Package ‘rlfsm’

October 14, 2022

Type Package

Title Simulations and Statistical Inference for Linear Fractional Stable Motions

Version 1.1.2

Maintainer Dmitry Otryakhin <d.otryakhin.acad@protonmail.ch>

Author Dmitry Otryakhin [aut, cre] (<https://orcid.org/0000-0002-4700-7221>),
Stepan Mazur [aut] (<https://orcid.org/0000-0002-1395-9427>),
Mathias Ljungdahl [ctb]

Description Contains functions for simulating the linear fractional stable motion according to the algorithm developed by Mazur and Otryakhin <doi:10.32614/RJ-2020-008> based on the method from Stoev and Taqqu (2004) <doi:10.1142/S0218348X04002379>, as well as functions for estimation of parameters of these processes introduced by Mazur, Otryakhin and Podolskij (2018) <arXiv:1802.06373>, and also different related quantities.

License GPL-3

URL https://gitlab.com/Dmitry_Otryakhin/Tools-for-parameter-estimation-of-the-linear-fractional-stable-motion

Encoding UTF-8

RoxygenNote 7.2.1

Depends methods, foreach, doParallel

Imports ggplot2, stabledist, reshape2, plyr, Rdpack, Rcpp

Suggests elliptic, testthat, stringi

RdMacros Rdpack

LinkingTo Rcpp

NeedsCompilation yes

Repository CRAN

Date/Publication 2022-08-27 13:20:05 UTC

R topics documented:

`alpha_hat` *Statistical estimator for alpha*

Description

Defined for the two frequencies as

$$\widehat{\alpha}_{high} := \frac{\log|\log\varphi_{high}(t_2;\widehat{H}_{high}(p,k)_n,k)_n| - \log|\log\varphi_{high}(t_1;\widehat{H}_{high}(p,k)_n,k)_n|}{\log t_2 - \log t_1}$$

$$\widehat{\alpha}_{low} := \frac{\log|\log\varphi_{low}(t_2;k)_n| - \log|\log\varphi_{low}(t_1;k)_n|}{\log t_2 - \log t_1}$$

Usage

```
alpha_hat(t1, t2, k, path, H, freq)
```

Arguments

t1, t2	real number such that t2 > t1 > 0
k	increment order
path	sample path of lfsm on which the inference is to be performed
H	Hurst parameter
freq	Frequency of the motion. It can take two values: "H" for high frequency and "L" for the low frequency setting.

Details

The function triggers function phi, thus Hurst parameter is required only in high frequency case. In the low frequency, there is no need to assign H a value because it will not be evaluated.

References

Mazur S, Otryakhin D, Podolskij M (2020). "Estimation of the linear fractional stable motion." *Bernoulli*, **26**(1), 226–252. https://doi.org/10.3150/19-BEJ1124.

Examples

```
m<-45; M<-60; N<-2^14-M
alpha<-1.8; H<-0.8; sigma<-0.3
freq='H'
r=1; k=2; p=0.4; t1=1; t2=2

# Estimating alpha in the high frequency case
# using preliminary estimation of H
lfsm<-path(N=N,m=m,M=M,alpha=alpha,H=H,
           sigma=sigma,freq='L',disable_X=FALSE,seed=3)$lfsm

H_est<-H_hat(p=p,k=k,path=lfsm)
H_est
alpha_est<-alpha_hat(t1=t1,t2=t2,k=k,path=lfsm,H=H_est,freq=freq)
alpha_est
```

---

a_p *Function a_p.*

Description

Computes the corresponding function value from Mazur et al. 2018.

Usage

```
a_p(p)
```

Arguments

p	power, real number from (-1,1)

References

Mazur S, Otryakhin D, Podolskij M (2020). "Estimation of the linear fractional stable motion." *Bernoulli*, **26**(1), 226–252. https://doi.org/10.3150/19-BEJ1124.

---

a_tilda — *Creates the corresponding value from the paper by Stoev and Taqqu (2004).*

Description

a_tilda triggers a_tilda_cpp which is written in C++ and essentially performs the computation of the value.

Usage

```
a_tilda(N, m, M, alpha, H)
```

Arguments

N	a number of points of the lfsm.
m	discretization. A number of points between two nearby motion points
M	truncation parameter. A number of points at which the integral representing the definition of lfsm is calculated. So, after M points back we consider the rest of the integral to be 0.
alpha	self-similarity parameter of alpha stable random motion.
H	Hurst parameter

References

Stoev S, Taqqu MS (2004). "Simulation methods for linear fractional stable motion and FARIMA using the Fast Fourier Transform." *Fractals*, **95**(1), 95-121. https://doi.org/10.1142/S0218348X04002379.

ContinEstim — *Parameter estimation procedure in continuous case.*

Description

Parameter freq is preserved to allow for investigation of the inference procedure in high frequency case.

