7089CEM: Statistical Methods Coursework

Antonio Mpembe Franco

Coventry university

Abstract

The paper consists of analyzing the EEG signal data set with univariate and bivariate analysis, from using descriptive statistics to statistical inference by using techniques, such as ordinary least squares to predict and Bayesian inference with approximate Bayesian computation, interpreting the results and communicating them.

Keywords: Univariate, Bivariate, Regression, Bayesian

Preliminary data analysis (Task 1)

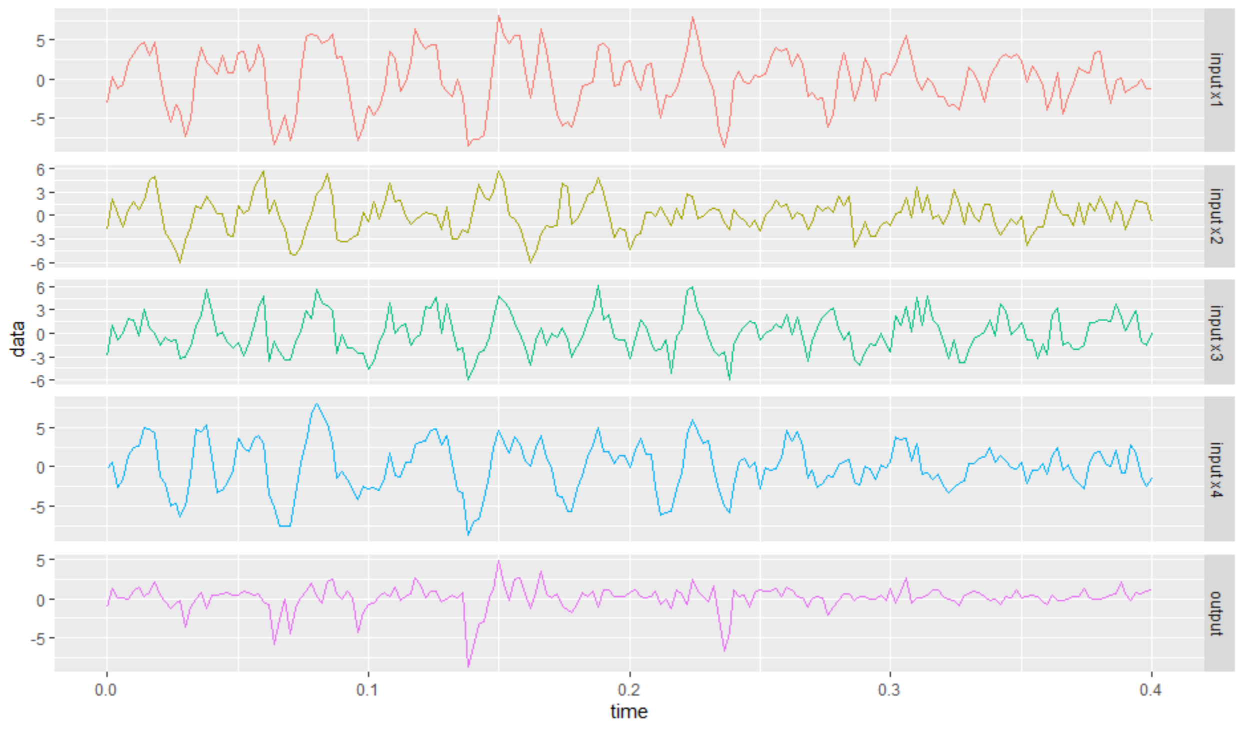
This section consists of a preliminary analysis of the given data from a simulated 5 EEG time-series data. With 4 input EEG signals, an output EEG signal, and a sampling time of the simulated EEG data in seconds.

Time series

This plot represents a time series on some variable input and output where there are 4 different inputs, and one output ate a given time. The data points are indexed in a timely order form.

Figure 1

Time series plot of the input and output data over the given time



Note. Time is in seconds

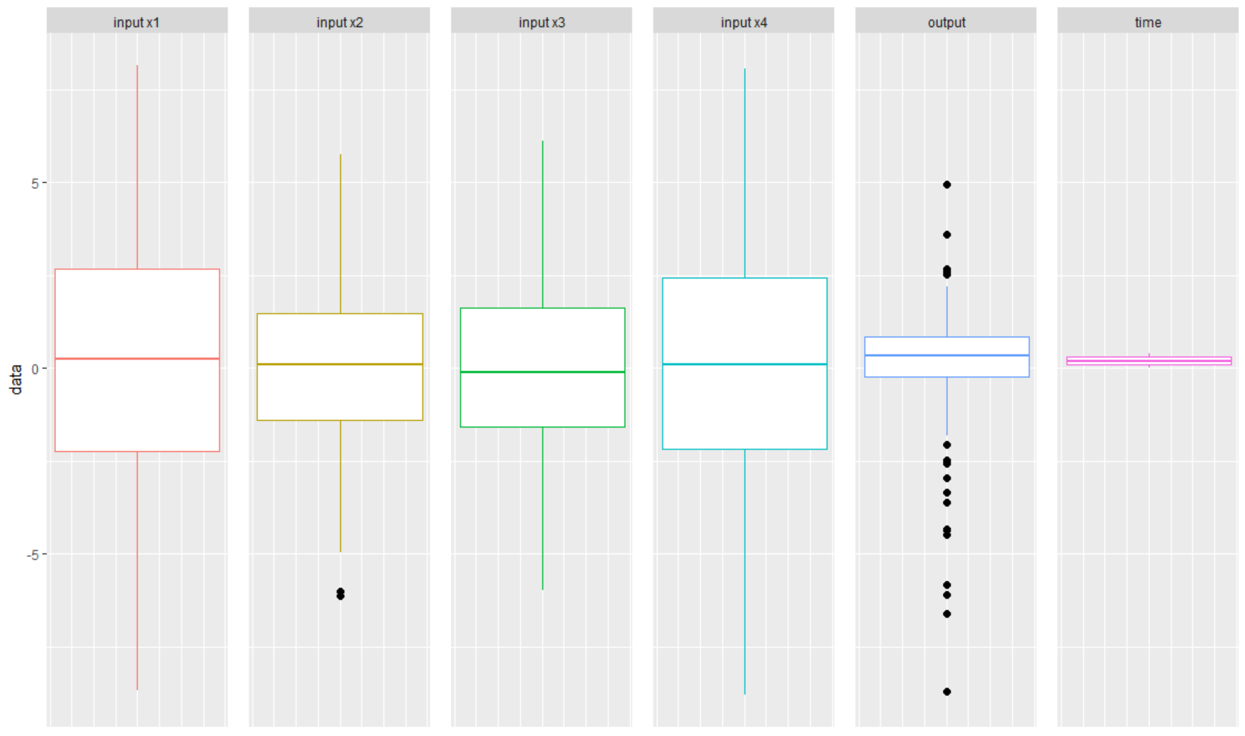
The complete set of times range from *0* to *0.4* seconds and observations are measured at equally spaced intervals of *0.002* seconds. No particular trend is visible, it is clear to see that the data,” input and output”, are volatile, ranging from the minimal to maximal of each input category of the data mainly from second *0* to *0.25*, thus, the pattern that can be observed is a cyclic pattern, as it can be seen the data exhibit a rise and falls that are not in particular sequence or period (Hyndman, 2011), for the output data no clear pattern that can be observed, but what can be seen is a sudden rise to the maximal value and fall to the minimal value, analytically this may indicate outliers in the data.

Univariate analysis (Distribution for each EEG signal and time)

Involves describing a single variable's distribution, including its central pattern and dispersion.

Figure 2

Box plot of all the data



Note. The black points represent the outliers of each distribution

Figure 2, an example of numerical groups of data by their quartiles, can be seen in the plot above. Where lines spread from the boxes that show variability beyond the upper and lower quartiles. *25%* of the data is below the lower quartile and *75%* of the data is below the upper quartile. Thus, the box splits off the highest *25* per cent of data from the lowest *75* per cent of data. The line in the middle of the box is the second quartile, known as median or *50th* percentile.

The lines extending from the boxes indicate variability outside the upper and lower quartiles are calculated with interquartile range (IQR), which is the distance between the upper and lower quartiles $IQR = q_n(0.75) - q_1(0.25)$, and add to the upper quartile and subtract from the lower quartile the following if the medcoupe value of MC is bigger than *0* the value to subtract and add is $1.5 \times IQR \times e^{3 \times MC}, 1.5 \times IQR \times e^{-4 \times MC}$ but if MC is smaller than *0* the value to subtract and add is $1.5 \times IQR \times e^{4 \times MC}, 1.5 \times IQR \times e^{-3 \times MC}$. And if the value is zero it does matter because one of the two can be used. For symmetrical distributions, the medcoupe will be zero, and this reduces to $1.5 \times IQR$ for both lines extending from the boxes indicating variability (Box plot, 2020). Medcoupe is a robust statistic that can be used to measures skewness. It is defined as the difference of the left and right half of a distribution to the median divided to the difference of the left to the right half of a distribution $h(x_i, x_j) = \frac{(x_i - med(x)) - (med(x) - x_j)}{x_j - x_i}$. “It’s robustness makes it suitable for identifying outliers in adjusted boxplots” (Medcouple, 2020).

The box plot outliers are data points that are outside the range of lines extending from the boxes. An outlier is a data point that varies greatly from other data points in the same variable or a combination of several variables. Experimental error or variability in the calculation may cause them. (Outlier, 2020).

Input x1’s lower quartile is *-2.2482* and *2.6706* is the upper quartile, the median is *0.2447*, with an outlier range for values below *-7.764151* and higher than *11.84754.*

Input x2's lower quartile is *-1.40102* and *1.47238* is the upper quartile, the median is *0.08675*, with an outlier range for values below *-5.021092* and higher than *6.385025.*

Input x3's lower quartile is *-1.57921* and *1.62626* is the upper quartile, the median is *-0.09896*, with an outlier range for values below *-7.565909* and higher than *5.705511.*

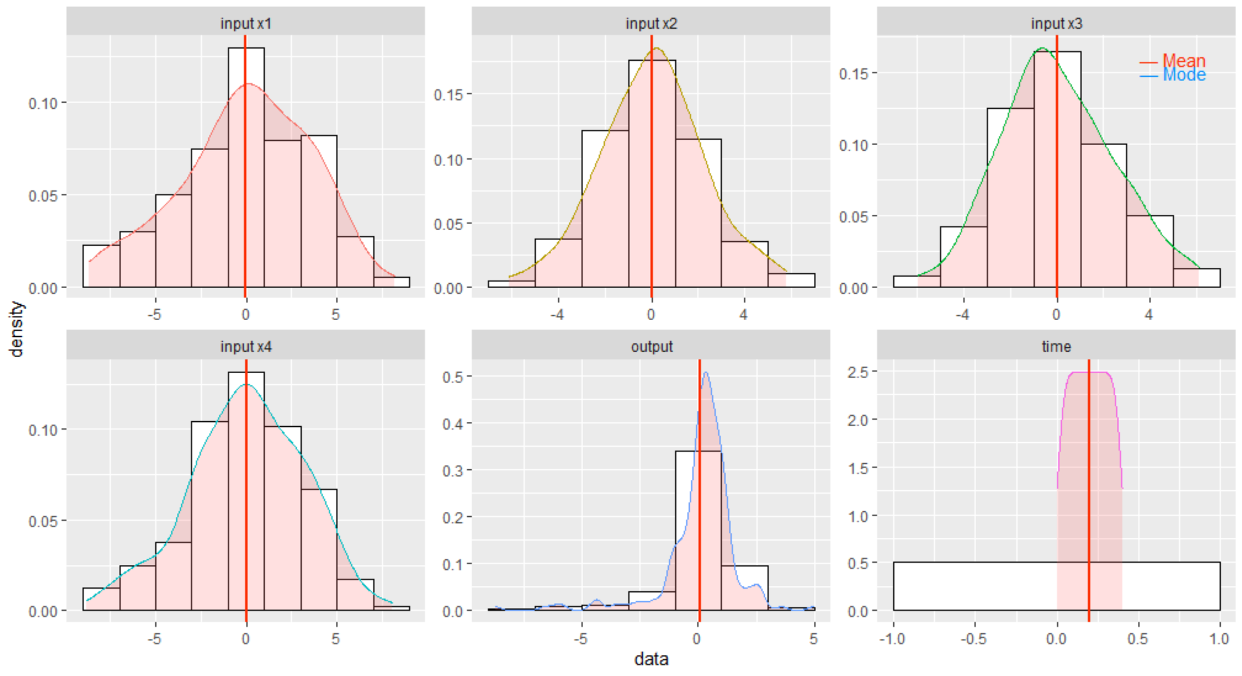
Input x4's lower quartile is *-2.18124* and *2.44413* is the upper quartile, the median is *0.08573*, with an outlier range for values below *-9.156648* and higher than *9.35429.*

Output's lower quartile is *-0.2230* and *0.8544* is the upper quartile, the median is *0.3281*, with an outlier range for values below *-1.242915* and higher than *3.136814.*

Time's lower quartile is *0.1* and *0.3* is the upper quartile, the median is *0.2*, with an outlier range for values below *-0.2* and higher than *0.6.*

Figure 3

Histogram and density of all the data



Note. The histogram of *2* bin range with density kernel

This plot is an approximated representation of the distribution of the numerical data points. The rectangles displayed in the histogram above, Figure 3, are defined for probability density. Thus, the total area of a histogram is *1*.

The density plot curve displayed above, Figure 3, range from the minimal value of the variable to the maximal value. This plot approximates the probability density of the observed data.

As it can be observed from the plot above, Figure 3, there are no mode lines, meaning there are no lines for the value that most often occurs in a set of data values. This is because all values from the *6* different data sets have the same frequency. From what can be observed from Figure 3, most of the data sets have a distribution that is closer to a normal distribution except for the data set of time. For this, the use of skewness, which is important to measure the asymmetry of the distribution, and kurtosis, which describes the shape of a probability distribution by measuring the tails of the distribution. The standard deviation is also relevant to understand the amount of dispersion of the data set. As it can be seen in Figure 3 the data set time show a uniform distribution. The probability density function of continuous uniform distribution or rectangular distribution is defined as $\begin{cases} \frac{1}{\mathfrak{b} - \mathfrak{a}} & for\ x \in [\mathfrak{a}, \mathfrak{b}] \\ 0 & otherwise \end{cases}$, from Figure 3 $[\mathfrak{a} = -1, \mathfrak{b} = 1]$ so $f(x) = \frac{1}{1-(-1)} = 0.5$, from Figure 3 it also can be conformed that $f(x) = 0.5$ and that otherwise is 0, the same thing can with the density plot where $[\mathfrak{a} = 0, \mathfrak{b} = 0.4]$ so $f(x) = \frac{1}{0.4 - 0} = 2.5$, which can be confirmed by the plot that $f(x) = 2.5$.

If the skewness is zero, some well-known distributions can be identified by the kurtosis, such as Laplace distribution with kurtosis of 6, normal distribution with kurtosis of 3, uniform distribution with kurtosis of 1.8 and many others. If the skewness is negative means that more the distribution is concentrated on the right of the mean, if it is positive meaning more of the distribution is concentrated on the left of the mean, and if it is zero means that it is a symmetrical distribution.

