Package ‘lg’

July 22, 2025

Title Locally Gaussian Distributions: Estimation and Methods

Version 0.4.1

Description An implementation of locally Gaussian distributions. It provides methods for implementing locally Gaussian multivariate density estimation, conditional density estimation, various independence tests for iid and time series data, a test for conditional independence and a test for financial contagion.

Depends R (>= 3.4)

License GPL-3

Encoding UTF-8

LazyData true

RoxygenNote 6.1.1

Suggests testthat

Imports mvtnorm, localgauss, logspline, ggplot2, ks, np, tseries

NeedsCompilation no

Author Håkon Otneim [aut, cre]

Maintainer Håkon Otneim <hakon.otneim@nhh.no>

Repository CRAN

Date/Publication 2019-12-05 12:40:02 UTC

Contents

`accept_reject` *Generate sample from a conditional density estimate*

Description

Generate a sample from a locally Gaussian conditional density estimate using the accept-reject algorithm. If the `transform_to_marginal_normality`- component of the lg_object is `TRUE`, the replicates will be on the standard normal scale.

Usage

```
accept_reject(lg_object, condition, n_new, nodes, M = NULL,
  M_sim = 1500, M_corr = 1.5, n_corr = 1.2, return_just_M = FALSE,
  extend = 0.3)
```

Arguments

`lg_object`	An object of type `lg`, as produced by the `lg_main`-function
`condition`	The value of the conditioning variables
`n_new`	The number of observations to generate
`nodes`	Either the number of equidistant nodes to generate, or a vector of nodes supplied by the user
`M`	The value for M in the accept-reject algorithm if already known
`M_sim`	The number of replicates to simulate in order to find a value for M
`M_corr`	Correction factor for M, to be on the safe side
`n_corr`	Correction factor for n_new, so that we mostly will generate enough observations in the first go
`return_just_M`	`TRUE` if we just want to find M, without actually generating any replications.
`extend`	How far to extend the grid beyond the extreme data points when interpolating, in share of the range

---

`bw_select` *Bandwidth selection for local Gaussian correlation.*

Description

Takes a matrix of data points and returns the bandwidths used for estimating the local Gaussian correlations.

Usage

```
bw_select(x, bw_method = "plugin", est_method = "1par",
  plugin_constant_marginal = 1.75, plugin_exponent_marginal = -1/5,
  plugin_constant_joint = 1.75, plugin_exponent_joint = -1/6,
  tol_marginal = 10^(-3), tol_joint = 10^(-3))
```

Arguments

`x`	A matrix or data frame with data, one column per variable, one row per observation.
`bw_method`	The method used for bandwidth selection. Must be either `"cv"` (cross-validation, slow, but accurate) or `"plugin"` (fast, but crude).
`est_method`	The estimation method, must be either "1par", "5par" or "5par_marginals_fixed", see `lg_main`.
`plugin_constant_marginal`	The constant `c` in `cn^a` used for finding the plugin bandwidth for locally Gaussian marginal density estimates, which we need if estimation method is "5par_marginals_fixed".

`plugin_exponent_marginal`
: The constant `a` in `cn^a` used for finding the plugin bandwidth for locally Gaussian marginal density estimates, which we need if estimation method is "5par_marginals_fixed".

`plugin_constant_joint`
: The constant `c` in `cn^a` used for finding the plugin bandwidth for estimating the pairwise local Gaussian correlation between two variables.

`plugin_exponent_joint`
: The constant `a` in `cn^a` used for finding the plugin bandwidth for estimating the pairwise local Gaussian correlation between two variables.

`tol_marginal`
: The absolute tolerance in the optimization for finding the marginal bandwidths when using cross validation.

`tol_joint`
: The absolute tolerance in the optimization for finding the joint bandwidths when using cross-validation.

Details

This is the main bandwidth selection function within the framework of locally Gaussian distributions as described in Otneim and Tjøstheim (2017). This function takes in a data set of arbitrary dimension, and calculates the bandwidths needed to find the pairwise local Gaussian correlations, and is mainly used by the main `lg_main` wrapper function.

Value

A list with three elements, `marginal` contains the bandwidths used for the marginal locally Gaussian estimation, `marginal_convergence` contains the convergence flags for the marginal bandwidths, as returned by the `optim` function, and `joint` contains the pairwise bandwidths and convergence flags.

References

Otneim, Håkon, and Dag Tjøstheim. "The locally gaussian density estimator for multivariate data." Statistics and Computing 27, no. 6 (2017): 1595-1616.

Examples

```
x <- cbind(rnorm(100), rnorm(100), rnorm(100))
bw <- bw_select(x)
```

`bw_select_cv_bivariate` *Cross-validation for bivariate distributions*

Description

Uses cross-validation to find the optimal bandwidth for a bivariate locally Gaussian fit

Usage

```
bw_select_cv_bivariate(x, tol = 10^(-3), est_method = "1par",
  bw_marginal = NULL)
```

Arguments

`x`	The matrix of data points.
`tol`	The absolute tolerance in the optimization, used by the `optim`-function.
`est_method`	The estimation method for the bivariate fit. If estimation method is `5par_marginals_fixed`, the marginal bandwidths must be supplied as well through the argument `bw_marginal`. This is automatically handled by the `lg_main` wrapper function.
`bw_marginal`	The bandwidths for estimation of the marginals if method `5par_fixed_marginals` is used.

Details

This function provides an implementation for the Cross Validation algorithm for bandwidth selection described in Otneim & Tjøstheim (2017), Section 4. Let $\hat{f}_h(x)$ be the bivariate locally Gaussian density estimate obtained using the bandwidth h, then this function returns the bandwidth that maximizes

$$CV(h) = n^{-1} \sum_{i=1}^{n} \log \hat{f}_h^{(-i)}(x_i),$$

where $\hat{f}_h^{(-i)}$ is the density estimate calculated without observation x_i.

The recommended use of this function is through the `lg_main` wrapper function.

Value

The function returns a list with two elements: `bw` is the selected bandwidths, and `convergence` is the convergence flag returned by the `optim`-function.

References

Otneim, Håkon, and Dag Tjøstheim. "The locally gaussian density estimator for multivariate data." Statistics and Computing 27, no. 6 (2017): 1595-1616.

Examples

```
  ## Not run:
    x <- cbind(rnorm(100), rnorm(100))
    bw <- bw_select_cv_univariate(x)

## End(Not run)
```

bw_select_cv_trivariate

Cross-validation for trivariate distributions

Description

Uses cross-validation to find the optimal bandwidth for a trivariate locally Gaussian fit

Usage

```
bw_select_cv_trivariate(x, tol = 10^(-3))
```

Arguments

x	The matrix of data points.
tol	The absolute tolerance in the optimization, used by the optim-function.

