Team Project

```
#load the data
airbnb <- read.csv("train.csv")
```

First try to pre-process the airbnb dataset

```
library(tidyverse)
```

```
## Warning: package 'tidyverse' was built under R version 3.5.2
```

```
## -- Attaching packages ---------------------------------------------------------- tidyverse 1.2.1 --
```

```
## v ggplot2 3.1.0       v purrr   0.3.0
## v tibble  2.0.1       v dplyr   0.8.0.1
## v tidyr   0.8.3       v stringr 1.4.0
## v readr   1.3.1       v forcats 0.3.0
```

```
## Warning: package 'ggplot2' was built under R version 3.5.2
```

```
## Warning: package 'tibble' was built under R version 3.5.2
```

```
## Warning: package 'tidyr' was built under R version 3.5.2
```

```
## Warning: package 'readr' was built under R version 3.5.2
```

```
## Warning: package 'purrr' was built under R version 3.5.2
```

```
## Warning: package 'dplyr' was built under R version 3.5.2
```

```
## Warning: package 'stringr' was built under R version 3.5.2
```

```
## Warning: package 'forcats' was built under R version 3.5.2
```

```
## -- Conflicts ------------------------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
#The id variable is certainly irrelevant
airbnb <- select(airbnb,-id)
#We may need both the original price and the log price variables
airbnb <- mutate(airbnb,price=exp(log_price))
airbnb <- select(airbnb,price,everything())
#For the property type, we may keep those factors with 700+ (count more than 1% of the total observatio
summary(airbnb$property_type)
```

```
##          Apartment    Bed & Breakfast               Boat
##              49003                462                 65
##     Boutique hotel           Bungalow              Cabin
##                 69                366                 72
##          Camper/RV    Casa particular             Castle
##                 94                  1                 13
##               Cave             Chalet        Condominium
##                  2                  6               2658
##              Dorm        Earth House        Guest suite
##                142                  4                123
##         Guesthouse             Hostel              House
##                498                 70              16511
##               Hut             In-law             Island
##                 8                 71                  1
```

```
##         Lighthouse               Loft              Other
##                  1               1244                607
##      Parking Space Serviced apartment               Tent
##                  1                 21                 18
##          Timeshare               Tipi          Townhouse
##                 77                  3               1692
##              Train          Treehouse      Vacation home
##                  2                  7                 11
##              Villa               Yurt
##                179                  9
```

```
levels(airbnb$property_type) <-
  c("Apartment","Others","Others","Others","Others","Others","Others","Others","Others","Others","Others
#Instead of treating amenities as a categorical predictor, we can count the number of ameniteis each ai
airbnb$amenities <- as.character(airbnb$amenities)
airbnb$amenities <- str_count(airbnb$amenities,pattern = ",")
airbnb <- mutate(airbnb,amenities=amenities+1)
#Since the Real Bed count for most of bed_type, we may rename the predictor as "real_bed", change the "
summary(airbnb$bed_type)
```

```
##        Airbed         Couch         Futon Pull-out Sofa      Real Bed
##           477           268           753           585         72028
```

```
airbnb <- rename(airbnb, real_bed = bed_type)
levels(airbnb$real_bed) <- c("False","False","False","False","True")
#Change accmmodates variable as numerical
airbnb$accommodates <- as.numeric(airbnb$accommodates)
#Change bathrooms variable as factor, plus combing some levels
airbnb$bathrooms <- as.factor(airbnb$bathrooms)
levels(airbnb$bathrooms) <-
  c("< 1","< 1","1","1.5","2","> 2","> 2","> 2","> 2","> 2","> 2","> 2","> 2","> 2","> 2","> 2")
#Combine "super_strict_30" and "super_strict_60" with "strict" for cancellation_policy
summary(airbnb$cancellation_policy)
```

```
##        flexible        moderate          strict super_strict_30
##           22545           19063           32374             112
## super_strict_60
##              17
```

```
levels(airbnb$cancellation_policy) <- c("flexible","moderate","strict","strict","strict")
#Drop description variable as it is hard to quantify
airbnb <- select(airbnb,-description)
#Drop first_review as it is highly correlated with host_since
airbnb <- select(airbnb,-first_review)
#Drop host_has_profile_pic as most of them do have pictures; delete observations with no photos
summary(airbnb$host_has_profile_pic)
```

```
##           f     t
##   188   226 73697
```

```
airbnb <- airbnb %>% filter(host_has_profile_pic=="t") %>%
  select(-host_has_profile_pic)
#Re-facotor the host_identity_verified vairable
airbnb$host_identity_verified <- factor(airbnb$host_identity_verified)
#Drop host_idenity_verified variable because there are too many NA values
airbnb <- select(airbnb, -host_response_rate)
```

```
#Transform host_since variable to a continuous vairable showing the number of days s/he becoming a host
library(lubridate)
```

```
## Warning: package 'lubridate' was built under R version 3.5.2
```

```
##
## Attaching package: 'lubridate'
```

```
## The following object is masked from 'package:base':
##
##     date
```

```
airbnb <- mutate(airbnb,host_since=difftime(Sys.Date(),parse_date_time(as.character(airbnb$host_since),
airbnb$host_since <- as.numeric(airbnb$host_since)
#Drop the last_review variable because there are too many NA data.
airbnb <- select(airbnb,-last_review)
#Change the number of reviews variable as a numerical type.
airbnb$number_of_reviews <- as.numeric(airbnb$number_of_reviews)
#Drop latitude & longitude as they are highly correlated with city vairbale
airbnb <- select(airbnb,-latitude,-longitude)
#Drop name & neighbourhood variable as it is hard to quantify
airbnb <- select(airbnb,-name,-neighbourhood)
#Drop the review_scores_rating variable since there are too many NA values
airbnb <- select(airbnb, -review_scores_rating)
#Drop thumbnail_url as it is irrelevant
airbnb <- select(airbnb,-thumbnail_url)
#Drop zipcode is it is highly correlated with city variable
airbnb <- select(airbnb,-zipcode)
#Change the bedroom variable into a factor with 5 levels.
airbnb$bedrooms <- as.factor(airbnb$bedrooms)
levels(airbnb$bedrooms) <- c("0","1","2","3",">3",">3",">3",">3",">3",">3",">3")
#Change the beds variable into a factor.
airbnb$beds <- as.factor(airbnb$beds)
levels(airbnb$beds) <- c("<=1","<=1","2","3",">3",">3",">3",">3",">3",">3",">3",">3",">3",">3",">3",">3
```

Look back again at the new airbnb dataset

```
#Delete the NA values, if any
airbnb <- na.omit(airbnb)
summary(airbnb)
```

```
##      price          log_price         property_type
##  Min.   :   1.0   Min.   :0.000   Apartment  :48454
##  1st Qu.:  75.0   1st Qu.:4.317   Others     : 2972
##  Median : 111.0   Median :4.710   Condominium: 2638
##  Mean   : 160.4   Mean   :4.782   House      :16383
##  3rd Qu.: 185.0   3rd Qu.:5.220   Loft       : 1228
##  Max.   :1999.0   Max.   :7.600   Townhouse  : 1680
##            room_type       amenities      accommodates   bathrooms
##  Entire home/apt:40955   Min.   : 1.00   Min.   : 1.00   < 1:  404
##  Private room   :30269   1st Qu.:13.00   1st Qu.: 2.00   1  :57624
##  Shared room    : 2131   Median :17.00   Median : 2.00   1.5: 3784
##                          Mean   :17.64   Mean   : 3.16   2  : 7890
##                          3rd Qu.:22.00   3rd Qu.: 4.00   > 2: 3653
##                          Max.   :86.00   Max.   :16.00
##   real_bed     cancellation_policy cleaning_fee       city
```

```
##  False: 2061   flexible:22244      False:19408   Boston : 3448
##  True :71294   moderate:18909      True :53947   Chicago: 3706
##                strict  :32202                    DC     : 5643
##                                                  LA     :22266
##                                                  NYC    :31902
##                                                  SF     : 6390
##  host_identity_verified   host_since   instant_bookable number_of_reviews
##  f:23825                Min.   : 519   f:54078          Min.   :  0.00
##  t:49530                1st Qu.:1171   t:19277          1st Qu.:  1.00
##                         Median :1619                    Median :  6.00
##                         Mean   :1688                    Mean   : 20.93
##                         3rd Qu.:2142                    3rd Qu.: 23.00
##                         Max.   :4021                    Max.   :605.00
##  bedrooms    beds
##  0 : 6664   <=1:44700
##  1 :49271   2  :16587
##  2 :11282   3  : 6410
##  3 : 4290   >3 : 5658
##  >3: 1848
##
```

```
str(airbnb)
```

```
## 'data.frame':    73355 obs. of  17 variables:
##  $ price                 : num  150 169 145 750 115 ...
##  $ log_price             : num  5.01 5.13 4.98 6.62 4.74 ...
##  $ property_type         : Factor w/ 6 levels "Apartment","Others",..: 1 1 1 4 1 1 1 3 4 4 ...
##  $ room_type             : Factor w/ 3 levels "Entire home/apt",..: 1 1 1 1 1 2 1 1 2 2 ...
##  $ amenities             : num  9 15 19 15 12 10 21 26 21 13 ...
##  $ accommodates          : num  3 7 5 4 2 2 3 2 2 2 ...
##  $ bathrooms             : Factor w/ 5 levels "< 1","1","1.5",..: 2 2 2 2 2 2 2 2 2 2 ...
##  $ real_bed              : Factor w/ 2 levels "False","True": 2 2 2 2 2 2 2 2 2 2 ...
##  $ cancellation_policy   : Factor w/ 3 levels "flexible","moderate",..: 3 3 2 1 2 3 2 2 2 2 ...
##  $ cleaning_fee          : Factor w/ 2 levels "False","True": 2 2 2 2 2 2 2 2 2 2 ...
##  $ city                  : Factor w/ 6 levels "Boston","Chicago",..: 5 5 5 6 3 6 4 4 6 4 ...
##  $ host_identity_verified: Factor w/ 2 levels "f","t": 2 1 2 2 2 2 1 2 1 1 ...
##  $ host_since            : num  2537 626 863 1418 1467 ...
##  $ instant_bookable      : Factor w/ 2 levels "f","t": 1 2 2 1 2 2 2 1 1 2 ...
##  $ number_of_reviews     : num  2 6 10 0 4 3 15 9 159 2 ...
##  $ bedrooms              : Factor w/ 5 levels "0","1","2","3",..: 2 4 2 3 1 2 2 2 2 2 ...
##  $ beds                  : Factor w/ 4 levels "<=1","2","3",..: 1 3 3 2 1 1 1 1 1 1 ...
##  - attr(*, "na.action")= 'omit' Named int  34 39 197 298 375 639 1339 1812 2024 2340 ...
##   ..- attr(*, "names")= chr  "34" "39" "197" "298" ...
```

