Google Analytics G-Store Purchasing EDA and Speculative Feature Selection

Dylan Wiwad

2018-09-17

Bringing in the data

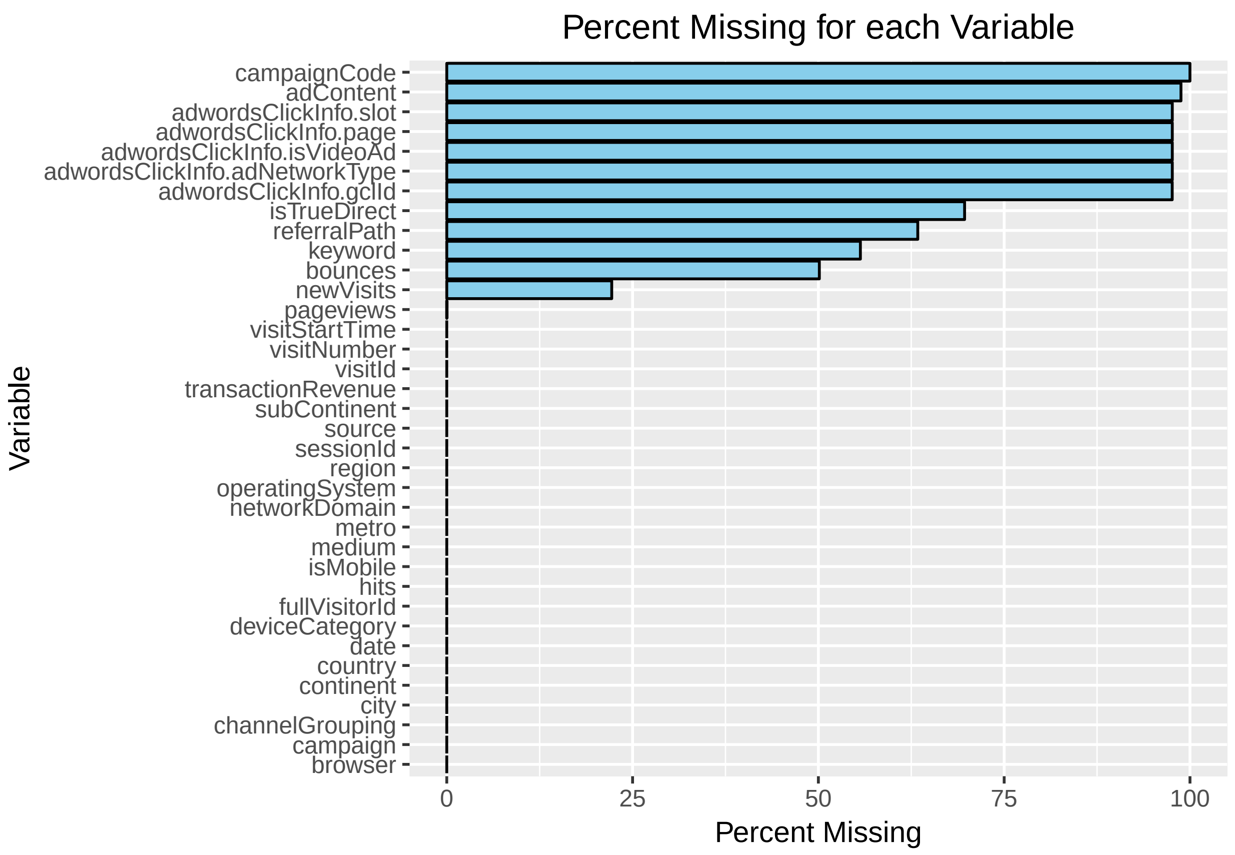
I already used a kernel (credit to @ML) that flattened the data and pulled everything meaningful out of the handful of JSON columns. So in the below chunk, I just bring in this flattened data and also just remove all of the columns that are singular - no use in trying to use a predictor with 0 variance.

```r
knitr::opts_chunk$set(echo = TRUE)
# Set the wd and grab the flattened data
setwd("/users/dylanwiwad/downloads/all/")

# Training data - used a script from the Kernel to flatten the JSON columns
train <- read.csv("train_flat.csv", colClasses=c("fullVisitorId"="character"))
test <- read.csv("test_flat.csv", colClasses=c("fullVisitorId"="character"))

# Right off the bat, I'm going to remove any columns that have singular data
# (e.g., only one value); no variance = no predictive power
train <- Filter(function(x)(length(unique(x))>1), train)
test <- Filter(function(x)(length(unique(x))>1), test)

# Just so we have it right away, let's compute our DV. There is lot's of NA's in our
# DV that should actually be zero - the person spent no money. Let's turn all the NAs into 0s
train$transactionRevenue[is.na(train$transactionRevenue)] <- 0
```

Missingness

First things first, I’m just going to visualize the amount of missing data, in percentages.

```r
missingness <- sapply(train, function(x) sum(is.na(x)))
missingness <- as.data.frame(missingness)
# Get  percentages for a nice viz
missingness$perc <- (missingness$missingness / 903653)*100

library(ggplot2)
missingness$var <- rownames(missingness)
missingness <- missingness[order(-missingness$perc),]
ggplot(data=missingness, aes(x=reorder(var, perc), y=perc)) +
  geom_bar(stat="identity", color="black", fill="skyblue") +
  coord_flip() + scale_colour_brewer(palette='Dark2') + xlab("Variable") +
  ylab("Percent Missing") + ggtitle("Percent Missing for each Variable") +
  theme(plot.title = element_text(hjust = 0.5))
```



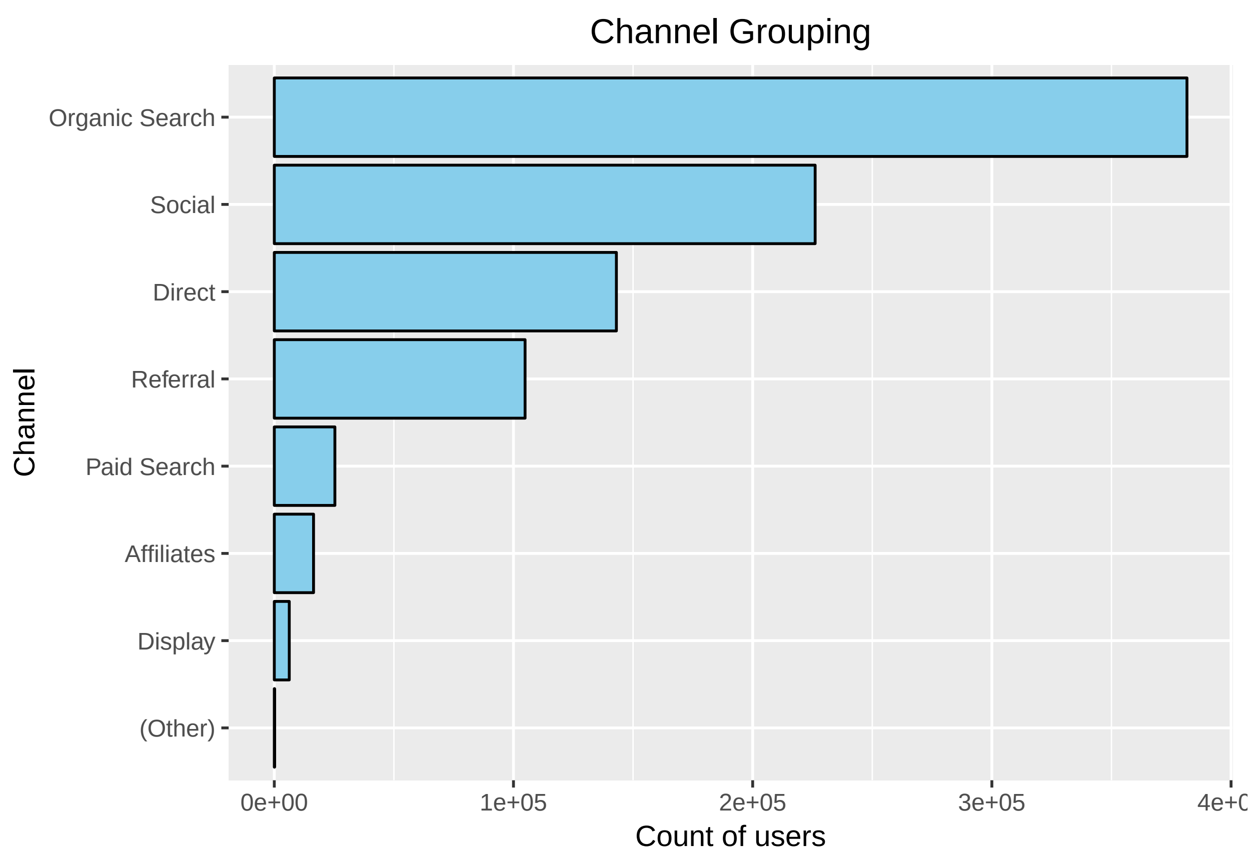
Okay so the variables we need to dive deeper into here are CampaignCode, adContent, all the adwords variables, isTrueDirect, referralPath, keyword, bounces, and newvisits. I'm going to dive a bit into these just to see what the levels are and if they can be converted into dummy variables that are meaningful. I've noticed that a lot of the kernels where people are posting their EDA, they are simply deleting these "adwords" columns. I think that is a bit of a mistake - I think the missingness will be meaningful in some cases.

Here is a website that describes what each and every one of these columns from google analytics:

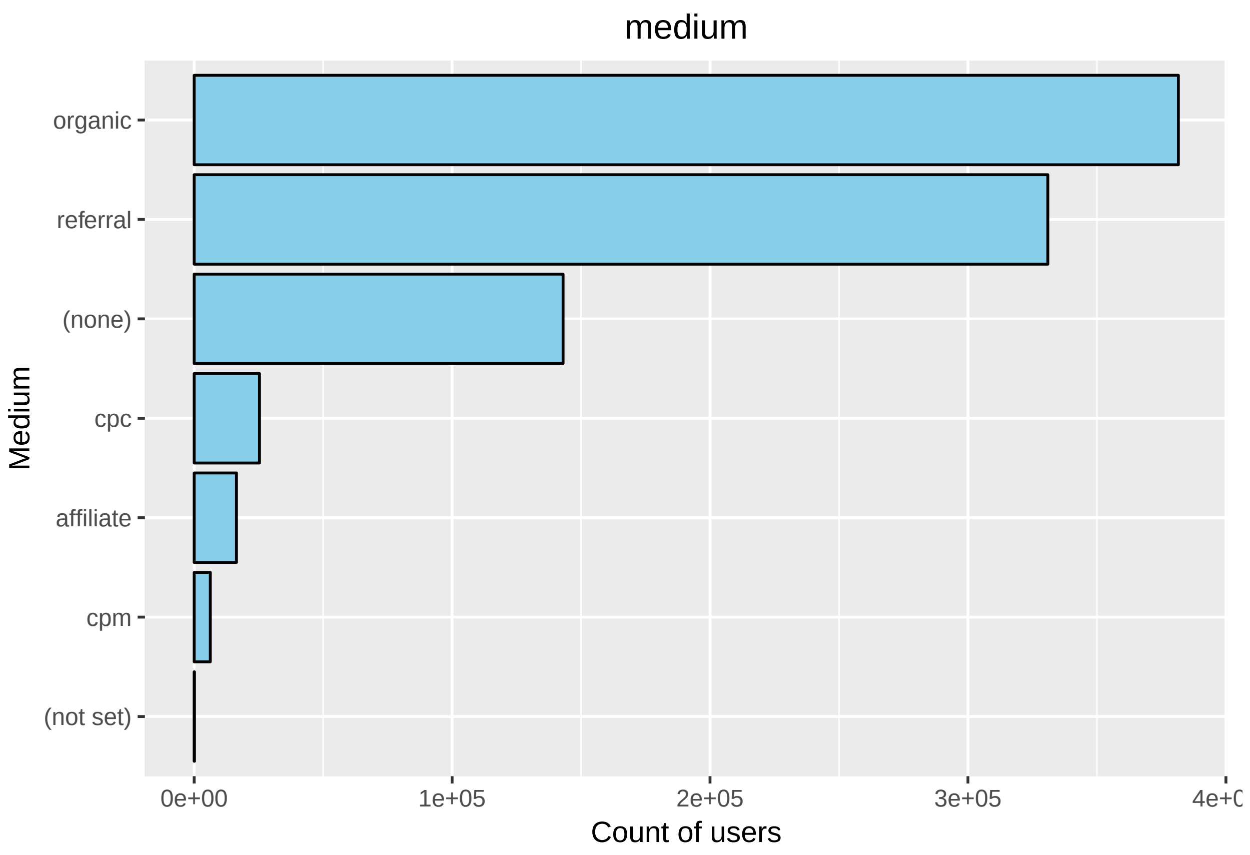
https://support.google.com/analytics/answer/3437719?hl=en

