lg: An R package for Local Gaussian Approximations

by Håkon Otneim

Abstract The package lg for the R programming language provides implementations of recent methodological advances on applications of the local Gaussian correlation. This includes estimation of the local Gaussian correlation itself, multivariate density estimation, conditional density estimation, various tests for independence and conditional independence, as well as a graphical module for creating dependence maps. This paper describes the lg package, its principles, and its practical use.

Introduction

Tjøstheim and Hufthammer (2013) propose the *local Gaussian correlation* (LGC) as a new measure of statistical dependence between two stochastic variables X_1 and X_2, which has the following important property yet unrivaled in the literature: It can separate between positive and negative nonlinear dependence while still reducing to the ordinary Pearson correlation coefficient if X_1 and X_2 are jointly normally distributed. The R-package **localgauss** (Berentsen et al., 2014) provides two important functions in this context; one that calculates the sample LGC based on observed values of (X_1, X_2), and one that uses the estimated LGC to perform a local test of independence between X_1 and X_2 as described in detail by Berentsen and Tjøstheim (2014).

We have lately seen a number of new applications of the LGC that the **localgauss**-package does not support, however. Støve et al. (2014) use the LGC to test for financial contagion across markets during crises. Otneim and Tjøstheim (2017) present a procedure for estimating multivariate density functions via the LGC, which Otneim and Tjøstheim (2018) modify in order to compute estimates of conditional density functions. Lacal and Tjøstheim (2017) present a test for serial independence within a time series, which Lacal and Tjøstheim (2018) extend in order to include a test for cross-correlation between two time series. Finally, Otneim and Tjøstheim (2021) develop the local Gaussian *partial* correlation (LGPC) as a measure of conditional dependence, and a corresponding test for conditional independence.

This paper describes the **lg**-package (Otneim, 2019), which provides a unified framework to implement all these methods, as well as a tool for visualizing the LGC and LGPC as dependence maps. Jordanger and Tjøstheim (2020) use the LGC in spectral analysis of time series, but those methods have their own computational ecosystem in the **localgaussSpec**-package, Jordanger (2018).

In Section 2.2 we provide a brief introduction to the LGC as well as the methods and applications referred to above. In Section 2.3 we describe the core function in the **lg**-package and move on to demonstrate the implementation of various applications in Section 2.4. We conclude this paper in Section 2.5 by demonstrating the graphical capabilities of the **lg**-package.

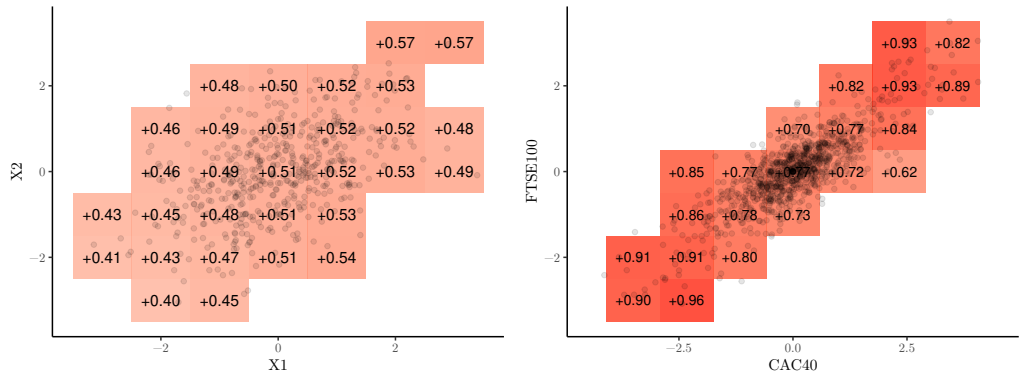
Statistical background

Consider a random vector X having the unknown probability density function $f_X(x)$. It is a standard task to estimate f_X based on a random sample $X_1, \ldots, X_n$, and the statistical literature provides an abundance of methods to accomplish this. One may, for example, make the assumption that the unknown density function has a particular parametric form, $f_X \in F_\theta$, where $F_\theta = \{f(x;\theta), \theta \in \Theta\}$ is a family of probability density functions indexed by some parameter θ, and where Θ is the parameter space. Under this assumption we will typically produce an estimate of the parameter θ, written $\widehat{\theta}$, using the maximum likelihood method. The estimated probability density function is then given as $\widehat{f}_X(x) = f\left(x;\widehat{\theta}\right)$.

A different approach is to estimate $f_X(\cdot)$ without any prior parametric assumptions. The classical method for non-parametric density estimation is the kernel estimator

$$\widehat{f}_X(x) = \frac{1}{nb}\sum_{i=1}^{n} K\left(\frac{X_i - x}{b}\right),$$

where K is a symmetric density function (the kernel) and b is a tuning parameter (the bandwidth) that controls the smoothness of the estimate $\widehat{f}_X(\cdot)$. See Silverman (1986) for an introduction to this topic. There is also a massive statistical literature on density estimation containing extensions and



(a) Estimated local Gaussian correlations between two jointly normal variables having correlation equal to 0.5, based on 1000 observations.

(b) Local Gaussian correlation between daily returns from the CAC40 and FTSE100 stock indices, based on 1000 consecutive trading days.

Figure 1: Two dependence maps

improvements to the classical methods to be used in various practical situations.

Hjort and Jones (1996) provide one such idea. They consider a parametric family F_θ, but instead of searching for a single parameter value θ_0, for which $f_X(x) = f(x;\theta_0)$ (or approximately so), they rather assert that different members of F_θ may approximate f_X locally in different parts of its domain. In other words, they seek to estimate a parameter *function* $\theta_0(x)$ for which $f_X(x) = f(x;\theta_0(x))$ (or approximately so), and do this by maximizing a *local likelihood function* in each point x:

$$\begin{aligned}\widehat{\theta}(x) &= \arg\max_{\theta\in\Theta} L_n(\theta, x) \\ &= \arg\max_{\theta\in\Theta} \frac{1}{nb}\sum_{i=1}^{n} K\left(\frac{X_i - x}{b}\right)\log f(X_i;\theta) - \int \frac{1}{b}K\left(\frac{y-x}{b}\right) f(y;\theta)\,\mathrm{d}y, \end{aligned} \tag{1}$$

where, again, K is a symmetric density function and b is a bandwidth parameter that controls the smoothness of the estimate. The second term in the local likelihood function is a penalty that ensures that the estimated density $\widehat{f}_X(x) = f\left(x;\widehat{\theta}(x)\right)$ converges correctly to the true density function $f_X(x)$ as the sample size n increases to infinity, and the bandwidth b decreases towards zero. See Hjort and Jones (1996) for a detailed discussion about this construction.

Tjøstheim and Hufthammer (2013) consider the bivariate case $X = (X_1, X_2)$ and take F_θ to be the family of bivariate normal distributions consisting of densities on the form

$$\begin{aligned} f(x;\theta) &= \psi(x_1, x_2; \mu_1, \mu_2, \sigma_1, \sigma_2, \rho) \\ &= \frac{1}{2\pi\sigma_1\sigma_2\sqrt{1-\rho^2}} \\ &\quad \times \exp\left\{-\frac{1}{2(1-\rho^2)}\left(\frac{(x_1-\mu_1)^2}{\sigma_1^2} - 2\rho\frac{(x_1-\mu_1)(x_2-\mu_2)}{\sigma_1\sigma_2} + \frac{(x_2-\mu_2)^2}{\sigma_2^2}\right)\right\}, \end{aligned} \tag{2}$$

where $\theta = (\mu_1, \mu_2, \sigma_1, \sigma_2, \rho)$ is the vector of parameters. Using a sample $\{X_{1i}, X_{2i}\}$, $i = 1, \ldots, n$, they estimate θ locally in the point x by maximizing the local likelihood function (1), producing

$$\widehat{\theta}(x) = \left(\widehat{\mu}_1(x), \widehat{\mu}_2(x), \widehat{\sigma}_1(x), \widehat{\sigma}_2(x), \widehat{\rho}(x)\right),$$

and take special interest in the estimated correlation function $\widehat{\rho}(x)$ (i.e. the LGC) because it serves as an attractive local measure of statistical dependence between X_1 and X_2. They show that the LGC reveals many types of nonlinear statistical dependence that are not captured by the ordinary (global) Pearson correlation coefficient. Furthermore, the LGC distinguishes between positive and negative dependence and reduces to the Pearson ρ if X_1 and X_2 are jointly normal. We refer to Tjøstheim and Hufthammer (2013) for a detailed treatment of the theoretical foundations of the LGC as well as several examples, and rather present two simple illustrations at this point in order to demonstrate the concept.

