Take home 2

Pieter Luyten

December 16, 2019

Excercise 1

(a)

Fit a linear model, assuming that the strong Gaussian assumption is relevant.

The value for the intercept of the fit is 1.0321764 and for the rico of the fit is 0.1904323

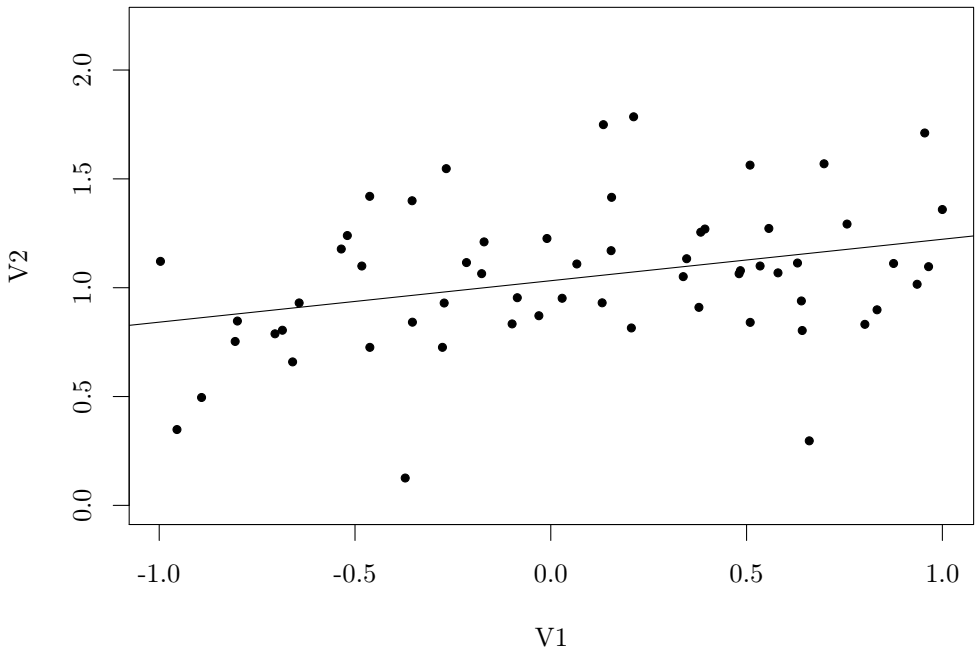

Figure 1: line fit throught the data in Ex1.txt

(b)

Test whether $\beta_1 = 0$ at the level $\alpha = 0.01$.

Using the result from section 7.4.1 in [1] we know that the random variable

$$T = \frac{\beta_1}{\sqrt{\frac{S^2}{\sum_{i=1}^{n}(x_i - \bar{x}^2)}}}$$

has a Student-t distribution with $n-2$ degrees of freedom. The test value is 2.603. Using a student-t distribution with $60-2 = 58$ degrees of freedom we find a p-value of 0.0117. At the confidence level $\alpha = 0.01$, the null hypothesis that $\beta_1 = 0$ holds. The 99% confidence region for the test value is $[-2.663, 2.663]$.

(c)

Explain what a Q-Q plot is and apply this to the residuals.

A q-q plot is a plot where the quantiles of the assumed distribution are plotted against the quantiles from the sample. So in a sample of n points where the observation $x_i, i \in \{1, 2, \ldots, n\}$ are labeled from lowest to highest, the ith point will be plotted at the coordinate $(Q(i/n), x_i)$. Here $Q(x)$ is the quantile function of the assumed distribution, apart from a translation and/or rescaling. This is a function such that $P(X < Q(p)) = p$. If the assumed distribution is a good model for the observed sample, the points will well fitted by a linear function.

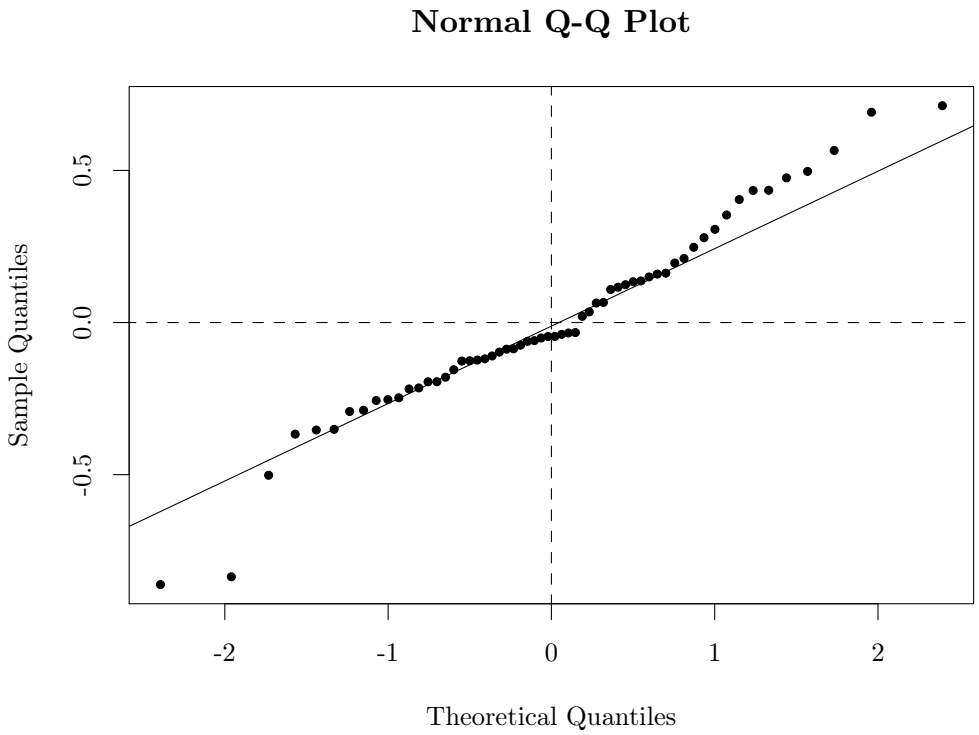

Figure 2: qq plot for the errors on the fit against a standard normal distribution

We see that in this case the line fits the observed quatiles well. In part (d) a Shapiro-Wilk test is done to test if the null-hypothesis of normality of the errors is valid. Notice that the line goes almost exactly through (0,0), from which we can conclude that the distribution will have a mean 0.

(d)

Perform a test that does not require normality of the errors.

To do a test that does not require normality of the errors we first derive an assymptotic result for the distribution of $\hat{\beta}$. We use that $\hat{\beta}$ is given by:

$$\hat{\beta} = (X^T X)^{-1} X^T Y \tag{1}$$

And that we can rewrite Y as $Y = X\beta + \epsilon$ where ϵ is the random vector of errors. Filling this in in equation 1 yields

$$\begin{aligned}\hat{\beta} &= (X^T X)^{-1} X^T (X\beta + \epsilon) \\ &= (X^T X)^{-1}(X^T X)\beta + (X^T X)^{-1} X^T \epsilon \\ &= \beta + (X^T X)^{-1} X^T \epsilon\end{aligned}$$

We see that the distribution of $\hat{\beta}$ is a linear combination of the distributions of the individual errors.

$$\hat{\beta}_i = \beta_i + \sum_{j=1}^{n} [(X^T X)^{-1} X^T]_{ij} \epsilon_j$$

Let $A = (X^T X)^{-1} X^T$. If we assume that the ϵ_i are i.i.d. with $E(\epsilon_i) = 0$ and $Var(\epsilon_i) = \sigma^2$ we see that the distribution of $\hat{\beta}_i$ has expectation value $E(\hat{\beta}_i) = \beta_i$ and $Var(\hat{\beta}_i) = (A_i \cdot A_i^T)\sigma^2$.

