E-Commerce Data Analysis

Ramin Barfinezhadfeli, Jack McCullers, Andrew Zazueta

Load libraries and setting work directory

```
library("tidyverse")
library("gmodels")
library("funModeling")
library("GGally")
library("e1071")
library("class")
library("psych")
setwd("C:/Users/mzazu/OneDrive/Documents/USD papers/500B")
```

1. Data Importing and Pre-processing

The first code cell of this section showcases loading in the CSV file and taking a look at its contents before manipulating any of the data. The CSV file contains information on e-commerce transactions taken throughout a year. Each row represents a single session, and the columns describe what kind of sites the user was on and how long they were on, and also if a transaction was made or not.

```
# imported data by loading the tidyverse library and used the function "read_csv()" to make
# a tibble called "commerce" which holds all the data from the CSV file.
commerce <- read_csv("online_shoppers_intention.csv")
```

```
##
## -- Column specification --------------------------------------------------------
## cols(
##   Administrative = col_double(),
##   Administrative_Duration = col_double(),
##   Informational = col_double(),
##   Informational_Duration = col_double(),
##   ProductRelated = col_double(),
##   ProductRelated_Duration = col_double(),
##   BounceRates = col_double(),
##   ExitRates = col_double(),
##   PageValues = col_double(),
##   SpecialDay = col_double(),
##   Month = col_character(),
##   OperatingSystems = col_double(),
##   Browser = col_double(),
##   Region = col_double(),
##   TrafficType = col_double(),
```

```
##   VisitorType = col_character(),
##   Weekend = col_logical(),
##   Revenue = col_logical()
## )
```

```
# dimensions: 12330 rows and 18 columns
dim(commerce)
```

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

```
# data types
glimpse(commerce)
```

```
## Rows: 12,330
## Columns: 18
## $ Administrative          <dbl> 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0...
## $ Administrative_Duration <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0...
## $ Informational           <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0...
## $ Informational_Duration  <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0...
## $ ProductRelated          <dbl> 1, 2, 1, 2, 10, 19, 1, 0, 2, 3, 3, 16, 7, 6...
## $ ProductRelated_Duration <dbl> 0.000000, 64.000000, 0.000000, 2.666667, 62...
## $ BounceRates             <dbl> 0.200000000, 0.000000000, 0.200000000, 0.05...
## $ ExitRates               <dbl> 0.200000000, 0.100000000, 0.200000000, 0.14...
## $ PageValues              <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0...
## $ SpecialDay              <dbl> 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.4, 0.0, 0.8...
## $ Month                   <chr> "Feb", "Feb", "Feb", "Feb", "Feb", "Feb", "...
## $ OperatingSystems        <dbl> 1, 2, 4, 3, 3, 2, 2, 1, 2, 2, 1, 1, 1, 2, 3...
## $ Browser                 <dbl> 1, 2, 1, 2, 3, 2, 4, 2, 2, 4, 1, 1, 1, 5, 2...
## $ Region                  <dbl> 1, 1, 9, 2, 1, 1, 3, 1, 2, 1, 3, 4, 1, 1, 3...
## $ TrafficType             <dbl> 1, 2, 3, 4, 4, 3, 3, 5, 3, 2, 3, 3, 3, 3, 3...
## $ VisitorType             <chr> "Returning_Visitor", "Returning_Visitor", "...
## $ Weekend                 <lgl> FALSE, FALSE, FALSE, FALSE, TRUE, FALSE, FA...
## $ Revenue                 <lgl> FALSE, FALSE, FALSE, FALSE, FALSE, FALSE, F...
```

The majority of data types within this dataframe are doubles, which are numeric. There is one character data type in the Month column and logical data types in the Weekend and Revenue columns. The logical data type columns are TRUE if a visitor was on during a weekend and TRUE if a purchase was made. It is FALSE otherwise.

The second code cell in this section shows code which gets rid of any NA's in the dataframe. This was executed by simply reading the rows which had complete cases (no NA's) and ignoring the rows which did not. The old tibble was read over with the new tibble.

```
# getting rid of any rows with NA's. The 0's in the data all appear to be correct and should
# not be altered. The reasoning behind getting rid of the NA's instead of trying to replace
# these values is due to not wanting to make up data we have no information about.

commerce <- commerce[complete.cases(commerce),]
```

The next code cell tried to transform the data using various techniques. The technique used here is feature construction. Feature construction is when existing data is altered to make it easier to make predictions or handle other data related tasks. Looking at the commerce data, it was noticed that the dataframe had its months randomly ordered. To fix this, each month was assigned its numerical value in a new column titled

"MonthNum." Then, it was ordered so that so that each row was in calender order starting at February (two months are missing from the data: January and April). More compact methods of coding like using the library "Lubridate" were tried. However, the Month column could not be changed to a date-time data type due to the characters in the column being "not in a standard unambiguous format," so functions were not able to transform the data itself. This process can help future analysis by making grouping data by month easier.

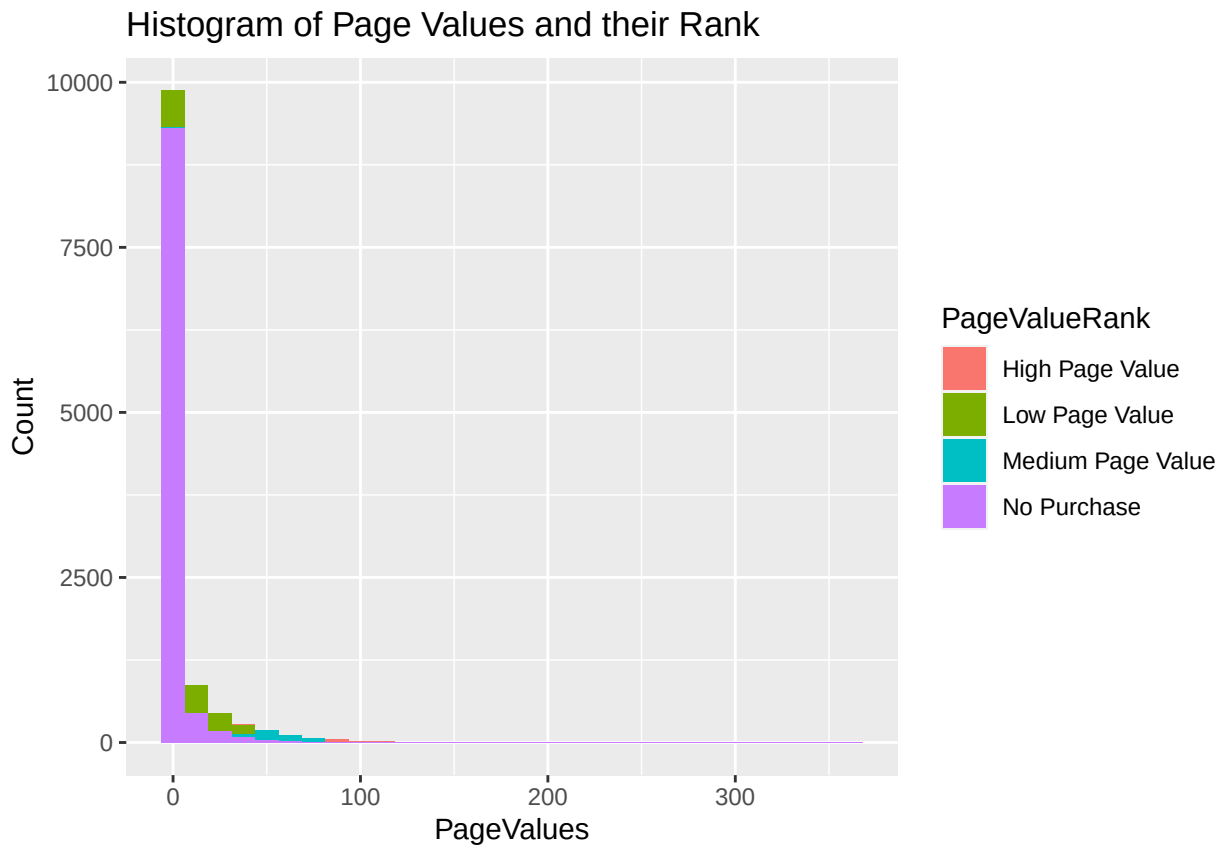
```
# making a new column in the tibble
commerce <- commerce %>%
  mutate(MonthNum = Month)

# giving each month its numrical value
for(i in 1:length(commerce$MonthNum)){
  if(commerce[i, "Month"] == "Feb")
  {
    commerce[i, "MonthNum"] <- "2"
  }
  if(commerce[i, "Month"] == "Mar")
  {
    commerce[i, "MonthNum"] <- "3"
  }
  if(commerce[i, "Month"] == "May")
  {
    commerce[i, "MonthNum"] <- "5"
  }
  if(commerce[i, "Month"] == "June")
  {
    commerce[i, "MonthNum"] <- "6"
  }
  if(commerce[i, "Month"] == "Jul")
  {
    commerce[i, "MonthNum"] <- "7"
  }
  if(commerce[i, "Month"] == "Aug")
  {
    commerce[i, "MonthNum"] <- "8"
  }
  if(commerce[i, "Month"] == "Sep")
  {
    commerce[i, "MonthNum"] <- "9"
  }
  if(commerce[i, "Month"] == "Oct")
  {
    commerce[i, "MonthNum"] <- "10"
  }
  if(commerce[i, "Month"] == "Nov")
  {
    commerce[i, "MonthNum"] <- "11"
  }
  if(commerce[i, "Month"] == "Dec")
  {
    commerce[i, "MonthNum"] <- "12"
  }
}
```

```r
# the numbers have to be converted to a numeric data type due to the column being a character
# data type when the mutate function was called.
commerce$MonthNum <- as.numeric(commerce$MonthNum)

# ordering the rows by month
commerce <- commerce[order(commerce$MonthNum),]
```

The last code cell for this section is to remove redundant data and perform need based discretization. The columns "operatingSystems," "Browser," "Region," and "TrafficType" were removed because the numbers in these columns gave no information that could be utilized. There is no way to know what "Region 9" is, what "Browser 1" is, etc. Next, need based discretization was performed by making a new column for Page Vales titled "PageValueRank." The "PageValues" column "represents the average value for a web page that a user visited before completing an e-commerce transaction." The purpose of the new column "PageValueRank" is to assign rows with four different categories: "No Purchase," "Low Page Value," "Medium Page Value," and "High Page Value." If no e-commerce trade is performed, there can be no average time before completing an e-commerce transaction, so by default these values are zero. So, whenever the Revenue column read "FALSE," the category the page value was assigned to was "No Purchase." If the Revenue column read "TRUE," then the three other categories were assigned based on the value of the page values. Page values less than or equal to 40 were given the "Low Page Value" rank. Page values between 40 and 80 were assigned "Medium Page Value," and "High Page Value" was given to page values greater than 80. This process was done so that data analysis can be preformed on different page value rank categories to see if there is anything interesting or different in each one.

```r
# removing columns that give us no useful information
commerce <- commerce %>%
  select(-OperatingSystems, -Browser, -Region, -TrafficType)

# There are a lot of  0's in the PageValues variable due to most of the sessions not resulting
# in a purchase, so a new column was made to divide these values into 4 parts, "No Purchase",
# "Low Page Value", "Medium Page Value", and "High Page Value." These categories are in the new
# column titled "PageValueRank".

commerce <-
  commerce %>%
  mutate(PageValueRank = PageValues)

commerce$PageValueRank <- as.character(commerce$PageValueRank)

for(i in 1:length(commerce$PageValues)){
  if(commerce[i, "Revenue"] == FALSE)
  {
    commerce[i, "PageValueRank"] <- "No Purchase"
  }
  else
  {
    if(commerce[i, "PageValues"] <= 40)
    {
      commerce[i, "PageValueRank"] <- "Low Page Value"
    }
    if(commerce[i, "PageValues"] > 40 & commerce[i, "PageValues"] <= 80)
    {
      commerce[i, "PageValueRank"] <- "Medium Page Value"
    }
```

```
    if(commerce[i, "PageValues"] > 80)
    {
      commerce[i, "PageValueRank"] <- "High Page Value"
    }
  }
}

# It is interesting to see that there are some page values above 0 when it states that revenue is
# false for that session. I am unsure if this is an error or not. There are page values that
# exceed 300 that cannot be seen due to the count of zeros being so large in comparison.
commerce %>%
  ggplot(aes(PageValues, fill = PageValueRank)) +
  geom_histogram() +
  labs(title = "Histogram of Page Values and their Rank", y = "Count")
```



It can be noticed in this histogram that there are some pages values are above 0 but still list revenue as "FALSE." Also, there are times when a purchase was made, but the average page value is 0. Both of these occurrences in the data are contradictory.

