Package 'Rmfrac'

August 6, 2025

Type Package

Title Simulation and statistical analyses of multifractional processes

Version 0.1.0

Author Andriy Olenko, Nemini Samarakoon

Maintainer Nemini Samarakoon <neminisamarakoon95@gmail.com>

Description Simulation of multifractional processes. Estimation of Hurst functions and local fractal dimension. Clustering realisations based on the Hurst functions and several functions to estimate and plot geometric statistics of the processes and time series.

License GPL-3

Encoding UTF-8

Imports parallel, parallelly, ggplot2(>= 3.5.0), zoo, matrixStats, proxy, rlang, stats, foreach, doParallel, plotly, shiny, graphics, shinycssloaders, xts, fields

LazyData false

RoxygenNote 7.3.1

Roxygen list(markdown = TRUE)

URL https://github.com/Nemini-S/Rmfrac

BugReports https://github.com/Nemini-S/Rmfrac/issues

Contents

`Rmfrac-package` *Rmfrac: Simulation and analyses of multifractional processes*

Description

A collection of tools for simulating, analysing and visualising multifractional processes and time series. The package includes built-in estimation techniques for the Hurst function, Local Fractal Dimension and several other geometric statistics. It provides highly customisable plotting functions for simulated realisations, user-provided time series and their statistics.

Features

- Simulation of Brownian motion, fractional Brownian motion, fractional Gaussian noise, Brownian bridge and fractional Brownian bridge
- Simulation of Gaussian Haar-based multifractional process (GHBMP)
- Estimation of Hurst function and Local Fractal Dimension
- Customisable plotting functions for GHBMP and user provided time series with estimates of Hurst function and Local Fractal Dimension
- Estimation and visualisation of geometric statistics using realisations of stochastic processes and time series. Clustering based on the Hurst function, sojourn measure, excursion area, etc.
- An interactive Shiny application that provides options to explore and visualise the core functionalities of the package through simulations and user-provided time series.

Author(s)

- Andriy Olenko ORCID a.olenko@latrobe.edu.au
- Nemini Samarakoon ORCID neminisamarakoon95@gmail.com

See Also

Useful links:

- https://github.com/Nemini-S/Rmfrac
- Report bugs at https://github.com/Nemini-S/Rmfrac/issues

`Bbridge` *Simulation of Brownian bridge*

Description

This function simulates a realisation of the Brownian bridge over the time interval `[t_start,t_end]` which has the initial value `x_start` and terminates at `x_end` with `N` time steps.

Usage

```
Bbridge(x_end, t_end, x_start = 0, t_start = 0, N = 1000, plot = FALSE)
```

Arguments

`x_end`	Value of the process at the terminating time point.
`t_end`	Terminal time point.
`x_start`	Value of the process at the initial time point.
`t_start`	Initial time point.
`N`	Number of sub-intervals the interval `[t_start,t_end]` is split into. Default set to 1000.
`plot`	Logical: If `TRUE`, the realisation of the Brownian bridge is plotted.

Value

A data frame where the first column is `t` and second column is simulated values of the realisation of Brownian bridge.

References

Bianchi, S., Frezza, M., Pianese, A., Palazzo, A.M. (2022). Modelling H-Volatility with Fractional Brownian Bridge. In: Corazza, M., Perna, C., Pizzi, C., Sibillo, M. (eds) Mathematical and Statistical Methods for Actuarial Sciences and Finance. MAF 2022. Springer, Cham. doi:doi.org/10.1007/9783030996383_16.

See Also

`FBm`, `Bm`, `GHBMP`, `FBbridge`

Examples

```
Bbridge(x_end=2,t_end=1,plot=TRUE)
```

Bm *Simulation of Brownian motion*

Description

This function simulates a realisation of the Brownian motion over the time interval [t_start,t_end] with N time steps and initial value x_start.

Usage

```
Bm(x_start = 0, t_start = 0, t_end = 1, N = 1000, plot = FALSE)
```

Arguments

x_start	Value of the process at the initial time point (additive constant mean).
t_start	Initial time point.
t_end	Terminal time point.
N	Number of sub-intervals the interval [t_start,t_end] is split into. Default set to 1000.
plot	Logical: If TRUE, the realisation of the Brownian motion is plotted.

Value

A data frame where the first column is t and second column is simulated values of the realisation of Brownian motion with added constant mean.

See Also

GHBMP, FBm, FGn, Bbridge, FBbridge

Examples

```
Bm(x_start=0,t_start=0,t_end=2,plot=TRUE)
```

cov_GHBMP *Covariance of Gaussian Haar-based multifractional processes*

Description

Computes the theoretical covariance matrix of a Gaussian Haar-based multifractional process.

Usage

```
cov_GHBMP(
  t,
  H,
  J = 8,
  theta = NULL,
  plot = FALSE,
  num.cores = availableCores(omit = 1)
)
```

Arguments

`t`	Time point or time sequence on the interval $[0, 1]$.
`H`	Hurst function $H(t)$ which depends on `t`.
`J`	Positive integer. For large J values could be rather time consuming. Default is set to 8.
`theta`	Optional: Smoothing parameter.
`plot`	Logical: If TRUE, a 3D surface plot of the covariance function is plotted.
`num.cores`	Number of cores to set up the clusters for parallel computing.

Details

To make it comparable with the empirical covariance function the same smoothing parameter `theta` can be used if needed.

Value

An $m \times m$ matrix, where m is the length of `t`.

References

Ayache, A., Olenko, A. and Samarakoon, N. (2025). On Construction, Properties and Simulation of Haar-Based Multifractional Processes. doi:10.48550/arXiv.2503.07286. (submitted).

See Also

`GHBMP`

Examples

```
t <- seq(0,1,by=0.01)
H <- function(t) {return(0.5-0.4* sin(6*3.14*t))}

#Smoothed covariance function
cov_GHBMP(t,H,theta=0.1,plot=TRUE)

#Non-smoothed covariance function
cov_GHBMP(t,H,plot=TRUE)
```

`cross_mean` *Mean time between crossings*

Description

Computes the mean duration between crossings of a time series at a specified constant level for the provided time interval or its sub-interval.

