FDS-A2: Data Wrangling

Suman Paudel

Task 1

```
library(tidyverse)
library(magrittr)
library(reticulate)
library(scales)
use_python("C:\\Python311\\python.exe")
```

Load the Data

```
# load the dataset
auto_data <- read.csv("automobile.data", header = FALSE, na.strings = "?")
```

Basic Data Exploration

```
# Basic Data exploration
# Check dimensions
dim(auto_data)
```

```
## [1] 205  26
```

```
# View first few rows
head(auto_data)
```

```
##   V1  V2          V3  V4  V5   V6          V7  V8    V9  V10   V11  V12  V13
## 1  3  NA alfa-romero gas std  two convertible rwd front 88.6 168.8 64.1 48.8
## 2  3  NA alfa-romero gas std  two convertible rwd front 88.6 168.8 64.1 48.8
## 3  1  NA alfa-romero gas std  two   hatchback rwd front 94.5 171.2 65.5 52.4
## 4  2 164        audi gas std four       sedan fwd front 99.8 176.6 66.2 54.3
## 5  2 164        audi gas std four       sedan 4wd front 99.4 176.6 66.4 54.3
## 6  2  NA        audi gas std  two       sedan fwd front 99.8 177.3 66.3 53.1
##    V14  V15  V16 V17  V18  V19  V20  V21 V22  V23 V24 V25   V26
## 1 2548 dohc four 130 mpfi 3.47 2.68  9.0 111 5000  21  27 13495
## 2 2548 dohc four 130 mpfi 3.47 2.68  9.0 111 5000  21  27 16500
## 3 2823 ohcv  six 152 mpfi 2.68 3.47  9.0 154 5000  19  26 16500
## 4 2337  ohc four 109 mpfi 3.19 3.40 10.0 102 5500  24  30 13950
## 5 2824  ohc five 136 mpfi 3.19 3.40  8.0 115 5500  18  22 17450
## 6 2507  ohc five 136 mpfi 3.19 3.40  8.5 110 5500  19  25 15250
```

```
# Check data types
str(auto_data)
```

```
## ’data.frame’:    205 obs. of  26 variables:
##  $ V1 : int  3 3 1 2 2 2 1 1 1 0 ...
##  $ V2 : int  NA NA NA 164 164 NA 158 NA 158 NA ...
##  $ V3 : chr  "alfa-romero" "alfa-romero" "alfa-romero" "audi" ...
##  $ V4 : chr  "gas" "gas" "gas" "gas" ...
##  $ V5 : chr  "std" "std" "std" "std" ...
##  $ V6 : chr  "two" "two" "two" "four" ...
##  $ V7 : chr  "convertible" "convertible" "hatchback" "sedan" ...
##  $ V8 : chr  "rwd" "rwd" "rwd" "fwd" ...
##  $ V9 : chr  "front" "front" "front" "front" ...
##  $ V10: num  88.6 88.6 94.5 99.8 99.4 ...
##  $ V11: num  169 169 171 177 177 ...
##  $ V12: num  64.1 64.1 65.5 66.2 66.4 66.3 71.4 71.4 71.4 67.9 ...
##  $ V13: num  48.8 48.8 52.4 54.3 54.3 53.1 55.7 55.7 55.9 52 ...
##  $ V14: int  2548 2548 2823 2337 2824 2507 2844 2954 3086 3053 ...
##  $ V15: chr  "dohc" "dohc" "ohcv" "ohc" ...
##  $ V16: chr  "four" "four" "six" "four" ...
##  $ V17: int  130 130 152 109 136 136 136 136 131 131 ...
##  $ V18: chr  "mpfi" "mpfi" "mpfi" "mpfi" ...
##  $ V19: num  3.47 3.47 2.68 3.19 3.19 3.19 3.19 3.19 3.13 3.13 ...
##  $ V20: num  2.68 2.68 3.47 3.4 3.4 3.4 3.4 3.4 3.4 3.4 ...
##  $ V21: num  9 9 9 10 8 8.5 8.5 8.5 8.3 7 ...
##  $ V22: int  111 111 154 102 115 110 110 110 140 160 ...
##  $ V23: int  5000 5000 5000 5500 5500 5500 5500 5500 5500 5500 ...
##  $ V24: int  21 21 19 24 18 19 19 19 17 16 ...
##  $ V25: int  27 27 26 30 22 25 25 25 20 22 ...
##  $ V26: int  13495 16500 16500 13950 17450 15250 17710 18920 23875 NA ...
```

```
# Summary statistics
summary(auto_data)
```

```
##        V1                V2            V3                 V4           
##   Min.   :-2.0000   Min.   : 65   Length:205         Length:205        
##   1st Qu.: 0.0000   1st Qu.: 94   Class :character   Class :character  
##   Median : 1.0000   Median :115   Mode  :character   Mode  :character  
##   Mean   : 0.8341   Mean   :122                                        
##   3rd Qu.: 2.0000   3rd Qu.:150                                        
##   Max.   : 3.0000   Max.   :256                                        
##                     NA’s   :41                                         
##       V5                 V6                 V7                 V8           
##   Length:205         Length:205         Length:205         Length:205        
##   Class :character   Class :character   Class :character   Class :character  
##   Mode  :character   Mode  :character   Mode  :character   Mode  :character  
##
##
##
##
##       V9                 V10              V11             V12       
##   Length:205         Min.   : 86.60   Min.   :141.1   Min.   :60.30  
##   Class :character   1st Qu.: 94.50   1st Qu.:166.3   1st Qu.:64.10  
##   Mode  :character   Median : 97.00   Median :173.2   Median :65.50  
##                      Mean   : 98.76   Mean   :174.0   Mean   :65.91  
##                      3rd Qu.:102.40   3rd Qu.:183.1   3rd Qu.:66.90  
##                      Max.   :120.90   Max.   :208.1   Max.   :72.30  
```

```
##
##       V13             V14           V15                V16
##  Min.   :47.80   Min.   :1488   Length:205         Length:205
##  1st Qu.:52.00   1st Qu.:2145   Class :character   Class :character
##  Median :54.10   Median :2414   Mode  :character   Mode  :character
##  Mean   :53.72   Mean   :2556
##  3rd Qu.:55.50   3rd Qu.:2935
##  Max.   :59.80   Max.   :4066
##
##       V17            V18                 V19             V20
##  Min.   : 61.0   Length:205         Min.   :2.54   Min.   :2.070
##  1st Qu.: 97.0   Class :character   1st Qu.:3.15   1st Qu.:3.110
##  Median :120.0   Mode  :character   Median :3.31   Median :3.290
##  Mean   :126.9                      Mean   :3.33   Mean   :3.255
##  3rd Qu.:141.0                      3rd Qu.:3.59   3rd Qu.:3.410
##  Max.   :326.0                      Max.   :3.94   Max.   :4.170
##                                     NA's   :4      NA's   :4
##       V21             V22             V23            V24             V25
##  Min.   : 7.00   Min.   : 48.0   Min.   :4150   Min.   :13.00   Min.   :16.00
##  1st Qu.: 8.60   1st Qu.: 70.0   1st Qu.:4800   1st Qu.:19.00   1st Qu.:25.00
##  Median : 9.00   Median : 95.0   Median :5200   Median :24.00   Median :30.00
##  Mean   :10.14   Mean   :104.3   Mean   :5125   Mean   :25.22   Mean   :30.75
##  3rd Qu.: 9.40   3rd Qu.:116.0   3rd Qu.:5500   3rd Qu.:30.00   3rd Qu.:34.00
##  Max.   :23.00   Max.   :288.0   Max.   :6600   Max.   :49.00   Max.   :54.00
##                  NA's   :2       NA's   :2
##       V26
##  Min.   : 5118
##  1st Qu.: 7775
##  Median :10295
##  Mean   :13207
##  3rd Qu.:16500
##  Max.   :45400
##  NA's   :4
```

Data Wrangling

```
# there was no header in the dataset I had to change the column names
# change the names of columns of the auto_data dataframe
colnames(auto_data) <- c("symboling", "normalized_losses", "make", "fuel_type",
                         "aspiration", "num_doors", "body_style",
                         "drive_wheels", "engine_location", "wheel_base",
                         "length", "width", "height", "curb_weight",
                         "engine_type", "num_cylinders", "engine_size",
                         "fuel_system", "bore", "stroke", "compression_ratio",
                         "horsepower", "peak_rpm", "city_mpg",
                         "highway_mpg", "price")
colnames(auto_data)
```

```
##  [1] "symboling"         "normalized_losses" "make"
##  [4] "fuel_type"         "aspiration"        "num_doors"
##  [7] "body_style"        "drive_wheels"      "engine_location"
## [10] "wheel_base"        "length"            "width"
## [13] "height"            "curb_weight"       "engine_type"
## [16] "num_cylinders"     "engine_size"       "fuel_system"      
## [19] "bore"              "stroke"            "compression_ratio"
## [22] "horsepower"        "peak_rpm"          "city_mpg"         
## [25] "highway_mpg"       "price"
```

```
# Check for missing values
colSums(is.na(auto_data))
```

```
##         symboling normalized_losses              make         fuel_type 
##                 0                41                 0                 0 
##        aspiration         num_doors        body_style      drive_wheels 
##                 0                 2                 0                 0 
##   engine_location        wheel_base            length             width 
##                 0                 0                 0                 0 
##            height       curb_weight       engine_type     num_cylinders 
##                 0                 0                 0                 0 
##       engine_size       fuel_system              bore            stroke 
##                 0                 0                 4                 4 
## compression_ratio        horsepower          peak_rpm          city_mpg 
##                 0                 2                 2                 0 
##       highway_mpg             price 
##                 0                 4
```

```
# Impute the missing values
auto_data <- auto_data %>%
  mutate(
    num_doors = ifelse(is.na(num_doors), names(which.max(table(auto_data$num_doors))), num_doors),
    bore = ifelse(is.na(bore), median(bore, na.rm = TRUE), bore),
    stroke = ifelse(is.na(stroke), median(stroke, na.rm = TRUE), stroke),
    horsepower = ifelse(is.na(horsepower), median(horsepower, na.rm = TRUE), horsepower),
    peak_rpm = ifelse(is.na(peak_rpm), median(peak_rpm, na.rm = TRUE), peak_rpm),
    price = ifelse(is.na(price), median(price, na.rm = TRUE), price)
)

head(auto_data)
```

```
##   symboling normalized_losses        make fuel_type aspiration num_doors
## 1         3                NA alfa-romero       gas        std       two
## 2         3                NA alfa-romero       gas        std       two
## 3         1                NA alfa-romero       gas        std       two
## 4         2               164        audi       gas        std      four
## 5         2               164        audi       gas        std      four
## 6         2                NA        audi       gas        std       two
##    body_style drive_wheels engine_location wheel_base length width height
## 1 convertible          rwd           front       88.6  168.8  64.1   48.8
## 2 convertible          rwd           front       88.6  168.8  64.1   48.8
## 3   hatchback          rwd           front       94.5  171.2  65.5   52.4
## 4       sedan          fwd           front       99.8  176.6  66.2   54.3
## 5       sedan          4wd           front       99.4  176.6  66.4   54.3
## 6       sedan          fwd           front       99.8  177.3  66.3   53.1
##   curb_weight engine_type num_cylinders engine_size fuel_system bore stroke
## 1        2548        dohc          four         130        mpfi 3.47   2.68
## 2        2548        dohc          four         130        mpfi 3.47   2.68
## 3        2823        ohcv           six         152        mpfi 2.68   3.47
## 4        2337         ohc          four         109        mpfi 3.19   3.40
## 5        2824         ohc          five         136        mpfi 3.19   3.40
## 6        2507         ohc          five         136        mpfi 3.19   3.40
##   compression_ratio horsepower peak_rpm city_mpg highway_mpg price
## 1               9.0        111     5000       21          27 13495
## 2               9.0        111     5000       21          27 16500
## 3               9.0        154     5000       19          26 16500
## 4              10.0        102     5500       24          30 13950
## 5               8.0        115     5500       18          22 17450
## 6               8.5        110     5500       19          25 15250
```

```
colSums(is.na(auto_data))
```

```
##         symboling normalized_losses              make         fuel_type
##                 0                41                 0                 0
##        aspiration         num_doors        body_style      drive_wheels
##                 0                 0                 0                 0
##   engine_location        wheel_base            length             width
##                 0                 0                 0                 0
##            height       curb_weight       engine_type     num_cylinders
##                 0                 0                 0                 0
##       engine_size       fuel_system              bore            stroke
##                 0                 0                 0                 0
## compression_ratio        horsepower          peak_rpm          city_mpg
##                 0                 0                 0                 0
##       highway_mpg             price
##                 0                 0
```

