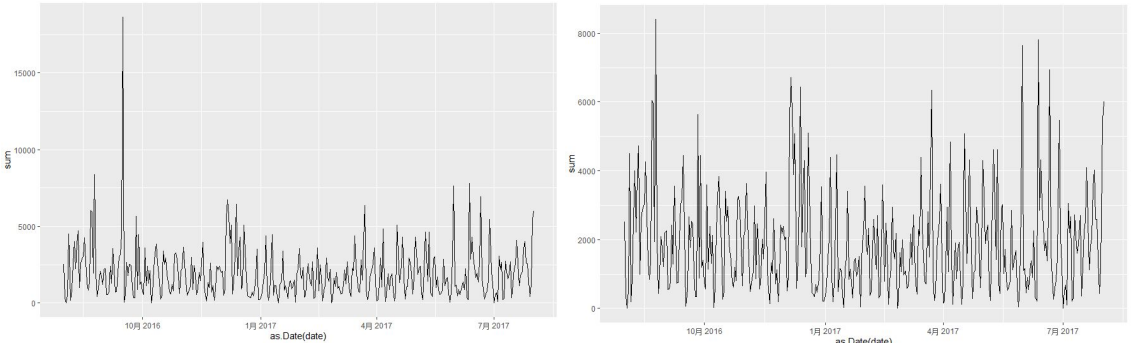
ISE 5103 Intelligent Data Analytics
Homework #4-5 Group 7

Problem (a)

1. Check total revenue everyday

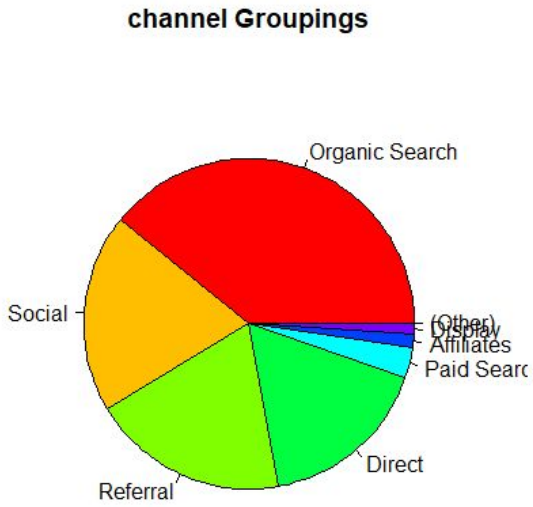
```
revenueEveryDay <- data %>% group_by(date) %>% summarise(sum =
sum(revenue, na.rm = T)) %>% arrange(date)
ggplot(revenueEveryDay, aes(as.Date(date), sum)) +
  geom_line()
revenueEveryDayWithoutTheOutlier <- revenueEveryDay[revenueEveryDay$sum <
15000, ]
ggplot(revenueEveryDayWithoutTheOutlier, aes(as.Date(date), sum)) +
  geom_line()
```

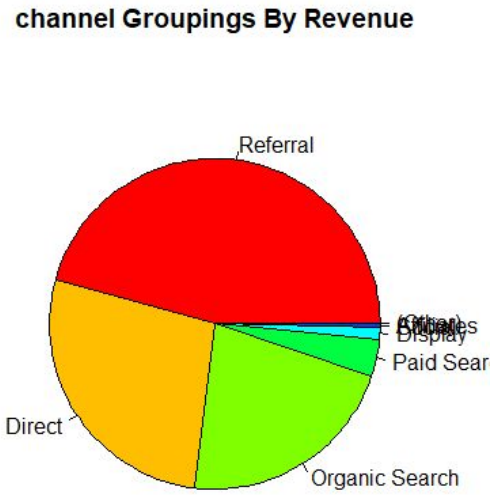


Description: Revenues change every day but we can see there are some special days that the revenue is higher than others. Maybe they did some sales promotion or something.

