PRROC,
pvclust,
qrnn,
randomForest,
randomForestSRC,
ranger,
rCUR,
RColorBrewer,
RNifti,
ROCR,
rpart,
rpart.plot,
rstudioapi,
Rtsne,
rsvg,
R.utils,
scales (>= 0.2.5),
sgd,
sparklyr,
sparseLDA,
splines2,
spls,
survival,
tgp,
threejs,
vegan,
visNetwork,
xgboost,
rmarkdown,
usmap,
uwot,
testthat (>= 3.0.0)

Remotes egenn/gamsel2, gbm-developers/gbm3

RoxygenNote 7.1.2

VignetteBuilder knitr

Config/testthat/edition 3

R topics documented:

`rtemis-package` **rtemis**: *Machine Learning and Visualization*

Description

Advanced Machine Learning made easy, efficient, reproducible

Online Documentation and Vignettes

https://rtemis.lambdamd.org

System Setup

There are some options you can define in your .Rprofile (usually found in your home directory), so you do not have to define each time you execute a function.

rt.theme General plotting theme; set to e.g. "whiteigrid" or "darkgraygrid"

rt.palette Name of default palette to use in plots. See options by running ‘rtpalette()‘

rt.font Font family to use in plots.

rt.cores Number of cores to use. By default, rtemis will use available cores reported by future::availableCores(). In shared systems, you should limit this as appropriate.

future.plan Default plan to use for parallel processing.

Visualization

Static graphics are handled using the `mplot3` family. Dynamic graphics are handled using the `dplot3` family.

Supervised Learning

Functions for Regression and Classification begin with `s_*`. Run modSelect to get a list of available algorithms The documentation of each supervised learning function indicates in brackets, after the title whether the function supports classification, regression, and survival analysis `[C, R, S]`

Clustering

Functions for Clustering begin with `c_*`. Run clustSelect to get a list of available algorithms

Decomposition

Functions for Decomposition and Dimensionality reduction begin with `d_*`. Run decomSelect to get a list of available algorithms

Cross-Decomposition

Functions for Cross-Decomposition begin with `x_*`. Run xdecomSelect to get a list of available algorithms

Meta-Modeling

Meta models are trained using `meta*` functions.

Notes

Function documentation includes input type (e.g. "String", "Integer", "Float", etc) and range in interval notation where applicable. For example, Float: [0, 1)" means floats between 0 and 1 including 0, but excluding 1

For all classification models, the outcome should be provided as a factor, with the first level of the factor being the 'positive' class, if applicable. A character vector supplied as outcome will be converted to factors, where by default the levels are set alphabetically and therefore the positive class may not be set correctly.

`anyConstant` *Check for constant columns*

Description

Checks if any column of a data frame have zero variance

Usage

```
anyConstant(x)
```

Arguments

`x`	Input Data Frame

Author(s)

E.D. Gennatas

`as.boost` `as.boost` *Place model in boost structure*

Description

`as.boost` Place model in boost structure

Usage

```
as.boost(
  object,
  x = NULL,
  y = NULL,
  x.valid = NULL,
  y.valid = NULL,
  learning.rate = 1,
  init = 0,
  apply.lr = TRUE,
  tolerance = 1e-05,
  tolerance.valid = 1e-05
)
```

Arguments

`object`	rtMod model
`x`	Data.frame, optional: if provided, use to calculate fitted values of new boost object
`y`	Float, vector: Outcome
`x.valid`	Data.frame; optional: Validation data
`y.valid`	Float, vector; optional: Validation outcome
`learning.rate`	Float: Learning rate for new boost object. Default = 1
`init`	Float: Initial value for new boost object. Default = 0
`apply.lr`	Logical: Only considered is `x = NULL`. If TRUE, new boost object's fitted values will be object$fitted * learning.rate, otherwise object$fitted
`tolerance`	Float: error tolerance for new boost object. See boost
`tolerance.valid`	Float: error tolerance for validation set. See boost

Author(s)

E.D. Gennatas

as.cartLinBoostTV *Place model in cartLinBoostTV structure*

Description

Place model in cartLinBoostTV structure

Usage

```
as.cartLinBoostTV(
  object,
  x,
  y = NULL,
  x.valid = NULL,
  y.valid = NULL,
  learning.rate = 1,
  init = 0,
  apply.lr = TRUE
)
```

Arguments

object	rtMod model
x	Data.frame, optional: if provided, use to calculate fitted values of new boost object
y	Float, vector: Outcome
x.valid	Data.frame; optional: Validation data
y.valid	Float, vector; optional: Validation outcome
learning.rate	Float: Learning rate for new boost object. Default = 1
init	Float: Initial value for new boost object. Default = 0
apply.lr	Logical: Only considered is x = NULL. If TRUE, new boost object's fitted values will be object$fitted * learning.rate, otherwise object$fitted

Author(s)

E.D. Gennatas

as.cartLiteBoostTV *Place model in cartLiteBoostTV structure*

Description

Place model in cartLiteBoostTV structure

Usage

```
as.cartLiteBoostTV(
  object,
  x,
  y = NULL,
  x.valid = NULL,
  y.valid = NULL,
  learning.rate = 1,
  init = 0,
  apply.lr = TRUE
)
```

Arguments

object	rtMod model
x	Data.frame, optional: if provided, use to calculate fitted values of new boost object
y	Float, vector: Outcome
x.valid	Data.frame; optional: Validation data
y.valid	Float, vector; optional: Validation outcome
learning.rate	Float: Learning rate for new boost object. Default = 1
init	Float: Initial value for new boost object. Default = 0
apply.lr	Logical: Only considered is x = NULL. If TRUE, new boost object's fitted values will be object$fitted * learning.rate, otherwise object$fitted

Author(s)

E.D. Gennatas

---

as.data.tree.rpart *Convert* rpart *rules to* data.tree *object*

Description

Convert an rpart object to a data.tree object, which can be plotted with dplot3_cart

Usage

```
as.data.tree.rpart(object, verbose = FALSE)
```

Arguments

object	rpart object
verbose	Logical: If TRUE, print messages to console

Value

data.tree object

Author(s)

E.D. Gennatas

`as.data.tree.shyoptleaves` *Convert shyoptleaves to* `data.tree` *object*

Description

Convert shyoptleaves to `data.tree` object

Usage

```
as.data.tree.shyoptleaves(object)
```

Arguments

`object`	shyoptleaves object

`as.data.tree.shytreegamleaves` *Convert shytreegamleaves to* `data.tree` *object*

Description

Convert shytreegamleaves to `data.tree` object

Usage

```
as.data.tree.shytreegamleaves(object)
```

Arguments

`object`	shytreegamleaves object

`as.data.tree.shytreeLeavesRC` *Convert shytreeLeavesRC to* `data.tree` *object*

Description

Convert shytreeLeavesRC to `data.tree` object

Usage

```
as.data.tree.shytreeLeavesRC(object)
```

Arguments

`object`	shytreeLeavesRC object

as.glmLiteBoostTV *Place model in glmLiteBoostTV structure*

Description

Place model in glmLiteBoostTV structure

Usage

```
as.glmLiteBoostTV(
  object,
  x,
  y = NULL,
  x.valid = NULL,
  y.valid = NULL,
  learning.rate = 1,
  init = 0,
  apply.lr = TRUE
)
```

Arguments

object	rtMod model
x	Data.frame, optional: if provided, use to calculate fitted values of new boost object
y	Float, vector: Outcome
x.valid	Data.frame; optional: Validation data
y.valid	Float, vector; optional: Validation outcome
learning.rate	Float: Learning rate for new boost object. Default = 1
init	Float: Initial value for new boost object. Default = 0
apply.lr	Logical: Only considered is x = NULL. If TRUE, new boost object's fitted values will be object$fitted * learning.rate, otherwise object$fitted

Author(s)

E.D. Gennatas

auc *Area under the ROC Curve*

Description

Get the Area under the ROC curve to assess classifier performance using pROC

Usage

```
auc(prob, labels, method = c("auc_pairs", "pROC", "ROCR"), verbose = FALSE)
```

Arguments

prob	Numeric, Vector: Probabilities or model scores (e.g. c(.32, .75, .63), etc)
labels	True labels of outcomes (e.g. c(0, 1, 1))
method	Character: "pROC", "auc_pairs", or "ROCR": Method to use. Will use pROC::roc, auc_pairs, ROCR::performance, respectively. They should all give the same result, they are included for testing.
verbose	Logical: If TRUE, print messages to output

Details

Consider looking at Balanced Accuracy and F1 as well

Important Note: We assume that true labels are a factor where the first level is the "positive" case, a.k.a. the event. All methods used here, "pROC", "auc_pairs", "ROCR", have been setup to expect this. This goes against the default sertting for both "pROC" and "ROCR", which will not give an AUC less than .5 because they will reorder levels. We don't want this because you can have a classifier perform worse than .5 and tt can be very confusing if levels are reordered automatically and different functions give you different AUC.

---

auc_pairs *Area under the Curve by pairwise concordance*

Description

Get the Area under the ROC curve to assess classifier performance using pairwise concordance

Usage

```
auc_pairs(estimated.score, true.labels, verbose = TRUE)
```

Arguments

estimated.score	Float, Vector: Probabilities or model scores (e.g. c(.32, .75, .63), etc)
true.labels	True labels of outcomes (e.g. c(0, 1, 1))
verbose	Logical: If TRUE, print messages to output

Details

The first level of `true.labels` must be the positive class, and high numbers in `estimated.score` should correspond to the positive class.

Examples

```
true.labels <- factor(c("a", "a", "a", "b", "b", "b", "b"))
estimated.score = c(0.7, 0.55, 0.45, 0.25, 0.6, 0.7, 0.2)
auc_pairs(estimated.score, true.labels, verbose = TRUE)
```

bacc *Balanced Accuracy*

Description

Balanced Accuracy of a binary classifier

Usage

```
bacc(true, predicted, harmonize = FALSE, verbose = TRUE)
```

Arguments

true	True labels
predicted	Estimated labels
harmonize	Logical: passed to sensitivity and specificity, which use factorHarmonize. Default = FALSE
verbose	Logical: If TRUE, print messages to output

Details

BAcc = .5 * (Sensitivity + Specificity)

bag *Bag an* **rtemis** *learner for regression or classification [C, R]*

Description

Train a bagged ensemble using any learner

Usage

```
bag(
  x,
  y = NULL,
  x.test = NULL,
  y.test = NULL,
  weights = NULL,
  mod = "cart",
  k = 10,
  mtry = NULL,
  mod.params = list(),
  ipw = TRUE,
  ipw.type = 2,
  upsample = FALSE,
  downsample = FALSE,
  resample.seed = NULL,
  .resample = rtset.resample(resampler = "strat.boot", n.resamples = k),
  aggr.fn = NULL,
```

```
  x.name = NULL,
  y.name = NULL,
  question = NULL,
  base.verbose = FALSE,
  verbose = TRUE,
  trace = 0,
  print.plot = TRUE,
  plot.fitted = NULL,
  plot.predicted = NULL,
  plot.theme = rtTheme,
  print.base.plot = FALSE,
  n.workers = rtCores,
  parallel.type = ifelse(.Platform$OS.type == "unix", "fork", "psock"),
  outdir = NULL,
  ...
)
```

Arguments

x	Numeric vector or matrix / data frame of features i.e. independent variables
y	Numeric vector of outcome, i.e. dependent variable
x.test	Numeric vector or matrix / data frame of testing set features Columns must correspond to columns in x
y.test	Numeric vector of testing set outcome
weights	Numeric vector: Weights for cases. For classification, weights takes precedence over ipw, therefore set weights = NULL if using ipw. Note: If weight are provided, ipw is not used. Leave NULL if setting ipw = TRUE. Default = NULL
mod	Character: Algorithm to bag, for options, see modSelect
k	Integer: Number of base learners to train
mtry	Integer: Number of features to randomly sample for each base learner.
mod.params	Named list of arguments for mod
ipw	Logical: If TRUE, apply inverse probability weighting (for Classification only). Note: If weights are provided, ipw is not used. Default = TRUE
ipw.type	Integer 0, 1, 2 1: class.weights as in 0, divided by max(class.weights) 2: class.weights as in 0, divided by min(class.weights) Default = 2
upsample	Logical: If TRUE, upsample cases to balance outcome classes (for Classification only) Note: upsample will randomly sample with replacement if the length of the majority class is more than double the length of the class you are upsampling, thereby introducing randomness
downsample	Logical: If TRUE, downsample majority class to match size of minority class
resample.seed	Integer: If provided, will be used to set the seed during upsampling. Default = NULL (random seed)
.resample	List: Resample settings to use. There is no need to edit this, unless you want to change the type of resampling. It will use stratified bootstrap by default. Use rtset.resample for convenience. Default = rtset.resample(resampler = "strat.boot", n.resamples = k)
aggr.fn	Function: used to average base learners' predictions. Default = mean for Classification, median for Regression

x.name	Character: Name for feature set
y.name	Character: Name for outcome
question	Character: the question you are attempting to answer with this model, in plain language.
base.verbose	Logical: verbose argument passed to learner
verbose	Logical: If TRUE, print summary to screen.
trace	Integer: If > 0, print diagnostic info to console
print.plot	Logical: if TRUE, produce plot using mplot3 Takes precedence over plot.fitted and plot.predicted. Default = TRUE
plot.fitted	Logical: if TRUE, plot True (y) vs Fitted
plot.predicted	Logical: if TRUE, plot True (y.test) vs Predicted. Requires x.test and y.test
plot.theme	Character: "zero", "dark", "box", "darkbox"
print.base.plot	Logical: Passed to print.plot argument of base learner, i.e. if TRUE, print error plot for each base learner
n.workers	Integer: Number of cores to use
parallel.type	Character: "fork" or "psock". Type of parallelization. Default = "fork" for macOS and Linux, "psock" for Windows
outdir	Character: Path to output directory to save model. Default = NULL
...	Additional parameters to be passed to learner

Author(s)

E.D. Gennatas

Examples

```
## Not run:
# Data ----
set.seed(2018)
x <- rnormmat(500, 50)
colnames(x) <- paste0("Feature", 1:50)
w <- rnorm(50)
y <- .7 * x[, 3]^2 + 1.2 * x[, 10] + .5 * x[, 15] + .8 * x[, 20] + rnorm(500)
dat <- data.frame(x, y)
res <- resample(dat, seed = 2018)
dat_train <- dat[res$Subsample_1, ]
dat_test <- dat[-res$Subsample_1, ]

# bag ----
mod <- bag(dat_train, dat_test)

## End(Not run)
```

betas.lihad *Extract coefficients from Additive Tree leaves*

Description

Extract coefficients from Additive Tree leaves

Usage

```
betas.lihad(object, newdata, verbose = FALSE, trace = 0)
```

Arguments

object	lihad object
newdata	matrix/data.frame of features
verbose	Logical: If TRUE, print output to console
trace	Integer 0:2 Increase verbosity

Author(s)

E.D. Gennatas

bias_variance *Bias-Variance Decomposition*

Description

Bias-Variance Decomposition

Usage

```
bias_variance(
  x,
  y,
  mod,
  res1_train.p = 0.7,
  params = list(),
  resample.rtset = rtset.resample(n.resamples = 100),
  seed = NULL,
  verbose = TRUE,
  res.verbose = FALSE,
  ...
)
```

Arguments

x	Predictors
y	Outcome
mod	Character: rtemis learner
res1_train.p	Numeric: Proportion of cases to use for training
params	List of mod parameters
resample.rtset	Output of rtset.resample
verbose	Logical: If TRUE, print messages to console
...	Additional arguments passed to resLearn_future

Author(s)

E.D. Gennatas

---

binmat2vec *Binary matrix times character vector*

Description

Binary matrix times character vector

Usage

```
binmat2vec(x, labels = colnames(x))
```

Arguments

x	A binary matrix or data.frame
labels	Character vector length equal to ncol(x)

Value

a character vector

boost *Boost an* **rtemis** *learner for regression*

Description

Train an ensemble using boosting of any learner

Usage

```
boost(
  x,
  y = NULL,
  x.valid = NULL,
  y.valid = NULL,
  x.test = NULL,
  y.test = NULL,
  mod = "cart",
  resid = NULL,
  boost.obj = NULL,
  mod.params = list(),
  case.p = 1,
  weights = NULL,
  learning.rate = 0.1,
 earlystop.params = rtset.earlystop(window = 30, window_decrease_pct_min = 0.01),
  earlystop.using = "train",
  tolerance = 0,
  tolerance.valid = 1e-05,
  max.iter = 10,
  init = NULL,
  x.name = NULL,
  y.name = NULL,
  question = NULL,
  base.verbose = FALSE,
  verbose = TRUE,
  trace = 0,
  print.progress.every = 5,
  print.error.plot = "final",
  prefix = NULL,
  plot.theme = rtTheme,
  plot.fitted = NULL,
  plot.predicted = NULL,
  print.plot = FALSE,
  print.base.plot = FALSE,
  plot.type = "l",
  outdir = NULL,
  ...
)
```

Arguments

x	Numeric vector or matrix / data frame of features i.e. independent variables

y	Numeric vector of outcome, i.e. dependent variable
x.valid	Data.frame; optional: Validation data
y.valid	Float, vector; optional: Validation outcome
x.test	Numeric vector or matrix / data frame of testing set features Columns must correspond to columns in x
y.test	Numeric vector of testing set outcome
mod	Character: Algorithm to train base learners, for options, see modSelect. Default = "cart"
resid	Float, vector, length = length(y): Residuals to work on. Do not change unless you know what you're doing. Default = NULL, for regular boosting
boost.obj	[Internal use]
mod.params	Named list of arguments for mod
case.p	Float (0, 1]: Train each iteration using this perceent of cases. Default = 1, i.e. use all cases
weights	Numeric vector: Weights for cases. For classification, weights takes precedence over ipw, therefore set weights = NULL if using ipw. Note: If weight are provided, ipw is not used. Leave NULL if setting ipw = TRUE. Default = NULL
learning.rate	Float (0, 1] Learning rate for the additive steps
earlystop.params	List with early stopping parameters. Set using rtset.earlystop
earlystop.using	Character: "train" or "valid". For the latter, requires x.valid
tolerance	Float: If training error <= this value, training stops
tolerance.valid	Float: If validation error <= this value, training stops
max.iter	Integer: Maximum number of iterations (additive steps) to perform. Default = 10
init	Float: Initial value for prediction. Default = mean(y)
x.name	Character: Name for feature set
y.name	Character: Name for outcome
question	Character: the question you are attempting to answer with this model, in plain language.
base.verbose	Logical: verbose argument passed to learner
verbose	Logical: If TRUE, print summary to screen.
trace	Integer: If > 0, print diagnostic info to console
print.progress.every	Integer: Print progress over this many iterations
print.error.plot	String or Integer: "final" plots a training and validation (if available) error curve at the end of training. If integer, plot training and validation error curve every this many iterations during training. "none" for no plot.
prefix	Internal
plot.theme	Character: "zero", "dark", "box", "darkbox"
plot.fitted	Logical: if TRUE, plot True (y) vs Fitted

plot.predicted	Logical: if TRUE, plot True (y.test) vs Predicted. Requires x.test and y.test
print.plot	Logical: if TRUE, produce plot using mplot3 Takes precedence over plot.fitted and plot.predicted. Default = TRUE
print.base.plot	Logical: Passed to print.plot argument of base learner, i.e. if TRUE, print error plot for each base learner
plot.type	Character: "l" or "p". Plot using lines or points.
outdir	Path to output directory. If defined, will save Predicted vs. True plot, if available, as well as full model output, if save.mod is TRUE
...	Additional parameters to be passed to learner define by mod

Details

If learning.rate is set to 0, a nullmod will be created

Author(s)

E.D. Gennatas

---

bootstrap *Bootstrap Resampling*

Description

Bootstrap Resampling

Usage

```
bootstrap(x, n.resamples = 10, seed = NULL)
```

Arguments

x	Input vector
n.resamples	Integer: Number of resamples to make. Default = 10
seed	Integer: If provided, set seed for reproducibility. Default = NULL

Author(s)

E.D. Gennatas

catrange *Print range of continuous variable*

Description

Print range of continuous variable

Usage

```
catrange(x, ddSci = TRUE, decimal.places = 1, na.rm = TRUE)
```

Arguments

x	Numeric vector
ddSci	Logical: If TRUE, use ddSci or range. Default = TRUE
decimal.places	Integer: Number of decimal place to use if ddSci = TRUE. Default = 1
na.rm	Logical: passed to base::range

Author(s)

E.D. Gennatas

catsize *Size*

Description

Get NCOL(x) and NROW{x}

Usage

```
catsize(x, verbose = TRUE)
```

Arguments

x	R object (usually that inherits from matrix or data.frame)
verbose	Logical: If TRUE, print NROW and NCOL to console. Default = TRUE

Value

vector of NROW, NCOL invisibly

Author(s)

E.D. Gennatas

cc *Concatenate Vectors*

Description

Concatenate that maintains factors

Usage

```
cc(...)
```

Arguments

...	Two or more vectors of any type

Details

A `c()` replacement that maintains factors as factors because it doesn't make sense not to. If all inputs are factors, they are converted to character, concatenated and converted to factor again. Otherwise, they are passed to `c()`

Value

Concatenation of ...

Author(s)

E.D. Gennatas

checkData *Check Data*

Description

Runs a series of simple checks on a dataset that may be important to perform ahead of data analysis. This function makes no changes to data, but reports potential course of action that can be taken using preprocess

Usage

```
checkData(
  x,
  name = NULL,
  str = FALSE,
  recommend = TRUE,
  reportCases.thres = NULL,
  reportFeatures.thres = NULL,
  toHTML = FALSE,
  verbose = TRUE
)
```

Arguments

x	Input dataset: will be converted to data.frame
name	String (optional): Name of dataset. (This is helpful when applying preprocess on a list of items by vectorization, e.g. using *ply commands, where the names of the list elements will not be displayed correctly)
str	Logical: If TRUE, show output of str. Default = FALSE
recommend	Logical: If TRUE, print recommendations based on check. Default = TRUE
reportCases.thres	Float (0, 1]: Report, by number, all cases missing greater or equal to this fraction of features. Default = NULL
reportFeatures.thres	Float (0, 1]: Report, by name, all features missing in greater or equal to this fraction of cases. Default = NULL
toHTML	Logical: If TRUE, convert output to HTML using the fansi package. Default = FALSE
verbose	Logical: If TRUE, print messages to console

Author(s)

E.D. Gennatas

Examples

```
checkData(iris)
```

---

checkData_live *Check Data (HTML output)*

Description

Check Data (HTML output)

Usage

```
checkData_live(
  x,
  name = NULL,
  recommend = TRUE,
  font.family = "Fira Sans",
  color = "#e7e7e7",
  background.color = "#121212",
  class = "checkData",
  verbose = TRUE
)
```

Arguments

x	data.frame, data.table or similar structure
name	Character: Dataset name
recommend	Logical: If TRUE, output recommendations at the end. Default = TRUE
font.family	Character: Font family to use. Default = "Fira Sans"
color	Character: Text color. Default = "#e7e7e7"
background.color	Character: Background color. Default = "transparent"
class	Character: CSS class to assign to div output. Default = "checkData"
verbose	Logical: If TRUE, print output in HTML viewer. Default = TRUE

Author(s)

E.D. Gennatas

Examples

```
## Not run:
checkData_live(iris)

## End(Not run)
```

---

checkpoint_earlystop *Early stopping check*

Description

Returns list with relative variance over n.steps, absolute.threshold, last value, and logical "stop", if conditions are met and training should stop. The final stop decision is: `check.thresh | (check.rthresh & check.rvar)` if `combine.relative.thresholds = "AND"` or `check.thresh | (check.rthresh | check.rvar)` if `combine.relative.thresholds = "OR"`

Usage

```
checkpoint_earlystop(
  x,
  absolute.threshold = NA,
  relative.threshold = NA,
  minimize = TRUE,
  relativeVariance.threshold = NA,
  n.steps = 10,
  combine.relative.thresholds = "AND",
  min.steps = 50,
  na.response = c("stop", "continue"),
  verbose = TRUE
)
```

Arguments

x	Float, vector: Input - this would normally be the loss at each iteration
absolute.threshold	Float: If set and the last value of x is less than or equal to this (if minimize = TRUE) or greater than or equal to this (if minimize = FALSE), then return stop = TRUE. See output under Value. Default = NA
relative.threshold	Float: If set, checks if the relative change from the first to last value of x exceeds this number. i.e. if set to .9 and minimize = TRUE, if there is a 90% drop from x[1] to x[length(x)], then the function returns stop = TRUE. If minimize = FALSE, then checks if there is a 90% increase, accordingly.
minimize	Logical: See absolute.threshold. Default = TRUE
relativeVariance.threshold	Float: If relative variance over last n.steps is less than or equal to this, return stop = TRUE. See output under Value
n.steps	Integer; > 1: Calculate relative variance over this many last values of x
combine.relative.thresholds	Character: "AND" or "OR": How to combine the criteria relative.threshold and relativeVariance.threshold. Default = "AND", which means both must be TRUE to stop. The scenario is you might want to check relastiveVariance threshold only after a certain amount of learning has taken place, which you can't predict with min.steps but would rather quantify with relative.threshold.
min.steps	Integer: Do not calculate relativeVariance unless x is at least this length
na.response	Character: "stop" or "continue": what should happen if the last value of x is NA
verbose	Logical: If TRUE, print messages to output

Value

List with the following items:

last.value	Float: Last value of x
relativeVariance	Float: relative variance of last n.steps
check.thresh	Logical: TRUE, if absolute threshold was reached
check.rvar	Logical: TRUE, if relative variance threshold was reached
stop	Logical: TRUE, if either criterion was met - absolute threshold or relativeVariance.threshold

Author(s)

E.D. Gennatas

chill *Chill*

Description

Relax. Use Ctrl-C to exit (but try to stay relaxed)

Usage

```
chill(sleep = 0.5, text = NULL, max = 1000)
```

Arguments

sleep	Float: Time in seconds between drawings. Default = .5
text	Character: Text to display
max	Integer: Max times to repeat. Default = 1000

classError *Classification Error*

Description

Calculates Classification Metrics

Usage

```
classError(true, estimated, estimated.prob = NULL, trace = 0)
```

Arguments

true	Factor vector: True values
estimated	Factor vector: Estimated values
estimated.prob	Numeric vector: Estimated probabilities
trace	Integer: If > 0, print diagnostic messages. Default = 0

Value

S3 object of type "classError"

Author(s)

E.D. Gennatas

classImbalance *Class Imbalance*

Description

Calculate class imbalance as given by:

$$I = K \cdot \sum_{i=1}^{K} (n_i/N - 1/K)^2$$

where K is the number of classes, and n_i is the number of instances of class i

Usage

```
classImbalance(x)
```

Arguments

x	Vector, factor: Labels of outcome. If x has more than 1 column, the last one will be used

Author(s)

E.D. Gennatas

clean_colnames *Clean column names*

Description

Clean column names by replacing all spaces and punctuation with a single underscore

Usage

```
clean_colnames(x)
```

Arguments

x	Character, vector

Author(s)

E.D. Gennatas

clust *Clustering with* **rtemis**

Description

Convenience function to perform any **rtemis** clustering

Usage

```
clust(x, clust = "kmeans", x.test = NULL, verbose = TRUE, ...)
```

Arguments

x	Numeric matrix / data frame: Input data
clust	Character: Decomposition algorithm name, e.g. "nmf" (case-insensitive)
x.test	Numeric matrix / Data frame: Testing set data if supported by clust
verbose	Logical: if TRUE, print messages to screen
...	Additional arguments to be passed to clusterer clust

Value

rtClust object

Author(s)

E.D. Gennatas

clustSelect *Select* **rtemis** *Clusterer*

Description

Accepts clusterer name (supports abbreviations) and returns **rtemis** function name or the function itself. If run with no parameters, prints list of available algorithms.

Usage

```
clustSelect(clust, fn = FALSE, desc = FALSE)
```

Arguments

clust	Character: Clustering algorithm name. Case insensitive, supports partial matching. e.g. "hop" for HOPACH
fn	Logical: If TRUE, return function, otherwise name of function. Defaults to FALSE
desc	Logical: If TRUE, return full name of algorithm clust

Value

Name of function (Default) or function (`fn=TRUE`) or full name of algorithm (`desc=TRUE`)

Author(s)

E.D. Gennatas

`coef.lihad` *Extract coefficients from Hybrid Additive Tree leaves*

Description

Extract coefficients from Hybrid Additive Tree leaves

Usage

```
## S3 method for class 'lihad'
coef(object, newdata, verbose = FALSE, trace = 0, ...)
```

Arguments

`object`	`lihad` object
`newdata`	matrix/data.frame of features
`verbose`	Logical: If TRUE, print output to console
`trace`	Integer 0:2 Increase verbosity
`...`	Not used

Author(s)

E.D. Gennatas

`col2grayscale` *Color to Grayscale*

Description

Convert a color to grayscale

Usage

```
col2grayscale(x, what = c("color", "decimal"))
```

Arguments

`x`	Color to convert to grayscale
`what`	Character: "color" returns a hexadecimal color, "decimal" returns a decimal between 0 and 1

Details

Uses the NTSC grayscale conversion: 0.299 * R + 0.587 * G + 0.114 * B

Examples

```
col2grayscale("red")
col2grayscale("red", "dec")
```

---

col2hex *Convert R color to hexadecimal code*

Description

Convert a color that R understands into the corresponding hexadecimal code

Usage

```
col2hex(color)
```

Arguments

color	Color(s) that R understands

Author(s)

E.D. Gennatas

Examples

```
col2hex(c("gray50", "skyblue"))
```

---

colMax *Collapse data.frame to vector by getting column max*

Description

Collapse data.frame to vector by getting column max

Usage

```
colMax(x, na.rm = TRUE)
```

Arguments

x	Matrix or Data frame input
na.rm	Logical: passed to max, If TRUE, ignore NA values, otherwise if NA is present in any column, NA will be returned. Default = TRUE

Author(s)

E.D. Gennatas

colorAdjust *Adjust HSV Color*

Description

Modify alpha, hue, saturation and value (HSV) of a color

Usage

```
colorAdjust(color, alpha = NULL, hue = 0, sat = 0, val = 0)
```

Arguments

color	Input color. Any format that grDevices::col2rgb() recognizes
alpha	Numeric: Scale alpha by this amount. Future: replace with absolute setting
hue	Float: How much hue to add to color
sat	Float: How much saturation to add to color
val	Float: How much to increase value of color by

Value

Adjusted color

Author(s)

E.D. Gennatas

colorGrad *Color Gradient*

Description

Create a gradient of colors and optionally a colorbar

Usage

```
colorGrad(
  n = 21,
  colors = NULL,
  space = c("rgb", "Lab"),
  lo = "#18A3AC",
  lomid = NULL,
  mid = NULL,
  midhi = NULL,
  hi = "#F48024",
  preview = FALSE,
  colorbar = FALSE,
  cb.n = 21,
  cb.mar = c(1, 1, 1, 1),
```

```
  cb.add = FALSE,
  cb.add.mar = c(5, 0, 2, 5),
  cb.axis.pos = 1.1,
  cb.axis.las = 1,
  cb.axis.hadj = 0,
  cb.cex = 6,
  bar.min = -1,
  bar.mid = 0,
  bar.max = 1,
  cex = 1.2,
  filename = NULL,
  pdf.width = 3,
  pdf.height = 7,
  theme = getOption("rt.theme", "light"),
  bg = NULL,
  col.text = NULL,
  plotlycb = FALSE,
  plotly.width = 80,
  plotly.height = 500,
  rtrn.plotly = FALSE,
  margins = c(0, 0, 0, 0),
  pad = 0,
  par.reset = TRUE
)
```

Arguments

n	Integer: How many distinct colors you want. If not odd, converted to n + 1 Defaults to 21
colors	Character: Acts as a shortcut to defining lo, mid, etc for a number of defaults: "french", "penn", "grnblkred",
space	Character: Which colorspace to use. Option: "rgb", or "Lab". Default = "rgb". Recommendation: If mid is "white" or "black" (default), use "rgb", otherwise "Lab"
lo	Color for low end
lomid	Color for low-mid
mid	Color for middle of the range or "mean", which will result in colorOp(c(lo, hi), "mean"). If mid = NA, then only lo and hi are used to create the color gradient.
midhi	Color for middle-high
hi	Color for high end
preview	Logical: Plot the colors horizontally
colorbar	Logical: Create a vertical colorbar
cb.n	Integer: How many steps you would like in the colorbar
cb.mar	Vector, length 4: Colorbar margins. Default: c(1, 1, 1, 1)
cb.add	Logical: If TRUE, colorbar will be added to existing plot
cb.add.mar	Vector: Margins for colorbar (See par("mar"))
cb.axis.pos	Float: Position of axis (See axis("pos"))

cb.axis.las	Integer 0,1,2,3: Style of axis labels. 0: Always parallel to the axis, 1: Horizontal, 2: Perpendicular, 3: Vertical. Default = 1
cb.axis.hadj	Float: Adjustment parallel to the reading direction (See par("adj"))
cb.cex	FLoat: Character expansion factor for colorbar (See par("cex"))
bar.min	Numeric: Lowest value in colorbar
bar.mid	Numeric: Middle value in colorbar
bar.max	Numeric: Max value in colorbar
cex	Float: Character expansion for axis
filename	String (Optional: Path to file to save colorbar
pdf.width	Float: Width for PDF output. Default = 3
pdf.height	Float: Height for PDF output. Default = 7
theme	Character: "light", "dark"
bg	Color: Background color
col.text	Color: Colorbar text color
plotlycb	Logical: Create colorbar using plotly (instead of base R graphics)
plotly.width	Float: Width for plotly colorbar. Default = 80
plotly.height	Float: Height for plotly colorbar. Default = 500
rtrn.plotly	Logical: If TRUE, return plotly object
margins	Vector: Plotly margins. Default = c(0, 0, 0, 0)
pad	Float: Padding for plotly. Default = 0
par.reset	Logical: If TRUE (Default), reset par settings after running

Details

It is best to provide an odd number, so that there is always an equal number of colors on either side of the midpoint. For example, if you want a gradient from -1 to 1 or equivalent, an n = 11, will give 5 colors on either side of 0, each representing a 20% change from the next.

colors can be defined as a sequence of 3-letter color abbreviations of 2, 3, 4, or 5 colors which will correspond to values: {"lo","hi"}; {"lo", "mid", "hi"}; {"lo", "mid", "midhi", "hi"}, and {"lo", "lomid", "mid", "midhi", "hi"}, respectively. For example, try colorGrad(21, "blugrnblkredyel", colorbar = TRUE) 3-letter color abbreviations: wht: white; blk: black; red; grn: green; blu: blue; yel: yellow; rng: orange; prl: purple

Value

Invisible vector of hexadecimal colors / plotly object if rtrn.plotly = TRUE

Author(s)

E.D. Gennatas

colorGrad.x *Color gradient for continuous variable*

Description

Color gradient for continuous variable

Usage

```
colorGrad.x(x, color = c("gray20", "#18A3AC"), space = "Lab")
```

Arguments

x	Float, vector
color	Color, vector, length 2
space	Character: "rgb" or "Lab", Default = "Lab"

Author(s)

E.D. Gennatas

colorgradient.x *Color gradient for continuous variable*

Description

Color gradient for continuous variable

Usage

```
colorgradient.x(
  x,
  symmetric = FALSE,
  lo.col = "#0290EE",
  mid.col = "#1A1A1A",
  hi.col = "#FFBD4F",
  space = "Lab"
)
```

Arguments

x	Float, vector
symmetric	Logical: If TRUE, make symmetric gradient between -max(abs(x)) and max(abs(x))
lo.col	Low color
mid.col	Middle color
hi.col	High color
space	Character: "rgb" or "Lab". Default = "Lab"

Author(s)

E.D. Gennatas

Examples

```
## Not run:
x <- seq(-10, 10, length.out = 51)
previewcolor(colorgradient.x(x))
x <- sort(rnorm(40))
previewcolor(colorgradient.x(x, mid.col = "white"))
# Notice how most values are near zero therefore almost white

## End(Not run)
```

---

colorMix *Create an alternating sequence of graded colors*

Description

Create an alternating sequence of graded colors

Usage

```
colorMix(color, n = 4)
```

Arguments

color	List: List of two or more elements, each containing two colors. A gradient will be created from the first to the second color of each element
n	Integer: Number of steps in each gradient.

Author(s)

E.D. Gennatas

Examples

```
color <- list(blue = c("#82afd3", "#000f3a"),
              gray = c("gray10", "gray85"))
previewcolor(desaturate(colorMix(color, 6), .3))

color <- list(blue = c("#82afd3", "#57000a"),
              gray = c("gray10", "gray85"))
previewcolor(desaturate(colorMix(color, 6), .3))

color <- list(blue = c("#82afd3", "#000f3a"),
              purple = c("#23001f", "#c480c1"))
previewcolor(desaturate(colorMix(color, 5), .3))
```

colorOp *Simple Color Operations*

Description

Invert a color or calculate the mean of two colors in HSV or RGB space. This may be useful in creating colors for plots

Usage

```
colorOp(col, fn = c("invert", "mean"), space = c("HSV", "RGB"))
```

Arguments

col	Input color(s)
fn	Character: "invert", "mean": Function to perform
space	Character: "HSV", "RGB": Colorspace to operate in - for averaging only

Details

The average of two colors in RGB space will often pass through gray, which is likely undesirable. Averaging in HSV space, better for most applications.

Value

Color

Author(s)

E.D. Gennatas

cols2list *Convert data frame columns to list elements*

Description

Convenience function to create a list out of data frame columns

Usage

```
cols2list(x)
```

Arguments

x	Input: Will be coerced to data.frame, then each column will become an element of a list

Author(s)

E.D. Gennatas

crules *Combine rules*

Description

Combine rules

Usage

```
crules(...)
```

Arguments

...	Character: Rules

Author(s)

E.D. Gennatas

cube *Cube*

Description

Cube

Usage

```
cube(x)
```

Arguments

x	Vector, Float: Input

cutmidpoint *Get midpoint of 'cut' label*

Description

Get midpoint of 'cut' label

Usage

```
cutmidpoint(x)
```

c_CMEANS *Fuzzy C-means Clustering*

Description

Perform fuzzy C-means clustering using `e1071::cmeans`

Usage

```
c_CMEANS(
  x,
  k = 2,
  iter.max = 100,
  dist = "euclidean",
  method = "cmeans",
  m = 2,
  rate.par = NULL,
  weights = 1,
  control = list(),
  verbose = TRUE,
  ...
)
```

Arguments

`x`	Input matrix / data.frame
`k`	Integer: Number of clusters to get. Default = 2
`iter.max`	Integer: Maximum number of iterations. Default = 100
`dist`	Character: Distance measure to use: 'euclidean' or 'manhattan'. Default = "euclidean"
`method`	Character: "cmeans" - fuzzy c-means clustering; "ufcl": on-line update. Default = "cmeans"
`m`	Float (>1): Degree of fuzzification. Default = 2
`rate.par`	Float (0, 1): Learning rate for the online variant. (Default = .3)
`weights`	Float (>0): Case weights
`control`	List of control parameters. See `e1071::cmeans`
`verbose`	Logical: If TRUE, print messages to screen
`...`	Additional parameters to be passed to `e1071::cmeans`

Value

rtClust object

Author(s)

E.D. Gennatas

See Also

Other Clustering: c_DBSCAN(), c_EMC(), c_H2OKMEANS(), c_HARDCL(), c_HOPACH(), c_KMEANS(), c_MEANSHIFT(), c_NGAS(), c_PAMK(), c_PAM(), c_SPEC()

---

c_DBSCAN *Density-based spatial clustering of applications with noise*

Description

Perform DBSCAN clustering

Usage

```
c_DBSCAN(
  x,
  x.test = NULL,
  eps = 1,
  minPts = NCOL(x) + 1,
  weights = NULL,
  borderPoints = TRUE,
  search = c("kdtree", "linear", "dist"),
  verbose = TRUE,
  ...
)
```

Arguments

x	Input matrix / data.frame
x.test	Testing set matrix / data.frame
eps	Numeric: Radius of the epsilon neighborhood
minPts	Integer: Number of minimum points required in the eps neighborhood for core points (including the point itself).
weights	Numeric vector: Data points' weights. Needed for weighted clustering.
borderPoints	Logical: If TRUE, assign border points to clusters, otherwise they are considered noise
search	Character: "kdtree", "linear" or "dist": nearest neighbor search strategy
verbose	Logical: If TRUE, print messages to screen
...	Additional parameters to be passed to flexclust::cclust

Details

See dbscan::dbscan for info on how to choose eps and minPts

Author(s)

Efstathios D. Gennatas

See Also

Other Clustering: c_CMEANS(), c_EMC(), c_H2OKMEANS(), c_HARDCL(), c_HOPACH(), c_KMEANS(), c_MEANSHIFT(), c_NGAS(), c_PAMK(), c_PAM(), c_SPEC()

c_EMC *Expectation Maximization Clustering*

Description

Perform EM Clustering using `EMCluster::emcluster`

Usage

```
c_EMC(
  x,
  x.test = NULL,
  k = 2,
  lab = NULL,
  EMC = EMCluster::.EMC,
  verbose = TRUE,
  ...
)
```

Arguments

`x`	Input matrix / data.frame
`x.test`	Testing set matrix / data.frame
`k`	Integer: Number of clusters to get
`lab`	Vector, length `NROW(x)`: Labels for semi-supervised clustering
`EMC`	List of control parameters for `EMCluster::emcluster`. Default = `EMCluster::.EMC`
`verbose`	Logical: If TRUE, print messages to screen
`...`	Additional parameters to be passed to `EMCluster::emcluster`

Details

First, `EMCluster::simple.init(x, nclass = k)` is run, followed by `EMCluster::emcluster(x, emobj = emobj, assign.class = TRUE, ...)`

This can be very slow.

Author(s)

E.D. Gennatas

See Also

Other Clustering: `c_CMEANS()`, `c_DBSCAN()`, `c_H2OKMEANS()`, `c_HARDCL()`, `c_HOPACH()`, `c_KMEANS()`, `c_MEANSHIFT()`, `c_NGAS()`, `c_PAMK()`, `c_PAM()`, `c_SPEC()`

c_H2OKMEANS *K-Means Clustering on H2O*

Description

K-Means clustering using h2o::h2o.kmeans Check out the H2O Flow at [ip]:[port], Default IP:port is "localhost:54321" e.g. if running on localhost, point your web browser to localhost:54321

Usage

```
c_H2OKMEANS(
  x,
  x.test = NULL,
  k = 2,
  estimate.k = FALSE,
  nfolds = 0,
  max.iterations = 10,
  ip = "localhost",
  port = 54321,
  n.cores = rtCores,
  seed = -1,
  init = c("Furthest", "Random", "PlusPlus", "User"),
  categorical.encoding = c("AUTO", "Enum", "OneHotInternal", "OneHotExplicit",
    "Binary", "Eigen", "LabelEncoder", "SortByResponse", "EnumLimited"),
  verbose = TRUE,
  ...
)
```

Arguments

x	Input matrix / data.frame
x.test	Testing set matrix / data.frame
k	Integer: Number of clusters to get
estimate.k	Logical: if TRUE, estimate k up to a maximum set by the k argument
nfolds	Integer: Number of cross-validation folds
max.iterations	Integer: Maximum number of iterations
ip	Character: IP address of H2O server. Default = "localhost"
port	Integer: Port number of H2O server. Default = 54321
n.cores	Integer: Number of cores to use
seed	Integer: Seed for H2O's random number generator. Default = -1 (time-based ranodm number)
init	Character: Initialization mode: "Furthest", "Random", "PlusPlus", "User". Default = "Furthest"
categorical.encoding	Character: How to encode categorical variables: "AUTO", "Enum", "OneHotInternal", "OneHotExplicit", "Binary", "Eigen", "LabelEncoder", "SortByResponse", "EnumLimited". Default = "AUTO"
verbose	Logical: If TRUE, print messages to screen
...	Additional arguments to pass to h2p::h2o.kmeans

Details

For additional information, see help on `h2o::h2o.kmeans`

Value

rtMod object

Author(s)

E.D. Gennatas

See Also

Other Clustering: `c_CMEANS()`, `c_DBSCAN()`, `c_EMC()`, `c_HARDCL()`, `c_HOPACH()`, `c_KMEANS()`, `c_MEANSHIFT()`, `c_NGAS()`, `c_PAMK()`, `c_PAM()`, `c_SPEC()`

---

c_HARDCL *Clustering by Hard Competitive Learning*

Description

Perform clustering by Hard Competitive Learning using `flexclust::cclust`

Usage

```
c_HARDCL(x, x.test = NULL, k = 2, dist = "euclidean", verbose = TRUE, ...)
```

Arguments

`x`	Input matrix / data.frame
`x.test`	Testing set matrix / data.frame
`k`	Integer: Number of clusters to get
`dist`	Character: Distance measure to use: 'euclidean' or 'manhattan'
`verbose`	Logical: If TRUE, print messages to screen
`...`	Additional parameters to be passed to `flexclust::cclust`

Author(s)

E.D. Gennatas

See Also

Other Clustering: `c_CMEANS()`, `c_DBSCAN()`, `c_EMC()`, `c_H2OKMEANS()`, `c_HOPACH()`, `c_KMEANS()`, `c_MEANSHIFT()`, `c_NGAS()`, `c_PAMK()`, `c_PAM()`, `c_SPEC()`

c_HOPACH *Hierarchical Ordered Partitioning and Collapsing Hybrid*

Description

Perform HOPACH clustering using hopach::hopach

Usage

```
c_HOPACH(
  x,
  dmat = NULL,
  metric = c("cosangle", "abscosangle", "euclid", "abseuclid", "cor", "abscor"),
  k = 15,
  kmax = 9,
  khigh = 9,
  trace = 0,
  verbose = TRUE,
  ...
)
```

Arguments

x	Input matrix / data.frame
dmat	Matrix (numeric, no missing values) or hdist object of pairwise distances. If NULL, it is computed based on metric
metric	Character: Dissimilarity metric to be used. Options: "cosangle", "abscosangle", "euclid", "abseuclid", "cor", "abscor"
k	Integer, (0:15]: Maximum number of levels
kmax	Integer, [1:9]: Maximum number of children at each node in the tree
khigh	Integer, [1:9]: Maximum number of children at each nod ein the tree when computing the the Mean/Median Split Silhouette. Usually same as kmax
trace	Integer: If trace > 0, print messages during HOPACH run. Default = 0
verbose	Logical: If TRUE, print messages to screen
...	Additional parameters to be passed to cluster::hopach

Author(s)

E.D. Gennatas

See Also

Other Clustering: c_CMEANS(), c_DBSCAN(), c_EMC(), c_H2OKMEANS(), c_HARDCL(), c_KMEANS(), c_MEANSHIFT(), c_NGAS(), c_PAMK(), c_PAM(), c_SPEC()

c_KMEANS *K-means Clustering*

Description

Perform K-means clustering using `flexclust::cclust`

Usage

```
c_KMEANS(x, x.test = NULL, k = 2, dist = "euclidean", verbose = TRUE, ...)
```

Arguments

`x`	Input matrix / data.frame
`x.test`	Testing set matrix / data.frame
`k`	Integer: Number of clusters to get
`dist`	Character: Distance measure to use: 'euclidean' or 'manhattan'
`verbose`	Logical: If TRUE, print messages to screen
`...`	Additional parameters to be passed to `flexclust::cclust`

Author(s)

E.D. Gennatas

See Also

Other Clustering: `c_CMEANS()`, `c_DBSCAN()`, `c_EMC()`, `c_H2OKMEANS()`, `c_HARDCL()`, `c_HOPACH()`, `c_MEANSHIFT()`, `c_NGAS()`, `c_PAMK()`, `c_PAM()`, `c_SPEC()`

c_MEANSHIFT *Mean Shift Clustering*

Description

Perform Mean Shift clustering using `meanShiftR::meanShift`

Usage

```
c_MEANSHIFT(
  x,
  nNeighbors = NROW(x),
  algorithm = c("LINEAR", "KDTREE"),
  kernelType = c("NORMAL", "EPANECHNIKOV", "BIWEIGHT"),
  bandwidth = rep(1, NCOL(x)),
  alpha = 0,
  iterations = 10,
  epsilon = 1e-08,
  epsilonCluster = 1e-04,
```

```
  parameters = NULL,
  verbose = TRUE,
  ...
)
```

Arguments

x	Input matrix
nNeighbors	Integer: Number of neighbors to consider for kernel density estimate
algorithm	Character: "LINEAR" or "KDTREE"
kernelType	Character: "NORMAL", "EPANECHNIKOV", "BIWEIGHT"
bandwidth	Numeric vector, length = ncol(x): Use in kernel density estimation for steepest ascent classification.
alpha	Numeric: A scalar tuning parameter for normal kernels. When this parameter is set to zero, the mean shift algorithm will operate as usual. When this parameter is set to one, the mean shift algorithm will be approximated through Newton's Method. When set to a value between zero and one, a generalization of Newton's Method and mean shift will be used instead providing a means to balance convergence speed with stability.
iterations	Integer: Number of iterations to perform
epsilon	Numeric: used to determine when to terminate the iteration of an individual query point. If the distance between the query point at iteration i and i+1 is less than epsilon, then iteration ceases on this point.
epsilonCluster	Numeric: Used to determine the minimum distance between distinct clusters. This distance is applied after all iterations have finished and in order of the rows of queryData.
parameters	A scalar or vector of paramters used by the specific algorithm. There are no optional parameters for the "LINEAR" method, "KDTREE" supports optional parameters for the maximum number of points to store in a leaf node and the maximum value for the quadratic form in the normal kernel, ignoring the constant value -0.5.
verbose	Logical: If TRUE, print messages to console
...	Additional parameters to be passed to `flexclust::cclust`

Author(s)

E.D. Gennatas

See Also

Other Clustering: c_CMEANS(), c_DBSCAN(), c_EMC(), c_H2OKMEANS(), c_HARDCL(), c_HOPACH(), c_KMEANS(), c_NGAS(), c_PAMK(), c_PAM(), c_SPEC()

c_NGAS *Neural Gas Clustering*

Description

Perform Neural Gas clustering using `flexclust::cclust`

Usage

```
c_NGAS(x, x.test = NULL, k = 2, dist = "euclidean", verbose = TRUE, ...)
```

Arguments

`x`	Input matrix / data.frame
`x.test`	Testing set matrix / data.frame
`k`	Integer: Number of clusters to get
`dist`	Character: Distance measure to use: 'euclidean' or 'manhattan'
`verbose`	Logical: If TRUE, print messages to screen
`...`	Additional parameters to be passed to `flexclust::cclust`

Value

rtClust object

Author(s)

E.D. Gennatas

See Also

Other Clustering: `c_CMEANS()`, `c_DBSCAN()`, `c_EMC()`, `c_H2OKMEANS()`, `c_HARDCL()`, `c_HOPACH()`, `c_KMEANS()`, `c_MEANSHIFT()`, `c_PAMK()`, `c_PAM()`, `c_SPEC()`

c_PAM *Partitioning Around Medoids*

Description

Perform PAM clustering using `cluster::pam`

Usage

```
c_PAM(
  x,
  k = 2,
  diss = FALSE,
  metric = "euclidean",
  do.swap = TRUE,
  verbose = TRUE,
  ...
)
```

Arguments

x	Input matrix / data.frame
k	Integer: Number of clusters to get
diss	Logical: If TRUE, x should be a dist or dissimilarity matrix. Otherwise, x should be a matrix of cases by features. Default = FALSE
metric	Character: Dissimilarity metric to be used. Options: 'euclidean', 'manhattan'
do.swap	Logical: If TRUE, perform the swap phase (See cluster::pam), as in the original PAM algorithm. This is computationally intensive and can be skipped. Default = TRUE
verbose	Logical: If TRUE, print messages to screen
...	Additional parameters to be passed to cluster::pam

Author(s)

E.D. Gennatas

See Also

Other Clustering: c_CMEANS(), c_DBSCAN(), c_EMC(), c_H2OKMEANS(), c_HARDCL(), c_HOPACH(), c_KMEANS(), c_MEANSHIFT(), c_NGAS(), c_PAMK(), c_SPEC()

---

c_PAMK *Partitioning Around Medoids with k Estimation*

Description

Estimate PAM clustering solution and optimal k using fpc::pamk

Usage

```
c_PAMK(
  x,
  krange = 2:10,
  criterion = "asw",
  usepam = ifelse(nrow(x) < 2000, TRUE, FALSE),
  scaling = TRUE,
  diss = inherits(data, "dist"),
  metric = "euclidean",
  do.swap = TRUE,
  trace = 0,
  verbose = TRUE,
  ...
)
```

Arguments

x	Input matrix / data.frame
krange	Integer vector: Range of k values to try
criterion	Character: Criterion to use for selecting k: "asw", "multiasw" or "ch". See fpc::pamk
usepam	Logical: If TRUE, use cluster::pam, otherwise use cluster::clara. Default = TRUE
scaling	Logical or Numeric vector: If TRUE, scale input. If numeric vector of length equal to number of features, the features are divided by the corresponding value. Default = TRUE
diss	Logical: If TRUE, treat x as a dissimilarity matrix, otherwise as a matrix of cases by features. Default = TRUE, if x inherits from class dist, FALSE otherwise.
metric	Character: Dissimilarity metric to be used. Options: 'euclidean', 'manhattan'
do.swap	Logical: If TRUE, perform the swap phase (See cluster::pam), as in the original PAM algorithm. This is computationally intensive and can be skipped. Default = TRUE
trace	Integer [0, 3]: Trace level for fpc::pamk
verbose	Logical: If TRUE, print messages to screen
...	Additional parameters to be passed to fpc::pamk and/or cluster::pam

Value

rtClust object

Author(s)

E.D. Gennatas

See Also

Other Clustering: c_CMEANS(), c_DBSCAN(), c_EMC(), c_H2OKMEANS(), c_HARDCL(), c_HOPACH(), c_KMEANS(), c_MEANSHIFT(), c_NGAS(), c_PAM(), c_SPEC()

---

c_SPEC *Spectral Clustering*

Description

Perform Spectral Clustering using kernlab::specc

Usage

```
c_SPEC(
  x,
  k = 2,
  kernel = "rbfdot",
  kpar = "automatic",
  nystrom.red = FALSE,
```

```
  nystrom.sample = dim(x)[1]/6,
  iterations = 200,
  mod.sample = 0.75,
  na.action = na.omit,
  verbose = TRUE,
  ...
)
```

Arguments

x	Input matrix / data.frame
k	Integer: Number of clusters to get
kernel	Character: Kernel to use: "rbfdot", "polydot", "vanilladot", tanhdot", "laplacedot", "besseldot", "anovadot", "splinedot", "stringdot"
kpar	String OR List: "automatic", "local" OR list with: sigma (for "rbfdor", "laplacedot"); degree, scale, offset (for "polydot"); scale, offset (for "tanhdot"); sigma, order, degree (for "besseldot"); sigma, degree (for "anovadot"); length, lambda, normalized (for "stringdot")
nystrom.red	Logical: if TRUE, use nystrom method to calculate eigenvectors (Default = FALSE)
nystrom.sample	Integer: Number of points to use for estimating the eigenvalues when `nystrom.red = TRUE` Default = `dim(x)[1]/6`
iterations	Integer: Number of iterations allowed
mod.sample	Float (0, 1): Proportion of data to use when estimating sigma. Default = .75
na.action	Function: Action to perform on NA (Default = `na.omit`)
verbose	Logical: If TRUE, print messages to screen
...	Additional parameters to be passed to `flexclust::cclust`

Author(s)

E.D. Gennatas

See Also

Other Clustering: `c_CMEANS()`, `c_DBSCAN()`, `c_EMC()`, `c_H2OKMEANS()`, `c_HARDCL()`, `c_HOPACH()`, `c_KMEANS()`, `c_MEANSHIFT()`, `c_NGAS()`, `c_PAMK()`, `c_PAM()`

---

dat2bsplinemat *B-Spline matrix from dataset*

Description

Convert a dataset to its b-spline basis set

Usage

```
dat2bsplinemat(
  x,
  df = NULL,
  knots = NULL,
  degree = 3L,
  intercept = FALSE,
  Boundary.knots = range(x, na.rm = TRUE),
  return.deriv = FALSE,
  as.data.frame = TRUE
)
```

Arguments

x	data.frame: Input
df	Integer: Degrees of freedom. See splines::bSpline
knots	Float, vector: Internal breakpoints. See splines::bSpline
degree	Integer (>0): Degree of the piecewise polynomial. See splines::bSpline
intercept	Logical: If TRUE, an intercept is included. Default = FALSE
Boundary.knots	Float, vector (length = 2): Boundary points to anchor the spline basis.
return.deriv	Logical: If TRUE, return list containing a data frame with the splines and another data frame with their derivatives
as.data.frame	Logical: If TRUE, return data.frame, otherwise matrix. Default = TRUE See splines::bSpline

Value

If return.deriv=F, a data frame where each original feature is replaced with its basis set or a list, otherwise a list containing a data frame with splines and a data frame with their derivatives

Author(s)

E.D. Gennatas

---

dat2poly *Create n-degree polynomial from data frame*

Description

This is a convenience function that will take each column of the input and calculate 1:degree powers and concatenate into a data.frame of dimensions ‘n * (degree * p)‘ given an ‘n * p‘ input

Usage

```
dat2poly(
  dat,
  method = c("simple", "poly"),
  degree = 2,
  raw = FALSE,
  as.data.frame = TRUE
)
```

Arguments

dat	Numeric, matrix / data.frame: Input
method	Character: "simple", "poly". "simple": raise each column of dat up to degree. "poly": use stats::poly with arguments degree and raw
degree	Integer: degree of polynomials to create. Default = 2
raw	Logical: If TRUE, create simple polynomial, not orthogonalized. Default = FALSE
as.data.frame	Logical: If TRUE, return data.frame. Default = TRUE

Author(s)

E.D. Gennatas

---

dataPrepare rtemis-internals: dataPrepare

Description

Prepare data for **rtemis** supervised learning

Usage

```
dataPrepare(
  x,
  y = NULL,
  x.test = NULL,
  y.test = NULL,
  x.valid = NULL,
  y.valid = NULL,
  filter.y.na = FALSE,
  ipw = FALSE,
  ipw.type = 2,
  upsample = FALSE,
  downsample = FALSE,
  resample.seed = NULL,
  removeDots = FALSE,
  .preprocess = NULL,
  verbose = FALSE
)
```

Arguments

x	Either training set features or combined training features and outcome in last column
y	Either training set outcome or combined testing set features and outcome in last column.
x.test	Testing set features
y.test	Testing set outcome

x.valid	Matrix / Data frame: Validation set features
y.valid	Vector: Validation outcome
filter.y.na	Logical: If TRUE, filter out cases with missing values in y
ipw	Logical: If TRUE, return class weights for inverse probability weighting (for Classification)
ipw.type	1, 2: 1:
upsample	Logical: If TRUE, downsample majority class to match size of minority class
downsample	Logical: If TRUE, downsample majority class to match size of minority class
resample.seed	Integer: If set, use set.seed for reproducibility. Default = NULL
removeDots	Logical: If TRUE, replace dots in variable names with underscores. Some algorithms do not work with variable names containing dots (SparkML)
.preprocess	List: Preprocessing parameters to be passed to preprocess. Set with rtset.preprocess
verbose	Logical: If TRUE, print messages to console

Author(s)

E.D. Gennatas

`date2factor` *Date to factor time bin*

Description

Convert Date to time bin factor.

Usage

```
date2factor(
  x,
  time_bin = c("year", "quarter", "month", "day"),
  make_bins = c("range", "present"),
  bin_range = range(x, na.rm = TRUE),
  ordered = FALSE
)
```

Arguments

x	Date vector
time_bin	Character: "year", "quarter", "month", or "day"
make_bins	Character: "range" or "preseent". If "range" the factor levels will include all time periods define by time_bin within bin_range. This means factor levels can be empty. Otherwise, if "present", factor levels only include time periods present in data.
bin_range	Date, vector, length 2: Range of dates to make levels for. Defaults to range of input dates x
ordered	Logical: If TRUE, factor output is ordered. Default = FALSE

Details

Order of levels will be chronological (important e.g. for plotting) Additionally, can output ordered factor with `ordered = TRUE`

Value

factor of time periods

Author(s)

E.D. Gennatas

Examples

```
## Not run:
library(data.table)
startDate <- as.Date("2018-01-01")
endDate <- as.Date("2020-12-31")
time <- sample(seq(startDate, endDate, length.out = 100))
date2factor(time)
date2factor(time, "quarter")
date2factor(time, "month")
date2factor(time, "day")
# range vs present
x <- sample(seq(as.Date("2018-01-01"), as.Date("2021-01-01"), by = 1), 10)
date2factor(x, time_bin = "quarter", make_bins = "present")
date2factor(x, time_bin = "quarter", make_bins = "range")

## End(Not run)
```

`date2ym` *Date to year-month factor*

Description

Date to year-month factor

Usage

```
date2ym(x, ordered = FALSE)
```

Arguments

`x`	Date vector
`ordered`	Logical: If TRUE, return ordered factor. Default = FALSE

Author(s)

E.D. Gennatas

date2yq *Date to year-quarter factor*

Description

Date to year-quarter factor

Usage

```
date2yq(x, ordered = FALSE)
```

Arguments

x	Date vector
ordered	Logical: If TRUE, return ordered factor. Default = FALSE

Author(s)

E.D. Gennatas

ddSci *Format Numbers for Printing*

Description

2 Decimal places, otherwise scientific notation

Usage

```
ddSci(x, decimal.places = 2, hi = 1e+06, asNumeric = FALSE)
```

Arguments

x	Vector of numbers
decimal.places	Integer: Return this many decimal places. Default = 2
hi	Float: Threshold at or above which scientific notation is used. Default = 1e06
asNumeric	Logical: If TRUE, convert to numeric before returning. Default = FALSE. This will not force all numbers to print 2 decimal places. For example: 1.2035 becomes "1.20" if asNumeric = FALSE, but 1.2 otherwise This can be helpful if you want to be able to use the output as numbers / not just for printing.

Details

Numbers will be formatted to 2 decimal places, unless this results in 0.00 (e.g. if input was .0032), in which case they will be converted to scientific notation with 2 significant figures. ddSci will return 0.00 if the input is exactly zero. This function can be used to format numbers in plots, on the console, in logs, etc.

Value

Formatted number

Author(s)

E.D. Gennatas

Examples

```
x <- .34876549
ddSci(x)
# "0.35"
x <- .00000000457823
ddSci(x)
# "4.6e-09"
```

`decom` *Matrix Decomposition with* **rtemis**

Description

Convenience function to perform any **rtemis** decomposition

Usage

```
decom(x, decom = "ICA", x.test = NULL, verbose = TRUE, ...)
```

Arguments

`x`	Numeric matrix / data frame: Input data
`decom`	Character: Decomposer name. See]linkdecomSelect. Default = "ICA"
`x.test`	Numeric matrix / data frame: Testing set data if supported by `decom`
`verbose`	Logical: if TRUE, print messages to screen
`...`	Additional arguments to be passed to `decom`

Details

`decom` returns an R6 class object `rtDecom`

Value

rtDecom object

Author(s)

E.D. Gennatas

decomSelect *Select* **rtemis** *Decomposer*

Description

Accepts decomposer name (supports abbreviations) and returns **rtemis** function name or the function itself. If run with no parameters, prints list of available algorithms.

Usage

```
decomSelect(decom, fn = FALSE, desc = FALSE)
```

Arguments

decom	Character: Decomposition name. Case insensitive. e.g. "iso" for isomap
fn	Logical: If TRUE, return function, otherwise name of function. Defaults to FALSE
desc	Logical: If TRUE, return full name of algorithm decom

Value

Function or name of function (see param fn) or full name of algorithm (desc)

Author(s)

E.D. Gennatas

delayTime *Delay and Reverb Time Calculator*

Description

Calculates delay and reverb time in milliseconds given tempo in beats per minute (BPM) and delay/reverb time in note duration

Usage

```
delayTime(bpm = 120, note = "1/4")
```

Arguments

bpm	Integer: Beats per minute. Default = 120
note	Character: Delay/Reverb time in note duration: "2", "1", "1/2", "1/2T", "1/4D", "1/4", "1/4T", "1/8D", "1/8", "1/8T", "1/16D", "1/16", "1/16T", "1/32D", "1/32", "1/32T". "2" means a double note, "1" a whole, and so on. "T" denotes a triple note, "D" denotes a dotted note. Case insensitive. Default = "1/4" (quarter note)

Author(s)

E.D. Gennatas

dependency_check **rtemis** *internal: Dependencies check*

Description

Checks if dependencies can be loaded; names missing dependencies if not.

Usage

```
dependency_check(..., verbose = FALSE)
```

Arguments

...	List or vector of strings defining namespaces to be checked
verbose	Logical. If TRUE, print messages to consolde. Note: An error will always printed if dependencies are missing. Setting this to FALSE stops it from printing "Dependencies check passed".

Author(s)

E.D. Gennatas

---

desaturate *Pastelify a color (make a color more pastel)*

Description

Lower a color's saturation by a given percent in the HSV color system

Usage

```
desaturate(color, s = 0.3)
```

Arguments

color	Color, vector: Color(s) to operate on
s	Float: Decrease saturation by this fraction. Default = .3, which means if saturation of given color is 1, it will become .7

Value

List of adjusted colors

Author(s)

E.D. Gennatas

Examples

```
color <- c("red", "green", "blue")
color.p <- desaturate(color)
```

df_movecolumn *Move data frame column*

Description

Move data frame column

Usage

```
df_movecolumn(x, from, to = ncol(x))
```

Arguments

x	data.frame
from	String or Integer: Define which column holds the vector you want to move
to	Integer: Define which column number you want the vector to be moved to. Default = ncol(x) i.e. the last column.

Author(s)

E.D. Gennatas

Examples

```
mtcars_hp <- df_movecolumn(mtcars, "hp")
```

distillTreeRules *Distill rules from trained RF and GBM learners*

Description

Extract rules from RF or GBM model, prune, and remove unnecessary rules using inTrees

Usage

```
distillTreeRules(
  mod,
  x,
  y = NULL,
  n.trees = NULL,
  maxdepth = 100,
  maxDecay = 0.05,
  typeDecay = 2,
  verbose = TRUE
)
```

Arguments

mod	A trained RF or GBM model
x	The training set features
y	The training set outcomes. If NULL, assumed to be last column of x
n.trees	Integer: Number of trees to extract
maxdepth	Integer: Max depth to consider
maxDecay	Float: See inTree=es::pruneRule
typeDecay	Integer: See inTreees::pruneRule
verbose	Logical: If TRUE, print messages to output

Details

Models must be trained with s_RF or s_GBM

Author(s)

E.D. Gennatas

---

dplot3_addtree *Plot ADDTREE trees*

Description

Plot ADDTREE trees trained with s_ADDTREE using data.tree::plot.Node

Usage

```
dplot3_addtree(
  addtree,
  col.positive = "#F48024DD",
  col.negative = "#18A3ACDD",
  node.col = "#666666",
  node.shape = "none",
  node.labels = TRUE,
  node.labels.pct.pos = NULL,
  pos.name = NULL,
  edge.col = "#999999",
  edge.width = 2,
  edge.labels = FALSE,
  arrowhead = "vee",
  layout = "dot",
  drop.leaves = FALSE,
  rankdir = "TB",
  splines = "polyline",
  fontname = "helvetica",
  bg.color = "#ffffff",
  overlap = "false",
  prune = NULL,
  prune.empty.leaves = TRUE,
```

```
  remove.bad.parents = FALSE,
  filename = NULL
)
```

Arguments

col.positive	Color for outcome positive.
col.negative	Color for negative outcome.
node.col	Color for non-terminal leaves.
edge.col	Color for edges.
bg.color	Background color.
filename	Character: Path to filename to save PDF Requires packages DiagrammeRsvg and rsvg. Default = NULL

Details

Edge info and styles have been removed because of problems with DiagrammeR

Author(s)

E.D. Gennatas

---

dplot3_bar *Interactive Barplots*

Description

Draw interactive barplots using plotly

Usage

```
dplot3_bar(
  x,
  main = NULL,
  xlab = NULL,
  ylab = NULL,
  col = NULL,
  alpha = 1,
  horizontal = FALSE,
  theme = rtTheme,
  palette = rtPalette,
  barmode = c("group", "relative", "stack", "overlay"),
  group.names = NULL,
  order.by.val = FALSE,
  ylim = NULL,
  hovernames = NULL,
  feature.names = NULL,
  font.size = 16,
  annotate = FALSE,
  annotate.col = theme$labs.col,
```

```
  legend = NULL,
  legend.col = NULL,
  legend.xy = c(1, 1),
  legend.orientation = "v",
  legend.xanchor = "left",
  legend.yanchor = "auto",
  hline = NULL,
  hline.col = NULL,
  hline.width = 1,
  hline.dash = "solid",
  hline.annotate = NULL,
  hline.annotation.x = 1,
  margin = list(b = 65, l = 65, t = 50, r = 10, pad = 0),
  automargin.x = TRUE,
  automargin.y = TRUE,
  padding = 0,
  displayModeBar = TRUE,
  filename = NULL,
  output.format = "svg",
  file.width = 500,
  file.height = 500,
  file.scale = 1,
  trace = 0,
  ...
)
```

Arguments

x	vector (possibly named), matrix, or data.frame: If matrix or data.frame, rows are groups (can be 1 row), columns are features
main	Character: Plot title. Default = NULL
xlab	Character: x-axis label. Default = NULL
ylab	Character: y-axis label. Default = NULL
col	Color, vector: Color for bars. Default NULL, which will draw colors from palette
alpha	Float (0, 1]: Transparency for bar colors. Default = .8
horizontal	Logical: If TRUE, plot bars horizontally
theme	Character: Theme to use: Use themes() to get available themes
palette	Character: Name of **rtemis** palette to use. Default = "rtCol1". Only used if col = NULL
barmode	Character: Type of bar plot to make: "group", "relative", "stack", "overlay". Default = "group". Use "relative" for stacked bars, wich handles negative values correctly, unlike "stack", as of writing.
group.names	Character, vector, length = NROW(x): Group names. Default = NULL, which uses rownames(x)
order.by.val	Logical: If TRUE, order bars by increasing value. Only use for single group data. Default = NULL
ylim	Float, vector, length 2: y-axis limits.
hovernames	Character, vector: Optional character vector to show on hover over each bar.

feature.names	Character, vector, length = NCOL(x): Feature names. Default = NULL, which uses colnames(x)
font.size	Float: Font size for all labels. Default = 16
annotate	Logical: If TRUE, annotate stacked bars
annotate.col	Color for annotations
legend	Logical: If TRUE, draw legend. Default = NULL, and will be turned on if there is more than one feature present
legend.col	Color: Legend text color. Default = NULL, determined by theme
legend.xy	Numeric, vector, length 2: x and y for plotly's legend
legend.orientation	"v" or "h" for vertical or horizontal
legend.xanchor	Character: Legend's x anchor: "left", "center", "right", "auto"
legend.yanchor	Character: Legend's y anchor: "top", "middle", "bottom", "auto"
hline	Float: If defined, draw a horizontal line at this y value.
hline.col	Color for hline. Default = "#ff0000" (red)
hline.width	Float: Width for hline. Default = 1
hline.dash	Character: Type of line to draw: "solid", "dot", "dash", "longdash", "dashdot", or "longdashdot"
hline.annotate	Character: Text of horizontal line annotation if hline is set
hline.annotation.x	Numeric: x position to place annotation with paper as reference. 0: to the left of the plot area; 1: to the right of the plot area
margin	Named list: plot margins.
automargin.x	Logical: If TRUE, automatically set x-axis amrgins
automargin.y	Logical: If TRUE, automatically set y-axis amrgins
padding	Integer: N pixels to pad plot. Default = 0
displayModeBar	Logical: If TRUE, show plotly's modebar
filename	Character: Path to file to save static plot. Default = NULL
output.format	Character: "svg", "png", "jpeg", "pdf"
file.width	Integer: File width in pixels for when filename is set.
file.height	Integer: File height in pixels for when filename is set.
file.scale	Numeric: If saving to file, scale plot by this number
trace	Integer: The height the number the more diagnostic info is printed to the console
...	Additional arguments passed to theme

Author(s)

E.D. Gennatas

Examples

```
## Not run:
dplot3_bar(VADeaths, legend.xy = c(0, 1))
dplot3_bar(VADeaths, legend.xy = c(1, 1), legend.xanchor = "left")
# simple individual bars
a <- c(4, 7, 2)
dplot3_bar(a)
# if input is a data.frame, each row is a group and each column is a feature
b <- data.frame(x = c(3, 5, 7), y = c(2, 1, 8), z = c(4, 5, 2))
rownames(b) <- c("Jen", "Ben", "Ren")
dplot3_bar(b)
# stacked
dplot3_bar(b, barmode = "stack")

## End(Not run)
```

---

dplot3_box *Interactive Boxplots & Violin plots*

Description

Draw interactive boxplots or violin plots using **plotly**

Usage

```
dplot3_box(
  x,
  time = NULL,
  time.bin = c("year", "quarter", "month", "day"),
  type = c("box", "violin"),
  group = NULL,
  x.transform = c("none", "scale", "minmax"),
  main = NULL,
  xlab = "",
  ylab = NULL,
  col = NULL,
  alpha = 0.6,
  bg = NULL,
  plot.bg = NULL,
  theme = rtTheme,
  palette = rtPalette,
  boxpoints = "outliers",
  quartilemethod = "linear",
  violin.box = TRUE,
  orientation = "v",
  annotate_n = FALSE,
  annotate_n_y = 1,
  annotate.col = theme$labs.col,
  xnames = NULL,
  group.lines = TRUE,
  group.lines.col = NULL,
  group.lines.alpha = 0.5,
```

```
  labelify = TRUE,
  order.by.fn = NULL,
  font.size = 16,
  legend = NULL,
  legend.col = NULL,
  legend.xy = NULL,
  legend.orientation = "v",
  legend.xanchor = "auto",
  legend.yanchor = "auto",
  xaxis.type = "category",
  margin = list(b = 65, l = 65, t = 50, r = 10, pad = 0),
  automargin.x = TRUE,
  automargin.y = TRUE,
  boxgroupgap = NULL,
  hovertext = NULL,
  show_n = FALSE,
  htest = "none",
  htest.thresh = 0.05,
  htest.y = 1,
  htest.annotate = TRUE,
  htest.annotate.x = 0,
  htest.annotate.y = -0.05,
  use.plotly.group = FALSE,
  displayModeBar = TRUE,
  filename = NULL,
  output.format = "svg",
  file.width = 500,
  file.height = 500,
  file.scale = 1,
  ...
)
```

Arguments

x	Vector or List of vectors: Input
time	Date or date-time vector
time.bin	Character: "year", "quarter", "month", or "day". What to bin by
type	Character: "box" or "violin"
group	Factor to group by
x.transform	Character: "none", "scale", or "minmax" to use raw values, scaled and centered values or min-max normalized to 0-1, respectively. Transform is applied to each variable before grouping, so that groups are comparable
main	Character: Plot title.
xlab	Character: x-axis label.
ylab	Character: y-axis label.
col	Color, vector: Color for boxes. Default NULL, which will draw colors from palette
alpha	Float (0, 1]: Transparency for box colors.
bg	Color: Background color. Default = "white"

plot.bg	Color: Background color for plot area.
theme	Character: Theme to use: Run themes() for available themes
palette	Character: Name of **rtemis** palette to use. Default = "rtCol1". Only used if col = NULL
boxpoints	Character or FALSE: "all", "suspectedoutliers", "outliers" See https://plotly.com/r/box-plots/#choosing-the-algorithm-for-computing-quartiles
quartilemethod	Character: "linear", "exclusive", "inclusive"
violin.box	Logical: If TRUE and type is "violin" show box within violin plot
orientation	Character: "v" or "h" for vertical, horizontal
annotate_n	Logical: If TRUE, annotate with N in each box
annotate_n_y	Numeric: y position for annotate_n
annotate.col	Color for annotations
xnames	Character, vector, length = NROW(x): x-axis names. Default = NULL, which tries to set names appropriately
group.lines	Logical: If TRUE, add separating lines between groups of boxplots
group.lines.col	Color for group.lines
group.lines.alpha	Numeric: transparency for group.lines.col
labelify	Logical: If TRUE, labelify x names
order.by.fn	Function: If defined, order boxes by increasing value of this function (e.g. median). Default = NULL
font.size	Float: Font size for all labels. Default = 16
legend	Logical: If TRUE, draw legend. Default = TRUE
legend.col	Color: Legend text color. Default = NULL, determined by theme
legend.xy	Float, vector, length 2: Relative x, y position for legend. Default = NULL, which places the legend top right beside the plot area. For example, c(0, 1) places the legend top left within the plot area
legend.orientation	"v" or "h" for vertical, horizontal
legend.xanchor	Character: Legend's x anchor: "left", "center", "right", "auto"
legend.yanchor	Character: Legend's y anchor: "top", "middle", "bottom", "auto"
xaxis.type	Character: "linear", "log", "date", "category", "multicategory" Default = "category"
margin	Named list: plot margins. Default = list(b = 65, l = 65, t = 50, r = 10, pad = 0)
automargin.x	Logical: If TRUE, automatically set x-axis amrgins
automargin.y	Logical: If TRUE, automatically set y-axis amrgins
boxgroupgap	Numeric: Sets the gap (in plot fraction) between boxes of the same location coordinate
hovertext	Character vector: Text to show on hover for each data point
show_n	Logical: If TRUE, show N in each box

htest	Character: e.g. "t.test", "wilcox.test" to compare each box to the *first* box. If grouped, compare within each group to the first box. If p-value of test is less than htest.thresh, add asterisk above/ to the side of each box
htest.thresh	Numeric: Significance threshold for htest
htest.annotate	Logical: if TRUE, include htest annotation e.g. "*pval < 0.05"
htest.annotate.x	Numeric: x-axis paper coordinate for htest annotation
htest.annotate.y	Numeric: y-axis paper coordinate for htest annotation
use.plotly.group	If TRUE, use plotly's group arg to group boxes.
displayModeBar	Logical: If TRUE, show plotly's modebar
filename	Character: Path to file to save static plot.
output.format	Character: "svg", "png", "jpeg", "pdf"
file.width	Integer: File width in pixels for when filename is set.
file.height	Integer: File height in pixels for when filename is set.
file.scale	Numeric: If saving to file, scale plot by this number
...	Additional arguments passed to theme

Details

For multiple box plots, the recommendation is: - 'x=dat[, columnindex]' for multiple variables of a data.frame - 'x=list(a=..., b=..., etc.)' for multiple variables of potentially different length - 'x=split(var, group)' for one variable with multiple groups: group names appear below boxplots - 'x=dat[, columnindex], group = factor' for grouping multiple variables: group names appear in legend

If orientation = "h", xlab is applied to y-axis and vice versa. Similarly, x.axist.type applies to y-axis - this defaults to "category" and would not normally need changing.

Author(s)

E.D. Gennatas

Examples

```
## Not run:
# A.1 Box plot of 4 variables
dplot3_box(iris[, 1:4])
# A.2 Grouped Box plot
dplot3_box(iris[, 1:4], group = iris$Species)
dplot3_box(iris[, 1:4], group = iris$Species, annotate_n = TRUE)
# B. Boxplot binned by time periods
# Synthetic data with an instantenous shift in distributions
set.seed(2021)
dat1 <- data.frame(alpha = rnorm(200, 0), beta = rnorm(200, 2), gamma = rnorm(200, 3))
dat2 <- data.frame(alpha = rnorm(200, 5), beta = rnorm(200, 8), gamma = rnorm(200, -3))
x <- rbind(dat1, dat2)
startDate <- as.Date("2019-12-04")
endDate <- as.Date("2021-03-31")
time <- seq(startDate, endDate, length.out = 400)
dplot3_box(x[, 1], time, "year", ylab = "alpha")
```

```
dplot3_box(x, time, "year", legend.xy = c(0, 1))
dplot3_box(x, time, "quarter", legend.xy = c(0, 1))
dplot3_box(x, time, "month",
    legend.orientation = "h",
    legend.xy = c(0, 1),
    legend.yanchor = "bottom"
)
# (Note how the boxplots widen when the period includes data from both dat1 and dat2)

## End(Not run)
```

---

dplot3_cart dplot3: data.tree *trees*

Description

Plot data.tree trees using data.tree::plot.Node

Usage

```
dplot3_cart(
  object,
  col.positive = "#F48024DD",
  col.negative = "#18A3ACDD",
  col.lo = "#80ffff",
  col.mid = "gray20",
  col.hi = "#F4A0FF",
  node.col = "#666666",
  node.shape = "none",
  node.labels = TRUE,
  node.cond = TRUE,
  node.prob = TRUE,
  node.estimate = NULL,
  node.n = TRUE,
  edge.col = "#999999",
  edge.width = 2,
  edge.labels = FALSE,
  arrowhead = "vee",
  layout = "dot",
  drop.leaves = FALSE,
  rankdir = "TB",
  splines = "polyline",
  fontname = "helvetica",
  bg.color = "white",
  overlap = "false",
  prune = FALSE,
  prune.empty.leaves = TRUE,
  remove.bad.parents = TRUE,
  rpart.cp = NULL,
  verbose = TRUE
)
```

Arguments

col.positive	Color for outcome positive.
col.negative	Color for negative outcome.
node.col	Color for non-terminal leaves.
node.cond	Logical: If TRUE, print the splitting condition inside each node. Default = TRUE
node.prob	Logical: If TRUE, print the probability estimate for the first class of the outcome inside each node. Default = TRUE
node.estimate	Logical: If TRUE, print the estimated outcome level inside each node. Default = FALSE
node.n	Logical: If TRUE, print the number of cases (from training data) that matched this condition
edge.col	Color for edges.
bg.color	Background color.

Details

If you want to show split conditions as edge labels (`edge.labels = TRUE`), it is recommened to set `rankdir = "LR"` and `node.cond = FALSE`. Edge labels in graphviz are shown to the right of the edge when `rankdir = "TB"` and above when `rankdir = "LR"`.

Author(s)

E.D. Gennatas

---

dplot3_graphd3 *Plot graph using* **networkD3**

Description

Plot graph using **networkD3**

Usage

```
dplot3_graphd3(
  net,
  groups = NULL,
  color.scale = NULL,
  edge.col = NULL,
  node.col = NULL,
  node.alpha = 0.5,
  edge.alpha = 0.33,
  zoom = TRUE,
  legend = FALSE,
  palette = rtPalette,
  theme = rtTheme,
  ...
)
```

Arguments

net	**igraph** network
groups	Vector, length n nodes indicating group/cluster/community membership of nodes in net
color.scale	D3 colorscale (e.g. networkD3::JS("d3.scaleOrdinal(d3.schemeCategory20b);"))
edge.col	Color for edges
node.col	Color for nodes
node.alpha	Float [0, 1]: Node opacity. Default = .5
edge.alpha	Float [0, 1]: Edge opacity. Default = .33
zoom	Logical: If TRUE, graph is zoomable. Default = TRUE
legend	Logical: If TRUE, display legend for groups
palette	Vector of colors, or Character defining a builtin palette - get options with rtpalette()
theme	rtemis theme to use
...	Additional arguments to pass to networkD3

Author(s)

E.D. Gennatas

---

dplot3_graphjs *Plot network using* **threejs::graphjs**

Description

Interactive plotting of an **igraph** net using **threejs**

Usage

```
dplot3_graphjs(
  net,
  vertex.size = 1,
  vertex.col = NULL,
  vertex.label.col = NULL,
  vertex.label.alpha = 0.66,
  vertex.frame.col = NA,
  vertex.label = NULL,
  vertex.shape = "circle",
  edge.col = NULL,
  edge.alpha = 0.5,
  edge.curved = 0.35,
  edge.width = 2,
 layout = c("fr", "dh", "drl", "gem", "graphopt", "kk", "lgl", "mds", "sugiyama"),
  coords = NULL,
  layout_params = list(),
  cluster = NULL,
  groups = NULL,
  cluster_params = list(),
```

```
  cluster_mark_groups = TRUE,
  cluster_color_vertices = FALSE,
  main = "",
  theme = rtTheme,
  theme_extra_args = list(),
  palette = rtPalette,
  mar = rep(0, 4),
  par.reset = TRUE,
  filename = NULL,
  verbose = TRUE,
  ...
)
```

Arguments

net	igraph network
vertex.size	Numeric: Vertex size
vertex.col	Color for vertices
vertex.label.col	Color for vertex labels
vertex.label.alpha	Numeric: transparency for vertex.label.col
vertex.frame.col	Color for vertex border (frame)
vertex.label	Character vector: Vertex labels. Default = NULL, which will keep existing names in net if any. Set to NA to avoid printing vertex labels
vertex.shape	Character, vector, length 1 or N nodes: Vertex shape. See graphjs("vertex.shape"). Default = "circle"
edge.col	Color for edges
edge.alpha	Numeric: Transparency for edges
edge.curved	Numeric: Curvature of edges. Default = .35
edge.width	Numeric: Edge thickness
layout	Character: one of: "fr", "dh", "drl", "gem", "graphopt", "kk", "lgl", "mds", "sugiyama", corresponding to all the available layouts in **igraph**
coords	Output of precomputed **igraph** layout. If provided, layout is ignored
layout_params	List of parameters to pass to layout function
cluster	Character: one of: "edge_betweenness", "fast_greedy", "infomap", "label_prop", "leading_eigen", "louvain", "optimal", "spinglass", "walktrap", corresponding to all the available **igraph** clustering functions
groups	Output of precomputed **igraph** clustering. If provided, cluster is ignored
cluster_params	List of parameters to pass to cluster function
cluster_mark_groups	Logical: If TRUE, draw polygons to indicate clusters, if groups or cluster defined
cluster_color_vertices	Logical: If TRUE, color vertices by cluster membership
main	Character: main title

theme	**rtemis** theme to use
theme_extra_args	List of extra arguments to pass to the theme function defined by theme. This argument is used when the extra args (...) are passed the plotting function, in this case igraph::plot.igraph and not to the theme function
palette	Color vector or name of rtemis palette
mar	Numeric vector, length 4: par's margin argument
par.reset	Logical: If TRUE, reset par before exiting. Default = TRUE
filename	Character: If provided, save plot to this filepath
verbose	Logical, If TRUE, print messages to console. Default = TRUE
...	Extra arguments to pass to igraph::plot.igraph()

Author(s)

E.D. Gennatas

---

dplot3_heatmap *Interactive Heatmaps*

Description

Draw interactive heatmaps using heatmaply

Usage

```
dplot3_heatmap(
  x,
  Rowv = TRUE,
  Colv = TRUE,
  cluster = FALSE,
  symm = FALSE,
  cellnote = NULL,
  colorGrad.n = 101,
  colors = NULL,
  space = "rgb",
  lo = "#18A3AC",
  lomid = NULL,
  mid = NULL,
  midhi = NULL,
  hi = "#F48024",
  k_row = 1,
  k_col = 1,
  grid.gap = 0,
  limits = NULL,
  margins = NULL,
  main = NULL,
  xlab = NULL,
  ylab = NULL,
  key.title = NULL,
```

```
  showticklabels = NULL,
  colorbar_len = 0.7,
  plot_method = "plotly",
  theme = rtTheme,
  row_side_colors,
  row_side_palette = NULL,
  col_side_colors,
  col_side_palette = NULL,
  font.size = NULL,
  padding = 0,
  filename = NULL,
  ...
)
```

Arguments

x	Input matrix
Rowv	Logical or dendrogram. If Logical: Compute dendrogram and reorder rows. Defaults to FALSE If dendrogram: use as is, without reordering See more at `heatmaply::heatmaply("Rowv")`
Colv	Logical or dendrogram. If Logical: Compute dendrogram and reorder columns. Defaults to FALSE If dendrogram: use as is, without reordering See more at `heatmaply::heatmaply("Colv")`
cluster	Logical: If TRUE, set Rowv and Colv to TRUE
symm	Logical: If TRUE, treat x symmetrically - x must be a square matrix. Default = FALSE
cellnote	Matrix with values to be desplayed on hover. Defaults to `ddSci(x)`
colorGrad.n	Integer: Number of distinct colors to generate using colorGrad. Default = 101
colors	Character: Acts as a shortcut to defining lo, mid, etc for a number of defaults: "french", "penn", "grnblkred",
space	Character: Which colorspace to use. Option: "rgb", or "Lab". Default = "rgb". Recommendation: If mid is "white" or "black" (default), use "rgb", otherwise "Lab"
lo	Color for low end
lomid	Color for low-mid
mid	Color for middle of the range or "mean", which will result in `colorOp(c(lo, hi), "mean")`. If mid = NA, then only lo and hi are used to create the color gradient.
midhi	Color for middle-high
hi	Color for high end
k_row	Integer: Number of desired number of groups by which to color dendrogram branches in the rows. Default = NA (determined automatically). See `heatmaply::heatmaply("k_ro`
k_col	Integer: Number of desired number of groups by which to color dendrogram branches in the columns. Default = NA (determined automatically). See `heatmaply::heatmaply("k`
grid.gap	Integer: Space between cells. Default = 0 (no space)
limits	Float, length 2: Determine color range. Default = NULL, which automatically centers values around 0

margins	Float, length 4: Heatmap margins. Default = c(30, 30, 30, 30)
main	Character: Plot title
xlab	Character: x-axis label
ylab	Character: y-axis label
key.title	Character: Title for the color key. Default = NULL (no title)
plot_method	Character: Update February 2021: "ggplot" causes R session to hand on MacOS but plotly" seems to work
theme	Character: "light", "dark"
filename	String (Optional: Path to file to save colorbar
...	Additional arguments to be passed to heatmaply::heatmaply

Author(s)

E.D. Gennatas

Examples

```
## Not run:
x <- rnormmat(200, 20)
xcor <- cor(x)
dplot3_heatmap(xcor)

## End(Not run)
```

---

dplot3_leaflet *Plot interactive choropleth map using* **leaflet**

Description

Plot interactive choropleth map using **leaflet**

Usage

```
dplot3_leaflet(
  fips,
  values,
  names = NULL,
  fillOpacity = 1,
  palette = NULL,
  color.mapping = c("Numeric", "Bin"),
  col.lo = "#0290EE",
  col.hi = "#FE4AA3",
  col.na = "#303030",
  col.highlight = "#FE8A4F",
  col.interpolate = c("linear", "spline"),
  col.bins = 21,
  domain = NULL,
  weight = 0.5,
  color = "black",
```

```
  alpha = 1,
  bg.tile.provider = leaflet::providers$Stamen.TonerBackground,
  bg.tile.alpha = 0.67,
  fg.tile.provider = leaflet::providers$Stamen.TonerLabels,
  legend.position = c("topright", "bottomright", "bottomleft", "topleft"),
  legend.alpha = 0.8,
  legend.title = NULL,
  init.lng = -98.5418083333333,
  init.lat = 39.2074138888889,
  init.zoom = 3,
  stroke = TRUE
)
```

Arguments

fips	Character vector of FIPS codes. (If numeric, it will be appropriately zero-padded)
values	Values to map to fips
names	Character vector: Optional county names to appear on hover along values
fillOpacity	Float: Opacity for fill colors. Default = 1
palette	Character: Color palette to use
color.mapping	Character: "Numeric" or "Bin"
col.lo	Overlay color mapped to lowest value
col.hi	Overaly color mapped to highest value
col.na	Color mappes to NA values
col.highlight	Hover border color. Default = "#FE8A4F" (orange)
col.interpolate	Character: "linear" or "spline"
col.bins	Integer: Number of color bins to create if color.mapping = "Bin". Default = 21
domain	Limits for mapping colors to values. Default = NULL and set to range
weight	Float: Weight of county border lines. Default = .5
color	Color of county border lines. Default = "black"
alpha	Float: Overaly transparency. Default = 1
bg.tile.provider	Background tile (below overlay colors), one of leaflet::providers
bg.tile.alpha	Float: Background tile transparency. Default = .67
fg.tile.provider	Foreground tile (above overlay colors), one of leaflet::providers
legend.position	Character: One of: "topright", "bottomright", "bottomleft", "topleft". Default = "topright"
legend.alpha	Float: Legend box transparency. Default = .8
legend.title	Character: Defaults to name of values variable.
init.lng	Float: Center map around this longitude (in decimal form). Default = -98.54180833333334 (US geographic center)

init.lat	Float: Center map around this latitude (in decimal form). Default = 39.207413888888894 (US geographic center)
init.zoom	Integer: Initial zoom level (depends on device, i.e. window, size). Default = 3
stroke	Logical: If TRUE, draw polygon borders. Default = TRUE

Author(s)

E.D. Gennatas

Examples

```
## Not run:
fips <- c(06075, 42101)
population <- c(874961, 1579000)
names <- c("SF", "Philly")
dplot3_leaflet(fips, supervals, names)

## End(Not run)
```

---

dplot3_linad *Plot a Linear Additive Tree trained by* s_LINAD *using* **visNetwork**

Description

Plot a Linear Additive Tree trained by s_LINAD using **visNetwork**

Usage

```
dplot3_linad(
  x,
  main = NULL,
  bg = "#FFFFFF",
  shape = "box",
  nodelabels = TRUE,
  ncases.inlabels = TRUE,
  rules.on.edges = FALSE,
  log = FALSE,
  top = NULL,
  root.col = "#202020",
  node.col = "#5a5a5a",
  leaf.col = "#178CCB",
  edge.col = "#848484",
  edge.width = 4,
  arrow.scale = 0.7,
  arrow.middle = FALSE,
  col.highlight = "#FE4AA3",
  node.font.col = NULL,
  edge.font.col = "#000000",
  sort.coefs = FALSE,
  height = NULL,
```

```
  width = NULL,
  levelSeparation = 100,
  tree.font.size = 22,
  edgethickness.by.ncases = FALSE,
  font.family = "Lato",
  tooltip.coefs = TRUE,
  tooltip.delay = 50,
  table.font.size = "16px",
  table.dat.padding = "0px",
  table.lo.col = "#0290EE",
  table.hi.col = "#FE4AA3",
  dragNodes = FALSE,
  zoomView = FALSE,
  nodeSpacing = 150,
  blockShifting = TRUE,
  edgeMinimization = TRUE,
  parentCentralization = TRUE,
  direction = "UD",
  trace = 0
)
```

Arguments

main	Character: Title. Default = NULL
bg	Background color. Default = "#FFFFFF" (white)
shape	Character: Node shape; one of: "square", "triangle", "box", "circle", "dot", "star", "ellipse", "database", "text", "diamond". Default = "box"
nodelabels	Logical: If TRUE, inlcude node labels. Default = TRUE
ncases.inlabels	Logical: If TRUE, include number of cases with the node labels. Default = TRUE
rules.on.edges	Logical: If TRUE, display rules on edges instead of nodes. Default = FALSE
node.col	Color for nodes. Default = #7F7F7F" (some gray)
leaf.col	Color for leaf nodes. Default = "#18A3AC" (teal)
edge.col	Color for edges. Default = "#848484" (another gray)
col.highlight	Color for surrounding edges when node is selected. Default = "#F48024" (orange)
node.font.col	Color for node labels. Default varies by shape, black or white depending if visNetwork draws labels on node or underneath
edge.font.col	Color for edge labels. Default = "#000000" (black)
sort.coefs	Logical: If TRUE, sort each coefs table. Default = FALSE
height	Float: Height for visNetwork. Default = NULL, i.e. auto
width	Float: Width for visNetwork. Default = NULL, i.e. auto
levelSeparation	Float: N of pixels to separate tree levels. Default = 100
tree.font.size	Integer: Font size for tree labels. Default = 22
font.family	Character: Font to use throughout. Default = 'Helvetica Neue', because otherwise it will fail on a number of external viewers, but feel free to play around, esp. within RStudio

`tooltip.coefs` Logical: If TRUE, show html coefficient tables on hover over nodes. This was placed here before a custom html table creation function was made to replace some impossibly slow alternatives.

