Package 'WVPlots'

April 3, 2018

Type Package

Title Common Plots for Analysis

Version 0.3.1

Date 2018-04-03

URL https://github.com/WinVector/WVPlots,
https://winvector.github.io/WVPlots/

Maintainer John Mount <jmount@win-vector.com>

BugReports https://github.com/WinVector/WVPlots/issues

Description Select data analysis plots, under a standardized calling interface implemented on top of 'ggplot2' and 'plotly'.
Plots of interest include: 'ROC', gain curve, scatter plot with marginal distributions,
conditioned scatter plot with marginal densities,
box and stem with matching theoretical distribution, and density with matching theoretical distribution.

License GPL-3

VignetteBuilder knitr

Imports ggplot2 (>= 2.2.0), wrapr (>= 1.3.0), sigr (>= 0.2.4), utils,
grid, gridExtra, graphics, mgcv

Suggests knitr, testthat, cdata (>= 0.6.0), RSQLite, plotly

RoxygenNote 6.0.1

ByteCompile true

NeedsCompilation no

Author John Mount [aut, cre],
Nina Zumel [aut],
Win-Vector LLC [cph]

Repository CRAN

Date/Publication 2018-04-03 21:06:32 UTC

R topics documented:

`BinaryYScatterPlot` *Plot a scatter plot of a binary variable. xvar is the continuous independent variable and yvar is the dependent binary variable*

Description

Plot a scatter plot of a binary variable. xvar is the continuous independent variable and yvar is the dependent binary variable

Usage

```
BinaryYScatterPlot(frame, xvar, yvar, title, ..., se = FALSE,
  use_glm = TRUE)
```

Arguments

frame	data frame to get values from
xvar	name of the independent column in frame
yvar	name of the dependent (output or result to be modeled) column in frame
title	title to place on plot
...	no unnamed argument, added to force named binding of later arguments.
se	if TRUE, add error bars (defaults to FALSE). Ignored if useGLM is TRUE
use_glm	if TRUE, "smooths" with a one-variable logistic regression (defaults to TRUE)

Examples

```
set.seed(34903490)
x = rnorm(50)
y = 0.5*x^2 + 2*x + rnorm(length(x))
frm = data.frame(x=x,y=y,yC=y>=as.numeric(quantile(y,probs=0.8)))
frm$absY <- abs(frm$y)
frm$posY = frm$y > 0
frm$costX = 1
WVPlots::BinaryYScatterPlot(frm, "x", "posY",
   title="Example 'Probability of Y' Plot")
```

---

calcPR *calculate precision/recall curve.*

Description

Based on: http://blog.revolutionanalytics.com/2016/08/roc-curves-in-two-lines-of-code.html

Usage

```
calcPR(modelPredictions, yValues)
```

Arguments

modelPredictions	numeric predictions (not empty)
yValues	logical truth (not empty, same lenght as model predictions)

Details

See also https://github.com/WinVector/sigr

Value

line graph, point graph, and summaries

`ClevelandDotPlot` *Plot a Cleveland dot plot.*

Description

assumes that xvar is a factor variable sort < 0 sorts the factor levels in decreasing order (most frequent level first) sort > 0 sorts the factor levels in increasing order (good when used in conjunction with coord_flip()) sort = 0 leaves the factor levels in "natural order" – usually alphabetical stem = FALSE will plot only the dots, without the stem to the y=0 line. limit_n = NULL plots all the levels, N an integer limits to the top N most populous levels

Usage

```
ClevelandDotPlot(frm, xvar, title, ..., sort = -1, limit_n = NULL,
  stem = TRUE)
```

Arguments

`frm`	data frame to get values from
`xvar`	name of the independent (input or model) column in frame
`title`	title to place on plot
`...`	no unnamed argument, added to force named binding of later arguments.
`sort`	if TRUE sort data
`limit_n`	if not NULL number of items to plot
`stem`	if TRUE add stems/whiskers to plot

Examples

```
set.seed(34903490)
# discrete variable: letters of the alphabet
# frequencies of letters in English
# source: http://en.algoritmy.net/article/40379/Letter-frequency-English
letterFreqs = c(8.167, 1.492, 2.782, 4.253, 12.702, 2.228,
                2.015, 6.094, 6.966, 0.153, 0.772, 4.025, 2.406, 6.749, 7.507, 1.929,
                0.095, 5.987, 6.327, 9.056, 2.758, 0.978, 2.360, 0.150, 1.974, 0.074)
letterFreqs = letterFreqs/100
letterFrame = data.frame(letter = letters, freq=letterFreqs)
# now let's generate letters according to their letter frequencies
N = 1000
randomDraws = data.frame(draw=1:N,
  letter=sample(letterFrame$letter, size=N,
  replace=TRUE, prob=letterFrame$freq))
WVPlots::ClevelandDotPlot(randomDraws, "letter",
  title = "Example Cleveland-style dot plot")
```

