```
library(devtools)      library(ggbiplot)      library(mlbench)      library(tidyverse)
library(caret)         library(MASS)          library(stats)        library(dplyr)
library(factoextra)    library(Amelia)        library(mice)         library(VIM)
library(magrittr)  #library(plyr)        library(naniar)       library(lubridate)
```

1. Exploratory Data Analysis

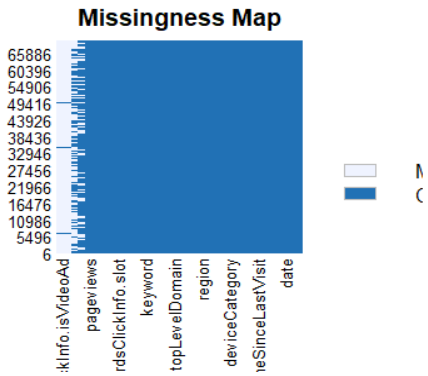
```
train_metric = read.csv(file = 'Train.csv')
df_status(train_metric)
```

this shows the number of 0s and 1s that the data has, the number and percentage of the null values in the data set.

variable	q_zeros	p_zeros	q_na	p_na	q_inf	p_inf	type	unique
sessionId	0	0	0	0	0	0	numeric	70071
custId	0	0	0	0	0	0	integer	47249
date	0	0	0	0	0	0	character	366
channelGr	0	0	0	0	0	0	character	8
visitStartT	0	0	0	0	0	0	integer	69951
visitNumb	0	0	0	0	0	0	integer	155
timeSince	47249	67.43	0	0	0	0	integer	20970
browser	0	0	0	0	0	0	character	28
operating	0	0	0	0	0	0	character	16
isMobile	53995	77.05	0	0	0	0	integer	2
deviceCat	0	0	0	0	0	0	character	3
continent	0	0	0	0	0	0	character	6
subContin	0	0	0	0	0	0	character	23
country	0	0	0	0	0	0	character	177
region	0	0	0	0	0	0	character	310
metro	0	0	0	0	0	0	character	73
city	0	0	0	0	0	0	character	478
networkDo	0	0	0	0	0	0	character	5015
topLevelD	0	0	0	0	0	0	character	184
campaign	0	0	0	0	0	0	character	7
source	0	0	0	0	0	0	character	132
medium	0	0	0	0	0	0	character	6
keyword	0	0	0	0	0	0	character	416
isTrueDire	42026	59.98	0	0	0	0	integer	2
referralPa	0	0	0	0	0	0	character	384
adContent	0	0	0	0	0	0	character	28
adwordsC	0	0	68260	97.42	0	0	integer	5
adwordsC	0	0	0	0	0	0	character	3
adwordsC	0	0	0	0	0	0	character	1406
adwordsC	0	0	0	0	0	0	character	2
adwordsC	1811	2.58	68260	97.42	0	0	logical	1
pageviews	0	0	8	0.01	0	0	integer	154
bounces	0	0	40719	58.11	0	0	integer	1
newVisits	0	0	23944	34.17	0	0	integer	1
revenue	64222	91.65	0	0	0	0	numeric	5850

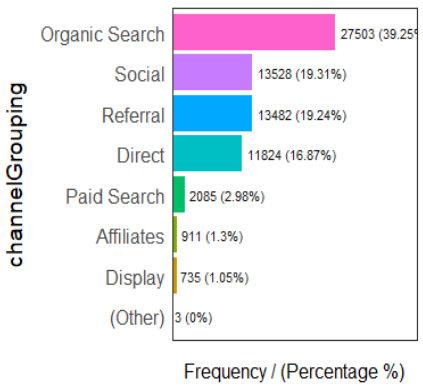
From this table, it is seen that a majority of the variables in the dataset are of type character, implying categorical variables. The revenue has a 91.65% of values with 0, going to show that the purchase from the site are from a little subset of the site visits.

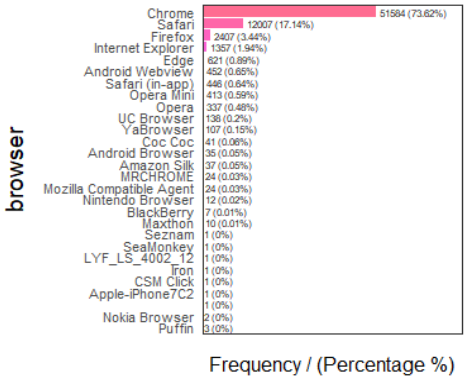
a) missmap(data, legend = TRUE) #missingness map
b) gg_miss_var(data, show_pct = TRUE) #looking at the % of missingness for each variable

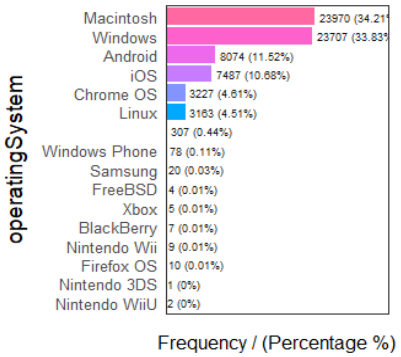


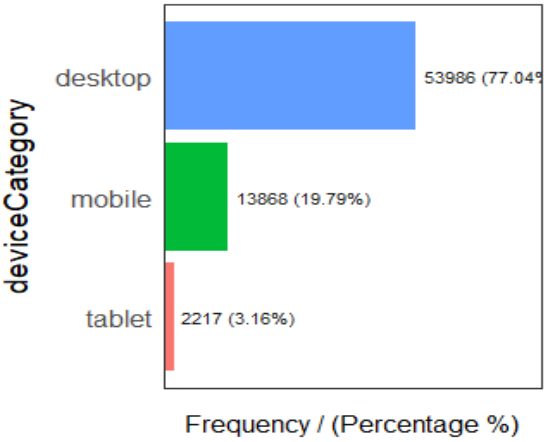
#From the maps, it is seen that variables that had the highest percentage of missingness are related to “adwords”. This influenced the decision to exclude these variables from the predictor variables. Bounce and new visits with missing values were imputed accordingly as they were considered necessary for prediction.