ChannelGrouping and Medium

```r
i <- ggplot(as.data.frame(plyr::count(train$channelGrouping)), aes(x=reorder(x, freq), y=freq)) +
  geom_bar(stat="identity", color="black", fill="skyblue") +
  coord_flip() + xlab("Channel") +
  ylab("Count of users") + ggtitle("Channel Grouping") +
  theme(plot.title = element_text(hjust = 0.5))
i
```



```
i <- ggplot(as.data.frame(plyr::count(train$medium)), aes(x=reorder(x, freq), y=freq)) +
  geom_bar(stat="identity", color="black", fill="skyblue") +
  coord_flip() + xlab("Medium") +
  ylab("Count of users") + ggtitle("medium") +
  theme(plot.title = element_text(hjust = 0.5))
i
```

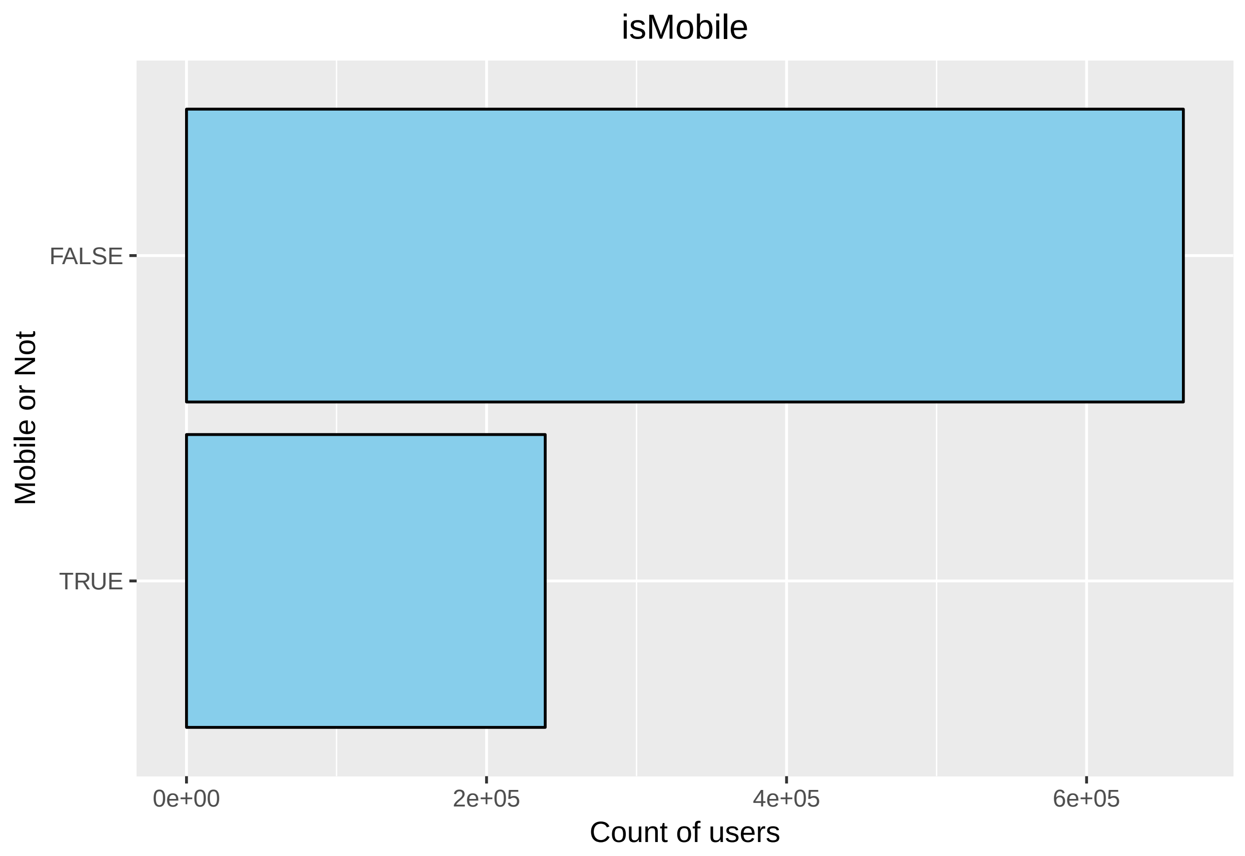


So this variable tells us the channel the spender came in from. Mostly, people are coming from an organic search, social media, directly typing in the website, or a referral. Regardless, this is a useful column. It's pretty clear from this that medium is almost the same as channel grouping, with the latter being more descriptive. Medium is not useful here relative to channel grouping.

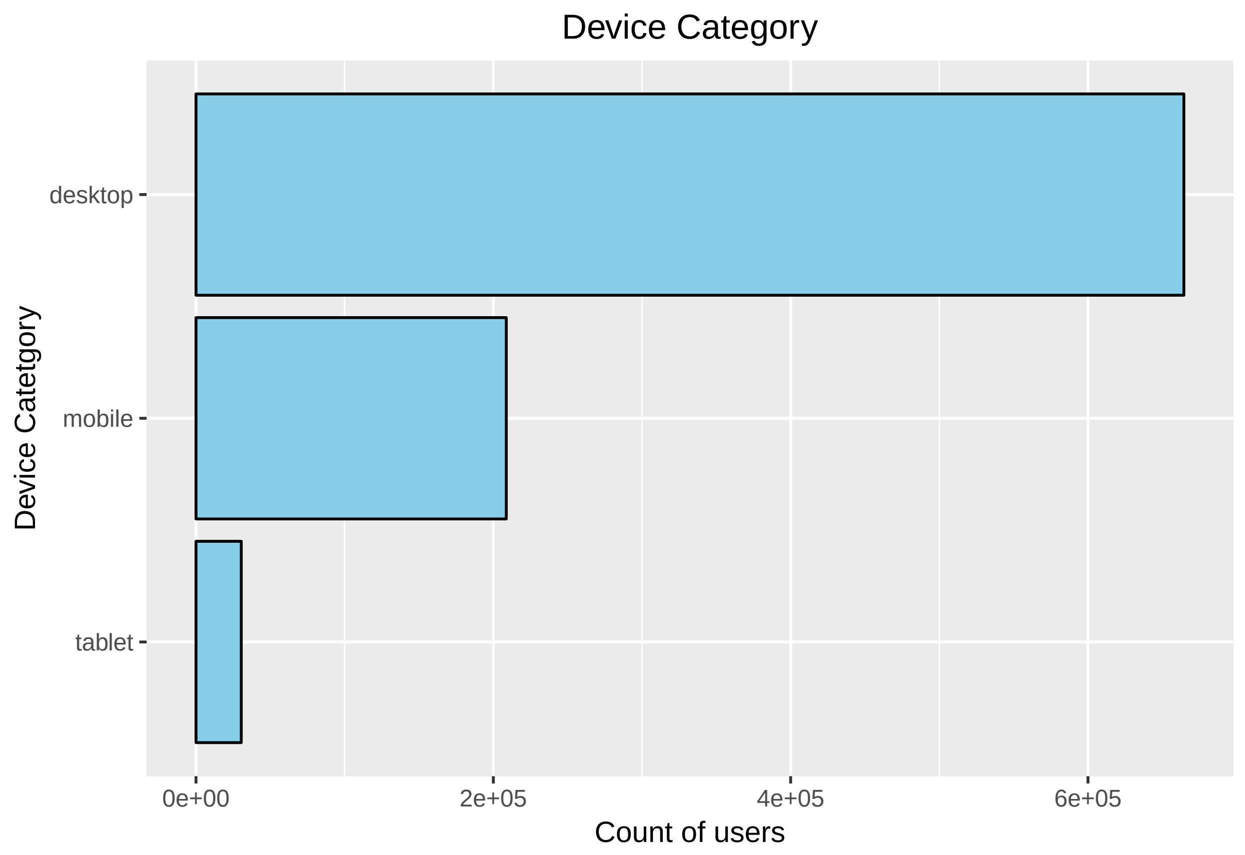
Device Category and Mobile

Again, I've learned this from some of my other EDA. isMobile is almost a direct subset of the device category:

```r
i <- ggplot(as.data.frame(plyr::count(train$isMobile)), aes(x=reorder(x, freq), y=freq)) +
  geom_bar(stat="identity", color="black", fill="skyblue") +
  coord_flip() + xlab("Mobile or Not") +
  ylab("Count of users") + ggtitle("isMobile") +
  theme(plot.title = element_text(hjust = 0.5))
i
```



