GROUP ASSIGNMENT

TECHONOLOGY PARK MALAYSIA

CT127-3-2-PFDA

PROGRAMMING FOR DATA ANALYSIS

APU2F2402CS(AI)/APD2F2402CS(AI)

HAND OUT DATE:	**25 MARCH 2024**
HAND IN DATE:	**3 JUNE 2024**
WEIGHTAGE:	**50%**

INSTRUCTION TO CANDIDATES:

1. **Students are advised to underpin their answer with the use of references (cited using the American Psychological Association (APA) Referencing).**
2. **Late submission will be awarded zero (0) unless Extenuating Circumstances (EC) are upheld.**
3. **Cases of plagiarism will be penalized.**
4. **Submit the assignment to APU Learning Management System.**
5. **You must obtain 50% overall to pass this module.**

Group No: 61
- **Jeff Hong Chee Hin -TP074560**
- **Lee De Xian - TP068265**
- **Ooi Yin Yao - TP074427**
- **Angelina Leanore – TP072929**

Table of Contents

1.0 Introduction

There are 28 columns and 100,000 rows in this dataset. An individual's credit score, debt status, and numerous financial capacities are listed in these columns. They are ID, Customer ID, Month, Name, Age, SSN (Social Security Number), Occupation, Annual Income, Monthly Inhand Salary, Number of Bank Account, Number of Credit Card, Interest Rate, Number of Loan, Type of Loan, Number of Days Delayed from the Payment Date, Number of Delayed Payment, Percentage Change in Credit Card Limit, Number of Credit Card Inquires, Classification of the Mix of Credits, Outstanding Debt, Utilisation Ratio of Credit Card, Age of Credit History of the Person, Payment of Minimum Amount, Monthly EMI Payments (Equated Monthly Instalment), Amount Invested Monthly, Payment Behaviour, Monthly Balance and Credit Score. These columns are split up into servely different categories. ID, Customer ID, Age, SSN, and Occupation are in the Individual's Personal Information category. Individual Economic Capability is a further category that includes Monthly Balance, Annual Income, Monthly Inhand Salary, Monthly EMI Payments, Monthly Investly Amount, Payment Behaviour, and Payment of Minimum Amount. The categories of Loan Status are Loan Number, Loan Type, and Outstanding Debt. The remainder falls within the category of economic data. Various kinds of data are represented by these categories. While Loan Status and Economic Information contain both discrete and categorical data, Individual Economic Capability is continuous data. These two types of data will give rise to two different data analysis models.

2.0 Hypothesis and Objective

Hypothesis:

The credit scores of bank customers are influenced by number of delayed payment, annual income, number of loans and number of credit cards.

Objective:

To investigate how the number of credit cards can affect each person's credit score.

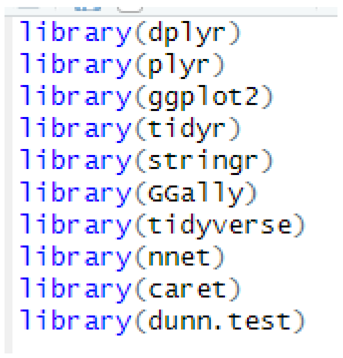
To investigate the impact of delayed payments to credit score.

To investigate the relationship between annual income and a person's credit score.

To investigate the relationship between the number of loans and a person's credit score.

3.0 Data Import / Exploration / Cleaning / Pre-processing

3.1 Install Package

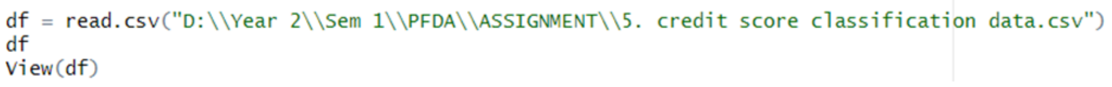


```
library(dplyr)
library(plyr)
library(ggplot2)
library(tidyr)
library(stringr)
library(GGally)
library(tidyverse)
library(nnet)
library(caret)
library(dunn.test)
```

Figure 1: R libraries

This analysis relies on 10 R libraries: dplyr, plyr, ggplot2, tidyr, stringr, GGally, tidyverse, nnet, caret, and dunn.test. Each has a specific function: dplyr and plyr for data manipulation, ggplot2 for visualization, tidyr for data reshaping, stringr for string handling, GGally for advanced plots, tidyverse as a collection of data science packages, nnet for neural networks, caret for machine learning, and lastly dunn.test for post-hoc Kruskal-Wallis analysis.

3.2 Data Import



```
df = read.csv("D:\\Year 2\\Sem 1\\PFDA\\ASSIGNMENT\\5. credit score classification data.csv")
df
View(df)
```

Figure 2: Import csv file

The file name "D:\\Year 2\\Sem 1\\PFDA\\ASSIGNMENT\\5. credit score classification data.csv" will have the data moved and imported from the drive with the.csv file extension.

3.3 Data Exploration

```
# check missing values
plot_missing(df)
```

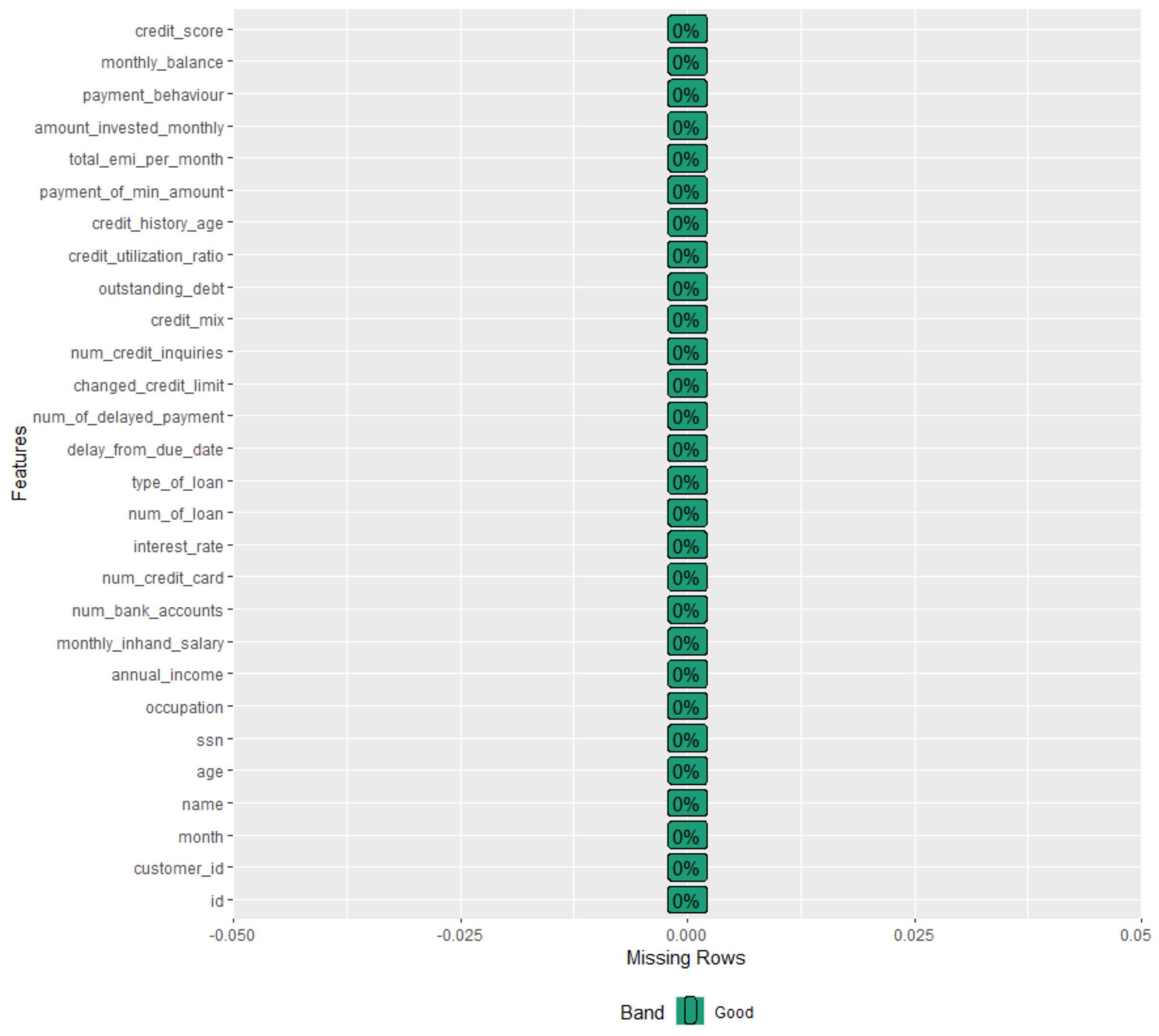
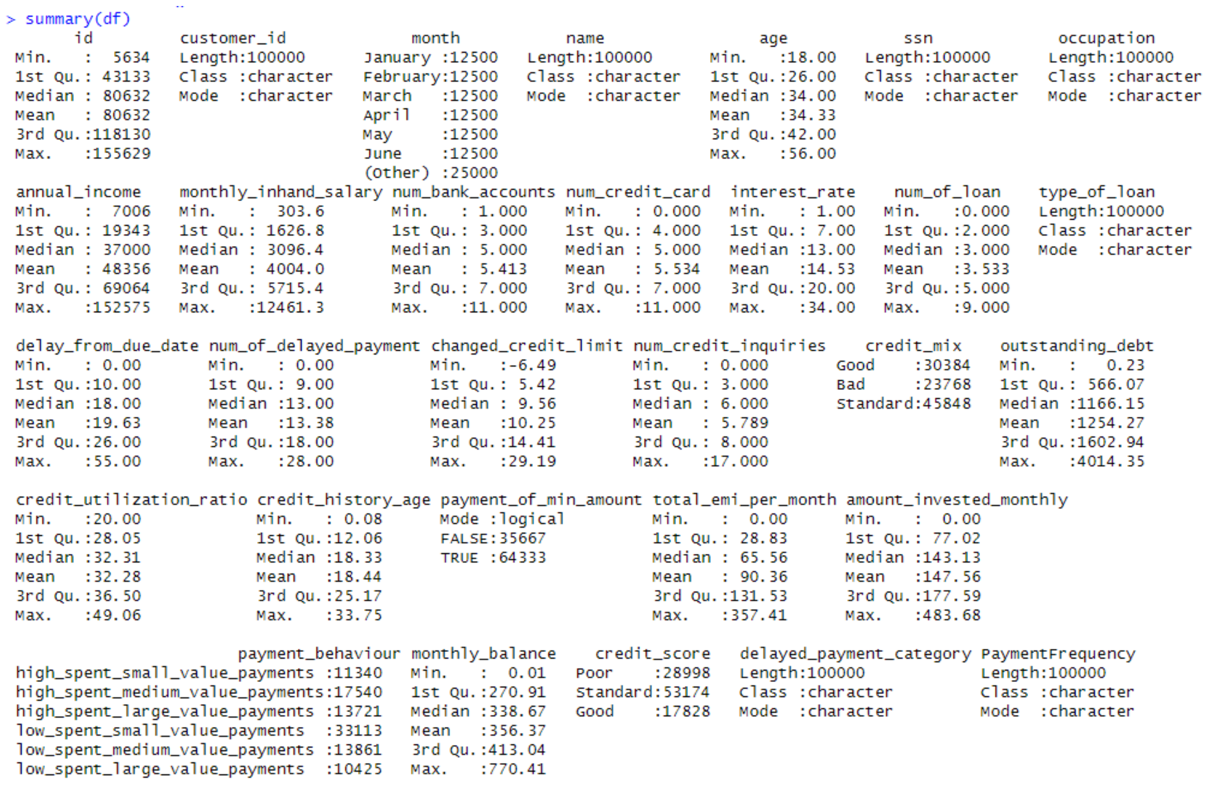


Figure 3: Missing Plot and the codes

The missing plot is to check all the missing rows of the dataset to justify that the dataset does not have any null values left in each of the variables. It will show as 0% in the plot of each variable in the dataset.

Data Summary

```
#Data Summary
summary(df)
```

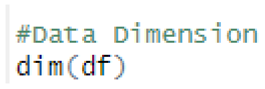


```
> summary(df)
       id          customer_id              month              name                age              ssn             occupation       
 Min.   :  5634   Length:100000      January :12500   Length:100000      Min.   :18.00   Length:100000      Length:100000     
 1st Qu.: 43133   Class :character   February:12500   Class :character   1st Qu.:26.00   Class :character   Class :character  
 Median : 80632   Mode  :character   March   :12500   Mode  :character   Median :34.00   Mode  :character   Mode  :character  
 Mean   : 80632                      April   :12500                      Mean   :34.33                                        
 3rd Qu.:118130                      May     :12500                      3rd Qu.:42.00                                        
 Max.   :155629                      June    :12500                      Max.   :56.00                                        
                                     (Other) :25000                                                                           
 annual_income    monthly_inhand_salary num_bank_accounts num_credit_card  interest_rate    num_of_loan    type_of_loan      
 Min.   :  7006   Min.   :  303.6        Min.   : 1.000    Min.   : 0.000   Min.   : 1.00   Min.   :0.000   Length:100000     
 1st Qu.: 19343   1st Qu.: 1626.8        1st Qu.: 3.000    1st Qu.: 4.000   1st Qu.: 7.00   1st Qu.:2.000   Class :character  
 Median : 37000   Median : 3096.4        Median : 5.000    Median : 5.000   Median :13.00   Median :3.000   Mode  :character  
 Mean   : 48356   Mean   : 4004.0        Mean   : 5.413    Mean   : 5.534   Mean   :14.53   Mean   :3.533                     
 3rd Qu.: 69064   3rd Qu.: 5715.4        3rd Qu.: 7.000    3rd Qu.: 7.000   3rd Qu.:20.00   3rd Qu.:5.000                     
 Max.   :152575   Max.   :12461.3        Max.   :11.000    Max.   :11.000   Max.   :34.00   Max.   :9.000                     

 delay_from_due_date num_of_delayed_payment changed_credit_limit num_credit_inquiries    credit_mix     outstanding_debt
 Min.   : 0.00       Min.   : 0.00          Min.   :-6.49        Min.   : 0.000       Good    :30384   Min.   :   0.23  
 1st Qu.:10.00       1st Qu.: 9.00          1st Qu.: 5.42        1st Qu.: 3.000       Bad     :23768   1st Qu.: 566.07  
 Median :18.00       Median :13.00          Median : 9.56        Median : 6.000       Standard:45848   Median :1166.15  
 Mean   :19.63       Mean   :13.38          Mean   :10.25        Mean   : 5.789                        Mean   :1254.27  
 3rd Qu.:26.00       3rd Qu.:18.00          3rd Qu.:14.41        3rd Qu.: 8.000                        3rd Qu.:1602.94  
 Max.   :55.00       Max.   :28.00          Max.   :29.19        Max.   :17.000                        Max.   :4014.35  

 credit_utilization_ratio credit_history_age payment_of_min_amount total_emi_per_month amount_invested_monthly
 Min.   :20.00            Min.   : 0.08      Mode :logical         Min.   :  0.00      Min.   :  0.00         
 1st Qu.:28.05            1st Qu.:12.06      FALSE:35667           1st Qu.: 28.83      1st Qu.: 77.02         
 Median :32.31            Median :18.33      TRUE :64333           Median : 65.56      Median :143.13         
 Mean   :32.28            Mean   :18.44                            Mean   : 90.36      Mean   :147.56         
 3rd Qu.:36.50            3rd Qu.:25.17                            3rd Qu.:131.53      3rd Qu.:177.59         
 Max.   :49.06            Max.   :33.75                            Max.   :357.41      Max.   :483.68         

                        payment_behaviour monthly_balance    credit_score   delayed_payment_category PaymentFrequency  
 high_spent_small_value_payments :11340   Min.   :  0.01   Poor    :28998   Length:100000            Length:100000     
 high_spent_medium_value_payments:17540   1st Qu.:270.91   Standard:53174   Class :character         Class :character  
 high_spent_large_value_payments :13721   Median :338.67   Good    :17828   Mode  :character         Mode  :character  
 low_spent_small_value_payments  :33113   Mean   :356.37                                                               
 low_spent_medium_value_payments :13861   3rd Qu.:413.04                                                               
 low_spent_large_value_payments  :10425   Max.   :770.41                                                               
```

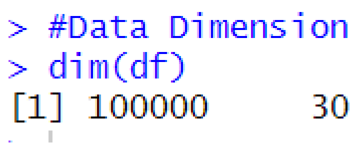
Figure 4: Summary of the dataset

The summary of the data frame 'df' provides statistical insight into 100,000 customer records from credit score classification data.

Data Dimension



```
#Data Dimension
dim(df)
```

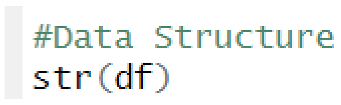


```
> #Data Dimension
> dim(df)
[1] 100000     30
```

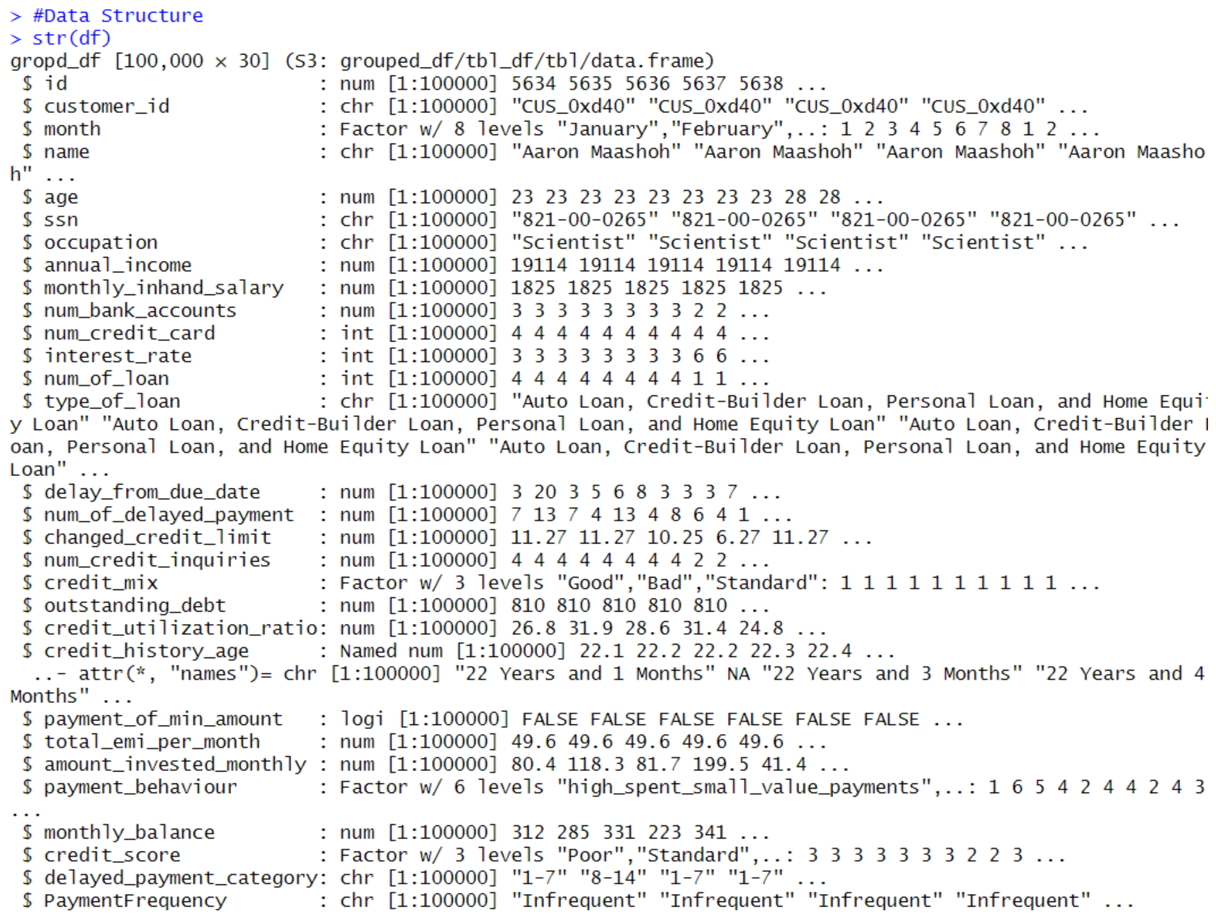
Figure 5: Dimension of the dataset

This function provides the dimension of the data frame. The result '100000 30' indicates that the data frame contains 100,000 rows and 30 columns.