2. Data Analytics and Visualization

We want to identify categorical, ordinal, and numerical variables within dataframe: The dataset consists of: - 5 categorical attributes: (OperatingSystems, Browser, Region, TrafficType, VisitorType) - 2

binary attributes: (Weekend, Revenue) - 1 ordinal attributes: (Month) - 10 numerical attributes: (Administrative, Administrative_Duration, Informational, Informational_Duration, ProductRelated, ProductRelated_Duration, BounceRates, ExitRates, PageValues, SpecialDay)

```
knitr::kable(sapply(commerce, class), "simple",col.names = c('Data type'))
```

	Data type
Administrative	numeric
Administrative_Duration	numeric
Informational	numeric
Informational_Duration	numeric
ProductRelated	numeric
ProductRelated_Duration	numeric
BounceRates	numeric
ExitRates	numeric
PageValues	numeric
SpecialDay	numeric
Month	character
VisitorType	character
Weekend	logical
Revenue	logical
MonthNum	numeric
PageValueRank	character

As we see the target variable is a boolian variable with 1042 FALSE and 1908 TRUE values.

```
summary(commerce$Revenue)
```

```
##    Mode   FALSE    TRUE
## logical   10097    1851
```

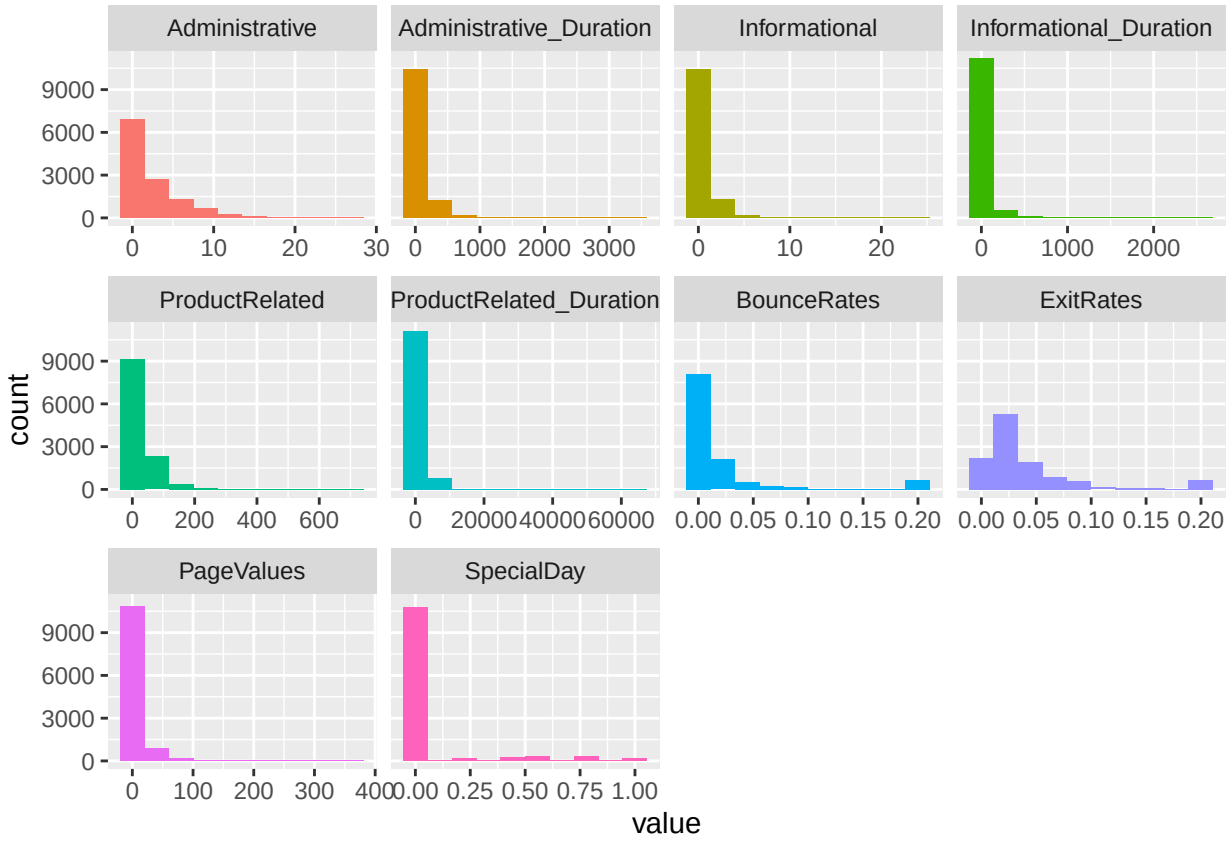
Now we convert Revenue field to 0,1 values.

```
commerce <- commerce %>% mutate(Revenue = ifelse(Revenue == "FALSE",0,1))
```

Exploratory Data Analysis: Numerical Analysis

We visually analyze to see revenue for users is they are Administrative or not. Lets see histogram of all the continuous variables.

```
commerceNumberics <- commerce[,0:10]
plot_num(commerceNumberics)
```



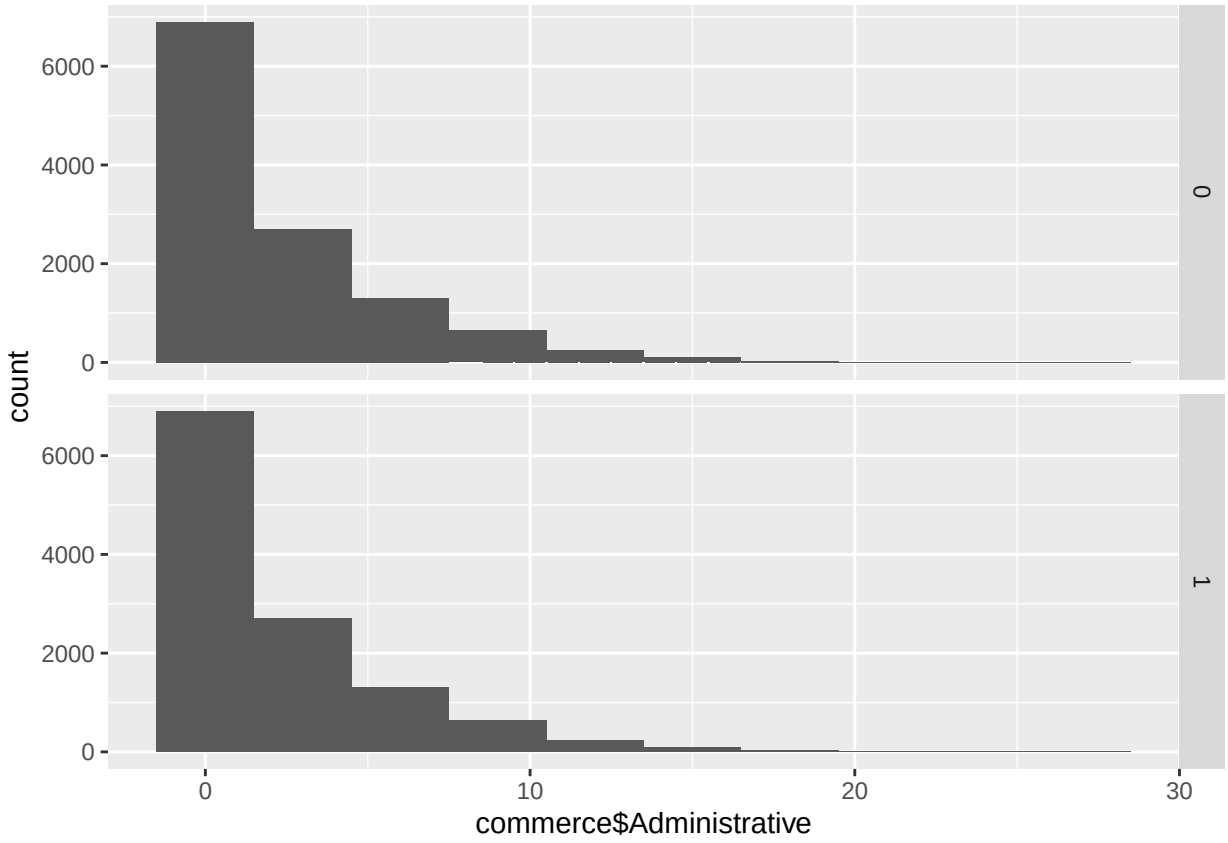
Administrative

As we see in most of cases there was no revenue when page view was from Administrative.

```
summary(commerce$Administrative)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   0.000   0.000   1.000   2.314   4.000  27.000
```

```
ggplot() +
aes(x = commerce$Administrative) +
geom_histogram(bins = 10) +
geom_bar() +
facet_grid(commerce$Revenue ~ .)
```

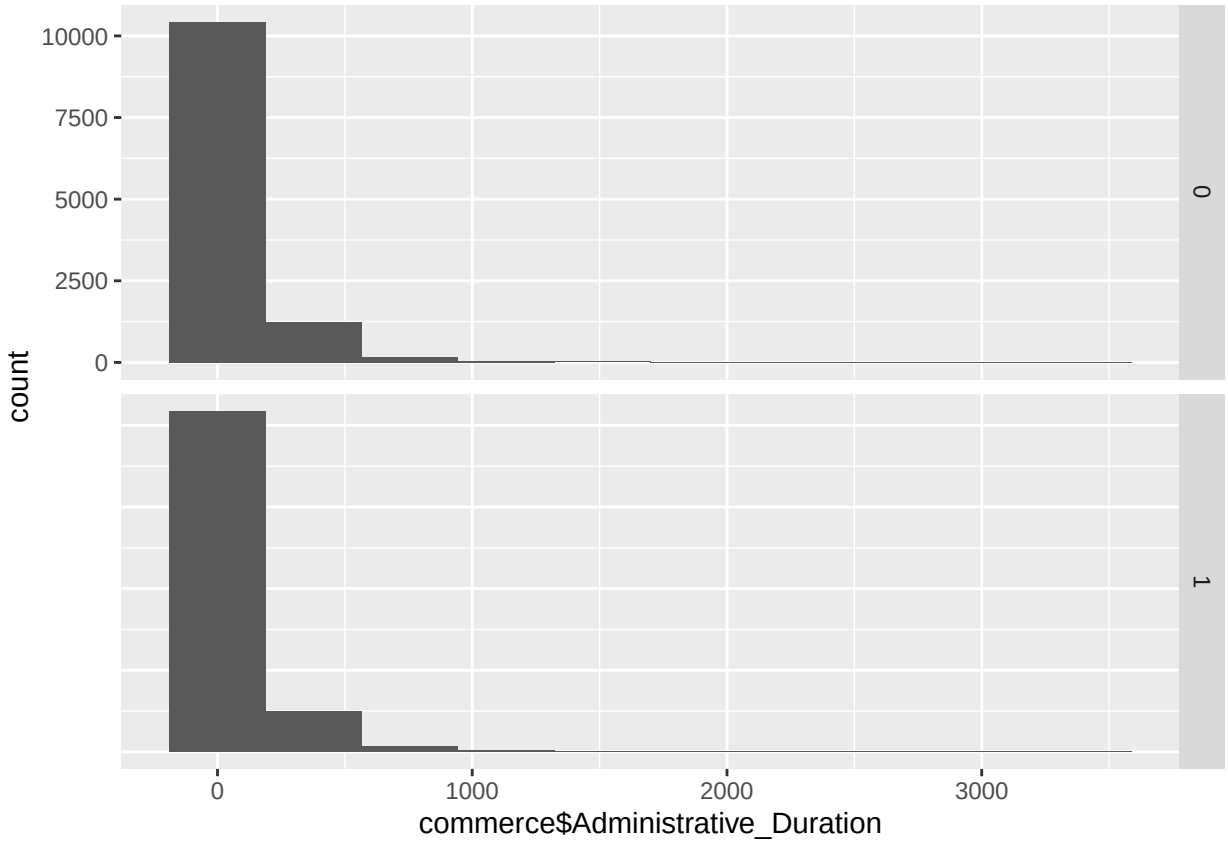


Administrative_Duration

```
summary(commerce$Administrative_Duration)
```

```
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
##    0.000    0.000    7.112   80.975   94.000 3398.750
```

```
ggplot() +
aes(x = commerce$Administrative_Duration) +
geom_histogram(bins = 10) +
facet_grid(commerce$Revenue ~ .,
           scales = "free_y")
```