2. channelGroup by search vs by revenue

```
channelGroupingsByRevenue <- data %>% group_by(channelGrouping) %>%
summarise(sum = sum(revenue, na.rm = T)) %>% arrange(desc(sum))
par(mfrow = c(1, 2))
pie(channelGroupings, main="channel Groupings", col =
rainbow(length(channelGroupings)))
pie(channelGroupingsByRevenue$sum, labels =
channelGroupingsByRevenue$channelGrouping, main="channel Groupings By
Revenue", col = rainbow(nrow(channelGroupingsByRevenue)))
```

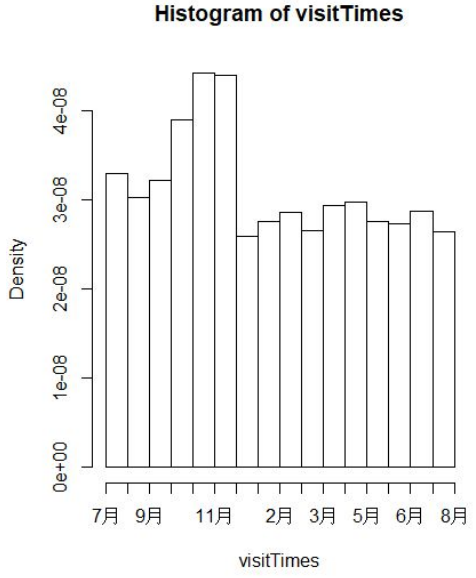


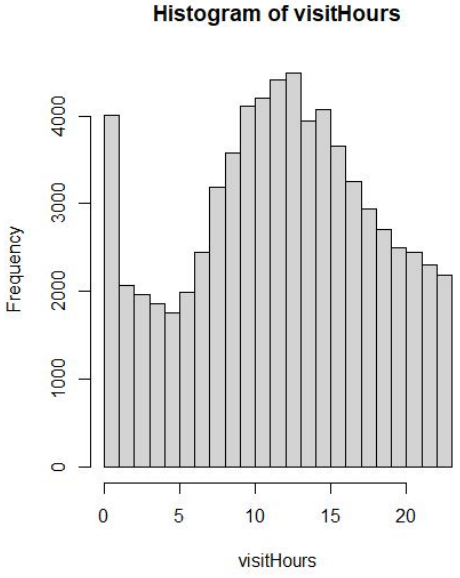


Description: Even though there are some categories which are more frequently been searched, they don't yield more revenue. For example the Social group, maybe customers just come and see what it is, while Referral can be a better factor to cause customers to buy something.

3. Do they usually visit in a specific time or date?

```
visitTimes <- as.POSIXct(data$visitStartTime, origin = "1970-01-01")
hist(visitTimes, breaks = 12)
visitHours <- as.numeric(visitTimes %>% substr(12, 13))
hist(visitHours)
```

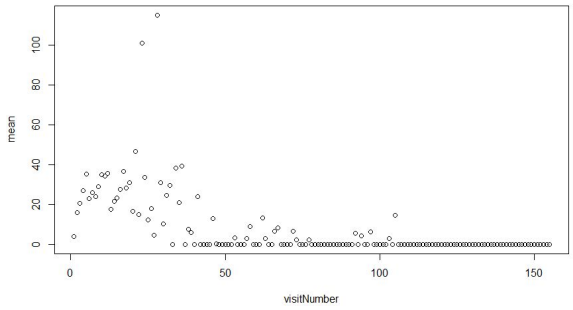


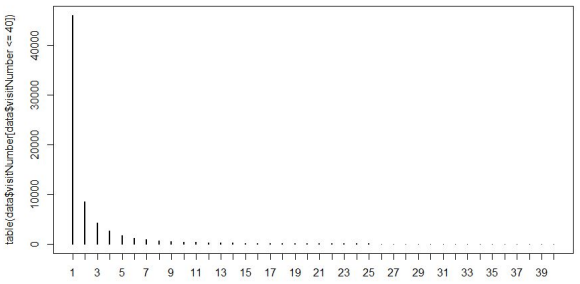


Description: Customers visited more frequently in the winter season of 2016 than others. I guess they must be busy buying something for Thanksgiving and Christmas. But for the visiting hour during one day, I am a little bit confused why they like to visit during work time. Well there is a huge error I might make is that the timestamp recorded is the system time instead of the local time of the customer's country.