Details

This function provides an implementation for the Cross Validation algorithm for bandwidth selection described in Otneim & Tjøstheim (2017), Section 4, but for trivariate distributions. Let $\hat{f}_h(x)$ be the trivariate locally Gaussian density estimate obtained using the bandwidth h, then this function returns the bandwidth that maximizes

$$CV(h) = n^{-1} \sum_{i=1}^{n} \log \hat{f}_h^{(-i)}(x_i),$$

where $\hat{f}_h^{(-i)}$ is the density estimate calculated without observation x_i.

The recommended use of this function is through the `lg_main` wrapper function.

Value

The function returns a list with two elements: `bw` is the selected bandwidths, and `convergence` is the convergence flag returned by the `optim`-function.

References

Otneim, Håkon, and Dag Tjøstheim. "The locally gaussian density estimator for multivariate data." Statistics and Computing 27, no. 6 (2017): 1595-1616.

Examples

```
  ## Not run:
    x <- cbind(rnorm(100), rnorm(100), rnorm(100))
    bw <- bw_select_cv_trivariate(x)

## End(Not run)
```

bw_select_cv_univariate
Cross-validation for univariate distributions

Description

Uses cross-validation to find the optimal bandwidth for a univariate locally Gaussian fit

Usage

```
bw_select_cv_univariate(x, tol = 10^(-3))
```

Arguments

x	The vector of data points.
tol	The absolute tolerance in the optimization, passed to the optim-function using the BFGS-method.

Details

This function provides the univariate version of the Cross Validation algorithm for bandwidth selection described in Otneim & Tjøstheim (2017), Section 4. Let $\hat{f}_h(x)$ be the univariate locally Gaussian density estimate obtained using the bandwidth h, then this function returns the bandwidth that maximizes

$$CV(h) = n^{-1} \sum_{i=1}^{n} \log \hat{f}_h^{(-i)}(x_i),$$

where $\hat{f}_h^{(-i)}$ is the density estimate calculated without observation x_i.

Value

The function returns a list with two elements: `bw` is the selected bandwidth, and `convergence` is the convergence flag returned by the `optim`-function.

References

Otneim, Håkon, and Dag Tjøstheim. "The locally gaussian density estimator for multivariate data." Statistics and Computing 27, no. 6 (2017): 1595-1616.

Examples

```
x <- rnorm(100)
bw <- bw_select_cv_univariate(x)
```

bw_select_plugin_multivariate

Plugin bandwidth selection for multivariate data

Description

Returns a plugin bandwidth for multivariate data matrices for the estimation of local Gaussian correlations

Usage

```
bw_select_plugin_multivariate(x = NULL, n = nrow(x), c = 1.75,
  a = -1/6)
```

Arguments

x	The data matrix.
n	The number of data points. Can provide only this if we do not want to supply the entire data vector.
c	A constant, se details.
a	A constant, se details.

Details

This function takes in a data matrix with `n` rows, and returns a the real number `c*n^a`, which is a quick and dirty way of selecting a bandwidth for locally Gaussian density estimation. The number `c` is by default set to `1.75`, and `c = -1/6` is the usual exponent, that stems from the asymptotic convergence rate of the density estimate. This function is usually called from the `lg_main` wrapper function.

Value

A number, the selected bandwidth.

Examples

```
x <- cbind(rnorm(100), rnorm(100))
bw <- bw_select_plugin_multivariate(x = x)
bw <- bw_select_plugin_multivariate(n = 100)
```

bw_select_plugin_univariate
Plugin bandwidth selection for univariate data

Description

Returns a plugin bandwidth for data vectors for use with univariate locally Gaussian density estimation

Usage

```
bw_select_plugin_univariate(x = NULL, n = length(x), c = 1.75,
  a = -1/5)
```

Arguments

x	The data vector.
n	The number of data points. Can provide only this if we do not want to supply the entire data vector.
c	A constant, se details.
a	A constant, se details.

Details

This function takes in a data vector of length `n`, and returns a the real number `c*n^a`, which is a quick and dirty way of selecting a bandwidth for univariate locally Gaussian density estimation. The number `c` is by default set to `1.75`, and `c = -1/5` is the usual exponent that stems from the asymptotic convergence rate of the density estimate. Recommended use of this function is through the `lg_main` wrapper function.

Value

A number, the selected bandwidth.

Examples

```
x <- rnorm(100)
bw <- bw_select_plugin_univariate(x = x)
bw <- bw_select_plugin_univariate(n = 100)
```

bw_simple *Create simple bandwidth object*

Description

Create a simple bandwidths object for local Gaussian correlations

Usage

```
bw_simple(joint = 1, marg = NA, x = NULL, dim = NULL)
```

Arguments

joint	Joint bandwidth
marg	Marginal bandwidths
x	The data set
dim	The number of variables

Details

This function provides a quick way of producing a bandwidth object that may be used in the `lg_main()`-function. The user must specify a bandwidth `joint` that is used for all joint bandwidths, and the user may specify `marg`, a marginal bandwidth that will be used for all marginal bandwidths. This is needed if the subsequent analyses use `est_method = "5par_marginals_fixed"`.

The function must know the dimension of the problem, which is achieved by either supplying the data set `x` or the number of variables `dim`.

Examples

```
bw_object <- bw_simple(joint = 1, marg = 1, dim = 3)
```

check_bw_bivariate *Check bandwidth vector*

Description

Checks that the bandwidth vector supplied to the bivariate density function is a numeric vector of length 2.

Usage

```
check_bw_bivariate(bw)
```

Arguments

bw	The bandwidth vector to be checked

check_bw_method *Check bw method*

Description

Checks that the bandwidth method is one of the allowed values, currently "cv" or "plugin".

Usage

```
check_bw_method(bw_method)
```

Arguments

bw_method	Check if equal to "cv" or "plugin"

check_bw_trivariate *Check bandwidth vector*

Description

Checks that the bandwidth vector supplied to the bivariate density function is a numeric vector of length 3.

Usage

```
check_bw_trivariate(bw)
```

Arguments

bw	The bandwidth vector to be checked

check_data *Check the data and grid*

Description

Checks that the data or grid provided is of the correct form. This function is an auxiliary function that can quickly check that a supplied data set or grid is a matrix or a data frame, and that it has the correct dimension, as defined by the `dim_check` parameter. The `type` argument is simply a character vector "data" or "grid" that is used for printing error messages.

Usage

```
check_data(x, dim_check = NA, type)
```

Arguments

x	Data or grid
dim_check	How many columns do we expect?
type	Is it the "grid" or "data" for use in error messages.

---

check_dmvnorm_arguments — *Check the arguments for the* dmvnorm_wrapper *function*

Description

Checks that the arguments provided to the dmvnorm_wrapper-function are numerical vectors, all having the same lengths.

Usage

```
check_dmvnorm_arguments(eval_points, mu_1, mu_2, sig_1, sig_2, rho)
```

Arguments

eval_points	A kx2 matrix with evaluation points
mu_1	The first expectation vector
mu_2	The second expectation vector
sig_1	The first standard deviation vector
sig_2	The second standard deviation vector
rho	The correlation vector

---

check_est_method — *Check estimation method*

Description

Checks that the estimation method is one of the allowed values, currently "1par", "5par" and "5par_marginals_fixed".

