# Package 'diplot'

January 1, 2010

**Type** Package

**Title** Tools to plot descriptive statistics, posterior distributions, and more.

**Version** 0.8

**Date** 2008-08-24

**Author** Stephen R. Haptonstahl <srh@haptonstahl.org>

**Maintainer** Stephen R. Haptonstahl <srh@haptonstahl.org>

**Depends** nnet

**Description** Tools to plot descriptive statistics, posterior distributions, etc.

**License** What license is it under?

## R topics documented:



---

`descripBarplot` *Barplot for Descriptive Statistics*

---

### Description

Given a data frame with one column designated as the response/dependent variable, draws an annotated bar plot giving a breakdown of the other repsonse/independent variables.

## Usage

```
descripBarplot(X,
  yName,
  yValues=sort(unique(X[,yName])),
  factorNames=sort(names(X)[-grep(yName,names(X))]),
  fancyFactorNames=factorNames,
  fancyFactors,
  main=yName,
  sub="",
  xlab="",
  ylab="",
  horizScale="percent",
  mai=c(.75,0,.5,0),
  overallLabel="Overall",
  barFill="lines",
  barHeight=.25,
  barWidth=3,
  barCols
)
```

## Arguments

| | |
|---|---|
| X | the dataframe containing the data to be described |
| yName | name of the column with the response variable |
| yValues | specifies the order (left to right) of the values of y; must be a permutation of unique(X[,yName]) |
| factorNames | vector of strings containing names of columns in X containing factors (coerced if necessary) |
| fancyFactorNames | printed names of the categories |
| fancyFactors | printed names of the category values |
| main | passed to plot |
| sub | passed to plot |
| xlab | passed to plot |
| ylab | passed to plot |
| horizScale | either of "percent" or "fraction" |
| mai | space around box containing bars c(bottom, left, top, right) in inches |
| overallLabel | Title for "Overall" (Wow, how descriptive!) |
| barFill | one of c("lines", "color", "grey") |
| barHeight | in inches |
| barWidth | in inches |
| barCols | vector of colors; if set, barFill is automatically set to "color" |

## Value

Returns 'NULL" invisibly.

## Author(s)

Stephen R. Haptonstahl <srh@haptonstahl.org>

## See Also

barplot, ~~~

## Examples

```
n <- 200
sampleData <- data.frame(
  response=sample(c(0:3), n, replace=TRUE),
  f1=as.factor(sample(c("A", "B", "C", "D"), n, replace=TRUE)),
  f2=as.factor(sample(c("E", "F"), n, replace=TRUE))
)

descripBarplot(X=sampleData, yName="response")

descripBarplot(X=sampleData, yName="response", factorNames=c("f1", "f2"),
  main="Test plot", sub="Sub text", xlab="X label",
  fancyFactorNames=c("Factor 1 thing", "Factor 2 thing"),
  fancyFactors=list(c("A thingy", "B whosits", "C-note", "D-fault"),
    c("eMail", "F-Troop")),
  barFill="lines"
)

descripBarplot(X=sampleData, yName="response", factorNames=c("f1", "f2"),
  main="Test plot", sub="Sub text", xlab="X label",
  fancyFactorNames=c("Factor 1 thing", "Factor 2 thing"),
  fancyFactors=list(c("A thingy", "B whosits", "C-note", "D-fault"),
    c("eMail", "F-Troop")),
  barFill="color"
)

descripBarplot(sampleData, yName="response", factorNames=c("f1", "f2"),
  main="Test plot", sub="Sub text", xlab="X label",
  fancyFactorNames=c("Factor 1 thing", "Factor 2 thing"),
  fancyFactors=list(c("A thingy", "B whosits", "C-note", "D-fault"),
    c("eMail", "F-Troop")),
  barFill="grey"
)
```

---

# insertColumn *Insert one or more columns into a data frame*

---

## Description

Given a data frrame and a list of new column names, returns a data frame with columns added at the position specified.

## Usage

```
insertColumn(X, new.col.name, after.column=ncol(X), default=NA)
```

### Arguments

| | |
|---|---|
| X | data frame to which you want to add columns |
| new.col.name | character vector of the names of the columns to add |
| after.column | an integer in 0:ncol(X) giving the column of X that will be immediately to the left of the new columns; alternatively, the exact name of the column that will be immediately to the left of the new columns |
| default | value inserted in the new columns |

### Value

A data frame with the same number of rows as X but with length(new.col.name) more columns.

### Author(s)

Stephen R. Haptonstahl <srh@haptonstahl.org>

### See Also

data.frame

### Examples

```
X <- data.frame(w=1:3, z=6:8)
# both of the following given the same result
Xplus <- insertColumn(X, new.col.name=c("x", "y"), after.column=1)
Xplus <- insertColumn(X, new.col.name=c("x", "y"), after.column="w")
```

---

## linePlotModels *Plot posterior distributions*

---

### Description

'linePlotModels' generates plots of posterior distributions of parameters in a way that shows the same kind of information shown in a typical table of regression coefficients. For each coefficient, plots a horizontal bar like:

```
  - - - ------------O----------- - - - -
min    1st        median        3rd      max
     quartile                 quartile
```

Bars are grouped together for different models so effects can be easily compared across models. A dotted line is plotted at zero to recognize easily significant difference from zero. Mean and +/- 1 standard deviation can be graphed instead.

