Predicting AirBnB Rental Rates

Trevor Isaacson, Jonathan Olavarria, Jasmine DeMeyer

12/10/2021

Introduction

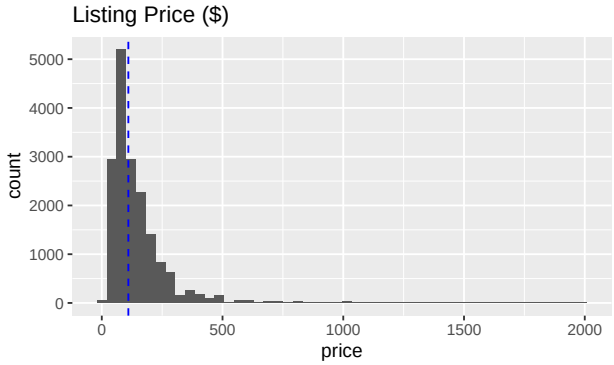
The Data

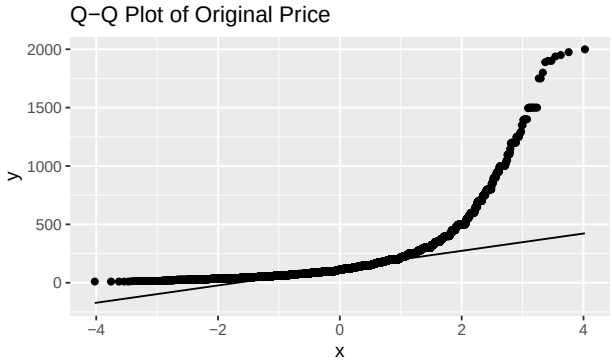
As a company, AirBnB is very open and transparent with the data they collect about their rental properties. They provide data about rental spaces in their system for cities and countries all over the world. Because of this, we were able to find a large dataset on Kaggle with AirBnB listings in major US cities including New York City, Los Angeles, San Francisco and others. The dataset available on Kaggle has over 74,000 entries and was used as a competition a few years ago. For the sake of time and processing, we trimmed our training data to about 17,500 entries and our test data to about 5,000 entries. We did this by taking a random sample of the provided training data. This allowed for easier access and faster processing while maintaining a large amount of data and individual AirBnB listings.

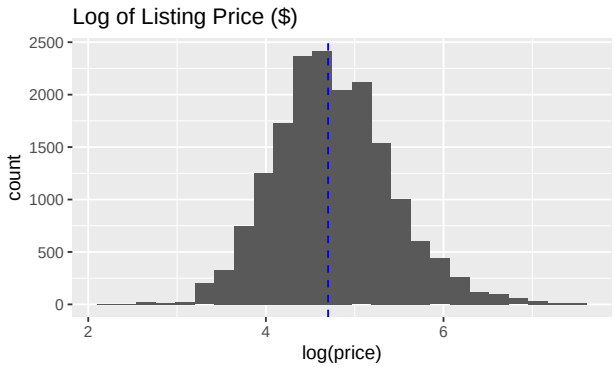
The original dataset contained 30 variables about each listing. Due to high correlations and lack of relevancy, our final dataset consisted of twenty-two variables. Those twenty-two variables can be split into four categories: property, location, host and host reviews.

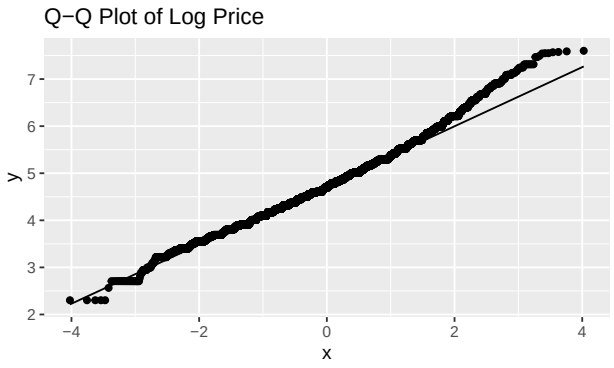
Property includes:

- price: listing price
  - Because the original price data is very heavily skewed, we needed to log transform the prices. As shown in the histogram, we have a very heavy right tail because there are a some listings with very high prices compared to the median price of $110 (blue line). This non-normal shape and distribution is clearly evident in the Q-Q plot. The observations clearly curve away from the line depicting how skewed the distribution is.