Data Structure



```
#Data Structure
str(df)
```

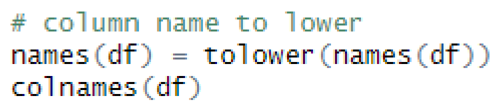


```
> #Data Structure
> str(df)
gropd_df [100,000 × 30] (S3: grouped_df/tbl_df/tbl/data.frame)
 $ id                      : num [1:100000] 5634 5635 5636 5637 5638 ...
 $ customer_id             : chr [1:100000] "CUS_0xd40" "CUS_0xd40" "CUS_0xd40" "CUS_0xd40" ...
 $ month                   : Factor w/ 8 levels "January","February",..: 1 2 3 4 5 6 7 8 1 2 ...
 $ name                    : chr [1:100000] "Aaron Maashoh" "Aaron Maashoh" "Aaron Maashoh" "Aaron Maasho
h" ...
 $ age                     : num [1:100000] 23 23 23 23 23 23 23 23 28 28 ...
 $ ssn                     : chr [1:100000] "821-00-0265" "821-00-0265" "821-00-0265" "821-00-0265" ...
 $ occupation              : chr [1:100000] "Scientist" "Scientist" "Scientist" "Scientist" ...
 $ annual_income           : num [1:100000] 19114 19114 19114 19114 19114 ...
 $ monthly_inhand_salary   : num [1:100000] 1825 1825 1825 1825 1825 ...
 $ num_bank_accounts       : num [1:100000] 3 3 3 3 3 3 3 3 2 2 ...
 $ num_credit_card         : int [1:100000] 4 4 4 4 4 4 4 4 4 4 ...
 $ interest_rate           : int [1:100000] 3 3 3 3 3 3 3 3 6 6 ...
 $ num_of_loan             : int [1:100000] 4 4 4 4 4 4 4 4 1 1 ...
 $ type_of_loan            : chr [1:100000] "Auto Loan, Credit-Builder Loan, Personal Loan, and Home Equit
y Loan" "Auto Loan, Credit-Builder Loan, Personal Loan, and Home Equity Loan" "Auto Loan, Credit-Builder L
oan, Personal Loan, and Home Equity Loan" "Auto Loan, Credit-Builder Loan, Personal Loan, and Home Equity
Loan" ...
 $ delay_from_due_date     : num [1:100000] 3 20 3 5 6 8 3 3 3 7 ...
 $ num_of_delayed_payment  : num [1:100000] 7 13 7 4 13 4 8 6 4 1 ...
 $ changed_credit_limit    : num [1:100000] 11.27 11.27 10.25 6.27 11.27 ...
 $ num_credit_inquiries    : num [1:100000] 4 4 4 4 4 4 4 4 2 2 ...
 $ credit_mix              : Factor w/ 3 levels "Good","Bad","Standard": 1 1 1 1 1 1 1 1 1 1 ...
 $ outstanding_debt        : num [1:100000] 810 810 810 810 810 ...
 $ credit_utilization_ratio: num [1:100000] 26.8 31.9 28.6 31.4 24.8 ...
 $ credit_history_age      : Named num [1:100000] 22.1 22.2 22.2 22.3 22.4 ...
  ..- attr(*, "names")= chr [1:100000] "22 Years and 1 Months" NA "22 Years and 3 Months" "22 Years and 4
Months" ...
 $ payment_of_min_amount   : logi [1:100000] FALSE FALSE FALSE FALSE FALSE FALSE ...
 $ total_emi_per_month     : num [1:100000] 49.6 49.6 49.6 49.6 49.6 ...
 $ amount_invested_monthly : num [1:100000] 80.4 118.3 81.7 199.5 41.4 ...
 $ payment_behaviour       : Factor w/ 6 levels "high_spent_small_value_payments",..: 1 6 5 4 2 4 4 2 4 3
...
 $ monthly_balance         : num [1:100000] 312 285 331 223 341 ...
 $ credit_score            : Factor w/ 3 levels "Poor","Standard",..: 3 3 3 3 3 3 3 2 2 3 ...
 $ delayed_payment_category: chr [1:100000] "1-7" "8-14" "1-7" "1-7" ...
 $ PaymentFrequency        : chr [1:100000] "Infrequent" "Infrequent" "Infrequent" "Infrequent" ...
```

Figure 6: Structure of the dataset

since this data frame has 100,000 rows and 30 columns, this str is to indicate that the column includes various data types such as integers(int), numeric(num), characters(chr), and factors (Factor).

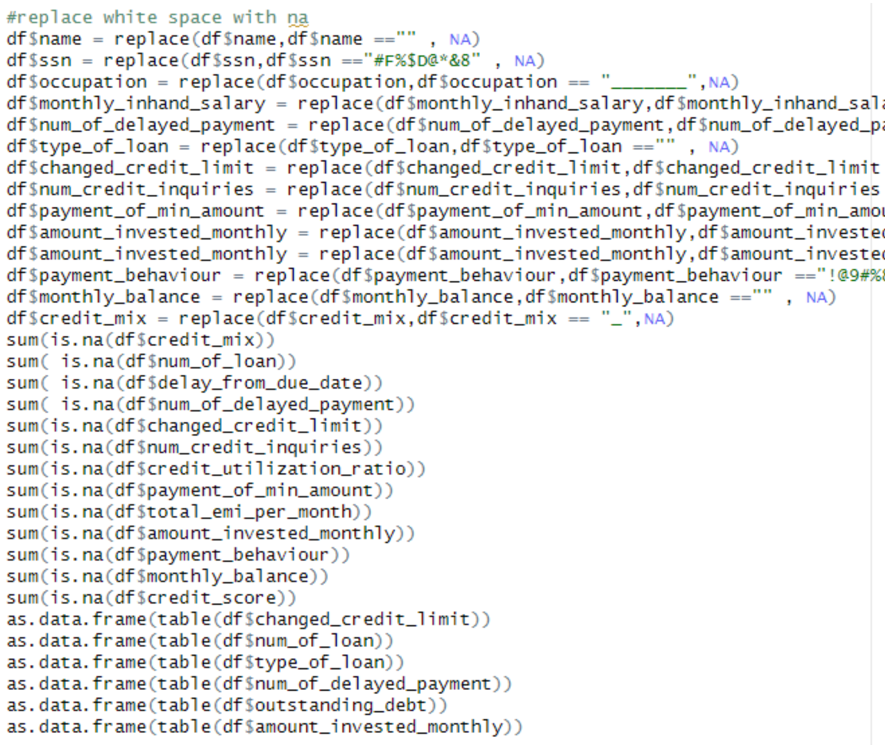
Data Cleaning



```
# column name to lower
names(df) = tolower(names(df))
colnames(df)
```

Figure 7: lowercase column name

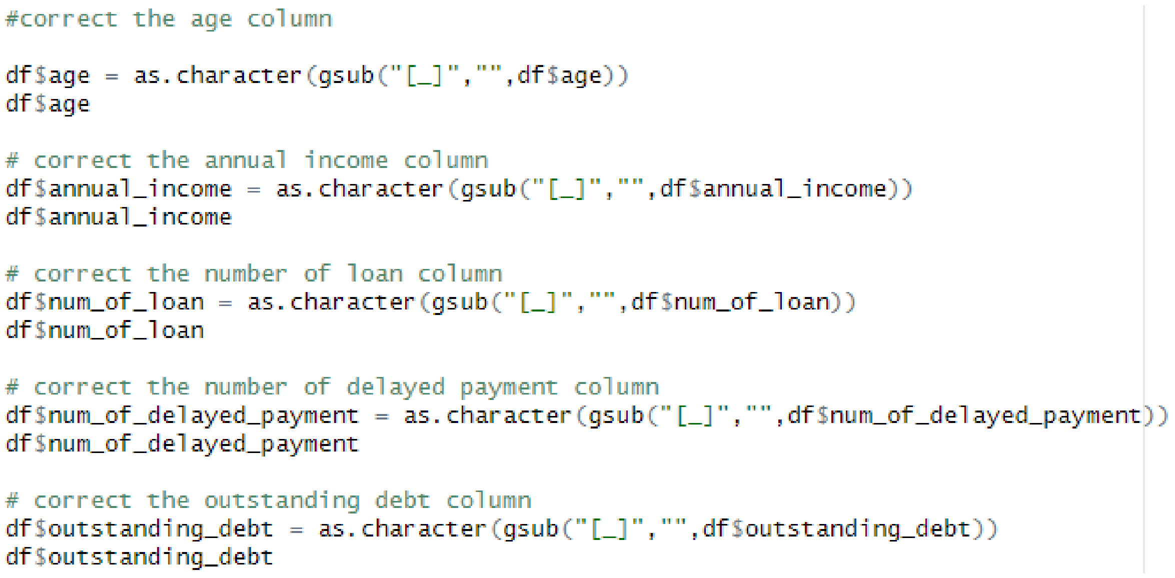
The code is to change the dataset column name from uppercase to lowercase, to brings convenience to the future coding.



```r
#replace white space with na
df$name = replace(df$name,df$name =="" , NA)
df$ssn = replace(df$ssn,df$ssn =="#F%$D@*&8" , NA)
df$occupation = replace(df$occupation,df$occupation == "_______",NA)
df$monthly_inhand_salary = replace(df$monthly_inhand_salary,df$monthly_inhand_salary =="" , NA)
df$num_of_delayed_payment = replace(df$num_of_delayed_payment,df$num_of_delayed_payment =="" , NA)
df$type_of_loan = replace(df$type_of_loan,df$type_of_loan =="" , NA)
df$changed_credit_limit = replace(df$changed_credit_limit,df$changed_credit_limit =="_" , NA)
df$num_credit_inquiries = replace(df$num_credit_inquiries,df$num_credit_inquiries =="" , NA)
df$payment_of_min_amount = replace(df$payment_of_min_amount,df$payment_of_min_amount =="NM" , NA)
df$amount_invested_monthly = replace(df$amount_invested_monthly,df$amount_invested_monthly =="" , NA)
df$amount_invested_monthly = replace(df$amount_invested_monthly,df$amount_invested_monthly =="__10000__" , NA)
df$payment_behaviour = replace(df$payment_behaviour,df$payment_behaviour =="!@9#%8" , NA)
df$monthly_balance = replace(df$monthly_balance,df$monthly_balance =="" , NA)
df$credit_mix = replace(df$credit_mix,df$credit_mix == "_",NA)
sum(is.na(df$credit_mix))
sum( is.na(df$num_of_loan))
sum( is.na(df$delay_from_due_date))
sum( is.na(df$num_of_delayed_payment))
sum(is.na(df$changed_credit_limit))
sum(is.na(df$num_credit_inquiries))
sum(is.na(df$credit_utilization_ratio))
sum(is.na(df$payment_of_min_amount))
sum(is.na(df$total_emi_per_month))
sum(is.na(df$amount_invested_monthly))
sum(is.na(df$payment_behaviour))
sum(is.na(df$monthly_balance))
sum(is.na(df$credit_score))
as.data.frame(table(df$changed_credit_limit))
as.data.frame(table(df$num_of_loan))
as.data.frame(table(df$type_of_loan))
as.data.frame(table(df$num_of_delayed_payment))
as.data.frame(table(df$outstanding_debt))
as.data.frame(table(df$amount_invested_monthly))
```

Figure 8: replace NA values

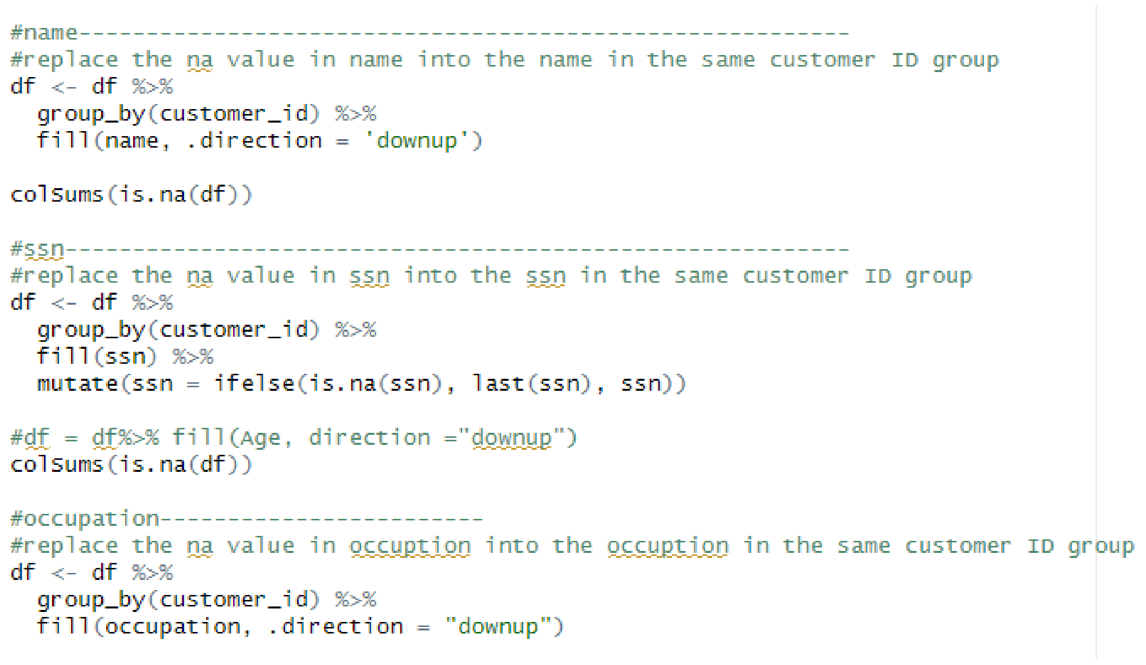
This code is to replace the columns of blank white space with NA values which makes it easier for the later process of data cleaning.



```r
#correct the age column

df$age = as.character(gsub("[_]","",df$age))
df$age

# correct the annual income column
df$annual_income = as.character(gsub("[_]","",df$annual_income))
df$annual_income

# correct the number of loan column
df$num_of_loan = as.character(gsub("[_]","",df$num_of_loan))
df$num_of_loan

# correct the number of delayed payment column
df$num_of_delayed_payment = as.character(gsub("[_]","",df$num_of_delayed_payment))
df$num_of_delayed_payment

# correct the outstanding debt column
df$outstanding_debt = as.character(gsub("[_]","",df$outstanding_debt))
df$outstanding_debt
```

Figure 9: remove the symbol of "_"

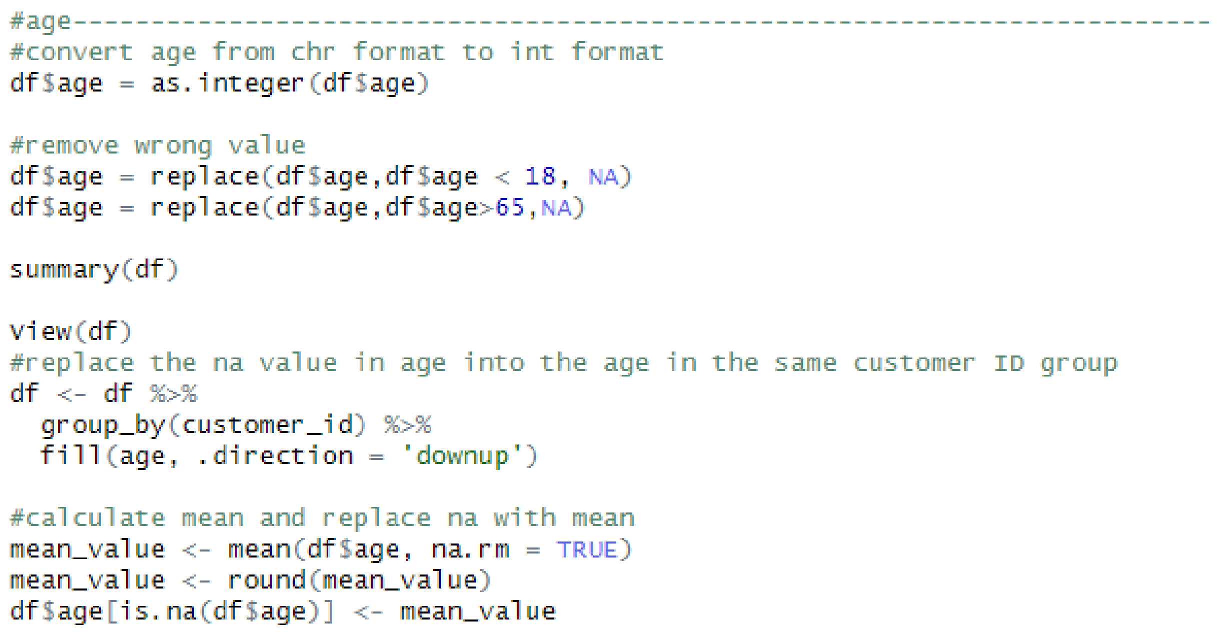
This process of data cleaning is to remove the column names that contain the "_" symbol. This process is just to remove noises.



```
#name--------------------------------------------------------------
#replace the na value in name into the name in the same customer ID group
df <- df %>%
  group_by(customer_id) %>%
  fill(name, .direction = 'downup')

colSums(is.na(df))

#ssn---------------------------------------------------------------
#replace the na value in ssn into the ssn in the same customer ID group
df <- df %>%
  group_by(customer_id) %>%
  fill(ssn) %>%
  mutate(ssn = ifelse(is.na(ssn), last(ssn), ssn))

#df = df%>% fill(Age, direction ="downup")
colSums(is.na(df))

#occupation------------------------
#replace the na value in occuption into the occuption in the same customer ID group
df <- df %>%
  group_by(customer_id) %>%
  fill(occupation, .direction = "downup")
```

Figure 10: replace the NA values with the same group data

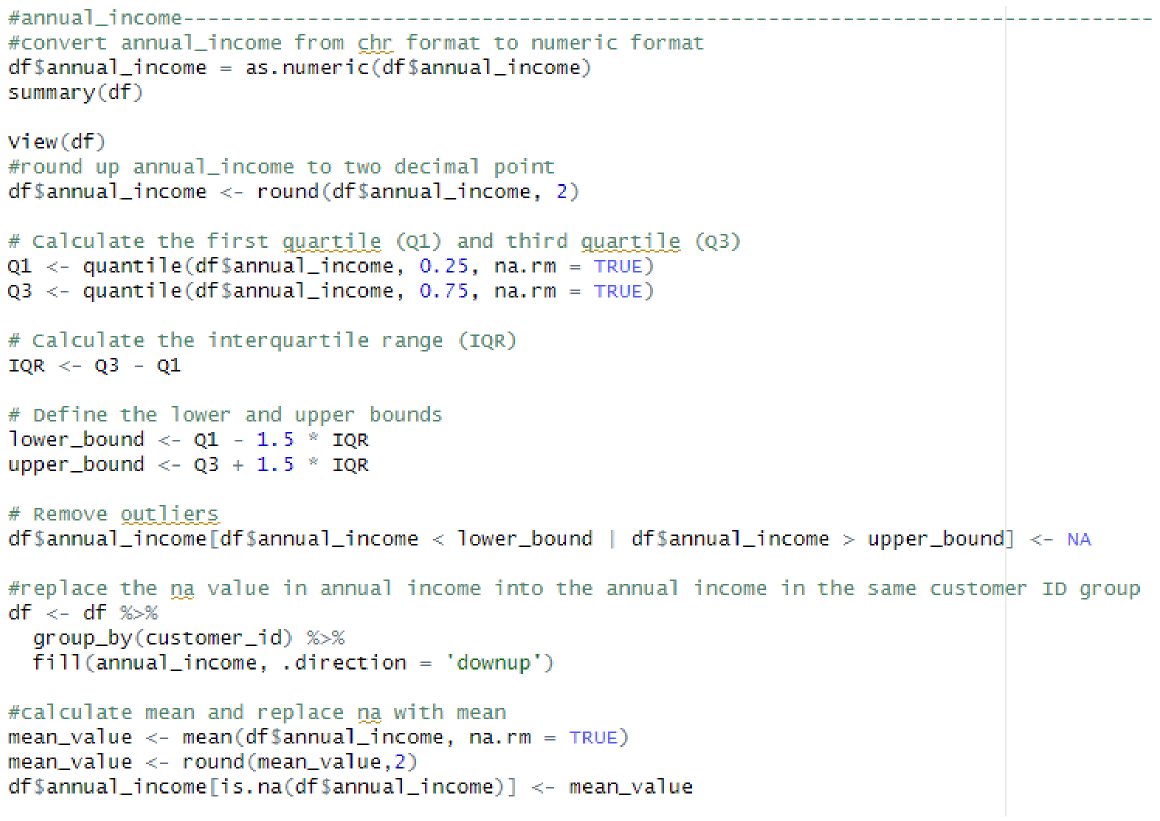
The process is to replace the NA values based on same customer ID group and check the columns if they're replaced.



```
#age---------------------------------------------------------------------
#convert age from chr format to int format
df$age = as.integer(df$age)

#remove wrong value
df$age = replace(df$age,df$age < 18, NA)
df$age = replace(df$age,df$age>65,NA)

summary(df)

View(df)
#replace the na value in age into the age in the same customer ID group
df <- df %>%
  group_by(customer_id) %>%
  fill(age, .direction = 'downup')

#calculate mean and replace na with mean
mean_value <- mean(df$age, na.rm = TRUE)
mean_value <- round(mean_value)
df$age[is.na(df$age)] <- mean_value
```

Figure 11: covert format and remove the wrong value

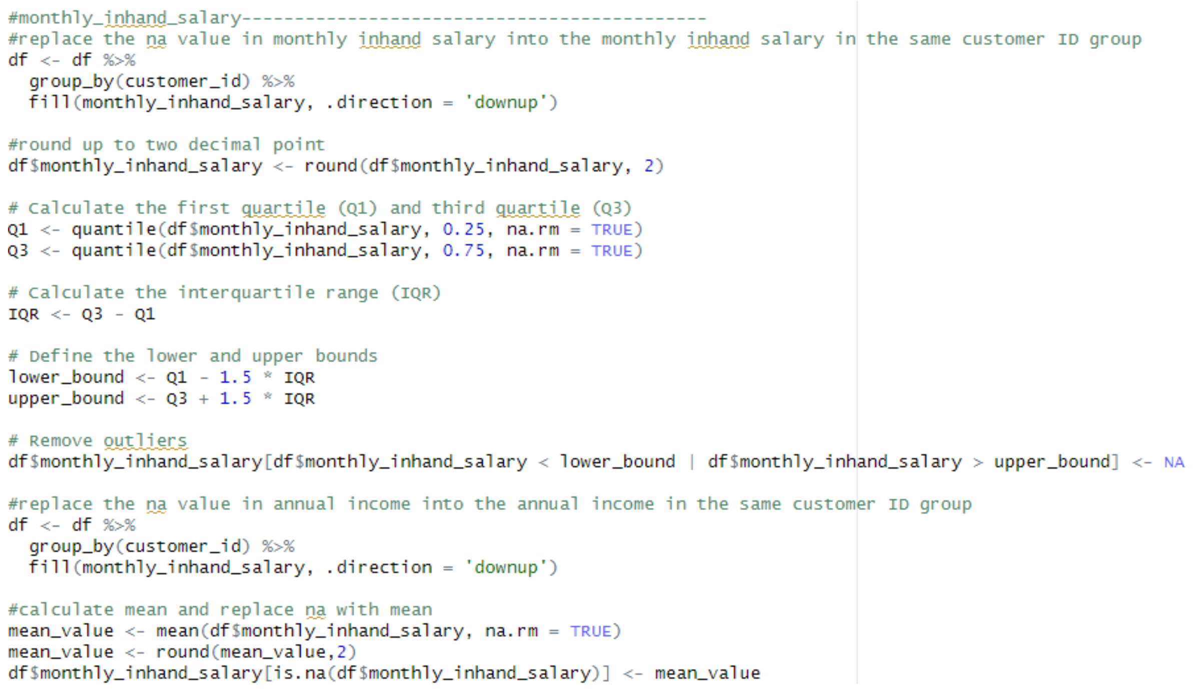
During this process, the class age is converted from character into integer format, and removing the wrong values with NA to avoid an age that does not make sense. Calculate the mean and age and replace the NA with the mean values.



```
#annual_income-------------------------------------------------------------------------------
#convert annual_income from chr format to numeric format
df$annual_income = as.numeric(df$annual_income)
summary(df)

View(df)
#round up annual_income to two decimal point
df$annual_income <- round(df$annual_income, 2)

# Calculate the first quartile (Q1) and third quartile (Q3)
Q1 <- quantile(df$annual_income, 0.25, na.rm = TRUE)
Q3 <- quantile(df$annual_income, 0.75, na.rm = TRUE)

# Calculate the interquartile range (IQR)
IQR <- Q3 - Q1

# Define the lower and upper bounds
lower_bound <- Q1 - 1.5 * IQR
upper_bound <- Q3 + 1.5 * IQR

# Remove outliers
df$annual_income[df$annual_income < lower_bound | df$annual_income > upper_bound] <- NA

#replace the na value in annual income into the annual income in the same customer ID group
df <- df %>%
  group_by(customer_id) %>%
  fill(annual_income, .direction = 'downup')

#calculate mean and replace na with mean
mean_value <- mean(df$annual_income, na.rm = TRUE)
mean_value <- round(mean_value,2)
df$annual_income[is.na(df$annual_income)] <- mean_value
```

