Package ‘adaptiveFTS’

November 14, 2024

Type Package

Version 0.1.0

Title Adaptive Estimation for Weakly Dependent Functional Time Series

Description Functions for adaptive estimation for weakly dependent functional time series that take into account Hölder continuous regularity of the sample paths.

Author Hassan Maissoro <hassan.maissoro@ensae.fr>

Maintainer Hassan Maissoro <hassan.maissoro@ensae.fr>

License GPL-3

Encoding UTF-8

Imports data.table,
matrixStats,
Rdpack,
MASS,
fastmatrix,
caret,
methods

RoxygenNote 7.3.2

Contents

`.constant_d` *Constant D(x,y) function*

Description

See the following paper https://doi.org/10.3390/fractalfract6020074.

Usage

```
.constant_d(x, y)
```

Arguments

x	`Float (positive)`. First argument of the function.
y	`Float (positive)`. Second argument of the function.

Value

A positive `Float` corresponding to the value the function evaluate at (x,y).

.covariance_mfBm — *Covariance matrix of the multi-fractional Brownian Motion*

Description

Covariance matrix of the multi-fractional Brownian Motion

Usage

```
.covariance_mfBm(t = seq(0.2, 0.8, len = 10), hurst_fun = hurst_logistic, ...)
```

Arguments

`t`	`vector (float)`. Points between 0 and 1 at which to compute the covariance function.
`hurst_fun`	`function`. Hurst function. It can be `hurst_arctan`, `hurst_linear`, `hurst_logistic`.
`...`	Hurst function additional arguments.

Value

a `matrix` of `t` x `t` covariance.

.format_data — *Format data for local regularity estimation*

Description

Format data for local regularity estimation

Usage

```
.format_data(data, idcol = NULL, tcol = "tobs", ycol = "X")
```

Arguments

`data` — `data.table` (or `data.frame`) or `list` of `data.table` (or `data.frame`) or `list` of `list`.

- If `data.table` It must contain the raw binding of the curve observations with at least 3 columns.
  - `idcol` : The name of the column that contains the index of the curve in the sample. Each index of a curve is repeated as many times as it has observation points.
  - `tcol` : The name of the column that contains the observation points associated to each curve index.
  - `ycol` : The name of the column that contains the observed value of a curve at each point of observation and for each index of the curve.
- If `list` of `data.table` In this case, each element of the given `list` corresponds to the observation scheme of a curve, which is given as `data.table` or `data.frame`. The data.table contains at least 2 columns.

- `tcol` : The name of the column that contains the observation points associated to the curve.
- `ycol` : The name of the column that contains the observed value of the curve.

- If `list` of `list` In the latter case, the `data` is a list `list` where each element is the observation scheme of a curve given as a `list` of 2 vectors.
  - `tcol` : The name of the vector that contains the observation points associated the curve.
  - `ycol` : The name of the vector that contains the observed value of the curve.

`idcol`	`character`. If `data` is given as `data.table` or `data.frame`, it is the name of the column that contains the index of the curve in the sample. Each index of a curve is repeated as many times as it has observation points. Opposite, if f `data` is given as `list of data.table (or data.frame) of list of list, idcol = NULL.`
`tcol`	`character`. The name of the column (or vector) that contains the observation points associated to the curves.
`ycol`	`character`. The name of the column that contains the observed value of the curves.

Value

A `data.table` containing 3 columns.

- id_curve : The index of the curve.
- tobs : The observation points associated to each curve `id_curve`.
- X : The observed values of the curve associated to `id_curve` at `tobs` observation points.

`.Qpq_fun` *Q_pq^ (ℓ), $(\ell \leq 0)$ function. See Rubìn and Panaretos (2020) Equation (B.7)*

Description

$Q_{pq}^{(\ell)}$, $(\ell \leq 0)$ function. See Rubìn and Panaretos (2020) Equation (B.7)

Usage

```
.Qpq_fun(
  data,
  idcol = "id_curve",
  tcol = "tobs",
  ycol = "X",
  s = 1/4,
  t = 1/2,
  lag = 1,
  p = 1,
  q = 1,
  h,
```

```
  dt_mean_rp = NULL,
  optbw_mean = NULL,
  smooth_ker = epanechnikov
)
```

Arguments

data
: data.table (or data.frame) or list of data.table (or data.frame) or list of list.

  - If data.table It must contain the raw binding of the curve observations with at least 3 columns.
    - idcol : The name of the column that contains the index of the curve in the sample. Each index of a curve is repeated as many times as it has observation points.
    - tcol : The name of the column that contains the observation points associated to each curve index.
    - ycol : The name of the column that contains the observed value of a curve at each point of observation and for each index of the curve.
  - If list of data.table In this case, each element of the given list corresponds to the observation scheme of a curve, which is given as data.table or data.frame. The data.table contains at least 2 columns.
    - tcol : The name of the column that contains the observation points associated to the curve.
    - ycol : The name of the column that contains the observed value of the curve.
  - If list of list In the latter case, the data is a list list where each element is the observation scheme of a curve given as a list of 2 vectors.
    - tcol : The name of the vector that contains the observation points associated the curve.
    - ycol : The name of the vector that contains the observed value of the curve.

idcol
: character. If data is given as data.table or data.frame, it is the name of the column that contains the index of the curve in the sample. Each index of a curve is repeated as many times as it has observation points. Opposite, if f data is given as list of data.table (or data.frame) of list of list, idcol = NULL.

tcol
: character. The name of the column (or vector) that contains the observation points associated to the curves.

ycol
: character. The name of the column that contains the observed value of the curves.

s
: vector (numeric). First argument of the autocovariance function. It corresponds to the observation points s in the pair (s, t). It has to be of the same length as the t

t
: vector (numeric). Second argument of the autocovariance function. It corresponds to the observation points t in the pair (s, t). It has to be of the same length as the s.

lag
: integer (positive integer). Lag of the autocovariance.

p
: numeric (integer). It is used as exponent.

q
: numeric (integer). It is used as exponent.

h	numeric (positive scalar). The bandwidth of the estimator.
dt_mean_rp	data.table. It contains the estimates of the mean function at each observation point for each curve. The name of the curve identification column must be id_curve, the observation points column tobs and the mean estimates column muhat_RP. Default dt_mean_rp = NULL and so it will be estimated.
optbw_mean	numeric (positive scalar). Optimal bandwidth for the mean function estimator. It is NULL if dt_mean_rp is not NULL.
smooth_ker	function. The kernel function of the Nadaraya-Watson estimator. Default smooth_ker = epanechnikov.

Value

A numeric scalar.

.random_design *Generate a random design of the place where the observation is made.*

Description

Generate a random design of the place where the observation is made.

Usage

```
.random_design(N, lambda, Mdistribution = rpois, tdistribution = runif, ...)
```

Arguments

N	integer. Number of curves.
lambda	integer. Mean of the number of observations per curve.
Mdistribution	function. Distribution of the number of observation points per curve. The first argument of the function must correspond to N and the second to lambda. Default Mdistribution = rpois.
tdistribution	function. Distribution of the observation point in the domain. Currently only runif is accepted.
...	Additional argument of tdistribution.

Value

A data.table containing 3 column :

- id_curve : Index of the curve. It goes from 1 to N.
- Mn : Number of sampled observation location.
- Tn : Sampled observation location.

.Spq_fun *$S\_pq\hat{}(\ell)$, $(\ell \leq 0)$ function. See Rubìn and Panaretos (2020) Equation (B.7)*

Description

$S_{pq}^{(\ell)}$, $(\ell \leq 0)$ function. See Rubìn and Panaretos (2020) Equation (B.7)

Usage

```
.Spq_fun(
  data,
  idcol = "id_curve",
  tcol = "tobs",
  ycol = "X",
  s = 1/4,
  t = 1/2,
  lag = 1,
  p = 1,
  q = 1,
  h,
  smooth_ker = epanechnikov
)
```

Arguments

data `data.table` (or `data.frame`) or `list` of `data.table` (or `data.frame`) or `list` of `list`.

- If `data.table` It must contain the raw binding of the curve observations with at least 3 columns.
  - `idcol` : The name of the column that contains the index of the curve in the sample. Each index of a curve is repeated as many times as it has observation points.
  - `tcol` : The name of the column that contains the observation points associated to each curve index.
  - `ycol` : The name of the column that contains the observed value of a curve at each point of observation and for each index of the curve.
- If `list` of `data.table` In this case, each element of the given `list` corresponds to the observation scheme of a curve, which is given as `data.table` or `data.frame`. The data.table contains at least 2 columns.
  - `tcol` : The name of the column that contains the observation points associated to the curve.
  - `ycol` : The name of the column that contains the observed value of the curve.
- If `list` of `list` In the latter case, the `data` is a list `list` where each element is the observation scheme of a curve given as a `list` of 2 vectors.
  - `tcol` : The name of the vector that contains the observation points associated the curve.

- `ycol` : The name of the vector that contains the observed value of the curve.

idcol	`character`. If `data` is given as `data.table` or `data.frame`, it is the name of the column that contains the index of the curve in the sample. Each index of a curve is repeated as many times as it has observation points. Opposite, if f `data` is given as `list` of `data.table` (or `data.frame`) of `list` of `list`, `idcol = NULL`.
tcol	`character`. The name of the column (or vector) that contains the observation points associated to the curves.
ycol	`character`. The name of the column that contains the observed value of the curves.
s	`vector (numeric)`. First argument of the autocovariance function. It corresponds to the observation points `s` in the pair (`s, t`). It has to be of the same length as the `t`
t	`vector (numeric)`. Second argument of the autocovariance function. It corresponds to the observation points `t` in the pair (`s, t`). It has to be of the same length as the `s`.
lag	`integer (positive integer)`. Lag of the autocovariance.
p	`numeric (integer)`. It is used as exponent.
q	`numeric (integer)`. It is used as exponent.
h	`numeric (positive scalar)`. The bandwidth of the estimator.
smooth_ker	`function`. The kernel function of the Nadaraya-Watson estimator. Default `smooth_ker = epanechnikov`.

Value

A `numeric` scalar.

`biweight` *Biweight kernel function*

Description

Biweight kernel function

Usage

```
biweight(u)
```

Arguments

u	`numeric`. Scalar or vector of numeric values at which to evaluate the function.

Value

A scalar or vector of `numeric`.

See Also

[triweight()], [tricube()], [epanechnikov()], [triangular()], and [uniform()].

epanechnikov *Epanechnikov kernel function*

Description

Epanechnikov kernel function

Usage

```
epanechnikov(u)
```

Arguments

u	`numeric`. Scalar or vector of numeric values at which to evaluate the function.