We can also use a formula found in [2] proposition 2.3:

Let ϵ_i be i.i.d., $\sigma^2 < \infty$, and let Q be finite and non-singular. Consider the test

$$H_0 : R\beta = r,$$

where R is (1) and r is a a scalar, both known. Then

$$\frac{R\hat{\beta} - r}{\sqrt{s^2 R (X^T X)^{-1} R^T}} \Rightarrow N(0,1)$$

This result was also seen in class. Our interest is in β_1 so we use as vector $R = (0, 1)$. The test value is the same as before, but now we use a standard normal distribution instead of a Student-t distribution to calculate the p-value. This p-value is 0.00865 so the null hypothesis that $\beta_1 = 0$ is not valid with this test. The 99% confidence interval for this test is $[-2.575829, 2.575829]$.

(e)

Determine a 99% confidence region for $\hat{\beta}$

We use the formula from section 7.5.4 in the book to find a confidence region for β. This is a region of the form:

$$\left\{ \beta \middle| \frac{(\hat{\beta} - \beta)^T (X^T X)(\hat{\beta} - \beta)}{pS^2} \leq F_{p,n-p}(1-\alpha) \right\} \tag{2}$$

Because the matrix $X^T X$ is positive definite, this is an ellips with center β and axes along the eigenvectors of $X^T X$. The upper bound in 2 for the 99% confidence region is: $F_{2,58(0.99)=4.9910}$ Filling in the values for $X, p = 2, n = 60$ and $S^2 = 0.0983$ we get the following result (obtained using the CAS system sympy):

$$305.059\beta_0^2 + 47.688\beta_0\beta_1 - 638.832\beta_0 + 96.923\beta_1^2 - 86.137\beta_1 + 337.895 = 4.9910$$

The resulting ellips is plotted in figure 3.

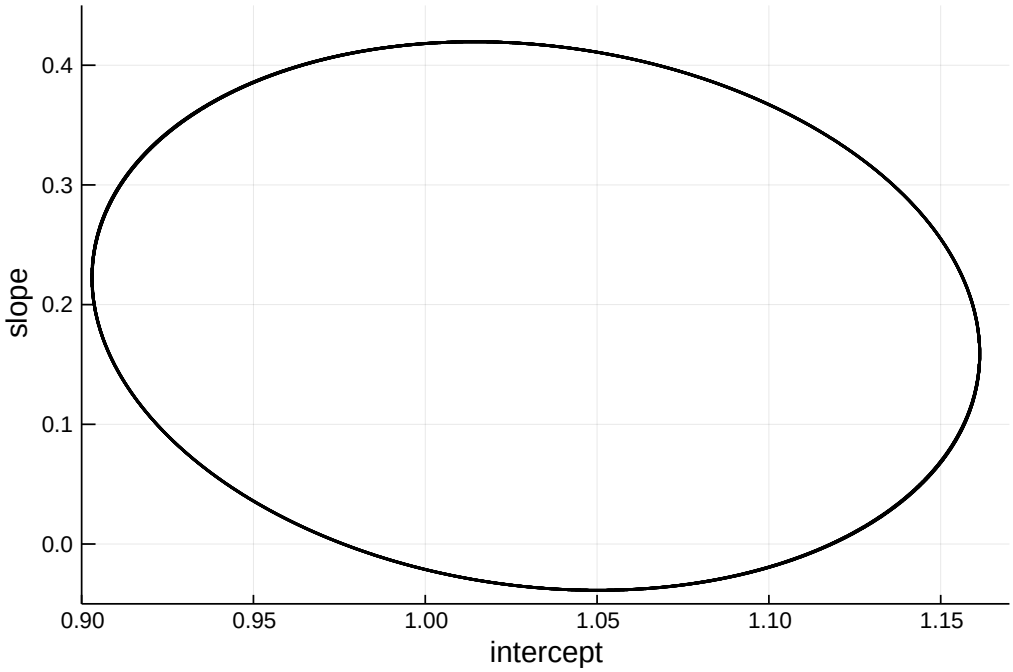

Figure 3: A plot of the 99% confidence region for β.

Question 2

(a)

Compute the ordinary least squares $\hat{\beta}_{OLS}$.

The value for $\hat{\beta}_{OLS}$ obtained using ordinary least squares with the correlated errors is:

$$\hat{\beta}_{OLS} = (3.923161, 1.214355)^T$$

(b)

Suppose we further know that the errors are correlated and satisfy the following equation:

$$\epsilon_i = \rho\epsilon_{i-1} + \eta_i, \quad \text{for} \quad i = 1, \ldots, n \tag{3}$$

Wwhere η_n are i.i.d. standar Gaussian and $\rho = 0.8$. Use 3 to compute the variance of ϵ.

we know that the variance is the same for all ϵ_i and that they obey the recursive relation:

$$\epsilon_i = \rho\epsilon_{i-1} + \eta_i$$

with η_i standard normaly distributed. Using $Var(\epsilon_i) = Var(\epsilon_j)$ for all $i, j \in \{1, 2, \dots 80\}$ we find:

$$\begin{aligned} Var(\epsilon_i) &= \rho^2 Var(\epsilon_{i-1}) + Var(\eta_i) \\ \Leftrightarrow Var(\epsilon_i) &= \rho^2 Var(\epsilon_i) + 1 \\ \Leftrightarrow Var(\epsilon_i) &= \frac{1}{1-\rho^2} \\ &= \frac{25}{9} \\ &= 2.777\dots \end{aligned}$$

We conclude that the variance of ϵ is $\frac{25}{9}$.

