Google Analytics Customer Revenue Prediction

Team Member: Ji Li; Junchi Zhang; Jianwei Wu

Section A: Exploratory data analysis

(a) The goals of the Kaggle challenge:

In this competition, we're supposed to analyze a Google Merchandise Store customer dataset to predict natural log of the sum of all transactions per user. Finally, we need to get the root mean squared error of our prediction, where it is the natural log of the actual summed revenue value plus one.

(b) Basic information about the dataset:

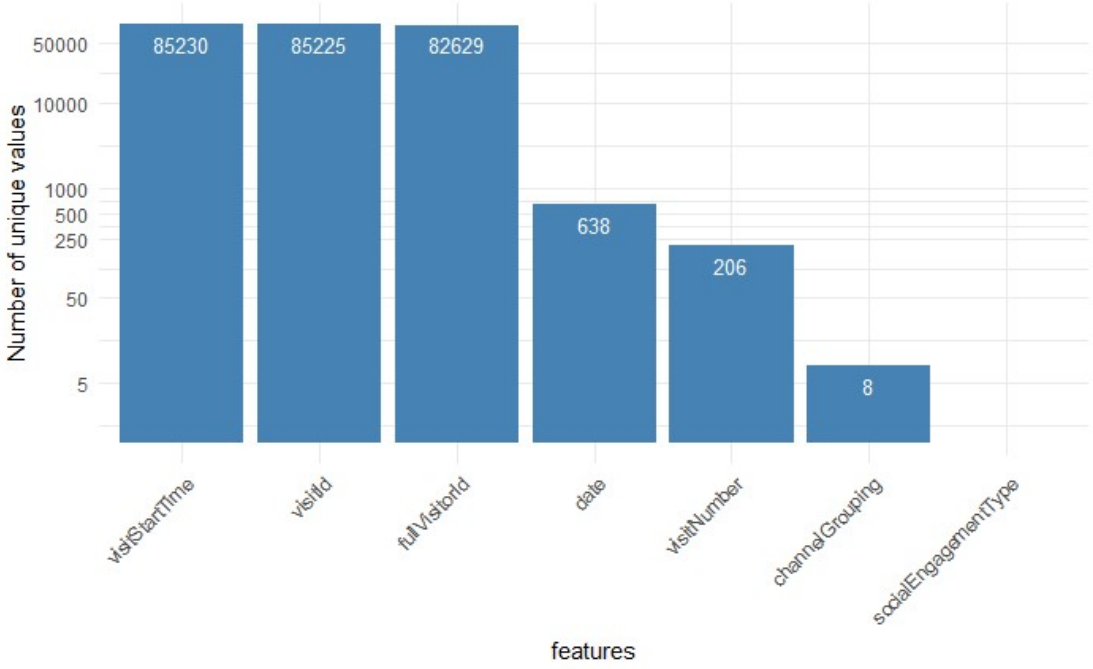
The original training dataset has 903653 rows and 13 columns. Because of the large size of the data, we randomly sampled 10% as our training dataset. Here're introductions of some columns:
date - the date on which the user visited the Store;
channelGrouping - the channel via which the user came to the Store;
device - the specifications for the device used to access the Store;
totals - this section contains aggregate values across the session;
geoNetwork - this section contains information about the geography of the user;
socialEngagementType - engagement type, either "Socially Engaged" or "Not Socially Engaged";
trafficSource - this section contains information about the Traffic Source from which the session originated;

```
Variables: 14
$ X1                 <dbl> 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, ...
$ channelGrouping    <chr> "Organic Search", "Social", "Social", "Direct", "Organic Search", "Organic Search", "Direct", "Social",...
$ customDimensions   <chr> "[{'index': '4', 'value': 'APAC'}]", "[]", "[]", "[{'index': '4', 'value': 'North America'}]", "[{'inde...
$ date               <dbl> 20170408, 20161023, 20161119, 20180104, 20171117, 20170105, 20170213, 20180222, 20170729, 20161022, 201...
$ device             <chr> "{\"browser\": \"Safari\", \"browserVersion\": \"not available in demo dataset\", \"browserSize\": \"no...
$ fullVisitorId      <chr> "2890301279085222383", "5926399220179584913", "1868327608361880110", "3372045887828356170", "2180988947...
$ geoNetwork         <chr> "{\"continent\": \"Oceania\", \"subContinent\": \"Australasia\", \"country\": \"Australia\", \"region\"...
$ hits               <chr> "[{'hitNumber': '1', 'time': '0', 'hour': '21', 'minute': '14', 'isInteraction': True, 'isEntrance': Tr...
$ socialEngagementType <chr> "Not Socially Engaged", "Not Socially Engaged", "Not Socially Engaged", "Not Socially Engaged", "Not So...
$ totals             <chr> "{\"visits\": \"1\", \"hits\": \"8\", \"pageviews\": \"8\", \"timeOnSite\": \"307\", \"newVisits\": \"1...
$ trafficSource      <chr> "{\"campaign\": \"(not set)\", \"source\": \"google\", \"medium\": \"organic\", \"keyword\": \"(not pro...
$ visitId            <dbl> 1491711293, 1477260328, 1479572698, 1515120070, 1510945111, 1483687445, 1487036565, 1519313355, 1501374...
$ visitNumber        <dbl> 1, 1, 1, 1, 1, 1, 3, 1, 1, 1, 1, 1, 3, 1, 1, 1, 1, 1, 1, 1, 10, 1, 1, 2, 1, 1, 3, 1, 1, 1, 1, 1, 22, 1,...
$ visitStartTime     <dbl> 1491711293, 1477260328, 1479572698, 1515120070, 1510945111, 1483687445, 1487036565, 1519313355, 1501374...
```

From the "glimpse" of training, we can see that six of the columns have the format of JSON. Plenty of important information, including our target variable, are in these columns. We should split them and get much more variables.

For a times search on the internet, we finally decide to use the package "jsonlite" to parse the variables.

First, we take a look at the number of values we have in some simple features. We choose fullVisitorId, channelGrouping, date, socialEngagementType, visitId, visitNumber and visitStartTime.
"socialEngagementType" only has one value. So, we should delete it later on.



