```r
#Set WD
rm(list = ls()) #to clear global environment
setwd("C:/Users/bryan/Desktop/Winter 2019 Quarter/Marketing Analytics in R/Directory")
library(ggplot2)
library(data.table)

## Warning: package 'data.table' was built under R version 3.5.2

library(lfe)

## Warning: package 'lfe' was built under R version 3.5.2

## Loading required package: Matrix

library(stargazer)

##
## Please cite as:

##  Hlavac, Marek (2018). stargazer: Well-Formatted Regression and Summary Statistics Tables.

##  R package version 5.2.2. https://CRAN.R-project.org/package=stargazer

library(rmarkdown)

trains = fread("train_subset.csv")
is.data.table(trains)

## [1] TRUE

trainsrand = subset(trains, random_bool == 1) #This is the random dataset
trainsnorm = subset(trains, random_bool == 0) #This is the ranking generated by Expedias algorithm.
uniqueN(trains[, .(srch_id, prop_id)] ) / trains[,.N]

## [1] 1

#Set Key
#Question 1

setkey(trains,srch_id, prop_id)
setkey(trainsnorm,srch_id, prop_id)
setkey(trainsrand,srch_id, prop_id)

## The key We found is the combination of srch_id, which is searching ID, and prop_id, which is the hotel ID.
## Each observations represent a consequent click on the search results of accommodations appearing on Expedia's websites. The srch_id is a index that records the search, and the search ID might occur more than once with
```

```
different hotel ID since each search can come with multiple matched
accommodations and visitors might click more than one of them.

trainsnorm[, prop_starrating := as.numeric(prop_starrating) ]
lm1 = lm(position~log(price_usd + 1) + prop_starrating, data=trainsnorm)

#Site_ID
lm2 = lm(position~log(price_usd + 1) + prop_starrating + site_id,
data=trainsnorm)
#visitor_location_country_id

lm3 = lm(position~log(price_usd + 1) + prop_starrating + site_id +
visitor_location_country_id, data=trainsnorm)

#visitor_hist_starrating
trainsnorm[, visitor_hist_starrating := as.numeric(visitor_hist_starrating) ]
lm4 = lm(position~log(price_usd + 1) + prop_starrating + site_id +
visitor_location_country_id + visitor_hist_starrating, data=trainsnorm)

#visitor_hist_adr_usd
trainsnorm[, visitor_hist_adr_usd := as.numeric(visitor_hist_adr_usd) ]
lm5 = lm(position~log(price_usd + 1) + prop_starrating + site_id +
visitor_location_country_id + visitor_hist_starrating + visitor_hist_adr_usd,
data=trainsnorm)

#Remove site_id and visitor location id.
lm6 = lm(position~log(price_usd + 1) + prop_starrating +
visitor_hist_starrating + visitor_hist_adr_usd, data=trainsnorm)

stargazer(lm1, lm2, lm3, lm4, lm5, lm6,
          title="Position", type="text",
          column.labels=c( "Random", "Not-Random"),
          df=FALSE, digits=2, star.cutoffs = c(0.05,0.01,0.001))
```

Going forward, we will not include site_id, visitor_location_id, or visitor_hist_adr_usd due to the insignificant coefficients and conceptually.

```
#Prop_country_Id
lm7 = lm(position~log(price_usd + 1) + prop_starrating +
visitor_hist_starrating + prop_country_id, data=trainsnorm)

#Prop Review Score
trainsnorm[, prop_review_score := as.numeric(prop_review_score) ]
lm8 = lm(position~log(price_usd + 1) + prop_starrating +
visitor_hist_starrating + prop_country_id + prop_review_score,
data=trainsnorm)
#Prop Brand Bool
lm9 = lm(position~log(price_usd + 1) + prop_starrating +
visitor_hist_starrating + prop_review_score + prop_brand_bool,
data=trainsnorm) #Best Model so far.