Usage

```
cross_mean(X, A, subI = NULL, plot = FALSE)
```

Arguments

X	Data frame where the first column is a time sequence t and the second one is the values of the time series $X(t)$.
A	Constant level as a numeric value.
subI	Time sub-interval is a vector, where the lower bound is the first element and the upper bound is the second. Optional: If provided mean crossing times for the sub-interval is returned, otherwise the whole time interval is considered.
plot	Logical: If TRUE, the time series, the constant level and crossing points are plotted.

Value

The estimated mean time between crossings. If plot=TRUE, the time series with the constant level and crossing points are plotted.

See Also

cross_T, cross_rate

Examples

```
t <- seq(0,1,length=100)
TS <- data.frame("t"=t,"X(t)"=rnorm(100))
cross_mean(TS,0.1,subI=c(0.2,0.8),plot=TRUE)
```

---

cross_rate *Crossing rate*

Description

Computes the rate at which a time series crosses a specific constant level for the provided time interval or its sub-interval.

Usage

```
cross_rate(X, A, subI = NULL, plot = FALSE)
```

Arguments

X	Data frame where the first column is a time sequence t and the second one is the values of the time series $X(t)$.
A	Constant level as a numeric value.
subI	Time sub-interval as a vector, where the lower bound is the first element and the upper bound is the second. Optional: If provided crossing rate for the sub-interval is returned, otherwise the whole time interval is considered.
plot	Logical: If TRUE, the time series, the constant level and crossing points are plotted.

Value

The crossing rate, which gives average number of crossings per time unit. If `plot=TRUE`, the time series with the constant level and crossing points are plotted.

See Also

`cross_T`, `cross_mean`

Examples

```
t <- seq(0,1,length=100)
TS <- data.frame("t"=t,"X(t)"=rnorm(100))
cross_rate(TS,0.1,subI=c(0.2,0.8),plot=TRUE)
```

`cross_T` *Estimated crossing times*

Description

Computes the estimated t value(s), in which a time series crosses a specific constant level for the provided time interval or its sub-interval.

Usage

```
cross_T(X, A, subI = NULL, plot = FALSE, vline = FALSE)
```

Arguments

`X`	Data frame where the first column is a time sequence t and the second one is the values of the time series $X(t)$.
`A`	Constant level as a numeric value.
`subI`	Time sub-interval as a vector, where the lower bound is the first element and the upper bound is the second. Optional: If provided level crossing times of the sub-interval is returned, otherwise the whole time interval is considered.
`plot`	Logical: If `TRUE`, the time series, the constant level and corresponding t values are plotted.
`vline`	Logical: If `TRUE`, a vertical line is plotted at the crossing point(s).

Value

The estimated crossing times at a given level. If `plot=TRUE`, the time series with the constant level crossing and level crossing times are plotted.

See Also

`cross_rate` `cross_mean`

Examples

```
t <- seq(0,1,length=100)
TS <- data.frame("t"=t,"X(t)"=rnorm(100))
cross_T(TS,0.1,subI=c(0.2,0.8),plot=TRUE,vline=TRUE)
```

est_cov *Empirical covariance function*

Description

Computes the empirical covariance function of a process, for each pair of time points in the time sequence using M realisations of the process.

Usage

```
est_cov(X, theta = 0.1, plot = FALSE)
```

Arguments

X	A data frame where the first column is the time sequence and the remaining columns are the values of each realisation of the process.
theta	Smoothing parameter.
plot	Logical: If TRUE, a 3D surface plot of the covariance function is plotted.

Details

The smoothing parameter `theta` can help to better visualise changes between neighbour estimated values.

Value

An $m \times m$ matrix, where m is the number of time points. Each element represents the estimated value of covariance function for the corresponding time points. Time points are arranged in ascending order.

Examples

```
#Matrix of empirical covariance estimates of the GHBMP with Hurst function H.
t <- seq(0,1,by=(1/2)^8)
H <- function(t) {return(0.5-0.4*sin(6*3.14*t))}
#Only 5 realisations of GHBMP are used in this example to reduce the computational time.
X.t <- replicate(5, GHBMP(t,H), simplify = FALSE)
X <- do.call(rbind, lapply(X.t, function(df) df[, 2]))
Data <- data.frame(t,t(X))
cov.mat <- est_cov(Data,theta=0.2,plot=TRUE)
cov.mat
```

exc_Area *Excursion area*

Description

Computes the excursion area where a time series $X(t)$ is greater or lower than the constant level A for the provided time interval or its sub-interval.

Usage

```
exc_Area(X, A, N = 10000, level = "greater", subI = NULL, plot = FALSE)
```

Arguments

X	Data frame where the first column is a time sequence t and the second one is the values of the time series $X(t)$.
A	Constant level as a numeric value.
N	Number of steps the time interval (or time sub-interval) is split into. Default set to 10000.
level	A vector of character strings which specifies whether the excursion area is required for X, "greater" or "lower" than A. Default set to "greater".
subI	Time sub-interval is a vector, where the lower bound is the first element and the upper bound is the second. Optional: If provided, the excursion area for the sub-interval is returned, otherwise the whole time interval is considered.
plot	Logical: If TRUE, the time series, constant level and excursion area are plotted.

Value

Excursion area. If plot=TRUE, the time series, the constant level and excursion area are plotted.

See Also

sojourn

Examples

```
t <- seq(0,1,length=1000)
TS <- data.frame("t"=t,"X(t)"=rnorm(1000))
exc_Area(TS,0.8,level='lower',subI=c(0.5,0.8),plot=TRUE)
```

FBbridge *Simulation of fractional Brownian bridge*

Description

This function simulates a realisation of the fractional Brownian bridge for a provided Hurst parameter over the time interval `[t_start,t_end]` which has the initial value `x_start` and terminates at `x_end` with `N` time steps.

