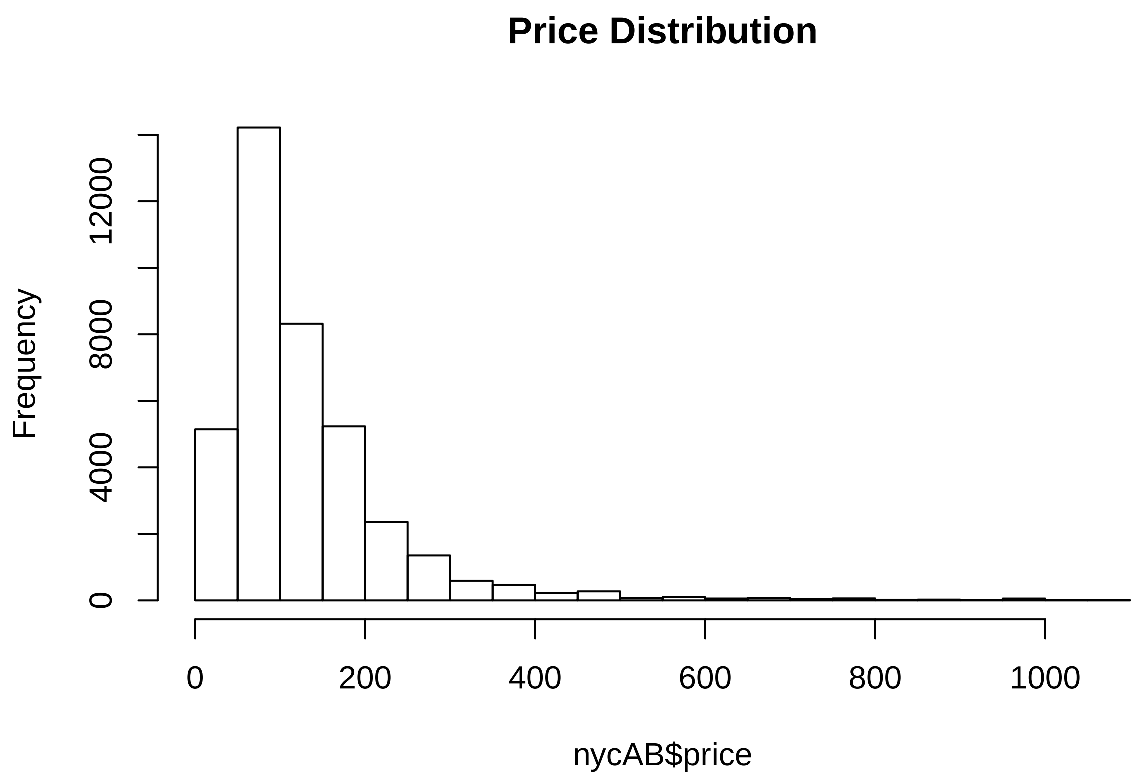
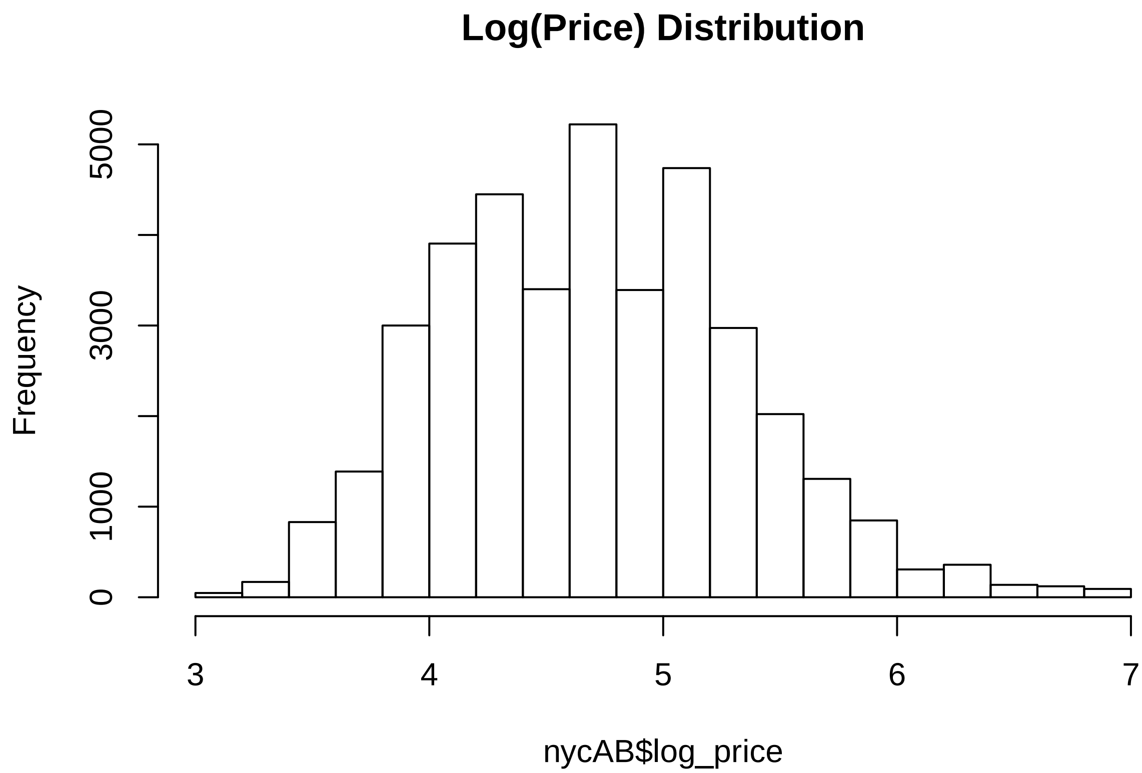
Code Output