`tooltip.delay` Float: Delay (in milliseconds) on mouse over before showing tooltip. Default = 50

`table.font.size`
Character: Font size for html coefficient on-hover tables. Default = "14px"

`table.dat.padding`
Ignore, has no visible effect. Otherwise, Character: html table padding. Default = "0px"

`table.lo.col` Color for lowest coefficient values (negative). Default = "#80FFFF" (light blue)

`table.hi.col` Color for highest coefficient values (positive). Default = "#FFBE00" (light orange)

`trace` Integer: If > 0, print info to console (not particularly informative). Default = 0

`tree` `s_LINAD` tree

`edgethickness.by.ncases.`
Logical: If TRUE, scale edge thickness by number of cases with weight = 1

`dplot3_pie` *Interactive Pie Chart*

Description

Draw interactive pie charts using `plotly`

Usage

```
dplot3_pie(
  x,
  main = NULL,
  xlab = NULL,
  ylab = NULL,
  col = NULL,
  alpha = 0.8,
  bg = NULL,
  plot.bg = NULL,
  theme = getOption("rt.theme", "black"),
  palette = rtPalette,
  category.names = NULL,
  textinfo = "label+percent",
  font.size = 16,
  labs.col = NULL,
  legend = TRUE,
  legend.col = NULL,
  sep.col = NULL,
  margin = list(b = 50, l = 50, t = 50, r = 20),
  padding = 0,
  displayModeBar = TRUE,
  filename = NULL,
```

```
  file.width = 500,
  file.height = 500,
  file.scale = 1,
  ...
)
```

Arguments

x	data.frame: Input: Either a) 1 numeric column with categories defined by rownames, or b) two columns, the first is category names, the second numeric or c) a numeric vector with categories defined using the `category.names` argument
main	Character: Plot title. Default = NULL, which results in colnames(x)[1],
xlab	Character: x-axis label. Default = NULL
ylab	Character: y-axis label. Default = NULL
col	Color, vector: Color for bars. Default NULL, which will draw colors from `palette`
alpha	Float (0, 1]: Transparency for bar colors. Default = .8
theme	Character: "light", "dark". Default = `getOption("rt.theme", "light")`
palette	Character: Name of **rtemis** palette to use. Default = "rtCol1". Only used if `col = NULL`
category.names	Character, vector, length = NROW(x): Category names. Default = NULL, which uses either `rownames(x)`, or the first column of `x` if `ncol(x) = 2`
textinfo	Character: Info to show over each slince: "label", "percent", "label+percent" Default = "label+percent"
font.size	Float: Font size for all labels. Default = 16
legend	Logical: If TRUE, draw legend. Default = NULL, and will be turned on if there is more than one feature present
legend.col	Color: Legend text color. Default = NULL, determined by theme
sep.col	Separator color
margin	Named list: plot margins.
padding	Integer: N pixels to pad plot. Default = 0
displayModeBar	Logical: If TRUE, show plotly's modebar
filename	Character: Path to file to save static plot. Default = NULL
file.width	Integer: File width in pixels for when `filename` is set.
file.height	Integer: File height in pixels for when `filename` is set.
file.scale	Numeric: If saving to file, scale plot by this number
...	Additional arguments passed to theme

Author(s)

E.D. Gennatas

Examples

```
## Not run:
dplot3_pie(VADeaths[, 1, drop = F])

## End(Not run)
```

dplot3_pvals *Barplot p-values using dplot3_bar*

Description

Plot 1 - p-values as a barplot

Usage

```
dplot3_pvals(
  x,
  xnames = NULL,
  yname = NULL,
  p.adjust.method = "none",
  pval.hline = 0.05,
  hline.col = "#FE4AA3",
  hline.dash = "dash",
  ...
)
```

Arguments

x	Float, vector: p-values
xnames	Character, vector: feature names
yname	Character: outcome name
p.adjust.method	Character: method for p.adjust. Default = "none"
pval.hline	Float: Significance level at which to plot horizontal line. Default = .05
hline.col	Color for pval.hline. Default = "#FE4AA3"
hline.dash	Character: type of line to draw. Default = "dash"
...	Additional arguments passed to dplot3_bar

Author(s)

E.D. Gennatas

dplot3_shytreecoef *Coefficient heatmap for The Hybrid Tree*

Description

Coefficient heatmap for The Hybrid Tree

Usage

```
dplot3_shytreecoef(
  tree,
  Rowv = FALSE,
  Colv = FALSE,
  leaf.col = "#18A3AC",
  lo = "#2B27F1",
  mid = "black",
  hi = "#FFBE00"
)
```

Author(s)

E.D. Gennatas

---

dplot3_table *Simple HTML table*

Description

Draw an html table using `plotly`

Usage

```
dplot3_table(
  x,
  .ddSci = TRUE,
  main = NULL,
  main.col = "black",
  main.x = 0,
  main.xanchor = "auto",
  fill.col = "#18A3AC",
  table.bg = "white",
  bg = "white",
  line.col = "white",
  lwd = 1,
  header.font.col = "white",
  table.font.col = "gray20",
  font.size = 14,
  font.family = "Helvetica Neue",
  margin = list(l = 0, r = 5, t = 30, b = 0, pad = 0)
)
```

Arguments

`x`	data.frame: Table to draw
`.ddSci`	Logical: If TRUE, apply ddSci to numeric columns. Default = TRUE
`main`	Character: Table tile. Default = NULL
`main.col`	Color: Title color. Default = "black"
`main.x`	Float [0, 1]: Align title: 0: left, .5: center, 1: right. Default = 0

main.xanchor	Character: "auto", "left", "right": plotly's layout xanchor for title. Default = "auto"
fill.col	Color: Used to fill header with column names and first column with row names. Default = "#18A3AC" (teal)
table.bg	Color: Table background. Default = "white"
bg	Color: Background. Default = "white"
line.col	Color: Line color. Default = "white"
lwd	Float: Line width. Default = 1
header.font.col	Color: Header font color. Default = "white"
table.font.col	Color: Table font color. Default = "gray20"
font.size	Integer: Font size. Default = 14
font.family	Character: Font family. Default = "Helvetica Neue"
margin	List: plotly's margins. Default = `list(l = 0, r = 5, t = 30, b = 0, pad = 0)`

Author(s)

E.D. Gennatas

`dplot3_ts` *Interactive Timeseries Plots*

Description

Draw interactive timeseries plots using `plotly`

Usage

```
dplot3_ts(
  x,
  time,
  window = 7L,
  group = NULL,
  roll.fn = c("mean", "median", "max", "none"),
  roll.col = NULL,
  roll.alpha = 1,
  roll.lwd = 2,
  roll.name = NULL,
  alpha = NULL,
  align = "center",
  group.names = NULL,
  xlab = "Time",
  n.xticks = 12,
  scatter.type = "scatter",
  legend = TRUE,
  x.showspikes = TRUE,
  y.showspikes = FALSE,
  spikedash = "solid",
```

```
  spikemode = "across",
  spikesnap = "hovered data",
  spikecolor = NULL,
  spikethickness = 1,
  displayModeBar = TRUE,
  theme = rtTheme,
  palette = rtPalette,
  filename = NULL,
  ...
)
```

Arguments

x	Numeric vector of values to plot or list of vectors
time	Numeric or Date vector of time corresponding to values of x
window	Integer: apply roll.fn over this many units of time
group	Factor defining groups
roll.fn	Character: "mean", "median", "max", or "sum": Function to apply on rolling windows of x
roll.col	Color for rolling line
roll.alpha	Numeric: transparency for rolling line
roll.lwd	Numeric: width of rolling line
roll.name	Rolling function name (for annotation)
alpha	Numeric [0, 1]: Transparency
align	Character: "center", "right", or "left"
group.names	Character vector of group names
xlab	Character: x-axis label
n.xticks	Integer: number of x-axis ticks to use (approximately)
scatter.type	Character: "scatter" or "lines"
legend	Logical: If TRUE, show legend
x.showspikes	Logical: If TRUE, show x-axis spikes on hover
y.showspikes	Logical: If TRUE, show y-axis spikes on hover
spikedash	Character: dash type string ("solid", "dot", "dash", "longdash", "dashdot", or "longdashdot") or a dash length list in px (eg "5px,10px,2px,2px")
spikemode	Character: If "toaxis", spike line is drawn from the data point to the axis the series is plotted on. If "across", the line is drawn across the entire plot area, and supercedes "toaxis". If "marker", then a marker dot is drawn on the axis the series is plotted on
spikesnap	Character: "data", "cursor", "hovered data". Determines whether spikelines are stuck to the cursor or to the closest datapoints.
spikecolor	Color for spike lines
spikethickness	Numeric: spike line thickness
displayModeBar	Logical: If TRUE, display plotly's modebar
theme	Character: theme name or list of theme parameters
palette	Character: palette name, or list of colors
filename	Character: Path to filename to save plot
...	Additional arguments to be passed to dplot3_xy

Author(s)

E.D. Gennatas

Examples

```
## Not run:
time <- sample(seq(as.Date("2020-03-01"), as.Date("2020-09-23"), length.out = 140))
x1 <- rnorm(140)
x2 <- rnorm(140, 1, 1.2)
# Single timeseries
dplot3_ts(x1, time)
# Multiple timeseries input as list
dplot3_ts(list(Alpha = x1, Beta = x2), time)
# Multiple timeseries grouped by group, different lengths
time1 <- sample(seq(as.Date("2020-03-01"), as.Date("2020-07-23"), length.out = 100))
time2 <- sample(seq(as.Date("2020-05-01"), as.Date("2020-09-23"), length.out = 140))
time <- c(time1, time2)
x <- c(rnorm(100), rnorm(140, 1, 1.5))
group <- c(rep("Alpha", 100), rep("Beta", 140))
dplot3_ts(x, time, 7, group)

## End(Not run)
```

---

dplot3_varimp *Interactive Variable Importance Plot*

Description

Plot variable importance using `plotly`

Usage

```
dplot3_varimp(
  x,
  names = NULL,
  main = NULL,
  xlab = "Variable Importance",
  ylab = "",
  plot.top = 1,
  labelify = TRUE,
  col = NULL,
  alpha = 1,
  palette = NULL,
  mar = NULL,
  font.size = 16,
  axis.font.size = 14,
  theme = rtTheme,
  showlegend = TRUE,
  ...
)
```

Arguments

x	Vector, numeric: Input
names	Vector, string: Names of features
main	Character: main title
xlab	Character: x-axis label
ylab	Character: y-axis label
plot.top	Float or Integer: If <= 1, plot this percent highest absolute values, otherwise plot this many top values. i.e.: plot.top = .2 will print the top 20 highest values
labelify	Logical: If TRUE convert names(x) using labelify. Default = TRUE
col	Vector, colors: Single value, or multiple values to define bar (feature) color(s)
alpha	Numeric: Transparency
palette	Character: Name of **rtemis** palette to use. Default = "rtCol1". Only used if col = NULL
mar	Vector, numeric, length 4: Plot margins in pixels (NOT inches). Default = c(50, 110, 50, 50)
font.size	Integer: Overall font size to use (essentially for the title at this point). Default = 14
axis.font.size	Integer: Font size to use for axis labels and tick labels (Seems not to be in same scale as font.size for some reason. Experiment!)
theme	Output of an rtemis theme function (list of parameters) or theme name. Use themes() to print available themes.
showlegend	Logical: If TRUE, show legend
...	Additional arguments passed to theme
pad	Integer: Pad plot by this many pixels. Default = 10
font.family	Character: Font to use. Default = "Open Sans"
font.color	Color for all text

Details

A simple plotly wrapper to plot horizontal barplots, sorted by value, which can be used to visualize variable importance, model coefficients, etc.

Author(s)

E.D. Gennatas

Examples

```
# made-up data
x <- rnorm(10)
names(x) <- paste0("Feature_", seq(x))
dplot3_varimp(x)
```

dplot3_volcano *Volcano Plot*

Description

Volcano Plot

Usage

```
dplot3_volcano(
  x,
  pvals,
  xnames = NULL,
  group = NULL,
  x.thresh = 0,
  p.thresh = 0.05,
  p.transform = function(x) -log10(x),
 p.adjust.method = c("holm", "hochberg", "hommel", "bonferroni", "BH", "BY", "fdr",
    "none"),
  legend = NULL,
  legend.lo = NULL,
  legend.hi = NULL,
  label.lo = "Low",
  label.hi = "High",
  xlab = NULL,
  ylab = NULL,
  margin = list(b = 65, l = 65, t = 50, r = 10, pad = 0),
  xlim = NULL,
  ylim = NULL,
  alpha = NULL,
  hline = NULL,
  hline.col = NULL,
  hline.width = 1,
  hline.dash = "solid",
  hline.annotate = NULL,
  hline.annotation.x = 1,
  annotate.col = theme$labs.col,
  theme = rtTheme,
  font.size = 16,
  palette = NULL,
  legend.x.lo = NULL,
  legend.x.hi = NULL,
  legend.y = 0.97,
  annotate.n = 7,
  ay.lo = NULL,
  ay.hi = NULL,
  annotate = TRUE,
  annotate.alpha = 0.7,
  hovertext = NULL,
  displayModeBar = FALSE,
  filename = NULL,
```

```
  file.width = 500,
  file.height = 500,
  file.scale = 1,
  verbose = TRUE,
  ...
)
```

Arguments

x	Numeric vector: Input values, e.g. log2 fold change, coefficients, etc.
pvals	Numeric vector: p-values
xnames	Character vector: x names
group	Factor: Used to color code points. If NULL, significant points below x.thresh, non-significant points, and significant points above x.thresh will be plotted with the first, second and third color fo palette
x.thresh	Numeric x-axis threshold separating low from high
p.thresh	Numeric: p-value threshold of significance. Default = .05
p.transform	function. Default = \(x) -log10(x)
p.adjust.method	Character: p-value adjustment method. "holm", "hochberg", "hommel", "bonferroni", "BH", "BY", "fdr", "none" Default = "holm". Use "none" for raw p-values.
legend	Logical: If TRUE, show legend. Will default to FALSE, if group = NULL, otherwise to TRUE
legend.lo	Character: Legend to annotate significant points below the x.thresh
legend.hi	Character: Legend to annotate significant points above the x.thresh
label.lo	Character: label for low values
label.hi	Character: label for high values
xlab	Character: x-axis label
ylab	Character: y-axis label
margin	Named list of plot margins. Default = list(b = 65, l = 65, t = 50, r = 10, pad = 0)
xlim	Numeric vector, length 2: x-axis limits
ylim	Numeric vector, length 2: y-axis limits
alpha	Numeric: point transparency
hline	Float: If defined, draw a horizontal line at this y value.
hline.col	Color for hline. Default = "#ff0000" (red)
hline.width	Float: Width for hline. Default = 1
hline.dash	Character: Type of line to draw: "solid", "dot", "dash", "longdash", "dashdot", or "longdashdot"
hline.annotate	Character: Text of horizontal line annotation if hline is set
hline.annotation.x	Numeric: x position to place annotation with paper as reference. 0: to the left of the plot area; 1: to the right of the plot area
annotate.col	Color for annotations

Author(s)

E.D. Gennatas

Examples

```
## Not run:
set.seed(2019)
x <- rnormmat(500, 500)
y <- x[, 3] + x[, 5] - x[, 9] + x[, 15] + rnorm(500)
mod <- massGLM(y, x)
dplot3_volcano(mod$summary$`Coefficient y`, mod$summary$`p_value y`)

## End(Not run)
```

---

dplot3_x *Interactive Univariate Plots*

Description

Draw interactive univariate plots using `plotly`

Usage

```
dplot3_x(
  x,
  type = c("density", "histogram"),
  mode = c("overlap", "ridge"),
  group = NULL,
  main = NULL,
  xlab = NULL,
  ylab = NULL,
  col = NULL,
  alpha = 0.75,
  plot.bg = NULL,
  theme = rtTheme,
  palette = rtPalette,
  axes.square = FALSE,
  group.names = NULL,
  font.size = 16,
  font.alpha = 0.8,
  legend = NULL,
  legend.xy = c(0, 1),
  legend.col = NULL,
  legend.bg = "#FFFFFF00",
  legend.border.col = "#FFFFFF00",
  bargap = 0.05,
  bingroup = 1,
  vline = NULL,
  vline.col = theme$fg,
  vline.width = 1,
```

```
  vline.dash = "dot",
  text = NULL,
  text.x = 1,
  text.xref = "paper",
  text.xanchor = "left",
  text.y = 1,
  text.yref = "paper",
  text.yanchor = "top",
  text.col = theme$fg,
  margin = list(b = 65, l = 65, t = 50, r = 10, pad = 0),
  automargin.x = TRUE,
  automargin.y = TRUE,
  zerolines = FALSE,
  histnorm = c("density", "percent", "probability", "probability density"),
  histfunc = c("count", "sum", "avg", "min", "max"),
  hist.n.bins = 20,
  barmode = "overlay",
  ridge.sharex = TRUE,
  ridge.y.labs = FALSE,
  ridge.order.on.mean = TRUE,
  displayModeBar = TRUE,
  width = NULL,
  height = NULL,
  filename = NULL,
  file.width = 500,
  file.height = 500,
  file.scale = 1,
  ...
)
```

Arguments

x	Numeric, vector / data.frame /list: Input. If not a vector, each column of or each element
type	Character: "density" or "histogram"
mode	Character: "overlap", "ridge". How to plot different groups; on the same axes ("overlap"), or on separate plots with the same x-axis ("ridge")
group	Vector: Will be converted to factor; levels define group members. Default = NULL
main	Character: Main title
xlab	Character: x-axis label. Default = NULL
ylab	Character: y-axis label. Default = NULL
col	Color, vector: Color for bars. Default NULL, which will draw colors from `palette`
alpha	Float (0, 1]: Transparency for bar colors. Default = .8
theme	Character: Theme to use: Use `themes()` to get available themes
palette	Character: Name of **rtemis** palette to use. Default = "rtCol1". Only used if `col = NULL`

axes.square	Logical: If TRUE: draw a square plot to fill the graphic device. Default = FALSE. Note: If TRUE, the device size at time of call is captured and height and width are set so as to draw the largest square available. This means that resizing the device window will not automatically resize the plot.
group.names	Character, vector, length = NROW(x): Group names. Default = NULL, which uses rownames(x)
font.size	Float: Font size for all labels. Default = 16
legend	Logical: If TRUE, draw legend. Default = NULL, which will be set to TRUE if x is a list of more than 1 element
legend.xy	Float, vector, length 2: Relative x, y position for legend. Default = c(0, 1), which places the legend top left within the plot area. Set to NULL to place legend top right beside the plot area
legend.col	Color: Legend text color. Default = NULL, determined by theme
bargap	Float: The gap between adjacent histogram bars in plot fraction.
vline	Float, vector: If defined, draw a vertical line at this x value(s). Default = NULL
vline.col	Color for vline. Default = theme$fg
vline.width	Float: Width for vline. Default = 1
vline.dash	Character: Type of line to draw: "solid", "dot", "dash", "longdash", "dashdot", or "longdashdot"
text	Character: If defined, add this text over the plot
text.x	Float: x-coordinate for text
text.xref	Character: "x": text.x refers to plot's x-axis; "paper": text.x refers to plotting area from 0-1
text.xanchor	Character: "auto", "left", "center", "right"
text.yref	Character: "y": text.y refers to plot's y-axis; "paper": text.y refers to plotting area from 0-1
text.yanchor	Character: "auto", "top", "middle", "bottom"
text.col	Color for text. Default = theme$fg
margin	Named list: plot margins.
automargin.x	Logical: If TRUE, automatically set x-axis amrgins
automargin.y	Logical: If TRUE, automatically set y-axis amrgins
zerolines	Logical: If TRUE: draw lines at y = 0.
histnorm	Character: NULL, "percent", "probability", "density", "probability density"
histfunc	Character: "count", "sum", "avg", "min", "max".
hist.n.bins	Integer: Number of bins to use if type = "histogram".
barmode	Character: Type of bar plot to make: "group", "relative", "stack", "overlay". Default = "group". Use "relative" for stacked bars, wich handles negative values correctly, unlike "stack", as of writing.
ridge.sharex	Logical: If TRUE, draw single x-axis when mode = "ridge".
ridge.y.labs	Lofical: If TRUE, show individual y labs when mode = "ridge".
ridge.order.on.mean	Logical: If TRUE, order groups by mean value when mode = "ridge". Turn to FALSE, if, for example, groups are ordered by date or similar.
displayModeBar	Logical: If TRUE, show plotly's modebar

width	Float: Force plot size to this width. Default = NULL, i.e. fill available space
height	Float: Force plot size to this height. Default = NULL, i.e. fill available space
filename	Character: Path to file to save static plot. Default = NULL
file.width	Integer: File width in pixels for when filename is set.
file.height	Integer: File height in pixels for when filename is set.
file.scale	Numeric: If saving to file, scale plot by this number
...	Additional arguments passed to theme

Details

If input is data.frame, non-numeric variables will be removed

Author(s)

E.D. Gennatas

Examples

```
## Not run:
dplot3_x(iris)
dplot3_x(split(iris$Sepal.Length, iris$Species), xlab = "Sepal Length")

## End(Not run)
```

---

dplot3_xy *Interactive Scatter Plots*

Description

Draw interactive scatter plots using plotly

Usage

```
dplot3_xy(
  x,
  y = NULL,
  fit = NULL,
  se.fit = FALSE,
  se.times = 1.96,
  cluster = NULL,
  cluster.params = list(k = 2),
  group = NULL,
  formula = NULL,
  rsq = TRUE,
  mode = "markers",
  order.on.x = NULL,
  main = NULL,
  xlab = NULL,
  ylab = NULL,
```

```
    col = NULL,
    alpha = NULL,
    bg = NULL,
    plot.bg = NULL,
    theme = rtTheme,
    palette = rtPalette,
    axes.square = FALSE,
    group.names = NULL,
    font.size = 16,
    marker.col = NULL,
    marker.size = 8,
    symbol = "circle",
    fit.col = NULL,
    fit.alpha = 0.8,
    fit.lwd = 2.5,
    se.col = NULL,
    se.alpha = 0.4,
    scatter.type = "scatter",
    legend = NULL,
    legend.xy = c(0, 1),
    legend.xanchor = "left",
    legend.yanchor = "auto",
    legend.orientation = "v",
    legend.col = NULL,
    legend.bg = "#FFFFFF00",
    legend.border.col = "#FFFFFF00",
    legend.borderwidth = 0,
    legend.group.gap = 0,
    x.showspikes = FALSE,
    y.showspikes = FALSE,
    spikedash = "solid",
    spikemode = "across",
    spikesnap = "hovered data",
    spikecolor = NULL,
    spikethickness = 1,
    margin = list(b = 65, l = 65, t = 50, r = 10, pad = 0),
    automargin.x = TRUE,
    automargin.y = TRUE,
    xlim = NULL,
    ylim = NULL,
    axes.equal = FALSE,
    diagonal = FALSE,
    diagonal.col = NULL,
    diagonal.alpha = 0.2,
    fit.params = list(),
    vline = NULL,
    vline.col = theme$fg,
    vline.width = 1,
    vline.dash = "dot",
    hline = NULL,
    hline.col = theme$fg,
    hline.width = 1,
```

```
  hline.dash = "dot",
  hovertext = NULL,
  width = NULL,
  height = NULL,
  displayModeBar = TRUE,
  trace = 0,
  filename = NULL,
  file.width = 500,
  file.height = 500,
  file.scale = 1,
  ...
)
```

Arguments

x	Numeric, vector/data.frame/list: x-axis data. If y is NULL and NCOL(x) > 1, first two columns used as x and y, respectively
y	Numeric, vector/data.frame/list: y-axis data
fit	Character: **rtemis** model to calculate y ~ x fit. Options: see modSelect Can also be Logical, which will give a GAM fit if TRUE. If you specify "NLA", the activation function should be passed as a string.
se.fit	Logical: If TRUE, draw the standard error of the fit
se.times	Draw polygon or lines at +/- se.times * standard error of fit. Defaults to 1.96, which corresponds to 95% confidence interval
cluster	Character: Clusterer name. Will cluster data.frame(x, y) and pass result to group. Run clustSelect for options
cluster.params	List: Names list of parameters to pass to the cluster function
group	Vector: will be converted to factor. If data is provided, name of variable, unquoted.
formula	Formula: Provide a formula to be solved using s_NLS. If provided, fit is forced to 'nls'. e.g. y ~ b * m ^ x for a power curve. Note: nls is powerful but is prone to errors and warnings. Use single letters for parameter names, no numbers.
rsq	Logical: If TRUE, print R-squared values in legend if fit is set
mode	Character, vector: "markers", "lines", "markers+lines". Default = "markers"
order.on.x	Logical: If TRUE, order x and y on x. Default = NULL, which becomes TRUE if mode includes lines.
main	Character: Plot title. Default = NULL
xlab	Character: x-axis label. Default = NULL
ylab	Character: y-axis label. Default = NULL
col	Color for markers. Default=NULL, which will draw colors from palette
alpha	Float (0, 1]: Transparency for bar colors. Default = .8
theme	Character: Theme to use: Use themes() to get available themes
palette	Character: Name of **rtemis** palette to use. Default = "rtCol1". Only used if col = NULL
axes.square	Logical: If TRUE: draw a square plot to fill the graphic device. Default = FALSE. Note: If TRUE, the device size at time of call is captured and height and width are set so as to draw the largest square available. This means that resizing the device window will not automatically resize the plot.

group.names	Character, vector, length = NROW(x): Group names. Default = NULL, which uses rownames(x)
font.size	Float: Font size for all labels. Default = 16
marker.col	Color for marker
fit.col	Color: Color of the fit line.
fit.alpha	Float [0, 1]: Transparency for fit line
fit.lwd	Float: Fit line width
se.col	Color for se.fit
se.alpha	Alpha for se.fit
legend	Logical: If TRUE, draw legend. Default = NULL, which will be set to TRUE if there are more than 1 groups, or fit is set
legend.xy	Numeric, vector, length 2: x and y for plotly's legend
legend.xanchor	Character: Legend's x anchor: "left", "center", "right", "auto"
legend.yanchor	Character: Legend's y anchor: "top", "middle", "bottom", "auto"
legend.orientation	"v" or "h" for vertical or horizontal
legend.col	Color: Legend text color. Default = NULL, determined by theme
margin	Named list: plot margins.
automargin.x	Logical: If TRUE, automatically set x-axis amrgins
automargin.y	Logical: If TRUE, automatically set y-axis amrgins
xlim	Float vector, length 2: x-axis limits
ylim	Float, vector, length 2: y-axis limits.
axes.equal	Logical: Should axes be equal? Defaults to FALSE
diagonal	Logical: If TRUE, draw diagonal line. Default = FALSE
diagonal.col	Color: Color for diagonal. Default = "gray50"
diagonal.alpha	Float: Alpha for diagonal Default = .5
fit.params	List: Arguments for learner defined by fit. Default = NULL, i.e. use default learner parameters
vline	Vector: x-value(s) for vertical lines. Default = NULL
vline.col	Color for vertical lines
hline	Float: If defined, draw a horizontal line at this y value.
hline.col	Color for hline. Default = "#ff0000" (red)
hline.width	Float: Width for hline. Default = 1
hline.dash	Character: Type of line to draw: "solid", "dot", "dash", "longdash", "dashdot", or "longdashdot"
hovertext	List of character vectors with hovertext to include for each group of markers
width	Float: Force plot size to this width. Default = NULL, i.e. fill available space
height	Float: Force plot size to this height. Default = NULL, i.e. fill available space
displayModeBar	Logical: If TRUE, show plotly's modebar
trace	Integer: The height the number the more diagnostic info is printed to the console
filename	Character: Path to file to save static plot. Default = NULL
file.width	Integer: File width in pixels for when filename is set.
file.height	Integer: File height in pixels for when filename is set.
file.scale	Numeric: If saving to file, scale plot by this number
...	Additional arguments passed to theme

Details

use theme$tick.labels.col for both tick color and tick label color - this may change

Author(s)

E.D. Gennatas

Examples

```
## Not run:
dplot3_xy(iris$Sepal.Length, iris$Petal.Length,
    fit = "gam", se.fit = TRUE, group = iris$Species
)

## End(Not run)
```

---

dplot3_xyz *Interactive 3D Plots*

Description

Draw interactive 3D plots using `plotly`

Usage

```
dplot3_xyz(
  x,
  y = NULL,
  z = NULL,
  fit = NULL,
  cluster = NULL,
  cluster.params = list(k = 2),
  group = NULL,
  formula = NULL,
  rsq = TRUE,
  mode = "markers",
  order.on.x = NULL,
  main = NULL,
  xlab = NULL,
  ylab = NULL,
  zlab = NULL,
  col = NULL,
  alpha = 0.8,
  bg = NULL,
  plot.bg = NULL,
  theme = rtTheme,
  palette = rtPalette,
  axes.square = FALSE,
  group.names = NULL,
  font.size = 16,
```

```
  marker.col = NULL,
  marker.size = 8,
  fit.col = NULL,
  fit.alpha = 0.7,
  fit.lwd = 2.5,
  tick.font.size = 12,
  spike.col = NULL,
  legend = NULL,
  legend.xy = c(0, 1),
  legend.xanchor = "left",
  legend.yanchor = "auto",
  legend.orientation = "v",
  legend.col = NULL,
  legend.bg = "#FFFFFF00",
  legend.border.col = "#FFFFFF00",
  legend.borderwidth = 0,
  legend.group.gap = 0,
  margin = list(t = 30, b = 0, l = 0, r = 0),
  zerolines = TRUE,
  fit.params = list(),
  width = NULL,
  height = NULL,
  padding = 0,
  displayModeBar = TRUE,
  trace = 0,
  filename = NULL,
  file.width = 500,
  file.height = 500,
  file.scale = 1,
  ...
)
```

Arguments

x	Numeric, vector/data.frame/list: x-axis data. If y is NULL and NCOL(x) > 1, first two columns used as x and y, respectively
y	Numeric, vector/data.frame/list: y-axis data
z	Numeric, vector/data.frame/list: z-axis data
fit	Character: **rtemis** model to calculate y ~ x fit. Options: see modSelect Can also be Logical, which will give a GAM fit if TRUE. If you specify "NLA", the activation function should be passed as a string.
cluster	Character: Clusterer name. Will cluster data.frame(x, y) and pass result to group. Run clustSelect for options
cluster.params	List: Names list of parameters to pass to the cluster function
group	Vector: will be converted to factor. If data is provided, name of variable, unquoted.
formula	Formula: Provide a formula to be solved using s_NLS. If provided, fit is forced to 'nls'. e.g. y ~ b * m ^ x for a power curve. Note: nls is powerful but is prone to errors and warnings. Use single letters for parameter names, no numbers.
rsq	Logical: If TRUE, print R-squared values in legend if fit is set

mode	Character, vector: "markers", "lines", "markers+lines". Default = "markers"
order.on.x	Logical: If TRUE, order x and y on x. Default = NULL, which becomes TRUE if mode includes lines.
main	Character: Plot title. Default = NULL
xlab	Character: x-axis label. Default = NULL
ylab	Character: y-axis label. Default = NULL
zlab	Character: z-axis label
col	Color for markers. Default=NULL, which will draw colors from palette
alpha	Float (0, 1]: Transparency for bar colors. Default = .8
theme	Character: Theme to use: Use themes() to get available themes
palette	Character: Name of **rtemis** palette to use. Default = "rtCol1". Only used if col = NULL
axes.square	Logical: If TRUE: draw a square plot to fill the graphic device. Default = FALSE. Note: If TRUE, the device size at time of call is captured and height and width are set so as to draw the largest square available. This means that resizing the device window will not automatically resize the plot.
group.names	Character, vector, length = NROW(x): Group names. Default = NULL, which uses rownames(x)
font.size	Float: Font size for all labels. Default = 16
marker.col	Color for marker
fit.col	Color: Color of the fit line.
fit.alpha	Float [0, 1]: Transparency for fit line
fit.lwd	Float: Fit line width
legend	Logical: If TRUE, draw legend. Default = NULL, which will be set to TRUE if there are more than 1 groups, or fit is set
legend.xy	Numeric, vector, length 2: x and y for plotly's legend
legend.xanchor	Character: Legend's x anchor: "left", "center", "right", "auto"
legend.yanchor	Character: Legend's y anchor: "top", "middle", "bottom", "auto"
legend.orientation	"v" or "h" for vertical or horizontal
legend.col	Color: Legend text color. Default = NULL, determined by theme
margin	Numeric, named list: Margins for top, bottom, left, right. Default = list(t = 30, b = 0, l = 0, r = 0
fit.params	List: Arguments for learner defined by fit. Default = NULL, i.e. use default learner parameters
width	Float: Force plot size to this width. Default = NULL, i.e. fill available space
height	Float: Force plot size to this height. Default = NULL, i.e. fill available space
displayModeBar	Logical: If TRUE, show plotly's modebar
trace	Integer: The height the number the more diagnostic info is printed to the console
filename	Character: Path to file to save static plot. Default = NULL
file.width	Integer: File width in pixels for when filename is set.
file.height	Integer: File height in pixels for when filename is set.
file.scale	Numeric: If saving to file, scale plot by this number
...	Additional arguments passed to theme

Details

Note that dplot3_xyz uses the theme's plot.bg as grid.col

Author(s)

E.D. Gennatas

Examples

```
## Not run:
dplot3_xyz(iris, group = iris$Species, theme = "darkgrid")

## End(Not run)
```

---

drange *Set Dynamic Range*

Description

rtemis preproc: Adjusts the dynamic range of a vector or matrix input. By default normalizes to 0-1 range.

Usage

```
drange(x, lo = 0, hi = 1, byCol = TRUE)
```

Arguments

x	Numeric vector or matrix / data frame: Input
lo	Target range minimum. Defaults to 0
hi	Target range maximum. Defaults to 1
byCol	Logical: If TRUE: if x is matrix, drange each column separately

Author(s)

E.D. Gennatas

Examples

```
x <- runif(20, -10, 10)
x <- drange(x)
```

d_CUR *CUR Decomposition*

Description

Performs CUR decomposition using `rCUR::CUR`

Usage

```
d_CUR(
  x,
  c = dim(x)[2],
  r = dim(x)[1],
  k = NULL,
  sv = NULL,
  method = "highest.ranks",
  matrix.return = TRUE,
  error.return = FALSE,
  scale = TRUE,
  center = TRUE,
  verbose = TRUE,
  ...
)
```

Arguments

`x`	Numeric matrix / data.frame: Input data
`c`	Integer: Number of columns to be selected
`r`	Integer: Number of rows to be selected
`k`	Integer: Rank of decomposition (Creates `k` components)
`sv`	The SVD of `x`, if already available
`method`	Character: "random", "top.scores", "ortho.top.scores", "exact.num.random", "highest.ranks" (Default). See `rCUR::CUR("method")`
`matrix.return`	Logical: if TRUE, matrices C, U, and R are returned. If FALSE, U is not computed, which can be expensive, if `r` and `c` are large. Default = TRUE
`error.return`	Logical: if TRUE, the Frobenius norm of the difference between the original matrix and the CUR approximation is returned. Effective only if `matrix.return = TRUE`. Default = FALSE
`scale`	Logical: If TRUE, scale input
`center`	Logical: If TRUE, center input
`verbose`	Logical: If TRUE, print messages to output
`...`	Additional parameters to be passed to `rCUR::CUR`

Details

Note that `k` here does not correspond with `k` in the other decomposition functions. Use `c` to determine dimensionality of resulting decomposition

Value

rtDecom object

Author(s)

E.D. Gennatas

See Also

Other Decomposition: d_H2OAE(), d_H2OGLRM(), d_ICA(), d_ISOMAP(), d_KPCA(), d_LLE(), d_MDS(), d_NMF(), d_PCA(), d_SPCA(), d_SVD(), d_TSNE(), d_UMAP()

---

d_H2OAE *Autoencoder using H2O*

Description

Train an Autoencoder using h2o::h2o.deeplearning Check out the H2O Flow at [ip]:[port], Default IP:port is "localhost:54321" e.g. if running on localhost, point your web browser to localhost:54321

Usage

```
d_H2OAE(
  x,
  x.test = NULL,
  x.valid = NULL,
  ip = "localhost",
  port = 54321,
  n.hidden.nodes = c(ncol(x), 3, ncol(x)),
  extract.layer = ceiling(length(n.hidden.nodes)/2),
  epochs = 5000,
  activation = "Tanh",
  loss = "Automatic",
  input.dropout.ratio = 0,
  hidden.dropout.ratios = rep(0, length(n.hidden.nodes)),
  learning.rate = 0.005,
  learning.rate.annealing = 1e-06,
  l1 = 0,
  l2 = 0,
  stopping.rounds = 50,
  stopping.metric = "AUTO",
  scale = TRUE,
  center = TRUE,
  n.cores = rtCores,
  verbose = TRUE,
  save.mod = FALSE,
  outdir = NULL,
  ...
)
```

Arguments

x	Vector / Matrix / Data Frame: Training set Predictors
x.test	Vector / Matrix / Data Frame: Testing set Predictors
x.valid	Vector / Matrix / Data Frame: Validation set Predictors
ip	Character: IP address of H2O server. Default = "localhost"
port	Integer: Port number for server. Default = 54321
n.hidden.nodes	Integer vector of length equal to the number of hidden layers you wish to create
extract.layer	Integer: Which layer to extract. For regular autoencoder, this is the middle layer. Default = ceiling(length(n.hidden.nodes)/2)
epochs	Integer: How many times to iterate through the dataset. Default = 5000
activation	Character: Activation function to use: "Tanh" (Default), "TanhWithDropout", "Rectifier", "RectifierWithDropout", "Maxout", "MaxoutWithDropout"
loss	Character: "Automatic" (Default), "CrossEntropy", "Quadratic", "Huber", "Absolute"
input.dropout.ratio	Float (0, 1): Dropout ratio for inputs
hidden.dropout.ratios	Vector, Float (0, 2): Dropout ratios for hidden layers
learning.rate	Float: Learning rate. Default = .005
learning.rate.annealing	Float: Learning rate annealing. Default = 1e-06
l1	Float (0, 1): L1 regularization (introduces sparseness; i.e. sets many weights to 0; reduces variance, increases generalizability)
l2	Float (0, 1): L2 regularization (prevents very large absolute weights; reduces variance, increases generalizability)
stopping.rounds	Integer: Stop if simple moving average of length stopping.rounds of the stopping.metric does not improve. Set to 0 to disable. Default = 50
stopping.metric	Character: Stopping metric to use: "AUTO", "deviance", "logloss", "MSE", "RMSE", "MAE", "RMSLE", "AUC", "lift_top_group", "misclassification", "mean_per_class_error". Default = "AUTO" ("logloss" for Classification, "deviance" for Regression)
scale	Logical: If TRUE, scale input before training autoencoder. Default = TRUE
center	Logical: If TRUE, center input before training autoencoder. Default = TRUE
n.cores	Integer: Number of cores to use
verbose	Logical: If TRUE, print summary to screen.
save.mod	Logical: If TRUE, save all output to an RDS file in outdir save.mod is TRUE by default if an outdir is defined. If set to TRUE, and no outdir is defined, outdir defaults to paste0("./s.", mod.name)
outdir	Path to output directory. If defined, will save Predicted vs. True plot, if available, as well as full model output, if save.mod is TRUE
...	Additional arguments to pass to h2p::h2o.deeplearning

Value

rtDecom object

Author(s)

E.D. Gennatas

See Also

decom

Other Decomposition: d_CUR(), d_H2OGLRM(), d_ICA(), d_ISOMAP(), d_KPCA(), d_LLE(), d_MDS(), d_NMF(), d_PCA(), d_SPCA(), d_SVD(), d_TSNE(), d_UMAP()

Other Deep Learning: s_H2ODL(), s_TFN()

---

d_H2OGLRM *Generalized Low-Rank Models (GLRM) on H2O*

Description

Perform GLRM decomposition using h2o::h2o.glrm Given Input matrix A: A(m x n) = X(m x k) %*% Y(k x n)

Usage

```
d_H2OGLRM(
  x,
  x.test = NULL,
  x.valid = NULL,
  k = 3,
  ip = "localhost",
  port = 54321,
  transform = "NONE",
  loss = "Quadratic",
  regularization.x = "None",
  regularization.y = "None",
  gamma.x = 0,
  gamma.y = 0,
  max_iterations = 1000,
  max_updates = 2 * max_iterations,
  init_step_size = 1,
  min_step_size = 1e-04,
  seed = -1,
  init = "PlusPlus",
  svd.method = "Randomized",
  verbose = TRUE,
  print.plot = TRUE,
  plot.theme = rtTheme,
  n.cores = rtCores,
  ...
)
```

Arguments

x	Input data
x.test	Optional test set. Will be projected on to NMF basis
x.valid	Optional validation set
k	Integer: Rank of decomposition
ip	Character: IP address of H2O server. Default = "localhost"
port	Integer: Port number for server. Default = 54321
transform	Character: Transformation of input prior to decomposition
loss	Character: Numeric loss function: "Quadratic", "Absolute", "Huber", "Poisson", "Hinge", "Logistic", "Periodic". Default = "Quadratic"
regularization.x	Character: Regularization function for X matrix: "None", "Quadratic", "L2", "L1", "NonNegative", "OneSparse", "UnitOneSparse", "Simplex". Default = "None"
regularization.y	Character: Regularization function for Y matrix: "None", "Quadratic", "L2", "L1", "NonNegative", "OneSparse", "UnitOneSparse", "Simplex". Default = "None"
gamma.x	Float: Regularization weight on X matrix. Default = 0
gamma.y	Float: Regularization weight on Y matrix. Default = 0
max_iterations	Integer: Maximum number of iterations. Default = 1000
max_updates	Integer: Maximum number of iterations. Default = 2 * max_iterations
init_step_size	Float: Initial step size. Default = 1
min_step_size	Float: Minimum step size. Default = .0001
seed	Integer: Seed for random number generator. Default = -1 (time-based)
init	Character: Initialization mode: "Random", "SVD", "PlusPlus", "User". Default = "PlusPlus"
svd.method	Character: SVD method for initialization: "GramSVD", "Power", "Randomized". Default = "Randomized"
verbose	Logical: If TRUE, print console messages
print.plot	Logical: If TRUE, print objective score against iteration number
plot.theme	Character: Theme to pass to mplot3_xy if print.plot = TRUE
n.cores	Integer: Number of cores to use
...	Additional parameters to be passed to h2o::h2o.glrm

Details

Learn more about GLRM from the H2O tutorial https://github.com/h2oai/h2o-tutorials/blob/master/tutorials/glrm/glrm-tutorial.md

Value

rtDecom object

Author(s)

E.D. Gennatas

See Also

Other Decomposition: d_CUR(), d_H2OAE(), d_ICA(), d_ISOMAP(), d_KPCA(), d_LLE(), d_MDS(), d_NMF(), d_PCA(), d_SPCA(), d_SVD(), d_TSNE(), d_UMAP()

---

d_ICA *Independent Component Analysis*

Description

Perform ICA decomposition using the fastICA algorithm in fastICA::fastICA or ica::fastica

Usage

```
d_ICA(
  x,
  k = 3,
  package = c("fastICA", "ica"),
  alg.type = "parallel",
  maxit = 100,
  scale = TRUE,
  center = TRUE,
  verbose = TRUE,
  trace = 0,
  ...
)
```

Arguments

x	Input data
k	Integer vector of length 1 or greater. Rank of decomposition
package	Character: Which package to use for ICA. "fastICA" will use fastICA::fastICA, "ica" will use ica::fastica. Default = "fastICA". Note: only fastICA works with k = 1
alg.type	Character: For package = "fastICA", "parallel" or "deflation".
maxit	Integer: Maximum N of iterations
scale	Logical: If TRUE, scale input data before decomposition.
center	Logical: If TRUE, also center input data if scale is TRUE.
verbose	Logical: If TRUE, print messages to screen. Default = TRUE
trace	Integer: If > 0, print messages during ICA run. Default = 0
...	Additional parameters to be passed to fastICA::fastICA or ica::icafast

Details

Project scaled variables to ICA components. Input must be n by p, where n represents number of cases, and p represents number of features. fastICA will be applied to the transpose of the n x p matrix. fastICA will fail if there are any NA values or constant features: remove them using preprocess

Value

rtDecom object

Author(s)

E.D. Gennatas

See Also

Other Decomposition: d_CUR(), d_H2OAE(), d_H2OGLRM(), d_ISOMAP(), d_KPCA(), d_LLE(), d_MDS(), d_NMF(), d_PCA(), d_SPCA(), d_SVD(), d_TSNE(), d_UMAP()

---

d_ISOMAP *Isomap*

Description

Perform ISOMAP decomposition using vegan::isomap

Usage

```
d_ISOMAP(
  x,
  k = 2,
  dist.method = "euclidean",
  nsd = 0,
  path = c("shortest", "extended"),
  center = TRUE,
  scale = TRUE,
  verbose = TRUE,
  n.cores = rtCores,
  ...
)
```

Arguments

x	Input data
k	Integer vector of length 1 or greater. Rank of decomposition
dist.method	Character: Distance calculation method. See vegan::vegdist
nsd	Integer: Number of shortest dissimilarities retained
path	Character: The path argument of vegan::isomap
center	Logical: If TRUE, center data prior to decomposition. Default = TRUE
scale	Logical: If TRUE, scale data prior to decomposition. Default = TRUE
verbose	Logical: If TRUE, print messages to output
n.cores	Integer: Number of cores to use
...	Additional parameters to be passed to vegan::isomap

Details

Project scaled variables to ISOMAP components Input must be n by p, where n represents number of cases, and p represents number of features. ISOMAP will be applied to the transpose of the n x p matrix.

Value

rtDecom object

Author(s)

E.D. Gennatas

See Also

Other Decomposition: d_CUR(), d_H2OAE(), d_H2OGLRM(), d_ICA(), d_KPCA(), d_LLE(), d_MDS(), d_NMF(), d_PCA(), d_SPCA(), d_SVD(), d_TSNE(), d_UMAP()

---

d_KPCA *Kernel Principal Component Analysis*

Description

Perform kernel PCA decomposition using `kernlab::kpca`

Usage

```
d_KPCA(
  x,
  x.test = NULL,
  k = 2,
  th = 1e-04,
  kernel = "rbfdot",
  kpar = NULL,
  center = TRUE,
  scale = TRUE,
  verbose = TRUE,
  ...
)
```

Arguments

x	Input data
x.test	Optional test set. Will be projected on to KPCA basis
k	Integer vector of length 1 or greater. N of components to return If set to 0, th determines eigenvalue below which PCs are ignored
th	Threshold for eigenvalue below which PCs are ignored if k is set to 0
kernel	Character: Type of kernel to use. See kernlab::kpca
kpar	List of hyperparameters: See kernlab::kpca("kpar")

center	Logical: If TRUE, center data prior to decomposition. Default = TRUE
scale	Logical: If TRUE, scale data prior to decomposition. Default = TRUE
verbose	Logical: If TRUE, print messages to screen. Default = TRUE
...	Additional parameters to be passed to fastKPCA::fastKPCA

Details

Project scaled variables to KPCA components. Input must be n by p, where n represents number of cases, and p represents number of features. KPCA will be applied to the transpose of the n x p matrix.

Value

rtDecom object

Author(s)

E.D. Gennatas

See Also

Other Decomposition: d_CUR(), d_H2OAE(), d_H2OGLRM(), d_ICA(), d_ISOMAP(), d_LLE(), d_MDS(), d_NMF(), d_PCA(), d_SPCA(), d_SVD(), d_TSNE(), d_UMAP()

---

d_LLE *Locally Linear Embedding*

Description

Perform LLE decomposition using lle::lle

Usage

```
d_LLE(
  x,
  k = 2,
  nn = 0,
  nn.min = 1,
  nn.max = 20,
  plot.calcnn = FALSE,
  id = FALSE,
  iLLE = FALSE,
  nnk = TRUE,
  reg = 2,
  v = 0.9,
  verbose = TRUE,
  n.cores = 1,
  ...
)
```

Arguments

x	Input data
k	Integer vector of length 1 or greater. Rank of decomposition
nn	Integer: Number of neighbors. If Set to 0 (default), will use `lle::calc_k` to estimate optimal number
nn.min	Integer: Minimum n of neighbors to consider in search, used if `nn = 0`
nn.max	Integer: Maximum n of enighbors to consider in search, used if `nn = 0`
plot.calcnn	Logical: If TRUE, print plot after estimation of number of neighbors. Default = FALSE
id	Logical: If TRUE, calculate k (the intrinsic dimension)
iLLE	Logical: If TRUE, use the improved LLE algorithm; see Details in `lle::lle` Notice: It causes warnings for matrix dimensions (check `lle` code)
nnk	Logical: If TRUE, use k nearest neighbors method; otherwise, epsilon environment neighbourhood will be used
reg	Integer 1, 2, 3: Regularization methods: See `lle::lle("reg")`
v	Float: Threshold value for intrinsic dimension estimation. Suggestion for noiseless data: .99, for noisy data: .9. Default = .9
verbose	Logical: If TRUE, print messages to screen. Default = TRUE
n.cores	Integer: Number of cores to use. Default = 1. At some point using more than one cores stopped working. The `lle` package has not been updated since February 2015 - we will switch to a different implementation soon
...	Additional parameters to be passed to `LLE::LLE`

Details

Project scaled variables to LLE components Input must be n by p, where n represents number of cases, and p represents number of features. LLE will be applied to the transpose of the n x p matrix.

Value

rtDecom object

Author(s)

E.D. Gennatas

See Also

Other Decomposition: `d_CUR()`, `d_H2OAE()`, `d_H2OGLRM()`, `d_ICA()`, `d_ISOMAP()`, `d_KPCA()`, `d_MDS()`, `d_NMF()`, `d_PCA()`, `d_SPCA()`, `d_SVD()`, `d_TSNE()`, `d_UMAP()`

d_MDS *Multidimensional Scaling*

Description

Perform MDS decomposition using `stats:cmdscale`

Usage

```
d_MDS(
  x,
  k = 2,
  dist.method = c("euclidean", "maximum", "manhattan", "canberra", "binary",
    "minkowski"),
  eig = FALSE,
  add = FALSE,
  x.ret = FALSE,
  scale = TRUE,
  center = TRUE,
  verbose = TRUE,
  ...
)
```

Arguments

`x`	Input data
`k`	Integer vector of length 1 or greater. Rank of decomposition
`dist.method`	Character: method to use to calculate distance. See `stats::dist("method")`
`eig`	Logical: If TRUE, return eigenvalues. Default = FALSE
`add`	Logical: If TRUE, an additive constant `c*` will be computed and added to the non-diagonal dissimilarities, which makes the Euclidean. Default = FALSE
`x.ret`	Logical: If TRUE, return the doubly centered symmetric distance matrix. Default = FALSE
`scale`	Logical: If TRUE, scale input data before decomposition. Default = TRUE
`center`	Logical: If TRUE, also center input data if `scale` is `TRUE`. Default = TRUE
`verbose`	Logical: If TRUE, print messages to screen. Default = TRUE
`...`	Additional parameters to be passed to `svd`

Details

Project scaled variables to MDS components. Input must be n by p, where n represents number of cases, and p represents number of features. fastMDS will be applied to the transpose of the n x p matrix. fastMDS will fail if there are any NA values or constant features: remove them using preprocess

Value

rtDecom object

Author(s)

E.D. Gennatas

See Also

Other Decomposition: d_CUR(), d_H2OAE(), d_H2OGLRM(), d_ICA(), d_ISOMAP(), d_KPCA(), d_LLE(), d_NMF(), d_PCA(), d_SPCA(), d_SVD(), d_TSNE(), d_UMAP()

---

d_NMF *Non-negative Matrix Factorization (NMF)*

Description

Perform NMF decomposition using NMF::nmf

Usage

```
d_NMF(
  x,
  x.test = NULL,
  k = 2,
  method = "brunet",
  nrun = 30,
  scale = TRUE,
  center = FALSE,
  verbose = TRUE,
  ...
)
```

Arguments

x	Input data
x.test	Optional test set. Will be projected on to NMF basis
k	Integer vector of length 1 or greater. Rank of decomposition
method	NMF method. Defaults to "brunet". See NMF::nmf
nrun	Integer: Number of runs to perform
scale	Logical: If TRUE, scale input data before projecting
center	Logical: If TRUE, also center input data if scale is TRUE
verbose	Logical: If TRUE, print messages to screen. Default = TRUE
...	Additional parameters to be passed to NMF::nmf

Details

Project scaled variables to NMF bases. Input must be n by p, where n represents number of cases, and p represents number of features. NMF will be applied to the transpose of the n x p matrix.

Value

rtDecom object

Author(s)

E.D. Gennatas

See Also

Other Decomposition: d_CUR(), d_H2OAE(), d_H2OGLRM(), d_ICA(), d_ISOMAP(), d_KPCA(), d_LLE(), d_MDS(), d_PCA(), d_SPCA(), d_SVD(), d_TSNE(), d_UMAP()

---

d_PCA *Principal Component Analysis*

Description

Perform PCA decomposition using `stats::prcomp`

Usage

```
d_PCA(
  x,
  x.test = NULL,
  k = NULL,
  scale = TRUE,
  center = TRUE,
  verbose = TRUE,
  ...
)
```

Arguments

x	Input matrix
x.test	Optional test set. Will be projected on to PCA basis
k	Integer: Number of right singular vectors to compute (svd's nv)
scale	Logical: If TRUE, scale input data before doing SVD
center	Logical: If TRUE, also center input data if scale is TRUE
verbose	Logical: If TRUE, print messages to screen. Default = TRUE
...	Additional parameters to be passed to PCA::PCA

Details

Same solution as d_SVD. d_PCA runs prcomp, which has useful summary output

Value

rtDecom object

Author(s)

E.D. Gennatas

See Also

Other Decomposition: d_CUR(), d_H2OAE(), d_H2OGLRM(), d_ICA(), d_ISOMAP(), d_KPCA(), d_LLE(), d_MDS(), d_NMF(), d_SPCA(), d_SVD(), d_TSNE(), d_UMAP()

---

d_SPCA *Sparse Principal Component Analysis*

Description

Perform sparse and/or non-negative PCA or cumulative PCA decomposition using nsprcomp::nsprcomp or nsprcomp::nscumcomp respectively

Usage

```
d_SPCA(
  x,
  x.test = NULL,
  k = 1,
  nz = floor(0.5 * NCOL(x)),
  nneg = FALSE,
  gamma = 0,
  method = c("cumulative", "vanilla"),
  scale = TRUE,
  center = TRUE,
  verbose = TRUE,
  ...
)
```

Arguments

x	Input matrix
x.test	Optional test set. Will be projected on to SPCA basis
k	Integer vector of length 1 or greater. N of components to return If set to 0, th determines eigenvalue below which PCs are ignored
nz	Integer: Upper bound on non-zero loadings. See nsprcomp::nscumcomp("k")
nneg	Logical: If TRUE, calculate non-negative loadings only. Default = FALSE
gamma	Float (>0): Penalty on the divergence from otrhonormality of the pseudo-rotation matrix. Default = 0, i.e. no penalty. May need to increase with collinear features.
method	Character: "cumulative" or "vanilla" sparse PCA. Default = "cumulative"
scale	Logical: If TRUE, scale input data before projecting. Default = TRUE
center	Logical: If TRUE, also center input data if scale is TRUE. Default = FALSE
verbose	Logical: If TRUE, print messages to screen. Default = TRUE
...	Additional parameters to be passed to fastSPCA::fastSPCA

Details

Project scaled variables to sparse and/or non-negative PCA components. Input must be n by p, where n represents number of cases, and p represents number of features. SPCA will be applied to the transpose of the n x p matrix.

Value

rtDecom object

Author(s)

E.D. Gennatas

See Also

Other Decomposition: d_CUR(), d_H2OAE(), d_H2OGLRM(), d_ICA(), d_ISOMAP(), d_KPCA(), d_LLE(), d_MDS(), d_NMF(), d_PCA(), d_SVD(), d_TSNE(), d_UMAP()

---

d_SVD *Singular Value Decomposition*

Description

Perform SVD decomposition using base::svd

Usage

```
d_SVD(
  x,
  x.test = NULL,
  k = 2,
  nu = 0,
  scale = TRUE,
  center = TRUE,
  verbose = TRUE,
  ...
)
```

Arguments

x	Input matrix
x.test	Optional test set matrix. Will be projected on to SVD bases
k	Integer: Number of right singular vectors to compute (svd's nv)
nu	Integer: Number of left singular vectors to compute
scale	Logical: If TRUE, scale input data before doing SVD. Default = TRUE
center	Logical: If TRUE, also center input data if scale is TRUE. Default = TRUE
verbose	Logical: If TRUE, print messages to screen. Default = TRUE
...	Additional parameters to be passed to svd

Details

Same solution as d_PCA

Value

rtDecom object

Author(s)

E.D. Gennatas

See Also

Other Decomposition: `d_CUR()`, `d_H2OAE()`, `d_H2OGLRM()`, `d_ICA()`, `d_ISOMAP()`, `d_KPCA()`, `d_LLE()`, `d_MDS()`, `d_NMF()`, `d_PCA()`, `d_SPCA()`, `d_TSNE()`, `d_UMAP()`

---

d_TSNE *t-distributed Stochastic Neighbor Embedding*

Description

Perform t-SNE decomposition using `Rtsne::Rtsne`

Usage

```
d_TSNE(
  x,
  k = 3,
  initial.dims = 50,
  perplexity = 15,
  theta = 0,
  check.duplicates = TRUE,
  pca = TRUE,
  max.iter = 1000,
  scale = FALSE,
  center = FALSE,
  is.distance = FALSE,
  verbose = TRUE,
  outdir = "./",
  ...
)
```

Arguments

x	Input matrix
k	Integer. Number of t-SNE components required
initial.dims	Integer: Number of dimensions to retain in initial PCA. Default = 50
perplexity	Numeric: Perplexity parameter
theta	Float: 0.0: exact TSNE. Increase for higher speed, lower accuracy. Default = 0

check.duplicates	Logical: If TRUE, Checks whether duplicates are present. Best to set test manually
pca	Logical: If TRUE, perform initial PCA step. Default = TRUE
max.iter	Integer: Maximum number of iterations. Default = 1000
scale	Logical: If TRUE, scale before running t-SNE using base::scale. Default = FALSE
center	Logical: If TRUE, and scale = TRUE, also center. Default = FALSE
is.distance	Logical: If TRUE, x should be a distance matrix. Default = FALSE
verbose	Logical: If TRUE, print messages to output
outdir	Path to output directory
...	Options for Rtsne::Rtsne

Value

rtDecom object

Author(s)

E.D. Gennatas

See Also

Other Decomposition: d_CUR(), d_H2OAE(), d_H2OGLRM(), d_ICA(), d_ISOMAP(), d_KPCA(), d_LLE(), d_MDS(), d_NMF(), d_PCA(), d_SPCA(), d_SVD(), d_UMAP()

---

d_UMAP *Uniform Manifold Approximation and Projection*

Description

Perform UMAP decomposition using github package jlmelville/uwot

Usage

```
d_UMAP(
  x,
  x.test = NULL,
  k = 2,
  n.neighbors = 15,
  init = "spectral",
  metric = c("euclidean", "cosine", "manhattan", "hamming", "categorical"),
  epochs = NULL,
  learning.rate = 1,
  scale = TRUE,
  verbose = TRUE,
  ...
)
```

Arguments

x	Input matrix
x.test	Optional test set matrix. Will be projected on to UMAP bases
k	Integer: Number of projections
n.neighbors	Integer: Number of keighbors
init	Character: Initialization type. See uwot::umap "init"
metric	Character: Distance metric to use: "euclidean", "cosine", "manhattan", "hamming", "categorical". Default = "euclidean"
epochs	Integer: Number of epochs
learning.rate	Float: Learning rate. Default = 1
scale	Logical: If TRUE, scale input data before doing UMAP. Default = TRUE
verbose	Logical: If TRUE, print messages to screen. Default = TRUE
...	Additional parameters to be passed to uwot::umap

Value

rtDecom object

Author(s)

E.D. Gennatas

See Also

Other Decomposition: d_CUR(), d_H2OAE(), d_H2OGLRM(), d_ICA(), d_ISOMAP(), d_KPCA(), d_LLE(), d_MDS(), d_NMF(), d_PCA(), d_SPCA(), d_SVD(), d_TSNE()

---

earlystop *Early stopping*

Description

Check loss vector for early stopping criteria: - either total percent decrease from starting error (e.g. if predictions started at expectation) - or minimum percent decrease (relative to the first value of the vector) over a window of last n steps

Usage

```
earlystop(
  x,
  window = 10,
  window_decrease_pct_min = 0.01,
  total_decrease_pct_max = NULL,
  verbose = TRUE
)
```

Arguments

x	Numeric vector: loss at each iteration
window	Integer: Number of steps to consider
window_decrease_pct_min	Float: Stop if improvement is less than this percent over last ‘window‘ steps
total_decrease_pct_max	Float: Stop if improvement from first to last step exceeds this percent. If defined, overrides ‘window_decrease_pct_min‘
verbose	Logical: If TRUE, print messages to console. Default = TRUE

Details

If the first loss value was set to be the loss when yhat = mean(y) (e.g. in boosting), then `total_decrease_pct_max` corresponds to R-squared and `window_decrease_pct_min` to percent R-squared improvement over `window` last steps.

Author(s)

E.D. Gennatas

`eightball` *Magic 8-Ball*

Description

Magic 8-Ball

Usage

```
eightball(question = NULL)
```

Arguments

question	Character: Your question for the magic 8-ball

Author(s)

E.D. Gennatas

elevate *Tune, Train, and Test an* **rtemis** *Learner by Nested Resampling*

Description

elevate is a high-level function to tune, train, and test an **rtemis** model by nested resampling, with optional preprocessing and decomposition of input features

Usage

```
elevate(
  x,
  y = NULL,
  mod = "ranger",
  mod.params = list(),
  .preprocess = NULL,
  .decompose = NULL,
  .resample = NULL,
  weights = NULL,
  resampler = "strat.sub",
  n.resamples = 10,
  n.repeats = 1,
  stratify.var = NULL,
  train.p = 0.8,
  strat.n.bins = 4,
  target.length = NULL,
  seed = NULL,
  res.index = NULL,
  res.group = NULL,
  bag.fn = median,
  x.name = NULL,
  y.name = NULL,
  save.mods = TRUE,
  save.tune = TRUE,
  cex = 1.4,
  col = "#18A3AC",
  bag.fitted = FALSE,
  outer.n.workers = 1,
  parallel.type = ifelse(.Platform$OS.type == "unix", "fork", "psock"),
  print.plot = TRUE,
  plot.fitted = FALSE,
  plot.predicted = TRUE,
  plot.theme = rtTheme,
  print.res.plot = FALSE,
  question = NULL,
  verbose = TRUE,
  res.verbose = FALSE,
  trace = 0,
  headless = FALSE,
  outdir = NULL,
  save.plots = FALSE,
```

```
  save.rt = ifelse(!is.null(outdir), TRUE, FALSE),
  save.mod = TRUE,
  save.res = FALSE,
  backend = "future",
  debug = FALSE,
  ...
)
```

Arguments

x	Numeric vector or matrix / data frame of features i.e. independent variables
y	Numeric vector of outcome, i.e. dependent variable
mod	Character: Learner to use. Options: modSelect
mod.params	Optional named list of parameters to be passed to mod. All parameters can be passed as part of ... as well
.preprocess	Optional named list of parameters to be passed to preprocess. Set using rtset.preprocess, e.g. decom = rtset.preprocess(impute = TRUE)
.decompose	Optional named list of parameters to be used for decomposition / dimensionality reduction. Set using rtset.decompose, e.g. decom = rtset.decompose("ica", 12)
.resample	Optional named list of parameters to be passed to resample. NOTE: If set, this takes precedence over setting the individual resampling arguments
weights	Numeric vector: Weights for cases. For classification, weights takes precedence over ipw, therefore set weights = NULL if using ipw. Note: If weight are provided, ipw is not used. Leave NULL if setting ipw = TRUE. Default = NULL
resampler	Character: Type of resampling to perform: "bootstrap", "kfold", "strat.boot", "strat.sub". Default = "strat.boot" for length(y) < 200, otherwise "strat.sub"
n.resamples	Integer: Number of training/testing sets required
n.repeats	Integer: Number of times the external resample should be repeated. This allows you to do, for example, 10 times 10-fold crossvalidation. Default = 1. In most cases it makes sense to use 1 repeat of many resamples, e.g. 25 stratified subsamples,
stratify.var	Numeric vector: Used to stratify external sampling (if applicable) Defaults to outcome y
train.p	Float (0, 1): Fraction of cases to assign to traininig set for resampler = "strat.sub"
strat.n.bins	Integer: Number of groups to use for stratification for resampler = "strat.sub" / "strat.boot"
target.length	Integer: Number of cases for training set for resampler = "strat.boot". Default = length(y)
seed	Integer: (Optional) Set seed for random number generator, in order to make output reproducible. See ?base::set.seed
res.index	List where each element is a vector of training set indices. Use this for manual or precalculated train/test splits
res.group	Integer, vector, length = length(y): Integer vector, where numbers define fold membership. e.g. for 10-fold on a dataset with 1000 cases, you could use group = rep(1:10, each = 100)
bag.fn	Function to use to average prediction if bag.fitted = TRUE. Default = median

x.name	Character: Name of predictor dataset
y.name	Character: Name of outcome
save.mods	Logical: If TRUE, retain trained models in object, otherwise discard (save space if running many resamples). Default = TRUE
save.tune	Logical: If TRUE, save the best.tune data frame for each resample (output of gridSearchLearn)
cex	Float: cex parameter for elevate plot
col	Color for elevate plot
bag.fitted	Logical: If TRUE, use all models to also get a bagged prediction on the full sample. To get a bagged prediction on new data using the same models, use predict.rtModCV
outer.n.workers	Integer: Number of cores to use. Default = 1. You are likely parallelizing either in the inner (tuning) or the learner itself is parallelized. Don't parallelize the parallelization
parallel.type	Character: "psock" (Default), "fork"
print.plot	Logical: if TRUE, produce plot using mplot3 Takes precedence over plot.fitted and plot.predicted. Default = TRUE
plot.fitted	Logical: if TRUE, plot True (y) vs Fitted
plot.predicted	Logical: if TRUE, plot True (y.test) vs Predicted. Requires x.test and y.test
plot.theme	Character: "zero", "dark", "box", "darkbox"
print.res.plot	Logical: Print model performance plot for each resample. Defaults to FALSE from all resamples. Defaults to TRUE
question	Character: the question you are attempting to answer with this model, in plain language.
verbose	Logical: If TRUE, print summary to screen.
res.verbose	Logical: Passed to resLearn_future, passed to each individual learner's verbose argument
trace	Integer: (Not really used) Print additional information if > 0. Default = 0
headless	Logical: If TRUE, turn off all plotting.
outdir	Character: Path where output should be saved
save.plots	Logical: If TRUE, save plots to outdir
save.rt	Logical: If TRUE and outdir is set, save all models to outdir
save.mod	Logical: If TRUE, save all output to an RDS file in outdir save.mod is TRUE by default if an outdir is defined. If set to TRUE, and no outdir is defined, outdir defaults to paste0("./s.", mod.name)
save.res	Logical: If TRUE, save the full output of each model trained on differents resamples under subdirectories of outdir
backend	(For testing use only)
debug	Logical: If TRUE, sets outer.n.workers to 1, and options(error=recover)
...	Additional mod.params to be passed to learner (Will be concatenated with mod.params, so that you can use either way to pass learner arguments)

Details

- Note on resampling: You can never use an outer resampling method with replacement if you will also be using an inner resampling (for tuning). The duplicated cases from the outer resampling may appear both in the training and testing sets of the inner resamples, leading to artificially decreased error.

- If there is an error while running either the outer or inner resamples in parallel, the error message returned by R will likely be unhelpful. Repeat the command after setting both inner and outer resample run to use a single core, which should provide an informative message.

Value

Object of class `rtModCV` (Regression) or `rtModCVClass` (Classification)

`error.test.repeats`
: the mean or aggregate error, as appropriate, for each repeat

`error.test.repeats.mean`
: the mean error of all repeats, i.e. the mean of `error.test.repeats`

`error.test.repeats.sd`
: if `n.repeats` > 1, the standard deviation of `error.test.repeats`

`error.test.res`
: the error for each resample, for each repeat

Author(s)

E.D. Gennatas

Examples

```
## Not run:
# Regression

x <- rnormmat(100, 50)
w <- rnorm(50)
y <- x %*% w + rnorm(50)
mod <- elevate(x, y)

# Classification

data(Sonar, package = "mlbench")
mod <- elevate(Sonar)

# Example usage of debug in elevate

# Train on a resample which has no cases for one level
ir <- iris[1:100, ]

# ranger works, but read those warnings!
mod <- elevate(ir)

# rpart fails but you can't tell what's going on
mod <- elevate(ir, mod = "cart")

# Enabling debug helps you find out what's going on where
mod <- elevate(ir, mod = "cart", debug = TRUE)

## End(Not run)
```

`expand.boost` *Expand boosting series*

Description

Expand a boost object by adding more iterations

Usage

```
expand.boost(
  object,
  x,
  y = NULL,
  x.valid = NULL,
  y.valid = NULL,
  x.test = NULL,
  y.test = NULL,
  mod = NULL,
  resid = NULL,
  mod.params = NULL,
  max.iter = 10,
  learning.rate = NULL,
  case.p = 1,
  prefix = NULL,
  verbose = TRUE,
  trace = 0,
  print.error.plot = "final",
  print.plot = FALSE
)
```

Arguments

`object`	boost object
`x`	Numeric vector or matrix / data frame of features i.e. independent variables
`y`	Numeric vector of outcome, i.e. dependent variable
`x.valid`	Data.frame; optional: Validation data
`y.valid`	Float, vector; optional: Validation outcome
`x.test`	Numeric vector or matrix / data frame of testing set features Columns must correspond to columns in `x`
`y.test`	Numeric vector of testing set outcome
`mod`	Character: Algorithm to train base learners, for options, see modSelect. Default = "cart"
`resid`	Float, vector, length = length(y): Residuals to work on. Do not change unless you know what you're doing. Default = NULL, for regular boosting
`mod.params`	Named list of arguments for `mod`
`max.iter`	Integer: Maximum number of iterations (additive steps) to perform. Default = 10
`learning.rate`	Float (0, 1] Learning rate for the additive steps

case.p	Float (0, 1]: Train each iteration using this perceent of cases. Default = 1, i.e. use all cases
prefix	Internal
verbose	Logical: If TRUE, print summary to screen.
trace	Integer: If > 0, print diagnostic info to console
print.error.plot	String or Integer: "final" plots a training and validation (if available) error curve at the end of training. If integer, plot training and validation error curve every this many iterations during training. "none" for no plot.
print.plot	Logical: if TRUE, produce plot using mplot3 Takes precedence over plot.fitted and plot.predicted. Default = TRUE

Author(s)

E.D. Gennatas

---

expand.cartLinBoostTV *Expand boosting series*

Description

Expand a cartLinBoostTV object by adding more iterations

Usage

```
expand.cartLinBoostTV(
  object,
  x,
  y = NULL,
  x.valid = NULL,
  y.valid = NULL,
  x.test = NULL,
  y.test = NULL,
  resid = NULL,
  cart.params = NULL,
  glm.params = NULL,
  mod.params = NULL,
  max.iter = 10,
  learning.rate = NULL,
  weights.p = 1,
  weights.0 = 0,
  seed = NULL,
  prefix = NULL,
  verbose = TRUE,
  trace = 0,
  print.error.plot = "final",
  print.plot = FALSE
)
```

Arguments

object	cartLinBoostTV object
x	Numeric vector or matrix / data frame of features i.e. independent variables
y	Numeric vector of outcome, i.e. dependent variable
x.valid	Data.frame; optional: Validation data
y.valid	Float, vector; optional: Validation outcome
x.test	Numeric vector or matrix / data frame of testing set features Columns must correspond to columns in x
y.test	Numeric vector of testing set outcome
resid	Float, vector, length = length(y): Residuals to work on. Do not change unless you know what you're doing. Default = NULL, for regular boosting
mod.params	Named list of arguments for mod
max.iter	Integer: Maximum number of iterations (additive steps) to perform. Default = 10
learning.rate	Float (0, 1] Learning rate for the additive steps
prefix	Internal
verbose	Logical: If TRUE, print summary to screen.
trace	Integer: If > 0, print diagnostic info to console
print.error.plot	String or Integer: "final" plots a training and validation (if available) error curve at the end of training. If integer, plot training and validation error curve every this many iterations during training. "none" for no plot.
print.plot	Logical: if TRUE, produce plot using mplot3 Takes precedence over plot.fitted and plot.predicted. Default = TRUE

Author(s)

E.D. Gennatas

---

expand.cartLiteBoostTV

Expand boosting series

Description

Expand a cartLiteBoostTV object by adding more iterations

Usage

```
expand.cartLiteBoostTV(
  object,
  x,
  y = NULL,
  x.valid = NULL,
  y.valid = NULL,
  x.test = NULL,
```

```
  y.test = NULL,
  resid = NULL,
  mod.params = NULL,
  max.iter = 10,
  learning.rate = NULL,
  weights.p = 1,
  weights.0 = 0,
  seed = NULL,
  prefix = NULL,
  verbose = TRUE,
  trace = 0,
  print.error.plot = "final",
  print.plot = FALSE
)
```

Arguments

`object`	cartLiteBoostTV object
`x`	Numeric vector or matrix / data frame of features i.e. independent variables
`y`	Numeric vector of outcome, i.e. dependent variable
`x.valid`	Data.frame; optional: Validation data
`y.valid`	Float, vector; optional: Validation outcome
`x.test`	Numeric vector or matrix / data frame of testing set features Columns must correspond to columns in `x`
`y.test`	Numeric vector of testing set outcome
`resid`	Float, vector, length = length(y): Residuals to work on. Do not change unless you know what you're doing. Default = NULL, for regular boosting
`mod.params`	Named list of arguments for `mod`
`max.iter`	Integer: Maximum number of iterations (additive steps) to perform. Default = 10
`learning.rate`	Float (0, 1] Learning rate for the additive steps
`prefix`	Internal
`verbose`	Logical: If TRUE, print summary to screen.
`trace`	Integer: If > 0, print diagnostic info to console
`print.error.plot`	String or Integer: "final" plots a training and validation (if available) error curve at the end of training. If integer, plot training and validation error curve every this many iterations during training. "none" for no plot.
`print.plot`	Logical: if TRUE, produce plot using `mplot3` Takes precedence over `plot.fitted` and `plot.predicted`. Default = TRUE

Author(s)

E.D. Gennatas

`expand.glmLiteBoostTV` *Expand boosting series*

Description

Expand a glmLiteBoostTV object by adding more iterations

Usage

```
expand.glmLiteBoostTV(
  object,
  x,
  y = NULL,
  x.valid = NULL,
  y.valid = NULL,
  x.test = NULL,
  y.test = NULL,
  resid = NULL,
  mod.params = NULL,
  max.iter = 10,
  learning.rate = NULL,
  weights.p = 1,
  weights.0 = 0,
  seed = NULL,
  prefix = NULL,
  verbose = TRUE,
  trace = 0,
  print.error.plot = "final",
  print.plot = FALSE
)
```

Arguments

`object`	glmLiteBoostTV object
`x`	Numeric vector or matrix / data frame of features i.e. independent variables
`y`	Numeric vector of outcome, i.e. dependent variable
`x.valid`	Data.frame; optional: Validation data
`y.valid`	Float, vector; optional: Validation outcome
`x.test`	Numeric vector or matrix / data frame of testing set features Columns must correspond to columns in `x`
`y.test`	Numeric vector of testing set outcome
`resid`	Float, vector, length = length(y): Residuals to work on. Do not change unless you know what you're doing. Default = NULL, for regular boosting
`mod.params`	Named list of arguments for `mod`
`max.iter`	Integer: Maximum number of iterations (additive steps) to perform. Default = 10
`learning.rate`	Float (0, 1] Learning rate for the additive steps
`prefix`	Internal

verbose	Logical: If TRUE, print summary to screen.
trace	Integer: If > 0, print diagnostic info to console
print.error.plot	String or Integer: "final" plots a training and validation (if available) error curve at the end of training. If integer, plot training and validation error curve every this many iterations during training. "none" for no plot.
print.plot	Logical: if TRUE, produce plot using mplot3 Takes precedence over plot.fitted and plot.predicted. Default = TRUE

Author(s)

E.D. Gennatas

---

expand.hytboostnow *Expand boosting series*

Description

Add iterations to a boost object

Usage

```
expand.hytboostnow(
  object,
  x,
  y = NULL,
  x.valid = NULL,
  y.valid = NULL,
  resid = NULL,
  mod.params = NULL,
  max.iter = 10,
  learning.rate = NULL,
  case.p = 1,
  cxrcoef = FALSE,
  prefix = NULL,
  verbose = TRUE,
  trace = 0,
  print.error.plot = "final"
)
```

Arguments

object	boost object
x	Data frame: Input features
y	Vector: Output
mod.params	Named list of arguments for mod
max.iter	Integer: Maximum number of iterations (additive steps) to perform. Default = 10
learning.rate	Float (0, 1] Learning rate for the additive steps

`cxrcoef`	Logical: If TRUE, pass `cxr = TRUE, cxrcoef = TRUE` to predict.hytreenow
`verbose`	Logical: If TRUE, print summary to screen.
`trace`	Integer: If > 0, print diagnostic info to console
`print.error.plot`	String or Integer: "final" plots a training and validation (if available) error curve at the end of training. If integer, plot training and validation error curve every this many iterations during training for each base learner

Author(s)

E.D. Gennatas

`f1` *F1 score*

Description

Calculate the F1 score for classification:

Usage

```
f1(precision, recall)
```

Arguments

`precision`	Float [0, 1]: Precision a.k.a. Positive Predictive Value
`recall`	Float [0, 1]: Recall a.k.a. Sensitivity

Details

$$F1 = 2\frac{Recall \cdot Precision}{Recall + Precision}$$

Author(s)

E.D. Gennatas

factorHarmonize *Factor harmonize*

Description

Factor harmonize

Usage

```
factorHarmonize(reference, x, verbose = TRUE)
```

Arguments

reference	Reference factor
x	Input factor
verbose	Logical: If TRUE, print messages to console. Default = TRUE

factoryze *Factor Analysis*

Description

Perform parallel analysis, factor analysis, bifactor analysis and hierarchical clustering

Usage

```
factoryze(
  x,
  n.factors = NULL,
  method = "minres",
  rotation = "oblimin",
  scores = "regression",
  cor = "cor",
  fa.n.iter = 100,
  omega.method = "minres",
  omega.rotation = c("oblimin", "simplimax", "promax", "cluster", "target"),
  omega.n.iter = 1,
  x.name = NULL,
  print.plot = TRUE,
  do.pa = TRUE,
  do.fa = TRUE,
  do.bifactor = TRUE,
  do.hclust = FALSE,
  verbose = TRUE,
  ...
)
```

Arguments

x	Data. Will be coerced to data frame
n.factors	Integer: If NULL, will be estimated using parallel analysis
method	Character: Factor analysis method: "minres": minimum residual (OLS), "wls": weighted least squares (WLS); "gls": generalized weighted least squares (GLS); "pa": principal factor solution; "ml": maximum likelihood; "minchi": minimize the sample size weighted chi square when treating pairwise correlations with different number of subjects per pair; "minrank": minimum rank factor analysis. Default = "minres"
rotation	Character: Rotation methods. No rotation: "none"; Orthogonal: "varimax", "quartimax", "bentlerT", "equamax", "varimin", "geominT", "bifactor"; Oblique: "promax", "oblimin", "simplimax", "bentlerQ, "geominQ", "biquartimin", "cluster". Default = "oblimin"
scores	Character: Factor score estimation method. Options: "regression", "Thurstone": simple regression, "tenBerge": correlation-preserving, "Anderson", "Barlett". Default = "regression"
cor	Character: Correlation method: "cor": Pearson correlation, "cov": Covariance, "tet": tetrachoric, "poly": polychoric, "mixed": mixed cor for a mixture of tetrachorics, polychorics, Pearsons, biserials, and polyserials, "Yuleb": Yulebonett, "Yuleq" and "YuleY": Yule coefficients
fa.n.iter	Integer: Number of iterations for factor analysis. Default = 100
omega.method	Character: Factor analysis method for the bifactor analysis. Same options as method Default = "minres"
omega.rotation	Character: Rotation method for bifactor analysis: "oblimin", "simplimax", "promax", "cluster", "target". Default = "oblimin"
omega.n.iter	Integer: Number of iterations for bifactor analysis. Default = 1
x.name	Character: Name your dataset. Used for plotting
print.plot	Logical: If TRUE, print plots along the way. Default = TRUE
do.pa	Logical: If TRUE, perform parallel analysis. Default = TRUE
do.fa	Logical: If TRUE, perform factor analysis. Default = TRUE
do.bifactor	Logical: If TRUE, perform bifactor analysis. Default = TRUE
do.hclust	Logical: If TRUE, perform hierarchical cluster analysis. Default = TRUE
verbose	Logical: If TRUE, print messages to output. Default = TRUE
...	Additional arguments to pass to psych::fa

Details

Consult psych::fa for more information on the parameters

Author(s)

E.D. Gennatas

factor_NA2missing *Factor NA to "missing" level*

Description

Set NA values of a factor vector to a new level indicating missingness

Usage

```
factor_NA2missing(x, na_level_name = "missing")
```

Arguments

x	Factor
na_level_name	Character: Name of new level to create that will be assigned to all current NA values. Default = "missing"

Author(s)

E.D. Gennatas

Examples

```
x <- factor(sample(letters[1:3], 100, TRUE))
x[sample(1:100, 10)] <- NA
xm <- factor_NA2missing(x)
```

format.call *Format method for* call *objects*

Description

Format method for call objects

Usage

```
## S3 method for class 'call'
format(x, as.html = FALSE, class = "rtcode", ...)
```

Arguments

x	call object
as.html	Logical: If TRUE, output HTML span element
class	Character: CSS class to assign to span containing code
...	Not used

Author(s)

E.D. Gennatas

Examples

```
## Not run:
irmod <- elevate(iris,
                 mod = "cart",
                 maxdepth = 2:3,
                 n.resamples = 9,
                 train.p = .85)
format(irmod$call) |> cat()

## End(Not run)
```

`formatRules` *Format rules*

Description

Converts R-executable logical expressions to a more human-friendly format

Usage

```
formatRules(x, space.after.comma = FALSE, decimal.places = NULL)
```

Arguments

`x`	Vector, string: Logical expressions
`space.after.comma`	Logical: If TRUE, place spaces after commas. Default = false
`decimal.places`	Integer: Limit all floats (numbers of the form 9.9) to this many decimal places

Author(s)

E.D. Gennatas

`fwhm2sigma` *FWHM to Sigma*

Description

Convert Full width at half maximum values to sigma

Usage

```
fwhm2sigma(fwhm)
```

Arguments

`fwhm`	FWHM value

Value

sigma

Author(s)

E.D. Gennatas

Examples

```
fwhm2sigma(8)
# FWHM of 8 is equivalent to sigma = 3.397287
```

---

get-names *Get factor/numeric/logical/character names from data.frame/data.table*

Description

Get factor/numeric/logical/character names from data.frame/data.table

Usage

```
getfactornames(x)
```

Arguments

x	data.frame or data.table (or data.frame-compatible object)

Author(s)

E.D. Gennatas

---

getMode *Get the mode of a factor or integer*

Description

Returns the mode of a factor or integer

Usage

```
getMode(x, na.exclude = TRUE, getlast = TRUE, retain.class = TRUE)
```

Arguments

x	Vector, factor or integer: Input data
na.exclude	Logical: If TRUE, exclude NAs
getlast	Logical: If TRUE, get
retain.class	Logical: If TRUE, output is always same class as input

Value

The mode of x

Author(s)

E.D. Gennatas

Examples

```
x <- c(9, 3, 4, 4, 0, 2, 2)
getModwe(x)
x <- c(9, 3, 2, 2, 0, 4, 4)
getMode(x)
getMode(x, getlast = FALSE)
```

---

getnames *Get names by string matching*

Description

Get names by string matching

Usage

```
getnames(
  x,
  pattern = NULL,
  starts_with = NULL,
  ends_with = NULL,
  ignore.case = TRUE
)

getnumericnames(x)

getlogicalnames(x)

getcharacternames(x)

getdatenames(x)
```

Arguments

x	object with `names()` method
pattern	Character: pattern to match anywhere in names of x
starts_with	Character: pattern to match in the beginning of names of x
ends_with	Character: pattern to match at the end of names of x
ignore.case	Logical: If TRUE, well, ignore case. Default = TRUE

Author(s)

E.D. Gennatas

getnamesandtypes *Get data.frame names and types*

Description

Get data.frame names and types

Usage

```
getnamesandtypes(x)
```

Arguments

x	data.frame / data.table or similar

Value

character vector of column names with attribute "type" holding the class of each column

getTerms *Get terms of a formula*

Description

Get terms of a formula

Usage

```
getTerms(formula)
```

Arguments

formula	formula with more than x & y, e.g. y ~ b * m ^ x

ggtheme_dark **rtemis** ggplot2 *dark theme*

Description

rtemis ggplot2 dark theme

Usage

```
ggtheme_dark(
  base_size = 14,
  base_family = "Helvetica Neue",
  base_line_size = base_size/22,
  base_rect_size = base_size/22,
  axis.text.size.rel = 1,
  legend.key.fill = NA,
  legend.text.size.rel = 1,
  legend.position = "right",
  strip.background.fill = "gray25"
)
```

Arguments

base_size	Float: Base font size. Default = 14
base_family	Character: Font family. Default = "Helvetica Neue"
base_line_size	Float: Line size. Default = base_size/22
base_rect_size	Float: Size for rect elements. Default = base_size/22
axis.text.size.rel	Float: Relative size for axis text. Default = 1
legend.key.fill	Color: Fill color for legend. Default = NA (no color)
legend.text.size.rel	Float: Relative size for legend text. Default = 1
legend.position	Character: Legend position, "top", "bottom", "right", "left" Default = "right"
strip.background.fill	Color: Fill color from facet labels. Default = "gray25"

Author(s)

E.D. Gennatas

Examples

```
## Not run:
(p <- ggplot(iris, aes(Sepal.Length, Petal.Length, color = Species)) +
geom_point() +
ggtheme_light())

## End(Not run)
```

ggtheme_light **rtemis** ggplot2 *light theme*

Description

rtemis ggplot2 light theme

Usage

```
ggtheme_light(
  base_size = 14,
  base_family = "Helvetica Neue",
  base_line_size = base_size/22,
  base_rect_size = base_size/22,
  axis.text.size.rel = 1,
  legend.key.fill = NA,
  legend.text.size.rel = 1,
  legend.position = "right",
  strip.background.fill = "grey85"
)
```

Arguments

base_size	Float: Base font size. Default = 14
base_family	Character: Font family. Default = "Helvetica Neue"
base_line_size	Float: Line size. Default = base_size/22
base_rect_size	Float: Size for rect elements. Default = base_size/22
axis.text.size.rel	Float: Relative size for axis text. Default = 1
legend.key.fill	Color: Fill color for legend. Default = NA (no color)
legend.text.size.rel	Float: Relative size for legend text. Default = 1
legend.position	Character: Legend position, "top", "bottom", "right", "left" Default = "right"
strip.background.fill	Color: Fill color from facet labels. Default = "grey85"

Author(s)

E.D. Gennatas

Examples

```
## Not run:
(p <- ggplot(iris, aes(Sepal.Length, Petal.Length, color = Species)) +
geom_point() +
ggtheme_light())

## End(Not run)
```

glm2table — *Collect summary table from list of massGLMs with same predictors, different outcome ("massy")*

Description

Collect summary table from list of massGLMs with same predictors, different outcome ("massy")

Usage

```
glm2table(x, xnames = NULL, include_anova_pvals = NA)
```

Arguments

x	list of glm models
xnames	Character, vector: names of models
include_anova_pvals	Logical: If TRUE, also output ANOVA p-values

Value

data.table with glm summaries

Author(s)

E.D. Gennatas

glmLite — *Bare bones decision tree derived from* rpart

Description

A super-stripped down decision tree for when space and performance are critical

Usage

```
glmLite(
  x,
  y,
  weights = NULL,
  method = c("glmnet", "cv.glmnet", "lm.ridge", "allSubsets", "forwardStepwise",
    "backwardStepwise", "glm", "sgd", "solve"),
  alpha = 0,
  lambda = 0.01,
  lambda.seq = NULL,
  cv.glmnet.nfolds = 5,
  which.cv.glmnet.lambda = c("lambda.min", "lambda.1se"),
  nbest = 1,
  nvmax = 8,
```

```
  sgd.model = "glm",
  sgd.model.control = list(lambda1 = 0, lambda2 = 0),
  sgd.control = list(method = "ai-sgd"),
  save.fitted = FALSE,
  ...
)
```

Arguments

x	Feature matrix or data.frame. Will be coerced to data.frame for method = "allSubsets", "forwardStepwise", or "backwardStepwise"
y	Outcome
weights	Float, vector: Case weights
method	Character: Method to use: • "glm": uses stats::lm.wfit • "glmnet": uses glmnet::glmnet • "cv.glmnet": uses glmnet:cv.glmnet • "lm.ridge": uses MASS::lm.ridge • "allsubsets": uses leaps::regsubsets with method = "exhaustive" • "forwardStepwise": uses leaps::regsubsets with method = "forward" • "backwardStepwise": uses leaps::regsubsets with method = "backward" • "sgd": uses sgd::sgd • "solve": uses base::solve • "none": fits no model and returns all zeroes, for programming convenience in special cases
alpha	Float: alpha for method = glmnet or cv.glmnet. Default = 0
lambda	Float: The lambda value for glmnet, cv.glmnet, lm.ridge Note: For glmnet and cv.glmnet, this is the lambda used for prediction. Training uses lambda.seq. Default = .01
lambda.seq	Float, vector: lambda sequence for glmnet and cv.glmnet. Default = NULL
cv.glmnet.nfolds	Integer: Number of folds for cv.glmnet
which.cv.glmnet.lambda	Character: Whitch lambda to pick from cv.glmnet: "lambda.min": Lambda that gives minimum cross-validated error;
nbest	Integer: For method = "allSubsets", number of subsets of each size to record. Default = 1
nvmax	Integer: For method = "allSubsets", maximum number of subsets to examine.
sgd.model	Character: Model to use for method = "sgd". Default = "glm"
sgd.model.control	List: model.control list to pass to sgd::sgd
sgd.control	List: sgd.control list to pass to sgd::sgd "lambda.1se": Largest lambda such that error is within 1 s.e. of the minimum.
save.fitted	Logical: If TRUE, save fitted values in output. Default = FALSE

Author(s)

E.D. Gennatas

`glmLiteBoostTV` *Boost an* **rtemis** *learner for regression*

Description

Perform regression by boosting a base learner

Usage

```
glmLiteBoostTV(
  x,
  y = NULL,
  x.valid = NULL,
  y.valid = NULL,
  x.test = NULL,
  y.test = NULL,
  resid = NULL,
  boost.obj = NULL,
  mod.params = list(),
  weights.p = 1,
  weights.0 = 0,
  weights = NULL,
  learning.rate = 0.1,
  max.iter = 10,
  init = NULL,
  seed = NULL,
  x.name = NULL,
  y.name = NULL,
  question = NULL,
  base.verbose = FALSE,
  verbose = TRUE,
  trace = 0,
  print.progress.every = 5,
  print.error.plot = "final",
  prefix = NULL,
  plot.theme = rtTheme,
  plot.fitted = NULL,
  plot.predicted = NULL,
  print.plot = FALSE,
  print.base.plot = FALSE,
  plot.type = "l",
  n.cores = rtCores,
  outdir = NULL,
  ...
)
```

Arguments

x	Numeric vector or matrix / data frame of features i.e. independent variables
y	Numeric vector of outcome, i.e. dependent variable

x.valid	Data.frame; optional: Validation data
y.valid	Float, vector; optional: Validation outcome
x.test	Numeric vector or matrix / data frame of testing set features Columns must correspond to columns in x
y.test	Numeric vector of testing set outcome
resid	Float, vector, length = length(y): Residuals to work on. Do not change unless you know what you're doing. Default = NULL, for regular boosting
boost.obj	[Internal use]
mod.params	Named list of arguments for glmLite
weights	Numeric vector: Weights for cases. For classification, weights takes precedence over ipw, therefore set weights = NULL if using ipw. Note: If weight are provided, ipw is not used. Leave NULL if setting ipw = TRUE. Default = NULL
learning.rate	Float (0, 1] Learning rate for the additive steps
max.iter	Integer: Maximum number of iterations (additive steps) to perform. Default = 10
init	Float: Initial value for prediction. Default = mean(y)
x.name	Character: Name for feature set
y.name	Character: Name for outcome
question	Character: the question you are attempting to answer with this model, in plain language.
base.verbose	Logical: verbose argument passed to learner
verbose	Logical: If TRUE, print summary to screen.
trace	Integer: If > 0, print diagnostic info to console
print.progress.every	Integer: Print progress over this many iterations
print.error.plot	String or Integer: "final" plots a training and validation (if available) error curve at the end of training. If integer, plot training and validation error curve every this many iterations during training. "none" for no plot.
prefix	Internal
plot.theme	Character: "zero", "dark", "box", "darkbox"
plot.fitted	Logical: if TRUE, plot True (y) vs Fitted
plot.predicted	Logical: if TRUE, plot True (y.test) vs Predicted. Requires x.test and y.test
print.plot	Logical: if TRUE, produce plot using mplot3 Takes precedence over plot.fitted and plot.predicted. Default = TRUE
print.base.plot	Logical: Passed to print.plot argument of base learner, i.e. if TRUE, print error plot for each base learner
plot.type	Character: "l" or "p". Plot using lines or points.
outdir	Path to output directory. If defined, will save Predicted vs. True plot, if available, as well as full model output, if save.mod is TRUE
...	Additional parameters to be passed to glmLite

Details

If learning.rate is set to 0, a nullmod will be created

Author(s)

E.D. Gennatas

---

gp *Bayesian Gaussian Processes [R]*

Description

Fit a gaussian process

Usage

```
gp(
  x,
  y,
  new.x = NULL,
  x.name = "x",
  y.name = "y",
  print.plot = TRUE,
  lwd = 3,
  cex = 1.2,
  par.reset = TRUE,
  ...
)
```

Arguments

x	Numeric vector or matrix of features, i.e. independent variables
y	Numeric vector of outcome, i.e. dependent variable
new.x	(Optional) Numeric vector or matrix of new set of features Must have same set of columns as x
x.name	Character: Name for feature set
y.name	Character: Name for outcome
print.plot	Logical: if TRUE, draw plot when done
lwd	Line width for plotting
cex	Character expansion factor for plotting
par.reset	Logical. Reset par to its original state
...	Additional arguments to be passed to tgp::bgp

Author(s)

E.D. Gennatas

gplot3_map *Plot US state or county choropleth map*

Description

Plot US state or county choropleth map using **ggplot2** based on `usmap::plot_usmap()`

Usage

```
gplot3_map(
  dat,
  regions = c("states", "state", "counties", "county"),
  include = NULL,
  exclude = NULL,
  limits = NULL,
  trans = "identity",
  col.lo = "#0290EE",
  col.mid = "black",
  col.hi = "#FE4AA3",
  col.na = "grey40",
  colorscale.midpoint = 0,
  colorbar.height = 0.15,
  colorbar.width = 0.03,
  border.width = 0.1,
  main = NULL,
  legend.title = NULL,
  scale.accuracy = 0.1,
  labelify = TRUE,
  theme = rt_gtheme_map(),
  labels = FALSE,
  col.labels = "gray50",
  filename = NULL,
  file.width = 7,
  file.height = 5,
  ...
)
```

Arguments

`dat`	data.frame with 2 columns: fips, value
`regions`	Vector, character: "states", "state", "counties", "county"
`include`	Vector, character: Names of states or counties to include
`exclude`	Vector, character: Names of states or counties to exclude
`col.lo`	Color: Color mapped ot lowest negative values. Default = "#0290EE"
`col.mid`	Color: Color mapped to value `colorscale.midpoint`, only used if `colorscale.midpoint` is not NULL. If set to NA, `colorscale.midpoint` will be set to NULL. Default = "black".
`col.hi`	Color: Color mapped to highest positive values. Default = "#FE4AA3"
`col.na`	Color: Color mapped to NA values. Default = "gray40"

colorscale.midpoint	Float: Midpoint for colorscale. Default = 0
colorbar.height	FLoat: Colorbar height, will be used as ggplot2::unit(colorbar.height, "npc")
colorbar.width	FLoat: Colorbar width, will be used as ggplot2::unit(colorbar.width, "npc")
border.width	Float: Map border width. Default = .1
main	Character: Main title. Default = NULL
legend.title	Character: Legend title. Default = NULL
theme	**ggplot2** theme to use. Default = rt_gtheme_map()
labels	Logical: If TRUE, label states. Default = FALSE
col.labels	Color for labels. Default = "gray50"

Author(s)

E.D. Gennatas

---

graph_node_metrics *Node-wise (i.e. vertex-wise) graph metrics*

Description

Node-wise (i.e. vertex-wise) graph metrics

Usage

```
graph_node_metrics(x, verbose = TRUE)
```

Arguments

x	**igraph** network

Author(s)

E.D. Gennatas

Examples

```
## Not run:
datcor <- cor(rnormmat(20, 20, seed = 2021))
datcor[sample(seq(datcor), 250)] <- 0
x <- igraph::graph_from_adjacency_matrix(adjmatrix = datcor,
                                         mode = "lower",
                                         weighted = TRUE,
                                         diag = FALSE)

graph_node_metrics(x)

## End(Not run)
```

gridCheck — **rtemis** *internal: Grid check*

Description

Checks if grid search needs to be performed. All tunable parameters should be passed to this function, individually or as a list. If any argument has more than one assigned values, the function returns TRUE, otherwise FALSE. This can be used to check whether gridSearchLearn must be run.

