Package 'PMmisc'

November 21, 2018

Type Package

Title P&M Miscellaneous R Functions

Version 0.1.2

Author Manuel Russon <manuelrusson@gmail.com>, Xuanhua ``Peter'' Yin <peteryin.sju@hotmail.com>

Maintainer Xuanhua ``Peter'' Yin <peteryin.sju@hotmail.com>

Imports robust, ggplot2, grid

Description Miscellaneous functions for data analysis, graphics, data manipulation, statistical investigation, including descriptive statistics, creating leading and lagging variables, portfolio return analysis, time series difference and percentage change calculation, stacking data for higher efficient analysis.

License GPL (>= 2)

Encoding UTF-8

LazyData true

RoxygenNote 6.0.1

NeedsCompilation no

Repository CRAN

Date/Publication 2018-11-21 16:40:03 UTC

R topics documented:

cor.lag *Lag/Lead Correlation*

Description

Calculating correlation of two vectors with lag and lead periods. The correlations are used to determine the lag or lead effect between two variables. The correlation function uses "na.or.complete" method and calculate the Pearson's correlation.

Usage

```
cor.lag(x,y,lag,lead)
```

Arguments

x	:the moving vector
y	:the fixed vector
lag	:number of lag periods
lead	:number of lead periods

Examples

```
cor.lag(mtcars[,1],mtcars[,2],3,3)
```

cor.spearman *Spearman rank correlation*

Description

Calculate Spearman Rank Correlation, which is the nonparametric version of the Pearson product-moment correlation.

Usage

```
cor.spearman(x,y)
```

Arguments

x	:a numeric variable
y	:a numeric variable

Examples

```
cor.spearman(mtcars[,1], mtcars[,3])
```

cv.annu.fv *Calculate future value of annuity*

Description

Calculate future value of an ordinary annuity or an annuity due.

Usage

```
cv.annu.fv(pmt,i,n,type = 0)
```

Arguments

pmt	:the equal amount of payment of each period
i	:interest rate according to the period
n	:number of periods
type	:type = 0 for ordinary annuity, type = 1 for annuity due

Examples

```
cv.annu.fv(100,0.0248,10,0)
```

cv.annu.pv *Calculate present value of annuity*

Description

Calculate present value of an ordinary annuity or an annuity due.

Usage

```
cv.annu.pv(pmt,i,n,k)
```

Arguments

pmt	:the equal amount of payment of each period
i	:interest rate according to the period
n	:number of periods
k	:number of periods deferred until first payment

Examples

```
cv.annu.pv(100,0.0248,10,4)
```

cv.axp *Create logarithm with a random base*

Description

Create a new variable with the base of a random number and power of the selected variable

Usage

```
cv.axp(dataframe, var, n, range)
```

Arguments

dataframe	:a data frame
var	:the variable selected
n	:number of new variables created
range	:the range of base

Examples

```
cv.axp(mtcars,"wt",5,c(1, 2))
```

cv.bondprice *Calculate the plain vanilla bond price*

Description

Calculate the plain vanilla bond price

Usage

```
cv.bondprice(par,c,yield,n,m)
```

Arguments

par	:the face value of the bond
c	:the annual coupon rate of the bond
yield	:the annual yield to maturity of a bond
n	:number of years
m	:compounding period in a year

Examples

```
cv.bondprice(1000,0.0248,0.0248,10,2)
```

cv.diff *Calculating the difference of a time series*

Description

Calculate the difference of a time series, with a specific lag period. The difference is used to show the change in value over set period.

Usage

```
cv.diff(x,n)
```

Arguments

x	: a numeric vector
n	: number of lag periods

Examples

```
cv.diff(mtcars[,2],1)
```

cv.drawdown *Largest draw down of returns*

Description

Calculate largest draw down of a series of returns. This function calculates the maximum decrease in percentage over time, which can be used to test portfolio returns.

