Determining the Price of San Francisco AirBnB Listings

The Outliers: Hailey han, Joshua Minwoo Kang, Jihong Lee

```
#### Initial Setup

# read in dataset
setwd(".")
listings_raw <- read.csv("listings.csv")
dim(listings_raw)
```

```
## [1] 8111  106
```

```
# choose subset with variables of interest
listings_raw <- subset(listings_raw, select=c("price", "host_response_rate", "host_is_superhost", "neigh
                                      "latitude", "longitude", "property_type", "room_type", "accommoda
                                      "bathrooms", "bedrooms", "number_of_reviews", "review_scores_ratin

# organize into variables for later ease
listings_raw$price <- as.numeric(gsub("[$]", "", listings_raw$price))
listings_raw$host_response_rate <- as.numeric(gsub("[%]", "", listings_raw$host_response_rate))

# clean of invalid observations
listings_raw <- listings_raw[complete.cases(listings_raw), ]

# fix 0 price entry for log(price) transformation ( get rid of OUTLIER)
listings = listings_raw[-which(listings_raw$price == 0),]
dim(listings)
```

```
## [1] 5691   13
```

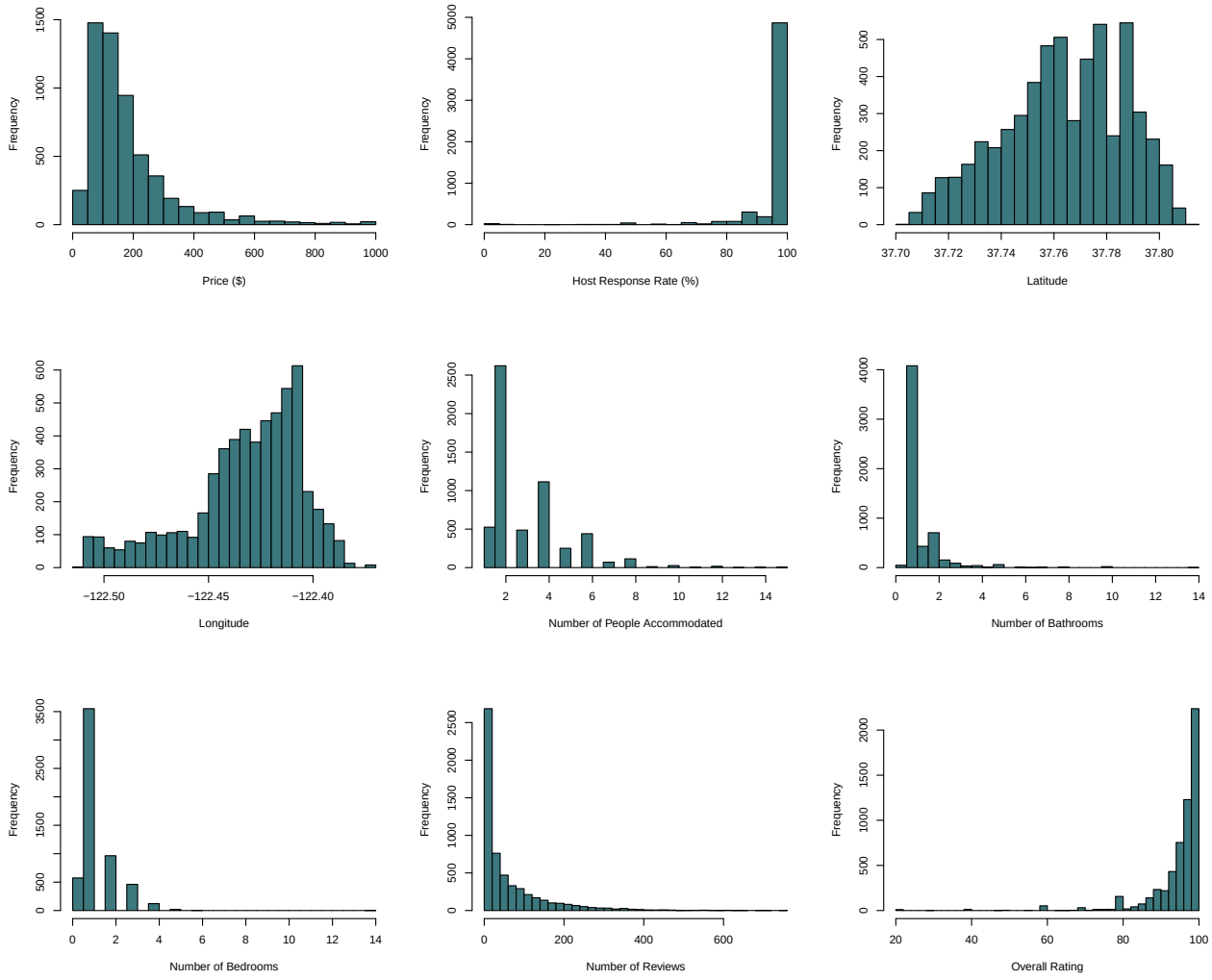
```
#### Exploratory Data Analysis

# quantitative variables in dataset: price, host response rate, latitude, longitude, accommodates, bath
listings_quant <- subset(listings, select=c("price", "host_response_rate", "latitude", "longitude", "ac
                                            "bathrooms", "bedrooms", "number_of_reviews", "review_scores
# categorical variables in dataset: host is superhost, neighborhood, property type, room type
listings_cat <- subset(listings, select=c("host_is_superhost", "neighbourhood", "property_type", "room_t

## Univariate EDA

# statistical display of quantitative variables
l_mean <- apply(listings_quant, 2, mean)
l_sd <- apply(listings_quant, 2, sd)
l_med <- apply(listings_quant, 2, median)
l_iqr <- apply(listings_quant, 2, IQR, na.rm=T)
l_min <- apply(listings_quant, 2, min)
l_max <- apply(listings_quant, 2, max)

statsum <- data.frame(l_mean, l_sd, l_med, l_iqr, l_min, l_max)
rownames(statsum) <- c('Price ($)', 'Host Response Rate (%)', 'Latitude', 'Longitude', 'Number of People
