A comparison of the local Gaussian correlation and the local dependence function

Christopher James Brokstad

June 2022

Master's thesis in Actuarial Science,
Department of Mathematics,
University of Bergen

Abstract

Correlation is a method to measure the relation between two or more variables. In this thesis, a method of measuring correlation and a method of measuring dependence are used. These two methods, are the local Gaussian correlation and the local dependence function. The goal was to build a bridge between these two measurements. The hypothesis is that for a bivariate normal density, both methods will locally approximate the densities correlation coefficient. The local dependence function is a measure of dependence and it has to be transformed to give the function of the correlation. However, there was not a clear connection between the two methods. Instead of the local dependence function, the precision matrix was utilised. The precision matrix provides an opportunity to find the correlation coefficient locally for the bivariate normal density. Thus, a bridge between the local Gaussian correlation and the correlation estimate from the precision matrix can be built. To obtain the correlation estimate from the precision matrix, it has to be transformed to its inverse, the covariance matrix. However, while a connection was made, some details remain unclear. This is observed, with the local Gaussian correlation being defined for the range of the density. While the estimated correlation for the precision matrix has areas that are undefined for certain densities. The function of the estimated correlation from the precision matrix is discussed, to explain why some areas are undefined and why the estimate takes the form it does for different densities. The two methods also produced differing correlation estimates for given areas. In this thesis, the same set of test case densities are used, as the ones in the introductory paper for the local dependence function (Jones, 1996), and its preliminary paper (Doksum et al., 1994). As the local Gaussian correlation is an empirical method of measuring correlation, it needs data. For this thesis there is no observed data, therefore instead simulated data are used in the analyses. The correlation estimate from the precision matrix, on the other hand requires the densities to be known. To further explore the precision matrix's correlation estimate, two novel methods are explored; the Box-Cox transformation and the Gaussian kernel estimates, but they require further work. In conclusion, while a bridge is constructed between the local Gaussian correlation and the precision matrix's correlation estimate, more work is needed to establish a clearer connection.

Acknowledgments

I would like to thank my supervisor Professor Hans J. Skaug for being patient, interesting discussion and help guiding me throughout this Master's thesis. I would like to thank senior consultant Kristine Lysnes for answering my questions related and unrelated to the master's project. I would also like to thank my friends and family for their support during the writing of this thesis.

Table of Contents

1 Introduction **1**

2 Correlation **3**

3 Local Gaussian Correlation **7**

4 Local Dependence Function **10**

5 Experiments **11**

6 Variance **23**

7 Results and Discussion **26**
7.1 Pear 26
7.2 Twisted Pear 30
7.3 Cauchy 31
7.4 Transformed Normal Density 33

8 Box-Cox Transformation **34**

9 Kernel Smoother **40**

10 Kernel Estimates **44**
10.1 Contours 44
10.2 Double Derivatives 49

11 Conclusions and Future Perspective **58**

Appendices **65**

Appendix A Example of the Precision matrix's Correlation Function **65**

Appendix B Examples of C++ Code **68**

Appendix C R-code **70**

List of Figures

1 Simulated datapoints from the function $Y = \sin(3x) + \epsilon$ 4
2 The Pear density, simulated observations, precision matrix's correlation estimate and the local Gaussian correlation map. . 16
3 The twisted Pear density, simulated observations, precision matrix's correlation estimate and the local Gaussian correlation map. 18
4 The Cauchy density, simulated observations, precision matrix's correlation estimate and the local Gaussian correlation map. 20
5 The transformed normal density, simulated observations, precision matrix's correlation estimate and the local Gaussian correlation map. 22
6 Variance estimates for the Pear 24
7 Variance estimates for the twisted Pear 24
8 Variance estimates for the Cauchy density 25
9 Variance estimates for the transformed normal density 25
10 Outlier variance estimates for the Pear density 28
11 The Pear densities $\hat{\sigma}$ negative values 29
12 The correlation estimates from the local dependence function for the Cauchy distribution 32
13 Correlation estimates using the precision matrix for the Box-Cox transformed Cauchy density for 2 λ values 35
14 Box-Cox transformed Cauchy density for 5 different λ values. . 37
15 Correlation estimates using the precision matrix for the Box-Cox transformed Cauchy density for 5 different λ values 39
16 Four examples of kernel functions 43
17 Bivariate Gaussian kernel estimates for the Pear, twisted Pear, Cauchy and transformed normal density. 46
18 Bivariate Gaussian kernel estimates of the transformed normal density for 3 different h values 48
19 The correlation estimates from the precision matrix for the bivariate Gaussian kernel estimates of the densities. 53
20 Histograms of $\hat{\rho}(x, y)$ estimates for 5 different h levels for bivariate Gaussian kernel estimates of the Cauchy density 54
21 Contours of $\hat{\rho}(x, y)$ for the bivariate Gaussian kernel estimates of Cauchy density at 5 different h values. 56

22 Undefined areas for $\hat{\rho}(x, y)$ for the bivariate Gaussian kernel estimates of the Cauchy density for 2 different h values. 57
23 Local Gaussian correlation maps of a bivariate normal density for 4 different sample sizes. 61

Table of Abbreviations

AISB - Asymptotic Integrated Variance
AIV - Asymptotic Integrated Square Bias
AMISE - Asymptotic Mean Integrated Square Error
ARMA - Auto-regressive Moving Average
i.i.d - Independent and Identically Distributed
IQR - InterQuartile Range
MISE - Mean Integrated Square Error
MSE - Mean Square Error

1 Introduction

Correlation is the measure of the relation between two or more variable and is a commonly studied subject as it can be important to understand how variables in a sample relate to each other. The local Gaussian correlation, is a measure of correlation, and measures the correlation locally for areas of the data. It does this by approximating areas to Gaussian densities and the correlation estimates comes from these approximations. My introduction to local Gaussian correlation was through my bachelor thesis (Brokstad, 2020). This bachelor thesis provided a general overview of the local Gaussian correlation. In addition, to using the local Gaussian correlation to investigate the relation between COVID-19 data and its impact on the index stock for Oslo Børs. The local Gaussian correlation's primary use has been for investigating the stock market. An example of this is the paper by Støve et al. (2014), where they use the local Gaussian correlation to investigate different financial crises. Another example of this is Nguyen et al. (2020) that looks at correlation before and after economic crises. Some work has also been conducted in other areas of statistics. Such as Jordanger and Tjøstheim (2021) paper where they use the local Gaussian correlation to inspect upper and lower tails of spectral densities. Similarly, Berentsen et al. (2014) used the local Gaussian correlation to examine copula models and their characteristics.

The other method used is the local dependence function. The local dependence function was introduced in Jones (1996). The local dependence function is a measure of dependence. To be able to find an estimate of the correlation using it, it has to be transformed. The areas of use for the dependence function are less specific than the area of use for the local Gaussian correlation. There are papers such as Gupta et al. (2010) that proved, when the local dependence function is always equal or more than 0 the density is totally positive of the second order. Jones and Koch (2003) introduces dependence maps, with the goal of making the local dependence function easier to interpret.

The goal of this thesis is try and build a bridge between the local Gaussian correlation and the local dependence function. This bridge can be found if they can both locally approximate the same correlation as the correlation coefficient for a bivariate normal density. As there was not, a clear connection between the local Gaussian correlation and the local dependence function.

Instead, the precision matrix was utilised over the local dependence function. The precision matrix is the inverse of the covariance matrix. In order to get the correlation coefficient, the precision matrix has to be transformed to the covariance matrix. From the covariance matrix, it is possible to get the correlation coefficient. As the estimated correlation obtained from the precision matrix can locally approximate to the correlation coefficient for the bivariate normal density, the bridge between it and the local Gaussian correlation can be built. However, some details remain unclear. One of the problems is that the local Gaussian correlation is defined over the entire area of the density, while the estimated correlation from the precision matrix has areas that are undefined. In addition, there are certain areas for the different densities that the local Gaussian correlation and the estimated correlation from the precision matrix give differing estimates. The reason for why they may differ and why there are undefined areas is discussed further in this thesis. For the implementation of the local Gaussian correlation the package **lg** (Otneim, 2019) for the programming language **R** is used. For finding the precision matrix the package **TMB**, is used (Kristensen et al., 2016). There is also the implementation of the Box-Cox transformation and the bivariate Gaussian kernel estimate in relation to the precision matrix. However they are only briefly used so they are not as extensively studied as the regular precision matrix. Finally a discussion about the overall results and what problems remain with the precision matrix's estimated correlation.

2 Correlation

Correlation is the interaction between two variables and how they affect each other. Generally the most used form of correlation is Pearson's ρ that Pearson (1896) introduced, and its formal definition is

$$\rho = \frac{E(XY) - E(X)E(Y)}{\sigma_X \sigma_Y}, \tag{1}$$

where $E(XY)$ is the expected value for X times Y. $E(X)$ is the expected value for X and $E(Y)$ is the expected value for Y. σ_X and σ_Y are the standard deviations for the X and Y variables, respectively. The ρ value is within the range of $[-1, 1]$. Positive ρ values indicate a positive correlation between the two variables. Positive correlation is the positive association between the variables, so as X increases in value, Y increases in value as well. For the negative correlation, as one of the variables increases in value the other variable will decrease in value. The strength of this association is described by how close or equal ρ is to 1 or -1. A ρ value of 0.9 indicates a stronger correlation then a ρ value of 0.2. For $\rho = 0$, it indicates that there is no linear correlation between the variables.
Although ρ has been given for its population form, it can also be approximated empirically by replacing the expected value and variance with their empirical forms. The empirical forms are the sample mean and sample variance. Then for a given dataset with the observations $(X_1, Y_1), ..., (X_n, Y_n)$, the mean for the observations X and Y are given by $\bar{X}$ and $\bar{Y}$, the sample standard deviations are given by s_X and s_Y. The empirical version of Pearson's ρ is

$$r = \frac{\sum_{i=1}^{n}(X_i - \bar{X})(Y_i - \bar{Y})}{\sqrt{\sum_{i=1}^{n}(X_i - \bar{X})^2}\sqrt{\sum_{i=1}^{n}(Y_i - \bar{Y})^2}}. \tag{2}$$

For both the population variant and the empirical variant, a singular value of the correlation ρ is found for all of the data.

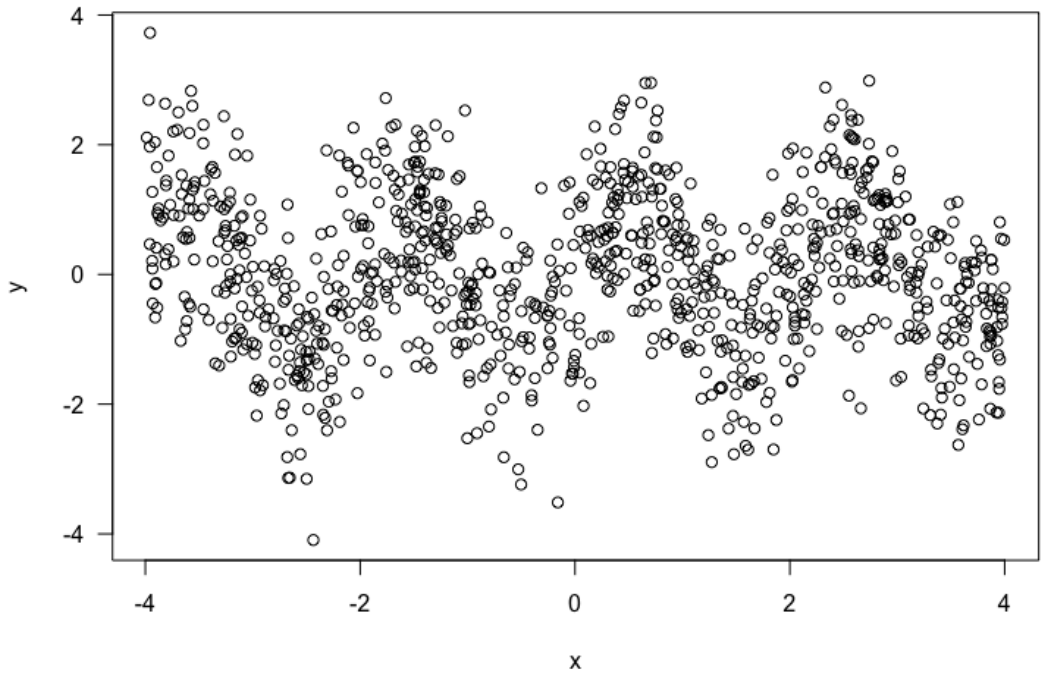


Figure 1: 1 000 simulated datapoints for the function $Y = \sin(3x) + \epsilon$, where ϵ is independent and identically distributed noise.

While Pearson's ρ gives us a good overview of the relation between the x and y variable, it does have its weak points. For certain datasets we are more interested in specified areas correlation than the whole data's correlation coefficient There are at least two big disadvantages for Pearsons ρ. One of the two problems is that ρ is sensitive to outliers. For example for a hypothetical dataset of $(1, 3), (4, 6), (5, 2), (13, 9)$, the correlation is $r = 0.8$. If we removed $(13, 9)$ from the dataset, then instead $r = 0.038$. Obviously the proposed hypothetical dataset is small so the ρ value is more susceptible to changes for this dataset, than for a dataset of size $n = 1000$ for example. But even for that larger hypothetical dataset, the ρ value would still be sensitive to outliers. The reason is that both the formula for the empirical (2) and the population variant (1) include measurements of the mean. Outlier values will have bigger impacts on the mean value then data within the normal range.

The second problem is the fact that Pearson's ρ is set up to measure linear correlation between the two variables. Thus for nonlinear correlation, it will not describe the association between the two variables well. An example discussed in Tjøstheim et al. (2022) of this issue that ρ is for $Y = X^2 + \epsilon$. The reason Pearson breaks down for this example is that it is not able to capture the nonlinear association between the two variables. As Y and X will be negatively correlated for negative values of X. Whereas for positive values of X there will be a positive correlation between the two variables. Similarly, another example of nonlinearity that ρ will under perform on is $Y = \sin(3x)+\epsilon$ as seen in figure 1 where ϵ is identical independent distributed (i.i.d) noise.

There are methods to reduce susceptibility for outliers for ρ. An easy solution is to exclude or remove outliers from the dataset. There are other methods such as transforming or altering the formula for how ρ is calculated, to make it less susceptible to this problem. Building on the previous example, a quick and dirty solution would be to only use data points within a given percentile. There are also more complex alters such as Spearman's or Kendall's methods which are also outlined in Tjøstheim et al. (2022). Spearman's rank correlation looks at the observations by ranking. While Kendall's correlation coefficient τ looks at the difference between concordant pairs and discordant pairs. Concordant pairs are pairs where for $i < j$, $X_i > X_j$ and $Y_i > Y_j$ or $X_i < X_j$ and $Y_i < Y_j$. A discordant pair would entail either X_i or Y_i having a smaller value than either X_j or Y_j. However both of these variations will still have problems with describing the relation of nonlinear data.

So far, the paper has mainly presented potential weaknesses for ρ. Yet it still is the most commonly used method of measuring correlation. As previously stated ρ gives an overview of the whole dataset's relation unlike the local correlations which is discussed later. In addition, ρ is very simple to compute as its components are simple to find, such as the standard deviation and the mean. There are also other benefits of ρ such as being able to describe the relation of X_t and X_{t+h} for linear time series models such as Autoregressive-moving average (ARMA) models (Tjøstheim et al., 2022). Another group of models that ρ is important for, is the multidimensional normal density and similar densities from the same family. As one of the parameters used to describe the density is actually Pearson's ρ which appears

in the covariance matrix. ρ is also useful in linear regression models. For the form of $Y = \alpha + \beta X + \epsilon$, where ϵ is zero-mean i.i.d noise. Then β can be given as (Tjøstheim et al., 2022):

$$\beta = \rho \frac{\sigma_Y}{\sigma_X}. \tag{3}$$

For the linear regression model if σ_X, σ_Y and ρ are unknown, we can instead use their empirical counterparts $\{s_X, s_Y, r\}$ to get a similar result.

3 Local Gaussian Correlation

The local Gaussian correlation is method of measuring correlation by locally approximating a Gaussian density to a dataset. For a given datapoint $z = (x, y)$. The approximated density will have the running variables (v_1, v_2). It will also have $\mu_1(z)$ and $\mu_2(z)$ as the local mean vectors and $\sigma_1^2(z)$ and $\sigma_2^2(z)$ will be the local variance functions. $\rho(z)$ is the covariance value for the density. Then for the datapoint (x, y), the approximated density is

$$\begin{aligned}
&\psi(v,\mu_1(z),\mu_1(z),\sigma_1^2(z),\sigma_2^2(z),\rho(z)) \\
&=\frac{1}{2\pi\sigma_1(z)\sigma_2(z)\sqrt{1-\rho^2(z)}} \\
&\quad\times\exp\left[-\frac{1}{2}\frac{1}{1-\rho^2(z)}\left(\frac{(v_1-\mu_1(z))^2}{\sigma_1^2(z)}\right.\right. \\
&\quad\left.\left.-2\rho(z)\frac{(v_1-\mu_1(z))(v_2-\mu_2(z))}{\sigma_1(z)\sigma_2(z)}+\frac{(v_2-\mu_2(z))^2}{\sigma_2^2(z)}\right)\right].
\end{aligned} \tag{4}$$

From the approximation, the most important parameter is $\rho(z)$. It is from the parameter $\rho(z)$, the local Gaussian correlation gets its name from. The dataset itself does not have to have a Gaussian density, as the local Gaussian correlation will approximate a local area of the dataset to a Gaussian density. If the dataset is actually Gaussian distributed. Then for a given density f, the local Gaussian density will approximate to f's density for any value of f. Although for the Gaussian data, as the local Gaussian correlation approximates the datapoints in a given area there can be multiple possible densities that it could approximate to (Tjøstheim et al., 2013). Thus to find the best fit for the approximation, a penalty function is used. The penalty function utilised for the local Gaussian correlation is a locally weighted Kullback-Leibler distance between f and ψ. As the local Gaussian correlation is built on the work of Hjort and Jones (1996). Many of the functions and equations for the local Gaussian correlation's penalty function, are derived from that paper. In that paper (Hjort and Jones, 1996), they present how to find a local dependence measurement using local likelihood. However unlike the local Gaussian correlation they do not specify what family the approximating density $\hat{f}$ will have. While for the local Gaussian correlation the $\hat{f}$ takes the form of ψ.

The penalty function is given as

$$q = \int K_h(v-z)[\psi(v,\theta(z)) - \log\psi\{v,\theta(z)\}f(v)]dv. \tag{5}$$

In the penalty function K_h is a product kernel containing

$$K_h(v-z) = (h_1h_2)^{-1}K(h_1^{-1}(v_1-x))K(h_2^{-1}(v_2-y)). \tag{6}$$

For K_h, the bandwidth is $h = (h_1, h_2)$ (Tjøstheim et al., 2013). Also

$$\theta(z) = (\mu_1(z), \mu_2(z), \sigma_1^2(z), \sigma_2^2(z), \rho(z)). \tag{7}$$

Then to minimize $\theta(z)$ for the penalty function

$$\begin{aligned} &\int K_h(v-z)\frac{\partial}{\partial\theta_j}log(\psi(v,\theta(z)))[f(v)-\psi(v,\theta(z))]dv = 0 \\ &j = 1,...,5. \end{aligned} \tag{8}$$

For the density f, where $(X_1,...,X_n)$ and $(Y_1,...,Y_n)$ are i.i.d observations and $Z_i = (X_i, Y_i)$. To get an estimate for $\theta(z)$ for a fixed z, we maximize the local log likelihood (Tjøstheim et al., 2013).

$$\begin{aligned} L(Z_1,...,Z_n,\theta_b(z)) =& n^{-1}\sum_i K_h(Z_i-z)\log\psi(Z_i,\theta_h(z)) \\ &- \int K_h(v-z)\psi(v,\theta_h(z))dv, \end{aligned} \tag{9}$$

K_h is a kernel function as described previously for the penalty function (6). From here, one can obtain the derivative $\partial L/\partial\theta_j$ (Tjøstheim et al., 2013),

$$\begin{aligned} \frac{\partial L}{\partial\theta_j} =& n^{-1}\sum_i K_h(Z_i-z)\frac{\partial}{\partial\theta_j}\log\{\psi(Z_i,\theta_h(z))\} \\ &- \int K_h(v-z)\frac{\partial}{\partial\theta_j}\log\{\psi(v,\theta_h(z))\}\psi(v,\theta_h(z))dv \\ &\to \int K_h(v-z)\frac{\partial}{\partial\theta_j}\log\{\psi(v,\theta_h(z))\}[f(v)-\psi(v,\theta_h(z))]dv, \end{aligned} \tag{10}$$

$\partial L/\partial\theta_j$ is found by using the law of large numbers, and the assumption that

$$E[K_h(Z_i-z)\log\psi(z_i,\theta_b(z))] < \infty. \tag{11}$$

Then if $\partial L/\partial\theta_j = 0$, we can find the maximum likelihood estimates $\widehat{\theta_b}$ (Tjøstheim et al., 2013). From minimizing $\theta(z)$ for the penalty function we can get

$$\int K_h(v-z)\alpha(v)dv = \alpha(z) + \frac{1}{2}\sum_{i=1}^{2}\sum_{j=1}^{2}\sigma^2_{K_{i,j}}\frac{\partial^2\alpha(z_i,z_j)}{\partial z_i\partial z_j}b_ib_j + o(b^Tb). \quad (12)$$

In the equation, $\sigma^2_{K_{i,j}} = \int c_1^i c_2^j K(c_1,c_2)dc_1dc_2$. Where $c = A_i^{-1}(z-a_i)$. $A_i = \Sigma_i^{1/2}$, where Σ is the covariance matrix and $a_i = \mu_i$, for further explanation see Tjøstheim et al. (2013). The important part is

$$\alpha(v) = \frac{\partial}{\partial\theta_j}log\{\psi(v,\theta_h(z))\}[f(v) - \psi(v,\theta_h(z))], \quad (13)$$

for $\theta^T = [\theta_1,...,\theta_5]$, which means that as $b \to 0$ then at the same time $f(z) - \psi(z,\theta_h(z)) \to 0$. For $f_1(z) = f_2(z)$ within a neighbourhood then for a certain bandwidth b_0 it is possible to get $\theta(f_1,z) = \theta(f_2,z)$ (Tjøstheim et al., 2013). For creating the subsequent figures, the **R** package **lg** (Otneim, 2019) is used. The reason is it finds the local Gaussian correlation estimates for datasets. In addition as there is no real data, the data is simulated using the Markov chain Monte Carlo method.

4 Local Dependence Function

Another method for measuring local dependence is described in Jones (1996) paper "The local dependence function". The local dependence function Jones describes takes the form of

$$\gamma(x, y) = \frac{\partial^2}{\partial x \partial y} \log(f(x, y)). \tag{14}$$

As the local dependence function is the double derivative of the log density, it is not restricted to being values within the range of -1 and 1. This is unlike Pearson's ρ or the local Gaussian correlation. In Jones's paper, they also present multiple properties for the function $\gamma(x, y)$ (Jones, 1996). One of those properties is that $\gamma(x, y)$ is finite everywhere. A different property is that if X and Y are independent then and only then is $\gamma = 0$. Another important property is that, when we are looking at $\gamma(x, y)$ for a bivariate normal density. Then the γ function is constant(Jones, 1996) and should have the form of

$$\gamma(x, y) = \frac{\rho}{(1 - \rho^2)}. \tag{15}$$

Another property is that for a stronger correlation between x and y, the $\gamma(x, y)$ function will begin to increase exponentially towards $\pm\infty$. This can be seen in equation (15). As ρ for the bivariate normal density goes towards ± 1 then the denominator of the equation (15) will go towards 0.

5 Experiments

As the local dependence function is defined as a function of ρ for bivariate normal densities. Then by solving the equation, ρ can instead be described as a function of γ.

$$\begin{aligned} \gamma(x,y) &= \frac{\rho}{(1-\rho^2)} \\ \gamma(x,y)(1-\rho^2) &= \rho \\ \rho^2\gamma(x,y) + \rho + \gamma(x,y) &= 0. \end{aligned} \tag{16}$$

From the inverse transformation we get the two possible values for ρ,

$$\begin{aligned} \rho_1 &= \frac{-1+\sqrt{1+4\gamma(x,y)^2}}{2\gamma(x,y)} \\ \rho_2 &= \frac{-1-\sqrt{1+4\gamma(x,y)^2}}{2\gamma(x,y)}. \end{aligned} \tag{17}$$

From experimentation using real ρ values within the range $[-1,1]$ and using the $\gamma(x,y)$ function. ρ_1 returns the real ρ values. This is also discussed in the introductory paper for the local dependence function (Jones, 1996). ρ_2 on the other hand will return values that are outside the range of ρ's defined area of $[-1,1]$. However while Pearsons ρ can be equal to 0, $\rho_1 \neq 0$. This can be seen by using equation (17), and (15). If $\rho = 0$ then the γ function from equation (15) is also equal to 0. The problem occurs in equation (17) as ρ_1 has γ in the denominator and we cannot divide by 0. Although $\rho_1 \neq 0$, it does approach the value 0 as the γ function goes towards 0 from both the positive and negative sides. This is shown by using L'Hôpital's rule

$$\begin{aligned} f(\gamma) &= -1+\sqrt{1+4\gamma^2} & g(\gamma) &= 2\gamma \\ f(0) &= 0 & g(0) &= 0 \\ f'(\gamma) &= \frac{4\gamma}{\sqrt{1+4\gamma^2}} & g'(\gamma) &= 2. \end{aligned}$$

Then the resulting equation is

$$\lim_{\gamma\to 0} \frac{1}{2}\frac{4\gamma}{\sqrt{1+4\gamma^2}} = 0. \tag{18}$$

Although Jones (1996) states that "it is constant if f is the bivariate normal density, and then takes the value $\rho/(1-\rho^2)$ where ρ is the Pearson correlation coefficient;". The statement can be proven to be false. By deriving the mixed partial derivative of the logarithm of a bivariate normal density f, the γ function takes the form of

$$\gamma(x,y) = \frac{\partial^2}{\partial x \partial y} \log(f(x,y)) = \frac{\rho}{1-\rho^2} \times \frac{1}{\sigma_x \sigma_y}. \tag{19}$$

The reason that the γ function takes the form of the above equation (19) instead of the previously given form in the equation (15), is because the bivariate normal density is defined as

$$\begin{aligned} f(x,y) = & \frac{1}{2\pi\sigma_x\sigma_y\sqrt{1-\rho^2}} \\ & \times \exp\left[-\frac{1}{2}\frac{1}{1-\rho^2}\left(\frac{(x-\mu_x)^2}{\sigma_x^2} \right.\right. \\ & \left.\left. - 2\rho\frac{(x-\mu_x)(y-\mu_y)}{\sigma_x\sigma_y} + \frac{(y-\mu_y)^2}{\sigma_y^2} \right) \right]. \end{aligned} \tag{20}$$

From the definition (20), the only part that contains both a x and a y variable is the part of

$$-2\rho\frac{(x-\mu_x)(y-\mu_y)}{\sigma_x\sigma_y}. \tag{21}$$

Thus when we try to find the local dependence function for the density, the denominator with σ_x and σ_y is not derived away. Just to further exemplify this, if equation (19) and equation (17) are used to check that $\rho_1 = \rho$. Then γ for a bivariate normal density with $\sigma_x = \sigma_y = 1$ will take the form given in equation (15) and all possible ρ_1 values will be equal to ρ. While if one or both of the σ are different than 1 then $\rho_1 \neq \rho$. This conclusion is further supported by Jones (1998), that it is for a standard bivariate normal density that γ takes the form given in equation (15). Using the given definition of the local dependence function for the Pear which is a transformed normal density, the γ function is

$$\gamma(x,y) = \frac{\rho}{1-\rho^2} \times \frac{3x^2}{\sigma_x\sigma_y}. \tag{22}$$

The Pear density was included in Doksum et al. (1994) and can be seen in figure 2a. For the transformed normal density seen in figure 5a, the γ takes

the form of

$$\gamma(x,y) = \frac{\rho}{1-\rho^2} \times \frac{y}{\sigma_x \sigma_y}. \tag{23}$$

As shown there are some problems with trying to build a connection between the local dependence function and the local Gaussian correlation. Therefor my supervisor Professor Skaug proposed the use of the precision matrix instead. The precision matrix for the bivariate normal density can be defined as

$$\begin{bmatrix} \frac{1}{(1-\rho^2)\sigma_x^2} & \frac{-\rho}{(1-\rho^2)\sigma_x\sigma_y} \\ \frac{-\rho}{(1-\rho^2)\sigma_x\sigma_y} & \frac{1}{(1-\rho^2)\sigma_y^2} \end{bmatrix} = \begin{bmatrix} -\frac{\partial^2}{\partial x^2} logf(x,y) & -\frac{\partial^2}{\partial x \partial y} logf(x,y) \\ -\frac{\partial^2}{\partial x \partial y} logf(x,y) & -\frac{\partial^2}{\partial y^2} logf(x,y) \end{bmatrix}. \tag{24}$$