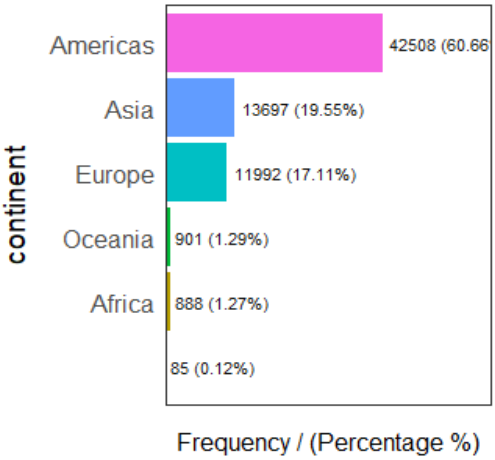
```
profiling_num(train_metric) #stastitic distribution of the data and the quantile ranges of the data.
freq(train_metric) #shows the frequency of each of some of the category variables and barcharts are also plotted to show the counts of each variable.
```

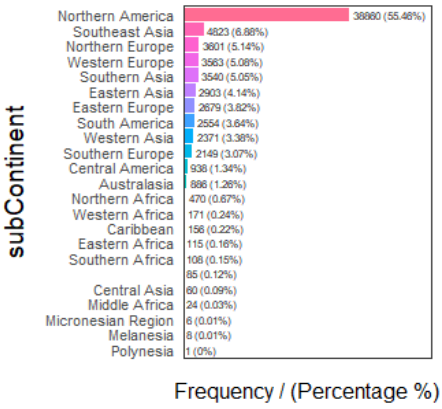


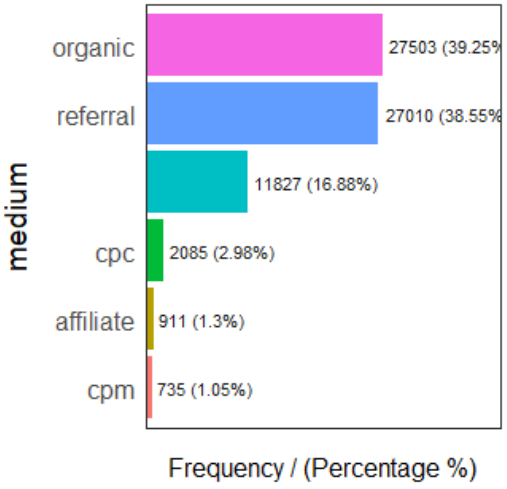


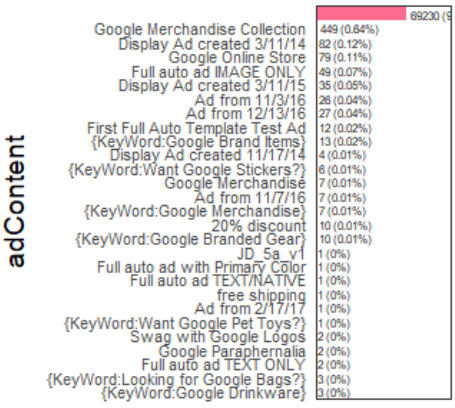


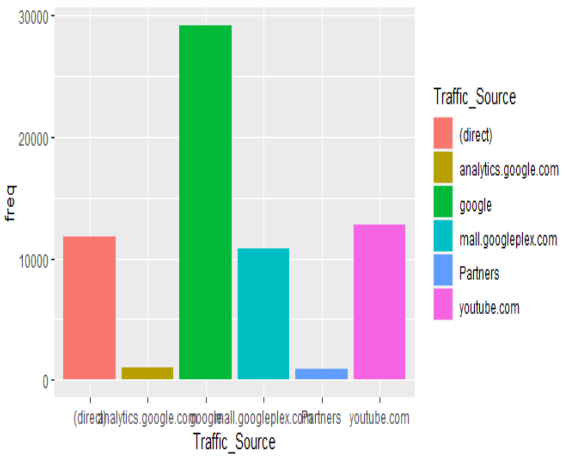








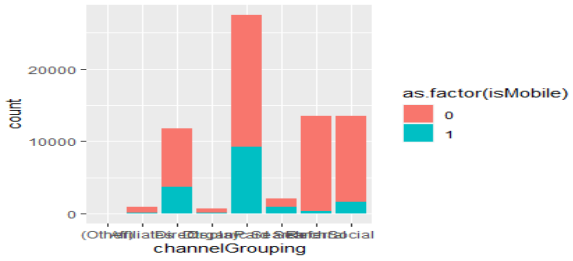




It is seen that organic search has the highest frequency of visits to the site. Implying that the highest traffic from the site comes through a search engine. The code below is used to generate a plot for the top 5 sources. As is seen, google is the traffic source with the highest frequency