In Figure 1 we have plotted the estimated LGC for two bivariate data sets on a grid; 1000 simulated observations from a binormal distribution having correlation equal to 0.5, and the daily return on the CAC40 and FTSE100 stock indices on 1000 consecutive trading days starting on May 5th 2014

(Datastream, 2018). In the first panel, we see that the estimated local correlation coincides with the global correlation, except for the estimation error which is comparable to the uncertainty observed in other non-parametric estimation methods such as the kernel density estimator (see, for instance, Otneim and Tjøstheim (2017) for a formal asymptotic analysis of relevant convergence rates). In the second panel we see clearly that the local correlation, and thus the dependence, is stronger in the lower left and upper right regions of the distribution than in the central parts. The phenomenon of local dependence is well known in the financial literature, and using the LGC it can be measured, interpreted, and visualized in a natural way. The interpretation of this particular figure is that extreme observations on the two stock indices are more strongly dependent than the less extreme observations.

One may obtain these particular estimates from the older **localgauss**-package (as well as the **lg**-package of course), but the plotting routine that was used to produce these figures is included in the **lg**-package and will be described in more detail in Section 2.5.

Taking the LGC as a measure of dependence opens up for a number of possibilities to construct statistical tests. Berentsen and Tjøstheim (2014) show that $\rho(x) \equiv 0$ implies that X_1 and X_2 are independent. They show further that independence between X_1 and X_2 implies $\rho(x) \equiv 0$ if the population values of the local mean and standard deviation functions satisfy the following conditions: $\mu_i(x_1, x_2) = \mu_i(x_i)$ and $\sigma_i(x_1, x_2) = \sigma_i(x_i)$ for $i = 1, 2$. Equivalence between independence and $\rho(x) \equiv 0$ holds in general if the observations have been suitably transformed according to a procedure presented later in this section. It follows then that departures from $\widehat{\rho}(x) \equiv 0$ may be taken as evidence against the hypothesis that X_1 and X_2 are statistically independent. Berentsen and Tjøstheim (2014) formalize this notion by testing whether $\rho(x) \equiv 0$ for all $x \in S \subset \mathbb{R}^2$ using the test statistic

$$T_{n,b} = \int_S h\left(\widehat{\rho}(x)\right) \, \mathrm{d}F_n(x) \tag{3}$$

for some non-negative function h, for example $h(x) = x^2$ or $h(x) = |x|$. Critical values may be obtained by permutations of the data under the null hypothesis, and we demonstrate the implementation of this test using the **lg**-package in Section 2.4.2.

Consider next the stationary time series $\{X_t\}$. The autocorrelation function (ACF) $\rho_k = \rho(X_t, X_{t-k})$ is a well known concept for describing the serial dependence in the time series, but the ACF is, again, only capable to completely capture *linear* serial dependence. Lacal and Tjøstheim (2017) seek to remedy this by rather calculating the the local correlation between X_t and X_{t-k}. This leads to a test for serial independence in the natural way. In fact, this work is mainly a theoretical exercise in order to accommodate time series dependence. Testing for independence between X_t and X_{t-k} using observations $\{X_t, X_{t-k}\}_{t=k+1}^{T}$ leads to the same test statistic (3) and bootstrap procedure as the test for independence between X_1 and X_2 that we described above.

Lacal and Tjøstheim (2018) extend this problem to test for serial cross-dependence between two time series $\{X_t, Y_t\}$ by measuring the LGC between X_t and Y_{t-k}. Departures from $\widehat{\rho}(x, y) \equiv 0$ is again taken as evidence against independence, and the test statistic (3) provides an aggregate measure of this discrepancy in the specified region S. In this case, however, we can not obtain replicates of the test statistic under the null hypothesis by simple permutations of the data. Lacal and Tjøstheim (2018) suggest rather two block bootstrap procedures to this end, using fixed and random block sizes respectively. The tests for serial dependence, and serial cross-dependence are both implemented in the **lg**-package, as we demonstrate in Section 2.4.2.

We find another application of the local Gaussian approximation in the work by Støve et al. (2014), who measure and test for financial contagion. They define contagion by "a significant increase in cross-market linkages after a shock to one country" (Forbes and Rigobon, 2002, p. 2223) and employ the LGC to quantify this potential linkage. The authors estimate the LGC on a grid $\{x_1, x_2\}_k$, $k = 1, \ldots, K$ along the diagonal $D = \{(x_1, x_2) : x_1 = x_2\}$ *before* and *after* some critical event in the financial markets, denoted as the crisis (C) and the non-crisis (NC) periods respectively. They compare the two estimates using the following test statistic;

$$T_{n,b}^{D} = \sum_{k=1}^{K} \left\{\widehat{\rho}_C(x_k, x_k) - \widehat{\rho}_{NC}(x_k, x_k)\right\} w(x_k, x_k),$$

where $w(\cdot, \cdot)$ is a weight function that serve the same purpose as the integration area S in (3). In this case, Støve et al. (2014) show that a standard bootstrap will suffice in order to produce approximate replicates of $T_{n,b}^{D}$ under the null hypothesis of no financial contagion, and we demonstrate the implementation of this test using the **lg**-package in Section 2.4.3.

Although the original work by Hjort and Jones (1996) provide a general framework for local likelihood density estimation using any p-variate parametric family as the local family, it is evident that the method may struggle in multivariate applications much in the same way as the kernel density estimator does. This is a consequence of the curse of dimensionality, the effect of which is

sought remedied by an algorithm provided by Otneim and Tjøstheim (2017). The idea is to fit the p-variate normal distribution $\psi(\mu, \Sigma)$ locally, where μ is the vector of p expectations, and Σ is the $p \times p$ covariance matrix (to which the correlation matrix R corresponds), but under the following structural simplifications:

$$\mu_i(x) = \mu_i(x_1, \ldots, x_p) \stackrel{\text{def}}{=} \mu_i(x_i) \tag{4}$$

$$\sigma_i(x) = \sigma_i(x_1, \ldots, x_p) \stackrel{\text{def}}{=} \sigma_i(x_i) \tag{5}$$

$$\rho_{ij}(x) = \rho_{ij}(x_1, \ldots, x_p) \stackrel{\text{def}}{=} \rho_{ij}\left(x_i, x_j\right). \tag{6}$$

Otneim and Tjøstheim (2017) estimate the local parameters above by first obtaining univariate marginal locally Gaussian fits (eqs. 4 and 5), and then pairwise bivariate locally Gaussian fits (eq. 6). In the second step the estimates $\widehat{\mu}_i(x_i)$, $\widehat{\mu}_j\left(x_j\right)$, $\widehat{\sigma}_i(x_i)$ and $\widehat{\sigma}_j\left(x_j\right)$ are kept fixed in the estimation of the pairwise local correlation. The motivation for introducing the simplifications defined in equations 4-6 can be compared to the practical advantages of estimating additive regression models, where $\mathrm{E}(Y) = f(x_1, \ldots, x_p) \stackrel{\text{def}}{=} f_1(x_1) + \cdots + f_p(x_p)$. Otneim and Tjøstheim (2017) argue further that the following transformation technique will produce better density estimates in many situations.

Denote by $F_i(x_i)$, $i = 1, \ldots, p$ the marginal distribution functions of the stochastic vector X, and by $\widehat{F}_i(x_i) = n^{-1}\sum_{i=1}^{n} 1(X_i \leq x_i)$ their empirical counterparts. They then estimate the density $f_Z(z)$ of the vector $Z = \left\{\Phi^{-1}(F_i(X_i))\right\}_{i=1,\ldots,p}$, in practice approximated by

$$\widehat{Z} = \left\{\Phi^{-1}\left(\widehat{F}_i(X_i)\right)\right\}_{i=1,\ldots,p}, \tag{7}$$

and where $\Psi(\cdot)$ is the univariate standard normal cdf. In that case, they simplify the estimation problem even further and fix

$$\mu_i(z_i) \stackrel{\text{def}}{=} 0 \text{ and } \sigma_i(z_i) \stackrel{\text{def}}{=} 1, \ i = 1, \ldots, p, \tag{8}$$

so that the only parameter functions left to estimate are the pairwise local Gaussian correlations $R(z) = \left\{\rho_{ij}\left(z_i, z_j\right)\right\}_{i<j}$. We use the notation Z, z_i and z_j to signify that the estimation is performed on the (approximate) standard normal scale, or z-scale for short. We can then estimate the joint density $f_Z(z)$ of Z as

$$\widehat{f}_Z(z) = \psi\left(z; \mu(z) = 0, \sigma(z) = 1, R = \widehat{R}(z)\right), \tag{9}$$

where $\mu(z) = \{\mu_i(z)\}$ and $\sigma(z) = \{\sigma_i(z)\}$ for $i = 1, \ldots, p$, and then substitute f_Z for $\widehat{f}_Z$ in the following relation obtained by Otneim and Tjøstheim (2017) in order to estimate the unknown density f_X:

$$f(x) = f_Z\left(\Phi^{-1}(F_1(x_1)), \ldots, \Phi^{-1}(F_p(x_p))\right) \times \prod_{i=1}^{p} \frac{f_i(x_i)}{\phi\left(\Phi^{-1}(F_i(x_i))\right)}, \tag{10}$$

where $\phi(\cdot)$ is the standard normal pdf. This estimator is implemented the lg-package as demonstrated in Section 2.4.1.

