An Analysis of Airbnb Reviews

Dylan Loader

December 8th, 2017

Abstract

Since 2008 Airbnb has been a disruptive force in the short-term rental market. Currently operating in 191 countries and with over three million concurrent listings, the platform allows renters to book accommodations remotely from anywhere on Earth. With this increased exposure comes both increased risks and rewards. For a host to become successful they must maintain a high user rating which is based on their guests' experiences. In this paper I attempt to establish a model to estimate the listing attributes which contribute to guests leaving higher reviews. The data is obtained directly from Airbnb.com using Python 3, containing 12,715 listings from Toronto, Canada. From the data a ggmap is created using the locational information provide from the listings. Variable selection is performed using cross validation. The review scores are converted to the following categories, 1-3.5 stars ("Unsatisfied") and 4-5 stars ("Satisfied"). The most appropriate logistic regression fit, included the following predictors: total number of host listings, price and number of reviews. This allows us to deduce that price and longevity are important in determining if an Airbnb host is successful. Hence hosts should aim to provide the listings at a reasonable price. Moreover, the significant covariates related to longevity are less likely casual and more likely a property of successful listings. These results suggest that Airbnb host ratings are more likely affected by qualitative aspects of user experience than quantitative. Renters using Airbnb should attempt to find low cost, established renters to maximize their experiences.

Contents

1 Introduction

With approximately 48% of the World's population (3.648 Billion people) with access to the internet worldwide as of 2017 (Development, 2017), virtual commerce is at an unprecedented level. Airbnb.com is an online marketplace which allows hosts to rent their properties on a short-term basis. The hosts may rent entire homes, private room, or shared rooms, with experiences being recently added to the Airbnb site. The site excels in the short-term rental space, primarily by offering lower nightly rates than traditional establishments, more accessible bookings and unique experiences from a lower renter to host ratio. Airbnb offers rentals from as low as $10 USD per night, in over 191 countries (https://www.airbnb.ca/about/about-us, 2017). In most markets this is far below the average rate. Due the unique market advantage of Airbnb many consider it a disruptive institution (Guttentag, 2015). With this disruption comes an increased level of criticism from communities and the government (Horton, 2015). The lower level of verification for renters, often leads to conflict between hosts and their surrounding neighbors. In some cases, this has lead to the termination of rentals, fines and extensive damage to the rental property. The increased oversight from the government has lead to pressure for stronger regulations on short-term rentals. For current and perspective hosts, the issue of remaining profitable is essential. Thus hosts should attempt to maximize their rating on the site to ensure continuing business. In this paper I attempt to establish a model to estimate the listing attributes which contribute to guests leaving higher reviews, and thus producing more profitable outcomes for hosts.

2 Methods

2.1 Data Description

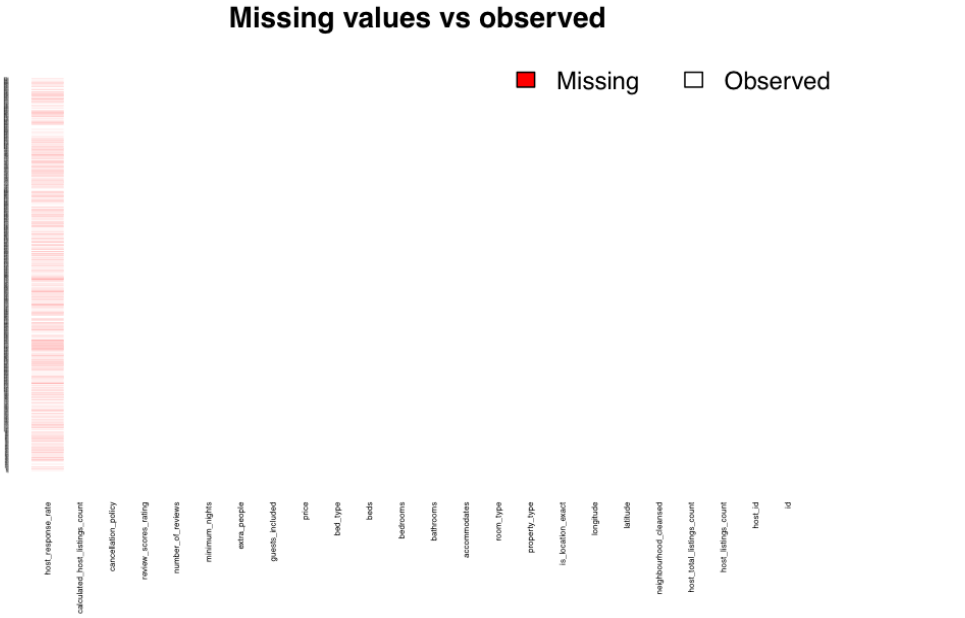
The data are scraped from Airbnb.com listings in Toronto, Canada on June 3rd, 2017. The data initially includes 12,714 observations. The scraping method utilized by Slee is implemented in Python 3, which uses a JSON (JavaScript Object Notation) module to collect and organize the publicly available information from Airbnb.com into a .csv. The data contains 24 variables, with several variables which are redundant or merely represent the identification tag used by the scraping code, which while providing interesting context do not provide interpretable metrics within the study. The scraped data quantifies a host's rating as continuous between 20-100, representing 1-5 stars averaged across all host ratings. To

prepare for logistic regression techniques this ratio scale data is converted to ordinal data with some information loss.

2.2 Visual Exploration of Data

2.2.1 Missingness Plot (Amelia II)

```
missmap(air.omit, main = "Missing values vs observed",
        col = c("red","white"),
        y.cex = 0.1,
        x.cex = 0.35)
```

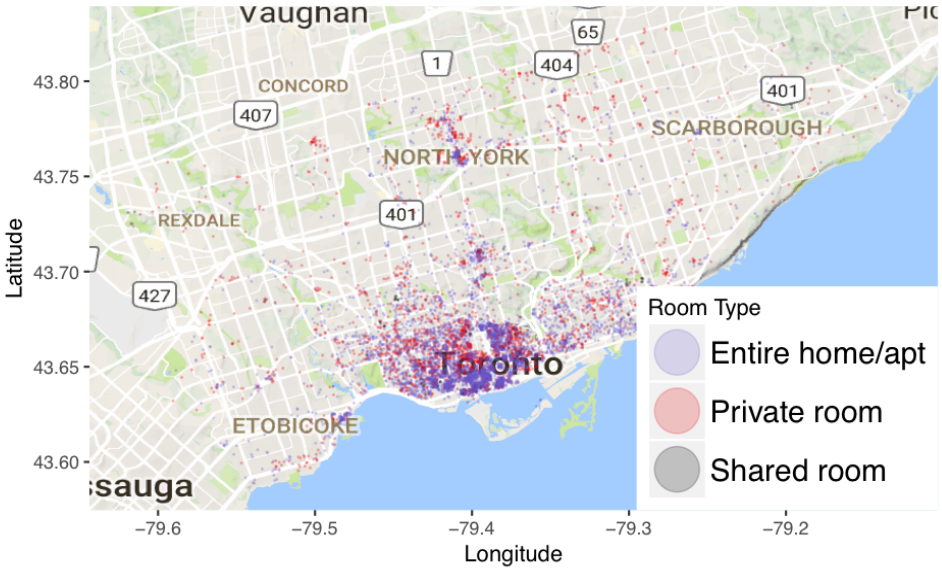


Using the Amelia II package we can see some degree of the missingness of our data. In this figure the x-axis is the variables ordered left to right from most missing to least missing. The y axis expresses the observation number. From the above figure we can see that host response rating is heavily missing and should most likely be removed.