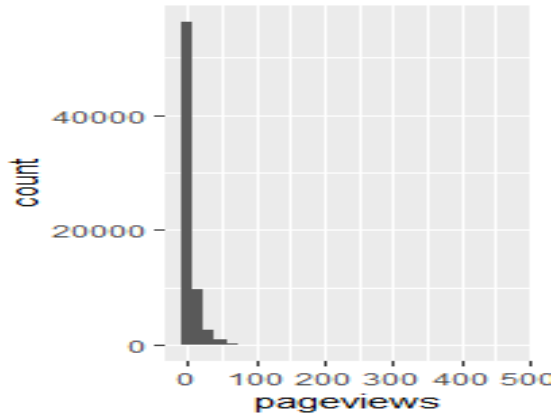
```
source = count(train_metric,'source') # count the number of occurrences
source = data.frame(source) # converting to a data frame
```

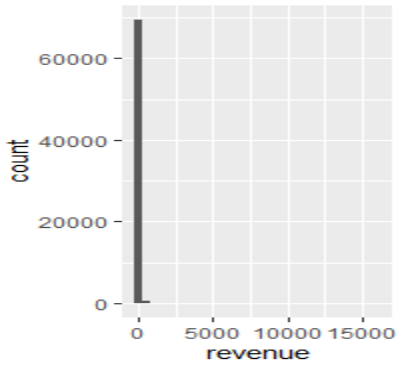
```
source = arrange(source, desc(freq)) # arrange in descending order
source = head (source)
names(source)[1] <- "Traffic_Source"
names(source)[2] <- "freq"
ggplot(source)+ geom_bar(mapping = aes(x = Traffic_Source, y = freq, fill = Traffic_Source), stat =
"identity")
```

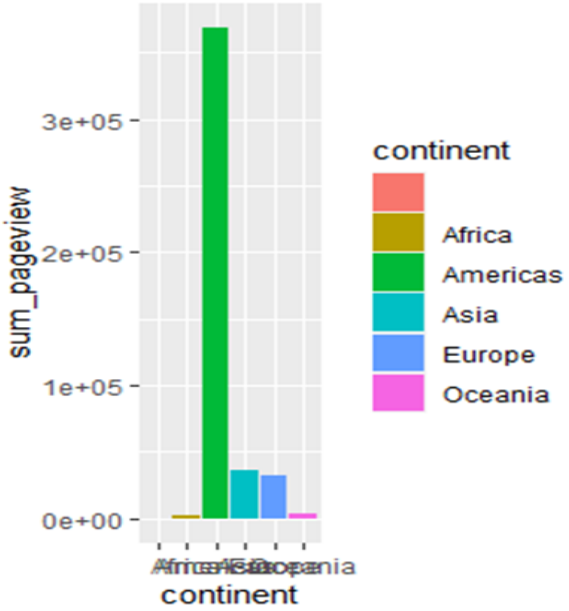



```
profiling_num(train_metric)  #stastitic distribution of the data and the quantile ranges of the data.
ggplot(train_metric) + geom_bar(mapping = aes(x = channelGrouping, fill = as.factor(isMobile)))
#shows the distribution of the channel grouping and the mobile variable, majority use computers than mobile
ggplot(train_metric) + geom_histogram(mapping = aes(x = pageviews))
ggplot(train_metric) + geom_histogram(mapping = aes(x = revenue))
```

#the revenue distribution is skewed with a high value. This goes to show that most purchases from the site are from a few individuals, and that the page views are few (below 100) for most of the views.







ggplot(train_metric) + geom_bar(mapping = aes(x = continent, fill = continent)) #africa has the lowest distribution, as shown in page 2

amer = train_metric %>%filter(continent == "Americas")
data_pageview = train_metric %>% group_by(continent) %>% summarise(sum_pageview = sum(pageviews, na.rm = TRUE)) %>% arrange(desc(sum_pageview))

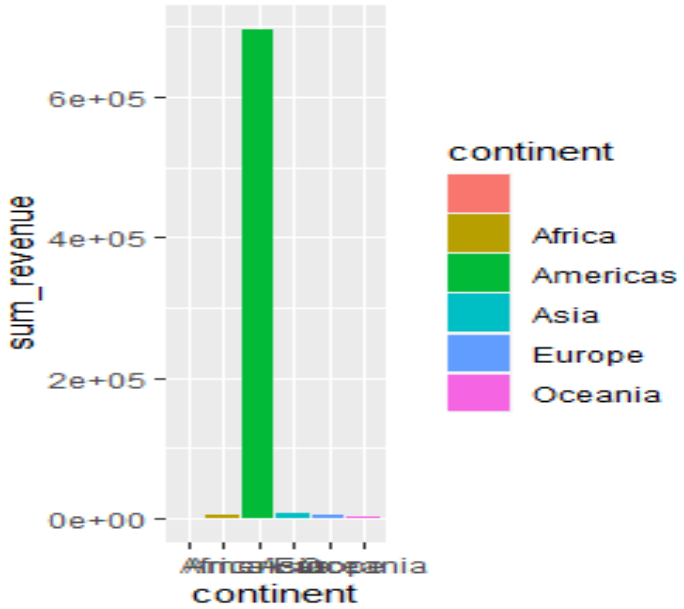
ggplot(data = data_pageview, aes(x= continent, y=sum_pageview, fill = continent)) + geom_bar(stat="identity")

