# Week13_Core

Brendah

2022-03-26

##1. Introduction ### 1.1Research Question Use clustering (K-Means and Hierarchical) to group customers according to their behavior.

### 2.Metric for Success

### 3.Context

Kira Plastinina is a Russian brand that is sold through a defunct chain of retail stores in Russia, Ukraine, Kazakhstan, Belarus, China, Philippines, and Armenia. The brand's Sales and Marketing team would like to understand their customer's behavior from data that they have collected over the past year. More specifically, they would like to learn the characteristics of customer groups.

### 4.Experimental Design

1.Problem Definition 2.Data Sourcing 3.Check the Data 4.Perform Data Cleaning 5.Perform Exploratory Data Analysis (Univariate, Bivariate & Multivariate) 6.Implement the Solution 7.Challenge the Solution 8.Follow up Questions

### 5.Appropriateness of Data

The dataset can be found here: http://bit.ly/EcommerceCustomersDataset

Variable definitions are as follows:

.The dataset consists of 10 numerical and 8 categorical attributes. The 'Revenue' attribute can be used as the class label.

."Administrative", "Administrative Duration", "Informational", "Informational Duration", "Product Related" and "Product Related Duration" represents the number of different types of pages visited by the visitor in that session and total time spent in each of these page categories. The values of these features are derived from the URL information of the pages visited by the user and updated in real-time when a user takes an action, e.g. moving from one page to another.

.The "Bounce Rate", "Exit Rate" and "Page Value" features represent the metrics measured by "Google Analytics" for each page in the e-commerce site.

.The value of the "Bounce Rate" feature for a web page refers to the percentage of visitors who enter the site from that page and then leave ("bounce") without triggering any other requests to the analytics server during that session. The value of the "Exit Rate" feature for a specific web page is calculated as for all pageviews to the page, the percentage that was the last in the session.

.The "Page Value" feature represents the average value for a web page that a user visited before completing an e-commerce transaction.

.The “Special Day” feature indicates the closeness of the site visiting time to a specific special day (e.g. Mother’s Day, Valentine’s Day) in which the sessions are more likely to be finalized with the transaction. The value of this attribute is determined by considering the dynamics of e-commerce such as the duration between the order date and delivery date. For example, for Valentine’s day, this value takes a nonzero value between February 2 and February 12, zero before and after this date unless it is close to another special day, and its maximum value of 1 on February 8.

.The dataset also includes the operating system, browser, region, traffic type, visitor type as returning or new visitor, a Boolean value indicating whether the date of the visit is weekend, and month of the year.

```
1. Perform clustering stating insights drawn from your analysis and visualizations.
2. Upon implementation, provide comparisons between the approaches learned this week i.e. K-Means cl
```

### 2. Data Cleaning & Preparation

```
data<-read.csv("http://bit.ly/EcommerceCustomersDataset")
head(data)
```

```
##   Administrative Administrative_Duration Informational Informational_Duration
## 1              0                       0             0                      0
## 2              0                       0             0                      0
## 3              0                      -1             0                     -1
## 4              0                       0             0                      0
## 5              0                       0             0                      0
## 6              0                       0             0                      0
##   ProductRelated ProductRelated_Duration BounceRates ExitRates PageValues
## 1              1                0.000000  0.20000000 0.2000000          0
## 2              2               64.000000  0.00000000 0.1000000          0
## 3              1               -1.000000  0.20000000 0.2000000          0
## 4              2                2.666667  0.05000000 0.1400000          0
## 5             10              627.500000  0.02000000 0.0500000          0
## 6             19              154.216667  0.01578947 0.0245614          0
##   SpecialDay Month OperatingSystems Browser Region TrafficType
## 1          0   Feb                1       1      1           1
## 2          0   Feb                2       2      1           2
## 3          0   Feb                4       1      9           3
## 4          0   Feb                3       2      2           4
## 5          0   Feb                3       3      1           4
## 6          0   Feb                2       2      1           3
##         VisitorType Weekend Revenue
## 1 Returning_Visitor   FALSE   FALSE
## 2 Returning_Visitor   FALSE   FALSE
## 3 Returning_Visitor   FALSE   FALSE
## 4 Returning_Visitor   FALSE   FALSE
## 5 Returning_Visitor    TRUE   FALSE
## 6 Returning_Visitor   FALSE   FALSE
```

```
# displaying the number of rows and columns
dim(data)
```

```
## [1] 12330    18
```

As seen, the dataset has 12,330 rows and 18 columns

```
# previewing our dataset's basic information
str(data)
```

```
## 'data.frame':    12330 obs. of  18 variables:
##  $ Administrative         : int  0 0 0 0 0 0 0 1 0 0 ...
##  $ Administrative_Duration: num  0 0 -1 0 0 0 -1 -1 0 0 ...
##  $ Informational          : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ Informational_Duration : num  0 0 -1 0 0 0 -1 -1 0 0 ...
##  $ ProductRelated         : int  1 2 1 2 10 19 1 1 2 3 ...
##  $ ProductRelated_Duration: num  0 64 -1 2.67 627.5 ...
##  $ BounceRates            : num  0.2 0 0.2 0.05 0.02 ...
##  $ ExitRates              : num  0.2 0.1 0.2 0.14 0.05 ...
##  $ PageValues             : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ SpecialDay             : num  0 0 0 0 0 0 0.4 0 0.8 0.4 ...
##  $ Month                  : chr  "Feb" "Feb" "Feb" "Feb" ...
##  $ OperatingSystems       : int  1 2 4 3 3 2 2 1 2 2 ...
##  $ Browser                : int  1 2 1 2 3 2 4 2 2 4 ...
##  $ Region                 : int  1 1 9 2 1 1 3 1 2 1 ...
##  $ TrafficType            : int  1 2 3 4 4 3 3 5 3 2 ...
##  $ VisitorType            : chr  "Returning_Visitor" "Returning_Visitor" "Returning_Visitor" "Returni
##  $ Weekend                : logi  FALSE FALSE FALSE FALSE TRUE FALSE ...
##  $ Revenue                : logi  FALSE FALSE FALSE FALSE FALSE FALSE ...
```

```
# checking for duplicated records
anyDuplicated(data)
```

```
## [1] 159
```

There are 159 duplicated records which will be removed to reduce redundancy

```
# removing duplicates
data <- unique(data)
dim(data)
```

```
## [1] 12211    18
```

the duplicated records have been removed

```
# re-checking for duplicated records
anyDuplicated(data)
```

```
## [1] 0
```

Just to confirm, now there are no duplicates

```
# checking for missing values
colSums(is.na(data))
```

```
##          Administrative Administrative_Duration           Informational
##                      12                      12                      12
##  Informational_Duration          ProductRelated ProductRelated_Duration
##                      12                      12                      12
##             BounceRates               ExitRates              PageValues
##                      12                      12                       0
##              SpecialDay                   Month        OperatingSystems
##                       0                       0                       0
##                 Browser                  Region             TrafficType
##                       0                       0                       0
##             VisitorType                 Weekend                 Revenue
##                       0                       0                       0
```

Since there seems to be enough records, it feels safe to drop the records with missing values

```
data <- na.omit(data)
colSums(is.na(data))
```

```
##          Administrative Administrative_Duration           Informational
##                       0                       0                       0
##  Informational_Duration          ProductRelated ProductRelated_Duration
##                       0                       0                       0
##             BounceRates               ExitRates              PageValues
##                       0                       0                       0
##              SpecialDay                   Month        OperatingSystems
##                       0                       0                       0
##                 Browser                  Region             TrafficType
##                       0                       0                       0
##             VisitorType                 Weekend                 Revenue
##                       0                       0                       0
```

Now there isn't any record with missing values

```
#just to ensure that the records are really enough and not much are lost
dim(data)
```

```
## [1] 12199    18
```

The OperatingSystems, Browser, Region, TrafficType weekend and revenue variables will be converted from the data type numerical to categorical to make them easier to work with

```
data$OperatingSystems <- as.factor(data$OperatingSystems)
data$Browser <- as.factor(data$Browser)
data$Region <- as.factor(data$Region)
data$TrafficType <- as.factor(data$TrafficType)
data$Weekend <- as.factor(data$Weekend)
data$Revenue <- as.factor(data$Revenue)
#just to confirm the changes
str(data)
```

```
## 'data.frame':    12199 obs. of  18 variables:
##  $ Administrative         : int  0 0 0 0 0 0 0 1 0 0 ...
##  $ Administrative_Duration: num  0 0 -1 0 0 0 -1 -1 0 0 ...
##  $ Informational          : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ Informational_Duration : num  0 0 -1 0 0 0 -1 -1 0 0 ...
##  $ ProductRelated         : int  1 2 1 2 10 19 1 1 2 3 ...
##  $ ProductRelated_Duration: num  0 64 -1 2.67 627.5 ...
##  $ BounceRates            : num  0.2 0 0.2 0.05 0.02 ...
##  $ ExitRates              : num  0.2 0.1 0.2 0.14 0.05 ...
##  $ PageValues             : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ SpecialDay             : num  0 0 0 0 0 0 0.4 0 0.8 0.4 ...
##  $ Month                  : chr  "Feb" "Feb" "Feb" "Feb" ...
##  $ OperatingSystems       : Factor w/ 8 levels "1","2","3","4",..: 1 2 4 3 3 2 2 1 2 2 ...
##  $ Browser                : Factor w/ 13 levels "1","2","3","4",..: 1 2 1 2 3 2 4 2 2 4 ...
##  $ Region                 : Factor w/ 9 levels "1","2","3","4",..: 1 1 9 2 1 1 3 1 2 1 ...
##  $ TrafficType            : Factor w/ 20 levels "1","2","3","4",..: 1 2 3 4 4 3 3 5 3 2 ...
##  $ VisitorType            : chr  "Returning_Visitor" "Returning_Visitor" "Returning_Visitor" "Returni
##  $ Weekend                : Factor w/ 2 levels "FALSE","TRUE": 1 1 1 1 2 1 1 2 1 1 ...
##  $ Revenue                : Factor w/ 2 levels "FALSE","TRUE": 1 1 1 1 1 1 1 1 1 1 ...
##  - attr(*, "na.action")= 'omit' Named int [1:12] 1050 1116 1117 1118 1119 1443 1444 1445 1446 1996 .
##   ..- attr(*, "names")= chr [1:12] "1066" "1133" "1134" "1135" ...
```

The variables are now in their appropriate data type

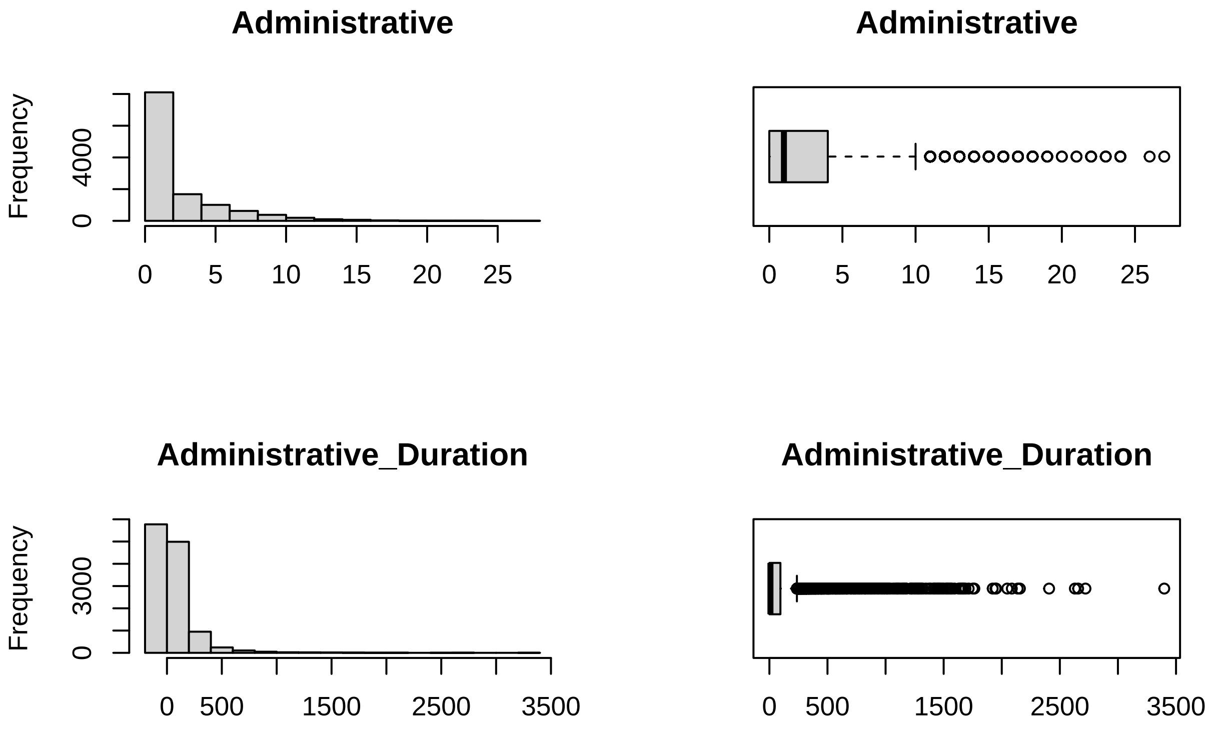
### 3. Exploratory Data Analysis

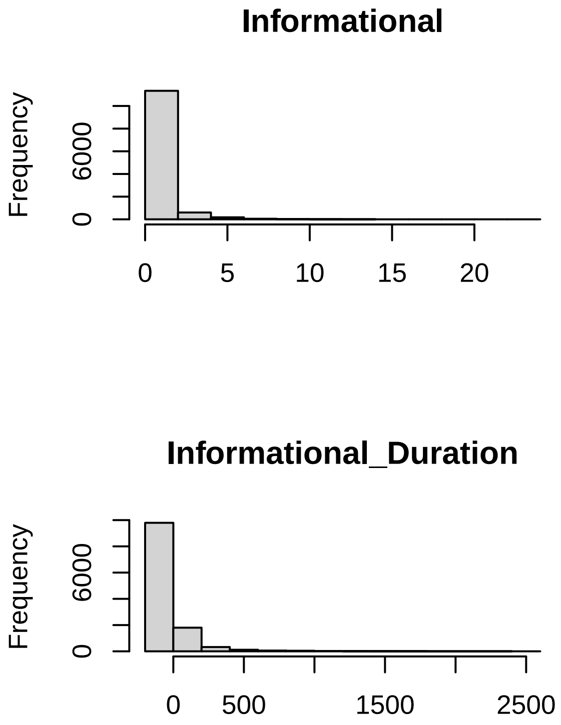
```
# getting the main summary of the dataset
summary(data)
```

