

8 Appendix

REGRESSION ANALYSIS

Loading librarys:

```
library(readxl)
library(car)
library(leaps)
library(MASS)
```

Loading the original data:

```
project_data = read.csv("AB_NYC_2019.csv")
```

Omitting null values:

```
nrow(project_data)

## [1] 48895

project_data = na.omit(project_data)
nrow(project_data)

## [1] 38843
```

Random sampling:

```
data_sample = project_data[sample(nrow(project_data) ,
                                    nrow(project_data)*(5000/nrow(project_data))),]
```

We saved off one randome sample of the data in order to be working with the same data set for analysis:

```
setwd("~/Documents/ISYE 4031 Regression & Forcasting/Project/new-york-city-airbnb-open-data")
project_data_sample = read_excel("sample_data.xlsx")

## New names:
## * ` ` -> ...1
attach(project_data_sample)
```

Creating dataframe of numerical data for correlation:

```
df = project_data_sample[c("price","minimum_nights","number_of_reviews","reviews_per_month",
                           "calculated_host_listings_count","availability_365")]

cor(df)

##                                     price minimum_nights
## price                            1.0000000000  0.0008842091
## minimum_nights                   0.0008842091  1.0000000000
## number_of_reviews                -0.0428759061 -0.0830325449
```

```

## reviews_per_month           -0.0418979190  -0.1441522316
## calculated_host_listings_count  0.0644480350   0.0779305734
## availability_365            0.1115295098   0.0763575569
##                                         number_of_reviews reviews_per_month
## price                           -0.04287591    -0.041897919
## minimum_nights                  -0.08303254    -0.144152232
## number_of_reviews                1.000000000   0.524745585
## reviews_per_month                 0.52474559    1.000000000
## calculated_host_listings_count -0.05849106    -0.004301983
## availability_365                0.20041460    0.183818170
##                                         calculated_host_listings_count
## price                            0.064448035
## minimum_nights                   0.077930573
## number_of_reviews                 -0.058491060
## reviews_per_month                 -0.004301983
## calculated_host_listings_count   1.000000000
## availability_365                 0.185098224
##                                         availability_365
## price                            0.11152951
## minimum_nights                   0.07635756
## number_of_reviews                 0.20041460
## reviews_per_month                 0.18381817
## calculated_host_listings_count   0.18509822
## availability_365                 1.000000000

```

There was no severe multicollinearity at this point -> highest being .524

Creation of base model:

```

basemodel = lm(price~minimum_nights+number_of_reviews+reviews_per_month
               +calculated_host_listings_count+availability_365)
summary(basemodel)

##
## Call:
## lm(formula = price ~ minimum_nights + number_of_reviews + reviews_per_month +
##     calculated_host_listings_count + availability_365)
##
## Residuals:
##    Min      1Q  Median      3Q     Max 
## -162.3   -72.5  -31.4   29.7  4814.6 
##
## Coefficients:
##                               Estimate Std. Error t value Pr(>|t|)    
## (Intercept)             135.25344   3.42564  39.483 < 2e-16 ***
## minimum_nights          -0.23102   0.15199  -1.520  0.12857    
## number_of_reviews        -0.14164   0.05428  -2.610  0.00909 **  
## reviews_per_month        -4.09871   1.53334  -2.673  0.00754 **  
## calculated_host_listings_count  0.25830   0.09103  2.837  0.00457 **  
## availability_365         0.14704   0.01766   8.324 < 2e-16 ***
## ---                        
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 154.7 on 4994 degrees of freedom

```

```

## Multiple R-squared:  0.02,  Adjusted R-squared:  0.01902
## F-statistic: 20.39 on 5 and 4994 DF,  p-value: < 2.2e-16

```

According to the summary, the significant fields (p-value < 0.01) are number_of_reviews, reviews_per_month, calculated_host_listings_count, availability_365(most significant).

Variance Inflation Factor (VIF) for base model:

```
vif(basemodel)
```

```

##           minimum_nights          number_of_reviews
##                1.036736                  1.413968
## reviews_per_month calculated_host_listings_count
##                1.416381                  1.049445
##           availability_365
##                1.104509

```

According the VIF test, there is still no severe multicollinearity among the eight independent variables.

Choosing the best model from all subsets

```

all_model<-regsubsets(price~.,data=df,nbest=2,method = "exhaustive")
all_sum = summary(all_model)
Rsq = round(all_sum$rsq*100, digit=1)
adj_Rsq = round(all_sum$adjr2*100, digit=1)
Cp = round(all_sum$cp, digit=1)
SSE = all_sum$rss
k = as.numeric(rownames(all_sum$which))
n = all_model$nn
S = round(sqrt(all_sum$rss/(n-(k+1))), digit=2)
SSTO = sum((price - mean(price))^2)
aic = round(2*(k+1)+n*log(SSE/n),digits=2)
SSE = round(SSE,digits=2)
cbind(Rsq, adj_Rsq, Cp, S, all_sum$outmat)

##           Rsq    adj_Rsq   Cp      S           minimum_nights  number_of_reviews
## 1 ( 1 ) "1.2" "1.2" "36.6" "155.25" " "        " "
## 1 ( 2 ) "0.4" "0.4" "78.8" "155.9" " "        " "
## 2 ( 1 ) "1.7" "1.6" "16"   "154.92" " "        "*"
## 2 ( 2 ) "1.6" "1.6" "18"   "154.95" " "        " "
## 3 ( 1 ) "1.8" "1.8" "10.5" "154.82" " "        "*"
## 3 ( 2 ) "1.8" "1.8" "11"   "154.82" " "        " "
## 4 ( 1 ) "2"   "1.9" "6.3"  "154.74" " "        "*"
## 4 ( 2 ) "1.9" "1.8" "10.8" "154.8"  "*"       " "
## 5 ( 1 ) "2"   "1.9" "6"    "154.71" "*"       "*"
##           reviews_per_month calculated_host_listings_count availability_365
## 1 ( 1 ) " "        " "                    "*"          "
## 1 ( 2 ) " "        "*"                   " "          "
## 2 ( 1 ) " "        " "                    "*"          "
## 2 ( 2 ) "*"       " "                    "*"          "
## 3 ( 1 ) " "        "*"                   "*"          "
## 3 ( 2 ) "*"       "*"                   "*"          "
## 4 ( 1 ) "*"       "*"                   "*"          "
## 4 ( 2 ) "*"       "*"                   "*"          "
## 5 ( 1 ) "*"       "*"                   "*"          "

```

The best models with the lowest C_p that is less than or equal to $k + 1$, high adjusted R^2 , and lowest variance:

5(1): minimum_nights + number_of_reviews + reviews_per_month + calculated_host_listings_count + availability_365

4(1) number_of_reviews + reviews_per_month + calculated_host_listings_count + availability_365

Creation of the full model:

```
full.lm <- lm(price ~ . , data=df)
min.lm <- lm(price ~ 1, data=df)
step_both = step(min.lm, list(upper=full.lm), direction='both')

## Start: AIC=50512.86
## price ~ 1
##
##                               Df Sum of Sq      RSS      AIC
## + availability_365           1   1517286 120462487 50452
## + calculated_host_listings_count  1   506649 121473124 50494
## + number_of_reviews            1   224241 121755532 50506
## + reviews_per_month            1   214128 121765645 50506
## <none>                           121979773 50513
## + minimum_nights               1       95 121979678 50515
##
## Step: AIC=50452.28
## price ~ availability_365
##
##                               Df Sum of Sq      RSS      AIC
## + number_of_reviews            1   540705 119921782 50432
## + reviews_per_month            1   491555 119970932 50434
## + calculated_host_listings_count  1   242358 120220129 50444
## <none>                           120462487 50452
## + minimum_nights               1     7147 120455341 50454
## - availability_365             1   1517286 121979773 50513
##
## Step: AIC=50431.79
## price ~ availability_365 + number_of_reviews
##
##                               Df Sum of Sq      RSS      AIC
## + calculated_host_listings_count  1   177558 119744224 50426
## + reviews_per_month              1   145232 119776550 50428
## <none>                           119921782 50432
## + minimum_nights                1   25398 119896384 50433
## - number_of_reviews              1   540705 120462487 50452
## - availability_365              1   1833750 121755532 50506
##
## Step: AIC=50426.38
## price ~ availability_365 + number_of_reviews + calculated_host_listings_count
##
##                               Df Sum of Sq      RSS      AIC
## + reviews_per_month            1   149260 119594965 50422
## <none>                           119744224 50426
## + minimum_nights               1   33530 119710695 50427
## - calculated_host_listings_count  1   177558 119921782 50432
## - number_of_reviews              1   475904 120220129 50444
```

```

## - availability_365           1   1541716 121285940 50488
##
## Step: AIC=50422.14
## price ~ availability_365 + number_of_reviews + calculated_host_listings_count +
##       reviews_per_month
##
##                                     Df Sum of Sq      RSS      AIC
## + minimum_nights                  1    55303 119539661 50422
## <none>                           119594965 50422
## - reviews_per_month               1    149260 119744224 50426
## - number_of_reviews                1    159884 119754849 50427
## - calculated_host_listings_count  1    181585 119776550 50428
## - availability_365                 1   1616392 121211356 50487
##
## Step: AIC=50421.83
## price ~ availability_365 + number_of_reviews + calculated_host_listings_count +
##       reviews_per_month + minimum_nights
##
##                                     Df Sum of Sq      RSS      AIC
## <none>                           119539661 50422
## - minimum_nights                  1    55303 119594965 50422
## - number_of_reviews                1   163020 119702681 50427
## - reviews_per_month                1   171033 119710695 50427
## - calculated_host_listings_count  1   192719 119732380 50428
## - availability_365                 1   1658663 121198324 50489

```

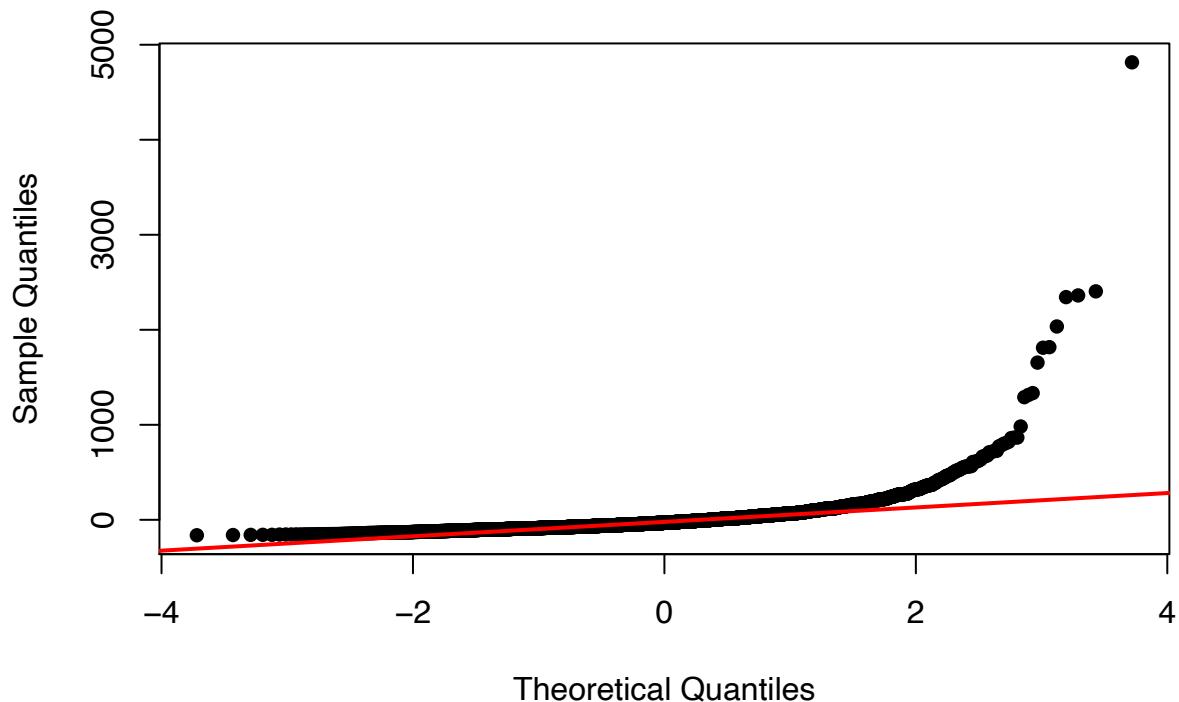
The best model has 5 variables:

```

bestmodel= lm(price~number_of_reviews+reviews_per_month+calculated_host_listings_count
              +availability_365 + minimum_nights)
qqnorm(resid(bestmodel),pch=16,main="Normal Probability Plot of Residuals")
qqline(resid(bestmodel),col='red',lwd=2)

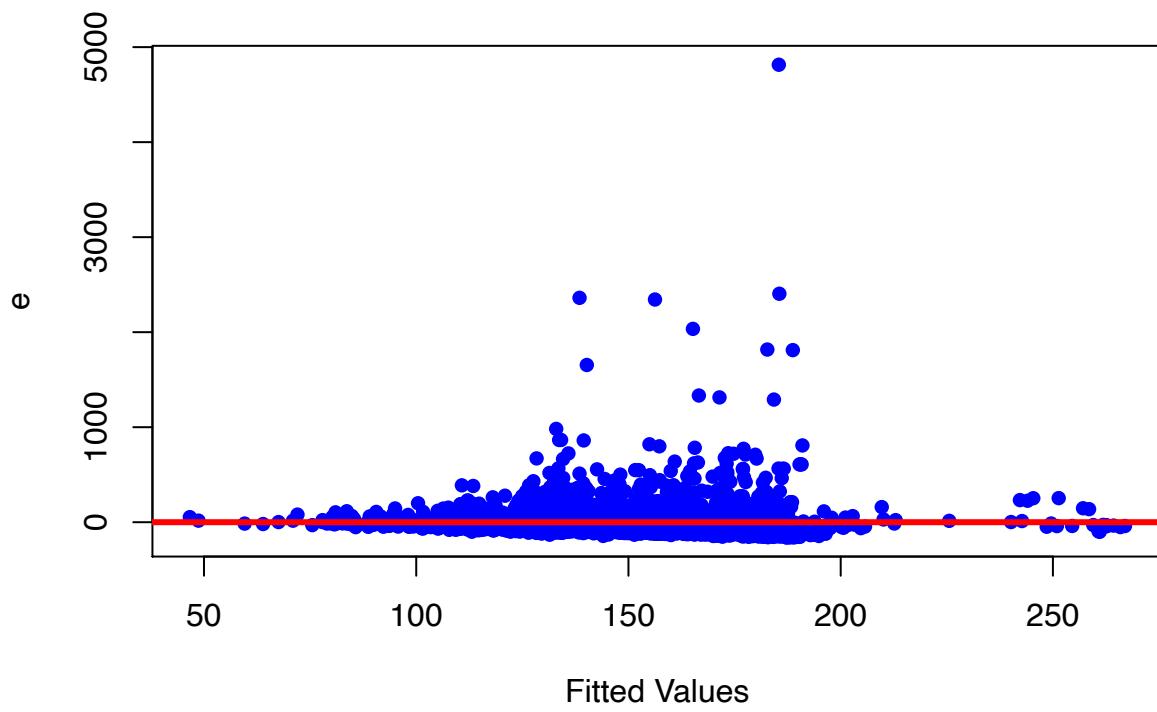
```

Normal Probability Plot of Residuals



```
plot(fitted(bestmodel),resid(bestmodel), pch=16, col="blue", ylab=bquote(paste("e")),
      xlab=bquote(paste("Fitted Values")),main="Plot of e vs Fitted Values")
abline(0,0,col="red",lwd=3)
```

Plot of e vs Fitted Values

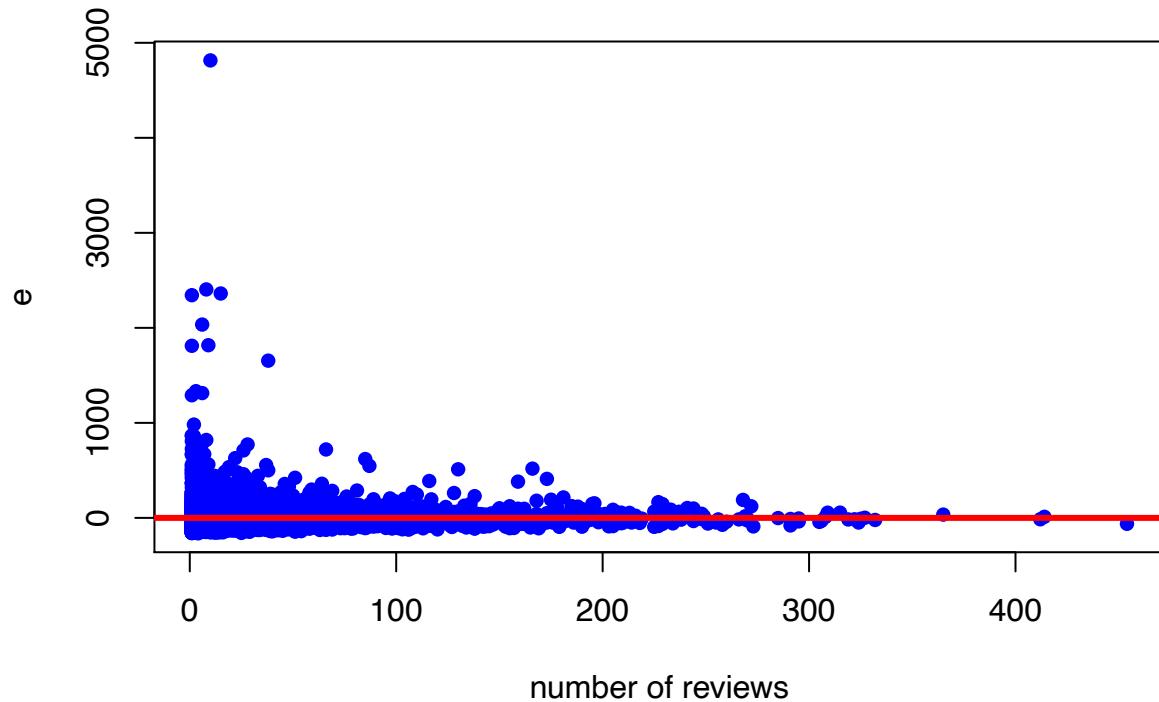


```

plot(number_of_reviews,resid(bestmodel), pch=16, col="blue", ylab=bquote(paste("e")),
      xlab=bquote(paste("number of reviews")),main="Plot of e vs number of reviews")
abline(0,0,col="red",lwd=3)

```

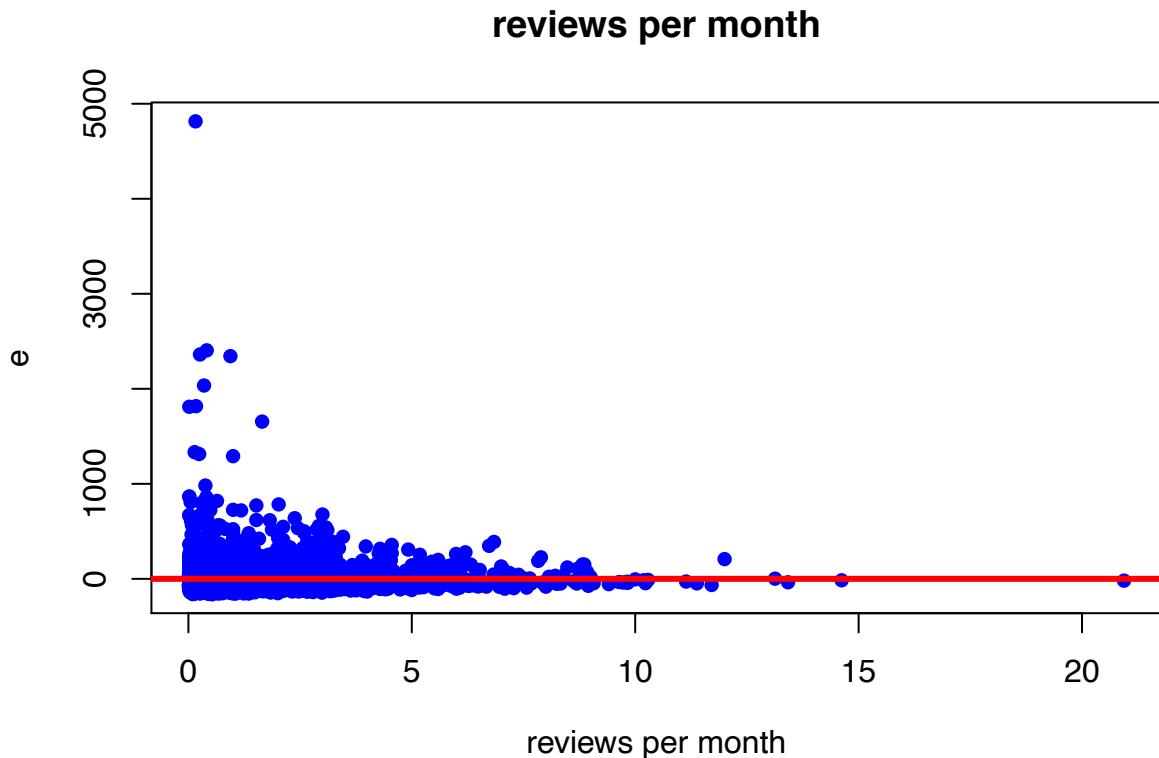
Plot of e vs number of reviews



```

plot(reviews_per_month,resid(bestmodel), pch=16, col="blue", ylab=bquote(paste("e")),
      xlab=bquote(paste("reviews per month")),main="reviews per month")
abline(0,0,col="red",lwd=3)

