Airbnb Model Selection

Ariana Bucio

11/21/2019

```
# Ariana Bucio
# Jadyn Gonzalez
# Matt Mead
# Bryce Viorst


######################
# Nov. 14, Submission#
######################
# Setting working directory and importing data
setwd("~/Desktop/MGSC310-Project-master")
nycAB = read.csv("AB_NYC_2019.csv")
# Dropping name and host name as these variables will not provide any insights
nycAB = nycAB[ , !(names(nycAB) %in% c("name","id","host_name", "latitude", "longitude", "last_review",

# Check for missing values
sapply(nycAB, function(x) sum(is.na(x)))
```

```
##             host_id neighbourhood_group       neighbourhood
##                   0                   0                   0
##           room_type               price      minimum_nights
##                   0                   0                   0
##   number_of_reviews   reviews_per_month    availability_365
##                   0               10052                   0
```

```
# It looks like the only missing value is in reviews per month
# We can handle this by just setting NaN values to 0
nycAB[is.na(nycAB)] = 0
sum(is.na(nycAB))
```

```
## [1] 0
```

```
# Check structure of data
str(nycAB)
```

```
## 'data.frame':    48895 obs. of  9 variables:
##  $ host_id            : int  2787 2845 4632 4869 7192 7322 7356 8967 7490 7549 ...
##  $ neighbourhood_group: Factor w/ 5 levels "Bronx","Brooklyn",..: 2 3 3 2 3 3 2 3 3 3 ...
##  $ neighbourhood      : Factor w/ 221 levels "Allerton","Arden Heights",..: 109 128 95 42 62 138 14 9
##  $ room_type          : Factor w/ 3 levels "Entire home/apt",..: 2 1 2 1 1 1 2 2 2 1 ...
##  $ price              : int  149 225 150 89 80 200 60 79 79 150 ...
##  $ minimum_nights     : int  1 1 3 1 10 3 45 2 2 1 ...
##  $ number_of_reviews  : int  9 45 0 270 9 74 49 430 118 160 ...
##  $ reviews_per_month  : num  0.21 0.38 0 4.64 0.1 0.59 0.4 3.47 0.99 1.33 ...
##  $ availability_365   : int  365 355 365 194 0 129 0 220 0 188 ...
```

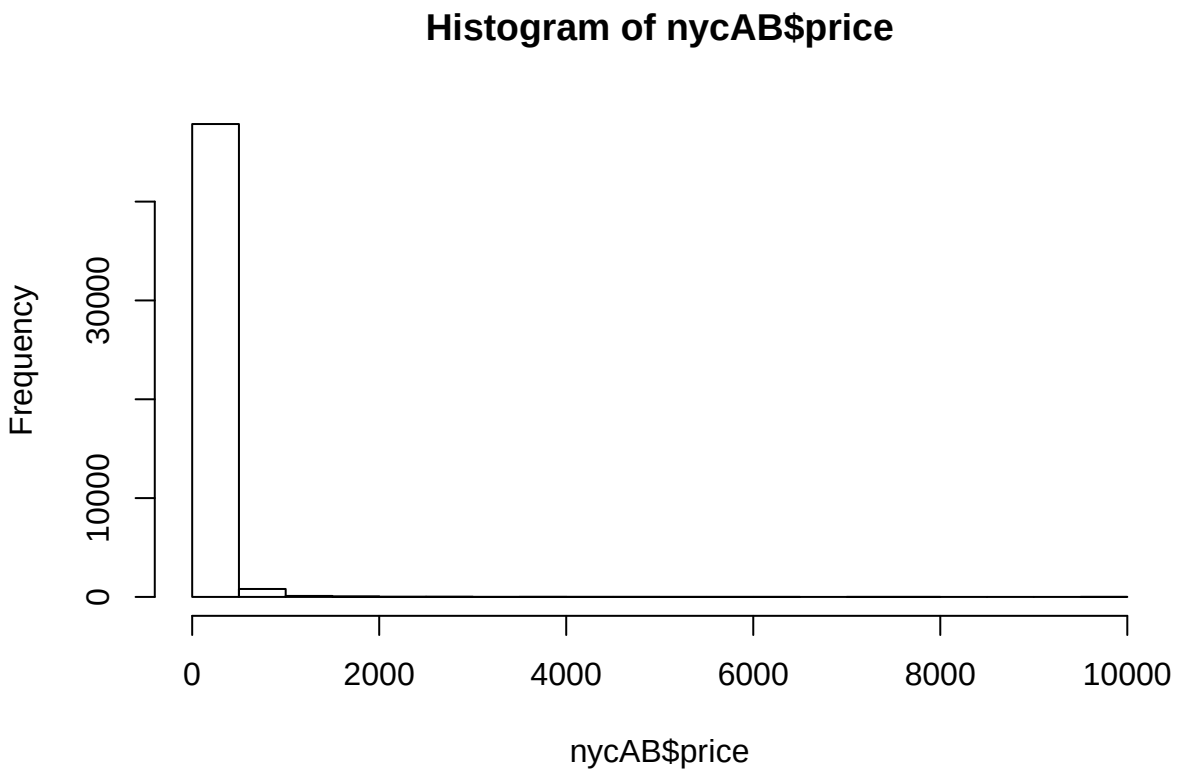
```r
nycAB$host_id = as.factor(nycAB$host_id)
str(nycAB)
```

```
## 'data.frame':    48895 obs. of  9 variables:
##  $ host_id            : Factor w/ 37457 levels "2438","2571",..: 3 4 14 15 19 22 25 37 26 28 ...
##  $ neighbourhood_group: Factor w/ 5 levels "Bronx","Brooklyn",..: 2 3 3 2 3 3 2 3 3 3 ...
##  $ neighbourhood      : Factor w/ 221 levels "Allerton","Arden Heights",..: 109 128 95 42 62 138 14 9
##  $ room_type          : Factor w/ 3 levels "Entire home/apt",..: 2 1 2 1 1 1 2 2 2 1 ...
##  $ price              : int  149 225 150 89 80 200 60 79 79 150 ...
##  $ minimum_nights     : int  1 1 3 1 10 3 45 2 2 1 ...
##  $ number_of_reviews  : int  9 45 0 270 9 74 49 430 118 160 ...
##  $ reviews_per_month  : num  0.21 0.38 0 4.64 0.1 0.59 0.4 3.47 0.99 1.33 ...
##  $ availability_365   : int  365 355 365 194 0 129 0 220 0 188 ...
```

