Package 'FFTrees'

July 21, 2025

Type Package

Title Generate, Visualise, and Evaluate Fast-and-Frugal Decision Trees

Version 2.0.0

Date 2023-06-06

Maintainer Hansjoerg Neth <h.neth@uni.kn>

Description Create, visualize, and test fast-and-frugal decision trees (FFTs) using the algorithms and methods described by Phillips, Neth, Woike & Gaissmaier (2017), <doi:10.1017/S1930297500006239>.
FFTs are simple and transparent decision trees for solving binary classification problems.
FFTs can be preferable to more complex algorithms because they require very little information, are easy to understand and communicate, and are robust against overfitting.

LazyData true

Encoding UTF-8

Depends R(>= 3.5.0)

Imports caret, rpart, randomForest, e1071, cli, dplyr, knitr, magrittr, scales, stringr, testthat, tibble, tidyselect

Suggests rmarkdown, spelling

License CC0

URL https://CRAN.R-project.org/package=FFTrees, https://github.com/ndphillips/FFTrees/

BugReports https://github.com/ndphillips/FFTrees/issues

VignetteBuilder knitr

RoxygenNote 7.2.3

Language en-US

NeedsCompilation no

Author Nathaniel Phillips [aut] (ORCID: <https://orcid.org/0000-0002-8969-7013>), Hansjoerg Neth [aut, cre] (ORCID: <https://orcid.org/0000-0001-5427-3141>),

Jan Woike [aut] (ORCID: <https://orcid.org/0000-0002-6816-121X>),
Wolfgang Gaissmaier [aut] (ORCID:
<https://orcid.org/0000-0001-6273-178X>)

Repository CRAN

Date/Publication 2023-06-05 23:30:02 UTC

Contents

`add_fft_df` *Add an FFT definition to tree definitions*

Description

`add_fft_df` adds the definition(s) of one or more FFT(s) (in the multi-line format of an `FFTrees` object) or a single FFT (as a tidy data frame) to the multi-line FFT definitions of an `FFTrees` object.

`add_fft_df` allows for collecting and combining (sets of) tree definitions after manipulating them with other tree trimming functions.

Usage

```
add_fft_df(fft, ffts_df = NULL, quiet = FALSE)
```

Arguments

`fft`	A (set of) FFT definition(s) (in the multi-line format of an `FFTrees` object) or one FFT definition (as a data frame in tidy format, with one row per node).
`ffts_df`	A set of FFT definitions (as a data frame, usually from an `FFTrees` object, with suitable variable names to pass `verify_ffts_df`. Default: `ffts_df = NULL`.
`quiet`	Hide feedback messages (as logical)? Default: `quiet = FALSE`.

Value

A (set of) FFT definition(s) in the one line FFT definition format used by an `FFTrees` object (as a data frame).

See Also

`get_fft_df` for getting the FFT definitions of an `FFTrees` object; `read_fft_df` for reading one FFT definition from tree definitions; `write_fft_df` for writing one FFT to tree definitions; `FFTrees` for creating FFTs from and applying them to data.

Other tree definition and manipulation functions: `add_nodes()`, `drop_nodes()`, `edit_nodes()`, `flip_exits()`, `get_fft_df()`, `read_fft_df()`, `reorder_nodes()`, `select_nodes()`, `write_fft_df()`

add_nodes *Add nodes to an FFT definition*

Description

add_nodes allows adding one or more nodes to an existing FFT definition (in the tidy data frame format).

add_nodes allows to directly set and change the value(s) of class, cue, direction, threshold, and exit, in an FFT definition for the specified nodes.

There is only rudimentary verification for plausible entries. Importantly, however, as add_nodes is ignorant of data, the values of its variables are not validated for a specific set of data.

Values in nodes refer to their new position in the final FFT. Duplicate values of nodes are ignored (and only the last entry is used).

When a new exit node is added, the exit type of a former final node is set to the signal value (i.e., exit_types[2]).

Usage

```
add_nodes(
  fft,
  nodes = NA,
  class = NA,
  cue = NA,
  direction = NA,
  threshold = NA,
  exit = NA,
  quiet = FALSE
)
```

Arguments

fft	One FFT definition (as a data frame in tidy format, with one row per node).
nodes	The FFT nodes to be added (as an integer vector). Values refer to their new position in the final FFT (i.e., after adding all nodes to fft). Default: nodes = NA.
class	The class values of nodes (as character).
cue	The cue names of nodes (as character).
direction	The direction values of nodes (as character).
threshold	The threshold values of nodes (as character).
exit	The exit values of nodes (as values from exit_types).
quiet	Hide feedback messages (as logical)? Default: quiet = FALSE.

Value

One FFT definition (as a data frame in tidy format, with one row per node).

See Also

drop_nodes for deleting nodes from an FFT definition; edit_nodes for editing nodes in an FFT definition; flip_exits for reversing exits in an FFT definition; reorder_nodes for reordering nodes of an FFT definition; select_nodes for selecting nodes in an FFT definition; get_fft_df for getting the FFT definitions of an FFTrees object; read_fft_df for reading one FFT definition from tree definitions; add_fft_df for adding FFTs to tree definitions; FFTrees for creating FFTs from and applying them to data.

Other tree definition and manipulation functions: add_fft_df(), drop_nodes(), edit_nodes(), flip_exits(), get_fft_df(), read_fft_df(), reorder_nodes(), select_nodes(), write_fft_df()

---

add_stats *Add decision statistics to data (based on frequency counts of a 2x2 classification outcomes)*

Description

add_stats assumes the input of the 4 essential classification outcomes (as frequency counts in a data frame "data" with variable names "hi", "fa", "mi", and "cr") and uses them to compute various decision accuracy measures.

Usage

```
add_stats(
  data,
  correction = 0.25,
  sens.w = NULL,
  my.goal = NULL,
  my.goal.fun = NULL,
  cost.outcomes = NULL,
  cost.each = NULL
)
```

Arguments

data	A data frame with 4 frequency counts (as integer values, named "hi", "fa", "mi", and "cr").
correction	numeric. Correction added to all counts for calculating dprime. Default: correction = .25.
sens.w	numeric. Sensitivity weight (for computing weighted accuracy, wacc). Default: sens.w = NULL (to ensure that values are passed by calling function).
my.goal	Name of an optional, user-defined goal (as character string). Default: my.goal = NULL.
my.goal.fun	User-defined goal function (with 4 arguments hi fa mi cr). Default: my.goal.fun = NULL.

`cost.outcomes` list. A list of length 4 named `"hi"`, `"fa"`, `"mi"`, `"cr"`, and specifying the costs of a hit, false alarm, miss, and correct rejection, respectively. E.g.; `cost.outcomes = listc("hi" = 0, "fa" = 10, "mi" = 20, "cr" = 0)` means that a false alarm and miss cost 10 and 20 units, respectively, while correct decisions incur no costs. Default: `cost.outcomes = NULL` (to ensure that values are passed by calling function).

`cost.each` numeric. An optional fixed cost added to all outputs (e.g., the cost of using the cue). Default: `cost.each = NULL` (to ensure that values are passed by calling function).

Details

Providing numeric values for `cost.each` (as a vector) and `cost.outcomes` (as a named list) allows computing cost information for the counts of corresponding classification decisions.

Value

A data frame with variables of computed accuracy and cost measures (but dropping inputs).

`blood` *Blood donation data*

Description

Data taken from the Blood Transfusion Service Center in Hsin-Chu City in Taiwan

Usage

```
blood
```

Format

A data frame containing 748 rows and 5 columns.

recency Months since last donation

frequency Total number of donations

total Total blood donated (in c.c.)

time Months since first donation

donation.crit *Criterion*: Did the person donate blood (in March 2007)?
Values: `0`/no vs. 1/yes (76.2% vs.\ 23.8%).

Source

https://archive.ics.uci.edu/ml/datasets/Blood+Transfusion+Service+Center

Original owner and donor:

Prof. I-Cheng Yeh

Department of Information Management

Chung-Hua University

See Also

Other datasets: breastcancer, car, contraceptive, creditapproval, fertility, forestfires, heart.cost, heart.test, heart.train, heartdisease, iris.v, mushrooms, sonar, titanic, voting, wine

---

`breastcancer` *Physiological data of patients tested for breast cancer*

Description

Physiological data of patients tested for breast cancer

Usage

```
breastcancer
```

Format

A data frame containing 699 patients (rows) and 9 variables (columns).

thickness Clump Thickness

cellsize.unif Uniformity of Cell Size

cellshape.unif Uniformity of Cell Shape

adhesion Marginal Adhesion

epithelial Single Epithelial Cell Size

nuclei.bare Bare Nuclei

chromatin Bland Chromatin

nucleoli Normal Nucleoli

mitoses Mitoses

diagnosis *Criterion*: Absence/presence of breast cancer.
Values: `FALSE` vs. `TRUE` (65.0% vs.\ 35.0%).

Details

We made the following enhancements to the original data for improved usability:

- The ID number of the cases was excluded.
- The numeric criterion with value "2" for benign and "4" for malignant was converted to logical TRUE/FALSE.
- 16 cases were excluded because they contained NAs.

Other than that, the data remains consistent with the original dataset.

Source

https://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+(Original)

Original creator:

Dr. William H. Wolberg (physician)

University of Wisconsin Hospitals

Madison, Wisconsin, USA

See Also

Other datasets: blood, car, contraceptive, creditapproval, fertility, forestfires, heart.cost, heart.test, heart.train, heartdisease, iris.v, mushrooms, sonar, titanic, voting, wine

---

car *Car acceptability data*

Description

A dataset on car evaluations based on basic features, derived from a simple hierarchical decision model.

Usage

```
car
```

Format

A data frame containing 1728 cars (rows) and 7 variables (columns).

buying.price price for buying the car, Factor (high, low, med, vhigh)

maint.price price of the maintenance, Factor (high, low, med, vhigh)

doors number of doors, Factor (2, 3, 4, 5more)

persons capacity in terms of persons to carry, Factor (2, 4, more)

luggage the size of luggage boot, Factor (big, med, small)

safety estimated safety of the car, Factor (high, low, med)

acceptability *Criterion*: Category of acceptability rating.

Values: unacc/ vgood/ good/ acc

Details

The criterion variable is a car's acceptability rating.

The *criterion* for this dataset has not yet been binarized. Before using it with an *FFTree*, this necessary prerequisite step should be completed based on individual preferences.

Source

http://archive.ics.uci.edu/ml/datasets/Car+Evaluation

Original creator and donor:

Marko Bohanec and Blaz Zupan

References

Bohanec, M., Rajkovic, V. (1990): Expert system for decision making. *Sistemica*, *1* (1), 145–157.

See Also

Other datasets: blood, breastcancer, contraceptive, creditapproval, fertility, forestfires, heart.cost, heart.test, heart.train, heartdisease, iris.v, mushrooms, sonar, titanic, voting, wine

---

classtable *Compute classification statistics for binary prediction and criterion (e.g.; truth) vectors*

Description

The main input are 2 logical vectors of prediction and criterion values.

Usage

```
classtable(
  prediction_v = NULL,
  criterion_v = NULL,
  correction = 0.25,
  sens.w = NULL,
  cost.outcomes = NULL,
  cost_v = NULL,
  my.goal = NULL,
  my.goal.fun = NULL,
  quiet_mis = FALSE,
  na_prediction_action = "ignore"
)
```

Arguments

prediction_v	logical. A logical vector of predictions.
criterion_v	logical. A logical vector of (TRUE) criterion values.
correction	numeric. Correction added to all counts for calculating dprime. Default: correction = .25.
sens.w	numeric. Sensitivity weight parameter (from 0 to 1, for computing wacc). Default: sens.w = NULL (to ensure that values are passed by calling function).