One particular feature enjoyed by the jointly normally distributed vector X is that, for any partitioning $X = \left(X^{(1)}, X^{(2)}\right)$, the conditional distribution of $X^{(1)}|X^{(2)} = x^{(2)}$ is also normal. In fact, if $X \sim \mathcal{N}(\mu, \Sigma)$, and μ and Σ is partitioned according to $\left(X^{(1)}, X^{(2)}\right)$ as

$$\mu = \begin{pmatrix} \mu_1 \\ \mu_2 \end{pmatrix} \text{ and } \Sigma = \begin{pmatrix} \Sigma_{11} & \Sigma_{12} \\ \Sigma_{21} & \Sigma_{22} \end{pmatrix},$$

then $X^{(1)}|X^{(2)} = x^{(2)} \sim \mathcal{N}(\mu^*, \Sigma^*)$, where

$$\mu^* = \mu_1 + \Sigma_{12}\Sigma_{22}^{-1}\left(x^{(2)} - \mu_2\right) \tag{11}$$

$$\Sigma^* = \Sigma_{11} - \Sigma_{12}\Sigma_{22}^{-1}\Sigma_{21}, \tag{12}$$

see e.g. (Johnson and Wichern, 2007, chapter 4). Otneim and Tjøstheim (2018) demonstrate that this property may be translated into a corresponding local argument without modification. That is, if the

joint density $f_X(\cdot)$ can be written on a locally Gaussian form

$$f_X(x) = \psi(x, \mu(x), \Sigma(x)),$$

then the conditional density of $X^{(1)}|X^{(2)} = x^{(2)}$ can also be written on the same locally Gaussian form with local parameters given by equations (11) and (12), except that all quantities are x-dependent. If we use the transformation technique described above together with simplification (8), the local versions of equations (11) and (12) simplify to

$$\mu^*(z) = R_{12}(z) R_{22}(z)^{-1} z^{(2)}, \tag{13}$$

$$\Sigma^*(z) = R_{11}(z) - R_{12}(z) R_{22}(z)^{-1} R_{21}(z), \tag{14}$$

where we, again, switch to z-notation in order to make it clear that these quantities are estimated on the standard normal z-scale. An estimator $\widehat{f}_{X^{(1)}|X^{(2)}}(\cdot|\cdot)$ of the conditional density $f_{X^{(1)}|X^{(2)}}(\cdot|\cdot)$ follows immediately from an expression corresponding to (10), and the **lg**-package provides functions for implementing this estimator in R. We describe the implementation of this functionality in Section 2.4.1.

Finally, we refer to Otneim and Tjøstheim (2021) who take the local version of the conditional covariance matrix (12) (or (14) in the transformed case) as a measure for conditional dependence, and thus as an instrument to test for conditional independence. Consider the stochastic vector $X = \left(X^{(1)}, X^{(2)}, X^{(3)}\right)$, where $X^{(1)}$ and $X^{(2)}$ are scalars and $X^{(3)}$ may be a vector. $X^{(1)}$ is conditionally independent from $X^{(2)}$ given $X^{(3)}$, written $X^{(1)} \perp X^{(2)} \mid X^{(3)}$, if the stochastic variables $X^{(1)} \mid X^{(3)}$ and $X^{(2)} \mid X^{(3)}$ are independent, or, equivalently, if the joint conditional density of $X^{(1)}$ and $X^{(2)}$ given $X^{(3)}$ can be written as the product

$$f_{X^{(1)},X^{(2)}|X^{(3)}}\left(x^{(1)}, x^{(2)}|x^{(3)}\right) = f_{X^{(1)}|X^{(3)}}\left(x^{(1)}|x^{(3)}\right) \times f_{X^{(2)}|X^{(3)}}\left(x^{(2)}|x^{(3)}\right). \tag{15}$$

In this case, denote by $\alpha(z)$ the off-diagonal element in the 2×2 local correlation matrix $R^*(z)$ (which derives directly from $\Sigma^*(z)$ as given in (14)). If X has a local Gaussian distribution, the conditional independence (15) is equivalent to $\alpha(z) \equiv 0$, and Otneim and Tjøstheim (2021) take departures from this relation as evidence against the hypothesis of conditional independence between $X^{(1)}$ and $X^{(2)}$ given $X^{(3)}$. The natural way to quantify this is the test functional

$$T_{n,b}^{CI} = \int h\left(\widehat{\alpha}(z)\right) \mathrm{d}F_n(z). \tag{16}$$

Otneim and Tjøstheim (2021) describe a bootstrap procedure for generating replicates of $T_{n,b}^{CI}$ under the null hypothesis. In Section 2.4.4 we demonstrate how the **lg**-package may be used to extract estimates of the local partial correlation and perform tests for conditional independence according to this scheme.

The first step: Creating the lg-object

The local Gaussian correlation may be used to perform a number of statistical analyses, as is evident from the preceding section. The practitioner must first, however, make three quite specific modeling choices; namely (i) to choose an estimation method, i.e. the level of simplification in multivariate applications, (ii) to determine whether the data should be transformed towards marginal standard normality before estimating the LGC, and (iii) to choose a set of bandwidths, or at least a method for calculating bandwidths. The architecture of the **lg**-packages requires the user to make these choices *before* endeavoring further into specific applications by imposing a strict, two step procedure:

1. Create an lg-object.
2. Apply relevant analysis functions to the lg-object.

In the following we assume that one has a *data set* `x` loaded into the R workspace, which must be an $n \times p$ matrix (one column per variable, one row per observation), possibly including `NA`s which will be excluded from the analysis, or a data frame having the same dimensions. The fundamental syntax for creating an lg-object is `lg_object <-lg_main(x)`, and we will in this section explain how the modeling decisions (i)-(iii) can be encoded into the lg-object by using appropriate arguments in this function.

Selecting the estimation method

Given a data set x having n rows and $p \geq 2$ columns, the user must choose between four distinct estimation methods, and specify this choice by using the argument `est_method` to the `lg_main()`-function. We look at the built-in bivariate data set `faithful`, which records the waiting time between eruptions and the duration of the eruption for the Old Faithful geyser in the Yellowstone National Park, USA (see the help file in R for more details: `?faithful`), and load the lg-package in order to demonstrate the implementation:

```
R> x <- faithful
R> library(lg)
```

1. A full locally Gaussian fit for bivariate data. If $p = 2$, we may fit the bivariate normal $\psi(x, \theta)$ locally to $f(x)$, and by a "full local fit" we mean that we jointly estimate the five parameters

$$\theta(x) = \left(\mu_1(x_1, x_2), \mu_2(x_1, x_2), \sigma_1(x_1, x_2), \sigma_2(x_1, x_2), \rho(x_1, x_2)\right)$$

by optimizing the local likelihood function (1) in the grid point $x = (x_1, x_2)$. To use this estimation method in the subsequent analysis, specify `est_method = "5par"` in the call to `lg_main()`:

```
R> lg_object <- lg_main(x, est_method = "5par")
```

The resulting `lg_object` is a list of class `lg`, and we may confirm that the assignment has been carried out correctly by inspecting its `est_method`-element:

```
R> lg_object$est_method

[1] "5par"
```

Note that the full locally Gaussian fit for raw data is not available if the number of variables p is greater than 2. The `lg_main()`-function will check for this and print out an error message if $p > 2$ and `est_method = "5par"`.

2. A simplified locally Gaussian fit for multivariate data. As described in the preceding section, we may construct a simplified estimation procedure for calculating the LGC in two steps, which in principle works for any number of dimensions (including $p = 2$):

1. Calculate $\mu_i(x_i)$ and $\sigma_i(x_i)$, $i = 1, \ldots, p$ by fitting the univariate normal distribution locally to each *marginal* density $f_i(x_i)$ of $f(x)$.
2. Keep $\widehat{\mu}_i(x)$ and $\widehat{\sigma}_i(x_i)$, $i = 1, \ldots, p$ from step 1 fixed when estimating $\rho_{ij}\left(x_i, x_j\right)$, $1 \leq i < j \leq p$ by fitting the bivariate normal distribution to each *pair* of variables X_i and X_j.