Jadyn Gonzalez, Ariana Bucio, Matt Mead, Bryce Viorst

12/10/2019

We are trying to predict price. Upon inspecting the original 'price' variable, we found a very skewed distribution. We used a log transformation to get a closer to normal distribution of price.





Our first model we used was a lasso model to attempt to find the best variables to select in other models. We ran the model and found the best value for lambda.

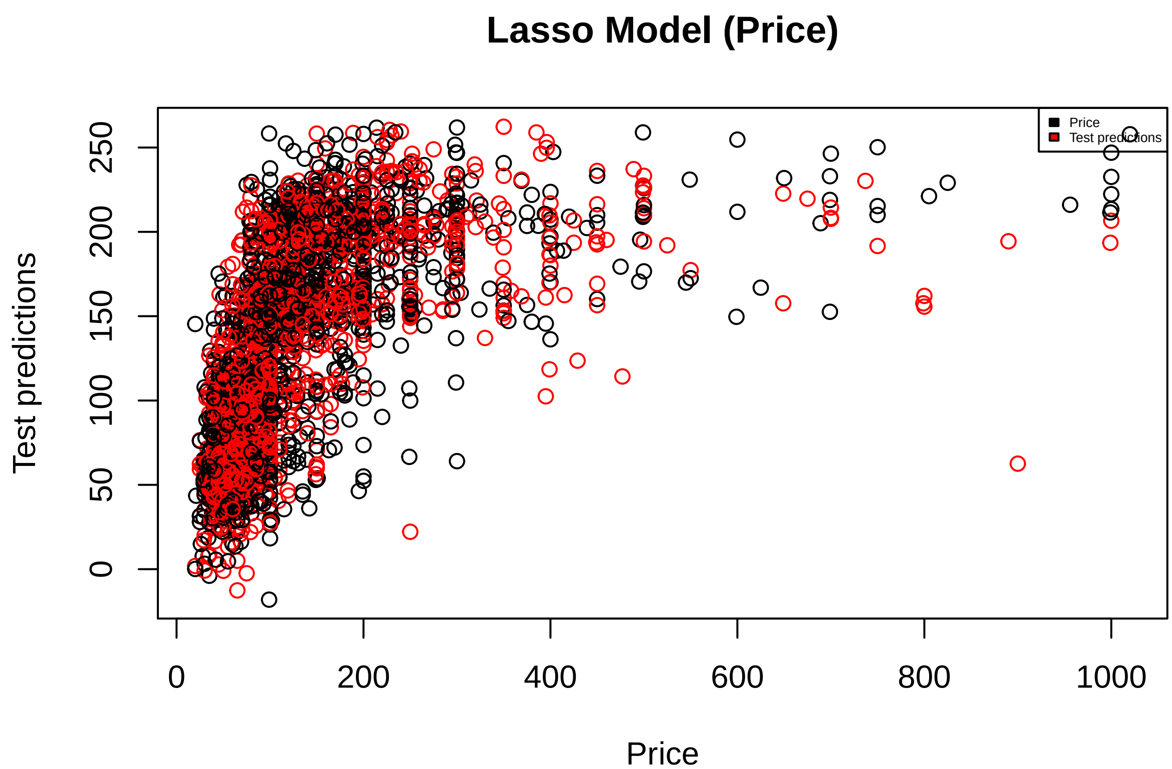
```
## [1] 0.05948865
```

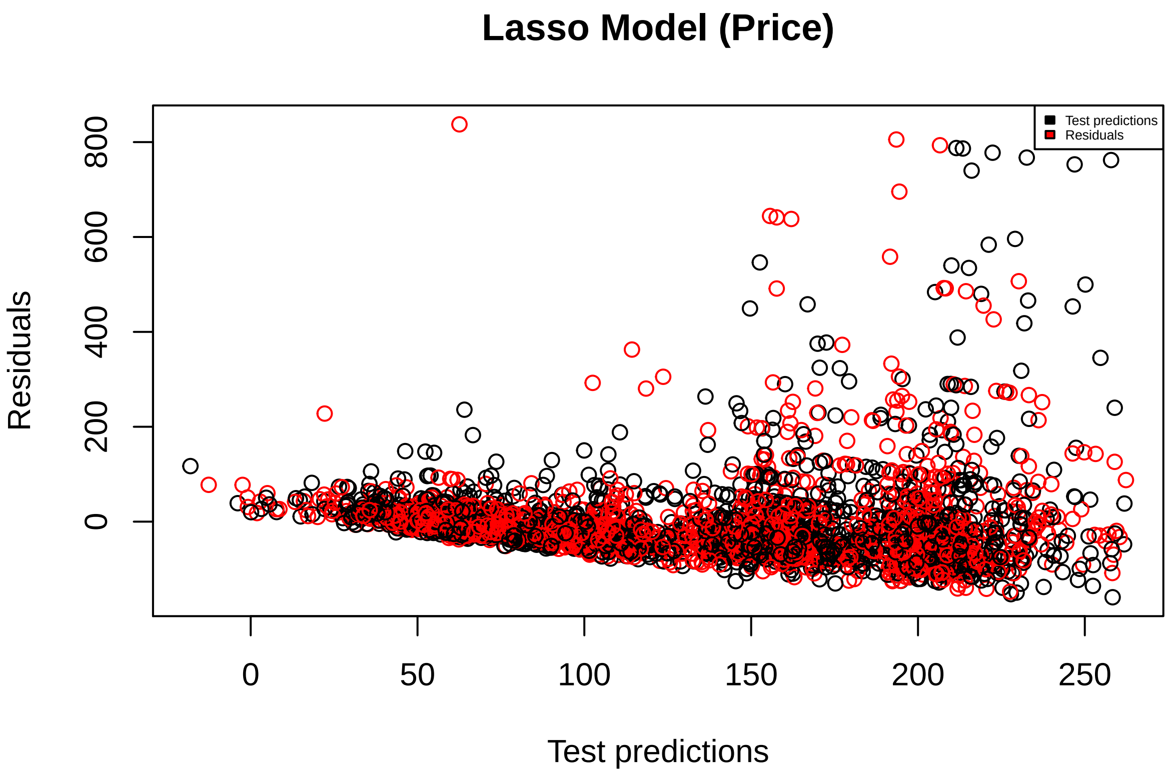
Once we found lambda, we re-trained the model, found the RMSE, and then generated our plots.

```
##   RMSE_train_LASSO1 RMSE_test_LASSO1
## 1          92.16207         95.21181
```

```
## 16 x 1 sparse Matrix of class "dgCMatrix"
##                                s0
## (Intercept)            81.45209427
## neighbourhood_group1  -28.15757481
## neighbourhood_group2    .
## neighbourhood_group3   45.88488602
## neighbourhood_group4  -18.84415477
## neighbourhood_group5  -39.23124813
## room_type1            101.38941322
## room_type2              .
## room_type3            -33.95420207
## minimum_nights         -0.06708542
## number_of_reviews      -0.07109291
## reviews_per_month      -3.71106781
## availability_365        0.13984070
```

```
## log_num_reviews          -6.20032411
## reviews_per_month_log  13.98975966
## minimum_nights_log    -11.87091041
```