The precision matrix's inverse is the covariance matrix, which for a bivariate normal density is given as

$$\begin{bmatrix} \sigma_x^2 & \rho\sigma_x\sigma_y \\ \rho\sigma_x\sigma_y & \sigma_y^2 \end{bmatrix}. \tag{25}$$

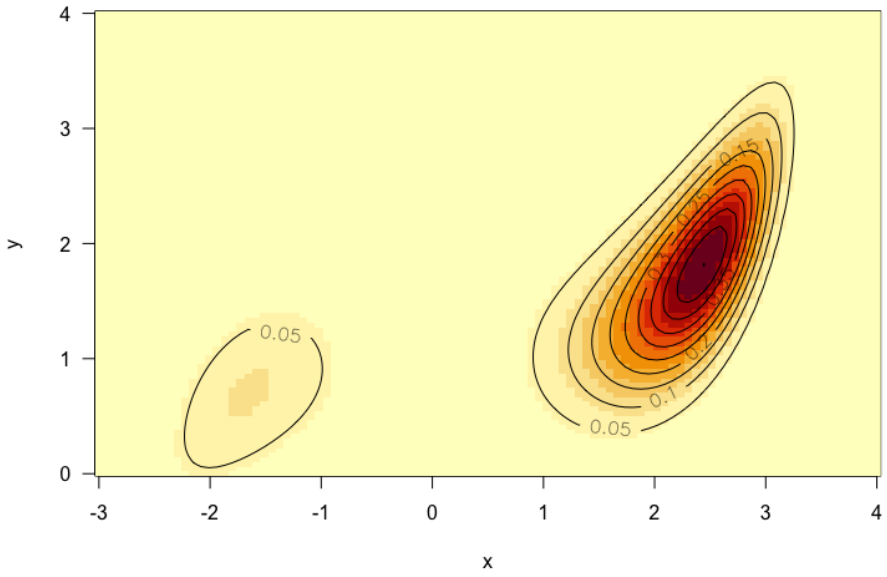
As the covariance matrix is defined to have a ρ value in it, it can be solved to extract the ρ value. This is done by taking either $(1,2)$ or $(2,1)$ as they contain $\rho\sigma_x\sigma_y$ from matrix (25). Then by dividing $(2,1)$ by the square roots of $(1,1)$ and $(2,2)$ for the equation (25) as they contain σ_x^2 and σ_y^2. Then the only variable left is ρ. This approach is generalized for other densities so that an estimate of ρ can be obtained. By going through the steps outlined above, the estimate is referred to as

$$\hat{\rho}(x,y). \tag{26}$$

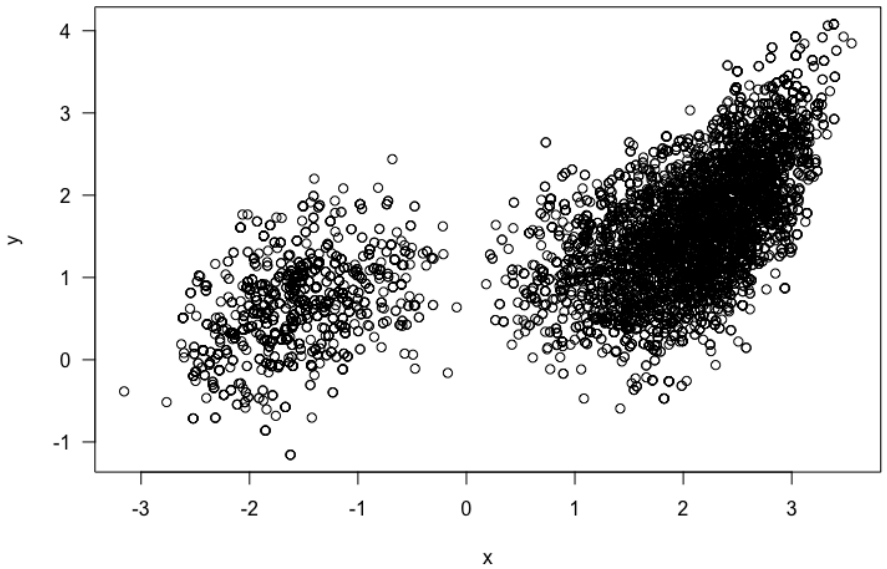
This estimate will always be $\hat{\rho}(x,y) = \rho$ for the bivariate normal density. Thus a connection between it and the local Gaussian correlation can be found as they both locally approximate the correlation coefficient for the bivariate normal density. Again likewise with the local Gaussian correlation, $\hat{\rho}(x,y)$ was found in **R**, this time by using the **TMB** package (Kristensen et al., 2016) which allows one to compile **C++** files in **R**. The reason why the **TMB** package is useful, is because it can return the double derivative values

for functions. It can also return the derivatives value if that is needed. The one thing to note is that the package does not give the explicit form of the derivatives or double derivatives. So to find out how the double derivatives actually look has to be done by hand.

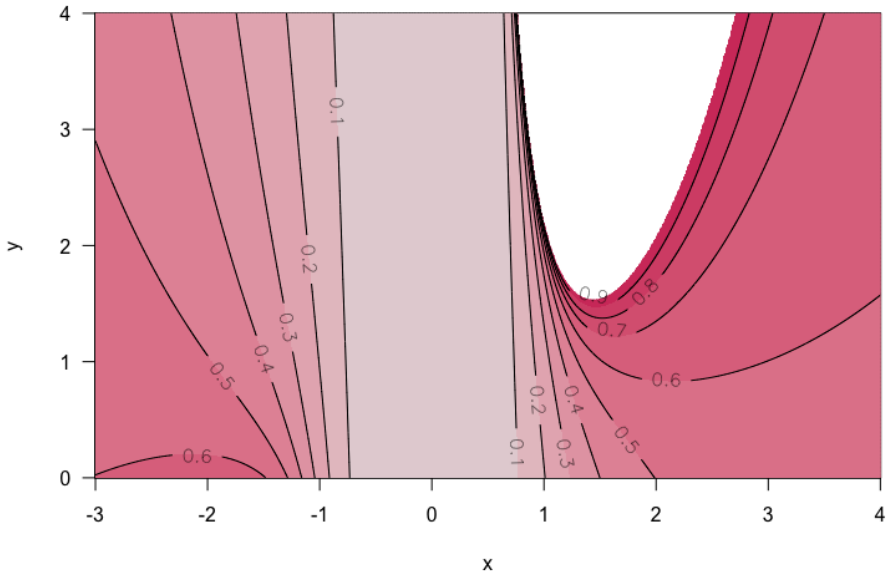
(a) Density.



(b) 10 000 Simulated observations from the density.



(c) $\hat{\rho}(x, y)$ estimates from the precision matrix.



(d) Local Gaussian correlation map for the simulated data.

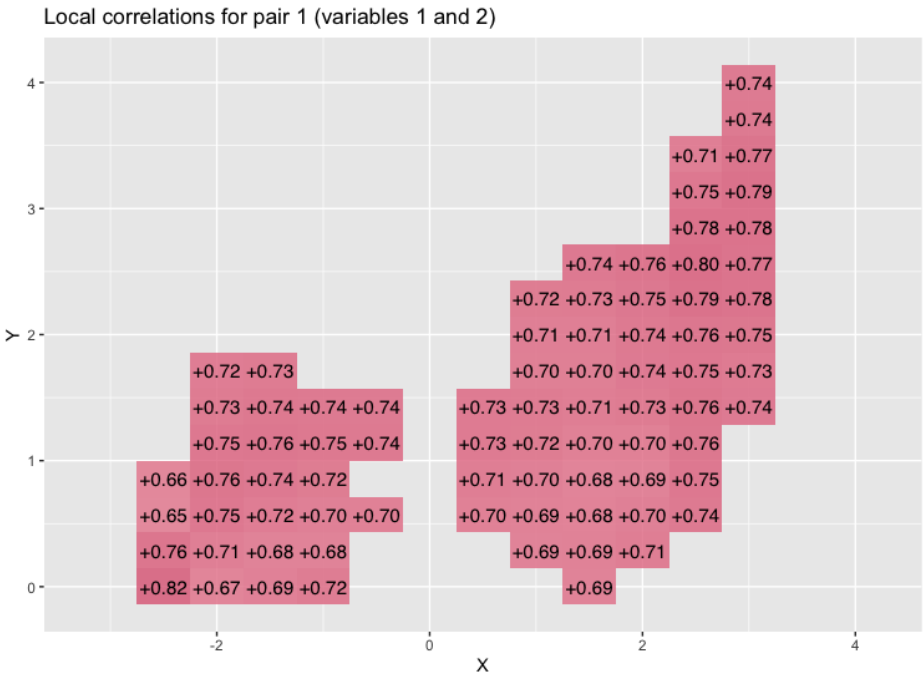
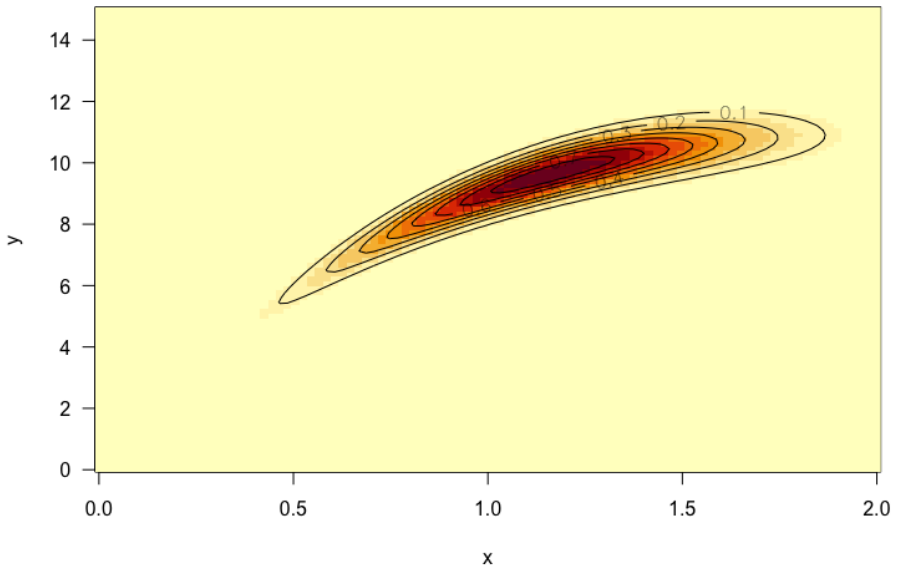
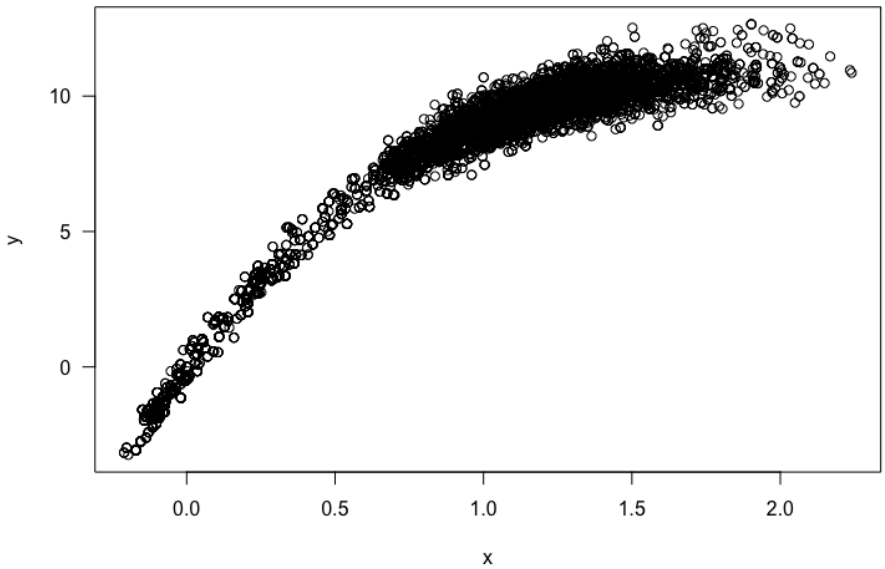


Figure 2: The density of $N(x, y, 10, 1.55, 10^2, 0.775^2, 0.75)$ where the density is a transformed normal density from (U, V) where $x = U^{1/3}$, $y = V$ and simulated observations of it. a) contour, b) simulated observations from the density, c) estimated correlation from the precision matrix and d) local Gaussian correlation map.

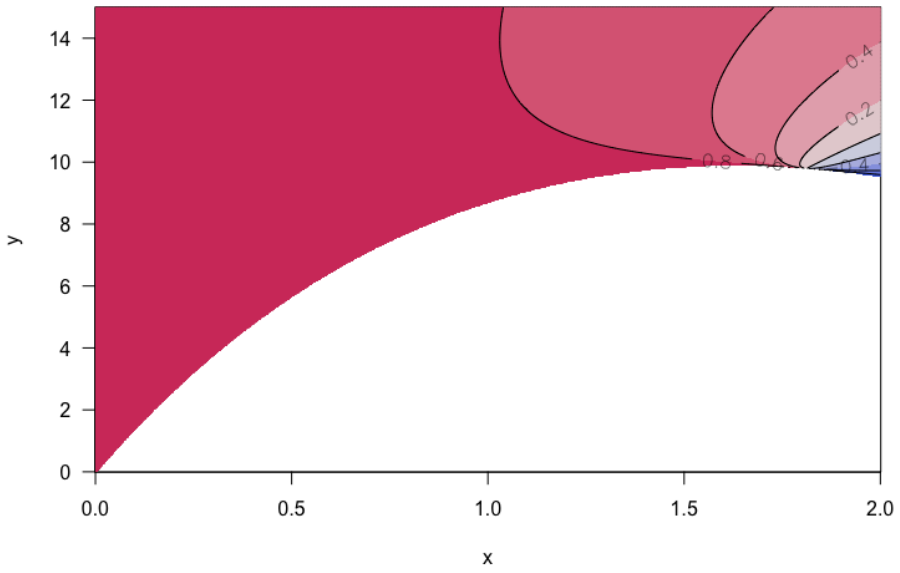
(a) Density.



(b) 10 000 Simulated observations for the density.



(c) $\hat{\rho}(x, y)$ estimates from the precision matrix.



(d) Local Gaussian correlation map for the simulated data.

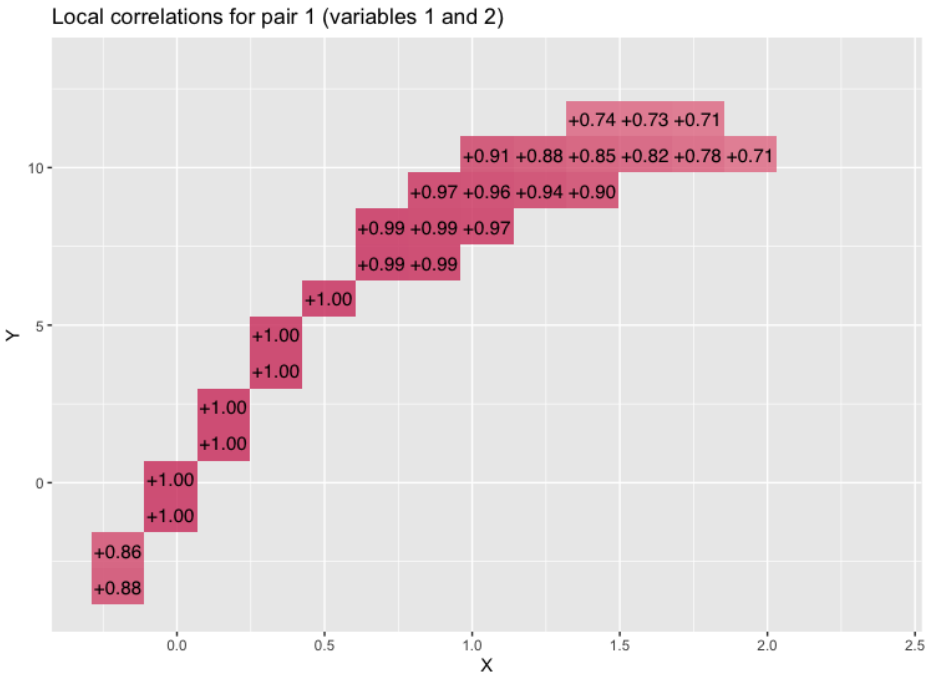
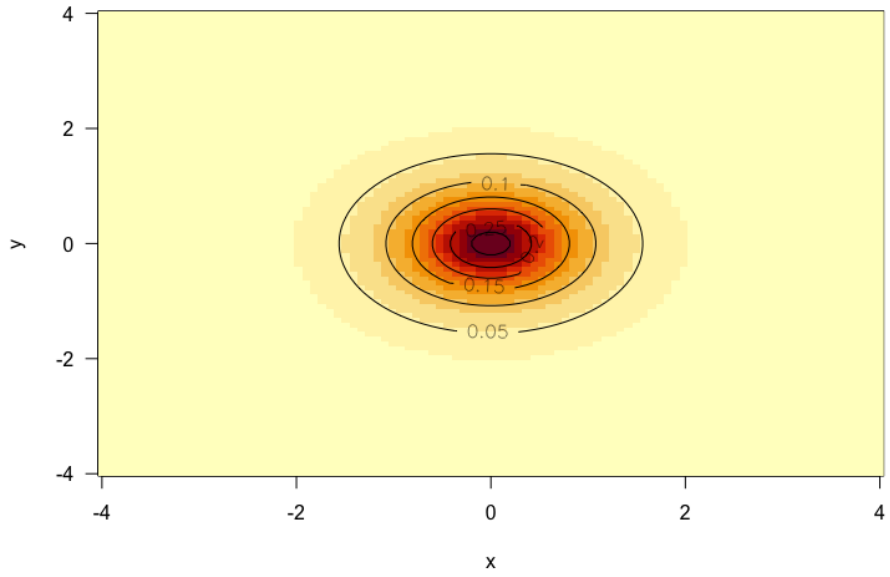
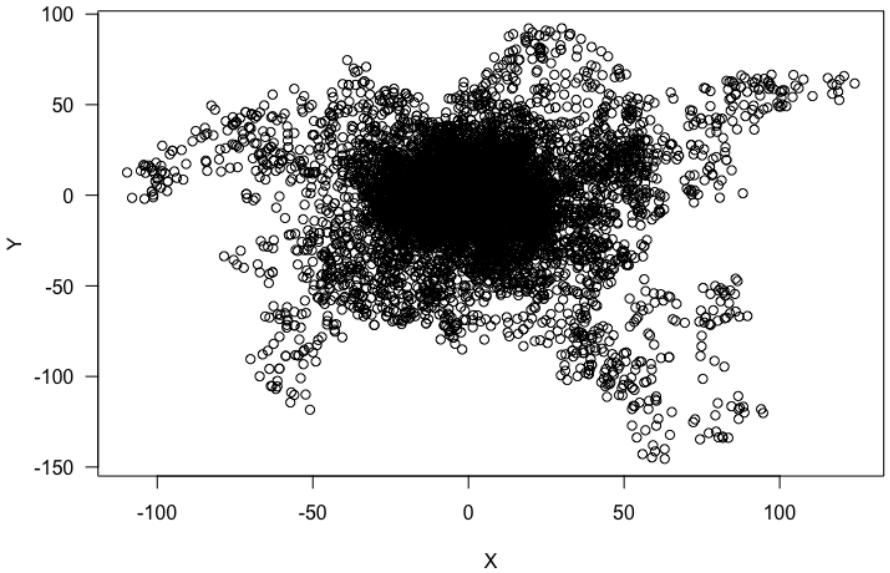


Figure 3: The density of twisted Pear $f(x, y) = f(x)f(y|x)$, where $f(x) = N(1.2, (1/3)^2)$, $f(y|x) = N(\mu(x), \sigma^2(x))$, $\mu(x) = (x/10)\exp(5 - (x/2))$, $\sigma^2(x) = [(1 + 0.5x)/3]^2$ and simulated observations from it. a) contour, b) simulated observations, c) precision matrix and d) local Gaussian correlation.

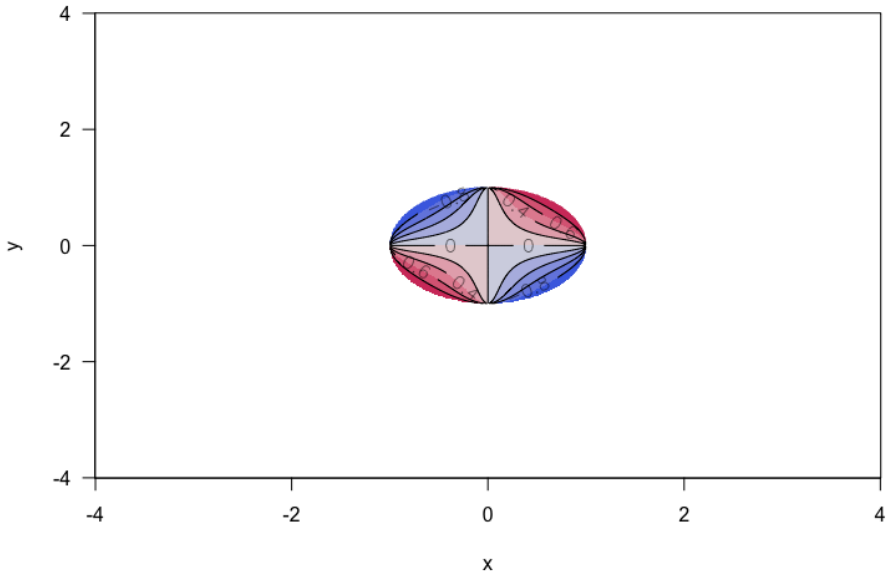
(a) Density.



(b) 100 000 Simulated observations from the density.



(c) $\hat{\rho}(x, y)$ estimates from the precision matrix.



(d) Local Gaussian correlation map for the simulated data.

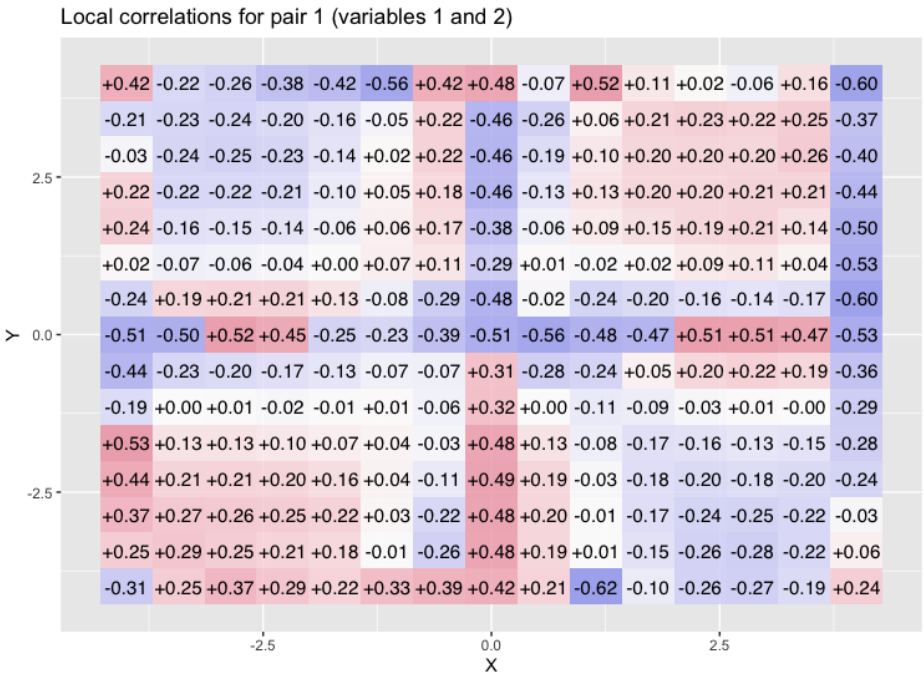
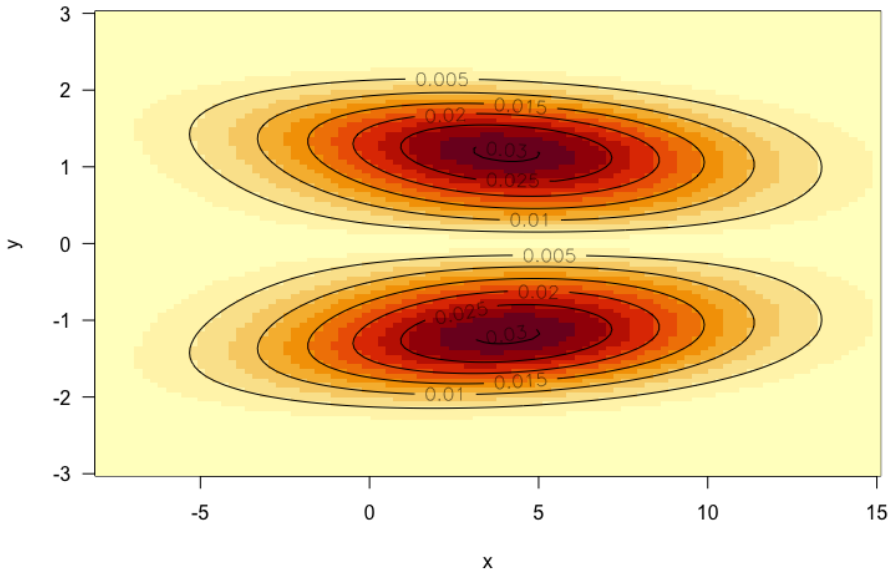
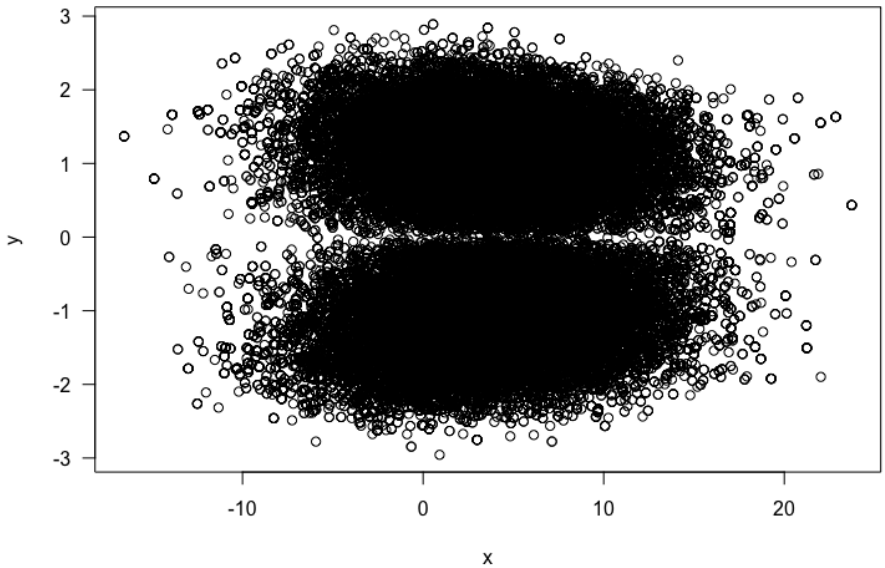


Figure 4: The Cauchy density and simulated observations from it. a) contour, b) simulated observations from the density, c) estimated correlation from the precision matrix and d) local Gaussian correlation map.

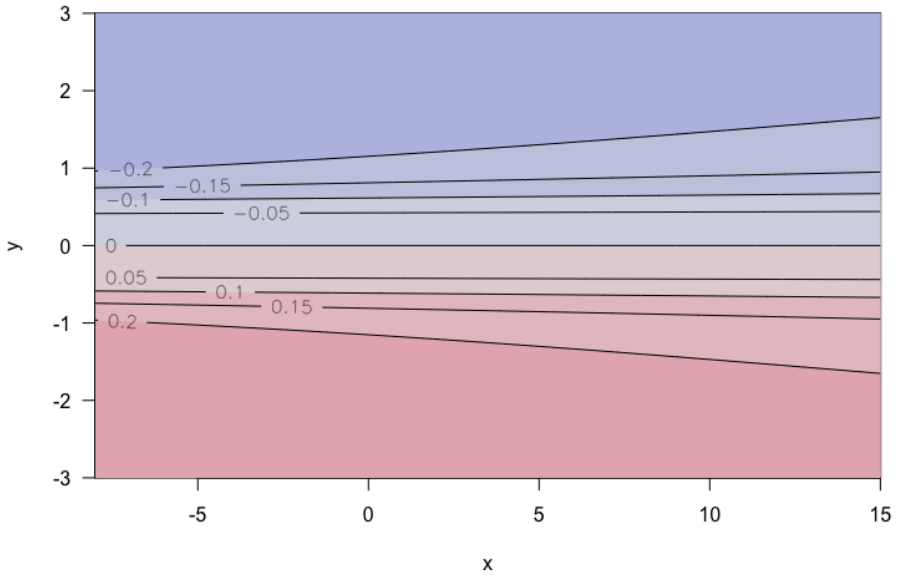
(a) Density.