Usage

```
ContinEstim(t1, t2, p, k, path, freq)
```

Arguments

`t1, t2`	real number such that t2 > t1 > 0
`p`	power
`k`	increment order
`path`	sample path of lfsm on which the inference is to be performed
`freq`	Frequency of the motion. It can take two values: "H" for high frequency and "L" for the low frequency setting.

References

Mazur S, Otryakhin D, Podolskij M (2020). "Estimation of the linear fractional stable motion." *Bernoulli*, **26**(1), 226–252. https://doi.org/10.3150/19-BEJ1124.

Examples

```
m<-45; M<-60; N<-2^10-M
alpha<-0.8; H<-0.8; sigma<-0.3
p<-0.3; k=3; t1=1; t2=2

lfsm<-path(N=N,m=m,M=M,alpha=alpha,H=H,
           sigma=sigma,freq='L',disable_X=FALSE,seed=3)$lfsm
ContinEstim(t1,t2,p,k,path=lfsm,freq='L')
```

GenHighEstim — *High frequency estimation procedure for lfsm.*

Description

General estimation procedure for high frequency case when 1/alpha is not a natural number. "Unnecessary" parameter freq is preserved to allow for investigation of the inference procedure in low frequency case

Usage

```
GenHighEstim(p, p_prime, path, freq, low_bound = 0.01, up_bound = 4)
```

Arguments

p	power
p_prime	power
path	sample path of lfsm on which the inference is to be performed
freq	Frequency of the motion. It can take two values: "H" for high frequency and "L" for the low frequency setting.
low_bound	positive real number
up_bound	positive real number

Details

In this algorithm the preliminary estimate of alpha is found via using uniroot function. The latter is given the lower and the upper bounds for alpha via low_bound and up_bound parameters. It is not possible to pass 0 as the lower bound because there are numerical limitations on the alpha estimate, caused by the length of the sample path and by numerical errors. p and p_prime must belong to the interval (0,1/2) (in the notation kept in rlfsm package) The two powers cannot be equal.

References

Mazur S, Otryakhin D, Podolskij M (2020). "Estimation of the linear fractional stable motion." *Bernoulli*, **26**(1), 226–252. https://doi.org/10.3150/19-BEJ1124.

Examples

```
m<-45; M<-60; N<-2^10-M
sigma<-0.3
p<-0.2; p_prime<-0.4

#### Continuous case
lfsm<-path(N=N,m=m,M=M,alpha=1.8,H=0.8,
           sigma=sigma,freq='L',disable_X=FALSE,seed=3)$lfsm

GenHighEstim(p=p,p_prime=p_prime,path=lfsm,freq="H")

#### H-1/alpha<0 case
lfsm<-path(N=N,m=m,M=M,alpha=0.8,H=0.8,
           sigma=sigma,freq='H',disable_X=FALSE,seed=3)$lfsm

GenHighEstim(p=p,p_prime=p_prime,path=lfsm,freq="H")
```

GenLowEstim — *Low frequency estimation procedure for lfsm.*

Description

General estimation procedure for low frequency case when 1/alpha is not a natural number.

Usage

```
GenLowEstim(t1, t2, p, path, freq = "L")
```

Arguments

`t1, t2`	real number such that t2 > t1 > 0
`p`	power
`path`	sample path of lfsm on which the inference is to be performed
`freq`	Frequency of the motion. It can take two values: "H" for high frequency and "L" for the low frequency setting.

References

Mazur S, Otryakhin D, Podolskij M (2020). "Estimation of the linear fractional stable motion." *Bernoulli*, **26**(1), 226–252. https://doi.org/10.3150/19-BEJ1124.

Examples

```
m<-45; M<-60; N<-2^10-M
sigma<-0.3
p<-0.3; k=3; t1=1; t2=2

#### Continuous case
lfsm<-path(N=N,m=m,M=M,alpha=1.8,H=0.8,
           sigma=sigma,freq='L',disable_X=FALSE,seed=3)$lfsm

GenLowEstim(t1=t1,t2=t2,p=p,path=lfsm,freq="L")

#### H-1/alpha<0 case
lfsm<-path(N=N,m=m,M=M,alpha=0.8,H=0.8,
           sigma=sigma,freq='L',disable_X=FALSE,seed=3)$lfsm

GenLowEstim(t1=t1,t2=t2,p=p,path=lfsm,freq="L")

#### The procedure works also for high frequency case
lfsm<-path(N=N,m=m,M=M,alpha=1.8,H=0.8,
           sigma=sigma,freq='H',disable_X=FALSE,seed=3)$lfsm

GenLowEstim(t1=t1,t2=t2,p=p,path=lfsm,freq="H")
```

H_hat — *Statistical estimator of H in high/low frequency setting*

Description

The statistic is defined as

$$\widehat{H}_{\text{high}}(p,k)_n := \frac{1}{p}\log_2 R_{\text{high}}(p,k)_n, \qquad \widehat{H}_{\text{low}}(p,k)_n := \frac{1}{p}\log_2 R_{\text{low}}(p,k)_n$$