4. The relationship between visitNumber and revenue

```
visitNumbers <- data %>% group_by(visitNumber) %>% summarise(mean =
mean(revenue, na.rm = T)) %>% arrange(visitNumber)
plot(visitNumbers)
plot(table(data$visitNumber[data$visitNumber <= 40]))
```





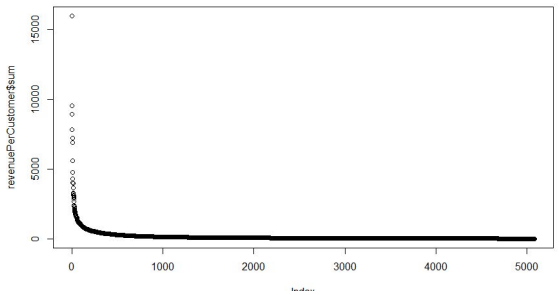
Description: As visit number increases, the mean of the revenue per visit number kinda increases until the visit number reaches around 40. There are some outliers that I have not handled yet. But we can see that most of the customers only visited once.

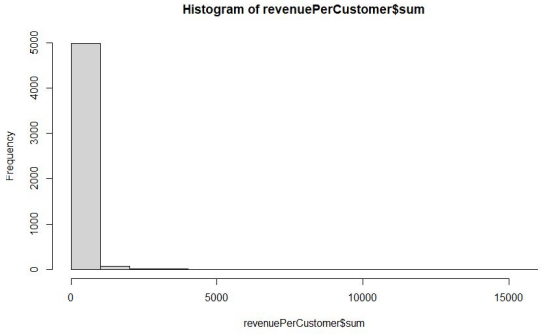
5. take a glimpse of revenue

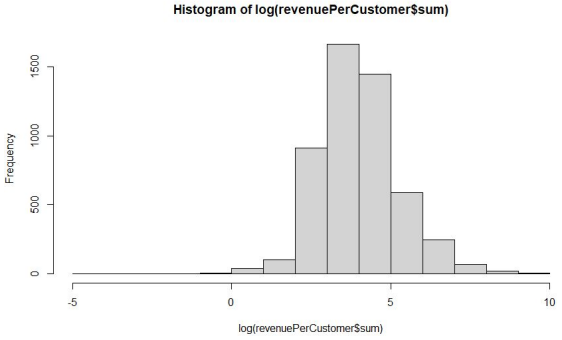
```
sum(data$revenue, na.rm = T)
mean(data$revenue, na.rm = T)
mean(data$revenue[data$revenue > 0], na.rm = T)
revenuePerCustomer <- data[data$revenue > 0, ] %>% group_by(custId) %>%
summarise(sum = sum(revenue, na.rm = T)) %>% arrange(desc(sum))
plot(revenuePerCustomer$sum)
hist(revenuePerCustomer$sum)
hist(log(revenuePerCustomer$sum))
```

```
> sum(data$revenue, na.rm = T)
[1] 712279.2
> mean(data$revenue, na.rm = T)
[1] 10.16583
> mean(data$revenue[data$revenue>0], na.rm = T)
[1] 121.7779
```

```
> mean(revenuePerCustomer$sum)
[1] 140.0746
```







Description: The mean revenue per record is 10.17. The mean revenue per record that has a positive revenue is 121.78. The mean revenue per customer that has a positive revenue is 140.07. These are very important numbers for us to build an intuition.

problem (b)

1. Missing value imputation/transformations: For some variables, if the values are missing or the categories contribute little to modeling, I just fill in the NAs and group them together.

```
for (i in 1 : nrow(originData))
   if (is.na(dataTemp$medium[i])) {
```

```
    dataTemp$medium[i] <- 'medium_Others'
  } else if (dataTemp$medium[i] != 'affiliate' &
             dataTemp$medium[i] != 'cpc' &
             dataTemp$medium[i] != 'cpm' &
             dataTemp$medium[i] != 'organic' &
             dataTemp$medium[i] != 'referral')
  {
    dataTemp$medium[i] <- 'medium_Others'
  }
```