Input x1's mean is *-0.0154,* with values varying between *-8.6660 to 8.1559*, with a standard deviation is *3.578254*, with kurtosis of *2.645798* and skewness of *-0.3166455.*

Input x2's mean is *0.03334,* with values varying between *-6.13928 to 5.76075*, with a standard deviation is *2.245316*, with kurtosis of *3.08229* and skewness of *-0.006438418.*

Input x3's mean is *0.03360,* with values varying between *-5.97673 to 6.12148*, with a standard deviation is *2.409854*, with kurtosis of *3.08229* and skewness of *0.2062838.*

Input x4's mean is *0.01156,* with values varying between *-8.79904 to 8.06003*, with a standard deviation is *3.163452*, with kurtosis of *2.872291* and skewness of *0.2062838.*

Output's mean is *0.1085,* with values varying between *-8.7165 to 4.9507,* with a standard deviation is *1.573837*, with kurtosis of *11.23342* and skewness of *-2.036788.*

Time's mean is *0.2,* with values varying between *0 to 0.4,* with a standard deviation is *0.1163357.*

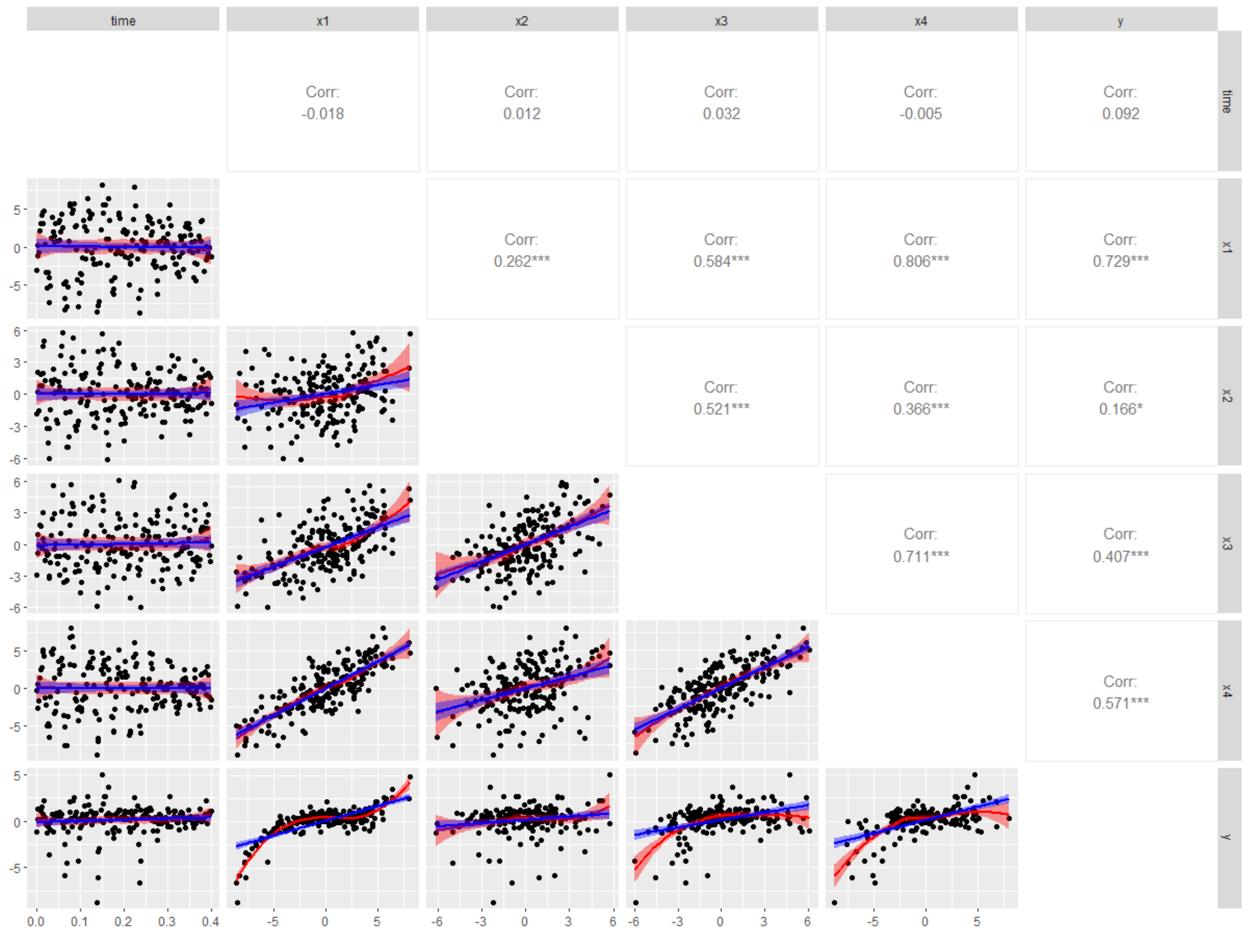
Input x1 is the data with the biggest range and the biggest amount of dispersion, Time data set is the smallest range and smallest amount of dispersion. The data set with the distribution closer to a normal distribution data set is the Input x2 data set and the data set with the distribution most distant from a normal distribution is the output data set.

Bivariate analysis (Correlation and scatter plots)

One of the simplest types of statistical analysis is bivariate analysis. To evaluate the empirical relationship between two variables.

Figure 4

Correlation and scatter plots



Note. Redline represents the local regression, Blueline represents a linear model

The correlation helps to understand the linear statistical relationship between the pair of variables represented in Figure 4. This is just a normalized covariance achieved by dividing it with the product of the standard deviations, $Correlation = \frac{cov(X,Y)}{\sigma X * \sigma Y}$, with a range between -1 and 1 (Correlation and dependence, 2021). Taking a look at the linear relationship between the output and other variables, from Figure 4 it is clear to see that the output is more linearly dependence on the variable x1, *0.729*, and less linearly dependent on the variable x2, *0.166*. The correlation is also illustrated on the scatter plots on Figura 4, where the blue line is a representation of an equation for the correlation between the pair of variables as it is a representation of the best fit linear model. To illustrate a nonlinear relationship between the variable a local regression is represented by a red line as a locally estimated scatterplot smoothing (Scatter plot, 2021). Analysing the representation of the relationship between the output and other variables, the lines can be used to predict the output. And using the Coefficient of determination to evaluate the prediction by finding the ratio variance of the prediction to the variance of the sample, or in other words the ratio between the explained variance to the total variance, $R^2 = \frac{\sum_i(\hat{y}_i - \mu)^2}{\sum_i(y_i - \mu)^2}$, (Coefficient of determination, 2021), which is just a square correlation between two vectors, estimated output and true output.

The equation of time as an explanatory variable to predict the output is $y = -0.1403 + 1.2429 *$ time, with the $R^2 = 0.008$.

The equation of x1 as an explanatory variable to predict the output is $y = 0.1134 + 0.3206 *$ x1, with the $R^2 = 0.531$.

The equation of x2 as an explanatory variable to predict the output is $y = 0.1046 + 0.1166 *$ x2, with the $R^2 = 0.028$.

The equation of x3 as an explanatory variable to predict the output is $y = 0.09955 + 0.26579 *$ x3, with the $R^2 = 0.166$.

The equation of x4 as an explanatory variable to predict the output is $y = 0.1052 + 0.2838 *$ x4, with the $R^2 = 0.325$.

Regression analysis (Task 2)

The regression analysis for this paper will consist of a set of statistical processes for estimating the relationships between a dependent variable, "*y*" or the output EEG signal and independent variables *x* or the input EEG signal.

The ordinary least squares method will be used to compute the hyperplane. And to consider the result underlying assumptions will be made and, such as assuming the sample is representative of the population, following the Gauss-Markov assumptions or conditions, there is the condition of linearity stating that the model must be linear in its parameters, the assumption of strict exogeneity stating that the error-term should not be correlated to explanatory variables, the condition of full rank stating no explanatory variable should be perfectly dependent to others and the spherical errors assumption stating that the error-term should have uniform variance and no serial dependence (Gauss-Markov theorem, 2021). One additional assumption about the data is that it does contain noise that is independent, meaning that the occurrence of one does not affect the odds of the other, and they have the same Gaussian probability distribution with the mean of zero and unknown variance.

Model 1: $y = \theta_1 x_4 + \theta_2 x_1^2 + \theta_3 x_1^3 + \theta_4 x_3^4 + \theta_{bias} + \varepsilon$

Model 2: $y = \theta_1 x_3^3 + \theta_2 x_3^4 + \theta_{bias} + \varepsilon$

Model 3: $y = \theta_1 x_2 + \theta_2 x_1^3 + \theta_3 x_3^4 + \theta_{bias} + \varepsilon$

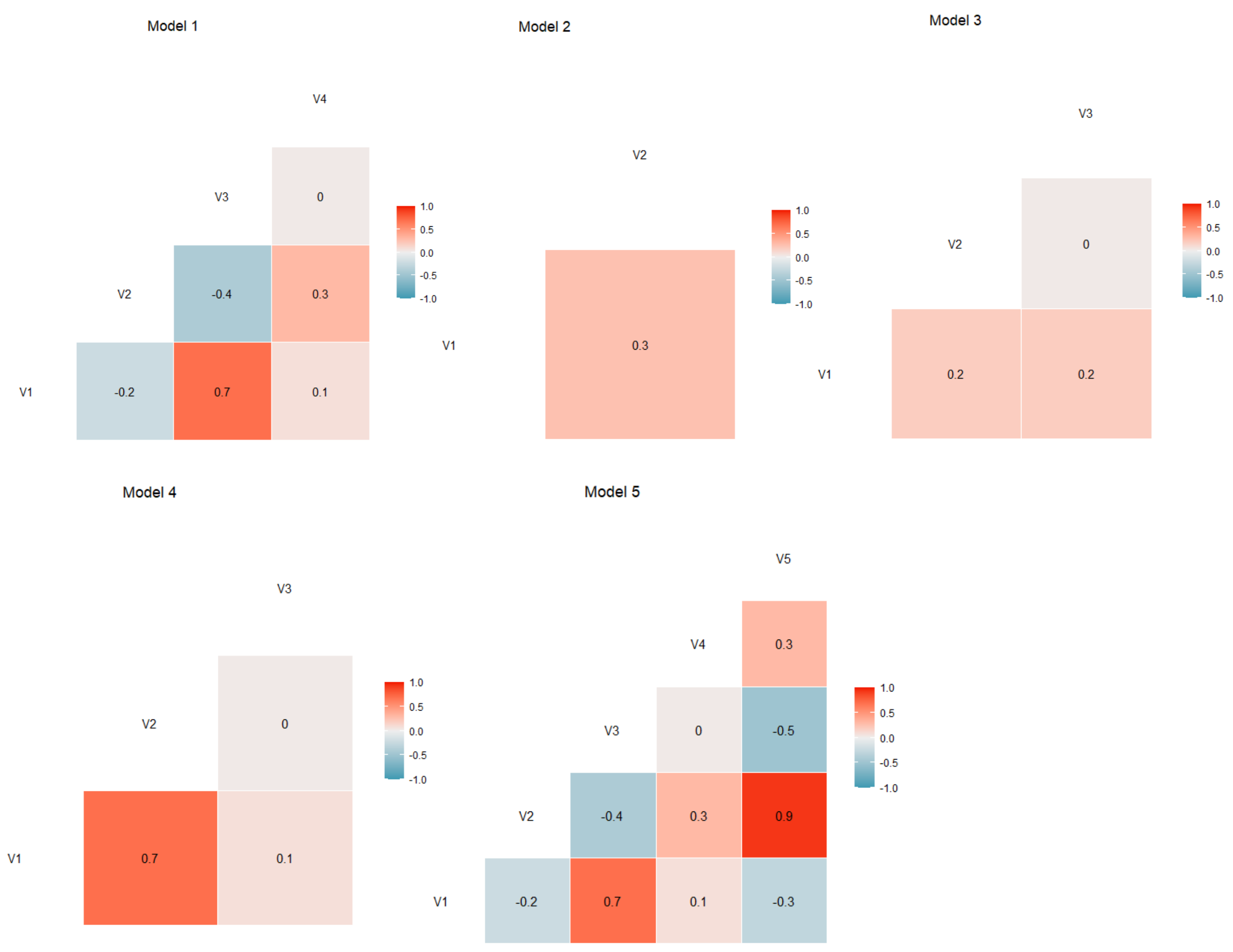
Model 4: $y = \theta_1 x_4 + \theta_2 x_1^3 + \theta_3 x_3^4 + \theta_{bias} + \varepsilon$

Model 5: $y = \theta_1 x_4 + \theta_2 x_1^3 + \theta_3 x_1^3 + \theta_4 x_3^4 + \theta_5 x_1^4 + \theta_{bias} + \varepsilon$

From the presented models, it is clear to observe that the models are linear in their parameters.

Figure 5

Models' parameters Correlation plot



Note. The column of ones was excluded and V1 to Vn represent the order of the variables

As it can be observed from Figure 5, no perfect bivariate correlation but model *5* shows a very high correlation between the 5th variable and the 2nd variable, and on the subject of a high degree of multicollinearity, a computer algorithm may not be able to obtain an approximate inverse, and if it does obtain one it may be numerically inaccurate even if the matrix $X^T X$ is invertible (Multicollinearity, 2021).

Estimation of model parameters (Task 2.1)

The ordinary least squares $\theta = (X^T X)^{-1} X^T y$, where X is the explorative variables and y the output. This method is used to compute unknown parameters from statistical problems involved in linear regression.

These are the estimated parameters for Model 1, $\theta_1 = -0.03410198,\ \theta_2 = -0.001849575,$ $\theta_3 = 0.01038181,\ \theta_4 = -0.001949154,\ \theta_{bias} = $ 0.4685517.

These are the estimated parameters for Model 2, $\theta_1 = $ 0.01633468, $\theta_2 = -0.002713985,$ $\theta_{bias} = 0.3035011$.

These are the estimated parameters for Model 3, $\theta_1 = $ 0.03810926, $\theta_2 = 0.009827804,$ $\theta_3 = -0.002092558,\ \theta_{bias} = 0.4482995$.

These are the estimated parameters for Model 4, $\theta_1 = -0.03488177,\ \theta_2 = $ 0.01048218, $\theta_3 = -0.001990496,\ \theta_{bias} = 0.4503292$.

These are the estimated parameters for Model 5, $\theta_1 = -0.03410198,\ \theta_2 = -0.0002701644,$ $\theta_3 = 0.01034764,\ \theta_4 = -0.001943452,\ \theta_5 = -3.083194e-05,\ \theta_{bias} = $ 0.4685517.

This Regression Coefficients can be interpreted as the marginal effect of having one more of the same designated explorative variable, for example for Model 2: $y = 0.01633468\ x_3^3 +$ $-0.002713985\ x_3^4 + 0.3035011 + \varepsilon$, if everything is maintained constant and adding one more x_3^3, $y = 0.01633468\ (1 + x_3^3) + -0.002713985\ x_3^4 + 0.3035011 + \varepsilon$, the variation of y will be equal to 0.01633468 or $y = \theta_1$, meaning that $\frac{\partial y}{\partial x_3^3} = \theta_1$ (Bounthavong, 2018).

Residual sum of squares (Task 2.2)

The sum of the squares of residuals is the sum of deviations predicted from actual empirical values of data, measuring the discrepancy between an estimation model and the results (Residual sum of squares, 2020).

The use of o RSS to evaluate the models is reasonable in this context since all the models are attempting to predict the same data point. If this was not the case R^2 a better way to evaluate the models, in this context the computing the RSS is as good as R^2, because the total sum of squares

$TSS = \sum_i (y_{i} - \mu)^2$, will be the same for all models and also because $RSS = \sum_i (y_{i} - \hat{y}_i)^2$ is the unexplained variance of the model, and is negatively correlated with $R^2 = 1 - \frac{RSS}{TSS}$.

This is the RSS for Model 1, *57.32927.*

This is the RSS for Model 2, *351.4147.*

This is the RSS for Model 3, *57.32536.*

This is the RSS for Model 4, *57.46405.*

This is the RSS for Model 5, *57.30923.*

From the RSS computed for each model, there is less error in model 5. It would be naïve to say that model 5 is the best because adding explorative variables does reduce the RSS. And model 5 is the most complex.

Log-likelihood function (Task 2.3)

Log-likelihood function, $ln\ p(D|\theta) = -\frac{n}{2} ln(2\pi) - \frac{n}{2} ln(\hat{\sigma}^2) - \frac{1}{2\hat{\sigma}^2} RSS$, is another statistical measurement of the goodness of fit of a model just like R^2 that is a logarithmic transformation of a describe "hypersurface whose peak, if it exists, represents the combination of model parameter values that maximize the probability of drawing the sample obtained" (Likelihood function, 2020).

This is the log-likelihood for Model 1, *-159.1313.*

This is the log-likelihood for Model 2, *-341.3534.*

This is the log-likelihood for Model 3, *-159.1244.*

This is the log-likelihood for Model 4, *-159.3673.*

This is the log-likelihood for Model 5, *-159.0962.*

In this context it fair to say that computing the likelihood function is futile, because the models are fitted to the same data, thus the only non-constant variable is RSS and because it is the only non-constant variable of $\hat{\sigma}^2$, then there is a straightforward relationship between the likelihood function, R^2 and residual sum of squares. But the likelihood function can then use to compute other measurements to evaluate the models. As it can be observer into the next topic.

