1. *(25 points)* Conduct exploratory data analysis (EDA)

Let’s see the structure/status of the imported Train data

```
> funModeling::df_status(TRAIN_data)
```

```
> funModeling::df_status(TRAIN_data)
                         variable q_zeros p_zeros  q_na  p_na q_inf p_inf    type unique
1                       sessionId       0    0.00     0  0.00     0     0 numeric  70071
2                          custId       0    0.00     0  0.00     0     0 integer  47249
3                            date       0    0.00     0  0.00     0     0  factor    366
4                 channelGrouping       0    0.00     0  0.00     0     0  factor      8
5                  visitStartTime       0    0.00     0  0.00     0     0 integer  69951
6                     visitNumber       0    0.00     0  0.00     0     0 integer    155
7              timeSinceLastVisit   47249   67.43     0  0.00     0     0 integer  20970
8                         browser       0    0.00     0  0.00     0     0  factor     28
9                 operatingSystem       0    0.00     0  0.00     0     0  factor     16
10                       isMobile   53993   77.05     0  0.00     0     0 integer      2
11                 deviceCategory       0    0.00     0  0.00     0     0  factor      3
12                      continent       0    0.00     0  0.00     0     0  factor      6
13                   subContinent       0    0.00     0  0.00     0     0  factor     23
14                        country       0    0.00     0  0.00     0     0  factor    177
15                         region       0    0.00     0  0.00     0     0  factor    310
16                          metro       0    0.00     0  0.00     0     0  factor     73
17                           city       0    0.00     0  0.00     0     0  factor    478
18                  networkDomain       0    0.00     0  0.00     0     0  factor   5015
19                 topLevelDomain       0    0.00     0  0.00     0     0  factor    184
20                       campaign       0    0.00     0  0.00     0     0  factor      7
21                         source       0    0.00     0  0.00     0     0  factor    132
22                         medium       0    0.00     0  0.00     0     0  factor      6
23                        keyword       0    0.00     0  0.00     0     0  factor    416
24                   isTrueDirect   42026   59.98     0  0.00     0     0 integer      2
25                   referralPath       0    0.00     0  0.00     0     0  factor    384
26                      adContent       0    0.00     0  0.00     0     0  factor     28
27           adwordsClickInfo.page       0    0.00 68260 97.42     0     0 integer      5
28           adwordsClickInfo.slot       0    0.00     0  0.00     0     0  factor      3
29          adwordsClickInfo.gclId       0    0.00     0  0.00     0     0  factor   1406
30  adwordsClickInfo.adNetworkType       0    0.00     0  0.00     0     0  factor      2
31      adwordsClickInfo.isVideoAd    1811    2.58 68260 97.42     0     0 logical      1
32                      pageviews       0    0.00     8  0.01     0     0 integer    154
33                        bounces       0    0.00 40719 58.11     0     0 integer      1
34                      newVisits       0    0.00 23944 34.17     0     0 integer      1
35                        revenue   64222   91.65     0  0.00     0     0 numeric   5850
```