Usage

```
FBbridge(H, x_end, t_end, x_start = 0, t_start = 0, N = 1000, plot = FALSE)
```

Arguments

`H`	Hurst parameter which lies between 0 and 1.
`x_end`	Value of the process at the terminating time point.
`t_end`	Terminal time point.
`x_start`	Value of the process at the initial time point.
`t_start`	Initial time point.
`N`	Number of sub-intervals the interval `[t_start,t_end]` is split into. Default set to 1000.
`plot`	Logical: If `TRUE`, the realisation of the fractional Brownian bridge is plotted.

Value

A data frame where the first column is `t` and second column is simulated values of the realisation of fractional Brownian bridge.

References

Bianchi, S., Frezza, M., Pianese, A., Palazzo, A.M. (2022). Modelling H-Volatility with Fractional Brownian Bridge. In: Corazza, M., Perna, C., Pizzi, C., Sibillo, M. (eds) Mathematical and Statistical Methods for Actuarial Sciences and Finance. MAF 2022. Springer, Cham. doi:doi.org/10.1007/9783030996383_16.

See Also

`FBm`, `Bm`, `GHBMP`, `Bbridge`

Examples

```
FBbridge(H=0.5,x_end=2,t_end=1,plot=TRUE)
```

FBm *Simulation of fractional Brownian motion*

Description

This function simulates a realisation of the fractional Brownian motion over the time interval [t_start,t_end] for a provided Hurst parameter.

Usage

```
FBm(H, x_start = 0, t_start = 0, t_end = 1, N = 1000, plot = FALSE)
```

Arguments

H	Hurst parameter which lies between 0 and 1.
x_start	Value of the process at the initial time point (additive constant mean).
t_start	Initial time point.
t_end	Terminal time point.
N	Number of sub-intervals the interval [t_start,t_end] is split into. Default set to 1000.
plot	Logical: If TRUE, the realisation of the fractional Brownian motion is plotted.

Value

A data frame where the first column is t and second column is simulated values of the realisation of fractional Brownian motion with added constant mean.

References

Banna, O., Mishura, Y., Ralchenko, K., & Shklyar, S. (2019). Fractional Brownian motion: Approximations and Projections. John Wiley & Sons. doi:doi/10.1002/9781119476771.app3.

See Also

FGn, Bm, GHBMP, Bbridge, FBbridge

Examples

```
FBm(H=0.5,plot=TRUE)
```

FGn — Simulation of fractional Gaussian noise

Description

This function simulates a realisation of the fractional Gaussian noise over the time interval `[t_start,t_end]` for a provided Hurst parameter.

Usage

```
FGn(H, t_start = 0, t_end = 1, n = 1000, plot = FALSE)
```

Arguments

`H`	Hurst parameter which lies between 0 and 1.
`t_start`	Initial time point.
`t_end`	Terminal time point.
`n`	Number of time points where the simulation is performed on the interval `[t_start,t_end]`. Default set to 1000.
`plot`	Logical: If `TRUE`, the realisation of the fractional Gaussian noise is plotted.

Value

A data frame where the first column is `t` and second column is simulated values of the realisation of fractional Gaussian noise.

References

Banna, O., Mishura, Y., Ralchenko, K., & Shklyar, S. (2019). Fractional Brownian motion: Approximations and Projections. John Wiley & Sons. doi:doi/10.1002/9781119476771.app3.

See Also

`FBm`, `Bm`, `GHBMP`, `Bbridge` , `FBbridge`

Examples

```
FGn(H=0.5,plot=TRUE)
```

GHBMP *Simulation of Gaussian Haar-based multifractional processes*

Description

This function simulates a realisation of a Gaussian Haar-based multifractional process at any time point or time sequence on the interval $[0, 1]$.

Usage

```
GHBMP(t, H, J = 15, num.cores = availableCores(omit = 1))
```

Arguments

t	Time point or time sequence on the interval $[0, 1]$.
H	Hurst function which depends on t $(H(t))$. See Examples for usage.
J	Positive integer. J is recommended to be greater than $\log_2(length(t))$. For large J values could be rather time consuming. Default is set to 15.
num.cores	Number of cores to set up the clusters for parallel computing.

Details

The following formula defined in Ayache, A., Olenko, A. & Samarakoon, N. (2025) was used in simulating Gaussian Haar-based multifractional process.

$X(t) := \sum_{j=0}^{+\infty} \sum_{k=0}^{2^j-1} \left(\int_0^1 (t-s)_+^{H_j(k/2^j)-1/2} h_{j,k}(s) ds \right) \varepsilon_{j,k},$

where

$\int_0^1 (t-s)_+^{H_{j,k}-\frac{1}{2}} h_{j,k}(s) ds = 2^{-jH_{j,k}} h^{[H_{j,k}]}(2^j t - k)$

with $h^{[\lambda]}(x) = \int_{\mathbb{R}} (x-s)_+^{\lambda-\frac{1}{2}} h(s) ds$. h is the Haar mother wavelet, j and k are positive integers, t is time, H is the Hurst function and $\varepsilon_{j,k}$ is a sequence of independent $\mathcal{N}(0,1)$ Gaussian random variables. For simulations, the truncated version of this formula with first summation up to J is used.

Value

A data frame of class "mp" where the first column is time moments t and second column is simulated values of $X(t)$.

Note

See Examples for the usage of constant, time-varying, piecewise or step Hurst functions.

References

Ayache, A., Olenko, A. and Samarakoon, N. (2025). On Construction, Properties and Simulation of Haar-Based Multifractional Processes. doi:10.48550/arXiv.2503.07286. (submitted).

See Also

Hurst, plot.mp, Bm, FBm, FGn, Bbridge , FBbridge

Examples

```
#Constant Hurst function
t <- seq(0,1,by=(1/2)^10)
H <- function(t) {return(0.4 +0*t)}
GHBMP(t,H)

#Linear Hurst function
t <- seq(0,1,by=(1/2)^10)
H <- function(t) {return(0.2+0.45*t)}
GHBMP(t,H)

#Oscillating Hurst function
t <- seq(0,1,by=(1/2)^10)
H <- function(t) {return(0.5-0.4*sin(6*3.14*t))}
GHBMP(t,H)

#Piecewise Hurst function
t <- seq(0,1,by=(1/2)^10)
H <- function(x) {
ifelse(x >= 0 & x <= 0.8, 0.375 * x + 0.2,
      ifelse(x > 0.8 & x <= 1,-1.5 * x + 1.7, NA))
}
GHBMP(t,H)
```

`hclust_hurst` *Hierarchical clustering*

Description

This function performs hierarchical clustering of realisations of multifractional processes based on the estimated Hurst functions.