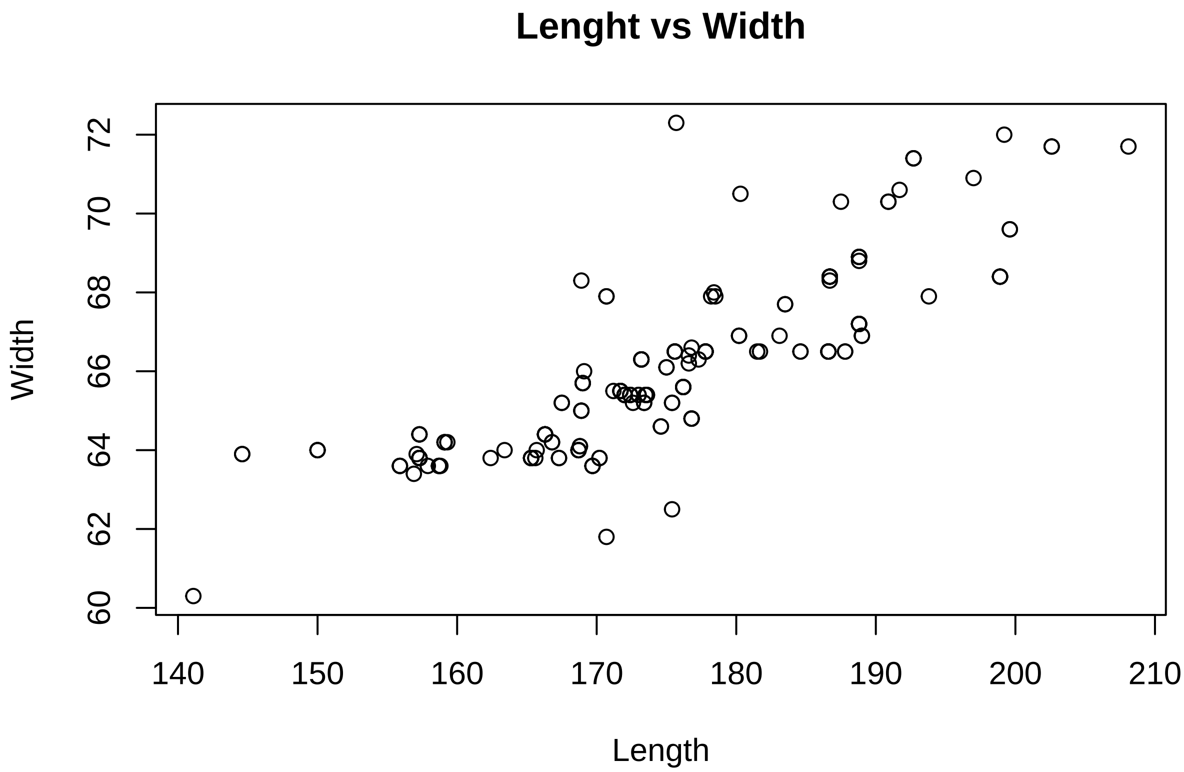
```
# for categorical value
auto_data <- auto_data %>%
  mutate(normalized_losses = replace_na(normalized_losses, mean(normalized_losses, na.rm = TRUE)))

auto_data$num_doors[is.na(auto_data$num_doors)] <- names(which.max(table(auto_data$num_doors)))

# Again check if there are null values.
colSums(is.na(auto_data))
```

```
##         symboling normalized_losses              make         fuel_type
##                 0                 0                 0                 0
##        aspiration         num_doors        body_style      drive_wheels
##                 0                 0                 0                 0
##   engine_location        wheel_base            length             width
##                 0                 0                 0                 0
##            height       curb_weight       engine_type     num_cylinders
##                 0                 0                 0                 0
##       engine_size       fuel_system              bore            stroke
##                 0                 0                 0                 0
## compression_ratio        horsepower          peak_rpm          city_mpg
##                 0                 0                 0                 0
##       highway_mpg             price
##                 0                 0
```

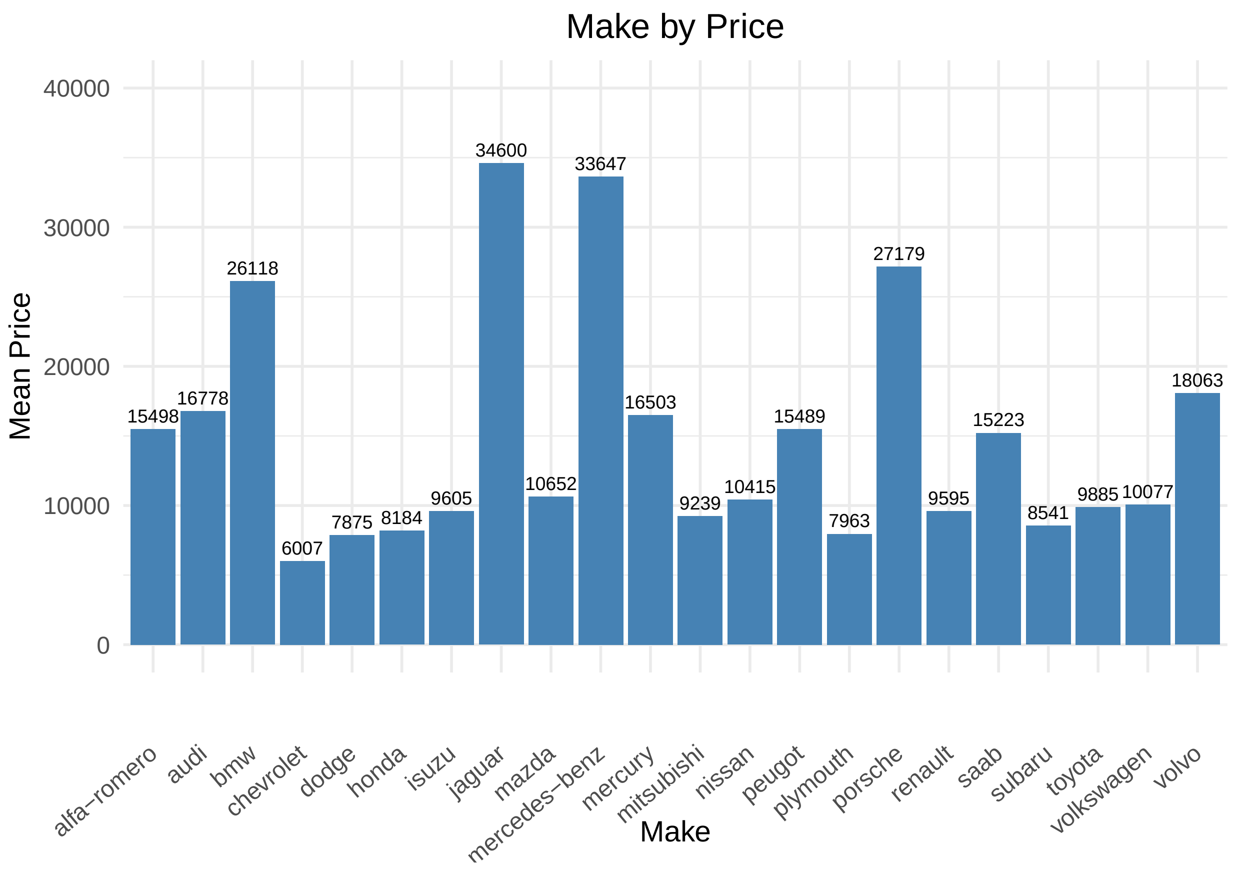
```r
# saving data in csv so that I can use the visualization from python later on
auto_data_output <- auto_data %>% select_if(is.numeric)
# Save the transformed data to a CSV file
write.csv(auto_data_output, "transformed_data.csv", row.names = FALSE)
```

Exloratory Data Analysis

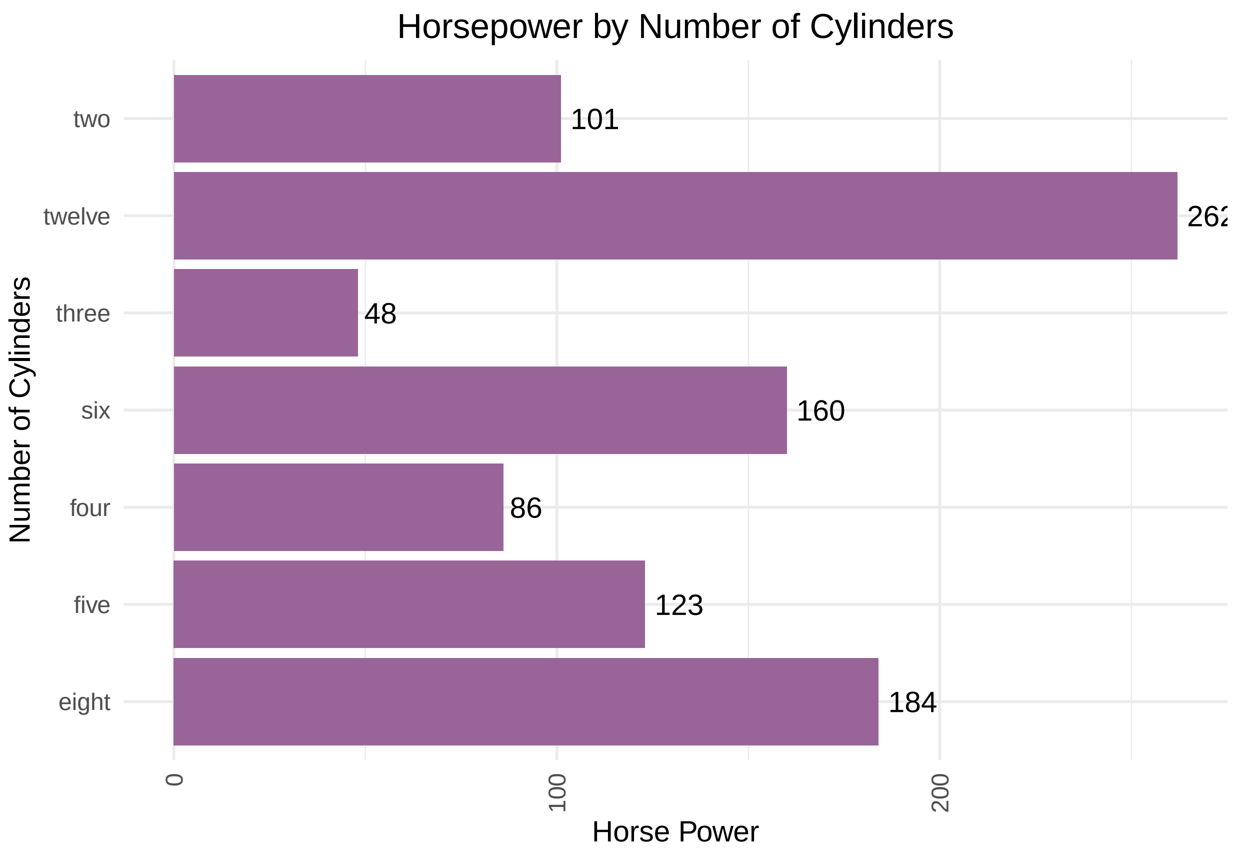
```r
# relationship between length and width of a automobile
plot(auto_data$length, auto_data$width, main='Lenght vs Width', xlab = 'Length', ylab = 'Width')
```



```r
# price by make
make_price <- as_tibble(aggregate(price ~ make, auto_data, mean))
make_price$price <- as.integer(make_price$price)

ggplot(make_price, aes(x = make, y = price)) +
  geom_bar(stat = "identity", fill = "steelblue") +
  geom_text(aes(label = price), vjust = -0.5, size = 2.5) +
  theme_minimal() +
  theme(
    axis.text.x = element_text(angle = 40, vjust = 0.5, hjust = 1),
    plot.title = element_text(hjust = 0.5))+
  ylim(0,40000)+
  labs(x = "Make", y = "Mean Price", title = "Make by Price")
```



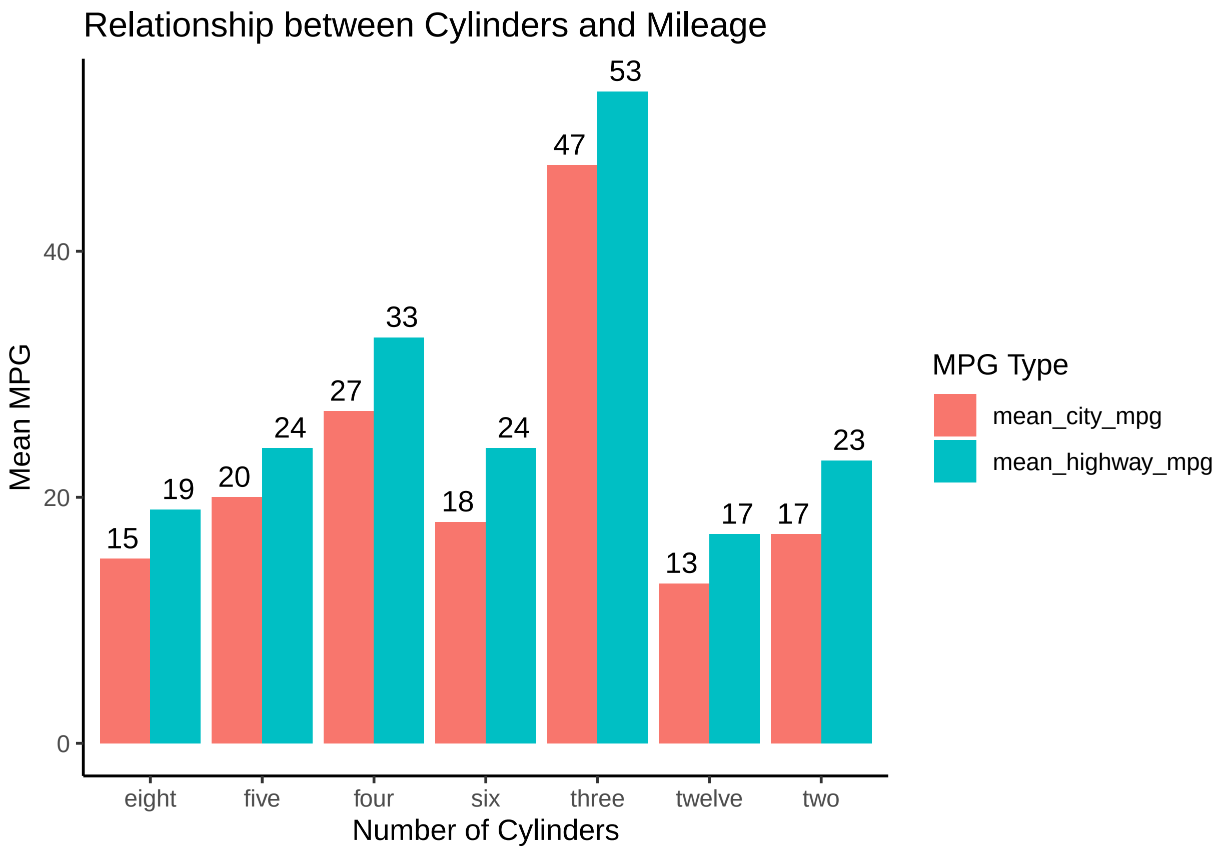
```
# relationship between horsepower by number of cylinders
hp_cyl<- as_tibble(aggregate(horsepower ~ num_cylinders, auto_data, median))
ggplot(hp_cyl, aes(x = horsepower, y = num_cylinders)) +
  geom_bar(stat = "identity", fill = "#996498") +
  geom_text(aes(label = horsepower), vjust = 0.5, hjust = -0.2) +
  theme_minimal() +
  theme(
    axis.text.x = element_text(angle = 90, vjust = 0.5, hjust = 1),
    plot.title = element_text(hjust = 0.5))+
  labs(x = "Horse Power", y = "Number of Cylinders", title = "Horsepower by Number of Cylinders")
```