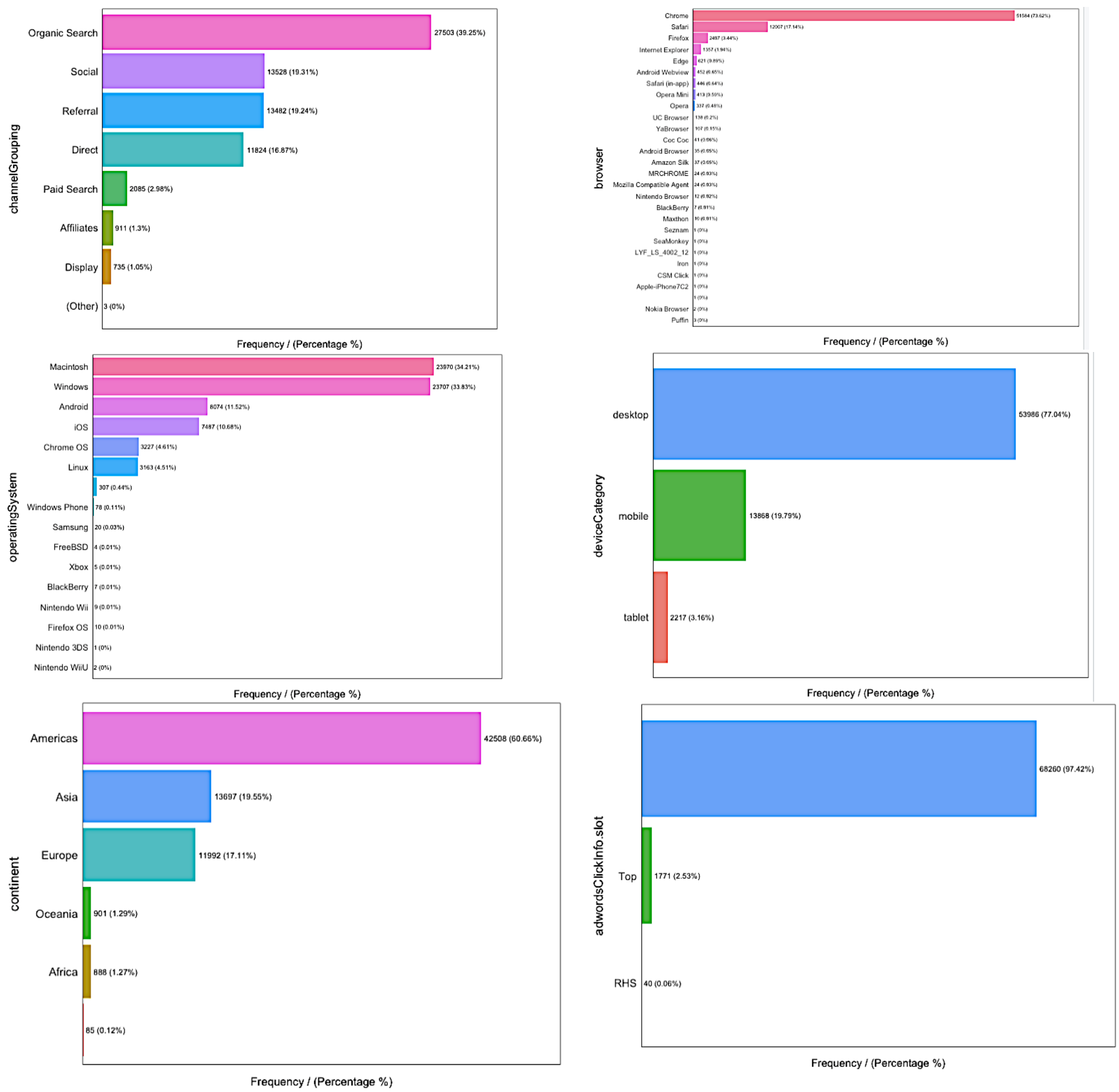
Here we can see all variables, i.e. factors, integers, numeric etc. If we think some variables are in different data type than they need to be transformed accordingly. For example, Date which is a factor variable and we can convert that into any other desired type.

I’m using package called ***funmodeling*** to analyze train data. ***df_status*** in getting the metrics about data types, zeros, infinite numbers, and missing values. It returns a table, so it is easy to keep with variables that match certain conditions like I decided not to consider the variables which has more than 90% of missing variables. Like ***adwordsClickInfo.page*** which has 97% missingness. ***adwordsClickInfo.isVideoAd*** also has 97% missingness. However, I decided to consider this because it is logical type (TRUE/FALSE) where it also showing it has only one unique type (All FLASE). Depending on context information I can decide in future whether to impute it with True or not to consider this variable.

Here missing means it will consider only NA values but in some cases (especially Categorical) the missingness will be represented in different data formats. This can only be known by checking into each variable.

I’m using ***freq*** function for Analyzing categorical variables. Which runs for all factor or character variables automatically.

```
funModeling::freq((TRAIN_data))
```