`cost.outcomes` list. A list of length 4 with names 'hi', 'fa', 'mi', and 'cr' specifying the costs of a hit, false alarm, miss, and correct rejection, respectively. For instance, `cost.outcomes = listc("hi" = 0, "fa" = 10, "mi" = 20, "cr" = 0)` means that a false alarm and miss cost 10 and 20, respectively, while correct decisions have no cost. Default: `cost.outcomes = NULL` (to ensure that values are passed by calling function).

`cost_v` numeric. Additional cost value of each decision (as an optional vector of numeric values). Typically used to include the cue cost of each decision (as a constant for the current level of an FFT). Default: `cost_v = NULL` (to ensure that values are passed by calling function).

`my.goal` Name of an optional, user-defined goal (as character string). Default: `my.goal = NULL`.

`my.goal.fun` User-defined goal function (with 4 arguments `hi fa mi cr`). Default: `my.goal.fun = NULL`.

`quiet_mis` A logical value passed to hide/show `NA` user feedback (usually `x$params$quiet$mis` of the calling function). Default: `quiet_mis = FALSE` (i.e., show user feedback).

`na_prediction_action` What happens when no prediction is possible? (Experimental and currently unused.)

Details

The primary confusion matrix is computed by `confusionMatrix` of the **caret** package.

`comp_pred` *Fit and predict competing classification algorithms*

Description

`comp_pred` provides a wrapper for running (i.e., fit or predict) alternative classification algorithms to data (i.e., `data.train` or `data.test`, respectively).

Usage

```
comp_pred(
  formula,
  data.train,
  data.test = NULL,
  algorithm = NULL,
  model = NULL,
  sens.w = NULL,
  new.factors = "exclude",
  quiet_mis = FALSE
)
```

Arguments

formula	A formula (usually x$formula, for an FFTrees object x).
data.train	A training dataset (as a data frame).
data.test	A testing dataset (as a data frame).
algorithm	A character string specifying an algorithm in the set: • "lr": Logistic regression (using glm from **stats** with family = "binomial"); • "rlr": Regularized logistic regression (currently not supported); • "cart": Decision trees (using rpart from **rpart**); • "svm": Support vector machines (using svm from **e1071**); • "rf": Random forests (using randomForest from **randomForest**.
model	An optional existing model (as a model), to be applied to the test data.
sens.w	Sensitivity weight parameter (numeric, from 0 to 1), required to compute wacc.
new.factors	What should be done if new factor values are discovered in the test set (as a character string)? Available options: • "exclude": exclude case (i.e., remove these cases, used by default); • "base": predict the base rate of the criterion.
quiet_mis	A logical value passed to hide/show NA user feedback (usually x$params$quiet$mis of the calling function). Default: quiet_mis = FALSE (i.e., show user feedback).

Details

The range of competing algorithms currently available includes logistic regression (stats::glm), CART (rpart::rpart), support vector machines (e1071::svm), and random forests (randomForest::randomForest).

The current support for handling missing data (or NA values) is only rudimentary. When enabled (via the global options allow_NA_pred or allow_NA_crit), any rows in data.train or data.test with incomplete cases are being removed prior to fitting or predicting a model (by using na.omit from **stats**). See the specifications of each model for more sophisticated ways of handling missing data.

---

contraceptive *Contraceptive use data*

Description

A subset of the 1987 National Indonesia Contraceptive Prevalence Survey.

Usage

```
contraceptive
```

Format

A data frame containing 1473 cases (rows) and 10 variables (columns).

wife.age Wife's age, Numeric

wife.edu Wife's education, Nummeric, (1=low, 2, 3, 4=high)

hus.ed Husband's education, Nummeric, (1=low, 2, 3, 4=high)

children Number of children ever born, Numeric

wife.rel Wife's religion, Numeric, (0=Non-Islam, 1=Islam)

wife.work Wife's now working?, Nummeric, (0=Yes, 1=No)

hus.occ Husband's occupation, Nummeric, (1, 2, 3, 4)

sol Standard-of-living index, Nummeric, (1=low, 2, 3, 4=high)

media Media exposure, Numeric, (0=Good, 1=Not good)

cont.crit *Criterion*: Use of a contraceptive (as logical).
Values: `FALSE` vs. `TRUE` (42.7% vs. 57.3%).

Details

The samples describe married women who were either not pregnant or do not know if they were pregnant at the time of the interview.

The problem consists in predicting a woman's current contraceptive method choice (here: binarized `cont.crit`) based on her demographic and socio-economic characteristics.

We made the following enhancements to the original data for improved usability:

- The criterion was binarized from a class attribute variable with three levels (`1=No-use`, `2=Long-term`, `3=Short-term`) , into a logical variable with two levels (`TRUE` vs. `FALSE`).

Other than that, the data remains consistent with the original dataset.

Source

https://archive.ics.uci.edu/ml/datasets/Contraceptive+Method+Choice

Original creator and donor:

Tjen-Sien Lim

See Also

Other datasets: `blood`, `breastcancer`, `car`, `creditapproval`, `fertility`, `forestfires`, `heart.cost`, `heart.test`, `heart.train`, `heartdisease`, `iris.v`, `mushrooms`, `sonar`, `titanic`, `voting`, `wine`

`creditapproval` *Credit approval data*

Description

This data reports predictors and the result of credit card applications. Its attribute names and values have been changed to symbols to protect confidentiality.

Usage

```
creditapproval
```

Format

A data frame containing 690 cases (rows) and 15 variables (columns).

c.1 categorical: b, a

c.2 continuous

c.3 continuous

c.4 categorical: u, y, l, t

c.5 categorical: g, p, gg

c.6 categorical: c, d, cc, i, j, k, m, r, q, w, x, e, aa, ff

c.7 categorical: v, h, bb, j, n, z, dd, ff, o

c.8 continuous

c.9 categorical: t, f

c.10 categorical: t, f

c.11 continuous

c.12 categorical: t, f

c.13 categorical: g, p, s

c.14 continuous

c.15 continuous

crit *Criterion*: Credit approval.
Values: `TRUE` (+) vs. `FALSE` (-) (44.5% vs. 55.5%).

Details

This dataset contains a mix of attributes – continuous, nominal with small Ns, and nominal with larger Ns. There are also a few missing values.

We made the following enhancements to the original data for improved usability:

- Any missing values, denoted as "?" in the dataset, were transformed into NAs.
- Binary factor variables with exclusive "t" and "f" values were converted to logical TRUE/FALSE vectors.

Other than that, the data remains consistent with the original dataset.

Source

https://archive.ics.uci.edu/ml/datasets/Credit+Approval

See Also

Other datasets: blood, breastcancer, car, contraceptive, fertility, forestfires, heart.cost, heart.test, heart.train, heartdisease, iris.v, mushrooms, sonar, titanic, voting, wine

---

`describe_data` *Describe data*

Description

Calculate key descriptive statistics for a given set of data.

Usage

```
describe_data(data, data_name, criterion_name, baseline_value)
```

Arguments

`data`	A data frame with a criterion variable `criterion_name`.
`data_name`	A character string specifying a name for the data.
`criterion_name`	A character string specifying the criterion name.
`baseline_value`	The value in `criterion_name` denoting the baseline (e.g., `TRUE` or `FALSE`).

Value

A data frame with the descriptive statistics.

Examples

```
data(heartdisease)
describe_data(heartdisease, "heartdisease",
              criterion_name = "diagnosis",
              baseline_value = TRUE)
```

drop_nodes *Drop a node from an FFT definition*

Description

drop_nodes deletes one or more nodes from an existing FFT definition (by removing the corresponding rows from the FFT definition in the tidy data frame format).

When dropping the final node, the last remaining node becomes the new final node (i.e., gains a second exit).

Duplicates in nodes are dropped only once (rather than incrementally) and nodes not in the range 1:nrow(fft) are ignored. Dropping all nodes yields an error.

drop_nodes is the inverse function of select_nodes. Inserting new nodes is possible by add_nodes.

Usage

```
drop_nodes(fft, nodes = NA, quiet = FALSE)
```

Arguments

fft	One FFT definition (as a data frame in tidy format, with one row per node).
nodes	The FFT nodes to drop (as an integer vector). Default: nodes = NA.
quiet	Hide feedback messages (as logical)? Default: quiet = FALSE.

Value

One FFT definition (as a data frame in tidy format, with one row per node).

See Also

add_nodes for adding nodes to an FFT definition; edit_nodes for editing nodes in an FFT definition; select_nodes for selecting nodes in an FFT definition; get_fft_df for getting the FFT definitions of an FFTrees object; read_fft_df for reading one FFT definition from tree definitions; add_fft_df for adding FFTs to tree definitions; FFTrees for creating FFTs from and applying them to data.

Other tree definition and manipulation functions: add_fft_df(), add_nodes(), edit_nodes(), flip_exits(), get_fft_df(), read_fft_df(), reorder_nodes(), select_nodes(), write_fft_df()

edit_nodes *Edit nodes in an FFT definition*

Description

edit_nodes allows manipulating one or more nodes from an existing FFT definition (in the tidy data frame format).

edit_nodes allows to directly set and change the value(s) of class, cue, direction, threshold, and exit, in an FFT definition for the specified nodes.

There is only rudimentary verification for plausible entries. Importantly, however, as edit_nodes is ignorant of data, the values of its variables are not validated for a specific set of data.

Repeated changes of a node are possible (by repeating the corresponding integer value in nodes).

Usage

```
edit_nodes(
  fft,
  nodes = NA,
  class = NA,
  cue = NA,
  direction = NA,
  threshold = NA,
  exit = NA,
  quiet = FALSE
)
```

Arguments

fft	One FFT definition (as a data frame in tidy format, with one row per node).
nodes	The FFT nodes to be edited (as an integer vector). Default: nodes = NA.
class	The class values of nodes (as character).
cue	The cue names of nodes (as character).
direction	The direction values of nodes (as character).
threshold	The threshold values of nodes (as character).
exit	The exit values of nodes (as values from exit_types).
quiet	Hide feedback messages (as logical)? Default: quiet = FALSE.

Value

One FFT definition (as a data frame in tidy format, with one row per node).

See Also

add_nodes for adding nodes to an FFT definition; drop_nodes for deleting nodes from an FFT definition; flip_exits for reversing exits in an FFT definition; reorder_nodes for reordering nodes of an FFT definition; select_nodes for selecting nodes in an FFT definition; get_fft_df for getting the FFT definitions of an FFTrees object; read_fft_df for reading one FFT definition from tree definitions; add_fft_df for adding FFTs to tree definitions; FFTrees for creating FFTs from and applying them to data.

Other tree definition and manipulation functions: add_fft_df(), add_nodes(), drop_nodes(), flip_exits(), get_fft_df(), read_fft_df(), reorder_nodes(), select_nodes(), write_fft_df()

---

fact_clean *Clean factor variables in prediction data*

Description

Clean factor variables in prediction data

Usage

```
fact_clean(data.train, data.test, show.warning = T)
```

Arguments

data.train	A training dataset
data.test	A testing dataset
show.warning	logical

---

fertility *Fertility data*

Description

This dataset describes a sample of 100 volunteers providing a semen sample that was analyzed according to the WHO 2010 criteria.

Usage

```
fertility
```

Format

A data frame containing 100 rows and 10 columns.

season Season in which the analysis was performed. (winter, spring, summer, fall)

age Age at the time of analysis

child.dis Childish diseases (ie , chicken pox, measles, mumps, polio) (yes(1), no(0))

trauma Accident or serious trauma (yes(1), no(0))

surgery Surgical intervention (yes(1), no(0))

fevers High fevers in the last year (less than three months ago(-1), more than three months ago (0), no. (1))

alcohol Frequency of alcohol consumption (several times a day, every day, several times a week, once a week, hardly ever or never)

smoking Smoking habit (never(-1), occasional (0)) daily (1))

sitting Number of hours spent sitting per day

diagnosis *Criterion*: Diagnosis normal (TRUE) vs. altered (FALSE) (88.0% vs.\ 22.0%).