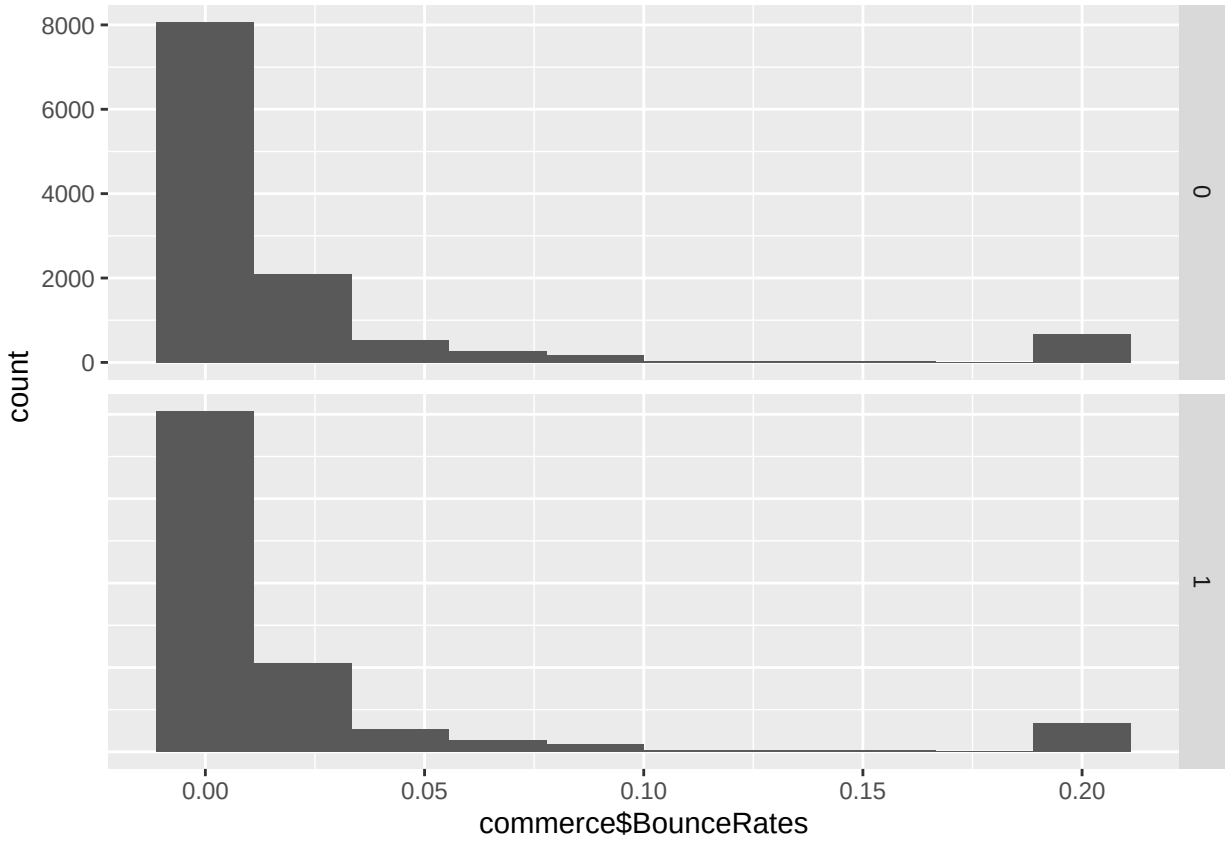
Bounce Rates

Plot shows to have higher Revenue we need to have lower BounceRates.

```
summary(commerce$BounceRates)
```

```
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
## 0.000000 0.000000 0.003107 0.022163 0.016667 0.200000
```

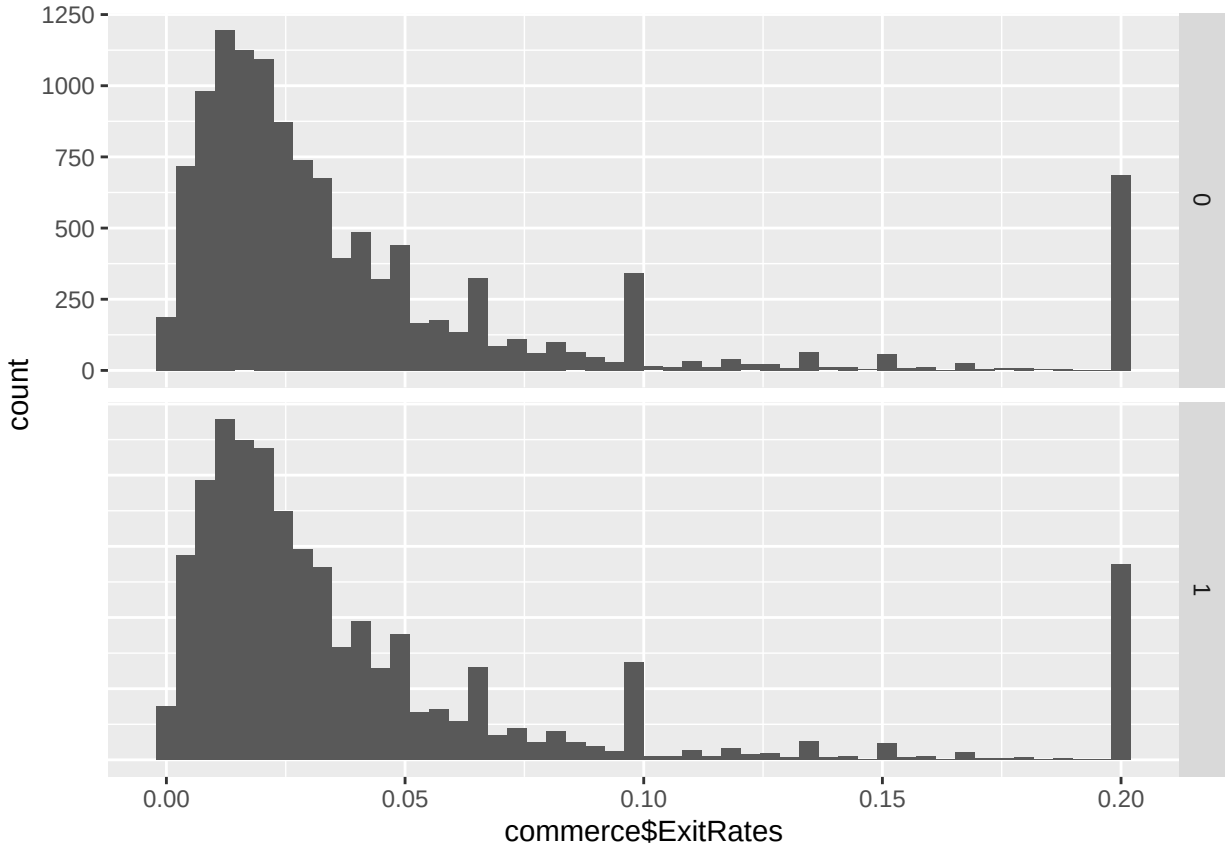
```
ggplot() +
aes(x = commerce$BounceRates) +
geom_histogram(bins = 10) +
facet_grid(commerce$Revenue ~ .,
           scales = "free_y")
```



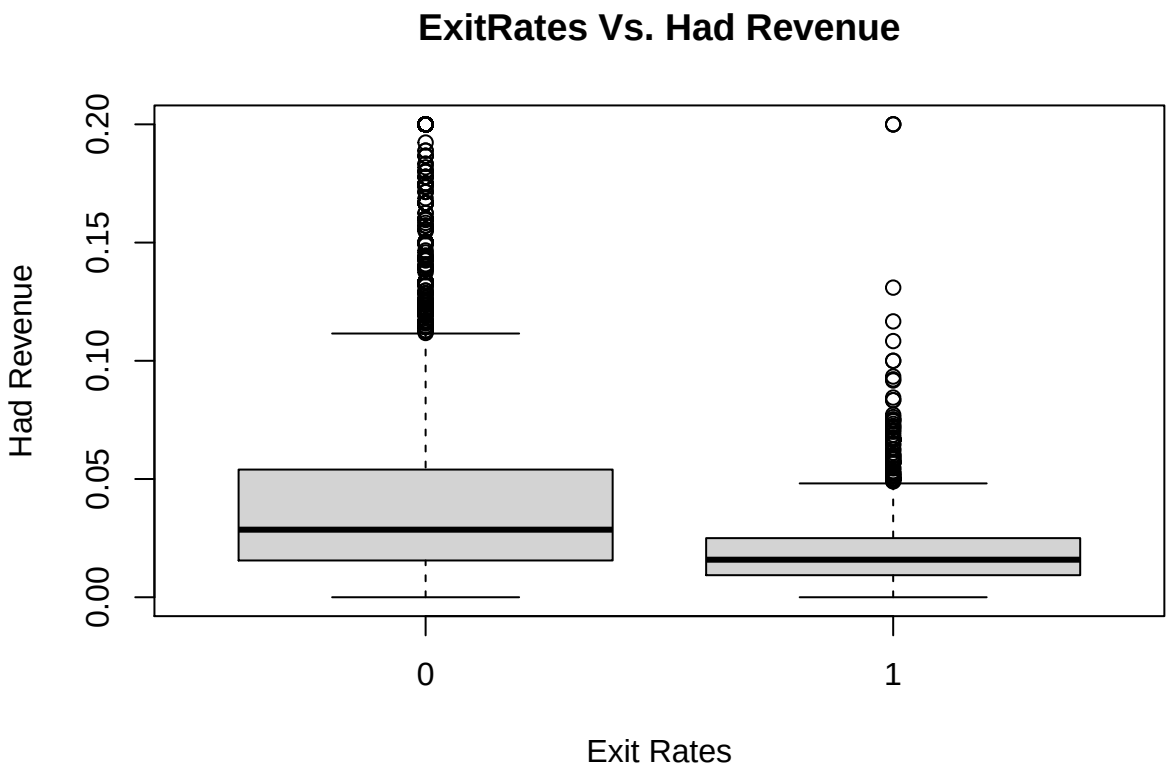
ExitRates

Plot shows to have higher revenue we need to have lower ExitRates Also we see in some pages we had high Exit rate and higher count of conversions. we assume those conversions happened on the checkout page and we consider them as outliers.

```
ggplot() +
aes(x = commerce$ExitRates) +
geom_histogram(bins = 50) +
facet_grid(commerce$Revenue ~ .,
           scales = "free_y") #outliers
```




```
boxplot(ExitRates~Revenue,data=commerce, main="ExitRates Vs. Had Revenue",
   xlab="Exit Rates", ylab="Had Revenue")
```




```
summary(commerce$ExitRates)
```

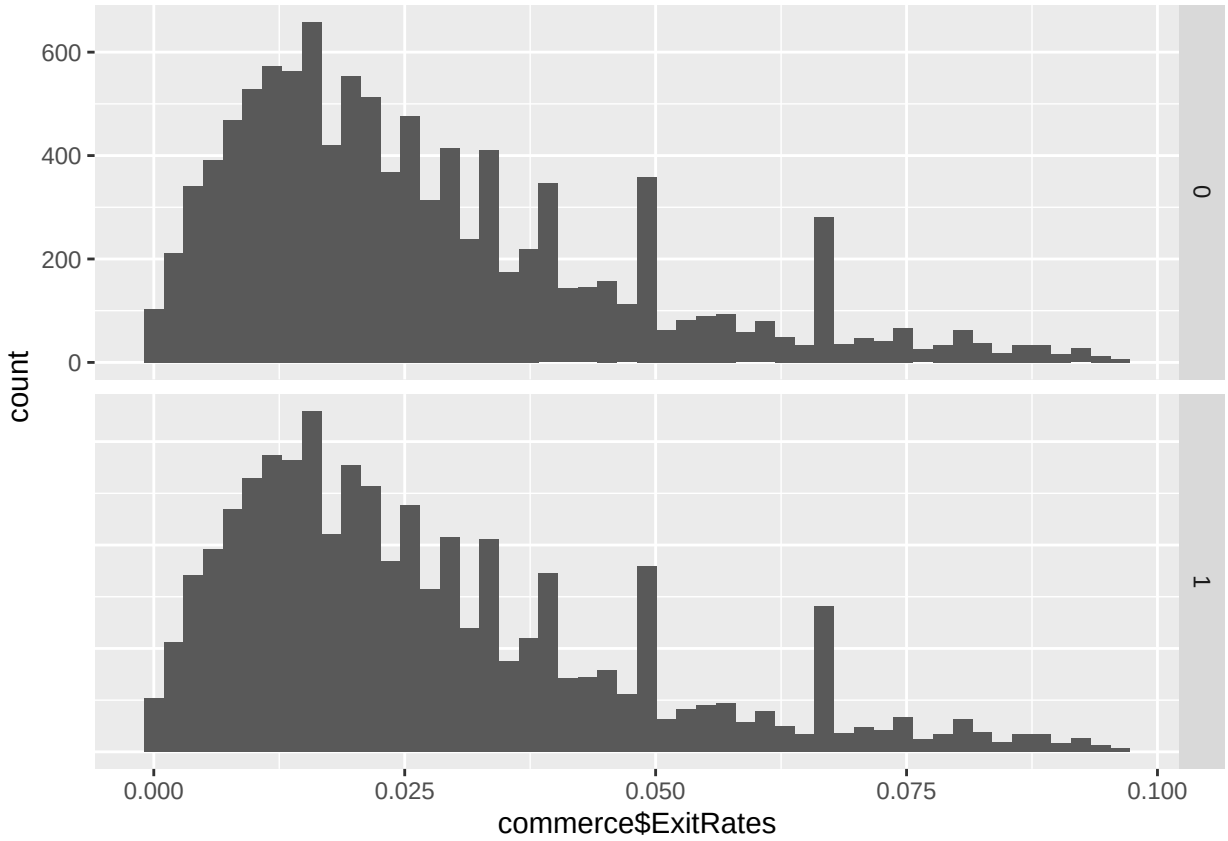
```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## 0.00000 0.01429 0.02511 0.04305 0.05000 0.20000
```