Usage

```
cv.drawdown(x)
```

Arguments

x	: a numeric vector of returns

Examples

```
# rnorm() is used to simulate portfolio returns
returns <- rnorm(100)
cv.drawdown(returns)
```

cv.lag *Create a lag variable*

Description

Create a lag variable, with a choice of lag periods. The lag variable can be used to test lag effects between variables.

Usage

```
cv.lag(x,n)
```

Arguments

x	:a vector
n	:number of lag periods

Examples

```
cv.lag(mtcars[,2],3)
data.frame(mtcars,cv.lag(mtcars[,3], 1))
```

cv.lead *Create a lead variable*

Description

Create a lead variable, with a choice of lead periods. The lead variable can be used to test lead effects between variables.

Usage

```
cv.lead(x,n)
```

Arguments

x	:a vector
n	:number of lead periods

Examples

```
cv.lead(mtcars[,2],3)
data.frame(mtcars,cv.lead(mtcars[,3], 3))
```

cv.logs *Create logarithm with a random base*

Description

Create a new variable that is the logarithm of the selected variable with the base of a random number

Usage

```
cv.logs(dataframe, var, n, range)
```

Arguments

dataframe	:a data frame
var	:the variable selected
n	:number of new variables created
range	:the range of base

Examples

```
cv.logs(mtcars,"wt",5,c(1, 2))
```

cv.pctcng *Calculating rate of return of a vector*

Description

Calculating the percentage change of a time series vector for further analysis, including calculating beta of companies, plotting to see the trend of the stock for technical analysis.

Usage

```
cv.pctcng(x,n)
```

Arguments

x	:a numeric vector
n	: number of lag periods

Examples

```
cv.pctcng(mtcars[,1],1)
```

cv.powers *Create nth power variable*

Description

Create a new variable that is the nth power of the selected variable

Usage

```
cv.powers(dataframe, var, n, range)
```

Arguments

dataframe	:a data frame
var	:the variable selected
n	:number of new variables created
range	:the range of power

Examples

```
cv.powers(mtcars,"wt",5,c(1, 2))
```

df.sortcol *Sort a data frame by a column*

Description

Sort a data frame by a column of choice. The column of choice is specified by the number of the column.

Usage

```
df.sortcol(x,n,desc)
```

Arguments

x	:a data frame
n	:number column to sort
desc	:the order of sorting, default set to TRUE; for ascending order set to FALSE

Examples

```
df.sortcol(mtcars,2,desc = TRUE)
```

df.stack *Stack data frame by one classifier*

Description

Stack data frame by one classifier. This function takes the first column as a ordering variable. Then it take the variables names and repeat as the second column. The last column will be data under each variable name. This function is created to enable easier investigation with apply functions.

Usage

```
df.stack(df,name)
```

Arguments

df	: a data frame used to stack
name	: new variable names of the data frame

Examples

```
df <- data.frame(matrix(nrow=100,ncol=100))
for(i in 1:100){
 df[,i] <- rep(runif(1,1,100),100)
}
dim(df)
hdf <- df.stack(df,c("date","tkr","price"))
```

ds.corm *Correlation matrix*

Description

Calculating the correlation matrix of a data frame and return in a data frame object

Usage

```
ds.corm(x,n)
```

Arguments

x	:a data frame
n	:number of decimal points

Examples

```
ds.corm(mtcars,3)
```

ds.kurt *Calculating kurtosis for numeric data.*

Description

Kurtosis

Usage

```
ds.kurt(x)
```

Arguments

x	:a numeric variable

Examples

```
ds.kurt(mtcars[,2])
```

ds.mode *Calculating mode for numeric data*

Description

Calculating mode for numeric data.

Usage

```
ds.mode(x)
```

Arguments

x	:a numeric variable

Examples

```
ds.mode(mtcars[,2])
```

ds.skew *Calculating skewness for numeric data*

Description

Calculating Pearson's skewness in three types: mode, median, and mean.