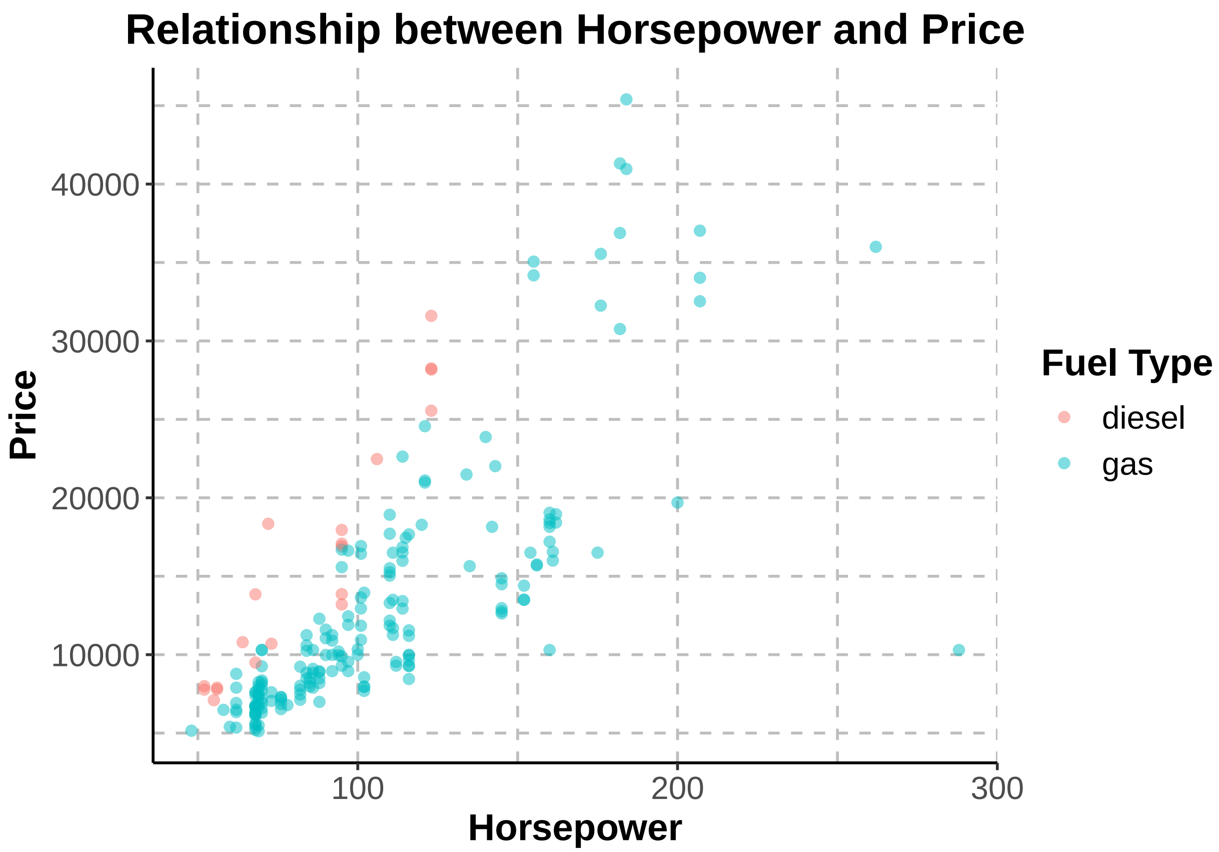
```
# cylinder vs mpg (city, highway)
mpg_summary <- auto_data %>%
  group_by(num_cylinders) %>%
  summarize(mean_city_mpg = round(mean(city_mpg)), mean_highway_mpg = round(mean(highway_mpg)))
mpg_summary
```

```
## # A tibble: 7 x 3
##   num_cylinders mean_city_mpg mean_highway_mpg
##   <chr>                 <dbl>            <dbl>
## 1 eight                    15               19
## 2 five                     20               24
## 3 four                     27               33
## 4 six                      18               24
## 5 three                    47               53
## 6 twelve                   13               17
## 7 two                      17               23
```

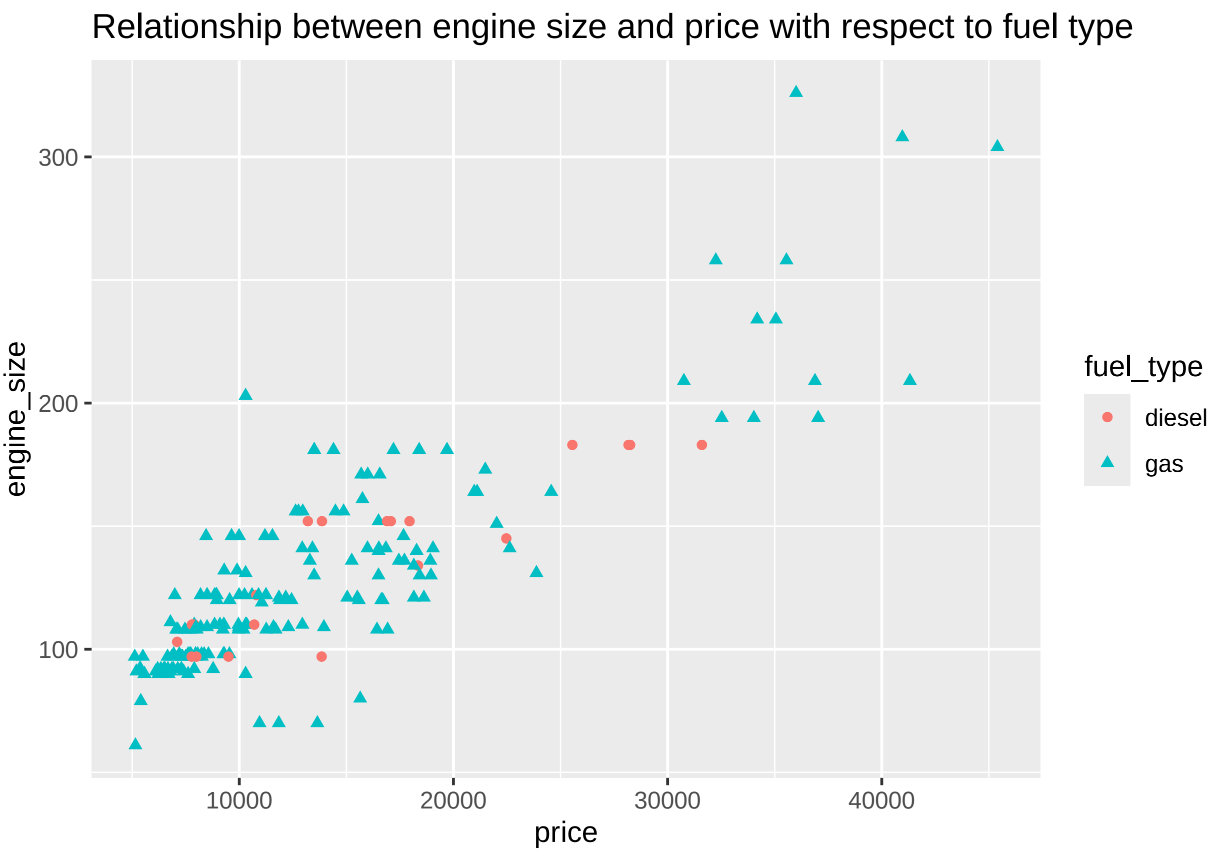
```
mpg_summary %>%
  pivot_longer(c(mean_city_mpg, mean_highway_mpg),
               names_to = "mpg_type",
               values_to = "mean_mpg") %>%
  ggplot(aes(x = num_cylinders, y = mean_mpg, fill = mpg_type)) +
  geom_col(position = "dodge") +
  geom_text(aes(label = round(mean_mpg, 1)), position = position_dodge(width = 1), vjust = -0.5) +
  labs(x = "Number of Cylinders", y = "Mean MPG", fill = "MPG Type") +
  theme_classic() +
  ggtitle("Relationship between Cylinders and Mileage")
```



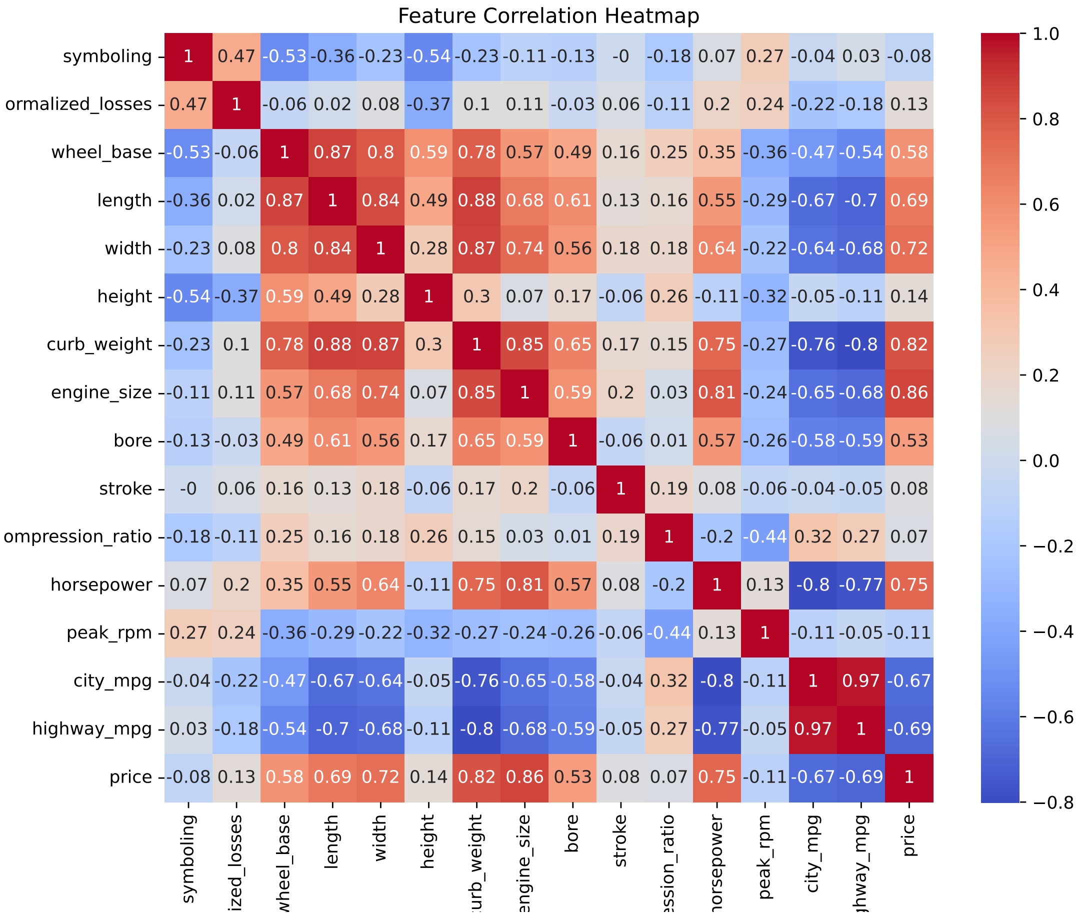
```
# relationship between horsepower and price
ggplot(auto_data, aes(x = horsepower, y = price, color = fuel_type)) +
  geom_point(alpha = 0.5) +
  labs(x = "Horsepower", y = "Price", color = "Fuel Type") +
  theme_classic() +
  theme(panel.grid.major = element_line(size = 0.5, linetype = "dashed", color = "gray"),
        panel.grid.minor = element_line(size = 0.5, linetype = "dashed", color = "gray"),
        axis.text = element_text(size = 12),
        axis.title = element_text(size = 14, face = "bold"),
        plot.title = element_text(size = 16, face = "bold", hjust = 0.5),
        legend.text = element_text(size = 12),
        legend.title = element_text(size = 14, face = "bold")) +
  ggtitle("Relationship between Horsepower and Price")
```



```
# relationship between engine size and price with respect to fuel type
ggplot(data = auto_data, aes(x = price, y = engine_size)) +
  geom_point(aes(color = fuel_type, shape = fuel_type)) +
  ggtitle('Relationship between engine size and price with respect to fuel type')
```



```
# creating heatmap since R doesn't have good visualization for heat map
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt


# # Load the transformed data from converted from R
transformed_data = pd.read_csv("transformed_data.csv")
corr_matrix = transformed_data.corr()
corr_matrix = round(corr_matrix,2)

# Plot the correlation
plt.figure(figsize=(10, 8))
sns.heatmap(corr_matrix, annot=True, cmap="coolwarm", square=True)
plt.title("Feature Correlation Heatmap")
```



```
# hypothesis testing

# hypothesis test: does high horsepower means higher price?
cor.test(auto_data$horsepower,auto_data$price)
```

```
##
##  Pearson's product-moment correlation
##
## data:  auto_data$horsepower and auto_data$price
## t = 16.152, df = 203, p-value < 2.2e-16
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  0.6830811 0.8043001
## sample estimates:
##       cor
## 0.7499191
```

```
# indicates the good correlation between horsepower and price

# hypothesis test: does lengthier car has higher price?
cor.test(auto_data$length,auto_data$price)
```

```
##
##  Pearson's product-moment correlation
##
## data:  auto_data$length and auto_data$price
## t = 13.454, df = 203, p-value < 2.2e-16
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  0.6066040 0.7527782
## sample estimates:
##       cor
## 0.6865674
```

```
# indicates the good correlation between length and price

# does engine size impact horse power
cor.test(auto_data$engine_size,auto_data$horsepower)
```

```
##
##  Pearson's product-moment correlation
##
## data:  auto_data$engine_size and auto_data$horsepower
## t = 19.695, df = 203, p-value < 2.2e-16
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  0.7572577 0.8525897
## sample estimates:
##      cor
## 0.810216
```

```
# indicates the strong correlation between engine_size and horsepower

# does engine size impact compression ratio
cor.test(auto_data$engine_size,auto_data$compression_ratio)
```

```
##
##  Pearson's product-moment correlation
##
## data:  auto_data$engine_size and auto_data$compression_ratio
## t = 0.41295, df = 203, p-value = 0.6801
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  -0.1084944  0.1653499
## sample estimates:
##        cor
## 0.02897136
```

```
# there is no statistically significant evidence that engine size impact compression ratio does accepti
```