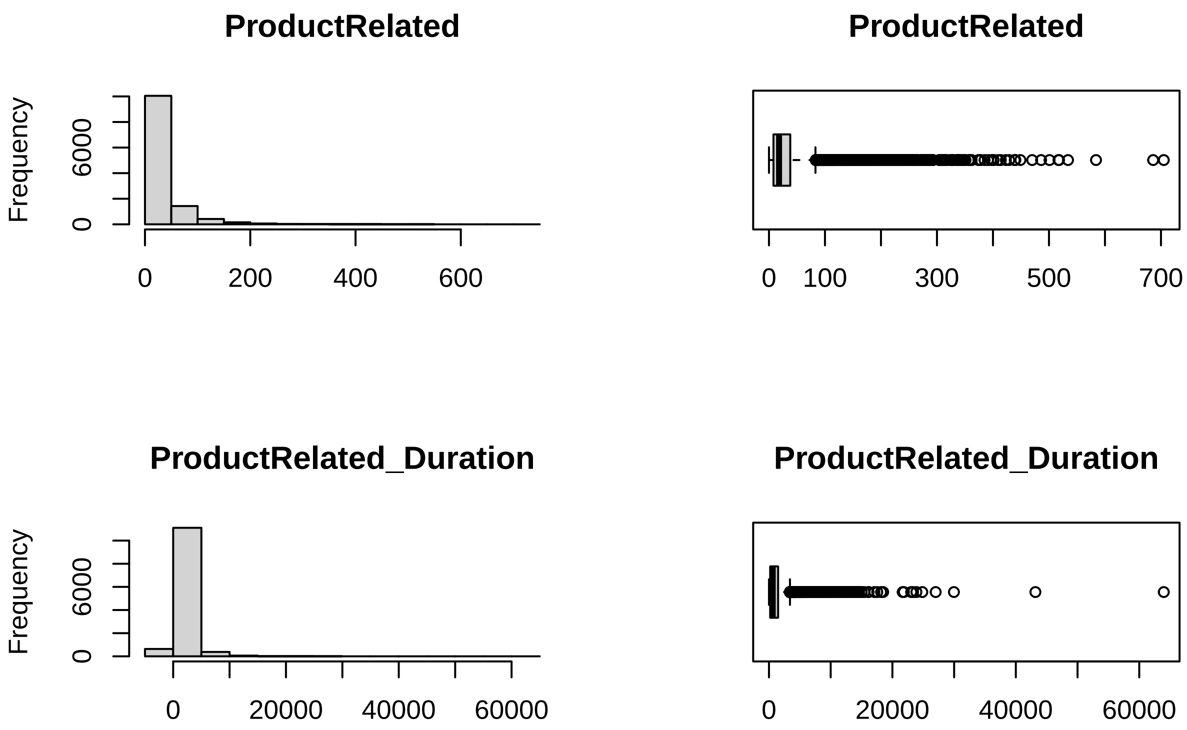
```
##  Administrative  Administrative_Duration Informational
##  Min.   : 0.00   Min.   :  -1.00         Min.   : 0.0000
##  1st Qu.: 0.00   1st Qu.:   0.00         1st Qu.: 0.0000
##  Median : 1.00   Median :   9.00         Median : 0.0000
##  Mean   : 2.34   Mean   :  81.68         Mean   : 0.5088
##  3rd Qu.: 4.00   3rd Qu.:  94.75         3rd Qu.: 0.0000
##  Max.   :27.00   Max.   :3398.75         Max.   :24.0000
##
##  Informational_Duration ProductRelated   ProductRelated_Duration
##  Min.   :  -1.00        Min.   :  0.00   Min.   :   -1.0
##  1st Qu.:   0.00        1st Qu.:  8.00   1st Qu.:  193.6
##  Median :   0.00        Median : 18.00   Median :  609.5
##  Mean   :  34.84        Mean   : 32.06   Mean   : 1207.5
##  3rd Qu.:   0.00        3rd Qu.: 38.00   3rd Qu.: 1477.6
##  Max.   :2549.38        Max.   :705.00   Max.   :63973.5
##
##   BounceRates        ExitRates         PageValues        SpecialDay
##  Min.   :0.00000   Min.   :0.00000   Min.   :  0.000   Min.   :0.00000
##  1st Qu.:0.00000   1st Qu.:0.01422   1st Qu.:  0.000   1st Qu.:0.00000
##  Median :0.00293   Median :0.02500   Median :  0.000   Median :0.00000
##  Mean   :0.02045   Mean   :0.04150   Mean   :  5.952   Mean   :0.06197
##  3rd Qu.:0.01667   3rd Qu.:0.04848   3rd Qu.:  0.000   3rd Qu.:0.00000
##  Max.   :0.20000   Max.   :0.20000   Max.   :361.764   Max.   :1.00000
##
##     Month            OperatingSystems    Browser         Region
##  Length:12199       2      :6536     2      :7878   1      :4711
##  Class :character   1      :2548     1      :2426   3      :2382
##  Mode  :character   3      :2530     4      : 730   4      :1168
##                     4      : 478     5      : 466   2      :1127
##                     8      :  75     6      : 174   6      : 800
##                     6      :  19     10     : 163   7      : 758
##                     (Other):  13     (Other): 362   (Other):1253
##   TrafficType   VisitorType         Weekend      Revenue
##  2      :3907   Length:12199       FALSE:9343   FALSE:10291
##  1      :2383   Class :character   TRUE :2856   TRUE : 1908
##  3      :2017   Mode  :character
##  4      :1066
##  13     : 728
##  10     : 450
##  (Other):1648
```

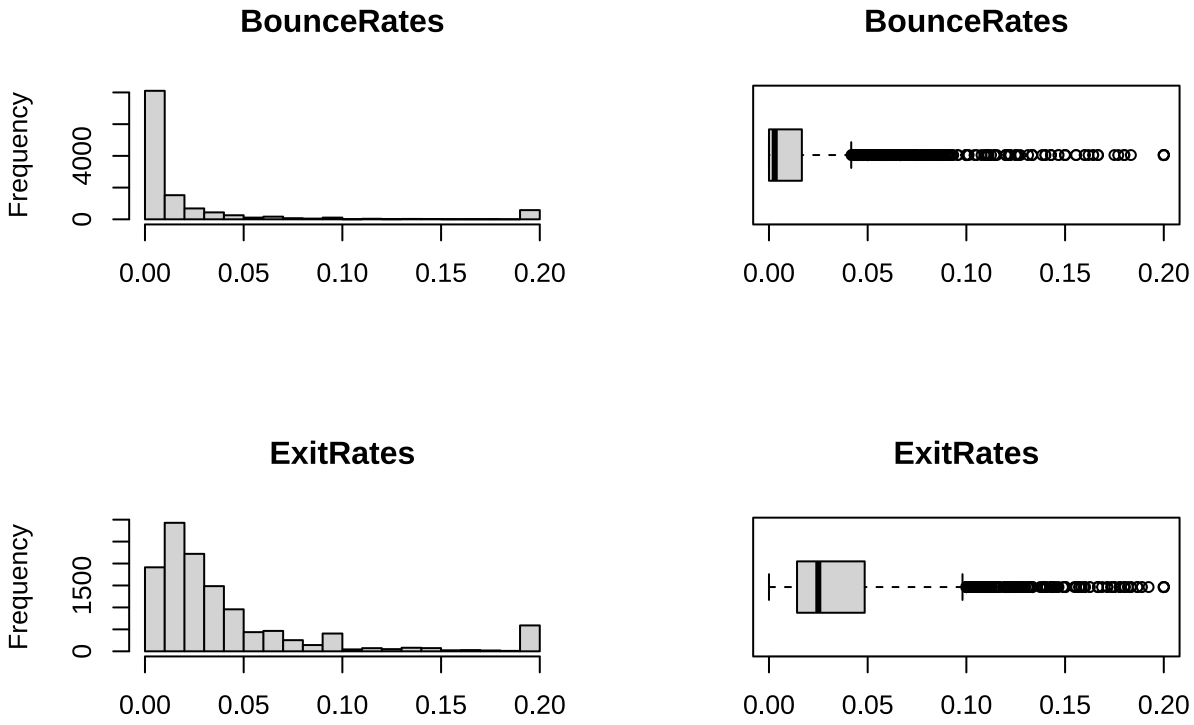
### 3.1 Univariate Analysis

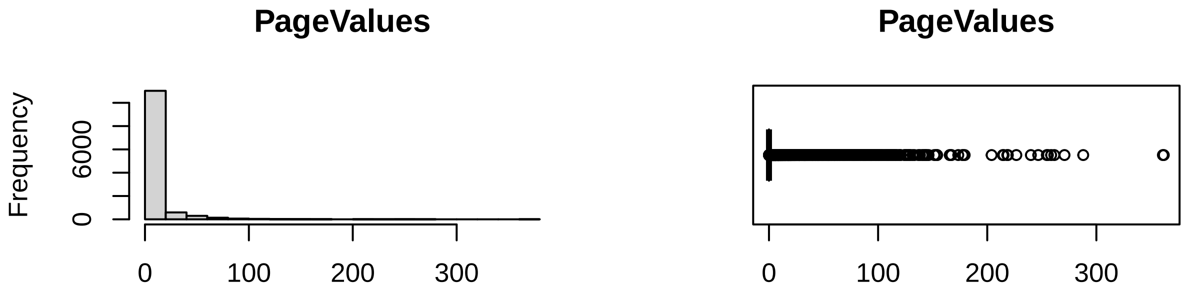
```
# previewing the numerical variables' histograms and boxplots
par(mfrow=c(2,2))
for(i in 1:10) {
   hist(data[, i], main=names(data)[i], xlab = NULL)
    boxplot(data[,i], main=names(data)[i], horizontal = TRUE)
}
```

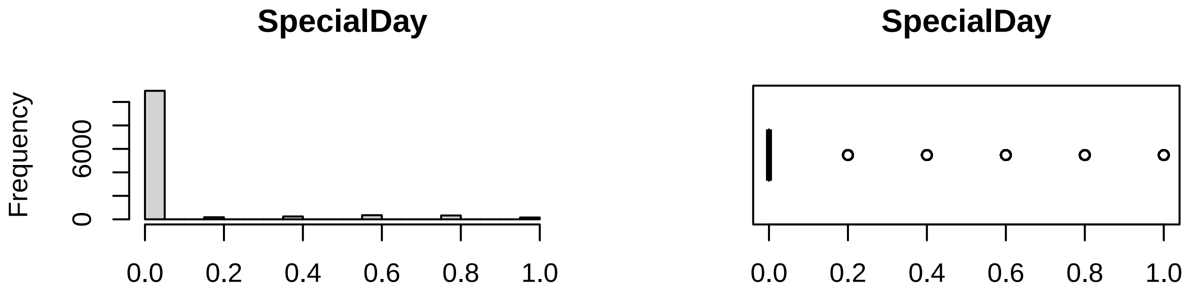



Informational
Frequency
0
6000
0
5
10
15
20
Informational
0
5
10
15
20
Informational_Duration
Frequency
0
6000
0
500
1500
2500
Informational_Duration
0
500
1000
2000


ProductRelated
Frequency
0
6000
0
200
400
600
ProductRelated
0
100
300
500
700
ProductRelated_Duration
Frequency
0
6000
0
20000
40000
60000
ProductRelated_Duration
0
20000
40000
60000


BounceRates
Frequency
0
4000
0.00
0.05
0.10
0.15
0.20
BounceRates
0.00
0.05
0.10
0.15
0.20
ExitRates
Frequency
0
1500
0.00
0.05
0.10
0.15
0.20
ExitRates
0.00
0.05
0.10
0.15
0.20



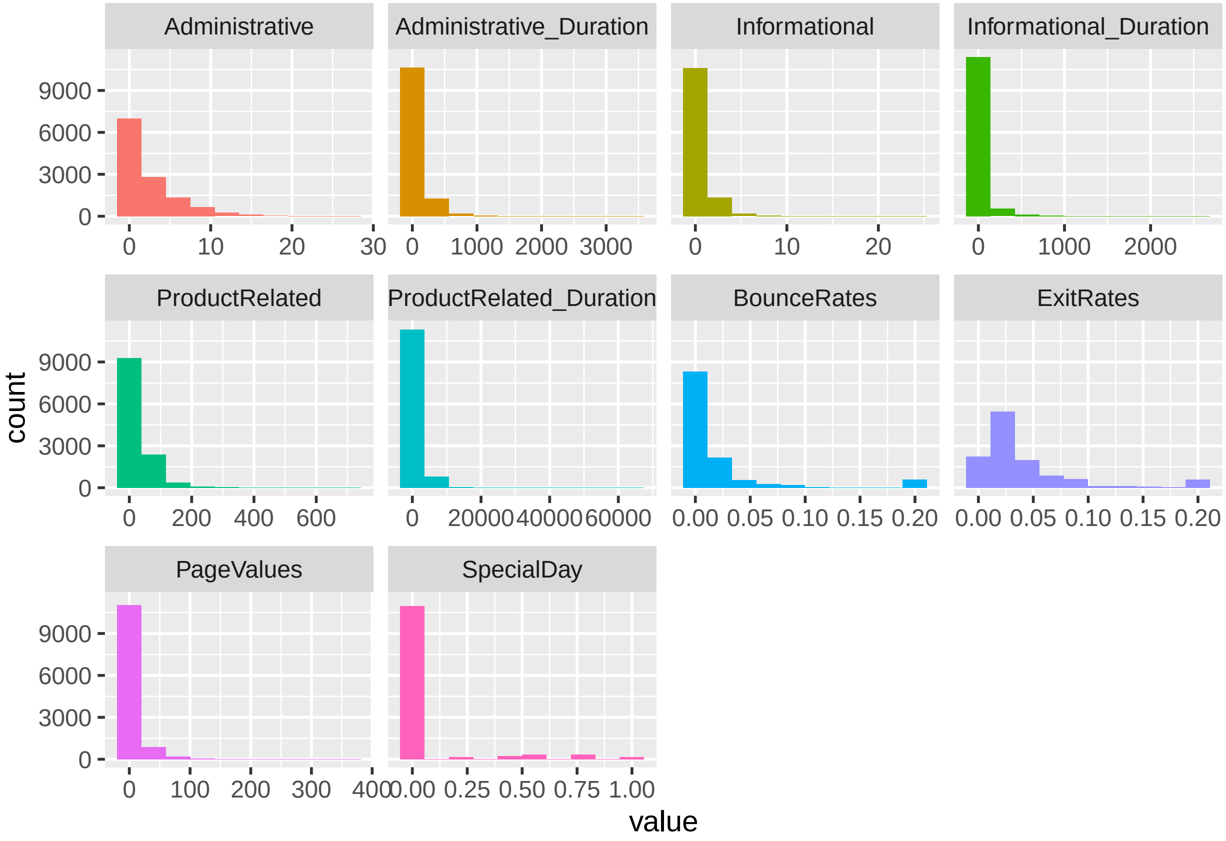


```
library(funModeling) #Package gives us colourful frequency plots
```

```
## Loading required package: Hmisc

## Loading required package: lattice

## Loading required package: survival

## Loading required package: Formula

## Loading required package: ggplot2

##
## Attaching package: 'Hmisc'

## The following objects are masked from 'package:base':
##
##     format.pval, units

## funModeling v.1.9.4 :)
## Examples and tutorials at livebook.datascienceheroes.com
##  / Now in Spanish: librovivodecienciadedatos.ai
```

```
data_num <- data[1:10]

plot_num(data_num) # Histogram of all the continuous variables
```

```
## Warning: `guides(<scale> = FALSE)` is deprecated. Please use `guides(<scale> =
## "none")` instead.
```



The numerical values are positively skewed

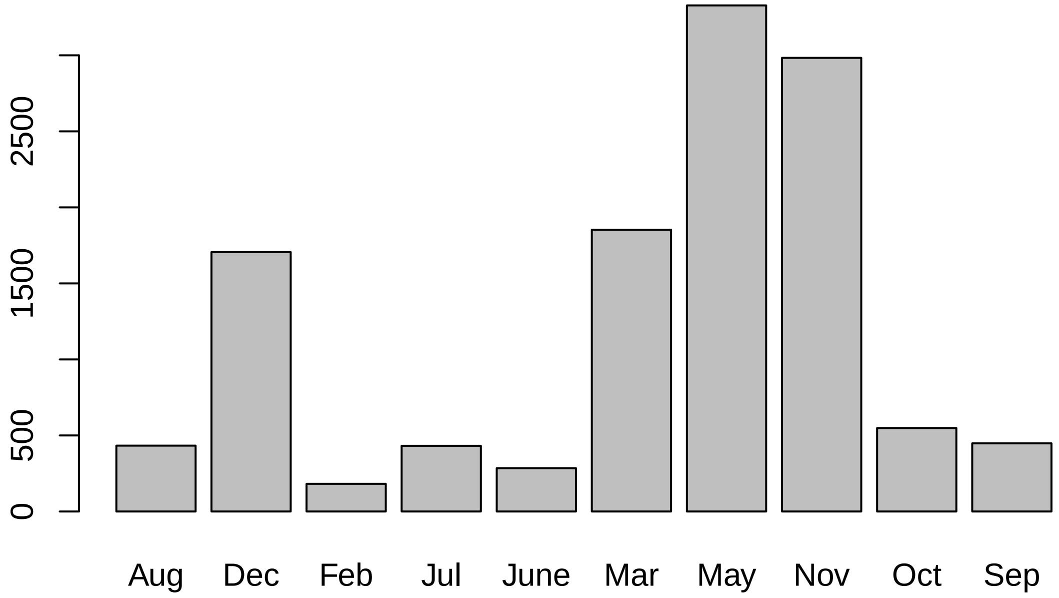
```
# create tables of all categorical variables to be able to create bar plots with them
month_table <- table(data$Month)
os_table <- table(data$OperatingSystems)
browser_table <- table(data$Browser)
region_table <- table(data$Region)
traffic_table <- table(data$TrafficType)
visitor_table <- table(data$VisitorType)
weekend_table <- table(data$Weekend)
revenue_table <- table(data$Revenue)
```

```
# function for adjusting plot size
set_plot_dimensions <- function(width_choice, height_choice) {
    options(repr.plot.width = width_choice, repr.plot.height = height_choice)
}
```

```
# barplot of Month
set_plot_dimensions(5, 4)
month_table
```

```
##
##  Aug  Dec  Feb  Jul June  Mar  May  Nov  Oct  Sep
##  433 1706  182  432  285 1853 3328 2983  549  448
```

```
barplot(month_table)
```