Usage

```
H_hat(p, k, path)
```

Arguments

p	power
k	increment order
path	sample path of lfsm on which the inference is to be performed

References

Mazur S, Otryakhin D, Podolskij M (2020). "Estimation of the linear fractional stable motion." *Bernoulli*, **26**(1), 226–252. https://doi.org/10.3150/19-BEJ1124.

h_kr — *Function h_kr*

Description

Function $h_{k,r} : R \to R$ is given by

$$h_{k,r}(x) = \sum_{j=0}^{k} (-1)^j \binom{k}{j} (x - rj)_+^{H-1/\alpha}, \quad x \in R$$

Usage

```
h_kr(k, r, x, H, alpha, l = 0)
```

Arguments

k	order of the increment, a natural number
r	difference step, a natural number
x	real number
H	Hurst parameter
alpha	self-similarity parameter of alpha stable random motion.
l	a value by which we shift x. Is used for computing function f_.+l and is passed to integrate function.

References

Mazur S, Otryakhin D, Podolskij M (2020). "Estimation of the linear fractional stable motion." *Bernoulli*, **26**(1), 226–252. https://doi.org/10.3150/19-BEJ1124.

Examples

```
#### Plot h_kr ####
s<-seq(0,10, by=0.01)
h_val<-sapply(s,h_kr, k=5, r=1, H=0.3, alpha=1)
plot(s,h_val)
```

`increment` *Higher order increments*

Description

Difference of the kth order. Defined as following:

$$\Delta_{i,k}^{n,r} X := \sum_{j=0}^{k} (-1)^j \binom{k}{j} X_{(i-rj)/n}, i \geq rk.$$

Index i here is a coordinate in terms of point_num. Although R uses vector indexes that start from 1, increment has i varying from 0 to N, so that a vector has a length N+1. It is done in order to comply with the notation of the paper. This function doesn't allow for choosing frequency n. The frequency is determined by the path supplied, thus n equals to either the length of the path in high frequency setting or 1 in low frequency setting. increment() gives increments at certain point passed as i, which is a vector here. increments() computes high order increments for the whole sample path. The first function evaluates the formula above, while the second one uses structure diff(diff(...)) because the formula is slower at higher k.

Usage

```
increment(r, i, k, path)

increments(k, r, path)
```

Arguments

`r`	difference step, a natural number
`i`	index of the point at which the increment is to be computed, a natural number.
`k`	order of the increment, a natural number
`path`	sample path for which a kth order increment is computed

References

Mazur S, Otryakhin D, Podolskij M (2020). "Estimation of the linear fractional stable motion." *Bernoulli*, **26**(1), 226–252. https://doi.org/10.3150/19-BEJ1124.

Examples

```
m<-45; M<-60; N<-2^10-M
alpha<-0.8; H<-0.8; sigma<-0.3

lfsm<-path(N=N,m=m,M=M,alpha=alpha,H=H,
           sigma=sigma,freq='L',disable_X=FALSE,seed=3)$lfsm

tryCatch(
increment(r=1,i=length(lfsm),k=length(lfsm)+100,path=lfsm),
error=function(c) 'An error occures when k is larger then the length of the sample path')

increment(r=3,i=50,k=3,path=lfsm)


path=c(1,4,3,6,8,5,3,5,8,5,1,8,6)

r=2; k=3
n <- length(path) - 1
DeltaX = increment(seq(r*k, n), path = path, k = k, r = r)
DeltaX == increments(k=k,r=r,path)
```

`MCestimLFSM` *Numerical properties of statistical estimators operating on the linear fractional stable motion.*

Description

The function is useful, for instance, when one needs to compute standard deviation of $\widehat{\alpha}_{high}$ estimator given a fixed set of parameters.

Usage

```
MCestimLFSM(Nmc, s, m, M, alpha, H, sigma, fr, Inference, ...)
```

Arguments

`Nmc`	Number of Monte Carlo repetitions
`s`	sequence of path lengths
`m`	discretization. A number of points between two nearby motion points
`M`	truncation parameter. A number of points at which the integral representing the definition of lfsm is calculated. So, after M points back we consider the rest of the integral to be 0.
`alpha`	self-similarity parameter of alpha stable random motion.
`H`	Hurst parameter
`sigma`	Scale parameter of lfsm

`fr`	frequency. Either "H" or "L"
`Inference`	statistical function to apply to sample paths
`...`	parameters to pass to Inference

Details

MCestimLFSM performs Monte-Carlo experiments to compute parameters according to procedure Inference. More specifically, for each element of s it generates Nmc lfsm sample paths with length equal to s[i], performs the statistical inference on each, obtaining the estimates, and then returns their different statistics. It is vital that the estimator returns a list of named parameters (one or several of 'sigma', 'alpha' and 'H'). MCestimLFSM uses the names to lookup the true parameter value and compute its bias.