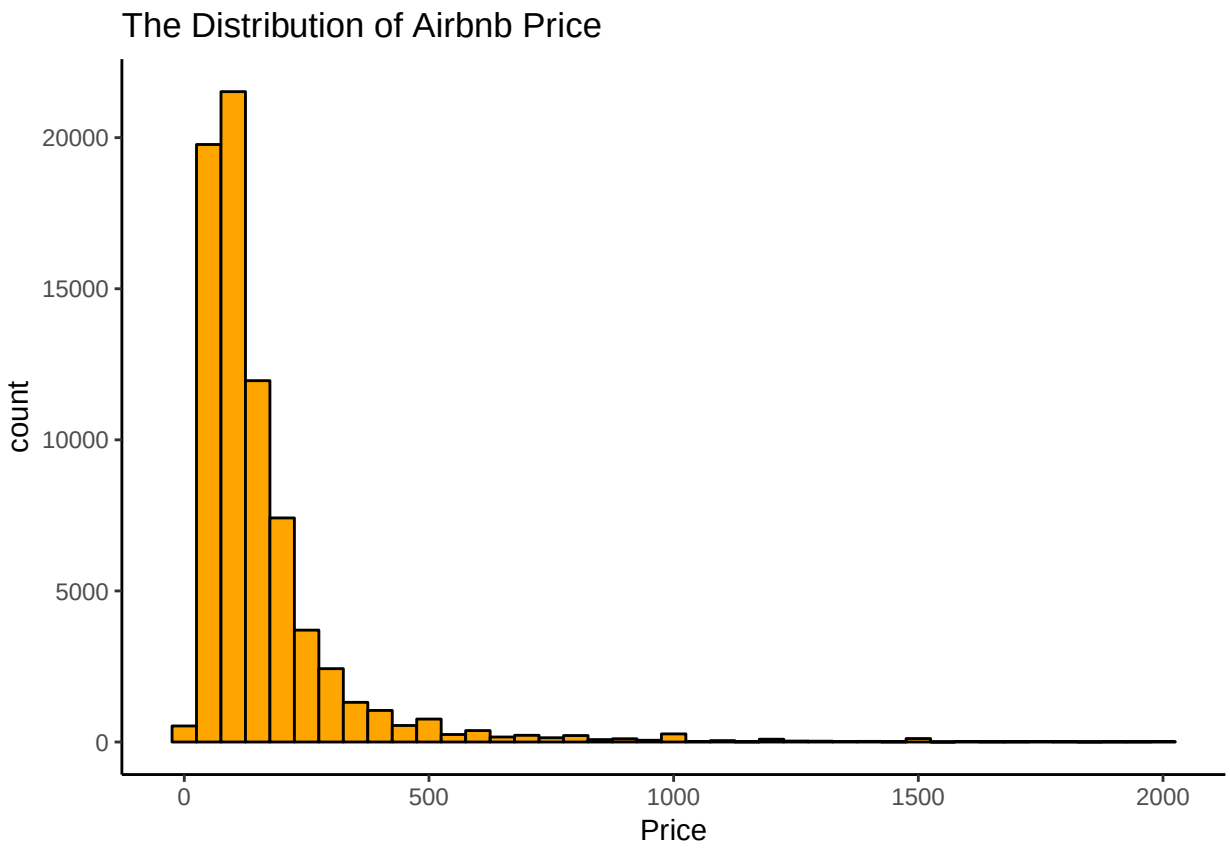
EDA

```
#Look at the price variable
library(corrplot)
```

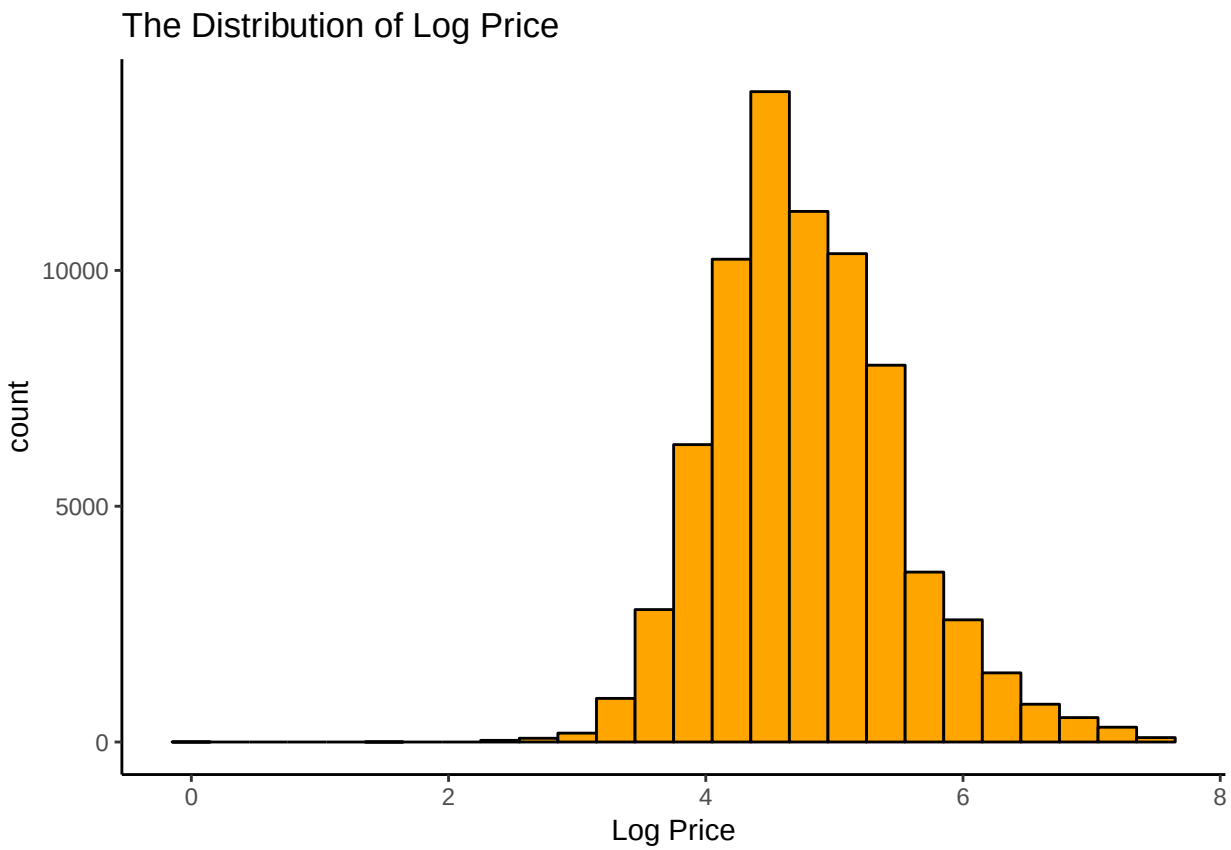
```
## Warning: package 'corrplot' was built under R version 3.5.2
```

```
## corrplot 0.84 loaded
```

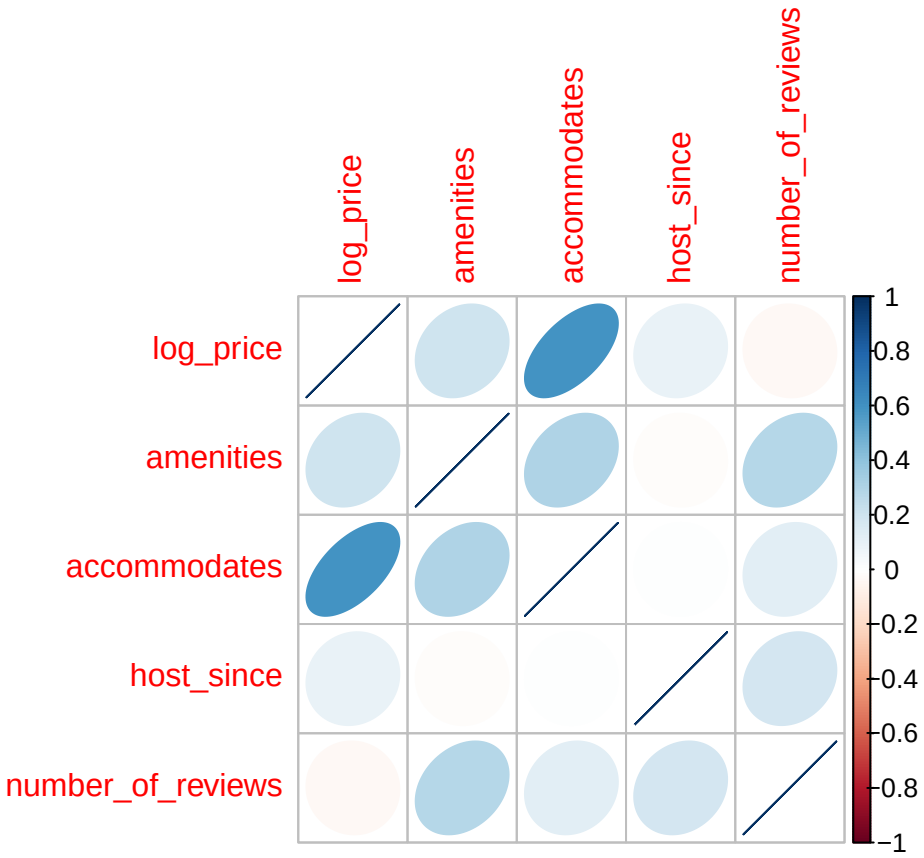
```
theme_set(theme_classic())
ggplot(data = airbnb, mapping = aes(price)) +
  geom_histogram(binwidth = 50, fill = "orange", col = "black") +
  labs(title = "The Distribution of Airbnb Price", x = "Price")
```



