Package 'ggRandomForests'

September 7, 2016

Type Package

Title Visually Exploring Random Forests

Version 2.0.1

Date 2016-09-07

Author John Ehrlinger <john.ehrlinger@gmail.com>

Maintainer John Ehrlinger <john.ehrlinger@gmail.com>

License GPL (>= 3)

URL https://github.com/ehrlinger/ggRandomForests

BugReports https://github.com/ehrlinger/ggRandomForests/issues

Description Graphic elements for exploring Random Forests using the 'randomForest' or 'randomForestSRC' package for survival, regression and classification forests and 'ggplot2' package plotting.

Depends R (>= 3.1.0), randomForestSRC (>= 1.5.5)

Imports randomForest, ggplot2, survival, parallel, tidyr

Suggests testthat, rmdformats, RColorBrewer, MASS, dplyr, knitr, rmarkdown, plot3D

RoxygenNote 5.0.1

NeedsCompilation no

Repository CRAN

Date/Publication 2016-09-07 23:21:30

R topics documented:

`ggRandomForests-package`

ggRandomForests: Visually Exploring Random Forests

Description

`ggRandomForests` is a utility package for `randomForestSRC` (Iswaran et.al. 2014, 2008, 2007) for survival, regression and classification forests and uses the `ggplot2` (Wickham 2009) package for plotting results. `ggRandomForests` is structured to extract data objects from the random forest and provides S3 functions for printing and plotting these objects.

The `randomForestSRC` package provides a unified treatment of Breiman's (2001) random forests for a variety of data settings. Regression and classification forests are grown when the response is numeric or categorical (factor) while survival and competing risk forests (Ishwaran et al. 2008, 2012) are grown for right-censored survival data.

Many of the figures created by the `ggRandomForests` package are also available directly from within the `randomForestSRC` package. However, `ggRandomForests` offers the following advantages:

- Separation of data and figures: ggRandomForest contains functions that operate on either the rfsrc forest object directly, or on the output from randomForestSRC post processing functions (i.e. plot.variable, var.select, find.interaction) to generate intermediate ggRandomForests data objects. S3 functions are provide to further process these objects and plot results using the ggplot2 graphics package. Alternatively, users can use these data objects for additional custom plotting or analysis operations.
- Each data object/figure is a single, self contained object. This allows simple modification and manipulation of the data or ggplot2 objects to meet users specific needs and requirements.
- The use of ggplot2 for plotting. We chose to use the ggplot2 package for our figures to allow users flexibility in modifying the figures to their liking. Each S3 plot function returns either a single ggplot2 object, or a list of ggplot2 objects, allowing users to use additional ggplot2 functions or themes to modify and customise the figures to their liking.

The ggRandomForests package contains the following data functions:

- gg_rfsrc: randomForest[SRC] predictions.
- gg_error: randomForest[SRC] convergence rate based on the OOB error rate.
- gg_roc: ROC curves for randomForest classification models.
- gg_vimp: Variable Importance ranking for variable selection.
- gg_minimal_depth: Minimal Depth ranking for variable selection (Ishwaran et.al. 2010).
- gg_minimal_vimp: Comparing Minimal Depth and VIMP rankings for variable selection.
- gg_interaction: Minimal Depth interaction detection (Ishwaran et.al. 2010)
- gg_variable: Marginal variable dependence.
- gg_partial: Partial (risk adjusted) variable dependence.
- gg_partial_coplot: Partial variable conditional dependence (computationally expensive).
- gg_survival: Kaplan-Meier/Nelson-Aalon hazard analysis.

Each of these data functions has an associated S3 plot function that returns ggplot2 objects, either individually or as a list, which can be further customised using standard ggplot2 commands.

References

Breiman, L. (2001). Random forests, Machine Learning, 45:5-32.

Ishwaran H. and Kogalur U.B. (2014). Random Forests for Survival, Regression and Classification (RF-SRC), R package version 1.5.5.12.

Ishwaran H. and Kogalur U.B. (2007). Random survival forests for R. R News 7(2), 25–31.

Ishwaran H., Kogalur U.B., Blackstone E.H. and Lauer M.S. (2008). Random survival forests. Ann. Appl. Statist. 2(3), 841–860.

Ishwaran, H., U. B. Kogalur, E. Z. Gorodeski, A. J. Minn, and M. S. Lauer (2010). High-dimensional variable selection for survival data. J. Amer. Statist. Assoc. 105, 205-217.

Ishwaran, H. (2007). Variable importance in binary regression trees and forests. Electronic J. Statist., 1, 519-537.

Wickham, H. ggplot2: elegant graphics for data analysis. Springer New York, 2009.

`cache_rfsrc_datasets` *Recreate the cached data sets for the ggRandomForests package*

Description

Recreate the cached data sets for the ggRandomForests package

Usage

```
cache_rfsrc_datasets(set = NA, save = TRUE, pth, ...)
```

Arguments

`set`	Defaults to all sets (NA), however for individual sets specify one or more of c("airq", "Boston", "iris", "mtcars", "pbc", "veteran")
`save`	Defaults to write files to the current data directory.
`pth`	the directory to store files.
`...`	extra arguments passed to randomForestSRC functions.

Details

Constructing random forests are computationally expensive, and the `ggRandomForests` operates directly on `randomForestSRC` objects. We cache computationally intensive `randomForestSRC` objects to improve the `ggRandomForests` examples, diagnostics and vignettes run times. The set of precompiled `randomForestSRC` objects are stored in the package data subfolder, however version changes in the dependant packages may break some functionality. This function was created to help the package developer deal with thoses changes. We make the function available to end users to create objects for further experimentation.

For the following data sets: #'

- `_iris` - The `iris` data set.
- `_airq` - The `airquality` data set.
- `_mtcars` - The `mtcars` data set.
- `_Boston` - The `Boston` housing data set (`MASS` package).
- `_pbc` - The `pbc` data set (`randomForestSRC` package).
- `_veteran` - The `veteran` data set (`randomForestSRC` package).

See Also

`iris` `airq` `mtcars` `Boston` `pbc` `veteran`

calc_auc *Area Under the ROC Curve calculator*

Description

Area Under the ROC Curve calculator

Usage

```
calc_auc(x)
```

Arguments

x	gg_roc object

Details

calc_auc uses the trapezoidal rule to calculate the area under the ROC curve.

This is a helper function for the gg_roc functions.

Value

AUC. 50% is random guessing, higher is better.

See Also

calc_roc gg_roc plot.gg_roc

Examples

```
##
## Taken from the gg_roc example
rfsrc_iris <- rfsrc(Species ~ ., data = iris)
#data(rfsrc_iris)

## Not run:
gg_dta <- gg_roc(rfsrc_iris, which.outcome=1)

calc_auc(gg_dta)

## End(Not run)

gg_dta <- gg_roc(rfsrc_iris, which.outcome=2)

calc_auc(gg_dta)
```

calc_roc.rfsrc *Reciever Operator Characteristic calculator*

Description

Reciever Operator Characteristic calculator

Usage

```
calc_roc.rfsrc(object, yvar, which.outcome = "all", oob = TRUE)
```

Arguments

object	rfsrc or predict.rfsrc object containing predicted response
yvar	True response variable
which.outcome	If defined, only show ROC for this response.
oob	Use OOB estimates, the normal validation method (TRUE)

Details

For a randomForestSRC prediction and the actual response value, calculate the specificity (1-False Positive Rate) and sensitivity (True Positive Rate) of a predictor.

This is a helper function for the gg_roc functions, and not intended for use by the end user.

Value

A gg_roc object

See Also

calc_auc gg_roc plot.gg_roc

Examples

```
## Taken from the gg_roc example
 rfsrc_iris <- rfsrc(Species ~ ., data = iris)
#data(rfsrc_iris)
gg_dta <- calc_roc.rfsrc(rfsrc_iris, rfsrc_iris$yvar, which.outcome=1, oob=TRUE)
gg_dta <- calc_roc.rfsrc(rfsrc_iris, rfsrc_iris$yvar, which.outcome=1, oob=FALSE)
```

`combine.gg_partial` *combine two gg_partial objects*

Description

The `combine.gg_partial` function assumes the two `gg_partial` objects were generated from the same `rfsrc` object. So, the function joins along the `gg_partial` list item names (one per partial plot variable). Further, we combine the two `gg_partial` objects along the group variable.

Hence, to join three `gg_partial` objects together (i.e. for three different time points from a survival random forest) would require two `combine.gg_partial` calls: One to join the first two `gg_partial` object, and one to append the third `gg_partial` object to the output from the first call. The second call will append a single `lbls` label to the `gg_partial` object.

Usage

```
combine.gg_partial(x, y, lbls, ...)
```

Arguments

`x`	`gg_partial` object
`y`	`gg_partial` object
`lbls`	vector of 2 strings to label the combined data.
`...`	not used

Value

`gg_partial` or `gg_partial_list` based on class of x and y.

Examples

```
## Not run:
# Load a set of plot.variable partial plot data
data(partial_pbc)

# A list of 2 plot.variable objects
length(partial_pbc)
class(partial_pbc)

class(partial_pbc[[1]])
class(partial_pbc[[2]])

# Create gg_partial objects
ggPrtl.1 <- gg_partial(partial_pbc[[1]])
ggPrtl.2 <- gg_partial(partial_pbc[[2]])

# Combine the objects to get multiple time curves
# along variables on a single figure.
ggpart <- combine.gg_partial(ggPrtl.1, ggPrtl.2,
```

```
                                lbls = c("1 year", "3 years"))

# Plot each figure separately
plot(ggpart)

# Get the continuous data for a panel of continuous plots.
ggcont <- ggpart
ggcont$edema <- ggcont$ascites <- ggcont$stage <- NULL
plot(ggcont, panel=TRUE)

# And the categorical for a panel of categorical plots.
nms <- colnames(sapply(ggcont, function(st){st}))
for(ind in nms){
   ggpart[[ind]] <- NULL
}
plot(ggpart, panel=TRUE)

## End(Not run)
```

`gg_error` *randomForestSRC error rate data object*

Description

Extract the cumulative (OOB) `randomForestSRC` error rate as a function of number of trees.

Usage

```
gg_error(object, ...)
```

Arguments

`object`	`rfsrc` object.
`...`	optional arguments (not used).

Details

The `gg_error` function simply returns the `rfsrc$err.rate` object as a data.frame, and assigns the class for connecting to the S3 `plot.gg_error` function.

Value

`gg_error` `data.frame` with one column indicating the tree number, and the remaining columns from the `rfsrc$err.rate` return value.

References

Breiman L. (2001). Random forests, Machine Learning, 45:5-32.

Ishwaran H. and Kogalur U.B. (2007). Random survival forests for R, Rnews, 7(2):25-31.

Ishwaran H. and Kogalur U.B. (2013). Random Forests for Survival, Regression and Classification (RF-SRC), R package version 1.4.