For sample path generation MCestimLFSM uses a light-weight version of path, path_fast. In order to be applied, function Inference must accept argument 'path' as a sample path.

Value

It returns a list containing the following components:

`data`	a data frame, values of the estimates depending on path length s
`data_nor`	a data frame, normalized values of the estimates depending on path length s
`means, biases, sds`	data frames: means, biases and standard deviations of the estimators depending on s
`Inference`	a function used to obtain estimates
`alpha, H, sigma`	the parameters for which MCestimLFSM performs path generation
`freq`	frequency, either 'L' for low- or 'H' for high frequency

Examples

```
#### Set of global parameters ####
m<-25; M<-60
p<-.4; p_prime<-.2; k<-2
t1<-1; t2<-2
NmonteC<-5e1
S<-c(1e2,3e2)
alpha<-1.8; H<-0.8; sigma<-0.3


# How to plot empirical density

theor_3_1_H_clt<-MCestimLFSM(s=S,fr='H',Nmc=NmonteC,
                     m=m,M=M,alpha=alpha,H=H,
                     sigma=sigma,ContinEstim,
                     t1=t1,t2=t2,p=p,k=k)
l_plot<-Plot_dens(par_vec=c('sigma','alpha','H'),
                  MC_data=theor_3_1_H_clt, Nnorm=1e7)
```

```
# For MCestimLFSM() it is vital that the estimator returns a list of named parameters

H_hat_f <- function(p,k,path) {hh<-H_hat(p,k,path); list(H=hh)}
theor_3_1_H_clt<-MCestimLFSM(s=S,fr='H',Nmc=NmonteC,
                     m=m,M=M,alpha=alpha,H=H,
                     sigma=sigma,H_hat_f,
                     p=p,k=k)


# The estimator can return one, two or three of the parameters.

est_1 <- function(path) list(H=1)
theor_3_1_H_clt<-MCestimLFSM(s=S,fr='H',Nmc=NmonteC,
                     m=m,M=M,alpha=alpha,H=H,
                     sigma=sigma,est_1)

est_2 <- function(path) list(H=0.8, alpha=1.5)
theor_3_1_H_clt<-MCestimLFSM(s=S,fr='H',Nmc=NmonteC,
                     m=m,M=M,alpha=alpha,H=H,
                     sigma=sigma,est_2)

est_3 <- function(path) list(sigma=5, H=0.8, alpha=1.5)
theor_3_1_H_clt<-MCestimLFSM(s=S,fr='H',Nmc=NmonteC,
                     m=m,M=M,alpha=alpha,H=H,
                     sigma=sigma,est_3)
```

---

m_pk *m(-p,k)*

Description

defined as $m_{p,k} := E[|\Delta_{k,k}X|^p]$ for positive powers. When p is negative (-p is positive) the equality does not hold.

Usage

```
m_pk(k, p, alpha, H, sigma)
```

Arguments

k	increment order
p	a positive number
alpha	self-similarity parameter of alpha stable random motion.
H	Hurst parameter
sigma	Scale parameter of lfsm

Details

The following identity is used for computations:

$$m_{-p,k} = \frac{(\sigma \|h_k\|_\alpha)^{-p}}{a_{-p}} \int_{\mathsf{R}} \exp(-|y|^\alpha)|y|^{-1+p} dy = \frac{2(\sigma \|h_k\|_\alpha)^{-p}}{\alpha a_{-p}} \Gamma(p/\alpha)$$

References

Mazur S, Otryakhin D, Podolskij M (2020). "Estimation of the linear fractional stable motion." *Bernoulli*, **26**(1), 226–252. https://doi.org/10.3150/19-BEJ1124.

`Norm_alpha` *Alpha-norm of an arbitrary function*

Description

Alpha-norm of an arbitrary function

Usage

```
Norm_alpha(fun, alpha, ...)
```

Arguments

`fun`	a function to compute a norm
`alpha`	self-similarity parameter of alpha stable random motion.
`...`	a set of parameters to pass to integrate

Details

fun must accept a vector of values for evaluation. See ?integrate for further details. Most problems with this function appear because of rather high precision. Try to tune rel.tol parameter first.

References

Mazur S, Otryakhin D, Podolskij M (2020). "Estimation of the linear fractional stable motion." *Bernoulli*, **26**(1), 226–252. https://doi.org/10.3150/19-BEJ1124.

