Package ‘YRmisc’

March 25, 2020

Type Package

Title Y&R Miscellaneous R Functions

Version 0.1.6

Author
Manuel Russon <RUSSONM@stjohns.edu>, Xuanhua (Peter) Yin <peteryin.sju@hotmail.com>

Maintainer Xuanhua (Peter) Yin <peteryin.sju@hotmail.com>

Imports ggplot2, grid, gridExtra

Description Miscellaneous functions for data analysis, portfolio management, 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 7.0.2

NeedsCompilation no

Repository CRAN

Date/Publication 2020-03-25 16:30:05 UTC

R topics documented:

cor.lag ................................................................. 3
cor.spearman ....................................................... 4
cv.annu.fv ......................................................... 4
cv.annu.pv ......................................................... 5
cv.axp ............................................................. 5
cv.bondprice ....................................................... 6
cv.diff ............................................................. 6
cv.drawdown ....................................................... 7
cv.lag ............................................................. 7
cv.lead ............................................................ 8
R topics documented:

- cv.logs .................................................. 8
- cv.pctcng ............................................. 9
- cv.powers ............................................. 9
- df.sortcol ............................................ 10
- df.stack ............................................. 10
- ds.corm .............................................. 11
- ds.kurt .............................................. 11
- ds.mode .............................................. 12
- ds.skew .............................................. 12
- ds.summ .............................................. 13
- pl.2ts ............................................... 13
- pl.2tsgg ............................................ 14
- pl.3smoothtxt ...................................... 14
- pl.3smoothtxtgg ................................... 15
- pl.3txt ............................................. 15
- pl.3txtgg ........................................... 16
- pl.coplot ........................................... 17
- pl.hist .............................................. 17
- pl.histgg ........................................... 18
- pl.hs ............................................... 18
- pl.hsd .............................................. 19
- pl.hsdgg ........................................... 19
- pl.mv ............................................... 20
- pl.s ................................................ 20
- pl.sgg ............................................. 21
- pl.sm .............................................. 21
- pl.smgg ............................................ 22
- pl.ts ............................................... 22
- pl.tsgg ............................................ 23
- pl.tss .............................................. 23
- pt.alpha ............................................ 24
- pt.annexrtn ....................................... 24
- pt.annrtn ......................................... 25
- pt.anmsd .......................................... 25
- pt.beta ............................................ 26
- pt.bias .............................................. 26
- pt.btavg ........................................... 27
- pt.cmexrtn ....................................... 27
- pt.cmrtin ......................................... 28
- pt.dalpha .......................................... 28
- pt.dbeta .......................................... 28
- pt.exploss ........................................ 29
- pt.hismv .......................................... 30
- pt.info ............................................... 30
- pt.jalpha .......................................... 31
- pt.m2 ................................................ 31
- pt.probloss ........................................ 32
- pt.roy ............................................... 33
**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**

\[ \text{cor.lag}(x, y, \text{lag}, \text{lead}) \]

**Arguments**

- \text{x} : the moving vector
- \text{y} : the fixed vector
- \text{lag} : number of lag periods
- \text{lead} : number of lead periods
Examples

\texttt{cor.lag(mtcars[,1], mtcars[,2], 3, 3)}

\begin{verbatim}
  cor.spearman
  Spearman rank correlation
\end{verbatim}

Description

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

Usage

\texttt{cor.spearman(x, y)}

Arguments

- \texttt{x} : a numeric variable
- \texttt{y} : a numeric variable

Examples

\texttt{cor.spearman(mtcars[,1], mtcars[,3])}

\begin{verbatim}
  cv.annu.fv
  Calculate future value of annuity
\end{verbatim}

Description

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

Usage

\texttt{cv.annu.fv(pmt, i, n, type = 0)}

Arguments

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

Examples

\texttt{cv.annu.fv(100, 0.0248, 10, 0)}
cv.annu.pv

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

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

---

cv.axp

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

**Usage**
```r
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**
```r
cv.axp(mtcars, "wt", 5, c(1, 2))
```
cv.bondprice  
*Calculate the plain vanilla bond price*

**Description**

Calculate the plain vanilla bond price

**Usage**

```r
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**

```r
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**

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

**Arguments**

- `x`: a numeric vector
- `n`: number of lag periods

**Examples**

```r
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))
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))

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**

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

**Arguments**

- `x`: a numeric variable

**Examples**

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

---

**ds.skew**

Calculating skewness for numeric data

**Description**

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

**Usage**

```r
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**

```r
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.2tsgg

Time series plot for two variables with ggplot2

Description
Plotting two time series in one plot, with title and label. If both variables are time series object, they will be merged by time. If both variables are not time series object, they will be merged by order. The first variable is set to be a solid line and the second variable is set to be a dashed line. If the variables are of different type a warning message will be given.