ConditionalSmoothedScatterPlot

Plot a scatter plot with smoothing line, with smoothing window aligned either left, center or right, xvar is the continuous independent variable and yvar is the dependent binary variable. Smoothing is by a square window of width k

Description

Plot a scatter plot with smoothing line, with smoothing window aligned either left, center or right, xvar is the continuous independent variable and yvar is the dependent binary variable. Smoothing is by a square window of width k

Usage

```
ConditionalSmoothedScatterPlot(frame, xvar, yvar, groupvar, title, ..., k = 3,
  align = "center")
```

Arguments

frame	data frame to get values from
xvar	name of the independent column in frame. Assumed to be regularly spaced
yvar	name of the dependent (output or result to be modeled) column in frame
groupvar	name of the grouping column in frame. Can be NULL for an unconditional plot
title	title for plot
...	no unnamed argument, added to force named binding of later arguments.
k	width of smoothing window. Must be odd for a center-aligned plot. Defaults to 3
align	smoothing window alignment: 'center', 'left', or 'right'. Defaults to 'center'

Examples

```
y = c(1,2,3,4,5,10,15,18,20,25)
x = seq_len(length(y))
df = data.frame(x=x,y=y)
WVPlots::ConditionalSmoothedScatterPlot(df, "x", "y", NULL,
   title="left smooth, one group", align="left")
```

DiscreteDistribution *Plot distribution of a single continuous variable.*

Description

Plot distribution of a single continuous variable.

Usage

```
DiscreteDistribution(frm, xvar, title, ..., stem = TRUE)
```

Arguments

frm	data frame to get values from
xvar	name of the independent (input or model) column in frame
title	title to place on plot
...	no unnamed argument, added to force named binding of later arguments.
stem	if TRUE add whisker/stems to plot

Examples

```
frmx = data.frame(x = rbinom(1000, 20, 0.5))
WVPlots::DiscreteDistribution(frmx, "x","Discrete example")
```

DoubleDensityPlot *Plot two density plots conditioned on truthVar.*

Description

Plot two density plots conditioned on truthVar.

Usage

```
DoubleDensityPlot(frame, xvar, truthVar, title, ...)
```

Arguments

frame	data frame to get values from
xvar	name of the independent (input or model) column in frame
truthVar	name of the dependent (output or result to be modeled) column in frame
title	title to place on plot
...	no unnamed argument, added to force named binding of later arguments.

Examples

```
set.seed(34903490)
x = rnorm(50)
y = 0.5*x^2 + 2*x + rnorm(length(x))
frm = data.frame(score=x,
   truth=(y>=as.numeric(quantile(y,probs=0.8))),
   stuck=TRUE,
   rare=FALSE)
frm[1,'rare'] = TRUE
WVPlots::DoubleDensityPlot(frm, "score", "truth", title="Example double density plot")
WVPlots::DoubleDensityPlot(frm, "score", "stuck", title="Example double density plot")
WVPlots::DoubleDensityPlot(frm, "score", "rare", title="Example double density plot")
```

`DoubleHistogramPlot` *Plot two histograms conditioned on truthVar.*

Description

Plot two histograms conditioned on truthVar.

Usage

```
DoubleHistogramPlot(frame, xvar, truthVar, title, ..., breaks = 40)
```

Arguments

`frame`	data frame to get values from
`xvar`	name of the independent (input or model) column in frame
`truthVar`	name of the dependent (output or result to be modeled) column in frame
`title`	title to place on plot
`...`	no unnamed argument, added to force named binding of later arguments.
`breaks`	breaks to pass to histogram

Examples

```
set.seed(34903490)
x = rnorm(50)
y = 0.5*x^2 + 2*x + rnorm(length(x))
frm = data.frame(x=x,y=y,yC=y>=as.numeric(quantile(y,probs=0.8)))
frm$absY <- abs(frm$y)
frm$posY = frm$y > 0
frm$costX = 1
WVPlots::DoubleHistogramPlot(frm, "x", "yC", title="Example double histogram plot")
```

GainCurvePlot *Plot the gain curve of a sort-order.*

Description

Plot the gain curve of a sort-order.

Usage

```
GainCurvePlot(frame, xvar, truthVar, title, ..., compute_sig = TRUE,
  large_count = 1000)
```

Arguments

frame	data frame to get values from
xvar	name of the independent (input or model) column in frame
truthVar	name of the dependent (output or result to be modeled) column in frame
title	title to place on plot
...	no unnamed argument, added to force named binding of later arguments.
compute_sig	logical, if TRUE compute significance.
large_count	numeric, number of plotting points to consider large (and cut down).