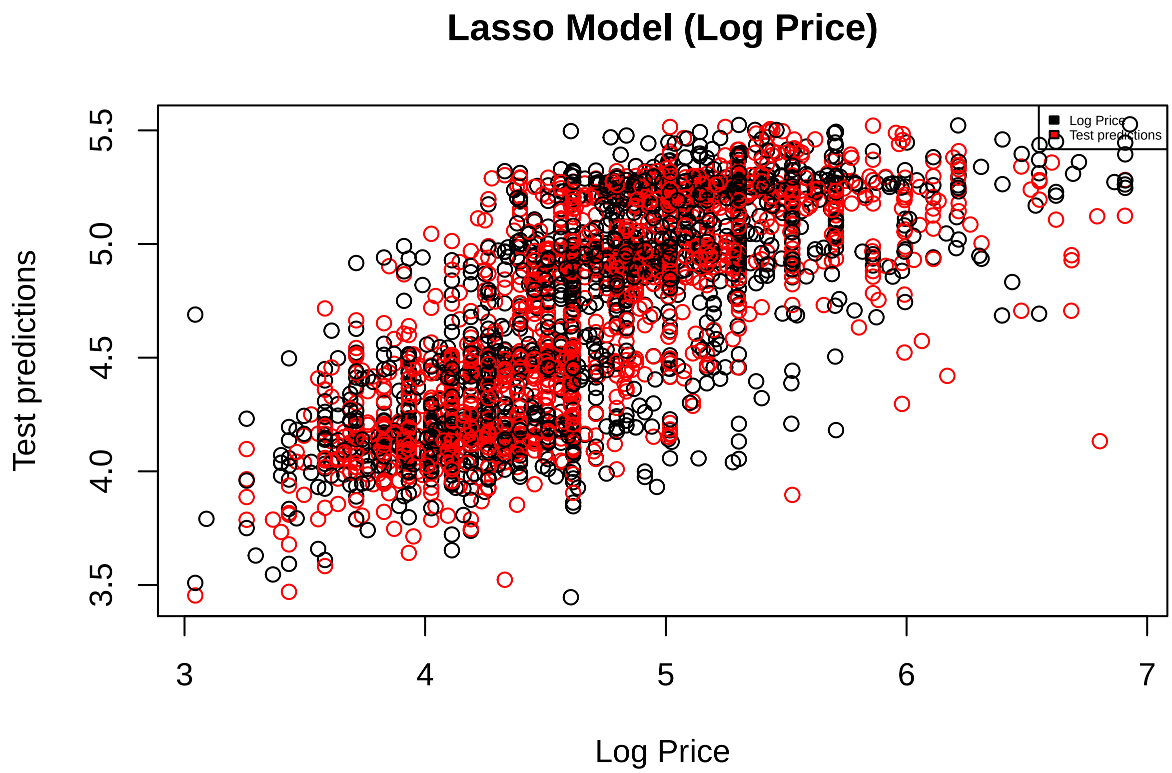


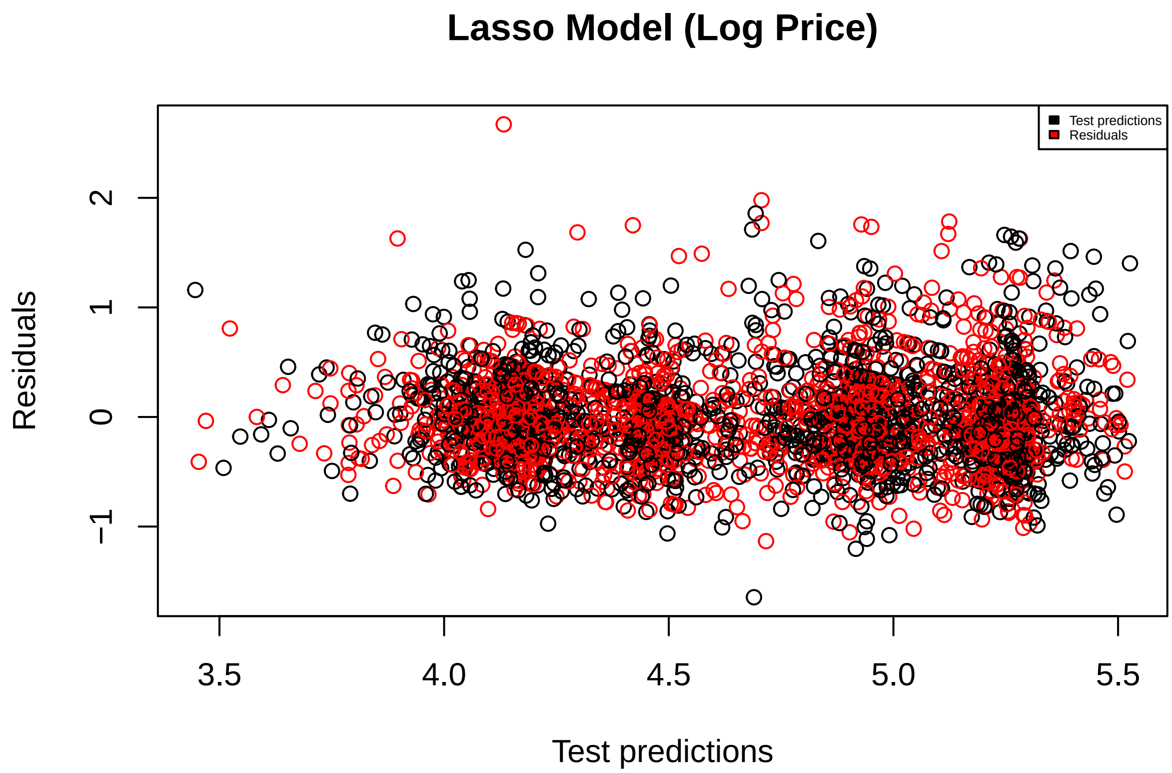
Now we followed the same steps using the log of price as the output variable.

```
## [1] 0.0003843989
```

```
##   RMSE_train_LASSO2 RMSE_test_LASSO2
## 1          1.556107         1.550764
```

```
## 16 x 1 sparse Matrix of class "dgCMatrix"
##                                  s0
## (Intercept)             4.2804399581
## neighbourhood_group1   -0.2708510635
## neighbourhood_group2    .
## neighbourhood_group3    0.3072219089
## neighbourhood_group4   -0.1609740474
## neighbourhood_group5   -0.3083413599
## room_type1              0.7829308254
## room_type2              .
## room_type3             -0.4114584451
## minimum_nights          .
## number_of_reviews      -0.0004145433
## reviews_per_month      -0.0193414123
## availability_365        0.0006935239
## log_num_reviews        -0.0161582488
## reviews_per_month_log   0.0566029825
## minimum_nights_log     -0.0893523178
```


Lasso Model (Log Price)
Log Price
Test predictions
Test predictions
3.5
4.0
4.5
5.0
5.5
3
4
5
6
7
Log Price