2. Creating new categories/feature extraction: By creating the two new categories: visitMonth and visitHour, we can check if there is a season in a year or some period of time that is the best result

```
visitTime <- dataTemp$visitStartTime %>% as.POSIXct(origin = "1970-01-01")
 dataTemp$visitMonth <- as.numeric(visitTime %>% substr(6, 7))
 dataTemp$visitHour <- as.numeric(visitTime %>% substr(12, 13))
```

3. Aggregations: Summing up of the revenue grouped by customers helps build a better understanding of the data.

```
revenuePerCustomer <- data[data$revenue > 0, ] %>%
 group_by(custId) %>%
 dplyr::summarise(sum = sum(revenue, na.rm = T)) %>%
 arrange(desc(sum))
```

4. Transformation: Create a new category and transform the visit number to its natural log.

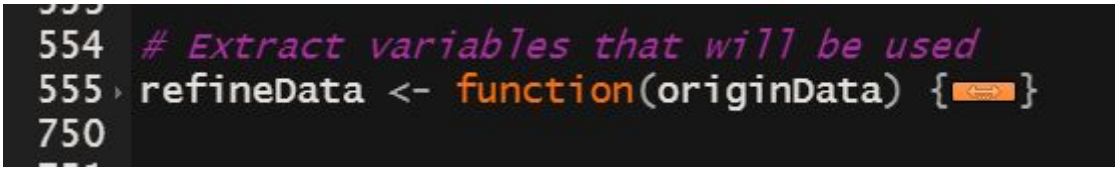
```
dataTemp$logVisitNumber <- log(dataTemp$visitNumber)
```

5. Feature engineering or feature extraction: Based on the timestamp given, we can calculate hours since last visit.

```
dataTemp$hoursSinceLastVisit <- dataTemp$timeSinceLastVisit / 1000 / 60 / 60
```

Problem (c)

1. We created a function to refine the data so each time we can use it by calling the function, and from the refined data, we need to make some decision which categories are needed. We chose all the categories that we think will be the best, cleansed, transformed or created, to help build our model.




```
# refine the data
dataRefined = refineData(data)
dataReborn <- dataRefined[, c('custId', 'month', 'visitMonth', 'visitHour', 'channelGroups',
'logVisitNumber', 'hoursSinceLastVisit', 'browserCategories', 'operatingSystemCategories',
'isMobile', 'deviceCategory', 'continent', 'subContinent', 'country', 'source', 'medium',
'isTrueDirect', 'logPageviews', 'bounces', 'newVisit', 'revenue')]
```

2. Then we transform all variables into numeric to take a look at the relation between each other.

```
# transform into numeric variables
dataNumeric <- dataReborn
dataNumeric$channelGroups <- as.numeric(as.factor(dataNumeric$channelGroups))
dataNumeric$browserCategories <- as.numeric(as.factor(dataNumeric$browserCategories))
dataNumeric$operatingSystemCategories <- as.numeric(as.factor
(dataNumeric$operatingSystemCategories))
dataNumeric$deviceCategory <- as.numeric(as.factor(dataNumeric$deviceCategory))
dataNumeric$continent <- as.numeric(as.factor(dataNumeric$continent))
dataNumeric$subContinent <- as.numeric(as.factor(dataNumeric$subContinent))
dataNumeric$country <- as.numeric(as.factor(dataNumeric$country))
dataNumeric$source <- as.numeric(as.factor(dataNumeric$source))
dataNumeric$medium <- as.numeric(as.factor(dataNumeric$medium))
dataNumeric$isTrueDirect <- as.numeric(as.factor(dataNumeric$isTrueDirect))
```

3. After that we took a look at the correlations of the categories between each other. And removes some that are almost colinear.

```
# check the corelation map
df_cor = cor(dataNumeric)
df_cor

# find and remove other column that absolute correlation value is larger than 0.9
highCor <- findCorrelation(cor(dataNumeric), 0.9)
highCor
highCorNames <- names(dataNumeric)[highCor]
# dataNumeric <- dataNumeric[, -c(5, 10, 2)]
```

4. Finally, we removed some outliers in the revenue to help build a better model.

```
# remove the outlier rows
plot(dataNumeric$revenue)
dataNumericNoOutlier <- dataNumeric[dataNumeric$revenue < 2500, ]
```