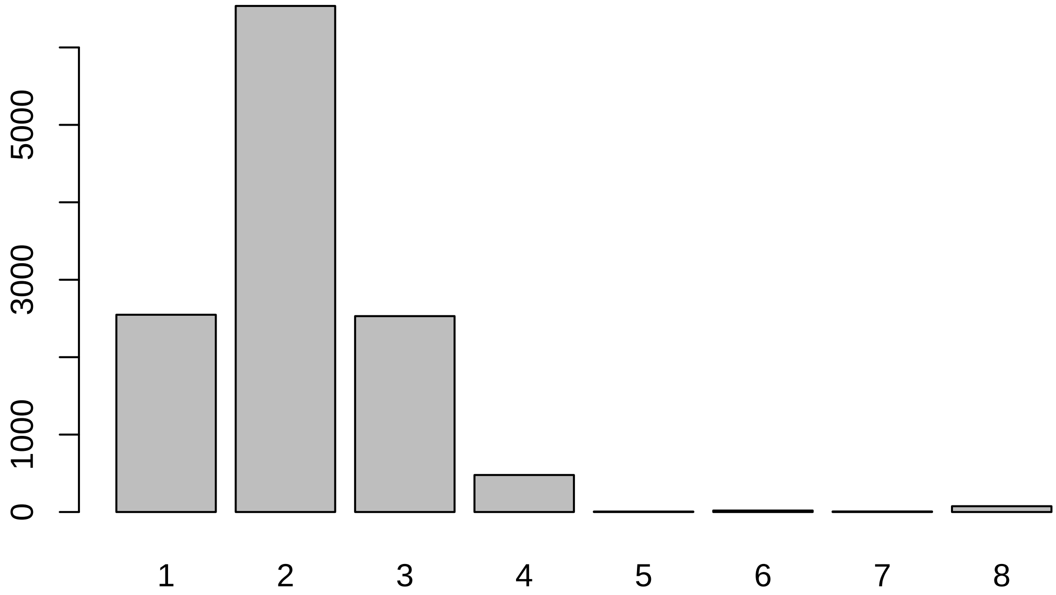


May is the most frequently occuring month while February is the least frequently occuring.

```
# barplot of Operating System
set_plot_dimensions(5, 4)
os_table
```

```
##
##    1    2    3    4    5    6    7    8
## 2548 6536 2530  478    6   19    7   75
```

```
barplot(os_table)
```

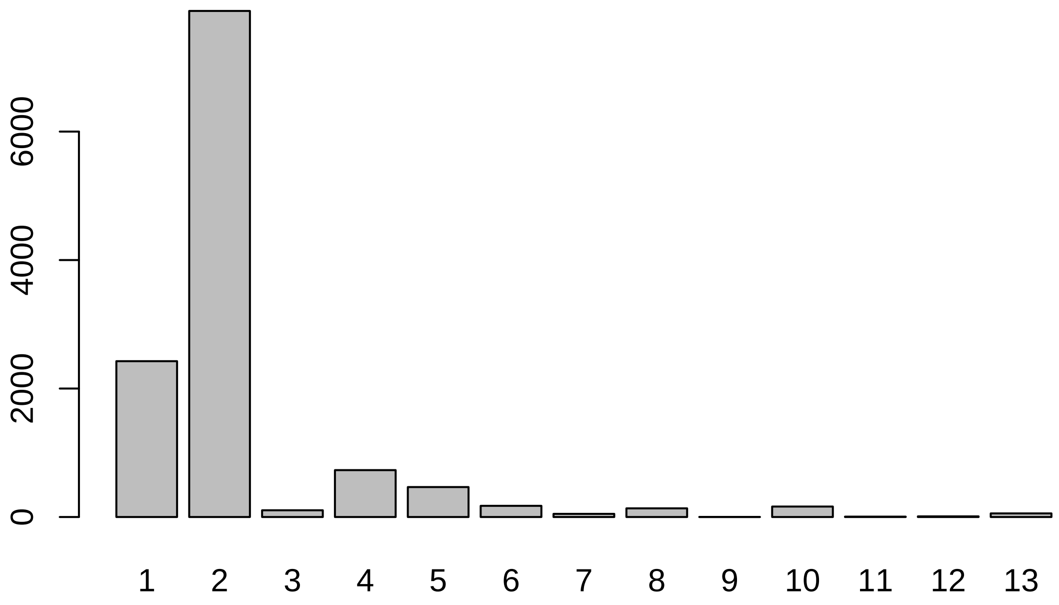


Operating System 2 is the most used Operating System , followed by OS 1 and 2 which seem to be almost the same. OS 5 is the least used operating system.

```
# barplot of Browser
set_plot_dimensions(5, 4)
browser_table
```

```
##
##    1    2    3    4    5    6    7    8    9   10   11   12   13
## 2426 7878  105  730  466  174   49  135    1  163    6   10   56
```

```
barplot(browser_table)
```

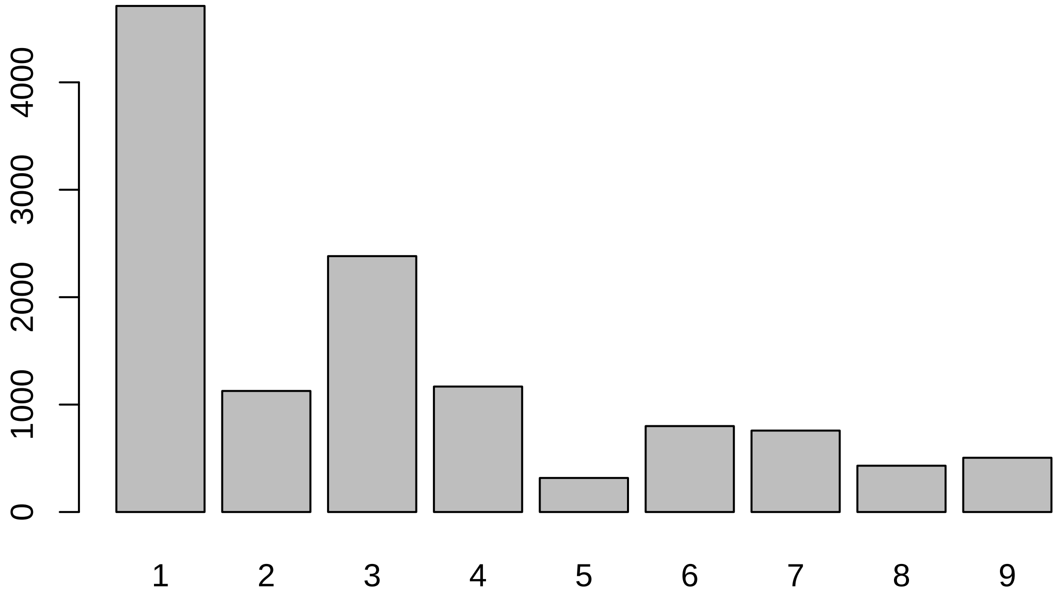


Browser 2 is the most widely used browser and it's followed by Browser 1. Browsers 9, 11, and 12 appear to be the least used browsers.

```
# barplot of Region
set_plot_dimensions(5, 4)
region_table
```

```
##
##    1    2    3    4    5    6    7    8    9
## 4711 1127 2382 1168  317  800  758  431  505
```

```
barplot(region_table)
```

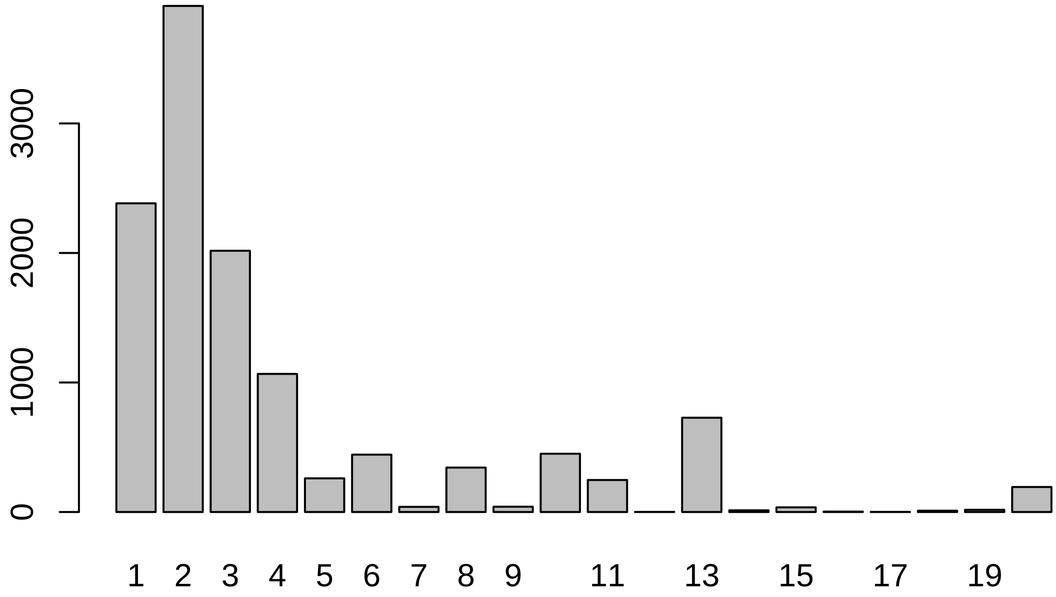


Region 1 is the most occuring region while Region 5 is the least occuring.

```
# barplot of TrafficType
set_plot_dimensions(5, 4)
traffic_table
```

```
##
##    1    2    3    4    5    6    7    8    9   10   11   12   13   14   15   16
## 2383 3907 2017 1066  260  443   40  343   41  450  247    1  728   13   36    3
##   17   18   19   20
##    1   10   17  193
```

```
barplot(traffic_table)
```

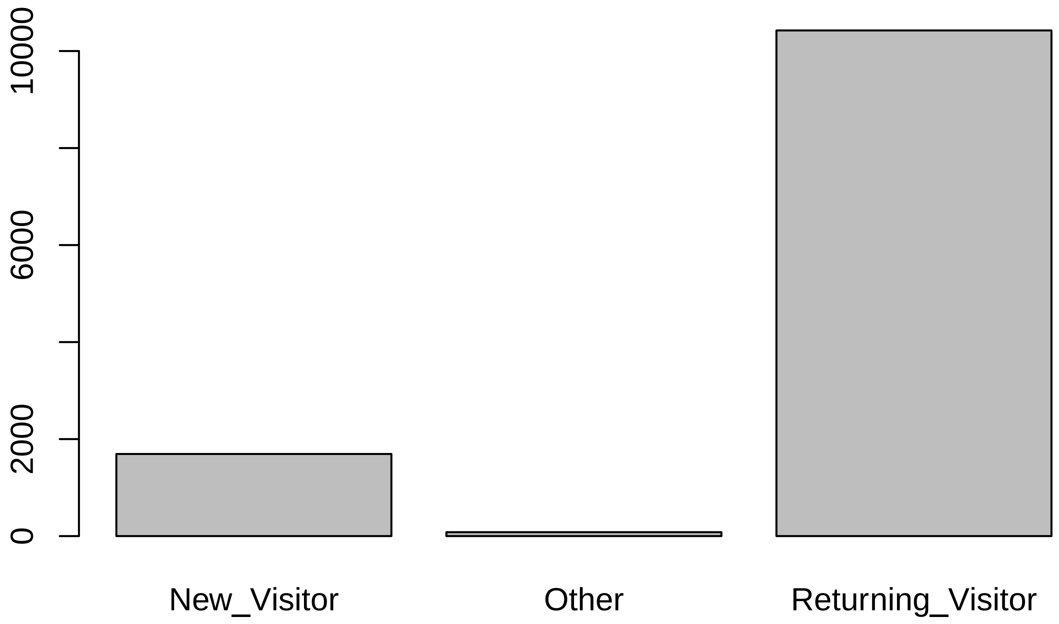


Traffic Type 2 is the highest while Types 12, 16, and 17 appear to be the least frequently occuring types in the dataset.

```
# barplot of VisitorType
set_plot_dimensions(5, 4)
visitor_table
```

```
##
##       New_Visitor             Other Returning_Visitor
##              1693                81             10425
```

```
barplot(visitor_table)
```

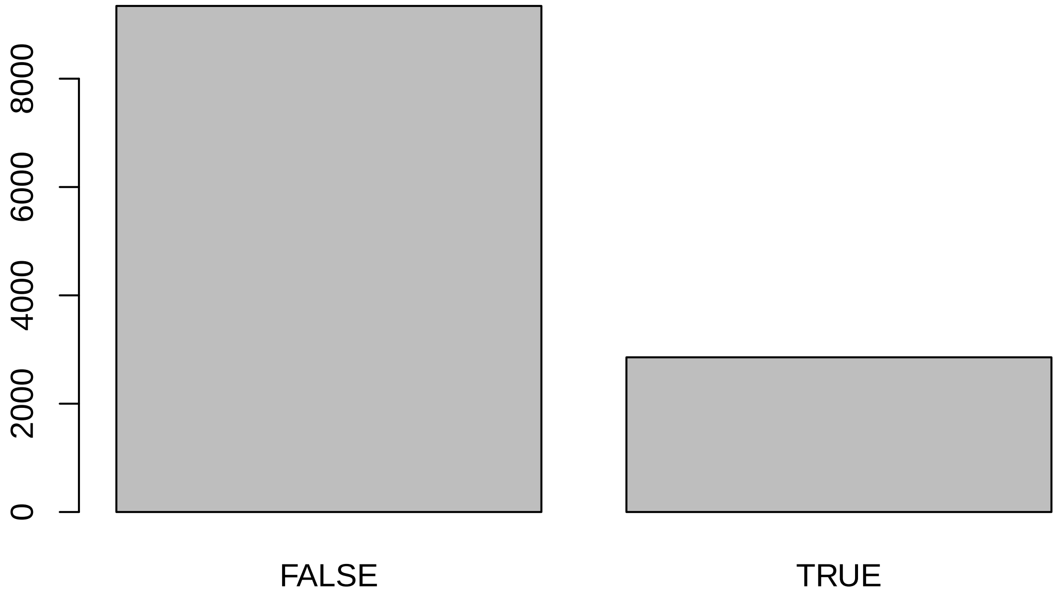


Majority of the visitors are returning

```
# barplot of Weekend
set_plot_dimensions(5, 4)
weekend_table
```

```
##
## FALSE  TRUE
##  9343  2856
```

```
barplot(weekend_table)
```

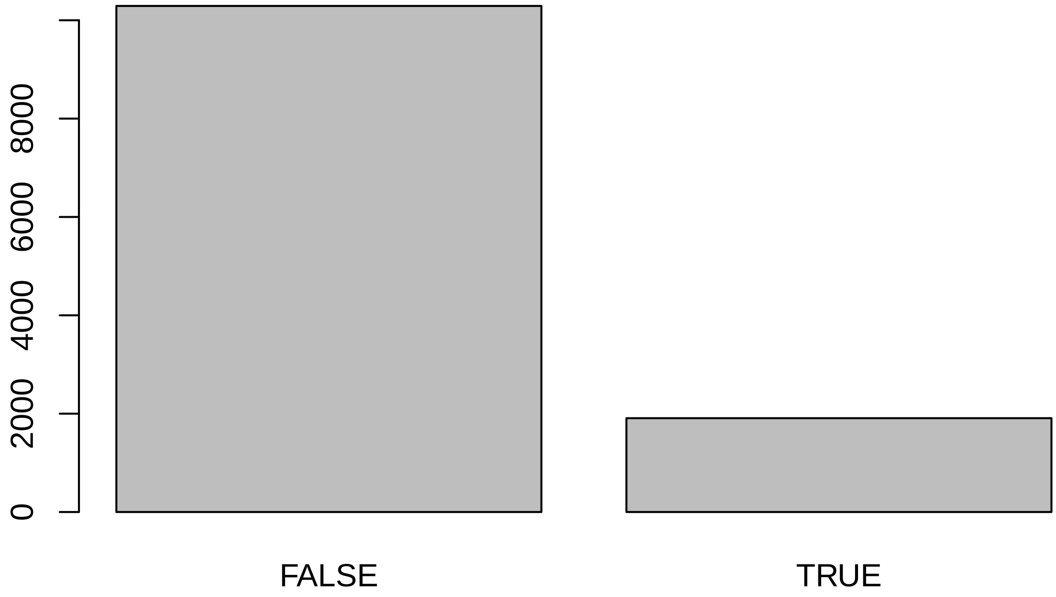


Weekday appear to be more than the weekends

```
# barplot of Revenue
set_plot_dimensions(5, 4)
revenue_table
```

```
##
## FALSE  TRUE
## 10291  1908
```

```
barplot(revenue_table)
```

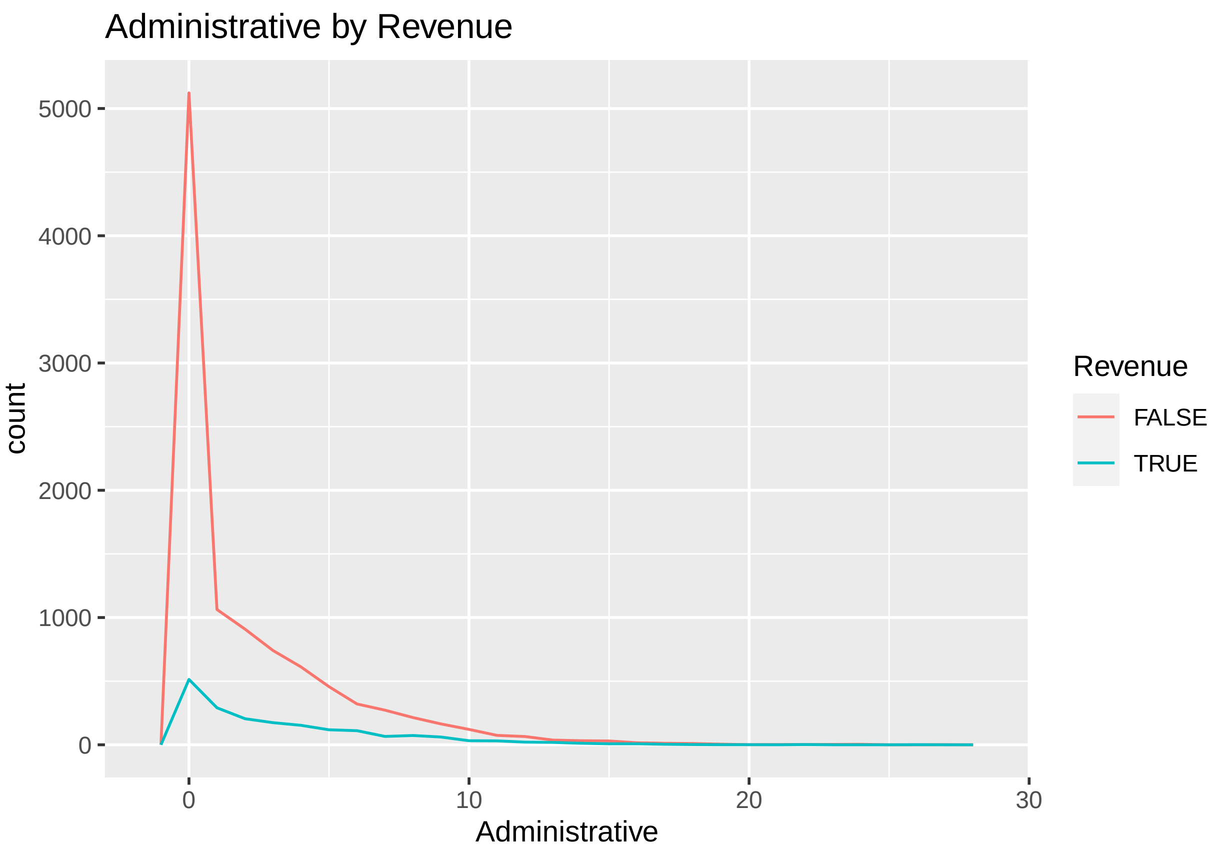


There are more FALSE revenues than true ones by a huge margin.