Examples

```
Norm_alpha(h_kr,alpha=1.8,k=2,r=1,H=0.8,l=4)
```

path *Generator of linear fractional stable motion*

Description

The function creates a 1-dimensional LFSM sample path using the numerical algorithm from the paper by Otryakhin and Mazur. The theoretical foundation of the method comes from the article by Stoev and Taqqu. Linear fractional stable motion is defined as

$$X_t = \int_{\mathbb{R}} \left\{ (t-s)_+^{H-1/\alpha} - (-s)_+^{H-1/\alpha} \right\} dL_s$$

Usage

```
path(
  N = NULL,
  m,
  M,
  alpha,
  H,
  sigma,
  freq,
  disable_X = FALSE,
  levy_increments = NULL,
  seed = NULL
)
```

Arguments

N	a number of points of the lfsm.
m	discretization. A number of points between two nearby motion points
M	truncation parameter. A number of points at which the integral representing the definition of lfsm is calculated. So, after M points back we consider the rest of the integral to be 0.
alpha	self-similarity parameter of alpha stable random motion.
H	Hurst parameter
sigma	Scale parameter of lfsm
freq	Frequency of the motion. It can take two values: "H" for high frequency and "L" for the low frequency setting.
disable_X	is needed to disable computation of X. The default value is FALSE. When it is TRUE, only a levy motion is returned, which in turn reduces the computation time. The feature is particularly useful for reproducibility when combined with seeding.
levy_increments	increments of Levy motion underlying the lfsm.
seed	this parameter performs seeding of path generator

Value

It returns a list containing the motion, the underlying Levy motion, the point number of the motions from 0 to N and the corresponding coordinate (which depends on the frequency), the parameters that were used to generate the lfsm, and the predefined frequency.

References

Mazur S, Otryakhin D (2020). "Linear Fractional Stable Motion with the rlfsm R Package." *The R Journal*, **12**(1), 386–405. doi:10.32614/RJ2020008.

Stoev S, Taqqu MS (2004). "Simulation methods for linear fractional stable motion and FARIMA using the Fast Fourier Transform." *Fractals*, **95**(1), 95-121. `https://doi.org/10.1142/S0218348X04002379`.

See Also

`paths` simulates a number of lfsm sample paths.

Examples

```
# Path generation

m<-256; M<-600; N<-2^10-M
alpha<-1.8; H<-0.8; sigma<-0.3
seed=2

List<-path(N=N,m=m,M=M,alpha=alpha,H=H,
           sigma=sigma,freq='L',disable_X=FALSE,seed=3)

# Normalized paths
Norm_lfsm<-List[['lfsm']]/max(abs(List[['lfsm']]))
Norm_oLm<-List[['levy_motion']]/max(abs(List[['levy_motion']]))

# Visualization of the paths
plot(Norm_lfsm, col=2, type="l", ylab="coordinate")
lines(Norm_oLm, col=3)
leg.txt <- c("lfsm", "oLm")
legend("topright",legend = leg.txt, col =c(2,3), pch=1)


# Creating Levy motion
levyIncrems<-path(N=N, m=m, M=M, alpha, H, sigma, freq='L',
                  disable_X=TRUE, levy_increments=NULL, seed=seed)

# Creating lfsm based on the levy motion
  lfsm_full<-path(m=m, M=M, alpha=alpha,
                  H=H, sigma=sigma, freq='L',
                  disable_X=FALSE,
                  levy_increments=levyIncrems$levy_increments,
                  seed=seed)

sum(levyIncrems$levy_increments==
    lfsm_full$levy_increments)==length(lfsm_full$levy_increments)
```

paths *Generator of a set of lfsm paths.*

Description

It is essentially a wrapper for `path` generator, which exploits the latest to create a matrix with paths in its columns.

Usage

```
paths(N_var, parallel, seed_list = rep(x = NULL, times = N_var), ...)
```

Arguments

`N_var`	number of lfsm paths to generate
`parallel`	a TRUE/FALSE flag which determines if the paths will be created in parallel or sequentially
`seed_list`	a numerical vector of seeds to pass to `path`
`...`	arguments to pass to path

See Also

`path`

Examples

```
m<-45; M<-60; N<-2^10-M
alpha<-1.8; H<-0.8; sigma<-0.3
freq='L'
r=1; k=2; p=0.4

Y<-paths(N_var=10,parallel=TRUE,N=N,m=m,M=M,
         alpha=alpha,H=H,sigma=sigma,freq='L',
         disable_X=FALSE,levy_increments=NULL)

Hs<-apply(Y,MARGIN=2,H_hat,p=p,k=k)
hist(Hs)
```

`Path_array` *Path array generator*

Description

The function takes a list of parameters (alpha, H) and uses `expand.grid` to obtain all possible combinations of them. Based on each combination, the function simulates an lfsm sample path. It is meant to be used in conjunction with function `Plot_list_paths`.

Usage

```
Path_array(N, m, M, l, sigma)
```

Arguments

`N`	a number of points of the lfsm.
`m`	discretization. A number of points between two nearby motion points
`M`	truncation parameter. A number of points at which the integral representing the definition of lfsm is calculated. So, after M points back we consider the rest of the integral to be 0.
`l`	a list of parameters to expand
`sigma`	Scale parameter of lfsm

Value

The returned value is a data frame containing paths and the corresponding values of alpha, H and frequency.

