Package ‘mvDFA’

June 15, 2023

Type Package

Title Multivariate Detrended Fluctuation Analysis

Description This R package provides an implementation of multivariate extensions of a well-known fractal analysis technique, Detrended Fluctuations Analysis (DFA; Peng et al., 1995<doi:10.1063/1.166141>), for multivariate time series: multivariate DFA (mvDFA). Several coefficients are implemented that take into account the correlation structure of the multivariate time series to varying degrees. These coefficients may be used to analyze long memory and changes in the dynamic structure that would by univariate DFA. Therefore, this R package aims to extend and complement the original univariate DFA (Peng et al., 1995) for estimating the scaling properties of nonstationary time series.

Version 0.0.4

Author Julien Patrick Irmer [aut, cre, cph]
(<https://orcid.org/0000-0002-7544-6483>),
Sebastian Wallot [aut, ctb]

Maintainer Julien Patrick Irmer <jirmer@psych.uni-frankfurt.de>

URL https://github.com/jpirmer/mvDFA

BugReports https://github.com/jpirmer/mvDFA/issues

License GPL-3

Imports longmemo, stats, pbapply, deSolve, RobPer, mvtnorm, pracma

Encoding UTF-8

RoxygenNote 7.2.3

Suggests knitr, rmarkdown

VignetteBuilder knitr

NeedsCompilation no

Repository CRAN

Date/Publication 2023-06-15 17:40:02 UTC

R topics documented:

DFA 2
mvDFA 3
print.DFA 5
print.mvDFA 5
simulate_cMTS 6
simulate_Lorenz_noise 7
simulate_MTS_mixed_white_pink_brown 9

Index **11**

DFA *Analyze univariate time series and estimate long memory using Detrended Fluctuations Analysis (DFA; Peng et al., 1995)*

Description

Analyze univariate time series and estimate long memory using Detrended Fluctuations Analysis (DFA; Peng et al., 1995)

Usage

```
DFA(X, steps = 50, brownian = FALSE, degree = 1, verbose = TRUE, cores = 1)
```

Arguments

`X`	Univariate time series.
`steps`	Maximum number of window sizes. These are spread logarithmically. If time series is short and steps is large, fewer window sizes are drawn. Default to `50`.
`brownian`	Indicator whether time series is assumed to be brownian (i.e. variance increases proportional to time)
`degree`	The maximum order of the detrending polynomial in the segments. This influences the smallest window size `minS` such that `minS` = `degree` + `2`.
`verbose`	Indicator whether additional info should be printed. Default to `TRUE`.
`cores`	Number of cores used in computation. Default to `1`.

Value

Returns list of Root Mean Squares per window size `RMS_s`, the window sizes `S` and the estimated long memory coefficient `L` - the Hurst Exponent.

References

Peng, C. K., Havlin, S., Stanley, H. E., & Goldberger, A. L. (1995). Quantification of scaling exponents and crossover phenomena in nonstationary heartbeat time-series. Chaos, 5, 82–87. <doi:10.1063/1.166141>

Examples

```
X <- rnorm(500) # generate Gaussian white noise (i.i.d. standard normal variables)
DFA(X = X, steps = 5) # steps = 5 is only for demonstration,
                        # use many steps instead, e.g. steps = 50!
```

---

mvDFA *Analyze multivariate correlated time series and estimate long memory by the extension of the using univariate Detrended Fluctuations Analysis (DFA; Peng et al., 1995) to multivariate time series: mvDFA*

Description

Analyze multivariate correlated time series and estimate long memory by the extension of the using univariate Detrended Fluctuations Analysis (DFA; Peng et al., 1995) to multivariate time series: mvDFA

Usage

```
mvDFA(
  X,
  steps = 50,
  degree = 1,
  verbose = FALSE,
  cores = 1,
  covlist = FALSE,
  brownian = FALSE
)
```

Arguments

`X`	Matrix or data.frame containing the time series in long format.
`steps`	Maximum number of window sizes. These are spread logarithmically. If time series is short and steps is large, fewer window sizes are drawn. Default to `50`. The dimensions (`ncol(X)`) and the `degree` influence the smallest possible window size.
`degree`	The maximum order of the detrending polynomial in the segments. This influences the smallest window size `minS` such that `minS` = `d` + `degree` + `2`, where `d` is the dimension of the time series.
`verbose`	Indicator whether additional info should be printed. Default to `TRUE`.
`cores`	Number of cores used in computation. Default to `1`.
`covlist`	Indicator whether covariance of the time series per window size should be saved in a list.
`brownian`	Indicator whether time series are assumed to be brownian (i.e. variance increases proportional to time)

Value

An object of class `mvDFA` containing long memory coefficients (Hurst exponents) and corresponding further informations:

