Coventry University

Softwarica College of IT & E-commerce

**Module Code & Module Title**

**STW7089CEM Introduction to Statistical Methods for Data Science**

**Assessment Weightage & Type**

**100% Individual Coursework**

**Year and Semester**

**2023-24 Spring**

**Students Name: Ayush Rayamajhi**

**Coventry ID: 15368843**

**Assignment Due Date: June 30, 2024**

**Assignment Submission Date: June 30, 2024**

# Acknowledgement

I am deeply grateful to Mr. Dhurba Adhikari and Mr. Shrawan Thakur, as well as Coventry University, for granting me the opportunity to undertake this coursework and for their constant support and guidance, which were crucial to the successful completion of my project. My thanks also go to our college Softwarica College of IT & E-commerce for providing a supportive workspace and to the module teachers for their invaluable help throughout this course. Tackling and completing the assignment posed significant challenges for me, and I wouldn't have succeeded without the assistance of my friends and tutors. Therefore, I would like to express my appreciation to my teachers for providing all the necessary guidelines.

# Abstract

The primary objective of this assignment is to identify the optimal regression model from a set of nonlinear regression models to accurately describe the relationship among several simulated gene expression time-series data. Gene expression is a vital biological process where genetic information is utilized to synthesize functional products like proteins

The provided data consists of time-series expressions of five genes (x1, x2, x3, x4, x5) subject to additive Gaussian noise, available in three formats: CSV, XLSX, and R Data. The initial task involves preliminary data analysis, including time series plots, distribution analysis, and correlation and scatter plots to examine dependencies among the genes. The subsequent task focuses on regression modeling to elucidate the relationship between the output gene x2 and the other input genes (x1, x3, x4, x5). The goal is to determine the most suitable polynomial regression model from five candidate models to accurately represent this gene regulation relationship. This analysis aims to enhance the understanding of gene interactions and contribute to the broader field of genetic research.

# Contents

# Table of Figures

# List of Tables

# Introduction

Gene expression is a fundamental biological process that involves the transformation of genetic information encoded in a gene into a functional product. Understanding the regulatory mechanisms between genes is crucial for understanding complex diseases and cellular responses to various environmental stimulants. (Bruce Alberts, 2022)

Our goal in this assignment is to examine and model the relationships between a set of simulated gene expression and the set of time-series data. The given dataset consists expression levels of five genes identified as x1, x2, x3, x4 and x5 which are recorded at various point of time, with additive noise applied to each gene expression. Our objective is to determine which nonlinear polynomial regression model best fits the data that describes the relationship between the expression of gene x2 (considered as the output gene) and the other four genes x1, x3, x4, and x5 (considered as input genes regulating x2).

The analysis of the give data is split into two main tasks:

- **Preliminary Data Analysis**
- **Regression- Modelling Gene Expression Relationship**

The tasks outlined will be elaborated upon in the report with their respective sections.

# Data Cleaning

An excel file named “gene_data.csv.xlsv” was provided with the assignment which contained the expression levels of five genes identified as x1, x2, x3, x4 and x5 and the time it was recorded on. It consisted of six tables named “Table 1”, “Table 4”, “Table 7”, “Table 9”, “Table 11” and “Table 13”. All the information about the genes and its recorded time were split in these six tables.

Tables 4, 7, 9, 11, and 13 had the Time and value of x1 merged into a single column, which needed to be split into their respective columns. Additionally, the column name "Time (sec)" was simplified to "Time" for better accessibility while coding. Once all the data was cleaned, it was merged into a single table and file extension for the given file ““gene_data.csv.xlsv”” was also changed into ““gene_data.csv” for easier access and coding efficiency. Considering the minimal cleaning requirements for this particular dataset, manual cleaning was the most effective approach. Utilizing data cleaning packages would have been unnecessary.

# TASK 1: Preliminary Data Analysis

In this phase, preliminary analysis of the given gene expression will be conducted. As a part of preliminary analysis of the data, Time series plot, Distribution analysis using Density plot and Distribution plot and Correlation and Scatter Plots between different combination of genes will be conducted.

## Time Series Plot

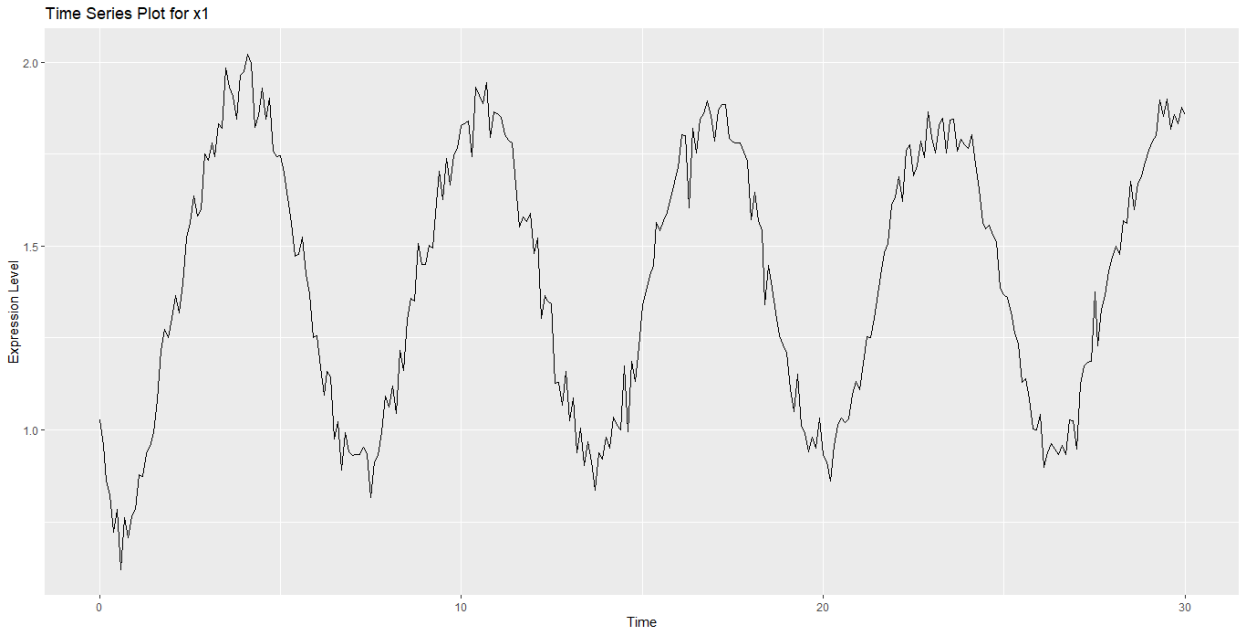
A time series plot is a graphical representation of data points in a time series which helps us to visualize the data in Two-Dimensional Plane i.e., x-axis and y-axis. where time is plotted in x-axis and observed data points are plotted in y-axis. (Rob J Hyndman, April 2018)

On the Time Series Plots given below, x-axis will be denoting the time interval with the increment of 10 whereas, y-axis denote the expression level of the genes.

## Time Series Plot for Gene x1

The time series plot for gene x1 depicts a spike in trend which reach its peak expression level on around 5 seconds time interval with gradual decrease on trend for around 3 to 4 seconds before increasing again. This trend is followed over the observed timeframe. This suggests a general increase and decrease in the gene x1, however its pattern over the course of the study is stable.

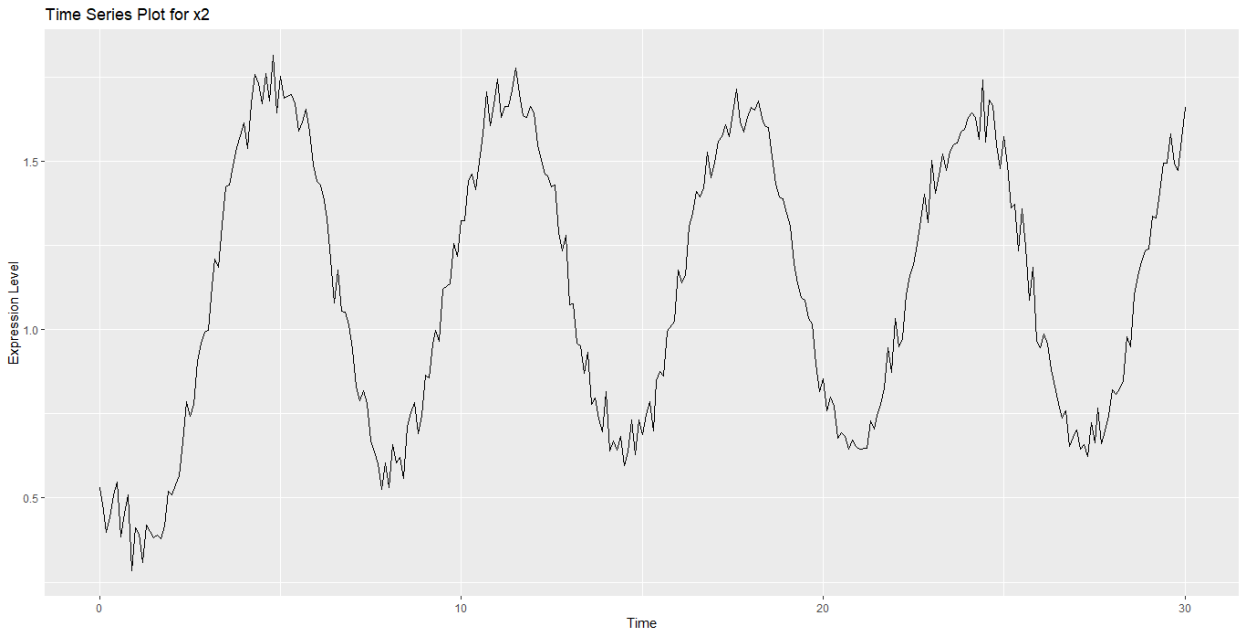
*Figure 1: Time Series Plot for x1*



## Time Series Plot for Gene x2

The time series plot for gene x2 depicts a spike in trend which reach its peak expression level on around 5 seconds time interval with gradual decrease on trend which takes to reach its lowest point at around 3 to 4 seconds before increasing again. This trend is followed over the observed timeframe. This suggests a general increase and decrease in the gene x2, however its pattern over the course of the study is stable.

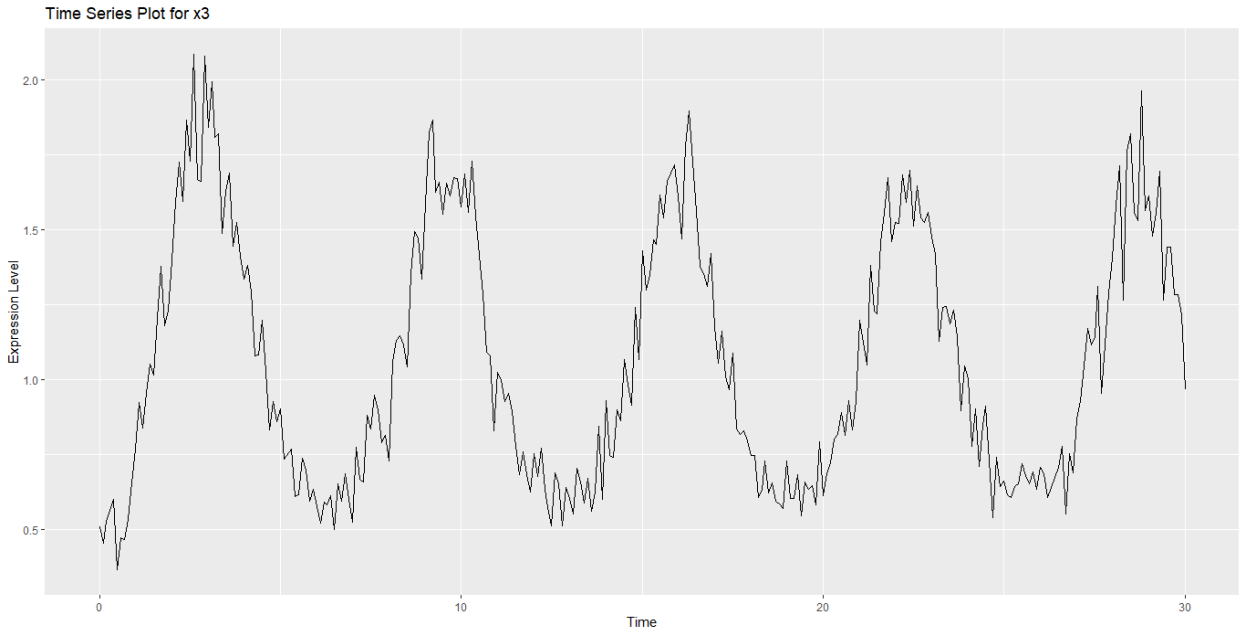
*Figure 2: Time Series Plot for x2*



## Time Series Plot for Gene x3

The time series plot for gene x3 depicts a spike in trend which reach its peak expression level on around 3 seconds time interval with gradual decrease on trend which takes to reach its lowest point at around 4 to 5 seconds before increasing again. This trend is followed over the observed timeframe. This suggests a general increase and decrease in the gene x3, however its pattern over the course of the study shows that its peak in expression level is slightly decreasing while its lowest points seem generally stable.

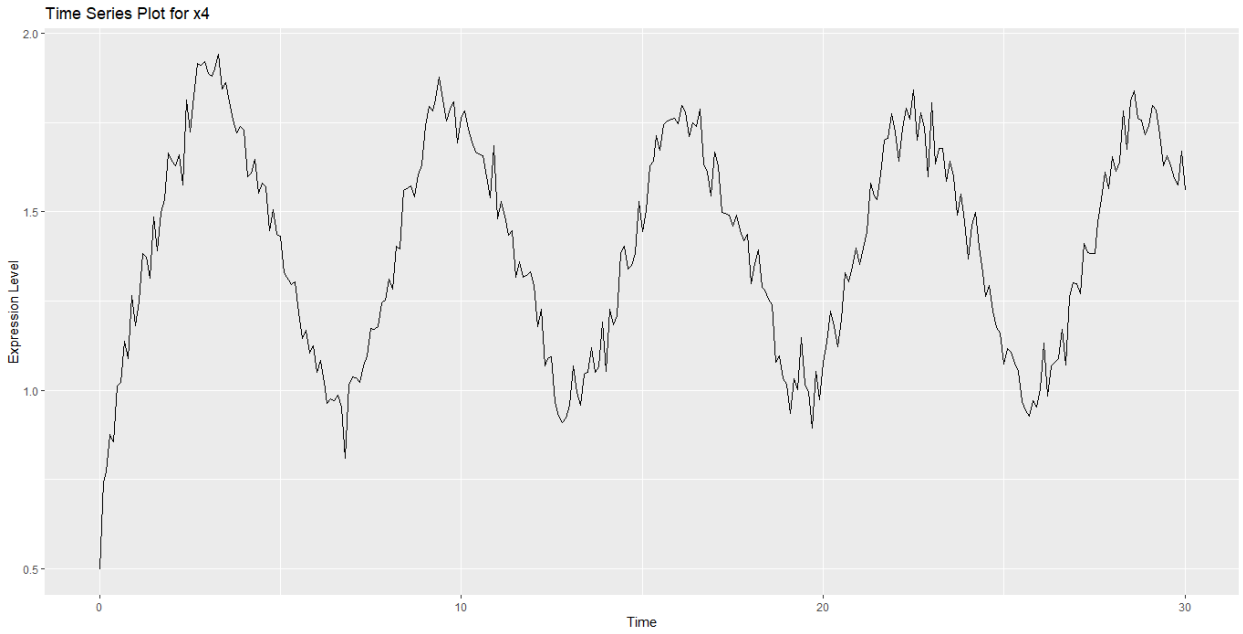
*Figure 3: Time Series Plot for x3*



## Time Series Plot for Gene x4

The time series plot for gene x4 depicts an sudden spike in trend which reach its peak at interval of 3 seconds followed by sudden downfall in expression level which reach its lowest point in interval of around 3 seconds. This trend is followed over the observed timeframe. This suggests a general increase and decrease in the gene x4, however its pattern over the course of the study shows that its peak in expression level is slightly decreasing while its lowest points seem generally stable.