Examples

```
l=list(H=c(0.2,0.8),alpha=c(1,1.8), freq="H")
arr<-Path_array(N=300,m=30,M=100,l=l,sigma=0.3)
str(arr)
head(arr)
```

`phi` *Phi*

Description

Defined as

$$\varphi_{\text{high}}(t;H,k)_n := V_{\text{high}}(\psi_t;k)_n \qquad \text{and} \qquad \varphi_{\text{low}}(t;k)_n := V_{\text{low}}(\psi_t;k)_n$$

, where $\psi_t(x) := cos(tx)$

Usage

```
phi(t, k, path, H, freq)
```

Arguments

`t`	positive real number
`k`	increment order
`path`	sample path of lfsm on which the inference is to be performed
`H`	Hurst parameter
`freq`	Frequency of the motion. It can take two values: "H" for high frequency and "L" for the low frequency setting.

Details

Hurst parameter is required only in high frequency case. In the low frequency, there is no need to assign H a value because it will not be evaluated.

References

Mazur S, Otryakhin D, Podolskij M (2020). "Estimation of the linear fractional stable motion." *Bernoulli*, **26**(1), 226–252. https://doi.org/10.3150/19-BEJ1124.

`phi_of_alpha` *Inverse alpha estimator*

Description

A function from a general estimation procedure which is defined as m^p_-p'_k /m^p'_-p_k, originally proposed in [13].

Usage

```
phi_of_alpha(p, p_prime, alpha)
```

Arguments

`p`	power
`p_prime`	power
`alpha`	self-similarity parameter of alpha stable random motion.

References

Mazur S, Otryakhin D, Podolskij M (2020). "Estimation of the linear fractional stable motion." *Bernoulli*, **26**(1), 226–252. https://doi.org/10.3150/19-BEJ1124.

`Plot_dens` *(alpha,H,sigma)- density plot*

Description

Plots the densities of the parameters (alpha,H,sigma) estimated in Monte-Carlo experiment. Works in conjunction with `MCestimLFSM` function.

Usage

```
Plot_dens(par_vec = c("alpha", "H", "sigma"), MC_data, Nnorm = 1e+07)
```

Arguments

`par_vec`	vector of parameters which are to be plotted
`MC_data`	a list created by `MCestimLFSM`
`Nnorm`	number of point sampled from standard normal distribution

See Also

`Plot_vb` to plot variance- and bias dependencies on n.

Examples

```
m<-45; M<-60

p<-.4; p_prime<-.2
t1<-1; t2<-2; k<-2

NmonteC<-5e2
S<-c(1e3,1e4)
alpha<-.8; H<-0.8; sigma<-0.3
theor_4_1_clt_new<-MCestimLFSM(s=S,fr='L',Nmc=NmonteC,
                       m=m,M=M,
                       alpha=alpha,H=H,sigma=sigma,
                       GenLowEstim,t1=t1,t2=t2,p=p)
l_plot<-Plot_dens(par_vec=c('sigma','alpha','H'), MC_data=theor_4_1_clt_new, Nnorm=1e7)
l_plot
```

Plot_list_paths *Rendering of path lattice*

Description

Rendering of path lattice

Usage

```
Plot_list_paths(arr)
```

Arguments

arr	a data frame produced by Path_array.

Examples

```
l=list(H=c(0.2,0.8),alpha=c(1,1.8), freq="H")
arr<-Path_array(N=300,m=30,M=100,l=l,sigma=0.3)
p<-Plot_list_paths(arr)
p
```

Plot_vb *A function to plot variance- and bias dependencies of estimators on the lengths of sample paths. Works in conjunction with MCestimLFSM function.*

Description

A function to plot variance- and bias dependencies of estimators on the lengths of sample paths. Works in conjunction with MCestimLFSM function.

Usage

```
Plot_vb(data)
```

Arguments

data	a list created by MCestimLFSM

Value

The function returns a ggplot2 graph.

See Also

Plot_dens

Examples

```
# Light weight computaions

m<-25; M<-50
alpha<-1.8; H<-0.8; sigma<-0.3
S<-c(1:3)*1e2
p<-.4; p_prime<-.2; t1<-1; t2<-2
k<-2; NmonteC<-50

# Here is the continuous H-1/alpha inference procedure
theor_3_1_H_clt<-MCestimLFSM(s=S,fr='H',Nmc=NmonteC,
                     m=m,M=M,alpha=alpha,H=H,
                     sigma=sigma,ContinEstim,
                     t1=t1,t2=t2,p=p,k=k)
Plot_vb(theor_3_1_H_clt)


# More demanding example (it is better to use multicore setup)
# General low frequency inference

m<-45; M<-60
alpha<-0.8; H<-0.8; sigma<-0.3
S<-c(1:15)*1e2
p<-.4; t1<-1; t2<-2
NmonteC<-50

# Here is the continuous H-1/alpha inference procedure
theor_4_1_H_clt<-MCestimLFSM(s=S,fr='H',Nmc=NmonteC,
                     m=m,M=M,alpha=alpha,H=H,
                     sigma=sigma,GenLowEstim,
                     t1=t1,t2=t2,p=p)
Plot_vb(theor_4_1_H_clt)
```