(c)

Transform the model in such a way that the errors are no longer correlated. Compute the ordinary least squares for this new model and compare it to the one obtained in (a)

To transform the errors to a basis where they are no longer correlated we will use the variance-covariance matrix V of the errors. The off-diagonal elements, so the covariances, are (suppose $i < j$):

$$\begin{aligned} Cov(\epsilon_i, \epsilon_j) &= E[(\epsilon_i - E(\epsilon_i))(\epsilon_j - E(\epsilon_j))] \\ &= E[\epsilon_i \epsilon_j] \\ &= E[\epsilon_i(\rho\epsilon_{j-1}\eta_j)] \\ &\vdots \\ &= E[\epsilon_i(\rho^{j-i}\epsilon_i + \rho^{j-i-1}\eta_i + \rho^{j-i-2}\eta_{i-2} + \dots + \eta_j)] \\ &= E[\epsilon_i \rho^{i-j} \epsilon_i] \\ &= \rho^{i-j} Var(\epsilon) \\ &= \rho^{i-j} \frac{25}{9} \end{aligned}$$

So the matrix V has $\frac{25}{9}$ on the diagonal, $\frac{25}{9}\rho$ on the two sub-diagonals next to the diagonal, $\frac{25}{9}\rho^k$ on the kth sub-diagonal. the matrix V can be diagonalized with an orthogonal (or unitary) transformation U: $\Lambda = UVU^T$. When we use the definition of the variance-covariance matrix (where ϵ denotes the column vector with the errors) we find:

$$\begin{aligned} V &= E((\epsilon - E(\epsilon))(\epsilon - E(\epsilon))^T) \\ \Leftrightarrow \Lambda &= UE(\epsilon\epsilon^T)U^T \\ \Leftrightarrow \Lambda &= E((U\epsilon)(U\epsilon)^T) \end{aligned}$$

Where we used that $E(\epsilon) = \mathbf{0}$. The vector $U\epsilon$ has variance-covariance matrix Λ, therefore it is a vector of uncorrelated variables. To be able to use the ordinary least squares method however, we need a vector of *i.i.d.* variables which this vector is not, the variances are the eigenvalues of V. To get a transformation we can use to do the fit we need to correct this. Let $\epsilon' = \sqrt{\Lambda^{-1}}U\epsilon$. The variance-covariance matrix of this vector is:

$$\begin{aligned} E(\epsilon'\epsilon'^T) &= \sqrt{\Lambda^{-1}}UE(\epsilon\epsilon^T)U^T\sqrt{\Lambda^{-1}} \\ &= \sqrt{\Lambda^{-1}}\Lambda\sqrt{\Lambda^{-1}} \\ &= Id \end{aligned}$$

where Id is the $n \times n$ identity matrix. The vector ϵ' is a vector of uncorrelated random variables with standard deviation 1. Because they have mean 0 and are linear combinations of Guassian distributions, we conclude that this is a vector of uncorrelated standard Gaussian distributions so

when we transform the model in this way the strong Gaussian assumption is valid and we can use ordinary least squares to fit the model. To transform the errors, we need to transform the matrix X and the vector Y. Let $A = \sqrt{\Lambda^{-1}}U$, $X' = AX$ and $Y' = AY$. Using this in the relation between Y and X:

$$\begin{aligned} Y &= X\beta + \epsilon \\ \Leftrightarrow AY &= AX\beta + \epsilon' \\ \Leftrightarrow Y' &= X'\beta + \epsilon' \end{aligned}$$

We see that the parameter $\hat{\beta}$ from a fit trough the new model is an estimate for the same β as the $\hat{\beta}$ from a fit trough the original model. The fitted parameters in this model are given by:

$$\begin{aligned} \hat{\beta} &= (X'^T X')^{-1} X'^T Y' \\ &= (X^T A^T A X)^{-1} (X^T A^T A Y) \\ &= (X^T W^{-1} X)^{-1} X^T W^{-1} Y \end{aligned}$$

where we used that $A^T A = U^T \Lambda^{-1} U = (U \Lambda U^T)^{-1} = W^{-1}$ which is the same formula for weighted least squares with the eigenvalues of the variance-covariance matrix as weights. As estimate for the parametes this yields:

$$\hat{\beta_0} = 3.630090 \qquad\qquad t\beta_1 = 1.180644$$

with $RSS(\hat{\beta}) = 2.074712$. The residual least square value is a lot smaller than the one obtained with the correlated errorss. The difference in the obtained values is small, but because of the smaller RSS, the one obtained by the model with uncorrelated errors is a lot better. In figure 4 the two 99% confidence regions are plotted. The smaller ellips is the confidence region for the estimate obtained with the model where the errors are uncorrelated. Its clear that this model gives a more accurate estimate of the true value of the parameters. In figure **??** the data is plotted together with the two fits.

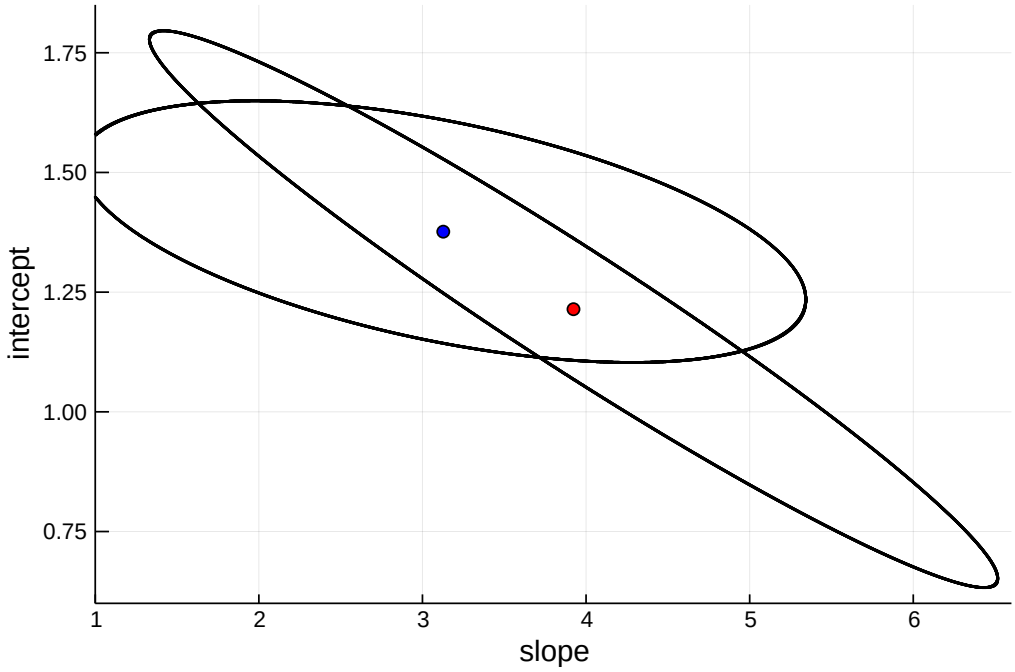

Figure 4: The 99% confidene regions and estimate for $\hat{\beta}_{OLS}$ obtained with the original data (stretched ellips and red point) and with the model where the errors are uncorrelated (less stretched elips and blue point).