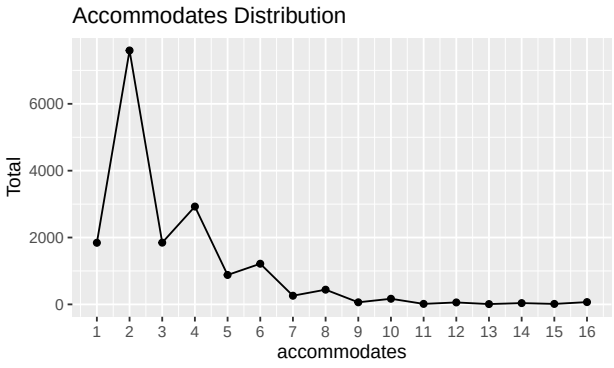


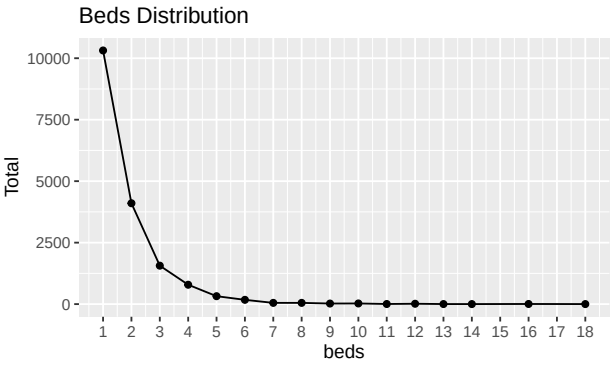


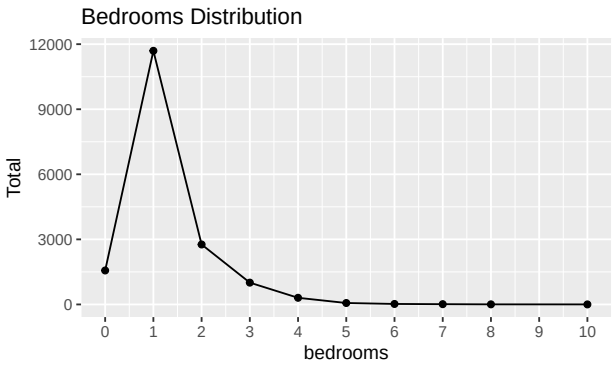


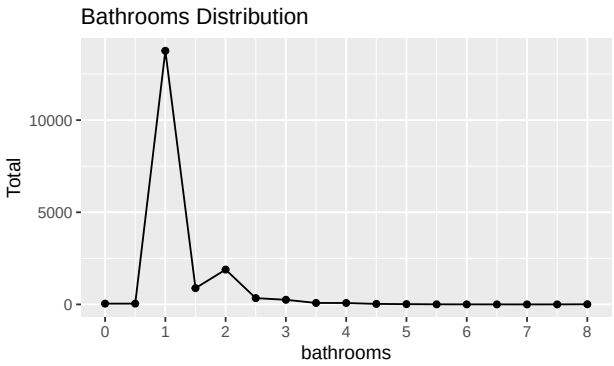
By applying a log transformation, we now have a more normal shaped distribution. Our Q-Q plot shows some evidence of a right tail but this can be expected since the original distribution is very right tailed.

- property_type: defines the type of property listed
  - There are 21 different types ranging from apartments, houses, and condos to boats, cabins, hostels and even castles
- room_type: defines type of rental within the property
  - Includes entire home/apt, private room and shared room
- accommodates: number of people the property can comfortably accommodate
- bedrooms: number of bedrooms within the property
- beds: number of beds within the property
- bed_type: type of bed available
  - This includes a Real Bed, Futon, Pull-out Sofa, Airbed or Couch
  - Only 463 listings have something other than a Real Bed
- bathrooms: number of bathrooms within the property



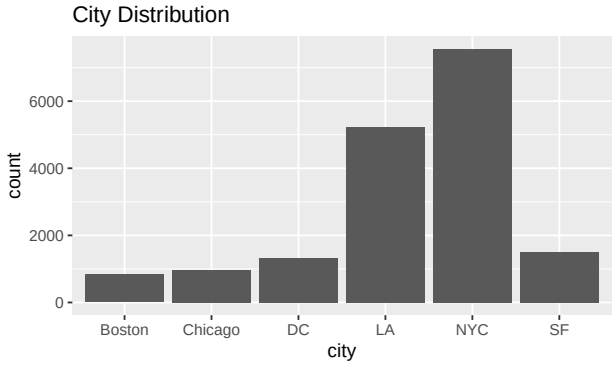


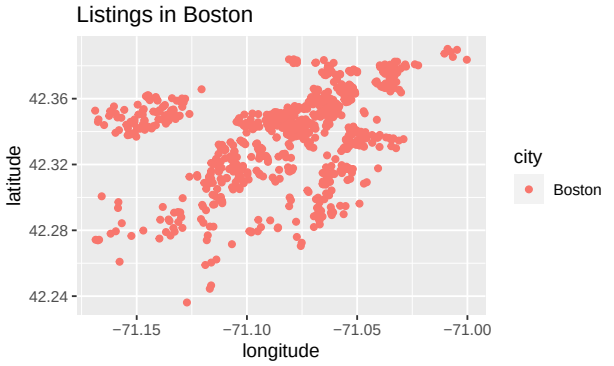




Location includes:

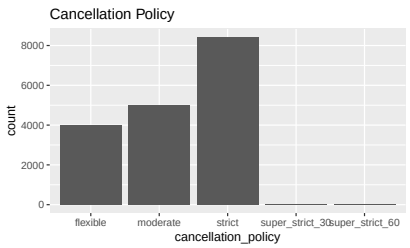
- city: Location of listing
- latitude and longitude: latitude and longitude coordinates of the listing

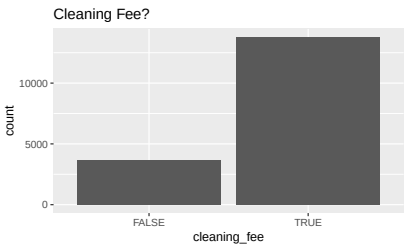


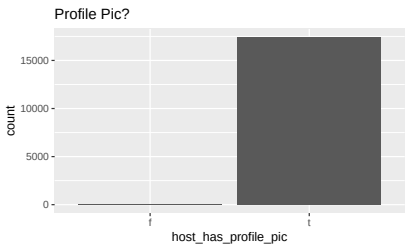


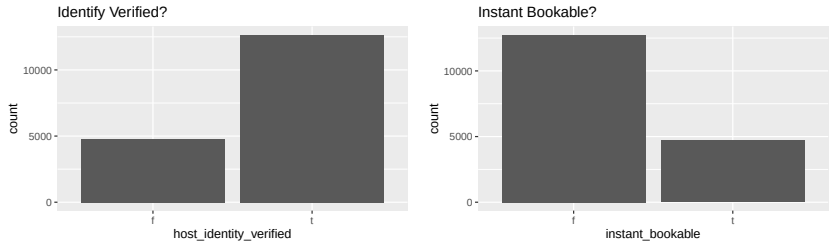
Host includes:

- cancellation_policy: strictness of cancellation policy set by the host
  - Levels include strict, moderate, flexible, super_strict_30 and super_strict_60
- cleaning_fee: TRUE/FALSE determines if host charges a cleaning fee
- host_has_profile_pic: TRUE/FALSE determines if the host has uploaded a picture to their profile
- host_identify_verified: TRUE/FALSE determines if the host's identity has been verified by AirBnB
- instant_bookable: TRUE/FALSE determines if the property can be booked in short notice
- host_response_rate: how often does the host reply to potential clients?









Host Reviews:

- number_of_reviews: Number of reviews the host has received
- review_scores_rating: average review rating for the host and property
- first_review_year: year of the first review
- last_review_year: year of most recent review
- host_since_year: year the property was first listed on AirBnB

As a group, we felt these twenty-two predictors were all relevant and important in helping predict price.

Models

There are have a lot of machine learning methods discussed this past semester and we wanted to incorporate some of our favorites into our research. Thus, we have included linear regression, splines, general additive models, PCR, PLS, trees, bagging, random forests and bagging.

In order to both train and test using the training data file, we needed to split the 17,500 total observations into roughly a 50/50 split. To do this, we took a random sample of 9000 observations and made that the training set. From now on, this random sample with be referred to as the training set. The remaining 7500 observations became our testing set for determining the the performance of each method.