(b) 100 000 simulated observations for the density.



(c) $\hat{\rho}(x, y)$ values from the precision matrix.



(d) Local Gaussian correlation map for the simulated data.

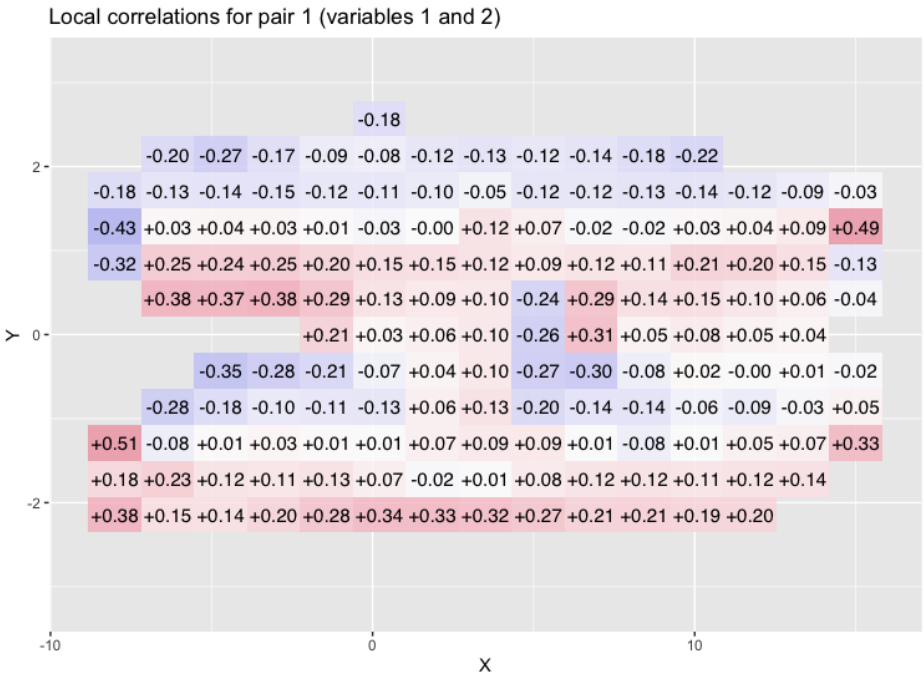


Figure 5: The density of transformed bivariate normal density $N(x, y, 4, 2, 5^2, 2^2, -0.27)$ where $x = U + 1, Y = \sqrt{V - 2}$ and simulated observations of it. a) contour, b) simulated observations from the density, c) estimated correlation from the precision matrix and d) local Gaussian correlation map.

6 Variance

The different densities for $\hat{\rho}(x, y)$ from equation (26) are displayed in figures 2c $-$ 5c. With the exception of the $\hat{\rho}(x, y)$ for the transformed bivariate normal distribution, the other figures 2c $-$ 4c have areas that are white. These white areas occur because $\hat{\rho}(x, y)$ is imaginary. The reason the estimate can be imaginary, is because it gets variance estimates for σ_x^2 and σ_y^2 and takes the square root of them. Thus where these estimates are negative, the resulting $\hat{\rho}(x, y)$ is imaginary. So while, estimating the variance is not the primary focus of the thesis, there is some values to analysing them. These estimates for the different densities are shown in figures 6, 7, 8 and 9. These estimates are taken from sequentially increasing x and y values instead of the sample. So as the sequences are dependent on the areas they are used over, the estimated variances will likewise also be dependent on those areas. Thus these estimates have to at least be taken with a bit of skepticism. In terms of the different estimates, only the estimates for the transformed normal density which corresponds to figure 9 have all of its values within the range of the histogram. This is demonstrated in figure 5c, as it is the only one that has no white areas. The other figures 6, 7 and 8 have excluded 5% or less of the variance estimates. The only exception is figure 6a, which is missing 68 309 observations out of 1 000 000. In terms of the negative values that are estimated there is some information to glean. For example the subfigures 8a and 8b have a reasonable amount of estimations that are negative, which at a surface level seems to correspond to why there is so much white area in figure 4c. Another point to note is the fact that for the different figures, most of the estimates fit within the range of $[-1, 1]$. While most of the values not included are just outside of this range, there are also occurrences of extreme estimated values in at least the thousands if not more. This will be further expanded upon in the next section.

(a) $\hat{\sigma}_x^2$ (b) $\hat{\sigma}_y^2$

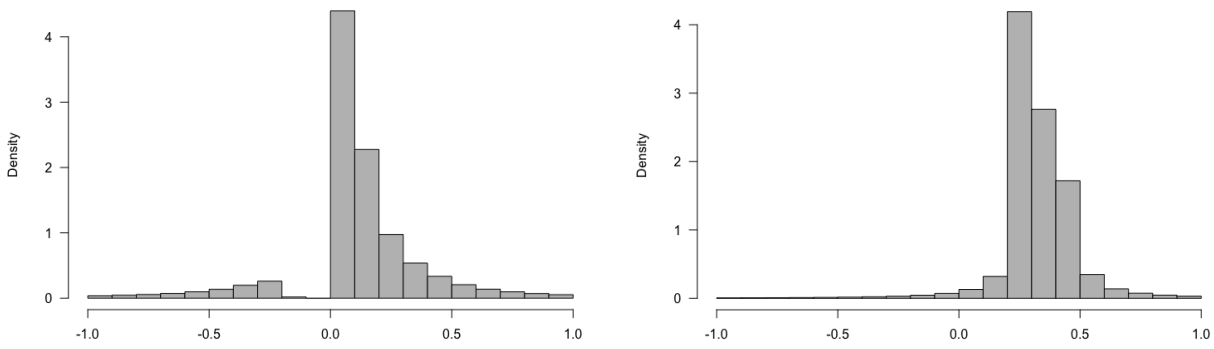


Figure 6: Variance estimates for the Pear density. *a*) the variance estimate of X and *b*) variance estimate of Y.

(a) $\hat{\sigma}_x^2$ (b) $\hat{\sigma}_y^2$

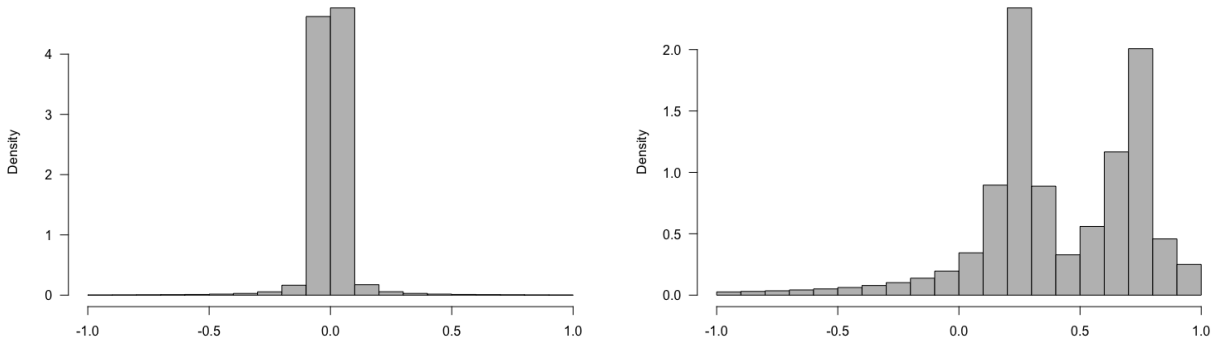


Figure 7: Variance estimates for the twisted Pear density. a) the variance estimate of X and b) variance estimate of Y.

(a) $\hat{\sigma}_x^2$ (b) $\hat{\sigma}2_y$

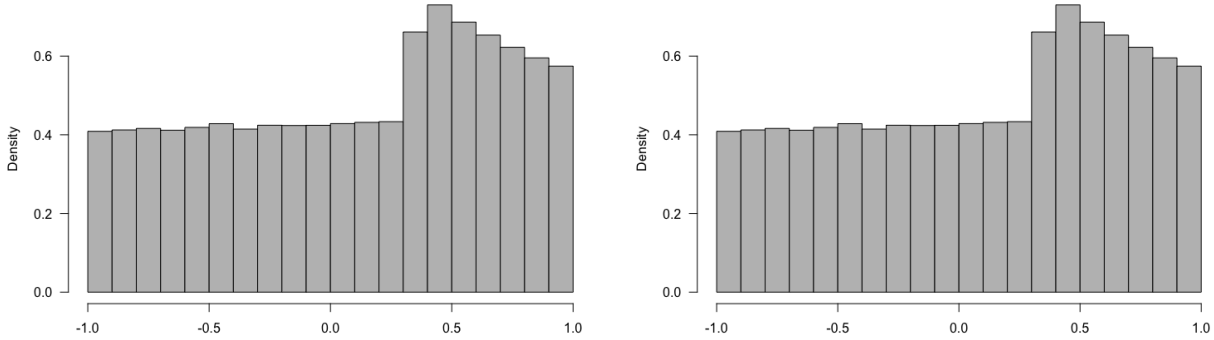


Figure 8: Variance estimates for the Cauchy density. a) the variance estimate of X and b) variance estimate of Y.

(a) $\hat{\sigma}_x^2$ (b) $\hat{\sigma}_y^2$

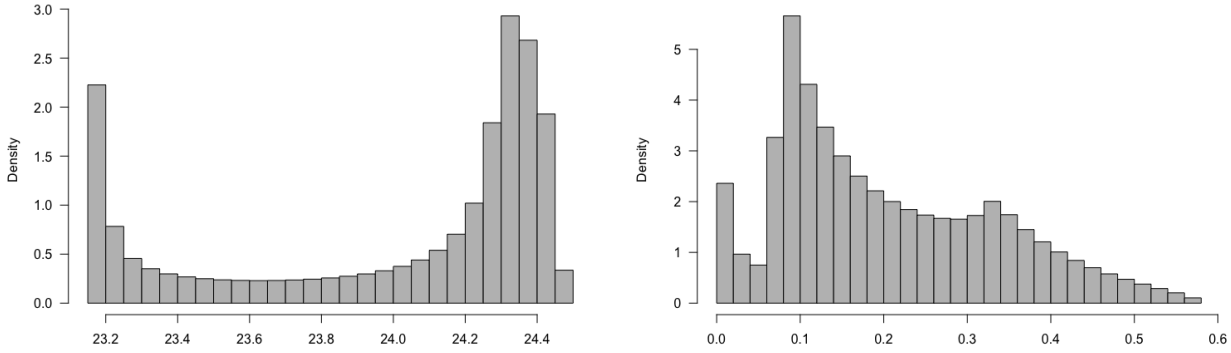


Figure 9: Variance estimates for the transformed bivariate normal density. a) the variance estimate of X and b) variance estimate of Y.

7 Results and Discussion

One of the main results from figures 2, 3, 4 and 5 is that both the local Gaussian correlation and the $\hat{\rho}(x, y)$ estimate from equation (26) indicate if the correlation is positive or negative for the given areas. With the figures 3c and 3d seeming to be the most similar. The other main result is that $\hat{\rho}(x, y)$ is within the range of $[-1, 1]$. Compared to the local Gaussian correlation the estimate $\hat{\rho}(x, y)$ is a function, so there is a smoother transition from one area to another. While for the local Gaussian correlation, these areas are less interconnected, so there is the possibility of an area of negative correlation occurring in a wider area that is positively correlated. The actual function for $\hat{\rho}(x, y)$ can be complex because of it using the double derivatives. As some of the chosen densities do not reduce when derived, particularly the twisted Pear. With the local Gaussian correlation, it is harder to grasp why certain areas return the correlation estimates, they do. As the package **lg** does the estimations for you. Another thing to note is that the transformed normal density and the Cauchy density have more simulated observations than the twisted Pear and the Pear density. This was done because as one can see in figures 4b and 5b, the simulations extend outside of the range of the distribution. As the correlation estimates look at the same range as the densities, some extra observations were simulated to make sure that approximately the same amount of observations are in the restricted ranges.

7.1 Pear

The first Pear transformed normal density is from Doksum et al. (1994). The Pear density is a transformed normal density shown in figure 2c, where x, y are transformed from (U, V). For the transformation $x = U^{1/3}$, $y = V$ and the density is given as

$$\frac{3x^2}{2\pi\sigma_x\sigma_y\sqrt{1-\rho^2}} \exp\left[-\frac{1}{2(1-\rho^2)}\left(\frac{(x^3-\mu_x)^2}{\sigma_x^2} - 2\rho\frac{(x^3-\mu_x)(y-\mu_y)}{\sigma_x\sigma_y} + \frac{(y-\mu_y)^2}{\sigma_y^2)}\right)\right]. \tag{27}$$

For this density, $\mu_x = \sigma_x = 10, \mu_y = 1.55, \sigma_y = 0.775$ and $\rho = 0.75$. Generally the correlation estimates in figure 2d seem to have approximated well to the

real ρ. While the estimates in figure 2c are less accurate at estimating the real value of ρ. For $\hat{\rho}(x, y)$, there seems to be no correlation in area between the two tops of the densities seen in figure 2a. In addition, while not observable there is a straight line at $x = 0$ that is undefined. The reason it is undefined is because the double derivative of x includes $2/x$ as part of the equation. On the other hand, there is a clear undefined area approximately around x for the values of $(1, 3)$ and y for the values of $(2, 4)$ in figure 3c. Along the edge of this undefined region is where the strongest correlation estimates occur as well as where the strongest variance estimates occur. These variance estimates are displayed in figure 10. These variance estimates being the positive and negative thousands. The reason for these negative approximations for $\hat{\sigma}_x^2$ and $\hat{\sigma}_y^2$ are because they are given by

$$\hat{\sigma_x^2} = \left[\frac{1}{0.775^2(1-0.75^2)}\right]/k$$
$$\hat{\sigma_y^2} = \left[\frac{1}{2(1-0.75^2)}\left(\frac{3x^4-12x}{10} - \frac{9x(y-1.55)}{7.75}\right) + \frac{2}{x^2}\right]/k,$$

where k is given as

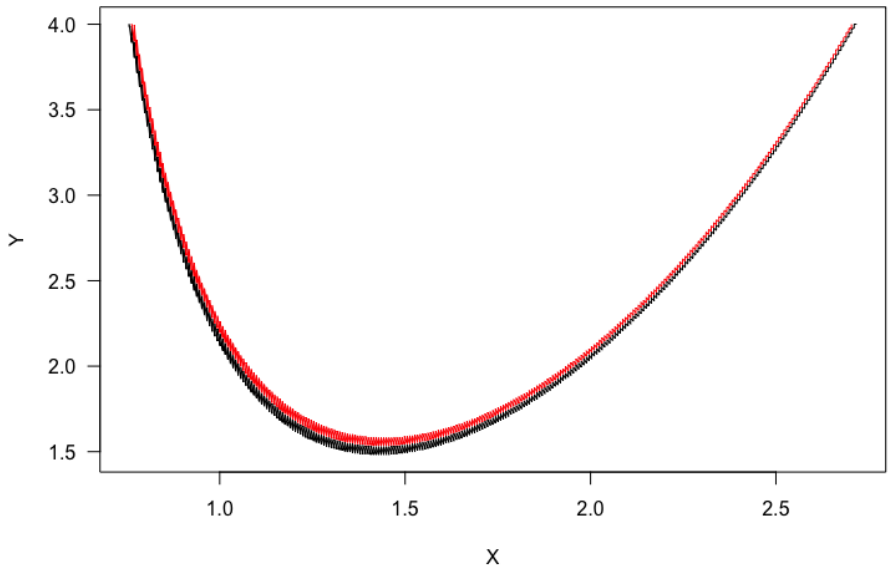
$$k = \left[\frac{1}{2(1-0.75^2)}\left(\frac{3x^4-12x}{10} - \frac{9x(y-1.55)}{7.75}\right) + \frac{2}{x^2}\right]\left(\frac{1}{0.775^2(1-0.75^2)}\right) - \left(\frac{4.5x^2}{15.5(1-0.75^2)}\right)^2.$$

From the denominator of $\hat{\sigma}_x^2$ we can easily set it up the inequality for ≤ 0. Solving the inequality for $\hat{\sigma_x^2}$, the end result is

$$39.75x^6 - 232.258x^3y - 120x^3 + 350 \leq 0, \tag{28}$$

Which explains the undefined area in figure 2c.

(a) $\hat{\sigma}_x^2$



(b) $\hat{\sigma}_y^2$

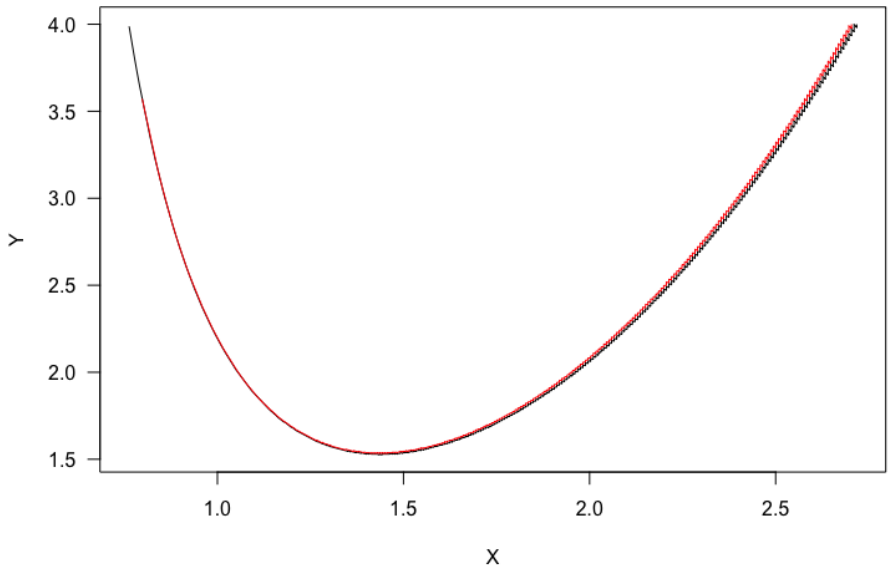


Figure 10: Outlier variance estimates for the Pear density. Red line are negative values, while the black line are positive values. a) estimates for X and b) estimates for Y.

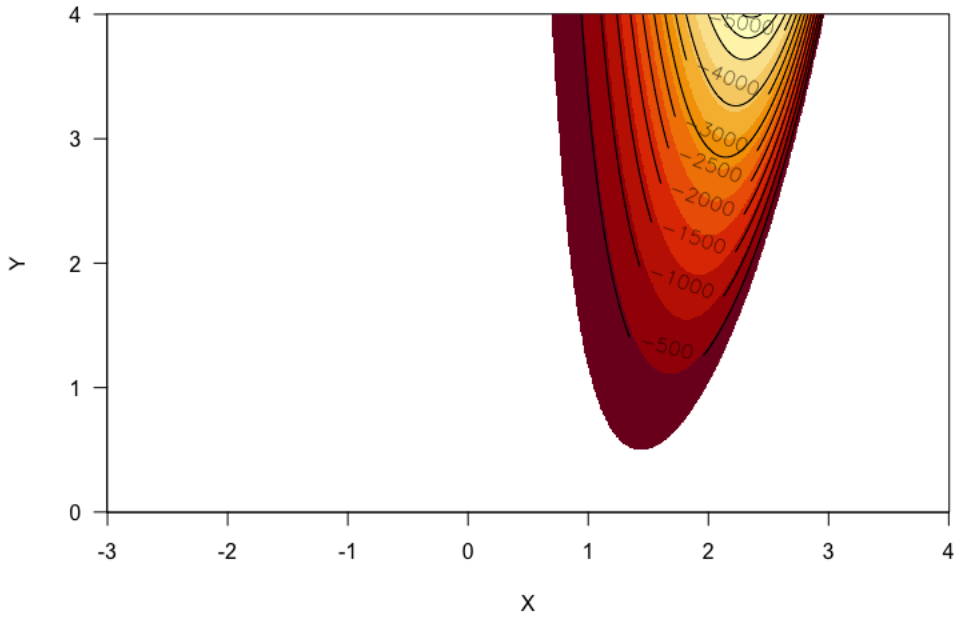


Figure 11: Negative values corresponding to the inequality for equation (28).

7.2 Twisted Pear

The twisted Pear density is from the introductory paper for the local dependence function (Jones, 1996), and originally used in Doksum et al. (1994). The twisted Pears density is given as:

$$\begin{aligned} f(x) &= \frac{1}{\sigma_x\sqrt{2\pi}} \times \exp\left[-\frac{1}{2}\left(\frac{x-\mu_x}{\sigma_x}\right)^2 \right] \\ f(y|x) &= \frac{1}{\sqrt{2\pi}\sigma(x)} \exp\left[-\frac{1}{2}\left(\frac{y-\mu(x)}{\sigma(x)}\right)^2 \right] \\ f(x,y) &= f(y|x) \times f(x). \end{aligned} \tag{29}$$

Where the functions $\mu(x)$ and $\sigma(x)$ are

$$\begin{aligned} \mu(x) &= \frac{x}{10}\exp(5-\frac{x}{2}) \\ \sigma(x) &= \left(\frac{1+0.5x}{3}\right), \end{aligned} \tag{30}$$

and $\sigma_x = 1/3$ and $\mu_x = 1.2$. In addition, the correlation coefficient ρ is not given. In terms of the estimates, this is the closest that $\hat{\rho}(x,y)$ and the local Gaussian correlation get for the given densities. As the correlation trend seems to be a very strong positive correlation along the left tail for figures 3c and 3d. Then towards the rightmost end of the figures, the correlation decreases in value and in the case of $\hat{\rho}(x,y)$ it gets into the negatives. So there seems to be a nonlinear dependence between the variables x and y.

7.3 Cauchy

The Cauchy density used is the bivariate form and takes the form of

$$f(x,y) = \frac{1}{\pi(1+x^2+y^2)^{3/2}}. \tag{31}$$

This is the same density as the one in the introductory paper for the local dependence function (Jones, 1996). The most prominent feature of the Cauchy densities are that they do not have defined variance. So both σ_x^2 and σ_y^2 are undefined, this means that the spread of observations can be the entire range of $(-\infty, \infty)$. For the given densities, the Cauchy density is the only one that $\partial^2/\partial x^2$ and $\partial^2/\partial y^2$ mirror each other. This can be seen by taking the double derivatives

$$\begin{aligned}
\frac{\partial^2}{\partial x^2}\log f(x,y) &= \frac{3(x^2-y^2-1)}{(x^2+y^2+1)^2} \\
\frac{\partial^2}{\partial y^2}\log f(x,y) &= \frac{3(y^2-x^2-1)}{(x^2+y^2+1)^2} \\
\frac{\partial^2}{\partial xy}\log f(x,y) &= \frac{6xy}{(x^2+y^2+1)^2}.
\end{aligned} \tag{32}$$

The only difference being whether the numerator contains $3(x^2-y^2-1)$ or $3(y^2-x^2-1)$. This results in figures 8a and 8b mirroring each other. We can show this by finding the covariance matrix using the derivatives. The covariance matrix then takes the form of

$$\frac{x^2+y^2+1}{3(x^2+y^2-1)} \times \begin{bmatrix} y^2-x^2-1 & -2xy \\ -2xy & x^2-y^2-1 \end{bmatrix}. \tag{33}$$

As we can see $(1,1)$ and $(2,2)$ also mirror each other for the equation (33). With the difference being whether there is a minus in front of x^2 or in front ofy^2. We can take this further and see why the defined areas for $\hat{\rho}(x,y)$ is circular for the Cauchy density. $\hat{\rho}(x,y)$ is then given as

$$\hat{\rho}(x,y) = \frac{\dfrac{-2xy(x^2+y^2+1)}{3(x^2+y^2-1)}}{\left(\sqrt{\dfrac{(y^2-x^2-1)(x^2+y^2+1)}{3(x^2+y^2-1)}}\right) \times \left(\sqrt{\dfrac{(x^2-y^2-1)(x^2+y^2+1)}{3(x^2+y^2-1)}}\right)}. \tag{34}$$

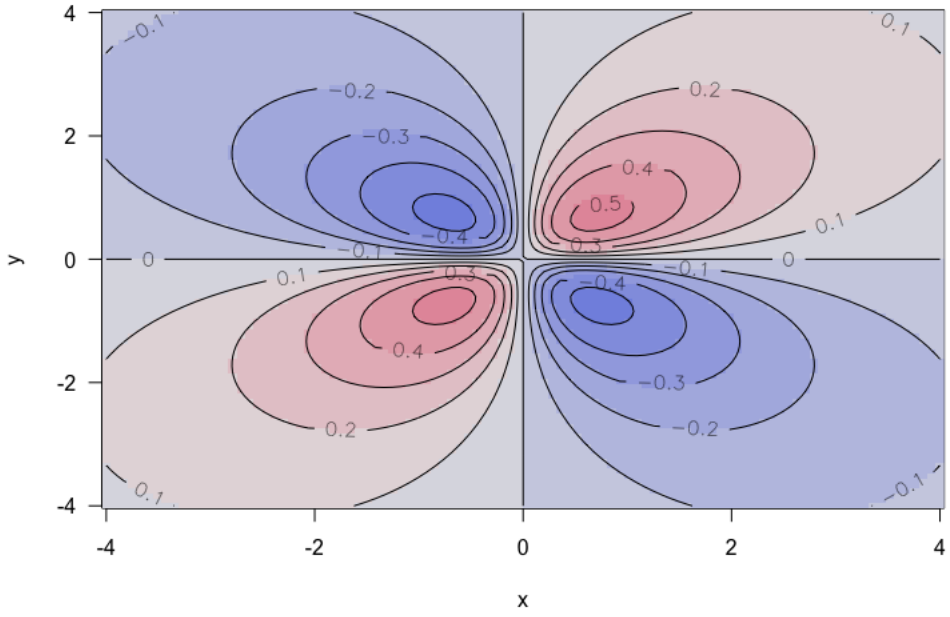


Figure 12: The correlation estimates from the local dependence function for the Cauchy distribution.

From equation (34), the restrictions end up being

$$\begin{aligned} &x^2 + y^2 \neq 1, \\ &x^4 \leq (y^2 - 1)^2, \\ &x^4 + 1 \geq 2x^2 + y^4. \end{aligned} \tag{35}$$

Thus resulting in the circular form. The other thing to note about $\hat{\rho}(x, y)$ is that when both variables have the same sign the defined areas in figure 4c are positive. While when the signs differ, there is negative estimated correlation. This seems to line up with figure 4d, although it does have some areas that break this trend. Interestingly, figure 4d resembles the local dependence function more as shown in figure 12 as the estimated correlation for the areas are more similar. It should be noted that figure 12 has the ρ estimates that are calculated from the γ function, using the method described in equation (17) from the experiments chapter.

7.4 Transformed Normal Density

For the final density, it seems that both the local Gaussian correlation and the $\hat{\rho}(x, y)$ indicate that there is a negative correlation on the positive side of the y-axis and positive correlation along the negative side of y. This is more clearly shown in figure 5c than in figure 5d, as there is a clearer transition for the estimate $\hat{\rho}(x, y)$. The local Gaussian correlation, on the other hand has a few areas that contradict the overall trend. In addition, only the top and bottom areas showing weak correlations. The middle area has correlation estimates close to 0. The reason for some areas contradicting the overall trend could be because of the areas being under sampled or, alternatively because of the weak negative correlation not being captured. Though the overall correlation trend seems to be, the further away from the y-axis the stronger correlated the variables are. So in a way pushing, the observations towards the axis. This may seem a bit counter intuitive when looking at the density displayed in figure 5a, but the highest probability areas have maximum probability of 0.03.