Details

Sperm concentration are related to socio-demographic data, environmental factors, health status, and life habits.

We made the following enhancements to the original data for improved usability:

- The criterion was redefined from a factor variable with two levels (`N=Normal`, `O=Altered`) into a logical variable (`TRUE` vs. `FALSE`).

Other than that, the data remains consistent with the original dataset.

Source

https://archive.ics.uci.edu/ml/datasets/Fertility

Original contributors:

David Gil Lucentia Research Group Department of Computer Technology University of Alicante

Jose Luis Girela Department of Biotechnology University of Alicante

See Also

Other datasets: `blood`, `breastcancer`, `car`, `contraceptive`, `creditapproval`, `forestfires`, `heart.cost`, `heart.test`, `heart.train`, `heartdisease`, `iris.v`, `mushrooms`, `sonar`, `titanic`, `voting`, `wine`

FFTrees *Main function to create and apply fast-and-frugal trees (FFTs)*

Description

FFTrees is the workhorse function of the **FFTrees** package for creating fast-and-frugal trees (FFTs).

FFTs are decision algorithms for solving binary classification tasks, i.e., they predict the values of a binary criterion variable based on 1 or multiple predictor variables (cues).

Using FFTrees on data usually generates a range of FFTs and corresponding summary statistics (as an FFTrees object) that can then be printed, plotted, and examined further.

The criterion and predictor variables are specified in formula notation. Based on the settings of data and data.test, FFTs are trained on a (required) training dataset (given the set of current goal values) and evaluated on (or predict) an (optional) test dataset.

If an existing FFTrees object object or tree.definitions are provided as inputs, no new FFTs are created. When both arguments are provided, tree.definitions take priority over the FFTs in an existing object. Specifically,

- If tree.definitions are provided, these are assigned to the FFTs of x.
- If no tree.definitions are provided, but an existing FFTrees object object is provided, the trees from object are assigned to the FFTs of x.

Create and evaluate fast-and-frugal trees (FFTs).

Usage

```
FFTrees(
  formula = NULL,
  data = NULL,
  data.test = NULL,
  algorithm = "ifan",
  train.p = 1,
  goal = NULL,
  goal.chase = NULL,
  goal.threshold = NULL,
  max.levels = NULL,
  numthresh.method = "o",
  numthresh.n = 10,
  repeat.cues = TRUE,
  stopping.rule = "exemplars",
  stopping.par = 0.1,
  sens.w = 0.5,
  cost.outcomes = NULL,
  cost.cues = NULL,
  main = NULL,
  decision.labels = c("False", "True"),
```

```
  my.goal = NULL,
  my.goal.fun = NULL,
  my.tree = NULL,
  object = NULL,
  tree.definitions = NULL,
  do.comp = TRUE,
  do.cart = TRUE,
  do.lr = TRUE,
  do.rf = TRUE,
  do.svm = TRUE,
  quiet = list(ini = TRUE, fin = FALSE, mis = FALSE, set = TRUE),
  comp = NULL,
  force = NULL,
  rank.method = NULL,
  rounding = NULL,
  store.data = NULL,
  verbose = NULL
)
```

Arguments

`formula`	A formula. A `formula` specifying a binary criterion variable (as logical) as a function of 1 or more predictor variables (cues).
`data`	A data frame. A dataset used for training (fitting) FFTs and alternative algorithms. `data` must contain the binary criterion variable specified in `formula` and potential predictors (which can be categorical or numeric variables).
`data.test`	A data frame. An optional dataset used for model testing (prediction) with the same structure as data.
`algorithm`	A character string. The algorithm used to create FFTs. Can be `'ifan'`, `'dfan'`.
`train.p`	numeric. What percentage of the data to use for training when `data.test` is not specified? For example, `train.p = .50` will randomly split `data` into a 50% training set and a 50% test set. Default: `train.p = 1` (i.e., using *all* data for training).
`goal`	A character string indicating the statistic to maximize when *selecting trees*: `"acc"` = overall accuracy, `"bacc"` = balanced accuracy, `"wacc"` = weighted accuracy, `"dprime"` = discriminability, `"cost"` = costs (based on `cost.outcomes` and `cost.cues`).
`goal.chase`	A character string indicating the statistic to maximize when *constructing trees*: `"acc"` = overall accuracy, `"bacc"` = balanced accuracy, `"wacc"` = weighted accuracy, `"dprime"` = discriminability, `"cost"` = costs (based on `cost.outcomes` and `cost.cues`).
`goal.threshold`	A character string indicating the criterion to maximize when *optimizing cue thresholds*: `"acc"` = overall accuracy, `"bacc"` = balanced accuracy, `"wacc"` = weighted accuracy, `"dprime"` = discriminability, `"cost"` = costs (based only on `cost.outcomes`, as `cost.cues` are constant per cue). All default goals are set in `fftrees_create`.

`max.levels` integer. The maximum number of nodes (or levels) considered for an FFT. As all combinations of possible exit structures are considered, larger values of `max.levels` will create larger sets of FFTs.

`numthresh.method`
How should thresholds for numeric cues be determined (as character)? `"o"` will optimize thresholds (for `goal.threshold`), while `"m"` will use the median. Default: `numthresh.method = "o"`.

`numthresh.n` The number of numeric thresholds to try (as integer). Default: `numthresh.n = 10`.

`repeat.cues` May cues occur multiple times within a tree (as logical)? Default: `repeat.cues = TRUE`.

`stopping.rule` A character string indicating the method to stop growing trees. Available options are:

- `"exemplars"`: A tree grows until only a small proportion of unclassified exemplars remain;
- `"levels"`: A tree grows until a certain level is reached;
- `"statdelta"`: A tree grows until the change in the criterion statistic `goal.chase` exceeds some threshold level. (This setting is currently experimental and includes the first level beyond threshold. As tree statistics can be non-monotonic, this option may yield inconsistent results.)

All stopping methods use `stopping.par` to set a numeric threshold value. Default: `stopping.rule = "exemplars"`.

`stopping.par` numeric. A numeric parameter indicating the criterion value for the current `stopping.rule`. For stopping.rule `"levels"`, this is the number of desired levels (as an integer). For stopping rule `"exemplars"`, this is the smallest proportion of exemplars allowed in the last level. For stopping.rule `"statdelta"`, this is the minimum required change (in the `goal.chase` value) to include a level. Default: `stopping.par = .10`.

`sens.w` A numeric value from `0` to `1` indicating how to weight sensitivity relative to specificity when optimizing *weighted* accuracy (e.g., `goal = 'wacc'`). Default: `sens.w = .50` (i.e., `wacc` corresponds to `bacc`).

`cost.outcomes` A list of length 4 specifying the cost value for one of the 4 possible classification outcomes. The list elements must be named `'hi'`, `'fa'`, `'mi'`, and `'cr'` (for specifying the costs of a hit, false alarm, miss, and correct rejection, respectively) and provide a numeric cost value. E.g.; `cost.outcomes = listc("hi" = 0, "fa" = 10, "mi" = 20, "cr" = 0)` imposes false alarm and miss costs of `10` and `20` units, respectively, while correct decisions have no costs.

`cost.cues` A list containing the cost of each cue (in some common unit). Each list element must have a name corresponding to a cue (i.e., a variable in `data`), and should be a single (positive numeric) value. Cues in `data` that are not present in `cost.cues` are assumed to have no costs (i.e., a cost value of `0`).

`main` string. An optional label for the dataset. Passed on to other functions, like `plot.FFTrees`, and `print.FFTrees`.

`decision.labels`
: A vector of strings of length 2 for the text labels for negative and positive decision/prediction outcomes (i.e., left vs. right, noise vs. signal, 0 vs. 1, respectively, as character). E.g.; `decision.labels = c("Healthy", "Diseased")`.

`my.goal`
: The name of an optimization measure defined by `my.goal.fun` (as a character string). Example: `my.goal = "my_acc"` (see `my.goal.fun` for corresponding function). Default: `my.goal = NULL`.

`my.goal.fun`
: The definition of an outcome measure to optimize, defined as a function of the frequency counts of the 4 basic classification outcomes `hi, fa, mi, cr` (i.e., an R function with 4 arguments `hi, fa, mi, cr`). Example: `my.goal.fun = function(hi, fa, mi, cr){(hi + cr)/(hi + fa + mi + cr)}` (i.e., accuracy). Default: `my.goal.fun = NULL`.

`my.tree`
: A verbal description of an FFT, i.e., an "FFT in words" (as character string). For example, `my.tree = "If age > 20, predict TRUE. If sex = {m}, predict FALSE. Otherwise, predict TRUE."`.

`object`
: An optional existing `FFTrees` object. When specified, no new FFTs are fitted, but existing trees are applied to `data` and `data.test`. When `formula`, `data` or `data.test` are not specified, the current values of `object` are used.

`tree.definitions`
: An optional `data.frame` of hard-coded FFT definitions (in the format of `x$trees$definitions` of an `FFTrees` object `x`). If specified, no new FFTs are being fitted (i.e., `algorithm` and functions for evaluating cues and creating FFTs are skipped). Instead, the tree definitions provided are used to re-evaluate the current `FFTrees` object on current data.

`do.comp`, `do.lr`, `do.cart`, `do.svm`, `do.rf`
: Should alternative algorithms be used for comparison (as logical)? All options are set to `TRUE` by default. Available options correspond to:

  - `do.lr`: Logistic regression (LR, using `glm` from **stats** with `family = "binomial"`);
  - `do.cart`: Classification and regression trees (CART, using `rpart` from **rpart**);
  - `do.svm`: Support vector machines (SVM, using `svm` from **e1071**);
  - `do.rf`: Random forests (RF, using `randomForest` from **randomForest**.

  Specifying `do.comp = FALSE` sets all available options to `FALSE`.

`quiet`
: A list of 4 logical arguments: Should detailed progress reports be suppressed? Setting list elements to `FALSE` is helpful when diagnosing errors. Default: `quiet = list(ini = TRUE, fin = FALSE, mis = FALSE, set = TRUE)`, for initial vs. final steps, missing cases, and parameter settings, respectively. Providing a single logical value sets all elements to `TRUE` or `FALSE`.

`comp`, `force`, `rank.method`, `rounding`, `store.data`, `verbose`
: Deprecated arguments (unused or replaced, to be retired in future releases).

Value

An `FFTrees` object with the following elements:

criterion_name The name of the binary criterion variable (as character).

cue_names The names of all potential predictor variables (cues) in the data (as character).

formula The formula specified when creating the FFTs.

trees A list of FFTs created, with further details contained in n, best, definitions, inwords, stats, level_stats, and decisions.

data The original training and test data (if available).

params A list of defined control parameters (e.g.; algorithm, goal, sens.w, as well as various thresholds, stopping rule, and cost parameters).

competition Models and classification statistics for competitive classification algorithms: Logistic regression (lr), classification and regression trees (cart), random forests (rf), and support vector machines (svm).

cues A list of cue information, with further details contained in thresholds and stats.

See Also

print.FFTrees for printing FFTs; plot.FFTrees for plotting FFTs; summary.FFTrees for summarizing FFTs; inwords for obtaining a verbal description of FFTs; showcues for plotting cue accuracies.