Akaike information criterion and Bayesian information criterion (Task 2.4)

The Akaike information criterion (AIC) and Bayesian criterion information (BIC) are estimators of prediction error dealing with the trade-off between the goodness of fit of the model and the simplicity of the model, meaning deals with the risk both overfitting and underfitting.

The difference between AIC and BIC is how the number of parameters k , Where $AIC = 2 \cdot k - 2 \cdot log - likelihood$ and $BIC = k \cdot ln\,(number\ of\ data\ points) - 2 \cdot log - likelihood$.

These are the AIC and BIC for Model 1 respectively, *328.2626* and *344.7791*.

These are the AIC and BIC for Model 2 respectively, *688.7068* and *698.6168.*

These are the AIC and BIC for Model 3 respectively, *326.2489* and *339.4621*.

These are the AIC and BIC for Model 4 respectively, *326.7346* and *339.9478.*

These are the AIC and BIC for Model 5 respectively, *330.1923* and *350.0121.*

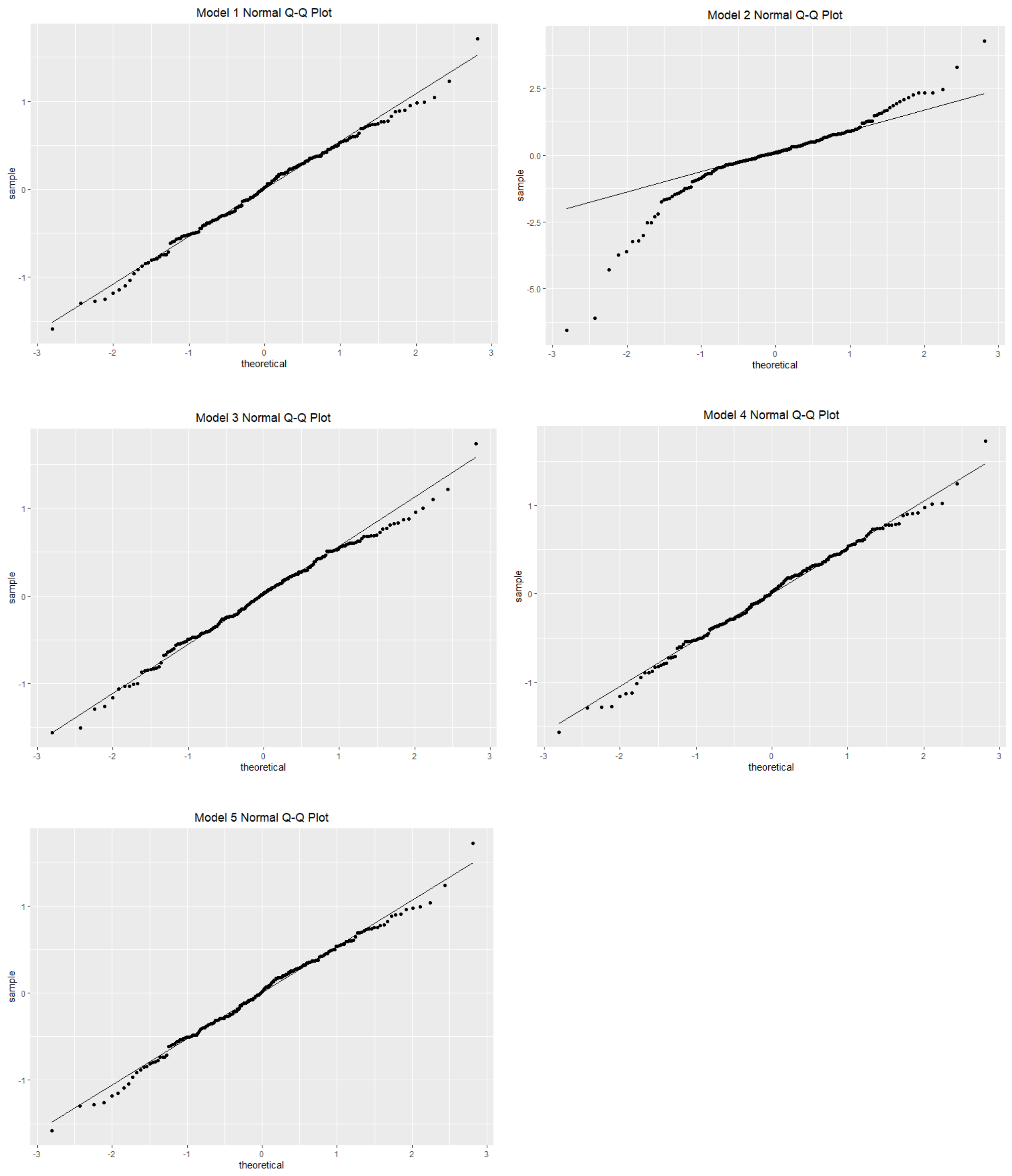
Residual distribution (Task 2.5)

Residual and error are two different concepts that are they apply to two different worlds, on one hand, the residual is the difference between the observed value and the estimated value, and the error is the deviation of the observed value from the true value (Errors and residuals, 2021).

Normality of residual is not necessarily good at least without any additional assumptions than the Gauss–Markov assumptions, but in this context, it is good, because it stated that the noise is independent and identically normally distributed, with a mean of zero and unknown variance. This means that the residual produced by the prediction should behave similarly.

Figure 6

Normal quantile-quantile plots of residuals for each model



Note. Theoretical values are the z-score value of the sample values, which are the residual values

Most of the models except for model 2 have their residuals distribution are similar to a normal distribution.

These are the kurtosis and skewness of Model 1 respectively *3.109115*, *-0.151375,* and variance of *0.2866463.*

These are the kurtosis and skewness of Model 2 respectively *8.507902*, *-1.324471,* and variance of *1.757073.*

These are the kurtosis and skewness of Model 3 respectively *3.225729*, *-0.1934425,* and variance of *0.2866268.*

These are the kurtosis and skewness of Model 4 respectively *3.133473*, *-0.1424218,* and variance of *0.2873203.*

These are the kurtosis and skewness of Model 5 respectively *3.127801*, *-0.150944,* and variance of *0.2865462.*

With all model's mean close to zero, ranging from *-3.916175e-15* to *4.325433e-16*.

For the independence of the noise, the Breusch–Godfrey test is used validity of the assumption with a null hypothesis that says that there is no serial correlation of any order up to p. In this context, the p or order of freedom value will be 1. And the percentages of no serial correlation of order 1, at least as extreme, given the residuals are 56%, 0.01%, 67%, 57% and 57% for Model 1, Model 2, Model 3, Model 4 and Model 5 respectively. Thus, rejecting the null hypothesis for Model 2, because is below 5%.

Machine learning regression prediction (Task 2.6)

Given the task done in this section of regression analysis (Task 2) the best regression model according to the AIC, BIC and distribution of model residuals is Model 3. Because it has the lowest AIC and BIC and from the distribution of model residuals and most of the models except

Model 2's residuals did behave similarly to the assumption given about the noise. Well, some of the assumptions from the Gauss-Markov theorem were not tested, such as strict exogeneity and heteroskedasticity due to the constraints of this paper.

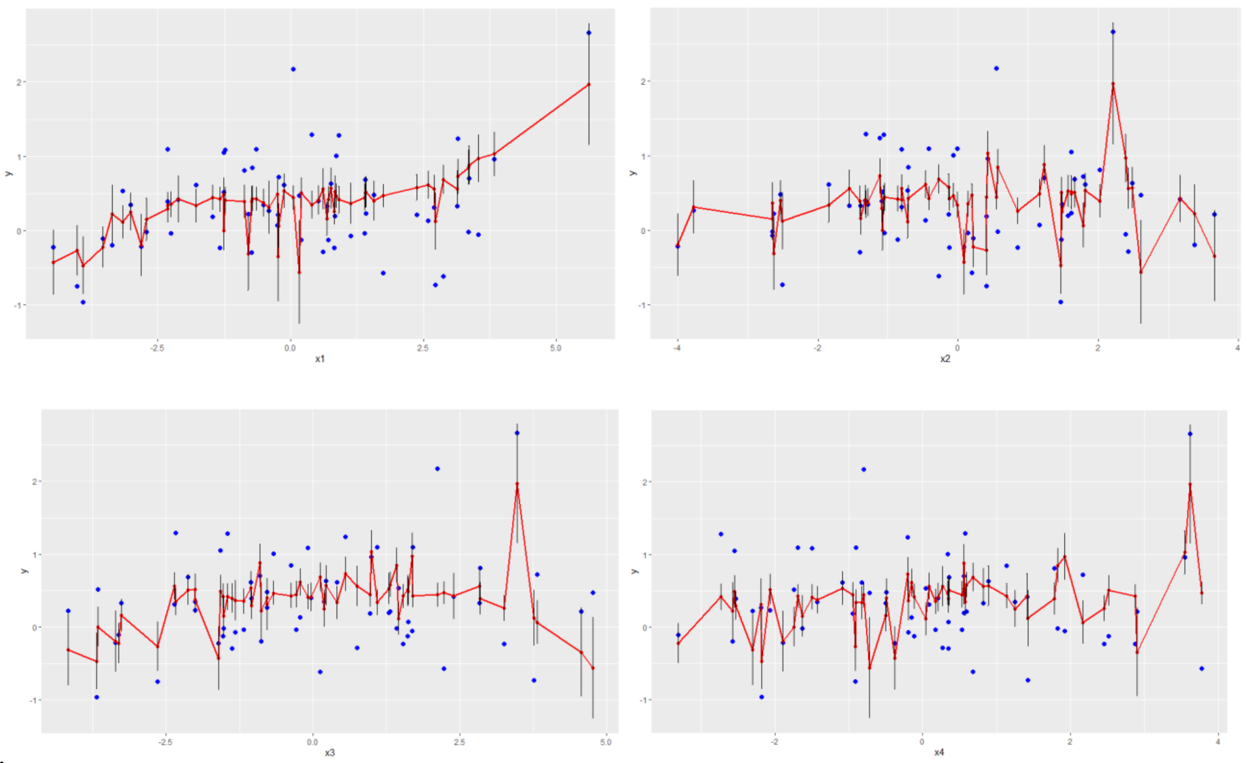
Machine learning regression prediction (Task 2.7)

This task is about training the model with a subset of 70% of the data and testing the model to an unseen subset of 30% of the data.

The model that is used in this task is model 3, which was considered to be the best in the previous task of comparing models. In this case with $R^2 = 0.21$, $\theta_1 = 0.039872473$, $\theta_2 = 0.009830500$, $\theta_3 = -0.002197782$, $\theta_{bias} = 0.462816573$.

Figure 7

Error bars plot for each input variable



.

Note. The blue points represent the output value, the redline connect the predicted output and the black bar represent the 95% confidence interval predicted output

Estimated outputs are subject to uncertainty, the confidence interval estimates a range of plausible values from the observed data for the true value in this case with a confidence level of 95%. So, any value proposed in this range would be accepted as the true value, and any value proposed outside the range would be rejected.

Approximate Bayesian Computation (ABC) (Task 3)

This task is to compute the posterior distributions of 2 coefficients with the largest absolute values from what is considered to be the best model from this paper's regression analysis, using rejection ABC.

These are coefficients of Model 3, $\theta_1 = 0.03810926$, $\theta_{bias} = 0.4482995$.

Prior

The probability distribution that would express the beliefs about this quantity before some evidence of rejection ABC and based on the coefficients.

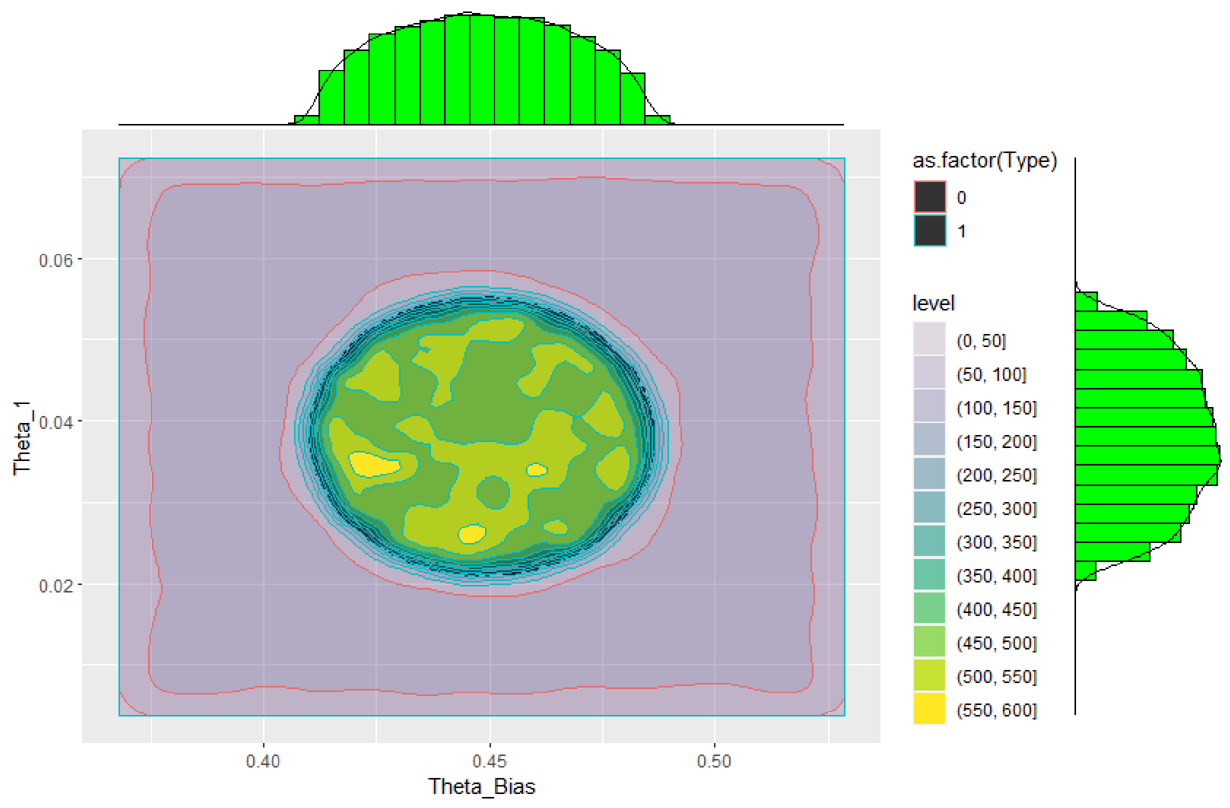
Based on the coefficients θ_1, θ_{bias}, ranging approximately from the coefficient values - their (standard deviation = *0.0174904086*, *0.0409993310* respectively * z-score of 95% = 1.96) to approximately the coefficient values + their (standard deviation * z-score of 95%), with 100000 samples uniformly distributed.

Posterior distributions

This is the conditional probability that is assigned after the relevant evidence of rejection ABC.

Figure 8

Joint and marginal distribution plot



Note. Redline area with the rejected coefficients and blueline accepted

This is process is expressed by the relation between the posterior knowledge, the prior knowledge, and the knowledge of the distance between the prior model *RSS* and updated model *RSS* compared to the mean squared error of the prior model.

The joint probability of the 2 coefficients being accepted is *0.003%*. The marginal distribution of both coefficients is a beta distribution, which is just a binomial distribution but where the probability is a random variable, $g(p) = \frac{1}{B(\propto,\beta)} 1^{\propto - 1}(1 - p)^{\beta - 1}$, $\propto - 1$ can be thought of as the number of success and $\beta - 1$ as the number of failures. Both marginal distributions are approximately X ~ Beta(7/5, 8/5), θ_1 *and* θ_{bias} with the expected value *0.0380966* and *0.4485198* respectively.