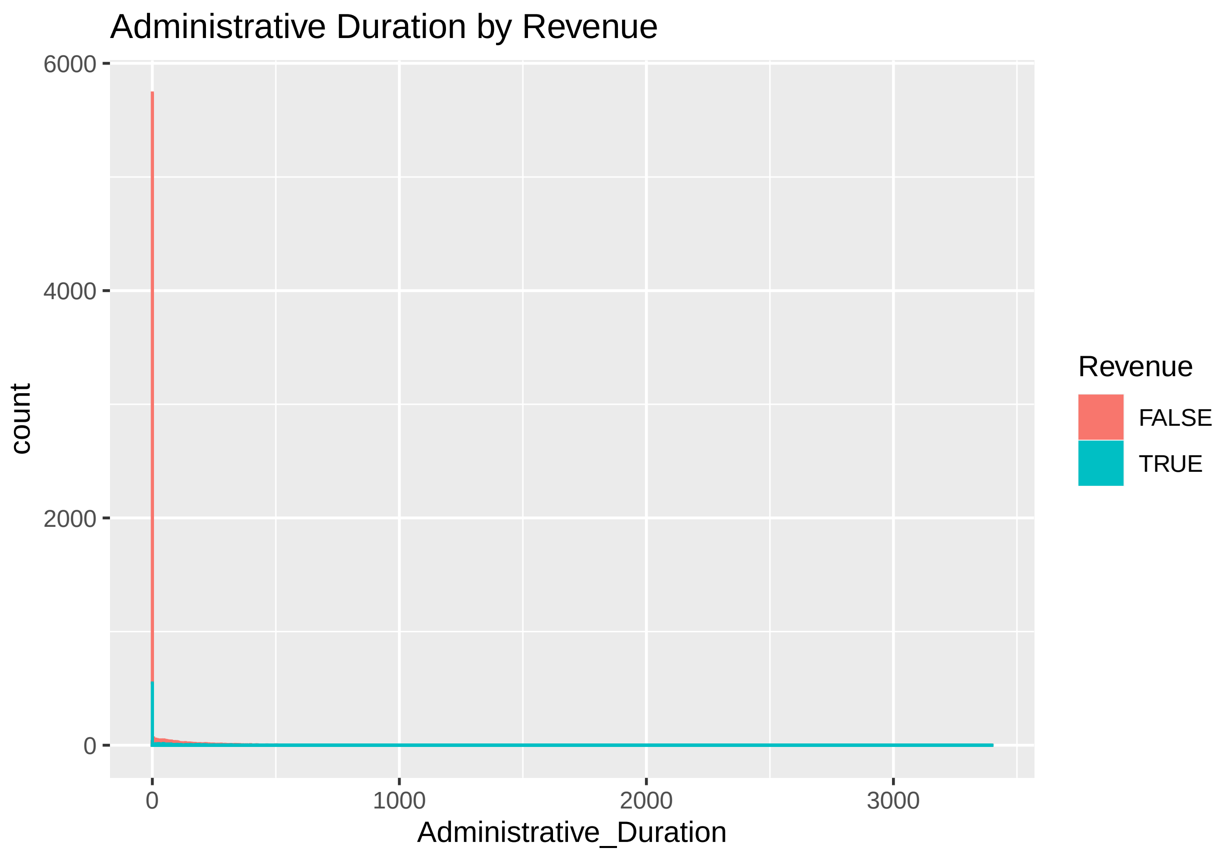
### 3.2 Bivariate Analysis

```
library(ggplot2)
```

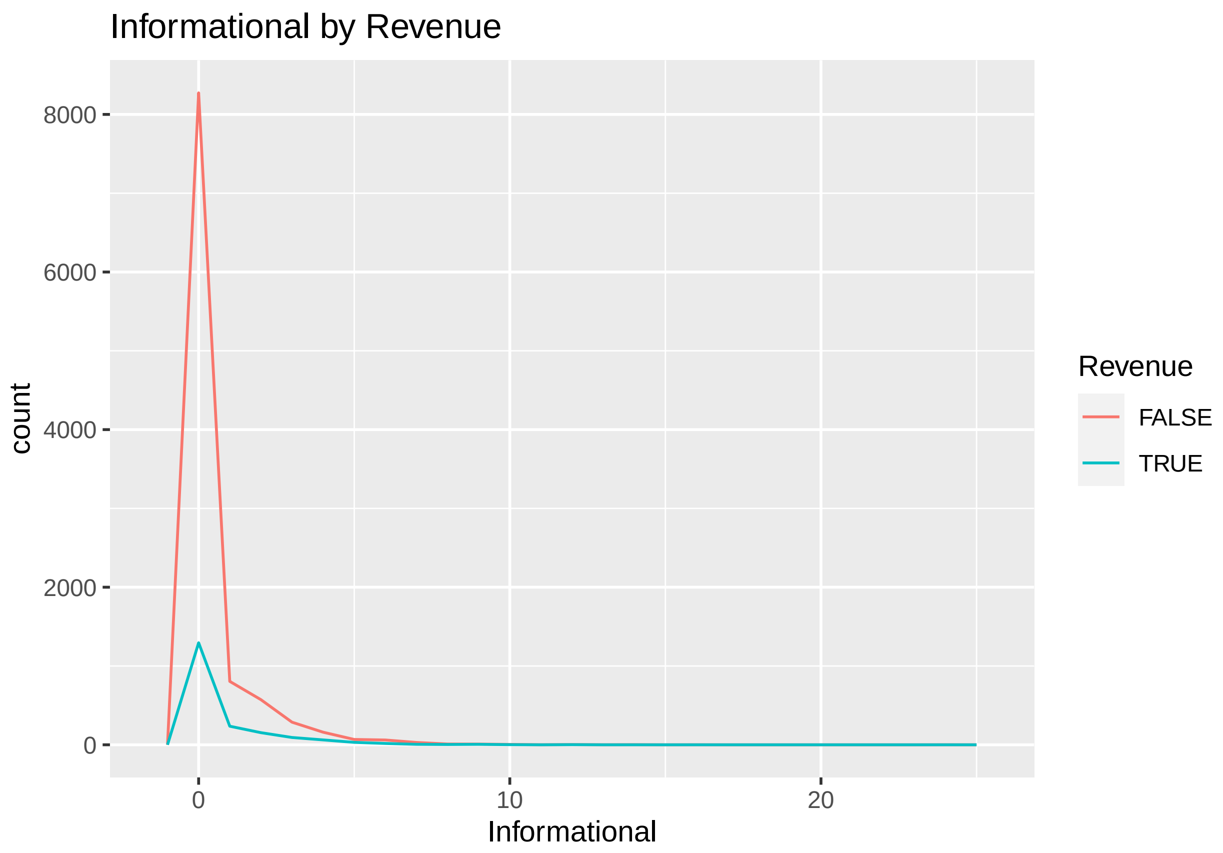
```
# Administrative by Revenue
set_plot_dimensions(5, 4)
ggplot(data, aes(x = Administrative, fill = Revenue, color = Revenue)) +
geom_freqpoly(binwidth = 1) +
labs(title = "Administrative by Revenue")
```



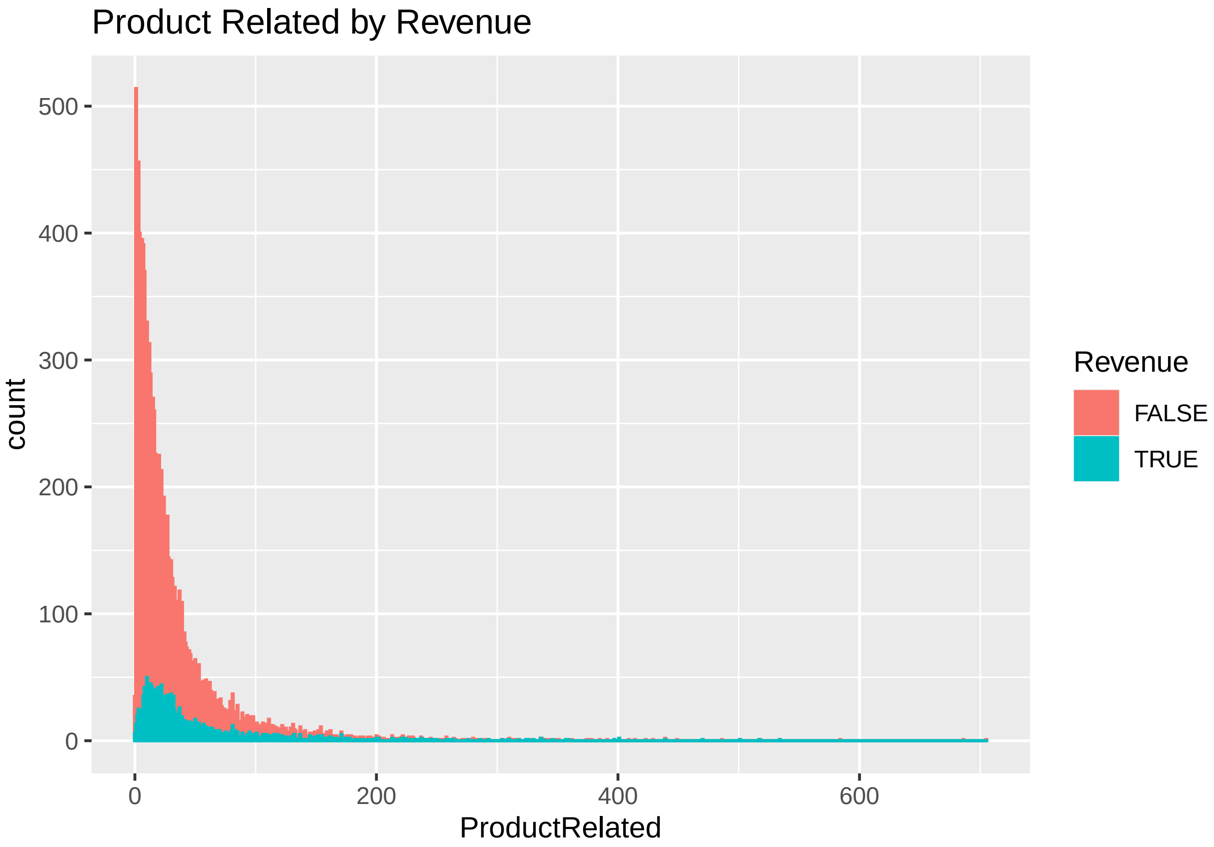
```
# Administrative Duration by Revenue
set_plot_dimensions(5, 4)
ggplot(data, aes(x = Administrative_Duration, fill = Revenue, color = Revenue)) +
geom_histogram(binwidth = 1) +
labs(title = "Administrative Duration by Revenue")
```



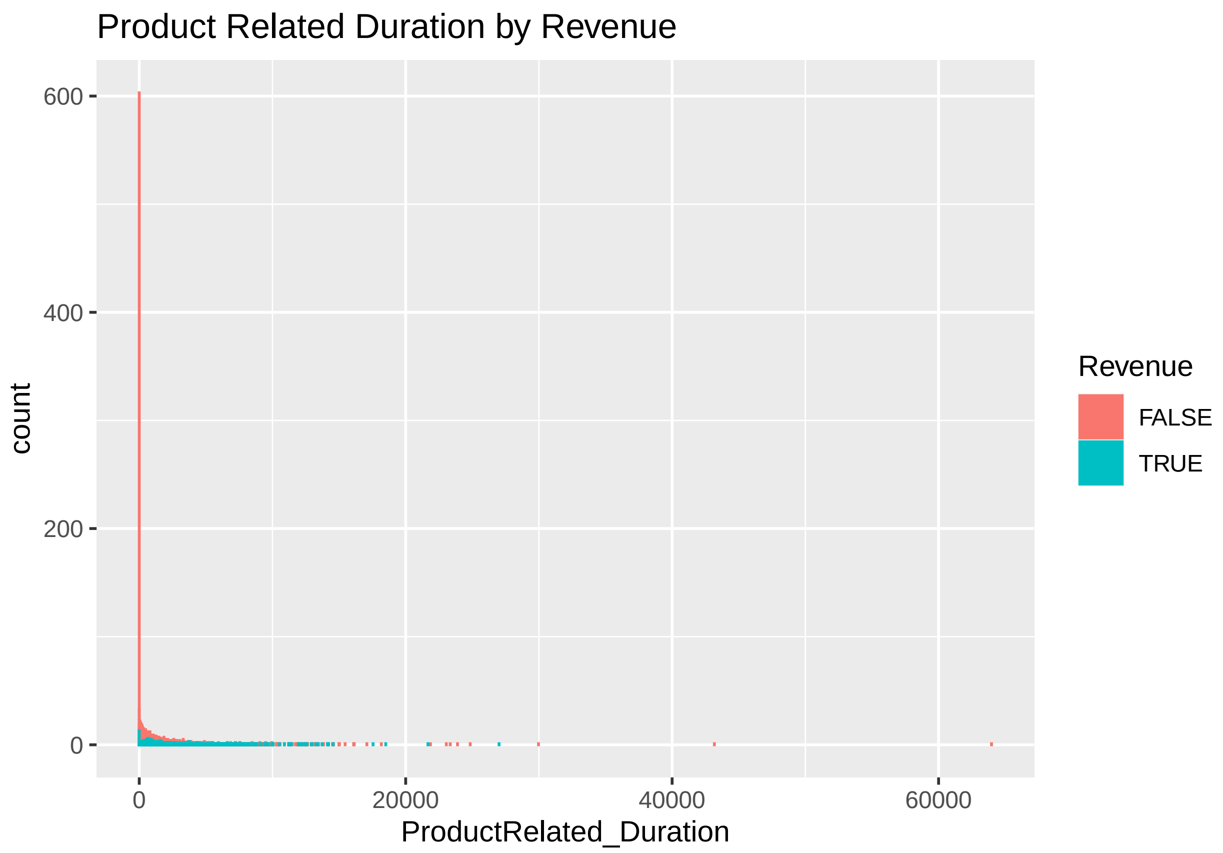
```r
# Informational by Revenue
set_plot_dimensions(5, 4)
ggplot(data, aes(x = Informational, fill = Revenue, color = Revenue)) +
geom_freqpoly(binwidth = 1) +
labs(title = "Informational by Revenue")
```