```



Outlier analysis:

```

k = 5
n = length(price)
LV_cutoff = 2*(k+1)/n; print(LV_cutoff)

## [1] 0.0024

Hi = hatvalues(bestmodel)
Hi[Hi>LV_cutoff]

##      14       26       40      115      140      149
## 0.002522602 0.003711449 0.008459252 0.007991201 0.003234863 0.002476020
##      198      209      224      227      249      259
## 0.002963222 0.006947654 0.002417892 0.007845635 0.005940952 0.003610938
##      281      283      290      306      312      320
## 0.003035470 0.003065521 0.004348396 0.004043015 0.006471384 0.003510556
##      324      345      350      356      364      381
## 0.003448336 0.008302446 0.003133159 0.013510081 0.034922588 0.003267111
##      383      385      393      433      440      468
## 0.003598141 0.003948391 0.003069451 0.004664993 0.004639866 0.007036337
##      473      478      494      512      516      521
## 0.036936159 0.003720652 0.002431800 0.004708325 0.002574757 0.004018389
##      522      551      552      559      564      574
## 0.004224196 0.003493510 0.002849927 0.003221705 0.006683083 0.005913257
##      601      613      644      650      660      666
## 0.003801395 0.003198287 0.034677305 0.002839477 0.003546620 0.006819212
##      703      759      760      767      793      813
## 0.005270799 0.003479479 0.004488949 0.004585076 0.003979616 0.005230298
##      818      821      822      834      865      869

```

```

## 0.002707626 0.006624475 0.004254303 0.004142646 0.003199051 0.004437243
##     883      890      900      910      926      967
## 0.003969310 0.005393451 0.007296868 0.003304665 0.003078140 0.004987843
##     970      975      976      986      990      996
## 0.002619482 0.008391394 0.002738862 0.003497561 0.003001551 0.003599175
##    1015     1022     1027     1031     1064     1068
## 0.003113545 0.122966053 0.003010231 0.034896745 0.014751771 0.003858320
##    1112     1132     1157     1158     1168     1182
## 0.064638450 0.002494741 0.002587536 0.003621971 0.002509404 0.003400794
##    1203     1213     1246     1268     1272     1287
## 0.006896052 0.003380493 0.002817283 0.003225680 0.083071968 0.035310538
##    1292     1294     1345     1367     1396     1409
## 0.004168601 0.003387739 0.035324329 0.003089018 0.003510130 0.035417123
##    1413     1418     1442     1445     1458     1464
## 0.003126688 0.007642430 0.003776359 0.016705329 0.006644867 0.003342389
##    1487     1512     1537     1539     1540     1557
## 0.006962231 0.003219815 0.003190514 0.003028388 0.003500428 0.034846487
##    1561     1586     1599     1608     1634     1648
## 0.034665204 0.003254317 0.003269651 0.002849715 0.002860008 0.003097908
##    1650     1663     1696     1712     1752     1753
## 0.067625450 0.003568238 0.003648083 0.003215460 0.002836581 0.003120561
##    1765     1766     1771     1779     1786     1797
## 0.003498316 0.034868869 0.005797592 0.002795953 0.008666333 0.003025721
##    1824     1828     1901     1932     1940     1968
## 0.003350459 0.003958937 0.003305205 0.004033948 0.003595417 0.003406886
##    1976     1983     2001     2046     2053     2065
## 0.034975235 0.004864847 0.008228342 0.002808369 0.002680854 0.003522957
##    2083     2114     2134     2146     2161     2213
## 0.005302469 0.003190124 0.008766591 0.002493261 0.004233086 0.034901888
##    2224     2233     2289     2320     2366     2371
## 0.003652673 0.003158702 0.002814818 0.003261535 0.004876405 0.005248680
##    2373     2382     2398     2415     2442     2448
## 0.003422079 0.003786494 0.005505313 0.003018490 0.004560327 0.002844196
##    2449     2470     2479     2491     2495     2498
## 0.003592405 0.125091106 0.006656775 0.002993852 0.035225785 0.020018802
##    2499     2512     2557     2565     2575     2589
## 0.003068758 0.002599592 0.029139743 0.004000230 0.005518019 0.002667048
##    2591     2594     2616     2620     2630     2632
## 0.002543322 0.003602259 0.003107967 0.003176713 0.004221576 0.002659103
##    2643     2649     2663     2664     2685     2688
## 0.003783856 0.004010869 0.002910607 0.125200682 0.004482128 0.003573110
##    2689     2707     2733     2752     2763     2781
## 0.014131511 0.003132450 0.004755574 0.011753062 0.003912254 0.003770303
##    2785     2794     2808     2830     2851     2863
## 0.010508066 0.004921253 0.002532225 0.003239942 0.003103512 0.034868610
##    2865     2909     2934     2946     2987     2994
## 0.005107823 0.003641210 0.003888108 0.002558039 0.004236793 0.004761857
##    3009     3021     3023     3039     3043     3059
## 0.002801115 0.005446694 0.003233720 0.002976330 0.003120704 0.003154562
##    3079     3140     3154     3164     3172     3198
## 0.004199816 0.003337111 0.003953768 0.004208732 0.004105032 0.003709710
##    3212     3216     3224     3226     3232     3251
## 0.008602744 0.003719287 0.002663286 0.003817642 0.004294326 0.003237697
##    3277     3285     3355     3357     3364     3367

```

```

## 0.002842208 0.005540077 0.016101913 0.013771660 0.005001712 0.007993835
##      3382      3392      3403      3405      3418      3430
## 0.003245554 0.002429079 0.002828763 0.004489871 0.002926780 0.002550042
##      3435      3465      3475      3502      3504      3511
## 0.002424768 0.003804483 0.004341065 0.002435752 0.006708951 0.002479718
##      3546      3560      3588      3634      3640      3650
## 0.034927465 0.002481044 0.006075799 0.007416557 0.011095788 0.017474412
##      3668      3714      3723      3728      3733      3755
## 0.004042200 0.007851366 0.003642627 0.003253954 0.002792981 0.004427914
##      3791      3808      3828      3840      3873      3874
## 0.009101531 0.002448022 0.003917530 0.003258894 0.005045912 0.005106068
##      3884      3899      3906      3918      3931      3933
## 0.003233893 0.002713257 0.003145530 0.011508215 0.003545213 0.003052742
##      3953      3959      3963      3970      3984      3994
## 0.004105375 0.002731368 0.002903346 0.004691024 0.002510677 0.003904864
##      3995      3999      4000      4015      4020      4032
## 0.007209701 0.002472497 0.002691892 0.003190979 0.003926032 0.006406253
##      4035      4051      4070      4085      4100      4113
## 0.005310442 0.004635526 0.035553976 0.003172733 0.005355637 0.003913515
##      4126      4138      4202      4228      4276      4285
## 0.006955530 0.005052855 0.002551104 0.005160906 0.036128130 0.003012734
##      4286      4351      4373      4404      4419      4436
## 0.003604519 0.002652881 0.002917745 0.003508809 0.007294479 0.002523768
##      4446      4453      4505      4506      4518      4531
## 0.005229064 0.005305419 0.002732241 0.035063160 0.008220801 0.003673717
##      4543      4565      4588      4597      4601      4603
## 0.002636776 0.007908950 0.035408256 0.003761733 0.002632938 0.034882621
##      4614      4627      4629      4661      4676      4688
## 0.007684276 0.006803763 0.002499270 0.003356588 0.003471265 0.010023581
##      4693      4694      4698      4704      4711      4748
## 0.003369665 0.002919855 0.035151555 0.004770149 0.006324688 0.004117794
##      4777      4785      4790      4795      4800      4836
## 0.003165602 0.034834351 0.006048057 0.006983270 0.034901892 0.002560851
##      4850      4857      4881      4890      4894      4931
## 0.003037584 0.005810815 0.034896129 0.003349645 0.003227326 0.004827617
##      4934      4945      4965      4993
## 0.003091229 0.003583784 0.010605291 0.035284258

```

These are the outliers.

Studentized Deleted Residuals (SDR)

```

Res_SE = summary(bestmodel)$sigma
di = resid(bestmodel)/(1-Hi)
SSE = Res_SE^2*(n-(k+1))
SDR = resid(bestmodel)*sqrt((n-k-2)/(SSE*(1-Hi)-(resid(bestmodel))^2))
SDR[abs(SDR)>qt(0.975,n-k-2)] #some evidence

```

```

##      21      41     112     202     206     319     369
## 4.070703 5.239872 8.405900 3.544375 3.941335 2.098437 3.311446
##      401     424     439     459     477     478     549
## 4.132448 2.731495 2.328101 2.836344 2.071854 2.251589 2.766181
##      600     604     611     697     751     753     790
## 5.010102 15.940243 2.756368 1.996238 4.616233 2.028928 5.579848

```

```

##      843     1000    1154     1161    1240     1295    1318
## 3.549353 15.510904 2.097619 3.202641 3.020367 8.554335 2.310004
##      1360     1425    1434     1451    1457     1472    1492
## 2.339152 2.391363 2.376218 2.324857 2.497322 4.012644 1.991918
##      1539     1583    1725     1730    1732     1746    1748
## 2.515631 3.496112 10.820696 3.626003 4.667683 2.280833 2.345261
##      1780     1817    1849     1868    1896     1943    2004
## 2.029912 2.129898 2.733191 2.145636 2.107037 3.939377 34.684247
##      2185     2187    2259     2312    2344     2361    2362
## 4.340589 2.207392 2.512124 2.236311 3.601311 2.055105 3.638736
##      2400     2408    2444     2475    2561     2652    2661
## 2.192896 3.626056 5.178456 3.403189 2.046218 2.518616 2.360450
##      2791     2879    3028     3042    3066     3091    3119
## 4.388774 2.053870 3.241396 4.309396 5.614878 2.096660 4.601583
##      3180     3182    3222     3269    3279     3300    3312
## 6.373757 5.077773 3.358769 2.037009 3.107381 3.248337 2.233702
##      3341     3348    3393     3489    3502     3560    3564
## 3.460408 2.043322 2.082849 5.315740 2.659211 2.470611 4.351110
##      3614     3630    3715     3717    3760     3805    3830
## 3.060194 2.376522 3.002642 3.648465 3.026963 2.287340 2.382893
##      3863     3997    4007     4139    4162     4252    4264
## 3.355454 2.862761 2.947511 4.707467 3.011097 3.994691 15.637837
##      4271     4318    4319     4328    4337     4355    4396
## 4.691184 3.942548 2.864306 13.388104 2.067308 2.791614 3.667013
##      4409     4450    4475     4496    4540     4552    4572
## 8.684772 2.691685 1.981647 11.878671 3.656419 2.601080 2.700536
##      4589     4677    4728     4738    4751     4759    4816
## 2.360054 3.380080 2.080073 3.139684 11.920139 2.153474 2.973391
##      4821     4873    4907     4948
## 5.611364 2.334515 2.208886 2.562440

```

`SDR[abs(SDR)>qt(0.995,n-k-2)] #strong evidence`

```

##      21      41     112     202     206     369     401
## 4.070703 5.239872 8.405900 3.544375 3.941335 3.311446 4.132448
##      424     459     549     600     604     611     751
## 2.731495 2.836344 2.766181 5.010102 15.940243 2.756368 4.616233
##      790     843     1000    1161     1240     1295    1472
## 5.579848 3.549353 15.510904 3.202641 3.020367 8.554335 4.012644
##      1583     1725    1730     1732     1849     1943    2004
## 3.496112 10.820696 3.626003 4.667683 2.733191 3.939377 34.684247
##      2185     2344    2362     2408     2444     2475    2791
## 4.340589 3.601311 3.638736 3.626056 5.178456 3.403189 4.388774
##      3028     3042    3066     3119     3180     3182    3222
## 3.241396 4.309396 5.614878 4.601583 6.373757 5.077773 3.358769
##      3279     3300    3341     3489     3502     3564    3614
## 3.107381 3.248337 3.460408 5.315740 2.659211 4.351110 3.060194
##      3715     3717    3760     3863     3997     4007    4139
## 3.002642 3.648465 3.026963 3.355454 2.862761 2.947511 4.707467
##      4162     4252    4264     4271     4318     4319    4328
## 3.011097 3.994691 15.637837 4.691184 3.942548 2.864306 13.388104
##      4355     4396    4409     4450     4496     4540    4552
## 2.791614 3.667013 8.684772 2.691685 11.878671 3.656419 2.601080
##      4572     4677    4728     4738     4751     4816    4821
## 2.700536 3.380080 3.139684 11.920139 2.973391 5.611364

```

This test outputs a list of data points that show some and strong evidence of being an outlier with respect to its y value.

Cook's Distance Measure

```
CooksD = cooks.distance(bestmodel)
CooksD[CooksD>qf(0.5,k+1, n-k-1)] #Influential

## named numeric(0)
#CooksD[CooksD<qf(0.2,k+1, n-k-1)] #Not Influential
CooksD[CooksD<qf(0.5,k+1, n-k-1) & CooksD>qf(0.2,k+1, n-k-1)]
```

named numeric(0)

This test outputs a list of data points that show influential/non-influential evidence of being an outlier.

Creation of dummy variables (made of room type and all location related variables)

```
attach(project_data_sample)

## The following objects are masked from project_data_sample (pos = 3):
##
##      ...1, availability_365, calculated_host_listings_count,
##      host_id, host_name, id, last_review, latitude, longitude,
##      minimum_nights, name, neighbourhood, neighbourhood_group,
##      number_of_reviews, price, reviews_per_month, room_type

brooklyn_dummy = as.numeric(neighbourhood_group == 'Brooklyn')
queens_dummy = as.numeric(neighbourhood_group == 'Queens')
bronx_dummy = as.numeric(neighbourhood_group == 'Bronx')
staten_dummy = as.numeric(neighbourhood_group == 'Staten Island')
private_room_dummy = as.numeric(room_type == 'Private room')
shared_room_dummy = as.numeric(room_type == 'Shared room')
```

Creation of Location/Price Model and Price/Staten Dummy Model:

```
location_price_model = lm(price ~ + brooklyn_dummy +
                           queens_dummy + bronx_dummy + staten_dummy)

summary(location_price_model)

##
## Call:
## lm(formula = price ~ +brooklyn_dummy + queens_dummy + bronx_dummy +
##     staten_dummy)
##
## Residuals:
##      Min    1Q Median    3Q   Max 
## -163.0  -64.4  -29.0   25.6 4876.6 
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)    
## (Intercept) 178.999    3.316  53.982 < 2e-16 ***
## brooklyn_dummy -55.605    4.677 -11.888 < 2e-16 ***
## queens_dummy   -85.849    7.130 -12.040 < 2e-16 ***
```

```

## bronx_dummy      -91.286      15.061  -6.061 1.45e-09 ***
## staten_dummy     -101.621     23.000  -4.418 1.02e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 152.7 on 4995 degrees of freedom
## Multiple R-squared:  0.04546,   Adjusted R-squared:  0.0447
## F-statistic: 59.48 on 4 and 4995 DF,  p-value: < 2.2e-16
anova(location_price_model)

## Analysis of Variance Table
##
## Response: price
##              Df    Sum Sq Mean Sq F value    Pr(>F)
## brooklyn_dummy     1 1331938 1331938 57.140 4.801e-14 ***
## queens_dummy       1 2940495 2940495 126.147 < 2.2e-16 ***
## bronx_dummy        1 818008 818008 35.092 3.355e-09 ***
## staten_dummy       1 455051 455051 19.522 1.016e-05 ***
## Residuals        4995 116434281   23310
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
staten_model = lm(price~staten_dummy)

summary(staten_model)

##
## Call:
## lm(formula = price ~ staten_dummy)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -142.8  -73.8  -37.8   32.2 4857.2
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 142.837     2.218   64.41 < 2e-16 ***
## staten_dummy -65.459     23.376   -2.80  0.00513 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 156.1 on 4998 degrees of freedom
## Multiple R-squared:  0.001567,   Adjusted R-squared:  0.001367
## F-statistic: 7.842 on 1 and 4998 DF,  p-value: 0.005125
anova(staten_model)

## Analysis of Variance Table
##
## Response: price
##              Df    Sum Sq Mean Sq F value    Pr(>F)
## staten_dummy     1 191083 191083  7.8417 0.005125 **
## Residuals      4998 121788690   24367
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

```

Creation of the BASE model that includes dummy variables

```

overall = lm(price ~ number_of_reviews + minimum_nights + reviews_per_month
             + calculated_host_listings_count+ availability_365 + shared_room_dummy
             + staten_dummy+ brooklyn_dummy + queens_dummy + bronx_dummy + private_room_dummy)

full.lm = lm(price ~ number_of_reviews + minimum_nights + reviews_per_month
             + calculated_host_listings_count+ availability_365 + shared_room_dummy
             + staten_dummy+ brooklyn_dummy + queens_dummy + bronx_dummy
             + private_room_dummy, data = project_data_sample)

summary(full.lm)

##
## Call:
## lm(formula = price ~ number_of_reviews + minimum_nights + reviews_per_month +
##     calculated_host_listings_count + availability_365 + shared_room_dummy +
##     staten_dummy + brooklyn_dummy + queens_dummy + bronx_dummy +
##     private_room_dummy, data = project_data_sample)
##
## Residuals:
##    Min      1Q Median      3Q     Max
## -191.7   -53.5  -15.1   19.1  4876.5
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 213.86270  4.22492 50.619 < 2e-16 ***
## number_of_reviews -0.20184  0.04956 -4.072 4.73e-05 ***
## minimum_nights -0.64761  0.13913 -4.655 3.33e-06 ***
## reviews_per_month -1.40665  1.41241 -0.996  0.319
## calculated_host_listings_count -0.03833  0.08366 -0.458  0.647
## availability_365  0.19493  0.01632 11.946 < 2e-16 ***
## shared_room_dummy -135.87966 14.41519 -9.426 < 2e-16 ***
## staten_dummy -107.97891 21.32911 -5.063 4.29e-07 ***
## brooklyn_dummy -46.55858  4.36708 -10.661 < 2e-16 ***
## queens_dummy -66.63348  6.77516 -9.835 < 2e-16 ***
## bronx_dummy -84.93433 13.99659 -6.068 1.39e-09 ***
## private_room_dummy -108.38674  4.11035 -26.369 < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 140.9 on 4988 degrees of freedom
## Multiple R-squared:  0.1877, Adjusted R-squared:  0.1859
## F-statistic: 104.8 on 11 and 4988 DF,  p-value: < 2.2e-16

```

Variance Inflation Factor (VIF) for base model:

```

vif(full.lm)

##
##          number_of_reviews           minimum_nights
##                  1.420853                  1.046777
## reviews_per_month calculated_host_listings_count
##                  1.448094                  1.068111

```

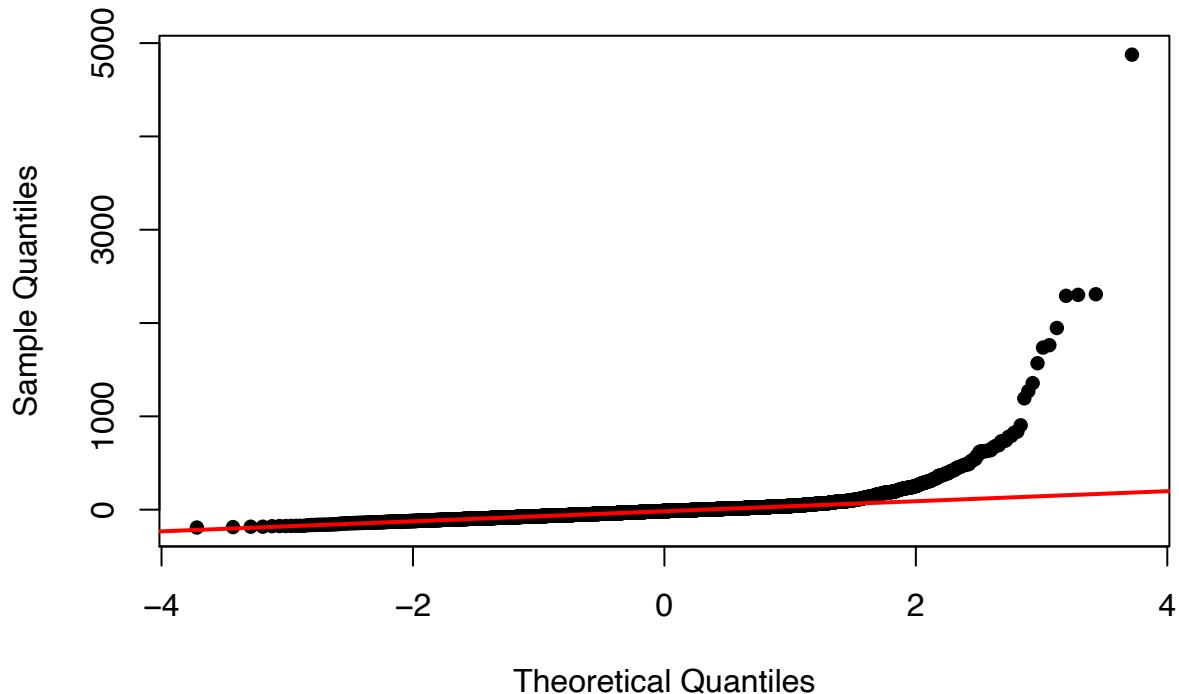
```

##           availability_365          shared_room_dummy
##                           1.135628                           1.025125
##           staten_dummy            brooklyn_dummy
##                           1.021271                           1.175448
##           queens_dummy           bronx_dummy
##                           1.193617                           1.042063
##           private_room_dummy
##                           1.053551

qqnorm(resid(full.lm),pch=16,main="Normal Probability Plot of Residuals")
qqline(resid(full.lm),col='red',lwd=2)