Examples

```
set.seed(34903490)
y = abs(rnorm(20)) + 0.1
x = abs(y + 0.5*rnorm(20))
frm = data.frame(model=x, value=y)
frm$costs=1
frm$costs[1]=5
frm$rate = with(frm, value/costs)
frm$isValuable = (frm$value >= as.numeric(quantile(frm$value, probs=0.8)))
WVPlots::GainCurvePlot(frm, "model", "value",
   title="Example Continuous Gain Curve")
```

GainCurvePlotC *Plot the gain curve of a sort-order with costs.*

Description

Plot the gain curve of a sort-order with costs.

Usage

```
GainCurvePlotC(frame, xvar, costVar, truthVar, title, ..., compute_sig = TRUE,
  large_count = 1000)
```

Arguments

frame	data frame to get values from
xvar	name of the independent (input or model) column in frame
costVar	cost of each item (drives x-axis sum)
truthVar	name of the dependent (output or result to be modeled) column in frame
title	title to place on plot
...	no unnamed argument, added to force named binding of later arguments.
compute_sig	logical, if TRUE compute significance
large_count	numeric, number of plotting points to consider large (and cut down).

Examples

```
set.seed(34903490)
y = abs(rnorm(20)) + 0.1
x = abs(y + 0.5*rnorm(20))
frm = data.frame(model=x, value=y)
frm$costs=1
frm$costs[1]=5
frm$rate = with(frm, value/costs)
frm$isValuable = (frm$value >= as.numeric(quantile(frm$value, probs=0.8)))
WVPlots::GainCurvePlotC(frm, "model", "costs", "value",
   title="Example Continuous Gain CurveC")
```

GainCurvePlotWithNotation *Take the standard WVPlots gain curve and add extra notation*

Description

Take the standard WVPlots gain curve and add extra notation

Usage

```
GainCurvePlotWithNotation(frame, xvar, truthVar, title, gainx, labelfun, ...,
  compute_sig = TRUE, large_count = 1000)
```

Arguments

frame	data frame to get values from
xvar	name of the independent (input or model) column in frame
truthVar	name of the dependent (output or result to be modeled) column in frame
title	title to place on plot
gainx	the point on the x axis corresponding to the desired label
labelfun	a function to return a label for the marked point
...	no unarmed argument, added to force named binding of later arguments.
compute_sig	logical, if TRUE compute significance
large_count	numeric, number of plotting points to consider large (and cut down).

Examples

```
set.seed(34903490)
y = abs(rnorm(20)) + 0.1
x = abs(y + 0.5*rnorm(20))
frm = data.frame(model=x, value=y)
frm$costs=1
frm$costs[1]=5
frm$rate = with(frm, value/costs)
frm$isValuable = (frm$value >= as.numeric(quantile(frm$value, probs=0.8)))
gainx = 0.10  # get the top 10% most valuable points as sorted by the model
# make a function to calculate the label for the annotated point
labelfun = function(gx, gy) {
  pctx = gx*100
  pcty = gy*100

  paste("The top ", pctx, "% most valuable points by the model\n",
        "are ", pcty, "% of total actual value", sep='')
}
WVPlots::GainCurvePlotWithNotation(frm, "model", "value",
```

```
  title="Example Gain Curve with annotation",
  gainx=gainx,labelfun=labelfun)
```

---

graphROC *calculate AUC.*

Description

Based on: http://blog.revolutionanalytics.com/2016/08/roc-curves-in-two-lines-of-code.html

Usage

```
graphROC(modelPredictions, yValues)
```

Arguments

modelPredictions	numeric predictions (not empty)
yValues	logical truth (not empty, same lenght as model predictions)

Details

See also https://github.com/WinVector/sigr

Value

line graph, point graph, and area under curve

Examples

```
set.seed(34903490)
x = rnorm(50)
y = 0.5*x^2 + 2*x + rnorm(length(x))
frm = data.frame(x=x,yC=y>=as.numeric(quantile(y,probs=0.8)))
WVPlots::graphROC(frm$x, frm$yC)
```

LogLogPlot *Plot a trend on log-log paper.*

Description

This plot is intended for ploting functions that are observed costs or durations as a function of problem size. In this case we expect the ideal or expected cost function to be non-decreasing. Any negative trends are assumed to arise from the noise model. The graph is specialized to compare non-decreasing linear and non-decreasing quadratic growth.