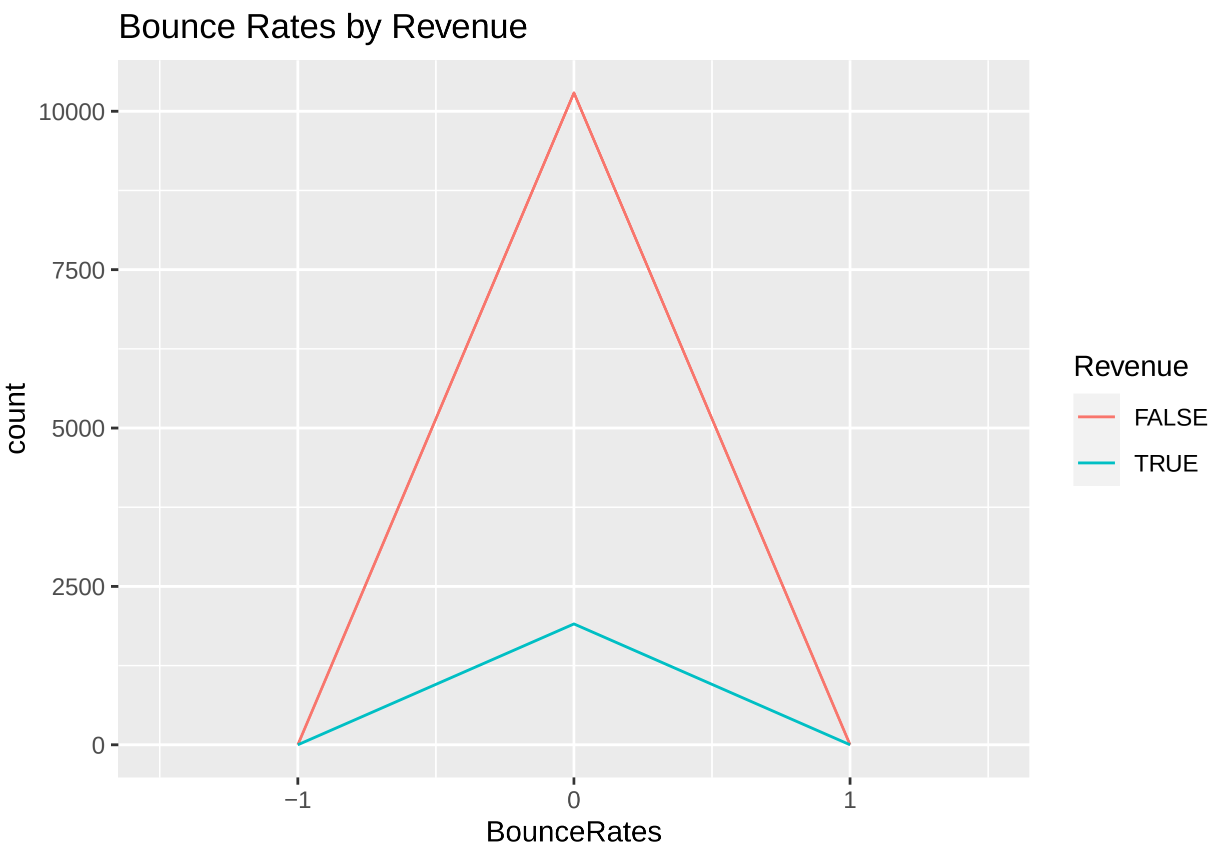
```
# product related by Revenue
set_plot_dimensions(5, 4)
ggplot(data, aes(x = ProductRelated, fill = Revenue, color = Revenue)) +
geom_histogram(binwidth = 1) +
labs(title = "Product Related by Revenue")
```



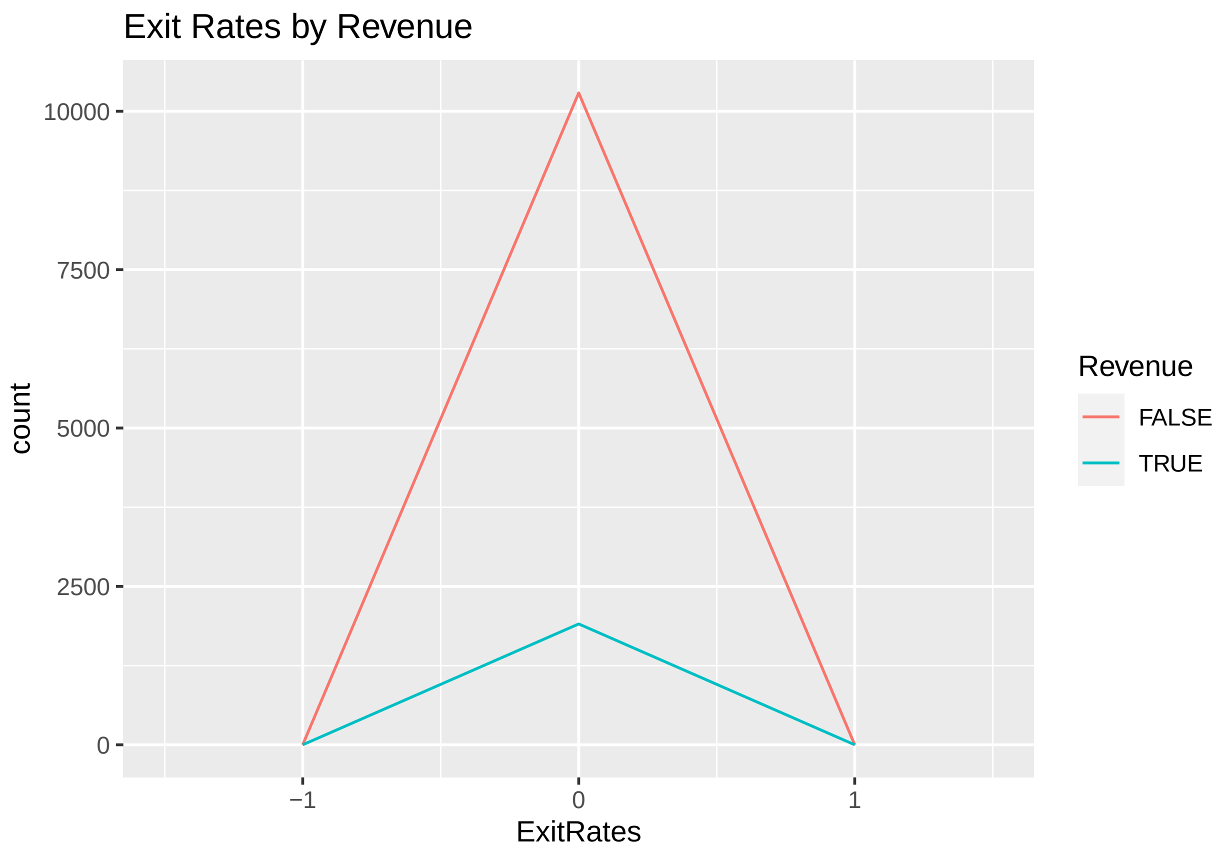
```
# product related duration by Revenue
set_plot_dimensions(5, 4)
ggplot(data, aes(x = ProductRelated_Duration, fill = Revenue, color = Revenue)) +
geom_histogram(binwidth = 1) +
labs(title = "Product Related Duration by Revenue")
```



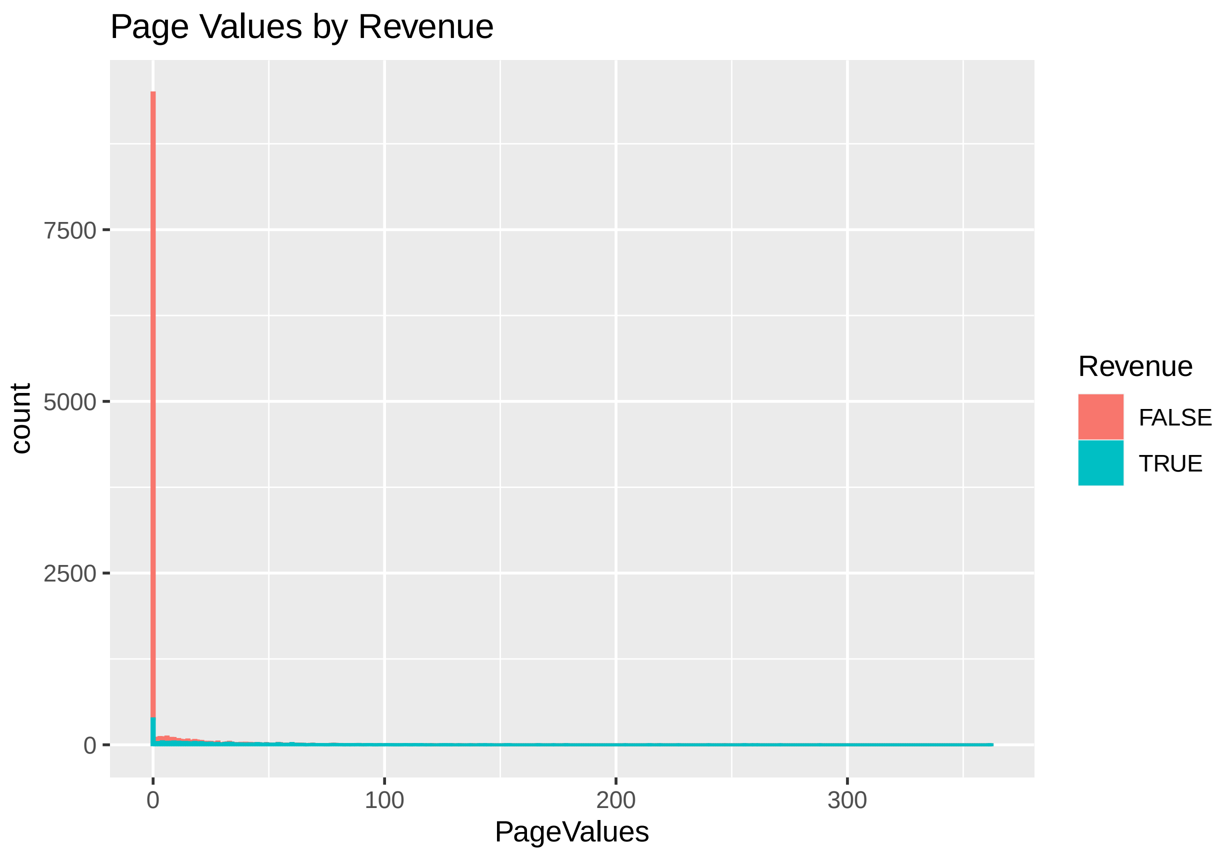
```
# bounce rates by Revenue
set_plot_dimensions(5, 4)
ggplot(data, aes(x = BounceRates, fill = Revenue, color = Revenue)) +
geom_freqpoly(binwidth = 1) +
labs(title = "Bounce Rates by Revenue")
```



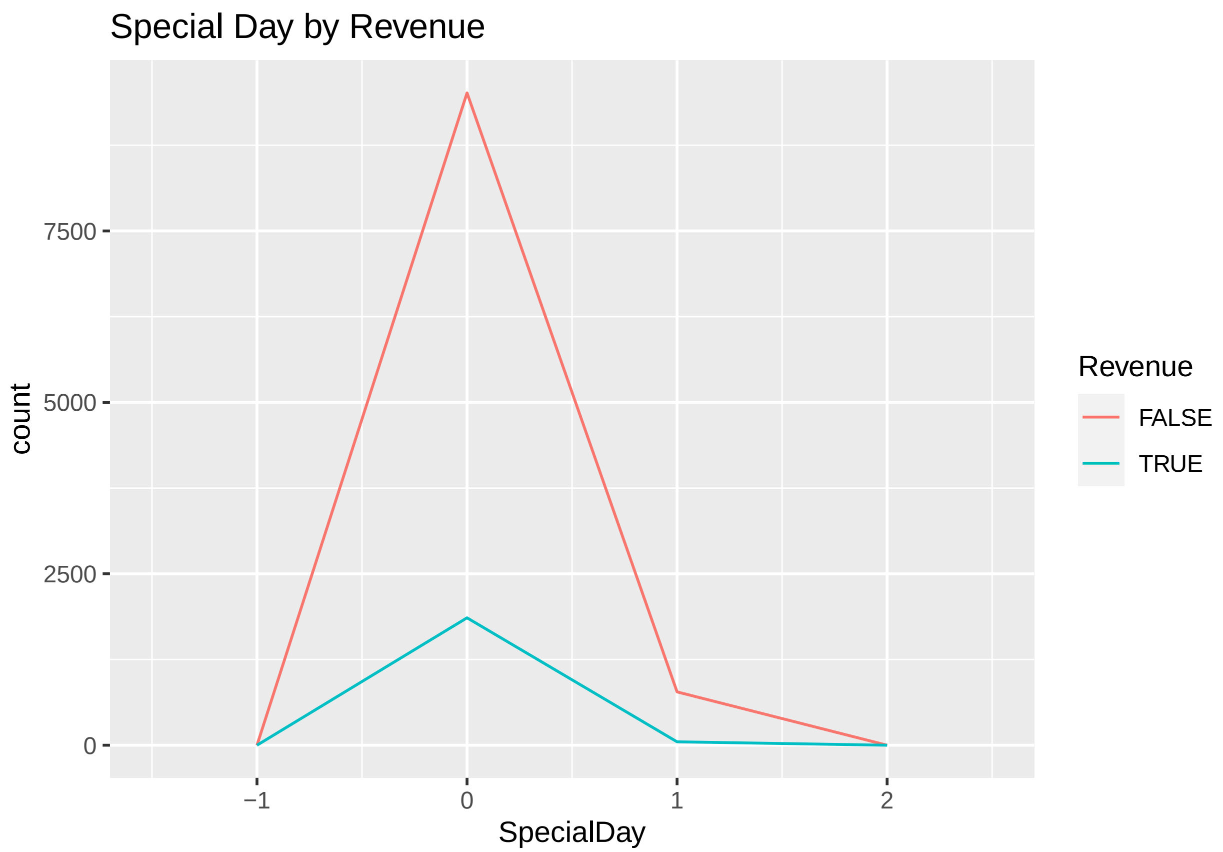
```
# exit rates by Revenue
set_plot_dimensions(5, 4)
ggplot(data, aes(x = ExitRates, fill = Revenue, color = Revenue)) +
geom_freqpoly(binwidth = 1) +
labs(title = "Exit Rates by Revenue")
```



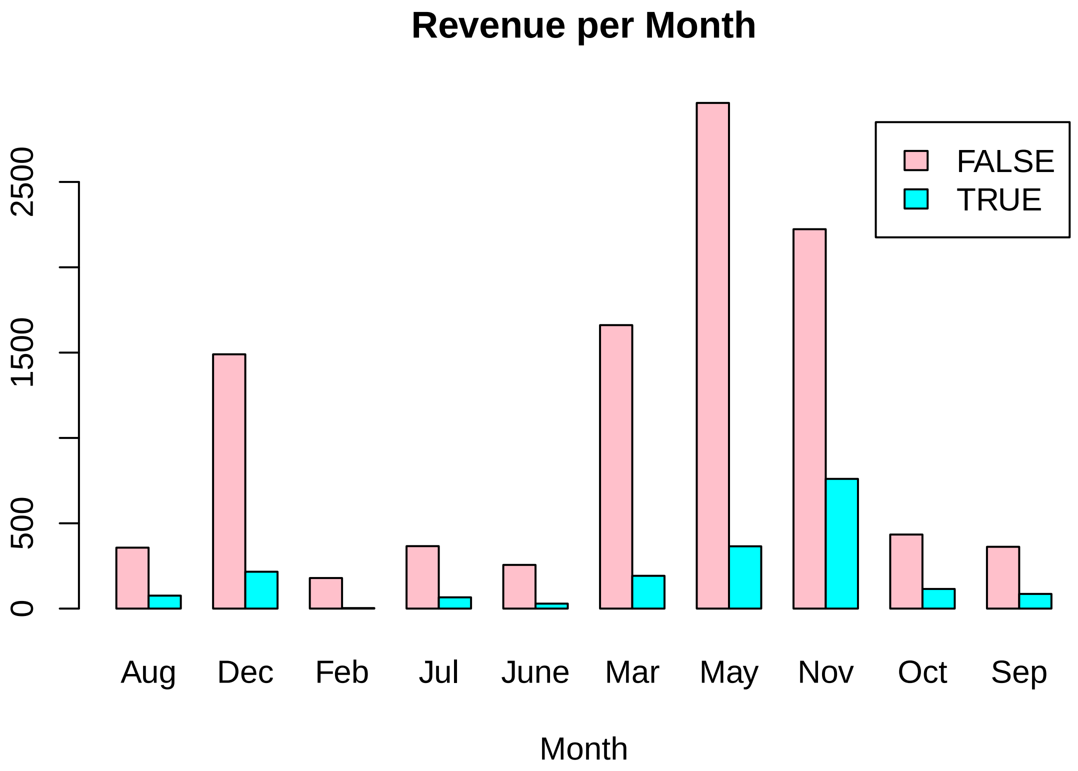
```
# Page Values by Revenue
set_plot_dimensions(5, 4)
ggplot(data, aes(x = PageValues, fill = Revenue, color = Revenue)) +
geom_histogram(binwidth = 1) +
labs(title = "Page Values by Revenue")
```



```
# special day by Revenue
set_plot_dimensions(5, 4)
ggplot(data, aes(x = SpecialDay, fill = Revenue, color = Revenue)) +
geom_freqpoly(binwidth = 1) +
labs(title = "Special Day by Revenue")
```