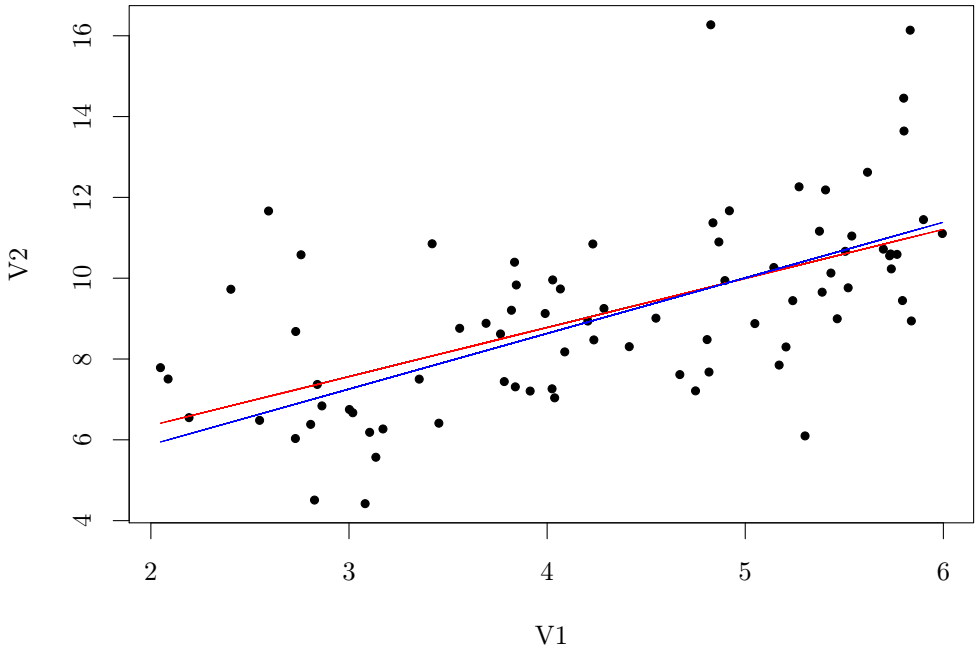

Figure 5: The data together with the two fits, red line is fit trough data with correlated errors, blue the fit trough data with uncorrelated errors.

Question 3

(a)

Compute the weighted least square estimator $\hat{\beta}_{WLS}$

The values for the fitted coefficients with ordinary least squares are:

$[0.4994, 1.1985, -2.4959, 1.2082]$.

In figure 6 the fit is plotted along with the data.

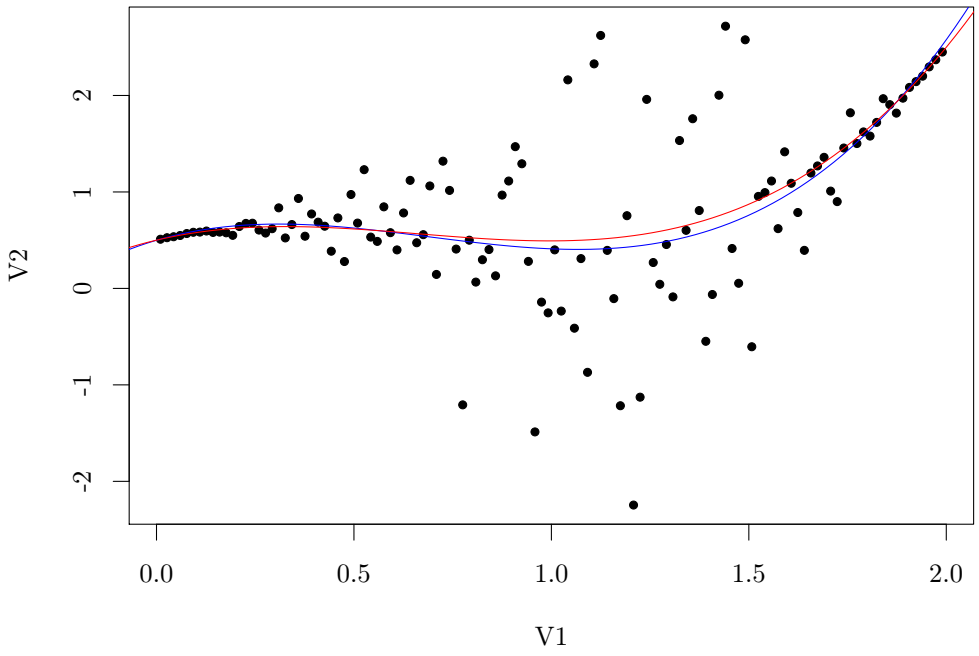

Figure 6: fits of a cubic function using ordinary least square (blue) and weighted least squares (red)

(b)

Suppose we further know that the errors are Gaussian but heteroscedastic as follows:

$$\sigma(x) = \begin{cases} x^2 & \text{if} x \in [0, 4/3] \\ 4(x-2)^2 & \text{if} x \in [4/3, 2] \end{cases}$$

Compute the weighted least square estimator $\hat{\beta}_{WLS}$.

The formula to calculate an estimate for β with the weighted least squares method is given by:

$$\hat{\beta}_{WLS} = (X^T W^{-1} X)^{-1} X^T W^{-1} Y \tag{4}$$

Where W is the variance-covariance matrox of the errors. In this cas this is a diagonal matrix with the variance of ϵ_i at position W_{ii}. The coeffitients fitted with the weighted least square algorithm are:

$\boxed{[0.5003, 0.9669, -1.9655, 0.9910].}$

(c)

Compare $\hat{\beta}_{WLS}$ to $\hat{\beta}_{OLS}$ and with the true parameter i.e. $\beta = (0.5, 1, -2, 1)^T$

The coefficients that are calculated using the weighted least squares method are closer to the real values than the ones calculated using the ordinary least squares. To really say something about the difference and how well they match the given real values of the function we need more information about the distribution of the parameters. In figure 6 the two fits are plotted together with the data used for the fit.

(d)

Determine the distribution of $\hat{\beta}_{WLS}$

To derive the distribution of $\hat{\beta}_{WLS}$ we start from the distribution for the errors, which is known:

$$\epsilon \sim N_n(0, W)$$

Where W is a diagonal matrix with at position W_{ii} the variance of ϵ_i. This is the same matrix W used in equation 4. here Y is the vector with samples used to calculate the least squares estimator. X is the matrix as defined on page 193 in the course notes, this matrix is deterministic. Y can be rewritten as follows:

$$Y = X\beta + \epsilon$$

substituting this into equation **??** yields:

$$\begin{aligned}
\hat{\beta}_{WLS} &= (X^T W^{-1} X)^{-1} X^T W^{-1} (X\beta + \epsilon) \\
&= (X^T W^{-1} X)^{-1} (X^T W^{-1} X)\beta + (X^T W^{-1} X)^{-1} X^T W^{-1} \epsilon \\
&= \beta + X^{-1} W X^{-T} X^T W^{-1} \epsilon \\
&= \beta + X^{-1} \epsilon
\end{aligned}$$

Note that the matrix X^{-1} used here is not well-defined as it is not unique. We take any matrix $A = X^{-1}$ such that $X^{-1}X = Id_p$. We can see that such a mtrix exist by adding extra columns to X with the values $x_i^p, x_i^{p+1}, \ldots, x_i^{n-1}$ with p the amount of parameters in the model and n the mamount of sample points. The determinant of this extended matrix B is the determinant of Vandermonde and is non-zero if all x_i are different. Thus B has an inverse and we can take the first p rows of B as the matrix X^{-1}.

We see this is a linear transformation of a multivariate normal distribution with mean $\mathbf{0}$ and as variance-covariance matrix $\Sigma_\epsilon = W$. Again using the results from section 6.2, page 172 in the course notes we find that the distribution of $\hat{\beta}_{WLS}$ is a multivariate normal distribution with mean β and variance-covariance matrix $X^{-1}WX^{-T} = (X^T W^{-1} X)^{-1}$. We conclude that:

$\boxed{\hat{\beta}_{WLS} \sim N_p(\beta, (X^T W^{-1} X)^{-1})}$

Notice that there is no X^{-1} in the final expression and we only used the fact that $X^{-1}X = Id_p$ so there is no ambiguity in the final expression for the distribution of $\hat{\beta}_{WLS}$.