Figure 12: annual income

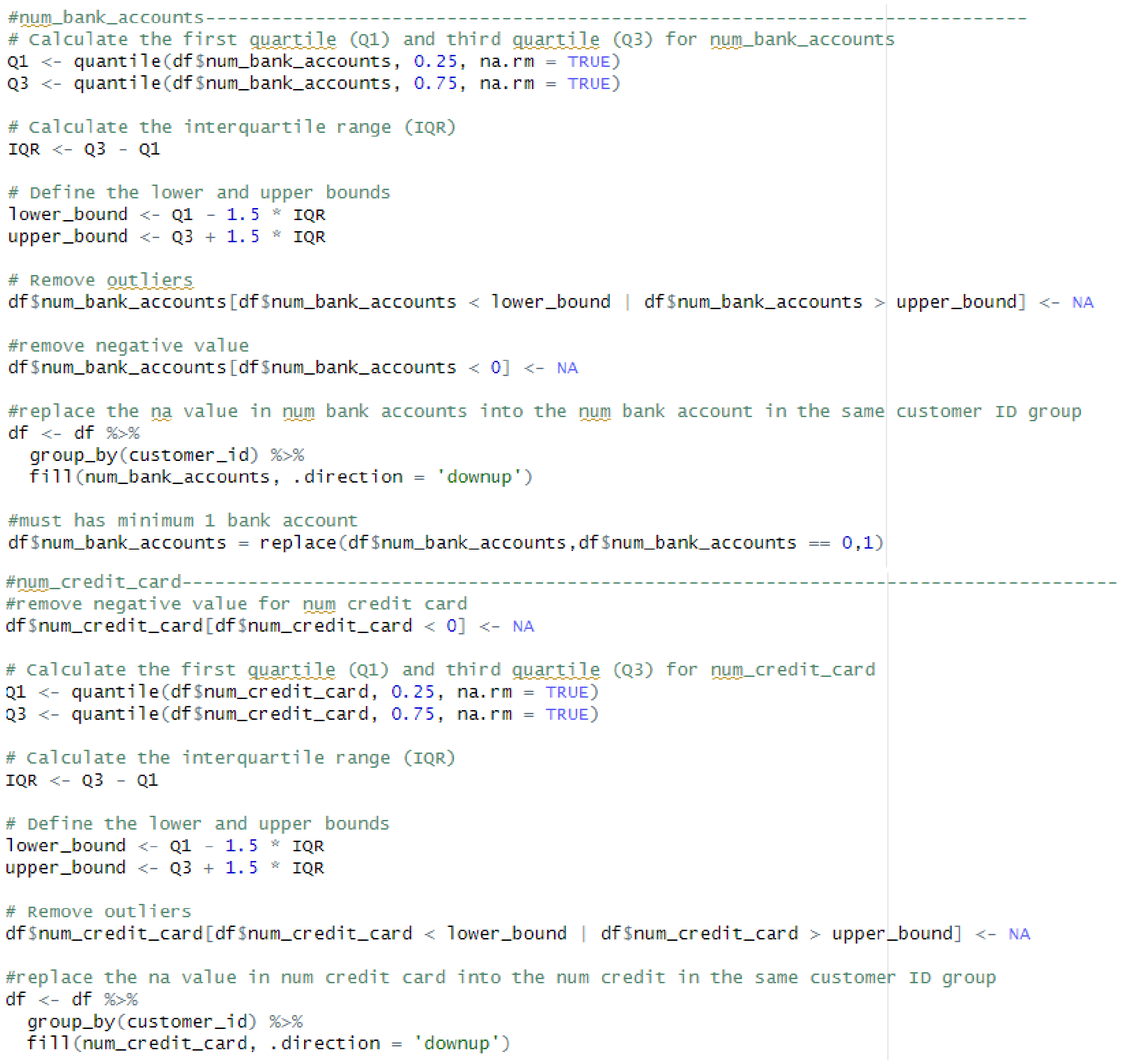
The annual income has been converted from character to integer format, making the values 2 decimal points. Calculating the first and third quartile and interquartile range, defining the lower boundary and upper boundary. The outliers is removed and the mean is calculated to replace the NA values.



```
#monthly_inhand_salary--------------------------------------------
#replace the na value in monthly inhand salary into the monthly inhand salary in the same customer ID group
df <- df %>%
  group_by(customer_id) %>%
  fill(monthly_inhand_salary, .direction = 'downup')

#round up to two decimal point
df$monthly_inhand_salary <- round(df$monthly_inhand_salary, 2)

# Calculate the first quartile (Q1) and third quartile (Q3)
Q1 <- quantile(df$monthly_inhand_salary, 0.25, na.rm = TRUE)
Q3 <- quantile(df$monthly_inhand_salary, 0.75, na.rm = TRUE)

# Calculate the interquartile range (IQR)
IQR <- Q3 - Q1

# Define the lower and upper bounds
lower_bound <- Q1 - 1.5 * IQR
upper_bound <- Q3 + 1.5 * IQR

# Remove outliers
df$monthly_inhand_salary[df$monthly_inhand_salary < lower_bound | df$monthly_inhand_salary > upper_bound] <- NA

#replace the na value in annual income into the annual income in the same customer ID group
df <- df %>%
  group_by(customer_id) %>%
  fill(monthly_inhand_salary, .direction = 'downup')

#calculate mean and replace na with mean
mean_value <- mean(df$monthly_inhand_salary, na.rm = TRUE)
mean_value <- round(mean_value,2)
df$monthly_inhand_salary[is.na(df$monthly_inhand_salary)] <- mean_value
```

Figure 13: monthly in hand salary

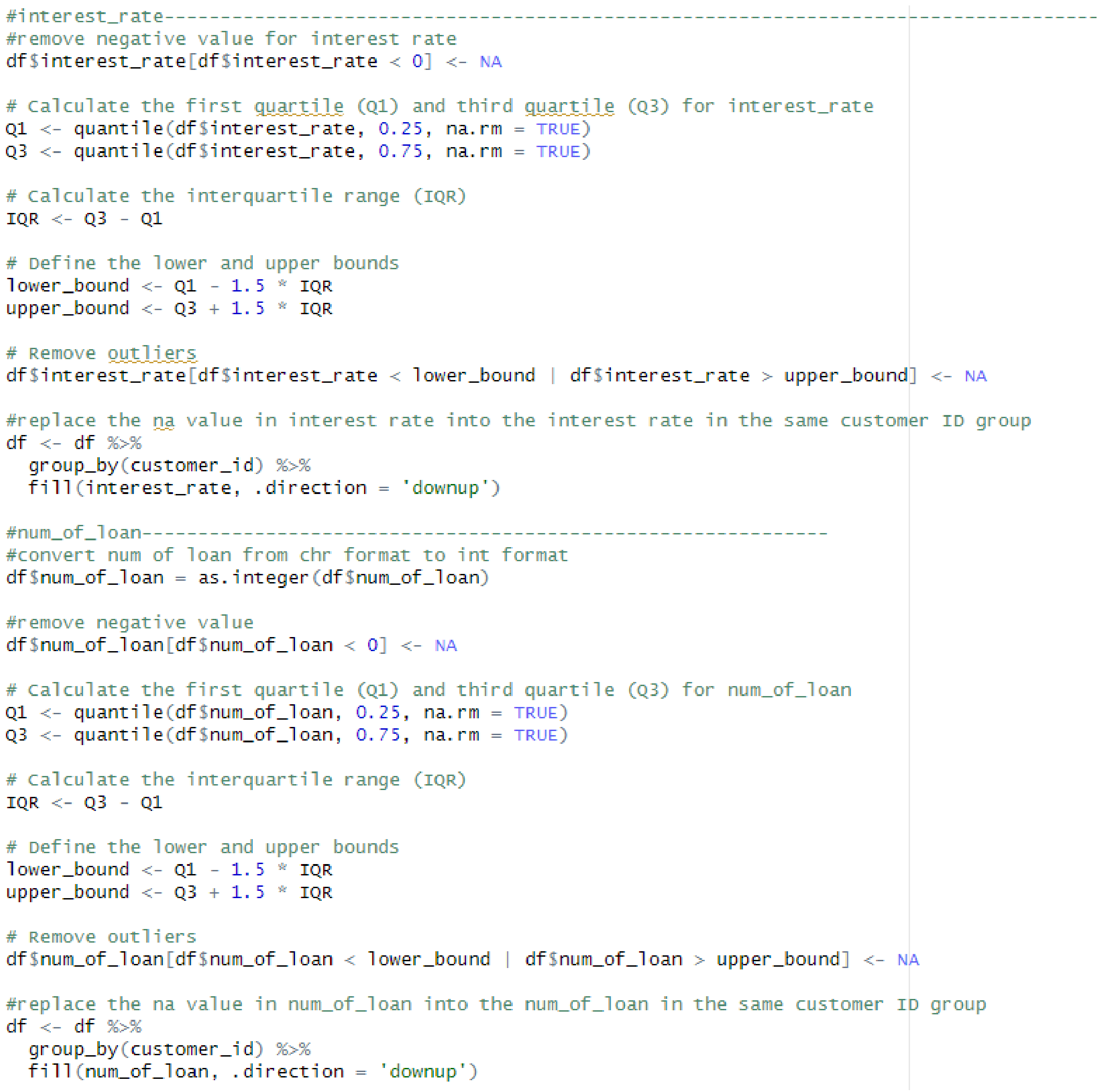
We use the quartile methods and convert the values into 2 decimal point to remove the outliers to later replace the NA values based on the same customer ID group.



```
#num_bank_accounts---------------------------------------------------------------------------
# Calculate the first quartile (Q1) and third quartile (Q3) for num_bank_accounts
Q1 <- quantile(df$num_bank_accounts, 0.25, na.rm = TRUE)
Q3 <- quantile(df$num_bank_accounts, 0.75, na.rm = TRUE)

# Calculate the interquartile range (IQR)
IQR <- Q3 - Q1

# Define the lower and upper bounds
lower_bound <- Q1 - 1.5 * IQR
upper_bound <- Q3 + 1.5 * IQR

# Remove outliers
df$num_bank_accounts[df$num_bank_accounts < lower_bound | df$num_bank_accounts > upper_bound] <- NA

#remove negative value
df$num_bank_accounts[df$num_bank_accounts < 0] <- NA

#replace the na value in num bank accounts into the num bank account in the same customer ID group
df <- df %>%
  group_by(customer_id) %>%
  fill(num_bank_accounts, .direction = 'downup')

#must has minimum 1 bank account
df$num_bank_accounts = replace(df$num_bank_accounts,df$num_bank_accounts == 0,1)

#num_credit_card------------------------------------------------------------------------------------
#remove negative value for num credit card
df$num_credit_card[df$num_credit_card < 0] <- NA

# Calculate the first quartile (Q1) and third quartile (Q3) for num_credit_card
Q1 <- quantile(df$num_credit_card, 0.25, na.rm = TRUE)
Q3 <- quantile(df$num_credit_card, 0.75, na.rm = TRUE)

# Calculate the interquartile range (IQR)
IQR <- Q3 - Q1

# Define the lower and upper bounds
lower_bound <- Q1 - 1.5 * IQR
upper_bound <- Q3 + 1.5 * IQR

# Remove outliers
df$num_credit_card[df$num_credit_card < lower_bound | df$num_credit_card > upper_bound] <- NA

#replace the na value in num credit card into the num credit in the same customer ID group
df <- df %>%
  group_by(customer_id) %>%
  fill(num_credit_card, .direction = 'downup')
```

Figure 14: number of bank accounts and credit card

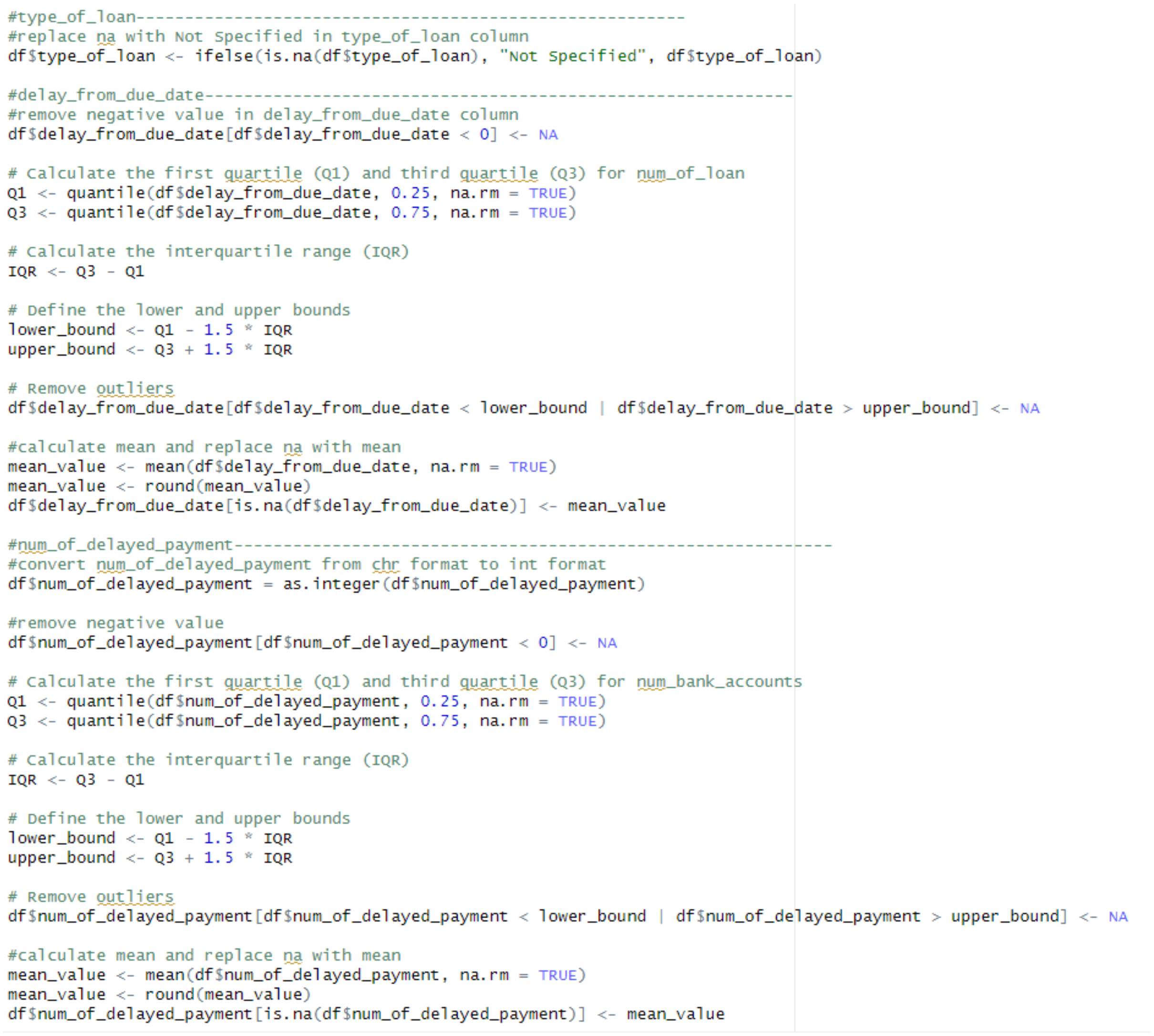
Using the same methods to remove outliers and some negative values. Replace the NA values with the same customer ID group at their columns. To make sure that they have the same value in the same customer ID group.



```r
#interest_rate-------------------------------------------------------------------------------
#remove negative value for interest rate
df$interest_rate[df$interest_rate < 0] <- NA

# Calculate the first quartile (Q1) and third quartile (Q3) for interest_rate
Q1 <- quantile(df$interest_rate, 0.25, na.rm = TRUE)
Q3 <- quantile(df$interest_rate, 0.75, na.rm = TRUE)

# Calculate the interquartile range (IQR)
IQR <- Q3 - Q1

# Define the lower and upper bounds
lower_bound <- Q1 - 1.5 * IQR
upper_bound <- Q3 + 1.5 * IQR

# Remove outliers
df$interest_rate[df$interest_rate < lower_bound | df$interest_rate > upper_bound] <- NA

#replace the na value in interest rate into the interest rate in the same customer ID group
df <- df %>%
  group_by(customer_id) %>%
  fill(interest_rate, .direction = 'downup')

#num_of_loan-----------------------------------------------------------
#convert num of loan from chr format to int format
df$num_of_loan = as.integer(df$num_of_loan)

#remove negative value
df$num_of_loan[df$num_of_loan < 0] <- NA

# Calculate the first quartile (Q1) and third quartile (Q3) for num_of_loan
Q1 <- quantile(df$num_of_loan, 0.25, na.rm = TRUE)
Q3 <- quantile(df$num_of_loan, 0.75, na.rm = TRUE)

# Calculate the interquartile range (IQR)
IQR <- Q3 - Q1

# Define the lower and upper bounds
lower_bound <- Q1 - 1.5 * IQR
upper_bound <- Q3 + 1.5 * IQR

# Remove outliers
df$num_of_loan[df$num_of_loan < lower_bound | df$num_of_loan > upper_bound] <- NA

#replace the na value in num_of_loan into the num_of_loan in the same customer ID group
df <- df %>%
  group_by(customer_id) %>%
  fill(num_of_loan, .direction = 'downup')
```