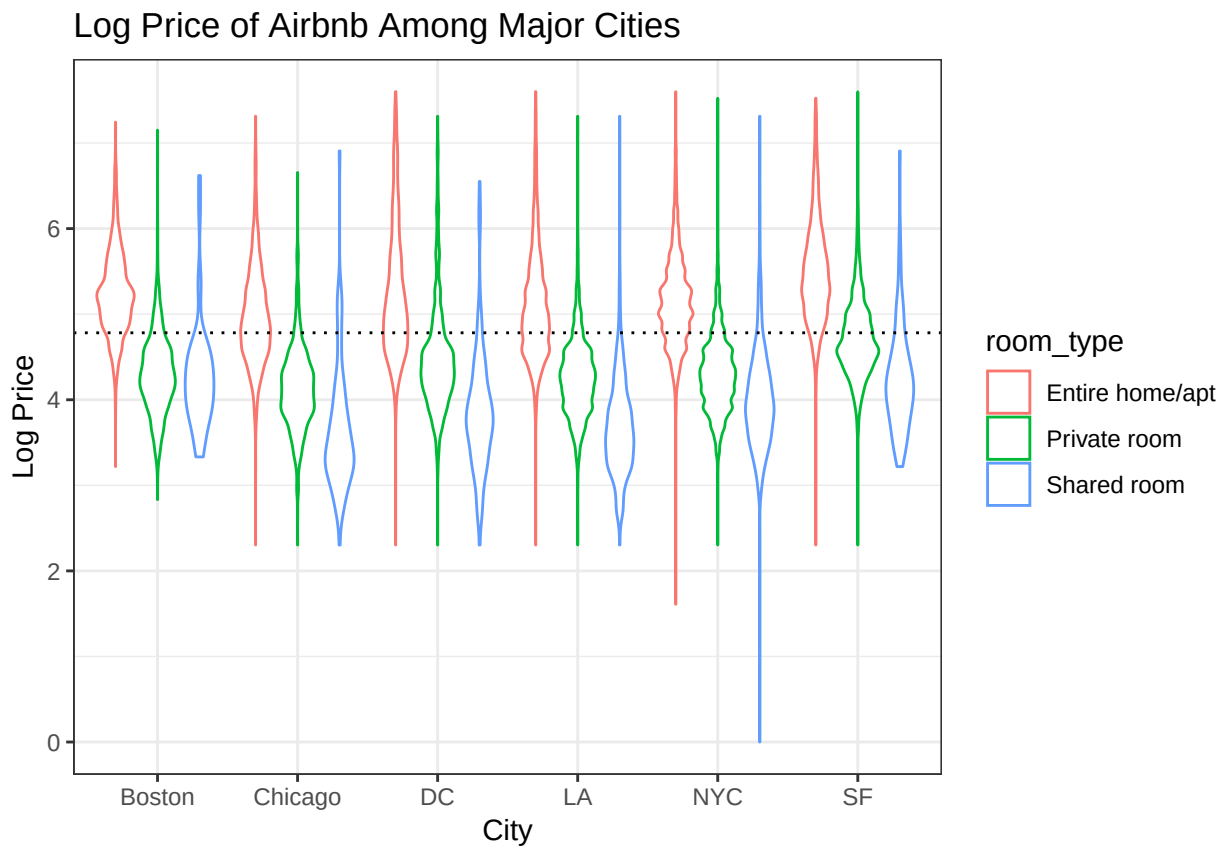

```
#We notice that it is strongly right-skewed, so the log transformation is suggested.
ggplot(data = airbnb, mapping = aes(log_price)) +
  geom_histogram(binwidth = 0.3, fill = "orange", col = "black") +
  labs(title = "The Distribution of Log Price", x="Log Price")
```




```
#We may further explore some relationships among variables
airbnb.cor <- cor(airbnb[,c(2,5,6,13,15)], method = c("spearman"))
corrplot(airbnb.cor, method ="ellipse")
```

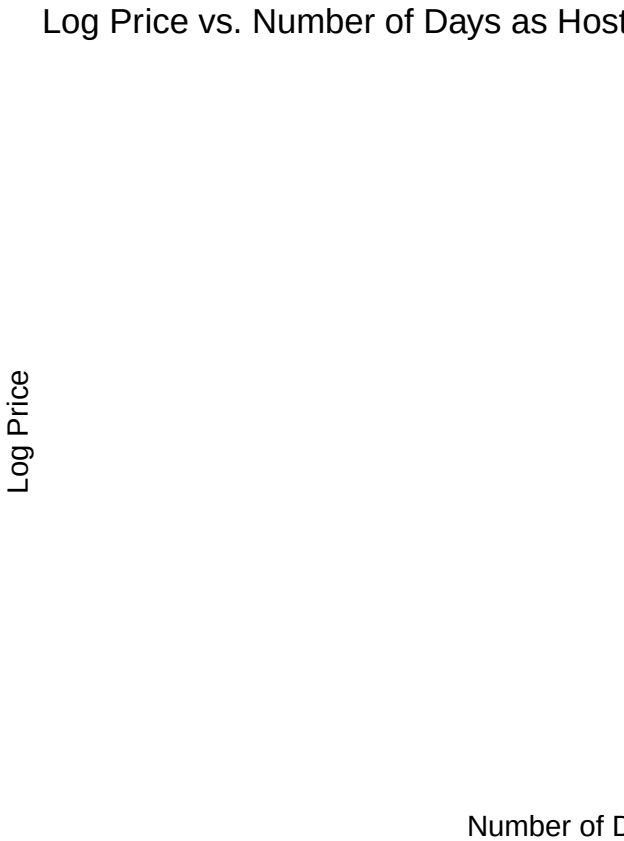



```
theme_set(theme_bw())
ggplot(data = airbnb, mapping = aes(x = city, y = log_price)) +
  geom_violin(aes(color = room_type)) +
  geom_hline(aes(yintercept = mean(log_price)), linetype = "dotted") +
  labs(title="Log Price of Airbnb Among Major Cities",
       x="City",
       y="Log Price")
```




```
theme_set(theme_classic())
ggplot(airbnb, aes(host_since, log_price)) +
  geom_hex() +
  scale_fill_distiller(palette ="RdBu", direction = -1) +
  labs(title="Log Price vs. Number of Days as Hosts",
       x="Number of Days as Hosts",
       y="Log Price") +
  theme(legend.title = element_blank())
```

```
## Warning: Computation failed in `stat_binhex()`:
## Package `hexbin` required for `stat_binhex`.
## Please install and try again.
```




```
#We notice there is outlier at the left bottom corner, and we can drop it.
summary(airbnb$log_price==1)
```

```
##    Mode   FALSE
## logical   73355
```

```
airbnb <- filter(airbnb, log_price > 0)
```

From now on, we need to split the dataset into training set and test set in order to fit the model and make predictions.

```
#Split the data by random
set.seed(1)
n <- nrow(airbnb)
shuffle <- cut(sample.int(n), breaks = c(0, quantile(1:n, 0.75), n),
               labels = c("train", "test"))
train <- split(airbnb, shuffle)$train
test <- split(airbnb, shuffle)$test
```