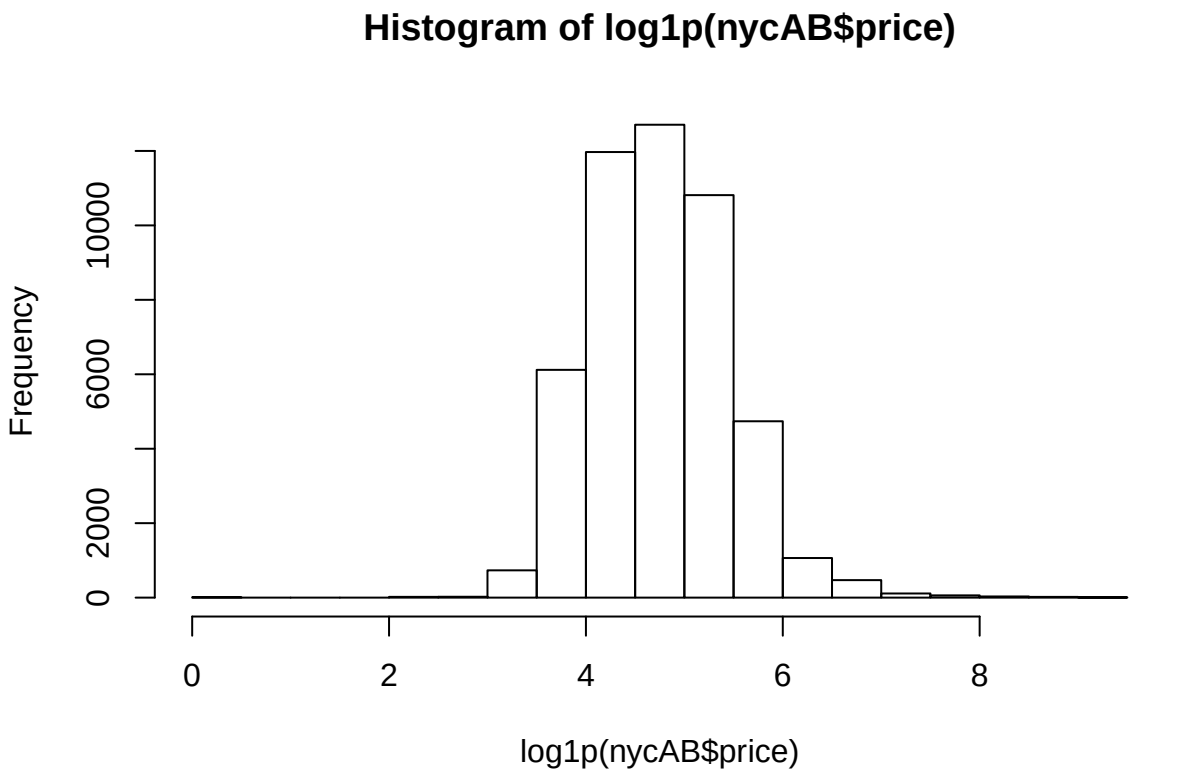
```r
# Summary of data
summary(nycAB)
```

```
##       host_id         neighbourhood_group            neighbourhood
##  219517861:  327   Bronx        : 1091    Williamsburg      : 3920
##  107434423:  232   Brooklyn     :20104    Bedford-Stuyvesant: 3714
##  30283594 :  121   Manhattan    :21661    Harlem            : 2658
##  137358866:  103   Queens       : 5666    Bushwick          : 2465
##  12243051 :   96   Staten Island:  373    Upper West Side   : 1971
##  16098958 :   96                         Hell's Kitchen    : 1958
##  (Other)  :47920                         (Other)           :32209
##            room_type         price          minimum_nights
##  Entire home/apt:25409   Min.   :    0.0   Min.   :   1.00
##  Private room   :22326   1st Qu.:   69.0   1st Qu.:   1.00
##  Shared room    : 1160   Median :  106.0   Median :   3.00
##                          Mean   :  152.7   Mean   :   7.03
##                          3rd Qu.:  175.0   3rd Qu.:   5.00
##                          Max.   :10000.0   Max.   :1250.00
##
##  number_of_reviews reviews_per_month availability_365
##  Min.   :  0.00    Min.   : 0.000    Min.   :  0.0
##  1st Qu.:  1.00    1st Qu.: 0.040    1st Qu.:  0.0
##  Median :  5.00    Median : 0.370    Median : 45.0
##  Mean   : 23.27    Mean   : 1.091    Mean   :112.8
##  3rd Qu.: 24.00    3rd Qu.: 1.580    3rd Qu.:227.0
##  Max.   :629.00    Max.   :58.500    Max.   :365.0
##
```

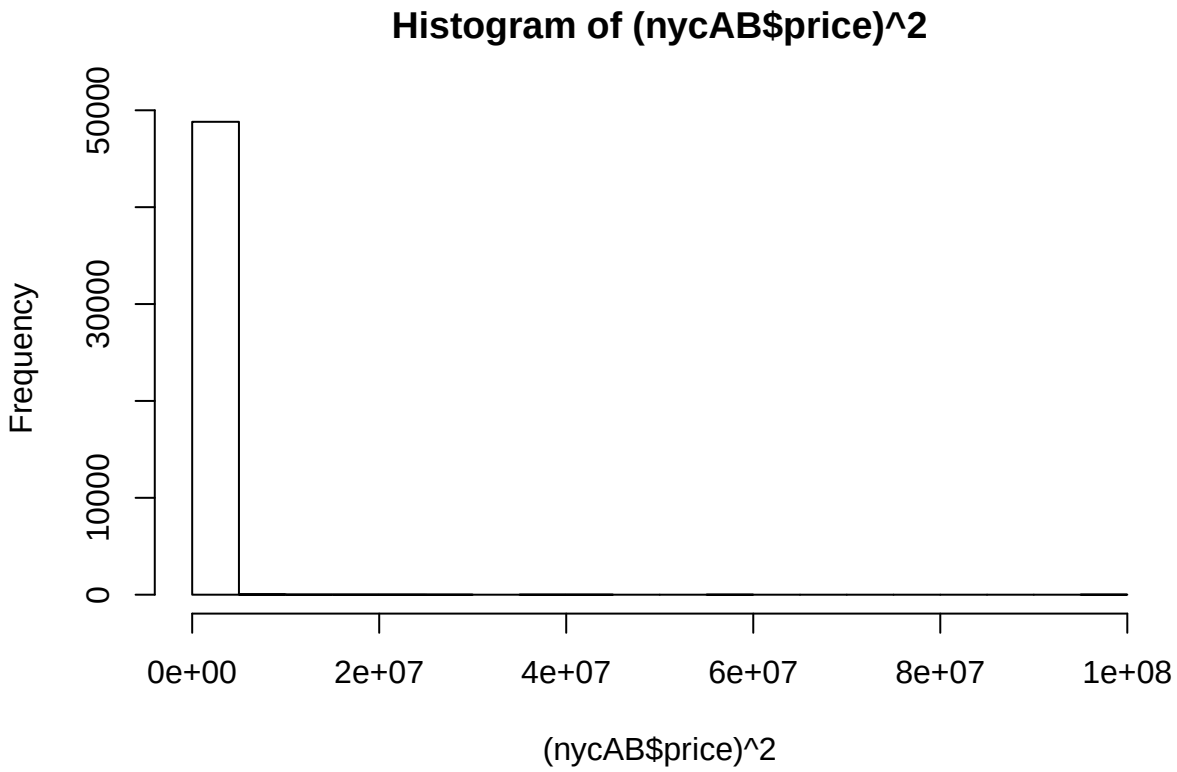
```r
# Checking distributions of data
hist(nycAB$price)
```



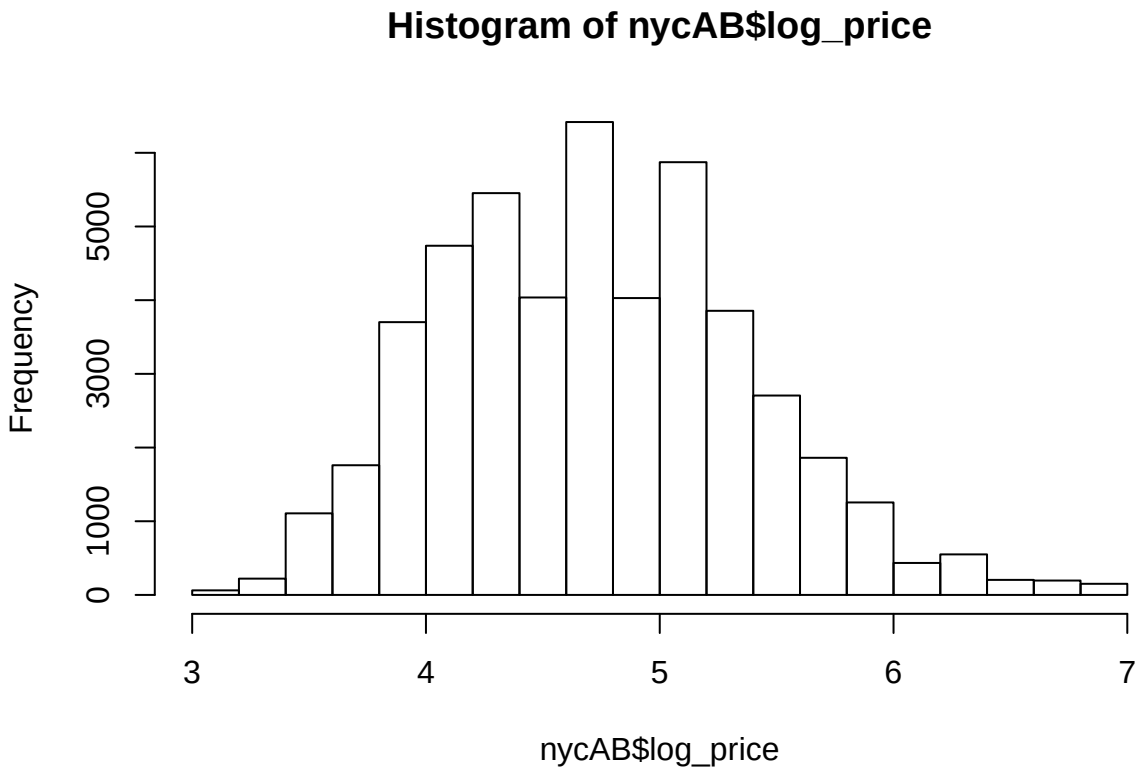

```
# Not a good distrubution, try log and squared
hist(log1p(nycAB$price))
```