Then we use the package "jsonlite" to work on the JSON format columns.

```
Variables: 41
$ channelGrouping                <fct> Organic Search, Organic Search, Organic Search, Social, Direct, Organic Search, Direct, Socia...
$ visitNumber                    <dbl> 2, 2, 1, 1, 1, 1, 1, 1, 1, 1, 2, 1, 1, 1, 4, 1, 1, 1, 1, 2, 1, 1, 1, 44, 1, 1, 1, 1, 1, 1, 1,...
$ browser                        <fct> Chrome, Opera Mini, Chrome, Chrome, Opera, Internet Explorer, Chrome, Chrome, Chrome, Interne...
$ operatingSystem                <fct> Android, Samsung, Windows, Windows, Android, Windows, Macintosh, Windows, Windows, Windows, W...
$ isMobile                       <int> 1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 0, 1, 1, 0, ...
$ deviceCategory                 <fct> mobile, mobile, desktop, desktop, mobile, desktop, desktop, desktop, desktop, desktop, deskto...
$ continent                      <fct> Americas, Asia, Americas, Asia, Europe, Europe, Europe, Europe, Europe, Europe, Europe, Ameri...
$ subContinent                   <fct> Northern America, Southern Asia, South America, Southeast Asia, Northern Europe, Northern Eur...
$ country                        <fct> United States, India, Brazil, Thailand, Denmark, United Kingdom, Sweden, Serbia, Portugal, Ge...
$ region                         <fct> California, NA, NA, Bangkok, NA, NA, Stockholm County, NA, NA, NA, NA, NA, New York, NA, Cali...
$ metro                          <fct> San Francisco-Oakland-San Jose CA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, New York NY, N...
$ city                           <fct> San Francisco, NA, NA, Bangkok, NA, NA, Stockholm, NA, NA, NA, NA, NA, New York, NA, Mountain...
$ networkDomain                  <fct> comcast.net, NA, pucrs.br, totbb.net, tdc.net, cdd.ac.uk, NA, NA, vodafone.pt, NA, iol.cz, nb...
$ campaign                       <fct> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, Data Share Promo,...
$ source                         <fct> (direct), google, (direct), youtube.com, (direct), (direct), (direct), youtube.com, google, g...
$ medium                         <fct> NA, organic, NA, referral, NA, NA, NA, referral, organic, organic, NA, referral, organic, org...
$ isTrueDirect                   <int> 1, 1, NA, NA, 1, NA, 1, NA, NA, NA, 1, NA, NA, NA, 1, NA, NA, NA, 1, 1, 1, 1, NA, 1, NA, NA, ...
$ keyword                        <fct> NA, www.google.com pouch, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,...
$ referralPath                   <fct> NA, NA, NA, /yt/about/th/, NA, NA, NA, /yt/about/sr/, NA, NA, NA, /yt/about/policies/, NA, NA...
$ adContent                      <fct> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, N...
$ adwordsClickInfo.page          <fct> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, N...
$ adwordsClickInfo.slot          <fct> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, N...
$ adwordsClickInfo.gclId         <fct> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, N...
$ adwordsClickInfo.adNetworkType <fct> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, N...
$ adwordsClickInfo.isVideoAd     <int> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, N...
$ pageviews                      <int> 2, 2, 8, 3, 1, 1, 3, 5, 1, 1, 5, 2, 1, 3, 1, 12, 1, 2, 26, 3, 1, 1, 2, 1, 4, 1, 1, 1, 2, 1, 1...
$ timeOnSite                     <fct> 247, 16, 183, 121, NA, NA, 248, 438, NA, NA, 70, 78, NA, 77, NA, 213, NA, 62, 436, 3, NA, NA,...
$ sessionQualityDim              <fct> NA, 1, NA, NA, 1, NA, NA, NA, 1, 1, NA, 1, 1, NA, 1, 2, 1, NA, 8, 1, 1, 1, NA, NA, 2, 1, 1, N...
$ newVisits                      <int> NA, NA, 1, 1, 1, 1, 1, 1, 1, 1, NA, 1, 1, 1, NA, 1, 1, 1, 1, NA, 1, 1, 1, NA, 1, 1, 1, 1, 1, ...
$ bounces                        <int> NA, NA, NA, NA, 1, 1, NA, NA, 1, 1, NA, NA, 1, NA, 1, NA, 1, NA, NA, NA, 1, 1, NA, 1, NA, 1, ...
$ transactions                   <fct> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, N...
$ totalTransactionRevenue        <fct> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, N...
$ transactionRevenue             <dbl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, N...
$ year                           <fct> 2016, 2017, 2016, 2016, 2017, 2016, 2017, 2016, 2018, 2017, 2016, 2017, 2018, 2017, 2018, 201...
$ wday                           <fct> 6, 1, 2, 3, 6, 4, 3, 5, 2, 7, 4, 4, 4, 7, 4, 1, 3, 5, 2, 6, 6, 4, 6, 3, 2, 6, 2, 5, 2, 5, 4, ...
$ hour                           <fct> 7, 6, 16, 3, 21, 12, 8, 16, 0, 14, 16, 18, 0, 0, 22, 4, 10, 22, 11, 13, 11, 17, 23, 21, 20, 0...
$ pageviews_mean_vn              <dbl> 4.315583, 4.315583, 3.432894, 3.432894, 3.432894, 3.432894, 3.432894, 3.432894, 3.432894, 3.4...
$ pageviews_mean_country         <dbl> 5.091425, 3.081594, 2.216667, 1.854730, 3.391304, 2.491892, 2.466667, 1.783784, 3.283333, 2.6...
$ pageviews_mean_city            <dbl> 4.747253, 2.000000, 8.000000, 2.032787, 1.000000, 1.000000, 3.153846, 5.000000, 1.000000, 1.0...
$ pageviews_mean_dom             <dbl> 5.359191, 2.000000, 8.000000, 1.602941, 1.090909, 1.000000, 3.000000, 5.000000, 2.416667, 1.0...
$ pageviews_mean_ref             <dbl> 2.000000, 2.000000, 8.000000, 1.439655, 1.000000, 1.000000, 3.000000, 1.800000, 1.000000, 1.0...
```

After extracting the variables in the json format variables, we want to visualize our data to make it more clearly and we can analyze them more convenient.

In next step, we are going to check the content of each variables. We found several columns that have only one value so we removed them because they're useless to predict our target.

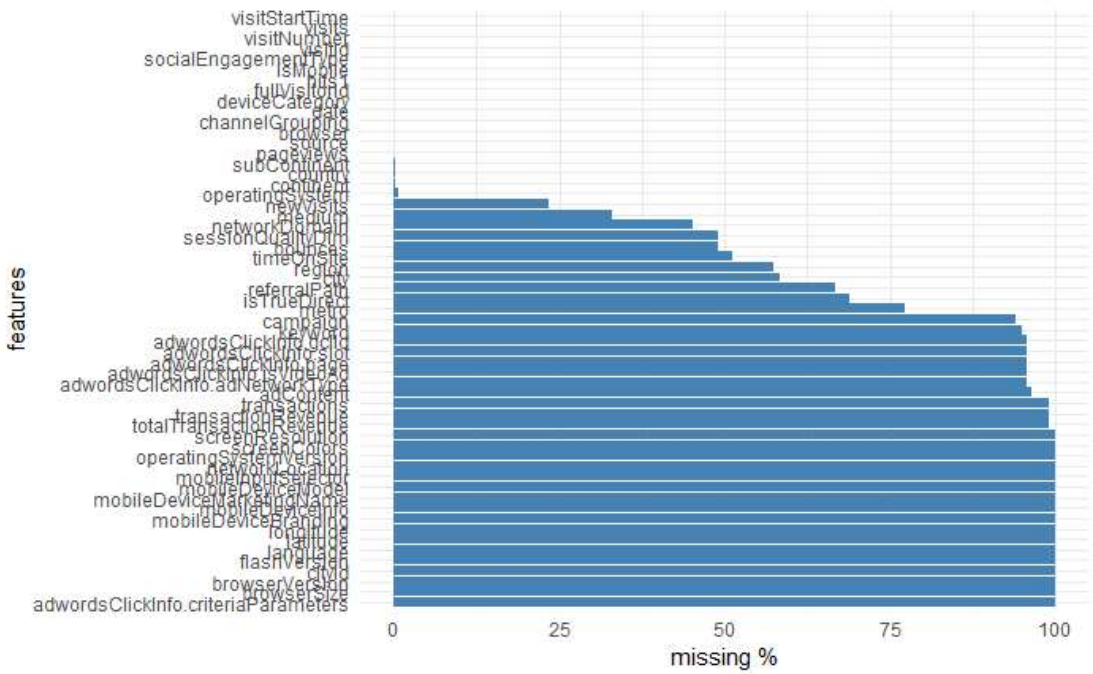
By going through the whole data set, we found there're a lot of different types of unavailable numbers. So, we mutate all of them as NA to make it easier to use them next.

Of course, the formats are also very important parts in the data analysis. But we found that most of variables formats are variable, so we use “ymd”, “as.integer” etc. functions to convert the variables to their own formats.

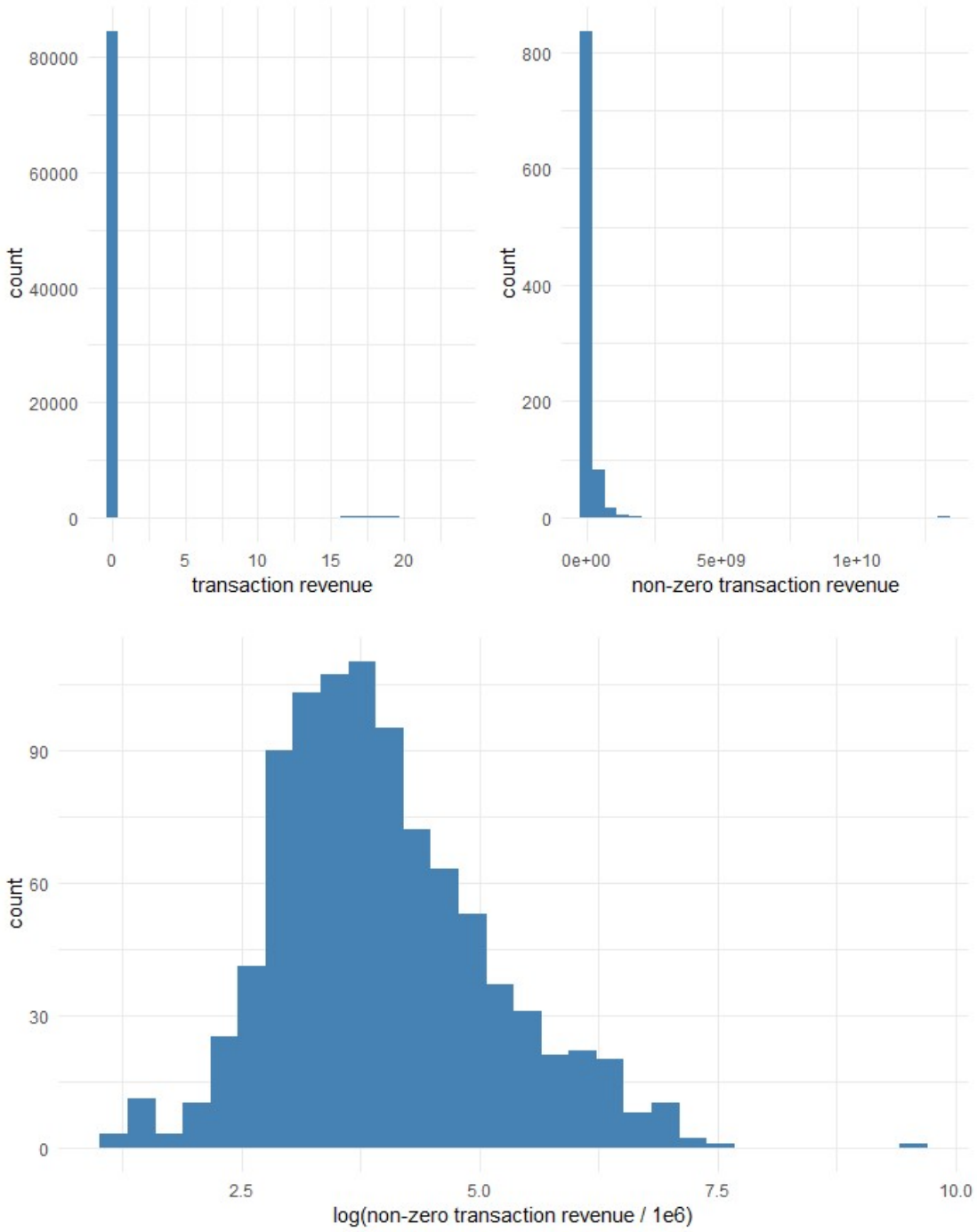
Then, as we want to do some unsupervised method, we extract our response variable, Transaction Revenue, out from the train set and set a vector Y to store this value to be used in the later analysis.

The visualization of data set:

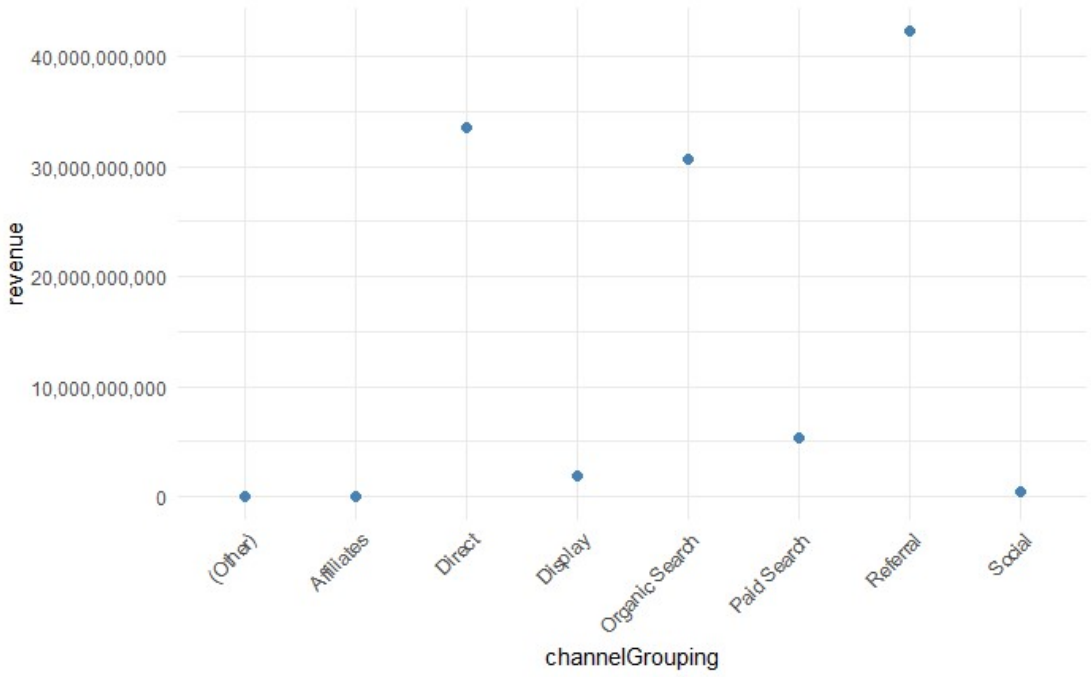
The following is the plot of the percentages of missing data. About 12 variables have more than 85% missing data.



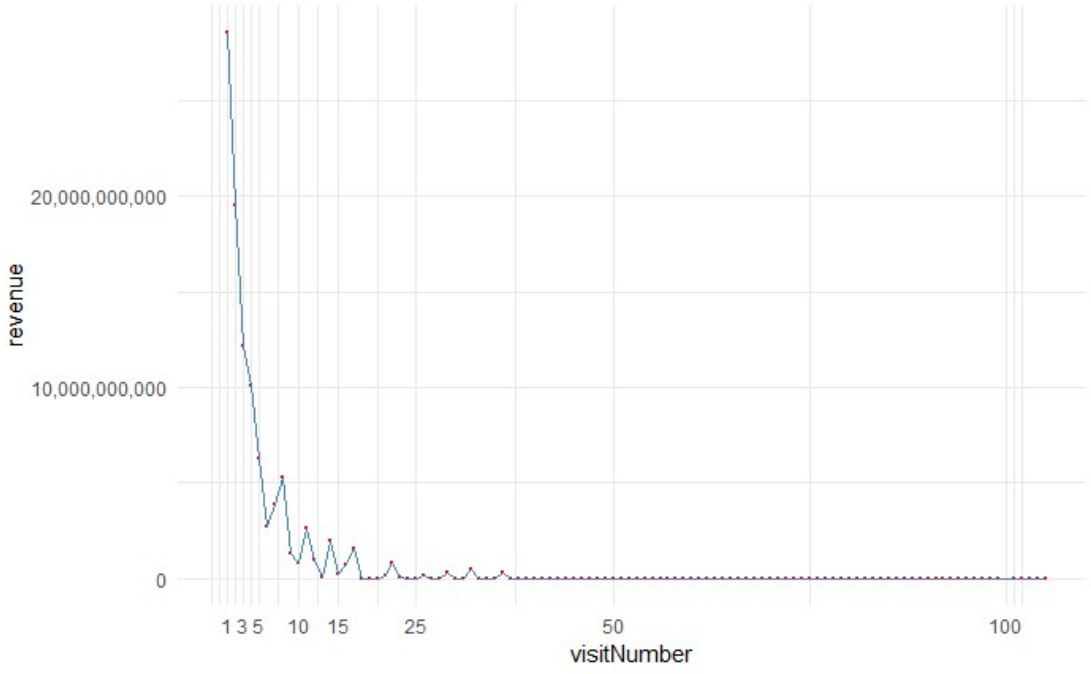
These are the plot of transaction revenue and its log1p. The target variable has a wide range of values. We will use log-transformed target. Only 2% of all transactions are not zero.

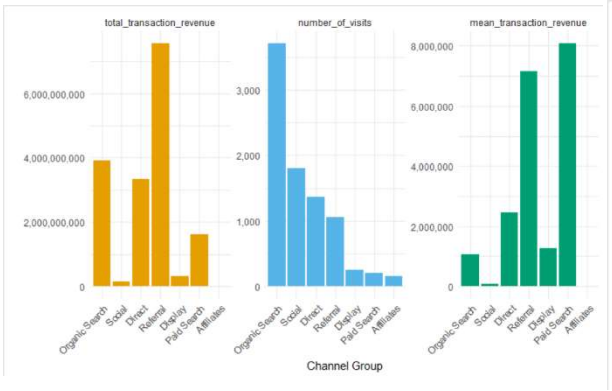


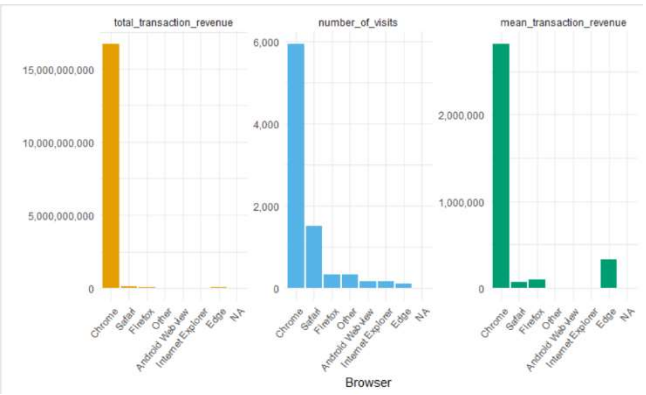
The figure shows that users who use Affiliates, Social and other channels do not generate revenue. Referral has the highest revenue.

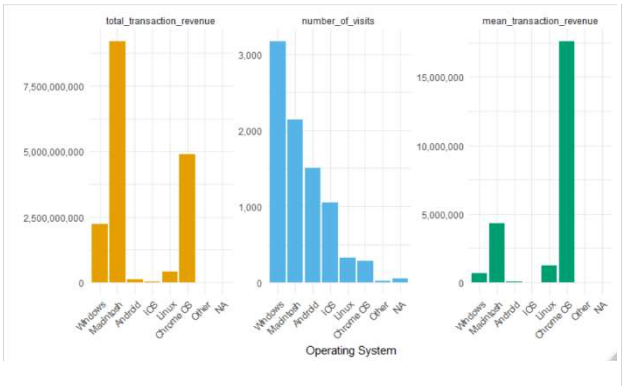


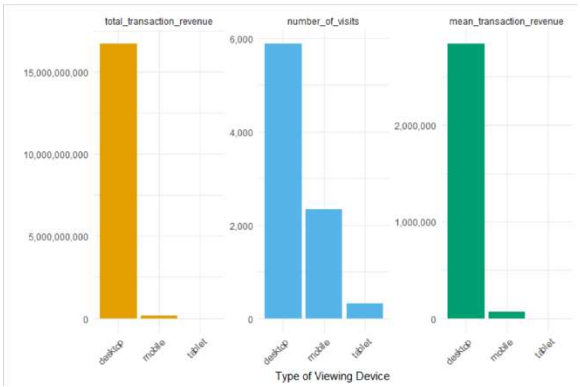
From this plot we can see the relationship between visitNumber and revenue. In general, fewer visit number means more total revenue:

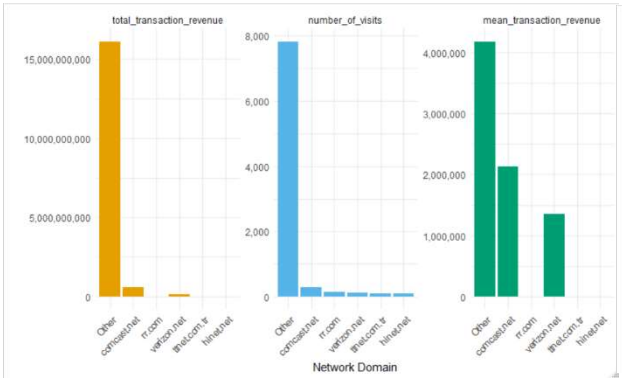


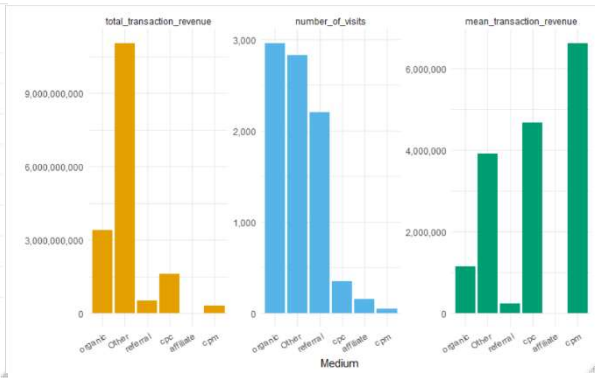

total_transaction_revenue
number_of_visits
mean_transaction_revenue
Organic Search
Social
Direct
Referral
Display
Paid Search
Affiliates
Channel Group