Value

A scalar or vector of `numeric`.

See Also

[biweight()], [triweight()], [tricube()], [triangular()], and [uniform()].

estimate_autocov *Estimate lag-ℓ ($\ell > 0$) autocovariance function*

Description

Estimate lag-ℓ ($\ell > 0$) autocovariance function

Usage

```
estimate_autocov(
  data,
  idcol = "id_curve",
  tcol = "tobs",
  ycol = "X",
  s = c(1/5, 2/5, 4/5),
  t = c(1/4, 1/2, 3/4),
  lag = 1,
  optbw = NULL,
  bw_grid = seq(0.005, 0.15, len = 45),
  Hs = NULL,
  Ls = NULL,
  Ht = NULL,
  Lt = NULL,
  Delta = NULL,
  h = NULL,
  center = TRUE,
  mean_estimates_s = NULL,
```

```
  mean_estimates_t = NULL,
  smooth_ker = epanechnikov
)
```

Arguments

data `data.table` (or `data.frame`) or `list` of `data.table` (or `data.frame`) or `list` of `list`.

- If `data.table` It must contain the raw binding of the curve observations with at least 3 columns.
  - `idcol` : The name of the column that contains the index of the curve in the sample. Each index of a curve is repeated as many times as it has observation points.
  - `tcol` : The name of the column that contains the observation points associated to each curve index.
  - `ycol` : The name of the column that contains the observed value of a curve at each point of observation and for each index of the curve.
- If `list` of `data.table` In this case, each element of the given `list` corresponds to the observation scheme of a curve, which is given as `data.table` or `data.frame`. The data.table contains at least 2 columns.
  - `tcol` : The name of the column that contains the observation points associated to the curve.
  - `ycol` : The name of the column that contains the observed value of the curve.
- If `list` of `list` In the latter case, the `data` is a list `list` where each element is the observation scheme of a curve given as a `list` of 2 vectors.
  - `tcol` : The name of the vector that contains the observation points associated the curve.
  - `ycol` : The name of the vector that contains the observed value of the curve.

idcol `character`. If `data` is given as `data.table` or `data.frame`, it is the name of the column that contains the index of the curve in the sample. Each index of a curve is repeated as many times as it has observation points. Opposite, if f `data` is given as `list` of `data.table` (or `data.frame`) of `list` of `list`, `idcol = NULL`.

tcol `character`. The name of the column (or vector) that contains the observation points associated to the curves.

ycol `character`. The name of the column that contains the observed value of the curves.

s `vector (numeric)`. First argument of the autocovariance function. It corresponds to the observation points `s` in the pair (`s`, `t`). It has to be of the same length as the `t`

t `vector (numeric)`. Second argument of the autocovariance function. It corresponds to the observation points `t` in the pair (`s`, `t`). It has to be of the same length as the `s`.

lag `integer (positive integer)`. Lag of the autocovariance.

optbw `vector (numeric)`. The optimal bandwidth parameter for lag-ℓ ($\ell > 0$) autocovariance function estimation for each pair (`s`, `t`). Default `optbw = NULL` and thus it will be estimated using estimate_autocov_risk function.

bw_grid
: vector (numeric). The bandwidth grid in which the best smoothing parameter is selected for each pair (s, t). It can be NULL and that way it will be defined as an exponential grid of $N \times \lambda$.

Hs
: vector (numeric). The estimates of the local exponent for each s. Default Hs = NULL and thus it will be estimated.

Ls
: vector (numeric). The estimates of the Hölder constant for each s. It corresponds to L_s^2. Default Ls = NULL and thus it will be estimated.

Ht
: vector (numeric). The estimates of the local exponent for each t. Default Ht = NULL and thus it will be estimated.

Lt
: vector (numeric). The estimates of the Hölder constant for each t. It corresponds to L_t^2. Default Lt = NULL and thus it will be estimated.

Delta
: numeric (positive). The length of the neighbourhood of s and t around which the local regularity is to be estimated. Default Delta = NULL and thus it will be estimated from the data.

h
: numeric (positive vector or scalar). The bandwidth of the Nadaraya-Watson estimator for the local regularity estimation. Default h = NULL and thus it will be estimated by Cross-Validation on a subset of curves. If h is a scalar, then all curves will be smoothed with the same bandwidth. Otherwise, if h is a vector, its length must be equal to the number of curves in data and each element of the vector must correspond to a curve given in the same order as in data.

center
: logical (TRUE or FALSE). Default center = TRUE and so the curves are centred when the autocovariance is estimated: $\mathbb{E}(X_0(s) - \mu(s))(X_\ell(t) - \mu(t))$. Otherwise, the two parts $\mathbb{E}X_0(s)X_\ell(t)$ and $\mu(s)\mu(t)$ will be estimated separately. The first part with a bandwidth obtained with estimate_autocov_risk and the second part with a bandwidth obtained with estimate_mean_risk.

mean_estimates_s
: vector (numeric). Mean function estimates at each s. The default is mean_estimates_s = NULL and so it is estimated if center = FALSE.

mean_estimates_t
: vector (numeric). Mean function estimates at each t if center = FALSE. The default is mean_estimates_t = NULL and so it is estimated.

smooth_ker
: function. The kernel function of the Nadaraya-Watson estimator. Default smooth_ker = epanechnikov.

Value

A data.table containing the following columns.

- s : The first argument of the autocovariance function.
- t : The second argument of the autocovariance function.
- locreg_bw : The bandwidth used to estimate the local regularity parameters.
- Hs : The estimates of the local exponent for each s.
- Ls : The estimates of the Hölder constant for each s. It corresponds to L_s^2.
- Ht : The estimates of the local exponent for each t.
- Lt : The estimates of the Hölder constant for each t. It corresponds to L_t^2.
- lag : The lag of the autocovariance. It corresponds to $\ell > 0$.
- optbw_gamma : The optimal bandwidth for $\gamma_\ell(s,t)$, $\ell > 0$.
- optbw_muhat_s : The optimal bandwidth for $\mu(s)$.

- optbw_muhat_t : The optimal bandwidth for $\mu(t)$.
- muhat_s : The estimates of the mean function $\mu(s)$ for each s.
- muhat_t : The estimates of the mean function $\mu(t)$ for each t.
- gammahat : The estimates of the $\gamma_\ell(s,t)$ function for each (s, t).
- autocovhat : The estimates of the lag-ℓ autocovariance function for each (s, t).

See Also

[estimate_mean()], [estimate_locreg()], [estimate_sigma()], [estimate_nw()], [estimate_autocov_risk()], [estimate_autocov_risk()].

Examples

```
## Not run:
# Coming ...

# Coming ...

# Coming ...

## End(Not run)
```

---

estimate_autocov_bw_rp

Bandwidth estimation using cross-validation for the Rubìn and Panaretos (2020) autocovariance function estimator.

Description

Bandwidth estimation using cross-validation for the Rubìn and Panaretos (2020) autocovariance function estimator.

Usage

```
estimate_autocov_bw_rp(
  data,
  idcol = "id_curve",
  tcol = "tobs",
  ycol = "X",
  Kfold = 10,
  bw_grid = seq(0.001, 0.15, len = 45),
  optbw_mean = NULL,
  dt_mean_rp = NULL,
  smooth_ker = epanechnikov
)
```

Arguments

`data` — data.table (or data.frame) or list of data.table (or data.frame) or list of list.

- If data.table It must contain the raw binding of the curve observations with at least 3 columns.
  - idcol : The name of the column that contains the index of the curve in the sample. Each index of a curve is repeated as many times as it has observation points.
  - tcol : The name of the column that contains the observation points associated to each curve index.
  - ycol : The name of the column that contains the observed value of a curve at each point of observation and for each index of the curve.
- If list of data.table In this case, each element of the given list corresponds to the observation scheme of a curve, which is given as data.table or data.frame. The data.table contains at least 2 columns.
  - tcol : The name of the column that contains the observation points associated to the curve.
  - ycol : The name of the column that contains the observed value of the curve.
- If list of list In the latter case, the data is a list list where each element is the observation scheme of a curve given as a list of 2 vectors.
  - tcol : The name of the vector that contains the observation points associated the curve.
  - ycol : The name of the vector that contains the observed value of the curve.

`idcol` — character. If data is given as data.table or data.frame, it is the name of the column that contains the index of the curve in the sample. Each index of a curve is repeated as many times as it has observation points. Opposite, if f data is given as list of data.table (or data.frame) of list of list, idcol = NULL.

`tcol` — character. The name of the column (or vector) that contains the observation points associated to the curves.

`ycol` — character. The name of the column that contains the observed value of the curves.

`Kfold` — integer (positive). Number of fold for the cross-validation.

`bw_grid` — vector (numeric). The bandwidth grid.

`optbw_mean` — numeric (positive scalar). Optimal bandwidth for the mean function estimator. It is NULL if dt_mean_rp is not NULL.

`dt_mean_rp` — data.table. It contains the estimates of the mean function at each observation point for each curve. The name of the curve identification column must be id_curve, the observation points column tobs and the mean estimates column muhat_RP. Default dt_mean_rp = NULL and so it will be estimated.

`smooth_ker` — function. The kernel function of the Nadaraya-Watson estimator. Default smooth_ker = epanechnikov.

Value

A data.table containing the following columns.

- h : The candidate bandwidth.
- cv_error : The estimates of the Cross-Validation error for each h.

See Also

[estimate_mean_rp()], [estimate_mean_bw_rp()], [estimate_autocov_rp()]

Examples

```
## Not run:
# Coming ...

## End(Not run)
```

---

estimate_autocov_risk *Estimate the risk of the lag-ℓ ($\ell > 0$) autocovariance function*

Description

Estimate the risk of the lag-ℓ ($\ell > 0$) autocovariance function

Usage

```
estimate_autocov_risk(
  data,
  idcol = "id_curve",
  tcol = "tobs",
  ycol = "X",
  s = c(1/5, 2/5, 4/5),
  t = c(1/4, 1/2, 3/4),
  lag = 1,
  bw_grid = seq(0.005, 0.15, len = 45),
  smooth_ker = epanechnikov,
  Hs = NULL,
  Ls = NULL,
  Ht = NULL,
  Lt = NULL,
  Delta = NULL,
  h = NULL
)
```

Arguments

data — `data.table` (or `data.frame`) or `list` of `data.table` (or `data.frame`) or `list` of `list`.