To use this method, create the lg-object by running the following line:

```
R> lg_object2 <- lg_main(x, est_method = "5par_marginals_fixed")
```

3. A simplified locally Gaussian fit for marginally standard normal data. This estimation method is applicable for marginally standard normal data, or data that have been transformed to approximate marginal standard normality by e.g. the transformation (7). In that case, we fix the marginal expectation functions and standard deviation functions to the constant values 0 and 1 respectively, and estimate only the pairwise local Gaussian correlations as in (9). To use this estimation method, create the lg-object by running

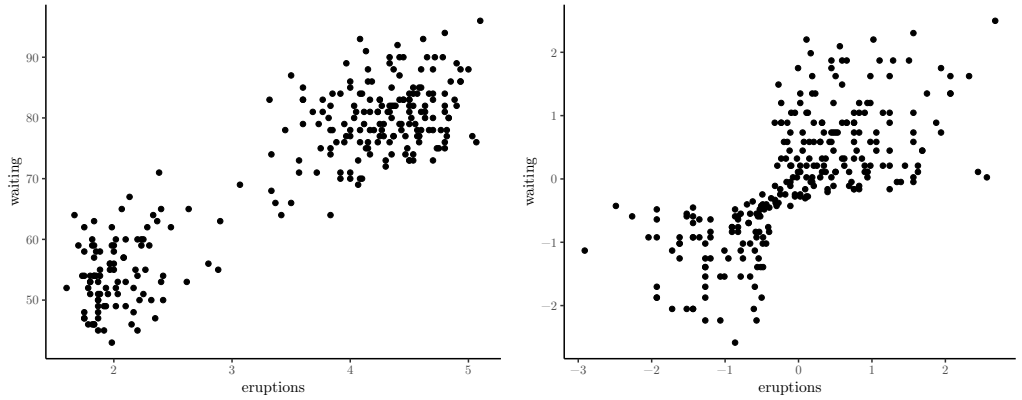
```
R> lg_object3 <- lg_main(x, est_method = "1par")
```

Note that the function call above will issue a warning if the option for transforming the data to marginal standard normality is not at the same time set to `TRUE`, see the next sub-section on data transformation for details.

4. A full locally Gaussian fit for trivariate data. If the number of variables p is equal to 3 *and* we choose to transform the data to marginal standard normality (see the next sub-section), the transformed density $f_Z(\cdot)$ in (9) may be estimated by jointly estimating the three local correlations $\rho_{12}(z_1, z_2, z_3)$, $\rho_{13}(z_1, z_2, z_3)$ and $\rho_{23}(z_1, z_2, z_3)$. This estimation method was introduced recently by Otneim and Tjøstheim (2021) in order to increase the power of their conditional independence test, but it can be used in any application described in this paper that consider trivariate data. To use this estimation method, create the lg-object by running

```
R> lg_object4 <- lg_main(x, est_method = "trivariate")
```

This command will throw an error if the data set x does not have exactly three columns.



(a) The original data.

(b) The same data transformed to marginal standard normality.

Figure 2: The same data on different scales.

Data transformation

Next, the user must determine if the local Gaussian correlation should be estimated directly on the raw data or on the marginally normal pseudo observations (7). This is carried out by using the logical `transform_to_marginal_normality`-argument in `lg_main`, for example:

```
R> lg_object <- lg_main(x, transform_to_marginal_normality = TRUE)
```

The resulting `lg_object` now includes the element `transform_to_marginal_normality` set according to the input, and if this is `TRUE`, it also includes the `transformed_data` and a function `trans_new()` that may be used later to apply the same transformation to e.g. grid points. If the transformation option is set to `FALSE`, the `transformed_data` element contains the input data `x`, and `trans_new()` is nothing more than the identity mapping for points in $\mathbb{R}^p$. See Figure 2 for two plots that demonstrate the effect of the data transformation on the example data.

Bandwidth selection

Finally, the user must specify a set of bandwidths, or a method for calculating them. Given that the different estimation methods described in Section 2.3.1 require different sets of bandwidths (i.e joint, marginal and/or pairwise), the easiest approach for the user is to leave the selection and formatting of the bandwidths to the `lg_main()`-function.

The bandwidth plays a slightly different role in local likelihood estimation than elsewhere in the nonparametric literature. It controls the *level of localization* and thus only indirectly the *smoothness* of the estimates. Indeed, if we concentrate on the univariate case for the moment and assume that the (single) bandwidth b is small, we see from the local likelihood function (1) that only the very few observations closest to a fixed grid point x_0 will have significant weight when determining the local parameter estimates $\widehat{\theta}_0(x)$ at that point. Moving on to another nearby point x_1 may then lead to a fairly different estimate $\widehat{\theta}(x_1)$ because the set of observations having weight in this point is very different. This may again lead to rougher parameter estimates $\widehat{\theta}(x)$ and in turn also to rougher density estimates $f\left(x, \widehat{\theta}(x)\right)$.

If the bandwidth b grows large, on the other hand, all observations receive similar weights, and furthermore: the local likelihood function (1) becomes approximately proportional to the ordinary (global) likelihood function $L_n(\theta) = \sum_{i=1}^{n} \log f(X_i; \theta)$. In other words, the local parameter estimates $\widehat{\theta}(x)$ are smoothed towards the constant maximum likelihood estimates $\widehat{\theta}_{ML}$, and the estimated density $f\left(x; \widehat{\theta}(x)\right)$ towards the maximum likelihood estimate $f\left(x; \widehat{\theta}_{ML}\right)$. This means that the bandwidth may be chosen to reflect the goodness-of-fit of $f(x; \theta)$ to the true density $f(x)$.

In the multivariate applications referred to in this paper, the bandwidth b in (1) is a diagonal matrix, and $1/b$ is naturally taken to represent its inverse.

We have in practice seen two automatic bandwidth selectors employed in the applications referred to in Section 2.2: a cross-validation procedure, that is fairly slow to compute, but accurate with respect to density estimation, and a plug-in bandwidth that is much quicker to calculate, but less accurate

with respect to density estimation. We use the argument `bw_method` to the `lg_main()`-function in order to choose between the two.

1. Choosing bandwidths by cross-validation. The functional

$$CV\left(b\right) = -\frac{1}{n}\sum_{i=1}^{n}\log f\left(X_i;\widehat{\theta}^{(-i)}\left(X_i\right)\right),$$

where $\widehat{\theta}^{(-i)}\left(x\right)$ is the parameter estimate obtained after deleting observation X_i from the data, is proportional to a quantity that estimates the Kullback-Leibler distance between $f\left(\cdot,\widehat{\theta}\left(\cdot\right)\right)$ and the true density $f\left(\cdot\right)$, see Berentsen and Tjøstheim (2014). The cross-validated bandwidth b_{CV} is hence given by

$$b_{CV} = \arg\min_{b}\ CV\left(b\right).$$

If we, for example, wish to use the simplified estimation procedure on the transformed data, we need bandwidths for the marginal estimates of the local means and standard deviations, as well as a 2×2 diagonal bandwidth matrix for each pair of variables. This is accomplished by the following call to `lg_main()`:

```
R> # Create the lg-object with bandwidths chosen by cross-validation
R> lg_object <- lg_main(x,
R+                      est_method = "5par_marginals_fixed",
R+                      transform_to_marginal_normality = TRUE,
R+                      bw_method = "cv")
```

The `lg_object` now contains the necessary bandwidths for this configuration, as can be seen by inspecting the contents of its `bw`-element:

```
R> # Print out the bandwidths
R> lg_object$bw

$marginal
[1] 0.9989327 0.9875333

$marginal_convergence
[1] 0 0

$joint
x1 x2         bw1      bw2 convergence
1  1  2 0.2946889 0.331971           0
```

This is itself a list, containing the crucial elements `marginal` for the p marginal bandwidths, and `joint` that contains the $p(p-1)/2$ bandwidth matrices, one for each pair of variables (which in this bivariate example just *one* variable pair, (`x1, x2`)). The convergence flags stem from the built-in R functions `optim()` and `optimize()` that we use to obtain the minimizer of $CV\left(\cdot\right)$, and 0 indicates successful convergence.

2. Using plug-in bandwidths. Obtaining cross-validated bandwidths is unfortunately fairly slow on a standard computer. For sample sizes in the 500-1000 range, the process may take several minutes, which is unfeasible when embarking on analyses that require e.g. resampling. We have therefore implemented a quick plug-in bandwidth selector as well, that may suffice in many practical situations, especially at the initial or exploratory stage.

