Package ‘ambit’

October 12, 2022

Type Package

Title Simulation and Estimation of Ambit Processes

Version 0.1.2

Description Simulation and estimation tools for
various types of ambit processes, including trawl processes and weighted
trawl processes.

Language en-GB

License GPL-3

Encoding UTF-8

RoxygenNote 7.2.1

Depends R (>= 4.0.0)

Imports base, fBasics, Rcpp, stats

LinkingTo Rcpp

Suggests ggplot2, knitr, latex2exp, rmarkdown, testthat (>= 3.0.0)

VignetteBuilder knitr

Config/testthat/edition 3

NeedsCompilation yes

Author Almut E. D. Veraart [aut, cre, cph]
(<https://orcid.org/0000-0001-8582-3652>)

Maintainer Almut E. D. Veraart <a.veraart@imperial.ac.uk>

Repository CRAN

Date/Publication 2022-08-19 11:40:05 UTC

R topics documented:

`acf_Exp` *Autocorrelation function of the exponential trawl function*

Description

This function computes the autocorrelation function associated with the exponential trawl function.

Usage

```
acf_Exp(x, lambda)
```

Arguments

`x`	The argument (lag) at which the autocorrelation function associated with the exponential trawl function will be evaluated
`lambda`	parameter in the exponential trawl

Details

The trawl function is parametrised by the parameter $\lambda > 0$ as follows:

$$g(x) = e^{\lambda x}, \text{ for } x \le 0.$$

Its autocorrelation function is given by:

$$r(x) = e^{-\lambda x}, \text{ for } x \ge 0.$$

Value

The autocorrelation function of the exponential trawl function evaluated at x

Examples

```
acf_Exp(1,0.1)
```

`acf_LM` *Autocorrelation function of the long memory trawl function*

Description

This function computes the autocorrelation function associated with the long memory trawl function.

Usage

```
acf_LM(x, alpha, H)
```

Arguments

`x`	The argument (lag) at which the autocorrelation function associated with the long memory trawl function will be evaluated
`alpha`	parameter in the long memory trawl
`H`	parameter in the long memory trawl

Details

The trawl function is parametrised by the two parameters $H > 1$ and $\alpha > 0$ as follows:

$$g(x) = (1 - x/\alpha)^{-H}, \text{ for } x \le 0.$$

Its autocorrelation function is given by

$$r(x) = (1 + x/\alpha)^{(1-H)}, \text{ for } x \ge 0.$$

Value

The autocorrelation function of the long memory trawl function evaluated at x

Examples

```
acf_LM(1,0.3,1.5)
```

acf_supIG *Autocorrelation function of the supIG trawl function*

Description

This function computes the autocorrelation function associated with the supIG trawl function.

Usage

```
acf_supIG(x, delta, gamma)
```

Arguments

x	The argument (lag) at which the autocorrelation function associated with the supIG trawl function will be evaluated
delta	parameter in the supIG trawl
gamma	parameter in the supIG trawl

Details

The trawl function is parametrised by the two parameters $\delta \geq 0$ and $\gamma \geq 0$ as follows:

$$g(x) = (1 - 2x\gamma^{-2})^{-1/2} \exp(\delta\gamma(1 - (1 - 2x\gamma^{-2})^{1/2})), \text{ for } x \leq 0.$$

It is assumed that δ and γ are not simultaneously equal to zero. Its autocorrelation function is given by:

$$r(x) = \exp(\delta\gamma(1 - \sqrt{1 + 2x/\gamma^2})), \text{ for } x \geq 0.$$

Value

The autocorrelation function of the supIG trawl function evaluated at x

Examples

```
acf_supIG(1,0.3,0.1)
```

AddSlices_Rcpp — *Add slices and return vector of the sums of slices*

Description

Add slices and return vector of the sums of slices

Usage

```
AddSlices_Rcpp(slicematrix)
```

Arguments

slicematrix	A matrix of slices.

Value

Returns the vector of the sums of the slices

AddWeightedSlices_Rcpp — *Add slices and return vector of the weighted sums of slices*

Description

Add slices and return vector of the weighted sums of slices

Usage

```
AddWeightedSlices_Rcpp(slicematrix, weightvector)
```

Arguments

slicematrix	A matrix of slices.
weightvector	A vector of weights.

Value

Returns the vector of the weighted sums of the slices `asymptotic_variance` *Computing the true asymptotic variance in the CLT of the trawl estimation*

Description

This function computes the theoretical asymptotic variance appearing in the CLT of the trawl process for a given trawl function and fourth cumulant.