- If `data.table` It must contain the raw binding of the curve observations with at least 3 columns.
  - `idcol` : The name of the column that contains the index of the curve in the sample. Each index of a curve is repeated as many times as it has observation points.

- tcol : The name of the column that contains the observation points associated to each curve index.
- ycol : The name of the column that contains the observed value of a curve at each point of observation and for each index of the curve.

• If list of data.table In this case, each element of the given list corresponds to the observation scheme of a curve, which is given as data.table or data.frame. The data.table contains at least 2 columns.
  - tcol : The name of the column that contains the observation points associated to the curve.
  - ycol : The name of the column that contains the observed value of the curve.

• If list of list In the latter case, the data is a list list where each element is the observation scheme of a curve given as a list of 2 vectors.
  - tcol : The name of the vector that contains the observation points associated the curve.
  - ycol : The name of the vector that contains the observed value of the curve.

idcol — character. If data is given as data.table or data.frame, it is the name of the column that contains the index of the curve in the sample. Each index of a curve is repeated as many times as it has observation points. Opposite, if f data is given as list of data.table (or data.frame) of list of list, idcol = NULL.

tcol — character. The name of the column (or vector) that contains the observation points associated to the curves.

ycol — character. The name of the column that contains the observed value of the curves.

s — vector (numeric). First argument of the autocovariance function. It corresponds to the observation points s in the pair (s, t). It has to be of the same length as the t

t — vector (numeric). Second argument of the autocovariance function. It corresponds to the observation points t in the pair (s, t). It has to be of the same length as the s.

lag — integer (positive integer). Lag of the autocovariance.

bw_grid — vector (numeric). The bandwidth grid in which the best smoothing parameter is selected for each pair (s, t). It can be NULL and that way it will be defined as an exponential grid of $N \times \lambda$.

smooth_ker — function. The kernel function of the Nadaraya-Watson estimator. Default smooth_ker = epanechnikov.

Hs — vector (numeric). The estimates of the local exponent for each s. Default Hs = NULL and thus it will be estimated.

Ls — vector (numeric). The estimates of the Hölder constant for each s. It corresponds to L_s^2. Default Ls = NULL and thus it will be estimated.

Ht — vector (numeric). The estimates of the local exponent for each t. Default Ht = NULL and thus it will be estimated.

Lt — vector (numeric). The estimates of the Hölder constant for each t. It corresponds to L_t^2. Default Lt = NULL and thus it will be estimated.

Delta — numeric (positive). The length of the neighbourhood of s and t around which the local regularity is to be estimated. Default Delta = NULL and thus it will be estimated from the data.

h — numeric (positive vector or scalar). The bandwidth of the Nadaraya-Watson estimator for the local regularity estimation. Default h = NULL and thus it will be estimated by Cross-Validation on a subset of curves. If h is a scalar, then all curves will be smoothed with the same bandwidth. Otherwise, if h is a vector, its length must be equal to the number of curves in data and each element of the vector must correspond to a curve given in the same order as in data.

Value

A data.table containing the following columns.

- s : The first argument of the autocovariance function.
- t : The second argument of the autocovariance function.
- h : The candidate bandwidth.
- Hs : The estimates of the local exponent for each s. It corresponds to H_s
- Ls : The estimates of the Hölder constant for each s. It corresponds to L_s^2.
- Ht : The estimates of the local exponent for each t. It corresponds to H_t
- Lt : The estimates of the Hölder constant for each t. It corresponds to L_t^2.
- bias_term : The bias term of the risk function.
- varriance_term : The variance term of the risk function.
- dependence_term : The dependence term of the risk function.
- autocov_risk : The estimates of the risk function of the autocovariance function.

See Also

[estimate_mean()], [estimate_locreg()], [estimate_sigma()], [estimate_nw()], [estimate_empirical_XsXt_autocov()].

Examples

```
## Not run:
# Generate a sample path of FTS
dt_far <- simulate_far(N = 50, lambda = 70,
                       tdesign = "random",
                       Mdistribution = rpois,
                       tdistribution = runif,
                       tcommon = NULL,
                       hurst_fun = hurst_logistic,
                       L = 4,
                       far_kernel = get_real_data_far_kenel,
                       far_mean = get_real_data_mean,
                       int_grid = 100L,
                       burnin = 100L,
                       remove_burnin = TRUE)

# Add noise
dt_far[, X := X + rnorm(n = .N, mean = 0, sd = 0.9 ** (0.1)), by = id_curve]

# Estimate risk function
dt_autocov_risk <- estimate_autocov_risk(
  data = dt_far, idcol = "id_curve", tcol = "tobs", ycol = "X",
  s = c(1/5, 2/5, 4/5),
  t = c(1/4, 1/2, 3/4),
```

```
  lag = 3,
  bw_grid = seq(0.005, 0.15, len = 45),
  Delta = NULL, h = NULL, smooth_ker = epanechnikov
)

# Plot mean risk function
dt_dcast <- data.table::dcast(data = dt_autocov_risk,
                              formula = h ~ s + t ,
                              value.var = "autocov_risk")

manipulateWidget::combineWidgets(
  list = list(
    dygraphs::dygraph(data = dt_dcast[, .(h, "(s, t) = (0.2, 0.25)" = `0.2_0.25`)],
                      main = "lag = 3 - (s, t) = (0.2, 0.25)",
                      xlab = "h",
                      ylab = "risk function"),
    dygraphs::dygraph(data = dt_dcast[, .(h, "(s, t) = (0.4, 0.5)" = `0.4_0.5`)],
                      main = "lag = 3 - (s, t) = (0.4, 0.5)",
                      xlab = "h",
                      ylab = "risk function"),
    dygraphs::dygraph(data = dt_dcast[, .(h, "(s, t) = (0.8, 0.75)" = `0.8_0.75`)],
                      main = "lag = 3 - (s, t) = (0.8, 0.75)",
                      xlab = "h",
                      ylab = "risk function")
  ),
  nrow = 3
)


## End(Not run)
```

`estimate_autocov_rp` *Estimate lag-ℓ ($\ell \leq 0$) autocovariance function using Rubìn and Panaretos (2020) method*

Description

Estimate lag-ℓ ($\ell \leq 0$) autocovariance function using Rubìn and Panaretos (2020) method

Usage

```
estimate_autocov_rp(
  data,
  idcol = "id_curve",
  tcol = "tobs",
  ycol = "X",
  s = c(1/5, 2/5, 4/5),
  t = c(1/4, 1/2, 3/4),
  lag = 1,
  h,
  optbw_mean = NULL,
  dt_mean_rp = NULL,
  smooth_ker = epanechnikov
)
```

Arguments

data
: data.table (or data.frame) or list of data.table (or data.frame) or list of list.

- If data.table It must contain the raw binding of the curve observations with at least 3 columns.
  - idcol : The name of the column that contains the index of the curve in the sample. Each index of a curve is repeated as many times as it has observation points.
  - tcol : The name of the column that contains the observation points associated to each curve index.
  - ycol : The name of the column that contains the observed value of a curve at each point of observation and for each index of the curve.
- If list of data.table In this case, each element of the given list corresponds to the observation scheme of a curve, which is given as data.table or data.frame. The data.table contains at least 2 columns.
  - tcol : The name of the column that contains the observation points associated to the curve.
  - ycol : The name of the column that contains the observed value of the curve.
- If list of list In the latter case, the data is a list list where each element is the observation scheme of a curve given as a list of 2 vectors.
  - tcol : The name of the vector that contains the observation points associated the curve.
  - ycol : The name of the vector that contains the observed value of the curve.

idcol
: character. If data is given as data.table or data.frame, it is the name of the column that contains the index of the curve in the sample. Each index of a curve is repeated as many times as it has observation points. Opposite, if f data is given as list of data.table (or data.frame) of list of list, idcol = NULL.

tcol
: character. The name of the column (or vector) that contains the observation points associated to the curves.

ycol
: character. The name of the column that contains the observed value of the curves.

s
: vector (numeric). First argument of the autocovariance function. It corresponds to the observation points s in the pair (s, t). It has to be of the same length as the t

t
: vector (numeric). Second argument of the autocovariance function. It corresponds to the observation points t in the pair (s, t). It has to be of the same length as the s.

lag
: integer (positive integer). Lag of the autocovariance.

h
: numeric (positive scalar). The bandwidth of the estimator.

optbw_mean
: numeric (positive scalar). Optimal bandwidth for the mean function estimator. It is NULL if dt_mean_rp is not NULL.

dt_mean_rp
: data.table. It contains the estimates of the mean function at each observation point for each curve. The name of the curve identification column must be id_curve, the observation points column tobs and the mean estimates column muhat_RP. Default dt_mean_rp = NULL and so it will be estimated.

smooth_ker — function. The kernel function of the Nadaraya-Watson estimator. Default smooth_ker = epanechnikov.

Value

A data.table containing the following columns.

- s : The first argument of the autocovariance function.
- t : The second argument of the autocovariance function.
- lag : The lag of the autocovariance. It corresponds to $\ell \leq 0$.
- optbw_mean : The optimal bandwidth for the mean function estimator.
- h : The bandwidth used to estimate the lag-ℓ, $\ell \leq 0$ autocovariance function
- autocovhat_rp : The estimates of the lag-ℓ autocovariance function for each (s, t) using Rubìn and Panaretos (2020) method.

---

estimate_empirical_autocov

Estimate empirical autocovariance function

Description

Estimate empirical autocovariance function

Usage

```
estimate_empirical_autocov(
  data,
  idcol = NULL,
  tcol = "tobs",
  ycol = "X",
  t = c(1/4, 1/2, 3/4),
  lag = c(0, 1, 2),
  h = NULL,
  smooth_ker = epanechnikov
)
```

Arguments

data — data.table (or data.frame) or list of data.table (or data.frame) or list of list.

- If data.table It must contain the raw binding of the curve observations with at least 3 columns.
  - idcol : The name of the column that contains the index of the curve in the sample. Each index of a curve is repeated as many times as it has observation points.
  - tcol : The name of the column that contains the observation points associated to each curve index.
  - ycol : The name of the column that contains the observed value of a curve at each point of observation and for each index of the curve.

