

Practice 1

Jordan Lian

1/31/2021

An organization has collected data on customer visits, transactions, operating system, and gender and desires to build a model to predict revenue. For the moment, the goal is to prepare the data for modeling. Analyze the data set in the following manner:

1. (0 pts) Either install [Base R](#) and [R Studio](#) on your computer or create an account at [RStudio.cloud](#) and then learn how to build [R Markdown Notebooks](#) to execute your code and organize your output into a readable report. For those working on Windows, you may also use [Microsoft Open R](#).
2. (5 pts) Download this [data set](#) and then upload the data into RStudio Cloud. Each row represents a customer's interactions with the organization's web store. The first column is the number of visits of a customer, the second the number of transactions of that customer, the third column is the customer's operating system, and the fourth column is the customer's reported gender, while the last column is revenue, i.e., the total amount spent by that customer.

I imported the tidyverse and readr libraries and used the `read_csv()` function to import the data set.

```
library(tidyverse)

## -- Attaching packages -----

## v ggplot2 3.3.3      v purrr  0.3.4
## v tibble  3.0.3      v dplyr  1.0.2
## v tidyr   1.1.2      v stringr 1.4.0
## v readr   1.3.1      v forcats 0.5.0

## -- Conflicts -----
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()

library(readr)

setwd("C:/Users/Jordan Lian/OneDrive - Northeastern University/Spring 2021/DA 5030/Practice 1")
data <- read_csv(file = "customertxndata.csv", header = FALSE)
names(data) <- c('Visits', 'Transactions', 'Op_Sys', 'Gender', 'Revenue')
head(data)

##   Visits Transactions Op_Sys Gender  Revenue
## 1      7            0 Android  Male    0.0000
## 2     20            1    iOS   <NA> 576.8668
## 3     22            1    iOS  Female 850.0000
```

```
## 4      24          2      iOS Female 1050.0000
## 5       1          0 Android   Male    0.0000
## 6      13          1 Android   Male   460.0000
```

3. (10 pts) Calculate the following summative statistics: total transaction amount (revenue), mean number of visits, median revenue, standard deviation of revenue, most common gender. Exclude any cases where there is a missing value.

I used basic descriptive statistical functions to get the desired values.

```
# Total Transaction Amount (Revenue)
sum(data$Revenue)
```

```
## [1] 10372524
```

```
# Mean Number of Visits
mean(data$Visits)
```

```
## [1] 12.48649
```

```
# Median Revenue
median(data$Revenue)
```

```
## [1] 344.6516
```

```
# Standard Deviation of Revenue
sd(data$Revenue)
```

```
## [1] 425.9884
```

```
# Most Common Gender (Exclude any cases where there is a missing value)
gender <- data[!is.na(data$Gender),]
max(gender$Gender)
```

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

4. (10 pts) Create a bar/column chart of gender (x-axis) versus revenue (y-axis). Omit missing values, i.e., where gender is NA or missing.

There are two ways you can do this.

1. Use the `aggregate()` function to get the sum of the revenue by gender.
2. Mutate the original data frame using `%>%`, and use `group_by`, `summarise()`, and `sum()`.

Either way, either option will yield a table that you can use `barplot` to then graph.

```
library(dplyr)

# Aggregate the total revenue based on gender
total_revenue <- aggregate(Revenue~Gender, gender, sum)
names(total_revenue)[2] <- 'Revenue'
total_revenue
```

```
## Gender Revenue
## 1 Female 2790500
## 2 Male 5059692
```