`Ltot`	the estimated long memory coefficient for the multivariate time series using the total variance approach
`Lgen`	the generalized approach
`Lfull`	the average covariance approach
`LmeanUni`	average Hurst exponent across all time series
`univariate_DFA`	univariate Hurst exponents
`R2tot`	R-squared of total variance approach in regression of log10(RMS) vs log10(S)
`R2gen`	R-squared of generalized variance approach in regression of log10(RMS) vs log10(S)
`R2full`	R-squared of covariance approach in regression of log10(RMS) vs log10(S)
`R2meanUni`	average R-squared across all time series in regression of log10(RMS) vs log10(S)
`R2univariate_DFA`	R-squares of single time series approach in regression of log10(RMS) vs log10(S)
`RMS_tot`	a list of Root Mean Squares per window size corresponding to the total variance approach
`RMS_gen`	a list of Root Mean Squares per window size corresponding to the total generalized approach
`Cov_RMS_s`	a list of Root Mean Squares per window size corresponding to the covariance approach
`S`	window sizes used
`CovRMS_list`	a list of covariance matrices per `S` may be returned

References

Peng, C. K., Havlin, S., Stanley, H. E., & Goldberger, A. L. (1995). Quantification of scaling exponents and crossover phenomena in nonstationary heartbeat time-series. Chaos, 5, 82–87. <doi:10.1063/1.166141>

Examples

```
Sigma <- matrix(.5, 3, 3); diag(Sigma) <- 1
# generate correlated Gaussian white noise (i.i.d. multivariate normal variables)
X <- mvtnorm::rmvnorm(n = 500, sigma = Sigma)
mvDFA(X = X, steps = 5) # steps = 5 is only for demonstration,
                        # use many steps instead, e.g. steps = 50!
```

print.DFA *print object of class DFA*

Description

print object of class DFA

Usage

```
## S3 method for class 'DFA'
print(x, ...)
```

Arguments

x	object of class DFA to print.
...	further parameters.

Value

Truncates the output printed into the console of objects of class DFA, but does not change object itself.

print.mvDFA *print object of class mvDFA*

Description

print object of class mvDFA

Usage

```
## S3 method for class 'mvDFA'
print(x, ...)
```

Arguments

x	object of class DFA to print.
...	further parameters.

Value

Truncates the output printed into the console of objects of class mvDFA, but does not change object itself.

simulate_cMTS *Approximate correlated time series with given Hurst Exponent*

Description

Approximation of correlated time series with given "Hurst" exponents. Internally `longmemo::simFGN0` or `longmemo::simFGN.fft` are used which simulate Gaussian series by generating fractional ARIMA(0,h,0) models (with $h=H-1/2$, `longmemo::FGN0`), or fractional Gaussian noise `longmemo::FGN.fft`. We cautiously note that we use empirical scaling (i.e., the variances are scaled to be 1 in the sample not the population), hence the between sample variance may be underrepresented. We further note that the covariance estimates for correlated time series (not using increments) is unstable.

Usage

```
simulate_cMTS(
  N,
  H,
  Sigma,
  simulation_process = "FGN0",
  decomposition = "chol",
  cor_increments = TRUE,
  X0 = rep(0, ncol(Sigma))
)
```

Arguments

`N` Length of Times Series

`H` Hurst Exponents for `d` time series. These are then mixed using one of two different decompositions of the given covariance matrix `Sigma`.

`Sigma` Positive semi definite covariance matrix of desired multi-dimensional time series.

`simulation_process` The simulation process passed to the `longmemo::sim...` function. Can either be `longmemo::simFGN.fft` (using FFT) or `longmemo::simFGN0` (using fractional gaussian processes). FGN0 looks more like `rnorm`, when `H=0.5`. `DEFAULT` to `"FGN0"`. Use `simulation_process="FGN.fft"` to use the FFT based version.

`decomposition` Character whether the Cholesky decomposition `"chol"` (or `"cholesky"`) should be used or whether the eigen decomposition should be used (`decomposition = "eigen"`). `DEFAULT` to `"chol"`.

`cor_increments` Logical, whether to correlate the increments or the time series themselves. Default to `TRUE`.

`X0` Starting values for the time series if increments are correlated. Default to `rep(0, ncol(Sigma))`, i.e., the zero vector of required length.

Value

Returns a multivariate correlated time series with covariance matrix Sigma. The Hurst exponents are only approximating the univariate ones, since they result from mixed time series. Uncorrelated time series keep their univariate Hurst exponents H.

Examples

```
Sigma <- matrix(.5, 3, 3); diag(Sigma) <- c(1,2,3)
data <- simulate_cMTS(N = 10^5, Sigma = Sigma, H = c(.2, .5, .7),
                       cor_increments = TRUE)
cov(data)
cov(apply(data,2,diff))
```

---

simulate_Lorenz_noise *Simulate the Lorenz System with noise*

Description

Simulate the Lorenz System with noise

Usage

```
simulate_Lorenz_noise(
  N = 1000,
  delta_t = NULL,
  tmax = 50,
  X0 = 0,
  Y0 = 1,
  Z0 = 1,
  sdX = NULL,
  sdY = NULL,
  sdZ = NULL,
  sdnoiseX,
  sdnoiseY,
  sdnoiseZ,
  s = 10,
  r = 28,
  b = 8/3,
  naive = FALSE,
  return_time = TRUE
)
```

Arguments

N	Length of Times Series
delta_t	Step size for time scale. If NULL this is derived using N and tmax. DEFAULT to NULL.