Usage

```
ds.skew(x, type = 3)
```

Arguments

x	:a numeric variable
type	:type = 1 for mode skewness; type = 2 for median skewness; type = 3 for mean skewness

Examples

```
ds.skew(mtcars[,1])
```

ds.summ *Descriptive statistics of a data frame*

Description

Calculating the descriptive statistics of a data frame and exporting in a data frame. The report data frame contains: number of observations, maximum value, minimum value, mean, median, mode, variance, standard deviation, skewness and kurtosis.

Usage

```
ds.summ(x,n)
```

Arguments

x	:a data frame
n	:number of decimal points rounded

Examples

```
ds.summ(mtcars,3)
```

pl.2ts *Time series plot for two variables*

Description

Plotting two time series in one plot, with title.

Usage

```
pl.2ts(ts1,ts2,title)
```

Arguments

ts1	:time series variable one
ts2	:time series variable two
title	:title for the plot

Examples

```
DAX <- EuStockMarkets[,1]
FTSE <- EuStockMarkets[,4]
pl.2ts(DAX,FTSE, "Times Series Plot of DAX and FTSE")
```

pl.3smoothtxt *Scatter smooth plot with text overlay*

Description

Generate a scatter plot with text overlay, with a smooth curve fitted by loess.

Usage

```
pl.3smoothtxt(x,y,txt,ce)
```

Arguments

x	: a numeric vector
y	: a numeric vector
txt	: a vector used as labels
ce	: text size, which default is set as 0.5

Examples

```
pl.3smoothtxt(mtcars[,1], mtcars[,3], row.names(mtcars))
```

pl.3txt *Scatter plot with text overlay*

Description

Generate a scatter plot with text overlay. This plot is to better show the effect of the text variable in the domain of x and y variable.

Usage

```
pl.3txt(x,y,txt,title)
```

Arguments

x	:a numeric vector
y	:a numeric vector
txt	:a vector used as labels
title	:title of the graph

Examples

```
pl.3txt(mtcars[,1], mtcars[,3], row.names(mtcars),"mpg v. cyl")
```

pl.coplot *Scatter plot of x and y divided by z*

Description

Generate 4 scatter plots of x and y divided by variable z, with a fitted line using a robust linear regression method.

Usage

```
pl.coplot(x,y,z,varN)
```

Arguments

x	:x-axis value
y	:y-axis value
z	:classification variable used to condition plots based on ascending values of z
varN	:variable name of z

Examples

```
pl.coplot(mtcars[,1], mtcars[,3], mtcars[,4], "hp")
```

pl.hist *Plot histograms for a data frame*

Description

Plotting histograms for a data frame, with titles and label numbers.

Usage

```
pl.hist(x, l = 1)
```

Arguments

x	:a data frame
l	: the beginning label number in the title (default set to 1)

Examples

```
pl.hist(mtcars,1)
```

pl.histgg *Plot histograms for a data frame with ggplot2*

Description

Plotting histograms for a data frame with 4 per image, with titles and label numbers automatically generated.

Usage

```
pl.histgg(x, l = 1)
```

Arguments

x	:a data frame
l	: the beginning label number in the title (default set to 1)

Note

This function uses Freedman-Diaconis rule for histogram bin size calculation. In some occasions feed discrete variables to this function may yield terrible results.

References

Freedman, David; Diaconis, Persi (December 1981). "On the histogram as a density estimator: L2 theory" (PDF). Probability Theory and Related Fields. Heidelberg: Springer Berlin. 57 (4): 453–476. doi:10.1007/BF01025868. ISSN 0178-8051. Retrieved 2009-01-06.

Examples

```
pl.histgg(as.data.frame(EuStockMarkets),1)
```

`pl.hs` *Plot histograms and scatter plots for a data frame*

Description

Plotting histograms or scatter plots of your choice for a data frame. Also the function will name the graphs and number them. The purpose of the function is to save time when plotting graphs for a regression analysis or other usage. The function can plot, name and number the graphs at one step.