See Also

plot.gg_error rfsrc plot.rfsrc

Examples

```
## Examples from RFSRC package...
## ------------------------------------------------------------
## classification example
## ------------------------------------------------------------
## ------------- iris data
## You can build a randomForest
rfsrc_iris <- rfsrc(Species ~ ., data = iris)
# ... or load a cached randomForestSRC object
# data(rfsrc_iris, package="ggRandomForests")

# Get a data.frame containing error rates
gg_dta<- gg_error(rfsrc_iris)

# Plot the gg_error object
plot(gg_dta)

## ------------------------------------------------------------
## Regression example
## ------------------------------------------------------------
## Not run:
## ------------- airq data
rfsrc_airq <- rfsrc(Ozone ~ ., data = airquality, na.action = "na.impute")

# Get a data.frame containing error rates
gg_dta<- gg_error(rfsrc_airq)

# Plot the gg_error object
plot(gg_dta)

## End(Not run)
## Not run:
## ------------- Boston data
data(rfsrc_Boston, package="ggRandomForests")

# Get a data.frame containing error rates
gg_dta<- gg_error(rfsrc_Boston)

# Plot the gg_error object
plot(gg_dta)
```

```
## End(Not run)
## Not run:
## ------------- mtcars data

# Get a data.frame containing error rates
gg_dta<- gg_error(rfsrc_mtcars)

# Plot the gg_error object
plot(gg_dta)

## End(Not run)

## ------------------------------------------------------------
## Survival example
## ------------------------------------------------------------
## Not run:
## ------------- veteran data
## randomized trial of two treatment regimens for lung cancer
data(veteran, package = "randomForestSRC")
rfsrc_veteran <- rfsrc(Surv(time, status) ~ ., data = dta$veteran, ...)

gg_dta <- gg_error(rfsrc_veteran)
plot(gg_dta)

## End(Not run)
## Not run:
## ------------- pbc data
# Load a cached randomForestSRC object
data(rfsrc_pbc, package="ggRandomForests")

gg_dta <- gg_error(rfsrc_pbc)
plot(gg_dta)

## End(Not run)
```

`gg_interaction` *Minimal Depth Variable Interaction data object (`find.interaction`).*

Description

Converts the matrix returned from `find.interaction` to a `data.frame` and add attributes for S3 identification. If passed a `rfsrc` object, `gg_interaction` first runs the `find.interaction` function with all optional arguments.

Usage

```
gg_interaction(object, ...)
```

Arguments

object	a rfsrc object or the output from the find.interaction function call.
...	optional extra arguments passed to find.interaction.

Value

gg_interaction object

References

Ishwaran H. (2007). Variable importance in binary regression trees and forests, Electronic J. Statist., 1:519-537.

Ishwaran H., Kogalur U.B., Gorodeski E.Z, Minn A.J. and Lauer M.S. (2010). High-dimensional variable selection for survival data. J. Amer. Statist. Assoc., 105:205-217.

Ishwaran H., Kogalur U.B., Chen X. and Minn A.J. (2011). Random survival forests for high-dimensional data. Statist. Anal. Data Mining, 4:115-132.

See Also

rfsrc find.interaction max.subtree var.select vimp plot.gg_interaction

Examples

```
## Examples from randomForestSRC package...
## ------------------------------------------------------------
## find interactions, classification setting
## ------------------------------------------------------------
## Not run:
## -------- iris data
## iris.obj <- rfsrc(Species ~., data = iris)
## TODO: VIMP interactions not handled yet....
## randomForestSRC::find.interaction(iris.obj, method = "vimp", nrep = 3)
## interaction_iris <- randomForestSRC::find.interaction(iris.obj)
data(interaction_iris, package="ggRandomForests")
gg_dta <- gg_interaction(interaction_iris)

plot(gg_dta, xvar="Petal.Width")
plot(gg_dta, panel=TRUE)

## End(Not run)
## ------------------------------------------------------------
## find interactions, regression setting
## ------------------------------------------------------------
## Not run:
## -------- air quality data
## airq.obj <- rfsrc(Ozone ~ ., data = airquality)
##
## TODO: VIMP interactions not handled yet....
## randomForestSRC::find.interaction(airq.obj, method = "vimp", nrep = 3)
## interaction_airq <- randomForestSRC::find.interaction(airq.obj)
```

```
data(interaction_airq, package="ggRandomForests")
gg_dta <- gg_interaction(interaction_airq)

plot(gg_dta, xvar="Temp")
plot(gg_dta, xvar="Solar.R")

plot(gg_dta, panel=TRUE)

## End(Not run)
## Not run:
## -------- Boston data
data(interaction_Boston, package="ggRandomForests")
gg_dta <- gg_interaction(interaction_Boston)

plot(gg_dta, panel=TRUE)

## End(Not run)
## Not run:
## -------- mtcars data
data(interaction_mtcars, package="ggRandomForests")
gg_dta <- gg_interaction(interaction_mtcars)

plot(gg_dta, panel=TRUE)

## End(Not run)

## ------------------------------------------------------------
## find interactions, survival setting
## ------------------------------------------------------------
## -------- pbc data
## data(pbc, package = "randomForestSRC")
## pbc.obj <- rfsrc(Surv(days,status) ~ ., pbc, nsplit = 10)
## interaction_pbc <- randomForestSRC::find.interaction(pbc.obj, nvar = 8)
## Not run:
data(interaction_pbc, package="ggRandomForests")
gg_dta <- gg_interaction(interaction_pbc)

plot(gg_dta, xvar="bili")
plot(gg_dta, panel=TRUE)

## End(Not run)
## Not run:
## -------- veteran data
data(interaction_veteran, package="ggRandomForests")
gg_dta <- gg_interaction(interaction_veteran)

plot(gg_dta, panel=TRUE)

## End(Not run)
```

gg_minimal_depth *Minimal depth data object (*[randomForestSRC]{var.select}*)*

Description

the [randomForestSRC]{var.select} function implements random forest variable selection using tree minimal depth methodology. The gg_minimal_depth function takes the output from [randomForestSRC]{var.select} and creates a data.frame formatted for the plot.gg_minimal_depth function.

Usage

```
gg_minimal_depth(object, ...)
```

Arguments

object	A [randomForestSRC]{rfsrc} object, [randomForestSRC]{predict} object or the list from the [randomForestSRC]{var.select.rfsrc} function.
...	optional arguments passed to the [randomForestSRC]{var.select} function if operating on an [randomForestSRC]{rfsrc} object.

Value

gg_minimal_depth object, A modified list of variables from the [randomForestSRC]{var.select} function, ordered by minimal depth rank.

See Also

[randomForestSRC]{var.select} plot.gg_minimal_depth

Examples

```
## Examples from RFSRC package...
## ------------------------------------------------------------
## classification example
## ------------------------------------------------------------
## Not run:
 ## -------- iris data
## You can build a randomForest
# rfsrc_iris <- rfsrc(Species ~ ., data = iris)
# varsel_iris <- randomForestSRC::var.select(rfsrc_iris)
# ... or load a cached randomForestSRC object
data(varsel_iris, package="ggRandomForests")

# Get a data.frame containing minimaldepth measures
gg_dta<- gg_minimal_depth(varsel_iris)

# Plot the gg_minimal_depth object
plot(gg_dta)
```

```
## End(Not run)
## ------------------------------------------------------------
## Regression example
## ------------------------------------------------------------
## Not run:
## -------- air quality data
# rfsrc_airq <- rfsrc(Ozone ~ ., data = airquality, na.action = "na.impute")
# varsel_airq <- randomForestSRC::var.select(rfsrc_airq)
# ... or load a cached randomForestSRC object
data(varsel_airq, package="ggRandomForests")

# Get a data.frame containing error rates
gg_dta<- gg_minimal_depth(varsel_airq)

# Plot the gg_minimal_depth object
plot(gg_dta)

## End(Not run)
## Not run:
## -------- Boston data
data(varsel_Boston, package="ggRandomForests")

# Get a data.frame containing error rates
plot(gg_minimal_depth(varsel_Boston))

## End(Not run)
## Not run:
## -------- mtcars data
data(varsel_mtcars, package="ggRandomForests")

# Get a data.frame containing error rates
plot.gg_minimal_depth(varsel_mtcars)

## End(Not run)

## ------------------------------------------------------------
## Survival example
## ------------------------------------------------------------
## Not run:
## -------- veteran data
## veteran data
## randomized trial of two treatment regimens for lung cancer
# data(veteran, package = "randomForestSRC")
# rfsrc_veteran <- rfsrc(Surv(time, status) ~ ., data = veteran, ntree = 100)
# varsel_veteran <- randomForestSRC::var.select(rfsrc_veteran)
# Load a cached randomForestSRC object
data(varsel_veteran, package="ggRandomForests")

gg_dta <- gg_minimal_depth(varsel_veteran)
plot(gg_dta)

## End(Not run)
```

```
## Not run:
## -------- pbc data
data(varsel_pbc, package="ggRandomForests")

gg_dta <- gg_minimal_depth(varsel_pbc)
plot(gg_dta)

## End(Not run)
```

---

gg_minimal_vimp *Minimal depth vs VIMP camparison by variable rankings.*

Description

Minimal depth vs VIMP camparison by variable rankings.

Usage

```
gg_minimal_vimp(object, ...)
```

Arguments

object	A rfsrc object, predict.rfsrc object or the list from the var.select.rfsrc function.
...	optional arguments passed to the var.select function if operating on an rfsrc object.
	@return gg_minimal_vimp comparison object.
	@seealso plot.gg_minimal_vimp var.select
	@aliases gg_minimal_vimp

Examples

```
## Examples from RFSRC package...
## ------------------------------------------------------------
## classification example
## ------------------------------------------------------------
## Not run:
## -------- iris data
## You can build a randomForest
# rfsrc_iris <- rfsrc(Species ~ ., data = iris)
# varsel_iris <- randomForestSRC::var.select(rfsrc_iris)
# ... or load a cached randomForestSRC object
data(varsel_iris, package="ggRandomForests")

# Get a data.frame containing minimaldepth measures
gg_dta<- gg_minimal_vimp(varsel_iris)

# Plot the gg_minimal_depth object
plot(gg_dta)
```

```
## End(Not run)
## ------------------------------------------------------------
## Regression example
## ------------------------------------------------------------
## Not run:
## -------- air quality data
# rfsrc_airq <- rfsrc(Ozone ~ ., data = airquality, na.action = "na.impute")
# varsel_airq <- randomForestSRC::var.select(rfsrc_airq)
# ... or load a cached randomForestSRC object
data(varsel_airq, package="ggRandomForests")

# Get a data.frame containing error rates
gg_dta<- gg_minimal_vimp(varsel_airq)

# Plot the gg_minimal_vimp object
plot(gg_dta)

## End(Not run)
## Not run:
## -------- Boston data
data(varsel_Boston, package="ggRandomForests")

# Get a data.frame containing error rates
gg_dta<- gg_minimal_vimp(varsel_Boston)

# Plot the gg_minimal_vimp object
plot(gg_dta)

## End(Not run)
## Not run:
## -------- mtcars data
data(varsel_mtcars, package="ggRandomForests")

# Get a data.frame containing error rates
gg_dta<- gg_minimal_vimp(varsel_mtcars)

# Plot the gg_minimal_vimp object
plot(gg_dta)

## End(Not run)
## ------------------------------------------------------------
## Survival example
## ------------------------------------------------------------
## Not run:
## -------- veteran data
## randomized trial of two treatment regimens for lung cancer
# data(veteran, package = "randomForestSRC")
# rfsrc_veteran <- rfsrc(Surv(time, status) ~ ., data = veteran, ntree = 100)
# varsel_veteran <- randomForestSRC::var.select(rfsrc_veteran)
# Load a cached randomForestSRC object
data(varsel_veteran, package="ggRandomForests")
```

```
gg_dta <- gg_minimal_vimp(varsel_veteran)
plot(gg_dta)

## End(Not run)
## Not run:
## -------- pbc data
data(varsel_pbc, package="ggRandomForests")

gg_dta <- gg_minimal_vimp(varsel_pbc)
plot(gg_dta)

## End(Not run)
```

---

gg_partial *Partial variable dependence object*

Description

The plot.variable function returns a list of either marginal variable dependance or partial variable dependence data from a rfsrc object. The gg_partial function formulates the plot.variable output for partial plots (where partial=TRUE) into a data object for creation of partial dependence plots using the plot.gg_partial function.