8 Box-Cox Transformation

The $\hat{\rho}(x, y)$ from equation 26 ends up with a small defined area. A possible way to mitigate this is by transforming the density, using the Box-Cox transformation is utilized. The Box-Cox transformation is a power transformation that makes the data look more normally distributed. The Box-Cox transformation that G. Box and D. Cox introduced in their 1964 paper (Box and Cox, 1964). The transformation is defined as

$$f(x,y)^{(\lambda)} = \begin{cases} \dfrac{f(x,y)^{\lambda} - 1}{\lambda} & (\lambda \neq 0), \\ \log(f(x,y)) & (\lambda = 0), \end{cases} \tag{36}$$

and for the two parameter transformation (Box and Cox, 1964)

$$f(x,y)^{(\lambda)} = \begin{cases} \dfrac{(f(x,y) - \lambda_2)^{\lambda_1} - 1}{\lambda_1} & (\lambda_1 \neq 0), \\ \log(f(x,y) + \lambda_2) & (\lambda_1 = 0). \end{cases} \tag{37}$$

For both of these transformations, there are restrictions. For equation (36) the restrictions is that $f(x, y) > 0$, otherwise when $\lambda = 0$ then $f(x, y)^{(\lambda)}$ is imaginary. For equation (37), the restriction is similar, which is that $f(x, y) > -\lambda_2$. Of these two transformations the uni parametric method is used. Thus using the Box-Cox transformation the precision matrix takes the form of

$$\begin{bmatrix} -\dfrac{\partial^2}{\partial x^2} f(x,y)^{(\lambda)} & -\dfrac{\partial^2}{\partial x \partial y} f(x,y)^{(\lambda)} \\ -\dfrac{\partial^2}{\partial x \partial y} f(x,y)^{(\lambda)} & -\dfrac{\partial^2}{\partial y^2} f(x,y)^{(\lambda)} \end{bmatrix}. \tag{38}$$

In addition, as $\lambda \to 0$ then $f(x, y)^{(\lambda)}$ will go towards $\log(f(x, y))$. This is shown in figures 15a as it almost identical to figure 4c. Another thing to note is that $\hat{\rho}(x, y)$ estimates for the Box-Cox transformed density are no longer bound to the range of $[-1, 1]$. For the density, as λ increase then the probability decreases and becomes more homogeneous as seen in figure 14. This is similar for the estimate $\hat{\rho}(x, y)$, as there is a decrease in estimated value as λ increases as figure 13 shows. The exception to this is $\lambda = 0.001$. As one can see in figure 15, the strongest correlation is estimated outside of the circle, in the diagonal areas. it seems like as λ decreases, the diagonal

(a) $\lambda = 0.1$ (b) $\lambda = 10$

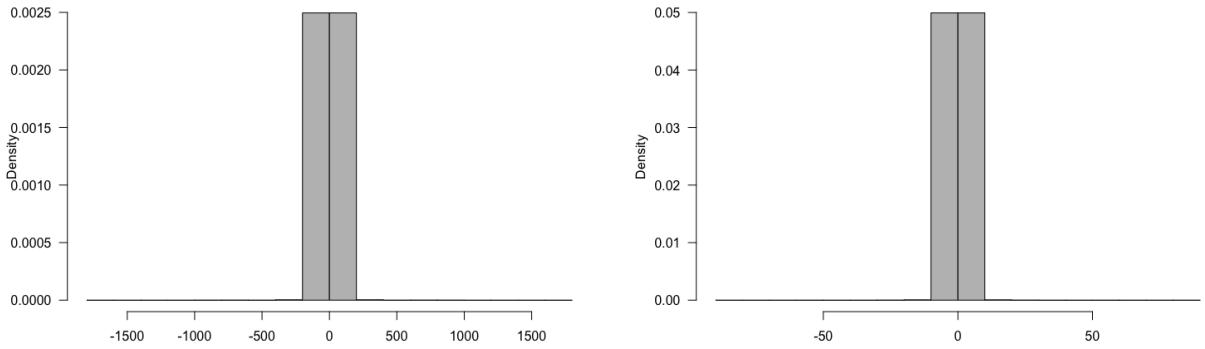
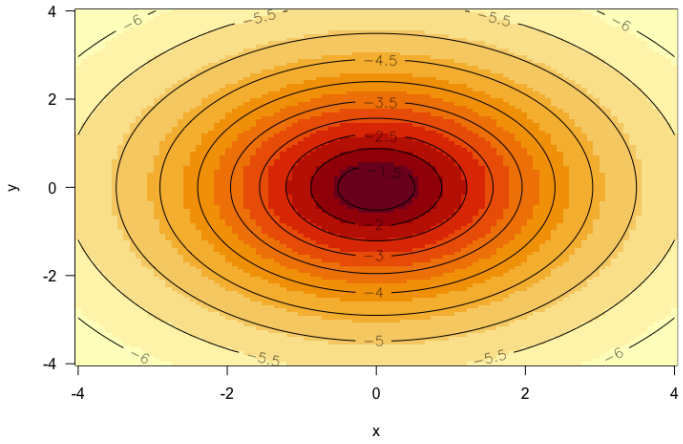


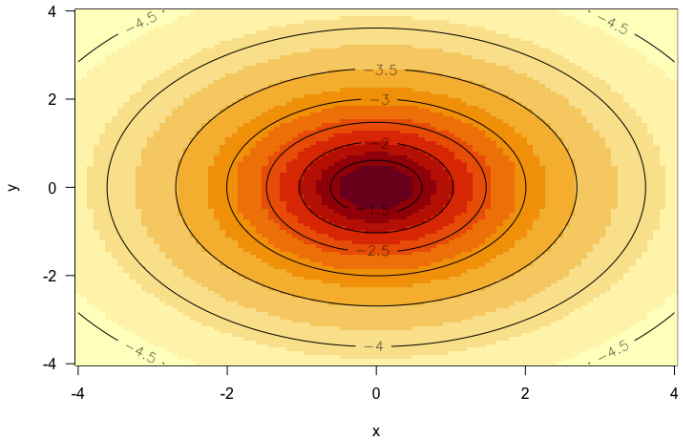
Figure 13: $\hat{\rho}(x, y)$ values for the Box-Cox transformed Cauchy density for 2 λ values. a) $\lambda = 0.1$ and b) $\lambda = 10$.

areas thin down and increase in value. While λ increases, the circle decreases and the diagonals increase. Additionally although hard to see, the edges of the diagonal areas are where the strongest estimated correlation occur. This is further exemplified in figure 15b.

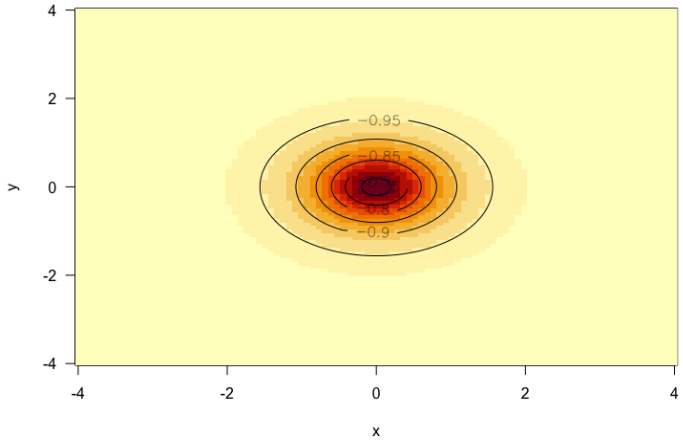
(a) $\lambda = 0.001$



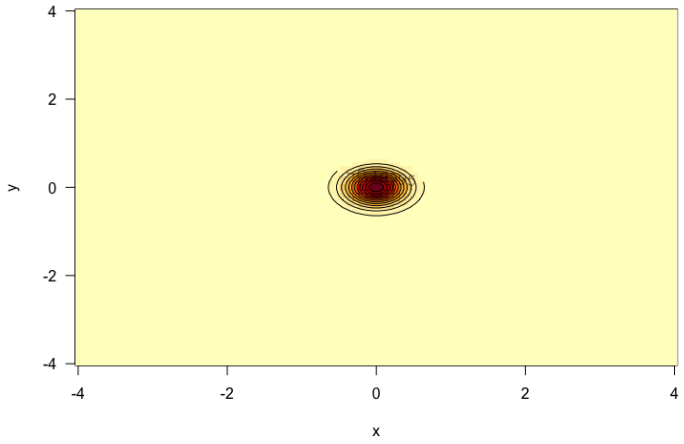
(b) $\lambda = 0.1$



(c) $\lambda = 1$



(d) $\lambda = 5$



(e) $\lambda = 10$

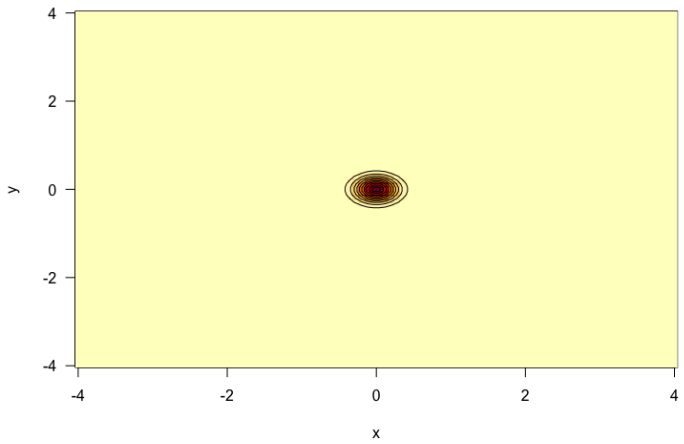
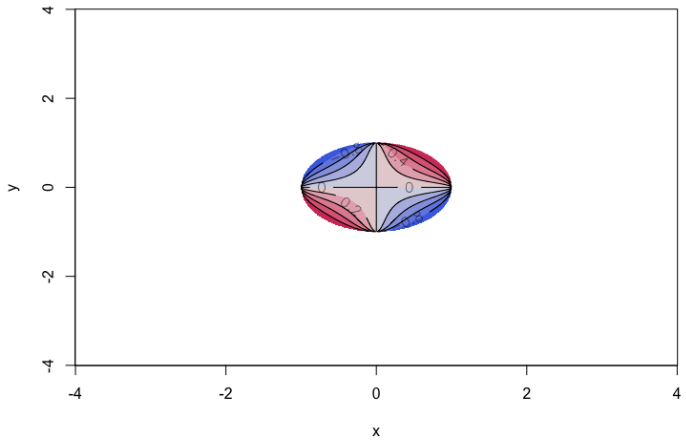
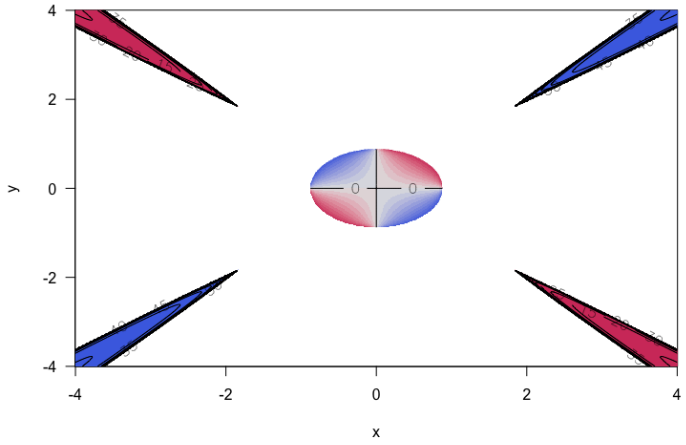


Figure 14: Box-Cox transformed Cauchy density for 5 different λ values. a) $\lambda = 0.001$, b) $\lambda = 0.1$, c) $\lambda = 1$, d) $\lambda = 5$ and e) $\lambda = 10$.

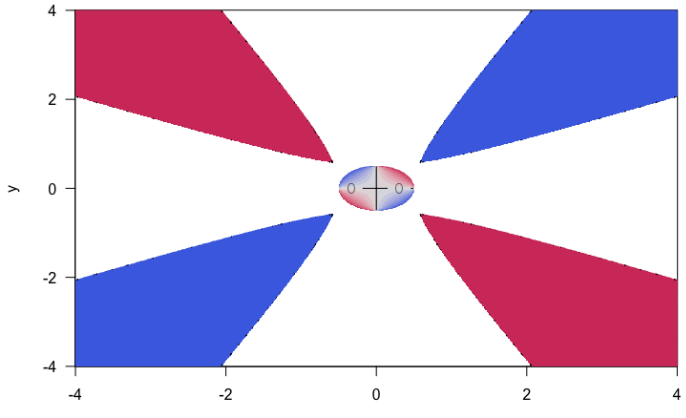
(a) $\lambda = 0.001$



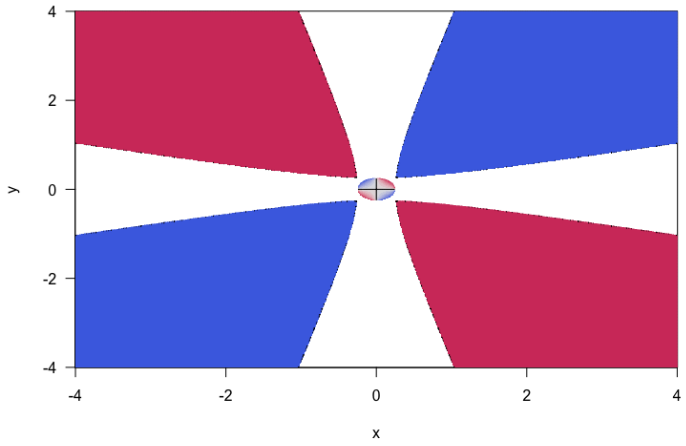
(b) $\lambda = 0.1$



(c) $\lambda = 1$



(d) $\lambda = 5$



(e) $\lambda = 10$

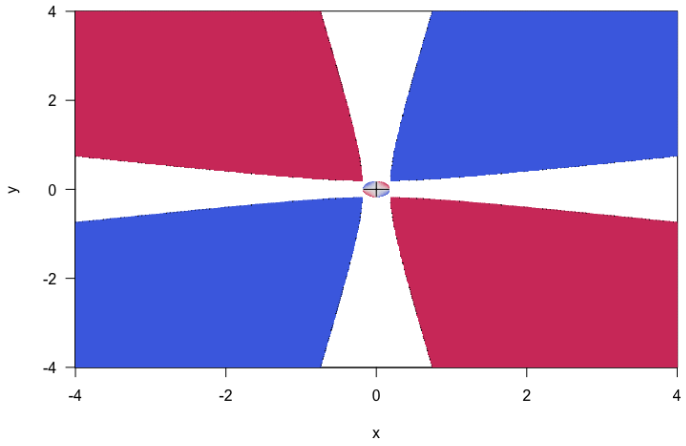


Figure 15: Correlation estimates using the precision matrix for the Box-Cox transformed Cauchy density for 5 different λ values. a) $\lambda = 0.001$, b) $\lambda = 0.1$, c) $\lambda = 1$, d) $\lambda = 5$ and e) $\lambda = 10$.

9 Kernel Smoother

Kernels are a statistical application with multiple uses. The application this paper will focus on is kernel smoothers. As the name implies kernel smoothers are used to transform or smooth the data within specified areas using kernels. In terms of how the data is transformed, there are a multitude of different kernel functions. The chosen examples are; the uniform kernel, triangle kernel, Gaussian kernel and Epanechnikov kernel. These examples are seen in figure 16. The different functions in the figure are; the uniform kernel function is (Ivanka, 2012)

$$K(x) = \frac{1}{2} I_{[-1,1]}(x), \tag{39}$$

where I is an indicator function taking on the form of

$$I_{[-1,1]}(x) = \begin{cases} 1 & \text{if } x \in [-1,1], \\ 0 & \text{otherwise.} \end{cases} \tag{40}$$

The triangle kernel function has the form of

$$K(x) = (1-|x|) I_{[-1,1]}(x). \tag{41}$$

The Gaussian kernel function is given as

$$K(x) = \frac{1}{\sqrt{2\pi}} \exp(-\frac{1}{2} x^2). \tag{42}$$

and finally the Epanechnikov kernel takes the form of

$$K(x) = \frac{3}{4}(1 - x^2) I_{[-1,1]}(x). \tag{43}$$

As previously mentioned these examples are just a few of many more kernel functions. They all serve the purpose of transforming data within given areas. For the kernel smoothers, the bandwidth h is included. h is a parameter that allows one to control how smooth the transformed data is. So for example the Gaussian kernel smoother with the addition of the parameter h is

$$K(\frac{x - X_i}{h}) = \exp\left(- \frac{(x - X_i)^2}{2h^2} \right), \tag{44}$$

where X_i is our ith observation for the data. The variables x and h are parameters that we can set to smooth out our observations. Choosing the optimal kernel smoother is done by using the Mean Integrated Square Error (MISE) and the Asymptotic Mean Integrated Square Error (AMISE). AMISE and MISE are extensions of the Mean Square Error (MSE), and are accuracy measurements. Which instead of taking summation over the area, they instead integrate over the area. MISE is given as an integration over the area of the data. So MSE is given as (Ivanka, 2012)

$$MSE[\hat{f}(x,h)] = \sum_{i=1}^{n}(\hat{f}(x,h) - f(x))^2, \tag{45}$$

f is the given density for the observed data. $\hat{f}$ is the sum of the kernel estimates for the data. The kernel density estimate was introduced by Parzen and Rosenblatt in their 1956 paper (Murray, 1956) and takes the form of

$$\hat{f}(x,h) = \frac{1}{nh}\sum_{i=1}^{n} K\left(\frac{x - X_i}{h}\right), \tag{46}$$

in the equation X_i is the ith observed data out of the dataset $(X_1, ..., X_n)$. For the above equation, $K(\frac{x - X_i}{h})$ are the different kernels for the different given areas. Finally MISE is given as:

$$MISE[\hat{f}(x,h)] = \int MSE[\hat{f}(x,h)]dx, \tag{47}$$

and AMISE is given as:

$$AMISE = \frac{1}{nh}\int K(x)^2dx + \frac{1}{4}\int x^2K(x)dx \times h^4 \int (f''(x))^2dx. \tag{48}$$

In AMISE, the first part is the Asymptotic Integrated Variance (AIV) and the second part is the Asymptotic Integrated Square Bias (AISB) so then $AMISE = AIV + AISB$ (Ivanka, 2012) and MISE likewise can be defined as

$$MISE(\hat{f}(x,h)) = AMISE(\hat{f}(x,h)) + o\{\frac{1}{nh} + h^4\}. \tag{49}$$

In terms of minimizing the MISE and AMISE score, one tries to find the optimal bandwidth. The optimal bandwidth can be found by solving the derivative of AMISE. This differential equation is set up as

$$\begin{aligned}\frac{\partial}{\partial h} AMISE(\hat{f}(x,h)) = &-\frac{1}{nh^2}\int K(x)^2 dx \\ &+ \int x^2 K(x) dx \times h^3 \int (f''(x))^2 dx = 0.\end{aligned} \tag{50}$$

Furthermore the optimal bandwidth h takes the form of

$$h_{AMISE} = \frac{\left[\int K(x)^2 dx\right]^{1/5}}{\left[\int x^2 K(x) dx\right]^{2/5}\left[\int f''(x)^2 dx\right]^{1/5}} n^{-1/5}. \tag{51}$$

Luckily there are simpler ways to approximate the optimal h value. Specifically for the Gaussian kernel. For the Gaussian kernel, there are rule of thumb estimates such as Scott's rule of thumb (Scott, 2015) and Silverman's rule of thumb (Silverman, 1986). Silverman's rule of thumb and Scott's rule of thumb are similarly defined with the only big difference being the constant they use. Silverman's rule of thumb bandwidth is

$$h = 0.9 \min\{\hat{\sigma}, IQR/1.34\} n^{-1/5}, \tag{52}$$

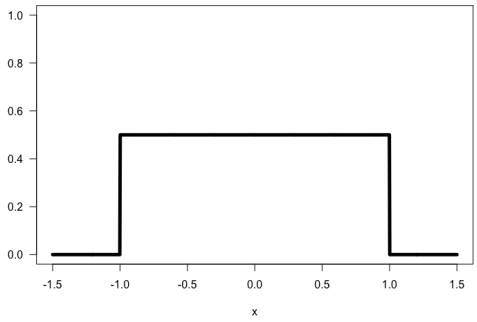
and Scott's rule of thumb is

$$h = \hat{\sigma} n^{-1/(d+4)}. \tag{53}$$

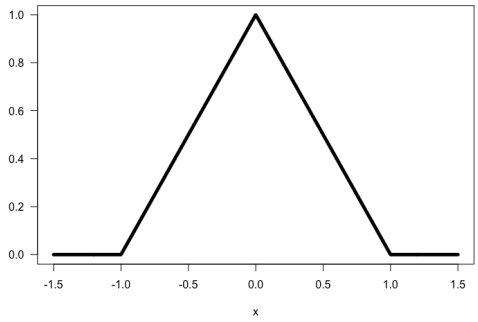
For both of these estimates, $\hat{\sigma}$ is the empirical standard deviation, n is the length of the dataset, IQR is the interquartile range and d is the amount of dimensions for the dataset. These two rules of thumb can also be up scaled to also work for multivariate kernels. For the implantation of them in **R**, both functions are built in. So Silverman is **bw.nrd0** and Scott is **bw.nrd**.

So far the kernel functions have been implemented for univariate data. However, they can also be expanded to being multivariate. In this paper the only multivariate kernel function used, is the bivariate Gaussian kernel. The

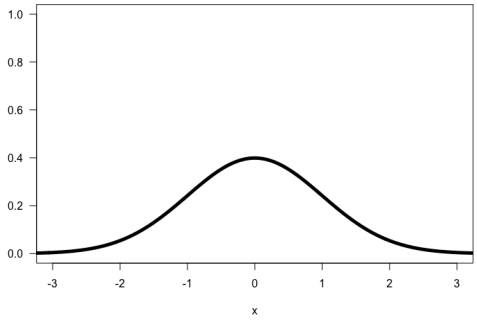
(a) Uniform kernel function.



(b) Triangle kernel function.



(c) Gaussian kernel function.



(d) Epanechnikov kernel function.

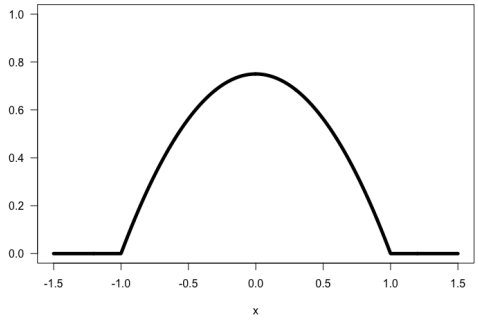


Figure 16: Four examples of kernel functions.a) Uniform kernel function, b) Triangle kernel function, c) Gaussian kernel function, d) Epanechnikov kernel function.

bivariate Gaussian kernel is given as

$$\begin{aligned} K\left(\frac{x}{h},\frac{y}{h}\right) =& \left[\frac{1}{\sqrt{2\pi h}}\exp\left(\frac{-1}{2}\frac{x^2}{h}\right)\right] \times \left[\frac{1}{\sqrt{2\pi h}}\exp\left(\frac{-1}{2}\frac{y^2}{h}\right)\right], \\ \hat{f}(x,y,h) =& \frac{1}{n}\sum_{i=1}^{n} K\left(\frac{x-X_i}{h},\frac{y-Y_i}{h}\right), \end{aligned} \tag{54}$$

where x and y are chosen values, and X_i and Y_i are the observations from the dataset with $i = \{1, ..., n\}$.

10 Kernel Estimates

Similar to what was done in the Box-Cox transformation. One can instead of using $f(x, y)$ for the precision matrix, use the bivariate Gaussian kernel estimated density $\hat{f}(x, y, h)$. Then the precision matrix takes the form of

$$\begin{bmatrix} -\frac{\partial^2}{\partial x^2}\hat{f}(x,y,h) & -\frac{\partial^2}{\partial x \partial y}\hat{f}(x,y,h) \\ -\frac{\partial^2}{\partial x \partial y}\hat{f}(x,y,h) & -\frac{\partial^2}{\partial y^2}\hat{f}(x,y,h) \end{bmatrix}. \tag{55}$$

Just as before the same steps are taken. Then the resulting correlation estimate, is given as

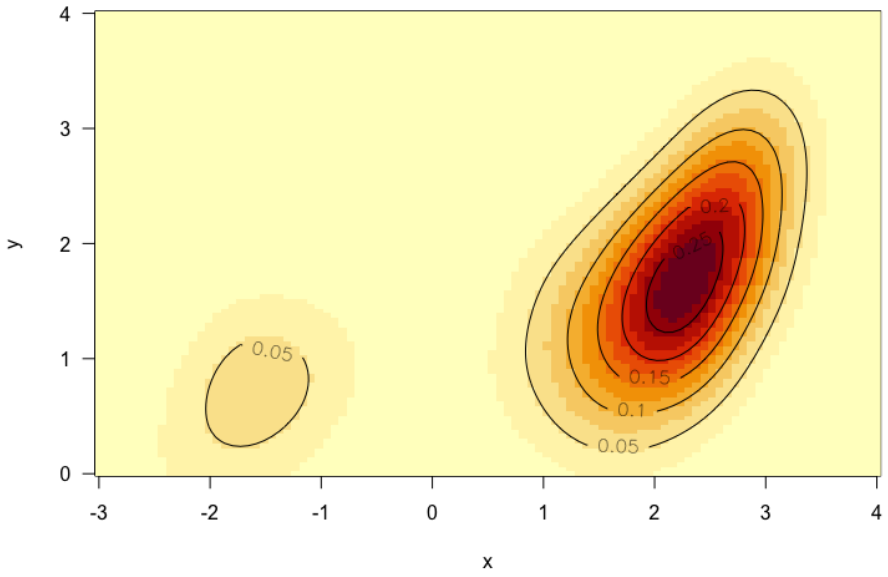
$$\hat{\rho}_K(x, y). \tag{56}$$

The practical reason for doing this, is because for real datasets the densities that produce them may be unknown. Thus resulting in the need to estimate the densities.

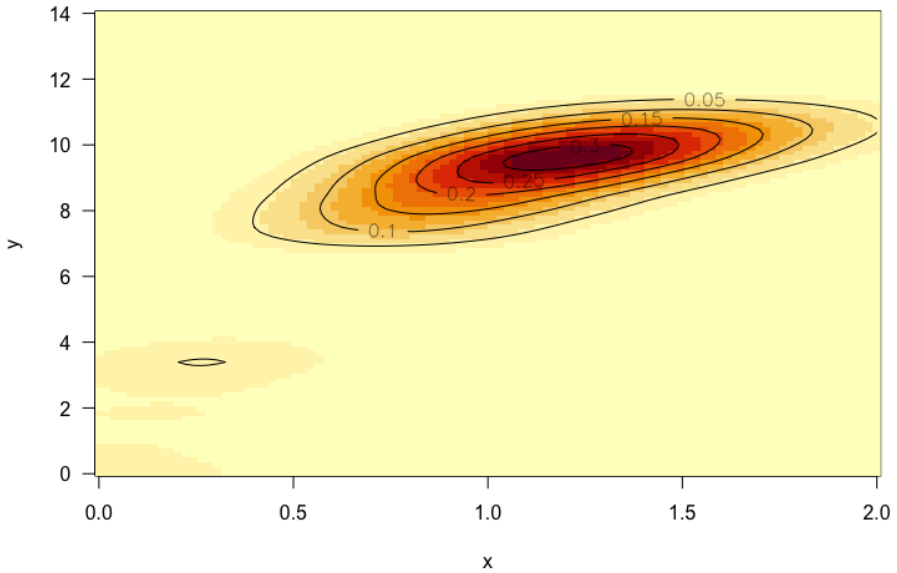
10.1 Contours

To give an overview over the different fits for the estimates, the subfigures 17a and 17d from figure 17 have approximated the form of the real densities closely. Specifically subfigure 17a is almost identical to subfigure 2a. The probabilities of the Gaussian kernel estimate of the Cauchy density are too low compared to the real Cauchy's probabilities. This also occurs for the fit of the twisted Pear, as the probability in the center of the density is too low. Just to exemplify how the change in bandwidth changes the bivariate Gaussian kernel estimated density $\hat{f}$, the transformed normal density is used. The figure 18 shows that as the bandwidth decreases towards 0 then the bands become less smooth. While for higher bandwidths the smoothness increases. In addition, the probabilities increase for lower bandwidths and decrease for higher bandwidths.

