Practice 1

Smit Patil

```
#Problem 1

#Importing Libraries
library(data.table)

#Importing Data
cust_data <- read.csv('customertxndata.csv')
setDT(cust_data)

#Total Revenue
sum_revenue <- cust_data[,na.omit(sum(revenue))]
sprintf("total transaction amount: %s", sum_revenue)
```

```
## [1] "total transaction amount: 10372523.7237093"
```

```
#Mean Visits
mean_visits <- cust_data[,na.omit(mean(visits))]
sprintf("mean number of visits: %s", mean_visits)
```

```
## [1] "mean number of visits: 12.4864912280702"
```

```
#Mean Revenue
median_revenue <- cust_data[,na.omit(median(revenue))]
sprintf("median revenue: %s", median_revenue)
```

```
## [1] "median revenue: 344.6516138"
```

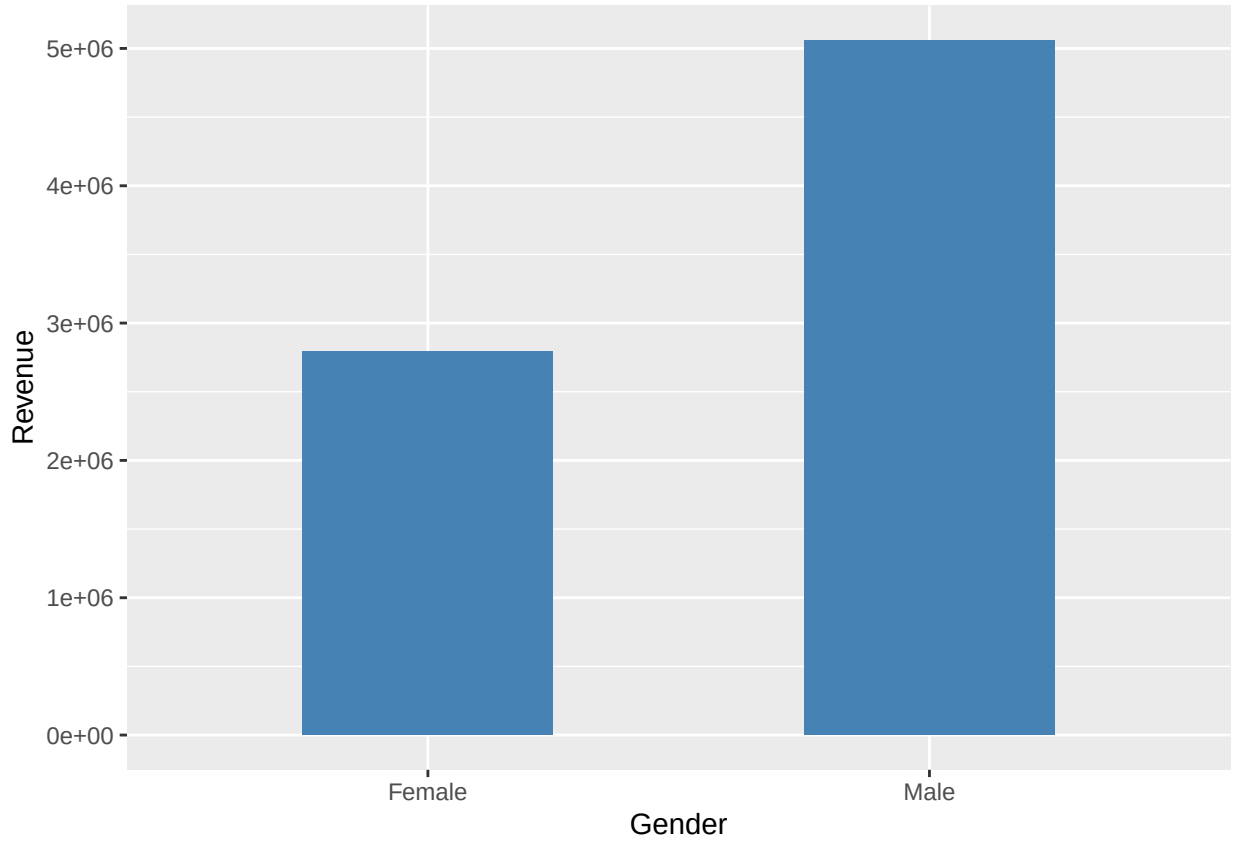
```
#Median Revenue
SD_revenue <- cust_data[,na.omit(sd(revenue))]
sprintf("standard deviation of revenue: %s", SD_revenue)
```

```
## [1] "standard deviation of revenue: 425.988388355701"
```

```
#Most Common Gender
gender_count <- cust_data[,.(count = .N), by = "gender"][order(-count)]
sprintf("most common gender is %s", gender_count[1,1])
```