Question 4

Let $\mathbf{X}$ be a 3-dimensional Gaussian vector with parameters

$$\mu = \begin{pmatrix} 1 \\ 1 \\ -2 \end{pmatrix} \quad \text{and} \quad \Sigma = \begin{pmatrix} 1.25 & 1.50 & 0.5 \\ 1.5 & 5.25 & 3.5 \\ 0.5 & 3.5 & 3.0 \end{pmatrix}$$

(a)

Produce $n = 200$ simulations of $\mathbf{X}$

To generate a sample from a multi-normal distribution we Propositio 6.2 (b) from [1]. We are given the variance-covariance matrix Σ_X of a multivariate normal model. This matrix must be positive definite, so the square root of this matrix exists.

Let A be a matrix such that $AA^T = \Sigma_X$. This matrix can be found by diagonalizing Σ_X: $\Sigma_X = U\Lambda U^T$ and then taking $A = U\sqrt{\Lambda}$. We can do this because all eigenvalues of Σ_X are positive.

Proposition 6.2 b) tells us that $X = AY + \mu$ where Y is a vector of standard normal distributions Because $A\Sigma_Y A^T = \Sigma_X$ where we used that Σ_Y is the $n \times n$ identity matrix. This observation can be used to draw a sample from X by drawing n independent samples from a standard normal distribution, filling them in in a vector y and the calculating the sample from X as $x = Ay + \mu$. This was implemented in R and projections of this sample are plotted in figure 7 together with a 3D-plot.

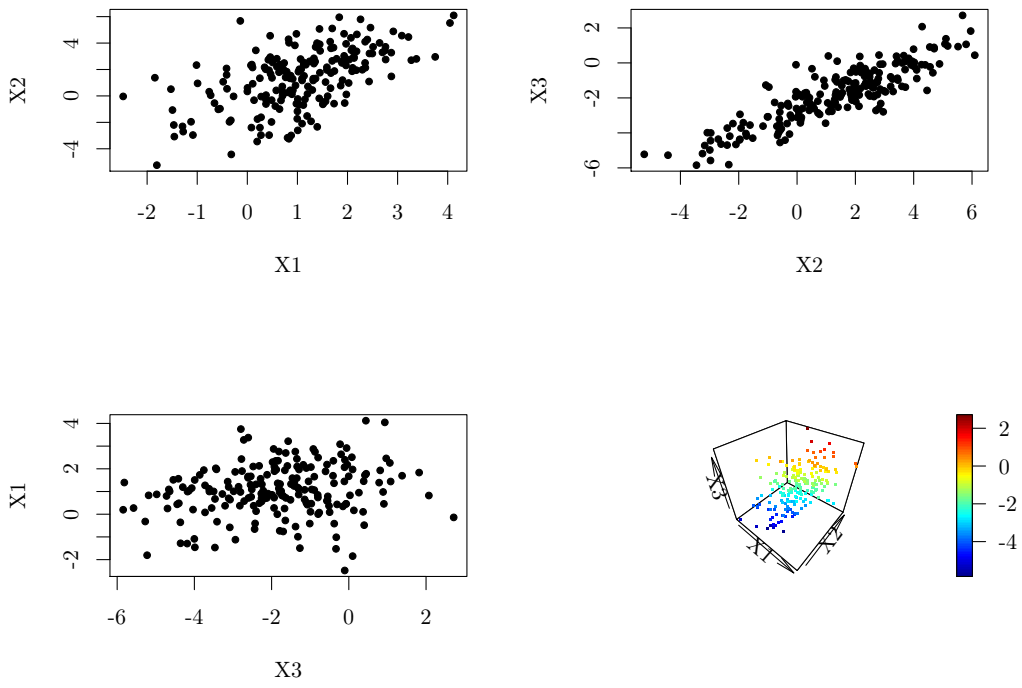

Figure 7: Projection on the xy-plane of the sample from the multivariate normal distribution

(b)

Compute $P(X_1 > 1|X_2 = 1, X_3 = -2)$ and $P(X_1 > 1|X_2 + X_3 = -1)$.

To compute these conditional chances we use proposition 6.2 from the course notes with the following vectors and matrices:

$$\Sigma_{11} = (1.25) \qquad \Sigma_{12} = (1.50, 0.5)$$
$$\Sigma_{21} = \begin{pmatrix} 1.50 \\ 0.50 \end{pmatrix} \qquad \Sigma_{22} = \begin{pmatrix} 5.25 & 3.5 \\ 3.5 & 3.0 \end{pmatrix}$$
$$\mathbf{X}_1 = (X_1) \qquad \mathbf{X}_2 = \begin{pmatrix} X_2 \\ X_3 \end{pmatrix} \mathbf{x}_2 = \begin{pmatrix} 1 \\ -2 \end{pmatrix}$$

When we fill these values in the the formula for the conditional distribution $f_{\mathbf{X}_1|\mathbf{X}_2}(\mathbf{x}_1, \mathbf{X}_2 = \mathbf{x}_2)$ we find the following:

$$f_{\mathbf{X}_1|\mathbf{X}_2}(\mathbf{x}_1, \mathbf{X}_2 = \mathbf{x}_2) = N_1(1, 0.446) \tag{5}$$

The probability $P(X_1 > 1|X_2 = 1, X_3 = -2)$ is equal to 0.5 because of the symetry of the conditional probability distribution around 1.

To calculate the probability $P(X_1 > 1|X_2 + X_3 = 1)$ we first transform the random vector $\mathbf{X}$ to the vector $\mathbf{Y} = (X_1, X_2 + X_3)^T = \mathbf{A}\mathbf{X}$ where

$$\mathbf{A} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 1 \end{pmatrix}$$

Using part (b) of proposition 6.2 we find that the distribution of $\mathbf{Y}$ is given by:

$$\mathbf{Y} \sim N(A\mu, \mathbf{A}\Sigma\mathbf{A}^T) = N(...)$$

Observe that $P(X_1 > 1|X_2 + X_3 = -1) = P(Y_1 > 1|Y_2 = -1)$. We can again use part (c) of proposition 6.2 to calculate this probability. This yields the following conditional distribution:

$$f_{\mathbf{Y}_1|\mathbf{Y}_2}(\mathbf{y}_1|\mathbf{y}_2 = -1) = N(1, 0.9877)$$

Therefore $P(X_1 > 1|X_2 + X_3 = -1) = 0.5$.

(c)

Let $\mathbf{Y} = (X_1, X_2)^T$. Represent graphically the density contours that comprise 95% of the probability mass of $\mathbf{Y}$

To determine the density function of $\mathbf{Y} = (X_1, X_2)^T$ We make again use of proposition 6.2 (c) with the following matrix:

$$\mathbf{A} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix}$$

From which we find the probability distribution of $\mathbf{Y}$

$$\mathbf{Y} \sim N_2\left(\begin{pmatrix} 1 \\ 1 \end{pmatrix}, \begin{pmatrix} 1.25 & 1.50 \\ 1.50 & 5.25 \end{pmatrix}\right)$$

We can now use proposition 6.1 to find the contours of of this density function. The contour that comprises 95% of the probability mass is given by the following equation:

$$Q_{\chi^2_2}(0.95) = (\mathbf{Y} - \mu)^T \Sigma^{-1} (\mathbf{Y} - \mu)$$

This region is plotted in figure 8.