Usage

```
hclust_hurst(
  X.t,
  k = NULL,
  h = NULL,
  dist.method = "euclidean",
  method = "complete",
  dendrogram = FALSE,
  N = 100,
  Q = 2,
  L = 2
)
```

Arguments

`X.t`	A list of data frames. In each data frame, the first column is a time sequence and the second gives the values of the multifractional process. See Examples for usage.

k	The desired number of clusters.
h	The height where the dendrogram should be cut into. Either k or h must be specified. If both are provided k is used.
dist.method	A string which specifies a registered distance from proxy::dist(). The default is "euclidean".
method	A string which specifies the hierarchical method used. Available methods are "ward.D", "ward.D2", "single", "complete", "average", "mcquitty", "median" and "centroid". The default method is "complete".
dendrogram	Logical: If TRUE the dendrogram is plotted indicating the clusters.
N	Argument used for the estimation of Hurst functions. Number of sub-intervals on which the estimation is performed on. Default is set to 100 sub-intervals.
Q	Argument used for the estimation of Hurst functions. Fixed integer greater than or equal to 2. Default is set to 2.
L	Argument used for the estimation of Hurst functions. Fixed integer greater than or equal to 2. Default is set to 2.

Details

The Hurst function of each realisation is estimated using the function Hurst and the smoothed Hurst estimates are used for the cluster analysis. The distances between smoothed Hurst estimates are computed by the dist.method provided and passed into the hclust for hierarchical clustering.

Value

An object list of class "hc_hurst" with print and plot methods. The list has following components:

cluster_info A data frame indicating the cluster number and distance to cluster center from each smoothed estimated Hurst function (item). Distance is obtained from the dist.method.

cluster A vector with cluster number of each item.

cluster_sizes Number of items in each cluster.

centers A data frame of cluster centers. Center obtained as the average of each smoothed estimated Hurst function in the cluster. Columns denote time points in which estimates were obtained. Row names denote cluster numbers.

smoothed_Hurst_estimates A data frame of smoothed Hurst estimates. Columns denote time points in which estimates were obtained. Rows denote estimates for each realisation.

raw_Hurst_estimates A list of data frames of raw Hurst estimates.

call Information about the input parameters used.

See Also

print.hc_hurst, plot.hc_hurst, kmeans_hurst

Examples

```
#Simulation of multifractional processes
t <- seq(0,1,by=(1/2)^10)
H1 <- function(t) {return(0.1+0*t)}
H2 <- function(t) {return(0.2+0.45*t)}
H3 <- function(t) {return(0.5-0.4*sin(6*3.14*t))}
X.list.1 <- replicate(3, GHBMP(t,H1),simplify = FALSE)
```

```
X.list.2 <- replicate(3, GHBMP(t,H2),simplify = FALSE)
X.list.3 <- replicate(3, GHBMP(t,H3),simplify = FALSE)
X.list <- c(X.list.1,X.list.2,X.list.3)

#Hierarchical clustering based on k=3 clusters with dendrogram plotted
HC <- hclust_hurst(X.list,k=3,dendrogram=TRUE)
print(HC)

#Plot of smoothed Hurst functions in each cluster with cluster centers
plot(HC,type ="ec")
```

`Hurst` *Statistical estimation of the Hurst function*

Description

This function computes statistical estimates for the Hurst function of multifractional processes.

Usage

```
Hurst(X, N = 100, Q = 2, L = 2)
```

Arguments

`X`	Data frame where the first column is a time sequence and the second the values of the multifractional process. For reliable estimates the data frame should be of at least 500 data points.
`N`	Number of sub-intervals on which the estimation is performed on. Default is set to 100 sub-intervals.
`Q`	Fixed integer greater than or equal to 2. Default is set to 2.
`L`	Fixed integer greater than or equal to 2. Default is set to 2.

Details

Statistical estimation of the Hurst function is done based on the results of Ayache, A., & Bouly, F. (2023). The estimator is built through generalized quadratic variations of the process associated with its increments.

Value

A data frame of where the first column is a time sequence and second column is estimated values of the Hurst function.

References

Ayache, A. and Bouly, F. (2023). Uniformly and strongly consistent estimation for the random Hurst function of a multifractional process. Latin American Journal of Probability and Mathematical Statistics, 20(2):1587–1614. doi:10.30757/alea.v2060.

See Also

`LFD`, `plot.mp`, `plot_tsest`, `plot.H_LFD`

Examples

```
#Hurst function of a multifractional process simulated using GHBMP function
T <- seq(0,1,by=(1/2)^10)
H <- function(t) {return(0.5-0.4*sin(6*3.14*t))}
X <- GHBMP(T,H)
Hurst(X)
```

H_LFD *Creates objects of class* H_LFD

Description

For user provided time series creates objects of class "H_LFD" with the Hurst function estimated using Hurst,local fractal dimension estimated using LFD and smoothed estimated Hurst function and LFD added.

Usage

```
H_LFD(X, N = 100, Q = 2, L = 2)
```

Arguments

X	Data frame where the first column is a time sequence t and the second one is the values of the time series $X(t)$. For accurate estimated Hurst functions, X should be of at least 500 data points.
N	Argument used for the estimation of Hurst functions and LFD. Number of sub-intervals on which the estimation is performed on. Default is set to 100 sub-intervals.
Q	Argument used for the estimation of Hurst functions and LFD. Fixed integer greater than or equal to 2. Default is set to 2.
L	Argument used for the estimation of Hurst functions and LFD. Fixed integer greater than or equal to 2. Default is set to 2.
...	Other arguments.

Value

The return from H_LFD is an object list of class "H_LFD" with the following components:

Raw_Hurst_estimates A data frame of where the first column is a time sequence and second column is estimated values of the Hurst function.