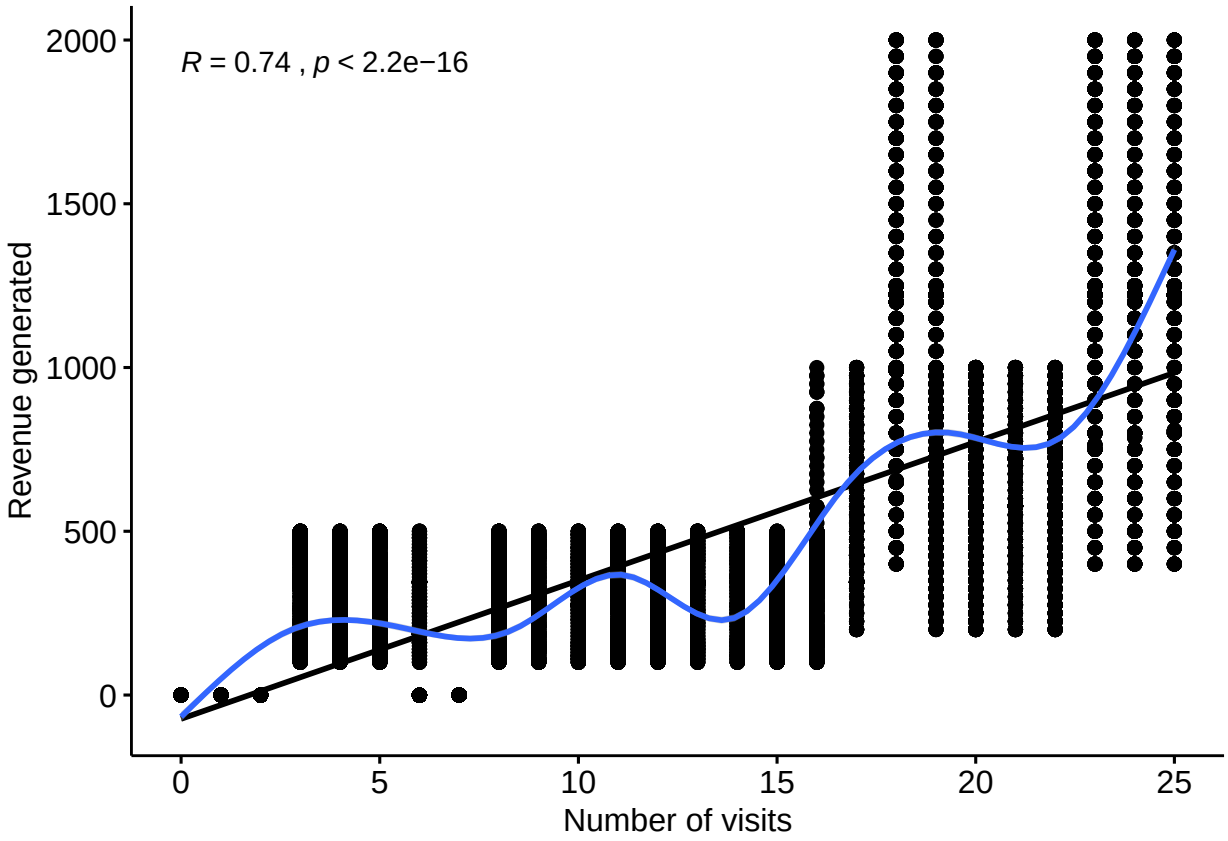
```
## [1] "most common gender is Male"
```

```r
#Problem 2
library("ggplot2")

#Total revenue by gender
revenue_by_gender <- cust_data[, sum(revenue), by = gender]
revenue_by_gender <- na.omit(revenue_by_gender)

#Bar chart
ggplot(data=revenue_by_gender, aes(x=gender, y=V1)) +
  geom_bar(stat="identity", fill="steelblue", width = 0.5) +
  xlab("Gender") + ylab("Revenue")
```




```r
#Problem 3
library("ggpubr")

#Pearson Moment of Correlation between number of visits and revenue
ggscatter(cust_data, x = "visits", y = "revenue",
          add = "reg.line", conf.int = TRUE,
          cor.coef = TRUE, cor.method = "pearson",
          xlab = "Number of visits", ylab = "Revenue generated") +
  geom_smooth(method = 'gam', formula = y ~ s(x, bs = "cs"))
```