Usage
pl.2tsgg(ts1, ts2, title, ylab)

Arguments
- ts1: a time series variable or a numeric variable
- ts2: a time series variable or a numeric variable
- title: title for the plot
- ylab: y-axis label

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

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

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

---

### pl.3smoothtxtgg

**Scatter smooth plot with text overlay using ggplot2**

**Description**

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

**Usage**

```r
pl.3smoothtxtgg(x,y,txt,size,title,xlab,ylab)
```

**Arguments**

- `x`: a numeric vector
- `y`: a numeric vector
- `txt`: a vector used as labels
- `size`: text size, which default is set as 3
- `title`: graph title
- `xlab`: x-axis label
- `ylab`: y-axis label

**Examples**

```r
pl.3smoothtxtgg(mtcars[,1], mtcars[,3], row.names(mtcars), 3, "MPG v. DISP","mpg","disp")
```

---

### 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**

```r
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

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

---

**Description**

Generate a scatter plot with text overlay with ggplot2. This plot is to better show the effect of the text variable in the domain of x and y variable.

**Usage**

```r
pl.3txtgg(x, y, txt, size, title, xlab, ylab)
```

**Arguments**

- **x**: a numeric vector
- **y**: a numeric vector
- **txt**: a vector used as labels
- **size**: text size, which default is set as 3
- **title**: title of the graph
- **xlab**: x-axis label
- **ylab**: y-axis label

**Examples**

```r
pl.3txtgg(mtcars[,1], mtcars[,3], row.names(mtcars), 3, "mpg v. cyl", "mpg", "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 simple linear regression method.

**Usage**

```r
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**

```r
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**

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

**Arguments**

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

**Examples**

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

*Plot histograms for a data frame with ggplot2*

**Description**

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

**Usage**

```
pl.histgg(x, l, bin)
```

**Arguments**

- `x`: a data frame
- `l`: the beginning label number in the title (default set to 1)
- `bin`: bin width of histogram (default set to 30)

**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.hsdgg

Plot histograms for a data frame with ggplot2

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

Usage
pl.hsdgg(x,l,bin)

Arguments
- x : a data frame
- l : the beginning label number in the title (default set to 1)
- bin : bin width of the graph

Examples
pl.hsdgg(as.data.frame(EuStockMarkets),1,100)
pl.mv  
*Plot mean-variance simulation result*

Description
This function is used to plot the result of portfolio simulation by pt.mv().

Usage
```
pl.mv(port)
```

Arguments
- `port`: portfolio simulation result from pt.mv()

Examples
```
set.seed(1)
rt <- data.frame(runif(120,-1,1),runif(120,-1,1),runif(120,-1,1),runif(120,-1,1))
names(rt) <- c("asset1","asset2","asset3","asset4")
portfolio <- pt.hismv(rt,1000,0)
pl.mv(portfolio)
```

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 plot
- `l`: the beginning label number in the title (default set to 1)

Examples
```
pl.s(mtcars,"mpg",1)
```
Plots scatter plots for a data frame using ggplot2

**Description**

Plotting scatter plots for a data frame using ggplot2, with titles and label numbers. The output will be 4 graphs per page.