#America has the highest view and Africa has the lowest views.

data_rev = train_metric %>% group_by(continent) %>% summarise(sum_revenue = sum(revenue)) %>% arrange(desc(sum_revenue))

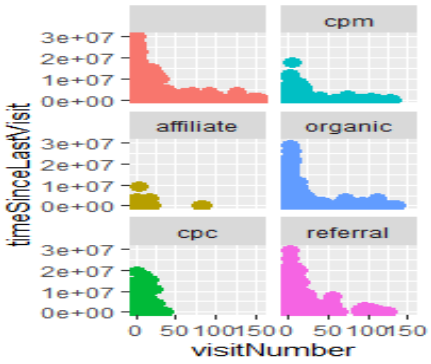
ggplot(data = data_rev, aes(x= continent, y=sum_revenue, fill = continent)) + geom_bar(stat="identity")

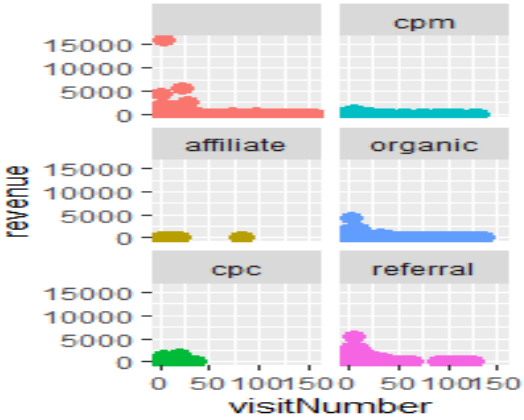
#the highest revenues are from america and the next is from asia and then africa., as shown below



Although Africa has the least page views from the continents, a closer look at the distribution shows that Africa has the third highest revenue from the continents. This implies that a good percentage of visits from Africa results in purchases.

```
ggplot(train_metric , aes(x=visitNumber, y= timeSinceLastVisit, color=medium))+ geom_point(size=3) +
  facet_wrap(~medium , dir="v") +theme(legend.position="none")
# there doesn't seem to be a good trend in comparing visit number with timeSinceLastVisit
```



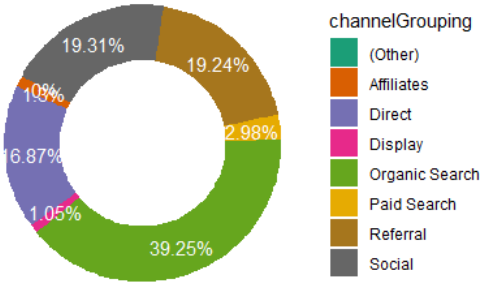



```
ggplot(train_metric , aes(x=visitNumber, y=
revenue, color=medium))+geom_point(size=3) +
facet_wrap(~medium , dir="v") + theme(legend.position="none")# there doesn't seem to be a good trend
in comparing visit number with timeSinceLastVisit
```