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




```
#Problem 4
library(tidyverse)
```

```
## -- Attaching packages ---------------------------------------------------------------------------

## v tibble  3.0.1     v dplyr   0.8.5
## v tidyr   1.0.3     v stringr 1.4.0
## v readr   1.3.1     v forcats 0.5.0
## v purrr   0.3.4

## -- Conflicts ------------------------------------------------------------------------------------
## x dplyr::between()   masks data.table::between()
## x dplyr::filter()    masks stats::filter()
## x dplyr::first()     masks data.table::first()
## x dplyr::lag()       masks stats::lag()
## x dplyr::last()      masks data.table::last()
## x purrr::transpose() masks data.table::transpose()
```

```
#creating a table containing only the NA values
missing_data <- cust_data %>% filter_all(any_vars(is.na(.)))

#rows having missing data is in datatabe "missing_data"

#Every column has missing data, and 7,200 rows have missing values.
#We can impute the missing values by the mean of the column, and for columns having
#outliers we must ignore them before calculating the mean.
```

```r
#Problem 5

#Mean transactions
clean_cust_data <- na.omit(cust_data)

#Calculating mean for transactions
clean_cust_data[, round(mean(transactions))]
```

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

```r
#Assigning mean value to NA in transactions column
impute_trans <- cust_data
impute_trans[is.na(cust_data$transactions)] <- 1

#Creating function for mode
getmode <- function(v)
  {
   uniqv <- unique(v)
   uniqv[which.max(tabulate(match(v, uniqv)))]
  }

#Mode of gender
cust_data[, getmode(gender)]
```

```
## [1] "Male"
```

```r
#Assigning mode value to NA in gender column
impute_gender <- cust_data
impute_gender[is.na(cust_data$gender)] <- "Male"
```

```
## Warning in `[<-.data.table`(`*tmp*`, is.na(cust_data$gender), value = "Male"):
## Coercing 'character' RHS to 'integer' to match the type of the target column
## (column 1 named 'visits').

## Warning in `[<-.data.table`(`*tmp*`, is.na(cust_data$gender), value = "Male"):
## NAs introduced by coercion

## Warning in `[<-.data.table`(`*tmp*`, is.na(cust_data$gender), value = "Male"):
## Coercing 'character' RHS to 'integer' to match the type of the target column
## (column 2 named 'transactions').

## Warning in `[<-.data.table`(`*tmp*`, is.na(cust_data$gender), value = "Male"):
## NAs introduced by coercion

## Warning in `[<-.data.table`(`*tmp*`, is.na(cust_data$gender), value = "Male"):
## Coercing 'character' RHS to 'double' to match the type of the target column
## (column 5 named 'revenue').

## Warning in `[<-.data.table`(`*tmp*`, is.na(cust_data$gender), value = "Male"):
## NAs introduced by coercion
```

```r
#Problem 6

#Spliting the data into a trainig and validation dataset

training_data <- cust_data[rep(c(TRUE,FALSE), length = .N), ]
#logical vector c(TRUE,FALSE) will only return 1st,3rd,5th...values

validation_data <- cust_data[rep(c(FALSE,TRUE), length = .N), ]
#logical vector c(FALSE,TRUE) will only return 2nd,4th,6th...values
```

```r
#Problem 7

#Calculating mean revenue for trainig dataset
mean_train <- training_data[,na.exclude(mean(revenue))]
sprintf("mean of training data: %s", mean_train)
```

```
## [1] "mean of training data: 449.610487789368"
```

```r
#Calculating mean revenue for validation dataset
mean_val <- validation_data[,na.exclude(mean(revenue))]
sprintf("mean of validation data: %s", mean_val)
```

```
## [1] "mean of validation data: 460.260014290395"
```

```r
#Problem 8
set.seed(77654)

#Creating first sample to set 60% of data for training
sample_1 <- sample.int(n = nrow(cust_data), size = floor(.60*nrow(cust_data)), replace = F)

#Creating dataset for training
training <- cust_data[sample_1,]

#Creating a dataset of the remaining data for testing and validation
remaining_data <- cust_data[-sample_1,]

#Creating second sample to split the reamining dataset for testing and validation
sample_2 <- sample.int(n = nrow(remaining_data), size = floor(.50*nrow(remaining_data)), replace = F)

#Creating dataset for testing
testing <- remaining_data[sample_2, ]

#Creating dataset for validation
validation <- remaining_data[-sample_2, ]
```