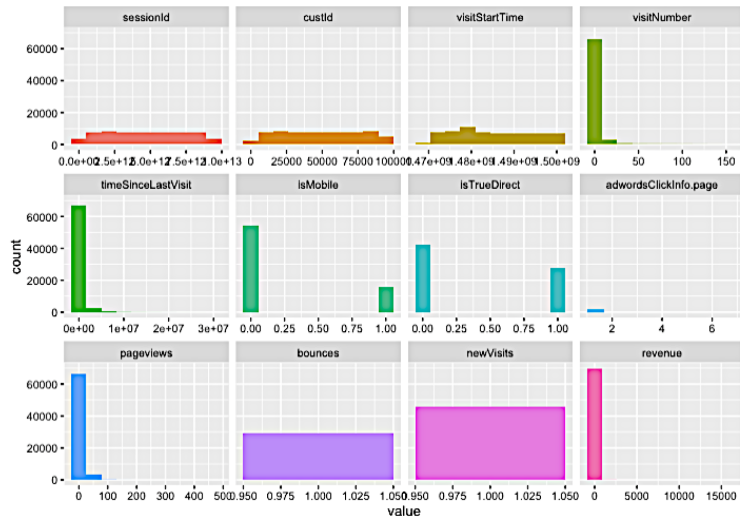


These freq plots helps to understand the data more. Here I have only 6 example plots we can analyze each category to see draw some meaningful insights. Up until now we are calculating the missing values based on NA. Here If we see there are some blank data (‘’) example *browser*, *adwordsClickInfo.slot* and *continent*. These can be replaced with N/A and handle during our cleaning process.

From these plot’s we can also see many variables which are having more than 100 categories also. We can consider those variables which having many categories with less frequency values as outliers and can handle accordingly.

For analyzing numerical variables using ***plot_num()*** function from same package ***funModeling***

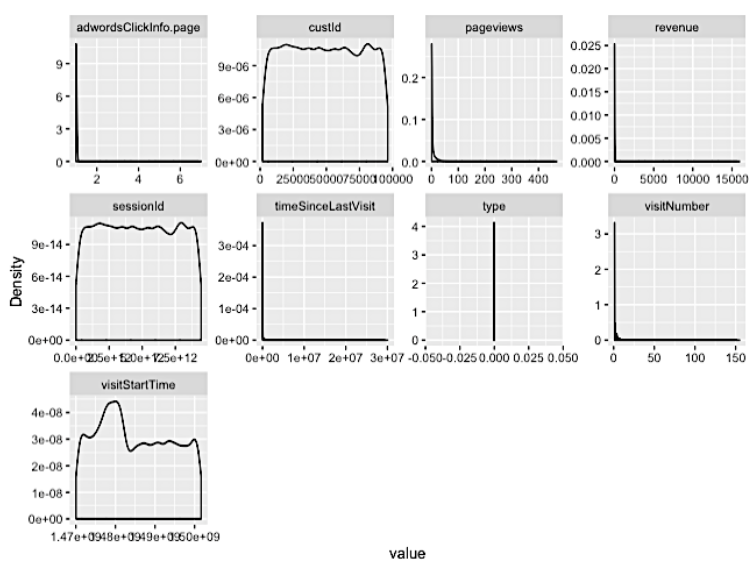
```
funModeling::plot_num(TRAIN_data)
```



The above plot is the histogram's for numeric plot which indicating the number of data points that lie within a range of values nothing but distribution. We can think about applying skew transformations for VistNumber, LastVisit, PageViews which has more skewness. And few variables like bounces, isMobile, isTrueDirect can be treated as ***logical*** variables.

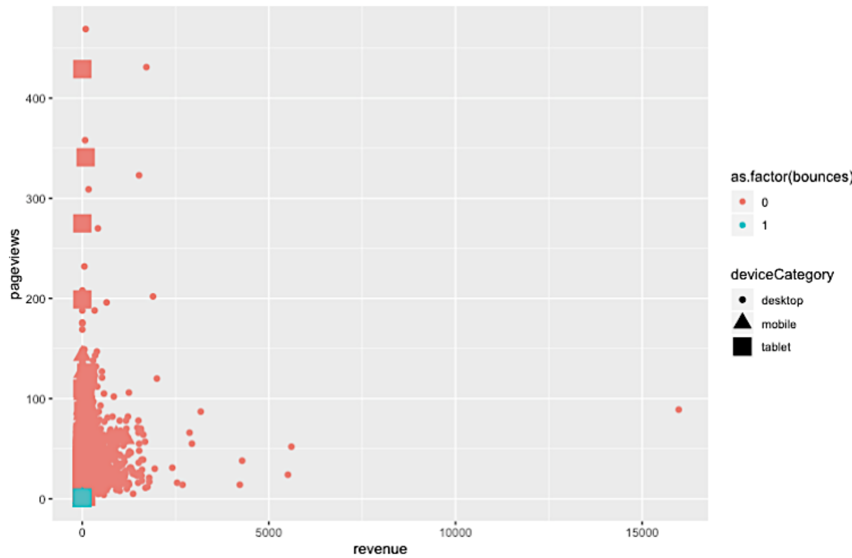
We can also see direct density plots using plot_density() from DataExplorer package. Which excludes the binary/logical type of data and take only the continuous variables to show density plots. This is cool package for exploring data.

```
DataExplorer::plot_density(TRAIN_data)
```