Usage

```
asymptotic_variance(t, c4, varlevyseed = 1, trawlfct, trawlfct_par)
```

Arguments

`t`	Time point at which the asymptotic variance is computed
`c4`	The fourth cumulant of the Levy seed of the trawl process
`varlevyseed`	The variance of the Levy seed of the trawl process, the default is 1
`trawlfct`	The trawl function for which the asymptotic variance will be computed (Exp, supIG or LM)
`trawlfct_par`	The parameter vector of the trawl function (Exp: lambda, supIG: delta, gamma, LM: alpha, H)

Details

As derived in Sauri and Veraart (2022), the asymptotic variance in the central limit theorem for the trawl function estimation is given by

$$\sigma_a^2(t) = c_4(L')a(t) + 2\{\int_0^\infty a(s)^2 ds + \int_0^t a(t-s)a(t+s)ds - \int_t^\infty a(s-t)a(t+s)ds\},$$

for $t > 0$. The integrals in the above formula are approximated numerically.

Value

The function returns $\sigma_a^2(t)$.

Examples

```
#Compute the asymptotic variance at time t for an exponential trawl with
#parameter 2; here we assume that the fourth cumulant equals 1.
av<-asymptotic_variance(t=1, c4=1, varlevyseed=1, trawlfct="Exp", trawlfct_par=2)
#Print the av
av$v
#Print the four components of the asymptotic variance separately
av$v1
av$v2
av$v3
```

```
av$v4

#Note that v=v1+v2+v3+v4
av$v
av$v1+av$v2+av$v3+av$v4
```

---

asymptotic_variance_est *Estimating the asymptotic variance in the trawl function CLT*

Description

This function estimates the asymptotic variance which appears in the CLT for the trawl function estimation.

Usage

```
asymptotic_variance_est(t, c4, varlevyseed = 1, Delta, avector, N = NULL)
```

Arguments

t	The time point at which to compute the asymptotic variance
c4	The fourth cumulant of the Levy seed of the trawl process
varlevyseed	The variance of the Levy seed of the trawl process, the default is 1
Delta	The width Delta of the observation grid
avector	The vector $(\hat{a}(0), \hat{a}(\Delta_n), ..., \hat{a}((n-1)\Delta_n))$
N	The optional parameter to specify the upper bound N_n in the computations of the estimators

Details

As derived in Sauri and Veraart (2022), the estimated asymptotic variance is given by

$$\hat{\sigma}_a^2(t) = \hat{v}_1(t) + \hat{v}_2(t) + \hat{v}_3(t) + \hat{v}_4(t),$$

where

$$\hat{v}_1(t) := \widehat{c_4(L')}\hat{a}(t) = RQ_n\hat{a}(t)/\hat{a}(0),$$

for

$$RQ_n := \frac{1}{\sqrt{2n\Delta_n}} \sum_{k=0}^{n-2} (X_{(k+1)\Delta_n} - X_{k\Delta_n})^4,$$

and

$$\hat{v}_2(t) := 2\sum_{l=0}^{N_n} \hat{a}^2(l\Delta_n)\Delta_n,$$

$$\hat{v}_3(t) := 2 \sum_{l=0}^{\min\{i,n-1-i\}} \hat{a}((i-l)\Delta_n)\hat{a}((i+l)\Delta_n)\Delta_n,$$

$$\hat{v}_4(t) := -2 \sum_{l=i}^{N_n-i} \hat{a}((l-i)\Delta_n)\hat{a}((i+l)\Delta_n)\Delta_n.$$

Value

The estimated asymptotic variance $\hat{v} = \hat{\sigma}_a^2(t)$ and its components $\hat{v}_1, \hat{v}_2, \hat{v}_3, \hat{v}_4$.

Examples

```
##Simulate a trawl process
##Determine the sampling grid
my_n <- 1000
my_delta <- 0.1
my_t <- my_n*my_delta

###Choose the model parameter
#Exponential trawl function:
my_lambda <- 2
#Poisson marginal distribution trawl
my_v <- 1

#Set the seed
set.seed(123)
#Simulate the trawl process
Poi_data <- sim_weighted_trawl(my_n, my_delta,
                               "Exp", my_lambda, "Poi", my_v)$path

#Estimate the trawl function
my_lag <- 100+1
trawl <- nonpar_trawlest(Poi_data, my_delta, lag=my_lag)$a_hat

#Estimate the fourth cumulant of the trawl process
c4_est <- c4est(Poi_data, my_delta)

asymptotic_variance_est(t=1, c4=c4_est, varlevyseed=1,
                        Delta=my_delta, avector=trawl)$v
```

---

c4est *Estimating the fourth cumulant of the trawl process*

Description

This function estimates the fourth cumulant of the trawl process.