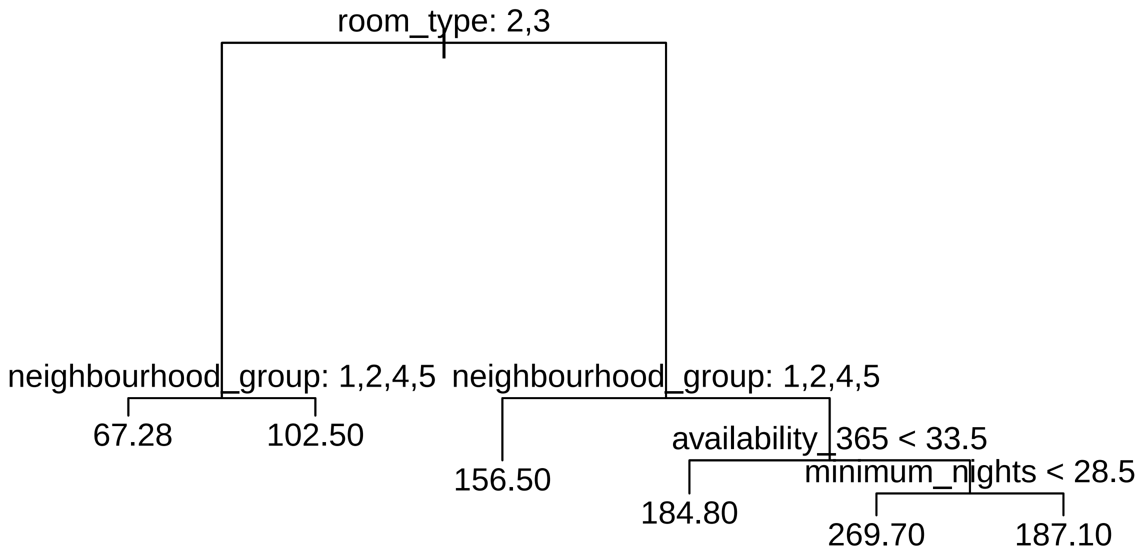
We wanted to compare predictions of both price and log(price) against actual values of price and log(price). We stored our results in a dataframe.

```
##              X1 LogPrice      X1.1 Price
## 22034 4.495465 4.330733 107.26377    75
## 34496 4.504848 3.713572 110.19634    40
## 37086 4.442489 3.931826 115.13914    50
## 23454 4.044292 4.110874  43.94955    60
## 17881 4.117792 4.060443  51.77777    57
## 36337 4.529334 4.394449 120.41686    80
```

Our second model choice was a decision tree.

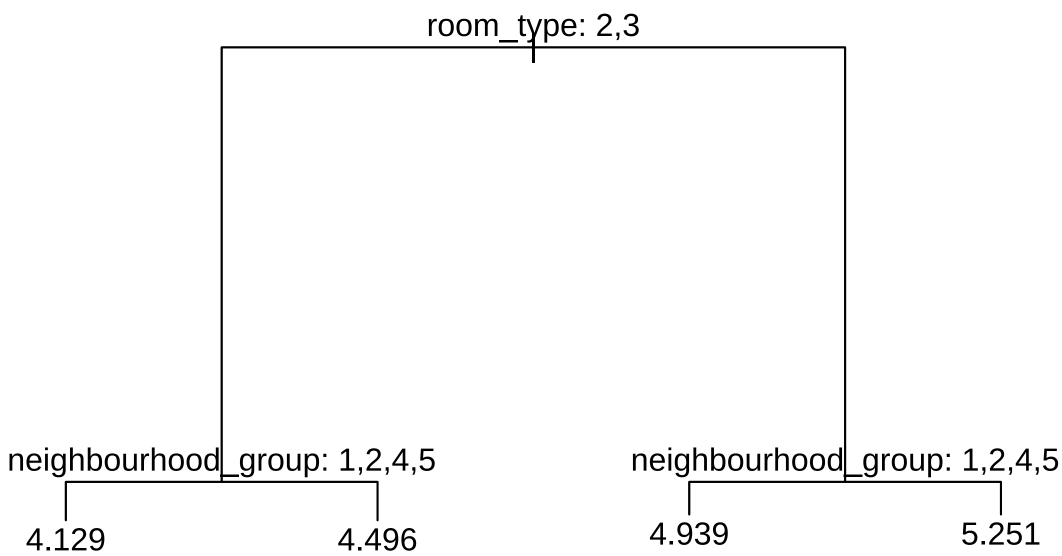
The tree output for 'price'.

```r
plot(regMod)
text(regMod, pretty=0)
```



The tree output for 'log_price'.

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



We found the best tree size for 'price' and 'log__price' respectively.

```
## [1] 6
```

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

Then we compared RMSE. Since the model appeared to be overfit, we continued to the random forest model.

```
## [1] "RMSE train v test of 'price'."
```

```
## [1] 92.59612
```

```
## [1] 96.96846
```

```
## [1] "RMSE train v test of 'log_price'."
```