Usage

```
gridCheck(...)
```

Arguments

...	Parameters; will be converted to a list

Details

The idea is that if you know which parameter values you want to use, you define them directly e.g. `alpha = 0, lambda = .2`. If you don't know, you enter the set of values to be tested, e.g. `alpha = c(0, .5, 1), lambda = seq(.1, 1, .1)`.

gtTable — *Greater-than Table*

Description

Compare vectors element-wise, and tabulate N times each vector is greater than the others

Usage

```
gtTable(x = list(), x.name = NULL, na.rm = TRUE, verbose = TRUE)
```

Arguments

x	List of vectors of same length
x.name	Character: Name of measure being compared
na.rm	Passed to sum to handle missing values
verbose	Logical: If TRUE, write output to console

Author(s)

E.D. Gennatas

htest *Basic Bivariate Hypothesis Testing and Plotting*

Description

Basic Bivariate Hypothesis Testing and Plotting

Usage

```
htest(
  y,
  group = NULL,
  x = NULL,
  yname = NULL,
  groupname = NULL,
  xname = NULL,
 test = c("t.test", "wilcox.test", "aov", "kruskal.test", "chisq.test", "fisher.test",
    "cor.test", "pearson", "kendall", "spearman", "ks"),
  print.plot = TRUE,
  plot.args = list(),
  theme = rtTheme,
  verbose = TRUE,
  ...
)
```

Arguments

y	Float, vector: Outcome of interest
group	Factor: Groups to compare
x	Float, vector: Second outcome for correlation tests
yname	Character: y variable name
groupname	Character: group variable name
xname	Character: x variable name
test	Character: Test to use; one of: • Continuous outcome by group: "t.test", "wilcox.test", "aov", "kruskal.test" • Categorical outcome by group: "chisq.test", "fisher.test", "cor.test" • Two continuous variables: "pearson", "kendall", "spearman"
print.plot	Logical: If TRUE, print plot. Default = TRUE
plot.args	List of arguments to pass to plotting function
theme	Character: Run `themes()` for available themes
verbose	Logical: If TRUE, print messages to console. Default = TRUE
...	Additional arguments to pass to test call

Author(s)

E.D. Gennatas

Examples

```
## Not run:
# t.test, wilcoxon
y <- c(rnorm(200, 2, 1.2), rnorm(300, 2.5, 1.4))
group <- c(rep(1, 200), rep(2, 300))

ht_ttest <- htest(y, group, test = "t.test")
ht_wilcoxon <- htest(y, group, test = "wilcox.test")

# aov, kruskal
y <- c(rnorm(200, 2, 1.2), rnorm(300, 2.5, 1.4), rnorm(100, 2.3, 1.1))
group <- c(rep(1, 200), rep(2, 300), rep(3, 100))

ht_aov <- htest(y, group, test = "aov")
ht_kruskal <- htest(y, group, test = "kruskal.test")

# chisq, fisher
y <- c(sample(c(1, 2), 100, T, c(.7, .3)), sample(c(1, 2), 100, T, c(.35, .65)))
group <- c(rep(1, 100), rep(2, 100))
ht_chisq <- htest(y, group, test = "chisq")
ht_fisher <- htest(y, group, test = "fisher")

# cor.test
x <- rnorm(300)
y <- x * .3 + rnorm(300)
ht_pearson <- htest(x = x, y = y, test = "pearson")
ht_kendall <- htest(x = x, y = y, test = "kendall")
ht_kendall <- htest(x = x, y = y, test = "spearman")

## End(Not run)
```

---

ifNotNull *Say No to* NULL

Description

Returns the input, unless it is NULL, in which case it returns an empty vector / list, etc of defined type

Usage

```
ifNotNull(x, defType)
```

Arguments

x	Input of any type, may be NULL
defType	If x is NULL, return empty vector of this type. Options: list, numeric, character, integer

Details

This can be useful when creating S4, RC, or R6 objects

Author(s)

E.D. Gennatas

---

invlogit *Inverse Logit*

Description

Inverse Logit

Usage

```
invlogit(x)
```

Arguments

x	Float: Input data

Value

The inverse logit of the input

Author(s)

E.D. Gennatas

---

is.constant *Check if vector is constant*

Description

Check if vector is constant

Usage

```
is.constant(x)
```

Arguments

x	Vector: Input

Author(s)

E.D. Gennatas

is.discrete *Check if variable is discrete (factor or integer)*

Description

Check if variable is discrete (factor or integer)

Usage

```
is.discrete(x)
```

Arguments

x	Input

Author(s)

E.D. Gennatas

kfold *K-fold Resampling*

Description

K-fold Resampling

Usage

```
kfold(
  x,
  k = 10,
  stratify.var = NULL,
  strat.n.bins = 4,
  seed = NULL,
  verbose = TRUE
)
```

Arguments

x	Input Vector
k	Integer: Number of folds. Default = 10
stratify.var	Numeric vector (optional): Variable used for stratification. Defaults to y
strat.n.bins	Integer: Number of groups to use for stratification for resampler = "strat.sub" / "strat.boot"
seed	Integer: (Optional) Set seed for random number generator, in order to make output reproducible. See ?base::set.seed
verbose	Logical: If TRUE, print messages to screen

Author(s)

E.D. Gennatas

`labelify` *Format text for label printing*

Description

Format text for label printing

Usage

```
labelify(
  x,
  underscoresToSpaces = TRUE,
  dotsToSpaces = TRUE,
  toLower = FALSE,
  toTitleCase = TRUE,
  capitalize.strings = c("id"),
  stringsToSpaces = c("\\$", "`")
)
```

Arguments

`x`	Character: Input
`underscoresToSpaces`	Logical: If TRUE, convert underscores to spaces. Default = TRUE
`dotsToSpaces`	Logical: If TRUE, convert dots to spaces. Default = TRUE
`toLower`	Logical: If TRUE, convert to lowercase (precedes `toTitleCase`). Default = FALSE (Good for getting all-caps words converted to title case, bad for abbreviations you want to keep all-caps)
`toTitleCase`	Logical: If TRUE, convert to Title Case. Default = TRUE (This does not change all-caps words, set `toLower` to TRUE if desired)
`capitalize.strings`	Character, vector: Always capitalize these strings, if present. Default = "id"
`stringsToSpaces`	Character, vector: Replace these strings with spaces. Escape as needed for `gsub`. Default = "\$", which formats common input of the type `data.frame$variable`

Author(s)

E.D. Gennatas

labels2niftis *Write Matrix to Multiple Nifti Files*

Description

Writes matrix /data frame to nifti files columnwise. Each column of the matrix should correspond to the labels of a nifti file. i.e. nrow of matrix = n of labels in labeledNifti Niftis are written in parallel using `parallel::parApply`

Usage

```
labels2niftis(
  datamat,
  labeledNifti,
  prefix,
  verbose = TRUE,
  n.cores = future::availableCores()
)
```

Arguments

`datamat`	Matrix / Data Frame: Input
`labeledNifti`	Character: Path to labeled nifti file
`prefix`	Character: File output prefix
`verbose`	Logical: If TRUE, print messages to output
`n.cores`	Integer: Number of cores to use

Author(s)

E.D. Gennatas

labels2nii *Write Data to a Nifti File*

Description

Writes ROI values to nifti

Usage

```
labels2nii(label.vals, labeledNifti, prefix, datatype = "auto", verbose = TRUE)
```

Arguments

label.vals	Vector: The index of this vector should correspond to label numbers. Its values will replace the corresponding label numbers to form the new nifti file. This means that the values of this vector need to be in the same order as the labels. For example if labelefNifti has four labels: 0 (background), 10, 11, 20, then your label.vals should should have three values and they will be assigned to labels 10, 11, and 20.
labeledNifti	Character: Path to the labeled file whose labels corresponds to the values in label.vals
prefix	Character: Prefix of the output file. ".nii.gz" will be added automatically
datatype	Integer or "auto": Defines the datatype of the output. Options: 2: uint8, 4: int16, 8: int32, 16: float (Default), 64: double. Default = "auto"
verbose	Logical: If TRUE, print messages to output

Author(s)

E.D. Gennatas

---

learn *Supervised Learning with* **rtemis**

Description

Train any **rtemis** model

Usage

```
learn(
  x,
  y = NULL,
  mod = "ppr",
  x.test = NULL,
  y.test = NULL,
  x.name = NULL,
  y.name = NULL,
  args = list(),
  question = NULL,
  verbose = TRUE,
  print.plot = TRUE,
  ...
)
```

Arguments

x	Numeric vector or matrix / data frame of features i.e. independent variables
y	Numeric vector of outcome, i.e. dependent variable
mod	Character: Learner to use. To get list of options, run modSelect()
x.test	Numeric vector or matrix / data frame of testing set features Columns must correspond to columns in x

y.test	Numeric vector of testing set outcome
x.name	Character: Name for feature set
y.name	Character: Name for outcome
args	Optional list of parameters to be passed to learner
question	Character: the question you are attempting to answer with this model, in plain language.
verbose	Logical: If TRUE, print summary to screen.
print.plot	Logical: if TRUE, produce plot using mplot3 Takes precedence over plot.fitted and plot.predicted. Default = TRUE
...	Additional arguments to be passed to learner

Details

args and ... allow you to either pass named arguments, or a list of arguments (or both)

Author(s)

E.D. Gennatas

---

lincoef *Linear Model Coefficients*

Description

Get linear model coefficients

Usage

```
lincoef(
  x,
  y,
  weights = NULL,
  method = "glmnet",
  type = c("Regression", "Classification", "Survival"),
  alpha = 1,
  lambda = 0.05,
  lambda.seq = NULL,
  cv.glmnet.nfolds = 5,
  which.cv.glmnet.lambda = c("lambda.min", "lambda.1se"),
  nbest = 1,
  nvmax = 8,
  sgd.model = "glm",
  sgd.model.control = list(lambda1 = 0, lambda2 = 0),
  sgd.control = list(method = "ai-sgd"),
  trace = 0
)
```

Arguments

x	Feature matrix or data.frame. Will be coerced to data.frame for method = "allSubsets", "forwardStepwise", or "backwardStepwise"
y	Outcome
weights	Float, vector: Case weights
method	Character: Method to use: • "glm": uses stats::lm.wfit • "glmnet": uses glmnet::glmnet • "cv.glmnet": uses glmnet:cv.glmnet • "lm.ridge": uses MASS::lm.ridge • "allsubsets": uses leaps::regsubsets with method = "exhaustive" • "forwardStepwise": uses leaps::regsubsets with method = "forward" • "backwardStepwise": uses leaps::regsubsets with method = "backward" • "sgd": uses sgd::sgd • "solve": uses base::solve • "none": fits no model and returns all zeroes, for programming convenience in special cases
type	Character: "Regression", "Classification", or "Survival"
alpha	Float: alpha for method = glmnet or cv.glmnet. Default = 0
lambda	Float: The lambda value for glmnet, cv.glmnet, lm.ridge Note: For glmnet and cv.glmnet, this is the lambda used for prediction. Training uses lambda.seq. Default = .01
lambda.seq	Float, vector: lambda sequence for glmnet and cv.glmnet. Default = NULL
cv.glmnet.nfolds	Integer: Number of folds for cv.glmnet
which.cv.glmnet.lambda	Character: Whitch lambda to pick from cv.glmnet: "lambda.min": Lambda that gives minimum cross-validated error;
nbest	Integer: For method = "allSubsets", number of subsets of each size to record. Default = 1
nvmax	Integer: For method = "allSubsets", maximum number of subsets to examine.
sgd.model	Character: Model to use for method = "sgd". Default = "glm"
sgd.model.control	List: model.control list to pass to sgd::sgd
sgd.control	List: sgd.control list to pass to sgd::sgd "lambda.1se": Largest lambda such that error is within 1 s.e. of the minimum.
trace	Integer: If set to zero, all warnings are ignored

Details

This function minimizes checks for speed. It doesn't check dimensionality of x. Only use methods "glm", "sgd", or "solve" if there is only one feature in x.

Value

Named numeric vector of linear coefficients

Author(s)

E.D. Gennatas

`loadedPackageVersions` *Get version of all loaded packages (namespaces)*

Description

Get version of all loaded packages (namespaces)

Usage

```
loadedPackageVersions()
```

Value

Data frame with columns "Package_Name" and "Version"

Author(s)

E.D. Gennatas

`logistic` *Logistic function*

Description

Logistic function

Usage

```
logistic(x, x0 = 0, L = 1, k = 1)
```

Arguments

`x`	Float: Input
`x0`	x-value of the midpoint. Default = 0
`L`	maximum value. Default = 1
`k`	steepness of the curve. Default = 1

logit *Logit transform*

Description

Logit transform

Usage

```
logit(x)
```

Arguments

x	Float [0, 1] Input

logloss *Log Loss for a binary classifier*

Description

Log Loss for a binary classifier

Usage

```
logloss(true, estimated.prob)
```

Arguments

true	Factor: True labels. First level is the positive case
estimated.prob	Float, vector: Estimated probabilities

Author(s)

E.D. Gennatas

loocv *Leave-one-out Resampling*

Description

Leave-one-out Resampling

Usage

```
loocv(x)
```

Arguments

x	Input vector

Author(s)

E.D. Gennatas

lotri2edgeList *Connectivity Matrix to Edge List*

Description

Turn the lower triangle of a connectivity matrix (e.g. correlation matrix or similar) to an edge list of the form: Source, Target, Weight

Usage

```
lotri2edgeList(A, filename = NULL, verbose = TRUE)
```

Arguments

A	Square matrix
filename	Character: Path for csv file. Defaults to "conmat2edgelist.csv"
verbose	Logical: If TRUE, print messages to console

Details

The output can be read, for example, into gephi

Author(s)

E.D. Gennatas

lsapply lsapply

Description

lsapply

Usage

```
lsapply(X, FUN, ..., outnames = NULL, simplify = FALSE)
```

Arguments

X	a vector (atomic or list) or an expression object. Other objects (including classed objects) will be coerced by base::as.list.
FUN	the function to be applied to each element of X: see 'Details'. In the case of functions like +, %*%, the function name must be backquoted or quoted.
...	optional arguments to FUN.
outnames	Character vector: Optional names to apply to output
simplify	logical or character string; should the result be simplified to a vector, matrix or higher dimensional array if possible? For sapply it must be named and not abbreviated. The default value, TRUE, returns a vector or matrix if appropriate, whereas if simplify = "array" the result may be an array of "rank" (=length(dim(.))) one higher than the result of FUN(X[[i]]).

massCART *Mass-univariate CART prediction and variable importance*

Description

Predict outcome from each predictor separately and rank by percent Variance explained or Classification Accuracy

Usage

```
massCART(
  x,
  y = NULL,
  x.test = NULL,
  y.test = NULL,
  metric = NULL,
  minsplit = 2,
  minbucket = round(minsplit/3),
  cp = 0.01,
  maxcompete = 0,
  maxsurrogate = 0,
  usesurrogate = 2,
  surrogatestyle = 0,
```

```
  maxdepth = 22,
  xval = 0,
  ipw = FALSE,
  upsample = FALSE,
  downsample = FALSE,
  resample.seed = NULL,
  n.cores = 1,
  parallel.type = ifelse(.Platform$OS.type == "unix", "fork", "psock"),
  save.mod = FALSE,
  grid.print.plot = FALSE,
  verbose = TRUE,
  grid.verbose = verbose,
  print.plot = FALSE,
  ...
)
```

Arguments

x	Numeric vector or matrix / data frame of features i.e. independent variables
y	Numeric vector of outcome, i.e. dependent variable
x.test	Numeric vector or matrix / data frame of testing set features Columns must correspond to columns in x
y.test	Numeric vector of testing set outcome
metric	Character: Metric to minimize, or maximize if maximize = TRUE during grid search. Default = NULL, which results in "Balanced Accuracy" for Classification, "MSE" for Regression, and "Coherence" for Survival Analysis.
minsplit	[gS] Integer: Minimum number of cases that must belong in a node before considering a split.
minbucket	[gS] Integer: Minimum number of cases allowed in a child node.
cp	[gS] Float: Complexity threshold for allowing a split.
maxcompete	Integer: The number of competitor splits saved in the output
maxsurrogate	Integer: The number of surrogate splits retained in the output (See rpart::rpart.control).
usesurrogate	See rpart::rpart.control
surrogatestyle	See rpart::rpart.control
maxdepth	[gS] Integer: Maximum depth of tree.
xval	Integer: Number of cross-validations
ipw	Logical: If TRUE, apply inverse probability weighting (for Classification only). Note: If weights are provided, ipw is not used. Default = TRUE
upsample	Logical: If TRUE, upsample cases to balance outcome classes (for Classification only) Note: upsample will randomly sample with replacement if the length of the majority class is more than double the length of the class you are upsampling, thereby introducing randomness
downsample	Logical: If TRUE, downsample majority class to match size of minority class
resample.seed	Integer: If provided, will be used to set the seed during upsampling. Default = NULL (random seed)
n.cores	Integer: Number of cores to use
parallel.type	Character: "fork" (does not work in Windows) or "psock" (Works in all[?] OSs)

save.mod	Logical: If TRUE, save all output to an RDS file in outdir save.mod is TRUE by default if an outdir is defined. If set to TRUE, and no outdir is defined, outdir defaults to paste0("./s.", mod.name)
verbose	Logical: If TRUE, print summary to screen.
grid.verbose	Logical: Passed to gridSearchLearn
print.plot	Logical: if TRUE, produce plot using mplot3 Takes precedence over plot.fitted and plot.predicted. Default = TRUE

Author(s)

E.D. Gennatas

---

massGAM *Mass-univariate GAM Analysis*

Description

Fits a GAM for each of multiple outcomes using a fixed set of features (many y's, one X).

Usage

```
massGAM(
  x,
  y,
  covariates = NULL,
  x.name = NULL,
  y.name = NULL,
  k = NULL,
  family = gaussian(),
  weights = NULL,
  method = "REML",
  n.cores = rtCores,
  save.mods = FALSE,
  save.summary = TRUE,
  print.plots = FALSE,
  outdir = NULL,
  labeledNifti = NULL,
  save.plots = FALSE,
  new.x.breaks = 9
)
```

Arguments

x	Numeric vector or matrix / data frame of features i.e. independent variables
y	Numeric matrix / data frame: Outcomes
covariates	Numeric matrix / data.frame of additional covariates
x.name	Character: Name of the predictor
y.name	Character, vector: Names of the outcomes
k	Integer: Basis dimension for smoothing spline

family	family argument for mgcv::gam
weights	Vector, numeric: Weights for GAM
method	Estimation method for GAM
n.cores	Integer. Number of cores to use
save.mods	Logical. Should models be saved
save.summary	Logical. Should model summary be saved
print.plots	Logical Should plots be shown
outdir	Path to save output
labeledNifti	String. Path to labeled nifti file.
save.plots	Logical. Should plots be saved
new.x.breaks	Integer. Number of splits in the range of x to form vector of features for estimation of fitted values

Details

NA in the input will be kept as NA in the results, maintaining n of cases.

Author(s)

E.D. Gennatas

---

massGLM *Mass-univariate GLM Analysis*

Description

Run a mass-univariate analysis with either: a) single outome (y) and multiple predictors (x), one at a time, with an optional common set of covariates in each model - "massx" b) multiple different outcomes (y) with a fixed set of predictors (x) - "massy" Therefore, the term mass-univariate refers to looking at one variable of interest (with potential covariates of no interest) at a time

Usage

```
massGLM(
  x,
  y,
  scale.x = FALSE,
  scale.y = FALSE,
  type = NULL,
  xnames = NULL,
  ynames = NULL,
  save.mods = TRUE,
  print.plot = FALSE,
  include_anova_pvals = NA,
  verbose = TRUE,
  trace = 0,
  n.cores = 1
)
```

Arguments

x	Matrix / data frame of features
y	Matrix / data frame of outcomes
scale.x	Logical: If TRUE, scale and center x
scale.y	Logical: If TRUE, scale and center y
type	Character: "massx" or "massy". Default = NULL, where if (NCOL(x) > NCOL(y)) "massx" else "massy"
xnames	Character vector: names of x feature(s)
ynames	Character vector: names of y feature(s)
save.mods	Logical: If TRUE, save models. Default = TRUE
print.plot	Logical: If TRUE, print plot. Default = FALSE (best to choose which p-values you want to plot directly)
include_anova_pvals	Logical: If TRUE, include ANOVA p-values, generated by glm2table
verbose	Logical: If TRUE, print messages during run
trace	Integer: If > 0, print more verbose output to console.
n.cores	Integer: Number of cores to use. (Testing only, do not change from 1)

Author(s)

E.D. Gennatas

Examples

```
## Not run:
# Common usage is "reversed":
# x: outcome of interest as first column, optional covariates
# in the other columns
# y: features whose association with x we want to study
set.seed(2022)
features <- rnormmat(500, 40)
outcome <- features[, 3] - features[, 5] + features[, 14] + rnorm(500)
massmod <- massGLM(outcome, features)
plot(massmod)
plot(massmod, what = "coef")
plot(massmod, what = "volcano")

## End(Not run)
```

---

massUni *Mass-univariate Analysis*

Description

Run a mass-univariate analysis: same features (predictors) on multiple outcomes

Usage

```
massUni(
  x,
  y,
  mod = "gam",
  save.mods = FALSE,
  verbose = TRUE,
  n.cores = rtCores,
  ...
)
```

Arguments

x	Matrix / data frame of features
y	Matrix / data frame of outcomes
mod	**rtemis** algorithm to use. Options: run `modSelect()`
save.mods	Logical: If TRUE, save fitted models
verbose	Logical: If TRUE, print messages during run
n.cores	Integer: Number of cores to use
...	Arguments to be passed to `mod`

Author(s)

E.D. Gennatas

---

matchcases *Match cases by covariates*

Description

Find one or more cases from a 'pool' data.frame that match cases in a target data.frame. Match exactly and/or by distance (sum of squared distance).

Usage

```
matchcases(
  target,
  pool,
  n.matches = 1,
  target.id = NULL,
  pool.id = NULL,
  exactmatch.factors = TRUE,
  exactmatch.cols = NULL,
  distmatch.cols = NULL,
  norepeats = TRUE,
  ignore.na = FALSE,
  verbose = TRUE
)
```

Arguments

target	data.frame you are matching against
pool	data.frame you are looking for matches from
n.matches	Integer: Number of matches to return
target.id	Character: Column name in target that holds unique cases IDs. Default = NULL, in which case integer case numbers will be used
pool.id	Character: Same as target.id for pool
exactmatch.factors	Logical: If TRUE, selected cases will have to exactly match factors available in target
exactmatch.cols	Character: Names of columns that should be matched exactly
distmatch.cols	Character: Names of columns that should be distance-matched
norepeats	Logical: If TRUE, cases in pool can only be chosen once.
ignore.na	Logical: If TRUE, ignore NA values during exact matching.
verbose	Logical: If TRUE, print messages to console. Default = TRUE

Author(s)

E.D. Gennatas

Examples

```
set.seed(2021)
cases <- data.frame(PID = paste0("PID", seq(4)),
                    Sex = factor(c(1, 1, 0, 0)),
                    Handedness = factor(c(1, 1, 0, 1)),
                    Age = c(21, 27, 39, 24),
                    Var = c(.7, .8, .9, .6),
                    Varx = rnorm(4))
controls <- data.frame(CID = paste0("CID", seq(50)),
                       Sex = factor(sample(c(0, 1), 50, TRUE)),
                       Handedness = factor(sample(c(0, 1), 50, TRUE, c(.1, .9))),
                       Age = sample(16:42, 50, TRUE),
                       Var = rnorm(50),
                       Vary = rnorm(50))

mc <- matchcases(cases, controls, 2, "PID", "CID")
```

---

matchCasesByRules *Match Rules to Cases*

Description

Match Rules to Cases

Usage

```
matchCasesByRules(x, rules, verbose = TRUE)
```

Arguments

x	Matrix / data frame: Input features
rules	Vector, string: Rules (MUST be string, not factor)
verbose	Logical: If TRUE, print messages to console

Value

cases-by-rules matrix (binary; 1: match, 0: no match)

Author(s)

E.D. Gennatas

---

mergelongtreatment *Merge panel data treatment and outcome data*

Description

Merge long format treatment and outcome data from multiple sources with possibly hierarchical matching IDs using **data.table**

Usage

```
mergelongtreatment(
  x,
  group_varnames,
  time_varname = "Date",
  start_date,
  end_date,
  interval_days = 14,
  verbose = TRUE,
  trace = 1
)
```

Arguments

x	Named list: Long form datasets to merge. Will be converted to `data.table`
group_varnames	Vector, character: Variable names to merge by, in order. If first is present on a given pair of datasets, merge on that, otherwise try the next in line.
time_varname	Character: Name of column that should be present in all datasets containing time information. Default = "Date"
start_date	Date or characcter: Start date for final dataset in format "YYYY-MM-DD"
end_date	Date or character: End dat for final dataset in format "YYYY-MM-DD"
interval_days	Integer: Starting with `start_date` create timepoints every this many days. Default = 14
verbose	Logical: If TRUE, print messages to console. Default = TRUE
trace	Integer: If > 0 print additional info to console. Default = 1

Value

Merged **data.table**

metaMod *Meta Models for Regression (Model Stacking)*

Description

Train a meta model from the output of base learners trained using different learners (algorithms)

Usage

```
metaMod(
  x,
  y = NULL,
  x.test = NULL,
  y.test = NULL,
  base.mods = c("mars", "ranger"),
  base.params = vector("list", length(base.mods)),
  base.resample.rtset = rtset.resample(resampler = "kfold", n.resamples = 4),
  meta.mod = "gam",
  meta.params = list(),
  x.name = NULL,
  y.name = NULL,
  save.base.res = TRUE,
  save.base.full = FALSE,
  col = NULL,
  se.lty = 3,
  print.base.plot = FALSE,
  print.plot = TRUE,
  plot.fitted = NULL,
  plot.predicted = NULL,
  plot.theme = rtTheme,
  question = NULL,
  verbose.base.res.mods = FALSE,
  verbose.base.mods = FALSE,
  verbose = TRUE,
  trace = 0,
  base.n.cores = 1,
  n.cores = rtCores,
  save.mod = FALSE,
  outdir = NULL,
  ...
)
```

Arguments

x	Numeric vector or matrix of features, i.e. independent variables
y	Numeric vector of outcome, i.e. dependent variable
x.test	(Optional) Numeric vector or matrix of validation set features must have set of columns as x
y.test	(Optional) Numeric vector of validation set outcomes
base.mods	Character vector: Two or more base learners. Options: modSelect

base.params	List of length equal to N of base.mods. Each element should be a list of arguments to pass to the corresponding base mod
meta.mod	String. Meta learner. Options: modSelect
x.name	Character: Name for predictor set. (What kind of data is it?)
y.name	Character: Name for outcome
se.lty	How to plot standard errors. If a number, it corresponds to par("lty") line types and is plotted with lines(). If "solid", a transparent polygon is plotted using polygon()
resampler	String. Resampling method to use. Options: "bootstrap", "kfold", "strat.boot", "strat.sub"

Details

- Train a set of base learners on resamples of the training set x
- Train a meta learner to map bases' validation set predictions to outcomes
- Train base learners on full training set x
- Use the meta learner to predict test set outcome y.test from testing set (x.test)

---

mgetnames *Get names by string matching multiple patterns*

Description

Get names by string matching multiple patterns

Usage

```
mgetnames(
  x,
  pattern = NULL,
  starts_with = NULL,
  ends_with = NULL,
  ignore.case = TRUE,
  return.index = FALSE
)
```

Arguments

x	Character vector or object with names() method
pattern	Character vector: pattern(s) to match anywhere in names of x
starts_with	Character: pattern to match in the beginning of names of x
ends_with	Character: pattern to match at the end of names of x
ignore.case	Logical: If TRUE, well, ignore case. Default = TRUE
return.index	Logical: If TRUE, return integer index of matches instead of names

Value

Character vector of matched names or integer index

Author(s)

E.D. Gennatas

`mhist` *Histograms*

Description

Draws a histogram using lines.

Usage

```
mhist(
  x,
  breaks = "Sturges",
  measure = c("density", "counts"),
  lwd = 3,
  xlim = NULL,
  ylim = NULL,
  plot.axes = FALSE,
  xaxis = TRUE,
  yaxis = TRUE,
  xaxis.line = 0,
  yaxis.line = 0,
  xlab = NULL,
  ylab = measure,
  xaxs = "r",
  yaxs = "r",
  box = FALSE,
  grid = FALSE,
  col = pennCol$lighterBlue,
  horiz = FALSE,
  main = "",
  add = FALSE,
  ...
)
```

Arguments

`x`	Input vector
`breaks`	See `hist("breaks")` Default = "Sturges"
`measure`	Character: "density"(Default), "counts"
`lwd`	Float: Line width
`xlim`	Vector, length 2: x-axis limits
`ylim`	Vector, length 2: y-axis limits
`plot.axes`	Logical: If TRUE, draws plot axes. Separate from `xaxis` and `yaxis`
`xaxis`	Logical: If TRUE, draws x-axis
`yaxis`	Logical: If TRUE, draws y-axis

xaxis.line	Float: Number of lines into the margin to position xaxis. See axis("line")
yaxis.line	Float: Number of lines into the margin to position yaxis. See axis("line")
xlab	Character: x-axis label
ylab	Character: y-axis label
xaxs	Character: 'r' (Default): Extends x-axis range by 4 percent at each end, 'i': Does not extend x-axis range
yaxs	Character: 'r' (Default): Extends y-axis range by 4 percent at each end, 'i': Does not extend y-axis range
box	Logical: If TRUE, draws a box around plot
grid	Logical: If TRUE, draws a grid
col	Color to use for histogram lines
horiz	Logical: If TRUE, switches x and y axes. Important: Provide all other arguments as if for a non-rotated plot - i.e. xlab will become the y-axis label
main	Character: Main title
add	Logical: If TRUE, add histogram to existing plot (Caution: make sure the axes line up!)
...	Additional arguments to be passed to graphics::plot

Details

Using horiz = TRUE, you can draw vertical histograms (as used by mplot3_xym)

Author(s)

E.D. Gennatas

---

mlegend *Add legend to* mplot3 *plot*

Description

Add legend to mplot3 plot

Usage

```
mlegend(
  lims,
  title = NULL,
  group.names,
  title.col = "black",
  col = rtpalette("rtCol1"),
  horiz.pad = 0.04,
  footer = NULL,
  font = 1,
  font.family = "Helvetica Neue"
)
```

Arguments

lims	List with plot limits in the form list(xlim = xlim, ylim = ylim) e.g. as returned by mplot3_xy
title	Character: Legend title
group.names	Character: group names
title.col	Title color
col	Color vector
horiz.pad	Numeric: Proportion of plot width to pad by
footer	Character: Footer annotation
font	1 or 2 for regular and bold
font.family	Character: Font family to use

Author(s)

E.D. Gennatas

---

modError *Error Metrics for Supervised Learning*

Description

Calculate error metrics for pair of vector, e.g. true and estimated values from a model

Usage

```
modError(
  true,
  estimated,
  estimated.prob = NULL,
  verbose = FALSE,
  type = NULL,
  rho = FALSE,
  tau = FALSE,
  na.rm = TRUE
)
```

Arguments

true	Vector: True values
estimated	Vector: Estimated values
estimated.prob	Vector: Estimated probabilities for Classification, if available.
verbose	Logical: If TRUE, print output to screen
type	Character: "Regression", "Classification", or "Survival". If not provided, will be set to Regression if y is numeric.
rho	Logical: If TRUE, calculate Spearman's rho.
tau	Logical: If TRUE, calculate Kendall's tau. This can be slow for long vectors
na.rm	Logical: Passed to mean and range functions.

Details

In regression, NRMSE = RMSE / range(observed)

Value

Object of class `modError`

Author(s)

E.D. Gennatas

`modSelect` *Select* **rtemis** *Learner*

Description

Accepts learner name (supports abbreviations) and returns **rtemis** function name or the function itself. If run with no parameters, prints list of available algorithms.

Usage

```
modSelect(mod, fn = FALSE, desc = FALSE)
```

Arguments

`mod`	Character: Model name. Case insensitive. e.g. "XGB" for xgboost
`fn`	Logical: If TRUE, return function, otherwise name of function. Defaults to FALSE
`desc`	Logical: If TRUE, return full name / description of algorithm `mod`

Value

function or name of function (see param `fn`) or full name of algorithm (`desc`)

Author(s)

E.D. Gennatas

mplot3_adsr mplot3*: ADSR Plot*

Description

Plot Attack Decay Sustain Release Envelope Generator using mplot3

Usage

```
mplot3_adsr(
  Attack = 300,
  Decay = 160,
  Sustain = 40,
  Release = 500,
  Value = 80,
  I = 200,
  O = 1800,
  lty = 1,
  lwd = 4,
  main = "ADSR Envelope",
  main.line = 1.6,
  main.col = "white",
  Attack.col = "#44A6AC",
  Decay.col = "#F4A362",
  Sustain.col = "#3574A7",
  Release.col = "#C23A70",
  Decay.nl = FALSE,
  draw.poly = FALSE,
  poly.alpha = 0.15,
  draw.verticals = TRUE,
  v.lty = 1,
  v.lwd = 0.8,
  arrow.code = 2,
  arrow.length = 0.09,
  grid = FALSE,
  grid.lty = 1,
  grid.lwd = 0.4,
  grid.col = NULL,
  zerolines.col = "gray50",
  theme = "darkgray",
  labs.col = "gray70",
  tick.col = "gray70",
  on.lty = 2,
  on.lwd = 7,
  on.alpha = 0.4,
  on.col = "gray70",
  off.col = "gray70",
  pty = "m",
  mar = c(3, 3, 3.2, 0.5),
  xaxs = "i",
  yaxs = "i",
```

```
  par.reset = TRUE,
  ...
)
```

Arguments

Attack	Float: Attack time (in milliseconds)
Decay	Float: Decay time (in milliseconds)
Sustain	Float: Sustain Level (percent)
Release	Float: Release time (in milliseconds)
Value	Float: Value (percent)
I	Float: Note on time (in milliseconds)
O	Float: Note off time (in milliseconds)
theme	Character: "light" or "dark" (Default)
...	Additional arguments to pass to mplot3_xy

Details

Learn more: (https://en.wikipedia.org/wiki/Synthesizer#Attack_Decay_Sustain_Release_.28ADSR.29_envelope "ADSR Wikipedia")

Author(s)

E.D. Gennatas

Examples

```
## Not run:
mplot3_adsr()

## End(Not run)
```

---

mplot3_bar mplot3*: Barplot*

Description

Draw barplots

Usage

```
mplot3_bar(
  x,
  error = NULL,
  col = NULL,
  error.col = "white",
  error.lwd = 2,
  alpha = 1,
  beside = TRUE,
  border = NA,
```

```
  width = 1,
  space = NULL,
  xlim = NULL,
  ylim = NULL,
  xlab = NULL,
  ylab = NULL,
  main = NULL,
  las = 1.5,
  xnames = NULL,
  xnames.srt = 0,
  xnames.adj = ifelse(xnames.srt == 0, 0.5, 1),
  xnames.line = 0.5,
  xnames.font = 1,
  xnames.cex = 1,
  xnames.y.pad = 0.08,
  xnames.at = NULL,
  color.bygroup = FALSE,
  group.legend = NULL,
  legend.x = NULL,
  legend.y = NULL,
  group.names = NULL,
  legend.font = 1,
  bartoplabels = NULL,
  bartoplabels.line = 0,
  bartoplabels.font = 1,
  mar = c(2.5, 3, 2, 1),
  pty = "m",
  barplot.axes = FALSE,
  yaxis = TRUE,
  ylim.pad = 0.04,
  theme = rtTheme,
  palette = rtPalette,
  autolabel = letters,
  par.reset = TRUE,
  pdf.width = 6,
  pdf.height = 6,
  filename = NULL,
  ...
)
```

Arguments

x	Vector or Matrix: If Vector, each value will be drawn as a bar. If Matrix, each column is a vector, so multiple columns signify a different group. e.g. Columns could be months and rows could be N days sunshine, N days rainfall, N days snow, etc.
col	Vector of colors to use
alpha	Float: Alpha to be applied to col
border	Color if you wish to draw border around bars, NA for no borders (Default)
space	Float: Space left free on either side of the bars, as a fraction of bar width. A single number or a vector, one value per bar. If x is a matrix, space can be

	length 2 vector, signifying space between bars within group and between groups. Default = c(0, 1) if x is matrix and `beside = TRUE`, otherwise Default = .2
`xlim`	Float vector, length 2: x-axis limits
`ylim`	Float vector, length 2: y-axis limits
`xlab`	Character: x-axis label
`ylab`	Character: y-axis label
`main`	Character: Plot title
`color.bygroup`	Logical: If TRUE, and input is a matrix, each group's bars will be given the same color, otherwise bars across groups will be given the same sequence of colors. Default = FALSE
`group.legend`	Logical: If TRUE, place `group.names` in a legend
`group.names`	(Optional) If multiple groups are plotted, use these names if `group.title = TRUE`
`mar`	Float, vector, length 4: Margins; see `par("mar")`
`pty`	Character: "s" gives a square plot; "m" gives a plot that fills graphics device size. Default = "m" (See `par("pty")`)
`theme`	Character: Run `themes()` for available themes
`palette`	Vector of colors, or Character defining a builtin palette - get options with `rtpalette()`
`autolabel`	Vector to be used to generate autolabels when using rtlayout with `autolabel = TRUE`. Default = `letters`
`par.reset`	Logical: If TRUE, reset `par` setting before exiting.
`pdf.width`	Float: Width in inches for pdf output (if `filename` is set).
`pdf.height`	Float: Height in inches for pdf output.
`filename`	Character: Path to file to save plot. Default = NULL
`...`	Additional arguments to `graphics::barplot`
`legend`	Logical: If TRUE, and input is matrix, draw legend for each case. Note: you may need to adjust `mar` and `legend.inset` if you want to place the legend outside the plot (can use e.g.`legend.inset = c(-.5, 0)`)

Author(s)

E.D. Gennatas

`mplot3_box` `mplot3`*: Boxplot*

Description

Draw boxplots of a vector (single box), data.frame (one box per column) or list (one box per element - good for variable of different length)

Usage

```
mplot3_box(
  x,
  col = NULL,
  alpha = 0.66,
  border = NULL,
  border.alpha = 1,
  group.spacing = 0.25,
  xlim = NULL,
  ylim = NULL,
  xlab = NULL,
  ylab = NULL,
  boxwex = NULL,
  staplewex = 0.5,
  horizontal = FALSE,
  main = NULL,
  names.arg = NULL,
  axisnames = FALSE,
  groupnames = NULL,
  xnames = NULL,
  xnames.at = NULL,
  xnames.y = NULL,
  xnames.font = 1,
  xnames.adj = NULL,
  xnames.pos = NULL,
  xnames.srt = NULL,
  order.by.fn = NULL,
  legend = FALSE,
  legend.names = NULL,
  legend.position = "topright",
  legend.inset = c(0, 0),
  mar = NULL,
  oma = rep(0, 4),
  pty = "m",
  yaxis = TRUE,
  ylim.pad = 0,
  theme = rtTheme,
  labelify = TRUE,
  autolabel = letters,
  na.rm = TRUE,
  palette = rtPalette,
  par.reset = TRUE,
  pdf.width = 6,
  pdf.height = 6,
  filename = NULL,
  ...
)
```

Arguments

x	Vector, data.frame or list: Each data.frame column or list element will be drawn as a box

col	Vector of colors to use
alpha	Float: Alpha to be applied to col
border	Color for lines around boxes
xlim	Float vector, length 2: x-axis limits
ylim	Float vector, length 2: y-axis limits
xlab	Character: x-axis label
ylab	Character: y-axis label
boxwex	Float: Scale factor for box width. Default = .5
staplewex	Float: max and min line ("staple") width proportional to box. Default = .5
horizontal	Logical: If TRUE, draw horizontal boxplot(s). Default = FALSR
main	Character: Plot title
order.by.fn	Character: "mean", "median" or any function that outputs a single number: E stimate function on each vector and order boxes (when input is data.frame or list) by ascending order. Default = NULL, i.e. no reordering
mar	Float, vector, length 4: Margins; see par("mar")
oma	Float, vector, length 4: Outer margins; see par("oma")
pty	Character: "s" gives a square plot; "m" gives a plot that fills graphics device size. Default = "m" (See par("pty"))
theme	Character: Run themes() for available themes
autolabel	Vector to be used to generate autolabels when using rtlayout with autolabel = TRUE. Default = letters
na.rm	Logical: If TRUE, remove NA values, otherwise function will give error. Default = TRUE
palette	Vector of colors, or Character defining a builtin palette - get options with rtpalette()
par.reset	Logical: If TRUE, reset par setting before exiting.
pdf.width	Float: Width in inches for pdf output (if filename is set).
pdf.height	Float: Height in inches for pdf output.
filename	Character: Path to file to save plot. Default = NULL
...	Additional arguments to graphics::boxplot

Details

Note that argument ‘xnames‘ refers to the x-axis labels below each box. If not specified, these are inferred from the input when possible. Argument ‘xlab‘ is a single label for the x-axis as per usual and often omitted if ‘xnames‘ suffice.

Author(s)

E.D. Gennatas

Examples

```
## Not run:
## vector
x <- rnorm(500)
mplot3_box(x)

## data.frame - each column one boxplot
x <- data.frame(alpha = rnorm(50), beta = rnorm(50), gamma = rnorm(50))
mplot3_box(x)

## list of vectors - allows different length vectors
x <- list(alpha = rnorm(50),
          beta = rnorm(80, 4, 1.5),
          gamma = rnorm(30, -3, .5))
mplot3_box(x)

## grouped boxplots: input a list of lists. outer list: groups; inner lists: matched data vectors
x <- list(Cases = list(Weight = rnorm(50), Temperature = rnorm(45, 1)),
          Controls = list(Weight = rnorm(80), Temperature = rnorm(72)))
mplot3_box(x)

## End(Not run)
```

---

mplot3_conf *Plot confusion matrix*

Description

Plots confusion matrix and classification metrics

Usage

```
mplot3_conf(
  object,
  main = "auto",
  xlab = "Reference",
  ylab = "Estimated",
  plot.metrics = TRUE,
  mod.name = NULL,
  oma = c(0, 0, 0, 0),
  dim.main = NULL,
  dim.lab = 1,
  dim.in = 4,
  dim.out = -1,
  font.in = 2,
  font.out = 1,
  cex.main = 1.2,
  cex.in = 1.2,
  cex.lab = 1.2,
  cex.lab2 = 1.2,
  cex.lab3 = 1,
  cex.out = 1,
```

```
  col.main = "auto",
  col.lab = "auto",
  col.text.out = "auto",
  col.bg = "auto",
  col.bg.out1 = "auto",
  col.bg.out2 = "auto",
  col.text.hi = "auto",
  col.text.lo = "auto",
  show.ba = TRUE,
  theme = getOption("rt.theme", "white"),
  mid.col = "auto",
  hi.color.pos = "#18A3AC",
  hi.color.neg = "#C23A70",
  par.reset = TRUE,
  pdf.width = 7,
  pdf.height = 7,
  filename = NULL,
  ...
)
```

Arguments

object	Either a classification rtMod, or a table/matrix/data.frame
main	Character: Plot title
xlab	Character: x-axis label
ylab	Character: y-axis label
plot.metrics	Logical: If TRUE, draw classification metrics next to confusion matrix. Default = TRUE
mod.name	Character: Name of the algorithm used to make predictions. If NULL, will look for object$mod.name.
oma	Float, vector, length 4: Outer margins.
dim.main	Float: Height for title.
dim.lab	Float: Height for labels.
dim.in	Float: Height/Width for confusion matrix cells. Default = 4
dim.out	Float: Height for metrics cells. Default = -1, which autoadjusts depending on number of output classes
font.in	Integer: The font parameter for confusion matrix cells. Default = 2
font.out	Integer: The font parameter for metrics cells.
cex.in	Float: The cex parameter for confusion matrix cells.
cex.lab	Float: The cex parameter for first line of label cells.
cex.lab2	Float: The cex parameter for second line of label cells.
cex.lab3	Float: The cex parameter for classification metrics.
cex.out	Float: The cex parameter for metrics cells.
col.main	Color for title. Default = "auto", determined by theme
col.lab	Color for labels. Default = "auto", determined by theme
col.text.out	Color for metrics cells' text. Default = "auto", determined by theme

col.bg	Color for background. Default = "auto", determined by theme
col.bg.out1	Color for metrics cells' background (row1). Default = "auto", determined by theme
col.bg.out2	Color for metrics cells' background (row2). Default = "auto", determined by theme
col.text.hi	Color for high confusion matrix values. Default = "auto", determined by theme
col.text.lo	Color for low confusion matrix values. Default = "auto", determined by theme
show.ba	Logical: If TRUE, show Balanced Accuracy at bottom right corner.
theme	Character: "light", or "dark". Set to options("rt.theme"), if set, otherwise "light"
mid.col	Color: The mid color for the confusion matrix. Default = "auto", determined by theme
hi.color.pos	Color: The hi color for correct classification.
hi.color.neg	Color: The hi color for missclassification.
par.reset	Logical: If TRUE, reset par before exit.
pdf.width	Float: PDF width, if filename is set
pdf.height	Float: PDF height, if filename is set
filename	Character: If specified, save plot to this path.

Details

This function uses its multiple cex args instead of the theme's 'cex' parameter

Value

List of metrics, invisibly

Author(s)

E.D. Gennatas

Examples

```
## Not run:
true <- c("alpha", "alpha", "alpha", "alpha", "beta", "beta", "beta", "beta")
predicted <- c("alpha", "alpha", "alpha", "beta", "beta", "alpha", "alpha", "beta")
mplot3_conf(table(predicted, true))

## End(Not run)
```

`mplot3_confbin` *Plot extended confusion matrix for binary classification*

Description

Plots an extended confusion matrix using mplot3_img

Usage

```
mplot3_confbin(
  object,
  main = NULL,
  xlab = "True",
  ylab = "Estimated",
  mod.name = NULL,
  mar = c(4, 5, 4, 3),
  dim.lab = 1,
  dim.in = 4,
  dim.out = 2,
  font.in = 2,
  font.out = 2,
  cex.in = 1.2,
  cex.lab = 1.2,
  cex.lab2 = 1,
  cex.out = 1,
  col.text.out = "white",
  col.bg.out = "gray50",
  theme = "light",
  mid.color = NULL,
  hi.color.pos = "#18A3AC",
  hi.color.neg = "#716FB2",
  par.reset = TRUE,
  pdf.width = 8.7,
  pdf.height = 8.7,
  filename = NULL,
  ...
)
```

Arguments

`object`	Either 1. a classification `rtMod`, b. a `caret::confusionMatrix` object, or c. a matrix / data.frame / table
`main`	Character: Plot title
`xlab`	Character: x-axis label
`ylab`	Character: y-axis label
`mod.name`	Character: Name of the algorithm used to make predictions. If NULL, will look for `object$mod.name`. Default = NULL
`mar`	Numeric, vector, length 4: Overall margins
`dim.lab`	Float: Height for labels

dim.in	Float: Width and height for confusion matrix cells
dim.out	Float: Height for metrics cells
font.in	Integer: The font parameter for confusion matrix cells
font.out	Integer: The font parameter for metrics cells
cex.in	Float: The cex parameter for confusion matrix cells
cex.lab	Float: The cex parameter for first line of label cells
cex.lab2	Float: The cex parameter for second line of label cells
cex.out	Float: The cex parameter for metrics cells
col.text.out	Color for metrics cells' text
col.bg.out	Color for metrics cells' background
theme	Character: "light", or "dark"
mid.color	Color: The mid color for the confusion matrix. Default = "white" for theme = "light", "black" for "dark"
hi.color.pos	Color: The hi color for correct classification.
hi.color.neg	Color: The hi color for missclassification
par.reset	Logical: If TRUE, reset par before exit. Default = TRUE
pdf.width	Float: PDF width, if filename is set
pdf.height	Float: PDF height, if filename is set
filename	Character: If specified, save plot to this path. Default = NULL
...	Not used

Value

List of metrics, invisibly

Author(s)

E.D. Gennatas

`mplot3_decision` `mplot3`*: Decision boundaries*

Description

Plot classification decision boundaries of rtemis models

Usage

```
mplot3_decision(
  rtmod,
  data,
  vars = c(1, 2),
  dots.per.axis = 100,
  bg.cex = 0.5,
  bg.alpha = 0.4,
  bg.pch = 15,
```

```
  par.reset = TRUE,
  theme = "white",
  col = c("#18A3AC", "#F48024"),
  contour.col = "black",
  contour.lwd = 0.1,
  point.pch = c(3, 4),
  point.alpha = 1
)
```

Arguments

rtmod	rtemics trained model
data	Matrix / data frame of features; last column is class
vars	Integer vector, length 2: Index of features (columns of x) to use to draw decision boundaries. Default = c(1, 2)
dots.per.axis	Integer: Draw a grid with this many dots on each axis. Default = 100
bg.cex	Float: Point cex for background / decision surface. Default = .5
bg.alpha	Float: Point alpha for background / decision surface. Default = .2
bg.pch	Integer vector: pch for background / decision surface. Default = c(3, 4)
par.reset	Logical: If TRUE, reset par before exiting. Default = TRUE
theme	Character: Theme for mplot3_xy, "light" or "dark". Default = "light'
col	Color vector for classes. Default = ucsfPalette
contour.col	Color for decision boundary. Default = "black"
contour.lwd	Float: Line width for decision boundary. Default = .3
point.pch	Integer: pch for data points. Default = c(3, 4)
point.alpha	Float: Alpha for data points. Default = 1

Details

If data has more than 2 variables, any variable not selected using vars will be fixed to their mean Underlying model (e.g. randomForest, rpart, etc) must support standard R predict format for classification: predict(model, newdata, type = "class")

Value

Predicted labels for background grid (invisibly)

Author(s)

E.D. Gennatas

Examples

```
## Not run:
dat <- as.data.frame(mlbench::mlbench.2dnormals(200))
mod.cart <- s_CART(dat)
mod.rf <- s_RF(dat)
mplot3_decision(mod.cart, dat)
mplot3_decision(mod.rf, dat)

## End(Not run)
```

`mplot3_fit` `mplot3`*: True vs. Fitted plot*

Description

An `mplot3_xy` alias with defaults for plotting a learner's performance

Usage

```
mplot3_fit(
  x,
  y,
  fit = "lm",
  se.fit = TRUE,
  fit.error = TRUE,
  axes.equal = TRUE,
  diagonal = TRUE,
  theme = rtTheme,
  marker.col = NULL,
  fit.col = NULL,
  pty = "s",
  zerolines = FALSE,
  fit.legend = FALSE,
  mar = NULL,
  ...
)
```

Arguments

`x`	Vector, numeric / factor / survival for regression, classification, survival: True values
`y`	Vector, numeric / factor / survival for regression, classification, survival: Predicted values
`fit`	Character: **rtemis** model to calculate y ~ x fit. Options: see `modSelect` Can also be Logical, which will give a GAM fit if TRUE. If you specify "NLA", the activation function should be passed as a string.
`se.fit`	Logical: If TRUE, draw the standard error of the fit
`fit.error`	Logical: If TRUE,
`axes.equal`	Logical: Should axes be equal? Defaults to FALSE
`diagonal`	Logical: If TRUE, draw diagonal line. Default = FALSE
`theme`	Character: Run `themes()` for available themes
`marker.col`	Color for marker
`fit.col`	Color: Color of the fit line.
`pty`	Character: "s" for square plot, "m" to fill device. Default = "s"
`fit.legend`	Logical: If TRUE, show fit legend
`mar`	Float, vector, length 4: Margins; see `par("mar")`
`...`	Additional argument to be passed to mplot3_conf (classification) or mplot3_xy (regression)

`mplot3_fret` `mplot3`*: Guitar Fretboard*

Description

Draw color-coded notes on a guitar fretboard for standard E-A-D-G-B-e tuning

Usage

```
mplot3_fret(
  theme = rtTheme,
  useSharps = FALSE,
  strings.col = "auto",
  frets.col = "auto",
  inlays = TRUE,
  inlays.col = "auto",
  inlays.cex = 2,
  par.reset = TRUE,
  ...
)
```

Arguments

`theme`	Character: "light" or "dark"
`useSharps`	Logical: If TRUE, draw sharp instead of flat notes. Default = FALSE
`strings.col`	Color for strings
`frets.col`	Color for frets
`inlays`	Logical: Draw fretboard inlays. Default = TRUE
`inlays.col`	Color for inlays
`inlays.cex`	Float: Character expansion factor for inlays. Default = 2
`par.reset`	Logical: If TRUE, reset par before exit

Details

Plot is very wide and short. Adjust plot window accordingly. Practice every day.

Author(s)

E.D. Gennatas

mplot3_graph *Plot* igraph *networks*

Description

Plot igraph networks

Usage

```
mplot3_graph(
  net,
  vertex.size = 12,
  vertex.col = NULL,
  vertex.alpha = 0.33,
  vertex.label.col = NULL,
  vertex.label.alpha = 0.66,
  vertex.frame.col = NA,
  vertex.label = NULL,
  vertex.shape = "circle",
  edge.col = NULL,
  edge.alpha = 0.2,
  edge.curved = 0.35,
  edge.width = 2,
  layout = c("fr", "dh", "drl", "gem", "graphopt", "kk", "lgl", "mds", "sugiyama"),
  coords = NULL,
  layout_params = list(),
  cluster = NULL,
  groups = NULL,
  cluster_params = list(),
  cluster_mark_groups = TRUE,
  mark.col = NULL,
  mark.alpha = 0.3,
  mark.border = NULL,
  mark.border.alpha = 1,
  cluster_color_vertices = FALSE,
  theme = rtTheme,
  theme_extra_args = list(),
  palette = rtPalette,
  mar = rep(0, 4),
  par.reset = TRUE,
  filename = NULL,
  verbose = TRUE,
  ...
)
```

Arguments

net	igraph network
vertex.size	Numeric: Vertex size
vertex.col	Color for vertices

`vertex.alpha` Numeric: Transparency for `vertex.col`

`vertex.label.col`
Color for vertex labels

`vertex.label` Character vector: Vertex labels. Default = NULL, which will keep existing names in `net` if any. Set to NA to avoid printing vertex labels

`vertex.shape` Character: Vertex shape. See `igraph::plot.igraph("vertex.shape")`. Default = "circle"

`edge.col` Color for edges

`edge.alpha` Numeric: Transparency for edges. Default = .2

`edge.curved` Numeric: Curvature of edges. Default = .35

`edge.width` Numeric: Edge thickness

`layout` Character: one of: "fr", "dh", "drl", "gem", "graphopt", "kk", "lgl", "mds", "sugiyama", corresponding to all the available layouts in **igraph**

`coords` Output of precomputed **igraph** layout. If provided, `layout` is ignored

`layout_params` List of parameters to pass to `layout` function

`cluster` Character: one of: "edge_betweenness", "fast_greedy", "infomap", "label_prop", "leading_eigen", "louvain", "optimal", "spinglass", "walktrap", corresponding to all the available **igraph** clustering functions

`groups` Output of precomputed **igraph** clustering. If provided, `cluster` is ignored

`cluster_params` List of parameters to pass to `cluster` function

`cluster_mark_groups`
Logical: If TRUE, draw polygons to indicate clusters, if `groups` or `cluster` defined

`mark.col` Colors, one per group for polygon surrounding cluster. Note: You won't know the number of groups unless they are precomputed. The colors will be recycled as needed.

`mark.alpha` Float [0, 1]: Transparency for `mark.col`. Default = .3

`mark.border` Colors, similar to `mark.col` for border

`mark.border.alpha`
Float [0, 1]: Transparency for `mark.border`. Default = 1

`cluster_color_vertices`
Logical: If TRUE, color vertices by cluster membership. Default = FALSE

`theme` **rtemis** theme to use

`theme_extra_args`
List of extra arguments to pass to the theme function defined by `theme`. This argument is used when the extra args (...) are passed the plotting function (in this case `igraph::plot.igraph`) and not to the theme function

`mar` Numeric vector, length 4: `par`'s margin argument

`par.reset` Logical: If TRUE, reset par before exiting. Default = TRUE

`filename` Character: If provided, save plot to this filepath

`verbose` Logical, If TRUE, print messages to console. Default = TRUE

`...` Extra arguments to pass to `igraph::plot.igraph()`

`vertex.frame.vcol`
Color for vertex border (frame)

Author(s)

E.D. Gennatas

`mplot3_harmonograph` *Plot a harmonograph*

Description

Plot a [harmonograph](https://en.wikipedia.org/wiki/Harmonograph)

Usage

```
mplot3_harmonograph(
  steps = seq(1, 500, by = 0.01),
  seed = NULL,
  col = "white",
  alpha = 0.2,
  bg = "black",
  lwd = 1,
  text = NULL,
  text.side = 1,
  text.line = -1,
  text.adj = 0,
  text.padj = 0,
  text.col = NULL,
  mar = c(0, 0, 0, 0),
  oma = c(0, 0, 0, 0),
  xlim = NULL,
  ylim = NULL,
  new = FALSE,
  par.reset = TRUE
)
```

Arguments

`steps`	Float, vector
`seed`	Integer
`col`	Line color. Default = "white"
`alpha`	Alpha for line color `col`. Default = .2
`bg`	Color for background. Default = "black"
`lwd`	Float: Line width
`text`	Character: Text you want printed along with the harmonograph. Default = NULL
`text.side`	Integer 1, 2, 3, 4: `side` argument for `mtext`
`text.line`	Float: `line` argument for `mtext`
`text.adj`	Float: `adj` argument for `mtext`
`text.padj`	Float: `padj` argument for `mtext`
`text.col`	Color: Text color. Default is same as `col`
`mar`	Float vector, length 4: Plot margins. (`par`'s `mar` argument) Default = c(0, 0, 0, 0)
`par.reset`	Logical. If TRUE, reset par before exit

Details

Unless you define a seed, each graph will be random. Try different seeds if you want to reproduce your graphs. Some seeds to try: 9, 17, 26, 202, 208, ...

Author(s)

E.D. Gennatas

`mplot3_heatmap` `mplot3` *Heatmap (*`image`*; modified* `heatmap`*)*

Description

Customized heatmap with optional colorbar

Usage

```
mplot3_heatmap(
  x,
  colorGrad.n = 101,
  colorGrad.col = NULL,
  lo = "#18A3AC",
  lomid = NULL,
  mid = NULL,
  midhi = NULL,
  hi = "#F48024",
  space = "rgb",
  theme = getOption("rt.theme", "white"),
  colorbar = TRUE,
  cb.n = 21,
  cb.title = NULL,
  cb.cex = NULL,
  cb.title.cex = 1,
  cb.mar = NULL,
  Rowv = TRUE,
  Colv = TRUE,
  distfun = dist,
  hclustfun = hclust,
  reorderfun = function(d, w) reorder(d, w),
  add.expr,
  symm = FALSE,
  revC = identical(Colv, "Rowv"),
  scale = "none",
  na.rm = TRUE,
  margins = NULL,
  group.columns = NULL,
  group.legend = !is.null(group.columns),
  column.palette = rtPalette,
  group.rows = NULL,
  row.palette = rtPalette,
  ColSideColors,
```

```
  RowSideColors,
  cexRow = 0.2 + 1/log10(nr),
  cexCol = 0.2 + 1/log10(nc),
  labRow = NULL,
  labCol = NULL,
  labCol.las = NULL,
  main = "",
  main.adj = 0,
  main.line = NA,
  xlab = NULL,
  ylab = NULL,
  xlab.line = NULL,
  ylab.line = NULL,
  keep.dendro = FALSE,
  trace = 0,
  zlim = NULL,
  autorange = TRUE,
  filename = NULL,
  par.reset = TRUE,
  pdf.width = 7,
  pdf.height = 7,
  ...
)
```

Arguments

x	Input matrix
colorGrad.n	Integer: Number of distinct colors to generate using colorGrad. Default = 101
colorGrad.col	Character: the colors argument of colorGrad: Character: Acts as a shortcut to defining lo, mid, etc for a number of defaults: "french", "penn", "grnblkred"
lo	Color for low end
lomid	Color for low-mid
mid	Color for middle of the range or "mean", which will result in colorOp(c(lo, hi), "mean"). If mid = NA, then only lo and hi are used to create the color gradient.
midhi	Color for middle-high
hi	Color for high end
space	Character: Which colorspace to use. Option: "rgb", or "Lab". Default = "rgb". Recommendation: If mid is "white" or "black" (default), use "rgb", otherwise "Lab"
theme	Character: Defaults to option "rt.theme", if set, otherwise "light"
colorbar	Logical: If TRUE, plot colorbar next to heatmap. Default = TRUE
cb.n	Integer: Number of steps in colorbar. Default = 21, which gives 10 above and 10 below midline. If midline is zero, this corresponds to 10 percent increments / decrements
cb.title	Character: Title for the colorbar. Default = NULL
cb.cex	Float: Character expansion (cex) for colobar. Default = 1
cb.title.cex	Float: cex for colorbar title. Default = 1
cb.mar	Float, vector, length 4: Margins for colorbar. (passed to colorGrad's cb.add.mar). Default set automatically

Rowv	Logical OR a dendrogram OR integer vector that determines index for reordering OR NA to suppress. Default = TRUE
Colv	See Rowv
distfun	Function: used to compute the distance/dissimilarity matrix between rows and columns. Default = dist
hclustfun	Function: used to determined hierarchical clustering when Rowv or Colv are not dendrograms. Default = hclust (Should take as argument a result of distfun and return an object to which as.dendrogram can be applied)
reorderfun	Function (d, w): function of dendrogram and weights that determines reordering of row and column dendrograms. Default uses reorder.dendrogram
add.expr	Expression: will be evaluated after the call to image. Can be used to add components to the plot
symm	Logical: If TRUE, treat x symmetrically. Can only be used if x is square
revC	Logical: If TRUE, reverse column order for plotting. Default = TRUE, if Rowv and Colv are identical
scale	Character: "row", "column", or "none". Determines whether values are centered and scaled in either the row or column direction. Default = "none"
na.rm	Logical: If TRUE, NAs are removed. Default = TRUE
margins	Float, vector, length 2: bottom and right side margins. Automatically determined by length of variable names
ColSideColors	Color, vector, length = ncol(x): Colors for a horizontal side bar to annotate columns of x
RowSideColors	Color, vector, length = nrow(x): Like ColSideColors, but for rows
cexRow	Float: cex.axis for rows
cexCol	Float: cex.axis for columns
labRow	Character, vector: Row labels to use. Default = rownames(x)
labCol	Character, vector: Column labels to use. Default = colnames(x)
labCol.las	Integer 0:3: par's las argument. Default set by length of labCol
main	Character: Plot title
main.adj	Float: par's adj argument for title
main.line	Float: title's line argument
xlab	Character: x-axis label
ylab	Character: y-axis label
xlab.line	Float: mtext's line argument for x-axis label
ylab.line	Float: mtext's line argument for y-axis label
keep.dendro	Logical: If TRUE, dedrogram is returned invisibly. Default = FALSE
trace	Integer: If > 0, print diagnostic messages to console. Default = 0
zlim	Float, vector, length 2: Passed to graphics::image. Default = +/- max(abs(x)) if autorange = TRUE, otherwise = range(x).
autorange	Logical: See zlim
filename	Character: If provided, save heatmap to file. Default = NULL
par.reset	Logical: If TRUE, reset par before exit. Default = TRUE
pdf.width	Float: Width of PDF output, if filename is set
pdf.height	Float: Height of PDF output, if filename is set
...	Additional arguments passed to graphics::image

Details

The main difference from the original `stats::heatmap` is the addition of a colorbar on the side. This is achieved with colorGrad. Other differences: - Dendrograms are not drawn by default. Set `Rowv = T` and `Colv = T` to get them. - Column labels are only drawn perpendicular to the x-axis if any one is longer than two characters. Otherwise, the arguments are the same as in `stats::heatmap`

Author(s)

E.D. Gennatas modified from original `stats::heatmap` by Andy Liaw, R. Gentleman, M. Maechler, W. Huber

Examples

```
## Not run:
x <- rnormmat(200, 20)
xcor <- cor(x)
mplot3_heatmap(xcor)

## End(Not run)
```

`mplot3_img` *Draw image (False color 2D)*

Description

Draw a bitmap from a matrix of values.

Usage

```
mplot3_img(
  z,
  as.mat = TRUE,
  col = NULL,
  xnames = NULL,
  xnames.y = 0,
  ynames = NULL,
  main = NULL,
  main.adj = 0,
  x.axis.side = 3,
  y.axis.side = 2,
  x.axis.line = -0.5,
  y.axis.line = -0.5,
  x.axis.las = 0,
  y.axis.las = 1,
  x.tick.labs.adj = NULL,
  y.tick.labs.adj = NULL,
  x.axis.font = 1,
  y.axis.font = 1,
  xlab = NULL,
  ylab = NULL,
  xlab.adj = 0.5,
```

```
  ylab.adj = 0.5,
  xlab.line = 1.7,
  ylab.line = 1.7,
  xlab.padj = 0,
  ylab.padj = 0,
  xlab.side = 1,
  ylab.side = 2,
  main.col = NULL,
  axlab.col = NULL,
  axes.col = NULL,
  labs.col = NULL,
  tick.col = NULL,
  cell.lab.hi.col = NULL,
  cell.lab.lo.col = NULL,
  cex = 1.2,
  cex.ax = NULL,
  cex.x = NULL,
  cex.y = NULL,
  zlim = NULL,
  autorange = TRUE,
  pty = "m",
  mar = NULL,
  asp = NULL,
  ann = FALSE,
  axes = FALSE,
  cell.labs = NULL,
  cell.labs.col = NULL,
  cell.labs.autocol = TRUE,
  bg = NULL,
  theme = getOption("rt.theme", "white"),
  autolabel = letters,
  filename = NULL,
  file.width = NULL,
  file.height = NULL,
  par.reset = TRUE,
  ...
)
```

Arguments

z	Input matrix
as.mat	Logical: If FALSE, rows and columns of z correspond to x and y coordinates accoridngly. This is the image default. If TRUE (default), resulting image's cells will be in the same order as values appear when you print z in the console. This is t(apply(z, 2, rev)). In this case, you can think of z as a table of values you want to pictures with colors. For example, you can convert a correlation table to a figure. In this case, you might want to add cell.labs with the values. Consider first using ddSci.
col	Colors to use. Defaults to colorGrad(100)
cell.labs	Matrix of same dimensions as z (Optional): Will be printed as strings over cells
cell.labs.col	Color for cell.labs. If NULL, the upper and lower quartiles will be set to "white", the rest "black".

bg	Background color
filename	String (Optional): Path to file where image should be saved. R-supported extensions: ".pdf", ".jpeg", ".png", ".tiff".
file.width	Output Width in inches
file.height	Output height in inches
par.reset	Logical: If TRUE, par will be reset to original settings before exit. Default = TRUE
...	Additional arguments to be passed to graphics::image
revR	Logical: If TRUE, reverse rows. Defaults to TRUE for a top-left to bottom-right diagonal

Details

This is also a good way to plot a large heatmap. This function calls image which is a lot faster than drawing heatmaps

Author(s)

E.D. Gennatas

---

mplot3_laterality *Laterality scatter plot*

Description

Laterality scatter plot

Usage

```
mplot3_laterality(
  x,
  regionnames,
  main = NULL,
  ylab = "Left to Right",
  summary.fn = "median",
  summary.lty = 1,
  summary.lwd = 2.5,
  summary.col = NULL,
  arrowhead.length = 0.075,
  deltas = TRUE,
  line.col = theme$fg,
  line.alpha = 0.25,
  lty = 1,
  lwd = 0.3,
  ylim = NULL,
  theme = rtTheme,
  labelify = TRUE,
  autolabel = letters,
  mar = NULL,
```

```
  oma = rep(0, 4),
  pty = "m",
  palette = rtPalette,
  par.reset = TRUE,
  pdf.width = 6,
  pdf.height = 6,
  filename = NULL,
  ...
)
```

Arguments

x	data.frame or data.table which includes columns with ROI names ending in "_L" or "_R"
regionnames	Character, vector: Regions to plot. For example, if regionnames contains "Ant_Insula", x must contain columns Ant_Insula_L and Ant_Insula_R
main	Character: Plot title
ylab	Character: y-axis label
summary.fn	Character: Name of function to summarize left and right values. Default = "median"
summary.lty	Integer: line type for summary arrows
summary.lwd	Float: line width for summary arrows
summary.col	Color for summary arrows
arrowhead.length	Float: arrowhead length in inches. Default = .075
deltas	Logical, If TRUE, show summary statistics. Default = TRUE
line.col	Color for individual cases' lines
line.alpha	Float: transparency for individual lines
lty	Integer: Line type for individual lines. Default = 1
lwd	Float: Line width for individual lines. Default = .3
ylim	Float, vector, length 2: y-axis limits
theme	Character: Run themes() for available themes
autolabel	Vector to be used to generate autolabels when using rtlayout with autolabel = TRUE. Default = letters
mar	Float, vector, length 4: Margins; see par("mar")
oma	Float, vector, length 4: Outer margins; see par("oma")
pty	Character: "s" gives a square plot; "m" gives a plot that fills graphics device size. Default = "m" (See par("pty"))
palette	Vector of colors, or Character defining a builtin palette - get options with rtpalette()
par.reset	Logical: If TRUE, reset par setting before exiting.
pdf.width	Float: Width in inches for pdf output (if filename is set).
pdf.height	Float: Height in inches for pdf output.
filename	Character: Path to file to save plot. Default = NULL
...	Additional arguments to be passed to theme function

Author(s)

E.D. Gennatas

mplot3_lolli mplot3 *Lollipop Plot*

Description

mplot3 Lollipop Plot

Usage

```
mplot3_lolli(
  x,
  order.on.x = TRUE,
  plot.top = 1,
  orientation = c("horizontal", "vertical"),
  xnames = NULL,
  points = TRUE,
  segments = TRUE,
  main = NULL,
  col = NULL,
  cex = 1.2,
  matching.segment.col = FALSE,
  segment.alpha = 0.333,
  lty = 3,
  lwd = 2,
  theme = rtTheme,
  palette = rtPalette,
  autolabel = letters,
  par.reset = TRUE,
  pdf.width = 6,
  pdf.height = 6,
  mar = c(2.5, 3, 2, 1),
  pty = "m",
  pch = 16,
  x.axis.at = NULL,
  y.axis.at = NULL,
  xlab = NULL,
  ylab = NULL,
  label.las = 1,
  label.padj = 0.5,
  xaxs = "r",
  yaxs = "r",
  xlab.adj = 0.5,
  ylab.adj = 0.5,
  filename = NULL,
  ...
)
```

Arguments

x	Float, vector: Input data
order.on.x	Logical: If TRUE, order by value of x. Default = TRUE

plot.top	Float or Integer: If <= 1, plot this percent highest absolute values, otherwise plot this many top values. i.e.: plot.top = .2 will print the top 20 highest values
xnames	Character, vector: Names of x
main	Character: Main title
col	Color, vector: Lollipop color
cex	Float: Character expansion factor for points. Default = 1.2
matching.segment.col	Logical: If TRUE, color line segments using col, i.e. same as
segment.alpha	Float: Transparency for line segments. Default = .5 points. Default = FALSE, in which case they are colored with theme$fg
lty	Integer: Line type for segment segments. See par("lty") Default = 1
lwd	Float: Width for segment segments. See par("lty") Default = 1
theme	Character: Run themes() for available themes
palette	Vector of colors, or Character defining a builtin palette - get options with rtpalette()
autolabel	Vector to be used to generate autolabels when using rtlayout with autolabel = TRUE. Default = letters
par.reset	Logical: If TRUE, reset par setting before exiting.
pdf.width	Float: Width in inches for pdf output (if filename is set).
pdf.height	Float: Height in inches for pdf output.
mar	Float, vector, length 4: Margins; see par("mar")
pty	Character: "s" gives a square plot; "m" gives a plot that fills graphics device size. Default = "m" (See par("pty"))
pch	Integer: Point character. Default = 16
x.axis.at	Float, vector: x coordinates to place tick marks. Default = NULL, determined by graphics::axis aautomatically
y.axis.at	As x.axis.at for y-axis
xlab	Character: x-axis label
ylab	Character: y-axis label
xaxs	Character: "r": Extend plot x-axis limits by 4 "i": Use exact x-axis limits. Default = "r"
yaxs	Character: as xaxs for the y-axis. Default = "r"
xlab.adj	Float: adj for xlab (See par("adj"))
ylab.adj	Float: adj for ylab (See par("adj"))
filename	Character: Path to file to save plot. Default = NULL
...	Additional arguments to be passed to theme function

Author(s)

E.D. Gennatas

Examples

```
## Not run:
x <- rnorm(12)
mplot3_lolli(x)
# a "rounded" barplot
mplot3_lolli(x, segments = T, points = F,
             lty = 1, matching.segment.col = T,
             lwd = 10, segment.alpha = 1)

## End(Not run)
```

`mplot3_missing` *Plot missingness*

Description

Plot missingness

Usage

```
mplot3_missing(
  x,
  feat.names = NULL,
  case.names = NULL,
  main = NULL,
  col.missing = "#FE4AA3",
  show = c("percent", "total"),
  names.srt = 90,
  case.names.x = 0.25,
  case.names.every = NULL,
  theme = rtTheme,
  use.alpha = FALSE,
  mar = c(3, 3.5, 5.5, 1),
  oma = c(0.5, 0.5, 0.5, 0.5),
  par.reset = TRUE,
  ...
)
```

Arguments

`x`	Data matrix or data.frame
`feat.names`	Character: Feature names. Defaults to `colnames(x)`
`case.names`	Character: Case names. Defaults to `rownames(x)`
`main`	Character: Main title
`col.missing`	Color for missing cases. Default = "#FE4AA3" (magenta)
`show`	Character: "percent" or "total". Show percent missing or total missing per column on the x-axis
`theme`	Character: Run `themes()` for available themes
`mar`	Float, vector, length 4: Margins; see `par("mar")`

oma	Float, vector, length 4: Outer margins; see par("oma")
par.reset	Logical: If TRUE, reset par setting before exiting.
...	Additional arguments to be passed to theme function

Examples

```
## Not run:
dat <- iris
dat[c(1, 5, 17:20, 110, 115, 140), 1] <-
dat[c(12, 15, 55, 73, 100:103), 2] <-
dat[sample(1:150, 25), 4] <- NA
mplot_missing(dat)

## End(Not run)
```

---

mplot3_mosaic *Mosaic plot*

Description

Plots a mosaic plot using graphics::mosaicplot

Usage

```
mplot3_mosaic(
  x,
  main = NULL,
  xlab = NULL,
  ylab = NULL,
  border = FALSE,
  theme = rtTheme,
  theme.args = list(),
  palette = rtPalette,
  mar = NULL,
  oma = rep(0, 4),
  par.reset = TRUE,
  new = FALSE,
  filename = NULL,
  pdf.width = 5,
  pdf.height = 5,
  ...
)
```

Arguments

x	contingency table, e.g. output of table()
main	Character: Main title
xlab	Character: x-axis label
ylab	Character: y-axis label

border	Color vector for cell borders or FALSE to turn off. Default = FALSE
theme	Character: Run themes() for available themes
theme.args	List of arguments to pass to theme. Optional, same args can be passed to theme function
palette	Vector of colors, or Character defining a builtin palette - get options with rtpalette()
new	Logical: If TRUE, add plot to existing plot. See par("new")
filename	Character: Path to file to save plot. Default = NULL
pdf.width	Float: Width in inches for PDF output, if filename is defined
pdf.height	Float: Height in inches for PDF output, if filename is defined

Author(s)

E.D. Gennatas

Examples

```
## Not run:
party <- as.table(rbind(c(762, 327, 468), c(484, 239, 477)))
dimnames(party) <- list(gender = c("F", "M"),
                        party = c("Democrat","Independent", "Republican"))
mplot3_mosaic(party)

## End(Not run)
```

---

mplot3_pr *mplot3 Precision Recall curves*

Description

Plot Precision Recall curve for a binary classifier

Usage

```
mplot3_pr(
  prob,
  labels,
  f1 = FALSE,
  main = "",
  col = NULL,
  cex = 1.2,
  lwd = 2.5,
  diagonal = TRUE,
  diagonal.lwd = 2.5,
  diagonal.lty = 3,
  group.legend = FALSE,
  annotation = TRUE,
  annotation.col = col,
  annot.line = NULL,
  annot.adj = 1,
  annot.font = 1,
```

```
  mar = c(2.5, 3, 2.5, 1),
  theme = getOption("rt.theme", "whitegrid"),
  palette = rtPalette,
  par.reset = TRUE,
  verbose = TRUE,
  filename = NULL,
  pdf.width = 5,
  pdf.height = 5,
  ...
)
```

Arguments

prob	Vector, Float [0, 1]: Predicted probabilities (i.e. c(.1, .8, .2, .9))
labels	Vector, Integer 0, 1: True labels (i.e. c(0, 1, 0, 1))
f1	Logical: If TRUE, annotate the point of maximal F1 score. Default = FALSE
main	Character: Plot title. Default = ""
col	Color, vector: Colors to use for ROC curve(s)
cex	Float: Character expansion factor. Default = 1.2
lwd	Float: Line width. Default = 2.5
diagonal	Logical: If TRUE, draw diagonal. Default = TRUE
diagonal.lwd	Float: Line width for diagonal. Default = 2.5
diagonal.lty	Integer: Line type for diagonal. Default = 1
group.legend	Logical
annotation	Character: Add annotation at the bottom right of the plot
mar	Float, vector, length 4: Margins; see par("mar")
theme	Character: Run themes() for available themes
palette	Vector of colors, or Character defining a builtin palette - get options with rtpalette()
par.reset	Logical: If TRUE, reset par setting before exiting.
filename	Path to file: If supplied, plot will be printed to file
pdf.width	Float: Width in inches for pdf output (if filename is set).
pdf.height	Float: Height in inches for pdf output.
...	Additional parameters to pass to mplot3_xy

Value

List with Precision, Recall, and Threshold values, invisibly

Author(s)

E.D. Gennatas

mplot3_prp *Plot CART Decision Tree*

Description

Plot output of a regression or classification tree created using `rpart` A wrapper for `rpart.plot::rpart.plot`

Usage

```
mplot3_prp(
  object,
  type = 0,
  extra = "auto",
  branch.lty = 1,
  under = FALSE,
  fallen.leaves = TRUE,
  palette = NULL,
  filename = NULL,
  pdf.width = 7,
  pdf.height = 5,
  ...
)
```

Arguments

`object`	Output of s_CART
`palette`	Color vector

mplot3_res `mplot3` *Plot* `resample`

Description

Visualizes resampling output using mplot3_img

Usage

```
mplot3_res(res, col = NULL, mar = NULL, theme = rtTheme, ...)
```

Arguments

`res`	rtemis resample object
`col`	Color vector
`mar`	Numeric vector: image margins
`theme`	rtemis theme
`...`	Additional theme arguments

Details

For resampling with no replacement where each case may be selected 0 or 1 time, 0 is white and 1 is teal For resampling with replacement, 0 is white, 1 is blue, 2 is teal

Author(s)

E.D. Gennatas

Examples

```
## Not run:
x <- rnorm(500)
res <- resample(x)
mplot3_res(res)

## End(Not run)
```

---

mplot3_roc mplot3 *ROC curves*

Description

Plot ROC curve for a binary classifier

Usage

```
mplot3_roc(
  prob,
  labels,
  method = c("rt", "pROC"),
  type = "TPR.FPR",
  balanced.accuracy = FALSE,
  main = "",
  col = NULL,
  cex = 1.2,
  lwd = 2.5,
  diagonal = TRUE,
  diagonal.lwd = 2.5,
  diagonal.lty = 3,
  group.legend = FALSE,
  annotation = TRUE,
  annotation.col = col,
  annot.line = NULL,
  annot.adj = 1,
  annot.font = 1,
  pty = "s",
  mar = c(2.5, 3, 2, 1),
  theme = getOption("rt.theme", "whitegrid"),
  palette = rtPalette,
  verbose = TRUE,
  par.reset = TRUE,
```

```
  filename = NULL,
  pdf.width = 5,
  pdf.height = 5,
  ...
)
```

Arguments

prob	Vector, Float [0, 1]: Predicted probabilities (i.e. c(.1, .8, .2, .9))
labels	Vector, Integer 0, 1: True labels (i.e. c(0, 1, 0, 1))
method	Character: "rt" or "pROC" will use rtROC and pROC::roc respectively to get points of the ROC. Default = "rt"
type	Character: "TPR.FPR" or "Sens.Spec". Only changes the x and y labels. True positive rate vs. False positive rate and Sensitivity vs. Specificity. Default = "TPR.FPR"
balanced.accuracy	Logical: If TRUE, annotate the point of maximal Balanced Accuracy. Default = FALSE
main	Character: Plot title. Default = ""
col	Color, vector: Colors to use for ROC curve(s)
cex	Float: Character expansion factor. Default = 1.2
lwd	Float: Line width. Default = 2.5
diagonal	Logical: If TRUE, draw diagonal. Default = TRUE
diagonal.lwd	Float: Line width for diagonal. Default = 2.5
diagonal.lty	Integer: Line type for diagonal. Default = 1
group.legend	Logical
annotation	Character: Add annotation at the bottom right of the plot
pty	Character: "s" gives a square plot; "m" gives a plot that fills graphics device size. Default = "m" (See par("pty"))
mar	Float, vector, length 4: Margins; see par("mar")
theme	Character: Run themes() for available themes
palette	Vector of colors, or Character defining a builtin palette - get options with rtpalette()
par.reset	Logical: If TRUE, reset par setting before exiting.
filename	Path to file: If supplied, plot will be printed to file
pdf.width	Float: Width in inches for pdf output (if filename is set).
pdf.height	Float: Height in inches for pdf output.
...	Additional parameters to pass to mplot3_xy

Author(s)

E.D. Gennatas

mplot3_surv mplot3: *Survival Plots*

Description

Plots survival step functions using mplot3_xy

Usage

```
mplot3_surv(
  x,
  lty = 1,
  lwd = 2,
  alpha = 1,
  col = NULL,
  mark.censored = TRUE,
  normalize.time = FALSE,
  cex = 1.2,
  xlab = NULL,
  ylab = "Survival",
  main = "Kaplan-Meier curve",
  theme = rtTheme,
  palette = rtPalette,
  plot.error = FALSE,
  error.lty = 2,
  error.alpha = 0.5,
  group.legend = NULL,
  group.title = "",
  group.names = NULL,
  group.side = 3,
  group.adj = 0.98,
  group.padj = 2,
  group.at = NA,
  par.reset = TRUE,
  ...
)
```

Arguments

x	Survival object / list of Survival objects created using survival::Surv
lty	Integer: Line type. Default = 1. See par("lty")
lwd	Float: Line width. Default = 2
alpha	Float: Alpha for lines. Default = 1
normalize.time	Logical: If TRUE, convert each input's time to 0-1 range. This is useful when survival estimates are not provided in original time scale. Default = FALSE.
xlab	Character: x-axis label
ylab	Character: y-axis label
main	Character: Plot title
theme	Character: Run themes() for available themes

palette	Vector of colors, or Character defining a builtin palette - get options with rtpalette()
group.legend	Logical: If TRUE, place group.names in a legend
group.title	Character: Group title, shown above group names. e.g. if group names are c("San Francisco", "Philadelphia"), group.title can be "City"
group.names	(Optional) If multiple groups are plotted, use these names if group.title = TRUE
group.side	Integer: Side to show group legend
group.adj	Float: adj for group legend. See mtext("adj")
group.padj	Float: padj for group legend See mtext("padj")
group.at	Float: location for group legend. See mtext("at")
par.reset	Logical: If TRUE, reset par setting before exiting.
...	Additional arguments to pass to mplot3_xy

Author(s)

E.D. Gennatas

Examples

```
## Not run:
library(survival)
mplot3_surv(Surv(time = lung$time, event = lung$status))

## End(Not run)
```

---

mplot3_survfit mplot3: *Plot* survfit *objects*

Description

Plots survival step functions using mplot3_xy

Usage

```
mplot3_survfit(
  x,
  lty = 1,
  lwd = 1.5,
  alpha = 1,
  col = NULL,
  plot.median = FALSE,
  group.median = FALSE,
  median.lty = 3,
  median.lwd = 2,
  median.col = theme$fg,
  median.alpha = 0.5,
  censor.mark = TRUE,
  censor.col = NULL,
  censor.alpha = 0.4,
```

```
  censor.pch = "I",
  censor.cex = 0.8,
  mark.censored = FALSE,
  nrisk.table = FALSE,
  nrisk.pos = "below",
  nrisk.spacing = 0.9,
  table.font = 1,
  time.at = NULL,
  time.by = NULL,
  xlim = NULL,
  ylim = NULL,
  xlab = "Time",
  ylab = "Survival",
  main = "",
  theme = rtTheme,
  palette = rtPalette,
  plot.error = FALSE,
  error.alpha = 0.33,
  autonames = TRUE,
  group.legend = NULL,
  group.legend.type = c("legend", "mtext"),
  group.names = NULL,
  group.title = NULL,
  group.line = NULL,
  group.side = NULL,
  legend.x = NULL,
  mar = c(2.5, 3, 2, 1),
  oma = NULL,
  par.reset = TRUE,
  ...
)
```

Arguments

x	survfit object (output of survival::survfit)
lty	Integer: Line type. Default = 1. See par("lty")
lwd	Float: Line width. Default = 2
alpha	Float: Alpha for lines. Default = 1
col	Color, vector: Color(s) to use for survival curves and annotations. If NULL, taken from palette
plot.median	Logical: If TRUE, draw lines at 50"%" median survival. Default = FALSE,
group.median	Logical: If TRUE, include median survival times with group legend
median.lty	Integer: Median survival line type
median.lwd	Float: Median line width. Default = 2
median.col	Color for median survival lines
median.alpha	Float, (0, 1): Transparency for median survival lines. Default = .5
censor.mark	Logical: If TRUE, mark each censored case. Default = TRUE
censor.col	Color to mark censored cases if censor.mark = TRUE
censor.alpha	Transparency for censor.col. Default = .4

censor.pch	Character: Point character for censored marks. Default = "I"
censor.cex	Float: Character expansion factor for censor marks. Default = .8
mark.censored	Logical: This is an alternative to censor.mark which whill mark censored cases using the same color as the survival curve. It can be harder to distinguish the censoring marks from the curve itself, therefore not preferred.
nrisk.table	Logical: If TRUE, print Number at risk table. Default = FALSE
nrisk.pos	Character: "above" or "below": where to place nrisk.table
nrisk.spacing	Float: Determines spacing between nrisk.table rows. Default = .9
table.font	Integer: 1: regular font, 2: bold. Default = 1
time.at	Float, vector: x-axis positions to place tickmarks and labels as well as n at risk values if nrisk.table = TRUE
time.by	Float: Divide time by this amount to determine placing of tickmarks
xlim	Float, vector, length 2: x-axis limits
ylim	Float, vector, length 2: y-axis limits
xlab	Character: x-axis label
ylab	Character: y-axis label
main	Character: main title
theme	Character: Run themes() for available themes
palette	Vector of colors, or Character defining a builtin palette - get options with rtpalette()
autonames	Logical: If TRUE, extract grouping variable names and level labels from x and use for legend. It is best to give informative level labels, like female, male instead of 0, 1 when using this. Default = TRUE
group.legend	Logical: If TRUE, include group legend
group.names	Character, vector: Group names to use. If NULL, extracted from x
group.title	Character: Group legend title
group.line	Float, vector: Lines to print group legend using mtext
group.side	Integer: Side to print group legend. Default is determined by survival curves, to avoid overlap of legend with curves.
mar	Float, vector, length 4: Margins. See par("mar")
oma	Float, vector, length 4: Outer margins. See par("oma")
par.reset	Logical: If TRUE, reset par to initial values before exit
...	Additional arguments to pass to theme

Author(s)

E.D. Gennatas

Examples

```
## Not run:
# Get the lung dataset
data(cancer, package = "survival")
sf1 <- survival::survfit(survival::Surv(time, status) ~ 1, data = lung)
mplot3_survfit(sf1)
sf2 <- survival::survfit(survival::Surv(time, status) ~ sex, data = lung)
mplot3_survfit(sf2)
```

```
# with N at risk table
mplot3_survfit(sf2, nrisk.table = TRUE)

## End(Not run)
```

`mplot3_varimp` `mplot3`*: Variable Importance*

Description

Draw horizontal barplots for variable importance

Usage

```
mplot3_varimp(
  x,
  error = NULL,
  names = NULL,
  names.pad = 0.02,
  plot.top = 1,
  labelify = TRUE,
  col = NULL,
  palette = NULL,
  alpha = 1,
  error.col = theme$fg,
  error.lwd = 2,
  beside = TRUE,
  border = NA,
  width = 1,
  space = 0.75,
  xlim = NULL,
  ylim = NULL,
  xlab = "Variable Importance",
  xlab.line = 1.3,
  ylab = NULL,
  ylab.line = 1.5,
  main = NULL,
  names.arg = NULL,
  axisnames = FALSE,
  sidelabels = NULL,
  mar = NULL,
  pty = "m",
  barplot.axes = FALSE,
  xaxis = TRUE,
  x.axis.padj = -1.2,
  tck = -0.015,
  theme = rtTheme,
  zerolines = FALSE,
  par.reset = TRUE,
  pdf.width = NULL,
  pdf.height = NULL,
```

```
  trace = 0,
  filename = NULL,
  ...
)
```

Arguments

x	Vector, numeric: Input
error	Vector, numeric; length = length(x): Plot error bars with given error.
names	Vector, string; optional: Names of variables in x
plot.top	Float or Integer: If <= 1, plot this percent highest absolute values, otherwise plot this many top values. i.e.: plot.top = .2 will print the top 20 highest values
labelify	Logical: If TRUE convert names(x) using labelify. Default = TRUE
col	Colors: Gradient to use for barplot fill.
alpha	Float (0, 1): Alpha for col
error.col	Color: For error bars
trace	Integer: If trace > 0 prints out the automatically set mar (so you can adjust if needed) names provided

Details

"NA" values in input are set to zero.

Value

Position of bar centers (invisibly)

Author(s)

E.D. Gennatas

---

mplot3_x *mplot3: Univariate plots: index, histogram, density, QQ-line*

Description

Draw plots of 1-dimensional data: index, histogram, density, and Q-Q plots.