1b) Data Preparation

```
# analyzing the variables and grouping them with respect to the total revenue obtained from the
predictors. i.e. AGGREGATION. Aggregation gives an idea of the variables that contribute to total revenue
and would be important for selecting our predictor variables

channelGrouping = train_metric %>% group_by(channelGrouping) %>% dplyr::summarize(freq = n())
channelGrouping_rev = aggregate(train_metric $revenue, by = list(train_metric $channelGrouping), FUN
= sum)
channelGrouping_rev$x = round(channelGrouping_rev$x)

channelGrouping = data.frame(channelGrouping_rev, channelGrouping)
#colnames(channelGrouping)
#renaming the columns
names(channelGrouping)[1]= "channelGrouping"
names(channelGrouping)[2]= "Tot. Rev"
names(channelGrouping)[4]= "freq"
channelGrouping = channelGrouping[,c(1,2,4)]
channelGrouping %>% arrange(desc(`Tot. Rev`))

channelGrouping$f_fraction = channelGrouping$freq / sum(channelGrouping$freq) #computing %
channelGrouping$f_ymax = cumsum(channelGrouping$f_fraction) # computing the cum %
channelGrouping$f_ymin = c(0, head(channelGrouping$f_ymax, n = -1))
channelGrouping$f_labelPosition = (channelGrouping$f_ymax + channelGrouping$f_ymin )/2

# Compute a good label this is for a pie chart plot
```

```
channelGrouping$f_label <- paste0(channelGrouping$channelGrouping, "\n value: ",
channelGrouping$freq)

c_grp = train_metric %>%  filter(channelGrouping != "NA") %>% group_by(channelGrouping) %>%
  count() %>%  ungroup()%>% arrange(desc(channelGrouping)) %>% mutate(percentage =
round(freq/sum(freq),4)*100, lab.pos = cumsum(percentage)-.5*percentage)

# a donut chart plot to see the data with the channel with highest traffic to the site
ggplot(data = c_grp, aes(x = 2, y = percentage, fill = channelGrouping))+
  geom_bar(stat = "identity") + coord_polar("y", start = 200) +
  geom_text(aes(y = lab.pos, label = paste(percentage,"%", sep = "")), col = "white") +
  theme_void() + scale_fill_brewer(palette = "Dark2")+ xlim(.2,2.5)
```




```
# Another type of visualization, showing the percentage of frequency of channel grouping. This
visualization could be used in place of histograms, to see the percentages of the distribution

#Agregating for browser
brows_rev = aggregate(train_metric $revenue, by = list(data$browser), FUN = sum)
browser_NA <- aggregate((train_metric),by=list(train_metric $browser),function(x) sum(is.na(x)))
# trying to check if browser has empty values
browser = train_metric %>% group_by(browser) %>% dplyr::summarize(n = n())
browser = data.frame(brows_rev, browser)
colnames(browser)
#renaming the columns
names(browser)[1]= "browser"
names(browser)[2]= "Tot. Rev"
names(browser)[4]= "freq"
browser = browser[,c(1,2,4)]
browser %>% arrange(desc(`Tot. Rev`))
# from the analysis browser is an important variable, as is seen chrome has the highest frequency of
occurrence and shows the highest total revenue

#Analysis for operating system
os_rev = aggregate(train_metric $revenue, by = list(train_metric $operatingSystem), FUN = sum)
```

```
os = train_metric %>% group_by(operatingSystem) %>% dplyr::summarize(n = n())
os = data.frame(os_rev, os)
colnames(os)
#renaming the columns
names(os)[1]= "operatingSystem"
names(os)[2]= "Tot. Rev"
names(os)[4]= "freq"
os = os[,c(1,2,4)]
os = os %>% arrange(desc(`Tot. Rev`))
# from the analysis it is seen that no revenue comes from individuals who use devices other than computers. These analyses provide information that we can delete 2 of either isMobile, deviceCategory or operatingSystem.

medium = train_metric %>%  group_by(medium) %>%dplyr:: summarize(n = n())
medium_rev = aggregate(train_metric$revenue, by = list(train_metric$medium), FUN = sum)
# medium has a lot of missing values, so it could be removed
medium = data.frame(medium_rev, medium)
colnames(browser)
#renaming the columns
names(medium)[1]= "medium"
names(medium)[2]= "Tot. Rev"
names(medium)[4]= "freq"
medium = medium[,c(1,2,4)]
medium = medium %>% arrange(desc(`Tot. Rev`))

continent = train_metric %>% group_by(continent) %>% summarize(n = n())
continent_rev = aggregate(train_metric $revenue, by = list(train_metric $continent), FUN = sum)
continent = data.frame(continent_rev, continent)
colnames(continent)
#renaming the columns
names(continent)[1]= "continent"
names(continent)[2]= "Tot. Rev"
names(continent)[4]= "freq"
continent = continent[,c(1,2,4)]
continent = continent %>% arrange(desc(`Tot. Rev`))

country = train_metric %>%  group_by(country) %>% dplyr::summarize(n = n())
country_rev = aggregate(train_metric$revenue, by = list(train_metric$country), FUN = sum)
country = data.frame(country_rev, country)
colnames(country)
#renaming the columns
names(country)[1]= "country"
names(country)[2]= "Tot. Rev"
names(country)[4]= "freq"
```

```
country = country[,c(1,2,4)]
country = country %>% arrange(desc(`Tot. Rev`))
```

This analysis is continued for the variables to guide the selection of appropriate variables to use as predictor variables. The dataframes below show the ranking of the variables as a function of total revenue

	operatingSystem	Tot. Rev	freq
1	Macintosh	408606.87	23970
2	Windows	156998.90	23707
3	Chrome OS	96635.58	3227
4	Linux	24423.63	3163
5	Android	14241.00	8074
6	iOS	11373.20	7487

	browser	Tot. Rev	freq
1	Chrome	6.708464e+05	51584
2	Safari	2.927524e+04	12007
3	Firefox	5.442832e+03	2407
4	Edge	3.978797e+03	621
5	Internet Explorer	2.453603e+03	1357
6	Opera	1.536751e+02	337
		8.874722e+01	446

	medium	Tot. Rev	freq
1	referral	327746.0900	27010
2		194597.5996	11827
3	organic	156004.9981	27503
4	cpc	26033.5669	2085
5	cpm	7819.1233	735
6	affiliate	77.8137	911

	country	Tot. Rev	freq
1	United States	6.658513e+05	36941
2	Canada	1.870389e+04	1918
3	Venezuela	8.371449e+03	231
4	Nigeria	3.298414e+03	107
5	Japan	1.874781e+03	1116
6	Kenya	1.588298e+03	40
7	Indonesia	1.296318e+03	624
8	Puerto Rico	1.081203e+03	55
9	Australia	1.035070e+03	755

	continent	Tot. Rev	freq
1	Americas	696099.25022	42508
2	Asia	6785.61943	13697
3	Africa	4886.71200	888
4	Europe	3411.20418	11992
5	Oceania	1083.06731	901
6		13.33848	85

Macintosh, chrome, referral and united states are categories that generarte the most revenue. Africa being the third continent with the most revenue, shows that Nigeria is the 4th country with the most revenue over 170+ countries.