2.2.2 GGplot Mapping

```
# Overlay the longitude and latitude values onto the google map
mapPoints <- ggmap(map.1, extent  = "normal") +
```

```
   geom_point(aes(x = location.df$longitude, y = location.df$latitude,
                  colour = location.df$room_type),
                  data = location.df, alpha = 0.2, size = 0.1) +
                  scale_color_manual(values=c("slateblue","red2", "black"),
                                     name="Room Type") +
                  theme(legend.justification=c(1,0),
                        legend.position=c(1,0),
                        legend.text=element_text(size=15),
                        legend.key.size = unit(0.5, "cm")) +
                  coord_fixed(xlim = c(min(location.df$longitude),
                                       max(location.df$longitude)),
                              ylim = c(min(location.df$latitude),
                                       max(location.df$latitude)),
                              ratio = 1.2)  +
                  labs(x = "Longitude",
                  y = "Latitude") +
                  guides(colour = guide_legend(override.aes = list(size=10)))
# Return the ggmap object
mapPoints
```



Using the ggmap package, the following plot is created by accessing a current Google maps image, and overlaying the data points using their longitude and latitude. This plot only considers the observations in which the location data is reported as exact by Airbnb. From the plot we can see the majority of reviews are for Entire homes and apartment and Private rooms. Shared rooms account for a small, almost non-existent proportion of reviews.

2.3 Data Cleaning

The number of bathrooms is rounded using the ceiling function in R to reduce the number of factors needed in later modeling. Price is treated as continuous, although it is discrete as the range of prices was $4486 CAD. Pricing was limited to $13 to $1600 to eliminate cases where the pricing reflected on the site was incorrectly entered by the host as monthly. The value of $1600 was chosen as at the time of writing the most expensive nightly rate in Toronto was under $1600. The function OutlierKD was run on some variables but seems to remove too many valid observations. The remaining variables have some observations removed for missingness, leaving 9741. Since the Python 3 scraping script has a consistent format, several scrapes can be merged to increase sample size. As there is no guarantee this has not occurred, the data are removed if their unique room ID number is repeated. To prepare the data for modeling, ID, host ID, host response rate, host listing count, latitude, longitude and calculated host listing counts are removed.

2.4 Modeling

The response of review score for the logistic model is ordinal fitting a cumulative logistic. First by categorizing the review scores, and then by running the saturated cumulative logistic model.

```
# So we have a left skewed distribution for stars.
# Change the review scores into categorical
 working.df$categorical.rating <- cut(working.df$` reviews.score`,
                        breaks = c(-Inf, 40, 60, 80, 100, Inf),
                        labels = c("1-1.5 stars",
                                   "2-2.5 stars",
                                   "3-3.5 stars",
                                   "4-4.5 stars",
                                   "5 stars"),
                        right = FALSE)
# https://rpubs.com/kaz_yos/VGAM
# Begin by not assuming parallel
 cumulative.df <- working.df
 model.1 <- vglm(categorical.rating ~ .,
                 family=cumulative(parallel=FALSE),
                 data = cumulative.df)
 summary(model.1)
 is.parallel(model.1)
# Since we see strong evidence of parallelism run model 2
 model.2 <- vglm(categorical.rating ~ .,
                 family=cumulative(parallel=TRUE),
                 data = cumulative.df)
 summary(model.2)
```

```
# Polr will not work for our data
model.3 <- polr(formula = categorical.rating ~ .,
                data = cumulative.df,
                Hess = TRUE)

# Remove some variables based on low significance as AIC is not reported
 cumulative.df <- cumulative.df[, -c(4,5,8,9)]

 sapply(cumulative.df, typeof)
 model.4 <- vglm(categorical.rating ~ .,
                 family=cumulative(parallel=TRUE),
                 data = cumulative.df)
 summary(model.4)
 lrtest_vglm(model.2,model.4)
```

This code is not active as it causes errors when run in R. The polr model and non-parallel vglm model do not run due to perfect separation in the review score response(Allison, 2008). The parallel vglm model does produce estimate but they will be overly biased due to non-convergent MLE.

```
multi.model <- nnet::multinom(formula = overall_satisfaction ~ . ,
                              MaxNWts = 2000,
                              maxit = 350,
                              data = multinom.df)
model.summary.1 <- summary(multi.model)
model.summary.1
```

Likewise running a nominal multinomial saturated model with the nnet package does converge, but produces inflated coefficients. To attempt to rectify the perfect separation in the response, methods of variable selection are run to remove the unknown combination of variables causing separation.

2.5 Variable Selection

```
#Subset regression is too slow
regsubsets.out <-
    regsubsets(working.df$` reviews.score` ~.,
               data = working.df,
               nbest = 1,       # 1 best model for each number of predictors
               nvmax = NULL,    # NULL for no limit on number of variables
               force.in = NULL, force.out = NULL,
               method = "exhaustive")
fit.1 <- lm(working.df$` reviews.score` ~., data = working.df)
```

```r
summary(fit.1)
car::vif(fit.1)
#a gvif^(1/(2*df)) is near 2 so we may want to remove variables.
```

Initially exhaustive subset selection is run on a linear model. Although the number of variables is computable, the exhaustive selection fails to complete most likely due to the prefect separation. An other method to remove perfect separation is to remove multicollinear variables (UCLA, 2016). Running VIF on the saturated model, shows few variables with GVIF > 2 and removing them does not rectify the perfect separation. Since perfect separation is caused by non-convergence of maximum likelihood, another approach is to use cross-validation to remove variables based on Mean Squared Error.

2.6 Cross-Validation

```r
####################################################
# Variable selection using glmnet
####################################################
training.rows <- sample(1:dim(glm.df)[1], 1*dim(glm.df)[1])
y.training <- glm.df$categorical.rating[training.rows]
y.testing <- glm.df$categorical.rating[-training.rows]

glm.df <- glm.df[, -(13)]
x.training <- as.matrix(glm.df[training.rows, ])
x.testing <- as.matrix(glm.df[-training.rows, ])
# Check the cross validation
cvfit = cv.glmnet(x.training, y.training, family = "binomial")
```

```r
# Get coefficients from the cross validation using lambda 1se
coef(cvfit, s = "lambda.1se")
```

```
## 13 x 1 sparse Matrix of class "dgCMatrix"
##                                1
## (Intercept)           3.384868
## total.listings        .
## room.type             .
## bathrooms             .
## bedrooms              .
## beds                  .
## bed.type              .
## price                 .
## persons.allowed       .
## extra.persons.allowed .
## minimum.stay          .
## number.of.reviews     .
```

```
## cancellation.policy    .
```

```
## lambda.1se is a better choice in general for handling over regularization but returne
coef(cvfit, s = "lambda.min")
```

```
## 13 x 1 sparse Matrix of class "dgCMatrix"
##                                1
## (Intercept)            3.8554542369
## total.listings        -0.0159162044
## room.type             -0.3773248829
## bathrooms              0.0059379633
## bedrooms               .
## beds                   .
## bed.type               .
## price                 -0.0003818999
## persons.allowed        0.0306869667
## extra.persons.allowed  .
## minimum.stay           0.0041377132
## number.of.reviews      0.0016265934
## cancellation.policy    .
```

The cross-validation performed is 10-fold. The sample size of the data frame is pseudo-randomly partitioned into 10 sub-samples of equal size. 9 of the sub-samples are used as training data for a binomial logistic regression, which is then used to predict the 10th excluded sample. This process is repeated 10 times, which assures each observation is used to validate only once (OpenML, 2017). With the cvfit, lambda.1se is "the largest [lambda] at which the MSE is within one standard error of the minimal MSE." (Hastie and Qian, 2014), where as lambda.min is " the [lambda] at which the minimal MSE is achieved." (Hastie and Qian, 2014). Lambda.1se is a better choice of tuning parameter as it chooses a simpler model. In this case, choosing the lambda.1se suggests no covariates, so we choose lambda.min to retain some variables.