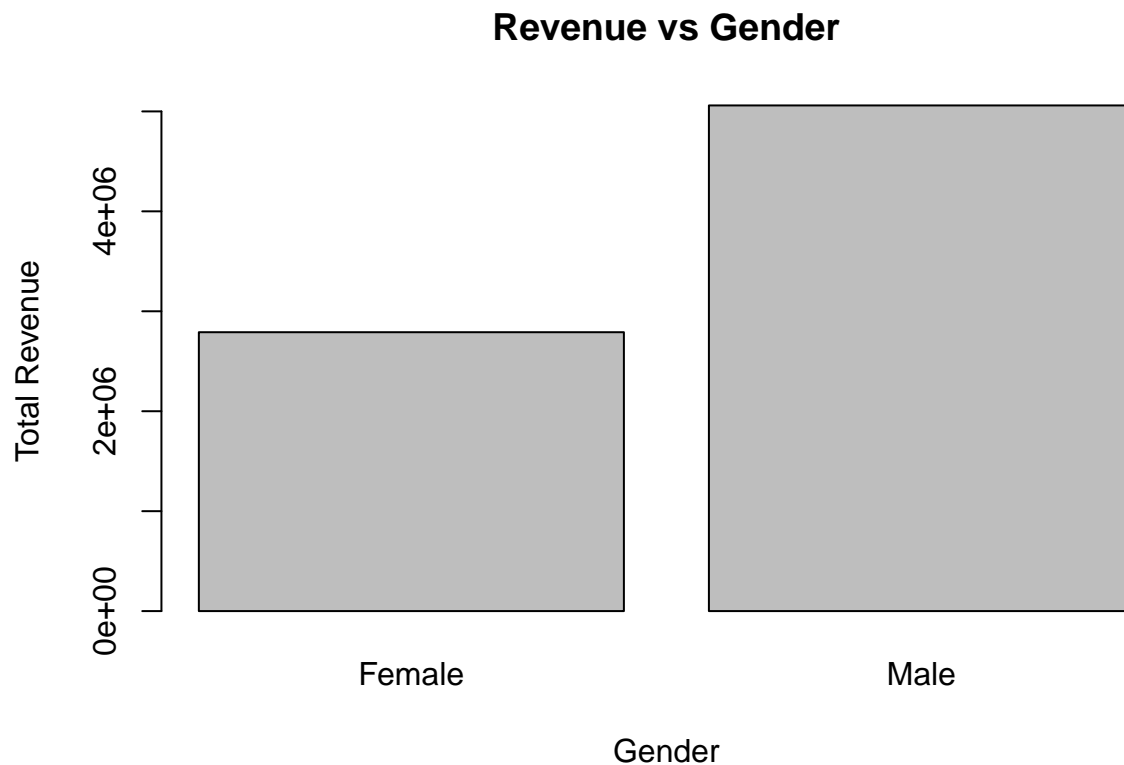
```
# Mutate the data frame using %>%
total_revenue <- gender %>%
  group_by(Gender) %>%
  summarise(Revenue=sum(Revenue))
```

```
## 'summarise()' ungrouping output (override with '.groups' argument)
```

```
# Both methods get the same result
total_revenue
```

```
## # A tibble: 2 x 2
##   Gender Revenue
##   <chr>     <dbl>
## 1 Female 2790500
## 2 Male 5059692.
```

```
# Plot: Revenue vs Gender
barplot(total_revenue$Revenue, main="Revenue vs Gender",
        xlab="Gender",
        ylab="Total Revenue",
        names.arg=c("Female", "Male"))
```



5. (5 pts) What is the Pearson Moment of Correlation between number of visits and revenue? Comment on the correlation.

To calculate the correlation value, I used the `cor.test()` function. To plot the graph and show the correlation value, I used `ggplot()`, `geom_point()`, `geom_smooth()`, and `stat_cor()` to plot points, to get a smooth regression line, and to show the correlation value respectively.