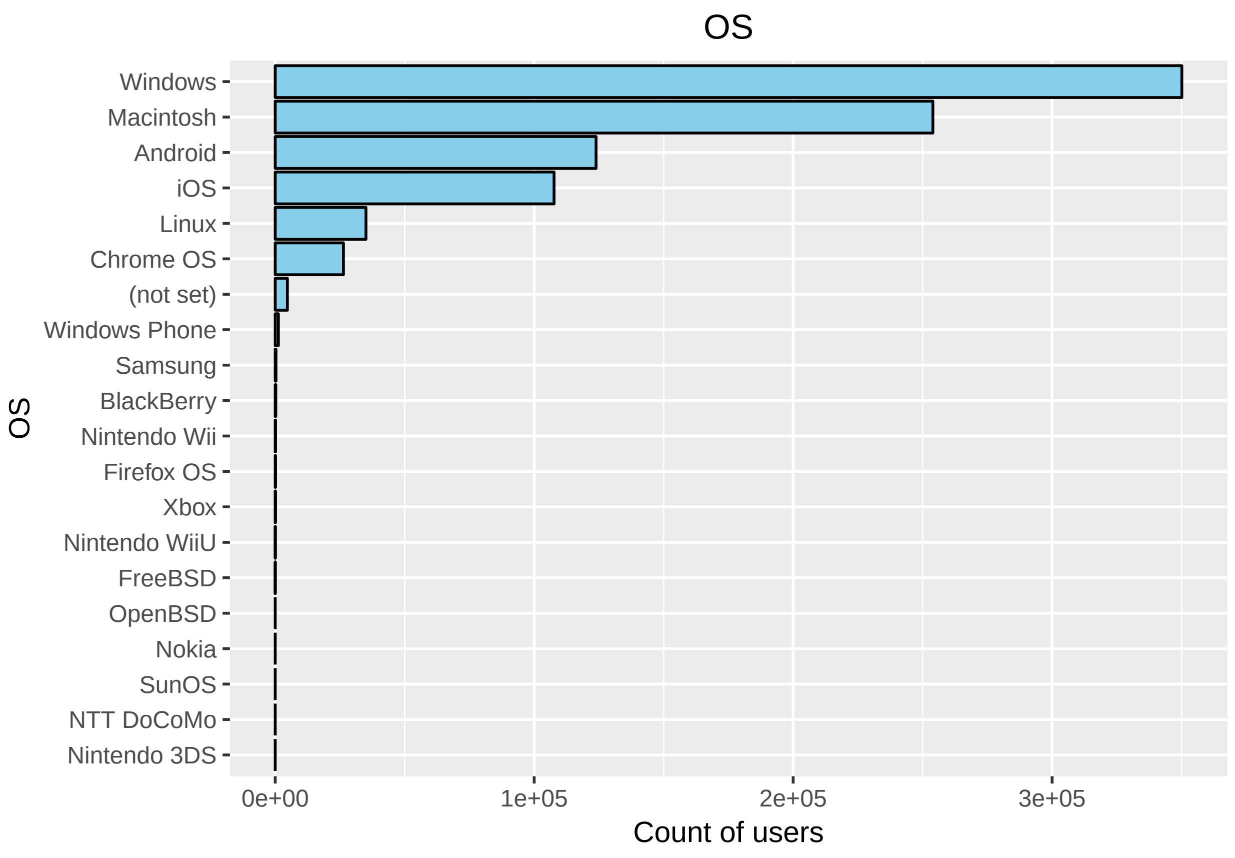
```
i <- ggplot(as.data.frame(plyr::count(train$deviceCategory)), aes(x=reorder(x, freq), y=freq)) +
  geom_bar(stat="identity", color="black", fill="skyblue") +
  coord_flip() + xlab("Device Catetgory") +
  ylab("Count of users") + ggtitle("Device Category") +
  theme(plot.title = element_text(hjust = 0.5))
i
```



If you look at the actual counts, you'll see that "TRUE" for is mobile is simply mobile + tablet in device category. Thus, we don't need to keep isMobile.

Operating System

```r
i <- ggplot(as.data.frame(plyr::count(train$operatingSystem)), aes(x=reorder(x, freq), y=freq)) +
  geom_bar(stat="identity", color="black", fill="skyblue") +
  coord_flip() + xlab("OS") +
  ylab("Count of users") + ggtitle("OS") +
  theme(plot.title = element_text(hjust = 0.5))
i
```



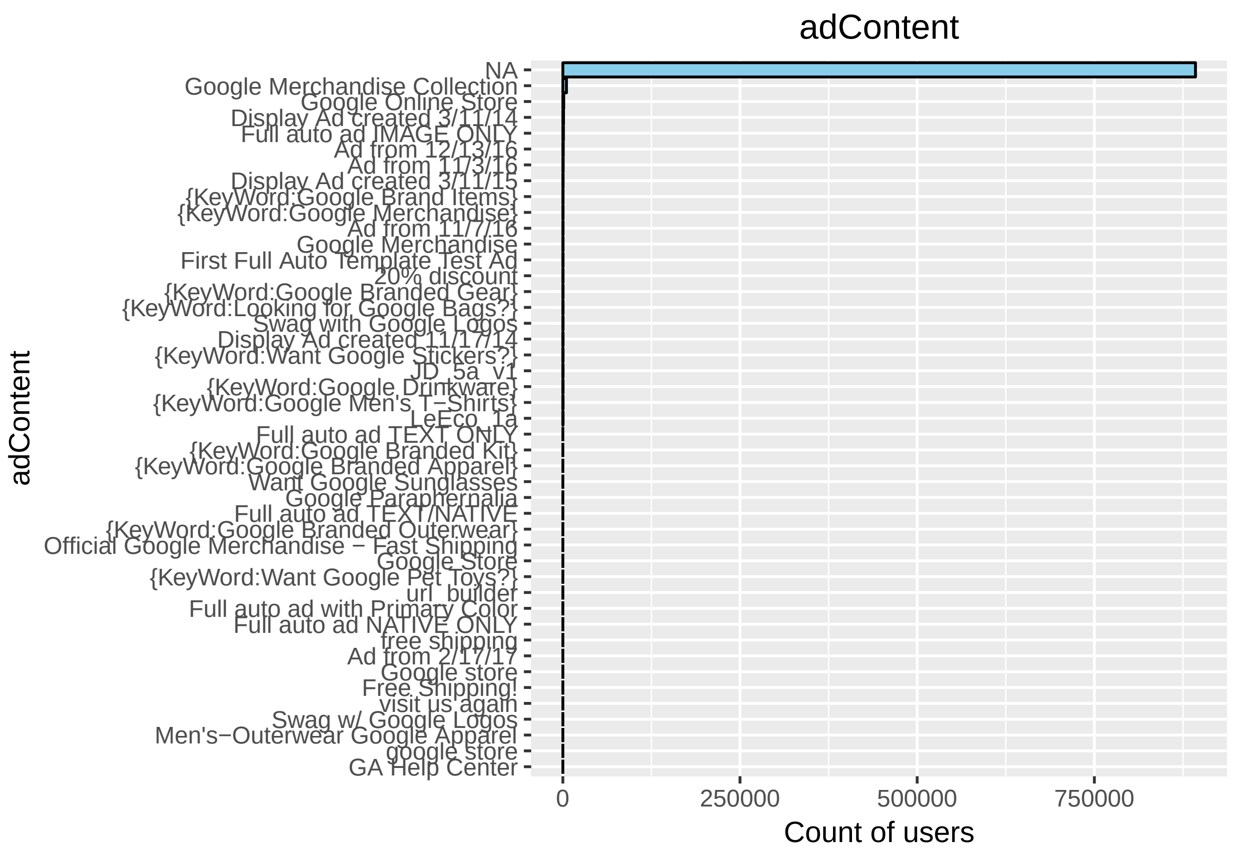
Most users are on Windows and Mac. Not surprising, given most users were on desktop. **So, for this, do we use OS, or do we just create a couple dummy coded vars for the big ones? More importantly, will this tell us anything that desktop versus mobile won't tell us?**

adwords Variables

Again - A few of these are actually meaningful! Let's quickly cycle through.

adContent

```r
i <- ggplot(as.data.frame(plyr::count(train$adContent)), aes(x=reorder(x, freq), y=freq)) +
  geom_bar(stat="identity", color="black", fill="skyblue") +
  coord_flip() + xlab("adContent") +
  ylab("Count of users") + ggtitle("adContent") +
  theme(plot.title = element_text(hjust = 0.5))
i
```

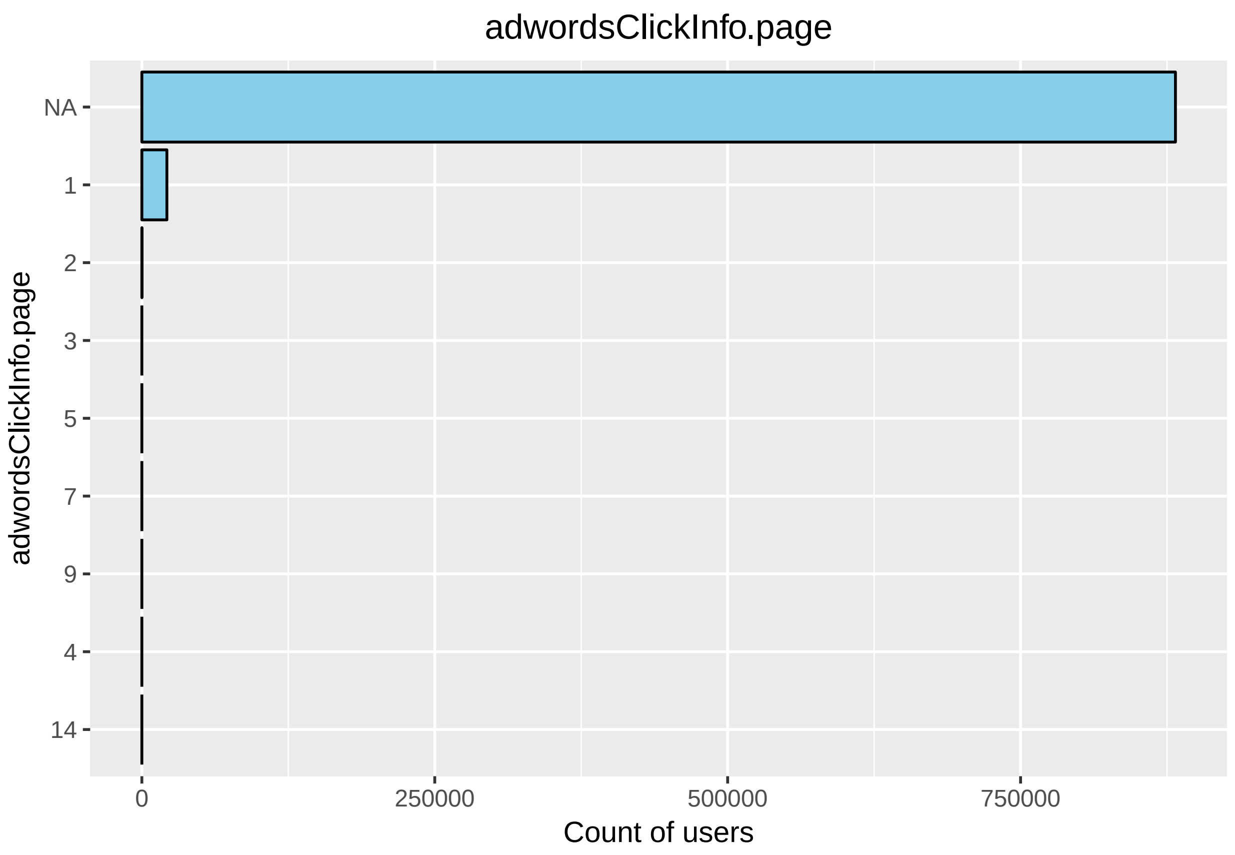


This seems far too noisy, and too full of missingness to be useful. Tthere is nothing worth exploring here.

page

“Page number in search results where the ad was shown” - mostly NA, not very useful.

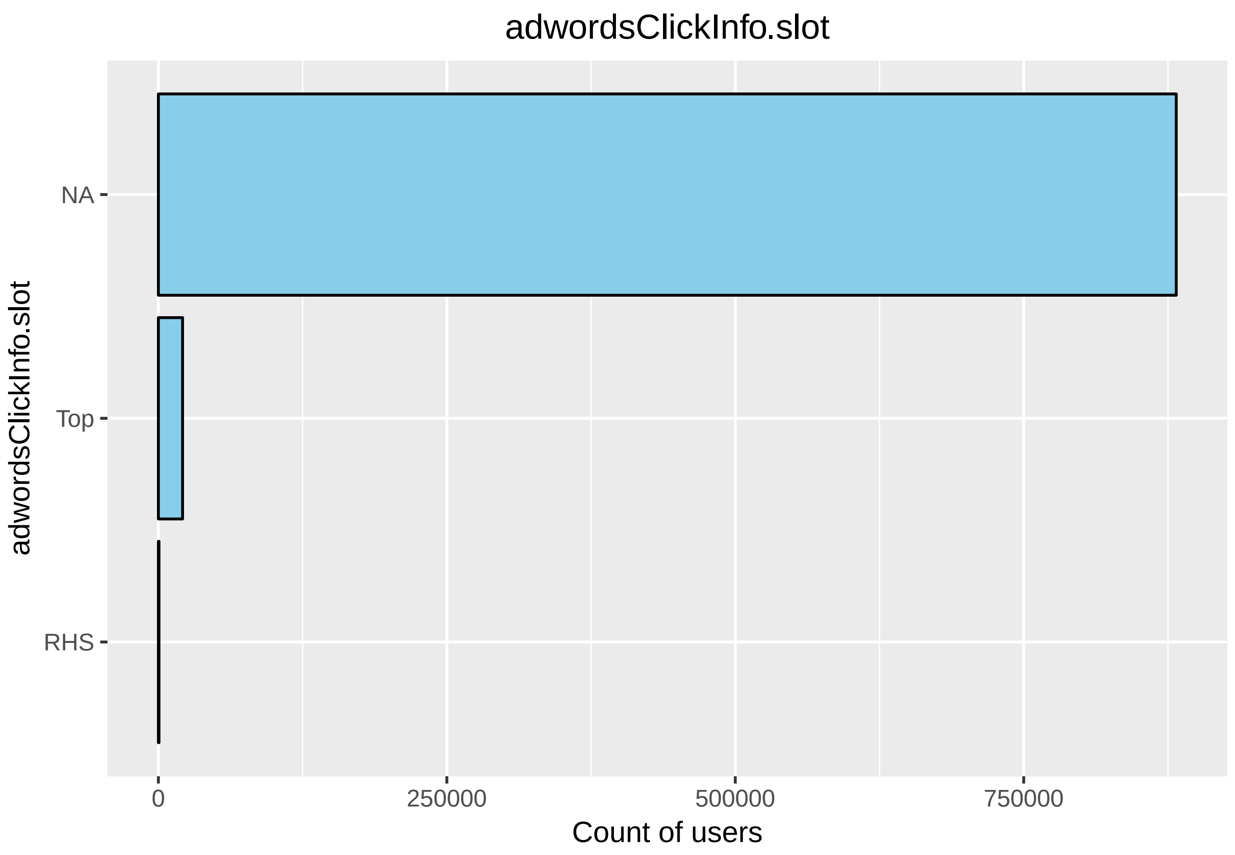
```r
i <- ggplot(as.data.frame(plyr::count(train$adwordsClickInfo.page)), aes(x=reorder(x, freq), y=freq)) +
  geom_bar(stat="identity", color="black", fill="skyblue") +
  coord_flip() + xlab("adwordsClickInfo.page") +
  ylab("Count of users") + ggtitle("adwordsClickInfo.page") +
  theme(plot.title = element_text(hjust = 0.5))
i
```



### slot

This is a randomly assigned metric - does the ad appear on the right hand side or the top of the page? This could be somewhat meaningful, but the data is almost entirely missing. Two options here are just to ignore it or to dummy code the Top and RHS categories and use those.

```
i <- ggplot(as.data.frame(plyr::count(train$adwordsClickInfo.slot)), aes(x=reorder(x, freq), y=freq)) +
  geom_bar(stat="identity", color="black", fill="skyblue") +
  coord_flip() + xlab("adwordsClickInfo.slot") +
  ylab("Count of users") + ggtitle("adwordsClickInfo.slot") +
  theme(plot.title = element_text(hjust = 0.5))
i
```



### gclId

This is the “Google click Id” - I’m not really sure what this is, and I don’t think it’s very useful due to the missingness. I can’t even plot this one because there is a million unique values. But as you can see, its almost entirely missing.

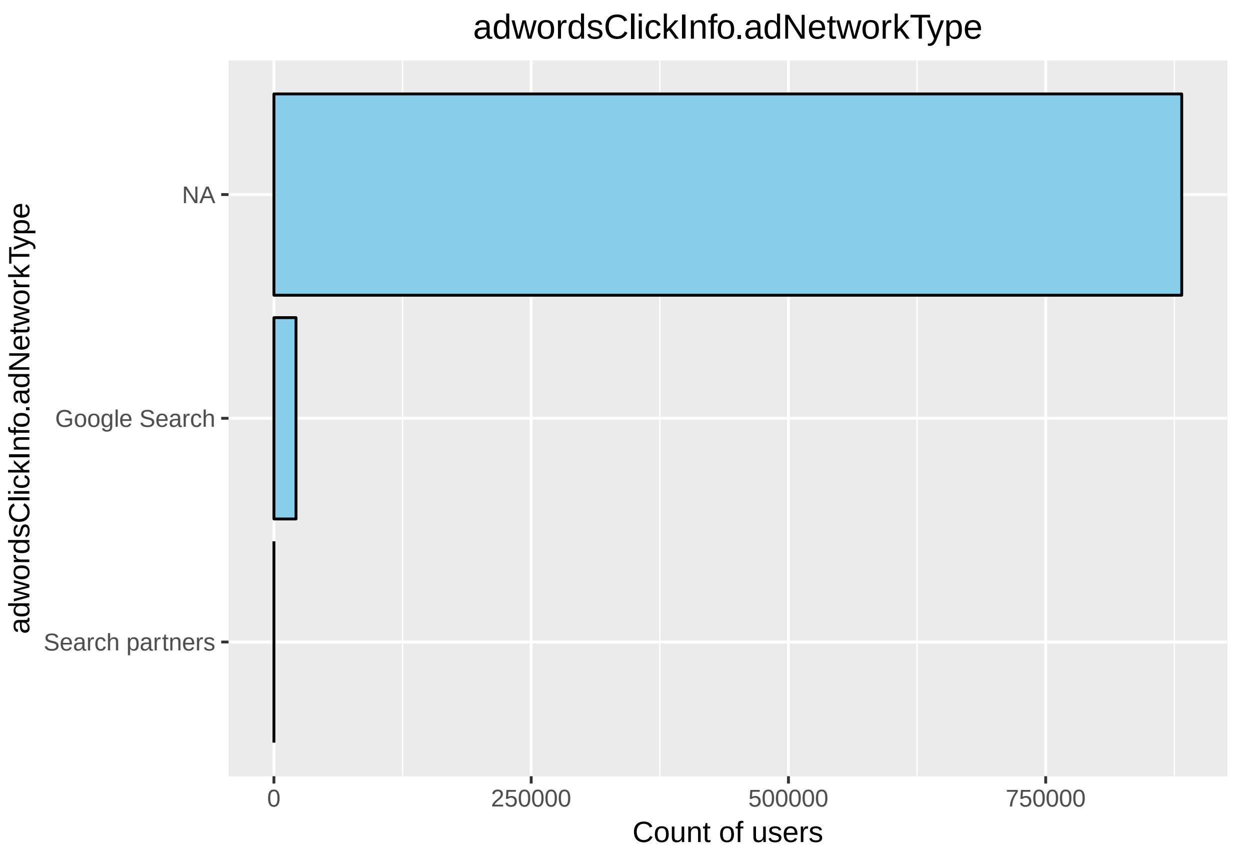
```
sum(is.na(train$adwordsClickInfo.gclId))
```

```
## [1] 882092
```

adNetworkType

Where the ad originated - Google or somewhere else. Most were not from ads.

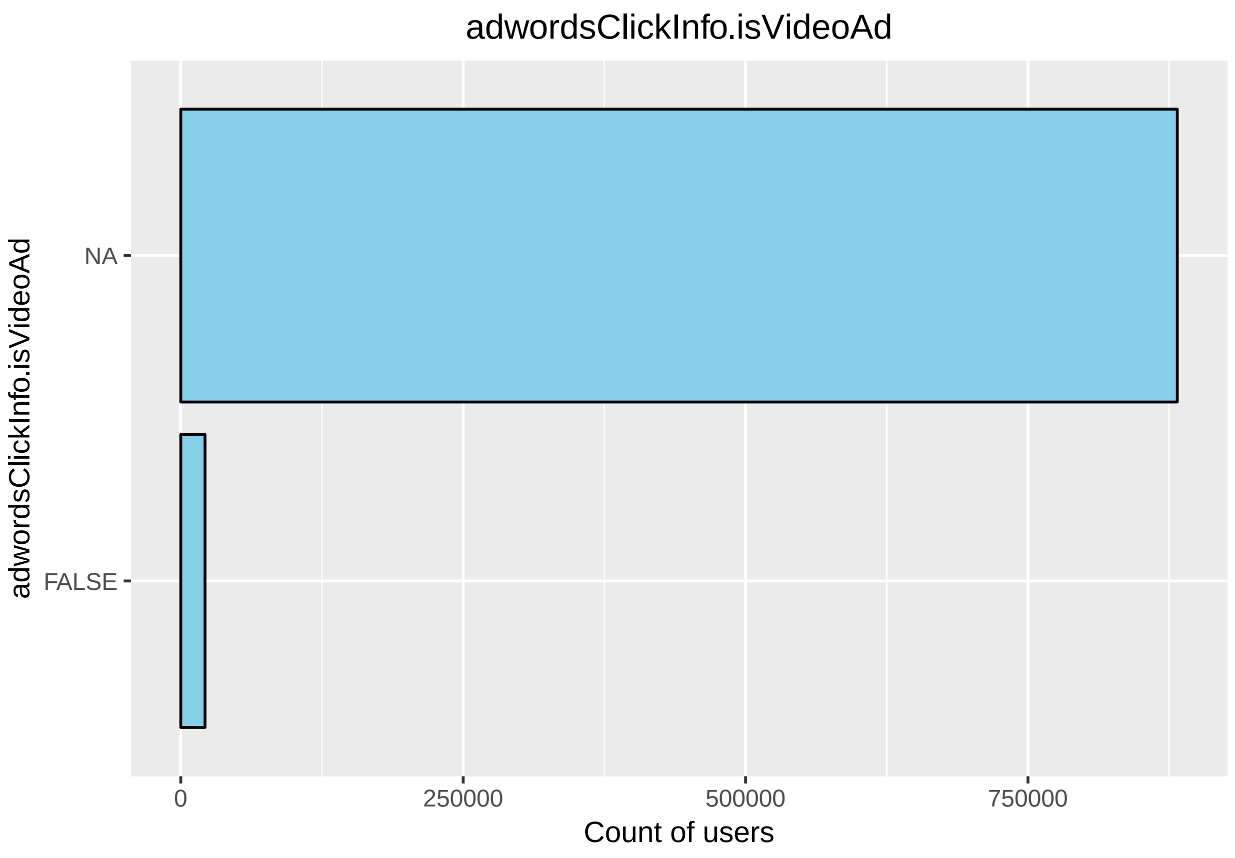
```
i <- ggplot(as.data.frame(plyr::count(train$adwordsClickInfo.adNetworkType)), aes(x=reorder(x, freq), y
  geom_bar(stat="identity", color="black", fill="skyblue") +
  coord_flip() + xlab("adwordsClickInfo.adNetworkType") +
  ylab("Count of users") + ggtitle("adwordsClickInfo.adNetworkType") +
  theme(plot.title = element_text(hjust = 0.5))
i
```



isVideoad

For the ones that were ads, were they video or not.

```r
i <- ggplot(as.data.frame(plyr::count(train$adwordsClickInfo.isVideoAd)), aes(x=reorder(x, freq), y=fre
  geom_bar(stat="identity", color="black", fill="skyblue") +
  coord_flip() + xlab("adwordsClickInfo.isVideoAd") +
  ylab("Count of users") + ggtitle("adwordsClickInfo.isVideoAd") +
  theme(plot.title = element_text(hjust = 0.5))
i
```



What did we learn from the adcontent variables?

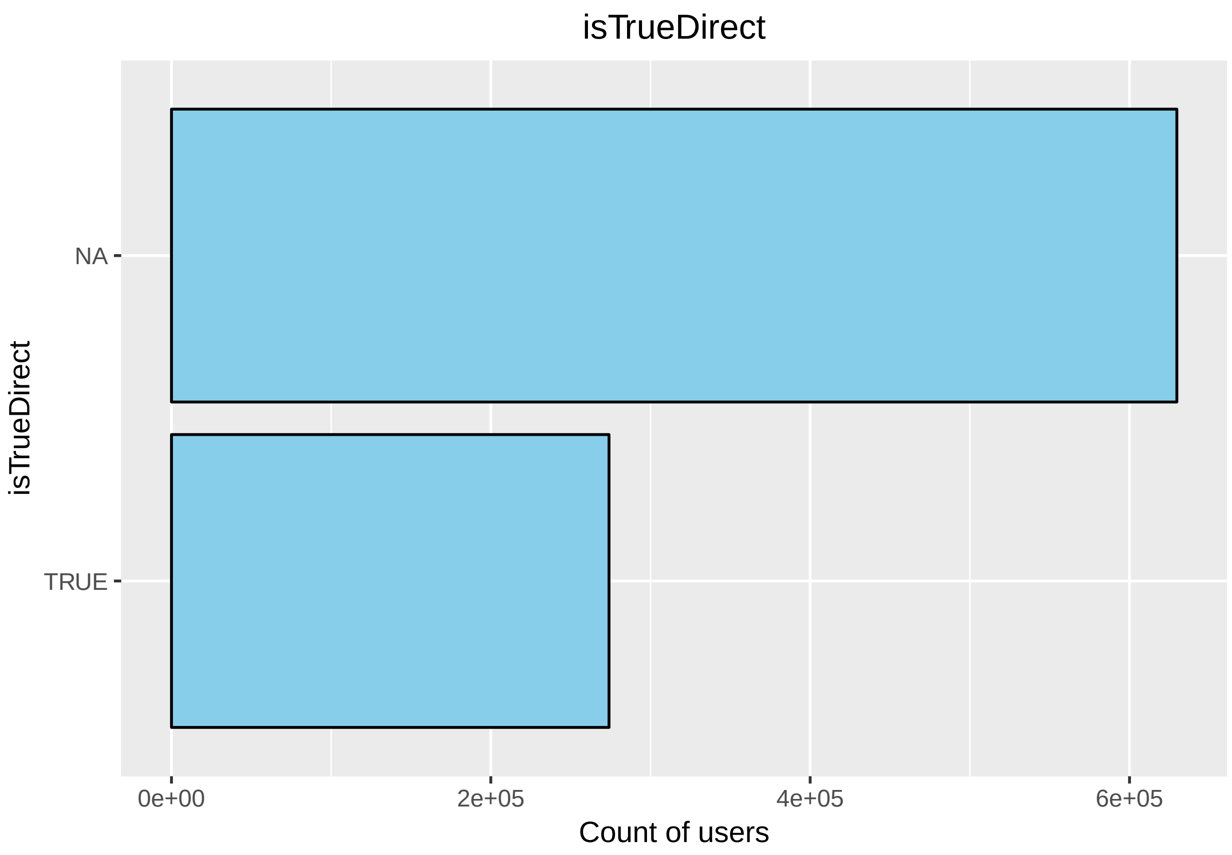
This is less clear in the bar graphs, more clear if you look at the actual frequencies, but there were 21,460 users who came to the store through an advertisement. This is clear through the exactly same frequencies in a few of these variables. Therefore, I'm going to use this last variable (video ad or not) to create a column that just indicates whether or not the person came to the store through an ad. There were no video ads, so all the "False" values are every visitor who came through an ad.

```r
train$ad_or_not <- ifelse(is.na(train$adwordsClickInfo.isVideoAd), c(0), c(1)) # 1 = ad
test$ad_or_not <- ifelse(is.na(test$adwordsClickInfo.isVideoAd), c(0), c(1)) # 1 = ad
```

isTrueDirect

This categorizes whether or not the person directly typed in the URL of the site. I think if this is the case they are much more likely to be intending to buy. Here, according to google, the NAs are false, so they are clearly meaningful data. Convert this into a binary and remove the original, we won't need it.

```r
i <- ggplot(as.data.frame(plyr::count(train$isTrueDirect)), aes(x=reorder(x, freq), y=freq)) +
  geom_bar(stat="identity", color="black", fill="skyblue") +
  coord_flip() + xlab("isTrueDirect") +
  ylab("Count of users") + ggtitle("isTrueDirect") +
  theme(plot.title = element_text(hjust = 0.5))
i
```



```
train$direct_click <-  ifelse(is.na(train$isTrueDirect), c(0), c(1))
train$isTrueDirect <- NULL
test$direct_click <-  ifelse(is.na(test$isTrueDirect), c(0), c(1))
test$isTrueDirect <- NULL
```

referralPath

According ot the analytics website this refers to the path they came from, it's completely redundant with source. Again, Can't visualize it because there is a bajillion unique referral paths.

They could possible be aggregated? Like things that come from gmail etc.

```
head(plyr::count(train$referralPath))
```

```
##
## 1
## 2
## 3 /_/scs/mail-static/_/js/k=gmail.main.en_GB.M4uGaVCZ8dg.O/m=m_i,t/am=OosHBMD8v-8PxjGMAjLSByrM-57nm0
## 4 /_/scs/mail-static/_/js/k=gmail.main.en.OGPH2uMSzCM.O/m=m_i,t,it/am=OovHBMD8v-8PxrUMALLSByrM-_7nmE
## 5      /_/scs/mail-static/_/js/k=gmail.main.en.OneUhUgV9iO.O/m=m_i,t/am=OqtHBrD_7_3BuIZhQFf6at3573--
## 6   /_/scs/mail-static/_/js/k=gmail.main.en.4A652805qDM.O/m=m_i,t,it/am=OotHDjD_7_3BuIZhQFf6at3573--
##    freq
## 1 75523
## 2     1
## 3     1
## 4     1
## 5     1
## 6      1
```

Keyword

I'm not going to bother visualizing this. According to the google analytics website it's completely redundant with the medium column.

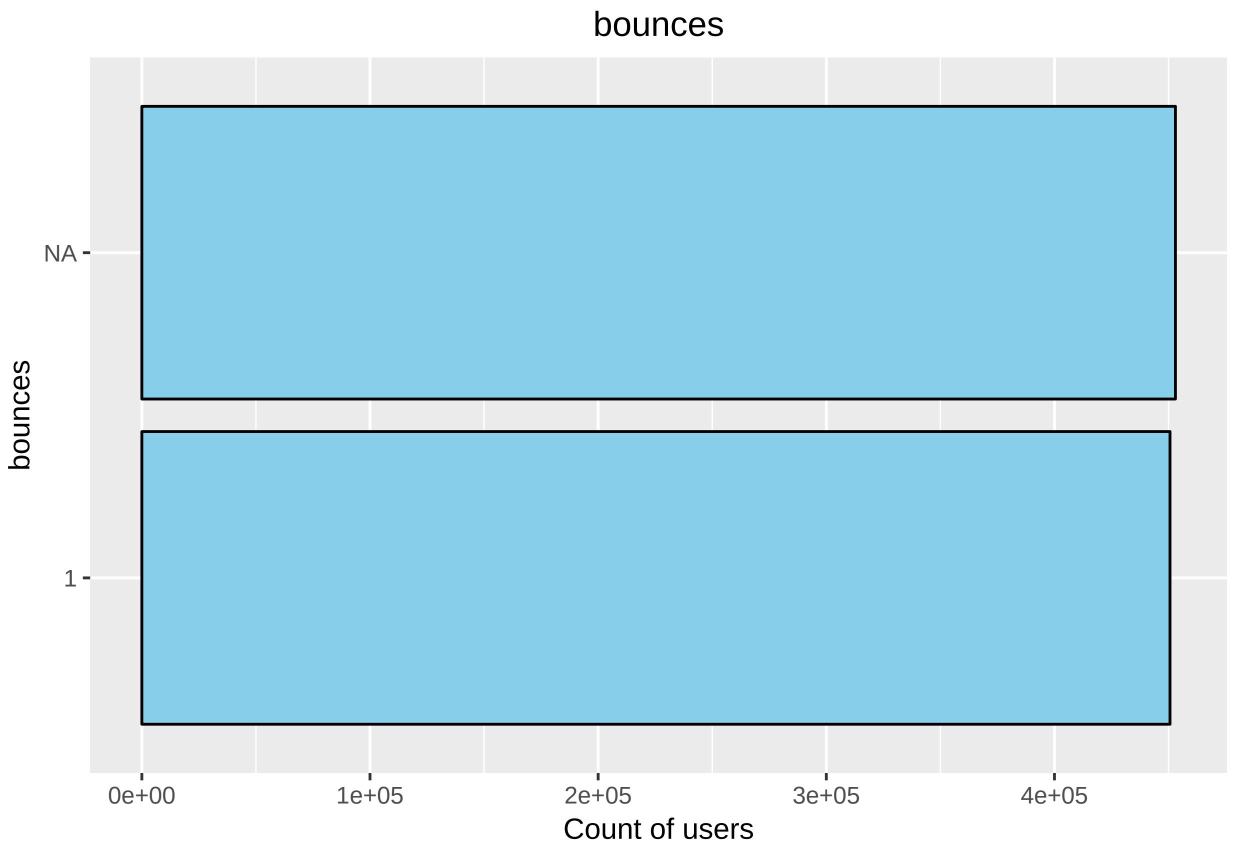
Bounces

A "bounce" is when the person comes to your website and then doesn't leave their initial landing page. They literally just come, look, and then leave. So bounces can be meaningful. See the below link:

https://support.google.com/analytics/answer/1009409?hl=en

According to the description there, a 1 means the person "bounced," and an NA means otherwise. So it's actually meaningful; recode so that a bounce is 0 and 1 means they stayed. Will call it did_not_bounce.

```
i <- ggplot(as.data.frame(plyr::count(train$bounces)), aes(x=reorder(x, freq), y=freq)) +
  geom_bar(stat="identity", color="black", fill="skyblue") +
  coord_flip() + xlab("bounces") +
  ylab("Count of users") + ggtitle("bounces") +
  theme(plot.title = element_text(hjust = 0.5))
i
```