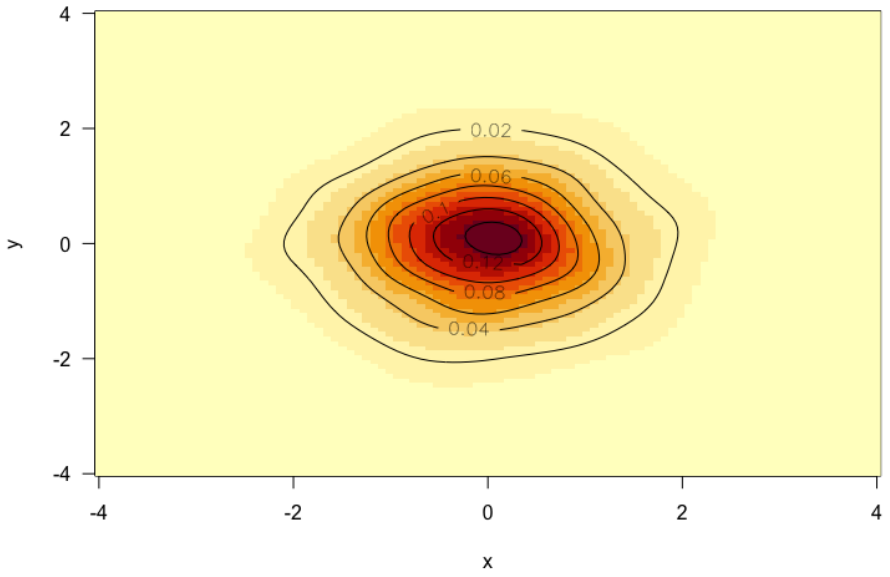
(a) $h = 0.119$



(b) $h = 0.08$



(c) $h = 0.093$



(d) $h = 0.28$

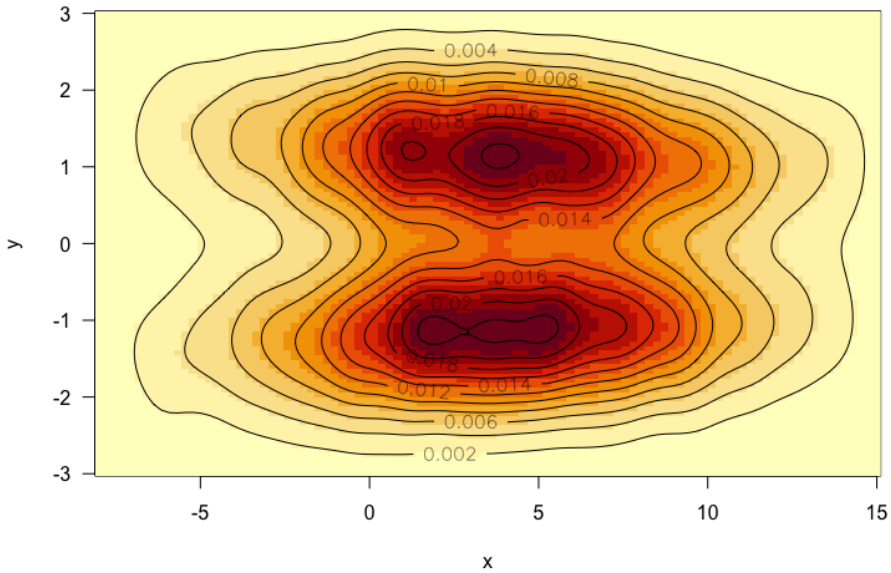
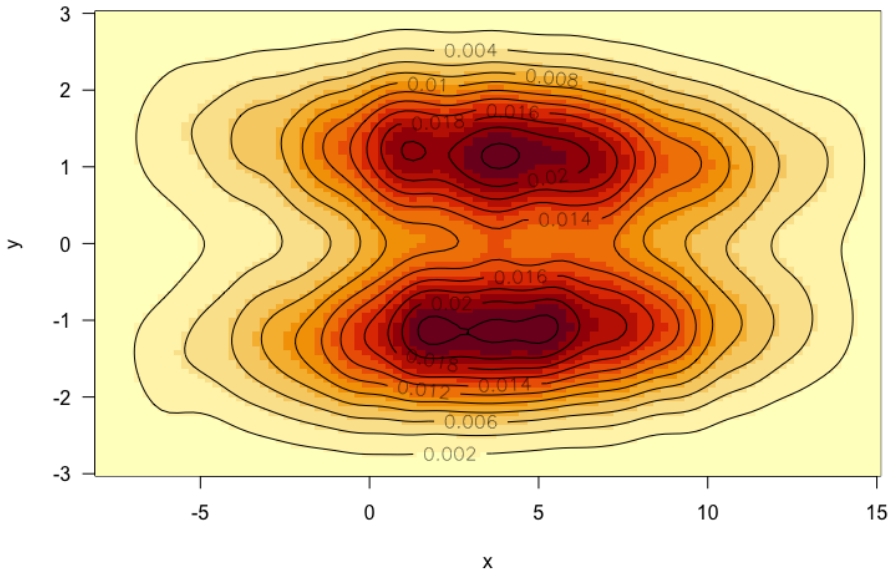
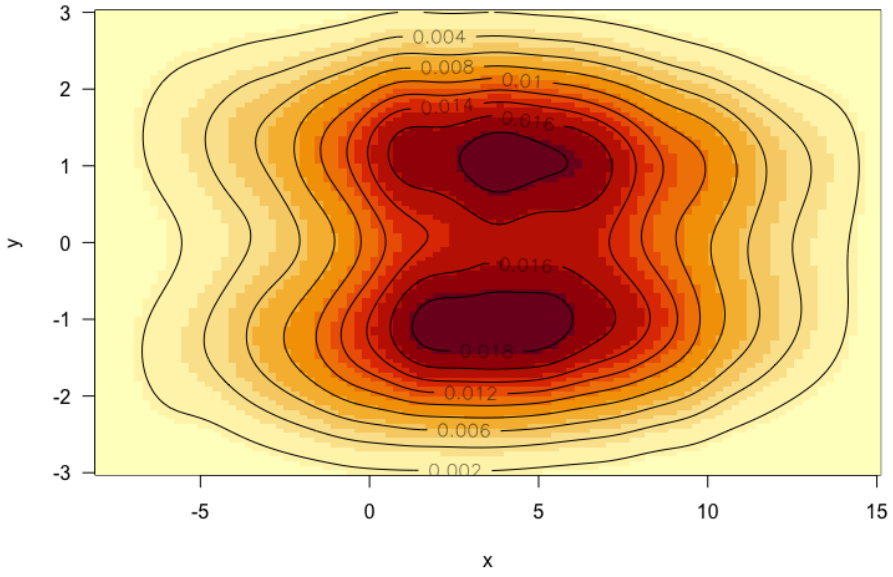


Figure 17: Bivariate Gaussian kernel estimates using the rule of thumb bandwidths. a) Pear density for $h = 0.119$, b) twisted Pear for $h = 0.08$, c) Cauchy for $h = 0.093$ and d) transformed normal density for $h = 0.28$.

(a) $h = 0.28$


0.004
0.008
0.01
0.016
0.018
0.02
0.014
0.016
0.02
0.018
0.012
0.014
0.006
0.002
3
2
1
0
-1
-2
-3
y
-5
0
5
10
15
x

(b) $h = 0.5$


0.004
0.008
0.01
0.014
0.016
0.016
0.018
0.012
0.006
0.002
3
2
1
0
-1
-2
-3
y
-5
0
5
10
15
x

(c) $h = 1$

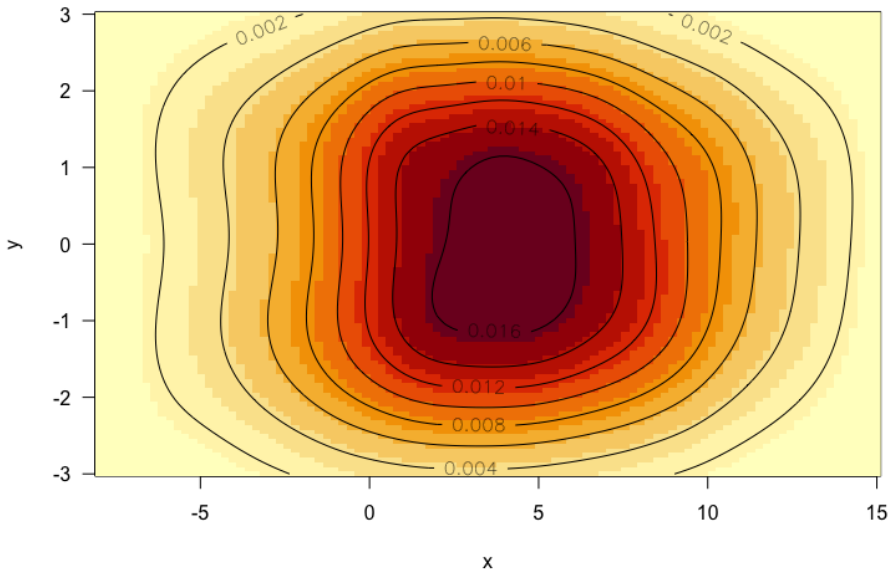


Figure 18: Bivariate Gaussian kernel estimate of the transformed normal density for 3 different h values. a) $h = 0.28$, b) $h = 0.5$ and c) $h = 1$.

10.2 Double Derivatives

While the actual $\hat{f}$ approximations we get might look similar. The estimated correlation maps tell a different story. Even from a cursory glance the subfigures of 19 are vastly different from the $\hat{\rho}(x, y)$ estimates from figures 2c, 3c, 4c and 5c. Of these, only figure 19a has a resemblances to the actual $\hat{\rho}(x, y)$ from figure 2c. Similarly to the Box-Cox transformation, the $\hat{\rho}_K(x, y)$ values from equation (56) are not limited to the range of $[-1, 1]$. But just like with the contours as the bandwidth increases, the spectrum of which $\hat{\rho}$ is defined decreases back within the $[-1, 1]$ as seen in figure 20. Based on figure 20, h values of 1 and above, result in $\hat{\rho}_K(x, y)$ being defined within the $[-1, 1]$ range. Generally from figure 19, the stronger correlation estimates seem to occur within localised areas. For figure 19a this occurs in the small fang like area at $(1, 4)$. For figures 19c and 19d there are specks of high correlation.

To observe the impact of the smoothing parameter h, the bivariate Gaussian kernel estimated Cauchy density was chosen. The results of this are shown in figures 20 and 21. As h, increases the range of $\hat{\rho}_K(x, y)$ decreases. As seen in figure 21e the $\hat{\rho}_K(x, y)$ are so minuscule that the estimate are essentially equal to 0. On the other side for $h = 0.001$ in figure 21a there are only specks of correlation in between a larger area of undefined values. As h increases the amount of undefined areas decrease, as the difference between figures 21a and 21d is palpable. Although both figures 21c and 21d seem to show similar trends to figure 4c. As when x and y are positive, the estimated correlation is also positive. While when x and y have differing signs, the estimated correlation is negative. What figure 21 has shown is that $\hat{\rho}_K(x, y)$ has to consciously implemented for the bivariate Gaussian kernel estimates. As low of h values return a picture that can be too chaotic to read. While high values of h become too homogeneous to discern. In addition, to the fact that sub optimal h values for the fit, may give better correlation estimates. To understand why there are undefined areas for the estimated $\hat{\rho}_K(x, y)$ is harder than the function for $\hat{\rho}(x, y)$ from equation (26), as the function for $\hat{\rho}_K(x, y)$ is more obtuse . The log double derivatives of the bivariate Gaussian

kernel estimates are

$$\begin{aligned}\frac{\partial^2}{\partial x^2}\log\hat{f}(x,y,h) =& \frac{1}{h^2\Big(\sum\exp\Big[-\frac{1}{2h}((X_i-x)^2+(Y_i-y)^2)\Big]\Big)^2}\\ &\times\Bigg\{\Big(\sum\exp\Big[-\frac{1}{2h}((X_i-x)^2+(Y_i-y)^2)\Big]\Big)\\ &\times\Big(-h\sum\exp\Big[-\frac{1}{2h}((X_i-x)^2+(Y_i-y)^2)\Big]\\ &+\sum(X_i-x)^2\exp\Big[-\frac{1}{2h}((X_i-x)^2+(Y_i-y)^2)\Big]\Big)\\ &-\Big(\sum X_i-x)\exp\Big[-\frac{1}{2h}((X_i-x)^2+(Y_i-y)^2)\Big]\Big)^2\Bigg\},\end{aligned} \tag{57}$$

$$\begin{aligned}\frac{\partial^2}{\partial x^2}\log\hat{f}(x,y,h) =& \frac{1}{h^2\Big(\sum\exp\Big[-\frac{1}{2h}((X_i-x)^2+(Y_i-y)^2)\Big]\Big)^2}\\ &\times\Bigg\{\Big(\sum\exp\Big[-\frac{1}{2h}((X_i-x)^2+(Y_i-y)^2)\Big]\Big)\\ &\times\Big(-h\sum\exp\Big[-\frac{1}{2h}((X_i-x)^2+(Y_i-y)^2)\Big]\\ &+\sum(Y_i-y)^2\exp\Big[-\frac{1}{2h}((X_i-x)^2+(Y_i-y)^2)\Big]\Big)\\ &-\Big(\sum(Y_i-y)\exp\Big[-\frac{1}{2h}((X_i-x)^2+(Y_i-y)^2)\Big]\Big)^2\Bigg\},\end{aligned} \tag{58}$$

and

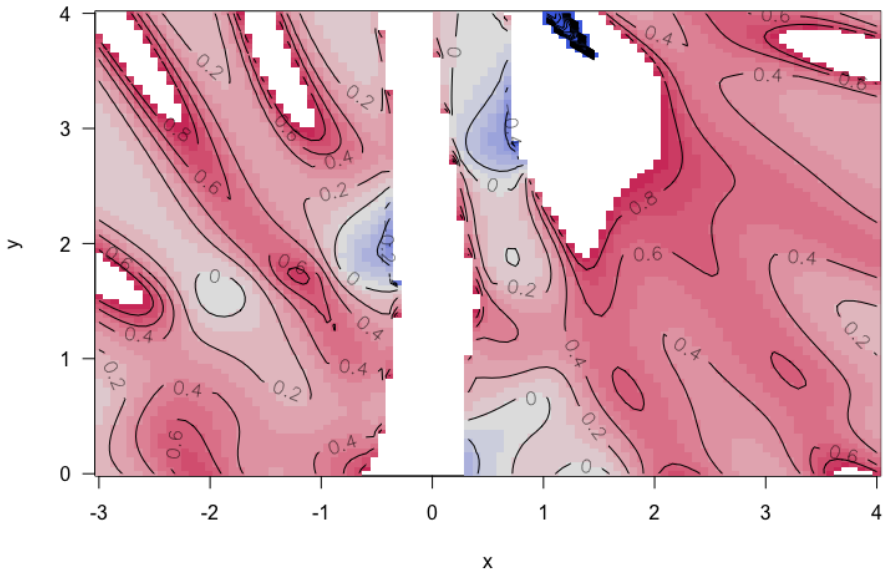
$$\begin{aligned}\frac{\partial^2}{\partial x \partial y} \log \hat{f}(x,y,h) =& \frac{\frac{1}{h^2}\sum (x-X_i)(y-Y_i)\exp\left[-\frac{1}{2h}((x-X_i)^2+(y-Y_i)^2)\right]}{\sum \exp\left[-\frac{1}{2h}((x-X_i)^2+(y-Y_i)^2)\right]} \\ & - \frac{1}{\sum \exp\left[-\frac{1}{2h}((x-X_i)^2+(y-Y_i)^2)\right]^2} \\ & \times \left\{\left(-\frac{1}{h}\sum (x-X_i)\exp\left[-\frac{1}{2h}((x-X_i)^2+(y-Y_i)^2)\right]\right)\right. \\ & \left.\times \left(-\frac{1}{h}\sum (y-Y_i)\exp\left[-\frac{1}{2h}((x-X_i)^2+(y-Y_i)^2)\right]\right)\right\}. \end{aligned} \tag{59}$$

As shown these three double derivatives are long, and because of the amount of observations complex. In addition, they cannot be reduced, which creates a needlessly complex formula for $\hat{\rho}_K(x,y)$. One can find where the white areas occur, but even that is not really reliable nor fully understood. As all parts of $\hat{\rho}_K(x,y)$ contain

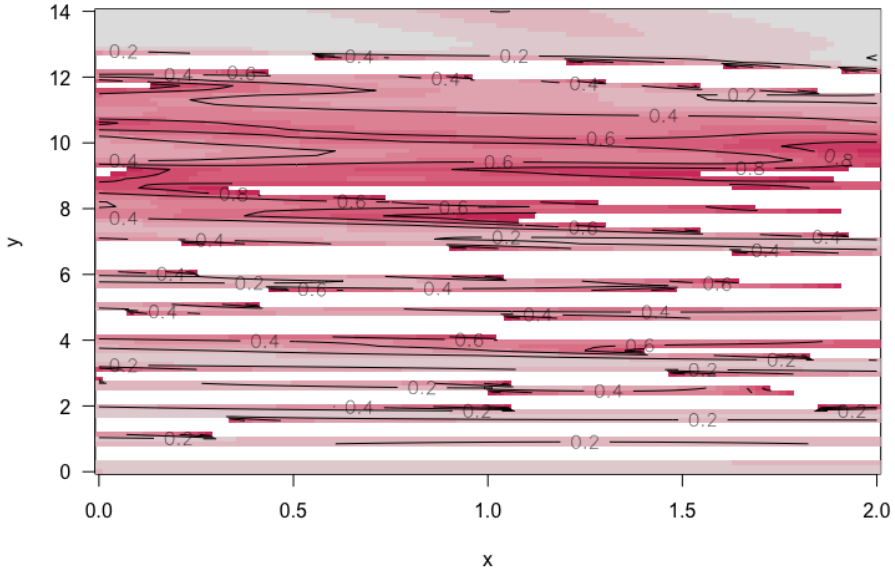
$$\frac{\partial^2}{\partial x^2} \log \hat{f}(x,y,h) \frac{\partial^2}{\partial y^2} \log \hat{f}(x,y,h) - \frac{\partial^2}{\partial x \partial y} \log \hat{f}(x,y,h)^2, \tag{60}$$

in their denominator. The equation (60) results in figure 22. For figure 22, the areas that are less than 0 are areas that are undefined in figures 21c and 21b. Although the area at $(-2,2)$ in figure 22b is 0 or lower, the same area in figure 21c is defined.

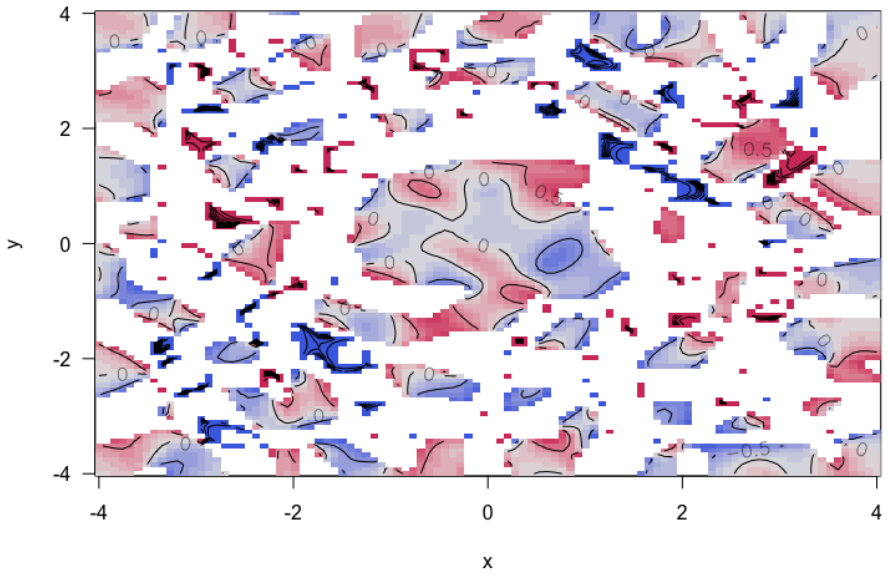
(a) $h = 0.119$



(b) $h = 0.08$



(c) $h = 0.093$



(d) $h = 0.28$

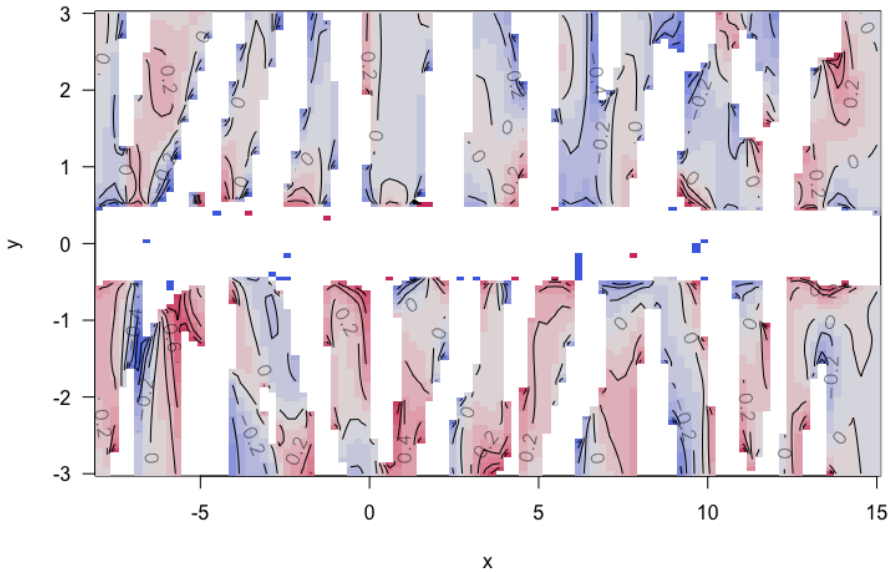
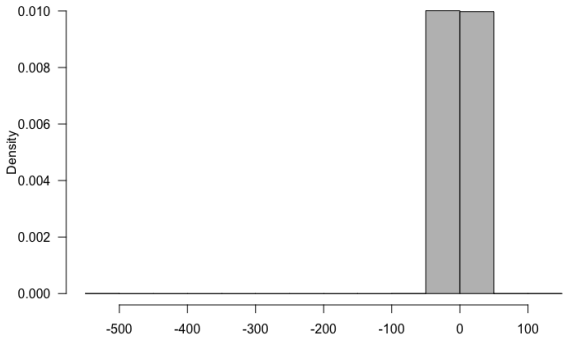
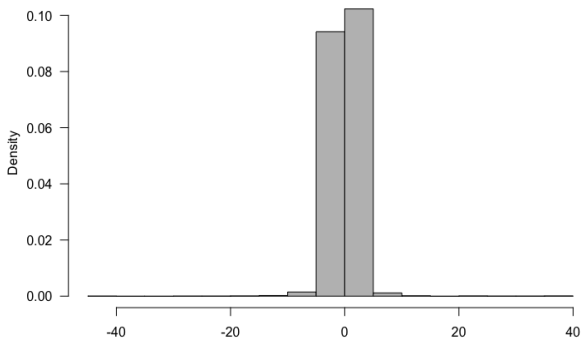


Figure 19: $\hat{\rho}_K(x, y)$ estimates for the bivariate Gaussian kernel estimates of the densities. The densities represented are *a*) Pear density for $h = 0.119$, b) twisted Pear for $h = 0.08$, c) Cauchy for $h = 0.093$ and d) transformed normal density for $h = 0.28$.

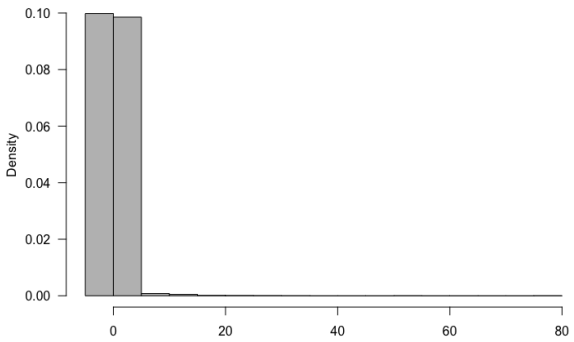
(a) $h = 0.001$



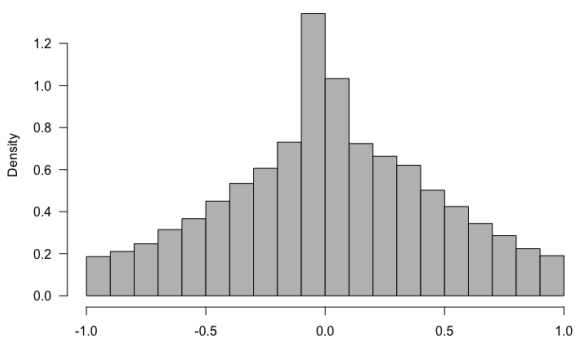
(b) $h = 0.093$



(c) $h = 0.5$



(d) $h = 1$



(e) $h = 100$

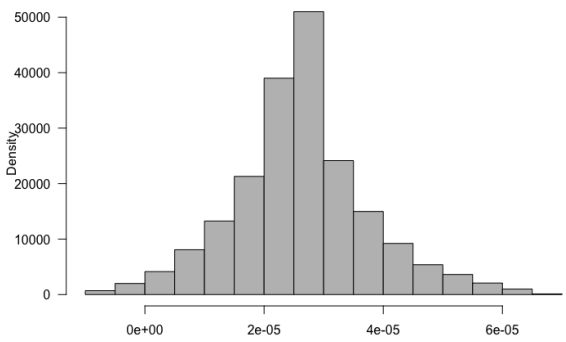
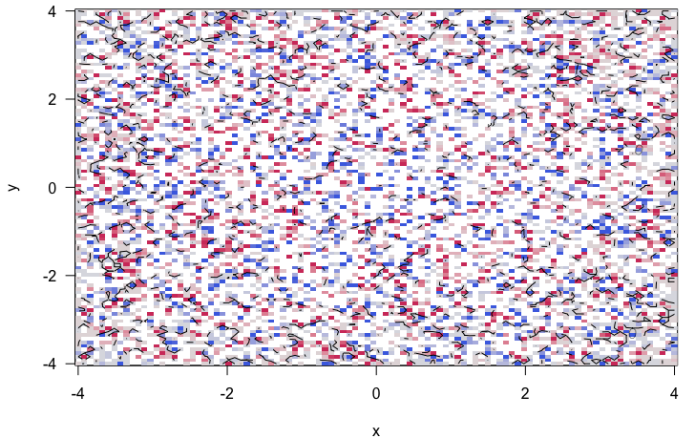
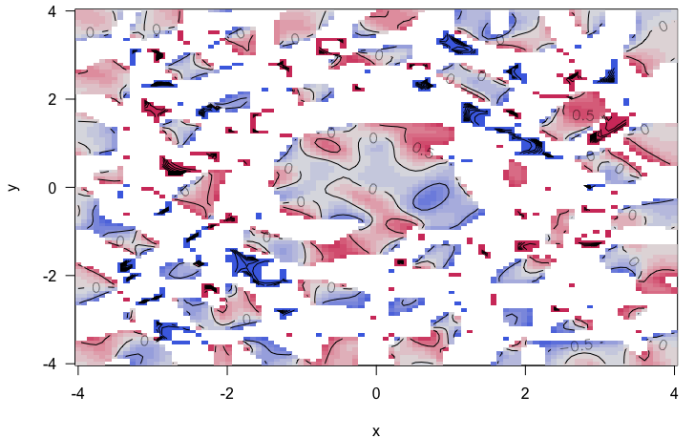


Figure 20: $\hat{\rho}(x, y)$ estimates for 5 values of h for the Gaussian kernel estimate of the Cauchy density. $a)h = 0.001$, $b)h = 0.093$, $c)h = 0.5$,$d)h = 1$, $e)h = 100$.

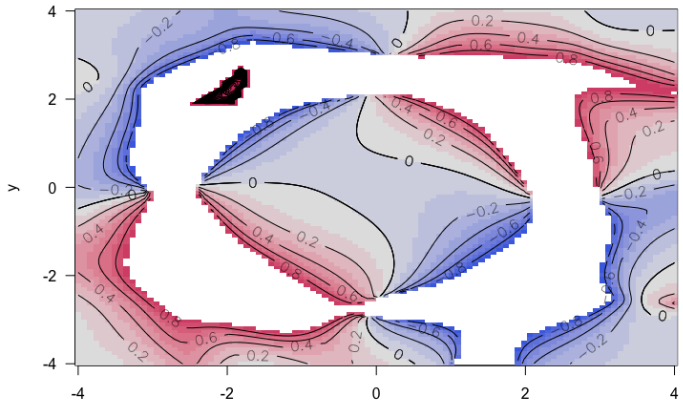
(a) $h = 0.001$



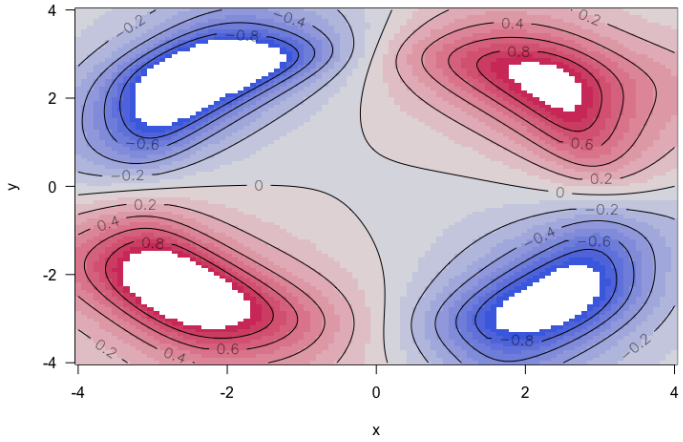
(b) $h = 0.093$



(c) $h = 0.5$



(d) $h = 1$



(e) $h = 100$

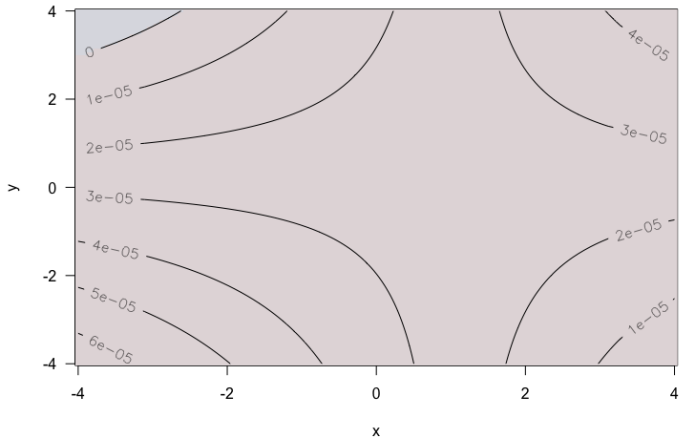
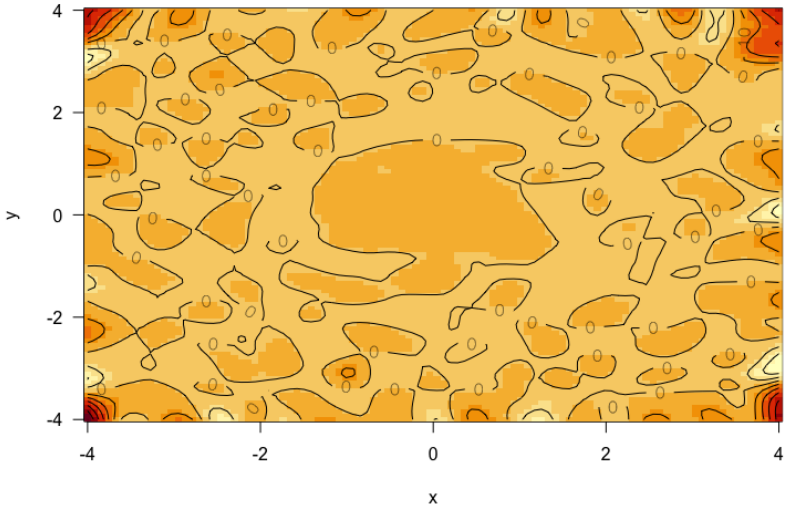


Figure 21: Contours of $\hat{\rho}(x, y)$ for the Gaussian kernel estimates of Cauchy simulated data at 5 different h values. a) $h = 0.001$, b) $h = 0.093$, c) $h = 0.5$, d) $h = 1$ and e) $h = 100$.