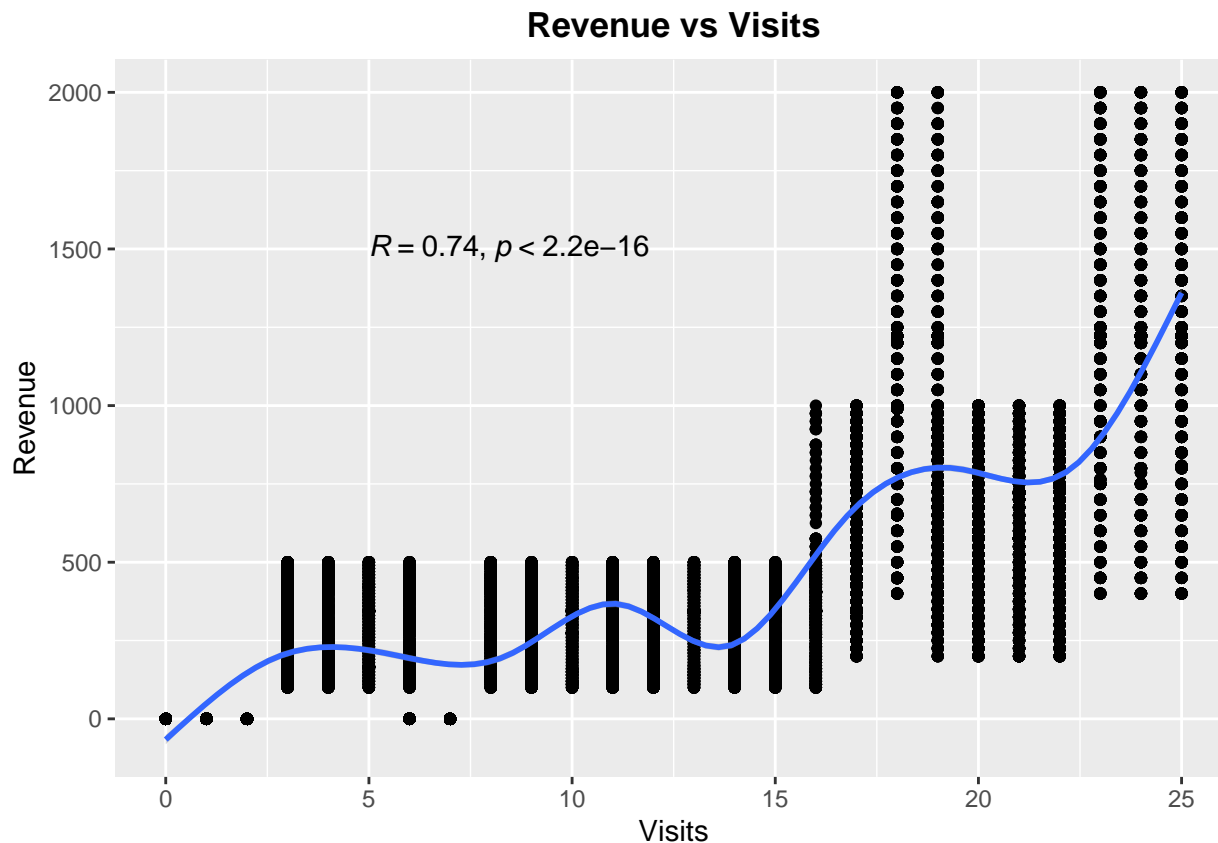
```
library(ggplot2)
library(ggpubr)

# Correlation Calculation
cor.test(data$Visits, data$Revenue)

##
## Pearson's product-moment correlation
##
## data: data$Visits and data$Revenue
## t = 165.55, df = 22798, p-value < 2.2e-16
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
## 0.7328932 0.7446832
## sample estimates:
## cor
## 0.7388448

# Plot: Revenue vs Visits
ggplot(data, aes(x=Visits, y=Revenue)) +
  geom_point() +
  geom_smooth() +
  stat_cor(method = "pearson", label.x = 5, label.y = 1500) +
  ggtitle("Revenue vs Visits") +
  theme(plot.title = element_text(face = "bold", hjust = 0.5))

## 'geom_smooth()' using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
```



The correlation value is 0.74. This means that there is a strong correlation between the number of visits and revenue. The higher number of visits someone has, the more money that the company is likely to make.

6. (10 pts) Which columns have missing data? How did you recognize them? How would you impute missing values?

I used the `is.na()` function to get the columns with missing data. I counted the number of rows with missing data and called for any column with more than 0 rows with missing data. You can use the mean, median, or mode and insert one of those values into missing values, you can simply remove them from your data set, or you could use regression (linear, logistic, etc). I personally would impute missing values with one of the descriptive statistics that is most reflective of the data (depends on the distribution the outliers).

```
colnames(data)[colSums(is.na(data)) > 0]
```

```
## [1] "Transactions" "Gender"
```

7. (15 pts) Impute missing transaction and gender values. Use the mean for transaction (rounded to the nearest whole number) and the mode for gender.

Finding the mean transaction and the mode for gender

I created a new data frame, where I eliminated the rows where the transaction value was NA. I then used the mode for gender that I found in Question 3.

```
# Mean Transaction
mean_trans <- mean(data$Transactions, na.rm = TRUE)
mean_trans
```

```
## [1] 0.993
```

```
# Round Value
round(mean_trans)
```

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

```
# Mode for Gender
max(gender$Gender)
```

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

Imputing the missing transaction and gender values

To impute the values, I used `replace_na()` function along with `%>%` to replace the values given their respective columns.

```
library(tidyr)

# Insert the mean and mode into the respective columns using %>%
mod_data <- data %>% replace_na(list(Transactions = round(mean_trans), Gender = max(gender$Gender)))
head(mod_data)
```

```
##   Visits Transactions Op_Sys Gender  Revenue
## 1      7            0 Android  Male    0.0000
## 2     20            1    iOS   Male  576.8668
## 3     22            1    iOS  Female 850.0000
## 4     24            2    iOS  Female 1050.0000
## 5      1            0 Android  Male    0.0000
## 6     13            1 Android  Male   460.0000
```

8. (20 pts) Split the data set into two equally sized data sets where one can be used for training a model and the other for validation. Take every odd numbered case and add them to the training data set and every even numbered case and add them to the validation data set, i.e., row 1, 3, 5, 7, etc. are training data while rows 2, 4, 6, etc. are validation data.