Smoothed_Hurst_estimates A data frame of where the first column is a time sequence and second column is smoothed estimates of the Hurst function.

LFD_estimates A data frame of where the first column is a time sequence and second column is Local fractal dimension estimates.

LFD_Smoothed_estimates A data frame of where the first column is a time sequence and second column is smoothed estimates of Local fractal dimension.

Data User provided time series.

See Also

plot.H_LFD Hurst, LFD

Examples

```
TS <- data.frame("t"=seq(0,1,length=1000),"X(t)"=rnorm(1000))
Object <- H_LFD(TS,N=100,Q=2,L=2)
plot(Object,Raw_Est_H=TRUE,Smooth_Est_H=TRUE,LFD_EST=TRUE,LFD_Smooth_Est=TRUE)
```

---

kmeans_hurst *K-means clustering*

Description

This function performs k-means clustering of realisations of multifractional processes based on the estimated Hurst functions.

Usage

```
kmeans_hurst(X.t, k, ..., N = 100, Q = 2, L = 2)
```

Arguments

X.t	A list of data frames. In each data frame, the first column is a time sequence and the second gives the values of the multifractional process. See Examples for usage.
k	The desired number of clusters.
...	Optional arguments: iter.max, nstart and algorithm. Refer kmeans.
N	Argument used for the estimation of Hurst functions. Number of sub-intervals on which the estimation is performed on. Default is set to 100 sub-intervals.
Q	Argument used for the estimation of Hurst functions. Fixed integer greater than or equal to 2. Default is set to 2.
L	Argument used for the estimation of Hurst functions. Fixed integer greater than or equal to 2. Default is set to 2.

Details

The Hurst function of each realisation is estimated using Hurst. The smoothed Hurst estimates are used for k-means clustering in kmeans. The Hartigan and Wong algorithm is used as the default k-means clustering algorithm.

Value

An object list of class "k_hurst" with print and plot methods. The list has following components:

cluster_info A data frame indicating the cluster number and euclidean distance to cluster center of each smoothed estimated Hurst function (item)

cluster A vector of cluster number of each item.

cluster_sizes Number of item in each cluster.

`centers` A data frame of cluster centers. Center obtained as the average of each smoothed estimated Hurst function in the cluster. Columns denote time points in which estimates were obtained. Row names denote cluster numbers.

`smoothed_Hurst_estimates` A data frame of smoothed Hurst estimates. Columns denote time points in which estimates were obtained. Rows denote estimates for each realisation.

`raw_Hurst_estimates` A list of data frames of raw Hurst estimates.

`call` Information about the input parameters used

See Also

`print.k_hurst`, `plot.k_hurst`, `hclust_hurst`

Examples

```
#Simulation of multifractional processes
t <- seq(0,1,by=(1/2)^10)
H1 <- function(t) {return(0.1+0*t)}
H2 <- function(t) {return(0.2+0.45*t)}
H3 <- function(t) {return(0.5-0.4*sin(6*3.14*t))}
X.list.1 <- replicate(3, GHBMP(t,H1),simplify = FALSE)
X.list.2 <- replicate(3, GHBMP(t,H2),simplify = FALSE)
X.list.3 <- replicate(3, GHBMP(t,H3),simplify = FALSE)
X.list <- c(X.list.1,X.list.2,X.list.3)

#K-means clustering based on k=3 clusters
KC <- kmeans_hurst(X.list,k=3)
print(KC)

#Plot of smoothed Hurst functions in each cluster with cluster centers
plot(KC,type ="ec")
```

LFD *Estimation of the local fractal dimension*

Description

This function computes the estimates for the local fractal dimension of multifractional processes.

Usage

```
LFD(X, N = 100, Q = 2, L = 2)
```

Arguments

`X`	Data frame where the first column is a time sequence and the second is the values of the multifractional process. For reliable estimates the data frame should be of at least 500 data points.
`N`	The same argument that is used for the estimation of Hurst function. Number of sub-intervals on which the estimation is performed on. Default is set to 100 sub-intervals.
`Q`	The same argument that is used for the estimation of Hurst function. Fixed integer greater than or equal to 2. Default is set to 2.

L	The same argument that is used for the estimation of Hurst function. Fixed integer greater than or equal to 2. Default is set to 2.

Details

The formula $\widehat{LFD} = 2 - \widehat{H}(t)$ is used to compute the estimated local fractal dimension, where $\widehat{H}(t)$ is the estimated Hurst function.

Value

A data frame where the first column is a time sequence and the second column is estimated values of the local fractal dimension.

References

Gneiting, T., and Schlather, M. (2004). Stochastic models that separate fractal dimension and the Hurst effect. SIAM Review, 46(2):269-282. doi:doi.org/10.1137/S0036144501394387.

See Also

Hurst, plot.mp, plot_tsest, plot.H_LFD

Examples

```
#LFD of a multifractional process simulated using GHBMP function
T <- seq(0,1,by=(1/2)^10)
H <- function(t) {return(0.5-0.4*sin(6*3.14*t))}
X <- GHBMP(T,H)
LFD(X)
```

---

long_streak *Longest increasing/decreasing streak*

Description

Computes the time span of the longest increasing or decreasing streak of a time series for the provided time interval or its sub-interval.

Usage

```
long_streak(X, direction = "increasing", subI = NULL, plot = FALSE)
```

Arguments

X	Data frame where the first column is a time sequence t and the second one is the values of the time series $X(t)$.
direction	A character string which specifies the direction of the streak: "increasing" or "decreasing".
subI	Time sub-interval is a vector, where the lower bound is the first element and the upper bound is the second. Optional: If provided level crossing times of the sub-interval is returned, otherwise the whole time interval is considered.
plot	Logical: If TRUE, the time series and the longest streak of increasing/decreasing is plotted.

Value

Time span t and the corresponding $X(t)$ of the longest increasing/decreasing streak.

See Also

mean_streak

Examples

```
t <- seq(0,1,length=100)
TS <- data.frame("t"=t,"X(t)"=rnorm(100))
long_streak(TS,direction='decreasing',subI=c(0.2,0.8),plot=TRUE)
```

---

mean_streak — *Mean time span of increasing/decreasing streaks*

Description

Computes the mean time span of the increasing/decreasing streaks for the provided time interval or its sub-interval.