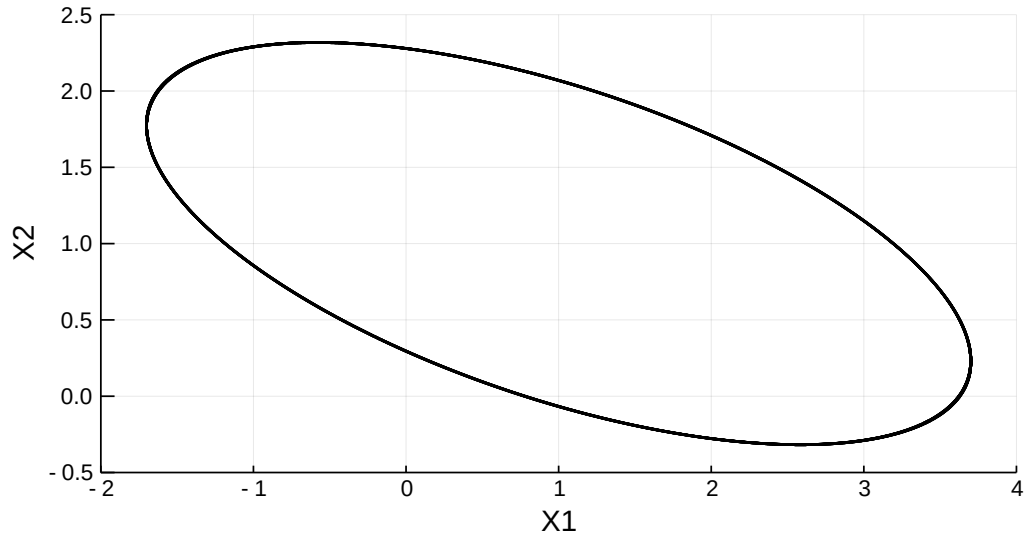

Figure 8: The contour that comprises 95% of the density of the probability density function of the random vector $\mathbf{Y}$.

Acknowledgements

Rune Buckinx and Michaël Maex for helping to fix stupid mistakes in stupid R code Seppe for stupid discussions about how well the questions are asked Thibeau for organising a group crying session Robbe for genuinly good input about trying to solve stupid questions

References

[1] J Beirlandt and I. Gijbels. *Statistical inference and data analysis: course notes*. 2019.

[2] C. Flinn. *Asymptotic Results for the Linear Regression Model*. 1999. URL: `http://www.econ.nyu.edu/user/flinnc/notes1.pdf`.