I used the `dplyr::filter` function to split the dataset into two parts. I used the `%>%` operator to get odds and evens respectively.

```
# Training Data
training <- mod_data %>% dplyr::filter(row_number() %% 2 == 1) ## Odd rows

# Validation Data
validation <- mod_data %>% dplyr::filter(row_number() %% 2 == 0) ## Even rows

head(training)
```

```
##   Visits Transactions Op_Sys Gender Revenue
## 1      7           0 Android  Male      0
## 2     22           1    iOS Female    850
## 3      1           0 Android  Male      0
## 4     23           2    iOS  Male   1850
## 5     11           1 Android  Male    110
## 6     17           1 Android  Male    225
```

```
head(validation)
```

```
##   Visits Transactions Op_Sys Gender Revenue
## 1     20           1    iOS  Male  576.8668
## 2     24           2    iOS Female 1050.0000
## 3     13           1 Android  Male  460.0000
## 4     14           1 Android  Male  480.0000
## 5     24           2    iOS  Male 1950.0000
## 6     14           1 Android  Male  344.6516
```

9. (10 pts) Calculate the mean revenue for the training and the validation data sets and compare them. Comment on the difference.

```
# Training Data
mean(training$Revenue)
```

```
## [1] 449.6105
```

```
# Validation Data
mean(validation$Revenue)
```

```
## [1] 460.26
```

```
# Average of the two means vs Actual Mean
0.5 * (mean(training$Revenue) + mean(validation$Revenue))
```

```
## [1] 454.9353
```

```
mean(data$Revenue)
```

```
## [1] 454.9353
```

The means are pretty similar, which is to be expected. The average of the two means should equal the actual mean, which we mathematically verified above.

10. (15 pts) For many data mining and machine learning tasks, there are packages in R. Use the `sample()` function to split the data set, so that 60% is used for training and 20% is used for testing, and another 20% is used for validation. To ensure that your code is reproducible and that everyone gets the same result, use the number 77654 as your seed for the random number generator. Use the code fragment below for reference:

```

# Reference Code
set.seed(101) # Set Seed so that same sample can be reproduced in future also
# Now Selecting 75% of data as sample from total 'n' rows of the data
sample <- sample.int(n = nrow(data), size = floor(.75*nrow(data)), replace = F)
train <- data[sample, ]
test <- data[-sample, ]

# Set Seed so that same sample can be reproduced in future also
set.seed(77654)

# Get 60% of data as the training set
sample_1 <- sample.int(n = nrow(mod_data), size = floor(.60*nrow(mod_data)), replace = F)
train <- mod_data[sample_1, ]
test_val <- mod_data[-sample_1, ]

# Split the Remaining 40%
sample_2 <- sample.int(n = nrow(test_val), size = floor(.50*nrow(test_val)), replace = F)
test <- test_val[sample_2, ]
valid <- test_val[-sample_2, ]

# Datasets
head(train)

```

```

##      Visits Transactions Op_Sys Gender Revenue
## 19633      9           1 Android  Male  344.6516
## 14836      5           1 Android  Male  100.0000
## 19590      2           0 Android  Male    0.0000
## 13325     25           2     iOS   Male 1750.0000
## 11894      2           0 Android  Male    0.0000
## 16385      8           1 Android  Male  344.6516

```

```
head(test)
```

```

##      Visits Transactions Op_Sys Gender Revenue
## 4477      13           1 Android  Male  340.0000
## 9457      22           1     iOS   Male  576.8668
## 11134     19           1     iOS   Male  576.8668
## 17249     20           1     iOS   Male  575.0000
## 973       22           1     iOS   Male  450.0000
## 5891       7           0 Android  Male    0.0000

```

```
head(valid)
```

```

##      Visits Transactions Op_Sys Gender Revenue
## 2        20           1     iOS   Male  576.8668
## 5         1           0 Android  Male    0.0000
## 6        13           1 Android  Male  460.0000
## 10       24           2     iOS   Male 1950.0000
## 14        8           1 Android  Male  344.6516
## 36       22           1     iOS   Male  450.0000

```