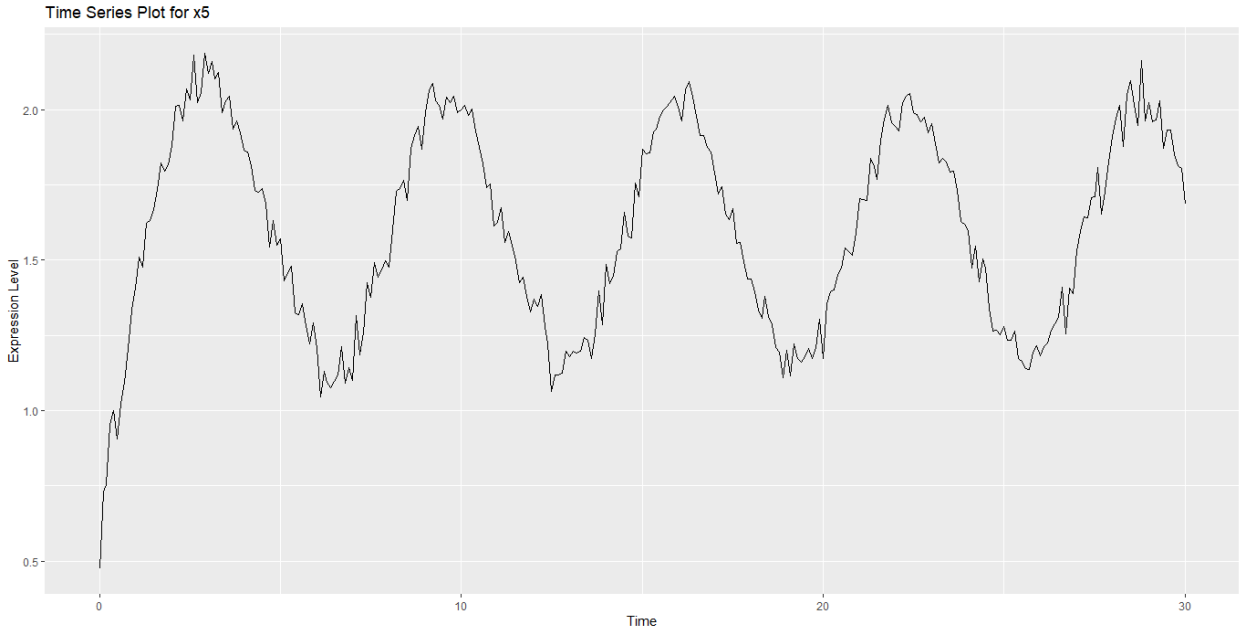
*Figure 4: Time Series Plot for x4*



## Time Series Plot for Gene x5

The time series plot for gene x5 depicts a sudden spike in trend which reach its peak at interval of 3 seconds followed by gradual downfall in expression level which reach its lowest point in interval of around 3 seconds. This trend is followed over the observed timeframe. However, after the first downfall, there is gradual increase and decrease in the trend of expression level. This pattern observed throughout the course of the study is stable.

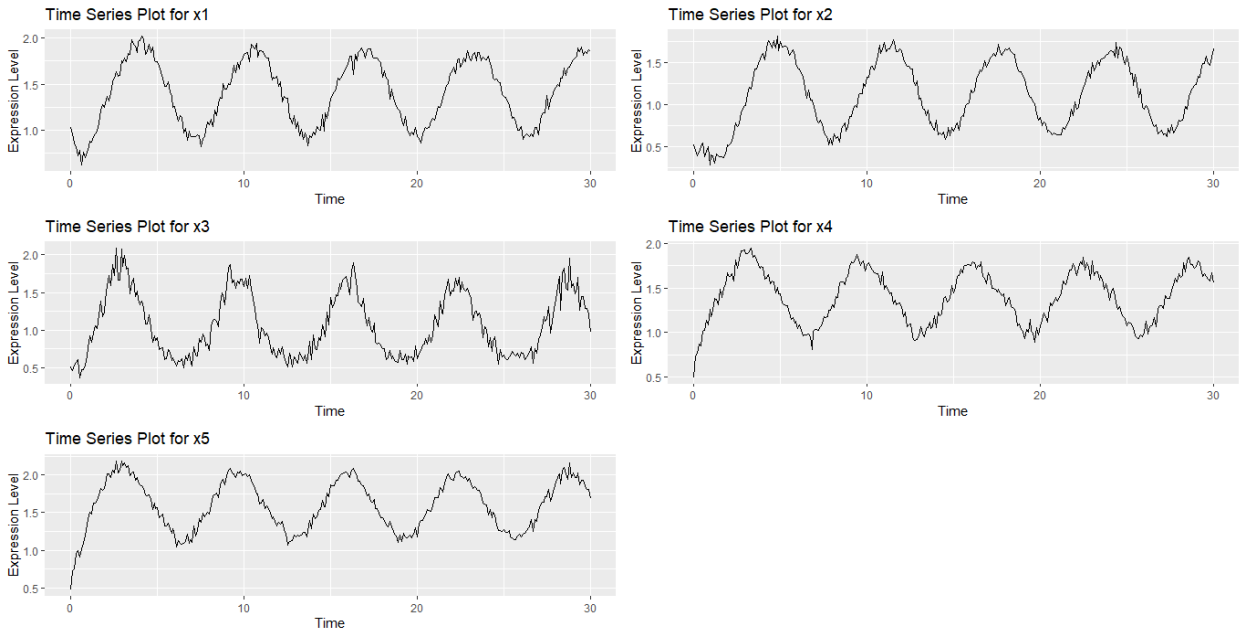
*Figure 5: Time Series Plot for x5*



## Combined Time Series Plot for all the Gene Expression

The below image represents the time series plot of all the genes with respect to time which we discussed above in detail.

*Figure 6: Time Series Plot for all the genes*



## Distribution of each Genes

To measure and provide graphical representation of the distribution of each gene, two major plots are used, density plot and Distribution plots with density curve. Each of these plots will be elaborate on their respective sections.

## Density Plot

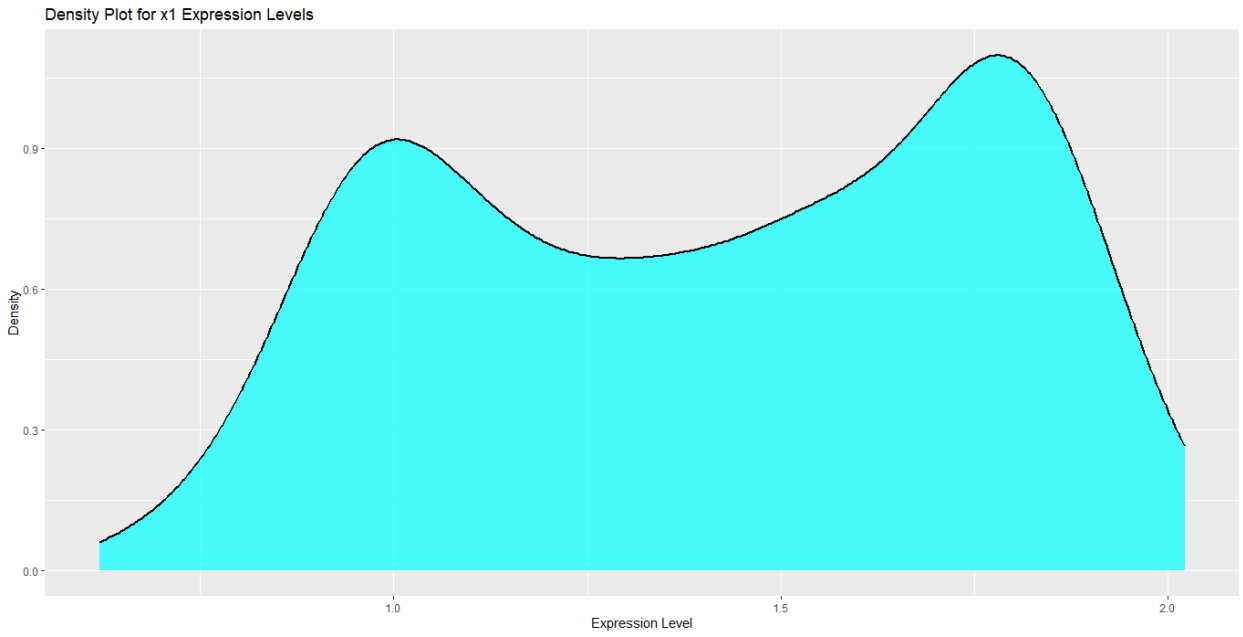
A density plot is a graphical representation that shows the distribution of a continuous numerical variable. It is a smoothed version of a histogram, often created using a kernel density estimate, which displays the probability density function of the variable. (Healy, 2019)

On the time series plots given below, density of data is being analyzed with respect to distribution of expression level of genes. The X-axis of plot represents Expression Level where as Y-axis of plot represents the density.

## Density Plot for Gene x1

The high point of the curve, around 1.0 on the x-axis, represents the expression level that is most common in the data set. The curve tapers off to the left and right of this peak, indicating that there are fewer data points at lower and higher expression levels. Overall, density plot of gene x1 suggests that the expression levels for x1 in this dataset are clustered around a central value, with some variation on either side.

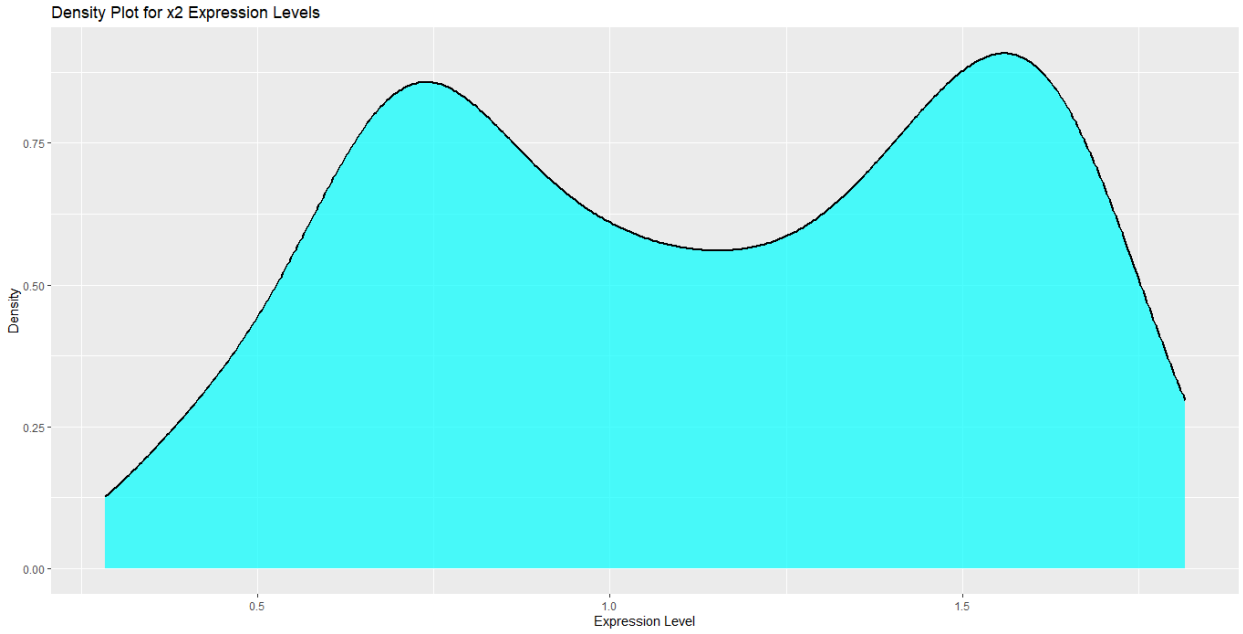
*Figure 7: Density Plot for x1*



## Density Plot for Gene x2

The most common expression level appears to be around 0.75 on the x-axis. The curve tapers off to the left and right of this peak, indicating that there are fewer data points at lower and higher expression levels of x2. Overall, density plot of Gene x2 suggests that the expression levels this dataset are clustered around 0.75, with some variation on either side.

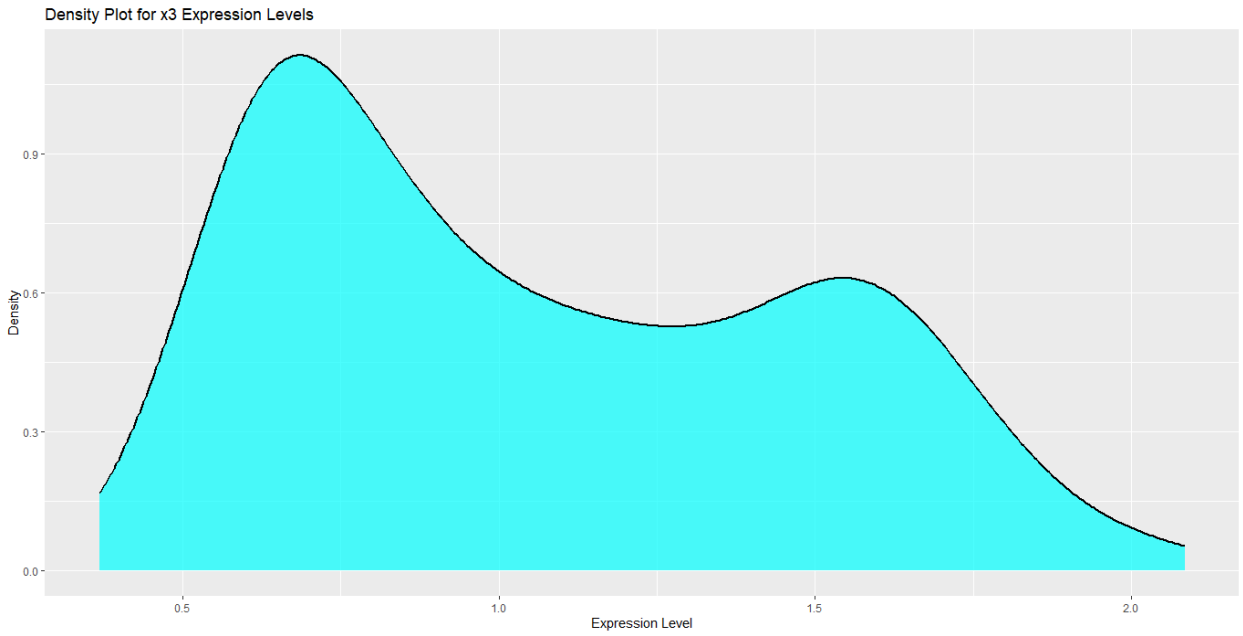
*Figure 8: Density Plot for x2*



## Density Plot for Gene x3

The most common expression level appears to be around 0.7 on the x-axis. The curve tapers off to the left and right of this peak, indicating that there are fewer data points at lower expression levels of x3. Overall, density plot of Gene x3 suggests that the expression levels this dataset are clustered around 0.9, with some variation on either side.

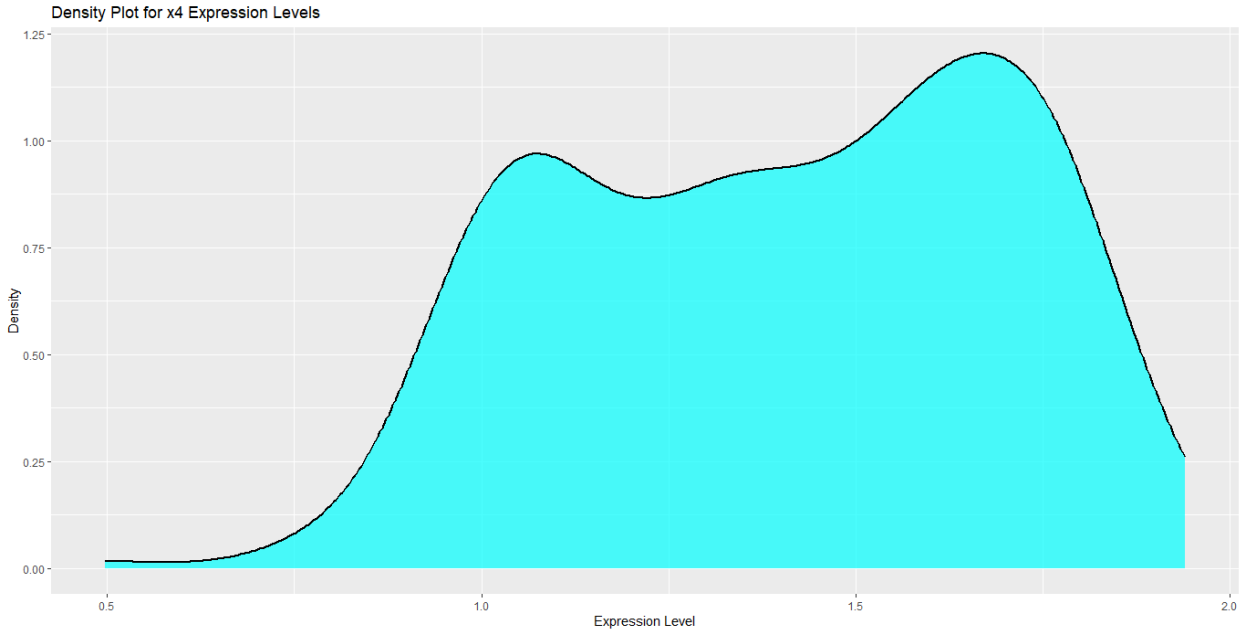
*Figure 9: Density Plot for x3*



## Density plot for Gene x4

The most common expression level appears to be around 1.75 on the x-axis. The curve tapers off to the left and right of this peak, indicating that there are fewer genes with very low or very high expression levels of x4. Overall, the density plot of Gene x4 suggests that the expression levels in this dataset are clustered around a central value, with some variation on either side.