Regression

To begin, we started with a simple multiple linear regression model. We wanted to give ourselves a baseline mean squared error value and because linear regression is the easiest to apply and interpret, we determined this is the best place to start. The model is fit using all twenty-two variables and the training set.

```
## [1] "R^2: 0.632 , Adjusted R^2:  0.63"
```

```
## [1] "MSE of Testing Set:  0.1652"
```

```
## [1] "Leave One Out Cross Validation:  0.1829"
```

```
## [1] "k-fold Cross Validation:  0.3063"
```

As shown in the results above, our linear regression model is able to set a good baseline for future methods with a mean squared error of 0.1652. With our linear regression model, we also applied some cross validation. For Leave One Out Cross Validation, we achieved a mean squared error 0.1829 and applying k-fold cross validation with $k = 10$, we achieved a mean square error of 0.3063. Clearly, Leave One Out Cross Validation performed better than the k-fold cross validation.

Regression Splines/Generalized Additive Models

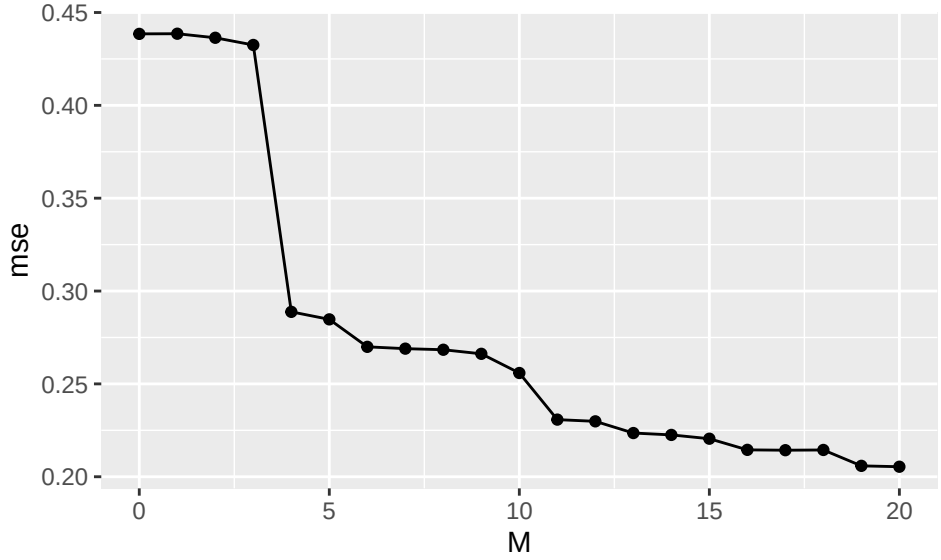
```
##   df accommodates reviews bedrooms bathrooms
## 1  3       0.6077  0.4417   0.5793    0.4940
## 2  4       0.6085  0.4418   0.5943    0.4938
## 3  5       0.6084  0.4422   0.5941    0.4937
```

```
##
## Call:
## lm(formula = price ~ . + bs(accommodates, df = 2) + bs(review_scores_rating,
##     df = 4) + bs(bathrooms, df = 4) + bs(bedrooms, df = 3), data = training)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2.60719 -0.25718 -0.00836  0.23976  2.51417
##
## Coefficients: (4 not defined because of singularities)
##                                    Estimate Std. Error t value Pr(>|t|)
## (Intercept)                       -7.443e+01  4.232e+00 -17.589  < 2e-16 ***
## property_typeBed & Breakfast       1.624e-01  4.571e-02   3.554 0.000382 ***
## property_typeBoat                  2.109e-01  1.072e-01   1.968 0.049067 *
## property_typeBoutique hotel        8.678e-02  1.633e-01   0.531 0.595097
## property_typeBungalow             -1.217e-01  5.554e-02  -2.191 0.028479 *
## property_typeCabin                -3.558e-02  1.209e-01  -0.294 0.768477
## property_typeCamper/RV            -2.702e-02  1.630e-01  -0.166 0.868352
## property_typeCastle                9.343e-01  2.818e-01   3.316 0.000917 ***
## property_typeCondominium           1.043e-01  2.412e-02   4.324 1.55e-05 ***
## property_typeDorm                 -3.425e-01  8.458e-02  -4.049 5.18e-05 ***
## property_typeGuest suite          -1.646e-01  1.111e-01  -1.482 0.138454
## property_typeGuesthouse           -7.264e-02  5.082e-02  -1.429 0.152901
## property_typeHostel               -7.019e-01  1.382e-01  -5.078 3.89e-07 ***
## property_typeHouse                -1.898e-02  1.176e-02  -1.614 0.106494
## property_typeIn-law               -2.268e-01  1.334e-01  -1.700 0.089236 .
## property_typeLoft                  1.257e-01  3.109e-02   4.044 5.30e-05 ***
## property_typeOther                 1.033e-02  4.995e-02   0.207 0.836115
## property_typeServiced apartment    4.807e-01  2.303e-01   2.087 0.036896 *
## property_typeTimeshare             9.722e-01  1.784e-01   5.450 5.18e-08 ***
## property_typeTownhouse            -1.779e-02  2.942e-02  -0.605 0.545466
## property_typeVilla                 2.628e-01  8.015e-02   3.279 0.001048 **
## room_typePrivate room             -5.425e-01  1.179e-02 -46.012  < 2e-16 ***
## room_typeShared room              -9.800e-01  2.967e-02 -33.028  < 2e-16 ***
## accommodates                       6.752e-02  5.492e-03  12.295  < 2e-16 ***
## bathrooms                          7.476e-02  2.319e-02   3.225 0.001266 **
## bed_typeCouch                      6.105e-03  9.075e-02   0.067 0.946367
## bed_typeFuton                     -4.252e-02  7.061e-02  -0.602 0.547025
## bed_typePull-out Sofa              2.805e-03  7.529e-02   0.037 0.970286
## bed_typeReal Bed                   1.285e-02  5.860e-02   0.219 0.826453
## cancellation_policymoderate       -1.429e-03  1.225e-02  -0.117 0.907116
## cancellation_policystrict          3.796e-02  1.153e-02   3.293 0.000995 ***
## cancellation_policysuper_strict_30 3.707e-01  1.272e-01   2.915 0.003567 **
## cancellation_policysuper_strict_60 7.056e-01  2.853e-01   2.473 0.013401 *
## cleaning_feeTRUE                  -1.524e-03  1.110e-02  -0.137 0.890837
## cityChicago                       -1.702e+01  7.889e-01 -21.569  < 2e-16 ***
## cityDC                            -5.538e+00  3.622e-01 -15.290  < 2e-16 ***
```

```
## cityLA                                -4.654e+01  2.321e+00 -20.049  < 2e-16 ***
## cityNYC                               -2.662e+00  1.742e-01 -15.284  < 2e-16 ***
## citySF                                -5.090e+01  2.464e+00 -20.654  < 2e-16 ***
## host_has_profile_pict                 -1.941e-01  8.942e-02  -2.170 0.030016 *
## host_identity_verifiedt                1.265e-02  9.968e-03   1.269 0.204533
## host_response_rate                     4.208e-03  1.269e-02   0.332 0.740247
## instant_bookablet                     -4.100e-02  9.893e-03  -4.144 3.44e-05 ***
## latitude                               1.691e-01  6.319e-02   2.676 0.007460 **
## longitude                             -1.011e+00  4.746e-02 -21.301  < 2e-16 ***
## number_of_reviews                     -3.469e-04  1.244e-04  -2.788 0.005308 **
## review_scores_rating                   1.811e-03  1.174e-03   1.543 0.122827
## bedrooms                              -6.913e-03  3.248e-02  -0.213 0.831474
## beds                                  -5.691e-02  6.515e-03  -8.735  < 2e-16 ***
## first_review_yearLess2014              4.762e-02  1.379e-02   3.452 0.000558 ***
## last_review_yearLess2014               5.568e-02  4.407e-02   1.263 0.206492
## host_since_yearLess2014                3.145e-02  9.775e-03   3.218 0.001297 **
## bs(accommodates, df = 2)1              4.335e-01  5.148e-02   8.421  < 2e-16 ***
## bs(accommodates, df = 2)2              6.536e-02  1.126e-01   0.580 0.561718
## bs(accommodates, df = 2)3                     NA         NA      NA       NA
## bs(review_scores_rating, df = 4)1      3.448e-01  1.580e-01   2.182 0.029110 *
## bs(review_scores_rating, df = 4)2     -4.852e-01  5.589e-02  -8.682  < 2e-16 ***
## bs(review_scores_rating, df = 4)3     -1.114e-02  1.502e-02  -0.742 0.458227
## bs(review_scores_rating, df = 4)4             NA         NA      NA       NA
## bs(bathrooms, df = 4)1                 4.003e-02  8.495e-02   0.471 0.637495
## bs(bathrooms, df = 4)2                -1.108e-02  1.049e-01  -0.106 0.915814
## bs(bathrooms, df = 4)3                 8.212e-01  2.485e-01   3.304 0.000956 ***
## bs(bathrooms, df = 4)4                        NA         NA      NA       NA
## bs(bedrooms, df = 3)1                  1.126e-01  1.085e-01   1.037 0.299674
## bs(bedrooms, df = 3)2                  1.994e+00  3.512e-01   5.677 1.41e-08 ***
## bs(bedrooms, df = 3)3                         NA         NA      NA       NA
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.398 on 8938 degrees of freedom
## Multiple R-squared:  0.6412, Adjusted R-squared:  0.6387
## F-statistic: 261.8 on 61 and 8938 DF,  p-value: < 2.2e-16

## [1] "Test MSE of GAM:  0.1612"
```