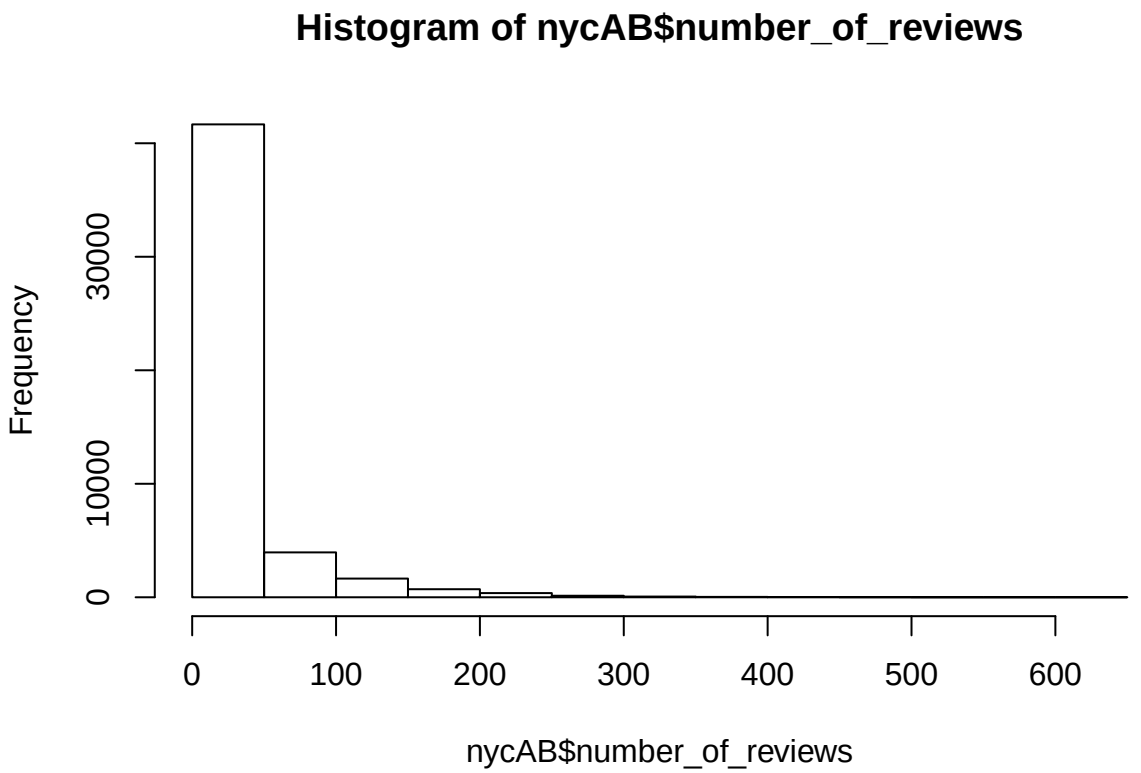
```r
hist((nycAB$price)^2)
```



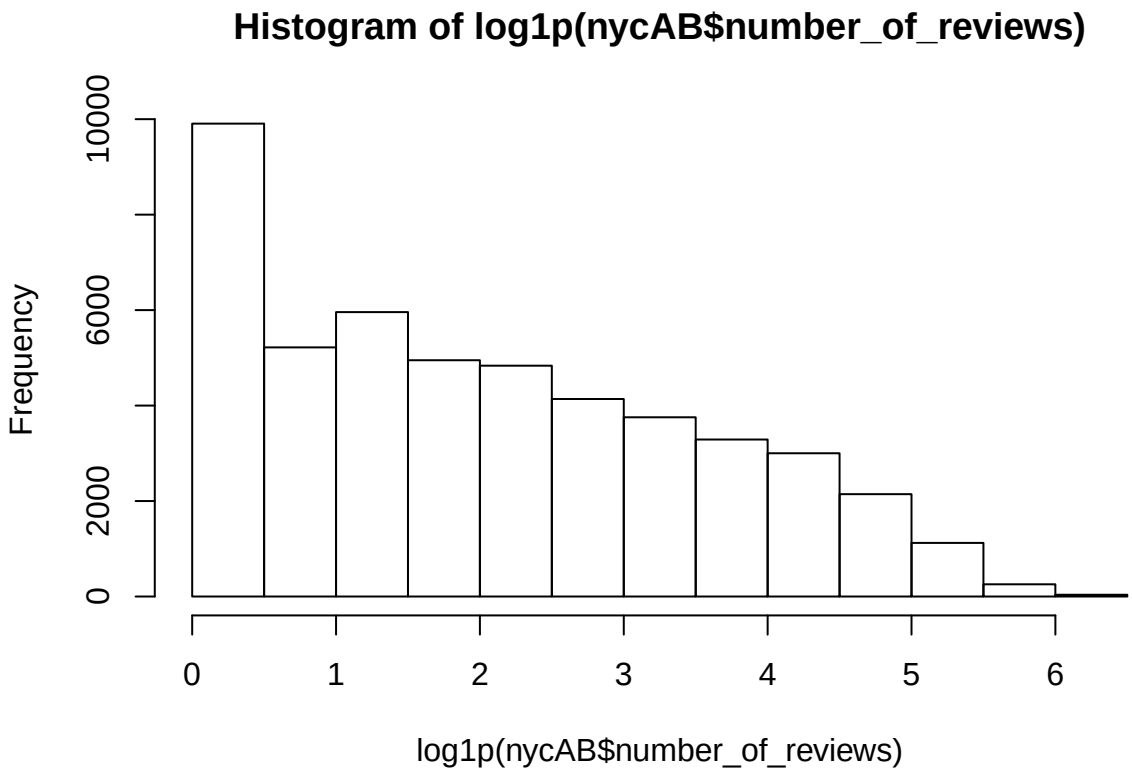

```r
# Log looks much better so well add the log variable to the set
nycAB$log_price = log1p(nycAB$price)
# Remove outliers with log price > 7 and < 3
nycAB = nycAB[!(nycAB$log_price < 3),]
nycAB = nycAB[!(nycAB$log_price > 7),]
hist(nycAB$log_price)
```