References

Bounthavong, M. (2018, July 3). *Estimating marginal effects using Stata Part 1 – Linear models*. Retrieved from mark bounthavong: https://mbounthavong.com/blog/2018/7/3/estimating-marginal-effects-using-stata-part-1-linear-models

Box plot. (2020, December 24). Retrieved from Wikipedia: https://en.wikipedia.org/wiki/Box_plot

Coefficient of determination. (2021, February 2). Retrieved from Wikipedia: https://en.wikipedia.org/wiki/Coefficient_of_determination

Correlation and dependence. (2021, January 31). Retrieved from Wikipedia: https://en.wikipedia.org/wiki/Correlation_and_dependence

Errors and residuals. (2021, February 10). Retrieved from Wikipedia: https://en.wikipedia.org/wiki/Errors_and_residuals

Gauss-Markov theorem. (2021, January 8). Retrieved from Wikipedia: https://en.wikipedia.org/wiki/Gauss%E2%80%93Markov_theorem

Hyndman, R. J. (2011, December 14). *Cyclic and seasonal time series*. Retrieved from Rob J Hyndman: https://robjhyndman.com/hyndsight/cyclicts/

Likelihood function. (2020, November 26). Retrieved from Wikipedia: https://en.wikipedia.org/wiki/Likelihood_function

Medcouple. (2020, May 21). Retrieved from Wikipedia: https://en.wikipedia.org/wiki/Medcouple

Multicollinearity. (2021, February 19). Retrieved from Wikipedia: https://en.wikipedia.org/wiki/Multicollinearity

Outlier. (2020, November 26). Retrieved from Wikipedia: https://en.wikipedia.org/wiki/Outlier

Residual sum of squares. (2020, December 13). Retrieved from Wikipedia:

https://en.wikipedia.org/wiki/Residual_sum_of_squares

Scatter plot. (2021, February 17). Retrieved from Wikipedia:

https://en.wikipedia.org/wiki/Scatter_plot

Appendix

```
#install.packages("tidyverse")
#install.packages("corrplot")
#install.packages("GGally")
#install.packages('mrfDepth')
#install.packages("equatiomatic")
#install.packages("scales")
#install.packages("ggExtra")
library(lmtest)

## Loading required package: zoo

##
## Attaching package: 'zoo'

## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric

library(forecast)

## Registered S3 method overwritten by 'quantmod':
##   method            from
##   as.zoo.data.frame zoo

library(ggExtra)

## Warning: package 'ggExtra' was built under R version 4.0.4

library(equatiomatic)
library(ggcorrplot)

## Loading required package: ggplot2

library(tidyverse)

## -- Attaching packages --------------------------------------- tidyverse 1.
3.0 --

## v tibble  3.0.5     v dplyr   1.0.3
## v tidyr   1.1.2     v stringr 1.4.0
## v readr   1.4.0     v forcats 0.5.1
## v purrr   0.3.4

## -- Conflicts ------------------------------------------ tidyverse_conflict
s() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()

library(corrplot)

## corrplot 0.84 loaded
```

```
library(GGally)

## Registered S3 method overwritten by 'GGally':
##   method from
##   +.gg   ggplot2

library(mrfDepth)
library(scales)

##
## Attaching package: 'scales'

## The following object is masked from 'package:purrr':
##
##     discard

## The following object is masked from 'package:readr':
##
##     col_factor

library(rayshader)

## Warning: package 'rayshader' was built under R version 4.0.4

#Importing data
  ## importing the file time
```

@param t is a variable that reads the csv file of the variable time

```
t = read.csv("~/Introduction to Statistical Methods for Data Science/time.csv
", header = FALSE, stringsAsFactors=FALSE)
t <- as.numeric(t[,1])

## Warning: NAs introduced by coercion

t[is.na(t)] <- 0
```

@param time is a data matrix of t

```
time <- data.matrix(t)
```

@param time_stats is the stats of time, such as minimal, mean and maximal

```
time_stats <- c(min(time, na.rm = TRUE),
                mean(time, na.rm = TRUE),
                max(time, na.rm = TRUE))
####
  ## importing the file input (x)
```

@param x t is a variable that reads the csv file of the variable input/x

```
x = read.csv("~/Introduction to Statistical Methods for Data Science/x.csv",
header = FALSE, stringsAsFactors=FALSE)
```

@param input is a data matrix of x

```
input = data.matrix(x[,1])
for (i in 2:length(x)) {
  xNextCol = matrix(x[,i])
  input = cbind(input, xNextCol)
}
input[is.na(input)] <- 0
```

@param input_stats is the stats of input, such as minimal, mean and maximal

```
input_stats <- c(min(input, na.rm = TRUE),
                mean(input, na.rm = TRUE),
                max(input, na.rm = TRUE))
####
  ##import the file output (y)
```

@param y t is a variable that reads the csv file of the variable output/y

```
y = read.csv("~/Introduction to Statistical Methods for Data Science/y.csv",
header = FALSE, stringsAsFactors=FALSE)
```

@param output is a data matrix of y

```
output <- data.matrix(y[,1])
```

@param output_stats is the stats of output, such as minimal, mean and maximal

```
output_stats <- c(min(output, na.rm = TRUE),
                mean(output, na.rm = TRUE),
                max(output, na.rm = TRUE))

timeM1 = matrix("input x1" , length(input[,1]) , 1)
timeM = cbind(input[,1], timeM1)
timeM = cbind(timeM, time[,1])
timeM2 = matrix("input x2" , length(input[,1]) , 1)
timeM_1 = cbind(input[,2], timeM2)
timeM_1 = cbind(timeM_1, time[,1])
timeM3 = matrix("input x3" , length(input[,1]) , 1)
timeM_2 = cbind(input[,3], timeM3)
timeM_2 = cbind(timeM_2, time[,1])
timeM4 = matrix("input x4" , length(input[,1]) , 1)
timeM_3 = cbind(input[,4], timeM4)
timeM_3 = cbind(timeM_3, time[,1])
timeM5 = matrix("output" , length(input[,1]) , 1)
timeM_4 = cbind(output[,1], timeM5)
timeM_4 = cbind(timeM_4, time[,1])
timeM = rbind(timeM, timeM_1)
timeM = rbind(timeM, timeM_2)
timeM = rbind(timeM, timeM_3)
```

@param timeM is a reshape of the data y, x1, x2, x3, x4 and time into time, variable and the corresponding variable (number of variable except time * length(variable), 3) matrix

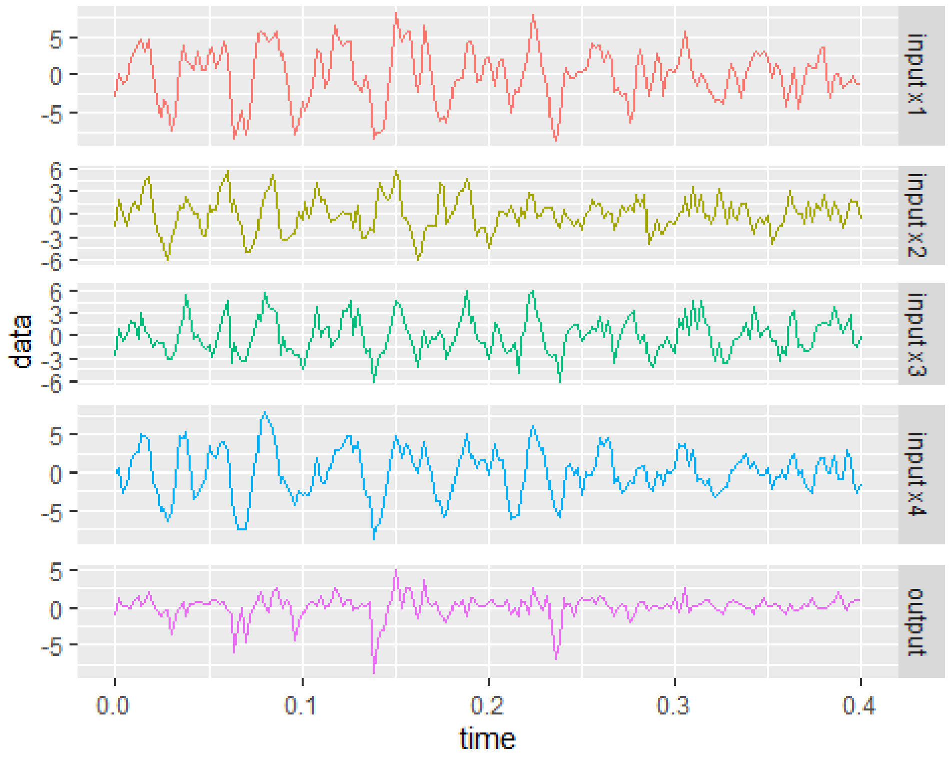
```
timeM = rbind(timeM, timeM_4)
```

@param timeDf is a data frame of the variable timeM

```
timeDf <- data.frame("time" = sapply( timeM[,3], as.numeric),
                     "data" = sapply( timeM[,1], as.numeric),
                     "type" = timeM[,2])
####
```

@param time_series_plot is a variable for the plotting of the facet time series

```
time_series_plot <- ggplot(timeDf, aes(x=time, y=data, group = type, colour =
 type)) +
  theme(panel.spacing = unit(0.5, "lines")) +
  facet_grid(type ~ .,scales = "free", space = "free") +
  geom_line() +
  guides(color=FALSE)

#plotting time series
plot(time_series_plot)
```



```
boxdata = c()
boxdata = matrix(timeM[,1])
```

@param boxdata is a matrix of the variables (number of variable * length(variable), 1) matrix

```
boxdata = rbind(boxdata, matrix(timeM[c(1:201),3]))
boxtype = c()
boxtype = matrix("time" , length(input[,1]) , 1)
```

@param boxtype is a matrix of corresponding a variable (number of variable * length(variable), 1) matrix

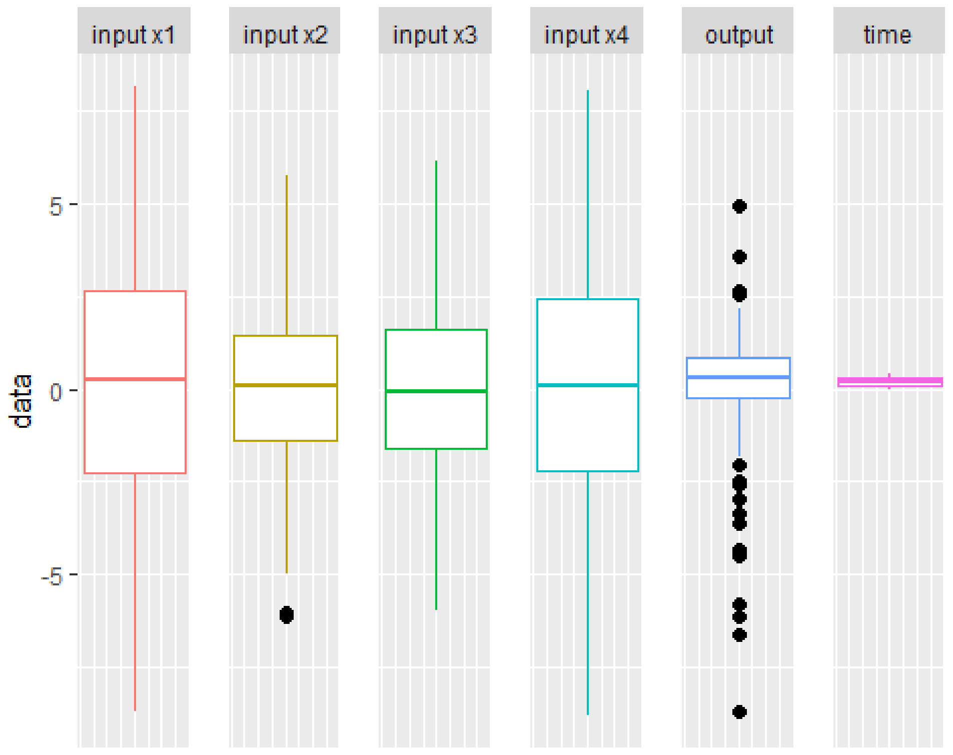
```
boxtype = rbind(matrix(timeM[,2]), boxtype)
```

@param boxDf is the data frame that combines boxtype and boxdata(number of variable * length(variable), 1) matrix

```
boxDf <- data.frame( "data" = sapply( boxdata[,1], as.numeric),
                     "type" = boxtype[,1])
####
```

@param is a variable for plotting the facet box plot

```
boxp <- ggplot(boxDf, aes(y=data, group = type, colour = type)) +
  facet_grid( ~ type)+theme(panel.spacing = unit(1, "lines")) +
  geom_boxplot(outlier.colour="black", outlier.shape=16,
               outlier.size=2, notch=FALSE)+
  guides(color=FALSE)+
  theme(axis.title.x=element_blank(),
        axis.text.x=element_blank(),
        axis.ticks.x=element_blank())

#plotting box plot
plot(boxp)
```



```
####
```

@param mydata is a data frame to the variables time, x1, x2, x3, x4

```
mydata <-  list("time" = time[,1],
                "x1" = input[,1],
                "x2" = input[,2],
                "x3" = input[,3],
                "x4" = input[,4],
                "y" = output[,1])
df = data.frame(mydata)
####
  ##Function that calculate the range outliers of one variable
outlier_range_fn <- function(data){
  Q1 = unname(quantile(data, 0.25))
  Q3 = unname(quantile(data, 0.75))
  med1 = medcouple(data, do.reflect = NULL)
  range = (Q3 - Q1)*1.5
  Out1LB = c()
  Out1UB = c()
  if(med1[1] < 0){
    Out1LB = Q1 - range*exp(4*med1[1])
    Out1UB = Q3 + range*exp(-3*med1[1])
  }else{
    Out1LB = Q1 - range*exp(4*med1[1])
    Out1UB = Q3 + range*exp(-3*med1[1])
  }
  outList = list("low" = Out1LB, "higher" = Out1UB[1])
  outList
}
####
  ##Function that calculate the range outliers of multiple variable
```

@param data is vectors of data points

```
m_outlier_range_fn <- function(data){
  for (i in 1:length(data[1,])) {
    print(toString(names(df)[i]))
    print(outlier_range_fn(data[,i]))
  }
}

#printing outlier range
print(m_outlier_range_fn(df))
```

```
## [1] "time"
## $low
## [1] -0.2
##
## $higher
## [1] 0.6
##
## [1] "x1"
## $low
## [1] -7.764256
##
## $higher
## [1] 11.84771
##
## [1] "x2"
## $low
## [1] -5.021092
##
## $higher
## [1] 6.385025
##
## [1] "x3"
## $low
## [1] -7.565909
##
## $higher
## [1] 5.705511
##
## [1] "x4"
## $low
## [1] -9.156648
##
```

```
## $higher
## [1] 9.35429
##
## [1] "y"
## $low
## [1] -1.242915
##
## $higher
## [1] 3.136814
##
## NULL
```

```
#printing a summary of each variable
print(summary(df))
```

```
##       time            x1                  x2                  x3
##  Min.   :0.0   Min.   :-8.6660   Min.   :-6.13928   Min.   :-5.97673
##  1st Qu.:0.1   1st Qu.:-2.2482   1st Qu.:-1.40102   1st Qu.:-1.57921
##  Median :0.2   Median : 0.2447   Median : 0.08675   Median :-0.09896
##  Mean   :0.2   Mean   :-0.0154   Mean   : 0.03334   Mean   : 0.03360
##  3rd Qu.:0.3   3rd Qu.: 2.6706   3rd Qu.: 1.47238   3rd Qu.: 1.62626
##  Max.   :0.4   Max.   : 8.1559   Max.   : 5.76075   Max.   : 6.12148
##        x4                  y
##  Min.   :-8.79904   Min.   :-8.7165
##  1st Qu.:-2.18124   1st Qu.:-0.2230
##  Median : 0.08573   Median : 0.3281
##  Mean   : 0.01156   Mean   : 0.1085
##  3rd Qu.: 2.44413   3rd Qu.: 0.8544
##  Max.   : 8.06003   Max.   : 4.9507
```

@param x is a group of data point @return The skewness of x

```
skewness_fn <- function (x){
  (sqrt(length(x)) * sum( (x - mean(x))^3 ) ) / (sqrt( sum( (x - mean(x))^2 )
 ))^3
}
```