Usage

```
check_est_method(est_method)
```

Arguments

est_method	Check if equal to a valid value

check_lg — *Check that an object has class "lg"*

Description

Checks that the provided object has class `lg`.

Usage

```
check_lg(check_object)
```

Arguments

`check_object`	The object to be checked

ci_test — *Test for conditional independence*

Description

Perform a test for conditional independence between the first two variables in the data set, given the remaining variables.

Usage

```
ci_test(lg_object, h = function(x) x^2, S = function(y) rep(T,
  nrow(y)), n_rep = 500, nodes = 100, M = NULL, M_sim = 1500,
  M_corr = 1.5, n_corr = 1.2, extend = 0.3, return_time = TRUE)
```

Arguments

`lg_object`	An object of type `lg`, as produced by the `lg_main`-function
`h`	The `h`-function used in the calculation of the test statistic. The default value is `h(x) = x^2`.
`S`	The integration area in the test statistic. Logical function that takes grid points as argument.
`n_rep`	The number of replicated bootstrap samples
`nodes`	Either the number of equidistant nodes to generate, or a vector of nodes supplied by the user
`M`	The value for M in the accept-reject algorithm if already known
`M_sim`	The number of replicates to simulate in order to find a value for M
`M_corr`	Correction factor for M, to be on the safe side

n_corr	Correction factor for n_new, so that we mostly will generate enough observations in the first go
extend	How far to extend the grid beyond the extreme data points when interpolating, in share of the range
return_time	Measure how long the test takes to run, and return along with the test result

---

ci_test_statistic — *Calculate the value of the test statistic for the conditional independence test*

Description

Calculate the test statistic in the test for conditional independence between the first two variables in the data set, given the remaining variables.

Usage

```
ci_test_statistic(lg_object, h = function(x) x^2, S = function(y)
  rep(T, nrow(y)))
```

Arguments

lg_object	An object of type lg, as produced by the lg_main-function
h	The h-function used in the calculation of the test statistic. The default value is h(x) = x^2.
S	The integration area in the test statistic. Logical function that takes grid points as argument.

---

clg — *The locally Gaussian conditional density estimator*

Description

Estimate a conditional density function using locally Gaussian approximations.

Usage

```
clg(lg_object, grid = NULL, condition = NULL,
  normalization_points = NULL, fixed_grid = NULL)
```

Arguments

`lg_object` An object of type `lg`, as produced by the `lg_main`-function.

`grid` A matrix of grid points, where we want to evaluate the density estimate. Number of columns *must* be the same as number of variables in X1.

`condition` A vector with conditions for the variables that we condition upon. Length of this vector *must* be the same as the number of variables in X2. The function will throw an error of there is any discrepancy in the dimensions of the `grid`, `condition` and data set.

`normalization_points` How many grid points for approximating the integral of the density estimate, to use for normalization?

`fixed_grid` Not used presently.

Details

This function is the conditional version of the locally Gaussian density estimator (LGDE), described in Otneim & Tjøstheim (2018). The function takes as arguments an `lg`-object as produced by the main `lg_main`- function, a grid of points where the density estimate should be estimated, and a set of conditions.

The variables must be sorted before they are supplied to this function. It will always assume that the free variables come before the conditioning variables.

Assume that X is a stochastic vector with two components X1 and X2. This function will thus estimate the conditional density of X1 given a specified value of X2.

Value

A list containing the conditional density estimate as well as all the running parameters that has been used. The elements are:

- `f_est`: The estimated conditional density.
- `c_mean`: The estimated local conditional means as defined in equation (10) of Otneim & Tjøstheim (2017).
- `c_cov`: The estimated local conditional covariance matrices as defined in equation (11) of Otneim & Tjøstheim (2017).
- `x`: The data set.
- `bw`: The bandwidth object.
- `transformed_data`: The data transformed to approximate marginal standard normality (if selected).
- `normalizing_constants`: The normalizing constants used to transform data and grid back and forth to the marginal standard normality scale, as seen in eq. (8) of Otneim & Tjøstheim (2017) (if selected).
- `grid`: The grid where the estimation was performed, on the original scale.
- `transformed_grid`: The grid where the estimation was performed, on the marginal standard normal scale.

- `normalization_points` Number of grid points used to approximate the integral of the density estimate, in order to normalize?
- `normalization_constant` If approximated, the integral of the non-normalized density estimate. NA if not normalized.
- `density_normalized` Logical, indicates whether the final density estimate (contained in f_est) has been approximately normalized to have unit integral.

References

Otneim, Håkon, and Dag Tjøstheim. "Conditional density estimation using the local Gaussian correlation" Statistics and Computing 28, no. 2 (2018): 303-321.

Examples

```
# A 3 variate example
x <- cbind(rnorm(100), rnorm(100), rnorm(100))

# Generate the lg-object with default settings
lg_object <- lg_main(x)

# Estimate the conditional density of X1|X2 = 0, X3 = 1 on a small grid
cond_dens <- clg(lg_object, grid = matrix(-4:4, ncol = 1), condition = c(0, 1))
```

---

cont_test *Test for financial contagion*

Description

Test for financial contagion by means of the local Gaussian correlation.

Usage

```
cont_test(lg_object_nc, lg_object_c,
  grid_range = quantile(rbind(lg_object_nc$x, lg_object_c$x), c(0.05,
  0.95)), grid_length = 30, n_rep = 1000, weight = function(y) {
  rep(1, nrow(y)) })
```

Arguments

`lg_object_nc`	An object of type `lg`, as produced by the `lg_main`-function for the observations covering the non-crisis period. The data must be two dimensional.
`lg_object_c`	An object of type `lg`, as produced by the `lg_main`-function for the observations covering the crisis period. The data must be two dimensional.
`grid_range`	This test measures the local correlations a long the diagonal specified by this vector of length two.
`grid_length`	The number of grid points.
`n_rep`	The number of bootstrap replicates.
`weight`	Weight function

Details

This function is an implementation of the test for financial contagion developed by Støve, Tjøstheim and Hufthammer (2013). They test whether the local correlations between two financial time series are different before and during crisis times. The distinction between crisis and non-crisis times must be made by the user.

Value

A list containing the test result as well as various parameters. The elements are:

- `observed` The observed value of the test statistic.
- `replicated` The replicated values of the test statistic.
- `p_value` The p-value of the test.
- `local_correlations` The local correlations measured along the diagonal, for the non-crisis and crisis periods respectively.

References

Støve, Bård, Dag Tjøstheim, and Karl Ove Hufthammer. "Using local Gaussian correlation in a nonlinear re-examination of financial contagion." Journal of Empirical Finance 25 (2014): 62-82.