(a) $h = 0.093$



(b) $h = 0.5$

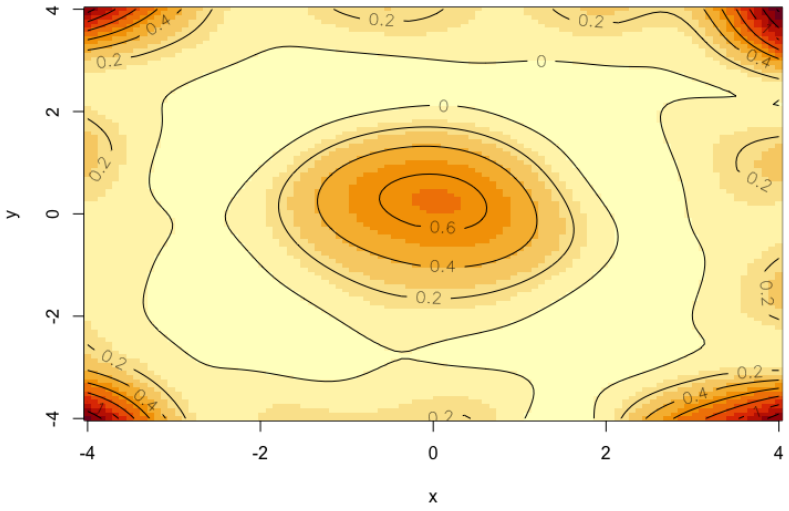


Figure 22: $\frac{\partial^2}{\partial x^2} \log \hat{f}(x, y, h) \frac{\partial^2}{\partial y^2} \log \hat{f}(x, y, h) - \frac{\partial^2}{\partial x \partial y} \log \hat{f}(x, y, h)^2$ for the bivariate Gaussian kernel estimates on the Cauchy simulated data for 2 h values. a) $h = 0.093$ and b) $h = 0.5$.

11 Conclusions and Future Perspective

Using the methods outlined in this thesis, we are able to establish a connection between the local Gaussian correlation and the $\hat{\rho}(x, y)$ estimate from the precision matrix. However, there remain limitations. As the results show, the correlation estimates give differing estimations for certain densities. The other thing of note is that the local Gaussian correlation is defined over the entire area of the densities, while the correlation estimate of the precision matrix results in undefined areas. As presented in the experiments section, the precision matrix's correlation estimate will always be equal to the correlation coefficient for any bivariate normal density. For example the bivariate normal density in figure 23, $\hat{\rho}(x, y) = \rho = 0.5$ with the log double derivatives taking on the form in equation (24). The problem of sample size for the local Gaussian correlation is shown in figure 23. For the chosen sample sizes, it is first at 10 000 observations that almost all areas start to converge to the true ρ. Another problem could be because the 4 densities used in this thesis are not typical densities, nor do they reduce when derived. The best example of the derivative not reducing the function, is the twisted Pear density. The reason why the twisted Pear is a good example is because of the variables

$$\mu(x) = \frac{x}{10}\exp(5 - \frac{x}{2}), \tag{61}$$

and

$$\sigma(x) = (\frac{1 + 0.5x}{3}), \tag{62}$$

which for the part $\frac{-1}{2}(\frac{y - \mu(x)}{\sigma(x)})^2$ when derived by x results in

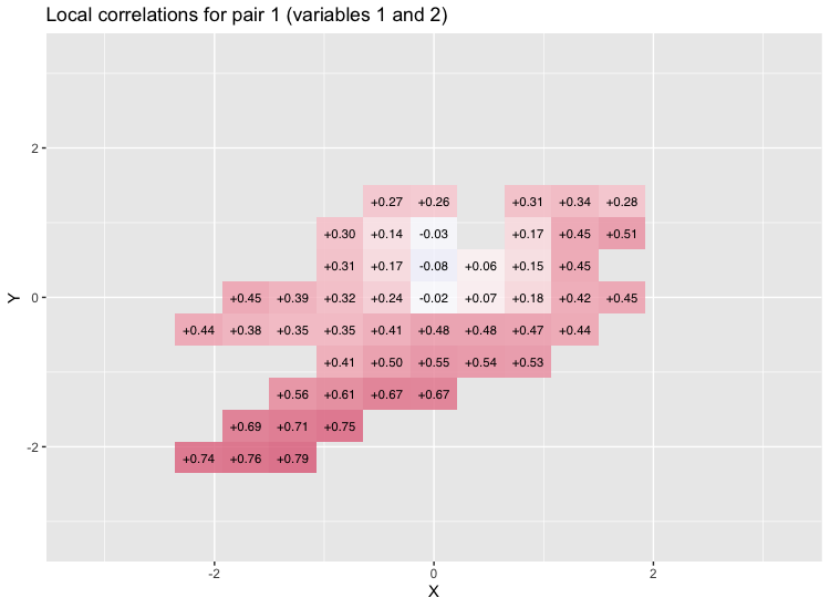
$$\begin{aligned}\frac{\partial}{\partial x}\frac{-1}{2}\left(\frac{y-\mu(x)}{\sigma(x)}\right)^2 =& \frac{-9}{2}\Bigg\{\left[\frac{y}{5}(0.5x-1)\exp(5-0.5x)\right.\\ &\left.+\frac{x}{5}(0.5x-1)\exp(10-x)\right]\left[1+x+\frac{x^2}{4}\right]\\ &-\left(1+2x\right)\left(y^2-\frac{xy}{5}\exp(5-0.5x)-\frac{x^2}{10}\exp(10-x)\right)\Bigg\}\\ &\Big/\left(1+x+0.25x^2\right)^2.\end{aligned} \tag{63}$$

Already from the first derivative, the resulting function has increased in both length and complexity. There is another jump in complexity for the double derivatives. Thus illuminating the fact that $\hat{\rho}(x, y)$ ends up being a highly complex function.

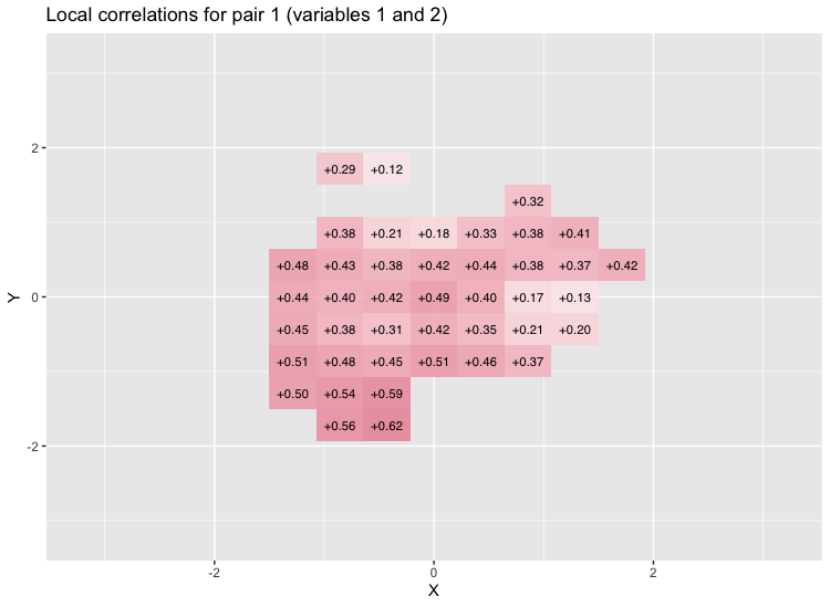
The Box-Cox transformation and the bivariate Gaussian kernel estimate were studied late in the process. To observe if they can be incorporated in future analysis. The Box-Cox transformation would be the easier of the two to further study, as the derivatives for the densities are less complex than the ones from the bivariate Gaussian kernel estimates. The problem for the bivariate Gaussian kernel estimate, is that the derivatives are not reducible. Thus each sum in equations (57), (58) and (59) would be complex functions containing number of parts equal to the number of observations. As the double derivatives occur multiple times in $\hat{\rho}_K(x, y)$, the resulting function requires more work than the scope of this thesis, but remains a promising avenue. There is also the possibility of using other bivariate kernel estimates, for estimating the densities.

In conclusion, a bridge between the local Gaussian correlation and the correlation estimate from the precision matrix has been established. Although further work is needed, to clarify remaining problems.

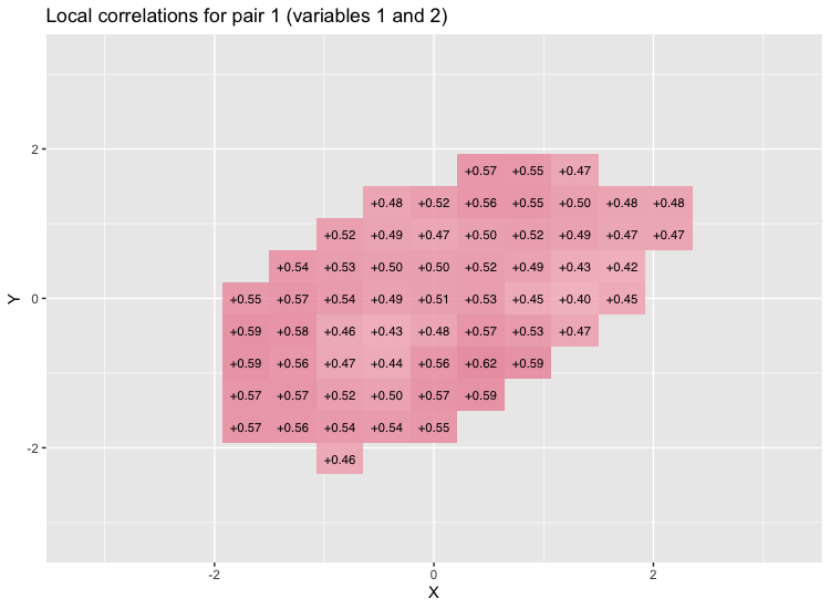
(a) $n = 100$



(b) $n = 1\ 000$



(c) $n = 10\ 000$



(d) $n = 100\ 000$

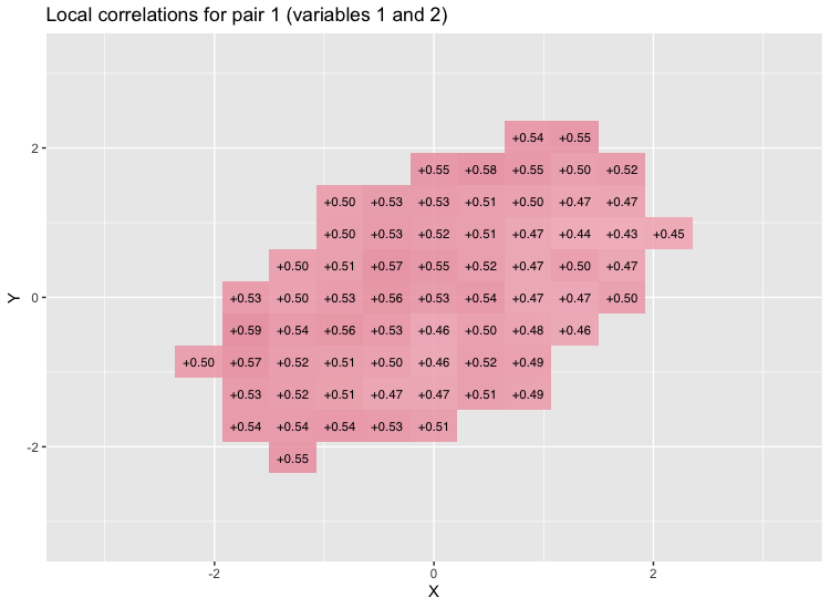


Figure 23: local Gaussian correlation for a bivariate normal density ($N(0, 0, 1, 1, 0.5)$) for 4 different n sample sizes. a) $n = 100$, b) $n = 1\ 000$, c) 10 000 and d) 100 000.

References

G. D. Berentsen, B. Støve, D. Tjøstheim, and T. Nordbø. Recognizing and visualizing copulas: An approach using local gaussian approximation. *Insurance: Mathematics and Economics*, 57:90–103, 2014. ISSN 0167-6687. doi: https://doi.org/10.1016/j.insmatheco.2014.04.005. URL https://www.sciencedirect.com/science/article/pii/S0167668714000432.

G. E. P. Box and D. R. Cox. An analysis of transformations. *Journal of the Royal Statistical Society. Series B (Methodological)*, 26(2):211–252, 1964. ISSN 00359246. URL http://www.jstor.org/stable/2984418.

C. Brokstad. Using methods of correlation to understand the covid-19 pandemic. page 36, May 2020.

K. Doksum, S. Blyth, E. Bradlow, X. Meng, and H. Zhao. Correlation curves as local measures of variance explained by regression. *Journal of the American Statistical Association*, 89(426):571–582, June 1994. URL https://www.jstor.org/stable/2290860?seq=1.

Ramesh C. Gupta, S.N.U.A. Kirmani, and H.M. Srivastava. Local dependence functions for some families of bivariate distributions and total positivity. *Applied Mathematics and Computation*, 216(4):1267–1279, 2010. ISSN 0096-3003. doi: https://doi.org/10.1016/j.amc.2010.02.019. URL https://www.sciencedirect.com/science/article/pii/S0096300310001876.

N. L. Hjort and M. C. Jones. Locally parametric nonparametric density estimation. *The Annals of Statistics*, 24(4):1619–1647, August 1996.

H. Ivanka. Kernel smoothing in matlab : theory and practice of kernel smoothing, 2012.

M.C. Jones. The local dependency function. *Biometrika*, pages 899–904, 1996. URL https://academic.oup.com/biomet/article/83/4/899/253542.

M.C. Jones. Constant local dependence. *Journal of Multivariate analysis*, 64:148–155, February 1998.

M.C. Jones and Koch. Dependence maps: Local dependence in practice. *Statistics and computing*, 13(3):241–255, 2003. ISSN 0960-3174.

L. A. Jordanger and D. Tjøstheim. Nonlinear spectral analysis: A local gaussian approach. *Journal of the American Statistical Association*, 0(0): 1–18, 2021. doi: 10.1080/01621459.2020.1840991. URL https://doi.org/10.1080/01621459.2020.1840991.

K. Kristensen, A. Nielsen, C. W. Berg, H. Skaug, and B. M. Bell. TMB: Automatic differentiation and Laplace approximation. *Journal of Statistical Software*, 70(5):1–21, 2016. doi: 10.18637/jss.v070.i05.

R. Murray. Remarks on some nonparametric estimates of a density function. *The Annals of mathematical statistics*, 27(3):832–837, 1956. ISSN 0003-4851.

Q. N. Nguyen, S. Aboura, J. Chevallier, L. Zhang, and B. Zhu. Local gaussian correlations in financial and commodity markets. *European journal of operational research*, 285(1):306–323, 2020. ISSN 0377-2217.

H. Otneim. Locally gaussian distributions: Estimation and methods, December 2019. URL https://cran.r-project.org/web/packages/lg/lg.pdf.

K. Pearson. Mathematical contributions to the theory of evolution. iii. regression, heredity, and panmixia. *Philosophical Transactionsof the Royal Society of London. Series A, Containing Papers of a Mathematical or a Physical Character*, 187:253–318, 1896. URL https://www.jstor.org/stable/90707.

D. W. Scott. Multivariate density estimation : theory, practice, and visualization. Wiley, 2015.

B.W Silverman. Density estimation for statistics and data analysis. Chapman and Hall, 1986.

B. Støve, D. Tjøstheim, and K. O. Hufthammer. Using local gaussian correlation in a nonlinear re-examination of financial contagion. *Journal of Empirical Finance*, 25:62–82, 2014. ISSN 0927-5398. doi: https://doi.org/10.1016/j.jempfin.2013.11.006. URL https://www.sciencedirect.com/science/article/pii/S0927539813000947.

D. Tjøstheim, , and K. O. Hufthammer. Local gaussian correlation: A new measure of dependence. *Journal of Econometrics*, 172(1):33–48, 2013. ISSN 0304-4076.

D. Tjøstheim, H. Otneim, and B. Støve. Statistical dependence: Beyond pearson's p. *Statistical science*, 37(1):90, 2022. ISSN 0883-4237.

Appendices

Appendix A Example of the Precision matrix's Correlation Function

The Pear's density is given as

$$f(x,y) = \frac{3x^2}{2\pi\sigma_x\sigma_y\sqrt{1-\rho^2}} \exp\Bigg[-\frac{1}{2(1-\rho^2)}\Bigg(\frac{(x^3-\mu_x)^2}{\sigma_x^2} - 2\rho\frac{(x^3-\mu_x)(y-\mu_y)}{\sigma_x\sigma_y} + \frac{(y-\mu_y)^2}{\sigma_y^2)}\Bigg)\Bigg]. \tag{64}$$

From here the log first order derivatives of the density are

$$\begin{aligned} \frac{\partial}{\partial x}\log f(x,y) &= \frac{1}{(1-\rho^2)}\Bigg(-\frac{3x^5-3\mu_x x^2}{\sigma_x^2} + \rho\frac{3x^2(y-\mu_y)}{\sigma_x\sigma_y}\Bigg) - \frac{2}{x} \\ \frac{\partial}{\partial y}\log f(x,y) &= \frac{1}{(1-\rho^2)}\Bigg(\rho\frac{(x^3-\mu_x)}{\sigma_x\sigma_y} - \frac{(y-\mu_y)}{\sigma_y^2)}\Bigg). \end{aligned} \tag{65}$$

The second order derivatives are

$$\begin{aligned} \frac{\partial^2}{\partial x^2}\log f(x,y) &= \frac{1}{(1-\rho^2)}\Bigg(-\frac{15x^4-6\mu_x x}{\sigma_x^2} + \rho\frac{6x(y-\mu_y)}{\sigma_x\sigma_y}\Bigg) + \frac{2}{x^2} \\ \frac{\partial^2}{\partial y\partial x}\log f(x,y) &= \frac{1}{(1-\rho^2)}\Bigg(\rho\frac{3x^2}{\sigma_x\sigma_y}\Bigg) \\ \frac{\partial^2}{\partial y^2}\log f(x,y) &= \frac{-1}{(1-\rho^2)\sigma_y^2}. \end{aligned} \tag{66}$$

Then with the double derivatives, we can put them into the precision matrix

$$\begin{bmatrix} -\frac{\partial^2}{\partial x^2} log f(x,y) & -\frac{\partial^2}{\partial x \partial y} log f(x,y) \\ -\frac{\partial^2}{\partial x \partial y} log f(x,y) & -\frac{\partial^2}{\partial y^2} log f(x,y) \end{bmatrix}$$
$$= \begin{bmatrix} \frac{1}{(1-\rho^2)}\left(\frac{15x^4 - 6\mu_x x}{\sigma_x^2} - \rho\frac{6x(y-\mu_y)}{\sigma_x\sigma_y}\right) + \frac{2}{x^2} & -\frac{1}{(1-\rho^2)}\left(\rho\frac{3x^2}{\sigma_x\sigma_y}\right) \\ -\frac{1}{(1-\rho^2)}\left(\rho\frac{3x^2}{\sigma_x\sigma_y}\right) & \frac{1}{(1-\rho^2)\sigma_y^2} \end{bmatrix} \tag{67}$$

From here we take the inverse as we know the inverse of the precision matrix is the covariance matrix.

$$\frac{1}{\frac{1}{(1-\rho^2)}\left(\frac{15x^4 - 6\mu_x x}{\sigma_x^2} - \rho\frac{6x(y-\mu_y)}{\sigma_x\sigma_y} + \frac{2}{x^2}\right)\left(\frac{1}{(1-\rho^2)\sigma_y^2}\right) - \left(\frac{1}{(1-\rho^2)}\left(\rho\frac{3x^2}{\sigma_x\sigma_y}\right)\right)^2}$$
$$\times \begin{bmatrix} \frac{1}{(1-\rho^2)\sigma_y^2} & -\frac{1}{(1-\rho^2)}\left(\rho\frac{3x^2}{\sigma_x\sigma_y}\right) \\ -\frac{1}{(1-\rho^2)}\left(\rho\frac{3x^2}{\sigma_x\sigma_y}\right) & \frac{1}{(1-\rho^2)}\left(\frac{15x^4 - 6\mu_x x}{\sigma_x^2} - \rho\frac{6x(y-\mu_y)}{\sigma_x\sigma_y}\right) \end{bmatrix}. \tag{68}$$

As the $(1,2)$ and $(2,1)$ represent $\rho\sigma_x\sigma_y$ for the equation(68). For the equation (68), $(1,1)$ represents σ_x^2 and $(2,2)$ represents σ_y^2. Thus to find $\hat{\rho}(x,y)$ we take $(1,2)$ divided by the square root of $(2,2)$ and $(1,1)$ so then

for the Pear density, $\hat{\rho}(x, y)$ is

$$\frac{-\frac{1}{(1-\rho^2)}\left(\rho\frac{3x^2}{\sigma_x\sigma_y}\right)}{\frac{1}{(1-\rho^2)}\left(\frac{15x^4-6\mu_x x}{\sigma_x^2}-\rho\frac{6x(y-\mu_y)}{\sigma_x\sigma_y}+\frac{2}{x^2}\right)\left(\frac{1}{(1-\rho^2)\sigma_y^2}\right)-\left(\frac{1}{(1-\rho^2)}\left(\rho\frac{3x^2}{\sigma_x\sigma_y}\right)\right)^2}$$
$$\Bigg/\left(\sqrt{\frac{\frac{1}{(1-\rho^2)}\left(-\frac{15x^4-6\mu_x x}{\sigma_x^2}+\rho\frac{6x(y-\mu_y)}{\sigma_x\sigma_y}\right)+\frac{2}{x^2}}{\frac{1}{(1-\rho^2)}\left(\frac{15x^4-6\mu_x x}{\sigma_x^2}-\rho\frac{6x(y-\mu_y)}{\sigma_x\sigma_y}+\frac{2}{x^2}\right)\left(\frac{1}{(1-\rho^2)\sigma_y^2}\right)-\left(\frac{1}{(1-\rho^2)}\left(\rho\frac{3x^2}{\sigma_x\sigma_y}\right)\right)^2}}\right.$$
$$\left.\times\sqrt{\frac{\frac{1}{(1-\rho^2)\sigma_y^2}}{\frac{1}{(1-\rho^2)}\left(\frac{15x^4-6\mu_x x}{\sigma_x^2}-\rho\frac{6x(y-\mu_y)}{\sigma_x\sigma_y}+\frac{2}{x^2}\right)\left(\frac{1}{(1-\rho^2)\sigma_y^2}\right)-\left(\frac{1}{(1-\rho^2)}\left(\rho\frac{3x^2}{\sigma_x\sigma_y}\right)\right)^2}}\right). \tag{69}$$

Appendix B Examples of C++ Code

Pear density in C++

```
#include <TMB.hpp>
#include <cmath>
#include <math.h>
#include <cstdlib>

template<class Type>
Type objective_function<Type>::operator() ()
{
  PARAMETER(x);
  PARAMETER(y);
  PARAMETER(mu_1);
  PARAMETER(mu_2);
  PARAMETER(sig_1);
  PARAMETER(sig_2);
  PARAMETER(rho);

  Type U = pow(x,3);

  Type a = 2 * M_PI * sig_1 *sig_2 * sqrt(1 - pow(rho,2));
  Type b = (pow(U - mu_1, 2)/pow(sig_1,2)) - 2 * rho * (U -
     mu_1) * (y - mu_2)/(sig_1 * sig_2) + pow(y - mu_2, 2)/pow(
     sig_2,2);
  Type c = exp(-b/(2 * (1 - pow(rho,2))));;
  return log(c) - log(a) + log(3) + 2 * log(fabs(fabs(x)));
}
```

Bivariate Gaussian kernel estimate in C++

```
#include <TMB.hpp>
#include <cmath>
#include <math.h>
#include <cstdlib>

template<class Type>
Type objective_function<Type>::operator() ()
{
  DATA_VECTOR(X_i1);
  DATA_VECTOR(X_i2);
  DATA_SCALAR(h);
  PARAMETER(X);
  PARAMETER(Y);

  Type n = X_i1.size();

  Type f = Type(0.0); // Kernel smoother

  for(int i=0;i<n;i++){
    f += dnorm(X,X_i1(i),sqrt(h)) * dnorm(Y,X_i2(i),sqrt(h));
  }
  return(log(f) - log(n));
}
```