Model Selection

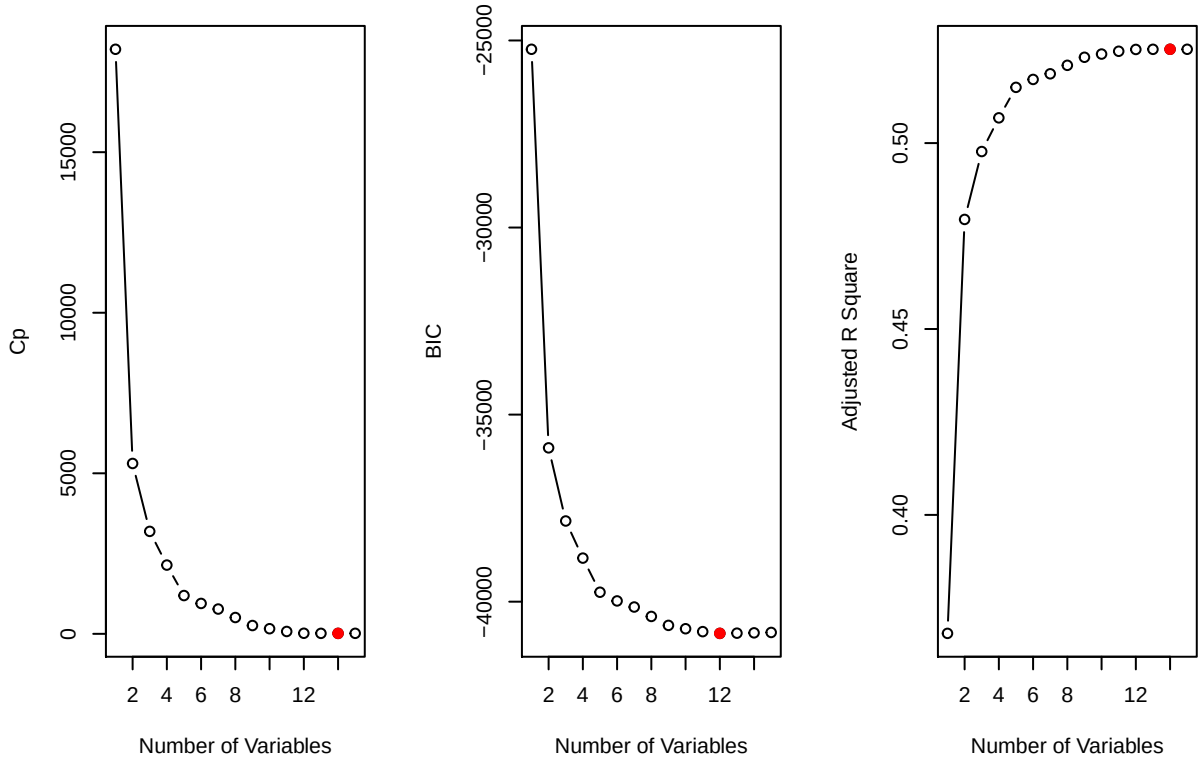
```
#Since there are too may categorical predictors, we are unable to calculate their correlations with the
#But we can try forward selection
#Get a design Matrix
X_design <- matrix(c(train$property_type,train$room_type,train$amenities,train$accommodates,train$bathr
#Run model selection and order the predictors based on importance
library(leaps)
```

```
## Warning: package 'leaps' was built under R version 3.5.2
```

```r
foward <- regsubsets(X_design, train$log_price, nvmax = 15, method = "forward")
foward_sum <- summary(foward)
print(summary(foward))
```

```
## Subset selection object
## 15 Variables  (and intercept)
##   Forced in Forced out
## a     FALSE      FALSE
## b     FALSE      FALSE
## c     FALSE      FALSE
## d     FALSE      FALSE
## e     FALSE      FALSE
## f     FALSE      FALSE
## g     FALSE      FALSE
## h     FALSE      FALSE
## i     FALSE      FALSE
## j     FALSE      FALSE
## k     FALSE      FALSE
## l     FALSE      FALSE
## m     FALSE      FALSE
## n     FALSE      FALSE
## o     FALSE      FALSE
## 1 subsets of each size up to 15
## Selection Algorithm: forward
##          a   b   c   d   e   f   g   h   i   j   k   l   m   n   o
## 1  ( 1 )  " " "*" " " " " " " " " " " " " " " " " " " " " " " " " "
## 2  ( 1 )  " " "*" " " " " " " " " " " " " " " " " " " " " " " " "*" " "
## 3  ( 1 )  " " "*" " " "*" " " " " " " " " " " " " " " " " " " " "*" " "
## 4  ( 1 )  " " "*" " " "*" " " " " " " " " " " " " " " " " " " " "*" " "
## 5  ( 1 )  " " "*" " " "*" "*" " " " " " " " " " " " " " " " " " "*" " "
## 6  ( 1 )  " " "*" " " "*" "*" " " " " " " " " " " " " " " " " " "*" " "
## 7  ( 1 )  " " "*" " " "*" "*" " " " " " " " " " " " " " " " " " "*" " "
## 8  ( 1 )  " " "*" " " "*" "*" " " " " " " " " " " " " " " " " " "*" " "
## 9  ( 1 )  " " "*" "*" "*" "*" " " " " " " " " " " " " " " " " " "*" " "
## 10  ( 1 ) " " "*" "*" "*" "*" " " " " " " " " " " " " " " " " " "*" " "
## 11  ( 1 ) " " "*" "*" "*" "*" " " " " " " " " " " " " " " " " " "*" " "
## 12  ( 1 ) " " "*" "*" "*" "*" " " " " " " " " " " " " " " " " " "*" "*"
## 13  ( 1 ) " " "*" "*" "*" "*" " " " " " " " " " " " " " " " " " "*" "*"
## 14  ( 1 ) "*" "*" "*" "*" "*" " " " " " " " " " " " " " " " " " "*" "*"
## 15  ( 1 ) "*" "*" "*" "*" "*" "*" "*" "*" "*" "*" "*" "*" "*" "*" "*"
```

```r
#Make decision based on Cp, BIC and Adjusted R Square
par(mfrow=c(1,3))
plot(foward_sum$cp, xlab = "Number of Variables", ylab = "Cp",type = "b")
points(which.min(foward_sum$cp), foward_sum$cp[which.min(foward_sum$cp)],
       pch = 21, col = "red", bg = "red")
plot(foward_sum$bic, xlab = "Number of Variables", ylab = "BIC", type = "b")
points(which.min(foward_sum$bic), foward_sum$bic[which.min(foward_sum$bic)],
       pch = 21, col = "red", bg = "red")
plot(foward_sum$adjr2, xlab = "Number of Variables", ylab = "Adjusted R Square",
     type = "b")
points(which.max(foward_sum$adjr2),foward_sum$adjr2[which.max(foward_sum$adjr2)],
       pch = 21, col = "red", bg = "red")
```




```
#The results of backword and exhaustive method appear just the same.
```

Fit Linear Regression Model

```
#We first fit a linear regression model excluding the two least important variables.
mod1 <- lm(data = train, log_price ~ . - price - real_bed)
summary(mod1)
```

```
##
## Call:
## lm(formula = log_price ~ . - price - real_bed, data = train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3.5092 -0.3088 -0.0304  0.2672  3.6048
##
## Coefficients:
##                               Estimate Std. Error  t value Pr(>|t|)
## (Intercept)                  4.522e+00  3.161e-02  143.056  < 2e-16 ***
## property_typeOthers          2.997e-02  1.075e-02    2.789 0.005285 **
## property_typeCondominium     1.369e-01  1.131e-02   12.102  < 2e-16 ***
## property_typeHouse          -3.950e-02  5.788e-03   -6.825 8.89e-12 ***
## property_typeLoft            1.508e-01  1.612e-02    9.357  < 2e-16 ***
## property_typeTownhouse      -2.890e-02  1.388e-02   -2.083 0.037299 *
## room_typePrivate room       -5.898e-01  5.590e-03 -105.518  < 2e-16 ***
## room_typeShared room        -1.030e+00  1.309e-02  -78.624  < 2e-16 ***
## amenities                    4.947e-03  3.269e-04   15.134  < 2e-16 ***
```

