Package ‘Rmfrac’

July 2, 2025

Type Package

Title Simulation and analyses of multifractional processes

Version 0.1.0

Author Andriy Olenko, Nemini Samarakoon

Maintainer Nemini Samarakoon <NWijesingheS@ltu.edu.au>

Description Simulation of the Guassian Haar Based Multifractional Processes (GHBMP). Estimation of Hurst functions, clustering realisations based on the Hurst functions and several functions to estimate and plot characteristics of the processes.

License GPL-3

Encoding UTF-8

Imports parallel, parallelly, ggplot2, zoo, matrixStats, proxy, rlang, stats, foreach, doParallel, plotly, shiny, graphics, shinycssloaders, xts

LazyData false

RoxygenNote 7.3.1

Roxygen list(markdown = TRUE)

Contents

`A.excursion` *Excursion area*

Description

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

Usage

```
A.excursion(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 points a time interval to be 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))
A.excursion(TS,0.8,level='lower',subI=c(0.5,0.8),plot=TRUE)
```

`Bm` *Simulation of Brownian motion*

Description

This function simulates a realisation of the Brownian motion over time interval $[t_{start}, t_{end}]$ with `N` increments and initial value `x1`.

Usage

```
Bm(x1 = 0, t_start = 0, t_end = 1, N = 1000, plot = FALSE)
```

Arguments

`x1`	Value of the process at the initial time point.
`t_start`	Initial time point.
`t_end`	Terminal time point.
`N`	Number of time steps 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.

See Also

`GHBMP`, `FBM`, `sim_FGN`

Examples

```
Bm(x1=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, 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.
`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.

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

```
## Not run:
#Covariance of a GHBMP with H=0.5-0.4* sin(6*3.14*t)
t <- seq(0,1,by=0.01)
H <- function(t) {return(0.5-0.4* sin(6*3.14*t))}
cov_GHBMP(t,H,plot=TRUE)

## End(Not run)
```

`cross_mean` *Mean time between crossings*

Description

Computes the mean duration between crossings of a time series across 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 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 crossing points are plotted.

Value

The estimated mean time between crossings. If `plot=TRUE`, a plot of 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 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. If `plot=TRUE`, a plot of 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 level crossing times*

Description

Computes the estimated t value(s), where 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 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 across the crossing point(s).

Value

The estimated level crossing times. If `plot=TRUE`, a plot of 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 of a process*

Description

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

Usage

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

Arguments

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

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 realizations 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,plot=TRUE)
cov.mat
```

FBM *Simulation of fractional Brownian motion*

Description

This function simulates a realisation of the fractional Brownian motion over time interval $[0, t_{end}]$ for a provided Hurst parameter.

Usage

```
FBM(H, t_end, N = 1000, plot = FALSE)
```

Arguments

`H`	Hurst parameter which lies between 0 and 1.
`t_end`	Terminal time point.
`N`	Number of time steps the interval $[0, t_{end}]$ 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.

See Also

`sim_FGN`, `Bm`, `GHBMP`

Examples

```
FBM(H=0.3,t_end=1,plot=TRUE)
```

`GHBMP` *Simulation of Gaussian Haar-based multifractional processes*

Description

This function simulates 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 considered.

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 Hurst functions, for example, constant, time-varying, piecewise or step 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, sim_FGN

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 of multifractional processes*

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 from 0 to 1 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.t, N = 100, Q = 2, L = 2)
```

Arguments

`X.t`	Data frame where the first column is a time sequence from 0 to 1 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 class `"est"` 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

`plot.est`

Examples

```
#Example 1: For 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)
```

`kmeans_hurst` *K-means clustering of multifractional processes*

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 from 0 to 1 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.t, N = 100, Q = 2, L = 2)
```

Arguments

`X.t`	Data frame where the first column is a time sequence from 0 to 1 and the second the values of the multifractional process. For reliable estimates the data frame should be of at least 500 data points.
`N`	Argument 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`	Argument used for the estimation of Hurst function. Fixed integer greater than or equal to 2. Default is set to 2.
`L`	Argument used for the estimation of Hurst function. Fixed integer greater than or equal to 2. Default is set to 2.