Usage

```
LogLogPlot(frame, xvar, yvar, title, ..., use_coord_trans = FALSE)
```

Arguments

`frame`	data frame to get values from
`xvar`	name of the independent (input or model) column in frame
`yvar`	name of the dependent (output or result to be modeled) column in frame
`title`	title to place on plot
`...`	no unnamed argument, added to force named binding of later arguments.
`use_coord_trans`	logical if TRUE, use coord_trans instead of `coord_trans(x = "log10", y = "log10")` instead of `scale_x_log10() + scale_yu_log10()` (useful when there is not enough range to show ticks).

Details

Some care must be taken in drawing conclusions from log-log plots, as the transform is fairly violent. Please see: "(Mar's Law) Everything is linear if plotted log-log with a fat magic marker" (from Akin's Laws of Spacecraft Design http://spacecraft.ssl.umd.edu/akins_laws.html), and "So You Think You Have a Power Law" http://bactra.org/weblog/491.html.

Examples

```
set.seed(5326)
frm = data.frame(x = 1:20)
frm$y <- 5 + frm$x + 0.2 * frm$x * frm$x + 0.1*abs(rnorm(nrow(frm)))
WVPlots::LogLogPlot(frm, "x", "y", title="Example Trend")
```

PlotDistCountNormal *plot distribution details as a histogram plus matching normal*

Description

assumes that xvar is a factor variable sort < 0 sorts the factor levels in decreasing order (most frequent level first) sort > 0 sorts the factor levels in increasing order (good when used in conjunction with coord_flip()) sort = 0 leaves the factor levels in "natural order" – usually alphabetical stem = FALSE will plot only the dots, without the stem to the y=0 line. limit_n = NULL plots all the levels, N an integer limits to the top N most populous levels

Usage

```
PlotDistCountNormal(frm, xvar, title, ..., binWidth = c())
```

Arguments

frm	data frame to get values from
xvar	name of the independent (input or model) column in frame
title	title to place on plot
...	no unarmed argument, added to force named binding of later arguments.
binWidth	with of histogram bins

Examples

```
set.seed(52523)
d <- data.frame(wt=100*rnorm(100))
PlotDistCountNormal(d,'wt','example')
```

PlotDistDensityBeta *plot distribution details as a density plus matching beta*

Description

assumes that xvar is a factor variable sort < 0 sorts the factor levels in decreasing order (most frequent level first) sort > 0 sorts the factor levels in increasing order (good when used in conjunction with coord_flip()) sort = 0 leaves the factor levels in "natural order" – usually alphabetical stem = FALSE will plot only the dots, without the stem to the y=0 line. limit_n = NULL plots all the levels, N an integer limits to the top N most populous levels

Usage

```
PlotDistDensityBeta(frm, xvar, title, ...)
```

Arguments

frm	data frame to get values from
xvar	name of the independent (input or model) column in frame
title	title to place on plot
...	no unnamed argument, added to force named binding of later arguments.

Examples

```
set.seed(52523)
d <- data.frame(wt=rbeta(100,shape1=1,shape2=0.5))
PlotDistDensityBeta(d,'wt','example')
```

PlotDistDensityNormal *plot distribution details as a density plus matching normal*

Description

assumes that xvar is a factor variable sort < 0 sorts the factor levels in decreasing order (most frequent level first) sort > 0 sorts the factor levels in increasing order (good when used in conjunction with coord_flip()) sort = 0 leaves the factor levels in "natural order" – usually alphabetical stem = FALSE will plot only the dots, without the stem to the y=0 line. limit_n = NULL plots all the levels, N an integer limits to the top N most populous levels

Usage

```
PlotDistDensityNormal(frm, xvar, title, ...)
```

Arguments

frm	data frame to get values from
xvar	name of the independent (input or model) column in frame
title	title to place on plot
...	no unnamed argument, added to force named binding of later arguments.

Examples

```
set.seed(52523)
d <- data.frame(wt=100*rnorm(100))
PlotDistDensityNormal(d,'wt','example')
```

PlotDistHistBeta *plot distribution details as a density plus matching beta*

Description

assumes that xvar is a factor variable sort < 0 sorts the factor levels in decreasing order (most frequent level first) sort > 0 sorts the factor levels in increasing order (good when used in conjunction with coord_flip()) sort = 0 leaves the factor levels in "natural order" – usually alphabetical stem = FALSE will plot only the dots, without the stem to the y=0 line. limit_n = NULL plots all the levels, N an integer limits to the top N most populous levels

Usage

```
PlotDistHistBeta(frm, xvar, title, ...)
```

Arguments

frm	data frame to get values from
xvar	name of the independent (input or model) column in frame
title	title to place on plot
...	no unnamed argument, added to force named binding of later arguments.

Examples

```
set.seed(52523)
d <- data.frame(wt=rbeta(100,shape1=0.5,shape2=0.5))
PlotDistHistBeta(d,'wt','example')
```

plotlyROC *Use* plotly *to produce a ROC plot.*

Description

Use plotly to produce a ROC plot.