The cross-validation suggests the best model uses total listings, price, number or reviews, room type, persons allowed and minimum stay, to estimate binary review score.

2.7 Final Model Selection

From the cross-validation a binomial logistic regression is fit of the form:

$$logit\left(\frac{\hat{\pi}}{1-\hat{\pi}}\right) = \beta_0 + \beta_1 total.listings + \beta_2 room.type + \beta_3 price + \beta_4 persons.allowed + \beta_5 number.of.reviews$$

```
summary(binary.model)
```

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

```
## glm(formula = categorical.rating ~ total.listings + room.type +
##     price + persons.allowed + number.of.reviews, family = "binomial",
##     data = binary.df)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -3.7531   0.1571   0.2449   0.3019   1.3150
##
## Coefficients:
##                            Estimate Std. Error z value Pr(>|z|)
## (Intercept)                2.768621   0.394294   7.022 2.19e-12 ***
## total.listings            -0.007079   0.001761  -4.021 5.81e-05 ***
## room.typeEntire home/apt   0.069769   0.408045   0.171  0.86424
## room.typePrivate room     -0.363198   0.396269  -0.917  0.35938
## price                      0.003166   0.001117   2.834  0.00459 **
## persons.allowed           -0.082882   0.065567  -1.264  0.20620
## number.of.reviews          0.048271   0.006268   7.700 1.36e-14 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 2808.2  on 9733  degrees of freedom
## Residual deviance: 2631.7  on 9727  degrees of freedom
## AIC: 2645.7
##
## Number of Fisher Scoring iterations: 8
```

From the model we can see the room type and number of persons allowed seem to not be significant. Since the cross validation is only performed once, these may be significant by chance. Under this assumption a reduced model is fit removing variables with low significance of the form:

```
summary(binary.model.2)
```

```
##
## Call:
## glm(formula = categorical.rating ~ total.listings + price + number.of.reviews,
##     family = "binomial", data = binary.df)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -3.8221   0.1563   0.2509   0.3141   1.3663
##
## Coefficients:
##                     Estimate Std. Error z value Pr(>|z|)
```

```
## (Intercept)        2.3931048  0.1152945  20.756  < 2e-16 ***
## total.listings    -0.0070075  0.0017654  -3.969 7.21e-05 ***
## price              0.0044508  0.0009555   4.658 3.19e-06 ***
## number.of.reviews  0.0486562  0.0062326   7.807 5.87e-15 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 2808.2  on 9733  degrees of freedom
## Residual deviance: 2642.4  on 9730  degrees of freedom
## AIC: 2650.4
##
## Number of Fisher Scoring iterations: 8
```

$$logit\left(\frac{\hat{\pi}}{1-\hat{\pi}}\right) = 2.3931 - 0.0070 * total.listings + 0.0045 * price + 0.0487 * number.of.reviews$$

```
exp(coefficients(binary.model))
```

```
##              (Intercept)           total.listings room.typeEntire home/apt
##                15.9366459                0.9929463                1.0722604
##    room.typePrivate room                    price          persons.allowed
##                 0.6954484                1.0031711                0.9204593
##        number.of.reviews
##                 1.0494545
```

```
exp(coefficients(binary.model.2))
```

```
##       (Intercept)    total.listings             price number.of.reviews
##         10.947431          0.993017          1.004461          1.049859
```

From the exponentiation of betas the impact on log odds of satisfaction in reviews seems small in both cases, as they are close to 1 for all covariates. For this reason a likelihood ratio test is run to see if reducing the model is adequate. From the large sample size using a significance level of $\alpha = 0.05$ is adequate.

$$H_0 : logit\left(\frac{\hat{\pi}}{1-\hat{\pi}}\right) = \beta_0 + \beta_1 total.listings + \beta_2 room.type + \beta_3 price + \beta_4 persons.allowed + \beta_5 number.of.reviews$$

$$H_1 : logit\left(\frac{\hat{\pi}}{1-\hat{\pi}}\right) = \beta_0 + \beta_1 total.listings + \beta_2 price + \beta_3 number.of.reviews$$

```
lrtest(binary.model, binary.model.2)
```

```
## Likelihood ratio test
##
## Model 1: categorical.rating ~ total.listings + room.type + price + persons.allowed +
##     number.of.reviews
## Model 2: categorical.rating ~ total.listings + price + number.of.reviews
##   #Df  LogLik Df  Chisq Pr(>Chisq)
## 1   7 -1315.8
## 2   4 -1321.2 -3 10.691    0.01352 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Since p-value $= .01352 < \alpha = 0.05$ reject H_0 and the data do suggest the reduced model is adequate. Since the most adequate model has been found, a goodness of fit likelihood ratio is run.

```
lrtest(binary.model.2)
```

```
## Likelihood ratio test
##
## Model 1: categorical.rating ~ total.listings + price + number.of.reviews
## Model 2: categorical.rating ~ 1
##   #Df  LogLik Df  Chisq Pr(>Chisq)
## 1   4 -1321.2
## 2   1 -1404.1 -3 165.87  < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

H_0 : The model is not an adequate fit.

H_1 : The model is an adequate fit.

$\alpha = 0.05$

H_0 is rejected at the $\alpha = 0.05$ significance level, and the model is found to have an adequate fit. To get a rough idea of the predictive power of the model McFadden's pseudo-R^2 is calculated.

```
pR2(binary.model.2)
```

```
##           llh       llhNull            G2      McFadden          r2ML
## -1.321182e+03 -1.404117e+03  1.658713e+02  5.906606e-02  1.689604e-02
##          r2CU
##  6.741932e-02
```

The McFadden's pseudo-R^2 is approximately 0.06. When interpreted roughly as a linear regression R^2, the model explains around 6% of the variation in the data.

3 Results

```
##       (Intercept)    total.listings             price number.of.reviews
##         10.947431          0.993017          1.004461          1.049859
```

The data do suggest that a host's total number of listings decreases the estimated odds for a renter feeling satisfied by a multiplicative factor of about 0.993, per increase of one listing, holding other covariates constant. Where as the price and number of reviews increase the estimated odds by a multiplicative factor of approximately 1.004 and 1.050 respectively. These increases are holding other covariates as fixed within the range of our same, with price being spending one additional Canadian dollar and number of reviews being one more review left by renters.

4 Discussion

4.1 Model Significance

Using the logistic model:

$$logit\Big(\frac{\hat{\pi}}{1-\hat{\pi}}\Big) = 2.3931 - 0.0070 * total.listings + 0.0045 * price + 0.0487 * number.of.reviews$$

I find that the price a renter pays and number of reviews a host currently has do increase the likelyhood that a renter reviews their stay as satisfactory. This is agrees logicallly, as the more a renter pays, the more likely their accomodation will be habitable and comfortable. Similarily the more reviews a host has is indicative of their longevity on the site which usually correlates to hosts whom take their duties as short-term leasers more seriously. The model suggests that the total number of listings a hosts has correlates to a lower likelihood of a renter being satisfied with their accomodation. This result is slightly harder to explain, but I accredit this to hosts with several listings being less attentive to their individual guests' needs. The overall pseudo-predictive power of the model is low, which seems to be related to two factors. Firstly the data collected is limited to Airbnb's front-facing site. Most data driven companies, such as Airbnb retain their predictive data to cement their market position. Secondly the data are from an online voluntary survey which falls victim to survey methodology flaws.