```

Normal Probability Plot of Residuals

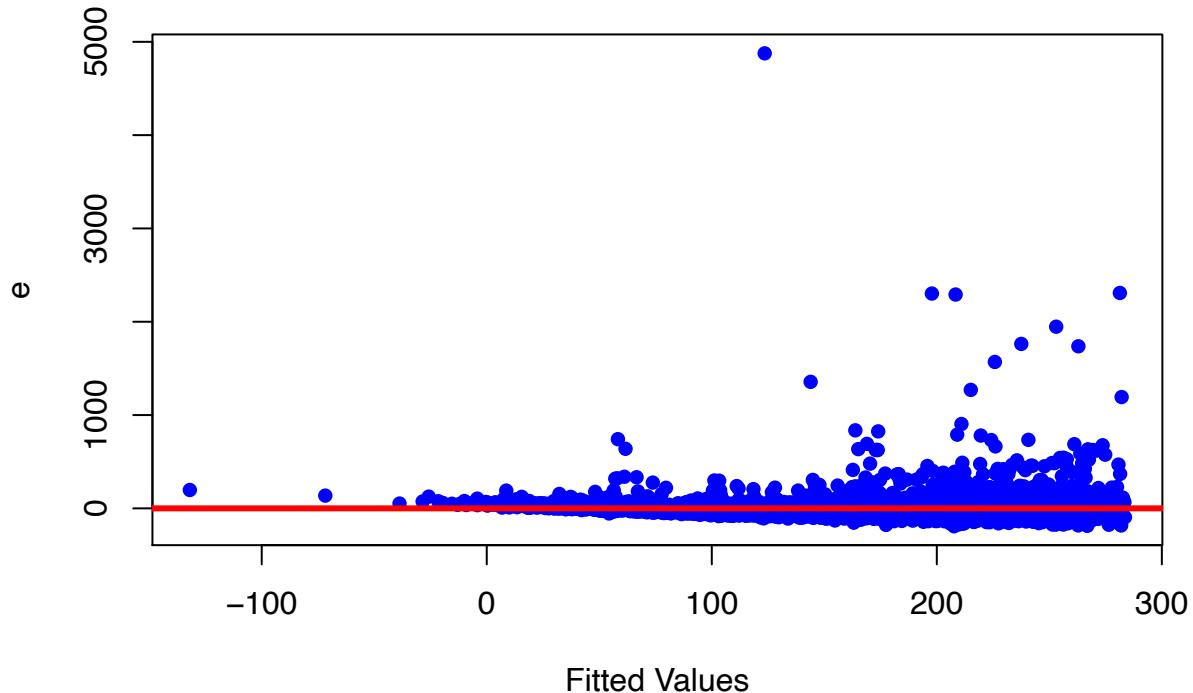


```

plot(fitted(full.lm),resid(full.lm), pch=16, col="blue", ylab=bquote(paste("e")),
      xlab=bquote(paste("Fitted Values")),main="Plot of e vs Fitted Values")
abline(0,0,col="red",lwd=3)

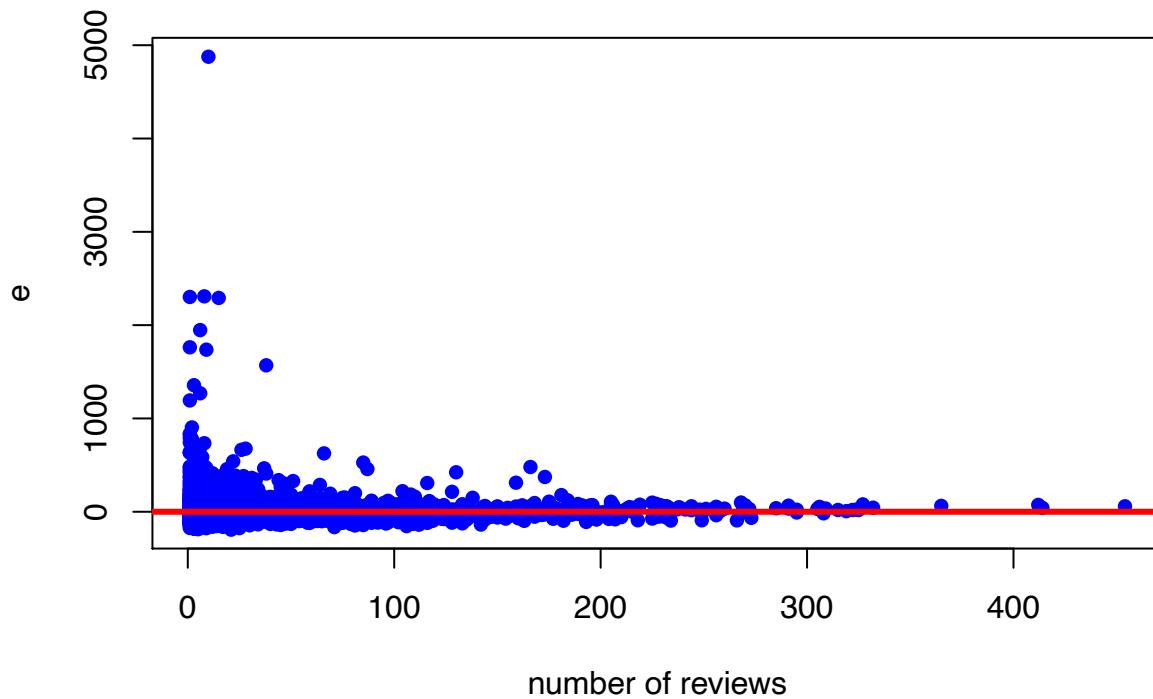
```

Plot of e vs Fitted Values



```
plot(number_of_reviews,resid(full.lm), pch=16, col="blue", ylab=bquote(paste("e")),
      xlab=bquote(paste("number of reviews")),main="Plot of e vs number of reviews")
abline(0,0,col="red",lwd=3)
```

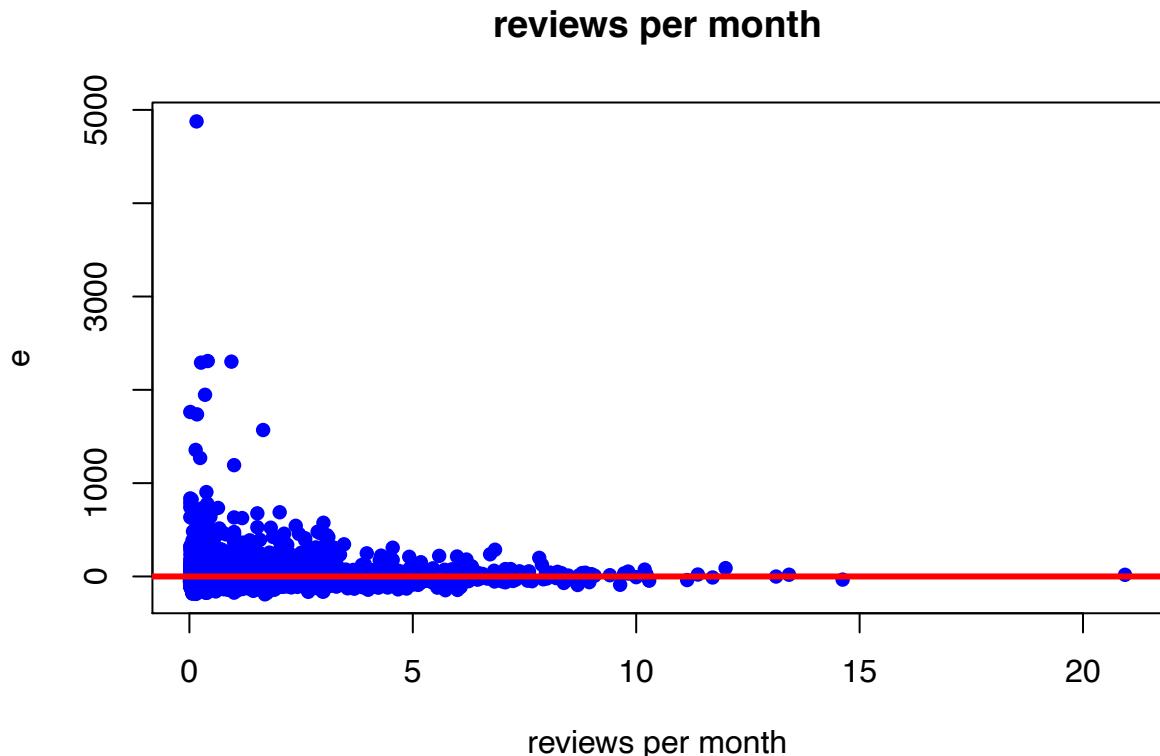
Plot of e vs number of reviews



```

plot(reviews_per_month,resid(full.lm), pch=16, col="blue", ylab=bquote(paste("e")),
      xlab=bquote(paste("reviews per month")),main="reviews per month")
abline(0,0,col="red",lwd=3)

```



Checking for outliers using Cutoff

```

k = 7
n = length(price)
LV_cutoff = 2*(k+1)/n; print(LV_cutoff)

## [1] 0.0032

Hi = hatvalues(full.lm)
Hi[Hi>LV_cutoff]

##          16         26         30         40         49         50
## 0.010753016 0.004153212 0.010749984 0.008998285 0.011855027 0.011723032
##          57         65         66         86        115        138
## 0.010682609 0.011701607 0.011991981 0.010812465 0.008766859 0.009661109
##         140        149        168        190        191        198
## 0.003744112 0.003958574 0.003635246 0.012624820 0.011036110 0.003337397
##         209        227        237        249        259        281
## 0.016259779 0.008775704 0.022787477 0.006426755 0.003780122 0.003293959
##         283        290        306        312        320        324
## 0.003522848 0.005000955 0.004528270 0.007173413 0.003675939 0.004450859
##         336        345        350        356        364        376
## 0.010522118 0.009150562 0.003360925 0.014317056 0.034947399 0.023028936
##         381        383        385        393        399        433
## 0.003775176 0.003776373 0.004502464 0.003469903 0.010639670 0.005229369
##         440        446        468        473        478        484

```

```

## 0.005125129 0.011692510 0.008354252 0.037237642 0.004451337 0.011036397
##      488      508      512      521      522      542
## 0.010596026 0.010421299 0.005098637 0.013089689 0.013618413 0.012099225
##      550      551      552      559      564      574
## 0.009472801 0.004081218 0.003591173 0.003674819 0.006879180 0.006414093
##      595      601      606      609      613      640
## 0.010579469 0.005896609 0.022723728 0.011136620 0.003770590 0.011444782
##      644      650      660      664      666      690
## 0.034713776 0.003601003 0.004677004 0.010186889 0.007491634 0.010193266
##      703      735      759      760      767      793
## 0.005803744 0.022894159 0.003647555 0.004861574 0.005198161 0.004479088
##      813      821      822      828      830      834
## 0.005360150 0.007232055 0.004747477 0.010706980 0.009923756 0.005440392
##      835      840      864      865      869      883
## 0.009921844 0.011203488 0.022890519 0.003917640 0.004933833 0.005106898
##      888      890      895      896      900      910
## 0.011403571 0.006257268 0.010040299 0.009866207 0.007450186 0.003911328
##      926      937      959      967      970      971
## 0.003227223 0.011678319 0.023582873 0.005989406 0.003237657 0.022518180
##      973      975      978      986      990      996
## 0.022668990 0.008894404 0.023233183 0.004236789 0.003632074 0.003776902
##      1008     1009     1015     1022     1027     1031
## 0.009558528 0.009792137 0.004738085 0.123161284 0.004380590 0.034918594
##      1064     1068     1071     1090     1102     1112
## 0.015349825 0.004533171 0.022640935 0.022530075 0.010735937 0.065608506
##      1122     1138     1143     1157     1158     1178
## 0.011282216 0.009860108 0.009449471 0.003351321 0.004235294 0.009776862
##      1182     1198     1203     1210     1213     1224
## 0.003823747 0.003397033 0.007370527 0.022650470 0.003522730 0.011032838
##      1231     1239     1246     1258     1268     1272
## 0.010483316 0.022674103 0.004203338 0.009694523 0.003789039 0.084444317
##      1287     1292     1294     1332     1345     1359
## 0.035345827 0.004613061 0.003924473 0.022995283 0.035352672 0.010779429
##      1363     1367     1394     1396     1409     1413
## 0.009954241 0.003431350 0.012508609 0.003932213 0.035447626 0.003348650
##      1418     1442     1445     1448     1458     1464
## 0.008282527 0.003908583 0.017792035 0.003266771 0.007259814 0.004082017
##      1479     1487     1503     1512     1516     1537
## 0.010087256 0.007125246 0.010886469 0.003877799 0.012101398 0.003803285
##      1539     1540     1557     1561     1572     1573
## 0.003642561 0.003659769 0.034869346 0.034701169 0.022930194 0.011700636
##      1586     1599     1600     1608     1634     1646
## 0.003427779 0.003887280 0.010017862 0.003397023 0.003334893 0.003834268
##      1648     1650     1663     1664     1679     1696
## 0.004313083 0.067734515 0.004224180 0.011015235 0.009840358 0.003826208
##      1697     1704     1712     1726     1735     1752
## 0.011084439 0.011967600 0.003349067 0.011186293 0.010124188 0.003460467
##      1753     1760     1763     1765     1766     1771
## 0.003758942 0.010693379 0.010922014 0.012228122 0.034890852 0.006186104
##      1779     1786     1794     1795     1797     1821
## 0.011681184 0.009503956 0.003426615 0.022983549 0.004283238 0.023848855
##      1824     1828     1836     1888     1901     1931
## 0.003496194 0.004406487 0.011902204 0.010927131 0.003905742 0.003285345
##      1932     1940     1968     1976     1983     1992

```

```

## 0.014331038 0.003763932 0.004389029 0.035000381 0.005297108 0.010078125
##      2001      2020      2026      2039      2046      2053
## 0.008833166 0.023114331 0.022606057 0.003297626 0.003925900 0.003415119
##      2064      2065      2074      2083      2110      2114
## 0.010558481 0.003686670 0.009764693 0.005434211 0.009583924 0.003323825
##      2134      2157      2161      2163      2170      2213
## 0.009314193 0.010644893 0.004868485 0.022885996 0.022788282 0.034925395
##      2224      2230      2233      2277      2289      2305
## 0.003825407 0.011993175 0.003682578 0.009841296 0.003319054 0.003483420
##      2320      2321      2330      2356      2366      2371
## 0.003581965 0.010693980 0.022760481 0.019417068 0.005449789 0.006987913
##      2373      2382      2389      2398      2399      2415
## 0.004035310 0.004369094 0.009851213 0.006098956 0.003221635 0.004171037
##      2420      2422      2427      2434      2442      2448
## 0.010250618 0.011008885 0.010627008 0.009724013 0.005053274 0.003495117
##      2449      2454      2462      2470      2479      2480
## 0.003762193 0.009672918 0.003200198 0.126888125 0.007495722 0.012370818
##      2491      2495      2496      2498      2499      2512
## 0.012228031 0.035253040 0.010000261 0.021150931 0.011749509 0.003237129
##      2551      2557      2559      2562      2565      2575
## 0.009781585 0.029753501 0.010919353 0.019445580 0.004522459 0.006263095
##      2576      2594      2597      2599      2600      2610
## 0.010187610 0.003728098 0.011596218 0.009913633 0.009682665 0.011126226
##      2616      2620      2630      2643      2649      2659
## 0.003248445 0.003596003 0.004592816 0.005778894 0.004558177 0.010458315
##      2663      2664      2668      2685      2688      2689
## 0.003523621 0.125512815 0.010028723 0.004921618 0.004070638 0.014772862
##      2693      2694      2695      2707      2721      2722
## 0.009554856 0.010661466 0.011172972 0.003860310 0.010123150 0.009696686
##      2733      2752      2763      2764      2765      2781
## 0.005168970 0.012684871 0.004408112 0.009988032 0.003567708 0.005838214
##      2785      2787      2794      2803      2830      2851
## 0.011131032 0.009984499 0.006707922 0.009683745 0.003721286 0.003692232
##      2863      2865      2869      2880      2905      2908
## 0.034891395 0.006791601 0.003311255 0.010055881 0.010708041 0.009940452
##      2909      2912      2913      2929      2934      2946
## 0.003982093 0.009897299 0.010095706 0.010082501 0.005290308 0.004025483
##      2987      2992      2994      3000      3001      3009
## 0.005221378 0.010683075 0.005363721 0.022829245 0.010221483 0.003405276
##      3021      3023      3036      3038      3039      3043
## 0.006098312 0.003850818 0.022889335 0.010253346 0.003422586 0.003800537
##      3050      3059      3064      3079      3084      3097
## 0.012234039 0.003291012 0.011120593 0.004777507 0.010706980 0.034856707
##      3117      3129      3140      3154      3162      3164
## 0.010625459 0.010693301 0.003516718 0.004471839 0.011099719 0.004675283
##      3172      3198      3206      3212      3216      3226
## 0.004537947 0.005292913 0.023010290 0.009213593 0.003897922 0.004402816
##      3231      3232      3234      3247      3251      3252
## 0.009776686 0.004939594 0.022752637 0.011560584 0.003432296 0.022683966
##      3277      3285      3298      3302      3308      3339
## 0.003622330 0.006856673 0.011557544 0.010518922 0.011082031 0.023281796
##      3355      3357      3362      3364      3367      3382
## 0.016664316 0.015163343 0.009628228 0.007141471 0.008429832 0.004378786
##      3394      3403      3405      3410      3416      3418

```

```

## 0.010918552 0.004053142 0.005158292 0.010676233 0.009687828 0.004122655
##      3419      3424      3428      3430      3435      3454
## 0.010010249 0.023051629 0.009569324 0.004356077 0.003737174 0.009929745
##      3465      3475      3484      3493      3503      3504
## 0.004387322 0.004771084 0.010738152 0.010299677 0.010082435 0.007247328
##      3511      3515      3523      3532      3546      3549
## 0.003902462 0.009640459 0.011510743 0.011093850 0.034950125 0.011680526
##      3555      3564      3566      3588      3598      3621
## 0.010481672 0.011268702 0.010880864 0.007961612 0.011526847 0.009750784
##      3629      3634      3640      3650      3653      3668
## 0.009935446 0.007570148 0.011743978 0.028615149 0.004393772 0.004496806
##      3705      3709      3714      3723      3728      3730
## 0.010800842 0.010085430 0.008261051 0.005649495 0.003892918 0.010022077
##      3733      3753      3755      3768      3773      3777
## 0.003284348 0.022580718 0.005206439 0.009800853 0.010111479 0.011734376
##      3791      3796      3797      3800      3804      3817
## 0.009480714 0.010275364 0.009967630 0.009582013 0.010159455 0.010458402
##      3828      3840      3871      3873      3874      3884
## 0.004036301 0.003432532 0.011804560 0.005633076 0.005220060 0.003850833
##      3898      3899      3906      3918      3921      3931
## 0.009860257 0.003298858 0.003610823 0.011606491 0.009823913 0.003708826
##      3933      3953      3959      3963      3970      3981
## 0.003649881 0.004593938 0.003415169 0.003574218 0.005129310 0.010231848
##      3984      3987      3989      3994      3995      3999
## 0.003929948 0.011569587 0.023025354 0.004336732 0.008044228 0.011550955
##      4000      4006      4013      4014      4015      4017
## 0.003288779 0.011880265 0.010631924 0.010491100 0.003804565 0.022929678
##      4020      4029      4032      4035      4042      4051
## 0.005826988 0.003239657 0.007275256 0.005442055 0.009955581 0.005435228
##      4052      4055      4070      4081      4085      4088
## 0.010139043 0.011066315 0.035587128 0.009892478 0.003307321 0.022830201
##      4100      4113      4114      4126      4128      4138
## 0.005865225 0.004680846 0.003627415 0.007582640 0.011151512 0.005447072
##      4147      4198      4205      4213      4226      4228
## 0.010151320 0.023597413 0.009741800 0.011590852 0.023281021 0.005653171
##      4233      4253      4265      4276      4285      4286
## 0.024498183 0.010301229 0.009913726 0.036172393 0.004220481 0.004297219
##      4287      4294      4308      4323      4351      4357
## 0.010099203 0.003805582 0.009913471 0.022604952 0.003210539 0.010004946
##      4361      4364      4365      4366      4369      4373
## 0.022781376 0.010472924 0.003915324 0.009943694 0.009526060 0.003420634
##      4382      4392      4404      4419      4436      4446
## 0.011763696 0.011167905 0.004131137 0.007448526 0.004268072 0.005356459
##      4453      4456      4467      4481      4486      4505
## 0.014312685 0.022908750 0.011933111 0.009764674 0.011208146 0.003970006
##      4506      4512      4517      4518      4531      4535
## 0.035088378 0.010720241 0.010968219 0.009476691 0.004154854 0.023048724
##      4543      4555      4559      4565      4582      4588
## 0.004141082 0.010665732 0.011138990 0.009024801 0.010333320 0.035447136
##      4593      4597      4601      4602      4603      4614
## 0.010693143 0.005811301 0.003242164 0.011052002 0.034904989 0.008304959
##      4616      4627      4629      4661      4676      4688
## 0.009960611 0.016028921 0.003206251 0.003538445 0.003627481 0.010926749
##      4693      4694      4698      4704      4711      4725

```

```

## 0.003747562 0.003511218 0.035181140 0.005416740 0.007033313 0.012617618
##      4737        4747        4748        4777        4785        4790
## 0.009991684 0.022609782 0.004630244 0.003653147 0.034857817 0.006794442
##      4795        4800        4803        4833        4836        4850
## 0.016600370 0.034928758 0.010031667 0.009950993 0.003812592 0.003633373
##      4857        4868        4881        4884        4890        4893
## 0.006200659 0.003320502 0.034918613 0.003836872 0.013329351 0.010007924
##      4894        4910        4923        4931        4934        4945
## 0.003900093 0.003746695 0.010910042 0.005466427 0.003237071 0.005379513
##      4952        4957        4958        4961        4965        4991
## 0.023580901 0.022743243 0.023712754 0.010706980 0.010738167 0.010831242
##      4993
## 0.035312272

```

These are the outliers.