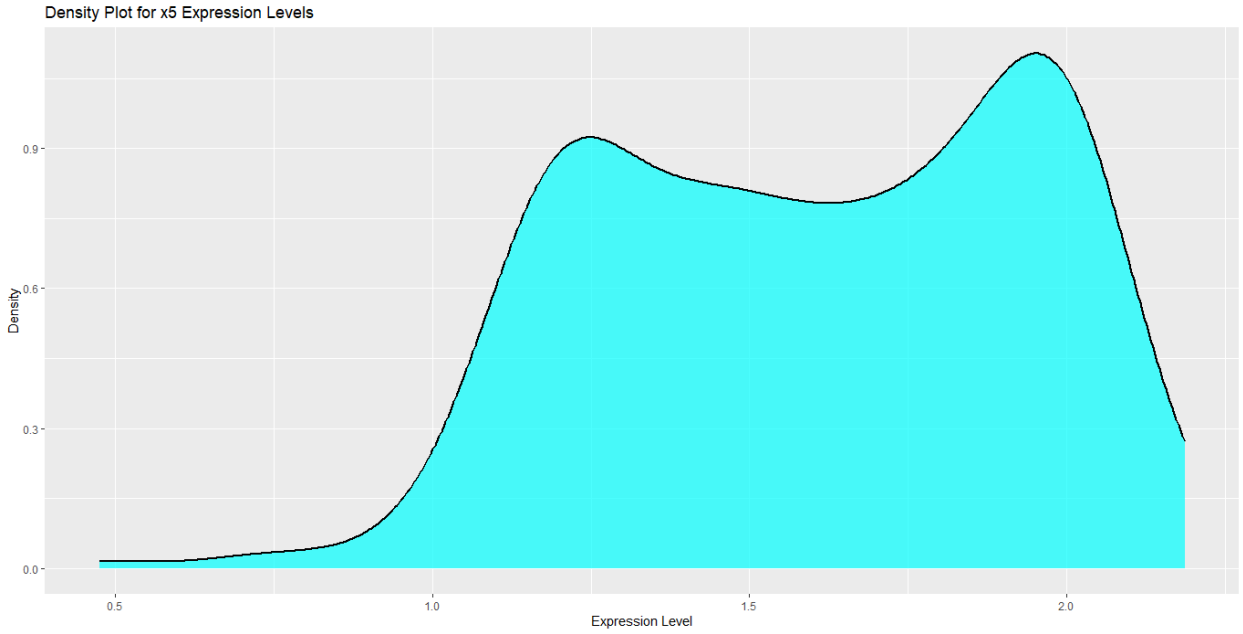
*Figure 10: Density Plot for x4*



## Density Plot for Gene x5

The most common expression level appears to be around 1.95 on the x-axis. The curve tapers off to the left and right of this peak, indicating that there are fewer data points at lower and higher expression levels of x5. Overall, this density plot suggests that the expression levels for x5 in this dataset are clustered around 1.9, with some variation on either side.

*Figure 11: Density Plot for x5*



## Distribution plots with density curves

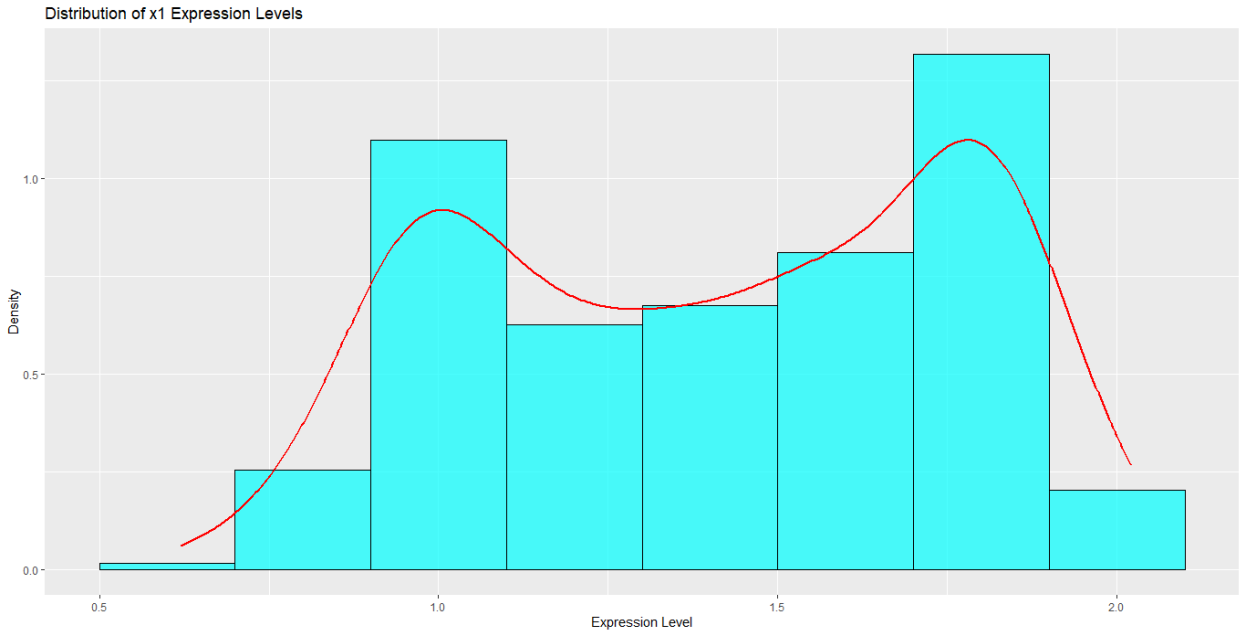
A distribution plot is a graphical way to show how your data is spread out. It helps visualize the shape of your data, whether it's clustered in the center, skewed to one side, or has multiple peaks.

On the distribution plots given below, density of distribution of expression levels of a gene is being analyzed. The x-axis represents the expression level and the y-axis represents the density.

## Distribution Plot with Density Curve for Gene x1

According to the distribution plot, the expression level of Gene x1 seems to be normally distributed. This means that most of the data points are clustered around the average, and there are fewer data points as you move away from the average in either direction. (Rob J Hyndman, April 2018)

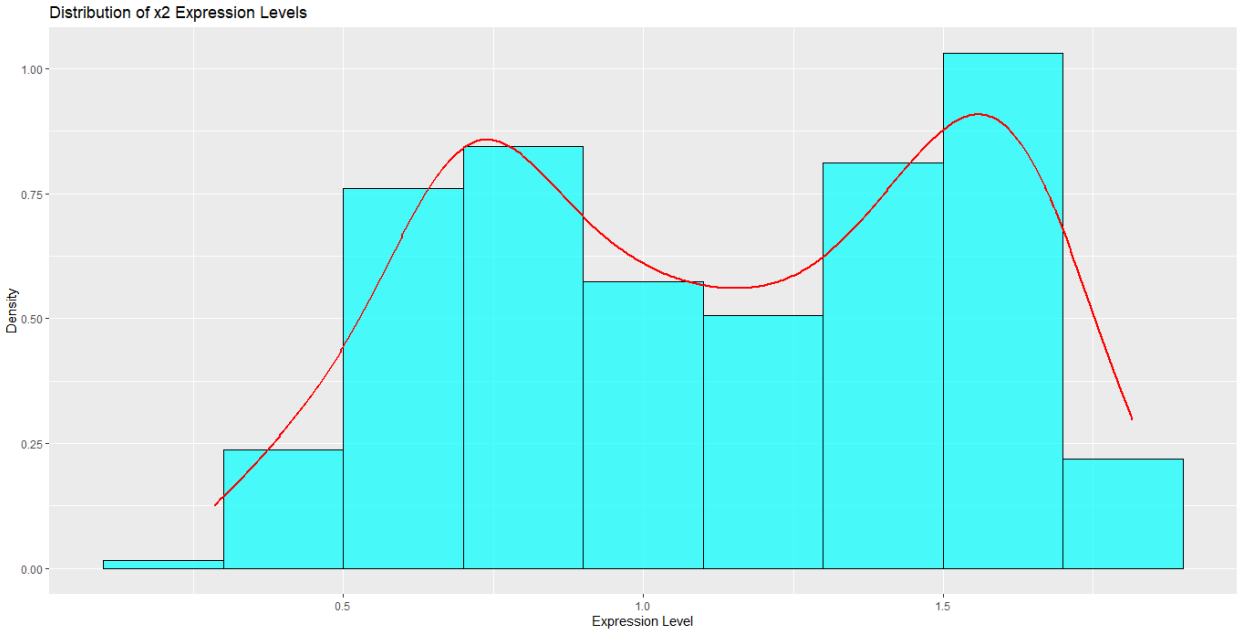
*Figure 12: Distribution plots with density curves x1*



## Distribution Plot with Density Curve for Gene x2

According to the distribution plot, the expression level of Gene x2 seems to be normally distributed. This means that most of the data points are clustered around the average, and there are fewer data points as you move away from the average in either direction.

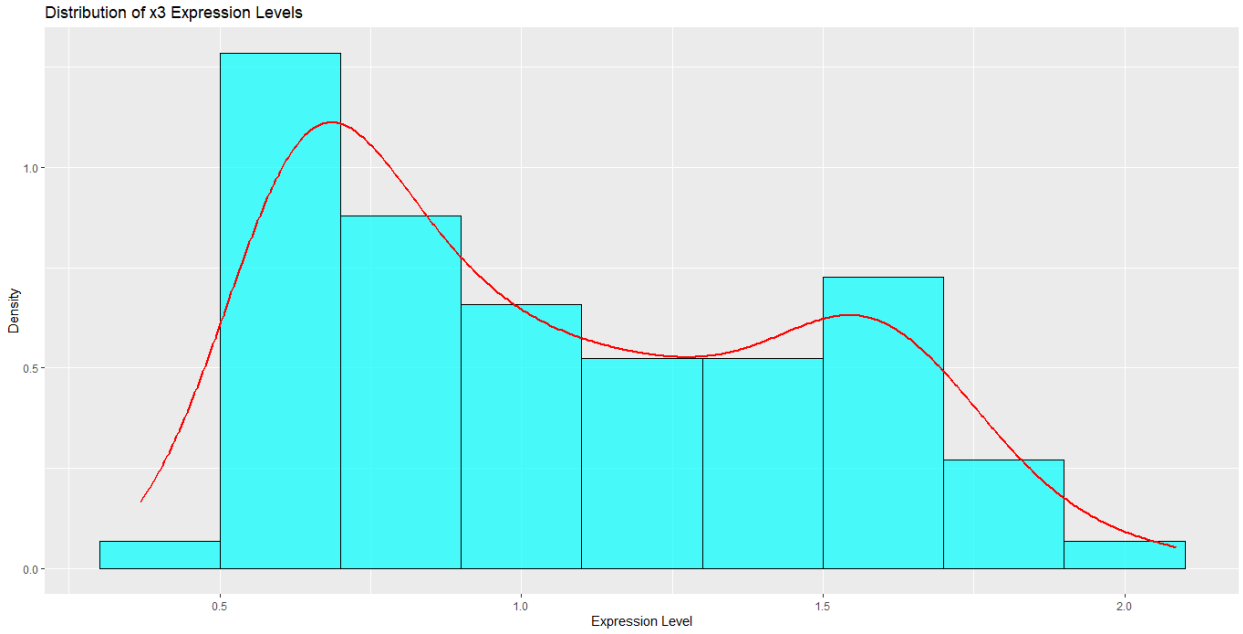
*Figure 13: Distribution plots with density curves x2*



## Distribution Plot with Density Curve for Gene x3

According to the distribution plot, the expression level of Gene x3 seems to be slightly more distributed to the left. This means that most of the data points are clustered around the 1.5 or lesser expression level on x-axis, and there are fewer data points as you move away from the average in right direction.

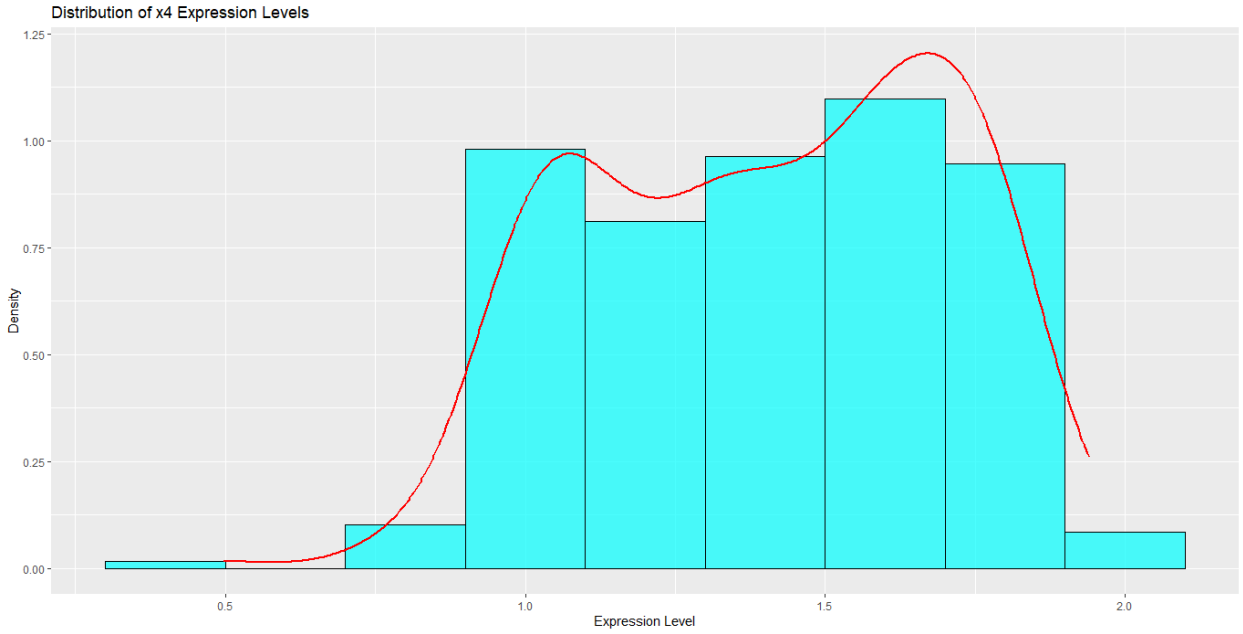
*Figure 14: Distribution plots with density curves x3*



## Distribution Plot with Density Curve for Gene x4

According to the distribution plot, the expression level of Gene x4 seems to be primarily distributed around the middle of the spectrum. This means that most of the data points are clustered around the average, and there are very few data points as you move away from the average in either direction.

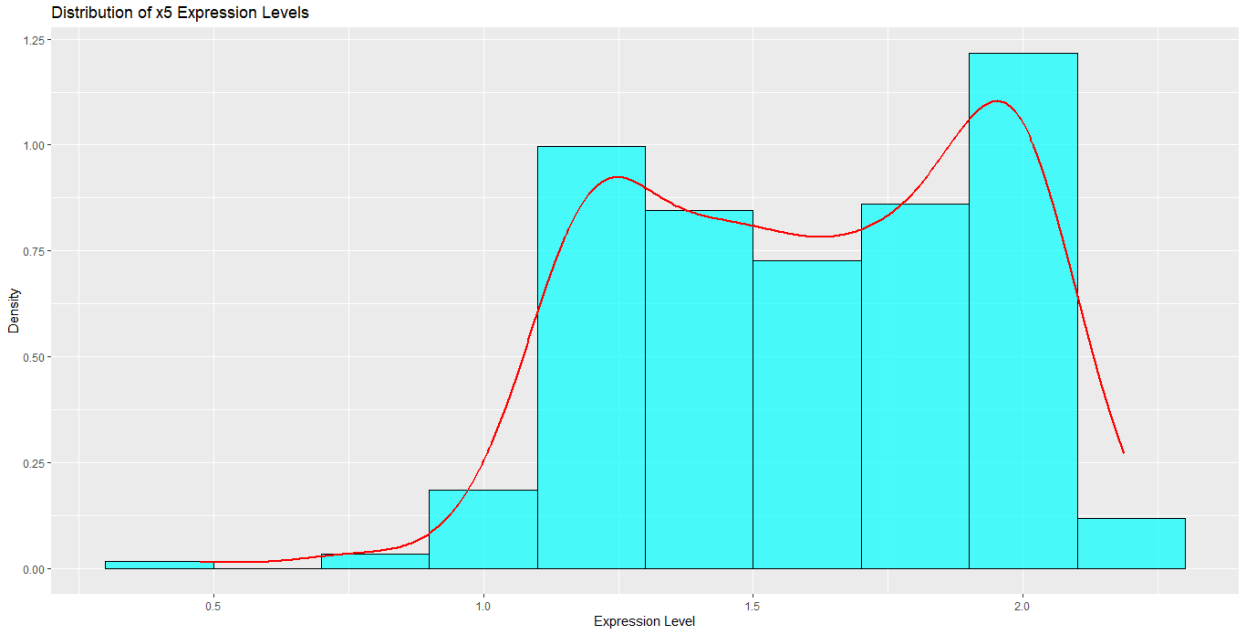
*Figure 15: Distribution plots with density curves x4*



## Distribution Plot with Density Curve for Gene x5

According to the distribution plot, the expression level of Gene x5 seems to be primarily distributed around the middle and right side the spectrum. This means that most of the data points are clustered around the average and slightly to the right, and there are very few data points as you move away from the average in either left direction.