4.2 Survey Methodology

The topic of surveys and specifically online surveys is complex, for this report I examine only two sub-categories of survey bias:

4.2.1 Content Control Bias

Since Airbnb is a private company, with no legal requirement to disclose their review data, they have control over what content is publically available. Looking at their content policy (https://www.airbnb.ca/help/article/546/what-is-airbnb-s-content-policy) we can see several possibilities for Airbnb to positively skew responses. When a host leaves a review, Airbnb has final say over whether they wish to post the review. In cases where the renter is dissatisfied with their stay, Airbnb is financially incentivized to supress negative responses to retain public perception of positive experiences with Airbnb. I suggest a few sample cases where Airbnb may censor negative responses. If a reviewer suggests staying at a hotel instead of Airbnb, the review can be deleted for advertising another commercial company. Frustrated renters may also be more likely to use vulgar language to express their frustrations, which again is grounds for removal. Reviewers are also restricted from referring to details of Airbnb investigations. In this case even if a dispute is solved unfavorably for the renter, they cannot post a negative review of their stay.

4.2.2 Extreme Response Bias

In their paper Muchnik, Aral and Taylor(2013) find that people who have their persona with their reviews are more likely to leave extreme responses such as 1 or 5 stars due to their preconcieved notions of judgement against them for their views. Herding bias can also contribute to the response bias. The prevailing opinion on a topic, in this case a host's rating effects individual behaviors(Muchnik, Aral, Taylor, 2013). In this case guests are less likely to leave a negative review of a host which currently has a positive review. This also occurs when a host has overwhelmingly negative reviews, and a renter wishes to leave a positive review. This may be a topic of futher exploration, although the number of negative reviews in the data are too low to conclude something substantial.

5 Conclusion

The model fit suggests significance of some variables in the data, but seems to have a low predictive power. I suggest that a more complete data set of collected may provide a better insight into what makes a host successful in gaining a renter's satisfaction. Unfortunately Airbnb does not provide this data willingly. To rectify the weaknesses of the study we should consider further data collection to account for covariates not offered by Airbnb, and possibly performing a qualitative analysis of the text of posted reviews on Airbnb.

6 References

Data Source:

-http://insideairbnb.com/get-the-data.html

Python Scraping Code:

-https://github.com/tomslee/airbnb-data-collection/

Data Visualization:

-https://gking.harvard.edu/amelia

Perfect Separation:

- P. Allison, Convergence Failures in Logistic Regression, SAS Global Forum 2008
- https://stats.idre.ucla.edu/other/mult-pkg/faq/general/faqwhat-is-complete-or-quasi-complete-separa (Introduction to SAS. UCLA: Statistical Consulting Group. from https://stats.idre.ucla.edu/sas/modules/sas-learning-moduleintroduction-to-the-features-of-sas/ (accessed August 22, 2016).)

Glmnet: - https://web.stanford.edu/~hastie/glmnet/glmnet_alpha.html

Cross-Validation: - https://www.openml.org/a/estimation-procedures/1

Survey Bias: - https://www.airbnb.ca/help/article/546/what-is-airbnb-s-content-policy - Lev Muchnik, Sinan Aral, Sean J. Taylor,Social Influence Bias: A Randomized Experiment, July 2013

```
knitr::opts_chunk$set(cache=TRUE)
# Clear the variable environment, to ensure dataframes are created properly
rm(list=ls())

#Install the required packages


options(repos="http://cran.rstudio.com/")
set.seed("820")
# Install amelia for the missing map function
if(!require("Amelia")){install.packages("Amelia")}

# Run a multiple linear regression to find possible multicollinearity.
# A simple MLR to examine first.
if(!require("car")){install.packages("car")}
if(!require("lme4")){install.packages("lme4")}
if(!require("VGAM")){install.packages("VGAM")}
if(!require("ResourceSelection")){install.packages("ResourceSelection")}
if(!require("regsubsets")){install.packages("regsubsets")}
```

```
## Loading required package: regsubsets
```

```
## Warning in library(package, lib.loc = lib.loc, character.only = TRUE,
## logical.return = TRUE, : there is no package called 'regsubsets'
```

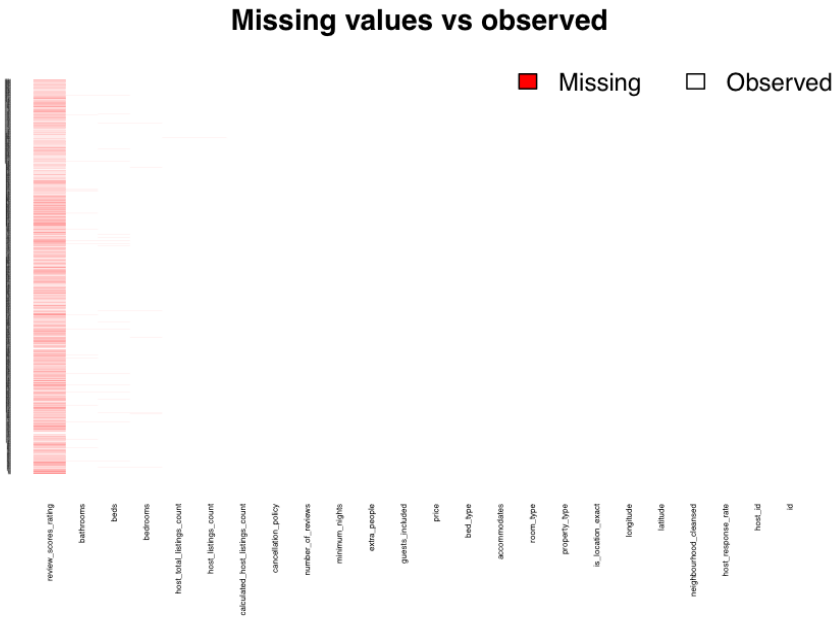
```
## Warning: package 'regsubsets' is not available (for R version 3.4.2)
```

```
if(!require("MASS")){install.packages("MASS")}
if(!require("bestglm")){install.packages("bestglm")}

# For Likelihood ratio test
if(!require("lmtest")){install.packages("lmtest")}
# For Hosmer Lemeshow goodness of fit test
if(!require("MKmisc")){install.packages("MKmisc")}

# Glmnet packages
if(!require("glmnet")){install.packages("glmnet")}
if(!require("doParallel")){install.packages("doParallel")}
if(!require("pROC")){install.packages("pROC")}
pkgs <- list("glmnet", "doParallel", "foreach", "pROC")
lapply(pkgs, require, character.only = T)
```

```
## [[1]]
## [1] TRUE
##
## [[2]]
## [1] TRUE
##
## [[3]]
## [1] TRUE
##
## [[4]]
## [1] TRUE
```

```
registerDoParallel(cores = 6)



# Scraped Data
#http://tomslee.net/airbnb-data-collection-get-the-data
# Read the data file using choose file
air.omit <- read.csv(file = "listings-rich.csv",
                     sep=",",
                     stringsAsFactors = F,
                     header = T)

##############################################################################
# Data Cleaning
##############################################################################
# An initial look at missing observations
# https://www.r-bloggers.com/how-to-perform-a-logistic-regression-in-r/
```

```
# Suggests that there are not many missing values.



missmap(air.omit, main = "Missing values vs observed",
        col = c("red","white"),
        y.cex = 0.1,
        x.cex = 0.35)
```




```
#sapply(air.omit, typeof)
# Convert strings to numeric for the required variables
air.omit$host_response_rate <- as.double(as.character(air.omit$host_response_rate))
```

```
## Warning: NAs introduced by coercion
```