Studentized Deleted Residuals (SDR)

```

Res_SE = summary(full.lm)$sigma
di = resid(full.lm)/(1-Hi)
SSE = Res_SE^2*(n-(k+1))
SDR = resid(full.lm)*sqrt((n-k-2)/(SSE*(1-Hi)-(resid(full.lm))^2))
SDR[abs(SDR)>qt(0.975,n-k-2)] #some evidence

```

```

##      21        41        57       112       202       206       292
## 3.840474 5.879217 1.970867 8.531874 3.255321 4.450823 2.257411
##      369        401        424        439        459        549       600
## 3.005770 3.858117 2.336190 1.999103 2.588921 2.714026 4.812918
##      604        611        677        751        790       843      1000
## 16.853226 2.857142 2.367642 4.404025 5.556061 4.542620 16.794963
##      1092       1161       1240       1295       1318       1360      1425
## 2.107011 3.226858 2.772334 9.088746 2.189291 2.290561 2.357198
##      1434       1451       1457       1472       1539       1583      1725
## 2.061002 2.044910 2.577996 3.748673 2.037627 3.142260 11.277178
##      1730       1732       1748       1780       1849       1943      2004
## 3.453233 4.451259 2.025062 1.979400 2.787585 4.449010 39.724870
##      2185       2259       2344       2362       2408       2444      2475
## 4.170508 2.188172 3.307926 3.437014 3.445016 5.209225 3.206000
##      2538       2613       2652       2661       2791       3028      3042
## 2.406145 2.286629 2.154374 2.038190 4.091448 2.909875 4.515425
##      3066       3091       3119       3180       3182       3222      3279
## 5.955942 2.166938 4.724410 6.442902 4.894943 3.410965 2.751256
##      3300       3341       3489       3502       3560       3564      3614
## 3.102705 3.243999 5.225723 2.651402 2.214295 5.306768 2.988102
##      3630       3715       3717       3760       3830       3863      3997
## 2.082725 2.618805 3.335943 2.968248 2.058708 3.014280 2.928838
##      4007       4139       4162       4252       4264       4271      4318
## 2.655966 4.501436 2.754245 3.713762 16.716136 4.916308 4.455322
##      4319       4328       4355       4396       4409       4450      4486
## 2.446670 14.091333 2.594594 3.472396 9.716556 2.403656 2.132986
##      4496       4540       4552       4572       4589       4677      4738
## 12.716730 3.657422 2.605854 2.608957 2.040661 3.381107 2.762539
##      4751       4816       4821       4873       4907       4948
## 12.528049 2.562962 5.624146 2.348138 2.167482 2.153366

```

```
SDR[abs(SDR)>qt(0.995,n-k-2)] #strong evidence
```

```
##      21       41      112      202      206      369      401
## 3.840474 5.879217 8.531874 3.255321 4.450823 3.005770 3.858117
##      459      549      600      604      611      751      790
## 2.588921 2.714026 4.812918 16.853226 2.857142 4.404025 5.556061
##      843     1000     1161     1240     1295     1457     1472
## 4.542620 16.794963 3.226858 2.772334 9.088746 2.577996 3.748673
##     1583     1725     1730     1732     1849     1943     2004
## 3.142260 11.277178 3.453233 4.451259 2.787585 4.449010 39.724870
##     2185     2344     2362     2408     2444     2475     2791
## 4.170508 3.307926 3.437014 3.445016 5.209225 3.206000 4.091448
##     3028     3042     3066     3119     3180     3182     3222
## 2.909875 4.515425 5.955942 4.724410 6.442902 4.894943 3.410965
##     3279     3300     3341     3489     3502     3564     3614
## 2.751256 3.102705 3.243999 5.225723 2.651402 5.306768 2.988102
##     3715     3717     3760     3863     3997     4007     4139
## 2.618805 3.335943 2.968248 3.014280 2.928838 2.655966 4.501436
##     4162     4252     4264     4271     4318     4328     4355
## 2.754245 3.713762 16.716136 4.916308 4.455322 14.091333 2.594594
##     4396     4409     4496     4540     4552     4572     4677
## 3.472396 9.716556 12.716730 3.657422 2.605854 2.608957 3.381107
##     4738     4751     4821
## 2.762539 12.528049 5.624146
```

This test outputs a list of data points that show some and strong evidence of being an outlier with respect to its y value.

Cook's Distance Measure

```
CooksD = cooks.distance(full.lm)
CooksD[CooksD>qf(0.5,k+1, n-k-1)] #Influential
```

```
## named numeric(0)
#CooksD[CooksD<qf(0.2,k+1, n-k-1)] #Not Influential
CooksD[CooksD<qf(0.5,k+1, n-k-1) & CooksD>qf(0.2,k+1, n-k-1)]
```

```
## named numeric(0)
```

This test outputs a list of data points that show influential/non-influential evidence of being an outlier.

Stepwise selection of variables from BASE model

```
min.lm = lm(price ~1, data = project_data_sample)

step_both = step(min.lm, list(upper = full.lm), direction = 'both')

## Start: AIC=50512.86
## price ~ 1
##
##          Df Sum of Sq      RSS      AIC
## + private_room_dummy      1  14865376 107114397 49865
## + queens_dummy            1  1596999 120382774 50449
## + availability_365        1  1517286 120462487 50452
```

```

## + brooklyn_dummy 1 1331938 120647835 50460
## + shared_room_dummy 1 571494 121408279 50491
## + calculated_host_listings_count 1 506649 121473124 50494
## + bronx_dummy 1 328283 121651490 50501
## + number_of_reviews 1 224241 121755532 50506
## + reviews_per_month 1 214128 121765645 50506
## + staten_dummy 1 191083 121788690 50507
## <none> 121979773 50513
## + minimum_nights 1 95 121979678 50515
##
## Step: AIC=49865.07
## price ~ private_room_dummy
##
## Df Sum of Sq RSS AIC
## + availability_365 1 1670550 105443847 49788
## + shared_room_dummy 1 1606675 105507723 49792
## + brooklyn_dummy 1 993491 106120906 49820
## + queens_dummy 1 649257 106465140 49837
## + bronx_dummy 1 206735 106907662 49857
## + calculated_host_listings_count 1 201270 106913127 49858
## + number_of_reviews 1 184344 106930054 49858
## + staten_dummy 1 155207 106959190 49860
## + minimum_nights 1 83991 107030406 49863
## + reviews_per_month 1 72624 107041774 49864
## <none> 107114397 49865
## - private_room_dummy 1 14865376 121979773 50513
##
## Step: AIC=49788.48
## price ~ private_room_dummy + availability_365
##
## Df Sum of Sq RSS AIC
## + shared_room_dummy 1 1834033 103609814 49703
## + queens_dummy 1 904942 104538906 49747
## + brooklyn_dummy 1 851172 104592675 49750
## + number_of_reviews 1 493415 104950432 49767
## + bronx_dummy 1 301774 105142073 49776
## + reviews_per_month 1 265335 105178512 49778
## + staten_dummy 1 239026 105204821 49779
## + minimum_nights 1 153334 105290513 49783
## + calculated_host_listings_count 1 44569 105399278 49788
## <none> 105443847 49788
## - availability_365 1 1670550 107114397 49865
## - private_room_dummy 1 15018640 120462487 50452
##
## Step: AIC=49702.75
## price ~ private_room_dummy + availability_365 + shared_room_dummy
##
## Df Sum of Sq RSS AIC
## + brooklyn_dummy 1 893513 102716301 49661
## + queens_dummy 1 779753 102830061 49667
## + number_of_reviews 1 507281 103102533 49680
## + bronx_dummy 1 305131 103304683 49690
## + reviews_per_month 1 262215 103347599 49692
## + staten_dummy 1 241305 103368509 49693

```

```

## + minimum_nights           1   182313 103427501 49696
## <none>                      103609814 49703
## + calculated_host_listings_count 1   34743 103575071 49703
## - shared_room_dummy          1   1834033 105443847 49788
## - availability_365            1   1897908 105507723 49792
## - private_room_dummy          1   16156589 119766403 50425
##
## Step: AIC=49661.44
## price ~ private_room_dummy + availability_365 + shared_room_dummy +
##       brooklyn_dummy
##
##                                     Df Sum of Sq      RSS      AIC
## + queens_dummy                  1   1559894 101156408 49587
## + number_of_reviews              1   507137 102209164 49639
## + bronx_dummy                   1   458788 102257513 49641
## + reviews_per_month              1   330473 102385828 49647
## + staten_dummy                  1   322454 102393847 49648
## + minimum_nights                 1   209596 102506705 49653
## <none>                          102716301 49661
## + calculated_host_listings_count 1   13202 102703100 49663
## - brooklyn_dummy                1   893513 103609814 49703
## - availability_365              1   1745332 104461633 49744
## - shared_room_dummy              1   1876374 104592675 49750
## - private_room_dummy             1   15833013 118549314 50376
##
## Step: AIC=49586.93
## price ~ private_room_dummy + availability_365 + shared_room_dummy +
##       brooklyn_dummy + queens_dummy
##
##                                     Df Sum of Sq      RSS      AIC
## + bronx_dummy                   1   679912 100476496 49555
## + number_of_reviews              1   481001 100675406 49565
## + staten_dummy                  1   437947 100718461 49567
## + minimum_nights                 1   282382 100874026 49575
## + reviews_per_month              1   195950 100960457 49579
## <none>                          101156408 49587
## + calculated_host_listings_count 1   1274 101155133 49589
## - queens_dummy                  1   1559894 102716301 49661
## - brooklyn_dummy                1   1673654 102830061 49667
## - shared_room_dummy              1   1709934 102866342 49669
## - availability_365              1   2044232 103200640 49685
## - private_room_dummy             1   14113067 115269474 50238
##
## Step: AIC=49555.21
## price ~ private_room_dummy + availability_365 + shared_room_dummy +
##       brooklyn_dummy + queens_dummy + bronx_dummy
##
##                                     Df Sum of Sq      RSS      AIC
## + staten_dummy                  1   482195 99994301 49533
## + number_of_reviews              1   466743 100009753 49534
## + minimum_nights                 1   306198 100170298 49542
## + reviews_per_month              1   167672 100308824 49549
## <none>                          100476496 49555
## + calculated_host_listings_count 1   6 100476490 49557

```

```

## - bronx_dummy 1 679912 101156408 49587
## - shared_room_dummy 1 1707491 102183987 49637
## - queens_dummy 1 1781017 102257513 49641
## - brooklyn_dummy 1 1978364 102454860 49651
## - availability_365 1 2214704 102691200 49662
## - private_room_dummy 1 13764783 114241280 50195
##
## Step: AIC=49533.15
## price ~ private_room_dummy + availability_365 + shared_room_dummy +
##      brooklyn_dummy + queens_dummy + bronx_dummy + staten_dummy
##
##                                     Df Sum of Sq      RSS      AIC
## + number_of_reviews 1 464098 99530204 49512
## + minimum_nights 1 331906 99662396 49519
## + reviews_per_month 1 160669 99833632 49527
## <none> 99994301 49533
## + calculated_host_listings_count 1 780 99993522 49535
## - staten_dummy 1 482195 100476496 49555
## - bronx_dummy 1 724159 100718461 49567
## - shared_room_dummy 1 1706433 101700735 49616
## - queens_dummy 1 1916797 101911099 49626
## - brooklyn_dummy 1 2163311 102157612 49638
## - availability_365 1 2363787 102358088 49648
## - private_room_dummy 1 13624902 113619203 50170
##
## Step: AIC=49511.89
## price ~ private_room_dummy + availability_365 + shared_room_dummy +
##      brooklyn_dummy + queens_dummy + bronx_dummy + staten_dummy +
##      number_of_reviews
##
##                                     Df Sum of Sq      RSS      AIC
## + minimum_nights 1 418815 99111389 49493
## <none> 99530204 49512
## + calculated_host_listings_count 1 9067 99521136 49513
## + reviews_per_month 1 3916 99526287 49514
## - number_of_reviews 1 464098 99994301 49533
## - staten_dummy 1 479550 100009753 49534
## - bronx_dummy 1 709363 100239567 49545
## - shared_room_dummy 1 1720512 101250716 49596
## - queens_dummy 1 1885368 101415572 49604
## - brooklyn_dummy 1 2149149 101679352 49617
## - availability_365 1 2693086 102223290 49643
## - private_room_dummy 1 13601976 113132180 50150
##
## Step: AIC=49492.81
## price ~ private_room_dummy + availability_365 + shared_room_dummy +
##      brooklyn_dummy + queens_dummy + bronx_dummy + staten_dummy +
##      number_of_reviews + minimum_nights
##
##                                     Df Sum of Sq      RSS      AIC
## <none> 99111389 49493
## + reviews_per_month 1 20183 99091206 49494
## + calculated_host_listings_count 1 4649 99106740 49495
## - minimum_nights 1 418815 99530204 49512

```

```

## - staten_dummy           1   508356  99619745 49516
## - number_of_reviews      1   551007  99662396 49519
## - bronx_dummy            1   738046  99849435 49528
## - shared_room_dummy      1   1761571 100872960 49579
## - queens_dummy            1   1989449 101100838 49590
## - brooklyn_dummy          1   2248835 101360224 49603
## - availability_365        1   2889215 102000604 49634
## - private_room_dummy       1   13867786 112979175 50146

```

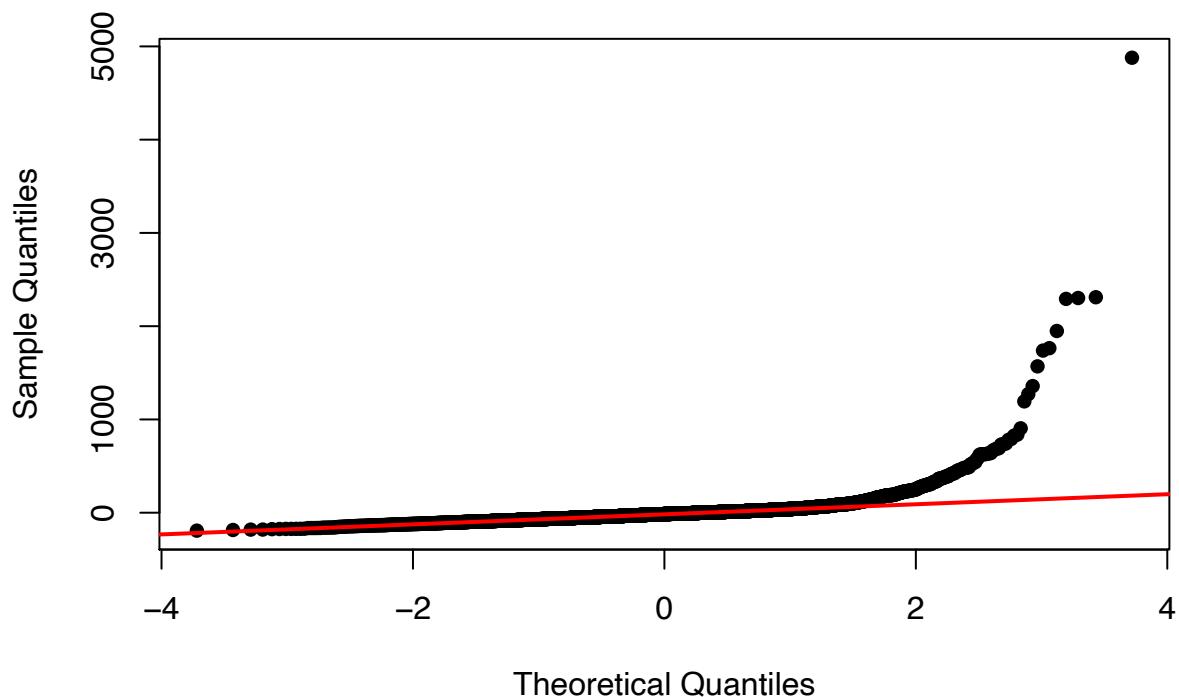
Best model with dummies (location, roomtype) = price ~ private_room_dummy + availability_365 + shared_room_dummy + brooklyn_dummy + bronx_dummy + number_of_reviews + minimum_nights + queens_dummy + staten_dummy

```

newbestmodel= lm(price ~ private_room_dummy + availability_365 +
                  shared_room_dummy + brooklyn_dummy + bronx_dummy +
                  number_of_reviews + minimum_nights + queens_dummy +
                  staten_dummy)
qqnorm(resid(newbestmodel),pch=16,main="Normal Probability Plot of Residuals")
qqline(resid(newbestmodel),col='red',lwd=2)

```

Normal Probability Plot of Residuals



```
summary(newbestmodel)
```

```

##
## Call:
## lm(formula = price ~ private_room_dummy + availability_365 +
##     shared_room_dummy + brooklyn_dummy + bronx_dummy + number_of_reviews +
##     minimum_nights + queens_dummy + staten_dummy)
##
## Residuals:
##    Min     1Q Median     3Q    Max 
## -192.3 -53.6 -15.0  19.1 4878.4

```

```

## 
## Coefficients:
##                               Estimate Std. Error t value Pr(>|t|)    
## (Intercept)            212.55929   4.05706  52.392 < 2e-16 ***
## private_room_dummy    -108.39142   4.10207 -26.424 < 2e-16 ***
## availability_365      0.19200   0.01592  12.061 < 2e-16 ***
## shared_room_dummy     -135.72689   14.41210 -9.418 < 2e-16 ***
## brooklyn_dummy        -46.21379   4.34315 -10.641 < 2e-16 ***
## bronx_dummy           -85.16244   13.97069 -6.096 1.17e-09 ***
## number_of_reviews      -0.22470   0.04266 -5.267 1.44e-07 ***
## minimum_nights         -0.63347   0.13795 -4.592 4.50e-06 ***
## queens_dummy           -67.19520   6.71403 -10.008 < 2e-16 ***
## staten_dummy          -107.82482  21.31308 -5.059 4.36e-07 ***
## --- 
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## 
## Residual standard error: 140.9 on 4990 degrees of freedom
## Multiple R-squared:  0.1875, Adjusted R-squared:  0.186 
## F-statistic: 127.9 on 9 and 4990 DF,  p-value: < 2.2e-16

```

The above plot is for the best model with dummies

Checking for outliers using Cutoff

```

k = 7
n = length(price)
LV_cutoff = 2*(k+1)/n; print(LV_cutoff)

## [1] 0.0032

Hi = hatvalues(newbestmodel)
Hi[Hi>LV_cutoff]

##      16       26       30       40       49       50
## 0.010295099 0.003580248 0.010708091 0.008801343 0.010958321 0.011506680
##      57       65       66       86      138      140
## 0.010622078 0.011546727 0.011962369 0.010800933 0.009654897 0.003525011
##     149      168      190      191      209      237
## 0.003840322 0.003497515 0.012562061 0.011022621 0.016238858 0.022664212
##     249      283      290      306      336      350
## 0.005386329 0.003472463 0.004514577 0.004395417 0.010420508 0.003315117
##     376      381      393      399      440      446
## 0.022859433 0.003746894 0.003430099 0.010455347 0.004509451 0.011654737
##     484      488      508      512      521      522
## 0.010839399 0.010584059 0.010314380 0.005045478 0.010642129 0.013532977
##     542      550      551      559      564      574
## 0.011858203 0.009470571 0.004066207 0.003459526 0.006741544 0.005822696
##     595      606      609      640      664      666
## 0.010551873 0.022697295 0.010957079 0.011430447 0.010153258 0.006195028
##     690      703      735      760      793      821
## 0.009773882 0.005657417 0.022829862 0.004583740 0.004329470 0.006205029
##     822      828      830      835      840      864
## 0.004272088 0.010653301 0.009853165 0.009711298 0.010478385 0.022766622
##     869      888      895      896      900      910
## 0.004911291 0.011111108 0.009843056 0.009842786 0.007332060 0.003731103