Usage

```
c4est(data, Delta)
```

Arguments

`data`	The data set used to estimate the fourth cumulant
`Delta`	The width Delta of the observation grid

Details

According to Sauri and Veraart (2022), estimator based on $X_0, X_{\Delta_n}, \ldots, X_{(n-1)\Delta_n}$ is given by

$$\hat{c}_4(L') = RQ_n/\hat{a}(0),$$

where

$$RQ_n := \frac{1}{\sqrt{2n\Delta_n}} \sum_{k=0}^{n-2} (X_{(k+1)\Delta_n} - X_{k\Delta_n})^4,$$

and

$$\hat{a}(0) = \frac{1}{2\Delta_n n} \sum_{k=0}^{n-2} (X_{(k+1)\Delta_n} - X_{k\Delta_n})^2.$$

Value

The function returns the estimated fourth cumulant of the Levy seed: $\hat{c}_4(L')$.

Examples

```
##Simulate a trawl process
##Determine the sampling grid
my_n <- 1000
my_delta <- 0.1
my_t <- my_n*my_delta

###Choose the model parameter
#Exponential trawl function:
my_lambda <- 2
#Poisson marginal distribution trawl
my_v <- 1

#Set the seed
set.seed(123)
#Simulate the trawl process
Poi_data<-ambit::sim_weighted_trawl(my_n, my_delta, "Exp", my_lambda, "Poi", my_v)$path

#Estimate the fourth cumulant of the trawl process
c4est(Poi_data, my_delta)
```

LebA_est *Nonparametric estimation of the trawl set Leb(A)*

Description

This function estimates the size of the trawl set given by Leb(A).

Usage

```
LebA_est(data, Delta, biascor = FALSE)
```

Arguments

data	Data to be used in the trawl function estimation.
Delta	Width of the grid on which we observe the data
biascor	A binary variable determining whether a bias correction should be computed, the default is FALSE

Details

Estimation of the trawl function using the methodology proposed in Sauri and Veraart (2022).

Value

The estimated Lebesgue measure of the trawl set

Examples

```
##Simulate a trawl process
##Determine the sampling grid
my_n <- 5000
my_delta <- 0.1
my_t <- my_n*my_delta

###Choose the model parameter
#Exponential trawl function:
my_lambda <- 2
#Poisson marginal distribution trawl
my_v <- 1

#Set the seed
set.seed(1726)
#Simulate the trawl process
Poi_data<-ambit::sim_weighted_trawl(my_n, my_delta, "Exp", my_lambda, "Poi", my_v)$path

#Estimate the trawl set without bias correction
LebA1 <-LebA_est(Poi_data, my_delta)
LebA1
```

```
#Estimate the trawl set with bias correction
LebA2 <-LebA_est(Poi_data, my_delta, biascor=TRUE)
LebA2

#Note that Leb(A)=1/my_lambda for an exponential trawl
```

`LebA_slice_est` *Nonparametric estimation of the trawl (sub-) sets Leb(A), Leb(A intersection A_h), Leb(A setdifference A_h)*

Description

This function estimates Leb(A), Leb(A intersection A_h), Leb(A\ A_h).

Usage

```
LebA_slice_est(data, Delta, h, biascor = FALSE)
```

Arguments

`data`	Data to be used in the trawl function estimation.
`Delta`	Width of the grid on which we observe the data
`h`	Time point used in A intersection A_h and the setdifference A setdifference A_h
`biascor`	A binary variable determining whether a bias correction should be computed, the default is FALSE

Details

Estimation of the trawl function using the methodology proposed in Sauri and Veraart (2022).

Value

LebA

LebAintersection

LebAsetdifference

Examples

```
##Simulate a trawl process
##Determine the sampling grid
my_n <- 5000
my_delta <- 0.1
my_t <- my_n*my_delta
```

```
###Choose the model parameter
#Exponential trawl function:
my_lambda <- 2
#Poisson marginal distribution trawl
my_v <- 1

#Set the seed
set.seed(1726)
#Simulate the trawl process
Poi_data<-ambit::sim_weighted_trawl(my_n, my_delta, "Exp", my_lambda, "Poi", my_v)$path

#Estimate the trawl set and its two slices at time h=2 without bias correction
est1 <- LebA_slice_est(Poi_data, my_delta, h=2)
est1$LebA
est1$LebAintersection
est1$LebAsetdifference

#Estimate the trawl set and its two slices at time h=2 without bias correction
est2 <- LebA_slice_est(Poi_data, my_delta, h=2, biascor=TRUE)
est2$LebA
est2$LebAintersection
est2$LebAsetdifference

#Note that Leb(A)=1/my_lambda for an exponential trawl
```

LebA_slice_ratio_est_acfbased *Nonparametric estimation of the ratios Leb(A intersection A_h)/Leb(A), Leb(A setdifference A_h)/Leb(A)*

Description

This function estimates the ratios Leb(A intersection A_h)/Leb(A), Leb(A\ A_h)/Leb(A).