**Usage**

```r
pl.sgg(x, dependent, l)
```

**Arguments**

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

**Examples**

```r
pl.sgg(mtcars, "mpg", 1)
```

---

Plots scatter smooth plots for a data frame

**Description**

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

**Usage**

```r
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**

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

Description
Plotting scatter plots for a data frame using ggplot2, with titles and label numbers. A smooth line will be added using a chosen method. The output will be 4 graphs per page.

Usage
pl.smgg(x, dependent, l, mtd)

Arguments
x : a data frame, which includes the dependent variable
dependent : the dependent variable for scatter plot
l : the beginning label number in the title (default set to 1)
mtd : something method to use, accepts either a character vector or a function, e.g. MASS::rlm, base::lm, base::loess, mgcv::gam

Examples
pl.smgg(mtcars, "mpg", 1, lm)
pl.smgg(mtcars, "mpg", 1, loess)

pl.ts

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.tsgg

Plot times series plot for a data frame with ggplot2

Description
Plotting time series plot for a data frame with 4 per page, with titles and label numbers automatically generated.

Usage
pl.tsgg(x, l)

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

Examples
pl.tsgg(as.data.frame(EuStockMarkets), 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")
\[ \text{pt.alpha} \]  
*Stock return alpha*

**Description**

Alpha is the intercept of a fitted line when dependent variable is the benchmark return and independent variable is a asset return of the same period. It is a measure of the active return on an investment. Alpha, along with beta, is one of the two key coefficients in the CAPM used modern portfolio theory.

**Usage**

\[ \text{pt.alpha}(ar, br) \]

**Arguments**

- `ar`: a vector of a risk asset return
- `br`: a vector of benchmark return

**Examples**

\[
\begin{align*}
\text{brtn} & \leftarrow \text{runif}(100, -1, 1) \\
\text{artn} & \leftarrow \text{runif}(100, -1, 1) \\
\text{pt.alpha}(\text{artn}, \text{brtn})
\end{align*}
\]

\[ \text{pt.annexrtn} \]  
*Annualized excess return*

**Description**

Annualized excess return is the difference between the annualized and cumulative return of the two series. Usually, one series are portfolio returns and the other is a benchmark returns.

**Usage**

\[ \text{pt.annexrtn}(ar, br) \]

**Arguments**

- `ar`: a vector of a risk asset return
- `br`: a vector of benchmark return

**Examples**

\[
\begin{align*}
\text{brtn} & \leftarrow \text{runif}(100, -1, 1) \\
\text{artn} & \leftarrow \text{runif}(100, -1, 1) \\
\text{pt.annexrtn}(\text{artn}, \text{brtn})
\end{align*}
\]
**pt.annrtn**

### Annualized return

**Description**

This function takes a series of annual returns and calculate the annualized return.

**Usage**

```r
pt.annrtn(r, n)
```

**Arguments**

- `r`: annual returns
- `n`: number of years

**Examples**

```r
r <- runif(100, -1, 1) # generate random number to simulate returns
annualizedreturn <- pt.annrtn(r, 100)
```

---

**pt.anmsd**

### Annualized standard deviation

**Description**

The annualized standard deviation is the standard deviation multiplied by the square root of the number of periods in one year.

**Usage**

```r
pt.anmsd(r, n)
```

**Arguments**

- `r`: a vector of a risk asset return
- `n`: number of periods in a year

**Examples**

```r
rtn <- runif(30, -1, 1)
n <- 30
pt.anmsd(rtn, n)
```
### pt.beta

**Stock return beta**

**Description**
Beta is the slope of a fitted line when dependent variable is the benchmark return and independent variable is an asset return of the same period. It is a measure the risk arising from exposure to general market movements.

**Usage**

```
pt.beta(ar, br)
```

**Arguments**
- `ar`: a vector of a risk asset return
- `br`: a vector of benchmark return

**Examples**
```
brtn <- runif(100, -1, 1)
artn <- runif(100, -1, 1)
pt.beta(artn, brtn)
```

### pt.bias

**Bias ratio**

**Description**
The bias ratio is an indicator used in finance analyze the returns of a portfolio, and in performing due diligence.

**Usage**

```
pt.bias(r)
```

**Arguments**
- `r`: a vector of a risk asset return

**Examples**
```
r <- runif(100, 0, 1)  # generate random number to simulate returns
pt.bias(r)
```
**pt.btavg**  
*Batting average*

**Description**

The batting average of the asset is the ratio between the number of periods where the asset outperforms a benchmark and the total number of periods.

**Usage**

\[
\text{pt.btavg}(ar, br)
\]

**Arguments**

- \(ar\) : a vector of a risk asset return
- \(br\) : a vector of a benchmark return

**Examples**

\[
\text{artn} \leftarrow \text{runif}(100, -1, 1)
\text{brtn} \leftarrow \text{runif}(100, -1, 1)
\text{pt.btavg}(\text{artn}, \text{brtn})
\]

---

**pt.cmexrtn**  
*Cumulative excess return*

**Description**

Cumulative return is the compounded return in a given period. The excess return is the difference between the cumulative return of a risky asset and the cumulative return of a benchmark.

**Usage**

\[
\text{pt.cmexrtn}(ar, br)
\]

**Arguments**

- \(ar\) : a vector of risky asset returns
- \(br\) : a vector of benchmark returns

**Examples**

\[
\text{brtn} \leftarrow \text{runif}(12, -1, 1)
\text{artn} \leftarrow \text{runif}(12, -1, 1)
\text{pt.cmexrtn}(\text{artn}, \text{brtn})
\]
**pt.cmrttn**  
*Cumulative return*

**Description**
Cumulative return is the compounded return in a given period.

**Usage**
```r
pt.cmrttn(r)
```

**Arguments**
- `r`: a vector of periodic returns

**Examples**
```r
rtn <- runif(12,-1,1) # generate random number to simulate returns
pt.cmrttn(rtn)
```

**pt.dalpha**  
*Dual-alpha*

**Description**
Dual-alpha method is to divide market alpha into downside beta and upside alpha. The principle behind is that upside and downside alphas are not the same.

**Usage**
```r
pt.dalpha(ar,mr,rf)
```

**Arguments**
- `ar`: a vector of a risk asset return
- `mr`: a vector of market return
- `rf`: risk free rate

**Examples**
```r
artn <- runif(24,0,1) # generate random number to simulate returns
mrtn <- runif(24,-1,1)
pt.dalpha(artn,mrtn,0.024)
```
**pt.dbeta**

### Description

Dual-beta method is to divide market beta into downside beta and upside beta. The principle behind is that upside and downside betas are not the same.

### Usage

\[
\text{pt.dbeta}(ar, mr, rf)
\]

### Arguments

- **ar**: a vector of a risk asset return
- **mr**: a vector of market return
- **rf**: risk free rate

### Examples