```

| | 937 | 959 | 967 | 971 | 973 | 975 |
|----|-------------|-------------|-------------|-------------|-------------|-------------|
| ## | 0.011630582 | 0.022916760 | 0.004098918 | 0.022514552 | 0.022663327 | 0.008299023 |
| ## | 978 | 990 | 1008 | 1009 | 1015 | 1022 |
| ## | 0.023166159 | 0.003480826 | 0.009554413 | 0.009756069 | 0.004138170 | 0.121442438 |
| ## | 1068 | 1071 | 1090 | 1102 | 1112 | 1122 |
| ## | 0.004414267 | 0.022431218 | 0.022468100 | 0.010696497 | 0.064895379 | 0.011264908 |
| ## | 1138 | 1143 | 1158 | 1178 | 1182 | 1203 |
| ## | 0.009856390 | 0.009447411 | 0.003773607 | 0.009742422 | 0.003457355 | 0.007308783 |
| ## | 1210 | 1224 | 1231 | 1239 | 1258 | 1272 |
| ## | 0.022645861 | 0.010402877 | 0.009593017 | 0.022591634 | 0.009570620 | 0.083468832 |
| ## | 1292 | 1294 | 1332 | 1359 | 1363 | 1367 |
| ## | 0.003973433 | 0.003544460 | 0.022965213 | 0.010609887 | 0.009944586 | 0.003228448 |
| ## | 1394 | 1396 | 1413 | 1418 | 1458 | 1479 |
| ## | 0.010577998 | 0.003307723 | 0.003323726 | 0.008155576 | 0.007099998 | 0.010082193 |
| ## | 1487 | 1503 | 1512 | 1516 | 1572 | 1573 |
| ## | 0.007048381 | 0.010781020 | 0.003840108 | 0.011818130 | 0.022654448 | 0.011688052 |
| ## | 1600 | 1608 | 1646 | 1650 | 1664 | 1679 |
| ## | 0.009934889 | 0.003380397 | 0.003530683 | 0.066920314 | 0.010911278 | 0.009819500 |
| ## | 1697 | 1704 | 1726 | 1735 | 1760 | 1763 |
| ## | 0.009864161 | 0.011732147 | 0.011009082 | 0.009930212 | 0.010644413 | 0.010851518 |
| ## | 1765 | 1771 | 1779 | 1795 | 1821 | 1828 |
| ## | 0.009511177 | 0.005429146 | 0.009993707 | 0.022946301 | 0.022494449 | 0.004114601 |
| ## | 1836 | 1888 | 1901 | 1931 | 1932 | 1983 |
| ## | 0.011751257 | 0.010750965 | 0.003662231 | 0.003270666 | 0.014078821 | 0.004682932 |
| ## | 1992 | 2001 | 2020 | 2026 | 2039 | 2064 |
| ## | 0.009962258 | 0.008780184 | 0.023021315 | 0.022590153 | 0.003283511 | 0.010550276 |
| ## | 2074 | 2110 | 2134 | 2157 | 2163 | 2170 |
| ## | 0.009607951 | 0.009560068 | 0.009097691 | 0.010610993 | 0.022703120 | 0.022735659 |
| ## | 2230 | 2233 | 2277 | 2289 | 2320 | 2321 |
| ## | 0.011747510 | 0.003580589 | 0.009801758 | 0.003238564 | 0.003246061 | 0.010449365 |
| ## | 2330 | 2356 | 2366 | 2371 | 2373 | 2389 |
| ## | 0.022680565 | 0.019321319 | 0.005370212 | 0.005967216 | 0.003307317 | 0.009721691 |
| ## | 2420 | 2422 | 2427 | 2434 | 2442 | 2448 |
| ## | 0.010146758 | 0.010853376 | 0.010564146 | 0.009564968 | 0.004389631 | 0.003213941 |
| ## | 2454 | 2470 | 2480 | 2491 | 2496 | 2498 |
| ## | 0.009644958 | 0.124941356 | 0.010915675 | 0.012205170 | 0.009630020 | 0.021025304 |
| ## | 2499 | 2551 | 2557 | 2559 | 2562 | 2565 |
| ## | 0.009602797 | 0.009748085 | 0.029370522 | 0.010438569 | 0.019404967 | 0.004503662 |
| ## | 2576 | 2597 | 2599 | 2600 | 2610 | 2630 |
| ## | 0.009987161 | 0.011444471 | 0.009855128 | 0.009611969 | 0.010420672 | 0.004454952 |
| ## | 2659 | 2664 | 2668 | 2685 | 2688 | 2689 |
| ## | 0.010445793 | 0.123670142 | 0.009919850 | 0.004862202 | 0.003928554 | 0.013859287 |
| ## | 2693 | 2694 | 2695 | 2721 | 2722 | 2733 |
| ## | 0.009504615 | 0.010637174 | 0.011064230 | 0.010021540 | 0.009564772 | 0.004891647 |
| ## | 2763 | 2764 | 2785 | 2787 | 2794 | 2803 |
| ## | 0.003920231 | 0.009828888 | 0.010840750 | 0.009824247 | 0.005678595 | 0.009673527 |
| ## | 2830 | 2851 | 2865 | 2880 | 2905 | 2908 |
| ## | 0.003272698 | 0.003567350 | 0.006633250 | 0.009905950 | 0.010589388 | 0.009894792 |
| ## | 2909 | 2912 | 2913 | 2929 | 2946 | 2992 |
| ## | 0.003553916 | 0.009842659 | 0.010068043 | 0.009581903 | 0.003450845 | 0.010641817 |
| ## | 2994 | 3000 | 3001 | 3021 | 3023 | 3036 |
| ## | 0.004196322 | 0.022630023 | 0.010147569 | 0.005911329 | 0.003838247 | 0.022730526 |
| ## | 3038 | 3039 | 3050 | 3064 | 3079 | 3084 |
| ## | 0.009558404 | 0.003410303 | 0.011863699 | 0.010977942 | 0.004726713 | 0.010653301 |

```

##      3097      3117      3129      3154      3162      3164
## 0.032823822 0.010620107 0.010643939 0.004437352 0.011060748 0.004485908
##      3172      3198      3206      3212      3231      3234
## 0.004117100 0.004368017 0.022981744 0.008722026 0.009743956 0.022616399
##      3247      3252      3285      3298      3302      3308
## 0.011506568 0.022679324 0.006263807 0.011432548 0.010513401 0.010891318
##      3339      3355      3357      3362      3364      3367
## 0.022640597 0.016441186 0.014819034 0.009590828 0.006728899 0.007825808
##      3394      3405      3410      3416      3419      3424
## 0.010857225 0.004378259 0.010619964 0.009672650 0.009865656 0.023044455
##      3428      3430      3454      3475      3484      3493
## 0.009499726 0.004125753 0.009868733 0.004070367 0.010658189 0.010095227
##      3503      3504      3515      3523      3532      3549
## 0.009962258 0.007092474 0.009549090 0.011490444 0.010868122 0.011667007
##      3555      3564      3566      3588      3598      3621
## 0.010480829 0.011249343 0.010466709 0.007066860 0.011514995 0.009710331
##      3629      3634      3650      3653      3668      3705
## 0.009896001 0.007327116 0.026675471 0.004345575 0.004233035 0.010714082
##      3709      3714      3723      3728      3730      3733
## 0.010005272 0.007797415 0.005359558 0.003578373 0.009865656 0.003238431
##      3753      3768      3773      3777      3791      3796
## 0.022533415 0.009782794 0.009878379 0.011528866 0.008337898 0.010059935
##      3797      3800      3804      3817      3871      3873
## 0.009954379 0.009556631 0.010099366 0.010457045 0.011647689 0.005571411
##      3884      3898      3906      3918      3921      3953
## 0.003837472 0.009647144 0.003548395 0.011600822 0.009791340 0.003815292
##      3970      3981      3987      3989      3994      3995
## 0.004458836 0.010024995 0.011519160 0.022984550 0.004235562 0.008030979
##      3999      4006      4013      4014      4017      4020
## 0.010963828 0.011735352 0.010627645 0.010465608 0.022743248 0.005729851
##      4042      4052      4055      4081      4088      4100
## 0.009852242 0.010093333 0.011009047 0.009838037 0.022689492 0.005857919
##      4126      4128      4138      4147      4198      4205
## 0.007039947 0.010973915 0.004896825 0.010014529 0.023576812 0.009615079
##      4213      4226      4228      4233      4253      4265
## 0.011525167 0.023279327 0.005635443 0.022699229 0.010243212 0.009784257
##      4287      4308      4323      4357      4361      4364
## 0.009891446 0.009828521 0.022542922 0.009965049 0.022710005 0.009692862
##      4366      4369      4382      4392      4419      4436
## 0.009908615 0.009494837 0.011757319 0.010987074 0.007415582 0.003919361
##      4453      4456      4467      4481      4486      4512
## 0.009558403 0.022709406 0.011787942 0.009639946 0.011082241 0.010470174
##      4517      4518      4531      4535      4555      4559
## 0.010621190 0.008552161 0.004038480 0.022993509 0.010076738 0.010961942
##      4565      4582      4593      4602      4614      4616
## 0.008258742 0.010090721 0.010680279 0.010567583 0.007246476 0.009859232
##      4627      4693      4725      4737      4747      4748
## 0.015175585 0.003721506 0.012543245 0.009863520 0.022559979 0.004319770
##      4777      4795      4803      4833      4857      4890
## 0.003396378 0.016600238 0.009953749 0.009863026 0.005632152 0.013061928
##      4893      4894      4910      4923      4945      4952
## 0.009932485 0.003425796 0.003657655 0.010788923 0.005128471 0.023544421
##      4957      4958      4961      4965      4991
## 0.022724764 0.023694048 0.010653301 0.010705755 0.010589198

```

These are the outliers.

Studentized Deleted Residuals (SDR)

```
Res_SE = summary(newbestmodel)$sigma
di = resid(newbestmodel)/(1-Hi)
SSE = Res_SE^2*(n-(k+1))
SDR = resid(newbestmodel)*sqrt((n-k-2)/(SSE*(1-Hi)-(resid(newbestmodel))^2))
SDR[abs(SDR)>qt(0.975,n-k-2)] #some evidence
```

| | | | | | | | |
|----|-----------|-----------|----------|----------|-----------|----------|-----------|
| ## | 21 | 41 | 57 | 112 | 202 | 206 | 292 |
| ## | 3.853657 | 5.892351 | 1.979116 | 8.538694 | 3.262445 | 4.463722 | 2.263725 |
| ## | 369 | 401 | 424 | 439 | 459 | 549 | 600 |
| ## | 3.009309 | 3.848844 | 2.347303 | 2.008181 | 2.592684 | 2.709101 | 4.818513 |
| ## | 604 | 611 | 677 | 751 | 790 | 843 | 1000 |
| ## | 16.867655 | 2.873079 | 2.365211 | 4.415477 | 5.561931 | 4.553753 | 16.796701 |
| ## | 1092 | 1161 | 1240 | 1295 | 1318 | 1360 | 1425 |
| ## | 2.114810 | 3.226870 | 2.781198 | 9.099605 | 2.160651 | 2.299968 | 2.365744 |
| ## | 1434 | 1451 | 1457 | 1472 | 1539 | 1583 | 1725 |
| ## | 2.069323 | 2.057379 | 2.588667 | 3.762961 | 1.983599 | 3.126510 | 11.278966 |
| ## | 1730 | 1732 | 1748 | 1780 | 1849 | 1943 | 2004 |
| ## | 3.448920 | 4.466615 | 2.035191 | 1.976892 | 2.791414 | 4.462136 | 39.745394 |
| ## | 2185 | 2259 | 2344 | 2362 | 2408 | 2444 | 2475 |
| ## | 4.182580 | 2.183472 | 3.296828 | 3.445494 | 3.438025 | 5.202624 | 3.197826 |
| ## | 2538 | 2613 | 2652 | 2661 | 2791 | 3028 | 3042 |
| ## | 2.398785 | 2.293178 | 2.167217 | 2.047135 | 4.077212 | 2.902987 | 4.521918 |
| ## | 3066 | 3091 | 3119 | 3180 | 3182 | 3222 | 3279 |
| ## | 5.962354 | 2.170818 | 4.738610 | 6.448606 | 4.890060 | 3.420201 | 2.756332 |
| ## | 3300 | 3341 | 3489 | 3502 | 3560 | 3564 | 3614 |
| ## | 3.094489 | 3.221710 | 5.232924 | 2.669620 | 2.220639 | 5.312045 | 2.981829 |
| ## | 3630 | 3715 | 3717 | 3760 | 3830 | 3863 | 3997 |
| ## | 2.090807 | 2.631627 | 3.351504 | 2.972100 | 2.060612 | 3.008673 | 2.935692 |
| ## | 4007 | 4139 | 4162 | 4252 | 4264 | 4271 | 4318 |
| ## | 2.664286 | 4.506514 | 2.762643 | 3.709251 | 16.725313 | 4.919260 | 4.468221 |
| ## | 4319 | 4328 | 4355 | 4396 | 4409 | 4450 | 4486 |
| ## | 2.431705 | 14.102615 | 2.599374 | 3.481171 | 9.728200 | 2.411823 | 2.145445 |
| ## | 4496 | 4540 | 4552 | 4572 | 4589 | 4677 | 4738 |
| ## | 12.731293 | 3.665871 | 2.612446 | 2.612771 | 2.049298 | 3.383142 | 2.753379 |
| ## | 4751 | 4816 | 4821 | 4873 | 4907 | 4948 | |
| ## | 12.539492 | 2.551592 | 5.632746 | 2.354783 | 2.163355 | 2.140510 | |

```
SDR[abs(SDR)>qt(0.995,n-k-2)] #strong evidence
```

| | | | | | | | |
|----|----------|-----------|----------|-----------|----------|----------|-----------|
| ## | 21 | 41 | 112 | 202 | 206 | 369 | 401 |
| ## | 3.853657 | 5.892351 | 8.538694 | 3.262445 | 4.463722 | 3.009309 | 3.848844 |
| ## | 459 | 549 | 600 | 604 | 611 | 751 | 790 |
| ## | 2.592684 | 2.709101 | 4.818513 | 16.867655 | 2.873079 | 4.415477 | 5.561931 |
| ## | 843 | 1000 | 1161 | 1240 | 1295 | 1457 | 1472 |
| ## | 4.553753 | 16.796701 | 3.226870 | 2.781198 | 9.099605 | 2.588667 | 3.762961 |
| ## | 1583 | 1725 | 1730 | 1732 | 1849 | 1943 | 2004 |
| ## | 3.126510 | 11.278966 | 3.448920 | 4.466615 | 2.791414 | 4.462136 | 39.745394 |
| ## | 2185 | 2344 | 2362 | 2408 | 2444 | 2475 | 2791 |
| ## | 4.182580 | 3.296828 | 3.445494 | 3.438025 | 5.202624 | 3.197826 | 4.077212 |
| ## | 3028 | 3042 | 3066 | 3119 | 3180 | 3182 | 3222 |
| ## | 2.902987 | 4.521918 | 5.962354 | 4.738610 | 6.448606 | 4.890060 | 3.420201 |

```

##      3279      3300      3341      3489      3502      3564      3614
##  2.756332  3.094489  3.221710  5.232924  2.669620  5.312045  2.981829
##      3715      3717      3760      3863      3997      4007      4139
##  2.631627  3.351504  2.972100  3.008673  2.935692  2.664286  4.506514
##      4162      4252      4264      4271      4318      4328      4355
##  2.762643  3.709251 16.725313  4.919260  4.468221 14.102615  2.599374
##      4396      4409      4496      4540      4552      4572      4677
##  3.481171  9.728200 12.731293  3.665871  2.612446  2.612771  3.383142
##      4738      4751      4821
##  2.753379 12.539492  5.632746

```

This test outputs a list of data points that show some and strong evidence of being an outlier with respect to its y value.

Cook's Distance Measure

```

CooksD = cooks.distance(newbestmodel)
CooksD[CooksD>=qf(0.5,k+1, n-k-1)] #Influential

## named numeric(0)
#CooksD[CooksD<=qf(0.2,k+1, n-k-1)] #Not Influential
CooksD[CooksD<=qf(0.5,k+1, n-k-1) & CooksD>=qf(0.2,k+1, n-k-1)] 

## named numeric(0)

```

This test outputs a list of data points that show influential/non-influential evidence of being an outlier.

DATA EXPLORATION PLOTS

Load packages:

```

library(ggmap)
library(dplyr)
library("ggcorrplot")
library(gridExtra)
#library(ggplot)
library(plotrix)
register_google([REDACTED])

```

Custom Colors:

```

pink = "#f547c1"
orange = "#f59847"
yellow = "#f5de47"
green = "#3dcc64"
blue = "#4fc7ff"
customcolors = c(pink,orange,green,blue,yellow)

```

Before cutting down the data:

```

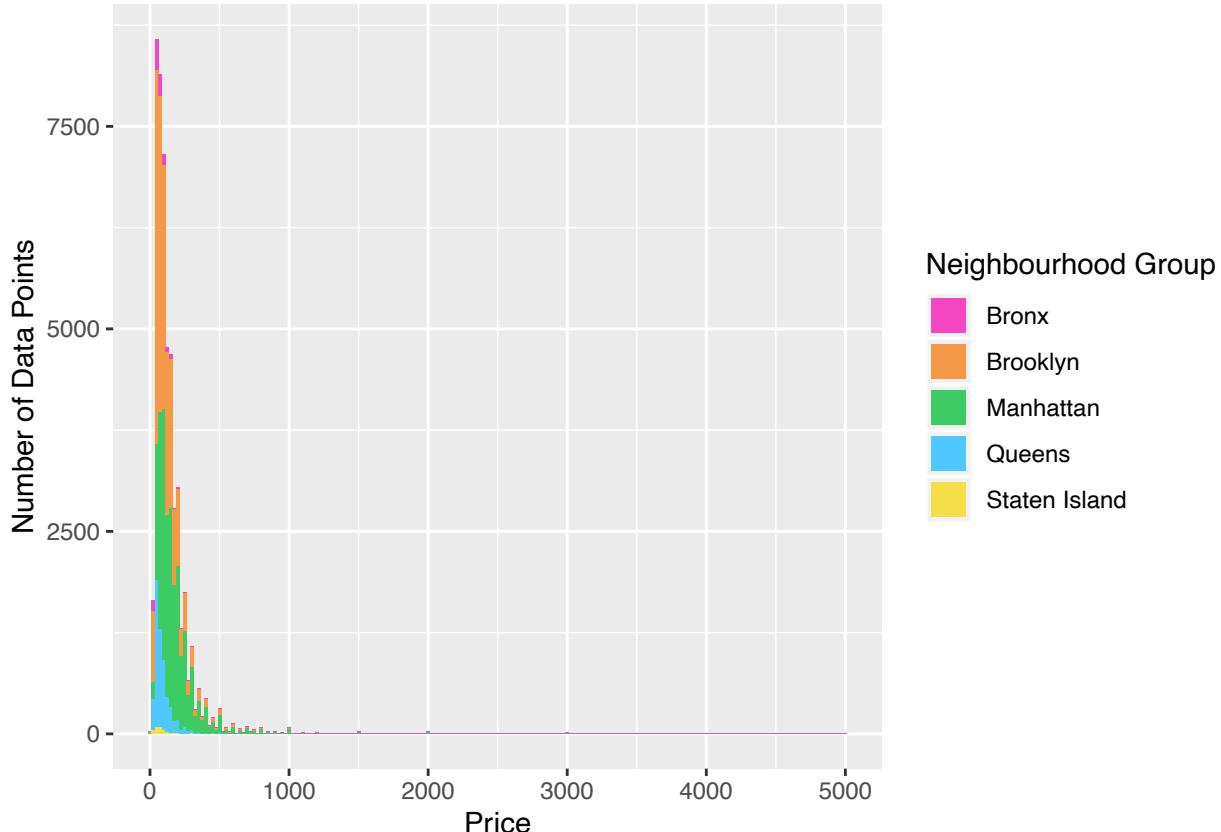
airbnb = read.csv("AB_NYC_2019.csv")
attach(airbnb)

```

```

histdata = airbnb[airbnb$price <= 5000,]
ggplot(histdata, aes(histdata$price, fill = histdata$neighbourhood_group)) +
  geom_histogram(binwidth = 25, size = 2) +
  scale_fill_manual(values=customcolors) +
  labs(x = "Price", y = "Number of Data Points", fill="Neighbourhood Group")

```



```
#ggsave("Hist_Price_NG_Before Random Sampling.png", dpi=1000)
```

Data Sample:

```

data_sample_final = read.csv('sample_data.csv')
data_sample_final = data_sample_final[,c(6:13,15:17)]

```

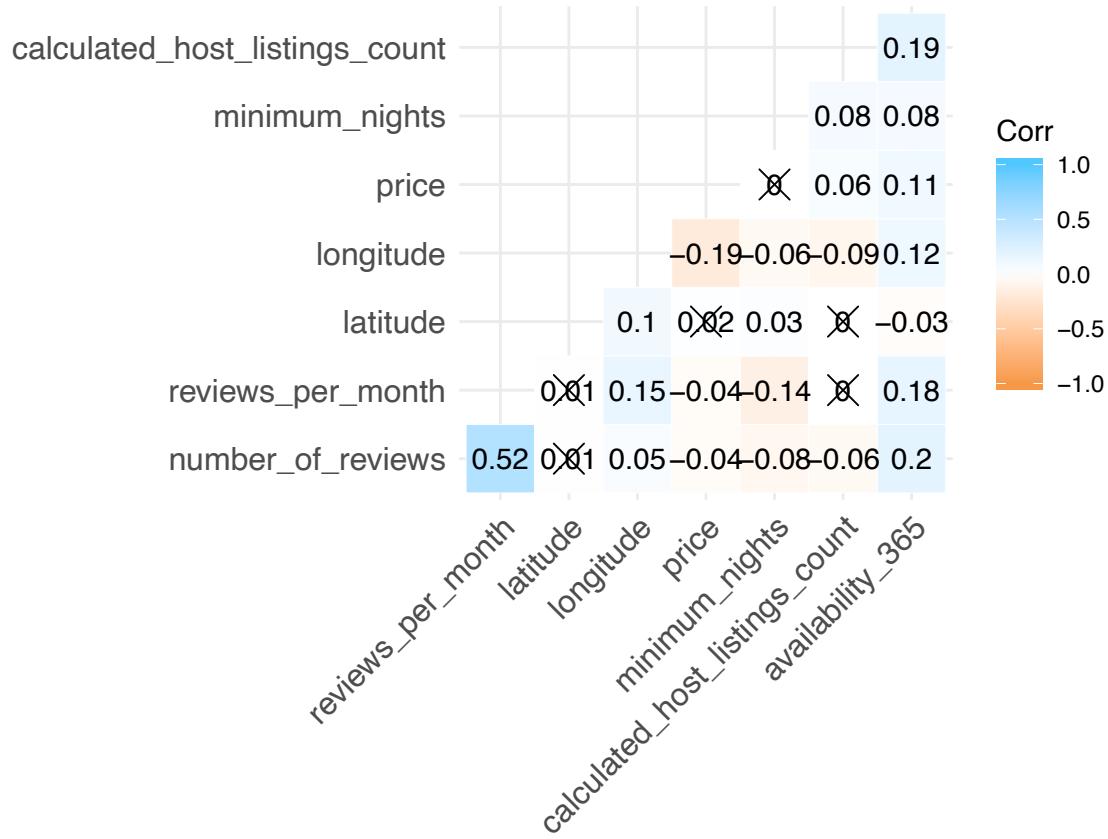
Correlation Matrix:

```

corr_data = data_sample_final[,c(1,3:11)]
corr_data = data.frame(corr_data)

corr <- round(cor(corr_data[,c(2,3,5,6,7,8,9,10)]),2)
ggcorrplot(corr, p.mat = cor_pmat(corr_data[,c(2,3,5,6,7,8,9,10)]),
           hc.order = TRUE, type = "lower",
           color = c(orange, "white", blue),
           outline.col = "white",
           lab = TRUE)