In the box plot we see there are outliers for those pageviews had revenue, so we need to handle the outliers:

```
# with quantile() we find the 25th and the 75th percentile of the dataset.
# with IQR() gives us the difference of the 75th and 25th percentiles
Q <- quantile(commerce$ExitRates, probs=c(.25, .75), na.rm = FALSE)
iqr <- IQR(commerce$ExitRates)
up <-  Q[2]+1.3*iqr # Upper Range
low<- Q[1]-1.3*iqr # Lower Range
# Eliminating Outliers
commerce<- subset(commerce, commerce$ExitRates > (low) & commerce$ExitRates < (up))
```

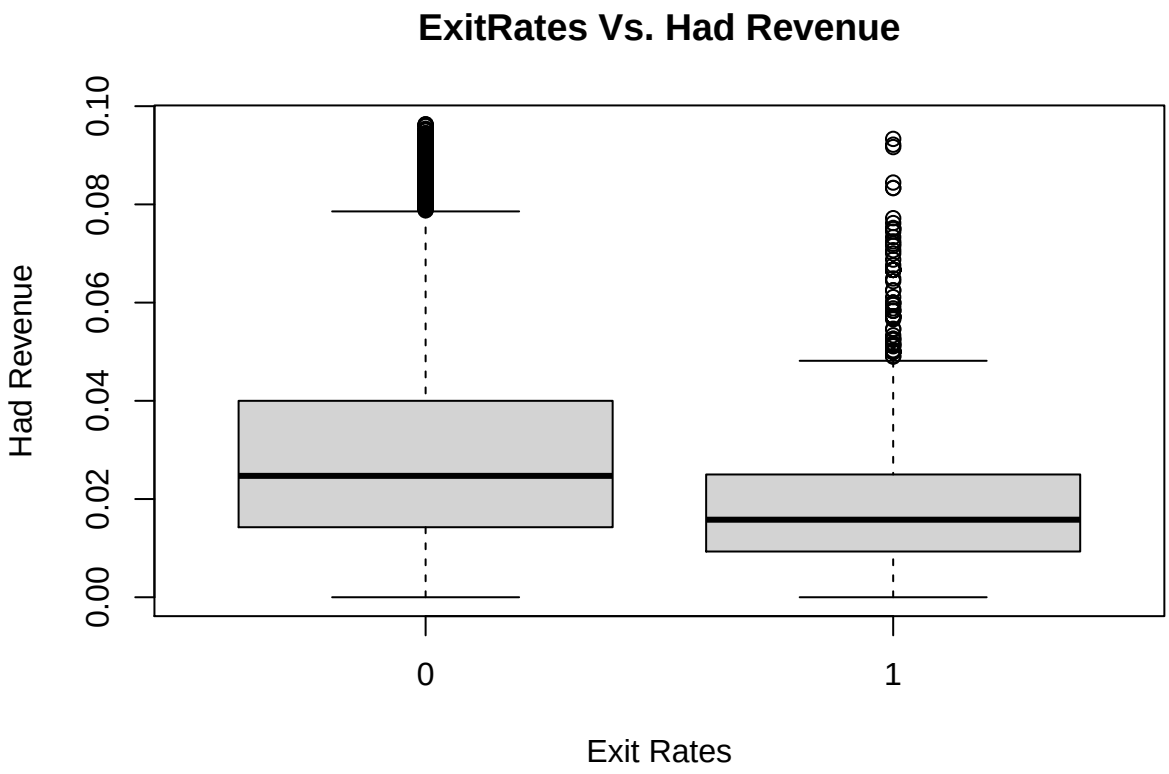
Check outliers again

```
ggplot() +
aes(x = commerce$ExitRates) +
geom_histogram(bins = 50) +
facet_grid(commerce$Revenue ~ .,
           scales = "free_y") #outliers
```



We check to see if we could handle the outliers for Exit Rates:

```
boxplot(ExitRates~Revenue,data=commerce, main="ExitRates Vs. Had Revenue",
   xlab="Exit Rates", ylab="Had Revenue")
```




```
summary(commerce$ExitRates)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## 0.00000 0.01286 0.02222 0.02756 0.03774 0.09630
```

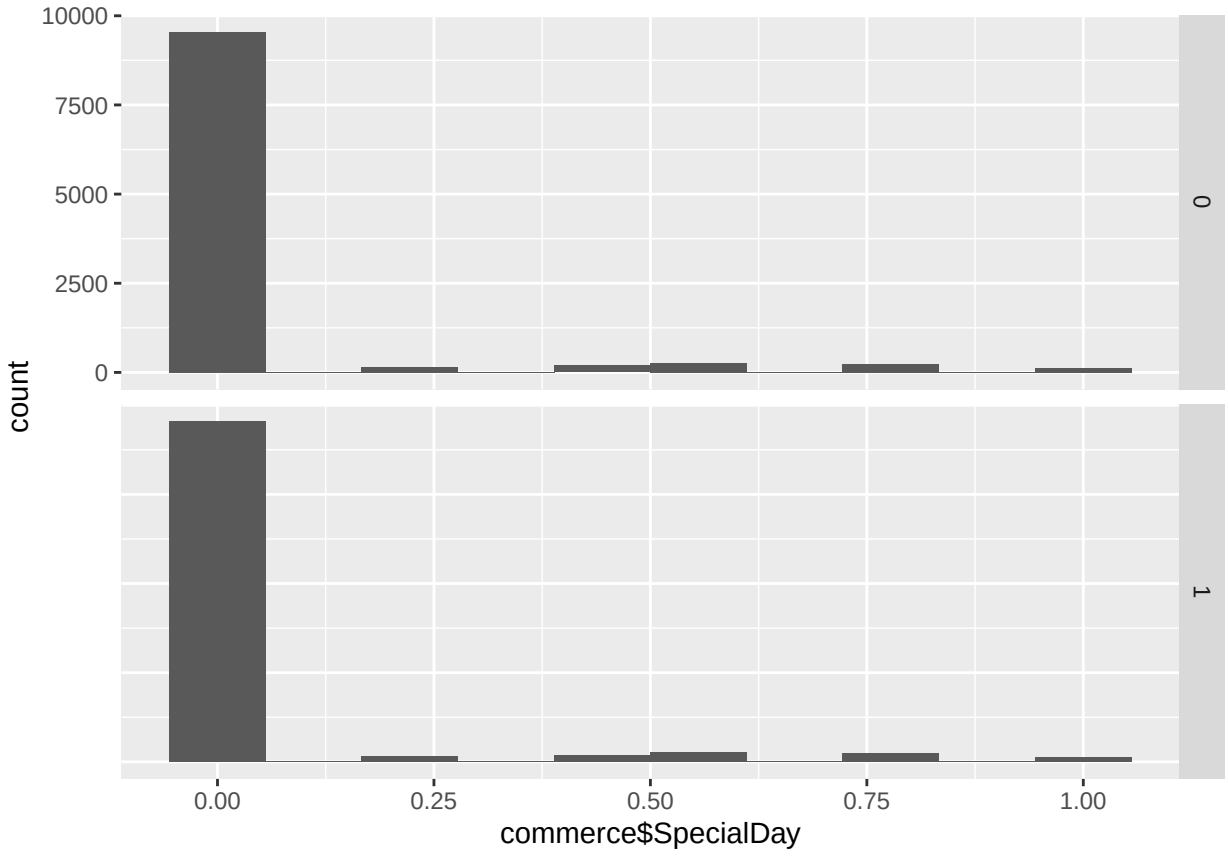
SpecialDay

SpecialDay indicates the amount of closeness of the visiting time to a special day. The chart clearly shows in special days site had more sales.

```
summary(commerce$SpecialDay)
```

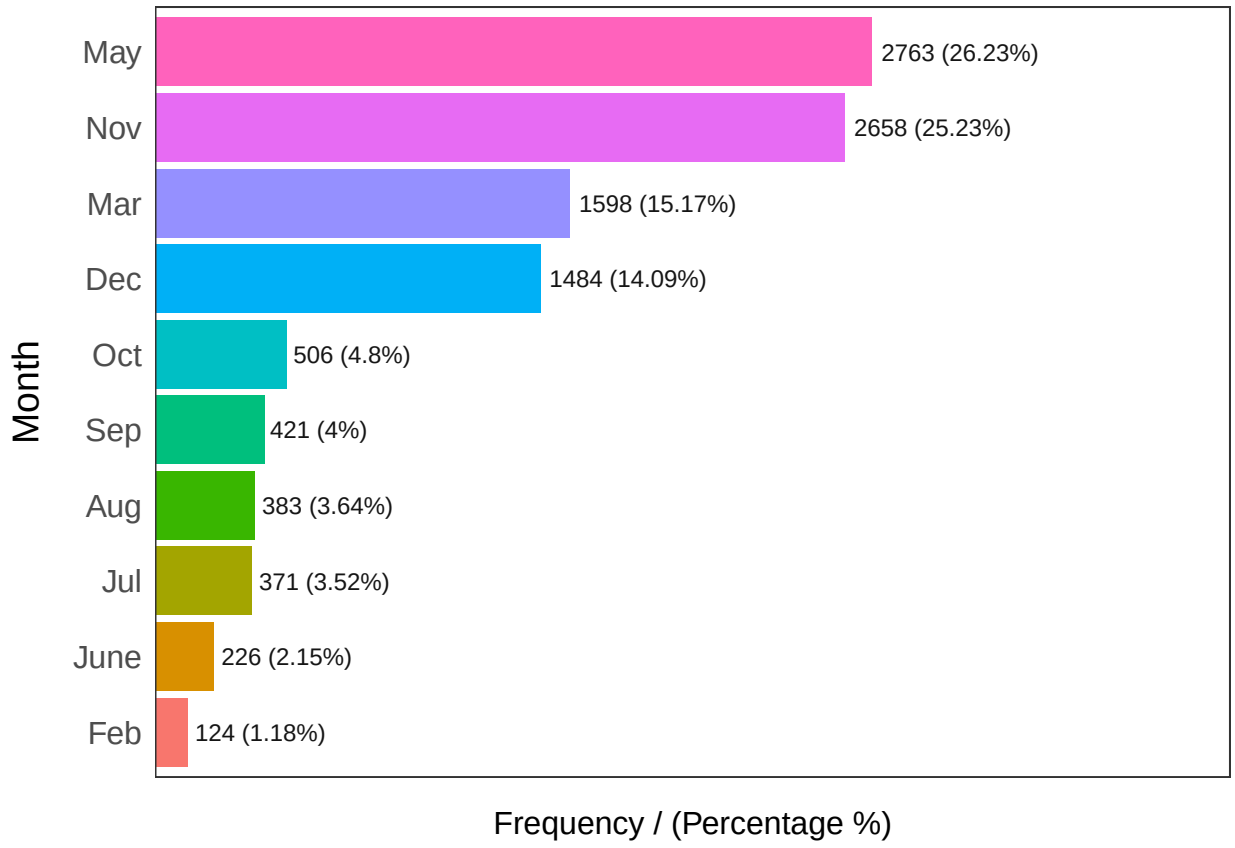
```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## 0.00000 0.00000 0.00000 0.05629 0.00000 1.00000
```

```
ggplot() +
aes(x = commerce$SpecialDay) +
geom_histogram(bins = 10) +
facet_grid(commerce$Revenue ~ .,
           scales = "free_y")
```