DATA WRANGLING
#creating new variables
the date was split into year, month, and day. As the month could probably influence modeling results

```
train_metric = train_metric %>% separate (sessionId, into= c("Customer", "number of visit"), sep = 5)%>%
separate (date, into = c("year", "month", "day"))
train_metric_new = train_metric[, c(1,2,3,4,5,6,7,10,11,12,13,15,16,17,24,25,27,35,36,37,38)]
```

#MISSING VALUE IMPUTATION.
The NA here were changed to 0 as this means they are bounced

```
train_metric_new$bounces[is.na(train_metric_new$bounces)]=0
```

The NA were converted to zero because NA here means this is not their first visit to the site

```
train_metric_new$newVisits[is.na(train_metric_new$newVisits)]=0
```

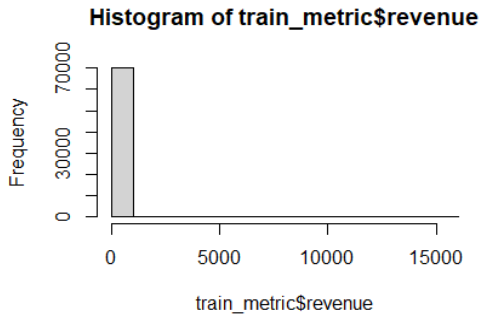
#OUTLIER

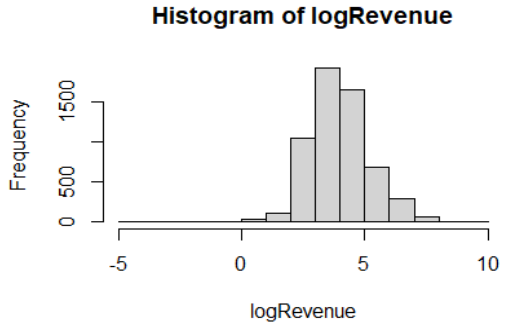
```
grubbs.test(train_metric$revenue)  #univariate test for 'revenue' outliers
outlier(train_metric$revenue)   #univariate test for 'revenue' outlier
```

There is an outlier in revenue, but because revenue is the outcome variable, this outlier was not removed, as it is considered important

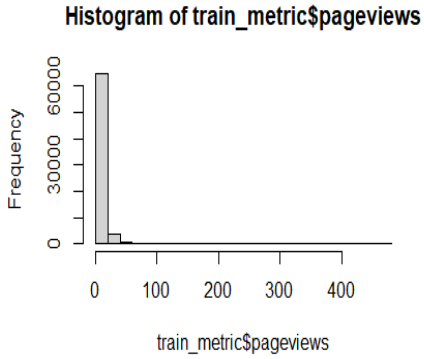
FEATURE TRANSFORMATION. This was conducted on the numeric variables, revenue and custpage views by taking the log of the variable +1. This is implemented during the modeling.

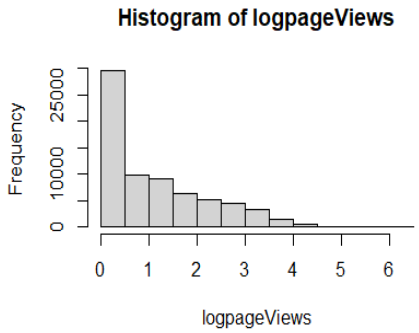
```
hist (train_metric$revenue) # looking at the distribution of revenue before transformation
logRevenue = log(train_metric$revenue)
hist (logRevenue) # log transformation of revenue variable
```

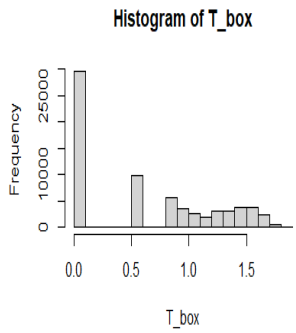





```
hist (train_metric$pageviews) # looking at the distribution of pageViews before transformation
logpageViews = log(train_metric$pageviews)
hist (logpageViews) # log transformation of pageViews variable

# Using the boxcox function for transformation of customer page views
Box = boxcox(train_metric$pageviews~1, lambda = seq(-3,3,0.1))
cox  = data.frame(Box$x, Box$y)
cox2 = cox[with(cox, order(-cox$Box.y)),]
cox2[1,]
lambda = cox2[1,"Box.x"]
lambda
T_box = (train_metric$pageviews^ (lambda) - 1)/(lambda)
hist(T_box) # distribution after transformation of custpageviews variable
```







From the transformed distribution for customer page reviews, there is not much between log transformation and optimizing for lambda, using the box cox function, so we decided to stick with the log transform.

c. The initial dataset given was untidy as a single observation (custId) was shown on different rows and the data had to be cleaned up. This included the lumping of these rows of the same variables together. The revenue and pageviews are added together but these is not possible with categorical variables. The most frequent variable was chosen as the category for the custId.

#this was done for all the categorical variables considered.

```
train_metric_cont = train_metric_new %>% group_by(custId, continent) %>%
  summarise (c = n()) %>%
  filter (row_number(desc(c))==1) %>%
  rename (mostfreqCont = continent)
```

#sum of revenue and pagevisits

```
train_data = train_metric_new %>% group_by(custId) %>%
  summarise(custRevenue = sum(revenue), custpageviews = sum(pageviews))
test_data = test_metric_new %>% group_by(custId) %>%
  summarise(custpageviews = sum(pageviews))
```

The data was first trained for with all the countries and the browsers in the dataset. There was a total of over 150 countries and browsers in the data. There was an error was the trained model was applied on the test set because there were new factor levels in the test data that was unavailable in the train set.