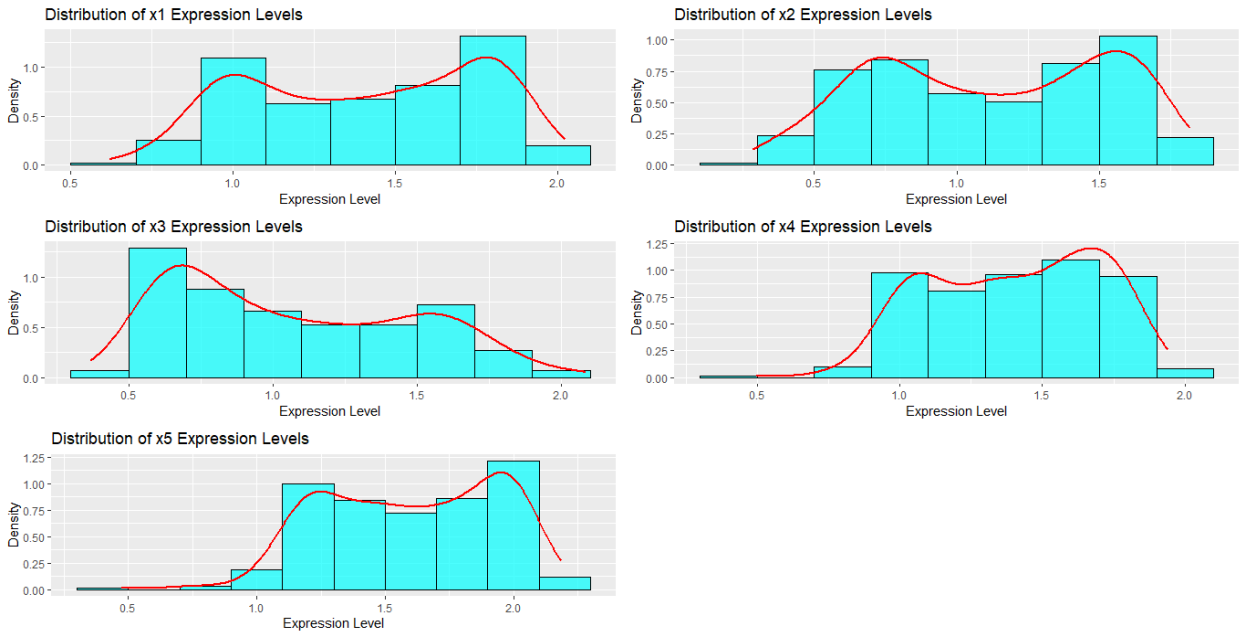
*Figure 16: Distribution plots with density curves x5*



## Distribution Plot with Density Curve for all the Gene expression

The below image represents the distribution plot with density curve of all the genes which we discussed above in detail.

*Figure 17: Distribution plots with density curves for all genes*

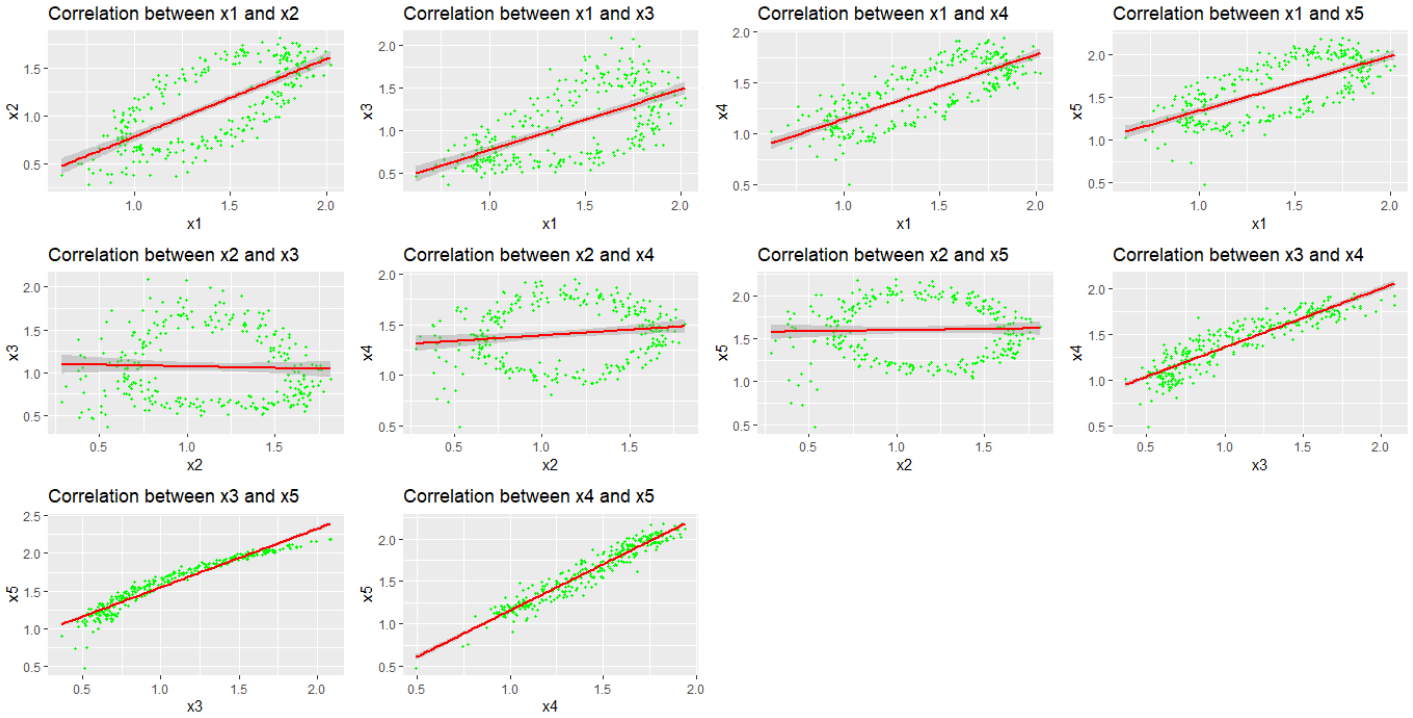


## Correlation and Scatter plots

Correlation and scatter plots are statistical and graphical techniques used to explore relationships between pairs of variables. Correlation quantifies the strength and direction of a linear relationship between two variables whereas scatter plot visually displays the relationship between variables by plotting individual data points. Together, they provide insights into how changes in one variable relate to changes in another. (Hand, 2013)

The correlation and scatter plot given below represents all the plots of combination of two genes (x1 and x2, x1 and x3, x1 and x4, x1 and x5, x2 and x3, x2 and x4, x2 and x5, x3 and x4, x3 and x5 and x4 and x5). The resulting plot from the R programming is attached below and their analysis is done respectively.

*Figure 18: Correlation and scatter plots (between different combination of two genes)*




### Correlation between x1 and x2

This scatter plot shows a positive correlation between x1 and x2. Points tend to increase diagonally from the bottom left to the upper right, indicating that as the value of x1 increases, the value of x2 also tends to increase.

**Correlation between x1 and x3**

This scatter plot shows a weak positive correlation between x1 and x3. There's a slight upward trend in the data points, but the data is scattered, making it difficult to see a clear association.

**Correlation between x1 and x4**

This scatter plot is inconclusive. There is no clear pattern to the data points, making it difficult to determine any correlation between x1 and x4.

**Correlation between x1 and x5**

This scatter plot shows a weak negative correlation between x1 and x5. There's a slight downward trend in the data points, but the data is scattered, making it difficult to see a clear association.

**Correlation between x2 and x3**

This scatter plot shows a positive correlation between x2 and x3. Points tend to increase diagonally from the bottom left to the upper right, indicating that as the value of x2 increases, the value of x3 also tends to increase.

**Correlation between x2 and x4**

This scatter plot shows a weak positive correlation between x2 and x4. There's a slight upward trend in the data points, but the data is scattered, making it difficult to see a clear association.

**Correlation between x2 and x5**

This scatter plot shows a weak positive correlation between x2 and x5. There's a slight upward trend in the data points, but the data is scattered, making it difficult to see a clear association.

**Correlation between x3 and x4**

This scatter plot shows a positive correlation between x3 and x4. Points tend to increase diagonally from the bottom left to the upper right, indicating that as the value of x3 increases, the value of x4 also tends to increase.

**Correlation between x3 and x5**

This scatter plot shows a strong positive correlation between x3 and x5. There's a clear upward trend in the data points, indicating a strong positive association between x3 and x5.

**Correlation between x4 and x5**

This scatter plot shows a positive correlation between x4 and x5. Points tend to increase diagonally from the bottom left to the upper right, indicating that as the value of x4 increases, the value of x5 also tends to increase.

## Task 2 Regression

## Task 2.1

Since the true value of the distribution is unknown, we use a random variable (estimator) to estimate its various properties. For calculating the estimator model parameters of a given input gene expression, we use a least squares method ($\hat{\theta}$, ThetaHat) with the variable name coeffModel. To determine $\hat{\theta}$ we use the formula $\hat{\theta} = (X^TX)^{-1}X^TY$ [coeffModel = $(X^TX)^{-1}X^TY$] where X is the dependent variable (input gene) and Y is the independent variable (output gene).

Given Candidate models:

Model 1: $y = \theta_1 x_4 + \theta_2 x_3^2 + \theta_{bias}$
Model 2: $y = \theta_1 x_4 + \theta_2 x_3^2 + \theta_3 x_5 + \theta_{bias}$
Model 3: $y = \theta_1 x_3 + \theta_2 x_4 + \theta_3 x_5^3$
Model 4: $y = \theta_1 x_4 + \theta_2 x_3^2 + \theta_3 x_5^3 + \theta_{bias}$
Model 5: $y = \theta_1 x_4 + \theta_2 x_1^2 + \theta_3 x_3^2 + \theta_{bias}$

To calculate the formula $\hat{\theta}$ (Least Squares) from the model mentioned above, the columns of the X (Input Gene) dataset are combined, and the least squares formula is applied using R programming. The results for all candidate models are attached below.

*Table 1: All Model Coefficient Values*

| **Coefficient Mode 1 (thetaHat $\hat{\theta}$)** | **Coefficient Mode 2 (thetaHat $\hat{\theta}$)** | **Coefficient Mode3 (thetaHat $\hat{\theta}$)** | **Coefficient Mode 4 (thetaHat $\hat{\theta}$)** | **Coefficient Mode 5 (thetaHat $\hat{\theta}$)** |
|---|---|---|---|---|
| [1] 0.04863538 | [1] 0.801640195 | [1,] -1.0002640 | [1] -0.9709444 | [1] 1.2951518 |
| [2] 1.03408905 | [2] 2.154751940 | [2] 0.5586768 | [2] 2.4521862 | [2] -0.8312983 |
| [3] -0.29218987 | [3] -0.007475522 | [3] 2.1363214 | [3] 0.3471645 | [3] 0.5385828 |
| | [4] -1.673939686 | [4] -0.3188386 | [4] -0.3904069 | [4] -0.1096679 |

## Task 2.2 RSS

The Residual Sum of Squares (RSS) tells us how closely a model's predictions match the real data. It measures the squared and mean difference between actual values and estimated values, serving as an indicator of accuracy (Michael H. Kutner, 2005). RSS is computed by assessing the errors of each candidate model using the θ values from task 2.1. A lower RSS indicates a better fit, because it means the model's estimates are, on average, closer to the real values.

$$RSS = \sum_{i=1}^{n} \left(y_i - \mathbf{x}_i \widehat{\boldsymbol{\theta}}\right)^2$$

In R, we can calculate the RSS using a formula that considers the number of data points, the differences between actual and predicted values, and the model's parameters. By comparing the RSS of different models, we can choose the one that best captures the real trends in the data. Below is the RSS values of the data given.

*Table 2: All model RSS values*

| RSS_Model1 | RSS_Model2 | RSS_Model3 | RSS_Model4 | RSS_Model5 |
|---|---|---|---|---|
| [1] 42.18919 | [1] 39.56195 | [1] 38.9683 | [1] 38.11274 | [1] 8.516473 |

## Task 2.3

The likelihood function assesses how well an observed value corresponds to the sample data of a specific model when the parameters are unknown. It helps identify the best-fitting distribution for the data based on a given observed value. By taking the logarithm of the likelihood function, we obtain the log likelihood, which maximizes the likelihood, equating to maximizing the maximum log-likelihood. This approach simplifies the estimation process, making it practical and widely used in various fields. (Edwards, 1972)

$$\ln p\left(D|\hat{\boldsymbol{\theta}}\right) = -\frac{n}{2}\ln(2\pi) - \frac{n}{2}\ln(\hat{\sigma}^2) - \frac{1}{2\hat{\sigma}^2}\text{RSS}$$

In the equation above, lnp(D| θˆ) represents the log-likelihood function, where n is the number of Y signals, and σˆ denotes the variance of the RSS, calculated as σˆ = RSS/(n-1). The variance (σˆ) is computed using R Programming and then applied to the given formula to determine the log-likelihood function. The resulting calculation is attached below.

*Table 3: All Model Likelyhood Values*

| Likelihood_model1 | Likelihood_model2 | Likelihood_model3 | Likelihood_model4 | Likelihood_model5 |
|---|---|---|---|---|
| -131.6805 | -122.1707 | -119.933 | -116.6474 | 105.1374 |

## Task 2.4 AIC and BIC

**Akaike Information Criterion (AIC)** predicts prediction error, indicating how well a model fits the dataset while avoiding overfitting. It assesses model quality by comparing different candidate models (W. N. Venables, 2002). In this context, 'k' is the number of parameters estimated in task 2.1, and 'L' is the likelihood function calculated in task 2.3.

$$AIC = 2k - 2\ln p(D|\hat{\boldsymbol{\theta}})$$

*Table 4: All Model AIC Values*

| AIC_model1 | AIC_model2 | AIC_model3 | AIC_model4 | AIC_model5 |
|---|---|---|---|---|
| 269.3611 | 252.3413 | 247.866 | 241.2949 | -202.2748 |

**The Bayesian Information Criterion (BIC)** is employed as a standard for selecting the best model from a finite set of choices. It assigns scores to various candidate models, favoring the one with the lowest score. Like the AIC, BIC depends on specific aspects of the likelihood function (W. N. Venables, 2002). Here, "k" signifies the number of estimated parameters, as determined in task 2.1, and "L" denotes the likelihood function computed in task 2.3.

$$BIC = k \cdot \ln(n) - 2\ln p(D|\hat{\boldsymbol{\theta}})$$

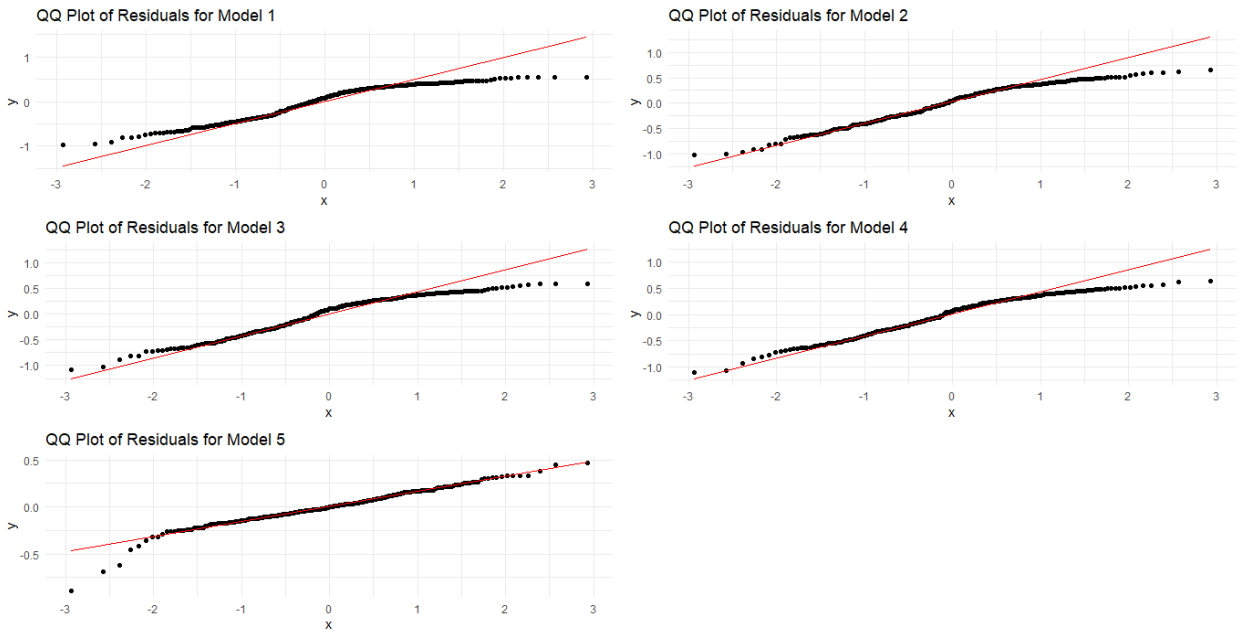
*Table 5: All Model BIC values*

| BIC_model1 | BIC_model2 | BIC_model3 | BIC_model4 | BIC_model5 |
|---|---|---|---|---|
| 280.4322 | 267.1028 | 262.6275 | 256.0563 | -187.5134 |

## Task 2.5 Distribution of Model Prediction Errors

In regression analysis, the residual (or error) is the difference between the predicted target (Expected Output Ŷ, as mentioned in 2.2) and the actual observed target (Output Y). This residual is used to assess model accuracy. For the error distribution plot, the predicted output Ŷ is subtracted from the actual output Y. The resulting error distribution is then utilized to create a QQ Plot and QQ Line using R programming. The resulting plot from the R programming is attached below and their analysis is done respectively.

*Figure 19: QQ Plot for Distribution of Model Prediction Errors*




### QQ Plot of Residuals for Model 1

The QQ plot for Model 1 shows significant deviations from the reference line, especially at the tails. The residuals do not align well with the line, indicating that they are not normally distributed. This suggests that the model may not adequately capture the underlying data structure or there may be outliers influencing the model fit. The S-shape curve indicates heavy tails and potential skewness in the residuals.

### QQ Plot of Residuals for Model 2

In the QQ plot for Model 2, the residuals follow the reference line more closely than in Model 1, although there are still deviations at the extremes. The central portion of the plot shows a reasonably good fit, but the tails exhibit some departures from normality. This suggests that while the model is an improvement over Model 1, it still faces some issues, particularly with extreme values.