Usage

```
mean_streak(X, direction = "increasing", subI = NULL, plot = FALSE)
```

Arguments

X	Data frame where the first column is a time sequence t and the second one is the values of the time series $X(t)$.
direction	A character string which specifies the direction of the streak: "increasing" or "decreasing".
subI	Time sub-interval is a vector, where the lower bound is the first element and the upper bound is the second. Optional: If provided level crossing times of the sub-interval is returned, otherwise the whole time interval is considered.
plot	Logical: If TRUE, the time series and the increasing/decreasing streaks are plotted.

Value

Mean time span of the increasing/decreasing streaks

See Also

long_streak

Examples

```
t <- seq(0,1,length=100)
TS <- data.frame("t"=t,"X(t)"=rnorm(100))
mean_streak(TS,direction='decreasing',subI=c(0.2,0.8),plot=TRUE)
```

plot.hc_hurst *Plot smoothed Hurst functions in each cluster with cluster centers*

Description

Creates a plot of the smoothed Hurst functions of realisations of multifractional processes separately in each cluster with cluster centers using the return from hclust_hurst. Options to plot only estimates, only centers or both are available.

Usage

```
## S3 method for class 'hc_hurst'
plot(x, type = "estimates", ...)
```

Arguments

x	Return from hclust_hurst.
type	The type of plot required: "estimates" Only the smoothed Hurst functions in each cluster. "centers" Only the cluster centers. Center denotes average of all smoothed Hurst functions in the cluster "ec" Both "estimates" and "centers".
...	Other arguments

Value

A ggplot object which is used to plot the relevant type of plot: "estimates", "centers" or "ec".

See Also

hclust_hurst

Examples

```
#Simulation of multifractional processes
t <- seq(0,1,by=(1/2)^10)
H1 <- function(t) {return(0.1+0*t)}
H2 <- function(t) {return(0.2+0.45*t)}
H3 <- function(t) {return(0.5-0.4*sin(6*3.14*t))}
X.list.1 <- replicate(3, GHBMP(t,H1),simplify = FALSE)
X.list.2 <- replicate(3, GHBMP(t,H2),simplify = FALSE)
X.list.3 <- replicate(3, GHBMP(t,H3),simplify = FALSE)
X.list <- c(X.list.1,X.list.2,X.list.3)

#Hierarchical clustering based on k=3 clusters with dendrogram plotted
HC<- hclust_hurst(X.list,k=3,dendrogram=TRUE)
print(HC)

#Plot of smoothed Hurst functions in each cluster with cluster centers
plot(HC,type ="ec")
```

plot.H_LFD — *Plot the estimated Hurst functions and local fractal dimension estimates for objects of class* H_LFD

Description

Creates a plot of the user provided time series with the Hurst function estimated using Hurst, the smoothed estimated Hurst function and local fractal dimension estimated using LFD for objects of class "H_LFD".

Usage

```
## S3 method for class 'H_LFD'
plot(
  x,
  Raw_Est_H = TRUE,
  Smooth_Est_H = TRUE,
  LFD_Est = TRUE,
  LFD_Smooth_Est = TRUE,
  ...
)
```

Arguments

x	Return from H_LFD.
Raw_Est_H	Logical: If TRUE, the Hurst function estimated by using Hurst is plotted.
Smooth_Est_H	Logical: If TRUE, the smoothed estimated Hurst function is plotted. The estimated Hurst function is smoothed using the loess method.
LFD_Est	Logical: If TRUE, the local fractal dimension estimates are plotted.
LFD_Smooth_Est	Logical: If TRUE, the smoothed estimates of local fractal dimension is plotted.
...	Other arguments.

Details

Compared to plot_tsest, the function's argument is a "H_LFD" object, not a time series.

Value

A ggplot object which is used to plot the time series with theoretical, raw and smoothed estimates of Hurst function and raw and smoothed estimates of local fractal dimension.

See Also

H_LFD Hurst, LFD, plot_tsest

Examples

```
TS <- data.frame("t"=seq(0,1,length=1000),"X(t)"=rnorm(1000))
Object <- H_LFD(TS,N=100,Q=2,L=2)
plot(Object,Raw_Est_H=TRUE,Smooth_Est_H=TRUE,LFD_EST=TRUE,LFD_Smooth_Est=TRUE)
```

`plot.k_hurst` *Plot smoothed Hurst functions in each cluster with cluster centers*

Description

Creates a plot of the smoothed Hurst functions of realisations of multifractional processes separately in each cluster with cluster centers using the return from `kmeans_hurst`. Options to plot only estimates, only centers or both are available.

Usage

```
## S3 method for class 'k_hurst'
plot(x, type = "estimates", ...)
```

Arguments

`x`	Return from `kmeans_hurst`.
`type`	The type of plot required. `"estimates"` Only the smoothed Hurst functions in each cluster. `"centers"` Only the cluster centers. Center denotes average of all smoothed Hurst functions in the cluster `"ec"` Both `"estimates"` and `"centers"`.
`...`	Other arguments

Value

A ggplot object which is used to plot the relevant `type` of plot: `"estimates"`, `"centers"` or `"ec"`.

See Also

`kmeans_hurst`

Examples

```
#Simulation of multifractional processes
t <- seq(0,1,by=(1/2)^10)
H1 <- function(t) {return(0.1+0*t)}
H2 <- function(t) {return(0.2+0.45*t)}
H3 <- function(t) {return(0.5-0.4*sin(6*3.14*t))}
X.list.1 <- replicate(3, GHBMP(t,H1),simplify = FALSE)
X.list.2 <- replicate(3, GHBMP(t,H2),simplify = FALSE)
X.list.3 <- replicate(3, GHBMP(t,H3),simplify = FALSE)
X.list <- c(X.list.1,X.list.2,X.list.3)

#K-means clustering based on k=3 clusters
KC <- kmeans_hurst(X.list,k=3)
print(KC)

#Plot of smoothed Hurst functions in each cluster with cluster centers
plot(KC,type ="ec")
```

plot.mp *Plot Gaussian Haar-based multifractional processes with their theoretical and estimated Hurst functions and local fractal dimension*

Description

Creates a plot of the Gaussian Haar-based multifractional process simulated by using GHBMP with theoretical Hurst function (if provided), Hurst function estimated using Hurst, the smoothed estimated Hurst function and local fractal dimension estimated using LFD and smoothed estimates of local fractal dimension.