R-code

The code can also be found at: https://github.com/Cubedsheep/statinf-take-home-2.

```
# R-code for take-home 2 for the course "Statistical inference and data analysis"
# author: Pieter Luyten

##########################################################################
#  ___       _ _   _       _ _          _   _                            #
# |_ _|_ __ (_) |_(_) __ _| (_)______ _| |_(_) ___  _ __                #
#  | || '_ \| | __| |/ _` | | |_  / _` | __| |/ _ \| '_ \               #
#  | || | | | | |_| | (_| | | |/ / (_| | |_| | (_) | | | |              #
# |___|_| |_|_|\__|_|\__,_|_|_/___\__,_|\__|_|\___/|_| |_|              #
#                                                                        #
##########################################################################

library(ggplot2)
# used to output tikz code to render graphs in latex
library(tikzDevice)

# the linewidth of the latex-document in Inches. Used to correctly scale the
# graphs for the report so font sizes are consistent.
LINEWIDTH = 6.02872


###########################################################
#   ___                  _   _                _          #
#  / _ \ _   _  ___  ___| |_(_) ___  _ __    / |         #
# | | | | | | |/ _ \/ __| __| |/ _ \| '_ \   | |         #
# | |_| | |_| |  __/\__ \ |_| | (_) | | | |  | |         #
#  \__\_\\__,_|\___||___/\__|_|\___/|_| |_|  |_|         #
#                                                         #
###########################################################
print("Question 1:")

# read the data and save it to a dataframe
# read the data in ex1.txt and save it to a dataframe
data = read.csv("ex1.txt", sep=" ");
X1 = data$V1
Y = data$V2
X = cbind(rep(1, length(X1)), X1)

#####
# a #
#####
print("q1-a:")

# fit a linear relation with data in column V1 as explanatory variable and
# data in column V2 as dependent
# get the matrix XtX
beta = solve(t(X)%*%X) %*% t(X)%*%Y      # P. 190

# make a plot of the fit
# initalization
tikz(file = "fit_1a.tex", width=0.9*LINEWIDTH, height = 0.7*LINEWIDTH);
par(mfrow=c(1,1))

# plot the data
```

```
plot(X1, Y, xlab="V1", ylab="V2", pch=20, ylim=c(0,2.2))
# plot the line fit
x = seq(min(X1)-0.5, max(X1)+0.5, length.out=20)
coeffs = beta

# print the fitted coefficients
print("coefficienten fit:")
print(coeffs)
y = coeffs[1] + coeffs[2]*x
lines(x, y)
dev.off()


#####
# b #
#####
print("q1-b")
Df = length(X1) - 2             # degrees of freedom of the data
X_inv = solve(t(X)%*%X)         # calculate inverse of the matrix x
C = c(0, 1)                     # we are interested in the second variable (the slope)
RSS = t(Y-X%*%beta) %*% (Y-X%*%beta)    # calculate the sum of squares of the residuals
S2 = RSS/(Df)                   # estimate for the variance of the errors

# calculate the value to test, formula P. 200
test_val = coeffs[2]/sqrt(t(C) %*% X_inv %*% C*S2)
# calculate the 99% confidence region
alpha = 0.01
x_max = qt(1-alpha/2, Df, 0)

# do the test
clevel = pt(test_val, Df, 0)    # chance of x being smaller than test_val.
sprintf("p-value: %f", (1-clevel)*2)    # (1-clevel)*2 is the chance of |x| > test_val
sprintf("test value: %f", test_val)
sprintf("alpha=%f confidence region: [%f, %f]", alpha*100, -x_max, x_max)


#####
# c #
#####
print("q1-c")

# get the residuals
res = Y - X%*%beta
# qq-plot of the residuals
tikz(file = "qq-plot.tex", width = 0.9*LINEWIDTH, height = 0.7*LINEWIDTH)
par(mfrow=c(1, 1))
qqnorm(res, pch=20)
qqline(res)

# add lines to make clear the line goes almost throug (0,0)
lines(c(0,0), c(-1,1), lty=2)
lines(c(-3,3), c(0,0), lty=2)
# save the plot
dev.off()

#####
# d #
#####
```

```
print("q1-d")

# the test value:
test_val = coeffs[2]/sqrt(S2* t(C)%*%X_inv%*%C)
print(test_val)
print(pnorm(test_val))
print((1-pnorm(test_val))*2)
print(qnorm(1-alpha/2))


#####
# e #
#####
print("q1-e")

# test the assumption of normality of the errors with the shapiro-wilk test
print(shapiro.test(res))
# p-value of 0.14, we accept normality.
# the confidence level and corresponding t-value
alpha = 0.01
t_val = qf(1-alpha, 2, Df)
S2 = RSS/Df

sprintf("test value: %.10f", t_val)
sprintf("S^2: %.9f", S2)
print("matrix X^TX: ")
print(t(X)%*%X)
##print(eigen(t(X)%*%X))

# R-code for take-home 2 for the course "Statistical inference and data analysis"
# author: Pieter Luyten

##################################################################
#  ___       _ _   _       _ _          _   _                    #
# |_ _|_ __ (_) |_(_) __ _| (_)______ _| |_(_) ___  _ __         #
#  | || '_ \| | __| |/ _` | | |_  / _` | __| |/ _ \| '_ \        #
#  | || | | | | |_| | (_| | | |/ / (_| | |_| | (_) | | | |       #
# |___|_| |_|_|\__|_|\__,_|_|_/___\__,_|\__|_|\___/|_| |_|       #
#                                                                #
##################################################################
library(ggplot2)
# used to output tikz code to render graphs in latex
library(tikzDevice)

# the linewidth of the latex-document in Inches. Used to correctly scale the
# graphs for the report so font sizes are consistent.
LINEWIDTH = 6.02872


###########################################################
#   ___                  _   _               ____         #
#  / _ \ _   _  ___  ___| |_(_) ___  _ __   |___ \        #
# | | | | | | |/ _ \/ __| __| |/ _ \| '_ \    __) |       #
# | |_| | |_| |  __/\__ \ |_| | (_) | | | |  / __/        #
#  \__\_\\__,_|\___||___/\__|_|\___/|_| |_| |_____|       #
#                                                         #
###########################################################

data = read.csv("ex2.txt", sep=" ");
```

```
X1 = data$V1
Y = data$V2
X = cbind(rep(1, length(X1)), X1)

# plot the data
tikz(file="fit-2.tex", width=0.9*LINEWIDTH, height=0.7*LINEWIDTH);
par(mfrow=c(1,1))
plot(X1, Y, xlab="V1", ylab="V2", pch=20)

#####
# a #
#####
print("q2-a:")

beta = solve(t(X)%*%X) %*% t(X)%*%Y
print(beta)
# calculate the RSS
RSS = t(Y-X%*%beta) %*% (Y-X%*%beta)
print(RSS)

# plot the fit
lines(X1, X%*%beta, col="red")

# print parameters to plot the confidence region
print("parameters for the confidence region:")
sprintf("S2: %.9f", RSS/(length(X1)-2))
print(t(X)%*%X)
sprintf("t_val: %.9f", qf(1-0.01, 2, length(X1)-2))
sprintf("Df: %d", length(X1)-2)

#####
# c #
#####
print("q2-c:")

# construct the diagonal matrix
n = length(data$V1)
m = matrix(nrow=n, ncol=n)
# fill the matrix
sigma = 25/9
rho = 0.8
for (i in 1:(n-2)) {
        diag(m[(i+1):(n),(1):(n-i)]) = rep(sigma*rho^(i), n-i)
        diag(m[(1):(n-i),(i+1):(n)]) = rep(sigma*rho^(i), n-i)
}
m[1,n] = sigma*rho^(n-1)
m[n,1] = sigma*rho^(n-1)
diag(m) = rep(sigma, n)
U = eigen(m)$vec
Lambda = diag(eigen(m)$val)
##print(eigen(m)$val)
##print(U%*%Lambda%*%t(U)-m)
A = sqrt(solve(Lambda))%*%t(U)
##print(Pi %*% m %*% P)
##print(P %*% Pi)
# this is (almost) the identity matrix
Xprime = A %*% X
Yprime = A %*% Y
```

```
##errors = A %*% rep(sqrt(sigma), n)
##errors = errors^2
##print(P)
##print(X2)
##print(Y2)
##print(errors)
i
# do the linear regression
beta = solve(t(Xprime)%*%Xprime) %*% t(Xprime)%*%Yprime
beta2 = solve(t(X)%*%solve(m)%*%X) %*% t(X)%*%solve(m)%*%Y
#print(df)
print(beta)
print(beta2)
# calculate the RSS
RSS = t(Yprime-Xprime%*%beta2) %*% (Yprime-Xprime%*%beta2)
print(RSS)

# print parameters to plot the confidence region
print("parameters for the confidence region:")
sprintf("S2: %.9f", RSS/(length(X1)-2))
print(t(Xprime)%*%Xprime)


# plot the fit
lines(X1, X%*%beta, col="blue")
dev.off()

#############################################################################
#  ___          _ _     _          _ _                 _   _                 #
# |_ _|_ __ (_) |_(_) __ _| |  (_)______ _| |_(_) ___  _ __   #
#  | || '_ \| | __| |/ _` | |  | |_  / _` | __| |/ _ \| '_ \  #
#  | || | | | | |_| | (_| | |  | |/ / (_| | |_| | (_) | | | | #
# |___|_| |_|_|\__|_|\__,_|_|_|_/___\__,_|\__|_|\___/|_| |_| #
#                                                             #
#############################################################################
library(ggplot2)
# used to output tikz code to render graphs in latex
library(tikzDevice)

# the linewidth of the latex-document in Inches. Used to correctly scale the
# graphs for the report so font sizes are consistent.
LINEWIDTH = 6.02872

#################################################################
#   ___                        _   _                  _____    #
#  / _ \ _   _  ___  ___| |_(_) ___  _ __   |___ /  #
# | | | | | | |/ _ \/ __| __| |/ _ \| '_ \    |_ \  #
# | |_| | |_| |  __/\__ \ |_| | (_) | | | |  ___) | #
#  \__\_\\__,_|\___||___/\__|_|\___/|_| |_| |____/  #
#                                                   #
#################################################################

# import the data
data = read.csv("ex3.txt", sep=" ")
X1 = data$V1
Y = data$V2
X = cbind(rep(1, length(X1)), X1, X1^2, X1^3)

#####
```

```
# a #
#####
print("ex-3a:")
# fit the model without taking the size of the errors into account
beta = solve(t(X)%*%X) %*% t(X)%*%Y      # P. 190
print(beta)

# make a plot of the data and the fitted model
# initalization
tikz(file = "fit_3.tex", width=0.9*LINEWIDTH, height = 0.7*LINEWIDTH);
par(mfrow=c(1,1))
# plot the data
plot(X1, Y, xlab="V1", ylab="V2", pch=20)

coeffs = beta
x = seq(min(X1)-0.1, max(X1)+0.1, length.out=100)
y = coeffs[1] + coeffs[2]*x + coeffs[3]*x^2 + coeffs[4]*x^3
lines(x, y, col="blue")


#####
# b #
#####
print("ex-3b:")
# calculate the vector of weights
sigma = (X1 < 4/3)*(X1^2) + (X1>= 4/3)*4*(X1-2)^2
W = diag(sigma^-2)

beta = solve(t(X)%*%W%*%X) %*% t(X)%*%W%*%Y
print(beta)

coeffs = beta
x = seq(min(X1)-0.1, max(X1)+0.1, length.out=100)
y = coeffs[1] + coeffs[2]*x + coeffs[3]*x^2 + coeffs[4]*x^3
lines(x, y, col="red")
dev.off()

# R-code for take-home 2 for the course "Statistical inference and data analysis"
# author: Pieter Luyten

##########################################################################
#   ___       _ _   _       _ _          _   _                           #
#  |_ _|_ __ (_) |_(_) __ _| (_)______ _| |_(_) ___  _ __                #
#   | || '_ \| | __| |/ _` | | |_  / _` | __| |/ _ \| '_ \               #
#   | || | | | | |_| | (_| | | |/ / (_| | |_| | (_) | | | |              #
#  |___|_| |_|_|\__|_|\__,_|_|_/___\__,_|\__|_|\___/|_| |_|              #
#                                                                        #
##########################################################################

library(ggplot2)
# used to output tikz code to render graphs in latex
library(tikzDevice)
library(plot3D)

# the linewidth of the latex-document in Inches. Used to correctly scale the
# graphs for the report so font sizes are consistent.
LINEWIDTH = 6.02872

##################################################################
```

```
#   ___                  _   _               _  _    #
#  / _ \ _   _  ___  ___| |_(_) ___  _ __   | || |   #
# | | | | | | |/ _ \/ __| __| |/ _ \| '_ \  | || |_  #
# | |_| | |_| |  __/\__ \ |_| | (_) | | | | |__   _| #
#  \__\_\\__,_|\___||___/\__|_|\___/|_| |_|    |_|   #
#                                                    #
######################################################

#####
# a #
#####

# the variance-covariance matrix
Sigma = matrix(c(1.25, 1.50, 0.5, 1.50, 5.25, 3.5, 0.5, 3.5, 3.0), nrow=3, ncol=3)
mu = c(1,1,-2)
# calculate the matrix A from the matrix Sigma_X
U = eigen(Sigma)$vec           # the unitary transformation from Sigma to a diagonal m
L = diag(eigen(Sigma)$val)     # the diagonal matrix with eigenvalues on the diagonal
A = U %*% sqrt(L)              # the matrix A

# take 3x200 samples from a standard normal distribution and transform them
samples_standard = matrix(rnorm(3*200), nrow=3, ncol=200)
samples = A %*% samples_standard + mu

tikz(file = "sample-4a-xy.tex", width=0.9*LINEWIDTH, height = 0.7*LINEWIDTH);
par(mfrow=c(2,2))
plot(samples[1,], samples[2,], pch=20, xlab="X1", ylab="X2")
plot(samples[2,], samples[3,], pch=20, xlab="X2", ylab="X3")
plot(samples[3,], samples[1,], pch=20, xlab="X3", ylab="X1")
points3D(samples[1,], samples[2,], samples[3,], pch=".", xlab="X1", ylab="X2", zlab="X3'

# save the projection on the xy-plane
dev.off()

#####
# b #
#####

# calculate the necessary matrices for the conditional distributions
# for the first conditional distribution:
Sigma11 = Sigma[1:1,1:1]
Sigma12 = Sigma[1:1,2:3]
Sigma21 = Sigma[2:3,1:1]
Sigma22 = Sigma[2:3,2:3]

x2 = c(1, -2)

muprime = mu[1] + Sigma12 %*% solve(Sigma22) %*% (x2-mu[2:3])
sprintf("mu_1:")
print(muprime)
sigmaprime = Sigma11 - Sigma12%*%solve(Sigma22)%*%Sigma21
sprintf("sigma_prime:")
print(sigmaprime)

# for the first conditional distribution
# first do the transformation to the random vetor Y:
A = cbind(c(1,0), c(0,1), c(0,1))
mu2 = A %*% mu
```

```
Sigma2 = A %*% Sigma %*% t(A)

# define the submatrices (just numbers in this case as Sigma2 is a 2x2 matrix)
# (but use series indexing to preserve the matrix structure for the following calculatio
Sigma11 = Sigma2[1:1,1:1]
Sigma12 = Sigma2[1:1,2:2]
Sigma21 = Sigma2[2:2,1:1]
Sigma22 = Sigma2[2:2,2:2]

x2 = c(-1)

muprime = mu2[1:1] + Sigma12 %*% solve(Sigma22) %*% (x2-mu2[2:2])
sprintf("mu_2:")
print(muprime)
sigmaprime = Sigma11 - Sigma12%*%solve(Sigma22)%*%Sigma21
sprintf("sigma_prime2:")
print(sigmaprime)

#####
# c #
#####
A = cbind(c(1,0), c(0,1), c(0,0))
Sigma3 = A %*% Sigma %*% t(A)
mu3 = A %*% mu
print(Sigma3)
print(mu3)

print(qchisq(0.95, df=2))
```