- If list of data.table In this case, each element of the given list corresponds to the observation scheme of a curve, which is given as data.table or data.frame. The data.table contains at least 2 columns.
  - tcol : The name of the column that contains the observation points associated to the curve.
  - ycol : The name of the column that contains the observed value of the curve.
- If list of list In the latter case, the data is a list list where each element is the observation scheme of a curve given as a list of 2 vectors.
  - tcol : The name of the vector that contains the observation points associated the curve.
  - ycol : The name of the vector that contains the observed value of the curve.

idcol — character. If data is given as data.table or data.frame, it is the name of the column that contains the index of the curve in the sample. Each index of a curve is repeated as many times as it has observation points. Opposite, if f data is given as list of data.table (or data.frame) of list of list, idcol = NULL.

tcol — character. The name of the column (or vector) that contains the observation points associated to the curves.

ycol — character. The name of the column that contains the observed value of the curves.

t — vector (numeric). Observation points at which we want to estimate the empirical autocovariance function.

lag — vector (integer). Lag of the autocovariance.

h — numeric (positive vector or scalar). The smoothing bandwidth parameter. Default h = NULL and thus it will be estimated by Cross-Validation on a subset of curves. If h is a scalar, then all curves will be smoothed with the same bandwidth. Otherwise, if h is a vector, its length must be equal to the number of curves in data and each element of the vector must correspond to a curve given in the same order as in data.

smooth_ker — function. The kernel function of the Nadaraya-Watson estimator. Default smooth_ker = epanechnikov.

Value

A data.table with three columns: t, lag and autocov corresponding to the estimated autocovariance.

See Also

[estimate_nw()]

`estimate_empirical_XsXt_autocov`
Estimate empirical $X\_0(s)X\_\ell(t)$ autocovariance function

Description

Estimate empirical $X_0(s)X_\ell(t)$ autocovariance function

Usage

```
estimate_empirical_XsXt_autocov(
  data,
  idcol = NULL,
  tcol = "tobs",
  ycol = "X",
  s = c(1/5, 2/5, 4/5),
  t = c(1/4, 1/2, 3/4),
  cross_lag = 1,
  lag = c(0, 1, 2),
  h = NULL,
  smooth_ker = epanechnikov,
  center = FALSE
)
```

Arguments

`data` — `data.table` (or `data.frame`) or `list` of `data.table` (or `data.frame`) or `list` of `list`.

- If `data.table` It must contain the raw binding of the curve observations with at least 3 columns.
  - `idcol` : The name of the column that contains the index of the curve in the sample. Each index of a curve is repeated as many times as it has observation points.
  - `tcol` : The name of the column that contains the observation points associated to each curve index.
  - `ycol` : The name of the column that contains the observed value of a curve at each point of observation and for each index of the curve.
- If `list` of `data.table` In this case, each element of the given `list` corresponds to the observation scheme of a curve, which is given as `data.table` or `data.frame`. The data.table contains at least 2 columns.
  - `tcol` : The name of the column that contains the observation points associated to the curve.
  - `ycol` : The name of the column that contains the observed value of the curve.
- If `list` of `list` In the latter case, the `data` is a list `list` where each element is the observation scheme of a curve given as a `list` of 2 vectors.
  - `tcol` : The name of the vector that contains the observation points associated the curve.

- ycol : The name of the vector that contains the observed value of the curve.

idcol
: character. If data is given as data.table or data.frame, it is the name of the column that contains the index of the curve in the sample. Each index of a curve is repeated as many times as it has observation points. Opposite, if f data is given as list of data.table (or data.frame) of list of list, idcol = NULL.

tcol
: character. The name of the column (or vector) that contains the observation points associated to the curves.

ycol
: character. The name of the column that contains the observed value of the curves.

s
: vector (numeric). First argument in $X_0(s)X_\ell(t)$. It corresponds to the observation points s in the pair (s, t). It has to be of the same length as the t

t
: vector (numeric). Second argument in $X_0(s)X_\ell(t)$. It corresponds to the observation points t in the pair (s, t). It has to be of the same length as the s.

cross_lag
: integer (positive integer). It corresponds to the lag ℓ in $X_0(s)X_\ell(t)$.

lag
: vector (integer). Lag of the autocovariance of the random variable $X_0(s)X_\ell(t)$. If lag = NULL, only $\mathbb{E}X_0(s)X_\ell(t)$ is returned.

h
: numeric (positive vector or scalar). The smoothing bandwidth parameter. Default h = NULL and thus it will be estimated by Cross-Validation on a subset of curves. If h is a scalar, then all curves will be smoothed with the same bandwidth. Otherwise, if h is a vector, its length must be equal to the number of curves in data and each element of the vector must correspond to a curve given in the same order as in data.

smooth_ker
: function. The kernel function of the Nadaraya-Watson estimator. Default smooth_ker = epanechnikov.

center
: logical (TRUE or FALSE). Default center = FALSE and so the curves are not centred when the autocovariance is estimated: $\mathbb{E}X_0(s)X_\ell(t)$. Otherwise, the curves are centred and the estimated autocovariance is: $\mathbb{E}(X_0(s)-\mu(s))(X_\ell(t)-\mu(t))$.

Value

A data.table with three columns: s, t, cross_lag, lag, EXsXt_cross_lag and XsXt_autocov corresponding to the estimated autocovariance of the random variable $X_0(s)X_\ell(t)$.

A data.table containing the following columns.

- s : The first argument in $X_0(s)X_\ell(t)$.
- t : The second argument in $X_0(s)X_\ell(t)$.
- cross_lag : The lag ℓ in $X_0(s)X_\ell(t)$.
- lag : The lags at which the autocovariance of the random variable $X_0(s)X_\ell(t)$ is estimated. It contains NA if lag = NULL.
- EXsXt_cross_lag : Mean of the random variable $X_0(s)X_\ell(t)$.
- XsXt_autocov : The estimates of the autocovariance of the random variable $X_0(s)X_\ell(t)$ for each lag. It contains NA if lag = NULL.

See Also

[estimate_nw()]

estimate_locreg *Local Regularity Parameters Estimation*

Description

Local Regularity Parameters Estimation

Usage

```
estimate_locreg(
  data,
  idcol = "id_curve",
  tcol = "tobs",
  ycol = "X",
  t = 1/2,
  Delta = NULL,
  h = NULL,
  smooth_ker = epanechnikov,
  center = TRUE
)
```

Arguments

data — `data.table` (or `data.frame`) or `list` of `data.table` (or `data.frame`) or `list` of `list`.

- If `data.table` It must contain the raw binding of the curve observations with at least 3 columns.
  - `idcol` : The name of the column that contains the index of the curve in the sample. Each index of a curve is repeated as many times as it has observation points.
  - `tcol` : The name of the column that contains the observation points associated to each curve index.
  - `ycol` : The name of the column that contains the observed value of a curve at each point of observation and for each index of the curve.
- If `list` of `data.table` In this case, each element of the given `list` corresponds to the observation scheme of a curve, which is given as `data.table` or `data.frame`. The data.table contains at least 2 columns.
  - `tcol` : The name of the column that contains the observation points associated to the curve.
  - `ycol` : The name of the column that contains the observed value of the curve.
- If `list` of `list` In the latter case, the `data` is a list `list` where each element is the observation scheme of a curve given as a `list` of 2 vectors.
  - `tcol` : The name of the vector that contains the observation points associated the curve.
  - `ycol` : The name of the vector that contains the observed value of the curve.

idcol	character. If data is given as data.table or data.frame, it is the name of the column that contains the index of the curve in the sample. Each index of a curve is repeated as many times as it has observation points. Opposite, if f data is given as list of data.table (or data.frame) of list of list, idcol = NULL.
tcol	character. The name of the column (or vector) that contains the observation points associated to the curves.
ycol	character. The name of the column that contains the observed value of the curves.
t	vector (numeric). Observation points at which we want to estimate the local regularity parameters of the underlying process.
Delta	numeric (positive). The length of the neighbourhood of t around which the local regularity is to be estimated. Default Delta = NULL and thus it will be estimated from the data.
h	numeric (positive vector or scalar). The bandwidth of the Nadaraya-Watson estimator for the local regularity estimation. Default h = NULL and thus it will be estimated by Cross-Validation on a subset of curves. If h is a scalar, then all curves will be smoothed with the same bandwidth. Otherwise, if h is a vector, its length must be equal to the number of curves in data and each element of the vector must correspond to a curve given in the same order as in data.
smooth_ker	function. The kernel function of the Nadaraya-Watson estimator. Default smooth_ker = epanechnikov.
center	logical. If TRUE, the curves are centered.

Value

A data.table containing the following columns.

- t : The points around which the local regularity parameters are estimated.
- locreg_bw : The presmoothing bandwidth.
- Delta : The length of the neighbourhood of t around which the local regularity is to be estimated.
- Nused : The number of curves that give non-degenerate estimates around t.
- Ht : The local exponent estimates for each t. It corresponds to H_t
- Lt : The Hölder constant estimates t. It corresponds to L_t^2.

See Also

[estimate_nw()], [estimate_nw_bw()], [simulate_far()], etc.

Examples

```
## Not run:
  # Generate a sample of FAR(1)
Hfun <- function(t) {
  hurst_logistic(t = t, h_left = 0.4, h_right = 0.8, slope = 5)
}

## Hölder constant
L <- 4
```

```
dt_far <- simulate_far(N = 200L, lambda = 100L,
                       tdesign = "random",
                       Mdistribution = rpois,
                       tdistribution = runif,
                       tcommon = NULL,
                       hurst_fun = Hfun,
                       L = L,
                       far_kernel = function(s,t) 9/4 * exp( - (t + 2 * s) ** 2),
                       far_mean = function(t) 4 * sin(1.5 * pi * t),
                       int_grid = 100L,
                       burnin = 100L,
                       remove_burnin = TRUE)

# Estimate local regularity at
t0 <- seq(0.2, 0.8, len = 8)

## If data is a data.table or a data. frame
dt_locreg <- estimate_locreg(data = dt_far,
                             idcol = "id_curve",
                             tcol = "tobs",
                             ycol = "X",
                             t = t0,
                             Delta = NULL,
                             h = NULL,
                             smooth_ker = epanechnikov)
DT::datatable(dt_locreg)

## If data is a list of data.table (or data. frame)
list_dt_far <- lapply(unique(dt_far[, id_curve]), function(idx){
  dt_far[id_curve == idx, list(tobs, X)]
})

dt_locreg_2 <- estimate_locreg(data = list_dt_far,
                               idcol = NULL,
                               tcol = "tobs",
                               ycol = "X",
                               t = t0,
                               Delta = NULL,
                               h = NULL,
                               smooth_ker = epanechnikov)
DT::datatable(dt_locreg_2)

## If data is a list of list
list_list_far <- lapply(unique(dt_far[, id_curve]), function(idx){
  list("Obs_point" = dt_far[id_curve == idx, tobs],
       "Xobs" = dt_far[id_curve == idx, X])
})

dt_locreg_3 <- estimate_locreg(data = list_list_far,
                               idcol = NULL,
                               tcol = "Obs_point",
                               ycol = "Xobs",
                               t = t0,
                               Delta = NULL,
                               h = NULL,
                               smooth_ker = epanechnikov)
```

```
DT::datatable(dt_locreg_2)


## End(Not run)
```

---

estimate_mean *Estimate mean function*

Description

Mean function estimation using the adaptive estimator of \insertCitemaissoro2024adaptive;textualadaptiveFTS.