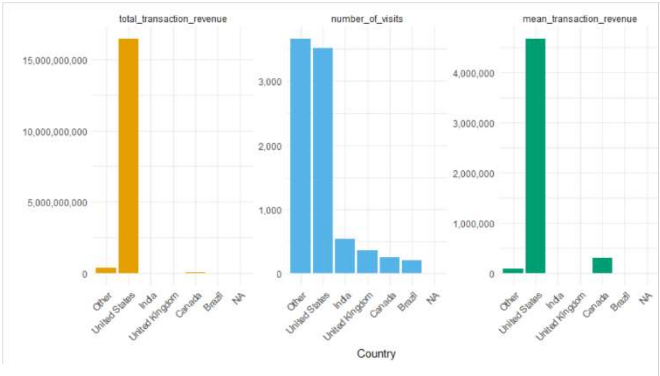

total_transaction_revenue
number_of_visits
mean_transaction_revenue
Chrome
Safari
Firefox
Other
Android Web View
Internet Explorer
Edge
NA
Browser

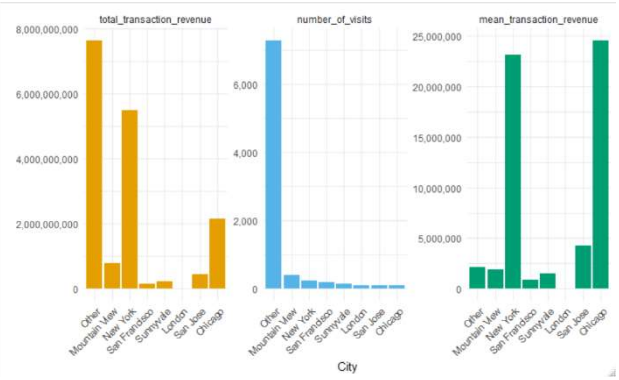

total_transaction_revenue
number_of_visits
mean_transaction_revenue
Windows
Macintosh
Android
iOS
Linux
Chrome OS
Other
NA
Operating System


total_transaction_revenue
number_of_visits
mean_transaction_revenue
desktop
mobile
tablet
Type of Viewing Device


total_transaction_revenue
number_of_visits
mean_transaction_revenue
Other
comcast.net
rr.com
verizon.net
ttnet.com.tr
hinet.net
Network Domain


total_transaction_revenue
number_of_visits
mean_transaction_revenue
organic
Other
referral
cpc
affiliate
cpm
Medium


total_transaction_revenue
number_of_visits
mean_transaction_revenue
Other
United States
India
United Kingdom
Canada
Brazil
NA
Country


total_transaction_revenue
number_of_visits
mean_transaction_revenue
Other
Mountain View
New York
San Francisco
Sunnyvale
London
San Jose
Chicago
City

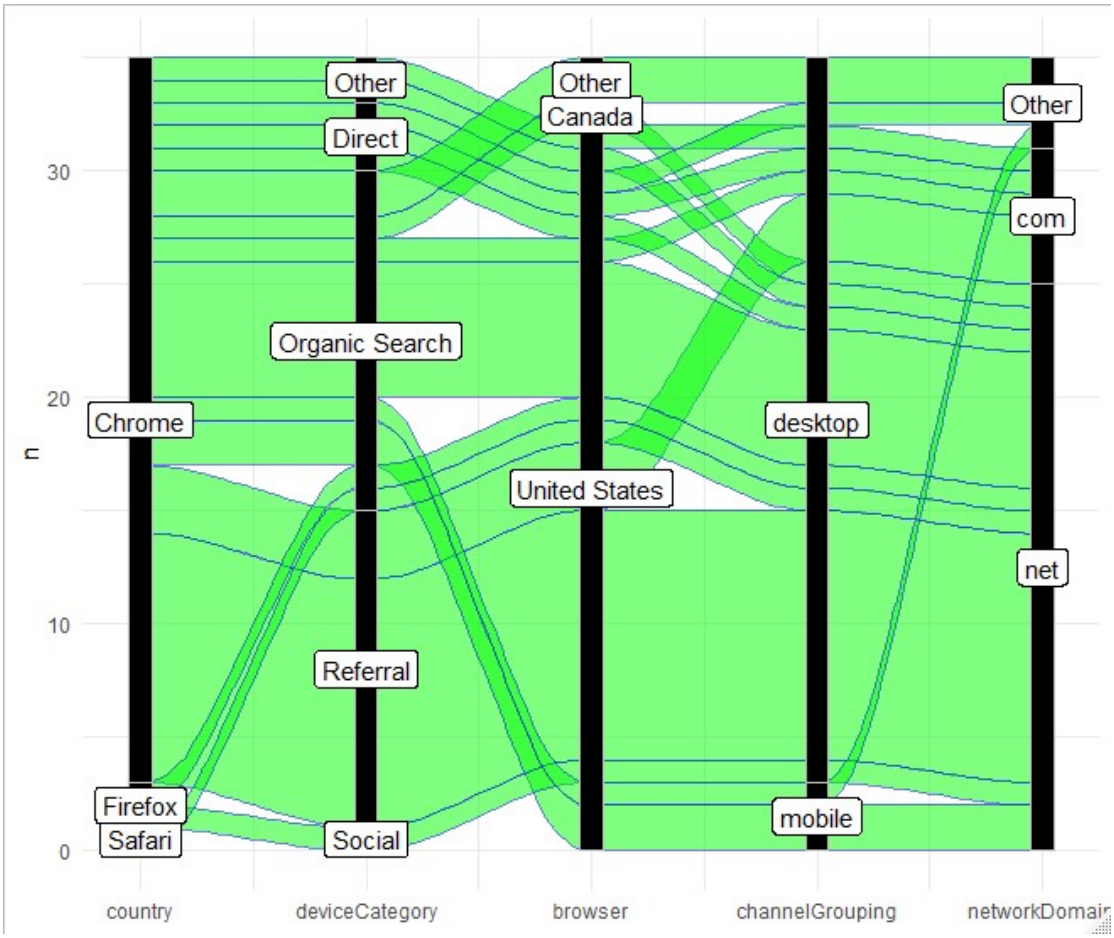
Looking the plots of distributions above we can find that:
The channels with the most visit is OrganicSearch. Chrome is the most popular browser (has more visit than all other browsers combined) and its users generate the highest revenue. Windows and MacOS are the most popular operating systems. MacOS users generate the highest total revenue. However, ChromeOS users yield the highest mean revenue. Among all three devices, desktops generate the most visit and revenue. Among all the countries here, though other countries may generate high number of visits, US contributes the most total revenue (higher than all other countries combined).

Correlations:

For the correlation, we firstly create a model matrix and filter the correlation coefficients which are larger than 0.9. We can see that there are 32 pairs which have coefficients larger than 0.9. However, most of them are between the dummy variables and it is difficult to judge whether or not we should delete these variables. The pairs between “isMobile” and “isTrueDirect” is 1.0 and they are both not dummy, so we should delete one of them in the following sections.

