DATA MINING

REPORT ON R

Exploratory Data Analysis

Assignment-1

Submitted by

Prem Atul Jethwa (1001861810)
Shubham Sharma (1001964524)
Lavanya Srinivasan (1002040671)

PA 1: Exploratory Analysis over Dataset_R

You will also learn how to use visualization libraries to identify patterns in data that will help in your further data analysis. You will also explore most popular chart types and how to use different libraries and styles to make your visualizations more attractive.

In [261]:
```
#Import R packages
import ggplot2()
import psych()
import Hmisc()
import dplyr()
install.packages(c("psych","GPArotation"),dependencies=TRUE)
install.packages("ctv") #this downloads the task view package
install.packages("chron")
install.packages("tidyverse")
install.packages("dplyr")
install.views("Psychometrics")
install.packages("ggplot2")
install.packages("rattle")
library(dplyr)
library(modeest)
library(ctv) #this activates the ctv package
library(psych)
library(tidyr)
library(tidyverse)
library(rpart)
library(rpart.plot)
library(ggplot2)
install.packages("zoo")
install.packages("reshape2")
```

In [264]:
```
# Read the dataset_R CSV file

df_data <-read.csv("Dataset_R.csv", stringsAsFactors = FALSE)

df_data
```

A data.frame: 614 × 13

Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History	Prop
<chr>	<chr>	<chr>	<chr>	<chr>	<chr>	<int>	<dbl>	<int>	<int>	<int>	
LP001002	Male	No	0	Graduate	No	5849	0	NA	360	1	
LP001003	Male	Yes	1	Graduate	No	4583	1508	128	360	1	
LP001005	Male	Yes	0	Graduate	Yes	3000	0	66	360	1	
LP001006	Male	Yes	0	Not Graduate	No	2583	2358	120	360	1	
LP001008	Male	No	0	Graduate	No	6000	0	141	360	1	
LP001011	Male	Yes	2	Graduate	Yes	5417	4196	267	360	1	
LP001013	Male	Yes	0	Not Graduate	No	2333	1516	95	360	1	
LP001014	Male	Yes	3+	Graduate	No	3036	2504	158	360	0	S

In [263]:
```
# return the first 10 rows of the dataset

head(df_data, n = 10)
```

A data.frame: 10 × 13

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History
	<chr>	<chr>	<chr>	<chr>	<chr>	<chr>	<int>	<dbl>	<int>	<int>	<int>
1	LP001002	Male	No	0	Graduate	No	5849	0	NA	360	1
2	LP001003	Male	Yes	1	Graduate	No	4583	1508	128	360	1
3	LP001005	Male	Yes	0	Graduate	Yes	3000	0	66	360	1
4	LP001006	Male	Yes	0	Not Graduate	No	2583	2358	120	360	1
5	LP001008	Male	No	0	Graduate	No	6000	0	141	360	1
6	LP001011	Male	Yes	2	Graduate	Yes	5417	4196	267	360	1
7	LP001013	Male	Yes	0	Not Graduate	No	2333	1516	95	360	1
8	LP001014	Male	Yes	3+	Graduate	No	3036	2504	158	360	0
9	LP001018	Male	Yes	2	Graduate	No	4006	1526	168	360	1
10	LP001020	Male	Yes	1	Graduate	No	12841	10968	349	360	1

Task 1: Statistical Exploratory Data Analysis(17.5 points)