Otneim and Tjøstheim (2017) show that the simplified version of the local Gaussian fit have the same convergence rates as the corresponding nonparametric kernel density estimator, for which Silverman (1986) derives the plug-in formula $b = 1.08\cdot sd\left(x\right)\cdot n^{-1/5}$. By specifying `bw_method = "plugin"`, the `lg_main()`-function will select the bandwidths correspondingly, except that the exponent changes to $-1/6$ for joint bandwidths, and the proportionality constant is by default set to 1.75. The latter number is the result of regressing b_{CV} on $n^{-1/6}$ in a large simulation experiment covering various data generating processes (Otneim, 2016). We see the effect of switching to plugin bandwidths in the code below:

```
R> # Make the lg-object with plugin bandwidths
R> lg_object <- lg_main(x,
R+                      est_method = "5par_marginals_fixed",
R+                      transform_to_marginal_normality = TRUE,
R+                      bw_method = "plugin")
```

Argument	Explanation	Default value
x	The data, an $n \times p$ matrix or data frame	
bw_method	Method for calculating the bandwidths	"plugin"
est_method	Estimation method	"1par"
transform_to_ marginal_normality	Transform the data?	TRUE
bw	The bandwidths to use if already calculated	NULL
plugin_constant_ marginal	Prop. const. in plugin formula for marg. bw.	1.75
plugin_exponent_ marginal	Exponent in plugin formula for marg. bw.	$-1/5$
plugin_constant_ joint	Prop. const. in plugin formula for joint bw.	1.75
plugin_exponent_ joint	Exponent in plugin formula for joint bw.	$-1/6$
tol_marginal	Abs. tolerance when optimizing $CV\,(b)$, marg.	10^{-3}
tol_joint	Abs. tolerance when optimizing $CV\,(b)$, joint	10^{-3}

Table 1: Arguments to the initialization function lg_main()

```
R> # Print out the bandwidths
R> lg_object$bw

$marginal
[1] 0.5703274 0.5703274

$marginal_convergence
[1] NA NA

$joint
  x1 x2       bw1       bw2 convergence
1  1  2 0.6875061 0.6875061          NA
```

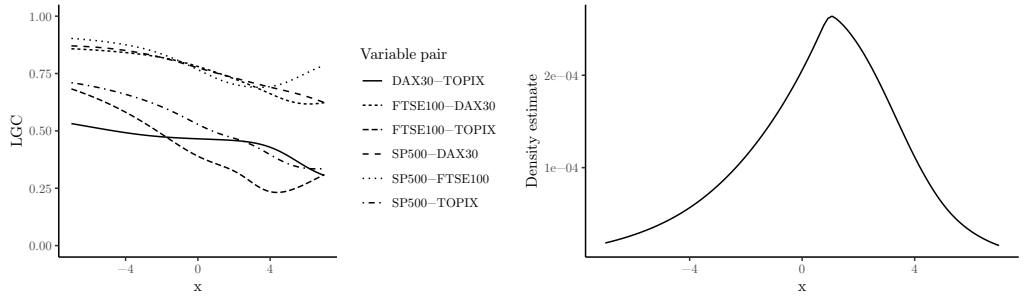
Summary of the initialization function

In the sub-section above we present the three most important arguments to lg_main(). Each of them allows the user to configure one of the three crucial modeling choices. Let us complete this treatment by covering some possibilities to make further adjustments to those choices.

1. The user may supply the bandwidths directly to lg_main() by passing them to the bw-argument. They have to be in the correct format though, which is a list containing the vector $marginal if est_method = "5par_marginals_fixed", and always a data frame $joint specifying all variable pairs in the x1 and x2 columns and the corresponding bandwidths in the bw1 and bw2 columns. The function bw_simple() will assist in creating bandwidth objects.
2. If bw_method = "plugin" the user may change the proportionality constant and exponent in the plugin formula for the joint and, if applicable, the marginal bandwidths. See Table 1 for the necessary argument names.
3. If bw_method = "cv", the user may change the numerical tolerance in the optimization of $CV\,(b)$. See Table 1 for the necessary argument names.

Statistical inference using the lg-package

We proceed in this section to demonstrate how to implement each of the tasks that we discussed in Section 2.2. The general pattern is to pass the lg-object to one of the estimation or test functions provided in the lg-package. We will look at some financial data in the examples: the monthly returns on



(a) The estimated pairwise LGC for the stock data along the diagonal indicating that the pairs of stocks are more strongly dependent in the lower part of the distribution than the upper part.

(b) The corresponding joint density estimate of the four return series, plotted along the diagonal in $\mathbb{R}^4$.

Figure 3: Local correlations and density estimates calculated using the `dlg()-function`.

the S&P500, FTSE100, DAX30 and TOPIX stock indices from January 1985 to March 2018 (Datastream, 2018).

Density estimation

We start by introducing a basic function for estimating the LGC on a grid as described by Otneim and Tjøstheim (2017), and thus also a probability density estimate. We create a grid, `x0`, having the same number of columns as the data in the code below. Note that we use the pipe operator `%>%` from the **magrittr**-package (Bache and Wickham, 2014) as well as functions from the **dplyr**-package (Wickham et al., 2018) for easy manipulation of data frames. We then pass the grid and the lg-object containing our modeling choices to the `dlg()`-function in order to do the estimation.

```
R> # Create an lg-object
R> lg_object <- lg_main(x = stock_data %>% select(-Date),
R+                      est_method = "1par",
R+                      bw_method = "plugin",
R+                      transform_to_marginal_normality = TRUE)
R>
R> # Construct a grid diagonally through the data.
R> grid_size <- 100
R> x0 <- stock_data %>%
R+   select(-Date) %>%
R+   apply(2, function(y) seq(from = -7,
R+                             to = 7,
R+                             length.out = grid_size))
R>
R> # Estimate the local Gaussian correlation on the grid
R> density_object <- dlg(lg_object, grid = x0)
```

The last line of code creates a list containing a number of elements. The two most important are `$loc_cor`, which is a matrix of local correlations having one row per grid point and one column per *pair of variables* (the columns correspond to the rows in `density_object$pairs`), as well as `$f_est`, which is a vector containing the estimate $\widehat{f}_X(x)$ of the joint density $f_X(x)$ in the grid points specified in `x0`. The estimated local correlations for this example is plotted in Figure 3a, and the corresponding density estimate is plotted (along the diagonal $x_1 = x_2 = x_3 = x_4 = x$) in Figure 3b.

The list `density_object` contains the estimated standard deviations of the local correlations in `$loc_cor_sd`, as well as lower and upper confidence bands `$loc_cor_lower` and `$loc_cor_upper` at the 95% level. We refer to Table 2 for a complete overview of the arguments to `dlg()`.

Note that the configuration `transform_to_marginal_normality = TRUE` and `est_method = 5par` in the bivariate case coincides with the situation considered by Tjøstheim and Hufthammer (2013). In that case, `dlg()` serves as a wrapper for the function `localgauss()` in the **localgauss**-package (Berentsen et al., 2014).

Obtaining the estimate of a conditional density using the Otneim and Tjøstheim (2018) algorithm described in Section 2.2 is very similar, but one must take particular care of the *ordering* of the variables in the data set. The estimation function, `clg()`, will *always* assume that the free variables come first,

Argument	Explanation	Default value
`lg_object`	The lg-object created by `lg_main()`	
`grid`	The evaluation points for the LGC	`NULL`
`level`	Level for confidence bands	0.95
`normalization _points`	The estimated density does not integrate to one by construction. `dlg()` will generate the given number of normal variables, having the same moments as the data, approximate $\int \widehat{f}_X(x)\, \mathrm{d}x$ by a Monte Carlo integral, and then normalize the density estimate accordingly	`NULL`
`bootstrap`	Calculate bootstrapped confidence intervals instead of asymptotic expressions	`FALSE`
`B`	Number of bootstrap replicates	`500`

Table 2: Arguments to the `dlg()`-function

and that the conditioning variables come last. Let us illustrate this in the following code chunk by estimating the joint conditional density of S&P500 and FTSE100, given that DAX30 = TOPIX = 0.

```
R> # We must make sure that the free variables come first
R> returns1 <- stock_data %>% select(SP500, FTSE100, DAX30, TOPIX)
R>
R> # Create the lg-object
R> lg_object <- lg_main(returns1,
R+                      est_method = "1par",
R+                      bw_method = "plugin",
R+                      transform_to_marginal_normality = TRUE)
R>
R> # Create a grid
R> x0 <- returns1 %>%
R+   select(SP500, FTSE100) %>%
R+   apply(2, function(y) seq(from = -7,
R+                            to = 7,
R+                            length.out = grid_size))
R>
R> # Calculate the conditional density
R> cond_density <- clg(lg_object, grid = x0, cond = c(0, 0))
```

The key argument in the call to `clg()` above is `cond = c(0, 0)`. This means that the last two variables are conditioning variables (and hence that the first $4 - 2 = 2$ variables are free), and that the value of the conditioning variables are fixed at DAX30 = 0 and TOPIX = 0 respectively. This also means that the number of columns in the grid `x0` plus the number of elements in `cond` must equal the number of variables p in the data set, and the call to `clg()` will result in an error message if this requirement is not fulfilled. The `clg()`-function takes mostly the same arguments as `dlg()` listed in Table 2, and the conditional density estimate in our example is available in the vector `cond_density$f_est`.