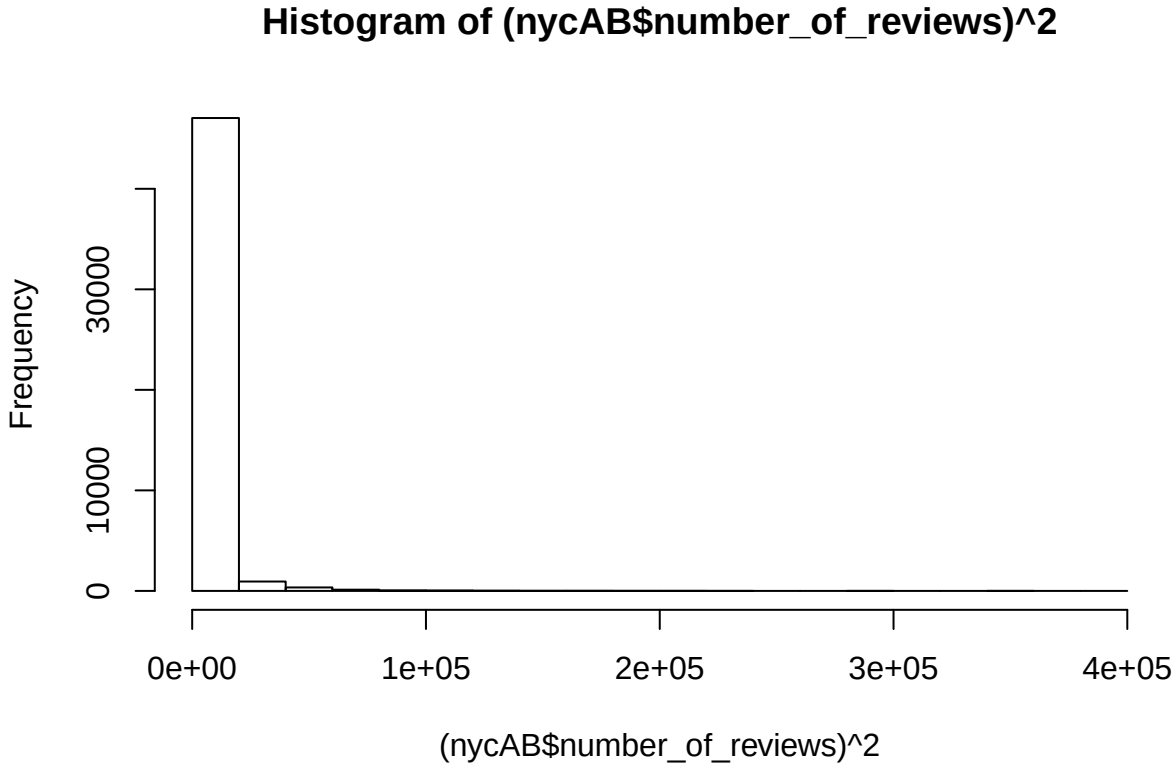

```
hist(nycAB$number_of_reviews)
```



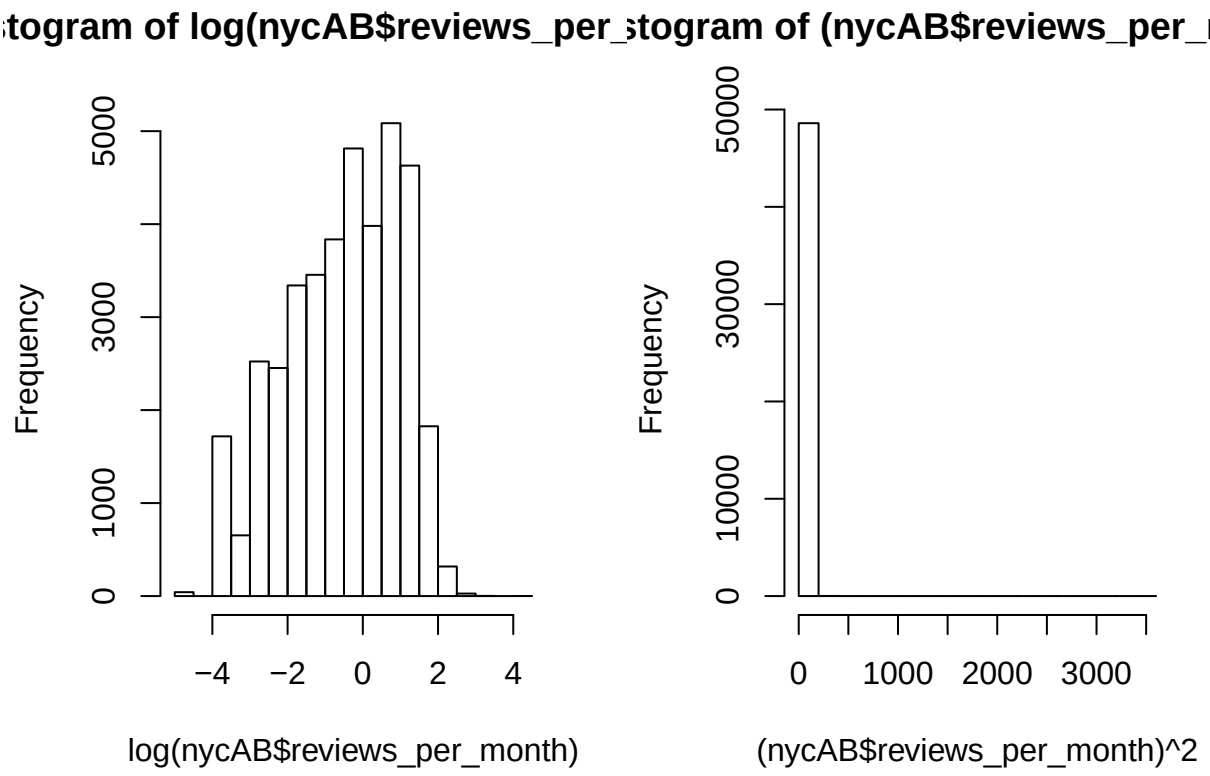
```r
# Skewed distribution as well
hist(log1p(nycAB$number_of_reviews))
```




```r
hist((nycAB$number_of_reviews)^2)
```



```
# Again log distribution is much better so we'll add that
nycAB$log_num_reviews = log1p(nycAB$number_of_reviews)

par(mfrow=c(1,2))
hist(log(nycAB$reviews_per_month))
hist((nycAB$reviews_per_month)^2)
```




```
nycAB$reviews_per_month_log = log1p(nycAB$reviews_per_month)

par(mfrow=c(1,2))
hist(log(nycAB$minimum_nights))
hist((nycAB$minimum_nights)^2)
#Log looks better, so let's use that.
nycAB$minimum_nights_log = log1p(nycAB$minimum_nights)

# Investigate listings on the neighborhood groups
levels(nycAB$neighbourhood_group)
```

```
## [1] "Bronx"         "Brooklyn"      "Manhattan"     "Queens"
## [5] "Staten Island"
```

```
# Looking at average price, max, and min for each group
mean(nycAB$price[nycAB$neighbourhood_group=="Bronx"])
```

```
## [1] 85.43107
```

```r
max(nycAB$price[nycAB$neighbourhood_group=="Bronx"])
```

```
## [1] 1000
```

```r
min(nycAB$price[nycAB$neighbourhood_group=="Bronx"])
```

```
## [1] 20
```

```r
mean(nycAB$price[nycAB$neighbourhood_group=="Brooklyn"])
```

```
## [1] 118.1739
```

```r
max(nycAB$price[nycAB$neighbourhood_group=="Brooklyn"])
```

```
## [1] 1095
```

```r
min(nycAB$price[nycAB$neighbourhood_group=="Brooklyn"])
```

```
## [1] 20
```

```r
mean(nycAB$price[nycAB$neighbourhood_group=="Manhattan"])
```

```
## [1] 179.3047
```

```r
max(nycAB$price[nycAB$neighbourhood_group=="Manhattan"])
```

```
## [1] 1075
```

```r
min(nycAB$price[nycAB$neighbourhood_group=="Manhattan"])
```

```
## [1] 20
```

```r
mean(nycAB$price[nycAB$neighbourhood_group=="Queens"])
```

```
## [1] 95.16634
```

```r
max(nycAB$price[nycAB$neighbourhood_group=="Queens"])
```

```
## [1] 1000
```

```r
min(nycAB$price[nycAB$neighbourhood_group=="Queens"])
```

```
## [1] 20
```

```r
mean(nycAB$price[nycAB$neighbourhood_group=="Staten Island"])
```

```
## [1] 98.81622
```