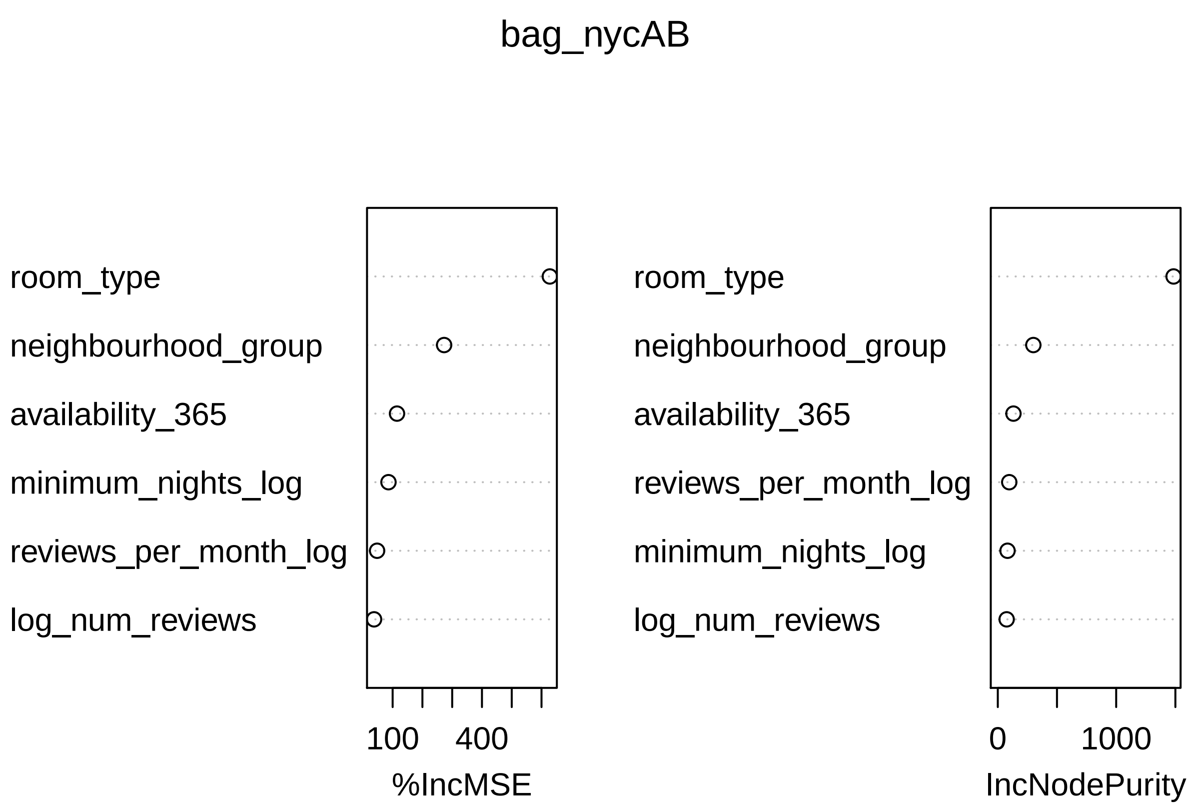
```
## [1] 1.583465
```

```
## [1] 1.586467
```

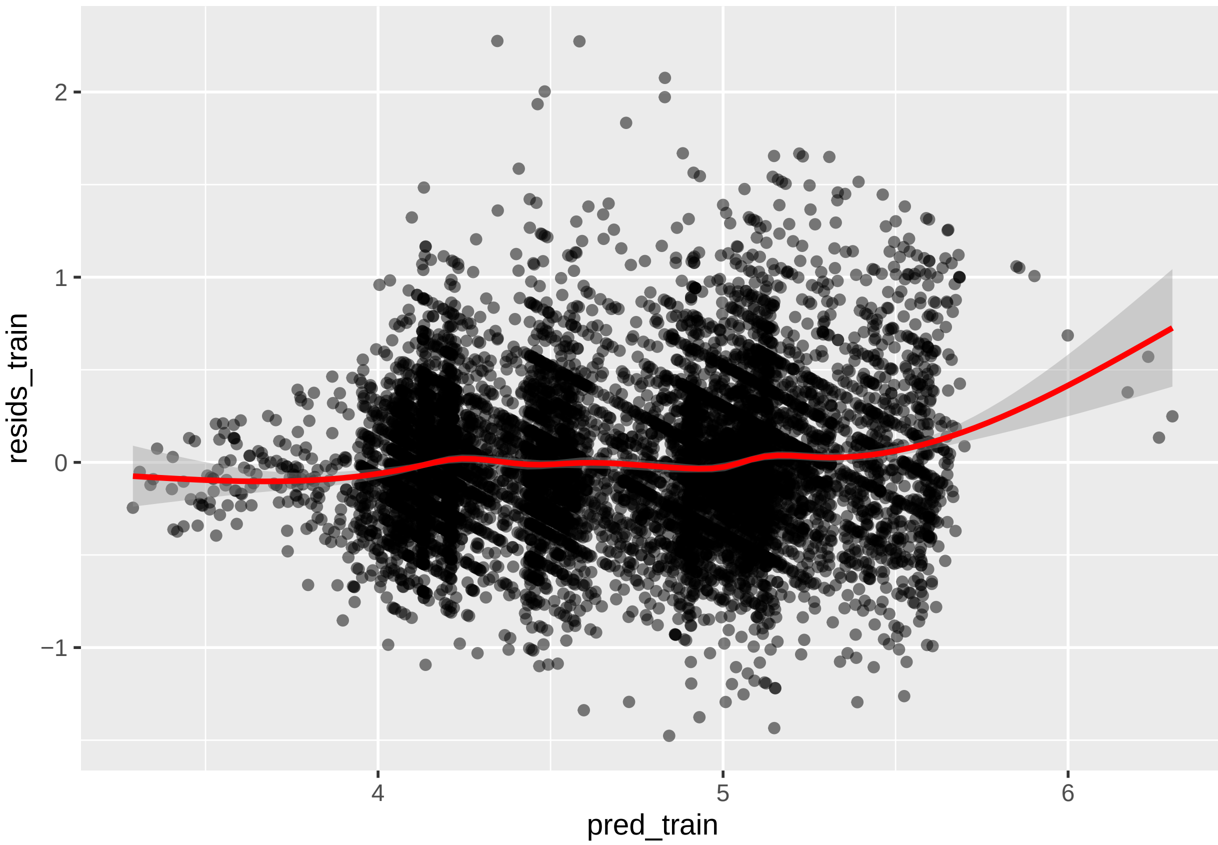
The output for the random forest model.

```
## Registered S3 method overwritten by 'xts':
##   method     from
##   as.zoo.xts zoo
```