Usage

```
LebA_slice_ratio_est_acfbased(data, Delta, h)
```

Arguments

data	Data to be used in the trawl function estimation.
Delta	Width of the grid on which we observe the data
h	Time point used in A intersection A_h and the setdifference A setdifference A_h

Details

Estimation of the trawl function using the methodology proposed in Sauri and Veraart (2022) which is based on the empirical acf.

Value

LebAintersection_ratio: LebAintersection/LebA

LebAsetdifference_ratio: LebAsetdifference/LebA

Examples

```
##Simulate a trawl process
##Determine the sampling grid
my_n <- 5000
my_delta <- 0.1
my_t <- my_n*my_delta

###Choose the model parameter
#Exponential trawl function:
my_lambda <- 2
#Poisson marginal distribution trawl
my_v <- 1

#Set the seed
set.seed(1726)
#Simulate the trawl process
Poi_data<-ambit::sim_weighted_trawl(my_n, my_delta, "Exp", my_lambda, "Poi", my_v)$path

#Estimate the trawl set and its two slices at time h=0.5
est <- LebA_slice_ratio_est_acfbased(Poi_data, my_delta, h=0.5)
#Print the ratio LebAintersection/LebA
est$LebAintersection_ratio
#Print the ratio LebAsetdifference/LebA
est$LebAsetdifference_ratio
```

---

my_mae *my_mse*

Description

Returns the mean absolute error between two vectors

Usage

```
my_mae(x, y)
```

Arguments

x	vector
y	vector

Value

Mean absolute error between the two vectors x and y

Examples

```
#Simulate two vectors of i.i.d.~standard normal data
set.seed(456)
x <- rnorm(100)
y <- rnorm(100)
#Compute the mean absolute error between both vectors
my_mae(x,y)
```

---

my_mse *my_mse*

Description

Returns the mean squared error between two vectors

Usage

```
my_mse(x, y)
```

Arguments

x	vector
y	vector

Value

Mean square error between the two vectors x and y

Examples

```
#Simulate two vectors of i.i.d.~standard normal data
set.seed(456)
x <- rnorm(100)
y <- rnorm(100)
#Compute the mean squared error between both vectors
my_mse(x,y)
```

`my_results` *my_results*

Description

Returns summary statistics

Usage

```
my_results(x, sd = 1, digits = 3)
```

Arguments

`x`	data
`sd`	Optional parameter giving the standard deviation of the normal distribution used for computing the coverage probabilities
`digits`	Optional parameter to how many digits the results should be rounded, the default is three.

Details

This functions returns the sample mean, sample standard deviation and the coverage probabilities at level 75%, 80%, 85%, 90%, 95%, 99% compared to the standard normal quantiles.

Value

The vector of the sample mean, sample standard deviation and the coverage probabilities at level 75%, 80%, 85%, 90%, 95%, 99% compared to the standard normal quantiles.

Examples

```
#Simulate i.i.d.~standard normal data
set.seed(456)
data <- rnorm(10000)
#Display the sample mean, standard deviation and coverage probabilities:
my_results(data)
```

`nonpar_trawlest` *Nonparametric estimation of the trawl function*

Description

This function implements the nonparametric trawl estimation proposed in Sauri and Veraart (2022).

Usage

```
nonpar_trawlest(data, Delta, lag = 100)
```

Arguments

`data`	Data to be used in the trawl function estimation.
`Delta`	Width of the grid on which we observe the data
`lag`	The lag until which the trawl function should be estimated

Details

Estimation of the trawl function using the methodology proposed in Sauri and Veraart (2022). Suppose the data is observed on the grid 0, Delta, 2Delta, ..., (n-1)Delta. Given the path contained in data, the function returns the lag-dimensional vector

$$(\hat{a}(0), \hat{a}(\Delta), \ldots, \hat{a}((lag-1)\Delta)).$$

In the case when lag=n, the n-1 dimensional vector

$$(\hat{a}(0), \hat{a}(\Delta), \ldots, \hat{a}((n-2)\Delta))$$

is returned.

Value

ahat Returns the lag-dimensional vector $(\hat{a}(0), \hat{a}(\Delta), \ldots, \hat{a}((lag-1)\Delta))$. Here, $\hat{a}(0)$ is estimated based on the realised variance estimator.

a0_alt Returns the alternative estimator of a(0) using the same methodology as the one used for t>0. Note that this is not the recommended estimator to use, but can be used for comparison purposes.