PCR and PLS



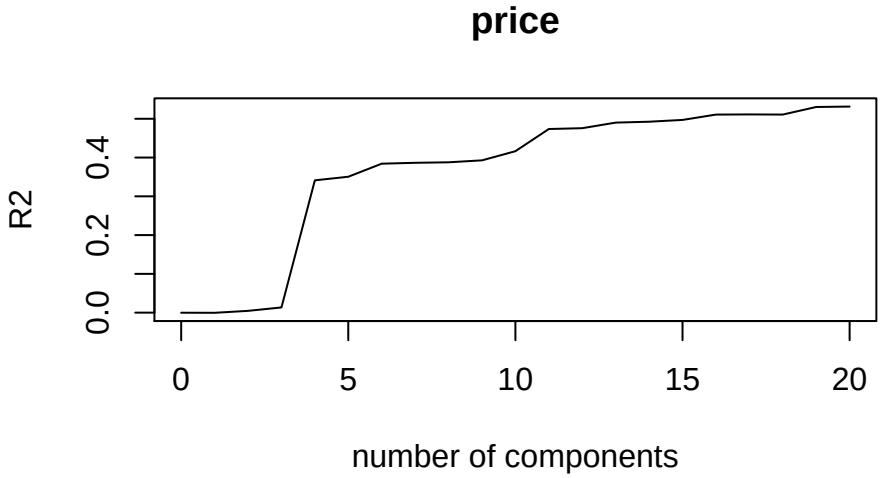

```
## , , model = (Intercept)
##
##          response
## estimate     price
##    CV    0.4384905
##    adjCV 0.4384905
##
## , , model = 1 comps
##
##          response
## estimate     price
##    CV    0.4385671
##    adjCV 0.4385548
##
## , , model = 2 comps
##
##          response
## estimate     price
##    CV    0.4363861
##    adjCV 0.4363712
##
## , , model = 3 comps
##
##          response
## estimate     price
##    CV    0.4325057
##    adjCV 0.4324898
##
## , , model = 4 comps
##
##          response
## estimate     price
##    CV    0.2887998
```