In [242]:
```
# 1-a Print the details of dataframe and find out the number of rows and columns in dataset

print("1-a The details of dataframe:")
str(df_data)
print("#1-a The number of rows in dataset:")
print(nrow(df_data))
print("#1-a The number of columns in dataset:")
print(ncol(df_data))
```

```
[1] "1-a The details of dataframe:"
'data.frame':   614 obs. of  13 variables:
 $ Loan_ID           : chr  "LP001002" "LP001003" "LP001005" "LP001006" ...
 $ Gender            : chr  "Male" "Male" "Male" "Male" ...
 $ Married           : chr  "No" "Yes" "Yes" "Yes" ...
 $ Dependents        : chr  "0" "1" "0" "0" ...
 $ Education         : chr  "Graduate" "Graduate" "Graduate" "Not Graduate" ...
 $ Self_Employed     : chr  "No" "No" "Yes" "No" ...
 $ ApplicantIncome   : int  5849 4583 3000 2583 6000 5417 2333 3036 4006 12841 ...
 $ CoapplicantIncome: num  0 1508 0 2358 0 ...
 $ LoanAmount        : int  NA 128 66 120 141 267 95 158 168 349 ...
 $ Loan_Amount_Term : int  360 360 360 360 360 360 360 360 360 360 ...
 $ Credit_History    : int  1 1 1 1 1 1 1 0 1 1 ...
 $ Property_Area     : chr  "Urban" "Rural" "Urban" "Urban" ...
 $ Loan_Status       : chr  "Y" "N" "Y" "Y" ...
[1] "#1-a The number of rows in dataset:"
[1] 614
[1] "#1-a The number of columns in dataset:"
[1] 13
```

In [243]:
```
# 1-b Print descriptive detail of a column in dataset

print("#1-b Descriptive detail of a column in dataset:")
print(summary(df_data))
```

```
[1] "#1-b Descriptive detail of a column in dataset:"
   Loan_ID             Gender            Married           Dependents
 Length:614         Length:614         Length:614         Length:614
 Class :character   Class :character   Class :character   Class :character
 Mode  :character   Mode  :character   Mode  :character   Mode  :character



  Education          Self_Employed      ApplicantIncome CoapplicantIncome
 Length:614         Length:614         Min.    :  150   Min.   :    0
 Class :character   Class :character   1st Qu.: 2878   1st Qu.:    0
 Mode  :character   Mode  :character   Median : 3812   Median : 1188
                                       Mean    : 5403   Mean   : 1621
                                       3rd Qu.: 5795   3rd Qu.: 2297
                                       Max.    :81000   Max.   :41667

   LoanAmount    Loan_Amount_Term Credit_History   Property_Area
 Min.    :  9.0   Min.    : 12      Min.   :0.0000   Length:614
 1st Qu.:100.0   1st Qu.:360       1st Qu.:1.0000   Class :character
 Median :128.0   Median :360       Median :1.0000   Mode  :character
 Mean   :146.4   Mean   :342       Mean   :0.8422
 3rd Qu.:168.0   3rd Qu.:360       3rd Qu.:1.0000
 Max.   :700.0   Max.   :480       Max.   :1.0000
 NA's   :22      NA's   :14        NA's   :50
 Loan_Status
 Length:614
 Class :character
 Mode  :character
```

In [244]:
```
# 1-c Check for missing values for each column in dataset

print("#1-c Total number of missing values for each column in dataset:")
missing_values <- apply(df_data, 2, function(col) sum(is.na(col)| col == ""))
print(missing_values)
```

```
[1] "#1-c Total number of missing values for each column in dataset:"
          Loan_ID            Gender           Married        Dependents
                0                13                 3                15
        Education     Self_Employed   ApplicantIncome CoapplicantIncome
                0                32                 0                 0
       LoanAmount  Loan_Amount_Term    Credit_History     Property_Area
               22                14                50                 0
      Loan_Status
                0
```