Usage

```
plotlyROC(d, predCol, outcomeCol, outcomeTarget, title, ...)
```

Arguments

d	dataframe
predCol	name of column with numeric predictions
outcomeCol	name of column with truth
outcomeTarget	value considred true
title	character title for plot
...	no unnamed argument, added to force named binding of later arguments.

Value

plotly plot

Examples

```
d <- data.frame(x= 1:5, y= c(TRUE, FALSE , TRUE, TRUE, TRUE))
plotlyROC(d, 'x', 'y', TRUE, 'example plot')
```

---

plot_fit_trajectory *Plot the trajectory of a model fit.*

Description

Plot a history of fit over a trajetory of times. Example below gives a fit plot for a history report from Keras R package. Please see http://www.win-vector.com/blog/2017/12/plotting-deep-learning-model-performa for some examples and details.

Usage

```
plot_fit_trajectory(d, column_description, title, ..., epoch_name = "epoch",
  needs_flip = c(), pick_metric = NULL, discount_rate = NULL,
  draw_ribbon = FALSE, draw_segments = FALSE)
```

Arguments

d	data frame to get values from.
column_description	description of column measures (data.frame with columns measure, validation, and training).
title	character title for plot.
...	force later arguments to be bound by name
epoch_name	name for epoch or trajectory column.

needs_flip	charater array of measures that need to be flipped.
pick_metric	character metric to maximize.
discount_rate	numeric what fraction of over-fit to subtract from validation performance.
draw_ribbon	present the difference in training and validation performance as a ribbon rather than two curves? (default FALSE)
draw_segments	logical if TRUE draw over-fit/under-fit segments.

Value

ggplot2 plot

See Also

plot_Keras_fit_trajectory

Examples

```
d <- data.frame(
  epoch    = c(1,         2,         3,         4,         5),
  val_loss = c(0.3769818, 0.2996994, 0.2963943, 0.2779052, 0.2842501),
  val_acc  = c(0.8722000, 0.8895000, 0.8822000, 0.8899000, 0.8861000),
  loss     = c(0.5067290, 0.3002033, 0.2165675, 0.1738829, 0.1410933),
  acc      = c(0.7852000, 0.9040000, 0.9303333, 0.9428000, 0.9545333) )

cT <- data.frame(
  measure =    c("minus binary cross entropy", "accuracy"),
  training =   c("loss",                       "acc"),
  validation = c("val_loss",                   "val_acc"),
  stringsAsFactors = FALSE)

plt <- plot_fit_trajectory(
  d,
  column_description = cT,
  needs_flip = "minus binary cross entropy",
  title = "model performance by epoch, dataset, and measure",
  epoch_name = "epoch",
  pick_metric = "minus binary cross entropy",
  discount_rate = 0.1)

suppressWarnings(print(plt)) # too few points for loess
```

---

plot_Keras_fit_trajectory

Plot the trajectory of a Keras model fit.

Description

Plot a history of fit over a trajetory of times. Example below gives a fit plot for a history report from Keras R package. Please see http://winvector.github.io/FluidData/PlotExample/KerasPerfPlot.html for some details.

Usage

```
plot_Keras_fit_trajectory(d, title, ..., epoch_name = "epoch",
  lossname = "loss", loss_pretty_name = "minus binary cross entropy",
  perfname = "acc", perf_pretty_name = "accuracy",
  pick_metric = loss_pretty_name, fliploss = TRUE, discount_rate = NULL,
  draw_ribbon = FALSE)
```

Arguments

d	data frame to get values from.
title	character title for plot.
...	force later arguments to be bound by name
epoch_name	name for epoch or trajectory column.
lossname	name of training loss column (default 'loss')
loss_pretty_name	name for loss on graph (default 'minus binary cross entropy')
perfname	name of training performance column (default 'acc')
perf_pretty_name	name for performance metric on graph (default 'accuracy')
pick_metric	character: metric to maximize (NULL for no pick line - default loss_pretty_name)
fliploss	flip the loss so that "larger is better"? (default TRUE)
discount_rate	numeric: what fraction of over-fit to subtract from validation performance.
draw_ribbon	present the difference in training and validation performance as a ribbon rather than two curves? (default FALSE)

Details

Assumes a performance matrix that carries information for both training and validation loss, and an additional training and validation performance metric, in the format that a Keras history object returns.

By default, flips the loss so that better performance is larger for both the loss and the performance metric, and then draws a vertical line at the minumum validation loss (maximum flipped validation loss). If you choose not to flip the loss, you should not use the loss as the pick_metric.