Examples

```
# 1. Create fast-and-frugal trees (FFTs) for heart disease:
heart.fft <- FFTrees(formula = diagnosis ~ .,
                     data = heart.train,
                     data.test = heart.test,
                     main = "Heart Disease",
                     decision.labels = c("Healthy", "Diseased")
                     )

# 2. Print a summary of the result:
heart.fft  # same as:
# print(heart.fft, data = "train", tree = "best.train")

# 3. Plot an FFT applied to training data:
plot(heart.fft)  # same as:
# plot(heart.fft, what = "all", data = "train", tree = "best.train")

# 4. Apply FFT to (new) testing data:
plot(heart.fft, data = "test")            # predict for Tree 1
plot(heart.fft, data = "test", tree = 2)  # predict for Tree 2

# 5. Predict classes and probabilities for new data:
predict(heart.fft, newdata = heartdisease)
predict(heart.fft, newdata = heartdisease, type = "prob")

# 6. Create a custom tree (from verbal description) with my.tree:
custom.fft <- FFTrees(
  formula = diagnosis ~ .,
  data = heartdisease,
  my.tree = "If age < 50, predict False.
             If sex = 1, predict True.
```

```
            If chol > 300, predict True, otherwise predict False.",
  main = "My custom FFT")

# Plot the (pretty bad) custom tree:
plot(custom.fft)
```

`FFTrees.guide` *Open the* **FFTrees** *package guide*

Description

Open the **FFTrees** package guide

Usage

```
FFTrees.guide()
```

Value

No return value, called for side effects.

`fftrees_cuerank` *Calculate thresholds that optimize some statistic (goal) for cues in data*

Description

`fftrees_cuerank` takes an `FFTrees` object `x` and optimizes its `goal.threshold` (from `x$params`) for all cues in `newdata` (of type `data`).

Usage

```
fftrees_cuerank(x = NULL, newdata = NULL, data = "train", rounding = NULL)
```

Arguments

`x`	An `FFTrees` object.
`newdata`	A dataset with cues to be ranked (as data frame).
`data`	The type of data with cues to be ranked (as character: `'train'`, `'test'`, or `'dynamic'`). Default: `data = 'train'`.
`rounding`	integer. An integer value indicating the decimal digit to which non-integer numeric cue thresholds are to be rounded. Default: `rounding = NULL` (i.e., no rounding).

Details

fftrees_cuerank creates a data frame cuerank_df that is added to x$cues$stats.

Note that the cue directions and thresholds computed by **FFTrees** always predict positive criterion values (i.e., TRUE or signal, rather than FALSE or noise). Using these thresholds for negative exits (i.e., for predicting instances of FALSE or noise) usually requires a reversal (e.g., negating cue direction).

fftrees_cuerank is called (twice) by the fftrees_grow_fan algorithm to grow fast-and-frugal trees (FFTs).

Value

A modified FFTrees object (with cue rank information for the current data type in x$cues$stats).

---

fftrees_ffttowords *Describe a fast-and-frugal tree (FFT) in words*

Description

fftrees_ffttowords provides a verbal description of tree definition (as defined in an FFTrees object). Thus, fftrees_ffttowords translates an abstract FFT definition into natural language output.

fftrees_ffttowords is the complement function to fftrees_wordstofftrees, which parses a verbal description of an FFT into the abstract tree definition of an FFTrees object.

The final sentence (or tree node) of the FFT's description always predicts positive criterion values (i.e., TRUE instances) first, before predicting negative criterion values (i.e., FALSE instances). Note that this may require a reversal of exit directions, if the final cue predicted FALSE instances.

Note that the cue directions and thresholds computed by **FFTrees** always predict positive criterion values (i.e., TRUE or signal, rather than FALSE or noise). Using these thresholds for negative exits (i.e., for predicting instances of FALSE or noise) usually requires a reversal (e.g., negating cue direction).

Usage

```
fftrees_ffttowords(x = NULL, mydata = "train", digits = 2)
```

Arguments

x	An FFTrees object created with FFTrees.
mydata	The type of data to which a tree is being applied (as character string "train" or "test"). Default: mydata = "train".
digits	How many digits to round numeric values (as integer)?

Value

A modified FFTrees object x with x$trees$inwords containing a list of string vectors.

See Also

fftrees_wordstofftrees for converting a verbal description of an FFT into an FFTrees object; fftrees_create for creating FFTrees objects; fftrees_grow_fan for creating FFTs by applying algorithms to data; print.FFTrees for printing FFTs; plot.FFTrees for plotting FFTs; summary.FFTrees for summarizing FFTs; FFTrees for creating FFTs from and applying them to data.

Examples

```
heart.fft <- FFTrees(diagnosis ~ .,
  data = heartdisease,
  decision.labels = c("Healthy", "Disease")
)

inwords(heart.fft)
```

---

fftrees_fitcomp *Fit competitive algorithms*

Description

fftrees_fitcomp fits competitive algorithms for binary classification tasks (e.g., LR, CART, RF, SVM) to the data and parameters specified in an FFTrees object.

fftrees_fitcomp is called by the main FFTrees function when creating FFTs from and applying them to data (unless do.comp = FALSE).

Usage

```
fftrees_fitcomp(x)
```

Arguments

x	An FFTrees object.

See Also

FFTrees for creating FFTs from and applying them to data.

fftrees_grow_fan *Grow fast-and-frugal trees (FFTs) using the* fan *algorithms*

Description

fftrees_grow_fan is called by fftrees_define to create new FFTs by applying the fan algorithms (specifically, either ifan or dfan) to data.

Usage

```
fftrees_grow_fan(x, repeat.cues = TRUE)
```

Arguments

x	An FFTrees object.
repeat.cues	Can cues be considered/used repeatedly (as logical)? Default: repeat.cues = TRUE, but only relevant when using the dfan algorithm.

See Also

fftrees_create for creating FFTrees objects; fftrees_define for defining FFTs; fftrees_grow_fan for creating FFTs by applying algorithms to data; fftrees_wordstofftrees for creating FFTs from verbal descriptions; FFTrees for creating FFTs from and applying them to data.

fftrees_ranktrees *Rank FFTs by current goal*

Description

fftrees_ranktrees ranks trees in an FFTrees object x based on the current goal (either "cost" or as specified in x$params$goal).

fftrees_ranktrees is called by the main FFTrees function when creating FFTs from and applying them to (training) data.

Usage

```
fftrees_ranktrees(x, data = "train")
```

Arguments

x	An FFTrees object.
data	The type of data to be used (as character). Default: data = "train".

See Also

FFTrees for creating FFTs from and applying them to data.

`fftrees_threshold_factor_grid` *Perform a grid search over factor and return accuracy statistics for a given factor cue*

Description

Perform a grid search over factor and return accuracy statistics for a given factor cue

Usage

```
fftrees_threshold_factor_grid(
  thresholds = NULL,
  cue_v = NULL,
  criterion_v = NULL,
  directions = "=",
  goal.threshold = NULL,
  sens.w = NULL,
  my.goal = NULL,
  my.goal.fun = NULL,
  cost.each = NULL,
  cost.outcomes = NULL
)
```

Arguments

`thresholds`	numeric. A vector of factor thresholds to consider.
`cue_v`	numeric. Feature/cue values.
`criterion_v`	logical. A logical vector of (TRUE) criterion values.
`directions`	character. Character vector of threshold directions to consider.
`goal.threshold`	A character string indicating the criterion to maximize when *optimizing cue thresholds*: `"acc"` = overall accuracy, `"bacc"` = balanced accuracy, `"wacc"` = weighted accuracy, `"dprime"` = discriminability, `"cost"` = costs (based only on `cost.outcomes`, as `cost.cues` are constant per cue). Default: `goal.threshold = "bacc"`.
`sens.w`	numeric. Sensitivity weight parameter (from `0` to `1`, for computing `wacc`). Default: `sens.w = .50`.
`my.goal`	Name of an optional, user-defined goal (as character string). Default: `my.goal = NULL`.
`my.goal.fun`	User-defined goal function (with 4 arguments `hi fa mi cr`). Default: `my.goal.fun = NULL`.
`cost.each`	numeric. A constant cost value to add to each value (e.g., the cost of the cue).

`cost.outcomes` list. A list of length 4 with names 'hi', 'fa', 'mi', and 'cr' specifying the costs of a hit, false alarm, miss, and correct rejection, respectively, in some common currency. For instance, `cost.outcomes = listc("hi" = 0, "fa" = 10, "mi" = 20, "cr" = 0)` means that a false alarm and miss cost `10` and `20` units, respectively, while correct decisions have no cost.

Value

A data frame containing accuracy statistics for factor thresholds.

See Also

`fftrees_threshold_numeric_grid` for numeric cues.

`fftrees_threshold_numeric_grid` *Perform a grid search over thresholds and return accuracy statistics for a given numeric cue*

Description

Perform a grid search over thresholds and return accuracy statistics for a given numeric cue

Usage

```
fftrees_threshold_numeric_grid(
  thresholds,
  cue_v,
  criterion_v,
  directions = c(">", "<="),
  goal.threshold = NULL,
  sens.w = NULL,
  my.goal = NULL,
  my.goal.fun = NULL,
  cost.each = NULL,
  cost.outcomes = NULL
)
```

Arguments

`thresholds`	numeric. A vector of thresholds to consider.
`cue_v`	numeric. Feature values.
`criterion_v`	logical. A logical vector of (TRUE) criterion values.
`directions`	character. Possible directions to consider.

goal.threshold	A character string indicating the criterion to maximize when *optimizing cue thresholds*: "acc" = overall accuracy, "bacc" = balanced accuracy, "wacc" = weighted accuracy, "dprime" = discriminability, "cost" = costs (based only on cost.outcomes, as cost.cues are constant per cue). Default: goal.threshold = "bacc".
sens.w	numeric. Sensitivity weight parameter (from 0 to 1, for computing wacc). Default: sens.w = .50.
my.goal	Name of an optional, user-defined goal (as character string). Default: my.goal = NULL.
my.goal.fun	User-defined goal function (with 4 arguments hi fa mi cr). Default: my.goal.fun = NULL.
cost.each	numeric. A constant cost value to add to each value (e.g., the cost of the cue).
cost.outcomes	list. A list of length 4 with names 'hi', 'fa', 'mi', and 'cr' specifying the costs of a hit, false alarm, miss, and correct rejection, respectively, in some common currency. For instance, cost.outcomes = listc("hi" = 0, "fa" = 10, "mi" = 20, "cr" = 0) means that a false alarm and miss cost 10 and 20 units, respectively, while correct decisions have no cost.

Value

A data frame containing accuracy statistics for numeric thresholds.

See Also

fftrees_threshold_factor_grid for factor cues.

---

fftrees_wordstofftrees

Convert a verbal description of an FFT into an FFTrees *object*

Description

fftrees_wordstofftrees converts a verbal description of an FFT (provided as a string of text) into a tree definition (of an FFTrees object). Thus, fftrees_wordstofftrees provides a simple natural language parser for FFTs.

fftrees_wordstofftrees is the complement function to fftrees_ffttowords, which converts an abstract tree definition (of an FFTrees object) into a verbal description (i.e., provides natural language output).

To increase robustness, the parsing of fftrees_wordstofftrees allows for lower- or uppercase spellings (but not typographical variants) and ignores the else-part of the final sentence (i.e., the part beginning with "otherwise").

Usage

```
fftrees_wordstofftrees(x, my.tree)
```

Arguments

x	An FFTrees object.
my.tree	A character string. A verbal description (as a string of text) defining an FFT.

Value

An FFTrees object with a new tree definition as described by my.tree.

See Also

fftrees_ffttowords for converting FFTs into verbal descriptions; print.FFTrees for printing FFTs; plot.FFTrees for plotting FFTs; summary.FFTrees for summarizing FFTs; FFTrees for creating FFTs from and applying them to data.

---

flip_exits *Flip exits in an FFT definition*

Description

flip_exits reverses the exits of one or more nodes from an existing FFT definition (in the tidy data frame format).

flip_exits alters the value(s) of the non-final exits specified in nodes (from 0 to 1, or from 1 to 0). By contrast, exits of final nodes remain unchanged.

Duplicates in nodes are flipped only once (rather than repeatedly) and nodes not in the range 1:nrow(fft) are ignored.

flip_exits is a more specialized function than edit_nodes.

Usage

```
flip_exits(fft, nodes = NA, quiet = FALSE)
```

Arguments

fft	One FFT definition (as a data frame in tidy format, with one row per node).
nodes	The FFT nodes whose exits are to be flipped (as an integer vector). Default: nodes = NA.
quiet	Hide feedback messages (as logical)? Default: quiet = FALSE.