@param x is a group of data point @return The kurtosis of x

```
myKurosis <- function(data){
  kurtosis = ((sum((data - mean(data))^4)/length(data))/
                ((sum((data - mean(data))^2)/length(data))^2))
  kurtosis
}

#printing the standard deviation of the variable time
sd(df$time)
```

```
## [1] 0.1163357
```

```
#printing the standard deviation of the variable x1
sd(df$x1)
```

```
## [1] 3.578254

#printing the standard deviation of the variable x2
sd(df$x2)

## [1] 2.245316

#printing the standard deviation of the variable x3
sd(df$x3)

## [1] 2.409854

#printing the standard deviation of the variable x4
sd(df$x4)

## [1] 3.163452

#printing the standard deviation of the variable x5
sd(df$y)

## [1] 1.573837

#printing the skewness of the variable time
skewness_fn(df$time)

## [1] -2.575346e-16

#printing the skewness of the variable x1
skewness_fn(df$x1)

## [1] -0.3166455

#printing the skewness of the variable x2
skewness_fn(df$x2)

## [1] -0.006438418

#printing the skewness of the variable x3
skewness_fn(df$x3)

## [1] 0.2062838

#printing the skewness of the variable x4
skewness_fn(df$x4)

## [1] -0.244741

#printing the skewness of the variable y
skewness_fn(df$y)

## [1] -2.036788

#printing the kurtosis of the variable time
myKurosis(df$time)
```

```
## [1] 1.799941

#printing the kurtosis of the variable x1
myKurosis(df$x1)

## [1] 2.645798

#printing the kurtosis of the variable x2
myKurosis(df$x2)

## [1] 3.08229

#printing the kurtosis of the variable x3
myKurosis(df$x3)

## [1] 2.76076

#printing the kurtosis of the variable x4
myKurosis(df$x4)

## [1] 2.872291

#printing the kurtosis of the variable y
myKurosis(df$y)

## [1] 11.23342
```

@param x is a group of data point @return The mode of x

```
Mode <- function(x) {
  ux <- unique(x)
  us <- unique(x)
  tab <- tabulate(match(x, ux))
  ux[tab == max(tab)]
  if(length(ux) == length(us)){
    return(NA)
  }else{
    return(ux[tab == max(tab)])
  }
}

####
```

@param vline_mean is a data frame with the Setting the mean value of each variable

```
vline_mean <- data.frame(
  type = c("input x1", "input x2", "input x3",
             "input x4", "output", "time"),
  mean = c(mean(df$x1), mean(df$x2), mean(df$x3),
           mean(df$x4), mean(df$y), mean(df$time)))
```

@param vline_mode is a data frame with the Setting the mode value of each variable

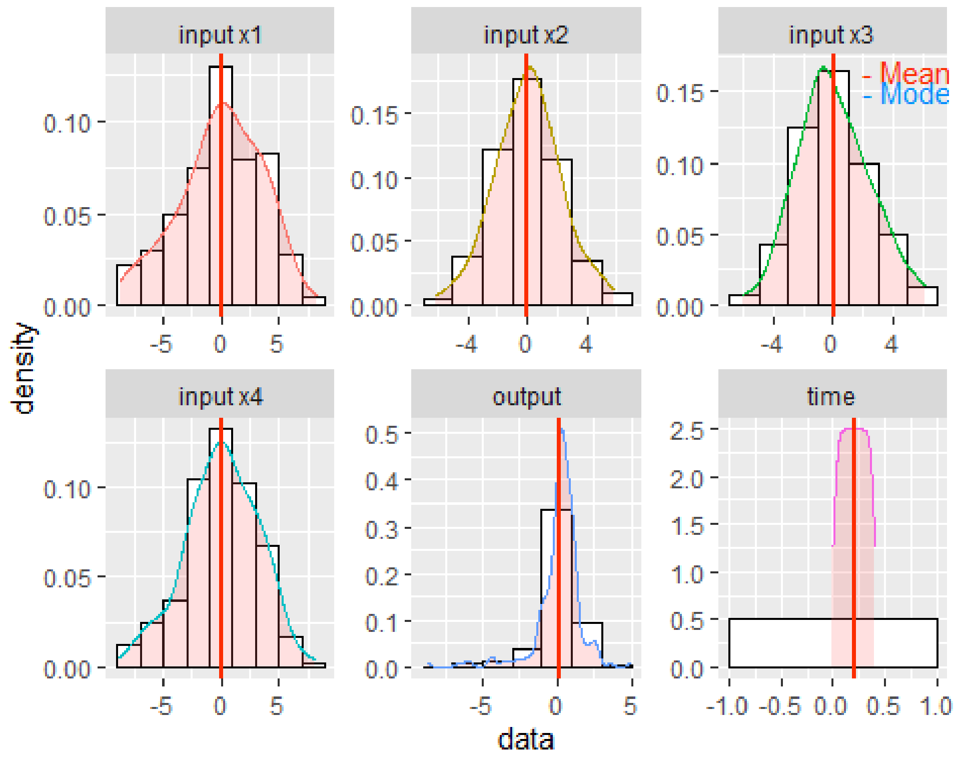
```
vline_mode <- data.frame(
  type = c("input x1", "input x2", "input x3",
           "input x4", "output", "time"),
  mode = c(Mode(df$x1), Mode(df$x2), Mode(df$x3),
           Mode(df$x4), Mode(df$y), Mode(df$time)))
```

@param ann_text is a data frame with the Setting for the annotation of the mean and mod

```
ann_text <- data.frame(x = 5, y = 0.15,lab = c("- Mean\n", "- Mode"),
                       type = factor("input x3",levels = c("input x1", "input
 x2", "input x3",
                                                          "input x4", "output", "time
")))
```

@param histp is plotting variable for the facet histogram and the density and the mean and mode line

```
histp <- ggplot(boxDf, aes(x=data, group = type, colour = type)) +
  geom_histogram(aes(y = stat(density)), binwidth = 2,colour="black", fill="w
hite" ) +
  facet_wrap(~ type, scales = "free")+
  geom_density(alpha=.2, fill="#FF6666") +
  geom_vline(data= vline_mean, aes(group = type, xintercept=as.numeric(mea
n)),col='#FC2C00',size=1)+
  geom_vline(data= vline_mode, aes(group = type, xintercept=as.numeric(mod
e)),col="#008DFC",size=1)+
  guides(color=FALSE)+
  geom_text(ann_text, mapping=aes(x=x, y=y, label=lab), colour =  c("#FC2C00
", "#008DFC") )

#plotting Histogram and density plot
plot(histp)

## Warning: Removed 6 rows containing missing values (geom_vline).
```

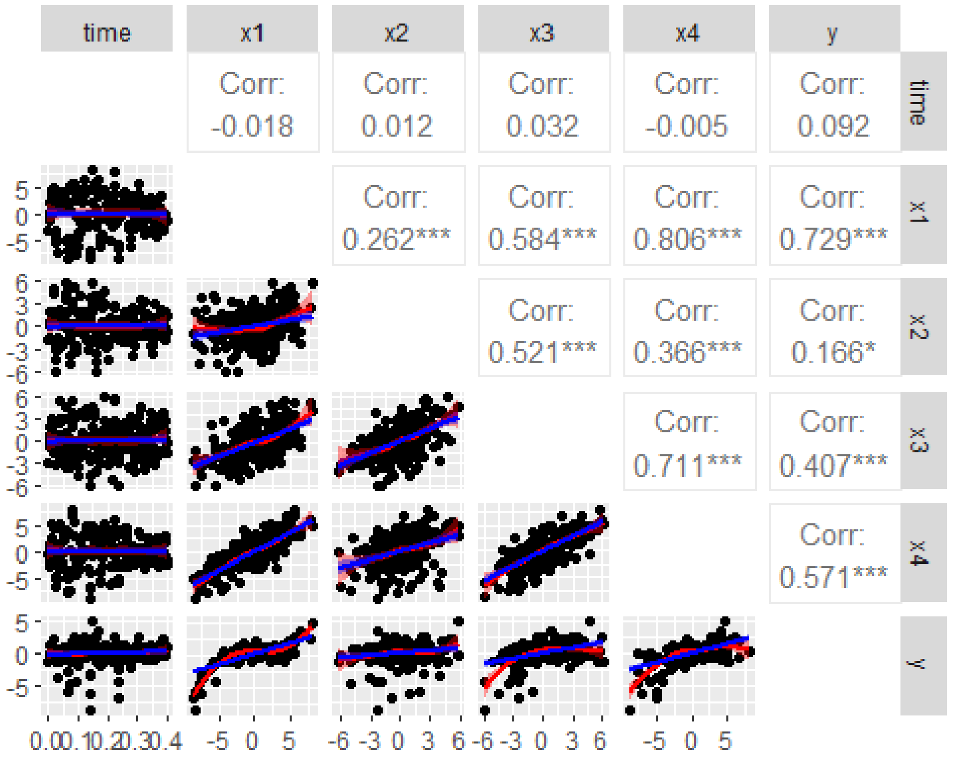


@param my_low_fn is a variable for lm and loess lines

```
my_low_fn <- function(data, mapping, ...){
  p <- ggplot(data = data, mapping = mapping) +
    geom_point() +
    geom_smooth(method=loess, fill="red", color="red", ...) +
    geom_smooth(method=lm, fill="blue", color="blue", ...)
  p
}
```

@param g is a variable for correlation and scatter plot

```
g <- ggpairs(df, lower = list(continuous = my_low_fn),  diag = list(continuou
s = "blankDiag"))

#plotting scatter and correlation plot
print(g)

## `geom_smooth()` using formula 'y ~ x'
## `geom_smooth()` using formula 'y ~ x'
## `geom_smooth()` using formula 'y ~ x'
## `geom_smooth()` using formula 'y ~ x'
## `geom_smooth()` using formula 'y ~ x'
## `geom_smooth()` using formula 'y ~ x'
## `geom_smooth()` using formula 'y ~ x'
## `geom_smooth()` using formula 'y ~ x'
## `geom_smooth()` using formula 'y ~ x'
## `geom_smooth()` using formula 'y ~ x'
## `geom_smooth()` using formula 'y ~ x'
## `geom_smooth()` using formula 'y ~ x'
## `geom_smooth()` using formula 'y ~ x'
## `geom_smooth()` using formula 'y ~ x'
## `geom_smooth()` using formula 'y ~ x'
## `geom_smooth()` using formula 'y ~ x'
## `geom_smooth()` using formula 'y ~ x'
## `geom_smooth()` using formula 'y ~ x'
## `geom_smooth()` using formula 'y ~ x'
## `geom_smooth()` using formula 'y ~ x'
## `geom_smooth()` using formula 'y ~ x'
## `geom_smooth()` using formula 'y ~ x'
## `geom_smooth()` using formula 'y ~ x'
## `geom_smooth()` using formula 'y ~ x'
## `geom_smooth()` using formula 'y ~ x'
## `geom_smooth()` using formula 'y ~ x'
## `geom_smooth()` using formula 'y ~ x'
## `geom_smooth()` using formula 'y ~ x'
## `geom_smooth()` using formula 'y ~ x'
## `geom_smooth()` using formula 'y ~ x'
```



@param data is a collection of variables @param y_index is the index of the variable to be predicted

```
lm_eqn <- function(data, y_index){
  for (i in 1:length(data[1,])) {
```

```r
    if (i == y_index){}
    else{
    y = data[,y_index]
    x = data[, i]
    m <- lm(y ~ x, data)
    print(colnames(data)[i])
    print(m)
    print(summary(m)$r.squared)
    }
  }
}

#print lm equation properties such as r squared and coefficiente values
lm_eqn(df, 6)
```

```
## [1] "time"
## 
## Call:
## lm(formula = y ~ x, data = data)
## 
## Coefficients:
## (Intercept)            x  
##     -0.1403       1.2439  
## 
## [1] 0.008454278
## [1] "x1"
## 
## Call:
## lm(formula = y ~ x, data = data)
## 
## Coefficients:
## (Intercept)            x  
##      0.1134       0.3206  
## 
## [1] 0.5312935
## [1] "x2"
## 
## Call:
## lm(formula = y ~ x, data = data)
## 
## Coefficients:
## (Intercept)            x  
##      0.1046       0.1166  
## 
## [1] 0.02768746
## [1] "x3"
## 
## Call:
## lm(formula = y ~ x, data = data)
## 
```

```
## Coefficients:
## (Intercept)            x
##     0.09955      0.26579
##
## [1] 0.1656291
## [1] "x4"
##
## Call:
## lm(formula = y ~ x, data = data)
##
## Coefficients:
## (Intercept)            x
##      0.1052       0.2838
##
## [1] 0.3254795
```

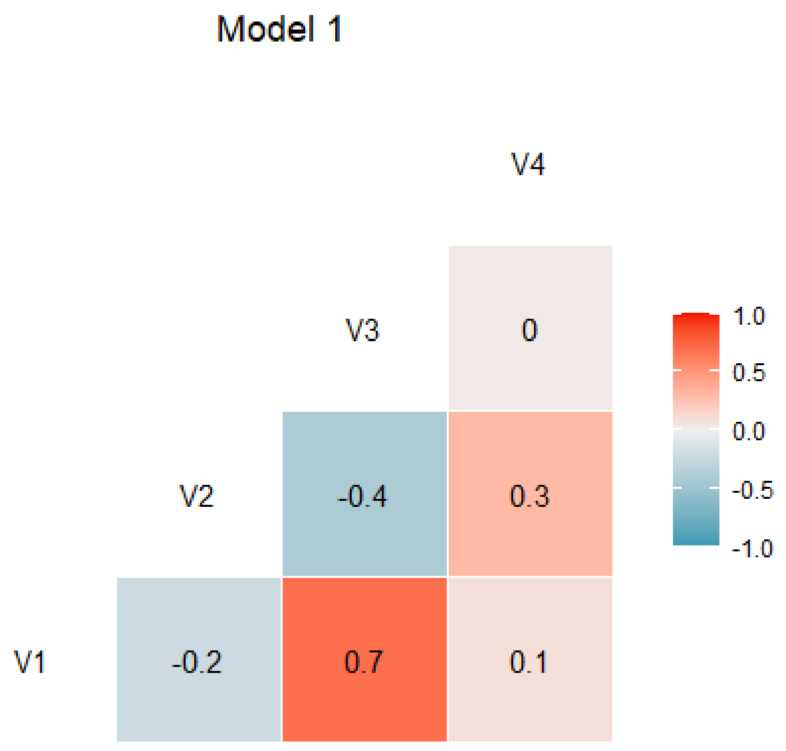
@param ones is a matrix of ones

```
ones = matrix(1 ,length(x[,1]) , 1)

#Model 1
model1X1 = matrix(input[,4])
model1Input = cbind(ones, model1X1)
model1X2 = matrix(input[,1]^2)
model1Input = cbind(model1Input, model1X2)
model1X3 = matrix(input[,1]^3)
model1Input = cbind(model1Input, model1X3)
model1X4 = matrix(input[,3]^4)
model1Input = cbind(model1Input, model1X4)

#model 2
model2X1 = matrix(input[,3]^3)
model2Input = cbind(ones, model2X1)
model2X2 = matrix(input[,3]^4)
model2Input = cbind(model2Input, model2X2)

#model 3
model3X1 = matrix(input[,2])
model3Input = cbind(ones, model3X1)
model3X2 = matrix(input[,1]^3)
model3Input = cbind(model3Input, model3X2)
model3X3 = matrix(input[,3]^4)
model3Input = cbind(model3Input, model3X3)

#model 4
model4X1 = matrix(input[,4])
model4Input = cbind(ones, model4X1)
model4X2 = matrix(input[,1]^3)
model4Input = cbind(model4Input, model4X2)
model4X3 = matrix(input[,3]^4)
```

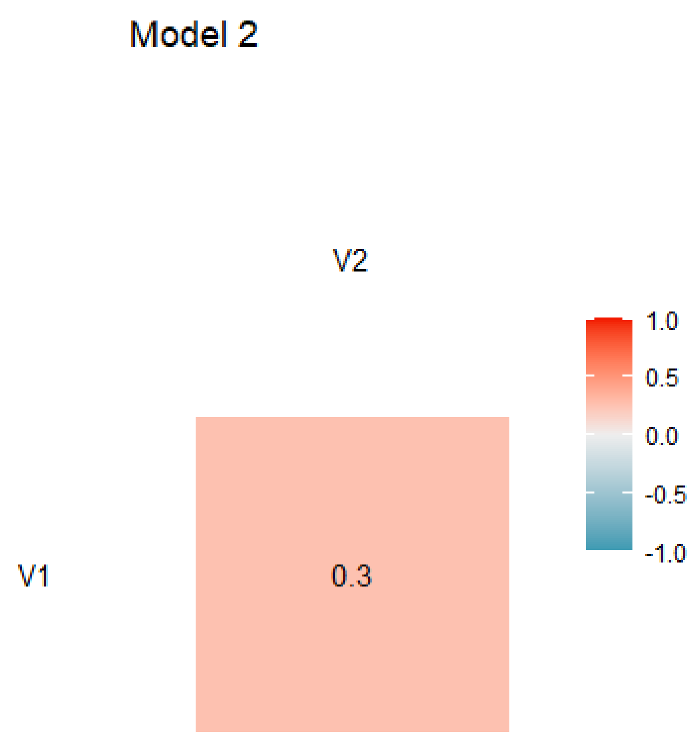
```
model4Input = cbind(model4Input, model4X3)

#model 5
model5X1 = matrix(input[,4])
model5Input = cbind(ones, model5X1)
model5X2 = matrix(input[,1]^2)
model5Input = cbind(model5Input, model5X2)
model5X3 = matrix(input[,1]^3)
model5Input = cbind(model5Input, model5X3)
model5X4 = matrix(input[,3]^4)
model5Input = cbind(model5Input, model5X4)
model5X5 = matrix(input[,1]^4)
model5Input = cbind(model5Input, model5X5)
```