```

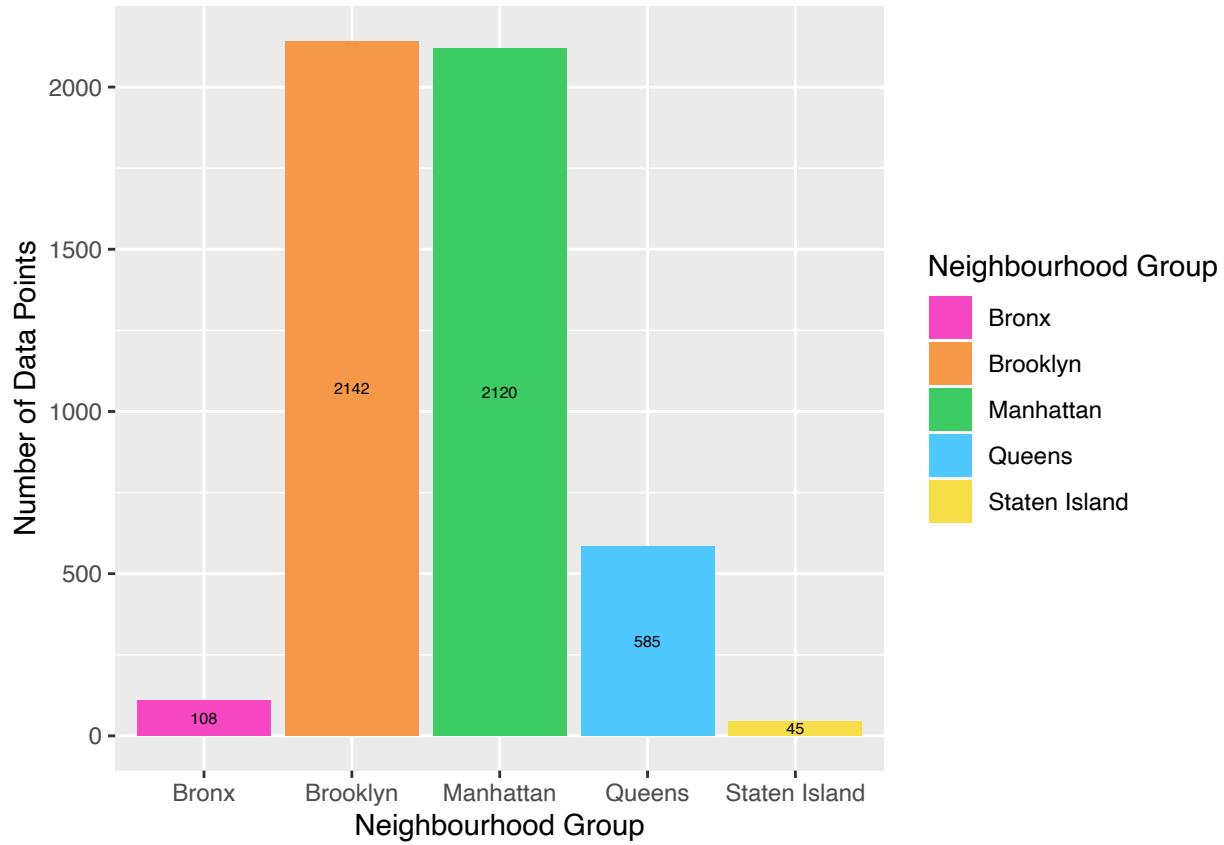


```
#ggsave("corr_matrix.png", dpi=1000)
```

Counts per room type:

```
counts = count(data_sample_final, neighbourhood_group)

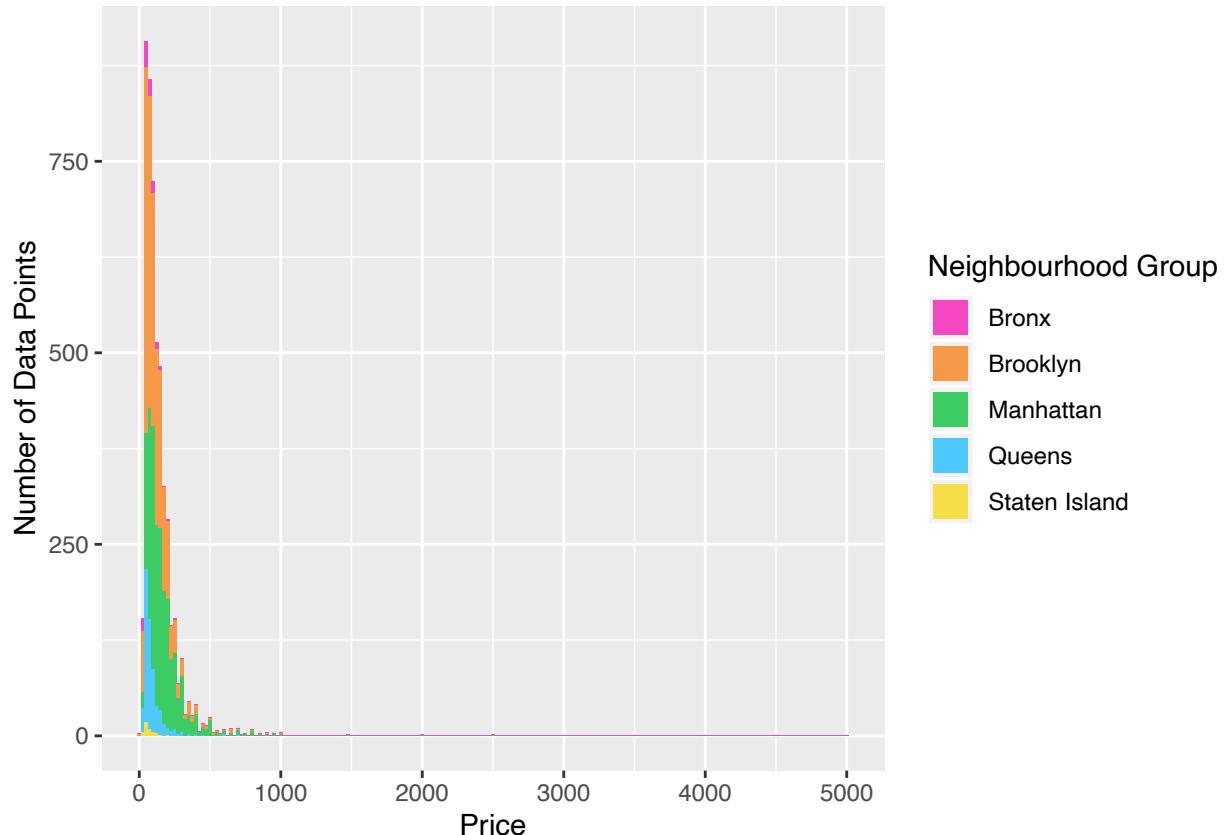
ggplot(counts, aes(x = counts$neighbourhood_group, y = counts$n, fill = counts$neighbourhood_group, lab=
  geom_bar(stat = "identity") +
  geom_text(size = 2, position = position_stack(vjust = 0.5)) +
  scale_fill_manual(values=customcolors) +
  labs(x = "Neighbourhood Group", y = "Number of Data Points", fill="Neighbourhood Group")
```



```
#ggsave("num_of_room_type.png", dpi=1000)
```

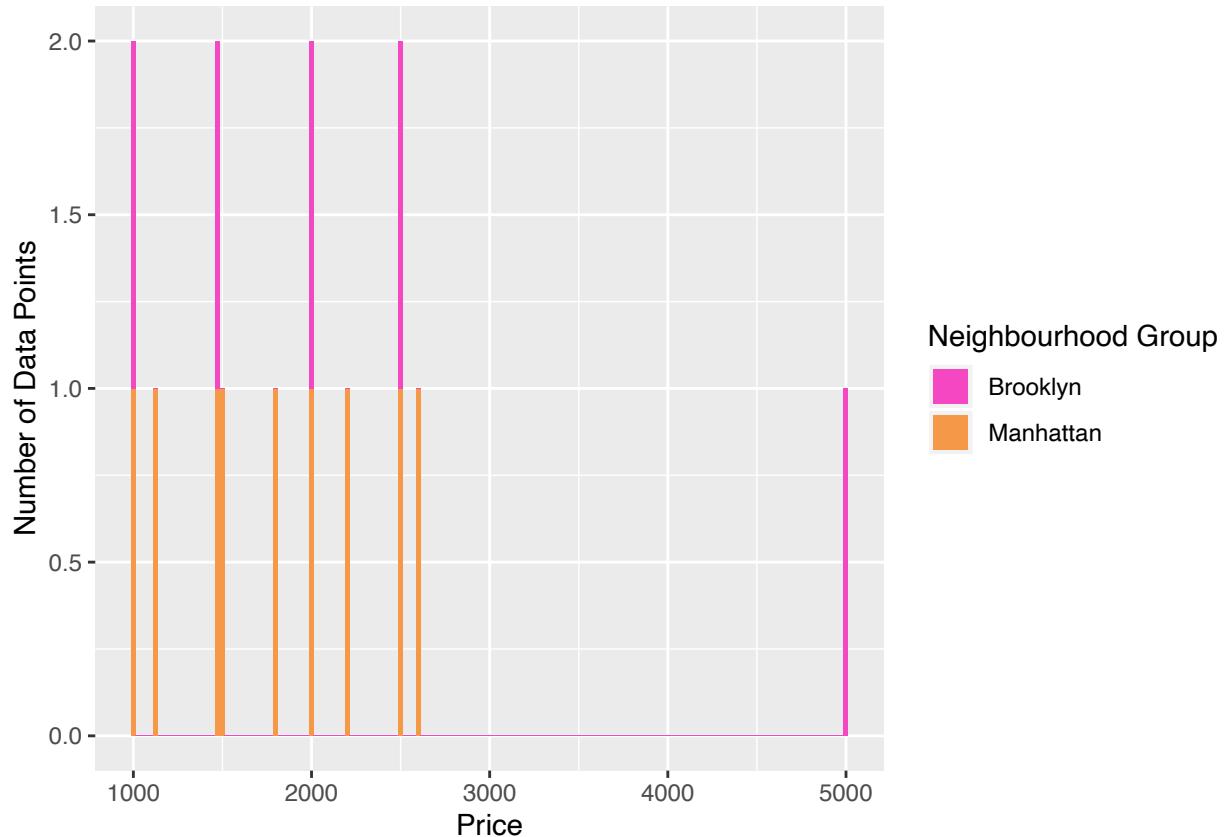
Histogram of all Prices points:

```
histdata = data_sample_final
ggplot(histdata, aes(histdata$price, fill = histdata$neighbourhood_group)) +
  geom_histogram(binwidth = 25, size = 2) +
  scale_fill_manual(values=customcolors) +
  labs(x = "Price", y = "Number of Data Points", fill="Neighbourhood Group")
```



Histogram of the 14 points that are greater than 1000

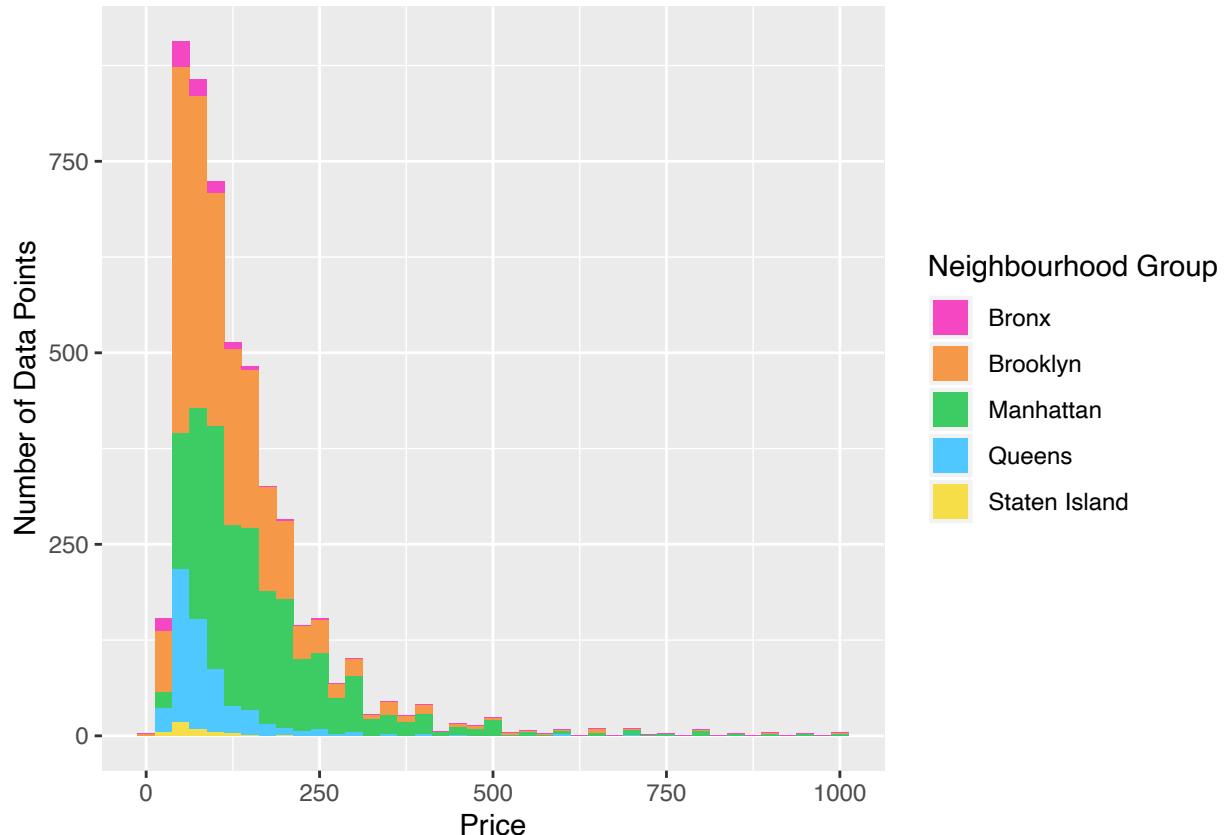
```
histdata = data_sample_final[data_sample_final$price >= 1000,]
ggplot(histdata, aes(histdata$price, fill = histdata$neighbourhood_group)) +
  geom_histogram(binwidth = 25, size = 2) +
  scale_fill_manual(values=customcolors) +
  labs(x = "Price", y = "Number of Data Points", fill="Neighbourhood Group")
```



```
#ggsave("Hist_Price_gt_1000_NG.png", dpi=1000)
```

Histogram excluding the 14 points that are greater than 1000

```
histdata = data_sample_final[data_sample_final$price <= 1000,]
ggplot(histdata, aes(histdata$price, fill = histdata$neighbourhood_group)) +
  geom_histogram(binwidth = 25, size = 2) +
  scale_fill_manual(values=customcolors) +
  labs(x = "Price", y = "Number of Data Points", fill="Neighbourhood Group")
```



```
#ggsave("Hist_Price_lt_1000_NG.png", dpi=1000)
```

Some variables for maps:

```
# lat/long centers
ALL_centerlong = (min(data_sample_final$longitude) + max(data_sample_final$longitude))/2
ALL_centerlat = (min(data_sample_final$latitude) + max(data_sample_final$latitude))/2

# neighbourhood group centers
NG_centerlong = c()
NG_centerlat = c()

neighbourhood_groups = as.character(unique(data_sample_final$neighbourhood_group))

for (i in 1:length(neighbourhood_groups)) {
  lat = data_sample_final$latitude[data_sample_final$neighbourhood_group == neighbourhood_groups[i]]
  long = data_sample_final$longitude[data_sample_final$neighbourhood_group == neighbourhood_groups[i]]
  NG_centerlong[i] = (min(long) + max(long))/2
  NG_centerlat[i] = (min(lat) + max(lat))/2
}

NG_centers = data.frame(neighbourhood_groups, NG_centerlong, NG_centerlat)

#neighbourhood centers
N_centerlong = c()
N_centerlat = c()
```

```

neighbourhoods = as.character(unique(data_sample_final$neighbourhood))

for (i in 1:length(neighbourhoods)) {
  lat = data_sample_final$latitude[data_sample_final$neighbourhood == neighbourhoods[i]]
  long = data_sample_final$longitude[data_sample_final$neighbourhood == neighbourhoods[i]]
  N_centerlong[i] = (min(long) + max(long))/2
  N_centerlat[i] = (min(lat) + max(lat))/2
}

N_centers = data.frame(neighbourhoods, N_centerlong, N_centerlat)

```

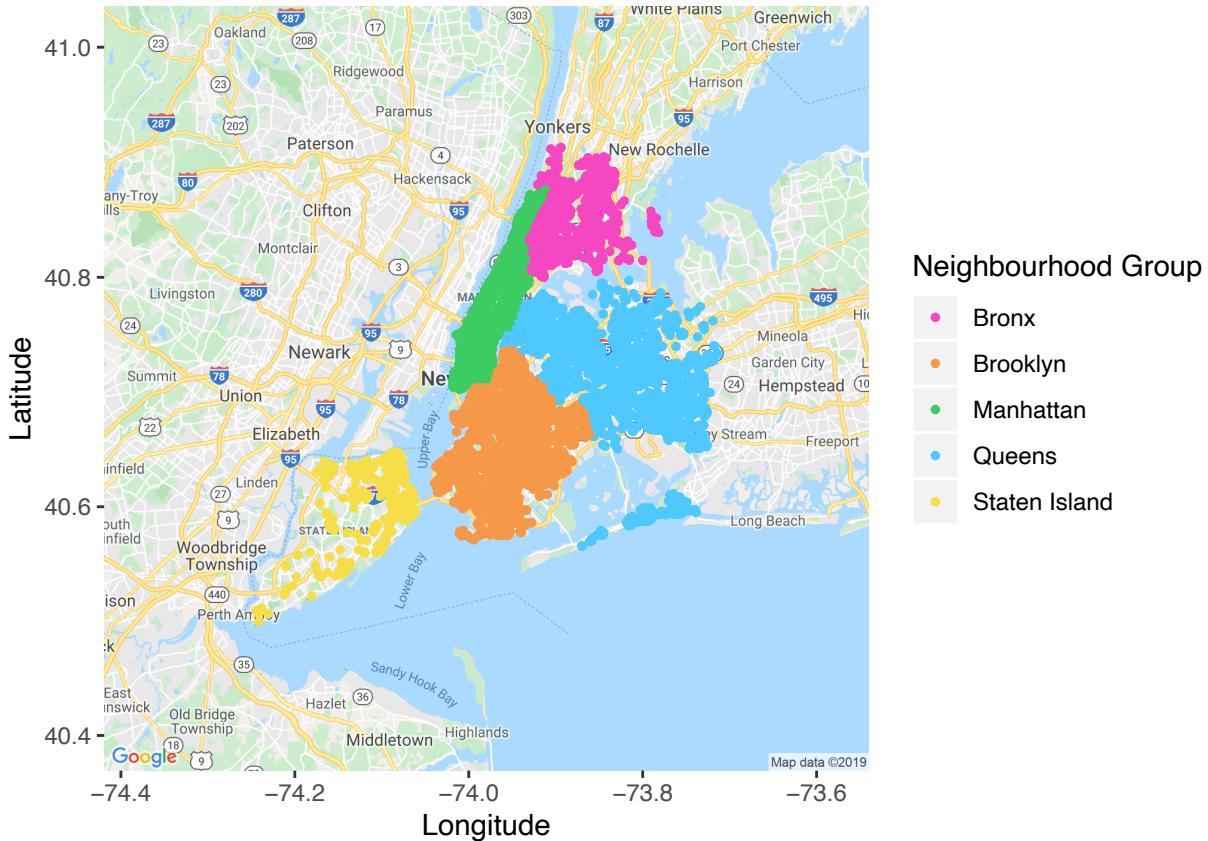
Map of Neighbourhoods FULL DATA:

```

map = get_map(location = c(ALL_centerlong, ALL_centerlat),
              source = "google",
              zoom = 10,
              maptype = "roadmap",
              size = c(640, 640),
              scale = 2)

print(ggmap(map)
  + geom_point(data = airbnb,
                aes(x = airbnb$longitude,
                    y = airbnb$latitude,
                    colour = factor(airbnb$neighbourhood_group)
                ),
                size = 1)
  + labs(x = "Longitude",
         y = "Latitude",
         colour="Neighbourhood Group")
  + scale_color_manual(values=customcolors)
)

```

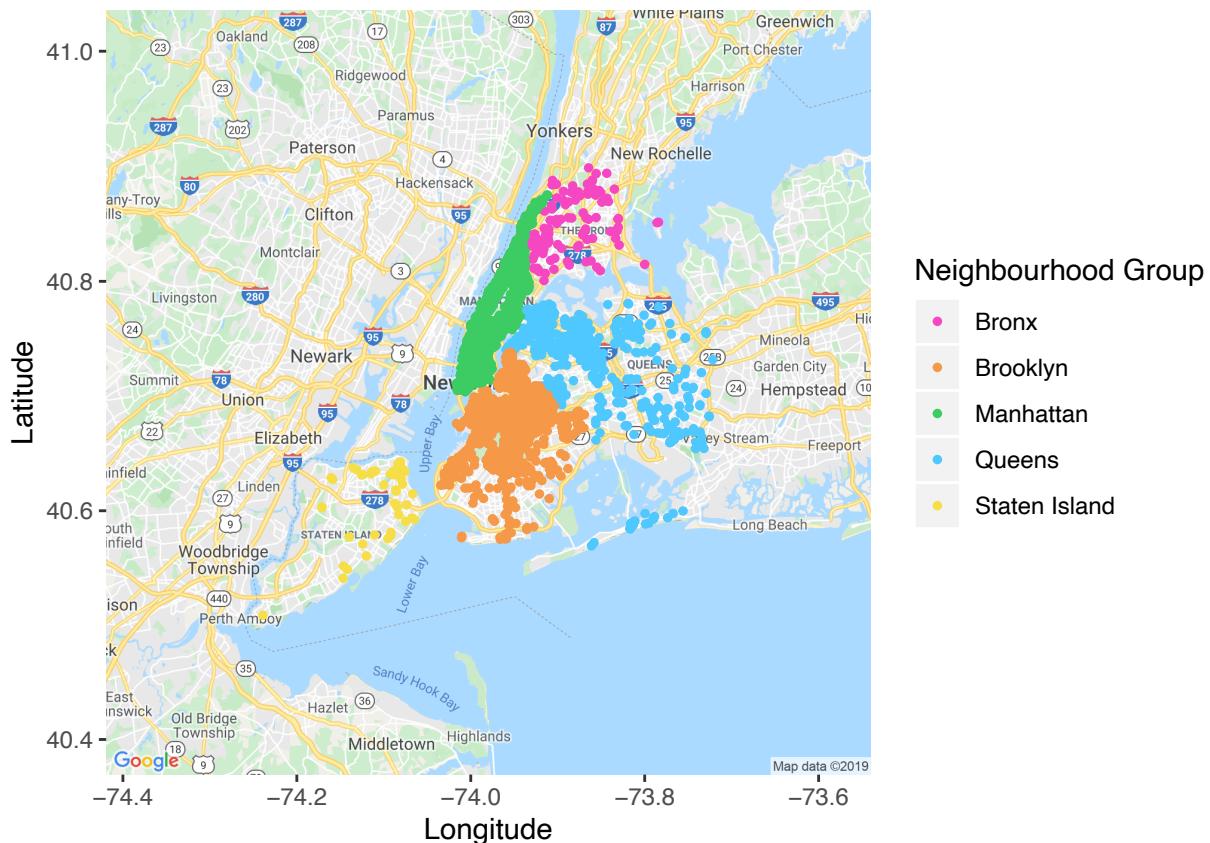


```
#ggsave("map_of_neighbourhoods_FULLDATA.png", dpi=1000)
```