**QQ Plot of Residuals for Model 3**

The QQ plot for Model 3 shows a pattern similar to Model 1, with clear deviations from the reference line, particularly in the tails. The residuals exhibit a noticeable S-shape, indicating significant departures from normality. This suggests that the model may have issues with fit, potentially due to skewness or heavy tails in the data. The model does not adequately capture the distribution of the residuals.

**QQ Plot of Residuals for Model 4**

The QQ plot for Model 4 indicates a moderate fit to the reference line, with residuals showing better alignment than in Models 1 and 3. However, there are still some deviations at the tails, suggesting that while the residuals are closer to normality, the model is not perfect. The fit is better in the central part of the distribution but shows some lack of fit at the extremes.

**QQ Plot of Residuals for Model 5**

The QQ plot for Model 5 shows the residuals closely following the reference line, indicating that they are approximately normally distributed. This suggests that the model captures the data well and has the best fit among the five models. There are minimal deviations in the tails, which means that the model's assumptions about the residuals are likely satisfied. This plot indicates a well-fitting model with appropriately distributed residuals.

In summary, Model 5 demonstrates the best fit, while Models 2 and 4 show moderate fits with some issues at the tails, and Models 1 and 3 exhibit significant deviations from normality.

## Task 2.6 Regression Model Selection

To determine the optimal regression model, Akaike Information Criterion (AIC) and Bayesian Information Criterion (BIC) were computed in task 2.4, while RSS was calculated in task 2.2. A comparison chart of AIC, BIC, and RSS is provided below for model selection.

*Table 6: AIC, BIC and RSS values for all Models*

| MODEL | AIC | BIC | RSS |
|---|---|---|---|
| **Model 1** | 269.3611 | 280.4322 | 42.18919 |
| **Model 2** | 252.3413 | 267.1028 | 39.56195 |
| **Model 3** | 247.866 | 262.6275 | 38.9683 |
| **Model 4** | 241.2949 | 256.0563 | 38.11274 |
| **Model 5** | -202.2748 | -187.5134 | 8.516473 |

In this assignment, the AIC and BIC metrics were used to deal with complex models. According to the analyses of both AIC and BIC, Model 5 exhibited the lowest values. Model 5 is indicated as the best model by AIC and BIC, and this is further supported by the RSS, where Model 5 has the lowest value. Thus, considering the results from AIC, BIC, and RSS, Model 5 is identified as the best regression model.

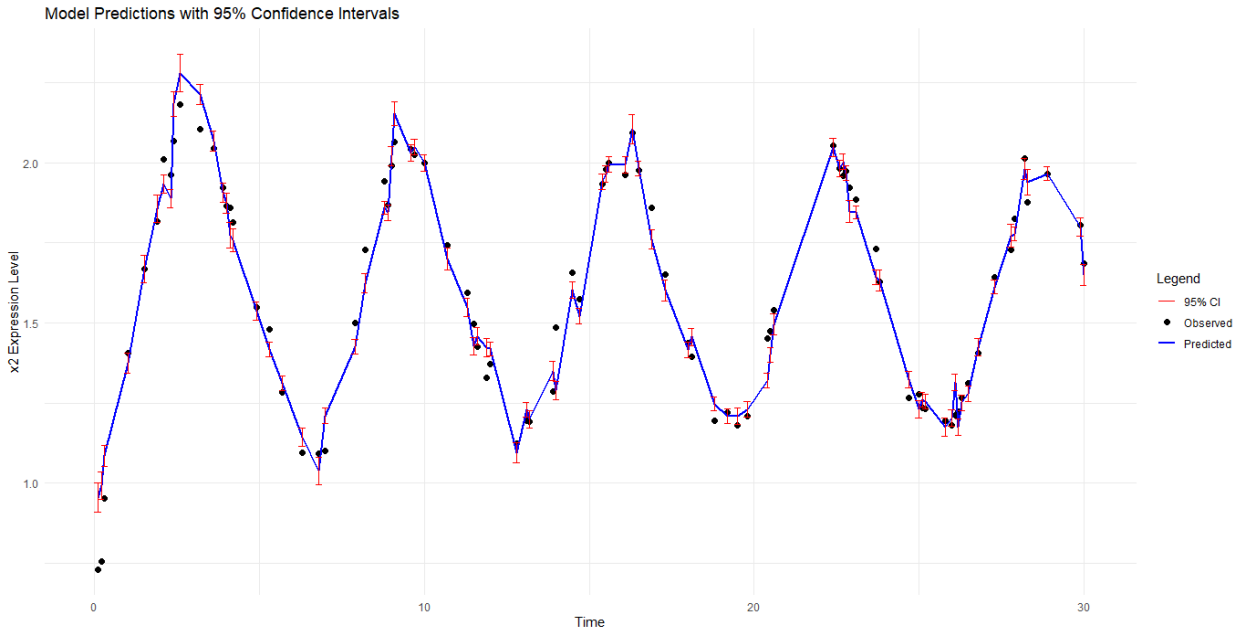
## Task 2.7 Data Splitting, Model Training, and Evaluation

The input data (X) and output data (Y) were initially split into two segments: the training dataset and the testing dataset. This split followed the provided guidelines, with 70% of the data allocated for training and the remaining 30% reserved for testing.

The model parameters were derived using the training data, specifically employing the chosen best model, model 5, to determine the value of q. These estimated parameters were then used to generate the model's output and predictions on the testing data. This process included calculating the Residual Sum of Squares (RSS) and using it for model prediction.

After computing the RSS and predicting on the testing data, a 95% confidence interval was established. This interval indicated the range within which the true parameter values were likely to fall, thereby adding a level of certainty to the model's predictions.

*Figure 20: Predicted vs. Observed Expression (95% CI)*




The plot indicates that the model effectively captures the overall trend and underlying patterns in the data, with predicted values aligning with observed values across most time points. The 95% confidence intervals generally encompass the observed values, suggesting reasonable uncertainty estimates. While there are a few points where the observed values fall outside these intervals, the deviations are generally small and randomly scattered, indicating no systematic bias. The confidence intervals widen at peaks and troughs, reflecting higher uncertainty in extreme values, which is typical in time-series models. Overall, the model effectively captures the data's underlying patterns, making reliable predictions with appropriately sized confidence intervals, despite some minor deviations.

# Task 3 Approximate Bayesian Computation

Approximate Bayesian Computation (ABC) refers to a set of computational techniques in Bayesian statistics designed to estimate posterior distributions when it is unfeasible to compute the likelihood function directly. Rather than computing the likelihood, ABC methods use simulations to create data and then compare this simulated data to the observed data through a selected distance metric (Beaumont, 2002).

Model 5 is selected as the best model as it fulfills the requirement criteria of *largest absolute value in* your least squares estimation calculated on Task 2.1. The two highest values are used to estimate the model parameters for determining the posterior distribution. Based on the least squares from Model 5, the first two values, θbias and Thetaone, are selected for calculating the posterior value, while the other two estimated parameters are kept constant as outlined below.

Parameters from Model 5 thetahat (coeffModel)

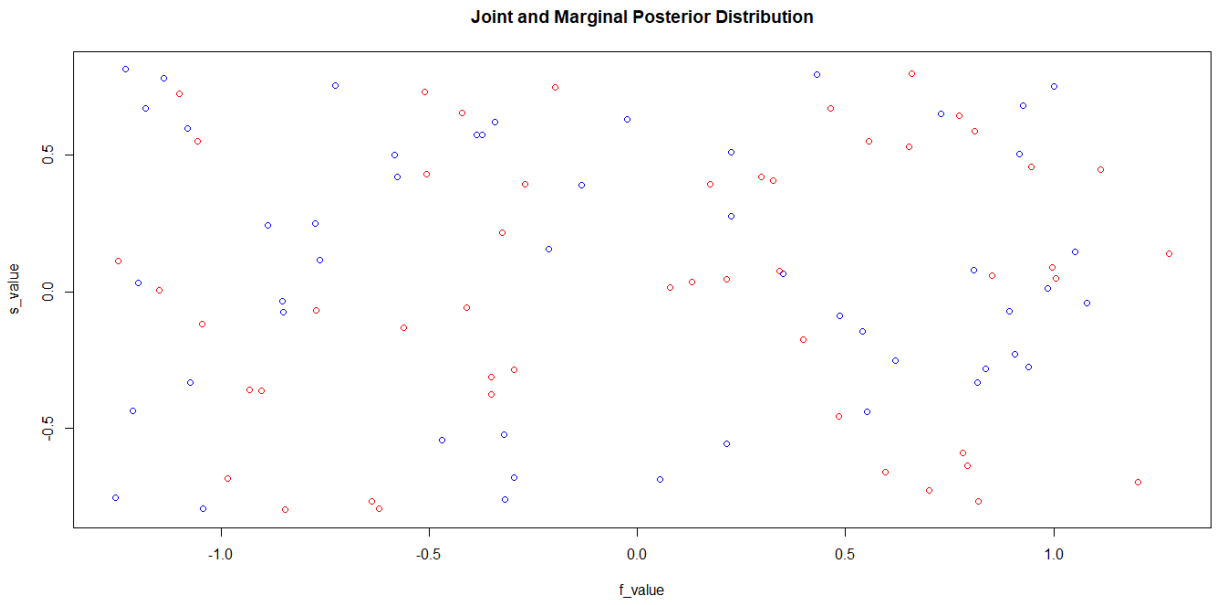
thetebias <- 1.2951518 (Chosen parameter)

thetaone <- 0.8312983 (Chosen parameter)

thetatwo <- 0.5385828 (Set Constant)

thetathree <- 0.1096679 (Set Constant)

*Figure 21: Scatter Plot of Joint and Marginal Posterior Distribution*




The joint and marginal posterior distribution plot of the two parameters shows a broad and relatively uniform spread, indicating significant uncertainty about their true values. The points are symmetrically distributed around the origin, suggesting that the uniform prior distributions,

centered around zero, were not substantially altered by the data. The lack of distinct clustering or pattern implies that the posterior distributions remain broad, potentially due to the high acceptance threshold in the rejection ABC method, which allows for considerable variability. Furthermore, there is no evident correlation between the two parameters, indicating they independently contribute to the model. This analysis suggests that the data does not strongly constrain the parameter values, resulting in a wide range of plausible values.

## Conclusion

In conclusion, this report analyzed and modeled the relationship between simulated gene expressions. We began by meticulously cleaning the provided dataset, ensuring accurate and efficient analysis. We performed preliminary data analysis through time series plots, distribution plots, and correlation analysis to understand expression level trends and potential gene interactions. Next, we employed regression modeling to identify the best-fitting model. Model 5 emerged as the optimal choice based on AIC, BIC, and RSS metrics, effectively capturing the underlying data patterns. The evaluation of this model demonstrated its ability to predict gene expression with reasonable uncertainty estimates. Finally, we utilized Approximate Bayesian Computation (ABC) to explore parameter estimation. The analysis revealed significant uncertainty in the true parameter values, suggesting that the data may not provide strong constraints. In conclusion, this report successfully identified the most suitable model for the given gene expression data and highlighted areas for further exploration, such as incorporating biological knowledge and improving parameter estimation through different techniques.

# References


Beaumont, M. A. (2002). Approximate Bayesian computation with application to statistics. *Approximate Bayesian computation with application to statistics*, 1619-1629.

Bruce Alberts, R. H. (2022). *Molecular Biology of the Cell* (4th ed.). New York, NY: W. W. Norton & Company.

Edwards, A. W. (1972). On the Likelihood Function and Its Role in Inference. *On the Likelihood Function and Its Role in Inference*, 98-113.

Hand, D. J. (2013). *Statistics: A Very Short Introduction.* Woodstock, Oxfordshire: Oxford University Press.

Healy, K. (2019). *Data Visualization: A Practical Introduction.* Woodstock, Oxfordshire: Princeton University Press.

Michael H. Kutner, C. J. (2005). *Applied Linear Statistical Models.* New York, NY: The McGraw-Hill Companies.

Rob J Hyndman, G. A. (April 2018). *Forecasting: Principles and Practice.* Monash University, Australia: OTexts.

W. N. Venables, B. D. (2002). *Modern Applied Statistics with S-Plus.* Springer-Verlag.

# Appendix

R code and its output for all the tasks are provided below. Also, the R file and dataset are available on the github repository provided below.

https://github.com/MrVexxx/15368843_Ayush_Rayamajhi_STW7089CEM.git

### Installing and importing packages

```
# Load necessary libraries
install.packages("ggplot2")
install.packages("gridExtra")
install.packages("corrplot")
install.packages("GGally")
install.packages("caret")
library(ggplot2)
library(gridExtra)
library(corrplot)
library(GGally)
library(tidyr)
library(rsample)
library(caret)
library(MASS)
```

### Loading dataset and validating it

```
# Load the dataset
dataset <- read.csv("E:/Stat CW/gene_data.csv")

# Define the required columns
required_columns <- c("Time", "x1", "x2", "x3", "x4", "x5")

# Check if all required columns are present and numeric
validation_results <- sapply(dataset[required_columns], is.numeric)

# Print the validation results
if (all(validation_results)) {
  print("The dataset contains all required columns and they are all numeric.")
} else {
  print("The dataset is missing the following required numeric columns:")
  print(names(validation_results)[!validation_results])
}
```

**Output**

```
> # Load the dataset
> dataset <- read.csv("E:/Stat CW/gene_data.csv")
>
>
>
> # Define the required columns
> required_columns <- c("Time", "x1", "x2", "x3", "x4", "x5")
>
> # Check if all required columns are present and numeric
> validation_results <- sapply(dataset[required_columns], is.numeric)
>
> # Print the validation results
> if (all(validation_results)) {
+   print("The dataset contains all required columns and they are all numeric.")
+ } else {
+   print("The dataset is missing the following required numeric columns:")
+   print(names(validation_results)[!validation_results])
+ }
[1] "The dataset contains all required columns and they are all numeric."
```

**Extracting columns from a dataset and then peek at the data to understand it better**

```
#separate columns from CSV
Time <- dataset$Time
x1 <- dataset$x1
x2 <-dataset$x2
x3 <-dataset$x3
x4 <-dataset$x4
x5 <-dataset$x5


# View the first few rows of the dataset
head(dataset)

# View the structure of the dataset
str(dataset)
```

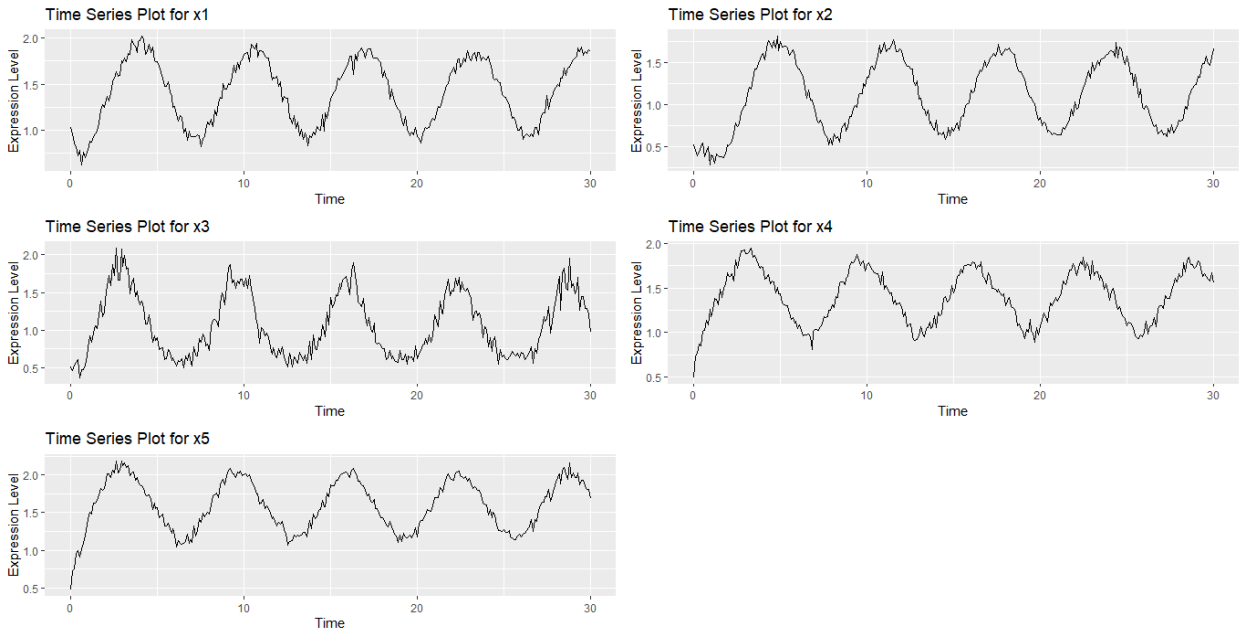
**Output**

```
> #separate columns from CSV
> Time <- dataset$Time
> x1 <- dataset$x1
> x2 <-dataset$x2
> x3 <-dataset$x3
> x4 <-dataset$x4
> x5 <-dataset$x5
>
>
>
> # View the first few rows of the dataset
> head(dataset)
  Time       x1       x2       x3       x4       x5
1  0.0 1.0271878 0.5313600 0.5108363 0.4976243 0.4749560
2  0.1 0.9566834 0.4626480 0.4553944 0.7431478 0.7301713
3  0.2 0.8603279 0.3983510 0.5298411 0.7717019 0.7547976
4  0.3 0.8228828 0.4405441 0.5645301 0.8762425 0.9510399
5  0.4 0.7204525 0.5094900 0.6016827 0.8558132 1.0006503
6  0.5 0.7839422 0.5465611 0.3680059 1.0108087 0.9062990
>
> # View the structure of the dataset
> str(dataset)
'data.frame':    296 obs. of  6 variables:
 $ Time: num  0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 ...
 $ x1  : num  1.027 0.957 0.86 0.823 0.72 ...
 $ x2  : num  0.531 0.463 0.398 0.441 0.509 ...
 $ x3  : num  0.511 0.455 0.53 0.565 0.602 ...
 $ x4  : num  0.498 0.743 0.772 0.876 0.856 ...
 $ x5  : num  0.475 0.73 0.755 0.951 1.001 ...
>
```