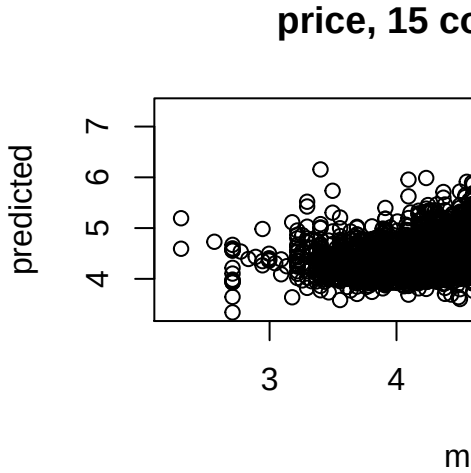
```
##    adjCV 0.2887861
##
## , , model = 5 comps
##
##         response
## estimate     price
##    CV    0.2847489
##    adjCV 0.2847294
##
## , , model = 6 comps
##
##         response
## estimate     price
##    CV    0.2699636
##    adjCV 0.2698675
##
## , , model = 7 comps
##
##         response
## estimate     price
##    CV    0.2689757
##    adjCV 0.2689439
##
## , , model = 8 comps
##
##         response
## estimate     price
##    CV    0.2683957
##    adjCV 0.2683617
##
## , , model = 9 comps
##
##         response
## estimate     price
##    CV    0.2661956
##    adjCV 0.2661671
##
## , , model = 10 comps
##
##         response
## estimate     price
##    CV    0.2558713
##    adjCV 0.2558510
##
## , , model = 11 comps
##
##         response
## estimate     price
##    CV    0.2307664
##    adjCV 0.2299904
##
## , , model = 12 comps
##
##         response
```

```
## estimate     price
##    CV    0.2298372
##    adjCV 0.2297196
##
## , , model = 13 comps
##
##         response
## estimate     price
##    CV    0.2235358
##    adjCV 0.2234473
##
## , , model = 14 comps
##
##         response
## estimate     price
##    CV    0.2225318
##    adjCV 0.2224805
##
## , , model = 15 comps
##
##         response
## estimate     price
##    CV    0.2204851
##    adjCV 0.2204334
##
## , , model = 16 comps
##
##         response
## estimate     price
##    CV    0.2144811
##    adjCV 0.2144175
##
## , , model = 17 comps
##
##         response
## estimate     price
##    CV    0.2142972
##    adjCV 0.2142436
##
## , , model = 18 comps
##
##         response
## estimate     price
##    CV    0.2144246
##    adjCV 0.2143878
##
## , , model = 19 comps
##
##         response
## estimate     price
##    CV    0.2058633
##    adjCV 0.2058005
##
## , , model = 20 comps
```

```
##
##         response
## estimate     price
##    CV    0.2054016
##    adjCV 0.2053409

## Data:    X dimension: 9000 61
##  Y dimension: 9000 1
## Fit method: svdpc
## Number of components considered: 20
##
## VALIDATION: RMSEP
## Cross-validated using 10 random segments.
##        (Intercept)  1 comps  2 comps  3 comps  4 comps  5 comps  6 comps
## CV          0.6622   0.6622   0.6606   0.6577   0.5374   0.5336   0.5196
## adjCV       0.6622   0.6622   0.6606   0.6576   0.5374   0.5336   0.5195
##        7 comps  8 comps  9 comps  10 comps  11 comps  12 comps  13 comps
## CV      0.5186   0.5181   0.5159    0.5058    0.4804    0.4794    0.4728
## adjCV   0.5186   0.5180   0.5159    0.5058    0.4796    0.4793    0.4727
##        14 comps  15 comps  16 comps  17 comps  18 comps  19 comps  20 comps
## CV       0.4717    0.4696    0.4631    0.4629    0.4631    0.4537    0.4532
## adjCV    0.4717    0.4695    0.4631    0.4629    0.4630    0.4537    0.4531
##
## TRAINING: % variance explained
##        1 comps  2 comps  3 comps  4 comps  5 comps  6 comps  7 comps  8 comps
## X      75.4658  96.6542   99.480    99.76    99.85    99.87    99.89    99.90
## price   0.0139   0.5229    1.412    34.18    35.13    38.55    38.78    38.94
##        9 comps  10 comps  11 comps  12 comps  13 comps  14 comps  15 comps
## X        99.92     99.93     99.94     99.95     99.96     99.96     99.97
## price    39.43     41.80     47.56     47.81     49.24     49.48     49.92
##        16 comps  17 comps  18 comps  19 comps  20 comps
## X         99.97     99.98     99.98     99.99     99.99
## price     51.31     51.35     51.35     53.29     53.41
```