The corrective step was to aggregate all country into the top 10 and the browsers into the top 9 to ensure the test and train data set had the same test and train factor levels.

```
 train_data$mostfreqCountry = fct_lump_n(train_data$mostfreqCountry, 10 ,
                                         other_level = "Other")
train_data$mostfreqBroswer = fct_lump_n(train_data$mostfreqBroswer, 9 ,
                                         other_level = "Other")
```

The custpageviews was transformed to get a histogram with a normal distribution. The data was run without this transformation and with the transformation, the model with the transformation gave a better model. This is because of the initially skewed distribution of the pageviews.

```
total_test$custpageviews = log((total_test$custpageviews)+1)
train_data$custpageviews = log((train_data$custpageviews)+1)
```

The total revenue of each customer was converted to a logarithm value. This helps with the transformation of the highly skewed dataset.

```
train_data = total %>%
  mutate(total, targetRevenue = log((custRevenue+1)))
```

The lasso and ridge regression model weren't accepting the categorical variables. The way out of this was to convert the categorical variables to numeric by using dummy variables in the "funmodelling" package. The result was then loaded into a matrix and piped into the lasso and ridge regression analysis.

#this was also done for the test set.

```
dataf <- dummy_cols(train_data, select_columns = c('mostfreqCountry', 'mostfreqchangroup',
'mostfreqMonth', 'mostfreqBroswer', 'mostfreqCont'))
```

Multicollinearity of the data was done to check how related to each other the variables in the linerar regression were. The highly collinear variables were removed from the analysis such as the "continent" and the "operating system".

```
car::vif(model1)
```

Check the residual plot of the model to determine the behavior of the residual. An iterative process was done at the stage to get the model that gives the best residual plot.

```
plot(model1)
```

The custpageviews are was had some NA values which was causing errors in the model prediction. This variable was checked for NA values and 10 NAs were detected in the model. The resolution of the missing values was done by inputs with 0. And this resolved the missing value errors in the model prediction.

d. Modelling

OLS

```
model1 = lm(data = train_data, targetRevenue ~ custpageviews
        + mostfreqBroswer + mostfreqMonth + mostfreqCountry +mostfreqCont
        + mostfreqchangroup + mostfreqMobile +
          mostfreqnewbounces + mostfreqnewvisit)

total_test$custpageviews = log((total_test$custpageviews)+1)
plot(model1)

predict_rev = predict(model1, total_test)

submissionDf = data.frame(custId=total_test$custId, predRevenue = predict_rev)

summary(model1)

#try to see the data collienarity
car::vif(model1)
write.csv(submissionDf, "submission1.csv", row.names=FALSE)
```

Ridge Regression

```
train.ridge <- lm.ridge(targetRevenue ~ custpageviews + mostfreqCountry +
                  mostfreqchangroup + mostfreqMonth + mostfreqMobile +
                  mostfreqBroswer +
                  mostfreqCont + mostfreqnewbounces + mostfreqnewvisit,
                data=train_data,
                lambda = seq(0,30,0.1))
```

```
td = tidy(train.ridge)
head(td)
g = glance(train.ridge)
g
methods(class = 'ridgelm')
select(train.ridge)
const = as.numeric(names(which.min(train.ridge$GCV)))
const
total_test_new = total_test[,-c(1,8)]

dataf <- dummy_cols(total_test_new, select_columns = c('mostfreqCountry',
                                    'mostfreqchangroup',
                                    'mostfreqMonth',
                                     'mostfreqBroswer',
                                    'mostfreqCont'))
dataf = dataf[, -c(2,3,4,6,8)]
new  = dataf[,-c(46,5,16,24,36)]

predicted  = train.ridge$ym  + scale(new, center = train.ridge$xm, scale =
                              train.ridge$scales) %*%
  train.ridge$coef[,which.min(train.ridge$GCV)]

#as.numeric(predicted)

submissionDf2 = data.frame(custId=data_test$custId, predRevenue = predicted)
write.csv(submissionDf2, "submission2.csv", row.names=FALSE)

#ridge, lasso, elastic net
total_test1 <- dummy_cols(total_test, select_columns = c('mostfreqCountry',
                                      'mostfreqchangroup',
                                      'mostfreqMonth',
                                      'mostfreqBroswer',
                                      'mostfreqCont'))

y = total_test$custId
dataf <- dummy_cols(train_data, select_columns = c('mostfreqCountry',
                                    'mostfreqchangroup',
                                    'mostfreqMonth',
                                    'mostfreqBroswer',
                                    'mostfreqCont'))

dataf1 = dataf[,-c(1,2,4,5,6,7,8,9,10,13)]
total_test1  =total_test1[,-c(1,3,4,5,6,7,8,10,13)]

train_cont <- trainControl(method = "repeatedcv",
                     number = 10,
                     repeats = 5,
                     search = "random",
```

```
                    verboseIter = TRUE)
```

ELASTIC

```
elastic_reg <- train(targetRevenue ~  custpageviews +
                       mostfreqCountry +
                       mostfreqchangroup + mostfreqMonth + mostfreqMobile +
                       mostfreqBroswer +
                       mostfreqCont + mostfreqnewbounces + mostfreqnewvisit,
                     data = train_data,
                     method = "glmnet",
                     preProcess = c("center", "scale"),
                     tuneLength = 10,
                     trControl = train_cont)

# Best tuning parameter
elastic_reg$bestTune

# Make predictions on training set
predictions_train <- predict(elastic_reg, train_data)

eval_results = function(true, predicted, df) {
  SSE <- sum((predicted - true)^2)
  SST <- sum((true - mean(true))^2)
  R_square <- 1 - SSE / SST
  RMSE = sqrt(SSE/nrow(df))

  # Model performance metrics
  data.frame(
    RMSE = RMSE,
    Rsquare = R_square)
}

eval_results(train_data$targetRevenue, predictions_train, train_data)

predictions_test <- predict(elastic_reg, total_test)
submissionDf3 = data.frame(custId=y, predRevenue = predictions_test)
write.csv(submissionDf3, "submission3.csv", row.names=FALSE)
```