Value

One FFT definition (as a data frame in tidy format, with one row per node).

See Also

add_nodes for adding nodes to an FFT definition; edit_nodes for editing nodes in an FFT definition; drop_nodes for deleting nodes from an FFT definition; reorder_nodes for reordering nodes of an FFT definition; select_nodes for selecting nodes in an FFT definition; get_fft_df for getting the FFT definitions of an FFTrees object; read_fft_df for reading one FFT definition from tree definitions; add_fft_df for adding FFTs to tree definitions; FFTrees for creating FFTs from and applying them to data.

Other tree definition and manipulation functions: add_fft_df(), add_nodes(), drop_nodes(), edit_nodes(), get_fft_df(), read_fft_df(), reorder_nodes(), select_nodes(), write_fft_df()

---

forestfires *Forest fires data*

Description

A dataset of forest fire statistics.

Usage

```
forestfires
```

Format

A data frame containing 517 rows and 13 columns.

X Integer -x-axis spatial coordinate within the Montesinho park map: 1 to 9

Y Integer - y-axis spatial coordinate within the Montesinho park map: 2 to 9

month Factor - month of the year: "jan" to "dec"

day Factor -day of the week: "mon" to "sun"

FFMC Numeric -FFMC index from the FWI system: 18.7 to 96.20

DMC Numeric - DMC index from the FWI system: 1.1 to 291.3

DC Numeric - DC index from the FWI system: 7.9 to 860.6

ISI Numeric - ISI index from the FWI system: 0.0 to 56.10

temp Numeric - temperature in Celsius degrees: 2.2 to 33.30

RH Numeric - relative humidity in percent: 15.0 to 100

wind Numeric - wind speed in km/h: 0.40 to 9.40

rain Numeric - outside rain in mm/m2 : 0.0 to 6.4

fire.crit *Criterion*: Was there a fire (greater than 1.00 ha)?
Values: TRUE (yes) vs. FALSE (no) (47.0% vs. 53.0%).

Details

We made the following enhancements to the original data for improved usability:

- The criterion was redefined from a numeric variable that indicated the number of hectares that burned in a fire into a logical variable (TRUE (for values >1) vs. FALSE (for values <=1)).

Other than that, the data remains consistent with the original dataset.

Source

http://archive.ics.uci.edu/ml/datasets/Forest+Fires

Original creator: Prof. Paulo Cortez and Aníbal Morais Department of Information Systems University of Minho, Portugal

See Also

Other datasets: blood, breastcancer, car, contraceptive, creditapproval, fertility, heart.cost, heart.test, heart.train, heartdisease, iris.v, mushrooms, sonar, titanic, voting, wine

---

get_best_tree *Select the best tree (from current set of FFTs)*

Description

get_best_tree selects (looks up and identifies) the best tree (as an integer) from the set (or "fan") of FFTs contained in the current FFTrees object x, an existing type of data ('train' or 'test'), and a goal for which corresponding statistics are available in the designated data type (in x$trees$stats).

Usage

```
get_best_tree(x, data, goal, my.goal.max = TRUE)
```

Arguments

x	An FFTrees object.
data	The type of data to consider (as character: either 'train' or 'test').
goal	A goal (as character) to be maximized or minimized when selecting a tree from an existing FFTrees object x (with existing x$trees$stats).
my.goal.max	Default direction for user-defined my.goal (as logical): Should my.goal be maximized? Default: my.goal.max = TRUE.

Details

Importantly, `get_best_tree` only identifies and selects the 'tree' *identifier* (as an integer) from the set of *existing* trees with known statistics, rather than creating new trees or computing new cue thresholds. More specifically, `goal` is used for identifying and selecting the 'tree' identifier (as an integer) of the best FFT from an existing set of FFTs, but not for computing new cue thresholds (see `goal.threshold` and `fftrees_cuerank()`) or creating new trees (see `goal.chase` and `fftrees_ranktrees()`).

Value

An integer denoting the `tree` that maximizes/minimizes `goal` in `data`.

See Also

`FFTrees` for creating FFTs from and applying them to data.

Other utility functions: `get_exit_type()`, `get_fft_df()`

`get_exit_type` *Get exit type (from a vector* `x` *of FFT exit descriptions)*

Description

`get_exit_type` checks and converts a vector `x` of FFT exit descriptions into exits of an FFT that correspond to the current options of `exit_types` (as a global constant).

Usage

```
get_exit_type(x, verify = TRUE)
```

Arguments

`x`	A vector of FFT exit descriptions.
`verify`	A flag to turn verification on/off (as logical). Default: `verify = TRUE`.

Details

`get_exit_type` also verifies that the exit types conform to an FFT (e.g., only the exits of the final node are bi-directional).

Value

A vector of `exit_types` (or an error).

See Also

`FFTrees` for creating FFTs from and applying them to data.

Other utility functions: `get_best_tree()`, `get_fft_df()`

Examples

```
get_exit_type(c(0, 1, .5))
get_exit_type(c(FALSE,   " True ",  2/4))
get_exit_type(c("noise", "signal", "final"))
get_exit_type(c("left",  "right",  "both"))
```

`get_fft_df` *Get FFT definitions (from an* `FFTrees` *object* `x`*)*

Description

`get_fft_df` gets the FFT definitions of an `FFTrees` object `x` (as a `data.frame`).

Usage

```
get_fft_df(x)
```

Arguments

`x`	An `FFTrees` object.

Details

The FFTs in the `data.frame` returned are represented in the one-line per FFT definition format used by an `FFTrees` object.

In addition to looking up `x$trees$definitions`, `get_fft_df` verifies that the FFT definitions are valid (given current settings).

Value

A set of FFT definitions (as a `data.frame`/`tibble`, in the one-line per FFT definition format used by an `FFTrees` object).

See Also

`read_fft_df` for reading one FFT definition from tree definitions; `write_fft_df` for writing one FFT to tree definitions; `add_fft_df` for adding FFTs to tree definitions; `FFTrees` for creating FFTs from and applying them to data.

Other utility functions: `get_best_tree()`, `get_exit_type()`

Other tree definition and manipulation functions: `add_fft_df()`, `add_nodes()`, `drop_nodes()`, `edit_nodes()`, `flip_exits()`, `read_fft_df()`, `reorder_nodes()`, `select_nodes()`, `write_fft_df()`

heart.cost *Cue costs for the heartdisease data*

Description

This data further characterizes the variables (cues) in the heartdisease dataset.

Usage

```
heart.cost
```

Format

A list of length 13 containing the cost of each cue in the heartdisease dataset (in dollars). Each list element is a single (positive numeric) value.

Source

https://archive.ics.uci.edu/ml/machine-learning-databases/heart-disease/costs/

See Also

heartdisease dataset.

Other datasets: blood, breastcancer, car, contraceptive, creditapproval, fertility, forestfires, heart.test, heart.train, heartdisease, iris.v, mushrooms, sonar, titanic, voting, wine

heart.test *Heart disease testing data*

Description

Testing data for a heartdisease data. This subset is used to test the prediction performance of a model trained on the heart.train data. The dataset heartdisease contains both datasets.

Usage

```
heart.test
```

Format

A data frame containing 153 rows and 14 columns (see heartdisease for details).

Source

https://archive.ics.uci.edu/ml/datasets/Heart+Disease

See Also

heartdisease dataset.

Other datasets: blood, breastcancer, car, contraceptive, creditapproval, fertility, forestfires, heart.cost, heart.train, heartdisease, iris.v, mushrooms, sonar, titanic, voting, wine

---

heart.train *Heart disease training data*

Description

Training data for a binary prediction model (here: FFT) on (a subset of) the heartdisease data. The complementary subset for model testing is heart.test. The data in heartdisease contains both subsets.

Usage

```
heart.train
```

Format

A data frame containing 150 rows and 14 columns (see heartdisease for details).

Source

https://archive.ics.uci.edu/ml/datasets/Heart+Disease

See Also

heartdisease dataset.

Other datasets: blood, breastcancer, car, contraceptive, creditapproval, fertility, forestfires, heart.cost, heart.test, heartdisease, iris.v, mushrooms, sonar, titanic, voting, wine

---

heartdisease *Heart disease data*

Description

A dataset predicting the diagnosis of 303 patients tested for heart disease.

Usage

```
heartdisease
```

Format

A data frame containing 303 rows and 14 columns, with the following variables:

diagnosis True value of binary criterion: TRUE = Heart disease, FALSE = No Heart disease

age Age (in years)

sex Sex, 1 = male, 0 = female

cp Chest pain type: ta = typical angina, aa = atypical angina, np = non-anginal pain, a = asymptomatic

trestbps Resting blood pressure (in mm Hg on admission to the hospital)

chol Serum cholestoral in mg/dl

fbs Fasting blood sugar > 120 mg/dl: 1 = true, 0 = false

restecg Resting electrocardiographic results. "normal" = normal, "abnormal" = having ST-T wave abnormality (T wave inversions and/or ST elevation or depression of > 0.05 mV), "hypertrophy" = showing probable or definite left ventricular hypertrophy by Estes' criteria.

thalach Maximum heart rate achieved

exang Exercise induced angina: 1 = yes, 0 = no

oldpeak ST depression induced by exercise relative to rest

slope The slope of the peak exercise ST segment.

ca Number of major vessels (0-3) colored by flourosopy

thal "normal" = normal, "fd" = fixed defect, "rd" = reversible defect

Source

https://archive.ics.uci.edu/ml/datasets/Heart+Disease

See Also

heart.cost dataset for cost information.

Other datasets: blood, breastcancer, car, contraceptive, creditapproval, fertility, forestfires, heart.cost, heart.test, heart.train, iris.v, mushrooms, sonar, titanic, voting, wine

---

inwords *Provide a verbal description of an FFT*

Description

inwords generates and provides a verbal description of a fast-and-frugal tree (FFT) from an FFTrees object.

When data remains unspecified, inwords will only look up x$trees$inwords. When data is set to either "train" or "test", inwords first employs fftrees_ffttowords to re-generate the verbal descriptions of FFTs in x.

Usage

```
inwords(x, data = NULL, tree = 1)
```

Arguments

x	An FFTrees object.
data	The type of data to which a tree is being applied (as character string "train" or "test"). Default: data = NULL will only look up x$trees$inwords.
tree	The tree to display (as an integer).

Value

A verbal description of an FFT (as a character string).

See Also

fftrees_ffttowords for converting FFTs into verbal descriptions; print.FFTrees for printing FFTs; plot.FFTrees for plotting FFTs; summary.FFTrees for summarizing FFTs; FFTrees for creating FFTs from and applying them to data.

---

iris.v *Iris data*

Description

A famous dataset from R.A. Fisher (1936) simplified to predict only the virginica class (i.e., as a binary classification problem).

Usage

```
iris.v
```

Format

A data frame containing 150 rows and 4 columns.

sep.len sepal length in cm

sep.wid sepal width in cm

pet.len petal length in cm

pet.wid petal width in cm

virginica *Criterion*: Does an iris belong to the class "virginica"?
Values: TRUE vs. FALSE (33.33% vs.66.67%).

Details

To improve usability, we made the following changes:

- The criterion was binarized from a factor variable with three levels (`Iris-setosa`, `Iris-versicolor`, `Iris-virginica`), into a logical variable (i.e., `TRUE` for all instances of `Iris-virginica` and `FALSE` for the two other levels).

Other than that, the data remains consistent with the original dataset.

Source

https://archive.ics.uci.edu/ml/datasets/Iris

References

Fisher, R.A. (1936): The use of multiple measurements in taxonomic problems. Annual Eugenics, 7, Part II, pp. 179–188.