Usage

```
estimate_mean(
  data,
  idcol = "id_curve",
  tcol = "tobs",
  ycol = "X",
  t = c(1/4, 1/2, 3/4),
  optbw = NULL,
  bw_grid = seq(0.005, 0.15, len = 45),
  Ht = NULL,
  Lt = NULL,
  Delta = NULL,
  h = NULL,
  smooth_ker = epanechnikov
)
```

Arguments

data: `data.table` (or `data.frame`) or `list` of `data.table` (or `data.frame`) or `list` of `list`.

- If `data.table` It must contain the raw binding of the curve observations with at least 3 columns.
  - `idcol` : The name of the column that contains the index of the curve in the sample. Each index of a curve is repeated as many times as it has observation points.
  - `tcol` : The name of the column that contains the observation points associated to each curve index.
  - `ycol` : The name of the column that contains the observed value of a curve at each point of observation and for each index of the curve.
- If `list` of `data.table` In this case, each element of the given `list` corresponds to the observation scheme of a curve, which is given as `data.table` or `data.frame`. The data.table contains at least 2 columns.
  - `tcol` : The name of the column that contains the observation points associated to the curve.

- ycol : The name of the column that contains the observed value of the curve.
- If list of list In the latter case, the data is a list list where each element is the observation scheme of a curve given as a list of 2 vectors.
  - tcol : The name of the vector that contains the observation points associated the curve.
  - ycol : The name of the vector that contains the observed value of the curve.

idcol
: character. If data is given as data.table or data.frame, it is the name of the column that contains the index of the curve in the sample. Each index of a curve is repeated as many times as it has observation points. Opposite, if f data is given as list of data.table (or data.frame) of list of list, idcol = NULL.

tcol
: character. The name of the column (or vector) that contains the observation points associated to the curves.

ycol
: character. The name of the column that contains the observed value of the curves.

t
: vector (numeric). Observation points at which we want to estimate the mean function of the underlying process.

optbw
: vector (numeric). The optimal bandwidth parameter for mean function estimation for each t. Default optbw = NULL and thus it will be estimated using estimate_mean_risk function.

bw_grid
: vector (numeric). The bandwidth grid in which the best smoothing parameter is selected for each t. It can be NULL and that way it will be defined as an exponential grid of $N \times \lambda$.

Ht
: vector (numeric). The estimates of the local exponent for each t. Default Ht = NULL and thus it will be estimated.

Lt
: vector (numeric). The estimates of the Hölder constant for each t. It corresponds to L_t^2. Default Lt = NULL and thus it will be estimated.

Delta
: numeric (positive). The length of the neighbourhood of t around which the local regularity is to be estimated. Default Delta = NULL and thus it will be estimated from the data.

h
: numeric (positive vector or scalar). The bandwidth of the Nadaraya-Watson estimator for the local regularity estimation. Default h = NULL and thus it will be estimated by Cross-Validation on a subset of curves. If h is a scalar, then all curves will be smoothed with the same bandwidth. Otherwise, if h is a vector, its length must be equal to the number of curves in data and each element of the vector must correspond to a curve given in the same order as in data.

smooth_ker
: function. The kernel function of the Nadaraya-Watson estimator. Default smooth_ker = epanechnikov.

Value

A data.table containing the following columns.

- t : The points at which the risk function is estimated.
- locreg_bw : The bandwidth used to estimate the local regularity parameters.
- Ht : The estimates of the local exponent for each t. It corresponds to H_t

- Lt : The estimates of the Hölder constant for each t. It corresponds to L_t^2.
- optbw : The optimal bandwidth. That is the bandwidth which minimises the risk function.
- PN : Number of selected curves for each t.
- muhat : The estimates of the mean function.

References

insertAllCited

See Also

[estimate_mean_risk()], [estimate_locreg()], [estimate_sigma()], [estimate_nw()], [estimate_empirical_autocov()].

Examples

```
## Not run:
# Generate a FAR A process
dt_far <- simulate_far(N = 50, lambda = 70,
                       tdesign = "random",
                       Mdistribution = rpois,
                       tdistribution = runif,
                       tcommon = NULL,
                       hurst_fun = hurst_logistic,
                       L = 4,
                       far_kernel = get_real_data_far_kenel,
                       far_mean = get_real_data_mean,
                       int_grid = 100L,
                       burnin = 100L,
                       remove_burnin = TRUE)

# Add noise
dt_far[, X := X + rnorm(n = .N, mean = 0, sd = 0.9 ** (0.1)), by = id_curve]

# Estimate mean function
dt_mean <- estimate_mean(
  data = dt_far, idcol = "id_curve", tcol = "tobs", ycol = "X",
  t = c(1/4, 1/2, 3/4), bw_grid = seq(0.005, 0.15, len = 45),
  Delta = NULL, h = NULL, smooth_ker = epanechnikov)

# Table of the estimates of the mean function
DT::datatable(data = dt_mean[, lapply(.SD, function(X) round(X, 3))])

## End(Not run)
```

---

estimate_mean_bw_rp   *Bandwidth estimation using cross-validation for the \insertCiterubin2020;textualadaptiveFTS mean function estimator.*

Description

Bandwidth estimation using cross-validation for the \insertCiterubin2020;textualadaptiveFTS mean function estimator.

Usage

```
estimate_mean_bw_rp(
  data,
  idcol = "id_curve",
  tcol = "tobs",
  ycol = "X",
  Kfold = 10,
  bw_grid = seq(0.001, 0.15, len = 45),
  smooth_ker = epanechnikov
)
```

Arguments

data — `data.table` (or `data.frame`) or `list` of `data.table` (or `data.frame`) or `list` of `list`.

- If `data.table` It must contain the raw binding of the curve observations with at least 3 columns.
  - `idcol` : The name of the column that contains the index of the curve in the sample. Each index of a curve is repeated as many times as it has observation points.
  - `tcol` : The name of the column that contains the observation points associated to each curve index.
  - `ycol` : The name of the column that contains the observed value of a curve at each point of observation and for each index of the curve.
- If `list` of `data.table` In this case, each element of the given `list` corresponds to the observation scheme of a curve, which is given as `data.table` or `data.frame`. The data.table contains at least 2 columns.
  - `tcol` : The name of the column that contains the observation points associated to the curve.
  - `ycol` : The name of the column that contains the observed value of the curve.
- If `list` of `list` In the latter case, the `data` is a list `list` where each element is the observation scheme of a curve given as a `list` of 2 vectors.
  - `tcol` : The name of the vector that contains the observation points associated the curve.
  - `ycol` : The name of the vector that contains the observed value of the curve.

idcol — `character`. If `data` is given as `data.table` or `data.frame`, it is the name of the column that contains the index of the curve in the sample. Each index of a curve is repeated as many times as it has observation points. Opposite, if f `data` is given as `list` of `data.table` (or `data.frame`) of `list` of `list`, `idcol = NULL`.

tcol — `character`. The name of the column (or vector) that contains the observation points associated to the curves.

ycol — `character`. The name of the column that contains the observed value of the curves.

Kfold — `integer (positive)`. Number of fold for the cross-validation.

bw_grid — `vector (numeric)`. The bandwidth grid.

smooth_ker — `function`. The kernel function of the Nadaraya-Watson estimator. Default `smooth_ker = epanechnikov`.

Value

A `data.table` containing the following columns.

- h : The candidate bandwidth.
- cv_error : The estimates of the Cross-Validation error for each `h`.

References

insertAllcited

See Also

[estimate_mean_rp()]

Examples

```
## Not run:
# Generate a FAR A process
dt_far <- simulate_far(N = 50, lambda = 70,
                       tdesign = "random",
                       Mdistribution = rpois,
                       tdistribution = runif,
                       tcommon = NULL,
                       hurst_fun = hurst_logistic,
                       L = 4,
                       far_kernel = get_real_data_far_kenel,
                       far_mean = get_real_data_mean,
                       int_grid = 100L,
                       burnin = 100L,
                       remove_burnin = TRUE)

# Add noise
dt_far[, X := X + rnorm(n = .N, mean = 0, sd = 0.9 ** (0.1)), by = id_curve]

## Estimate the bandwidth by Cross-Validation
dt_bw_mean_rp <- estimate_mean_bw_rp(
  data = dt_far, idcol = "id_curve", tcol = "tobs", ycol = "X",
  Kfold = 10, bw_grid = seq(0.001, 0.15, len = 45),
  smooth_ker = epanechnikov)

## Plot the Cross-Validation error
dygraphs::dygraph(dt_bw_mean_rp)

## Select the best bandwidth
optbw <- dt_bw_mean_rp[, h[which.min(cv_error)]]

## Estimate the mean function
dt_mean_rp <- estimate_mean_rp(
  data = dt_far, idcol = "id_curve", tcol = "tobs", ycol = "X",
  t = c(1/4, 1/2, 3/4), h = optbw, smooth_ker = epanechnikov)

DT::datatable(data = dt_mean_rp[, lapply(.SD, function(X) round(X, 5))])


## End(Not run)
```

estimate_mean_risk — *Estimate the risk of the mean function*

Description

The risk of the mean function is the function $R_\mu(t; h)$ in Section 4.1 of \insertCitemaissoro2024adaptive;textualadaptiveF

Usage

```
estimate_mean_risk(
  data,
  idcol = "id_curve",
  tcol = "tobs",
  ycol = "X",
  t = c(1/4, 1/2, 3/4),
  bw_grid = seq(0.005, 0.15, len = 45),
  Ht = NULL,
  Lt = NULL,
  Delta = NULL,
  h = NULL,
  smooth_ker = epanechnikov
)
```

Arguments

`data` — `data.table` (or `data.frame`) or `list` of `data.table` (or `data.frame`) or `list` of `list`.