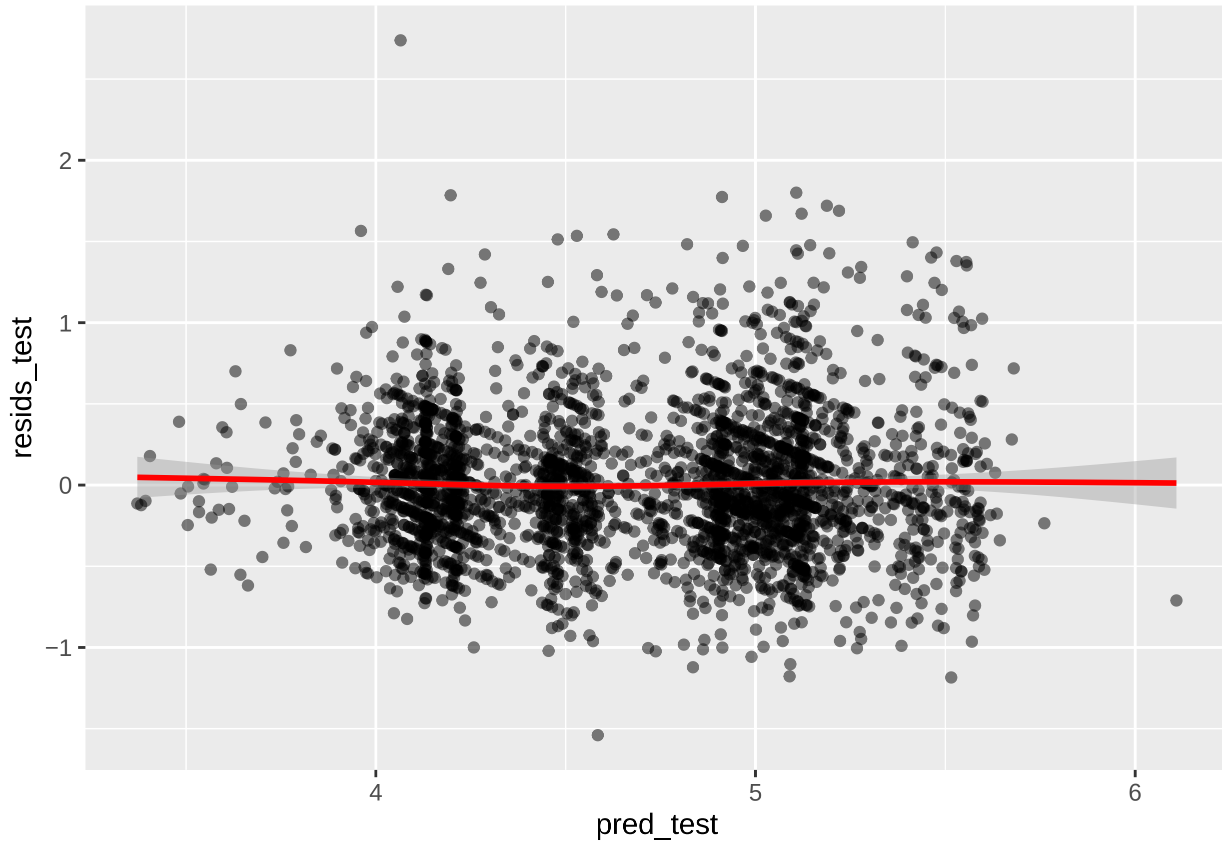
```
## Registered S3 method overwritten by 'quantmod':
##   method            from
##   as.zoo.data.frame zoo

## Registered S3 methods overwritten by 'forecast':
##   method             from
##   fitted.fracdiff    fracdiff
##   residuals.fracdiff fracdiff

##
## 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: 0.1875766
##                     % Var explained: 54.12

##                         %IncMSE IncNodePurity
## neighbourhood_group   272.86560     300.26048
## room_type             627.93867    1484.79088
## availability_365      114.68042     132.11790
## log_num_reviews        37.63119      74.88056
## reviews_per_month_log  47.81748      96.80946
## minimum_nights_log     86.07159      82.30310
```