Figure 15: interest rate and number of loans

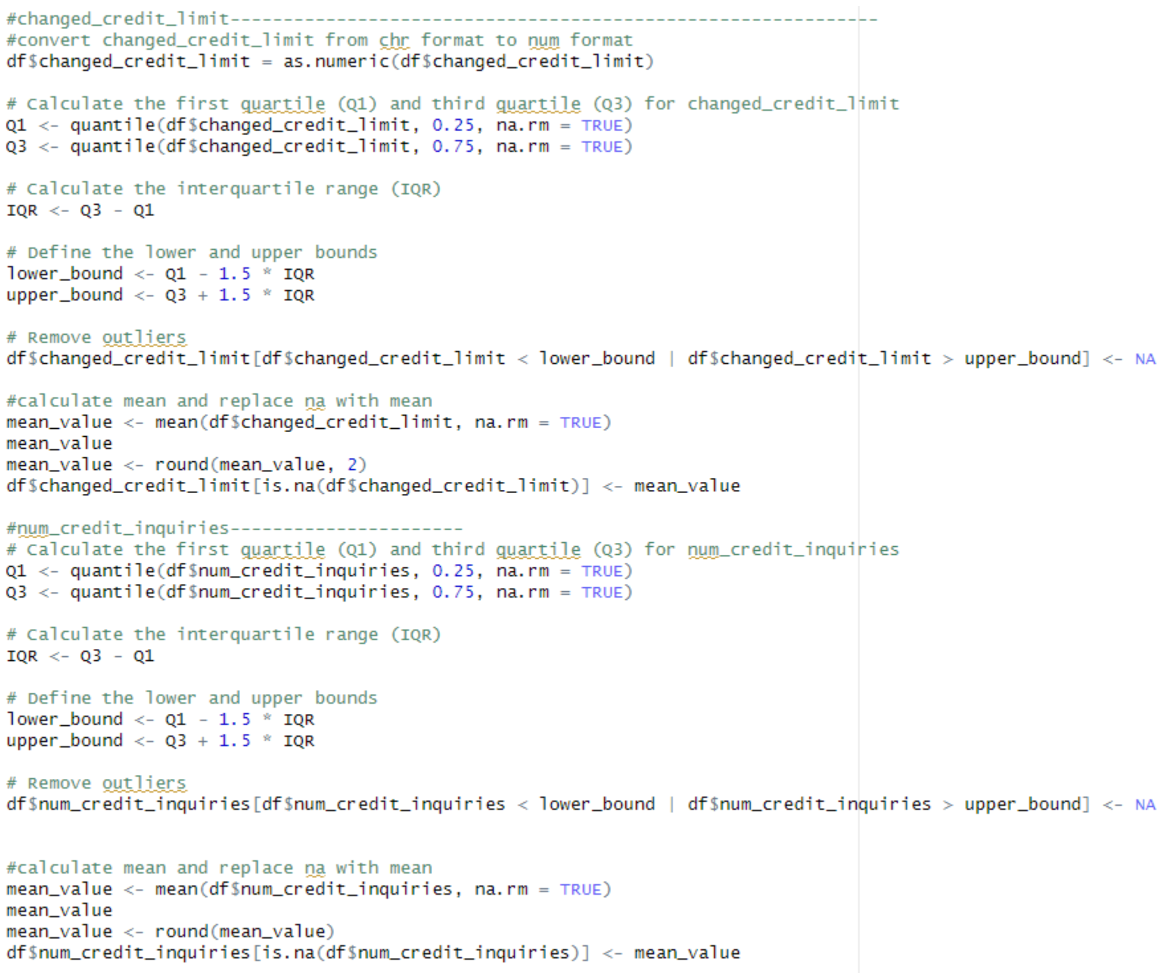
Convert the number of loans format from character to integer and use the quartile methods to remove the outliers, remove the negative values and replace it with the values group by the same customer ID group.



```
#type_of_loan---------------------------------------------------------
#replace na with Not Specified in type_of_loan column
df$type_of_loan <- ifelse(is.na(df$type_of_loan), "Not Specified", df$type_of_loan)

#delay_from_due_date--------------------------------------------------------------
#remove negative value in delay_from_due_date column
df$delay_from_due_date[df$delay_from_due_date < 0] <- NA

# Calculate the first quartile (Q1) and third quartile (Q3) for num_of_loan
Q1 <- quantile(df$delay_from_due_date, 0.25, na.rm = TRUE)
Q3 <- quantile(df$delay_from_due_date, 0.75, na.rm = TRUE)

# Calculate the interquartile range (IQR)
IQR <- Q3 - Q1

# Define the lower and upper bounds
lower_bound <- Q1 - 1.5 * IQR
upper_bound <- Q3 + 1.5 * IQR

# Remove outliers
df$delay_from_due_date[df$delay_from_due_date < lower_bound | df$delay_from_due_date > upper_bound] <- NA

#calculate mean and replace na with mean
mean_value <- mean(df$delay_from_due_date, na.rm = TRUE)
mean_value <- round(mean_value)
df$delay_from_due_date[is.na(df$delay_from_due_date)] <- mean_value

#num_of_delayed_payment--------------------------------------------------------------
#convert num_of_delayed_payment from chr format to int format
df$num_of_delayed_payment = as.integer(df$num_of_delayed_payment)

#remove negative value
df$num_of_delayed_payment[df$num_of_delayed_payment < 0] <- NA

# Calculate the first quartile (Q1) and third quartile (Q3) for num_bank_accounts
Q1 <- quantile(df$num_of_delayed_payment, 0.25, na.rm = TRUE)
Q3 <- quantile(df$num_of_delayed_payment, 0.75, na.rm = TRUE)

# Calculate the interquartile range (IQR)
IQR <- Q3 - Q1

# Define the lower and upper bounds
lower_bound <- Q1 - 1.5 * IQR
upper_bound <- Q3 + 1.5 * IQR

# Remove outliers
df$num_of_delayed_payment[df$num_of_delayed_payment < lower_bound | df$num_of_delayed_payment > upper_bound] <- NA

#calculate mean and replace na with mean
mean_value <- mean(df$num_of_delayed_payment, na.rm = TRUE)
mean_value <- round(mean_value)
df$num_of_delayed_payment[is.na(df$num_of_delayed_payment)] <- mean_value
```

Figure 16: type of loan, delay from due date and number of delayed payments

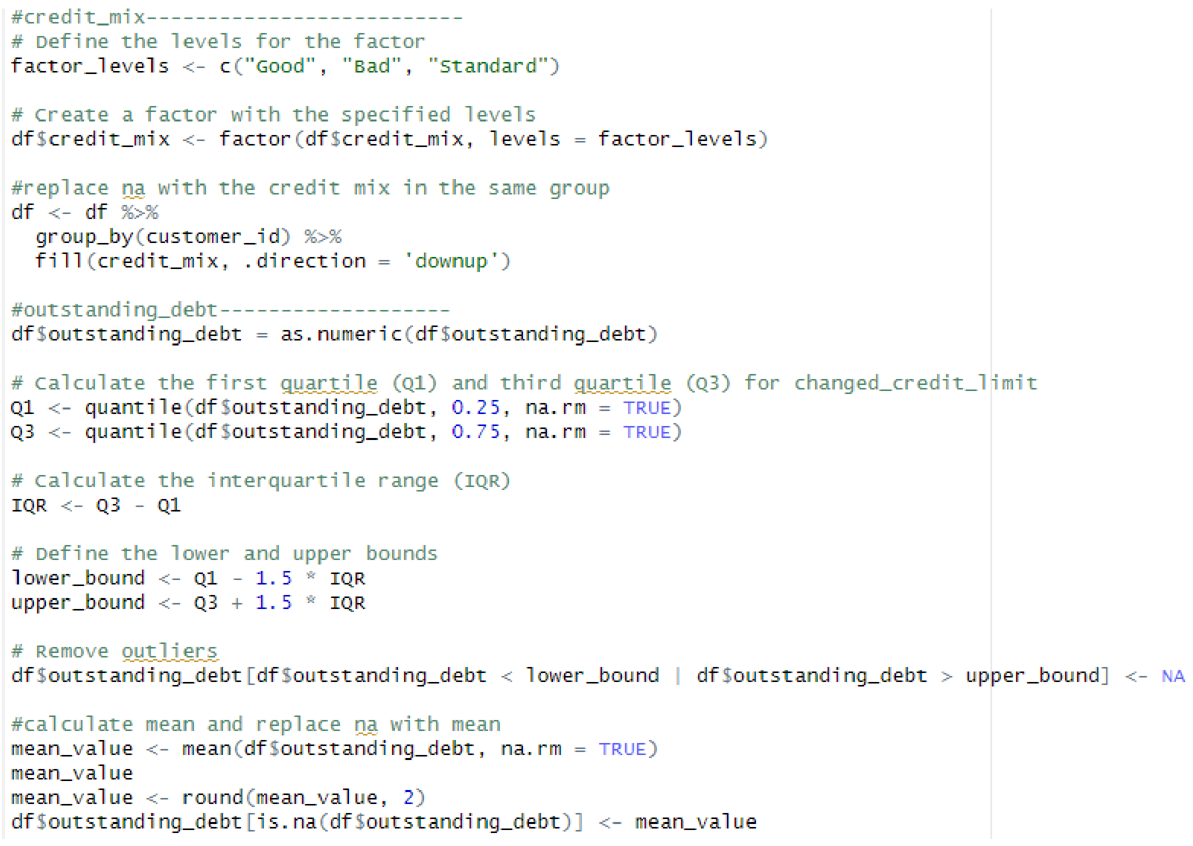
The above figure shows the columns type of loan, we replace the NA values with Not Specified word. Number of delayed payment class needs to replace to integer format and use quartile method to remove the outliers and replace the NA values with mean.



```
#changed_credit_limit--------------------------------------------------------------
#convert changed_credit_limit from chr format to num format
df$changed_credit_limit = as.numeric(df$changed_credit_limit)

# Calculate the first quartile (Q1) and third quartile (Q3) for changed_credit_limit
Q1 <- quantile(df$changed_credit_limit, 0.25, na.rm = TRUE)
Q3 <- quantile(df$changed_credit_limit, 0.75, na.rm = TRUE)

# Calculate the interquartile range (IQR)
IQR <- Q3 - Q1

# Define the lower and upper bounds
lower_bound <- Q1 - 1.5 * IQR
upper_bound <- Q3 + 1.5 * IQR

# Remove outliers
df$changed_credit_limit[df$changed_credit_limit < lower_bound | df$changed_credit_limit > upper_bound] <- NA

#calculate mean and replace na with mean
mean_value <- mean(df$changed_credit_limit, na.rm = TRUE)
mean_value
mean_value <- round(mean_value, 2)
df$changed_credit_limit[is.na(df$changed_credit_limit)] <- mean_value

#num_credit_inquiries----------------------
# Calculate the first quartile (Q1) and third quartile (Q3) for num_credit_inquiries
Q1 <- quantile(df$num_credit_inquiries, 0.25, na.rm = TRUE)
Q3 <- quantile(df$num_credit_inquiries, 0.75, na.rm = TRUE)

# Calculate the interquartile range (IQR)
IQR <- Q3 - Q1

# Define the lower and upper bounds
lower_bound <- Q1 - 1.5 * IQR
upper_bound <- Q3 + 1.5 * IQR

# Remove outliers
df$num_credit_inquiries[df$num_credit_inquiries < lower_bound | df$num_credit_inquiries > upper_bound] <- NA


#calculate mean and replace na with mean
mean_value <- mean(df$num_credit_inquiries, na.rm = TRUE)
mean_value
mean_value <- round(mean_value)
df$num_credit_inquiries[is.na(df$num_credit_inquiries)] <- mean_value
```

Figure 17: changed credit limit and number of credit inquiries

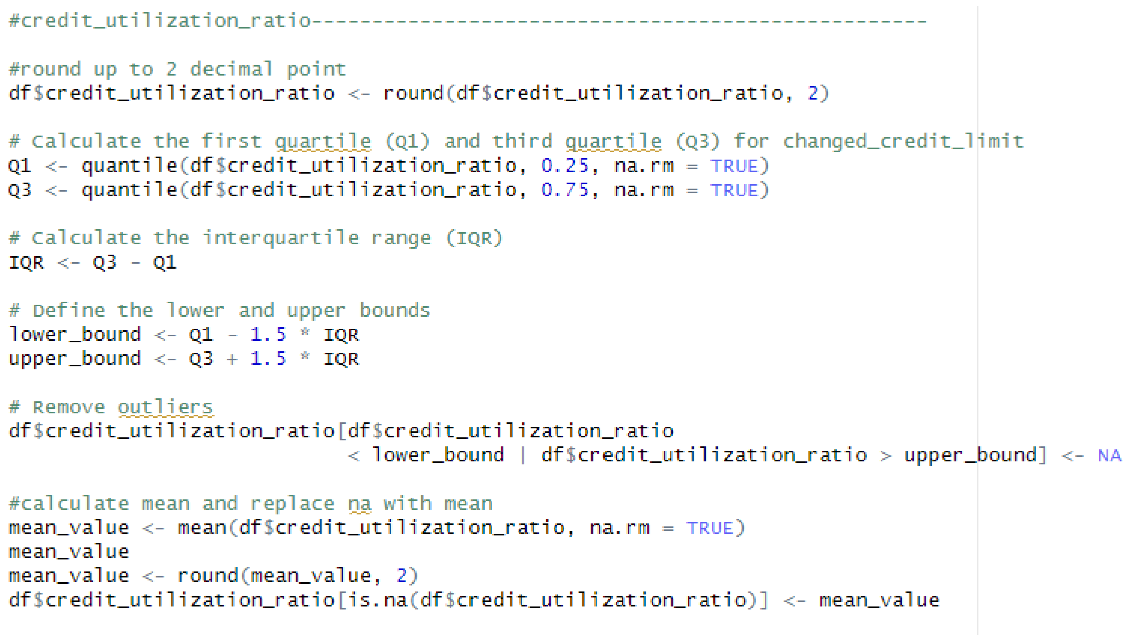
Quartile method is using to remove outliers for both columns and replace NA values with mean.



```
#credit_mix--------------------------
# Define the levels for the factor
factor_levels <- c("Good", "Bad", "Standard")

# Create a factor with the specified levels
df$credit_mix <- factor(df$credit_mix, levels = factor_levels)

#replace na with the credit mix in the same group
df <- df %>%
  group_by(customer_id) %>%
  fill(credit_mix, .direction = 'downup')

#outstanding_debt-------------------
df$outstanding_debt = as.numeric(df$outstanding_debt)

# Calculate the first quartile (Q1) and third quartile (Q3) for changed_credit_limit
Q1 <- quantile(df$outstanding_debt, 0.25, na.rm = TRUE)
Q3 <- quantile(df$outstanding_debt, 0.75, na.rm = TRUE)

# Calculate the interquartile range (IQR)
IQR <- Q3 - Q1

# Define the lower and upper bounds
lower_bound <- Q1 - 1.5 * IQR
upper_bound <- Q3 + 1.5 * IQR

# Remove outliers
df$outstanding_debt[df$outstanding_debt < lower_bound | df$outstanding_debt > upper_bound] <- NA

#calculate mean and replace na with mean
mean_value <- mean(df$outstanding_debt, na.rm = TRUE)
mean_value
mean_value <- round(mean_value, 2)
df$outstanding_debt[is.na(df$outstanding_debt)] <- mean_value
```

Figure 18: credit mix and outstanding debt

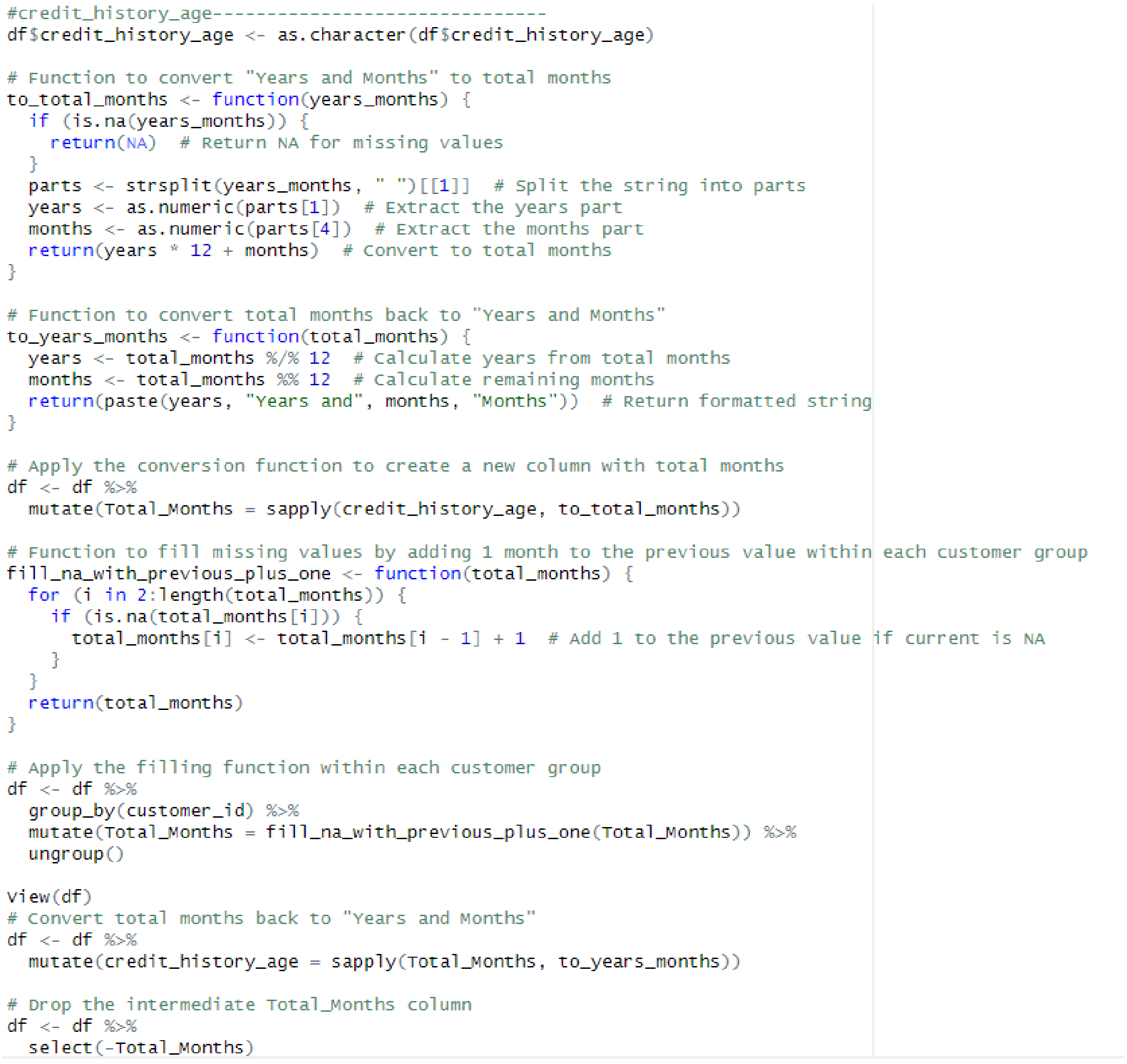
We use factor_levels to category the credit mix with 3 levels. Convert the class of outstanding debt to numeric format and use quartile method to remove the outliers. Replace the NA values with mean.



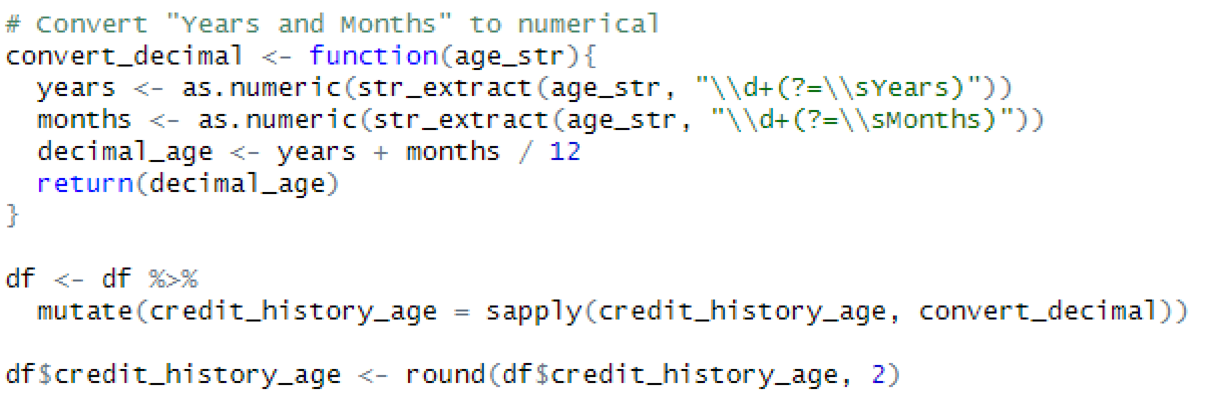
```
#credit_utilization_ratio--------------------------------------------------

#round up to 2 decimal point
df$credit_utilization_ratio <- round(df$credit_utilization_ratio, 2)

# Calculate the first quartile (Q1) and third quartile (Q3) for changed_credit_limit
Q1 <- quantile(df$credit_utilization_ratio, 0.25, na.rm = TRUE)
Q3 <- quantile(df$credit_utilization_ratio, 0.75, na.rm = TRUE)

# Calculate the interquartile range (IQR)
IQR <- Q3 - Q1

# Define the lower and upper bounds
lower_bound <- Q1 - 1.5 * IQR
upper_bound <- Q3 + 1.5 * IQR

# Remove outliers
df$credit_utilization_ratio[df$credit_utilization_ratio
                            < lower_bound | df$credit_utilization_ratio > upper_bound] <- NA

#calculate mean and replace na with mean
mean_value <- mean(df$credit_utilization_ratio, na.rm = TRUE)
mean_value
mean_value <- round(mean_value, 2)
df$credit_utilization_ratio[is.na(df$credit_utilization_ratio)] <- mean_value
```

Figure 19: credit utilization ratio

The values will be converted to 2 decimal point and remover the outliers with quartile method. Replace the NA value with mean.