Appendix C R-code

```
###Important packages####
library(lg)
library(mcmc)
library(mvtnorm)
library(ggplot2)
library(RColorBrewer)
library(dplyr)
library(pheatmap)
library(TMB)
dat <- read.table("Master/data_pear_bivar.txt", header = TRUE)

###rho matrix function and Reverse gamma function####
###rho matrix function###
matrix_function <- function(z_dx2, z_dxdy, z_dy2, row_length =
    1000){
  #asssumes the data is the form of a 1000 times 1000 matrix
  z_rho <- matrix(data = rep(0,row_length^2), nrow = row_length
      , ncol = row_length)
  z_sig_x <- matrix(data = rep(0,row_length^2), nrow = row_
     length, ncol = row_length)
  z_sig_y <- matrix(data = rep(0,row_length^2), nrow = row_
     length, ncol = row_length)

  for (i in c(1:row_length)){
    for(j in c(1:row_length)){
      poten_matrix <- matrix(data = c(-z_dx2[i,j], -z_dxdy[i,j
         ],
                                      -z_dxdy[i,j], -z_dy2[i,j]),
                                         nrow = 2, ncol = 2)
      if (anyNA(poten_matrix) == TRUE){
        z_val[i,j] <- NaN
      }
      if(z_dx2[i,j] == z_dy2[i,j] & z_dx2[i,j] == z_dxdy[i,j]){
        z_val[i,j] <- NaN
      }
```

```
      else{
        sol_matrix <- solve(poten_matrix)
        z_rho[i,j] <- sol_matrix[1,2]/(sqrt(sol_matrix[1,1]) *
           sqrt(sol_matrix[2,2]))
        z_sig_x[i,j] <- sol_matrix[1,1]
        z_sig_y[i,j] <- sol_matrix[2,2]
      }
    }
  }
  return(list(z_rho = z_rho, z_sig_x = z_sig_x, z_sig_y = z_sig
     _y))
}

##Reverse gamma function##
reverse_gam <- function(gam){
  return((-1 + sqrt(1 + 4*gam^2))/(2*gam))
}

##extreme val function##
extreme_val_function <- function(z_vals, min_val_testing = 0,
    max_val_testing = 10){
  i_val <- c()
  j_val <- c()
  for(i in c(1:1000)){
    for(j in c(1:1000)){
      if (anyNA(z_vals[i,j]) == FALSE){
        if(z_vals[i,j] <= min_val_testing | z_vals[i,j] >= max_
           val_testing){
          i_val <- c(i_val, i)
          j_val <- c(j_val, j)
        }
      }
    }
  }
  return(matrix(c(i_val, j_val), ncol = 2))
}
```

```
###Kernel example####
x_seq <- seq(-1.5, 1.5, length.out = 1000)

###Uniform kernel###
uniform_kernel_func <- function(x){
  if(abs(x) <= 1){
    return(1/2)
  }
  else{
    return(0)
  }
}

uniform_vals <- rep(0,1000)
for(i in c(1:1000)){
  uniform_vals[i] <- uniform_kernel_func(x_seq[i])
}
#Figure 16. a)
plot(x_seq, uniform_vals, type = "l", xlab = "x", ylab = "",
    ylim = c(0,1), lwd = 5, las = 1)

###Triangle kernel###
triangle_kernel_func <- function(x){
  if(abs(x) <= 1){
    return(1 - abs(x))
  }
  else{
    return(0)
  }
}

triangle_vals <- rep(0,1000)
for(i in c(1:1000)){
  triangle_vals[i] <- triangle_kernel_func(x_seq[i])
}
#Figure 16. b)
plot(x_seq, triangle_vals, type = "l", xlab = "x", ylab = "",
    lwd = 5, las = 1)
```

```
###Gaussian kernel###
x_seq_gauss <- seq(-5,5, length.out = 1000)

gaussian_kernel_func <- function(x){
  return((1/(sqrt(2*pi))) * exp(-0.5 * x^2))
}

gaussian_vals <- rep(0,1000)
for(i in c(1:1000)){
  gaussian_vals[i] <- gaussian_kernel_func(x_seq_gauss[i])
}
#Figure 16. c)
plot(x_seq_gauss, gaussian_vals, xlim = c(-3,3), ylim = c(0,1),
     type = "l", xlab = "x", ylab = "", lwd = 5, las = 1)

###Epanechnikov###
Epanechnikov_kernel_func <- function(x){
  if(abs(x) <= 1){
    return(3/4 * (1 - x^2))
  }
  else{
    return(0)
  }
}

Epanechnikov_vals <- rep(0,1000)
for(i in c(1:1000)){
  Epanechnikov_vals[i] <- Epanechnikov_kernel_func(x_seq[i])
}
#Figure 16. d)
plot(x_seq, Epanechnikov_vals, ylim = c(0,1), type = "l", xlab
    = "x", ylab = "", lwd = 5, las = 1)

###Rho example and sigma functions ####
lnx_func <- function(x){
  return(sin(3*x))
}
```

```
set.seed(3)
x_sample <- runif(1000, min = -4,max = 4)
y_sample <- lnx_func(x_sample) + rnorm(1000)
#Figure 1.
plot(x_sample, y_sample, xlab = "x", ylab = "y", las = 1)

###neg sigma^2_x for pear###:
neg_sig_pear_func <- function(x){
  a <- (39.75 * x[1]^6) - (232.2580645 * x[1]^3 *x[2]) - (120 *
      x[1]^3) + 350
  return(a)
}

x <- seq(-3,4, length.out = 1000)
y <- seq(0,4, length.out = 1000)
z_mat <- matrix(rep(0,1000^2), nrow = 1000, ncol = 1000)

for(i in c(1:1000)){
  for(j in c(1:1000)){
    if(anyNA(neg_sig_pear_func(c(x[i],y[j]))) == FALSE){
      if(neg_sig_pear_func(c(x[i], y[j])) < c(0)){
        z_mat[i,j] <- neg_sig_pear_func(c(x[i], y[j]))
      }
      else{
        z_mat[i,j] <- NaN
      }
    }
  }
}

#Figure 11.
image(x,y,z_mat, xlab = "X", ylab = "Y", las = 1)
contour(x,y,z_mat,labcex = 1 ,add = TRUE)

###Compiling Gaussian kernel ####
data_gauss_kern <- read.table("Master/Gauss_kern_data.txt")
compile("Master/Gauss_kern.cpp")
```

```
dyn.load(dynlib("Master/Gauss_kern"))

###Pear####
Parameters <- list(x = 0, y = 0, mu_1 = 10, mu_2 = 1.55, sig_1
   = 10, sig_2 = 0.775, rho = 0.75)

compile("Master/pear_bivar.cpp")
dyn.load(dynlib("Master/pear_bivar"))

obj <- MakeADFun(data = dat, parameters = Parameters, DLL = "
   pear_bivar")

x <- seq(-3,4,length.out =100)
y <- seq(0,4,length.out =100)
x_ldf <- seq(-10,10,length.out =100)
y_ldf <- seq(-30,30,length.out =100)
z <- matrix(data = c(rep(0,10000)), nrow = 100,ncol = 100)
for (i in c(1:100)){
  for (j in c(1:100)){
    z[i,j] <- exp(obj$fn(c(x[i], y[j] , 10, 1.55, 10, 0.775,
       0.75)))
  }
}
#Figure 2. a)
image(x, y, z, las = 1)
contour(x, y, z, zlim = c(0.025,0.5), xlim = c(0,4), ylim = c
   (0,4), labcex = 1, add = TRUE)

##MCMC sampling pear##
bivar_pear <- function(x){
  U = x[1]^3
  a = 1/(2 * pi * 10 * 0.775* sqrt(1- 0.75^2))
  b = (-1/(2 * (1 - 0.75^2))) * (((U - 10)/ 10)^2 - 2 * 0.75 *
                                  ((U - 10)/ 10) * ((x[2] - 1.55)
                                     /0.775) + ((x[2] - 1.55)/
                                     0.775)^2)
  return (a * exp(b) * 3 *abs(x[1]) * abs(x[1]))
}
```

```
log_bivar_pear <- function(x){
  return(log(bivar_pear(x)))
}

set.seed(123)
mcmc_pear <- metrop(log_bivar_pear,initial = rep(4,2), blen =
    1,nspac = 1,scale =1.05,nbatch = 10000)
mcmc_pear_sample = mcmc_pear$batch
#Figure 2. b)
plot(mcmc_pear_sample, xlab = "x", ylab = "y", las = 1)
mcmc_pear$accept

##LGC for Pear##
lg.pear <- lg_main(mcmc_pear_sample, est_method = "1par")
x <- seq(-3, 4, length.out = 15)
y <- seq(0, 4, length.out = 15)
d1 <- expand.grid(x,y)
zd1 <- dlg(lg.pear, grid = d1)
#Figure 2. d)
corplot(zd1, plot_thres = 0.02, low_color = "#4A6FE3", high_
    color = "#D33F6A", label_size = 4)

##matrix function##
#remember all of the values are for 1000 squared
#double derivative values
x <- seq(-3,4,length.out =1000)
y <- seq(0,4,length.out =1000)
z_dxdy <- matrix(data = c(rep(0,1000000)), nrow =1000, ncol =
    1000)
z_dx2 <- matrix(data = c(rep(0,1000000)), nrow = 1000, ncol =
    1000)
z_dy2 <- matrix(data = c(rep(0,1000000)), nrow = 1000, ncol =
    1000)

for(i in c(1:1000)){
  for(j in c(1:1000)){
    z_dxdy[i,j] <- obj$he(c(x[i], y[j], 10, 1.55, 10, 0.775,
```

```
      0.75))[2,1]
    z_dx2[i,j] <- obj$he(c(x[i], y[j], 10, 1.55, 10, 0.775,
      0.75))[1,1]
    z_dy2[i,j] <- obj$he(c(x[i], y[j], 10, 1.55, 10, 0.775,
      0.75))[2,2]
  }
}

pear_doub_derivs <- matrix_function(z_dx2 = z_dx2, z_dxdy = z_
   dxdy, z_dy2 = z_dy2)
z_rho <- pear_doub_derivs$z_rho
z_sig_x <- pear_doub_derivs$z_sig_x
z_sig_y <- pear_doub_derivs$z_sig_y

#plot
#Figure 2. c)
image(x,y,z_rho, col = c("#E4D3D6", "#E6C4C9", "#E6B4BD", "#
   E5A5B1", "#E495A5",
                        "#E28699", "#DF758D", "#DB6581", "#
                           D75376", "#D33F6A"), las = 1)
contour(x,y,z_rho, add = TRUE, labcex = 1)

z_sig_x_trunk <- z_sig_x[z_sig_x <1 & z_sig_x > -1]
z_sig_y_trunk <- z_sig_y[z_sig_y <1 & z_sig_y > -1]
#Figure 6. a) and b)
hist(z_sig_x[z_sig_x <1 & z_sig_x > -1], main = "", xlab = "",
   freq = FALSE, col = "grey", las = 1)
hist(z_sig_y[z_sig_y <1 & z_sig_y > -1], main = "", xlab = "",
   freq = FALSE, col = "grey", las = 1)
length(z_sig_x)- length(z_sig_x_trunk)
length(z_sig_y)- length(z_sig_y_trunk)

#Extreme values for the pear distribution#
pear_sig_x_quatiles <- quantile(z_sig_x, probs = seq(0,1,0.05))
pear_sig_y_quatiles <- quantile(z_sig_y, probs = c
   (0,0.05,0.1,0.9,0.95,1))

pear_sig_x_max_vals <- extreme_val_function(z_vals = z_sig_x,
```

```
   min_val_testing = -1e7, max_val_testing = 10)
pear_sig_x_min_vals <- extreme_val_function(z_vals = z_sig_x,
   min_val_testing = -10, max_val_testing = 1e8)
pear_sig_y_max_vals <- extreme_val_function(z_vals = z_sig_y,
   min_val_testing = -1e7, max_val_testing = 10)
pear_sig_y_min_vals <- extreme_val_function(z_vals = z_sig_y,
   min_val_testing = -10, max_val_testing = 1e8)

#plots
#Figure 10. a)
plot(x[pear_sig_x_max_vals[,1]], y[pear_sig_x_max_vals[,2]],
   type = "l", xlab = "X", ylab = "Y", las = 1)
lines(x[pear_sig_x_min_vals[,1]], y[pear_sig_x_min_vals[,2]],
   col = "red")

#Figure 10. b)
plot(x[pear_sig_y_max_vals[,1]], y[pear_sig_y_max_vals[,2]],
   type = "l", xlab = "X", ylab = "Y", las = 1)
lines(x[pear_sig_y_min_vals[,1]], y[pear_sig_y_min_vals[,2]],
   col = "red")

###Gaussian kernel estimates###
data_list <- list(X_i1 = mcmc_pear_sample[,1], X_i2 = mcmc_pear
   _sample[,2], h = 0.119)
params <- list(X = 1, Y = 1)
pear_kern_func <- MakeADFun(data = data_list, parameters =
   params, DLL = "Gauss_kern")

##density est##
bw.nrd(mcmc_pear_sample)
bw.nrd0(mcmc_pear_sample) #rule of thumb bandwidth used
x <- seq(-3,4, length.out = 100)
y <- seq(0,4, length.out = 100)
z <- matrix(rep(0,100^2), nrow = 100, ncol = 100)
z_dx2 <- matrix(rep(0,100^2), nrow = 100, ncol = 100)
z_dy2 <- matrix(rep(0,100^2), nrow = 100, ncol = 100)
z_dxdy <- matrix(rep(0,100^2), nrow = 100, ncol = 100)
```

```
for(i in c(1:100)){
  for(j in c(1:100)){
    z[i,j] <- pear_kern_func$fn(c(x[i], y[j]))
    pear_kern_he <- pear_kern_func$he(c(x[i], y[j]))
    z_dx2[i,j] <- pear_kern_he[1,1]
    z_dy2[i,j] <- pear_kern_he[2,2]
    z_dxdy[i,j] <- pear_kern_he[1,2]
  }
}

#Figure 17. a)
image(x,y,exp(z), las = 1)
contour(x,y,exp(z), labcex = 1 ,add = TRUE)

##Double derivative
pear_kern_deriv <- matrix_function(z_dx2 = z_dx2,z_dxdy = z_
    dxdy, z_dy2 = z_dy2, row_length = 100)

#Figure
hist(pear_kern_deriv$z_rho, main = "", xlab = "")
#Figure 19. a)
image(x,y,pear_kern_deriv$z_rho, breaks = c(-15,seq(-1,1,
    length.out = 21)), col = hcl.colors(21, "BLUE␣RED␣2"), las
    = 1)
contour(x,y,pear_kern_deriv$z_rho, add = TRUE, nlevels = 70,
    labcex = 1)

###Pear 2.####
param_pear_2 <- list(x = 0, y = 0, mu_1 = 1.2, sig_1 = 1/3)

require(TMB)
compile("Master/pear_bivar2.cpp")
dyn.load(dynlib("Master/pear_bivar2"))

obj <- MakeADFun(data = dat, parameters = param_pear_2, DLL = "
    pear_bivar2")

x <- seq(0,2,length.out =100)
```

```
y <- seq(0,15,length.out =100)
z <- matrix(data = c(rep(0,10000)), nrow = 100,ncol = 100)

for (i in c(1:100)){
  for (j in c(1:100)){
    z[i,j] <- exp(obj$fn(c(x[i], y[j] , 1.2, 1/3)))
  }
}
#Figure 3. a)
image(x,y,z, las = 1)
contour(x,y,z, zlim = c(0.01,0.9), xlim = c(0,2), ylim = c
   (0,15), labcex = 1, add = TRUE)

##MCMC sampling pear2##
bivar_pear2 <- function(x){
  a = 1/((1/3) * sqrt(2* pi))
  b = - 0.5 * ((x[1] - 1.2)/(1/3))^2
  f_x = a * exp(b)

  mu_2 = (x[1]/10) * exp(5 - x[1]/2)
  sig_2 = (1 + 0.5 * x[1])/3

  c = 1/(sig_2 * sqrt(2 * pi))
  d = - 0.5 * ((x[2] - mu_2)/sig_2)^2
  f_y = c * exp(d);

  return (f_x * f_y)
}

log_bivar_pear2 <- function(x){
  return(log(bivar_pear2(x)))
}

set.seed(333)
mcmc_pear2 <- metrop(log_bivar_pear2,initial = rep(0,2), blen =
    1,nspac = 1,scale = 0.4,nbatch = 10000)
mcmc_pear2_sample = mcmc_pear2$batch
#Figure 3. b)
```

```
plot(mcmc_pear2_sample, xlab = "x", ylab = "y", las = 1)
mcmc_pear2$accept

##LGC for Pear2##
lg.pear2 <- lg_main(mcmc_pear2_sample, est_method = "1par")
x <- seq(-0.2, 2.3, length.out = 15)
y <- seq(-3.3, 12.7, length.out = 15)
d1 <- expand.grid(x,y)
zd1 <- dlg(lg.pear2, grid = d1)
#Figure 3. d)
corplot(zd1, low_color = "#4A6FE3", high_color = "#D33F6A",
   plot_thres = 0.03, label_size = 4)

##matrix function##
#remember all of the values are for 1000 squared
#double derivative values
x <- seq(0,2,length.out =1000)
y <- seq(0,15,length.out =1000)
z_dxdy <- matrix(data = c(rep(0,1000000)), nrow =1000, ncol =
   1000)
z_dx2 <- matrix(data = c(rep(0,1000000)), nrow = 1000, ncol =
   1000)
z_dy2 <- matrix(data = c(rep(0,1000000)), nrow = 1000, ncol =
   1000)

for(i in c(1:1000)){
  for(j in c(1:1000)){
    z_dxdy[i,j] <- obj$he(c(x[i], y[j], 1.2, 1/3))[2,1]
    z_dx2[i,j] <- obj$he(c(x[i], y[j], 1.2, 1/3))[1,1]
    z_dy2[i,j] <- obj$he(c(x[i], y[j], 1.2, 1/3))[2,2]
  }
}

pear2_doub_derivs <- matrix_function(z_dx2 = z_dx2, z_dxdy = z_
   dxdy, z_dy2 = z_dy2)
z_rho <- pear2_doub_derivs$z_rho
z_sig_x <- pear2_doub_derivs$z_sig_x
z_sig_y <- pear2_doub_derivs$z_sig_y
```

```
#plot
#Figure 3. c)
image(x,y,z_rho, col = hcl.colors(10, "Blue␣Red␣2"), las = 1)
contour(x,y,z_rho, labcex = 1, add = TRUE)

z_sig_x_trunk <- z_sig_x[z_sig_x <1 & z_sig_x > -1]
z_sig_y_trunk <- z_sig_y[z_sig_y <2 & z_sig_y > -2]
#Figure 7. a) and b)
hist(z_sig_x[z_sig_x <1 & z_sig_x > -1], main = "", xlab = "",
   freq = FALSE, col = "grey", las = 1)
hist(z_sig_y[z_sig_y <1 & z_sig_y > -1], main = "", xlab = "",
   freq = FALSE, col = "grey", las = 1)
length(z_sig_x)- length(z_sig_x_trunk)
length(z_sig_y)- length(z_sig_y_trunk)

###Gaussian kernel estimates###
data_list <- list(X_i1 = mcmc_pear2_sample[,1], X_i2 = mcmc_
   pear2_sample[,2], h = 0.08)
params <- list(X = 1, Y = 1)
pear2_kern_func <- MakeADFun(data = data_list, parameters =
   params, DLL = "Gauss_kern")

##density est##
bw.nrd(mcmc_pear2_sample)
bw.nrd0(mcmc_pear2_sample)
x <- seq(0,2, length.out = 100)
y <- seq(0,14, length.out = 100)
z <- matrix(rep(0,100^2), nrow = 100, ncol = 100)
z_dx2 <- matrix(rep(0,100^2), nrow = 100, ncol = 100)
z_dy2 <- matrix(rep(0,100^2), nrow = 100, ncol = 100)
z_dxdy <- matrix(rep(0,100^2), nrow = 100, ncol = 100)

for(i in c(1:100)){
  for(j in c(1:100)){
    z[i,j] <- pear2_kern_func$fn(c(x[i], y[j]))
    pear2_kern_he <- pear2_kern_func$he(c(x[i], y[j]))
    z_dx2[i,j] <- pear2_kern_he[1,1]
```

```
    z_dy2[i,j] <- pear2_kern_he[2,2]
    z_dxdy[i,j] <- pear2_kern_he[1,2]
  }
}
#Figure 17. b)
image(x,y,exp(z),las = 1)
contour(x,y,exp(z), labcex = 1 ,add = TRUE)

##Double derivative
pear2_kern_deriv <- matrix_function(z_dx2 = z_dx2,z_dxdy = z_
    dxdy, z_dy2 = z_dy2, row_length = 100)

hist(pear2_kern_deriv$z_rho)
#Figure 19. b)
image(x,y,pear2_kern_deriv$z_rho, col = hcl.colors(21, "BLUE␣
    RED␣2"), zlim = c(-1,1), las = 1)
contour(x,y,pear2_kern_deriv$z_rho, add = TRUE, labcex = 1,
    nlevels = 10, zlim = c(-1,1))

###Cauchy####
compile("Master/bivariate_cauchy_corr.cpp")
dyn.load(dynlib("Master/bivariate_cauchy_corr"))

param <- list(X = 0, Y = 0)

cauchy.obj <- MakeADFun(data = dat, parameters = param_alt, DLL
     = "bivariate_cauchy_corr")
x <- seq(-4,4, length.out = 100)
y <- seq(-4,4, length.out = 100)
Z_ldf <- matrix(data = c(rep(0,10000)), nrow = 100,ncol = 100)
z <- matrix(data = c(rep(0,10000)), nrow = 100,ncol = 100)

for (i in c(1:100)){
  for (j in c(1:100)){
    z[i,j] <- cauchy.obj$fn(c(x[i], y[j]))
    Z_ldf[i,j] <- cauchy.obj$he(c(x[i], y[j]))[1,2]
  }
}
```

```
#Figure 4. a)
image(x,y,exp(z), las = 1)
contour(x,y,exp(z), labcex = 1, add = TRUE)

Z_rho <- reverse_gam(Z_ldf)
#Figure 12.
image(x,y,Z_rho, xlab= "x", ylab = "y", col = hcl.colors(20,
   palette = "Blue␣Red␣2")[4:15],
     breaks = c(-0.6,-0.5,-0.4,-0.3,-0.2,-0.1,0,
        0.1,0.2,0.3,0.4,0.5,0.6), las = 1)
contour(x,y,Z_rho, labcex = 1,add = TRUE)

##MCMC sampling cauchy gamma##
log_cauchy <- function(x){
  a = pi * (sqrt((1 + x[1]^2 + x[2]^2)^3))
  return (log(1/a))
}

cauchy_alt <- function(x){
  a = pi * (sqrt((1 + x[1]^2 + x[2]^2)^3))
  return (((a)^0.01- 1)/0.01)
}

set.seed(33)
mcmc_cauchy <- metrop(log_cauchy, initial = rep(0,2), scale =
   5, nbatch = 100000)
mcmc_cauchy_sample <- mcmc_cauchy$batch
mcmc_cauchy$accept
#Figure 4. b)
plot(mcmc_cauchy_sample, xlab = "X", ylab = "Y", las = 1)

mcmc_cauchy_sample_x_lim <- mcmc_cauchy_sample[,1][abs(mcmc_
   cauchy_sample[,1]) <= 4 & abs(mcmc_cauchy_sample[,2]) <= 4]
mcmc_cauchy_sample_y_lim<- mcmc_cauchy_sample[,2][abs(mcmc_
   cauchy_sample[,1]) <= 4 & abs(mcmc_cauchy_sample[,2]) <= 4]
mcmc_cauchy_sample_lims <- matrix(c(mcmc_cauchy_sample_x_lim,
   mcmc_cauchy_sample_y_lim), ncol = 2)
```

```
##LGC for cauchy##
lg.cauchy <- lg_main(mcmc_cauchy_sample_lims, est_method = "1
    par")
x <- seq(-4, 4, length.out = 15)
y <- seq(-4, 4, length.out = 15)
d1 <- expand.grid(x,y)
zd1 <- dlg(lg.cauchy, grid = d1)
#Figure 4. d)
corplot(zd1, low_color = "#4A6FE3", high_color = "#D33F6A",
    label_size = 4, las = 1)

##matrix function##
#remember all of the values are for 1000 squared
#double derivative values
x <- seq(-4,4,length.out =1000)
y <- seq(-4,4,length.out =1000)
z_dxdy <- matrix(data = c(rep(0,1000000)), nrow =1000, ncol =
    1000)
z_dx2 <- matrix(data = c(rep(0,1000000)), nrow = 1000, ncol =
    1000)
z_dy2 <- matrix(data = c(rep(0,1000000)), nrow = 1000, ncol =
    1000)

for(i in c(1:1000)){
  for(j in c(1:1000)){
    z_dxdy[i,j] <- cauchy.obj$he(c(x[i], y[j]))[2,1]
    z_dx2[i,j] <- cauchy.obj$he(c(x[i], y[j]))[1,1]
    z_dy2[i,j] <- cauchy.obj$he(c(x[i], y[j]))[2,2]
  }
}

cauchy_doub_derivs <- matrix_function(z_dx2 = z_dx2, z_dxdy = z
    _dxdy, z_dy2 = z_dy2)
z_rho <- cauchy_doub_derivs$z_rho
z_sig_x <- cauchy_doub_derivs$z_sig_x
z_sig_y <- cauchy_doub_derivs$z_sig_y

#plot
```

```
#Figure 4. c)
image(x,y,z_rho, col = hcl.colors(10, "Blue-Red␣2"), las = 1)
contour(x,y,z_rho, add = TRUE, labcex = 1)

z_sig_x_trunk <- z_sig_x[z_sig_x <10 & z_sig_x > -10]
z_sig_y_trunk <- z_sig_y[z_sig_y <10 & z_sig_y > -10]
#Figure 8. a) and b)
hist(z_sig_x[z_sig_x <1 & z_sig_x > -1], main = "", xlab = "",
   freq = FALSE, col = "grey", las = 1)
hist(z_sig_y[z_sig_y <1 & z_sig_y > -1], main = "", xlab = "",
   freq = FALSE, col = "grey", las = 1)
length(z_sig_x)- length(z_sig_x_trunk)
length(z_sig_y)- length(z_sig_y_trunk)

###Box-Cox transformed Cauchy###
param_alt <- list(X = 0, Y = 0, Lam = 0)

dat <- read.table("Master/data_cauchy_alt.txt", header = TRUE)
compile("Master/bivariate_cauchy_alt.cpp")
dyn.load(dynlib("Master/bivariate_cauchy_alt"))

cauchy.obj_alt <- MakeADFun(data = dat, parameters = param_alt,
    DLL = "bivariate_cauchy_alt")
x <- seq(-4,4, length.out = 100)
y <- seq(-4,4, length.out = 100)
Lam <- 0.1 #Changing Lam to the values of 0.001, 0.1, 1, 5 and
   10 will allow one to find figures 13. - 15.
z <- matrix(data = c(rep(0,10000)), nrow = 100,ncol = 100)

for (i in c(1:100)){
  for (j in c(1:100)){
    z[i,j] <- cauchy.obj_alt$fn(c(x[i], y[j], Lam))
  }
}
#Density for figure 14.
image(x,y, z, las = 1)
contour(x,y,z, add = TRUE, labcex = 1)
```

```
#z_rho
x <- seq(-4,4,length.out =1000)
y <- seq(-4,4,length.out =1000)
z_dxdy <- matrix(data = c(rep(0,1000000)), nrow =1000, ncol =
    1000)
z_dx2 <- matrix(data = c(rep(0,1000000)), nrow = 1000, ncol =
    1000)
z_dy2 <- matrix(data = c(rep(0,1000000)), nrow = 1000, ncol =
    1000)

for(i in c(1:1000)){
  for(j in c(1:1000)){
    z_dxdy[i,j] <- cauchy.obj_alt$he(c(x[i], y[j], Lam))[2,1]
    z_dx2[i,j] <- cauchy.obj_alt$he(c(x[i], y[j], Lam))[1,1]
    z_dy2[i,j] <- cauchy.obj_alt$he(c(x[i], y[j], Lam))[2,2]
  }
}

cauchy_doub_derivs <- matrix_function(z_dx2 = z_dx2, z_dxdy = z
    _dxdy, z_dy2 = z_dy2)
z_rho <- cauchy_doub_derivs$z_rho

#Figure 13.
hist(z_rho, main = "", xlab = "", col = "grey", freq = FALSE,
    las = 1)

#Figure 15.
image(x,y,z_rho, breaks = c(-1500, seq(-1,1, length.out = 21),
    1500),col = hcl.colors(22, "Blue-Red␣2"), las = 1)
contour(x,y,z_rho, add = TRUE, nlevels = 500 ,labcex = 1)

###Gaussian kernel estimates###
#Changing h to the values of 0.001, 0.093, 0.5, 1,5,100 allows
    one to find Figures 17. c), 19. c), 20. and 21.
data_list <- list(X_i1 = mcmc_cauchy_sample_lims[,1], X_i2 =
    mcmc_cauchy_sample_lims[,2], h = 100)
params <- list(X = 1, Y = 1)
cauchy_kern_func <- MakeADFun(data = data_list, parameters =
```

```
    params, DLL = "Gauss_kern")

##density est##
bw.nrd(mcmc_cauchy_sample_lims)
bw.nrd0(mcmc_cauchy_sample_lims)
x <- seq(-4,4, length.out = 100)
y <- seq(-4,4, length.out = 100)
z <- matrix(rep(0,100^2), nrow = 100, ncol = 100)
z_dx2 <- matrix(rep(0,100^2), nrow = 100, ncol = 100)