Task 2

Attached is the sales dataset for retail store. Perform data munging on the given dataset.

```
library(readr)
library(reshape2)
library(tidyverse)
library(RColorBrewer)

# Load the data
df <- read_csv("Sales.csv")
```

Basic Data Exploration

```
# Print the shape of the data
dim(df)
```

```
## [1] 1000   17
```

```
head(df)
```

```
## # A tibble: 6 x 17
##   ‘Invoice ID‘ Branch City    ‘Customer type‘ Gender ‘Product line‘ ‘Unit price‘
##   <chr>        <chr>  <chr>   <chr>           <chr>  <chr>                 <dbl>
## 1 750-67-8428  A      Yangon  Member          Female Health and be~         74.7
## 2 226-31-3081  C      Naypyi~ Normal          Female Electronic ac~         15.3
## 3 631-41-3108  A      Yangon  Normal          Male   Home and life~         46.3
## 4 123-19-1176  A      Yangon  Member          Male   Health and be~         58.2
## 5 373-73-7910  A      Yangon  Normal          Male   Sports and tr~         86.3
## 6 699-14-3026  C      Naypyi~ Normal          Male   Electronic ac~         85.4
## # i 10 more variables: Quantity <dbl>, ‘Tax 5%‘ <dbl>, Total <dbl>, Date <chr>,
## #   Time <time>, Payment <chr>, cogs <dbl>, ‘gross margin percentage‘ <dbl>,
## #   ‘gross income‘ <dbl>, Rating <dbl>
```

```
# Check column info of the data
str(df)
```

```
## spc_tbl_ [1,000 x 17] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
##  $ Invoice ID             : chr [1:1000] "750-67-8428" "226-31-3081" "631-41-3108" "123-19-1176" ...
##  $ Branch                 : chr [1:1000] "A" "C" "A" "A" ...
##  $ City                   : chr [1:1000] "Yangon" "Naypyitaw" "Yangon" "Yangon" ...
##  $ Customer type          : chr [1:1000] "Member" "Normal" "Normal" "Member" ...
##  $ Gender                 : chr [1:1000] "Female" "Female" "Male" "Male" ...
##  $ Product line           : chr [1:1000] "Health and beauty" "Electronic accessories" "Home and lifes
##  $ Unit price             : num [1:1000] 74.7 15.3 46.3 58.2 86.3 ...
##  $ Quantity               : num [1:1000] 7 5 7 8 7 7 6 10 2 3 ...
##  $ Tax 5%                 : num [1:1000] 26.14 3.82 16.22 23.29 30.21 ...
##  $ Total                  : num [1:1000] 549 80.2 340.5 489 634.4 ...
##  $ Date                   : chr [1:1000] "1/5/2019" "3/8/2019" "3/3/2019" "1/27/2019" ...
##  $ Time                   : 'hms' num [1:1000] 13:08:00 10:29:00 13:23:00 20:33:00 ...
##   ..- attr(*, "units")= chr "secs"
##  $ Payment                : chr [1:1000] "Ewallet" NA "Credit card" "Ewallet" ...
##  $ cogs                   : num [1:1000] 522.8 76.4 324.3 465.8 604.2 ...
##  $ gross margin percentage: num [1:1000] 4.76 4.76 4.76 4.76 4.76 ...
##  $ gross income           : num [1:1000] 26.14 3.82 16.22 23.29 30.21 ...
##  $ Rating                 : num [1:1000] 9.1 9.6 7.4 8.4 5.3 4.1 5.8 8 7.2 5.9 ...
##  - attr(*, "spec")=
##   .. cols(
##   ..   ‘Invoice ID‘ = col_character(),
##   ..   Branch = col_character(),
##   ..   City = col_character(),
##   ..   ‘Customer type‘ = col_character(),
##   ..   Gender = col_character(),
##   ..   ‘Product line‘ = col_character(),
##   ..   ‘Unit price‘ = col_double(),
##   ..   Quantity = col_double(),
##   ..   ‘Tax 5%‘ = col_double(),
##   ..   Total = col_double(),
##   ..   Date = col_character(),
##   ..   Time = col_time(format = ""),
##   ..   Payment = col_character(),
##   ..   cogs = col_double(),
##   ..   ‘gross margin percentage‘ = col_double(),
##   ..   ‘gross income‘ = col_double(),
##   ..   Rating = col_double()
##   .. )
##  - attr(*, "problems")=<externalptr>
```

```
spec(df)
```

```
## cols(
##   ‘Invoice ID‘ = col_character(),
##   Branch = col_character(),
##   City = col_character(),
##   ‘Customer type‘ = col_character(),
##   Gender = col_character(),
##   ‘Product line‘ = col_character(),
##   ‘Unit price‘ = col_double(),
##   Quantity = col_double(),
##   ‘Tax 5%‘ = col_double(),
##   Total = col_double(),
##   Date = col_character(),
##   Time = col_time(format = ""),
##   Payment = col_character(),
##   cogs = col_double(),
##   ‘gross margin percentage‘ = col_double(),
##   ‘gross income‘ = col_double(),
##   Rating = col_double()
## )
```

Data Munging

```
# Check for missing values
colSums(is.na(df))
```

```
##              Invoice ID                  Branch                    City
##                      10                       0                       0
##           Customer type                  Gender            Product line
##                       0                       0                       0
##              Unit price                Quantity                  Tax 5%
##                       2                      16                      14
##                   Total                    Date                    Time
##                      14                       0                       1
##                 Payment                    cogs gross margin percentage
##                       3                       0                      29
##            gross income                  Rating
##                       0                       0
```