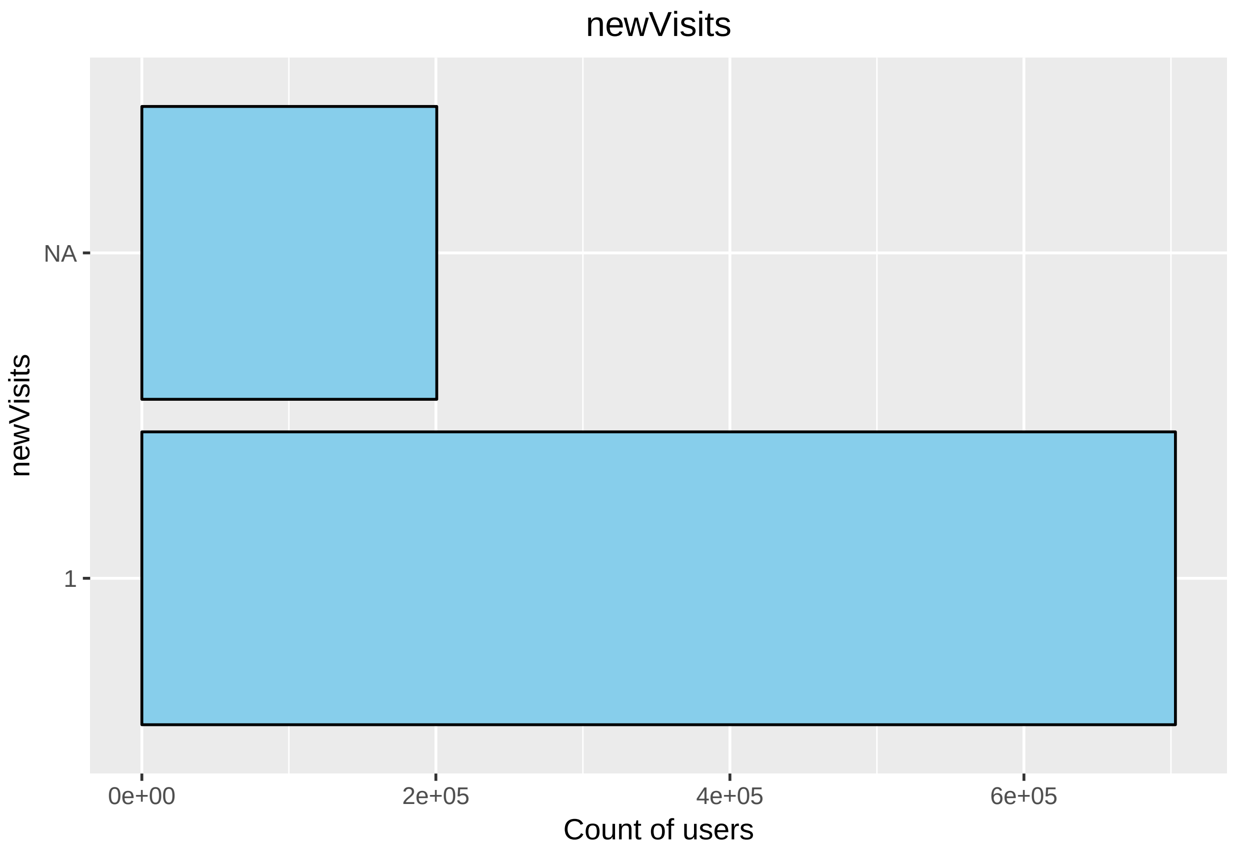
```
train$did_not_bounce <- ifelse(is.na(train$bounces), c(1), c(0))
train$bounces <- NULL
```

```
test$did_not_bounce <- ifelse(is.na(test$bounces), c(1), c(0))
test$bounces <- NULL
```

newVisits

Lastly - newVisits. According to the website, this is coded as a 1 if they are new and have never been here before, and it's NA if they have been there before. So I'm going to recode into a new variable and then do away with the original. 0 will mean they are new, 1 will mean they have been here before. The new variable measures whether they are a recurring visit.

```
i <- ggplot(as.data.frame(plyr::count(train$newVisits)), aes(x=reorder(x, freq), y=freq)) +
  geom_bar(stat="identity", color="black", fill="skyblue") +
  coord_flip() + xlab("newVisits") +
  ylab("Count of users") + ggtitle("newVisits") +
  theme(plot.title = element_text(hjust = 0.5))
i
```



```
train$recurring_visit <- ifelse(is.na(train$newVisits), c(1), c(0))
train$newVisits <- NULL
test$recurring_visit <- ifelse(is.na(test$newVisits), c(1), c(0))
test$newVisits <- NULL
```

Summary of the single-variable EDA.

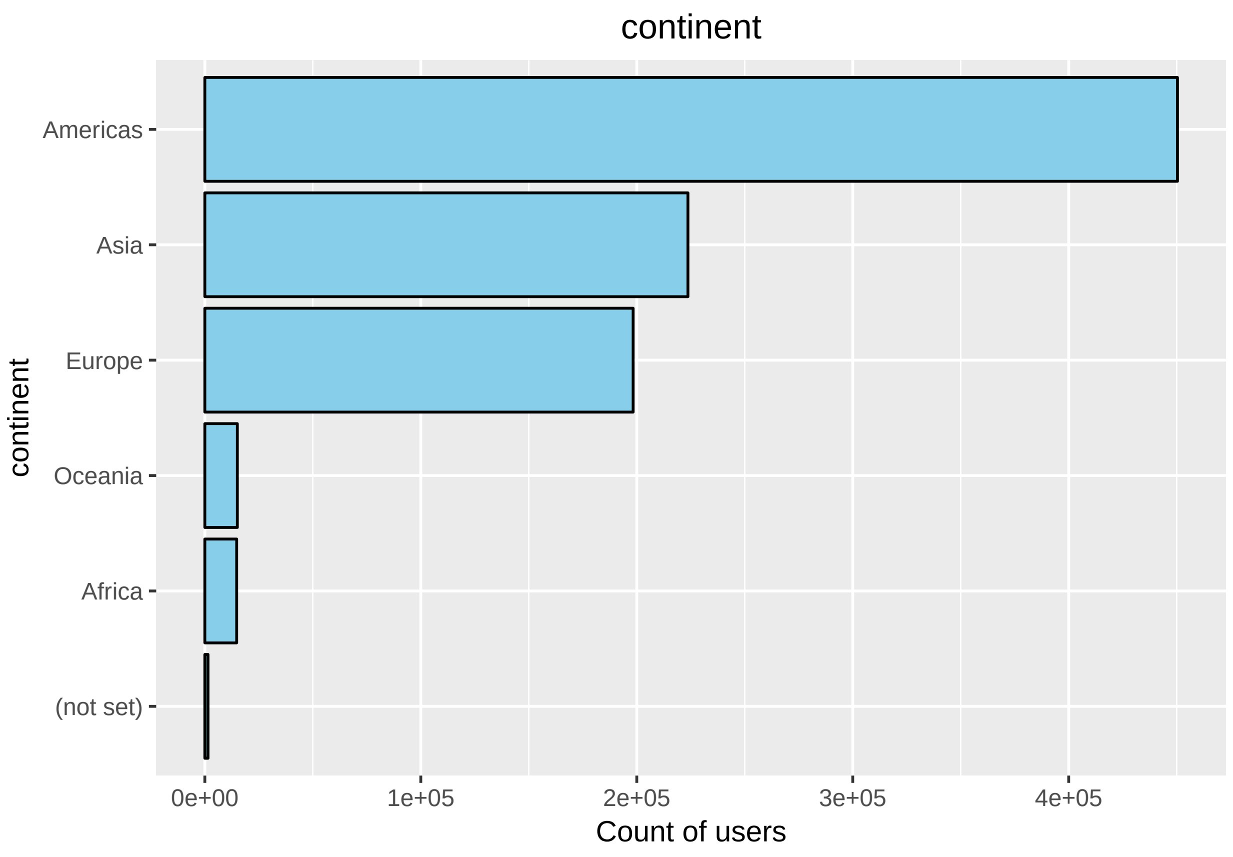
From here I'm going to just trim the dataset down into only the most relevant columns. I'm just going to drop some of the ones from above that were deemed somewhat meaningless.

```r
# Just dropping by column position.
train <- subset(train, select=-c(10,18,23,25,26,29,30,31,32,33))
```

Geographical Location

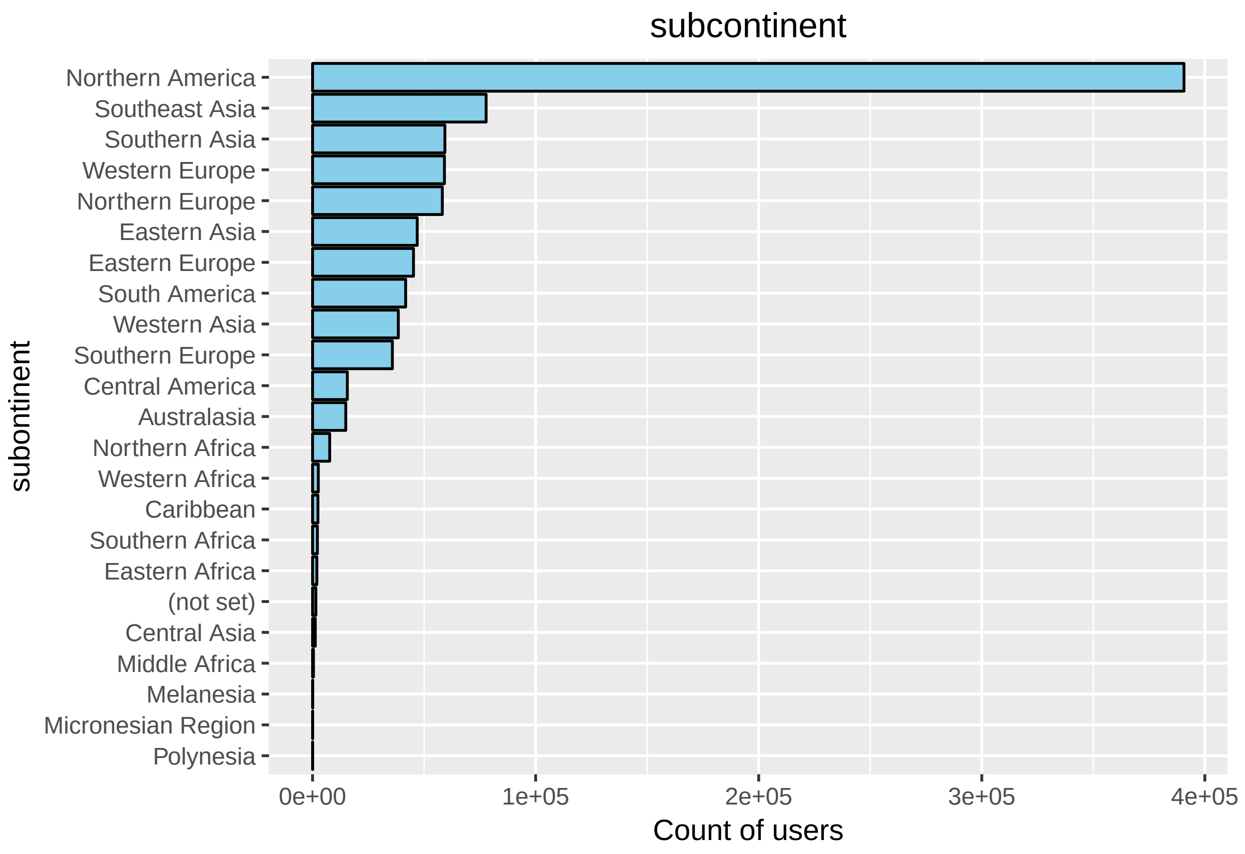
Let's just get a quick sense of where these people are at. It's worth noting that I already ran a handful of nested models, just to see if there is any meaningful clustering in the DV by country/continent/whatever. There is very much not so I don think there is any need to bother with nesting the data in that way.

```r
i <- ggplot(as.data.frame(plyr::count(train$continent)), aes(x=reorder(x, freq), y=freq)) +
  geom_bar(stat="identity", color="black", fill="skyblue") +
  coord_flip() + xlab("continent") +
  ylab("Count of users") + ggtitle("continent") +
  theme(plot.title = element_text(hjust = 0.5))
i
```