@param dataTrain data of the input to train the model or used to compute the coefficient @param dataTest data used to predict the output @param outTrain data of the output to train the model or used to compute the coefficient @param outTest data of the output compare the predicted output to true output values @param String name of the model @return a list regression analysis related things, like coefficients, model evaluation criterion, residuals, plots

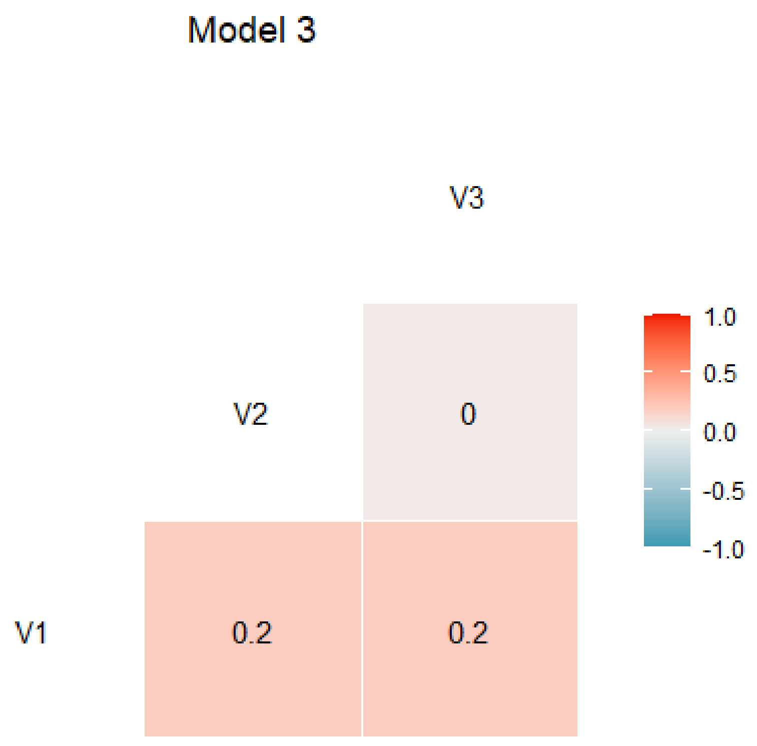
```
Taske2_myfn <- function(dataTrain, dataTest, outTrain, outTest, string){
  Theta_hat = solve(t(dataTrain)%*%dataTrain)%*%t(dataTrain)%*%outTrain
  y_hat = dataTest%*%Theta_hat
  residual = outTest - y_hat
  rss = norm(residual , type = "2")^2
  n = length(dataTest[,1])
  sigma_2 = rss/( n - 1 )
  log_like <- (-1)*(n/2)*log(2*pi) + (-1) * (n/2) * log(sigma_2) + (-1) * (1/
(2*sigma_2)) * rss
  AIC <- 2*length(Theta_hat) + (-1) * 2 * log_like
  BIC <- log(n)*length(Theta_hat) + (-1) * 2 * log_like
  summary <- summary(residual)
  residual = data.matrix(residual)
  skewness = skewness_fn(residual)
  kurtosis = myKurosis(residual)
  correlationPlot <- ggcorr(dataTrain[,c(2:length(dataTrain[1,]))], label = T
RUE)+
    ggplot2::labs(title = string)
  residual = data.frame(residual)
  qqplot <- ggplot(residual, aes(sample = residual)) +
    stat_qq() + stat_qq_line()+ theme_update(plot.title = element_text(hjust
= 0.5))+
    ggtitle( paste(string, "Normal Q-Q Plot", sep=" "))
  list("Theta_hat" = Theta_hat,
                  "y_hat" = y_hat,
                  "rss" = rss,
                  "sigma_2" = sigma_2,
                  "residual" = residual,
                  "skewness" = skewness,
                    "kurtosis" = kurtosis,
                    "log_like" = log_like,
                    "AIC" = AIC,
                    "BIC" = BIC,
                    "summary" = summary,
                    "correlationPlot" = correlationPlot,
                    "qqplot" = qqplot)
}

#computing task 2.1 to 2.6
Model1_Stats = Taske2_myfn(model1Input, model1Input, output, output, "Model 1
")
Model2_Stats = Taske2_myfn(model2Input, model2Input, output, output, "Model 2
")
Model3_Stats = Taske2_myfn(model3Input, model3Input, output, output, "Model 3
")
Model4_Stats = Taske2_myfn(model4Input, model4Input, output, output, "Model 4
")
Model5_Stats = Taske2_myfn(model5Input, model5Input, output, output, "Model 5
")

#plotting exploitative parameters correlation plot
plot(Model1_Stats$correlationPlot)
```



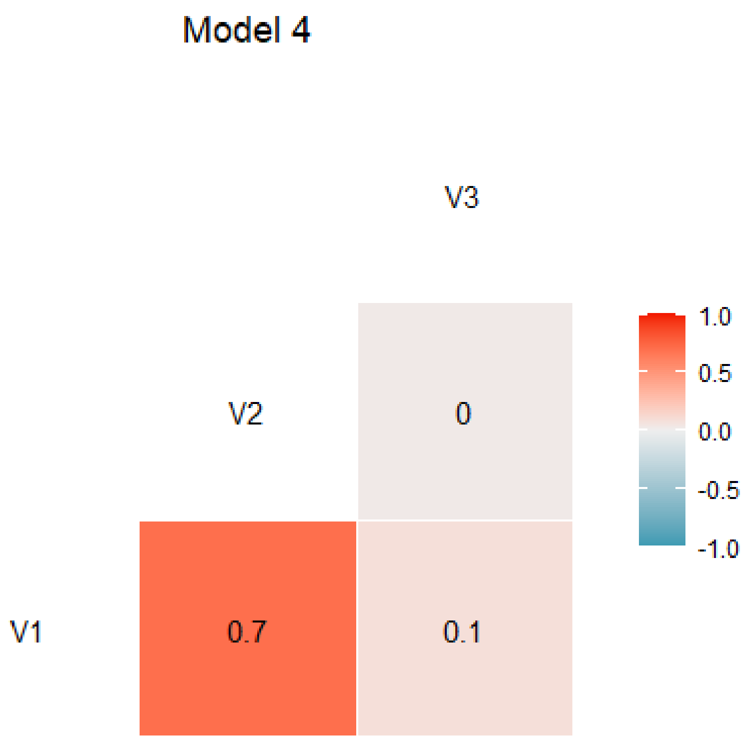
```
plot(Model2_Stats$correlationPlot)
```



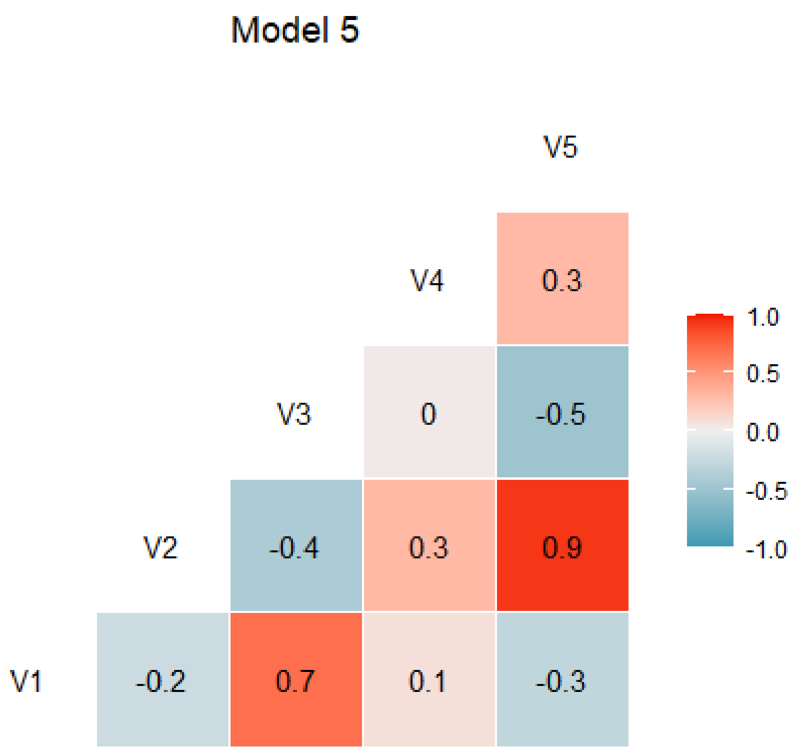
```
plot(Model3_Stats$correlationPlot)
```



```
plot(Model4_Stats$correlationPlot)
```



```
plot(Model5_Stats$correlationPlot)
```



@param data data to be printed

```
print_myfn <- function(data){
  for (i in 1:length(data[,1])) {
    print(data[i,])
  }
}

# printing the estimated coefficient
print_myfn(Model1_Stats$Theta_hat)
```

```
## [1] 0.4685517
## [1] -0.03410198
## [1] -0.001849575
## [1] 0.01038181
## [1] -0.001949154
```

```
print_myfn(Model2_Stats$Theta_hat)
```

```
## [1] 0.3035011
## [1] 0.01633468
## [1] -0.002713985
```

```
print_myfn(Model3_Stats$Theta_hat)
```

```
## [1] 0.4482995
## [1] 0.03810926
## [1] 0.009827804
## [1] -0.002092558
```

```
print_myfn(Model3_Stats$Theta_hat)
```

```
## [1] 0.4482995
## [1] 0.03810926
## [1] 0.009827804
## [1] -0.002092558
```

```
print_myfn(Model4_Stats$Theta_hat)
```

```
## [1] 0.4503292
## [1] -0.03488177
## [1] 0.01048218
## [1] -0.001990496
```

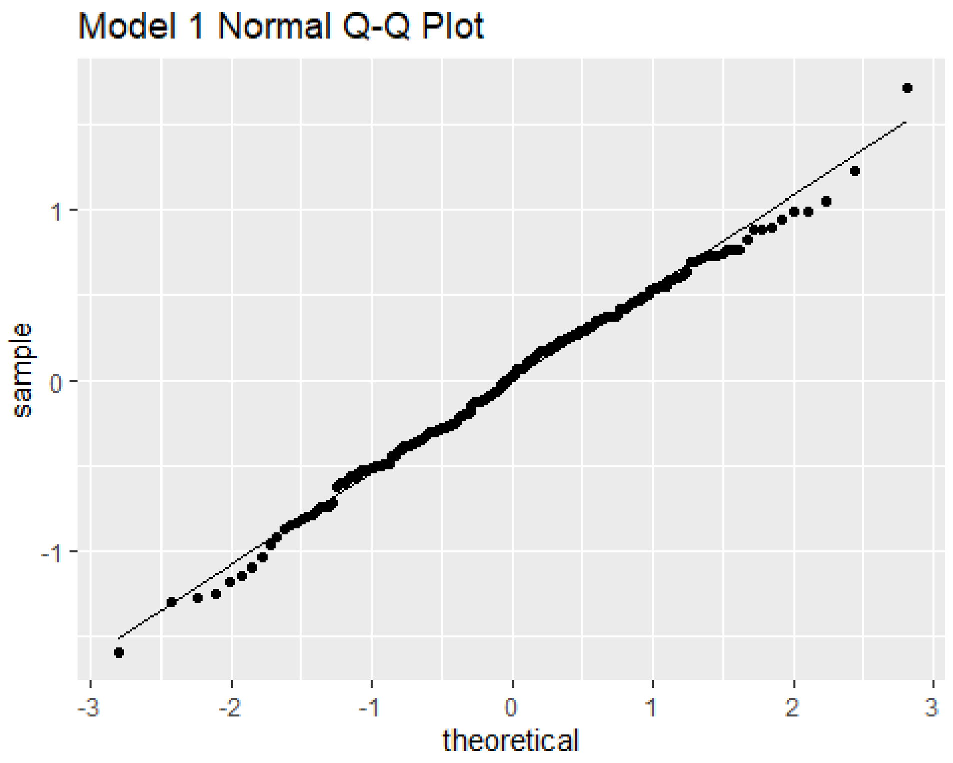
```
print_myfn(Model5_Stats$Theta_hat)
```

```
## [1] 0.460656
## [1] -0.03395313
## [1] -0.0002701644
## [1] 0.01034764
## [1] -0.001943452
## [1] -3.083194e-05
```

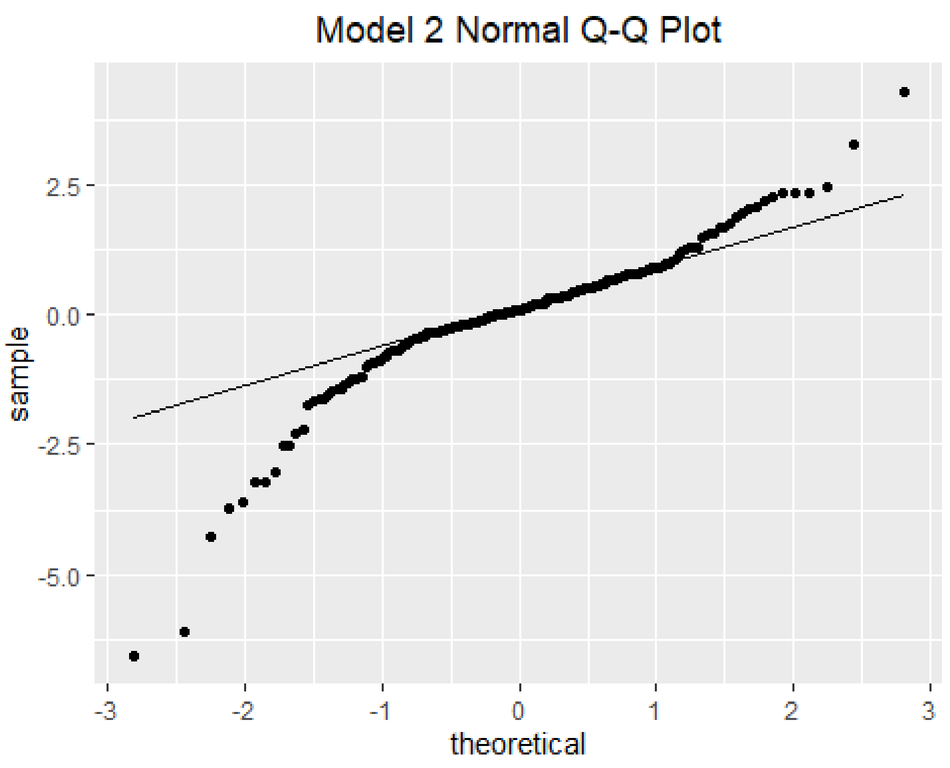
```
#printing Residual sum of squared
print(Model1_Stats$rss)
```

```
## [1] 57.32927

print(Model2_Stats$rss)

## [1] 351.4147

print(Model3_Stats$rss)

## [1] 57.32536

print(Model4_Stats$rss)

## [1] 57.46405

print(Model5_Stats$rss)

## [1] 57.30923

# printing Log_likelihood
print(Model1_Stats$log_like)

## [1] -159.1313

print(Model2_Stats$log_like)

## [1] -341.3534

print(Model3_Stats$log_like)

## [1] -159.1244

print(Model4_Stats$log_like)

## [1] -159.3673

print(Model5_Stats$log_like)

## [1] -159.0962

#printing AIC of the models
print(Model1_Stats$AIC)

## [1] 328.2626

print(Model2_Stats$AIC)

## [1] 688.7068

print(Model3_Stats$AIC)

## [1] 326.2489

print(Model4_Stats$AIC)

## [1] 326.7346
```