```r
artn <- runif(24,0,1) # generate random number to simulate returns
mrttn <- runif(24,-1,1)
pt.dbeta(artn,mrttn,0.024)
```

**pt.exploss**

### Description

This function give the expected loss of given asset returns.

### Usage

\[
\text{pt.exploss}(r, p)
\]

### Arguments

- **r**: a vector of periodic returns
- **p**: target return

### Examples

```r
rt <- runif(12,-1,1) # generate random number to simulate returns
pt.exploss(rt,0)
pt.exploss(rt,1)
```
Mean-variance model with historical average returns and standard deviations

Description
This function will perform portfolio simulation with historical average returns and standard deviations. Mean-variance model, or modern portfolio theory, is a mathematical framework for accessing a portfolio. It uses the variance of asset returns as a risk proxy. This function will return a number of simulated portfolio with different weights.

Usage
pt.hismv(r,n,mini)

Arguments
r : a data frame of asset returns
n : number of portfolio simulated
mini : minimal weight; choose 0 if long only; choose 1 for possible short position

Examples
set.seed(20)
rt <- data.frame(runif(120, -1, 1), runif(120, -1, 1), runif(120, -1, 1), runif(120, -1, 1))
names(rt) <- c("asset1", "asset2", "asset3", "asset4")
portfolio <- pt.hismv(rt, 1000, 0)
plot(portfolio[,6], portfolio[,5], xlab = "standard deviation", ylab = "expected return")

Information ratio

Description
The information ratio of asset's returns versus benchmark returns, is the quotient of the annualized excess return and the annualized standard deviation of the excess return.

Usage
pt.info(ar,br,n)

Arguments
ar : a vector of a risk asset return
br : a vector of benchmark return
n : number of years
Examples

```r
brtn <- runif(100, -1, 1)
artn <- runif(100, 0, 1)
pt.info(artn, brtn, 100)
```

---

**pt.jalpha**  
*Jensen’s alpha*

Description

Jensen’s alpha is a financial statistic used to quantify the abnormal return of a security or portfolio over the theoretical expected return. Unlike standard alpha, it uses theoretical performance return instead of a market return.

Usage

```r
pt.jalpha(pr, mr, rf, beta)
```

Arguments

- **pr**: portfolio return
- **mr**: market return
- **rf**: risk free rate
- **beta**: portfolio beta

Examples

```r
prtn <- runif(24, -1, 1)
mrtn <- runif(24, -1, 1)
rf <- 0.024
pt.jalpha(mean(prtn), mean(mrtn), rf, pt.beta(prtn,mrtn))
```

---

**pt.m2**  
*Modigliani risk-adjusted performance*

Description

Modigliani risk-adjusted performance is a financial measure of risk-adjusted returns of a portfolio. It measures the returns of the portfolio after adjusting it relative to some benchmark.

Usage