Most people are in the Americas, but where?

```r
i <- ggplot(as.data.frame(plyr::count(train$subContinent)), aes(x=reorder(x, freq), y=freq)) +
  geom_bar(stat="identity", color="black", fill="skyblue") +
  coord_flip() + xlab("subontinent") +
  ylab("Count of users") + ggtitle("subcontinent") +
  theme(plot.title = element_text(hjust = 0.5))
i
```



When you dig a bit deeper you see that most people live in the USA. For the sake of correlations, I'm going to just make some dummy coded variables for North America versus not and USA versus not.

```
train$north_am <- ifelse(train$subContinent == "Northern America", c(1), c(0))
train$usa <- ifelse(train$country == "United States", c(1), c(0))
test$north_am <- ifelse(test$subContinent == "Northern America", c(1), c(0))
test$usa <- ifelse(test$country == "United States", c(1), c(0))
```

Dates

The day and time that people visit is going to be important but those data needed to be parsed a bit. date is easy - it's just YYYYMMDD. So I'll just split those up. visitStartTime is a little trickier. It appears like "1472830365" for example. No idea. Use lubdridate?

Computing the DV

This is where there is a bunch of contention of Kaggle. Right from the competition, they say they want the log of the sum of the transactions for each user. There are some repeated users in the dataset so the data will need ot be grouped by user.

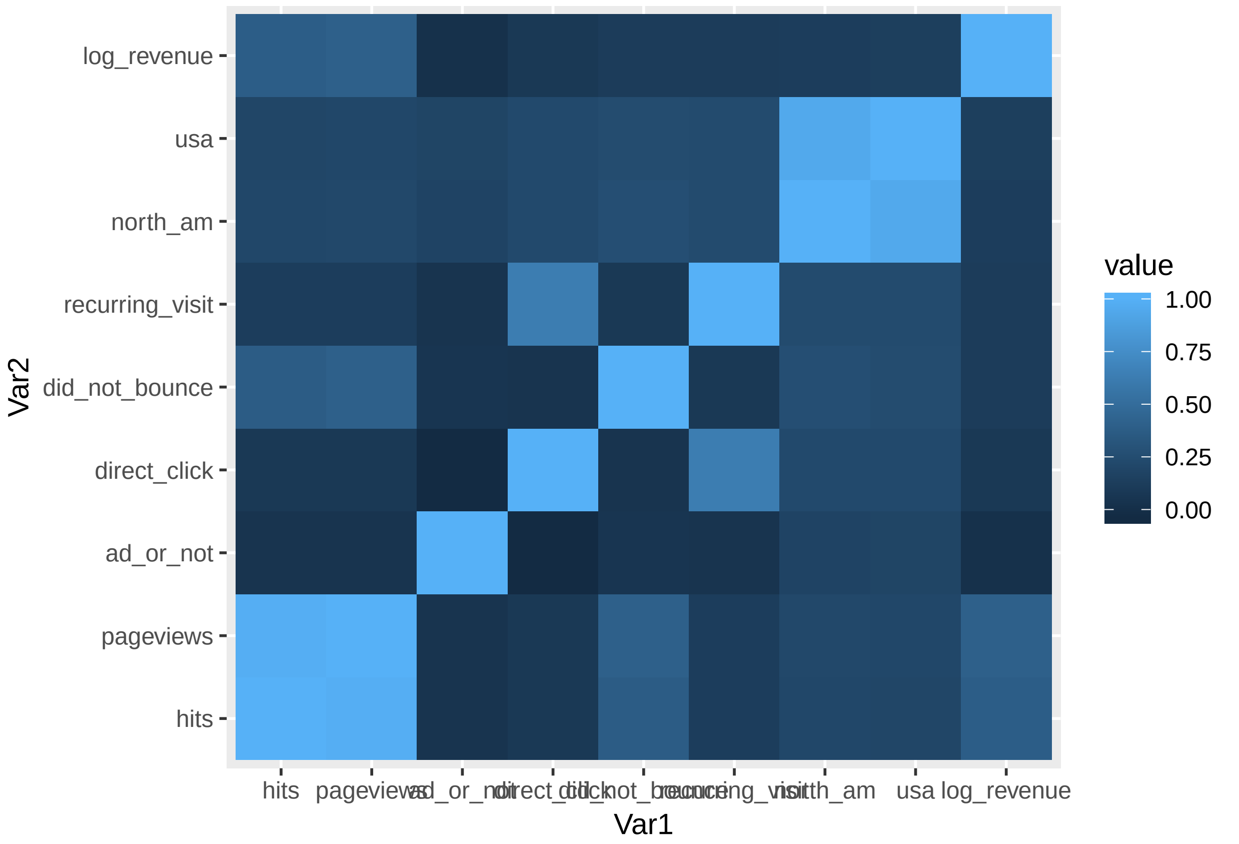
Some people are reporting they've been getting good results using the sum of the logs (instead of log of sums). I think sticking to the log of sums is probably the best idea.

It also brings in some contention - train the model on the visit level data, or visitor level data? I've tried both. It seems like when I train the model on the individual level data I get an RMSE of around 1.60, but then when I aggregate my predictions and recalc the RMSE it drops to about 2.00. This is something that remains to be solved. For now, I'm going to just leave it at the individual level.

Correlations with the numeric variables

Just going to make a quick correlation heatmap. See whats going on with all our numeric variables.

```r
# Take the log of the transactionRevenue
train$log_revenue <- log1p(train$transactionRevenue)
# Need to replace the missing values in pageviews - NAs need to be 0s
train$pageviews[is.na(train$pageviews)] <- 0
num_cols <- c("hits", "pageviews", "ad_or_not", "direct_click", "did_not_bounce", "recurring_visit", "no
cor_data <- train[,num_cols]
cormat <- round(cor(cor_data), 2)

library(reshape2)
melted_cormat <- melt(cormat)

library(ggplot2)
ggplot(data = melted_cormat, aes(x=Var1, y=Var2, fill=value)) +
  geom_tile()
```



So obviously north america and USA are highly correlated. They also each correlate pretty dismally with with the DV. Also pageviews and hits are almost perfectly correlated, and equally relate to the DV. So I think we could drop usa and either pageviews or hits.

It's worth noting that nearly every numeric dv poorly correlates with the DV. Which is weird because in some of the models I've run so far pageviews is by far the most important feature.

Let's toss everything into one huge regression to see what comes out

```
summary(lm(log_revenue~channelGrouping+visitNumber+deviceCategory+hits+pageviews+ad_or_not+direct_click
```

```
##
## Call:
## lm(formula = log_revenue ~ channelGrouping + visitNumber + deviceCategory +
##     hits + pageviews + ad_or_not + direct_click + did_not_bounce +
##     recurring_visit + usa + north_am, data = train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -65.828  -0.221  -0.002   0.147  20.961
##
## Coefficients:
##                                 Estimate Std. Error t value Pr(>|t|)
## (Intercept)                   -0.4477689  0.1654904  -2.706  0.00682 **
## channelGroupingAffiliates      0.1387671  0.1660499   0.836  0.40333
## channelGroupingDirect          0.1278984  0.1658175   0.771  0.44052
## channelGroupingDisplay         0.0872556  0.1669530   0.523  0.60123
## channelGroupingOrganic Search  0.1467542  0.1654650   0.887  0.37512
## channelGroupingPaid Search     0.1342137  0.1679410   0.799  0.42419
## channelGroupingReferral        0.4775185  0.1655222   2.885  0.00392 **
## channelGroupingSocial          0.2954275  0.1655094   1.785  0.07427 .
## visitNumber                   -0.0026060  0.0002132 -12.222  < 2e-16 ***
## deviceCategorymobile          -0.1047342  0.0047735 -21.941  < 2e-16 ***
## deviceCategorytablet          -0.1411824  0.0107160 -13.175  < 2e-16 ***
## hits                          -0.1067322  0.0010978 -97.228  < 2e-16 ***
## pageviews                      0.2634208  0.0015318 171.967  < 2e-16 ***
## ad_or_not                     -0.0631442  0.0318482  -1.983  0.04741 *
## direct_click                   0.1072809  0.0090798  11.815  < 2e-16 ***
## did_not_bounce                -0.3539313  0.0043231 -81.870  < 2e-16 ***
## recurring_visit                0.1829211  0.0077542  23.590  < 2e-16 ***
## usa                            0.2934895  0.0117116  25.060  < 2e-16 ***
## north_am                      -0.1034893  0.0116678  -8.870  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.811 on 903634 degrees of freedom
## Multiple R-squared:  0.1827, Adjusted R-squared:  0.1827
## F-statistic: 1.122e+04 on 18 and 903634 DF,  p-value: < 2.2e-16
```

So based on this, so far, I think we need to dummy code channelGrouping, just keep the referral column. We can maybe drop ad_or_not, butt everything else predicts nicely! Onward in feature selection. Let's try that really quickly:

```
train$ref <- ifelse(train$channelGrouping == "Referral", c(1), c(0))
test$ref <- ifelse(test$channelGrouping == "Referral", c(1), c(0))

summary(lm(log_revenue~ref+visitNumber+deviceCategory+hits+pageviews+ad_or_not+direct_click+did_not_bou
```

```
##
## Call:
```

```
## lm(formula = log_revenue ~ ref + visitNumber + deviceCategory +
##     hits + pageviews + ad_or_not + direct_click + did_not_bounce +
##     recurring_visit + usa + north_am, data = train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -65.515  -0.218   0.045   0.118  20.905
##
## Coefficients:
##                       Estimate Std. Error t value Pr(>|t|)
## (Intercept)          -0.224887   0.003318 -67.785  < 2e-16 ***
## ref                   0.300976   0.006490  46.372  < 2e-16 ***
## visitNumber          -0.002597   0.000213 -12.193  < 2e-16 ***
## deviceCategorymobile -0.131414   0.004671 -28.137  < 2e-16 ***
## deviceCategorytablet -0.166651   0.010672 -15.615  < 2e-16 ***
## hits                 -0.106201   0.001098 -96.714  < 2e-16 ***
## pageviews             0.262128   0.001532 171.124  < 2e-16 ***
## ad_or_not            -0.096047   0.012916  -7.436 1.04e-13 ***
## direct_click          0.058061   0.005410  10.732  < 2e-16 ***
## did_not_bounce       -0.361192   0.004318 -83.656  < 2e-16 ***
## recurring_visit       0.197621   0.006164  32.059  < 2e-16 ***
## usa                   0.296853   0.011711  25.349  < 2e-16 ***
## north_am             -0.138471   0.011612 -11.925  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.812 on 903640 degrees of freedom
## Multiple R-squared:  0.1819, Adjusted R-squared:  0.1819
## F-statistic: 1.674e+04 on 12 and 903640 DF,  p-value: < 2.2e-16
```

Okay. Not all the *bs* are huge, but we're getting somewhere. I think this is a really good starting point in terms of features. Just need to ad in date and time, and I think we can get cracking!