Value

ggplot2 plot

See Also

plot_fit_trajectory

Examples

```
# example data (from Keras)
d <- data.frame(
  val_loss = c(0.3769818, 0.2996994, 0.2963943, 0.2779052, 0.2842501),
  val_acc  = c(0.8722000, 0.8895000, 0.8822000, 0.8899000, 0.8861000),
  loss     = c(0.5067290, 0.3002033, 0.2165675, 0.1738829, 0.1410933),
  acc      = c(0.7852000, 0.9040000, 0.9303333, 0.9428000, 0.9545333) )

plt <- plot_Keras_fit_trajectory(
  d,
  title = "model performance by epoch, dataset, and measure")

suppressWarnings(print(plt)) # too few points for loess
```

PRPlot *Plot Precision-Recall plot.*

Description

See http://www.nature.com/nmeth/journal/v13/n8/full/nmeth.3945.html

Usage

```
PRPlot(frame, xvar, truthVar, truthTarget, title, ...)
```

Arguments

frame	data frame to get values from
xvar	name of the independent (input or model) column in frame
truthVar	name of the dependent (output or result to be modeled) column in frame
truthTarget	value we consider to be positive
title	title to place on plot
...	no unnamed argument, added to force named binding of later arguments.

Examples

```
set.seed(34903490)
x = rnorm(50)
y = 0.5*x^2 + 2*x + rnorm(length(x))
frm = data.frame(x=x,y=y,yC=y>=as.numeric(quantile(y,probs=0.8)))
frm$absY <- abs(frm$y)
frm$posY = frm$y > 0
frm$costX = 1
WVPlots::PRPlot(frm, "x", "yC", TRUE, title="Example Precision-Recall plot")
```

ROCPlot *Plot receiver operating characteristic plot.*

Description

Plot receiver operating characteristic plot.

Usage

```
ROCPlot(frame, xvar, truthVar, truthTarget, title, ..., returnScores = FALSE,
  nrep = 100, parallelCluster = NULL)
```

Arguments

frame	data frame to get values from
xvar	name of the independent (input or model) column in frame
truthVar	name of the dependent (output or result to be modeled) column in frame
truthTarget	value we consider to be positive
title	title to place on plot
...	no unnamed argument, added to force named binding of later arguments.
returnScores	logical if TRUE return detailed permutedScores
nrep	number of permutation repititions to estimate p values.
parallelCluster	(optional) a cluster object created by package parallel or package snow.

Examples

```
set.seed(34903490)
x = rnorm(50)
y = 0.5*x^2 + 2*x + rnorm(length(x))
frm = data.frame(x=x,yC=y>=as.numeric(quantile(y,probs=0.8)))
WVPlots::ROCPlot(frm, "x", "yC", TRUE, title="Example ROC plot")
```

ROCPlotPair *Plot two receiver operating characteristic from the same data.frame.*

Description

Plot two receiver operating characteristic from the same data.frame.

Usage

```
ROCPlotPair(frame, xvar1, xvar2, truthVar, truthTarget, title, ...,
  returnScores = FALSE, nrep = 100, parallelCluster = NULL)
```

Arguments

frame	data frame to get values from
xvar1	name of the first independent (input or model) column in frame
xvar2	name of the second independent (input or model) column in frame
truthVar	name of the dependent (output or result to be modeled) column in frame
truthTarget	value we consider to be positive
title	title to place on plot
...	no unnamed argument, added to force named binding of later arguments.
returnScores	logical if TRUE return detailed permutedScores
nrep	number of permutation repititions to estimate p values.
parallelCluster	(optional) a cluster object created by package parallel or package snow.

Examples

```
set.seed(34903490)
x1 = rnorm(50)
x2 = rnorm(length(x1))
y = 0.2*x2^2 + 0.5*x2 + x1 + rnorm(length(x1))
frm = data.frame(x1=x1,x2=x2,yC=y>=as.numeric(quantile(y,probs=0.8)))
# WVPlots::ROCPlot(frm, "x1", "yC", TRUE, title="Example ROC plot")
# WVPlots::ROCPlot(frm, "x2", "yC", TRUE, title="Example ROC plot")
WVPlots::ROCPlotPair(frm, "x1", "x2", "yC", TRUE,
   title="Example ROC pair plot")
```

ROCPlotPair2 *Plot two receiver operating characteristic plots from unrelated frames.*

Description

Plot two receiver operating characteristic plots from unrelated frames.