- If `data.table` It must contain the raw binding of the curve observations with at least 3 columns.
  - `idcol` : The name of the column that contains the index of the curve in the sample. Each index of a curve is repeated as many times as it has observation points.
  - `tcol` : The name of the column that contains the observation points associated to each curve index.
  - `ycol` : The name of the column that contains the observed value of a curve at each point of observation and for each index of the curve.
- If `list` of `data.table` In this case, each element of the given `list` corresponds to the observation scheme of a curve, which is given as `data.table` or `data.frame`. The data.table contains at least 2 columns.
  - `tcol` : The name of the column that contains the observation points associated to the curve.
  - `ycol` : The name of the column that contains the observed value of the curve.
- If `list` of `list` In the latter case, the `data` is a list `list` where each element is the observation scheme of a curve given as a `list` of 2 vectors.
  - `tcol` : The name of the vector that contains the observation points associated the curve.
  - `ycol` : The name of the vector that contains the observed value of the curve.

idcol
: character. If data is given as data.table or data.frame, it is the name of the column that contains the index of the curve in the sample. Each index of a curve is repeated as many times as it has observation points. Opposite, if f data is given as list of data.table (or data.frame) of list of list, idcol = NULL.

tcol
: character. The name of the column (or vector) that contains the observation points associated to the curves.

ycol
: character. The name of the column that contains the observed value of the curves.

t
: vector (numeric). Observation points at which we want to estimate the mean function of the underlying process.

bw_grid
: vector (numeric). The bandwidth grid in which the best smoothing parameter is selected for each t. It can be NULL and that way it will be defined as an exponential grid of $N \times \lambda$.

Ht
: vector (numeric). The estimates of the local exponent for each t. Default Ht = NULL and thus it will be estimated.

Lt
: vector (numeric). The estimates of the Hölder constant for each t. It corresponds to L_t^2. Default Lt = NULL and thus it will be estimated.

Delta
: numeric (positive). The length of the neighbourhood of t around which the local regularity is to be estimated. Default Delta = NULL and thus it will be estimated from the data.

h
: numeric (positive vector or scalar). The bandwidth of the Nadaraya-Watson estimator for the local regularity estimation. Default h = NULL and thus it will be estimated by Cross-Validation on a subset of curves. If h is a scalar, then all curves will be smoothed with the same bandwidth. Otherwise, if h is a vector, its length must be equal to the number of curves in data and each element of the vector must correspond to a curve given in the same order as in data.

smooth_ker
: function. The kernel function of the Nadaraya-Watson estimator. Default smooth_ker = epanechnikov.

Value

A data.table containing the following columns.

- t : The points at which the risk function is estimated.
- h : The candidate bandwidth.
- Ht : The estimates of the local exponent for each t. It corresponds to H_t
- Lt : The estimates of the Hölder constant for each t. It corresponds to L_t^2.
- locreg_bw : The bandwidth used to estimate the local regularity parameters.
- bias_term : The bias term of the risk function.
- varriance_term : The variance term of the risk function.
- dependence_term : The dependence term of the risk function.
- mean_risk : The estimates of the risk function of the mean.

References

\insertAllCited

See Also

[estimate_mean()], [estimate_locreg()], [estimate_sigma()], [estimate_nw()], [estimate_empirical_autocov()].

Examples

```
## Not run:
# Generate a sample path of FTS
dt_far <- simulate_far(N = 50, lambda = 70,
                       tdesign = "random",
                       Mdistribution = rpois,
                       tdistribution = runif,
                       tcommon = NULL,
                       hurst_fun = hurst_logistic,
                       L = 4,
                       far_kernel = get_real_data_far_kenel,
                       far_mean = get_real_data_mean,
                       int_grid = 100L,
                       burnin = 100L,
                       remove_burnin = TRUE)

# Add noise
dt_far[, X := X + rnorm(n = .N, mean = 0, sd = 0.9 ** (0.1)), by = id_curve]

# Estimate risk function
dt_mean_risk <- estimate_mean_risk(
  data = dt_far, idcol = "id_curve", tcol = "tobs", ycol = "X",
  t = c(1/4, 1/2, 3/4), bw_grid = seq(0.005, 0.15, len = 45),
  Delta = NULL, h = NULL, smooth_ker = epanechnikov)

# Plot mean risk
dt_dcast <- data.table::dcast(data = dt_mean_risk, formula = h ~ t, value.var = "mean_risk")
manipulateWidget::combineWidgets(
  list = list(
    dygraphs::dygraph(
      data = dt_dcast[, list(h, "t = 0.25" = `0.25`)],
      main = "t = 0.25", xlab = "h", ylab = "risk function"),
    dygraphs::dygraph(
      data = dt_dcast[, list(h, "t = 0.5" = `0.5`)],
      main = "t = 0.5", xlab = "h", ylab = "risk function"),
    dygraphs::dygraph(
      data = dt_dcast[, list(h, "t = 0.75" = `0.75`)],
      main = "t = 0.75", xlab = "h", ylab = "risk function")
  ),
  nrow = 3
)


## End(Not run)
```

`estimate_mean_rp` *Estimate mean function using \insertCiterubin2020;textualadaptiveFTS method.*

Description

Estimate mean function using \insertCiterubin2020;textualadaptiveFTS method.

Usage

```
estimate_mean_rp(
  data,
  idcol = "id_curve",
  tcol = "tobs",
  ycol = "X",
  t = c(1/4, 1/2, 3/4),
  h,
  smooth_ker = epanechnikov
)
```

Arguments

`data` — `data.table` (or `data.frame`) or `list` of `data.table` (or `data.frame`) or `list` of `list`.

- If `data.table` It must contain the raw binding of the curve observations with at least 3 columns.
  - `idcol` : The name of the column that contains the index of the curve in the sample. Each index of a curve is repeated as many times as it has observation points.
  - `tcol` : The name of the column that contains the observation points associated to each curve index.
  - `ycol` : The name of the column that contains the observed value of a curve at each point of observation and for each index of the curve.
- If `list` of `data.table` In this case, each element of the given `list` corresponds to the observation scheme of a curve, which is given as `data.table` or `data.frame`. The data.table contains at least 2 columns.
  - `tcol` : The name of the column that contains the observation points associated to the curve.
  - `ycol` : The name of the column that contains the observed value of the curve.
- If `list` of `list` In the latter case, the `data` is a list `list` where each element is the observation scheme of a curve given as a `list` of 2 vectors.
  - `tcol` : The name of the vector that contains the observation points associated the curve.
  - `ycol` : The name of the vector that contains the observed value of the curve.

`idcol` — `character`. If `data` is given as `data.table` or `data.frame`, it is the name of the column that contains the index of the curve in the sample. Each index of a curve is repeated as many times as it has observation points. Opposite, if f `data` is given as `list` of `data.table` (or `data.frame`) of `list` of `list`, `idcol = NULL`.

`tcol` — `character`. The name of the column (or vector) that contains the observation points associated to the curves.

`ycol` — `character`. The name of the column that contains the observed value of the curves.

t	vector (numeric). Observation points at which we want to estimate the mean function of the underlying process.
h	numeric (positive scalar). The bandwidth of the estimator.
smooth_ker	function. The kernel function of the Nadaraya-Watson estimator. Default smooth_ker = epanechnikov.

Value

A data.table containing the following columns.

- t : The Observation points at which the mean function is estimated.
- h : The bandwidth parameter.
- muhat_RP : The estimates of the mean function using Rubìn and Panaretos (2020) method.

References

insertAllcited

See Also

[estimate_mean_bw_rp()]

Examples

```
## Not run:
# Generate a FAR A process
dt_far <- simulate_far(N = 50, lambda = 70,
                       tdesign = "random",
                       Mdistribution = rpois,
                       tdistribution = runif,
                       tcommon = NULL,
                       hurst_fun = hurst_logistic,
                       L = 4,
                       far_kernel = get_real_data_far_kenel,
                       far_mean = get_real_data_mean,
                       int_grid = 100L,
                       burnin = 100L,
                       remove_burnin = TRUE)

# Add noise
dt_far[, X := X + rnorm(n = .N, mean = 0, sd = 0.9 ** (0.1)), by = id_curve]

# Estimate mean function using Rubìn and Panaretos (2020) method
dt_mean_rp <- estimate_mean_rp(
  data = dt_far, idcol = "id_curve", tcol = "tobs", ycol = "X",
  t = c(1/4, 1/2, 3/4), h = 5/70, smooth_ker = epanechnikov)

DT::datatable(data = dt_mean_rp[, lapply(.SD, function(X) round(X, 5))])


## End(Not run)
```

estimate_nw *Nadaraya-Watson estimator*

Description

Nadaraya-Watson estimator

Usage

```
estimate_nw(y, t, tnew, h = NULL, smooth_ker = epanechnikov)
```

Arguments

y	vector (numeric). A numeric vector containing the observed values of the independent variable corresponding to the observation points t.
t	vector (numeric). A numeric vector containing the observed values of the dependent variable.
tnew	vector (numeric). New t values at which we want to estimate the regression function.
h	numeric (positive). The bandwidth parameter such that h > (2 * length(x)). Default h = NULL and such it will be computed automatically.
smooth_ker	function. The kernel function of the estimator.

Value

A data.table containing

- h : The bandwidth used to estimate the regression function.
- inKernelSupp : For each t_i in tnew, it is the number of t between t_i - h and t_i + h .
- tnew : The vector new.
- yhat : The regression function's vector of estimates at tnew.

See Also

[estimate_nw_bw()], [epanechnikov()], [biweight()], [triweight()], [tricube()], [uniform()], etc.

Examples

```
## Not run:
# The model
## Let
m <- function(t) 4 * sin(1.5 * pi * t)

## Observation points
t <- runif(n = 200, min = 0, max = 1)
t <- sort(t)

## Measure error
e <- rnorm(n = 200, mean = 0, sd = 0.2)

## Regression model
```

```
y <- m(t) + e

plot(x = t, y = y, main = "Observed points and true regression function")
lines(x = t, y = m(t), type = "l", col = "red")

## Estimate the best bandwidth
bw_grid <- seq(1 / (2 * length(t)), length(t) ** (- 1/3), len = 100)
hbest <- estimate_nw_bw(y = y, t = t,
                        bw_grid = bw_grid,
                        smooth_ker = epanechnikov)

## Estimate the regression function
dt_nw <- estimate_nw(y = y, t = t,
                     tnew = seq(0.01, 0.99, len = 100),
                     h = hbest, smooth_ker = epanechnikov)

plot(x = dt_nw[, tnew], y = dt_nw[, yhat], type = "l", col = "blue",
     main = "Estimated and true regression function.")
lines(x = dt_nw[, tnew], y = m(dt_nw[, yhat]), type = "l", col = "red")
legend(x = 0.64, y = 4.1, fill = c("blue", "red"),legend = c("Estimated m", "True m"))


## End(Not run)
```

`estimate_nw_bw` *Nadaraya-Watson Bandwidth Selection using cross validation.*

Description

Nadaraya-Watson Bandwidth Selection using cross validation.