Tests for independence

Three independence tests based on the LGC have appeared in the literature thus far:

1. A test for independence between the stochastic variables X_1 and X_2 based on iid data, cf. Berentsen and Tjøstheim (2014).
2. A test for serial independence between X_t and X_{t-k} within a time series $\{X_t\}$, cf. Lacal and Tjøstheim (2017).
3. A test for serial cross-dependence between X_t and Y_{t-k} for two time series $\{X_t\}$ and $\{Y_t\}$, cf. Lacal and Tjøstheim (2018).

Argument	Explanation	Default value
`lg_object`	The lg-object created by `lg_main()`	
`h`	The function $h(\cdot)$ in (3)	`function(x) x^2`
`S`	The integration area S in (3). Must be a logical function on potential grid points in $\mathbb{R}^2$	`function(x) as.logical(rep(1, nrow(x)))`
`bootstrap_type`	The bootstrap method, must be either `"plain"`, `"block"` or `"stationary"`	`"plain"`
`block_length`	Block length for the block bootstrap, mean block length for the stationary bootstrap. Calculated by `np::b.star()` (Hayfield and Racine, 2008) if not provided	`NULL`
`n_rep`	Number of bootstrap replicates	1000

Table 3: Arguments to the `ind_test()`-function

As we noted in Section 2.2, their practical implementations are very similar, and the lg-package provides the function `ind_test()` to perform the tests. Let us first consider the iid case, and generate 500 observations `test_x` from the well known parabola model $X_2 = X_1^2 + \varepsilon$, where both X_1 and ε are independent and standard normal. In this case, X_1 and X_2 are uncorrelated, but obviously strongly dependent. Berentsen and Tjøstheim (2014) considers mainly the full bivariate fit to the raw data, which we easily encode into the lg-object as before. The implementation of the test using 100 bootstrap replicates is shown below.

```
R> # Make the lg-object
R> lg_object <- lg_main(test_x,
R+                      est_method = "5par",
R+                      transform_to_marginal_normality = TRUE)
R> # Perform the independence test
R> test_result <- ind_test(lg_object, n_rep = 100)
R> # Print out the p-value of the test
R> test_result$p_value

[1] 0
```

This may take a few minutes to run on a desktop computer due to the bootstrapping. The small p-value indicate that we reject the null-hypothesis of independence between X_1 and X_2 in the parabola model defined above. We can further specify the function h and the integration area S in the test statistic (3), see Table 3 for details.

The only difference when testing for serial independence within a time series $\{X_t\}$ is to create a two-column data set consisting of X_t and X_{t-k}. For example, if we wish to perform this test for $k = 1$ for one of the variables in the stock-exchange series, create the matrix of observations as below, and proceed exactly as in the iid case.

```
R> returns2 <- stock_data %>% select(SP500) %>%
R+   mutate(sp500_lagged = lag(SP500))
```

Finally, the only thing that we must alter in order to perform the third test for serial cross-dependence is the bootstrap method. In the applications above, it suffices to use the standard bootstrap, where we resample with replacement from the data. This is implemented in the `ind_test()`-function by setting the `bootstrap_type`-argument to `"plain"`, which is the default option. When testing for serial cross-dependence we need to use a block-bootstrap procedure, and Lacal and Tjøstheim (2018) consider two options here: The block bootstrap with either fixed (Kunsch, 1989) or random (Politis and Romano, 1994) block sizes, and this choice is specified by choosing `bootstrap_type = "block"` or `bootstrap_type = "stationary"`, respectively, in the call to `ind_test()`. Lacal and Tjøstheim (2018) do not report significant differences in test performance using the different bootstrap types.

Argument	Explanation	Default value
`lg_object_nc`	The lg-object covering the non-crisis period	
`lg_object_c`	The lg-object covering the crisis period	
`grid_range`	Range of diagonal for measuring the LGC	(5%, 95%) quantiles
`grid_length`	The number of grid points to use	30
`n_rep`	Number of bootstrap replicates	1000
`weight`	Weight function	`function(y) rep(1,nrow(y))`

Table 4: Arguments to the `cont_test()-function`

Test for financial contagion

Assume that we observe two financial time series $\{X_t\}$ and $\{Y_t\}$ at times $t = 1, \ldots, T$, and that some crisis occurs at time $T^* < T$. As described in Section 2.2, Støve et al. (2014) measure the local correlation between $\{X_t\}$ and $\{Y_t\}$ *before* and *after* T^*, and take significant differences between these measurements as evidence against the null hypothesis of no linkage, or contagion, between the time series. In order to implement this test using the lg-package, one must create *two* lg-objects: one for the observations covering the non crisis period and one covering the crisis period. We do not enter a discussion here how to empirically identify such time periods; this is a job that must be done by the practitioner before performing the statistical test.

Let us illustrate the implementation of this test by looking at the same financial returns data that we have used in preceding sections, but this time we will, in the spirit of Støve et al. (2014), concentrate on *GARCH(1,1)-filtrated daily returns on the S&P500 and FTSE100 indices from 2 January 1985 to 29 April 1987* in order to test for financial contagion between the US and UK stock markets following the global stock market crash of 19 October 1987 ("Black Monday"). Assume that these observations are loaded into the R workspace as the $n_1 \times 2$ data frame `x_nc` containing the observations covering the $n_1 = 728$ days preceding the crisis , and the $n_2 \times 2$ data frame `x_c` containing the observations covering the $n_2 = 140$ consecutive trading days starting on *Black Monday* (see the online code supplement for details concerning the GARCH-filtration and data processing). In the code below we construct one lg-object for each of these data frames with configuration matching the setup used by Støve et al. (2014), and perform the test by means of the `cont_test()`-function.

This function returns a list containing the estimated p-value as well as other useful statistics, including the empirical local correlations measured in the two time periods. See Table 4 for details concerning other arguments that may be passed to this function.

```
R> # Create the two lg-objects
R> lg_object_nc <- lg_main(x_nc,
R+                         est_method = "5par",
R+                         transform_to_marginal_normality = FALSE)
R>
R> lg_object_c <- lg_main(x_c,
R+                         est_method = "5par",
R+                         transform_to_marginal_normality = FALSE)
R>
R> # Run the test with a limited number of bootstrap replicates for
R> # demonstration purposes.
R> result <- cont_test(lg_object_nc, lg_object_c, n_rep = 100)
R>
R> # Print out the p-value
R> result$p_value

[1] 0.01
```

The small p-value means that we reject the null-hypothesis of no financial contagion between the time series after the crisis.

Partial local correlation

Consider finally the work by Otneim and Tjøstheim (2021) who take the off-diagonal element in the local correlation matrix corresponding to the local conditional covariance matrix (12) or (14) as a local measure of conditional dependence between two stochastic variables $X^{(1)}$ and $X^{(2)}$ given the stochastic vector $X^{(3)}$. Furthermore, in the case with data having been transformed to marginal standard normality, they take the statistic

$$T_{n,b}^{CI} = \int h\left(\widehat{\alpha}\left(z\right)\right) \, \mathrm{d}F_n\left(z\right) \tag{17}$$

as a measure of *global* conditional dependence. The **lg**-package provides two key functions in this framework. The first, `partial_cor()`, calculates the local partial correlations as well as their estimated standard deviations on a specified grid in the $\left(x^{(1)}, x^{(2)}, x^{(3)}\right)$-space, and is essentially a wrapper function for the `clg()`-function presented in Section 2.4.1. See Table 5 for details. The second function, `ci_test()`, performs a test for conditional independence between the first two variables in a data set given the remaining variables using the test statistic (17) and a special bootstrap procedure (described briefly below) for approximating the null distribution.

It is well known in the econometrics literature that conditional independence tests are instrumental in the empirical detection of Granger causality (Granger, 1980). For example, if we continue to concentrate on the monthly stock returns data that we have already loaded into memory, we may test whether

$$\mathrm{H}_0: \qquad R_{\mathrm{FTSE100},t} \perp R_{\mathrm{SP500},t-1} \mid R_{\mathrm{FTSE100},t-1} \tag{18}$$

in the period starting in January 2009, the converse of which is a sufficient, but not necessary, condition for $R_{\mathrm{SP500},t}$ Granger causing $R_{\mathrm{FTSE100},t}$. We perform the test by running the code below, where `x` is a data frame having the following columns strictly ordered as $R_{\mathrm{FTSE100},t}$, $R_{\mathrm{SP500},t-1}$ and $R_{\mathrm{FTSE100},t-1}$ (see the online supplement for the pre-processing of data).