**Task -1: Time Series Plot**

```
# Time series plots
plot1 <- ggplot(dataset, aes(x = Time, y = x1)) +
  geom_line() +
  ggtitle("Time Series Plot for x1") +
  xlab("Time") +
  ylab("Expression Level")

plot2 <- ggplot(dataset, aes(x = Time, y = x2)) +
  geom_line() +
  ggtitle("Time Series Plot for x2") +
  xlab("Time") +
  ylab("Expression Level")

plot3 <- ggplot(dataset, aes(x = Time, y = x3)) +
  geom_line() +
  ggtitle("Time Series Plot for x3") +
  xlab("Time") +
  ylab("Expression Level")

plot4 <- ggplot(dataset, aes(x = Time, y = x4)) +
  geom_line() +
  ggtitle("Time Series Plot for x4") +
  xlab("Time") +
  ylab("Expression Level")

plot5 <- ggplot(dataset, aes(x = Time, y = x5)) +
  geom_line() +
  ggtitle("Time Series Plot for x5") +
  xlab("Time") +
  ylab("Expression Level")


# Print each plot one at a time
print(plot1)
print(plot2)
print(plot3)
print(plot4)
print(plot5)


# Arrange plots in a grid
grid.arrange(plot1, plot2, plot3, plot4, plot5, ncol = 2)
```

## Output

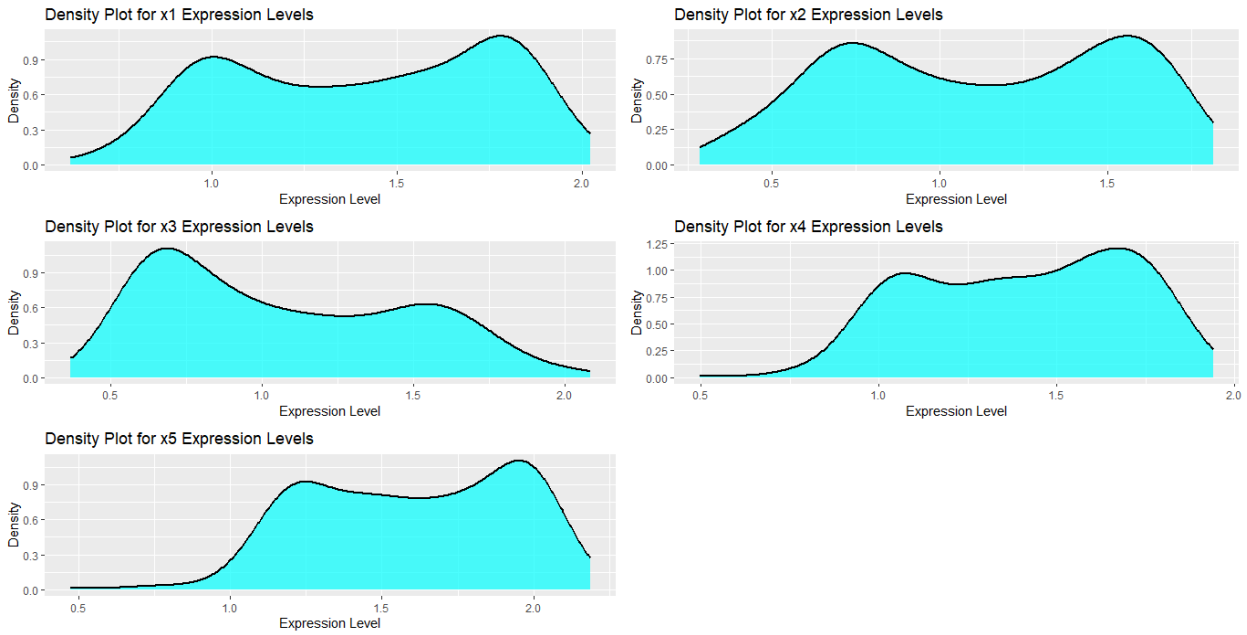



## Task-1: Density Plot

```
# Density plots
density_plot1 <- ggplot(dataset, aes(x = x1)) +
  geom_density(fill = "cyan", color = "black", alpha = 0.7, size = 1) +
  ggtitle("Density Plot for x1 Expression Levels") +
  xlab("Expression Level") +
  ylab("Density")

density_plot2 <- ggplot(dataset, aes(x = x2)) +
  geom_density(fill = "cyan", color = "black", alpha = 0.7, size = 1) +
  ggtitle("Density Plot for x2 Expression Levels") +
  xlab("Expression Level") +
  ylab("Density")

density_plot3 <- ggplot(dataset, aes(x = x3)) +
  geom_density(fill = "cyan", color = "black", alpha = 0.7, size = 1) +
  ggtitle("Density Plot for x3 Expression Levels") +
  xlab("Expression Level") +
  ylab("Density")

density_plot4 <- ggplot(dataset, aes(x = x4)) +
  geom_density(fill = "cyan", color = "black", alpha = 0.7, size = 1) +
  ggtitle("Density Plot for x4 Expression Levels") +
```

```
  xlab("Expression Level") +
  ylab("Density")

density_plot5 <- ggplot(dataset, aes(x = x5)) +
  geom_density(fill = "cyan", color = "black", alpha = 0.7, size = 1) +
  ggtitle("Density Plot for x5 Expression Levels") +
  xlab("Expression Level") +
  ylab("Density")


# Print each density plot one at a time
print(density_plot1)
print(density_plot2)
print(density_plot3)
print(density_plot4)
print(density_plot5)



# Arrange and print all density plots together
grid.arrange(density_plot1, density_plot2, density_plot3, density_plot4, density_plot5, ncol =
2)
```

## Output

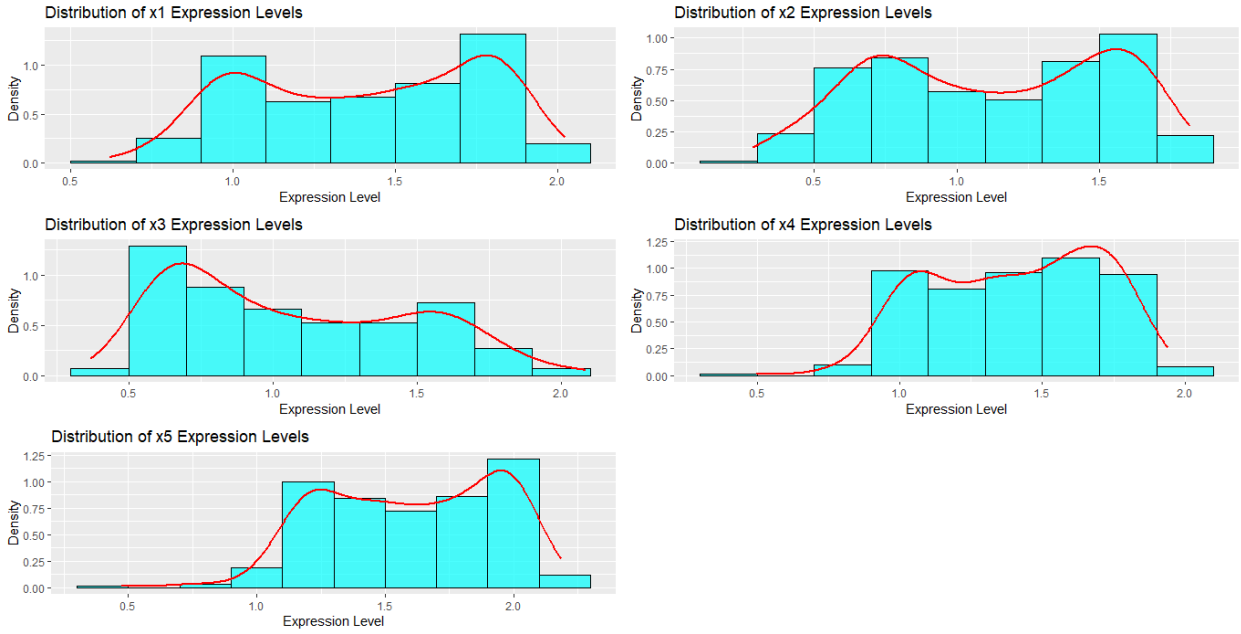


**Task-1: Distribution Plot**

```
# Distribution plots with density curves
dist_plot1 <- ggplot(dataset, aes(x = x1)) +
  geom_histogram(aes(y = ..density..), binwidth = 0.2, fill = "cyan", color = "black", alpha =
0.7) +
  geom_density(color = "red", size = 1) +
  ggtitle("Distribution of x1 Expression Levels") +
  xlab("Expression Level") +
  ylab("Density")

dist_plot2 <- ggplot(dataset, aes(x = x2)) +
  geom_histogram(aes(y = ..density..), binwidth = 0.2, fill = "cyan", color = "black", alpha =
0.7) +
  geom_density(color = "red", size = 1) +
  ggtitle("Distribution of x2 Expression Levels") +
  xlab("Expression Level") +
  ylab("Density")

dist_plot3 <- ggplot(dataset, aes(x = x3)) +
  geom_histogram(aes(y = ..density..), binwidth = 0.2, fill = "cyan", color = "black", alpha =
0.7) +
  geom_density(color = "red", size = 1) +
  ggtitle("Distribution of x3 Expression Levels") +
  xlab("Expression Level") +
  ylab("Density")

dist_plot4 <- ggplot(dataset, aes(x = x4)) +
  geom_histogram(aes(y = ..density..), binwidth = 0.2, fill = "cyan", color = "black", alpha =
0.7) +
  geom_density(color = "red", size = 1) +
  ggtitle("Distribution of x4 Expression Levels") +
  xlab("Expression Level") +
  ylab("Density")

dist_plot5 <- ggplot(dataset, aes(x = x5)) +
  geom_histogram(aes(y = ..density..), binwidth = 0.2, fill = "cyan", color = "black", alpha =
0.7) +
  geom_density(color = "red", size = 1) +
  ggtitle("Distribution of x5 Expression Levels") +
  xlab("Expression Level") +
  ylab("Density")


# Print each distribution plot one at a time
print(dist_plot1)
```

```
print(dist_plot2)
print(dist_plot3)
print(dist_plot4)
print(dist_plot5)



# Arrange and print all distribution plots together
grid.arrange(dist_plot1, dist_plot2, dist_plot3, dist_plot4, dist_plot5, ncol = 2)
```

## Output

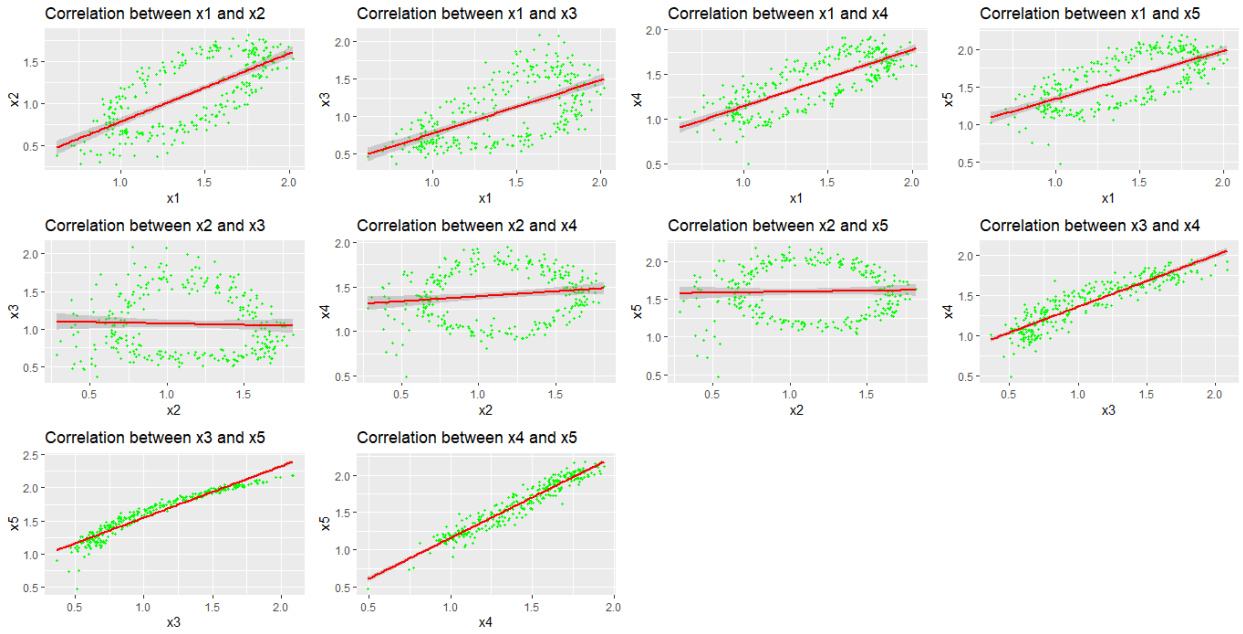



## Task1- Correlation and scatter plots

```
# Create a list to store plots
plot_list <- list()

# Loop through combinations of genes to create plots
for (i in 1:4) {
  for (j in (i+1):5) {
    # Create scatter plot with regression line
    p <- ggplot(data = dataset, aes_string(x = names(dataset)[i + 1], y = names(dataset)[j + 1]))
+
      geom_point(color = "green", size = 0.8 ) +
      geom_smooth(method = "lm", color = "red") +
```

```
      labs(title = paste("Correlation between", names(dataset)[i + 1], "and", names(dataset)[j +
1]),
           x = names(dataset)[i + 1], y = names(dataset)[j + 1])

    # Add plot to plot_list
    plot_list[[length(plot_list) + 1]] <- p
  }
}

# Arrange and print all plots together in a grid layout
grid.arrange(grobs = plot_list, ncol = 4)
```

## Output




## Task 2.1: Creating Model

```
#Creating Model

#Task 2.1
# Model 1
onesMatrix <- matrix(1, length(x1), 1)
y <- dataset$x2

xModel1 <- cbind(onesMatrix, x4, x3 ^ 2)

coeffModel1 <- solve(t(xModel1) %*% xModel1) %*% t(xModel1) %*% y
```

```
print(coeffModel1)


# Model 2
xModel2 <- cbind(onesMatrix, x4, x3^5, x5)

coeffModel2 <- solve(t(xModel2) %*% xModel2) %*% t(xModel2) %*% y
print(coeffModel2)


# Model 3
xModel3 <- cbind(onesMatrix, x3, x4, x5 ^ 3)

coeffModel3 <- solve(t(xModel3) %*% xModel3) %*% t(xModel3) %*% y
print(coeffModel3)


# Model 4
xModel4 <- cbind(onesMatrix, x4, x3 ^ 2, x5 ^ 3)

coeffModel4 <- solve(t(xModel4) %*% xModel4) %*% t(xModel4) %*% y
print(coeffModel4)


# Model 5
xModel5 <- cbind(onesMatrix, x4, x1 ^ 2, x3 ^ 2)

coeffModel5 <- solve(t(xModel5) %*% xModel5) %*% t(xModel5) %*% y
print(coeffModel5)

#Task 2.1 End
```

**Output**

```
> #Task 2.1
> # Model 1
> onesMatrix <- matrix(1, length(x1), 1)
> y <- dataset$x2
>
> xModel1 <- cbind(onesMatrix, x4, x3 ^ 2)
>
> coeffModel1 <- solve(t(xModel1) %*% xModel1) %*% t(xModel1) %*% y
> print(coeffModel1)
          [,1]
   0.04863538
x4  1.03408905
```

```
  -0.29218987
>
>
> # Model 2
> xModel2 <- cbind(onesMatrix, x4, x3^5, x5)
>
> coeffModel2 <- solve(t(xModel2) %*% xModel2) %*% t(xModel2) %*% y
> print(coeffModel2)
         [,1]
   0.801640195
x4  2.154751940
  -0.007475522
x5 -1.673939686
>
>
> # Model 3
> xModel3 <- cbind(onesMatrix, x3, x4, x5 ^ 3)
>
> coeffModel3 <- solve(t(xModel3) %*% xModel3) %*% t(xModel3) %*% y
> print(coeffModel3)
        [,1]
  -1.0002640
x3  0.5586768
x4  2.1363214
  -0.3188386
>
>
> # Model 4
> xModel4 <- cbind(onesMatrix, x4, x3 ^ 2, x5 ^ 3)
>
> coeffModel4 <- solve(t(xModel4) %*% xModel4) %*% t(xModel4) %*% y
> print(coeffModel4)
        [,1]
  -0.9709444
x4  2.4521862
   0.3471645
  -0.3904069
>
>
> # Model 5
> xModel5 <- cbind(onesMatrix, x4, x1 ^ 2, x3 ^ 2)
>
> coeffModel5 <- solve(t(xModel5) %*% xModel5) %*% t(xModel5) %*% y
> print(coeffModel5)
        [,1]
   1.2951518
```

```
x4 -0.8312983
    0.5385828
   -0.1096679
>
```