Usage

```
## S3 method for class 'mp'
plot(
  x,
  H = NULL,
  Raw_Est_H = TRUE,
  Smooth_Est_H = TRUE,
  LFD_Est = TRUE,
  LFD_Smooth_Est = TRUE,
  N = 100,
  Q = 2,
  L = 2,
  ...
)
```

Arguments

x	Return from GHBMP. For accurate estimated Hurst functions, x should be of at least 500 data points.
H	Theoretical Hurst function. Optional: If provided, the theoretical Hurst function is plotted.
Raw_Est_H	Logical: If TRUE, the Hurst function estimated by using Hurst is plotted.
Smooth_Est_H	Logical: If TRUE, the smoothed estimated Hurst function is plotted. The estimated Hurst function is smoothed using the loess method.
LFD_Est	Logical: If TRUE, the local fractal dimension estimates are plotted.
LFD_Smooth_Est	Logical: If TRUE, the smoothed estimates of local fractal dimension is plotted.
N	Argument used for the estimation of Hurst functions and LFD. Number of subintervals on which the estimation is performed on. Default is set to 100 subintervals.
Q	Argument used for the estimation of Hurst functions and LFD. Fixed integer greater than or equal to 2. Default is set to 2.
L	Argument used for the estimation of Hurst functions and LFD. Fixed integer greater than or equal to 2. Default is set to 2.
...	Other arguments.

Value

A ggplot object which is used to plot the multifractional process with theoretical, raw and smoothed estimates of Hurst function and raw and smoothed estimates of local fractal dimension.

See Also

GHBMP, Hurst, LFD

Examples

```
#Simulation of the multifractional process and estimation of the Hurst function
T <- seq(0,1,by=(1/2)^10)
H <- function(t) {return(0.5-0.4*sin(6*3.14*t))}
X <- GHBMP(T,H)

#Plot of process, theoretical, estimated and smoothed Hurst functions and LFD estimate
plot(X,H=H)

#Plot of process, estimated and smoothed Hurst functions and LFD estimate
plot(X)
```

`plot_tsest` *Plot the estimated Hurst functions and local fractal dimension estimates for a user provided time series*

Description

Creates a plot of the user provided time series with the Hurst function estimated using Hurst, the smoothed estimated Hurst function and local fractal dimension estimated using LFD and smoothed estimates of local fractal dimension.

Usage

```
plot_tsest(
  X,
  Raw_Est_H = TRUE,
  Smooth_Est_H = TRUE,
  LFD_Est = TRUE,
  LFD_Smooth_Est = TRUE,
  N = 100,
  Q = 2,
  L = 2
)
```

Arguments

X	Data frame where the first column is a time sequence t and the second one is the values of the time series $X(t)$. For accurate estimated Hurst functions, X should be of at least 500 data points.
Raw_Est_H	Logical: If TRUE, the Hurst function estimated by using Hurst is plotted.
Smooth_Est_H	Logical: If TRUE, the smoothed estimated Hurst function is plotted. The estimated Hurst function is smoothed using the loess method.

LFD_Est	Logical: If TRUE, the local fractal dimension estimates are plotted.
LFD_Smooth_Est	Logical: If TRUE, the smoothed estimates of local fractal dimension is plotted.
N	Argument used for the estimation of Hurst functions and LFD. Number of sub-intervals on which the estimation is performed on. Default is set to 100 sub-intervals.
Q	Argument used for the estimation of Hurst functions and LFD. Fixed integer greater than or equal to 2. Default is set to 2.
L	Argument used for the estimation of Hurst functions and LFD. Fixed integer greater than or equal to 2. Default is set to 2.

Details

Compared to plot.H_LFD the function's first argument is a time series, not H_LFD object.

Value

A ggplot object which is used to plot the time series with theoretical, raw and smoothed estimates of Hurst function and raw and smoothed estimates of local fractal dimension.

See Also

Hurst, LFD, plot.H_LFD

Examples

```
TS <- data.frame("t"=seq(0,1,length=1000),"X(t)"=rnorm(1000))
plot_tsest(TS,Raw_Est_H=TRUE,Smooth_Est_H=TRUE,LFD_Est=TRUE,LFD_Smooth_Est=TRUE)
```

---

print.hc_hurst — *Print method for "hc_hurst" class objects*

Description

Prints the results of hierarchical clustering of realisations of processes.

Usage

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

Arguments

x	Object of class "hc_hurst".
...	Other arguments

See Also

hclust_hurst

`print.k_hurst` *Print method for "k_hurst" class objects*

Description

Prints the results of k-means clustering of realisations of processes.

Usage

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

Arguments

`x`	Object of class `"k_hurst"`.
`...`	Other arguments

See Also

`kmeans_hurst`

`RS_Index` *Relative strength index*

Description

This function computes the Relative Strength Index (RSI) for a time series.

Usage

```
RS_Index(X, period = 14, plot = FALSE, overbought = 70, oversold = 30)
```

Arguments

`X`	A list, vector or `xts` object.
`period`	Period length used for smoothing. Default is set to 14.
`plot`	Logical: If `TRUE`, the time series and the RSI are plotted in the same window.
`overbought`	Horizontal line which indicates an overbought level in the RSI plot. Default is set to 70.
`oversold`	Horizontal line which indicates an oversold level in the RSI plot. Default is set to 30.

Details

To compute the RSI,

$$\mathrm{RSI} = 100\frac{\mathrm{Average\_gain}}{\mathrm{Average\_gain} + \mathrm{Average\_loss}}$$

formula is used. Average gain and average loss are computed using the Wilders's smoothing method.

