Package 'multiscale'

July 16, 2020

Type Package

Title Multiscale Inference for Nonparametric Time Trends

Version 0.0.0.9000

Date 2020-05-04

Description This package performs a multiscale analysis of a nonparametric regression with time series errors or nonparametric regressions. In case of one regression, it is possible to detect where the trend function is increasing or decreasing. In case of multiple regression, the test identifies regions where the trend functions are different from each other. See Khismatullina and Vogt (2019) and Khismatullina and Vogt (2020) for more information and theory.

License GPL (>= 2)

Imports Rcpp (>= 1.0.4)

LinkingTo Rcpp

RoxygenNote 7.1.0

Encoding UTF-8

Suggests knitr, rmarkdown

VignetteBuilder knitr

R topics documented:

`multiscale-package` *Multiscale Inference for Nonparametric Time Trends*

Description

This package performs a multiscale analysis of a single nonparametric time trends (Khismatullina and Vogt (2019)) or multiple nonparametric time trends (Khismatullina and Vogt (2020)).

In case of a single nonparametric regression, the multiscale method to test qualitative hypotheses about the nonparametric time trend m in the model $Y_t = m(t/T) + \epsilon_t$ with time series errors ϵ_t is provided. The method was first proposed in Khismatullina and Vogt (2019). It allows to test for shape properties (areas of monotonic decrease or increase) of the trend m.

This method require an estimator of the long-run error variance $\sigma^2 = \sum_{l=-\infty}^{\infty} Cov(\epsilon_0, \epsilon_l)$. Hence, the package also provides the difference-based estimator for the case that the errors belong to the class of $AR(\infty)$ processes. The estimator was also proposed in Khismatullina and Vogt (2019).

In case of multiple nonparametric regressions, we provide the multiscale method to test qualitative hypotheses about the nonparametric time trends in the context of epidemic modelling. Specifically, we assume that the we observe a sample of the count data $\{\mathcal{X}_i = \{X_{it} : 1 \leq 1 \leq T\}\}$, where X_{it} are quasi-Poisson distributed with time-varying intensity parameter $\lambda_i(t/T)$. The multiscale method allows to test whether intenisty parameters are different or not, and if they are, it detects with a prespicified significance level the regions where these differences most probably occur. The method was introduced in Khismatullina and Vogt (2020) and can be used for comparing the rates of infection of COVID-19 across countries.

References

Khismatullina M., Vogt M. Multiscale inference and long-run variance estimation in non-parametric regression with time series errors //Journal of the Royal Statistical Society: Series B (Statistical Methodology). - 2019.

Khismatullina M., Vogt M. Simultaneous statistical inference for epidemic trends //???. - 2020.

See Also

https://rss.onlinelibrary.wiley.com/doi/full/10.1111/rssb.12347

`compute_minimal_intervals` *Computes the set of minimal intervals as described in Duembgen (2002)*

Description

Given a set of intervals, this function computes the corresponding subset of minimal intervals which are defined as follows. For a given set of intervals $\mathcal{K}$, all intervals $\mathcal{I}_k \in \mathcal{K}$ such that $\mathcal{K}$ does not contain a proper subset of $\mathcal{I}_k$ are called minimal.

This function is needed for illustrative purposes. The set of all the intervals where our test rejects the null hypothesis may be quite large, hence, we would like to focus our attention on the smaller subset, for which we are still able to make simultaneous confidence intervals. This subset is the subset of minimal intervals, and it helps us to to precisely locate the intervals of further interest.

More details can be found in Duembgen (2002) and Khismatullina and Vogt (2019, 2020)

Usage

```
compute_minimal_intervals(dataset)
```

Arguments

dataset	Set of the intervals. It needs to contain the following columns: "startpoint" - left end of the interval; "endpoint" - right end of the interval.

Value

Subset of minimal intervals

Examples

```
startpoint   <- c(0, 0.5, 1)
endpoint     <- c(2, 2, 2)
dataset      <- data.frame(startpoint, endpoint)
minimal_ints <- compute_minimal_intervals(dataset)
```

---

compute_quantiles *Computes quantiles of the gaussian multiscale statistics.*

Description

Quantiles from the gaussian version of the test statistics which are used to approximate the critical values for the multiscale test.

Usage

```
compute_quantiles(
  t_len,
  n_ts = 1,
  grid = NULL,
  ijset = NULL,
  sigma = 1,
  deriv_order = 0,
  sim_runs = 1000,
  probs = seq(0.5, 0.995, by = 0.005)
)
```

Arguments

t_len	Sample size.
n_ts	Number of time series analyzed. Default is 1.
grid	Grid of location-bandwidth points as produced by the function construct_grid or construct_weekly_grid, list with the elements 'gset', 'bws', 'gtype'. If not provided, then the defalt grid is produced and used. For the construction of the default grid, see construct_grid.

ijset	A matrix of integers. In case of multiple time series, we need to know which pairwise comparisons to perform. This matrix consists of all pairs of indices (i, j) that we want to compare. If not provided, then all possible pairwise comparison are performed.
sigma	Value of $\sqrt{\sigma^2}$. In case of n_ts = 1, σ^2 denotes the long-run error variance, and in case of n_ts > 1, σ^2 denotes the overdispersion parameter. If not given, then the default is 1.
deriv_order	In case of a single time series analysed, this parameter denotes the order of the derivative of the trend function that is being estimated. Default is 0.
sim_runs	Number of simulation runs to produce quantiles. Default is 1000.
probs	A numeric vector of probability levels $(1 - \alpha)$ for which the quantiles are computed. Default is $(0.5, 0.505, 0.51, \ldots, 0.995)$.

Value

Matrix with 2 rows where the first row contains the vector of probabilities (probs) and the second contains corresponding quantiles of the gaussian statistics distribution.

Examples

```
compute_quantiles(100)
```

---

compute_statistics *Calculates the value of the test statistics both for single time series analysis and multiple time series analysis.*

Description

Calculates the value of the test statistics both for single time series analysis and multiple time series analysis.

Usage

```
compute_statistics(
  data,
  sigma,
  n_ts = 1,
  grid = NULL,
  ijset = NULL,
  deriv_order = 0
)
```

Arguments

data	Vector (in case of n_ts = 1) or matrix (in case of n_ts > 1) that contains (a number of) time series that needs to be analyzed. In the latter case, each column of the matrix must contain one time series.
sigma	The estimator of the square root of the long-run variance σ in case of n_ts = 1, or the estimator of the overdispersion parameter σ in case of n_ts > 1.
n_ts	Number of time series analysed. Default is 1.

grid	Grid of location-bandwidth points as produced by the functions construct_grid or construct_weekly_grid, it is a list with the elements 'gset', 'bws', 'gtype'. If not provided, then the defalt grid is used. For the construction of the default grid, see construct_grid.
ijset	In case of multiple time series (n_ts > 1), we need to know which pairs of time series to compare. This matrix consists of all pairs of indices (i, j) that we want to compare. If not provided, then all possible pairwise comparison are performed.
deriv_order	In case of a single time series, this denotes the order of the derivative of the trend that we estimate. Default is 0.

Value

In case of n_ts = 1, the function returns a list with the following elements:

stat	Value of the multiscale statistics.
gset_with_vals	A matrix that contains the values of the normalised kernel averages for each pair of location-bandwidth with the corresponding location and bandwidth.

In case of n_ts > 1, the function returns a list with the following elements:

stat	Value of the multiscale statistics.
stat_pairwise	Matrix of the values of the pairwise statistics.
ijset	The matrix that consists of all pairs of indices (i, j) that we compared. The order of these pairs corresponds to the order in the list gset_with_vals.
gset_with_vals	A list of matrices, each matrix corresponding to a specific pairwise comparison. The order of the list is determined by ijset. Each matrix contains the values of the normalisedkernel averages for each pair of location-bandwidth with the corresponding location and bandwidth.

---

construct_grid *Computes the location-bandwidth grid for the multiscale test.*

Description

Computes the location-bandwidth grid for the multiscale test.

Usage

```
construct_grid(t, u_grid = NULL, h_grid = NULL, deletions = NULL)
```

Arguments

t	Sample size.
u_grid	Vector of location points in the unit interval $[0, 1]$. If NULL, a default grid is used.
h_grid	Vector of bandwidths, each bandwidth is supposed to lie in $(0, 0.5)$. If NULL, a default grid is used.
deletions	Logical vector of the length len(u.grid) * len(h.grid). Each element is either TRUE, which means that the corresponding location-bandwidth point (u, h) is NOT deleted from the grid, or FALSE, which means that the corresponding location-bandwidth point (u, h) IS deleted from the grid. Default is NULL in which case nothing is deleted. See vignette for the use.

Value

A list with the following elements:

gset	Matrix of location-bandwidth points (u, h) that remains after deletions, the i-th row gset[i,] corresponds to the i-th point (u, h).
bws	Vector of bandwidths (after deletions).
lens	Vector of length = length(bws), lens[i] gives the number of locations in the grid for the i-th bandwidth level.
gtype	Type of grid that is used, either 'default' or 'non-default'.
gset_full	Matrix of all location-bandwidth pairs (u, h) including deleted ones.
pos_full	Logical vector indicating which points (u, h) have been deleted.

Examples

```
construct_grid(100)
construct_grid(100, u_grid = seq(from = 0.05, to = 1, by = 0.05),
               h_grid = c(0.1, 0.2, 0.3, 0.4))
```

---

construct_weekly_grid *Computes the location-bandwidth weekly grid for the multiscale test.*

Description

Computes the location-bandwidth weekly grid for the multiscale test.

Usage

```
construct_weekly_grid(t, min_len = 7, nmbr_of_wks = 4)
```

Arguments

t	Sample size.
min_len	Minimal length of the interval considered. The grid then consists of intervals with lengths min_len, 2 * min_len, 3 * min_len, ... Default is 7, i.e. a week.
nmbr_of_wks	Number that determines the longest intervals in the grid: the length of this interval is calculated then as min_len * nmbr_of_wks. Default is 4.

Value

A list with the following elements:

gset	Matrix of location-bandwidth points (u, h) the i-th row gset[i,] corresponds to the i-th point (u, h).
bws	Vector of bandwidths.
lens	Vector of length = length(bws), lens[i] gives the number of locations in the grid for the i-th bandwidth level.
gtype	Type of grid that is used, always 'default'.
gset_full	Matrix of all location-bandwidth pairs (u, h).

Examples

```
construct_weekly_grid(100)
construct_weekly_grid(100, min_len = 7, nmbr_of_wks = 2)
```

`estimate_lrv` *Computes estimator of the long-run variance of the error terms.*

Description

A difference based estimator for the coefficients and long-run variance in case of a nonparametric regression model are AR(p).

Specifically, we assume that we observe $Y(t)$ that satisfy the following equation:

$$Y(t) = m(t/T) + \epsilon_t.$$

Here, $m(\cdot)$ is an unknown function, and the errors ϵ_t are AR(p) with p known. Specifically, we ler $\{\epsilon_t\}$ be a process of the form

$$\epsilon_t = \sum_{j=1}^{p} a_j \epsilon_{t-j} + \eta_t,$$

where $a_1, a_2, \ldots, a_p$ are unknown coefficients and η_t are i.i.d.\ with $E[\eta_t] = 0$ and $E[\eta_t^2] = \nu^2$.

This function produces an estimator $\widehat{\sigma}^2$ of the long-run variance

$$\sigma^2 = \sum_{l=-\infty}^{\infty} cov(\epsilon_0, \epsilon_l)$$

of the error terms, as well as estimators $\widehat{a}_1, \ldots, \widehat{a}_p$ of the coefficients $a_1, a_2, \ldots, a_p$ and an estimator $\widehat{\nu}^2$ of the innovation variance ν^2.

The exact estimation procedure as well as description of the tuning parameters needed for this estimation can be found in Khismatullina and Vogt (2019).

Usage

```
estimate_lrv(data, q, r_bar, p)
```

Arguments

`data`	A vector of $Y(1), Y(2), \ldots, Y(T)$.
`q, r_bar`	Tuning parameters.
`p`	AR order of the error terms.

Value

A list with the following elements:

`lrv`	Estimator of the long run variance of the error terms σ^2.
`ahat`	Vector of length p of estimated AR coefficients $a_1, a_2, \ldots, a_p$.
`vareta`	Estimator of the variance of the innovation term ν^2.

References

Khismatullina M., Vogt M. Multiscale inference and long-run variance estimation in non-parametric regression with time series errors //Journal of the Royal Statistical Society: Series B (Statistical Methodology). - 2019.

`multiscale_test` *Carries out the multiscale test given that the values the estimatates of long-run variance have already been computed.*

Description

Carries out the multiscale test given that the values the estimatates of long-run variance have already been computed.

Usage

```
multiscale_test(
  data,
  sigma,
  n_ts = 1,
  grid = NULL,
  ijset = NULL,
  alpha = 0.05,
  sim_runs = 1000,
  deriv_order = 0
)
```

Arguments

`data`	Vector (in case of n_ts = 1) or matrix (in case of n_ts > 1) that contains (a number of) time series that needs to be analyzed. In the latter case, each column of the matrix must contain one time series.
`sigma`	The estimator of the square root of the long-run variance σ in case of n_ts = 1, or the estimator of the overdispersion parameter σ in case of n_ts > 1.
`n_ts`	Number of time series analysed. Default is 1.
`grid`	Grid of location-bandwidth points as produced by the functions `construct_grid` or `construct_weekly_grid`, it is a list with the elements 'gset', 'bws', 'gtype'. If not provided, then the defalt grid is used. For the construction of the default grid, see `construct_grid`.
`ijset`	In case of multiple time series (n_ts > 1), we need to know which pairs of time series to compare. This matrix consists of all pairs of indices (i, j) that we want to compare. If not provided, then all possible pairwise comparison are performed.
`alpha`	Significance level. Default is 0.05.
`sim_runs`	Number of simulation runs to produce quantiles. Default is 1000.
`deriv_order`	In case of a single time series, this denotes the order of the derivative of the trend that we estimate. Default is 0.

Value

In case of n_ts = 1, the function returns a list with the following elements:

`quant` Quantile that was used for testing calculated from the gaussian distribution.

`statistics` Value of the multiscale statistics.

`test_matrix` Matrix of the test results for the multiscale test defined in Khismatullina and Vogt (2019). The matrix is coded as follows:

- test_matrix[i,j] = -1: test rejects the null for the j-th location u and the i-th bandwidth h and indicates a decrease in the trend;
- test_matrix[i,j] = 0: test does not reject the null for the j-th location u and the i-th bandwidth h;
- test_matrix[i,j] = 1: test rejects the null for the j-th location u and the i-th bandwidth h and indicates an increase in the trend;
- test_matrix[i,j] = 2: no test is carried out at j-th location u and i-th bandwidth h (because the point (u, h) is excluded from the grid as specified by the 'deletions' option in the function `construct_grid`)

.

`gset_with_vals` A matrix that contains the values of the normalised kernel averages and test results for each pair of location-bandwidth with the corresponding location and bandwidth.

In case of n_ts > 1, the function returns a list with the following elements:

`quant` Quantile that was used for testing calculated from the gaussian distribution.stat

Value of the multiscale statistics.

`statistics` Value of the multiscale statistics.

`stat_pairwise` Matrix of the values of the pairwise statistics.

`ijset` The matrix that consists of all pairs of indices (i, j) that we compared. The order of these pairs corresponds to the order in the list gset_with_vals.

`gset_with_vals` A list of matrices, each matrix corresponding to a specific pairwise comparison. The order of the list is determined by ijset. Each matrix contains the values of the normalisedkernel averages for each pair of location-bandwidth with the corresponding location and bandwidth.

---

plot_sizer_map *Plots SiZer map from the test results of the multiscale testing procedure.*

Description

Plots SiZer map from the test results of the multiscale testing procedure.

Usage

```
plot_sizer_map(
  u_grid,
  h_grid,
  test_results,
  plot_title = NA,
  greyscale = FALSE,
  ...
)
```

Arguments

u_grid	Vector of location points in the unit interval $[0, 1]$.
h_grid	Vector of bandwidths from $(0, 0.5)$.
test_results	Matrix of test results created by multiscale_test.
plot_title	Title of the plot. Default is NA and no title is written.
greyscale	Whether SiZer map is plotted in grey scale. Default is FALSE.
...	Any further options to be passed to the image function.

---

select_order — *Calculates different information criterions for a single time series or multiple time series with AR(p) errors based on the long-run variance estimator(s) for a range of tuning parameters and different orders p.*

Description

This function fits AR(1), ... AR(9) models for all given time series and calculates different information criterions (FPE, AIC, AICC, SIC, HQ) for each of these fits. The result is the best fit in terms of minimizing the infromation criteria.

Usage

```
select_order(data, q = NULL, r = 5:15)
```

Arguments

data	One or a number of time series in a matrix. Column names of the matrix should be reasonable
q	A vector of integers that consisits of different tuning parameters to analyse. If not supplied, q is taken to be $[2 \log T] : ([2\sqrt{T}] + 1)$.
r	A vector of integers that consisits of different tuning parameters r_bar for estimate_lrv. If not supplied, $r = 5, \ldots, 15$.

Value

A list with a number of elements:

orders	A vector of chosen orders of length equal to the number of time series. For each time series the order is calculated as $\max(which.min(FPE), ...which.min(HQ))$
...	Matrices with the orders that were selected (among $1, \ldots, 9$) for each information criterion. One matrix for each time series.

Index