See Also

Other datasets: `blood`, `breastcancer`, `car`, `contraceptive`, `creditapproval`, `fertility`, `forestfires`, `heart.cost`, `heart.test`, `heart.train`, `heartdisease`, `mushrooms`, `sonar`, `titanic`, `voting`, `wine`

---

mushrooms *Mushrooms data*

Description

Data describing poisonous vs. non-poisonous mushrooms.

Usage

```
mushrooms
```

Format

A data frame containing 8,124 rows and 23 columns.

See http://archive.ics.uci.edu/ml/machine-learning-databases/mushroom/agaricus-lepiota.names for column descriptions.

poisonous *Criterion*: Is the mushroom poisonous?
Values: `TRUE` (poisonous) vs. `FALSE` (eatable) (48.2% vs.\ 52.8%).

cshape cap-shape, character (bell=b, conical=c, convex=x, flat=f, knobbed=k, sunken=s)

csurface cap-surface, character (fibrous=f, grooves=g, scaly=y, smooth=s)

ccolor cap-color, character (brown=n, buff=b, cinnamon=c, gray=g, green=r, pink=p, purple=u, red=e, white=w, yellow=y)

bruises Are there bruises? logical (TRUE/FALSE)

odor character (almond=a, anise=l, creosote=c, fishy=y, foul=f, musty=m, none=n, pungent=p, spicy=s)

gattach gill-attachment, character (attached=a, descending=d, free=f, notched=n)

gspace gill-spacing, character (close=c, crowded=w, distant=d)

gsize gill-size, character (broad=b, narrow=n)

gcolor gill-color, character (black=k, brown=n, buff=b, chocolate=h, gray=g, green=r, orange=o, pink=p, purple=u, red=e, white=w, yellow=y)

sshape stalk-shape, character (enlarging=e, tapering=t)

sroot stalk-root, character (bulbous=b ,club=c, cup=u, equal=e, rhizomorphs=z, rooted=r)

ssaring stalk-surface-above-ring, character (fibrous=f, scaly=y, silky=k, smooth=s)

ssbring stalk-surface-below-ring, character (fibrous=f, scaly=y, silky=k, smooth=s)

scaring stalk-color-above-ring, character (brown=n, buff=b, cinnamon=c, gray=g, orange=o, pink=p, red=e, white=w, yellow=y)

scbring stalk-color-below-ring, character (brown=n, buff=b, cinnamon=c, gray=g, orange=o, pink=p, red=e, white=w, yellow=y)

vtype veil-type, character (partial=p, universal=u)

vcolor veil-color, character (brown=n, orange=o, white=w, yellow=y)

ringnum character (none=n, one=o, two=t)

ringtype character (cobwebby=c, evanescent=e, flaring=f, large=l, none=n, pendant=p, sheathing=s, zone=z)

sporepc spore-print-color, character (black=k, brown=n, buff=b, chocolate=h, green=r, orange=o, purple=u, white=w, yellow=y)

population character(abundant=a, clustered=c, numerous=n, scattered=s, several=v, solitary=y)

habitat character (grasses=g, leaves=l, meadows=m, paths=p, urban=u, waste=w, woods=d)

Details

This dataset includes descriptions of hypothetical samples corresponding to 23 species of gilled mushrooms in the Agaricus and Lepiota Family. Each species is classified as `poisonous` (True or False). The Guide clearly states that there is no simple rule for determining the edibility of a mushroom; no rule like “leaflets three, let it be” for Poisonous Oak and Ivy.

We made the following enhancements to the original data for improved usability:

- Any missing values, denoted as "?" in the dataset, were transformed into NAs.
- Binary factor variables with exclusive "t" and "f" values were converted to logical `TRUE/FALSE` vectors.
- The binary factor *criterion* variable with exclusive "p" and "e" values was converted to a logical `TRUE/FALSE` vector.

Other than that, the data remains consistent with the original dataset.

Source

https://archive.ics.uci.edu/ml/datasets/Mushroom

References

Mushroom records drawn from The Audubon Society Field Guide to North American Mushrooms (1981). G.H. Lincoff (Pres.), New York: A.A. Knopf.

See Also

Other datasets: blood, breastcancer, car, contraceptive, creditapproval, fertility, forestfires, heart.cost, heart.test, heart.train, heartdisease, iris.v, sonar, titanic, voting, wine

---

plot.FFTrees *Plot an* FFTrees *object*

Description

plot.FFTrees visualizes an FFTrees object created by the FFTrees function.

plot.FFTrees is the main plotting function of the **FFTrees** package and called when evaluating the generic plot on an FFTrees object.

plot.FFTrees visualizes a selected FFT, key data characteristics, and various aspects of classification performance.

As x may not contain test data, plot.FFTrees by default plots the performance characteristics for training data (i.e., fitting), rather than for test data (i.e., for prediction). When test data is available, specifying data = "test" plots prediction performance.

Whenever the sensitivity weight (sens.w) is set to its default of sens.w = 0.50, a level shows *balanced* accuracy (bacc). If, however, sens.w deviates from its default, the level shows the tree's *weighted* accuracy value (wacc) and the current sens.w value (below the level).

Many aspects of the plot (e.g., its panels) and the FFT's appearance (e.g., labels of its nodes and exits) can be customized by setting corresponding arguments.

Usage

```
## S3 method for class 'FFTrees'
plot(
  x = NULL,
  data = "train",
  what = "all",
  tree = 1,
  main = NULL,
  cue.labels = NULL,
  decision.labels = NULL,
  cue.cex = NULL,
  threshold.cex = NULL,
```

```
  decision.cex = 1,
  comp = TRUE,
  show.header = NULL,
  show.tree = NULL,
  show.confusion = NULL,
  show.levels = NULL,
  show.roc = NULL,
  show.icons = NULL,
  show.iconguide = NULL,
  hlines = TRUE,
  label.tree = NULL,
  label.performance = NULL,
  n.per.icon = NULL,
  level.type = "bar",
  which.tree = NULL,
  decision.names = NULL,
  stats = NULL,
  ...
)
```

Arguments

x	An FFTrees object created by the FFTrees function.
data	The type of data in x to be plotted (as a string) or a test dataset (as a data frame). • A valid data string must be either 'train' (for fitting performance) or 'test' (for prediction performance). • For a valid data frame, the specified tree is evaluated and plotted for this data (as 'test' data), but the global FFTrees object x remains unchanged unless it is re-assigned. By default, data = 'train' (as x may not contain test data).
what	What should be plotted (as a character string)? Valid options are: **'all'** Plot the tree diagram with all corresponding guides and performance statistics, but excluding cue accuracies. **'cues'** Plot only the marginal accuracy of cues in ROC space. Note that cue accuracies are *not* shown when calling what = 'all' and use the showcues function. **'icontree'** Plot tree diagram with icon arrays on exit nodes. Consider also setting n.per.icon and show.iconguide. **'tree'** Plot only the tree diagram. **'roc'** Plot only the performance of tree(s) (and comparison algorithms) in ROC space. Default: what = 'all'.
tree	The tree to be plotted (as an integer, only valid when the corresponding tree argument is non-empty). Default: tree = 1. To plot the best training or best test tree with respect to the goal specified during FFT construction, use 'best.train' or 'best.test', respectively.

main	The main plot label (as a character string).
cue.labels	An optional string of labels for the cues / nodes (as character vector).
decision.labels	A character vector of length 2 indicating the content-specific names for noise and signal predictions/exits.
cue.cex	The size of the cue labels (as numeric).
threshold.cex	The size of the threshold labels (as numeric).
decision.cex	The size of the decision labels (as numeric).
comp	Should the performance of competitive algorithms (e.g.; logistic regression, random forests, etc.) be shown in the ROC plot (if available, as logical)?
show.header	Show header with basic data properties (in top panel, as logical)?
show.tree	Show nodes and exits of FFT (in middle panel, as logical)?
show.confusion	Show a 2x2 confusion matrix (in bottom panel, as logical)?
show.levels	Show performance levels (in bottom panel, as logical)?
show.roc	Show ROC curve (in bottom panel, as logical)?
show.icons	Show exit cases as icon arrays (in middle panel, as logical)?
show.iconguide	Show icon guide (in middle panel, as logical)?
hlines	Show horizontal panel separation lines (as logical)? Default: hlines = TRUE.
label.tree	A label for the FFT (optional, as character string).
label.performance	A label for the performance section (optional, as character string).
n.per.icon	The number of cases represented by each icon (as numeric).
level.type	The type of performance levels to be drawn at the bottom (as character string, either "bar" or "line". Default: level.type = "bar".
which.tree	Deprecated argument. Use tree instead.
decision.names	Deprecated argument. Use decision.labels instead.
stats	Deprecated argument. Should statistical information be plotted (as logical)? Use what = "all" to include performance statistics and what = "tree" to plot only a tree diagram.
...	Graphical parameters (passed to text of panel titles, to showcues when what = 'cues', or to title when what = 'roc').

Value

An invisible FFTrees object x and a plot visualizing and describing an FFT (as side effect).

See Also

showcues for plotting cue accuracies; print.FFTrees for printing FFTs; summary.FFTrees for summarizing FFTs; FFTrees for creating FFTs from and applying them to data.

Other plot functions: showcues()

Examples

```
# Create FFTs (for heartdisease data):
heart_fft <- FFTrees(formula = diagnosis ~ .,
                     data = heart.train)

# Visualize the default FFT (Tree #1, what = 'all'):
plot(heart_fft, main = "Heart disease",
     decision.labels = c("Absent", "Present"))

# Visualize cue accuracies (in ROC space):
plot(heart_fft, what = "cues",  main = "Cue accuracies for heart disease data")

# Visualize tree diagram with icon arrays on exit nodes:
plot(heart_fft, what = "icontree", n.per.icon = 2,
     main = "Diagnosing heart disease")

# Visualize performance comparison in ROC space:
plot(heart_fft, what = "roc", main = "Performance comparison for heart disease data")

# Visualize predictions of FFT #2 (for new test data) with custom options:
plot(heart_fft, tree = 2, data = heart.test,
     main = "Predicting heart disease",
     cue.labels = c("1. thal?", "2. cp?", "3. ca?", "4. exang"),
     decision.labels = c("ok", "sick"), n.per.icon = 2,
     show.header = TRUE, show.confusion = FALSE, show.levels = FALSE, show.roc = FALSE,
     hlines = FALSE, font = 3, col = "steelblue")

# # For details, see
# vignette("FFTrees_plot", package = "FFTrees")
```

---

predict.FFTrees *Predict classification outcomes or probabilities from data*

Description

`predict.FFTrees` predicts binary classification outcomes or their probabilities from `newdata` for an `FFTrees` object.

Usage

```
## S3 method for class 'FFTrees'
predict(
  object = NULL,
  newdata = NULL,
  tree = 1,
  type = "class",
  sens.w = NULL,
```

```
  method = "laplace",
  data = NULL,
  ...
)
```

Arguments

object	An FFTrees object created by the FFTrees function.
newdata	dataframe. A data frame of test data.
tree	integer. Which tree in the object should be used? By default, tree = 1 is used.
type	string. What should be predicted? Can be "class", which returns a vector of class predictions, "prob" which returns a matrix of class probabilities, or "both" which returns a matrix with both class and probability predictions.
sens.w, data	deprecated
method	string. Method of calculating class probabilities. Either 'laplace', which applies the Laplace correction, or 'raw' which applies no correction.
...	Additional arguments passed on to predict.

Value

Either a logical vector of predictions, or a matrix of class probabilities.

See Also

print.FFTrees for printing FFTs; plot.FFTrees for plotting FFTs; summary.FFTrees for summarizing FFTs; FFTrees for creating FFTs from and applying them to data.