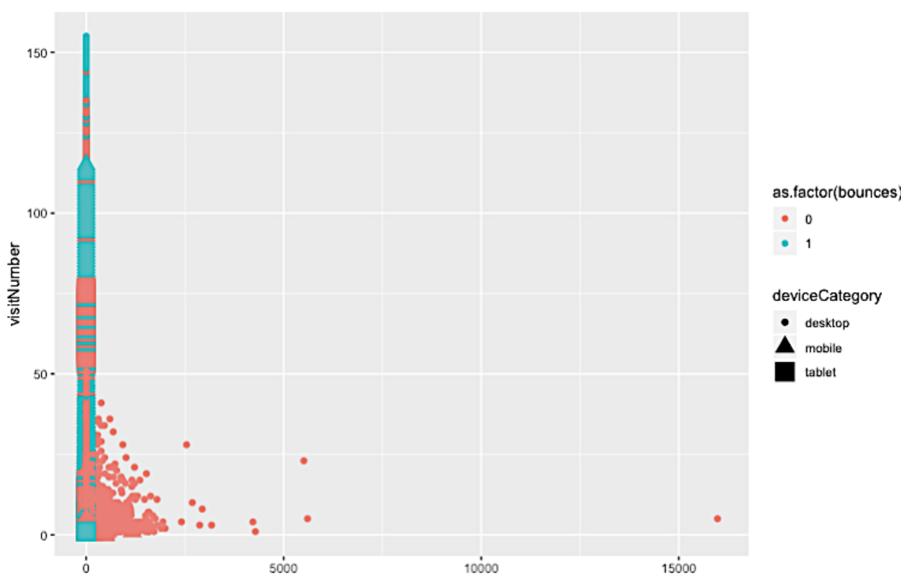
For better understanding the relation b/w the data variables let's ggplots.

```
ggplot(data = TRAIN_data) + geom_point(mapping = aes(x = revenue, y =
pageviews , colour = as.factor(bounces), shape = deviceCategory, size
= deviceCategory)) # cato variable
```



The above plot helps to understand the relation between the pageviews and revenue with respective with bounces and deviceCatogory. We can see where ever it bounces the revenue is almost equal to zero, and that it there are more chances of bouncing from tablet and mobile. It also doesn't matter if the page views are more there are many cases where the revenue is less if that session is from mobile/tablet.

```
ggplot(data = TRAIN_data) +geom_point(mapping = aes(x = revenue, y =
visitNumber , colour = as.factor(bounces), shape = deviceCategory,
size = deviceCategory)) # cato variable
```



From the above plot we can see the relation between visitnumber and revenue with respective to bonces and deviceCategoty. Though visitNumber is more for revenue is less. And browsing is from tablet/mobile again where it bounces more it leads to less revenue.

This way we can see what factors effect and leads to more revenue.

2. *(30 points)* **Prepare the data for modeling. This requires you to consider missing values, outliers, transformations, aggregations, and/or any other data preparation technique you find useful.**

At first combining both train and test data so that I can make preparation for both at the same level.

```
-  TRAIN_data$type<- 0
-  TEST_data$type <- 1
-  TEST_data$revenue <- NA
-  fulldata <- rbind(TRAIN_data,TEST_data)
```

Adding new attribute ***type*** to differentiate test and train for splitting it later.

```
-  fulldata[fulldata== ""] <- NA
```