The critical values of this test are calculated using the bootstrap under the null hypothesis by independently resampling replicates from the conditional density estimates $\widehat{f}_{X^{(1)}|X^{(3)}}\left(x^{(1)}|x^{(3)}\right)$ and $\widehat{f}_{X^{(2)}|X^{(3)}}\left(x^{(2)}|x^{(3)}\right)$, as obtained by the `clg()`-function, using an approximated accept-reject algorithm. In order to avoid excessive optimization of the local likelihood function (1), we estimate $f_{X^{(1)}|X^{(3)}}$ and $f_{X^{(2)}|X^{(3)}}$ on the univariate regular grids $x_0^{(1)}$ and $x_0^{(2)}$ respectively (while keeping $x^{(3)}$ fixed at the observed values of $X^{(3)}$), and produce interpolating functions $\tilde{f}_{X^{(1)}|X^{(3)}}$ and $\tilde{f}_{X^{(2)}|X^{(3)}}$ using cubic splines. It is much less computationally intensive to generate replicates from $\tilde{f}$ than directly from $\widehat{f}$.

We refer to the documentation of the **lg**-package for details on how to finely tune the behavior of the bootstrapping algorithm by altering the arguments of the `ci_test()`-function, and limit our treatment to describing the arguments most suitable for modifications by the user in Table 6.

```
R> # Create the lg-object
R> lg_object <- lg_main(returns4)
R>
R> # Perform the test
R> test_result <- ci_test(lg_object, n_rep = 100)
R>
R> # Print out result
R> test_result$p_value

[1] 0.51
```

The conditional independence test does not provide evidence against the null hypothesis (18).

Graphics

We conclude this article by describing the `corplot()` function for drawing dependence maps such as those displayed in Figure 1. Berentsen et al. (2014) report on such capabilities in the **localgauss**-package, but the possibility of creating dependence maps was unfortunately removed from **localgauss** in the latest version 0.4.0 due to incompatibilities with the **ggplot2** (Wickham, 2016) plotting engine. We make up for this loss by providing `corplot()`, a function that plots the estimated local correlations as provided by `dlg()`, or the estimated local *partial* correlations as provided by `partial_cor()`.

The plotting function is highly customizable and provides a number of options covering most

basic graphical options. Users well versed in the **ggplot2**-package may also modify the graphical object returned by `corplot()` in the standard way by adding layers as demonstrated in the example below.

In the first example we generate a set of bivariate normally distributed data using the **mvtnorm**-package (Genz et al., 2018) and estimate the local Gaussian correlation on a regular `grid` using the `dlg()`-function. Passing the resulting `dlg_object` to `corplot()` without further arguments results in Figure 4.

```
R> # Make a regular grid in the domain of the distribution
R> grid <- expand.grid(seq(-3, 3, length.out = 7),
R+                     seq(-3, 3, length.out = 7))
R>
R> x <- mvtnorm::rmvnorm(500, sigma = matrix(c(1, rho, rho, 1), 2))
R> lg_object <- lg_main(x,
R+                      est_method = "5par",
R+                      transform_to_marginal_normality = FALSE,
R+                      plugin_constant_joint = 4)
R> dlg_object <- dlg(lg_object, grid = grid)
R>
R> # Make a dependence map using default setup
R> corplot(dlg_object)
```

We may tweak the appearance of our dependence map by passing further arguments to `corplot()`. Some of the options are demonstrated in the code chunk below, in which we, for example, superimpose the observations (by setting `plot_obs = TRUE`) as well as preventing the the estimated local correlations from being plotted in areas without data. The latter option is available through the argument `plot_thres`, which works by calculating a bivariate kernel density estimate $\tilde{f}(x_1, x_2)$ for the pair of variables in question, and only allowing $\widehat{\rho}(x_1, x_2)$ to be plotted if $\tilde{f}(x_1, x_2) / \max \tilde{f}(\cdot) >$ `plot_thres`. Adding layers to a dependence map using the ordinary **ggplot2** syntax works as well, which we demonstrate in Figure 5 by changing the **ggplot2** theme.

The plotting function works in the same way when plotting the local partial correlations returned by `partial_cor()`, and the arguments of `corplot()` are summarized in Table 7.

```
R> corplot(dlg_object1,
R+         plot_obs = TRUE,
R+         plot_thres = 0.01,
R+         plot_labels = FALSE,
R+         alpha_point = 0.3,
R+         main = "",
R+         xlab = "",
R+         ylab = "") +
R+   theme_classic()
```

Conclusion

The statistical literature has seen a number of applications of local Gaussian approximations in the last decade, covering several topics in dependence modeling and inference, as well as the estimation of multivariate density and conditional density functions. In this paper, we demonstrate the implementation of these methods i the R programming language using the **lg**-package, as well as the graphical representation of the estimated local Gaussian correlation. The package is complete, in the sense that all major methods that have been published within this framework is now easily accessible to the practitioner. The package is also designed with a modular infrastructure that allows future methodological developments using local Gaussian approximations to be easily added to the package.

Acknowledgements

The author gratefully acknowledges the constructive comments made by two anonymous referees.

Argument	Explanation	Default value
`lg_object`	The lg-object created by `lg_main()`	
`grid`	The evaluation points for the LGPC, **must** be a data frame or matrix having 2 columns	`NULL`
`cond`	Vector with fixed values for $X^{(3)}$	`NULL`
`level`	Significance level for approximated confidence bands	0.95

Table 5: Arguments to the `partial_cor()-function`

Argument	Explanation	Default value
`lg_object`	The lg-object created by `lg_main()`	
`h`	The function $h(\cdot)$ in (16)	`function(x) x^2`
`n_rep`	Number of bootstrap replicates	500

Table 6: Arguments to the `ci_test()-function`

Argument	Explanation	Default value
`dlg_object`	The object created by `dlg()` or `partial_cor()`	
`pair`	Which pair to plot if more than two variables	1
`gaussian_scale`	Logical. Plot on the marginal st. normal scale?	`FALSE`
`plot_colormap`	Logical. Plot the colormap?	`TRUE`
`plot_obs`	Logical. Superimpose observations?	`FALSE`
`plot_labels`	Logical. Plot labels on dependence map?	`TRUE`
`plot_legend`	Logical. Add legend?	`FALSE`
`plot_thres`	Threshold for plotting the estimated LGC	0
`alpha_tile`	Transparency of color tiles	0.8
`alpha_point`	Transparency of points	0.8
`low_color`	Color representing $\widehat{\rho} = -1$	`"blue"`
`high_color`	Color representing $\widehat{\rho} = +1$	`"red"`
`break_int`	Break interval for color coding	0.2
`label_size`	Size of labels in plot	3
`font_family`	Font family for labels	`"sans"`
`point_size`	Size of points, if plotted	`NULL`
`xlim`, `ylim`	Axis limits	`NULL`
`xlab`, `ylab`	Axis labels	`NULL`
`rholab`	Title of legend	`NULL`
`main`, `subtitle`	Title and subtitle of plot	`NULL`