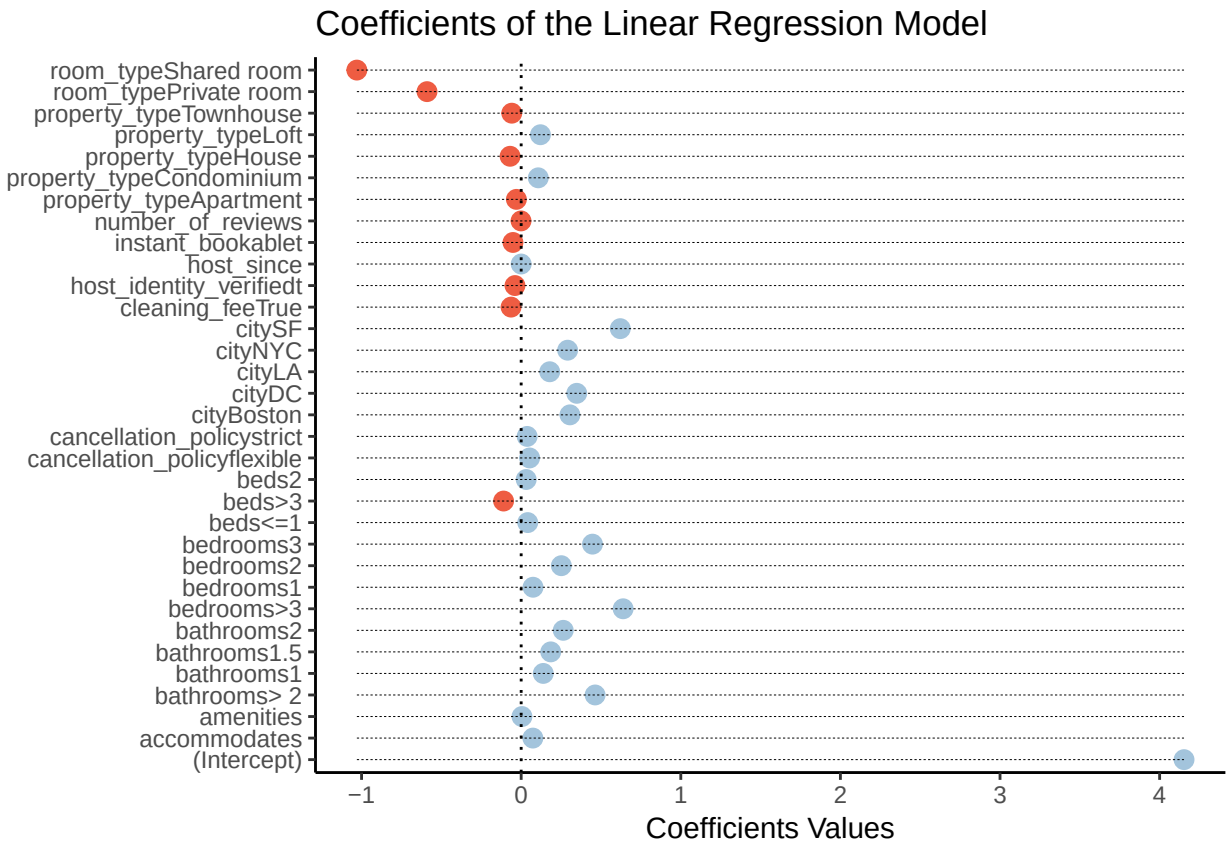
```
## accommodates                  7.330e-02  1.795e-03   40.838  < 2e-16 ***
## bathrooms1                    1.383e-01  2.826e-02    4.894 9.89e-07 ***
## bathrooms1.5                  1.853e-01  2.957e-02    6.266 3.73e-10 ***
## bathrooms2                    2.637e-01  2.908e-02    9.067  < 2e-16 ***
## bathrooms> 2                  4.633e-01  3.041e-02   15.236  < 2e-16 ***
## cancellation_policymoderate -5.270e-02  5.813e-03   -9.067  < 2e-16 ***
## cancellation_policystrict   -1.528e-02  5.379e-03   -2.842 0.004489 **
## cleaning_feeTrue            -6.350e-02  5.121e-03  -12.401  < 2e-16 ***
## cityChicago                 -3.054e-01  1.312e-02  -23.284  < 2e-16 ***
## cityDC                       4.286e-02  1.205e-02    3.557 0.000375 ***
## cityLA                      -1.266e-01  1.031e-02  -12.276  < 2e-16 ***
## cityNYC                     -1.425e-02  1.006e-02   -1.417 0.156537
## citySF                       3.155e-01  1.176e-02   26.826  < 2e-16 ***
## host_identity_verifiedt     -3.885e-02  4.759e-03   -8.163 3.33e-16 ***
## host_since                   5.168e-05  3.454e-06   14.965  < 2e-16 ***
## instant_bookablet           -5.136e-02  4.800e-03  -10.699  < 2e-16 ***
## number_of_reviews           -9.401e-04  5.838e-05  -16.104  < 2e-16 ***
## bedrooms1                    7.429e-02  7.857e-03    9.455  < 2e-16 ***
## bedrooms2                    2.517e-01  1.015e-02   24.803  < 2e-16 ***
## bedrooms3                    4.468e-01  1.451e-02   30.794  < 2e-16 ***
## bedrooms>3                   6.387e-01  2.064e-02   30.943  < 2e-16 ***
## beds2                       -9.519e-03  6.272e-03   -1.518 0.129071
## beds3                       -4.101e-02  1.023e-02   -4.009 6.11e-05 ***
## beds>3                      -1.513e-01  1.331e-02  -11.368  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.4784 on 54982 degrees of freedom
## Multiple R-squared:  0.5569, Adjusted R-squared:  0.5567
## F-statistic:  2160 on 32 and 54982 DF,  p-value: < 2.2e-16
```

```
#It seems that there are severeal insignificant coefficients. First we want to find if changing the ref
train2 <- within(train, property_type <- relevel(property_type, ref = 2))
train2 <- within(train2, cancellation_policy <- relevel(cancellation_policy, ref = 2))
train2 <- within(train2, city <- relevel(city, ref = 2))
train2 <- within(train2, beds <- relevel(beds, ref = 3))
mod2 <- lm(data = train2, log_price ~ . - price - real_bed)
summary(mod2)
```

```
##
## Call:
## lm(formula = log_price ~ . - price - real_bed, data = train2)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3.5092 -0.3088 -0.0304  0.2672  3.6048
##
## Coefficients:
##                               Estimate Std. Error  t value Pr(>|t|)
## (Intercept)                  4.153e+00  3.471e-02  119.643  < 2e-16 ***
## property_typeApartment      -2.997e-02  1.075e-02   -2.789 0.005285 **
## property_typeCondominium     1.069e-01  1.507e-02    7.096 1.30e-12 ***
## property_typeHouse          -6.948e-02  1.125e-02   -6.178 6.53e-10 ***
## property_typeLoft            1.208e-01  1.895e-02    6.376 1.84e-10 ***
## property_typeTownhouse      -5.887e-02  1.703e-02   -3.456 0.000549 ***
```

```
## room_typePrivate room       -5.898e-01  5.590e-03 -105.518  < 2e-16 ***
## room_typeShared room        -1.030e+00  1.309e-02  -78.624  < 2e-16 ***
## amenities                    4.947e-03  3.269e-04   15.134  < 2e-16 ***
## accommodates                 7.330e-02  1.795e-03   40.838  < 2e-16 ***
## bathrooms1                   1.383e-01  2.826e-02    4.894 9.89e-07 ***
## bathrooms1.5                 1.853e-01  2.957e-02    6.266 3.73e-10 ***
## bathrooms2                   2.637e-01  2.908e-02    9.067  < 2e-16 ***
## bathrooms> 2                 4.633e-01  3.041e-02   15.236  < 2e-16 ***
## cancellation_policyflexible  5.270e-02  5.813e-03    9.067  < 2e-16 ***
## cancellation_policystrict    3.742e-02  5.124e-03    7.303 2.86e-13 ***
## cleaning_feeTrue            -6.350e-02  5.121e-03  -12.401  < 2e-16 ***
## cityBoston                   3.054e-01  1.312e-02   23.284  < 2e-16 ***
## cityDC                       3.483e-01  1.177e-02   29.585  < 2e-16 ***
## cityLA                       1.788e-01  9.998e-03   17.886  < 2e-16 ***
## cityNYC                      2.912e-01  9.757e-03   29.842  < 2e-16 ***
## citySF                       6.209e-01  1.149e-02   54.047  < 2e-16 ***
## host_identity_verifiedt     -3.885e-02  4.759e-03   -8.163 3.33e-16 ***
## host_since                   5.168e-05  3.454e-06   14.965  < 2e-16 ***
## instant_bookablet           -5.136e-02  4.800e-03  -10.699  < 2e-16 ***
## number_of_reviews           -9.401e-04  5.838e-05  -16.104  < 2e-16 ***
## bedrooms1                    7.429e-02  7.857e-03    9.455  < 2e-16 ***
## bedrooms2                    2.517e-01  1.015e-02   24.803  < 2e-16 ***
## bedrooms3                    4.468e-01  1.451e-02   30.794  < 2e-16 ***
## bedrooms>3                   6.387e-01  2.064e-02   30.943  < 2e-16 ***
## beds<=1                      4.101e-02  1.023e-02    4.009 6.11e-05 ***
## beds2                        3.149e-02  9.172e-03    3.433 0.000597 ***
## beds>3                      -1.103e-01  1.155e-02   -9.547  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.4784 on 54982 degrees of freedom
## Multiple R-squared:  0.5569, Adjusted R-squared:  0.5567
## F-statistic:  2160 on 32 and 54982 DF,  p-value: < 2.2e-16
```

```r
#Plot of coefficients
coef.mod2 <- as.data.frame(coef(mod2))
coef.mod2[,2] <- (coef(mod2) >= 0)
theme_set(theme_classic())
ggplot(data = coef.mod2, aes(x = rownames(coef.mod2), y = coef(mod2))) +
  geom_point(aes(color = V2), size = 3) +
  scale_color_manual(values = c("tomato2", "#a3c4dc"), guide = FALSE) +
  geom_hline(yintercept = 0, linetype = "dotted") +
  geom_segment(aes(x = rownames(coef.mod2), xend = rownames(coef.mod2),
                   y = min(coef(mod2)), yend = max(coef(mod2))),
               linetype="dashed", size=0.1) +
  labs(title = "Coefficients of the Linear Regression Model",
       y = "Coefficients Values") +
  theme(axis.title.y = element_blank()) +
  coord_flip()
```




```
#We can also make an anova F test to see whether all levels are significant for the property type
mod3 <- lm(data = train2, log_price ~ . - price - beds - real_bed - property_type)
anova(mod2, mod3)
```

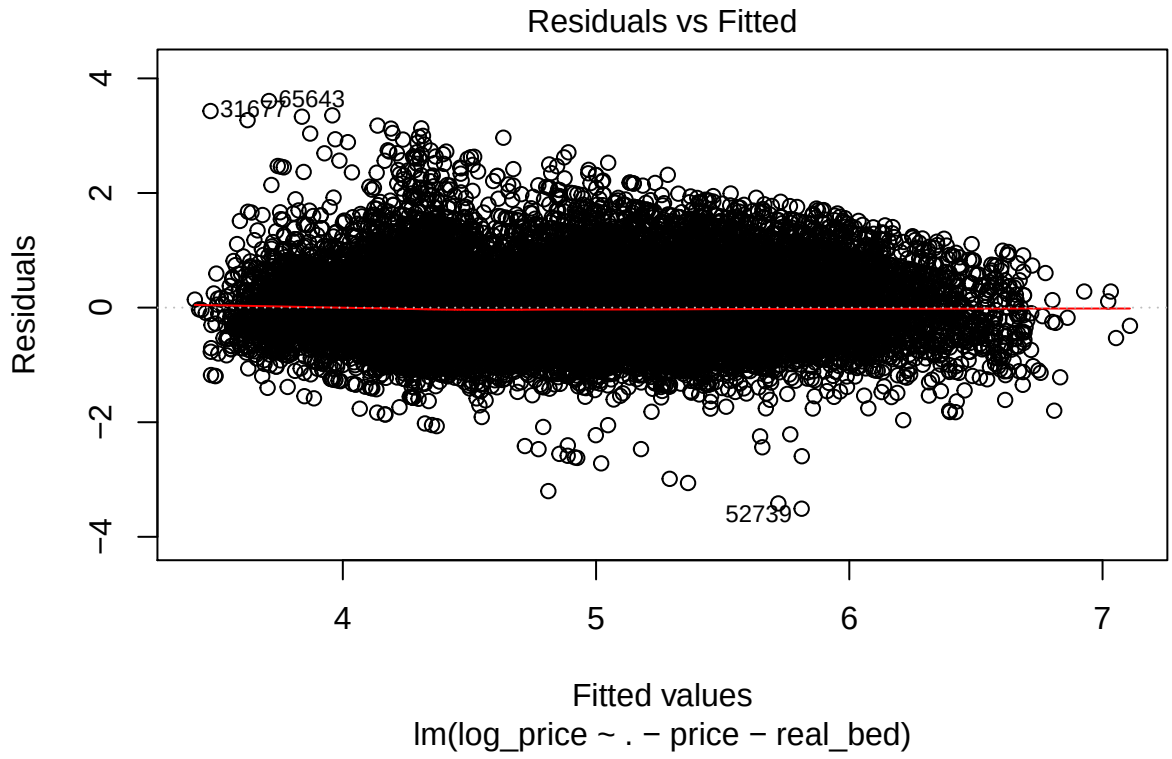
```
## Analysis of Variance Table
##
## Model 1: log_price ~ (price + property_type + room_type + amenities +
##     accommodates + bathrooms + real_bed + cancellation_policy +
##     cleaning_fee + city + host_identity_verified + host_since +
##     instant_bookable + number_of_reviews + bedrooms + beds) -
##     price - real_bed
## Model 2: log_price ~ (price + property_type + room_type + amenities +
##     accommodates + bathrooms + real_bed + cancellation_policy +
##     cleaning_fee + city + host_identity_verified + host_since +
##     instant_bookable + number_of_reviews + bedrooms + beds) -
##     price - beds - real_bed - property_type
##   Res.Df   RSS Df Sum of Sq      F    Pr(>F)
## 1  54982 12583
## 2  54990 12694 -8   -110.89 60.569 < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
#We may further decide whether to include interactions terms and higher level components.
#We now calculate the test error using the test dataset. Remember that we also have to change reference
pred1 <- predict(mod2, test)
mean((pred1 - test$log_price)^2)
```