Partial variable dependence plots are the risk adjusted estimates of the specified response as a function of a single covariate, possibly subsetted on other covariates.

An option named argument can name a column for merging multiple plots together

Usage

```
gg_partial(object, ...)
```

Arguments

object	the partial variable dependence data object from plot.variable function
...	optional arguments

Value

gg_partial object. A data.frame or list of data.frames corresponding the variables contained within the plot.variable output.

References

Friedman, Jerome H. 2000. "Greedy Function Approximation: A Gradient Boosting Machine." Annals of Statistics 29: 1189-1232.

See Also

plot.gg_partial plot.variable

Examples

```
## ------------------------------------------------------------
## classification
## ------------------------------------------------------------
## -------- iris data
## Not run:
## iris "Petal.Width" partial dependence plot
##
# rfsrc_iris <- rfsrc(Species ~., data = iris)
# partial_iris <- plot.variable(rfsrc_iris, xvar.names = "Petal.Width",
#                               partial=TRUE)
data(partial_iris, package="ggRandomForests")

gg_dta <- gg_partial(partial_iris)
plot(gg_dta)

## End(Not run)
## ------------------------------------------------------------
## regression
## ------------------------------------------------------------
## Not run:
## -------- air quality data
## airquality "Wind" partial dependence plot
##
# rfsrc_airq <- rfsrc(Ozone ~ ., data = airquality)
# partial_airq <- plot.variable(rfsrc_airq, xvar.names = "Wind",
#                               partial=TRUE, show.plot=FALSE)
data(partial_airq, package="ggRandomForests")

gg_dta <- gg_partial(partial_airq)
plot(gg_dta)

gg_dta.m <- gg_dta[["Month"]]
plot(gg_dta.m, notch=TRUE)

gg_dta[["Month"]] <- NULL
plot(gg_dta, panel=TRUE)

## End(Not run)
## Not run:
## -------- Boston data
data(partial_Boston, package="ggRandomForests")

gg_dta <- gg_partial(partial_Boston)
plot(gg_dta, panel=TRUE)

## End(Not run)
## Not run:
## -------- mtcars data
data(partial_mtcars, package="ggRandomForests")
gg_dta <- gg_partial(partial_mtcars)
```

```
gg_dta.cat <- gg_dta
gg_dta.cat[["disp"]] <- gg_dta.cat[["wt"]] <- gg_dta.cat[["hp"]] <- NULL
gg_dta.cat[["drat"]] <- gg_dta.cat[["carb"]] <- gg_dta.cat[["qsec"]] <- NULL

plot(gg_dta.cat, panel=TRUE, notch=TRUE)

gg_dta[["cyl"]] <- gg_dta[["vs"]] <- gg_dta[["am"]] <- NULL
gg_dta[["gear"]] <- NULL
plot(gg_dta, panel=TRUE)

## End(Not run)

## ------------------------------------------------------------
## survival examples
## ------------------------------------------------------------
## Not run:
## -------- veteran data
## survival "age" partial variable dependence plot
##
# data(veteran, package = "randomForestSRC")
# rfsrc_veteran <- rfsrc(Surv(time,status)~., veteran, nsplit = 10, ntree = 100)
#
## 30 day partial plot for age
# partial_veteran <- plot.variable(rfsrc_veteran, surv.type = "surv",
#                                  partial = TRUE, time=30,
#                                  xvar.names = "age",
#                                  show.plots=FALSE)
data(partial_veteran, package="ggRandomForests")

gg_dta <- gg_partial(partial_veteran[[1]])
plot(gg_dta)

gg_dta.cat <- gg_dta
gg_dta[["celltype"]] <- gg_dta[["trt"]] <- gg_dta[["prior"]] <- NULL
plot(gg_dta, panel=TRUE)

gg_dta.cat[["karno"]] <- gg_dta.cat[["diagtime"]] <- gg_dta.cat[["age"]] <- NULL
plot(gg_dta.cat, panel=TRUE, notch=TRUE)

gg_dta <- lapply(partial_veteran, gg_partial)
gg_dta <- combine.gg_partial(gg_dta[[1]], gg_dta[[2]] )

plot(gg_dta[["karno"]])
plot(gg_dta[["celltype"]])

gg_dta.cat <- gg_dta
gg_dta[["celltype"]] <- gg_dta[["trt"]] <- gg_dta[["prior"]] <- NULL
plot(gg_dta, panel=TRUE)

gg_dta.cat[["karno"]] <- gg_dta.cat[["diagtime"]] <- gg_dta.cat[["age"]] <- NULL
plot(gg_dta.cat, panel=TRUE, notch=TRUE)

## End(Not run)
```

```
## Not run:
## -------- pbc data
data("partial_pbc", package = "ggRandomForests")
data("varsel_pbc", package = "ggRandomForests")
xvar <- varsel_pbc$topvars

# Convert all partial plots to gg_partial objects
gg_dta <- lapply(partial_pbc, gg_partial)

# Combine the objects to get multiple time curves
# along variables on a single figure.
pbc_ggpart <- combine.gg_partial(gg_dta[[1]], gg_dta[[2]],
                                 lbls = c("1 Year", "3 Years"))

summary(pbc_ggpart)
class(pbc_ggpart[["bili"]])

# Plot the highest ranked variable, by name.
#plot(pbc_ggpart[["bili"]])

# Create a temporary holder and remove the stage and edema data
ggpart <- pbc_ggpart
ggpart$edema <- NULL

# Panel plot the remainder.
plot(ggpart, panel = TRUE)

#plot(pbc_ggpart[["edema"]], panel=TRUE) #,
     # notch = TRUE, alpha = .3, outlier.shape = NA)

## End(Not run)
```

`gg_partial_coplot.rfsrc` *Data structures for stratified partial coplots*

Description

Data structures for stratified partial coplots

Usage

```
## S3 method for class 'rfsrc'
gg_partial_coplot(object, xvar, groups, surv_type = c("mort",
  "rel.freq", "surv", "years.lost", "cif", "chf"), time, ...)
```

Arguments

`object`	`rfsrc` object

xvar	list of partial plot variables
groups	vector of stratification variable.
surv_type	for survival random forests, c("mort", "rel.freq", "surv", "years.lost", "cif", "chf")
time	vector of time points for survival random forests partial plots.
...	extra arguments passed to plot.variable function

Value

gg_partial_coplot object. An subclass of a gg_partial_list object

Examples

```
## Not run:
# Load the forest
data(rfsrc_pbc, package="ggRandomForests")

# Create the variable plot.
ggvar <- gg_variable(rfsrc_pbc, time = 1)

# Find intervals with similar number of observations.
copper_cts <-quantile_pts(ggvar$copper, groups = 6, intervals = TRUE)

# Create the conditional groups and add to the gg_variable object
copper_grp <- cut(ggvar$copper, breaks = copper_cts)

## End(Not run)
## Not run:
## We would run this, but it's expensive
partial_coplot_pbc <- gg_partial_coplot(rfsrc_pbc, xvar = "bili",
                                        groups = copper_grp,
                                        surv_type = "surv",
                                        time = 1,
                                        show.plots = FALSE)

## End(Not run)
## Not run:
## so load the cached set
data(partial_coplot_pbc, package="ggRandomForests")

# Partial coplot
plot(partial_coplot_pbc) #, se = FALSE)

## End(Not run)
```

gg_rfsrc.rfsrc *Predicted response data object*

Description

Extracts the predicted response values from the rfsrc object, and formats data for plotting the response using plot.gg_rfsrc.

Usage

```
## S3 method for class 'rfsrc'
gg_rfsrc(object, oob = TRUE, by, ...)
```

Arguments

object	rfsrc object
oob	boolean, should we return the oob prediction , or the full forest prediction.
by	stratifying variable in the training dataset, defaults to NULL
...	extra arguments

Details

surv_type ("surv", "chf", "mortality", "hazard") for survival forests

oob boolean, should we return the oob prediction , or the full forest prediction.

Value

gg_rfsrc object

See Also

plot.gg_rfsrc rfsrc plot.rfsrc gg_survival

Examples

```
## ------------------------------------------------------------
## classification example
## ------------------------------------------------------------
## -------- iris data
rfsrc_iris <- rfsrc(Species ~ ., data = iris)
#data(rfsrc_iris, package="ggRandomForests")
gg_dta<- gg_rfsrc(rfsrc_iris)

plot(gg_dta)

## ------------------------------------------------------------
## Regression example
## ------------------------------------------------------------
```

```
## Not run:
## -------- air quality data
# rfsrc_airq <- rfsrc(Ozone ~ ., data = airquality, na.action = "na.impute")
data(rfsrc_airq, package="ggRandomForests")
gg_dta<- gg_rfsrc(rfsrc_airq)

plot(gg_dta)

## End(Not run)
## Not run:
## -------- Boston data
data(rfsrc_Boston, package="ggRandomForests")
plot(rfsrc_Boston)

## End(Not run)
## Not run:
## -------- mtcars data
data(rfsrc_mtcars, package="ggRandomForests")
gg_dta<- gg_rfsrc(rfsrc_mtcars)

plot(gg_dta)

## End(Not run)
## ------------------------------------------------------------
## Survival example
## ------------------------------------------------------------
## Not run:
## -------- veteran data
## randomized trial of two treatment regimens for lung cancer
# data(veteran, package = "randomForestSRC")
# rfsrc_veteran <- rfsrc(Surv(time, status) ~ ., data = veteran, ntree = 100)
data(rfsrc_veteran, package = "ggRandomForests")
gg_dta <- gg_rfsrc(rfsrc_veteran)
plot(gg_dta)

gg_dta <- gg_rfsrc(rfsrc_veteran, conf.int=.95)
plot(gg_dta)

gg_dta <- gg_rfsrc(rfsrc_veteran, by="trt")
plot(gg_dta)

## End(Not run)
## Not run:
## -------- pbc data
## We don't run this because of bootstrap confidence limits
data(rfsrc_pbc, package = "ggRandomForests")

## End(Not run)
## Not run:
gg_dta <- gg_rfsrc(rfsrc_pbc)
plot(gg_dta)

gg_dta <- gg_rfsrc(rfsrc_pbc, conf.int=.95)
```

```
plot(gg_dta)

## End(Not run)
## Not run:
gg_dta <- gg_rfsrc(rfsrc_pbc, by="treatment")
plot(gg_dta)

## End(Not run)
```

`gg_roc.rfsrc` *ROC (Receiver operator curve) data from a classification random forest.*

Description

The sensitivity and specificity of a randomForests classification object.

Usage

```
## S3 method for class 'rfsrc'
gg_roc(object, which.outcome, oob = TRUE, ...)
```

Arguments

`object`	an `rfsrc` classification object
`which.outcome`	select the classification outcome of interest.
`oob`	use oob estimates (default TRUE)
`...`	extra arguments (not used)

Value

`gg_roc` `data.frame` for plotting ROC curves.