Usage

```
mplot3_x(
  x,
  type = c("density", "histogram", "hd", "lhist", "index", "ts", "qqline"),
  group = NULL,
  data = NULL,
  xlab = NULL,
  ylab = NULL,
  main = NULL,
  xlim = NULL,
```

```
    ylim = NULL,
    index.ypad = 0.1,
    axes.swap = FALSE,
    axes.col = NULL,
    tick.col = NULL,
    cex = 1.2,
    col = NULL,
    alpha = 0.75,
    index.type = c("p", "l"),
    hist.breaks = "Sturges",
    hist.type = c("bars", "lines"),
    hist.probability = FALSE,
    hist.lwd = 3,
    density.line = FALSE,
    density.shade = TRUE,
    density.legend.side = 3,
    density.legend.adj = 0.98,
    density.bw = "nrd0",
    density.kernel = "gaussian",
    density.params = list(na.rm = na.rm),
    qqline.col = "#18A3AC",
    qqline.alpha = 1,
    pch = 16,
    point.col = NULL,
    point.cex = 1,
    point.bg.col = NULL,
    point.alpha = 0.66,
    hline = NULL,
    vline = NULL,
    diagonal = FALSE,
    grid = FALSE,
    grid.col = NULL,
    grid.alpha = 0.5,
    grid.lty = 3,
    grid.lwd = 1,
    annotation = NULL,
    annot.col = NULL,
    group.legend = NULL,
    group.title = "",
    group.names = NULL,
    group.side = 3,
    group.adj = 0.02,
    group.at = NA,
    text.xy = NULL,
    text.x = NULL,
    text.y = NULL,
    text.xy.cex = 1,
    text.xy.col = "white",
    line.col = "#008E00",
    x.axis.padj = -1.1,
    y.axis.padj = 0.9,
    labs.col = NULL,
```

```
  lab.adj = 0.5,
  density.avg = ifelse(type == "density", TRUE, FALSE),
  density.avg.fn = c("median", "mean"),
  density.avg.line = FALSE,
  density.avg.lwd = 1.5,
  density.avg.lty = 3,
  hline.col = "black",
  hline.lwd = 1,
  hline.lty = 1,
  vline.col = "black",
  vline.lwd = 1,
  vline.lty = 1,
  lty = 1,
  lwd = 2,
  qqline.lwd = lwd,
  density.lwd = lwd,
  theme = rtTheme,
  palette = rtPalette,
  pty = "m",
  mar = NULL,
  oma = rep(0, 4),
  xaxs = "r",
  yaxs = "r",
  autolabel = letters,
  new = FALSE,
  alpha.off = FALSE,
  na.rm = TRUE,
  par.reset = TRUE,
  filename = NULL,
  pdf.width = 6,
  pdf.height = 6,
  ...
)
```

Arguments

x	Numeric vector or list of vectors, one for each group. If data is provided, x is name of variable in data
type	Character: "density", "histogram", "hd" (histogram bars & density lines), "lhist" (line histogram like mhist; same as type = "hist", hist.type = "lines"), "index", "ts", "qqline" Case-insensitive and supports partial matching: e.g. mplot3_x(x, "H") gives histogram
group	Vector denoting group membership. Will be converted to factor. If data is provided, group is name of variable if data
data	Optional data frame containing x data
xlab	Character: x-axis label
ylab	Character: y-axis label
main	Character: Plot title
xlim	Float vector, length 2: x-axis limits
ylim	Float vector, length 2: y-axis limits

index.ypad	Float: Expand ylim by this much for plot type "index" Default = .1 (Stops points being cut off)
index.type	Character: "p" for points (Default), "l" for lines (timeseries)
hist.breaks	See histogram("breaks")
hist.type	Character: "bars" or "lines". Default = "bars"
hist.lwd	Float: Line width for type = "histogram"; hist.type = "lines"
density.line	Logical: If TRUE, draw line for type = "density". Default = FALSE
density.shade	Logical: If TRUE, draw shaded polygon for type = "density". Default = TRUE
qqline.col	Color for Q-Q line
qqline.alpha	Float: Alpha for Q-Q line
pch	Integer: Point character. Default = 16
point.cex	Float: Character expansion for points. Default = 1
point.bg.col	Color: point background
hline	Vector: y-value(s) for horizontal lines. Default = NULL
vline	Vector: x-value(s) for vertical lines. Default = NULL
diagonal	Logical: If TRUE, draw diagonal line. Default = FALSE
annotation	Character: Add annotation at the bottom right of the plot
group.legend	Logical: If TRUE, include legend with group names
group.title	Character: Title above group names
group.names	(Optional) If multiple groups are plotted, use these names if group.title = TRUE
group.side	Integer: Side to show group legend
group.adj	Float: adj for group legend. See mtext("adj")
group.at	Float: location for group legend. See mtext("at")
line.col	Color for lines
labs.col	Color for labels
lab.adj	Adjust the axes labels. 0 = left adjust; 1 = right adjust; .5 = center (Default)
density.avg	Logical: If TRUE, print mean of x along plot. Default = TRUE, for type = "density"
density.avg.fn	Character: "median" or "mean". Function to use if density.avg = TRUE. Default = "median"
density.avg.line	Logical: If TRUE, draw vertical lines at the density average x-value
density.avg.lwd	Float: Line width for density.avg.line. Default = 1.5
density.avg.lty	Integer: Line type for density.avg.line. Default = 3
hline.col	Color for horizontal line(s)
hline.lwd	Float: Width for horizontal line(s)
hline.lty	Integer: Line type for horizontal line(s)
vline.col	Color for vertical lines
vline.lwd	Float: Width for vertical lines

vline.lty	Integer: Line type for vertical lines
lty	Integer: Line type. See par("lty")
lwd	Integer: Line width. Used for type = "ts" or "density"
theme	Character: Run themes() for available themes
palette	Vector of colors, or Character defining a builtin palette - get options with rtpalette()
pty	Character: "s" gives a square plot; "m" gives a plot that fills graphics device size. Default = "m" (See par("pty"))
mar	Float, vector, length 4: Margins; see par("mar")
oma	Float, vector, length 4: Outer margins; see par("oma")
xaxs	Character: "r": Extend plot x-axis limits by 4 "i": Use exact x-axis limits. Default = "r"
yaxs	Character: as xaxs for the y-axis. Default = "r"
autolabel	Vector to be used to generate autolabels when using rtlayout with autolabel = TRUE. Default = letters
new	Logical: If TRUE, add plot to existing plot. See par("new")
na.rm	Logical: Will be passed to all functions that support it. If set to FALSE, input containing NA values will result in error, depending on the type
par.reset	Logical: If TRUE, reset par setting before exiting.
filename	Path to file: If supplied, plot will be printed to file
pdf.width	Float: Width in inches for pdf output (if filename is set).
pdf.height	Float: Height in inches for pdf output.
...	Additional arguments to be passed to theme function

Details

You can group data either by supplying x as a list where each element contains one vector per group, or as a data frame where each column represents group, or by providing a group variable, which will be converted to factor. For bivariate plots, see mplot3_xy and mplot3_xym. For heatmaps, see mplot3_heatmap To plot histograms of multiple groups, it's best to use hist.type = "lines", which will use mhist and space apart the breaks for each group

Value

Invisibly returns the output of density, hist, qqnorm, or NULL

Author(s)

E.D. Gennatas

See Also

mplot3_xy, mplot3_xym, mplot3_heatmap

Examples

```
## Not run:
mplot3_x(iris)
mplot3_x(split(iris$Sepal.Length, iris$Species), xlab = "Sepal Length")

## End(Not run)
```

mplot3_xy mplot3*: XY Scatter and line plots*

Description

Plot points and lines with optional fits and standard error bands

Usage

```
mplot3_xy(
  x,
  y = NULL,
  fit = NULL,
  formula = NULL,
  se.fit = FALSE,
  fit.params = NULL,
  error.x = NULL,
  error.y = NULL,
  cluster = NULL,
  cluster.params = list(),
  data = NULL,
  type = "p",
  group = NULL,
  xlab = NULL,
  ylab = NULL,
  main = NULL,
  xlim = NULL,
  ylim = NULL,
  xpd = TRUE,
  xaxs = "r",
  yaxs = "r",
  log = "",
  rsq = NULL,
  rsq.pval = FALSE,
  rsq.side = 1,
  rsq.adj = 0.98,
  rsq.col = NULL,
  rsq.line = NULL,
  fit.error = FALSE,
  fit.error.side = 1,
  fit.error.padj = NA,
  xaxp = NULL,
  yaxp = NULL,
  scatter = TRUE,
  axes.equal = FALSE,
  pty = "m",
  annotation = NULL,
  annotation.col = NULL,
  tick.col = NULL,
  x.axis.at = NULL,
  x.axis.labs = TRUE,
```

```
y.axis.at = NULL,
y.axis.labs = TRUE,
xlab.adj = 0.5,
ylab.adj = 0.5,
mar = NULL,
oma = rep(0, 4),
point.cex = 1,
point.bg.col = NULL,
pch = ifelse(is.null(point.bg.col), 16, 21),
line.col = NULL,
line.alpha = 0.66,
lty = 1,
marker.col = NULL,
marker.alpha = NULL,
error.x.col = NULL,
error.y.col = NULL,
error.x.lty = 1,
error.y.lty = 1,
error.x.lwd = 1,
error.y.lwd = 1,
error.arrow.code = 3,
fit.col = NULL,
fit.lwd = 2.5,
fit.alpha = 1,
fit.legend = ifelse(is.null(fit), FALSE, TRUE),
se.lty = "poly",
se.lwd = 1,
se.col = NULL,
se.alpha = 0.5,
se.times = 1.96,
se.border = FALSE,
se.density = NULL,
hline = NULL,
hline.col = NULL,
hline.lwd = 1.5,
hline.lty = 3,
vline = NULL,
vline.lwd = 1.5,
vline.col = "blue",
vline.lty = 3,
diagonal = FALSE,
diagonal.inv = FALSE,
diagonal.lwd = 1.5,
diagonal.lty = 1,
diagonal.col = "gray50",
diagonal.alpha = 1,
group.legend = NULL,
group.title = NULL,
group.names = NULL,
group.side = 3,
group.adj = 0.02,
group.padj = 2,
```

```
  group.at = NA,
  fit.legend.col = NULL,
  fit.legend.side = 3,
  fit.legend.adj = 0.02,
  fit.legend.padj = 2,
  fit.legend.at = NA,
  labs.col = NULL,
  rm.na = TRUE,
  theme = rtTheme,
  palette = rtPalette,
  order.on.x = NULL,
  autolabel = letters,
  new = FALSE,
  par.reset = TRUE,
  return.lims = FALSE,
  pdf.width = 6,
  pdf.height = 6,
  trace = 0,
  filename = NULL,
  ...
)
```

Arguments

x	Numeric vector or list of vectors for x-axis. If data is provided, name of variable, unquoted.
y	Numeric vector of list of vectors for y-axis If data is provided, name of variable, unquoted.
fit	Character: **rtemis** model to calculate y ~ x fit. Options: see modSelect Can also be Logical, which will give a GAM fit if TRUE. If you specify "NLA", the activation function should be passed as a string.
formula	Formula: Provide a formula to be solved using s_NLS. If provided, fit is forced to 'nls'. e.g. y ~ b * m ^ x for a power curve. Note: nls is powerful but is prone to errors and warnings. Use single letters for parameter names, no numbers.
se.fit	Logical: If TRUE, draw the standard error of the fit
fit.params	List: Arguments for learner defined by fit. Default = NULL, i.e. use default learner parameters
error.x	Vector, float: Error in x (e.g. standard deviation) will be plotted as bars around point
error.y	Vector, float: Error in y (e.g. standard deviation) will be plotted as bars around point
cluster	Character: Clusterer name. Will cluster data.frame(x, y) and pass result to group. Run clustSelect for options
cluster.params	List: Names list of parameters to pass to the cluster function
data	(Optional) data frame, where x and y are defined
type	Character: "p" for points, "l" for lines, "s" for steps. Default = "p". If x and/or y contains multiple vectors, type can be a vector, e.g. c("p", "l", "l") will give a set of points and two sets of lines. Otherwise, type is recycled to length of x

group	Vector: will be converted to factor. If data is provided, name of variable, unquoted.
xlab	Character: x-axis label
ylab	Character: y-axis label
main	Character: Plot title
xlim	Float vector, length 2: x-axis limits
ylim	Float vector, length 2: y-axis limits
xpd	Logical or NA: FALSE: plotting clipped to plot region; TRUE: plotting clipped to figure region; NA: plotting clipped to device region.
xaxs	Character: "r": Extend plot x-axis limits by 4 "i": Use exact x-axis limits. Default = "r"
yaxs	Character: as xaxs for the y-axis. Default = "r"
rsq	Logical: If TRUE, add legend with R-squared (if fit is not NULL)
rsq.pval	Logical: If TRUE, add legend with R-squared and its p-value, if fit is not NULL
rsq.side	Integer: [1:4] Where to place the rsq annotation. Default = 1 (i.e. bottom)
rsq.adj	Float: Adjust rsq annotation. See mtext "adj"
rsq.col	Color: Color for rsq annotation. Default = NULL, which results in fit.col
fit.error	Logical: If TRUE: draw fit error annotation. Default = NULL, which results in TRUE, if fit is set
fit.error.side	Integer [1:4]: Which side to draw fit.error on.
fit.error.padj	Float: See mtext:padg Default = NA
xaxp	See par("xaxp")
yaxp	See par("yaxp")
scatter	Logical: If TRUE, plot (x, y) scatter points.
axes.equal	Logical: Should axes be equal? Defaults to FALSE
pty	Character: "s" gives a square plot; "m" gives a plot that fills graphics device size. Default = "m" (See par("pty"))
annotation	Character: Add annotation at the bottom right of the plot
annotation.col	Color for annotation
x.axis.at	Float, vector: x coordinates to place tick marks. Default = NULL, determined by graphics::axis aautomatically
x.axis.labs	See axis("labels")
y.axis.at	As x.axis.at for y-axis
y.axis.labs	See axis("labels")
xlab.adj	Float: adj for xlab (See par("adj"))
ylab.adj	Float: adj for ylab (See par("adj"))
mar	Float, vector, length 4: Margins; see par("mar")
oma	Float, vector, length 4: Outer margins; see par("oma")
point.cex	Float: Character expansion for points. Default = 1
point.bg.col	Color: point background
pch	Integer: Point character. Default = 16
line.col	Color for lines

line.alpha	Float [0, 1]: Transparency for lines
lty	Integer: Line type. See par("lty")
marker.col	Color for marker
marker.alpha	Float [0, 1]: Transparency for markers
error.x.col	Color for x-axis error bars
error.y.col	Color for y-axis error bars
error.x.lty	Integer: line type for x-axis error bars
error.y.lty	Integer: line type for y-axis error bars
error.x.lwd	Float: Line width for x-axis error bars
error.y.lwd	Float: Line width for y-axis error bars
error.arrow.code	Integer: Type of arrow to draw for error bars. See arrows("code")
fit.col	Color: Color of the fit line.
fit.lwd	Float: Fit line width
fit.alpha	Float [0, 1]: Transparency for fit line
fit.legend	Logical: If TRUE, show fit legend
se.lty	How to draw the se.fit "poly" draws a polygon around the fit line, otherwise an integer defines the lty (line type) for lines to be drawn
se.lwd	Float: Line width for standard error bounds
se.col	Color for se.fit
se.alpha	Alpha for se.fit
se.times	Draw polygon or lines at +/- se.times * standard error of fit. Defaults to 1.96, which corresponds to 95% confidence interval
se.border	Define border of polygon for se.fit. See border in graphics::polygon
se.density	Density of shading line of polygon for se.fit. See density in graphics::polygon
hline	Vector: y-value(s) for horizontal lines. Default = NULL
hline.col	Color for horizontal line(s)
hline.lwd	Float: Width for horizontal line(s)
hline.lty	Integer: Line type for horizontal line(s)
vline	Vector: x-value(s) for vertical lines. Default = NULL
vline.lwd	Float: Width for vertical lines
vline.col	Color for vertical lines
vline.lty	Integer: Line type for vertical lines
diagonal	Logical: If TRUE, draw diagonal line. Default = FALSE
diagonal.lwd	Float: Line width for diagonal. Default = 1.5
diagonal.lty	Integer: Line type for diagonal. Default = 1
diagonal.col	Color: Color for diagonal. Default = "gray50"
diagonal.alpha	Float: Alpha for diagonal Default = .5
group.legend	Logical: If TRUE, place group.names in a legend
group.title	Character: Group title, shown above group names. e.g. if group names are c("San Francisco", "Philadelphia"), group.title can be "City"

group.names	(Optional) If multiple groups are plotted, use these names if group.title = TRUE
group.side	Integer: Side to show group legend
group.adj	Float: adj for group legend. See mtext("adj")
group.padj	Float: padj for group legend See mtext("padj")
group.at	Float: location for group legend. See mtext("at")
fit.legend.col	Color for fit legend
fit.legend.side	Integer: Side for fit legend
fit.legend.adj	Float: adj for fit legend
fit.legend.padj	Float: padj for fit legend
fit.legend.at	Float: location for fit legend. See mtext("at")
rm.na	Logical: If TRUE, remove all NA values pairwise between x and y. Set to FALSE if you know your data has no missing values.
theme	Character: Run themes() for available themes
palette	Vector of colors, or Character defining a builtin palette - get options with rtpalette()
order.on.x	Logical: If TRUE, order (x, y) by increasing x. Default = NULL: will be set to TRUE if fit is set, otherwise FALSE
autolabel	Vector to be used to generate autolabels when using rtlayout with autolabel = TRUE. Default = letters
new	Logical: If TRUE, add plot to existing plot. See par("new")
par.reset	Logical: If TRUE, reset par setting before exiting.
return.lims	Logical: If TRUE, return xlim and ylim. Default = FALSE
pdf.width	Float: Width in inches for pdf output (if filename is set).
pdf.height	Float: Height in inches for pdf output.
trace	Integer: If > 0, pass verbose = TRUE to the cluster and fit functions, if used. Default = 0
filename	Character: Path to file to save plot. Default = NULL
...	Additional arguments to be passed to theme function
lwd	Float: Linew width

Author(s)

E.D. Gennatas

Examples

```
## Not run:
set.seed(1999)
x <- rnorm(500)
ycu <- x^3 + 12 + rnorm(500)
mplot3_xy(x, ycu)
mplot3_xy(x, ycu, fit = "gam")
ysq <- x^2 + 3 + rnorm(500)
mplot3_xy(x, list(squared = ysq, cubed = ycu), fit = "gam")

## End(Not run)
```

mplot3_xym mplot3 *Scatter plot with marginal density and/or histogram*

Description

Draw a scatter plot with fit line and marginal density and/or histogram

Usage

```
mplot3_xym(
  x,
  y,
  margin = c("histogram", "density", "both"),
  fit = "gam",
  se.fit = TRUE,
  xlim = NULL,
  ylim = NULL,
  col = "#18A3AC",
  density.alpha = 0.66,
  hist.breaks = 30,
  hist.alpha = 0.66,
  hist.space = 0.05,
  hist.lwd = 3,
  lwd = 4,
  main = NULL,
  main.adj = 0,
  axes.density = FALSE,
  pty = "m",
  mar = c(3, 3, 0, 0),
  mar.mar = 0.2,
  axes = TRUE,
  xaxs = "r",
  yaxs = "r",
  theme = rtTheme,
  par.reset = TRUE,
  widths = NULL,
  heights = NULL,
  filename = NULL,
  pdf.width = 7,
  pdf.height = 7,
  ...
)
```

Arguments

x	x-data
y	y-data
margin	What type of marginal plots to draw. Options: "density", "histogram", or "both"
fit	What model to use to draw y ~ x. Options: mode.select()
se.fit	Logical: If TRUE: plot +/- 2 * Standard Error of fit

xlim	Float vector, length 2: x-axis limits
ylim	Float vector, length 2: y-axis limits
col	Color for marginal plots
density.alpha	Alpha for density plots
hist.breaks	Integer. Number of histogram breaks
hist.alpha	Alpha for barplots
hist.space	Space between bars in barplots
lwd	Float: Line width
main	Character: Main title
axes.density	Logical: If TRUE, plot margin plot axes for density (debugging only)
pty	Character: "s" gives a square plot; "m" gives a plot that fills graphics device size. Default = "m" (See par("pty"))
mar	Float, vector, length 4: Margins; see par("mar")
xaxs	Character: "r": Extend plot x-axis limits by 4 "i": Use exact x-axis limits. Default = "r"
yaxs	Character: as xaxs for the y-axis. Default = "r"
theme	Character: Run themes() for available themes
par.reset	Logical. Resest par to original settings
filename	Character: Path to file to save plot. Default = NULL
pdf.width	Float: Width in inches for pdf output (if filename is set).
pdf.height	Float: Height in inches for pdf output.
...	Additional arguments to be passed to mplot3_xy

Details

At the moment, mplot3_xym is the only mplot3 function that does not support

To make wide plot, change 'widths': e.g. widths = c(7, 1)

Author(s)

E.D. Gennatas

Examples

```
## Not run:
x <- rnorm(500)
y <- x^3 + 12 + rnorm(500)
mplot3_xym(x, y)

## End(Not run)
```

`mplot_AGGTEobj` *Plot AGGTEobj object*

Description

Plot AGGTEobj object from the **did** package.

Usage

```
mplot_AGGTEobj(
  x,
  x.factor = 1,
  y.factor = 1,
  error = c("se", "95%ci"),
  main = "Average Effect by Length of Exposure",
  legend.title = "",
  group.names = c("Pre", "Post"),
  xlab = NULL,
  ylab = NULL,
  mar = c(2.5, 3.5, 2, 7),
  theme = rtTheme,
  col = c("#EC1848", "#18A3AC"),
  filename = NULL,
  file.width = 6.5,
  file.height = 5.5,
  par.reset = TRUE,
  ...
)
```

Arguments

`x`	AGGTEobj object
`main`	Character: Plot title
`group.names`	(Optional) If multiple groups are plotted, use these names if `group.title = TRUE`
`xlab`	Character: x-axis label
`ylab`	Character: y-axis label
`mar`	Float, vector, length 4: Margins; see `par("mar")`
`theme`	Character: Run `themes()` for available themes
`filename`	Character: Path to file to save plot. Default = NULL
`par.reset`	Logical: If TRUE, reset `par` setting before exiting.
`...`	Additional arguments to be passed to theme function

Author(s)

E.D. Gennatas

mplot_hsv *Plot HSV color range*

Description

Plot HSV color range

Usage

```
mplot_hsv(
  h.steps = seq(0, 1, 0.0125),
  s.steps = h.steps,
  v = 1,
  alpha = 1,
  pch = 15,
  bg = "black",
  axes = TRUE,
  pty = "s",
  cex = 1,
  mar = c(3, 3, 2, 0.5),
  lab.col = NULL,
  type = c("polar", "cartesian")
)
```

Arguments

h.steps	Float, vector: Hue values to plot. Default = seq(0, 1, .0125)
s.steps	Float, vector: Saturation values to plot. Default = same as h.steps
v	Float: Value. Default = 1
alpha	Float: Alpha. Default = 1
pch	Integer: pch plot parameter. Default = 15 (square)
bg	Color: Background color. Default = "black"
axes	Logical: for type = "cartesian": If TRUE, draw axes. Default = TRUE
pty	Character: for type = "cartesian": "s", "r", par's pty argument. Default = "s" (square plot)
cex	Float: par/plot's cex argument. Default = 1
mar	Float, vector: for type = "cartesian": par's mar argument. Default = c(3, 3, 2, .5)
lab.col	Color: Color for axes and labels. Defaults to inverse of bg, i.e. white if bg is black
type	Character: "cartesian" for square plot, "polar" for radial plot. Default = "polar"

Author(s)

E.D. Gennatas

mplot_raster *Plot Array as Raster Image*

Description

Plots 2D (grayscale) or 3D (color) array as Raster Image

Usage

```
mplot_raster(
  x,
  max.value = max(x),
  mar = NULL,
  main = NULL,
  main.line = 0,
  main.side = 3,
  main.col = "#ffffff",
  main.adj = 0,
  main.font = 2,
  mono = FALSE,
  mono.fn = mean,
  bg = "gray10",
  par.set = TRUE,
  par.reset = TRUE,
  verbose = TRUE
)
```

Arguments

x	Array, 2D or 3D: Input describing grayscale or color image in RGB space
mono	Logical: If TRUE, plot as grayscale using mono.fn to convert RGB to grayscale. Default = FALSE
mono.fn	Function: Apply this function to the array to convert to 2D for grayscale plotting. Default = mean
bg	Color: Background color (around the plotted image when window proportions do not match image). Default = "gray10"
par.reset	Logical: If TRUE, reset par settings before exiting. Default = TRUE
verbose	Logical: If TRUE, print messages to console. Default = TRUE

Author(s)

E.D. Gennatas

Examples

```
## Not run:
img <- imager::load.image("https://www.r-project.org/logo/Rlogo.png")
mplot_raster(img)

## End(Not run)
```

mse *Error functions*

Description

Convenience functions for calculating loss. These can be passed as arguments to learners that support custom loss functions.

Usage

```
mse(x, y, na.rm = TRUE)

msew(x, y, weights = rep(1, length(y)), na.rm = TRUE)

rmse(x, y, na.rm = TRUE)

mae(x, y, na.rm = TRUE)
```

Arguments

x	Vector of True values
y	Vector of Estimated values
na.rm	Logical: If TRUE, remove NA values before computation. Default = TRUE
weights	Float, vector: Case weights

msg *Message with provenance*

Description

Print message to output with a prefix including data and time, and calling function or full call stack

Usage

```
msg(
  ...,
  date = TRUE,
  caller = NULL,
  call.depth = 1,
  caller.id = 1,
  newline = FALSE,
  newline.pre = FALSE,
  as.message = FALSE,
  color = NULL,
  sep = " "
)

msg0(
```

```
  ...,
  date = TRUE,
  call.depth = 1,
  caller.id = 1,
  newline = FALSE,
  newline.pre = FALSE,
  as.message = FALSE,
  color = NULL
)
```

Arguments

`...`	Message to print
`date`	Logical: if TRUE, include date and time in the prefix
`caller`	Character: Name of calling function
`call.depth`	Integer: Print the system call path of this depth. Default = NULL
`caller.id`	Integer: Which function in the call stack to print
`newline`	Logical: If TRUE end with a new line. Default = FALSE
`newline.pre`	Logical: If TRUE begin with a new line. Default = FALSE
`as.message`	Logical: if TRUE, print using `message()`
`color`	Crayon color for message e.g. `crayon::red`
`sep`	Character: Use to separate objects in `...`

Details

If `msg` is called directly from the console, it will print `[interactive>]` in place of the call stack. `msg0`, similar to `paste0`, is `msg(..., sep = "")`

Value

Invisibly: List with call, message, and date

Author(s)

E.D. Gennatas

`multigplot` *Multipanel **ggplot2** plots*

Description

Plot a panel of **gplot2** plots

Usage

```
multigplot(plots = NULL, nrows = NULL, byrow = TRUE)
```

Arguments

plots	List of ggplot2 plots
nrows	Integer: number of rows for panel arrangement. Defaults to number of rows required to plot 2 plots per row
byrow	Logical: If TRUE, draw plots in order provided by row, otherwise by column. Default = TRUE

Author(s)

E.D. Gennatas

---

nCr *n Choose r*

Description

Calculate number of combinations

Usage

```
nCr(n, r)
```

Arguments

n	Integer: Total number of items
r	Integer: Number of items in each combination

Details

In plain language: You have n items. How many different cobinations of r items can you make?

Value

Integer: Number of combinations

Author(s)

E.D. Gennatas

oddsratio *Calculate odds ratio for a 2x2 contingency table*

Description

Calculate odds ratio for a 2x2 contingency table

Usage

```
oddsratio(x, verbose = TRUE)
```

Arguments

x	2x2 contingency table (created with `table(x, y)`, where `x` and `y` are factors with the first level being the control / unaffected / negative)
verbose	Logical: If TRUE, print messages to console

oddsratiotable *Odds ratio table from logistic regression*

Description

Odds ratio table from logistic regression

Usage

```
oddsratiotable(x, confint.method = c("default", "profilelikelihood"))
```

Arguments

x	glm object fit with `family = binomial`
confint.method	"default" or "profilelikelihood"

Value

matrix with 4 columns: OR, 2.5

Author(s)

E.D. Gennatas

Examples

```
## Not run:
ir2 <- iris[51:150, ]
ir2$Species <- factor(ir2$Species)
ir.fit <- glm(Species ~ ., data = ir2, family = binomial)
oddsratiotable(ir.fit)

## End(Not run)
```

oneHot *One hot encoding*

Description

One hot encode a vector or factors in a data.frame

Usage

```
oneHot(x, xname = NULL, verbose = FALSE)

## Default S3 method:
oneHot(x, xname = NULL, verbose = TRUE)

## S3 method for class 'data.frame'
oneHot(x, xname = NULL, verbose = TRUE)

## S3 method for class 'data.table'
oneHot(x, xname = NULL, verbose = TRUE)

oneHot_(x, xname = NULL, verbose = TRUE)
```

Arguments

x	Vector or data.frame
xname	Character: Variable name
verbose	Logical: If TRUE, print messages to console

Details

A vector input will be one-hot encoded regardless of type by looking at all unique values. With data.frame input, only column of type factor will be one-hot encoded. This function is used by preprocess. `oneHot.data.table` operates on a copy of its input. `oneHot_` performs one-hot encoding in-place.

Value

For vector input, a one-hot-encoded matrix, for data.frame frame input, an expanded data.frame where all factors are one-hot encoded

Author(s)

E.D. Gennatas

Examples

```
oneHot(iris) |> head()
ir <- data.table::as.data.table(iris)
ir_oh <- oneHot(ir)
ir_oh
ir <- data.table::as.data.table(iris)
# oneHot_ operates in-place; therefore no assignment is used:
```

```
oneHot_(ir)
ir
```

`onehot2factor` *Convert one-hot encoded matrix to factor*

Description

Convert one-hot encoded matrix to factor

Usage

```
onehot2factor(x, labels = colnames(x))
```

Arguments

`x`	one-hot encoded matrix or data.frame
`labels`	Character vector of level names. Default = `colnames(x)`

Details

If input has a single column, it will be converted to factor and returned

Author(s)

E.D. Gennatas

Examples

```
## Not run:
x <- data.frame(matrix(F, 10, 3))
colnames(x) <- c("Dx1", "Dx2", "Dx3")
x$Dx1[1:3] <- x$Dx2[4:6] <- x$Dx3[7:10] <- T
onehot2factor(x)

## End(Not run)
```

`order_colors` *Order colors*

Description

Order colors by RGB distance

Usage

```
order_colors(x, start_with = 1, order_by = c("similarity", "dissimilarity"))
```

Arguments

x	Vector of colors
start_with	Integer: Which color to output in first position
order_by	Character: "similarity" or "dissimilarity"

Author(s)

E.D. Gennatas

---

palettize *Palettize colors*

Description

Filter and order a set of colors to produce a palette suitable for multicolor plots

Usage

```
palettize(
  x,
  grayscale_hicut = 0.8,
  start_with = "#16A0AC",
  order_by = c("separation", "dissimilarity", "similarity")
)
```

Arguments

x	Color vector
grayscale_hicut	Numeric: exclude colors whose grayscale equivalent is greater than this value
start_with	Integer or color: For integer, start with this color out of x, otherwise find color x closer to this color and place it first
order_by	Character: "similarity" or "dissimilarity"

Author(s)

E.D. Gennatas

partLin **rtemis** *internal: Ridge and Stump*

Description

Fit a linear model on (x, y) and a tree on the residual yhat - y

Usage

```
partLin(
  x1,
  y1,
  lincoef.params = rtset.lincoef(),
  part.minsplit = 2,
  part.xval = 0,
  part.max.depth = 1,
  part.cp = 0,
  minobsinnode.lin = 5,
  verbose = TRUE,
  trace = 0
)
```

partLmw **rtemis** *internal: Ridge and Stump*

Description

Fit a linear model on (x, y) and a tree on the residual yhat - y

Usage

```
partLmw(
  x1,
  y1,
  weights,
  .env,
  minobsinnode.lin,
  lin.type,
  alpha,
  lambda,
  lambda.seq,
  cv.glmnet.nfolds,
  which.cv.glmnet.lambda,
  part.minsplit,
  part.xval,
  part.max.depth,
  part.cp,
  part.minbucket,
  verbose,
  trace
)
```

permute *Create permutations*

Description

Creates all possible permutations

Usage

```
permute(n)
```

Arguments

n	Integer: Length of elements to permute

Details

n higher than 10 will take a while, or may run out of memory in systems with limited RAM

Value

Matrix where each row is a different permutation

plot.massGLM *Plot* massGLM *object*

Description

Plots a massGLM object using dplot3_bar

Usage

```
## S3 method for class 'massGLM'
plot(
  x,
  predictor = NULL,
  main = NULL,
  what = c("coefs", "pvals", "volcano"),
  p.adjust.method = "none",
  p.transform = function(x) -log10(x),
  show = c("all", "signif"),
  pval.hline = c(0.05, 0.001),
  hline.col = "#ffffff",
  hline.dash = "dash",
  hline.annotate = as.character(pval.hline),
  ylim = NULL,
  ylab = NULL,
  group = NULL,
  col.neg = "#43A4AC",
  col.pos = "#FA9860",
```

```
  col.ns = "#7f7f7f",
  theme = rtTheme,
  alpha = NULL,
  volcano.annotate = TRUE,
  volcano.annotate.n = 7,
  volcano.hline = NULL,
  volcano.hline.dash = "dot",
  volcano.hline.annotate = NULL,
  volcano.p.transform = function(x) -log10(x),
  margin = NULL,
  displayModeBar = FALSE,
  trace = 0,
  filename = NULL,
  ...
)
```

Arguments

x	massGLM object
what	Character: "adjusted" or "raw" p-values to plot

Author(s)

E.D. Gennatas

---

plot.resample plot *method for* resample *object*

Description

Run mplot3_res on a resample object

Usage

```
## S3 method for class 'resample'
plot(x, col = NULL, ...)
```

Arguments

x	Vector; numeric or factor: Outcome used for resampling
col	Vector, color
...	Additional arguments passed to mplot3_res

Author(s)

E.D. Gennatas

plot.rtTest *Plot* rtTest *object*

Description

Plot rtTest object

Usage

```
## S3 method for class 'rtTest'
plot(
  x,
  main = NULL,
  mar = NULL,
  uni.type = c("density", "histogram", "hd"),
  boxplot.xlab = FALSE,
  theme = rtTheme,
  par.reset = TRUE,
  ...
)
```

Arguments

x	rtTest object
main	Character: Main title
theme	Character: Run themes() for available themes

Author(s)

E.D. Gennatas

plotly.heat *Heatmap with* plotly

Description

Draw a heatmap using plotly

Usage

```
plotly.heat(
  z,
  x = NULL,
  y = NULL,
  title = NULL,
  col = penn.heat(21),
  xlab = NULL,
  ylab = NULL,
  zlab = NULL,
  transpose = TRUE
)
```

Arguments

z	Input matrix
x, y	Vectors for x, y axes
title	Plot title
col	Set of colors to make gradient from
xlab	x-axis label
ylab	y-axis label
zlab	z value label
transpose	Logical: If TRUE, transpose matrix

Author(s)

E.D. Gennatas

---

precision *Precision (aka PPV)*

Description

The first factor level is considered the positive case.

Usage

```
precision(true, estimated, harmonize = FALSE, verbose = TRUE)
```

Arguments

true	Factor: True labels
estimated	Factor: Estimated labels
harmonize	Logical: If TRUE, run factorHarmonize first
verbose	Logical: If TRUE, print messages to output. Default = TRUE

---

predict.addtree *Predict Method for MediBoost Model*

Description

Obtains predictions from a trained MediBoost model

Usage

```
## S3 method for class 'addtree'
predict(object, newdata, verbose = FALSE, ...)
```

Arguments

object	A trained model of class "addtree"
newdata	Optional: a matrix / data.frame of features with which to predict
verbose	Logical: If TRUE, print messages to output
...	Not used

Author(s)

E.D. Gennatas

---

predict.boost *Predict method for* boost *object*

Description

Predict method for boost object

Usage

```
## S3 method for class 'boost'
predict(
  object,
  newdata = NULL,
  n.feat = NCOL(newdata),
  n.iter = NULL,
  as.matrix = FALSE,
  verbose = FALSE,
  n.cores = rtCores,
  ...
)
```

Author(s)

E.D. Gennatas

---

predict.cartLinBoostTV *Predict method for* cartLinBoostTV *object*

Description

Predict method for cartLinBoostTV object

Usage

```
## S3 method for class 'cartLinBoostTV'
predict(
  object,
  newdata = NULL,
  n.feat = NCOL(newdata),
  n.iter = NULL,
  as.matrix = FALSE,
  verbose = FALSE,
  n.cores = rtCores,
  ...
)
```

Arguments

object	cartLinBoostTV object
newdata	Set of predictors
n.feat	Integer: N of features to use. Default = NCOL(newdata)
n.iter	Integer: N of iterations to predict from. Default = (all available)
as.matrix	Logical: If TRUE, return predictions from each iterations. Default = FALSE
verbose	Logical: If TRUE, print messages to console. Default = FALSE
n.cores	Integer: Number of cores to use. Default = rtCores

Author(s)

E.D. Gennatas

---

predict.cartLite *Predict method for* cartLite *object*

Description

Predict method for cartLite object

Usage

```
## S3 method for class 'cartLite'
predict(object, newdata, verbose = FALSE, ...)
```

Arguments

object	cartLite object
newdata	Data frame of predictors
verbose	Logical: If TRUE, print messages to console. Default = FALSE

Author(s)

E.D. Gennatas

predict.cartLiteBoostTV *Predict method for* cartLiteBoostTV *object*

Description

Predict method for cartLiteBoostTV object

Usage

```
## S3 method for class 'cartLiteBoostTV'
predict(
  object,
  newdata = NULL,
  n.feat = NCOL(newdata),
  n.iter = NULL,
  as.matrix = FALSE,
  verbose = FALSE,
  n.cores = rtCores,
  ...
)
```

Arguments

object	cartLiteBoostTV object
newdata	Set of predictors
n.feat	Integer: N of features to use. Default = NCOL(newdata)
n.iter	Integer: N of iterations to predict from. Default = (all available)
as.matrix	Logical: If TRUE, return predictions from each iterations. Default = FALSE
verbose	Logical: If TRUE, print messages to console. Default = FALSE
n.cores	Integer: Number of cores to use. Default = rtCores

Author(s)

E.D. Gennatas

predict.gamselx2 *Predict Method for gamselx2 Fits*

Description

Obtains predictions from a fitted gamselx2 model object

Usage

```
## S3 method for class 'gamselx2'
predict(object, newdata = NULL, ...)
```

Author(s)

E.D. Gennatas

`predict.glmLite` *Predict method for* `glmLite` *object*

Description

Predict method for `glmLite` object

Usage

```
## S3 method for class 'glmLite'
predict(object, newdata, verbose = FALSE, ...)
```

Arguments

`object`	glmLite object
`newdata`	Data frame of predictors
`verbose`	Logical: If TRUE, print messages to console. Default = FALSE

Author(s)

E.D. Gennatas

`predict.glmLiteBoostTV` *Predict method for* `glmLiteBoostTV` *object*

Description

Predict method for `glmLiteBoostTV` object

Usage

```
## S3 method for class 'glmLiteBoostTV'
predict(
  object,
  newdata = NULL,
  n.feat = NCOL(newdata),
  n.iter = NULL,
  as.matrix = FALSE,
  verbose = FALSE,
  n.cores = rtCores,
  ...
)
```

Arguments

object	glmLiteBoostTV object
newdata	Set of predictors
n.feat	Integer: N of features to use. Default = NCOL(newdata)
n.iter	Integer: N of iterations to predict from. Default = (all available)
as.matrix	Logical: If TRUE, return predictions from each iteration. Default = FALSE
verbose	Logical: If TRUE, print messages to console. Default = FALSE
n.cores	Integer: Number of cores to use. Default = rtCores

Author(s)

E.D. Gennatas

---

predict.hytboost *Predict method for* hytboost *object*

Description

Predict method for hytboost object

Usage

```
## S3 method for class 'hytboost'
predict(
  object,
  newdata = NULL,
  n.iter = NULL,
  fixed.cxr = NULL,
  as.matrix = FALSE,
  n.cores = 1,
  verbose = FALSE,
  ...
)
```

Arguments

object	hytboost object
newdata	data.frame of predictors
n.iter	Integer: Use the first so many trees for prediction
fixed.cxr	[Internal use] Matrix: Cases by rules to use instead of matching cases to rules using newdata
as.matrix	Logical: If TRUE, output
n.cores	Integer: Number of cores to use
verbose	Logical: If TRUE, print messages to console
...	Not used

Author(s)

E.D. Gennatas

predict.hytboostnow *Predict method for* hytboostnow *object*

Description

Predict method for hytboostnow object

Usage

```
## S3 method for class 'hytboostnow'
predict(
  object,
  newdata = NULL,
  n.feat = NCOL(newdata),
  n.iter = NULL,
  fixed.cxr = NULL,
  as.matrix = FALSE,
  n.cores = 1,
  verbose = FALSE,
  ...
)
```

Arguments

object	hytboostnow object

Author(s)

E.D. Gennatas

predict.hytreenow *Predict method for* hytreeLite *object*

Description

Predict method for hytreeLite object

Usage

```
## S3 method for class 'hytreenow'
predict(
  object,
  newdata,
  n.feat = NCOL(newdata),
  fixed.cxr = NULL,
  cxr.newdata = NULL,
  cxr = FALSE,
  cxrcoef = FALSE,
  verbose = FALSE,
  trace = 0,
  ...
)
```

Arguments

object	hytreenow
newdata	Data frame of predictors
n.feat	[Internal use] Integer: Use first n.feat columns of newdata to predict. Defaults to all
fixed.cxr	[Internal use] Matrix: Cases by rules to use instead of matching cases to rules using newdata
cxr.newdata	[Internal use] Data frame: Use these values to match cases by rules
cxr	Logical: If TRUE, return list which includes cases-by-rules matrix along with predicted values
cxrcoef	Logical: If TRUE, return cases-by-rules * coefficients matrix along with predicted values
verbose	Logical: If TRUE, print messages to console
trace	Not used

Author(s)

E.D. Gennatas

---

predict.hytreew *Predict method for* hytreew *object*

Description

Predict method for hytreew object

Usage

```
## S3 method for class 'hytreew'
predict(
  object,
  newdata,
  n.feat = NCOL(newdata),
  fixed.cxr = NULL,
  cxr.newdata = NULL,
  cxr = FALSE,
  cxrcoef = FALSE,
  verbose = FALSE,
  trace = 0,
  ...
)
```

Arguments

object	hytreew
newdata	Data frame of predictors
n.feat	[Internal use] Integer: Use first n.feat columns of newdata to predict. Defaults to all

fixed.cxr	[Internal use] Matrix: Cases by rules to use instead of matching cases to rules using newdata
cxr.newdata	[Internal use] Data frame: Use these values to match cases by rules
cxr	Logical: If TRUE, return list which includes cases-by-rules matrix along with predicted values
cxrcoef	Logical: If TRUE, return cases-by-rules * coefficients matrix along with predicted values
verbose	Logical: If TRUE, print messages to console
trace	Not used

Author(s)

E.D. Gennatas

---

predict.lihad *Predict method for* lihad *object*

Description

Predict method for lihad object

Usage

```
## S3 method for class 'lihad'
predict(
  object,
  newdata = NULL,
  learning.rate = NULL,
  n.feat = NULL,
  verbose = FALSE,
  cxrcoef = FALSE,
  ...
)
```

Arguments

object	an rtMod trained with s_LIHAD or an lihad object
newdata	data frame of predictor features
learning.rate	Float: learning rate if object was lihad
n.feat	Integer: internal use only
verbose	Logical: If TRUE, print messages to console. Default = FALSE
cxrcoef	Logical: If TRUE, return matrix of cases by coefficients along with predictions. Default = FALSE

Author(s)

E.D. Gennatas

predict.nlareg *Predict method for* nlareg *object*

Description

Predict method for nlareg object

Usage

```
## S3 method for class 'nlareg'
predict(object, newdata, ...)
```

Arguments

object	nlareg object
newdata	Data frame of predictors
...	Unused

Author(s)

E.D. Gennatas

predict.nullmod **rtemis** *internal: predict for an object of class* nullmod

Description

rtemis internal: predict for an object of class nullmod

Usage

```
## S3 method for class 'nullmod'
predict(object, newdata = NULL, ...)
```

Arguments

object	Object of class nullmod
newdata	Not used
...	Not used

predict.rtBSplines *Predict S3 method for* rtBSplines

Description

Predict S3 method for rtBSplines

Usage

```
## S3 method for class 'rtBSplines'
predict(object, newdata = NULL, ...)
```

Author(s)

E.D. Gennatas

predict.rtTLS *predict.rtTLS: predict method for* rtTLS *object*

Description

predict.rtTLS: predict method for rtTLS object

print.rtTLS: print method for rtTLS object

Usage

```
## S3 method for class 'rtTLS'
predict(object, newdata, ...)

## S3 method for class 'rtTLS'
print(x, ...)
```

predict.rulefit *predict method for* rulefit *object*

Description

predict method for rulefit object

Usage

```
## S3 method for class 'rulefit'
predict(object, newdata = NULL, verbose = TRUE, ...)
```

Arguments

object	rulefit object
newdata	Feature matrix / data.frame: will be converted to data.table
verbose	Logical: If TRUE, print messages during execution. Default = TRUE
...	Ignored

Value

Vector of estimated values

---

predict.shyoptleaves *Predict method for* shyoptleaves *object*

Description

Predict method for shyoptleaves object

Usage

```
## S3 method for class 'shyoptleaves'
predict(
  object,
  newdata,
  type = c("response", "probability", "all", "step"),
  n.leaves = NULL,
  n.feat = NCOL(newdata),
  fixed.cxr = NULL,
  cxr.newdata = NULL,
  cxr = FALSE,
  cxrcoef = FALSE,
  verbose = FALSE,
  trace = 0,
  ...
)
```

Arguments

object	shytreeRaw
newdata	Data frame of predictors
n.feat	[Internal use] Integer: Use first n.feat columns of newdata to predict. Defaults to all
fixed.cxr	[Internal use] Matrix: Cases by rules to use instead of matching cases to rules using newdata
cxr.newdata	[Internal use] Data frame: Use these values to match cases by rules
cxr	Logical: If TRUE, return list which includes cases-by-rules matrix along with predicted values
cxrcoef	Logical: If TRUE, return cases-by-rules * coefficients matrix along with predicted values
verbose	Logical: If TRUE, print messages to console
trace	Not used

Author(s)

E.D. Gennatas

`predict.shytreegamleaves`
Predict method for `shytreegamleaves` *object*

Description

Predict method for `shytreegamleaves` object

Usage

```
## S3 method for class 'shytreegamleaves'
predict(
  object,
  newdata,
  type = c("response", "probability", "all", "step"),
  n.leaves = NULL,
  fixed.cxr = NULL,
  cxr.newdata = NULL,
  cxr = FALSE,
  cxrcoef = FALSE,
  verbose = FALSE,
  ...
)
```

Arguments

`object`	`shytreeRaw`
`newdata`	Data frame of predictors
`type`	Character: "response", "probability", "all", "step"
`n.leaves`	Integer: Use the first `n.leaves` of the tree for prediction
`fixed.cxr`	[Internal use] Matrix: Cases by rules to use instead of matching cases to rules using `newdata`
`cxr.newdata`	[Internal use] Data frame: Use these values to match cases by rules
`cxr`	Logical: If TRUE, return list which includes cases-by-rules matrix along with predicted values
`cxrcoef`	Logical: If TRUE, return cases-by-rules * coefficients matrix along with predicted values
`verbose`	Logical: If TRUE, print messages to console
`...`	Not used

Author(s)

E.D. Gennatas

predict.shytreeLeavesRC *Predict method for* shytreeLeavesRC *object*

Description

Predict method for shytreeLeavesRC object

Usage

```
## S3 method for class 'shytreeLeavesRC'
predict(
  object,
  newdata,
  type = c("response", "probability", "all", "step"),
  n.leaves = NULL,
  n.feat = NCOL(newdata),
  fixed.cxr = NULL,
  cxr.newdata = NULL,
  cxr = FALSE,
  cxrcoef = FALSE,
  verbose = FALSE,
  ...
)
```

Arguments

object	shytreeRaw
newdata	Data frame of predictors
n.feat	[Internal use] Integer: Use first n.feat columns of newdata to predict. Defaults to all
fixed.cxr	[Internal use] Matrix: Cases by rules to use instead of matching cases to rules using newdata
cxr.newdata	[Internal use] Data frame: Use these values to match cases by rules
cxr	Logical: If TRUE, return list which includes cases-by-rules matrix along with predicted values
cxrcoef	Logical: If TRUE, return cases-by-rules * coefficients matrix along with predicted values
verbose	Logical: If TRUE, print messages to console
trace	Integer: 0, 1, or 2. Provides increasing amount to information printed to the console

Author(s)

E.D. Gennatas

preorderTree.addtree rtemis-internal *Traverse ADDTREE tree by preorder*

Description

Recursively Traverses ADDTREE tree by preorder function and builds data frame representation

Usage

```
preorderTree.addtree(rt, x, verbose = FALSE)
```

Arguments

rt	rt Object from s_ADDTREE
x	Features
verbose	Logical: If TRUE, print messages to stdout

Author(s)

E.D. Gennatas

preprocess *Data preprocessing*

Description

Prepare data for analysis and visualization

Usage

```
preprocess(
  x,
  y = NULL,
  completeCases = FALSE,
  removeCases.thres = NULL,
  removeFeatures.thres = NULL,
  missingness = FALSE,
  impute = FALSE,
  impute.type = c("missRanger", "micePMM", "meanMode"),
  impute.missRanger.params = list(pmm.k = 3, maxiter = 10, num.trees = 500),
  impute.discrete = getMode,
  impute.numeric = mean,
  integer2factor = FALSE,
  integer2numeric = FALSE,
  logical2factor = FALSE,
  logical2numeric = FALSE,
  numeric2factor = FALSE,
  numeric2factor.levels = NULL,
  len2factor = 0,
```

```
  character2factor = FALSE,
  factorNA2missing = FALSE,
  factorNA2missing.level = "missing",
  nonzeroFactors = FALSE,
  scale = FALSE,
  center = scale,
  removeConstants = TRUE,
  removeDuplicates = FALSE,
  oneHot = FALSE,
  exclude = NULL,
  verbose = TRUE,
  parallel.type = ifelse(.Platform$OS.type == "unix", "fork", "psock")
)
```

Arguments

x	data.frame to be preprocessed
completeCases	Logical: If TRUE, only retain complete cases (no missing data). Default = FALSE
removeCases.thres	Float (0, 1): Remove cases with >= to this fraction of missing features.
removeFeatures.thres	Float (0, 1): Remove features with missing values in >= to this fraction of cases.
missingness	Logical: If TRUE, generate new boolean columns for each feature with missing values, indicating which cases were missing data.
impute	Logical: If TRUE, impute missing cases. See `impute.discrete` and `impute.numeric` for how
impute.type	Character: How to impute data: "missRanger" and "missForest" use the packages of the same name to impute by iterative random forest regression. "rfImpute" uses `randomForest::rfImpute` (see its documentation), "meanMode" will use mean and mode by default or any custom function defined in `impute.discrete` and `impute.numeric`. Default = "missRanger" (which is much faster than "missForest"). "missForest" is included for compatibility with older pipelines.
impute.missRanger.params	Named list with elements "pmm.k" and "maxiter", which are passed to `missRanger::missRanger`. `pmm.k` greater than 0 results in predictive mean matching. Default `pmm.k = 3` `maxiter = 10` `num.trees = 500`. Reduce `num.trees` for faster imputation especially in large datasets. Set `pmm.k = 0` to disable predictive mean matching to `missForest::missForest`
impute.discrete	Function that returns single value: How to impute discrete variables for `impute.type = "meanMode"`. Default = getMode
impute.numeric	Function that returns single value: How to impute continuous variables for `impute.type = "meanMode"`. Default = `mean`
integer2factor	Logical: If TRUE, convert all integers to factors
integer2numeric	Logical: If TRUE, convert all integers to numeric (will only work if `integer2factor = FALSE`)
logical2factor	Logical: If TRUE, convert all logical variables to factors

`logical2numeric`
: Logical: If TRUE, convert all logical variables to numeric

`numeric2factor`
: Logical: If TRUE, convert all numeric variables to factors

`len2factor`
: Integer (>=2): Convert all numeric variables with less than or equal to this number of unique values to factors. Default = NULL. For example, if binary variables are encoded with 1, 2, you could use `len2factor = 2` to convert them to factors. If race is encoded with 6 integers, you can use 6.

`character2factor`
: Logical: If TRUE, convert all character variables to factors

`factorNA2missing`
: Logical: If TRUE, make NA values in factors be of level `factorNA2missing.level`. In many cases this is the preferred way to handle missing data in categorical variables. Note that since this step is performed before imputation, you can use this option to handle missing data in categorical variables and impute numeric variables in the same `preprocess` call.

`factorNA2missing.level`
: Character: Name of level if `factorNA2missing = TRUE`. Default = "missing"

`nonzeroFactors`
: Logical: Shift factor values to exclude zeros.

`scale`
: Logical: If TRUE, scale columns of `x`

`center`
: Logical: If TRUE, center columns of `x`. Note that by default it is the same as `scale`

`removeConstants`
: Logical: If TRUE, remove constant columns.

`removeDuplicates`
: Logical: If TRUE, remove duplicated cases.

`oneHot`
: Logical: If TRUE, convert all factors using one-hot encoding

`exclude`
: Integer, vector: Exclude these columns from preprocessing.

`verbose`
: Logical: If TRUE, write messages to console.

Details

By default, removes constant features and duplicated cases (removeConstants = TRUE, removeDuplicates = TRUE), everything else must be specified.

Order of operations (reflected by order of arguments in usage):

- keep complete cases only
- remove duplicates
- remove cases by missingness threshold
- remove features by missingness threshold
- integer to factor
- integer to numeric
- logical to factor
- logical to numeric
- numeric to factor
- numeric with less than N unique values to factor
- character to factor

- factor NA to named level
- add missingness column
- impute
- scale and/or center
- remove constants
- one-hot encoding

Author(s)

E.D. Gennatas

`preprocess_` *Data preprocessing (in-place)*

Description

Prepare data for analysis and visualization

Usage

```
preprocess_(
  x,
  y = NULL,
  removeFeatures.thres = NULL,
  missingness = FALSE,
  integer2factor = FALSE,
  integer2numeric = FALSE,
  logical2factor = FALSE,
  logical2numeric = FALSE,
  numeric2factor = FALSE,
  numeric2factor.levels = NULL,
  len2factor = 0,
  character2factor = FALSE,
  factorNA2missing = FALSE,
  factorNA2missing.level = "missing",
  scale = FALSE,
  center = scale,
  removeConstants = FALSE,
  oneHot = FALSE,
  exclude = NULL,
  verbose = TRUE
)
```

Arguments

`x`	data.frame or data.table to be preprocessed. If data.frame, will be converted to data.table in-place of missing features.
`removeFeatures.thres`	Float (0, 1): Remove features with missing values in >= to this fraction of cases.

`missingness` Logical: If TRUE, generate new boolean columns for each feature with missing values, indicating which cases were missing data.

`integer2factor` Logical: If TRUE, convert all integers to factors

`integer2numeric`
Logical: If TRUE, convert all integers to numeric (will only work if `integer2factor = FALSE`)

`logical2factor` Logical: If TRUE, convert all logical variables to factors

`logical2numeric`
Logical: If TRUE, convert all logical variables to numeric

`numeric2factor` Logical: If TRUE, convert all numeric variables to factors

`len2factor` Integer (>=2): Convert all numeric variables with less than or equal to this number of unique values to factors. For example, if binary variables are encoded with 1, 2, you could use `len2factor = 2` to convert them to factors. If race is encoded with 6 integers, you can use 6.

`character2factor`
Logical: If TRUE, convert all character variables to factors

`factorNA2missing`
Logical: If TRUE, make NA values in factors be of level `factorNA2missing.level`. In many cases this is the preferred way to handle missing data in categorical variables. Note that since this step is performed before imputation, you can use this option to handle missing data in categorical variables and impute numeric variables in the same `preprocess` call.

`factorNA2missing.level`
Character: Name of level if `factorNA2missing = TRUE`.

`scale` Logical: If TRUE, scale columns of `x`

`center` Logical: If TRUE, center columns of `x`

`removeConstants`
Logical: If TRUE, remove constant columns.

`oneHot` Logical: If TRUE, convert all factors using one-hot encoding

`exclude` Integer, vector: Exclude these columns from preprocessing.

`verbose` Logical: If TRUE, write messages to console.

Details

This function (ending in "_") performs operations **in-place** and returns the preprocessed data.table silently (e.g. for piping). Note that imputation is not currently supported - use preprocess for imputation.

Order of operations is the same as the order of arguments in usage:

- keep complete cases only
- remove duplicates
- remove cases by missingness threshold
- remove features by missingness threshold
- integer to factor
- integer to numeric
- logical to factor
- logical to numeric

- numeric to factor
- numeric with less than N unique values to factor
- character to factor
- factor NA to named level
- add missingness column
- scale and/or center
- remove constants
- one-hot encoding

Author(s)

E.D. Gennatas

Examples

```
## Not run:
x <- data.table(a = sample(c(1:3), 30, T),
b = rnorm(30, 12),
c = rnorm(30, 200),
d = sample(c(21:22), 30, T),
e = rnorm(30, -100),
f = rnorm(30, 950),
g = rnorm(30),
h = rnorm(30))
## add duplicates
x <- rbind(x, x[c(1, 3), ])
## add constant
x[, z := 99]
preprocess_(x)

## End(Not run)
```

---

previewcolor *Preview color v2.0*

Description

Preview one or multiple colors using little rhombi with their little labels up top

Usage

```
previewcolor(
  x,
  main = NULL,
  bg = "#333333",
  main.col = "#b3b3b3",
  main.x = 0.7,
  main.y = 0.2,
  main.adj = 0,
  main.cex = 0.9,
  main.font = 1,
```

```
  width = NULL,
  xlim = NULL,
  ylim = c(0, 2.2),
  asp = 1,
  labels.y = 1.55,
  label.cex = NULL,
  mar = c(0, 0, 0, 1),
  par.reset = TRUE,
  filename = NULL,
  pdf.width = 8,
  pdf.height = 2.5
)
```

Arguments

x	Color, vector: One or more colors that R understands
main	Character: Title. Default = NULL, which results in deparse(substitute(x))
bg	Background color. Default = "#333333" (dark gray)
main.x	Float: x coordinate for main. Default = .75
main.y	Float: y coordinate for main. Default = 0
main.adj	Float: adj argument to mtext for main.
main.cex	Float: character expansion factor for main. Default = .9
main.font	Integer, 1 or 2: Weight of main 1: regular, 2: bold. Default = 2
width	Float: Plot width. Default = NULL, i.e. set automatically
xlim	Vector, length 2: x-axis limits. Default = NULL, i.e. set automatically
ylim	Vector, length 2: y-axis limits. Default = c(0.7, 2)
asp	Float: Plot aspect ratio. Default = 1
labels.y	Float: y coord for labels. Default = 1.55 (rhombi are fixed and range y .5 - 1.5)
label.cex	Float: Character expansion for labels. Default = NULL, and is calculated automatically based on
par.rest	Logical: If TRUE, reset par setting on exit. Default = TRUE length of x in order to look reasonable in your RStudio plot panel.

Value

Nothing, prints plot

Examples

```
colors <- colorgradient.x(seq(-5, 5))
previewcolor(colors)
```

print.addtree *Print method for* addtree *object created using s_ADDTREE*

Description

Print method for addtree object created using s_ADDTREE

Usage

```
## S3 method for class 'addtree'
print(x, ...)
```

Arguments

x	rtMod object created using s_ADDTREE

Author(s)

E.D. Gennatas

print.boost *Print method for boost object*

Description

Print method for boost object

Usage

```
## S3 method for class 'boost'
print(x, ...)
```

Author(s)

E.D. Gennatas

print.cartLinBoostTV *Print method for boost object*

Description

Print method for boost object

Usage

```
## S3 method for class 'boost'
print(x, ...)
```

Author(s)

E.D. Gennatas

print.cartLiteBoostTV *Print method for boost object*

Description

Print method for boost object

Usage

```
## S3 method for class 'boost'
print(x, ...)
```

Author(s)

E.D. Gennatas

print.classError *Print classError*

Description

Print classError

Usage

```
## S3 method for class 'classError'
print(x, decimal.places = 4, ...)
```

Arguments

x	Object of type classError
...	Not used

Author(s)

E.D. Gennatas

print.glmLiteBoostTV *Print method for boost object*

Description

Print method for boost object

Usage

```
## S3 method for class 'boost'
print(x, ...)
```

Author(s)

E.D. Gennatas

print.gridSearch — print *method for* gridSearch *object*

Description

print method for gridSearch object

Usage

```
## S3 method for class 'gridSearch'
print(x, ...)
```

Author(s)

E.D. Gennatas

print.hytboost — *Print method for boost object*

Description

Print method for boost object

Usage

```
## S3 method for class 'hytboost'
print(x, ...)
```

Author(s)

E.D. Gennatas

print.hytboostnow — *Print method for boost object*

Description

Print method for boost object

Usage

```
## S3 method for class 'hytboostnow'
print(x, ...)
```

Author(s)

E.D. Gennatas

print.lihad *Print method for* lihad *object*

Description

Print method for lihad object

Usage

```
## S3 method for class 'lihad'
print(x, ...)
```

Author(s)

E.D. Gennatas

print.massGLM *printmassGLM object*

Description

printmassGLM object

Usage

```
## S3 method for class 'massGLM'
print(x, ...)
```

Arguments

x	massGLM object

Author(s)

E.D. Gennatas

print.regError *Print* regError *object*

Description

print regError object

Usage

```
## S3 method for class 'regError'
print(x, ...)
```

Arguments

x	regError object

Author(s)

E.D. Gennatas

print.resample print *method for resample object*

Description

Print resample information

Usage

```
## S3 method for class 'resample'
print(x, ...)
```

Arguments

x	resample object

Author(s)

E.D. Gennatas

print.shyoptleaves *Print method for* shyoptleaves *object*

Description

Print method for shyoptleaves object

Usage

```
## S3 method for class 'shyoptleaves'
print(x, ...)
```

Arguments

x	shyoptleaves object

Author(s)

E.D. Gennatas

print.shytreegamleaves *Print method for* shytreegamleaves *object*

Description

Print method for shytreegamleaves object

Usage

```
## S3 method for class 'shytreegamleaves'
print(x, ...)
```

Arguments

x	shytreegamleaves object
...	Not used

Author(s)

E.D. Gennatas

print.shytreeLeavesRC *Print method for* shytreeLeavesRC *object*

Description

Print method for shytreeLeavesRC object

Usage

```
## S3 method for class 'shytreeLeavesRC'
print(x, ...)
```

Arguments

x	shytreeLeavesRC object

Author(s)

E.D. Gennatas

print.survError *Print survError*

Description

Print survError

Usage

```
## S3 method for class 'survError'
print(x, decimal.places = 4, ...)
```

Arguments

x	Object of type survError
...	Not used
decimal.place	Integer: Number of decimal places to print. Default = 4

Author(s)

E.D. Gennatas

`printdf` *Print data frame*

Description

Pretty print a data frame

Usage

```
printdf(
  x,
  pad = 0,
  spacing = 1,
  ddSci.dp = NULL,
  transpose = FALSE,
  justify = "right",
  colnames = TRUE,
  rownames = TRUE,
  column.col = rtHighlight$bold,
  row.col = silver,
  newline.pre = FALSE,
  newline = FALSE
)
```

Arguments

`x`	data frame
`pad`	Integer: Pad output with this many spaces. Default = 2
`spacing`	Integer: Number of spaces between columns. Default = 1
`ddSci.dp`	Integer: Number of decimal places to print using ddSci. Default = NULL for no formatting
`transpose`	Logical: If TRUE, transpose `x` before printing. Default = FALSE
`justify`	Character: "right", "left". Default = "right"
`colnames`	Logical: If TRUE, print column names. Default = TRUE
`rownames`	Logical: If TRUE, print row names. Default = TRUE
`column.col`	`crayon` color for printing column names. Default = **rtemis** default highlight
`row.col`	`crayon` color for printing row names. Default = silver
`newline.pre`	Logical: If TRUE, print a new line before printing data frame. Default = FALSE
`newlin`	Logical: If TRUE, print a new line after printing data frame. Default = FALSE

Details

By design, numbers will not be justified, but using ddSci.dp will convert to characters, which will be justified. This is intentional for internal use.

Author(s)

E.D. Gennatas

printdf1 *Print 1 x N data frame*

Description

Pretty print a data frame containing 1 row of data with named columns as a vertical list of " name : value" " other.name : other.value"

Usage

```
printdf1(x, pad = 2)
```

Arguments

x	data frame
pad	Integer: Pad output with this many spaces. Default = 2

Author(s)

E.D. Gennatas

printls *Pretty print list*

Description

Pretty print a list (or data frame) recursively

Usage

```
printls(
  x,
  prefix = "",
  pad = 3,
  center.title = TRUE,
  title = NULL,
  title.newline = FALSE,
  newline.pre = FALSE,
  color = NULL
)
```

Arguments

x	list or object that will be converted to a list
prefix	Character: Optional prefix for names
pad	Integer: Pad output with this many spaces. Default = 2
center.title	Logical: If TRUE, autopad title for centering, if present. Default TRUE
color	crayon color to be applied when printing values. Default = NULL (do not use crayon)

Details

Data frames in R began life as lists

Author(s)

E.D. Gennatas

`prune.addtree` *Prune ADDTREE tree*

Description

Prune an ADDTREE tree in Node format using `data.tree` to remove sister nodes with same class estimate.

Usage

```
prune.addtree(
  addtree,
  prune.empty.leaves = TRUE,
  remove.bad.parents = TRUE,
  verbose = TRUE
)
```

Arguments

`addtree`	rtMod trained with s_ADDTREE
`prune.empty.leaves`	Logical: If TRUE, remove leaves with 0 cases.
`remove.bad.parents`	Logical: If TRUE, remove nodes with no siblings but children and give their children to their parent.

Author(s)

E.D. Gennatas

`prune.rpart.rt` `prune.rpart` *experimental replacement*

Description

`prune.rpart` experimental replacement

Usage

```
prune.rpart.rt(tree, cp, ...)
```

Author(s)

E.D. Gennatas

psd *Population Standard Deviation*

Description

Estimate the population standard deviation:

$$sqrt(mean(x^2) - mean(x)^2)$$

Usage

```
psd(x)
```

Arguments

x	Numeric vector

Details

This will be particularly useful when the machines finally collect data on all humans. Caution is advised, however, as you never know how many may be hiding underground.

Value

Population standard deviation

Author(s)

E.D. Gennatas

qstat *SGE qstat*

Description

Run SGE qstat

Usage

```
qstat()
```

Details

alias for `system("qstat")`

readseglabels *Read nifti segmentation label file*

Description

Read nifti label file, a space/tab separated, headerless text file containing the following fields: LabelID LabelName R G B A

Usage

```
readseglabels(x)
```

Arguments

x	Character: Path to nifti label file

Author(s)

E.D. Gennatas

reduceList *Reduce List to single value per element*

Description

Reduce List to single value per element

Usage

```
reduceList(x, char.max = 50, return.df = TRUE)
```

relu *ReLU - Rectified Linear Unit*

Description

ReLU - Rectified Linear Unit

Usage

```
relu(x)
```

Arguments

x	Numeric: Input

`resample` *Resampling methods*

Description

Create resamples of your data, e.g. for model building or validation. "bootstrap" gives the standard bootstrap, i.e. random sampling with replacement, using bootstrap, "strat.sub" creates stratified subsamples using strat.sub, while "strat.boot" uses strat.boot which runs strat.sub and then randomly duplicates some of the training cases to reach original length of input (default) or length defined by `target.length`.

Usage

```
resample(
  y,
  n.resamples = 10,
  resampler = c("strat.sub", "strat.boot", "kfold", "bootstrap", "loocv"),
  index = NULL,
  group = NULL,
  stratify.var = y,
  train.p = 0.75,
  strat.n.bins = 4,
  target.length = NROW(y),
  rtset = NULL,
  seed = NULL,
  verbose = TRUE
)
```

Arguments

`y`	Numeric vector. Usually the outcome; `length(y)` defines sample size
`n.resamples`	Integer: Number of training/testing sets required
`resampler`	Character: Type of resampling to perform: "bootstrap", "kfold", "strat.boot", "strat.sub". Default = "strat.boot" for `length(y)` < `200`, otherwise "strat.sub"
`index`	List where each element is a vector of training set indices. Use this for manual or precalculated train/test splits
`group`	Integer, vector, length = `length(y)`: Integer vector, where numbers define fold membership. e.g. for 10-fold on a dataset with 1000 cases, you could use `group = rep(1:10, each = 100)`
`stratify.var`	Numeric vector (optional): Variable used for stratification. Defaults to `y`
`train.p`	Float (0, 1): Fraction of cases to assign to traininig set for `resampler = "strat.sub"`
`strat.n.bins`	Integer: Number of groups to use for stratification for `resampler = "strat.sub" / "strat.boot"`
`target.length`	Integer: Number of cases for training set for `resampler = "strat.boot"`. Default = `length(y)`
`rtset`	List: Output of an rtset.resample (or named list with same structure). NOTE: Overrides all other arguments. Default = NULL
`seed`	Integer: (Optional) Set seed for random number generator, in order to make output reproducible. See `?base::set.seed`
`verbose`	Logical: If TRUE, print messages to screen

Details

resample is used by multiple **rtemis** learners, gridSearchLearn, and elevate. Note that option 'kfold', which uses kfold results in resamples of slightly different length for y of small length, so avoid all operations which rely on equal-length vectors. For example, you can't place resamples in a data.frame, but must use a list instead.

Author(s)

E.D. Gennatas

See Also

elevate

Examples

```
y <- rnorm(200)
# 10-fold (stratified)
res <- resample(y, 10, "kfold")
# 25 stratified subsamples
res <- resample(y, 25, "strat.sub")
# 100 stratified bootstraps
res <- resample(y, 100, "strat.boot")
```

---

resLearn_future **rtemis** *internal: Resample Learn*

Description

Train an **rtemis** learner on a set of resamples

Usage

```
resLearn_future(
  x,
  y,
  mod,
  resample.rtset = rtset.cv.resample(),
  weights = NULL,
  params = list(),
  mtry = NULL,
  .preprocess = NULL,
  verbose = TRUE,
  res.verbose = FALSE,
  trace = 0,
  save.mods = TRUE,
  outdir = NULL,
  n.workers = 1,
  parallel.type = "nobodycares"
)
```

Arguments

x	features - training set
y	outcome - training set
mod	Character: **rtemis** model. See modSelect gives available models
resample.rtset	List: output of rtset (or a list of same structure)
params	List of named elements, each is a single value
verbose	Logical: If TRUE, print messages to screen
res.verbose	Logical: Will be passed to each mod's verbose argument
save.mods	Logical: If TRUE, save all models, otherwise discard after training. Use with elevate when training a large number of resamples. Default = TRUE
outdir	Character: Path to save output. Default = NULL
n.workers	Integer: Number of cores to use.

Details

Input: features (x) and outcome (y) Procedure: resample, train learners Output: trained learners This is used internally by elevate and for bagging, when the bag.resampler argument is set in a learner.

Author(s)

E.D. Gennatas

---

resLearn_pbapply **rtemis** *internal: Resample Learn*

Description

Train an **rtemis** learner on a set of resamples

Usage

```
resLearn_pbapply(
  x,
  y,
  mod,
  resample.rtset = rtset.cv.resample(),
  weights = NULL,
  params = list(),
  mtry = NULL,
  .preprocess = NULL,
  verbose = TRUE,
  res.verbose = FALSE,
  trace = 0,
  save.mods = TRUE,
  outdir = NULL,
  n.workers = rtCores,
  parallel.type = ifelse(.Platform$OS.type == "unix", "fork", "psock")
)
```

Arguments

x	features - training set
y	outcome - training set
mod	Character: **rtemis** model. See modSelect gives available models
resample.rtset	List: output of rtset (or a list of same structure)
params	List of named elements, each is a single value
verbose	Logical: If TRUE, print messages to screen
res.verbose	Logical: Will be passed to each mod's verbose argument
save.mods	Logical: If TRUE, save all models, otherwise discard after training. Use with elevate when training a large number of resamples. Default = TRUE
outdir	Character: Path to save output. Default = NULL

Details

Input: features (x) and outcome (y) Procedure: resample, train learners Output: trained learners This is used internally by elevate and for bagging, when the bag.resampler argument is set in a learner.

Author(s)

E.D. Gennatas

---

reverseLevels *Reverse factor levels*

Description

Reverse the order of a factor's levels

Usage

```
reverseLevels(x)
```

Arguments

x	Factor

Author(s)

E.D. Gennatas

revfactorlevels *Reverse factor level order*

Description

Reverse factor level order

Usage

```
revfactorlevels(x)
```

Arguments

x	factor

Author(s)

E.D. Gennatas

rfVarSelect *Variable Selection by Random Forest*

Description

Select important variables from a set of features based on RF-estimated variable importance

Usage

```
rfVarSelect(x, y, p = 0.2, print.plot = TRUE, verbose = TRUE)
```

Arguments

x	Predictors
y	outcome
p	Float (0, 1): Fraction of variables in x to select. `p * ncol(x)`. May help to set to a fraction twice what you expect to be the true fraction of useful variables, to reduce false negatives at the expense of false positives which can be dealt by an appropriate learning algorithm.

Author(s)

E.D. Gennatas

rnormmat *Random Normal Matrix*

Description

Create a matrix or data frame of defined dimensions, whose columns are random normal vectors

Usage

```
rnormmat(
  nrow = 10,
  ncol = 10,
  mean = 0,
  sd = 1,
  return.df = FALSE,
  seed = NULL
)
```

Arguments

nrow	Integer: Number of rows. Default = 10
ncol	Integer: Number of columns. Default = 10
mean	Float: Mean. Default = 0
sd	Float: Standard deviation. Default = 1
return.df	Logical: If TRUE, return data.frame, otherwise matrix. Default = TRUE
seed	Integer: Set seed for rnorm. Default = NULL

Author(s)

E.D. Gennatas

roundtohalf *Round to nearest .5*

Description

Round to nearest .5

Usage

```
roundtohalf(x)
```

Arguments

x	numeric vector

Author(s)

E.D. Gennatas

rowMax *Collapse data.frame to vector by getting row max*

Description

Collapse data.frame to vector by getting row max

Usage

```
rowMax(x, na.rm = TRUE)
```

Arguments

x	Input vector
na.rm	Logical. If TRUE, missing values are not considered. Default = TRUE

Author(s)

E.D. Gennatas

rsd *Coefficient of Variation (Relative standard deviation)*

Description

Calculates the coefficient of variation, also known as relative standard deviation, which is given by

$$sd(x)/mean(x)$$

Usage

```
rsd(x, as.percentage = TRUE, na.rm = TRUE, adjust = FALSE, adjust.lo = 1)
```

Arguments

x	Numeric: Input
as.percentage	Logical: If TRUE, multiply by 100
na.rm	Logical: If TRUE, remove missing values before computation
adjust	Logical: If TRUE, if x contains values < adjust.lo, x will be shifted up by adding its minimum
adjust.lo	Float: Threshold to be used if adjust = TRUE

Details

This is not meaningful if mean is close to 0. For such cases, set adjust = TRUE. This will add min(x) to x

Examples

```
## Not run:
mplot3_x(sapply(1:100, function(x) cov(rnorm(100))), 'd', xlab = 'rnorm(100) x 100 times')
# cov of rnorm without adjustment is all over the place
mplot3_x(sapply(1:100, function(x) cov(rnorm(100), adjust = T)), 'd',
xlab = 'rnorm(100) x 100 times')
# COV after shifting above 1 is what you probably want

## End(Not run)
```

---

rsq *R-squared*

Description

R-squared

Usage

```
rsq(x, y)
```

Arguments

x	Float, vector: True values
y	Float, vector: Estimated values

Author(s)

E.D. Gennatas

---

rstudio_theme_rtemis *Apply rtemis theme for RStudio*

Description

Apply the rtemis RStudio theme, an adaptation of the rscodeio theme (https://github.com/anthonynorth/rscodeio)
Recommended to use the Fira Code font with the theme (https://fonts.google.com/specimen/Fira+Code?query=fira+code)

Usage

```
rstudio_theme_rtemis(theme = "dark")
```

`rtClust-class` *R6 class for* **rtemis** *clustering*

Description

R6 class for **rtemis** clustering

R6 class for **rtemis** clustering

Details

rtemis clustering object

Public fields

`clust.name` Character: Name of clustering algorithm

`k` Integer: Number of clusters

`xnames` Column names of x

`clust` Clustering algorithm output

`clusters.train` Cluster assignment for training set

`clusters.test` Cluster assignment for testing set (if supported by algorithm and test set provided)

`parameters` List of clustering algorithm parameters

`extra` List: Algorithm-specific output

Methods

Public methods:

- `rtClust$new()`
- `rtClust$print()`
- `rtClust$clone()`

Method `new()`**:** Create a new rtClust object

Usage:

```
rtClust$new(
  clust.name = character(),
  k = NULL,
  xnames = character(),
  clust = list(),
  clusters.train = numeric(),
  clusters.test = numeric(),
  parameters = list(),
  extra = list()
)
```

Arguments:

`clust.name` Character: Clustering algorithm name

`k` Integer: Number of cluster

`xnames` Character vector: feature names

`clust` Clustering object

`clusters.train` Training set clustering results

`clusters.test` Testing set clustering results

`parameters` List of clustering algorithm parameters

`extra` Optional list of algorithm-specific info

Method `print()`**:** Print method for `rtClust` objects

Usage:

```
rtClust$print()
```

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
rtClust$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

Author(s)

E.D. Gennatas

`rtClust-methods` *rtClust S3 methods*

Description

S3 methods for `rtClust` class.

Usage

```
## S3 method for class 'rtClust'
print(x, ...)
```

`rtDecom-class` *R6 Class for* **rtemis** *Decompositions*

Description

R6 Class for **rtemis** Decompositions

R6 Class for **rtemis** Decompositions

Details

rtemis decomposition R6 object

Public fields

`decom.name` Character: Name of decomposition algorithm

`xnames` Character vector: Column names of x

`decom` Decomposition model output

`parameters` List of decompotision parameters

`center` Numeric vector of column means if centering was applied using `scale()` prior to decomposition

`scale` Numeric vector of column scale factor if scaling was applied using `scale()` prior to decomposition

`projections.train` Input data projected on new axes / basis

`projections.test` Input test data projected on new axes / basis

`extra` List: Algorithm-specific output

Methods

Public methods:

- `rtDecom$new()`
- `rtDecom$print()`
- `rtDecom$clone()`

Method `new()`**:** Initialize `rtDecom` object

Usage:

```
rtDecom$new(
  decom.name = character(),
  xnames = character(),
  decom = list(),
  parameters = list(),
  center = numeric(),
  scale = numeric(),
  projections.train = numeric(),
  projections.test = numeric(),
  extra = list()
)
```

Arguments:

`decom.name` Character: Decomposition algorithm name

`xnames` Character vector: feature names

`decom` Decomposition object

`parameters` list of decomposition algorithm parameters

`center` Numeric vector of column means if centering was applied using `scale()` prior to decomposition

`scale` Numeric vector of column scale factor if scaling was applied using `scale()` prior to decomposition

`projections.train` Training set projections

`projections.test` Testing set projections

`extra` Optional list of algorithm-specific info

Method `print()`**:** Print method for `rtDecom` objects

Usage:
```
rtDecom$print()
```

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:
```
rtDecom$clone(deep = FALSE)
```
Arguments:

`deep` Whether to make a deep clone.

Author(s)

E.D. Gennatas

`rtemis_palette` *Access rtemis palette colors*

Description

Allows you to get 'n' colors of a defined palette, useful for passing to other functions, like ggplot

Usage

```
rtemis_palette(n, palette = rtPalette)
```

Arguments

`n`	Integer: Number of colors to output
`palette`	Character: Palette to use. See available options with `rtpalette()`. Default = `rtPalette`

Author(s)

E.D. Gennatas

Examples

```
rtemis_palette(3)
```

`rtInitProjectDir` *Initialize Project Directory*

Description

Initializes Directory Structure: "R", "Data", "Results"

Usage

```
rtInitProjectDir(verbose = TRUE)
```

Author(s)

E.D. Gennatas

rtlayout *Create multipanel plots with the* mplot3 *family*

Description

Set layout for drawing multiple plots in the same view

Usage

```
rtlayout(
  nrows = NULL,
  ncols = NULL,
  byrow = FALSE,
  autolabel = FALSE,
  pdf.width = NULL,
  pdf.height = NULL,
  filename = NULL
)
```

Arguments

nrows	Integer: N of rows
ncols	Integer: N of columns
byrow	Logical: If TRUE, draw add plots by row Default = FALSE
autolabel	Logical: If TRUE, place letter labels on the top left corner of each figure. Default = FALSE
pdf.width	Float: Width of PDF to save, if filename is provided. Default = ncols * 4.5
pdf.height	Float: Height of PDF to save, if filename is provided. Default = nrows * 4.5
filename	String, optional: Save multiplot to file. Default = NULL

Author(s)

E.D. Gennatas

rtLetters *Construct an n-length vector of letters*

Description

Returns an n-length vector of the latin alphabet, replicating for every 26 characters

Usage

```
rtLetters(n = 100, caps = FALSE)
```

Arguments

n	Length of vector to return
caps	Logical: If TRUE, return all caps

`rtMeta-class` **rtemis** *Meta Model Class*

Description

rtemis Meta Model Class

rtemis Meta Model Class

Details

R6 Class for **rtemis** Meta Models

Super class

`rtemis::rtMod` -> `rtMeta`

Public fields

`grid` Grid of algorithm names and resamples IDs

`base.resample.rtset` List of resampling parameters for base learner training

`base.mod.names` Character vector: names of base learner algorithms

`base.params` Named list of lists with base model parameters

`base.res.y.test` Base resamples' testing set outcomes

`base.res.predicted` Base resamples' predicted valuess

`base.mods` Base learners

`base.mods.error.train` Base learners' training error

`base.mods.error.test` Base learners' testing error

`meta.mod.name` Name of meta algorithm

`meta.params` List of meta model parameters

`meta.mod` Meta model

`sessionInfo` R session info at training time

Methods

Public methods:

- `rtMeta$new()`
- `rtMeta$print()`
- `rtMeta$clone()`

Method `new()`**:** Initialize `rtMeta` object

Usage:

```
rtMeta$new(
  mod.name = character(),
  y.train = numeric(),
  y.test = numeric(),
  x.name = character(),
  y.name = character(),
```

```
  xnames = character(),
  grid = data.frame(),
  base.resample.rtset = list(),
  base.mod.names = character(),
  base.params = list(),
  base.res.y.test = numeric(),
  base.res.predicted = numeric(),
  base.mods = list(),
  base.mods.error.train = list(),
  base.mods.error.test = list(),
  meta.mod.name = character(),
  meta.params = list(),
  meta.mod = list(),
  type = character(),
  fitted = numeric(),
  se.fit = numeric(),
  error.train = list(),
  predicted = numeric(),
  se.prediction = numeric(),
  error.test = list(),
  question = character(),
  extra = list()
)
```

Arguments:

`mod.name` Character: Algorithm name

`y.train` Training set output

`y.test` Testing set output

`x.name` Character: Feature set name

`y.name` Character: Output name

`xnames` Character vector: Feature names

`grid` Data frame with combinations of algorithm names and resample IDs

`base.resample.rtset` List with resampling parameters for base learners

`base.mod.names` Base learner algorithm names

`base.params` Named list of lists with base model parameters

`base.res.y.test` Base learners' predicted values

`base.res.predicted` Base resamples' predicted valuess

`base.mods` Base learners

`base.mods.error.train` Base learners' training error

`base.mods.error.test` Base learners' testing error

`meta.mod.name` Meta model algorithm name

`meta.params` Meta model hyperparameters

`meta.mod` Trained meta model

`type` Character: Type of model (Regression, Classification, Survival)

`fitted` Fitted values (training set predictions)

`se.fit` Standard error of fitted values

`error.train` Training set error

`predicted` Predicted values (Testing set predictions)

`se.prediction` Standard error of predicted values

`error.test` Testing set error

`question` Question the model is trying to answer

`extra` List of extra model info

`base.res.mods` Base learner fitted models

`sessionInfo` R session info at time of training

Method `print()`**:** Plot method for `rtMeta` objects

Usage:

```
rtMeta$print()
```

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
rtMeta$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

`rtMeta-methods` *rtMeta S3 methods*

Description

S3 methods for `rtMeta` class that differ from those of the `rtMod` superclass

Usage

```
## S3 method for class 'rtMeta'
predict(object, newdata, fn = median, ...)
```

Arguments

`object`	`rtMeta` object
`newdata`	Testing set features
`fn`	Function to average predictions

`rtMod-class` **rtemis** *Supervised Model Class*

Description

rtemis Supervised Model Class

rtemis Supervised Model Class

Details

R6 Class for **rtemis** Supervised Models

Note on terminology: The training set ('x.train') is used to build a model and calculate training error (rtMod$error.train). The testing set ('x.test') is not seen during training and only used to create predictions (rtMod$predicted) using the trained model and calculate error metrics (rtMod$error.test). This reflects generalizability of the model and is the error we care the most about. The validation set is used during model tuning. Within rtemis, validation sets are created automatically by resample when appropriate, they are not generally input by the user, with few exceptions, as documented in individual functions.

Public fields

`mod.name` Learner algorithm name

`y.train` Training y data

`y.test` Testing y data

`x.name` Name of x data

`y.name` Name of y data

`xnames` Character vector: Column names of x

`mod` Trained model

`type` Classification, Regression, or Survival

`gridsearch` Grid search output

`parameters` List of hyperparameters used when building model

`fitted` Fitted values

`se.fit` Standard error of the fit

`error.train` Training error

`predicted` Predicted values

`se.prediction` Standard error of the prediction

`error.test` Testing error

`varimp` Variable Importance. Note that this is a general concept and different ML algorithms provide very different approaches

`question` Question the model is hoping to answer

`extra` Algorithm-specific output

`sessionInfo` The output of `sessionInfo()` at the time the model was trained

Methods

Public methods:

- `rtMod$new()`
- `rtMod$print()`
- `rtMod$plot()`
- `rtMod$plotFitted()`
- `rtMod$plotPredicted()`
- `rtMod$plotFittedPredicted()`
- `rtMod$plotVarImp()`
- `rtMod$summary()`

- rtMod$describe()
- rtMod$clone()

Method `new()`**:** Initialize `rtMod` object

Usage:

```
rtMod$new(
  mod.name = character(),
  y.train = numeric(),
  y.test = numeric(),
  x.name = character(),
  y.name = character(),
  xnames = character(),
  mod = list(),
  type = character(),
  gridsearch = list(),
  parameters = list(),
  fitted = numeric(),
  se.fit = NULL,
  error.train = list(),
  predicted = NULL,
  se.prediction = NULL,
  error.test = NULL,
  varimp = NULL,
  question = character(),
  extra = list(),
  sessionInfo = NULL
)
```

Arguments:

`mod.name` Character: Algorithm name

`y.train` Training set output

`y.test` Testing set output

`x.name` Character: Feature set name

`y.name` Character: Output name

`xnames` Character vector: Feature names

`mod` Trained model

`type` Character: Type of model (Regression, Classification, Survival)

`gridsearch` Grid search output

`parameters` List of training parameters

`fitted` Fitted values (training set predictions)

`se.fit` Standard error of fitted values

`error.train` Training set error

`predicted` Predicted values (Testing set predictions)

`se.prediction` Standard error of predicted values

`error.test` Testing set error

`varimp` Variable importance

`question` Question the model is trying to answer

`extra` List of extra model info

`sessionInfo` R session info at time of training

Method `print()`**:** codeprint method for `rtMod` object

Usage:

```
rtMod$print()
```

Method `plot()`**:** `rtMod` plot method

Usage:

```
rtMod$plot(estimate = NULL, theme = rtTheme, filename = NULL, ...)
```

Arguments:

`estimate` Character: "fitted" or "predicted"

`theme` Theme to pass to plotting function

`filename` Character: Path to file to save plot

`...` Additional arguments passed to plotting function

Method `plotFitted()`**:** Plot fitted values

Usage:

```
rtMod$plotFitted(
  print.plot = TRUE,
  theme = rtTheme,
  main = NULL,
  filename = NULL,
  ...
)
```

Arguments:

`print.plot` Logical: If TRUE show plot

`theme` Theme to be passed on to plotting function

`main` Character: main title

`filename` Character: path to file to save plot

`...` Additional arguments passed to plotting function

Method `plotPredicted()`**:** Plot predicted values

Usage:

```
rtMod$plotPredicted(
  print.plot = TRUE,
  theme = rtTheme,
  main = NULL,
  filename = NULL,
  ...
)
```

Arguments:

`print.plot` Logical: If TRUE show plot

`theme` Theme to be passed on to plotting function

`main` Character: main title

`filename` Character: path to file to save plot

`...` Additional arguments passed to plotting function

Method `plotFittedPredicted()`**:** Plot fitted and predicted values

Usage:

```
rtMod$plotFittedPredicted(
  print.plot = TRUE,
  theme = rtTheme,
  filename = NULL,
  ...
)
```

Arguments:

`print.plot` Logical: If TRUE show plot

`theme` Theme to be passed on to plotting function

`filename` Character: path to file to save plot

`...` Additional arguments passed to `mplot3_conf`

Method `plotVarImp()`**:** Plot variable importance

Usage:

```
rtMod$plotVarImp(
  plot.top = 12,
  type = c("barplot", "lollipop"),
  theme = rtTheme,
  ...
)
```

Arguments:

`plot.top` Integer: Plot this many top features

`type` Character: "barplot" or "lollipop"

`theme` Theme to be passed on to plotting function

`...` Not used

Method `summary()`**:** Summary method for `rtMod` object

Usage:

```
rtMod$summary(
  plots = TRUE,
  cex = 1,
  fit.true.line = "lm",
  resid.fit.line = "gam",
  fit.legend = TRUE,
  se.fit = TRUE,
  single.fig = TRUE,
  theme = rtTheme,
  title.col = NULL,
  ...
)
```

Arguments:

`plots` Logical: If TRUE show plots

`cex` Numeric: Character expansion factor

`fit.true.line` Character: algorithm to use to fit true vs predicted curve

`resid.fit.line` Character: algorithm to use to fit residuals plot

`fit.legend` Logical: If TRUE, show legend in fit plot

`se.fit` Logical: If TRUE, include standard error band in fit plot

`single.fig` Logical: If TRUE, combine in single plot

`theme` Theme to pass to plotting functions

`title.col` Title color

`...` Extra argument to pass to

Method `describe()`**:** Describe model

Usage:

```
rtMod$describe()
```

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
rtMod$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

Author(s)

E.D. Gennatas

`rtMod-methods` `rtMod` *S3 methods*

Description

S3 methods for `rtMod` class. Excludes functions `print` and `plot` defined within the rtMod class itself.

Usage

```
## S3 method for class 'rtMod'
print(x, ...)

## S3 method for class 'rtMod'
fitted(object, ...)

## S3 method for class 'rtMod'
predict(object, newdata, trace = 0, verbose = TRUE, ...)

## S3 method for class 'rtMod'
residuals(object, ...)

## S3 method for class 'rtMod'
plot(x, estimate = NULL, theme = rtTheme, filename = NULL, ...)

## S3 method for class 'rtMod'
summary(
  object,
  plots = TRUE,
  cex = 1,
  fit.true.line = "lm",
  resid.fit.line = "gam",
```

```
  fit.legend = TRUE,
  se.fit = TRUE,
  single.fig = TRUE,
  summary = TRUE,
  theme = rtTheme,
  title.col = NULL,
  ...
)

## S3 method for class 'rtMod'
coef(object, verbose = TRUE, ...)

## S3 method for class 'rtModLite'
predict(object, newdata, ...)
```

Arguments

x	rtMod object
...	Additional argument passed to predict(object)
object	rtModLite object
newdata	Testing set features
trace	Integer: Set trace level
verbose	Logical: If TRUE, output messages to console
estimate	Character: "fitted" or "predicted"
theme	Character: theme to use. Options: "box", "darkbox", "light", "dark"
filename	Character: Path to file to save plot
plots	Logical: If TRUE, print plots. Default = TRUE
cex	Float: Character expansion factor
fit.true.line	**rtemis** algorithm to use for fitted vs. true line Options: modSelect()
resid.fit.line	**rtemis** algorithm to use for residuals vs. fitted line. Options: modSelect()
fit.legend	Logical: If TRUE, print fit legend. Default = TRUE
se.fit	Logical: If TRUE, plot 2 * standard error bands. Default = TRUE
single.fig	Logical: If TRUE, draw all plots in a single figure. Default = TRUE
summary	Logical: If TRUE, print summary. Default = TRUE
title.col	Color for main title

Author(s)

E.D. Gennatas

E.D. Gennatas

rtModBag-class **rtemis** *Bagged Supervised Model Class*

Description

rtemis Bagged Supervised Model Class

rtemis Bagged Supervised Model Class

Details

R6 Class for **rtemis** Bagged Supervised Models

Super class

`rtemis::rtMod` -> `rtModBag`

Public fields

`bag.resample.rtset` List of settings for resample. Set using rtset.bag.resample

`fitted.bag` Base learners' fitted values

`se.fit.bag` Base learners' fitted values' standard error

`predicted.bag` Base learners' predicted values

`se.predicted.bag` Base learners' predicted values' standard error

`aggr.fn` Function used to aggregated base learners' predictions

Methods

Public methods:

- `rtModBag$new()`
- `rtModBag$print()`
- `rtModBag$clone()`

Method `new()`**:** Initialize `rtModBag` object

Usage:

```
rtModBag$new(
  mod.name = character(),
  y.train = numeric(),
  y.test = numeric(),
  x.name = character(),
  y.name = character(),
  xnames = character(),
  bag.resample.rtset = list(),
  mod = list(),
  type = character(),
  parameters = list(),
  fitted.bag = numeric(),
  fitted = numeric(),
  se.fit.bag = numeric(),
  se.fit = numeric(),
```

```
  error.train = list(),
  predicted.bag = numaric(),
  predicted = numeric(),
  se.predicted.bag = numeric(),
  se.prediction = numeric(),
  aggr.fn = character(),
  error.test = list(),
  varimp = NULL,
  question = character(),
  extra = list()
)
```

Arguments:

`mod.name` Model (algorithm) name

`y.train` Training y data

`y.test` Testing y data

`x.name` Name of x data

`y.name` Name of y data

`xnames` Character vector: Column names of x

`bag.resample.rtset` List of settings for resample. Set using rtset.bag.resample

`mod` Trained model

`type` Character: Type of model (Regression, Classification, Survival)

`parameters` List of training parameters

`fitted.bag` Base learners' fitted values

`fitted` Fitted values

`se.fit.bag` Base learners' fitted values' standard error

`se.fit` Standard error of the fit

`error.train` Training error

`predicted.bag` Base learners' predicted values

`predicted` Predicted values

`se.predicted.bag` Base learners' predicted values' standard error

`se.prediction` Standard error of the prediction

`aggr.fn` Aggregating function

`error.test` Testing error

`varimp` Variable importance

`question` Question the model is hoping to answer

`extra` Algorithm-specific output

Method `print()`**:** `print` method for `rtModBag` object

Usage:

```
rtModBag$print()
```

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
rtModBag$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

rtModBag-methods *rtModBag S3 methods*

Description

S3 methods for rtModBag class that differ from those of the rtMod superclass

Usage

```
## S3 method for class 'rtModBag'
predict(object, newdata, aggr.fn = NULL, n.cores = 1, verbose = FALSE, ...)
```

Arguments

object	rtModBag object
newdata	Testing set features
aggr.fn	Character: Function to aggregate models' prediction. If NULL, defaults to "median"
n.cores	Integer: Number of cores to use
...	Not used

rtModClass-class **rtemis** *Classification Model Class*

Description

rtemis Classification Model Class

rtemis Classification Model Class

Details

R6 Class for **rtemis** Classification Models

Super class

rtemis::rtMod -> rtModClass

Public fields

fitted.prob Training set probability estimates

predicted.prob Testing set probability estimates

Methods

Public methods:

- rtModClass$new()
- rtModClass$plotROC()
- rtModClass$plotROCfitted()
- rtModClass$plotROCpredicted()
- rtModClass$plotPR()
- rtModClass$plotPRfitted()
- rtModClass$plotPRpredicted()
- rtModClass$clone()

Method `new()`**:** Initialize `rtModClass` object

Usage:

```
rtModClass$new(
  mod.name = character(),
  y.train = numeric(),
  y.test = numeric(),
  x.name = character(),
  y.name = character(),
  xnames = character(),
  mod = list(),
  type = character(),
  gridsearch = list(),
  parameters = list(),
  fitted = numeric(),
  fitted.prob = numeric(),
  se.fit = numeric(),
  error.train = list(),
  predicted = NULL,
  predicted.prob = NULL,
  se.prediction = NULL,
  error.test = NULL,
  varimp = NULL,
  question = character(),
  extra = list()
)
```

Arguments:

`mod.name` Character: Algorithm name

`y.train` Training set output

`y.test` Testing set output

`x.name` Character: Feature set name

`y.name` Character: Output name

`xnames` Character vector: Feature names

`mod` Trained model

`type` Character: Type of model (Regression, Classification, Survival)

`gridsearch` Grid search output

`parameters` List of training parameters

`fitted` Fitted values (training set predictions)

`fitted.prob` Training set probability estimates
`se.fit` Standard error of the fit
`error.train` Training set error
`predicted` Predicted values (Testing set predictions)
`predicted.prob` Testing set probability estimates
`se.prediction`
`error.test` Testing set error
`varimp` Variable importance
`question` Question the model is trying to answer
`extra` List of extra model info
`sessionInfo` R session info at time of training

Method `plotROC()`**:** plot ROC. Uses testing set if available, otherwise training

Usage:

```
rtModClass$plotROC(theme = rtTheme, filename = NULL, ...)
```

Arguments:

`theme` Theme to pass to plotting function
`filename` Character: Path to file to save plot
`...` Extra arguments to pass to plotting function

Method `plotROCfitted()`**:** Plot training set ROC

Usage:

```
rtModClass$plotROCfitted(
  main = "ROC Training",
  theme = rtTheme,
  filename = NULL,
  ...
)
```

Arguments:

`main` Character: Main title
`theme` Theme to pass to plotting function
`filename` Character: Path to file to save plot
`...` Extra arguments to pass to plotting function

Method `plotROCpredicted()`**:** plot testing set ROC

Usage:

```
rtModClass$plotROCpredicted(
  main = "ROC Testing",
  theme = rtTheme,
  filename = NULL,
  ...
)
```

Arguments:

`main` Character: Main title
`theme` Theme to pass to plotting function
`filename` Character: Path to file to save plot
`...` Extra arguments to pass to plotting function

Method `plotPR()`**:** plot Precision-Recall curve. Uses testing set if available, otherwise training

Usage:

```
rtModClass$plotPR(theme = rtTheme, filename = NULL, ...)
```

Arguments:

`theme` Theme to pass to plotting function

`filename` Character: Path to file to save plot

`...` Extra arguments to pass to plotting function

Method `plotPRfitted()`**:** Plot training set Precision-Recall curve.

Usage:

```
rtModClass$plotPRfitted(
  main = "P-R Training",
  theme = rtTheme,
  filename = NULL,
  ...
)
```

Arguments:

`main` Character: Main title

`theme` Theme to pass to plotting function

`filename` Character: Path to file to save plot

`...` Extra arguments to pass to plotting function

Method `plotPRpredicted()`**:** plot testing set Precision-Recall curve.

Usage:

```
rtModClass$plotPRpredicted(
  main = "P-R Testing",
  theme = rtTheme,
  filename = NULL,
  ...
)
```

Arguments:

`main` Character: Main title

`theme` Theme to pass to plotting function

`filename` Character: Path to file to save plot

`...` Extra arguments to pass to plotting function

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
rtModClass$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

`rtModCV-class` **rtemis** *Cross-Validated Supervised Model Class*

Description

rtemis Cross-Validated Supervised Model Class

rtemis Cross-Validated Supervised Model Class

Details

R6 Class for **rtemis** Cross-Validated Supervised Models

Public fields

`mod` `rtModCV` object

`mod.name` Model (algorithm) name

`type` "Classification", "Regression", or "Survival"

`y.train` Training set y data

`x.name` Name of x data

`y.name` Name of y data

`xnames` Character vector: Column names of x

`parameters` List of algorithm hyperparameters

`n.repeats` Integer: Number of repeats. This is the outermost iterator: i.e. You will run `resampler` this many times.

`resampler.params` List of resampling parameters

`resamples` Resamples produced by resample

`y.train.res` Resamples' fitted values

`y.train.res.aggr` Aggregated training set outcomes

`fitted.res` Resamples' fitted values

`fitted.res.aggr` Aggregated fitted values

`error.train.res` Resamples' training error

`error.train.res.mean` Resamples' mean training error

`error.train.res.aggr` Error of aggregated fitted values

`error.train.repeats` Mean training error for each repeat

`error.train.repeats.mean` Mean training error across all repeats

`error.train.repeats.sd` Standard deviation of training error across all repeats

`y.test.res` Resamples' predicted values

`y.test.res.aggr` Aggregated testing set outcomes

`predicted.res` Resamples' predicted values

`predicted.res.aggr` Aggregated predicted values

`error.test.res` Resamples' testing error

`error.test.res.mean` Resamples' mean testing error

`error.test.res.aggr` Error of aggregated predicted values

`error.test.repeats` Mean testing error for each repeat

`error.test.repeats.mean` Mean testing error across all repeats

`error.test.repeats.sd` Standard deviation of testing error across all repeats

`fitted.bag` Bagged model's fitted values

`error.bag` Bagged model's error

`varimp` Resamples' variable importance

`question` Question the model is hoping to answer

`call` elevate call

`sessionInfo` R session info at time of training

Methods

Public methods:

- `rtModCV$new()`
- `rtModCV$print()`
- `rtModCV$plot()`
- `rtModCV$plotPredicted()`
- `rtModCV$plotFitted()`
- `rtModCV$plotVarImp()`
- `rtModCV$describe()`
- `rtModCV$clone()`

Method `new()`**:** Initialize `rtModCV` object

Usage:

```
rtModCV$new(
  mod = NULL,
  mod.name = NULL,
  type = NULL,
  y.train = NULL,
  x.name = NULL,
  y.name = NULL,
  xnames = NULL,
  parameters = NULL,
  n.repeats = NULL,
  resampler.params = NULL,
  resamples = NULL,
  y.train.res = NULL,
  y.train.res.aggr = NULL,
  fitted.res = NULL,
  fitted.res.aggr = NULL,
  error.train.res = NULL,
  error.train.res.mean = NULL,
  error.train.res.aggr = NULL,
  error.train.repeats = NULL,
  error.train.repeats.mean = NULL,
  error.train.repeats.sd = NULL,
  y.test.res = NULL,
```

```
  y.test.res.aggr = NULL,
  predicted.res = NULL,
  predicted.res.aggr = NULL,
  error.test.res = NULL,
  error.test.res.mean = NULL,
  error.test.res.aggr = NULL,
  error.test.repeats = NULL,
  error.test.repeats.mean = NULL,
  error.test.repeats.sd = NULL,
  fitted.bag = NULL,
  error.bag = NULL,
  varimp = NULL,
  call = NULL,
  question = NULL
)
```

Arguments:

`mod` `rtModCV` object

`mod.name` Model (algorithm) name

`type` "Classification", "Regression", or "Survival"

`y.train` Training set y data

`x.name` Name of x data

`y.name` Name of y data

`xnames` Character vector: Column names of x

`parameters` List of algorithm hyperparameters

`n.repeats` Integer: Number of repeats. This is the outermost iterator: i.e. You will run `resampler` this many times.