	Var1	Var2	value
13153	isMobile	isTrueDirect	1.0000000
11571	channelGroupingAffiliates	campaignData Share Promo	1.0000000
12481	channelGroupingAffiliates	mediumaffiliate	1.0000000
12570	campaignData Share Promo	mediumaffiliate	1.0000000
12091	channelGroupingAffiliates	sourcePartners	0.9996953
12180	campaignData Share Promo	sourcePartners	0.9996953
12574	sourcePartners	mediumaffiliate	0.9996953
9554	regionNew York	cityNew York	0.9969468
15719	bounces	timeOnSiteOther	0.9967959
14800	adwordsClickInfo.slotRHS	adwordsClickInfo.adNetworkTypeContent	0.9918634
14931	adwordsClickInfo.slotTop	adwordsClickInfo.adNetworkTypeGoogle Search	0.9910656
9559	metroNew York NY	cityNew York	0.9905221
9297	metroLondon	cityLondon	0.9903418
14668	adwordsClickInfo.page1	adwordsClickInfo.gclIdOther	0.9901379
16505	transactions1	totalTransactionRevenueOther	0.9880876
8904	regionNew York	metroNew York NY	0.9878985
15327	hits1	pageviews	0.9804037
14796	adContentGoogle Merchandise Store	adwordsClickInfo.adNetworkTypeContent	0.9684747
14276	adContentGoogle Merchandise Store	adwordsClickInfo.slotRHS	0.9625094
8643	regionEngland	metroLondon	0.9571699
12227	channelGroupingSocial	sourceyoutube.com	0.9547607
9293	regionEngland	cityLondon	0.9479160
7575	subContinentNorthern America	countryUnited States	0.9417451
9032	regionCalifornia	metroSan Francisco-Oakland-San Jose CA	0.9334546
11915	referralPath/analytics/web/	sourceanalytics.google.com	0.9308783
13488	mediumcpc	keyword0	0.9304497
6149	subContinentSouthern Asia	countryIndia	0.9302743

This kind of plot is useful for discovering of multi-feature interactions. The vertical size of each block is proportional to the frequency of the feature. As most of observation's response variable are zero in this data set, we plot the non-zero part of data in the following graph. This plot shows high flows from the US from desktops with Chrome.



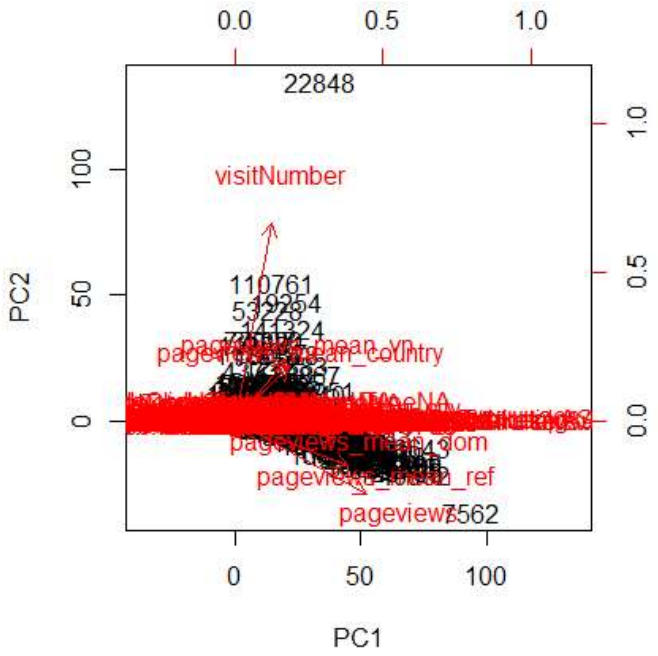
Unsupervised methods:

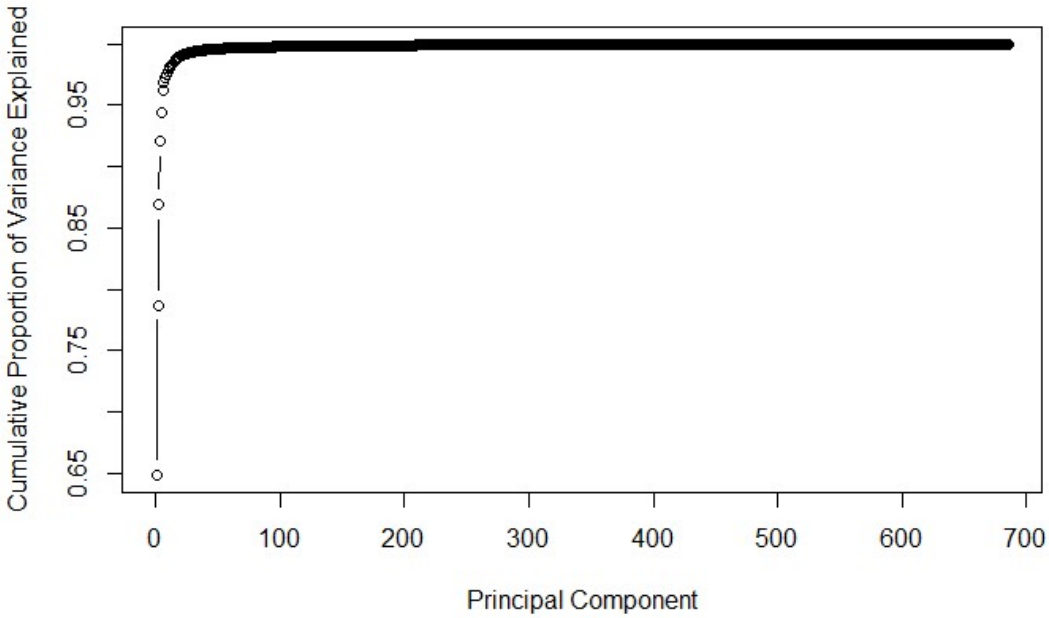
As our Kaggle competition's goal is to predict a continues variable, transaction Revenue, it is meaningless to do the clustering because it will not improve our final regression or other models' result. What we can do on unsupervised method is dimensional reduction, and we choose PCA to do it. From the result of PCA, we can see that two PCs can explain over 99% of variance. So, we can just choose PC1 and PC2.

Then we want to see the weight of our variables in these two PCs, from the table below we can see that "timeOnSite", "pageviews""and "totalTransactionRevenue" have relatively higher weights in PC1 and PC2. We may keep these 3 variables or pay more attention on them in the model building sections.

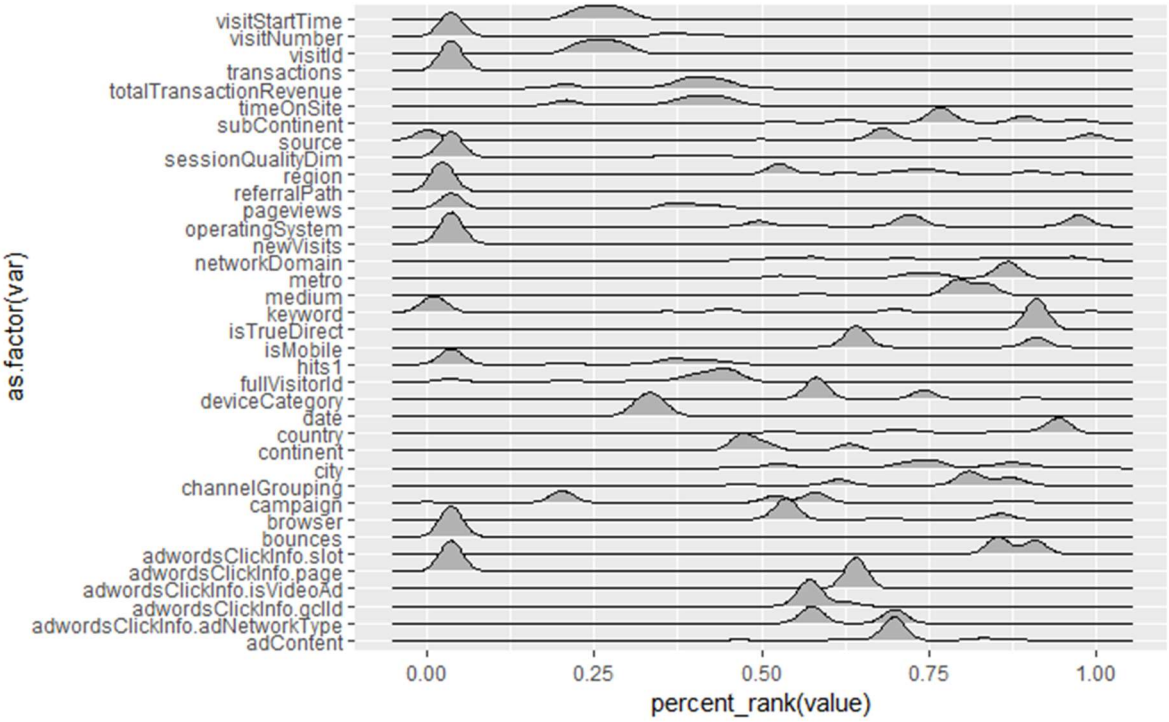
	PC1 <dbl>	PC2 <dbl>
pageviews	0.5581822	0.31041041
pageviews_mean_ref	0.4741803	0.18765077
pageviews_mean_dom	0.4221477	0.07087418
pageviews_mean_city	0.3552414	0.03980256
pageviews_mean_country	0.2274925	0.22904630
pageviews_mean_vn	0.2121461	0.25628263

6 rows





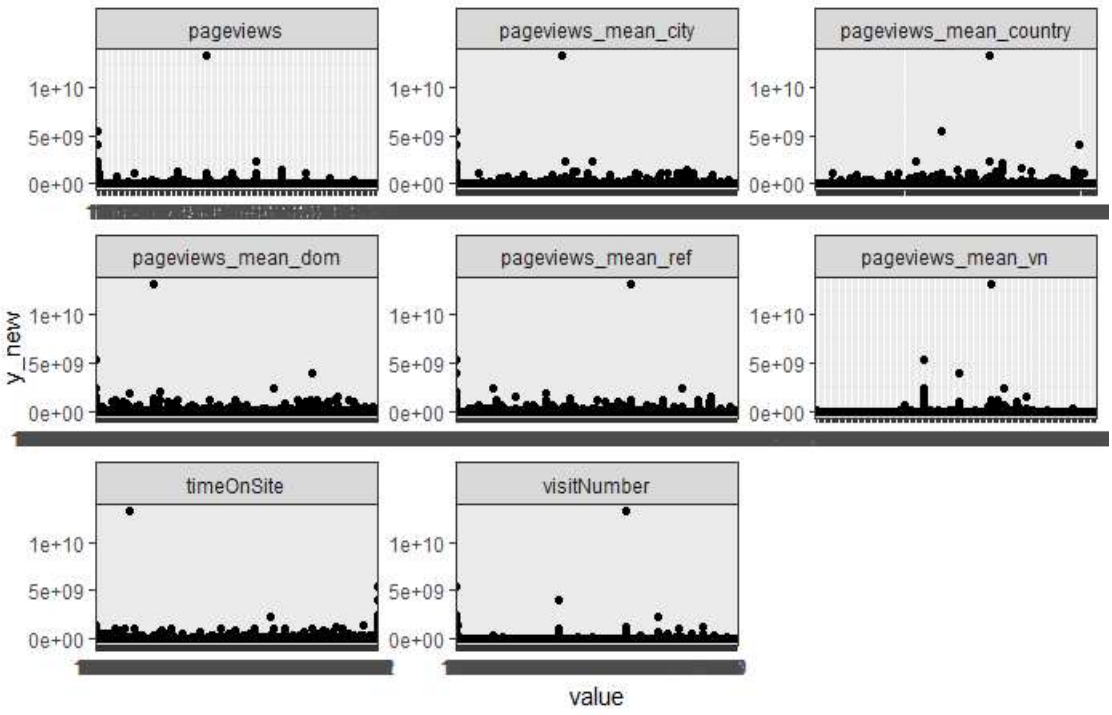
For an autoencoder to work well we have a strong initial assumption: that the distribution of variables for normal transactions is different from the distribution for fraudulent ones. We make some plots to verify this. Variables were transformed to a [0,1] interval for plotting.



Section B: Fitting Models

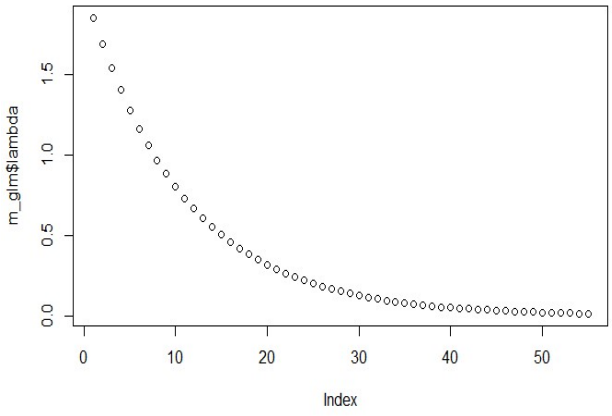
In the previous analysis, we add some additional variables pageviews_mean_vn, country, city, dom, ref, which are the group means of pageviews and visitNumber, country, city, networkDomain, referralPath.

We made some scatter plots of continuous variables and prepared to do some transformations for them. But it was hard to see the obvious pattern or trend. So, we might have to some random transformations such as log and sqrt.



Model1: GLMNET

We replace NA values with zeros and lump rare factor levels.
Parameter alpha=1 indicates lasso; family="gaussian".
The lowest lambda is 0.2035144.
The RMSE=0.1220766 is fairly well. But according to the coefficients, only "totalTransactionRevenueOther" and "transactionsOther" are significant, which does not make sense. We consider to apply other models to interpret the importance of variables.
[1] 0.1220766



Model2: keras

We use the same model matrix as before. Here's our structure of the keras model:

```
layer_dense(units = 256, activation = "relu", input_shape = ncol(X)) %>%
  layer_dropout(rate = 0.5) %>%
  layer_dense(units = 128, activation = "relu") %>%
  layer_dropout(rate = 0.25) %>%
  layer_dense(units = 1, activation = "linear")
```

The RMSE is 0.2234882, which is kind of bad compared with GLMNET model. We choose parameters as epochs = 50, batch_size = 128, verbose = 0, validation_split = 0.2. Since the result is not good, we give up to do more work on tuning parameters and making transformations on variables.