See Also

`plot.gg_roc` `rfsrc`

Examples

```
## ------------------------------------------------------------
## classification example
## ------------------------------------------------------------
## -------- iris data
rfsrc_iris <- rfsrc(Species ~ ., data = iris)
#data(rfsrc_iris, package="ggRandomForests")

# ROC for setosa
```

```
gg_dta <- gg_roc(rfsrc_iris, which.outcome=1)
plot(gg_dta)

# ROC for versicolor
gg_dta <- gg_roc(rfsrc_iris, which.outcome=2)
plot(gg_dta)

# ROC for virginica
gg_dta <- gg_roc(rfsrc_iris, which.outcome=3)
plot(gg_dta)
```

`gg_survival` *Nonparametric survival estimates.*

Description

Nonparametric survival estimates.

Usage

```
gg_survival(interval = NULL, censor = NULL, by = NULL, data,
  type = c("kaplan", "nelson"), ...)
```

Arguments

`interval`	name of the interval variable in the training dataset.
`censor`	name of the censoring variable in the training dataset.
`by`	stratifying variable in the training dataset, defaults to NULL
`data`	name of the training data.frame
`type`	one of ("kaplan","nelson"), defaults to kaplan-meier
`...`	extra arguments passed to kaplan or nelson functions.

Details

`gg_survival` is a wrapper function for generating nonparametric survival estimates using either `nelson`-aalen or `kaplan`-meier estimates.

Value

A `gg_survival` object created using the non-parametric kaplan-meier or nelson-aalon estimators.

See Also

`kaplan` `nelson` `plot.gg_survival`

Examples

```
## Not run:
## -------- pbc data
data(pbc, package="randomForestSRC")
pbc$time <- pbc$days/364.25

# This is the same as kaplan
gg_dta <- gg_survival(interval="time", censor="status",
                      data=pbc)

plot(gg_dta, error="none")
plot(gg_dta)

# Stratified on treatment variable.
gg_dta <- gg_survival(interval="time", censor="status",
                      data=pbc, by="treatment")

plot(gg_dta, error="none")
plot(gg_dta)

# ...with smaller confidence limits.
gg_dta <- gg_survival(interval="time", censor="status",
                      data=pbc, by="treatment", conf.int=.68)

plot(gg_dta, error="lines")

## End(Not run)
```

---

gg_variable *Marginal variable depedance data object.*

Description

plot.variable generates a data.frame containing the marginal variable dependance or the partial variable dependence. The gg_variable function creates a data.frame of containing the full set of covariate data (predictor variables) and the predicted response for each observation. Marginal dependence figures are created using the plot.gg_variable function.

Optional arguments time point (or vector of points) of interest (for survival forests only) time.labels If more than one time is specified, a vector of time labels for differentiating the time points (for survival forests only) oob indicate if predicted results should include oob or full data set.

Usage

```
gg_variable(object, ...)
```

Arguments

object	a rfsrc object
...	optional arguments

Details

The marginal variable dependence is determined by comparing relation between the predicted response from the randomforest and a covariate of interest.

The gg_variable function operates on a rfsrc object, or the output from the plot.variable function.

Value

gg_variable object

See Also

plot.gg_variable plot.variable

Examples

```
## ------------------------------------------------------------
## classification
## ------------------------------------------------------------
## -------- iris data
## iris
 rfsrc_iris <- rfsrc(Species ~., data = iris)
#data(rfsrc_iris, package="ggRandomForests")

gg_dta <- gg_variable(rfsrc_iris)
plot(gg_dta, xvar="Sepal.Width")
plot(gg_dta, xvar="Sepal.Length")

plot(gg_dta, xvar=rfsrc_iris$xvar.names,
     panel=TRUE) # , se=FALSE)

## ------------------------------------------------------------
## regression
## ------------------------------------------------------------
## Not run:
## -------- air quality data
#rfsrc_airq <- rfsrc(Ozone ~ ., data = airquality)
data(rfsrc_airq, package="ggRandomForests")
gg_dta <- gg_variable(rfsrc_airq)

# an ordinal variable
gg_dta[,"Month"] <- factor(gg_dta[,"Month"])

plot(gg_dta, xvar="Wind")
plot(gg_dta, xvar="Temp")
plot(gg_dta, xvar="Solar.R")


plot(gg_dta, xvar=c("Solar.R", "Wind", "Temp", "Day"), panel=TRUE)

plot(gg_dta, xvar="Month", notch=TRUE)
```

```
## End(Not run)
## Not run:
## -------- motor trend cars data
#rfsrc_mtcars <- rfsrc(mpg ~ ., data = mtcars)
data(rfsrc_mtcars, package="ggRandomForests")
gg_dta <- gg_variable(rfsrc_mtcars)

# mtcars$cyl is an ordinal variable
gg_dta$cyl <- factor(gg_dta$cyl)
gg_dta$am <- factor(gg_dta$am)
gg_dta$vs <- factor(gg_dta$vs)
gg_dta$gear <- factor(gg_dta$gear)
gg_dta$carb <- factor(gg_dta$carb)

plot(gg_dta, xvar="cyl")

# Others are continuous
plot(gg_dta, xvar="disp")
plot(gg_dta, xvar="hp")
plot(gg_dta, xvar="wt")


# panels
plot(gg_dta,xvar=c("disp","hp", "drat", "wt", "qsec"),  panel=TRUE)
plot(gg_dta, xvar=c("cyl", "vs", "am", "gear", "carb"), panel=TRUE, notch=TRUE)

## End(Not run)
## -------- Boston data

## ------------------------------------------------------------
## survival examples
## ------------------------------------------------------------
## Not run:
## -------- veteran data
## survival
# data(veteran, package = "randomForestSRC")
# rfsrc_veteran <- rfsrc(Surv(time,status)~., veteran, nsplit = 10, ntree = 100)
data(rfsrc_veteran, package="ggRandomForests")

# get the 1 year survival time.
gg_dta <- gg_variable(rfsrc_veteran, time=90)

# Generate variable dependance plots for age and diagtime
plot(gg_dta, xvar = "age")
plot(gg_dta, xvar = "diagtime", )

# Generate coplots
plot(gg_dta, xvar = c("age", "diagtime"), panel=TRUE, se=FALSE)

# If we want to compare survival at different time points, say 30, 90 day
# and 1 year
gg_dta <- gg_variable(rfsrc_veteran, time=c(30, 90, 365))
```

```
# Generate variable dependance plots for age and diagtime
plot(gg_dta, xvar = "age")

## End(Not run)
## -------- pbc data
```

---

gg_vimp *Variable Importance (VIMP) data object*

Description

gg_vimp Extracts the variable importance (VIMP) information from a a rfsrc object.

Usage

```
gg_vimp(object, nvar, ...)
```

Arguments

object	A rfsrc object or output from vimp
nvar	argument to control the number of variables included in the output.
...	arguments passed to the vimp.rfsrc function if the rfsrc object does not contain importance information.

Value

gg_vimp object. A data.frame of VIMP measures, in rank order.

References

Ishwaran H. (2007). Variable importance in binary regression trees and forests, *Electronic J. Statist.*, 1:519-537.

See Also

plot.gg_vimp rfsrc vimp

Examples

```
## ------------------------------------------------------------
## classification example
## ------------------------------------------------------------
## -------- iris data
rfsrc_iris <- rfsrc(Species ~ ., data = iris)
#data(rfsrc_iris, package="ggRandomForests")
gg_dta <- gg_vimp(rfsrc_iris)
plot(gg_dta)
```

```
## ------------------------------------------------------------
## regression example
## ------------------------------------------------------------
## Not run:
## -------- air quality data
# rfsrc_airq <- rfsrc(Ozone ~ ., airquality)
data(rfsrc_airq, package="ggRandomForests")
gg_dta <- gg_vimp(rfsrc_airq)
plot(gg_dta)

## End(Not run)
## Not run:
## -------- Boston data
data(rfsrc_Boston, package="ggRandomForests")
gg_dta <- gg_vimp(rfsrc_Boston)
plot(gg_dta)

## End(Not run)
## Not run:
## -------- mtcars data
data(rfsrc_mtcars, package="ggRandomForests")
gg_dta <- gg_vimp(rfsrc_mtcars)
plot(gg_dta)

## End(Not run)
## ------------------------------------------------------------
## survival example
## ------------------------------------------------------------
## Not run:
## -------- veteran data
data(rfsrc_veteran, package="ggRandomForests")
gg_dta <- gg_vimp(rfsrc_veteran)
plot(gg_dta)

## End(Not run)
## Not run:
## -------- pbc data
data(rfsrc_pbc, package="ggRandomForests")
gg_dta <- gg_vimp(rfsrc_pbc)
plot(gg_dta)

# Restrict to only the top 10.
gg_dta <- gg_vimp(rfsrc_pbc, nvar=10)
plot(gg_dta)

## End(Not run)
```

---

kaplan *nonparametric kaplan-meier estimates*

Description

nonparametric kaplan-meier estimates

Usage

```
kaplan(interval, censor, data, by = NULL, ...)
```

Arguments

interval	name of the interval variable in the training dataset.
censor	name of the censoring variable in the training dataset.
data	name of the training set data.frame
by	stratifying variable in the training dataset, defaults to NULL
...	arguments passed to the survfit function

Value

gg_survival object

See Also

gg_survival nelson plot.gg_survival

Examples

```
## Not run:
# These get run through the gg_survival examples.
data(pbc, package="randomForestSRC")
pbc$time <- pbc$days/364.25

# This is the same as gg_survival
gg_dta <- kaplan(interval="time", censor="status",
                     data=pbc)

plot(gg_dta, error="none")
plot(gg_dta)

# Stratified on treatment variable.
gg_dta <- gg_survival(interval="time", censor="status",
                     data=pbc, by="treatment")

plot(gg_dta, error="none")
plot(gg_dta)

## End(Not run)
```

logit_loess *logit_loess takes*

Description

logit_loess takes

Usage

```
logit_loess(gg_dta, xvar, level)
```

Arguments

gg_dta	dataset contains a yhat to smooth
xvar	name of x variable to smooth along
level	quantile level argument for qnorm function.

nelson *nonparametric Nelson-Aalen estimates*

Description

nonparametric Nelson-Aalen estimates

Usage

```
nelson(interval, censor, data, by = NULL, weight = NULL, ...)
```

Arguments

interval	name of the interval variable in the training dataset.
censor	name of the censoring variable in the training dataset.
data	name of the survival training data.frame
by	stratifying variable in the training dataset, defaults to NULL
weight	for each observation (default=NULL)
...	arguments passed to the survfit function

Value

gg_survival object

See Also

gg_survival nelson plot.gg_survival

Examples

```
## Not run:
# These get run through the gg_survival examples.
data(pbc, package="randomForestSRC")
pbc$time <- pbc$days/364.25

# This is the same as gg_survival
gg_dta <- nelson(interval="time", censor="status",
                     data=pbc)

plot(gg_dta, error="none")
plot(gg_dta)

# Stratified on treatment variable.
gg_dta <- gg_survival(interval="time", censor="status",
                     data=pbc, by="treatment")

plot(gg_dta, error="none")
plot(gg_dta, error="lines")
plot(gg_dta)

gg_dta <- gg_survival(interval="time", censor="status",
                     data=pbc, by="treatment",
                     type="nelson")

plot(gg_dta, error="bars")
plot(gg_dta)


## End(Not run)
```

`partial.rfsrc` *randomForestSRC partial dependence (data object) (modified from randomForestSRC V1.6.0)*

Description

calculate the partial dependence of an x-variable on the class probability (classification), response (regression), mortality (survival), or the expected years lost (competing risk) from a RF-SRC analysis.