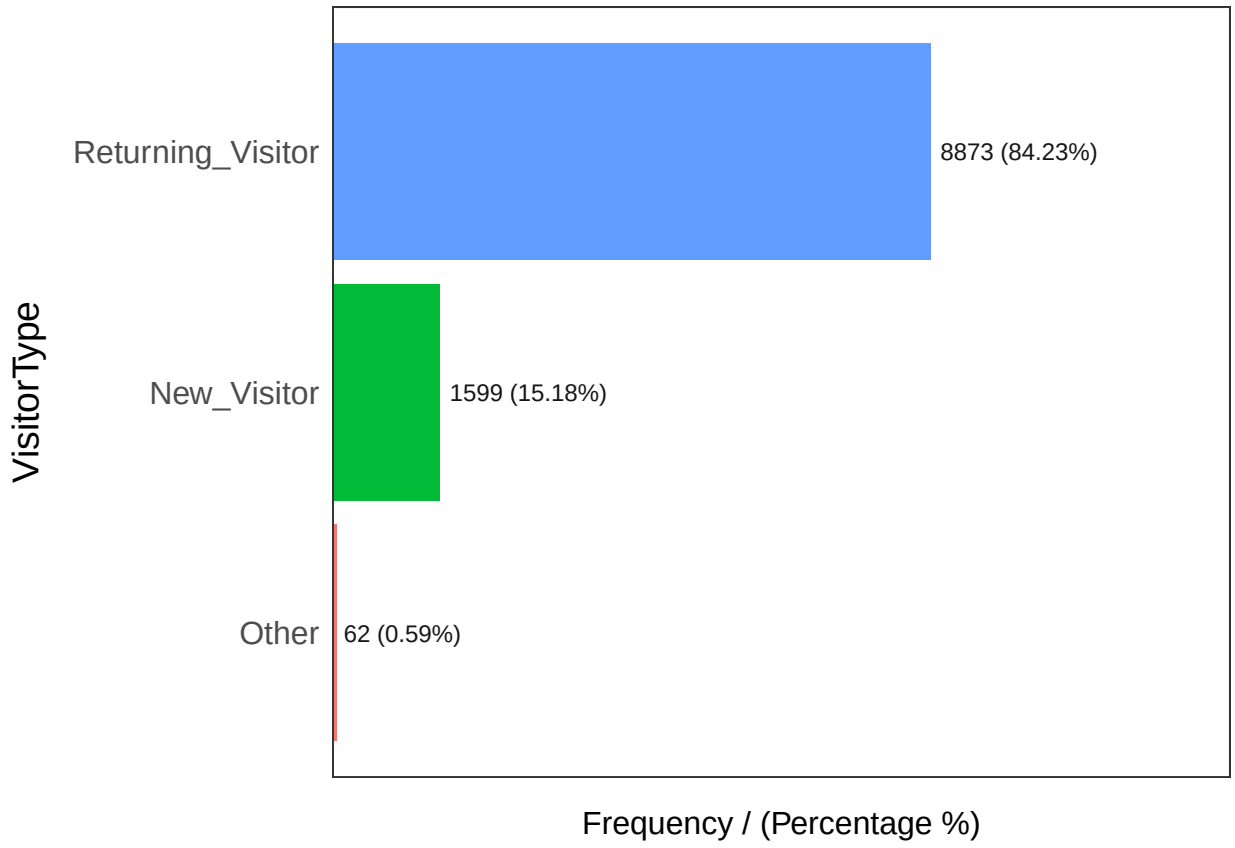


Categorical fields Analysis: frequency plots of catergorical variables

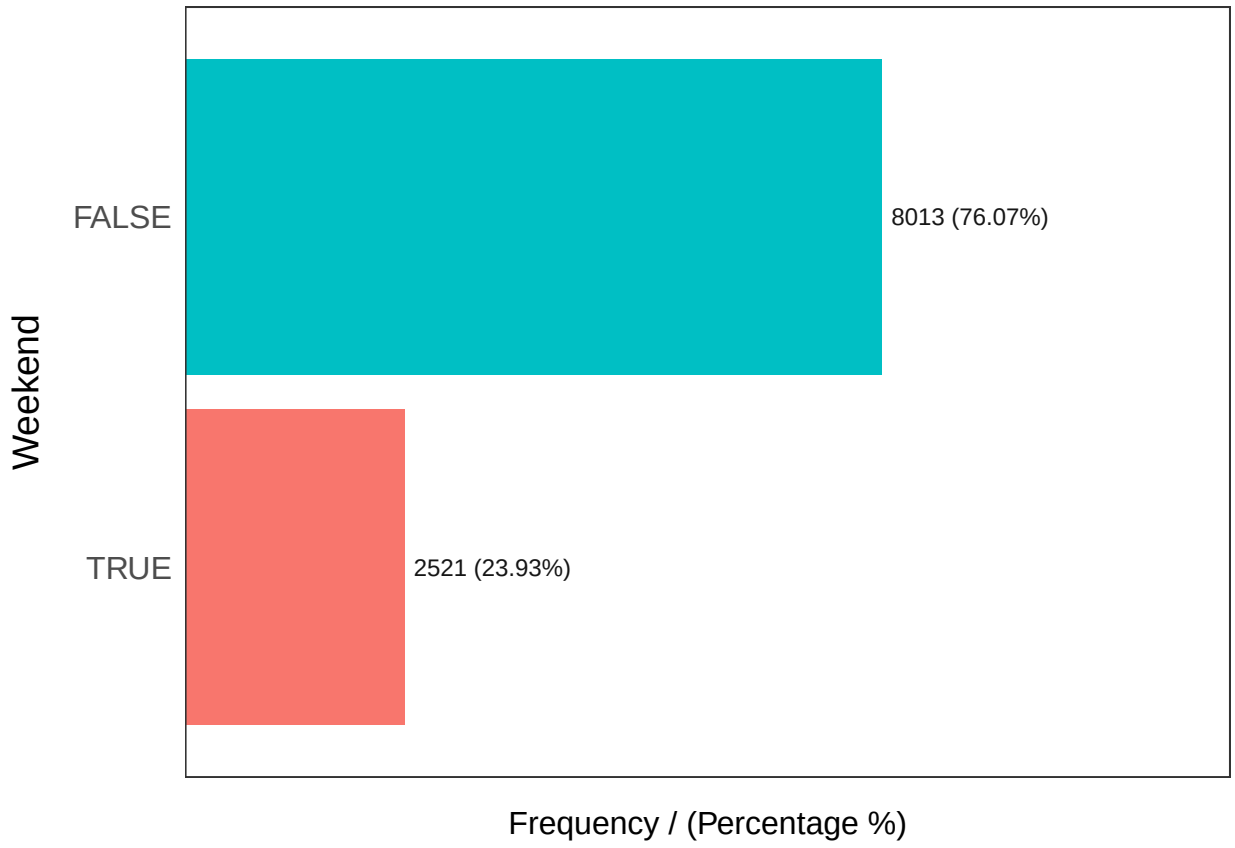
```
commerceCatergorical <- commerce[11:16]

commerceCatergorical$Weekend<- as.factor(commerceCatergorical$Weekend)
commerceCatergorical$Revenue<- as.factor(commerceCatergorical$Revenue)
print(freq(commerceCatergorical))
```



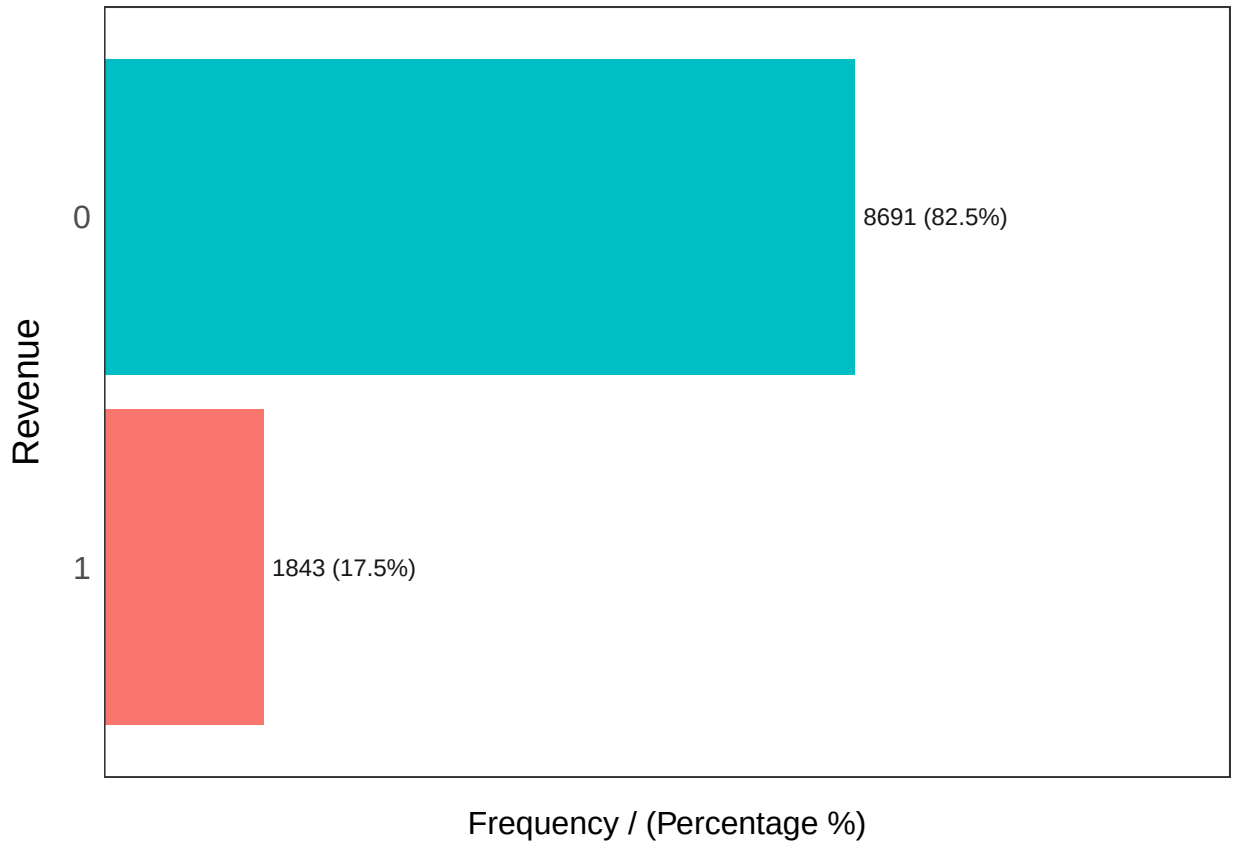

```
##    Month frequency percentage cumulative_perc
## 1    May      2763      26.23           26.23
## 2    Nov      2658      25.23           51.46
## 3    Mar      1598      15.17           66.63
## 4    Dec      1484      14.09           80.72
## 5    Oct       506       4.80           85.52
## 6    Sep       421       4.00           89.52
## 7    Aug       383       3.64           93.16
## 8    Jul       371       3.52           96.68
## 9   June       226       2.15           98.83
## 10   Feb       124       1.18          100.00
```



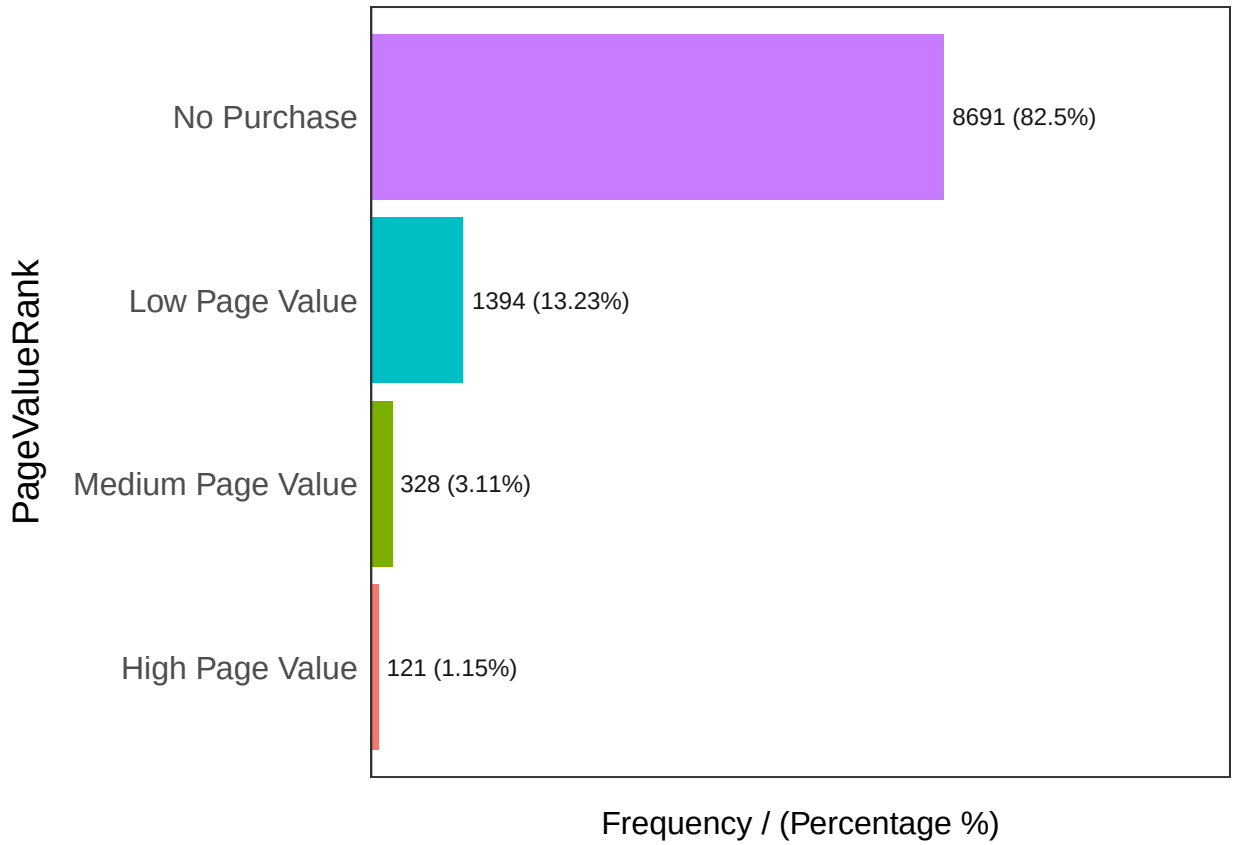

```
##         VisitorType frequency percentage cumulative_perc
## 1 Returning_Visitor      8873      84.23           84.23
## 2       New_Visitor      1599      15.18           99.41
## 3             Other        62       0.59          100.00
```




```
##   Weekend frequency percentage cumulative_perc
## 1   FALSE      8013      76.07           76.07
## 2    TRUE      2521      23.93          100.00
```