Problem (d)

(i)

1. Firstly, We made the partition.

```
set.seed(123)
trainingSize = round(nrow(dataNumericNoOutlier) * 0.7)
train_ind <- sample(seq_len(nrow(dataNumericNoOutlier)), size = trainingSize)
trainingData <- dataNumericNoOutlier[train_ind, ]
testingData <- dataNumericNoOutlier[-train_ind, ]
```

2. Then we wrapped the the prediction part into a function

```
prediction <- function(fit, data, isPredicting) {
  pred <- predict(fit, data)
  pred[pred < 0] <- 0
  if (!isPredicting) {
    result <- data.frame(obs = log(data$revenue + 1), pred = pred)
    defaultSummary(result)
  } else {
    return(pred)
  }
}
```

3. We then created some models to test.

```
############## OLS version ##############

fit <- lm(log(revenue + 1) ~ ., data = dataNumericNoOutlier)
summary(fit)

RSS_fit <- c(crossprod(fit$residuals))
MSE_fit <- RSS_fit / length(fit$residuals)
```

```
RMSE_fit <- sqrt(MSE_fit)
RMSE_fit
```

- output:
  a. model1:

```
> prediction(fit1, testingData, FALSE)
     RMSE  Rsquared       MAE
0.8881779 0.4162631 0.4382082
```

  b. model2:

```
> prediction(fit2, testingData, FALSE)
     RMSE  Rsquared       MAE
0.8930396 0.4078756 0.4341164
```

  c. model3:

```
> prediction(fit3, testingData, FALSE)
     RMSE  Rsquared       MAE
1.1987581 0.4002390 0.3317068
```

  d. model4:

```
> prediction(fit4, testingData, FALSE)
     RMSE  Rsquared       MAE
1.1987376 0.3970536 0.3317101
```

4. Testing the RMSE of the best OLS model

```
fit <- lm(log(revenue + 1) ~ ., data = dataNumericNoOutlier)
summary(fit)

RSS_fit <- c(crossprod(fit$residuals))
MSE_fit <- RSS_fit / length(fit$residuals)
RMSE_fit <- sqrt(MSE_fit)
RMSE_fit
```

```
> RMSE_fit
[1] 0.9103719
```

5. Test with Robust regression

```
huberFit <- rlm(log(revenue + 1) ~ ., data = dataNumericNoOutlier)
summary(huberFit)
```

```
RSS_huberfit <- c(crossprod(huberFit$residuals))
MSE_huberfit <- RSS_huberfit / length(huberFit$residuals)
RMSE_huberfit <- sqrt(MSE_huberfit)
RMSE_huberfit
```

```
> RMSE_huberfit
[1] 1.204183
```

6. Test with MARS

```
marsFit <- earth(log(revenue + 1) ~ ., data = dataNumericNoOutlier)

marsFit$residuals

RSS_mars <- c(crossprod(marsFit$residuals))
MSE_mars <- RSS_mars / length(marsFit$residuals)
RMSE_mars <- sqrt(MSE_mars)
RMSE_mars
```

```
> RMSE_mars
[1] 0.8549861
```

7. Test with PARTIAL LEAST SQUARES

```
plsFit <- plsr(log(revenue + 1) ~ .,
               data = dataNumericNoOutlier,
               10,
               method = "oscorespls")

summary(plsFit)
plsFit$residuals

RSS_pls <- c(crossprod(plsFit$residuals))
MSE_pls <- RSS_pls / length(plsFit$residuals)
RMSE_pls <- sqrt(MSE_pls)
RMSE_pls
```

```
> RMSE_pls
[1] 1.013732
```

The code for creating the output table:

```
Model <- c('OSL', 'Huber loss', 'MARS', 'PLS')
Method <- c('lm', 'rlm', 'earth', 'plsr')
Package <- c('stats', 'MASS', 'earth', 'pls')
Hyperparamter <- c(NA, NA, 'degree', NA)
Selection <- c(NA, NA, NA, NA)
Rsquare <- c(0.3807, 0.4173, 0.4537384, 1.027621)
RMSE <- c(RMSE_fit, RMSE_huberfit, RMSE_mars, RMSE_pls)
df <- data.frame(Model, Method, Package, Hyperparamter, Selection, Rsquare,
RMSE)
df
```