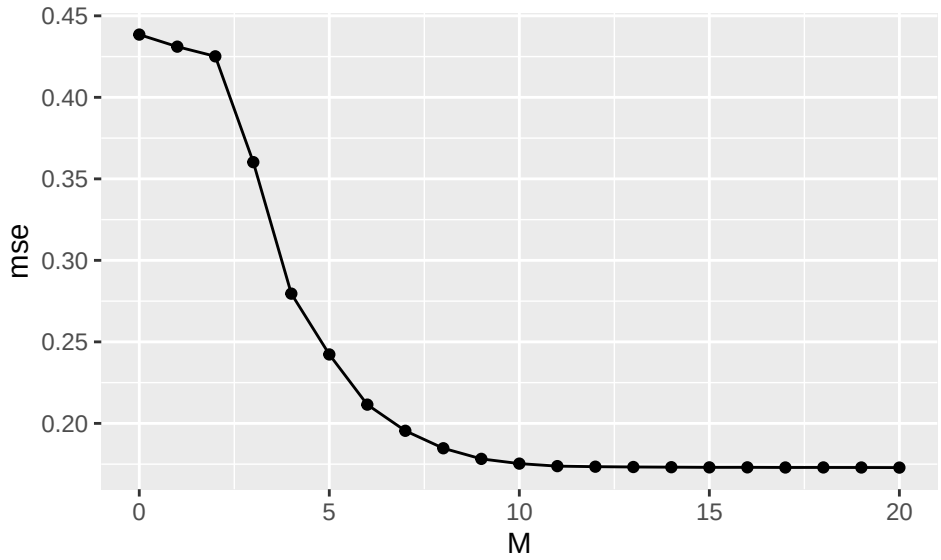




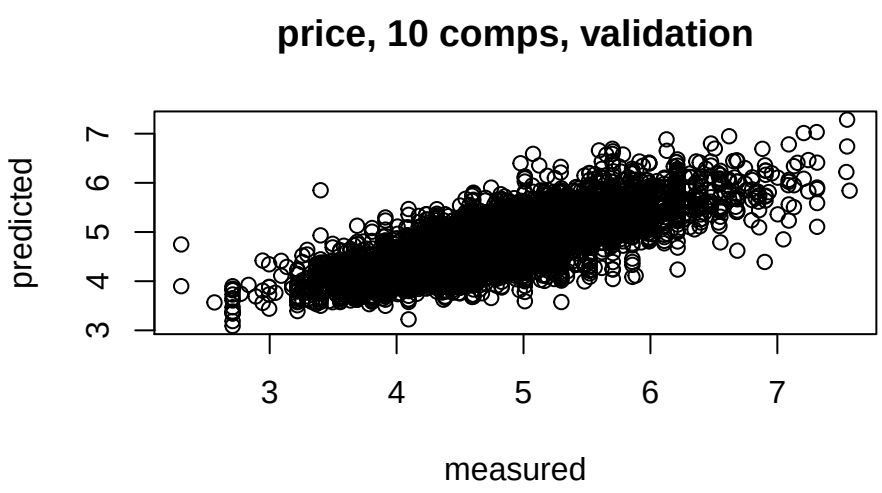

```
## [1] 0
```

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

```
## [1] 0.2192
```




```
## Data:    X dimension: 9000 61
##  Y dimension: 9000 1
## Fit method: kernelpls
## Number of components considered: 20
##
## VALIDATION: RMSEP
## Cross-validated using 10 random segments.
##        (Intercept)  1 comps  2 comps  3 comps  4 comps  5 comps  6 comps
## CV          0.6622   0.6566   0.6520   0.6002   0.5288   0.4922   0.4599
## adjCV       0.6622   0.6565   0.6521   0.6003   0.5287   0.4922   0.4598
##        7 comps  8 comps  9 comps  10 comps  11 comps  12 comps  13 comps
## CV      0.4421   0.4298   0.4222    0.4187    0.4169    0.4165    0.4162
## adjCV   0.4420   0.4297   0.4221    0.4186    0.4168    0.4164    0.4161
##        14 comps  15 comps  16 comps  17 comps  18 comps  19 comps  20 comps
## CV       0.4161    0.4159    0.4160    0.4159    0.4159    0.4159    0.4158
## adjCV    0.4160    0.4159    0.4159    0.4157    0.4158    0.4157    0.4156
##
## TRAINING: % variance explained
##        1 comps  2 comps  3 comps  4 comps  5 comps  6 comps  7 comps  8 comps
## X       34.151   94.290    98.62    99.76    99.82    99.86    99.88    99.88
## price    1.785    3.124    17.91    36.32    44.91    52.00    55.71    58.17
##        9 comps  10 comps  11 comps  12 comps  13 comps  14 comps  15 comps
## X        99.89     99.91     99.92     99.93     99.94     99.95     99.95
## price    59.68     60.33     60.67     60.77     60.83     60.86     60.90
##        16 comps  17 comps  18 comps  19 comps  20 comps
## X         99.96     99.96     99.97     99.97     99.97
## price     60.93     61.04     61.07     61.10     61.15
```



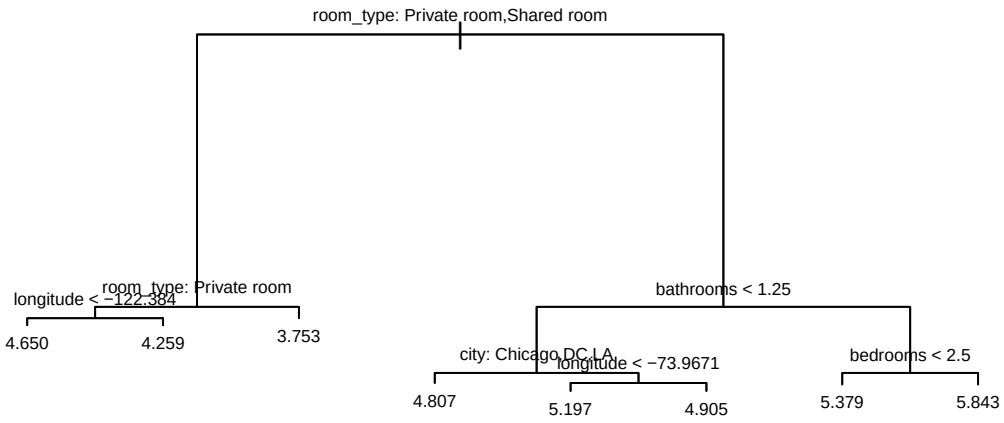

```
## [1] 0
```

```
## [1] 0.1765
```

Trees

Our next method is decision trees. Fitting the fit using all predictors, we obtained a tree where five variables were actually used in the tree construction. Those variables were room_type, longitude, bathrooms, city and bedrooms. The tree has eight terminal nodes and obtained a test mean squared error of 0.1926. Because the relationship between the features and the response is approximately linear, linear regression obtains a better MSE than the regression tree. The cv.tree function is used to perform cross-validation to determine the optimal level of tree complexity. The optimal level is 8 and then using the prune.tree function, we attempted to prune our tree to the chosen complexity but found that our original tree is the same as the pruned tree. The table below shows the log prices converted to dollars to help explain the terminal node values. Node 1 is the farthest node on the left.

```
## [1] "Test MSE of Tree:  0.1926"
```

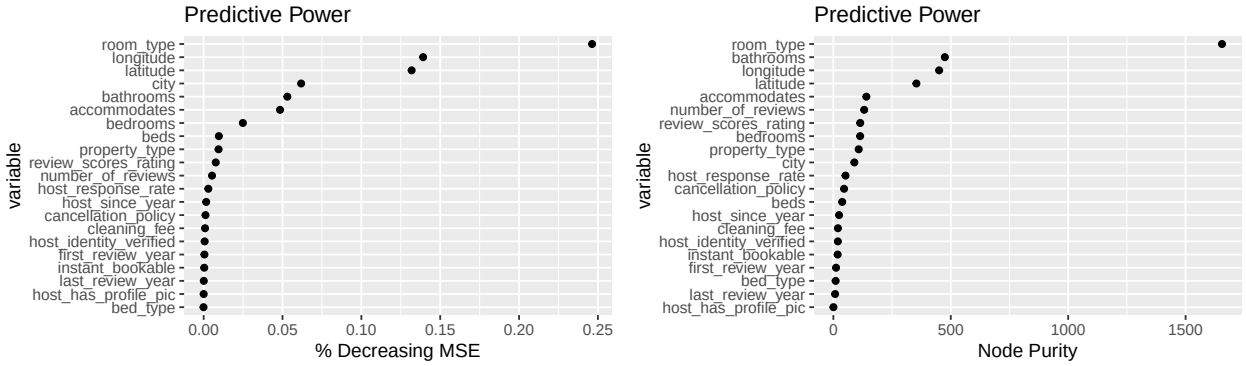


TerminalNode	Log_Price	Price
1	4.650	104.58
2	4.259	70.74
3	3.753	42.65
4	4.807	122.36
5	5.197	180.73
6	4.905	134.96
7	5.379	216.81
8	5.843	344.81