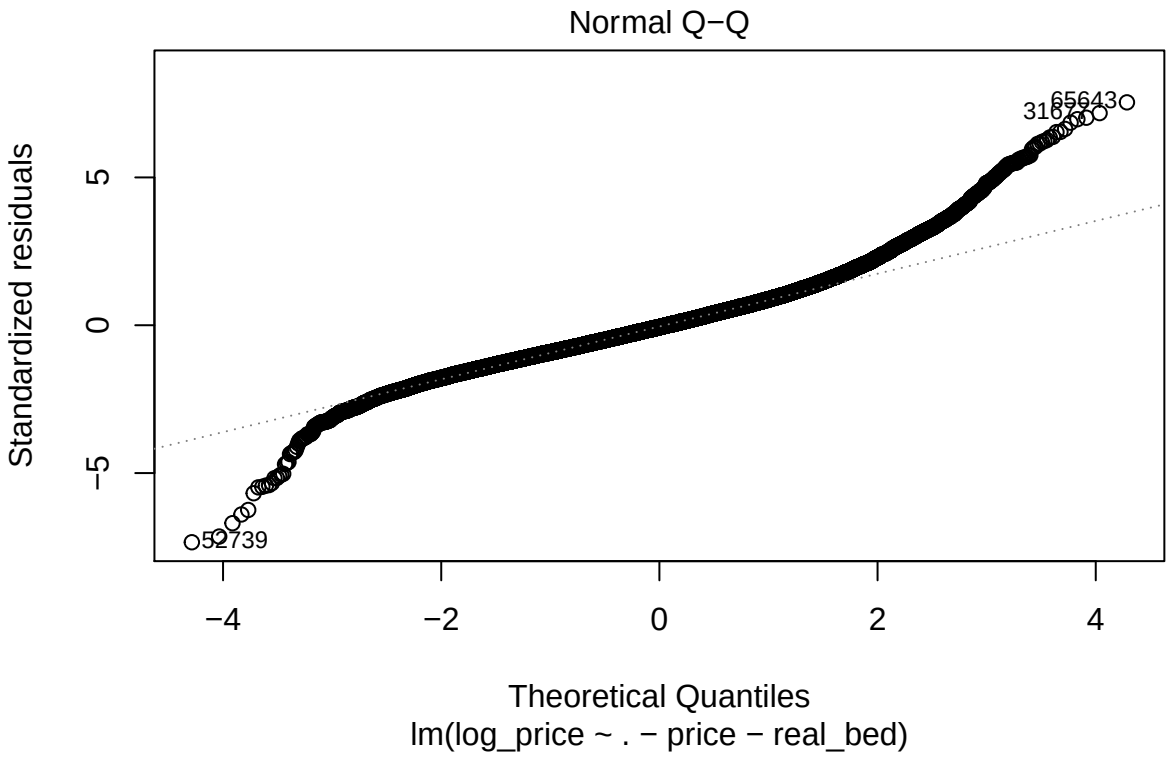
```
## [1] 0.2263378
```

Model diagnoses

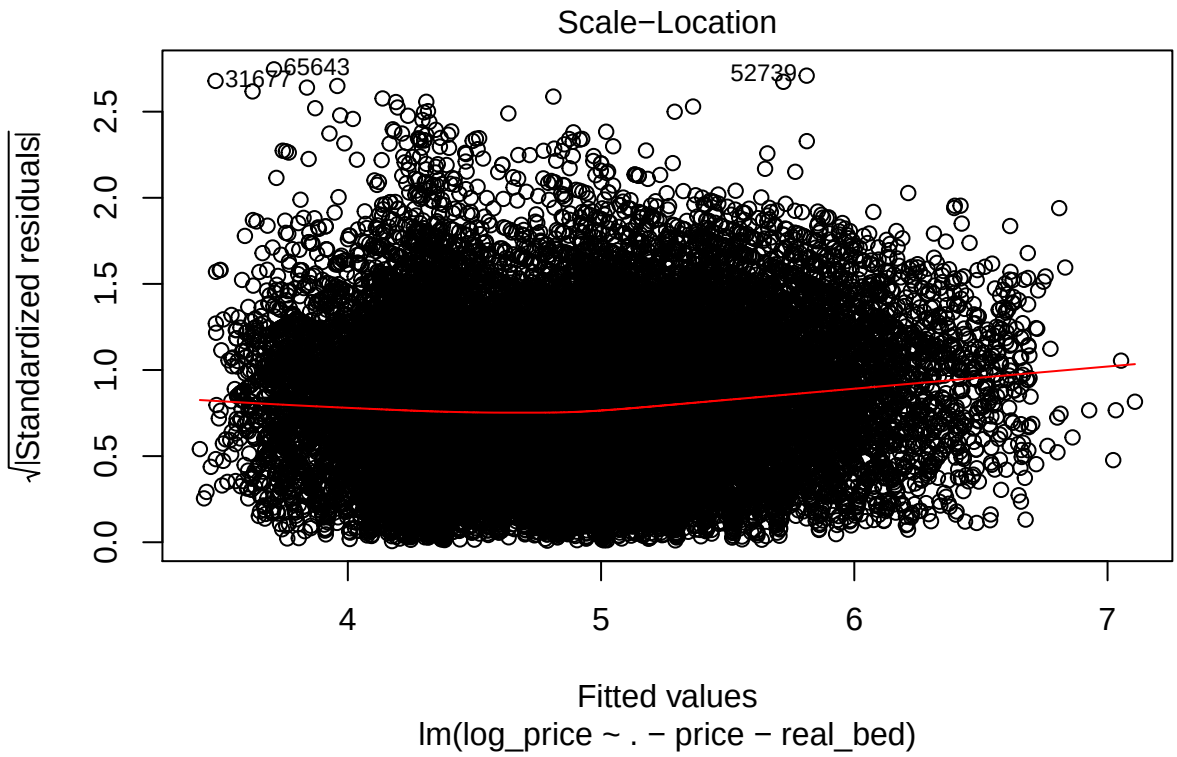
```
plot(mod2, which = 1)
```




```
plot(mod2, which = 2)
```




```
plot(mod2, which = 3)
```



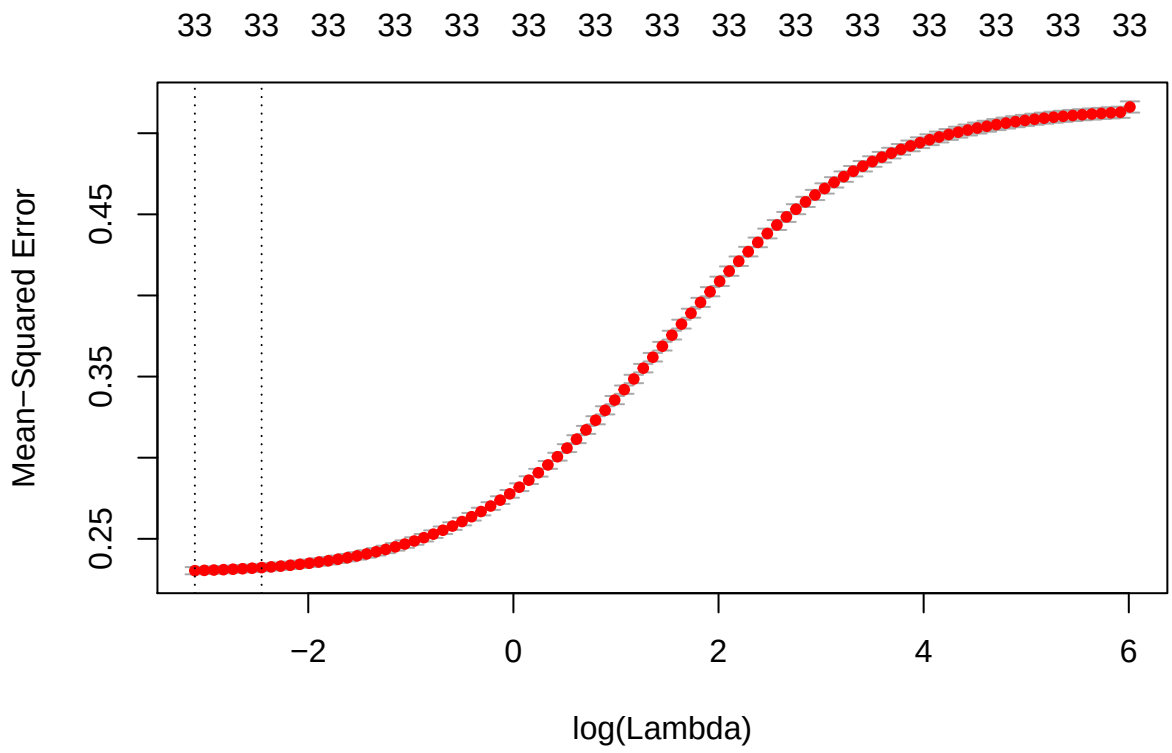

```
#We find that the normality assumption is violated. Residuals follow a long tail distribution.
```

Fit Ridge Regression Models

```
library(glmnet)
```

```
## Warning: package 'glmnet' was built under R version 3.5.2
## Loading required package: Matrix
##
## Attaching package: 'Matrix'
## The following object is masked from 'package:tidyr':
##
##     expand
## Loading required package: foreach
## Warning: package 'foreach' was built under R version 3.5.2
##
## Attaching package: 'foreach'
## The following objects are masked from 'package:purrr':
##
##     accumulate, when
## Loaded glmnet 2.0-16
```

```r
X_train <- model.matrix(log_price ~ . - price, data = train)[,-1]
X_test <- model.matrix(log_price ~ . - price, data = test)[,-1]
#Find best lambda value using 10-fold cross-validation
cv.ridge <- cv.glmnet(X_train, train$log_price, alpha = 0)
plot(cv.ridge)
```