### Usage

```
linePlotModels(X,
  orderShown, fancyVarNames,
  main="", xlab="", ylab="", sub= "",
  scaleBarThickness=1, scaleBarSpace=1, scaleBarLength=1,
  useMean=F, colLines, pchModels, lwd=1, xlim,
  ...
)
```

## Arguments

| | |
|---|---|
| X | |
| orderShown | A permutation of unique(unlist(lapply(X, colnames))); defualts to |
| fancyVarNames | Text to display for each variable, in order to be displayed |
| main | passed to 'plot' |
| xlab | passed to 'plot' |
| ylab | passed to 'plot' |
| sub | passed to 'plot' |
| scaleBarThickness | Scalar used to adjust bar thickness |
| scaleBarSpace | Scalar used to adjust spacing between bars |
| scaleBarLength | Scalar used to adjust length of bars |
| useMean | Use mean and standard deviation instead of median and 1st/3rd quartiles |
| colLines | Vector of colors for the lines |
| pchModels | Vector of 'pch' codes, one for each model |
| lwd | Used to set line width |
| xlim | passed to 'plot' |
| ... | passed to 'plot' |

## Value

Invisibly returns key plotting parameters, which are useful in generating a legend:

| | |
|---|---|
| rawVarNames | unique(unlist(lapply(X, colnames))) |
| orderShown | default value, or as passed to 'linePlotModels' |
| varNames | default value, or as passed to 'linePlotModels' |
| fancyVarNames | default value, or as passed to 'linePlotModels' |
| colLines | default value, or as passed to 'linePlotModels' |
| pchModels | default value, or as passed to 'linePlotModels' |
| lwd | default value, or as passed to 'linePlotModels' |

## Author(s)

Stephen R. Haptonstahl <srh@haptonstahl.org>

## Examples

```
n <- 200
model1 <- data.frame(
  b0=rnorm(n),
  b1=rnorm(n)*2-5,
  b2=rexp(n)+3
)
model2 <- data.frame(
```

```
  b0=rnorm(n)+4,
  b1=rnorm(n)*1.5-4,
  b3=rnorm(n)+2
)
model3 <- data.frame(
  b0=rnorm(n)+3,
  b1=rnorm(n)*1.5-4,
  b2=rexp(n)+3,
  b3=rnorm(n)+2
)

models <- list(model1, model2, model3)

linePlotModels(models)

linePlotModels(models,
  fancyVarNames=c("intercept", "first slope", "second slope",
    "third slope"),
  main="Posterior Distributions of Model Coefficients",
  xlab="x-label")

linePlotModels(models,
  fancyVarNames=c("intercept", "first slope", "second slope",
    "third slope"),
  main="Posterior Distributions of Model Coefficients",
  xlab="x-label",
  colLines=rainbow(3)
)
```

---

`roundNicely` *Round a number to a "nice round number"*

---

## Description

Given a number or numeric vector, rounds each value up or down to something suitable for using as the limit of a plot.

## Usage

```
roundNicely(x, down=F)
```

## Arguments

| | |
|---|---|
| `x` | number or numeric vector to be rounded |
| `down` | If TRUE rounds values down instead of up (default). |

## Value

Returns a value or numeric vector of rounded valules.

## Author(s)

Stephen R. Haptonstahl <srh@haptonstahl.org>

## See Also

ceiling

## Examples

```
rough <- sort(runif(10, min=-3, max=3))
data.frame(down=roundNicely(rough, down=TRUE), value=rough,
  up=roundNicely(rough))
```

---

# statMode *Statistical mode*

---

## Description

Compute the statistical mode, the most commonly occuring value.

## Usage

```
statMode(x, break.ties.randomly=F)
```

## Arguments

| | |
|---|---|
| x | |
| break.ties.randomly | If TRUE, ties are broken at random. If FALSE (default), ties are broken indiosyncratically but consistently for a given set of values. |

## Value

The modal value of x.

## Author(s)

Stephen R. Haptonstahl <srh@haptonstahl.org>

## See Also

mean, median

## Examples

```
statMode(c(1,2,3,4,4,5,6))
statMode(c("A", "A", "B", "B", "B"))
```

`unfactor` *Convert from factor*

## Description

Given a factor vector, returns a vector of appropriate type that is not a factor. Given a data frame, returns a data frame where each column has been converted from a factor to a variable of appropriate type.

## Usage

```
unfactor(X)
```

## Arguments

| | |
|---|---|
| `X` | vector or data frame to be converted |

## Value

'unfactor' returns a vector or data frame of the same size as X.

## Author(s)

Stephen R. Haptonstahl <srh@haptonstahl.org>

## See Also

`as.factor`

## Examples

```
rb100 <- as.factor(sample(
  c("red","orange","yellow","green","blue","indigo","violet"),
  100, replace=TRUE))
class(rb100)
uf.rb100 <- unfactor(rb100)
class(uf.rb100)

df.sample <- data.frame(rb=rb100,
  numbers=as.factor(sample(1:10, 100, replace=TRUE)))
is.factor(df.sample$rb)
is.factor(df.sample$numbers)
uf.df.sample <- unfactor(df.sample)
is.factor(uf.df.sample$rb)
is.factor(uf.df.sample$numbers)
```

# Index