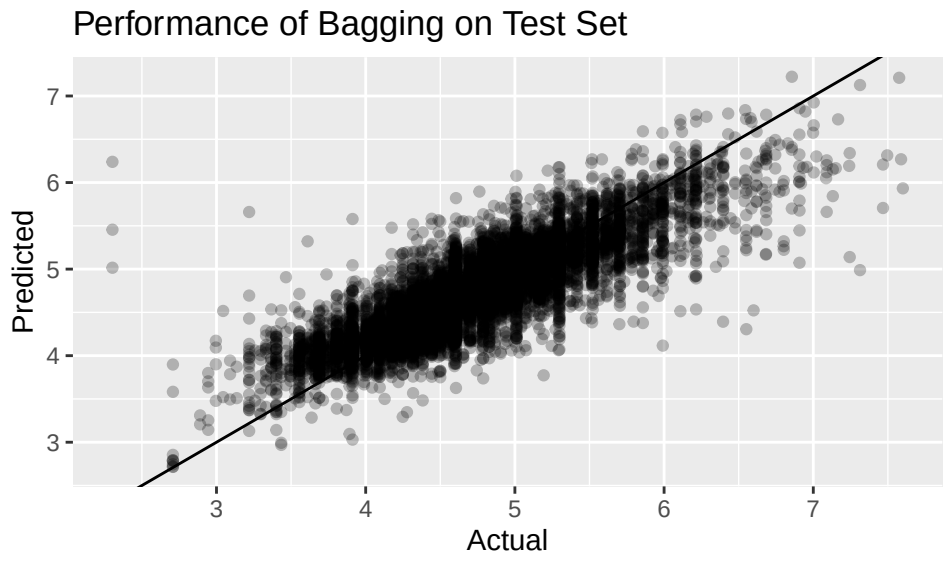
Bagging

Because decision trees suffer from high variance, we then moved onto bagging or bootstrap aggregation to see if we could lower the MSE of our trees. Although bagging decision trees can be slow as it has to average hundreds or thousands of trees together, it won't lead to any overfitting. We performed bagging on the training set to predict price and found that room type, bathrooms and latitude/longitude where the most important variables. Our bagging model is able to obtain a MSE of 0.1293 and is found to be one of our best methods.

```
## [1] "Test MSE of Bagging:  0.1292"
```



The plot below shows an Actual vs Predicted plot for the predicted values using bagging. In general, the data points move positively along the diagonal line with a slight horizontal tilt in the data points compared to the horizontal line. There are a few outliers especially on the right side of the plot. It appears bagging is predicting slightly higher than the actual values for prices less than 5 and slightly lower than the actual values for prices more than 5. However, this plot shows the overall effectiveness for bagging.

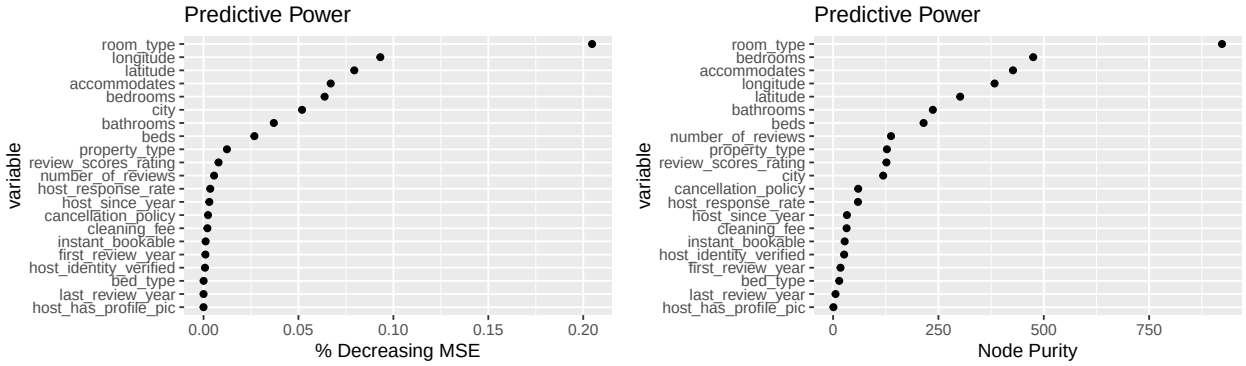


Random Forests

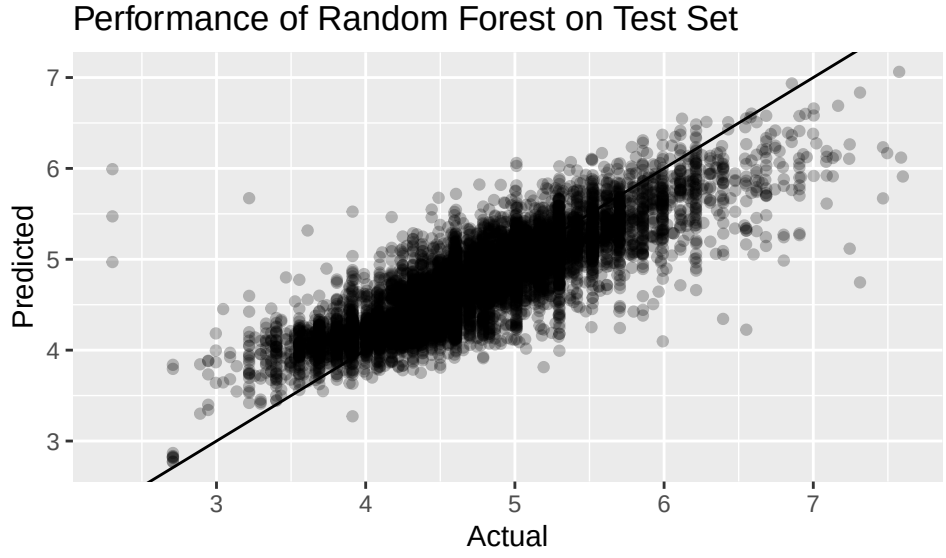
Next, we wanted to determine if random forests could provide an improvement over the bagged trees and see if decorrelating the trees will lead to better MSE result. The choice of m or the size of the predictor subset is set to the square root of the total predictor set. Similar to bagging, we didn't have to worry about overfitting and the importance of each variable can be examined.