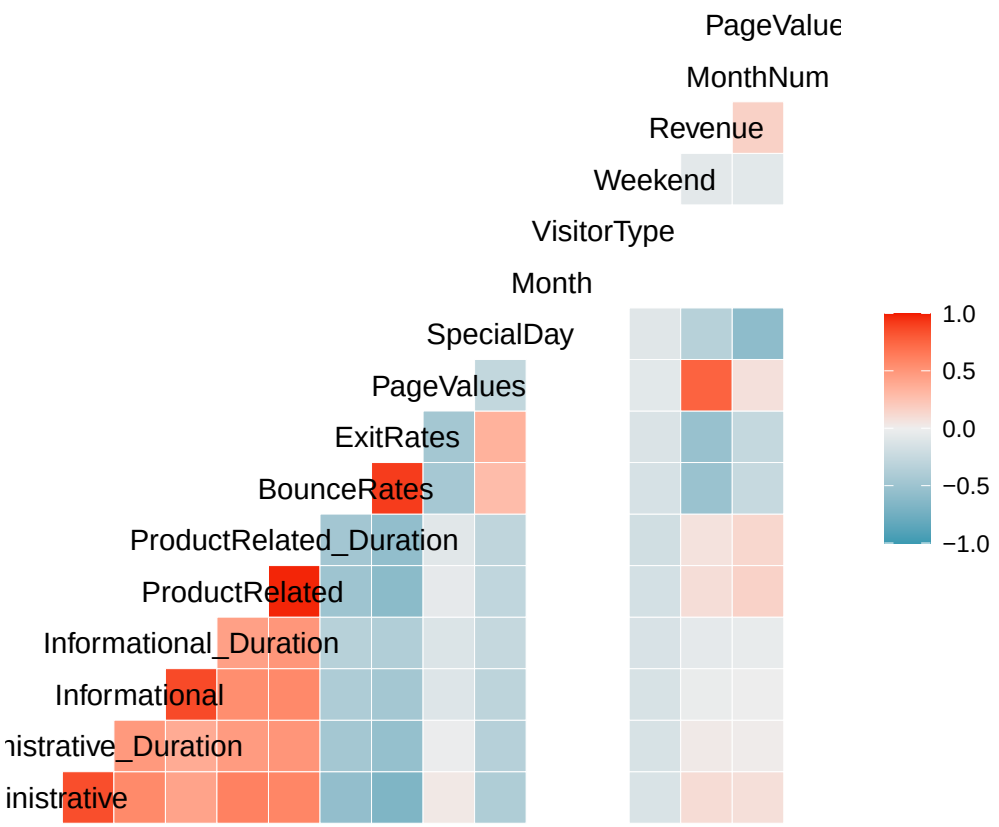

```
##   Revenue frequency percentage cumulative_perc
## 1       0      8691       82.5            82.5
## 2       1      1843       17.5           100.0
```




```
##       PageValueRank frequency percentage cumulative_perc
## 1       No Purchase      8691      82.50           82.50
## 2    Low Page Value      1394      13.23           95.73
## 3 Medium Page Value       328       3.11           98.84
## 4   High Page Value       121       1.15          100.00
##
## [1] "Variables processed: Month, VisitorType, Weekend, Revenue, PageValueRank"
```

We want to find correlation between columns lets take a look at all columns to have a big picture of any potential correlation between fields. Now we plot our new data frame to visually inspect the data:

```
commerce$Month<- as.numeric(commerce$Month)
commerce$VisitorType<- as.numeric(commerce$VisitorType)
commerce$PageValueRank <- as.numeric(commerce$PageValueRank)
ggcorr(cor(commerce))
```

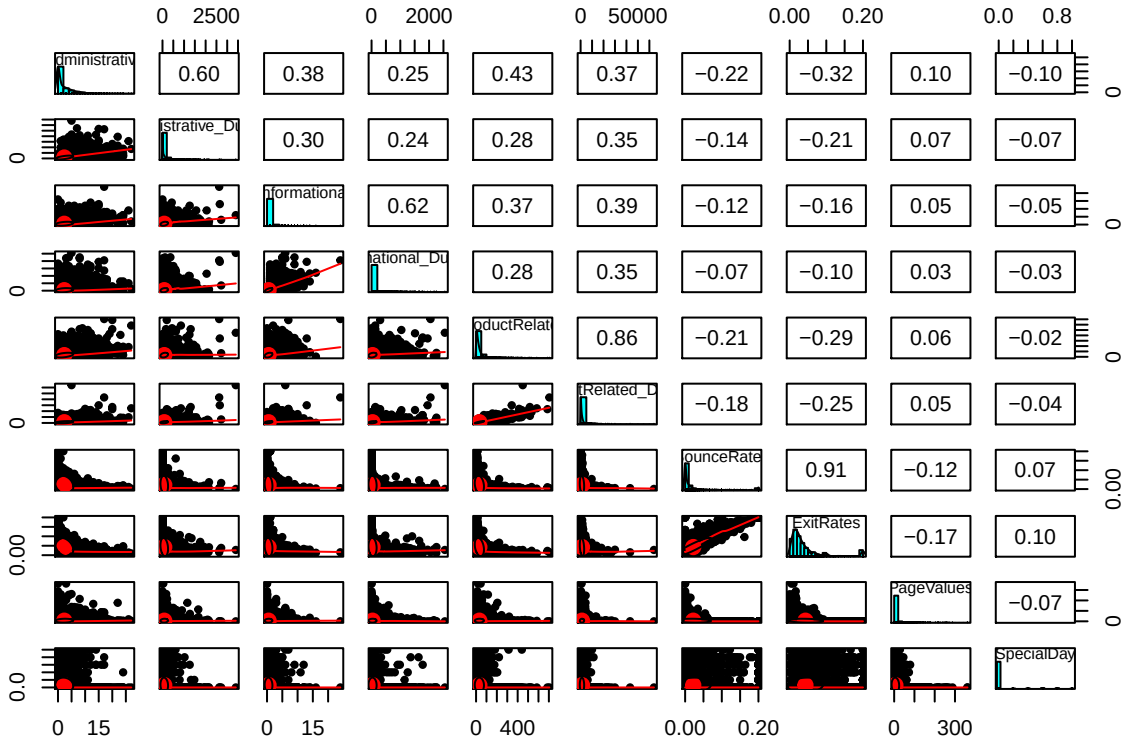


As we see there is strong correlation between Revenue and some of other columns.

list of strong positive correlations with revenue: 1- Page Values 2- ProductRelated_Duration 3- ProductRelated 4- Administrative
5- Administrative_Duration

list of strong negative correlations: 1- BounceRates
2- ExitRates

```
pairs.panels(commerceNumberics)
```

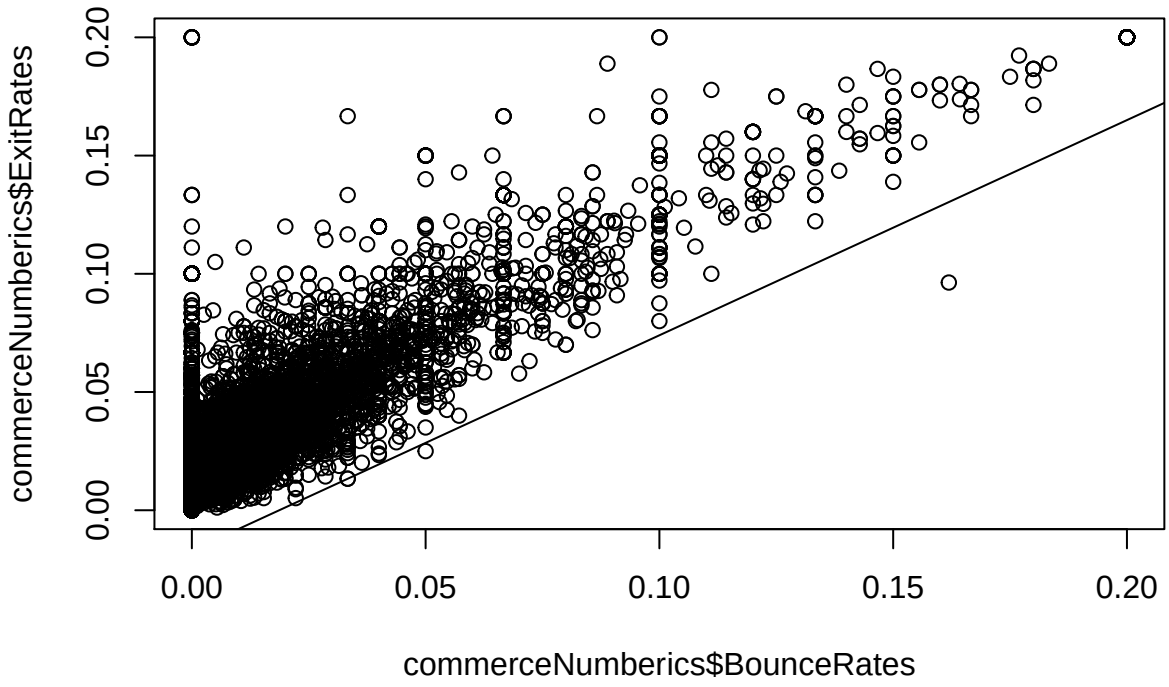


Here we see lots of strong correlation between fields, for example lets see correlation between BounceRates and ExitRates we are looking for a formula to predict BounceRates based on ExitRates, first lets see the relation visually:

```
BounceRates.ExitRates.lm <- lm(BounceRates~ExitRates, data = commerceNumberics)
BounceRates.ExitRates.lm
```

```
##
## Call:
## lm(formula = BounceRates ~ ExitRates, data = commerceNumberics)
##
## Coefficients:
## (Intercept)    ExitRates
##    -0.01702      0.91012
```

```
a = BounceRates.ExitRates.lm[[1]][1]
b = BounceRates.ExitRates.lm[[1]][2]
plot(commerceNumberics$BounceRates,commerceNumberics$ExitRates)+
abline(a, b)
```




```
## integer(0)
```

We see a positive strong correlation between BounceRates and ExitRates so we want to have a formula to predict BounceRates based on ExitRates and calculate Coefficent and intercept of predictor and have a formula like: BounceRates= -0.01702 + 0.9101206 * ExitRates BounceRates= 0.02399892 + -0.0003090976 * PageValues

Sample of strong correlation:

```
BounceRates.ExitRates.lm <- lm(BounceRates~ExitRates, data = commerceNumberics)
a = BounceRates.ExitRates.lm[[1]][1]
b = BounceRates.ExitRates.lm[[1]][2]
cat("BounceRates=",a,"+",b," * ExitRates")
```

```
## BounceRates= -0.01702 + 0.9101206  * ExitRates
```

Sample of weak correlation:

```
BounceRates.PageValues.lm <- lm(BounceRates~PageValues, data = commerceNumberics)
a = BounceRates.PageValues.lm[[1]][1]
b = BounceRates.PageValues.lm[[1]][2]
cat("BounceRates=",a,"+",b," * PageValues")
```

```
## BounceRates= 0.02399892 + -0.0003090976  * PageValues
```