Examples

```
# Run the test on some built-in stock data
data(EuStockMarkets)
x <- apply(EuStockMarkets, 2, function(x) diff(log(x)))[, 1:2]

# Define the crisis and non-crisis periods (arbitrarily for this simple
# example)
non_crisis <- x[1:100, ]
crisis     <- x[101:200, ]

# Create the lg-objects, with parameters that match the applications in the
# original publication describibg the test
lg_object_nc <- lg_main(non_crisis, est_method = "5par",
                        transform_to_marginal_normality = FALSE)
lg_object_c  <- lg_main(crisis, est_method = "5par",
                        transform_to_marginal_normality = FALSE)

## Not run:
# Run the test (with very few resamples for illustration)
test_result <- cont_test(lg_object_nc, lg_object_c,
                         n_rep = 10)

## End(Not run)
```

corplot	*Plot local correlation maps*

Description

Plot the estimated local correlation map (or local *partial* correlation map) for a pair of variables

Usage

```
corplot(dlg_object, pair = 1, gaussian_scale = FALSE,
  plot_colormap = TRUE, plot_obs = FALSE, plot_labels = TRUE,
  plot_legend = FALSE, plot_thres = 0, alpha_tile = 0.8,
  alpha_point = 0.8, low_color = "blue", high_color = "red",
  break_int = 0.2, label_size = 3, font_family = "sans",
  point_size = NULL, xlim = NULL, ylim = NULL, xlab = NULL,
  ylab = NULL, rholab = NULL, main = NULL, subtitle = NULL)
```

Arguments

`dlg_object`	The density estimation object produced by the dlg-function
`pair`	Integer indicating which pair of variables you want to plot. The function looks up the corresponding variables in the bandwidth object used to calculate the dlg object, and you can inspect this in `dlg_object$bw$joint`. Defaults to 1 (the first pair, usually variable 1 against variable 2).
`gaussian_scale`	Logical, if `TRUE` the plot is produced on the marginal standard Gaussian scale.
`plot_colormap`	Logical, if `TRUE` the plot includes a colormap to visualize the value of the local correlation.
`plot_obs`	Logical, if `TRUE` the observations are plotted.
`plot_labels`	Logical, if `TRUE` character labels with local correlation values are plotted.
`plot_legend`	Logical, if `TRUE` a color legend is plotted.
`plot_thres`	A number between 0 and 1 indicating the threshold value to be used for not plotting the estimated local correlation in areas with no data. Uses a quick bivariate kernel density estimate a criterion, and skips plotting in areas with kernel density estimate less than the fraction plot_thres of the maximum density estimate. If 0 (default), everything is plotted, if 1 nothing is plotted. Typical values may be in the 0.001-0.01-range.
`alpha_tile`	The alpha-value indicating the transparency of the color tiles. Number between 0 (transparent) and 1 (not transparent).
`alpha_point`	he alpha-value indicating the transparency of the observations. Number between 0 (transparent) and 1 (not transparent).
`low_color`	The color corresponding to correlation equal to -1 (default: blue).
`high_color`	The color corresponding to correlation equal to 1 (default: red).
`break_int`	Break interval in the color gradient.

label_size	Size of text labels, if plotted.
font_family	Font family used for text labels, if plotted.
point_size	Size of points used for plotting the observations.
xlim	x-limits
ylim	y-limits
xlab	x-label
ylab	y-label
rholab	Label for the legend, if plotted
main	Title of plot
subtitle	Subtitle of plot

Details

This function plots a map of estimated local Gaussian correlations of a specified pair (defaults to the first pair) of variables as produced by the dlg-function. This plot is heavily inspired by the local correlation plots produced by the 'localgauss'-package by Berentsen et. al (2014), but it is here more easily customized and specially adapted to the ecosystem within the `lg`-package. The plotting is carried out using the ggplot2-package (Wickham, 2009). This function now also accepts objects created by the `partial_cor()`-function, in order to create local *partial* correlation maps.

References

Berentsen, G. D., Kleppe, T. S., & Tjøstheim, D. (2014). Introducing localgauss, an R package for estimating and visualizing local Gaussian correlation. Journal of Statistical Software, 56(1), 1-18.

H. Wickham. ggplot2: Elegant Graphics for Data Analysis. Springer-Verlag New York, 2009.

---

dlg — *The locally Gaussian density estimator (LGDE)*

Description

Estimate a multivariate density function using locally Gaussian approximations

Usage

```
dlg(lg_object, grid, level = 0.95, normalization_points = NULL,
  bootstrap = F, B = 500)
```

Arguments

- `lg_object` An object of type `lg`, as produced by the `lg_main`-function.
- `grid` A matrix of grid points, where we want to evaluate the density estimate.
- `level` Specify a level if asymptotic standard deviations and confidence intervals should be returned.
- `normalization_points` How many grid points for approximating the integral of the density estimate, to use for normalization?
- `bootstrap` Calculate bootstrapped confidence intervals instead.
- `B` Number of bootstrap replications if using bootstrapped confidence intervals.

Details

This function does multivariate density estimation using the locally Gaussian density estimator (LGDE), that was introduced by Otneim & Tjøstheim (2017). The function takes as arguments an `lg`-object as produced by the main `lg_main`-function (where all the running parameters are specified), and a grid of points where the density estimate should be estimated.

Value

A list containing the density estimate as well as all the running parameters that has been used. The elements are:

- `f_est`: The estimated multivariate density.
- `loc_mean`: The estimated local means if `est_method` is "5par" or "5par_marginals_fixed", a matrix of zeros if `est_method` is "1par".
- `loc_sd`: The estimated local st. deviations if `est_method` is "5par" or "5par_marginals_fixed", a matrix of ones if `est_method` is "1par".
- `loc_cor`: Matrix of estimated local correlations, one column for each pair of variables, in the same order as specified in the bandwidth object.
- `x`: The data set.
- `bw`: The bandwidth object.
- `transformed_data`: The data transformed to approximate marginal standard normality.
- `normalizing_constants`: The normalizing constants used to transform data and grid back and forth to the marginal standard normality scale, as seen in eq. (8) of Otneim & Tjøstheim (2017).
- `grid`: The grid where the estimation was performed, on the original scale.
- `transformed_grid`: The grid where the estimation was performed, on the marginal standard normal scale.
- `normalization_points` Number of grid points used to approximate the integral of the density estimate, in order to normalize?
- `normalization_constant` If approximated, the integral of the non-normalized density estimate. NA if not normalized.

- `density_normalized` Logical, indicates whether the final density estimate (contained in f_est) has been approximately normalized to have unit integral.
- `loc_cor_sd` Estimated asymptotic standard deviation for the local correlations.
- `loc_cor_lower` Lower confidence limit based on the asymptotic standard deviation.
- `loc_cor_upper` Upper confidence limit based on the asymptotic standard deviation.

References

Otneim, Håkon, and Dag Tjøstheim. "The locally gaussian density estimator for multivariate data." Statistics and Computing 27, no. 6 (2017): 1595-1616.