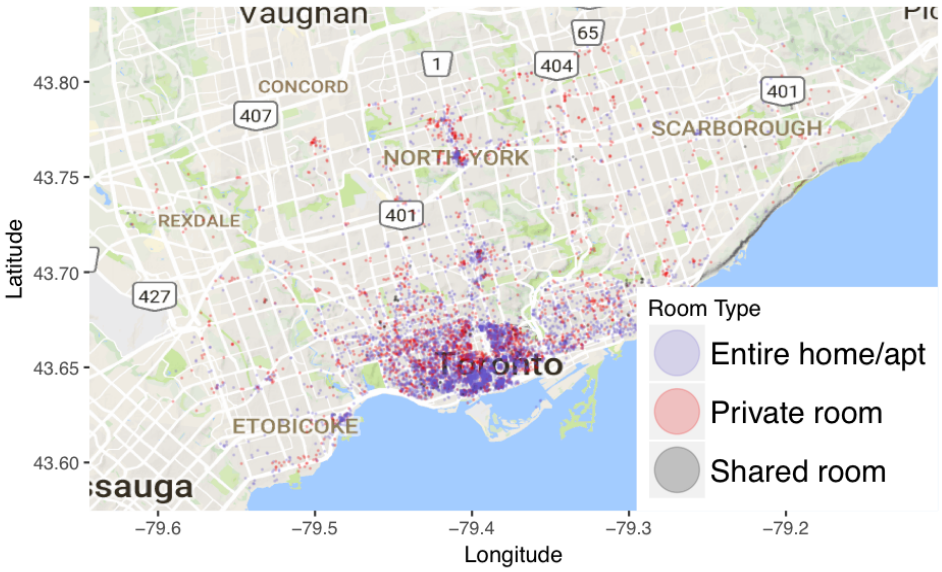
```
air.omit$bathrooms <- as.integer(ceiling(as.double(as.character(air.omit$bathrooms))))
air.omit$price <- as.double(as.character(air.omit$price))
# Change the rounding digits for full precision in longitude and latitude
options(digits=9)
air.omit$latitude <- as.double(as.character(air.omit$latitude))
air.omit$longitude <- as.double(as.character(air.omit$longitude))

# Remove entries with no reviews (new listings)
air.omit <- air.omit[(air.omit$number_of_reviews) != 0,]
```

```r
# Remove some entries with missing values
air.omit <- air.omit[(air.omit$review_scores_rating) != 0| is.na(air.omit$review_scores_

# Remove missing accomodations, bathrooms, bedrooms, and number of beds
air.omit <- air.omit[!(is.na(air.omit$accommodates)),]
air.omit <- air.omit[!(is.na(air.omit$bathrooms)),]
air.omit <- air.omit[!(is.na(air.omit$bedrooms)),]
air.omit <- air.omit[!(is.na(air.omit$beds)),]
air.omit <- air.omit[!(is.na(air.omit$review_scores_rating)),]
air.omit <- air.omit[!(is.na(air.omit$host_total_listings_count)),]

# Remove duplicated room id's cause by merging multiple scrapes, keeping the most rece
air.omit <- air.omit[!duplicated(air.omit$id, fromLast = T), ]



##############################################################################
# Geological Mapping
##############################################################################
# Install ggmap to map the data points
# Source: http://www.milanor.net/blog/maps-in-r-plotting-data-points-on-a-map/

if(!require("ggmap")){install.packages("ggmap")}
if(!require("mapproj")){install.packages("mapproj")}

# Create a data frame containing only user and room ids, and the location data for eac
location.starter.df <- air.omit

# Subset the location data frame for only the location variables
location.df <- subset(location.starter.df, select = c(longitude, latitude, is_location_e

# Convert room_type to factor for color
location.df$room_type <- factor(location.df$room_type)

# Remove non exact locations (no more ocean dwellers)
location.df <- location.df[!(location.df$is_location_exact == "f"), ]

# Get the map from google maps, cuts out some observations at zoom 10, but grants bett
map.1 <- get_map(location = "Toronto", source = "google", zoom = 10)
```

```
## Map from URL : http://maps.googleapis.com/maps/api/staticmap?center=Toronto&zoom=10&s
```

```
## Information from URL : http://maps.googleapis.com/maps/api/geocode/json?address=Toron
```

```r
# Overlay the longitude and latitude values onto the google map
mapPoints <- ggmap(map.1, extent  = "normal") +
   geom_point(aes(x = location.df$longitude, y = location.df$latitude,
                  colour = location.df$room_type),
                  data = location.df, alpha = 0.2, size = 0.1) +
                  scale_color_manual(values=c("slateblue","red2", "black"), name="Room T
                  theme(legend.justification=c(1,0), legend.position=c(1,0),
                        legend.text=element_text(size=15),
                        legend.key.size = unit(0.5, "cm")) +
                  coord_fixed(xlim = c(min(location.df$longitude), max(location.df$longi
                              ylim = c(min(location.df$latitude), max(location.df$latitu
                  labs(x = "Longitude",
                  y = "Latitude") +
                  guides(colour = guide_legend(override.aes = list(size=10)))

# Use ggsave to convert location mapping to a pdf for output
#ggsave(file="map.pdf", width=8, height=8)

# Return the ggmap object
mapPoints
```




```r
# Prepare data for modelling
# Remove, id, host_id, host_response_rate, host_listings_count,
# latitude, longitude, review_scores_rating
working.df <- air.omit[,-c(1:4, 7:9,24)]
```

```r
# Rename the variables for easier access.
name.vector <- c("total.listings", "neigbourhood", "property.type",
                 "room.type", "accomodates", "bathrooms", "bedrooms",
                 "beds", "bed.type", "price", "persons.allowed",
                 "extra.persons.allowed", "minimum.stay", "number.of.reviews",
                 " reviews.score", "cancellation.policy")

# Set the column names of the working data frame
colnames(working.df) <- name.vector

# Create a rating based on star review
categorize <- function(x, lower = 0, upper, by = 10,
                   sep = "-", above.char = "+") {
                   labs <- c(paste(seq(lower, upper - by, by = by),
                   seq(lower + by - 1, upper - 1, by = by),
                   sep = sep),
                   paste(upper, above.char, sep = ""))
                   cut(floor(x), breaks = c(seq(lower, upper, by = by), Inf),
                   right = FALSE, labels = labs)
}

# Check Categorization
table(categorize(working.df$` reviews.score`,lower = 0, upper = 100, by = 79))
```

```
##
## 0-78 100+
##  306 9435
```

```r
# Convert review scores to categorical
working.df$categorical.rating <- ifelse(working.df$` reviews.score` >= 80, 1, 0)


# http://www4.stat.ncsu.edu/~post/josh/LASSO_Ridge_Elastic_Net_-_Examples.html
# Backup the working data frame
working.restore.df <- working.df

working.df <- working.restore.df[,-c(2,3,5,15,24)]




####################################################
# Variable selection using glmnet
####################################################
```

```r
# Set a binary data frame
binary.df <- working.df

# Set the initial glm dataframe
glm.df <- binary.df

binary.df <- binary.df[(binary.df$price <= 1600), ]
glm.df <- binary.df


# convert all data into factor and then numeric for use in glmnet
for(i in 1:dim(glm.df)[2]){
  glm.df[,i] <- as.integer(as.factor(as.character(glm.df[,i])))
}

# Check to see if the conversion ran properly
sapply(glm.df, typeof)
```