Examples

```
##Simulate a trawl process
##Determine the sampling grid
my_n <- 5000
my_delta <- 0.1
my_t <- my_n*my_delta

###Choose the model parameter
```

```
#Exponential trawl function:
my_lambda <- 2
#Poisson marginal distribution trawl
my_v <- 1

#Set the seed
set.seed(1726)
#Simulate the trawl process
Poi_data<-ambit::sim_weighted_trawl(my_n, my_delta, "Exp", my_lambda, "Poi", my_v)$path

#Estimate the trawl function
my_lag <- 100+1
PoiEx_trawl <- nonpar_trawlest(Poi_data, my_delta, lag=my_lag)$a_hat

#Plot the estimated trawl function and superimpose the true one
l_seq <- seq(from = 0,to = (my_lag-1), by = 1)
esttrawlfct.data <- base::data.frame(l=l_seq[1:31],
                              value=PoiEx_trawl[1:31])
p1 <- ggplot2::ggplot(esttrawlfct.data, ggplot2::aes(x=l,y=value))+
  ggplot2::geom_point(size=3)+
 ggplot2::geom_function(fun = function(x) acf_Exp(x*my_delta,my_lambda), colour="red", size=1.5)+
  ggplot2::xlab("l")+
  ggplot2::ylab(latex2exp::TeX("$\\hat{a}(\\cdot)$ for Poisson trawl process"))
p1
```

---

rq *Computing the scaled realised quarticity*

Description

This function computes the scaled realised quarticity of a time series for a given width of the observation grid.

Usage

```
rq(data, Delta)
```

Arguments

`data`	The data set used to compute the scaled realised quarticity
`Delta`	The width Delta of the observation grid

Details

According to Sauri and Veraart (2022), the scaled realised quarticity for $X_0, X_{\Delta_n}, \ldots, X_{(n-1)\Delta_n}$ is given by

$$RQ_n := \frac{1}{\sqrt{2n\Delta_n}} \sum_{k=0}^{n-2} (X_{(k+1)\Delta_n} - X_{k\Delta_n})^4.$$

Value

The function returns the scaled realised quarticity RQ_n.

Examples

```
##Simulate a trawl process
##Determine the sampling grid
my_n <- 1000
my_delta <- 0.1
my_t <- my_n*my_delta

###Choose the model parameter
#Exponential trawl function:
my_lambda <- 2
#Poisson marginal distribution trawl
my_v <- 1

#Set the seed
set.seed(123)
#Simulate the trawl process
Poi_data<-ambit::sim_weighted_trawl(my_n, my_delta, "Exp", my_lambda, "Poi", my_v)$path

#Compute the scaled realised quarticity
rq(Poi_data, my_delta)
```

---

sim_weighted_trawl *Simulation of a weighted trawl process*

Description

This function simulates a weighted trawl process for various choices of the trawl function and the marginal distribution.

Usage

```
sim_weighted_trawl(
  n,
  Delta,
  trawlfct,
  trawlfct_par,
  distr,
  distr_par,
  kernelfct = NULL
)
```

Arguments

n	number of grid points to be simulated (excluding the starting value)
Delta	grid-width
trawlfct	the trawl function a used in the simulation (Exp, supIG or LM)
trawlfct_par	parameter vector of trawl function (Exp: lambda, supIG: delta, gamma, LM: alpha, H)
distr	marginal distribution. Choose from "Gamma" (Gamma), "Gauss" (Gaussian), "Cauchy" (Cauchy), "NIG" (Normal Inverse Gaussian), Poi" (Poisson), "NegBin" (Negative Binomial)
distr_par	parameters of the marginal distribution: (Gamma: shape, scale; Gauss: mu, sigma (i.e. the second parameter is the standard deviation, not the variance); Cauchy: l, s; NIG: alpha, beta, delta, mu; Poi: v, NegBin: m, theta)
kernelfct	the kernel function p used in the ambit process

Details

This functions simulates a sample path from a weighted trawl process given by

$$Y_t = \int_{(-\infty,t]\times(-\infty,\infty)} p(t-s) I_{(0,a(t-s))}(x) L(dx, ds),$$

for $t \geq 0$, and returns $Y_0, Y_\Delta, \ldots, Y_{n\Delta}$.

Value

path Simulated path

slice_sizes slice sizes used

S_matrix Matrix of all slices

kernelweights kernel weights used

Examples

```
#Simulation of a Gaussian trawl process with exponential trawl function
n<-2000
Delta<-0.1
trawlfct="Exp"
trawlfct_par <-0.5
distr<-"Gauss"
distr_par<-c(0,1) #mean 0, std 1
set.seed(233)
path <- sim_weighted_trawl(n, Delta, trawlfct, trawlfct_par, distr, distr_par)$path
#Plot the path
library(ggplot2)
df <- data.frame(time = seq(0,n,1), value=path)
p <- ggplot(df, aes(x=time, y=path))+
  geom_line()+
  xlab("l")+
```

```
ylab("Trawl process")
p
```

---

sim_weighted_trawl_gen
Simulation of a weighted trawl process with generic trawl function

---

Description

This function simulates a weighted trawl process for a generic trawl function and various choices the marginal distribution. The specific trawl function to be used can be supplied directly by the user.