Usage

```
estimate_nw_bw(y, t, bw_grid = NULL, smooth_ker = epanechnikov)
```

Arguments

`y`	`vector (numeric)`. A numeric vector containing the observed values of the independent variable corresponding to the observation points `t`.
`t`	`vector (numeric)`. A numeric vector containing the observed values of the dependent variable.
`bw_grid`	`vector (numeric)`. A grid of bandwidth to test. Default `bw_grid = NULL`, so it will be set as an exponential grid of `length(t)`.
`smooth_ker`	`function`. The kernel function of the estimator.

Value

A `numeric` value corresponding to the best bandwidth.

See Also

[estimate_nw()]

Examples

```
## Not run:
# The model
## Let
m <- function(t) 4 * sin(1.5 * pi * t)

## Observation points
t <- runif(n = 200, min = 0, max = 1)
t <- sort(t)

## Measure error
e <- rnorm(n = 200, mean = 0, sd = 0.2)

## Regression model
y <- m(t) + e

plot(x = t, y = y, main = "Observed points and true regression function")
lines(x = t, y = m(t), type = "l", col = "red")

## Estimate the best bandwidth
bw_grid <- seq(1 / (2 * length(t)), length(t) ** (- 1/3), len = 100)
hbest <- estimate_nw_bw(y = y, t = t,
                        bw_grid = bw_grid,
                        smooth_ker = epanechnikov)

## Estimate the regression function
dt_nw <- estimate_nw(y = y, t = t,
                     tnew = seq(0.01, 0.99, len = 100),
                     h = hbest, smooth_ker = epanechnikov)

plot(x = dt_nw[, tnew], y = dt_nw[, yhat], type = "l", col = "blue",
     main = "Estimated and true regression function.")
lines(x = dt_nw[, tnew], y = m(dt_nw[, tnew]), type = "l", col = "red")
legend(x = 0.64, y = 4.1, fill = c("blue", "red"),legend = c("Estimated m", "True m"))


## End(Not run)
```

`estimate_sigma` *Estimate the the standard deviation of the observation error*

Description

Estimate the the standard deviation of the observation error

Usage

```
estimate_sigma(
  data,
  idcol = NULL,
  tcol = "tobs",
  ycol = "X",
  t = c(1/4, 1/2, 3/4)
)
```

Arguments

`data`	`data.table (or data.frame)` or `list` of `data.table (or data.frame)` or `list` of `list`.
	• If `data.table` It must contain the raw binding of the curve observations with at least 3 columns.
	  – `idcol` : The name of the column that contains the index of the curve in the sample. Each index of a curve is repeated as many times as it has observation points.
	  – `tcol` : The name of the column that contains the observation points associated to each curve index.
	  – `ycol` : The name of the column that contains the observed value of a curve at each point of observation and for each index of the curve.
	• If `list` of `data.table` In this case, each element of the given `list` corresponds to the observation scheme of a curve, which is given as `data.table` or `data.frame`. The data.table contains at least 2 columns.
	  – `tcol` : The name of the column that contains the observation points associated to the curve.
	  – `ycol` : The name of the column that contains the observed value of the curve.
	• If `list` of `list` In the latter case, the `data` is a list `list` where each element is the observation scheme of a curve given as a `list` of 2 vectors.
	  – `tcol` : The name of the vector that contains the observation points associated the curve.
	  – `ycol` : The name of the vector that contains the observed value of the curve.

`idcol`	`character`. If `data` is given as `data.table` or `data.frame`, it is the name of the column that contains the index of the curve in the sample. Each index of a curve is repeated as many times as it has observation points. Opposite, if f `data` is given as `list` of `data.table (or data.frame)` of `list` of `list`, `idcol = NULL`.
`tcol`	`character`. The name of the column (or vector) that contains the observation points associated to the curves.
`ycol`	`character`. The name of the column that contains the observed value of the curves.
`t`	`vector (numeric)`. Observation points at which we want to estimate the standard deviation of the error.

Value

A data.table with two columns: `t` and `sig` corresponding to the estimated standard deviation.

`get_nw_optimal_bw` *Estimate Nadayara-Watson optimal bandwidth on all or a subset of curves*

Description

Estimate Nadayara-Watson optimal bandwidth on all or a subset of curves

Usage

```
get_nw_optimal_bw(
  data,
  idcol = "id_curve",
  tcol = "tobs",
  ycol = "X",
  bw_grid = NULL,
  nsubset = NULL,
  smooth_ker = epanechnikov
)
```

Arguments

- `data` — `data.table (or data.frame)` or `list` of `data.table (or data.frame)` or `list` of `list`.
  - If `data.table` It must contain the raw binding of the curve observations with at least 3 columns.
    - `idcol` : The name of the column that contains the index of the curve in the sample. Each index of a curve is repeated as many times as it has observation points.
    - `tcol` : The name of the column that contains the observation points associated to each curve index.
    - `ycol` : The name of the column that contains the observed value of a curve at each point of observation and for each index of the curve.
  - If `list` of `data.table` In this case, each element of the given `list` corresponds to the observation scheme of a curve, which is given as `data.table` or `data.frame`. The data.table contains at least 2 columns.
    - `tcol` : The name of the column that contains the observation points associated to the curve.
    - `ycol` : The name of the column that contains the observed value of the curve.
  - If `list` of `list` In the latter case, the `data` is a list `list` where each element is the observation scheme of a curve given as a `list` of 2 vectors.
    - `tcol` : The name of the vector that contains the observation points associated the curve.
    - `ycol` : The name of the vector that contains the observed value of the curve.
- `idcol` — `character`. If `data` is given as `data.table` or `data.frame`, it is the name of the column that contains the index of the curve in the sample. Each index of a curve is repeated as many times as it has observation points. Opposite, if f `data` is given as `list` of `data.table (or data.frame)` of `list` of `list`, `idcol = NULL`.
- `tcol` — `character`. The name of the column (or vector) that contains the observation points associated to the curves.
- `ycol` — `character`. The name of the column that contains the observed value of the curves.
- `bw_grid` — `vector (numeric)`. The cross-validation bandwidth grid. Default `bw_grid = NULL` and so it will be set as an exponential grid using the average of the number of observation points per curve.

nsubset — integer (positive integer). The number of subset curves to be randomly and uniformly selected. Default nsubset = NULL and thus an optimal bandwidth is calculated for each curve.

smooth_ker — function. The kernel function of the Nadaraya-Watson estimator. Default smooth_ker = epanechnikov.

Value

A data.table containing the following columns.

- id_curve :The index of the curve.
- optbw : The optimal bandwidth obtained by Cross-Validation.

---

get_real_data_far_kenel
FAR kernel learned from the voltage curves of the electricity

---

Description

For more details see the vignette: vignette("hybrid-simulation-setup", package = "adaptiveFTS")

Usage

```
get_real_data_far_kenel(s = 0.2, t = 0.3, operator_norm = 0.5)
```

Arguments

s — numeric (positive). A vector or scalar value(s) between 0 and 1.

t — numeric (positive). A vector or scalar value(s) between 0 and 1.

operator_norm — numeric (positive). A scalar corresponding to the norm of the integral operator associated with this kernel function.

Value

A vector (or scalar) of numeric values corresponding to the value of the kernel function evaluated at (s, t).

Examples

```
# get the value of the kernel at (s,t) = (0.2, 0.3)
kerval <- get_real_data_far_kenel(s = 0.2, t = 0.3, operator_norm = 0.5)
kerval
```

get_real_data_mean *Mean function learned from the voltage curves of the electricity*

Description

For more details see the vignette: `vignette("hybrid-simulation-setup", package = "adaptiveFTS")`

Usage

```
get_real_data_mean(t = seq(0.1, 0.9, len = 10))
```

Arguments

t	`vector (numeric)`. Points at which we want to return the mean function. It can be a scalar.

Value

A `data.table` containing 2 columns.

- `t` : The vector or scalar `t`.
- mean : The values of the mean function evaluated at `t`.

Examples

```
t0 <- seq(0.1, 0.9, len = 10)
m <- get_real_data_mean(t = t0)
plot(x = t0, y = m, type = "b", col = "red",
     xlab = "t", ylab = "mean", main = "Mean function")
```

hurst_arctan *Arctan Hurst function*

Description

Arctan Hurst function that can be used to generate multifractional Brownian motion (mfBm). See the following paper https://doi.org/10.3390/fractalfract6020074.

Usage

```
hurst_arctan(t = seq(0.2, 0.8, len = 10))
```

Arguments

t	`vector (float)`. Points between 0 and 1 at which to evaluate the function.

Value

A `vector (float)` corresponding to the value of the function evaluated at `t`.

See Also

[hurst_linear()], [hurst_logistic()].

Examples

```
t0 <- seq(0.2, 0.8, len = 10)
htan <- hurst_arctan(t = t0)
plot(x = t0, y = htan, type = "b", col = "red")
```

---

hurst_linear *Linear Hurst function*

Description

Linear Hurst function that can be used to generate multifractional Brownian motion (mfBm). See the following paper https://doi.org/10.3390/fractalfract6020074.

Usage

```
hurst_linear(t = seq(0.2, 0.8, len = 10), h_left = 0.2, h_right = 0.8)
```

Arguments

`t`	`vector (float)`. Points between 0 and 1 at which to evaluate the function.
`h_left`	`Float`. A scalar value in the interval between 0 and 1 indicating the minimum of the function.
`h_right`	`Float`. A scalar value in the interval between 0 and 1 indicating the maximum of the function.

Value

A `vector (float)` corresponding to the value of the function evaluated at `t`.

See Also

[hurst_arctan()], [hurst_logistic()].

Examples

```
t0 <- seq(0.2, 0.8, len = 10)
hlinear <- hurst_linear(t = t0)
plot(x = t0, y = hlinear, type = "b", col = "red")
```

hurst_logistic *Logistic Hurst function*

Description

Logistic Hurst function that can be used to generate multifractional Brownian motion (mfBm). See the following paper https://doi.org/10.3390/fractalfract6020074.