Examples

```
x <- cbind(rnorm(100), rnorm(100), rnorm(100))
lg_object <- lg_main(x)  # Put all the running parameters in here.
grid <- cbind(seq(-4, 4, 1), seq(-4, 4, 1), seq(-4, 4, 1))
density_estimate <- dlg(lg_object, grid = grid)
```

---

dlg_bivariate *Bivariate density estimation*

Description

`dlg_bivariate` returns the locally Gaussian density estimate of a bivariate distribution on a given grid.

Usage

```
dlg_bivariate(x, eval_points = NA, grid_size = 15, bw = c(1, 1),
  est_method = "1par", tol = .Machine$double.eps^0.25/10^4,
  run_checks = TRUE, marginal_estimates = NA, bw_marginal = NA)
```

Arguments

`x`	The data matrix (or data frame). Must have exactly 2 columns.
`eval_points`	The grid where the density should be estimated. Must have exactly 2 columns.
`grid_size`	If `eval_points` is not supplied, then the function will create a suitable grid diagonally through the data, with this many grid points.
`bw`	The two bandwidths, a numeric vector of length 2.
`est_method`	The estimation method, must either be "1par" for estimation with just the local correlation, or "5par" for a full locally Gaussian fit with all 5 parameters.
`tol`	The numerical tolerance to be used in the optimization. Only applicable in the 1-parameter optimization.

run_checks — Logical. Should sanity checks be run on the arguments? Useful to disable this when doing cross-validation for example.

marginal_estimates — Provide the marginal estimates here if estimation method is "5par_marginals_fixed", and the marginal estimates have already been found. Useful for cross-validation. List with two elements as returned by dlg_marginal_wrapper.

bw_marginal — Vector of bandwidths used to estimate the marginal distributions.

Details

This function serves as the backbone in the body of methods concerning local Gaussian correlation. It takes a bivariate data set, x, and a bivariate set of grid points eval_points, and returns the bivariate, locally Gaussian density estimate in these points. We also need a vector of bandwidths, bw, with two elements, and an estimation method est_method

Value

A list including the data set $x, the grid $eval_points, the bandwidths $bw, as well as a matrix of the estimated parameter estimates $par_est and the estimated bivariate density $f_est.

Examples

```
x <- cbind(rnorm(100), rnorm(100))
bw <- c(1, 1)
eval_points <- cbind(seq(-4, 4, 1), seq(-4, 4, 1))

estimate <- dlg_bivariate(x, eval_points = eval_points, bw = bw)
```

---

dlg_marginal — *Marginal density estimation*

Description

Function that estimates a univariate density estimation by local Gaussian approximations, as described in Hufthammer and Tjøstheim (2009).

Usage

```
dlg_marginal(x, bw = 1, eval_points = seq(quantile(x, 0.01),
  quantile(x, 0.99), length.out = grid_size), grid_size = 15)
```

Arguments

x — The data vector.

bw — The bandwidth (a single number).

eval_points — The grid where we want to evaluate the density. Chosen suitably if not provided, with length equal to grid_size.

grid_size — Number of grid points if grid is not provided.

Details

This function is mainly mean to be used as a tool in multivariate analysis as away to obtain the estimate of a univariate (marginal) density function, but it can of course be used in general to estimate univariate densities.

Value

A list including the data set `$x`, the grid `$eval_points`, the bandwidth `$bw`, as well as a matrix of the estimated parameter estimates `$par_est` and the estimated bivariate density `$f_est`.

References

Hufthammer, Karl Ove, and Dag Tjøstheim. "Local Gaussian Likelihood and Local Gaussian Correlation" PhD Thesis of Karl Ove Hufthammer, University of Bergen, 2009.

Examples

```
x <- rnorm(100)
estimate <- dlg_marginal(x, bw = 1, eval_points = -4:4)
```

`dlg_marginal_wrapper` *Marginal estimates for multivariate data*

Description

Estimates the marginal locally Gaussian parameters for a multivariate data set

Usage

```
dlg_marginal_wrapper(data_matrix, eval_matrix, bw_vector)
```

Arguments

`data_matrix`	The matrix of data points. One column constitutes an observation vector.
`eval_matrix`	The matrix of evaluation points. One column constitutes a vector of grid points.
`bw_vector`	The vector of bandwidths, one element per component.

Details

This function takes in a matrix of observations, a matrix of evaluation points and a vector of bandwidths, and does a locally Gaussian fit on each of the marginals using the `dlg_bivariate`-function. This function assumes that the data and evaluation points are organized column-wise in matrices, and that the bandwidth is found in the corresponding element in the bandwidth matrix. The primary use for this function is multivariate density estimation using the "5par_marginals_fixed"-method.

Value

A list with marginal parameter and density estimates as provided by the dlg_bivariate-function. One element per column in the data.

Examples

```
data_matrix <- cbind(rnorm(100), rnorm(100))
eval_matrix <- cbind(seq(-4, 4, 1), seq(-4, 4, 1))
bw <- c(1, 1)

estimate <- dlg_marginal_wrapper(data_matrix, eval_matrix = eval_matrix, bw = bw)
```

---

dlg_trivariate *Trivariate density estimation*

Description

dlg_trivariate returns the locally Gaussian density estimate of a trivariate distribution on a given grid.

Usage

```
dlg_trivariate(x, eval_points = NULL, grid_size = 15, bw = c(1, 1,
  1), est_method = "trivariate", run_checks = TRUE)
```

Arguments

x	The data matrix (or data frame). Must have exactly 2 columns.
eval_points	The grid where the density should be estimated. Must have exactly 2 columns.
grid_size	If eval_points is not supplied, then the function will create a suitable grid diagonally through the data, with this many grid points.
bw	The two bandwidths, a numeric vector of length 2.
est_method	The estimation method, must either be "1par" for estimation with just the local correlation, or "5par" for a full locally Gaussian fit with all 5 parameters.
run_checks	Logical. Should sanity checks be run on the arguments? Useful to disable this when doing cross-validation for example.

Details

In some applications it may be desired to produce a full locally Gaussian fit of a trivariate density function without having to resort to bivariate approximations. This function takes a trivariate data set, x, and a trivariate set of grid points eval_points, and returns the trivariate, locally Gaussian density estimate in these points. We also need a vector of bandwidths, bw, with three elements, and an estimation method est_method, which in this case is fixed at "trivariate", and included only to be fully compatible with the other methods in this package.

This function will only work on the marginally standard normal scale! Please use the wrapper function dlg() for density estimation. This will ensure that all parameters have proper values.

Value

A list including the data set $x, the grid $eval_points, the bandwidths $bw, as well as a matrix of the estimated parameter estimates $par_est and the estimated bivariate density $f_est.

Examples

```
x <- cbind(rnorm(100), rnorm(100), rnorm(100))
bw <- c(1, 1, 1)
eval_points <- cbind(seq(-4, 4, 1), seq(-4, 4, 1), seq(-4, 4, 1))

estimate <- dlg_trivariate(x, eval_points = eval_points, bw = bw)
```

---

dmvnorm_wrapper *Wrapper for* dmvnorm

Description

dmvnorm_wrapper is a function that evaluates the bivariate normal distribution in a matrix of evaluation points, with local parameters.