`tmax`	Upper bound of the time scale. This argument is ignored if `delta_t` is provided. `DEFAULT` to `50`.
`X0`	Initial value for X at t=0. `DEFAULT` to 0.
`Y0`	Initial value for Y at t=0. `DEFAULT` to 1.
`Z0`	Initial value for Z at t=0. `DEFAULT` to 1.
`sdX`	Use this argument to rescale the X-coordinate to have the desired standard deviation (exactly). This is ignored if set to `NULL`. `DEFAULT` to `NULL`.
`sdY`	Use this argument to rescale the Y-coordinate to have the desired standard deviation (exactly). This is ignored if set to `NULL`. `DEFAULT` to `NULL`.
`sdZ`	Use this argument to rescale the Z-coordinate to have the desired standard deviation (exactly). This is ignored if set to `NULL`. `DEFAULT` to `NULL`.
`sdnoiseX`	Standard deviation of Gaussian noise of X-coordinate. If set to `0`, no noise is created.
`sdnoiseY`	Standard deviation of Gaussian noise of Y-coordinate. If set to `0`, no noise is created.
`sdnoiseZ`	Standard deviation of Gaussian noise of Z-coordinate. If set to `0`, no noise is created.
`s`	s-parameter of the Lorenz ODE. See Vignette for further details. `DEFAULT` to `10`, which is the original value chosen by Lorenz.
`r`	r-parameter of the Lorenz ODE. See Vignette for further details. `DEFAULT` to `28`, which is the original value chosen by Lorenz.
`b`	b-parameter of the Lorenz ODE. See Vignette for further details. `DEFAULT` to `8/3`, which is the original value chosen by Lorenz.
`naive`	Logical whether naive calculation should be used. `DEFAULT` to `FALSE`.
`return_time`	Logical whether the time-coordinate should be included in the returned `data.frame`. `DEFAULT` to `TRUE`.

Value

Returns a three dimensional time series as `data.frame` following the Lorenz system (Lorenz, 1963, <doi:10.1175/1520-0469(1963)020<0130:DNF>2.0.CO;2>).

References

Lorenz, E. N. (1963). Deterministic nonperiodic flow. Journal of atmospheric sciences, 20(2), 130-141. <doi:10.1175/1520-0469(1963)020<0130:DNF>2.0.CO;2>

simulate_MTS_mixed_white_pink_brown

Approximate correlated time series from white, pink and brown noise from independent realization of normal variables

Description

Approximation of correlated time series representing "white", "pink" or "brown" noise from independent realization of normal variates Internally normal variables are simulated using `rnorm` and then are cumulated for white or brown noise and we use `RobPer::TK95` for the generation of pink noise. We cautiously note that we use empirical scaling (i.e., the variances are scaled to be 1 in the sample not the population), hence the between sample variance may be underrepresented. We further note that the covariance estimates for correlated time series (not using increments) is unstable.

Usage

```
simulate_MTS_mixed_white_pink_brown(
  N,
  Sigma,
  process = "white",
  decomposition = "chol",
  cor_increments = TRUE,
  X0 = rep(0, ncol(Sigma))
)
```

Arguments

`N`	Length of multivariate Times Series
`Sigma`	Positive semi definite covariance matrix the increments of desired multi dimensional time series. The dimensionality of Sigma sets the dimension of the time series. The variance scale the time. If the variances are all 1, then each data point represents one unit of time.
`process`	Type of process. Can either be "white", "brown" or "pink". Default to "white". If process is a vector, a mixture of the three process is generated, correlated by Sigma.
`decomposition`	Character whether the Cholesky decomposition `"chol"` (or `"cholesky"`) should be used or whether the eigen decomposition should be used (`decomposition = "eigen"`). `DEFAULT` to `"chol"`.
`cor_increments`	Logical, whether to correlate the increments or the time series themselves. Default to `TRUE`.
`X0`	Starting values for the time series if increments are correlated. Default to `rep(0, ncol(Sigma))`, i.e., the zero vector of required length.

Value

Returns a multivariate correlated time series with covariance matrix ‘Sigma‘. The Hurst exponents are only approximating the univariate ones, since they result from mixed time series. Here, a mixture of "white", "pink" and "brown" noise can be chosen from. Uncorrelated time series keep their univariate Hurst exponent ‘H‘.

Examples

```
Sigma <- matrix(.5, 3, 3); diag(Sigma) <- c(1,2,3)
data <- simulate_MTS_mixed_white_pink_brown(N = 10^5, Sigma = Sigma,
                                            process = c("white", "pink", "brown"),
                                            cor_increments = FALSE)
cov(data) # unstable covariances
cov(apply(data,2,diff))
```

Index

DFA, 2

mvDFA, 3

print.DFA, 5
print.mvDFA, 5

simulate_cMTS, 6
simulate_Lorenz_noise, 7
simulate_MTS_mixed_white_pink_brown, 9