Retrieve_stats *Retrieve statistics(bias, variance) of estimators based on a set of paths*

Description

Retrieve statistics(bias, variance) of estimators based on a set of paths

Usage

```
Retrieve_stats(paths, true_val, Est, ...)
```

Arguments

paths	real-valued matrix representing sample paths of the stochastic process being studied
true_val	true value of the estimated parameter
Est	estimator (i.e. H_hat)
...	parameters to pass to Est

Examples

```
m<-45; M<-60; N<-2^10-M
alpha<-1.8; H<-0.8; sigma<-0.3
freq='L';t1=1; t2=2
r=1; k=2; p=0.4

Y<-paths(N_var=10,parallel=TRUE,N=N,m=m,M=M,
         alpha=alpha,H=H,sigma=sigma,freq='L',
         disable_X=FALSE,levy_increments=NULL)

Retrieve_stats(paths=Y,true_val=sigma,Est=sigma_hat,t1=t1,k=2,alpha=alpha,H=H,freq="L")
```

R_hl *R high /low*

Description

Defined as

$$R_{\text{high}}(p,k)_n := \frac{\sum_{i=2k}^{n} \left|\Delta_{i,k}^{n,2} X\right|^p}{\sum_{i=k}^{n} \left|\Delta_{i,k}^{n,1} X\right|^p},$$

$$R_{\text{low}}(p,k)_n := \frac{\sum_{i=2k}^{n} \left|\Delta_{i,k}^{2} X\right|^p}{\sum_{i=k}^{n} \left|\Delta_{i,k}^{1} X\right|^p}$$

Usage

```
R_hl(p, k, path)
```

Arguments

p	power
k	increment order
path	sample path of lfsm on which the inference is to be performed

Details

The computation procedure for high- and low frequency cases is the same, since there is no way to control frequency given a sample path.

References

Mazur S, Otryakhin D, Podolskij M (2020). "Estimation of the linear fractional stable motion." *Bernoulli*, **26**(1), 226–252. https://doi.org/10.3150/19-BEJ1124.

Examples

```
m<-45; M<-60; N<-2^10-M
alpha<-0.8; H<-0.8; sigma<-0.3
p<-0.3; k=3

lfsm<-path(N=N,m=m,M=M,alpha=alpha,H=H,
           sigma=sigma,freq='L',disable_X=FALSE,seed=3)$lfsm
R_hl(p=p,k=k,path=lfsm)
```

---

sf *Statistic V*

Description

Statistic of the form

$$V_{\text{high}}(f;k,r)_n := \frac{1}{n}\sum_{i=rk}^{n} f\left(n^H \Delta_{i,k}^{n,r} X\right),$$

$$V_{\text{low}}(f;k,r)_n := \frac{1}{n}\sum_{i=rk}^{n} f\left(\Delta_{i,k}^{r} X\right)$$

Usage

```
sf(path, f, k, r, H, freq, ...)
```

Arguments

path	sample path for which the statistic is to be calculated.
f	function applied to high order increments.
k	order of the increments.
r	step of high order increments.
H	Hurst parameter.
freq	frequency.
...	parameters to pass to function f

Details

Hurst parameter is required only in high frequency case. In the low frequency, there is no need to assign H a value because it will not be evaluated.

References

Mazur S, Otryakhin D, Podolskij M (2020). "Estimation of the linear fractional stable motion." *Bernoulli*, **26**(1), 226–252. https://doi.org/10.3150/19-BEJ1124.

See Also

phi computes V statistic with f(.)=cos(t.)

Examples

```
m<-45; M<-60; N<-2^10-M
alpha<-1.8; H<-0.8; sigma<-0.3
freq='L'
r=1; k=2; p=0.4
S<-(1:20)*100

path_lfsm<-function(...){

    List<-path(...)
    List$lfsm

}

Pths<-lapply(X=S,FUN=path_lfsm,
             m=m, M=M, alpha=alpha, sigma=sigma, H=H,
             freq=freq, disable_X = FALSE,
             levy_increments = NULL, seed = NULL)

f_phi<-function(t,x) cos(t*x)
f_pow<-function(x,p) (abs(x))^p

V_cos<-sapply(Pths,FUN=sf,f=f_phi,k=k,r=r,H=H,freq=freq,t=1)
ex<-exp(-(abs(sigma*Norm_alpha(h_kr,alpha=alpha,k=k,r=r,H=H,l=0)$result)^alpha))

 # Illustration of the law of large numbers for phi:
plot(y=V_cos, x=S, ylim = c(0,max(V_cos)+0.1))
abline(h=ex, col='brown')

# Illustration of the law of large numbers for power functions:
Mpk<-m_pk(k=k, p=p, alpha=alpha, H=H, sigma=sigma)

sf_mod<-function(Xpath,...) {

    Path<-unlist(Xpath)
    sf(path=Path,...)
}
```

```
V_pow<-sapply(Pths,FUN=sf_mod,f=f_pow,k=k,r=r,H=H,freq=freq,p=p)
plot(y=V_pow, x=S, ylim = c(0,max(V_pow)+0.1))
abline(h=Mpk, col='brown')
```