Usage

```
partial.rfsrc(x, xvar.names, which.outcome, surv.type = c("mort", "rel.freq",
  "surv", "years.lost", "cif", "chf"), nvar, npts = 25, subset, granule = 5,
  ...)
```

Arguments

x	An object of class (rfsrc, grow), (rfsrc, synthetic), (rfsrc, predict).
xvar.names	Names of the x-variables to be used.
which.outcome	For classification families, an integer or character value specifying the class to focus on (defaults to the first class). For competing risk families, an integer value between 1 and J indicating the event of interest, where J is the number of event types. The default is to use the first event type.
surv.type	For survival families, specifies the predicted value. See details below.
nvar	Number of variables to be plotted. Default is all.
npts	Maximum number of points used when generating partial plots for continuous variables.
subset	Vector indicating which rows of the x-variable matrix x$xvar to use. All rows are used if not specified.
granule	Integer value controlling minimum number of unique values required to treat a variable as continuous. If there are fewer, the variable is treated as a factor
...	other used arguments. Included for compatibility with plot.variable calls.

Details

The vertical axis displays the ensemble predicted value, while x-variables are plotted on the horizontal axis.

1. For regression, the predicted response is used.
2. For classification, it is the predicted class probability specified by which.outcome.
3. For survival, the choices are:
   - Mortality (mort).
   - Relative frequency of mortality (rel.freq).
   - Predicted survival (surv)
4. For competing risks, the choices are:
   - The expected number of life years lost (years.lost).
   - The cumulative incidence function (cif).
   - The cumulative hazard function (chf).

   In all three cases, the predicted value is for the event type specified by which.outcome.

The y-value for a variable X, evaluated at $X = x$, is

$$\tilde{f}(x) = \frac{1}{n}\sum_{i=1}^{n} \hat{f}(x, x_{i,o}),$$

where $x_{i,o}$ represents the value for all other variables other than X for individual i and $\hat{f}$ is the predicted value. Generating partial plots can be very slow. Choosing a small value for npts can speed up computational times as this restricts the number of distinct x values used in computing $\tilde{f}$.

Calculating partial dependence data can be slow. Setting npts to a smaller number can help.

Author(s)

Hemant Ishwaran and Udaya B. Kogalur (Modified by John Ehrlinger)

References

Friedman J.H. (2001). Greedy function approximation: a gradient boosting machine, Ann. of Statist., 5:1189-1232.

Ishwaran H., Kogalur U.B. (2007). Random survival forests for R, Rnews, 7(2):25-31.

Ishwaran H., Kogalur U.B., Blackstone E.H. and Lauer M.S. (2008). Random survival forests, Ann. App. Statist., 2:841-860.

Ishwaran H., Gerds T.A., Kogalur U.B., Moore R.D., Gange S.J. and Lau B.M. (2014). Random survival forests for competing risks. To appear in Biostatistics.

See Also

`rfsrc`, `rfsrcSyn`, `predict.rfsrc`

Examples

```
## ------------------------------------------------------------
## survival/competing risk
## ------------------------------------------------------------

## survival
## Not run:
data(veteran, package = "randomForestSRC")
v.obj <- rfsrc(Surv(time,status)~., veteran, nsplit = 10, ntree = 100)
plot.variable(v.obj, plots.per.page = 3)
plot.variable(v.obj, plots.per.page = 2, xvar.names = c("trt", "karno", "age"))
plot.variable(v.obj, surv.type = "surv", nvar = 1)
plot.variable(v.obj, surv.type = "surv", partial = TRUE, smooth.lines = TRUE)
plot.variable(v.obj, surv.type = "rel.freq", partial = TRUE, nvar = 2)

## example of plot.variable calling a pre-processed plot.variable object
p.v <- plot.variable(v.obj, surv.type = "surv", partial = TRUE, smooth.lines = TRUE)
plot.variable(p.v)
p.v$plots.per.page <- 1
p.v$smooth.lines <- FALSE
plot.variable(p.v)

## End(Not run)
## Not run:
## competing risks
data(follic, package = "randomForestSRC")
follic.obj <- rfsrc(Surv(time, status) ~ ., follic, nsplit = 3, ntree = 100)
plot.variable(follic.obj, which.outcome = 2)

## End(Not run)
## ------------------------------------------------------------
## regression
```

```
## ------------------------------------------------------------
## Not run:
## airquality
airq.obj <- rfsrc(Ozone ~ ., data = airquality)
plot.variable(airq.obj, partial = TRUE, smooth.lines = TRUE)
## motor trend cars
mtcars.obj <- rfsrc(mpg ~ ., data = mtcars)
plot.variable(mtcars.obj, partial = TRUE, smooth.lines = TRUE)

## End(Not run)
## ------------------------------------------------------------
## classification
## ------------------------------------------------------------

## iris
#rfsrc_iris <- rfsrc(Species ~., data = iris)
#data(rfsrc_iris, package="ggRandomForests")
#gg_dta <- partial.rfsrc(rfsrc_iris, )

## Not run:
## motor trend cars: predict number of carburetors
mtcars2 <- mtcars
mtcars2$carb <- factor(mtcars2$carb,
                       labels = paste("carb", sort(unique(mtcars$carb))))
mtcars2.obj <- rfsrc(carb ~ ., data = mtcars2)
plot.variable(mtcars2.obj, partial = TRUE)

## End(Not run)
```

`plot.gg_error` *Plot a* `gg_error` *object*

Description

A plot of the cumulative OOB error rates of the random forest as a function of number of trees.

Usage

```
## S3 method for class 'gg_error'
plot(x, ...)
```

Arguments

`x`	gg_error object created from a `rfsrc` object
`...`	extra arguments passed to `ggplot` functions

Details

The gg_error plot is used to track the convergence of the randomForest. This figure is a reproduction of the error plot from the plot.rfsrc function.

Value

ggplot object

References

Breiman L. (2001). Random forests, Machine Learning, 45:5-32.

Ishwaran H. and Kogalur U.B. (2007). Random survival forests for R, Rnews, 7(2):25-31.

Ishwaran H. and Kogalur U.B. (2013). Random Forests for Survival, Regression and Classification (RF-SRC), R package version 1.4.

See Also

gg_error rfsrc plot.rfsrc

Examples

```
## Not run:
## Examples from RFSRC package...
## ------------------------------------------------------------
## classification example
## ------------------------------------------------------------
## ------------- iris data
## You can build a randomForest
# rfsrc_iris <- rfsrc(Species ~ ., data = iris)
# ... or load a cached randomForestSRC object
data(rfsrc_iris, package="ggRandomForests")

# Get a data.frame containing error rates
gg_dta<- gg_error(rfsrc_iris)

# Plot the gg_error object
plot(gg_dta)

## ------------------------------------------------------------
## Regression example
## ------------------------------------------------------------
## ------------- airq data
# rfsrc_airq <- rfsrc(Ozone ~ ., data = airquality, na.action = "na.impute")
# ... or load a cached randomForestSRC object
data(rfsrc_airq, package="ggRandomForests")

# Get a data.frame containing error rates
gg_dta<- gg_error(rfsrc_airq)

# Plot the gg_error object
plot(gg_dta)
```

```
## ------------- Boston data
data(rfsrc_Boston, package="ggRandomForests")

# Get a data.frame containing error rates
gg_dta<- gg_error(rfsrc_Boston)

# Plot the gg_error object
plot(gg_dta)

## ------------- mtcars data
data(rfsrc_mtcars, package="ggRandomForests")

# Get a data.frame containing error rates
gg_dta<- gg_error(rfsrc_mtcars)

# Plot the gg_error object
plot(gg_dta)

## ------------------------------------------------------------
## Survival example
## ------------------------------------------------------------
## ------------- veteran data
## randomized trial of two treatment regimens for lung cancer
# data(veteran, package = "randomForestSRC")
# rfsrc_veteran <- rfsrc(Surv(time, status) ~ ., data = veteran, ntree = 100)

# Load a cached randomForestSRC object
data(rfsrc_veteran, package="ggRandomForests")

gg_dta <- gg_error(rfsrc_veteran)
plot(gg_dta)

## ------------- pbc data
# Load a cached randomForestSRC object
data(rfsrc_pbc, package="ggRandomForests")

gg_dta <- gg_error(rfsrc_pbc)
plot(gg_dta)

## End(Not run)
```

`plot.gg_interaction` *plot.gg_interaction Plot a* `gg_interaction` *object,*

Description

plot.gg_interaction Plot a `gg_interaction` object,

Usage

```
## S3 method for class 'gg_interaction'
plot(x, xvar, lbls, ...)
```

Arguments

x	gg_interaction object created from a rfsrc object
xvar	variable (or list of variables) of interest.
lbls	A vector of alternative variable names.
...	arguments passed to the gg_interaction function.

Value

ggplot object

References

Breiman L. (2001). Random forests, Machine Learning, 45:5-32.

Ishwaran H. and Kogalur U.B. (2007). Random survival forests for R, Rnews, 7(2):25-31.

Ishwaran H. and Kogalur U.B. (2013). Random Forests for Survival, Regression and Classification (RF-SRC), R package version 1.4.

See Also

rfsrc find.interaction max.subtree var.select vimp plot.gg_interaction

Examples

```
## Not run:
## Examples from randomForestSRC package...
## ------------------------------------------------------------
## find interactions, classification setting
## ------------------------------------------------------------
## -------- iris data
## iris.obj <- rfsrc(Species ~., data = iris)
## TODO: VIMP interactions not handled yet....
## find.interaction(iris.obj, method = "vimp", nrep = 3)
## interaction_iris <- find.interaction(iris.obj)
data(interaction_iris, package="ggRandomForests")
gg_dta <- gg_interaction(interaction_iris)

plot(gg_dta, xvar="Petal.Width")
plot(gg_dta, xvar="Petal.Length")
plot(gg_dta, panel=TRUE)

## ------------------------------------------------------------
## find interactions, regression setting
## ------------------------------------------------------------
## -------- air quality data
```

```
## airq.obj <- rfsrc(Ozone ~ ., data = airquality)
##
## TODO: VIMP interactions not handled yet....
## find.interaction(airq.obj, method = "vimp", nrep = 3)
## interaction_airq <- find.interaction(airq.obj)
data(interaction_airq, package="ggRandomForests")
gg_dta <- gg_interaction(interaction_airq)

plot(gg_dta, xvar="Temp")
plot(gg_dta, xvar="Solar.R")
plot(gg_dta, panel=TRUE)

## -------- Boston data
data(interaction_Boston, package="ggRandomForests")
gg_dta <- gg_interaction(interaction_Boston)

plot(gg_dta, panel=TRUE)

## -------- mtcars data
data(interaction_mtcars, package="ggRandomForests")
gg_dta <- gg_interaction(interaction_mtcars)

plot(gg_dta, panel=TRUE)

## ------------------------------------------------------------
## find interactions, survival setting
## ------------------------------------------------------------
## -------- pbc data
## data(pbc, package = "randomForestSRC")
## pbc.obj <- rfsrc(Surv(days,status) ~ ., pbc, nsplit = 10)
## interaction_pbc <- find.interaction(pbc.obj, nvar = 8)
data(interaction_pbc, package="ggRandomForests")
gg_dta <- gg_interaction(interaction_pbc)

plot(gg_dta, xvar="bili")
plot(gg_dta, xvar="copper")
plot(gg_dta, panel=TRUE)

## -------- veteran data
data(interaction_veteran, package="ggRandomForests")
gg_dta <- gg_interaction(interaction_veteran)

plot(gg_dta, panel=TRUE)


## End(Not run)
```

---

plot.gg_minimal_depth *Plot a* gg_minimal_depth *object for random forest variable ranking.*

Description

Plot a gg_minimal_depth object for random forest variable ranking.