Usage

```
pl.hs(x,a,dependent,l)
```

Arguments

`x`	:a data frame
`a`	:the type of graph you want; a = 1 for histograms; a = 2 for scatter plots; a = 0 for both
`dependent`	:the dependent variable for scatterplots
`l`	: the beginning label number in the title (default set to 1)

Examples

```
pl.hs(mtcars,0,"mpg",1)
```

`pl.hsd` *Plot histogram with density line for a data frame*

Description

Plotting histogram with density for a data frame, with titles and label numbers.

Usage

```
pl.hsd(dataframe,l)
```

Arguments

`dataframe`	:a data frame
`l`	: the beginning label number in the title (default set to 1)

Examples

```
pl.hsd(mtcars,1)
```

pl.multiplot *Multiple plot function for ggplot2 objects*

Description

Render multiple ggplt2 plots on one page

Usage

```
pl.multiplot(..., plotlist = NULL, cols = 1, layout = NULL)
```

Arguments

...	:ggplot2 objects
plotlist	:a list of ggplot2 objects
cols	:number of columns in the layout
layout	:a matrix that specifies the layout. It has higher priority than 'cols'

References

https://stackoverflow.com/questions/24387376/r-error-could-not-find-function-multiplot-using-cookbook-example

pl.s *Plot scatter plots for a data frame*

Description

Plotting scatter plots for a data frame, with titles and label numbers.

Usage

```
pl.s(x,dependent,l)
```

Arguments

x	:a data frame, which includes the dependent variable
dependent	:the dependent variable for scatter pl.s
l	: the beginning label number in the title (default set to 1)

Examples

```
pl.s(mtcars,"mpg",1)
```

pl.sm *Plot scatter smooth plots for a data frame*

Description

Plotting scatter smooth plots for a data frame, with titles and label numbers.

Usage

```
pl.sm(x,dependent,l)
```

Arguments

x	:a data frame, which includes the dependent variable
dependent	:the dependent variable for scatter smooth plots
l	: the beginning label number in the title (default set to 1)

Examples

```
pl.sm(mtcars,"mpg",1)
```

pl.ts *Plot time series plots for a data frame*

Description

Plotting time series plots for a data frame, with titles and label numbers.

Usage

```
pl.ts(x, l = 1)
```

Arguments

x	:a data frame
l	: the beginning label number in the title (default set to 1)

Examples

```
pl.ts(mtcars,1)
```

pl.tss *Time series plot with multiple variables*

Description

This function will return a time series plot with up to 6 variables, each with different line type.

Usage

```
pl.tss(dataframe,ylb,title)
```

Arguments

dataframe	:a data frame
ylb	:y-axis label
title	:plot title

Examples

```
pl.tss(EuStockMarkets,"Price","Daily Closing Prices of Major European Stock Indices")
```

reg.adj.r.squared *Adjusted R-squared for lm.fit*

Description

Calculate Adjusted R-squared for the outcome of lm.fit. This function is built for reg.linreg() for higher efficiency only. It can't be used for calculating Adjusted R-squared in general operation.

Usage

```
reg.adj.r.squared(r,n,p)
```

Arguments

r	:R-squared for regression
n	:number of observations aka. sample size
p	:number of explanatory variables in the model

Examples

```
X <- as.matrix(cbind(1,EuStockMarkets[,1:2])) # create the design matrix
Y <- as.data.frame(EuStockMarkets)$FTSE
fit <- lm.fit(x = X, y = Y)
SSR <- sum((fit$fitted.values - mean(Y))^2)
SSTO <- sum((Y - mean(Y))^2)
r <- reg.r.squared(SSR,SSTO)
n <- dim(X)[1]; p <- dim(X)[2]
reg.adj.r.squared(r,n,p)
```

---

reg.aic *AIC for lm.fit*

Description

Calculate AIC for the outcome of AIC. This function is built for reg.linreg for higher efficiency only. It can't be used for calculating AIC in general operation.