colnames(statsum) <- c("Mean", "Standard Deviation", "Median", "IQR", "Minimum", "Maximum")
```

```
# Print summary
library(xtable)
print(xtable(statsum, caption = "Summary of continuous variables"), comment=FALSE, caption.placement =
```

Table 1: Summary of continuous variables

	Mean	Standard Deviation	Median	IQR	Minimum	Maximum
Price ($)	188.39	147.67	145.00	126.00	10.00	999.00
Host Response Rate (%)	97.09	10.28	100.00	0.00	0.00	100.00
Latitude	37.76	0.02	37.76	0.03	37.70	37.81
Longitude	-122.43	0.03	-122.43	0.03	-122.51	-122.37
Number of People Accommodated	3.12	1.84	2.00	2.00	1.00	15.00
Number of Bathrooms	1.37	0.95	1.00	0.50	0.00	14.00
Number of Bedrooms	1.31	0.88	1.00	1.00	0.00	14.00
Number of Reviews	60.46	86.80	24.00	77.00	1.00	757.00
Overall Rating	95.24	7.75	98.00	6.00	20.00	100.00

```
# graphical display of quantitative variables
par(mfrow=c(3,3))
quantvars <- with(listings, cbind(price, host_response_rate, latitude, longitude, accommodates, bathroo
quantnames <- c('Price ($)', 'Host Response Rate (%)', 'Latitude', 'Longitude', 'Number of People Accom
for(i in 1:quantnum) {
  hist(quantvars[, i], main="",
       xlab=quantnames[i], ylab="Frequency",
       breaks=30, col="#3C787E")
}
```




```
##
## Attaching package: 'dplyr'
## The following objects are masked from 'package:stats':
##
##     filter, lag
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```r
# Group property types

# Based on EDA barplot of log(price) ~ property_type
# (1) properties with higher prices: Villa, Resort, Hotel, and Boutique Hotel
expensive_property = c("Villa", "Resort", "Hotel", "Loftl")
listings$property_high =  1*(listings$property_type %in% expensive_property)

# (2) Properties with lower prices: Hostel and Bed and breakfast
cheaper_property = c("Hostel", "Bed and breakfast")
listings$property_low =  1*(listings$property_type %in% cheaper_property)
```

```r
# CATEGORICAL VARIABLES:

# statistical display of categorical variables
superhost_tab <- table(listings$host_is_superhost)
property_tab <- table(listings$property_type)
room_tab <- table(listings$room_type)
downtown_tab <- table(listings$neigh_DT)
northern_tab <- table(listings$neigh_NO)
central_tab <- table(listings$neigh_CT)