Usage

```
## S3 method for class 'gg_minimal_depth'
plot(x, selection = FALSE, type = c("named",
  "rank"), lbls, ...)
```

Arguments

x	gg_minimal_depth object created from a rfsrc object
selection	should we restrict the plot to only include variables selected by the minimal depth criteria (boolean).
type	select type of y axis labels c("named","rank")
lbls	a vector of alternative variable names.
...	optional arguments passed to gg_minimal_depth

Value

ggplot object

References

Breiman L. (2001). Random forests, Machine Learning, 45:5-32.

Ishwaran H. and Kogalur U.B. (2007). Random survival forests for R, Rnews, 7(2):25-31.

Ishwaran H. and Kogalur U.B. (2014). Random Forests for Survival, Regression and Classification (RF-SRC), R package version 1.5.

See Also

var.select gg_minimal_depth

Examples

```
## Not run:
## Examples from RFSRC package...
## ------------------------------------------------------------
## classification example
## ------------------------------------------------------------
## -------- iris data
## You can build a randomForest
# rfsrc_iris <- rfsrc(Species ~ ., data = iris)
# varsel_iris <- var.select(rfsrc_iris)
# ... or load a cached randomForestSRC object
data(varsel_iris, package="ggRandomForests")

# Get a data.frame containing minimaldepth measures
```

```
gg_dta<- gg_minimal_depth(varsel_iris)

# Plot the gg_minimal_depth object
plot(gg_dta)

## ------------------------------------------------------------
## Regression example
## ------------------------------------------------------------
## -------- air quality data
# rfsrc_airq <- rfsrc(Ozone ~ ., data = airquality, na.action = "na.impute")
# varsel_airq <- var.select(rfsrc_airq)
# ... or load a cached randomForestSRC object
data(varsel_airq, package="ggRandomForests")

# Get a data.frame containing error rates
gg_dta<- gg_minimal_depth(varsel_airq)

# Plot the gg_minimal_depth object
plot(gg_dta)

## -------- Boston data
data(varsel_Boston, package="ggRandomForests")

# Get a data.frame containing error rates
plot(gg_minimal_depth(varsel_Boston))

## -------- mtcars data
data(varsel_mtcars, package="ggRandomForests")

# Get a data.frame containing error rates
plot.gg_minimal_depth(varsel_mtcars)

## ------------------------------------------------------------
## Survival example
## ------------------------------------------------------------
## -------- veteran data
## veteran data
## randomized trial of two treatment regimens for lung cancer
# data(veteran, package = "randomForestSRC")
# rfsrc_veteran <- rfsrc(Surv(time, status) ~ ., data = veteran, ntree = 100)
# varsel_veteran <- var.select(rfsrc_veteran)
# Load a cached randomForestSRC object
data(varsel_veteran, package="ggRandomForests")

gg_dta <- gg_minimal_depth(varsel_veteran)
plot(gg_dta)

## -------- pbc data
data(varsel_pbc, package="ggRandomForests")

gg_dta <- gg_minimal_depth(varsel_pbc)
plot(gg_dta)
```

```
## End(Not run)
```

`plot.gg_minimal_vimp` *Plot a* `gg_minimal_vimp` *object for comparing the Minimal Depth and VIMP variable rankings.*

Description

Plot a `gg_minimal_vimp` object for comparing the Minimal Depth and VIMP variable rankings.

Usage

```
## S3 method for class 'gg_minimal_vimp'
plot(x, nvar, lbls, ...)
```

Arguments

`x`	`gg_minimal_depth` object created from a `var.select` object
`nvar`	should the figure be restricted to a subset of the points.
`lbls`	a vector of alternative variable names.
`...`	optional arguments (not used)

Value

`ggplot` object

See Also

`gg_minimal_vimp` `var.select`

Examples

```
## Not run:
## Examples from RFSRC package...
## ------------------------------------------------------------
## classification example
## ------------------------------------------------------------
## -------- iris data
## You can build a randomForest
# rfsrc_iris <- rfsrc(Species ~ ., data = iris)
# varsel_iris <- var.select(rfsrc_iris)
# ... or load a cached randomForestSRC object
data(varsel_iris, package="ggRandomForests")

# Get a data.frame containing minimaldepth measures
gg_dta<- gg_minimal_vimp(varsel_iris)
```

```
# Plot the gg_minimal_depth object
plot(gg_dta)

## ------------------------------------------------------------
## Regression example
## ------------------------------------------------------------
## -------- air quality data
rfsrc_airq <- rfsrc(Ozone ~ ., data = airquality, na.action = "na.impute")
varsel_airq <- var.select(rfsrc_airq)

# Get a data.frame containing error rates
gg_dta<- gg_minimal_vimp(varsel_airq)

# Plot the gg_minimal_vimp object
plot(gg_dta)

## -------- Boston data
data(varsel_Boston, package="ggRandomForests")

# Get a data.frame containing error rates
gg_dta<- gg_minimal_vimp(varsel_Boston)

# Plot the gg_minimal_vimp object
plot(gg_dta)

## -------- mtcars data
data(varsel_mtcars, package="ggRandomForests")

# Get a data.frame containing error rates
gg_dta<- gg_minimal_vimp(varsel_mtcars)

# Plot the gg_minimal_vimp object
plot(gg_dta)

## ------------------------------------------------------------
## Survival example
## ------------------------------------------------------------
## -------- veteran data
## randomized trial of two treatment regimens for lung cancer
# data(veteran, package = "randomForestSRC")
# rfsrc_veteran <- rfsrc(Surv(time, status) ~ ., data = veteran, ntree = 100)
# varsel_veteran <- var.select(rfsrc_veteran)
# Load a cached randomForestSRC object
data(varsel_veteran, package="ggRandomForests")

gg_dta <- gg_minimal_vimp(varsel_veteran)
plot(gg_dta)

## -------- pbc data
data(varsel_pbc, package="ggRandomForests")

gg_dta <- gg_minimal_vimp(varsel_pbc)
```

```
plot(gg_dta)

## End(Not run)
```

`plot.gg_partial` *Partial variable dependence plot, operates on a* `gg_partial` *object.*

Description

Generate a risk adjusted (partial) variable dependence plot. The function plots the `rfsrc` response variable (y-axis) against the covariate of interest (specified when creating the `gg_partial` object).

Usage

```
## S3 method for class 'gg_partial'
plot(x, points = TRUE, error = c("none", "shade",
  "bars", "lines"), ...)
```

Arguments

`x`	`gg_partial` object created from a `rfsrc` forest object
`points`	plot points (boolean) or a smooth line.
`error`	"shade", "bars", "lines" or "none"
`...`	extra arguments passed to `ggplot2` functions.

Value

`ggplot` object

References

Breiman L. (2001). Random forests, Machine Learning, 45:5-32.

Ishwaran H. and Kogalur U.B. (2007). Random survival forests for R, Rnews, 7(2):25-31.

Ishwaran H. and Kogalur U.B. (2013). Random Forests for Survival, Regression and Classification (RF-SRC), R package version 1.4.

See Also

`plot.variable` `gg_partial` `plot.gg_partial_list` `gg_variable` `plot.gg_variable`

Examples

```
## Not run:
## ------------------------------------------------------------
## classification
## ------------------------------------------------------------
## -------- iris data

## iris "Petal.Width" partial dependence plot
##
# rfsrc_iris <- rfsrc(Species ~., data = iris)
# partial_iris <- plot.variable(rfsrc_iris, xvar.names = "Petal.Width",
#                               partial=TRUE)
data(partial_iris, package="ggRandomForests")

gg_dta <- gg_partial(partial_iris)
plot(gg_dta)

## ------------------------------------------------------------
## regression
## ------------------------------------------------------------
## -------- air quality data
## airquality "Wind" partial dependence plot
##
# rfsrc_airq <- rfsrc(Ozone ~ ., data = airquality)
# partial_airq <- plot.variable(rfsrc_airq, xvar.names = "Wind",
#                               partial=TRUE, show.plot=FALSE)
data(partial_airq, package="ggRandomForests")

gg_dta <- gg_partial(partial_airq)
plot(gg_dta)

gg_dta.m <- gg_dta[["Month"]]
plot(gg_dta.m, notch=TRUE)

gg_dta[["Month"]] <- NULL
plot(gg_dta, panel=TRUE)

## -------- Boston data
data(partial_Boston, package="ggRandomForests")

gg_dta <- gg_partial(partial_Boston)
plot(gg_dta)
plot(gg_dta, panel=TRUE)

## -------- mtcars data
data(partial_mtcars, package="ggRandomForests")

gg_dta <- gg_partial(partial_mtcars)

plot(gg_dta)

gg_dta.cat <- gg_dta
```

```
gg_dta.cat[["disp"]] <- gg_dta.cat[["wt"]] <- gg_dta.cat[["hp"]] <- NULL
gg_dta.cat[["drat"]] <- gg_dta.cat[["carb"]] <- gg_dta.cat[["qsec"]] <- NULL

plot(gg_dta.cat, panel=TRUE)

gg_dta[["cyl"]] <- gg_dta[["vs"]] <- gg_dta[["am"]] <- NULL
gg_dta[["gear"]] <- NULL
plot(gg_dta, panel=TRUE)

## ------------------------------------------------------------
## survival examples
## ------------------------------------------------------------
## -------- veteran data
## survival "age" partial variable dependence plot
##
# data(veteran, package = "randomForestSRC")
# rfsrc_veteran <- rfsrc(Surv(time,status)~., veteran, nsplit = 10, ntree = 100)
#
## 30 day partial plot for age
# partial_veteran <- plot.variable(rfsrc_veteran, surv.type = "surv",
#                                  partial = TRUE, time=30,
#                                  xvar.names = "age",
#                                  show.plots=FALSE)
data(partial_veteran, package="ggRandomForests")

gg_dta <- gg_partial(partial_veteran[[1]])
plot(gg_dta)

gg_dta.cat <- gg_dta
gg_dta[["celltype"]] <- gg_dta[["trt"]] <- gg_dta[["prior"]] <- NULL
plot(gg_dta, panel=TRUE)

gg_dta.cat[["karno"]] <- gg_dta.cat[["diagtime"]] <- gg_dta.cat[["age"]] <- NULL
plot(gg_dta.cat, panel=TRUE, notch=TRUE)

gg_dta <- lapply(partial_veteran, gg_partial)
length(gg_dta)
gg_dta <- combine.gg_partial(gg_dta[[1]], gg_dta[[2]] )

plot(gg_dta[["karno"]])
plot(gg_dta[["celltype"]])

gg_dta.cat <- gg_dta
gg_dta[["celltype"]] <- gg_dta[["trt"]] <- gg_dta[["prior"]] <- NULL
plot(gg_dta, panel=TRUE)

gg_dta.cat[["karno"]] <- gg_dta.cat[["diagtime"]] <- gg_dta.cat[["age"]] <- NULL
plot(gg_dta.cat, panel=TRUE, notch=TRUE)

## -------- pbc data

## End(Not run)
```

plot.gg_partial_list *Partial variable dependence plot, operates on a* gg_partial_list *object.*

Description

Generate a risk adjusted (partial) variable dependence plot. The function plots the rfsrc response variable (y-axis) against the covariate of interest (specified when creating the gg_partial_list object).