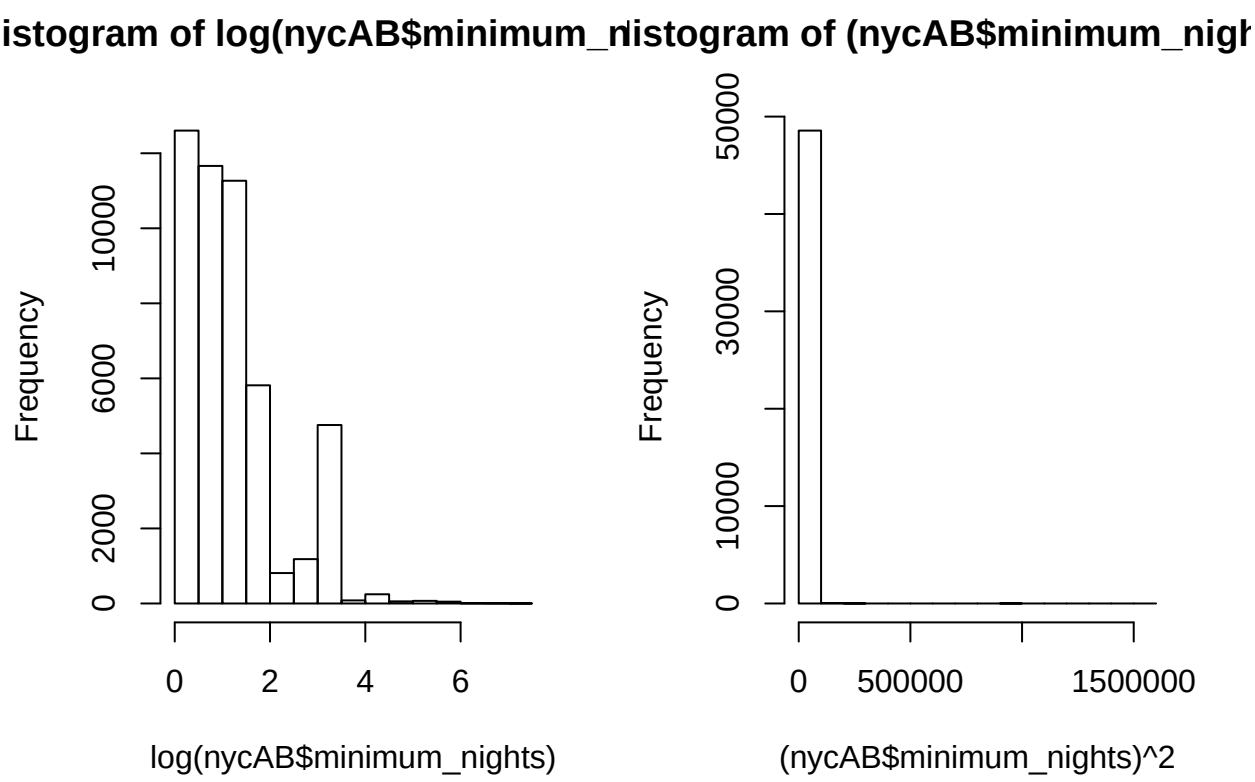
```r
max(nycAB$price[nycAB$neighbourhood_group=="Staten Island"])
```

```
## [1] 1000
```

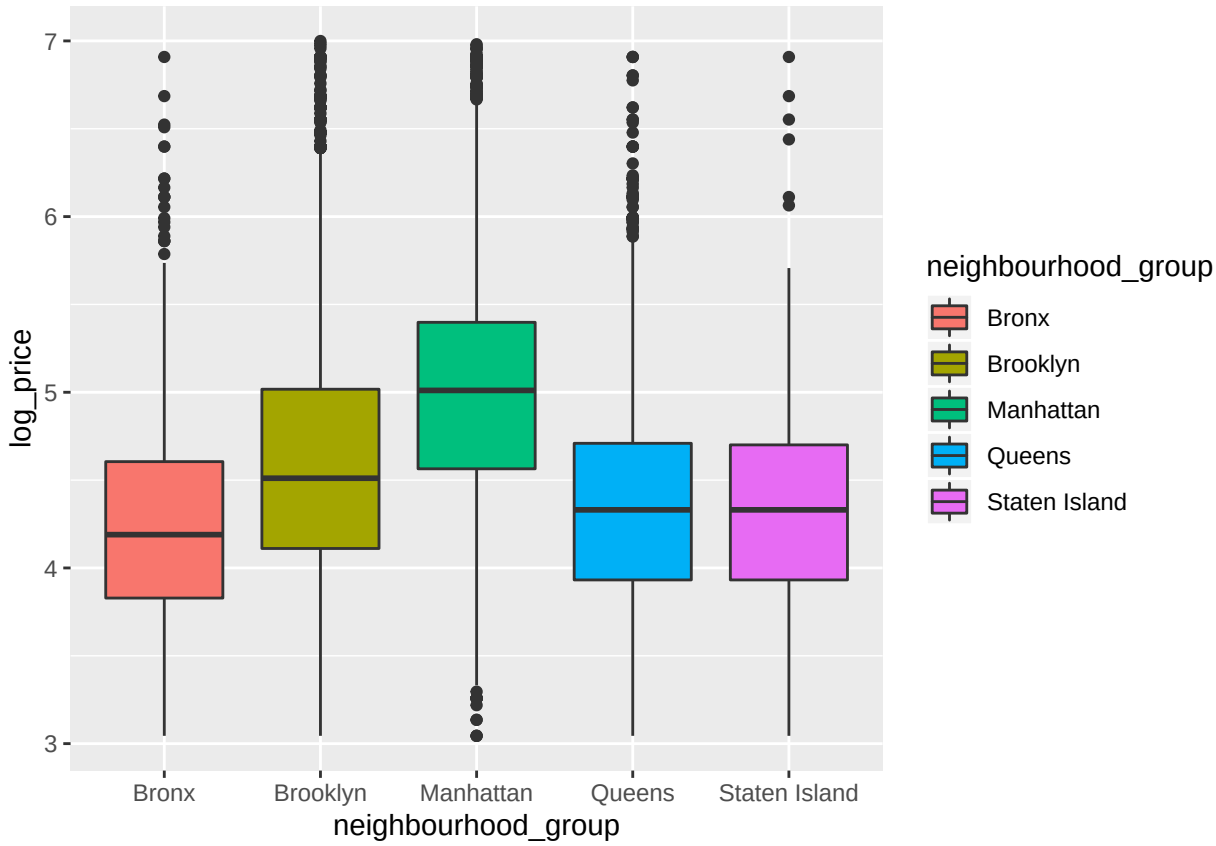
```r
min(nycAB$price[nycAB$neighbourhood_group=="Staten Island"])
```

```
## [1] 20
```

```r
# The Bronx has the lowest average price as well as the lowest max price
# Manhattan has the highest average price
library(ggplot2)
```



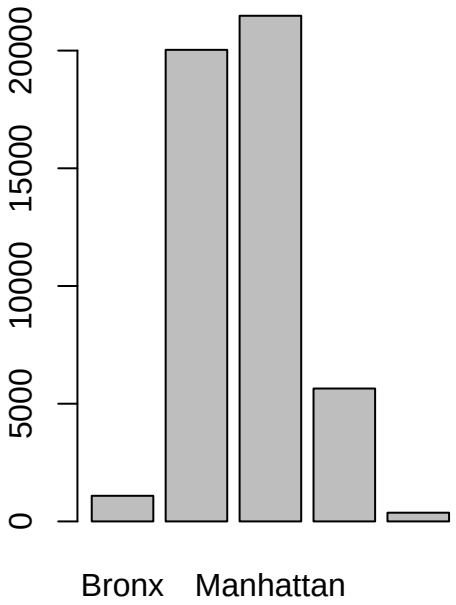

```r
# Summary from the means/max/min reflects in the boxplot
# Have to use log transformed to get best visualization
ggplot(nycAB,aes(x=neighbourhood_group, y=log_price, fill=neighbourhood_group)) + geom_boxplot()
```

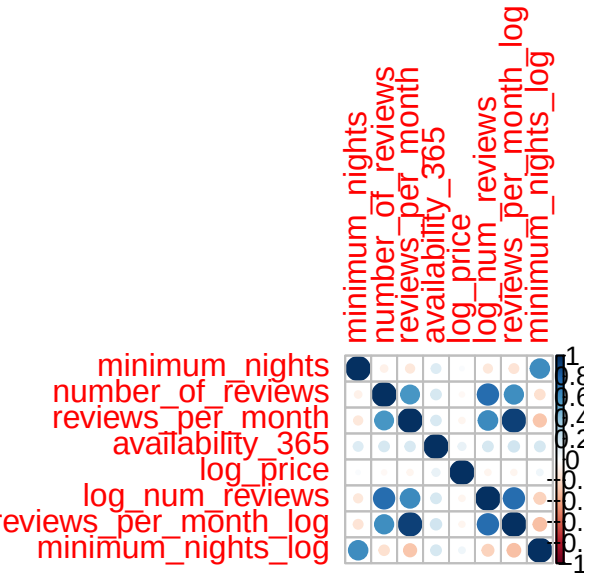



```
# We see that Manhattan has the higher price range compared to Queens and Staten Island, this
# makes sense as Manhattan is more of a city/tourist/business area while Queens and Staten Island are
# more residential areas, not sure about Bronx and Brooklyn.
# Let's check the count of listings, this may influence the distributions.
barplot(table(nycAB$neighbourhood_group))
# We see that Brooklyn and Manhattan have far more listings than the other cities.


# Lets check for correlations
library(corrplot)
```