other_neigh_tab <- table(listings$neigh_Other)
property_high_tab <- table(listings$property_high)
property_low_tab <- table(listings$property_low)


superhost_tab
```

```
##
##    f    t
## 2643 3048
```

```r
property_tab
```

```
##
##         Aparthotel          Apartment  Bed and breakfast     Boutique hotel
##                 36               2066                 41                153
##           Bungalow             Cabin             Castle        Condominium
##                 10                  2                  4                582
##            Cottage        Dome house        Earth house        Guest suite
##                 11                  1                  2                528
##         Guesthouse             Hostel              Hotel              House
##                 36                 71                102               1762
##             In-law              Loft              Other             Resort
##                  1                 60                 20                  7
## Serviced apartment         Tiny house          Townhouse              Villa
##                 82                  2                105                  7
```

```r
room_tab
```

```
##
## Entire home/apt      Hotel room    Private room     Shared room
##            3144             143            2269             135
```

```r
downtown_tab
```

```
##
##    0    1
## 4354 1337
```

```r
northern_tab
```

```
##
##    0    1
## 5058  633
```

```r
central_tab
```

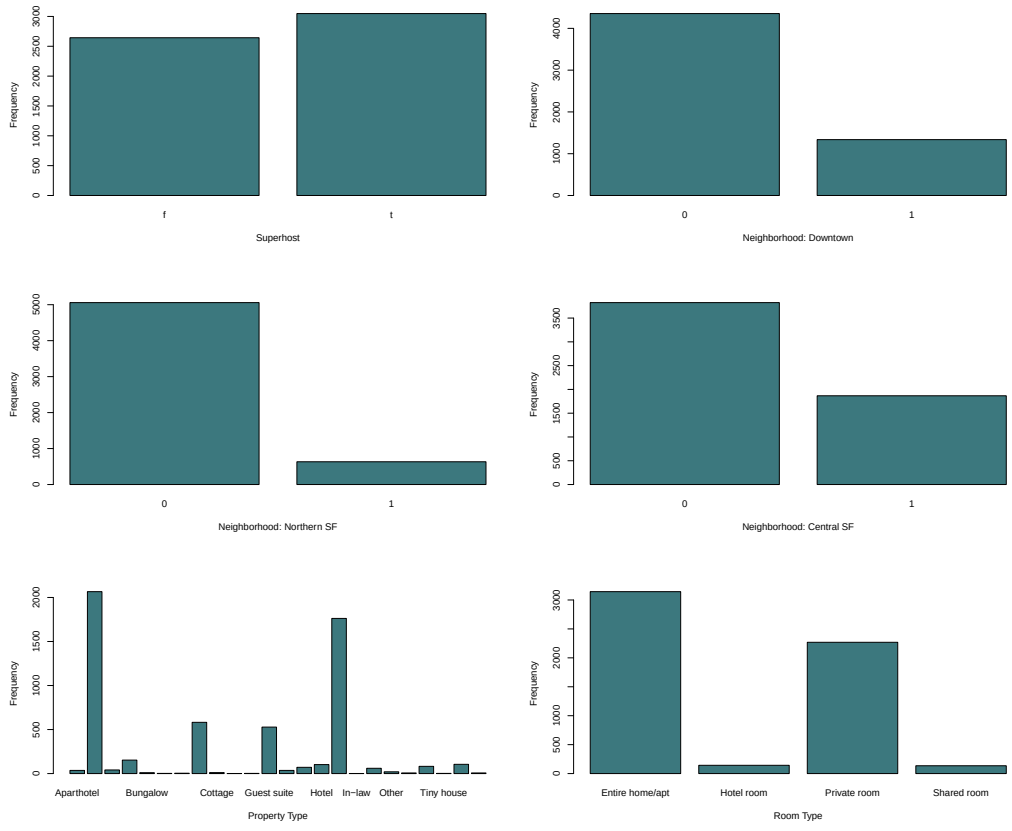
```
##
##    0    1
```

```
## 3825 1866
```

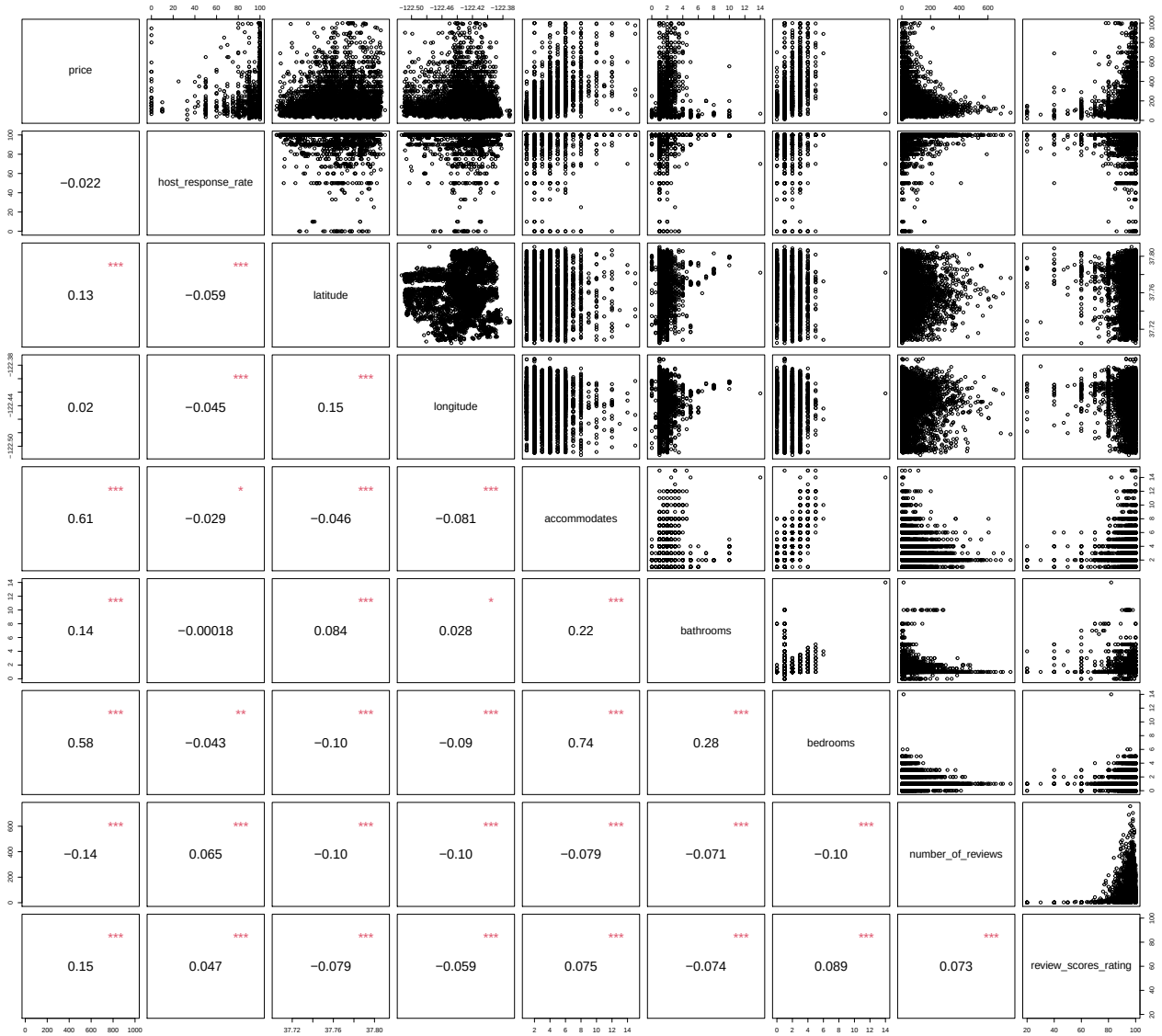
```
other_neigh_tab
```

```
##
##    0    1
## 3836 1855
```

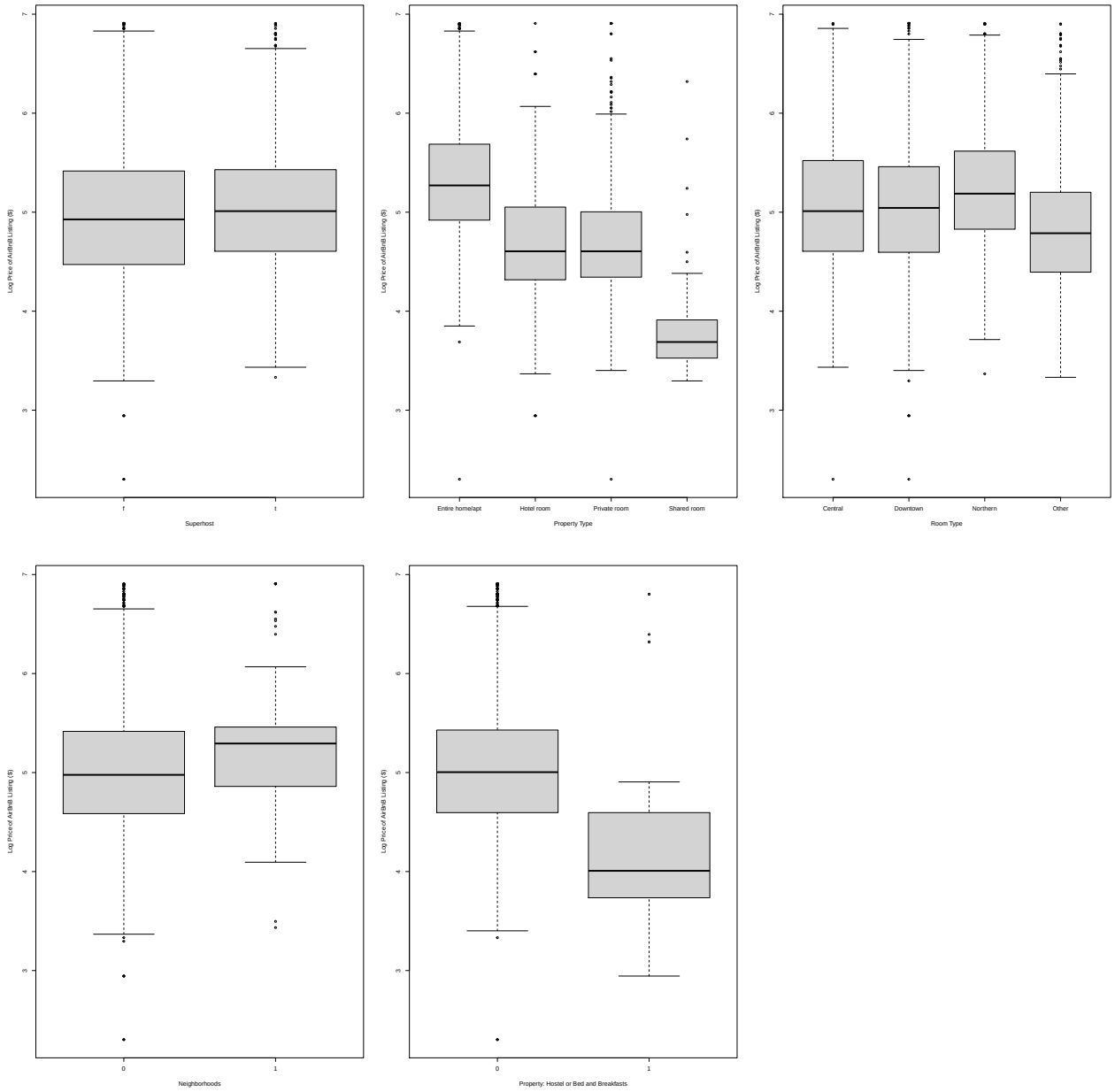
```
# graphical display of categorical variables
par(mfrow=c(3,2))
barplot(superhost_tab, main="",
          xlab="Superhost", ylab="Frequency",
          col="#3C787E")
barplot(downtown_tab, main="",
          xlab="Neighborhood: Downtown", ylab="Frequency",
          col="#3C787E")
barplot(northern_tab, main="",
          xlab="Neighborhood: Northern SF", ylab="Frequency",
          col="#3C787E")
barplot(central_tab, main="",
          xlab="Neighborhood: Central SF", ylab="Frequency",