Usage

```
## S3 method for class 'gg_partial_list'
plot(x, points = TRUE, panel = FALSE, ...)
```

Arguments

x	gg_partial_list object created from a gg_partial forest object
points	plot points (boolean) or a smooth line.
panel	should the entire list be plotted together?
...	extra arguments

Value

list of ggplot objects, or a single faceted ggplot object

References

Breiman L. (2001). Random forests, Machine Learning, 45:5-32.

Ishwaran H. and Kogalur U.B. (2007). Random survival forests for R, Rnews, 7(2):25-31.

Ishwaran H. and Kogalur U.B. (2013). Random Forests for Survival, Regression and Classification (RF-SRC), R package version 1.4.

See Also

plot.variable gg_partial plot.gg_partial gg_variable plot.gg_variable

Examples

```
## Not run:
## ------------------------------------------------------------
## classification
## ------------------------------------------------------------
## -------- iris data

## iris "Petal.Width" partial dependence plot
##
```

```
# rfsrc_iris <- rfsrc(Species ~., data = iris)
# partial_iris <- plot.variable(rfsrc_iris, xvar.names = "Petal.Width",
#                               partial=TRUE)
data(partial_iris, package="ggRandomForests")

gg_dta <- gg_partial(partial_iris)
plot(gg_dta)

## ------------------------------------------------------------
## regression
## ------------------------------------------------------------
## -------- air quality data
## airquality "Wind" partial dependence plot
##
# rfsrc_airq <- rfsrc(Ozone ~ ., data = airquality)
# partial_airq <- plot.variable(rfsrc_airq, xvar.names = "Wind",
#                               partial=TRUE, show.plot=FALSE)
data(partial_airq, package="ggRandomForests")

gg_dta <- gg_partial(partial_airq)
plot(gg_dta)

gg_dta.m <- gg_dta[["Month"]]
plot(gg_dta.m, notch=TRUE)

gg_dta[["Month"]] <- NULL
plot(gg_dta, panel=TRUE)

## -------- Boston data
data(partial_Boston, package="ggRandomForests")

gg_dta <- gg_partial(partial_Boston)
plot(gg_dta)
plot(gg_dta, panel=TRUE)

## -------- mtcars data
data(partial_mtcars, package="ggRandomForests")

gg_dta <- gg_partial(partial_mtcars)

plot(gg_dta)

gg_dta.cat <- gg_dta
gg_dta.cat[["disp"]] <- gg_dta.cat[["wt"]] <- gg_dta.cat[["hp"]] <- NULL
gg_dta.cat[["drat"]] <- gg_dta.cat[["carb"]] <- gg_dta.cat[["qsec"]] <- NULL

plot(gg_dta.cat, panel=TRUE)

gg_dta[["cyl"]] <- gg_dta[["vs"]] <- gg_dta[["am"]] <- NULL
gg_dta[["gear"]] <- NULL
plot(gg_dta, panel=TRUE)

## ------------------------------------------------------------
```

```
## survival examples
## ------------------------------------------------------------
## -------- veteran data
## survival "age" partial variable dependence plot
##
# data(veteran, package = "randomForestSRC")
# rfsrc_veteran <- rfsrc(Surv(time,status)~., veteran, nsplit = 10, ntree = 100)
#
## 30 day partial plot for age
# partial_veteran <- plot.variable(rfsrc_veteran, surv.type = "surv",
#                                  partial = TRUE, time=30,
#                                  xvar.names = "age",
#                                  show.plots=FALSE)
data(partial_veteran, package="ggRandomForests")

gg_dta <- gg_partial(partial_veteran[[1]])
plot(gg_dta)

gg_dta.cat <- gg_dta
gg_dta[["celltype"]] <- gg_dta[["trt"]] <- gg_dta[["prior"]] <- NULL
plot(gg_dta, panel=TRUE)

gg_dta.cat[["karno"]] <- gg_dta.cat[["diagtime"]] <- gg_dta.cat[["age"]] <- NULL
plot(gg_dta.cat, panel=TRUE, notch=TRUE)

gg_dta <- lapply(partial_veteran, gg_partial)
length(gg_dta)
gg_dta <- combine.gg_partial(gg_dta[[1]], gg_dta[[2]] )

plot(gg_dta[["karno"]])
plot(gg_dta[["celltype"]])

gg_dta.cat <- gg_dta
gg_dta[["celltype"]] <- gg_dta[["trt"]] <- gg_dta[["prior"]] <- NULL
plot(gg_dta, panel=TRUE)

gg_dta.cat[["karno"]] <- gg_dta.cat[["diagtime"]] <- gg_dta.cat[["age"]] <- NULL
plot(gg_dta.cat, panel=TRUE, notch=TRUE)

## -------- pbc data

## End(Not run)
```

`plot.gg_rfsrc` *Predicted response plot from a* `gg_rfsrc` *object.*

Description

Plot the predicted response from a `gg_rfsrc` object, the `rfsrc` prediction, using the OOB prediction from the forest.

Usage

```
## S3 method for class 'gg_rfsrc'
plot(x, ...)
```

Arguments

x	gg_rfsrc object created from a rfsrc object
...	arguments passed to gg_rfsrc.

Value

ggplot object

References

Breiman L. (2001). Random forests, Machine Learning, 45:5-32.

Ishwaran H. and Kogalur U.B. (2007). Random survival forests for R, Rnews, 7(2):25-31.

Ishwaran H. and Kogalur U.B. (2013). Random Forests for Survival, Regression and Classification (RF-SRC), R package version 1.4.

See Also

gg_rfsrc rfsrc

Examples

```
## Not run:
## ------------------------------------------------------------
## classification example
## ------------------------------------------------------------
## -------- iris data
# rfsrc_iris <- rfsrc(Species ~ ., data = iris)
data(rfsrc_iris, package="ggRandomForests")
gg_dta<- gg_rfsrc(rfsrc_iris)

plot(gg_dta)

## ------------------------------------------------------------
## Regression example
## ------------------------------------------------------------
## -------- air quality data
# rfsrc_airq <- rfsrc(Ozone ~ ., data = airquality, na.action = "na.impute")
data(rfsrc_airq, package="ggRandomForests")
gg_dta<- gg_rfsrc(rfsrc_airq)

plot(gg_dta)

## -------- Boston data
data(rfsrc_Boston, package="ggRandomForests")
plot(rfsrc_Boston)
```

```
## -------- mtcars data
data(rfsrc_mtcars, package="ggRandomForests")
gg_dta<- gg_rfsrc(rfsrc_mtcars)

plot(gg_dta)

## ------------------------------------------------------------
## Survival example
## ------------------------------------------------------------
## -------- veteran data
## randomized trial of two treatment regimens for lung cancer
# data(veteran, package = "randomForestSRC")
# rfsrc_veteran <- rfsrc(Surv(time, status) ~ ., data = veteran, ntree = 100)
data(rfsrc_veteran, package = "ggRandomForests")
gg_dta <- gg_rfsrc(rfsrc_veteran)
plot(gg_dta)

gg_dta <- gg_rfsrc(rfsrc_veteran, conf.int=.95)
plot(gg_dta)

gg_dta <- gg_rfsrc(rfsrc_veteran, by="trt")
plot(gg_dta)

## -------- pbc data
data(rfsrc_pbc, package = "ggRandomForests")
gg_dta <- gg_rfsrc(rfsrc_pbc)
plot(gg_dta)

gg_dta <- gg_rfsrc(rfsrc_pbc, conf.int=.95)
plot(gg_dta)

gg_dta <- gg_rfsrc(rfsrc_pbc, by="treatment")
plot(gg_dta)



## End(Not run)
```

`plot.gg_roc` *ROC plot generic function for a* `gg_roc` *object.*

Description

ROC plot generic function for a `gg_roc` object.

Usage

```
## S3 method for class 'gg_roc'
plot(x, which.outcome = NULL, ...)
```

Arguments

x	gg_roc object created from a classification forest
which.outcome	for multiclass problems, choose the class for plotting
...	arguments passed to the gg_roc function

Value

ggplot object of the ROC curve

References

Breiman L. (2001). Random forests, Machine Learning, 45:5-32.

Ishwaran H. and Kogalur U.B. (2007). Random survival forests for R, Rnews, 7(2):25-31.

Ishwaran H. and Kogalur U.B. (2013). Random Forests for Survival, Regression and Classification (RF-SRC), R package version 1.4.

See Also

gg_roc rfsrc

Examples

```
## Not run:
## ------------------------------------------------------------
## classification example
## ------------------------------------------------------------
## -------- iris data
#rfsrc_iris <- rfsrc(Species ~ ., data = iris)
data(rfsrc_iris, package="ggRandomForests")

# ROC for setosa
gg_dta <- gg_roc(rfsrc_iris, which.outcome=1)
plot.gg_roc(gg_dta)

# ROC for versicolor
gg_dta <- gg_roc(rfsrc_iris, which.outcome=2)
plot.gg_roc(gg_dta)

# ROC for virginica
gg_dta <- gg_roc(rfsrc_iris, which.outcome=3)
plot.gg_roc(gg_dta)

# Alternatively, you can plot all three outcomes in one go
# by calling the plot function on the forest object.
plot.gg_roc(rfsrc_iris)


## End(Not run)
```

plot.gg_survival *Plot a* gg_survival *object.*

Description

Plot a gg_survival object.

Usage

```
## S3 method for class 'gg_survival'
plot(x, type = c("surv", "cum_haz", "hazard", "density",
  "mid_int", "life", "proplife"), error = c("shade", "bars", "lines", "none"),
  ...)
```

Arguments

x	gg_survival or a survival gg_rfsrc object created from a rfsrc object
type	"surv", "cum_haz","hazard","density","mid_int", "life","proplife"
error	"shade", "bars", "lines" or "none"
...	not used

Value

ggplot object

Examples

```
## Not run:
## -------- pbc data
data(pbc, package="randomForestSRC")
pbc$time <- pbc$days/364.25

# This is the same as kaplan
gg_dta <- gg_survival(interval="time", censor="status",
                      data=pbc)

plot(gg_dta, error="none")
plot(gg_dta)

# Stratified on treatment variable.
gg_dta <- gg_survival(interval="time", censor="status",
                      data=pbc, by="treatment")

plot(gg_dta, error="none")
plot(gg_dta)

# ...with smaller confidence limits.
gg_dta <- gg_survival(interval="time", censor="status",
```

```
                    data=pbc, by="treatment", conf.int=.68)
plot(gg_dta, error="lines")


## End(Not run)
```

---

plot.gg_variable *Plot a* gg_variable *object,*

Description

Plot a gg_variable object,

Usage

```
## S3 method for class 'gg_variable'
plot(x, xvar, time, time_labels, panel = FALSE,
  oob = TRUE, points = TRUE, smooth = TRUE, ...)
```

Arguments

x	gg_variable object created from a rfsrc object
xvar	variable (or list of variables) of interest.
time	For survival, one or more times of interest
time_labels	string labels for times
panel	Should plots be facetted along multiple xvar?
oob	oob estimates (boolean)
points	plot the raw data points (boolean)
smooth	include a smooth curve (boolean)
...	arguments passed to the ggplot2 functions.