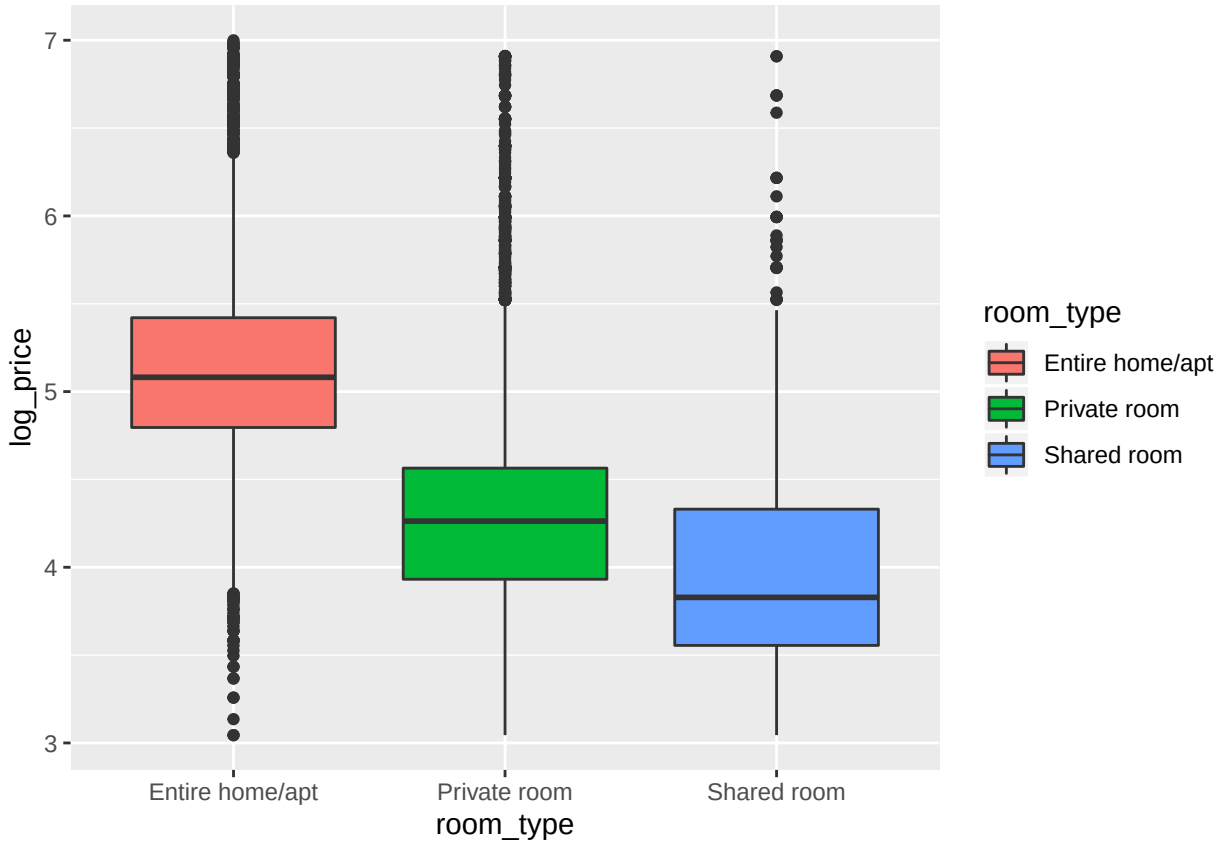
```
## corrplot 0.84 loaded
```

```
corrplot(cor(nycAB[, c(6,7,8,9,10,11,12,13)]))
```



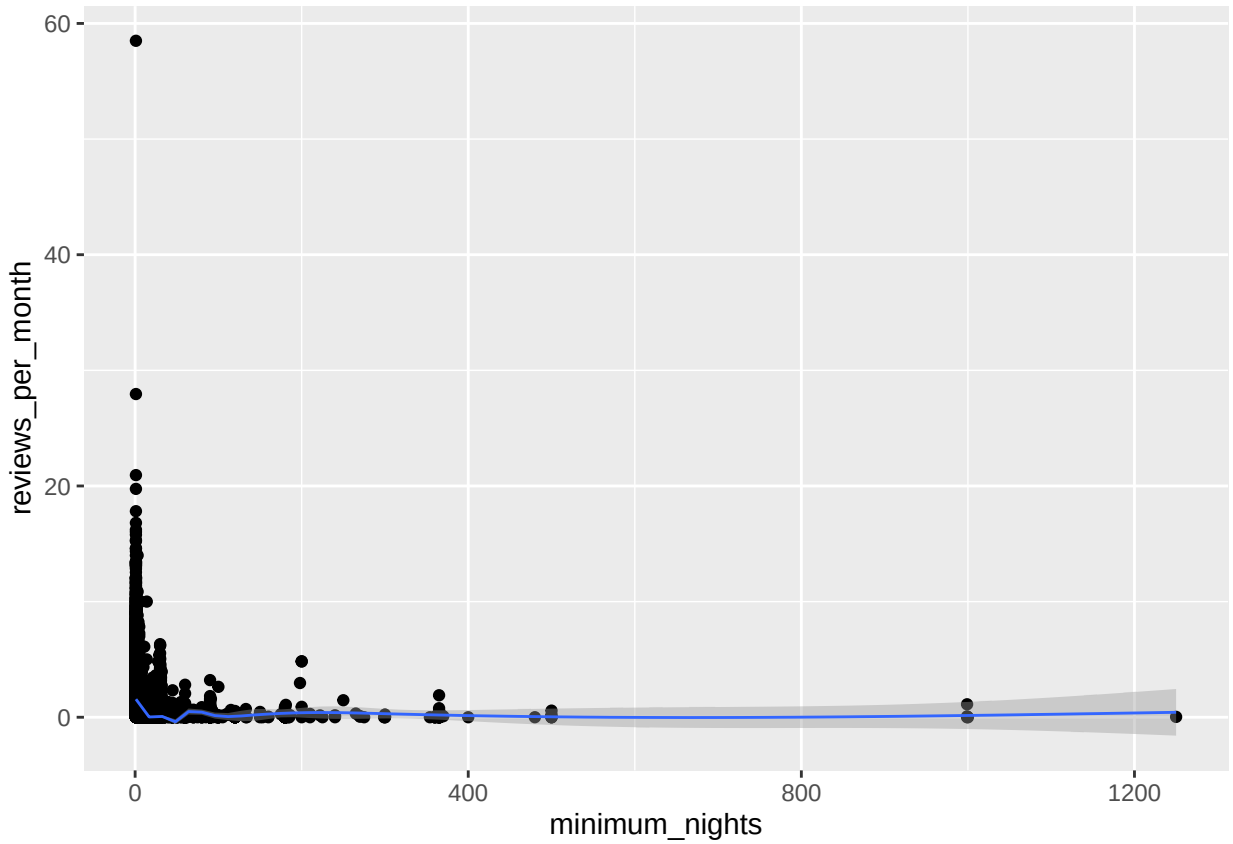



```
# We see that count of host listings and number of days available per year have some positive correlati
# with price, number of reviews and reviews per month shows some negative correlation but this may be
# insignificant as there  is no way to distinguise between positive and negative reviews

ggplot(nycAB,aes(x=room_type, y=log_price, fill = room_type)) + geom_boxplot()
```