3. Data Analysis

The first analytics performed was developing a linear regression model looking at product related duration and product related. We anticipate that these columns will have strong correlation and a high R squared. This was done to establish a baseline test of the data set.

```
Linmodel<- lm(ProductRelated_Duration ~ ProductRelated, data = commerce)
Linmodel
```

```
##
## Call:
## lm(formula = ProductRelated_Duration ~ ProductRelated, data = commerce)
##
## Coefficients:
##    (Intercept)  ProductRelated
##          31.73           36.91
```

```
summary(Linmodel)
```

```
##
## Call:
## lm(formula = ProductRelated_Duration ~ ProductRelated, data = commerce)
##
## Residuals:
##    Min     1Q Median     3Q    Max
##  -7176   -334   -113    190  47368
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)     31.7337    12.7871   2.482   0.0131 *
## ProductRelated  36.9130     0.2209 167.140   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1038 on 10532 degrees of freedom
## Multiple R-squared:  0.7262, Adjusted R-squared:  0.7262
## F-statistic: 2.794e+04 on 1 and 10532 DF,  p-value: < 2.2e-16
```

```
cor(commerce$ProductRelated_Duration, commerce$ProductRelated)
```

```
## [1] 0.8521814
```

```
#Here we identified the correlation between all numeric variables within our dataset.
cor(commerce[,unlist(lapply(commerce, is.numeric))])
```

```
##                          Administrative Administrative_Duration Informational
## Administrative               1.00000000              0.58626027    0.35613501
## Administrative_Duration      0.58626027              1.00000000    0.28901552
```

```
## Informational               0.35613501              0.28901552    1.00000000
## Informational_Duration      0.24148885              0.22913601    0.61961990
## ProductRelated              0.39707952              0.25449358    0.35325865
## ProductRelated_Duration     0.33938723              0.32788163    0.36830696
## BounceRates                -0.10817592             -0.05440423   -0.01809168
## ExitRates                  -0.26959615             -0.16563708   -0.10765848
## PageValues                  0.07469946              0.05069902    0.03377452
## SpecialDay                 -0.09042262             -0.07004296   -0.04315899
## Month                               NA                      NA           NA
## VisitorType                         NA                      NA           NA
## Revenue                     0.10297247              0.07127889    0.07410562
## MonthNum                    0.08392472              0.04613142    0.05389575
## PageValueRank                       NA                      NA           NA
##                         Informational_Duration ProductRelated
## Administrative                      0.24148885     0.39707952
## Administrative_Duration             0.22913601     0.25449358
## Informational                       0.61961990     0.35325865
## Informational_Duration              1.00000000     0.26788319
## ProductRelated                      0.26788319     1.00000000
## ProductRelated_Duration             0.33829710     0.85218135
## BounceRates                        -0.01438618    -0.06652198
## ExitRates                          -0.07608368    -0.23222613
## PageValues                          0.02119254     0.03181612
## SpecialDay                         -0.02735728    -0.01241273
## Month                                       NA             NA
## VisitorType                                 NA             NA
## Revenue                             0.05668019     0.12877807
## MonthNum                            0.03903236     0.15009814
## PageValueRank                               NA             NA
##                         ProductRelated_Duration BounceRates   ExitRates
## Administrative                       0.33938723 -0.10817592 -0.26959615
## Administrative_Duration              0.32788163 -0.05440423 -0.16563708
## Informational                        0.36830696 -0.01809168 -0.10765848
## Informational_Duration               0.33829710 -0.01438618 -0.07608368
## ProductRelated                       0.85218135 -0.06652198 -0.23222613
## ProductRelated_Duration              1.00000000 -0.04787249 -0.17090758
## BounceRates                         -0.04787249  1.00000000  0.65183940
## ExitRates                           -0.17090758  0.65183940  1.00000000
## PageValues                           0.02969908 -0.10944245 -0.19462986
## SpecialDay                          -0.02631774  0.13702217  0.15199178
## Month                                       NA          NA          NA
## VisitorType                                 NA          NA          NA
## Revenue                              0.12475927 -0.12421152 -0.20019951
## MonthNum                             0.13027483 -0.01969967 -0.06506180
## PageValueRank                               NA          NA          NA
##                          PageValues  SpecialDay Month VisitorType     Revenue
## Administrative           0.07469946 -0.09042262    NA          NA  0.10297247
## Administrative_Duration  0.05069902 -0.07004296    NA          NA  0.07127889
## Informational            0.03377452 -0.04315899    NA          NA  0.07410562
## Informational_Duration   0.02119254 -0.02735728    NA          NA  0.05668019
## ProductRelated           0.03181612 -0.01241273    NA          NA  0.12877807
## ProductRelated_Duration  0.02969908 -0.02631774    NA          NA  0.12475927
## BounceRates             -0.10944245  0.13702217    NA          NA -0.12421152
## ExitRates               -0.19462986  0.15199178    NA          NA -0.20019951
```

```
## PageValues                1.00000000 -0.06382562    NA          NA  0.48432154
## SpecialDay               -0.06382562  1.00000000    NA          NA -0.08212286
## Month                             NA          NA     1          NA          NA
## VisitorType                       NA          NA    NA           1          NA
## Revenue                   0.48432154 -0.08212286    NA          NA  1.00000000
## MonthNum                  0.06162089 -0.25343436    NA          NA  0.12218089
## PageValueRank                     NA          NA    NA          NA          NA
##                           MonthNum PageValueRank
## Administrative            0.08392472            NA
## Administrative_Duration   0.04613142            NA
## Informational             0.05389575            NA
## Informational_Duration    0.03903236            NA
## ProductRelated            0.15009814            NA
## ProductRelated_Duration   0.13027483            NA
## BounceRates              -0.01969967            NA
## ExitRates                -0.06506180            NA
## PageValues                0.06162089            NA
## SpecialDay               -0.25343436            NA
## Month                             NA            NA
## VisitorType                       NA            NA
## Revenue                   0.12218089            NA
## MonthNum                  1.00000000            NA
## PageValueRank                     NA             1
```

The R-squared was roughly .74 which means the model explains 74% of the variance. This strong relationship was anticipated. We will explore further in-depth analysis with multiple supervised learning techniques.

Multiple Linear Regression

```
fit<-lm(MonthNum~Administrative + Informational + ProductRelated, data = commerce)

summary(fit)
```

```
##
## Call:
## lm(formula = MonthNum ~ Administrative + Informational + ProductRelated,
##     data = commerce)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -9.7351 -2.7297  0.1611  3.3572  4.7067
##
## Coefficients:
##                  Estimate Std. Error t value Pr(>|t|)
## (Intercept)     7.3263379  0.0442065 165.730  < 2e-16 ***
## Administrative  0.0301495  0.0106990   2.818  0.00484 **
## Informational  -0.0165006  0.0268406  -0.615  0.53872
## ProductRelated  0.0103437  0.0007985  12.953  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3.337 on 10530 degrees of freedom
```

```
## Multiple R-squared:  0.02327,    Adjusted R-squared:  0.02299
## F-statistic: 83.61 on 3 and 10530 DF,  p-value: < 2.2e-16
```

As we analyze the summary of the multiple linear regression we noticed that it did not cocnlude with a strong model. The model did not perform as the single linear regression model for explaining product related duration with the feature special day.

Classification with Knn based off feature Revenue. This example of classification was used with the K nearest neighbor method.The point of this study was to be able to correctly classify future data points with the feature of Revenue. K was set to 3 to compare the 3 nearest data points. The commerce dataset was first subsetted to include only the below columns. After that, the subset was broken into samples, to provide for the training of the model and the testing future values. This helps calculate the accuracy of the model.

```
commerce.subset = commerce[c('Revenue','Administrative','Administrative_Duration',
                        'Informational','Informational_Duration','ProductRelated',
                        'ProductRelated_Duration','BounceRates','ExitRates')]
set.seed(123)

ind<-sample(2, nrow(commerce.subset), replace=TRUE,)

commerce.subset.train<- commerce.subset[ind==1, 1:9]
commerce.subset.test<- commerce.subset[ind==2, 1:9]

commerce.trainLabels<-commerce.subset[ind==1,1]
commerce.testLabels<- commerce.subset[ind==2,1]

commerce_pred<-knn(train=commerce.subset.train, test=commerce.subset.test,
                   cl=commerce.trainLabels$Revenue, k=3)
summary(commerce_pred)
```

```
##    0    1
## 4748  506
```

```
CrossTable(x=commerce_pred, y=commerce.testLabels$Revenue,prop.chisq = FALSE)
```

```
##
##
##    Cell Contents
## |-------------------------|
## |                       N |
## |           N / Row Total |
## |           N / Col Total |
## |         N / Table Total |
## |-------------------------|
##
##
## Total Observations in Table:  5254
##
##
##                | commerce.testLabels$Revenue
## commerce_pred |         0 |         1 | Row Total |
## --------------|-----------|-----------|-----------|
##             0 |      3952 |       796 |      4748 |
```

```
##                |     0.832 |     0.168 |     0.904 |
##                |     0.912 |     0.866 |           |
##                |     0.752 |     0.152 |           |
## --------------|-----------|-----------|-----------|
##              1 |       383 |       123 |       506 |
##                |     0.757 |     0.243 |     0.096 |
##                |     0.088 |     0.134 |           |
##                |     0.073 |     0.023 |           |
## --------------|-----------|-----------|-----------|
##  Column Total |      4335 |       919 |      5254 |
##                |     0.825 |     0.175 |           |
## --------------|-----------|-----------|-----------|
##
##
```

```
accuracy=((4646+99)/5931)
accuracy
```

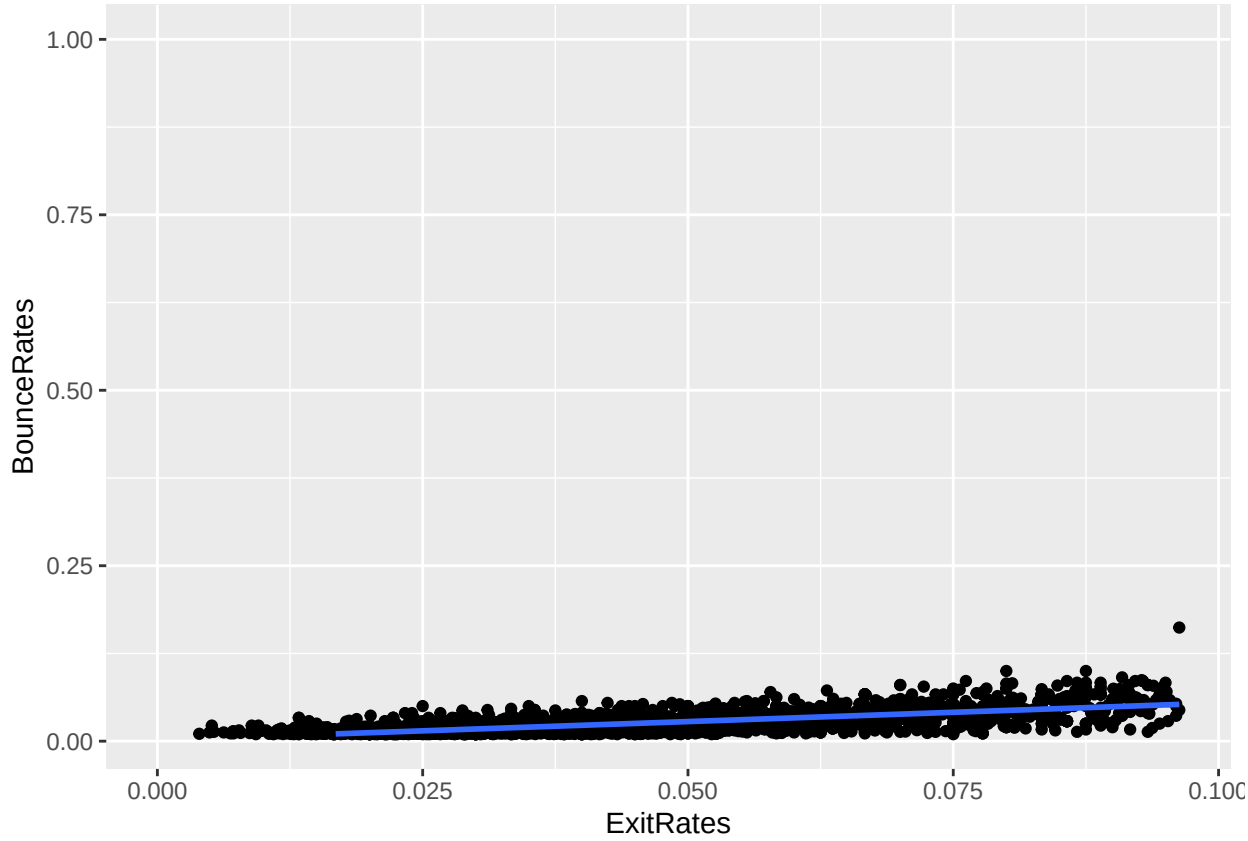
```
## [1] 0.8000337
```

We had an 80% accuracy for predicting Revenue off the classification predition model above. Out of the total 5931 observations, the model correctly precicted 4745 instances. The CrossTable outlines the False and True table outputs. Based off this cross-tabulation we can see how our predictions matched up with the truth.