Usage

```
dmvnorm_wrapper(eval_points, mu_1 = rep(0, nrow(eval_points)),
  mu_2 = rep(0, nrow(eval_points)), sig_1 = rep(1, nrow(eval_points)),
  sig_2 = rep(1, nrow(eval_points)), rho = rep(0, nrow(eval_points)),
  run_checks = TRUE)
```

Arguments

eval_points	A kx2 matrix with evaluation points
mu_1	The first expectation vector
mu_2	The second expectation vector
sig_1	The first standard deviation vector
sig_2	The second standard deviation vector
rho	The correlation vector
run_checks	Run sanity check for the arguments

Details

This functions takes as arguments a matrix of grid points, and vectors of parameter values, and returns the bivariate normal density at these points, with these parameter values.

dmvnorm_wrapper_single *Wrapper for* `dmvnorm` *- single point*

Description

Function that evaluates the bivariate normal in a single point

Usage

```
dmvnorm_wrapper_single(x1, x2, mu_1, mu_2, sig_1, sig_2, rho)
```

Arguments

x1	The first component of the evaluation point
x2	The second component of the evaluation point
mu_1	The first expectation
mu_2	The second expectation
sig_1	The first standard deviation
sig_2	The second standard deviation
rho	The correlation

gradient *Auxiliary function for calculating the asymptotic standard deviations for the local Gaussian correlations*

Description

Auxiliary function for calculating the asymptotic standard deviations for the local Gaussian correlations

Usage

```
gradient(sigma, sigma_k)
```

Arguments

sigma	sigma
sigma_k	sigma_k

ind_test *Independence tests*

Description

Independence tests based on the local Gaussian correlation

Usage

```
ind_test(lg_object, h = function(x) x^2, S = function(y)
  as.logical(rep(1, nrow(y))), bootstrap_type = "plain",
  block_length = NULL, n_rep = 1000)
```

Arguments

lg_object	An object of type lg, as produced by the lg_main-function. The data must be two dimensional.
h	The h-function used in the calculation of the test statistic. The default value is h(x) = x^2.
S	The integration area for the test statistic. Must be a logical function that accepts an n x 2 matrix and returns TRUE if a row is in S.
bootstrap_type	The bootstrap method. Choose "plain" for the ordinary nonparametric bootstrap valid for independence test for iid data and for serial dependence within a time series. Choose "stationary" or "block" for a test for cross dependence between two time series.
block_length	Block length if using block bootstrap for the cross dependence test. Calculated by np::b.star() if not supplied.
n_rep	Number of bootstrap replications.

Details

Implementation of three independence tests: For iid data (Berentsen et al., 2014), for serial dependence within a time series (Lacal and Tjøstheim, 2017a), and for serial cross-dependence between two time series (Lacal and Tjøstheim, 2017b). The first test has a different theoretical foundation than the latter two, but the implementations are similar and differ only in the bootstrap procedure. For the time series applications, the user must lag the series to his/her convenience before making the lg_object and calling this function.

Value

A list containing the test result as well as various parameters. The elements are:

- lg_object The lg-object supplied by the user.
- observed The observed value of the test statistic.
- replicated The replicated values of the test statistic.

- `bootstrap_type` The bootstrap type.
- `block_length` The block length used for the block bootstrap.
- `p_value` The p-value of the test.

References

Berentsen, Geir Drage, and Dag Tjøstheim. "Recognizing and visualizing departures from independence in bivariate data using local Gaussian correlation." Statistics and Computing 24.5 (2014): 785-801.

Lacal, Virginia, and Dag Tjøstheim. "Local Gaussian autocorrelation and tests for serial independence." Journal of Time Series Analysis 38.1 (2017a): 51-71.

Lacal, Virginia, and Dag Tjøstheim. "Estimating and testing nonlinear local dependence between two time series." Journal of Business & Economic Statistics just-accepted (2017b).

Examples

```
    # Remember to increase the number of bootstrap samplesin preactical
    # implementations.

    ## Not run:

    # Test for independence between two vectors, iid data.
    x1 <- cbind(rnorm(100), rnorm(100))
    lg_object1 <- lg_main(x1)
    test_result1 = ind_test(lg_object1,
                            bootstrap_type = "plain",
                            n_rep = 20)

    # Test for serial dependence in time series, lag 1
    data(EuStockMarkets)
    logreturns <- apply(EuStockMarkets, 2, function(x) diff(log(x)))
    x2 <- cbind(logreturns[1:100,1], logreturns[2:101, 1])
    lg_object2 <- lg_main(x2)
    test_result2 = ind_test(lg_object2,
                            bootstrap_type = "plain",
                            n_rep = 20)

    # Test for cross-dependence, lag 1
    x3 <- cbind(logreturns[1:100,1], logreturns[2:101, 2])
    lg_object3 <- lg_main(x3)
    test_result3 = ind_test(lg_object3,
                            bootstrap_type = "block",
                            n_rep = 20)

## End(Not run)
```

ind_teststat — *Function that calculates the test statistic in the independence tests.*

Description

This is an auxiliary function used by the independence tests.

Usage

```
ind_teststat(x_replicated, lg_object, S, h)
```

Arguments

`x_replicated`	A sample.
`lg_object`	An lg-object.
`S`	Integration area, see `?ind_test`.
`h`	h-function for test statistic, see `?ind_test`.

interpolate_conditional_density — *Interpolate a univariate conditional density function*

Description

Estimates the conditional density function for one free variable on a grid. Returns a function that interpolates between these grid points so that it can be evaluated more quickly, without new optimizations.

Usage

```
interpolate_conditional_density(lg_object, condition, nodes,
  extend = 0.3,
  gaussian_scale = lg_object$transform_to_marginal_normality)
```

Arguments

`lg_object`	An object of type `lg`, as produced by the `lg_main`-function
`condition`	A vector with conditions for the variables that we condition upon. Must have exactly one more element than there are columns in the data
`nodes`	Either the number of equidistant nodes to generate, or a vector of nodes supplied by the user
`extend`	How far to extend the grid beyond the extreme data points, in share of the range
`gaussian_scale`	Stay on the standard Gaussian scale, useful for the accept-reject algorithm

`lg` — `lg`: *A package for calculating the local Gaussian correlation in multivariate applications.*

Description

The `lg` package provides implementations for the multivariate density estimation and the conditional density estimation methods using local Gaussian correlation as presented in Otneim & Tjøstheim (2017) and Otneim & Tjøstheim (2018).

Details

The main function is called `lg_main`, and takes as argument a data set (represented by a matrix or data frame) as well as various (optional) configurations that is described in detail in the articles mentioned above, and in the documentation of this package. In particular, this function will calculate the bandwidths used for estimation, using either a plugin estimate (default), or a cross validation estimate. If `x` is the data set, then the following line of code will create an `lg` object using the default configuration, that can be used for density estimation afterwards:

`lg_object <- lg_main(x)`

You can change estimation method, bandwidth selection method and other parameters by using the arguments of the `lg_main` function.