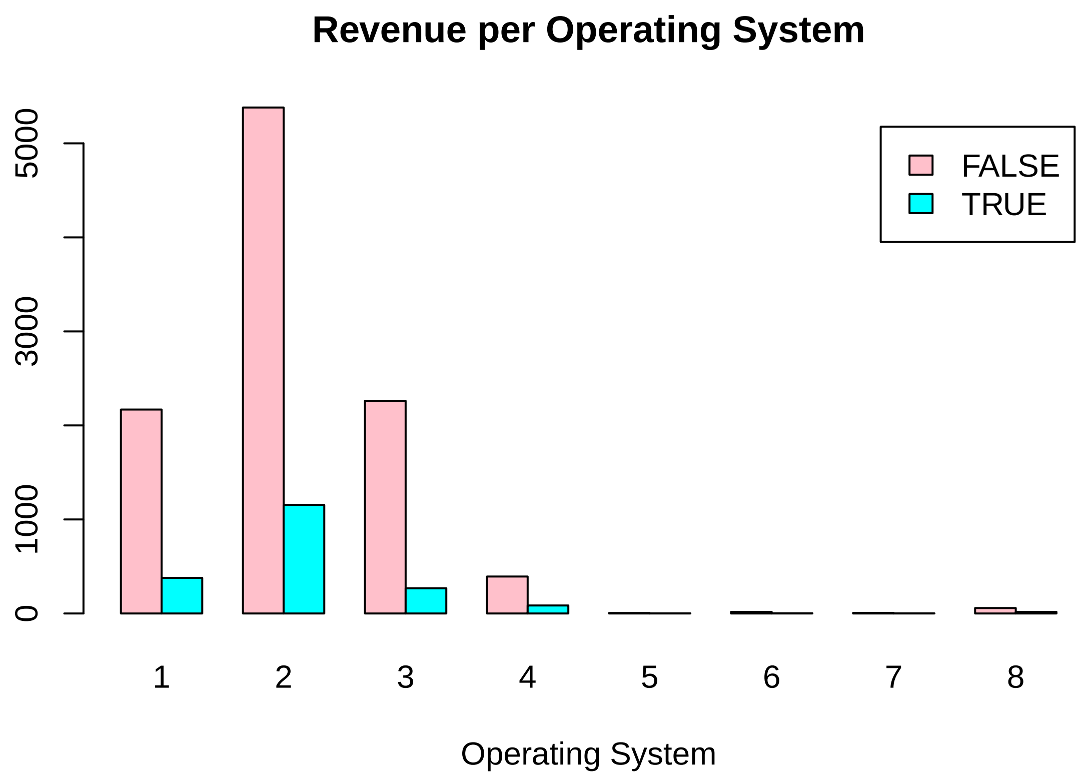


```
# plotting the distribution of Revenue per Month
set_plot_dimensions(6, 6)
rev_month <- table(data$Revenue, data$Month)
barplot(rev_month, main = "Revenue per Month", col = c("pink", "cyan"), beside = TRUE,
        legend = rownames(rev_month), xlab = "Month")
```



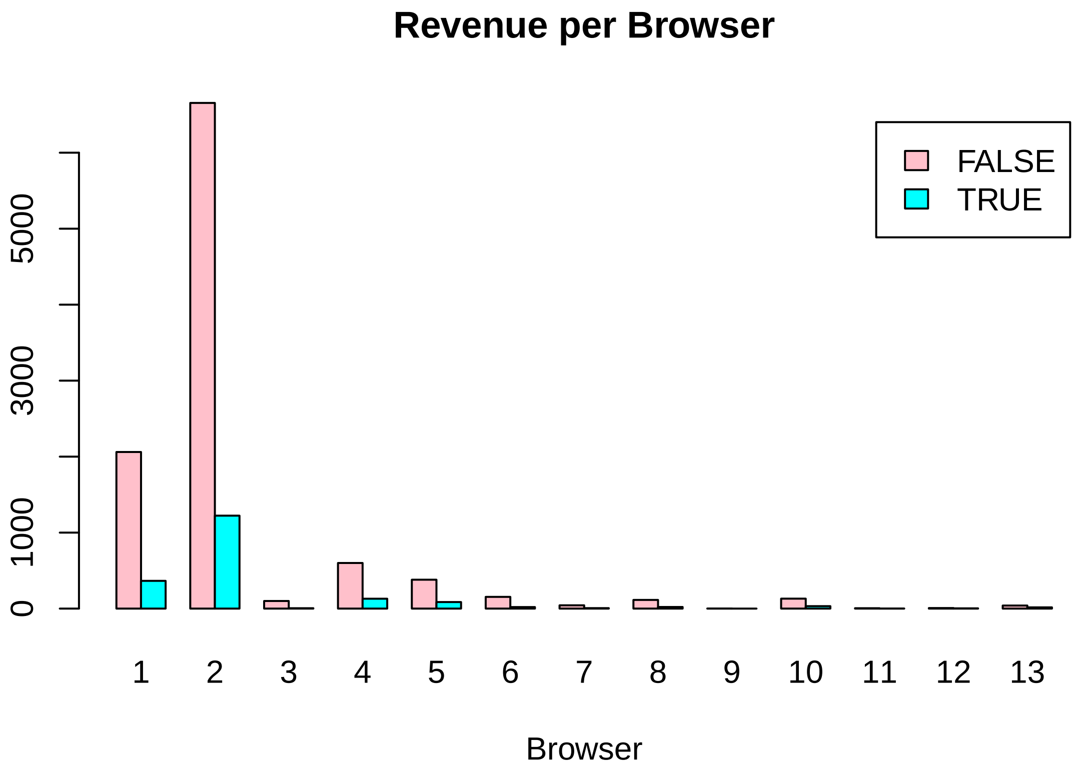
November returns the highest number of revenues while February returns the lowest.

```
# plotting the distribution of Revenue per Operating System
set_plot_dimensions(6, 6)
rev_os <- table(data$Revenue, data$OperatingSystems)
barplot(rev_os, main = "Revenue per Operating System", col = c("pink", "cyan"), beside = TRUE,
        legend = rownames(rev_os), xlab = "Operating System")
```



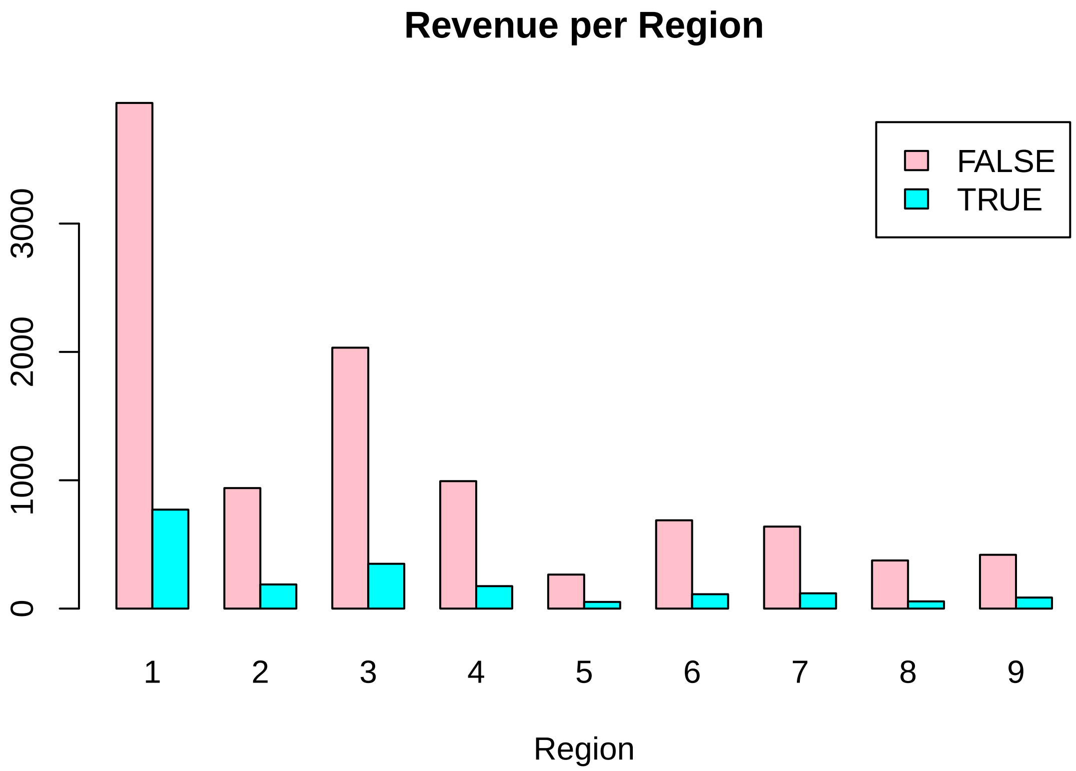
os 2 returns the highest revenue while os 5,6 and 7 return the least

```
# plotting the distribution of Revenue per Browser
set_plot_dimensions(6, 6)
rev_browser <- table(data$Revenue, data$Browser)
barplot(rev_browser, main = "Revenue per Browser", col = c("pink", "cyan"), beside = TRUE,
        legend = rownames(rev_browser), xlab = "Browser")
```



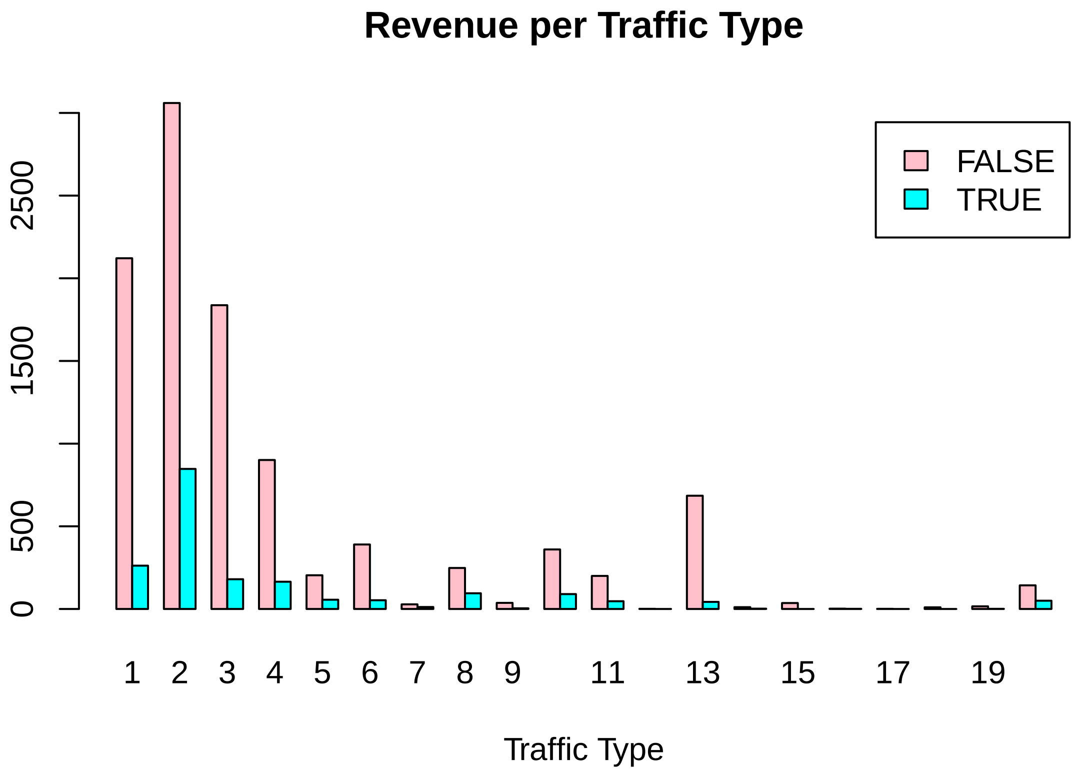
Browser 2 returns the highest revenue while browsers 9,11 and 12 return the least

```r
# plotting the distribution of Revenue per Region
set_plot_dimensions(6, 6)
rev_region <- table(data$Revenue, data$Region)
barplot(rev_region, main = "Revenue per Region", col = c("pink", "cyan"), beside = TRUE,
        legend = rownames(rev_region), xlab = "Region")
```



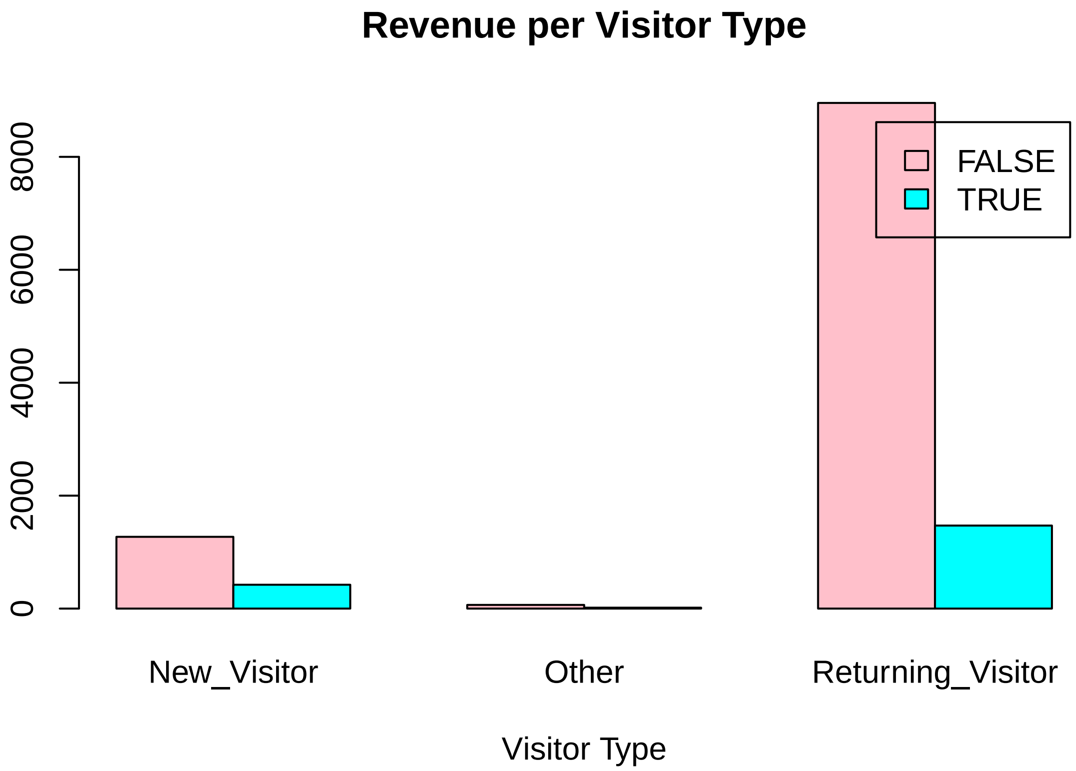
Region 1 returns the highest revenue while region 5 returns the least

```
# plotting the distribution of Revenue per Traffic Type
set_plot_dimensions(6, 6)
rev_traffic <- table(data$Revenue, data$TrafficType)
barplot(rev_traffic, main = "Revenue per Traffic Type", col = c("pink", "cyan"), beside = TRUE,
        legend = rownames(rev_traffic), xlab = "Traffic Type")
```



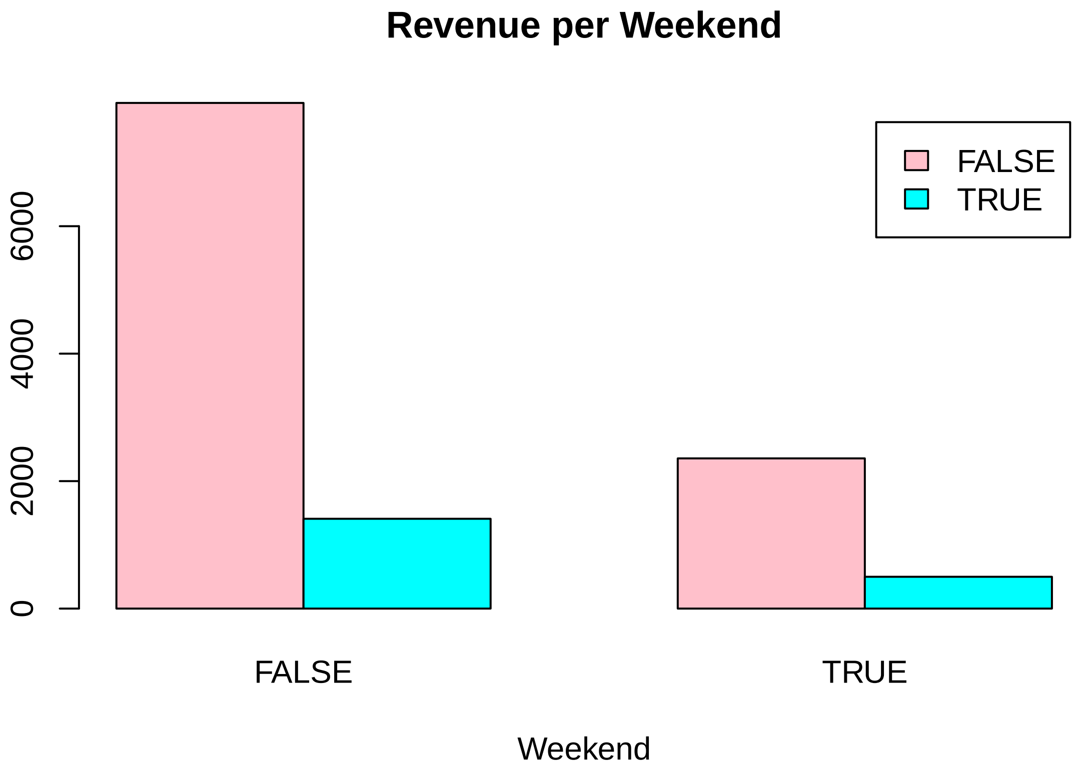
Traffic 2 has the highest number of revenues, 12, 14, 16, 17, and 18 return the lowest.

```r
# plotting the distribution of Revenue per Visitor Type
set_plot_dimensions(6, 6)
rev_visitor <- table(data$Revenue, data$VisitorType)
barplot(rev_visitor, main = "Revenue per Visitor Type", col = c("pink", "cyan"), beside = TRUE,
        legend = rownames(rev_visitor), xlab = "Visitor Type")
```