Value

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

See Also

`Hurst`, `plot.est`, `plot.mp`

Examples

```
#Example 1: For 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 streak of increasing/decreasing*

Description

Computes the time span of the longest increasing or decreasing streak 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 vector of character strings 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 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 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 vector of character strings 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 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.est` *Plot theoretical and estimated Hurst functions*

Description

Creates a plot of the estimated Hurst function with the theoretical Hurst function and the smoothed estimated Hurst function using the return from `Hurst`.

Usage

```
## S3 method for class 'est'
plot(x, H = NULL, Raw_Est_H = TRUE, Smooth_Est_H = TRUE, ...)
```

Arguments

`x`	Return from `Hurst`.
`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.
`...`	Unused arguments.

Value

A ggplot object which plots the theoretical, estimated and smoothed Hurst functions.

See Also

Hurst

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)
Est_H <- Hurst(X)

#Plot of theoretical, estimated and smoothed Hurst functions
plot(Est_H,H=H)

#Plot of estimated and smoothed Hurst functions
plot(Est_H)
```

---

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"`.
`...`	Unused arguments

Value

A ggplot object which plots 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.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 plots 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*

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 and the smoothed estimated Hurst function.

Usage

```
## S3 method for class 'mp'
plot(
  x,
  H = NULL,
  Raw_Est_H = TRUE,
  Smooth_Est_H = 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.
`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.
`...`	Unused arguments.

Value

A ggplot object which plots the multifractional process with theoretical, estimated and smoothed Hurst functions.

See Also

`GHBMP`, `Hurst`

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 multifractional process, theoretical, estimated and smoothed Hurst functions
plot(X,H=H)

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

`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"`.
`...`	Unused 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"`.
`...`	Unused arguments

See Also

kmeans_hurst

RS_Index — *Relative strength index*

Description

This function computes the Relative Strength Index (RSI) from a series of prices.

Usage

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

Arguments

`prices`	Series of prices as a list, vector or `xts` object.
`period`	Period length used for smoothing. Default is set to 14.
`plot`	Logical: If `TRUE`, a plot of the price series and a plot of the RSI are displayed 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, $100\frac{Average_gain}{Average_gain + Average_loss}$ 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`, a plot of the price series and a plot of the RSI with `overbought` and `oversold` levels are plotted.

Examples

```
#prices as a list
prices <- 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(prices,plot=TRUE)

#time series object
library(xts)
Date <- seq(as.Date("2025-07-01"), by = "days", length.out = length(prices))
price_ts <- xts(prices, order.by = Date)
RS_Index(price_ts,plot=TRUE)
```

`shinyapp_sim` *Shiny app to plot realisations of Gaussian Haar-based multifractional processes with theoretical and estimated Hurst functions*

Description

Launches a shiny app that plots the realisations of Gaussian Haar-based multifractional processes simulated using the function `GHBMP` with theoretical Hurst function, Hurst function estimated using `Hurst` and the smoothed estimated Hurst function.

Usage

```
shinyapp_sim()
```

Value

An interactive shiny app with the following user interface controls:

`Hurst function` Input the Hurst function in terms of `t`. The default is set to `0.2+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.

`Hurst function to plot` Select: `Theoretical Hurst function`, `Raw estimate of Hurst function`, `Smoothed estimate of Hurst function`

See Also

GHBMP, Hurst

Examples

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

## End(Not run)
```

`sim_FGN` *Simulation of fractional Gaussian noise*

Description

This function simulates a realisation of the fractional Gaussian noise over time interval $[t_{start}, t_{end}]$ for a provided Hurst parameter.

Usage

```
sim_FGN(H, t_start, t_end, 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 steps the interval $[t_{start}, t_{end}]$ is split into. 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.

See Also

FBM, Bm, GHBMP

Examples

```
sim_FGN(H=0.3,t_start=0,t_end=1,plot=TRUE)
```

sojourn *Estimated sojourn measure*

Description

Computes the estimated sojourn measure for $X(t)$ 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 points a time interval to be 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 and the excursion region are plotted.

See Also

A.excursion

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`, a plot of 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, a plot of 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