`resampler.params` List of resampling parameters

`resamples` Resamples produced by resample

`y.train.res` Resamples' fitted values

`y.train.res.aggr` Aggregated training set outcomes

`fitted.res` Resamples' fitted values

`fitted.res.aggr` Aggregated fitted values

`error.train.res` Resamples' training error

`error.train.res.mean` Resamples' mean training error

`error.train.res.aggr` Error of aggregated fitted values

`error.train.repeats` Mean training error for each repeat

`error.train.repeats.mean` Mean training error across all repeats

`error.train.repeats.sd` Standard deviation of training error across all repeats

`y.test.res` Resamples' predicted values

`y.test.res.aggr` Aggregated testing set outcomes

`predicted.res` Resamples' predicted values

`predicted.res.aggr` Aggregated predicted values

`error.test.res` Resamples' testing error

`error.test.res.mean` Resamples' mean testing error

`error.test.res.aggr` Error of aggregated predicted values

`error.test.repeats` Mean testing error for each repeat

`error.test.repeats.mean` Mean testing error across all repeats

`error.test.repeats.sd` Standard deviation of testing error across all repeats

`fitted.bag` Bagged model's fitted values

`error.bag` Bagged model's error

`varimp` Resamples' variable importance

`call` elevate call

`question` Question the model is hoping to answer

`y.test` Testing set y data

`resampler` List of settings for resample. Set using rtset.cv.resample

`sessionInfo` R session info at time of training

Method `print()`**:** `print` method for `rtModCV` object

Usage:

```
rtModCV$print()
```

Method `plot()`**:** `plot` method for `rtModCV` object

Usage:

```
rtModCV$plot(which.repeat = 1, ...)
```

Arguments:

`which.repeat` Integer: which repeat to plot

`...` Additional arguments passed to plotting function

Method `plotPredicted()`**:** Plot predicted values

Usage:

```
rtModCV$plotPredicted(
  which.repeat = 1,
  theme = rtTheme,
  filename = NULL,
  mar = c(2.5, 3, 2.5, 1),
  ...
)
```

Arguments:

`which.repeat` Integer: which repeat to plot

`theme` rtemis theme

`filename` Character: path to file to save plot

`mar` Numeric vector of plot margins

`...` Additional arguments passed to plotting function

Method `plotFitted()`**:** Plot fitted values

Usage:

```
rtModCV$plotFitted(
  which.repeat = 1,
  theme = rtTheme,
  filename = NULL,
  mar = c(2.5, 3, 2.5, 1),
  ...
)
```

Arguments:

`which.repeat` Integer: which repeat to plot

theme rtemis theme

filename Character: path to file to save plot

mar Numeric vector of plot margins

... Additional arguments passed to plotting function

Method `plotVarImp()`**:** Plot variable importance

Usage:

```
rtModCV$plotVarImp(
  which.repeat = 1,
  type = c("barplot", "lollipop"),
  plot.top = 12,
  theme = rtTheme,
  ...
)
```

Arguments:

which.repeat Integer: which repeat to plot

type Character: "barplot" or "lollipop"

plot.top Integer: Plotting this many top features

theme rtemis theme to use

... Additional arguments passed to plotting function

Method `describe()`**:** Describe `rtModCV`

Usage:

```
rtModCV$describe()
```

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
rtModCV$clone(deep = FALSE)
```

Arguments:

deep Whether to make a deep clone.

`rtModCV-methods` *S3 methods for* `rtModCV` *class that differ from those of the* `rtMod` *superclass*

Description

S3 methods for `rtModCV` class that differ from those of the `rtMod` superclass

`plot.rtModCV`: `plot` method for `rtModCV` object

`summary.rtModCV`: `summary` method for `rtModCV` object

`predict.rtModCV`: `predict` method for `rtModCV` object

Usage

```
## S3 method for class 'rtModCV'
plot(x, ...)

## S3 method for class 'rtModCV'
summary(object, ...)

## S3 method for class 'rtModCV'
predict(object, newdata, which.repeat = 1, bag.fn = mean, n.cores = 1, ...)
```

Arguments

x	rtModCV object
...	Not used
object	rtModCV object
newdata	Set of predictors to use
which.repeat	Integer: Which repeat to use for prediction
bag.fn	Function to use to average predictions of different models
n.cores	Integer: Number of cores to use

---

rtModCVClass-class **rtemis** *Cross-Validated Classification Model Class*

Description

rtemis Cross-Validated Classification Model Class

rtemis Cross-Validated Classification Model Class

Details

R6 Class for **rtemis** Cross-Validated Classification Models

Super class

rtemis::rtModCV -> rtModCVClass

Public fields

`fitted.prob.aggr` Aggregated training set probability estimates

`predicted.prob.aggr` Aggregated testing set probability estimates

Methods

Public methods:

- rtModCVClass$new()
- rtModCVClass$plotROC()
- rtModCVClass$plotROCfitted()
- rtModCVClass$plotROCpredicted()

- rtModCVClass$plotPR()
- rtModCVClass$plotPRfitted()
- rtModCVClass$plotPRpredicted()
- rtModCVClass$clone()

Method `new()`**:** Initialize `rtModCVClass` object

Usage:

```
rtModCVClass$new(
  mod = NULL,
  mod.name = NULL,
  type = NULL,
  y.train = NULL,
  x.name = NULL,
  y.name = NULL,
  xnames = NULL,
  parameters = NULL,
  n.repeats = NULL,
  resampler.params = NULL,
  resamples = NULL,
  y.train.res = NULL,
  y.train.res.aggr = NULL,
  fitted.res = NULL,
  fitted.res.aggr = NULL,
  fitted.prob.aggr = NULL,
  error.train.res = NULL,
  error.train.res.mean = NULL,
  error.train.res.aggr = NULL,
  error.train.repeats = NULL,
  error.train.repeats.mean = NULL,
  error.train.repeats.sd = NULL,
  y.test.res = NULL,
  y.test.res.aggr = NULL,
  predicted.res = NULL,
  predicted.res.aggr = NULL,
  predicted.prob.aggr = NULL,
  error.test.res = NULL,
  error.test.res.mean = NULL,
  error.test.res.aggr = NULL,
  error.test.repeats = NULL,
  error.test.repeats.mean = NULL,
  error.test.repeats.sd = NULL,
  fitted.bag = NULL,
  error.bag = NULL,
  varimp = NULL,
  call = NULL,
  question = NULL
)
```

Arguments:

`mod` `rtModCV` object

`mod.name` Model (algorithm) name

`type` "Classification", "Regression", or "Survival"

`y.train` Training set y data

`x.name` Name of x data

`y.name` Name of y data

`xnames` Character vector: Column names of x

`parameters` List of algorithm hyperparameters

`n.repeats` Integer: Number of repeats. This is the outermost iterator: i.e. You will run `resampler` this many times.

`resampler.params` List of resampling parameters

`resamples` Resamples produced by resample

`y.train.res` Resamples' fitted values

`y.train.res.aggr` Aggregated training set outcomes

`fitted.res` Resamples' fitted values

`fitted.res.aggr` Aggregated fitted values

`fitted.prob.aggr` Aggregated training set probability estimates

`error.train.res` Resamples' training error

`error.train.res.mean` Resamples' mean training error

`error.train.res.aggr` Error of aggregated fitted values

`error.train.repeats` Mean training error for each repeat

`error.train.repeats.mean` Mean training error across all repeats

`error.train.repeats.sd` Standard deviation of training error across all repeats

`y.test.res` Resamples' predicted values

`y.test.res.aggr` Aggregated testing set outcomes

`predicted.res` Resamples' predicted values

`predicted.res.aggr` Aggregated predicted values

`predicted.prob.aggr` Aggregated testing set probability estimates

`error.test.res` Resamples' testing error

`error.test.res.mean` Resamples' mean testing error

`error.test.res.aggr` Error of aggregated predicted values

`error.test.repeats` Mean testing error for each repeat

`error.test.repeats.mean` Mean testing error across all repeats

`error.test.repeats.sd` Standard deviation of testing error across all repeats

`fitted.bag` Bagged model's fitted values

`error.bag` Bagged model's error

`varimp` Resamples' variable importance

`call` elevate call

`question` Question the model is hoping to answer

`y.test` Testing set y data

`resampler` List of settings for resample. Set using rtset.cv.resample

`sessionInfo` R session info at time of training

Method `plotROC()`**:** Plot ROC

Usage:

```
rtModCVClass$plotROC(which.repeat = 1, ...)
```

Arguments:

`which.repeat` Integer: Which repeat to plot

`...` Additional arguments passed to plotting function

Method `plotROCfitted()`**:** Plot ROC plot for fitted values

Usage:

```
rtModCVClass$plotROCfitted(which.repeat = 1, main = "ROC Training", ...)
```

Arguments:

`which.repeat` Integer: Which repeat to plot

`main` Character: Main title

`...` Additional arguments passed to plotting function

Method `plotROCpredicted()`**:** Plot ROC plot for predicted values

Usage:

```
rtModCVClass$plotROCpredicted(which.repeat = 1, main = "ROC Testing", ...)
```

Arguments:

`which.repeat` Integer: Which repeat to plot

`main` Character: Main title

`...` Additional arguments passed to plotting function

Method `plotPR()`**:** Plot Precision-Recall curve

Usage:

```
rtModCVClass$plotPR(which.repeat = 1, ...)
```

Arguments:

`which.repeat` Integer: Which repeat to use

`...` Additional arguments passed to `plotPRpredicted` method

Method `plotPRfitted()`**:** Plot fitted Precision-Recall curve

Usage:

```
rtModCVClass$plotPRfitted(which.repeat = 1, main = "P-R Training", ...)
```

Arguments:

`which.repeat` Integer: Which repeat to use

`main` Character: main title

`...` Additional arguments passed to `mplot3_pr`

Method `plotPRpredicted()`**:** Plot predicted Precision-Recall curve

Usage:

```
rtModCVClass$plotPRpredicted(which.repeat = 1, main = "P-R Testing", ...)
```

Arguments:

`which.repeat` Integer: Which repeat to use

`main` Character: main title

`...` Additional arguments passed to `mplot3_pr`

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
rtModCVClass$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

Author(s)

E.D. Gennatas

`rtModLite-class` **rtemis** *Lite Supervised Model Class*

Description

rtemis Lite Supervised Model Class

rtemis Lite Supervised Model Class

Details

R6 Class for **rtemis** Lite Supervised Models

Note on terminology for all models: The training set is used to build a model. The testing set is a separate set never touched during training and only used to a. create predictions using the trained model and b. estimate error metrics. This reflects generalizability of the model and is the error we care the most about. It is saved in rtemis models as `error.test`. The validation set is used during model tuning. Within rtemis, validation sets are created and used automatically by resample when appropriate, they are not generally input by the user (with few exceptions).

Public fields

`mod.name` Algorithm name

`mod` Trained model

`fitted` Vector of fitted values

Methods

Public methods:

- `rtModLite$new()`
- `rtModLite$print()`
- `rtModLite$clone()`

Method `new()`**:** Initialize `rtModLite` object

Usage:

```
rtModLite$new(mod = list(), mod.name = character(), fitted = numeric())
```

Arguments:

`mod` Trained model

`mod.name` Algorithm name

`fitted` Vector of fitted values

Method `print()`**:** Plot method for `rtModLite` object

Usage:

```
rtModLite$print()
```

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
rtModLite$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

rtModLite-methods *rtModLite S3 methods*

Description

S3 methods for `rtModLite` class.

Usage

```
## S3 method for class 'rtModLite'
print(x, ...)
```

Arguments

`x`	`rtModLite` object
`...`	Not used

rtModLog-class **rtemis** *Supervised Model Log Class*

Description

rtemis Supervised Model Log Class

rtemis Supervised Model Log Class

Public fields

`mod.name` Learner algorithm name

`parameters` List of hyperparameters used when building model

`error.train` Training error

`error.test` Testing error

`sessionInfo` The output of `sessionInfo()` at the time the model was trained

Methods

Public methods:

- `rtModLog$new()`
- `rtModLog$print()`
- `rtModLog$clone()`

Method `new()`**:** Initialize `rtModLog` object

Usage:

```
rtModLog$new(
  mod.name = character(),
  parameters = list(),
  error.train = list(),
  error.test = NULL
)
```

Arguments:

`mod.name` Learner algorithm name

`parameters` List of hyperparameters used when building model

`error.train` Training error

`error.test` Testing error

Method `print()`**:** Print method for `rtModLog` object

Usage:

```
rtModLog$print()
```

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
rtModLog$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

Author(s)

E.D. Gennatas

`rtModLogger-class` **rtemis** *model logger*

Description

rtemis model logger

rtemis model logger

Details

R6 class to save trained models' parameters and performance. Keep your experiment results tidy in one place, with an option to write out to a multi-sheet Excel file.

Public fields

`mods` List of trained models

Methods

Public methods:

- `rtModLogger$new()`
- `rtModLogger$print()`
- `rtModLogger$add()`
- `rtModLogger$summarize()`
- `rtModLogger$summary()`
- `rtModLogger$tabulate()`
- `rtModLogger$plot()`
- `rtModLogger$clone()`

Method `new()`: Initialize `rtModLogger` object

Usage:

```
rtModLogger$new(mods = list())
```

Arguments:

`mods` List of trained models

Method `print()`: Print method for `rtModLogger` object

Usage:

```
rtModLogger$print()
```

Method `add()`: Add model to logger

Usage:

```
rtModLogger$add(mod, verbose = TRUE)
```

Arguments:

`mod` Model to add

`verbose` Logical: If TRUE, print messages to console

Method `summarize()`: Summary method for `rtModLogger`

Usage:

```
rtModLogger$summarize(
  class.metric = "Balanced Accuracy",
  reg.metric = "Rsq",
  surv.metric = "Coherence",
  decimal.places = 3,
  print.metric = FALSE
)
```

Arguments:

`class.metric` Character: Metric to use for Classification models

`reg.metric` Character: Metric to use for Regression models

`surv.metric` Character: Metric to use for Survival models

`decimal.places` Integer: Number of decimal places to display

`print.metric` Logical: If TRUE, print metric name

Method `summary()`: Summary method for `rtModLogger`

Usage:

```
rtModLogger$summary(
  class.metric = "Balanced Accuracy",
  reg.metric = "Rsq",
  surv.metric = "Coherence"
)
```

Arguments:

`class.metric` Character: Metric to use for Classification models

`reg.metric` Character: Metric to use for Regression models

`surv.metric` Character: Metric to use for Survival models

Method `tabulate()`: Tabulate models' parameters and performance

Usage:

```
rtModLogger$tabulate(filename = NULL)
```

Arguments:

`filename` Character: Path to file to save parameters and performance - will be saved as .xlsx file with multiple sheets

Method `plot()`**:** Plot method for `rtModLogger`

Usage:

```
rtModLogger$plot(
  names = NULL,
  col = unlist(rtpalette(rtPalette)),
  mar = NULL,
  ...
)
```

Arguments:

`names` Character: Model names

`col` Colors to use

`mar` Float, vector: plot margins

`...` Additional arguments to pass to plotting function

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
rtModLogger$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

Author(s)

E.D. Gennatas

`rtpalette` **rtemis** *Color Palettes*

Description

`rtPalette` prints names of available color palettes Each palette is a named list of hexadecimal color definitions which can be used with any graphics function.

Usage

```
rtpalette(palette = NULL)
```

Arguments

`palette`	Character: Name of palette to return. Default = NULL: available palette names are printed and no palette is returned

Value

A list of available palettes, invisibly

Examples

```
rtpalette("imperial")
```

`rtPalettes` *UCSF Colors*

Description

`ucsfCol`: UCSF color palette (https://identity.ucsf.edu/brand-guide/color)

`ucsfPalette`: Subset of `ucsfCol`

`ucdCol`: UC Davis color palette (https://marketingtoolbox.ucdavis.edu/visual-identity/color.html)

`berkeleyCol`: Berkeley color palette (https://brand.berkeley.edu/colors/)

`ucscCol`: UC Santa Cruz color palette (https://communications.ucsc.edu/visual-design/color/)

`ucmercedCol`: UC Merced color palette (https://publicrelations.ucmerced.edu/color-guidelines)

`ucsbCol`: UC Santa Barbara color palette (https://www.ucsb.edu/visual-identity/color)

`uclaCol`: UCLA color palette (http://brand.ucla.edu/identity/colors)

`ucrCol`: UC Riverside color palette (https://brand.ucr.edu/ucr-colors)

`uciCol`: UCI color palette (https://communications.uci.edu/campus-resources/graphic-standards/colors.php)

`ucsdCol`: UC San Diego color palette (https://ucpa.ucsd.edu/brand/elements/color-palette/)

`californiaCol`: University of California color palette (http://brand.universityofcalifornia.edu/guidelines/color.html#!pr
colors)

`stanfordCol`: Stanford color palette (https://identity.stanford.edu/color.html#digital-color)

`csuCol`: California State University color palette (https://www2.calstate.edu/csu-system/csu-branding-standards/Documents/CSU-Brand-Guidelines-8-2018.pdf)

`calpolyCol`: Cal Poly color palette (https://universitymarketing.calpoly.edu/brand-guidelines/colors/)

`caltechCol`: Caltech color palette (http://identity.caltech.edu/colors)

`scrippsCol`: Scripps Research color palette

`pennCol`: Penn color palette (http://www.upenn.edu/about/styleguide-color-type)

`cmuCol`: Carnegie Mellon color palette (https://www.cmu.edu/marcom/brand-standards/web-standards.html#colors)

`mitCol`: MIT color palette (http://web.mit.edu/graphicidentity/colors.html)

`princetonCol`: Princeton color palette (https://communications.princeton.edu/guides-tools/logo-graphic-identity)

`columbiaCol`: Columbia color palette (https://visualidentity.columbia.edu/content/web-0)

`brownCol`: Brown color palette (https://www.brown.edu/university-identity/sites/university-identity/files/Brown_Visual_
07-22.pdf)

`yaleCol`: Yale color palette (https://yaleidentity.yale.edu/web)

`cornellCol`: Yale color palette (https://brand.cornell.edu/design-center/colors/

`hmsCol`: Harvard Medical School color palette (https://identityguide.hms.harvard.edu/color)

`dartmouthCol`: Dartmouth color palette (https://communications.dartmouth.edu/visual-identity/design-elements/color-palette#web

usfCol: USF color palette (https://myusf.usfca.edu/marketing-communications/resources/graphics-resources/brand-standards/color-palette) Color conversions performed using https://www.pantone.com/color-finder/

uwCol: University of Washington color palette (http://www.washington.edu/brand/graphic-elements/primary-color-palette/)

jhuCol: Johns Hopkins University color palette (https://brand.jhu.edu/color/)

nyuCol: NYU color palette (https://www.nyu.edu/employees/resources-and-services/media-and-communications/styleguide/website/graphic-visual-design.html)

chicagoCol: University of Chicago color palette (https://news.uchicago.edu/sites/default/files/attachments/_uchicago.id

texasCol: Penn State color palette (https://brand.psu.edu/design-essentials.html#color)

sfsuCol: SF State color palette (https://logo.sfsu.edu/color-system)

illinoisCol: University of Illinois color palette (https://www.uillinois.edu/OUR/brand/color_palettes)

umdCol: University of Maryland color palette (https://osc.umd.edu/licensing-trademarks/brand-standards/logos/#color)

texasCol: University of Texas color palette (https://brand.utexas.edu/identity/color/)

emoryCol: Emory color palette (https://brand.emory.edu/color.html)

techCol: Georgia Tech color palette (http://www.licensing.gatech.edu/super-block/239)

vanderbiltCol: Vanderbilt color palette (https://www.vanderbilt.edu/communications/brand/color.php)

jeffersonCol: Jefferson color palette (http://creative.jefferson.edu/downloads/Jefferson-Brand-Guidelines.pdf)

hawaiiCol: University of Hawaii color palette (https://www.hawaii.edu/offices/eaur/graphicsstandards.pdf)

nihCol: NIH color palette (https://www.nlm.nih.gov/about/nlm_logo_guidelines_030414_508.pdf)

imperialCol: Imperial College London colour palette (https://www.imperial.ac.uk/brand-style-guide/visual-identity/brand-colours/)

uclCol: UCL colour palette (https://www.ucl.ac.uk/cam/brand/guidelines/colour)

oxfordCol: Oxford University colour palette (https://www.ox.ac.uk/sites/files/oxford/media_wysiwyg/Oxford

nhsCol: NHS colour palette (https://www.england.nhs.uk/nhsidentity/identity-guidelines/colours/)

ubcCol: UBC color palette (http://assets.brand.ubc.ca/downloads/ubc_colour_guide.pdf)

torontoCol: U Toronto color palette (https://trademarks.utoronto.ca/colors-fonts/)

mcgillCol: McGill color palette (https://www.mcgill.ca/visual-identity/visual-identity-guide)

ethCol: ETH color palette (https://ethz.ch/services/en/service/communication/corporate-design/colour.html)

rwthCol: RWTH Aachen color palette (http://www9.rwth-aachen.de/global/show_document.asp?id=aaaaaaaaaadpbhq)

mozillaCol: Mozilla design colors (https://mozilla.design/mozilla/color/)

firefoxCol: Firefox design colors (https://mozilla.design/firefox/color/)

appleCol: Apple Human Interface Guidelines color palette (https://developer.apple.com/design/human-interface-guidelines/ios/visual-design/color/)

googleCol: Google brand palette (https://brandpalettes.com/google-colors/)

amazonCol: Amazon brand palette (https://images-na.ssl-images-amazon.com/images/G/01/AdvertisingSite/pdfs/Amazo

microsoftCol: Microsoft brand palette (https://brandcolors.net/b/microsoft)

Usage

```
ucsfLegacyCol

ucsfPalette

ucdCol

berkeleyCol

ucscCol

ucmercedCol

ucsbCol

uclaCol

ucrColor

uciCol

ucsdCol

californiaCol

stanfordCol

csuCol

calpolyCol

caltechCol

scrippsCol

pennCol

pennPalette

pennLightPalette

cmuCol

mitCol

princetonCol

columbiaCol

brownCol

yaleCol
```

cornellCol

hmsCol

dartmouthCol

usfCol

uwCol

jhuCol

nyuCol

chicagoCol

pennstateCol

sfsuCol

illinoisCol

umdCol

texasCol

emoryCol

techCol

vanderbiltCol

jeffersonCol

hawaiiCol

nihCol

imperialCol

uclCol

oxfordCol

nhsCol

ubcCol

torontoCol

mcgillCol

`ethCol`

`rwthCol`

`mozillaCol`

`firefoxCol`

`appleCol`

`googleCol`

`amazonCol`

`microsoftCol`

Format

An object of class `list` of length 13.
An object of class `list` of length 8.
An object of class `list` of length 18.
An object of class `list` of length 18.
An object of class `list` of length 9.
An object of class `list` of length 7.
An object of class `list` of length 10.
An object of class `list` of length 13.
An object of class `list` of length 3.
An object of class `list` of length 8.
An object of class `list` of length 11.
An object of class `list` of length 17.
An object of class `list` of length 27.
An object of class `list` of length 3.
An object of class `list` of length 18.
An object of class `list` of length 19.
An object of class `list` of length 6.
An object of class `list` of length 30.
An object of class `list` of length 11.
An object of class `list` of length 5.
An object of class `list` of length 8.
An object of class `list` of length 3.
An object of class `list` of length 2.
An object of class `list` of length 19.
An object of class `list` of length 8.
An object of class `list` of length 10.

An object of class `list` of length 14.

An object of class `list` of length 14.

An object of class `list` of length 14.

An object of class `list` of length 3.

An object of class `list` of length 3.

An object of class `list` of length 21.

An object of class `list` of length 17.

An object of class `list` of length 25.

An object of class `list` of length 20.

An object of class `list` of length 9.

An object of class `list` of length 15.

An object of class `list` of length 3.

An object of class `list` of length 10.

An object of class `list` of length 22.

An object of class `list` of length 3.

An object of class `list` of length 9.

An object of class `list` of length 8.

An object of class `list` of length 11.

An object of class `list` of length 2.

An object of class `list` of length 26.

An object of class `list` of length 23.

An object of class `list` of length 24.

An object of class `list` of length 21.

An object of class `list` of length 6.

An object of class `list` of length 2.

An object of class `list` of length 27.

An object of class `list` of length 10.

An object of class `list` of length 60.

An object of class `list` of length 8.

An object of class `list` of length 8.

An object of class `list` of length 8.

An object of class `list` of length 4.

An object of class `list` of length 2.

An object of class `list` of length 4.

rtrandom *Random* **rtemis** *art*

Description

Draw random shapes and colors

Usage

```
rtrandom(
  pch = sample(15:18, 1),
  col = rtCol,
  text = "rtemis",
  text.col = "gray50",
  text.as.legend = FALSE,
  legend.bg = NULL,
  legend.alpha = 1,
  random.bg = TRUE
)
```

Arguments

pch	Point character to use
col	Colors to use
text	Text to print
text.col	Color for text

Author(s)

E.D. Gennatas

rtROC *Build an ROC curve*

Description

Calculate the points of an ROC curve and the AUC

Usage

```
rtROC(
  true.labels,
  predicted.probabilities,
  thresholds = NULL,
  plot = FALSE,
  theme = rtTheme,
  verbose = TRUE
)
```

Arguments

`true.labels`	Factor with true labels
`predicted.probabilities`	Numeric vector of predicted probabilities / estimated score
`thresholds`	Numeric vector of thresholds to consider
`plot`	Logical: If TRUE, print plot
`theme`	rtemis theme to use
`vecrboe`	Logical: If TRUE, print messages to console

Details

`true.labels` should be a factor (will be coerced to one) where the first level is the "positive" case. `predicted.probabilities` should be a vector of floats 0, 1 where `[0, .5)` corresponds to the first level and `[.5, 1]` corresponds to the second level. predicted.probabilities

Author(s)

E.D. Gennatas

`rtSave` *Write* **rtemis** *model to RDS file*

Description

Write **rtemis** model to RDS file

Usage

```
rtSave(rtmod, outdir, file.prefix = "s.", verbose = TRUE)
```

Arguments

`rtmod`	**rtemis** model
`outdir`	Path to output directory
`file.prefix`	Character: Prefix for filename
`verbose`	Logical: If TRUE, print messages to output

Author(s)

E.D. Gennatas

rtset — **rtemis** *default-setting functions*

Description

These functions output lists of default settings for different **rtemis** functions. This removes the need of passing named lists of arguments, and provides autocompletion, making it easier to setup functions without having to refer to the manual.

Author(s)

E.D. Gennatas

rtset.bag.resample — rtset.bag.resample*: resample defaults for* rtMod *bagging*

Description

rtset.bag.resample: resample defaults for rtMod bagging

Usage

```
rtset.bag.resample(
  resampler = "strat.sub",
  n.resamples = 10,
  stratify.var = NULL,
  train.p = 0.75,
  strat.n.bins = 4,
  target.length = NULL,
  verbose = TRUE
)
```

Arguments

resampler	Character: Type of resampling to perform: "bootstrap", "kfold", "strat.boot", "strat.sub". Default = "strat.boot" for length(y) < 200, otherwise "strat.sub"
n.resamples	Integer: Number of training/testing sets required
stratify.var	Numeric vector (optional): Variable used for stratification. Defaults to y
train.p	Float (0, 1): Fraction of cases to assign to traininig set for resampler = "strat.sub"
strat.n.bins	Integer: Number of groups to use for stratification for resampler = "strat.sub" / "strat.boot"
target.length	Integer: Number of cases for training set for resampler = "strat.boot". Default = length(y)
verbose	Logical: If TRUE, print messages to screen

`rtset.color` `rtset.color`: *Set parameters for colorGrad*

Description

`rtset.color`: Set parameters for colorGrad

Usage

```
rtset.color(
  n = 101,
  colors = NULL,
  space = "rgb",
  lo = "#01256E",
  lomid = NULL,
  mid = "white",
  midhi = NULL,
  hi = "#95001A",
  colorbar = FALSE,
  cb.mar = c(1, 1, 1, 1),
  ...
)
```

Arguments

`n`	Integer: How many distinct colors you want. If not odd, converted to `n + 1` Defaults to 21
`colors`	Character: Acts as a shortcut to defining `lo`, `mid`, etc for a number of defaults: "french", "penn", "grnblkred",
`space`	Character: Which colorspace to use. Option: "rgb", or "Lab". Default = "rgb". Recommendation: If `mid` is "white" or "black" (default), use "rgb", otherwise "Lab"
`lo`	Color for low end
`lomid`	Color for low-mid
`mid`	Color for middle of the range or "mean", which will result in `colorOp(c(lo, hi), "mean")`. If `mid = NA`, then only `lo` and `hi` are used to create the color gradient.
`midhi`	Color for middle-high
`hi`	Color for high end
`colorbar`	Logical: Create a vertical colorbar
`cb.mar`	Vector, length 4: Colorbar margins. Default: c(1, 1, 1, 1)

rtset.cv.resample rtset.cv.resample: *resample defaults for cross-validation*

Description

rtset.cv.resample: resample defaults for cross-validation

Usage

```
rtset.cv.resample(
  resampler = "kfold",
  n.resamples = 10,
  stratify.var = NULL,
  train.p = 0.75,
  strat.n.bins = 4,
  target.length = NULL,
  verbose = TRUE
)
```

Arguments

resampler	Character: Type of resampling to perform: "bootstrap", "kfold", "strat.boot", "strat.sub". Default = "strat.boot" for length(y) < 200, otherwise "strat.sub"
n.resamples	Integer: Number of training/testing sets required
stratify.var	Numeric vector (optional): Variable used for stratification. Defaults to y
train.p	Float (0, 1): Fraction of cases to assign to traininig set for resampler = "strat.sub"
strat.n.bins	Integer: Number of groups to use for stratification for resampler = "strat.sub" / "strat.boot"
target.length	Integer: Number of cases for training set for resampler = "strat.boot". Default = length(y)
verbose	Logical: If TRUE, print messages to screen

rtset.decompose rtset.decompose: *Set decomposition parameters for* elevate*'s* .decompose *argument*

Description

rtset.decompose: Set decomposition parameters for elevate's .decompose argument

Usage

```
rtset.decompose(decom = "ICA", k = 2, ...)
```

Arguments

decom	Character: Name of decomposer to use. Default = "ICA"
k	Integer: Number of dimensions to project to. Default = 2

`rtset.DN` `rtset.DN`*: Set parameters for s_DN*

Description

`rtset.DN`: Set parameters for s_DN

Usage

```
rtset.DN(
  hidden = 1,
  activation = NULL,
  learning.rate = 0.8,
  momentum = 0.5,
  learningrate_scale = 1,
  output = NULL,
  numepochs = 100,
  batchsize = NULL,
  hidden_dropout = 0,
  visible_dropout = 0,
  ...
)
```

`rtset.earlystop` `rtset.earlystop`*: Set parameters for earlystop*

Description

`rtset.earlystop`: Set parameters for earlystop

Usage

```
rtset.earlystop(
  window = 150,
  window_decrease_pct_min = 0.01,
  total_decrease_pct_max = NULL
)
```

Arguments

`window`	Integer: Number of steps to consider
`window_decrease_pct_min`	Float: Stop if improvement is less than this percent over last ‘window‘ steps
`total_decrease_pct_max`	Float: Stop if improvement from first to last step exceeds this percent. If defined, overrides ‘window_decrease_pct_min‘

`rtset.GBM` `rtset.GBM`: *Set parameters for s_GBM*

Description

`rtset.GBM`: Set parameters for s_GBM

Usage

```
rtset.GBM(
  interaction.depth = 2,
  shrinkage = 0.001,
  max.trees = 5000,
  min.trees = 100,
  bag.fraction = 0.9,
  n.minobsinnode = 5,
  grid.resample.rtset = rtset.resample("kfold", 5),
  ipw = TRUE,
  upsample = FALSE,
  downsample = FALSE,
  resample.seed = NULL,
  ...
)
```

Arguments

`interaction.depth`	[gS] Integer: Interaction depth. Default = 2
`shrinkage`	[gS] Float: Shrinkage (learning rate). Default = .01
`bag.fraction`	[gS] Float (0, 1): Fraction of cases to use to train each tree. Helps avoid overfitting. Default = .75
`n.minobsinnode`	[gS] Integer: Minimum number of observation allowed in node. Default = 5
`ipw`	Logical: If TRUE, apply inverse probability weighting (for Classification only). Note: If `weights` are provided, `ipw` is not used. Default = TRUE
`upsample`	Logical: If TRUE, upsample cases to balance outcome classes (for Classification only) Note: upsample will randomly sample with replacement if the length of the majority class is more than double the length of the class you are upsampling, thereby introducing randomness
`downsample`	Logical: If TRUE, downsample majority class to match size of minority class
`resample.seed`	Integer: If provided, will be used to set the seed during upsampling. Default = NULL (random seed)

rtset.grid.resample rtset.grid.resample: *resample defaults for gridSearchLearn*

Description

rtset.grid.resample: resample defaults for gridSearchLearn

Usage

```
rtset.grid.resample(
  resampler = "kfold",
  n.resamples = 5,
  stratify.var = NULL,
  train.p = 0.75,
  strat.n.bins = 4,
  target.length = NULL,
  verbose = TRUE
)
```

Arguments

resampler	Character: Type of resampling to perform: "bootstrap", "kfold", "strat.boot", "strat.sub". Default = "strat.boot" for length(y) < 200, otherwise "strat.sub"
n.resamples	Integer: Number of training/testing sets required
stratify.var	Numeric vector (optional): Variable used for stratification. Defaults to y
train.p	Float (0, 1): Fraction of cases to assign to traininig set for resampler = "strat.sub"
strat.n.bins	Integer: Number of groups to use for stratification for resampler = "strat.sub" / "strat.boot"
target.length	Integer: Number of cases for training set for resampler = "strat.boot". Default = length(y)
verbose	Logical: If TRUE, print messages to screen

rtset.LIHAD rtset.LIHAD: *Set parameters for s_LIHAD*

Description

rtset.LIHAD: Set parameters for s_LIHAD

Usage

```
rtset.LIHAD(
  max.depth = 2,
  learning.rate = 1,
  lincoef.params = rtset.lincoef("glmnet"),
  alpha = 0,
  lambda = 0.1,
```

```
  minobsinnode = 2,
  minobsinnode.lin = 20,
  ...
)
```

Arguments

max.depth	[gS] Integer: Max depth of additive tree. Default = 3
learning.rate	[gS] Float (0, 1): Learning rate. Default = 1
lincoef.params	Named List: Output of rtset.lincoef
alpha	[gS] Float: lincoef alpha Overrides lincoef.params alpha
lambda	[gS] Float: lincoef lambda. Overrides lincoef.params lambda
minobsinnode	[gS] Integer: Minimum N observations needed in node, before considering splitting

---

rtset.lincoef rtset.lincoef: *Set parameters for lincoef*

Description

rtset.lincoef: Set parameters for lincoef

Usage

```
rtset.lincoef(
  method = c("glmnet", "cv.glmnet", "lm.ridge", "allSubsets", "forwardStepwise",
    "backwardStepwise", "glm", "sgd", "solve"),
  alpha = 0,
  lambda = 0.01,
  lambda.seq = NULL,
  cv.glmnet.nfolds = 5,
  which.cv.glmnet.lambda = c("lambda.min", "lambda.1se"),
  nbest = 1,
  nvmax = 8,
  sgd.model = "glm",
  sgd.model.control = list(lambda1 = 0, lambda2 = 0),
  sgd.control = list(method = "ai-sgd")
)
```

Arguments

method	Character: Method to use: • "glm": uses stats::lm.wfit • "glmnet": uses glmnet::glmnet • "cv.glmnet": uses glmnet:cv.glmnet • "lm.ridge": uses MASS::lm.ridge • "allsubsets": uses leaps::regsubsets with method = "exhaustive" • "forwardStepwise": uses leaps::regsubsets with method = "forward" • "backwardStepwise": uses leaps::regsubsets with method = "backward"

- "sgd": uses sgd::sgd
- "solve": uses base::solve
- "none": fits no model and returns all zeroes, for programming convenience in special cases

alpha — Float: alpha for method = glmnet or cv.glmnet. Default = 0

lambda — Float: The lambda value for glmnet, cv.glmnet, lm.ridge Note: For glmnet and cv.glmnet, this is the lambda used for prediction. Training uses lambda.seq. Default = .01

lambda.seq — Float, vector: lambda sequence for glmnet and cv.glmnet. Default = NULL

cv.glmnet.nfolds — Integer: Number of folds for cv.glmnet

which.cv.glmnet.lambda — Character: Whitch lambda to pick from cv.glmnet: "lambda.min": Lambda that gives minimum cross-validated error;

nbest — Integer: For method = "allSubsets", number of subsets of each size to record. Default = 1

nvmax — Integer: For method = "allSubsets", maximum number of subsets to examine.

sgd.model — Character: Model to use for method = "sgd". Default = "glm"

sgd.model.control — List: model.control list to pass to sgd::sgd

sgd.control — List: sgd.control list to pass to sgd::sgd "lambda.1se": Largest lambda such that error is within 1 s.e. of the minimum.

---

rtset.MARS — rtset.MARS*: Set parameters for s_MARS*

Description

rtset.MARS: Set parameters for s_MARS

Usage

```
rtset.MARS(
  hidden = 1,
  activation = NULL,
  learning.rate = 0.8,
  momentum = 0.5,
  learningrate_scale = 1,
  output = NULL,
  numepochs = 100,
  batchsize = NULL,
  hidden_dropout = 0,
  visible_dropout = 0,
  ...
)
```

Arguments

... — Additional parameters to pass to earth::earth

rtset.meta.resample rtset.meta.resampler: *resample defaults for meta model training*

Description

rtset.meta.resampler: resample defaults for meta model training

Usage

```
rtset.meta.resample(
  resampler = "strat.sub",
  n.resamples = 4,
  stratify.var = NULL,
  train.p = 0.75,
  strat.n.bins = 4,
  target.length = NULL,
  verbose = TRUE
)
```

Arguments

resampler	Character: Type of resampling to perform: "bootstrap", "kfold", "strat.boot", "strat.sub". Default = "strat.boot" for length(y) < 200, otherwise "strat.sub"
n.resamples	Integer: Number of training/testing sets required
stratify.var	Numeric vector (optional): Variable used for stratification. Defaults to y
train.p	Float (0, 1): Fraction of cases to assign to traininig set for resampler = "strat.sub"
strat.n.bins	Integer: Number of groups to use for stratification for resampler = "strat.sub" / "strat.boot"
target.length	Integer: Number of cases for training set for resampler = "strat.boot". Default = length(y)
verbose	Logical: If TRUE, print messages to screen

rtset.preprocess rtset.preprocess: *Set preprocess parameters for elevate's* .preprocess *argument*

Description

rtset.preprocess: Set preprocess parameters for elevate's .preprocess argument

Usage

```
rtset.preprocess(
  completeCases = FALSE,
  removeCases.thres = NULL,
  removeFeatures.thres = NULL,
  impute = FALSE,
  impute.type = "missRanger",
  impute.missRanger.params = list(pmm.k = 0, maxiter = 10),
  impute.discrete = getMode,
  impute.numeric = mean,
  integer2factor = FALSE,
  integer2numeric = FALSE,
  logical2factor = FALSE,
  logical2numeric = FALSE,
  numeric2factor = FALSE,
  numeric2factor.levels = NULL,
  character2factor = FALSE,
  nonzeroFactors = FALSE,
  scale = FALSE,
  center = FALSE,
  removeConstants = TRUE,
  oneHot = FALSE,
  exclude = NULL
)
```

Arguments

decom	Character: Name of decomposer to use. Default = "ICA"
k	Integer: Number of dimensions to project to. Default = 2

---

rtset.RANGER rtset.RANGER: *Set parameters for s_RANGER*

Description

rtset.RANGER: Set parameters for s_RANGER

Usage

```
rtset.RANGER(
  n.trees = 1000,
  min.node.size = 1,
  mtry = NULL,
  grid.resample.rtset = rtset.resample("kfold", 5),
  ipw = TRUE,
  upsample = FALSE,
  downsample = FALSE,
  resample.seed = NULL,
  ...
)
```

Arguments

n.trees	Integer: Number of trees to grow. Default = 1000
min.node.size	[gS] Integer: Minimum node size
mtry	[gS] Integer: Number of features sampled randomly at each split. Defaults to square root of n of features for classification, and a third of n of features for regression.
grid.resample.rtset	List: Output of rtset.resample defining gridSearchLearn parameters.
ipw	Logical: If TRUE, apply inverse probability weighting (for Classification only). Note: If weights are provided, ipw is not used. Default = TRUE
upsample	Logical: If TRUE, upsample training set cases not belonging in majority outcome group
downsample	Logical: If TRUE, downsample majority class to match size of minority class
resample.seed	Integer: If provided, will be used to set the seed during upsampling. Default = NULL (random seed)
...	Additional arguments to be passed to ranger::ranger

---

rtset.resample rtset.resample: *Set resample settings*

Description

rtset.resample: Set resample settings

Usage

```
rtset.resample(
  resampler = "strat.sub",
  n.resamples = 10,
  stratify.var = NULL,
  train.p = 0.75,
  strat.n.bins = 4,
  target.length = NULL,
  seed = NULL
)
```

Arguments

resampler	Character: Type of resampling to perform: "bootstrap", "kfold", "strat.boot", "strat.sub". Default = "strat.boot" for length(y) < 200, otherwise "strat.sub"
n.resamples	Integer: Number of training/testing sets required
stratify.var	Numeric vector (optional): Variable used for stratification. Defaults to y
train.p	Float (0, 1): Fraction of cases to assign to traininig set for resampler = "strat.sub"
strat.n.bins	Integer: Number of groups to use for stratification for resampler = "strat.sub" / "strat.boot"
target.length	Integer: Number of cases for training set for resampler = "strat.boot". Default = length(y)
seed	Integer: (Optional) Set seed for random number generator, in order to make output reproducible. See ?base::set.seed

`rtversion` *Get rtemis and OS version info*

Description

Get rtemis and OS version info

Usage

```
rtversion()
```

`rtXDecom-class` *R6 class for* **rtemis** *cross-decompositions*

Description

R6 class for **rtemis** cross-decompositions

R6 class for **rtemis** cross-decompositions

Details

rtemis cross-decomposition R6 object

Public fields

`xdecom.name` Character: Name of cross-decomposition algorithm

`k` Integer: Number of projections

`xnames` Character vector: Column names of x

`znames` Character vector: Column names of z

`xdecom` Cross-decomposition model output

`xprojections.train` x data training set projections

`xprojections.test` x data test set data projections

`zprojections.train` z data training set projections

`zprojections.test` z data test set projections

`parameters` Cross-decomposition parameters

`extra` List: Algorithm-specific output

Methods

Public methods:

- `rtXDecom$new()`
- `rtXDecom$print()`
- `rtXDecom$clone()`

Method `new()`:

Usage:

```
rtXDecom$new(
  xdecom.name = character(),
  k = integer(),
  xnames = character(),
  znames = character(),
  xdecom = list(),
  xprojections.train = numeric(),
  xprojections.test = numeric(),
  zprojections.train = numeric(),
  zprojections.test = numeric(),
  parameters = list(),
  extra = list()
)
```

Arguments:

`xdecom.name` Character: Name of cross-decomposition algorithm

`k` Integer: Number of projections

`xnames` Character vector: Column names of x

`znames` Character vector: Column names of z

`xdecom` Cross-decomposition model output

`xprojections.train` x data training set projections

`xprojections.test` x data test set data projections

`zprojections.train` z data training set projections

`zprojections.test` z data test set projections

`parameters` Cross-decomposition parameters

`extra` List: Algorithm-specific output

Method `print()`**:** Print method for `rtXDecom` objects

Usage:

```
rtXDecom$print()
```

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
rtXDecom$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

Author(s)

E.D. Gennatas

rt_gtheme_map ***ggplot2** map theme*

Description

ggplot2 map theme for use with gplot3_map based on `usmap:::theme_map()`

Usage

```
rt_gtheme_map(
  base_size = 9,
  base_family = "Helvetica",
  legend.position = c(0.9, 0),
  plot.margin = ggplot2::unit(c(0.02, 0.1, 0.02, 0.02), "npc")
)
```

Arguments

`base_size`	Float: Character `base_size`. Default = 9
`base_family`	Character: Font family. Default = "Lato"

Author(s)

E.D. Gennatas

ruleDist *Rule distance*

Description

Calculate pairwise distance among a set of rules or between two sets of rules, where each rule defines a subpopulation

Usage

```
ruleDist(
  x,
  rules1,
  rules2 = NULL,
  print.plot = TRUE,
  plot.type = c("static", "interactive"),
  heat.lo = "black",
  heat.mid = NA,
  heat.hi = "#F48024",
  verbose = TRUE
)
```

Arguments

x	Data frame / matrix: Input features (cases by features)
rules1	Character, vector: Rules as combination of conditions on the features of x
rules2	String, vector, Optional: Rules as combination of conditions on the features of x
print.plot	Logical: If TRUE, plot heatmap for calculated distance
plot.type	Character: "static", "interactive": type of graphics to use, base or plotly, respectively. Default = "static"
heat.lo	Color: Heatmap low color. Default = "black"
heat.mid	Color: Heatmap mid color. Default = NA (i.e. create gradient from 'heat.lo' to 'heat.hi')
heat.hi	Colo: Heatmap hi colo. Default = "#F48024" (orange)
verbose	Logical: If TRUE, print console messages. Default = TRUE

Details

If only rules1 is provided, computes pairwise distance among rules1, otherwise computes pairwise distance between rules1 and rules2

Author(s)

E.D. Gennatas

---

rules2medmod — rtemis-internals: *Convert rules from cutoffs to median/mode and range*

Description

Convert rules from cutoffs to median (range) and mode (range) format

Usage

```
rules2medmod(rules, x, .ddSci = TRUE, verbose = TRUE, trace = 0)
```

Arguments

rules	Character, vector: Input rules
x	Data frame: Data to evaluate rules
.ddSci	Logical: If TRUE, format all continuous variables using ddSci, which will give either 2 decimal places, or scientific notation if two decimal places result in 0.00
verbose	Logical: If TRUE, print messages to console.
trace	Integer: If greater than zero, print progress

Author(s)

E.D. Gennatas

runifmat *Random Uniform Matrix*

Description

Create a matrix or data frame of defined dimensions, whose columns are random uniform vectors

Usage

```
runifmat(
  nrow = 10,
  ncol = 10,
  min = 0,
  max = 1,
  return.df = FALSE,
  seed = NULL
)
```

Arguments

nrow	Integer: Number of rows. Default = 10
ncol	Integer: Number of columns. Default = 10
min	Float: Min Default = 0
max	Float: Max. Default = 1
return.df	Logical: If TRUE, return data.frame, otherwise matrix. Default = TRUE
seed	Integer: Set seed for rnorm. Default = NULL

Author(s)

E.D. Gennatas

savePMML *Save rtemis model to PMML file*

Description

Save rtemis model to PMML file

Usage

```
savePMML(
  x,
  filename,
  transforms = NULL,
  model_name = NULL,
  model_version = NULL,
  description = NULL,
  copyright = NULL,
  ...
)
```

Arguments

x	rtemis model
filename	Character: path to file

Author(s)

E.D. Gennatas

---

se *Extract standard error of fit from rtemis model*

Description

Returns mod$se.fit

Usage

```
se(x)
```

Arguments

x	An rtMod object

Value

Standard error of fitted values of mod

Author(s)

E.D. Gennatas

---

seglabels2itksnap *Convert nifti segmentation label file to itksnap-compatible label file*

Description

Convert nifti segmentation label file to itksnap-compatible label file

Usage

```
seglabels2itksnap(
  x,
  visible = NULL,
  mesh = NULL,
  filename = NULL,
  verbose = TRUE
)
```

Arguments

x	Character: path to nifti label file (read with readseglabels)
visible	Integer, vector, length = number of labels in x: 1: visible; 0: invisible. Normally, the background label is set to invisible
mesh	Same as visible but for mesh renderings
filename	Character: Path to filename to write to

Author(s)

E.D. Gennatas

---

selectiter *Select N of learning iterations based on loss*

Description

Select N of learning iterations based on loss

Usage

```
selectiter(
  loss.valid,
  loss.train,
  smooth = TRUE,
  plot = FALSE,
  verbose = FALSE
)
```

Arguments

loss.valid	Float, vector: Validation loss. Can be NULL
loss.train	Float, vector: Training loss
smooth	Logical: If TRUE, smooth loss before finding minimum.
plot	Logical: If TRUE, plot loss curve.
verbose	Logical: If TRUE, print messages to console.

Author(s)

E.D. Gennatas

sensitivity *Sensitivity*

Description

The first factor level is considered the positive case.

Usage

```
sensitivity(true, estimated, harmonize = FALSE, verbose = TRUE)
```

Arguments

true	True labels
estimated	Estimated labels
harmonize	Logical: If TRUE, run factorHarmonize first
verbose	Logical: If TRUE, print messages to output. Default = TRUE

separate_colors *Separate colors*

Description

Separate colors by RGB distance

Usage

```
separate_colors(x, start_with = 1)
```

Arguments

x	Vector of colors
start_with	Integer: Which color to output in first position

Details

Starting with the first color defined by `start_with`, the next color is chosen to be max distance from all preceding colors

Author(s)

E.D. Gennatas

seql *Sequence generation with automatic cycling*

Description

Sequence generation with automatic cycling

Usage

```
seql(x, target)
```

Arguments

x	R object of some length
target	R object of some length

Author(s)

E.D. Gennatas

Examples

```
color <- c("red", "blue")
target <- 1:5
color[seql(color, target)]
# "red"  "blue" "red"  "blue" "red"
color <- c("red", "green", "blue", "yellow", "orange")
target <- 1:3
color[seql(color, target)]
# "red"   "green" "blue"
```

setdiffsym *Symmetric Set Difference*

Description

Symmetric Set Difference

Usage

```
setdiffsym(x, y)
```

Arguments

x	vector
y	vector of same type as x

Author(s)

E.D. Gennatas

Examples

```
setdiff(1:10, 1:5)
setdiff(1:5, 1:10)
setdiffsym(1:10, 1:5)
setdiffsym(1:5, 1:10)
```

`sge_submit` *Submit expression to SGE grid*

Description

Submit expression to SGE grid

Usage

```
sge_submit(
  expr,
  obj_names = NULL,
  packages = NULL,
  queue = NULL,
  n_threads = 4,
  sge_out = file.path(getwd(), "./sge_out"),
  sge_error = sge_out,
  sge_env = "#! /usr/bin/env bash",
  sge_opts = "#$ -cwd",
  R_command = NULL,
  system_command = NULL,
  h_rt = "00:25:00",
  mem_free = NULL,
  temp_dir = file.path(getwd(), ".sge_tempdir"),
  verbose = TRUE,
  trace = 1
)
```

Arguments

`expr`	R expression
`obj_names`	Character vector: Names of objects to copy to cluster R session
`packages`	Character vector: Names of packages to load in cluster R session
`queue`	Character: Name of SGE queue to submit to
`n_threads`	Integer: Number of threads to request from scheduler
`sge_out`	Character: Path to directory to write standard out message files
`sge_error`	Character: Path to directory to write error message files
`sge_env`	Character: Shell environment for script to be submitted to SGE
`sge_opts`	Character: SGE options that will be written in shell script. Default = "#$ -cwd"
`R_command`	Character: Optional R command(s) to run at the beginning of the R script
`system_command`	Character: system command to be run by shell script before executing R code. For example a command that export the R executable to use

h_rt	Character: Max time to request. Default = "00:25:00", i.e. 25 minutes
mem_free	Character: Amount of memory to request from the scheduler
temp_dir	Character: Temporary directory that is accessible to all execution nodes. Default = file.path(getwd(), ".sge_tempdir")
verbose	Logical: If TRUE, print messages to console. Default = TRUE
trace	Integer: If > 0 print diagnostic messages to console.

Author(s)

E.D. Gennatas

---

sigmoid *Sigmoid function*

Description

Sigmoid function

Usage

```
sigmoid(x)
```

Arguments

x	Vector, float: Input

---

size *Size of matrix or vector*

Description

Return the size of a matrix or vector as (Nrows, Ncolumns) Are you tired of getting NULL when you run dim() on a vector?

Usage

```
size(x)
```

Arguments

x	Vector or matrix input

Value

Integer vector of length 2: c(Nrow, Ncols)

Author(s)

E.D. Gennatas

Examples

```
x <- rnorm(20)
size(x)
# 20  1

x <- matrix(rnorm(100), 20, 5)
size(x)
# 20  5
```

`softmax` *Softmax function*

Description

Softmax function

Usage

```
softmax(x)
```

Arguments

x	Vector, Float: Input

`softplus` *Softplus function*

Description

Softplus function:

$$log(1 + e^x)$$

Usage

```
softplus(x)
```

Arguments

x	Vector, Float: Input

sortedlines *lines, but sorted*

Description

lines, but sorted

Usage

```
sortedlines(x, y, col = "red", ...)
```

Arguments

x	Input vector
y	Input vector
col	Line color. Default = "red"
...	Extra params to pass to lines

Author(s)

E.D. Gennatas

sparsernorm *Sparse rnorm*

Description

A sparse version of stats::rnorm Outputs a vector where a fraction of values are zeros (determined by sparseness) and the rest are drawn from a random normal distribution using stats::rnorm

Usage

```
sparsernorm(n, sparseness = 0.1, mean = 0, sd = 1)
```

Arguments

n	Integer: Length of output vector
sparseness	Float (0, 1): Fraction of required nonzero elements, i.e. output will have round(sparseness * n) nonzero elements. If sparseness = 0, a vector of zeros length n is returned, if sparseness = 1, rnorm(n, mean, sd) is returned. Default = 0.1
mean	Float: Target mean of nonzero elements, passed to stats::rnorm Default = 0
sd	Float: Target sd of nonzero elements, passed to stats::rnorm Default = 1

Author(s)

E.D. Gennatas

sparseVectorSummary *Sparseness and pairwise correlation of vectors*

Description

Get sparseness measure on a matrix of vectors

Usage

```
sparseVectorSummary(vectors, y = NULL)
```

Arguments

vectors	Matrix of column vectors
y	Optional numeric vector.

sparsify *Sparsify a vector*

Description

Keep top x

Usage

```
sparsify(x, sparseness)
```

Arguments

x	Input vector
sparseness	Percent of values of x to keep. The rest will be set to zero.

Author(s)

E.D. Gennatas

`specificity` *Specificity*

Description

The first factor level is considered the positive case.

Usage

```
specificity(true, estimated, harmonize = FALSE, verbose = TRUE)
```

Arguments

`true`	True labels
`estimated`	Estimated labels
`harmonize`	Logical: If TRUE, run factorHarmonize first
`verbose`	Logical: If TRUE, print messages to output. Default = TRUE

`splitlin_` `rtemis` *internal: Ridge and Stump*

Description

Edits environment 'g' in-place (no output)

Usage

```
splitlin_(
  g,
  type,
  node.index,
  gamma,
  n.quantiles,
  minobsinnode,
  minbucket,
  lin.type,
  alpha,
  lambda,
  lambda.seq,
  cv.glmnet.nfolds,
  which.cv.glmnet.lambda,
  nbest,
  nvmax,
  n.cores,
  trace
)
```

Arguments

node.index	Open nodes to work on
	Fit a linear model on (x, y) and split on the gradient Input: environment holding tree and index of node Output: None; Expands tree within environment g by splitting indexed node
tree	Node within tree environment

---

sqcoldist *Squared Color Distance*

Description

Get the squared RGB distance between two colors

Usage

```
sqcoldist(x, y)
```

Arguments

x	Color
y	Color

Author(s)

E.D. Gennatas

Examples

```
sqcoldist("red", "green")
sqcoldist("#16A0AC", "#FA6E1E")
```

---

square *Square*

Description

Square

Usage

```
square(x)
```

Arguments

x	Vector, Float: Input

stderror *Standard Error of the Mean*

Description

Calculate the standard error of the mean, which is equal to the standard deviation divided by the square root of the sample size. NA values are automatically removed

Usage

```
stderror(x)
```

Arguments

x	Vector, numeric: Input data

Author(s)

E.D. Gennatas

strat.boot *Stratified Bootstrap Resampling*

Description

Stratified Bootstrap Resampling

Usage

```
strat.boot(
  x,
  n.resamples = 10,
  train.p = 0.75,
  stratify.var = NULL,
  strat.n.bins = 4,
  target.length = NULL,
  seed = NULL,
  verbose = TRUE
)
```

Arguments

x	Input vector
n.resamples	Integer: Number of training/testing sets required
train.p	Float (0, 1): Fraction of cases to assign to traininig set for `resampler = "strat.sub"`
stratify.var	Numeric vector (optional): Variable used for stratification. Defaults to `y`
strat.n.bins	Integer: Number of groups to use for stratification for `resampler = "strat.sub"` / `"strat.boot"`

target.length	Integer: Number of cases for training set for resampler = "strat.boot". Default = length(y)
seed	Integer: (Optional) Set seed for random number generator, in order to make output reproducible. See ?base::set.seed
verbose	Logical: If TRUE, print messages to screen

Author(s)

E.D. Gennatas

---

strat.sub *Resample using Stratified Subsamples*

Description

Resample using Stratified Subsamples

Usage

```
strat.sub(
  x,
  n.resamples = 10,
  train.p = 0.75,
  stratify.var = NULL,
  strat.n.bins = 4,
  seed = NULL,
  verbose = TRUE
)
```

Arguments

x	Input vector
n.resamples	Integer: Number of training/testing sets required
train.p	Float (0, 1): Fraction of cases to assign to traininig set for resampler = "strat.sub"
stratify.var	Numeric vector (optional): Variable used for stratification. Defaults to y
strat.n.bins	Integer: Number of groups to use for stratification for resampler = "strat.sub" / "strat.boot"
seed	Integer: (Optional) Set seed for random number generator, in order to make output reproducible. See ?base::set.seed
verbose	Logical: If TRUE, print messages to screen

Author(s)

E.D. Gennatas

strata2factor *Convert* survfit *object's strata to a factor*

Description

Convert survfit object's strata to a factor

Usage

```
strata2factor(x)
```

Arguments

x	survfit object

Value

factor

Author(s)

E.D. Gennatas

strict *Strict assignment by class or type*

Description

Allow assignment only if input is of correct class and/or type

Usage

```
strict(x, accept.class = NULL, accept.type = NULL)
```

Arguments

x	Value to be assigned if type matches
accept.class	Required class of value
accept.type	Required type of value

Author(s)

E.D. Gennatas

summarize *Summarize numeric variables*

Description

Summarize numeric variables

Usage

```
summarize(
  x,
  varname,
  group_by = NULL,
  type = c("all", "median-range", "mean-sd"),
  na.rm = TRUE
)
```

Arguments

x	data.frame or data.table (will be coerced to data.table)
varname	Character, vector: Variable name(s) to summarize. Must be column names in x of type numeric.
group_by	Character, vector: Variable name(s) of factors to group by. Must be column names in x. Default = NULL
type	Character: "all", "median-range" or "mean-sd". Default = "all", which returns Mean, SD, Median, Range, NA (number of NA values)
na.rm	Logical: Passed to median and mean. Default = TRUE

Value

data.table with summary

Author(s)

E.D. Gennatas

summary.massGLM massGLM *object summary*

Description

massGLM object summary

Usage

```
## S3 method for class 'massGLM'
summary(object, ...)
```

Arguments

object	An object created by massGLM
...	Not used

Author(s)

E.D. Gennatas

---

survError *Survival Analysis Metrics*

Description

Survival Analysis Metrics

Usage

```
survError(true, estimated)
```

Arguments

true	Vector, numeric: True survival times
estimated	Vector, numeric: Estimated survival times

Author(s)

E.D. Gennatas

---

svd1 rtemis-internals *Project Variables to First Eigenvector*

Description

Convenience function for SVD k = 1

Usage

```
svd1(x, x.test = NULL)
```

Arguments

x	Input matrix / data frame
x.test	Optional test matrix / data frame

Author(s)

E.D. Gennatas

`synthMultiModal` *Create "Multimodal" Synthetic Data*

Description

Create "Multimodal" Synthetic Data using squares and arctangents

Usage

```
synthMultiModal(
  n.cases = 10000,
  init.fn = "runifmat",
  init.fn.params = list(min = -10, max = 10),
  n.groups = 4,
  n.feat.per.group = round(seq(10, 300, length.out = n.groups)),
  contrib.p = 0.33,
  linear.p = 0.66,
  square.p = 0.1,
  atan.p = 0.1,
  pair.multiply.p = 0.05,
  pair.square.p = 0.05,
  pair.atan.p = 0.05,
  verbose = TRUE,
  seed = NULL,
  filename = NULL
)
```

Arguments

`n.cases`	Integer: Number of cases to create. Default = 10000
`init.fn`	Character: "runifmat" or "rnormmat". Use the respective functions to generate features as random uniform and random normal variables, respectively. Default = "runifmat"
`init.fn.params`	Named list with arguments "min", "max" for "runifmat" and "mean", "sd" for "rnormmat". Default = `list(min = -10, max = 10)`
`n.groups`	Integer: Number of feature groups / modalities to create. Default = 4
`n.feat.per.group`	Integer, vector, length `n.groups`: Number of features per group to create. Default = `c(50, 100, 200, 300)`
`contrib.p`	Float (0, 1]: Ratio of features contributing to outcome per group. Default = .33, i.e. a third of the features in each group will be used to produce the outcome y
`linear.p`	Float [0, 1]: Ratio of contributing features to be included linearly. Default = .1, i.e. .1 of .33 of features in each group will be included
`square.p`	Float [0, 1]: Ratio of contributing features to be squared. Default = .1, i.e. .1 of .33 of features in each group will be squared
`atan.p`	Float [0, 1]: Ratio of contributing features whose `atan` will be used. These will be selected from the features that were NOT sampled for squaring. Default = .1, i.e. .1 of .33 of features in each group will be transformed using `atan`, but given these features were not already picked to be squared (see `square.p`)

pair.multiply.p
: Float [0, 1] Ratio of features will be divided into pairs and multiplied. Default = .05

pair.square.p
: Float [0, 1] Ratio of features which will be divided into pairs, multiplied and squared. Default = .05

pair.atan.p
: Float [0, 1] Ratio of features which will be divided into pairs, multiplied and transformed using `atan`. Default = .05

verbose
: Logical: If TRUE, print messages to console. Default = TRUE

seed
: Integer: If set, pass to `set.seed` for reproducibility

filename
: Character: Path to file to save output. Default = NULL

Details

There are no checks yet for compatibility among inputs and certain combinations may not work.

Value

List with elements x, y, index.square, index.atan, index.pair.square

Author(s)

E.D. Gennatas

Examples

```
xmm <- synthMultiModal(n.cases = 10000,
init.fn = "runifmat",
init.fn.params = list(min = -10, max = 10),
n.groups = 5,
n.feat.per.group = c(20, 50, 100, 200, 300),
contrib.p = .33,
linear.p = .66,
square.p = .1,
atan.p = .1,
pair.multiply.p = .1,
pair.square.p = .1,
pair.atan.p = .1,
seed = 2019)
```

---

synthRegData *Synthesize Simple Regression Data*

Description

Synthesize Simple Regression Data

Usage

```
synthRegData(
  nrow = 500,
  ncol = 50,
  noise.sd.factor = 1,
  resample.rtset = rtset.resample(),
  seed = NULL,
  verbose = FALSE
)
```

Arguments

nrow	Integer: Number of rows. Default = 500
ncol	Integer: Number of columns. Default = 50
noise.sd.factor	Numeric: Add rnorm(nrow, sd = noise.sd.factor * sd(y)). Default = 2
resample.rtset	Output of rtset.resample defining training/testing split. The first resulting resample will be used to create dat.train and dat.test output; all resample output under resamples
seed	Integer: Seed for random number generator. Default = NULL
verbose	Logical: If TRUE, print messages to console. Default = FALSE

Value

List with elements dat, dat.train, dat.test, resamples, w, seed

Author(s)

E.D. Gennatas

---

s_ADABOOST *Adaboost Binary Classifier [C]*

Description

Train an Adaboost Classifier using ada::ada

Usage

```
s_ADABOOST(
  x,
  y = NULL,
  x.test = NULL,
  y.test = NULL,
  loss = "exponential",
  type = "discrete",
  iter = 50,
  nu = 0.1,
  bag.frac = 0.5,
  upsample = FALSE,
```

```
  downsample = FALSE,
  resample.seed = NULL,
  x.name = NULL,
  y.name = NULL,
  print.plot = TRUE,
  plot.fitted = NULL,
  plot.predicted = NULL,
  plot.theme = rtTheme,
  question = NULL,
  rtclass = NULL,
  verbose = TRUE,
  trace = 0,
  outdir = NULL,
  save.mod = ifelse(!is.null(outdir), TRUE, FALSE),
  ...
)
```

Arguments

x	Numeric vector or matrix / data frame of features i.e. independent variables
y	Numeric vector of outcome, i.e. dependent variable
x.test	Numeric vector or matrix / data frame of testing set features Columns must correspond to columns in x
y.test	Numeric vector of testing set outcome
loss	Character: "exponential" (Default), "logistic"
type	Character: "discrete", "real", "gentle"
iter	Integer: Number of boosting iterations to perform. Default = 50
nu	Float: Shrinkage parameter for boosting. Default = .1
bag.frac	Float (0, 1]: Sampling fraction for out-of-bag samples
upsample	Logical: If TRUE, upsample cases to balance outcome classes (for Classification only) Note: upsample will randomly sample with replacement if the length of the majority class is more than double the length of the class you are upsampling, thereby introducing randomness
downsample	Logical: If TRUE, downsample majority class to match size of minority class
resample.seed	Integer: If provided, will be used to set the seed during upsampling. Default = NULL (random seed)
x.name	Character: Name for feature set
y.name	Character: Name for outcome
print.plot	Logical: if TRUE, produce plot using mplot3 Takes precedence over plot.fitted and plot.predicted. Default = TRUE
plot.fitted	Logical: if TRUE, plot True (y) vs Fitted
plot.predicted	Logical: if TRUE, plot True (y.test) vs Predicted. Requires x.test and y.test
plot.theme	Character: "zero", "dark", "box", "darkbox"
question	Character: the question you are attempting to answer with this model, in plain language.
rtclass	Character: Class type to use. "S3", "S4", "RC", "R6"
verbose	Logical: If TRUE, print summary to screen.

trace	Integer: If higher than 0, will print more information to the console. Default = 0
outdir	Path to output directory. If defined, will save Predicted vs. True plot, if available, as well as full model output, if save.mod is TRUE
save.mod	Logical: If TRUE, save all output to an RDS file in outdir save.mod is TRUE by default if an outdir is defined. If set to TRUE, and no outdir is defined, outdir defaults to paste0("./s.", mod.name)
...	Additional arguments

Details

ada::ada does not support case weights

Value

rtMod object

Author(s)

E.D. Gennatas

See Also

elevate for external cross-validation

Other Supervised Learning: s_ADDTREE(), s_BART(), s_BAYESGLM(), s_BRUTO(), s_C50(), s_CART(), s_CTREE(), s_ET(), s_EVTREE(), s_GAM.default(), s_GAM.formula(), s_GAMSELX2(), s_GAMSELX(), s_GAMSEL(), s_GAM(), s_GBM0(), s_GBM3.R(), s_GBM(), s_GLMNET(), s_GLMTREE(), s_GLM(), s_GLS(), s_H2ODL(), s_H2OGBM(), s_H2ORF(), s_IRF(), s_KNN(), s_LDA(), s_LMTREE(), s_LM(), s_MARS(), s_MLRF(), s_NBAYES(), s_NLA(), s_NLS(), s_NW(), s_POLYMARS(), s_PPR(), s_PPTREE(), s_QDA(), s_QRNN(), s_RANGER(), s_RFSRC(), s_RF(), s_SDA(), s_SGD(), s_SPLS(), s_SVM(), s_TFN(), s_XGBLIN(), s_XGBOOST(), s_XGB()

Other Tree-based methods: s_ADDTREE(), s_BART(), s_C50(), s_CART(), s_CTREE(), s_ET(), s_EVTREE(), s_GBM0(), s_GBM3.R(), s_GBM(), s_GLMTREE(), s_H2OGBM(), s_H2ORF(), s_IRF(), s_LMTREE(), s_MLRF(), s_PPTREE(), s_RANGER(), s_RFSRC(), s_RF(), s_XGBOOST(), s_XGB()

Other Ensembles: s_GBM0(), s_GBM3.R(), s_GBM(), s_RANGER(), s_RF()

---

s_ADDTREE *Additive Tree: Tree-Structured Boosting [C]*

Description

Train an Additive Tree model

Usage

```
s_ADDTREE(
  x,
  y = NULL,
  x.test = NULL,
  y.test = NULL,
  x.name = NULL,
```

```
  y.name = NULL,
  weights = NULL,
  update = c("exponential", "polynomial"),
  min.update = ifelse(update == "polynomial", 0.035, 1000),
  min.hessian = 0.001,
  min.membership = 1,
  steps.past.min.membership = 0,
  gamma = 0.8,
  max.depth = 30,
  learning.rate = 0.1,
  ipw = TRUE,
  ipw.type = 2,
  upsample = FALSE,
  downsample = FALSE,
  resample.seed = NULL,
  imetrics = TRUE,
  grid.resample.rtset = rtset.resample("kfold", 5),
  metric = "Balanced Accuracy",
  maximize = TRUE,
  rpart.params = NULL,
  match.rules = TRUE,
  print.plot = TRUE,
  plot.fitted = NULL,
  plot.predicted = NULL,
  plot.theme = rtTheme,
  question = NULL,
  rtclass = NULL,
  verbose = TRUE,
  prune.verbose = FALSE,
  trace = 1,
  grid.verbose = verbose,
  outdir = NULL,
  save.rpart = FALSE,
  save.mod = ifelse(!is.null(outdir), TRUE, FALSE),
  n.cores = rtCores,
  ...
)
```

Arguments

x	Numeric vector or matrix / data frame of features i.e. independent variables
y	Numeric vector of outcome, i.e. dependent variable
x.test	Numeric vector or matrix / data frame of testing set features Columns must correspond to columns in x
y.test	Numeric vector of testing set outcome
x.name	Character: Name for feature set
y.name	Character: Name for outcome
weights	Numeric vector: Weights for cases. For classification, weights takes precedence over ipw, therefore set weights = NULL if using ipw. Note: If weight are provided, ipw is not used. Leave NULL if setting ipw = TRUE. Default = NULL

update	Character: "exponential" or "polynomial". Type of weight update. Default = "exponential"
min.update	Float: Minimum update for gradient step
min.hessian	[gS] Float: Minimum second derivative to continue splitting. Default = .001
min.membership	Integer: Minimum number of cases in a node. Default = 1
steps.past.min.membership	Integer: N steps to make past min.membership - For testing. Default = 0
gamma	[gS] Float: acceleration factor = lambda/(1 + lambda). Default = .8
max.depth	[gS] Integer: maximum depth of the tree. Default = 30
learning.rate	[gS] learning rate for the Newton Raphson step that updates the function values of the node
ipw	Logical: If TRUE, apply inverse probability weighting (for Classification only). Note: If weights are provided, ipw is not used. Default = TRUE
ipw.type	Integer 0, 1, 2 1: class.weights as in 0, divided by max(class.weights) 2: class.weights as in 0, divided by min(class.weights) Default = 2
upsample	Logical: If TRUE, upsample cases to balance outcome classes (for Classification only) Note: upsample will randomly sample with replacement if the length of the majority class is more than double the length of the class you are upsampling, thereby introducing randomness
downsample	Logical: If TRUE, downsample majority class to match size of minority class
resample.seed	Integer: If provided, will be used to set the seed during upsampling. Default = NULL (random seed)
imetrics	Logical: If TRUE, save interpretability metrics, i.e. N total nodes in tree and depth, in output. Default = TRUE
rpart.params	List: rpart parameters, passed to rpart::rpart("parms")
match.rules	Logical: If TRUE, match cases to rules to get statistics per node, i.e. what percent of cases match each rule. If available, these are used by dplot3_addtree when plotting. Default = TRUE
print.plot	Logical: if TRUE, produce plot using mplot3 Takes precedence over plot.fitted and plot.predicted. Default = TRUE
plot.fitted	Logical: if TRUE, plot True (y) vs Fitted
plot.predicted	Logical: if TRUE, plot True (y.test) vs Predicted. Requires x.test and y.test
plot.theme	Character: "zero", "dark", "box", "darkbox"
question	Character: the question you are attempting to answer with this model, in plain language.
rtclass	Character: Class type to use. "S3", "S4", "RC", "R6"
verbose	Logical: If TRUE, print summary to screen.
trace	Integer: If higher than 0, will print more information to the console. Default = 0
outdir	Path to output directory. If defined, will save Predicted vs. True plot, if available, as well as full model output, if save.mod is TRUE
save.mod	Logical: If TRUE, save all output to an RDS file in outdir save.mod is TRUE by default if an outdir is defined. If set to TRUE, and no outdir is defined, outdir defaults to paste0("./s.", mod.name)
...	Additional arguments

Details

This function is for binary classification. The outcome must be a factor with two levels, the first level is the 'positive' class. Ensure there are no missing values in the data and that variables are either numeric (including integers) or factors. Use preprocess as needed to impute and convert characters to factors.

Factor levels should not contain the "/" character (it is used to separate conditions in the addtree object)

[gS] Indicates that more than one value can be supplied, which will result in grid search using internal resampling lambda = gamma/(1 - gamma)

Value

Object of class rtMod

Author(s)

E.D. Gennatas

References

Jose Marcio Luna, Efstathios D Gennatas, Lyle H Ungar, Eric Eaton, Eric S Diffenderfer, Shane T Jensen, Charles B Simone, Jerome H Friedman, Timothy D Solberg, Gilmer Valdes Building more accurate decision trees with the additive tree Proc Natl Acad Sci U S A. 2019 Oct 1;116(40):19887-19893. doi: 10.1073/pnas.1816748116

See Also

Other Supervised Learning: s_ADABOOST(), s_BART(), s_BAYESGLM(), s_BRUTO(), s_C50(), s_CART(), s_CTREE(), s_ET(), s_EVTREE(), s_GAM.default(), s_GAM.formula(), s_GAMSELX2(), s_GAMSELX(), s_GAMSEL(), s_GAM(), s_GBM0(), s_GBM3.R(), s_GBM(), s_GLMNET(), s_GLMTREE(), s_GLM(), s_GLS(), s_H2ODL(), s_H2OGBM(), s_H2ORF(), s_IRF(), s_KNN(), s_LDA(), s_LMTREE(), s_LM(), s_MARS(), s_MLRF(), s_NBAYES(), s_NLA(), s_NLS(), s_NW(), s_POLYMARS(), s_PPR(), s_PPTREE(), s_QDA(), s_QRNN(), s_RANGER(), s_RFSRC(), s_RF(), s_SDA(), s_SGD(), s_SPLS(), s_SVM(), s_TFN(), s_XGBLIN(), s_XGBOOST(), s_XGB()

Other Tree-based methods: s_ADABOOST(), s_BART(), s_C50(), s_CART(), s_CTREE(), s_ET(), s_EVTREE(), s_GBM0(), s_GBM3.R(), s_GBM(), s_GLMTREE(), s_H2OGBM(), s_H2ORF(), s_IRF(), s_LMTREE(), s_MLRF(), s_PPTREE(), s_RANGER(), s_RFSRC(), s_RF(), s_XGBOOST(), s_XGB()

Other Interpretable models: s_C50(), s_CART(), s_GLMNET(), s_GLMTREE(), s_GLM(), s_LMTREE()

---

s_BART *Bayesian Additive Regression Trees [C, R]*

Description

Trains a Bayesian Additive Regression Tree (BART) model using package `bartMachine`

Usage

```
s_BART(
  x,
  y = NULL,
  x.test = NULL,
  y.test = NULL,
  x.name = NULL,
  y.name = NULL,
  n.trees = c(100, 200),
  k_cvs = c(2, 3),
  nu_q_cvs = list(c(3, 0.9), c(10, 0.75)),
  k_folds = 5,
  n.burnin = 250,
  n.iter = 1000,
  n.cores = rtCores,
  upsample = FALSE,
  downsample = FALSE,
  resample.seed = NULL,
  print.plot = TRUE,
  plot.fitted = NULL,
  plot.predicted = NULL,
  plot.theme = rtTheme,
  question = NULL,
  rtclass = NULL,
  verbose = TRUE,
  trace = 0,
  outdir = NULL,
  save.mod = ifelse(!is.null(outdir), TRUE, FALSE),
  java.mem.size = 12,
  ...
)
```

Arguments

x	Numeric vector or matrix / data frame of features i.e. independent variables
y	Numeric vector of outcome, i.e. dependent variable
x.test	Numeric vector or matrix / data frame of testing set features Columns must correspond to columns in x
y.test	Numeric vector of testing set outcome
x.name	Character: Name for feature set
y.name	Character: Name for outcome
upsample	Logical: If TRUE, upsample cases to balance outcome classes (for Classification only) Note: upsample will randomly sample with replacement if the length of the majority class is more than double the length of the class you are upsampling, thereby introducing randomness
downsample	Logical: If TRUE, downsample majority class to match size of minority class
resample.seed	Integer: If provided, will be used to set the seed during upsampling. Default = NULL (random seed)
print.plot	Logical: if TRUE, produce plot using mplot3 Takes precedence over plot.fitted and plot.predicted. Default = TRUE

plot.fitted	Logical: if TRUE, plot True (y) vs Fitted
plot.predicted	Logical: if TRUE, plot True (y.test) vs Predicted. Requires x.test and y.test
plot.theme	Character: "zero", "dark", "box", "darkbox"
question	Character: the question you are attempting to answer with this model, in plain language.
rtclass	Character: Class type to use. "S3", "S4", "RC", "R6"
verbose	Logical: If TRUE, print summary to screen.
trace	Integer: If higher than 0, will print more information to the console. Default = 0
outdir	Path to output directory. If defined, will save Predicted vs. True plot, if available, as well as full model output, if save.mod is TRUE
save.mod	Logical: if TRUE, sets bartMachine's serialize to TRUE and saves model to outdir
...	Additional arguments to be passed to bartMachine::bartMachine

Details

Be warned this can take a very long time to train. If you are having trouble with rJava in Rstudio on macOS, see: https://support.rstudio.com/hc/en-us/community/posts/203663956/comments/249073727 bartMachine does not support case weights

Value

Object of class **rtemis**

Author(s)

E.D. Gennatas

See Also

elevate for external cross-validation

Other Supervised Learning: s_ADABOOST(), s_ADDTREE(), s_BAYESGLM(), s_BRUTO(), s_C50(), s_CART(), s_CTREE(), s_ET(), s_EVTREE(), s_GAM.default(), s_GAM.formula(), s_GAMSELX2(), s_GAMSELX(), s_GAMSEL(), s_GAM(), s_GBM0(), s_GBM3.R(), s_GBM(), s_GLMNET(), s_GLMTREE(), s_GLM(), s_GLS(), s_H2ODL(), s_H2OGBM(), s_H2ORF(), s_IRF(), s_KNN(), s_LDA(), s_LMTREE(), s_LM(), s_MARS(), s_MLRF(), s_NBAYES(), s_NLA(), s_NLS(), s_NW(), s_POLYMARS(), s_PPR(), s_PPTREE(), s_QDA(), s_QRNN(), s_RANGER(), s_RFSRC(), s_RF(), s_SDA(), s_SGD(), s_SPLS(), s_SVM(), s_TFN(), s_XGBLIN(), s_XGBOOST(), s_XGB()

Other Tree-based methods: s_ADABOOST(), s_ADDTREE(), s_C50(), s_CART(), s_CTREE(), s_ET(), s_EVTREE(), s_GBM0(), s_GBM3.R(), s_GBM(), s_GLMTREE(), s_H2OGBM(), s_H2ORF(), s_IRF(), s_LMTREE(), s_MLRF(), s_PPTREE(), s_RANGER(), s_RFSRC(), s_RF(), s_XGBOOST(), s_XGB()

s_BAYESGLM *Bayesian GLM*

Description

Train a bayesian GLM using `arm::bayesglm`

Usage

```
s_BAYESGLM(
  x,
  y = NULL,
  x.test = NULL,
  y.test = NULL,
  family = NULL,
  prior.mean = 0,
  prior.scale = NULL,
  prior.df = 1,
  prior.mean.for.intercept = 0,
  prior.scale.for.intercept = NULL,
  prior.df.for.intercept = 1,
  min.prior.scale = 1e-12,
  scaled = TRUE,
  keep.order = TRUE,
  drop.baseline = TRUE,
  maxit = 100,
  x.name = NULL,
  y.name = NULL,
  weights = NULL,
  ipw = TRUE,
  ipw.type = 2,
  upsample = FALSE,
  downsample = FALSE,
  resample.seed = NULL,
  metric = NULL,
  maximize = NULL,
  print.plot = TRUE,
  plot.fitted = NULL,
  plot.predicted = NULL,
  plot.theme = rtTheme,
  question = NULL,
  grid.verbose = verbose,
  verbose = TRUE,
  outdir = NULL,
  save.mod = ifelse(!is.null(outdir), TRUE, FALSE),
  ...
)
```

Arguments

x	Numeric vector or matrix / data frame of features i.e. independent variables

y	Numeric vector of outcome, i.e. dependent variable
x.test	Numeric vector or matrix / data frame of testing set features Columns must correspond to columns in x
y.test	Numeric vector of testing set outcome
family	Error distribution and link function. See stats::family
prior.mean	Float, vector: Prior mean for the coefficients. If scalar, it will be replicated to length N features. Default = 0
prior.scale	Float, vector: Prior scale for the coefficients. Default = NULL, which results in 2.5 for logit, 2.5*1.6 for probit. If scalar, it will be replicated to length N features.
prior.df	Float: Prior degrees of freedom for the coefficients. Set to 1 for t distribution; set to Inf for normal prior distribution. If scalar, it will be replicated to length N features. Default = 1
prior.mean.for.intercept	Float: Default = 0
prior.scale.for.intercept	Float: Default = NULL, which results in 10 for a logit model, and 10*1.6 for probit model
prior.df.for.intercept	Float: Default = 1
min.prior.scale	Float: Minimum prior scale for the coefficients. Default = 1e-12
scaled	Logical: If TRUE, the scale for the prior distributions are: For feature with single value, use prior.scale, for predictor with two values, use prior.scale/range(x), for more than two values, use prior.scale/(2*sd(x)). If response is gaussian, prior.scale is multiplied by 2 * sd(y). Default = TRUE
keep.order	Logical: If TRUE, the feature positions are maintained, otherwise they are reordered: main effects, interactions, second-order, third-order, etc. Default = TRUE
drop.baseline	Logical: If TRUE, drop the base level of factor features. Default = TRUE
maxit	Integer: Maximum number of iterations
x.name	Character: Name for feature set
y.name	Character: Name for outcome
weights	Numeric vector: Weights for cases. For classification, weights takes precedence over ipw, therefore set weights = NULL if using ipw. Note: If weight are provided, ipw is not used. Leave NULL if setting ipw = TRUE. Default = NULL
ipw	Logical: If TRUE, apply inverse probability weighting (for Classification only). Note: If weights are provided, ipw is not used. Default = TRUE
ipw.type	Integer 0, 1, 2 1: class.weights as in 0, divided by max(class.weights) 2: class.weights as in 0, divided by min(class.weights) Default = 2
upsample	Logical: If TRUE, upsample cases to balance outcome classes (for Classification only) Note: upsample will randomly sample with replacement if the length of the majority class is more than double the length of the class you are upsampling, thereby introducing randomness
downsample	Logical: If TRUE, downsample majority class to match size of minority class
resample.seed	Integer: If provided, will be used to set the seed during upsampling. Default = NULL (random seed)

print.plot	Logical: if TRUE, produce plot using mplot3 Takes precedence over plot.fitted and plot.predicted. Default = TRUE
plot.fitted	Logical: if TRUE, plot True (y) vs Fitted
plot.predicted	Logical: if TRUE, plot True (y.test) vs Predicted. Requires x.test and y.test
plot.theme	Character: "zero", "dark", "box", "darkbox"
question	Character: the question you are attempting to answer with this model, in plain language.
verbose	Logical: If TRUE, print summary to screen.
outdir	Path to output directory. If defined, will save Predicted vs. True plot, if available, as well as full model output, if save.mod is TRUE
save.mod	Logical: If TRUE, save all output to an RDS file in outdir save.mod is TRUE by default if an outdir is defined. If set to TRUE, and no outdir is defined, outdir defaults to paste0("./s.", mod.name)
...	Additional parameters to pass to arm::bayesglm

Author(s)

E.D. Gennatas

See Also

Other Supervised Learning: s_ADABOOST(), s_ADDTREE(), s_BART(), s_BRUTO(), s_C50(), s_CART(), s_CTREE(), s_ET(), s_EVTREE(), s_GAM.default(), s_GAM.formula(), s_GAMSELX2(), s_GAMSELX(), s_GAMSEL(), s_GAM(), s_GBM0(), s_GBM3.R(), s_GBM(), s_GLMNET(), s_GLMTREE(), s_GLM(), s_GLS(), s_H2ODL(), s_H2OGBM(), s_H2ORF(), s_IRF(), s_KNN(), s_LDA(), s_LMTREE(), s_LM(), s_MARS(), s_MLRF(), s_NBAYES(), s_NLA(), s_NLS(), s_NW(), s_POLYMARS(), s_PPR(), s_PPTREE(), s_QDA(), s_QRNN(), s_RANGER(), s_RFSRC(), s_RF(), s_SDA(), s_SGD(), s_SPLS(), s_SVM(), s_TFN(), s_XGBLIN(), s_XGBOOST(), s_XGB()

---

s_BRUTO *Projection Pursuit Regression (BRUTO) [R]*

Description

Trains a BRUTO model and validates it

Usage

```
s_BRUTO(
  x,
  y = NULL,
  x.test = NULL,
  y.test = NULL,
  x.name = NULL,
  y.name = NULL,
  grid.resample.rtset = rtset.grid.resample(),
  weights = NULL,
  weights.col = NULL,
  dfmax = 6,
```

```
  cost = 2,
  maxit.select = 20,
  maxit.backfit = 20,
  thresh = 1e-04,
  start.linear = TRUE,
  n.cores = rtCores,
  print.plot = TRUE,
  plot.fitted = NULL,
  plot.predicted = NULL,
  plot.theme = rtTheme,
  question = NULL,
  rtclass = NULL,
  verbose = TRUE,
  trace = 0,
  outdir = NULL,
  save.mod = ifelse(!is.null(outdir), TRUE, FALSE),
  ...
)
```

Arguments

x	Numeric vector or matrix / data frame of features i.e. independent variables
y	Numeric vector of outcome, i.e. dependent variable
x.test	Numeric vector or matrix / data frame of testing set features Columns must correspond to columns in x
y.test	Numeric vector of testing set outcome
x.name	Character: Name for feature set
y.name	Character: Name for outcome
weights	Numeric vector: Weights for cases. For classification, weights takes precedence over ipw, therefore set weights = NULL if using ipw. Note: If weight are provided, ipw is not used. Leave NULL if setting ipw = TRUE. Default = NULL
print.plot	Logical: if TRUE, produce plot using mplot3 Takes precedence over plot.fitted and plot.predicted. Default = TRUE
plot.fitted	Logical: if TRUE, plot True (y) vs Fitted
plot.predicted	Logical: if TRUE, plot True (y.test) vs Predicted. Requires x.test and y.test
plot.theme	Character: "zero", "dark", "box", "darkbox"
question	Character: the question you are attempting to answer with this model, in plain language.
rtclass	Character: Class type to use. "S3", "S4", "RC", "R6"
verbose	Logical: If TRUE, print summary to screen.
trace	Integer: If higher than 0, will print more information to the console. Default = 0
outdir	Path to output directory. If defined, will save Predicted vs. True plot, if available, as well as full model output, if save.mod is TRUE
save.mod	Logical: If TRUE, save all output to an RDS file in outdir save.mod is TRUE by default if an outdir is defined. If set to TRUE, and no outdir is defined, outdir defaults to paste0("./s.", mod.name)
...	Additional arguments to be passed to mda::bruto

Value

Object of class **rtemis**

Author(s)

E.D. Gennatas

See Also

elevate for external cross-validation

Other Supervised Learning: s_ADABOOST(), s_ADDTREE(), s_BART(), s_BAYESGLM(), s_C50(), s_CART(), s_CTREE(), s_ET(), s_EVTREE(), s_GAM.default(), s_GAM.formula(), s_GAMSELX2(), s_GAMSELX(), s_GAMSEL(), s_GAM(), s_GBM0(), s_GBM3.R(), s_GBM(), s_GLMNET(), s_GLMTREE(), s_GLM(), s_GLS(), s_H2ODL(), s_H2OGBM(), s_H2ORF(), s_IRF(), s_KNN(), s_LDA(), s_LMTREE(), s_LM(), s_MARS(), s_MLRF(), s_NBAYES(), s_NLA(), s_NLS(), s_NW(), s_POLYMARS(), s_PPR(), s_PPTREE(), s_QDA(), s_QRNN(), s_RANGER(), s_RFSRC(), s_RF(), s_SDA(), s_SGD(), s_SPLS(), s_SVM(), s_TFN(), s_XGBLIN(), s_XGBOOST(), s_XGB()

---

s_C50 *C5.0 Decision Trees and Rule-Based Models [C]*

Description

Train a C5.0 decision tree using 'C50::C5.0'

Usage

```
s_C50(
  x,
  y = NULL,
  x.test = NULL,
  y.test = NULL,
  trials = 10,
  rules = FALSE,
  weights = NULL,
  ipw = TRUE,
  ipw.type = 2,
  upsample = FALSE,
  downsample = FALSE,
  resample.seed = NULL,
  control = C50::C5.0Control(),
  costs = NULL,
  x.name = NULL,
  y.name = NULL,
  print.plot = TRUE,
  plot.fitted = NULL,
  plot.predicted = NULL,
  plot.theme = rtTheme,
  question = NULL,
  rtclass = NULL,
```

```
  verbose = TRUE,
  trace = 0,
  outdir = NULL,
  save.mod = ifelse(!is.null(outdir), TRUE, FALSE),
  ...
)
```

Arguments

x	Numeric vector or matrix / data frame of features i.e. independent variables
y	Numeric vector of outcome, i.e. dependent variable
x.test	Numeric vector or matrix / data frame of testing set features Columns must correspond to columns in x
y.test	Numeric vector of testing set outcome
trials	Integer [1, 100]: Number of boosting iterations
rules	Logical: If TRUE, decompose the tree to a rule-based model
weights	Numeric vector: Weights for cases. For classification, weights takes precedence over ipw, therefore set weights = NULL if using ipw. Note: If weight are provided, ipw is not used. Leave NULL if setting ipw = TRUE. Default = NULL
ipw	Logical: If TRUE, apply inverse probability weighting (for Classification only). Note: If weights are provided, ipw is not used. Default = TRUE
ipw.type	Integer 0, 1, 2 1: class.weights as in 0, divided by max(class.weights) 2: class.weights as in 0, divided by min(class.weights) Default = 2
upsample	Logical: If TRUE, upsample cases to balance outcome classes (for Classification only) Note: upsample will randomly sample with replacement if the length of the majority class is more than double the length of the class you are upsampling, thereby introducing randomness
downsample	Logical: If TRUE, downsample majority class to match size of minority class
resample.seed	Integer: If provided, will be used to set the seed during upsampling. Default = NULL (random seed)
x.name	Character: Name for feature set
y.name	Character: Name for outcome
print.plot	Logical: if TRUE, produce plot using mplot3 Takes precedence over plot.fitted and plot.predicted. Default = TRUE
plot.fitted	Logical: if TRUE, plot True (y) vs Fitted
plot.predicted	Logical: if TRUE, plot True (y.test) vs Predicted. Requires x.test and y.test
plot.theme	Character: "zero", "dark", "box", "darkbox"
question	Character: the question you are attempting to answer with this model, in plain language.
rtclass	Character: Class type to use. "S3", "S4", "RC", "R6"
verbose	Logical: If TRUE, print summary to screen.
trace	Integer: If higher than 0, will print more information to the console. Default = 0
outdir	Path to output directory. If defined, will save Predicted vs. True plot, if available, as well as full model output, if save.mod is TRUE
save.mod	Logical: If TRUE, save all output to an RDS file in outdir save.mod is TRUE by default if an outdir is defined. If set to TRUE, and no outdir is defined, outdir defaults to paste0("./s.", mod.name)
...	Additional arguments

Value

rtMod object

Author(s)

E.D. Gennatas

See Also

elevate for external cross-validation

Other Supervised Learning: s_ADABOOST(), s_ADDTREE(), s_BART(), s_BAYESGLM(), s_BRUTO(), s_CART(), s_CTREE(), s_ET(), s_EVTREE(), s_GAM.default(), s_GAM.formula(), s_GAMSELX2(), s_GAMSELX(), s_GAMSEL(), s_GAM(), s_GBM0(), s_GBM3.R(), s_GBM(), s_GLMNET(), s_GLMTREE(), s_GLM(), s_GLS(), s_H2ODL(), s_H2OGBM(), s_H2ORF(), s_IRF(), s_KNN(), s_LDA(), s_LMTREE(), s_LM(), s_MARS(), s_MLRF(), s_NBAYES(), s_NLA(), s_NLS(), s_NW(), s_POLYMARS(), s_PPR(), s_PPTREE(), s_QDA(), s_QRNN(), s_RANGER(), s_RFSRC(), s_RF(), s_SDA(), s_SGD(), s_SPLS(), s_SVM(), s_TFN(), s_XGBLIN(), s_XGBOOST(), s_XGB()

Other Tree-based methods: s_ADABOOST(), s_ADDTREE(), s_BART(), s_CART(), s_CTREE(), s_ET(), s_EVTREE(), s_GBM0(), s_GBM3.R(), s_GBM(), s_GLMTREE(), s_H2OGBM(), s_H2ORF(), s_IRF(), s_LMTREE(), s_MLRF(), s_PPTREE(), s_RANGER(), s_RFSRC(), s_RF(), s_XGBOOST(), s_XGB()

Other Interpretable models: s_ADDTREE(), s_CART(), s_GLMNET(), s_GLMTREE(), s_GLM(), s_LMTREE()

---

s_CART *Classification and Regression Trees [C, R, S]*

Description

Train a CART for regression or classification using `rpart`

Usage

```
s_CART(
  x,
  y = NULL,
  x.test = NULL,
  y.test = NULL,
  x.name = NULL,
  y.name = NULL,
  weights = NULL,
  ipw = TRUE,
  ipw.type = 2,
  upsample = FALSE,
  downsample = FALSE,
  resample.seed = NULL,
  method = "auto",
  parms = NULL,
  minsplit = 2,
  minbucket = round(minsplit/3),
  cp = 0.01,
  maxdepth = 20,
```

```
  maxcompete = 0,
  maxsurrogate = 0,
  usesurrogate = 2,
  surrogatestyle = 0,
  xval = 0,
  cost = NULL,
  model = TRUE,
  prune.cp = NULL,
  use.prune.rpart.rt = TRUE,
  return.unpruned = FALSE,
  grid.resample.rtset = rtset.resample("kfold", 5),
  grid.search.type = c("exhaustive", "randomized"),
  grid.randomized.p = 0.1,
  metric = NULL,
  maximize = NULL,
  na.action = na.exclude,
  n.cores = rtCores,
  print.plot = TRUE,
  plot.fitted = NULL,
  plot.predicted = NULL,
  plot.theme = rtTheme,
  question = NULL,
  verbose = TRUE,
  grid.verbose = verbose,
  outdir = NULL,
  save.mod = ifelse(!is.null(outdir), TRUE, FALSE)
)
```

Arguments

x	Numeric vector or matrix / data frame of features i.e. independent variables
y	Numeric vector of outcome, i.e. dependent variable
x.test	Numeric vector or matrix / data frame of testing set features Columns must correspond to columns in x
y.test	Numeric vector of testing set outcome
x.name	Character: Name for feature set
y.name	Character: Name for outcome
weights	Numeric vector: Weights for cases. For classification, weights takes precedence over ipw, therefore set weights = NULL if using ipw. Note: If weight are provided, ipw is not used. Leave NULL if setting ipw = TRUE. Default = NULL
ipw	Logical: If TRUE, apply inverse probability weighting (for Classification only). Note: If weights are provided, ipw is not used. Default = TRUE
ipw.type	Integer 0, 1, 2 1: class.weights as in 0, divided by max(class.weights) 2: class.weights as in 0, divided by min(class.weights) Default = 2
upsample	Logical: If TRUE, upsample cases to balance outcome classes (for Classification only) Note: upsample will randomly sample with replacement if the length of the majority class is more than double the length of the class you are upsampling, thereby introducing randomness
downsample	Logical: If TRUE, downsample majority class to match size of minority class

`resample.seed` Integer: If provided, will be used to set the seed during upsampling. Default = NULL (random seed)

`method` Character: "auto", "anova", "poisson", "class" or "exp".

`parms` List of additional parameters for the splitting function. See `rpart::rpart("parms")`

`minsplit` [gS] Integer: Minimum number of cases that must belong in a node before considering a split.

`minbucket` [gS] Integer: Minimum number of cases allowed in a child node.

`cp` [gS] Float: Complexity threshold for allowing a split.

`maxdepth` [gS] Integer: Maximum depth of tree.

`maxcompete` Integer: The number of competitor splits saved in the output

`maxsurrogate` Integer: The number of surrogate splits retained in the output (See `rpart::rpart.control`).

`usesurrogate` See `rpart::rpart.control`

`surrogatestyle` See `rpart::rpart.control`

`xval` Integer: Number of cross-validations

`cost` Vector, Float (> 0): One for each variable in the model. See `rpart::rpart("cost")`

`model` Logical: If TRUE, keep a copy of the model.

`prune.cp` [gS] Float: Complexity for cost-complexity pruning after tree is built

`use.prune.rpart.rt` [Testing only, do not change]

`return.unpruned` Logical: If TRUE and `prune.cp` is set, return unpruned tree under `extra` in rtMod.

`grid.resample.rtset` List: Output of rtset.resample defining gridSearchLearn parameters.

`grid.search.type` Character: Type of grid search to perform: "exhaustive" or "randomized".

`grid.randomized.p` Float (0, 1): If `grid.search.type = "randomized"`, randomly test this proportion of combinations.

`metric` Character: Metric to minimize, or maximize if `maximize = TRUE` during grid search. Default = NULL, which results in "Balanced Accuracy" for Classification, "MSE" for Regression, and "Coherence" for Survival Analysis.

`maximize` Logical: If TRUE, `metric` will be maximized if grid search is run.

`na.action` How to handle missing values. See `?na.fail`

`n.cores` Integer: Number of cores to use.

`print.plot` Logical: if TRUE, produce plot using `mplot3` Takes precedence over `plot.fitted` and `plot.predicted`. Default = TRUE

`plot.fitted` Logical: if TRUE, plot True (y) vs Fitted

`plot.predicted` Logical: if TRUE, plot True (y.test) vs Predicted. Requires `x.test` and `y.test`

`plot.theme` Character: "zero", "dark", "box", "darkbox"

`question` Character: the question you are attempting to answer with this model, in plain language.

`verbose` Logical: If TRUE, print summary to screen.

`grid.verbose` Logical: Passed to gridSearchLearn

outdir	Path to output directory. If defined, will save Predicted vs. True plot, if available, as well as full model output, if save.mod is TRUE
save.mod	Logical: If TRUE, save all output to an RDS file in outdir save.mod is TRUE by default if an outdir is defined. If set to TRUE, and no outdir is defined, outdir defaults to paste0("./s.", mod.name)

Details

[gS] indicates grid search will be performed automatically if more than one value is passed

Value

Object of class rtMod

Author(s)

E.D. Gennatas

See Also

elevate for external cross-validation

Other Supervised Learning: s_ADABOOST(), s_ADDTREE(), s_BART(), s_BAYESGLM(), s_BRUTO(), s_C50(), s_CTREE(), s_ET(), s_EVTREE(), s_GAM.default(), s_GAM.formula(), s_GAMSELX2(), s_GAMSELX(), s_GAMSEL(), s_GAM(), s_GBM0(), s_GBM3.R(), s_GBM(), s_GLMNET(), s_GLMTREE(), s_GLM(), s_GLS(), s_H2ODL(), s_H2OGBM(), s_H2ORF(), s_IRF(), s_KNN(), s_LDA(), s_LMTREE(), s_LM(), s_MARS(), s_MLRF(), s_NBAYES(), s_NLA(), s_NLS(), s_NW(), s_POLYMARS(), s_PPR(), s_PPTREE(), s_QDA(), s_QRNN(), s_RANGER(), s_RFSRC(), s_RF(), s_SDA(), s_SGD(), s_SPLS(), s_SVM(), s_TFN(), s_XGBLIN(), s_XGBOOST(), s_XGB()

Other Tree-based methods: s_ADABOOST(), s_ADDTREE(), s_BART(), s_C50(), s_CTREE(), s_ET(), s_EVTREE(), s_GBM0(), s_GBM3.R(), s_GBM(), s_GLMTREE(), s_H2OGBM(), s_H2ORF(), s_IRF(), s_LMTREE(), s_MLRF(), s_PPTREE(), s_RANGER(), s_RFSRC(), s_RF(), s_XGBOOST(), s_XGB()

Other Interpretable models: s_ADDTREE(), s_C50(), s_GLMNET(), s_GLMTREE(), s_GLM(), s_LMTREE()

---

s_CTREE *Conditional Inference Trees [C, R, S]*

Description

Train a conditional inference tree using partykit::ctree

Usage