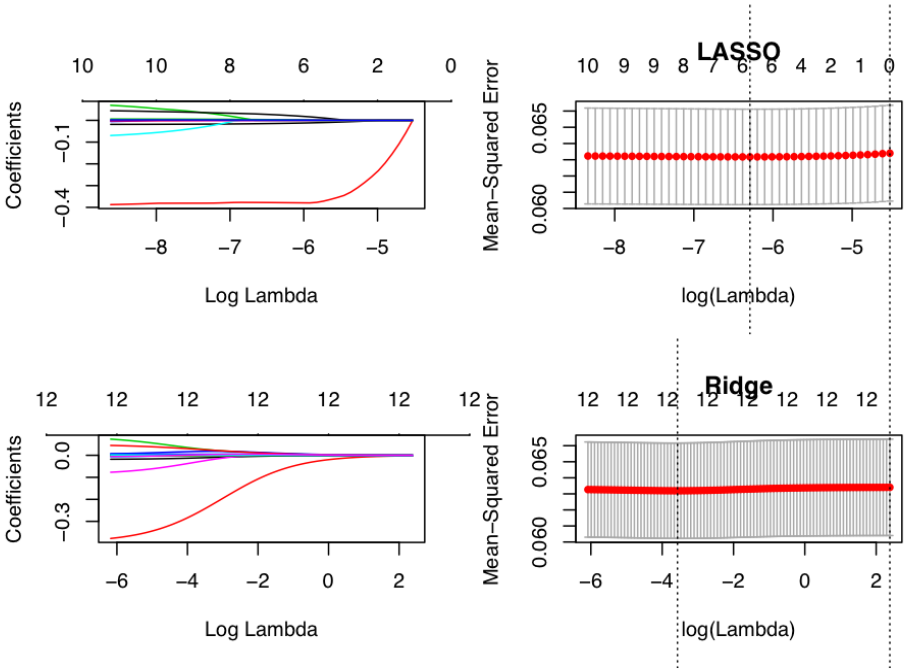
```
##        total.listings             room.type             bathrooms
##             "integer"             "integer"             "integer"
##              bedrooms                  beds              bed.type
##             "integer"             "integer"             "integer"
##                 price       persons.allowed extra.persons.allowed
##             "integer"             "integer"             "integer"
##          minimum.stay     number.of.reviews   cancellation.policy
##             "integer"             "integer"             "integer"
##    categorical.rating
##             "integer"
```

```r
sapply(glm.df, function(x) sum(length(which(is.na(x)))))
```

```
##        total.listings             room.type             bathrooms
##                     0                     0                     0
##              bedrooms                  beds              bed.type
##                     0                     0                     0
##                 price       persons.allowed extra.persons.allowed
##                     0                     0                     0
##          minimum.stay     number.of.reviews   cancellation.policy
##                     0                     0                     0
##    categorical.rating
##                     0
```

```r
# Pseudo random sample 2/3s of the row for training data
training.rows <- sample(1:dim(glm.df)[1], 1*dim(glm.df)[1])
y.training <- glm.df$categorical.rating[training.rows]
```

```
y.testing <- glm.df$categorical.rating[-training.rows]

glm.df <- glm.df[, -(13)]
x.training <- as.matrix(glm.df[training.rows, ])
x.testing <- as.matrix(glm.df[-training.rows, ])

#length(y.training)

fit.lasso <- glmnet(x.training, y.training, family="binomial", alpha=1)
fit.ridge <- glmnet(x.training, y.training, family="binomial", alpha=0)
fit.elnet <- glmnet(x.training, y.training, family="binomial", alpha=.5)

for (i in 0:10) {
    assign(paste("fit", i, sep=""), cv.glmnet(x.training, y.training, type.measure="mse'
                                              alpha=i/10,family="binomial"))
}

# Plot solution paths:
par(mfrow=c(2,2))
# Plot the glmnet fit equations.
plot(fit.lasso, xvar="lambda")
plot(fit10, main="LASSO")

plot(fit.ridge, xvar="lambda")
plot(fit0, main="Ridge")
```




```
plot(fit.elnet, xvar="lambda")
plot(fit5, main="Elastic Net")


fit = glmnet(x.training, y.training, family = "binomial")
print(fit)
```

```
##
## Call:  glmnet(x = x.training, y = y.training, family = "binomial")
##
##       Df          %Dev      Lambda
##  [1,]  0 -4.85193e-12 0.010843900
##  [2,]  1  2.15536e-03 0.009880550
##  [3,]  1  3.90404e-03 0.009002790
##  [4,]  1  5.32747e-03 0.008203010
##  [5,]  1  6.48931e-03 0.007474270
##  [6,]  2  7.63538e-03 0.006810280
##  [7,]  2  9.04653e-03 0.006205270
##  [8,]  2  1.02054e-02 0.005654010
##  [9,]  2  1.11582e-02 0.005151730
## [10,]  3  1.20729e-02 0.004694060
## [11,]  4  1.31711e-02 0.004277050
```