```

```
#Prop Location Score 1
lm10 = lm(position~log(price_usd + 1) + prop_starrating +
visitor_hist_starrating + prop_review_score + prop_brand_bool +
prop_location_score1, data=trainsnorm)
#Prop Location Score 2
trainsnorm[, prop_location_score2 := as.numeric(prop_location_score2) ]
lm11 = lm(position~log(price_usd + 1) + prop_starrating +
visitor_hist_starrating + prop_review_score + prop_brand_bool +
prop_location_score1 + prop_location_score2, data=trainsnorm)
#promotion_flag
lm12 = lm(position~log(price_usd + 1) + prop_starrating +
visitor_hist_starrating + prop_review_score + prop_brand_bool +
prop_location_score1 + prop_location_score2 + promotion_flag,
data=trainsnorm)

#srch_destination_id, INSIGNIFICANT
lm13 = lm(position~log(price_usd + 1) + prop_starrating +
visitor_hist_starrating + prop_review_score + prop_brand_bool +
prop_location_score1 + prop_location_score2 + promotion_flag +
srch_destination_id, data=trainsnorm) #srch_length_of_stay
lm14 = lm(position~log(price_usd + 1) + prop_starrating +
visitor_hist_starrating + prop_review_score + prop_brand_bool +
prop_location_score1 + prop_location_score2 + promotion_flag +
srch_length_of_stay, data=trainsnorm) #srch_booking_window, INSIGNIFICANT
lm15 = lm(position~log(price_usd + 1) + prop_starrating +
visitor_hist_starrating + prop_review_score + prop_brand_bool +
prop_location_score1 + prop_location_score2 + promotion_flag +
srch_length_of_stay + srch_booking_window, data=trainsnorm)
#srch_adults_count
lm16 = lm(position~log(price_usd + 1) + prop_starrating +
visitor_hist_starrating + prop_review_score + prop_brand_bool +
prop_location_score1 + prop_location_score2 + promotion_flag +
srch_length_of_stay + srch_adults_count, data=trainsnorm)

stargazer(lm7, lm8, lm9, lm10,  lm11, lm12, lm13, lm14, lm15, lm16,
          title="Position", type="text",
          column.labels=c( "Random", "Not-Random"),
          df=FALSE, digits=2, star.cutoffs = c(0.05,0.01,0.001))
```

When adding variables to this regression, we decided to exclude prop_country_id, prop_brand_bool, search_destination_id, and search_booking_window.

```
#srch_children_count
lm17 = lm(position~log(price_usd + 1) + prop_starrating +
visitor_hist_starrating + prop_review_score + prop_location_score1 +
prop_location_score2 + promotion_flag + srch_length_of_stay +
srch_adults_count + srch_children_count, data=trainsnorm)

#srch_room_count
lm18 = lm(position~log(price_usd + 1) + prop_starrating +
```

```
visitor_hist_starrating + prop_review_score + prop_location_score1 +
prop_location_score2 + promotion_flag + srch_length_of_stay +
srch_adults_count + srch_children_count + srch_room_count, data=trainsnorm)

#srch_saturday_night_bool
lm19 = lm(position~log(price_usd + 1) + prop_starrating +
visitor_hist_starrating + prop_review_score + prop_location_score1 +
prop_location_score2 + promotion_flag + srch_length_of_stay +
srch_adults_count + srch_children_count + srch_room_count +
srch_saturday_night_bool, data=trainsnorm) #Insignificant. Will not Add.

#srch_query_affinity_score
trainsnorm[, srch_query_affinity_score :=
as.numeric(srch_query_affinity_score) ]
lm20 = lm(position~log(price_usd + 1) + prop_starrating +
visitor_hist_starrating + prop_review_score + prop_location_score1 +
prop_location_score2 + promotion_flag + srch_length_of_stay +
srch_adults_count + srch_children_count + srch_room_count +
srch_query_affinity_score, data=trainsnorm) #Insignificant. Will not Add.
Plus it has low observations without null values.

#orig_destination_distance
trainsnorm[, orig_destination_distance :=
as.numeric(orig_destination_distance) ]
lm21 = lm(position~log(price_usd + 1) + prop_starrating +
visitor_hist_starrating + prop_review_score + prop_location_score1 +
prop_location_score2 + promotion_flag + srch_length_of_stay +
srch_adults_count + srch_children_count + srch_room_count +
orig_destination_distance, data=trainsnorm) #Insignificant. Will not add.

finalmodel = lm(position~log(price_usd + 1) + prop_starrating +
visitor_hist_starrating + prop_review_score + prop_location_score1 +
prop_location_score2 + promotion_flag + srch_length_of_stay +
srch_adults_count + srch_children_count + srch_room_count, data=trainsnorm)
#Same as lm18.

stargazer(lm17, finalmodel, lm19, lm20, lm21,
          title="Position", type="text",
          column.labels=c( "lm17", "Final Model"),
          df=FALSE, digits=2, star.cutoffs = c(0.05,0.01,0.001))

finalmodel = lm(position~log(price_usd + 1) + prop_starrating +
visitor_hist_starrating + prop_review_score + prop_location_score1 +
prop_location_score2 + promotion_flag + srch_length_of_stay +
srch_adults_count + srch_children_count + srch_room_count, data=trainsnorm)

#Need to conver all to numeric for the randomly selected dataset since I
subsetted the data beforehand.