Untitled1

December 16, 2019

[1]:
```
import sympy as sp
sp.init_printing()
```

[2]:
```
M = sp.Matrix([[1.25, 1.50], [1.50, 5.25]])
M.eigenvals()
```

[2]:
$$\left\{\frac{3}{4} : 1,\ \frac{23}{4} : 1\right\}$$

[3]:
```
M.eigenvects()
```

[3]:
$$\left[\left(0.75,\ 1,\ \left[\begin{bmatrix}-3.0\\1.0\end{bmatrix}\right]\right),\ \left(5.75,\ 1,\ \left[\begin{bmatrix}0.333333333333333\\1.0\end{bmatrix}\right]\right)\right]$$

[4]:
```
M = sp.Matrix([[60.000000, 4.689758],
[4.689758, 19.063202]])
[sp.N(eig) for eig in M.eigenvals().keys()]
```

[4]:
$$[18.5328108509994,\ 60.5303911490006]$$

[8]:
```
alpha, alpha_h, beta, beta_h, p, S = sp.symbols('beta_0, alpha_h, beta_1,␣
↪beta_h, p, S^2')
X = sp.Matrix([alpha_h-alpha, beta_h-beta])
f = X.transpose() * M * X/(p*S)
display(f[0])
f = f.subs({alpha_h : 1.0321764, beta_h : 0.1904323, p : 2, S : 0.098341450})[0]
sp.expand(f)
```

$$\frac{(\alpha_h - \beta_0)\,(60.0\alpha_h - 60.0\beta_0 - 4.689758\beta_1 + 4.689758\beta_h) + (-\beta_1 + \beta_h)\,(4.689758\alpha_h - 4.689758\beta_0 - 19.063202\beta_1 +}{S^2 p}$$

[8]:
$$305.059565422312\beta_0^2 + 47.6885179138603\beta_0\beta_1 - 638.83200219626\beta_0 + 96.9235352946291\beta_1^2 - 86.1377062422387\beta_1 + 337.895358874081$$

[9]:
```
print(sp.latex(sp.expand(f)))
```

```
305.059565422312 \beta_{0}^{2} + 47.6885179138603 \beta_{0} \beta_{1} -
638.83200219626 \beta_{0} + 96.9235352946291 \beta_{1}^{2} - 86.1377062422387
\beta_{1} + 337.895358874081
```