Linear Regression of Bounce Rates was perfomred using ggplot. This compared the features Exit rates and Bounce rates for the pages. We can see that there is a slight positive relationship between these two features.

```
ggplot(commerce, aes(x=ExitRates, y=BounceRates)) +
  geom_point() +
  stat_smooth(method = "lm") +
  ylim(0.01,1)
```

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




```
lm(BounceRates~ExitRates,commerce)
```

```
##
## Call:
## lm(formula = BounceRates ~ ExitRates, data = commerce)
##
## Coefficients:
## (Intercept)    ExitRates
##   -0.003924     0.440463
```

The linear regression model for bounce rates allows us to predict the bounce rate based off knowing the exit rate for the site. The regression model has an equation of yhat= -.01702+ .91012(x).

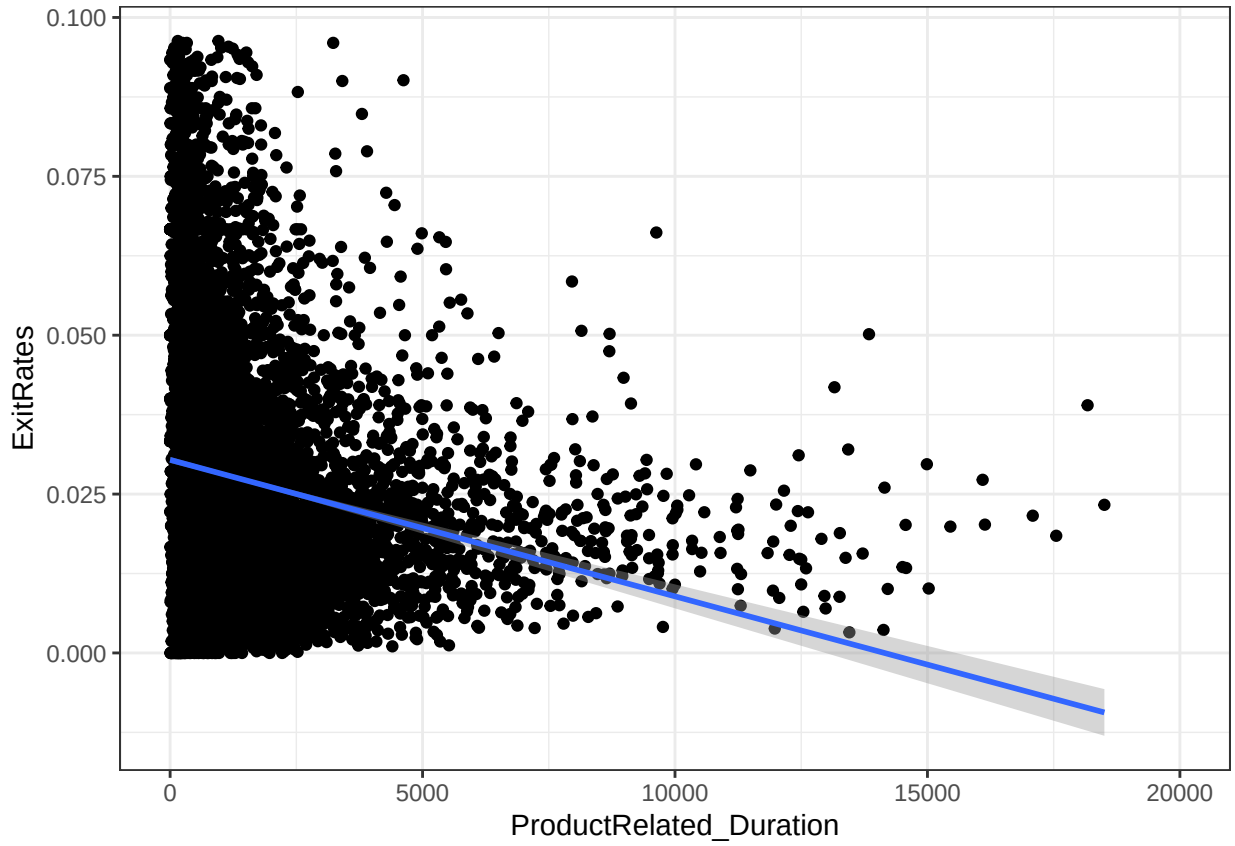
Conducting OLS regression for Exit Rates based off Product Related Duration

```
NROW(commerce.subset$ExitRates)
```

```
## [1] 10534
```

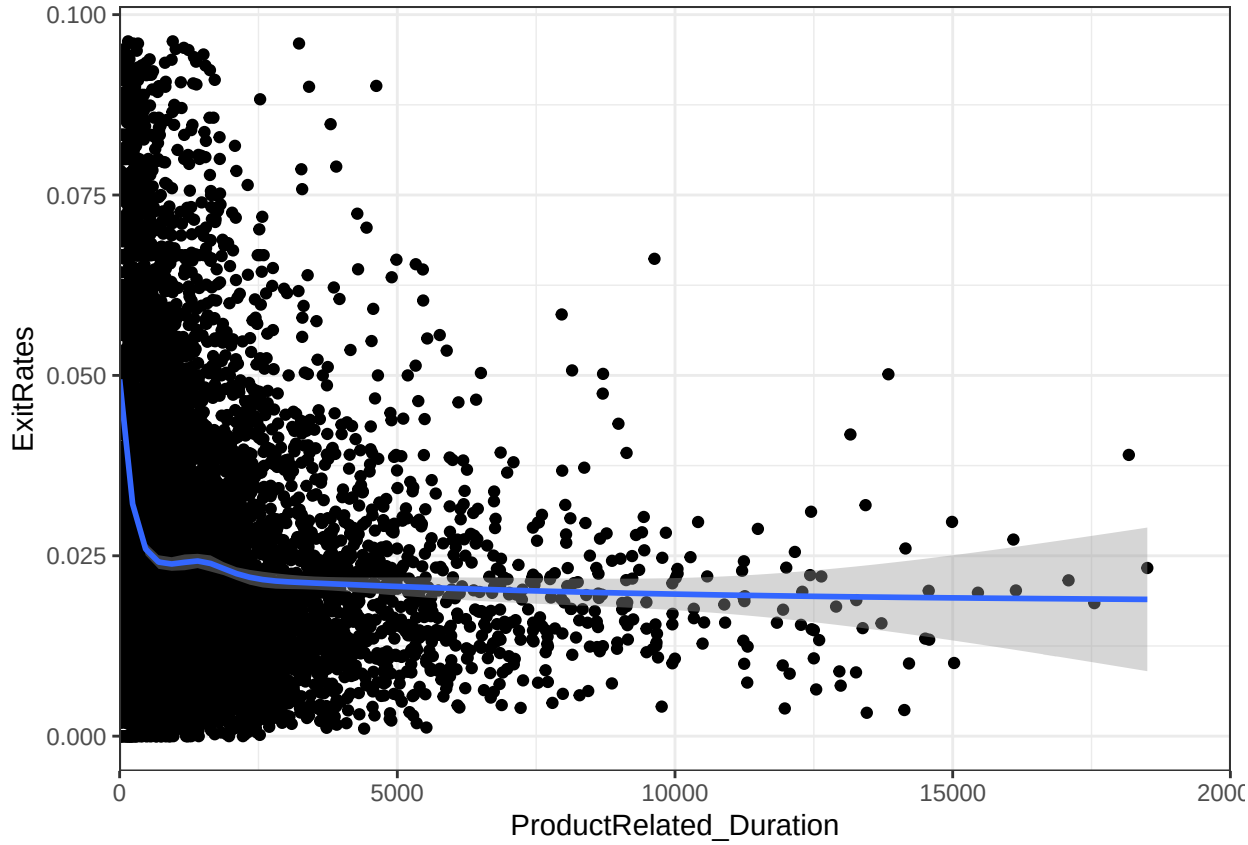
```
ggplot(commerce.subset, aes(x=ProductRelated_Duration, y=ExitRates)) +
  geom_point() +
  geom_smooth(method="lm") +
  scale_x_continuous(limits = c(0,20000), expand=c(0,1000))+theme_bw()
```

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




```
ggplot(commerce.subset, aes(x=ProductRelated_Duration, y=ExitRates)) +
  geom_point() +
  geom_smooth() +
  scale_x_continuous(limits = c(0,20000), expand=c(0,0)) +
  theme_bw()
```

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



When conducting this OLS regression we noticed that a smoothing method of linear did not accurately capture the tail end of the graph. Removing the linear smoothing allowed us to notice that as the product related duration increases we have a larger gap for possible exit rates.

A Naive Bayes model was completed next. This supervised learning method utilizes Bayes theorem to predict the feature Revenue based off the commerce dataset. After installing the e1071 package we then split up the dataset into training and testing samples. 70% of the datset was used as a training set, while the other 30% will be used to test the model.

```
set.seed(123)
id<- sample(2,nrow(commerce), prob = c(0.7, 0.3), replace = T)
ComTrain<- commerce[id==1,]
ComTest<- commerce[id==2,]

comModel<- naiveBayes(Revenue~., data = ComTrain)
print(comModel)
```

```
##
## Naive Bayes Classifier for Discrete Predictors
##
## Call:
## naiveBayes.default(x = X, y = Y, laplace = laplace)
##
## A-priori probabilities:
## Y
##     FALSE      TRUE
```

```
## 0.8449446 0.1550554
##
## Conditional probabilities:
##        Administrative
## Y           [,1]     [,2]
##   FALSE 2.110782 3.188976
##   TRUE  3.415515 3.695793
##
##        Administrative_Duration
## Y            [,1]     [,2]
##   FALSE  73.92779 166.2868
##   TRUE  119.45325 197.8629
##
##        Informational
## Y            [,1]     [,2]
##   FALSE 0.4405920 1.173953
##   TRUE  0.8072197 1.568476
##
##        Informational_Duration
## Y           [,1]     [,2]
##   FALSE 28.53300 126.6268
##   TRUE  60.42354 176.9514
##
##        ProductRelated
## Y           [,1]     [,2]
##   FALSE 28.23707 39.83506
##   TRUE  49.01536 59.03424
##
##        ProductRelated_Duration
## Y           [,1]     [,2]
##   FALSE 1051.820 1638.733
##   TRUE  1913.087 2377.608
##
##        BounceRates
## Y              [,1]       [,2]
##   FALSE 0.025604412 0.05228061
##   TRUE  0.005508098 0.01358107
##
##        ExitRates
## Y             [,1]       [,2]
##   FALSE 0.04770508 0.05160871
##   TRUE  0.01934156 0.01728922
##
##        PageValues
## Y            [,1]     [,2]
##   FALSE  2.011944  9.34226
##   TRUE  27.692848 35.35592
##
##        SpecialDay
## Y             [,1]      [,2]
##   FALSE 0.06830162 0.2073029
##   TRUE  0.02350230 0.1248545
##
##        Month
```

```
## Y               Aug         Dec         Feb         Jul        June         Mar
##   FALSE 0.034813249 0.140944327 0.018181818 0.035377026 0.026215645 0.164059197
##   TRUE  0.043778802 0.111367127 0.001536098 0.033026114 0.015360983 0.105222734
##        Month
## Y               May         Nov         Oct         Sep
##   FALSE 0.284566596 0.218886540 0.041437632 0.035517970
##   TRUE  0.185099846 0.407066052 0.053763441 0.043778802
##
##        VisitorType
## Y       New_Visitor       Other Returning_Visitor
##   FALSE 0.124594785 0.007329105       0.868076110
##   TRUE  0.215053763 0.009216590       0.775729647
##
##        Weekend
## Y           FALSE      TRUE
##   FALSE 0.7692741 0.2307259
##   TRUE  0.7434716 0.2565284
##
##        MonthNum
## Y           [,1]     [,2]
##   FALSE 7.467935 3.406210
##   TRUE  8.663594 3.117314
##
##        PageValueRank
## Y       High Page Value Low Page Value Medium Page Value No Purchase
##   FALSE      0.00000000     0.00000000        0.00000000  1.00000000
##   TRUE       0.06374808     0.75115207        0.18509985  0.00000000
```

```
prediction<-predict(comModel, newdata = ComTest)
#print(prediction)

table(prediction, ComTest$Revenue)
```

```
##
## prediction FALSE TRUE
##      FALSE  2837   15
##      TRUE    165  534
```

```
accuracy=((2837+534)/nrow(ComTest))
accuracy
```

```
## [1] 0.9493101
```

```
nrow(ComTest)
```

```
## [1] 3551
```

The output of the comModel contained a likelihood table as well as a-prioriprobabilities. This showed the frequency of the features, which would help us predict which predictors we would anticipate seeing in future. The prediction of Revenue will be influenced by the predictors in the dataset as well as their prevalence. To compare the accuracy of the model we created a table to display the correct predictions of Revenue from the test dataset. Here the accuracy was roughly 95%. Based on the table, the model correctly predicted 3371 of the 3551 testing data points.This is a highly reliable model utilizing Bayes theorem.