**Task 2.2: Compute the model residual (error) sum of squared errors (RSS) for every candidate model**

```
# Task 2.2: Compute the model residual (error) sum of squared errors (RSS) for every
candidate model

# Function to compute RSS
computeRSS <- function(y, y_pred) {
  residuals <- y - y_pred
  RSS <- sum(residuals^2)
  return(RSS)
}

# Model 1: y = θ1 * x4 + θ2 * x3^2 + θ_bias
y_pred_model1 <- xModel1 %*% coeffModel1
RSS_model1 <- computeRSS(y, y_pred_model1)
print("RSS for Model 1:")
print(RSS_model1)

# Model 2: y = θ1 * x4 + θ2 * x3^5 + θ3 * x5 + θ_bias
y_pred_model2 <- xModel2 %*% coeffModel2
RSS_model2 <- computeRSS(y, y_pred_model2)
print("RSS for Model 2:")
print(RSS_model2)

# Model 3: y = θ1 * x3 + θ2 * x4 + θ3 * x5^3 + θ_bias
y_pred_model3 <- xModel3 %*% coeffModel3
RSS_model3 <- computeRSS(y, y_pred_model3)
print("RSS for Model 3:")
print(RSS_model3)

# Model 4: y = θ1 * x4 + θ2 * x3^2 + θ3 * x5^3 + θ_bias
y_pred_model4 <- xModel4 %*% coeffModel4
RSS_model4 <- computeRSS(y, y_pred_model4)
print("RSS for Model 4:")
print(RSS_model4)

# Model 5: y = θ1 * x4 + θ2 * x1^2 + θ3 * x3^2 + θ_bias
```

```
y_pred_model5 <- xModel5 %*% coeffModel5
RSS_model5 <- computeRSS(y, y_pred_model5)
print("RSS for Model 5:")
print(RSS_model5)


#Task 2.2 end
```

**Output**

```
> # Task 2.2: Compute the model residual (error) sum of squared errors (RSS) for every
candidate model
>
> # Function to compute RSS
> computeRSS <- function(y, y_pred) {
+   residuals <- y - y_pred
+   RSS <- sum(residuals^2)
+   return(RSS)
+ }
>
> # Model 1: y = θ1 * x4 + θ2 * x3^2 + θ_bias
> y_pred_model1 <- xModel1 %*% coeffModel1
> RSS_model1 <- computeRSS(y, y_pred_model1)
> print("RSS for Model 1:")
[1] "RSS for Model 1:"
> print(RSS_model1)
[1] 42.18919
>
> # Model 2: y = θ1 * x4 + θ2 * x3^5 + θ3 * x5 + θ_bias
> y_pred_model2 <- xModel2 %*% coeffModel2
> RSS_model2 <- computeRSS(y, y_pred_model2)
> print("RSS for Model 2:")
[1] "RSS for Model 2:"
> print(RSS_model2)
[1] 39.56195
>
> # Model 3: y = θ1 * x3 + θ2 * x4 + θ3 * x5^3 + θ_bias
> y_pred_model3 <- xModel3 %*% coeffModel3
> RSS_model3 <- computeRSS(y, y_pred_model3)
> print("RSS for Model 3:")
[1] "RSS for Model 3:"
> print(RSS_model3)
[1] 38.9683
>
> # Model 4: y = θ1 * x4 + θ2 * x3^2 + θ3 * x5^3 + θ_bias
> y_pred_model4 <- xModel4 %*% coeffModel4
```

```
> RSS_model4 <- computeRSS(y, y_pred_model4)
> print("RSS for Model 4:")
[1] "RSS for Model 4:"
> print(RSS_model4)
[1] 38.11274
>
> # Model 5: y = θ1 * x4 + θ2 * x1^2 + θ3 * x3^2 + θ_bias
> y_pred_model5 <- xModel5 %*% coeffModel5
> RSS_model5 <- computeRSS(y, y_pred_model5)
> print("RSS for Model 5:")
[1] "RSS for Model 5:"
> print(RSS_model5)
[1] 8.516473
>
>
```

**Task 2.3: Compute the log-likelihood function for every candidate model**

```
# Task 2.3: Compute the log-likelihood function for every candidate model

# Function to compute log-likelihood
computeLogLikelihood <- function(y, y_pred, n, p) {
  residuals <- y - y_pred
  RSS <- sum(residuals^2)
  sigma2 <- RSS / (n - p)
  log_likelihood <- -n/2 * log(2 * pi) - n/2 * log(sigma2) - 1/(2 * sigma2) * RSS
  return(log_likelihood)
}

# Number of data samples
n <- length(y)

# Model 1: y = θ1 * x4 + θ2 * x3^2 + θ_bias
p1 <- ncol(xModel1)  # number of parameters
log_likelihood_model1 <- computeLogLikelihood(y, y_pred_model1, n, p1)
print("Log-likelihood for Model 1:")
print(log_likelihood_model1)

# Model 2: y = θ1 * x4 + θ2 * x3^5 + θ3 * x5 + θ_bias
p2 <- ncol(xModel2)  # number of parameters
log_likelihood_model2 <- computeLogLikelihood(y, y_pred_model2, n, p2)
print("Log-likelihood for Model 2:")
print(log_likelihood_model2)

# Model 3: y = θ1 * x3 + θ2 * x4 + θ3 * x5^3 + θ_bias
```

```
p3 <- ncol(xModel3)  # number of parameters
log_likelihood_model3 <- computeLogLikelihood(y, y_pred_model3, n, p3)
print("Log-likelihood for Model 3:")
print(log_likelihood_model3)

# Model 4: y = θ1 * x4 + θ2 * x3^2 + θ3 * x5^3 + θ_bias
p4 <- ncol(xModel4)  # number of parameters
log_likelihood_model4 <- computeLogLikelihood(y, y_pred_model4, n, p4)
print("Log-likelihood for Model 4:")
print(log_likelihood_model4)

# Model 5: y = θ1 * x4 + θ2 * x1^2 + θ3 * x3^2 + θ_bias
p5 <- ncol(xModel5)  # number of parameters
log_likelihood_model5 <- computeLogLikelihood(y, y_pred_model5, n, p5)
print("Log-likelihood for Model 5:")
print(log_likelihood_model5)

#Task 2.3 end
```

**Output**

```
> # Task 2.3: Compute the log-likelihood function for every candidate model
>
> # Function to compute log-likelihood
> computeLogLikelihood <- function(y, y_pred, n, p) {
+   residuals <- y - y_pred
+   RSS <- sum(residuals^2)
+   sigma2 <- RSS / (n - p)
+   log_likelihood <- -n/2 * log(2 * pi) - n/2 * log(sigma2) - 1/(2 * sigma2) * RSS
+   return(log_likelihood)
+ }
>
> # Number of data samples
> n <- length(y)
>
> # Model 1: y = θ1 * x4 + θ2 * x3^2 + θ_bias
> p1 <- ncol(xModel1)  # number of parameters
> log_likelihood_model1 <- computeLogLikelihood(y, y_pred_model1, n, p1)
> print("Log-likelihood for Model 1:")
[1] "Log-likelihood for Model 1:"
> print(log_likelihood_model1)
[1] -131.6805
>
> # Model 2: y = θ1 * x4 + θ2 * x3^5 + θ3 * x5 + θ_bias
> p2 <- ncol(xModel2)  # number of parameters
> log_likelihood_model2 <- computeLogLikelihood(y, y_pred_model2, n, p2)
```

```
> print("Log-likelihood for Model 2:")
[1] "Log-likelihood for Model 2:"
> print(log_likelihood_model2)
[1] -122.1707
>
> # Model 3: y = θ1 * x3 + θ2 * x4 + θ3 * x5^3 + θ_bias
> p3 <- ncol(xModel3)  # number of parameters
> log_likelihood_model3 <- computeLogLikelihood(y, y_pred_model3, n, p3)
> print("Log-likelihood for Model 3:")
[1] "Log-likelihood for Model 3:"
> print(log_likelihood_model3)
[1] -119.933
>
> # Model 4: y = θ1 * x4 + θ2 * x3^2 + θ3 * x5^3 + θ_bias
> p4 <- ncol(xModel4)  # number of parameters
> log_likelihood_model4 <- computeLogLikelihood(y, y_pred_model4, n, p4)
> print("Log-likelihood for Model 4:")
[1] "Log-likelihood for Model 4:"
> print(log_likelihood_model4)
[1] -116.6474
>
> # Model 5: y = θ1 * x4 + θ2 * x1^2 + θ3 * x3^2 + θ_bias
> p5 <- ncol(xModel5)  # number of parameters
> log_likelihood_model5 <- computeLogLikelihood(y, y_pred_model5, n, p5)
> print("Log-likelihood for Model 5:")
[1] "Log-likelihood for Model 5:"
> print(log_likelihood_model5)
[1] 105.1374
>
```

**Task 2.4: Compute AIC and BIC for every candidate model**

```
# Task 2.4: Compute the Akaike information criterion (AIC) and Bayesian information
criterion (BIC) for every candidate model

# Function to compute AIC
computeAIC <- function(log_likelihood, p) {
  AIC <- 2 * p - 2 * log_likelihood
  return(AIC)
}

# Function to compute BIC
computeBIC <- function(log_likelihood, p, n) {
  BIC <- log(n) * p - 2 * log_likelihood
  return(BIC)
```

```
}

# Number of data samples
n <- length(y)

# Model 1: AIC and BIC
log_likelihood_model1 <- computeLogLikelihood(y, y_pred_model1, n, p1)
AIC_model1 <- computeAIC(log_likelihood_model1, p1)
BIC_model1 <- computeBIC(log_likelihood_model1, p1, n)
print("AIC for Model 1:")
print(AIC_model1)
print("BIC for Model 1:")
print(BIC_model1)

# Model 2: AIC and BIC
log_likelihood_model2 <- computeLogLikelihood(y, y_pred_model2, n, p2)
AIC_model2 <- computeAIC(log_likelihood_model2, p2)
BIC_model2 <- computeBIC(log_likelihood_model2, p2, n)
print("AIC for Model 2:")
print(AIC_model2)
print("BIC for Model 2:")
print(BIC_model2)

# Model 3: AIC and BIC
log_likelihood_model3 <- computeLogLikelihood(y, y_pred_model3, n, p3)
AIC_model3 <- computeAIC(log_likelihood_model3, p3)
BIC_model3 <- computeBIC(log_likelihood_model3, p3, n)
print("AIC for Model 3:")
print(AIC_model3)
print("BIC for Model 3:")
print(BIC_model3)

# Model 4: AIC and BIC
log_likelihood_model4 <- computeLogLikelihood(y, y_pred_model4, n, p4)
AIC_model4 <- computeAIC(log_likelihood_model4, p4)
BIC_model4 <- computeBIC(log_likelihood_model4, p4, n)
print("AIC for Model 4:")
print(AIC_model4)
print("BIC for Model 4:")
print(BIC_model4)

# Model 5: AIC and BIC
log_likelihood_model5 <- computeLogLikelihood(y, y_pred_model5, n, p5)
AIC_model5 <- computeAIC(log_likelihood_model5, p5)
BIC_model5 <- computeBIC(log_likelihood_model5, p5, n)
print("AIC for Model 5:")
```

```
print(AIC_model5)
print("BIC for Model 5:")
print(BIC_model5)

# End Task 2.4
```

**Output**

```
> # Task 2.4: Compute the Akaike information criterion (AIC) and Bayesian information
criterion (BIC) for every candidate model
>
> # Function to compute AIC
> computeAIC <- function(log_likelihood, p) {
+   AIC <- 2 * p - 2 * log_likelihood
+   return(AIC)
+ }
>
> # Function to compute BIC
> computeBIC <- function(log_likelihood, p, n) {
+   BIC <- log(n) * p - 2 * log_likelihood
+   return(BIC)
+ }
>
> # Number of data samples
> n <- length(y)
>
> # Model 1: AIC and BIC
> log_likelihood_model1 <- computeLogLikelihood(y, y_pred_model1, n, p1)
> AIC_model1 <- computeAIC(log_likelihood_model1, p1)
> BIC_model1 <- computeBIC(log_likelihood_model1, p1, n)
> print("AIC for Model 1:")
[1] "AIC for Model 1:"
> print(AIC_model1)
[1] 269.3611
> print("BIC for Model 1:")
[1] "BIC for Model 1:"
> print(BIC_model1)
[1] 280.4322
>
> # Model 2: AIC and BIC
> log_likelihood_model2 <- computeLogLikelihood(y, y_pred_model2, n, p2)
> AIC_model2 <- computeAIC(log_likelihood_model2, p2)
> BIC_model2 <- computeBIC(log_likelihood_model2, p2, n)
> print("AIC for Model 2:")
[1] "AIC for Model 2:"
> print(AIC_model2)
```

```
[1] 252.3413
> print("BIC for Model 2:")
[1] "BIC for Model 2:"
> print(BIC_model2)
[1] 267.1028
>
> # Model 3: AIC and BIC
> log_likelihood_model3 <- computeLogLikelihood(y, y_pred_model3, n, p3)
> AIC_model3 <- computeAIC(log_likelihood_model3, p3)
> BIC_model3 <- computeBIC(log_likelihood_model3, p3, n)
> print("AIC for Model 3:")
[1] "AIC for Model 3:"
> print(AIC_model3)
[1] 247.866
> print("BIC for Model 3:")
[1] "BIC for Model 3:"
> print(BIC_model3)
[1] 262.6275
>
> # Model 4: AIC and BIC
> log_likelihood_model4 <- computeLogLikelihood(y, y_pred_model4, n, p4)
> AIC_model4 <- computeAIC(log_likelihood_model4, p4)
> BIC_model4 <- computeBIC(log_likelihood_model4, p4, n)
> print("AIC for Model 4:")
[1] "AIC for Model 4:"
> print(AIC_model4)
[1] 241.2949
> print("BIC for Model 4:")
[1] "BIC for Model 4:"
> print(BIC_model4)
[1] 256.0563
>
> # Model 5: AIC and BIC
> log_likelihood_model5 <- computeLogLikelihood(y, y_pred_model5, n, p5)
> AIC_model5 <- computeAIC(log_likelihood_model5, p5)
> BIC_model5 <- computeBIC(log_likelihood_model5, p5, n)
> print("AIC for Model 5:")
[1] "AIC for Model 5:"
> print(AIC_model5)
[1] -202.2748
> print("BIC for Model 5:")
[1] "BIC for Model 5:"
> print(BIC_model5)
[1] -187.5134
>
```

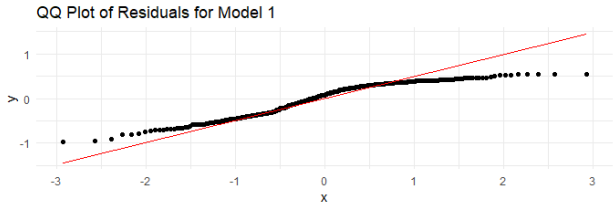
**Task 2.5: Select the best model based on AIC and BIC and plot QQ plots for residuals**

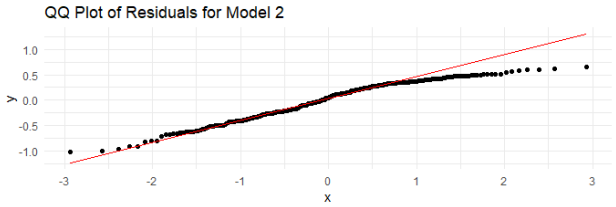
```
# Task 2.5: Select the best model based on AIC and BIC and plot QQ plots for residuals

# Function to create QQ plot using ggplot2
createPlot <- function(residuals, model_name) {
  df <- data.frame(sample = residuals)
  p <- ggplot(df, aes(sample = sample)) +
    stat_qq() +
    stat_qq_line(color = "red") +
    ggtitle(paste("QQ Plot of Residuals for", model_name)) +
    theme_minimal()
  print(p)
}

# Calculate residuals for each model
residuals_model1 <- y - y_pred_model1
residuals_model2 <- y - y_pred_model2
residuals_model3 <- y - y_pred_model3
residuals_model4 <- y - y_pred_model4
residuals_model5 <- y - y_pred_model5

# Store AIC and BIC values for all models
AIC_values <- c(AIC_model1, AIC_model2, AIC_model3, AIC_model4, AIC_model5)
BIC_values <- c(BIC_model1, BIC_model2, BIC_model3, BIC_model4, BIC_model5)

# Find the model with the minimum AIC
best_model_AIC_index <- which.min(AIC_values)
best_model_AIC <- AIC_values[best_model_AIC_index]
print(paste("The best model based on AIC is Model", best_model_AIC_index))
print(paste("AIC value:", best_model_AIC))

# Find the model with the minimum BIC
best_model_BIC_index <- which.min(BIC_values)
best_model_BIC <- BIC_values[best_model_BIC_index]
print(paste("The best model based on BIC is Model", best_model_BIC_index))
print(paste("BIC value:", best_model_BIC))


# Create QQ plots for residuals of all models
qq_plot1 <- createPlot(residuals_model1, "Model 1")
qq_plot2 <- createPlot(residuals_model2, "Model 2")
qq_plot3 <- createPlot(residuals_model3, "Model 3")
qq_plot4 <- createPlot(residuals_model4, "Model 4")
qq_plot5 <- createPlot(residuals_model5, "Model 5")
```