PLS

```
model <- train(targetRevenue ~ log(custpageviews+1) +
                 mostfreqCountry +
                 mostfreqchangroup + mostfreqMonth + mostfreqMobile +
                 mostfreqBroswer +
                 mostfreqCont + mostfreqnewbounces + mostfreqnewvisit,
               data = train_data, method = "pls",
               scale = TRUE,
               trControl = trainControl("cv", number = 10),
               tuneLength = 10)
```

```
predictions_test <- predict(model, total_test)
predictions_train <- predict(model, train_data)

eval_results(train_data$targetRevenue, predictions_train, train_data)
submissionDf5 = data.frame(custId=total_test$custId, predRevenue = predictions_test)
write.csv(submissionDf5, "submission5.csv", row.names=FALSE)
```

LASSO

```
dataf1 = dataf1[,-5]
x = as.matrix(dataf1)
x_test = as.matrix(total_test1)

y_train = train_data$targetRevenue

lambdas <- 10^seq(2, -3, by = -.1)
lasso_reg <- cv.glmnet(x, y_train, alpha = 1, lambda = lambdas, standardize = TRUE, nfolds = 5)
optimal_lambda <- lasso_reg$lambda.min
optimal_lambda

view(dataf1)
# Prediction and evaluation on train data
predictions_train <- predict(lasso_reg, s = optimal_lambda, newx = x)
eval_results(y_train, predictions_train, train_data)

# Prediction and evaluation on test data
predictions_test <- predict(lasso_reg, s = optimal_lambda, newx = x_test)
submissionDf3 = data.frame(custId=total_test$custId, predRevenue = predictions_test)

write.csv(submissionDf3, "submission3.csv", row.names=FALSE)
```

RIDGE

```
ridge_cv <- cv.glmnet(x, y_train, alpha = 0, lambda = lambdas, standardize = TRUE, nfolds = 5)
optimal_lambda <- ridge_reg$lambda.min
optimal_lambda

# Prediction and evaluation on train data
predictions_train <- predict(ridge_cv, s = optimal_lambda, newx = x)
eval_results(y_train, predictions_train, train_data)

# Prediction and evaluation on test data
predictions_test <- predict(ridge_cv, s = optimal_lambda, newx = x_test)
submissionDf4 = data.frame(custId=total_test$custId, predRevenue = predictions_test)
write.csv(submissionDf4, "submission4.csv", row.names=FALSE)
```

RLM

```
rlm_model = rlm(targetRevenue ~ custpageviews +
                mostfreqCountry +
                mostfreqchangroup + mostfreqMonth + mostfreqMobile +
```

```
    mostfreqBroswer +
    mostfreqCont + mostfreqnewbounces + mostfreqnewvisit, train_data,
  psi = psi.bisquare)

summary(rlm_model)
predictions_train <- predict(rlm_model, train_data)
predictions_train
#regr.eval(train_data$targetRevenue, predictions_train)
predictions_test <- predict(rlm_model, total_test)

submissionDf6 = data.frame(custId=total_test$custId, predRevenue = predictions_test)

write.csv(submissionDf6, "submission6.csv", row.names=FALSE)
```

SVM

```
model_svm = svm(targetRevenue~ custpageviews + mostfreqMonth+ mostfreqCountry,
        data = train_data)
svm_test = predict(model_svm, total_test)
submissionDf7 = data.frame(custId=total_test$custId, predRevenue = svm_test)
write.csv(submissionDf7, "submission7.csv", row.names=FALSE)
```

Model	Method	Package	Hyperparameter	Selection	R^2	RMSE
OLS	lm	stats	NA	NA	0.5868	0.8418
lasso	lasso	glmnet	lambda	0.001	0.5849	0.8437
ridge	rigde	glmnet	lambda	0.001	0.5932	0.8453
Elastic	elasticnent	glmnet	alpha	0.835	0.5871	0.8414
PLS	pls	caret	cv	10	0.587	0.84
SVM	eps regression	e1071	Gamma	0.1	0.586	0.845

d. The regularized regression models are performing better than the linear regression model. Overall, all the models are performing well with decent R-squared and stable RMSE values. The most ideal result would be an RMSE value of zero and R-squared value of 1, but that's almost impossible in real economic datasets. The date the observations were recorded were split into day, month and day and the month was used in the analysis as the month the observations were made can affect the purchase. The countries and browsers were aggregated to get the top 10 variables. This is because the number of countries are large and can affect the prediction of the regression models due to multicollinearity. The collinearity of the model was determined and the highly correlated data was removed.

Elastic net regression combines the properties of ridge and lasso regression. The last line of code prints the optimum values, which come out to be 0.8353904 for alpha and 0.001614618 for lambda. The elastic net picks an optimum alpha value between 0 and 1. Repeated cross validation was done on the train data set and a better accuracy was observed. We will build our model on the training set and evaluate its performance on the test set. This is called the cross validation approach for evaluating model performance.

The categorical variables gave while passing into some models, we had to change them inot dummy variables first.