trainsrand[, prop_starrating := as.numeric(prop_starrating) ]
```

```
trainsrand[, visitor_hist_starrating := as.numeric(visitor_hist_starrating) ]
trainsrand[, visitor_hist_adr_usd := as.numeric(visitor_hist_adr_usd) ]
trainsrand[, prop_review_score := as.numeric(prop_review_score) ]
trainsrand[, prop_location_score2 := as.numeric(prop_location_score2) ]
trainsrand[, orig_destination_distance :=
as.numeric(orig_destination_distance) ]


finalmodelrandom = lm(position~log(price_usd + 1) + prop_starrating +
visitor_hist_starrating + prop_review_score + prop_location_score1 +
prop_location_score2 + promotion_flag + srch_length_of_stay +
srch_adults_count + srch_children_count + srch_room_count, data=trainsrand)

stargazer(finalmodel, finalmodelrandom,
          title="Position", type="text",
          column.labels=c( "Final Model Expedia", "Final Model Randomly
Selected"),
          df=FALSE, digits=2, star.cutoffs = c(0.05,0.01,0.001))

trainsnorm[, rPosition := ceiling(10*runif(.N))] #Expedia ranked
trainsrand[, rPosition := ceiling(10*runif(.N))] #Randomly generated

finalmodelz = lm(rPosition~log(price_usd + 1) + prop_starrating +
visitor_hist_starrating + prop_review_score + prop_location_score1 +
prop_location_score2 + promotion_flag + srch_length_of_stay +
srch_adults_count + srch_children_count + srch_room_count, data=trainsnorm)

finalmodelrandomz = lm(rPosition~log(price_usd + 1) + prop_starrating +
visitor_hist_starrating + prop_review_score + prop_location_score1 +
prop_location_score2 + promotion_flag + srch_length_of_stay +
srch_adults_count + srch_children_count + srch_room_count, data=trainsrand)

stargazer(finalmodel, finalmodelrandom, finalmodelz, finalmodelrandomz,
          title="Position vs rPosition", type="text",
          column.labels=c( "Pos Expedia", "Pos Random", "rPos Expedia", "rPos
Random"),
          df=FALSE, digits=2, star.cutoffs = c(0.05,0.01,0.001))
```

For the interpretation, we see that once we randomize the position ourselves (rPosition), our model becomes totally insignificant and so if Expedia truely randomized their data, the randomized subset will be all insignificant. When we use this model on the original randomized position, we see that there are some variables that are insignificant. This details that Expedia did not properly randomize their samples as well as they could have.

```
randompos1 = subset(trainsrand, position == 1)
randompos10 = subset(trainsrand, position == 10) #Ask him about clickthrough
rate

#Clickthrough Rate
clickthroughpos1 = mean(randompos1$click_bool) #.143
clickthroughpos10 = mean(randompos10$click_bool)
```

```
clickthroughanswer= clickthroughpos1 - clickthroughpos10
#There is a 9.5 percent increase for clickthrough rate in position 1 versus postion 10.

#Booking Rate
bookingpos1 = mean(randompos1$booking_bool) #.01906
bookingpos10 = mean(randompos10$booking_bool)
bookinganswer = bookingpos1 - bookingpos10
#There is a 1.3% increase for bookings in position 1 versus position 10.
```

Question 3b. Yes, it is more beneficial to be at the top of the randomized list because we can still observe a positive increase in both click through and booking rate.

```
table <- data.frame("position"=NULL, "a"=NULL, "b"=NULL, "e"=NULL, "c"=NULL,"d"=NULL, "f"=NULL)
for (x in 1:10){
   table = rbind(table,data.frame("position" = x,"a" = mean(trains[position == x & random_bool == 0, click_bool]),"b" = mean(trains[position == x & random_bool == 1, click_bool]),"e" = (mean(trains[position == x & random_bool == 0, click_bool])-mean(trains[position == x & random_bool == 1, click_bool])),"c" = mean(trains[position == x & random_bool == 0, booking_bool]),"d" = mean(trains[position == x & random_bool == 1, booking_bool]), "f" = (mean(trains[position == x & random_bool == 0, booking_bool])-mean(trains[position == x & random_bool == 1, booking_bool]))))
}

names(table) = c("position", "click rate: algorithm","click rate: random"," difference of click rate", "booking rate: algorithm", "booking rate: random", "difference of booking rate")

#4A. From positions 1 to 4, the clickthrough rate of algorithms are significantly higher than the randomized dataset. From position 6 - 10, the clickthrough rate doesn't change too much and so we conclude that the algorithm is beneficial for the higher positions compared to lower positions.

#4B. We observe similar effects for positions 1 to 4 where we see a huge difference in booking rate between algorithm and randomized. For positions 6-10, we see there is still positive difference but not as much as the top positions.

#5. From our analysis, top ranking positions results in higher clickthrough and booking rate. But since Expedia's "randomized" dataset was not properly randomized, we cannot conclude the causal estimates of their algorithm since they don't have a proper control group. We assume that the Expedia algorithm is attempting to maximize bookings, but since the randomization is incorrect, the effectiveness of the algorithm is ambigous.
```