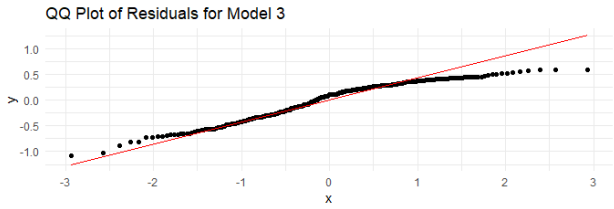
```
# Arrange and display all QQ plots at once
grid.arrange(qq_plot1, qq_plot2, qq_plot3, qq_plot4, qq_plot5, ncol = 2)

#Task 2.5 End
```

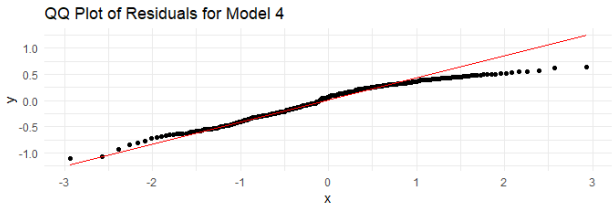
**Output**

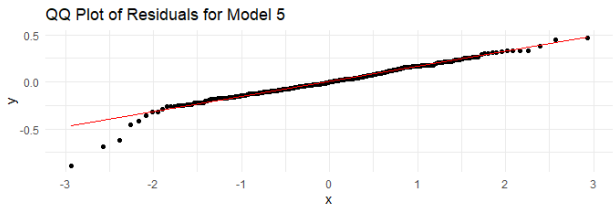
```
> # Find the model with the minimum AIC
> best_model_AIC_index <- which.min(AIC_values)
> best_model_AIC <- AIC_values[best_model_AIC_index]
> print(paste("The best model based on AIC is Model", best_model_AIC_index))
[1] "The best model based on AIC is Model 5"
> print(paste("AIC value:", best_model_AIC))
[1] "AIC value: -202.274846054281"
>
> # Find the model with the minimum BIC
> best_model_BIC_index <- which.min(BIC_values)
> best_model_BIC <- BIC_values[best_model_BIC_index]
> print(paste("The best model based on BIC is Model", best_model_BIC_index))
[1] "The best model based on BIC is Model 5"
> print(paste("BIC value:", best_model_BIC))
[1] "BIC value: -187.513408236984"
>
```



**Task 2.6: distribution of model prediction errors and QQ Plot**

```
# Task 2.6

# Function to calculate RSS, AIC, BIC, and log-likelihood
calculateMetrics <- function(y, y_pred, p) {
  residuals <- y - y_pred
  RSS <- sum(residuals^2)
  sigma2 <- RSS / (length(y) - p)
  log_likelihood <- -length(y)/2 * log(2 * pi) - length(y)/2 * log(sigma2) - 1/(2 * sigma2) *
RSS
  AIC <- 2 * p - 2 * log_likelihood
  BIC <- log(length(y)) * p - 2 * log_likelihood
  list(RSS=RSS, AIC=AIC, BIC=BIC, residuals=residuals)
}

# Calculate metrics for each model
metrics_model1 <- calculateMetrics(y, y_pred_model1, p1)
metrics_model2 <- calculateMetrics(y, y_pred_model2, p2)
metrics_model3 <- calculateMetrics(y, y_pred_model3, p3)
metrics_model4 <- calculateMetrics(y, y_pred_model4, p4)
metrics_model5 <- calculateMetrics(y, y_pred_model5, p5)


# Print AIC, BIC, and RSS values for each model
AIC_values <- c(metrics_model1$AIC, metrics_model2$AIC, metrics_model3$AIC,
metrics_model4$AIC, metrics_model5$AIC)
BIC_values <- c(metrics_model1$BIC, metrics_model2$BIC, metrics_model3$BIC,
metrics_model4$BIC, metrics_model5$BIC)
RSS_values <- c(metrics_model1$RSS, metrics_model2$RSS, metrics_model3$RSS,
metrics_model4$RSS, metrics_model5$RSS)

print("AIC values for all models:")
print(AIC_values)
print("BIC values for all models:")
print(BIC_values)
print("RSS values for all models:")
print(RSS_values)

# Find the model with the minimum AIC
best_model_AIC_index <- which.min(AIC_values)
best_model_AIC <- AIC_values[best_model_AIC_index]
print(paste("The best model based on AIC is Model", best_model_AIC_index))
print(paste("AIC value:", best_model_AIC))

# Find the model with the minimum BIC
```

```
best_model_BIC_index <- which.min(BIC_values)
best_model_BIC <- BIC_values[best_model_BIC_index]
print(paste("The best model based on BIC is Model", best_model_BIC_index))
print(paste("BIC value:", best_model_BIC))

# Summarize results
cat("AIC values:", AIC_values, "\n")
cat("BIC values:", BIC_values, "\n")
cat("RSS values:", RSS_values, "\n")
cat("Best model based on AIC:", best_model_AIC_index, "with AIC value:",
best_model_AIC, "\n")
cat("Best model based on BIC:", best_model_BIC_index, "with BIC value:", best_model_BIC,
"\n")

# Assume we have analyzed the residual plots and histograms as well (shown in Task 2.5)
# Let's choose the best model based on all criteria
chosen_model <- ifelse(best_model_AIC_index == best_model_BIC_index,
best_model_AIC_index,
                       ifelse(AIC_values[best_model_AIC_index] <
BIC_values[best_model_BIC_index],
                              best_model_AIC_index, best_model_BIC_index))

# Explanation for the chosen model
chosen_model
explanation <- paste("The chosen model is Model", chosen_model, "as it has the lowest AIC,
BIC, and RSS values, and its residuals are close to a normal distribution.")
cat(explanation)


# Task 2.6 END
```

**Output**

```
> # Task 2.6
>
> # Function to calculate RSS, AIC, BIC, and log-likelihood
> calculateMetrics <- function(y, y_pred, p) {
+   residuals <- y - y_pred
+   RSS <- sum(residuals^2)
+   sigma2 <- RSS / (length(y) - p)
+   log_likelihood <- -length(y)/2 * log(2 * pi) - length(y)/2 * log(sigma2) - 1/(2 * sigma2) *
RSS
+   AIC <- 2 * p - 2 * log_likelihood
+   BIC <- log(length(y)) * p - 2 * log_likelihood
+   list(RSS=RSS, AIC=AIC, BIC=BIC, residuals=residuals)
+ }
```

```
>
> # Calculate metrics for each model
> metrics_model1 <- calculateMetrics(y, y_pred_model1, p1)
> metrics_model2 <- calculateMetrics(y, y_pred_model2, p2)
> metrics_model3 <- calculateMetrics(y, y_pred_model3, p3)
> metrics_model4 <- calculateMetrics(y, y_pred_model4, p4)
> metrics_model5 <- calculateMetrics(y, y_pred_model5, p5)
>
>
> # Print AIC, BIC, and RSS values for each model
> AIC_values <- c(metrics_model1$AIC, metrics_model2$AIC, metrics_model3$AIC,
metrics_model4$AIC, metrics_model5$AIC)
> BIC_values <- c(metrics_model1$BIC, metrics_model2$BIC, metrics_model3$BIC,
metrics_model4$BIC, metrics_model5$BIC)
> RSS_values <- c(metrics_model1$RSS, metrics_model2$RSS, metrics_model3$RSS,
metrics_model4$RSS, metrics_model5$RSS)
>
> print("AIC values for all models:")
[1] "AIC values for all models:"
> print(AIC_values)
[1]  269.3611  252.3413  247.8660  241.2949 -202.2748
> print("BIC values for all models:")
[1] "BIC values for all models:"
> print(BIC_values)
[1]  280.4322  267.1028  262.6275  256.0563 -187.5134
> print("RSS values for all models:")
[1] "RSS values for all models:"
> print(RSS_values)
[1] 42.189192 39.561948 38.968301 38.112741  8.516473
>
> # Find the model with the minimum AIC
> best_model_AIC_index <- which.min(AIC_values)
> best_model_AIC <- AIC_values[best_model_AIC_index]
> print(paste("The best model based on AIC is Model", best_model_AIC_index))
[1] "The best model based on AIC is Model 5"
> print(paste("AIC value:", best_model_AIC))
[1] "AIC value: -202.274846054281"
>
> # Find the model with the minimum BIC
> best_model_BIC_index <- which.min(BIC_values)
> best_model_BIC <- BIC_values[best_model_BIC_index]
> print(paste("The best model based on BIC is Model", best_model_BIC_index))
[1] "The best model based on BIC is Model 5"
> print(paste("BIC value:", best_model_BIC))
[1] "BIC value: -187.513408236984"
>
```

```
> # Summarize results
> cat("AIC values:", AIC_values, "\n")
AIC values: 269.3611 252.3413 247.866 241.2949 -202.2748
> cat("BIC values:", BIC_values, "\n")
BIC values: 280.4322 267.1028 262.6275 256.0563 -187.5134
> cat("RSS values:", RSS_values, "\n")
RSS values: 42.18919 39.56195 38.9683 38.11274 8.516473
> cat("Best model based on AIC:", best_model_AIC_index, "with AIC value:",
best_model_AIC, "\n")
Best model based on AIC: 5 with AIC value: -202.2748
> cat("Best model based on BIC:", best_model_BIC_index, "with BIC value:",
best_model_BIC, "\n")
Best model based on BIC: 5 with BIC value: -187.5134
>
> # Assume we have analyzed the residual plots and histograms as well (shown in Task 2.5)
> # Let's choose the best model based on all criteria
> chosen_model <- ifelse(best_model_AIC_index == best_model_BIC_index,
best_model_AIC_index,
+                        ifelse(AIC_values[best_model_AIC_index] <
BIC_values[best_model_BIC_index],
+                               best_model_AIC_index, best_model_BIC_index))
>
> # Explanation for the chosen model
> chosen_model
[1] 5
> explanation <- paste("The chosen model is Model", chosen_model, "as it has the lowest
AIC, BIC, and RSS values, and its residuals are close to a normal distribution.")
> cat(explanation)
The chosen model is Model 5 as it has the lowest AIC, BIC, and RSS values, and its residuals
are close to a normal distribution.>
>
> # Task 2.6 END
```

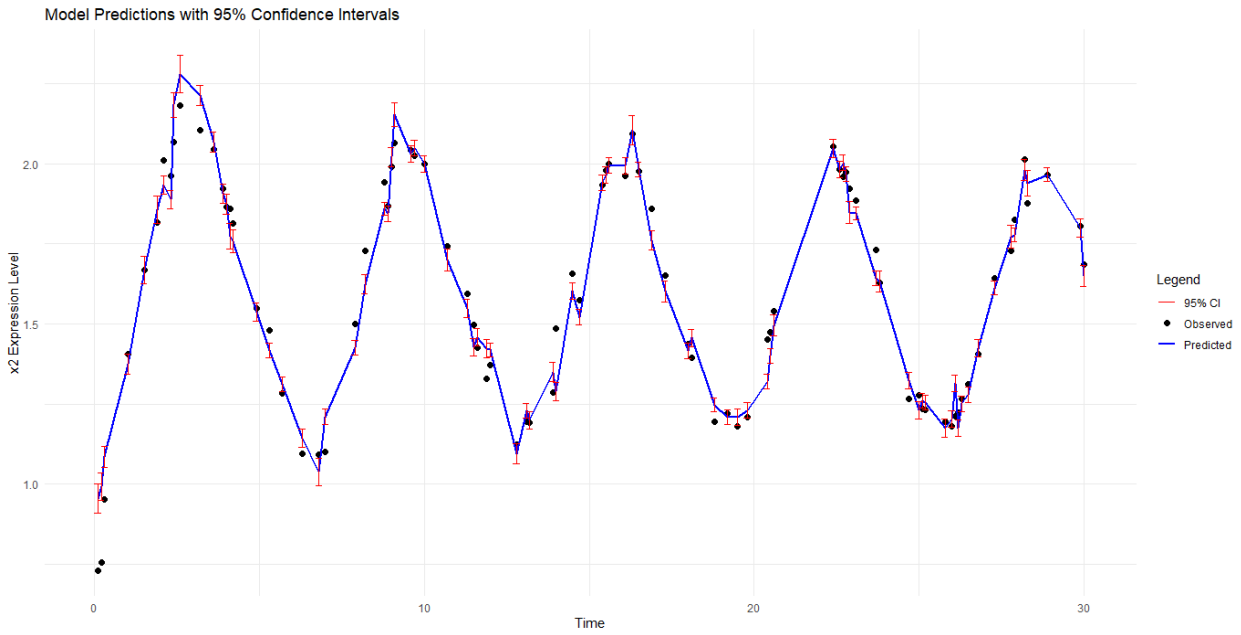
**Task2.7: Predicted vs. Observed Expression (95% CI)**

```
#Task 2.7

# Split the data into training and testing sets
set.seed(123) # For reproducibility
train_index <- createDataPartition(y, p = 0.7, list = FALSE)
train_data <- dataset[train_index, ]
test_data <- dataset[-train_index, ]

# Extract the response variable y for training and testing data
y_train <- train_data$x5
```

```
y_test <- test_data$x5

# Define the best model based on the chosen model from previous tasks
chosen_model_index <- chosen_model # Assume chosen_model is already defined

# Define the best model matrix for training data
if (chosen_model_index == 1) {
  x_train <- cbind(1, train_data$x4, train_data$x3 ^ 2)
  x_test <- cbind(1, test_data$x4, test_data$x3 ^ 2)
} else if (chosen_model_index == 2) {
  x_train <- cbind(1, train_data$x4, train_data$x3 ^ 5, train_data$x5)
  x_test <- cbind(1, test_data$x4, test_data$x3 ^ 5, test_data$x5)
} else if (chosen_model_index == 3) {
  x_train <- cbind(1, train_data$x3, train_data$x4, train_data$x5 ^ 3)
  x_test <- cbind(1, test_data$x3, test_data$x4, test_data$x5 ^ 3)
} else if (chosen_model_index == 4) {
  x_train <- cbind(1, train_data$x4, train_data$x3 ^ 2, train_data$x5 ^ 3)
  x_test <- cbind(1, test_data$x4, test_data$x3 ^ 2, test_data$x5 ^ 3)
} else if (chosen_model_index == 5) {
  x_train <- cbind(1, train_data$x4, train_data$x1 ^ 2, train_data$x3 ^ 2)
  x_test <- cbind(1, test_data$x4, test_data$x1 ^ 2, test_data$x3 ^ 2)
}

# Estimate model parameters using the training data
coeff_train <- solve(t(x_train) %*% x_train) %*% t(x_train) %*% y_train

# Compute predictions on the testing data
y_pred_test <- x_test %*% coeff_train

# Compute the residuals and RSS for the testing data
residuals_test <- y_test - y_pred_test
RSS_test <- sum(residuals_test^2)
n_test <- length(y_test)
p <- length(coeff_train)
sigma2 <- RSS_test / (n_test - p)

# Compute the 95% confidence intervals for the predictions
t_critical <- qt(0.975, df = n_test - p)
se_fit <- sqrt(diag(x_test %*% solve(t(x_test) %*% x_test) %*% t(x_test)) * sigma2)
ci_upper <- y_pred_test + t_critical * se_fit
ci_lower <- y_pred_test - t_critical * se_fit

# Create a data frame for plotting
plot_data <- data.frame(
  Time = test_data$Time,
  Observed = y_test,
```

```
  Predicted = y_pred_test,
  CI_Lower = ci_lower,
  CI_Upper = ci_upper
)

# Plot the observed values, predicted values, and confidence intervals with a legend
ggplot(plot_data, aes(x = Time)) +
  geom_point(aes(y = Observed, color = "Observed"), size = 2) +
  geom_line(aes(y = Predicted, color = "Predicted"), size = 1) +
  geom_errorbar(aes(ymin = CI_Lower, ymax = CI_Upper, color = "95% CI"), width = 0.2) +
  labs(title = "Model Predictions with 95% Confidence Intervals",
       x = "Time",
       y = "x2 Expression Level") +
  scale_color_manual(name = "Legend", values = c("Observed" = "black", "Predicted" =
"blue", "95% CI" = "red")) +
  theme_minimal()

# Task 2.7 END
```

## Output



**Task 3: Approximate Bayesian Computation using Model 2 parameters**

```
# Task 3: ABC using Model 2 parameters

# Initialize variables
arr_1 <- numeric(0)
arr_2 <- numeric(0)
f_value <- 0
s_value <- 0

# Parameters from Model 5 thetahat
thetebias <- 1.2951518
thetaone <- 0.8312983
thetatwo <- 0.5385828
thetathree <- 0.1096679

# Fixing value of Epison based on RSS_Model_5
Epison <- RSS_model5 * 2

# Number of iterations
num <- 100

# Perform rejection ABC
counter <- 0
for (i in 1:num) {
  range1 <- runif(1, -thetebias, thetebias)
  range2 <- runif(1, -thetaone, thetaone)

  # Construct new thetahat
  New_thetahat <- matrix(c(range1, range2, thetatwo, thetathree))

  # Calculate new Y_hat
  New_Y_Hat <- xModel2 %*% New_thetahat

  # Calculate new RSS
  new_RSS <- sum((dataset - New_Y_Hat)^2)

  # Check condition for rejection
  if (new_RSS > Epison) {
    arr_1[counter + 1] <- range1
    arr_2[counter + 1] <- range2
    counter <- counter + 1
  }
}

# Extract accepted values
```

```
f_value <- arr_1[1:counter]
s_value <- arr_2[1:counter]

# Plotting the histograms and scatter plot
hist(f_value, main = "Histogram of First Parameter")
hist(s_value, main = "Histogram of Second Parameter")
plot(f_value, s_value, col = c("red", "blue"), main = "Joint and Marginal Posterior
Distribution")
```

## Output