Imputing the missing values

```
# imputing missing values for invoice
impute_invoice <- function() {
  x <- sample(c(100:999), 10, replace = T)
  y <- sample(c(10:99), 10, replace = T)
  z <- sample(c(100:999), 10, replace = T)
  a <- paste0(x, "-", y, "-", z)
  return (a)
}

df[is.na(df$'Invoice ID'),1] <- impute_invoice()
head(df)
```

```
## # A tibble: 6 x 17
##   ‘Invoice ID‘ Branch City    ‘Customer type‘ Gender ‘Product line‘ ‘Unit price‘
##   <chr>        <chr>  <chr>   <chr>           <chr>  <chr>                 <dbl>
## 1 750-67-8428  A      Yangon  Member          Female Health and be~         74.7
## 2 226-31-3081  C      Naypyi~ Normal          Female Electronic ac~         15.3
## 3 631-41-3108  A      Yangon  Normal          Male   Home and life~         46.3
## 4 123-19-1176  A      Yangon  Member          Male   Health and be~         58.2
## 5 373-73-7910  A      Yangon  Normal          Male   Sports and tr~         86.3
## 6 699-14-3026  C      Naypyi~ Normal          Male   Electronic ac~         85.4
## # i 10 more variables: Quantity <dbl>, ‘Tax 5%‘ <dbl>, Total <dbl>, Date <chr>,
## #   Time <time>, Payment <chr>, cogs <dbl>, ‘gross margin percentage‘ <dbl>,
## #   ‘gross income‘ <dbl>, Rating <dbl>
```

```
colSums(is.na(df))
```

```
##              Invoice ID                  Branch                    City
##                       0                       0                       0
##           Customer type                  Gender            Product line
##                       0                       0                       0
##              Unit price                Quantity                  Tax 5%
##                       2                      16                      14
##                   Total                    Date                    Time
##                      14                       0                       1
##                 Payment                    cogs gross margin percentage
##                       3                       0                      29
##            gross income                  Rating
##                       0                       0
```

```
# missing value for Invoice is filled.
# now missing values for unit price
df[is.na(df$'Unit price'),c('Unit price','Total','Tax 5%','Quantity')]
```

```
## # A tibble: 2 x 4
##   ‘Unit price‘ Total ‘Tax 5%‘ Quantity
##          <dbl> <dbl>    <dbl>    <dbl>
## 1           NA  772.    36.8        10
## 2           NA  125.     5.96        5
```

```
df$'Unit price' <- ifelse(is.na(df$'Unit price'), (df$Total - df$'Tax 5%') / df$Quantity, df$'Unit pric
colSums(is.na(df))
```

```
##              Invoice ID                  Branch                    City
##                       0                       0                       0
##           Customer type                  Gender            Product line
##                       0                       0                       0
##              Unit price                Quantity                  Tax 5%
##                       0                      16                      14
##                   Total                    Date                    Time
##                      14                       0                       1
##                 Payment                    cogs gross margin percentage
##                       3                       0                      29
##            gross income                  Rating
##                       0                       0
```

```
# missing value for unit price is filled.

# missing values for Quantity
head(df[is.na(df$'Quantity'),])
```

```
## # A tibble: 6 x 17
##   ‘Invoice ID‘ Branch City    ‘Customer type‘ Gender ‘Product line‘ ‘Unit price‘
##   <chr>        <chr>  <chr>   <chr>           <chr>  <chr>                 <dbl>
## 1 418-05-0656  B      Mandal~ Normal          Female Fashion acces~         25.6
## 2 804-38-3935  A      Yangon  Member          Male   Electronic ac~         93.8
## 3 866-70-2814  B      Mandal~ Normal          Female Electronic ac~         52.8
## 4 101-81-4070  C      Naypyi~ Member          Female Health and be~         62.8
## 5 851-98-3555  B      Mandal~ Normal          Female Health and be~         82.9
## 6 186-71-5196  A      Yangon  Member          Female Food and beve~         79.5
## # i 10 more variables: Quantity <dbl>, ‘Tax 5%‘ <dbl>, Total <dbl>, Date <chr>,
## #   Time <time>, Payment <chr>, cogs <dbl>, ‘gross margin percentage‘ <dbl>,
## #   ‘gross income‘ <dbl>, Rating <dbl>
```

```
df$'Quantity' <- ifelse(is.na(df$'Quantity'), (df$Total - df$'Tax 5%') %/% df$'Unit price', df$'Quantit
colSums(is.na(df))
```

```
##              Invoice ID                  Branch                    City
##                       0                       0                       0
##           Customer type                  Gender            Product line
##                       0                       0                       0
##              Unit price                Quantity                  Tax 5%
##                       0                       0                      14
##                   Total                    Date                    Time
##                      14                       0                       1
##                 Payment                    cogs gross margin percentage
##                       3                       0                      29
##            gross income                  Rating
##                       0                       0
```

```
# missing value for Quantity is filled.

# missing values for Total
df[is.na(df$'Total'),c('Unit price','Total','Tax 5%','Quantity')]
```

```
## # A tibble: 14 x 4
##    `Unit price` Total `Tax 5%` Quantity
##           <dbl> <dbl>    <dbl>    <dbl>
##  1         88.4    NA    22.1         5
##  2         44.6    NA    11.1         5
##  3         23.1    NA    10.4         9
##  4         66.1    NA    13.2         4
##  5         80.0    NA    20.0         5
##  6         74.3    NA     3.71        1
##  7         24.8    NA     6.19        5
##  8         24.9    NA    11.2         9
##  9         54.4    NA     2.72        1
## 10         48.5    NA    17.0         7
## 11         99.4    NA     9.94        2
## 12         25.4    NA    10.2         8
## 13         21.6    NA     1.08        1
## 14         82.6    NA    41.3        10
```

```
df$'Total' <- ifelse(is.na(df$'Total'), (df$'Unit price' * df$'Quantity') + df$'Tax 5%', df$'Total')
colSums(is.na(df))
```

```
##              Invoice ID                  Branch                    City
##                       0                       0                       0
##           Customer type                  Gender            Product line
##                       0                       0                       0
##              Unit price                Quantity                  Tax 5%
##                       0                       0                      14
##                   Total                    Date                    Time
##                       0                       0                       1
##                 Payment                    cogs gross margin percentage
##                       3                       0                      29
##            gross income                  Rating
##                       0                       0
```

```
# missing value for Total is filled.

# missing values for Tax 5%
df[is.na(df$'Tax 5%'),c('Unit price','Total','Tax 5%','Quantity')]
```

```
## # A tibble: 14 x 4
##    ‘Unit price‘ Total ‘Tax 5%‘ Quantity
##           <dbl> <dbl>    <dbl>    <dbl>
##  1         93.7 590.        NA        6
##  2         68.9 507.        NA        7
##  3         72.6 457.        NA        6
##  4         89.5 940.        NA       10
##  5         62.1 652.        NA       10
##  6         48.5 153.        NA        3
##  7         87.9 923.        NA       10
##  8         57.1 420.        NA        7
##  9         72.5 609         NA        8
## 10         43.2 363.        NA        8
## 11         15.3  16.1       NA        1
## 12         23.5  49.3       NA        2
## 13         75.8  79.6       NA        1
## 14         73.0 767.        NA       10
```

```
df$'Tax 5%' <- ifelse(is.na(df$'Tax 5%'), (df$'Total' - (df$'Unit price') * df$'Quantity'), df$'Total')
colSums(is.na(df))
```

```
##              Invoice ID                  Branch                    City
##                       0                       0                       0
##           Customer type                  Gender            Product line
##                       0                       0                       0
##              Unit price                Quantity                  Tax 5%
##                       0                       0                       0
##                   Total                    Date                    Time
##                       0                       0                       1
##                 Payment                    cogs gross margin percentage
##                       3                       0                      29
##            gross income                  Rating
##                       0                       0
```

```
# missing value for Tax 5% is filled.

# missing values for Time
df[is.na(df$Time),c('Date', 'Time')]
```

```
## # A tibble: 1 x 2
##   Date     Time
##   <chr>    <time>
## 1 2/9/2019    NA
```

```
random_time <- as.character(sample(df$Time,1))
df[is.na(df$Time),'Time'] <- random_time
df$Time <- gsub(":00", "",df$Time)

colSums(is.na(df))
```

```
##              Invoice ID                  Branch                    City
##                       0                       0                       0
##           Customer type                  Gender            Product line 
##                       0                       0                       0 
##              Unit price                Quantity                  Tax 5% 
##                       0                       0                       0 
##                   Total                    Date                    Time 
##                       0                       0                       0 
##                 Payment                    cogs gross margin percentage 
##                       3                       0                      29 
##            gross income                  Rating 
##                       0                       0
```

```
# missing value for Time is filled.

# missing values for Payment
df[is.na(df$Payment),]
```

```
## # A tibble: 3 x 17
##   ‘Invoice ID‘ Branch City    ‘Customer type‘ Gender ‘Product line‘ ‘Unit price‘
##   <chr>        <chr>  <chr>   <chr>           <chr>  <chr>                 <dbl>
## 1 226-31-3081  C      Naypyi~ Normal          Female Electronic ac~         15.3
## 2 145-94-9061  B      Mandal~ Normal          Female Food and beve~         88.4
## 3 841-35-6630  C      Naypyi~ Normal          Female Electronic ac~         75.9
## # i 10 more variables: Quantity <dbl>, ‘Tax 5%‘ <dbl>, Total <dbl>, Date <chr>,
## #   Time <chr>, Payment <chr>, cogs <dbl>, ‘gross margin percentage‘ <dbl>,
## #   ‘gross income‘ <dbl>, Rating <dbl>
```

```
unique(df$Payment)
```

```
## [1] "Ewallet"     NA            "Credit card" "Cash"
```

```
most_payment <-  names(which.max(table(df$Payment)))
df[is.na(df$Payment),'Payment'] <- most_payment

colSums(is.na(df))
```

```
##              Invoice ID                  Branch                    City 
##                       0                       0                       0 
##           Customer type                  Gender            Product line 
##                       0                       0                       0 
##              Unit price                Quantity                  Tax 5% 
##                       0                       0                       0 
##                   Total                    Date                    Time 
##                       0                       0                       0 
##                 Payment                    cogs gross margin percentage 
##                       0                       0                      29 
##            gross income                  Rating 
##                       0                       0
```

```
# missing value for Tax 5% is Payment

# missing values for gross margin percentage

df[is.na(df$'gross margin percentage'),
   c('Total', 'gross income', 'gross margin percentage')]
```

```
## # A tibble: 29 x 3
##    Total ‘gross income‘ ‘gross margin percentage‘
##    <dbl>          <dbl>                     <dbl>
##  1  23.5           1.12                        NA
##  2 177.            8.45                        NA
##  3 102.            4.84                        NA
##  4 349.           16.6                         NA
##  5  85.5           4.07                        NA
##  6 336.           16.0                         NA
##  7 217.           10.3                         NA
##  8 175.            8.33                        NA
##  9 335.           16.0                         NA
## 10 329.           15.7                         NA
## # i 19 more rows
```

```
df[,c('Total', 'gross income', 'gross margin percentage')]
```

```
## # A tibble: 1,000 x 3
##    Total ‘gross income‘ ‘gross margin percentage‘
##    <dbl>          <dbl>                     <dbl>
##  1 549.           26.1                       4.76
##  2  80.2           3.82                      4.76
##  3 341.           16.2                       4.76
##  4 489.           23.3                       4.76
##  5 634.           30.2                       4.76
##  6 628.           29.9                       4.76
##  7 434.           20.7                       4.76
##  8 772.           36.8                       4.76
##  9  76.1           3.63                      4.76
## 10 173.            8.23                      4.76
## # i 990 more rows
```

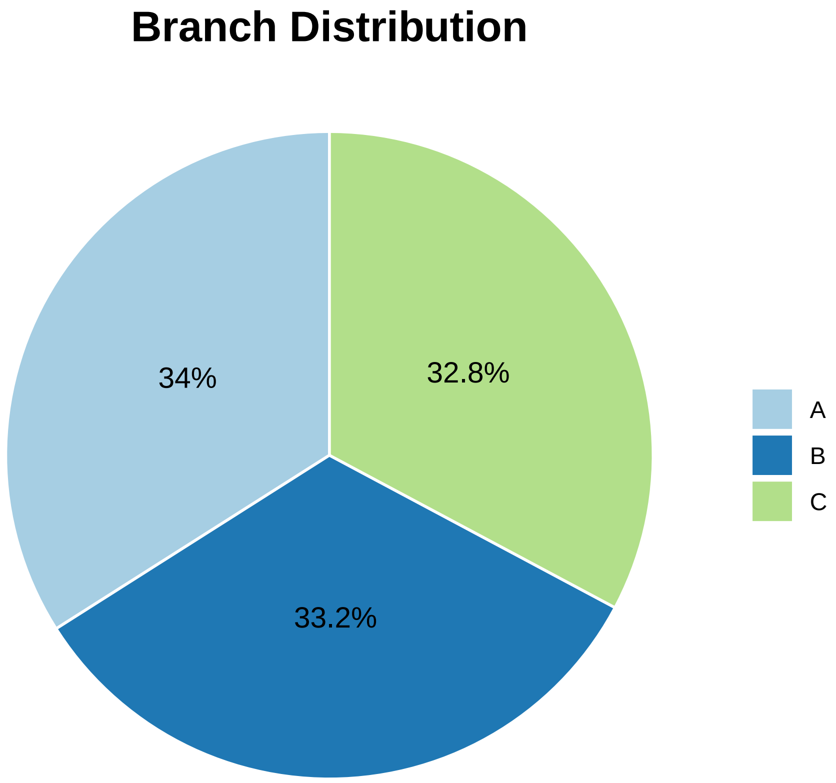
```
df[is.na(df$'gross margin percentage'),'gross margin percentage'] <- unique(df$'gross margin percentage

colSums(is.na(df))
```

```
##              Invoice ID                  Branch                    City
##                       0                       0                       0
##           Customer type                  Gender            Product line
##                       0                       0                       0
##              Unit price                Quantity                  Tax 5%
##                       0                       0                       0
##                   Total                    Date                    Time
##                       0                       0                       0
##                 Payment                    cogs gross margin percentage
##                       0                       0                       0
##            gross income                  Rating
##                       0                       0
```

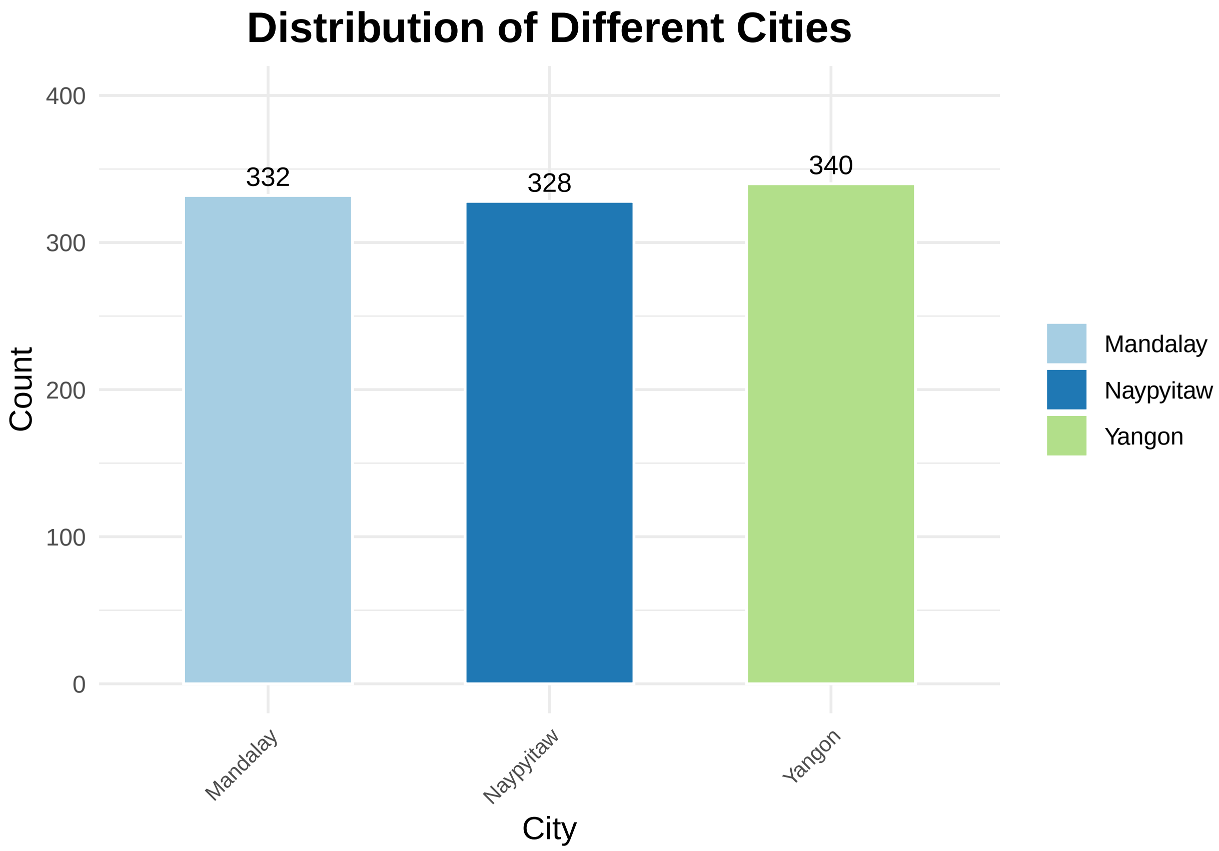
```
# missing value for gross margin percentage.

# Convert the date column to datetime format
df$Date <- as.Date(mdy(df$Date))
```

Basic EDA on Sales Dataset

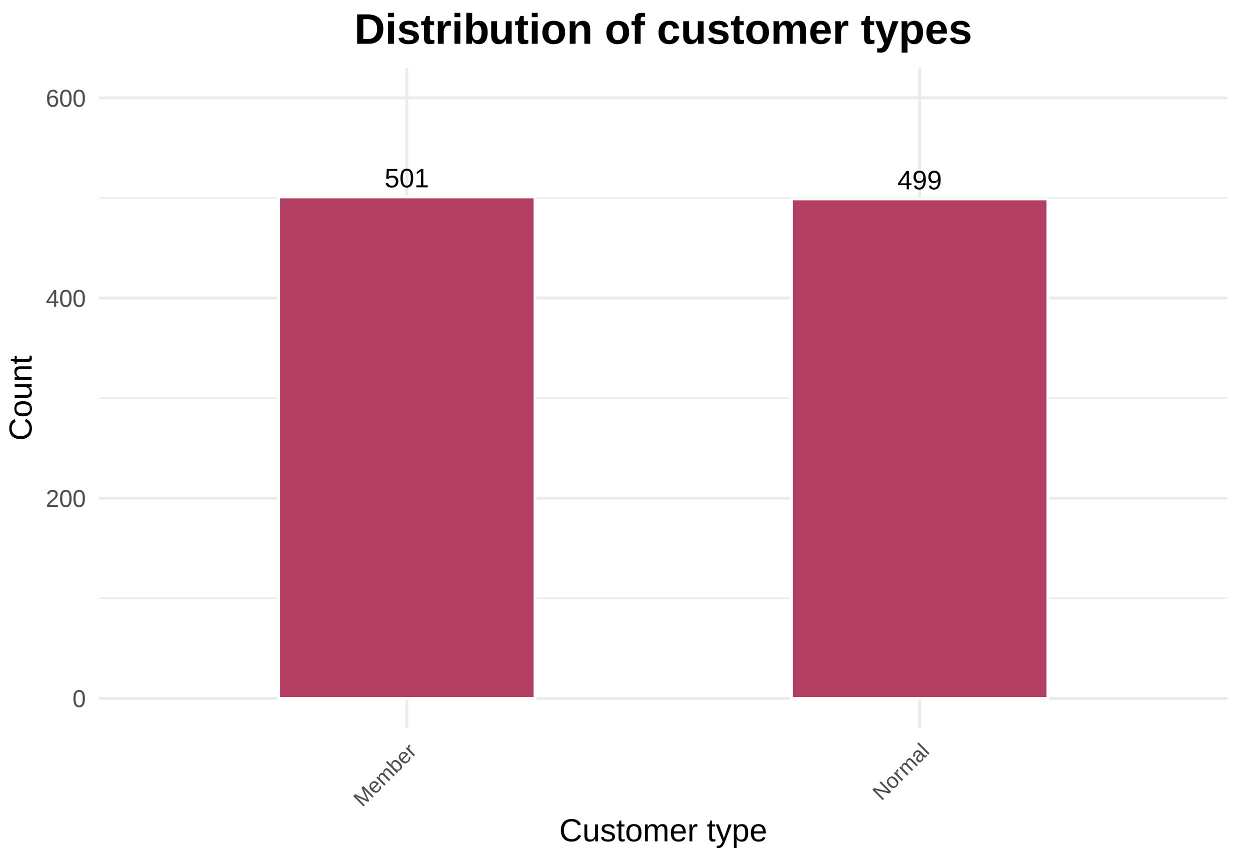
```
# Plot the distribution of different branches
branch_counts <- table(df$Branch)

colors <- brewer.pal(length(branch_counts), "Paired")
explode <- rep(0, length(branch_counts))
explode[which.max(branch_counts)] <- 0.1

ggplot(data.frame(Branch = names(branch_counts), Count = as.numeric(branch_counts)),
       aes(x = "", y = Count, fill = Branch)) +
  geom_bar(stat = "identity", width = 1, color = "white") +
  coord_polar("y", start = 0) +
  geom_text(aes(label = paste0(round(Count / sum(branch_counts) * 100, 1), "%")),
            position = position_stack(vjust = 0.5)) +
  scale_fill_manual(values = colors) +
  theme_void() +
  theme(legend.position = "right",
        legend.title = element_blank(),
        plot.title = element_text(hjust = 0.5, size = 16, face = "bold"),
        plot.subtitle = element_text(hjust = 0.5, size = 12)) +
  labs(title = "Branch Distribution")
```



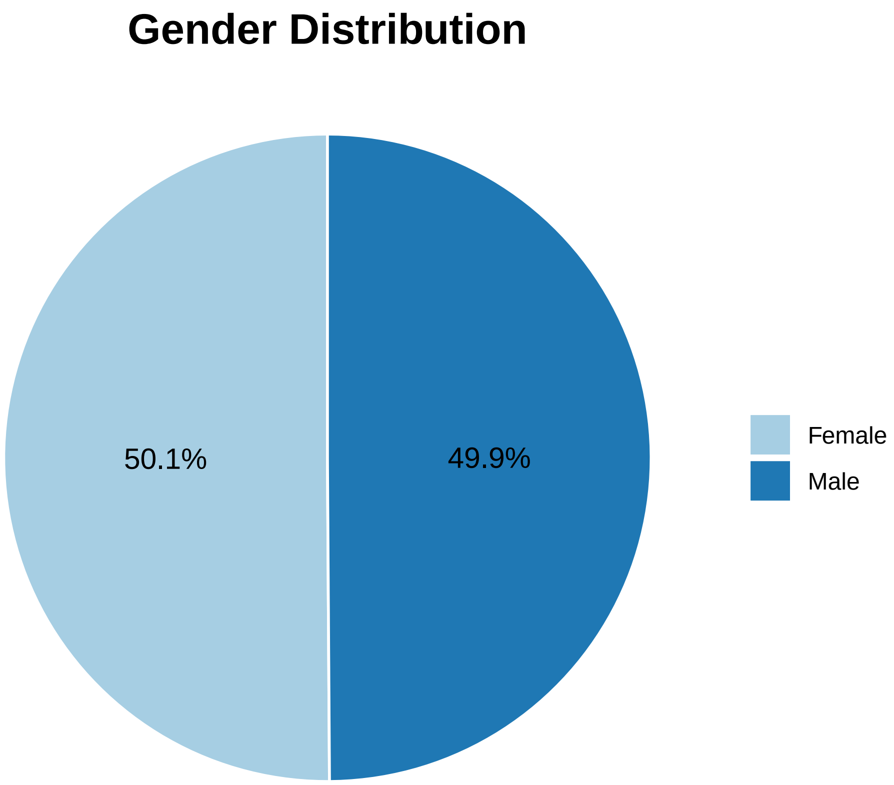
```
# Plot the distribution of different cities
ggplot(df, aes(x = City, fill = City)) +
  geom_bar(width = 0.6, color = "white") +
  scale_fill_manual(values = brewer.pal(length(unique(df$City)), "Paired")) +
  labs(title = "Distribution of Different Cities",
       x = "City",
       y = "Count") +
  geom_text(stat = "count", aes(label = after_stat(count)), vjust = -0.5, size = 3.5) +
  ylim(0, 400) +
  theme_minimal() +
  theme(plot.title = element_text(size = 16, face = "bold", hjust = 0.5),
        axis.text.x = element_text(angle = 45, hjust = 1, size = 8),
        axis.title = element_text(size = 12),
        legend.position = "right",
        legend.title = element_blank())
```



```
# Plot the distribution of customer types
ggplot(df, aes(x = df$'Customer type')) +
  geom_bar(width = 0.5, color = "white", fill='#B33F62') +
  labs(title = "Distribution of customer types", x = "Customer type", y = "Count") +
  geom_text(stat = "count", aes(label = comma(..count..)), vjust = -0.5, size = 3.5) +
  ylim(0,600) +
  theme_minimal() +
  theme(plot.title = element_text(size = 16, face = "bold", hjust = 0.5),
        axis.text.x = element_text(angle = 45, hjust = 1, size = 8),
        axis.title = element_text(size = 12),
        legend.position = "right",
        legend.title = element_blank())
```



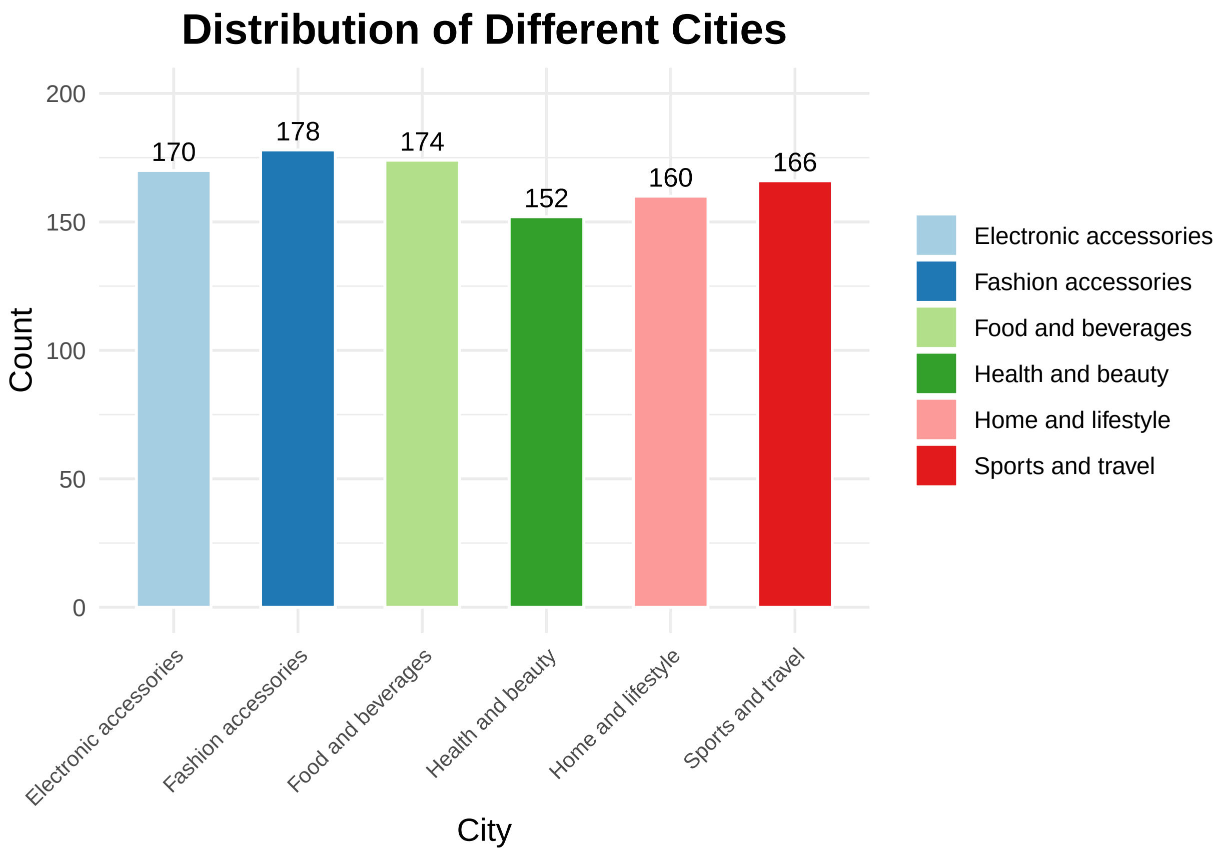
```
# Plot the distribution of genders
ggplot(data.frame(Gender = names(table(df$Gender)), Count = as.numeric(table(df$Gender))), aes(x = "",
  geom_bar(stat = "identity", width = 1, color = "white") +
  coord_polar("y", start = 0) +
  geom_text(aes(label = paste0(round(Count / sum(branch_counts) * 100, 1), "%")),
            position = position_stack(vjust = 0.5)) +
  scale_fill_manual(values = colors) +
  theme_void() +
  theme(legend.position = "right",
        legend.title = element_blank(),
        plot.title = element_text(hjust = 0.5, size = 16, face = "bold"),
        plot.subtitle = element_text(hjust = 0.5, size = 12)) +
  labs(title = "Gender Distribution")
```



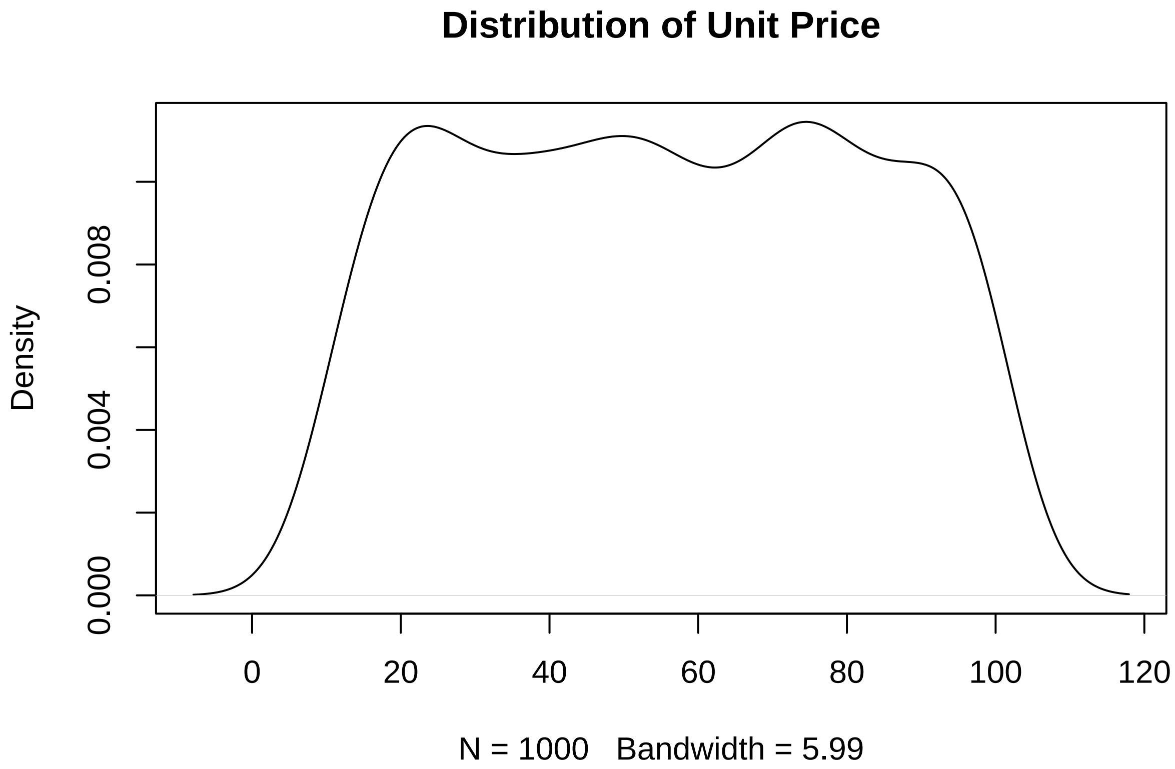
```
# Product Line
table(df$'Product line')
```

```
##
## Electronic accessories    Fashion accessories     Food and beverages
##                    170                    178                    174
##      Health and beauty     Home and lifestyle      Sports and travel
##                    152                    160                    166
```