Usage

```
sim_weighted_trawl_gen(
  n,
  Delta,
  trawlfct_gen,
  distr,
  distr_par,
  kernelfct = NULL
)
```

Arguments

n	number of grid points to be simulated (excluding the starting value)
Delta	grid-width
trawlfct_gen	the trawl function a used in the simulation
distr	marginal distribution. Choose from "Gamma" (Gamma), "Gauss" (Gaussian), "Cauchy" (Cauchy), "NIG" (Normal Inverse Gaussian), Poi" (Poisson), "NegBin" (Negative Binomial)
distr_par	parameters of the marginal distribution: (Gamma: shape, scale; Gauss: mu, sigma (i.e. the second parameter is the standard deviation, not the variance); Cauchy: l, s; NIG: alpha, beta, delta, mu; Poi: v, NegBin: m, theta)
kernelfct	the kernel function p used in the ambit process

Details

This functions simulates a sample path from a weighted trawl process given by

$$Y_t = \int_{(-\infty,t]\times(-\infty,\infty)} p(t-s) I_{(0,a(t-s))}(x) L(dx,ds),$$

for $t \geq 0$, and returns $Y_0, Y_\Delta, \ldots, Y_{n\Delta}$. The user needs to ensure that trawlfct_gen is a monotonic function.

Value

path Simulated path

slice_sizes slice sizes used

S_matrix Matrix of all slices

kernelweights kernel weights used

Examples

```
#Simulation of a Gaussian trawl process with exponential trawl function
n<-2000
Delta<-0.1

trawlfct_par <-0.5
distr<-"Gauss"
distr_par<-c(0,1) #mean 0, std 1
set.seed(233)

a <- function(x){exp(-trawlfct_par*x)}
path <- sim_weighted_trawl_gen(n, Delta, a,
                           distr, distr_par)$path
#Plot the path
library(ggplot2)
df <- data.frame(time = seq(0,n,1), value=path)
p <- ggplot(df, aes(x=time, y=path))+
  geom_line()+
  xlab("l")+
  ylab("Trawl process")
p
```

`test_asymnorm` *Computing the infeasible test statistic from the trawl function estimation CLT*

Description

This function computes the infeasible test statistic appearing in the CLT for the trawl function estimation.

Usage

```
test_asymnorm(ahat, n, Delta, k, c4, varlevyseed = 1, trawlfct, trawlfct_par)
```

Arguments

`ahat`	The term $\hat{a}(k\Delta_n)$ in the CLT
`n`	The number n of observations in the sample
`Delta`	The width Delta of the observation grid
`k`	The time point in $0, 1, \ldots, n-1$; the test statistic will be computed for the time point $k * \Delta_n$.
`c4`	The fourth cumulant of the Levy seed of the trawl process
`varlevyseed`	The variance of the Levy seed of the trawl process, the default is 1
`trawlfct`	The trawl function for which the asymptotic variance will be computed (Exp, supIG or LM)
`trawlfct_par`	The parameter vector of the trawl function (Exp: lambda, supIG: delta, gamma, LM: alpha, H)

Details

As derived in Sauri and Veraart (2022), the infeasible test statistic is given by

$$\frac{\sqrt{n\Delta_n}}{\sqrt{\sigma_a^2(k\Delta_n)}} \left(\hat{a}(k\Delta_n) - a(k\Delta_n)\right),$$

for $k \in \{0, 1, \ldots, n-1\}$.

Value

The function returns the infeasible test statistic specified above.

Examples

```
test_asymnorm(ahat=0.9, n=5000, Delta=0.1, k=1, c4=1, varlevyseed=1,
              trawlfct="Exp", trawlfct_par=0.1)
```

`test_asymnorm_est` *Computing the feasible statistic of the trawl function CLT*

Description

This function computes the feasible statistics associated with the CLT for the trawl function estimation.

Usage

```
test_asymnorm_est(
  data,
  Delta,
  trawlfct,
  trawlfct_par,
  biascor = FALSE,
  k = NULL
)
```

Arguments

`data`	The data set based on observations of $X_0, X_{\Delta_n}, \ldots, X_{(n-1)\Delta_n}$
`Delta`	The width Delta of the observation grid
`trawlfct`	The trawl function for which the asymptotic variance will be computed (Exp, supIG or LM)
`trawlfct_par`	The parameter vector of the trawl function (Exp: lambda, supIG: delta, gamma, LM: alpha, H)
`biascor`	A binary variable determining whether a bias correction should be computed, the default is FALSE
`k`	The optional parameter specifying the time point in $0, 1, \ldots, n-1$; the test statistic will be computed for the time point $k\Delta_n$.