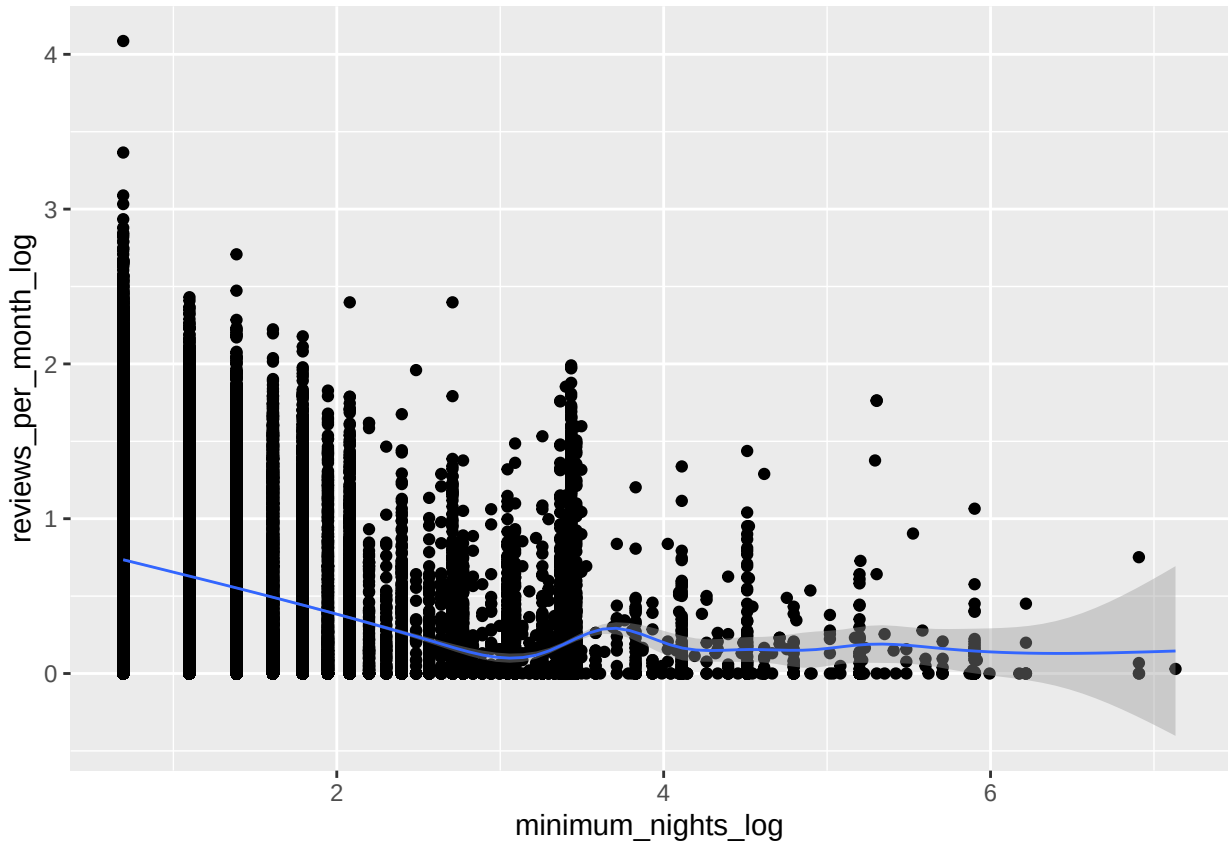
```
# We can see that listings where the entire home/apartment
#is offered generally has the highest price range, whereas
#a private room has the second highest price range,
#with a shared room falling in the lowest price range

ggplot(nycAB,aes(x=minimum_nights, y=reviews_per_month)) + geom_point() + geom_smooth(size = .5)
```

```
## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
```




```
#This plot shows a relationship between minimum nights and
#reviews per month. It shows that there is a negative
#relationship up to a certain point (as minimum nights increases,
#reviews per month decreases) and then it begins to even out,
#as there is little to no relationship between the two after
#a certain point

ggplot(nycAB,aes(x=minimum_nights_log, y=reviews_per_month_log)) + geom_point() + geom_smooth(size = .5)
```

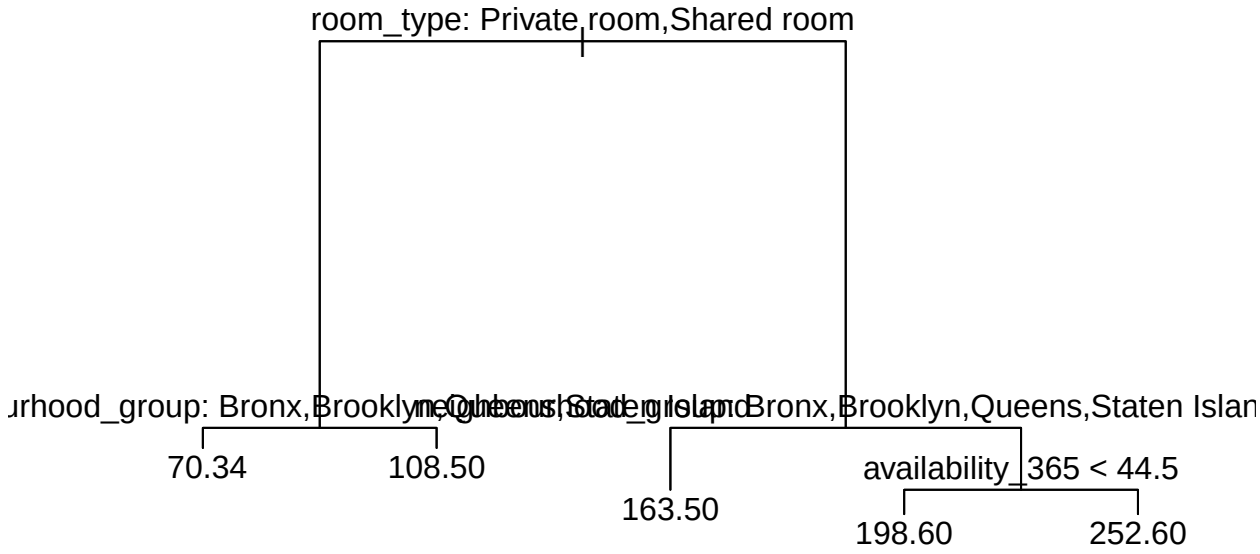
```
## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
```