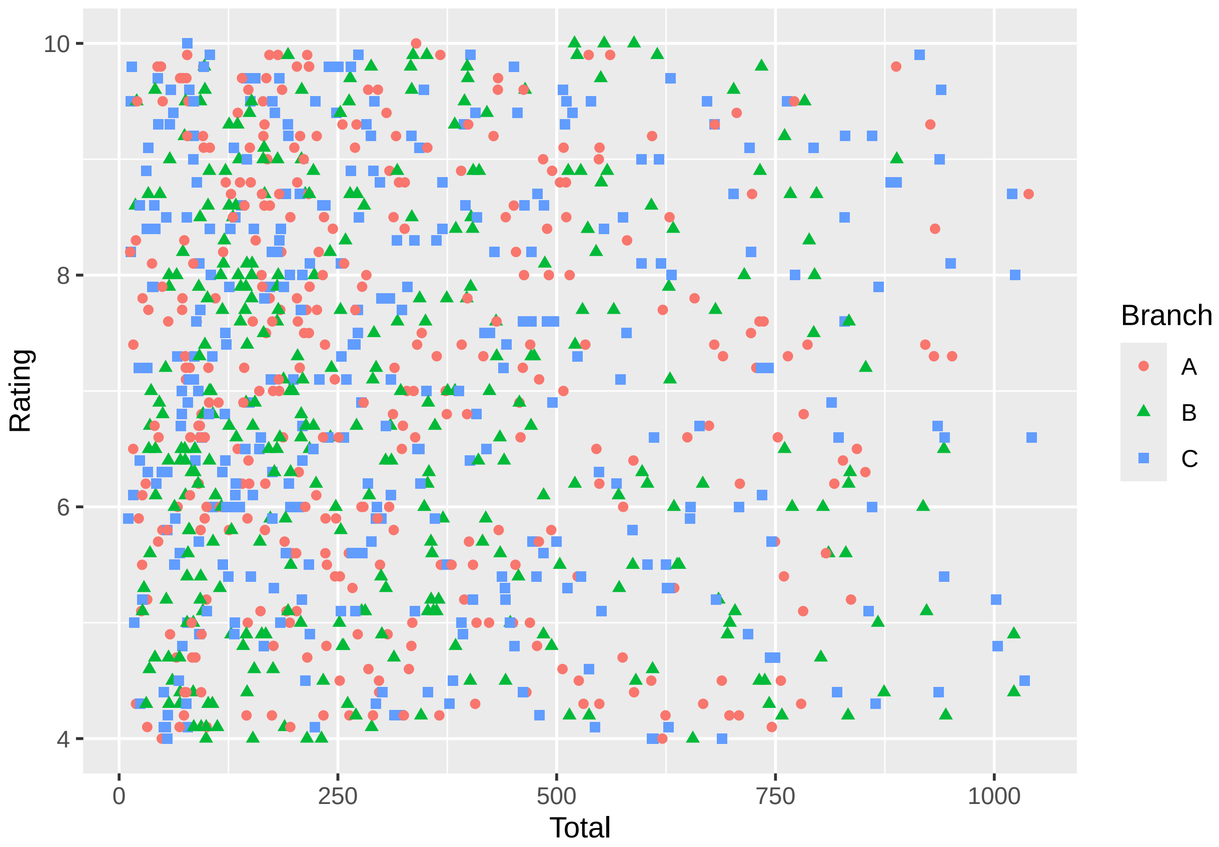
```
ggplot(df, aes(x = df$'Product line', fill = df$'Product line')) +
  geom_bar(width = 0.6, color = "white") +
  scale_fill_manual(values = brewer.pal(length(unique(df$'Product line')), "Paired")) +
  labs(title = "Distribution of Different Cities",
       x = "City",
       y = "Count") +
  geom_text(stat = "count", aes(label = after_stat(count)), vjust = -0.5, size = 3.5) +
  ylim(0, 200) +
  theme_minimal() +
  theme(plot.title = element_text(size = 16, face = "bold", hjust = 0.5),
        axis.text.x = element_text(angle = 45, hjust = 1, size = 8),
        axis.title = element_text(size = 12),
        legend.position = "right",
        legend.title = element_blank())
```



```
# Plot the distribution of unit prices
plot(density(df$'Unit price',na.rm = T), main = 'Distribution of Unit Price')
```



```
# plot scatterplot between total and rating with respect to Branch
ggplot(data=df, aes(x= Total, y= Rating))+
  geom_point(aes(color=Branch, shape=Branch))
```

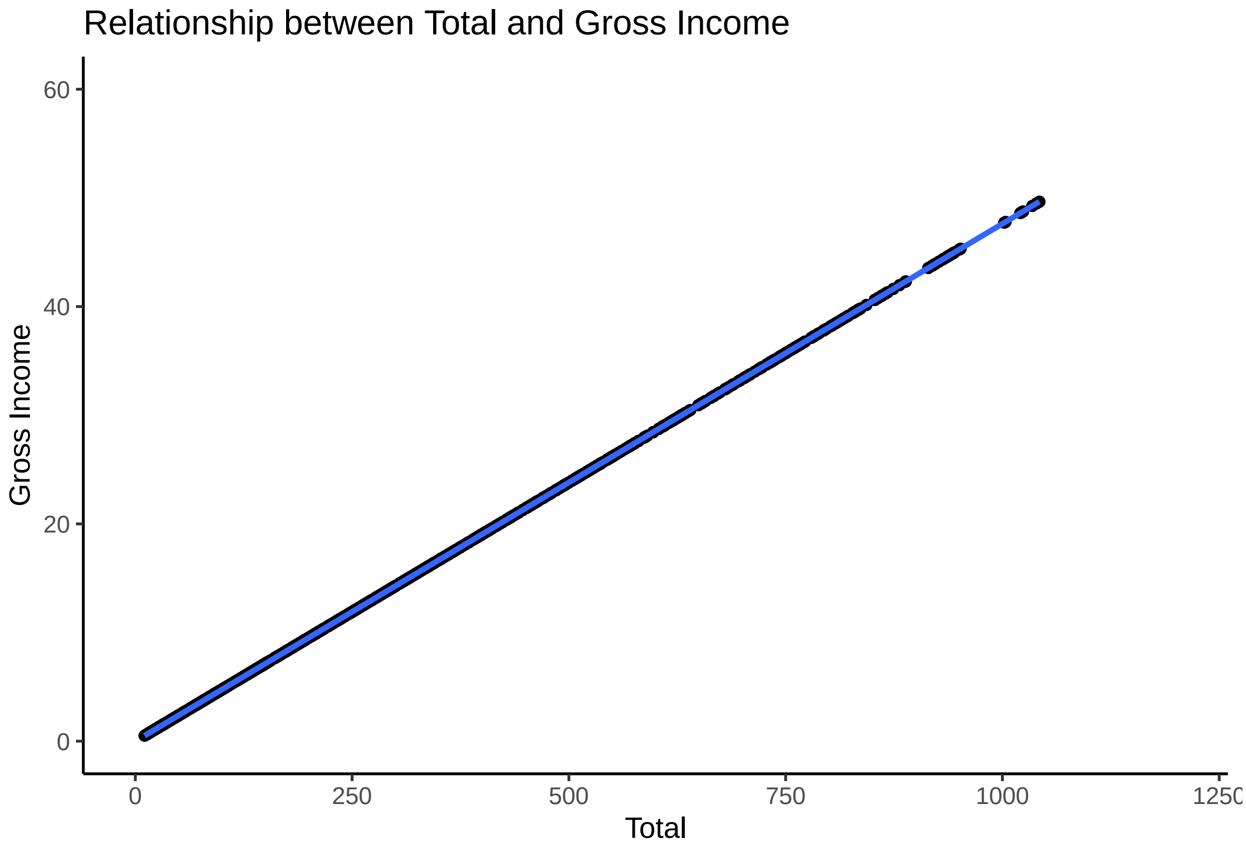


```
# Hypothesis testing: Does gross income increase when total sales increase?
cor.test(df$Total, df$'gross income')
```

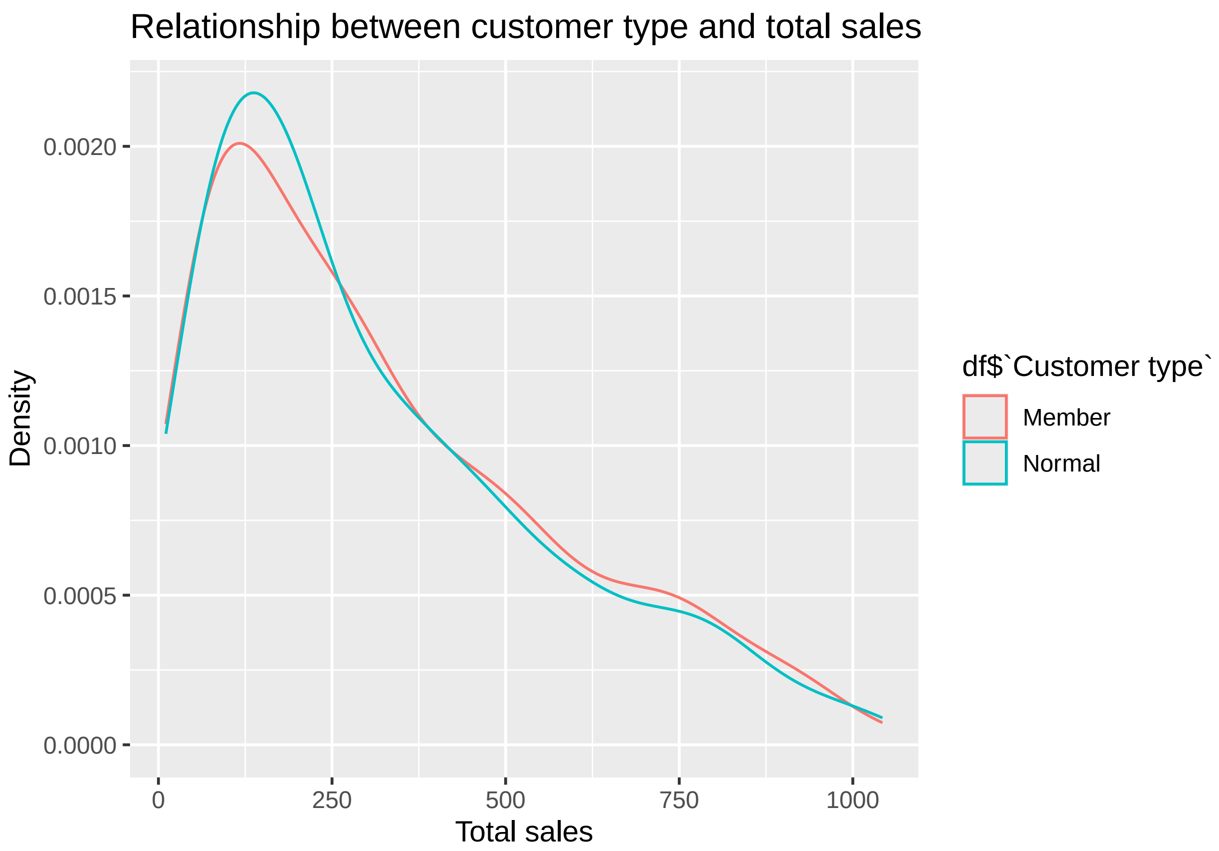
```
##
##  Pearson's product-moment correlation
##
## data:  df$Total and df$"gross income"
## t = Inf, df = 998, p-value < 2.2e-16
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  1 1
## sample estimates:
## cor
##   1
```

```
# it is indeed true, and is statistically significant
```

```
# Plot the relationship between total sales and gross income
ggplot(df, aes(x = Total, y = df$`gross income`)) +
  geom_point() +
  geom_smooth(method = "lm", se = FALSE) +
  labs(
    title = "Relationship between Total and Gross Income",
    x = "Total",
    y = "Gross Income"
  ) +
  theme_classic() +
  xlim(0,1200) + ylim(0,60)
```