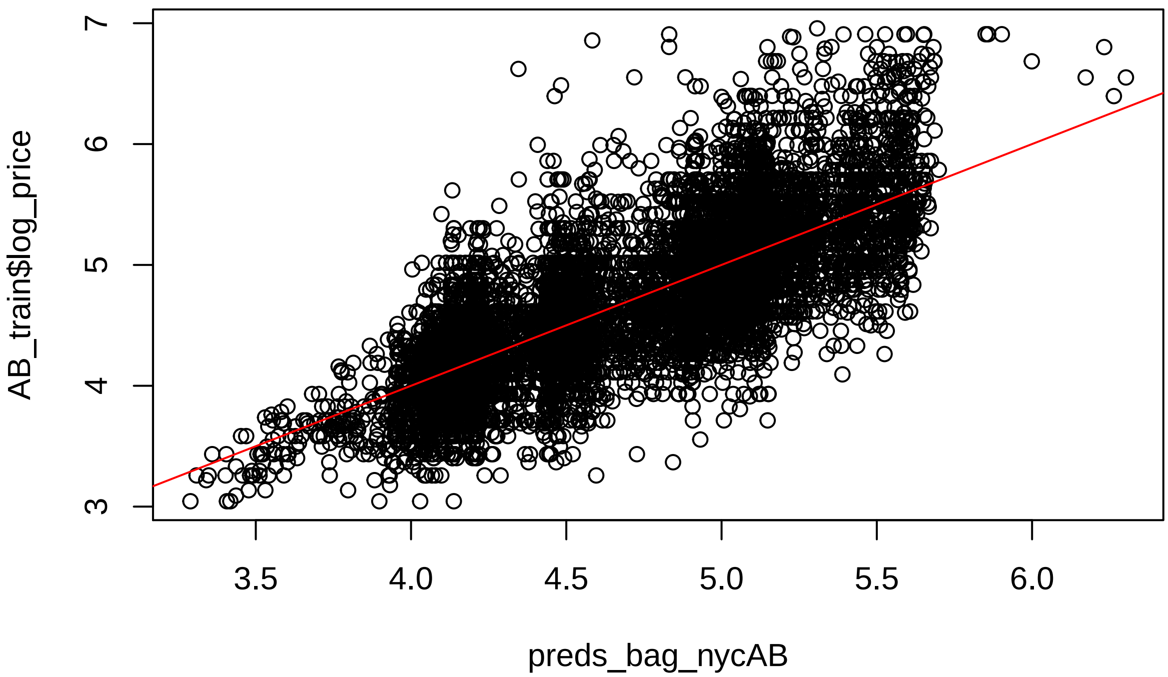
```
## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
```



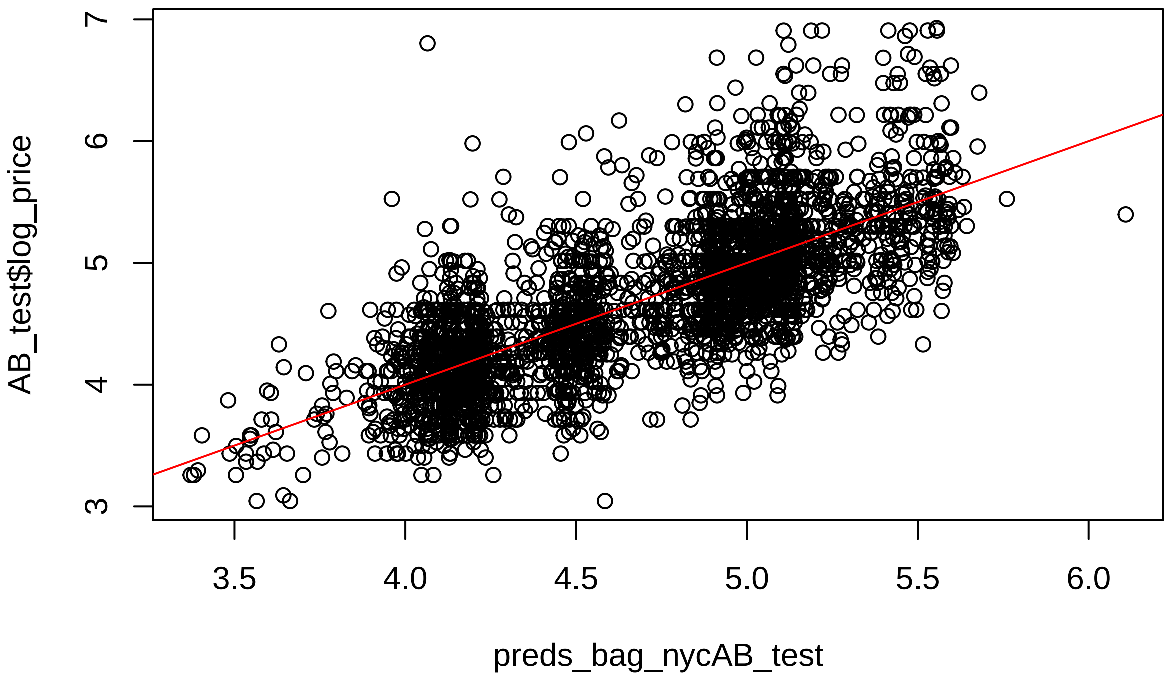

```
## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
```



2
1
0
−1
resids_test
4
5
6
pred_test



AB_train$log_price
preds_bag_nycAB
3
4
5
6
7
3.5
4.0
4.5
5.0
5.5
6.0



We compared RMSE and found that random forest showed the best results.

```
## [1] "RMSE test"

## [1] 1.533759

## [1] "RMSE train"

## [1] 1.493944
```