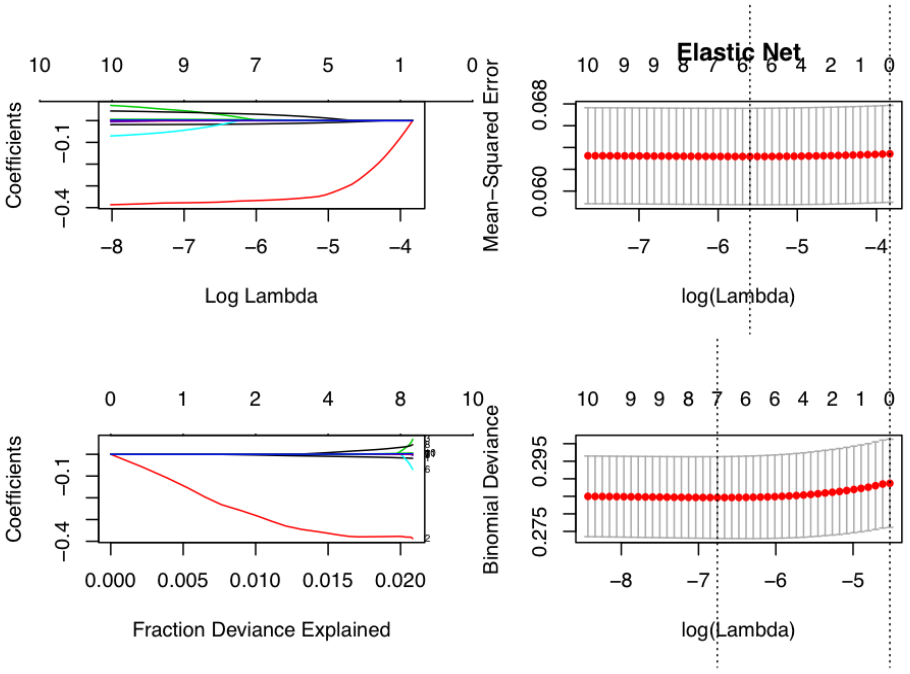
```
## [12,]  4  1.42065e-02 0.003897090
## [13,]  4  1.50662e-02 0.003550890
## [14,]  4  1.57802e-02 0.003235430
## [15,]  5  1.64239e-02 0.002948010
## [16,]  6  1.70400e-02 0.002686110
## [17,]  6  1.76104e-02 0.002447490
## [18,]  6  1.80854e-02 0.002230060
## [19,]  6  1.84808e-02 0.002031950
## [20,]  6  1.88099e-02 0.001851430
## [21,]  6  1.90838e-02 0.001686960
## [22,]  6  1.93118e-02 0.001537090
## [23,]  6  1.95014e-02 0.001400540
## [24,]  7  1.96594e-02 0.001276120
## [25,]  7  1.98159e-02 0.001162750
## [26,]  7  1.99463e-02 0.001059460
## [27,]  8  2.00636e-02 0.000965340
## [28,]  8  2.01731e-02 0.000879582
## [29,]  8  2.02644e-02 0.000801442
## [30,]  8  2.03405e-02 0.000730244
## [31,]  9  2.04164e-02 0.000665371
## [32,]  9  2.04841e-02 0.000606261
## [33,]  9  2.05403e-02 0.000552403
## [34,]  9  2.05871e-02 0.000503329
## [35,]  9  2.06259e-02 0.000458615
## [36,]  9  2.06582e-02 0.000417872
## [37,]  9  2.06850e-02 0.000380750
## [38,] 10  2.07075e-02 0.000346925
## [39,] 10  2.07368e-02 0.000316105
## [40,] 10  2.07610e-02 0.000288023
## [41,] 10  2.07812e-02 0.000262436
## [42,] 10  2.07980e-02 0.000239122
## [43,] 10  2.08119e-02 0.000217879
## [44,] 10  2.08235e-02 0.000198523
## [45,] 10  2.08331e-02 0.000180887
```

```
predict(fit,type="coef", s = 0.010843900)
```

```
## 13 x 1 sparse Matrix of class "dgCMatrix"
##                                1
## (Intercept)           3.38486834
## total.listings        .
## room.type             .
## bathrooms             .
## bedrooms              .
## beds                  .
```

```
## bed.type              .
## price                 .
## persons.allowed       .
## extra.persons.allowed .
## minimum.stay          .
## number.of.reviews     .
## cancellation.policy   .
```

```
# Plot the coefficients against the fraction deviance.
plot(fit, xvar = "dev", label = TRUE)
# Check the cross validation
cvfit = cv.glmnet(x.training, y.training, family = "binomial")
plot(cvfit)
```




```
# Get coefficients from the cross validation using lambda 1se
coef(cvfit, s = "lambda.1se")
```

```
## 13 x 1 sparse Matrix of class "dgCMatrix"
##                                1
## (Intercept)           3.38486834
## total.listings        .
## room.type             .
## bathrooms             .
```

```
## bedrooms              .
## beds                  .
## bed.type              .
## price                 .
## persons.allowed       .
## extra.persons.allowed .
## minimum.stay          .
## number.of.reviews     .
## cancellation.policy   .
```

```
## lambda.1se is a better choice in general for handling over regularization but returne
coef(cvfit, s = "lambda.min")
```

```
## 13 x 1 sparse Matrix of class "dgCMatrix"
##                                      1
## (Intercept)            3.855454236909
## total.listings        -0.015916204386
## room.type             -0.377324882900
## bathrooms              0.005937963314
## bedrooms               .
## beds                   .
## bed.type               .
## price                 -0.000381899926
## persons.allowed        0.030686966657
## extra.persons.allowed  .
## minimum.stay           0.004137713167
## number.of.reviews      0.001626593450
## cancellation.policy    .
```

```
#####################################################
# Binomial Model
#####################################################

# Convert the required variables to factor

binary.df <- working.df

# Filter our pricing that is above $1600 a night, this should remove monthly rentals
binary.df <- binary.df[(binary.df$price <= 1600), ]
binary.df$room.type <- as.factor(binary.df$room.type)
binary.df$room.type <- relevel(binary.df$room.type, ref = "Shared room")
binary.df$categorical.rating <- ifelse(binary.df$categorical.rating == 1, "Satisfied", "
binary.df$categorical.rating <- as.factor(binary.df$categorical.rating)
binary.df$categorical.rating <- relevel(binary.df$categorical.rating, ref = "Unsatisfied

sapply(binary.df,levels)
```

```
## $total.listings
## NULL
##
## $room.type
## [1] "Shared room"     "Entire home/apt" "Private room"
##
## $bathrooms
## NULL
##
## $bedrooms
## NULL
##
## $beds
## NULL
##
## $bed.type
## NULL
##
## $price
## NULL
##
## $persons.allowed
## NULL
##
## $extra.persons.allowed
## NULL
##
## $minimum.stay
## NULL
##
## $number.of.reviews
## NULL
##
## $cancellation.policy
## NULL
##
## $categorical.rating
## [1] "Unsatisfied" "Satisfied"
```

```
binary.df <- binary.df[,c("total.listings", "room.type", "price", "persons.allowed","num
binary.model <- glm(categorical.rating ~  total.listings + room.type + price +
                      persons.allowed + number.of.reviews, family = "binomial",data = bi

summary(binary.model)
```

```
##
## Call:
## glm(formula = categorical.rating ~ total.listings + room.type +
##     price + persons.allowed + number.of.reviews, family = "binomial",
##     data = binary.df)
##
## Deviance Residuals:
##       Min         1Q     Median         3Q        Max
## -3.753108   0.157100   0.244860   0.301867   1.314955
##
## Coefficients:
##                              Estimate  Std. Error  z value   Pr(>|z|)
## (Intercept)                2.76862123  0.39429397  7.02172 2.1916e-12 ***
## total.listings            -0.00707866  0.00176060 -4.02059 5.8053e-05 ***
## room.typeEntire home/apt   0.06976894  0.40804506  0.17098  0.8642368
## room.typePrivate room     -0.36319847  0.39626907 -0.91655  0.3593811
## price                      0.00316609  0.00111712  2.83414  0.0045949 **
## persons.allowed           -0.08288244  0.06556697 -1.26409  0.2061983
## number.of.reviews          0.04827052  0.00626850  7.70049 1.3554e-14 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 2808.234  on 9733  degrees of freedom
## Residual deviance: 2631.672  on 9727  degrees of freedom
## AIC: 2645.672
##
## Number of Fisher Scoring iterations: 8
```

```
binary.model.2 <- glm(categorical.rating ~  total.listings + price +
                      number.of.reviews, family = "binomial",data = binary.df)
summary(binary.model.2)
```

```
##
## Call:
## glm(formula = categorical.rating ~ total.listings + price + number.of.reviews,
##     family = "binomial", data = binary.df)
##
## Deviance Residuals:
##       Min         1Q     Median         3Q        Max
## -3.822128   0.156320   0.250875   0.314090   1.366276
##
## Coefficients:
##                       Estimate   Std. Error  z value   Pr(>|z|)
```

```
## (Intercept)        2.393104841  0.115294456 20.75646 < 2.22e-16 ***
## total.listings    -0.007007465  0.001765369 -3.96941 7.2052e-05 ***
## price              0.004450835  0.000955481  4.65821 3.1897e-06 ***
## number.of.reviews  0.048656239  0.006232580  7.80676 5.8678e-15 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 2808.234  on 9733  degrees of freedom
## Residual deviance: 2642.363  on 9730  degrees of freedom
## AIC: 2650.363
##
## Number of Fisher Scoring iterations: 8
```

```
binary.model.3 <- glm(categorical.rating ~  total.listings + number.of.reviews, family =
summary(binary.model.3)
```

```
##
## Call:
## glm(formula = categorical.rating ~ total.listings + number.of.reviews,
##     family = "binomial", data = binary.df)
##
## Deviance Residuals:
##       Min         1Q     Median         3Q        Max
## -3.894749   0.170991   0.264999   0.313738   1.322319
##
## Coefficients:
##                     Estimate  Std. Error  z value   Pr(>|z|)
## (Intercept)        2.84536546  0.07521119 37.83168 < 2.22e-16 ***
## total.listings    -0.00628441  0.00173879 -3.61424 0.00030123 ***
## number.of.reviews  0.04925146  0.00623533  7.89878 2.8165e-15 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 2808.234  on 9733  degrees of freedom
## Residual deviance: 2670.671  on 9731  degrees of freedom
## AIC: 2676.671
##
## Number of Fisher Scoring iterations: 8
```