Value

A list, vector or xts object of the RSI values. If plot=TRUE, the time series and the RSI with overbought and oversold levels are plotted.

References

Wilder, J. W. (1978). New concepts in technical trading systems. Greensboro, NC.

Examples

```
X <- c(74.44,74.19,74.25,73.65,74.37,74.73,75.15,75.46,75.88,76.78,
          75.81,76.53,75.11,76.28,76.68,76.08,76.53,76.11,76.42,75.58,
          75.44,75.46,74.98)
RS_Index(X,plot=TRUE)
```

---

shinyapp_sim *Shiny app to visualise and analyse processes*

Description

Launches a Shiny app to visualise and analyse realisations of Brownian motion (see Bm), Brownian bridge (see Bbridge), fractional Brownian motion (see FBm), fractional Brownian bridge (see FBbridge), fractional Gaussian noise(see FGn), Gaussian Haar-based multifractional processes (see GHBMP) and user-provided time series data.

Usage

```
shinyapp_sim()
```

Details

For Input time series, provide a .csv file that include headers with two columns: the first column should contain the time sequence t, and the second the corresponding time series values $X(t)$. Make sure there are no extra header rows or footnotes.

Value

An interactive Shiny app with the following user interface controls.

Hurst function and LFD estimation

Hurst function Input the Hurst function in terms of t. The default is set to 0.5+0*t.

Time sequence Input the time sequence which belongs to the interval $[0, 1]$. The default is set to seq(0,1,by=(1/2)^10).

J Input or select a positive integer. For large J could be rather time consuming. Default is set to 15.

Number of sub-intervals for estimation Default is set to 100.

Q Input or select an integer greater than or equal to 2. Default is set to 2.

L Input or select an integer greater than or equal to 2. Default is set to 2.

`Plot` Select: `Theoretical Hurst function`, `Raw estimate of Hurst function`, `Smoothed estimate of Hurst function`, `Raw estimate of Local Fractal Dimension`, `Smoothed estimate of Local Fractal Dimension`.

Excursion set and area

`Number of time steps` Input the number of steps the time interval needs to be split into.

`Constant level` Input the constant level.

`Compare to level` `Greater`, `Lower`.

`Plot` Select: `Excursion set`, `Excursion area`.

Longest streak

`Longest streak to plot` Select: `Increasing`, `Decreasing`.

Maximum and minimum

`Plot` Select: `Maximum`, `Minimum`.

See Also

`Bm`, `FBm`, `FGn`, `Bbridge`, `FBbridge`, `GHBMP`, `Hurst`

Examples

```
## Not run:
#To run the Shiny app
shinyapp_sim()

## End(Not run)
```

`sojourn` *Estimated sojourn measure*

Description

Computes the estimated sojourn measure for a time series $X(t)$ greater or lower than the constant level `A` for the provided time interval or its sub-interval.

Usage

```
sojourn(X, A, N = 10000, level = "greater", subI = NULL, plot = FALSE)
```

Arguments

`X`	Data frame where the first column is a time sequence t and the second one is the values of the time series $X(t)$.
`A`	Constant level as a numeric value.
`N`	Number of steps the time interval (or time sub-interval) is split into. Default set to 10000.

level	A vector of character strings which specifies which sojourn measure required for X, "greater" or "lower" than A. Default set to "greater".
subI	Time sub-interval is a vector, where the lower bound is the first element and the upper bound is the second. Optional: If provided, the estimated sojourn measure for the sub-interval is returned, otherwise the whole time interval is considered.
plot	Logical: If TRUE, the time series, constant level and the sojourn measure are plotted.

Value

Estimated sojourn measure. If plot=TRUE, the time series, the constant level (in blue) and the excursion region (in red) are plotted.

See Also

exc_Area

Examples

```
t <- seq(0,1,length=1000)
TS <- data.frame("t"=t,"X(t)"=rnorm(1000))
sojourn(TS,0.8,level='lower',subI=c(0.5,0.8),plot=TRUE)
```

X_max *Estimated maximum of a time series*

Description

This function computes the maximum of a time series for the provided time interval or its sub-interval.

Usage

```
X_max(X, subI = NULL, plot = FALSE, vline = FALSE, hline = FALSE)
```

Arguments

X	Data frame where the first column is a time sequence (t) and the second the values of the time series $(X(t))$.
subI	Time sub-interval is a vector where the lower bound is the first element and upper bound is the second. Optional: If provided maximum of the sub-interval is returned, otherwise the whole time sequence is considered.
plot	Logical: If TRUE, the time series, the maximum and corresponding t values are plotted.
vline	Logical: If TRUE, a vertical line is plotted across the maximum.
hline	Logical: If TRUE, a horizontal line is plotted across the maximum.

Value

Print the maximum of the time series and the corresponding t values. If plot=TRUE, the time series with with maximum and corresponding t values are plotted.

See Also

X_min

Examples

```
t <- seq(0,1,length=100)
TS <- data.frame("t"=t,"X(t)"=rnorm(100))
X_max(TS,subI=c(0.5,0.8),plot=TRUE)
```

X_min *Estimated minimum of a time series*

Description

This function computes the minimum of a time series for the provided time interval or its sub-interval.

Usage

```
X_min(X, subI = NULL, plot = FALSE, vline = FALSE, hline = FALSE)
```

Arguments

X	Data frame where the first column is a time sequence t and the second the values of the time series $X(t)$.
subI	Time sub-interval is a vector where the lower bound is the first element and upper bound is the second. Optional: If provided minimum of the sub-interval is returned, otherwise the whole time interval is considered.
plot	Logical: If TRUE, the time series, the minimum and corresponding t values are plotted.
vline	Logical: If TRUE, a vertical line is plotted across the minimum.
hline	Logical: If TRUE, a horizontal line is plotted across the minimum.

Value

Print the minimum of the time series and the corresponding t values. If plot=TRUE, the time series with with minimum and corresponding t values are plotted.

See Also

X_max

Examples

```
t <- seq(0,1,length=100)
TS <- data.frame("t"=t,"X(t)"=rnorm(100))
X_min(TS,subI=c(0.2,0.8),plot=TRUE)
```

Index