Package 'fractalrock'

July 2, 2014

Type Package

Title Generate fractal time series with non-normal returns distribution

Version 1.1.0

Date 2013-02-04

Author Brian Lee Yung Rowe

Maintainer Brian Lee Yung Rowe <r@zatonovo.com>

Depends futile.any (>= 1.3.0), futile.logger (>= 1.3.0), timeDate,quantmod

Description The basic principle driving fractal generation of time series is that data is generated iteratively based on increasing levels of resolution. The initial series is defined by a so-called initiator pattern and then generators are used to replace each segment of the initial pattern. Regular,repeatable patterns can be produced by using the same seed and generators. By using a set of generators, non-repeatable time series can be produced. This technique is the basis of the fractal time series process in this package.

License GPL-3

LazyLoad yes

Repository CRAN

Date/Publication 2013-02-05 05:41:05

NeedsCompilation no

R topics documented:

`fractalrock-package` *A package for generating time series data using fractals*

Description

Simulating stock market prices and returns can be accomplished using a number of techniques. Most commonly, geometric brownian motion (aka a random walk) is used to simulate stock prices. Using this technique results in a normal distribution of price returns. As an alternative technique, it is possible to generate price series using fractals. The advantage is that price returns tend to have volatility that clusters, similar to actual returns.

The basic principle driving fractal generation of time series is that data is generated iteratively based on increasing levels of resolution. The initial series is defined by a so-called initiator pattern and then generators are used to replace each segment of the initial pattern. Regular, repeatable patterns can be produced by using the same seed and generators. By using a set of generators, non-repeatable time series can be produced. This technique is the basis of the fractal time series process in this package.

At a later date, implementation of the [modified] rescaled range statistic will be included to provide more analytical insight into the time series data produced by this package.

Details

Package:	fractalrock
Type:	Package
Version:	1.1.0
Date:	2013-02-04
License:	GPL-3

To generate a set of asset prices, the function `getPortfolioPrices`is the most direct way to accomplish this. An xts object will be returned with one time series per 'asset' provided. In addition, the dates will be coerced to fit a given business day calendar based on timeDate.

Investigation into fractals via this package is best accomplished by calling the underlying `fractal` function. This function produces raw values useful for analysis of the fractal generation process.

Author(s)

Brian Lee Yung Rowe <r@zatonovo.com>

References

M. Frame, B. Mandelbrot, N. Neger. Fractal Geometry. 2009. http://classes.yale.edu/fractals/

`fractal` *Create time series based on fractal generators*

Description

The fractal function generates a time series of points using basic principles of fractal patterns. Fractal generation can be used to simulate a time series of asset prices, which has been shown to better reflect the distribution of returns than using a Gaussian random walk. Any number of points can be generated based on specifying the total count or by running over a number of epochs. The range of the data is defined by the given seed for the generation plus the available patterns.

Usage

```
fractal(seeds, patterns, count = NULL, epochs = NULL, ..., type = "uniform")

fractal.uniform(seed, patterns, count = NULL, epochs = NULL, origin = '1970-01-01', date.fun = as.Date

fractal.random(seed, patterns, count = NULL, epochs = NULL, origin = '1970-01-01', date.fun = as.Date,

next.seeds(old.seed, new.seed, pattern, idx, epoch)
```

Arguments

`seeds`	A list of seed patterns to use for generating the time series
`seed`	The seed pattern to use for generating the time series
`patterns`	A single pattern or list of patterns that get randomly selected for each segment being replaced
`count`	The total number of points to create. Either count or epochs must be provided. Specifying count indirectly sets the number of epochs to run and truncates the data appropriately to get the specified number of points.
`epochs`	The total number of epochs to run. Either count or epochs must be populated. Using epochs is good for experimentation to visualize what happens at every stage of the generation.
`origin`	The starting date for the generated time series
`date.fun`	The function to use to parse dates and/or times
`only`	Only use the nth pattern instead of randomly choosing from patterns
`...`	Additional arguments to pass to underlying function
`type`	The type of generation to perform. Uniform descends each level in a uniform manner (meaning all segments get replaced) whereas the random generation will randomly select segments to replace during each epoch.
`old.seed`	(Internal) Previous seed used to generate pattern
`new.seed`	(Internal) Next seed used to generate pattern
`pattern`	(Internal) Available patterns to use
`idx`	(Internal) Index of current iteration
`epoch`	(Internal) Current epoch

Value

An xts object containing a time series of values representing asset prices

Author(s)

Brian Lee Yung Rowe

References

M. Frame, B. Mandelbrot, N. Neger. Fractal Geometry. 2009. http://classes.yale.edu/fractals/

Examples

```
data(generators)
series <- fractal(sampleInitiators, sampleGenerators, count=10)

# View the results of a single iteration using the second pattern
series <- fractal(sampleInitiators, sampleGenerators, epochs=1, only=2)
```

---

getPortfolioPrices *Generate portfolio prices using the fractal process*

Description

This function will construct a portfolio of asset returns based on the time range specified or the number of 'observations' requested. The resulting time series will be based on the specified calendar, as defined by getTradingDates that uses the timeDate package under the hood.