Usage

```
reg.aic(fit,w)
```

Arguments

fit	:the outcome of lm.fit
w	:wright

Examples

```
X <- as.matrix(cbind(1,EuStockMarkets[,1:2])) # create the design matrix
Y <- as.data.frame(EuStockMarkets)$FTSE
fit <- lm.fit(x = X, y = Y)
w <- rep(1,length(Y))
reg.aic(fit,w)
```

---

reg.bic *BIC for lm.fit*

Description

Calculate BIC for the outcome of lm.fit This function is built for reg.linreg() for higher efficiency only. It can't be used for calculating BIC in general operation.

Usage

```
reg.bic(fit,w)
```

Arguments

fit	:the outcome of lm.fit
w	:wright

Examples

```
X <- as.matrix(cbind(1,EuStockMarkets[,1:2])) # create the design matrix
Y <- as.data.frame(EuStockMarkets)$FTSE
fit <- lm.fit(x = X, y = Y)
w <- rep(1,length(Y))
reg.bic(fit,w)
```

---

reg.dof *Degree of freedom for lim.fit*

Description

Calculate degree of freedom for the outcome of lm.fit(). This function is built for reg.linreg for higher efficiency only. It can't be used for calculating degree of freedom in general operation.

Usage

```
reg.dof(fit)
```

Arguments

fit	:outcome of lm.f

Examples

```
X <- as.matrix(cbind(1,EuStockMarkets[,1:2])) # create the design matrix
Y <- as.data.frame(EuStockMarkets)$FTSE
fit <- lm.fit(x = X, y = Y)
reg.dof(fit)
```

---

reg.dw *Durbin-Watson Test*

Description

Performs the Durbin-Watson Test for a regression model

Usage

```
reg.dw(fit)
```

Arguments

fit	:a lm object

Examples

```
fit <- lm(mpg~wt, mtcars, na.action = na.omit)
reg.dw(fit)
```

---

reg.linreg *Linear regression processor*

Description

This function will take a data frame and the dependent variable and fit all possible combinations of models. The result will be a data frame of models and test statistics for all the models possible. The test statistics are current set and contain all the following: R-squared, Adjusted R-squared, Degree of freedom, Residual standard error, AIC, BIC, Durbin-Watson statistic.

Usage

```
reg.linreg(dataframe,dependent)
```

Arguments

dataframe	:a data frame, which includes the dependent variable
dependent	:dependent variable

Examples

```
reg.linreg(mtcars,"mpg")
```

---

reg.model *Linear model generator*

Description

This function will take a data frame and generate all the combinations of linear model

Usage

```
reg.model(dataframe,dependent)
```

Arguments

dataframe	:a data frame
dependent	: dependent variable

Examples

```
reg.model(mtcars,"mpg")
```

---

reg.r.squared *R-squared for lm.fit*

Description

Calculate R-squared for the outcome of lm.fit(). This function is built for reg.linreg for higher efficiency only. It can't be used for calculating R-squared in general operation.

Usage

```
reg.r.squared(SSR,SSTO)
```

Arguments

SSR	:regression sum of squares or explained of squares
SSTO	:total sum of squares

Examples

```
X <- as.matrix(cbind(1,EuStockMarkets[,1:2])) # create the design matrix
Y <- as.data.frame(EuStockMarkets)$FTSE
fit <- lm.fit(x = X, y = Y)
me <- mean(Y)
SSR <- sum((fit$fitted.values - me)^2)
SSTO <- sum((Y - me)^2)
reg.r.squared(SSR,SSTO)
```

---

reg.std.err *Standard error for lim.fit*

Description

Calculate standard error for the outcome of lm.fit(). This function is built for reg.linreg for higher efficiency only. It can't be used for calculating standard error in general operation.

Usage

```
reg.std.err(SSE,dof)
```

Arguments

SSE	:error sum of squared aka. residual sum of squared
dof	:degree of freedom

Examples

```
X <- as.matrix(cbind(1,EuStockMarkets[,1:2])) # create the design matrix
Y <- as.data.frame(EuStockMarkets)$FTSE
fit <- lm.fit(x = X, y = Y)
SSE <- sum((Y - fit$fitted.values)^2)
dof <- reg.dof(fit)
reg.std.err(SSE,dof)
```

`tr.log` *Sigmoid function*

Description

Generate sigmoid curve series, which is a specific case of logistic function, with a control of top and bottom acceleration.