Details

As derived in Sauri and Veraart (2022), the feasible statistic, for $t > 0$, is given by

$$T(t)_n := \frac{\sqrt{n\Delta_n}}{\sqrt{\widehat{\sigma_a^2(t)}}} \left(\hat{a}(t) - a(t) - bias(t)\right).$$

For $t = 0$, we have

$$T(t)_n := \frac{\sqrt{n\Delta_n}}{\sqrt{RQ_n}} \left(\hat{a}(0) - a(0) - bias(0)\right),$$

where

$$RQ_n := \frac{1}{\sqrt{2n\Delta_n}} \sum_{k=0}^{n-2} (X_{(k+1)\Delta_n} - X_{k\Delta_n})^4.$$

We set $bias(t) = 0$ in the case when biascor==FALSE and $bias(t) = 0.5 * \Delta * \hat{a}'(t)$ otherwise.

Value

The function returns the vector of the feasible statistics $(T(0)_n, T((\Delta)_n, \ldots, T((n-2)\Delta_n))$ if no bias correction is required and $(T(0)_n, T((\Delta)_n, \ldots, T((n-3)\Delta_n))$ if bias correction is required if k is not provided, otherwise it returns the value $T(k\Delta_n)_n$. If the estimated asymptotic variance is <= 0, the value of the test statistic is set to 999.

Examples

```
##Simulate a trawl process
##Determine the sampling grid
my_n <- 1000
my_delta <- 0.1
my_t <- my_n*my_delta

###Choose the model parameter
#Exponential trawl function:
my_lambda <- 2
#Poisson marginal distribution trawl
my_v <- 1

#Set the seed
set.seed(123)
#Simulate the trawl process
Poi_data <- sim_weighted_trawl(my_n, my_delta,
                               "Exp", my_lambda, "Poi", my_v)$path

#Compute the test statistic for time t=0
##Either one can use:
test_asymnorm_est(Poi_data, my_delta,
                  trawlfct="Exp", trawlfct_par=my_lambda)[1]
#or:
test_asymnorm_est(Poi_data, my_delta,
                  trawlfct="Exp", trawlfct_par=my_lambda, k=0)
```

`test_asymnorm_est_dev` *Computing the feasible statistic of the trawl function CLT*

Description

This function computes the feasible test statistic appearing in the CLT for the trawl function estimation.

Usage

```
test_asymnorm_est_dev(
  ahat,
  n,
  Delta,
  k,
  c4,
  varlevyseed = 1,
  trawlfct,
  trawlfct_par,
  avector
)
```

Arguments

ahat	The estimated trawl function at time t: $\hat{a}(t)$
n	The number of observations in the data set
Delta	The width Delta of the observation grid
k	The time point in $0, 1, \ldots, n-1$; the test statistic will be computed for the time point $k * \Delta_n$.
c4	The fourth cumulant of the Levy seed of the trawl process
varlevyseed	The variance of the Levy seed of the trawl process, the default is 1
trawlfct	The trawl function for which the asymptotic variance will be computed (Exp, supIG or LM)
trawlfct_par	The parameter vector of the trawl function (Exp: lambda, supIG: delta, gamma, LM: alpha, H)
avector	The vector $(\hat{a}(0), \hat{a}(Delta_n), ..., \hat{a}((n-1)\Delta_n))$

Details

As derived in Sauri and Veraart (2022), the feasible statistic is given by

$$T(k\Delta_n)_n := \frac{\sqrt{n\Delta_n}}{\sqrt{\widehat{\sigma_a^2(\Delta_n)}}} \left(\hat{a}(\Delta_n) - a(\Delta_n)\right)$$

.

Value

The function returns the feasible statistic $T(\Delta_n)_n$ if the estimated asymptotic variance is positive and 999 otherwise.

---

trawl_deriv *Estimating the derivative of the trawl function using the empirical derivative*

Description

This function estimates the derivative of the trawl function using the empirical derivative of the trawl function.

Usage

```
trawl_deriv(data, Delta, lag = 100)
```

Arguments

data	The data set used to compute the derivative of the trawl function
Delta	The width Delta of the observation grid
lag	The lag until which the trawl function should be estimated

Details

According to Sauri and Veraart (2022), the derivative of the trawl function can be estimated based on observations $X_0, X_{\Delta_n}, \ldots, X_{(n-1)\Delta_n}$ by

$$\widehat{a}(t) = \frac{1}{\Delta_n}(\hat{a}(t+\Delta_n) - \hat{a}(\Delta_n)),$$

for $\Delta_n l \leq t < (l+1)\Delta_n$.

Value

The function returns the lag-dimensional vector $(\hat{a}'(0), \hat{a}'(\Delta), \ldots, \hat{a}'((lag-1)\Delta))$.