As we learned from EDA that there is ‘’ black data which is also treated as NA. So Now it’s easy to check for missingness of variables.

```
fulldata %>% mutate_all(is.na) %>% summarise_all(mean) # checking
missingness
```

```
> fulldata %>% mutate_all(is.na) %>% summarise_all(mean) # checking missingness
  sessionId custId date channelGrouping visitStartTime visitNumber timeSinceLastVisit      browser operatingSystem isMobile
1         0      0    0               0              0           0                  0 7.155994e-06     0.004093228        0
  deviceCategory   continent subContinent     country    region     metro      city networkDomain topLevelDomain  campaign
1              0 0.001259455  0.001259455 0.001259455 0.5505034 0.7028116 0.5566433     0.4747715      0.4747715 0.9586169
        source    medium   keyword isTrueDirect referralPath adContent adwordsClickInfo.page adwordsClickInfo.slot
1 6.440394e-05 0.1656326 0.9600195            0    0.6142705 0.9868688             0.9726927             0.9726927
  adwordsClickInfo.gclId adwordsClickInfo.adNetworkType adwordsClickInfo.isVideoAd    pageviews   bounces newVisits   revenue
1              0.9725568                      0.9726927                  0.9726927 0.0001359639 0.5801793 0.3394016 0.4985724
  type
1    0
```

Now we can see full missingness from each column. There is a date column which is in a factor format. So converting and considering new factors like year, month and week which many effect revenue

```
-  fulldata$date<- as.Date(fulldata$date, "%Y-%m-%d")
-  fulldata$month<- as.factor(strftime(fulldata$date,"%m"))
-  fulldata$year<- as.factor(strftime(fulldata$date,"%Y"))
-  fulldata$roundweek <- as.factor(weekdays(fulldata$date))
-  summary(fulldata)
-  fulldata <- fulldata[,-c(3)] # removing date
```

Separating numerical and categorical variables and will apply cleaning individually

```
-  full_cat<- fulldata[ ,sapply(fulldata, is.factor)]
-  colnames(full_cat)
-  full_num <- fulldata[ ,!sapply(fulldata, is.factor)]
-  colnames(full_num)
```

Removing columns which is having more than 80% of missingness. We can see more data missing from two columns. We are not considering them.

```
-   full_num<-full_num[,colMeans(is.na(full_num)) <= .80]
-   full_cat<-full_cat[,colMeans(is.na(full_cat)) <= .80]
```

I'm also not considering 'region','metro','city','networkDomain','topLevelDomain' variables which has too many categories types. And also which we feel are not necessary.

```
-   full_cat <- full_cat[ , !(names(full_cat) %in%
    c('region','metro','city','networkDomain','topLevelDomain'))]
```

Adding more frequent occurred values for categorical variables.

```
-   for(i in colnames(full_cat))
-   {
-   full_cat[,i]<- fct_explicit_na(full_cat[,i], na_level =
    names(sort(summary(full_cat[,i]), decreasing=T))[1])
-   }
```

For Numerical variables as explained in data information bounces given null if not. For remaining variables also considering 0 for imputing the missing data

```
-   full_num$bounces[is.na(full_num$bounces)] <- 0
-   full_num$newVisits[is.na(full_num$newVisits)] <- 0
-   full_num$pageviews[is.na(full_num$pageviews)] <- 0
```

Applying one hot encoding for categorical variables to convert it into numeric format.

```
-   catcols <- colnames(full_cat)
-   for(i in catcols)
-   {
-   print(i)
-   full_cat[i] <- str_replace_all(full_cat[,i], "[^[:alnum:]]", " ") #
    to remove special char
-   d <- dummyVars(" ~ .", data = full_cat[i])
-   onehot <- data.frame(predict(d, newdata = full_cat[i]))
-   full_cat <- cbind(full_cat,onehot)
-   }
```

Applying log transformation as discussed in EDA for few variables which can benefit from skew transformation

```
-   full_num$visitNumber <- log(full_num$visitNumber+1)
-   full_num$timeSinceLastVisit <- log(full_num$timeSinceLastVisit+1)
-   full_num$pageviews <- log(full_num$pageviews+1)
```

After combining numerical and categorical data, splitting rows back to get train and test prepared data. Now the data is ready for both building model using train data and predicting revenue for test data.