```r
pt.m2(pr, br, rf)
```
pt.probloss

Arguments

\[ \text{pr} \]: portfolio return
\[ \text{br} \]: benchmark return
\[ \text{rf} \]: risk free rate

Examples

\begin{verbatim}
prtn <- runif(12,-1,1)
brtn <- runif(12,-1,1)
rf <- 0.024
pt.m2(prtn,brtn,rf)
\end{verbatim}

\begin{verbatim}
pt.probloss(r,p)
\end{verbatim}

Description

This function gives the probability of loss of given asset returns.

Usage

\begin{verbatim}
pt.probloss(r,p)
\end{verbatim}

Arguments

\[ r \]: a vector of periodic returns
\[ p \]: target return

Examples

\begin{verbatim}
rt <- runif(12,-1,1)  # generate random number to simulate returns
pt.probloss(rt,0)
pt.probloss(rt,0.05)
\end{verbatim}
Roy's safety-first criterion

Description
Roy's safety-first criterion is a risk management technique that allows to choose a portfolio based on the criterion that the probability of the portfolio's return falling below a minimum desired threshold is minimized.

Usage
pt.roy(r, mar)

Arguments
- r: a vector of a risk asset return
- mar: minimum acceptable return

Examples
r <- runif(100, 0, 1) # generate random number to simulate returns
pt.roy(r, 0.024)

Standard deviation of excess return

Description
The standard deviation of excess return is simply the standard deviation of the asset return over the benchmark return.

Usage
pt.sdexrtn(ar, br)

Arguments
- ar: a vector of a risk asset return
- br: a vector of benchmark return

Examples
artn <- runif(12, -1, 1)
brtn <- runif(12, -1, 1)
pt.sdexrtn(artn, brtn)
pt.semivar  

**Semivariance of loss**

**Description**

This function gives the semivariance of a losing scenario.

**Usage**

\[ \text{pt.semivar}(r,p) \]

**Arguments**

- **r**: a vector of periodic returns
- **p**: target return

**Examples**

```r
rt <- runif(12,-1,1) # generate random number to simulate returns
pt.semivar(rt,0)
pt.semivar(rt,0.03)
```

pt.sharp  

**Sharp ratio**

**Description**

The Sharpe Ratio of an asset return is the quotient of the annualized excess return of the asset minus the annualized risk-free rate over the annualized standard deviation of the asset return.

**Usage**

\[ \text{pt.sharp}(r,n,m,rf) \]

**Arguments**

- **r**: a vector of asset returns
- **n**: number of years
- **m**: number of periods in a year; \( m = 12 \) if \( r \) is monthly returns
- **rf**: annualized risk-free rate

**Examples**

```r
set.seed(20)
rt <- runif(12,-0.5,1) # generate random number to simulate monthly returns
rfr <- 0.024 # set risk free rate at 2.4% annual
pt.sharp(rtn,1,12,rfr) # the return is for one year
```
**pt.sortino**  

**Sortino ratio**

**Description**

The Sortino ratio is an analog to the sharp ratio, with standard deviation replaced by the downside deviation.

**Usage**

```
pt.sortino(r,p,n,rf)
```

**Arguments**

- `r` : a vector of a risk asset return  
- `p` : target return, aka minimum acceptable return (MAR)  
- `n` : number of years of asset return, used to calculate annualized return  
- `rf` : risk free rate  

**Examples**

```
rtn <- runif(12, -1, 1)
pt.sortino(rtn,0.3,1,0.024)
```

---

**pt.te**  

**Tracking error**

**Description**

Tracking error, in finance, is a measure of risk in a portfolio that is due to active management decisions made by the manager. It indicates how closely the portfolio follows the benchmark of choosing.

**Usage**

```
pt.te(pr,br)
```

**Arguments**

- `pr` : portfolio return  
- `br` : benchmark return  

**Examples**

```
prtn <- runif(12,-1,1)
brtn <- runif(12,-1,1)
pt.te(prtn,brtn)
```
**pt.treynor**

**Treynor ratio**

**Description**

The Treynor ratio is an analog to the sharp ratio, with standard deviation replaced by the asset beta to benchmark.

**Usage**

```
pt.treynor(ar, br, n, rf)
```

**Arguments**

- `ar`: a vector of a risk asset return
- `br`: a vector of benchmark return
- `n`: number of years of asset return, used to calculate annualized return
- `rf`: risk free rate

**Examples**

```
rtn <- runif(24, -1, 1)
brtn <- runif(24, -1, 1)
pt.treynor(rtn, brtn, 2, 0.024)
```

---

**pt.udrtn**

**Average up and down returns**

**Description**

This function calculates the average up and down returns from a series of returns.

**Usage**

```
pt.udrtn(r)
```

**Arguments**

- `r`: a vector of periodic returns

**Examples**