          col="#3C787E")
barplot(property_tab, main="",
          xlab="Property Type", ylab="Frequency",
          col="#3C787E")
barplot(room_tab, main="",
          xlab="Room Type", ylab="Frequency",
          col="#3C787E")
```



```r
## Multivariate EDA

# define panel.cor function to use as personalized pairs plot
panel.cor <- function(x, y, digits=2, prefix="", cex.cor) {
  usr <- par("usr"); on.exit(par(usr))
  par(usr = c(0, 1, 0, 1))
  r <- cor(x, y)
  txt <- format(c(r, 0.123456789), digits=digits)[1]
  txt <- paste(prefix, txt, sep="")
  if(missing(cex.cor)) cex <- 2
  test <- cor.test(x, y)
  signif <- symnum(test$p.value, corr=FALSE, na=FALSE,
                   cutpoints = c(0, 0.001, 0.01, 0.05, 0.1, 1),
                   symbols = c("***", "**", "*", ".", " "))
  text(0.5, 0.5, txt, cex=cex)
  text(0.8, 0.8, signif, cex=cex, col=2)
}

# relationship between quantitative variables
pairs(listings_quant, lower.panel=panel.cor)
```



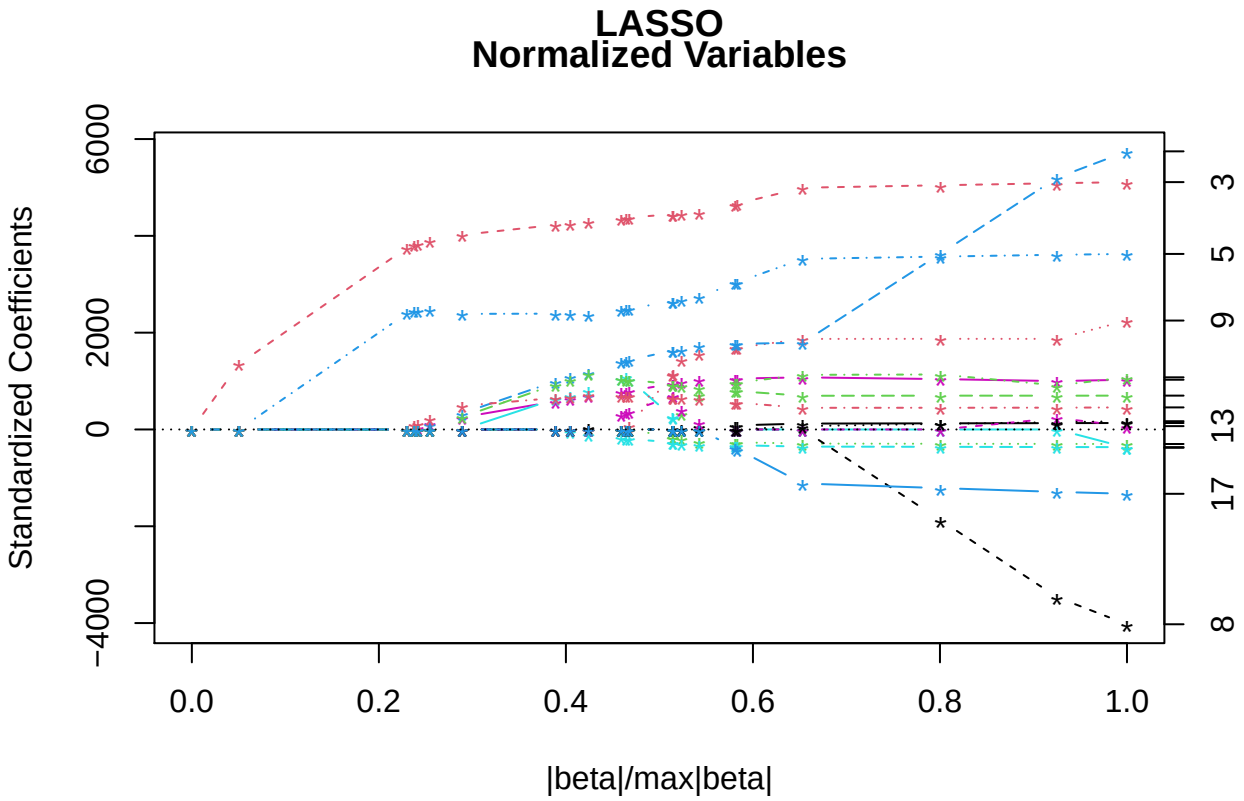

```
# relationship between categorical variables
par(mfrow=c(2,3))
catvars <- with(
  listings,
  cbind(host_is_superhost, room_type, neigh_PLOT, property_high, property_low));
catnum <- 5
catnames <-
  c('Superhost', 'Property Type', 'Room Type',
    'Neighborhoods','Property: Hostel or Bed and Breakfasts')

for(i in 1:catnum) {
  boxplot(log(listings$price) ~ catvars[, i], main="",
          ylab="Log Price of AirBnB Listing ($)",
          xlab=catnames[i])
}
```




```
#all variables + two interaction terms
lm.all <- lm(log(price)~1 + host_response_rate +accommodates+bathrooms+bedrooms+log(number_of_reviews)+
               review_scores_rating+
                 neigh_NO+neigh_DT+neigh_CT+
                 review_scores_rating*neigh_NO+review_scores_rating*neigh_DT+
                 review_scores_rating*neigh_CT+bedrooms*neigh_NO+bedrooms*neigh_DT+bedrooms*neigh_CT+
               accommodates*bedrooms,data=listings)
library(lars)
```

```
## Loaded lars 1.2
```

```
obj.norm<-lars(model.matrix(lm.all),listings$price,trace=F)
plot(obj.norm,breaks=F,main="Normalized Variables")
```




```
predict(obj.norm,type="coef")$coefficients[1:10,]
```

```