3. ***(45 points)*** **Build the best possible linear model using lm to predict the target value.**

3.1 ***(15 points)*** **For your best model, report the variables, coefficient estimates, and p-values. Additionally, report the re-sampled RMSE and R^2 values. What are your thoughts on the quality of the model? Did you have any problems during the modeling process? If so, how did you overcome those?**

I'm considering the variables based on correlation between revenue and rest. Considering both positive and negative correlated variables. i.e > +0.1 and < -0.1

```
- corv <- cor((outtrain), log(outtrain$revenue+1))
```

'revenue', 'custId', 'pageviews', 'sourcemall.googleplex.com', 'countryUnited.States', 'subContinentNorthern.America', 'referralPath.', 'continentAmericas', 'channelGroupingReferral', 'timeSinceLastVisit', 'operatingSystemMacintosh', 'isTrueDirect', 'browserChrome', 'visitNumber', 'deviceCategorydesktop', 'bounces', 'newVisits', 'channelGroupingSocial', 'continentAsia', 'sourceyoutube.com', 'continentEurope', 'isMobile'.

Aggregating revenue by custId and removing custId before building a model.

```
- souttrain <- data.frame(aggregate(souttrain,
  by=list(souttrain$custId),FUN=mean))
- souttrain <- souttrain[ , !(names(souttrain) %in% c('custId','
  Group.1'))]
- fit <- lm(log(revenue+1) ~ . , data = souttrain)
- summary(fit)
```

```
> summary(fit)

Call:
lm(formula = log(revenue + 1) ~ ., data = souttrain)

Residuals:
    Min      1Q  Median      3Q     Max 
-3.4270 -0.2010  0.0102  0.2076  6.3903 

Coefficients:
                             Estimate Std. Error t value Pr(>|t|)    
(Intercept)                 -2.914308   0.443099  -6.577 4.85e-11 ***
visitNumber                  0.246835   0.028525   8.653  < 2e-16 ***
timeSinceLastVisit           0.064092   0.002654  24.147  < 2e-16 ***
isMobile                     1.058097   0.440048   2.405 0.016198 *  
isTrueDirect                 0.090226   0.010691   8.439  < 2e-16 ***
pageviews                    0.981357   0.007123 137.765  < 2e-16 ***
bounces                      0.715805   0.010301  69.491  < 2e-16 ***
newVisits                    0.083135   0.034961   2.378 0.017415 *  
channelGroupingReferral     -0.052841   0.019637  -2.691 0.007128 ** 
channelGroupingSocial        0.060623   0.038416   1.578 0.114559    
browserChrome                0.014533   0.008128   1.788 0.073776 .  
operatingSystemMacintosh     0.093964   0.008651  10.861  < 2e-16 ***
deviceCategorydesktop        1.140487   0.439924   2.592 0.009532 ** 
continentAmericas           -0.025480   0.022860  -1.115 0.265027    
continentAsia               -0.016195   0.019832  -0.817 0.414166    
continentEurope             -0.015972   0.020032  -0.797 0.425280    
subContinentNorthern.America -0.092634  0.025247  -3.669 0.000244 ***
countryUnited.States         0.243978   0.021822  11.180  < 2e-16 ***
sourcemall.googleplex.com    0.605765   0.049140  12.327  < 2e-16 ***
sourceyoutube.com            0.068717   0.038830   1.770 0.076785 .  
referralPath.                0.008052   0.045369   0.177 0.859141    
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.7367 on 47228 degrees of freedom
Multiple R-squared:  0.5144,	Adjusted R-squared:  0.5142 
F-statistic:  2502 on 20 and 47228 DF,  p-value: < 2.2e-16
```

```
model.summary = summary(fit)$coefficients
model.summary
```

```
> model.summary
                                Estimate  Std. Error     t value      Pr(>|t|)
(Intercept)                 -2.914308312 0.443098776  -6.5771076  4.846797e-11
visitNumber                  0.246834968 0.028525359   8.6531766  5.163318e-18
timeSinceLastVisit           0.064092489 0.002654227  24.1473265 4.776712e-128
isMobile                     1.058096636 0.440047699   2.4045044  1.619824e-02
isTrueDirect                 0.090225624 0.010691116   8.4393086  3.281446e-17
pageviews                    0.981357302 0.007123440 137.7645149  0.000000e+00
bounces                      0.715804957 0.010300749  69.4905721  0.000000e+00
newVisits                    0.083135473 0.034961479   2.3779164  1.741473e-02
channelGroupingReferral     -0.052840840 0.019636770  -2.6909131  7.128183e-03
channelGroupingSocial        0.060623238 0.038416330   1.5780591  1.145587e-01
browserChrome                0.014533049 0.008127933   1.7880375  7.377636e-02
operatingSystemMacintosh     0.093963521 0.008651231  10.8612889  1.899297e-27
deviceCategorydesktop        1.140486602 0.439923522   2.5924656  9.532006e-03
continentAmericas           -0.025480177 0.022860334  -1.1146022  2.650266e-01
continentAsia               -0.016194620 0.019831985  -0.8165910  4.141663e-01
continentEurope             -0.015971735 0.020032242  -0.7973014  4.252800e-01
subContinentNorthern.America -0.092634091 0.025247115  -3.6690961  2.436771e-04
countryUnited.States         0.243978389 0.021822322  11.1802213  5.542181e-29
sourcemall.googleplex.com    0.605764707 0.049139803  12.3273735  7.300202e-35
sourceyoutube.com            0.068716808 0.038829842   1.7696907  7.678514e-02
referralPath.                0.008051504 0.045368696   0.1774683  8.591413e-01
```