In [246]:

```
# 1-d Data Pre-processing for R
# - handle null values, categorical data, missing data..etc.

# To check if the data set has any null values, you can use the is.na() function:
print("Total number of null values in the entire data set:")
print(sum(is.na(df_data)))

# If there are null values, you can remove them using the na.omit() function:
df_nulldata <- na.omit(df_data)
print("Total number of rows in data set after removing null values:")
print(nrow(df_nulldata))
print("New Data Set after removing null values is as below:")
df_nulldata

#-----------------------------------------------------------------------------------------------------------------------
# To convert categorical data to factors, you can use the factor() function:
# The categorical data is typically stored as character or string data type.
# However, many R functions and packages expect categorical data to be stored as factors. Factors are a special
# data type in R that are used to represent categorical data with a fixed set of possible values, or levels.

df_catdata <- df_data
df_catdata$Gender <- factor(df_catdata$Gender)
df_catdata$Loan_Status <- factor(df_catdata$Loan_Status)

# Once you have converted your categorical data to factors, you can use them in various R functions and
# packages that expect factors as input. For example, you can use factors in statistical models,
# plot them using ggplot2, and calculate summary statistics using dplyr.

# To create dummy variables for categorical data, you can use the model.matrix() function:
dummy_vars <- model.matrix(~ Gender + Loan_Status, data = df_catdata)
print("Display the dummy variable in the data set:")
dummy_vars

#-----------------------------------------------------------------------------------------------------------------------
# To fill in missing values with the last observed value, you can use the na.locf() function from the zoo package:
# Here we filled the missing values in Credit_History column with the help of na.locf()
#install.packages("zoo")

library(zoo)
df_missingdata <- df_data
df_missingdata$Credit_History <- na.locf(df_missingdata$Credit_History)
print("New Data Set after filling missing values in Credit_History column is as below:")
df_missingdata
```

```
[1] "Total number of null values in the entire data set:"
[1] 86
[1] "Total number of rows in data set after removing null values:"
[1] 529
[1] "New Data Set after removing null values is as below:"
```

A data.frame: 529 × 13

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History
	<chr>	<chr>	<chr>	<chr>	<chr>	<chr>	<int>	<dbl>	<int>	<int>	<int>
2	LP001003	Male	Yes	1	Graduate	No	4583	1508	128	360	1
3	LP001005	Male	Yes	0	Graduate	Yes	3000	0	66	360	1
4	LP001006	Male	Yes	0	Not Graduate	No	2583	2358	120	360	1
5	LP001008	Male	No	0	Graduate	No	6000	0	141	360	1
6	LP001011	Male	Yes	2	Graduate	Yes	5417	4196	267	360	1

Task 2: Aggregation & Filtering & Rank (30 points)

In [247]:
```
# Task 2-a: Find out the number of female graduates from Semiurban area

print("Task 2-a The number of female graduates from Semiurban area:")
print(nrow(df_data[df_data$Property_Area=='Semiurban'& df_data$Education=='Graduate' & df_data$Gender=='Female', ]))
```

```
[1] "Task 2-a The number of female graduates from Semiurban area:"
[1] 45
```

In [248]:
```
# Task 2-b: Determine the overall number of men with more than 3 Dependents, who did not graduate with Self employment

print("Task 2-b The overall number of men with more than 3 Dependents, who did not graduate with Self employment:")
print(nrow(df_data[df_data$Gender=='Male'& df_data$Dependents=='3+' & df_data$Education=='Not Graduate' & df_data$Self_
```

```
[1] "Task 2-b The overall number of men with more than 3 Dependents, who did not graduate with Self employment:"
[1] 3
```

In [249]:
```
# Task 2-c: Find the top 10 non-married female applicants who graduated with self employment and had the highest applic
# The data set only has 8 applicants under this category

print("Task 2c- The top 10 non-married female applicants who graduated with self employment and had the highest applica
library(magrittr)
library(dplyr)
# Filter for non-married females who graduated with self employment
filtered_data <- df_data %>%
  filter(df_data$Gender == 'Female' & df_data$Married == 'No' & df_data$Education == 'Graduate' & df_data$Self_Employed

# Sort by applicant income in descending order
sorted_data <- filtered_data %>%
  arrange(desc(ApplicantIncome))

# Select the top 10 rows
top_10 <- head(sorted_data, 10)

# Print the result
top_10
```

```
[1] "Task 2c- The top 10 non-married female applicants who graduated with self employment and had the highest applica
nt income"
```

A data.frame: 8 × 13

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History	F
	<chr>	<chr>	<chr>	<chr>	<chr>	<chr>	<int>	<dbl>	<int>	<int>	<int>	
1	LP002194	Female	No	0	Graduate	Yes	15759	0	55	360	1	
2	LP002377	Female	No	1	Graduate	Yes	8624	0	150	360	1	
3	LP001392	Female	No	1	Graduate	Yes	7451	0	NA	360	1	
4	LP002301	Female	No	0	Graduate	Yes	7441	0	194	360	1	
5	LP002990	Female	No	0	Graduate	Yes	4583	0	133	360	0	
6	LP001788	Female	No	0	Graduate	Yes	3463	0	122	360	NA	
7	LP001925	Female	No	0	Graduate	Yes	2600	1717	99	300	1	
8	LP002522	Female	No	0	Graduate	Yes	2500	0	93	360	NA	