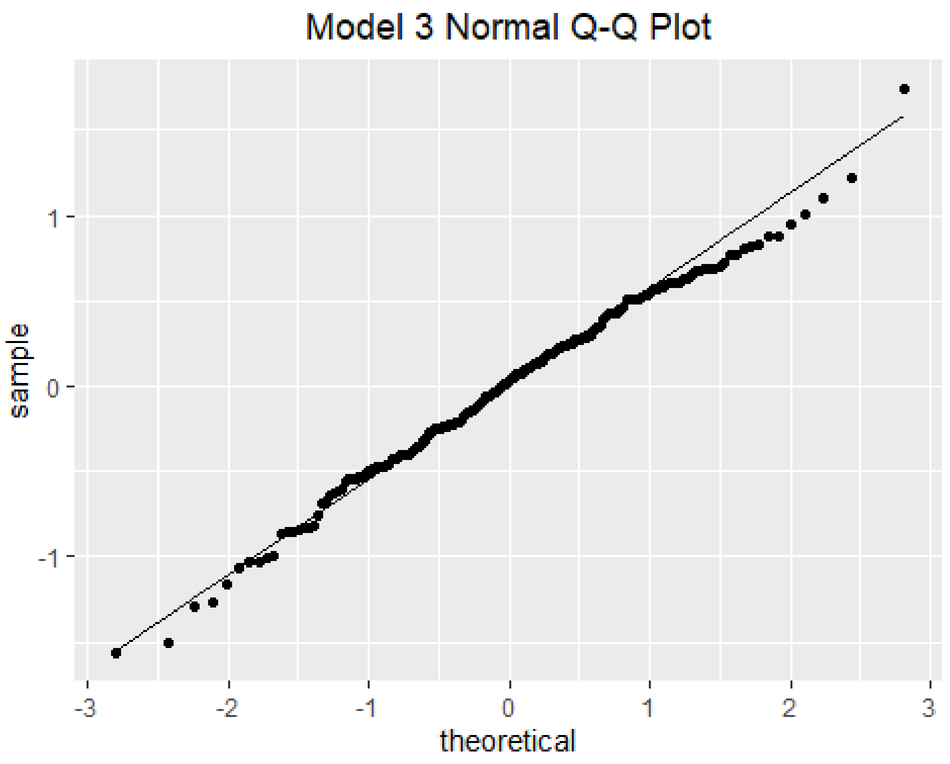
```
print(Model5_Stats$AIC)

## [1] 330.1923

#printing BIC of the models
print(Model1_Stats$BIC)

## [1] 344.7791

print(Model2_Stats$BIC)

## [1] 698.6168

print(Model3_Stats$BIC)

## [1] 339.4621

print(Model4_Stats$BIC)

## [1] 339.9478

print(Model5_Stats$BIC)

## [1] 350.0121

#printing qq plot
print(Model1_Stats$qqplot)
```



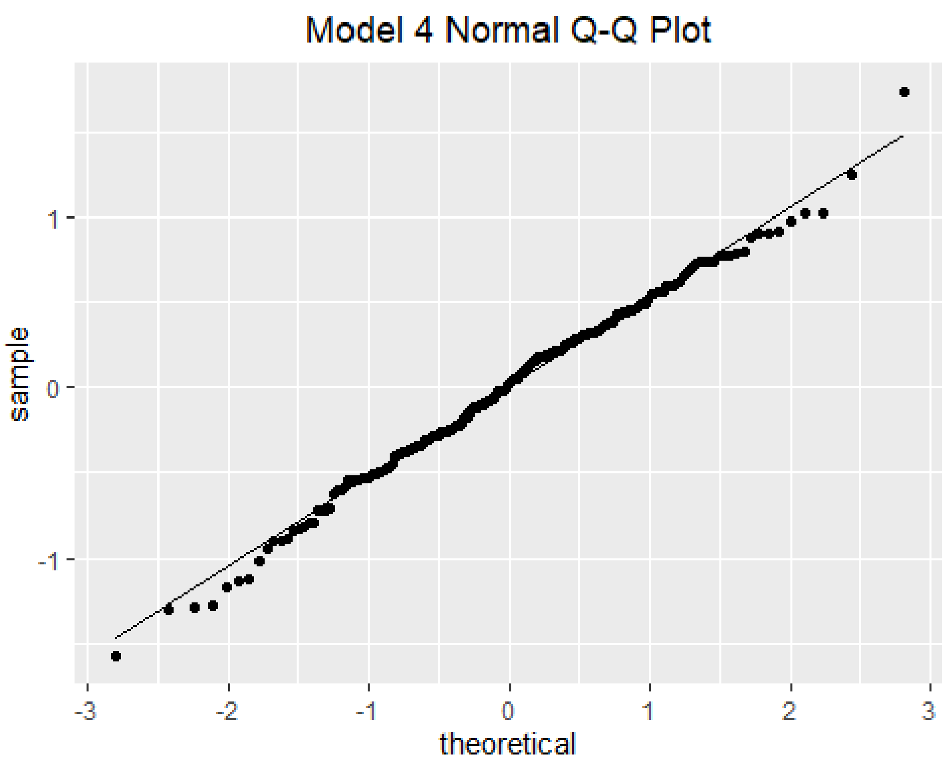
```
print(Model2_Stats$qqplot)
```



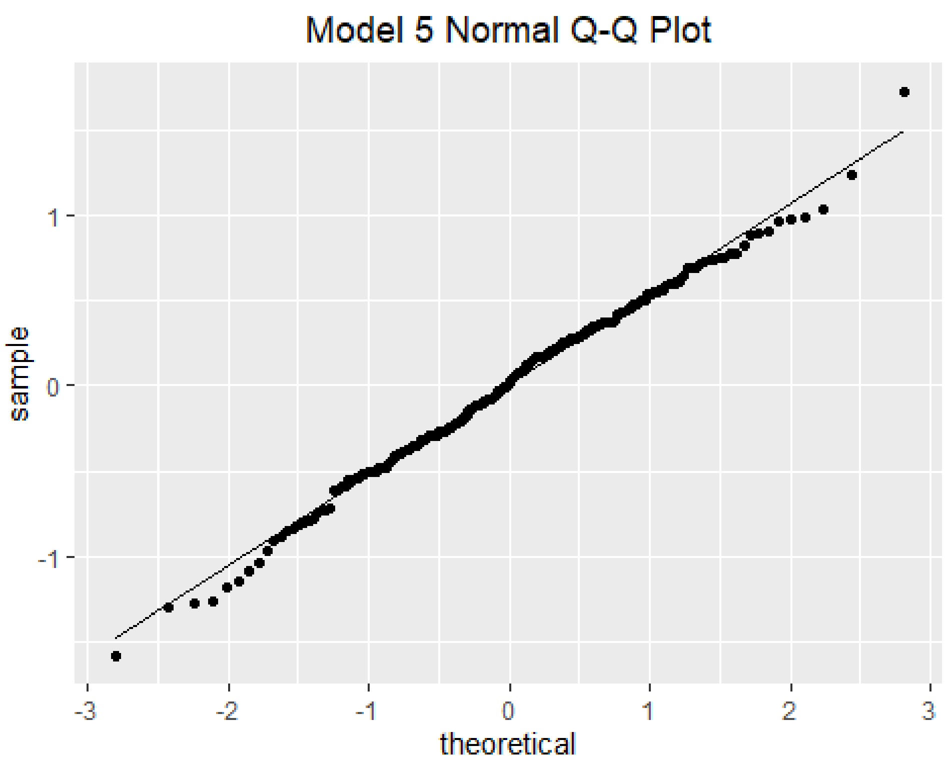
```
print(Model3_Stats$qqplot)
```



```
print(Model4_Stats$qqplot)
```



```
print(Model5_Stats$qqplot)
```



```
#printing kurtosis
print(Model1_Stats$kurtosis)

## [1] 3.109115

print(Model2_Stats$kurtosis)

## [1] 8.507902

print(Model3_Stats$kurtosis)

## [1] 3.225729

print(Model4_Stats$kurtosis)

## [1] 3.133473

print(Model5_Stats$kurtosis)

## [1] 3.127801

#printing skewness
print(Model1_Stats$skewness)

## [1] -0.151375

print(Model2_Stats$skewness)

## [1] -1.324471

print(Model3_Stats$skewness)

## [1] -0.1934425

print(Model4_Stats$skewness)

## [1] -0.1424218

print(Model5_Stats$skewness)

## [1] -0.150944

#printing residual mean
mean(Model1_Stats$residual$residual)

## [1] -8.624515e-16

mean(Model2_Stats$residual$residual)

## [1] -4.331048e-16

mean(Model3_Stats$residual$residual)

## [1] -1.623245e-16

mean(Model4_Stats$residual$residual)
```

```
## [1] 4.325433e-16

mean(Model5_Stats$residual$residual)

## [1] -3.916175e-15

#printing residual variance
var(Model1_Stats$residual$residual)

## [1] 0.2866463

var(Model2_Stats$residual$residual)

## [1] 1.757073

var(Model3_Stats$residual$residual)

## [1] 0.2866268

var(Model4_Stats$residual$residual)

## [1] 0.2873203

var(Model5_Stats$residual$residual)

## [1] 0.2865462

#for no serial correlation
bgtest(formula = lm(output ~ model1Input), order = 1)

##
##  Breusch-Godfrey test for serial correlation of order up to 1
##
## data:  lm(output ~ model1Input)
## LM test = 0.32833, df = 1, p-value = 0.5666

bgtest(formula = lm(output ~ model2Input), order = 1)

##
##  Breusch-Godfrey test for serial correlation of order up to 1
##
## data:  lm(output ~ model2Input)
## LM test = 14.488, df = 1, p-value = 0.0001411

bgtest(formula = lm(output ~ model3Input), order = 1)

##
##  Breusch-Godfrey test for serial correlation of order up to 1
##
## data:  lm(output ~ model3Input)
## LM test = 0.1819, df = 1, p-value = 0.6697

bgtest(formula = lm(output ~ model4Input), order = 1)
```

```
##
##  Breusch-Godfrey test for serial correlation of order up to 1
##
## data:  lm(output ~ model4Input)
## LM test = 0.33061, df = 1, p-value = 0.5653
```

```
bgtest(formula = lm(output ~ model5Input), order = 1)
```

```
##
##  Breusch-Godfrey test for serial correlation of order up to 1
##
## data:  lm(output ~ model5Input)
## LM test = 0.31693, df = 1, p-value = 0.5735
```

@param num_rows_Train number of rows for train data

```
num_rows_Train = as.integer(length(model3Input[,1]) * 0.7);
```

@param Best_Model3_Stats Model for tast 2.7

```
Best_Model3_Stats = Taske2_myfn(model3Input[(1:num_rows_Train),], model3Input
[-(1:num_rows_Train),],
                                output[(1:num_rows_Train),], output[-(1:num_r
ows_Train),], "Best Model 3")
```

@param data data for which the confidence interval are based on @param sigma_2 of the model @param ci z-score of confidente interval @return confidence interval

```
task_2_7_myfn <- function(data, sigma_2,  ci){
  n = length(data[,1])
  number_of_parameters = length(data[1,])
  cov_thetaHat = sigma_2 * (solve(t(data) %*% data))
  var_y_hat = matrix(0 , n , 1)
  for( i in 1:n){
    X_i = matrix( data[i,] , 1 , number_of_parameters )
    # X[i,] creates a vector. Convert it to matrix
    var_y_hat[i,1] = X_i %*% cov_thetaHat %*% t(X_i) # same as sigma_2 * ( X_
i %*% ( solve(t(X) %*% X)  ) %*% t(X_i) )
  }
  CI = ci * sqrt(var_y_hat) # Confidance interval
}
```

@param bestModel_CI confidence interval for predicted values of model in 2.7

```
bestModel_CI = task_2_7_myfn( model3Input[-(1:num_rows_Train),], Best_Model3_
Stats$sigma_2, 1.96)

#print the mean of confidence intervals
mean(bestModel_CI)

## [1] 0.2656766
```

@param DataCI_x1 data frame of x1 input, output and predicted output

```
DataCI_x1 = data.frame("x1" = df$x1[-(1:num_rows_Train)],
                    "y" = Best_Model3_Stats$y_hat,
                    "True_y" = output[-(1:num_rows_Train),])
```

@param DataCI_x2 data frame of x2 input, output and predicted output

```
DataCI_x2 = data.frame("x2" = df$x2[-(1:num_rows_Train)],
                       "y" = Best_Model3_Stats$y_hat,
                       "True_y" = output[-(1:num_rows_Train),])
```

@param DataCI_x1 data frame of x3 input, output and predicted output

```
DataCI_x3 = data.frame("x3" = df$x3[-(1:num_rows_Train)],
                       "y" = Best_Model3_Stats$y_hat,
                       "True_y" = output[-(1:num_rows_Train),])
```

@param DataCI_x1 data frame of x4 input, output and predicted output

```
DataCI_x4 = data.frame("x4" = df$x4[-(1:num_rows_Train)],
                       "y" = Best_Model3_Stats$y_hat,
                       "True_y" = output[-(1:num_rows_Train),])
```

@param dodge variable for dodging overlapping

```
dodge <- position_dodge(width=0.0005)
```

@param gx1 plot of confidence intervals of predicted output with predicted output data point connected with a red line x1 input in x axis and data point of the outp ut

```
gx1 <- ggplot(data=DataCI_x1, aes(x=x1, y=y,ymin = y-bestModel_CI, ymax =y+be
stModel_CI)) +
                        geom_point(aes(y=True_y), size = 2, position=dodge, c
olour = "blue") +
                        geom_line(aes(y=y), size = 0.8, position=dodge, colou
r = "red") +
                        geom_point(aes(y=y), size = 1.6, position=dodge, colo
ur = "red") +
                        geom_errorbar(position=dodge)
```

@param gx2 plot of confidence intervals of predicted output with predicted o utput data point connected with a red line x2 input in x axis and data point of the output

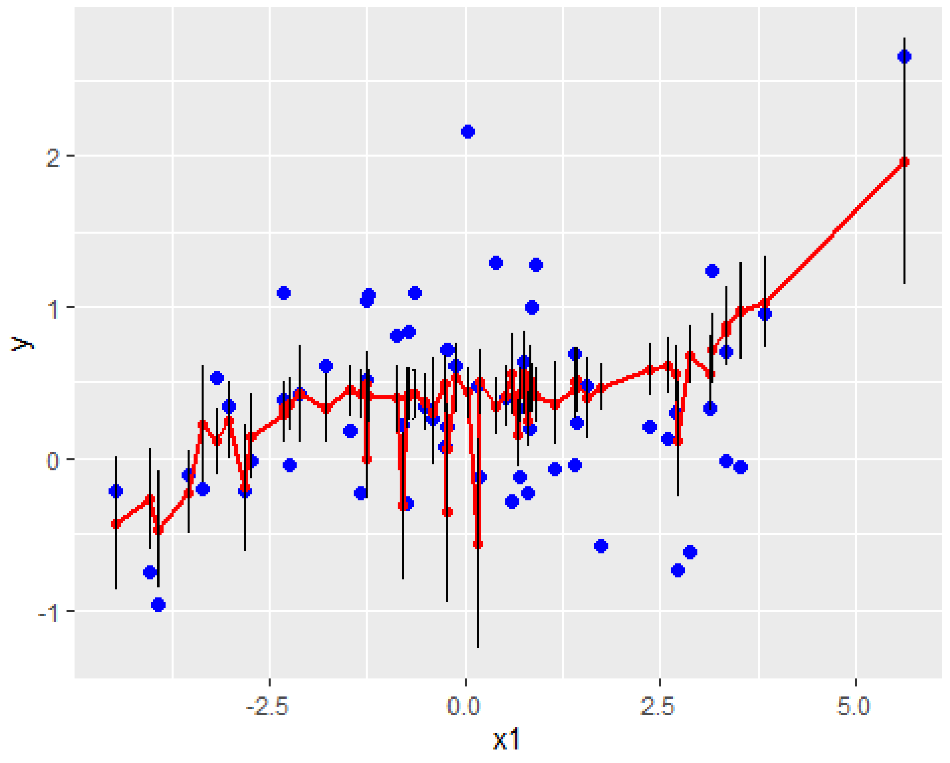
```
gx2 <- ggplot(data=DataCI_x2, aes(x=x2, y=y,ymin = y-bestModel_CI, ymax =y+be
stModel_CI)) +
                        geom_point(aes(y=True_y), size = 2, position=dodge, c
olour = "blue") +
                        geom_line(aes(y=y), size = 0.8, position=dodge, colou
r = "red") +
                        geom_point(aes(y=y), size = 1.6, position=dodge, colo
ur = "red") +
                        geom_errorbar(position=dodge)
```