##       (Intercept) host_response_rate accommodates bathrooms bedrooms
##  [1,]           0                  0     0.000000         0  0.00000
##  [2,]           0                  0     9.770993         0  0.00000
##  [3,]           0                  0    27.125633         0 36.10102
##  [4,]           0                  0    27.598684         0 36.76825
##  [5,]           0                  0    27.763964         0 36.96081
##  [6,]           0                  0    28.140540         0 37.26490
##  [7,]           0                  0    29.019835         0 35.85219
##  [8,]           0                  0    30.513498         0 35.91768
##  [9,]           0                  0    30.741179         0 35.83690
## [10,]           0                  0    31.054190         0 35.46667
##       log(number_of_reviews) review_scores_rating neigh_NO neigh_DT neigh_CT
##  [1,]              0.0000000           0.00000000        0        0        0
##  [2,]              0.0000000           0.00000000        0        0        0
##  [3,]              0.0000000           0.00000000        0        0        0
##  [4,]              0.0000000           0.00000000        0        0        0
##  [5,]              0.0000000           0.06373127        0        0        0
##  [6,]              0.0000000           0.19745141        0        0        0
##  [7,]              0.0000000           0.42974959        0        0        0
##  [8,]              0.0000000           0.99486667        0        0        0
##  [9,]             -0.3508457           1.08709197        0        0        0
## [10,]             -0.8009563           1.20757679        0        0        0
##       review_scores_rating:neigh_NO review_scores_rating:neigh_DT
##  [1,]                    0.00000000                     0.0000000
##  [2,]                    0.00000000                     0.0000000
##  [3,]                    0.00000000                     0.0000000
##  [4,]                    0.00000000                     0.0000000
```

```