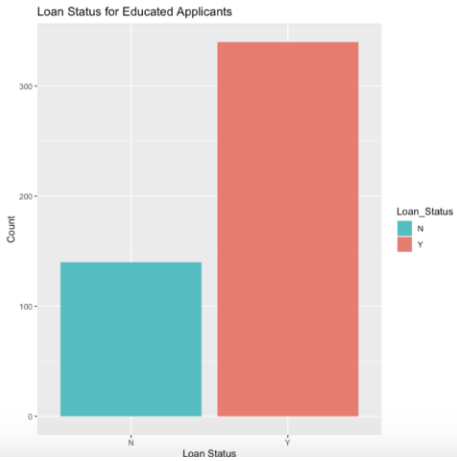
In [250]:
```
Task 2-d: Find the number of self-employed male applicants from urban area with exactly two dependents

rint("Task 2-d The number of self-employed male applicants from urban area with exactly two dependents:")
rint(nrow(df_data[df_data$Property_Area=='Urban'& df_data$Dependents=='2' & df_data$Self_Employed=='Yes' & df_data$Gende
```

```
[1] "Task 2-d The number of self-employed male applicants from urban area with exactly two dependents:"
[1] 3
```

Task 3: Visualization (30 points)

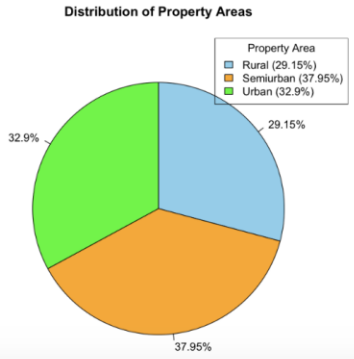
In [251]:
```
# Task 3-a Display a plot where educated applicants are granted loans

print("Task 3a : Plot where educated applicants are granted loans:")

library(ggplot2)

# create a subset of data for only educated applicants
educated_df <- df_data[df_data$Education == "Graduate", ]

# create a bar plot
ggplot(data = educated_df, aes(x = Loan_Status, fill = Loan_Status)) +
  geom_bar() +
  ggtitle("Loan Status for Educated Applicants") +
  xlab("Loan Status") +
  ylab("Count") +
  scale_fill_manual(values = c("#00BFC4", "#F8766D")) # set custom colors for fill
```

```
[1] "Task 3a : Plot where educated applicants are granted loans:"
```



In [260]:
```r
# Task 3-b Create a pie chart for Property_area and display percentages in legend respectively

print("Task 3b : Pie chart for Property_area and display percentages in legend respectively:")

library(ggplot2)

# calculate percentages for each category
prop_table <- prop.table(table(df_data$Property_Area))
percentages <- paste0(round(prop_table * 100, 2), "%")

# create the pie chart
pie(prop_table,
    labels = percentages,
    col = c("skyblue", "orange", "green"),
    main = "Distribution of Property Areas",
    clockwise = TRUE)

# add legend with percentages
legend("topright", legend = paste0(names(prop_table), " (", percentages, ")"),
       fill = c("skyblue", "orange", "green"), title = "Property Area")
```

```
[1] "Task 3b : Pie chart for Property_area and display percentages in legend respectively:"
```



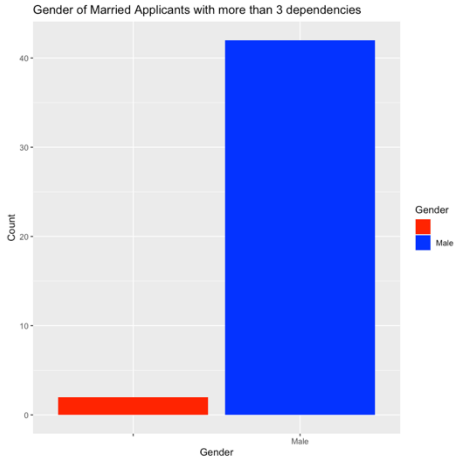
Task 4: Insights from the data (20 points)

Find out 'interesting' information from the dataset. Give two insights and Create a visualization for each of the insights. Explain in a few lines your reasoning.