Usage

```
tr.log(x,top,a,b)
```

Arguments

x	:a numeric vector
top	:a numeric value as vertical scaler
a	:a number to control top acceleration of the curve
b	:a number to control bottom acceleration of the curve

Examples

```
 sigc <- round(tr.log(seq(-3, 3, 0.1), 1, -3, 3), 3)
ts.plot(sigc)
```

`tr.logtb` *Logistic function*

Description

Generate logistic series, with set top and bottom value and acceleration.

Usage

```
tr.logtb(x,top,bot,a,b)
```

Arguments

x	:a vector
top	:higher level y asymptote
bot	:lower level y asymptote
a	:a number to control top acceleration of the curve
b	:a number to control bottom acceleration of the curve

Examples

```
tr.logtb(seq(-3, 3, 0.1), 1, 0.4, -3, 3)
```

---

tr.nd *Normal density function*

Description

Calculate normal density function value at x with a mean of mu and standard deviation of sig.

Usage

```
tr.nd(x,mu,sig)
```

Arguments

x	:x value
mu	:mean value
sig	:standard deviation

Examples

```
tr.nd(seq(-3, 3, 0.1), 0, 1)
```

tr.unli *Unit normal loss integral*

Description

Compute the value of the unit normal loss integral, with discontinuity and dispersion

Usage

```
tr.unli(x,disc,disp)
```

Arguments

x	:a vector
disc	:discontinuity
disp	:dispersion

Examples

```
tr.unli(-3:10, 1, 3)
```

xd.avt *Download data from Alpha Vantage*

Description

This function will return time series data of financial securities from Alpha Vantage. Before using this function, a onetime registration is required to obtain a APIKEY, which is unique for users. Also, the key is valid for lifetime.

Usage

```
xd.avt(ticker,type,size="full",apikey,interval)
```

Arguments

ticker	:the ticker for desired financial security. e.g. AAPL, MSFT
type	:data type; usable values: "TIME_SERIES_INTRADAY", "TIME_SERIES_DAILY", "TIME_SERIES_DAILY_ADJUSTED", "TIME_SERIES_WEEKLY", "TIME_SERIES_WEEKLY_AD "TIME_SERIES_MONTHLY", "TIME_SERIES_MONTHLY_ADJUSTED"
size	:the size of data downloaded; use "compact" for the latest 100 observations and "full" for at most the last 20 years as claimed by the website. However, most data only go back to Jan. 2000.
apikey	:a string that is obtained by a onetime registration
interval	:required for intraday data; usable values: "1min", "5min", "15min", "30min", "60min"

Examples

```
# All parameters are required to be strings
# The example is to download Ford Motor Company's daily adjusted price.
# The apikey here is just for demonstration purposes.
T <- xd.avt("T","TIME_SERIES_DAILY","full","QB45BDBGP0O7W8TB")
```

---

xd.fred *Download data from Federal Reserve Bank of St. Louis*

Description

This function returns a data from the Federal Reserve Bank of St. Louis database

Usage

```
xd.fred(tkr, start_date, end_date)
```

Arguments

tkr	:data tickers used by the database
start_date	:starting date of the data(default is set as 1900-01-01)
end_date	:ending date of the data(default is set as 2018-01-01)

Examples

```
cpi <- xd.fred("CPIAUCSL") # CPI data
head(cpi)
tail(cpi)

#Frequently used tickers:
#CPIAUCSL: Consumer Price Index for All Urban Consumers: All Items
#A191RL1Q225SBEA: Real Gross Domestic Product
#DGS10: 10-Year Treasury Constant Maturity Rate
#UNRATE: Civilian Unemployment Rate
```

Index