Returning visitors generated more revenue.

```
# plotting the distribution of Revenue per Weekend
set_plot_dimensions(6, 6)
rev_weekend <- table(data$Revenue, data$Weekend)
barplot(rev_weekend, main = "Revenue per Weekend", col = c("pink", "cyan"), beside = TRUE,
        legend = rownames(rev_weekend), xlab = "Weekend")
```



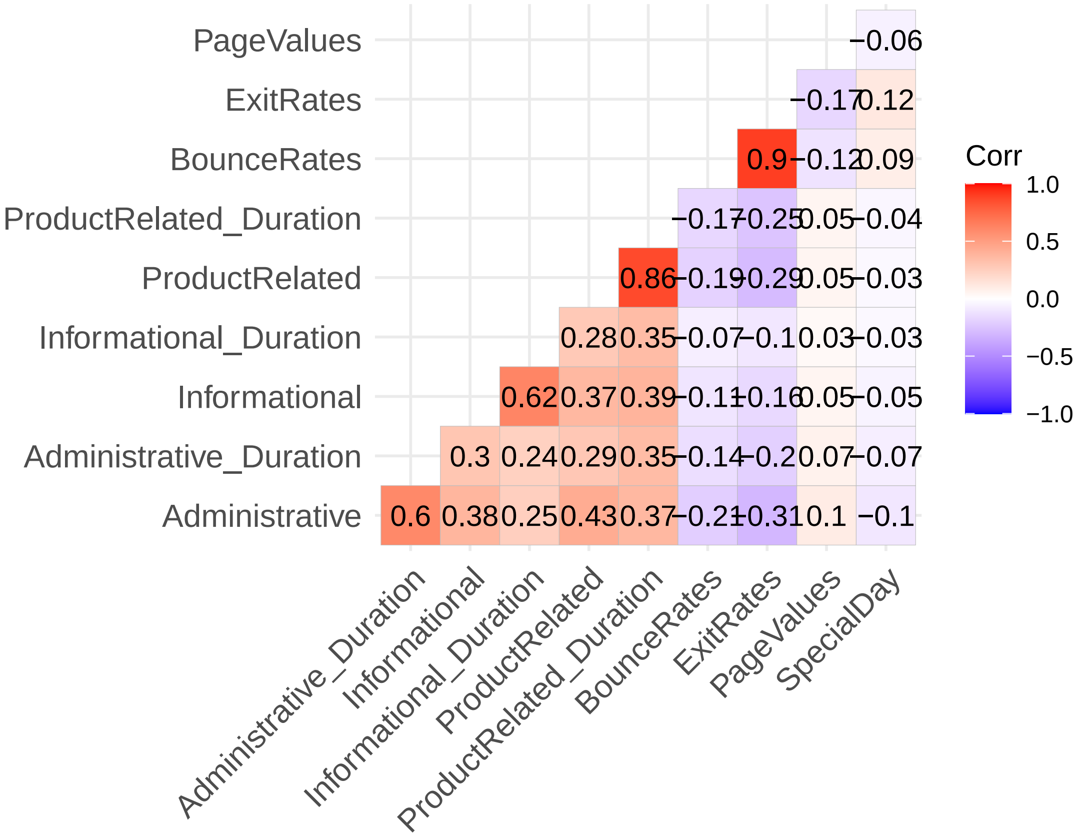
non-weekend(weekdays) returned the highest revenue

Getting correlations of the numerical variables

```
# get numerical columns
data_num <- data[,1:10]
head(data_num)
```

```
##   Administrative Administrative_Duration Informational Informational_Duration
## 1              0                       0             0                      0
## 2              0                       0             0                      0
## 3              0                      -1             0                     -1
## 4              0                       0             0                      0
## 5              0                       0             0                      0
## 6              0                       0             0                      0
##   ProductRelated ProductRelated_Duration BounceRates ExitRates PageValues
## 1              1                0.000000  0.20000000 0.2000000          0
## 2              2               64.000000  0.00000000 0.1000000          0
## 3              1               -1.000000  0.20000000 0.2000000          0
## 4              2                2.666667  0.05000000 0.1400000          0
## 5             10              627.500000  0.02000000 0.0500000          0
## 6             19              154.216667  0.01578947 0.0245614          0
##   SpecialDay
## 1          0
## 2          0
## 3          0
## 4          0
## 5          0
## 6          0
```

```
# using a heat map to visualize variable correlations
library(ggcorrplot)
set_plot_dimensions(6, 6)
corr_data <- cor(data_num)
ggcorrplot(round(corr_data, 2) ,lab = T,type = 'lower')
```



BounceRates is very highly correlated to ExitRates, Administrative is correlated with Administrative_Duration, Informational is highly correlated to Informational_Duration, and ProductRelated is highly correlated with ProductRated_Duration.To reduce redundancy and dimensionality, we will have to drop one variable of each of the highly correlated pairs

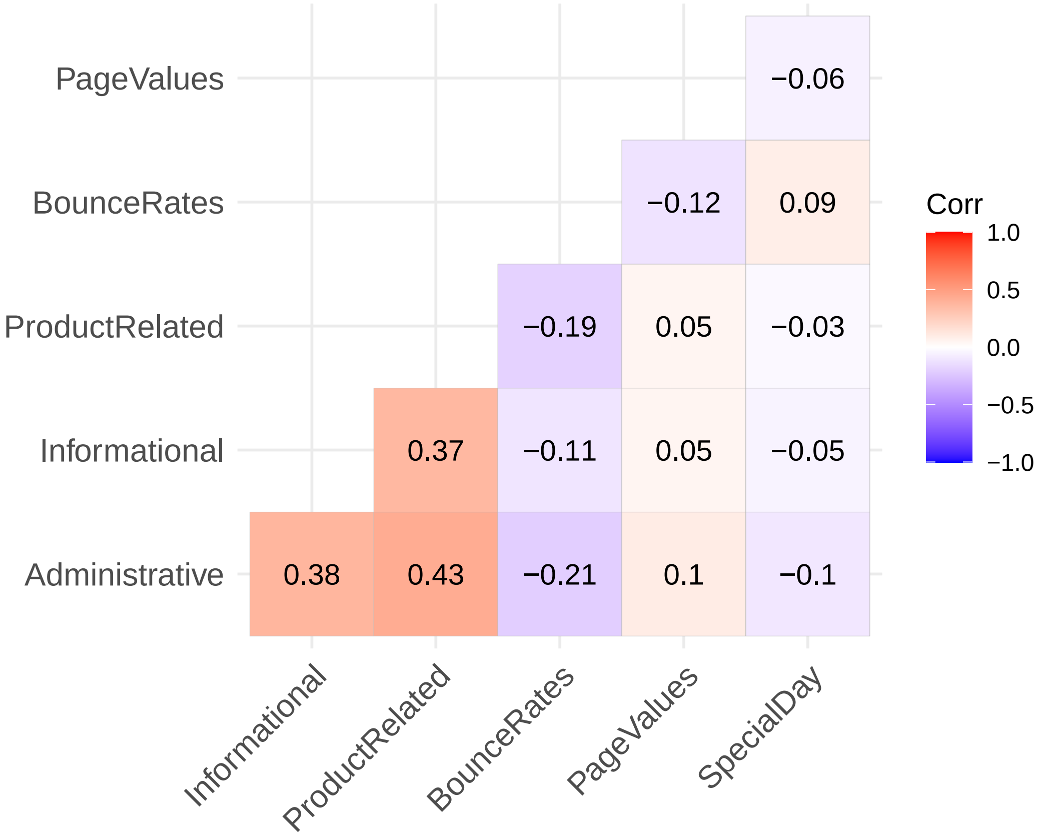
```
# dropping the highly correlated columns
to_drop <- c("Administrative_Duration", "Informational_Duration", "ProductRelated_Duration", "ExitRates

data <- data[, !names(data) %in% to_drop]
head(data)
```

```
##   Administrative Informational ProductRelated BounceRates PageValues SpecialDay
## 1              0             0              1  0.20000000          0          0
## 2              0             0              2  0.00000000          0          0
## 3              0             0              1  0.20000000          0          0
## 4              0             0              2  0.05000000          0          0
## 5              0             0             10  0.02000000          0          0
## 6              0             0             19  0.01578947          0          0
##   Month OperatingSystems Browser Region TrafficType       VisitorType Weekend
## 1   Feb                1       1      1           1 Returning_Visitor   FALSE
## 2   Feb                2       2      1           2 Returning_Visitor   FALSE
## 3   Feb                4       1      9           3 Returning_Visitor   FALSE
## 4   Feb                3       2      2           4 Returning_Visitor   FALSE
## 5   Feb                3       3      1           4 Returning_Visitor    TRUE
## 6   Feb                2       2      1           3 Returning_Visitor   FALSE
##   Revenue
## 1   FALSE
## 2   FALSE
## 3   FALSE
## 4   FALSE
## 5   FALSE
## 6   FALSE
```

```
# getting the numerical columns from the new dataframe
data_num <- data[,1:6]
head(data_num)
```

```
##   Administrative Informational ProductRelated BounceRates PageValues SpecialDay
## 1              0             0              1  0.20000000          0          0
## 2              0             0              2  0.00000000          0          0
## 3              0             0              1  0.20000000          0          0
## 4              0             0              2  0.05000000          0          0
## 5              0             0             10  0.02000000          0          0
## 6              0             0             19  0.01578947          0          0
```

```
# visualizing the correlations of the new dataset
set_plot_dimensions(6, 6)
new_corr_data <- cor(data_num)
ggcorrplot(round(new_corr_data, 2) ,lab = T,type = 'lower')
```



Removing the highly correlated variables reduced multicollinearity in our dataset and also made it easier to work with.

## 4. Modelling

### 4.1 Supervised learning

#### 4.1.1 Feature Engineering

```
library(lattice)
library(caret)
```

```
##
## Attaching package: 'caret'

## The following object is masked from 'package:survival':
##
##     cluster
```

```
# shuffling our data set to randomize the records
shuffle_index <- sample(1:nrow(data))
data <- data[shuffle_index, ]
dim(data)
```

```
## [1] 12199    14
```

```
head(data)
```

```
##       Administrative Informational ProductRelated BounceRates PageValues
## 12209              0             0             15 0.000000000    0.00000
## 1868               2             1             12 0.014285714    0.00000
## 7871               1             0             13 0.000000000   34.86423
## 4350               0             0              9 0.120000000    0.00000
## 10140              9             4             57 0.003030303   30.60956
## 7323               1             0             17 0.000000000    0.00000
##       SpecialDay Month OperatingSystems Browser Region TrafficType
## 12209        0.0   Nov                2       2      2           1
## 1868         0.0   Mar                2       2      7          10
## 7871         0.0   Nov                3       2      3           2
## 4350         0.6   May                2       2      1          13
## 10140        0.0   Nov                1       2      1           2
## 7323         0.0   Jul                3       2      1           3
##             VisitorType Weekend Revenue
## 12209 Returning_Visitor   FALSE   FALSE
## 1868  Returning_Visitor   FALSE   FALSE
## 7871  Returning_Visitor    TRUE    TRUE
## 4350  Returning_Visitor   FALSE   FALSE
## 10140 Returning_Visitor   FALSE    TRUE
## 7323  Returning_Visitor   FALSE    TRUE
```

```
# Normalizing the dataset
normalize <- function(x){
  return ((x-min(x)) / (max(x)-min(x)))
}

data$Administrative <- normalize(data$Administrative)
data$Informational <- normalize(data$Informational)
data$ProductRelated <- normalize(data$ProductRelated)
data$BounceRates <- normalize(data$BounceRates)
data$PageValues <- normalize(data$PageValues)
data$SpecialDay <- normalize(data$SpecialDay)
head(data)
```

```
##       Administrative Informational ProductRelated BounceRates PageValues
## 12209     0.00000000    0.00000000     0.02127660  0.00000000 0.00000000
## 1868      0.07407407    0.04166667     0.01702128  0.07142857 0.00000000
## 7871      0.03703704    0.00000000     0.01843972  0.00000000 0.09637293
## 4350      0.00000000    0.00000000     0.01276596  0.60000000 0.00000000
## 10140     0.33333333    0.16666667     0.08085106  0.01515151 0.08461201
## 7323      0.03703704    0.00000000     0.02411348  0.00000000 0.00000000
##       SpecialDay Month OperatingSystems Browser Region TrafficType
## 12209        0.0   Nov                2       2      2           1
## 1868         0.0   Mar                2       2      7          10
## 7871         0.0   Nov                3       2      3           2
## 4350         0.6   May                2       2      1          13
## 10140        0.0   Nov                1       2      1           2
## 7323         0.0   Jul                3       2      1           3
##              VisitorType Weekend Revenue
## 12209 Returning_Visitor   FALSE   FALSE
## 1868  Returning_Visitor   FALSE   FALSE
## 7871  Returning_Visitor    TRUE    TRUE
## 4350  Returning_Visitor   FALSE   FALSE
## 10140 Returning_Visitor   FALSE    TRUE
## 7323  Returning_Visitor   FALSE    TRUE
```

```
# splitting our data into training and testing sets
# we will split it 70:30
intrain <- createDataPartition(y = data$Revenue, p = 0.7, list = FALSE)
training <- data[intrain,]
testing <- data[-intrain,]
```

```
# checking the dimensions of our training and testing sets
dim(training)
```

```
## [1] 8540   14
```

```
dim(testing)
```

```
## [1] 3659   14
```

```
# checking the dimensions of our split
prop.table(table(data$Revenue)) * 100
```

```
## 
##    FALSE     TRUE 
## 84.35937 15.64063
```

```
prop.table(table(training$Revenue)) * 100
```

```
## 
##    FALSE     TRUE 
## 84.35597 15.64403
```

```
prop.table(table(testing$Revenue)) * 100
```

```
## 
##    FALSE     TRUE 
## 84.36731 15.63269
```

### 4.1.2 KNN

```
# splitting into train and test sets without the target variable
train <- training[, -14]
test <- testing[, -14]

# storing the training and test sets' target variable
train_rev <- training[, 14]
test_rev <- testing[, 14]
```

```r
# knn requires that all its predictor variables should be numerical in nature
train$Month <- as.numeric(train$Month)
```

```
## Warning: NAs introduced by coercion
```

```r
train$OperatingSystems <- as.numeric(train$OperatingSystems)
train$Browser <- as.numeric(train$Browser)
train$Region <- as.numeric(train$Region)
train$TrafficType <- as.numeric(train$TrafficType)
train$VisitorType <- as.numeric(train$VisitorType)
```

```
## Warning: NAs introduced by coercion
```

```r
train$Weekend <- as.numeric(train$Weekend)

test$Month <- as.numeric(test$Month)
```

```
## Warning: NAs introduced by coercion
```

```r
test$OperatingSystems <- as.numeric(test$OperatingSystems)
test$Browser <- as.numeric(test$Browser)
test$Region <- as.numeric(test$Region)
test$TrafficType <- as.numeric(test$TrafficType)
test$VisitorType <- as.numeric(test$VisitorType)
```

```
## Warning: NAs introduced by coercion
```

```r
test$Weekend <- as.numeric(test$Weekend)
```

```r
# checking the dimensions
dim(train)
```

```
## [1] 8540   13
```

```r
dim(test)
```

```
## [1] 3659   13
```

```r
length(train_rev)
```

```
## [1] 8540
```

```r
length(test_rev)
```

```
## [1] 3659
```

```
colSums(is.na(train))
```

```
##   Administrative    Informational   ProductRelated      BounceRates 
##                0                0                0                0 
##       PageValues       SpecialDay            Month OperatingSystems 
##                0                0             8540                0 
##          Browser           Region      TrafficType      VisitorType 
##                0                0                0             8540 
##          Weekend 
##                0
```

```
colSums(is.na(test))
```

```
##   Administrative    Informational   ProductRelated      BounceRates 
##                0                0                0                0 
##       PageValues       SpecialDay            Month OperatingSystems 
##                0                0             3659                0 
##          Browser           Region      TrafficType      VisitorType 
##                0                0                0             3659 
##          Weekend 
##                0
```

```
train[is.na(train)] <- 0
test[is.na(test)] <-0
```

```
# now modeling using the knn algorithm with 10 nearest neighbors and the training class
library(class)
require(class)

model <- knn(train = train, test = test,cl = train_rev, k = 20)
```

```
table(factor(model))
```

```
## 
## FALSE  TRUE 
##  3614    45
```

```
knn_table <- table(test_rev, model)
knn_table
```

```
##         model
## test_rev FALSE TRUE
##    FALSE  3078    9
##    TRUE    536   36
```

```
# calculating accuracy
knn_acc <- sum(diag(knn_table)/(sum(rowSums(knn_table)))) * 100
print(paste("KNN accuracy score:", knn_acc))
```

```
## [1] "KNN accuracy score: 85.105220005466"
```