Examples

```
# Create training and test data:
set.seed(100)
breastcancer <- breastcancer[sample(nrow(breastcancer)), ]
breast.train <- breastcancer[1:150, ]
breast.test  <- breastcancer[151:303, ]

# Create an FFTrees object from the training data:
breast.fft <- FFTrees(
  formula = diagnosis ~ .,
  data = breast.train
)

# Predict classification outcomes for test data:
breast.fft.pred <- predict(breast.fft,
  newdata = breast.test
)

# Predict class probabilities for test data:
breast.fft.pred <- predict(breast.fft,
  newdata = breast.test,
```

```
  type = "prob"
)
```

`print.FFTrees` *Print basic information of fast-and-frugal trees (FFTs)*

Description

`print.FFTrees` prints basic information on FFTs for an `FFTrees` object `x`.

As `x` may not contain test data, `print.FFTrees` by default prints the performance characteristics for training data (i.e., fitting), rather than for test data (i.e., for prediction). When test data is available, specify `data = "test"` to print prediction performance.

Usage

```
## S3 method for class 'FFTrees'
print(x = NULL, tree = 1, data = "train", ...)
```

Arguments

`x`	An `FFTrees` object created by `FFTrees`.
`tree`	The tree to be printed (as an integer, only valid when the corresponding tree argument is non-empty). Default: `tree = 1`. To print the best training or best test tree with respect to the `goal` specified during FFT construction, use `"best.train"` or `"best.test"`, respectively.
`data`	The type of data in `x` to be printed (as a string) or a test dataset (as a data frame). • A valid data string must be either `'train'` (for fitting performance) or `'test'` (for prediction performance). • For a valid data frame, the specified tree is evaluated and printed for this data (as 'test' data), but the global `FFTrees` object `x` remains unchanged unless it is re-assigned. By default, `data = 'train'` (as `x` may not contain test data).
`...`	additional arguments passed to `print`.

Value

An invisible `FFTrees` object `x` and summary information on an FFT printed to the console (as side effect).

See Also

`plot.FFTrees` for plotting FFTs; `summary.FFTrees` for summarizing FFTs; `inwords` for obtaining a verbal description of FFTs; `FFTrees` for creating FFTs from and applying them to data.

read_fft_df *Read an FFT definition from tree definitions*

Description

read_fft_df reads and returns the definition of a single FFT (as a tidy data frame) from the multi-line FFT definitions of an FFTrees object.

read_fft_df allows reading individual tree definitions to manipulate them with other tree trimming functions.

write_fft_df provides the inverse functionality.

Usage

```
read_fft_df(ffts_df, tree = 1)
```

Arguments

ffts_df	A set of FFT definitions (as a data frame, usually from an FFTrees object, with suitable variable names to pass verify_ffts_df.
tree	The ID of the to-be-selected FFT (as an integer), corresponding to a tree in ffts_df. Default: tree = 1.

Value

One FFT definition (as a data frame in tidy format, with one row per node).

See Also

get_fft_df for getting the FFT definitions of an FFTrees object; write_fft_df for writing one FFT to tree definitions; add_fft_df for adding FFTs to tree definitions; FFTrees for creating FFTs from and applying them to data.

Other tree definition and manipulation functions: add_fft_df(), add_nodes(), drop_nodes(), edit_nodes(), flip_exits(), get_fft_df(), reorder_nodes(), select_nodes(), write_fft_df()

reorder_nodes *Reorder nodes in an FFT definition*

Description

reorder_nodes allows reordering the nodes in an existing FFT definition (in the tidy data frame format).

reorder_nodes allows to directly set and change the node order in an FFT definition by specifying nodes.

When a former non-final node becomes a final node, the exit type of the former final node is set to the signal value (i.e., exit_types[2]).

Usage

```
reorder_nodes(fft, order = NA, quiet = FALSE)
```

Arguments

fft	One FFT definition (as a data frame in tidy format, with one row per node).
order	The desired node order (as an integer vector). The values of order must be a permutation of 1:nrow(fft). Default: order = NA.
quiet	Hide feedback messages (as logical)? Default: quiet = FALSE.

Value

One FFT definition (as a data frame in tidy format, with one row per node).

See Also

add_nodes for adding nodes to an FFT definition; edit_nodes for editing nodes in an FFT definition; drop_nodes for deleting nodes from an FFT definition; flip_exits for reversing exits in an FFT definition; select_nodes for selecting nodes in an FFT definition; get_fft_df for getting the FFT definitions of an FFTrees object; read_fft_df for reading one FFT definition from tree definitions; add_fft_df for adding FFTs to tree definitions; FFTrees for creating FFTs from and applying them to data.

Other tree definition and manipulation functions: add_fft_df(), add_nodes(), drop_nodes(), edit_nodes(), flip_exits(), get_fft_df(), read_fft_df(), select_nodes(), write_fft_df()

---

select_nodes *Select nodes from an FFT definition*

Description

select_nodes selects one or more nodes from an existing FFT definition (by filtering the corresponding row(s) from the FFT definition in the tidy data frame format).

When not selecting the final node, the last selected node becomes the new final node (i.e., gains a second exit).

Duplicates in nodes are selected only once (rather than incrementally) and nodes not in the range 1:nrow(fft) are ignored.

select_nodes is the inverse function of drop_nodes.

Usage

```
select_nodes(fft, nodes = NA, quiet = FALSE)
```

Arguments

fft	One FFT definition (as a data frame in tidy format, with one row per node).
nodes	The FFT nodes to select (as an integer vector). Default: nodes = NA.
quiet	Hide feedback messages (as logical)? Default: quiet = FALSE.

Value

One FFT definition (as a data frame in tidy format, with one row per node).

See Also

add_nodes for adding nodes to an FFT definition; drop_nodes for deleting nodes from an FFT definition; edit_nodes for editing nodes in an FFT definition; flip_exits for reversing exits in an FFT definition; reorder_nodes for reordering nodes of an FFT definition; get_fft_df for getting the FFT definitions of an FFTrees object; read_fft_df for reading one FFT definition from tree definitions; add_fft_df for adding FFTs to tree definitions; FFTrees for creating FFTs from and applying them to data.

Other tree definition and manipulation functions: add_fft_df(), add_nodes(), drop_nodes(), edit_nodes(), flip_exits(), get_fft_df(), read_fft_df(), reorder_nodes(), write_fft_df()

---

showcues *Visualize cue accuracies (as points in ROC space)*

Description

showcues plots the cue accuracies of an FFTrees object created by the FFTrees function (as points in ROC space).

If the optional arguments cue.accuracies and alt.goal are specified, their values take precedence over the corresponding settings of an FFTrees object x (but do not change x).

showcues is called when the main plot.FFTrees function is set to what = "cues".

Usage

```
showcues(
  x = NULL,
  cue.accuracies = NULL,
  alt.goal = NULL,
  main = NULL,
  top = 5,
  quiet = list(ini = TRUE, fin = FALSE, set = TRUE),
  ...
)
```

Arguments

x	An FFTrees object created by the FFTrees function.
cue.accuracies	An optional data frame specifying cue accuracies directly (without specifying FFTrees object x).
alt.goal	An optional alternative goal to sort the current cue accuracies (without using the goal of FFTrees object x).
main	A main plot title (as character string).
top	How many of the top cues should be highlighted (as an integer)?
quiet	Should user feedback messages be suppressed (as a list of 3 logical arguments)? Default: quiet = list(ini = TRUE, fin = FALSE, set = FALSE).
...	Graphical parameters (passed to plot).

Value

A plot showing cue accuracies (of an FFTrees object) (as points in ROC space).

See Also

print.FFTrees for printing FFTs; plot.FFTrees for plotting FFTs; summary.FFTrees for summarizing FFTs; FFTrees for creating FFTs from and applying them to data.

Other plot functions: plot.FFTrees()

Examples

```
# Create fast-and-frugal trees (FFTs) for heart disease:
heart.fft <- FFTrees(formula = diagnosis ~ .,
                     data = heart.train,
                     data.test = heart.test,
                     main = "Heart Disease",
                     decision.labels = c("Healthy", "Diseased")
                     )

# Show cue accuracies (in ROC space):
showcues(heart.fft,
         main = "Predicting heart disease")
```

---

sonar *Sonar data*

Description

The file contains patterns of sonar signals bounced off a metal cylinder or bounced off a roughly cylindrical rock at various angles and under various conditions. The transmitted sonar signal is a frequency-modulated chirp, rising in frequency.

Usage

```
sonar
```

Format

A data frame containing 208 rows and 60 columns.

V1 Number in the range 0.0 to 1.0 that represents the energy within a particular frequency band, integrated over a certain period of time.

V2 Number in the range 0.0 to 1.0 that represents the energy within a particular frequency band, integrated over a certain period of time.

V3 Number in the range 0.0 to 1.0 that represents the energy within a particular frequency band, integrated over a certain period of time.

V4 Number in the range 0.0 to 1.0 that represents the energy within a particular frequency band, integrated over a certain period of time.

V5 Number in the range 0.0 to 1.0 that represents the energy within a particular frequency band, integrated over a certain period of time.

V6 Number in the range 0.0 to 1.0 that represents the energy within a particular frequency band, integrated over a certain period of time.

V7 Number in the range 0.0 to 1.0 that represents the energy within a particular frequency band, integrated over a certain period of time.

V8 Number in the range 0.0 to 1.0 that represents the energy within a particular frequency band, integrated over a certain period of time.

V9 Number in the range 0.0 to 1.0 that represents the energy within a particular frequency band, integrated over a certain period of time.

V10 Number in the range 0.0 to 1.0 that represents the energy within a particular frequency band, integrated over a certain period of time.

V11 Number in the range 0.0 to 1.0 that represents the energy within a particular frequency band, integrated over a certain period of time.

V12 Number in the range 0.0 to 1.0 that represents the energy within a particular frequency band, integrated over a certain period of time.

V13 Number in the range 0.0 to 1.0 that represents the energy within a particular frequency band, integrated over a certain period of time.

V14 Number in the range 0.0 to 1.0 that represents the energy within a particular frequency band, integrated over a certain period of time.

V15 Number in the range 0.0 to 1.0 that represents the energy within a particular frequency band, integrated over a certain period of time.

V16 Number in the range 0.0 to 1.0 that represents the energy within a particular frequency band, integrated over a certain period of time.

V17 Number in the range 0.0 to 1.0 that represents the energy within a particular frequency band, integrated over a certain period of time.

V18 Number in the range 0.0 to 1.0 that represents the energy within a particular frequency band, integrated over a certain period of time.

V19 Number in the range 0.0 to 1.0 that represents the energy within a particular frequency band, integrated over a certain period of time.

V20 Number in the range 0.0 to 1.0 that represents the energy within a particular frequency band, integrated over a certain period of time.

V21 Number in the range 0.0 to 1.0 that represents the energy within a particular frequency band, integrated over a certain period of time.

V22 Number in the range 0.0 to 1.0 that represents the energy within a particular frequency band, integrated over a certain period of time.

V23 Number in the range 0.0 to 1.0 that represents the energy within a particular frequency band, integrated over a certain period of time.

V24 Number in the range 0.0 to 1.0 that represents the energy within a particular frequency band, integrated over a certain period of time.

V25 Number in the range 0.0 to 1.0 that represents the energy within a particular frequency band, integrated over a certain period of time.

V26 Number in the range 0.0 to 1.0 that represents the energy within a particular frequency band, integrated over a certain period of time.

V27 Number in the range 0.0 to 1.0 that represents the energy within a particular frequency band, integrated over a certain period of time.

V28 Number in the range 0.0 to 1.0 that represents the energy within a particular frequency band, integrated over a certain period of time.

V29 Number in the range 0.0 to 1.0 that represents the energy within a particular frequency band, integrated over a certain period of time.

V30 Number in the range 0.0 to 1.0 that represents the energy within a particular frequency band, integrated over a certain period of time.

V31 Number in the range 0.0 to 1.0 that represents the energy within a particular frequency band, integrated over a certain period of time.

V32 Number in the range 0.0 to 1.0 that represents the energy within a particular frequency band, integrated over a certain period of time.