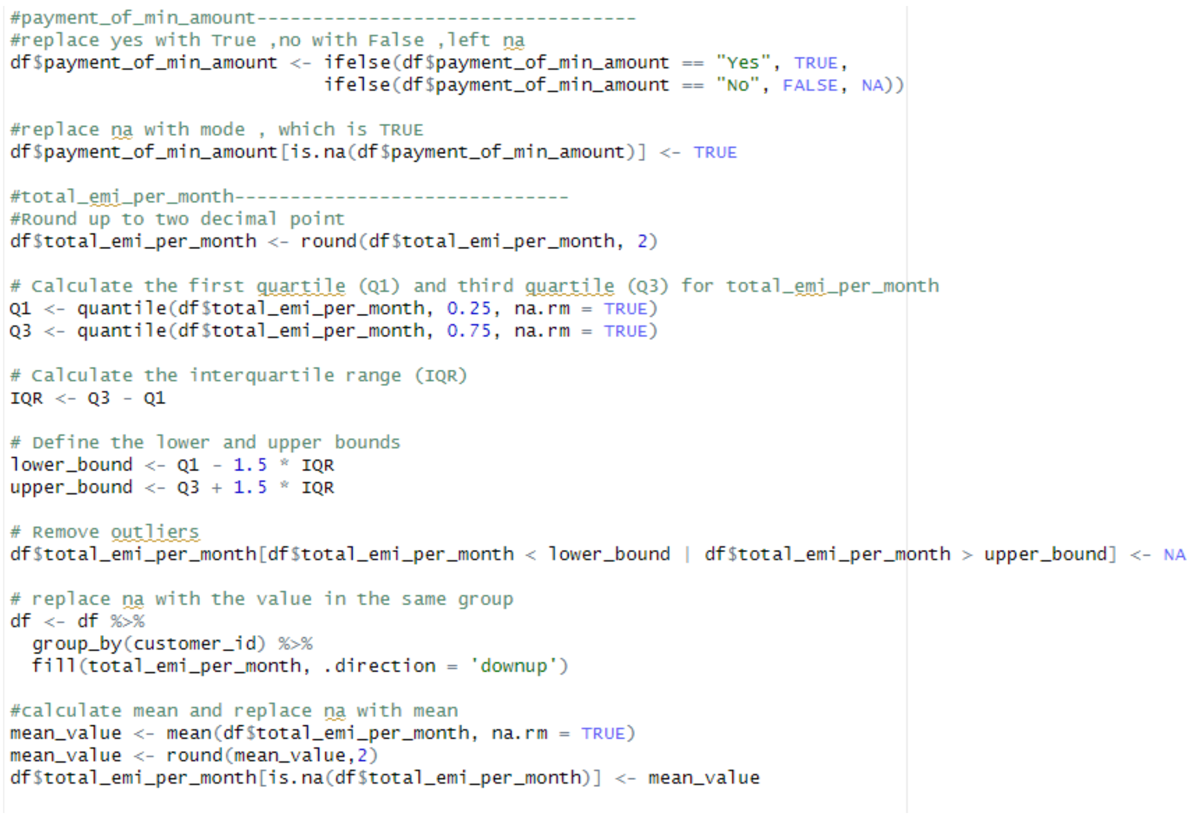
```r
#credit_history_age-------------------------------
df$credit_history_age <- as.character(df$credit_history_age)

# Function to convert "Years and Months" to total months
to_total_months <- function(years_months) {
  if (is.na(years_months)) {
    return(NA)  # Return NA for missing values
  }
  parts <- strsplit(years_months, " ")[[1]]  # Split the string into parts
  years <- as.numeric(parts[1])  # Extract the years part
  months <- as.numeric(parts[4])  # Extract the months part
  return(years * 12 + months)  # Convert to total months
}

# Function to convert total months back to "Years and Months"
to_years_months <- function(total_months) {
  years <- total_months %/% 12  # Calculate years from total months
  months <- total_months %% 12  # Calculate remaining months
  return(paste(years, "Years and", months, "Months"))  # Return formatted string
}

# Apply the conversion function to create a new column with total months
df <- df %>%
  mutate(Total_Months = sapply(credit_history_age, to_total_months))

# Function to fill missing values by adding 1 month to the previous value within each customer group
fill_na_with_previous_plus_one <- function(total_months) {
  for (i in 2:length(total_months)) {
    if (is.na(total_months[i])) {
      total_months[i] <- total_months[i - 1] + 1  # Add 1 to the previous value if current is NA
    }
  }
  return(total_months)
}

# Apply the filling function within each customer group
df <- df %>%
  group_by(customer_id) %>%
  mutate(Total_Months = fill_na_with_previous_plus_one(Total_Months)) %>%
  ungroup()

View(df)
# Convert total months back to "Years and Months"
df <- df %>%
  mutate(credit_history_age = sapply(Total_Months, to_years_months))

# Drop the intermediate Total_Months column
df <- df %>%
  select(-Total_Months)
```



```r
# Convert "Years and Months" to numerical
convert_decimal <- function(age_str){
  years <- as.numeric(str_extract(age_str, "\\d+(?=\\sYears)"))
  months <- as.numeric(str_extract(age_str, "\\d+(?=\\sMonths)"))
  decimal_age <- years + months / 12
  return(decimal_age)
}

df <- df %>%
  mutate(credit_history_age = sapply(credit_history_age, convert_decimal))

df$credit_history_age <- round(df$credit_history_age, 2)
```

Figure 20: credit history age

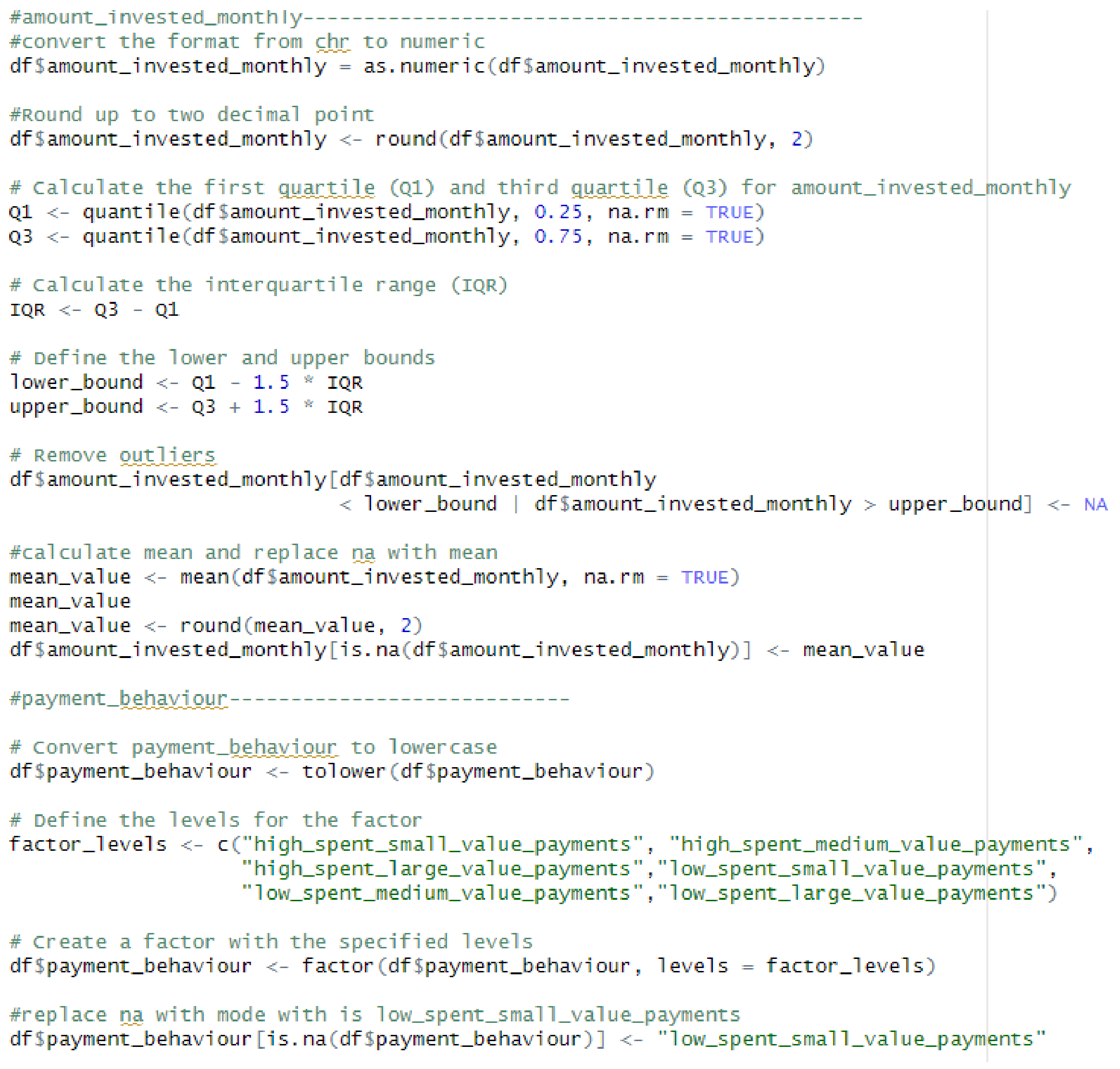
The credit history age is converted from character into decimal values and the values need to divide by the months with 12. We assume that the credit history age doesn't have data redundancy and shown in Years format for a clearer data represent.



```
#payment_of_min_amount----------------------------------
#replace yes with True ,no with False ,left na
df$payment_of_min_amount <- ifelse(df$payment_of_min_amount == "Yes", TRUE,
                            ifelse(df$payment_of_min_amount == "No", FALSE, NA))

#replace na with mode , which is TRUE
df$payment_of_min_amount[is.na(df$payment_of_min_amount)] <- TRUE

#total_emi_per_month-----------------------------
#Round up to two decimal point
df$total_emi_per_month <- round(df$total_emi_per_month, 2)

# Calculate the first quartile (Q1) and third quartile (Q3) for total_emi_per_month
Q1 <- quantile(df$total_emi_per_month, 0.25, na.rm = TRUE)
Q3 <- quantile(df$total_emi_per_month, 0.75, na.rm = TRUE)

# Calculate the interquartile range (IQR)
IQR <- Q3 - Q1

# Define the lower and upper bounds
lower_bound <- Q1 - 1.5 * IQR
upper_bound <- Q3 + 1.5 * IQR

# Remove outliers
df$total_emi_per_month[df$total_emi_per_month < lower_bound | df$total_emi_per_month > upper_bound] <- NA

# replace na with the value in the same group
df <- df %>%
  group_by(customer_id) %>%
  fill(total_emi_per_month, .direction = 'downup')

#calculate mean and replace na with mean
mean_value <- mean(df$total_emi_per_month, na.rm = TRUE)
mean_value <- round(mean_value,2)
df$total_emi_per_month[is.na(df$total_emi_per_month)] <- mean_value
```

Figure 21: payment of min amount and total emi per month

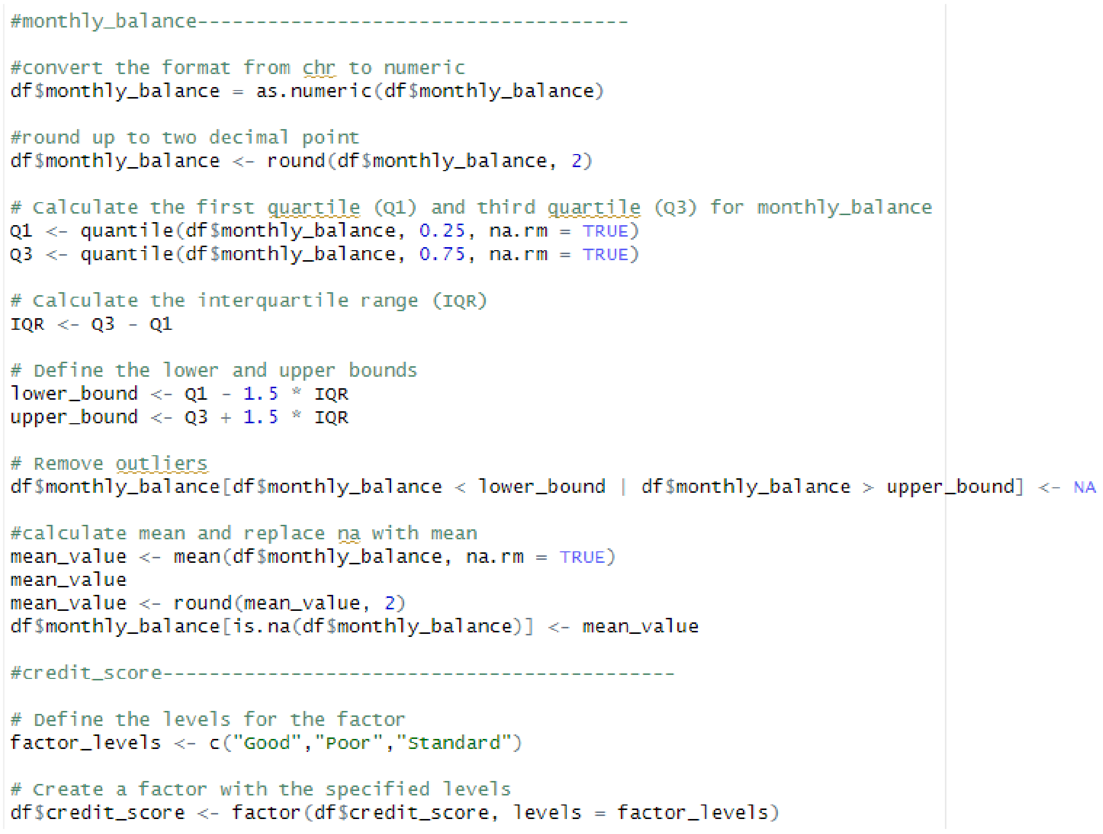
The values of payment of min amount will been replace yes with true and no with false. Mode is used for replacing the NA values which are true. Using quartile methods to remove the outliers and replace the NA values with mean in the total emi per month.



```
#amount_invested_monthly---------------------------------------------
#convert the format from chr to numeric
df$amount_invested_monthly = as.numeric(df$amount_invested_monthly)

#Round up to two decimal point
df$amount_invested_monthly <- round(df$amount_invested_monthly, 2)

# Calculate the first quartile (Q1) and third quartile (Q3) for amount_invested_monthly
Q1 <- quantile(df$amount_invested_monthly, 0.25, na.rm = TRUE)
Q3 <- quantile(df$amount_invested_monthly, 0.75, na.rm = TRUE)

# Calculate the interquartile range (IQR)
IQR <- Q3 - Q1

# Define the lower and upper bounds
lower_bound <- Q1 - 1.5 * IQR
upper_bound <- Q3 + 1.5 * IQR

# Remove outliers
df$amount_invested_monthly[df$amount_invested_monthly
                           < lower_bound | df$amount_invested_monthly > upper_bound] <- NA

#calculate mean and replace na with mean
mean_value <- mean(df$amount_invested_monthly, na.rm = TRUE)
mean_value
mean_value <- round(mean_value, 2)
df$amount_invested_monthly[is.na(df$amount_invested_monthly)] <- mean_value

#payment_behaviour---------------------------

# Convert payment_behaviour to lowercase
df$payment_behaviour <- tolower(df$payment_behaviour)

# Define the levels for the factor
factor_levels <- c("high_spent_small_value_payments", "high_spent_medium_value_payments",
                   "high_spent_large_value_payments","low_spent_small_value_payments",
                   "low_spent_medium_value_payments","low_spent_large_value_payments")

# Create a factor with the specified levels
df$payment_behaviour <- factor(df$payment_behaviour, levels = factor_levels)

#replace na with mode with is low_spent_small_value_payments
df$payment_behaviour[is.na(df$payment_behaviour)] <- "low_spent_small_value_payments"
```

Figure 22: amount invested monthly and payment behaviour

For the amount invested monthly, we converted its class into numeric format and round up to 2 decimal point. Quartile methods is used to remove outliers and replace the NA values with mean. Payment behaviour characters are converted into lowercase and define the levels with 6 different levels. We also create a factor with the specified levels and replace the NA values with mode with low_spent_small_value_payments.



```
#monthly_balance------------------------------------

#convert the format from chr to numeric
df$monthly_balance = as.numeric(df$monthly_balance)

#round up to two decimal point
df$monthly_balance <- round(df$monthly_balance, 2)

# Calculate the first quartile (Q1) and third quartile (Q3) for monthly_balance
Q1 <- quantile(df$monthly_balance, 0.25, na.rm = TRUE)
Q3 <- quantile(df$monthly_balance, 0.75, na.rm = TRUE)

# Calculate the interquartile range (IQR)
IQR <- Q3 - Q1

# Define the lower and upper bounds
lower_bound <- Q1 - 1.5 * IQR
upper_bound <- Q3 + 1.5 * IQR

# Remove outliers
df$monthly_balance[df$monthly_balance < lower_bound | df$monthly_balance > upper_bound] <- NA

#calculate mean and replace na with mean
mean_value <- mean(df$monthly_balance, na.rm = TRUE)
mean_value
mean_value <- round(mean_value, 2)
df$monthly_balance[is.na(df$monthly_balance)] <- mean_value

#credit_score------------------------------------------

# Define the levels for the factor
factor_levels <- c("Good","Poor","Standard")

# Create a factor with the specified levels
df$credit_score <- factor(df$credit_score, levels = factor_levels)
```

Figure 23: monthly balance and credit score

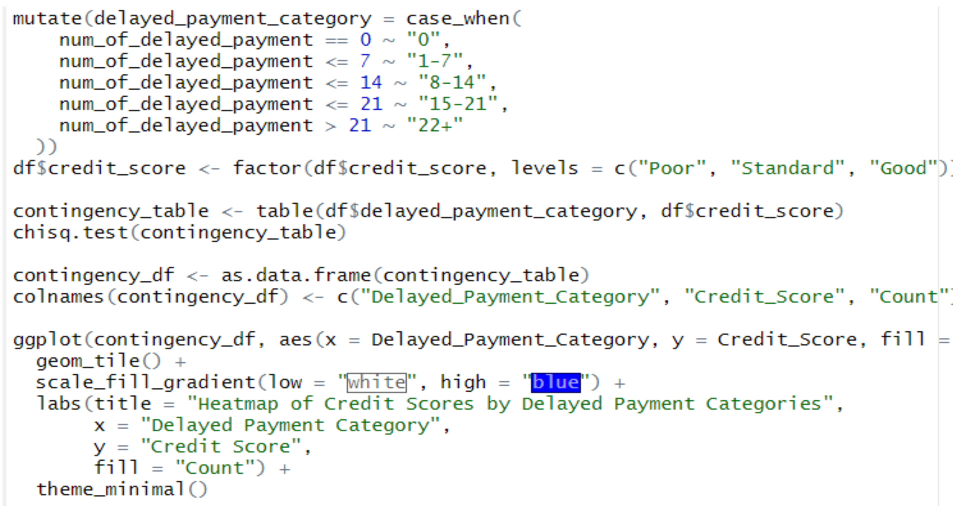
The monthly balance format has been converted into numeric format and make it to have 2 decimal points. Using quartile methods to remove outliers and replace the NA values with mean. For the credit score, we define the factor of credit score as 3 levels. After cleaning the credit score, the data-cleaning process is finished.

4.0 Data Analysis

4.1 To investigate the impact of delayed payment to credit score (Angelina Leanore TP072929)

The three analysis employ a variety of analysis techniques, including the Chi-square test of independence, group by, Visualization, and Descriptive analysis. There are three sorts of visualization used: heatmaps, line plots with points, and violin plots. Descriptive analysis compares the distribution of a variable across several categories.

4.1.1 Does the frequency of delayed payment significantly affect the credit score of individuals?



```
mutate(delayed_payment_category = case_when(
    num_of_delayed_payment == 0 ~ "0",
    num_of_delayed_payment <= 7 ~ "1-7",
    num_of_delayed_payment <= 14 ~ "8-14",
    num_of_delayed_payment <= 21 ~ "15-21",
    num_of_delayed_payment > 21 ~ "22+"
  ))
df$credit_score <- factor(df$credit_score, levels = c("Poor", "Standard", "Good"))

contingency_table <- table(df$delayed_payment_category, df$credit_score)
chisq.test(contingency_table)

contingency_df <- as.data.frame(contingency_table)
colnames(contingency_df) <- c("Delayed_Payment_Category", "Credit_Score", "Count")

ggplot(contingency_df, aes(x = Delayed_Payment_Category, y = Credit_Score, fill = Count)) +
  geom_tile() +
  scale_fill_gradient(low = "white", high = "blue") +
  labs(title = "Heatmap of Credit Scores by Delayed Payment Categories",
       x = "Delayed Payment Category",
       y = "Credit Score",
       fill = "Count") +
  theme_minimal()
```

Figure 24: Code for Implementing Heatmap

Extra Feature: Heatmap Visualization

Data Analysis Method: Chi-Square test of independence

According to this code, the number of categories with payment delays is used to determine credit score by determining a person’s credit score based on how many late payments they have made. Additionally, a contingency table is developed in order to examine the link between he types of delayed payments and credit scores. Lastly, we can view the data’s heatmap using geom_tile.

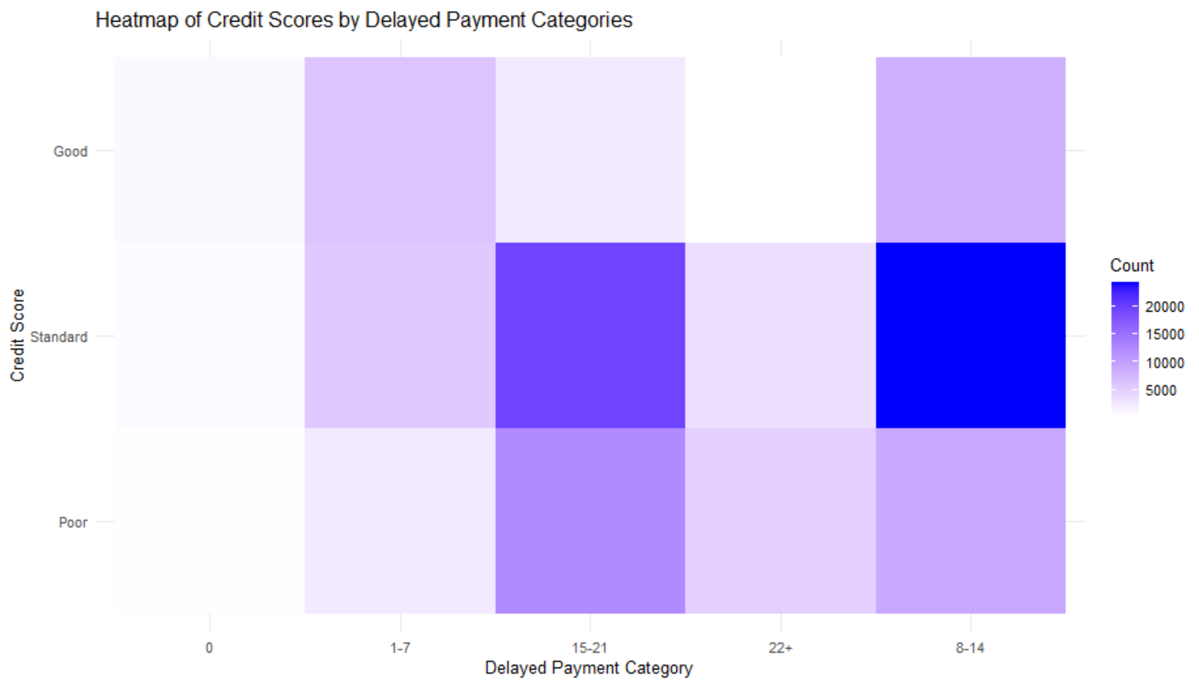
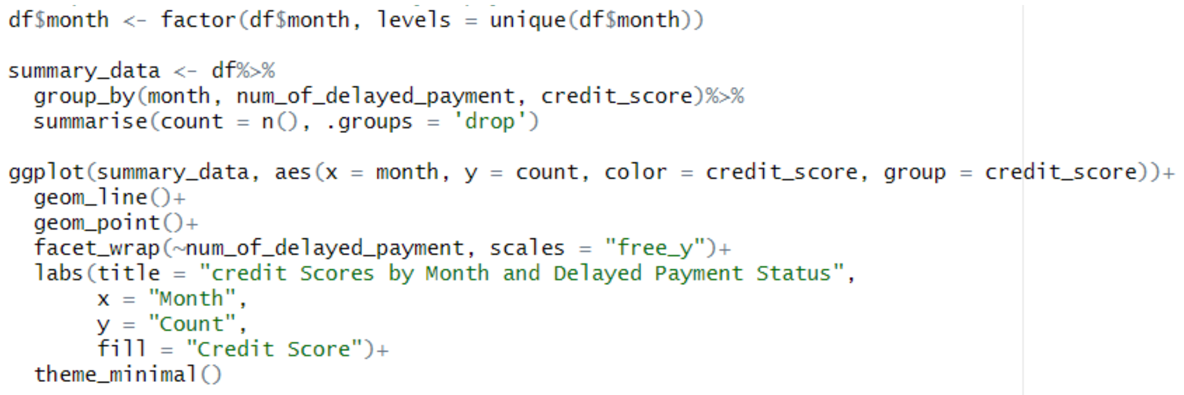


Figure 25: Heatmap

The heatmap demonstrates the relationship between credit scores and the frequency of late payments. There are most members in the “Standard” credit score group, especially in the 8-14 delayed payment range area. Comparatively, the credit score is largely affected by the 8-14 and 22+ delayed payment categories, where a large proportion of people fall into the “Standard” and “Poor” credit categories. Fewer late payments are linked to higher credit ratings, such as “Good” scores the majority of people in the “Poor” credit category with more than 22 late payments are most affected. This heatmap shows that a pattern of often missing payments is associated with a lower credit score.