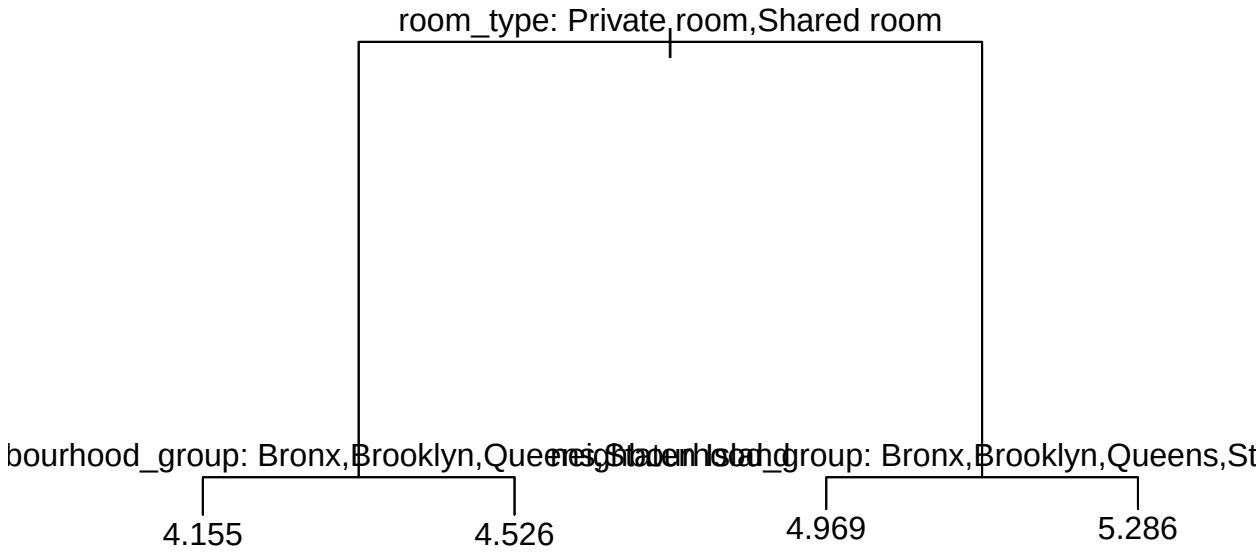

```
#this plot shows the relationship a between the two variables
#a bit more cleanly, as we have found the log of both.
```

```
############################
### Nov. 21, Submission ###
############################

# We'll run a regression model using a decision tree
# First we'll set the seed and split the data
set.seed(310)
trainidx = sample(1:nrow(nycAB),size=0.75*nrow(nycAB))
train = nycAB[trainidx,]
test = nycAB[-trainidx,]

#making sample of train and test data for random forest model
#efficiency
train_sample = train[sample(1:nrow(train),size=0.3*nrow(train)), ]
test_sample = test[sample(1:nrow(test),size=0.3*nrow(test)), ]
# Next well run two tree models, one with price and one with the log transform price
library(tree)
regMod = tree(price ~ neighbourhood_group + room_type
              + availability_365 + log_num_reviews + reviews_per_month_log
              + minimum_nights_log,
              data = train)

logMod = tree(log_price ~ neighbourhood_group + room_type
```

```
              + availability_365 + log_num_reviews + reviews_per_month_log
              + minimum_nights_log,
              data = train)

# We'll plot both
plot(regMod)
text(regMod, pretty=0)
```




```
plot(logMod)
text(logMod, pretty = 0)
```




```
# In both trees we can see that the variable that affects price the most is room type,
# more specifically if the room type is the whole house or not. The next most important node
# for both trees is neighbourhood_group, if it is Manhattan or not. The model using log_price only
# gives these nodes. The model using the given price variable includes a node if the room is the
# whole house and is in manhattan it looks at availability.
```

```r
# We'll use cross-validation to find the best tree size for both models.
cvTreeR = cv.tree(regMod)
cvTreeR
```

```
## $size
## [1] 5 4 3 2 1
##
## $dev
## [1] 364837133 370691820 377276194 395309429 508015797
##
## $k
## [1]      -Inf   5878954   7102239  18048446 112687423
##
## $method
## [1] "deviance"
##
## attr(,"class")
## [1] "prune"         "tree.sequence"
```

```r
bestIdx = which.min(cvTreeR$dev)
cvTreeR$size[bestIdx]
```

```
## [1] 5
```

```r
# Best size is 5

cvTreeL = cv.tree(logMod)
cvTreeL
```

```
## $size
## [1] 4 3 2 1
##
## $dev
## [1]  8434.909  8910.014  9468.515 15947.743
##
## $k
## [1]      -Inf  475.8630  558.8432 6479.1537
##
## $method
## [1] "deviance"
##
## attr(,"class")
## [1] "prune"         "tree.sequence"
```

```r
bestIdx = which.min(cvTreeL$dev)
cvTreeL$size[bestIdx]
```

```
## [1] 4
```

```r
# Best size is 4

# Now we'll prune the trees using the best size we got from CV and generate predictions
prunedTreeR = prune.tree(regMod, best = 5)
predsTrainR = predict(prunedTreeR)
predsTestR = predict(prunedTreeR, newdata = test)

prunedTreeL = prune.tree(logMod, best = 4)
predsTrainL = predict(prunedTreeL)
predsTestL = predict(prunedTreeL, newdata = test)

# Now we'll calculate MSE for both models
MSE = function(p,t){
  mean((t-p)^2)
}

MSE(predsTrainR, train$price) #9975.12
```

```
## [1] 9990.492
```

```r
MSE(predsTestR,test$price) #9570.264
```

```
## [1] 9516.046
```

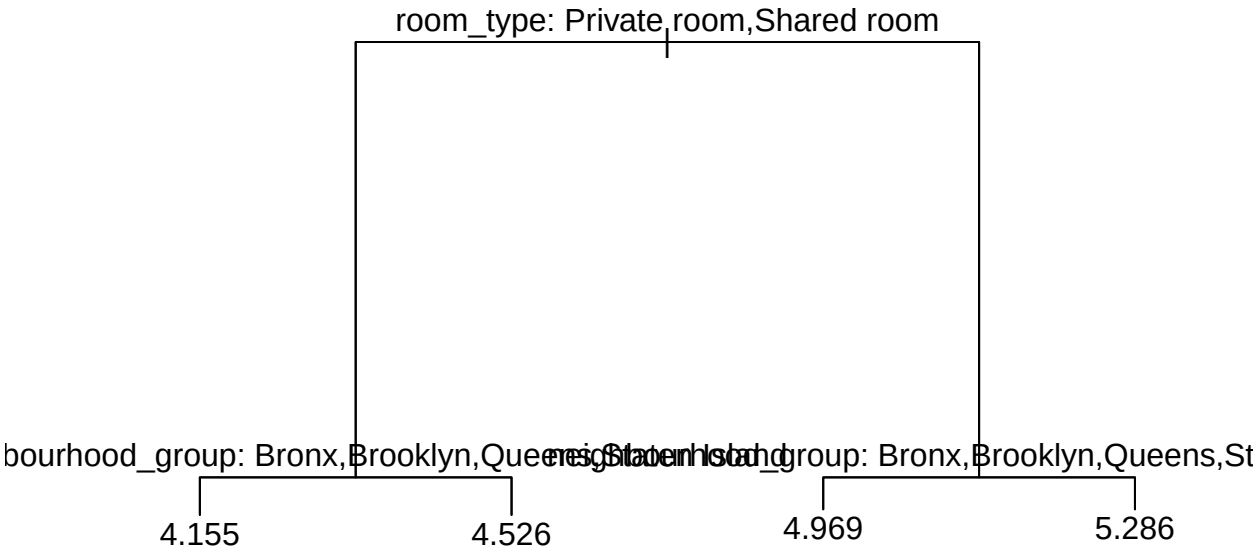
```r
MSE(predsTrainL, train$log_price) #0.231
```

```
## [1] 0.2312964
```

```r
MSE(predsTestL, test$log_price) #0.228
```

```
## [1] 0.2258194
```

```r
# MSE is much lower when we use the log transformation
# we do get a lower MSE in the test set which may be an indication that
# the model is overfitting the data.
plot(prunedTreeL)
text(prunedTreeL, pretty = 0)
```




```
#using random forest model
library(randomForest)
```

```
## randomForest 4.6-14
```

```
## Type rfNews() to see new features/changes/bug fixes.
```

```
##
## Attaching package: 'randomForest'
```

```
## The following object is masked from 'package:ggplot2':
##
##     margin
```

```
set.seed(2019)
#View(nycAB)
#setting mtry to 5 as cross validated best number
#maxnodes to 160 to minimize mse while still optimizing efficiency
#using sample of train to optimize efficiency
#using ntree of 500 to minimize mse
bag_nycAB <- randomForest(log_price~ neighbourhood_group + room_type
                          + availability_365 + log_num_reviews + reviews_per_month_log
                          + minimum_nights_log,
                          data = train_sample,
                          mtry = 5,
                          maxnodes = 160,
                          ntree = 500,
                          importance = TRUE)


bag_nycAB
```

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

```
##  randomForest(formula = log_price ~ neighbourhood_group + room_type +      availability_365 + log_nu
##                Type of random forest: regression
##                      Number of trees: 500
## No. of variables tried at each split: 5
##
##           Mean of squared residuals: 0.201634
##                     % Var explained: 54.08
```

```
#prediction of train data
preds_bag_nycAB <- predict(bag_nycAB, newdata = train)
#prediction of test data
preds_bag_nycAB_test<- predict(bag_nycAB, newdata = test)

#MSE test
MSE(preds_bag_nycAB_test, test$log_price)
```

```
## [1] 0.1956296
```

```
#.1951798

#MSE train
MSE(preds_bag_nycAB, train$log_price)
```

```
## [1] 0.1945877
```

```
#.1948749

#Random forest model has lowest error, so we will use this one.
```