V33 Number in the range 0.0 to 1.0 that represents the energy within a particular frequency band, integrated over a certain period of time.

V34 Number in the range 0.0 to 1.0 that represents the energy within a particular frequency band, integrated over a certain period of time.

V35 Number in the range 0.0 to 1.0 that represents the energy within a particular frequency band, integrated over a certain period of time.

V36 Number in the range 0.0 to 1.0 that represents the energy within a particular frequency band, integrated over a certain period of time.

V37 Number in the range 0.0 to 1.0 that represents the energy within a particular frequency band, integrated over a certain period of time.

V38 Number in the range 0.0 to 1.0 that represents the energy within a particular frequency band, integrated over a certain period of time.

V39 Number in the range 0.0 to 1.0 that represents the energy within a particular frequency band, integrated over a certain period of time.

V40 Number in the range 0.0 to 1.0 that represents the energy within a particular frequency band, integrated over a certain period of time.

V41 Number in the range 0.0 to 1.0 that represents the energy within a particular frequency band, integrated over a certain period of time.

V42 Number in the range 0.0 to 1.0 that represents the energy within a particular frequency band, integrated over a certain period of time.

V43 Number in the range 0.0 to 1.0 that represents the energy within a particular frequency band, integrated over a certain period of time.

V44 Number in the range 0.0 to 1.0 that represents the energy within a particular frequency band, integrated over a certain period of time.

V45 Number in the range 0.0 to 1.0 that represents the energy within a particular frequency band, integrated over a certain period of time.

V46 Number in the range 0.0 to 1.0 that represents the energy within a particular frequency band, integrated over a certain period of time.

V47 Number in the range 0.0 to 1.0 that represents the energy within a particular frequency band, integrated over a certain period of time.

V48 Number in the range 0.0 to 1.0 that represents the energy within a particular frequency band, integrated over a certain period of time.

V49 Number in the range 0.0 to 1.0 that represents the energy within a particular frequency band, integrated over a certain period of time.

V50 Number in the range 0.0 to 1.0 that represents the energy within a particular frequency band, integrated over a certain period of time.

V51 Number in the range 0.0 to 1.0 that represents the energy within a particular frequency band, integrated over a certain period of time.

V52 Number in the range 0.0 to 1.0 that represents the energy within a particular frequency band, integrated over a certain period of time.

V53 Number in the range 0.0 to 1.0 that represents the energy within a particular frequency band, integrated over a certain period of time.

V54 Number in the range 0.0 to 1.0 that represents the energy within a particular frequency band, integrated over a certain period of time.

V55 Number in the range 0.0 to 1.0 that represents the energy within a particular frequency band, integrated over a certain period of time.

V56 Number in the range 0.0 to 1.0 that represents the energy within a particular frequency band, integrated over a certain period of time.

V57 Number in the range 0.0 to 1.0 that represents the energy within a particular frequency band, integrated over a certain period of time.

V58 Number in the range 0.0 to 1.0 that represents the energy within a particular frequency band, integrated over a certain period of time.

V59 Number in the range 0.0 to 1.0 that represents the energy within a particular frequency band, integrated over a certain period of time.

V60 Number in the range 0.0 to 1.0 that represents the energy within a particular frequency band, integrated over a certain period of time.

mine.crit *Criterion*: Did a sonar signal bounce off a metal cylinder (or a rock)?
Values: `TRUE` (metal cylinder) vs. `FALSE` (rock) (53.37% vs.\ 46.63%).

Details

We made the following enhancements to the original data for improved usability:

- The binary factor *criterion* variable with exclusive "m" and "r" values was converted to a logical `TRUE/FALSE` vector.

Other than that, the data remains consistent with the original dataset.

Source

https://archive.ics.uci.edu/ml/datasets/Connectionist+Bench+(Sonar,+Mines+vs.+Rocks)

References

Gorman, R. P., and Sejnowski, T. J. (1988). "Analysis of Hidden Units in a Layered Network Trained to Classify Sonar Targets" in Neural Networks, Vol. 1, pp. 75-89.

See Also

Other datasets: `blood`, `breastcancer`, `car`, `contraceptive`, `creditapproval`, `fertility`, `forestfires`, `heart.cost`, `heart.test`, `heart.train`, `heartdisease`, `iris.v`, `mushrooms`, `titanic`, `voting`, `wine`

---

summary.FFTrees *Summarize an* FFTrees *object*

Description

`summary.FFTrees` summarizes key contents of an `FFTrees` object.

Usage

```
## S3 method for class 'FFTrees'
summary(object, tree = NULL, ...)
```

Arguments

`object`	An `FFTrees` object.
`tree`	The tree to summarize (as an integer, but may be a vector). If `tree = NULL` (as per default) or exceeding the possible range `1:object$trees$n`, information on all trees in `object` is returned.
`...`	Additional arguments (currently ignored).

Details

Given an FFTrees object x, summary.FFTrees selects key parameters from x$params and provides the definitions and performance statistics for tree from x$trees. Inspect and query x for additional details.

summary.FFTrees returns an invisible list containing two elements:

1. definitions and corresponding performance measures of trees;
2. stats on decision frequencies, derived probabilities, and costs (separated by train and test).

A header prints descriptive information of the FFTrees object (to the console): Its main title, number of trees (object$trees$n), and the name of the criterion variable (object$criterion_name).

Per default, information on all available trees is shown and returned. Specifying tree filters the output list elements for the corresponding tree(s). When only a single tree is specified, the printed header includes a verbal description of the corresponding tree.

While summary.FFTrees provides key details about the specified tree(s), the individual decisions (stored in object$trees$decisions) are not shown or returned.

Value

An invisible list with elements containing the definitions and performance stats of the FFT(s) specified by tree(s).

See Also

print.FFTrees for printing FFTs; plot.FFTrees for plotting FFTs; inwords for obtaining a verbal description of FFTs; FFTrees for creating FFTs from and applying them to data.

---

titanic *Titanic survival data*

Description

Data indicating who survived on the Titanic.

Usage

```
titanic
```

Format

A data frame containing 2,201 rows and 4 columns.

class Factor - Class (first, second, third, or crew)

age Factor - Age group (child or adult)

sex Factor - Sex (male or female)

survived Logical - Whether the passenger survived (TRUE) or not (FALSE)

Details

See Titanic of the R **datasets** package for details and the same data (in a 4-dimensional table).

Source

https://www.encyclopedia-titanica.org

References

Dawson, Robert J. MacG. (1995), The 'Unusual Episode' Data Revisited. Journal of Statistics Education, 3. doi: 10.1080/10691898.1995.11910499.

See Also

Other datasets: blood, breastcancer, car, contraceptive, creditapproval, fertility, forestfires, heart.cost, heart.test, heart.train, heartdisease, iris.v, mushrooms, sonar, voting, wine

---

voting *Voting data*

Description

A dataset of votes for each of the U.S. House of Representatives Congressmen on the 16 key votes identified by the CQA.

Usage

```
voting
```

Format

A data frame containing 435 rows and 16 columns.

handicapped handicapped-infants, logical (TRUE, FALSE)

water water-project-cost-sharing, logical (TRUE, FALSE)

adoption adoption-of-the-budget-resolution, logical (TRUE, FALSE)

physician physician-fee-freeze, logical (TRUE, FALSE)

elsalvador el-salvador-aid, logical (TRUE, FALSE)

religionschool religious-groups-in-schools, logical (TRUE, FALSE)

satellite anti-satellite-test-ban, logical (TRUE, FALSE)

nicaraguan aid-to-nicaraguan-contras, logical (TRUE, FALSE)

mxmissile mxmissile, logical (TRUE, FALSE)

immigration immigration, logical (TRUE, FALSE)

synfuels synfuels-corporation-cutback, logical (TRUE, FALSE)

education education-spending, logical (TRUE, FALSE)

superfund superfund-right-to-sue, logical (TRUE, FALSE)

crime crime, logical (TRUE, FALSE)

dutyfree duty-free-exports, logical (TRUE, FALSE)

southafrica export-administration-act-south-africa, logical (TRUE, FALSE)

party.crit *Criterion*: Where the voters democratic (or republican) congressmen?
Values: `TRUE` (democrat) / `FALSE` (republican) (61.52% vs. 38.48%).

Details

The CQA lists nine different types of votes: voted for, paired for, and announced for (these three simplified to yea), voted against, paired against, and announced against (these three simplified to nay), voted present, voted present to avoid conflict of interest, and did not vote or otherwise make a position known (these three simplified to an unknown disposition).

We made the following enhancements to the original data for improved usability:

- Any missing values, denoted as "?" in the dataset, were transformed into NAs.
- Binary factor variables with exclusive "y" and "n" values were converted to logical TRUE/FALSE vectors.
- The binary character *criterion* variable with exclusive "democrat" and "republican" values was converted to a logical `TRUE/FALSE` vector.

Other than that, the data remains consistent with the original dataset.

Source

https://archive.ics.uci.edu/ml/datasets/Congressional+Voting+Records

References

Congressional Quarterly Almanac, 98th Congress, 2nd session 1984, Volume XL: Congressional Quarterly Inc. Washington, D.C., 1985.

See Also

Other datasets: `blood`, `breastcancer`, `car`, `contraceptive`, `creditapproval`, `fertility`, `forestfires`, `heart.cost`, `heart.test`, `heart.train`, `heartdisease`, `iris.v`, `mushrooms`, `sonar`, `titanic`, `wine`

wine *Wine tasting data*

Description

Chemical and tasting data from wines in North Portugal.

Usage

```
wine
```

Format

A data frame containing 6497 rows and 13 columns.

fixed.acidity fixed acidity (nummeric)

volatile.acidity volatile acidity (nummeric)

citric.acid citric acid (nummeric)

residual.sugar residual sugar (nummeric)

chlorides chlorides (nummeric)

free.sulfur.dioxide free sulfur dioxide (nummeric)

total.sulfur.dioxide total sulfur dioxide (nummeric)

density density (nummeric)

pH PH Value (nummeric)

sulphates Sulphates (nummeric)

alcohol Alcohol (nummeric)

quality Quality (nummeric, score between 0 and 10)

type *Criterion*: Is the wine `red` or `white`? (24.61% vs.75.39%)

Source

http://archive.ics.uci.edu/ml/datasets/Wine+Quality

References

P. Cortez, A. Cerdeira, F. Almeida, T. Matos and J. Reis. Modeling wine preferences by data mining from physicochemical properties. In Decision Support Systems, *Elsevier*, 47(4):547-553. ISSN: 0167-9236.

See Also

Other datasets: `blood`, `breastcancer`, `car`, `contraceptive`, `creditapproval`, `fertility`, `forestfires`, `heart.cost`, `heart.test`, `heart.train`, `heartdisease`, `iris.v`, `mushrooms`, `sonar`, `titanic`, `voting`

write_fft_df *Write an FFT definition to tree definitions*

Description

write_fft_df writes the definition of a single FFT (as a tidy data frame) into the one-line FFT definition used by an FFTrees object.

write_fft_df allows turning individual tree definitions into the one-line FFT definition format used by an FFTrees object.

read_fft_df provides the inverse functionality.

Usage

```
write_fft_df(fft, tree = -99L)
```

Arguments

fft	One FFT definition (as a data frame in tidy format, with one row per node).
tree	The ID of the to-be-written FFT (as an integer). Default: tree = -99L.

Value

An FFT definition in the one line FFT definition format used by an FFTrees object (as a data frame).

See Also

get_fft_df for getting the FFT definitions of an FFTrees object; read_fft_df for reading one FFT definition from tree definitions; add_fft_df for adding FFTs to tree definitions; FFTrees for creating FFTs from and applying them to data.

Other tree definition and manipulation functions: add_fft_df(), add_nodes(), drop_nodes(), edit_nodes(), flip_exits(), get_fft_df(), read_fft_df(), reorder_nodes(), select_nodes()

Index