### 4.1.4 Naive Bayes

```
# Creating objects x which holds the predictor variables and y which holds the response variables
x = training[,-14]
y = training$Revenue
```

```
# Now building our model
model = train(x,y,'nb',trControl=trainControl(method='cv',number=10))
```

```
## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 1

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 590

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 1

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 86

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 103

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 437

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 491

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 517

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 590

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 640

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 748

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 37

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 45
## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 259

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 366

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 438

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 476

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 617

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 672

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 675

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 752

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 786

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 823

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 434

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 579

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 42

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 432

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 434

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 477

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 579
## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 705

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 763

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 698

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 208

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 290

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 698

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 144

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 173

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 827

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 846

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 168

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 495

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 581

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 672

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 674

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 843

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 40
## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 220

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 393

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 439

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 496

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 843

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 55

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 56

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 210

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 238

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 363

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 381

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 406

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 486

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 683

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 689

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 705

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 817
## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 853

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 81

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 253

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 267

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 331

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 573

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 651

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 653

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 656

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 687

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 785

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 788

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 15

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 342

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 345

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 358

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 418
## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 567

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 632
```

```
# making predictions
Predict <- predict(model,newdata = testing )
```

```
## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 3558
```

```
# using confusion matrix to check accuracy
confusionMatrix(Predict, testing$Revenue )
```

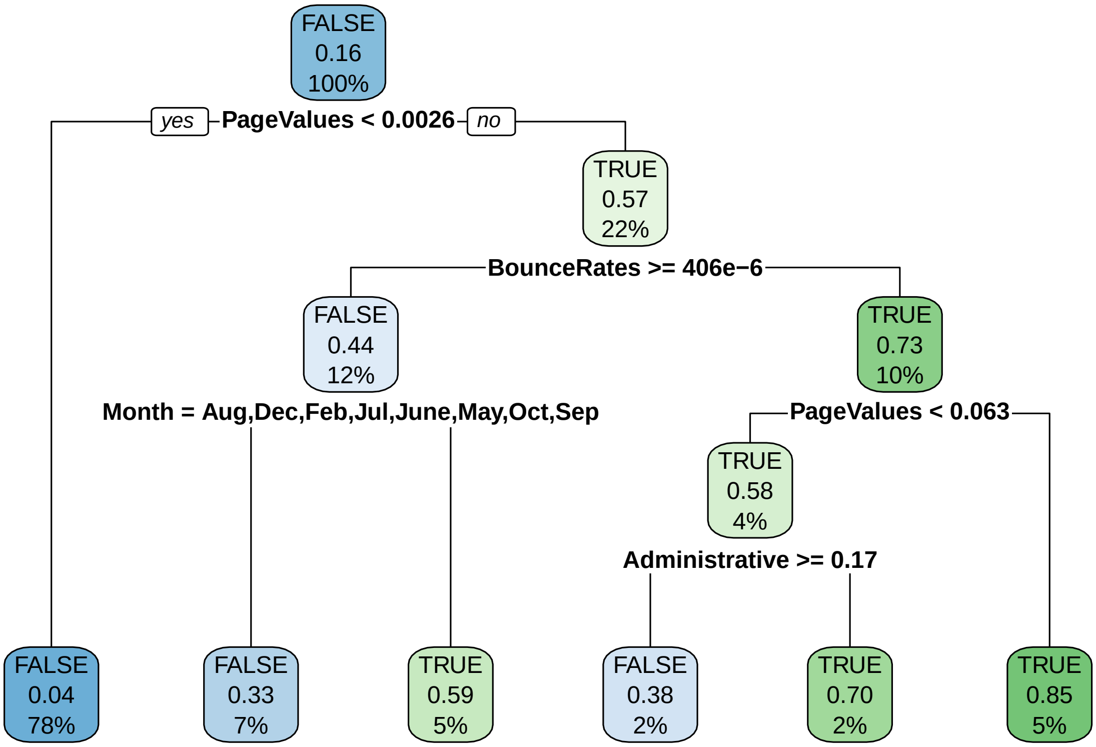
```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction FALSE TRUE
##      FALSE  2836  188
##      TRUE    251  384
##
##                Accuracy : 0.88
##                  95% CI : (0.8691, 0.8904)
##     No Information Rate : 0.8437
##     P-Value [Acc > NIR] : 2.207e-10
##
##                   Kappa : 0.5647
##
##  Mcnemar's Test P-Value : 0.003085
##
##             Sensitivity : 0.9187
##             Specificity : 0.6713
##          Pos Pred Value : 0.9378
##          Neg Pred Value : 0.6047
##              Prevalence : 0.8437
##          Detection Rate : 0.7751
##    Detection Prevalence : 0.8265
##       Balanced Accuracy : 0.7950
##
##        'Positive' Class : FALSE
##
```

Our model has 87.6% accuracy.

### 4.1.5 Decision Trees

```
library(rpart)
library(rpart.plot)
```

```
# fitting and training the model using the decision tree classifier
fit <- rpart(Revenue ~ ., data = training, method = 'class')
rpart.plot(fit, extra = 106)
```



```
# making predictions
predict_unseen <- predict(fit, testing, type = 'class')
```

```
# comparing predicted values to actual results
table_mat <- table(testing$Revenue, predict_unseen)
table_mat
```

```
##        predict_unseen
##         FALSE TRUE
##   FALSE  2954  133
##   TRUE    213  359
```

```
# calculating the accuracy
accuracy_Test <- sum(diag(table_mat)) / sum(table_mat)
print(paste('Accuracy:', accuracy_Test))
```

```
## [1] "Accuracy: 0.905438644438371"
```

The accuracy for this model is 90.8%.

### 4.2 Unsupervised Learning

Unsupervised learning requires data that has no labels. So we will create a new dataset that does not have the “Revenue” column.

```
data_new <- data[, -14]
data_new.class <- data[, "Revenue"]
head(data_new)
```

```
##       Administrative Informational ProductRelated BounceRates PageValues
## 12209     0.00000000    0.00000000     0.02127660  0.00000000 0.00000000
## 1868      0.07407407    0.04166667     0.01702128  0.07142857 0.00000000
## 7871      0.03703704    0.00000000     0.01843972  0.00000000 0.09637293
## 4350      0.00000000    0.00000000     0.01276596  0.60000000 0.00000000
## 10140     0.33333333    0.16666667     0.08085106  0.01515151 0.08461201
## 7323      0.03703704    0.00000000     0.02411348  0.00000000 0.00000000
##       SpecialDay Month OperatingSystems Browser Region TrafficType
## 12209        0.0   Nov                2       2      2           1
## 1868         0.0   Mar                2       2      7          10
## 7871         0.0   Nov                3       2      3           2
## 4350         0.6   May                2       2      1          13
## 10140        0.0   Nov                1       2      1           2
## 7323         0.0   Jul                3       2      1           3
##             VisitorType Weekend
## 12209 Returning_Visitor   FALSE
## 1868  Returning_Visitor   FALSE
## 7871  Returning_Visitor    TRUE
## 4350  Returning_Visitor   FALSE
## 10140 Returning_Visitor   FALSE
## 7323  Returning_Visitor   FALSE
```

```
# previewing our target class
head(data_new.class)
```

```
## [1] FALSE FALSE TRUE  FALSE TRUE  TRUE
## Levels: FALSE TRUE
```

```
# convert the factors into numerics
data_new$Month <- as.numeric(as.character(data_new$Month))
```

```
## Warning: NAs introduced by coercion
```

```
data_new$OperatingSystems <- as.numeric(as.character(data_new$OperatingSystems))
data_new$Browser <- as.numeric(as.character(data_new$Browser))
data_new$Region <- as.numeric(as.character(data_new$Region))
data_new$TrafficType <- as.numeric(as.character(data_new$TrafficType))
data_new$VisitorType <- as.numeric(as.character(data_new$VisitorType))
```

```
## Warning: NAs introduced by coercion
```

```
data_new$Weekend <- as.numeric(as.character(data_new$Weekend))
```

```
## Warning: NAs introduced by coercion
```

```
str(data_new)
```

```
## 'data.frame':    12199 obs. of  13 variables:
##  $ Administrative  : num  0 0.0741 0.037 0 0.3333 ...
##  $ Informational   : num  0 0.0417 0 0 0.1667 ...
##  $ ProductRelated  : num  0.0213 0.017 0.0184 0.0128 0.0809 ...
##  $ BounceRates     : num  0 0.0714 0 0.6 0.0152 ...
##  $ PageValues      : num  0 0 0.0964 0 0.0846 ...
##  $ SpecialDay      : num  0 0 0 0.6 0 0 0 0.8 0 0 ...
##  $ Month           : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ OperatingSystems: num  2 2 3 2 1 3 2 2 2 2 ...
##  $ Browser         : num  2 2 2 2 2 2 2 5 2 2 ...
##  $ Region          : num  2 7 3 1 1 1 9 1 4 4 ...
##  $ TrafficType     : num  1 10 2 13 2 3 3 3 1 1 ...
##  $ VisitorType     : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ Weekend         : num  NA NA NA NA NA NA NA NA NA NA ...
```

```
# checking for missing values
anyNA(data_new)
```

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

```
#deal with missing values
data_new[is.na(data_new)] <- 0
```

```
# checking for missing values
anyNA(data_new)
```

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

We'll have to scale data before performing k-means clustering

```
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:Hmisc':
##
##     src, summarize
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
rescale_data <- scale(data_new)
```

```
# previewing our rescaled dataset
head(rescale_data)
```

```
##       Administrative Informational ProductRelated BounceRates PageValues
## 12209     -0.7025315    -0.3988128     -0.3824686  -0.4503438 -0.3190356
## 1868      -0.1020844     0.3849986     -0.4497318  -0.1356978 -0.3190356
## 7871      -0.4023079    -0.3988128     -0.4273107  -0.4503438  1.5495795
## 4350      -0.7025315    -0.3988128     -0.5169950   2.1926823 -0.3190356
## 10140      1.9994805     2.7364325      0.5592162  -0.3836007  1.3215421
## 7323      -0.4023079    -0.3988128     -0.3376265  -0.4503438 -0.3190356
##       SpecialDay Month OperatingSystems    Browser      Region TrafficType
## 12209 -0.3103105   NaN       -0.1371074 -0.2093703 -0.48005021  -0.7656224
## 1868  -0.3103105   NaN       -0.1371074 -0.2093703  1.60116826   1.4755179
## 7871  -0.3103105   NaN        0.9654459 -0.2093703 -0.06380652  -0.5166068
## 4350   2.6940369   NaN       -0.1371074 -0.2093703 -0.89629390   2.2225647
## 10140 -0.3103105   NaN       -1.2396607 -0.2093703 -0.89629390  -0.5166068
## 7323  -0.3103105   NaN        0.9654459 -0.2093703 -0.89629390  -0.2675912
##       VisitorType Weekend
## 12209         NaN     NaN
## 1868          NaN     NaN
## 7871          NaN     NaN
## 4350          NaN     NaN
## 10140         NaN     NaN
## 7323          NaN     NaN
```

```
#replace missing data with 0
rescale_data[is.na(rescale_data)] <- 0
```

```
# previewing our rescaled dataset
head(rescale_data)
```

```
##       Administrative Informational ProductRelated BounceRates PageValues
## 12209     -0.7025315    -0.3988128     -0.3824686  -0.4503438 -0.3190356
## 1868      -0.1020844     0.3849986     -0.4497318  -0.1356978 -0.3190356
## 7871      -0.4023079    -0.3988128     -0.4273107  -0.4503438  1.5495795
## 4350      -0.7025315    -0.3988128     -0.5169950   2.1926823 -0.3190356
## 10140      1.9994805     2.7364325      0.5592162  -0.3836007  1.3215421
## 7323      -0.4023079    -0.3988128     -0.3376265  -0.4503438 -0.3190356
##       SpecialDay Month OperatingSystems    Browser      Region TrafficType
## 12209 -0.3103105     0       -0.1371074 -0.2093703 -0.48005021  -0.7656224
## 1868  -0.3103105     0       -0.1371074 -0.2093703  1.60116826   1.4755179
## 7871  -0.3103105     0        0.9654459 -0.2093703 -0.06380652  -0.5166068
## 4350   2.6940369     0       -0.1371074 -0.2093703 -0.89629390   2.2225647
## 10140 -0.3103105     0       -1.2396607 -0.2093703 -0.89629390  -0.5166068
## 7323  -0.3103105     0        0.9654459 -0.2093703 -0.89629390  -0.2675912
##       VisitorType Weekend
## 12209           0       0
## 1868            0       0
## 7871            0       0
## 4350            0       0
## 10140           0       0
## 7323            0       0
```

### 4.2.1 K-Means Clustering

```r
# applying k-means with k = 3
k_result <- kmeans(rescale_data, 3)
```

```r
# previewing the number of records in each cluster
k_result$size
```

```
## [1] 8750 1779 1670
```

```r
# previewing the cluster centers
k_result$centers
```

```
##   Administrative Informational ProductRelated BounceRates  PageValues
## 1     -0.2566223    -0.2620265     -0.2242399 -0.01412512 -0.04106002
## 2      1.5558112     1.5278915      1.3484151 -0.31538012  0.27373369
## 3     -0.3127802    -0.2547229     -0.2615157  0.40997367 -0.07646530
##    SpecialDay Month OperatingSystems     Browser       Region TrafficType
## 1  0.01553244     0      -0.11619041 -0.02169767 -0.003582001  -0.3593713
## 2 -0.16676382     0      -0.01687395 -0.09665693 -0.100306808  -0.1701686
## 3  0.09626586     0       0.62675736  0.21665104  0.125621750   2.0642087
##   VisitorType Weekend
## 1           0       0
## 2           0       0
## 3           0       0
```

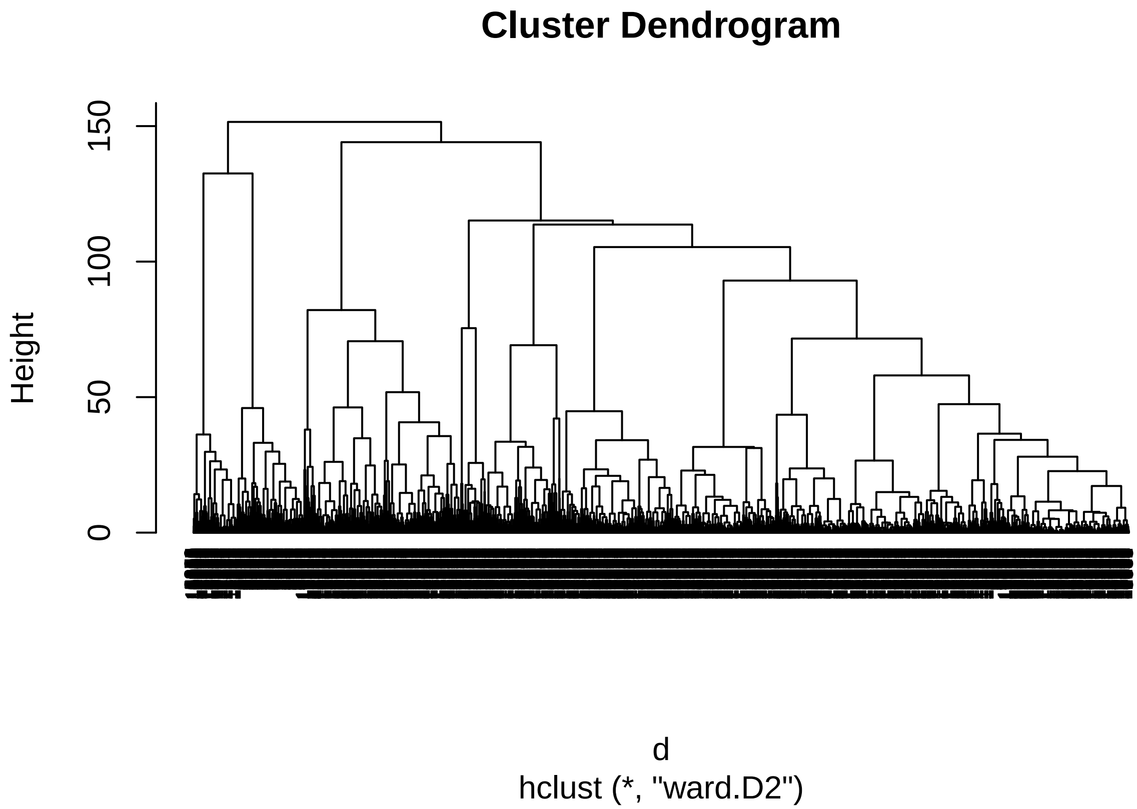
### 4.2.2 Hierarchical Clustering

As with K-means, we will use the rescaled dataset for hierarchical clustering.

```r
# first we compute the euclidean distance
d <- dist(rescale_data, method = "euclidean")
```

```r
# then we compute hierarchical clustering using the Ward method
hier <- hclust(d, method = "ward.D2" )
```

```r
# finally, we plot the dendogram
plot(hier, cex = 0.6, hang = -1)
```



## Conclusion

- Most of the revenue was from region 1.

*Traffic 2 has the highest number of revenues

- Returning visitors generated more revenue.
- More revenue was generated during the weekdays than the weekends.
- November returns the highest number of revenues