First Pass Model

Before I get into integrating date and time as predictors, I'm just going to show you all what I mean regarding my model performance not doing well at the aggregate-by-visitor level. First, I will train my model on the visit-level data, then aggregate the data by visitor, and recalculate the RMSE. I'm going to make the model not be verbose, just to not clutter the markdown doc.

In doing this, I'm going to calculate the dv as the sum of logs. I'm going to take the log revenue at the visit level, train the model, aggregate, and re run. XgBoost linear Regression.

```r
library(xgboost)
library(mltools)

cols <- c("ref","visitNumber","deviceCategory","hits","pageviews","ad_or_not","direct_click","did_not_bo
features <- train[cols]

dtrain <- xgb.DMatrix(data = data.matrix(features[,]),
                      label = train$log_revenue)

watchlist <- list(train = dtrain)
```

```r
param <- list(  objective = "reg:linear",
                booster = "gbtree",
                eta = 0.01,
                max_depth = 8,
                subsample = 0.8,
                colsample_bytree = 0.8,
                min_child_weight = 25
)

clf <- xgb.train(  params = param,
                   data = dtrain,
                   nrounds = 500,
                   verbose = FALSE,
                   watchlist = watchlist,
                   maximize = FALSE
)

# Looking at my training predictions
train_preds <- predict(clf, dtrain)
train_preds <- as.data.frame(train_preds)
train_preds$actuals <- train$log_revenue
train_preds$fullVisitorId <- train$fullVisitorId
p <- train_preds$train_pred
actuals <- train_preds$actuals
rmse(pred=p, actual=actuals, weights=1, na.rm=TRUE)
```

```
## [1] 1.652362
```

So, we see a really good RMSE - 1.65. We are quite accurate. But what happens when we take it up to the visitor level, aggregate by fullVisitorId? So, the sum of logs? Just taking that matrtix I made with predictions and actuals, aggregating it, and recalcing the RMSE

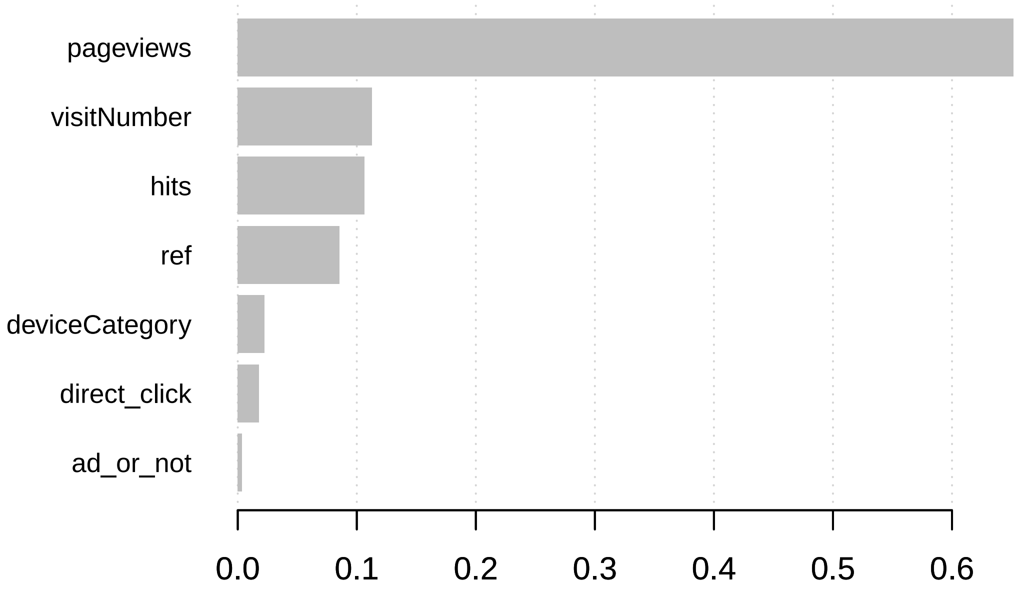
```r
library(sqldf)
```

```
## Loading required package: gsubfn

## Loading required package: proto

## Warning in doTryCatch(return(expr), name, parentenv, handler): unable to load shared object '/Librar
##   dlopen(/Library/Frameworks/R.framework/Resources/modules//R_X11.so, 6): Library not loaded: /opt/X
##   Referenced from: /Library/Frameworks/R.framework/Resources/modules//R_X11.so
##   Reason: image not found

## Could not load tcltk.  Will use slower R code instead.

## Loading required package: RSQLite
```

```r
grouped_train_preds <- sqldf("SELECT
                             SUM(train_preds) train_preds,
                             SUM(actuals) actuals,
                             fullVisitorId
                             FROM train_preds GROUP BY fullVisitorId;")

g_p <- grouped_train_preds$train_pred
g_actuals <- grouped_train_preds$actuals
rmse(pred=g_p, actual=g_actuals, weights=1, na.rm=TRUE)
```

```
## [1] 2.112521
```

```r
# Plot out the importance - what is carrying the weight in the model?
imp <- xgb.importance(feature_names = clf$feature_names, model = clf)
xgb.plot.importance(imp, top_n = 25)
```



So, we see when we aggregate the RMSE is dropping to shit - 2.11. Why does the model not generalize well like this? In an obvious next step, let me try training the model on already grouped data.

Also, looking at the graph above regarding what features are important, notice how pageviews is extremely important? Not reflected in the fact that it basically does not correlate with the DV at all. . . .

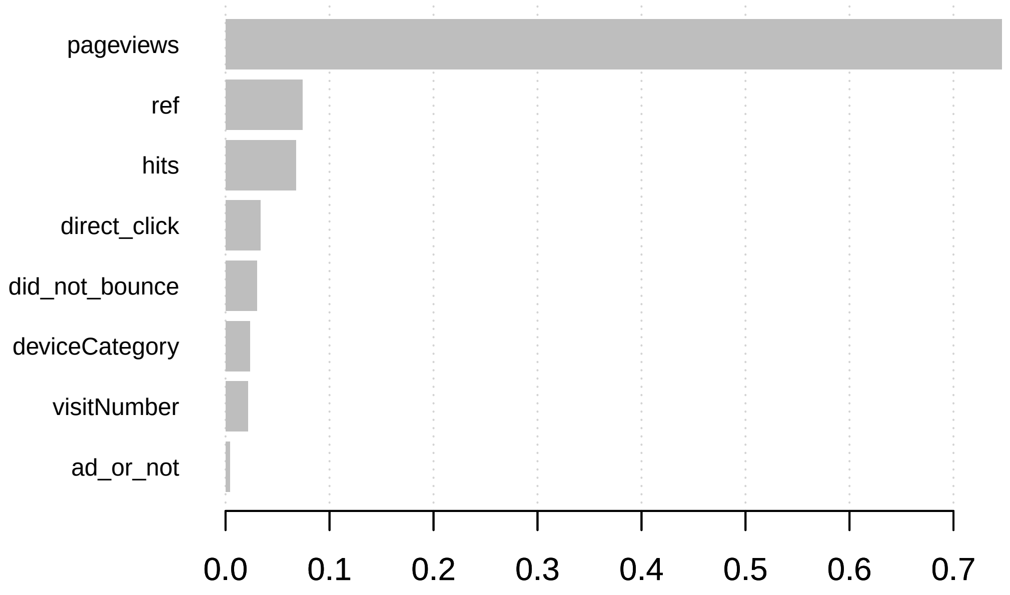
In grouping the data first, I can take the proper dv. I'm going to do log of sums this time.

```r
# Aggregate the data by visitor id
grouped_data <- sqldf("SELECT
                            ref, visitNumber, deviceCategory, SUM(hits) hits,
                            SUM(pageviews) pageviews, SUM(ad_or_not) ad_or_not,
                            SUM(direct_click) direct_click, SUM(did_not_bounce) did_not_bounce,
                            SUM(recurring_visit) recurring_visit, usa, north_am, fullVisitorId,
                            SUM(transactionRevenue) transactionRevenue
                      FROM train GROUP BY fullVisitorId;")

# calculate the DV
grouped_data$log_revenue <- log1p(grouped_data$transactionRevenue)

features <- grouped_data[cols]

dtrain <- xgb.DMatrix(data = data.matrix(features[,]),
                      label = grouped_data$log_revenue)

clf <- xgb.train(  params = param,
                   data = dtrain,
                   nrounds = 500,
                   verbose = FALSE,
                   watchlist = watchlist,
                   maximize = FALSE
)
```

```
# Looking at my training predictions
train_preds <- predict(clf, dtrain)
train_preds <- as.data.frame(train_preds)
train_preds$actuals <- grouped_data$log_revenue
train_preds$fullVisitorId <- grouped_data$fullVisitorId
p <- train_preds$train_pred
actuals <- train_preds$actuals
rmse(pred=p, actual=actuals, weights=1, na.rm=TRUE)
```

```
## [1] 1.639755
```

```
# Plot out the importance - what is carrying the weight in the model?
imp <- xgb.importance(feature_names = clf$feature_names, model = clf)
xgb.plot.importance(imp, top_n = 25)
```



Alright, when I run this model right on the grouped data it actually performs pretty well - 1.63. Just for shits I'm going to run this on the test and then submit it to Kaggle. See what we get.

```
grouped_test <- sqldf("SELECT
                        ref, visitNumber, deviceCategory, SUM(hits) hits,
                        SUM(pageviews) pageviews, SUM(ad_or_not) ad_or_not,
                        SUM(direct_click) direct_click, SUM(did_not_bounce) did_not_bounce,
                        SUM(recurring_visit) recurring_visit, usa, north_am, fullVisitorId
                  FROM test GROUP BY fullVisitorId;")

test2 <- grouped_test[,cols]

pred <- predict(clf, data.matrix(test2))
submission <- as.data.frame(pred)
submission$fullVisitorId <- grouped_test$fullVisitorId
names(submission)[names(submission) == 'pred2'] <- 'PredictedLogRevenue'
submission <- submission[,c(2,1)]

write.csv(submission, "XgBoost_sum_first.csv", row.names = FALSE)
```

Okay, file is written. Off to Kaggle. . . . and. . . .. 1.8027!

Actually not terrible - moved me 25 whole places up the leaderboard. Best model yet, currently sitting at 425/555.

I think this is a good starting point but far more feature engineering is needed. Most specifically, I think what the missing key is is the date and time. Just haven't had time yet to deal with those data. Also what needs addressing is the fact there is a huge discrepancy between CV and LB - my training RMSE was 1.63, so it dropped about .17 from my private score to the public leaderboard. this did not happen when I was running basic regression models.

Onward!