@param gx3 plot of confidence intervals of predicted output with predicted output data point connected with a red line x3 input in x axis and data point of the output

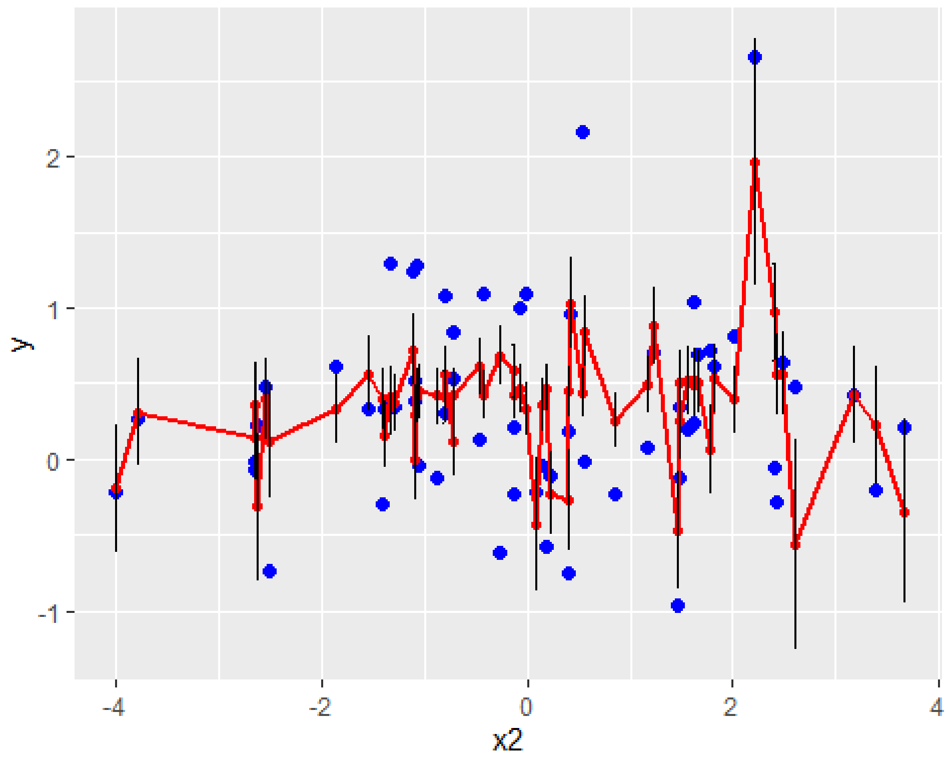
```
gx3 <- ggplot(data=DataCI_x3, aes(x=x3, y=y,ymin = y-bestModel_CI, ymax =y+be
stModel_CI)) +
                        geom_point(aes(y=True_y), size = 2, position=dodge, c
olour = "blue") +
                        geom_line(aes(y=y), size = 0.8, position=dodge, colou
r = "red") +
                        geom_point(aes(y=y), size = 1.6, position=dodge, colo
ur = "red") +
                        geom_errorbar(position=dodge)
```

@param gx4 plot of confidence intervals of predicted output with predicted output data point connected with a red line x4 input in x axis and data point of the output

```
gx4 <- ggplot(data=DataCI_x4, aes(x=x4, y=y,ymin = y-bestModel_CI, ymax =y+be
stModel_CI)) +
                        geom_point(aes(y=True_y), size = 2, position=dodge, c
olour = "blue") +
                        geom_line(aes(y=y), size = 0.8, position=dodge, colou
r = "red") +
                        geom_point(aes(y=y), size = 1.6, position=dodge, colo
ur = "red") +
                        geom_errorbar(position=dodge)

#plotting plot of confidence intervals of predicted output with predicted out
put data point connected with a red line inputs in x axis and data point of t
he output
plot(gx1)
```

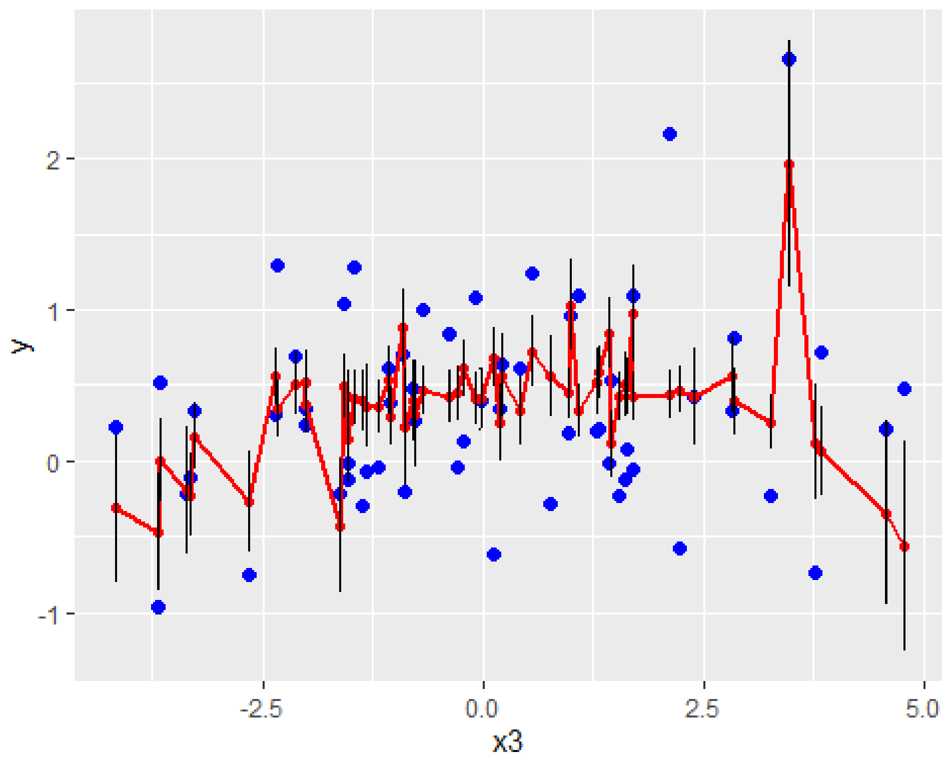


```
plot(gx2)
```

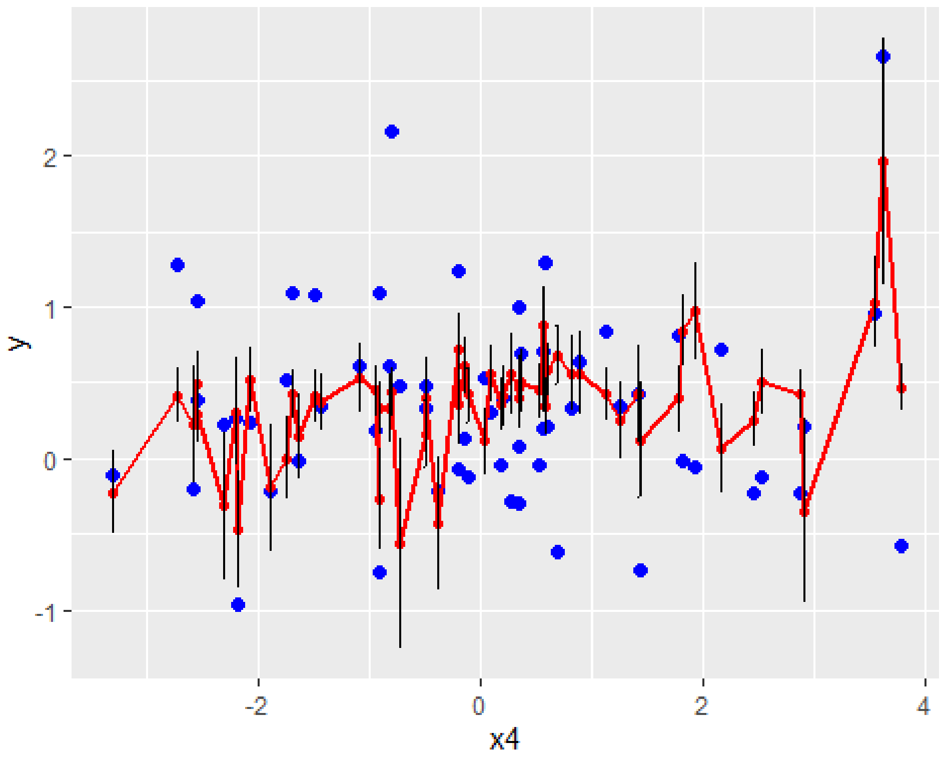


```
plot(gx3)
```

```
## Warning: position_dodge requires non-overlapping x intervals

## Warning: position_dodge requires non-overlapping x intervals

## Warning: position_dodge requires non-overlapping x intervals
```



```
plot(gx4)
```



```
#printing r squared of task 2.7
cor(df$x1, output)^2

##           [,1]
## [1,] 0.5312935
```

@param best_T3_model model for this task

```
best_T3_model = model3Input
```

@param data data for which standard deviation * a given z-score @param sigma_2 of the model @param ci given z score @return standard deviation * a given z-score

```
theta_var_myfn <- function(dataTest, sigma_2, ci){
  n = length(dataTest[1,])
  cov_thetaHat = sigma_2 * (solve(t(dataTest) %*% dataTest))
  CI = matrix(0 , n , 1)
  for( i in 1:n){
   CI[i,1]  = ci * sqrt(cov_thetaHat[i,i]) # Confidence interval
  }
  CI
}
```

@param T3_CI standard deviation * a given z-score of the coefficients

```
T3_cI = theta_var_myfn(model3Input, Model3_Stats$sigma_2, 1.96)
```

@param n_loop number of values to be generated

```r
n_loop = 100000
```

@param n number of samples @param min minimal of the range @param max the maximal of the range @return values uniformly distributed throughout the range

```r
myrunif <- function(n, min=0, max=1) {
  min + (sample(.Machine$integer.max, n) - 1) / (.Machine$integer.max - 1) *
    (max - min)
}
```

@param Param_1 matrix of generated values of Theta_bias

```r
Param_1 = matrix(myrunif(n=n_loop, min=Model3_Stats$Theta_hat[1,]-T3_cI[1,],m
ax=Model3_Stats$Theta_hat[1,]+T3_cI[1,]))
```

@param Param_2 matrix of generated values of Theta_1

```r
Param_2 = matrix(myrunif(n=n_loop, min=Model3_Stats$Theta_hat[2,]-T3_cI[2,],m
ax=Model3_Stats$Theta_hat[2,]+T3_cI[2,]))

Param_ = cbind(Param_1, Param_2)
```

@param Param_ matrix of the combined values of Theta_bias and theta_1

```r
Param_ = cbind(Param_ , matrix(0 ,length(Param_[,1]) , 1))

#rejection ABC
for ( i in 1:n_loop) {
  matrix_ = rbind(matrix(Param_[i,c(1:2)]), Model3_Stats$Theta_hat[3,])
  matrix_ = rbind(matrix_, Model3_Stats$Theta_hat[4,])

  out = best_T3_model%*%matrix_
  sse_ = sum( (output - out)^2)
  if(abs(sse_ - Model3_Stats$rss) <= Model3_Stats$rss/201){
      Param_[i,3] =  1
  }
}
```

@param postior_Theta data frame with accepted and rejected coefficients

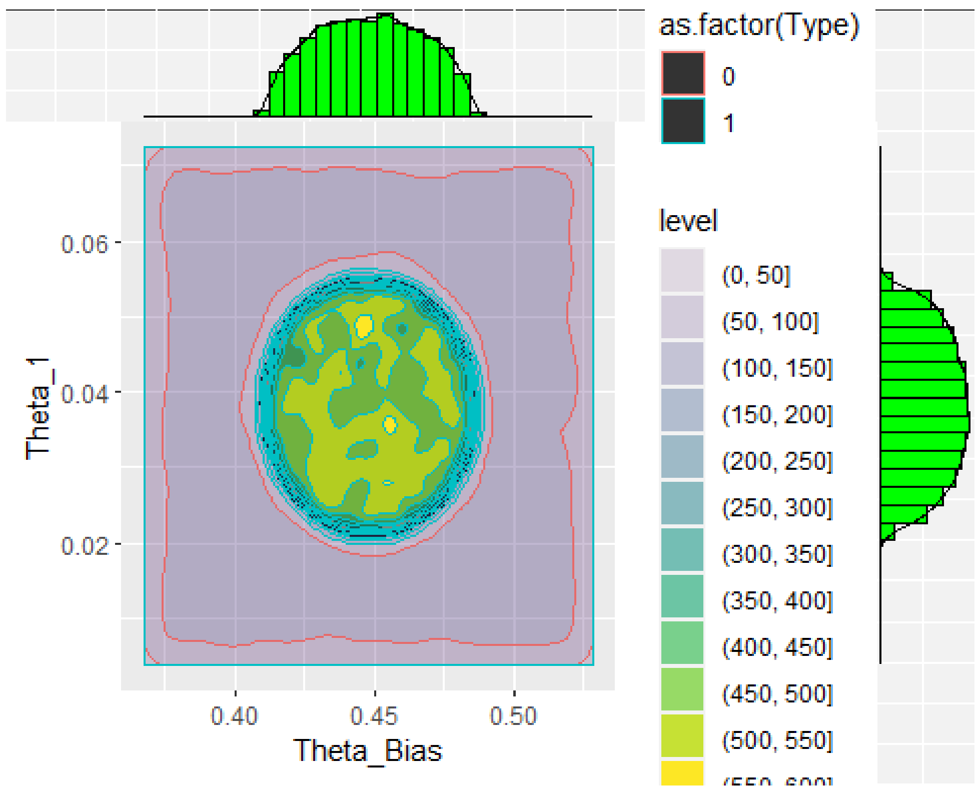
```r
postior_Theta = data.frame("Theta_Bias" = Param_[,1],
                           "Theta_1" = Param_[,2],
                           "Type" = Param_[,3])
```

@param v plot of density for accepted coefficient and rejected ones

```r
v <- ggplot(postior_Theta, aes(x=Theta_Bias, y=Theta_1, group=Type))+
  geom_point(data = postior_Theta[postior_Theta$Type == 1, ]) +
  stat_density_2d_filled(  mapping= aes(alpha = ..level.., color = as.factor
(Type)))
```

@param p plot of density for accepted coefficient and rejected ones and marginal plot for accepted coefficient

```
p <- ggMarginal(v, type= c("densigram"), colour="black", fill="green")

#plotting the marginal and joint plot
plot(p)
```



@param aplha_Theta_bias Alpha of Theta_bias of beta distribution

```
aplha_Theta_bias = 1 + length(postior_Theta[postior_Theta$Type == 1, ]$Theta_
Bias)/
  length(postior_Theta[postior_Theta$Theta_Bias <= max(postior_Theta[postior_
Theta$Type == 1, ]$Theta_Bias) &
                       postior_Theta$Theta_Bias >= min(postior_Theta[postior_
Theta$Type == 1, ]$Theta_Bias), ]$Theta_Bias)
print(aplha_Theta_bias)

## [1] 1.387522
```

@param aplha_Theta_1 Alpha of Theta_1 of beta distribution

```
aplha_Theta_1 = 1 + length(postior_Theta[postior_Theta$Type == 1, ]$Theta_1)/
  length(postior_Theta[postior_Theta$Theta_1 <= max(postior_Theta[postior_The
ta$Type == 1, ]$Theta_1) &
                       postior_Theta$Theta_1 >= min(postior_Theta[postior_The
ta$Type == 1, ]$Theta_1), ]$Theta_1)
print(aplha_Theta_1)
```

```
## [1] 1.37097
```

@param join_prob Alpha of Theta_1 of beta distribution

```
join_prob = length(postior_Theta[postior_Theta$Type == 1, ])/n_loop
print(join_prob)

## [1] 3e-05

# percentage of theta in the marginal range being accepted
marginal_Theta_1 = aplha_Theta_1 - 1
print(marginal_Theta_1)

## [1] 0.3709701

marginal_Theta_bias = aplha_Theta_bias - 1
print(marginal_Theta_bias)

## [1] 0.3875218

#expected value
mean(postior_Theta[postior_Theta$Type == 1, ]$Theta_1)

## [1] 0.0380964

mean(postior_Theta[postior_Theta$Type == 1, ]$Theta_Bias)

## [1] 0.4483218
```