Your work's uniqueness and quality will be taken into account when evaluating your work (having a meaningful result and an aesthetic visualization).

In [253]:
```r
# Code and explaination for Task4
# Task 4-a Plot of married applicants with more than 3 dependencies:

# We observe that there are no Female Applicants who are married and have more than 3 dependencies. That is
# we only have in total 44 Applicants under this category which are 42 Males applicants (Blue bar) and
# 2 missing values (Red bar).

# Bar plot: A bar plot displays the distribution of a categorical variable.
# This can be created using the geom_bar() function.

print("Task 4a : Plot of married applicants with more than 3 dependencies:")

library(ggplot2)

# create a subset of data for only educated applicants
married_df <- df_data[df_data$Married == "Yes" & df_data$Dependents == "3+", ]

# create a bar plot
ggplot(data = married_df, aes(x = Gender, fill = Gender)) +
  geom_bar() +
  ggtitle("Gender of Married Applicants with more than 3 dependencies") +
  xlab("Gender") +
  ylab("Count") +
  scale_fill_manual(values = c("Red", "Blue")) # set custom colors for fill
```

```
[1] "Task 4a : Plot of married applicants with more than 3 dependencies:"
```



In [259]:
```r
# Code and explaination for Task4
# Task 4-b To create a heatmap plot of the property area of the applicants whose loan is granted.

# Heatmap: A heatmap displays the relationship between two categorical variables by using color to represent the
# frequency or proportion of observations in each combination of categories. This can be created using the
# geom_tile() function.

# This will produce a heatmap plot with "Property Area" on the x-axis and "Frequency" on the y-axis, and
# the title "Property Area of Approved Loans". The color scale ranges from white (low frequency) to
# steelblue (high frequency), with each category of the "Property_Area" variable represented by a tile.
# The "theme" function is used to rotate the x-axis labels by 90 degrees to improve readability.
# The plot is based on the subset of the df_data where the loan is granted.

print("Task 4b : Heatmap plot of the property area of the applicants whose loan is approved:")

#Load the "ggplot2" and "reshape2" packages:
#install.packages("reshape2")
library(ggplot2)
library(reshape2)

#Create a subset of the loan data where the loan status is "Y":
loan_y <- subset(df_data, Loan_Status == "Y")

#Create a frequency table of the "Property_Area" variable in the loan subset:
property_freq <- table(loan_y$Property_Area)

#Convert the frequency table to a data frame and rename the columns:
property_df <- data.frame(property_freq)
colnames(property_df) <- c("Property_Area", "Frequency")

#Melt the data frame to create a format that can be used for the heatmap:
property_melt <- melt(property_df, id.vars = "Property_Area")

#Create the heatmap plot:
ggplot(property_melt, aes(x = Property_Area, y = variable)) +
  geom_tile(aes(fill = value), color = "white") +
  scale_fill_gradient(low = "white", high = "Dark Green") +
  theme(axis.text.x = element_text(angle = 90, vjust = 0.5, hjust = 1)) +
  labs(title = "Property Area of Approved Loans",
       x = "Property Area", y = "Frequency")
```

```
[1] "Task 4b : Heatmap plot of the property area of the applicants whose loan is approved:"
```

References:

https://www.tutorialspoint.com/r/index.htm

https://www.youtube.com/results?search_query=r+programming+

https://www.geeksforgeeks.org/r-programming-language-introduction/

https://r4ds.had.co.nz/

https://www.tutorialspoint.com/jupyter/jupyter_notebook_markdown_cells.htm

Contribution:

Python – Lavanya Srinivasan (1002040671)

R language – Prem Atul Jethwa (1001861810)

Weka – Shubham Sharma (1001964524)