```
s_CTREE(
  x,
  y = NULL,
  x.test = NULL,
  y.test = NULL,
  weights = NULL,
  control = partykit::ctree_control(),
  ipw = TRUE,
  ipw.type = 2,
```

```
  upsample = FALSE,
  downsample = FALSE,
  resample.seed = NULL,
  x.name = NULL,
  y.name = NULL,
  print.plot = TRUE,
  plot.fitted = NULL,
  plot.predicted = NULL,
  plot.theme = rtTheme,
  question = NULL,
  verbose = TRUE,
  outdir = NULL,
  save.mod = ifelse(!is.null(outdir), TRUE, FALSE),
  ...
)
```

Arguments

x	Numeric vector or matrix / data frame of features i.e. independent variables
y	Numeric vector of outcome, i.e. dependent variable
x.test	Numeric vector or matrix / data frame of testing set features Columns must correspond to columns in x
y.test	Numeric vector of testing set outcome
weights	Numeric vector: Weights for cases. For classification, weights takes precedence over ipw, therefore set weights = NULL if using ipw. Note: If weight are provided, ipw is not used. Leave NULL if setting ipw = TRUE. Default = NULL
control	List of parameters for the CTREE algorithms. Set using partykit::ctree_control
ipw	Logical: If TRUE, apply inverse probability weighting (for Classification only). Note: If weights are provided, ipw is not used. Default = TRUE
ipw.type	Integer 0, 1, 2 1: class.weights as in 0, divided by max(class.weights) 2: class.weights as in 0, divided by min(class.weights) Default = 2
upsample	Logical: If TRUE, upsample cases to balance outcome classes (for Classification only) Note: upsample will randomly sample with replacement if the length of the majority class is more than double the length of the class you are upsampling, thereby introducing randomness
downsample	Logical: If TRUE, downsample majority class to match size of minority class
resample.seed	Integer: If provided, will be used to set the seed during upsampling. Default = NULL (random seed)
x.name	Character: Name for feature set
y.name	Character: Name for outcome
print.plot	Logical: if TRUE, produce plot using mplot3 Takes precedence over plot.fitted and plot.predicted. Default = TRUE
plot.fitted	Logical: if TRUE, plot True (y) vs Fitted
plot.predicted	Logical: if TRUE, plot True (y.test) vs Predicted. Requires x.test and y.test
plot.theme	Character: "zero", "dark", "box", "darkbox"
question	Character: the question you are attempting to answer with this model, in plain language.

verbose	Logical: If TRUE, print summary to screen.
outdir	Path to output directory. If defined, will save Predicted vs. True plot, if available, as well as full model output, if save.mod is TRUE
save.mod	Logical: If TRUE, save all output to an RDS file in outdir save.mod is TRUE by default if an outdir is defined. If set to TRUE, and no outdir is defined, outdir defaults to paste0("./s.", mod.name)
...	Additional arguments

Value

rtMod object

Author(s)

E.D. Gennatas

See Also

elevate

Other Supervised Learning: s_ADABOOST(), s_ADDTREE(), s_BART(), s_BAYESGLM(), s_BRUTO(), s_C50(), s_CART(), s_ET(), s_EVTREE(), s_GAM.default(), s_GAM.formula(), s_GAMSELX2(), s_GAMSELX(), s_GAMSEL(), s_GAM(), s_GBM0(), s_GBM3.R(), s_GBM(), s_GLMNET(), s_GLMTREE(), s_GLM(), s_GLS(), s_H2ODL(), s_H2OGBM(), s_H2ORF(), s_IRF(), s_KNN(), s_LDA(), s_LMTREE(), s_LM(), s_MARS(), s_MLRF(), s_NBAYES(), s_NLA(), s_NLS(), s_NW(), s_POLYMARS(), s_PPR(), s_PPTREE(), s_QDA(), s_QRNN(), s_RANGER(), s_RFSRC(), s_RF(), s_SDA(), s_SGD(), s_SPLS(), s_SVM(), s_TFN(), s_XGBLIN(), s_XGBOOST(), s_XGB()

Other Tree-based methods: s_ADABOOST(), s_ADDTREE(), s_BART(), s_C50(), s_CART(), s_ET(), s_EVTREE(), s_GBM0(), s_GBM3.R(), s_GBM(), s_GLMTREE(), s_H2OGBM(), s_H2ORF(), s_IRF(), s_LMTREE(), s_MLRF(), s_PPTREE(), s_RANGER(), s_RFSRC(), s_RF(), s_XGBOOST(), s_XGB()

---

s_DN *Artificial Neural Network [C, R]*

Description

Train a deep net for Regression or Classification using **deepnet**

Usage

```
s_DN(
  x,
  y = NULL,
  x.test = NULL,
  y.test = NULL,
  weights = NULL,
  ipw = TRUE,
  ipw.type = 2,
  upsample = FALSE,
  downsample = FALSE,
  resample.seed = NULL,
```

```
  initW = NULL,
  initB = NULL,
  n.hidden.nodes = 10,
  activation = NULL,
  learning.rate = 0.8,
  momentum = 0.5,
  learningrate_scale = 1,
  output = NULL,
  numepochs = 200,
  batchsize = NULL,
  hidden_dropout = 0,
  visible_dropout = 0,
  metric = NULL,
  maximize = NULL,
  grid.resample.rtset = rtset.grid.resample(),
  .preprocess = NULL,
  verbose = TRUE,
  verbose.predict = FALSE,
  trace = 0,
  n.cores = rtCores,
  x.name = NULL,
  y.name = NULL,
  question = NULL,
  outdir = NULL,
  print.plot = TRUE,
  plot.fitted = NULL,
  plot.predicted = NULL,
  plot.theme = rtTheme,
  save.mod = FALSE
)
```

Arguments

x	Numeric vector or matrix / data frame of features i.e. independent variables
y	Numeric vector of outcome, i.e. dependent variable
x.test	Numeric vector or matrix / data frame of testing set features Columns must correspond to columns in x
y.test	Numeric vector of testing set outcome
weights	Numeric vector: Weights for cases. For classification, weights takes precedence over ipw, therefore set weights = NULL if using ipw. Note: If weight are provided, ipw is not used. Leave NULL if setting ipw = TRUE. Default = NULL
ipw	Logical: If TRUE, apply inverse probability weighting (for Classification only). Note: If weights are provided, ipw is not used. Default = TRUE
ipw.type	Integer 0, 1, 2 1: class.weights as in 0, divided by max(class.weights) 2: class.weights as in 0, divided by min(class.weights) Default = 2
upsample	Logical: If TRUE, upsample cases to balance outcome classes (for Classification only) Note: upsample will randomly sample with replacement if the length of the majority class is more than double the length of the class you are upsampling, thereby introducing randomness
downsample	Logical: If TRUE, downsample majority class to match size of minority class

resample.seed	Integer: If provided, will be used to set the seed during upsampling. Default = NULL (random seed)
n.hidden.nodes	Integer, vector: Length indicated number of hidden layers, value of each element determines number of nodes for given layer
verbose	Logical: If TRUE, print summary to screen.
trace	Integer: If higher than 0, will print more information to the console. Default = 0
x.name	Character: Name for feature set
y.name	Character: Name for outcome
question	Character: the question you are attempting to answer with this model, in plain language.
outdir	Path to output directory. If defined, will save Predicted vs. True plot, if available, as well as full model output, if save.mod is TRUE
print.plot	Logical: if TRUE, produce plot using mplot3 Takes precedence over plot.fitted and plot.predicted. Default = TRUE
plot.fitted	Logical: if TRUE, plot True (y) vs Fitted
plot.predicted	Logical: if TRUE, plot True (y.test) vs Predicted. Requires x.test and y.test
plot.theme	Character: "zero", "dark", "box", "darkbox"
save.mod	Logical: If TRUE, save all output to an RDS file in outdir save.mod is TRUE by default if an outdir is defined. If set to TRUE, and no outdir is defined, outdir defaults to paste0("./s.", mod.name)

Author(s)

E.D. Gennatas

---

s_ET *ExtraTrees [C, R]*

Description

Train an ExtraTrees model and validate

Usage

```
s_ET(
  x,
  y = NULL,
  x.test = NULL,
  y.test = NULL,
  x.name = NULL,
  y.name = NULL,
  n.trees = 500,
  mtry = if (!is.null(y) && !is.factor(y)) max(floor(NCOL(x)/3), 1) else
    floor(sqrt(NCOL(x))),
  nodesize = if (!is.null(y) && !is.factor(y)) 5 else 1,
  weights = NULL,
  ipw = TRUE,
```

```
  upsample = FALSE,
  downsample = FALSE,
  resample.seed = resample.seed,
  n.cores = future::availableCores(),
  print.plot = TRUE,
  plot.fitted = NULL,
  plot.predicted = NULL,
  plot.theme = rtTheme,
  question = NULL,
  rtclass = NULL,
  verbose = TRUE,
  trace = 0,
  outdir = NULL,
  save.mod = ifelse(!is.null(outdir), TRUE, FALSE),
  ...
)
```

Arguments

x	Numeric vector or matrix / data frame of features i.e. independent variables
y	Numeric vector of outcome, i.e. dependent variable
x.test	Numeric vector or matrix / data frame of testing set features Columns must correspond to columns in x
y.test	Numeric vector of testing set outcome
x.name	Character: Name for feature set
y.name	Character: Name for outcome
n.trees	Integer: Number of trees to grow. Default = 1000
mtry	[gS] Integer: Number of features sampled randomly at each split
nodesize	[gS]: Integer: Minimum size of terminal nodes. Default = 5 (Regression); 1 (Classification)
weights	Numeric vector: Weights for cases. For classification, weights takes precedence over ipw, therefore set weights = NULL if using ipw. Note: If weight are provided, ipw is not used. Leave NULL if setting ipw = TRUE. Default = NULL
ipw	Logical: If TRUE, apply inverse probability weighting (for Classification only). Note: If weights are provided, ipw is not used. Default = TRUE
upsample	Logical: If TRUE, upsample cases to balance outcome classes (for Classification only) Note: upsample will randomly sample with replacement if the length of the majority class is more than double the length of the class you are upsampling, thereby introducing randomness
downsample	Logical: If TRUE, downsample majority class to match size of minority class
resample.seed	Integer: If provided, will be used to set the seed during upsampling. Default = NULL (random seed)
n.cores	Integer. N of cores to use
print.plot	Logical: if TRUE, produce plot using mplot3 Takes precedence over plot.fitted and plot.predicted. Default = TRUE
plot.fitted	Logical: if TRUE, plot True (y) vs Fitted
plot.predicted	Logical: if TRUE, plot True (y.test) vs Predicted. Requires x.test and y.test

plot.theme	Character: "zero", "dark", "box", "darkbox"
question	Character: the question you are attempting to answer with this model, in plain language.
rtclass	Character: Class type to use. "S3", "S4", "RC", "R6"
verbose	Logical. Print summary to screen
trace	Integer: If higher than 0, will print more information to the console. Default = 0
outdir	Path to output directory. If defined, will save Predicted vs. True plot, if available, as well as full model output, if save.mod is TRUE
save.mod	Logical: If TRUE, save all output to an RDS file in outdir save.mod is TRUE by default if an outdir is defined. If set to TRUE, and no outdir is defined, outdir defaults to paste0("./s.", mod.name)
...	Additional parameters to be passed to extraTrees

Value

Object of class **rtemis**

Author(s)

E.D. Gennatas

See Also

elevate for external cross-validation

Other Supervised Learning: s_ADABOOST(), s_ADDTREE(), s_BART(), s_BAYESGLM(), s_BRUTO(), s_C50(), s_CART(), s_CTREE(), s_EVTREE(), s_GAM.default(), s_GAM.formula(), s_GAMSELX2(), s_GAMSELX(), s_GAMSEL(), s_GAM(), s_GBM0(), s_GBM3.R(), s_GBM(), s_GLMNET(), s_GLMTREE(), s_GLM(), s_GLS(), s_H2ODL(), s_H2OGBM(), s_H2ORF(), s_IRF(), s_KNN(), s_LDA(), s_LMTREE(), s_LM(), s_MARS(), s_MLRF(), s_NBAYES(), s_NLA(), s_NLS(), s_NW(), s_POLYMARS(), s_PPR(), s_PPTREE(), s_QDA(), s_QRNN(), s_RANGER(), s_RFSRC(), s_RF(), s_SDA(), s_SGD(), s_SPLS(), s_SVM(), s_TFN(), s_XGBLIN(), s_XGBOOST(), s_XGB()

Other Tree-based methods: s_ADABOOST(), s_ADDTREE(), s_BART(), s_C50(), s_CART(), s_CTREE(), s_EVTREE(), s_GBM0(), s_GBM3.R(), s_GBM(), s_GLMTREE(), s_H2OGBM(), s_H2ORF(), s_IRF(), s_LMTREE(), s_MLRF(), s_PPTREE(), s_RANGER(), s_RFSRC(), s_RF(), s_XGBOOST(), s_XGB()

Examples

```
## Not run:
x <- rnorm(100)
y <- .6 * x + 12 + rnorm(100)
mod <- s_ET(x, y)
## End(Not run)
```

s_EVTREE *Evolutionary Learning of Globally Optimal Trees [C, R]*

Description

Train a EVTREE for regression or classification using `evtree`

Usage

```
s_EVTREE(
  x,
  y = NULL,
  x.test = NULL,
  y.test = NULL,
  x.name = NULL,
  y.name = NULL,
  weights = NULL,
  ipw = TRUE,
  ipw.type = 2,
  upsample = FALSE,
  downsample = FALSE,
  resample.seed = NULL,
  control = evtree::evtree.control(),
  na.action = na.exclude,
  print.plot = TRUE,
  plot.fitted = NULL,
  plot.predicted = NULL,
  plot.theme = rtTheme,
  question = NULL,
  rtclass = NULL,
  verbose = TRUE,
  outdir = NULL,
  save.mod = ifelse(!is.null(outdir), TRUE, FALSE),
  ...
)
```

Arguments

`x`	Numeric vector or matrix / data frame of features i.e. independent variables
`y`	Numeric vector of outcome, i.e. dependent variable
`x.test`	Numeric vector or matrix / data frame of testing set features Columns must correspond to columns in `x`
`y.test`	Numeric vector of testing set outcome
`x.name`	Character: Name for feature set
`y.name`	Character: Name for outcome
`weights`	Numeric vector: Weights for cases. For classification, `weights` takes precedence over `ipw`, therefore set `weights = NULL` if using `ipw`. Note: If `weight` are provided, `ipw` is not used. Leave NULL if setting `ipw = TRUE`. Default = NULL

ipw	Logical: If TRUE, apply inverse probability weighting (for Classification only). Note: If weights are provided, ipw is not used. Default = TRUE
ipw.type	Integer 0, 1, 2 1: class.weights as in 0, divided by max(class.weights) 2: class.weights as in 0, divided by min(class.weights) Default = 2
upsample	Logical: If TRUE, upsample cases to balance outcome classes (for Classification only) Note: upsample will randomly sample with replacement if the length of the majority class is more than double the length of the class you are upsampling, thereby introducing randomness
downsample	Logical: If TRUE, downsample majority class to match size of minority class
resample.seed	Integer: If provided, will be used to set the seed during upsampling. Default = NULL (random seed)
control	Passed to evtree::evtree
na.action	How to handle missing values. See ?na.fail
print.plot	Logical: if TRUE, produce plot using mplot3 Takes precedence over plot.fitted and plot.predicted. Default = TRUE
plot.fitted	Logical: if TRUE, plot True (y) vs Fitted
plot.predicted	Logical: if TRUE, plot True (y.test) vs Predicted. Requires x.test and y.test
plot.theme	Character: "zero", "dark", "box", "darkbox"
question	Character: the question you are attempting to answer with this model, in plain language.
rtclass	Character: Class type to use. "S3", "S4", "RC", "R6"
verbose	Logical: If TRUE, print summary to screen.
outdir	Path to output directory. If defined, will save Predicted vs. True plot, if available, as well as full model output, if save.mod is TRUE
save.mod	Logical: If TRUE, save all output to an RDS file in outdir save.mod is TRUE by default if an outdir is defined. If set to TRUE, and no outdir is defined, outdir defaults to paste0("./s.", mod.name)
...	Additional arguments to be passed to evtree::evtree

Value

Object of class rtMod

Author(s)

E.D. Gennatas

See Also

elevate for external cross-validation

Other Supervised Learning: s_ADABOOST(), s_ADDTREE(), s_BART(), s_BAYESGLM(), s_BRUTO(), s_C50(), s_CART(), s_CTREE(), s_ET(), s_GAM.default(), s_GAM.formula(), s_GAMSELX2(), s_GAMSELX(), s_GAMSEL(), s_GAM(), s_GBM0(), s_GBM3.R(), s_GBM(), s_GLMNET(), s_GLMTREE(), s_GLM(), s_GLS(), s_H2ODL(), s_H2OGBM(), s_H2ORF(), s_IRF(), s_KNN(), s_LDA(), s_LMTREE(), s_LM(), s_MARS(), s_MLRF(), s_NBAYES(), s_NLA(), s_NLS(), s_NW(), s_POLYMARS(), s_PPR(), s_PPTREE(), s_QDA(), s_QRNN(), s_RANGER(), s_RFSRC(), s_RF(), s_SDA(), s_SGD(), s_SPLS(), s_SVM(), s_TFN(), s_XGBLIN(), s_XGBOOST(), s_XGB()

Other Tree-based methods: s_ADABOOST(), s_ADDTREE(), s_BART(), s_C50(), s_CART(), s_CTREE(), s_ET(), s_GBM0(), s_GBM3.R(), s_GBM(), s_GLMTREE(), s_H2OGBM(), s_H2ORF(), s_IRF(), s_LMTREE(), s_MLRF(), s_PPTREE(), s_RANGER(), s_RFSRC(), s_RF(), s_XGBOOST(), s_XGB()

---

s_GAM *Generalized Additive Model (GAM) C, R*

Description

Trains a GAM using mgcv::gam and validates it. Input will be used to create a formula of the form:

$$y = s(x_1, k = gam.k) + s(x_2, k = gam.k) + ... + s(x_n, k = gam.k)$$

Usage

```
s_GAM(x, ...)
```

Arguments

x	Numeric vector or matrix / data frame of features i.e. independent variables
...	Additional arguments to be passed to mgcv::gam
covariates	Factors to be included as covariates in model building
covariates.test	Factors to be included as covariates in model validation
k	Integer. Number of bases for smoothing spline

Details

Only s_GAM.default is actively maintained at the moment

Value

rtMod

Author(s)

E.D. Gennatas

See Also

elevate for external cross-validation

Other Supervised Learning: s_ADABOOST(), s_ADDTREE(), s_BART(), s_BAYESGLM(), s_BRUTO(), s_C50(), s_CART(), s_CTREE(), s_ET(), s_EVTREE(), s_GAM.default(), s_GAM.formula(), s_GAMSELX2(), s_GAMSELX(), s_GAMSEL(), s_GBM0(), s_GBM3.R(), s_GBM(), s_GLMNET(), s_GLMTREE(), s_GLM(), s_GLS(), s_H2ODL(), s_H2OGBM(), s_H2ORF(), s_IRF(), s_KNN(), s_LDA(), s_LMTREE(), s_LM(), s_MARS(), s_MLRF(), s_NBAYES(), s_NLA(), s_NLS(), s_NW(), s_POLYMARS(), s_PPR(), s_PPTREE(), s_QDA(), s_QRNN(), s_RANGER(), s_RFSRC(), s_RF(), s_SDA(), s_SGD(), s_SPLS(), s_SVM(), s_TFN(), s_XGBLIN(), s_XGBOOST(), s_XGB()

s_GAM.default *Generalized Additive Model (GAM) [C, R]*

Description

Trains a GAM using `mgcv::gam` and validates it. Input will be used to create a formula of the form:

$$y = s(x_1, k) + s(x_2, k) + ... + s(x_n, k)$$

Usage

```
## Default S3 method:
s_GAM(
  x,
  y = NULL,
  x.test = NULL,
  y.test = NULL,
  x.name = NULL,
  y.name = NULL,
  data = NULL,
  data.test = NULL,
  k = 6,
  family = NULL,
  weights = NULL,
  ipw = TRUE,
  ipw.type = 2,
  upsample = FALSE,
  downsample = FALSE,
  resample.seed = NULL,
  method = "REML",
  select = FALSE,
  removeMissingLevels = TRUE,
  spline.index = NULL,
  verbose = TRUE,
  trace = 0,
  print.plot = TRUE,
  plot.fitted = NULL,
  plot.predicted = NULL,
  plot.theme = rtTheme,
  na.action = na.exclude,
  question = NULL,
  n.cores = rtCores,
  outdir = NULL,
  save.mod = ifelse(!is.null(outdir), TRUE, FALSE),
  ...
)
```

Arguments

x	Numeric vector or matrix / data frame of features i.e. independent variables
y	Numeric vector of outcome, i.e. dependent variable

x.test	Numeric vector or matrix / data frame of testing set features Columns must correspond to columns in x
y.test	Numeric vector of testing set outcome
x.name	Character: Name for feature set
y.name	Character: Name for outcome
k	Integer. Number of bases for smoothing spline
family	Error distribution and link function. See stats::family
weights	Numeric vector: Weights for cases. For classification, weights takes precedence over ipw, therefore set weights = NULL if using ipw. Note: If weight are provided, ipw is not used. Leave NULL if setting ipw = TRUE. Default = NULL
ipw	Logical: If TRUE, apply inverse probability weighting (for Classification only). Note: If weights are provided, ipw is not used. Default = TRUE
ipw.type	Integer 0, 1, 2 1: class.weights as in 0, divided by max(class.weights) 2: class.weights as in 0, divided by min(class.weights) Default = 2
upsample	Logical: If TRUE, upsample cases to balance outcome classes (for Classification only) Note: upsample will randomly sample with replacement if the length of the majority class is more than double the length of the class you are upsampling, thereby introducing randomness
downsample	Logical: If TRUE, downsample majority class to match size of minority class
resample.seed	Integer: If provided, will be used to set the seed during upsampling. Default = NULL (random seed)
removeMissingLevels	Logical: If TRUE, finds factors in x.test that contain levels not present in x and substitutes with NA. This would result in error otherwise and no predictions would be made, ending s_GLM prematurely
verbose	Logical: If TRUE, print summary to screen.
trace	Integer: If higher than 0, will print more information to the console. Default = 0
print.plot	Logical: if TRUE, produce plot using mplot3 Takes precedence over plot.fitted and plot.predicted. Default = TRUE
plot.fitted	Logical: if TRUE, plot True (y) vs Fitted
plot.predicted	Logical: if TRUE, plot True (y.test) vs Predicted. Requires x.test and y.test
plot.theme	Character: "zero", "dark", "box", "darkbox"
na.action	How to handle missing values. See ?na.fail
question	Character: the question you are attempting to answer with this model, in plain language.
outdir	Path to output directory. If defined, will save Predicted vs. True plot, if available, as well as full model output, if save.mod is TRUE
save.mod	Logical: If TRUE, save all output to an RDS file in outdir save.mod is TRUE by default if an outdir is defined. If set to TRUE, and no outdir is defined, outdir defaults to paste0("./s.", mod.name)
...	Additional arguments to be passed to mgcv::gam

Value

rtMod

Author(s)

E.D. Gennatas

See Also

elevate for external cross-validation

Other Supervised Learning: s_ADABOOST(), s_ADDTREE(), s_BART(), s_BAYESGLM(), s_BRUTO(), s_C50(), s_CART(), s_CTREE(), s_ET(), s_EVTREE(), s_GAM.formula(), s_GAMSELX2(), s_GAMSELX(), s_GAMSEL(), s_GAM(), s_GBM0(), s_GBM3.R(), s_GBM(), s_GLMNET(), s_GLMTREE(), s_GLM(), s_GLS(), s_H2ODL(), s_H2OGBM(), s_H2ORF(), s_IRF(), s_KNN(), s_LDA(), s_LMTREE(), s_LM(), s_MARS(), s_MLRF(), s_NBAYES(), s_NLA(), s_NLS(), s_NW(), s_POLYMARS(), s_PPR(), s_PPTREE(), s_QDA(), s_QRNN(), s_RANGER(), s_RFSRC(), s_RF(), s_SDA(), s_SGD(), s_SPLS(), s_SVM(), s_TFN(), s_XGBLIN(), s_XGBOOST(), s_XGB()

---

s_GAM.formula *Generalized Additive Model (GAM) C, R*

Description

Trains a GAM using `mgcv::gam` and validates it. Input will be used to create a formula of the form:

$$y = s(x_1, k = gam.k) + s(x_2, k = gam.k) + ... + s(x_n, k = gam.k)$$

Usage

```
## S3 method for class 'formula'
s_GAM(
  formula,
  data,
  data.test = NULL,
  x.name = NULL,
  y.name = NULL,
  k = 6,
  family = gaussian(),
  weights = NULL,
  method = "REML",
  select = FALSE,
  verbose = TRUE,
  print.plot = TRUE,
  plot.fitted = NULL,
  plot.predicted = NULL,
  plot.theme = rtTheme,
  na.action = na.exclude,
  question = NULL,
  n.cores = rtCores,
  outdir = NULL,
  save.mod = ifelse(!is.null(outdir), TRUE, FALSE),
  ...
)
```

Arguments

x.name	Character: Name for feature set
y.name	Character: Name for outcome
k	Integer. Number of bases for smoothing spline
family	Error distribution and link function. See stats::family
weights	Numeric vector: Weights for cases. For classification, weights takes precedence over ipw, therefore set weights = NULL if using ipw. Note: If weight are provided, ipw is not used. Leave NULL if setting ipw = TRUE. Default = NULL
verbose	Logical: If TRUE, print summary to screen.
print.plot	Logical: if TRUE, produce plot using mplot3 Takes precedence over plot.fitted and plot.predicted. Default = TRUE
plot.fitted	Logical: if TRUE, plot True (y) vs Fitted
plot.predicted	Logical: if TRUE, plot True (y.test) vs Predicted. Requires x.test and y.test
plot.theme	Character: "zero", "dark", "box", "darkbox"
na.action	How to handle missing values. See ?na.fail
question	Character: the question you are attempting to answer with this model, in plain language.
outdir	Path to output directory. If defined, will save Predicted vs. True plot, if available, as well as full model output, if save.mod is TRUE
save.mod	Logical: If TRUE, save all output to an RDS file in outdir save.mod is TRUE by default if an outdir is defined. If set to TRUE, and no outdir is defined, outdir defaults to paste0("./s.", mod.name)
...	Additional arguments to be passed to mgcv::gam
covariates	Factors to be included as covariates in model building
covariates.test	Factors to be included as covariates in model validation

Details

s_GAM.default is the preferred way to train GAMs

Value

rtMod

Author(s)

E.D. Gennatas

See Also

elevate for external cross-validation

Other Supervised Learning: s_ADABOOST(), s_ADDTREE(), s_BART(), s_BAYESGLM(), s_BRUTO(), s_C50(), s_CART(), s_CTREE(), s_ET(), s_EVTREE(), s_GAM.default(), s_GAMSELX2(), s_GAMSELX(), s_GAMSEL(), s_GAM(), s_GBM0(), s_GBM3.R(), s_GBM(), s_GLMNET(), s_GLMTREE(), s_GLM(), s_GLS(), s_H2ODL(), s_H2OGBM(), s_H2ORF(), s_IRF(), s_KNN(), s_LDA(), s_LMTREE(), s_LM(), s_MARS(), s_MLRF(), s_NBAYES(), s_NLA(), s_NLS(), s_NW(), s_POLYMARS(), s_PPR(), s_PPTREE(), s_QDA(), s_QRNN(), s_RANGER(), s_RFSRC(), s_RF(), s_SDA(), s_SGD(), s_SPLS(), s_SVM(), s_TFN(), s_XGBLIN(), s_XGBOOST(), s_XGB()

s_GAMSEL *Regularized Generalized Additive Model (GAMSEL) [C, R]*

Description

Trains a GAMSEL model using `gamsel2::gamsel`.

Usage

```
s_GAMSEL(
  x,
  y = NULL,
  x.test = NULL,
  y.test = NULL,
  x.name = NULL,
  y.name = NULL,
  data = NULL,
  data.test = NULL,
  ipw = TRUE,
  ipw.type = 2,
  upsample = FALSE,
  downsample = FALSE,
  resample.seed = NULL,
  lambda = NULL,
  force.lambda = NULL,
  min.unique.perfeat = 4,
  num.lambda = 50,
  family = NULL,
  degrees = NULL,
  min.degree = 1,
  max.degree = 9,
  gamma = 0.4,
  dfs = NULL,
  min.df = 1,
  max.df = 5,
  n.folds = 10,
  which.lambda = c("lambda.min", "lambda.1se"),
  failsafe = TRUE,
  failsafe.lambda = 0.1,
  tol = 1e-04,
  max.iter = 2000,
  parallel = FALSE,
  verbose = TRUE,
  trace = 0,
  print.plot = TRUE,
  plot.fitted = NULL,
  plot.predicted = NULL,
  plot.theme = rtTheme,
  na.action = na.exclude,
  question = NULL,
  n.cores = rtCores,
```

```
  outdir = NULL,
  save.mod = ifelse(!is.null(outdir), TRUE, FALSE),
  ...
)
```

Arguments

x	Numeric vector or matrix / data frame of features i.e. independent variables
y	Numeric vector of outcome, i.e. dependent variable
x.test	Numeric vector or matrix / data frame of testing set features Columns must correspond to columns in x
y.test	Numeric vector of testing set outcome
x.name	Character: Name for feature set
y.name	Character: Name for outcome
ipw	Logical: If TRUE, apply inverse probability weighting (for Classification only). Note: If weights are provided, ipw is not used. Default = TRUE
ipw.type	Integer 0, 1, 2 1: class.weights as in 0, divided by max(class.weights) 2: class.weights as in 0, divided by min(class.weights) Default = 2
upsample	Logical: If TRUE, upsample cases to balance outcome classes (for Classification only) Note: upsample will randomly sample with replacement if the length of the majority class is more than double the length of the class you are upsampling, thereby introducing randomness
downsample	Logical: If TRUE, downsample majority class to match size of minority class
resample.seed	Integer: If provided, will be used to set the seed during upsampling. Default = NULL (random seed)
lambda	User-supplied lambda sequence. For best performance, leave as NULL and allow the routine to automatically select lambda. Otherwise, supply a (preferably gradually) decreasing sequence.
family	Error distribution and link function. See stats::family
degrees	An integer vector of length nvars specifying the maximum number of spline basis functions to use for each variable.
gamma	Penalty mixing parameter $0 \leq \gamma \leq 1$. Values $\gamma < 0.5$ penalize linear fit less than non-linear fit. The default is $\gamma = 0.4$, which encourages a linear term over a nonlinear term.
dfs	Numeric vector of length nvars specifying the maximum (end-of-path) degrees of freedom for each variable.
tol	Convergence threshold for coordinate descent. The coordinate descent loop continues until the total change in objective after a pass over all variables is less than tol. Default is 1e-4.
parallel	passed on to the pseudo.bases() function. Uses multiple process if available.
verbose	Logical: If TRUE, print summary to screen.
trace	Integer: If higher than 0, will print more information to the console. Default = 0
print.plot	Logical: if TRUE, produce plot using mplot3 Takes precedence over plot.fitted and plot.predicted. Default = TRUE
plot.fitted	Logical: if TRUE, plot True (y) vs Fitted
plot.predicted	Logical: if TRUE, plot True (y.test) vs Predicted. Requires x.test and y.test

plot.theme	Character: "zero", "dark", "box", "darkbox"
na.action	How to handle missing values. See ?na.fail
question	Character: the question you are attempting to answer with this model, in plain language.
outdir	Path to output directory. If defined, will save Predicted vs. True plot, if available, as well as full model output, if save.mod is TRUE
save.mod	Logical: If TRUE, save all output to an RDS file in outdir save.mod is TRUE by default if an outdir is defined. If set to TRUE, and no outdir is defined, outdir defaults to paste0("./s.", mod.name)
...	Additional arguments

Value

rtMod

Author(s)

E.D. Gennatas

See Also

elevate for external cross-validation

Other Supervised Learning: s_ADABOOST(), s_ADDTREE(), s_BART(), s_BAYESGLM(), s_BRUTO(), s_C50(), s_CART(), s_CTREE(), s_ET(), s_EVTREE(), s_GAM.default(), s_GAM.formula(), s_GAMSELX2(), s_GAMSELX(), s_GAM(), s_GBM0(), s_GBM3.R(), s_GBM(), s_GLMNET(), s_GLMTREE(), s_GLM(), s_GLS(), s_H2ODL(), s_H2OGBM(), s_H2ORF(), s_IRF(), s_KNN(), s_LDA(), s_LMTREE(), s_LM(), s_MARS(), s_MLRF(), s_NBAYES(), s_NLA(), s_NLS(), s_NW(), s_POLYMARS(), s_PPR(), s_PPTREE(), s_QDA(), s_QRNN(), s_RANGER(), s_RFSRC(), s_RF(), s_SDA(), s_SGD(), s_SPLS(), s_SVM(), s_TFN(), s_XGBLIN(), s_XGBOOST(), s_XGB()

---

s_GAMSELX *GAMSEL with Interaction Discovery [R]*

Description

Trains a GAMSEL model using gamsel2::gamsel after finding pairwise interactions by mass GLM.

Usage

```
s_GAMSELX(
  x,
  y = NULL,
  x.test = NULL,
  y.test = NULL,
  x.name = NULL,
  y.name = NULL,
  data = NULL,
  data.test = NULL,
  gamsel.params1 = list(),
```

```
  pairs.on.resid = TRUE,
  p.adjust.method = "holm",
  alpha = 0.05,
  gamsel.params2 = gamsel.params1,
  verbose = TRUE,
  trace = 0,
  print.plot = TRUE,
  plot.fitted = NULL,
  plot.predicted = NULL,
  plot.theme = rtTheme,
  na.action = na.exclude,
  question = NULL,
  n.cores = 1,
  outdir = NULL,
  save.mod = ifelse(!is.null(outdir), TRUE, FALSE),
  ...
)
```

Arguments

x	Numeric vector or matrix / data frame of features i.e. independent variables
y	Numeric vector of outcome, i.e. dependent variable
x.test	Numeric vector or matrix / data frame of testing set features Columns must correspond to columns in x
y.test	Numeric vector of testing set outcome
x.name	Character: Name for feature set
y.name	Character: Name for outcome
gamsel.params1	List of parameters to pass to s_GAMSEL for first model (linear and nonlinear main effects)
pairs.on.resid	Logical: If TRUE, train second gamsel on residuals of first. Default = TRUE. Should be kept TRUE, option available for experimentation/demonstration, etc.
p.adjust.method	Character: Method to use for multiple comparison correction after mass GLM. Default = "holm"
alpha	Float: significance level. Default = .05
gamsel.params2	List of parameters to pass to s_GAMSEL for final model
verbose	Logical: If TRUE, print summary to screen.
trace	Integer: If higher than 0, will print more information to the console. Default = 0
print.plot	Logical: if TRUE, produce plot using mplot3 Takes precedence over plot.fitted and plot.predicted. Default = TRUE
plot.fitted	Logical: if TRUE, plot True (y) vs Fitted
plot.predicted	Logical: if TRUE, plot True (y.test) vs Predicted. Requires x.test and y.test
plot.theme	Character: "zero", "dark", "box", "darkbox"
na.action	How to handle missing values. See ?na.fail
question	Character: the question you are attempting to answer with this model, in plain language.

outdir	Path to output directory. If defined, will save Predicted vs. True plot, if available, as well as full model output, if save.mod is TRUE
save.mod	Logical: If TRUE, save all output to an RDS file in outdir save.mod is TRUE by default if an outdir is defined. If set to TRUE, and no outdir is defined, outdir defaults to paste0("./s.", mod.name)
...	Additional arguments

Value

rtMod

Author(s)

E.D. Gennatas

See Also

elevate for external cross-validation

Other Supervised Learning: s_ADABOOST(), s_ADDTREE(), s_BART(), s_BAYESGLM(), s_BRUTO(), s_C50(), s_CART(), s_CTREE(), s_ET(), s_EVTREE(), s_GAM.default(), s_GAM.formula(), s_GAMSELX2(), s_GAMSEL(), s_GAM(), s_GBM0(), s_GBM3.R(), s_GBM(), s_GLMNET(), s_GLMTREE(), s_GLM(), s_GLS(), s_H2ODL(), s_H2OGBM(), s_H2ORF(), s_IRF(), s_KNN(), s_LDA(), s_LMTREE(), s_LM(), s_MARS(), s_MLRF(), s_NBAYES(), s_NLA(), s_NLS(), s_NW(), s_POLYMARS(), s_PPR(), s_PPTREE(), s_QDA(), s_QRNN(), s_RANGER(), s_RFSRC(), s_RF(), s_SDA(), s_SGD(), s_SPLS(), s_SVM(), s_TFN(), s_XGBLIN(), s_XGBOOST(), s_XGB()

---

s_GAMSELX2 *GAMSEL with Interaction Discovery [R]*

Description

Trains a GAMSEL model using gamsel2::gamsel after finding pairwise interactions by mass GLM.

Usage

```
s_GAMSELX2(
  x,
  y = NULL,
  x.test = NULL,
  y.test = NULL,
  x.name = NULL,
  y.name = NULL,
  data = NULL,
  data.test = NULL,
  autopreprocess = TRUE,
  min.unique.perfeat = 9,
  cart.params = list(maxdepth = 4, cp = 0.1),
  gamsel.params1 = list(),
  pairs.on.resid = TRUE,
  p.adjust.method = "holm",
```

```
  alpha = 0.05,
  gamsel.params2 = gamsel.params1,
  verbose = TRUE,
  trace = 0,
  print.plot = TRUE,
  plot.fitted = NULL,
  plot.predicted = NULL,
  plot.theme = rtTheme,
  na.action = na.exclude,
  question = NULL,
  n.cores = 1,
  outdir = NULL,
  save.mod = ifelse(!is.null(outdir), TRUE, FALSE),
  ...
)
```

Arguments

x	Numeric vector or matrix / data frame of features i.e. independent variables
y	Numeric vector of outcome, i.e. dependent variable
x.test	Numeric vector or matrix / data frame of testing set features Columns must correspond to columns in x
y.test	Numeric vector of testing set outcome
x.name	Character: Name for feature set
y.name	Character: Name for outcome
gamsel.params1	List of parameters to pass to s_GAMSEL for first model (linear and nonlinear main effects)
pairs.on.resid	Logical: If TRUE, train second gamsel on residuals of first. Default = TRUE. Should be kept TRUE, option available for experimentation/demonstration, etc.
p.adjust.method	Character: Method to use for multiple comparison correction after mass GLM. Default = "holm"
alpha	Float: significance level. Default = .05
gamsel.params2	List of parameters to pass to s_GAMSEL for final model
verbose	Logical: If TRUE, print summary to screen.
trace	Integer: If higher than 0, will print more information to the console. Default = 0
print.plot	Logical: if TRUE, produce plot using mplot3 Takes precedence over plot.fitted and plot.predicted. Default = TRUE
plot.fitted	Logical: if TRUE, plot True (y) vs Fitted
plot.predicted	Logical: if TRUE, plot True (y.test) vs Predicted. Requires x.test and y.test
plot.theme	Character: "zero", "dark", "box", "darkbox"
na.action	How to handle missing values. See ?na.fail
question	Character: the question you are attempting to answer with this model, in plain language.
outdir	Path to output directory. If defined, will save Predicted vs. True plot, if available, as well as full model output, if save.mod is TRUE

save.mod	Logical: If TRUE, save all output to an RDS file in outdir save.mod is TRUE by default if an outdir is defined. If set to TRUE, and no outdir is defined, outdir defaults to paste0("./s.", mod.name)
...	Additional arguments

Value

rtMod

Author(s)

E.D. Gennatas

See Also

elevate for external cross-validation

Other Supervised Learning: s_ADABOOST(), s_ADDTREE(), s_BART(), s_BAYESGLM(), s_BRUTO(), s_C50(), s_CART(), s_CTREE(), s_ET(), s_EVTREE(), s_GAM.default(), s_GAM.formula(), s_GAMSELX(), s_GAMSEL(), s_GAM(), s_GBM0(), s_GBM3.R(), s_GBM(), s_GLMNET(), s_GLMTREE(), s_GLM(), s_GLS(), s_H2ODL(), s_H2OGBM(), s_H2ORF(), s_IRF(), s_KNN(), s_LDA(), s_LMTREE(), s_LM(), s_MARS(), s_MLRF(), s_NBAYES(), s_NLA(), s_NLS(), s_NW(), s_POLYMARS(), s_PPR(), s_PPTREE(), s_QDA(), s_QRNN(), s_RANGER(), s_RFSRC(), s_RF(), s_SDA(), s_SGD(), s_SPLS(), s_SVM(), s_TFN(), s_XGBLIN(), s_XGBOOST(), s_XGB()

---

s_GBM *Gradient Boosting Machine [C, R, S]*

Description

Train a GBM model using gbm::gbm.fit

Usage

```
s_GBM(
  x,
  y = NULL,
  x.test = NULL,
  y.test = NULL,
  weights = NULL,
  ipw = TRUE,
  ipw.type = 2,
  upsample = FALSE,
  downsample = FALSE,
  resample.seed = NULL,
  distribution = NULL,
  interaction.depth = 2,
  shrinkage = 0.01,
  bag.fraction = 0.9,
  n.minobsinnode = 5,
  n.trees = 2000,
  max.trees = 5000,
```

```
  force.n.trees = NULL,
  n.tree.window = 0,
  gbm.select.smooth = TRUE,
  n.new.trees = 500,
  min.trees = 50,
  failsafe.trees = 1000,
  imetrics = FALSE,
  .gs = FALSE,
  grid.resample.rtset = rtset.resample("kfold", 5),
  grid.search.type = "exhaustive",
  metric = NULL,
  maximize = NULL,
  plot.tune.error = FALSE,
  exclude.test.lt.train = FALSE,
  exclude.lt.min.trees = FALSE,
  res.fail.thres = 0.99,
  n.extra.trees = 0,
  n.cores = rtCores,
  relInf = TRUE,
  varImp = FALSE,
  offset = NULL,
  misc = NULL,
  var.monotone = NULL,
  keep.data = TRUE,
  var.names = NULL,
  response.name = "y",
  checkmods = FALSE,
  group = NULL,
  plot.perf = FALSE,
  plot.res = ifelse(!is.null(outdir), TRUE, FALSE),
  plot.fitted = NULL,
  plot.predicted = NULL,
  plotRelInf = FALSE,
  plotVarImp = FALSE,
  print.plot = TRUE,
  plot.theme = rtTheme,
  x.name = NULL,
  y.name = NULL,
  question = NULL,
  verbose = TRUE,
  trace = 0,
  grid.verbose = verbose,
  gbm.fit.verbose = FALSE,
  outdir = NULL,
  save.gridrun = FALSE,
  save.rds = TRUE,
  save.res = FALSE,
  save.res.mod = FALSE,
  save.mod = ifelse(!is.null(outdir), TRUE, FALSE)
)
```

Arguments

x	Numeric vector or matrix / data frame of features i.e. independent variables
y	Numeric vector of outcome, i.e. dependent variable
x.test	Numeric vector or matrix / data frame of testing set features Columns must correspond to columns in x
y.test	Numeric vector of testing set outcome
weights	Numeric vector: Weights for cases. For classification, weights takes precedence over ipw, therefore set weights = NULL if using ipw. Note: If weight are provided, ipw is not used. Leave NULL if setting ipw = TRUE. Default = NULL
ipw	Logical: If TRUE, apply inverse probability weighting (for Classification only). Note: If weights are provided, ipw is not used. Default = TRUE
ipw.type	Integer 0, 1, 2 1: class.weights as in 0, divided by max(class.weights) 2: class.weights as in 0, divided by min(class.weights) Default = 2
upsample	Logical: If TRUE, upsample cases to balance outcome classes (for Classification only) Note: upsample will randomly sample with replacement if the length of the majority class is more than double the length of the class you are upsampling, thereby introducing randomness
downsample	Logical: If TRUE, downsample majority class to match size of minority class
resample.seed	Integer: If provided, will be used to set the seed during upsampling. Default = NULL (random seed)
interaction.depth	[gS] Integer: Interaction depth. Default = 2
shrinkage	[gS] Float: Shrinkage (learning rate). Default = .01
bag.fraction	[gS] Float (0, 1): Fraction of cases to use to train each tree. Helps avoid overfitting. Default = .75
n.minobsinnode	[gS] Integer: Minimum number of observation allowed in node. Default = 5
n.trees	Integer: Initial number of trees to fit
relInf	Logical: If TRUE (Default), estimate variables' relative influence.
varImp	Logical: If TRUE, estimate variable importance by permutation (as in random forests; noted as experimental in gbm). Takes longer than (default) relative influence. The two measures are highly correlated.
plot.fitted	Logical: if TRUE, plot True (y) vs Fitted
plot.predicted	Logical: if TRUE, plot True (y.test) vs Predicted. Requires x.test and y.test
print.plot	Logical: if TRUE, produce plot using mplot3 Takes precedence over plot.fitted and plot.predicted. Default = TRUE
plot.theme	Character: "zero", "dark", "box", "darkbox"
x.name	Character: Name for feature set
y.name	Character: Name for outcome
question	Character: the question you are attempting to answer with this model, in plain language.
verbose	Logical: If TRUE, print summary to screen.
trace	Integer: If higher than 0, will print more information to the console. Default = 0
outdir	Character: If defined, save log, 'plot.all' plots (see above) and RDS file of complete output

save.rds	Logical: If outdir is defined, should all data be saved in RDS file? s.SVDnetGBM will save mod.gbm, so no need to save again.
save.res.mod	Logical: If TRUE, save gbm model for each grid run. For diagnostic purposes only: Object size adds up quickly
save.mod	Logical: If TRUE, save all output to an RDS file in outdir save.mod is TRUE by default if an outdir is defined. If set to TRUE, and no outdir is defined, outdir defaults to paste0("./s.", mod.name)
stratify.var	If resampling is stratified, stratify against this variable. Defaults to outcome

Details

This is the older gbm package available on CRAN. It may be preferable to use s_GBM3 which uses gbm-developers/gbm3 from GitHub. Early stopping is implemented by fitting n.trees initially, checking the (smoothed) validation error curve, and adding n.new.trees if needed, until error does not reduce or max.trees is reached. [gS] in the argument description indicates that multiple values can be passed, in which case tuning will be performed using grid search. gS is supported for: interaction.depth, shrinkage, bag.fraction, and n.minobsinnode This function includes a workaround for when gbm.fit fails. If an error is detected, gbm.fit is rerun until successful and the procedure continues normally

Author(s)

E.D. Gennatas

See Also

elevate for external cross-validation

Other Supervised Learning: s_ADABOOST(), s_ADDTREE(), s_BART(), s_BAYESGLM(), s_BRUTO(), s_C50(), s_CART(), s_CTREE(), s_ET(), s_EVTREE(), s_GAM.default(), s_GAM.formula(), s_GAMSELX2(), s_GAMSELX(), s_GAMSEL(), s_GAM(), s_GBM0(), s_GBM3.R(), s_GLMNET(), s_GLMTREE(), s_GLM(), s_GLS(), s_H2ODL(), s_H2OGBM(), s_H2ORF(), s_IRF(), s_KNN(), s_LDA(), s_LMTREE(), s_LM(), s_MARS(), s_MLRF(), s_NBAYES(), s_NLA(), s_NLS(), s_NW(), s_POLYMARS(), s_PPR(), s_PPTREE(), s_QDA(), s_QRNN(), s_RANGER(), s_RFSRC(), s_RF(), s_SDA(), s_SGD(), s_SPLS(), s_SVM(), s_TFN(), s_XGBLIN(), s_XGBOOST(), s_XGB()

Other Tree-based methods: s_ADABOOST(), s_ADDTREE(), s_BART(), s_C50(), s_CART(), s_CTREE(), s_ET(), s_EVTREE(), s_GBM0(), s_GBM3.R(), s_GLMTREE(), s_H2OGBM(), s_H2ORF(), s_IRF(), s_LMTREE(), s_MLRF(), s_PPTREE(), s_RANGER(), s_RFSRC(), s_RF(), s_XGBOOST(), s_XGB()

Other Ensembles: s_ADABOOST(), s_GBM0(), s_GBM3.R(), s_RANGER(), s_RF()

---

s_GBM0 *Gradient Boosting Machine [C, R, S]*

Description

Train a GBM model using gbm::gbm.fit

Usage

```
s_GBM0(
  x,
  y = NULL,
  x.test = NULL,
  y.test = NULL,
  weights = NULL,
  ipw = TRUE,
  ipw.type = 2,
  upsample = FALSE,
  downsample = FALSE,
  resample.seed = NULL,
  distribution = NULL,
  interaction.depth = 2,
  shrinkage = 0.01,
  bag.fraction = 0.9,
  n.minobsinnode = 5,
  n.trees = 2000,
  max.trees = 5000,
  force.n.trees = NULL,
  n.tree.window = 0,
  gbm.select.smooth = TRUE,
  n.new.trees = 500,
  min.trees = 50,
  failsafe.trees = 1000,
  imetrics = FALSE,
  .gs = FALSE,
  grid.resample.rtset = rtset.resample("kfold", 5),
  grid.search.type = "exhaustive",
  metric = NULL,
  maximize = NULL,
  plot.tune.error = FALSE,
  exclude.test.lt.train = FALSE,
  exclude.lt.min.trees = FALSE,
  res.fail.thres = 0.99,
  n.extra.trees = 0,
  n.cores = rtCores,
  relInf = TRUE,
  varImp = FALSE,
  offset = NULL,
  misc = NULL,
  var.monotone = NULL,
  keep.data = TRUE,
  var.names = NULL,
  response.name = "y",
  group = NULL,
  plot.perf = FALSE,
  plot.res = ifelse(!is.null(outdir), TRUE, FALSE),
  plot.fitted = NULL,
  plot.predicted = NULL,
  plotRelInf = FALSE,
  plotVarImp = FALSE,
```

```
  print.plot = TRUE,
  plot.theme = rtTheme,
  x.name = NULL,
  y.name = NULL,
  question = NULL,
  verbose = TRUE,
  trace = 0,
  grid.verbose = verbose,
  gbm.fit.verbose = FALSE,
  outdir = NULL,
  save.gridrun = FALSE,
  save.rds = TRUE,
  save.res = FALSE,
  save.res.mod = FALSE,
  save.mod = ifelse(!is.null(outdir), TRUE, FALSE),
  ...
)
```

Arguments

x	Numeric vector or matrix / data frame of features i.e. independent variables
y	Numeric vector of outcome, i.e. dependent variable
x.test	Numeric vector or matrix / data frame of testing set features Columns must correspond to columns in x
y.test	Numeric vector of testing set outcome
weights	Numeric vector: Weights for cases. For classification, weights takes precedence over ipw, therefore set weights = NULL if using ipw. Note: If weight are provided, ipw is not used. Leave NULL if setting ipw = TRUE. Default = NULL
ipw	Logical: If TRUE, apply inverse probability weighting (for Classification only). Note: If weights are provided, ipw is not used. Default = TRUE
ipw.type	Integer 0, 1, 2 1: class.weights as in 0, divided by max(class.weights) 2: class.weights as in 0, divided by min(class.weights) Default = 2
upsample	Logical: If TRUE, upsample cases to balance outcome classes (for Classification only) Note: upsample will randomly sample with replacement if the length of the majority class is more than double the length of the class you are upsampling, thereby introducing randomness
downsample	Logical: If TRUE, downsample majority class to match size of minority class
resample.seed	Integer: If provided, will be used to set the seed during upsampling. Default = NULL (random seed)
interaction.depth	[gS] Integer: Interaction depth. Default = 2
shrinkage	[gS] Float: Shrinkage (learning rate). Default = .01
bag.fraction	[gS] Float (0, 1): Fraction of cases to use to train each tree. Helps avoid overfitting. Default = .75
n.minobsinnode	[gS] Integer: Minimum number of observation allowed in node. Default = 5
n.trees	Integer: Initial number of trees to fit
relInf	Logical: If TRUE (Default), estimate variables' relative influence.

varImp	Logical: If TRUE, estimate variable importance by permutation (as in random forests; noted as experimental in gbm). Takes longer than (default) relative influence. The two measures are highly correlated.
plot.fitted	Logical: if TRUE, plot True (y) vs Fitted
plot.predicted	Logical: if TRUE, plot True (y.test) vs Predicted. Requires x.test and y.test
print.plot	Logical: if TRUE, produce plot using mplot3 Takes precedence over plot.fitted and plot.predicted. Default = TRUE
plot.theme	Character: "zero", "dark", "box", "darkbox"
x.name	Character: Name for feature set
y.name	Character: Name for outcome
question	Character: the question you are attempting to answer with this model, in plain language.
verbose	Logical: If TRUE, print summary to screen.
trace	Integer: If higher than 0, will print more information to the console. Default = 0
outdir	Character: If defined, save log, 'plot.all' plots (see above) and RDS file of complete output
save.rds	Logical: If outdir is defined, should all data be saved in RDS file? s.SVDnetGBM will save mod.gbm, so no need to save again.
save.res.mod	Logical: If TRUE, save gbm model for each grid run. For diagnostic purposes only: Object size adds up quickly
save.mod	Logical: If TRUE, save all output to an RDS file in outdir save.mod is TRUE by default if an outdir is defined. If set to TRUE, and no outdir is defined, outdir defaults to paste0("./s.", mod.name)
...	Additional arguments
stratify.var	If resampling is stratified, stratify against this variable. Defaults to outcome

Details

This is the older gbm package available on CRAN. It may be preferable to use s_GBM3 which uses gbm-developers/gbm3 from GitHub. Early stopping is implemented by fitting n.trees initially, checking the (smoothed) validation error curve, and adding n.new.trees if needed, until error does not reduce or max.trees is reached. [gS] in the argument description indicates that multiple values can be passed, in which case tuning will be performed using grid search. gS is supported for: interaction.depth, shrinkage, bag.fraction, and n.minobsinnode This function includes a workaround for when gbm.fit fails. If an error is detected, gbm.fit is rerun until successful and the procedure continues normally

Author(s)

E.D. Gennatas

See Also

elevate for external cross-validation

Other Supervised Learning: s_ADABOOST(), s_ADDTREE(), s_BART(), s_BAYESGLM(), s_BRUTO(), s_C50(), s_CART(), s_CTREE(), s_ET(), s_EVTREE(), s_GAM.default(), s_GAM.formula(), s_GAMSELX2(), s_GAMSELX(), s_GAMSEL(), s_GAM(), s_GBM3.R(), s_GBM(), s_GLMNET(), s_GLMTREE(), s_GLM(), s_GLS(), s_H2ODL(), s_H2OGBM(), s_H2ORF(), s_IRF(), s_KNN(), s_LDA(), s_LMTREE(),

s_LM(), s_MARS(), s_MLRF(), s_NBAYES(), s_NLA(), s_NLS(), s_NW(), s_POLYMARS(), s_PPR(), s_PPTREE(), s_QDA(), s_QRNN(), s_RANGER(), s_RFSRC(), s_RF(), s_SDA(), s_SGD(), s_SPLS(), s_SVM(), s_TFN(), s_XGBLIN(), s_XGBOOST(), s_XGB()

Other Tree-based methods: s_ADABOOST(), s_ADDTREE(), s_BART(), s_C50(), s_CART(), s_CTREE(), s_ET(), s_EVTREE(), s_GBM3.R(), s_GBM(), s_GLMTREE(), s_H2OGBM(), s_H2ORF(), s_IRF(), s_LMTREE(), s_MLRF(), s_PPTREE(), s_RANGER(), s_RFSRC(), s_RF(), s_XGBOOST(), s_XGB()

Other Ensembles: s_ADABOOST(), s_GBM3.R(), s_GBM(), s_RANGER(), s_RF()

---

s_GBM3.R *Gradient Boosting Machine [C, R, S]*

Description

Train a GBM model using gbm-developers/gbm3

Usage

```
s_GBM3.R(
  x,
  y = NULL,
  x.test = NULL,
  y.test = NULL,
  weights = NULL,
  ipw = TRUE,
  ipw.type = 2,
  upsample = FALSE,
  downsample = FALSE,
  resample.seed = NULL,
  distribution = NULL,
  interaction.depth = 3,
  shrinkage = 0.01,
  bag.fraction = 0.9,
  mFeatures = NULL,
  n.minobsinnode = 5,
  n.trees = 2000,
  max.trees = 5000,
  force.n.trees = NULL,
  n.tree.window = 0,
  gbm.select.smooth = TRUE,
  smoother = c("loess", "supsmu"),
  n.new.trees = 500,
  min.trees = 50,
  failsafe.trees = 1000,
  imetrics = FALSE,
  .gs = FALSE,
  grid.resample.rtset = rtset.resample("kfold", 5),
  grid.search.type = c("exhaustive", "randomized"),
  grid.randomized.p = 0.1,
  metric = NULL,
  maximize = NULL,
```

```
  plot.tune.error = FALSE,
  exclude.test.lt.train = FALSE,
  exclude.lt.min.trees = FALSE,
  res.fail.thres = 0.99,
  n.extra.trees = 0,
  n.cores = rtCores,
  gbm.cores = 1,
  relInf = TRUE,
  varImp = FALSE,
  offset = NULL,
  var.monotone = NULL,
  keep.data = TRUE,
  var.names = NULL,
  response.name = "y",
  group = NULL,
  plot.perf = FALSE,
  plot.res = ifelse(!is.null(outdir), TRUE, FALSE),
  plot.fitted = NULL,
  plot.predicted = NULL,
  plotRelInf = FALSE,
  plotVarImp = FALSE,
  print.plot = TRUE,
  plot.theme = rtTheme,
  x.name = NULL,
  y.name = NULL,
  question = NULL,
  verbose = TRUE,
  trace = 0,
  grid.verbose = verbose,
  gbm.fit.verbose = FALSE,
  outdir = NULL,
  save.gridrun = FALSE,
  save.error.diagnostics = FALSE,
  save.rds = TRUE,
  save.res = FALSE,
  save.res.mod = FALSE,
  save.mod = ifelse(!is.null(outdir), TRUE, FALSE),
  ...
)
```

Arguments

x	Numeric vector or matrix / data frame of features i.e. independent variables
y	Numeric vector of outcome, i.e. dependent variable
x.test	Numeric vector or matrix / data frame of testing set features Columns must correspond to columns in x
y.test	Numeric vector of testing set outcome
weights	Numeric vector: Weights for cases. For classification, weights takes precedence over ipw, therefore set weights = NULL if using ipw. Note: If weight are provided, ipw is not used. Leave NULL if setting ipw = TRUE. Default = NULL
ipw	Logical: If TRUE, apply inverse probability weighting (for Classification only). Note: If weights are provided, ipw is not used. Default = TRUE

`ipw.type` Integer 0, 1, 2 1: class.weights as in 0, divided by max(class.weights) 2: class.weights as in 0, divided by min(class.weights) Default = 2

`upsample` Logical: If TRUE, upsample cases to balance outcome classes (for Classification only) Note: upsample will randomly sample with replacement if the length of the majority class is more than double the length of the class you are upsampling, thereby introducing randomness

`downsample` Logical: If TRUE, downsample majority class to match size of minority class

`resample.seed` Integer: If provided, will be used to set the seed during upsampling. Default = NULL (random seed)

`interaction.depth` [gS] Integer: Interaction depth. Default = 3

`shrinkage` [gS] Float: Shrinkage (learning rate). Default = .01

`bag.fraction` [gS] Float (0, 1): Fraction of cases to use to train each tree. Helps avoid overfitting. Default = .9

`mFeatures` [gS] Integer: Number of features to randomly choose from all available features to train at each step. Default = NULL which results in using all features.

`n.minobsinnode` [gS] Integer: Minimum number of observation allowed in node

`n.trees` Integer: Initial number of trees to fit. Default = 2000

`max.trees` Integer: Maximum number of trees to fit. Default = 5000

`grid.resample.rtset` List: Output of rtset.resample defining gridSearchLearn parameters.

`grid.search.type` Character: Type of grid search to perform: "exhaustive" or "randomized".

`grid.randomized.p` Float (0, 1): If `grid.search.type` = `"randomized"`, randomly test this proportion of combinations.

`metric` Character: Metric to minimize, or maximize if `maximize` = `TRUE` during grid search. Default = NULL, which results in "Balanced Accuracy" for Classification, "MSE" for Regression, and "Coherence" for Survival Analysis.

`maximize` Logical: If TRUE, `metric` will be maximized if grid search is run.

`n.cores` Integer: Number of cores to use.

`relInf` Logical: If TRUE (Default), estimate variables' relative influence.

`varImp` Logical: If TRUE, estimate variable importance by permutation (as in random forests; noted as experimental in gbm). Takes longer than (default) relative influence. The two measures are highly correlated.

`plot.fitted` Logical: if TRUE, plot True (y) vs Fitted

`plot.predicted` Logical: if TRUE, plot True (y.test) vs Predicted. Requires `x.test` and `y.test`

`print.plot` Logical: if TRUE, produce plot using `mplot3` Takes precedence over `plot.fitted` and `plot.predicted`. Default = TRUE

`plot.theme` Character: "zero", "dark", "box", "darkbox"

`x.name` Character: Name for feature set

`y.name` Character: Name for outcome

`question` Character: the question you are attempting to answer with this model, in plain language.

`verbose` Logical: If TRUE, print summary to screen.

trace	Integer: If higher than 0, will print more information to the console. Default = 0
grid.verbose	Logical: Passed to gridSearchLearn
outdir	Character: If defined, save log, 'plot.all' plots (see above) and RDS file of complete output
save.rds	Logical: If outdir is defined, should all data be saved in RDS file? s.SVDnetGBM will save mod.gbm, so no need to save again.
save.res.mod	Logical: If TRUE, save gbm model for each grid run. For diagnostic purposes only: Object size adds up quickly
save.mod	Logical: If TRUE, save all output to an RDS file in outdir save.mod is TRUE by default if an outdir is defined. If set to TRUE, and no outdir is defined, outdir defaults to paste0("./s.", mod.name)
...	Additional arguments
stratify.var	If resampling is stratified, stratify against this variable. Defaults to outcome

Details

Early stopping is implemented by fitting n.trees initially, checking the (smoothed) validation error curve, and adding n.new.trees if needed, until error does not reduce or max.trees is reached. [gS] in the argument description indicates that multiple values can be passed, in which case tuning will be performed using grid search. gS is supported for: interaction.depth, shrinkage, bag.fraction, mFeatures, and n.minobsinnode This function includes a workaround for when gbm.fit fails. If an error is detected, gbm.fit is rerun until successful and the procedure continues normally

Author(s)

E.D. Gennatas

See Also

elevate for external cross-validation

Other Supervised Learning: s_ADABOOST(), s_ADDTREE(), s_BART(), s_BAYESGLM(), s_BRUTO(), s_C50(), s_CART(), s_CTREE(), s_ET(), s_EVTREE(), s_GAM.default(), s_GAM.formula(), s_GAMSELX2(), s_GAMSELX(), s_GAMSEL(), s_GAM(), s_GBM0(), s_GBM(), s_GLMNET(), s_GLMTREE(), s_GLM(), s_GLS(), s_H2ODL(), s_H2OGBM(), s_H2ORF(), s_IRF(), s_KNN(), s_LDA(), s_LMTREE(), s_LM(), s_MARS(), s_MLRF(), s_NBAYES(), s_NLA(), s_NLS(), s_NW(), s_POLYMARS(), s_PPR(), s_PPTREE(), s_QDA(), s_QRNN(), s_RANGER(), s_RFSRC(), s_RF(), s_SDA(), s_SGD(), s_SPLS(), s_SVM(), s_TFN(), s_XGBLIN(), s_XGBOOST(), s_XGB()

Other Tree-based methods: s_ADABOOST(), s_ADDTREE(), s_BART(), s_C50(), s_CART(), s_CTREE(), s_ET(), s_EVTREE(), s_GBM0(), s_GBM(), s_GLMTREE(), s_H2OGBM(), s_H2ORF(), s_IRF(), s_LMTREE(), s_MLRF(), s_PPTREE(), s_RANGER(), s_RFSRC(), s_RF(), s_XGBOOST(), s_XGB()

Other Ensembles: s_ADABOOST(), s_GBM0(), s_GBM(), s_RANGER(), s_RF()

s_GLM *Generalized Linear Model [C, R]*

Description

Train a Generalized Linear Model for Regression or Classification (i.e. Logistic Regression) using `stats::glm`. If outcome `y` has more than two classes, Multinomial Logistic Regression is performed using `nnet::multinom`

Usage

```
s_GLM(
  x,
  y = NULL,
  x.test = NULL,
  y.test = NULL,
  x.name = NULL,
  y.name = NULL,
  family = NULL,
  interactions = NULL,
  class.method = NULL,
  weights = NULL,
  ipw = TRUE,
  ipw.type = 2,
  upsample = FALSE,
  downsample = FALSE,
  resample.seed = NULL,
  intercept = TRUE,
  polynomial = FALSE,
  poly.d = 3,
  poly.raw = FALSE,
  print.plot = TRUE,
  plot.fitted = NULL,
  plot.predicted = NULL,
  plot.theme = rtTheme,
  na.action = na.exclude,
  removeMissingLevels = TRUE,
  question = NULL,
  rtclass = NULL,
  verbose = TRUE,
  trace = 0,
  outdir = NULL,
  save.mod = ifelse(!is.null(outdir), TRUE, FALSE),
  ...
)
```

Arguments

x	Numeric vector or matrix / data frame of features i.e. independent variables
y	Numeric vector of outcome, i.e. dependent variable

x.test	Numeric vector or matrix / data frame of testing set features Columns must correspond to columns in x
y.test	Numeric vector of testing set outcome
x.name	Character: Name for feature set
y.name	Character: Name for outcome
family	Error distribution and link function. See stats::family
interactions	List of character pairs denoting column names in x that should be entered as interaction terms in the GLM formula
class.method	Character: Define "logistic" or "multinom" for classification. The only purpose of this is so you can try nnet::multinom instead of glm for binary classification
weights	Numeric vector: Weights for cases. For classification, weights takes precedence over ipw, therefore set weights = NULL if using ipw. Note: If weight are provided, ipw is not used. Leave NULL if setting ipw = TRUE. Default = NULL
ipw	Logical: If TRUE, apply inverse probability weighting (for Classification only). Note: If weights are provided, ipw is not used. Default = TRUE
ipw.type	Integer 0, 1, 2 1: class.weights as in 0, divided by max(class.weights) 2: class.weights as in 0, divided by min(class.weights) Default = 2
upsample	Logical: If TRUE, upsample cases to balance outcome classes (for Classification only) Note: upsample will randomly sample with replacement if the length of the majority class is more than double the length of the class you are upsampling, thereby introducing randomness
downsample	Logical: If TRUE, downsample majority class to match size of minority class
resample.seed	Integer: If provided, will be used to set the seed during upsampling. Default = NULL (random seed)
intercept	Logical: If TRUE, fit an intercept term. Default = TRUE
polynomial	Logical: if TRUE, run lm on poly(x, poly.d) (creates orthogonal polynomials)
poly.d	Integer: degree of polynomial. Default = 3
poly.raw	Logical: if TRUE, use raw polynomials. Default, which should not really be changed is FALSE
print.plot	Logical: if TRUE, produce plot using mplot3 Takes precedence over plot.fitted and plot.predicted. Default = TRUE
plot.fitted	Logical: if TRUE, plot True (y) vs Fitted
plot.predicted	Logical: if TRUE, plot True (y.test) vs Predicted. Requires x.test and y.test
plot.theme	Character: "zero", "dark", "box", "darkbox"
na.action	How to handle missing values. See ?na.fail
removeMissingLevels	Logical: If TRUE, finds factors in x.test that contain levels not present in x and substitutes with NA. This would result in error otherwise and no predictions would be made, ending s_GLM prematurely
question	Character: the question you are attempting to answer with this model, in plain language.
rtclass	Character: Class type to use. "S3", "S4", "RC", "R6"
verbose	Logical: If TRUE, print summary to screen.
trace	Integer: If higher than 0, will print more information to the console. Default = 0

outdir	Path to output directory. If defined, will save Predicted vs. True plot, if available, as well as full model output, if save.mod is TRUE
save.mod	Logical: If TRUE, save all output to an RDS file in outdir save.mod is TRUE by default if an outdir is defined. If set to TRUE, and no outdir is defined, outdir defaults to paste0("./s.", mod.name)
...	Additional arguments

Details

A common problem with glm arises when the testing set containts a predictor with more levels than those in the same predictor in the training set, resulting in error. This can happen when training on resamples of a data set, especially after stratifying against a different outcome, and results in error and no prediction. s_GLM automatically finds such cases and substitutes levels present in x.test and not in x with NA.

Value

rtMod

Author(s)

E.D. Gennatas

See Also

elevate for external cross-validation

Other Supervised Learning: s_ADABOOST(), s_ADDTREE(), s_BART(), s_BAYESGLM(), s_BRUTO(), s_C50(), s_CART(), s_CTREE(), s_ET(), s_EVTREE(), s_GAM.default(), s_GAM.formula(), s_GAMSELX2(), s_GAMSELX(), s_GAMSEL(), s_GAM(), s_GBM0(), s_GBM3.R(), s_GBM(), s_GLMNET(), s_GLMTREE(), s_GLS(), s_H2ODL(), s_H2OGBM(), s_H2ORF(), s_IRF(), s_KNN(), s_LDA(), s_LMTREE(), s_LM(), s_MARS(), s_MLRF(), s_NBAYES(), s_NLA(), s_NLS(), s_NW(), s_POLYMARS(), s_PPR(), s_PPTREE(), s_QDA(), s_QRNN(), s_RANGER(), s_RFSRC(), s_RF(), s_SDA(), s_SGD(), s_SPLS(), s_SVM(), s_TFN(), s_XGBLIN(), s_XGBOOST(), s_XGB()

Other Interpretable models: s_ADDTREE(), s_C50(), s_CART(), s_GLMNET(), s_GLMTREE(), s_LMTREE()

Examples

```
x <- rnorm(100)
y <- .6 * x + 12 + rnorm(100)/2
mod <- s_GLM(x, y)
```

---

s_GLMNET *GLM with Elastic Net Regularization [C, R, S]*

Description

Train an elastic net model

Usage

```
s_GLMNET(
  x,
  y = NULL,
  x.test = NULL,
  y.test = NULL,
  x.name = NULL,
  y.name = NULL,
  grid.resample.rtset = rtset.resample("kfold", 5),
  grid.search.type = c("exhaustive", "randomized"),
  grid.randomized.p = 0.1,
  intercept = TRUE,
  nway.interactions = 0,
  family = NULL,
  alpha = seq(0, 1, 0.2),
  lambda = NULL,
  nlambda = 100,
  which.cv.lambda = c("lambda.1se", "lambda.min"),
  penalty.factor = NULL,
  weights = NULL,
  ipw = TRUE,
  ipw.type = 2,
  upsample = FALSE,
  downsample = FALSE,
  resample.seed = NULL,
  res.summary.fn = mean,
  save.grid.run = FALSE,
  metric = NULL,
  maximize = NULL,
  .gs = FALSE,
  save.gs.mod = FALSE,
  n.cores = rtCores,
  print.plot = TRUE,
  plot.fitted = NULL,
  plot.predicted = NULL,
  plot.theme = rtTheme,
  question = NULL,
  rtclass = NULL,
  verbose = TRUE,
  outdir = NULL,
  save.mod = ifelse(!is.null(outdir), TRUE, FALSE),
  ...
)
```

Arguments

x	Numeric vector or matrix / data frame of features i.e. independent variables
y	Numeric vector of outcome, i.e. dependent variable
x.test	Numeric vector or matrix / data frame of testing set features Columns must correspond to columns in x
y.test	Numeric vector of testing set outcome

x.name	Character: Name for feature set
y.name	Character: Name for outcome
grid.resample.rtset	List: Output of rtset.resample defining gridSearchLearn parameters.
grid.search.type	Character: Type of grid search to perform: "exhaustive" or "randomized".
grid.randomized.p	Float (0, 1): If grid.search.type = "randomized", randomly test this proportion of combinations.
intercept	Logical: If TRUE, include intercept in the model.
family	Error distribution and link function. See stats::family
alpha	[gS] Float [0, 1]: The elasticnet mixing parameter: a = 0 is the ridge penalty, a = 1 is the lasso penalty
lambda	[gS] Float vector: Best left to NULL, cv.glmnet will compute its own lambda sequence
weights	Numeric vector: Weights for cases. For classification, weights takes precedence over ipw, therefore set weights = NULL if using ipw. Note: If weight are provided, ipw is not used. Leave NULL if setting ipw = TRUE. Default = NULL
ipw	Logical: If TRUE, apply inverse probability weighting (for Classification only). Note: If weights are provided, ipw is not used. Default = TRUE
ipw.type	Integer 0, 1, 2 1: class.weights as in 0, divided by max(class.weights) 2: class.weights as in 0, divided by min(class.weights) Default = 2
upsample	Logical: If TRUE, upsample cases to balance outcome classes (for Classification only) Note: upsample will randomly sample with replacement if the length of the majority class is more than double the length of the class you are upsampling, thereby introducing randomness
downsample	Logical: If TRUE, downsample majority class to match size of minority class
resample.seed	Integer: If provided, will be used to set the seed during upsampling. Default = NULL (random seed)
res.summary.fn	Function: Used to average resample runs.
metric	Character: Metric to minimize, or maximize if maximize = TRUE during grid search. Default = NULL, which results in "Balanced Accuracy" for Classification, "MSE" for Regression, and "Coherence" for Survival Analysis.
maximize	Logical: If TRUE, metric will be maximized if grid search is run.
n.cores	Integer: Number of cores to use.
print.plot	Logical: if TRUE, produce plot using mplot3 Takes precedence over plot.fitted and plot.predicted. Default = TRUE
plot.fitted	Logical: if TRUE, plot True (y) vs Fitted
plot.predicted	Logical: if TRUE, plot True (y.test) vs Predicted. Requires x.test and y.test
plot.theme	Character: "zero", "dark", "box", "darkbox"
question	Character: the question you are attempting to answer with this model, in plain language.
rtclass	Character: Class type to use. "S3", "S4", "RC", "R6"
verbose	Logical: If TRUE, print summary to screen.

outdir	Path to output directory. If defined, will save Predicted vs. True plot, if available, as well as full model output, if save.mod is TRUE
save.mod	Logical: If TRUE, save all output to an RDS file in outdir save.mod is TRUE by default if an outdir is defined. If set to TRUE, and no outdir is defined, outdir defaults to paste0("./s.", mod.name)
...	Additional arguments

Details

s_GLMNET runs glmnet::cv.glmnet for each value of alpha, for each resample in grid.resample.rtset. Mean values for min.lambda and MSE (Regression) or Accuracy (Classification) are aggregated for each alpha and resample combination

[gS] Indicates tunable hyperparameters: If more than a single value is provided, grid search will be automatically performed

Author(s)

E.D. Gennatas

See Also

elevate for external cross-validation

Other Supervised Learning: s_ADABOOST(), s_ADDTREE(), s_BART(), s_BAYESGLM(), s_BRUTO(), s_C50(), s_CART(), s_CTREE(), s_ET(), s_EVTREE(), s_GAM.default(), s_GAM.formula(), s_GAMSELX2(), s_GAMSELX(), s_GAMSEL(), s_GAM(), s_GBM0(), s_GBM3.R(), s_GBM(), s_GLMTREE(), s_GLM(), s_GLS(), s_H2ODL(), s_H2OGBM(), s_H2ORF(), s_IRF(), s_KNN(), s_LDA(), s_LMTREE(), s_LM(), s_MARS(), s_MLRF(), s_NBAYES(), s_NLA(), s_NLS(), s_NW(), s_POLYMARS(), s_PPR(), s_PPTREE(), s_QDA(), s_QRNN(), s_RANGER(), s_RFSRC(), s_RF(), s_SDA(), s_SGD(), s_SPLS(), s_SVM(), s_TFN(), s_XGBLIN(), s_XGBOOST(), s_XGB()

Other Interpretable models: s_ADDTREE(), s_C50(), s_CART(), s_GLMTREE(), s_GLM(), s_LMTREE()

---

s_GLMTREE *Generalized Linear Model Tree [R]*

Description

Train a GLMTREE for regression or classification using rpart

Usage

```
s_GLMTREE(
  x,
  y = NULL,
  x.test = NULL,
  y.test = NULL,
  x.name = NULL,
  y.name = NULL,
  weights = NULL,
  alpha = 0.05,
  bonferroni = TRUE,
```

```
  minsize = NULL,
  maxdepth = Inf,
  minsplit = minsize,
  minbucket = minsize,
  epsilon = 1e-08,
  maxit = 25,
  ipw = TRUE,
  ipw.type = 2,
  upsample = FALSE,
  downsample = FALSE,
  resample.seed = NULL,
  na.action = na.exclude,
  grid.resample.rtset = rtset.resample("kfold", 5),
  grid.search.type = c("exhaustive", "randomized"),
  grid.randomized.p = 0.1,
  metric = NULL,
  maximize = NULL,
  n.cores = rtCores,
  print.plot = TRUE,
  plot.fitted = NULL,
  plot.predicted = NULL,
  plot.theme = rtTheme,
  question = NULL,
  verbose = TRUE,
  grid.verbose = verbose,
  outdir = NULL,
  save.mod = ifelse(!is.null(outdir), TRUE, FALSE),
  ...
)
```

Arguments

x	Numeric vector or matrix / data frame of features i.e. independent variables
y	Numeric vector of outcome, i.e. dependent variable
x.test	Numeric vector or matrix / data frame of testing set features Columns must correspond to columns in x
y.test	Numeric vector of testing set outcome
x.name	Character: Name for feature set
y.name	Character: Name for outcome
weights	Numeric vector: Weights for cases. For classification, weights takes precedence over ipw, therefore set weights = NULL if using ipw. Note: If weight are provided, ipw is not used. Leave NULL if setting ipw = TRUE. Default = NULL
maxdepth	[gS] Integer: Maximum depth of tree.
minsplit	[gS] Integer: Minimum number of cases that must belong in a node before considering a split.
minbucket	[gS] Integer: Minimum number of cases allowed in a child node.
ipw	Logical: If TRUE, apply inverse probability weighting (for Classification only). Note: If weights are provided, ipw is not used. Default = TRUE
ipw.type	Integer 0, 1, 2 1: class.weights as in 0, divided by max(class.weights) 2: class.weights as in 0, divided by min(class.weights) Default = 2

upsample	Logical: If TRUE, upsample cases to balance outcome classes (for Classification only) Note: upsample will randomly sample with replacement if the length of the majority class is more than double the length of the class you are upsampling, thereby introducing randomness
downsample	Logical: If TRUE, downsample majority class to match size of minority class
resample.seed	Integer: If provided, will be used to set the seed during upsampling. Default = NULL (random seed)
na.action	How to handle missing values. See ?na.fail
grid.resample.rtset	List: Output of rtset.resample defining gridSearchLearn parameters.
grid.search.type	Character: Type of grid search to perform: "exhaustive" or "randomized".
grid.randomized.p	Float (0, 1): If grid.search.type = "randomized", randomly test this proportion of combinations.
metric	Character: Metric to minimize, or maximize if maximize = TRUE during grid search. Default = NULL, which results in "Balanced Accuracy" for Classification, "MSE" for Regression, and "Coherence" for Survival Analysis.
maximize	Logical: If TRUE, metric will be maximized if grid search is run.
n.cores	Integer: Number of cores to use.
print.plot	Logical: if TRUE, produce plot using mplot3 Takes precedence over plot.fitted and plot.predicted. Default = TRUE
plot.fitted	Logical: if TRUE, plot True (y) vs Fitted
plot.predicted	Logical: if TRUE, plot True (y.test) vs Predicted. Requires x.test and y.test
plot.theme	Character: "zero", "dark", "box", "darkbox"
question	Character: the question you are attempting to answer with this model, in plain language.
verbose	Logical: If TRUE, print summary to screen.
grid.verbose	Logical: Passed to gridSearchLearn
outdir	Path to output directory. If defined, will save Predicted vs. True plot, if available, as well as full model output, if save.mod is TRUE
save.mod	Logical: If TRUE, save all output to an RDS file in outdir save.mod is TRUE by default if an outdir is defined. If set to TRUE, and no outdir is defined, outdir defaults to paste0("./s.", mod.name)
...	Additional arguments passed to partykit::mob_control
offset	Numeric vector of a priori known offsets

Value

Object of class rtMod

Author(s)

E.D. Gennatas

See Also

elevate for external cross-validation

Other Supervised Learning: s_ADABOOST(), s_ADDTREE(), s_BART(), s_BAYESGLM(), s_BRUTO(), s_C50(), s_CART(), s_CTREE(), s_ET(), s_EVTREE(), s_GAM.default(), s_GAM.formula(), s_GAMSELX2(), s_GAMSELX(), s_GAMSEL(), s_GAM(), s_GBM0(), s_GBM3.R(), s_GBM(), s_GLMNET(), s_GLM(), s_GLS(), s_H2ODL(), s_H2OGBM(), s_H2ORF(), s_IRF(), s_KNN(), s_LDA(), s_LMTREE(), s_LM(), s_MARS(), s_MLRF(), s_NBAYES(), s_NLA(), s_NLS(), s_NW(), s_POLYMARS(), s_PPR(), s_PPTREE(), s_QDA(), s_QRNN(), s_RANGER(), s_RFSRC(), s_RF(), s_SDA(), s_SGD(), s_SPLS(), s_SVM(), s_TFN(), s_XGBLIN(), s_XGBOOST(), s_XGB()

Other Tree-based methods: s_ADABOOST(), s_ADDTREE(), s_BART(), s_C50(), s_CART(), s_CTREE(), s_ET(), s_EVTREE(), s_GBM0(), s_GBM3.R(), s_GBM(), s_H2OGBM(), s_H2ORF(), s_IRF(), s_LMTREE(), s_MLRF(), s_PPTREE(), s_RANGER(), s_RFSRC(), s_RF(), s_XGBOOST(), s_XGB()

Other Interpretable models: s_ADDTREE(), s_C50(), s_CART(), s_GLMNET(), s_GLM(), s_LMTREE()

---

s_GLS *Generalized Least Squares [R]*

Description

Train a Generalized Least Squares regression model using `nlme::gls`

Usage

```
s_GLS(
  x,
  y = NULL,
  x.test = NULL,
  y.test = NULL,
  x.name = NULL,
  y.name = NULL,
  interactions = FALSE,
  nway.interactions = 0,
  covariate = NULL,
  weights = NULL,
  intercept = TRUE,
  print.plot = TRUE,
  plot.fitted = NULL,
  plot.predicted = NULL,
  plot.theme = rtTheme,
  na.action = na.exclude,
  question = NULL,
  rtclass = NULL,
  verbose = TRUE,
  trace = 0,
  outdir = NULL,
  save.mod = ifelse(!is.null(outdir), TRUE, FALSE),
  ...
)
```

Arguments

x	Numeric vector or matrix / data frame of features i.e. independent variables
y	Numeric vector of outcome, i.e. dependent variable
x.test	Numeric vector or matrix / data frame of testing set features Columns must correspond to columns in x
y.test	Numeric vector of testing set outcome
x.name	Character: Name for feature set
y.name	Character: Name for outcome
interactions	List of character pairs denoting column names in x that should be entered as interaction terms in the GLM formula
weights	Numeric vector: Weights for cases. For classification, weights takes precedence over ipw, therefore set weights = NULL if using ipw. Note: If weight are provided, ipw is not used. Leave NULL if setting ipw = TRUE. Default = NULL
intercept	Logical: If TRUE, fit an intercept term. Default = TRUE
print.plot	Logical: if TRUE, produce plot using mplot3 Takes precedence over plot.fitted and plot.predicted. Default = TRUE
plot.fitted	Logical: if TRUE, plot True (y) vs Fitted
plot.predicted	Logical: if TRUE, plot True (y.test) vs Predicted. Requires x.test and y.test
plot.theme	Character: "zero", "dark", "box", "darkbox"
na.action	How to handle missing values. See ?na.fail
question	Character: the question you are attempting to answer with this model, in plain language.
rtclass	Character: Class type to use. "S3", "S4", "RC", "R6"
verbose	Logical: If TRUE, print summary to screen.
trace	Integer: If higher than 0, will print more information to the console. Default = 0
outdir	Path to output directory. If defined, will save Predicted vs. True plot, if available, as well as full model output, if save.mod is TRUE
save.mod	Logical: If TRUE, save all output to an RDS file in outdir save.mod is TRUE by default if an outdir is defined. If set to TRUE, and no outdir is defined, outdir defaults to paste0("./s.", mod.name)
...	Additional arguments

Value

rtMod

Author(s)

E.D. Gennatas

See Also

Other Supervised Learning: s_ADABOOST(), s_ADDTREE(), s_BART(), s_BAYESGLM(), s_BRUTO(), s_C50(), s_CART(), s_CTREE(), s_ET(), s_EVTREE(), s_GAM.default(), s_GAM.formula(), s_GAMSELX2(), s_GAMSELX(), s_GAMSEL(), s_GAM(), s_GBM0(), s_GBM3.R(), s_GBM(), s_GLMNET(), s_GLMTREE(), s_GLM(), s_H2ODL(), s_H2OGBM(), s_H2ORF(), s_IRF(), s_KNN(), s_LDA(), s_LMTREE(), s_LM(), s_MARS(), s_MLRF(), s_NBAYES(), s_NLA(), s_NLS(), s_NW(), s_POLYMARS(), s_PPR(), s_PPTREE(), s_QDA(), s_QRNN(), s_RANGER(), s_RFSRC(), s_RF(), s_SDA(), s_SGD(), s_SPLS(), s_SVM(), s_TFN(), s_XGBLIN(), s_XGBOOST(), s_XGB()

s_H2ODL *Deep Learning on H2O [C, R]*

Description

Trains a Deep Neural Net using H2O (http://www.h2o.ai) Check out the H2O Flow at [ip]:[port], Default IP:port is "localhost:54321" e.g. if running on localhost, point your web browser to localhost:54321

Usage

```
s_H2ODL(
  x,
  y = NULL,
  x.test = NULL,
  y.test = NULL,
  x.valid = NULL,
  y.valid = NULL,
  x.name = NULL,
  y.name = NULL,
  ip = "localhost",
  port = 54321,
  n.hidden.nodes = c(20, 20),
  epochs = 1000,
  activation = "Rectifier",
  mini.batch.size = 1,
  learning.rate = 0.005,
  adaptive.rate = TRUE,
  rho = 0.99,
  epsilon = 1e-08,
  rate.annealing = 1e-06,
  rate.decay = 1,
  momentum.start = 0,
  momentum.ramp = 1e+06,
  momentum.stable = 0,
  nesterov.accelerated.gradient = TRUE,
  input.dropout.ratio = 0,
  hidden.dropout.ratios = NULL,
  l1 = 0,
  l2 = 0,
  max.w2 = 3.4028235e+38,
  nfolds = 0,
  initial.biases = NULL,
  initial.weights = NULL,
  loss = "Automatic",
  distribution = "AUTO",
  stopping.rounds = 5,
  stopping.metric = "AUTO",
  upsample = FALSE,
  downsample = FALSE,
  resample.seed = NULL,
  na.action = na.fail,
```

```
  n.cores = rtCores,
  print.plot = TRUE,
  plot.fitted = NULL,
  plot.predicted = NULL,
  plot.theme = rtTheme,
  question = NULL,
  verbose = TRUE,
  trace = 0,
  outdir = NULL,
  save.mod = ifelse(!is.null(outdir), TRUE, FALSE),
  ...
)
```

Arguments

x	Vector / Matrix / Data Frame: Training set Predictors
y	Vector: Training set outcome
x.test	Vector / Matrix / Data Frame: Testing set Predictors
y.test	Vector: Testing set outcome
x.valid	Vector / Matrix / Data Frame: Validation set Predictors
y.valid	Vector: Validation set outcome
x.name	Character: Name for feature set
y.name	Character: Name for outcome
ip	Character: IP address of H2O server. Default = "localhost"
port	Integer: Port number for server. Default = 54321
n.hidden.nodes	Integer vector of length equal to the number of hidden layers you wish to create
epochs	Integer: How many times to iterate through the dataset. Default = 1000
activation	Character: Activation function to use: "Tanh", "TanhWithDropout", "Rectifier", "RectifierWithDropout", "Maxout", "MaxoutWithDropout". Default = "Rectifier"
learning.rate	Float: Learning rate to use for training. Default = .005
adaptive.rate	Logical: If TRUE, use adaptive learning rate. Default = TRUE
rate.annealing	Float: Learning rate annealing: rate / (1 + rate_annealing * samples). Default = 1e-6
input.dropout.ratio	Float (0, 1): Dropout ratio for inputs
hidden.dropout.ratios	Vector, Float (0, 2): Dropout ratios for hidden layers
l1	Float (0, 1): L1 regularization (introduces sparseness; i.e. sets many weights to 0; reduces variance, increases generalizability)
l2	Float (0, 1): L2 regularization (prevents very large absolute weights; reduces variance, increases generalizability)
upsample	Logical: If TRUE, upsample cases to balance outcome classes (for Classification only) Note: upsample will randomly sample with replacement if the length of the majority class is more than double the length of the class you are upsampling, thereby introducing randomness
downsample	Logical: If TRUE, downsample majority class to match size of minority class

resample.seed	Integer: If provided, will be used to set the seed during upsampling. Default = NULL (random seed)
na.action	How to handle missing values. See ?na.fail
n.cores	Integer: Number of cores to use
print.plot	Logical: if TRUE, produce plot using mplot3 Takes precedence over plot.fitted and plot.predicted. Default = TRUE
plot.fitted	Logical: if TRUE, plot True (y) vs Fitted
plot.predicted	Logical: if TRUE, plot True (y.test) vs Predicted. Requires x.test and y.test
plot.theme	Character: "zero", "dark", "box", "darkbox"
question	Character: the question you are attempting to answer with this model, in plain language.
verbose	Logical: If TRUE, print summary to screen.
trace	Integer: If higher than 0, will print more information to the console. Default = 0
outdir	Path to output directory. If defined, will save Predicted vs. True plot, if available, as well as full model output, if save.mod is TRUE
save.mod	Logical: If TRUE, save all output to an RDS file in outdir save.mod is TRUE by default if an outdir is defined. If set to TRUE, and no outdir is defined, outdir defaults to paste0("./s.", mod.name)
...	Additional parameters to pass to h2o::h2o.deeplearning

Details

x & y form the training set. x.test & y.test form the testing set used only to test model generalizability. x.valid & y.valid form the validation set used to monitor training progress

Value

rtMod object

Author(s)

E.D. Gennatas

See Also

elevate for external cross-validation

Other Supervised Learning: s_ADABOOST(), s_ADDTREE(), s_BART(), s_BAYESGLM(), s_BRUTO(), s_C50(), s_CART(), s_CTREE(), s_ET(), s_EVTREE(), s_GAM.default(), s_GAM.formula(), s_GAMSELX2(), s_GAMSELX(), s_GAMSEL(), s_GAM(), s_GBM0(), s_GBM3.R(), s_GBM(), s_GLMNET(), s_GLMTREE(), s_GLM(), s_GLS(), s_H2OGBM(), s_H2ORF(), s_IRF(), s_KNN(), s_LDA(), s_LMTREE(), s_LM(), s_MARS(), s_MLRF(), s_NBAYES(), s_NLA(), s_NLS(), s_NW(), s_POLYMARS(), s_PPR(), s_PPTREE(), s_QDA(), s_QRNN(), s_RANGER(), s_RFSRC(), s_RF(), s_SDA(), s_SGD(), s_SPLS(), s_SVM(), s_TFN(), s_XGBLIN(), s_XGBOOST(), s_XGB()

Other Deep Learning: d_H2OAE(), s_TFN()

s_H2OGBM *Gradient Boosting Machine on H2O [C, R]*

Description

Trains a Gradient Boosting Machine using H2O (http://www.h2o.ai)

Usage

```
s_H2OGBM(
  x,
  y = NULL,
  x.test = NULL,
  y.test = NULL,
  x.name = NULL,
  y.name = NULL,
  ip = "localhost",
  port = 54321,
  h2o.init = TRUE,
  gs.h2o.init = FALSE,
  h2o.shutdown.at.end = TRUE,
  grid.resample.rtset = rtset.resample("kfold", 5),
  metric = NULL,
  maximize = NULL,
  n.trees = 10000,
  force.n.trees = NULL,
  max.depth = 5,
  n.stopping.rounds = 50,
  stopping.metric = "AUTO",
  p.col.sample = 1,
  p.row.sample = 0.9,
  minobsinnode = 5,
  min.split.improvement = 1e-05,
  quantile.alpha = 0.5,
  learning.rate = 0.01,
  learning.rate.annealing = 1,
  weights = NULL,
  ipw = TRUE,
  ipw.type = 2,
  upsample = FALSE,
  downsample = FALSE,
  resample.seed = NULL,
  na.action = na.fail,
  grid.n.cores = 1,
  n.cores = rtCores,
  imetrics = FALSE,
  .gs = FALSE,
  print.plot = TRUE,
  plot.fitted = NULL,
  plot.predicted = NULL,
  plot.theme = rtTheme,
```