[1] 0.2234882

Model3: Xgboosting

In this case, we don't care about **NA** values, since XGB can handle them by default.

step1:

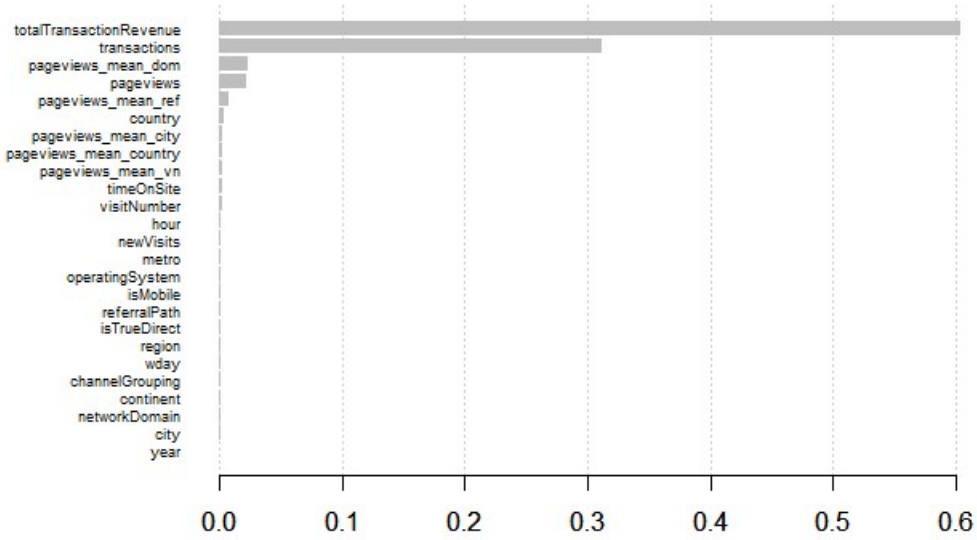
I fit all variables first. RMSE=0.1269855. We also plot the importance of variables. "transactions","totalTransactionRevenue" and "pageviews" are very important factors. Other factors may have small effect on revenue. We plot 25 of them to fit xgboosting again to see if RMSE can be improved.

The parameters arenthread = 4, eta = 0.05, max_depth = 7, min_child_weight = 5, gamma = 0, subsample = 0.8, colsample_bytree = 0.7, colsample_bylevel = 0.6, nrounds = 2000.

```
[1]     val-rmse:1.666551
Will train until val_rmse hasn't improved in 100 rounds.

[101]   val-rmse:0.115352
[201]   val-rmse:0.113845
Stopping. Best iteration:
[132]   val-rmse:0.112915

[1] 0.1269855
```



step2:

We change the parameters max_depth, min_child_weight and subsample first. The RMSE is now 0.124268.

[1] val-rmse:1.666584

Will train until val_rmse hasn't improved in 100 rounds.

[101] val-rmse:0.112623

[201] val-rmse:0.111072

Stopping. Best iteration:

[163] val-rmse:0.110804

[1] 0.124268

step3:

Then we select 25 most important factors. RMSE=0.1158011. It improves a little!

[1] 0.1158011

step4:

Then it's time to make some transformations on continuous variables (pageviews; visitNumber; pageviews_mean_vn; pageviews_mean_city; pageviews_mean_dom; pageviews_mean_ref) in the final xgboosting. I try log transformations.

I replace the original variable with its log transformation one by one. The results of RMSE are as follow:

pageviews:0.1190386

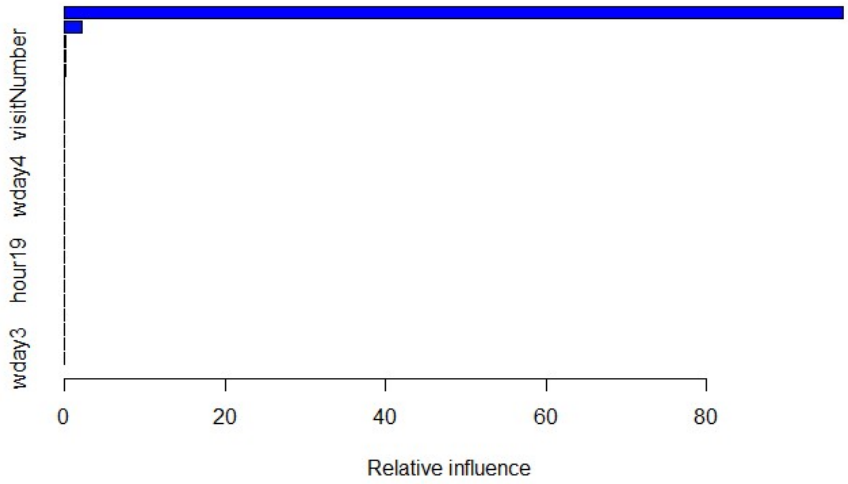
pageviews & visitNumber:0.1200756

pageviews & pageviews_mean_vn:0.1170185

pageviews & pageviews_mean_vn & pageviews_mean_city:0.1202961
pageviews & pageviews_mean_vn & pageviews_mean_dom:0.1196429
pageviews & pageviews_mean_vn & pageviews_mean_ref:0.1138189
We may conclude that make a log transformation on "pageviews & pageviews_mean_vn & pageviews_mean_ref" is good for our prediction.
The best result of RMSE in xgboosting is 0.1138189.

Model4: Gradient Boosting Machine

step1:
If we choose distribution = "gaussian", n.trees = 400 and interaction.depth = 8,
RMSE=0.1172499, which is the best so far. I get the summary and the plot that show some important variables.



	var <fctr>	rel.inf <dbl>
totalTransactionRevenueOther	totalTransactionRevenueOther	9.698525e+01
transactionsOther	transactionsOther	2.216460e+00
pageviews	pageviews	1.428097e-01
pageviews_mean_dom	pageviews_mean_dom	1.081817e-01
pageviews_mean_city	pageviews_mean_city	9.809915e-02
visitNumber	visitNumber	6.491014e-02
pageviews_mean_ref	pageviews_mean_ref	5.041003e-02
pageviews_mean_vn	pageviews_mean_vn	4.499652e-02
hourOther	hourOther	1.969443e-02
year2017	year2017	1.677303e-02

step2:
We choose the 27 most significant variables. The RMSE is 0.1172026.
[1] 0.1172026

step3:
Finally, we tune the parameters and choose n.trees = 300,interaction.depth = 7. The RMSE is 0.1156174.

[1] 0.1156174

Model5: Build an ensemble

The ensemble uses gbm to combine the predictions of my best models. However, I don't see a great improvement. The RMSE is 0.1173008, which doesn't have a big difference from the result of gbm.
[1] 0.1173008

Section C: Discussion

(a) Summary about our final model:

Our final model is the ensemble. In nature, it's a gbm model. The variables are the most important 27 features, which we select from the original gbm model, and predictions of other best models, such as xgboosting and glmnet. About the parameters We use `n.trees = 300`, `interaction.depth = 7`, `shrinkage = 0.1` and `n.minobsinnode = 20`.
The result of RMSE is around 0.117, which is fairly well. But we don't see a big difference from the results of gmb and xgboosting models.

(b) About future work:

Maybe we can also see how target variable changes over time. I think ARIMA model in time series is very powerful to predict the revenue if it truly has some trend in time.

Based on the diagram of Alluvial, we can add more interaction terms in the furture.

(c) Hurdles and problems:

In the data cleaning process, we fail to solve the format of two variables. They have similar format as JSON but can't be seperated by using "jsonlite" or "rjson" package.
When use "select" and "summarise" in package dplyr, sometimes it doesn't work correctly. I don't know why but when I use "dplyr:::" and "dplyr:", the problem is solved.
When fitting the random forest model, it takes a long time to run the "Tuning parameters" code and knit. Although we write the code of tuning parameters in "caret", sometimes we have to change one or two parameters by hand.
When we use PCA to do dimension reduction, it's hard to interpret the importance of variables because of the large number of factors and dummy variables. So, we use the sorted rotation scores to choose important factors.