Map of Neighbourhoods SAMPLE:

```
map = get_map(location = c(ALL_centerlong, ALL_centerlat),
              source = "google",
              zoom = 10,
              maptype = "roadmap",
              size = c(640, 640),
              scale = 2)

print(ggmap(map)
      + geom_point(data = data_sample_final,
                    aes(x = data_sample_final$longitude,
                        y = data_sample_final$latitude,
                        colour = factor(data_sample_final$neighbourhood_group)
                    ),
                    size = 1)
      + labs(x = "Longitude",
             y = "Latitude",
             colour="Neighbourhood Group")
      + scale_color_manual(values=customcolors)
)
```

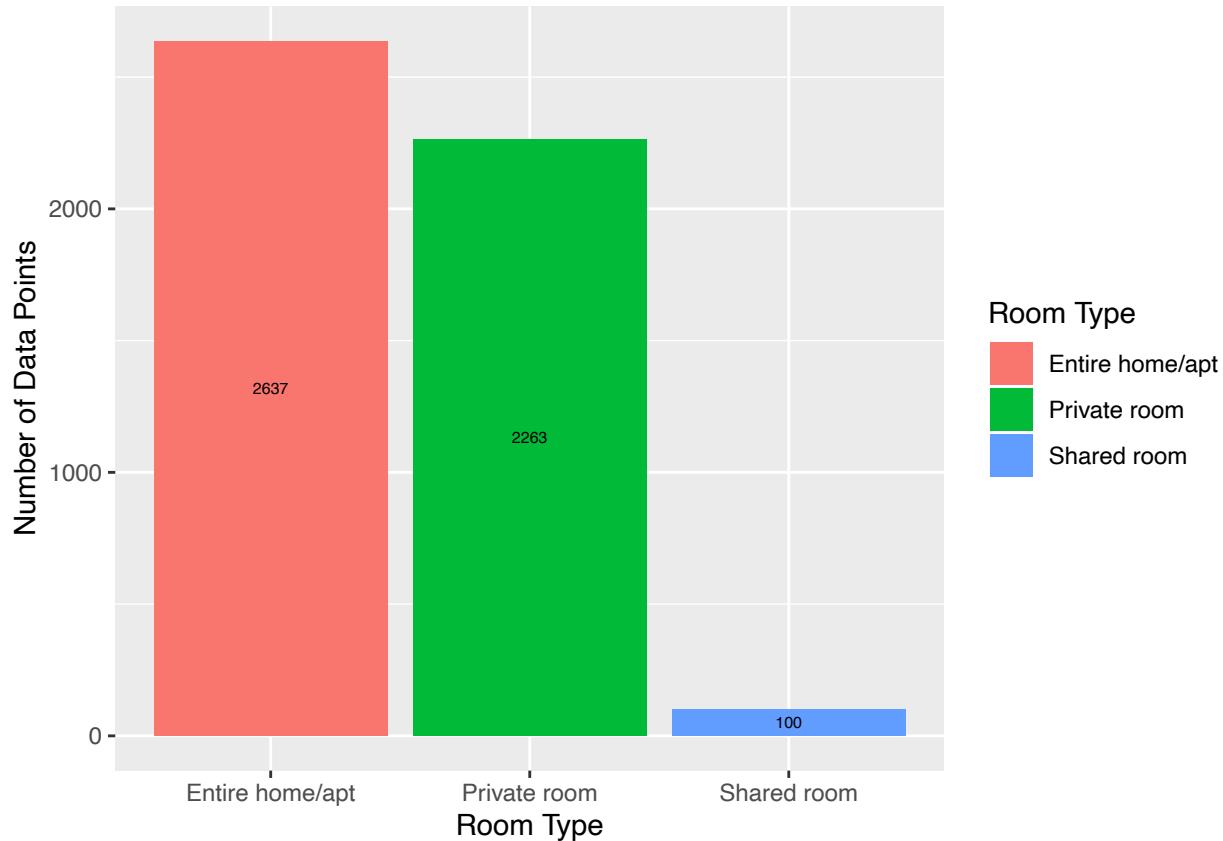


```
#ggsave("map_of_neighbourhoods_SAMPLE.png", dpi=1000)
```

Number of Room types:

```
counts = count(data_sample_final, room_type)

ggplot(counts, aes(x = counts$room_type, y = counts$n, fill = counts$room_type, label = counts$n)) +
  geom_bar(stat = "identity") +
  geom_text(size = 2, position = position_stack(vjust = 0.5)) +
  labs(x = "Room Type", y = "Number of Data Points", fill="Room Type")
```



```
#ggsave("num_of_Room_type.png", dpi=1000)
```

Getting Top 10 neighbourhoods:

```
counts = count(data_sample_final, neighbourhood_group, neighbourhood)
counts[order(counts$neighbourhood_group, counts$neighbourhood, -counts$n),]
```

```
## # A tibble: 178 x 3
##   neighbourhood_group neighbourhood      n
##   <fct>              <fct>          <int>
## 1 Bronx               Allerton        3
## 2 Bronx               Baychester     2
## 3 Bronx               Belmont        2
## 4 Bronx               Bronxdale      2
## 5 Bronx               City Island    2
## 6 Bronx               Claremont Village 5
## 7 Bronx               Clason Point   4
## 8 Bronx               Concourse      6
## 9 Bronx               Concourse Village 4
## 10 Bronx              East Morrisania 1
## # ... with 168 more rows
brooklyn = counts[counts$neighbourhood_group == 'Brooklyn',]
manhattan = counts[counts$neighbourhood_group == 'Manhattan',]
queens = counts[counts$neighbourhood_group == 'Queens',]
statenIsland = counts[counts$neighbourhood_group == 'Staten Island',]
bronx = counts[counts$neighbourhood_group == 'Bronx',]
```

```

nsi = statenIsland[order(statenIsland$neighbourhood_group, -statenIsland$n),][1:10,]
nq = queens[order(queens$neighbourhood_group, -queens$n),][1:10,]
nm = manhattan[order(manhattan$neighbourhood_group, -manhattan$n),][1:10,]
nbn = brooklyn[order(brooklyn$neighbourhood_group, -brooklyn$n),][1:10,]
nbx = bronx[order(bronx$neighbourhood_group, -bronx$n),][1:10,]

top10 = rbind(nsi,nq,nm,nbn,nbx)

```

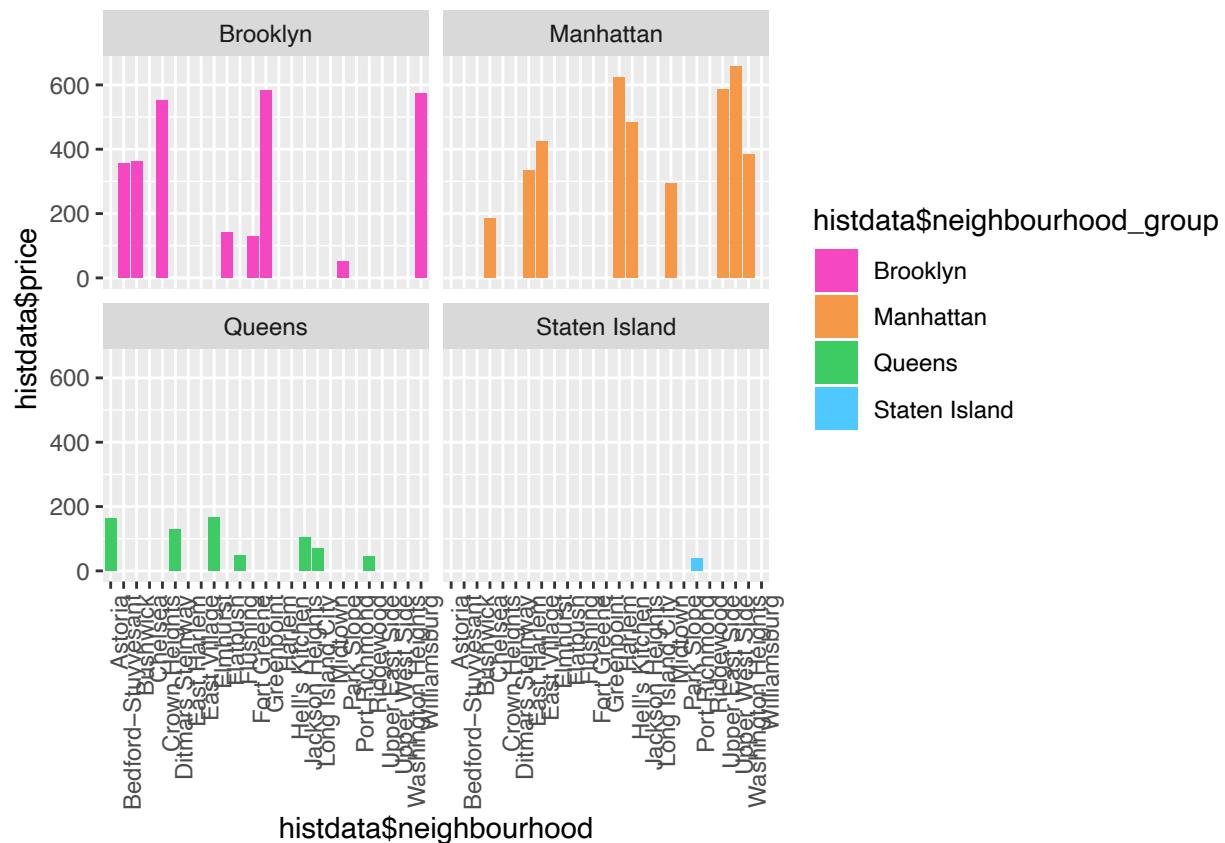
Price distribution for top 10 neighbourhoods:

```

histdata = data_sample_final[data_sample_final$neighbourhood == top10$neighbourhood,]
histdata = histdata[histdata$price <= 1000,]

ggplot(histdata, aes(x=histdata$neighbourhood, y=histdata$price, fill = histdata$neighbourhood_group)) +
  geom_bar(stat = "identity") +
  facet_wrap(~histdata$neighbourhood_group) +
  scale_fill_manual(values=customcolors) +
  theme(axis.text.x = element_text(angle = 90, hjust = 1))

```



Prices on map:

```

data = data_sample_final[data_sample_final$price<=1000,]

map = get_map(location = c(ALL_centerlong, ALL_centerlat),
              source = "google",

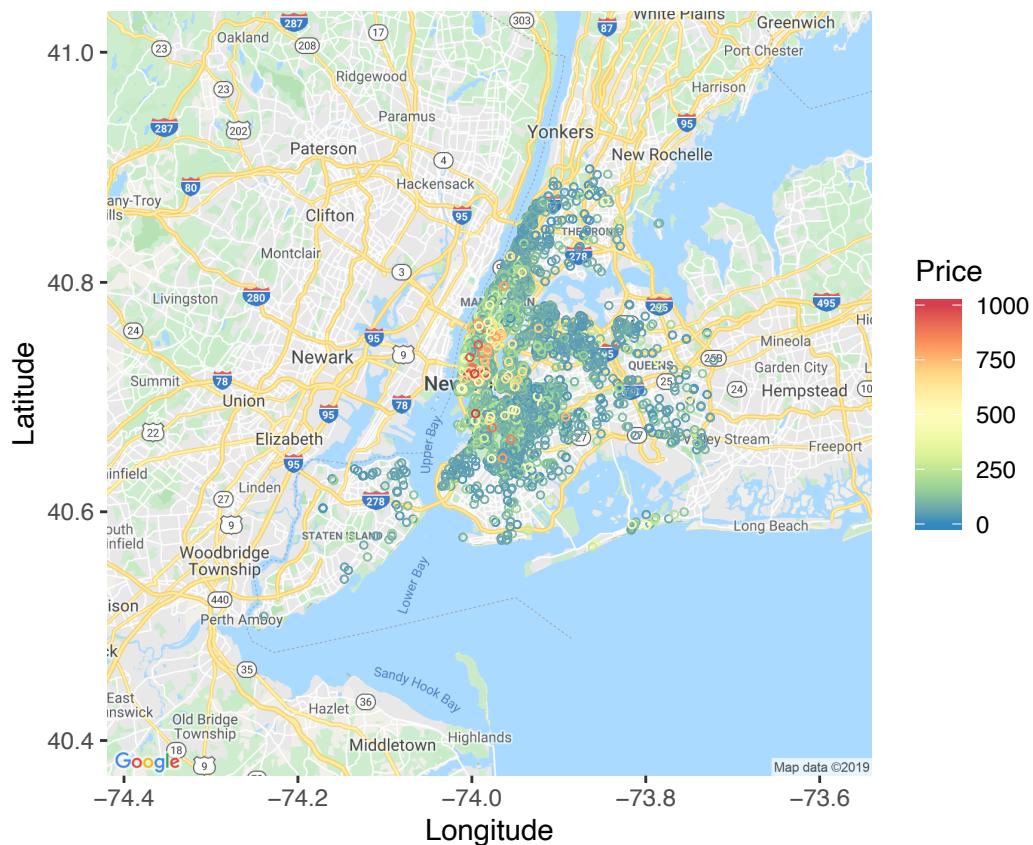
```

```

    zoom = 10,
    maptype = "roadmap",
    size = c(640, 640),
    scale = 2)

## Source : https://maps.googleapis.com/maps/api/staticmap?center=40.70373,-73.980465&zoom=10&size=640x640
print(ggmap(map)
+ geom_point(data = data[data$price<=500,],
             aes(x = data$longitude[data$price<=500],
                 y = data$latitude[data$price<=500],
                 colour = data$price[data$price<=500]#,
                 #size = data_sample_final$price
             ),
             shape = 1, size = 1)
+ geom_point(data = data[data$price>500,],
             aes(x = data$longitude[data$price>500],
                 y = data$latitude[data$price>500],
                 colour = data$price[data$price>500]#,
                 #size = data_sample_final$price
             ),
             shape = 1, size = 1)
+ scale_colour_distiller(palette = "Spectral")
+ labs(x = "Longitude",
       y = "Latitude",
       colour="Price")
)

```

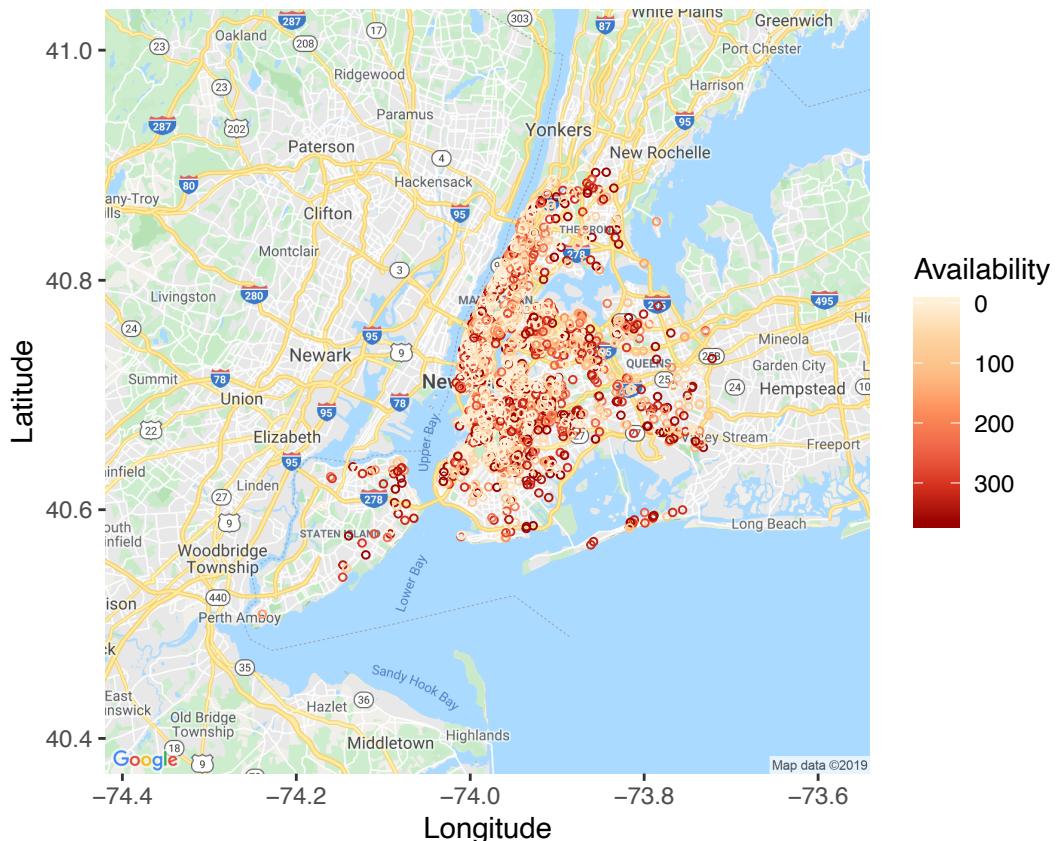


```
#ggsave("map_of_prices.png", dpi=1000)
```

Availability on map:

```
data = data_sample_final
map = get_map(location = c(ALL_centerlong, ALL_centerlat),
              source = "google",
              zoom = 10,
              maptype = "roadmap",
              size = c(640, 640),
              scale = 2)

## Source : https://maps.googleapis.com/maps/api/staticmap?center=40.70373,-73.980465&zoom=10&size=640x640
print(ggmap(map)
+ geom_point(data = data,
             aes(x = data$longitude,
                 y = data$latitude,
                 colour = data$availability_365
             ),
             shape = 1, size = 1)
+ scale_colour_distiller(palette = "OrRd", trans = 'reverse')
+ labs(x = "Longitude",
       y = "Latitude",
       colour="Availability"
     )
)
```

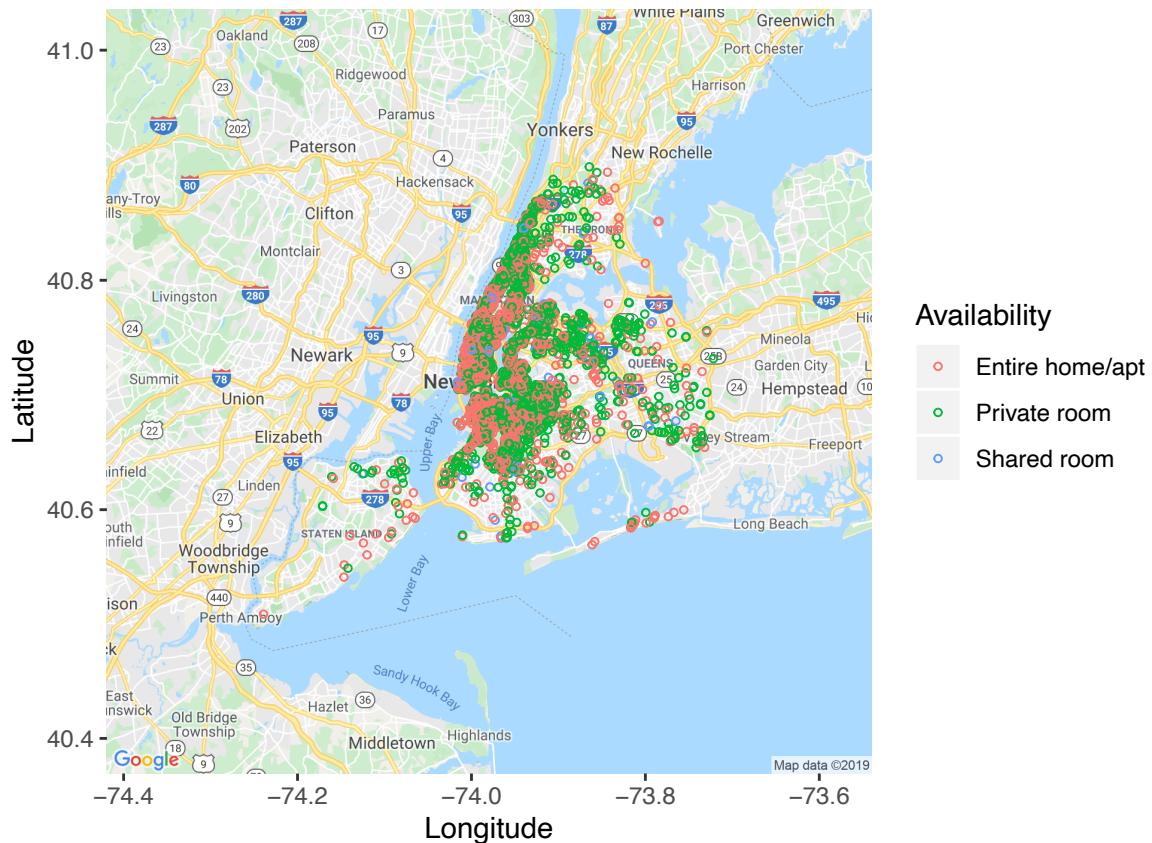


```
#ggsave("map_of_availability.png", dpi=1000)
```