Examples

```
##Simulate a trawl process
##Determine the sampling grid
my_n <- 1000
my_delta <- 0.1
my_t <- my_n*my_delta

###Choose the model parameter
#Exponential trawl function:
my_lambda <- 2
#Poisson marginal distribution trawl
my_v <- 1

#Set the seed
set.seed(123)
#Simulate the trawl process
Poi_data <- sim_weighted_trawl(my_n, my_delta,
                               "Exp", my_lambda, "Poi", my_v)$path

#Estimate the trawl function
my_lag <- 100+1
trawl <- nonpar_trawlest(Poi_data, my_delta, lag=my_lag)$a_hat

#Estimate the derivative of the trawl function
trawl_deriv <- trawl_deriv(Poi_data, my_delta, lag=100)
```

---

trawl_deriv_mod *Estimating the derivative of the trawl function*

Description

This function estimates the derivative of the trawl function using the modified version proposed in Sauri and Veraart (2022).

Usage

```
trawl_deriv_mod(data, Delta, lag = 100)
```

Arguments

data	The data set used to compute the derivative of the trawl function
Delta	The width Delta of the observation grid
lag	The lag until which the trawl function should be estimated

Details

According to Sauri and Veraart (2022), the derivative of the trawl function can be estimated based on observations $X_0, X_{\Delta_n}, \ldots, X_{(n-1)\Delta_n}$ by

$$\widehat{a}(t) = \frac{1}{\sqrt{n\Delta_n^2}} \sum_{k=l+1}^{n-2} (X_{(k+1)\Delta_n} - X_{k\Delta_n})(X_{(k-l+1)\Delta_n} - X_{(k-l)\Delta_n}),$$

for $\Delta_n l \leq t < (l+1)\Delta_n$.

Value

The function returns the lag-dimensional vector $(\hat{a}'(0), \hat{a}'(\Delta), \ldots, \hat{a}'((lag-1)\Delta))$.

Examples

```
##Simulate a trawl process
##Determine the sampling grid
my_n <- 1000
my_delta <- 0.1
my_t <- my_n*my_delta

###Choose the model parameter
#Exponential trawl function:
my_lambda <- 2
#Poisson marginal distribution trawl
my_v <- 1

#Set the seed
set.seed(123)
#Simulate the trawl process
Poi_data <- sim_weighted_trawl(my_n, my_delta,
                               "Exp", my_lambda, "Poi", my_v)$path

#Estimate the trawl function
my_lag <- 100+1
trawl <- nonpar_trawlest(Poi_data, my_delta, lag=my_lag)$a_hat

#Estimate the derivative of the trawl function
trawl_deriv <- trawl_deriv_mod(Poi_data, my_delta, lag=100)
```

trawl_Exp *Evaluates the exponential trawl function*

Description

Evaluates the exponential trawl function

Usage

```
trawl_Exp(x, lambda)
```

Arguments

x	the argument at which the exponential trawl function will be evaluated
lambda	the parameter λ in the exponential trawl

Details

The trawl function is parametrised by parameter $\lambda > 0$ as follows:

$$g(x) = e^{\lambda x}, \text{ for } x \leq 0.$$

Value

The exponential trawl function evaluated at x

Examples

```
trawl_Exp(-1,0.5)
```

trawl_LM *Evaluates the long memory trawl function*

Description

Evaluates the long memory trawl function

Usage

```
trawl_LM(x, alpha, H)
```

Arguments

x	the argument at which the supOU/long memory trawl function will be evaluated
alpha	the parameter α in the long memory trawl
H	the parameter H in the long memory trawl

Details

The trawl function is parametrised by the two parameters $H > 1$ and $\alpha > 0$ as follows:

$$g(x) = (1 - x/\alpha)^{-H}, \text{ for } x \leq 0.$$

If $H \in (1, 2]$, then the resulting trawl process has long memory, for $H > 2$, it has short memory.

Value

the long memory trawl function evaluated at x

Examples

```
trawl_LM(-1,0.5, 1.5)
```

`trawl_supIG` *Evaluates the supIG trawl function*

Description

Evaluates the supIG trawl function

Usage

```
trawl_supIG(x, delta, gamma)
```

Arguments

`x`	the argument at which the supIG trawl function will be evaluated
`delta`	the parameter δ in the supIG trawl
`gamma`	the parameter γ in the supIG trawl

Details

The trawl function is parametrised by the two parameters $\delta \geq 0$ and $\gamma \geq 0$ as follows:

$$gd(x) = (1 - 2x\gamma^{-2})^{-1/2} \exp(\delta\gamma(1 - (1 - 2x\gamma^{-2})^{1/2})), \text{ for } x \leq 0.$$

It is assumed that δ and γ are not simultaneously equal to zero.

Value

The supIG trawl function evaluated at x

Examples

```
trawl_supIG(-1,0.5,0.2)
```

Index