```
r <- runif(100, -1, 1) # generate random number to simulate returns
pt.udrtn(r)
```
**pt.updwcap**  

*Up and down capture*

**Description**

The up and down capture is a measure of how an asset was able to improve on benchmark returns or how it underperforms over the benchmark.

**Usage**

```r
tpt.updwcap(ar, br, n)
```

**Arguments**

- `ar`: a vector of a risk asset return  
- `br`: a vector of benchmark return  
- `n`: number of years of asset return, used to calculate annualized return

**Examples**

```r
artn <- runif(12, -1, 1)  
brtn <- runif(12,-1,1)  
tpt.updwcap(artn,brtn,1)
```

---

**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**

```r
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**

```r
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)
```

---

**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**

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

**Arguments**

- `fit`: the outcome of `lm.fit`
- `w`: wright

**Examples**

```r
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)
```

---

**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**

```r
reg.bic(fit,w)
```
**Arguments**

- **fit**: the outcome of `lm.fit`
- **w**: wright

**Examples**

```r
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)
```

---

**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**

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

**Arguments**

- **fit**: outcome of `lm.f`

**Examples**

```r
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)
```

---

**Description**

Performs the Durbin-Watson Test for a regression model

**Usage**

```r
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
**Example**

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

---

### 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

```r
reg.r.squared(SSR, SST0)
```

### Arguments

- **SSR**: regression sum of squares or explained of squares
- **SST0**: total sum of squares

### Examples

```r
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)
SST0 <- sum((Y - me)^2)
reg.r.squared(SSR, SST0)
```

---

### 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

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

### Arguments

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

```r
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)`
  - `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**
  ```r
  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)`
- **Examples**
  ```r
  sigc <- round(tr.log(seq(-3, 3, 0.1), 1, -3, 3), 3)
  ts.plot(sigc)
  ```
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.fred**  
*Download data from Federal Reserve Bank of St. Louis*

**Description**
This function returns data from the Federal Reserve Bank of St. Louis database. It can take one ticker or a string of tickers, which will output a merged data frame with all observations.

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

**Arguments**
- `tkr`: one data ticker or a string of 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
thead(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

---

**xd.fred.tickers**

*Federal Reserve Bank of St. Louis Economic Data Tickers*

---

**Description**

This function returns a data contains information of data name, type and tickers

**Usage**

```
xd.fred.tickers()
```

**Examples**

```
xd.fred.tickers()
```
Index

cor.lag, 3
cor.spearman, 4
cv.annu.fv, 4
cv.annu.pv, 5
cv.axp, 5
cv.bondprice, 6
cv.diff, 6
cv.drawdown, 7
cv.lag, 7
cv.lead, 8
cv.logs, 8
cv.pctcng, 9
cv.powers, 9
df.sortcol, 10
df.stack, 10
ds.corm, 11
ds.kurt, 11
ds.mode, 12
ds.skew, 12
ds.summ, 13

pl.2ts, 13
pl.2tsgg, 14
pl.3smoothtxt, 14
pl.3smoothtxgg, 15
pl.3txt, 15
pl.3txtgg, 16
pl.coplot, 17
pl.hist, 17
pl.histgg, 18
pl.hs, 18
pl.hsd, 19
pl.hsdgg, 19
pl.mv, 20
pl.s, 20
pl.sgg, 21
pl.sm, 21
pl.smgg, 22
pl.ts, 22
pl.tsgg, 23
pl.tss, 23
pt.alpha, 24
pt.annexrtn, 24
pt.annrtn, 25
pt.annsd, 25
pt.beta, 26
pt.bias, 26
pt.btavg, 27
pt.cmexrtn, 27
pt.cmrt, 28
pt.dalpha, 28
pt.dbeta, 29
pt.exploss, 29
pt.hismv, 30
pt.info, 30
pt.jalpha, 31
pt.m2, 31
pt.probloss, 32
pt.roy, 33
pt.sdexrtn, 33
pt.semiavar, 34
pt.sharp, 34
pt.sortino, 35
pt.te, 35
pt.treynor, 36
pt.udrtn, 36
pt.updwcap, 37
reg.adj.r.squared, 37
reg.aic, 38
reg.bic, 38
reg.dof, 39
reg.dw, 39
reg.linreg, 40
reg.model, 40
reg.r.squared, 41
reg.std.err, 41
tr.log, 42
INDEX

tr.logtb, 42
tr.nd, 43
tr.unli, 44

xd.fred, 44
xd.fred.tickers, 45