4.1.2 Do individuals with delayed payments in specific months have significantly different credit scores compared to those without delayed payments in those months?



```
df$month <- factor(df$month, levels = unique(df$month))

summary_data <- df%>%
  group_by(month, num_of_delayed_payment, credit_score)%>%
  summarise(count = n(), .groups = 'drop')

ggplot(summary_data, aes(x = month, y = count, color = credit_score, group = credit_score))+
  geom_line()+
  geom_point()+
  facet_wrap(~num_of_delayed_payment, scales = "free_y")+
  labs(title = "credit Scores by Month and Delayed Payment Status",
       x = "Month",
       y = "Count",
       fill = "Credit Score")+
  theme_minimal()
```

Figure 26: Code for Lineplot with Facets

Data Analysis Method: Group by method

This code creates a faceted line plot using ggplot2 to display the credit score count by month and the status of delayed payments. Organizing the data according to month, status of overdue payments, and credit score results in a visually appealing plot. Furthermore, by creating distinct subplots according to each delayed payment status using faceting, y scales can vary individually.

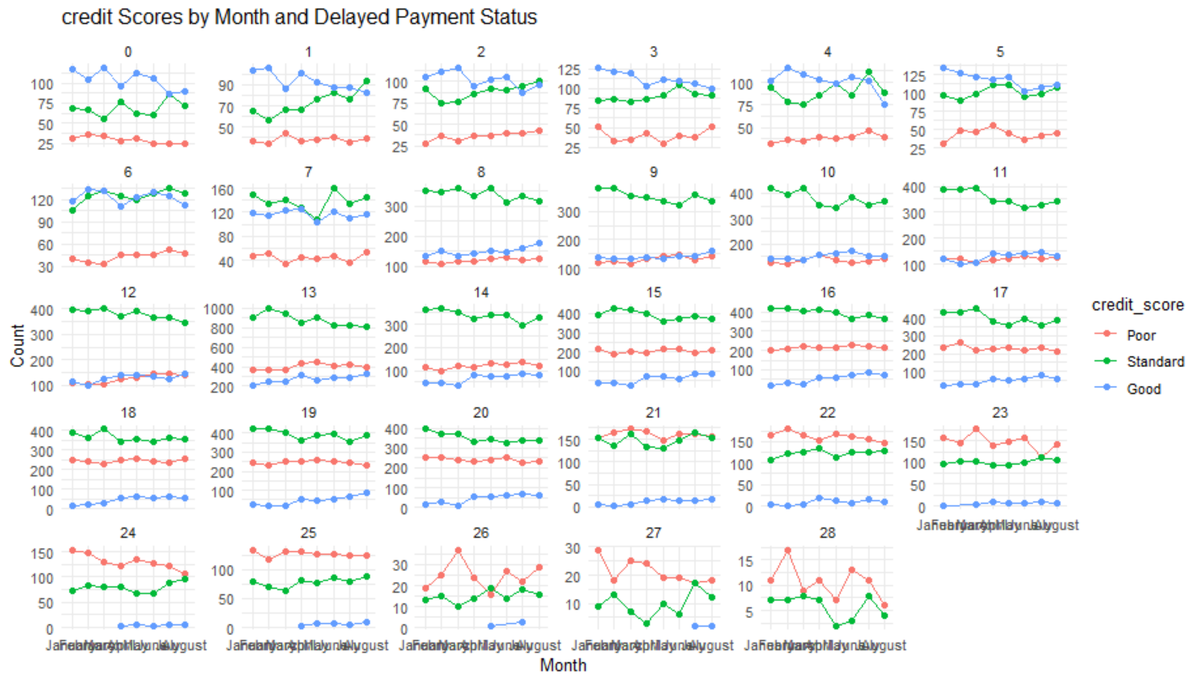
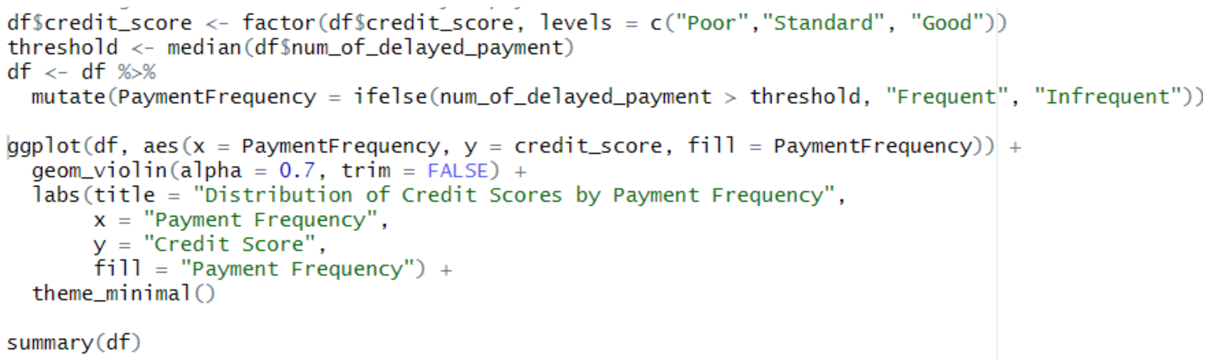


Figure 27: Lineplot with Facets

Based on the Graph above, there are many subplots, each with a numbers 0 to 28 signifying various amounts of overdue payments. Monthly distribution of credit scores is displayed in each subplot. In response, those is display in each subplot. In response, those who have fallen behind on their payments do, on average, have credit score distributions that differ from those of people who have not. “Standard” and “Poor” typically have greater counts than “Good” for the majority of delayed payment counts (0 to 8). A few subplots (26-28) suggest that some kinds of late payments have a more pronounced effect on credit score than others. Furthermore, there appear to be monthly variations in the counts for “Poor” credit score, indicating the potential influence of late payments.

4.1.3 Is there a significant difference in the average credit score between periods with frequent and infrequent numbers of delayed payments over time?



```
df$credit_score <- factor(df$credit_score, levels = c("Poor","Standard", "Good"))
threshold <- median(df$num_of_delayed_payment)
df <- df %>%
  mutate(PaymentFrequency = ifelse(num_of_delayed_payment > threshold, "Frequent", "Infrequent"))

ggplot(df, aes(x = PaymentFrequency, y = credit_score, fill = PaymentFrequency)) +
  geom_violin(alpha = 0.7, trim = FALSE) +
  labs(title = "Distribution of Credit Scores by Payment Frequency",
       x = "Payment Frequency",
       y = "Credit Score",
       fill = "Payment Frequency") +
  theme_minimal()

summary(df)
```

Figure 28: Code for Violin Plot

Extra Feature: Violin Plot Visualization

Data Analysis Method: Descriptive Analysis

It is shown in the code that, in order to differentiate between Frequent and Infrequent payments. The entry is labelled as “Frequent” if there have been more delayed payments than the threshold, and “Infrequent” if not. The distribution of credit scores by payment frequency can be seen by using the ggplot tool to construct a violin plot.

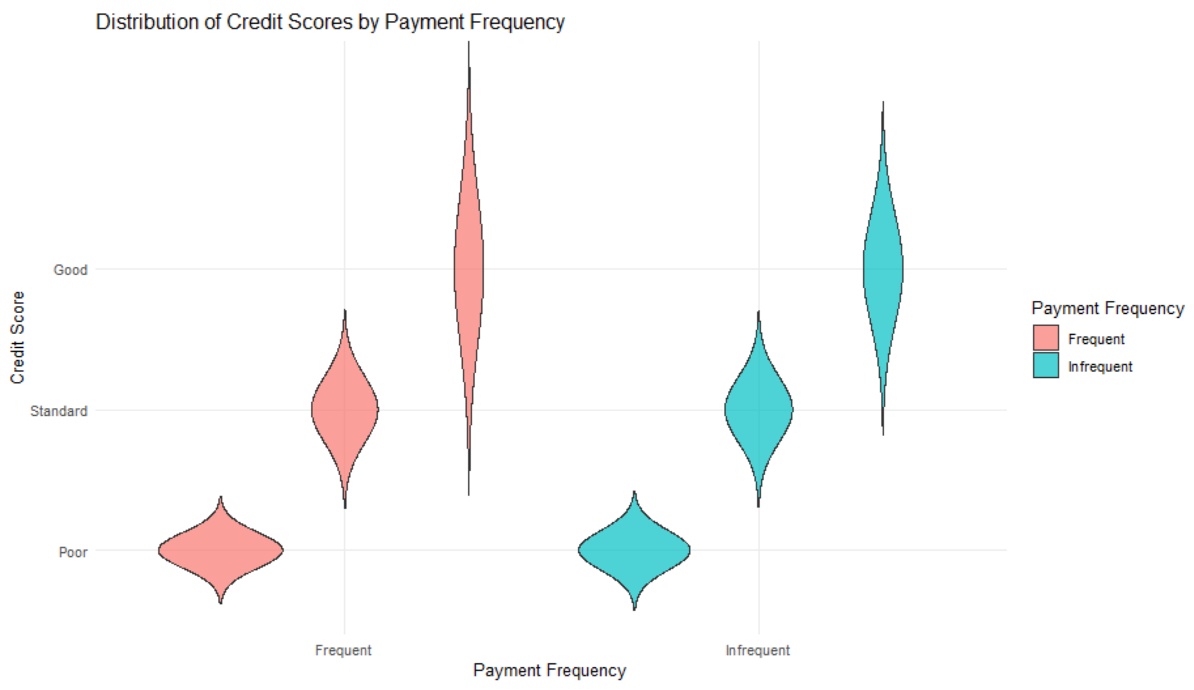


Figure 29: Violin Plot

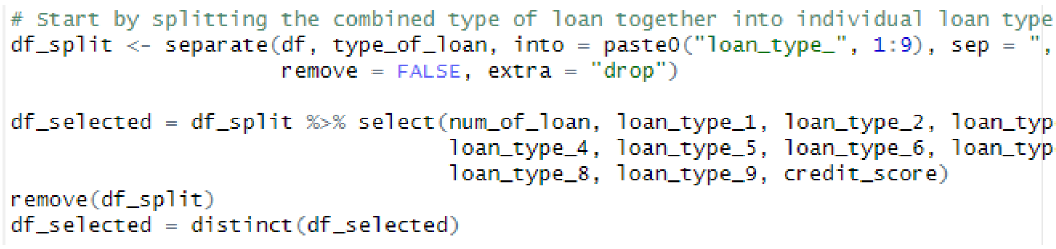
The distribution of credit scores clearly differs between periods with frequent and infrequently delayed payments, as the violin plot illustrates. Fewer people have excellent credit scores than poor or standard credit scores, which is based on the frequency of late payments. In contrast to the frequent category, the distribution of people with good credit scores is more balance among those who infrequently make late payments. Based on the frequency of late payments, this visual analysis indicates a considerable variation in average credit scores.

Conclusion

In conclusion, based on the findings of each investigation, we can conclude that delayed payments have a considerable impact on credit scores. Frequent late payments are related with a lower credit score, but fewer late payments are associated with higher credit score. Finally, it is show that controlling payment schedules and avoiding frequent delays can improve one's credit score.

4.2 To investigate impact of number of loans to a person's credit score (Lee De Xian TP068265)

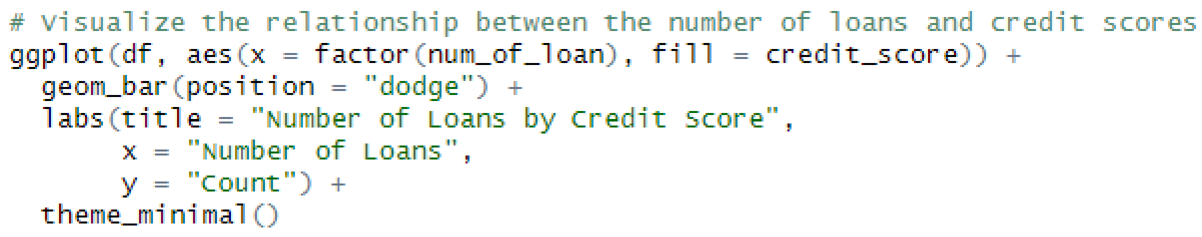
4.2.1 Does the number of loans affect the credit score of a person's credit score



```r
# Start by splitting the combined type of loan together into individual loan type
df_split <- separate(df, type_of_loan, into = paste0("loan_type_", 1:9), sep = ", and |, ",
                     remove = FALSE, extra = "drop")

df_selected = df_split %>% select(num_of_loan, loan_type_1, loan_type_2, loan_type_3,
                                  loan_type_4, loan_type_5, loan_type_6, loan_type_7,
                                  loan_type_8, loan_type_9, credit_score)
remove(df_split)
df_selected = distinct(df_selected)
```

Figure 30:

We start by breaking down the type of loan in the main df into a separate df by identifying the separator then putting it into their respective loan type arrangement which is maximum of 9 as there are only a maximum of 9 loans per person. After that we pick out the columns that are useful to us to reduce bloat in generating plots. To reduce repetition even further, we remove identical data.



```r
# Visualize the relationship between the number of loans and credit scores
ggplot(df, aes(x = factor(num_of_loan), fill = credit_score)) +
  geom_bar(position = "dodge") +
  labs(title = "Number of Loans by Credit Score",
       x = "Number of Loans",
       y = "Count") +
  theme_minimal()
```

Figure 31: Code for Bar Plot

The R script shown above is a basic generation of a grouped bar plot. The purpose of this plot is to show the correlation between the number of loans and the amount of different credit score.

I chose to use “dodge” for the purpose of avoiding overlapping bars of different categories into together.

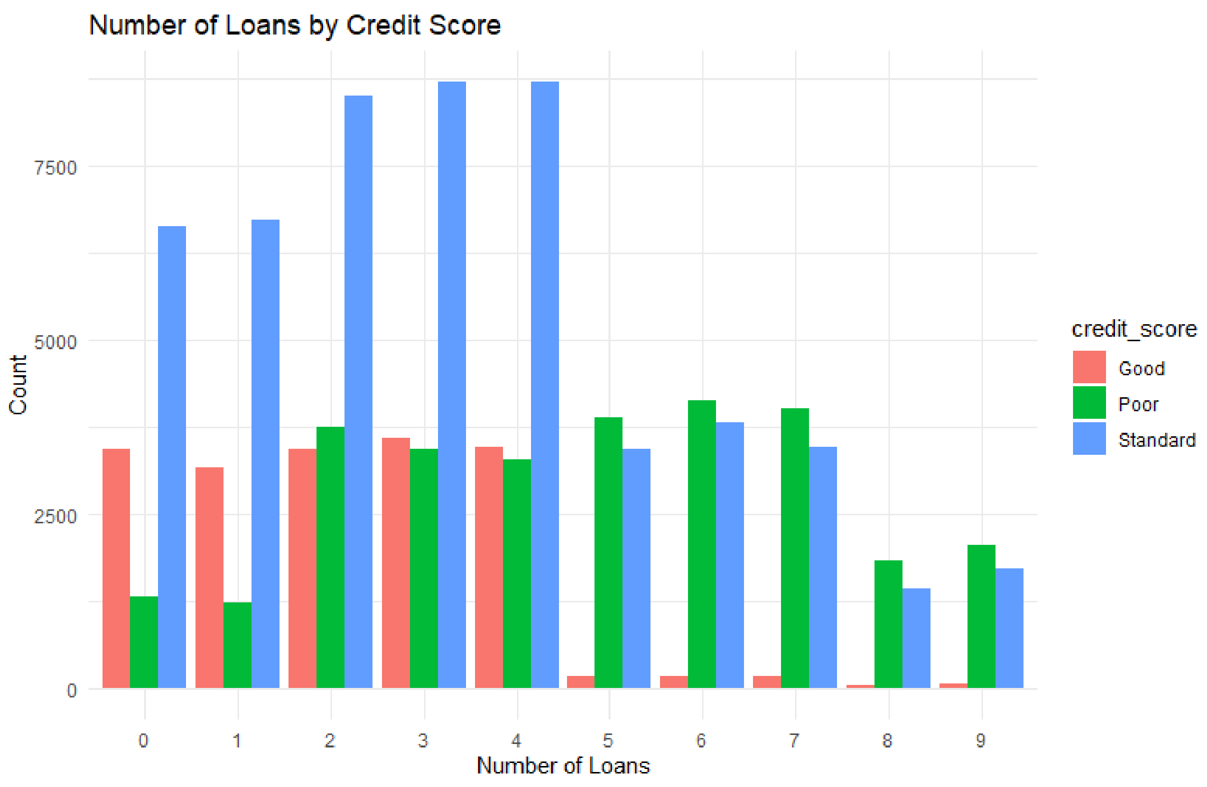


Figure 32: Bar plot

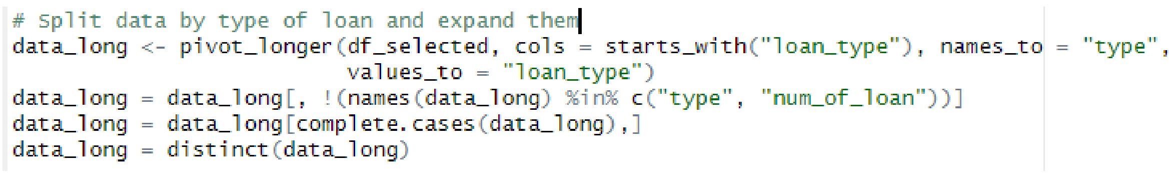
Extra Features: Expanding merged loan type into multiple loan type

Data Analysis Method: Data Visualization

This is what the bar plot looks like generated. At first glance the most obvious difference is the drastic change of counts from 4 loans to 5 loans. The other observation is that before 5 loans, the majority of people’s credit score are of Standard rating, but beyond that Poor rating becomes the majority. Good credit score also drops to almost non-existent beyond 4 while good score being more frequent from 4 and below majority of the time.

4.2.2 Does a particular type of loan affect the credit score?

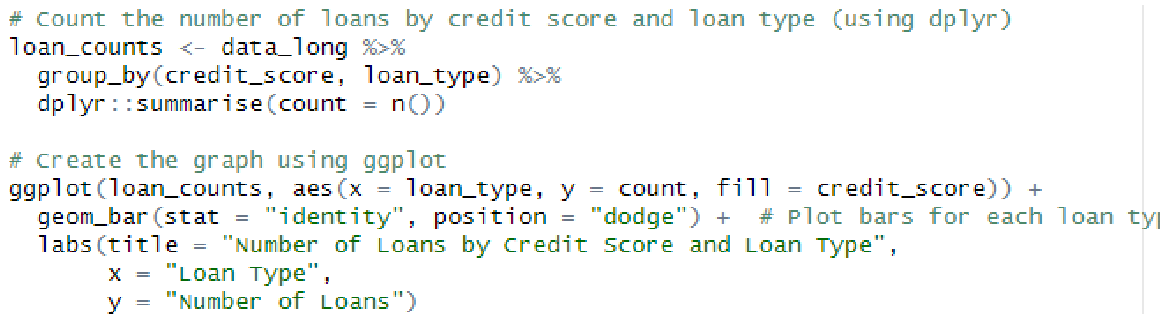
To achieve this, we must expand the type of loan from the long lists.



```
# Split data by type of loan and expand them
data_long <- pivot_longer(df_selected, cols = starts_with("loan_type"), names_to = "type",
                          values_to = "loan_type")
data_long = data_long[, !(names(data_long) %in% c("type", "num_of_loan"))]
data_long = data_long[complete.cases(data_long),]
data_long = distinct(data_long)
```

Figure 33: code for expanding loan type

Next, we will expand them, merging them into singular column named loan_type instead of 9 similar data plus removing type and num_of_loan which is not related to what we are trying to find. After removing empty data we remove duplicates ones again. This will result in having every loan type be compared to the customer's credit score.



```r
# Count the number of loans by credit score and loan type (using dplyr)
loan_counts <- data_long %>%
  group_by(credit_score, loan_type) %>%
  dplyr::summarise(count = n())

# Create the graph using ggplot
ggplot(loan_counts, aes(x = loan_type, y = count, fill = credit_score)) +
  geom_bar(stat = "identity", position = "dodge") +  # Plot bars for each loan type
  labs(title = "Number of Loans by Credit Score and Loan Type",
       x = "Loan Type",
       y = "Number of Loans")
```

Figure 34: Code for comparison between type of loan and credit score

Now to form a graph for the data, we first count every type of loan with their corresponding credit score rating. This will result with three data of credit_score, loan_type, and count. With this data we can create a graph suited for representing these data. After several tests, we had determined that using bar table is the most suitable for presenting the data.