##  [5,]                    0.00000000                    0.0000000
##  [6,]                    0.04050098                    0.0000000
##  [7,]                    0.14795223                    0.0000000
##  [8,]                    0.43760001                    0.2105076
##  [9,]                    0.47717235                    0.2353478
## [10,]                    0.51477259                    0.2641259
##       review_scores_rating:neigh_CT bedrooms:neigh_NO bedrooms:neigh_DT
##  [1,]                             0         0.0000000          0.000000
##  [2,]                             0         0.0000000          0.000000
##  [3,]                             0         0.0000000          0.000000
##  [4,]                             0         0.0000000          2.161939
##  [5,]                             0         0.0000000          2.989915
##  [6,]                             0         0.0000000          5.114640
##  [7,]                             0         0.0000000         11.165983
##  [8,]                             0         0.0000000         14.895839
##  [9,]                             0         0.0000000         15.575196
## [10,]                             0         0.9211392         16.683650
##       bedrooms:neigh_CT accommodates:bedrooms
##  [1,]          0.000000                     0
##  [2,]          0.000000                     0
##  [3,]          0.000000                     0
##  [4,]          0.000000                     0
##  [5,]          0.000000                     0
##  [6,]          0.000000                     0
##  [7,]          4.225563                     0
##  [8,]         15.136597                     0
##  [9,]         16.631140                     0
## [10,]         18.633538                     0
```

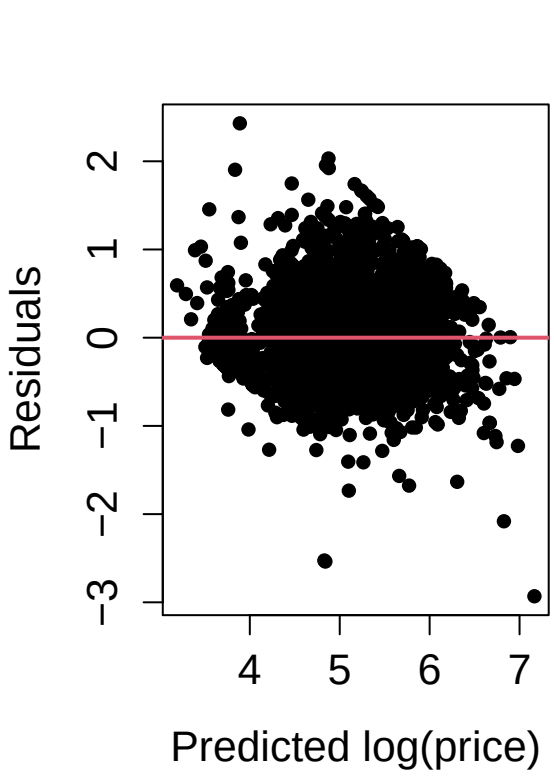
```r
lm.lasso <- lm(log(price)~ accommodates +
                host_response_rate +
                bedrooms + bathrooms +
                log(number_of_reviews) +
                review_scores_rating +
                room_type + property_high+
                neigh_NO + neigh_DT + neigh_CT+
                room_type*neigh_NO +
                room_type*neigh_DT+
                room_type*neigh_CT+
                bedrooms*neigh_NO+
                bedrooms*neigh_DT+
                bedrooms*neigh_CT,
              data=listings)
summary(lm.lasso)
```