z_dy2 <- matrix(rep(0,100^2), nrow = 100, ncol = 100)
z_dxdy <- matrix(rep(0,100^2), nrow = 100, ncol = 100)

for(i in c(1:100)){
  for(j in c(1:100)){
    z[i,j] <- cauchy_kern_func$fn(c(x[i], y[j]))
    cauchy_kern_he <- cauchy_kern_func$he(c(x[i], y[j]))
    z_dx2[i,j] <- cauchy_kern_he[1,1]
    z_dy2[i,j] <- cauchy_kern_he[2,2]
    z_dxdy[i,j] <- cauchy_kern_he[1,2]
  }
}

#Figure 17. c)
image(x,y,exp(z), las = 1)
contour(x,y,exp(z), labcex = 1 ,add = TRUE)

##Double derivative
cauchy_kern_deriv <- matrix_function(z_dx2 = z_dx2,z_dxdy = z_
    dxdy, z_dy2 = z_dy2, row_length = 0.093)

#histograms for Figure 20.
hist(cauchy_kern_deriv$z_rho, main = "", xlab = "", col = "grey
    ", freq = FALSE, las = 1)

#rho estimates for Figure 19.c) and 21.
image(x,y,cauchy_kern_deriv$z_rho, breaks = c(-40,seq(-1,1,
    length.out = 21), 40), col = hcl.colors(22, "BLUE␣RED␣2"),
    las = 1)
```

```
contour(x,y,cauchy_kern_deriv$z_rho,add = TRUE,nlevels = 200 ,
    labcex = 1)

###Bivar example 3.####
compile("Master/bivar_ex3.cpp")
dyn.load(dynlib("Master/bivar_ex3"))

Parameters3 <- list(x = 0, y = 0, mu_1 = 4, mu_2 = 2, sig_1 =
    5, sig_2 = 2, rho = -0.27)
obj_bivar3 <- MakeADFun(data = dat, parameters = Parameters3,
    DLL = "bivar_ex3")

x <- seq(-8,15,length.out =100)
y <- seq(-3,3,length.out =100)
z <- matrix(data = c(rep(0,10000)), nrow = 100,ncol = 100)
for (i in c(1:100)){
  for (j in c(1:100)){
    z[i,j] <- exp(obj_bivar3$fn(c(x[i], y[j], 4, 2, 5, 2,
        -0.27)))
  }
}
#Figure 5. a)
image(x,y,z, las = 1)
contour(x,y,z, add = TRUE, labcex = 1)

##MCMC for bivar example 3.##
bivar_ex3 <- function(x){
  U = c(x[1] - 1)
  V = c(x[2]^2 + 2)
  a = 1/(2 * pi * 5 * 2 * sqrt(1 - (-0.27)^2))
  b = - 1/(2*(1 - (-0.27)^2)) * ( ((U-4)/5)^2
                                  - 2 * (-0.27) * ((U-4)/5)* ((V
                                     -2)/2) + ((V-2)/2)^2)
  return (a * exp(b) * 2 *abs(abs(x[2])))
}

log_bivar_ex3 <- function(x){
  return(log(bivar_ex3(x)))
```

```
}

set.seed(365)
mcmc_bivar_ex3 <- metrop(log_bivar_ex3,initial = c(1,1), blen =
    1, nspac = 1,scale = 4, nbatch = 100000)
mcmc_bivar_ex3_sample = mcmc_bivar_ex3$batch
#Figure 5. b)
plot(mcmc_bivar_ex3_sample, xlab = "x", ylab = "y", las = 1)
mcmc_bivar_ex3$accept

a <- mcmc_bivar_ex3_sample[,1][mcmc_bivar_ex3_sample[,1] <= 15
   & mcmc_bivar_ex3_sample[,1] >= -8]
b <- mcmc_bivar_ex3_sample[,2][mcmc_bivar_ex3_sample[,1] <= 15
   & mcmc_bivar_ex3_sample[,1] >= -8]
a_b <- matrix(c(a,b), ncol = 2)

##LGC for bivar example 3.##
lg.bivar_ex3 <- lg_main(a_b, est_method = "1par")
x <- seq(-8, 15, length.out = 15)
y <- seq(-3, 3, length.out = 15)
d1 <- expand.grid(x,y)
zd1 <- dlg(lg.bivar_ex3, grid = d1)
#Figure 5. d)
corplot(zd1, plot_thres = 0.02, low_color = "#4A6FE3", high_
   color = "#D33F6A", label_size = 4)

##matrix function##
#remember all of the values are for 1000 squared
#double derivative values
x <- seq(-8,15,length.out =1000)
y <- seq(-3,3,length.out =1000)
z_dxdy <- matrix(data = c(rep(0,1000000)), nrow =1000, ncol =
   1000)
z_dx2 <- matrix(data = c(rep(0,1000000)), nrow = 1000, ncol =
   1000)
z_dy2 <- matrix(data = c(rep(0,1000000)), nrow = 1000, ncol =
   1000)
```

```
for(i in c(1:1000)){
  for(j in c(1:1000)){
    z_dxdy[i,j] <- obj_bivar3$he(c(x[i], y[j],4, 2, 5, 2,
       -0.27))[2,1]
    z_dx2[i,j] <- obj_bivar3$he(c(x[i], y[j], 4, 2, 5, 2,
       -0.27))[1,1]
    z_dy2[i,j] <- obj_bivar3$he(c(x[i], y[j], 4, 2, 5, 2,
       -0.27))[2,2]
  }
}

bivar_ex3_doub_derivs <- matrix_function(z_dx2 = z_dx2, z_dxdy
   = z_dxdy, z_dy2 = z_dy2)
z_rho <- bivar_ex3_doub_derivs$z_rho
z_sig_x <-bivar_ex3_doub_derivs$z_sig_x
z_sig_y <- bivar_ex3_doub_derivs$z_sig_y

#plot
#Figure 5. c)
image(x,y,z_rho, col = hcl.colors(21, "Blue␣Red␣2")[c(8:10,
   12:14)],
      breaks = c(-0.3,-0.2, -0.1, 0, 0.1, 0.2, 0.3), las = 1)
contour(x,y,z_rho, add = TRUE, labcex = 1)

#Figure 9. a) and b)
hist(z_sig_x, main = "", freq = FALSE, xlab = "", col = "grey",
     las = 1)
hist(z_sig_y, main = "", freq = FALSE, xlab = "", col = "grey",
     las = 1)

###Gaussian kernel estimates###
#For the h values 0.28, 0.5, 1 to get figures 17. d), 18. and
   19. d)
data_list <- list(X_i1 = a, X_i2 = b, h = 0.28)
params <- list(X = 1, Y = 1)
bivar_ex3_kern_func <- MakeADFun(data = data_list, parameters =
     params, DLL = "Gauss_kern")
```

```
##density est##
bw.nrd(a_b)
bw.nrd0(a_b)
x <- seq(-8,15, length.out = 100)
y <- seq(-3,3, length.out = 100)
z <- matrix(rep(0,100^2), nrow = 100, ncol = 100)
z_dx2 <- matrix(rep(0,100^2), nrow = 100, ncol = 100)
z_dy2 <- matrix(rep(0,100^2), nrow = 100, ncol = 100)
z_dxdy <- matrix(rep(0,100^2), nrow = 100, ncol = 100)

for(i in c(1:100)){
  for(j in c(1:100)){
    z[i,j] <- bivar_ex3_kern_func$fn(c(x[i], y[j]))
    bivar_ex3_kern_he <- bivar_ex3_kern_func$he(c(x[i], y[j]))
    z_dx2[i,j] <- bivar_ex3_kern_he[1,1]
    z_dy2[i,j] <- bivar_ex3_kern_he[2,2]
    z_dxdy[i,j] <- bivar_ex3_kern_he[1,2]
  }
}

#For figure 17. d) and 18.
image(x,y,exp(z), las = 1)
contour(x,y,exp(z), labcex = 1 ,add = TRUE)

##Double derivative
bivar_ex3_kern_deriv <- matrix_function(z_dx2 = z_dx2,z_dxdy =
   z_dxdy, z_dy2 = z_dy2, row_length = 100)

hist(bivar_ex3_kern_deriv$z_rho)
#For figure 19. d)
image(x,y,bivar_ex3_kern_deriv$z_rho, col = hcl.colors(22, "
   BLUE␣RED␣2"), breaks = c(-15,seq(-1,1,length.out = 21),15),
    las = 1)
contour(x,y,bivar_ex3_kern_deriv$z_rho, add = TRUE, labcex = 1,
    nlevels = 200)

###Functions derivatives and rho functions
   ##################################################
```

```
##cauchy:

d2_dx2_log_cauchy <- function(x){
  return(3 * (x[1]^2 - x[2]^2 - 1) / ((x[1]^2 + x[2]^2 + 1)^2))
}

d2_dy2_log_cauchy <- function(x){
  return(-3 * (x[1]^2 - x[2]^2 + 1) / ((x[1]^2 + x[2]^2 + 1)^2)
      )
}
d2_dxdy_log_cauchy <- function(x){
  return(6 * (x[1]*x[2]) / ((x[1]^2 + x[2]^2 + 1)^2))
}

x <- seq(-4,4, length.out = 1000)
y <- seq(-4,4, length.out = 1000)
z_val <- matrix(data = c(rep(0, 1000^2)), nrow = 1000)

for(i in c(1:1000)){
  for(j in c(1:1000)){
    z_val[i,j] <- d2_dxdy_log_cauchy(c(x[i], y[j]))
  }
}

contour(x,y,z_val)

inverse_mat_cauchy <- function(x){
  a <- -d2_dx2_log_cauchy(x)
  b <- -d2_dxdy_log_cauchy(x)
  c <- -d2_dy2_log_cauchy(x)
  d <- matrix(c(a,b,b,c), nrow = 2, ncol = 2)
  return(solve(d))
}

x <- seq(-2,2, length.out = 1000)
y <- seq(-2,2, length.out = 1000)

z_sigx <- matrix(c(0,1000^2), nrow = 1000, ncol = 1000)
```

```
z_sigy <- matrix(c(0,1000^2), nrow = 1000, ncol = 1000)
z_rho_sigxy <- matrix(c(0,1000^2), nrow = 1000, ncol = 1000)
z_rho_sigxy_alt <- matrix(c(0,1000^2), nrow = 1000, ncol =
    1000)
z_rho <- matrix(c(0,1000^2), nrow = 1000, ncol = 1000)

for (i in c(1:1000)){
  for(j in c(1:1000)){
    z_sigx[i,j] <- inverse_mat_cauchy(c(x[i], y[j]))[1,1]
    z_sigy[i,j] <- inverse_mat_cauchy(c(x[i], y[j]))[2,2]
    z_rho_sigxy[i,j] <- inverse_mat_cauchy(c(x[i], y[j]))[1,2]
    z_rho_sigxy_alt[i,j] <- inverse_mat_cauchy(c(x[i], y[j]))
        [2,1]
    z_rho[i,j] <- z_rho_sigxy[i,j] / (sqrt(z_sigx[i,j])*sqrt(z_
        sigy[i,j]))
  }
}

contour(x,y,z_sigx)
image(x,y, z_sigx, zlim = c(0,7000))
contour(x,y,z_sigy)
image(x,y, z_sigy, zlim = c(0,7000))
contour(x,y, z_rho_sigxy)
contour(x,y,z_rho)

sqrd_comp_1 <- function(x){
  return(sqrt((x[2]^4 - x[1]^4 - 2*x[1]^2-1)/(3*(x[1]^2 + x
      [2]^2 - 1))))
}
sqrd_comp_1_vals <- matrix(c(rep(0,1000^2)), nrow = 1000, ncol
    =1000)

for (i in c(1:1000)){
  for(j in c(1:1000)){
    sqrd_comp_1_vals[i,j] <- sqrd_comp_1(c(x[i], y[j]))
  }
}
```

```
image(x,y,sqrd_comp_1_vals, breaks = c(seq(0,1,length.out = 11)
    ,10,100))


##Pear
Parameters <- list(x = 0, y = 0, mu_1 = 10, mu_2 = 1.55, sig_1
    = 10, sig_2 = 0.775, rho = 0.75)

compile("Master/pear_bivar.cpp")
dyn.load(dynlib("Master/pear_bivar"))
obj <- MakeADFun(data = dat, parameters = Parameters, DLL = "
    pear_bivar")

#first order derivatives
x <- seq(-2,2, length.out = 1000)
y <- seq(-2,2, length.out = 1000)

z_dx <- matrix(data = c(rep(0,1000000)), nrow =1000, ncol =
    1000)
z_dy <- matrix(data = c(rep(0,1000000)), nrow =1000, ncol =
    1000)

pear_dx <- function(x){
  return((2/x[1]) - ((6 * x[1]^5 - 60 * x[1]^2)/100 - 4.5 * x
      [1]^2 * (x[2] - 1.55)/7.75)/ (2 * (1 - 0.75^2)))
}

pear_dy <- function(x){
  return(0.221198 * x[1]^3 - 3.80556 * x[2] + 3.68664)
}

pear_dx_vals <- matrix(data = c(rep(0,1000000)), nrow =1000,
    ncol = 1000)

for(i in c(1:1000)){
  for (j in c(1:1000)){
    z_dx[i,j] <- obj$gr(c(x[i], y[j], 10, 1.55, 10, 0.775,
        0.75))[1]
```

```
    z_dy[i,j] <- obj$gr(c(x[i], y[j], 10, 1.55, 10, 0.775,
       0.75))[2]
    pear_dx_vals[i,j] <- pear_dx(c(x[i], y[j]))
  }
}

#Second order
#few decimals off it seems
pear_rho_func <- function(x){
  a <- (1 / (2 * (1 - 0.75^2))) * ((3 * x[1]^4 - 12 * x[1])/10
     - 9 * x[1] * (x[2] - 1.55) / 7.75) + 2/(x[1]^2)
  b <- (-1 / (2 * (1 - 0.75^2))) * (4.5 * x[1]^2 / 7.75)
  d <- (1 / (1 - 0.75^2)) * (1 / 0.775^2)
  sig_x <- d /(a * d - b^2)
  sig_y <- a /(a * d - b^2)
  rho_sigx_sigy <- - b /(a * d - b^2)
  return( rho_sigx_sigy/(sqrt(sig_x) * sqrt(sig_y)))
}

x <- seq(-3,4,length.out =1000)
y <- seq(0,4,length.out =1000)
z_dxdy <- matrix(data = c(rep(0,1000000)), nrow =1000, ncol =
   1000)
z_dx2 <- matrix(data = c(rep(0,1000000)), nrow = 1000, ncol =
   1000)
z_dy2 <- matrix(data = c(rep(0,1000000)), nrow = 1000, ncol =
   1000)
pear_rho_vals <- matrix(data = c(rep(0,1000000)), nrow =1000,
   ncol = 1000)

for(i in c(1:1000)){
  for(j in c(1:1000)){
    z_dxdy[i,j] <- obj$he(c(x[i], y[j], 10, 1.55, 10, 0.775,
       0.75))[2,1]
    z_dx2[i,j] <- obj$he(c(x[i], y[j], 10, 1.55, 10, 0.775,
       0.75))[1,1]
    z_dy2[i,j] <- obj$he(c(x[i], y[j], 10, 1.55, 10, 0.775,
       0.75))[2,2]
```

```
    pear_rho_vals[i,j] <- pear_rho_func(c(x[i], y[j]))
  }
}

z_rho <- matrix(matrix_function(z_dx2 = z_dx2, z_dxdy = z_dxdy,
     z_dy2 = z_dy2), nrow = 1000, ncol = 1000)
image(x,y,pear_rho_vals)
image(x,y,z_rho)

### bivar ex3
compile("Master/bivar_ex3.cpp")
dyn.load(dynlib("Master/bivar_ex3"))

Parameters3 <- list(x = 0, y = 0, mu_1 = 4, mu_2 = 2, sig_1 =
   5, sig_2 = 2, rho = -0.27)
obj_bivar3 <- MakeADFun(data = dat, parameters = Parameters3,
   DLL = "bivar_ex3")

bivar_rho_func <- function(x){
  a <- 1 /((1 - 0.27^2) * 25)
  b <- (2 * 0.27 * x[2]) / ((1 - 0.27^2) * 10)
  d <- (1 / (2 * (1 - 0.27^2))) * ((12 * x[2]^2)/4 - 4 *
     (-0.27) * (x[1]- 6)/10) + 1 / (x[2]^2)
  sig_x <- d /(a * d - b^2)
  sig_y <- a /(a * d - b^2)
  rho_sigx_sigy <- - b /(a * d - b^2)
  return( rho_sigx_sigy/(sqrt(sig_x) * sqrt(sig_y)))
}

x <- seq(-8,15,length.out =1000)
y <- seq(-3,3,length.out =1000)
z_dxdy <- matrix(data = c(rep(0,1000000)), nrow =1000, ncol =
   1000)
z_dx2 <- matrix(data = c(rep(0,1000000)), nrow = 1000, ncol =
   1000)
z_dy2 <- matrix(data = c(rep(0,1000000)), nrow = 1000, ncol =
   1000)
bivar_rho_vals <- matrix(data = c(rep(0,1000000)), nrow =1000,
```

```
   ncol = 1000)

for(i in c(1:1000)){
  for(j in c(1:1000)){
    z_dxdy[i,j] <- obj_bivar3$he(c(x[i], y[j],4, 2, 5, 2,
       -0.27))[2,1]
    z_dx2[i,j] <- obj_bivar3$he(c(x[i], y[j], 4, 2, 5, 2,
       -0.27))[1,1]
    z_dy2[i,j] <- obj_bivar3$he(c(x[i], y[j], 4, 2, 5, 2,
       -0.27))[2,2]
    bivar_rho_vals[i,j] <- bivar_rho_func(c(x[i], y[j]))
  }
}

z_rho <- matrix(matrix_function(z_dx2 = z_dx2, z_dxdy = z_dxdy,
    z_dy2 = z_dy2), nrow = 1000, ncol = 1000)
image(x,y,bivar_rho_vals)
image(x,y,z_rho)

### Pear 2
param_pear_2 <- list(x = 0, y = 0, mu_1 = 1.2, sig_1 = 1/3)
require(TMB)
compile("Master/pear_bivar2.cpp")
dyn.load(dynlib("Master/pear_bivar2"))
obj <- MakeADFun(data = dat, parameters = param_pear_2, DLL = "
   pear_bivar2")

#er rett
pear2_dx2_func <- function(x){
  a_doub_deriv <- 0.25 /((0.5 * x[1] + 1)^2)
  b_doub_deriv <- -9
  c_doub_deriv <- (exp(-x[1])) * ((0.9* x[1]^3 + 7.2* x[1]^2 +
     10.8* x[1] - 43.2)*x[2]*exp(5 + 0.5 * x[1])
                                + exp(10)*(-0.18*x[1]^4 - 0.72 *
                                    x[1]^3 + 0.72* x[1]^2 + 4.32
                                    * x[1] - 1.44) - 108* exp(x
                                   [1])*x[2]^2)/((x[1] + 2)^4)
  return(a_doub_deriv + b_doub_deriv + c_doub_deriv)
```

```
}

#er rett
pear2_dxdy_func <- function(x){
  return(((-1.8 * x[1]^2 - 7.2 * x[1] + 7.2) * exp(5 - 0.5 * x
      [1]) + 72 * x[2])/((x[1] + 2)^3))
}

#er rett
pear2_dy2_func <- function(x){
  return(-9/ (0.5 * x[1] + 1)^2)
}

pear2_rho_func<- function(x){
  a = - pear2_dx2_func(c(x[1],x[2]))
  b = - pear2_dxdy_func(c(x[1],x[2]))
  d = - pear2_dy2_func(c(x[1],x[2]))
  sig_x <- d /(a * d - b^2)
  sig_y <- a /(a * d - b^2)
  rho_sigx_sigy <- - b /(a * d - b^2)
  return( rho_sigx_sigy/(sqrt(sig_x) * sqrt(sig_y)))
}

#er rett
pear2_dy_func <- function(x){
  return(-9 * (x[2] - 0.1 * x[1] * exp(5 - 0.5 * x[1]))/((0.5 *
      x[1] + 1)^2))
}

#er rett
pear2_dx_func <- function(x){
  a_val <- 5
  a_deriv <- -0.5 / (0.5 * x[1] + 1)
  b_deriv <- 10.8 - 9 * x[1]
  c_deriv <-((- 1.8 * x[1]^2 - 7.2 * x[1] + 7.2) * x[2] * exp(a
     _val - 0.5 * x[1]) +
              exp(2 * a_val - x[1]) * x[1] * (0.18 * x[1]^2 +
                 0.36 * x[1] - 0.72) + 36 * x[2]^2)/((x[1] + 2)
```

```
                ^3)
  return(a_deriv + b_deriv + c_deriv)
}

x <- seq(0,2,length.out =1000)
y <- seq(0,14,length.out =1000)
z_dxdy <- matrix(data = c(rep(0,1000000)), nrow =1000, ncol =
    1000)
z_dx2 <- matrix(data = c(rep(0,1000000)), nrow = 1000, ncol =
    1000)
z_dy2 <- matrix(data = c(rep(0,1000000)), nrow = 1000, ncol =
    1000)
pear2_rho_vals <- matrix(data = c(rep(0,1000000)), nrow =1000,
    ncol = 1000)

for(i in c(1:1000)){
  for(j in c(1:1000)){
    z_dxdy[i,j] <- obj$he(c(x[i], y[j] , 1.2, 1/3))[2,1]
    z_dx2[i,j] <- obj$he(c(x[i], y[j] , 1.2, 1/3))[1,1]
    z_dy2[i,j] <- obj$he(c(x[i], y[j] , 1.2, 1/3))[2,2]
    pear2_rho_vals[i,j] <- pear2_rho_func(c(x[i], y[j]))
  }
}

z_rho <- matrix(matrix_function(z_dx2 = z_dx2, z_dxdy = z_dxdy,
     z_dy2 = z_dy2), nrow = 1000, ncol = 1000)

image(x,y, z_rho)
image(x,y,pear2_rho_vals)
####Kernel Derivatives####
data_list <- list(X_i1 = mcmc_pear_sample[,1], X_i2 = mcmc_pear
    _sample[,2], h = 1)
params <- list(X = 1, Y = 1)
pear_kern_func <- MakeADFun(data = data_list, parameters =
    params, DLL = "Gauss_kern")
data_list_cauchy <- list(X_i1 = mcmc_cauchy_sample_lims[,1], X_
    i2 = mcmc_cauchy_sample_lims[,2], h = 0.093)
cauchy_kern_func <- MakeADFun(data = data_list_cauchy,
```

```
    parameters = params, DLL = "Gauss_kern")

##First order derivative##
z <- matrix(rep(0,100^2), nrow = 100, ncol = 100)
x <- seq(0,1, length.out = 100)
y <- seq(0,1, length.out = 100)
for(i in c(1:100)){
  for(j in c(1:100)){
    z[i,j] <- pear_kern_func$gr(c(x[i], y[j]))[2]
  }
}
contour(z)

kernel_x_deriv <- function(h,x,y, x_i, y_i){
  b <- sum(((x_i - x)/h) * exp(-0.5 * ((x_i - x)^2 + (y_i - y)
     ^2)/h))
  d <- sum(exp(-0.5 * ((x_i - x)^2 + (y_i - y)^2)/h))
  return(b/d)
}

kernel_y_deriv <- function(h,x,y, x_i, y_i){
  b <- sum(((y_i - y)/h) * exp(-0.5 * ((x_i - x)^2 + (y_i - y)
     ^2)/h))
  d <- sum(exp(-0.5 * ((x_i - x)^2 + (y_i - y)^2)/h))
  return(b/d)
}

z_alt <- matrix(rep(0,100^2), nrow = 100, ncol = 100)
x <- seq(0,1, length.out = 100)
y <- seq(0,1, length.out = 100)
for(i in c(1:100)){
  for(j in c(1:100)){
    z_alt[i,j] <- kernel_y_deriv(h = 1, x = x[i], y = y[j], x_i
        = mcmc_pear_sample[,1], y_i = mcmc_pear_sample[,2])
  }
}
contour(z_alt)
```

```
##Double derivative##

z <- matrix(rep(0,100^2), nrow = 100, ncol = 100)
x <- seq(0,1, length.out = 100)
y <- seq(0,1, length.out = 100)
for(i in c(1:100)){
  for(j in c(1:100)){
    z[i,j] <- pear_kern_func$he(c(x[i], y[j]))[1,1]
  }
}
contour(x,y,z)

kern_2x_deriv <- function(h, x, y, x_i, y_i){
  a <- sum(exp(-0.5 * ((x_i - x)^2 + (y_i - y)^2)/h))
  b <- sum((x_i - x)^2 *exp(-0.5 * ((x_i - x)^2 + (y_i - y)^2)/
     h))
  c <- sum((x_i - x) *exp(-0.5 * ((x_i - x)^2 + (y_i - y)^2)/h)
     )
  return(((a)*(-h * a + b) - (c^2))/((h^2) * (a^2)))
}

kern_2y_deriv <- function(h, x, y, x_i, y_i){
  a <- sum(exp(-0.5 * ((x_i - x)^2 + (y_i - y)^2)/h))
  b <- sum((y_i - y)^2 *exp(-0.5 * ((x_i - x)^2 + (y_i - y)^2)/
     h))
  c <- sum((y_i - y) *exp(-0.5 * ((x_i - x)^2 + (y_i - y)^2)/h)
     )
  return(((a)*(-h * a + b) - (c^2))/((h^2) * (a^2)))
}

kern_xy_deriv <- function(h, x, y, x_i, y_i){
  a <- sum((x - x_i) * (y - y_i) *exp(-0.5 * ((x_i - x)^2 + (y_
     i - y)^2)/h))
  b <- sum(exp(-0.5 * ((x_i - x)^2 + (y_i - y)^2)/h))
  part_1 <- a/ b * (1/(h^2))

  c <- - sum((x - x_i) * exp(-0.5 * ((x_i - x)^2 + (y_i - y)^2)
     /h))/h
```

```
  d <- - sum((y - y_i) * exp(-0.5 * ((x_i - x)^2 + (y_i - y)^2)
     /h))/h
  part_2 <- c * d /(b^2)
  return(part_1 - part_2)
}

z_alt <- matrix(rep(0,100^2), nrow = 100, ncol = 100)
x <- seq(0,1, length.out = 100)
y <- seq(0,1, length.out = 100)
for(i in c(1:100)){
  for(j in c(1:100)){
    z_alt[i,j] <- kern_2x_deriv(h = 0.119, x = x[i], y = y[j],
        x_i = mcmc_pear_sample[,1], y_i = mcmc_pear_sample[,2])
  }
}
contour(z_alt)

##testing##
data_list_cauchy <- list(X_i1 = mcmc_cauchy_sample_lims[,1], X_
    i2 = mcmc_cauchy_sample_lims[,2], h = 0.093)
cauchy_kern_func <- MakeADFun(data = data_list_cauchy,
    parameters = params, DLL = "Gauss_kern")

z <- matrix(rep(0,100^2), nrow = 100, ncol = 100)
z_sec <- matrix(rep(0,100^2), nrow = 100, ncol = 100)
z_thi <- matrix(rep(0,100^2), nrow = 100, ncol = 100)
z_fou <- matrix(rep(0,100^2), nrow = 100, ncol = 100)
z_fif <- matrix(rep(0,100^2), nrow = 100, ncol = 100)
x <- seq(-4,4, length.out = 100)
y <- seq(-4,4, length.out = 100)
for(i in c(1:100)){
  for(j in c(1:100)){
    derivs <- cauchy_kern_func$he(c(x[i], y[j]))
    z[i,j] <- derivs[1,1]* derivs[2,2]
    z_sec[i,j] <- derivs[1,2]^2
  }
}
#Figure 22. a) and b). Get them by changing h to 0.5 and 0.093
```

```
image(x,y, z -z_sec, las = 1)
contour(x,y,z -z_sec, labcex = 1,add = TRUE)

hist(z - z_sec, main= "", xlab = "", col = "grey")

###Bivar example 1.####
compile("Master/pear_bivar_deriv.cpp")
dyn.load(dynlib("Master/pear_bivar_deriv"))

obj <- MakeADFun(data = dat, parameters = Parameters, DLL = "
   pear_bivar_deriv")
lit_gamma <- obj$he(c(1,1 , 0, 0, 1, 1, 0.5))[2,1]

x <- seq(-3,3,length.out =100)
y <- seq(-3,3,length.out =100)
z <- matrix(data = c(rep(0,10000)), nrow = 100,ncol = 100)
for (i in c(1:100)){
  for (j in c(1:100)){
    z[i,j] <- exp(obj$fn(c(x[i], y[j], 0, 0, 1, 1, 0.5)))
  }
}
image(x,y,z)
contour(x,y,z, add = TRUE)

##local dependence function##
sig_x = sig_y = 1

bivar_gam <- obj$he(c(1, 1, 0, 0, 1, 1, 0.5))[2,1]
real_inverse_gamma_function(bivar_gam * sig_x * sig_y) #returns
    0.5

##MCMC for bivar example 1.##
bivar_ex1 <- function(x){
  a = 1/(2 * pi * 1 * 1* sqrt(1- 0.5^2))
  b = - 1/(2*(1 - 0.5 ^ 2)) * ( x[1]^2
                              - 2 * 0.5 * x[1]* x[2] + x[2]^2)
  return (a * exp(b))
}
```

```
log_bivar_ex1 <- function(x){
  return(log(bivar_ex1(x)))
}

set.seed(365)
#Changing nbatches amount to 100, 1000, 10000, 100000 gives
   figure 23.
mcmc_bivar_ex1 <- metrop(log_bivar_ex1,initial = rep(0,2),
   scale = 2,nbatch = 100000)
mcmc_bivar_ex1_sample = mcmc_bivar_ex1$batch
plot(mcmc_bivar_ex1_sample)
mcmc_bivar_ex1$accept

##LGC for bivar example 1.##
lg.bivar_ex1 <- lg_main(mcmc_bivar_ex1_sample, est_method = "1
   par")
x <- seq(-3, 3, length.out = 20)
y <- seq(-3, 3, length.out = 20)
d1 <- expand.grid(x,y)
zd1 <- dlg(lg.bivar_ex1, grid = d1)
#Figure 23.
corplot(zd1, plot_thres = 0.1, low_color = "#4A6FE3", high_
   color = "#D33F6A")

##matrix function##
#remember all of the values are for 1000 squared
#double derivative values
x <- seq(-3,3,length.out =1000)
y <- seq(-3,3,length.out =1000)
z_dxdy <- matrix(data = c(rep(0,1000000)), nrow =1000, ncol =
   1000)
z_dx2 <- matrix(data = c(rep(0,1000000)), nrow = 1000, ncol =
   1000)
z_dy2 <- matrix(data = c(rep(0,1000000)), nrow = 1000, ncol =
   1000)

for(i in c(1:1000)){
```

```
  for(j in c(1:1000)){
    z_dxdy[i,j] <- obj$he(c(x[i], y[j], 0, 0, 1, 1, 0.5))[2,1]
    z_dx2[i,j] <- obj$he(c(x[i], y[j], 0, 0, 1, 1, 0.5))[1,1]
    z_dy2[i,j] <- obj$he(c(x[i], y[j], 0, 0, 1, 1, 0.5))[2,2]
  }
}

hist(log(-z_dx2[z_dx2 <0]), main = "", xlab = "", freq = FALSE)
hist(log(-z_dy2[z_dy2 <0]), main = "", xlab = "", freq = FALSE)
```