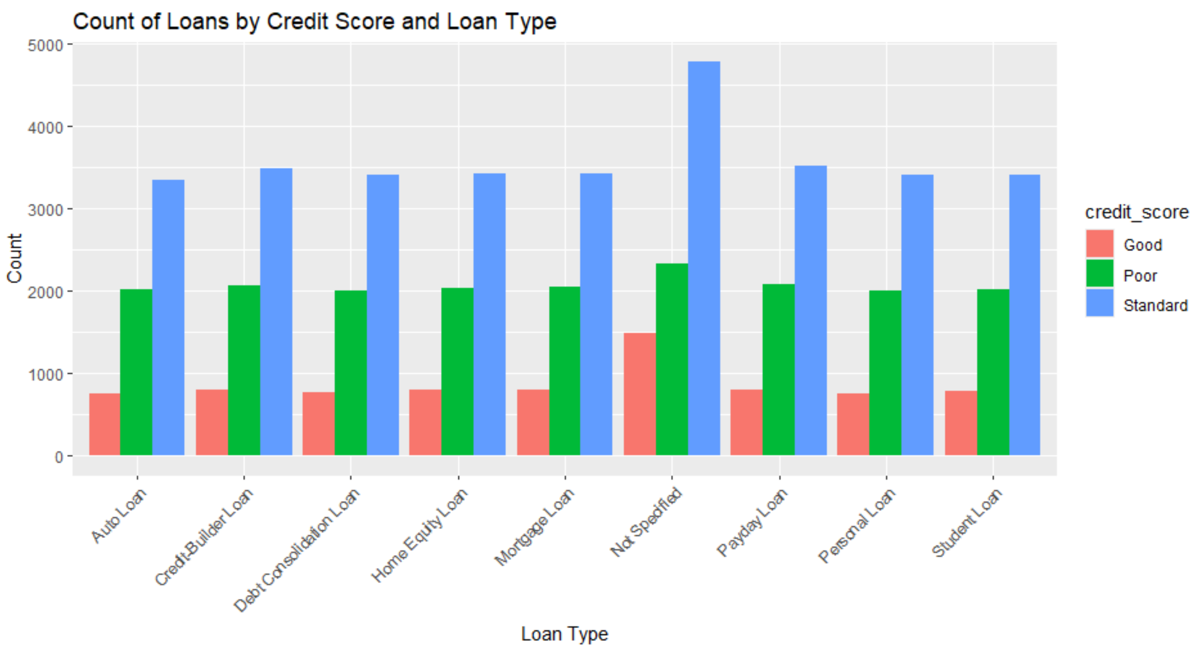
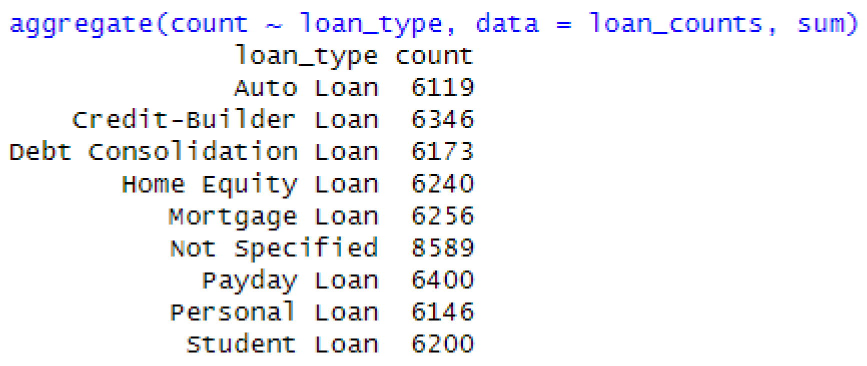


Figure 35: Type of loan compared to credit score

Data Analysis Method: Data Visualization

Disappointingly, as seen with the plot that there are no correlations between the type of loan and the credit score of a person. The only thing that we are able to learn from this graph is that most of the loan type are unspecified. With this thought lets also show the most common loan type.



```
aggregate(count ~ loan_type, data = loan_counts, sum)
               loan_type count
               Auto Loan  6119
     Credit-Builder Loan  6346
Debt Consolidation Loan  6173
        Home Equity Loan  6240
           Mortgage Loan  6256
           Not Specified  8589
             Payday Loan  6400
           Personal Loan  6146
            Student Loan  6200
```

Figure 36:

The summarization above shows that “Not Specified” loan type is indeed the largest portion of the registered loan. Following up by Payday loan and Credit-Builder loan as the third.

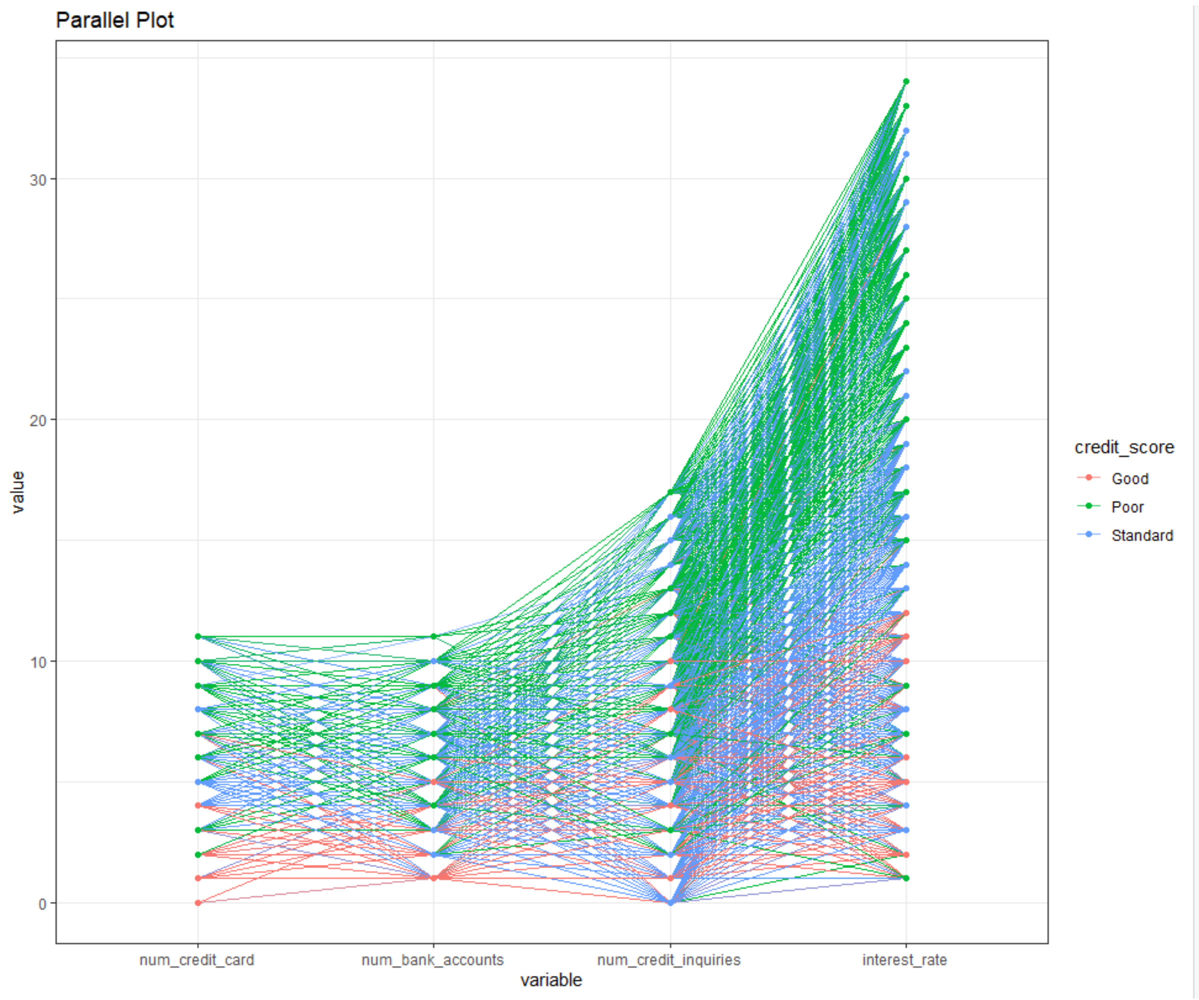
4.2.3 Conclusion

In the analysis done to find correlations between number of loan and credit score, we have found several discoveries but none that is significantly glaring. The first discovery is that the credit score of a person will drop drastically beyond having 4 loans whereas having a good credit score with 5 loans or above is nearly impossible. Its unfortunate that no correlation were found between type of loan and credit score, but we did discover that majority of loan type are unrecorded therefore being unspecified.

4.3 To investigate how the number of credit cards can affect each person's credit score. (Jeff Hong Chee Hin TP074560)

4.3.1 Does the number of credit cards will affect the credit score of a person with other three variables?

```
library(GGally)
ggparcoord(df,columns = c(11,10,18,12),  groupColumn = "credit_score",
           # transparency of lines
           alphaLines = 0.7,scale = "globalminmax",# scaling method
           showPoints = TRUE    # show data points) +
           # covered with black and white theme
           theme_bw()+
           # Title
           labs(title="Parallel Plot")
```



```
> max(df$num_credit_card)
[1] 11
> min(df$num_credit_card)
[1] 0
```

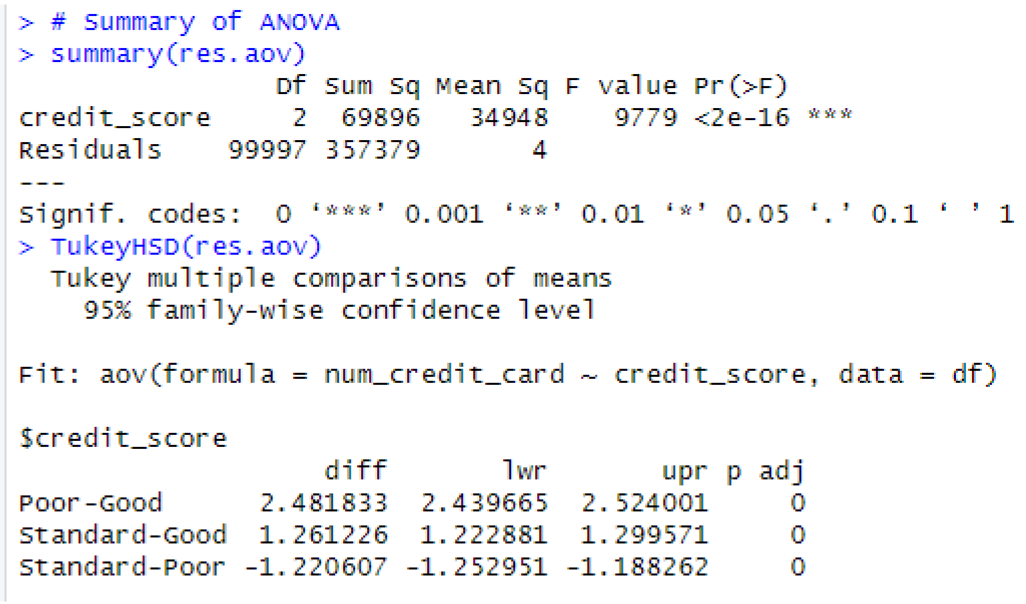
Figure 37: Parallel Coordinate Plot to find the category credit score of a person

Extra Features : Parallel Coordinate Plot

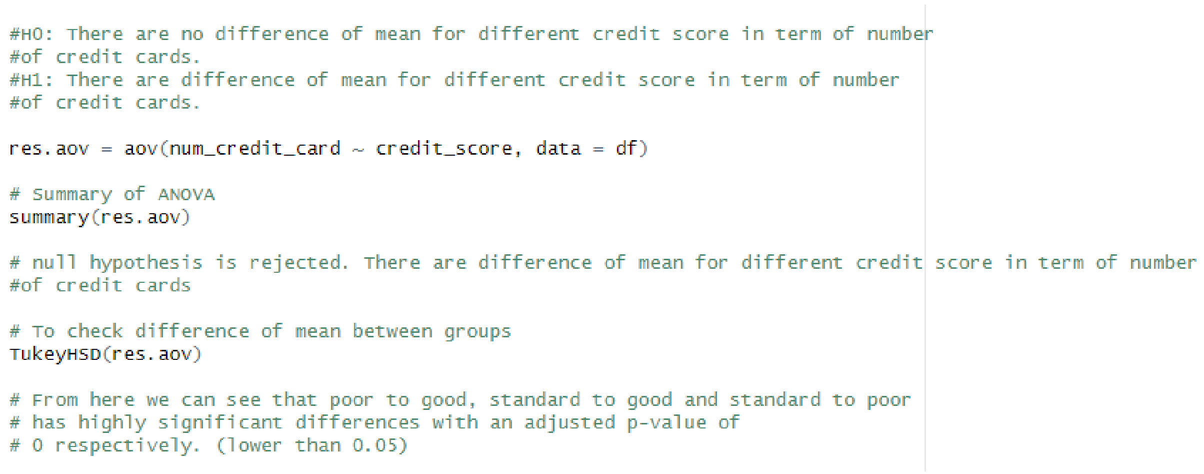
Data Analysis Method: Data Visualisation

The first implementation of the method to visualise the dataset of credit score classification dataset is the parallel coordinate plot. The parallel coordinate plot can show the relationship between multiple numeric variables and each of the findings is plotted horizontally (Gattu, 2021). This plot visualizes the relationships between credit scores and various factors: **the number of credit cards**, **bank accounts**, **credit inquiries**, and **interest rates** but **mostly focuses** on the various factors of the **number of credit cards**. When the **number of credit cards increases**, there are **many green colour lines** at the **top of the plot** representing that **people's credit scores are poor**, there correlation between the number of credit cards and a person's credit score value. The **red lines** usually at the **bottom of the plot** due to the l**esser of credit cards** will define **mostly people who have a good credit score**. According to the plot, the **credit score** of people who are **categorised as standard** mostly focused on the **number of credit cards around 5-8**. For example, the person **who doesn't have credit cards**, **only has one bank account**, around **five numbers of credit inquiries** and an **interest rate of around 2%-12%** will **represent** the person who has a **good credit score category** which the person can **apply the loan** from the bank **successfully**. In the plot findings, most people whose **credit scores** were **categorised as poorly** focused on the **green points** which having **several credit cards around 9-11**, including **bank accounts more than 4**, some **credit inquiries higher than 5**, and **interest rates above 19%** which they are **hard to apply for the loan from the bank**. According to the code given, the **minimum number of credit cards is 0** usually **categorised** as the person who has a **good credit score**. The m**aximum number of credit cards is 11**, which will mostly **categorised** in the **poor category of credit score.**

4.3.2 What is the difference in mean for different categories of credit scores in terms of the number of credit cards



```
> # Summary of ANOVA
> summary(res.aov)
                Df Sum Sq Mean Sq F value Pr(>F)    
credit_score     2  69896   34948    9779 <2e-16 ***
Residuals    99997 357379       4                   
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
> TukeyHSD(res.aov)
  Tukey multiple comparisons of means
    95% family-wise confidence level

Fit: aov(formula = num_credit_card ~ credit_score, data = df)

$credit_score
                   diff       lwr       upr p adj
Poor-Good      2.481833  2.439665  2.524001     0
Standard-Good  1.261226  1.222881  1.299571     0
Standard-Poor -1.220607 -1.252951 -1.188262     0
```



```
#H0: There are no difference of mean for different credit score in term of number
#of credit cards.
#H1: There are difference of mean for different credit score in term of number
#of credit cards.

res.aov = aov(num_credit_card ~ credit_score, data = df)

# Summary of ANOVA
summary(res.aov)

# null hypothesis is rejected. There are difference of mean for different credit score in term of number
#of credit cards

# To check difference of mean between groups
TukeyHSD(res.aov)

# From here we can see that poor to good, standard to good and standard to poor
# has highly significant differences with an adjusted p-value of
# 0 respectively. (lower than 0.05)
```

Figure 38: Summary of ANOVA and the codes

Extra Features : ANOVA and TukeyHSD

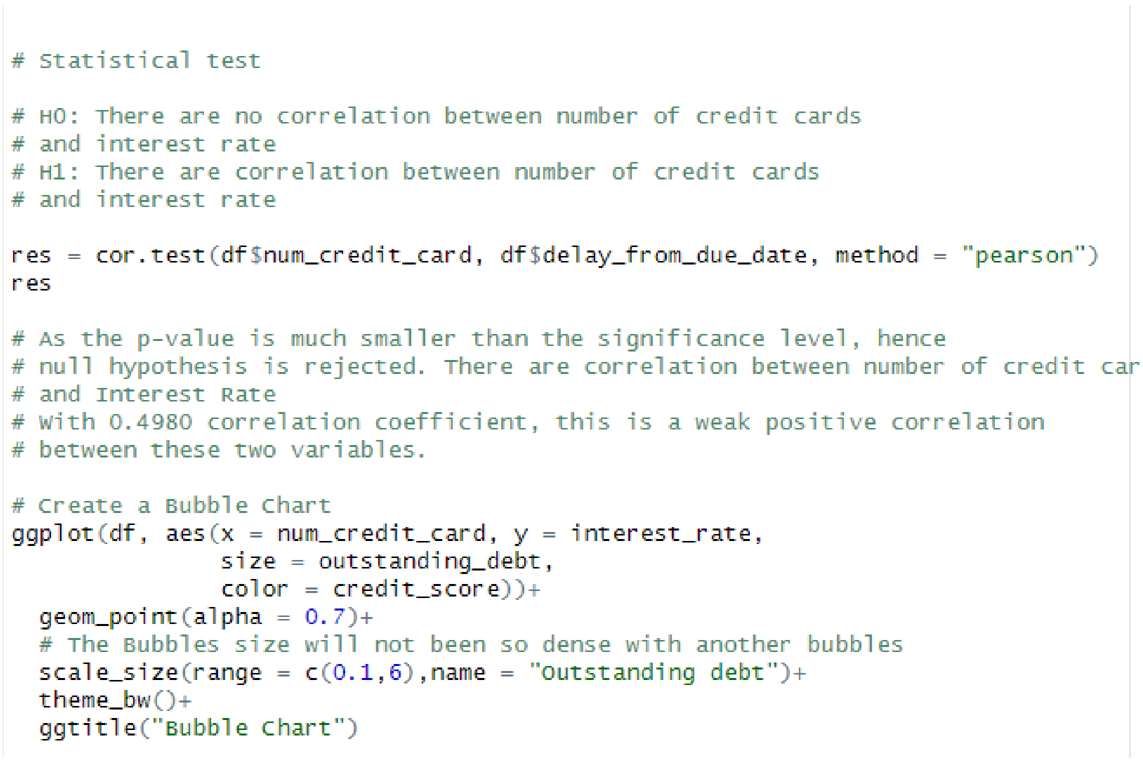
Data analysis method: Inferential Statistic

<u>Hypothesis Testing</u>

- **H0 -There is no difference in the mean for different credit scores regarding the number of credit cards.**
- **H1 - There is a difference in the mean for different credit scores regarding the number of credit cards.**

To investigate the difference in the mean number of credit cards across different credit score groups, using **hypothesis testing to determine if there is a statistically significant difference** between the **credit score groups**. We just set a **null hypothesis** which is **H0** with there being **no difference of mean for different credit scores in terms of the number of credit cards**. The **alternative hypothesis** which is **H1** is setting as there is the **difference of mean for different credit scores in terms of the number of credit cards**. From the **ANOVA testing**, we can see that the result shows that there is a **highly significant F-value**, and the **P-value is less than 0.05** which the value is **smaller than 2e^-16** (0.0000000000000002) and that's the difference of mean across the different credit groups. So, we can assume that the **null hypothesis is rejected**, and we can conclude that **there are differences of mean for different credit scores in terms of several credit cards**. Additionally, if the **ANOVA result is significant**, we can use the **TukeyHSD** to show the **significant difference in credit score groups**. Given the result, we can prove that the **poor credit score category** has **significantly fewer credit cards** than the **good credit score category**. The **standard credit score category** has h**igher statistical significance** than the **poor category** and **good category** of **credit scores**. All the **comparisons between these three groups** have brought the **p-adjusted value**, which is **0**, so we can analyze that the **p-adjusted value** is **highly statistically significant** because the **actual values differences** of these **three credit score groups** are **smaller than 0** and need to **show that as 0** cause the **p-value is extremely small**. It is also **strong evidence** due to the **highly significant p-value**. Due to the **larger dataset size**, **small differences** can become **highly statistically significant**.

