Package ‘rpatrec’

May 23, 2017

Type Package

Title Recognising Visual Charting Patterns in Time Series Data

Version 1.0.1

Date 2017-05-23

Description Generating visual charting patterns and noise,
smoothing to find a signal in noisy time series and enabling
users to apply their findings to real life data.

Depends R (>= 2.10)

License GPL-3

Encoding UTF-8

LazyData true

Imports np, stats

RoxygenNote 6.0.1.9000

URL `https://cran.r-project.org/package=rpatrec`

BugReports `http://github.com/maiers94/rpatrec/issues`

Suggests testthat, knitr, rmarkdown, graphics

VignetteBuilder knitr

NeedsCompilation no

Author Stephan Maier [aut, cre]

Maintainer Stephan Maier `<sm297@st-andrews.ac.uk>`

Repository CRAN

Date/Publication 2017-05-23 14:38:21 UTC

R topics documented:

`data` *Daily Closing Prices of major Stock Market Indices*

Description

The file `stockmarket.RData` contains the daily closing prices of the DAX30, Dow Jones Industrials and the Nikkei 225, from January 2000 until December 2016.

Usage

```
data
```

Format

An object of class `data.frame` with 4417 rows and 3 columns.

`generator` *Generate a time series containing a Visual Charting Pattern.*

Description

This function lets you generate any pattern you specify in the parameters. For known definitions, either check the examples or the vignettes. This is mainly used to test either your own or this smoothing and recognition function.

Usage

```
generator(start = 0, dlength = 100, tot.spread = 100, presig = 0,
  postsig = 0, plength = 0, parts = c(15, 25, 50, 75, 85), sprd = c(50,
  25, 100, 25, 50))
```

Arguments

`start`	Starting value
`dlength`	Integer. Length of the signal part of the time series
`tot.spread`	Integer. Difference between the lowest and highest value of the time series
`presig`	Integer. Length of the pre-signal part of the time series
`postsig`	Integer. Length of the post-signal part of the time series
`plength`	No longer needed, kept for compatability. Set to 0.
`parts`	Vector of Integers. Must be the same length as `sprd` Defines how far the extrema lie apart from another (in percent) PREVIOUSLY: Vector must contain plength + 2 elements, the first element being 0 and the last 100.
`sprd`	Vector of Integers. Must be the same length as `parts` Defines the value of the extrema in percent of `tot.spread` in relation to `start`. PREVIOUSLY: Vector must contain plength + 2 elements, the first and last elemnt should be 0.

Details

For an overview of the package capabilities, click here rpatrec

Value

Time series with (optional) pre- or post signal, and the specified pattern.

Examples

```
## Not run:
#create a standard HS pattern:
generator()
#

#create a shifted head and shoulders pattern
generator(sprd = c(20,10,90,40,60))
#
#create a Double Tops pattern
generator(plength=3,parts=c(25,50,75),sprd=c(80,40,80))
#
#create a Rectangle Tops pattern
generator(plength=5,parts=c(20,40,50,60,80),sprd=c(80,40,80,40,80))
#
#create a single peak, 10 data points, max is approximately 10
generator(0,10,10,0,0,0,50,100)

## End(Not run)
```

`interpret` *Recognise patterns in Time Series Data*

Description

Use this function to either check for the inbuilt financial markets pattern or to use your own recognition function as described in the readme.

Usage

```
interpret(window, useriq = FALSE, ...)
```

Arguments

`window`	Time Series Data
`useriq`	User-built recognition function. Set to `FALSE` if using inbuilt recognition capabilities. Refer to the readme or the report on how to build your own recognition function.
`...`	Parameters passed on to either the inbuilt or external recognition function. Check iq for the parameters of the internal function.

Details

For an overview of the package capabilities, click here rpatrec.

Value

A list containing the following:

- 'EXT' All extrema found in the sample, 0 for minima and 1 for maxima
- 'EXP' Value of these extrema (y-coordinate)
- 'EXP' Position of these extrema (x-coordinate)
- Recognition Output A list containing the extrema that form part of the pattern labelled by either the custom or iq function.
  - 'HSP' Can be either:
    - 'HS' (Head and Shoulders)
    - 'InvHS' (Inverse Head and Shoulders)
  - 'BTPorTTP' Can be either:
    - 'BTOP' (Broadening Top)
    - 'BBOT' (Broadening Bottom)
    - 'TTOP' (Triangle Top)
    - 'TBOT' (Triangle Bottom)
  - 'RTP' Can be either:
    - 'RTOP' (Rectangle Top)

        * 'RBOT' (Rectangle Bottom)
    - 'DTP' Can be either:
        * 'DTOP' (Double Top)
        * 'DBOT' (Double Bottom)

- 'RESULT' `TRUE` if any pattern is found, `FALSE` otherwise

Examples

```
## Not run:
#Generate HS patterns
a <- generator()
#switch off recognition for all patterns other than HS
interpret(window = a, useriq=FALSE, hsiq=TRUE, btpiq=FALSE, rtpiq=FALSE, dtpiq=FALSE)

## End(Not run)
```

`iq` *Inbuilt Recoqnition for 10 different financial markets patterns*

Description

Do not call individually. Switch recognition on/off for certain patterns, use the arguments in the interpret or slicer function.