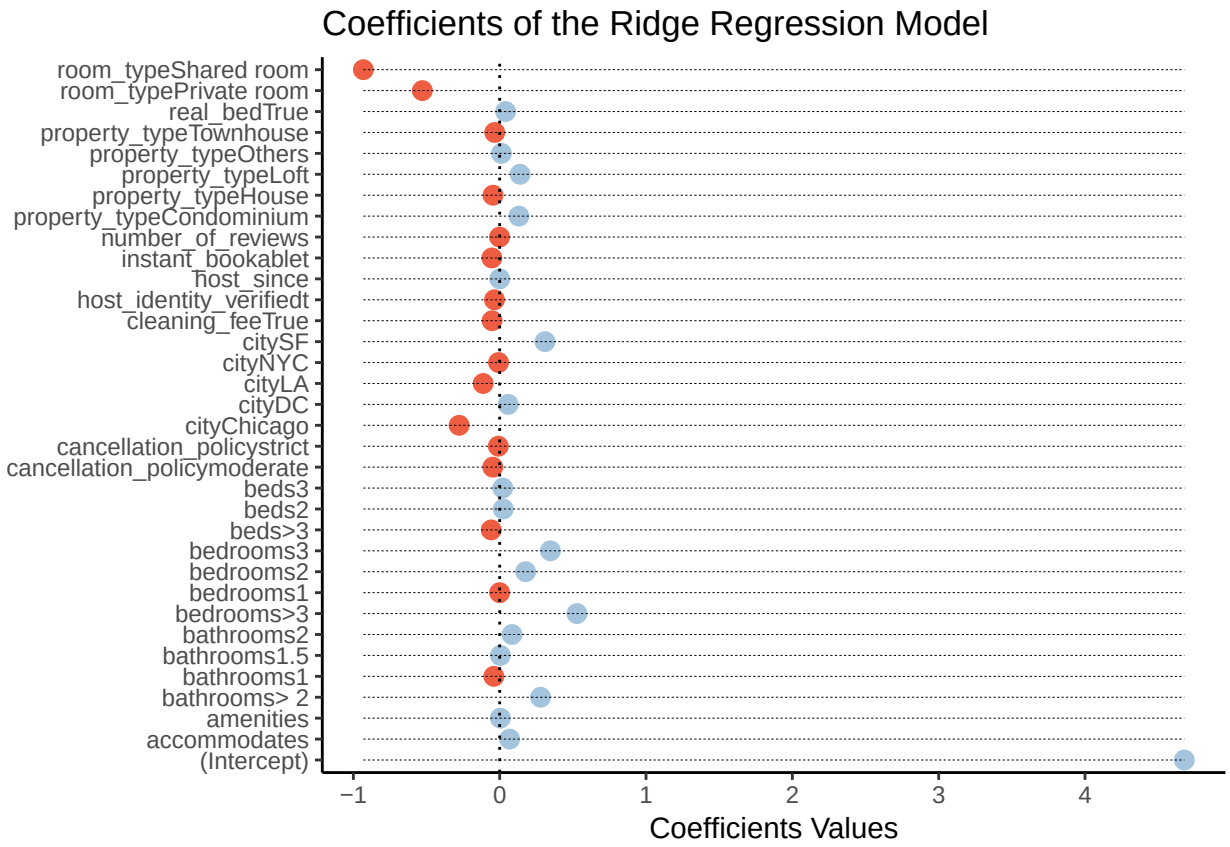



```r
#Using the optimal lambda to fit the model
lmd.ridge <- cv.ridge$lambda.min
mod4 <- glmnet(X_train, train$log_price, alpha = 0, lambda = lmd.ridge)
coef(mod4)
```

```
## 34 x 1 sparse Matrix of class "dgCMatrix"
##                                       s0
## (Intercept)                 4.678775e+00
## property_typeOthers         1.228998e-02
## property_typeCondominium    1.316773e-01
## property_typeHouse         -4.562468e-02
## property_typeLoft           1.394197e-01
## property_typeTownhouse     -3.381831e-02
## room_typePrivate room      -5.285234e-01
## room_typeShared room       -9.316475e-01
## amenities                   4.836473e-03
## accommodates                6.817182e-02
## bathrooms1                 -4.016792e-02
## bathrooms1.5                4.750537e-03
## bathrooms2                  8.432116e-02
```

```
## bathrooms> 2                 2.795913e-01
## real_bedTrue                 4.053648e-02
## cancellation_policymoderate -4.700423e-02
## cancellation_policystrict   -9.385228e-03
## cleaning_feeTrue            -5.177644e-02
## cityChicago                 -2.772205e-01
## cityDC                       5.843598e-02
## cityLA                      -1.129841e-01
## cityNYC                     -7.829585e-03
## citySF                       3.096626e-01
## host_identity_verifiedt     -3.570205e-02
## host_since                   5.220887e-05
## instant_bookablet           -5.361021e-02
## number_of_reviews           -9.218315e-04
## bedrooms1                   -3.390575e-04
## bedrooms2                    1.771458e-01
## bedrooms3                    3.467197e-01
## bedrooms>3                   5.281490e-01
## beds2                        2.437623e-02
## beds3                        2.126394e-02
## beds>3                      -5.795554e-02
```

```r
#Plot of coefficients
coef.mod4 <- as.data.frame(as.matrix(coef(mod4)))
coef.mod4[,2] <- (as.matrix(coef(mod4)) >= 0)
theme_set(theme_classic())
ggplot(data = coef.mod4, aes(x = rownames(coef.mod4), y = s0)) +
  geom_point(aes(color = V2), size = 3) +
  scale_color_manual(values = c("tomato2", "#a3c4dc"), guide = FALSE) +
  geom_hline(yintercept = 0, linetype = "dotted") +
  geom_segment(aes(x = rownames(coef.mod4), xend = rownames(coef.mod4),
                   y = min(s0), yend = max(s0)),
               linetype="dashed", size=0.1) +
  labs(title = "Coefficients of the Ridge Regression Model",
       y = "Coefficients Values") +
  theme(axis.title.y = element_blank()) +
  coord_flip()
```

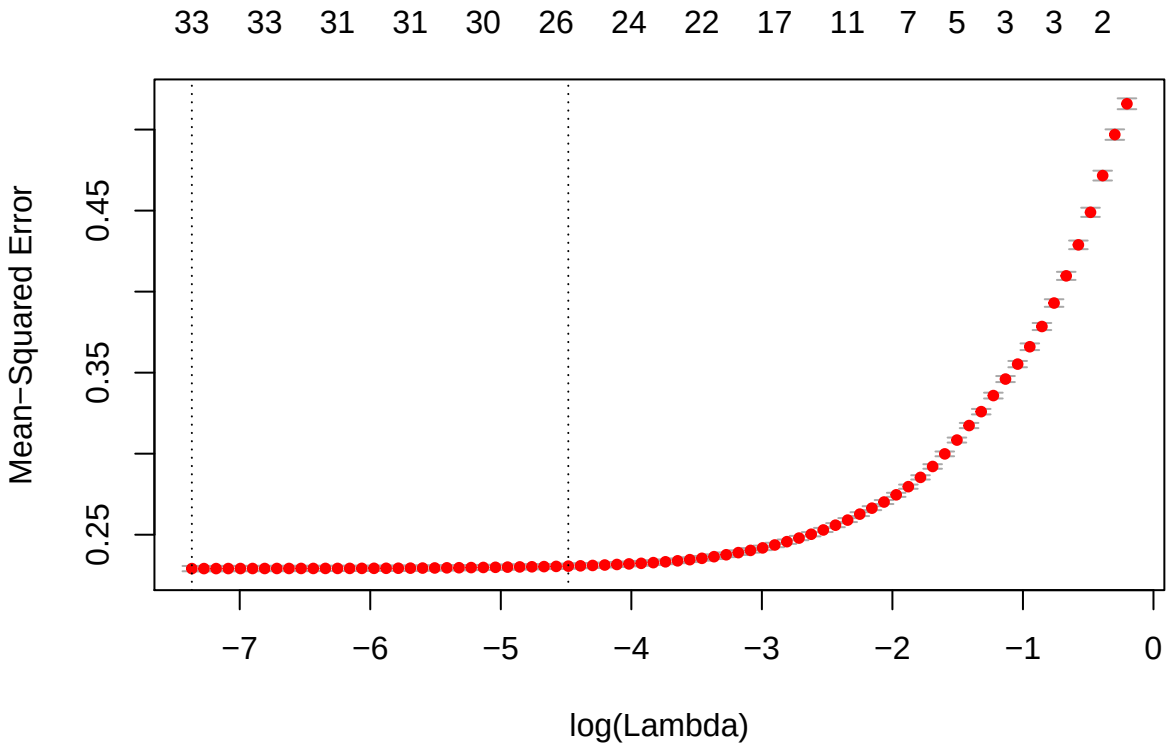



```
#We now calculate the test error using the test dataset.
pred2 <- predict(mod4, X_test)
mean((pred2 - test$log_price)^2)
```

```
## [1] 0.2276919
```

Fit Elastic net regularization Models

```
#Find best lambda value using 10-fold cross-validation
cv.elastic <- cv.glmnet(X_train, train$log_price, alpha = 0.5)
plot(cv.elastic)
```