4.3.3 How does the analysis of the impact of credit cards in the category of credit scores



```r
# Statistical test

# H0: There are no correlation between number of credit cards
# and interest rate
# H1: There are correlation between number of credit cards
# and interest rate

res = cor.test(df$num_credit_card, df$delay_from_due_date, method = "pearson")
res

# As the p-value is much smaller than the significance level, hence
# null hypothesis is rejected. There are correlation between number of credit cards
# and Interest Rate
# With 0.4980 correlation coefficient, this is a weak positive correlation
# between these two variables.

# Create a Bubble Chart
ggplot(df, aes(x = num_credit_card, y = interest_rate,
               size = outstanding_debt,
               color = credit_score))+
  geom_point(alpha = 0.7)+
  # The Bubbles size will not been so dense with another bubbles
  scale_size(range = c(0.1,6),name = "Outstanding debt")+
  theme_bw()+
  ggtitle("Bubble Chart")
```

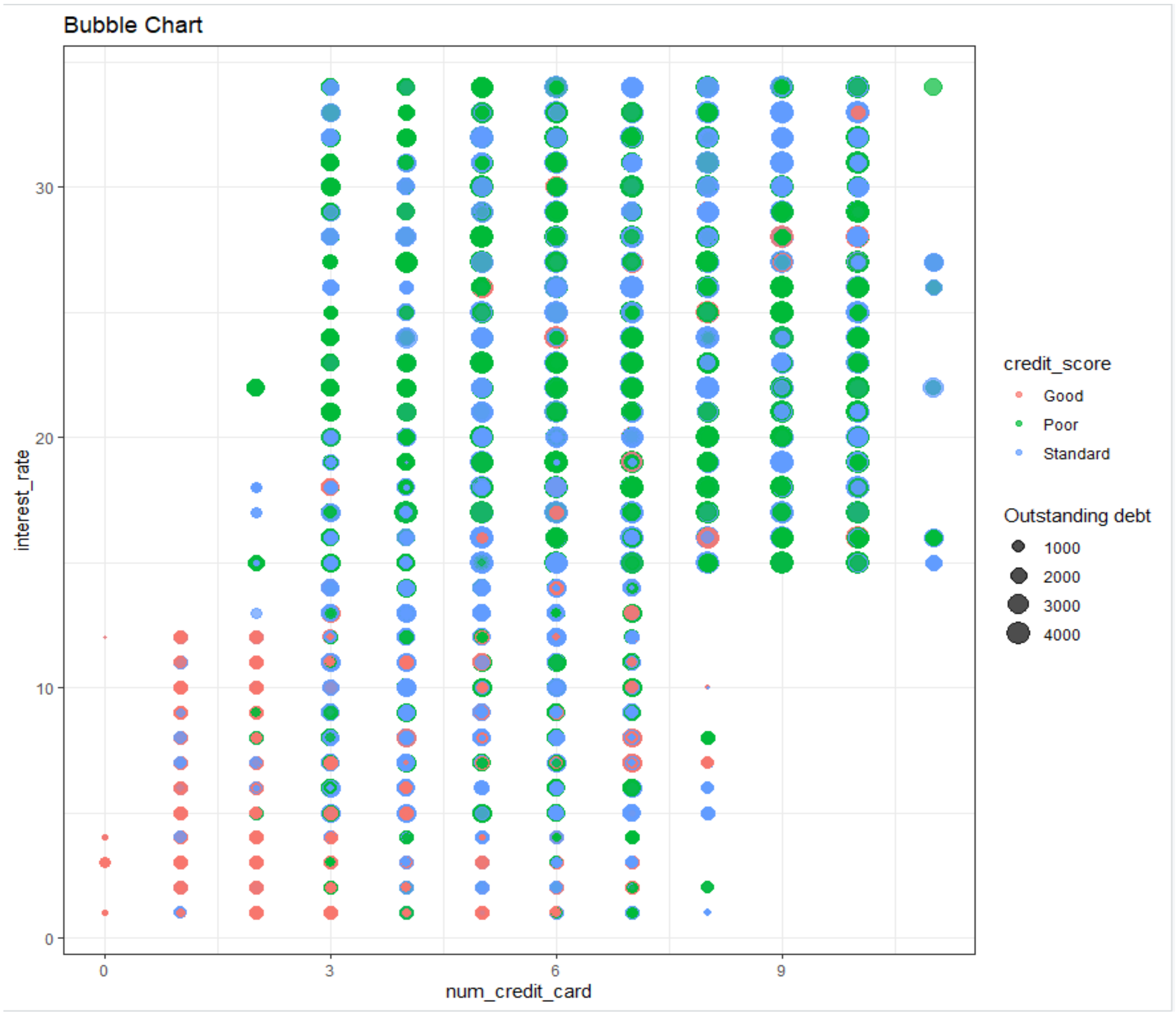


Figure 39: Summary of Statistical test, Bubble Charts and the codes

Extra Features : Bubble Chart and Statistical Test

Data analysis method: Data Visualization and Inferential Statistic

<u>Hypothesis Testing</u>

- **H0: There is no correlation between the number of credit cards and interest rate**
- **H1: There is a correlation between the number of credit cards and interest rate**

Before implementing the bubble chart for analysis of the relationship between credit score and several credit cards, it needs to **do a statistical test** for the **association between the two variables**, which are the **number of credit cards** and **interest rate**. The result shows that the **p-value is much smaller than** the **significance level**; hence, the **null hypothesis is rejected**, and **there is a correlation between them**. As the correlation coefficient shows, **0.4980 is a**

weak positive correlation, but this is the **highest correlation coefficient** compared to other variables. **Bubble charts use three numeric variables** to show the relationships between them to analyze the credit score. Based on the bubble chart, we can see that **red bubble represents a good credit score**, **blue bubble represents a standard credit score**, and **green bubble represents a poor credit score**. The **size of the bubbles** depends on the **outstanding debt value**. According to the chart, the **number of credit cards** is **less than 4**, and the **interest rate** is around **1%–12%**. **Most of the size of the bubbles** will be **small** due to the **outstanding debt of around 1000–2000**. On the **top of the bubble chart**, it shows **there are many green bubbles**, which **represent a poor credit score**. The **green bubbles** are **bigger than the red bubbles**, causing the **outstanding debt** to be **mostly around 3000–4000**. The **poor category of credit score** normally **has 3–10 credit cards**, and their **interest rate** is **higher compared** to the **good category of credit score**. The **standard category** of **credit score** is **blue bubbles**; they also have **quite fewer credit cards** and **a higher interest rate**, but they are in **between the poor** and **good categories**. The **standard category** of **credit score mostly** has **2000–4000 amounts of outstanding debt**, which makes the **scope of outstanding debts wider** than the **other two categories of credit score**. In a small conclusion, it is **analyzed that usually**, people with a **good credit score** will **have fewer credit cards**, **lower interest rates**, and **smaller amounts of outstanding debt** compared to people in people in the **poor category of credit score**, who have **fewer credit cards**, **a higher interest rate**, and a **larger amount of outstanding debt**.

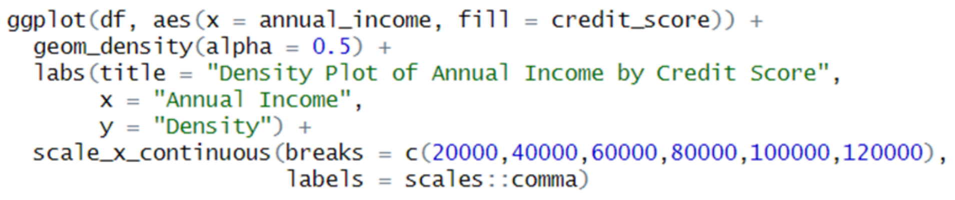
4.3.4 Conclusion

In conclusion, a person who **owns credit cards more than 5** will affect his credit score **category maintaining the poor category**. Meanwhile, a person who **owns 1-3 credit cards** makes their **credit score** stay in the **good category**. The **recommendation for finance** is encouraging people to have **fewer than 5 credit cards** to **avoid their credit scores becoming lower** because **many credit cards** will make a person to **apply many loans** and **delayed the payments** which **affects their credit score category**.

4.4 To investigate the relationship between annual income and a person’s credit score. (Ooi Yin Yao TP074427)

Three analysis methods are being used in this analysis. Density Plot, Logistic Regression, and Dunn's Test. Density Plots visualize data distribution. Logistic Regression predicts outcomes based on input features. Dunn's Test compares multiple groups to identify significant differences.

4.4.1 How does the annual income vary with the credit score?



```
ggplot(df, aes(x = annual_income, fill = credit_score)) +
  geom_density(alpha = 0.5) +
  labs(title = "Density Plot of Annual Income by Credit Score",
       x = "Annual Income",
       y = "Density") +
  scale_x_continuous(breaks = c(20000,40000,60000,80000,100000,120000),
                     labels = scales::comma)
```

Figure 40: Code for Density Plot

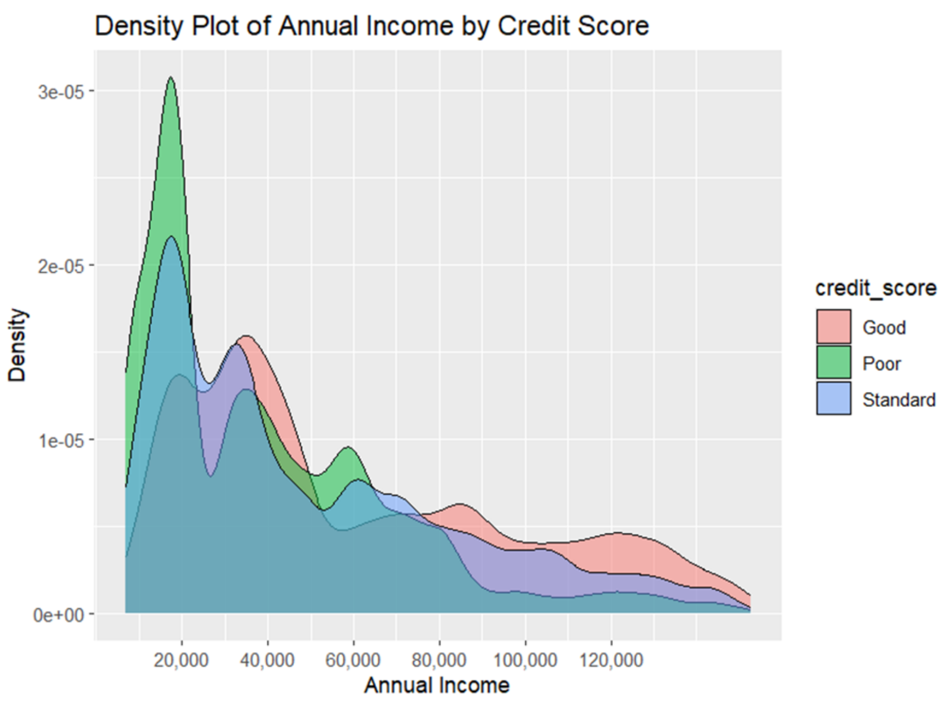


Figure 41: Density Plot

Extra Features: Density Plot

Data analysis method: Data Visualization

The density plot above shows a visualization of how annual income varies across different credit score categories: Good, Poor, and Standard. From the plot, it is evident that individuals with different credit scores exhibit distinct income distribution patterns.

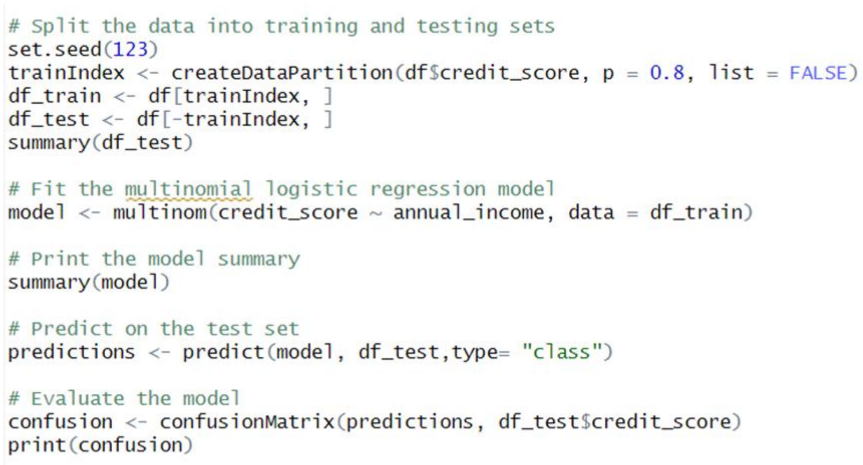
For those with a Poor credit score represented in green colour, the distribution is heavily skewed towards the lower end of the income, peaking around the $10,000 to $20,000 range. This indicates that a large proportion of individuals with poor credit scores have lower annual incomes. As the income increases beyond $30,000, the density for the poor credit group diminishes rapidly, showing fewer individuals with poor credit have higher incomes.

Individuals with a Standard credit score represented in blue colour. The density peaks between $20,000 and $40,000. However, it starts to drop when reaching $80,000, indicating a more diverse income range compared to the poor credit group.

For those with a Good credit score represented in red colour, the distribution tends to spread more towards higher income ranges. This group shows a significant rise in the $40,000 to $80,000 income range, and even beyond $100,000, the density remains noticeable, indicating a higher proportion of individuals with good credit scores also have higher annual incomes.

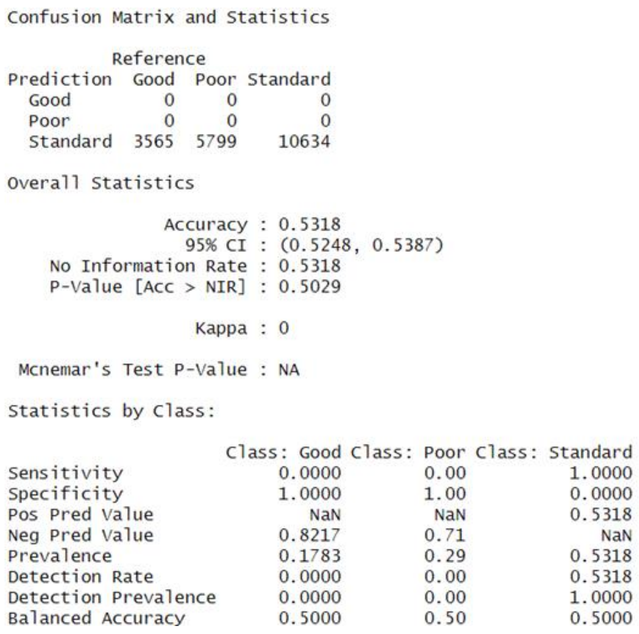
The density plot shows how annual income changes depending on credit scores. People with poor credit scores usually have lower annual income, while those with standard credit scores have incomes spread out more evenly across different levels. On the other hand, individuals with good credit scores tend to earn higher incomes. This suggests that having a higher income might help people maintain better credit scores.

4.4.2 Can the number of annual incomes predict the credit score?



```
# Split the data into training and testing sets
set.seed(123)
trainIndex <- createDataPartition(df$credit_score, p = 0.8, list = FALSE)
df_train <- df[trainIndex, ]
df_test <- df[-trainIndex, ]
summary(df_test)

# Fit the multinomial logistic regression model
model <- multinom(credit_score ~ annual_income, data = df_train)

# Print the model summary
summary(model)

# Predict on the test set
predictions <- predict(model, df_test,type= "class")

# Evaluate the model
confusion <- confusionMatrix(predictions, df_test$credit_score)
print(confusion)
```

Figure 42:Logistic Regression



```
Confusion Matrix and Statistics

          Reference
Prediction  Good  Poor Standard
  Good         0     0        0
  Poor         0     0        0
  Standard  3565  5799    10634

Overall Statistics

               Accuracy : 0.5318
                 95% CI : (0.5248, 0.5387)
    No Information Rate : 0.5318
    P-Value [Acc > NIR] : 0.5029

                  Kappa : 0

 Mcnemar's Test P-Value : NA

Statistics by Class:

                     Class: Good Class: Poor Class: Standard
Sensitivity               0.0000        0.00          1.0000
Specificity               1.0000        1.00          0.0000
Pos Pred Value               NaN         NaN          0.5318
Neg Pred Value            0.8217        0.71             NaN
Prevalence                0.1783        0.29          0.5318
Detection Rate            0.0000        0.00          0.5318
Detection Prevalence      0.0000        0.00          1.0000
Balanced Accuracy         0.5000        0.50          0.5000
```

Figure 43: Result of Logistic Regression

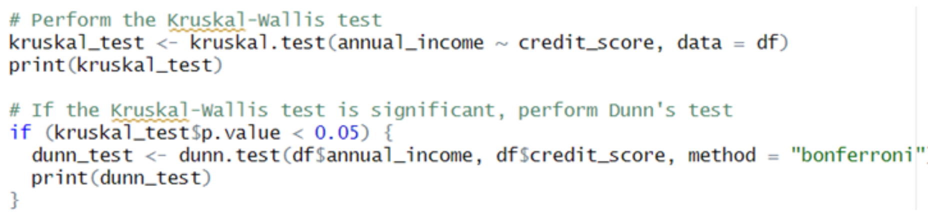
Extra Features: Logistic Regression

Data analysis method: Predictive Analysis

The confusion matrix and statistical summary provided above show the results of a logistic regression model that attempts to predict credit score categories (Good, Poor, Standard) based on annual income. The confusion matrix indicates that all predictions fall into the "Standard" category, regardless of the actual credit score. Overall statistics show an accuracy of 53.18, indicating that the model did not have a good performance. The p-value of 0.5029 further suggests that the model's accuracy is not statistically significant. The Kappa value is 0, which indicates no agreement between the predicted and actual classifications beyond what would be expected by chance. Examining the statistics by class, the sensitivity for both the Good and Poor classes is 0, meaning the model did not correctly identify any instances of these classes.

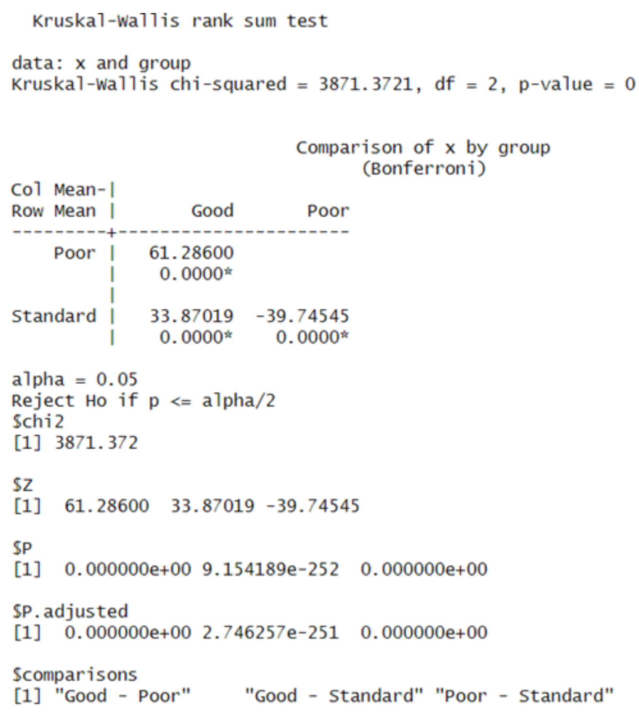
In summary, the logistic regression model's results indicate that using annual income alone is not an effective predictor of credit scores. The model failed to predict the annual income between the Good, Poor, and Standard credit score categories, instead predicting all annual income as Standard. This suggests that additional factors beyond annual income are necessary to accurately predict an individual's credit score.

4.4.3 Is there any relationship between credit score and annual income?



```
# Perform the Kruskal-Wallis test
kruskal_test <- kruskal.test(annual_income ~ credit_score, data = df)
print(kruskal_test)

# If the Kruskal-Wallis test is significant, perform Dunn's test
if (kruskal_test$p.value < 0.05) {
  dunn_test <- dunn.test(df$annual_income, df$credit_score, method = "bonferroni")
  print(dunn_test)
}
```

Figure 44:Dunn's Test



```
  Kruskal-Wallis rank sum test

data: x and group
Kruskal-Wallis chi-squared = 3871.3721, df = 2, p-value = 0


                           Comparison of x by group
                                 (Bonferroni)
Col Mean-|
Row Mean |       Good       Poor
---------+----------------------
    Poor |   61.28600
         |    0.0000*
         |
Standard |   33.87019  -39.74545
         |    0.0000*    0.0000*

alpha = 0.05
Reject Ho if p <= alpha/2
$chi2
[1] 3871.372

$Z
[1]  61.28600  33.87019 -39.74545

$P
[1]  0.000000e+00 9.154189e-252  0.000000e+00

$P.adjusted
[1]  0.000000e+00 2.746257e-251  0.000000e+00

$comparisons
[1] "Good - Poor"     "Good - Standard" "Poor - Standard"
```

Figure 45:Result of Dunn's Test

Extra Features: Dunn's Test

Data analysis method: Diagnostic Analysis

The findings of the Kruskal-Wallis test show that the annual incomes of the three credit score categories—Good, Standard, and Poor.

The mean rank differences between the groups are displayed in the comparison table. With a mean difference of 61, the mean rank of individuals with a poor credit score is much greater than that of the groups with good and standard credit scores. The Good group and the Standard group have a mean rank difference of 33. Additionally, the mean rank difference between the Standard group and the Poor group is -39.74545, showing that the ranks in the Standard group are much less. The statistical significance of the differences between all pairs of groups is confirmed by the incredibly low p-values, which are almost 0.

These findings illustrate a clear relationship between credit score and annual income. Specifically, individuals with poor credit scores tend to have lower annual incomes compared to those with standard or good credit scores. It shows that individuals with good credit scores are likely to have higher annual incomes. This suggests that higher income levels are associated with better credit scores.

4.4.4 Conclusion

After the above three different methods and data analysis from three different perspectives, it can be concluded that credit score and annual income have weak positive correlation. In the initial data analysis, the conclusion is that individuals with higher annual income will have better credit scores. However, in the subsequent analysis, it can be found that higher annual income does not mean better credit scores. On the contrary, it can be shown that people with lower annual income have a significantly lower credit score.

5.0 Conclusion

In the exploration of the given dataset via RStudio, we've made several understandings on the given dataset. Through the utilization of libraries such as dplyr, tidyr, along with R's other capabilities we can solve the data problems related to the dataset by cleaning up empty data, fixing corrupted data, modifying data to be more streamlined. After the data cleaning and corrections of data, each group members did their analysis based on their prior assumptions and hypotheses with extra features being implemented if needed.

Moving forward, there are several recommendations that can be used to enhance the effectiveness of the process. Firstly, focus should be put on investing in training and resources for expediting the learning process and empower users to be able to leverage the program to its full potential. Moreover, we should adopt practices for data organization to prevent the need for data fixing in the future.

The first glaring limitation is the obvious lack of experienced user in the works especially due to R language's learning curve on mastering R's syntax and libraries. Additionally, the given dataset has proven to be too big to weed out every wrong data as there might be existing data that are difficult to notice in being incorrect.

In terms of future directions, we should enhance the communication between the people responsible for collecting the information to ensure minimization on the frequency of data corruption or error. We should also push the continuation development of specialized packages for improved integration with other data management platforms.

6.0 Workload Matrix

Student Name	Task	Signature
Lee De Xian	Individual Data Analysis, Data Cleaning and Conclusion	[illegible]
Ooi Yin Yao	Individual Data Analysis, Introduction and Data Cleaning	[illegible]
Jeff Hong Chee Hin	Individual Data Analysis and Data Cleaning	[illegible]
Angelina Leanore	Individual Data Analysis and Data Exploration	[illegible]

7.0 References

Gattu, S. (2021, November 5). *Visualize data using parallel coordinates plot*. Analytics Vidhya. https://www.analyticsvidhya.com/blog/2021/11/visualize-data-using-parallel-coordinates-plot/

Steven P. Sanderson II, M. (2023, October 23). *How to create a bubble chart in R using ggplot2*. Steve's Data Tips and Tricks. https://www.spsanderson.com/steveondata/posts/2023-10-23/index.html

Gemignani, Z. (2021, September 15). *Better know a visualization: Understanding parallel coordinates charts - juice analytics*. Launch Customer-Facing Data Products - Juice Analytics. https://www.juiceanalytics.com/writing/writing/parallel-coordinates