Usage

```
iq(ext, exvals, expos, hsiq = TRUE, btpiq = TRUE, rtpiq = TRUE,
  dtpiq = TRUE)
```

Arguments

`ext`	Extrema
`exvals`	Values of the extrema
`expos`	Position of the extrema
`hsiq`	Logical. Recognise (inverse) Head and Shoulders pattern
`btpiq`	Logical. Recognise Triangle and/or Broadening tops and bottoms pattern
`rtpiq`	Logical. Recognise Rectangle tops and bottoms pattern
`dtpiq`	Logical. Recognise Double tops and bottoms pattern

Details

For an overview of the package capabilities, click here rpatrec.

`kernel` *Perform Kernel Regression on Time Series Data*

Description

Perform kernel regression in line with Lo et al. (2000). Either specify a bandwidth or let it be determined automatically.

Usage

```
kernel(input, bandwidth = "auto")
```

Arguments

`input`	Vector of Time Series Data
`bandwidth`	**numerical** Choice of Bandwith `'auto'` Choose bandwith automatically by Cross Validation for the given sample

Details

For an overview of the package capabilities, click here rpatrec.

Value

Vector containing smoothed time series data, prints the bandwidth used.

Examples

```
## Not run:
#create a standard HS pattern:
a <- generator()
#add noise to this patterns
b <- noise(a,'white',10)
#smooth to regain the signal
c <- kernel(b,2)

## End(Not run)
```

`loess.rpatrec` *Use the inbuilt function loess to smooth time series data.*

Description

Use local regression to fit a global non-parametric model to the data. If the span is smaller than 1 the regression is truly local, if it is larger than 1 all data points in the sample are taken into account

Usage

```
loess.rpatrec(input, span = 0.75, ...)
```

Arguments

`input`	Time series data passed to be smoothed
`span`	The main smoothing parameter.
`...`	Optional: Other arguments passed to loess

Details

For an overview of the package capabilities, click here rpatrec. This function is purely included to provide the standard interface coherent with other smoothers to the user.

Value

Smoothed time series data only, no additional output.

Examples

```
## Not run:
#create a standard HS pattern:
a <- generator()
#add noise to this patterns
b <- noise(a,'white',10)
#smooth to regain the signal
c <- loess.rpatrec(b)

## End(Not run)
```

mav — *Compute the moving average, exponential average or running median.*

Description

Use this function to compute any of these three moving average methods. These are the simplest smoothers available in the package so it may be a good idea to start initial testing with this function.

Usage

```
mav(input, len = 10, method)
```

Arguments

`input`	Time series data to use for computation
`len`	An integer to determine the number of datapoints used for computation
`method`	String. Determines the method of computation. Permissible values are `exponential`, `simple` and `median`

Details

For an overview of the package capabilities, click here rpatrec.

Value

Vector containing the smoothed time series data of length of `input` less length of `len`

Examples

```
## Not run:
#create a standard HS pattern:
a <- generator()
#add noise to this patterns
b <- noise(a,'white',10)
#smooth to regain the signal
c <- kernel(b,2)

## End(Not run)
##simply test the smoother
mav(1:10,5,'exponential')
```

noise — *Add noise to a time series*

Description

This function lets you add artificial noise to time series, normally to patterns generated by generator. Use different types and strengths of noise to test smoothers.

Usage

```
noise(input, type, final_level)
```

Arguments

`input`	Time series to which noise will be added
`type`	String. Defines what type of noise to be added. Acceptable values are: • `white` - white noise with variance `final_level` • `red` - red noise with variance `final_level` (use with caution)
`final_level`	Number or `var`. A number sets the standard deviation to a constant value at each point. `var` sets the standard deviation to increase proportional to the (absolute value of the) signal at peaks.

Details

For an overview of the package capabilities, click here rpatrec

Value

Time series with added noise.

Examples

```
## Not run:
#Generate a HS patterns
a <- generator()
#now add white noise with a standdard deviation of 10
b <- noise(a,'white',10)
plot(b)

## End(Not run)
```

rpatrec *rpatrec: A package to recognise Patterns in (financial) time series Data*

Description

Generating visual charting patterns and noise, smoothing to find a signal in noisy time series and enabling users to apply their findings to real life data.