```
  question = NULL,
  verbose = TRUE,
  trace = 0,
  grid.verbose = verbose,
  save.mod = FALSE,
  outdir = NULL,
  ...
)
```

Arguments

x	Numeric vector or matrix / data frame of features i.e. independent variables
y	Numeric vector of outcome, i.e. dependent variable
x.test	Numeric vector or matrix / data frame of testing set features Columns must correspond to columns in x
y.test	Numeric vector of testing set outcome
x.name	Character: Name for feature set
y.name	Character: Name for outcome
ip	Character: IP address of H2O server. Default = "localhost"
port	Integer: Port number for server. Default = 54321
n.trees	Integer: Number of trees to grow. Maximum number of trees if n.stopping.rounds > 0
max.depth	[gS] Integer: Depth of trees to grow
n.stopping.rounds	Integer: If > 0, stop training if stopping.metric does not improve for this many rounds
stopping.metric	Character: "AUTO" (Default), "deviance", "logloss", "MSE", "RMSE", "MAE", "RMSLE", "AUC", "lift_top_group", "misclassification", "mean_per_class_error"
p.col.sample	[gS]
p.row.sample	[gS]
minobsinnode	[gS]
learning.rate	[gS]
learning.rate.annealing	[gS]
weights	Numeric vector: Weights for cases. For classification, weights takes precedence over ipw, therefore set weights = NULL if using ipw. Note: If weight are provided, ipw is not used. Leave NULL if setting ipw = TRUE. Default = NULL
ipw	Logical: If TRUE, apply inverse probability weighting (for Classification only). Note: If weights are provided, ipw is not used. Default = TRUE
ipw.type	Integer 0, 1, 2 1: class.weights as in 0, divided by max(class.weights) 2: class.weights as in 0, divided by min(class.weights) Default = 2
upsample	Logical: If TRUE, upsample cases to balance outcome classes (for Classification only) Note: upsample will randomly sample with replacement if the length of the majority class is more than double the length of the class you are upsampling, thereby introducing randomness
downsample	Logical: If TRUE, downsample majority class to match size of minority class

resample.seed	Integer: If provided, will be used to set the seed during upsampling. Default = NULL (random seed)
na.action	How to handle missing values. See ?na.fail
n.cores	Integer: Number of cores to use
.gs	Internal use only
print.plot	Logical: if TRUE, produce plot using mplot3 Takes precedence over plot.fitted and plot.predicted. Default = TRUE
plot.fitted	Logical: if TRUE, plot True (y) vs Fitted
plot.predicted	Logical: if TRUE, plot True (y.test) vs Predicted. Requires x.test and y.test
plot.theme	Character: "zero", "dark", "box", "darkbox"
question	Character: the question you are attempting to answer with this model, in plain language.
verbose	Logical: If TRUE, print summary to screen.
trace	Integer: If higher than 0, will print more information to the console. Default = 0
save.mod	Logical: If TRUE, save all output to an RDS file in outdir save.mod is TRUE by default if an outdir is defined. If set to TRUE, and no outdir is defined, outdir defaults to paste0("./s.", mod.name)
outdir	Path to output directory. If defined, will save Predicted vs. True plot, if available, as well as full model output, if save.mod is TRUE
...	Additional arguments

Details

[gS] denotes tunable hyperparameters Warning: If you get an HTTP 500 error at random, use h2o.shutdown() to shutdown the server. It will be restarted when s_H2OGBM is called

Value

rtMod object

Author(s)

E.D. Gennatas

See Also

elevate for external cross-validation

Other Supervised Learning: s_ADABOOST(), s_ADDTREE(), s_BART(), s_BAYESGLM(), s_BRUTO(), s_C50(), s_CART(), s_CTREE(), s_ET(), s_EVTREE(), s_GAM.default(), s_GAM.formula(), s_GAMSELX2(), s_GAMSELX(), s_GAMSEL(), s_GAM(), s_GBM0(), s_GBM3.R(), s_GBM(), s_GLMNET(), s_GLMTREE(), s_GLM(), s_GLS(), s_H2ODL(), s_H2ORF(), s_IRF(), s_KNN(), s_LDA(), s_LMTREE(), s_LM(), s_MARS(), s_MLRF(), s_NBAYES(), s_NLA(), s_NLS(), s_NW(), s_POLYMARS(), s_PPR(), s_PPTREE(), s_QDA(), s_QRNN(), s_RANGER(), s_RFSRC(), s_RF(), s_SDA(), s_SGD(), s_SPLS(), s_SVM(), s_TFN(), s_XGBLIN(), s_XGBOOST(), s_XGB()

Other Tree-based methods: s_ADABOOST(), s_ADDTREE(), s_BART(), s_C50(), s_CART(), s_CTREE(), s_ET(), s_EVTREE(), s_GBM0(), s_GBM3.R(), s_GBM(), s_GLMTREE(), s_H2ORF(), s_IRF(), s_LMTREE(), s_MLRF(), s_PPTREE(), s_RANGER(), s_RFSRC(), s_RF(), s_XGBOOST(), s_XGB()

s_H2ORF *Random Forest on H2O [C, R]*

Description

Trains a Random Forest model using H2O (http://www.h2o.ai)

Usage

```
s_H2ORF(
  x,
  y = NULL,
  x.test = NULL,
  y.test = NULL,
  x.valid = NULL,
  y.valid = NULL,
  x.name = NULL,
  y.name = NULL,
  ip = "localhost",
  port = 54321,
  n.trees = 500,
  max.depth = 20,
  n.stopping.rounds = 50,
  mtry = -1,
  nfolds = 0,
  weights = NULL,
  weights.test = NULL,
  balance.classes = TRUE,
  upsample = FALSE,
  downsample = FALSE,
  resample.seed = NULL,
  na.action = na.fail,
  n.cores = rtCores,
  print.plot = TRUE,
  plot.fitted = NULL,
  plot.predicted = NULL,
  plot.theme = rtTheme,
  question = NULL,
  verbose = TRUE,
  trace = 0,
  save.mod = FALSE,
  outdir = NULL,
  ...
)
```

Arguments

x	Training set features
y	Training set outcome
x.test	Testing set features (Used to evaluate model performance)

y.test	Testing set outcome
x.valid	Validation set features (Used to build model / tune hyperparameters)
y.valid	Validation set outcome
x.name	Character: Name for feature set
y.name	Character: Name for outcome
ip	Character: IP address of H2O server. Default = "localhost"
port	Integer: Port to connect to at ip
n.trees	Integer: Number of trees to grow
weights	Numeric vector: Weights for cases. For classification, weights takes precedence over ipw, therefore set weights = NULL if using ipw. Note: If weight are provided, ipw is not used. Leave NULL if setting ipw = TRUE. Default = NULL
upsample	Logical: If TRUE, upsample cases to balance outcome classes (for Classification only) Note: upsample will randomly sample with replacement if the length of the majority class is more than double the length of the class you are upsampling, thereby introducing randomness
downsample	Logical: If TRUE, downsample majority class to match size of minority class
resample.seed	Integer: If provided, will be used to set the seed during upsampling. Default = NULL (random seed)
na.action	How to handle missing values. See ?na.fail
n.cores	Integer: Number of cores to use
print.plot	Logical: if TRUE, produce plot using mplot3 Takes precedence over plot.fitted and plot.predicted. Default = TRUE
plot.fitted	Logical: if TRUE, plot True (y) vs Fitted
plot.predicted	Logical: if TRUE, plot True (y.test) vs Predicted. Requires x.test and y.test
plot.theme	Character: "zero", "dark", "box", "darkbox"
question	Character: the question you are attempting to answer with this model, in plain language.
verbose	Logical: If TRUE, print summary to screen.
trace	Integer: If higher than 0, will print more information to the console. Default = 0
save.mod	Logical: If TRUE, save all output to an RDS file in outdir save.mod is TRUE by default if an outdir is defined. If set to TRUE, and no outdir is defined, outdir defaults to paste0("./s.", mod.name)
outdir	Path to output directory. If defined, will save Predicted vs. True plot, if available, as well as full model output, if save.mod is TRUE
...	Additional parameters to pass to h2o::h2o.randomForest
epochs	Numeric: How many times to iterate through the dataset. Default = 10

Value

rtMod object

Author(s)

E.D. Gennatas

See Also

elevate for external cross-validation

Other Supervised Learning: s_ADABOOST(), s_ADDTREE(), s_BART(), s_BAYESGLM(), s_BRUTO(), s_C50(), s_CART(), s_CTREE(), s_ET(), s_EVTREE(), s_GAM.default(), s_GAM.formula(), s_GAMSELX2(), s_GAMSELX(), s_GAMSEL(), s_GAM(), s_GBM0(), s_GBM3.R(), s_GBM(), s_GLMNET(), s_GLMTREE(), s_GLM(), s_GLS(), s_H2ODL(), s_H2OGBM(), s_IRF(), s_KNN(), s_LDA(), s_LMTREE(), s_LM(), s_MARS(), s_MLRF(), s_NBAYES(), s_NLA(), s_NLS(), s_NW(), s_POLYMARS(), s_PPR(), s_PPTREE(), s_QDA(), s_QRNN(), s_RANGER(), s_RFSRC(), s_RF(), s_SDA(), s_SGD(), s_SPLS(), s_SVM(), s_TFN(), s_XGBLIN(), s_XGBOOST(), s_XGB()

Other Tree-based methods: s_ADABOOST(), s_ADDTREE(), s_BART(), s_C50(), s_CART(), s_CTREE(), s_ET(), s_EVTREE(), s_GBM0(), s_GBM3.R(), s_GBM(), s_GLMTREE(), s_H2OGBM(), s_IRF(), s_LMTREE(), s_MLRF(), s_PPTREE(), s_RANGER(), s_RFSRC(), s_RF(), s_XGBOOST(), s_XGB()

---

s_IRF *Iterative Random Forest [C, R]*

Description

Train iterative Random Forests for regression or classification using iRF

Usage

```
s_IRF(
  x,
  y = NULL,
  x.test = NULL,
  y.test = NULL,
  x.name = NULL,
  y.name = NULL,
  n.trees = 1000,
  n.iter = 5,
  n.bootstrap = 30,
  interactions.return = NULL,
  classwt = NULL,
  ipw = TRUE,
  upsample = FALSE,
  downsample = FALSE,
  resample.seed = NULL,
  autotune = FALSE,
  n.trees.try = 500,
  stepFactor = 2,
  mtry = NULL,
  mtryStart = NULL,
  mtry.select.prob = NULL,
  proximity = FALSE,
  importance = TRUE,
  replace = TRUE,
  min.node.size = 1,
  strata = NULL,
  sampsize = NULL,
```

```
  tune.do.trace = FALSE,
  print.tune.plot = FALSE,
  print.plot = TRUE,
  plot.fitted = NULL,
  plot.predicted = NULL,
  plot.theme = rtTheme,
  n.cores = rtCores,
  question = NULL,
  verbose = TRUE,
  trace = 0,
  outdir = NULL,
  save.mod = ifelse(!is.null(outdir), TRUE, FALSE),
  ...
)
```

Arguments

x	Numeric vector or matrix / data frame of features i.e. independent variables
y	Numeric vector of outcome, i.e. dependent variable
x.test	Numeric vector or matrix / data frame of testing set features Columns must correspond to columns in x
y.test	Numeric vector of testing set outcome
x.name	Character: Name for feature set
y.name	Character: Name for outcome
n.trees	Integer: Number of trees to grow. Default = 1000
classwt	Vector, Float: Priors of the classes for classification only. Need not add up to 1
ipw	Logical: If TRUE, apply inverse probability weighting (for Classification only). Note: If weights are provided, ipw is not used. Default = TRUE
upsample	Logical: If TRUE, upsample training set cases not belonging in majority outcome group
downsample	Logical: If TRUE, downsample majority class to match size of minority class
resample.seed	Integer: If provided, will be used to set the seed during upsampling. Default = NULL (random seed)
autotune	Logical: If TRUE, use randomForest::tuneRF to determine mtry
n.trees.try	Integer: Number of trees to train for tuning, if autotune = TRUE
stepFactor	Float: If autotune = TRUE, at each tuning iteration, mtry is multiplied or divided by this value. Default = 1.5
mtry	[gS] Integer: Number of features sampled randomly at each split
mtryStart	Integer: If autotune = TRUE, start at this value for mtry
proximity	Logical: If TRUE, calculate proximity measure among cases.
importance	Logical: If TRUE, estimate variable relative importance.
replace	Logical: If TRUE, sample cases with replacement during training.
strata	Vector, Factor: Will be used for stratified sampling
sampsize	Integer: Size of sample to draw. In Classification, if strata is defined, this can be a vector of the same length, in which case, corresponding values determine how many cases are drawn from the strata.

tune.do.trace	Same as do.trace but for tuning, when autotune = TRUE
print.tune.plot	Logical: passed to randomForest::tuneRF.
print.plot	Logical: if TRUE, produce plot using mplot3 Takes precedence over plot.fitted and plot.predicted. Default = TRUE
plot.fitted	Logical: if TRUE, plot True (y) vs Fitted
plot.predicted	Logical: if TRUE, plot True (y.test) vs Predicted. Requires x.test and y.test
plot.theme	Character: "zero", "dark", "box", "darkbox"
n.cores	Integer: Number of cores to use.
question	Character: the question you are attempting to answer with this model, in plain language.
verbose	Logical: If TRUE, print summary to screen.
outdir	String, Optional: Path to directory to save output
save.mod	Logical: If TRUE, save all output to an RDS file in outdir save.mod is TRUE by default if an outdir is defined. If set to TRUE, and no outdir is defined, outdir defaults to paste0("./s.", mod.name)
...	Additional arguments to be passed to iRF::iRF

Details

If autotue = TRUE, iRF::tuneRF will be run to determine best mtry value.

Value

rtMod object

Author(s)

E.D. Gennatas

See Also

elevate for external cross-validation

Other Supervised Learning: s_ADABOOST(), s_ADDTREE(), s_BART(), s_BAYESGLM(), s_BRUTO(), s_C50(), s_CART(), s_CTREE(), s_ET(), s_EVTREE(), s_GAM.default(), s_GAM.formula(), s_GAMSELX2(), s_GAMSELX(), s_GAMSEL(), s_GAM(), s_GBM0(), s_GBM3.R(), s_GBM(), s_GLMNET(), s_GLMTREE(), s_GLM(), s_GLS(), s_H2ODL(), s_H2OGBM(), s_H2ORF(), s_KNN(), s_LDA(), s_LMTREE(), s_LM(), s_MARS(), s_MLRF(), s_NBAYES(), s_NLA(), s_NLS(), s_NW(), s_POLYMARS(), s_PPR(), s_PPTREE(), s_QDA(), s_QRNN(), s_RANGER(), s_RFSRC(), s_RF(), s_SDA(), s_SGD(), s_SPLS(), s_SVM(), s_TFN(), s_XGBLIN(), s_XGBOOST(), s_XGB()

Other Tree-based methods: s_ADABOOST(), s_ADDTREE(), s_BART(), s_C50(), s_CART(), s_CTREE(), s_ET(), s_EVTREE(), s_GBM0(), s_GBM3.R(), s_GBM(), s_GLMTREE(), s_H2OGBM(), s_H2ORF(), s_LMTREE(), s_MLRF(), s_PPTREE(), s_RANGER(), s_RFSRC(), s_RF(), s_XGBOOST(), s_XGB()

s_KNN *k-Nearest Neighbors Classification and Regression [C, R]*

Description

Train a k-Nearest Neighbors learner for regression or classification using FNN

Usage

```
s_KNN(
  x,
  y = NULL,
  x.test = NULL,
  y.test = NULL,
  x.name = NULL,
  y.name = NULL,
  k = 3,
  algorithm = "kd_tree",
  print.plot = TRUE,
  plot.fitted = NULL,
  plot.predicted = NULL,
  plot.theme = rtTheme,
  question = NULL,
  rtclass = NULL,
  verbose = TRUE,
  outdir = NULL,
  save.mod = ifelse(!is.null(outdir), TRUE, FALSE),
  ...
)
```

Arguments

x	Numeric vector or matrix of features, i.e. independent variables
y	Numeric vector of outcome, i.e. dependent variable
x.test	(Optional) Numeric vector or matrix of validation set features must have set of columns as x
y.test	(Optional) Numeric vector of validation set outcomes
k	Integer: Number of neighbors considered
algorithm	Character: Algorithm to use. Options: "kd_tree", "cover_tree", "brute"
outdir	Optional. Path to directory to save output

Details

Note: FNN's KNN does not have a predict function

Value

Object of class rtMod

Author(s)

E.D. Gennatas

See Also

elevate for external cross-validation

Other Supervised Learning: s_ADABOOST(), s_ADDTREE(), s_BART(), s_BAYESGLM(), s_BRUTO(), s_C50(), s_CART(), s_CTREE(), s_ET(), s_EVTREE(), s_GAM.default(), s_GAM.formula(), s_GAMSELX2(), s_GAMSELX(), s_GAMSEL(), s_GAM(), s_GBM0(), s_GBM3.R(), s_GBM(), s_GLMNET(), s_GLMTREE(), s_GLM(), s_GLS(), s_H2ODL(), s_H2OGBM(), s_H2ORF(), s_IRF(), s_LDA(), s_LMTREE(), s_LM(), s_MARS(), s_MLRF(), s_NBAYES(), s_NLA(), s_NLS(), s_NW(), s_POLYMARS(), s_PPR(), s_PPTREE(), s_QDA(), s_QRNN(), s_RANGER(), s_RFSRC(), s_RF(), s_SDA(), s_SGD(), s_SPLS(), s_SVM(), s_TFN(), s_XGBLIN(), s_XGBOOST(), s_XGB()

---

s_LDA *Linear Discriminant Analysis*

Description

Train an LDA Classifier using MASS::lda

Usage

```
s_LDA(
  x,
  y = NULL,
  x.test = NULL,
  y.test = NULL,
  prior = NULL,
  method = "moment",
  nu = NULL,
  upsample = TRUE,
  downsample = FALSE,
  resample.seed = NULL,
  x.name = NULL,
  y.name = NULL,
  print.plot = TRUE,
  plot.fitted = NULL,
  plot.predicted = NULL,
  plot.theme = rtTheme,
  question = NULL,
  rtclass = NULL,
  verbose = TRUE,
  outdir = NULL,
  save.mod = ifelse(!is.null(outdir), TRUE, FALSE),
  ...
)
```

Arguments

x	Numeric vector or matrix / data frame of features i.e. independent variables
y	Numeric vector of outcome, i.e. dependent variable
x.test	Numeric vector or matrix / data frame of testing set features Columns must correspond to columns in x
y.test	Numeric vector of testing set outcome
upsample	Logical: If TRUE, upsample cases to balance outcome classes (for Classification only) Note: upsample will randomly sample with replacement if the length of the majority class is more than double the length of the class you are upsampling, thereby introducing randomness
downsample	Logical: If TRUE, downsample majority class to match size of minority class
resample.seed	Integer: If provided, will be used to set the seed during upsampling. Default = NULL (random seed)
x.name	Character: Name for feature set
y.name	Character: Name for outcome
print.plot	Logical: if TRUE, produce plot using mplot3 Takes precedence over plot.fitted and plot.predicted. Default = TRUE
plot.fitted	Logical: if TRUE, plot True (y) vs Fitted
plot.predicted	Logical: if TRUE, plot True (y.test) vs Predicted. Requires x.test and y.test
plot.theme	Character: "zero", "dark", "box", "darkbox"
question	Character: the question you are attempting to answer with this model, in plain language.
rtclass	Character: Class type to use. "S3", "S4", "RC", "R6"
verbose	Logical: If TRUE, print summary to screen.
outdir	Path to output directory. If defined, will save Predicted vs. True plot, if available, as well as full model output, if save.mod is TRUE
save.mod	Logical: If TRUE, save all output to an RDS file in outdir save.mod is TRUE by default if an outdir is defined. If set to TRUE, and no outdir is defined, outdir defaults to paste0("./s.", mod.name)
...	Additional arguments

Details

Note: LDA requires all predictors to be numeric. The variable importance output ("varimp") is the vector of coefficients for LD1

Value

rtMod object

Author(s)

E.D. Gennatas

See Also

elevate for external cross-validation

Other Supervised Learning: s_ADABOOST(), s_ADDTREE(), s_BART(), s_BAYESGLM(), s_BRUTO(), s_C50(), s_CART(), s_CTREE(), s_ET(), s_EVTREE(), s_GAM.default(), s_GAM.formula(), s_GAMSELX2(), s_GAMSELX(), s_GAMSEL(), s_GAM(), s_GBM0(), s_GBM3.R(), s_GBM(), s_GLMNET(), s_GLMTREE(), s_GLM(), s_GLS(), s_H2ODL(), s_H2OGBM(), s_H2ORF(), s_IRF(), s_KNN(), s_LMTREE(), s_LM(), s_MARS(), s_MLRF(), s_NBAYES(), s_NLA(), s_NLS(), s_NW(), s_POLYMARS(), s_PPR(), s_PPTREE(), s_QDA(), s_QRNN(), s_RANGER(), s_RFSRC(), s_RF(), s_SDA(), s_SGD(), s_SPLS(), s_SVM(), s_TFN(), s_XGBLIN(), s_XGBOOST(), s_XGB()

---

s_LIHAD *The Linear Hard Hybrid Tree: Hard Additive Tree (no gamma) with Linear Nodes [R]*

Description

Train a Linear Hard Hybrid Tree for Regression

Usage

```
s_LIHAD(
  x,
  y = NULL,
  x.test = NULL,
  y.test = NULL,
  max.depth = 3,
  alpha = 0,
  lambda = 0.1,
  lincoef.params = rtset.lincoef("glmnet"),
  minobsinnode = 2,
  minobsinnode.lin = 10,
  learning.rate = 1,
  part.minsplit = 2,
  part.xval = 0,
  part.max.depth = 1,
  part.cp = 0,
  weights = NULL,
  metric = "MSE",
  maximize = FALSE,
  grid.resample.rtset = rtset.grid.resample(),
  keep.x = FALSE,
  simplify = TRUE,
  cxrcoef = FALSE,
  n.cores = rtCores,
  verbose = TRUE,
  verbose.predict = FALSE,
  trace = 0,
  x.name = NULL,
  y.name = NULL,
  question = NULL,
```

```
  outdir = NULL,
  print.plot = TRUE,
  plot.fitted = NULL,
  plot.predicted = NULL,
  plot.theme = rtTheme,
  save.mod = FALSE
)
```

Arguments

x	Numeric vector or matrix / data frame of features i.e. independent variables
y	Numeric vector of outcome, i.e. dependent variable
x.test	Numeric vector or matrix / data frame of testing set features Columns must correspond to columns in x
y.test	Numeric vector of testing set outcome
max.depth	[gS] Integer: Max depth of additive tree. Default = 3
alpha	[gS] Float: lincoef alpha Overrides lincoef.params alpha
lambda	[gS] Float: lincoef lambda. Overrides lincoef.params lambda
lincoef.params	Named List: Output of rtset.lincoef
minobsinnode	[gS] Integer: Minimum N observations needed in node, before considering splitting
learning.rate	[gS] Float (0, 1): Learning rate. Default = 1
part.max.depth	Integer: Max depth for each tree model within the additive tree
part.cp	[gS] Float: Minimum complexity needed to allow split by rpart. Default = 0
weights	Numeric vector: Weights for cases. For classification, weights takes precedence over ipw, therefore set weights = NULL if using ipw. Note: If weight are provided, ipw is not used. Leave NULL if setting ipw = TRUE. Default = NULL
cxrcoef	Logical: Passed to predict.lihad, if TRUE, returns cases by coefficients matrix. Default = FALSE
verbose	Logical: If TRUE, print summary to screen.
trace	Integer: If higher than 0, will print more information to the console. Default = 0
x.name	Character: Name for feature set
y.name	Character: Name for outcome
question	Character: the question you are attempting to answer with this model, in plain language.
outdir	Path to output directory. If defined, will save Predicted vs. True plot, if available, as well as full model output, if save.mod is TRUE
print.plot	Logical: if TRUE, produce plot using mplot3 Takes precedence over plot.fitted and plot.predicted. Default = TRUE
plot.fitted	Logical: if TRUE, plot True (y) vs Fitted
plot.predicted	Logical: if TRUE, plot True (y.test) vs Predicted. Requires x.test and y.test
plot.theme	Character: "zero", "dark", "box", "darkbox"
save.mod	Logical: If TRUE, save all output to an RDS file in outdir save.mod is TRUE by default if an outdir is defined. If set to TRUE, and no outdir is defined, outdir defaults to paste0("./s.", mod.name)

Details

The Hybrid Tree grows a tree using a sequence of regularized linear models and tree stumps Use s_LINAD for the standard Linear Additive Tree Algorithm, which grows branches stepwise and includes all observations weighted by gamma

Grid searched parameters: max.depth, alpha, lambda, minobsinnode, learning.rate, part.cp

Author(s)

E.D. Gennatas

---

s_LIHADBOOST *Boosting of Linear Hard Additive Trees [R]*

Description

Boost a Linear Hard Additive Tree (i.e. LIHAD, i.e. LINAD with hard splits)

Usage

```
s_LIHADBOOST(
  x,
  y = NULL,
  x.test = NULL,
  y.test = NULL,
  resid = NULL,
  boost.obj = NULL,
  learning.rate = 0.5,
  case.p = 1,
  max.depth = 5,
  gamma = 0.1,
  alpha = 0,
  lambda = 1,
  lambda.seq = NULL,
  minobsinnode = 2,
  minobsinnode.lin = 10,
  shrinkage = 1,
  part.minsplit = 2,
  part.xval = 0,
  part.max.depth = 1,
  part.cp = 0,
  part.minbucket = 5,
 lin.type = c("glmnet", "cv.glmnet", "lm.ridge", "allSubsets", "forwardStepwise",
    "backwardStepwise", "glm", "sgd", "solve", "none"),
  cv.glmnet.nfolds = 5,
  which.cv.glmnet.lambda = "lambda.min",
  max.iter = 10,
  tune.n.iter = TRUE,
  earlystop.params = rtset.earlystop(),
  lookback = TRUE,
  init = NULL,
```

```
  .gs = FALSE,
  grid.resample.rtset = rtset.resample("kfold", 5),
  grid.search.type = "exhaustive",
  metric = NULL,
  maximize = NULL,
  cxrcoef = FALSE,
  print.progress.every = 5,
  print.error.plot = "final",
  x.name = NULL,
  y.name = NULL,
  question = NULL,
  base.verbose = FALSE,
  verbose = TRUE,
  grid.verbose = FALSE,
  trace = 0,
  prefix = NULL,
  plot.fitted = NULL,
  plot.predicted = NULL,
  plot.theme = rtTheme,
  print.plot = TRUE,
  print.base.plot = FALSE,
  print.tune.plot = TRUE,
  plot.type = "l",
  save.gridrun = FALSE,
  outdir = NULL,
  n.cores = rtCores,
  save.mod = ifelse(!is.null(outdir), TRUE, FALSE),
  ...
)
```

Arguments

x	Data frame: Input features
y	Vector: Output
x.test	Numeric vector or matrix / data frame of testing set features Columns must correspond to columns in x
y.test	Numeric vector of testing set outcome
learning.rate	Float (0, 1] Learning rate for the additive steps
max.iter	Integer: Maximum number of iterations (additive steps) to perform. Default = 10
init	Float: Initial value for prediction. Default = mean(y)
cxrcoef	Logical: If TRUE, pass cxr = TRUE, cxrcoef = TRUE to predict.hytreew
print.error.plot	String or Integer: "final" plots a training and validation (if available) error curve at the end of training. If integer, plot training and validation error curve every this many iterations during training
x.name	Character: Name for feature set
y.name	Character: Name for outcome
question	Character: the question you are attempting to answer with this model, in plain language.

base.verbose	Logical: verbose argument passed to learner
verbose	Logical: If TRUE, print summary to screen.
trace	Integer: If > 0, print diagnostic info to console
plot.fitted	Logical: if TRUE, plot True (y) vs Fitted
plot.predicted	Logical: if TRUE, plot True (y.test) vs Predicted. Requires x.test and y.test
plot.theme	Character: "zero", "dark", "box", "darkbox"
print.plot	Logical: if TRUE, produce plot using mplot3 Takes precedence over plot.fitted and plot.predicted. Default = TRUE
print.base.plot	Logical: Passed to print.plot argument of base learner, i.e. if TRUE, print error plot for each base learner
outdir	Path to output directory. If defined, will save Predicted vs. True plot, if available, as well as full model output, if save.mod is TRUE
save.mod	Logical: If TRUE, save all output to an RDS file in outdir save.mod is TRUE by default if an outdir is defined. If set to TRUE, and no outdir is defined, outdir defaults to paste0("./s.", mod.name)
...	Additional parameters to be passed to learner

Details

By default, early stopping works by checking training loss.

Author(s)

E.D. Gennatas

---

s_LINAD *Linear Additive Tree [C, R]*

Description

Train a Linear Additive Tree for Regression or Binary Classification

Usage

```
s_LINAD(
  x,
  y = NULL,
  x.test = NULL,
  y.test = NULL,
  weights = NULL,
  ipw = TRUE,
  ipw.type = 1,
  upsample = FALSE,
  downsample = FALSE,
  resample.seed = NULL,
  max.leaves = 8,
  leaf.model = c("line", "spline"),
```

```
  gamlearner = "gamsel",
  gam.params = list(),
  nvmax = 3,
  force.max.leaves = NULL,
  lookback = TRUE,
  gamma = 0.5,
  gamma.on.lin = FALSE,
 lin.type = c("glmnet", "forwardStepwise", "cv.glmnet", "lm.ridge", "allSubsets",
    "backwardStepwise", "glm", "solve", "none"),
  single.lin.type = "glmnet",
  cv.glmnet.nfolds = 5,
  which.cv.glmnet.lambda = "lambda.min",
  alpha = 1,
  lambda = 0.05,
  lambda.seq = NULL,
  minobsinnode.lin = 10,
  learning.rate = 0.5,
  part.minsplit = 5,
  part.xval = 0,
  part.max.depth = 1,
  part.cp = 0,
  part.minbucket = 3,
  .rho = TRUE,
  rho.max = 1000,
  init = NULL,
  metric = "auto",
  maximize = NULL,
  grid.resample.rtset = rtset.resample("kfold", 5),
  grid.search.type = "exhaustive",
  save.gridrun = FALSE,
  grid.verbose = verbose,
  select.leaves.smooth = TRUE,
  cluster = FALSE,
  keep.x = FALSE,
  simplify = TRUE,
  cxrcoef = FALSE,
  n.cores = rtCores,
  .preprocess = NULL,
  verbose = TRUE,
  plot.tuning = TRUE,
  verbose.predict = FALSE,
  trace = 1,
  x.name = NULL,
  y.name = NULL,
  question = NULL,
  outdir = NULL,
  print.plot = TRUE,
  plot.fitted = NULL,
  plot.predicted = NULL,
  plot.theme = rtTheme,
  save.mod = FALSE,
  .gs = FALSE
```

)

Arguments

x	Numeric vector or matrix / data frame of features i.e. independent variables
y	Numeric vector of outcome, i.e. dependent variable
x.test	Numeric vector or matrix / data frame of testing set features Columns must correspond to columns in x
y.test	Numeric vector of testing set outcome
weights	Numeric vector: Weights for cases. For classification, weights takes precedence over ipw, therefore set weights = NULL if using ipw. Note: If weight are provided, ipw is not used. Leave NULL if setting ipw = TRUE. Default = NULL
ipw	Logical: If TRUE, apply inverse probability weighting (for Classification only). Note: If weights are provided, ipw is not used. Default = TRUE
ipw.type	Integer 0, 1, 2 1: class.weights as in 0, divided by max(class.weights) 2: class.weights as in 0, divided by min(class.weights) Default = 2
upsample	Logical: If TRUE, upsample cases to balance outcome classes (for Classification only) Note: upsample will randomly sample with replacement if the length of the majority class is more than double the length of the class you are upsampling, thereby introducing randomness
downsample	Logical: If TRUE, downsample majority class to match size of minority class
resample.seed	Integer: If provided, will be used to set the seed during upsampling. Default = NULL (random seed)
max.leaves	Integer: Maximum number of terminal nodes to grow
nvmax	[gS] Integer: Number of max features to use for lin.type "allSubsets", "forwardStepwise", or "backwardStepwise". If values greater than n of features in x are provided, they will be excluded
lookback	Logical: If TRUE, check validation error to decide best number of leaves to use. Default = TRUE
gamma	Numeric: Soft weighting parameter. Weights of cases that do not belong to node get multiplied by this amount
lin.type	Character: See lincoef for options
single.lin.type	Character same options as lin.type, linear model to fit when max.leaves = 1
lambda	Float: lambda parameter for MASS::lm.ridge Default = .01
part.max.depth	Integer: Max depth for each tree model within the additive tree
init	Initial value. Default = mean(y)
verbose	Logical: If TRUE, print summary to screen.
plot.tuning	Logical: If TRUE, plot validation error during gridsearch
trace	Integer: If higher than 0, will print more information to the console. Default = 0
x.name	Character: Name for feature set
y.name	Character: Name for outcome
question	Character: the question you are attempting to answer with this model, in plain language.
outdir	Path to output directory. If defined, will save Predicted vs. True plot, if available, as well as full model output, if save.mod is TRUE

print.plot	Logical: if TRUE, produce plot using mplot3 Takes precedence over plot.fitted and plot.predicted. Default = TRUE
plot.fitted	Logical: if TRUE, plot True (y) vs Fitted
plot.predicted	Logical: if TRUE, plot True (y.test) vs Predicted. Requires x.test and y.test
plot.theme	Character: "zero", "dark", "box", "darkbox"
save.mod	Logical: If TRUE, save all output to an RDS file in outdir save.mod is TRUE by default if an outdir is defined. If set to TRUE, and no outdir is defined, outdir defaults to paste0("./s.", mod.name)
.gs	internal use only

Details

The Linear Additive Tree trains a tree using a sequence of regularized linear models and splits. We specify an upper threshold of leaves using max.leaves instead of directly defining a number, because depending on the other parameters and the datasets, splitting may stop early.

Author(s)

E.D. Gennatas

---

s_LINOA *Linear Optimized Additive Tree [C, R]*

Description

Train a Linear Optimized Additive Tree

Usage

```
s_LINOA(
  x,
  y = NULL,
  x.test = NULL,
  y.test = NULL,
  weights = NULL,
  ipw = TRUE,
  ipw.type = 2,
  upsample = FALSE,
  downsample = FALSE,
  resample.seed = NULL,
  max.leaves = 8,
  learning.rate = 0.5,
  select.leaves.smooth = TRUE,
  force.max.leaves = NULL,
  lookback = TRUE,
  gamma = 0,
  n.quantiles = 20,
  minobsinnode = NULL,
  minbucket = NULL,
 lin.type = c("forwardStepwise", "glmnet", "cv.glmnet", "lm.ridge", "allSubsets",
```

```
    "backwardStepwise", "glm", "solve", "none"),
  alpha = 1,
  lambda = 0.05,
  lambda.seq = NULL,
  cv.glmnet.nfolds = 5,
  which.cv.glmnet.lambda = "lambda.min",
  nbest = 1,
  nvmax = 3,
  .rho = TRUE,
  rho.max = 1000,
  init = NULL,
  metric = "auto",
  maximize = NULL,
  grid.resample.rtset = rtset.resample("kfold", 5),
  grid.search.type = "exhaustive",
  save.gridrun = FALSE,
  grid.verbose = verbose,
  keep.x = FALSE,
  simplify = TRUE,
  cxrcoef = FALSE,
  n.cores = rtCores,
  splitline.cores = 1,
  .preprocess = NULL,
  plot.tuning = TRUE,
  verbose.predict = FALSE,
  x.name = NULL,
  y.name = NULL,
  question = NULL,
  outdir = NULL,
  print.plot = TRUE,
  plot.fitted = NULL,
  plot.predicted = NULL,
  plot.theme = rtTheme,
  save.mod = FALSE,
  .gs = FALSE,
  verbose = TRUE,
  trace = 1
)
```

Arguments

x	Numeric vector or matrix / data frame of features i.e. independent variables
y	Numeric vector of outcome, i.e. dependent variable
x.test	Numeric vector or matrix / data frame of testing set features Columns must correspond to columns in x
y.test	Numeric vector of testing set outcome
weights	Numeric vector: Weights for cases. For classification, weights takes precedence over ipw, therefore set weights = NULL if using ipw. Note: If weight are provided, ipw is not used. Leave NULL if setting ipw = TRUE. Default = NULL
ipw	Logical: If TRUE, apply inverse probability weighting (for Classification only). Note: If weights are provided, ipw is not used. Default = TRUE

ipw.type	Integer 0, 1, 2 1: class.weights as in 0, divided by max(class.weights) 2: class.weights as in 0, divided by min(class.weights) Default = 2
upsample	Logical: If TRUE, upsample cases to balance outcome classes (for Classification only) Note: upsample will randomly sample with replacement if the length of the majority class is more than double the length of the class you are upsampling, thereby introducing randomness
downsample	Logical: If TRUE, downsample majority class to match size of minority class
resample.seed	Integer: If provided, will be used to set the seed during upsampling. Default = NULL (random seed)
max.leaves	Integer: Maximum number of terminal nodes to grow
lookback	Logical: If TRUE, check validation error to decide when to stop growing tree. Default = FALSE
minobsinnode	Integer: Minimum N observations needed in node, before considering splitting
lambda	Float: lambda parameter for MASS::lm.ridge Default = .01
nvmax	[gS] Integer: Number of max features to use for lin.type "allSubsets", "forwardStepwise", or "backwardStepwise". If values greater than n of features in x are provided, they will be excluded
init	Initial value. Default = mean(y)
plot.tuning	Logical: If TRUE, plot validation error during gridsearch
x.name	Character: Name for feature set
y.name	Character: Name for outcome
question	Character: the question you are attempting to answer with this model, in plain language.
outdir	Path to output directory. If defined, will save Predicted vs. True plot, if available, as well as full model output, if save.mod is TRUE
print.plot	Logical: if TRUE, produce plot using mplot3 Takes precedence over plot.fitted and plot.predicted. Default = TRUE
plot.fitted	Logical: if TRUE, plot True (y) vs Fitted
plot.predicted	Logical: if TRUE, plot True (y.test) vs Predicted. Requires x.test and y.test
plot.theme	Character: "zero", "dark", "box", "darkbox"
save.mod	Logical: If TRUE, save all output to an RDS file in outdir save.mod is TRUE by default if an outdir is defined. If set to TRUE, and no outdir is defined, outdir defaults to paste0("./s.", mod.name)
.gs	internal use only
verbose	Logical: If TRUE, print summary to screen.
trace	Integer: If higher than 0, will print more information to the console. Default = 0
part.max.depth	Integer: Max depth for each tree model within the additive tree

Details

The Linear Optimized Additive Tree grows a tree by finding splits that minimize loss after linear models are fit on each child. We specify an upper threshold of leaves using max.leaves instead of directly defining a number, because depending on the other parameters and the datasets, splitting may stop early.

Author(s)

E.D. Gennatas

s_LM *Linear model*

Description

Fit a linear model and validate it. Options include base `lm()`, robust linear model using `MASS:rlm()`, generalized least squares using `nlme::gls`, or polynomial regression using `stats::poly` to transform features

Usage

```
s_LM(
  x,
  y = NULL,
  x.test = NULL,
  y.test = NULL,
  x.name = NULL,
  y.name = NULL,
  weights = NULL,
  ipw = TRUE,
  ipw.type = 2,
  upsample = FALSE,
  downsample = FALSE,
  resample.seed = NULL,
  intercept = TRUE,
  robust = FALSE,
  gls = FALSE,
  polynomial = FALSE,
  poly.d = 3,
  poly.raw = FALSE,
  print.plot = TRUE,
  plot.fitted = NULL,
  plot.predicted = NULL,
  plot.theme = rtTheme,
  na.action = na.exclude,
  question = NULL,
  rtclass = NULL,
  verbose = TRUE,
  trace = 0,
  outdir = NULL,
  save.mod = ifelse(!is.null(outdir), TRUE, FALSE),
  ...
)
```

Arguments

x	Numeric vector or matrix / data frame of features i.e. independent variables
y	Numeric vector of outcome, i.e. dependent variable
x.test	Numeric vector or matrix / data frame of testing set features Columns must correspond to columns in x

y.test	Numeric vector of testing set outcome
x.name	Character: Name for feature set
y.name	Character: Name for outcome
weights	Numeric vector: Weights for cases. For classification, weights takes precedence over ipw, therefore set weights = NULL if using ipw. Note: If weight are provided, ipw is not used. Leave NULL if setting ipw = TRUE. Default = NULL
ipw	Logical: If TRUE, apply inverse probability weighting (for Classification only). Note: If weights are provided, ipw is not used. Default = TRUE
ipw.type	Integer 0, 1, 2 1: class.weights as in 0, divided by max(class.weights) 2: class.weights as in 0, divided by min(class.weights) Default = 2
upsample	Logical: If TRUE, upsample cases to balance outcome classes (for Classification only) Note: upsample will randomly sample with replacement if the length of the majority class is more than double the length of the class you are upsampling, thereby introducing randomness
downsample	Logical: If TRUE, downsample majority class to match size of minority class
resample.seed	Integer: If provided, will be used to set the seed during upsampling. Default = NULL (random seed)
intercept	Logical: If TRUE, fit an intercept term. Default = TRUE
robust	Logical: if TRUE, use MASS::rlm() instead of base lm()
gls	Logical: if TRUE, use nlme::gls
polynomial	Logical: if TRUE, run lm on poly(x, poly.d) (creates orthogonal polynomials)
poly.d	Integer: degree of polynomial
poly.raw	Logical: if TRUE, use raw polynomials. Default, which should not really be changed is FALSE
print.plot	Logical: if TRUE, produce plot using mplot3 Takes precedence over plot.fitted and plot.predicted. Default = TRUE
plot.fitted	Logical: if TRUE, plot True (y) vs Fitted
plot.predicted	Logical: if TRUE, plot True (y.test) vs Predicted. Requires x.test and y.test
plot.theme	Character: "zero", "dark", "box", "darkbox"
na.action	How to handle missing values. See ?na.fail
question	Character: the question you are attempting to answer with this model, in plain language.
rtclass	Character: Class type to use. "S3", "S4", "RC", "R6"
verbose	Logical: If TRUE, print summary to screen.
trace	Integer: If higher than 0, will print more information to the console. Default = 0
outdir	Path to output directory. If defined, will save Predicted vs. True plot, if available, as well as full model output, if save.mod is TRUE
save.mod	Logical. If TRUE, save all output as RDS file in outdir save.mod is TRUE by default if an outdir is defined. If set to TRUE, and no outdir is defined, outdir defaults to paste0("./s.", mod.name)
...	Additional arguments to be passed to MASS::rlm if robust = TRUE or MASS::lm.gls if gls = TRUE

Details

GLS can be useful in place of a standard linear model, when there is correlation among the residuals and/or they have unequal variances. Warning: `nlme`'s implementation is buggy, and `predict` will not work because of environment problems, which means it fails to get predicted values if `x.test` is provided. `robut = TRUE` trains a robust linear model using `MASS::rlm`. `gls = TRUE` trains a generalized least squares model using `nlme::gls`.

Value

rtMod

Author(s)

E.D. Gennatas

See Also

elevate for external cross-validation

Other Supervised Learning: `s_ADABOOST()`, `s_ADDTREE()`, `s_BART()`, `s_BAYESGLM()`, `s_BRUTO()`, `s_C50()`, `s_CART()`, `s_CTREE()`, `s_ET()`, `s_EVTREE()`, `s_GAM.default()`, `s_GAM.formula()`, `s_GAMSELX2()`, `s_GAMSELX()`, `s_GAMSEL()`, `s_GAM()`, `s_GBM0()`, `s_GBM3.R()`, `s_GBM()`, `s_GLMNET()`, `s_GLMTREE()`, `s_GLM()`, `s_GLS()`, `s_H2ODL()`, `s_H2OGBM()`, `s_H2ORF()`, `s_IRF()`, `s_KNN()`, `s_LDA()`, `s_LMTREE()`, `s_MARS()`, `s_MLRF()`, `s_NBAYES()`, `s_NLA()`, `s_NLS()`, `s_NW()`, `s_POLYMARS()`, `s_PPR()`, `s_PPTREE()`, `s_QDA()`, `s_QRNN()`, `s_RANGER()`, `s_RFSRC()`, `s_RF()`, `s_SDA()`, `s_SGD()`, `s_SPLS()`, `s_SVM()`, `s_TFN()`, `s_XGBLIN()`, `s_XGBOOST()`, `s_XGB()`

Examples

```
x <- rnorm(100)
y <- .6 * x + 12 + rnorm(100)/2
mod <- s_LM(x, y)
```

---

s_LMTREE *Linear Model Tree [R]*

Description

Train a LMTREE for regression or classification using `rpart`

Usage

```
s_LMTREE(
  x,
  y = NULL,
  x.test = NULL,
  y.test = NULL,
  x.name = NULL,
  y.name = NULL,
  weights = NULL,
  offset = NULL,
  ipw = TRUE,
```

```
  ipw.type = 2,
  upsample = FALSE,
  downsample = FALSE,
  resample.seed = NULL,
  na.action = na.exclude,
  print.plot = TRUE,
  plot.fitted = NULL,
  plot.predicted = NULL,
  plot.theme = rtTheme,
  question = NULL,
  verbose = TRUE,
  outdir = NULL,
  save.mod = ifelse(!is.null(outdir), TRUE, FALSE),
  ...
)
```

Arguments

x	Numeric vector or matrix / data frame of features i.e. independent variables
y	Numeric vector of outcome, i.e. dependent variable
x.test	Numeric vector or matrix / data frame of testing set features Columns must correspond to columns in x
y.test	Numeric vector of testing set outcome
x.name	Character: Name for feature set
y.name	Character: Name for outcome
weights	Numeric vector: Weights for cases. For classification, weights takes precedence over ipw, therefore set weights = NULL if using ipw. Note: If weight are provided, ipw is not used. Leave NULL if setting ipw = TRUE. Default = NULL
offset	Numeric vector of a priori known offsets
ipw	Logical: If TRUE, apply inverse probability weighting (for Classification only). Note: If weights are provided, ipw is not used. Default = TRUE
ipw.type	Integer 0, 1, 2 1: class.weights as in 0, divided by max(class.weights) 2: class.weights as in 0, divided by min(class.weights) Default = 2
upsample	Logical: If TRUE, upsample cases to balance outcome classes (for Classification only) Note: upsample will randomly sample with replacement if the length of the majority class is more than double the length of the class you are upsampling, thereby introducing randomness
downsample	Logical: If TRUE, downsample majority class to match size of minority class
resample.seed	Integer: If provided, will be used to set the seed during upsampling. Default = NULL (random seed)
na.action	How to handle missing values. See ?na.fail
print.plot	Logical: if TRUE, produce plot using mplot3 Takes precedence over plot.fitted and plot.predicted. Default = TRUE
plot.fitted	Logical: if TRUE, plot True (y) vs Fitted
plot.predicted	Logical: if TRUE, plot True (y.test) vs Predicted. Requires x.test and y.test
plot.theme	Character: "zero", "dark", "box", "darkbox"
question	Character: the question you are attempting to answer with this model, in plain language.

verbose	Logical: If TRUE, print summary to screen.
outdir	Path to output directory. If defined, will save Predicted vs. True plot, if available, as well as full model output, if save.mod is TRUE
save.mod	Logical: If TRUE, save all output to an RDS file in outdir save.mod is TRUE by default if an outdir is defined. If set to TRUE, and no outdir is defined, outdir defaults to paste0("./s.", mod.name)
...	Additional arguments passed to partykit::mob_control

Value

Object of class rtMod

Author(s)

E.D. Gennatas

See Also

elevate for external cross-validation

Other Supervised Learning: s_ADABOOST(), s_ADDTREE(), s_BART(), s_BAYESGLM(), s_BRUTO(), s_C50(), s_CART(), s_CTREE(), s_ET(), s_EVTREE(), s_GAM.default(), s_GAM.formula(), s_GAMSELX2(), s_GAMSELX(), s_GAMSEL(), s_GAM(), s_GBM0(), s_GBM3.R(), s_GBM(), s_GLMNET(), s_GLMTREE(), s_GLM(), s_GLS(), s_H2ODL(), s_H2OGBM(), s_H2ORF(), s_IRF(), s_KNN(), s_LDA(), s_LM(), s_MARS(), s_MLRF(), s_NBAYES(), s_NLA(), s_NLS(), s_NW(), s_POLYMARS(), s_PPR(), s_PPTREE(), s_QDA(), s_QRNN(), s_RANGER(), s_RFSRC(), s_RF(), s_SDA(), s_SGD(), s_SPLS(), s_SVM(), s_TFN(), s_XGBLIN(), s_XGBOOST(), s_XGB()

Other Tree-based methods: s_ADABOOST(), s_ADDTREE(), s_BART(), s_C50(), s_CART(), s_CTREE(), s_ET(), s_EVTREE(), s_GBM0(), s_GBM3.R(), s_GBM(), s_GLMTREE(), s_H2OGBM(), s_H2ORF(), s_IRF(), s_MLRF(), s_PPTREE(), s_RANGER(), s_RFSRC(), s_RF(), s_XGBOOST(), s_XGB()

Other Interpretable models: s_ADDTREE(), s_C50(), s_CART(), s_GLMNET(), s_GLMTREE(), s_GLM()

---

s_LOESS *Local Polynomial Regression (LOESS) [R]*

Description

Fits a LOESS curve or surface

Usage

```
s_LOESS(
  x,
  y = NULL,
  x.test = NULL,
  y.test = NULL,
  x.name = NULL,
  y.name = NULL,
  print.plot = TRUE,
  plot.fitted = NULL,
  plot.predicted = NULL,
```

```
  plot.theme = rtTheme,
  question = NULL,
  rtclass = NULL,
  verbose = TRUE,
  trace = 0,
  outdir = NULL,
  save.mod = ifelse(!is.null(outdir), TRUE, FALSE),
  ...
)
```

Arguments

x	Numeric vector or matrix / data frame of features i.e. independent variables
y	Numeric vector of outcome, i.e. dependent variable
x.test	Numeric vector or matrix / data frame of testing set features Columns must correspond to columns in x
y.test	Numeric vector of testing set outcome
x.name	Character: Name for feature set
y.name	Character: Name for outcome
print.plot	Logical: if TRUE, produce plot using mplot3 Takes precedence over plot.fitted and plot.predicted. Default = TRUE
plot.fitted	Logical: if TRUE, plot True (y) vs Fitted
plot.predicted	Logical: if TRUE, plot True (y.test) vs Predicted. Requires x.test and y.test
plot.theme	Character: "zero", "dark", "box", "darkbox"
question	Character: the question you are attempting to answer with this model, in plain language.
rtclass	Character: Class type to use. "S3", "S4", "RC", "R6"
verbose	Logical: If TRUE, print summary to screen.
trace	Integer: If higher than 0, will print more information to the console. Default = 0
outdir	Path to output directory. If defined, will save Predicted vs. True plot, if available, as well as full model output, if save.mod is TRUE
save.mod	Logical: If TRUE, save all output to an RDS file in outdir save.mod is TRUE by default if an outdir is defined. If set to TRUE, and no outdir is defined, outdir defaults to paste0("./s.", mod.name)
...	Additional arguments to loess

Details

A maximum of 4 features are allowed in this implementation (stats::loess) The main use for this algorithm would be fitting curves in bivariate plots, where GAM or similar is preferable anyway. It is included in **rtemis** mainly for academic purposes - not for building predictive models.

Value

Object of class **rtemis**

Author(s)

E.D. Gennatas

See Also

elevate

`s_LOGISTIC` *Logistic Regression*

Description

Convenience alias for `s_GLM(family = binomial(link = "logit"))`.

Usage

```
s_LOGISTIC(
  x,
  y,
  x.test = NULL,
  y.test = NULL,
  family = binomial(link = "logit"),
  ...
)
```

Arguments

`x`	Numeric vector or matrix / data frame of features i.e. independent variables
`y`	Numeric vector of outcome, i.e. dependent variable
`x.test`	Numeric vector or matrix / data frame of testing set features Columns must correspond to columns in `x`
`y.test`	Numeric vector of testing set outcome
`family`	Error distribution and link function. See `stats::family`
`...`	Additional arguments

`s_MARS` *Multivariate adaptive regression splines (MARS) [C, R]*

Description

Trains a MARS model using `earth::earth`. [gS] in Arguments description indicates that hyperparameter will be tuned if more than one value are provided For more info on algorithm hyperparameters, see `?earth::earth`

Usage

```
s_MARS(
  x,
  y = NULL,
  x.test = NULL,
  y.test = NULL,
  x.name = NULL,
  y.name = NULL,
  grid.resample.rtset = rtset.grid.resample(),
  weights = NULL,
  ipw = TRUE,
  ipw.type = 2,
  upsample = FALSE,
  downsample = FALSE,
  resample.seed = NULL,
  glm = NULL,
  degree = 2,
  penalty = 3,
  nk = NULL,
  thresh = 0,
  minspan = 0,
  endspan = 0,
  newvar.penalty = 0,
  fast.k = 2,
  fast.beta = 1,
  linpreds = FALSE,
  pmethod = "forward",
  nprune = NULL,
  nfold = 4,
  ncross = 1,
  stratify = TRUE,
  wp = NULL,
  na.action = na.fail,
  metric = NULL,
  maximize = NULL,
  n.cores = rtCores,
  print.plot = TRUE,
  plot.fitted = NULL,
  plot.predicted = NULL,
  plot.theme = rtTheme,
  question = NULL,
  verbose = TRUE,
  trace = 0,
  save.mod = FALSE,
  outdir = NULL,
  ...
)
```

Arguments

x	Numeric vector or matrix of features, i.e. independent variables
y	Numeric vector of outcome, i.e. dependent variable

x.test	(Optional) Numeric vector or matrix of validation set features must have set of columns as x
y.test	(Optional) Numeric vector of validation set outcomes
degree	[gS] Integer: Maximum degree of interaction. Default = 2
penalty	[gS] Float: GCV penalty per knot. 0 penalizes only terms, not knots. -1 means no penalty. Default = 3
nk	[gS] Integer: Maximum number of terms created by the forward pass. See earth::earth
pmethod	[gS] Character: Pruning method: "backward", "none", "exhaustive", "forward", "seqrep", "cv". Default = "forward"
nprune	[gS] Integer: Max N of terms (incl. intercept) in the pruned model
...	Additional parameters to pass to earth::earth

Value

Object of class rtMod

Author(s)

E.D. Gennatas

See Also

elevate for external cross-validation

Other Supervised Learning: s_ADABOOST(), s_ADDTREE(), s_BART(), s_BAYESGLM(), s_BRUTO(), s_C50(), s_CART(), s_CTREE(), s_ET(), s_EVTREE(), s_GAM.default(), s_GAM.formula(), s_GAMSELX2(), s_GAMSELX(), s_GAMSEL(), s_GAM(), s_GBM0(), s_GBM3.R(), s_GBM(), s_GLMNET(), s_GLMTREE(), s_GLM(), s_GLS(), s_H2ODL(), s_H2OGBM(), s_H2ORF(), s_IRF(), s_KNN(), s_LDA(), s_LMTREE(), s_LM(), s_MLRF(), s_NBAYES(), s_NLA(), s_NLS(), s_NW(), s_POLYMARS(), s_PPR(), s_PPTREE(), s_QDA(), s_QRNN(), s_RANGER(), s_RFSRC(), s_RF(), s_SDA(), s_SGD(), s_SPLS(), s_SVM(), s_TFN(), s_XGBLIN(), s_XGBOOST(), s_XGB()

---

s_MLRF *Spark MLlib Random Forest [C, R]*

Description

Train an MLlib Random Forest model on Spark

Usage

```
s_MLRF(
  x,
  y = NULL,
  x.test = NULL,
  y.test = NULL,
  upsample = FALSE,
  downsample = FALSE,
  resample.seed = NULL,
```

```
  n.trees = 500,
  max.depth = 30L,
  subsampling.rate = 1,
  min.instances.per.node = 1,
  feature.subset.strategy = "auto",
  max.bins = 32L,
  x.name = NULL,
  y.name = NULL,
  spark.master = "local",
  print.plot = TRUE,
  plot.fitted = NULL,
  plot.predicted = NULL,
  plot.theme = rtTheme,
  question = NULL,
  verbose = TRUE,
  trace = 0,
  outdir = NULL,
  save.mod = ifelse(!is.null(outdir), TRUE, FALSE),
  ...
)
```

Arguments

x	vector, matrix or dataframe of training set features
y	vector of outcomes
x.test	vector, matrix or dataframe of testing set features
y.test	vector of testing set outcomes
upsample	Logical: If TRUE, upsample cases to balance outcome classes (for Classification only) Note: upsample will randomly sample with replacement if the length of the majority class is more than double the length of the class you are upsampling, thereby introducing randomness
downsample	Logical: If TRUE, downsample majority class to match size of minority class
resample.seed	Integer: If provided, will be used to set the seed during upsampling. Default = NULL (random seed)
n.trees	Integer. Number of trees to train
max.depth	Integer. Max depth of each tree
max.bins	Integer. Max N of bins used for discretizing continuous features and for choosing how to split on features at each node. More bins give higher granularity.
x.name	Character: Name for feature set
y.name	Character: Name for outcome
spark.master	Spark cluster URL or "local"
print.plot	Logical: if TRUE, produce plot using mplot3 Takes precedence over plot.fitted and plot.predicted. Default = TRUE
plot.fitted	Logical: if TRUE, plot True (y) vs Fitted
plot.predicted	Logical: if TRUE, plot True (y.test) vs Predicted. Requires x.test and y.test
plot.theme	Character: "zero", "dark", "box", "darkbox"
question	Character: the question you are attempting to answer with this model, in plain language.

verbose	Logical: If TRUE, print summary to screen.
trace	Integer: If higher than 0, will print more information to the console. Default = 0
outdir	Path to output directory. If defined, will save Predicted vs. True plot, if available, as well as full model output, if save.mod is TRUE
save.mod	Logical: If TRUE, save all output to an RDS file in outdir save.mod is TRUE by default if an outdir is defined. If set to TRUE, and no outdir is defined, outdir defaults to paste0("./s.", mod.name)
...	Additional arguments
type	"regression" for continuous outcome; "classification" for categorical outcome. "auto" will result in regression for numeric y and classification otherwise

Details

The overhead incurred by Spark means this should be used only for really large datasets on a Spark cluster, not on a regular local machine.

Value

rtMod object

Author(s)

E.D. Gennatas

See Also

elevate for external cross-validation

Other Supervised Learning: s_ADABOOST(), s_ADDTREE(), s_BART(), s_BAYESGLM(), s_BRUTO(), s_C50(), s_CART(), s_CTREE(), s_ET(), s_EVTREE(), s_GAM.default(), s_GAM.formula(), s_GAMSELX2(), s_GAMSELX(), s_GAMSEL(), s_GAM(), s_GBM0(), s_GBM3.R(), s_GBM(), s_GLMNET(), s_GLMTREE(), s_GLM(), s_GLS(), s_H2ODL(), s_H2OGBM(), s_H2ORF(), s_IRF(), s_KNN(), s_LDA(), s_LMTREE(), s_LM(), s_MARS(), s_NBAYES(), s_NLA(), s_NLS(), s_NW(), s_POLYMARS(), s_PPR(), s_PPTREE(), s_QDA(), s_QRNN(), s_RANGER(), s_RFSRC(), s_RF(), s_SDA(), s_SGD(), s_SPLS(), s_SVM(), s_TFN(), s_XGBLIN(), s_XGBOOST(), s_XGB()

Other Tree-based methods: s_ADABOOST(), s_ADDTREE(), s_BART(), s_C50(), s_CART(), s_CTREE(), s_ET(), s_EVTREE(), s_GBM0(), s_GBM3.R(), s_GBM(), s_GLMTREE(), s_H2OGBM(), s_H2ORF(), s_IRF(), s_LMTREE(), s_PPTREE(), s_RANGER(), s_RFSRC(), s_RF(), s_XGBOOST(), s_XGB()

---

s_MULTINOM *Multinomial Logistic Regression*

Description

Convenience alias for s_GLM(class.method = "multinom").

Usage

```
s_MULTINOM(x, y, x.test = NULL, y.test = NULL, class.method = "multinom", ...)
```

Arguments

x	Numeric vector or matrix / data frame of features i.e. independent variables
y	Numeric vector of outcome, i.e. dependent variable
x.test	Numeric vector or matrix / data frame of testing set features Columns must correspond to columns in x
y.test	Numeric vector of testing set outcome
class.method	Character: Define "logistic" or "multinom" for classification. The only purpose of this is so you can try nnet::multinom instead of glm for binary classification
...	Additional arguments

---

s_NBAYES *Naive Bayes Classifier [C]*

Description

Train a Naive Bayes Classifier using e1071::naiveBayes

Usage

```
s_NBAYES(
  x,
  y = NULL,
  x.test = NULL,
  y.test = NULL,
  laplace = 0,
  x.name = NULL,
  y.name = NULL,
  print.plot = TRUE,
  plot.fitted = NULL,
  plot.predicted = NULL,
  plot.theme = rtTheme,
  question = NULL,
  rtclass = NULL,
  verbose = TRUE,
  outdir = NULL,
  save.mod = ifelse(!is.null(outdir), TRUE, FALSE),
  ...
)
```

Arguments

x	Numeric vector or matrix / data frame of features i.e. independent variables
y	Numeric vector of outcome, i.e. dependent variable
x.test	Numeric vector or matrix / data frame of testing set features Columns must correspond to columns in x
y.test	Numeric vector of testing set outcome
laplace	Float (>0): Laplace smoothing. Default = 0 (no smoothing). This only affects categorical features

x.name	Character: Name for feature set
y.name	Character: Name for outcome
print.plot	Logical: if TRUE, produce plot using mplot3 Takes precedence over plot.fitted and plot.predicted. Default = TRUE
plot.fitted	Logical: if TRUE, plot True (y) vs Fitted
plot.predicted	Logical: if TRUE, plot True (y.test) vs Predicted. Requires x.test and y.test
plot.theme	Character: "zero", "dark", "box", "darkbox"
question	Character: the question you are attempting to answer with this model, in plain language.
rtclass	Character: Class type to use. "S3", "S4", "RC", "R6"
verbose	Logical: If TRUE, print summary to screen.
outdir	Path to output directory. If defined, will save Predicted vs. True plot, if available, as well as full model output, if save.mod is TRUE
save.mod	Logical: If TRUE, save all output to an RDS file in outdir save.mod is TRUE by default if an outdir is defined. If set to TRUE, and no outdir is defined, outdir defaults to paste0("./s.", mod.name)
...	Additional arguments

Details

The laplace argument only affects categorical predictors

Value

rtMod object

Author(s)

E.D. Gennatas

See Also

Other Supervised Learning: s_ADABOOST(), s_ADDTREE(), s_BART(), s_BAYESGLM(), s_BRUTO(), s_C50(), s_CART(), s_CTREE(), s_ET(), s_EVTREE(), s_GAM.default(), s_GAM.formula(), s_GAMSELX2(), s_GAMSELX(), s_GAMSEL(), s_GAM(), s_GBM0(), s_GBM3.R(), s_GBM(), s_GLMNET(), s_GLMTREE(), s_GLM(), s_GLS(), s_H2ODL(), s_H2OGBM(), s_H2ORF(), s_IRF(), s_KNN(), s_LDA(), s_LMTREE(), s_LM(), s_MARS(), s_MLRF(), s_NLA(), s_NLS(), s_NW(), s_POLYMARS(), s_PPR(), s_PPTREE(), s_QDA(), s_QRNN(), s_RANGER(), s_RFSRC(), s_RF(), s_SDA(), s_SGD(), s_SPLS(), s_SVM(), s_TFN(), s_XGBLIN(), s_XGBOOST(), s_XGB()

s_NLA *NonLinear Activation unit Regression (NLA) [R]*

Description

Train an equivalent of a 1 hidden unit neural network with a defined nonlinear activation function using optim

Usage

```
s_NLA(
  x,
  y = NULL,
  x.test = NULL,
  y.test = NULL,
  activation = softplus,
  b_o = mean(y),
  W_o = 1,
  b_h = 0,
  W_h = 0.01,
  optim.method = "BFGS",
  control = list(),
  lower = -Inf,
  upper = Inf,
  x.name = NULL,
  y.name = NULL,
  save.func = TRUE,
  print.plot = TRUE,
  plot.fitted = NULL,
  plot.predicted = NULL,
  plot.theme = rtTheme,
  question = NULL,
  rtclass = NULL,
  verbose = TRUE,
  trace = 0,
  outdir = NULL,
  save.mod = ifelse(!is.null(outdir), TRUE, FALSE),
  ...
)
```

Arguments

x	Numeric vector or matrix / data frame of features i.e. independent variables
y	Numeric vector of outcome, i.e. dependent variable
x.test	Numeric vector or matrix / data frame of testing set features Columns must correspond to columns in x
y.test	Numeric vector of testing set outcome
activation	Function: Activation function to use. Default = softplus
b_o	Float, vector (length y): Output bias. Defaults to mean(y)

W_o	Float: Output weight. Defaults to 1
b_h	Float: Hidden layer bias. Defaults to 0
W_h	Float, vector (length NCOL(x)): Hidden layer weights. Defaults to 0
optim.method	Character: Optimization method to use: "Nelder-Mead", "BFGS", "CG", "L-BFGS-B", "SANN", "Brent". See stats::optim for more details. Default = "BFGS"
x.name	Character: Name for feature set
y.name	Character: Name for outcome
print.plot	Logical: if TRUE, produce plot using mplot3 Takes precedence over plot.fitted and plot.predicted. Default = TRUE
plot.fitted	Logical: if TRUE, plot True (y) vs Fitted
plot.predicted	Logical: if TRUE, plot True (y.test) vs Predicted. Requires x.test and y.test
plot.theme	Character: "zero", "dark", "box", "darkbox"
question	Character: the question you are attempting to answer with this model, in plain language.
rtclass	Character: Class type to use. "S3", "S4", "RC", "R6"
verbose	Logical: If TRUE, print summary to screen.
trace	Integer: If higher than 0, will print more information to the console. Default = 0
outdir	Path to output directory. If defined, will save Predicted vs. True plot, if available, as well as full model output, if save.mod is TRUE
save.mod	Logical: If TRUE, save all output to an RDS file in outdir save.mod is TRUE by default if an outdir is defined. If set to TRUE, and no outdir is defined, outdir defaults to paste0("./s.", mod.name)
...	Additional arguments to be passed to sigreg

Details

Since we are using optim, results will be sensitive to the combination of optimizer method (See optim::method for details), initialization values, and activation function.

Value

Object of class **rtemis**

Author(s)

E.D. Gennatas

See Also

elevate for external cross-validation

Other Supervised Learning: s_ADABOOST(), s_ADDTREE(), s_BART(), s_BAYESGLM(), s_BRUTO(), s_C50(), s_CART(), s_CTREE(), s_ET(), s_EVTREE(), s_GAM.default(), s_GAM.formula(), s_GAMSELX2(), s_GAMSELX(), s_GAMSEL(), s_GAM(), s_GBM0(), s_GBM3.R(), s_GBM(), s_GLMNET(), s_GLMTREE(), s_GLM(), s_GLS(), s_H2ODL(), s_H2OGBM(), s_H2ORF(), s_IRF(), s_KNN(), s_LDA(), s_LMTREE(), s_LM(), s_MARS(), s_MLRF(), s_NBAYES(), s_NLS(), s_NW(), s_POLYMARS(), s_PPR(), s_PPTREE(), s_QDA(), s_QRNN(), s_RANGER(), s_RFSRC(), s_RF(), s_SDA(), s_SGD(), s_SPLS(), s_SVM(), s_TFN(), s_XGBLIN(), s_XGBOOST(), s_XGB()

s_NLS *Nonlinear Least Squares (NLS) [R]*

Description

Build a NLS model

Usage

```
s_NLS(
  x,
  y = NULL,
  x.test = NULL,
  y.test = NULL,
  formula = NULL,
  weights = NULL,
  start = NULL,
  control = nls.control(maxiter = 200),
  .type = NULL,
  default.start = 0.1,
  algorithm = "default",
  nls.trace = FALSE,
  x.name = NULL,
  y.name = NULL,
  save.func = TRUE,
  print.plot = TRUE,
  plot.fitted = NULL,
  plot.predicted = NULL,
  plot.theme = rtTheme,
  question = NULL,
  rtclass = NULL,
  verbose = TRUE,
  trace = 0,
  outdir = NULL,
  save.mod = ifelse(!is.null(outdir), TRUE, FALSE),
  ...
)
```

Arguments

x	Numeric vector or matrix / data frame of features i.e. independent variables
y	Numeric vector of outcome, i.e. dependent variable
x.test	Numeric vector or matrix / data frame of testing set features Columns must correspond to columns in x
y.test	Numeric vector of testing set outcome
weights	Numeric vector: Weights for cases. For classification, weights takes precedence over ipw, therefore set weights = NULL if using ipw. Note: If weight are provided, ipw is not used. Leave NULL if setting ipw = TRUE. Default = NULL
x.name	Character: Name for feature set

y.name	Character: Name for outcome
print.plot	Logical: if TRUE, produce plot using mplot3 Takes precedence over plot.fitted and plot.predicted. Default = TRUE
plot.fitted	Logical: if TRUE, plot True (y) vs Fitted
plot.predicted	Logical: if TRUE, plot True (y.test) vs Predicted. Requires x.test and y.test
plot.theme	Character: "zero", "dark", "box", "darkbox"
question	Character: the question you are attempting to answer with this model, in plain language.
rtclass	Character: Class type to use. "S3", "S4", "RC", "R6"
verbose	Logical: If TRUE, print summary to screen.
trace	Integer: If higher than 0, will print more information to the console. Default = 0
outdir	Path to output directory. If defined, will save Predicted vs. True plot, if available, as well as full model output, if save.mod is TRUE
save.mod	Logical: If TRUE, save all output to an RDS file in outdir save.mod is TRUE by default if an outdir is defined. If set to TRUE, and no outdir is defined, outdir defaults to paste0("./s.", mod.name)
...	Additional arguments to be passed to nls

Value

Object of class **rtemis**

Author(s)

E.D. Gennatas

See Also

elevate for external cross-validation

Other Supervised Learning: s_ADABOOST(), s_ADDTREE(), s_BART(), s_BAYESGLM(), s_BRUTO(), s_C50(), s_CART(), s_CTREE(), s_ET(), s_EVTREE(), s_GAM.default(), s_GAM.formula(), s_GAMSELX2(), s_GAMSELX(), s_GAMSEL(), s_GAM(), s_GBM0(), s_GBM3.R(), s_GBM(), s_GLMNET(), s_GLMTREE(), s_GLM(), s_GLS(), s_H2ODL(), s_H2OGBM(), s_H2ORF(), s_IRF(), s_KNN(), s_LDA(), s_LMTREE(), s_LM(), s_MARS(), s_MLRF(), s_NBAYES(), s_NLA(), s_NW(), s_POLYMARS(), s_PPR(), s_PPTREE(), s_QDA(), s_QRNN(), s_RANGER(), s_RFSRC(), s_RF(), s_SDA(), s_SGD(), s_SPLS(), s_SVM(), s_TFN(), s_XGBLIN(), s_XGBOOST(), s_XGB()

---

s_NW *Nadaraya-Watson kernel regression [R]*

Description

Computes a kernel regression estimate using np::npreg()

Usage

```
s_NW(
  x,
  y = NULL,
  x.test = NULL,
  y.test = NULL,
  x.name = NULL,
  y.name = NULL,
  bw = NULL,
  plot.bw = FALSE,
  print.plot = TRUE,
  plot.fitted = NULL,
  plot.predicted = NULL,
  plot.theme = rtTheme,
  question = NULL,
  rtclass = NULL,
  verbose = TRUE,
  trace = 0,
  outdir = NULL,
  save.mod = ifelse(!is.null(outdir), TRUE, FALSE),
  ...
)
```

Arguments

x	Numeric vector or matrix / data frame of features i.e. independent variables
y	Numeric vector of outcome, i.e. dependent variable
x.test	Numeric vector or matrix / data frame of testing set features Columns must correspond to columns in x
y.test	Numeric vector of testing set outcome
x.name	Character: Name for feature set
y.name	Character: Name for outcome
bw	Bandwidth as calculate by np::npregbw. Default = NULL, in which case np::npregbw will be run
plot.bw	Logical. Plot bandwidth selector results
print.plot	Logical: if TRUE, produce plot using mplot3 Takes precedence over plot.fitted and plot.predicted. Default = TRUE
plot.fitted	Logical: if TRUE, plot True (y) vs Fitted
plot.predicted	Logical: if TRUE, plot True (y.test) vs Predicted. Requires x.test and y.test
plot.theme	Character: "zero", "dark", "box", "darkbox"
question	Character: the question you are attempting to answer with this model, in plain language.
rtclass	Character: Class type to use. "S3", "S4", "RC", "R6"
verbose	Logical: If TRUE, print summary to screen.
trace	Integer: If higher than 0, will print more information to the console. Default = 0
outdir	Path to output directory. If defined, will save Predicted vs. True plot, if available, as well as full model output, if save.mod is TRUE

save.mod	Logical: If TRUE, save all output to an RDS file in outdir save.mod is TRUE by default if an outdir is defined. If set to TRUE, and no outdir is defined, outdir defaults to paste0("./s.", mod.name)
...	Additional parameters to be passed to npreg

Details

np::npreg allows inputs with mixed data types. NW automatically models interactions, like PPR, but the latter is a lot faster

Value

Object of class **rtemis**

Author(s)

E.D. Gennatas

See Also

elevate for external cross-validation

Other Supervised Learning: s_ADABOOST(), s_ADDTREE(), s_BART(), s_BAYESGLM(), s_BRUTO(), s_C50(), s_CART(), s_CTREE(), s_ET(), s_EVTREE(), s_GAM.default(), s_GAM.formula(), s_GAMSELX2(), s_GAMSELX(), s_GAMSEL(), s_GAM(), s_GBM0(), s_GBM3.R(), s_GBM(), s_GLMNET(), s_GLMTREE(), s_GLM(), s_GLS(), s_H2ODL(), s_H2OGBM(), s_H2ORF(), s_IRF(), s_KNN(), s_LDA(), s_LMTREE(), s_LM(), s_MARS(), s_MLRF(), s_NBAYES(), s_NLA(), s_NLS(), s_POLYMARS(), s_PPR(), s_PPTREE(), s_QDA(), s_QRNN(), s_RANGER(), s_RFSRC(), s_RF(), s_SDA(), s_SGD(), s_SPLS(), s_SVM(), s_TFN(), s_XGBLIN(), s_XGBOOST(), s_XGB()

Examples

```
## Not run:
x <- rnorm(100)
y <- .6 * x + 12 + rnorm(100)
mod <- s_NW(x, y)
## End(Not run)
```

---

s_POLY *Polynomial Regression*

Description

Convenience alias for s_GLM(polynomial = T). Substitutes all features with poly(x, poly.d)

Usage

```
s_POLY(x, y, x.test = NULL, y.test = NULL, poly.d = 3, poly.raw = FALSE, ...)
```

Arguments

x	Numeric vector or matrix / data frame of features i.e. independent variables
y	Numeric vector of outcome, i.e. dependent variable
x.test	Numeric vector or matrix / data frame of testing set features Columns must correspond to columns in x
y.test	Numeric vector of testing set outcome
poly.d	Integer: degree of polynomial(s) to use
poly.raw	Logical: if TRUE, use raw polynomials. Defaults to FALSE, resulting in orthogonal polynomials. See stats::poly
...	Additional arguments

---

s_POLYMARS *Multivariate adaptive polynomial spline regression (POLYMARS) [C, R]*

Description

Trains a POLYMARS model using polspline::polymars and validates it

Usage

```
s_POLYMARS(
  x,
  y = NULL,
  x.test = NULL,
  y.test = NULL,
  x.name = NULL,
  y.name = NULL,
  grid.resample.rtset = rtset.grid.resample(),
  bag.resample.rtset = NULL,
  weights = NULL,
  ipw = TRUE,
  ipw.type = 2,
  upsample = FALSE,
  downsample = FALSE,
  resample.seed = NULL,
  maxsize = ceiling(min(6 * (nrow(x)^{     1/3 }), nrow(x)/4, 100)),
  classify = NULL,
  n.cores = rtCores,
  print.plot = TRUE,
  plot.fitted = NULL,
  plot.predicted = NULL,
  plot.theme = rtTheme,
  question = NULL,
  verbose = TRUE,
  trace = 0,
  save.mod = FALSE,
  outdir = NULL,
  ...
)
```

Arguments

x	Numeric vector or matrix of features, i.e. independent variables
y	Numeric vector of outcome, i.e. dependent variable
x.test	(Optional) Numeric vector or matrix of validation set features must have set of columns as x
y.test	(Optional) Numeric vector of validation set outcomes
x.name	Character: Name for feature set
y.name	Character: Name for outcome
grid.resample.rtset	List: Output of rtset.resample defining gridSearchLearn parameters.
weights	Numeric vector: Weights for cases. For classification, weights takes precedence over ipw, therefore set weights = NULL if using ipw. Note: If weight are provided, ipw is not used. Leave NULL if setting ipw = TRUE. Default = NULL
ipw	Logical: If TRUE, apply inverse probability weighting (for Classification only). Note: If weights are provided, ipw is not used. Default = TRUE
ipw.type	Integer 0, 1, 2 1: class.weights as in 0, divided by max(class.weights) 2: class.weights as in 0, divided by min(class.weights) Default = 2
upsample	Logical: If TRUE, upsample cases to balance outcome classes (for Classification only) Note: upsample will randomly sample with replacement if the length of the majority class is more than double the length of the class you are upsampling, thereby introducing randomness
downsample	Logical: If TRUE, downsample majority class to match size of minority class
resample.seed	Integer: If provided, will be used to set the seed during upsampling. Default = NULL (random seed)
n.cores	Integer: Number of cores to use.
print.plot	Logical: if TRUE, produce plot using mplot3 Takes precedence over plot.fitted and plot.predicted. Default = TRUE
plot.fitted	Logical: if TRUE, plot True (y) vs Fitted
plot.predicted	Logical: if TRUE, plot True (y.test) vs Predicted. Requires x.test and y.test
plot.theme	Character: "zero", "dark", "box", "darkbox"
question	Character: the question you are attempting to answer with this model, in plain language.
verbose	Logical: If TRUE, print summary to screen.
save.mod	Logical: If TRUE, save all output to an RDS file in outdir save.mod is TRUE by default if an outdir is defined. If set to TRUE, and no outdir is defined, outdir defaults to paste0("./s.", mod.name)
outdir	Path to output directory. If defined, will save Predicted vs. True plot, if available, as well as full model output, if save.mod is TRUE
...	Additional parameters to pass to polspline::polymars

Value

Object of class rtMod

Author(s)

E.D. Gennatas

See Also

elevate for external cross-validation

Other Supervised Learning: s_ADABOOST(), s_ADDTREE(), s_BART(), s_BAYESGLM(), s_BRUTO(), s_C50(), s_CART(), s_CTREE(), s_ET(), s_EVTREE(), s_GAM.default(), s_GAM.formula(), s_GAMSELX2(), s_GAMSELX(), s_GAMSEL(), s_GAM(), s_GBM0(), s_GBM3.R(), s_GBM(), s_GLMNET(), s_GLMTREE(), s_GLM(), s_GLS(), s_H2ODL(), s_H2OGBM(), s_H2ORF(), s_IRF(), s_KNN(), s_LDA(), s_LMTREE(), s_LM(), s_MARS(), s_MLRF(), s_NBAYES(), s_NLA(), s_NLS(), s_NW(), s_PPR(), s_PPTREE(), s_QDA(), s_QRNN(), s_RANGER(), s_RFSRC(), s_RF(), s_SDA(), s_SGD(), s_SPLS(), s_SVM(), s_TFN(), s_XGBLIN(), s_XGBOOST(), s_XGB()

---

s_PPR *Projection Pursuit Regression (PPR) [R]*

Description

Train a Projection Pursuit Regression model

Usage

```
s_PPR(
  x,
  y = NULL,
  x.test = NULL,
  y.test = NULL,
  x.name = NULL,
  y.name = NULL,
  grid.resample.rtset = rtset.grid.resample(),
  grid.search.type = c("exhaustive", "randomized"),
  grid.search.randomized.p = 0.1,
  weights = NULL,
  nterms = NULL,
  max.terms = nterms,
  optlevel = 3,
  sm.method = "spline",
  bass = 0,
  span = 0,
  df = 5,
  gcvpen = 1,
  metric = "MSE",
  maximize = FALSE,
  n.cores = rtCores,
  print.plot = TRUE,
  plot.fitted = NULL,
  plot.predicted = NULL,
  plot.theme = rtTheme,
  question = NULL,
  rtclass = NULL,
  verbose = TRUE,
  trace = 0,
  outdir = NULL,
```

```
  save.mod = ifelse(!is.null(outdir), TRUE, FALSE),
  ...
)
```

Arguments

x	Numeric vector or matrix / data frame of features i.e. independent variables
y	Numeric vector of outcome, i.e. dependent variable
x.test	Numeric vector or matrix / data frame of testing set features Columns must correspond to columns in x
y.test	Numeric vector of testing set outcome
x.name	Character: Name for feature set
y.name	Character: Name for outcome
grid.resample.rtset	List: Output of rtset.resample defining gridSearchLearn parameters.
grid.search.type	Character: Type of grid search to perform: "exhaustive" or "randomized".
weights	Numeric vector: Weights for cases. For classification, weights takes precedence over ipw, therefore set weights = NULL if using ipw. Note: If weight are provided, ipw is not used. Leave NULL if setting ipw = TRUE. Default = NULL
nterms	[gS] Integer: number of terms to include in the final model
optlevel	[gS] Integer [0, 3]: optimization level (Default = 3). See Details in stats::ppr
sm.method	[gS] Character: "supsmu", "spline", or "gcvspline". Smoothing method. Default = "spline"
bass	[gS] Numeric [0, 10]: for sm.method = "supsmu": larger values result in greater smoother (Default = 5). See stats::ppr
span	[gS] Numeric [0, 1]: for sm.method = "supsmu": 0 (Default) results in automatic span selection by local crossvalidation. See stats::ppr
df	[gS] Numeric: for sm.method = "spline": Specify smoothness of each ridge term. See stats::ppr
gcvpen	[gs] Numeric: for sm.method = "gcvspline": Penalty used in the GCV selection for each degree of freedom used. Higher values result in greater smoothing. See stats::ppr Default = 5
metric	Character: Metric to minimize, or maximize if maximize = TRUE during grid search. Default = NULL, which results in "Balanced Accuracy" for Classification, "MSE" for Regression, and "Coherence" for Survival Analysis.
maximize	Logical: If TRUE, metric will be maximized if grid search is run.
n.cores	Integer: Number of cores to use.
print.plot	Logical: if TRUE, produce plot using mplot3 Takes precedence over plot.fitted and plot.predicted. Default = TRUE
plot.fitted	Logical: if TRUE, plot True (y) vs Fitted
plot.predicted	Logical: if TRUE, plot True (y.test) vs Predicted. Requires x.test and y.test
plot.theme	Character: "zero", "dark", "box", "darkbox"
question	Character: the question you are attempting to answer with this model, in plain language.
verbose	Logical: If TRUE, print summary to screen.

outdir	Path to output directory. If defined, will save Predicted vs. True plot, if available, as well as full model output, if save.mod is TRUE
save.mod	Logical: If TRUE, save all output to an RDS file in outdir save.mod is TRUE by default if an outdir is defined. If set to TRUE, and no outdir is defined, outdir defaults to paste0("./s.", mod.name)
...	Additional arguments to be passed to ppr

Details

[gS]: If more than one value is passed, parameter tuning via grid search will be performed on resamples of the training set prior to training model on full training set Interactions: PPR automatically models interactions, no need to specify them

Value

Object of class **rtemis**

Author(s)

E.D. Gennatas

See Also

elevate for external cross-validation

Other Supervised Learning: s_ADABOOST(), s_ADDTREE(), s_BART(), s_BAYESGLM(), s_BRUTO(), s_C50(), s_CART(), s_CTREE(), s_ET(), s_EVTREE(), s_GAM.default(), s_GAM.formula(), s_GAMSELX2(), s_GAMSELX(), s_GAMSEL(), s_GAM(), s_GBM0(), s_GBM3.R(), s_GBM(), s_GLMNET(), s_GLMTREE(), s_GLM(), s_GLS(), s_H2ODL(), s_H2OGBM(), s_H2ORF(), s_IRF(), s_KNN(), s_LDA(), s_LMTREE(), s_LM(), s_MARS(), s_MLRF(), s_NBAYES(), s_NLA(), s_NLS(), s_NW(), s_POLYMARS(), s_PPTREE(), s_QDA(), s_QRNN(), s_RANGER(), s_RFSRC(), s_RF(), s_SDA(), s_SGD(), s_SPLS(), s_SVM(), s_TFN(), s_XGBLIN(), s_XGBOOST(), s_XGB()

---

s_PPTREE *Projection Pursuit Tree Classification [C]*

Description

Train a PPTREE Classifier using PPtree::PP.Tree

Usage

```
s_PPTREE(
  x,
  y = NULL,
  x.test = NULL,
  y.test = NULL,
  PPmethod = "LDA",
  weight = TRUE,
  r = NULL,
  lambda = NULL,
  cooling = 0.999,
```

```
  temp = 1,
  energy = 0.01,
  upsample = FALSE,
  downsample = FALSE,
  resample.seed = NULL,
  x.name = NULL,
  y.name = NULL,
  print.plot = TRUE,
  plot.fitted = NULL,
  plot.predicted = NULL,
  plot.theme = rtTheme,
  question = NULL,
  rtclass = NULL,
  verbose = TRUE,
  trace = 0,
  outdir = NULL,
  save.mod = ifelse(!is.null(outdir), TRUE, FALSE),
  ...
)
```

Arguments

x	Numeric vector or matrix / data frame of features i.e. independent variables
y	Numeric vector of outcome, i.e. dependent variable
x.test	Numeric vector or matrix / data frame of testing set features Columns must correspond to columns in x
y.test	Numeric vector of testing set outcome
PPmethod	Character: "LDA": LDA index, "Lp": Lp index, "PDA": PDA index. Default = "LDA"
upsample	Logical: If TRUE, upsample cases to balance outcome classes (for Classification only) Note: upsample will randomly sample with replacement if the length of the majority class is more than double the length of the class you are upsampling, thereby introducing randomness
downsample	Logical: If TRUE, downsample majority class to match size of minority class
resample.seed	Integer: If provided, will be used to set the seed during upsampling. Default = NULL (random seed)
x.name	Character: Name for feature set
y.name	Character: Name for outcome
print.plot	Logical: if TRUE, produce plot using mplot3 Takes precedence over plot.fitted and plot.predicted. Default = TRUE
plot.fitted	Logical: if TRUE, plot True (y) vs Fitted
plot.predicted	Logical: if TRUE, plot True (y.test) vs Predicted. Requires x.test and y.test
plot.theme	Character: "zero", "dark", "box", "darkbox"
question	Character: the question you are attempting to answer with this model, in plain language.
rtclass	Character: Class type to use. "S3", "S4", "RC", "R6"
verbose	Logical: If TRUE, print summary to screen.
trace	Integer: If higher than 0, will print more information to the console. Default = 0

outdir	Path to output directory. If defined, will save Predicted vs. True plot, if available, as well as full model output, if save.mod is TRUE
save.mod	Logical: If TRUE, save all output to an RDS file in outdir save.mod is TRUE by default if an outdir is defined. If set to TRUE, and no outdir is defined, outdir defaults to paste0("./s.", mod.name)
...	Additional arguments

Details

Note: PP.Tree does not support case weights

Value

rtMod object

Author(s)

E.D. Gennatas

See Also

elevate for external cross-validation

Other Supervised Learning: s_ADABOOST(), s_ADDTREE(), s_BART(), s_BAYESGLM(), s_BRUTO(), s_C50(), s_CART(), s_CTREE(), s_ET(), s_EVTREE(), s_GAM.default(), s_GAM.formula(), s_GAMSELX2(), s_GAMSELX(), s_GAMSEL(), s_GAM(), s_GBM0(), s_GBM3.R(), s_GBM(), s_GLMNET(), s_GLMTREE(), s_GLM(), s_GLS(), s_H2ODL(), s_H2OGBM(), s_H2ORF(), s_IRF(), s_KNN(), s_LDA(), s_LMTREE(), s_LM(), s_MARS(), s_MLRF(), s_NBAYES(), s_NLA(), s_NLS(), s_NW(), s_POLYMARS(), s_PPR(), s_QDA(), s_QRNN(), s_RANGER(), s_RFSRC(), s_RF(), s_SDA(), s_SGD(), s_SPLS(), s_SVM(), s_TFN(), s_XGBLIN(), s_XGBOOST(), s_XGB()

Other Tree-based methods: s_ADABOOST(), s_ADDTREE(), s_BART(), s_C50(), s_CART(), s_CTREE(), s_ET(), s_EVTREE(), s_GBM0(), s_GBM3.R(), s_GBM(), s_GLMTREE(), s_H2OGBM(), s_H2ORF(), s_IRF(), s_LMTREE(), s_MLRF(), s_RANGER(), s_RFSRC(), s_RF(), s_XGBOOST(), s_XGB()

---

s_PSURV *Parametric Survival Regression [S]*

Description

Fit a parametric survival regression model using survival::survreg

Usage

```
s_PSURV(
  x,
  y,
  x.test = NULL,
  y.test = NULL,
  x.name = NULL,
  y.name = NULL,
  weights = NULL,
  dist = "weibull",
```

```
  control = survival::survreg.control(),
  print.plot = TRUE,
  plot.fitted = NULL,
  plot.predicted = NULL,
  plot.theme = rtTheme,
  question = NULL,
  verbose = TRUE,
  trace = 0,
  save.mod = FALSE,
  outdir = NULL,
  ...
)
```

Arguments

x	Numeric vector or matrix of features, i.e. independent variables
y	Object of class "Surv" created using survival::Surv
x.test	(Optional) Numeric vector or matrix of testing set features must have set of columns as x
y.test	(Optional) Object of class "Surv" created using survival::Surv
weights	Float: Vector of case weights
...	Additional parameters to pass to survival::survreg

Value

Object of class rtMod

Author(s)

E.D. Gennatas

See Also

elevate for external cross-validation

---

s_QDA *Quadratic Discriminant Analysis [C]*

Description

Train a QDA Classifier using MASS::qda

Usage

```
s_QDA(
  x,
  y = NULL,
  x.test = NULL,
  y.test = NULL,
  prior = NULL,
```

```
  method = "moment",
  nu = NULL,
  x.name = NULL,
  y.name = NULL,
  upsample = FALSE,
  downsample = FALSE,
  resample.seed = NULL,
  print.plot = TRUE,
  plot.fitted = NULL,
  plot.predicted = NULL,
  plot.theme = rtTheme,
  question = NULL,
  rtclass = NULL,
  verbose = TRUE,
  outdir = NULL,
  save.mod = ifelse(!is.null(outdir), TRUE, FALSE),
  ...
)
```

Arguments

x	Numeric vector or matrix / data frame of features i.e. independent variables
y	Numeric vector of outcome, i.e. dependent variable
x.test	Numeric vector or matrix / data frame of testing set features Columns must correspond to columns in x
y.test	Numeric vector of testing set outcome
x.name	Character: Name for feature set
y.name	Character: Name for outcome
upsample	Logical: If TRUE, upsample cases to balance outcome classes (for Classification only) Note: upsample will randomly sample with replacement if the length of the majority class is more than double the length of the class you are upsampling, thereby introducing randomness
downsample	Logical: If TRUE, downsample majority class to match size of minority class
resample.seed	Integer: If provided, will be used to set the seed during upsampling. Default = NULL (random seed)
print.plot	Logical: if TRUE, produce plot using mplot3 Takes precedence over plot.fitted and plot.predicted. Default = TRUE
plot.fitted	Logical: if TRUE, plot True (y) vs Fitted
plot.predicted	Logical: if TRUE, plot True (y.test) vs Predicted. Requires x.test and y.test
plot.theme	Character: "zero", "dark", "box", "darkbox"
question	Character: the question you are attempting to answer with this model, in plain language.
rtclass	Character: Class type to use. "S3", "S4", "RC", "R6"
verbose	Logical: If TRUE, print summary to screen.
outdir	Path to output directory. If defined, will save Predicted vs. True plot, if available, as well as full model output, if save.mod is TRUE
save.mod	Logical: If TRUE, save all output to an RDS file in outdir save.mod is TRUE by default if an outdir is defined. If set to TRUE, and no outdir is defined, outdir defaults to paste0("./s.", mod.name)
...	Additional arguments

Value

rtMod object

Author(s)

E.D. Gennatas

See Also

elevate for external cross-validation

Other Supervised Learning: s_ADABOOST(), s_ADDTREE(), s_BART(), s_BAYESGLM(), s_BRUTO(), s_C50(), s_CART(), s_CTREE(), s_ET(), s_EVTREE(), s_GAM.default(), s_GAM.formula(), s_GAMSELX2(), s_GAMSELX(), s_GAMSEL(), s_GAM(), s_GBM0(), s_GBM3.R(), s_GBM(), s_GLMNET(), s_GLMTREE(), s_GLM(), s_GLS(), s_H2ODL(), s_H2OGBM(), s_H2ORF(), s_IRF(), s_KNN(), s_LDA(), s_LMTREE(), s_LM(), s_MARS(), s_MLRF(), s_NBAYES(), s_NLA(), s_NLS(), s_NW(), s_POLYMARS(), s_PPR(), s_PPTREE(), s_QRNN(), s_RANGER(), s_RFSRC(), s_RF(), s_SDA(), s_SGD(), s_SPLS(), s_SVM(), s_TFN(), s_XGBLIN(), s_XGBOOST(), s_XGB()

---

s_QRNN *Quantile Regression Neural Network [R]*

Description

Train an ensemble of Neural Networks to perform Quantile Regression using qrnn

Usage

```
s_QRNN(
  x,
  y = NULL,
  x.test = NULL,
  y.test = NULL,
  x.name = NULL,
  y.name = NULL,
  n.hidden = 1,
  tau = 0.5,
  n.ensemble = 5,
  iter.max = 5000,
  n.trials = 5,
  bag = TRUE,
  lower = -Inf,
  eps.seq = 2^(-8:-32),
  Th = qrnn::sigmoid,
  Th.prime = qrnn::sigmoid.prime,
  penalty = 0,
  trace = T,
  print.plot = TRUE,
  plot.fitted = NULL,
  plot.predicted = NULL,
  plot.theme = rtTheme,
```

```
  question = NULL,
  rtclass = NULL,
  verbose = TRUE,
  outdir = NULL,
  save.mod = ifelse(!is.null(outdir), TRUE, FALSE),
  ...
)
```

Arguments

x	Numeric vector or matrix / data frame of features i.e. independent variables
y	Numeric vector of outcome, i.e. dependent variable
x.test	Numeric vector or matrix / data frame of testing set features Columns must correspond to columns in x
y.test	Numeric vector of testing set outcome
x.name	Character: Name for feature set
y.name	Character: Name for outcome
n.hidden	Integer. Number of hidden nodes
tau	Float. tau-quantile. Defaults to .5
n.ensemble	Integer. Number of NNs to train
iter.max	Integer. Max N of iteration of the optimization algorithm
n.trials	Integer. N of trials. Used to avoid local minima
bag	Logical. Should bagging be used?
trace	Integer: If higher than 0, will print more information to the console. Default = 0
print.plot	Logical: if TRUE, produce plot using mplot3 Takes precedence over plot.fitted and plot.predicted. Default = TRUE
plot.fitted	Logical: if TRUE, plot True (y) vs Fitted
plot.predicted	Logical: if TRUE, plot True (y.test) vs Predicted. Requires x.test and y.test
plot.theme	Character: "zero", "dark", "box", "darkbox"
question	Character: the question you are attempting to answer with this model, in plain language.
rtclass	Character: Class type to use. "S3", "S4", "RC", "R6"
verbose	Logical: If TRUE, print summary to screen.
outdir	Path to output directory. If defined, will save Predicted vs. True plot, if available, as well as full model output, if save.mod is TRUE
save.mod	Logical: If TRUE, save all output to an RDS file in outdir save.mod is TRUE by default if an outdir is defined. If set to TRUE, and no outdir is defined, outdir defaults to paste0("./s.", mod.name)
...	Additional arguments to be passed to qrnn::qrnn.fit

Author(s)

E.D. Gennatas

See Also

elevate for external cross-validation

Other Supervised Learning: s_ADABOOST(), s_ADDTREE(), s_BART(), s_BAYESGLM(), s_BRUTO(), s_C50(), s_CART(), s_CTREE(), s_ET(), s_EVTREE(), s_GAM.default(), s_GAM.formula(), s_GAMSELX2(), s_GAMSELX(), s_GAMSEL(), s_GAM(), s_GBM0(), s_GBM3.R(), s_GBM(), s_GLMNET(), s_GLMTREE(), s_GLM(), s_GLS(), s_H2ODL(), s_H2OGBM(), s_H2ORF(), s_IRF(), s_KNN(), s_LDA(), s_LMTREE(), s_LM(), s_MARS(), s_MLRF(), s_NBAYES(), s_NLA(), s_NLS(), s_NW(), s_POLYMARS(), s_PPR(), s_PPTREE(), s_QDA(), s_RANGER(), s_RFSRC(), s_RF(), s_SDA(), s_SGD(), s_SPLS(), s_SVM(), s_TFN(), s_XGBLIN(), s_XGBOOST(), s_XGB()

---

s_RANGER *Random Forest Classification and Regression [C, R]*

Description

Train a Random Forest for regression or classification using ranger

Usage

```
s_RANGER(
  x,
  y = NULL,
  x.test = NULL,
  y.test = NULL,
  x.name = NULL,
  y.name = NULL,
  n.trees = 1000,
  weights = NULL,
  ipw = TRUE,
  ipw.type = 2,
  ipw.case.weights = TRUE,
  ipw.class.weights = FALSE,
  upsample = FALSE,
  downsample = FALSE,
  resample.seed = NULL,
  autotune = FALSE,
  classwt = NULL,
  n.trees.try = 500,
  stepFactor = 2,
  mtry = NULL,
  mtryStart = NULL,
  inbag.resample = NULL,
  stratify.on.y = FALSE,
  grid.resample.rtset = rtset.resample("kfold", 5),
  grid.search.type = c("exhaustive", "randomized"),
  grid.randomized.p = 0.1,
  metric = NULL,
  maximize = NULL,
  probability = FALSE,
  importance = "impurity",
```