Table 7: Arguments to the `corplot()`-function

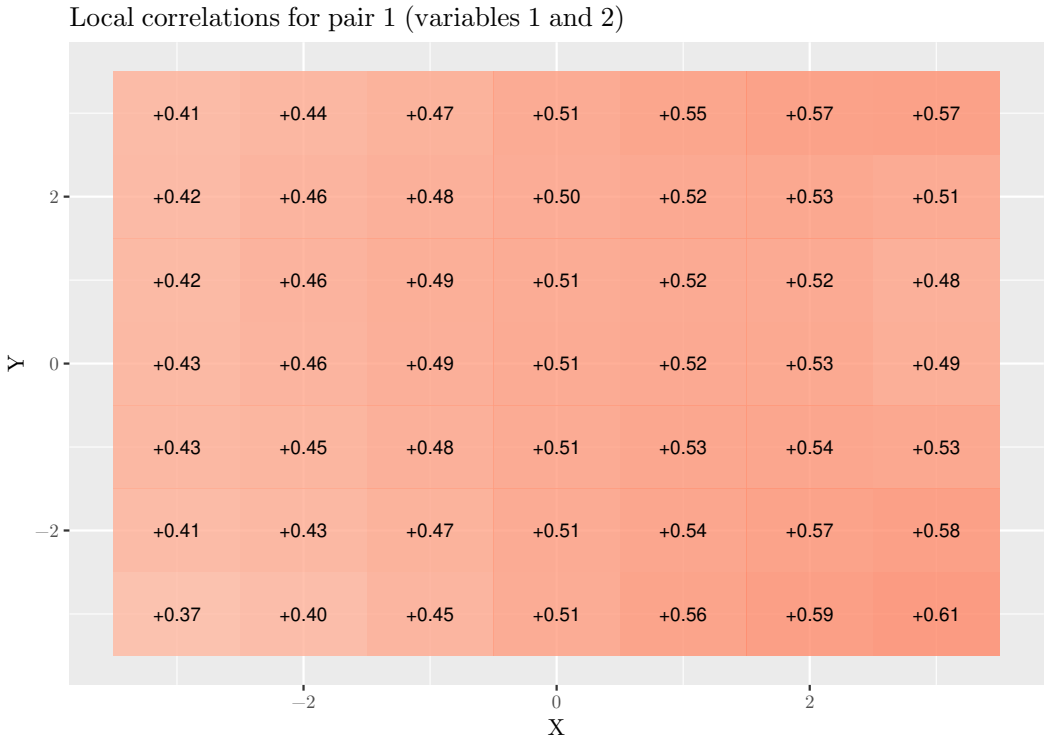


Figure 4: Dependence map produced by `corplot()` using the default configuration

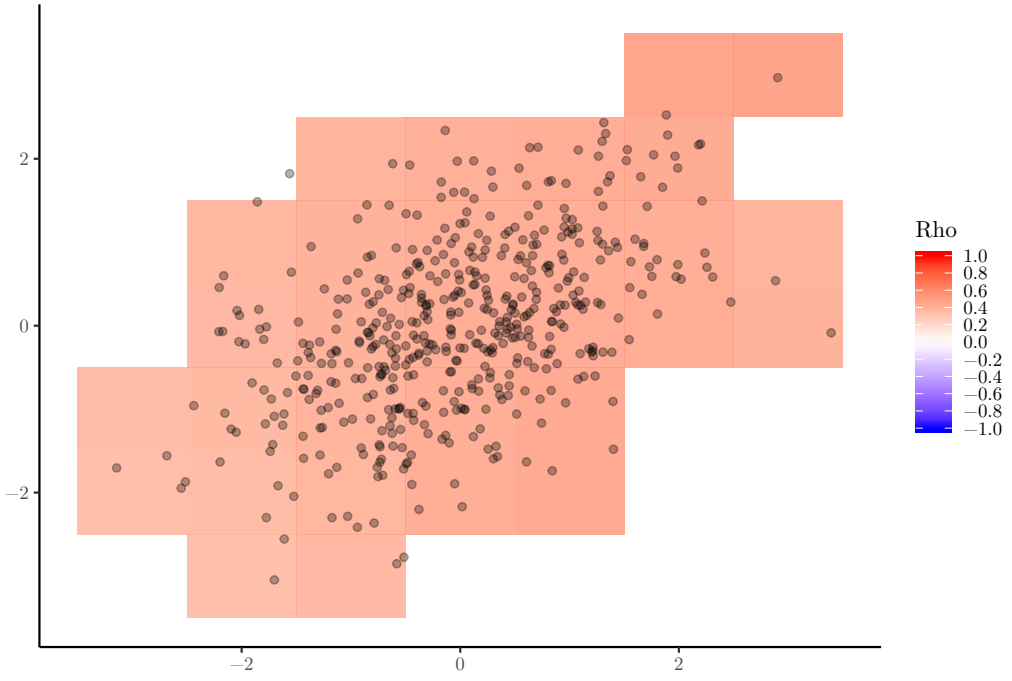


Figure 5: Dependence map produced by tweaking the arguments of `corplot()`

Bibliography

S. M. Bache and H. Wickham. *magrittr: A Forward-Pipe Operator for R*, 2014. URL https://CRAN.R-project.org/package=magrittr. R package version 1.5. [p10]

G. D. Berentsen and D. Tjøstheim. Recognizing and visualizing departures from independence in bivariate data using local Gaussian correlation. *Statistics and Computing*, 24(5):785–801, 2014. URL https://doi.org/10.1007/s11222-013-9402-8. [p1, 3, 8, 11, 12]

G. D. Berentsen, T. Kleppe, and D. Tjøstheim. Introducing localgauss, an R-package for estimating and visualizing local Gaussian correlation. *Journal of Statistical Software*, 56(12):1–18, 2014. URL https://doi.org/10.18637/jss.v056.i12. [p1, 10, 14]

Datastream, 2018. [p3, 10]

K. J. Forbes and R. Rigobon. No contagion, only interdependence: measuring stock market co-movements. *The Journal of Finance*, 57(5):2223–2261, 2002. URL https://doi.org/10.1111/0022-1082.00494. [p3]

A. Genz, F. Bretz, T. Miwa, X. Mi, F. Leisch, F. Scheipl, and T. Hothorn. *mvtnorm: Multivariate Normal and t Distributions*, 2018. URL https://CRAN.R-project.org/package=mvtnorm. R package version 1.0-8. [p15]

C. W. Granger. Testing for causality: a personal viewpoint. *Journal of Economic Dynamics and control*, 2: 329–352, 1980. URL https://doi.org/10.1016/0165-1889(80)90069-X. [p14]

T. Hayfield and J. S. Racine. Nonparametric econometrics: The np package. *Journal of Statistical Software*, 27(5), 2008. URL https://doi.org/10.3929/ethz-b-000073064. [p12]

N. Hjort and M. Jones. Locally parametric nonparametric density estimation. *Annals of Statistics*, 24(4): 1619–1647, 1996. URL https://doi.org/10.1214/aos/1032298288. [p2, 3]

R. A. Johnson and D. W. Wichern. *Applied Multivariate Statistical Analysis, Sixth Edition*. Pearson Education Iternational, 2007. [p4]

L. A. Jordanger. *localgaussSpec: An R-package for local Gaussian spectral analysis*, 2018. URL https://github.com/LAJordanger/localgaussSpec. R package. [p1]

L. A. Jordanger and D. Tjøstheim. Nonlinear spectral analysis: A local gaussian approach. *Journal of the American Statistical Association*, pages 1–55, 2020. URL https://doi.org/10.1080/01621459.2020.1840991. [p1]

H. R. Kunsch. The jackknife and the bootstrap for general stationary observations. *The Annals of Statistics*, 17(3):1217–1241, 1989. URL https://doi.org/10.1214/aos/1176347265. [p12]

V. Lacal and D. Tjøstheim. Local Gaussian autocorrelation and tests of serial dependence. *Journal of Time Series Analysis*, 38(1):51–71, 2017. URL https://doi.org/10.1111/jtsa.12195. [p1, 3, 11]

V. Lacal and D. Tjøstheim. Estimating and testing nonlinear local dependence between two time series. *Journal of Business & Economic Statistics*, pages 1–13, 2018. URL https://doi.org/10.1080/07350015.2017.1407777. [p1, 3, 11, 12]

H. Otneim. *Multivariate and conditional density estimation using local Gaussian approximations*. PhD thesis, University of Bergen, 2016. URL https://hdl.handle.net/1956/15333. [p8]

H. Otneim. *lg: Locally Gaussian Distributions: Estimation and Methods*, 2019. URL https://CRAN.R-project.org/package=lg. R package version 0.4.0. [p1]

H. Otneim and D. Tjøstheim. The locally Gaussian density estimator for multivariate data. *Statistics and Computing*, 27(6):1595–1616, 2017. URL https://doi.org/10.1007/s11222-016-9706-6. [p1, 3, 4, 8, 10]

H. Otneim and D. Tjøstheim. Conditional density estimation using the local Gaussian correlation. *Statistics and Computing*, 28(2):303–321, 2018. URL https://doi.org/10.1007/s11222-017-9732-z. [p1, 4, 10]

H. Otneim and D. Tjøstheim. The locally Gaussian partial correlation. *Journal of Business & Economic Statistics*, pages 1–33, 2021. URL https://doi.org/10.1080/07350015.2021.1886107. [p1, 5, 6, 14]

D. N. Politis and J. P. Romano. The stationary bootstrap. *Journal of the American Statistical Association*, 89(428):1303–1313, 1994. URL https://doi.org/10.1080/01621459.1994.10476870. [p12]

B. W. Silverman. *Density Estimation for Statistics and Data Analysis*. Chapman and Hall, London, 1986. URL https://doi.org/10.1201/9781315140919. [p1, 8]

B. Støve, D. Tjøstheim, and K. Hufthammer. Using local Gaussian correlation in a nonlinear re-examination of financial contagion. *Journal of Empirical Finance*, 25:785–801, 2014. URL https://doi.org/10.1016/j.jempfin.2013.11.006. [p1, 3, 13]

D. Tjøstheim and K. O. Hufthammer. Local Gaussian correlation: A new measure of dependence. *Journal of Econometrics*, 172(1):33–48, 2013. URL https://doi.org/10.1016/j.jeconom.2012.08.001. [p1, 2, 10]

H. Wickham. *ggplot2: Elegant Graphics for Data Analysis*. Springer-Verlag New York, 2016. ISBN 978-3-319-24277-4. URL https://doi.org/10.1007/978-0-387-98141-3. [p14]

H. Wickham, R. François, L. Henry, and K. Müller. *dplyr: A Grammar of Data Manipulation*, 2018. URL https://CRAN.R-project.org/package=dplyr. R package version 0.7.6. [p10]

Håkon Otneim
Department of Business and Management Science,
NHH Norwegian School of Economics,
Helleveien 30, 5045 BERGEN
Norway
ORCID: 0000-0002-6004-0237
hakon.otneim@nhh.no