Usage

```
getPortfolioPrices(symbols, obs = NULL, end = Sys.Date(), start = NULL, calendar = holidayNYSE, seeds

getTradingDates(end, start = NULL, obs = NULL, calendar = holidayNYSE)
```

Arguments

symbols	The names of the assets to generate prices for. This determines the total number of time series generated.
end	The last date in the time series
start	The starting date of the time series. All non-business days are removed in the resulting range. Either start or obs must be set.
obs	The total number of points to generate. The dates will follow a business day calendar as defined by timeDate, defaulting to NYSE. Either start or obs must be set.
calendar	The business day calendar to use. Defaults to NYSE.
seeds	A list of initiators to use for generating the time series

patterns	A list of generators to use for generating the time series
...	Additional arguments to send to the fractal generator
type	The type of fractal process to use. Defaults to uniform.

Details

The main entry point is getPortfolioPrices, which generates a TxM xts object based on the symbols provided. Prices generated by this function can be used in risk modeling, as a substitute for brownian motion in Monte Carlo simulations, and backtesting applications. Studying fractal generation of time series can be accomplished more directly by calling `fractal`.

In addition to the arguments above, it is necessary to pass the appropriate arguments to the the underlying fractal call. This includes passing in a seed and generator patterns. If none are provided predefined sets will be used, although users of this package are encouraged to create their own initiators and generators.

The getTradingDates function is a utility to generate proper business days for a given calendar. This is used to be compatible with other applications that load actual asset data.

Value

An xts object with either obs rows or points in the range [start,end] and a time series for each symbol provided.

Note

In the future, it may be possible to generate time series with an explicit R/S value or Hurst exponent.

Author(s)

Brian Lee Yung Rowe

Examples

```
data(generators)
ps <- getPortfolioPrices('IBM', '2009-02-24',obs=10,
  seeds=sampleInitiators, patterns=sampleGenerators)

getTradingDates('2009-02-26', obs=10)
```

---

plotReturns *Plot asset prices and returns for fractal analysis*

Description

This is a convenience function for studying the generated time series by the fragtalrock package. Given a time series of prices, plotReturns will plot both the original time series of prices and the returns series. This is a useful visual aid in determining the utility of the simulated time series.

Usage

```
plotReturns(series, ...)
```

Arguments

series	A time series
...	Additional arguments to pass to plot

Value

Invisibly returns the original series

Author(s)

Brian Lee Yung Rowe

Examples

```
data(generators)
ps <- fractal(sampleInitiators, sampleGenerators, epochs=3)
plotReturns(ps)
```

---

process *Generate a time series based on stochastic processes*

Description

A collection of functions to produce time series using stochastic processes.

Usage

```
ou.process(theta, mu = 0, sigma = 1, initial=mu, end = Sys.Date(), start = NULL, obs = NULL)
```

Arguments

theta	Rate of dissipation
mu	Mean
sigma	Volatility
initial	Initial value
end	The end date
start	The starting date
obs	Number of observations to produce

Details

The 'ou.process' function generates a mean-reverting time series according to the Ornstein-Uhlenbeck process.

Value

An xts object containing a time series of values representing asset prices whose evolution is defined by the given process.

Author(s)

Brian Lee Yung Rowe

Examples

```
series <- ou.process(1, 1.2, 0.3, obs=50)
```

---

sampleGenerators *Sample patterns for fractal generation of time series*

Description

This is a sample set of generators for creating time series using the fractalrock package.

Usage

```
sampleGenerators
```

Format

A list of available patterns to use as generators

Source

Inspiration

---

sampleInitiators *Sample seed for fractal generation of time series*

Description

This is a sample seed aka intiator for creating time series using the fractalrock package.

In the future, multiple intiators may be included.

Usage

```
sampleInitiators
```

Format

A matrix representing xy coordinates for the seed

Source

Inspiration

Index