```
  local.importance = FALSE,
  replace = TRUE,
  min.node.size = NULL,
  splitrule = NULL,
  strata = NULL,
  sampsize = if (replace) nrow(x) else ceiling(0.632 * nrow(x)),
  tune.do.trace = FALSE,
  imetrics = FALSE,
  n.cores = rtCores,
  print.tune.plot = FALSE,
  print.plot = TRUE,
  plot.fitted = NULL,
  plot.predicted = NULL,
  plot.theme = rtTheme,
  question = NULL,
  grid.verbose = verbose,
  verbose = TRUE,
  outdir = NULL,
  save.mod = ifelse(!is.null(outdir), TRUE, FALSE),
  ...
)
```

Arguments

x	Numeric vector or matrix / data frame of features i.e. independent variables
y	Numeric vector of outcome, i.e. dependent variable
x.test	Numeric vector or matrix / data frame of testing set features Columns must correspond to columns in x
y.test	Numeric vector of testing set outcome
x.name	Character: Name for feature set
y.name	Character: Name for outcome
n.trees	Integer: Number of trees to grow. Default = 1000
weights	Numeric vector: Weights for cases. For classification, weights takes precedence over ipw, therefore set weights = NULL if using ipw. Note: If weight are provided, ipw is not used. Leave NULL if setting ipw = TRUE. Default = NULL
ipw	Logical: If TRUE, apply inverse probability weighting (for Classification only). Note: If weights are provided, ipw is not used. Default = TRUE
ipw.type	Integer 0, 1, 2 1: class.weights as in 0, divided by max(class.weights) 2: class.weights as in 0, divided by min(class.weights) Default = 2
ipw.case.weights	Logical: If TRUE, define ranger's case.weights using IPW. Default = TRUE Note: Cannot use case.weights together with stratify.on.y or inbag.resample
ipw.class.weights	Logical: If TRUE, define ranger's class.weights using IPW. Default = FALSE
upsample	Logical: If TRUE, upsample training set cases not belonging in majority outcome group
downsample	Logical: If TRUE, downsample majority class to match size of minority class
resample.seed	Integer: If provided, will be used to set the seed during upsampling. Default = NULL (random seed)

autotune — Logical: If TRUE, use randomForest::tuneRF to determine mtry

classwt — Vector, Float: Priors of the classes for randomForest::tuneRF if autotune = TRUE. For classification only; need not add up to 1

n.trees.try — Integer: Number of trees to train for tuning, if autotune = TRUE

stepFactor — Float: If autotune = TRUE, at each tuning iteration, mtry is multiplied or divided by this value. Default = 1.5

mtry — [gS] Integer: Number of features sampled randomly at each split. Defaults to square root of n of features for classification, and a third of n of features for regression.

mtryStart — Integer: If autotune = TRUE, start at this value for mtry

inbag.resample — List, length n.tree: Output of rtset.resample to define resamples used for each tree. Default = NULL

stratify.on.y — Logical: If TRUE, overrides inbag.resample to use stratified bootstraps for each tree. This can help improve test set performance in imbalanced datasets. Default = FALSE. Note: Cannot be used with ipw.case.weights

grid.resample.rtset — List: Output of rtset.resample defining gridSearchLearn parameters.

grid.search.type — Character: Type of grid search to perform: "exhaustive" or "randomized".

grid.randomized.p — Float (0, 1): If grid.search.type = "randomized", randomly test this proportion of combinations.

metric — Character: Metric to minimize, or maximize if maximize = TRUE during grid search. Default = NULL, which results in "Balanced Accuracy" for Classification, "MSE" for Regression, and "Coherence" for Survival Analysis.

maximize — Logical: If TRUE, metric will be maximized if grid search is run.

probability — Logical: If TRUE, grow a probability forest. See ranger::ranger. Default = FALSE

importance — Character: "none", "impurity", "impurity_corrected", or "permutation" Default = "impurity"

local.importance — Logical: If TRUE, return local importance values. Only applicable if importance is set to "permutation".

replace — Logical: If TRUE, sample cases with replacement during training.

min.node.size — [gS] Integer: Minimum node size

splitrule — Character: For classification: "gini" (Default) or "extratrees"; For regression: "variance" (Default), "extratrees" or "maxstat". For survival "logrank" (Default), "extratrees", "C" or "maxstat".

strata — Vector, Factor: Will be used for stratified sampling

sampsize — Integer: Size of sample to draw. In Classification, if strata is defined, this can be a vector of the same length, in which case, corresponding values determine how many cases are drawn from the strata.

tune.do.trace — Same as do.trace but for tuning, when autotune = TRUE

imetrics — Logical: If TRUE, calculate interpretability metrics (N of trees and N of nodes) and save under the extra field of rtMod

n.cores — Integer: Number of cores to use.

print.tune.plot	Logical: passed to randomForest::tuneRF.
print.plot	Logical: if TRUE, produce plot using mplot3 Takes precedence over plot.fitted and plot.predicted. Default = TRUE
plot.fitted	Logical: if TRUE, plot True (y) vs Fitted
plot.predicted	Logical: if TRUE, plot True (y.test) vs Predicted. Requires x.test and y.test
plot.theme	Character: "zero", "dark", "box", "darkbox"
question	Character: the question you are attempting to answer with this model, in plain language.
grid.verbose	Logical: Passed to gridSearchLearn
verbose	Logical: If TRUE, print summary to screen.
outdir	String, Optional: Path to directory to save output
save.mod	Logical: If TRUE, save all output to an RDS file in outdir save.mod is TRUE by default if an outdir is defined. If set to TRUE, and no outdir is defined, outdir defaults to paste0("./s.", mod.name)
...	Additional arguments to be passed to ranger::ranger

Details

You should cconsider, or try, setting mtry to NCOL(x), especially for small number of features. By default mtry is set to NCOL(x) for NCOL(x) <= 20. For imbalanced datasets, setting stratify.on.y = TRUE should improve performance. If autotune = TRUE, randomForest::tuneRF will be run to determine best mtry value. [gS]: indicated parameter will be tuned by grid search if more than one value is passed

See Tech Report comparing balanced (ipw.case.weights = TRUE) and weighted (ipw.class.weights = TRUE) Random Forests.

Value

rtMod object

Author(s)

E.D. Gennatas

See Also

elevate for external cross-validation

Other Supervised Learning: s_ADABOOST(), s_ADDTREE(), s_BART(), s_BAYESGLM(), s_BRUTO(), s_C50(), s_CART(), s_CTREE(), s_ET(), s_EVTREE(), s_GAM.default(), s_GAM.formula(), s_GAMSELX2(), s_GAMSELX(), s_GAMSEL(), s_GAM(), s_GBM0(), s_GBM3.R(), s_GBM(), s_GLMNET(), s_GLMTREE(), s_GLM(), s_GLS(), s_H2ODL(), s_H2OGBM(), s_H2ORF(), s_IRF(), s_KNN(), s_LDA(), s_LMTREE(), s_LM(), s_MARS(), s_MLRF(), s_NBAYES(), s_NLA(), s_NLS(), s_NW(), s_POLYMARS(), s_PPR(), s_PPTREE(), s_QDA(), s_QRNN(), s_RFSRC(), s_RF(), s_SDA(), s_SGD(), s_SPLS(), s_SVM(), s_TFN(), s_XGBLIN(), s_XGBOOST(), s_XGB()

Other Tree-based methods: s_ADABOOST(), s_ADDTREE(), s_BART(), s_C50(), s_CART(), s_CTREE(), s_ET(), s_EVTREE(), s_GBM0(), s_GBM3.R(), s_GBM(), s_GLMTREE(), s_H2OGBM(), s_H2ORF(), s_IRF(), s_LMTREE(), s_MLRF(), s_PPTREE(), s_RFSRC(), s_RF(), s_XGBOOST(), s_XGB()

Other Ensembles: s_ADABOOST(), s_GBM0(), s_GBM3.R(), s_GBM(), s_RF()

s_RF *Random Forest Classification and Regression [C, R]*

Description

Train a Random Forest for regression or classification using `randomForest`

Usage

```
s_RF(
  x,
  y = NULL,
  x.test = NULL,
  y.test = NULL,
  x.name = NULL,
  y.name = NULL,
  n.trees = 1000,
  autotune = FALSE,
  n.trees.try = 1000,
  stepFactor = 1.5,
  mtry = NULL,
  nodesize = NULL,
  maxnodes = NULL,
  mtryStart = mtry,
  grid.resample.rtset = rtset.resample("kfold", 5),
  metric = NULL,
  maximize = NULL,
  classwt = NULL,
  ipw = TRUE,
  ipw.type = 2,
  upsample = FALSE,
  downsample = FALSE,
  resample.seed = NULL,
  importance = TRUE,
  proximity = FALSE,
  replace = TRUE,
  strata = NULL,
  sampsize = if (replace) nrow(x) else ceiling(0.632 * nrow(x)),
  sampsize.ratio = NULL,
  do.trace = NULL,
  tune.do.trace = FALSE,
  imetrics = FALSE,
  n.cores = rtCores,
  print.tune.plot = FALSE,
  print.plot = TRUE,
  plot.fitted = NULL,
  plot.predicted = NULL,
  plot.theme = rtTheme,
  proximity.tsne = FALSE,
  discard.forest = FALSE,
  tsne.perplexity = 5,
```

```
  plot.tsne.train = FALSE,
  plot.tsne.test = FALSE,
  question = NULL,
  rtclass = NULL,
  verbose = TRUE,
  grid.verbose = verbose,
  outdir = NULL,
  save.mod = ifelse(!is.null(outdir), TRUE, FALSE),
  ...
)
```

Arguments

x	Numeric vector or matrix / data frame of features i.e. independent variables
y	Numeric vector of outcome, i.e. dependent variable
x.test	Numeric vector or matrix / data frame of testing set features Columns must correspond to columns in x
y.test	Numeric vector of testing set outcome
x.name	Character: Name for feature set
y.name	Character: Name for outcome
n.trees	Integer: Number of trees to grow. Default = 1000
autotune	Logical: If TRUE, use randomForest::tuneRF to determine mtry
n.trees.try	Integer: Number of trees to train for tuning, if autotune = TRUE
stepFactor	Float: If autotune = TRUE, at each tuning iteration, mtry is multiplied or divided by this value. Default = 1.5
mtry	[gS] Integer: Number of features sampled randomly at each split
nodesize	[gS]: Integer: Minimum size of terminal nodes. Default = 5 (Regression); 1 (Classification)
maxnodes	[gS]: Integer: Maximum number of terminal nodes in a tree. Default = NULL; trees grown to maximum possible
mtryStart	Integer: If autotune = TRUE, start at this value for mtry
grid.resample.rtset	List: Output of rtset.resample defining gridSearchLearn parameters.
metric	Character: Metric to minimize, or maximize if maximize = TRUE during grid search. Default = NULL, which results in "Balanced Accuracy" for Classification, "MSE" for Regression, and "Coherence" for Survival Analysis.
maximize	Logical: If TRUE, metric will be maximized if grid search is run.
classwt	Vector, Float: Priors of the classes for classification only. Need not add up to 1
ipw	Logical: If TRUE, apply inverse probability weighting (for Classification only). Note: If weights are provided, ipw is not used. Default = TRUE
ipw.type	Integer 0, 1, 2 1: class.weights as in 0, divided by max(class.weights) 2: class.weights as in 0, divided by min(class.weights) Default = 2
upsample	Logical: If TRUE, upsample training set cases not belonging in majority outcome group
downsample	Logical: If TRUE, downsample majority class to match size of minority class

resample.seed	Integer: If provided, will be used to set the seed during upsampling. Default = NULL (random seed)
importance	Logical: If TRUE, estimate variable relative importance.
proximity	Logical: If TRUE, calculate proximity measure among cases.
replace	Logical: If TRUE, sample cases with replacement during training.
strata	Vector, Factor: Will be used for stratified sampling
sampsize	Integer: Size of sample to draw. In Classification, if strata is defined, this can be a vector of the same length, in which case, corresponding values determine how many cases are drawn from the strata.
sampsize.ratio	Float (0, 1): Heuristic of sorts to increase sensitivity in unbalanced cases. Sample with replacement from minority case to create bootstraps of length N cases. Select (sampsize.ratio * N minority cases) cases from majority class.
do.trace	Logical or integer: If TRUE, randomForest will outpout information while it is running. If an integer, randomForest will report progress every this many trees. Default = n.trees/10 if verbose = TRUE
tune.do.trace	Same as do.trace but for tuning, when autotune = TRUE
imetrics	Logical: If TRUE, calculate interpretability metrics (N of trees and N of nodes) and save under the extra field of rtMod
n.cores	Integer: Number of cores to use.
print.tune.plot	Logical: passed to randomForest::tuneRF.
print.plot	Logical: if TRUE, produce plot using mplot3 Takes precedence over plot.fitted and plot.predicted. Default = TRUE
plot.fitted	Logical: if TRUE, plot True (y) vs Fitted
plot.predicted	Logical: if TRUE, plot True (y.test) vs Predicted. Requires x.test and y.test
plot.theme	Character: "zero", "dark", "box", "darkbox"
proximity.tsne	Logical: If TRUE, perform t-SNE on proximity matrix. Will be saved under 'extra' field of rtMod. Default = FALSE
discard.forest	Logical: If TRUE, remove forest from rtMod object to save space. Default = FALSE
tsne.perplexity	Numeric: Perplexity parameter for Rtsne::Rtsne
plot.tsne.train	Logical: If TRUE, plot training set tSNE projections
plot.tsne.test	Logical: If TRUE, plot testing set tSNE projections
question	Character: the question you are attempting to answer with this model, in plain language.
rtclass	Character: Class type to use. "S3", "S4", "RC", "R6"
verbose	Logical: If TRUE, print summary to screen.
grid.verbose	Logical: Passed to gridSearchLearn
outdir	String, Optional: Path to directory to save output
save.mod	Logical: If TRUE, save all output to an RDS file in outdir save.mod is TRUE by default if an outdir is defined. If set to TRUE, and no outdir is defined, outdir defaults to paste0("./s.", mod.name)
...	Additional arguments to be passed to randomForest::randomForest

Details

If `autotue = TRUE`, `randomForest::tuneRF` will be run to determine best `mtry` value.

Value

rtMod object

Author(s)

E.D. Gennatas

See Also

elevate for external cross-validation

Other Supervised Learning: `s_ADABOOST()`, `s_ADDTREE()`, `s_BART()`, `s_BAYESGLM()`, `s_BRUTO()`, `s_C50()`, `s_CART()`, `s_CTREE()`, `s_ET()`, `s_EVTREE()`, `s_GAM.default()`, `s_GAM.formula()`, `s_GAMSELX2()`, `s_GAMSELX()`, `s_GAMSEL()`, `s_GAM()`, `s_GBM0()`, `s_GBM3.R()`, `s_GBM()`, `s_GLMNET()`, `s_GLMTREE()`, `s_GLM()`, `s_GLS()`, `s_H2ODL()`, `s_H2OGBM()`, `s_H2ORF()`, `s_IRF()`, `s_KNN()`, `s_LDA()`, `s_LMTREE()`, `s_LM()`, `s_MARS()`, `s_MLRF()`, `s_NBAYES()`, `s_NLA()`, `s_NLS()`, `s_NW()`, `s_POLYMARS()`, `s_PPR()`, `s_PPTREE()`, `s_QDA()`, `s_QRNN()`, `s_RANGER()`, `s_RFSRC()`, `s_SDA()`, `s_SGD()`, `s_SPLS()`, `s_SVM()`, `s_TFN()`, `s_XGBLIN()`, `s_XGBOOST()`, `s_XGB()`

Other Tree-based methods: `s_ADABOOST()`, `s_ADDTREE()`, `s_BART()`, `s_C50()`, `s_CART()`, `s_CTREE()`, `s_ET()`, `s_EVTREE()`, `s_GBM0()`, `s_GBM3.R()`, `s_GBM()`, `s_GLMTREE()`, `s_H2OGBM()`, `s_H2ORF()`, `s_IRF()`, `s_LMTREE()`, `s_MLRF()`, `s_PPTREE()`, `s_RANGER()`, `s_RFSRC()`, `s_XGBOOST()`, `s_XGB()`

Other Ensembles: `s_ADABOOST()`, `s_GBM0()`, `s_GBM3.R()`, `s_GBM()`, `s_RANGER()`

---

s_RFSRC *Random Forest for Classification, Regression, and Survival [C, R, S]*

Description

Train a Random Forest for Regression, Classification, or Survival Regression using `randomForestSRC`

Usage

```
s_RFSRC(
  x,
  y = NULL,
  x.test = NULL,
  y.test = NULL,
  x.name = NULL,
  y.name = NULL,
  n.trees = 1000,
  weights = NULL,
  ipw = TRUE,
  ipw.type = 2,
  upsample = FALSE,
  downsample = FALSE,
  resample.seed = NULL,
  bootstrap = "by.root",
```

```
  mtry = NULL,
  importance = TRUE,
  proximity = TRUE,
  nodesize = if (!is.null(y) && !is.factor(y)) 5 else 1,
  nodedepth = NULL,
  na.action = "na.impute",
  trace = FALSE,
  print.plot = TRUE,
  plot.fitted = NULL,
  plot.predicted = NULL,
  plot.theme = rtTheme,
  question = NULL,
  rtclass = NULL,
  verbose = TRUE,
  outdir = NULL,
  save.mod = ifelse(!is.null(outdir), TRUE, FALSE),
  ...
)
```

Arguments

x	Numeric vector or matrix of features, i.e. independent variables
y	Numeric vector of outcome, i.e. dependent variable
x.test	(Optional) Numeric vector or matrix of validation set features must have set of columns as x
y.test	(Optional) Numeric vector of validation set outcomes
n.trees	Integer: Number of trees to grow. The more the merrier.
bootstrap	Character:
mtry	Integer: Number of features sampled randomly at each split
outdir	Optional. Path to directory to save output
...	Additional arguments to be passed to randomForestSRC::rfsrc

Details

For Survival Regression, y must be an object of type Surv, created using survival::Surv(time, status) mtry is the only tunable parameter, but it usually only makes a small difference and is often not tuned.

Value

Object of class rtMod

Author(s)

E.D. Gennatas

See Also

elevate for external cross-validation

Other Supervised Learning: s_ADABOOST(), s_ADDTREE(), s_BART(), s_BAYESGLM(), s_BRUTO(), s_C50(), s_CART(), s_CTREE(), s_ET(), s_EVTREE(), s_GAM.default(), s_GAM.formula(),

s_GAMSELX2(), s_GAMSELX(), s_GAMSEL(), s_GAM(), s_GBM0(), s_GBM3.R(), s_GBM(), s_GLMNET(), s_GLMTREE(), s_GLM(), s_GLS(), s_H2ODL(), s_H2OGBM(), s_H2ORF(), s_IRF(), s_KNN(), s_LDA(), s_LMTREE(), s_LM(), s_MARS(), s_MLRF(), s_NBAYES(), s_NLA(), s_NLS(), s_NW(), s_POLYMARS(), s_PPR(), s_PPTREE(), s_QDA(), s_QRNN(), s_RANGER(), s_RF(), s_SDA(), s_SGD(), s_SPLS(), s_SVM(), s_TFN(), s_XGBLIN(), s_XGBOOST(), s_XGB()

Other Tree-based methods: s_ADABOOST(), s_ADDTREE(), s_BART(), s_C50(), s_CART(), s_CTREE(), s_ET(), s_EVTREE(), s_GBM0(), s_GBM3.R(), s_GBM(), s_GLMTREE(), s_H2OGBM(), s_H2ORF(), s_IRF(), s_LMTREE(), s_MLRF(), s_PPTREE(), s_RANGER(), s_RF(), s_XGBOOST(), s_XGB()

---

s_RLM *Robust linear model*

Description

Convenience alias for s_LM(robust = T). Uses MASS::rlm

Usage

```
s_RLM(x, y, x.test = NULL, y.test = NULL, ...)
```

Arguments

x	Numeric vector or matrix / data frame of features i.e. independent variables
y	Numeric vector of outcome, i.e. dependent variable
x.test	Numeric vector or matrix / data frame of testing set features Columns must correspond to columns in x
y.test	Numeric vector of testing set outcome
...	Additional parameters to be passed to MASS::rlm

---

s_RULEFIT *ruleFit [C, R]*

Description

Train a gradient boosting model, extract rules, and fit using LASSO

Usage

```
s_RULEFIT(
  x,
  y = NULL,
  x.test = NULL,
  y.test = NULL,
  n.trees = 100,
 gbm.params = list(bag.fraction = 0.5, shrinkage = 0.001, interaction.depth = 5, ipw =
    TRUE),
  meta.alpha = 1,
  meta.lambda = NULL,
```

```
  meta.extra.params = list(ipw = TRUE),
  cases.by.rules = NULL,
  x.name = NULL,
  y.name = NULL,
  n.cores = rtCores,
  which.gbm = c("gbm", "gbm3"),
  question = NULL,
  print.plot = TRUE,
  plot.fitted = NULL,
  plot.predicted = NULL,
  plot.theme = rtTheme,
  outdir = NULL,
  save.mod = if (!is.null(outdir)) TRUE else FALSE,
  verbose = TRUE
)
```

Arguments

x	Numeric vector or matrix / data frame of features i.e. independent variables
y	Numeric vector of outcome, i.e. dependent variable
x.test	Numeric vector or matrix / data frame of testing set features Columns must correspond to columns in x
y.test	Numeric vector of testing set outcome
n.trees	Integer: Initial number of trees to fit
gbm.params	Named list: Parameters for s_GBM
meta.alpha	Float [0, 1]: alpha for s_GLMNET, Default = 1
meta.lambda	Float: lambda for s_GLMNET. Default = NULL (will be determined automatically by crossvalidation)
meta.extra.params	Named list: Parameters for s_GLMNET for the feature selection step
cases.by.rules	Matrix of cases by rules from a previoue rulefit run. If provided, the GBM step is skipped. Default = NULL
x.name	Character: Name for feature set
y.name	Character: Name for outcome
n.cores	Integer: Number of cores to use
which.gbm	Character: "gbm" or "gbm3"
question	Character: the question you are attempting to answer with this model, in plain language.
print.plot	Logical: if TRUE, produce plot using mplot3 Takes precedence over plot.fitted and plot.predicted. Default = TRUE
plot.fitted	Logical: if TRUE, plot True (y) vs Fitted
plot.predicted	Logical: if TRUE, plot True (y.test) vs Predicted. Requires x.test and y.test
plot.theme	Character: "zero", "dark", "box", "darkbox"
outdir	Character: If defined, save log, 'plot.all' plots (see above) and RDS file of complete output
save.mod	Logical: If TRUE, save all output to an RDS file in outdir save.mod is TRUE by default if an outdir is defined. If set to TRUE, and no outdir is defined, outdir defaults to paste0("./s.", mod.name)
verbose	Logical: If TRUE, print summary to screen.

Details

Based on "Predictive Learning via Rule Ensembles" by Friedman and Popescu http://statweb.stanford.edu/~jhf/ftp/RuleFi

Value

rtMod object

Author(s)

E.D. Gennatas

References

Friedman JH, Popescu BE, "Predictive Learning via Rule Ensembles", http://statweb.stanford.edu/~jhf/ftp/RuleFit.pdf

---

s_SDA *Sparse Linear Discriminant Analysis*

Description

Train an SDA Classifier using `sparseLDA::sda`

Usage

```
s_SDA(
  x,
  y = NULL,
  x.test = NULL,
  y.test = NULL,
  lambda = 1e-06,
  stop = NULL,
  maxIte = 100,
  Q = NULL,
  tol = 1e-06,
  .preprocess = rtset.preprocess(scale = TRUE, center = TRUE),
  upsample = TRUE,
  downsample = FALSE,
  resample.seed = NULL,
  x.name = NULL,
  y.name = NULL,
  grid.resample.rtset = rtset.resample("kfold", 5),
  grid.search.type = c("exhaustive", "randomized"),
  grid.randomized.p = 0.1,
  metric = NULL,
  maximize = NULL,
  print.plot = TRUE,
  plot.fitted = NULL,
  plot.predicted = NULL,
  plot.theme = rtTheme,
  question = NULL,
  verbose = TRUE,
```

```
  grid.verbose = verbose,
  trace = 0,
  outdir = NULL,
  n.cores = rtCores,
  save.mod = ifelse(!is.null(outdir), TRUE, FALSE)
)
```

Arguments

x	Numeric vector or matrix / data frame of features i.e. independent variables
y	Numeric vector of outcome, i.e. dependent variable
x.test	Numeric vector or matrix / data frame of testing set features Columns must correspond to columns in x
y.test	Numeric vector of testing set outcome
lambda	L2-norm weight for elastic net regression
stop	If STOP is negative, its absolute value corresponds to the desired number of variables. If STOP is positive, it corresponds to an upper bound on the L1-norm of the b coefficients. There is a one to one correspondence between stop and t. The default is -p (-the number of variables).
maxIte	Integer: Maximum number of iterations
Q	Integer: Number of components
tol	Numeric: Tolerance for change in RSS, which is the stopping criterion
.preprocess	List of preprocessing parameters. Scaling and centering is enabled by default, because it is crucial for algorithm to learn.
upsample	Logical: If TRUE, upsample cases to balance outcome classes (for Classification only) Note: upsample will randomly sample with replacement if the length of the majority class is more than double the length of the class you are upsampling, thereby introducing randomness
downsample	Logical: If TRUE, downsample majority class to match size of minority class
resample.seed	Integer: If provided, will be used to set the seed during upsampling. Default = NULL (random seed)
x.name	Character: Name for feature set
y.name	Character: Name for outcome
grid.resample.rtset	List: Output of rtset.resample defining gridSearchLearn parameters.
grid.search.type	Character: Type of grid search to perform: "exhaustive" or "randomized".
grid.randomized.p	Float (0, 1): If grid.search.type = "randomized", randomly test this proportion of combinations.
metric	Character: Metric to minimize, or maximize if maximize = TRUE during grid search. Default = NULL, which results in "Balanced Accuracy" for Classification, "MSE" for Regression, and "Coherence" for Survival Analysis.
maximize	Logical: If TRUE, metric will be maximized if grid search is run.
print.plot	Logical: if TRUE, produce plot using mplot3 Takes precedence over plot.fitted and plot.predicted. Default = TRUE
plot.fitted	Logical: if TRUE, plot True (y) vs Fitted

plot.predicted	Logical: if TRUE, plot True (y.test) vs Predicted. Requires x.test and y.test
plot.theme	Character: "zero", "dark", "box", "darkbox"
question	Character: the question you are attempting to answer with this model, in plain language.
verbose	Logical: If TRUE, print summary to screen.
grid.verbose	Logical: Passed to gridSearchLearn
outdir	Path to output directory. If defined, will save Predicted vs. True plot, if available, as well as full model output, if save.mod is TRUE
n.cores	Integer: Number of cores to use.
save.mod	Logical: If TRUE, save all output to an RDS file in outdir save.mod is TRUE by default if an outdir is defined. If set to TRUE, and no outdir is defined, outdir defaults to paste0("./s.", mod.name)

Value

rtMod object

Author(s)

E.D. Gennatas

See Also

elevate for external cross-validation

Other Supervised Learning: s_ADABOOST(), s_ADDTREE(), s_BART(), s_BAYESGLM(), s_BRUTO(), s_C50(), s_CART(), s_CTREE(), s_ET(), s_EVTREE(), s_GAM.default(), s_GAM.formula(), s_GAMSELX2(), s_GAMSELX(), s_GAMSEL(), s_GAM(), s_GBM0(), s_GBM3.R(), s_GBM(), s_GLMNET(), s_GLMTREE(), s_GLM(), s_GLS(), s_H2ODL(), s_H2OGBM(), s_H2ORF(), s_IRF(), s_KNN(), s_LDA(), s_LMTREE(), s_LM(), s_MARS(), s_MLRF(), s_NBAYES(), s_NLA(), s_NLS(), s_NW(), s_POLYMARS(), s_PPR(), s_PPTREE(), s_QDA(), s_QRNN(), s_RANGER(), s_RFSRC(), s_RF(), s_SGD(), s_SPLS(), s_SVM(), s_TFN(), s_XGBLIN(), s_XGBOOST(), s_XGB()

Examples

```
## Not run:
datc2 <- iris[51:150, ]
datc2$Species <- factor(datc2$Species)
resc2 <- resample(datc2)
datc2_train <- datc2[resc2$Subsample_1, ]
datc2_test <- datc2[-resc2$Subsample_1, ]
# Without scaling or centering, fails to learn
mod_c2 <- s_SDA(datc2_train, datc2_test, .preprocess = NULL)
# Learns fine with default settings (scaling & centering)
mod_c2 <- s_SDA(datc2_train, datc2_test)

## End(Not run)
```

s_SGD *Stochastic Gradient Descent (SGD) [C, R]*

Description

Train a model by Stochastic Gradient Descent using `sgd::sgd`

Usage

```
s_SGD(
  x,
  y = NULL,
  x.test = NULL,
  y.test = NULL,
  x.name = NULL,
  y.name = NULL,
  model = NULL,
  model.control = list(lambda1 = 0, lambda2 = 0),
  sgd.control = list(method = "ai-sgd"),
  upsample = FALSE,
  downsample = FALSE,
  resample.seed = NULL,
  print.plot = TRUE,
  plot.fitted = NULL,
  plot.predicted = NULL,
  plot.theme = rtTheme,
  question = NULL,
  verbose = TRUE,
  outdir = NULL,
  save.mod = ifelse(!is.null(outdir), TRUE, FALSE),
  ...
)
```

Arguments

`x`	Numeric vector or matrix / data frame of features i.e. independent variables
`y`	Numeric vector of outcome, i.e. dependent variable
`x.test`	Numeric vector or matrix / data frame of testing set features Columns must correspond to columns in `x`
`y.test`	Numeric vector of testing set outcome
`x.name`	Character: Name for feature set
`y.name`	Character: Name for outcome
`model`	character specifying the model to be used: `"lm"` (linear model), `"glm"` (generalized linear model), `"cox"` (Cox proportional hazards model), `"gmm"` (generalized method of moments), `"m"` (M-estimation). See 'Details'.
`model.control`	a list of parameters for controlling the model.
	`family` (`"glm"`) a description of the error distribution and link function to be used in the model. This can be a character string naming a family function, a family function or the result of a call to a family function. (See `family` for details of family functions.)

rank ("glm") logical. Should the rank of the design matrix be checked?

fn ("gmm") a function $g(\theta, x)$ which returns a k-vector corresponding to the k moment conditions. It is a required argument if gr not specified.

gr ("gmm") a function to return the gradient. If unspecified, a finite-difference approximation will be used.

nparams ("gmm") number of model parameters. This is automatically determined for other models.

type ("gmm") character specifying the generalized method of moments procedure: "twostep" (Hansen, 1982), "iterative" (Hansen et al., 1996). Defaults to "iterative".

wmatrix ("gmm") weighting matrix to be used in the loss function. Defaults to the identity matrix.

loss ("m") character specifying the loss function to be used in the estimating equation. Default is the Huber loss.

lambda1 L1 regularization parameter. Default is 0.

lambda2 L2 regularization parameter. Default is 0.

sgd.control an optional list of parameters for controlling the estimation.

method character specifying the method to be used: "sgd", "implicit", "asgd", "ai-sgd", "momentum", "nesterov". Default is "ai-sgd". See 'Details'.

lr character specifying the learning rate to be used: "one-dim", "one-dim-eigen", "d-dim", "adagrad", "rmsprop". Default is "one-dim". See 'Details'.

lr.control vector of scalar hyperparameters one can set dependent on the learning rate. For hyperparameters aimed to be left as default, specify NA in the corresponding entries. See 'Details'.

start starting values for the parameter estimates. Default is random initialization around zero.

size number of SGD estimates to store for diagnostic purposes (distributed log-uniformly over total number of iterations)

reltol relative convergence tolerance. The algorithm stops if it is unable to change the relative mean squared difference in the parameters by more than the amount. Default is 1e-05.

npasses the maximum number of passes over the data. Default is 3.

pass logical. Should tol be ignored and run the algorithm for all of npasses?

shuffle logical. Should the algorithm shuffle the data set including for each pass?

verbose logical. Should the algorithm print progress?

upsample Logical: If TRUE, upsample cases to balance outcome classes (for Classification only) Note: upsample will randomly sample with replacement if the length of the majority class is more than double the length of the class you are upsampling, thereby introducing randomness

downsample Logical: If TRUE, downsample majority class to match size of minority class

resample.seed Integer: If provided, will be used to set the seed during upsampling. Default = NULL (random seed)

print.plot Logical: if TRUE, produce plot using mplot3 Takes precedence over plot.fitted and plot.predicted. Default = TRUE

plot.fitted Logical: if TRUE, plot True (y) vs Fitted

plot.predicted Logical: if TRUE, plot True (y.test) vs Predicted. Requires x.test and y.test

plot.theme	Character: "zero", "dark", "box", "darkbox"
question	Character: the question you are attempting to answer with this model, in plain language.
verbose	Logical: If TRUE, print summary to screen.
outdir	Path to output directory. If defined, will save Predicted vs. True plot, if available, as well as full model output, if save.mod is TRUE
save.mod	Logical: If TRUE, save all output to an RDS file in outdir save.mod is TRUE by default if an outdir is defined. If set to TRUE, and no outdir is defined, outdir defaults to paste0("./s.", mod.name)
...	Additional arguments to be passed to sgd.control

Details

From sgd::sgd: "Models: The Cox model assumes that the survival data is ordered when passed in, i.e., such that the risk set of an observation i is all data points after it."

Value

Object of class **rtemis**

Author(s)

E.D. Gennatas

See Also

elevate for external cross-validation

Other Supervised Learning: s_ADABOOST(), s_ADDTREE(), s_BART(), s_BAYESGLM(), s_BRUTO(), s_C50(), s_CART(), s_CTREE(), s_ET(), s_EVTREE(), s_GAM.default(), s_GAM.formula(), s_GAMSELX2(), s_GAMSELX(), s_GAMSEL(), s_GAM(), s_GBM0(), s_GBM3.R(), s_GBM(), s_GLMNET(), s_GLMTREE(), s_GLM(), s_GLS(), s_H2ODL(), s_H2OGBM(), s_H2ORF(), s_IRF(), s_KNN(), s_LDA(), s_LMTREE(), s_LM(), s_MARS(), s_MLRF(), s_NBAYES(), s_NLA(), s_NLS(), s_NW(), s_POLYMARS(), s_PPR(), s_PPTREE(), s_QDA(), s_QRNN(), s_RANGER(), s_RFSRC(), s_RF(), s_SDA(), s_SPLS(), s_SVM(), s_TFN(), s_XGBLIN(), s_XGBOOST(), s_XGB()

---

s_SPLS *Sparse Partial Least Squares Regression [C, R]*

Description

Train an SPLS model using spls::spls (Regression) and spls::splsda (Classification)

Usage

```
s_SPLS(
  x,
  y = NULL,
  x.test = NULL,
  y.test = NULL,
  x.name = NULL,
```

```
  y.name = NULL,
  upsample = TRUE,
  downsample = FALSE,
  resample.seed = NULL,
  k = 2,
  eta = 0.5,
  kappa = 0.5,
  select = "pls2",
  fit = "simpls",
  scale.x = TRUE,
  scale.y = TRUE,
  maxstep = 100,
  classifier = c("lda", "logistic"),
  grid.resample.rtset = rtset.resample("kfold", 5),
  grid.search.type = c("exhaustive", "randomized"),
  grid.randomized.p = 0.1,
  metric = NULL,
  maximize = NULL,
  print.plot = TRUE,
  plot.fitted = NULL,
  plot.predicted = NULL,
  plot.theme = rtTheme,
  question = NULL,
  verbose = TRUE,
  trace = 0,
  grid.verbose = verbose,
  outdir = NULL,
  save.mod = ifelse(!is.null(outdir), TRUE, FALSE),
  n.cores = rtCores,
  ...
)
```

Arguments

x	Numeric vector or matrix / data frame of features i.e. independent variables
y	Numeric vector of outcome, i.e. dependent variable
x.test	Numeric vector or matrix / data frame of testing set features Columns must correspond to columns in x
y.test	Numeric vector of testing set outcome
x.name	Character: Name for feature set
y.name	Character: Name for outcome
upsample	Logical: If TRUE, upsample cases to balance outcome classes (for Classification only) Note: upsample will randomly sample with replacement if the length of the majority class is more than double the length of the class you are upsampling, thereby introducing randomness
downsample	Logical: If TRUE, downsample majority class to match size of minority class
resample.seed	Integer: If provided, will be used to set the seed during upsampling. Default = NULL (random seed)
k	[gS] Integer: Number of components to estimate. Default = 2
eta	[gS] Float [0, 1): Thresholding parameter. Default = .5

kappa	[gS] Float [0, .5]: Only relevant for multivariate responses: controls effect of concavity of objective function. Default = .5
select	[gS] Character: "pls2", "simpls". PLS algorithm for variable selection. Default = "pls2"
fit	[gS] Character: "kernelpls", "widekernelpls", "simpls", "oscorespls". Algorithm for model fitting.
scale.x	Logical: if TRUE, scale features by dividing each column by its sample standard deviation
scale.y	Logical: if TRUE, scale outcomes by dividing each column by its sample standard deviation
maxstep	[gS] Integer: Maximum number of iteration when fitting direction vectors. Default = 100
classifier	Character: Classifier used by spls::splsda "lda" or "logistic": Default = "lda"
grid.resample.rtset	List: Output of rtset.resample defining gridSearchLearn parameters.
grid.search.type	Character: Type of grid search to perform: "exhaustive" or "randomized".
grid.randomized.p	Float (0, 1): If grid.search.type = "randomized", randomly test this proportion of combinations.
metric	Character: Metric to minimize, or maximize if maximize = TRUE during grid search. Default = NULL, which results in "Balanced Accuracy" for Classification, "MSE" for Regression, and "Coherence" for Survival Analysis.
maximize	Logical: If TRUE, metric will be maximized if grid search is run.
print.plot	Logical: if TRUE, produce plot using mplot3 Takes precedence over plot.fitted and plot.predicted. Default = TRUE
plot.fitted	Logical: if TRUE, plot True (y) vs Fitted
plot.predicted	Logical: if TRUE, plot True (y.test) vs Predicted. Requires x.test and y.test
plot.theme	Character: "zero", "dark", "box", "darkbox"
question	Character: the question you are attempting to answer with this model, in plain language.
verbose	Logical: If TRUE, print summary to screen.
trace	If > 0 print diagnostic messages
grid.verbose	Logical: Passed to gridSearchLearn
outdir	Path to output directory. If defined, will save Predicted vs. True plot, if available, as well as full model output, if save.mod is TRUE
save.mod	Logical: If TRUE, save all output to an RDS file in outdir save.mod is TRUE by default if an outdir is defined. If set to TRUE, and no outdir is defined, outdir defaults to paste0("./s.", mod.name)
n.cores	Integer: Number of cores to be used by gridSearchLearn_future
...	Additional parameters to be passed to npreg

Details

[gS] denotes argument can be passed as a vector of values, which will trigger a grid search using gridSearchLearn np::npreg allows inputs with mixed data types.

Value

Object of class **rtemis**

Author(s)

E.D. Gennatas

See Also

elevate for external cross-validation

Other Supervised Learning: s_ADABOOST(), s_ADDTREE(), s_BART(), s_BAYESGLM(), s_BRUTO(), s_C50(), s_CART(), s_CTREE(), s_ET(), s_EVTREE(), s_GAM.default(), s_GAM.formula(), s_GAMSELX2(), s_GAMSELX(), s_GAMSEL(), s_GAM(), s_GBM0(), s_GBM3.R(), s_GBM(), s_GLMNET(), s_GLMTREE(), s_GLM(), s_GLS(), s_H2ODL(), s_H2OGBM(), s_H2ORF(), s_IRF(), s_KNN(), s_LDA(), s_LMTREE(), s_LM(), s_MARS(), s_MLRF(), s_NBAYES(), s_NLA(), s_NLS(), s_NW(), s_POLYMARS(), s_PPR(), s_PPTREE(), s_QDA(), s_QRNN(), s_RANGER(), s_RFSRC(), s_RF(), s_SDA(), s_SGD(), s_SVM(), s_TFN(), s_XGBLIN(), s_XGBOOST(), s_XGB()

Examples

```
## Not run:
x <- rnorm(100)
y <- .6 * x + 12 + rnorm(100)
mod <- s_SPLS(x, y)
## End(Not run)
```

---

s_SVM *Support Vector Machines [C, R]*

Description

Train an SVM learner using `e1071::svm`

Usage

```
s_SVM(
  x,
  y = NULL,
  x.test = NULL,
  y.test = NULL,
  x.name = NULL,
  y.name = NULL,
  grid.resample.rtset = rtset.resample("kfold", 5),
  grid.search.type = c("exhaustive", "randomized"),
  grid.randomized.p = 0.1,
  class.weights = NULL,
  ipw = TRUE,
  ipw.type = 2,
  upsample = FALSE,
  downsample = FALSE,
  resample.seed = NULL,
```

```
  kernel = "radial",
  degree = 3,
  gamma = NULL,
  coef0 = 0,
  cost = 1,
  probability = TRUE,
  metric = NULL,
  maximize = NULL,
  plot.fitted = NULL,
  plot.predicted = NULL,
  print.plot = TRUE,
  plot.theme = rtTheme,
  n.cores = rtCores,
  question = NULL,
  rtclass = NULL,
  verbose = TRUE,
  grid.verbose = verbose,
  outdir = NULL,
  save.mod = ifelse(!is.null(outdir), TRUE, FALSE),
  ...
)
```

Arguments

x	Numeric vector or matrix / data frame of features i.e. independent variables
y	Numeric vector of outcome, i.e. dependent variable
x.test	Numeric vector or matrix / data frame of testing set features Columns must correspond to columns in x
y.test	Numeric vector of testing set outcome
x.name	Character: Name for feature set
y.name	Character: Name for outcome
grid.resample.rtset	List: Output of rtset.resample defining gridSearchLearn parameters.
grid.search.type	Character: Type of grid search to perform: "exhaustive" or "randomized".
grid.randomized.p	Float (0, 1): If grid.search.type = "randomized", randomly test this proportion of combinations.
class.weights	Float, length = n levels of outcome: Weights for each outcome class.For classification, class.weights takes precedence over ipw, therefore set class.weights = NULL if using ipw.
ipw	Logical: If TRUE, apply inverse probability weighting (for Classification only). Note: If weights are provided, ipw is not used. Default = TRUE
ipw.type	Integer 0, 1, 2 1: class.weights as in 0, divided by max(class.weights) 2: class.weights as in 0, divided by min(class.weights) Default = 2
upsample	Logical: If TRUE, upsample cases to balance outcome classes (for Classification only) Note: upsample will randomly sample with replacement if the length of the majority class is more than double the length of the class you are upsampling, thereby introducing randomness

downsample	Logical: If TRUE, downsample majority class to match size of minority class
resample.seed	Integer: If provided, will be used to set the seed during upsampling. Default = NULL (random seed)
kernel	Character: "linear", "polynomial", "radial", "sigmoid"
degree	[gS] Integer: Degree for kernel = "polynomial".
gamma	[gS] Float: Parameter used in all kernels except linear
coef0	[gS] Float: Parameter used by kernels polynomial and sigmoid
cost	[gS] Float: Cost of constraints violation; the C constant of the regularization term in the Lagrange formulation.
probability	Logical: If TRUE, model allows probability estimates
metric	Character: Metric to minimize, or maximize if maximize = TRUE during grid search. Default = NULL, which results in "Balanced Accuracy" for Classification, "MSE" for Regression, and "Coherence" for Survival Analysis.
maximize	Logical: If TRUE, metric will be maximized if grid search is run.
plot.fitted	Logical: if TRUE, plot True (y) vs Fitted
plot.predicted	Logical: if TRUE, plot True (y.test) vs Predicted. Requires x.test and y.test
print.plot	Logical: if TRUE, produce plot using mplot3 Takes precedence over plot.fitted and plot.predicted. Default = TRUE
plot.theme	Character: "zero", "dark", "box", "darkbox"
n.cores	Integer: Number of cores to use.
question	Character: the question you are attempting to answer with this model, in plain language.
rtclass	Character: Class type to use. "S3", "S4", "RC", "R6"
verbose	Logical: If TRUE, print summary to screen.
grid.verbose	Logical: Passed to gridSearchLearn
outdir	Path to output directory. If defined, will save Predicted vs. True plot, if available, as well as full model output, if save.mod is TRUE
save.mod	Logical: If TRUE, save all output to an RDS file in outdir save.mod is TRUE by default if an outdir is defined. If set to TRUE, and no outdir is defined, outdir defaults to paste0("./s.", mod.name)
...	Additional arguments to be passed to e1071::svm

Details

[gS] denotes parameters that will be tuned by cross-validation if more than one value is passed. Regarding SVM tuning, the following guide from the LIBSVM authors can be useful: http://www.csie.ntu.edu.tw/~cjlin/pap
They suggest searching for cost = 2 ^ seq(-5, 15, 2) and gamma = 2 ^ seq(-15, 3, 2)

Author(s)

E.D. Gennatas

See Also

elevate for external cross-validation

Other Supervised Learning: s_ADABOOST(), s_ADDTREE(), s_BART(), s_BAYESGLM(), s_BRUTO(), s_C50(), s_CART(), s_CTREE(), s_ET(), s_EVTREE(), s_GAM.default(), s_GAM.formula(), s_GAMSELX2(), s_GAMSELX(), s_GAMSEL(), s_GAM(), s_GBM0(), s_GBM3.R(), s_GBM(), s_GLMNET(), s_GLMTREE(), s_GLM(), s_GLS(), s_H2ODL(), s_H2OGBM(), s_H2ORF(), s_IRF(), s_KNN(), s_LDA(), s_LMTREE(), s_LM(), s_MARS(), s_MLRF(), s_NBAYES(), s_NLA(), s_NLS(), s_NW(), s_POLYMARS(), s_PPR(), s_PPTREE(), s_QDA(), s_QRNN(), s_RANGER(), s_RFSRC(), s_RF(), s_SDA(), s_SGD(), s_SPLS(), s_TFN(), s_XGBLIN(), s_XGBOOST(), s_XGB()

---

s_TFN *Artificial Neural Network with* **tensorflow** *[C, R]*

Description

Train an Artificial Neural Network using **keras** and **tensorflow**

Usage

```
s_TFN(
  x,
  y = NULL,
  x.test = NULL,
  y.test = NULL,
  x.valid = NULL,
  y.valid = NULL,
  class.weights = NULL,
  ipw = TRUE,
  ipw.type = 2,
  upsample = FALSE,
  downsample = FALSE,
  resample.seed = NULL,
  net = NULL,
  n.hidden.nodes = NULL,
  initializer = c("glorot_uniform", "glorot_normal", "he_uniform", "he_normal",
    "lecun_uniform", "lecun_normal", "random_uniform", "random_normal",
   "variance_scaling", "truncated_normal", "orthogonal", "zeros", "ones", "constant"),
  initializer.seed = NULL,
  dropout = 0,
  activation = c("relu", "selu", "elu", "sigmoid", "hard_sigmoid", "tanh",
    "exponential", "linear", "softmax", "softplus", "softsign"),
  kernel_l1 = 0.1,
  kernel_l2 = 0,
  activation_l1 = 0,
  activation_l2 = 0,
  batch.normalization = TRUE,
  output = NULL,
  loss = NULL,
 optimizer = c("rmsprop", "adadelta", "adagrad", "adam", "adamax", "nadam", "sgd"),
  learning.rate = NULL,
```

```
  metric = NULL,
  epochs = 100,
  batch.size = NULL,
  validation.split = 0.2,
  callback = keras::callback_early_stopping(patience = 150),
  scale = TRUE,
  x.name = NULL,
  y.name = NULL,
  print.plot = TRUE,
  print.error.plot = NULL,
  rtlayout.mat = c(2, 1),
  plot.fitted = NULL,
  plot.predicted = NULL,
  plot.theme = rtTheme,
  question = NULL,
  verbose = TRUE,
  verbose.checkpoint = FALSE,
  outdir = NULL,
  save.mod = ifelse(!is.null(outdir), TRUE, FALSE),
  ...
)
```

Arguments

x	Numeric vector or matrix / data frame of features i.e. independent variables
y	Numeric vector of outcome, i.e. dependent variable
x.test	Numeric vector or matrix / data frame of testing set features Columns must correspond to columns in x
y.test	Numeric vector of testing set outcome
ipw	Logical: If TRUE, apply inverse probability weighting (for Classification only). Note: If weights are provided, ipw is not used. Default = TRUE
ipw.type	Integer 0, 1, 2 1: class.weights as in 0, divided by max(class.weights) 2: class.weights as in 0, divided by min(class.weights) Default = 2
upsample	Logical: If TRUE, upsample cases to balance outcome classes (for Classification only) Note: upsample will randomly sample with replacement if the length of the majority class is more than double the length of the class you are upsampling, thereby introducing randomness
downsample	Logical: If TRUE, downsample majority class to match size of minority class
resample.seed	Integer: If provided, will be used to set the seed during upsampling. Default = NULL (random seed)
n.hidden.nodes	Integer vector: Length must be equal to the number of hidden layers you wish to create. Can be zero, in which case you get a linear model. Default = N of features, i.e. NCOL(x)
initializer	Character: Initializer to use for each layer: "glorot_uniform", "glorot_normal", "he_uniform", "he_normal", "cun_uniform", "lecun_normal", "random_uniform", "random_normal", "variance_scaling", "truncated_normal", "orthogonal", "zeros", "ones", "constant". Glorot is also known as Xavier initialization. Default = "glorot_uniform"
initializer.seed	Integer: Seed to use for each initializer for reproducibility. Default = NULL

dropout Floar, vector, (0, 1): Probability of dropping nodes. Can be a vector of length equal to N of layers, otherwise will be recycled. Default = 0

activation String vector: Activation type to use: "relu", "selu", "elu", "sigmoid", "hard_sigmoid", "tanh", "exponential", "linear", "softmax", "softplus", "softsign". Defaults to "relu" for Classification and "tanh" for Regression

kernel_l1 Float: l1 penalty on weights. Default = .1

kernel_l2 Float: l2 penalty on weights. Default = 0

activation_l1 Float: l1 penalty on layer output. Default = 0

activation_l2 Float: l2 penalty on layer output. Default = 0

batch.normalization
Logical: If TRUE, batch normalize after each hidden layer. Default = TRUE

output Character: Activation to use for output layer. Can be any as in activation. Default = "linear" for Regression, "sigmoid" for binary classification, "softmax" for multiclass

loss Character: Loss to use: Default = "mean_squared_error" for regression, "binary_crossentropy" for binary classification, "sparse_categorical_crossentropy" for multiclass

optimizer Character: Optimization to use: "rmsprop", "adadelta", "adagrad", "adam", "adamax", "nadam", "sgd". Default = "rmsprop"

learning.rate Float: learning rate. Defaults depend on optimizer used and are: rmsprop = .01, adadelta = 1, adagrad = .01, adamax = .002, adam = .001, nadam = .002, sgd = .1

metric Character: Metric used for evaluation during train. Default = "mse" for regression, "accuracy" for classification.

epochs Integer: Number of epochs. Default = 100

batch.size Integer: Batch size. Default = N of cases

validation.split
Float (0, 1): proportion of training data to use for validation. Default = .2

callback Function to be called by keras during fitting. Default = keras::callback_early_stopping(patien
= 150) for early stopping.

scale Logical: If TRUE, scale featues before training. Default = TRUE column means and standard deviation will be saved in rtMod$extra field to allow scaling ahead of prediction on new data

x.name Character: Name for feature set

y.name Character: Name for outcome

print.plot Logical: if TRUE, produce plot using mplot3 Takes precedence over plot.fitted and plot.predicted. Default = TRUE

plot.fitted Logical: if TRUE, plot True (y) vs Fitted

plot.predicted Logical: if TRUE, plot True (y.test) vs Predicted. Requires x.test and y.test

plot.theme Character: "zero", "dark", "box", "darkbox"

question Character: the question you are attempting to answer with this model, in plain language.

verbose Logical: If TRUE, print summary to screen.

outdir Path to output directory. If defined, will save Predicted vs. True plot, if available, as well as full model output, if save.mod is TRUE

save.mod	Logical: If TRUE, save all output to an RDS file in outdir save.mod is TRUE by default if an outdir is defined. If set to TRUE, and no outdir is defined, outdir defaults to paste0("./s.", mod.name)
...	Additional parameters

Details

For more information on arguments and hyperparameters, see (https://keras.rstudio.com/) and (https://keras.io/) It is important to define network structure and adjust hyperparameters based on your problem. You cannot expect defaults to work on any given dataset.

Author(s)

E.D. Gennatas

See Also

elevate for external cross-validation

Other Supervised Learning: s_ADABOOST(), s_ADDTREE(), s_BART(), s_BAYESGLM(), s_BRUTO(), s_C50(), s_CART(), s_CTREE(), s_ET(), s_EVTREE(), s_GAM.default(), s_GAM.formula(), s_GAMSELX2(), s_GAMSELX(), s_GAMSEL(), s_GAM(), s_GBM0(), s_GBM3.R(), s_GBM(), s_GLMNET(), s_GLMTREE(), s_GLM(), s_GLS(), s_H2ODL(), s_H2OGBM(), s_H2ORF(), s_IRF(), s_KNN(), s_LDA(), s_LMTREE(), s_LM(), s_MARS(), s_MLRF(), s_NBAYES(), s_NLA(), s_NLS(), s_NW(), s_POLYMARS(), s_PPR(), s_PPTREE(), s_QDA(), s_QRNN(), s_RANGER(), s_RFSRC(), s_RF(), s_SDA(), s_SGD(), s_SPLS(), s_SVM(), s_XGBLIN(), s_XGBOOST(), s_XGB()

Other Deep Learning: d_H2OAE(), s_H2ODL()

---

s_TLS *Total Least Squares Regression [R]*

Description

A minimal function to perform total least squares regression

Usage

```
s_TLS(
  x,
  y = NULL,
  x.test = NULL,
  y.test = NULL,
  x.name = "x",
  y.name = "y",
  print.plot = TRUE,
  plot.fitted = NULL,
  plot.predicted = NULL,
  plot.theme = rtTheme,
  question = NULL,
  rtclass = NULL,
  verbose = TRUE,
  outdir = NULL,
```

```
  save.mod = ifelse(!is.null(outdir), TRUE, FALSE),
  ...
)
```

Arguments

x	Numeric vector or matrix / data frame of features i.e. independent variables
y	Numeric vector of outcome, i.e. dependent variable
x.test	Numeric vector or matrix / data frame of testing set features Columns must correspond to columns in x
y.test	Numeric vector of testing set outcome
x.name	Character: Name for feature set
y.name	Character: Name for outcome
print.plot	Logical: if TRUE, produce plot using mplot3 Takes precedence over plot.fitted and plot.predicted. Default = TRUE
plot.fitted	Logical: if TRUE, plot True (y) vs Fitted
plot.predicted	Logical: if TRUE, plot True (y.test) vs Predicted. Requires x.test and y.test
plot.theme	Character: "zero", "dark", "box", "darkbox"
question	Character: the question you are attempting to answer with this model, in plain language.
rtclass	Character: Class type to use. "S3", "S4", "RC", "R6"
verbose	Logical: If TRUE, print summary to screen.
outdir	Path to output directory. If defined, will save Predicted vs. True plot, if available, as well as full model output, if save.mod is TRUE
save.mod	Logical: If TRUE, save all output to an RDS file in outdir save.mod is TRUE by default if an outdir is defined. If set to TRUE, and no outdir is defined, outdir defaults to paste0("./s.", mod.name)
...	Additional arguments

Details

The main differences between a linear model and TLS is that the latter assumes error in the features as well as the outcome. The solution is essentially the projection on the first principal axis. Because there is no model, predict and other methods are not currently working with s_TLS These features may be added in the future

Author(s)

E.D. Gennatas

s_XGBLIN *XGBoost with linear booster (wrapper for s_XGB)*

Description

Train an XGBoost learner with linear boosters

Usage

```
s_XGBLIN(
  x,
  y = NULL,
  x.test = NULL,
  y.test = NULL,
  weights = NULL,
  lambda = NULL,
  lambda.bias = 0,
  alpha = 0,
  early_stopping_rounds = 50,
  force.nrounds = NULL,
  resampler = "strat.sub",
  n.resamples = 10,
  train.p = 0.75,
  strat.n.bins = 4,
  stratify.var = NULL,
  target.length = NULL,
  seed = NULL,
  importance = TRUE,
  n.cores = 1,
  nthread = detectCores(),
  parallel.type = c("psock", "fork"),
  print.plot = TRUE,
  outdir = NULL,
  verbose = TRUE,
  xgb.verbose = FALSE,
  ...
)
```

Arguments

x	Numeric vector or matrix / data frame of features i.e. independent variables
y	Numeric vector of outcome, i.e. dependent variable
x.test	Numeric vector or matrix / data frame of testing set features Columns must correspond to columns in x
y.test	Numeric vector of testing set outcome
weights	Numeric vector: Weights for cases. For classification, weights takes precedence over ipw, therefore set weights = NULL if using ipw. Note: If weight are provided, ipw is not used. Leave NULL if setting ipw = TRUE. Default = NULL
lambda	[gS] L2 regularization on weights

alpha	[gS] L1 regularization on weights
force.nrounds	Integer: Number of rounds to run if not estimating optimal number by CV
nthread	Integer: Number of threads for xgboost using OpenMP. Only parallelize resamples using n.cores or the xgboost execution using this setting. At the moment of writing, parallelization via this parameter causes a linear booster to fail most of the times. Therefore, default is rtCores for 'gbtree', 1 for 'gblinear'
print.plot	Logical: if TRUE, produce plot using mplot3 Takes precedence over plot.fitted and plot.predicted. Default = TRUE
outdir	Path to output directory. If defined, will save Predicted vs. True plot, if available, as well as full model output, if save.mod is TRUE
verbose	Logical: If TRUE, print summary to screen.
xgb.verbose	Integer: Verbose level for XGB learners used for tuning.
...	Additional arguments to be passed to s_XGB

Details

[gS] denotes vector will trigger grid search Warning: Using nthread > 1 may lower accuracy - test on your data

Author(s)

E.D. Gennatas

See Also

elevate for external cross-validation

Other Supervised Learning: s_ADABOOST(), s_ADDTREE(), s_BART(), s_BAYESGLM(), s_BRUTO(), s_C50(), s_CART(), s_CTREE(), s_ET(), s_EVTREE(), s_GAM.default(), s_GAM.formula(), s_GAMSELX2(), s_GAMSELX(), s_GAMSEL(), s_GAM(), s_GBM0(), s_GBM3.R(), s_GBM(), s_GLMNET(), s_GLMTREE(), s_GLM(), s_GLS(), s_H2ODL(), s_H2OGBM(), s_H2ORF(), s_IRF(), s_KNN(), s_LDA(), s_LMTREE(), s_LM(), s_MARS(), s_MLRF(), s_NBAYES(), s_NLA(), s_NLS(), s_NW(), s_POLYMARS(), s_PPR(), s_PPTREE(), s_QDA(), s_QRNN(), s_RANGER(), s_RFSRC(), s_RF(), s_SDA(), s_SGD(), s_SPLS(), s_SVM(), s_TFN(), s_XGBOOST(), s_XGB()

---

s_XGBOOST *XGBoost Classification and Regression [C, R]*

Description

Tune hyperparameters using grid search and resampling, train a final model, and validate it

Usage

```
s_XGBOOST(
  x,
  y = NULL,
  x.test = NULL,
  y.test = NULL,
  x.name = NULL,
```

```
    y.name = NULL,
    booster = c("gbtree", "gblinear", "dart"),
    missing = NA,
    nrounds = 500L,
    force.nrounds = NULL,
    weights = NULL,
    ipw = TRUE,
    ipw.type = 2,
    upsample = FALSE,
    downsample = FALSE,
    resample.seed = NULL,
    obj = NULL,
    feval = NULL,
    xgb.verbose = NULL,
    print_every_n = 100L,
    early_stopping_rounds = 50L,
    eta = 0.1,
    gamma = 0,
    max_depth = 3,
    min_child_weight = 5,
    max_delta_step = 0,
    subsample = 0.75,
    colsample_bytree = 1,
    colsample_bylevel = 1,
    lambda = 0,
    lambda_bias = 0,
    alpha = 0,
    tree_method = "auto",
    sketch_eps = 0.03,
    num_parallel_tree = 1,
    base_score = NULL,
    objective = NULL,
    sample_type = "uniform",
    normalize_type = "forest",
    rate_drop = 0,
    one_drop = 0,
    skip_drop = 0,
    resampler = "strat.sub",
    n.resamples = 10,
    train.p = 0.75,
    strat.n.bins = 4,
    stratify.var = NULL,
    target.length = NULL,
    seed = NULL,
    error.curve = FALSE,
    plot.res = TRUE,
    save.res = FALSE,
    .gs = FALSE,
    grid.resample.rtset = rtset.resample("kfold", 5),
    grid.search.type = "exhaustive",
    metric = NULL,
    maximize = NULL,
```

```
  importance = FALSE,
  print.plot = TRUE,
  plot.fitted = NULL,
  plot.predicted = NULL,
  plot.theme = rtTheme,
  question = NULL,
  rtclass = NULL,
  save.dump = FALSE,
  verbose = TRUE,
  grid.verbose = FALSE,
  trace = 0,
  save.gridrun = FALSE,
  n.cores = 1,
  nthread = NULL,
  parallel.type = c("psock", "fork"),
  outdir = NULL,
  save.mod = ifelse(!is.null(outdir), TRUE, FALSE),
  ...
)
```

Arguments

x	Numeric vector or matrix / data frame of features i.e. independent variables
y	Numeric vector of outcome, i.e. dependent variable
x.test	Numeric vector or matrix / data frame of testing set features Columns must correspond to columns in x
y.test	Numeric vector of testing set outcome
x.name	Character: Name for feature set
y.name	Character: Name for outcome
booster	Character: Booster to use. Options: "gbtree", "gblinear"
missing	String or Numeric: Which values to consider as missing. Default = NA
nrounds	Integer: Maximum number of rounds to run. Can be set to a high number as early stopping will limit nrounds by monitoring inner CV error
force.nrounds	Integer: Number of rounds to run if not estimating optimal number by CV
weights	Numeric vector: Weights for cases. For classification, weights takes precedence over ipw, therefore set weights = NULL if using ipw. Note: If weight are provided, ipw is not used. Leave NULL if setting ipw = TRUE. Default = NULL
ipw	Logical: If TRUE, apply inverse probability weighting (for Classification only). Note: If weights are provided, ipw is not used. Default = TRUE
ipw.type	Integer 0, 1, 2 1: class.weights as in 0, divided by max(class.weights) 2: class.weights as in 0, divided by min(class.weights) Default = 2
upsample	Logical: If TRUE, upsample cases to balance outcome classes (for Classification only) Note: upsample will randomly sample with replacement if the length of the majority class is more than double the length of the class you are upsampling, thereby introducing randomness
downsample	Logical: If TRUE, downsample majority class to match size of minority class
resample.seed	Integer: If provided, will be used to set the seed during upsampling. Default = NULL (random seed)

obj	Function: Custom objective function. See ?xgboost::xgboost
feval	Function: Custom evaluation function. See ?xgboost::xgboost
xgb.verbose	Integer: Verbose level for XGB learners used for tuning.
print_every_n	Integer: Print evaluation metrics every this many iterations
early_stopping_rounds	Integer: Training on resamples of x.train (tuning) will stop if performance does not improve for this many rounds
eta	[gS] Numeric (0, 1): Learning rate. Default = .1
gamma	[gS] Numeric: Minimum loss reduction required to make further partition
max_depth	[gS] Integer: Maximum tree depth. (Default = 6)
subsample	[gS] Numeric: subsample ratio of the training instance
colsample_bytree	[gS] Numeric: subsample ratio of columns when constructing each tree
colsample_bylevel	[gS] Numeric
lambda	[gS] L2 regularization on weights
lambda_bias	[gS] for *linear* booster: L2 regularization on bias
alpha	[gS] L1 regularization on weights
tree_method	[gS] XGBoost tree construction algorithm (Default = "auto")
sketch_eps	[gS] Numeric (0, 1):
num_parallel_tree	Integer: N of trees to grow in parallel: Results in Random Forest -like algorithm. (Default = 1; i.e. regular boosting)
base_score	Numeric: The mean outcome response (Defaults to mean)
objective	(Default = NULL)
sample_type	Character. Default = "uniform"
normalize_type	Character. Default = "forest"
print.plot	Logical: if TRUE, produce plot using mplot3 Takes precedence over plot.fitted and plot.predicted. Default = TRUE
plot.fitted	Logical: if TRUE, plot True (y) vs Fitted
plot.predicted	Logical: if TRUE, plot True (y.test) vs Predicted. Requires x.test and y.test
plot.theme	Character: "zero", "dark", "box", "darkbox"
question	Character: the question you are attempting to answer with this model, in plain language.
rtclass	Character: Class type to use. "S3", "S4", "RC", "R6"
verbose	Logical: If TRUE, print summary to screen.
trace	Integer: If higher than 0, will print more information to the console. Default = 0
nthread	Integer: Number of threads for xgboost using OpenMP. Only parallelize resamples using n.cores or the xgboost execution using this setting. At the moment of writing, parallelization via this parameter causes a linear booster to fail most of the times. Therefore, default is rtCores for 'gbtree', 1 for 'gblinear'
outdir	Path to output directory. If defined, will save Predicted vs. True plot, if available, as well as full model output, if save.mod is TRUE
save.mod	Logical: If TRUE, save all output to an RDS file in outdir save.mod is TRUE by default if an outdir is defined. If set to TRUE, and no outdir is defined, outdir defaults to paste0("./s.", mod.name)
...	Additional arguments

Details

[gS]: indicates parameter will be autotuned by grid search if multiple values are passed. Learn more about XGBoost's parameters here: http://xgboost.readthedocs.io/en/latest/parameter.html

Value

rtMod object

Author(s)

E.D. Gennatas

See Also

elevate for external cross-validation

Other Supervised Learning: s_ADABOOST(), s_ADDTREE(), s_BART(), s_BAYESGLM(), s_BRUTO(), s_C50(), s_CART(), s_CTREE(), s_ET(), s_EVTREE(), s_GAM.default(), s_GAM.formula(), s_GAMSELX2(), s_GAMSELX(), s_GAMSEL(), s_GAM(), s_GBM0(), s_GBM3.R(), s_GBM(), s_GLMNET(), s_GLMTREE(), s_GLM(), s_GLS(), s_H2ODL(), s_H2OGBM(), s_H2ORF(), s_IRF(), s_KNN(), s_LDA(), s_LMTREE(), s_LM(), s_MARS(), s_MLRF(), s_NBAYES(), s_NLA(), s_NLS(), s_NW(), s_POLYMARS(), s_PPR(), s_PPTREE(), s_QDA(), s_QRNN(), s_RANGER(), s_RFSRC(), s_RF(), s_SDA(), s_SGD(), s_SPLS(), s_SVM(), s_TFN(), s_XGBLIN(), s_XGB()

Other Tree-based methods: s_ADABOOST(), s_ADDTREE(), s_BART(), s_C50(), s_CART(), s_CTREE(), s_ET(), s_EVTREE(), s_GBM0(), s_GBM3.R(), s_GBM(), s_GLMTREE(), s_H2OGBM(), s_H2ORF(), s_IRF(), s_LMTREE(), s_MLRF(), s_PPTREE(), s_RANGER(), s_RFSRC(), s_RF(), s_XGB()

---

table1 *Table 1*

Description

Build Table 1. Subject characteristics

Usage

```
table1(
  x,
  summaryFn1 = mean,
  summaryFn2 = sd,
  summaryFn1.extraArgs = list(na.rm = TRUE),
  summaryFn2.extraArgs = list(na.rm = TRUE),
  labelify = TRUE,
  verbose = TRUE,
  filename = NULL
)
```

Arguments

x	data.frame or matrix: Input data, cases by features
summaryFn1	Function: Summary function 1. Default = mean. See Details
summaryFn2	Function: Summary function 2. Default = sd. See Details
labelify	Logical: If TRUE, apply labelify to column names of x
verbose	Logical: If TRUE, print messages to console. Default = TRUE

Details

The output will look like "summaryFn1 (summaryFn2)". Using defaults this will be "mean (sd)"

Value

A data.frame, invisibly, with two columns: "Feature", "Value mean (sd) | N"

Author(s)

E.D. Gennatas

Examples

```
table1(iris)
```

---

themes *Print available rtemis themes*

Description

Print available rtemis themes

Usage

```
themes()
```

---

theme_black *Themes for* mplot3 *and* dplot3 *functions*

Description

Themes for mplot3 and dplot3 functions

Usage

```
theme_black(
  bg = "#000000",
  plot.bg = "transparent",
  fg = "#ffffff",
  pch = 16,
  cex = 1,
  lwd = 2,
  bty = "n",
  box.col = fg,
  box.alpha = 1,
  box.lty = 1,
  box.lwd = 0.5,
  grid = FALSE,
  grid.nx = NULL,
  grid.ny = NULL,
  grid.col = fg,
  grid.alpha = 0.2,
  grid.lty = 1,
  grid.lwd = 1,
  axes.visible = TRUE,
  axes.col = "transparent",
  tick.col = fg,
  tick.alpha = 0.5,
  tick.labels.col = fg,
  tck = -0.01,
  tcl = NA,
  x.axis.side = 1,
  y.axis.side = 2,
  labs.col = fg,
  x.axis.line = 0,
  x.axis.las = 0,
  x.axis.padj = -1.1,
  x.axis.hadj = 0.5,
  y.axis.line = 0,
  y.axis.las = 1,
  y.axis.padj = 0.5,
  y.axis.hadj = 0.5,
  xlab.line = 1.4,
  ylab.line = 2,
  zerolines = TRUE,
  zerolines.col = fg,
  zerolines.alpha = 0.5,
  zerolines.lty = 1,
  zerolines.lwd = 1,
  main.line = 0.25,
  main.adj = 0,
  main.font = 2,
  main.col = fg,
  font.family = getOption("rt.font", "Helvetica")
)
```

```
theme_blackgrid(
  bg = "#000000",
  plot.bg = "transparent",
  fg = "#ffffff",
  pch = 16,
  cex = 1,
  lwd = 2,
  bty = "n",
  box.col = fg,
  box.alpha = 1,
  box.lty = 1,
  box.lwd = 0.5,
  grid = TRUE,
  grid.nx = NULL,
  grid.ny = NULL,
  grid.col = fg,
  grid.alpha = 0.2,
  grid.lty = 1,
  grid.lwd = 1,
  axes.visible = TRUE,
  axes.col = "transparent",
  tick.col = fg,
  tick.alpha = 1,
  tick.labels.col = fg,
  tck = -0.01,
  tcl = NA,
  x.axis.side = 1,
  y.axis.side = 2,
  labs.col = fg,
  x.axis.line = 0,
  x.axis.las = 0,
  x.axis.padj = -1.1,
  x.axis.hadj = 0.5,
  y.axis.line = 0,
  y.axis.las = 1,
  y.axis.padj = 0.5,
  y.axis.hadj = 0.5,
  xlab.line = 1.4,
  ylab.line = 2,
  zerolines = TRUE,
  zerolines.col = fg,
  zerolines.alpha = 0.5,
  zerolines.lty = 1,
  zerolines.lwd = 1,
  main.line = 0.25,
  main.adj = 0,
  main.font = 2,
  main.col = fg,
  font.family = getOption("rt.font", "Helvetica")
)

theme_blackigrid(
```

```
  bg = "#000000",
  plot.bg = "#1A1A1A",
  fg = "#ffffff",
  pch = 16,
  cex = 1,
  lwd = 2,
  bty = "n",
  box.col = fg,
  box.alpha = 1,
  box.lty = 1,
  box.lwd = 0.5,
  grid = TRUE,
  grid.nx = NULL,
  grid.ny = NULL,
  grid.col = bg,
  grid.alpha = 1,
  grid.lty = 1,
  grid.lwd = 1,
  axes.visible = TRUE,
  axes.col = "transparent",
  tick.col = fg,
  tick.alpha = 1,
  tick.labels.col = fg,
  tck = -0.01,
  tcl = NA,
  x.axis.side = 1,
  y.axis.side = 2,
  labs.col = fg,
  x.axis.line = 0,
  x.axis.las = 0,
  x.axis.padj = -1.1,
  x.axis.hadj = 0.5,
  y.axis.line = 0,
  y.axis.las = 1,
  y.axis.padj = 0.5,
  y.axis.hadj = 0.5,
  xlab.line = 1.4,
  ylab.line = 2,
  zerolines = TRUE,
  zerolines.col = fg,
  zerolines.alpha = 0.5,
  zerolines.lty = 1,
  zerolines.lwd = 1,
  main.line = 0.25,
  main.adj = 0,
  main.font = 2,
  main.col = fg,
  font.family = getOption("rt.font", "Helvetica")
)

theme_darkgray(
  bg = "#121212",
```

```
  plot.bg = "transparent",
  fg = "#ffffff",
  pch = 16,
  cex = 1,
  lwd = 2,
  bty = "n",
  box.col = fg,
  box.alpha = 1,
  box.lty = 1,
  box.lwd = 0.5,
  grid = FALSE,
  grid.nx = NULL,
  grid.ny = NULL,
  grid.col = fg,
  grid.alpha = 0.2,
  grid.lty = 1,
  grid.lwd = 1,
  axes.visible = TRUE,
  axes.col = "transparent",
  tick.col = fg,
  tick.alpha = 0.5,
  tick.labels.col = fg,
  tck = -0.01,
  tcl = NA,
  x.axis.side = 1,
  y.axis.side = 2,
  labs.col = fg,
  x.axis.line = 0,
  x.axis.las = 0,
  x.axis.padj = -1.1,
  x.axis.hadj = 0.5,
  y.axis.line = 0,
  y.axis.las = 1,
  y.axis.padj = 0.5,
  y.axis.hadj = 0.5,
  xlab.line = 1.4,
  ylab.line = 2,
  zerolines = TRUE,
  zerolines.col = fg,
  zerolines.alpha = 0.5,
  zerolines.lty = 1,
  zerolines.lwd = 1,
  main.line = 0.25,
  main.adj = 0,
  main.font = 2,
  main.col = fg,
  font.family = getOption("rt.font", "Helvetica")
)

theme_darkgraygrid(
  bg = "#121212",
  plot.bg = "transparent",
```

```
  fg = "#ffffff",
  pch = 16,
  cex = 1,
  lwd = 2,
  bty = "n",
  box.col = fg,
  box.alpha = 1,
  box.lty = 1,
  box.lwd = 0.5,
  grid = TRUE,
  grid.nx = NULL,
  grid.ny = NULL,
  grid.col = "#404040",
  grid.alpha = 1,
  grid.lty = 1,
  grid.lwd = 1,
  axes.visible = TRUE,
  axes.col = "transparent",
  tick.col = "#00000000",
  tick.alpha = 1,
  tick.labels.col = fg,
  tck = -0.01,
  tcl = NA,
  x.axis.side = 1,
  y.axis.side = 2,
  labs.col = fg,
  x.axis.line = 0,
  x.axis.las = 0,
  x.axis.padj = -1.1,
  x.axis.hadj = 0.5,
  y.axis.line = 0,
  y.axis.las = 1,
  y.axis.padj = 0.5,
  y.axis.hadj = 0.5,
  xlab.line = 1.4,
  ylab.line = 2,
  zerolines = TRUE,
  zerolines.col = fg,
  zerolines.alpha = 0.5,
  zerolines.lty = 1,
  zerolines.lwd = 1,
  main.line = 0.25,
  main.adj = 0,
  main.font = 2,
  main.col = fg,
  font.family = getOption("rt.font", "Helvetica")
)

theme_darkgrayigrid(
  bg = "#121212",
  plot.bg = "#202020",
  fg = "#ffffff",
```

```
  pch = 16,
  cex = 1,
  lwd = 2,
  bty = "n",
  box.col = fg,
  box.alpha = 1,
  box.lty = 1,
  box.lwd = 0.5,
  grid = TRUE,
  grid.nx = NULL,
  grid.ny = NULL,
  grid.col = bg,
  grid.alpha = 1,
  grid.lty = 1,
  grid.lwd = 1,
  axes.visible = TRUE,
  axes.col = "transparent",
  tick.col = "transparent",
  tick.alpha = 1,
  tick.labels.col = fg,
  tck = -0.01,
  tcl = NA,
  x.axis.side = 1,
  y.axis.side = 2,
  labs.col = fg,
  x.axis.line = 0,
  x.axis.las = 0,
  x.axis.padj = -1.1,
  x.axis.hadj = 0.5,
  y.axis.line = 0,
  y.axis.las = 1,
  y.axis.padj = 0.5,
  y.axis.hadj = 0.5,
  xlab.line = 1.4,
  ylab.line = 2,
  zerolines = TRUE,
  zerolines.col = fg,
  zerolines.alpha = 0.5,
  zerolines.lty = 1,
  zerolines.lwd = 1,
  main.line = 0.25,
  main.adj = 0,
  main.font = 2,
  main.col = fg,
  font.family = getOption("rt.font", "Helvetica")
)

theme_white(
  bg = "#ffffff",
  plot.bg = "transparent",
  fg = "#000000",
  pch = 16,
```

```
  cex = 1,
  lwd = 2,
  bty = "n",
  box.col = fg,
  box.alpha = 1,
  box.lty = 1,
  box.lwd = 0.5,
  grid = FALSE,
  grid.nx = NULL,
  grid.ny = NULL,
  grid.col = fg,
  grid.alpha = 1,
  grid.lty = 1,
  grid.lwd = 1,
  axes.visible = TRUE,
  axes.col = "transparent",
  tick.col = fg,
  tick.alpha = 0.5,
  tick.labels.col = fg,
  tck = -0.01,
  tcl = NA,
  x.axis.side = 1,
  y.axis.side = 2,
  labs.col = fg,
  x.axis.line = 0,
  x.axis.las = 0,
  x.axis.padj = -1.1,
  x.axis.hadj = 0.5,
  y.axis.line = 0,
  y.axis.las = 1,
  y.axis.padj = 0.5,
  y.axis.hadj = 0.5,
  xlab.line = 1.4,
  ylab.line = 2,
  zerolines = TRUE,
  zerolines.col = fg,
  zerolines.alpha = 0.5,
  zerolines.lty = 1,
  zerolines.lwd = 1,
  main.line = 0.25,
  main.adj = 0,
  main.font = 2,
  main.col = fg,
  font.family = getOption("rt.font", "Helvetica")
)

theme_whitegrid(
  bg = "#ffffff",
  plot.bg = "transparent",
  fg = "#000000",
  pch = 16,
  cex = 1,
```

```
  lwd = 2,
  bty = "n",
  box.col = fg,
  box.alpha = 1,
  box.lty = 1,
  box.lwd = 0.5,
  grid = TRUE,
  grid.nx = NULL,
  grid.ny = NULL,
  grid.col = "#c0c0c0",
  grid.alpha = 1,
  grid.lty = 1,
  grid.lwd = 1,
  axes.visible = TRUE,
  axes.col = "transparent",
  tick.col = "#00000000",
  tick.alpha = 1,
  tick.labels.col = fg,
  tck = -0.01,
  tcl = NA,
  x.axis.side = 1,
  y.axis.side = 2,
  labs.col = fg,
  x.axis.line = 0,
  x.axis.las = 0,
  x.axis.padj = -1.1,
  x.axis.hadj = 0.5,
  y.axis.line = 0,
  y.axis.las = 1,
  y.axis.padj = 0.5,
  y.axis.hadj = 0.5,
  xlab.line = 1.4,
  ylab.line = 2,
  zerolines = TRUE,
  zerolines.col = fg,
  zerolines.alpha = 0.5,
  zerolines.lty = 1,
  zerolines.lwd = 1,
  main.line = 0.25,
  main.adj = 0,
  main.font = 2,
  main.col = fg,
  font.family = getOption("rt.font", "Helvetica")
)

theme_whiteigrid(
  bg = "#ffffff",
  plot.bg = "#E6E6E6",
  fg = "#000000",
  pch = 16,
  cex = 1,
  lwd = 2,
```

```
  bty = "n",
  box.col = fg,
  box.alpha = 1,
  box.lty = 1,
  box.lwd = 0.5,
  grid = TRUE,
  grid.nx = NULL,
  grid.ny = NULL,
  grid.col = bg,
  grid.alpha = 1,
  grid.lty = 1,
  grid.lwd = 1,
  axes.visible = TRUE,
  axes.col = "transparent",
  tick.col = "transparent",
  tick.alpha = 1,
  tick.labels.col = fg,
  tck = -0.01,
  tcl = NA,
  x.axis.side = 1,
  y.axis.side = 2,
  labs.col = fg,
  x.axis.line = 0,
  x.axis.las = 0,
  x.axis.padj = -1.1,
  x.axis.hadj = 0.5,
  y.axis.line = 0,
  y.axis.las = 1,
  y.axis.padj = 0.5,
  y.axis.hadj = 0.5,
  xlab.line = 1.4,
  ylab.line = 2,
  zerolines = TRUE,
  zerolines.col = fg,
  zerolines.alpha = 0.5,
  zerolines.lty = 1,
  zerolines.lwd = 1,
  main.line = 0.25,
  main.adj = 0,
  main.font = 2,
  main.col = fg,
  font.family = getOption("rt.font", "Helvetica")
)

theme_lightgraygrid(
  bg = "#dfdfdf",
  plot.bg = "transparent",
  fg = "#000000",
  pch = 16,
  cex = 1,
  lwd = 2,
  bty = "n",
```

```
    box.col = fg,
    box.alpha = 1,
    box.lty = 1,
    box.lwd = 0.5,
    grid = TRUE,
    grid.nx = NULL,
    grid.ny = NULL,
    grid.col = "#c0c0c0",
    grid.alpha = 1,
    grid.lty = 1,
    grid.lwd = 1,
    axes.visible = TRUE,
    axes.col = "transparent",
    tick.col = "#00000000",
    tick.alpha = 1,
    tick.labels.col = fg,
    tck = -0.01,
    tcl = NA,
    x.axis.side = 1,
    y.axis.side = 2,
    labs.col = fg,
    x.axis.line = 0,
    x.axis.las = 0,
    x.axis.padj = -1.1,
    x.axis.hadj = 0.5,
    y.axis.line = 0,
    y.axis.las = 1,
    y.axis.padj = 0.5,
    y.axis.hadj = 0.5,
    xlab.line = 1.4,
    ylab.line = 2,
    zerolines = TRUE,
    zerolines.col = fg,
    zerolines.alpha = 0.5,
    zerolines.lty = 1,
    zerolines.lwd = 1,
    main.line = 0.25,
    main.adj = 0,
    main.font = 2,
    main.col = fg,
    font.family = getOption("rt.font", "Helvetica")
  )

  theme_mediumgraygrid(
    bg = "#b3b3b3",
    plot.bg = "transparent",
    fg = "#000000",
    pch = 16,
    cex = 1,
    lwd = 2,
    bty = "n",
    box.col = fg,
```

```
  box.alpha = 1,
  box.lty = 1,
  box.lwd = 0.5,
  grid = TRUE,
  grid.nx = NULL,
  grid.ny = NULL,
  grid.col = "#d0d0d0",
  grid.alpha = 1,
  grid.lty = 1,
  grid.lwd = 1,
  axes.visible = TRUE,
  axes.col = "transparent",
  tick.col = "#00000000",
  tick.alpha = 1,
  tick.labels.col = fg,
  tck = -0.01,
  tcl = NA,
  x.axis.side = 1,
  y.axis.side = 2,
  labs.col = fg,
  x.axis.line = 0,
  x.axis.las = 0,
  x.axis.padj = -1.1,
  x.axis.hadj = 0.5,
  y.axis.line = 0,
  y.axis.las = 1,
  y.axis.padj = 0.5,
  y.axis.hadj = 0.5,
  xlab.line = 1.4,
  ylab.line = 2,
  zerolines = TRUE,
  zerolines.col = fg,
  zerolines.alpha = 0.5,
  zerolines.lty = 1,
  zerolines.lwd = 1,
  main.line = 0.25,
  main.adj = 0,
  main.font = 2,
  main.col = fg,
  font.family = getOption("rt.font", "Helvetica")
)
```

Arguments

bg	Color: Figure background
plot.bg	Color: Plot region background
fg	Color: Foreground color used as default for multiple elements like axes and labels, which can be defined separately
pch	Integer: Point character. Default = 16
cex	Float: Character expansion factor. Default = 1.2
lwd	Float: Line width. Default = 2

bty	Character: Box type: "o", "l", "7", "c", "u", or "]", or "n". Default = "n" (no box)
box.col	Box color if bty != "n"
box.alpha	Float: Box alpha
box.lty	Integer: Box line type
box.lwd	Float: Box line width
grid	Logical: If TRUE, draw grid in plot regions
grid.nx	Integer: N of vertical grid lines
grid.ny	Integer: N of horizontal grid lines
grid.col	Grid color
grid.alpha	Float: Grid alpha
grid.lty	Integer: Grid line type
grid.lwd	Float: Grid line width
axes.visible	Logical: If TRUE, draw axes
axes.col	Axes colors
tick.col	Tick color
tick.alpha	Float: Tick alpha
tick.labels.col	Tick labels' color
tck	graphic::parr's tck argument: Tick length, can be negative
tcl	graphic::parr's tcl argument
x.axis.side	Integer: Side to place x-axis. Default = 1 (bottom)
y.axis.side	Integer: Side to place y-axis. Default = 2 (left)
labs.col	Labels' color
x.axis.line	Numeric: axis's line argument for the x-axis
x.axis.las	Numeric: axis's las argument for the x-axis
x.axis.padj	Numeric: x-axis' padj: Adjustment for the x-axis tick labels' position
x.axis.hadj	Numeric: x-axis' hadj
y.axis.line	Numeric: axis's line argument for the y-axis
y.axis.las	Numeric: axis's las argument for the y-axis
y.axis.padj	Numeric: y-axis' padj
y.axis.hadj	Numeric: y-axis' hadj
xlab.line	Numeric: Line to place xlab
ylab.line	Numeric: Line to place ylab
zerolines	Logical: If TRUE, draw lines on x = 0, y = 0, if within plot limits
zerolines.col	Zerolines color
zerolines.alpha	Float: Zerolines alpha
zerolines.lty	Integer: Zerolines line type
zerolines.lwd	Float: Zerolines line width
main.line	Float: How many lines away from the plot region to draw title. Default = .5
main.adj	Float: How to align title. Default = 0 (left-align)
main.font	Integer: 1: Regular, 2: Bold
main.col	Title color
font.family	Character: Font to be used throughout plot.

timeProc *Time a process*

Description

timeProc measures how long it takes for a process to run

Usage

```
timeProc(..., verbose = TRUE)
```

Arguments

...	Command to be timed. (Will be converted using as.expression)
verbose	Logical: If TRUE, print messages to console

Author(s)

E.D. Gennatas

typeset *Set type of columns*

Description

Given an index of columns, convert identified columns of data.frame to factor, ordered factor, or integer. A number of datasets are distributed with an accompanying index of this sort, especially to define which variables should be treated as categorical (here, factors) for predicting modeling. This functions aims to make data type conversions in those cases easier.

Usage

```
typeset(
  x,
  factor.index = NULL,
  orderedfactor.index = NULL,
  integer.index = NULL
)
```

Arguments

x	data frame: input whose columns' types you want to edit
factor.index	Integer, vector: Index of columns to be converted to factors using factor(x)
orderedfactor.index	Integer, vector: Index of columns to be converted to ordered factors using factor(x, ordered = TRUE)
integer.index	Integer, vector: Index of columns to be converted to integers using as.integer

Author(s)

E.D. Gennatas

uniquevalsperfeat *Unique values per feature*

Description

Get number of unique values per features

Usage

```
uniquevalsperfeat(x)
```

Arguments

x	matrix or data frame input

Value

Vector, integer of length `NCOL(x)` with number of unique values per column/feature

Author(s)

E.D. Gennatas

Examples

```
## Not run:
uniquevalsperfeat(iris)

## End(Not run)
```

update.cartLinBoostTV **rtemis** *internals: Update cartLinBoostTV object's fitted values*

Description

Calculate new fitted values for a cartLinBoostTV object. Advanced use only: run with new `x` or after updating learning.rate in object

Usage

```
## S3 method for class 'cartLinBoostTV'
update(
  object,
  x = NULL,
  x.valid = NULL,
  trace = 0,
  last.step.only = FALSE,
  n.cores = rtCores,
  ...
)
```

Arguments

`object`	cartLinBoostTV object
`x`	Data frame: Features
`last.step.only`	Logical: If TRUE, `x` must be provided and only the last meta model will be updated using this `x`

Value

cartLinBoostTV object

Nothing; updates `object` in-place

Author(s)

E.D. Gennatas

`update.cartLiteBoostTV`

rtemis *internals: Update cartLiteBoostTV object's fitted values*

Description

Calculate new fitted values for a cartLiteBoostTV object. Advanced use only: run with new `x` or after updating learning.rate in object

Usage

```
## S3 method for class 'cartLiteBoostTV'
update(
  object,
  x = NULL,
  x.valid = NULL,
  trace = 0,
  last.step.only = FALSE,
  n.cores = rtCores,
  ...
)
```

Arguments

`object`	cartLiteBoostTV object
`x`	Data frame: Features
`last.step.only`	Logical: If TRUE, `x` must be provided and only the last meta model will be updated using this `x`

Value

cartLiteBoostTV object

Nothing; updates `object` in-place

Author(s)

E.D. Gennatas

`update.glmLiteBoostTV` **rtemis** *internals: Update glmLiteBoostTV object's fitted values*

Description

Calculate new fitted values for a glmLiteBoostTV object. Advanced use only: run with new `x` or after updating learning.rate in object

Usage

```
## S3 method for class 'glmLiteBoostTV'
update(
  object,
  x = NULL,
  x.valid = NULL,
  trace = 0,
  last.step.only = FALSE,
  n.cores = rtCores,
  ...
)
```

Arguments

`object`	glmLiteBoostTV object
`x`	Data frame: Features
`last.step.only`	Logical: If TRUE, `x` must be provided and only the last meta model will be updated using this `x`

Value

glmLiteBoostTV object

Nothing; updates `object` in-place

Author(s)

E.D. Gennatas

update.rtMod.boost — **rtemis** *internals: Update* rtMod *boost object's fitted values in-place*

Description

Calculate new fitted values for a boost object. Advanced use only: run with new x or after updating learning.rate in object

Usage

```
## S3 method for class 'rtMod.boost'
update(
  object,
  x = NULL,
  x.test = NULL,
  trace = 0,
  last.step.only = FALSE,
  n.cores = rtCores,
  ...
)
```

Arguments

object	boost object
x	Data frame: Features
last.step.only	Logical: If TRUE, x must be provided and only the last meta model will be updated using this x

Value

boost object

Nothing; updates object in-place

Author(s)

E.D. Gennatas

varSelect — *Variable Selection by Variable Importance*

Description

Select important variables from a set of features based on RANGER- or XGBLIN-estimated variable importance

Usage

```
varSelect(
  x,
  y,
  method = c("RANGER", "XGBLIN"),
  xgb.params = list(alpha = 0.1, lambda = 0.1),
  p = 0.2,
  print.plot = TRUE,
  verbose = TRUE
)
```

Arguments

x	Matrix / Data Frame of Predictors
y	Outcome vector
method	Character: "RANGER", "XGBLIN": Learner to use for estimating variable importance. Default = "RANGER"
xgb.params	List of parameters for `method = "XGBLIN"`
p	Float (0, 1): Fraction of variables in x to select. `p * ncol(x)`. May help to set to a fraction twice what you expect to be the true fraction of useful variables, to reduce false negatives at the expense of false positives which can be dealt by an appropriate learning algorithm. (Default = .2)
print.plot	Logical: If TRUE, print index plot of variable importance using mplot3_x
verbose	Logical: If TRUE, print messages to screen

Author(s)

E.D. Gennatas

---

winsorize *Winsorize vector*

Description

Replace extreme values by absolute or quantile threshold

Usage

```
winsorize(
  x,
  lo = NULL,
  hi = NULL,
  prob.lo = 0.025,
  prob.hi = 0.975,
  quantile.type = 7,
  verbose = TRUE
)
```

Arguments

x	Numeric vector: Input data
lo	Numeric: If not NULL, replace any values in x lower than this with this. Default = NULL
hi	Numeric: If not NULL, replace any values in x higher than this with this.
prob.lo	Numeric (0, 1): If not NULL and lo = NULL, find sample quantile that corresponds to this probability and set as lo.
prob.hi	Numeric (0, 1): If not NULL and hi = NULL, find sample quantile that corresponds to this probability and set as hi.
quantile.type	Integer: passed to stats::quantile
verbose	Logical: If TRUE, print messages to console.

Details

If both lo and prob.lo or both hi and prob.hi are NULL, cut-off is set to min(x) and max(x) respectively, i.e. no values are changed

Author(s)

E.D. Gennatas

Examples

```
# Winsorize a normally distributed variable
x <- rnorm(500)
xw <- winsorize(x)
# Winsorize an exponentially distributed variable only on
# the top 5% highest values
x <- rexp(500)
xw <- winsorize(x, prob.lo = NULL, prob.hi = .95)
```

---

xdecomSelect *Select* **rtemis** *cross-decomposer*

Description

Accepts decomposer name (supports abbreviations) and returns **rtemis** function name or the function itself. If run with no parameters, prints list of available algorithms.

Usage

```
xdecomSelect(xdecom, fn = FALSE, desc = FALSE)
```

Arguments

xdecom	Character: Cross-decomposition name; case insensitive
fn	Logical: If TRUE, return function, otherwise return name of function. Default = FALSE
desc	Logical: If TRUE, return full name of algorithm. Default = FALSE

Value

Function or name of function (see param `fn`) or full name of algorithm (`desc`)

Author(s)

E.D. Gennatas

See Also

Other Cross-Decomposition: `x_CCA()`

`x_CCA` *Sparse Canonical Correlation Analysis (CCA)*

Description

Run a sparse Canonical Correlation Analysis using the `PMA` package

Usage

```
x_CCA(
  x,
  z,
  x.test = NULL,
  z.test = NULL,
  y = NULL,
  outcome = NULL,
  k = 3,
  niter = 20,
  nperms = 50,
  permute.niter = 15,
  typex = "standard",
  typez = "standard",
  penaltyx = NULL,
  penaltyz = NULL,
  standardize = TRUE,
  upos = FALSE,
  vpos = FALSE,
  verbose = TRUE,
  n.cores = rtCores,
  outdir = NULL,
  save.mod = ifelse(!is.null(outdir), TRUE, FALSE),
  ...
)
```

Arguments

`x`	Matrix: Training x dataset
`z`	Matrix: Training z dataset
`x.test`	Matrix (Optional): Testing x set

z.test	Matrix (Optional): Testing z set
y	Outcome vector (Optional): If supplied, linear combinations of x and z need to be additionaly correlated with this
outcome	Character: Type of outcome y: "survival", "multiclass", "quantitative"
k	Integer: Number of components
niter	Integer: Number of iterations
nperms	Integer: Number of permutations to run with CCA.permute The higher, the better.
permute.niter	Integer: Number of iterations to run for each permutation with CCA.permute
typex	Character: "standard", "ordered". Use "standard" if columns of x are unordered; lasso penalty is applied to enforce sparsity. Otherwise, use "ordered"; fused lasso penalty is applied, to enforce both sparsity and smoothness.
typez	Character: "standard", "ordered". Same as typex for z dataset
penaltyx	Float: The penalty to be applied to the matrix x, i.e. the penalty that results in the canonical vector u. If typex is "standard" then the L1 bound on u is penaltyx*sqrt(ncol(x)). In this case penaltyx must be between 0 and 1 (larger L1 bound corresponds to less penalization). If "ordered" then it's the fused lasso penalty lambda, which must be non-negative (larger lambda corresponds to more penalization).
penaltyz	Float: The penalty to be applied to the matrix z, i.e. the penalty that results in the canonical vector v. If typez is "standard" then the L1 bound on v is penaltyz*sqrt(ncol(z)). In this case penaltyz must be between 0 and 1 (larger L1 bound corresponds to less penalization). If "ordered" then it's the fused lasso penalty lambda, which must be non-negative (larger lambda corresponds to more penalization).
standardize	Logical: If TRUE, center and scale columns of x and z
upos	Logical: Require elements of u to be positive
vpos	Logical: Require elements of v to be positive
verbose	Logical: Print messages, including trace from x_CCA.permute and PMA::CCA
n.cores	Integer: Number of cores to use
outdir	Path to output directory. Default = NULL
save.mod	Logical: If TRUE, and outdir is defined, will save trained CCA model to outdir. Default = TRUE if outdir is set, otherwise FALSE
...	Additional arguments to be passed to PMA::CCA

Details

#' x_CCA runs PMA::CCA. If penaltyx is NULL, penaltyx *and* penaltyz will be estimated automatically using x_CCA.permute (adapted to run in parallel)

Author(s)

E.D. Gennatas

See Also

Other Cross-Decomposition: xdecomSelect()

yhi *Get y above current plot area*

Description

Get y above current plot area

Usage

```
yhi(pct_higher = 0.08)
```

Arguments

pct_higher	Numeric: Get y this percent above top of plot

Author(s)

E.D. Gennatas

ylo *Get y below current plot area*

Description

Get y below current plot area

Usage

```
ylo(pct_lower = 0.08)
```

Arguments

pct_lower	Numeric: Get y this percent below bottom of plot

Author(s)

E.D. Gennatas

zip2longlat *Get Longitude and Lattitude for zip code(s)*

Description

Returns a data.table of longitude and lattitude for one or more zip codes, given an input dataset

Usage

```
zip2longlat(x, zipdt)
```

Arguments

x	Character vector: Zip code(s)
zipdt	data.table with "zip", "lng", and "lat" columns

zipdist *Get distance between pairs of zip codes*

Description

Get distance between pairs of zip codes

Usage

```
zipdist(x, y, zipdt)
```

Arguments

x	Character vector
y	Character vector, same length as x
zipdt	data.table with columns zip, lng, lat

Value

data.table with distances in meters

Author(s)

E.D. Gennatas

`%BC%` *Binary matrix times character vector*

Description

Binary matrix times character vector

Usage

```
x %BC% labels
```

Arguments

`x`	A binary matrix or data.frame
`labels`	Character vector length equal to `ncol(x)`

Value

a character vector

Author(s)

E.D. Gennatas

Index