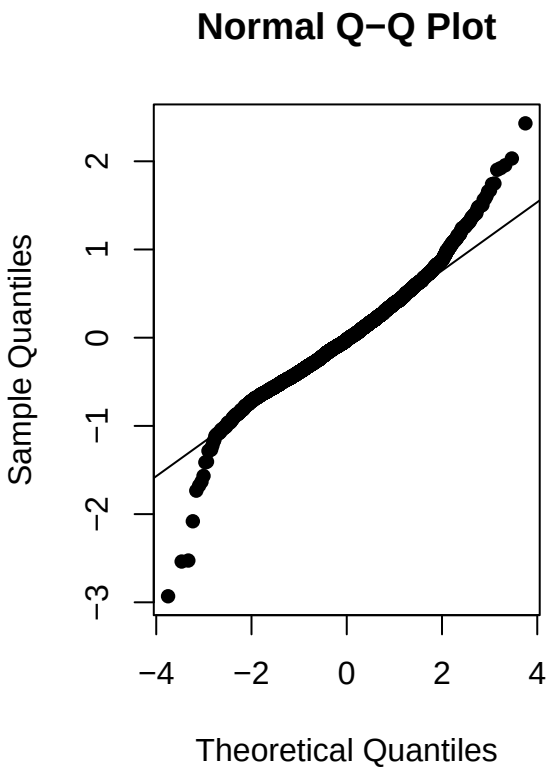
```
##
## Call:
## lm(formula = log(price) ~ accommodates + host_response_rate +
##     bedrooms + bathrooms + log(number_of_reviews) + review_scores_rating +
##     room_type + property_high + neigh_NO + neigh_DT + neigh_CT +
##     room_type * neigh_NO + room_type * neigh_DT + room_type *
##     neigh_CT + bedrooms * neigh_NO + bedrooms * neigh_DT + bedrooms *
##     neigh_CT, data = listings)
##
## Residuals:
```

```
##      Min       1Q   Median       3Q      Max
## -2.93119 -0.27895 -0.02382  0.24495  2.43002
##
## Coefficients: (1 not defined because of singularities)
##                                 Estimate Std. Error t value Pr(>|t|)
## (Intercept)                    3.1361223  0.0897769  34.932  < 2e-16 ***
## accommodates                   0.1015379  0.0049849  20.369  < 2e-16 ***
## host_response_rate             0.0020695  0.0005372   3.852 0.000118 ***
## bedrooms                       0.2047027  0.0145409  14.078  < 2e-16 ***
## bathrooms                     -0.0181188  0.0067501  -2.684 0.007291 **
## log(number_of_reviews)         0.0054381  0.0033815   1.608 0.107854
## review_scores_rating           0.0114706  0.0007375  15.554  < 2e-16 ***
## room_typeHotel room           -0.6108317  0.0609838 -10.016  < 2e-16 ***
## room_typePrivate room         -0.4042241  0.0213007 -18.977  < 2e-16 ***
## room_typeShared room          -1.1003411  0.0966175 -11.389  < 2e-16 ***
## property_high                  0.4442890  0.0426173  10.425  < 2e-16 ***
## neigh_NO                       0.4381751  0.0418227  10.477  < 2e-16 ***
## neigh_DT                       0.2782316  0.0353183   7.878 3.96e-15 ***
## neigh_CT                       0.2021820  0.0333460   6.063 1.42e-09 ***
## room_typeHotel room:neigh_NO   0.6001965  0.1282139   4.681 2.92e-06 ***
## room_typePrivate room:neigh_NO 0.0354034  0.0415407   0.852 0.394107
## room_typeShared room:neigh_NO  0.1064205  0.1967172   0.541 0.588541
## room_typeHotel room:neigh_DT   0.2264896  0.0790781   2.864 0.004197 **
## room_typePrivate room:neigh_DT 0.1509524  0.0332603   4.539 5.78e-06 ***
## room_typeShared room:neigh_DT -0.2030321  0.1088186  -1.866 0.062123 .