---

sigma_hat *Statistical estimator for sigma*

Description

Statistical estimator for sigma

Usage

```
sigma_hat(t1, k, path, alpha, H, freq)
```

Arguments

t1	real number such that t1 > 0
k	increment order
path	sample path of lfsm on which the inference is to be performed
alpha	self-similarity parameter of alpha stable random motion.
H	Hurst parameter
freq	Frequency of the motion. It can take two values: "H" for high frequency and "L" for the low frequency setting.

Examples

```
m<-45; M<-60; N<-2^14-M
alpha<-1.8; H<-0.8; sigma<-0.3
freq='H'
r=1; k=2; p=0.4; t1=1; t2=2

# Reproducing the work of ContinEstim
# in high frequency case
lfsm<-path(N=N,m=m,M=M,alpha=alpha,H=H,
           sigma=sigma,freq='L',disable_X=FALSE,seed=1)$lfsm

H_est<-H_hat(p=p,k=k,path=lfsm)
H_est
alpha_est<-alpha_hat(t1=t1,t2=t2,k=k,path=lfsm,H=H_est,freq=freq)
alpha_est

sigma_est<-tryCatch(
                    sigma_hat(t1=t1,k=k,path=lfsm,
                    alpha=alpha_est,H=H_est,freq=freq),
                    error=function(c) 'Impossible to compute sigma_est')
sigma_est
```

theta	*Function theta*

Description

Function of the form

$$\theta(g,h)_p = a_p^{-2} \int_{\mathsf{R}^2} |xy|^{-1-p} U_{g,h}(x,y)dxdy$$

Usage

```
theta(p, alpha, sigma, g, h)
```

Arguments

`p`	power, real number from (-1,1)
`alpha`	self-similarity parameter of alpha stable random motion.
`sigma`	Scale parameter of lfsm
`g, h`	functions $g, h : \mathsf{R} \to \mathsf{R}$ with finite alpha-norm (see `Norm_alpha`).

References

Mazur S, Otryakhin D, Podolskij M (2020). "Estimation of the linear fractional stable motion." *Bernoulli*, **26**(1), 226–252. https://doi.org/10.3150/19-BEJ1124.

U_g	*alpha norm of u*g*

Description

alpha norm of u*g

Usage

```
U_g(g, u, ...)
```

Arguments

`g`	function $g : \mathsf{R} to \mathsf{R}$ with finite alpha-norm (see `Norm_alpha`).
`u`	real number
`...`	additional parameters to pass to Norm_alpha

Examples

```
g<-function(x) exp(-x^2)
g<-function(x) exp(-abs(x))
U_g(g=g,u=4,alpha=1.7)
```

U_gh — *alpha-norm of u*g + v*h.*

Description

alpha-norm of u*g + v*h.

Usage

```
U_gh(g, h, u, v, ...)
```

Arguments

g, h	functions $g, h : \mathsf{R} \to \mathsf{R}$ with finite alpha-norm (see Norm_alpha).
v, u	real numbers
...	additional parameters to pass to Norm_alpha

Examples

```
g<-function(x) exp(-x^2)
h<-function(x) exp(-abs(x))
U_gh(g=g, h=h, u=4, v=3, alpha=1.7)
```

U_ghuv — *A dependence structure of 2 random variables.*

Description

It is used when random variables do not have finite second moments, and thus, the covariance matrix is not defined. For $X = \int_{\mathsf{R}} g_s dL_s$ and $Y = \int_{\mathsf{R}} h_s dL_s$ with $\|g\|_\alpha, \|h\|_\alpha < \infty$. Then the measure of dependence is given by $U_{g,h} : \mathsf{R}^2 \to \mathsf{R}$ via

$$U_{g,h}(u, v) = \exp(-\sigma^\alpha \|ug + vh\|_\alpha{}^\alpha) - \exp(-\sigma^\alpha (\|ug\|_\alpha{}^\alpha + \|vh\|_\alpha{}^\alpha))$$

Usage

```
U_ghuv(alpha, sigma, g, h, u, v, ...)
```

Arguments

alpha	self-similarity parameter of alpha stable random motion.
sigma	Scale parameter of lfsm
g, h	functions $g, h : \mathsf{R} \to \mathsf{R}$ with finite alpha-norm (see Norm_alpha).
v, u	real numbers
...	additional parameters to pass to U_gh and U_g

Examples

```
g<-function(x) exp(-x^2)
h<-function(x) exp(-abs(x))
U_ghuv(alpha=1.5, sigma=1, g=g, h=h, u=10, v=15,
rel.tol = .Machine$double.eps^0.25, abs.tol=1e-11)
```

Index