Usage

```
ROCPlotPair2(nm1, frame1, xvar1, truthVar1, truthTarget1, nm2, frame2, xvar2,
  truthVar2, truthTarget2, title, ..., returnScores = FALSE, nrep = 100,
  parallelCluster = NULL)
```

Arguments

nm1	name of first model
frame1	data frame to get values from
xvar1	name of the first independent (input or model) column in frame
truthVar1	name of the dependent (output or result to be modeled) column in frame
truthTarget1	value we consider to be positive
nm2	name of second model
frame2	data frame to get values from
xvar2	name of the first independent (input or model) column in frame
truthVar2	name of the dependent (output or result to be modeled) column in frame
truthTarget2	value we consider to be positive
title	title to place on plot
...	no unnamed argument, added to force named binding of later arguments.
returnScores	logical if TRUE return detailed permutedScores
nrep	number of permutation repititions to estimate p values.
parallelCluster	(optional) a cluster object created by package parallel or package snow.

Examples

```
set.seed(34903490)
x1 = rnorm(50)
x2 = rnorm(length(x1))
y = 0.2*x2^2 + 0.5*x2 + x1 + rnorm(length(x1))
frm = data.frame(x1=x1,x2=x2,yC=y>=as.numeric(quantile(y,probs=0.8)))
# WVPlots::ROCPlot(frm, "x1", "yC", TRUE, title="Example ROC plot")
# WVPlots::ROCPlot(frm, "x2", "yC", TRUE, title="Example ROC plot")
WVPlots::ROCPlotPair2('train',frm, "x1", "yC", TRUE,
                      'test', frm, "x2", "yC", TRUE,
                      title="Example ROC pair plot")
```

ScatterBoxPlot *Plot a scatter box plot. xvar is the discrete variable (input or model) and yvar is the continuous variable.*

Description

Plot a scatter box plot. xvar is the discrete variable (input or model) and yvar is the continuous variable.

Usage

```
ScatterBoxPlot(frm, xvar, yvar, title, ..., pt_alpha = 0.3)
```

Arguments

frm	data frame to get values from
xvar	name of the independent column in frame; assumed discrete. if frm[[xvar]] is an integer column, it will be converted to a factor. This means that additional layers that rely on continuous x scales (like geom_smooth) won't work
yvar	name of the continuous column in frame
title	plot title
...	(doesn't take additional arguments, used to force later arguments by name)
pt_alpha	transparency of points in scatter plot

Examples

```
classes = c("a", "b", "c")
means = c(2, 4, 3)
names(means) = classes
label = sample(classes, size=1000, replace=TRUE)
meas = means[label] + rnorm(1000)
frm2 = data.frame(label=label,
                  meas = meas)
WVPlots::ScatterBoxPlot(frm2, "label", "meas", pt_alpha=0.2, title="Example Scatter/Box plot")
```

ScatterBoxPlotH *Plot a scatter plot in horizontal mode. xvar is the continuous variable and yvar is the discrete variable (input or model) and*

Description

Plot a scatter plot in horizontal mode. xvar is the continuous variable and yvar is the discrete variable (input or model) and

Usage

```
ScatterBoxPlotH(frm, xvar, yvar, title = "", ..., pt_alpha = 0.3)
```

Arguments

frm	data frame to get values from
xvar	name of the continuous column in frame
yvar	name of the independent column in frame; assumed discrete. if frm[[yvar]] is an integer column, it will be converted to a factor. This means that additional layers that rely on continuous x scales (like geom_smooth) won't work
title	plot title
...	(doesn't take additional arguments, used to force later arguments by name)
pt_alpha	transparency of points in scatter plot

Examples

```
classes = c("a", "b", "c")
means = c(2, 4, 3)
names(means) = classes
label = sample(classes, size=1000, replace=TRUE)
meas = means[label] + rnorm(1000)
frm2 = data.frame(label=label,
                  meas = meas)
WVPlots::ScatterBoxPlotH(frm2, "meas", "label",  pt_alpha=0.2, title="Example Scatter/Box plot")
```

ScatterHist *Plot a scatter plot with marginals. xvar is the independent variable (input or model) and yvar is the dependent variable*

Description

Plot a scatter plot with marginals. xvar is the independent variable (input or model) and yvar is the dependent variable

Usage

```
ScatterHist(frame, xvar, yvar, title, ..., smoothmethod = "auto",
  annot_size = 5, minimal_labels = TRUE, binwidth_x = NULL,
  binwidth_y = NULL, adjust_x = 1, adjust_y = 1, point_alpha = 0.5,
  contour = FALSE)
```