Usage

```
hurst_logistic(
  t,
  h_left = 0.2,
  h_right = 0.8,
  slope = 30,
  change_point_position = 0.5
)
```

Arguments

t	vector (float). Points between 0 and 1 at which to evaluate the function.
h_left	Float. A scalar value in the interval between 0 and 1 indicating the minimum of the function.
h_right	Float. A scalar value in the interval between 0 and 1 indicating the maximum of the function.
slope	Float (positive). A scalar positive value corresponding to the slope of the logistic function.
change_point_position	Float. A scalar value in the interval between 0 and 1 corresponding ti the change point position.

Value

A vector (float) corresponding to the value of the function evaluated at t.

See Also

[hurst_arctan()], [hurst_linear()].

Examples

```
t0 <- seq(0.2, 0.8, len = 10)
hlogistic <- hurst_logistic(t = t0, h_left = 0.2,
                            h_right = 0.8, slope = 30,
                            change_point_position = 0.5)
plot(x = t0, y = hlogistic, type = "b", col = "red")
```

simulate_far *Functional Autoregressive process of order 1 (FAR(1)) simulation*

Description

Functional Autoregressive process of order 1 (FAR(1)) simulation

Usage

```
simulate_far(
  N = 2L,
  lambda = 70L,
  tdesign = "random",
  Mdistribution = rpois,
  tdistribution = runif,
  tcommon = seq(0.2, 0.8, len = 50),
  hurst_fun = hurst_logistic,
  L = 4,
  far_kernel = function(s, t) 9/4 * exp(-(t + 2 * s)^2),
  far_mean = function(t) 4 * sin(1.5 * pi * t),
  int_grid = 100L,
  burnin = 100L,
  remove_burnin = TRUE
)
```

Arguments

`N`	`integer`. Number of curves.
`lambda`	`integer`. Mean of the number of observations per curve.
`tdesign`	`character`. Type of the design. It is either 'random' or 'common'.
`Mdistribution`	`function`. Distribution of the number of observation points per curve. The first argument of the function must correspond to `N` and the second to `lambda`. Default `Mdistribution = rpois`.
`tdistribution`	`function (or NULL)`. Observation point distribution if `tdesign = 'random'` and `NULL` otherwise.
`tcommon`	`vector (float)`. Observation point vector if `tdesign = 'common'`. If `tdesign = 'random'` and if we want to run some tests at a particular observation position, this can also be specified.
`hurst_fun`	`function`. Hurst function. It can be `hurst_arctan`, `hurst_linear`, `hurst_logistic`.
`L`	`float (positive)`. Hölder constant.
`far_kernel`	`function`. Kernel function of the operator of the FAR(1).
`far_mean`	`function`. Mean function of the FAR(1).
`int_grid`	`integer`. Length of the grid used to approximate the integral.
`burnin`	`integer`. Burnin period of the FAR(1).
`remove_burnin`	`boolean`. If `TRUE`, burnin period is removed.

Value

A data.table containing 3 column :

- id_curve : Index of the curve. It goes from 1 to N.
- tobs : Sampled observation points, for each id_curve.
- ttag : Tag on the observations points, for each id_curve. It is either tcommon for common design grid or tcommon pour random design.
- far_mean : The mean of the process evaluate at tobs, for each id_curve.
- X : The process observed at tobs, for each id_curve.

Examples

```
## Not run:
dt_far <- simulate_far(N = 2L, lambda = 70L,
                       tdesign = "random",
                       Mdistribution = rpois,
                       tdistribution = runif,
                       tcommon = seq(0.2, 0.8, len = 50),
                       hurst_fun = hurst_logistic,
                       L = 4,
                       far_kernel = function(s,t) 9/4 * exp(- (t + 2 * s) ** 2),
                       far_mean = function(t) 4 * sin(1.5 * pi * t),
                       int_grid = 100L,
                       burnin = 100L,
                       remove_burnin = TRUE)


## End(Not run)
```

---

simulate_fBm *Draw a fractional Brownian motion sample path.*

Description

Draw a fractional Brownian motion sample path.

Usage

```
simulate_fBm(t = seq(0.2, 0.8, len = 20), hurst = 0.6, L = 1, tied = TRUE)
```

Arguments

t	vector (float). Grid of points between 0 and 1 where we want to generate the sample path.
hurst	float (positive). The Hurst exponent scalar value between 0 and 1.
L	float (positive). Hölder constant.
tied	boolean. If TRUE, the sample path is tied-down.

Value

A `data.table` containing 2 column : `t` and `mfBm`, the sample path.

Examples

```
t0 <- seq(0.2, 0.8, len = 20)
dt_fBm <- simulate_fBm(t = t0, hurst = 0.6, L = 1, tied = TRUE)
plot(x = dt_fBm$t, y = dt_fBm$fBm, type = "l", col = "red")
```

`simulate_fma` *Functional Moving Average process of order 1 (FMA(1)) simulation*

Description

Functional Moving Average process of order 1 (FMA(1)) simulation

Usage

```
simulate_fma(
  N = 2L,
  lambda = 70L,
  tdesign = "random",
  Mdistribution = rpois,
  tdistribution = runif,
  tcommon = seq(0.2, 0.8, len = 50),
  hurst_fun = hurst_logistic,
  L = 4,
  fma_kernel = function(s, t) 9/4 * exp(-(t + 2 * s)^2),
  fma_mean = function(t) 4 * sin(1.5 * pi * t),
  int_grid = 100L,
  burnin = 100L,
  remove_burnin = TRUE
)
```

Arguments

`N`	`integer`. Number of curves.
`lambda`	`integer`. Mean of the number of observations per curve.
`tdesign`	`character`. Type of the design. It is either 'random' or 'common'.
`Mdistribution`	`function`. Distribution of the number of observation points per curve. The first argument of the function must correspond to `N` and the second to `lambda`. Default `Mdistribution` = `rpois`.
`tdistribution`	`function (or NULL)`. Observation point distribution if `tdesign = 'random'` and `NULL` otherwise.
`tcommon`	`vector (float)`. Observation point vector if `tdesign = 'common'`. If `tdesign = 'random'` and if we want to run some tests at a particular observation position, this can also be specified.
`hurst_fun`	`function`. Hurst function. It can be `hurst_arctan`, `hurst_linear`, `hurst_logistic`.

L	float (positive). Hölder constant.
fma_kernel	function. Kernel function of the operator of the FMA(1).
fma_mean	function. Mean function of the FMA(1).
int_grid	integer. Length of the grid used to approximate the integral.
burnin	integer. Burnin period of the FMA(1).
remove_burnin	boolean. If TRUE, burnin period is removed.

Value

A data.table containing 3 column :

- id_curve : Index of the curve. It goes from 1 to N.
- tobs : Sampled observation points, for each id_curve.
- ttag : Tag on the observations points, for each id_curve. It is either tcommon for common design grid or tcommon pour random design.
- fma_mean : The mean of the process evaluate at tobs, for each id_curve.
- X : The process observed at tobs, for each id_curve.

Examples

```
## Not run:
dt_fma <- simulate_fma(N = 2L, lambda = 70L,
                       tdesign = "random",
                       Mdistribution = rpois,
                       tdistribution = runif,
                       tcommon = seq(0.2, 0.8, len = 50),
                       hurst_fun = hurst_logistic,
                       L = 4,
                       fma_kernel = function(s,t) 9/4 * exp(- (t + 2 * s) ** 2),
                       fma_mean = function(t) 4 * sin(1.5 * pi * t),
                       int_grid = 100L,
                       burnin = 100L,
                       remove_burnin = TRUE)
# plot simulated curve
library(ggplot2)

ggplot(data = dt_fma[ttag == "trandom", .("id_curve" = as.factor(id_curve), tobs, X)],
       mapping = aes(x = tobs, y = X, group = id_curve, color = id_curve)) +
  geom_line() +
  scale_colour_grey() +
  theme_minimal()


## End(Not run)
```

`simulate_mfBm` *Draw a multifractional Brownian motion sample path.*

Description

This function generates a sample path of a multifractional Brownian motion (mfBm) based on the provided Hurst function and other parameters.

Usage

```
simulate_mfBm(
  t = seq(0.2, 0.8, len = 50),
  hurst_fun = hurst_logistic,
  L = 1,
  shift_var = 1,
  tied = TRUE,
  ...
)
```

Arguments

`t`	`vector (float)`. Grid of points between 0 and 1 where the sample path will be generated.
`hurst_fun`	`function`. Hurst function. It can be `hurst_arctan`, `hurst_linear`, `hurst_logistic`, or any custom Hurst function.
`L`	`float (positive)`. Hölder constant.
`shift_var`	`float (positive)`. The variance of the shift Gaussian random variable. Default is `shift_var = 1`, meaning a normal random variable with mean 0 and variance 1 is added.
`tied`	`boolean`. If `TRUE`, the sample path is tied down.
`...`	Additional arguments for the Hurst function.

Value

A `data.table` containing 2 columns: `t` and `mfBm`, representing the grid points and the corresponding values of the mfBm sample path.

Examples

```
t0 <- seq(0.2, 0.8, len = 20)
dt_mfBm <- simulate_mfBm(t = t0, hurst_fun = hurst_logistic, L = 1, tied = TRUE)
plot(x = dt_mfBm$t, y = dt_mfBm$mfBm, type = "l", col = "red")
```

triangular — *Triangular kernel function*

Description

Triangular kernel function

Usage

```
triangular(u)
```

Arguments

u	`numeric`. Scalar or vector of numeric values at which to evaluate the function.

Value

A scalar or vector of `numeric`.

See Also

[biweight()], [triweight()], [tricube()], [epanechnikov()], and [uniform()].

tricube — *Tricube kernel function*

Description

Tricube kernel function

Usage

```
tricube(u)
```

Arguments

u	`numeric`. Scalar or vector of numeric values at which to evaluate the function.

Value

A scalar or vector of `numeric`.

See Also

[biweight()], [triweight()], [epanechnikov()], [triangular()], and [uniform()].

`triweight` *Triweight kernel function*

Description

Triweight kernel function

Usage

```
triweight(u)
```

Arguments

u	`numeric`. Scalar or vector of numeric values at which to evaluate the function.

Value

A scalar or vector of `numeric`.

See Also

[biweight()], [tricube()], [epanechnikov()], [triangular()], and [uniform()].

`uniform` *Uniform kernel function*

Description

Uniform kernel function

Usage

```
uniform(u)
```

Arguments

u	`numeric`. Scalar or vector of numeric values at which to evaluate the function.

Value

A scalar or vector of `numeric`.

See Also

[biweight()], [triweight()], [tricube()], [epanechnikov()], and [triangular()].

Index