You can evaluate the multivariate density estimate on a `grid` as described in Otneim & Tjøstheim (2017) using the `dlg`-function as follows:

`dens_est <- dlg(lg_object, grid = grid)`.

Assuming that the data set has **p** variables, you can evaluate the *conditional* density of the **p - q** first variables (counting from column 1), *given* the remaining **q** variables being equal to `condition = c(v1, ..., vq)`, on a `grid`, by running

`conditional_dens_est <- clg(lg_object, grid = grid, condition = condition)`.

References

Otneim, Håkon, and Dag Tjøstheim. "The locally gaussian density estimator for multivariate data." Statistics and Computing 27, no. 6 (2017): 1595-1616.

Otneim, Håkon, and Dag Tjøstheim. "Conditional density estimation using the local Gaussian correlation" Statistics and Computing 28, no. 2 (2018): 303-321.

lg_main *Create an* lg *object*

Description

Create an lg-object, that can be used to estimate local Gaussian correlations, unconditional and conditional densities, local partial correlation and for testing purposes.

Usage

```
lg_main(x, bw_method = "plugin", est_method = "1par",
  transform_to_marginal_normality = TRUE, bw = NULL,
  plugin_constant_marginal = 1.75, plugin_constant_joint = 1.75,
  plugin_exponent_marginal = -1/5, plugin_exponent_joint = -1/6,
  tol_marginal = 10^(-3), tol_joint = 10^(-3))
```

Arguments

- `x` — A matrix or data frame with data, on column per variable, one row per observation.
- `bw_method` — The method used for bandwidth selection. Must be either "cv" (cross-validation, slow, but accurate) or "plugin" (fast, but crude).
- `est_method` — The estimation method, must be either "1par", "5par", "5par_marginals_fixed" or "trivariate". (see details).
- `transform_to_marginal_normality` — Logical, TRUE if we want to transform our data to marginal standard normality. This is assumed by method "1par", but can of course be skipped using this argument if it has been done already.
- `bw` — Bandwidth object if it has already been calculated.
- `plugin_constant_marginal` — The constant c in cn^a used for finding the plugin bandwidth for locally Gaussian marginal density estimates, which we need if estimation method is "5par_marginals_fixed".
- `plugin_constant_joint` — The constant c in cn^a used for finding the plugin bandwidth for estimating the pairwise local Gaussian correlation between two variables.
- `plugin_exponent_marginal` — The constant a in cn^a used for finding the plugin bandwidth for locally Gaussian marginal density estimates, which we need if estimation method is "5par_marginals_fixed".
- `plugin_exponent_joint` — The constant a in cn^a used for finding the plugin bandwidth for estimating the pairwise local Gaussian correlation between two variables.
- `tol_marginal` — The absolute tolerance in the optimization for finding the marginal bandwidths, passed on to the optim-function.
- `tol_joint` — The absolute tolerance in the optimization for finding the joint bandwidths. Passed on to the optim-function.

Details

This is the main function in the package. It lets the user supply a data set and set a number of options, which is then used to prepare an `lg` object that can be supplied to other functions in the package, such as `dlg` (density estimation), `clg` (conditional density estimation). The details has been laid out in Otneim & Tjøstheim (2017) and Otneim & Tjøstheim (2018).

The papers mentioned above deal with the estimation of multivariate density functions and conditional density functions. The idea is to fit a multivariate Normal locally to the unknown density function by first transforming the data to marginal standard normality, and then estimate the local correlations **pairwise**. The local means and local standard deviations are held fixed and constantly equal to 0 and 1 respectively to reflect the knowledge that the marginals are approximately standard normal. Use `est_method = "1par"` for this strategy, which means that we only estimate one local parameter (the correlation) for each pair, and note that this method requires marginally standard normal data. If `est_method = "1par"` and `transform_to_marginal_normality = FALSE` the function will throw a warning. It might be okay though, if you know that the data are marginally standard normal already.

The second option is `est_method = "5par_marginals_fixed"` which is more flexible than `"1par"`. This method will estimate univariate local Gaussian fits to each marginal, thus producing local estimates of the local means: $\mu_i(x_i)$ and $\sigma_i(x_i)$ that will be held fixed in the next step when the **pairwise** local correlations are estimated. This method can in many situations provide a better fit, even if the marginals are standard normal. It also opens up for creating a multivariate locally Gaussian fit to any density without having to transform the marginals if you for some reason want to avoid that.

The third option is `est_method = "5par"`, which is a full nonparametric locally Gaussian fit of a bivariate density as laid out and used by Tjøstheim & Hufthammer (2013) and others. This is simply a wrapper for the `localgauss`-package by Berentsen et.al. (2014).

A recent option is described by Otneim and Tjøstheim (2019), who allow a full trivariate fit to a three dimensional data set that is transformed to marginal standard normality in the context of their test for conditional independence (see `?ci_test` for details), but this can of course be used as an option to estimate three-variate density functions as well.

References

Berentsen, Geir Drage, Tore Selland Kleppe, and Dag Tjøstheim. "Introducing localgauss, an R package for estimating and visualizing local Gaussian correlation." Journal of Statistical Software 56.1 (2014): 1-18.

Hufthammer, Karl Ove, and Dag Tjøstheim. "Local Gaussian Likelihood and Local Gaussian Correlation" PhD Thesis of Karl Ove Hufthammer, University of Bergen, 2009.

Otneim, Håkon, and Dag Tjøstheim. "The locally gaussian density estimator for multivariate data." Statistics and Computing 27, no. 6 (2017): 1595-1616.

Otneim, Håkon, and Dag Tjøstheim. "Conditional density estimation using the local Gaussian correlation" Statistics and Computing 28, no. 2 (2018): 303-321.

Otneim, Håkon, and Dag Tjøstheim. "The local Gaussian partial correlation" Working paper (2019).

Tjøstheim, D., & Hufthammer, K. O. (2013). Local Gaussian correlation: a new measure of dependence. Journal of Econometrics, 172(1), 33-48.

Examples

```
x <- cbind(rnorm(100), rnorm(100), rnorm(100))

# Quick example
lg_object1 <- lg_main(x, bw_method = "plugin", est_method = "1par")

# In the simulation experiments in Otneim & Tjøstheim (2017a),
# the cross-validation bandwidth selection is used:
## Not run:
lg_object2 <- lg_main(x, bw_method = "cv", est_method = "1par")

## End(Not run)

# If you do not wish to transform the data to standard normality,
# use the five parameter fit:
lg_object3 <- lg_main(x, est_method = "5par_marginals_fixed",
                transform_to_marginal_normality = FALSE)

# In the bivariate case, you can use the full nonparametric fit:
x_biv <- cbind(rnorm(100), rnorm(100))
lg_object4 <- lg_main(x_biv, est_method = "5par",
                transform_to_marginal_normality = FALSE)

# Whichever method you choose, the lg-object can now be passed on
# to the dlg- or clg-functions for evaluation of the density or
# conditional density estimate. Control the grid with the grid
# argument.
grid1 <- x[1:10,]
dens_est <- dlg(lg_object1, grid = grid1)

# The conditional density of X1 given X2 = 1 and X2 = 0:
grid2 <- matrix(-3:3, ncol = 1)
c_dens_est <- clg(lg_object1, grid = grid2, condition = c(1, 0))
```

`local_conditional_covariance` *Calculate the local conditional covariance between two variables*

Description

Wrapper for the `clg` function that extracts the local Gaussian conditional covariance between two variables from an object that is produced by the clg-function.