Arguments

frame	data frame to get values from
xvar	name of the independent (input or model) column in frame
yvar	name of the dependent (output or result to be modeled) column in frame
title	title to place on plot
...	no unnamed argument, added to force named binding of later arguments.
smoothmethod	(optional) one of 'auto' (the default), 'loess', 'gam', 'lm', or 'identity'. If smoothmethod is 'auto' or 'lm' a smoothing curve or line (respectively) is added and R-squared of the best linear fit of xvar to yvar is reported. If smoothmethod is 'identity' then the y=x line is added and the R-squared of xvar to yvar (without the linear transform used in the other smoothmethod modes) is reported.
annot_size	numeric scale annotation text (if present)
minimal_labels	logical drop some annotations
binwidth_x	numeric binwidth for x histogram
binwidth_y	numeric binwidth for y histogram
adjust_x	numeric adjust x density plot
adjust_y	numeric adjust y density plot
point_alpha	numeric opaqueness of the plot points
contour	logical if TRUE add a 2d contour plot

Examples

```
set.seed(34903490)
x = rnorm(50)
y = 0.5*x^2 + 2*x + rnorm(length(x))
```

```
frm = data.frame(x=x,y=y)
WVPlots::ScatterHist(frm, "x", "y", title="Example Fit",
  contour = TRUE)
```

ScatterHistC *Plot a conditional scatter plot with marginals. xvar is the independent variable (input or model), and yvar is the dependent variable, and cvar is the condition code*

Description

Plot a conditional scatter plot with marginals. xvar is the independent variable (input or model), and yvar is the dependent variable, and cvar is the condition code

Usage

```
ScatterHistC(frame, xvar, yvar, cvar, title, ..., annot_size = 3,
  colorPalette = "Dark2", adjust_x = 1, adjust_y = 1)
```

Arguments

frame	data frame to get values from
xvar	name of the independent (input or model) column in frame
yvar	name of the dependent (output or result to be modeled) column in frame
cvar	name of condition variable
title	title to place on plot
...	no unnamed argument, added to force named binding of later arguments.
annot_size	numeric scale annotation text (if present)
colorPalette	name of a Brewer palette (see http://colorbrewer2.org/)
adjust_x	numeric adjust x density plot
adjust_y	numeric adjust y density plot

Examples

```
set.seed(34903490)
frm = data.frame(x=rnorm(50),y=rnorm(50))
frm$cat <- frm$x+frm$y>0
WVPlots::ScatterHistC(frm, "x", "y", "cat",
                      title="Example Conditional Distribution")
```

ScatterHistN *Plot a height scatter plot with marginals. xvar is the independent variable (input or model), and yvar is the dependent variable, and zvar is the condition height.*

Description

Plot a height scatter plot with marginals. xvar is the independent variable (input or model), and yvar is the dependent variable, and zvar is the condition height.

Usage

```
ScatterHistN(frame, xvar, yvar, zvar, title, ..., annot_size = 3,
  colorPalette = "RdYlBu", nclus = 3, adjust_x = 1, adjust_y = 1)
```

Arguments

frame	data frame to get values from
xvar	name of the independent (input or model) column in frame
yvar	name of the dependent (output or result to be modeled) column in frame
zvar	name of height variable
title	title to place on plot
...	no unnamed argument, added to force named binding of later arguments.
annot_size	numeric scale annotation text (if present)
colorPalette	name of a Brewer palette (see http://colorbrewer2.org/)
nclus	scalar number of z-clusters to plot
adjust_x	numeric adjust x density plot
adjust_y	numeric adjust y density plot

Examples

```
set.seed(34903490)
frm = data.frame(x=rnorm(50),y=rnorm(50))
frm$z <- frm$x+frm$y
WVPlots::ScatterHistN(frm, "x", "y", "z", title="Example Joint Distribution")
```

ShadedDensity *Plot the distribution of a variable with a tail shaded*

Description

Plot the distribution of a variable with a tail shaded

Usage

```
ShadedDensity(frame, xvar, threshold, title, ..., tail = "left")
```

Arguments

frame	data frame to get values from
xvar	name of the variable to be density plotted
threshold	boundary value for the tail
title	title to place on plot
...	no unnamed argument, added to force named binding of later arguments.
tail	which tail to shade, 'left' (default) or 'right'

Examples

```
set.seed(52523)
d = data.frame(meas=rnorm(100))
threshold = -1.5
WVPlots::ShadedDensity(d, "meas", threshold,
                       title="Example shaded density plot, left tail")
WVPlots::ShadedDensity(d, "meas", -threshold, tail="right",
                       title="Example shaded density plot, right tail")
```

WVPlots *WVPlots: Common Plots for Analysis*

Description

Select data analysis plots, under a standardized calling interface implemented on top of 'ggplot2' and 'plotly'. Plots of interest include: 'ROC', gain curve, scatter plot with marginal distributions, conditioned scatter plot with marginal densities. box and stem with matching theoretical distribution, density with matching theoretical distribution.

Details

For more information:

- `vignette(package='WVPlots')`
- `RShowDoc('WVPlots_examples',package='WVPlots')`
- Website: https://github.com/WinVector/WVPlots

Index