```
exp(coefficients(binary.model))
```

```
##              (Intercept)           total.listings room.typeEntire home/apt
```

```
##             15.936645942              0.992946339              1.072260401
##    room.typePrivate room                    price          persons.allowed
##              0.695448393              1.003171108              0.920459349
##        number.of.reviews
##              1.049454513
```

```
exp(coefficients(binary.model.2))
```

```
##       (Intercept)    total.listings             price number.of.reviews
##       10.94743126        0.99301703        1.00446075        1.04985939
```

```
exp(coefficients(binary.model.3))
```

```
##       (Intercept)    total.listings number.of.reviews
##      17.207846325       0.993735293       1.050484467
```

```
# Likelihood ratio and Anova model fit tests

lrtest(binary.model, binary.model.2)
```

```
## Likelihood ratio test
##
## Model 1: categorical.rating ~ total.listings + room.type + price + persons.allowed +
##     number.of.reviews
## Model 2: categorical.rating ~ total.listings + price + number.of.reviews
##   #Df    LogLik Df    Chisq Pr(>Chisq)
## 1   7 -1315.836
## 2   4 -1321.181 -3 10.69056   0.013522 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
anova(binary.model.2, binary.model, test = "Chisq")
```

```
## Analysis of Deviance Table
##
## Model 1: categorical.rating ~ total.listings + price + number.of.reviews
## Model 2: categorical.rating ~ total.listings + room.type + price + persons.allowed +
##     number.of.reviews
##   Resid. Df Resid. Dev Df Deviance Pr(>Chi)
## 1      9730   2642.363
## 2      9727   2631.672  3 10.69056 0.013522 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
lrtest(binary.model.2)
```

```
## Likelihood ratio test
##
```

```
## Model 1: categorical.rating ~ total.listings + price + number.of.reviews
## Model 2: categorical.rating ~ 1
##   #Df    LogLik Df    Chisq Pr(>Chisq)
## 1   4 -1321.181
## 2   1 -1404.117 -3 165.8713 < 2.22e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
# Do not reject H0, using alpha = 0.01 and significance of predictors in model 1 are l
if(!require("pscl")){install.packages("pscl")}
pscl::pR2(binary.model.2)
```

```
##              llh         llhNull              G2        McFadden
## -1.32118150e+03 -1.40411717e+03  1.65871341e+02  5.90660611e-02
##            r2ML            r2CU
##  1.68960424e-02  6.74193169e-02
```

```
#HLgof.test(fit = fitted(binary.model), obs = binary.df$categorical.rating)
# hoslem.test(binary.model$categorical.rating ,binary.model)




# So we have a left skewed distribution for stars.
# Change the review scores into categorical
# working.df$categorical.rating <- cut(working.df$` reviews.score`,
                       # breaks = c(-Inf, 40, 60, 80, 100, Inf),
                       # labels = c("1-1.5 stars", "2-2.5 stars", "3-3.5 stars", "4-4.5
                       # right = FALSE)

# Convert to 0 to represent under 4 stars and 1 for 4 stars and above

# Subset regression is too slow
# regsubsets.out <-
#     regsubsets(bwt ~ age + lwt + race.cat + smoke + preterm + ht + ui + ftv.cat,
#                data = lbw,
#                nbest = 1,       # 1 best model for each number of predictors
#                nvmax = NULL,    # NULL for no limit on number of variables
#                force.in = NULL, force.out = NULL,
#                method = "exhaustive")
# fit.1 <- lm(working.df$` reviews.score` ~., data = working.df)
# summary(fit.1)
# car::vif(fit.1)
# a gvif^(1/(2*df)) is near 2 so we may want to remove these variables.

# binary.df$bed.type <- as.factor(binary.df$bed.type)
```

```
# binary.df$room.type <- as.factor(binary.df$room.type)
# binary.df$cancellation.policy <- as.factor(binary.df$cancellation.policy)
# sapply(working.df, levels)

## Cumulative logistic regression
# https://rpubs.com/kaz_yos/VGAM
# Begin by not assuming parallel
# cumulative.df <- working.df
# model.1 <- vglm(categorical.rating ~ ., family=cumulative(parallel=FALSE), data = cu
# summary(model.1)
# is.parallel(model.1)
# Since we see strong evidence of parallelism run model 2
# model.2 <- vglm(categorical.rating ~ ., family=cumulative(parallel=TRUE), data = cum
# summary(model.2)

# Polr will not work for our data
#model.3 <- polr(formula = categorical.rating ~ ., data = cumulative.df, Hess = TRUE)

# Remove some variables based on low significance as AIC is not reported
# cumulative.df <- cumulative.df[, -c(4,5,8,9)]

# sapply(cumulative.df, typeof)
# model.4 <- vglm(categorical.rating ~ ., family=cumulative(parallel=TRUE), data = cum
# summary(model.4)
# lrtest_vglm(model.2,model.4)
#
# ResourceSelection::hoslem.test(model.2, fitted(model.2), g =10 )

# We should clean some observations before glmnet
# https://www.r-bloggers.com/identify-describe-plot-and-remove-the-outliers-from-the-d

# outlierKD <- function(dt, var) {
#      var_name <- eval(substitute(var),eval(dt))
#      na1 <- sum(is.na(var_name))
#      m1 <- mean(var_name, na.rm = T)
#      par(mfrow=c(2, 2), oma=c(0,0,3,0))
#      boxplot(var_name, main="With outliers")
#      hist(var_name, main="With outliers", xlab=NA, ylab=NA)
#      outlier <- boxplot.stats(var_name)$out
#      mo <- mean(outlier)
#      var_name <- ifelse(var_name %in% outlier, NA, var_name)
#      boxplot(var_name, main="Without outliers")
#      hist(var_name, main="Without outliers", xlab=NA, ylab=NA)
#      title("Outlier Check", outer=TRUE)
```

```
#       na2 <- sum(is.na(var_name))
#       cat("Outliers identified:", na2 - na1, "\n")
#       cat("Propotion (%) of outliers:", round((na2 - na1) / sum(!is.na(var_name))*100
#       cat("Mean of the outliers:", round(mo, 2), "\n")
#       m2 <- mean(var_name, na.rm = T)
#       cat("Mean without removing outliers:", round(m1, 2), "\n")
#       cat("Mean if we remove outliers:", round(m2, 2), "\n")
#       response <- readline(prompt="Do you want to remove outliers and to replace with
#       if(response == "y" | response == "yes"){
#            dt[as.character(substitute(var))] <- invisible(var_name)
#            assign(as.character(as.list(match.call())$dt), dt, envir = .GlobalEnv)
#            cat("Outliers successfully removed", "\n")
#            return(invisible(dt))
#       } else{
#            cat("Nothing changed", "n")
#            return(invisible(var_name))
#       }
# }
# Model 1
# Multinomial
#if(!require('nnet')){install.packages("nnet")}
#multinom.df$overall_satisfaction <- factor(multinom.df$overall_satisfaction, ordered
#multinom.df$bedrooms <- factor(multinom.df$bedrooms)
#multinom.df$accommodates <- factor(multinom.df$accommodates)
#multinom.df$neighborhood <- factor(multinom.df$neighborhood)

# Check which are factors
#sapply(multinom.df, levels)
# Check the data type of each column.
#sapply(multinom.df, typeof)

# multi.model <- nnet::multinom(formula = overall_satisfaction ~ . ,
#                               MaxNWts = 2000,
#                               maxit = 350,
#                               data = multinom.df)
# model.summary.1 <- summary(multi.model)
# model.summary.1


# Model 2
# Link: https://cran.r-project.org/web/packages/VGAM/VGAM.pdf
#Multinomial model is complex so we look at cumulative logistic
#if(!require("MASS")){install.packages("MASS")}
# sapply(cumulative.df, levels)
```

```
# cumulative.df <- multinom.df
# cumulative.df<-cumulative.df[!(cumulative.df$price >= 100),]
# cumulative.df$overall_satisfaction<- as.integer(as.character(cumulative.df$overall_s
# sapply(cumulative.df, typeof)
# #cumulative.df$overall_satisfaction <- cumulative.df$overall_satisfaction * 2
# cumulative.df$overall_satisfaction <- factor(cumulative.df$overall_satisfaction, ord
# sapply(cumulative.df, levels)

# Try and figure out Hauck-Donner effect error
```