## room_typeHotel room:neigh_CT           NA         NA      NA       NA
## room_typePrivate room:neigh_CT 0.0596335  0.0298478   1.998 0.045773 *
## room_typeShared room:neigh_CT -0.0202623  0.1218411  -0.166 0.867926
## bedrooms:neigh_NO             -0.0837749  0.0200960  -4.169 3.11e-05 ***
## bedrooms:neigh_DT              0.0219216  0.0196192   1.117 0.263892
## bedrooms:neigh_CT              0.0108560  0.0166763   0.651 0.515086
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.4128 on 5666 degrees of freedom
## Multiple R-squared:  0.6022, Adjusted R-squared:  0.6005
## F-statistic: 357.4 on 24 and 5666 DF,  p-value: < 2.2e-16
```

```
# note low adjusted r sq
```

```
par(mfrow=c(1,2))
# lasso model diagnostics
plot(lm.lasso$fit,lm.lasso$res,pch=16,cex.lab=1.3,cex.axis=1.3,
       xlab="Predicted log(price)",ylab="Residuals")
abline(h=0,lwd=2,col=2)
#not fitted on tails...
qqnorm(lm.lasso$res,pch=16); qqline(lm.lasso$res)
```





Map Plots

```
library("ggmap")
```

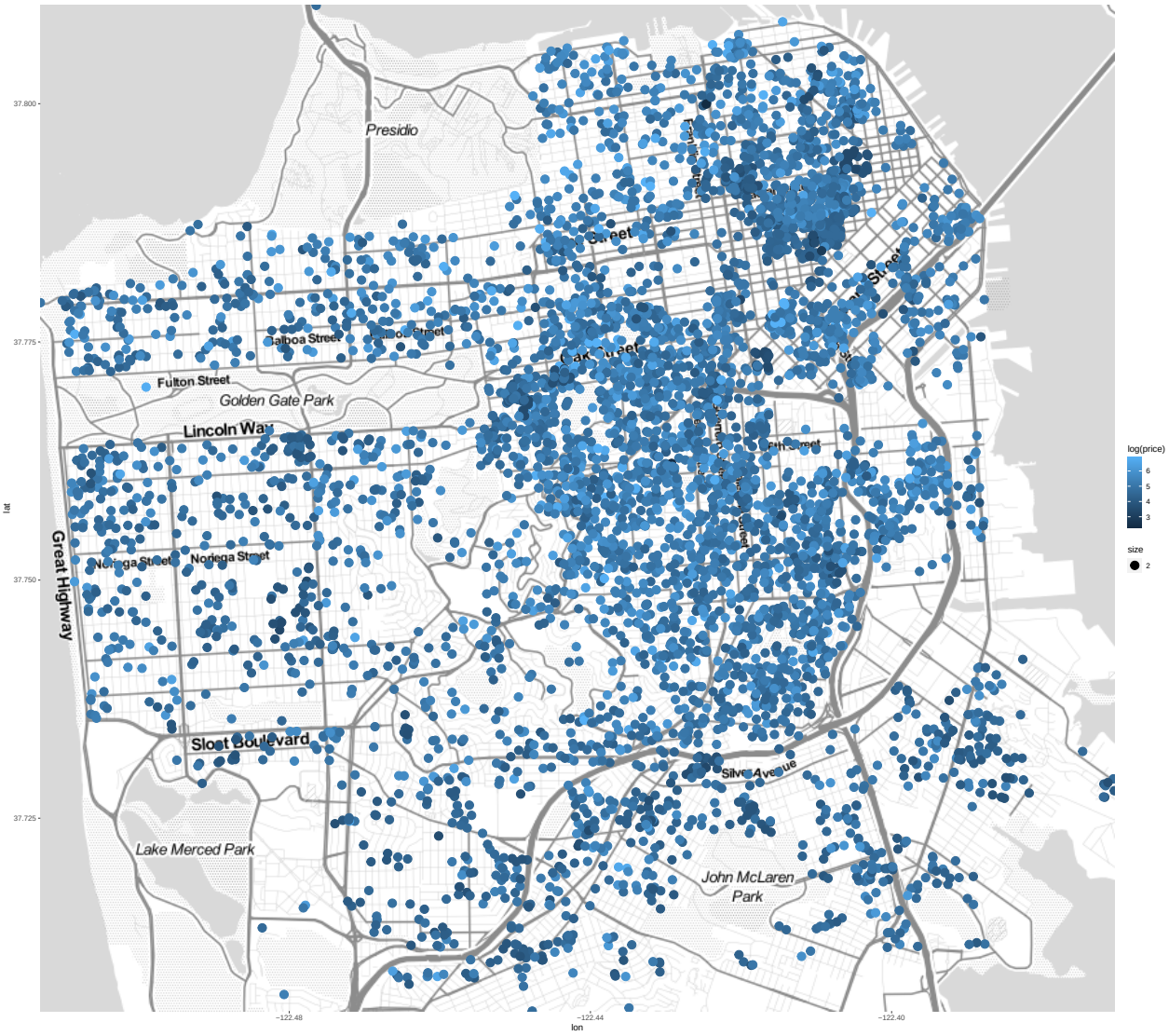
```
## Loading required package: ggplot2
```

```
## Google's Terms of Service: https://cloud.google.com/maps-platform/terms/.
```

```
## Please cite ggmap if you use it! See citation("ggmap") for details.
```

```
library("ggplot2")
library("forcats")

sf <- c(left =  min(listings$longitude),
        bottom = min(listings$latitude),
        right = max(listings$longitude),
        top = max(listings$latitude))
ggmap(get_stamenmap(sf, zoom = 13, maptype = "toner-lite")) +
  geom_point(data = listings, aes(x = longitude, y = latitude,
                                  color = log(price), size = 2))
```

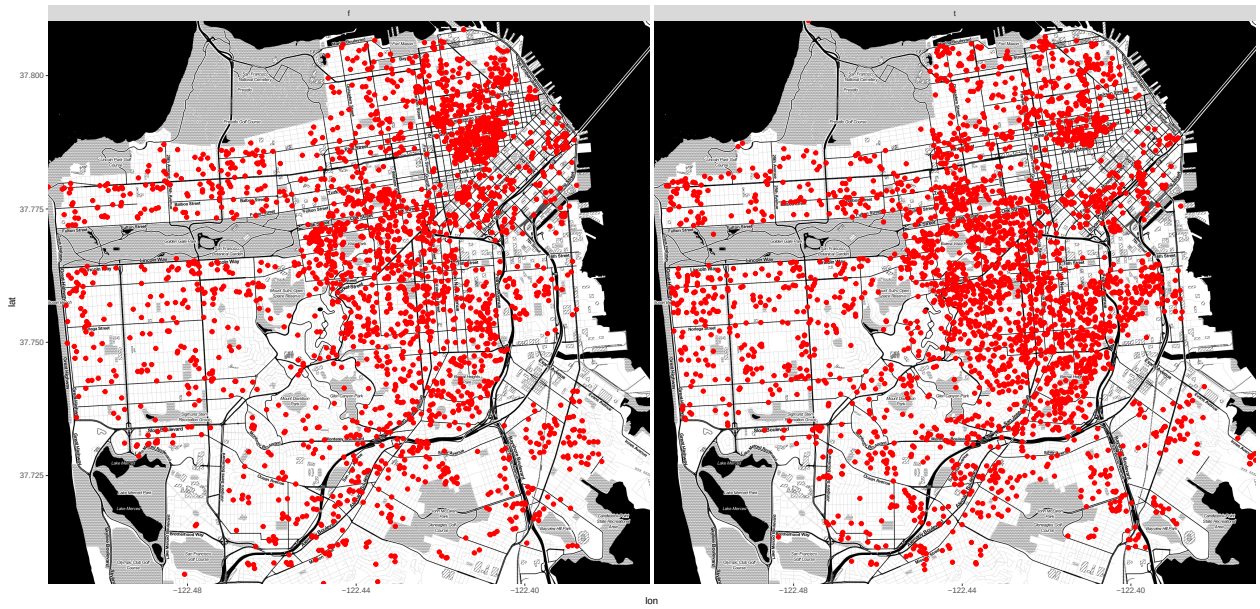
```
## Map tiles by Stamen Design, under CC BY 3.0. Data by OpenStreetMap, under ODbL.
```




```
ggmap(get_stamenmap(sf, zoom = 14, maptype = "toner"), legend = "right") +
  geom_point(data = listings, aes(x = longitude, y = latitude), color = "red", size = 2)+
  facet_wrap(~ host_is_superhost)
```

```
## Map tiles by Stamen Design, under CC BY 3.0. Data by OpenStreetMap, under ODbL.
```

```
## 49 tiles needed, this may take a while (try a smaller zoom).
```


f
t
37.800
37.775
37.750
37.725
lat
−122.48
−122.44
−122.40
lon