Value

A single ggplot object, or list of ggplot objects

References

Breiman L. (2001). Random forests, Machine Learning, 45:5-32.

Ishwaran H. and Kogalur U.B. (2007). Random survival forests for R, Rnews, 7(2):25-31.

Ishwaran H. and Kogalur U.B. (2013). Random Forests for Survival, Regression and Classification (RF-SRC), R package version 1.4.

Examples

```
## Not run:
## ------------------------------------------------------------
## classification
## ------------------------------------------------------------
## -------- iris data
## iris
#rfsrc_iris <- rfsrc(Species ~., data = iris)
data(rfsrc_iris, package="ggRandomForests")

gg_dta <- gg_variable(rfsrc_iris)
plot(gg_dta, xvar="Sepal.Width")
plot(gg_dta, xvar="Sepal.Length")

## !! TODO !! this needs to be corrected
plot(gg_dta, xvar=rfsrc_iris$xvar.names,
     panel=TRUE, se=FALSE)

## ------------------------------------------------------------
## regression
## ------------------------------------------------------------
## -------- air quality data
#rfsrc_airq <- rfsrc(Ozone ~ ., data = airquality)
data(rfsrc_airq, package="ggRandomForests")
gg_dta <- gg_variable(rfsrc_airq)

# an ordinal variable
gg_dta[,"Month"] <- factor(gg_dta[,"Month"])

plot(gg_dta, xvar="Wind")
plot(gg_dta, xvar="Temp")
plot(gg_dta, xvar="Solar.R")

plot(gg_dta, xvar=c("Solar.R", "Wind", "Temp", "Day"), panel=TRUE)

plot(gg_dta, xvar="Month", notch=TRUE)

## -------- motor trend cars data
#rfsrc_mtcars <- rfsrc(mpg ~ ., data = mtcars)
data(rfsrc_mtcars, package="ggRandomForests")
gg_dta <- gg_variable(rfsrc_mtcars)

# mtcars$cyl is an ordinal variable
gg_dta$cyl <- factor(gg_dta$cyl)
gg_dta$am <- factor(gg_dta$am)
gg_dta$vs <- factor(gg_dta$vs)
gg_dta$gear <- factor(gg_dta$gear)
gg_dta$carb <- factor(gg_dta$carb)

plot(gg_dta, xvar="cyl")

# Others are continuous
```

```
plot(gg_dta, xvar="disp")
plot(gg_dta, xvar="hp")
plot(gg_dta, xvar="wt")

# panel
plot(gg_dta,xvar=c("disp","hp", "drat", "wt", "qsec"),  panel=TRUE)
plot(gg_dta, xvar=c("cyl", "vs", "am", "gear", "carb") ,panel=TRUE)

## -------- Boston data

## ------------------------------------------------------------
## survival examples
## ------------------------------------------------------------
## -------- veteran data
## survival
data(veteran, package = "randomForestSRC")
rfsrc_veteran <- rfsrc(Surv(time,status)~., veteran, nsplit = 10, ntree = 100)

# get the 1 year survival time.
gg_dta <- gg_variable(rfsrc_veteran, time=90)

# Generate variable dependance plots for age and diagtime
plot(gg_dta, xvar = "age")
plot(gg_dta, xvar = "diagtime")

# Generate coplots
plot(gg_dta, xvar = c("age", "diagtime"), panel=TRUE)

# If we want to compare survival at different time points, say 30, 90 day
# and 1 year
gg_dta <- gg_variable(rfsrc_veteran, time=c(30, 90, 365))

# Generate variable dependance plots for age and diagtime
plot(gg_dta, xvar = "age")
plot(gg_dta, xvar = "diagtime")

# Generate coplots
plot(gg_dta, xvar =  c("age", "diagtime"), panel=TRUE)

## -------- pbc data

## End(Not run)
```

`plot.gg_vimp` *Plot a* `gg_vimp` *object, extracted variable importance of a* `rfsrc` *object*

Description

Plot a `gg_vimp` object, extracted variable importance of a `rfsrc` object

Usage

```
## S3 method for class 'gg_vimp'
plot(x, relative, lbls, ...)
```

Arguments

x	gg_vimp object created from a rfsrc object
relative	should we plot vimp or relative vimp. Defaults to vimp.
lbls	A vector of alternative variable labels. Item names should be the same as the variable names.
...	optional arguments passed to gg_vimp if necessary

Value

ggplot object

References

Breiman L. (2001). Random forests, Machine Learning, 45:5-32.

Ishwaran H. and Kogalur U.B. (2007). Random survival forests for R, Rnews, 7(2):25-31.

Ishwaran H. and Kogalur U.B. (2013). Random Forests for Survival, Regression and Classification (RF-SRC), R package version 1.4.

See Also

gg_vimp

Examples

```
## Not run:
## ------------------------------------------------------------
## classification example
## ------------------------------------------------------------
## -------- iris data
# rfsrc_iris <- rfsrc(Species ~ ., data = iris)
data(rfsrc_iris, package="ggRandomForests")
gg_dta <- gg_vimp(rfsrc_iris)
plot(gg_dta)

## ------------------------------------------------------------
## regression example
## ------------------------------------------------------------
## -------- air quality data
# rfsrc_airq <- rfsrc(Ozone ~ ., airquality)
data(rfsrc_airq, package="ggRandomForests")
gg_dta <- gg_vimp(rfsrc_airq)
plot(gg_dta)

## -------- Boston data
```

```
data(rfsrc_Boston, package="ggRandomForests")
gg_dta <- gg_vimp(rfsrc_Boston)
plot(gg_dta)

## -------- mtcars data
data(rfsrc_mtcars, package="ggRandomForests")
gg_dta <- gg_vimp(rfsrc_mtcars)
plot(gg_dta)

## ------------------------------------------------------------
## survival example
## ------------------------------------------------------------
## -------- veteran data
data(rfsrc_veteran, package="ggRandomForests")
gg_dta <- gg_vimp(rfsrc_veteran)
plot(gg_dta)

## -------- pbc data
data(rfsrc_pbc, package="ggRandomForests")
gg_dta <- gg_vimp(rfsrc_pbc)
plot(gg_dta)


## End(Not run)
```

`print.gg_minimal_depth` *Print a* `gg_minimal_depth` *object.*

Description

Print a `gg_minimal_depth` object.

Usage

```
## S3 method for class 'gg_minimal_depth'
print(x, ...)
```

Arguments

x	a `gg_minimal_depth` object.
...	optional arguments

Examples

```
## ------------------------------------------------------------
## classification example
## ------------------------------------------------------------
```

```
## Not run:
## You can build a randomForest
# rfsrc_iris <- rfsrc(Species ~ ., data = iris)
# varsel_iris <- var.select(rfsrc_iris)
# ... or load a cached randomForestSRC object
data(varsel_iris, package="ggRandomForests")

# Get a data.frame containing minimaldepth measures
gg_dta <- gg_minimal_depth(varsel_iris)
print(gg_dta)

## End(Not run)
## ------------------------------------------------------------
## regression example
## ------------------------------------------------------------
## Not run:
# ... or load a cached randomForestSRC object
data(varsel_airq, package="ggRandomForests")

# Get a data.frame containing minimaldepth measures
gg_dta<- gg_minimal_depth(varsel_airq)
print(gg_dta)

# To nicely print a rfsrc::var.select output...
print(varsel_airq)

## End(Not run)
## Not run:
# ... or load a cached randomForestSRC object
data(varsel_Boston, package="ggRandomForests")

# Get a data.frame containing minimaldepth measures
gg_dta<- gg_minimal_depth(varsel_Boston)
print(gg_dta)

# To nicely print a rfsrc::var.select output...
print(varsel_Boston)

## End(Not run)
```

`quantile_pts` *Find points evenly distributed along the vectors values.*

Description

This function finds point values from a vector argument to produce `groups` intervals. Setting `groups=2` will return three values, the two end points, and one mid point (at the median value of the vector).

The output can be passed directly into the breaks argument of the `cut` function for creating groups for coplots.

Usage

```
quantile_pts(object, groups, intervals = FALSE)
```

Arguments

object	vector object of values.
groups	how many points do we want
intervals	should we return the raw points or intervals to be passed to the cut function

Value

vector of groups+1 cut point values.

See Also

cut gg_partial_coplot

Examples

```
## Not run:
data(rfsrc_Boston)

# To create 6 intervals, we want 7 points.
# quantile_pts will find balanced intervals
rm_pts <- quantile_pts(rfsrc_Boston$xvar$rm, groups=6, intervals=TRUE)

# Use cut to create the intervals
rm_grp <- cut(rfsrc_Boston$xvar$rm, breaks=rm_pts)

summary(rm_grp)

## End(Not run)
```

---

shift *lead function to shift by one (or more).*

Description

lead function to shift by one (or more).

Usage

```
shift(x, shift_by = 1)
```

Arguments

x	a vector of values
shift_by	an integer of length 1, giving the number of positions to lead (positive) or lag (negative) by

Details

Lead and lag are useful for comparing values offset by a constant (e.g. the previous or next value)

Taken from: http://ctszkin.com/2012/03/11/generating-a-laglead-variables/

This function allows me to remove the dplyr::lead depends. Still suggest for vignettes though.

Examples

```
d<-data.frame(x=1:15)
#generate lead variable
d$df_lead2<-ggRandomForests:::shift(d$x,2)
#generate lag variable
d$df_lag2<-ggRandomForests:::shift(d$x,-2)
```

---

surface_matrix *Construct a set of (x, y, z) matrices for surface plotting a* gg_partial_coplot *object*

Description

Construct a set of (x, y, z) matrices for surface plotting a gg_partial_coplot object

Usage

```
surface_matrix(dta, xvar)
```

Arguments

dta	a gg_partial_coplot object containing at least 3 numeric columns of data
xvar	a vector of 3 column names from the data object, in (x, y, z) order

Details

To create a surface plot, the plot3D::surf3D function expects 3 matrices of n.x by n.y. Take the p+1 by n gg_partial_coplot object, and extract and construct the x, y and z matrices from the provided xvar column names.

Examples

```
## Not run:
## From vignette(randomForestRegression, package="ggRandomForests")
##
data(rfsrc_Boston)
rm_pts <- quantile_pts(rfsrc_Boston$xvar$rm, groups=49, intervals=TRUE)

# Load the stored partial coplot data.
data(partial_Boston_surf)

# Instead of groups, we want the raw rm point values,
```

```
# To make the dimensions match, we need to repeat the values
# for each of the 50 points in the lstat direction
rm.tmp <- do.call(c,lapply(rm_pts,
                           function(grp){rep(grp, length(partial_Boston_surf))}))

# Convert the list of plot.variable output to
partial_surf <- do.call(rbind,lapply(partial_Boston_surf, gg_partial))

# attach the data to the gg_partial_coplot
partial_surf$rm <- rm.tmp

# Transform the gg_partial_coplot object into a list of three named matrices
# for surface plotting with plot3D::surf3D
srf <- surface_matrix(partial_surf, c("lstat", "rm", "yhat"))

## End(Not run)

## Not run:
# surf3D is in the plot3D package.
library(plot3D)
# Generate the figure.
surf3D(x=srf$x, y=srf$y, z=srf$z, col=topo.colors(10),
       colkey=FALSE, border = "black", bty="b2",
       shade = 0.5, expand = 0.5,
       lighting = TRUE, lphi = -50,
       xlab="Lower Status", ylab="Average Rooms", zlab="Median Value"
)

## End(Not run)
```

Index