A - Data generation and Input

- generator Generate your own Patterns
- noise Generate artificial Noise
- sample.pre Prepare real sample data for use with the package

B - Smoothing and Signal Processing

- mav Moving Average/Median
- kernel Kernel Regression
- savgolay Savitzky-Golay Filter
- splines Smoothing Splines
- loess.rpatrec LOESS

C - Recognising Patterns & Testing

- interpret Recognise different Patterns in time series data
- slicer Recognise multiple Patterns in time series data
- test.smoother Test smoothing algorithms with generated data

Annexe - Data

- data Stock Market Indices - Daily Closing Prices

sample.pre *Prepare Data for using with the other functions in the package*

Description

This function removes bank holidays and other days for which only `NA` or repeated data is available. Designed to work with Datastream data.

Usage

```
sample.pre(input)
```

Arguments

`input`	Vector with time series data

Details

For an overview of the package capabilities, click here rpatrec.

Value

Returns a vector conataining time series data ready for further use.

`savgolay` *Perform Savitzgy-Golay smoothing on Time Series Data*

Description

Use this function to smooth your time series data using local polynomial regression, as first pouplarised by Savitzky and Golay (1964).

Usage

```
savgolay(input, width = 4, degree = 2)
```

Arguments

`input`	Input Vector with Time Series Data
`width`	Width of the filter (to each side of the centre)
`degree`	Highest degree polynomial

Details

For an overview of the package capabilities, click here rpatrec. See the report for detailed references

Value

Vector containing smoothed time series data.

Examples

```
savgolay(input=c(1,6,2,46,23,1,2,13,23,35,23,-2,3,23))
## Not run:
#create a standard HS pattern:
a <- generator()
#add noise to this patterns
b <- noise(a,'white',10)
#smooth to regain the signal
c <- savgolay(b,8,2)
```

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

slicer *Recognise Multiple Patterns in a sinlge time series*

Description

Break time series data into smaller 'windows' and pass them to the interpret function. The results are summarised in the output. For details, run interpret on specific windows only.

Usage

```
slicer(data, length, step = 1, useriq = FALSE, ...)
```

Arguments

`data`	Time series Data
`length`	Length of the 'windows'
`step`	Number of Data Points between windows
`useriq`	User-built recognition function. Set to `FALSE` if using inbuilt recognition capabilities. Refer to the readme or the report on how to build your own recognition function
`...`	Parameters passed on to either the inbuilt or external recognition function. Check iq for the parameters.

Details

For an overview of the package capabilities, click here rpatrec

Value

A list containing:

- A vector for every window analysed showing 0 if no pattern and 1 if at least 1 pattern has been found.
- A vector with the starting index of those windows where a pattern has been found.

Examples

```
## Not run:
#Generate 2 HS patterns
a <- c(generator(),generator())
#recognise both HS patterns
#set window size to 100, step size to 100
#switch off recognition for all patterns other than HS
slicer(data = a, length = 100, step = 100, hsiq=TRUE, btpiq=FALSE, rtpiq=FALSE, dtpiq=FALSE)

## End(Not run)
```

---

splines — *Use the inbuilt function smooth.spline to smooth time series data*

Description

This function provides smoothing capabilities using the cubic B-spline

Usage

```
splines(input, spar = 0.5, ...)
```

Arguments

`input`	Time series data passed to be smoothed
`spar`	Smoothing Parameter, value should be between 0 and 1. `NULL` for automatic computation
`...`	Optional: Other arguments passed to smooth.spline

Details

For an overview of the package capabilities, click here rpatrec. This function is purely included to provide the standard interface coherent with other smoothers to the user.

Value

Smoothed time series data only, no additional output.

Examples

```
## Not run:
#create a standard HS pattern:
a <- generator()
#add noise to this patterns
b <- noise(a,'white',10)
#smooth to regain the signal
c <- splines(b)
```

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

`test.smoother` *Test how well a smoother can filter noise from data*

Description

Use this function to find out about the rate of successful recognition of a simple HS pattern for any smoothing function of your choice. Speciffy the noise and other testing conditions.

Usage

```
test.smoother(n = 1, m = 5, incr = 1, max = 20, smoother,
  pattern = TRUE, ntype = "white", ...)
```

Arguments

`n`	number of runs
`m`	number of runs per level of noise
`incr`	value by which the error is increased in each turn
`max`	max number of times the error is increased
`smoother`	Function with pre-defined inputs, so that only the parameter `input` is left to be defined
`pattern`	Check whether pattern was recognised. If `FALSE` only the correct position of extrema is checked.
`ntype`	Noise type. See `noise` function for details.
`...`	other parameters the smoother requires

Details

For an overview of the package capabilities, click here rpatrec. Note that this function may be extremely computationally demanding.

Value

Vector of recognition rates for specified levels of noise

Examples

```
## Not run:
#Test the kernel regression smoother
a <- test.smoother(n=5,m=10,incr=0.5,max=50,smoother = kernel,bandwidth=1)
#Plot the result
plot(a,type='l')

## End(Not run)
```

Index