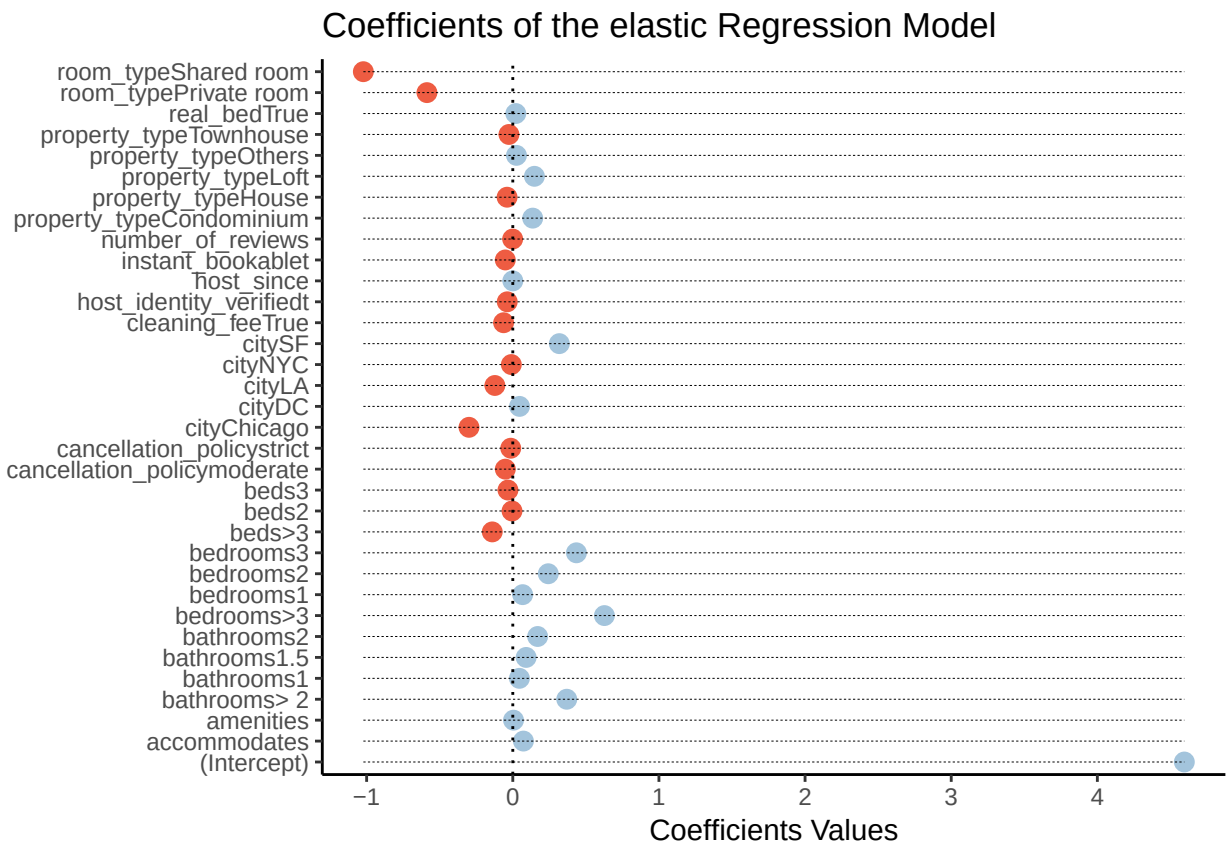



```
#Using the optimal lambda to fit the model
lmd.elastic <- cv.elastic$lambda.min
mod5 <- glmnet(X_train, train$log_price, alpha = 0.5, lambda = lmd.elastic)
coef(mod5) #The only 0-coeffcient variable is cityNYC, which will change if we change the reference lev
```

```
## 34 x 1 sparse Matrix of class "dgCMatrix"
##                                       s0
## (Intercept)                 4.595181e+00
## property_typeOthers         2.512164e-02
## property_typeCondominium    1.363274e-01
## property_typeHouse         -3.915236e-02
## property_typeLoft           1.479903e-01
## property_typeTownhouse     -2.625036e-02
## room_typePrivate room      -5.873976e-01
## room_typeShared room       -1.022562e+00
## amenities                   4.908990e-03
## accommodates                7.269444e-02
## bathrooms1                  4.537431e-02
## bathrooms1.5                9.130829e-02
## bathrooms2                  1.700978e-01
## bathrooms> 2                3.691772e-01
## real_bedTrue                2.055610e-02
## cancellation_policymoderate -5.112513e-02
## cancellation_policystrict   -1.390911e-02
## cleaning_feeTrue            -6.273909e-02
## cityChicago                 -2.995907e-01
```

```
## cityDC                       4.613083e-02
## cityLA                      -1.226693e-01
## cityNYC                     -9.929127e-03
## citySF                       3.182946e-01
## host_identity_verifiedt     -3.799446e-02
## host_since                   5.131295e-05
## instant_bookablet           -5.125467e-02
## number_of_reviews           -9.372336e-04
## bedrooms1                    6.744315e-02
## bedrooms2                    2.427352e-01
## bedrooms3                    4.353132e-01
## bedrooms>3                   6.269146e-01
## beds2                       -5.311423e-03
## beds3                       -3.304456e-02
## beds>3                      -1.404500e-01
```

```r
#Plot of coefficients
coef.mod5 <- as.data.frame(as.matrix(coef(mod5)))
coef.mod5[,2] <- as.numeric((as.matrix(coef(mod5)) >= 0))
coef.mod5[as.matrix(coef(mod5)) == 0,2] = 2
coef.mod5$V2 <- as.factor(coef.mod5$V2)
theme_set(theme_classic())
ggplot(data = coef.mod5, aes(x = rownames(coef.mod5), y = s0)) +
  geom_point(aes(color = V2), size = 3) +
  scale_color_manual(values = c("tomato2", "#a3c4dc", "yellow"), guide = FALSE) +
  geom_hline(yintercept = 0, linetype = "dotted") +
  geom_segment(aes(x = rownames(coef.mod5), xend = rownames(coef.mod5),
                   y = min(s0), yend = max(s0)),
               linetype="dashed", size=0.1) +
  labs(title = "Coefficients of the elastic Regression Model",
       y = "Coefficients Values") +
  theme(axis.title.y = element_blank()) +
  coord_flip()
```




```
#We now calculate the test error using the test dataset.
pred3 <- predict(mod5, X_test)
mean((pred3 - test$log_price)^2)
```

```
## [1] 0.2264376
```

Fit Regression Tree Model

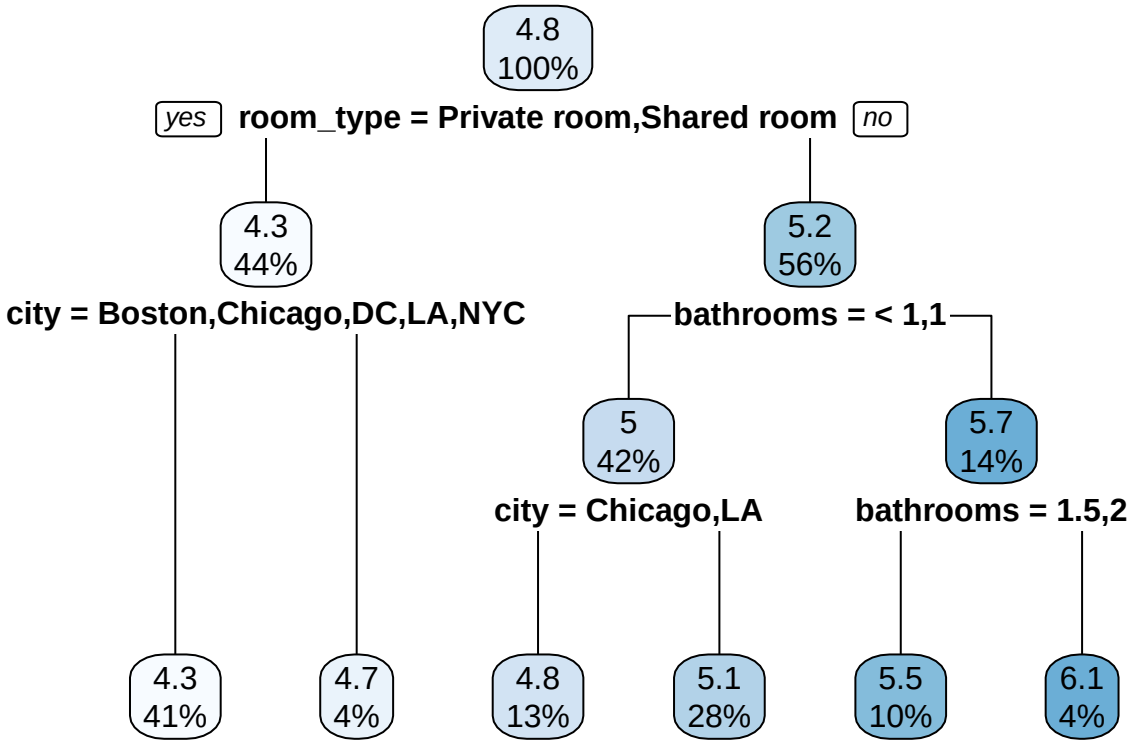
```
library(rpart)
```

```
## Warning: package 'rpart' was built under R version 3.5.2
```

```
library(rpart.plot)
```

```
## Warning: package 'rpart.plot' was built under R version 3.5.2
```

```
mod6 <- rpart(log_price ~ . - price, data = train, method = "anova")
rpart.plot(mod6)
```




```
#We now calculate the test error using the test dataset.
pred4 <- predict(mod6, test)
mean((pred4 - test$log_price)^2)
```

```
## [1] 0.2577679
```

Fit Random Forrest Model

```
library(randomForest)
```

```
## Warning: package 'randomForest' was built under R version 3.5.2
```

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

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

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

```
## The following object is masked from 'package:dplyr':
##
##     combine
```

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

```
set.seed(1)
sample <- train[sample(nrow(train), 10000), ]
mod7 <- randomForest(sample[-c(1,2)],sample$log_price,ntree=200,importance=TRUE)
#We now calculate the test error using the test dataset.
```

```r
pred5 <- predict(mod7, test)
mean((pred5 - test$log_price)^2)
```

```
## [1] 0.2066635
```

```r
#Making importance Plot
rf_importance <- importance(mod7, type = 1)
rf_importance <- data.frame(rf_importance)
rf_importance$x <- row.names(rf_importance)
theme_set(theme_bw())
ggplot(rf_importance, aes(x=x,y=X.IncMSE)) +
  geom_point(size=3)+
  geom_segment(aes(x=x,xend=x,y=0,yend=X.IncMSE)) +
  labs(title="Mean Decrease Accuracy Chart of Random Forest Tree Method",
       x="Variables",
       y="Mean Decrease Accuracy") +
  theme(axis.text.x = element_text(angle=65, vjust=0.6))
```