After performing the random forests with 5000 trees, we found that room type is by far the most important variable with longitude, latitude, bedrooms and accommodates also showing importance. With a MSE of 0.1301, the overall performance is slightly worse than bagging but still very close.

```
## [1] "Test MSE of Random Forest:  0.1301"
```

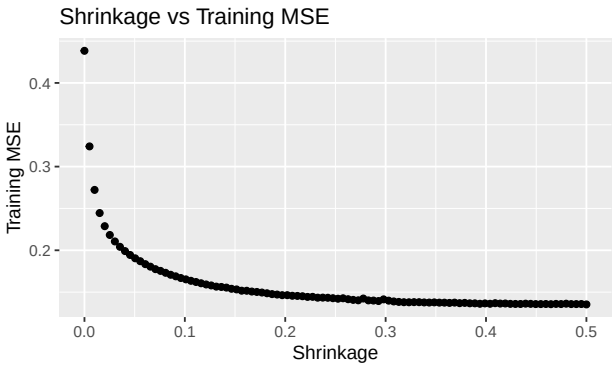


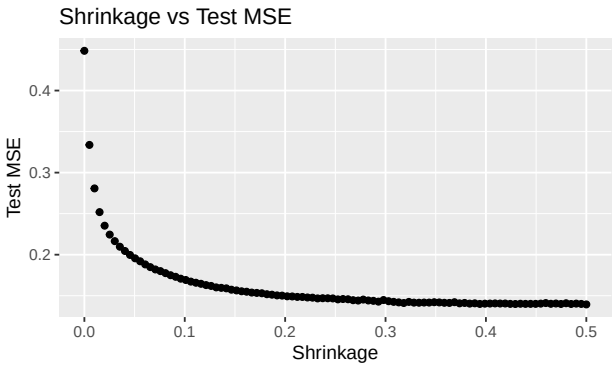
Similar to bagging, the plot below shows an Actual vs Predicted plot for the predicted values using random forests. We see the points still move positively along the diagonal line and there are a few outliers. This plot is very similar to the plot with bagging and because the methods are very similar, this is what we should expect.

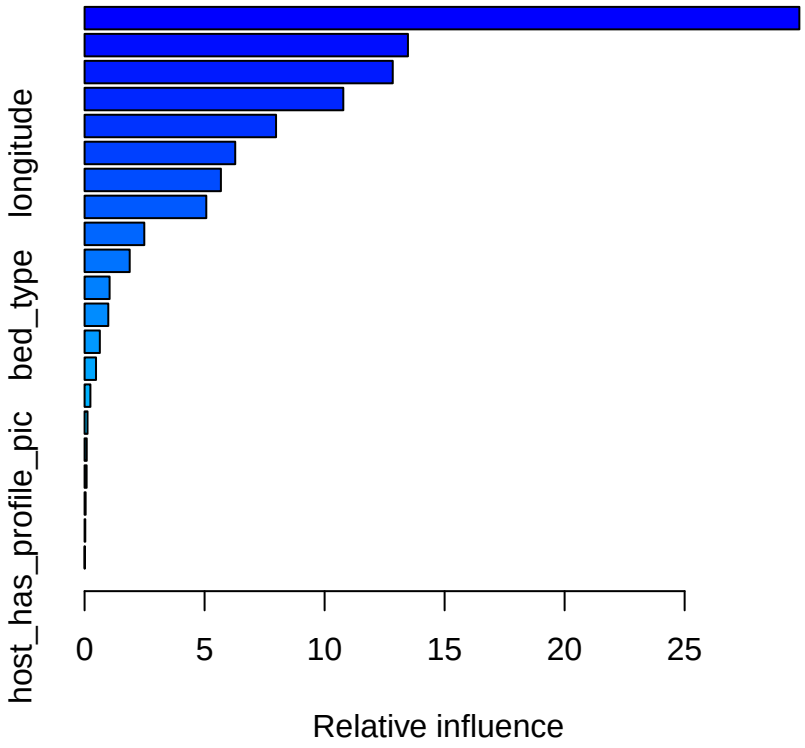


Boosting

In an attempt to improve our prediction results from our trees, our last method is boosting. Before boosting, we run a simulation to find the best lambda value and found that the best lambda value to occur around 0.05. With the number of trees set to 5000, boosting is conducted. The most important variable was actually property type with bathrooms, room type and bedrooms also relatively important. It's interesting to see property type with such high importance when bagging and random forests had room type with the largest importance. The MSE is 0.1317 which is also close to the results for bagging and random forests.








```
##                                      var      rel.inf
## property_type                property_type 29.779802653
## bathrooms                       bathrooms 13.472402813
## room_type                       room_type 12.839234976
## bedrooms                         bedrooms 10.779398886
## accommodates                 accommodates  7.977330195
## longitude                       longitude  6.276126148
## latitude                         latitude  5.679358467
## beds                                beds  5.070692325
## review_scores_rating review_scores_rating  2.487085780
```

```
## city                                     city  1.879404356
## number_of_reviews           number_of_reviews  1.042334213
## bed_type                             bed_type  0.987888976
## host_response_rate         host_response_rate  0.635311569
## last_review_year             last_review_year  0.476462492
## cancellation_policy       cancellation_policy  0.241651202
## host_since_year               host_since_year  0.122746510
## host_identity_verified host_identity_verified  0.088495309
## first_review_year           first_review_year  0.084816737
## cleaning_fee                     cleaning_fee  0.042991062
## instant_bookable             instant_bookable  0.028654999
## host_has_profile_pic     host_has_profile_pic  0.007810332
```

```
## [1] "Testing MSE for Boosted Model: 0.1317"
```

MSE Results

Methods	MSE	MSE_Dollars
Linear Regression	0.3063	1.36
PCR	0.2192	1.25
PLS	0.1765	1.19
Splines	0.4422	1.56
GAM	0.1612	1.17
Trees	0.1926	1.21
Bagging	0.1292	1.14
Random Forest	0.1301	1.14
Boosting	0.1317	1.14

After implementing all of our methods, we see that bagging and random forests perform the best with boosting not far behind and splines having the worst performance. To put these errors rate in terms of dollars and not logarithmic dollars, the MSE_Dollars is provided. As shown, each method has about a dollar of error. We think that is very good when considering the amount of listings and variables used to train these models. Given present data, we think this could potentially be useful for current or future AirBnB hosts and AirBnB renters.

City Prediction

Our original data set is from a past and completed Kaggle competition. Thus, the data was split into training and testing files. The testing file has no response column included because that is how they would determine a winner. Even though our testing results can't be compared to the actual prices, we can still use it for testing our final model.

References: