# Appendix part 1

*47602, 35261, 38844, 41781*

*12/12/2019*

## Data Cleaning and Transformation

The code belows cleans and transforms the original dataset from the website.

```
# Large file, code not ran
amsterdam <- read_csv('listingsamsterdam.csv')

amsterdam <- select(amsterdam, price, review_scores_rating, host_since,
                    host_is_superhost, neighbourhood_cleansed, host_listings_count,
                    host_identity_verified, room_type,
                    bathrooms, bedrooms, minimum_nights,
                    number_of_reviews, cancellation_policy, instant_bookable,accommodates, weekly_price

# Data transformation
amsterdam$price <- as.numeric(gsub('\\$|,', '', amsterdam$price))
amsterdam$weekly_price <- as.numeric(gsub('\\$|,', '', amsterdam$weekly_price))
amsterdam$monthly_price <- as.numeric(gsub('\\$|,', '', amsterdam$monthly_price))
amsterdam$cleaning_fee <- as.numeric(gsub('\\$|,', '', amsterdam$cleaning_fee))

amsterdam$host_since <- as.Date(amsterdam$host_since)
amsterdam <- mutate(amsterdam, host_is_superhost = ifelse(host_is_superhost == TRUE, 1 , 0))
amsterdam <- mutate(amsterdam, host_identity_verified = ifelse(host_identity_verified == TRUE, 1 , 0))

amsterdam$location_3ways <- ifelse(amsterdam$neighbourhood_cleansed == 'Centrum-West' |
                                     amsterdam$neighbourhood_cleansed == 'Centrum-Oost' |
                                     amsterdam$neighbourhood_cleansed == 'Zuid', "near_centre",
                                   ifelse(amsterdam$neighbourhood_cleansed == 'Bijlmer-Oost' |
                                            amsterdam$neighbourhood_cleansed == 'Gaasperdam - Driemond'
                                            amsterdam$neighbourhood_cleansed == 'Bijlmer-Centrum' |
                                            amsterdam$neighbourhood_cleansed == 'Osdorp' |
                                            amsterdam$neighbourhood_cleansed == 'Geuzenveld - Slotermee
                                            amsterdam$neighbourhood_cleansed == 'Slotervaart' |
                                            amsterdam$neighbourhood_cleansed == 'De Aker - Nieuw Sloten
                                            amsterdam$neighbourhood_cleansed == 'Bos en Lommer' |
                                            amsterdam$neighbourhood_cleansed == 'Noord-Oost' |
                                            amsterdam$neighbourhood_cleansed == 'Noord-West' |
                                            amsterdam$neighbourhood_cleansed == 'Oostelijk Havengebied
                                          "far_from_centre", "Moderate"))

amsterdam$realprice <- rep(NA, length(amsterdam$price))

for (i in 1:length(amsterdam$realprice)){
  if (amsterdam$minimum_nights[i] > 27){
    if (!is.na(amsterdam$monthly_price[i])){
    amsterdam$realprice[i] <- amsterdam$monthly_price[i]/30
    } else {
      amsterdam$realprice[i] <- amsterdam$price[i]
```

```
    }
  } else if (amsterdam$minimum_nights[i] > 6) {
    if (!is.na(amsterdam$weekly_price[i])){
    amsterdam$realprice[i] <- amsterdam$weekly_price[i]/7
    } else {
      amsterdam$realprice[i] <- amsterdam$price[i]
    }
  } else {
  amsterdam$realprice[i] <- amsterdam$price[i]
  }
}

# review_scores_rating has 2565 NA's,
# cleaning_fee 3611 NA's, removing those NAs after removing r_s_r NA's leads to a drop of ~2000 observa
# amsterdam <- filter(amsterdam, !is.na(cleaning_fee))

# 5 NAs in host_since, 5 NAs in host_is_superhost, 5 NAs in host_listingscount
# Host response rate and time have 8536 NA's, removed them
# Experiences_offered contains only none, removed it
# 33 types of property, removed it

# Did not manage to convert host_verifications, removed it
# Mininum nights has maximum of 1001 (outlier I suppose)
# 5 types of cancellation policy
# 7NAs bathrooms, 14NAs bedrooms, 8 NAs beds

drops <- c("weekly_price", "monthly_price")
amsterdam <- amsterdam[ , !(names(amsterdam) %in% drops)]

# Add in host's "age" on AirBnb
Date_scrap <- as.Date("14/09/19","%d/%m/%y")
amsterdam$host_since_duration <- Date_scrap-amsterdam$host_since

amsterdam <- drop_na(amsterdam)
check <- amsterdam[amsterdam$realprice < 1,] # There is an outlier with price = 0

amsterdam <- amsterdam[amsterdam$realprice > 1,]

amsterdam$logprice <- log(amsterdam$realprice)
```

## EDA

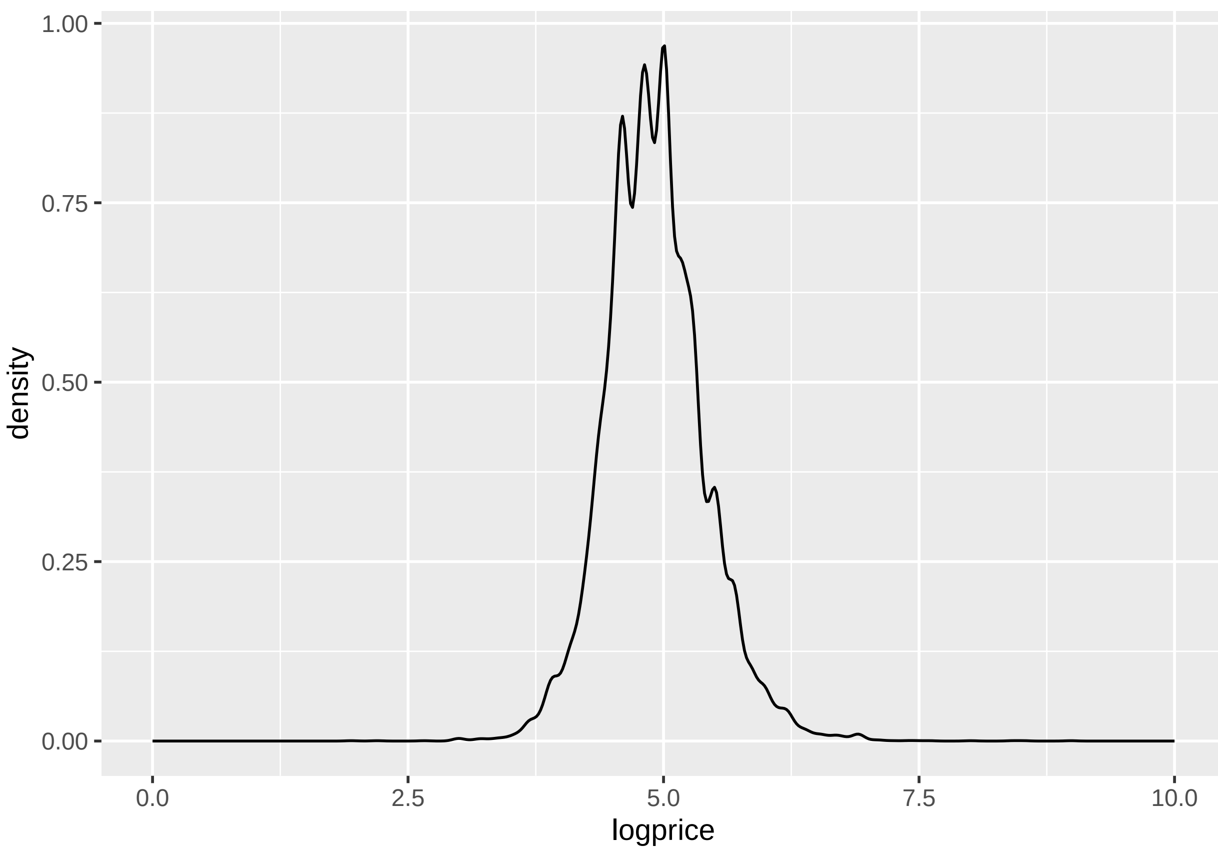
```
amsterdam <- read_csv('st443_final_data')
```

```
## Warning: Missing column names filled in: 'X1' [1]
```

```
## Parsed with column specification:
## cols(
##   X1 = col_double(),
##   review_scores_rating = col_double(),
##   host_is_superhost = col_double(),
##   host_listings_count = col_double(),
##   host_identity_verified = col_double(),
```

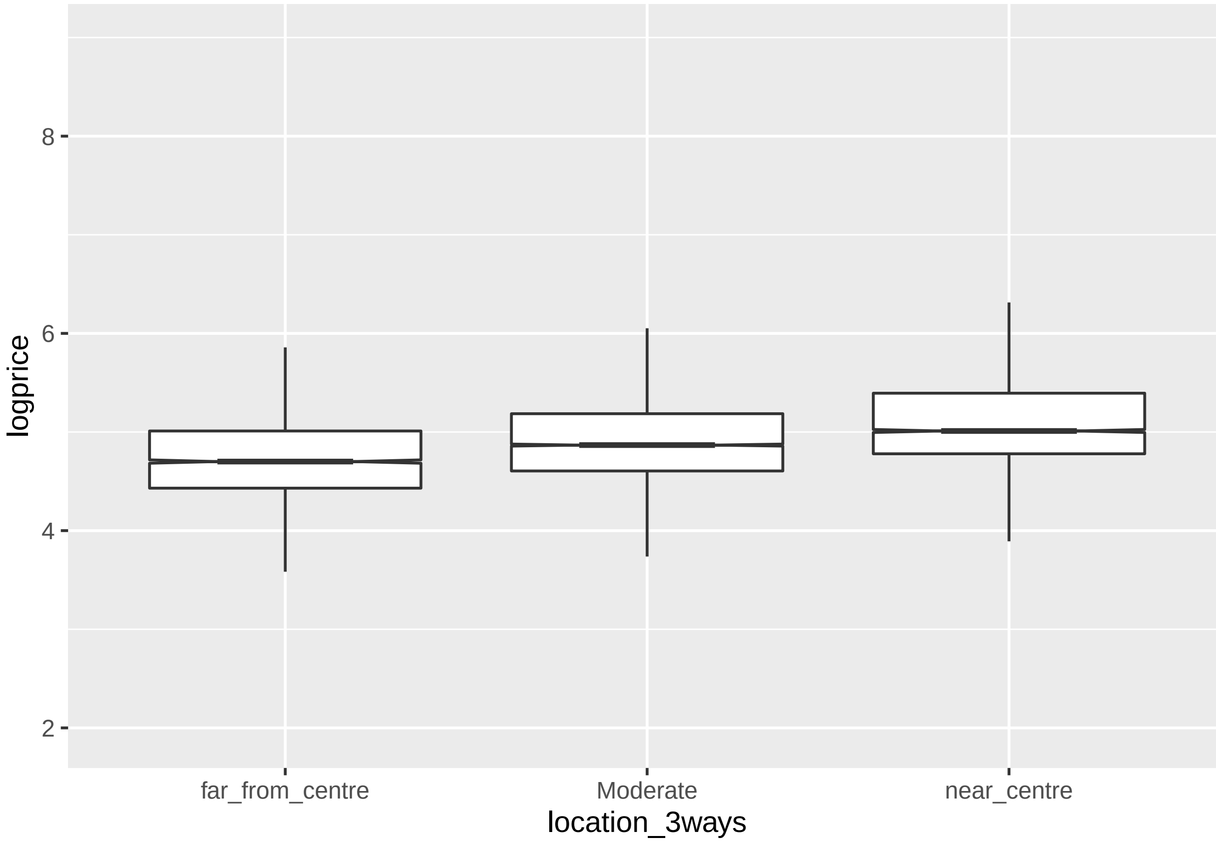
```
##   room_type = col_character(),
##   bathrooms = col_double(),
##   bedrooms = col_double(),
##   minimum_nights = col_double(),
##   number_of_reviews = col_double(),
##   cancellation_policy = col_character(),
##   instant_bookable = col_logical(),
##   cleaning_fee = col_double(),
##   location_3ways = col_character(),
##   realprice = col_double(),
##   host_since_duration = col_double(),
##   logprice = col_double()
## )
```

```
# Based on boxplots, there are too many outliers at different price points to be taken out.
# Histograms of numeric variables shows that all are skewed. There is no right way to set limits
# Using logprice transformation however, removes skewness in price. Subsequent OLS shows no outliers up
# Using price as the y variable, plot residuals/leverage and identify 2 outliers,
# they were remove and OLS re-iterated, with more outliers. It will be too long a process.

# ------------------------------------------------------------------------------------
# Price
# Using ggplots:
ggplot(amsterdam, aes(logprice)) +
  geom_freqpoly(stat='density') + xlim(0,10)
```



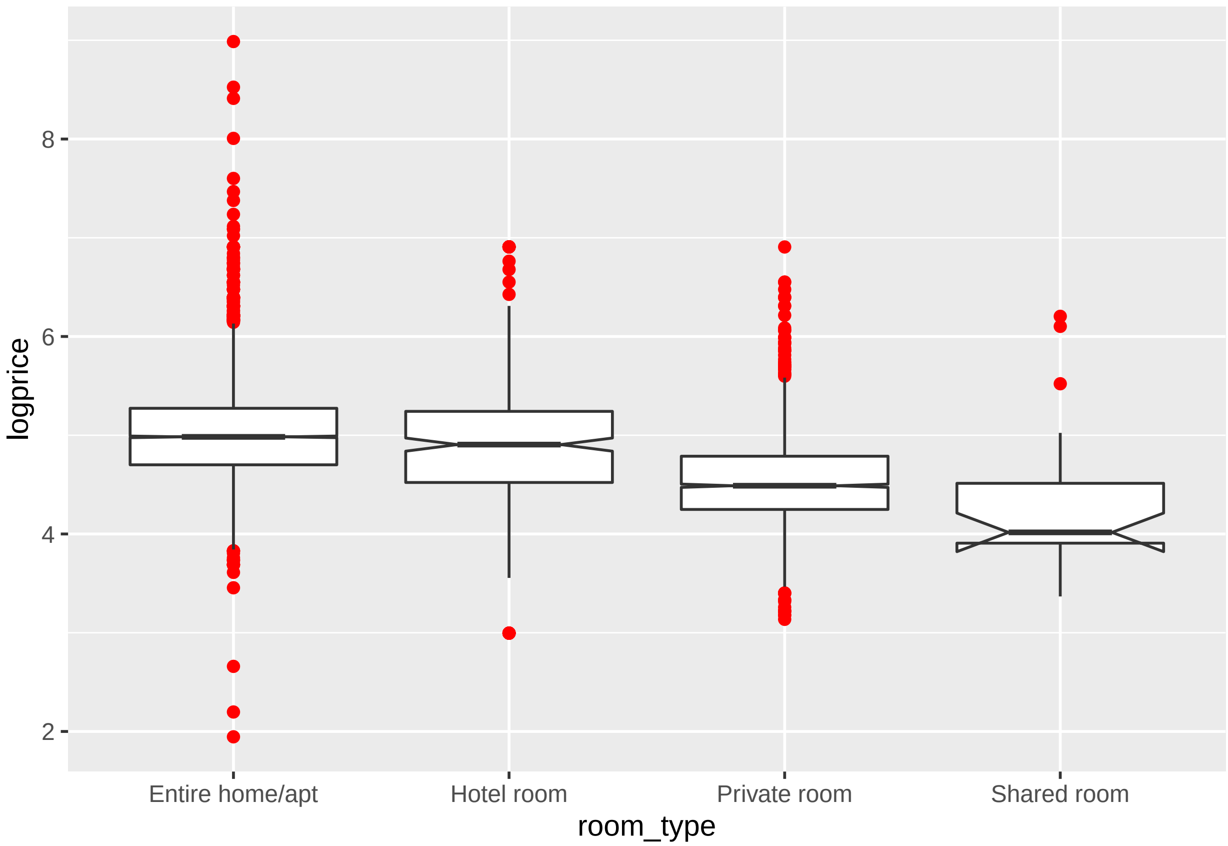

```
# log transformation of price result in a less skewed variable "price" without losing any data points
```

```r
#boxplot of location vs price

ggplot(amsterdam, aes(x=location_3ways, y=logprice)) +
  geom_boxplot(outlier.colour="dark blue", outlier.shape=16,
               outlier.size=2, notch=TRUE)
```



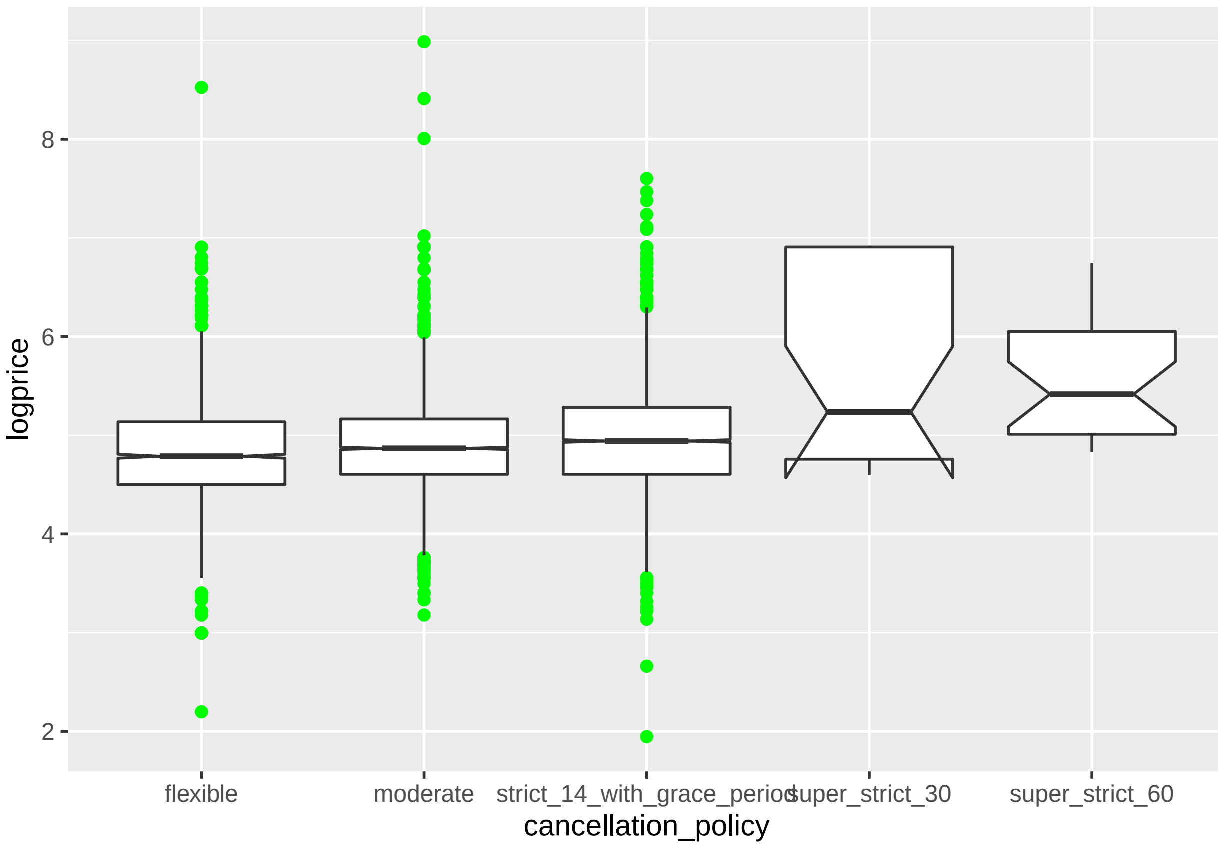

```r
# lesser outliers detected using boxplot of logprice / location

#boxplot of room_type vs price
ggplot(amsterdam, aes(x=room_type, y=logprice)) +
  geom_boxplot(outlier.colour="red", outlier.shape=16, outlier.size=2, notch=TRUE)
```

```
## notch went outside hinges. Try setting notch=FALSE.
```

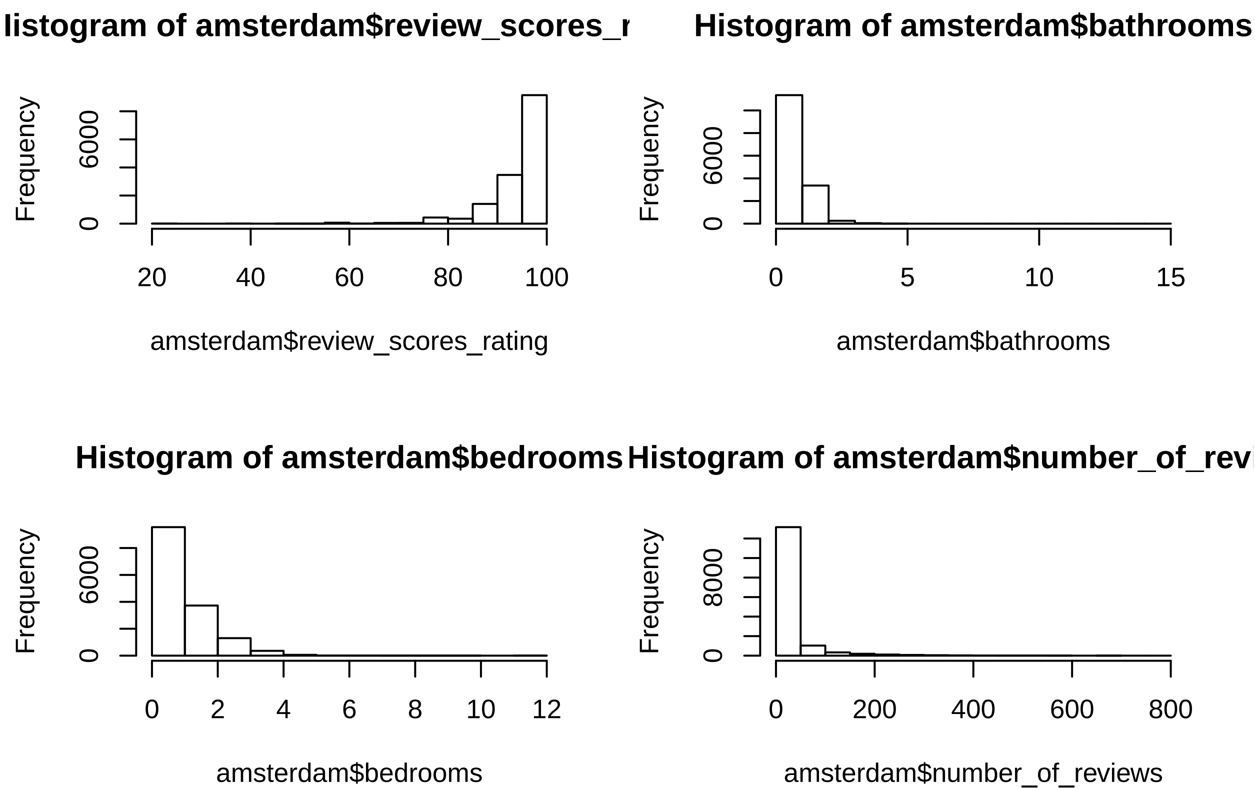



```
# lesser outliers detected using boxplot of logprice / location

#boxplot of cancellation_policy vs price
ggplot(amsterdam, aes(x=cancellation_policy, y=logprice)) +
  geom_boxplot(outlier.colour="green", outlier.shape=16, outlier.size=2, notch=TRUE)
```

```
## notch went outside hinges. Try setting notch=FALSE.
```




```r
# lesser outliers detected using boxplot of logprice / location

#histograms of numeric variables
par(mfrow=c(2,2))
hist(amsterdam$review_scores_rating)
hist(amsterdam$bathrooms)
hist(amsterdam$bedrooms)
hist(amsterdam$number_of_reviews)
```

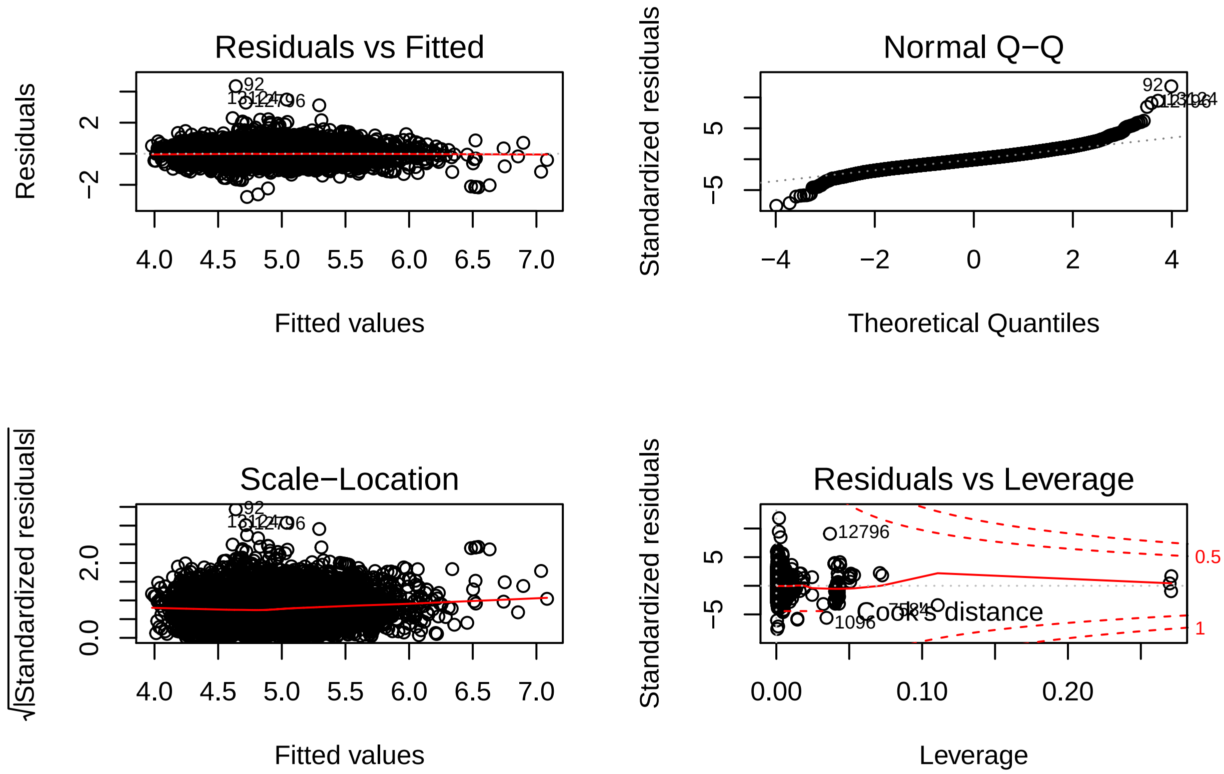



```r
# As expected, all are skewed.

#Using log price - interpretation of coefficients will be different
logPrice_ols <- lm(logprice ~ review_scores_rating + host_is_superhost +
                  host_listings_count + host_identity_verified +
                  room_type + bathrooms + bedrooms +
                  minimum_nights + number_of_reviews + cancellation_policy +
                  instant_bookable + host_since_duration + location_3ways + cleaning_fee, data=amsterda
summary(logPrice_ols)
```

```
##
## Call:
## lm(formula = logprice ~ review_scores_rating + host_is_superhost +
##     host_listings_count + host_identity_verified + room_type +
##     bathrooms + bedrooms + minimum_nights + number_of_reviews +
##     cancellation_policy + instant_bookable + host_since_duration +
##     location_3ways + cleaning_fee, data = amsterdam)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2.7812 -0.2309 -0.0147  0.2067  4.3489
##
## Coefficients:
##                                                  Estimate Std. Error t value
## (Intercept)                                     3.932e+00  4.868e-02  80.771
## review_scores_rating                            3.685e-03  4.915e-04   7.498
## host_is_superhost                               9.071e-02  8.623e-03  10.520
## host_listings_count                            -5.577e-04  1.070e-04  -5.213
## host_identity_verified                         -1.146e-03  6.593e-03  -0.174
```

```
## room_typeHotel room                           -7.004e-02  2.326e-02  -3.012
## room_typePrivate room                         -3.462e-01  9.113e-03 -37.990
## room_typeShared room                          -5.522e-01  7.543e-02  -7.320
## bathrooms                                      1.079e-01  8.778e-03  12.294
## bedrooms                                       1.661e-01  4.019e-03  41.331
## minimum_nights                                -2.834e-04  1.922e-04  -1.475
## number_of_reviews                             -3.453e-04  6.940e-05  -4.976
## cancellation_policymoderate                    1.529e-02  8.966e-03   1.706
## cancellation_policystrict_14_with_grace_period  5.472e-02  8.934e-03   6.125
## cancellation_policysuper_strict_30             6.686e-01  7.370e-02   9.072
## cancellation_policysuper_strict_60             2.900e-01  7.568e-02   3.831
## instant_bookableTRUE                           3.068e-02  7.546e-03   4.065
## host_since_duration                           -1.888e-05  4.712e-06  -4.006
## location_3waysModerate                         1.658e-01  8.051e-03  20.593
## location_3waysnear_centre                      3.533e-01  9.163e-03  38.560
## cleaning_fee                                   3.468e-03  1.492e-04  23.239
##                                               Pr(>|t|)
## (Intercept)                                    < 2e-16 ***
## review_scores_rating                          6.86e-14 ***
## host_is_superhost                              < 2e-16 ***
## host_listings_count                           1.88e-07 ***
## host_identity_verified                        0.861977
## room_typeHotel room                           0.002602 **
## room_typePrivate room                          < 2e-16 ***
## room_typeShared room                          2.60e-13 ***
## bathrooms                                      < 2e-16 ***
## bedrooms                                       < 2e-16 ***
## minimum_nights                                0.140276
## number_of_reviews                             6.58e-07 ***
## cancellation_policymoderate                   0.088094 .
## cancellation_policystrict_14_with_grace_period 9.28e-10 ***
## cancellation_policysuper_strict_30             < 2e-16 ***
## cancellation_policysuper_strict_60            0.000128 ***
## instant_bookableTRUE                          4.82e-05 ***
## host_since_duration                           6.21e-05 ***
## location_3waysModerate                         < 2e-16 ***
## location_3waysnear_centre                      < 2e-16 ***
## cleaning_fee                                   < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.3688 on 14997 degrees of freedom
## Multiple R-squared:  0.4223, Adjusted R-squared:  0.4216
## F-statistic: 548.2 on 20 and 14997 DF,  p-value: < 2.2e-16
```

```r
plot(logPrice_ols) # no outliers
```




```r
# Descriptive statistics command
lapply(amsterdam, summary)
```

```
## $X1
##     Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##        1    3755    7510    7510   11264   15018
##
## $review_scores_rating
##     Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    20.00   93.00   97.00   95.09  100.00  100.00
##
## $host_is_superhost
##     Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   0.0000  0.0000  0.0000  0.1794  0.0000  1.0000
##
## $host_listings_count
##     Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    0.000   1.000   1.000   5.619   1.000 932.000
##
## $host_identity_verified
##     Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   0.0000  0.0000  0.0000  0.4175  1.0000  1.0000
##
## $room_type
##     Length     Class      Mode
##      15018 character character
##
## $bathrooms
##     Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
```

```
##   0.000   1.000   1.000   1.169   1.000  15.000
##
## $bedrooms
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   0.000   1.000   1.000   1.471   2.000  12.000
##
## $minimum_nights
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
##    1.000    2.000    2.000    3.294    3.000 1001.000
##
## $number_of_reviews
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    0.00    5.00   12.00   27.49   27.00  786.00
##
## $cancellation_policy
##    Length     Class      Mode
##     15018 character character
##
## $instant_bookable
##    Mode   FALSE    TRUE
## logical   11342    3676
##
## $cleaning_fee
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    0.00   25.00   39.00   39.07   50.00  500.00
##
## $location_3ways
##    Length     Class      Mode
##     15018 character character
##
## $realprice
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##     7.0   100.0   132.0   156.3   180.0  8000.0
##
## $host_since_duration
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##      14    1231    1756    1699    2225    4007
##
## $logprice
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   1.946   4.605   4.883   4.919   5.193   8.987
```

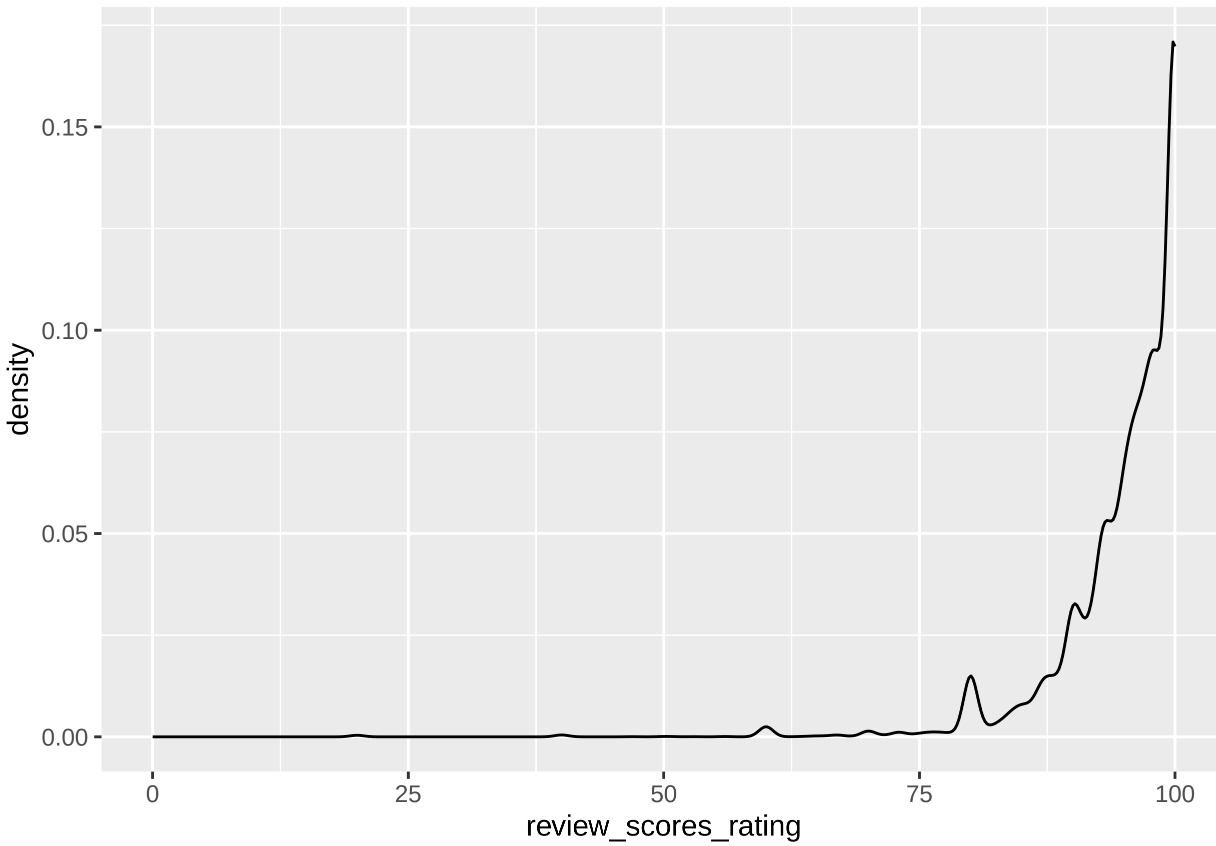
```r
# DATA EXPLORATION

#amsterdam %>% count(neighbourhood_cleansed) %>% arrange(desc(n)) %>% print(n=30)
amsterdam %>% count(room_type) %>% arrange(desc(n)) %>% print(n=30)
```

```
## # A tibble: 4 x 2
##   room_type           n
##   <chr>           <int>
## 1 Entire home/apt 12079
## 2 Private room     2624
## 3 Hotel room        291
## 4 Shared room        24
```

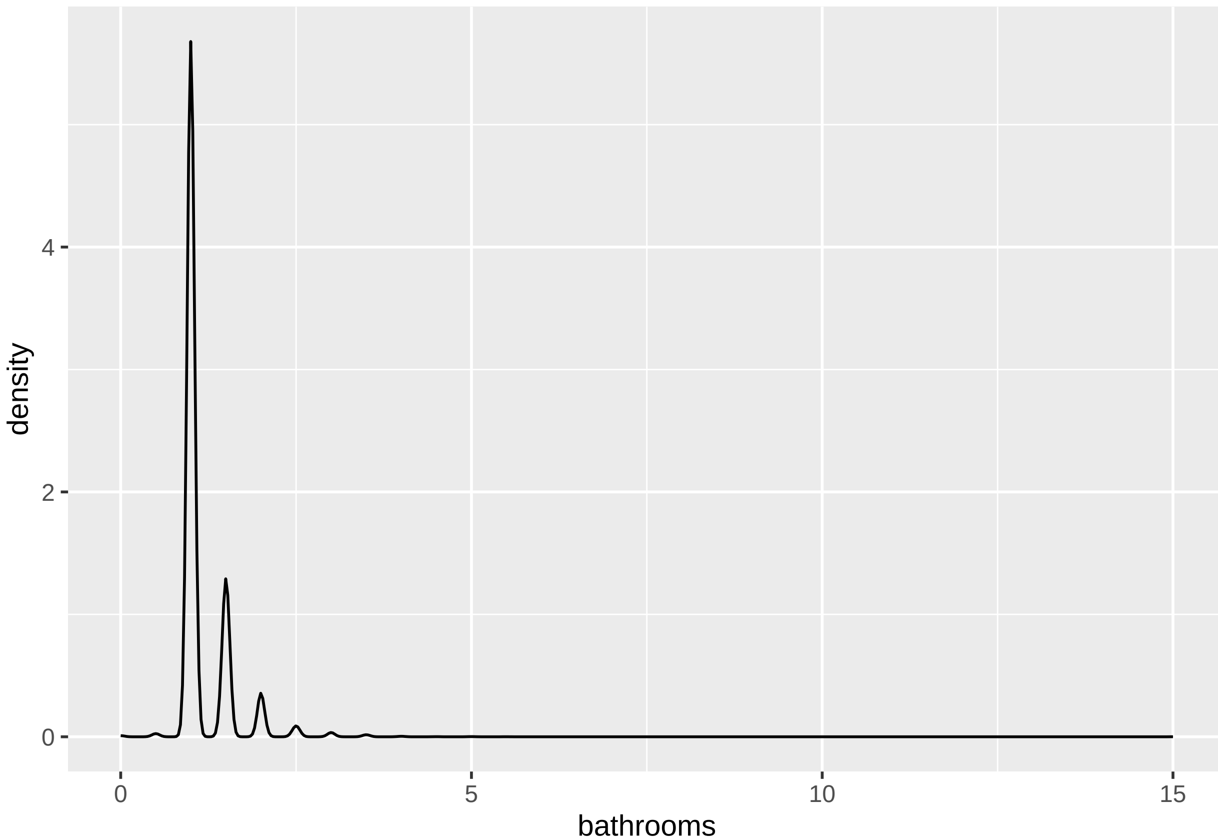
```r
amsterdam %>% count(cancellation_policy) %>% arrange(desc(n)) %>% print(n=30)
```

```
## # A tibble: 5 x 2
##   cancellation_policy            n
##   <chr>                      <int>
## 1 strict_14_with_grace_period  6621
## 2 moderate                    5893
## 3 flexible                    2453
## 4 super_strict_30               26
## 5 super_strict_60               25
```

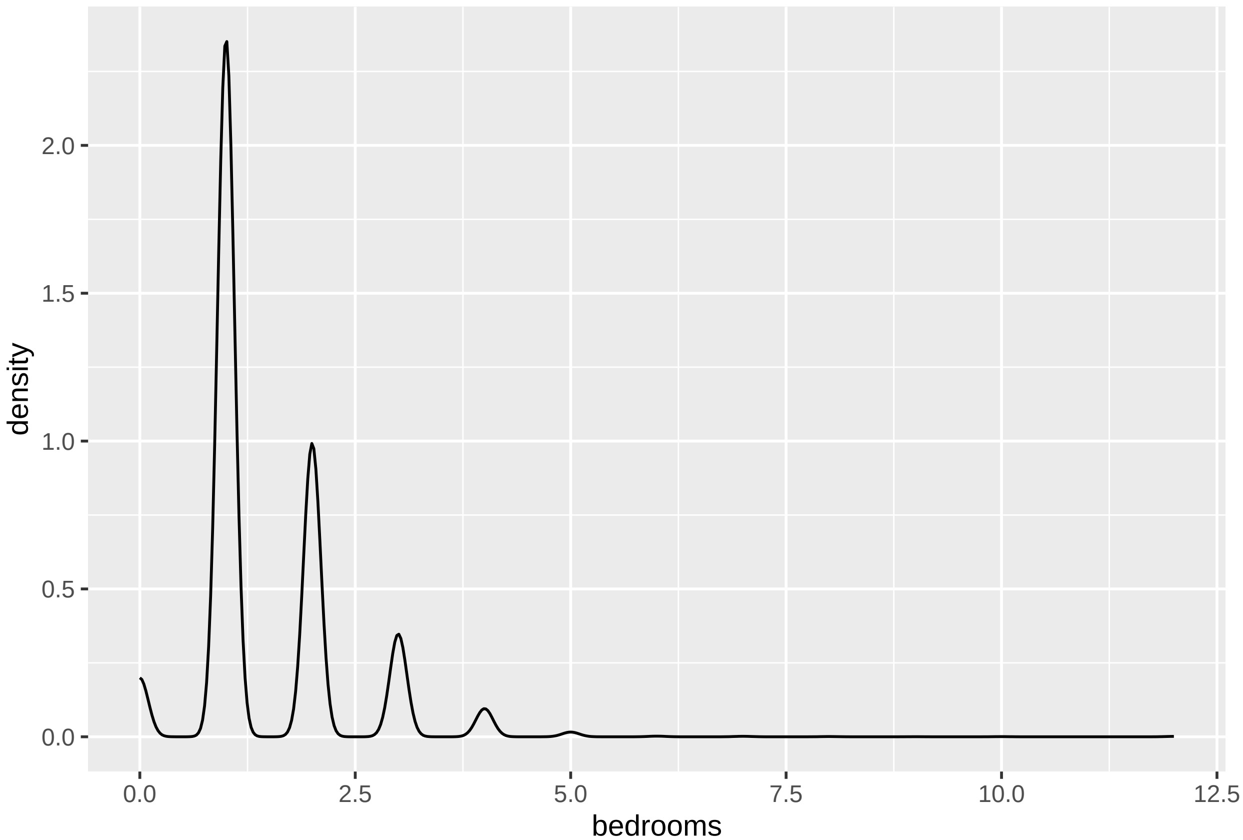
```r
# mean review rating is 95... extremely skewed and probably uninteresting density
ggplot(amsterdam, aes(review_scores_rating)) +
  geom_freqpoly(stat='density') + xlim(0,100)
```




```r
ggplot(amsterdam, aes(bathrooms)) +
  geom_freqpoly(stat='density')
```

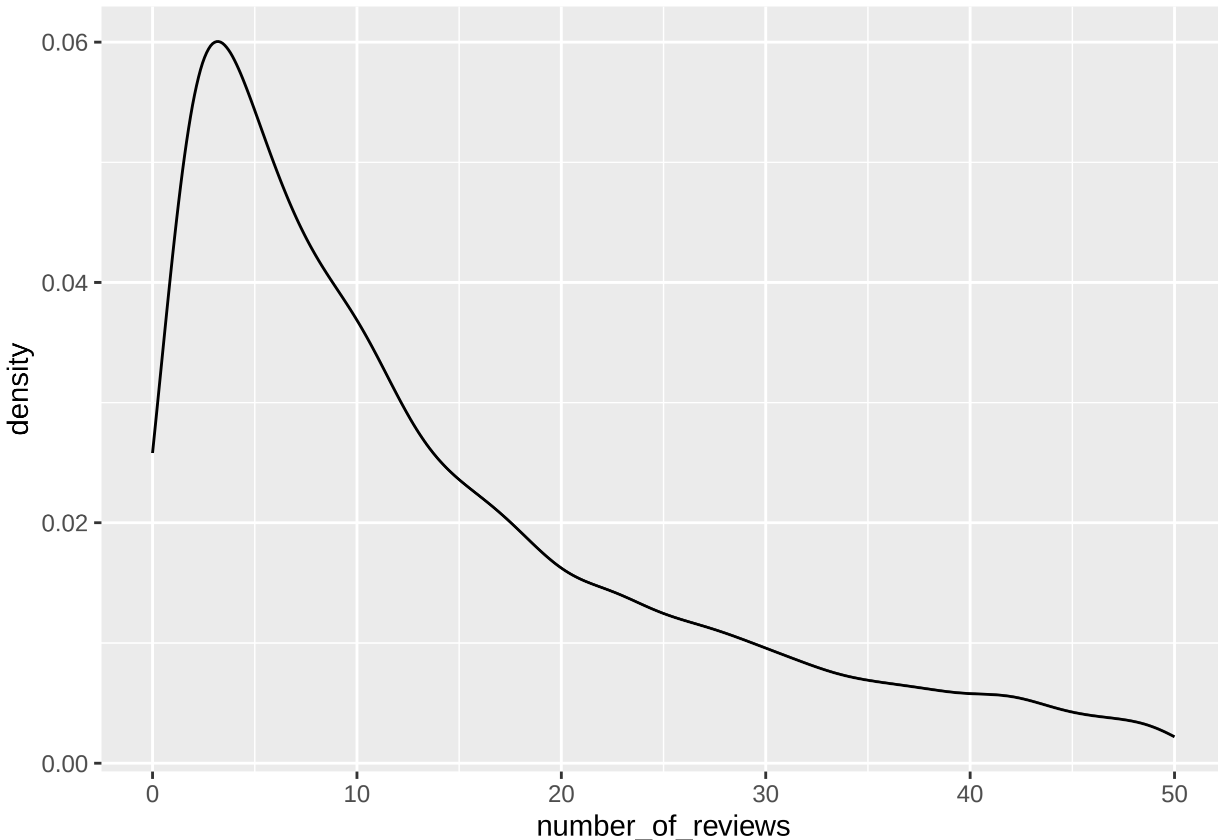



```r
ggplot(amsterdam, aes(bedrooms)) +
  geom_freqpoly(stat='density')
```



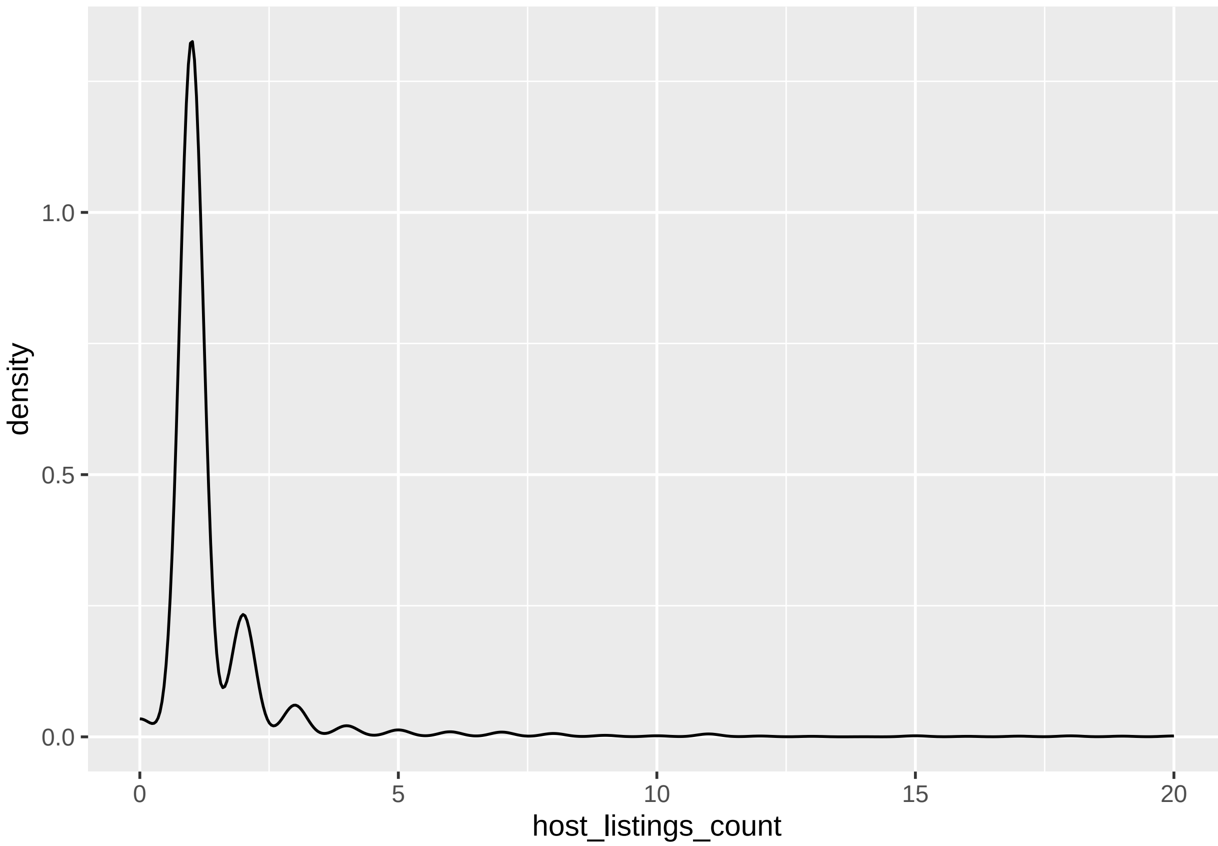

```r
# skewed to the right
ggplot(amsterdam, aes(number_of_reviews)) +
  geom_freqpoly(stat='density')  + xlim(0,50)
```

```
## Warning: Removed 1853 rows containing non-finite values (stat_density).
```



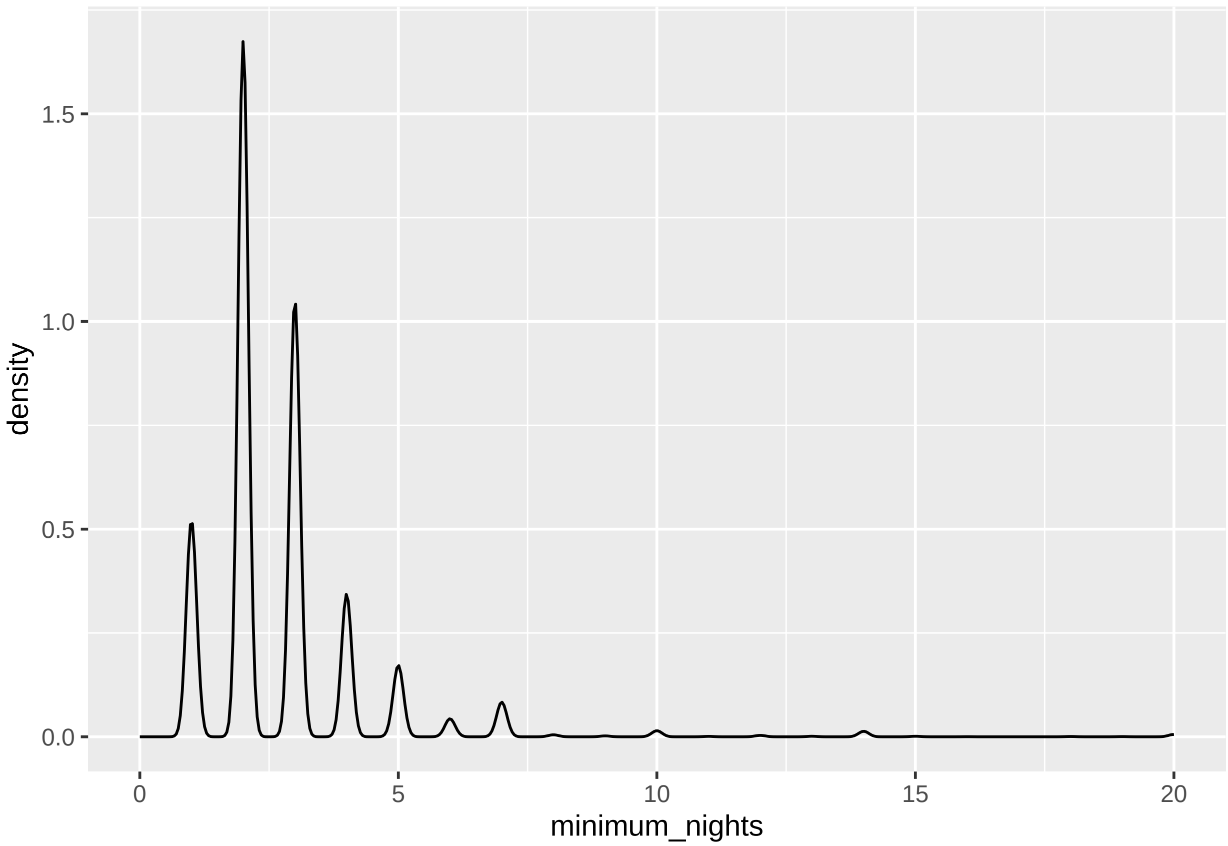

```
# Useless variables, will not add much
ggplot(amsterdam, aes(host_listings_count)) +
  geom_freqpoly(stat='density') + xlim(0,20)
```

```
## Warning: Removed 469 rows containing non-finite values (stat_density).
```




```r
# Not sure if interesting
ggplot(amsterdam, aes(minimum_nights)) +
  geom_freqpoly(stat='density') + xlim(0,20)
```

```
## Warning: Removed 113 rows containing non-finite values (stat_density).
```




```
# make three/four neighbourhoods in terms of price?
#amsterdam %>% group_by(neighbourhood_cleansed) %>%
#  summarise(price = mean(price), avgrating = mean(review_scores_rating), n = n()) %>% #arrange(desc(pr

# cancellation relevant interms of price
amsterdam %>% group_by(cancellation_policy) %>%
  summarise(price = mean(logprice), avgrating = mean(review_scores_rating), n = n()) %>% arrange(desc(pr
```

```
## # A tibble: 5 x 4
##   cancellation_policy         price avgrating     n
##   <chr>                       <dbl>     <dbl> <int>
## 1 super_strict_30              5.66      92.8    26
## 2 super_strict_60              5.51      88.8    25
## 3 strict_14_with_grace_period  4.98      94.9  6621
## 4 moderate                     4.89      95.2  5893
## 5 flexible                     4.82      95.3  2453
```

```
# room_type significant differences (in terms of price)
amsterdam %>% group_by(room_type) %>%
  summarise(price = mean(logprice), avgrating = mean(review_scores_rating), n = n()) %>% arrange(desc(pr
```

```
## # A tibble: 4 x 4
##   room_type       price avgrating     n
##   <chr>           <dbl>     <dbl> <int>
## 1 Entire home/apt  5.01      95.3 12079
## 2 Hotel room       4.94      94.1   291
## 3 Private room     4.52      94.4  2624
## 4 Shared room      4.29      94.5    24
```

```r
# instant_bookable provides no information in terms of price and rating
amsterdam %>% group_by(instant_bookable) %>%
  summarise(price = mean(logprice), avgrating = mean(review_scores_rating), n = n()) %>% arrange(desc(p
```

```
## # A tibble: 2 x 4
##   instant_bookable price avgrating     n
##   <lgl>            <dbl>     <dbl> <int>
## 1 FALSE             4.93      95.6 11342
## 2 TRUE              4.88      93.6  3676
```

### Best Subset Selection, NICE assumptions check

```r
#split the data into training and testing dataset
#airbnb1 take out realprice and X columns
#amsterdam <- read_csv('st443_final_data')
airbnb1 = subset(amsterdam, select = -c(1,15))
traingsize = floor(0.7*nrow(airbnb1))
set.seed(123)
train_ind = sample(seq_len(nrow(airbnb1)),size = traingsize)
train=airbnb1[train_ind,]
test=airbnb1[-train_ind,]

airbnb <- amsterdam
attach(airbnb)
str(airbnb)
```

```
## Classes 'spec_tbl_df', 'tbl_df', 'tbl' and 'data.frame': 15018 obs. of  17 variables:
##  $ X1                    : num  1 2 3 4 5 6 7 8 9 10 ...
##  $ review_scores_rating  : num  98 100 99 95 98 97 80 94 93 97 ...
##  $ host_is_superhost     : num  1 0 1 0 1 0 1 0 0 1 ...
##  $ host_listings_count   : num  1 2 1 1 2 1 1 1 1 2 ...
##  $ host_identity_verified: num  0 0 1 1 0 1 0 1 1 0 ...
##  $ room_type             : chr  "Private room" "Entire home/apt" "Private room" "Entire home/apt" ..
##  $ bathrooms             : num  1.5 1 1 1 1 1 1 1.5 1 2 ...
##  $ bedrooms              : num  1 1 1 3 1 2 1 2 1 3 ...
##  $ minimum_nights        : num  3 14 2 3 3 13 30 3 3 6 ...
##  $ number_of_reviews     : num  269 3 200 32 460 690 61 36 3 190 ...
##  $ cancellation_policy   : chr  "strict_14_with_grace_period" "strict_14_with_grace_period" "strict_
##  $ instant_bookable      : logi  TRUE FALSE TRUE FALSE TRUE FALSE ...
##  $ cleaning_fee          : num  60 40 0 60 35 35 50 50 50 0 ...
##  $ location_3ways        : chr  "far_from_centre" "near_centre" "near_centre" "near_centre" ...
##  $ realprice             : num  59 92.9 155 219 159 ...
##  $ host_since_duration   : num  4007 3585 3462 3397 3331 ...
##  $ logprice              : num  4.08 4.53 5.04 5.39 5.07 ...
##  - attr(*, "spec")=
##   .. cols(
##   ..   X1 = col_double(),
##   ..   review_scores_rating = col_double(),
##   ..   host_is_superhost = col_double(),
##   ..   host_listings_count = col_double(),
##   ..   host_identity_verified = col_double(),
##   ..   room_type = col_character(),
##   ..   bathrooms = col_double(),
```

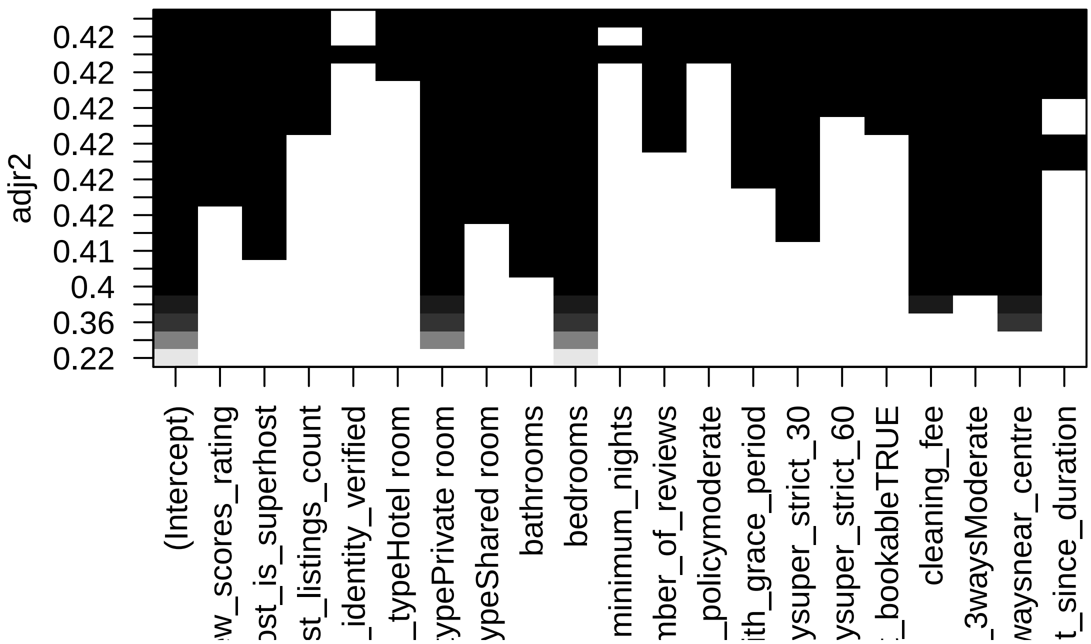
```
##   ..   bedrooms = col_double(),
##   ..   minimum_nights = col_double(),
##   ..   number_of_reviews = col_double(),
##   ..   cancellation_policy = col_character(),
##   ..   instant_bookable = col_logical(),
##   ..   cleaning_fee = col_double(),
##   ..   location_3ways = col_character(),
##   ..   realprice = col_double(),
##   ..   host_since_duration = col_double(),
##   ..   logprice = col_double()
##   .. )
```

```
reg1 = lm(logprice ~., train)
summary(reg1)
```

```
##
## Call:
## lm(formula = logprice ~ ., data = train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2.7700 -0.2298 -0.0119  0.2085  3.2918
##
## Coefficients:
##                                          Estimate Std. Error t value
## (Intercept)                             3.917e+00  5.752e-02  68.099
## review_scores_rating                    3.741e-03  5.791e-04   6.459
## host_is_superhost                       8.191e-02  1.027e-02   7.979
## host_listings_count                    -5.597e-04  1.225e-04  -4.569
## host_identity_verified                 -2.416e-04  7.783e-03  -0.031
## room_typeHotel room                    -6.027e-02  2.744e-02  -2.196
## room_typePrivate room                  -3.397e-01  1.080e-02 -31.446
## room_typeShared room                   -5.944e-01  8.856e-02  -6.712
## bathrooms                               1.034e-01  1.106e-02   9.350
## bedrooms                                1.692e-01  4.893e-03  34.587
## minimum_nights                         -2.281e-04  1.963e-04  -1.162
## number_of_reviews                      -3.147e-04  8.172e-05  -3.851
## cancellation_policymoderate             1.602e-02  1.061e-02   1.510
## cancellation_policystrict_14_with_grace_period  5.426e-02  1.058e-02   5.129
## cancellation_policysuper_strict_30      6.286e-01  8.750e-02   7.185
## cancellation_policysuper_strict_60      3.237e-01  9.069e-02   3.569
## instant_bookableTRUE                    3.460e-02  8.959e-03   3.862
## cleaning_fee                            3.721e-03  1.839e-04  20.236
## location_3waysModerate                  1.609e-01  9.551e-03  16.847
## location_3waysnear_centre               3.456e-01  1.086e-02  31.809
## host_since_duration                    -1.764e-05  5.621e-06  -3.139
##                                        Pr(>|t|)
## (Intercept)                             < 2e-16 ***
## review_scores_rating                   1.10e-10 ***
## host_is_superhost                      1.63e-15 ***
## host_listings_count                    4.96e-06 ***
## host_identity_verified                 0.975238
## room_typeHotel room                    0.028087 *
## room_typePrivate room                   < 2e-16 ***
## room_typeShared room                   2.01e-11 ***
```

```
## bathrooms                                          < 2e-16 ***
## bedrooms                                           < 2e-16 ***
## minimum_nights                                    0.245118
## number_of_reviews                                 0.000118 ***
## cancellation_policymoderate                       0.131051
## cancellation_policystrict_14_with_grace_period 2.97e-07 ***
## cancellation_policysuper_strict_30                7.20e-13 ***
## cancellation_policysuper_strict_60                0.000359 ***
## instant_bookableTRUE                              0.000113 ***
## cleaning_fee                                       < 2e-16 ***
## location_3waysModerate                             < 2e-16 ***
## location_3waysnear_centre                          < 2e-16 ***
## host_since_duration                               0.001699 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.3645 on 10491 degrees of freedom
## Multiple R-squared:  0.4259, Adjusted R-squared:  0.4248
## F-statistic: 389.1 on 20 and 10491 DF,  p-value: < 2.2e-16
```

```r
#Use the best subset selection with respect to Adjusted R^2 and BIC
library(leaps)
reg2=regsubsets(logprice~.,nvmax = 20,data = train)
plot(reg2, scale = "adjr2")
```




```r
summary(reg2)
```

```
## Subset selection object
## Call: regsubsets.formula(logprice ~ ., nvmax = 20, data = train)
## 20 Variables  (and intercept)
##                                                Forced in Forced out
## review_scores_rating                               FALSE      FALSE
## host_is_superhost                                  FALSE      FALSE
## host_listings_count                                FALSE      FALSE
```

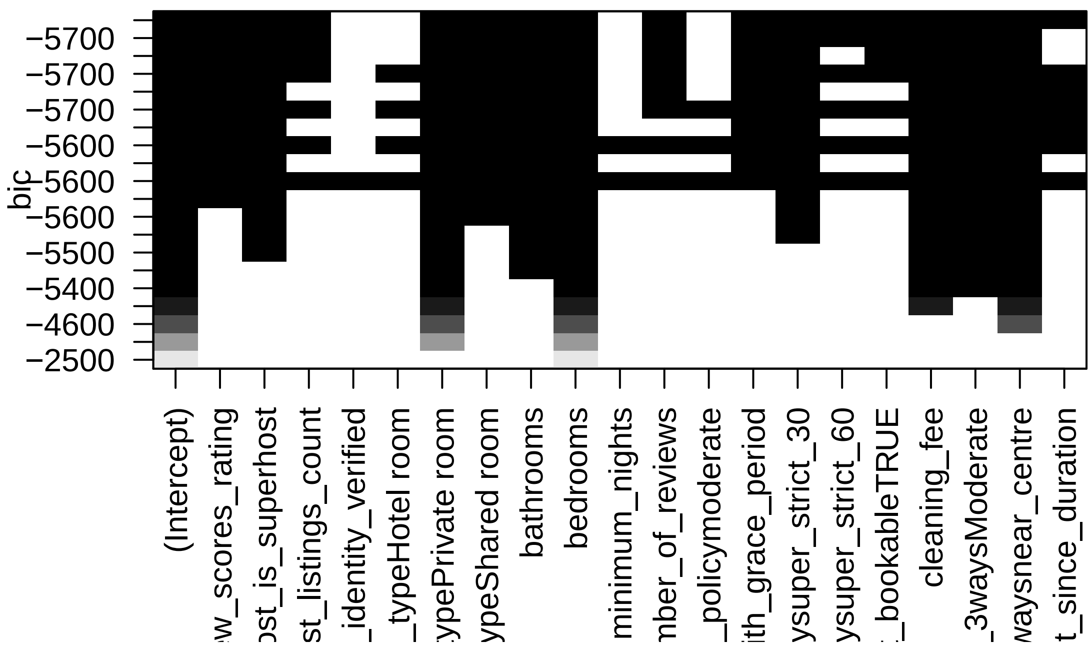
```
## host_identity_verified                            FALSE      FALSE
## room_typeHotel room                               FALSE      FALSE
## room_typePrivate room                             FALSE      FALSE
## room_typeShared room                              FALSE      FALSE
## bathrooms                                         FALSE      FALSE
## bedrooms                                          FALSE      FALSE
## minimum_nights                                    FALSE      FALSE
## number_of_reviews                                 FALSE      FALSE
## cancellation_policymoderate                       FALSE      FALSE
## cancellation_policystrict_14_with_grace_period    FALSE      FALSE
## cancellation_policysuper_strict_30                FALSE      FALSE
## cancellation_policysuper_strict_60                FALSE      FALSE
## instant_bookableTRUE                              FALSE      FALSE
## cleaning_fee                                      FALSE      FALSE
## location_3waysModerate                            FALSE      FALSE
## location_3waysnear_centre                         FALSE      FALSE
## host_since_duration                               FALSE      FALSE
## 1 subsets of each size up to 20
## Selection Algorithm: exhaustive
##           review_scores_rating host_is_superhost host_listings_count
## 1  ( 1 )  " "                  " "               " "
## 2  ( 1 )  " "                  " "               " "
## 3  ( 1 )  " "                  " "               " "
## 4  ( 1 )  " "                  " "               " "
## 5  ( 1 )  " "                  " "               " "
## 6  ( 1 )  " "                  " "               " "
## 7  ( 1 )  " "                  "*"               " "
## 8  ( 1 )  " "                  "*"               " "
## 9  ( 1 )  " "                  "*"               " "
## 10  ( 1 ) "*"                  "*"               " "
## 11  ( 1 ) "*"                  "*"               " "
## 12  ( 1 ) "*"                  "*"               " "
## 13  ( 1 ) "*"                  "*"               " "
## 14  ( 1 ) "*"                  "*"               "*"
## 15  ( 1 ) "*"                  "*"               "*"
## 16  ( 1 ) "*"                  "*"               "*"
## 17  ( 1 ) "*"                  "*"               "*"
## 18  ( 1 ) "*"                  "*"               "*"
## 19  ( 1 ) "*"                  "*"               "*"
## 20  ( 1 ) "*"                  "*"               "*"
##           host_identity_verified room_typeHotel room room_typePrivate room
## 1  ( 1 )  " "                    " "                 " "
## 2  ( 1 )  " "                    " "                 "*"
## 3  ( 1 )  " "                    " "                 "*"
## 4  ( 1 )  " "                    " "                 "*"
## 5  ( 1 )  " "                    " "                 "*"
## 6  ( 1 )  " "                    " "                 "*"
## 7  ( 1 )  " "                    " "                 "*"
## 8  ( 1 )  " "                    " "                 "*"
## 9  ( 1 )  " "                    " "                 "*"
## 10  ( 1 ) " "                    " "                 "*"
## 11  ( 1 ) " "                    " "                 "*"
## 12  ( 1 ) " "                    " "                 "*"
## 13  ( 1 ) " "                    " "                 "*"
```

```
## 14  ( 1 ) " "                    " "                 "*"
## 15  ( 1 ) " "                    " "                 "*"
## 16  ( 1 ) " "                    " "                 "*"
## 17  ( 1 ) " "                    "*"                 "*"
## 18  ( 1 ) " "                    "*"                 "*"
## 19  ( 1 ) " "                    "*"                 "*"
## 20  ( 1 ) "*"                    "*"                 "*"
##           room_typeShared room bathrooms bedrooms minimum_nights
## 1  ( 1 )  " "                  " "       "*"      " "
## 2  ( 1 )  " "                  " "       "*"      " "
## 3  ( 1 )  " "                  " "       "*"      " "
## 4  ( 1 )  " "                  " "       "*"      " "
## 5  ( 1 )  " "                  " "       "*"      " "
## 6  ( 1 )  " "                  "*"       "*"      " "
## 7  ( 1 )  " "                  "*"       "*"      " "
## 8  ( 1 )  " "                  "*"       "*"      " "
## 9  ( 1 )  "*"                  "*"       "*"      " "
## 10  ( 1 ) "*"                  "*"       "*"      " "
## 11  ( 1 ) "*"                  "*"       "*"      " "
## 12  ( 1 ) "*"                  "*"       "*"      " "
## 13  ( 1 ) "*"                  "*"       "*"      " "
## 14  ( 1 ) "*"                  "*"       "*"      " "
## 15  ( 1 ) "*"                  "*"       "*"      " "
## 16  ( 1 ) "*"                  "*"       "*"      " "
## 17  ( 1 ) "*"                  "*"       "*"      " "
## 18  ( 1 ) "*"                  "*"       "*"      " "
## 19  ( 1 ) "*"                  "*"       "*"      "*"
## 20  ( 1 ) "*"                  "*"       "*"      "*"
##           number_of_reviews cancellation_policymoderate
## 1  ( 1 )  " "               " "
## 2  ( 1 )  " "               " "
## 3  ( 1 )  " "               " "
## 4  ( 1 )  " "               " "
## 5  ( 1 )  " "               " "
## 6  ( 1 )  " "               " "
## 7  ( 1 )  " "               " "
## 8  ( 1 )  " "               " "
## 9  ( 1 )  " "               " "
## 10  ( 1 ) " "               " "
## 11  ( 1 ) " "               " "
## 12  ( 1 ) " "               " "
## 13  ( 1 ) "*"               " "
## 14  ( 1 ) "*"               " "
## 15  ( 1 ) "*"               " "
## 16  ( 1 ) "*"               " "
## 17  ( 1 ) "*"               " "
## 18  ( 1 ) "*"               "*"
## 19  ( 1 ) "*"               "*"
## 20  ( 1 ) "*"               "*"
##           cancellation_policystrict_14_with_grace_period
## 1  ( 1 )  " "
## 2  ( 1 )  " "
## 3  ( 1 )  " "
## 4  ( 1 )  " "
```

```
## 5  ( 1 )  " "
## 6  ( 1 )  " "
## 7  ( 1 )  " "
## 8  ( 1 )  " "
## 9  ( 1 )  " "
## 10  ( 1 ) " "
## 11  ( 1 ) "*"
## 12  ( 1 ) "*"
## 13  ( 1 ) "*"
## 14  ( 1 ) "*"
## 15  ( 1 ) "*"
## 16  ( 1 ) "*"
## 17  ( 1 ) "*"
## 18  ( 1 ) "*"
## 19  ( 1 ) "*"
## 20  ( 1 ) "*"
##           cancellation_policysuper_strict_30 cancellation_policysuper_strict_60
## 1  ( 1 )  " "                                " "
## 2  ( 1 )  " "                                " "
## 3  ( 1 )  " "                                " "
## 4  ( 1 )  " "                                " "
## 5  ( 1 )  " "                                " "
## 6  ( 1 )  " "                                " "
## 7  ( 1 )  " "                                " "
## 8  ( 1 )  "*"                                " "
## 9  ( 1 )  "*"                                " "
## 10  ( 1 ) "*"                                " "
## 11  ( 1 ) "*"                                " "
## 12  ( 1 ) "*"                                " "
## 13  ( 1 ) "*"                                " "
## 14  ( 1 ) "*"                                " "
## 15  ( 1 ) "*"                                "*"
## 16  ( 1 ) "*"                                "*"
## 17  ( 1 ) "*"                                "*"
## 18  ( 1 ) "*"                                "*"
## 19  ( 1 ) "*"                                "*"
## 20  ( 1 ) "*"                                "*"
##           instant_bookableTRUE cleaning_fee location_3waysModerate
## 1  ( 1 )  " "                  " "          " "
## 2  ( 1 )  " "                  " "          " "
## 3  ( 1 )  " "                  " "          " "
## 4  ( 1 )  " "                  "*"          " "
## 5  ( 1 )  " "                  "*"          "*"
## 6  ( 1 )  " "                  "*"          "*"
## 7  ( 1 )  " "                  "*"          "*"
## 8  ( 1 )  " "                  "*"          "*"
## 9  ( 1 )  " "                  "*"          "*"
## 10  ( 1 ) " "                  "*"          "*"
## 11  ( 1 ) " "                  "*"          "*"
## 12  ( 1 ) " "                  "*"          "*"
## 13  ( 1 ) " "                  "*"          "*"
## 14  ( 1 ) "*"                  "*"          "*"
## 15  ( 1 ) "*"                  "*"          "*"
## 16  ( 1 ) "*"                  "*"          "*"
```

```
## 17  ( 1 ) "*"                    "*"           "*"
## 18  ( 1 ) "*"                    "*"           "*"
## 19  ( 1 ) "*"                    "*"           "*"
## 20  ( 1 ) "*"                    "*"           "*"
##           location_3waysnear_centre host_since_duration
## 1  ( 1 )  " "                       " "
## 2  ( 1 )  " "                       " "
## 3  ( 1 )  "*"                       " "
## 4  ( 1 )  "*"                       " "
## 5  ( 1 )  "*"                       " "
## 6  ( 1 )  "*"                       " "
## 7  ( 1 )  "*"                       " "
## 8  ( 1 )  "*"                       " "
## 9  ( 1 )  "*"                       " "
## 10  ( 1 ) "*"                       " "
## 11  ( 1 ) "*"                       " "
## 12  ( 1 ) "*"                       "*"
## 13  ( 1 ) "*"                       "*"
## 14  ( 1 ) "*"                       " "
## 15  ( 1 ) "*"                       " "
## 16  ( 1 ) "*"                       "*"
## 17  ( 1 ) "*"                       "*"
## 18  ( 1 ) "*"                       "*"
## 19  ( 1 ) "*"                       "*"
## 20  ( 1 ) "*"                       "*"
```

```
plot(reg2, scale = "bic")
```




```
outbs=summary(reg2)
#Check which variables to remove/remain
which.max(outbs$adjr2)
```

```
## [1] 19
```

```r
which.min(outbs$bic)
```

```
## [1] 16
```