Usage

```
local_conditional_covariance(clg_object, coord = c(1, 2))
```

Arguments

clg_object	The object produced by the clg-function
coord	The variables for which the conditional covariance should be extracted

Details

This function is a wrapper for the clag-function, and extracts the estimated local conditional covariance between the first two variables in the data matrix, on the grid specified to the clg-function.

---

make_C — *Auxiliary function for calculating the asymptotic standard deviations for the local Gaussian correlations*

Description

Auxiliary function for calculating the asymptotic standard deviations for the local Gaussian correlations

Usage

```
make_C(r, pairs, p)
```

Arguments

r	r
pairs	pairs
p	p

---

mvnorm_eval — *Evaluate the multivariate normal*

Description

Function that evaluates the multivariate normal distribution with local parameters

Usage

```
mvnorm_eval(eval_points, loc_mean, loc_sd, loc_cor, pairs)
```

Arguments

`eval_points`	A matrix of grid points
`loc_mean`	A matrix of local means, one row per grid point, one column per component
`loc_sd`	A matrix of local standard deviations, one row per grid point, one column per component
`loc_cor`	A matrix of local correlations, one row per grid point, on column per pair of variables
`pairs`	A data frame specifying the components that make up each pair,

Details

Takes in a grid, where we want to evaluate the multivariate normal, and in each grid point we have a new set of parameters.

`partial_cor` *Calculate the local Gaussian partial correlation*

Description

A function that calculates the local Gaussian partial correlation for a pair of variables, given the values of some conditioning variables.

Usage

```
partial_cor(lg_object, grid = NULL, condition = NULL, level = NULL)
```

Arguments

`lg_object`	An object of type `lg`, as produced by the `lg_main`-function.
`grid`	A matrix of grid points, where we want to evaluate the density estimate. Number of columns *must* be equal to 2.
`condition`	A vector with conditions for the variables that we condition upon. Length of this vector *must* be the same as the number of variables in X3. The function will throw an error of there is any discrepancy in the dimensions of the `grid`, `condition` and data set.
`level`	Specify a level if asymptotic standard deviations and confidence intervals should be returned. If not, set to `NULL`.

Details

This function is a wrapper for the clg-function (for conditional density estimation) that returns the local conditional, or partial, correlations described by Otneim & Tjøstheim (2018). The function takes as arguments an lg-object as produced by the main lg_main- function, a grid of points where the density estimate should be estimated, and a set of conditions.

The variables must be sorted before they are supplied to this function. It will always assume that the free variables come before the conditioning variables, see ?clg for details.

Assume that X is a stochastic vector with scalar components X1 and X2, and a possibly d-dimensional component X3. This function will thus compute the local *partial* correlation between X1 and X2 given X3 = x3.

Value

A list containing the local partial Gaussian correlations as well as all the running parameters that has been used. The elements are:

- grid The grid where the estimation was performed, on the original scale.
- partial_correlations The estimated local partial Gaussian correlations.
- cond_density The estimated conditional density of X1 and X2 given X3, as described by Otneim & Tjøstheim (2018).
- transformed_grid: The grid where the estimation was performed, on the marginal standard normal scale.
- bw: The bandwidth object.
- partial_correlations_sd Estimated standard deviations of the local partial Gaussian correlations, as described in a forthcoming paper.
- partial_correlations_lower Lower confidence limit based on the asymptotic standard deviation.
- partial_correlations_upper Upper confidence limit based on the asymptotic standard deviation.

References

Otneim, Håkon, and Dag Tjøstheim. "Conditional density estimation using the local Gaussian correlation" Statistics and Computing 28, no. 2 (2018): 303-321.

Examples

```
# A 3 variate example
x <- cbind(rnorm(100), rnorm(100), rnorm(100))

# Generate the lg-object with default settings
lg_object <- lg_main(x)

# Estimate the local partial Gaussian correlation between X1 and X2 given X3 = 1 on
# a small grid
partial_correlations <- partial_cor(lg_object,
                                    grid = cbind(-4:4, -4:4),
```

```
condition = 1)
```

`replicate_under_ci` *Bootstrap replication under the null hypothesis*

Description

Generate bootstrap replicates under the null hypothesis that the first two variables are conditionally independent given the rest of the variables.

Usage

```
replicate_under_ci(lg_object, n_rep, nodes, M = NULL, M_sim = 1500,
  M_corr = 1.5, n_corr = 1.2, extend = 0.3)
```

Arguments

`lg_object`	An object of type `lg`, as produced by the `lg_main`-function
`n_rep`	The number of replicated bootstrap samples
`nodes`	Either the number of equidistant nodes to generate, or a vector of nodes supplied by the user
`M`	The value for M in the accept-reject algorithm if already known
`M_sim`	The number of replicates to simulate in order to find a value for M
`M_corr`	Correction factor for M, to be on the safe side
`n_corr`	Correction factor for n_new, so that we mostly will generate enough observations in the first go
`extend`	How far to extend the grid beyond the extreme data points when interpolating, in share of the range

---

`trans_normal` *Transform the marginals of a multivariate data set to standard normality based on the logspline density estimator (Kooperberg and Stone, 1991). See Otneim and Tjøstheim (2017) for details.*

Description

Transform the marginals of a multivariate data set to standard normality based on the logspline density estimator (Kooperberg and Stone, 1991). See Otneim and Tjøstheim (2017) for details.

Usage

```
trans_normal(x)
```

Arguments

`x`	The data matrix, one row per observation.

Value

A list containing the transformed data ($transformed_data), and a function ($trans_new) that can be used to transform grid points and obtain normalizing constants for use in density estimation functions

References

Kooperberg, Charles, and Charles J. Stone. "A study of logspline density estimation." Computational Statistics & Data Analysis 12.3 (1991): 327-347.

Otneim, Håkon, and Dag Tjøstheim. "The locally gaussian density estimator for multivariate data." Statistics and Computing 27, no. 6 (2017): 1595-1616.

`u` *Auxiliary function for calculating the local score function u*

Description

Auxiliary function for calculating the local score function u

Usage

```
u(z1, z2, rho)
```

Arguments

`z1`	z1
`z2`	z2
`rho`	rho

Details

This function is used to estimate the asymptotic variance of the estimates.

Index