Room Type on Map:

```
data = data_sample_final
map = get_map(location = c(ALL_centerlong, ALL_centerlat),
              source = "google",
              zoom = 10,
              maptype = "roadmap",
              size = c(640, 640),
              scale = 2)

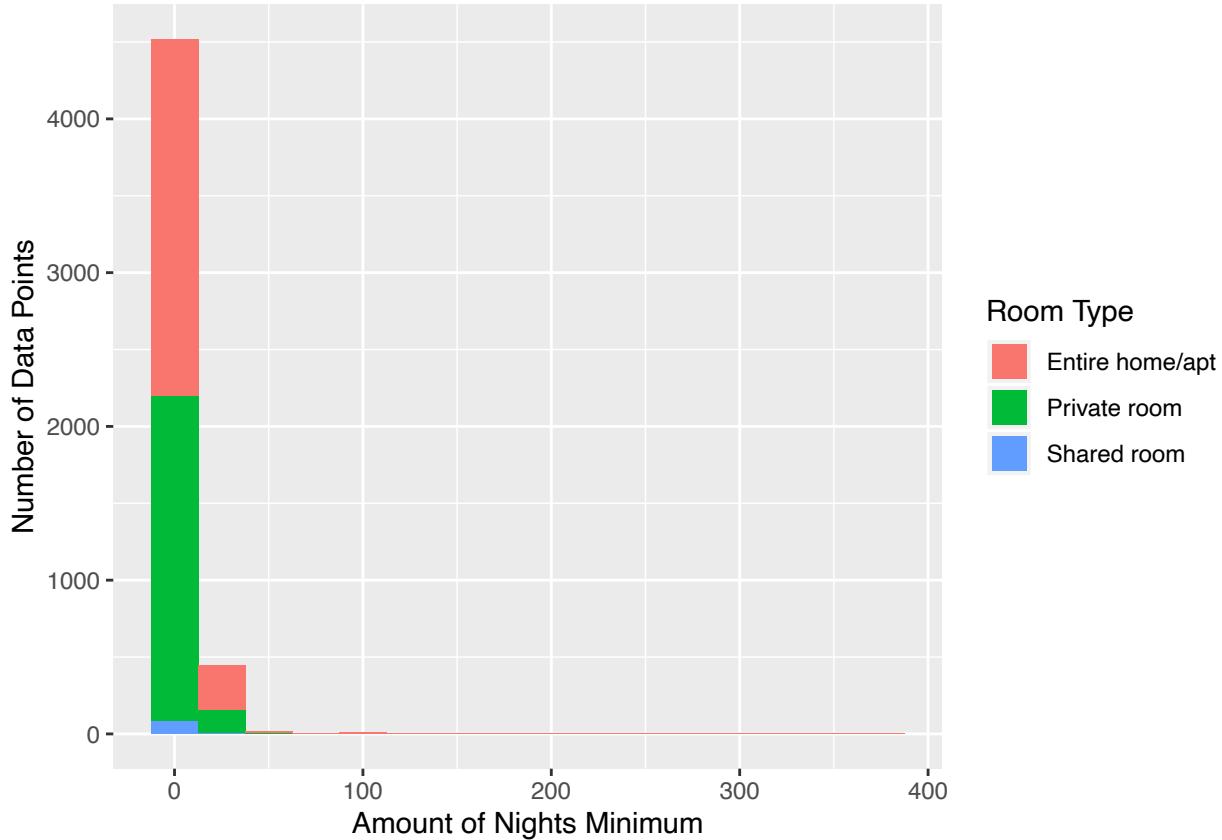
## Source : https://maps.googleapis.com/maps/api/staticmap?center=40.70373,-73.980465&zoom=10&size=640x640
print(ggmap(map)
+ geom_point(data = data,
             aes(x = data$longitude,
                 y = data$latitude,
                 colour = data$room_type
             ),
             shape = 1, size = 1)
#+ scale_colour_distiller(palette = "OrRd", trans = 'reverse')
+ labs(x = "Longitude",
       y = "Latitude",
       colour="Availability"
     )
)
```



```
#ggsave("map_of_roomtype.png", dpi=1000)
```

Histogram of Minimum nights:

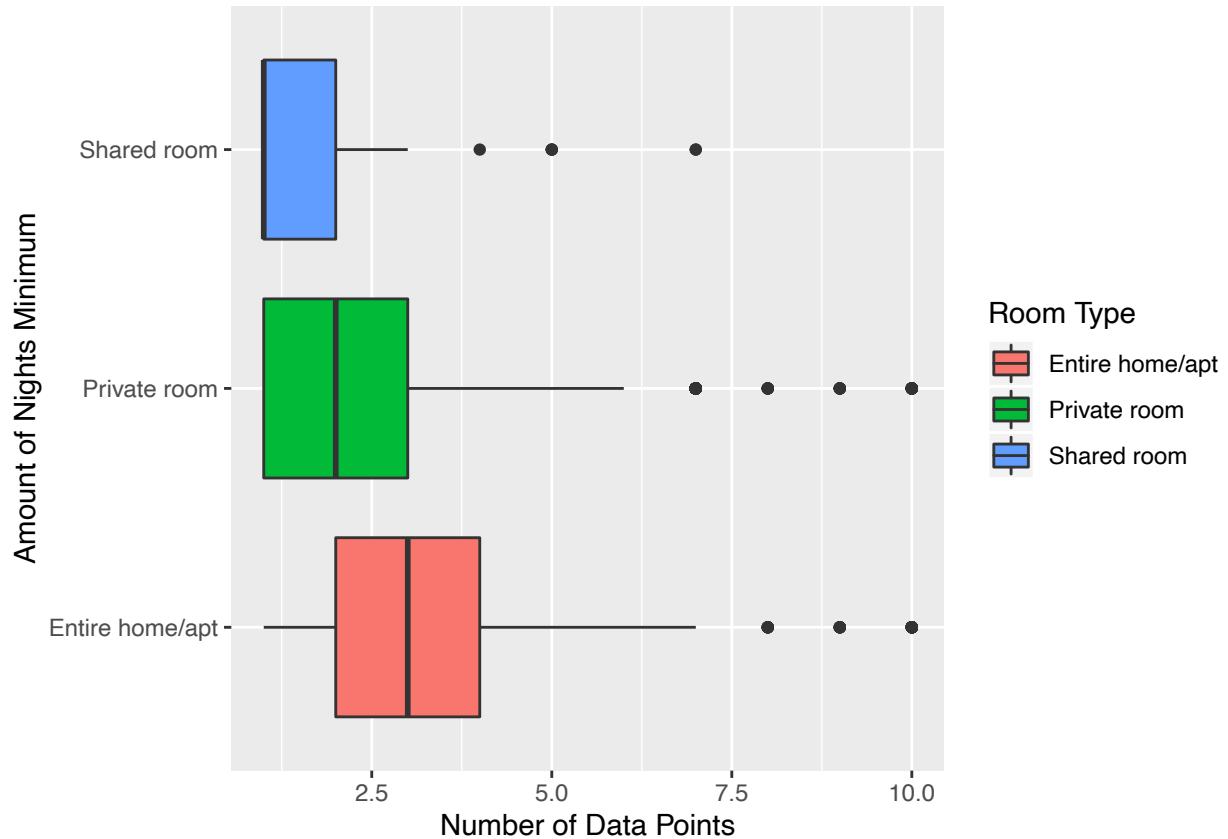
```
histdata = data_sample_final  
ggplot(histdata, aes(histdata$minimum_nights, fill = histdata$room_type)) +  
  geom_histogram(binwidth = 25, size = 2) +  
  labs(x = "Amount of Nights Minimum", y = "Number of Data Points", fill="Room Type")
```



```
#ggsave("Hist_MinNights_RoomType.png", dpi=1000)
```

Box Plot of Minimum Nights:

```
data = data_sample_final[data_sample_final$minimum_nights <= 10,]  
ggplot(data, aes(x=data$room_type, y=data$minimum_nights, fill=data$room_type)) +  
  geom_boxplot() +  
  coord_flip() +  
  labs(x = "Amount of Nights Minimum", y = "Number of Data Points", fill="Room Type")
```



```
#ggsave("Box_MinNights_RoomType.png", dpi=1000)
```

Price by location broken out by neighbourhood_group:

```
data_cut = data_sample_final[data_sample_final$price<=1000,]

for (i in 1:length(neighbourhood_groups)) {
  NG = as.character(NG_centers[i,1])
  data = data_cut[data_cut$neighbourhood_group == NG,]
  centerlong = (min(data$longitude) + max(data$longitude))/2
  centerlat = (min(data$latitude) + max(data$latitude))/2

  map = get_map(location = c(centerlong, centerlat),
                source = "google",
                zoom = 11,
                maptype = "roadmap",
                size = c(640, 640),
                scale = 2)

  print(ggmap(map)
    + geom_point(data = data[data$price<=500,],
                  aes(x = data$longitude[data$price<=500],
                      y = data$latitude[data$price<=500],
                      colour = data$price[data$price<=500]
                      ),
                  shape = 1, size = 1)
```

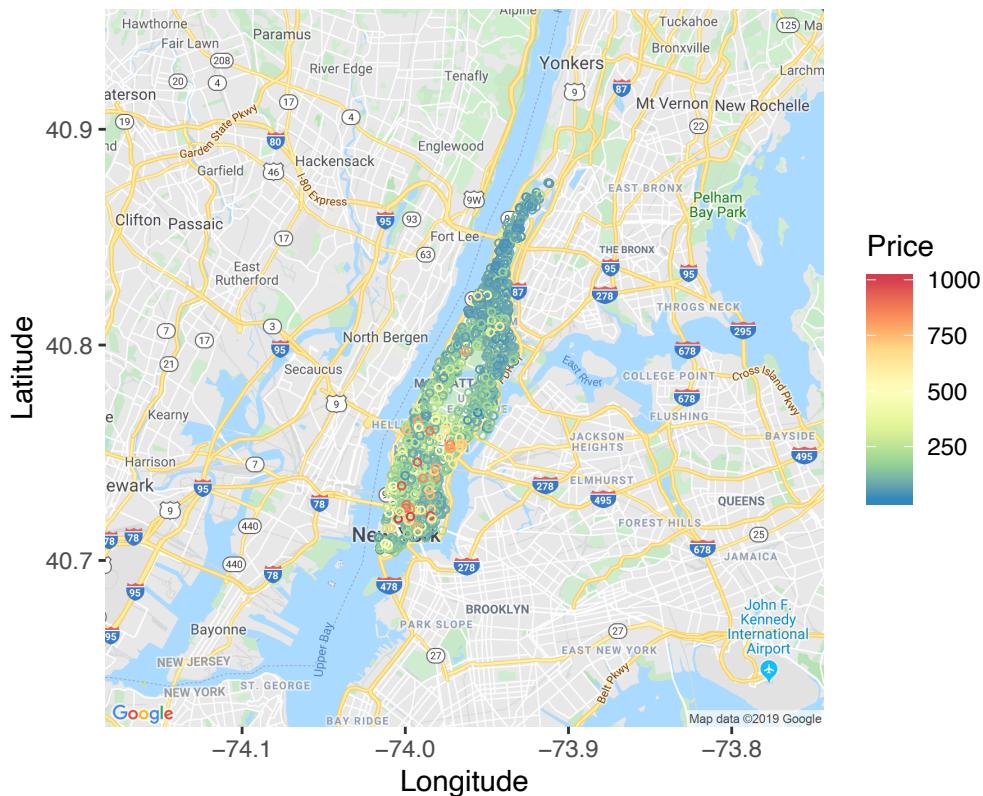
```

+ geom_point(data = data[data$price>500,],
             aes(x = data$longitude[data$price>500],
                 y = data$latitude[data$price>500],
                 colour = data$price[data$price>500]
                 ),
             shape = 1, size = 1)
+ scale_colour_distiller(palette = "Spectral")
+ labs(x = "Longitude",
       y = "Latitude",
       colour="Price",
       title = NG)
)

print(cat('\n'))
}

```

Manhattan

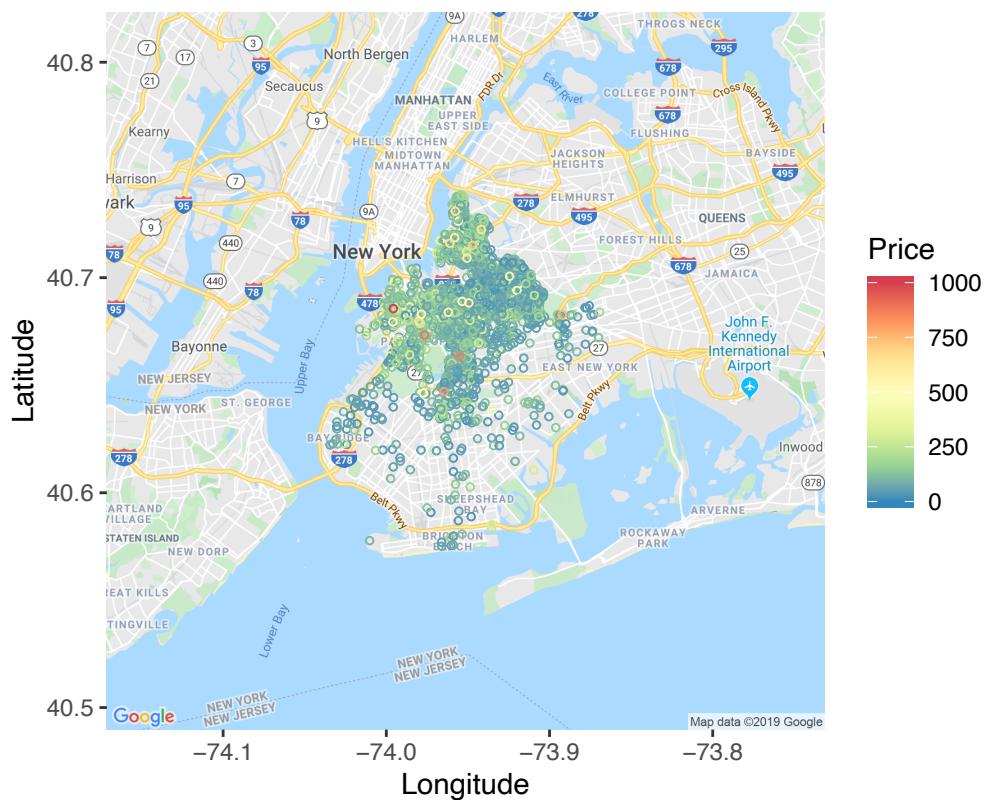


```

##  
## NULL

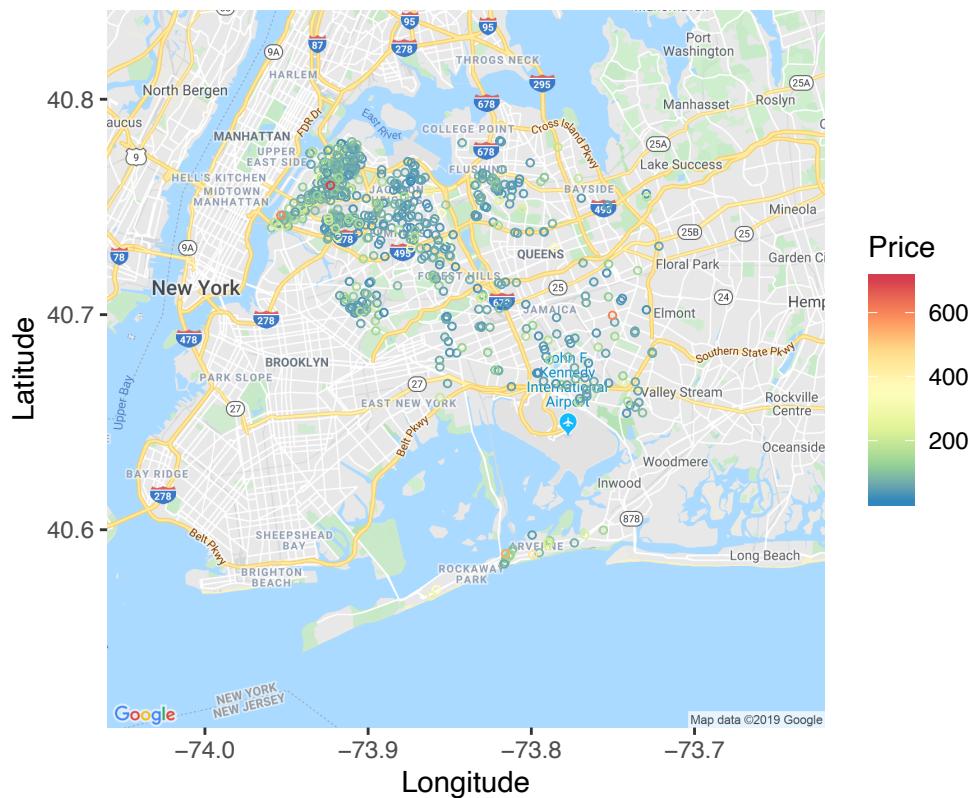
```

Brooklyn



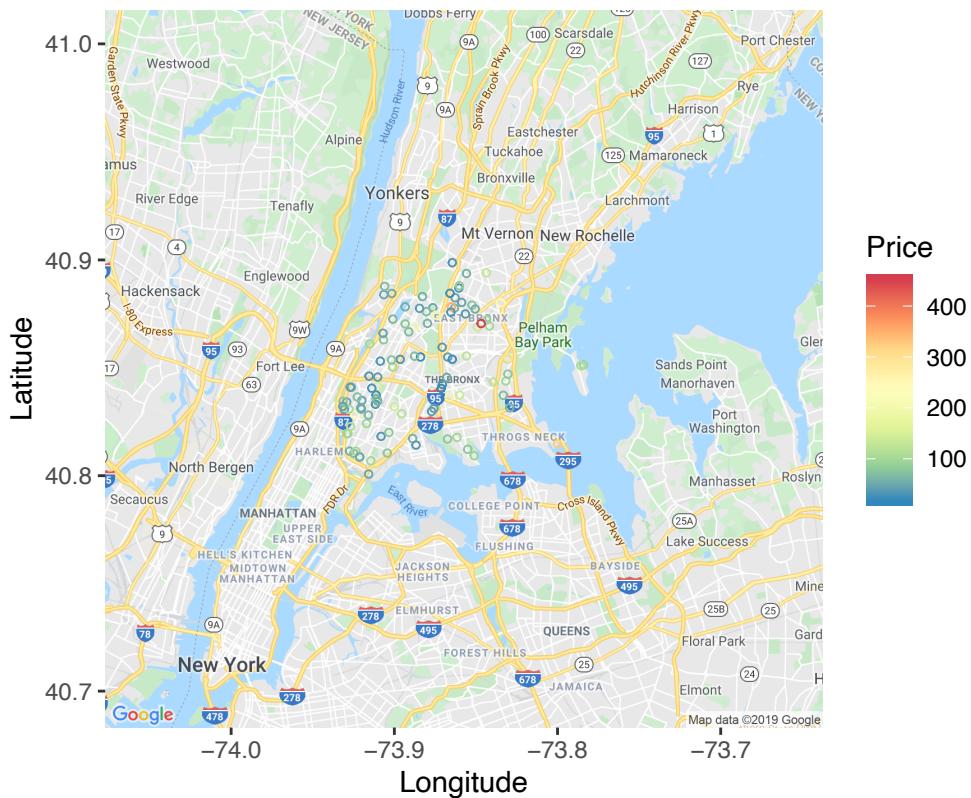
```
##  
## NULL
```

Queens



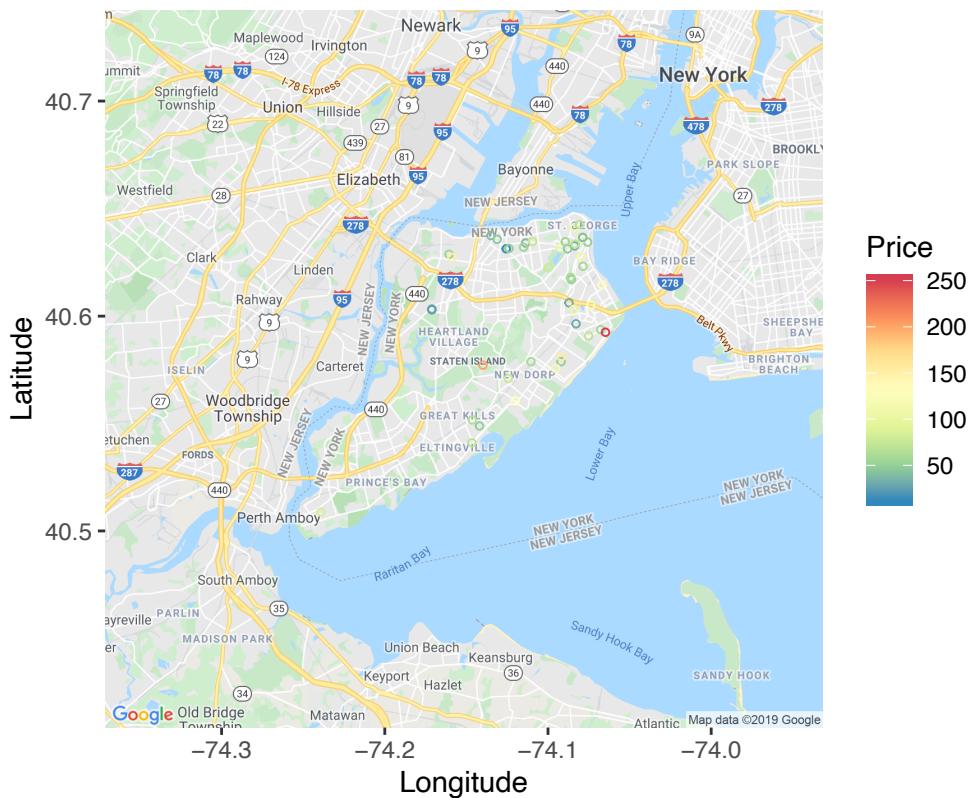
```
##  
## NULL
```

Bronx



```
##  
## NULL
```

Staten Island



```
##  
## NULL
```