The above result shows estimated coefficient values, Std error, and p values.

```
summary(fit)$r.squared # if more it is it will be good
[1] 0.5144175
```

We got r squared value 0.514. Which is not a great result. The more it is the better the model will perform.

```
test <- predict(fit, souttrain)
rmse(log(souttrain$revenue+1),test)
[1] 0.7365773
```

The above rmse score is from the whole train data. Here also the same we got 0.73. The less we get the better the model will perform.

Cross Validation of our model:

We are using caret package to do cross validation

```
-   data_ctrl <- trainControl(method = "cv", number = 5)
-   model_caret <- train(log(revenue+1) ~ .,   # model to fit
                         data = souttrain,
                         trControl = data_ctrl,            # folds
                         method = "lm",       # specifying regression
                         model
                         na.action = na.pass)
-   model_caret
```

```
> model_caret
Linear Regression

47249 samples
   20 predictor

No pre-processing
Resampling: Cross-Validated (5 fold)
Summary of sample sizes: 37799, 37799, 37800, 37799, 37799
Resampling results:

  RMSE   Rsquared  MAE
  0.737  0.514     0.435

Tuning parameter 'intercept' was held constant at a value of TRUE
```

The resampled RMSE and R squared of our model is 0.737 and 0.514 respectively. Lets check for each fold to see if there is any variable.

```
> model_caret$resample
   RMSE Rsquared   MAE Resample
1 0.741    0.503 0.439    Fold1
2 0.740    0.512 0.437    Fold2
3 0.723    0.513 0.428    Fold3
4 0.747    0.523 0.440    Fold4
5 0.733    0.519 0.430    Fold5
```

These is no much difference between the results of 5 different folds. From which we can say there is no over fitting and out model is working same for different sampled data.

Final model coefficient values

```
> model_caret$finalModel

Call:
lm(formula = .outcome ~ ., data = dat)

Coefficients:
                (Intercept)                 visitNumber           timeSinceLastVisit
                   -2.91431                     0.24683                      0.06409
                   isMobile                isTrueDirect                    pageviews
                    1.05810                     0.09023                      0.98136
                    bounces                   newVisits       channelGroupingReferral
                    0.71580                     0.08314                     -0.05284
      channelGroupingSocial               browserChrome      operatingSystemMacintosh
                    0.06062                     0.01453                      0.09396
      deviceCategorydesktop           continentAmericas                 continentAsia
                    1.14049                    -0.02548                     -0.01619
            continentEurope  subContinentNorthern.America           countryUnited.States
                   -0.01597                    -0.09263                      0.24398
   sourcemall.googleplex.com            sourceyoutube.com                 referralPath.
                    0.60576                     0.06872                      0.00805
```

Problems during model processing:

It took a while to understand data. What to consider for the model. Still not sure if we should consider variables based on the correlation with the target variable or with the p value (<0.05). For this model we considered variables which has more correlation with revenue.

Another problem we faced is with categorical variables. Which has more than 150 categories like country. One hot encoding is really big task here. Can't do for all since there is some limit with the columns vector. We considered only which impact more on revenue in the Realtime.

3.2 ***(30 points)*** **Submit your model predictions to the Kaggle.com competition website and outperform your peers in high quality predictions on the test data.**

Submitted test revenue results on to Kaggle. Please find our team name – **(C) AS 07.**