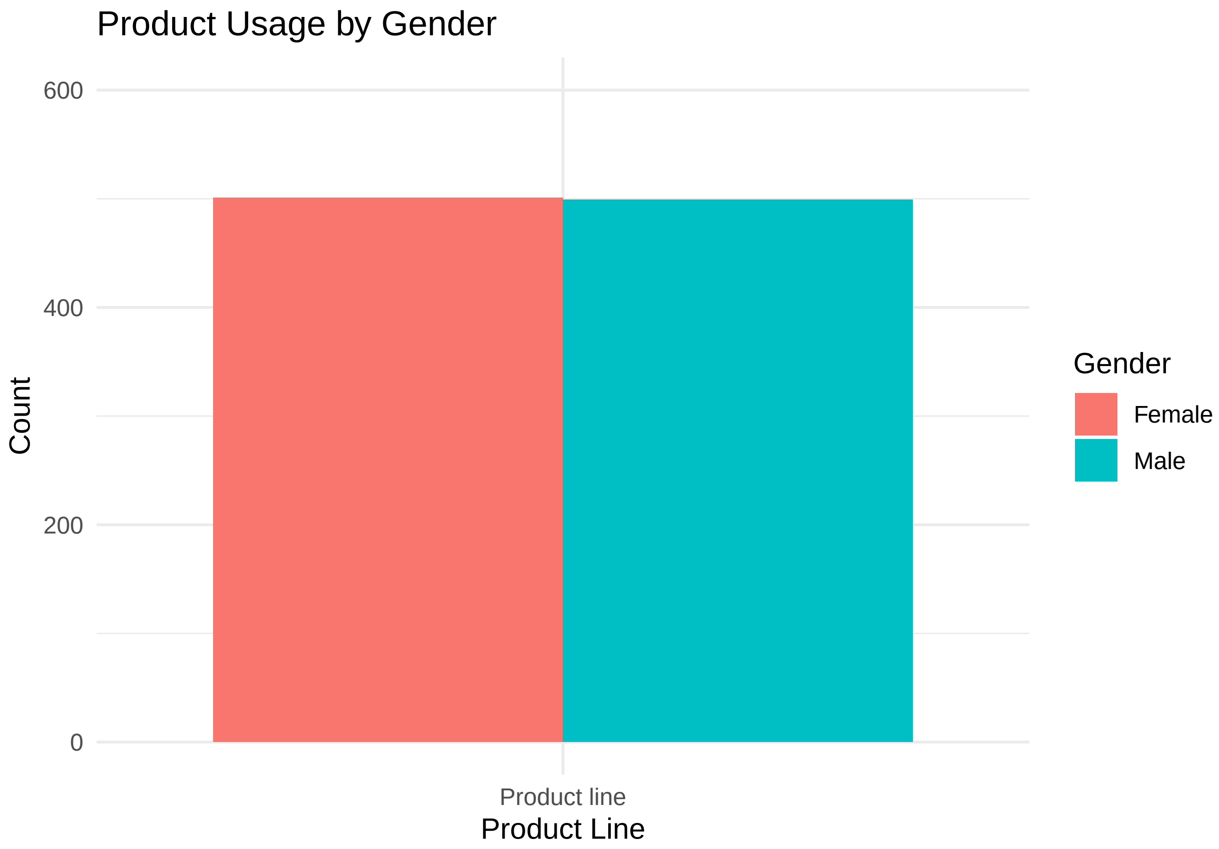
```
# Plot the relationship between customer type and total sales
ggplot(df, aes(x = Total, color = df$`Customer type`)) +
  geom_density() +
  labs(title = "Relationship between customer type and total sales", x = "Total sales", y = "Density")
```



```
# Which branch sales has the highest ratings?
df %>%
  group_by(Branch) %>%
  summarise(mean_rating = mean(Rating)) %>%
  arrange(desc(mean_rating))
```

```
## # A tibble: 3 x 2
##   Branch mean_rating
##   <chr>        <dbl>
## 1 C             7.07
## 2 A             7.03
## 3 B             6.82
```

```
# which product is used by different group of Gender
product_usage <- df %>%
  group_by(Gender, 'Product line') %>%
  summarize(Count = n()) %>%
  arrange(desc(Count))

ggplot(product_usage, aes(x = 'Product line', y = Count,
  fill = Gender)) +
  geom_bar(stat = "identity", position = "dodge") +
  labs(
    title = "Product Usage by Gender",
    x = "Product Line",
    y = "Count",
    fill = "Gender"
  ) +
  theme_minimal() +
  theme(
    legend.position = "right"
  )+ ylim(0,600)
```



```
product_usage_city <- df %>%
  group_by(City, 'Product line') %>%
  summarize(Count = n()) %>%
  arrange(desc(Count))

## ‘summarise()‘ has grouped output by ’City’. You can override using the
## ‘.groups‘ argument.

names(product_usage_city) <- c('City','Product','Count')
product_usage_city

## # A tibble: 3 x 3
## # Groups:   City [3]
##   City      Product      Count
##   <chr>     <chr>        <int>
## 1 Yangon    Product line   340
## 2 Mandalay  Product line   332
## 3 Naypyitaw Product line   328
```

Task 3

Refer the Opinion published on Himalayan times on Dec 19, 2023 and perform a text preprocessing and generate word cloud.

```
#
import requests
from bs4 import BeautifulSoup
url = 'https://thehimalayantimes.com/opinion/navigating-nepals-digital-frontier-\
understanding-cybersecurity-in-the-digital-age-ensuring-data-safety-and-the-role-of-ai'

x = requests.get(url)
soup = BeautifulSoup(x.content, 'html.parser')
post_content = soup.find('div', {'class': 'post-content'})
paragraphs = post_content.find_all('p')
final_list = ''
for paragraph in range(0, len(paragraphs)-2):
    final_list += (paragraphs[paragraph].text)

with open('himalayan_times.txt','w') as file:
    file.write(final_list)
```

```
## 5513
```

```
library(tm)
library(Rgraphviz)
library(wordcloud)

text_document <- readLines('himalayan_times.txt')
corpus <- Corpus(VectorSource(text_document))
```

Text Preprocessing:

- Remove Punctutaion
- Remove Stop Words
- Stemming
- Convert to Lower
- Remove any Numbers
- Any customer remove words

```
my_stopwords <- c("can","may","used")
corpus <- tm_map(corpus, removeWords, my_stopwords)
my_tdm <- TermDocumentMatrix(
  corpus,
  control =
    list(
      removePunctuation = TRUE,
      stopwords = TRUE,
      tolower = TRUE,
      stemming = FALSE,
      removeNumbers = TRUE,
      bounds = list(global = c(1, Inf)),
      wordLenghts = c(1,Inf),
      removeWords = (c("can","may","used")))
)
```

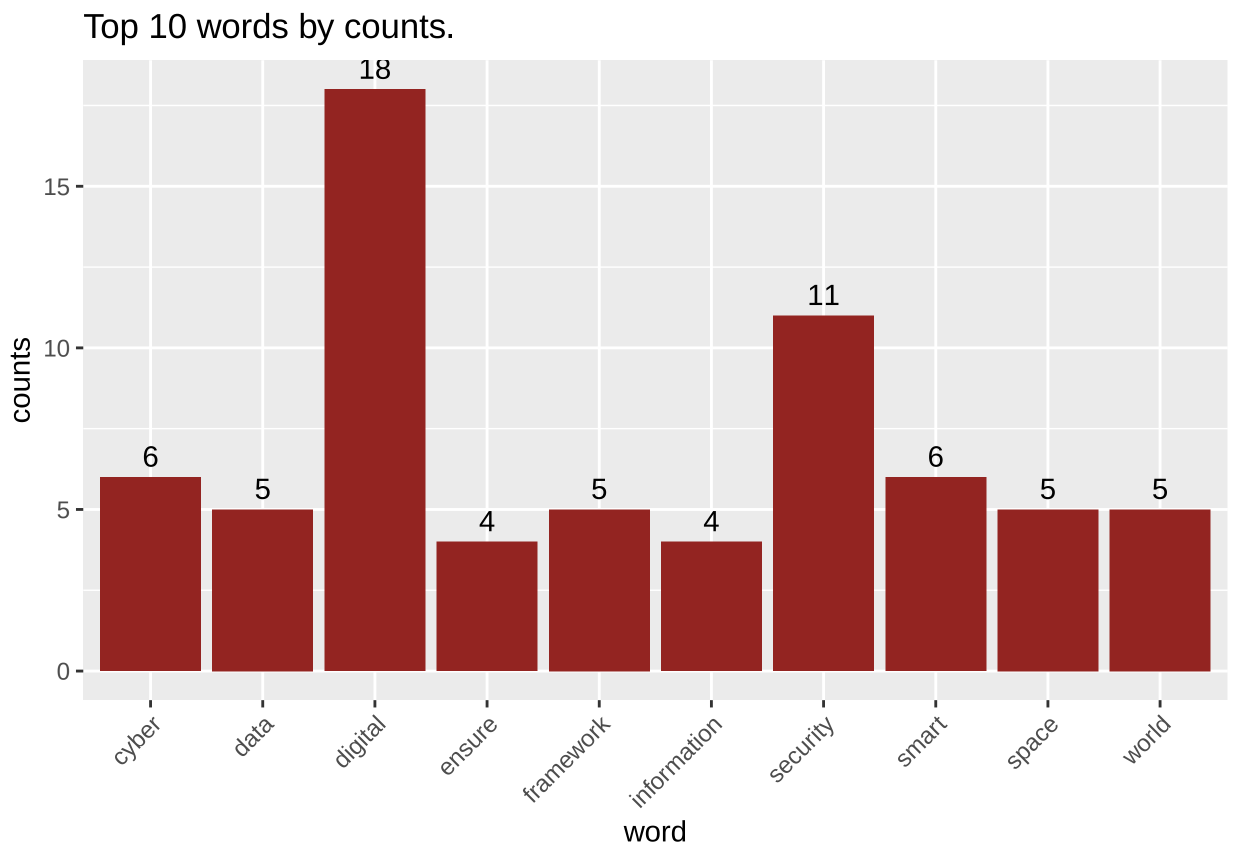
```
# find the frequent_terms in the corpus
frequent_terms <- findFreqTerms(my_tdm)
head(frequent_terms,20)
```

```
##  [1] "ability"       "access"        "accessed"      "achieve"
##  [5] "additionally"  "adoption"      "advances"      "advent"
##  [9] "ais"           "aithe"         "aligned"       "alikebuilding"
## [13] "allocated"     "allocating"    "allowing"      "along"
## [17] "already"       "also"          "always"        "amounts"
```

```
mat <- as.matrix(my_tdm)
freq <- mat %>% rowSums() %>% sort(decreasing = T)

df <- my_tdm %>%
      as.matrix() %>%
      rowSums() %>%
      sort(decreasing = TRUE) %>%
      head(10) %>%
      enframe(name = "word", value = "counts")
head(df)
```

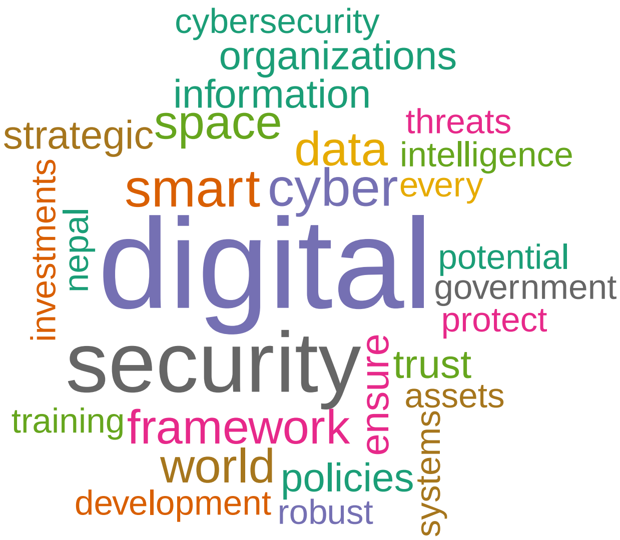
```
## # A tibble: 6 x 2
##   word      counts
##   <chr>      <dbl>
## 1 digital       18
## 2 security      11
## 3 cyber          6
## 4 smart          6
## 5 data           5
## 6 framework      5
```

```
# top 10 words and counts using bargraph
library(ggplot2)
ggplot(df, aes(word, counts)) +
  geom_bar(stat = "identity", fill = "#932421") +
  theme(axis.text.x = element_text(angle = 45, hjust = 1)) +
  labs(title = "Top 10 words by counts.") +
  geom_text(aes(label = counts), vjust = -0.5)
```



```
# plot word cloud
wordcloud(
  words = names(freq),
  freq = freq,
  random.order = FALSE,
  colors = brewer.pal(8, "Dark2"),
  scale = c(4, 0.5),
  random.color = TRUE,
)
grid.text("Himalayan Times Word Cloud", x = 0.5, y = 0.9, gp = gpar(fontsize = 18, fontface = "bold"))
```

Himalayan Times Word Cloud


cybersecurity
organizations
information
strategic
space
threats
data
intelligence
smart
cyber
every
investments
nepal
digital
potential
government
protect
security
ensure
trust
assets
training
framework
systems
world
policies
development
robust