The Table:

	Model	Method	Package	Hyperparamter	Selection	Rsquare	RMSE
1	OSL	lm	stats	<NA>	NA	0.3807000	0.9103719
2	Huber loss	rlm	MASS	<NA>	NA	0.4173000	1.2041834
3	MARS	earth	earth	degree	NA	0.4537384	0.8549861
4	PLS	plsr	pls	<NA>	NA	1.0276210	1.0137325

(ii) At first, we were trying to figure out if we can treat each record as a customer. Then we started to build a model based on that. We didn't sum up each customer's total revenue until the last part. But it turns out we didn't make a high score model.

As to our best model, we compared a bunch of models using different ways to test but the best one is the MARS one which can achieve 0.85 at its best. We used 7-3(train-validation) partitions to do our testing, modified the dummy variables back and forth and tested different variable combinations and found the best one.

We encountered some problems when we were trying to extract some useful information from the given variables. There are so many combinations we can try and there are so many ways we can think about that. But finally we got some ideas about that through discussions.

Secret sauce? Maybe that was a bad decision. We found that when bounces value is 1, revenue is 0. So after we filled in the predicted data, we then filled all revenue values with 0 if the corresponding bounces is 1.

(iii)

We dealt with the test data in the same way we handle the train data.

```
# deal with test data
dataTest <- read.csv('Test.csv', na.strings = c("", "NA"))

dataTest <- dataTest %>% arrange(dataTest$custId)

summary(dataTest)
testDataRefined <- refineData(dataTest)

testDataReborn <- testDataRefined[, c('custId', 'month', 'visitMonth', 'visitHour',
'channelGroups', 'logVisitNumber', 'hoursSinceLastVisit', 'browserCategories',
'operatingSystemCategories', 'isMobile', 'deviceCategory', 'continent', 'subContinent', 'country',
'source', 'medium', 'isTrueDirect', 'logPageviews', 'bounces', 'newVisit')]

# transform into numeric variables
testDataNumeric <- testDataReborn
testDataNumeric$channelGroups <- as.numeric(as.factor(testDataNumeric$channelGroups))
testDataNumeric$browserCategories <- as.numeric(as.factor(testDataNumeric$browserCategories))
testDataNumeric$operatingSystemCategories <- as.numeric(as.factor
(testDataNumeric$operatingSystemCategories))
testDataNumeric$deviceCategory <- as.numeric(as.factor(testDataNumeric$deviceCategory))
testDataNumeric$continent <- as.numeric(as.factor(testDataNumeric$continent))
testDataNumeric$subContinent <- as.numeric(as.factor(testDataNumeric$subContinent))
testDataNumeric$country <- as.numeric(as.factor(testDataNumeric$country))
testDataNumeric$source <- as.numeric(as.factor(testDataNumeric$source))
testDataNumeric$medium <- as.numeric(as.factor(testDataNumeric$medium))
testDataNumeric$isTrueDirect <- as.numeric(as.factor(testDataNumeric$isTrueDirect))
```

And after that we did the prediction part.

```
# Predict it!
pred <- prediction(marsFit, testDataNumeric, TRUE)
pred
head(testDataNumeric)

# save pred
n <- nrow(testDataNumeric)
dataOutput <- data.frame(custId = testDataNumeric$custId, predRevenue = -1)
head(dataOutput)
for (i in 1 : n)
  dataOutput$predRevenue[i] <- pred[i]
```

We saved the data in an output file and submitted several times.

```
# Fill in 0 if bounces in the origin data is equal to 1
for(i in 1 : n)
  if(!is.na(dataTest$bounces[i])) {
    dataOutput$predRevenue[i] <- 0
  }

# Sum up the revenue grouped by custId
dataFinalized <- dataOutput %>% group_by(custId) %>% dplyr::summarise(predRevenue = sum
(predRevenue))
str(dataFinalized)

# Output file
write.csv(dataFinalized, "output.csv", row.names = FALSE)
```