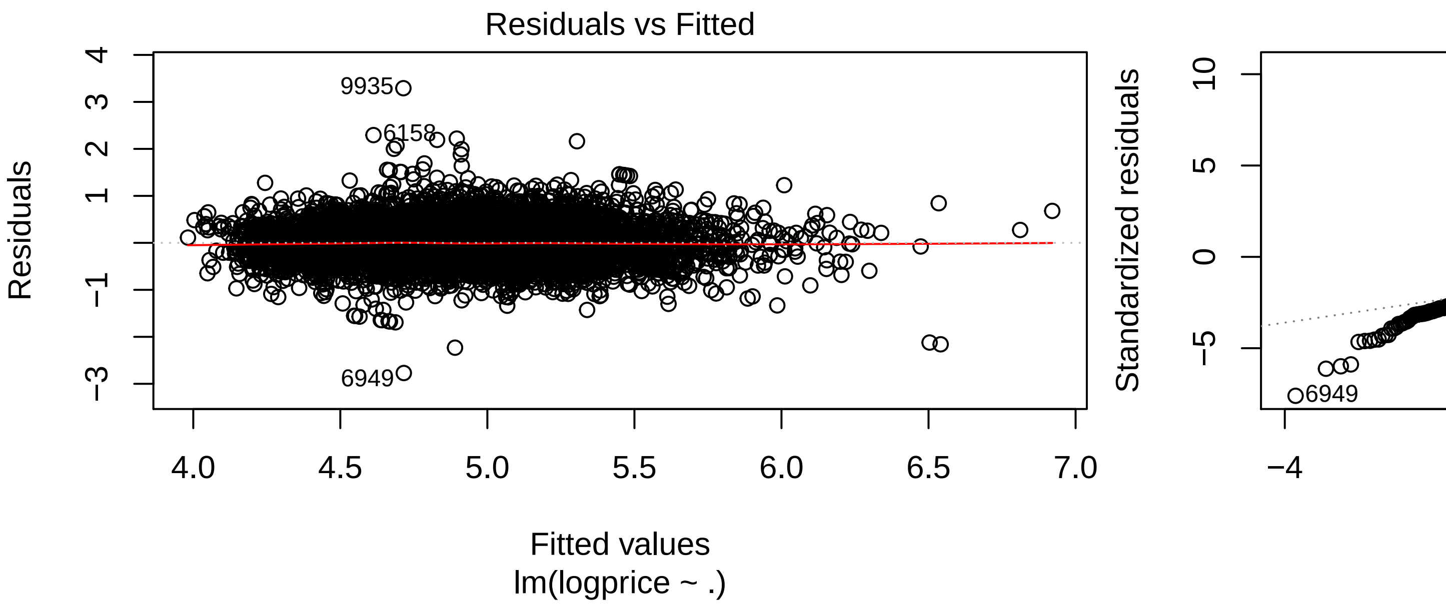
```r
#check multicollinearity
library(car)
```

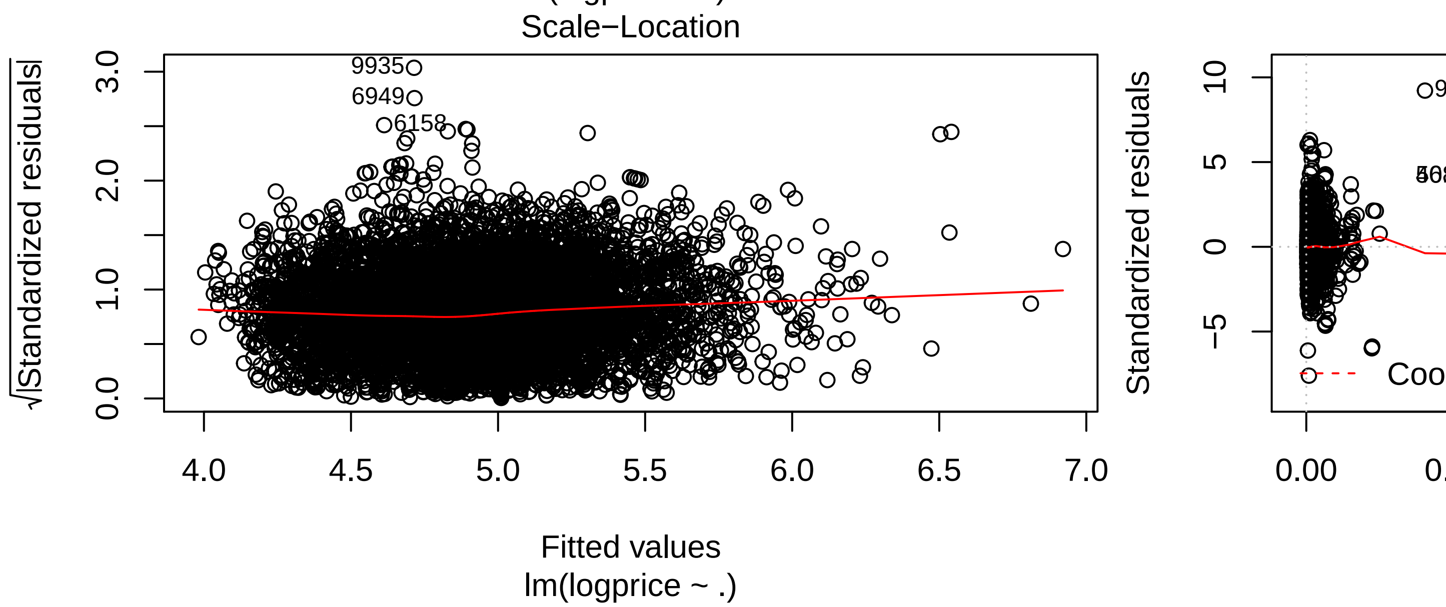
```
## Loading required package: carData

##
## Attaching package: 'car'

## The following object is masked from 'package:dplyr':
##
##     recode

## The following object is masked from 'package:purrr':
##
##     some
```

```r
vif(reg1)
```

```
##                             GVIF Df GVIF^(1/(2*Df))
## review_scores_rating    1.079635  1        1.039055
## host_is_superhost       1.211551  1        1.100705
## host_listings_count     1.108619  1        1.052910
## host_identity_verified 1.168946  1        1.081178
## room_type               1.445793  3        1.063370
## bathrooms               1.280840  1        1.131742
## bedrooms                1.432958  1        1.197062
## minimum_nights          1.002208  1        1.001104
## number_of_reviews       1.363517  1        1.167697
## cancellation_policy     1.145920  4        1.017172
## instant_bookable        1.173188  1        1.083138
## cleaning_fee            1.340805  1        1.157932
## location_3ways          1.078301  2        1.019025
## host_since_duration     1.210094  1        1.100043
```

```r
airbnb2 = subset(train, select = -c(host_identity_verified))
reg3 = lm(logprice~., airbnb2)
##Check NICE(Normality, Independence, constant variance and expectation of residuals is 0) property
plot(reg3)
```




```r
##Check if multicollinearity exists
vif(reg3)
```

```
##                           GVIF Df GVIF^(1/(2*Df))
## review_scores_rating 1.076580  1        1.037584
## host_is_superhost    1.211465  1        1.100666
## host_listings_count  1.101879  1        1.049704
## room_type            1.444603  3        1.063224
## bathrooms            1.280564  1        1.131620
## bedrooms             1.432548  1        1.196891
## minimum_nights       1.002075  1        1.001037
## number_of_reviews    1.357228  1        1.165001
## cancellation_policy  1.144420  4        1.017005
## instant_bookable     1.165138  1        1.079416
## cleaning_fee         1.339441  1        1.157342
## location_3ways       1.078102  2        1.018978
```

```
## host_since_duration  1.101082  1        1.049324
```

```r
summary(reg3)
```

```
##
## Call:
## lm(formula = logprice ~ ., data = airbnb2)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2.7701 -0.2298 -0.0118  0.2084  3.2919
##
## Coefficients:
##                                              Estimate Std. Error t value
## (Intercept)                                 3.917e+00  5.746e-02  68.180
## review_scores_rating                        3.740e-03  5.783e-04   6.467
## host_is_superhost                           8.191e-02  1.026e-02   7.980
## host_listings_count                        -5.594e-04  1.221e-04  -4.581
## room_typeHotel room                        -6.026e-02  2.744e-02  -2.196
## room_typePrivate room                      -3.397e-01  1.080e-02 -31.452
## room_typeShared room                       -5.945e-01  8.855e-02  -6.714
## bathrooms                                   1.034e-01  1.106e-02   9.351
## bedrooms                                    1.692e-01  4.892e-03  34.593
## minimum_nights                             -2.282e-04  1.962e-04  -1.163
## number_of_reviews                          -3.149e-04  8.153e-05  -3.862
## cancellation_policymoderate                 1.602e-02  1.061e-02   1.510
## cancellation_policystrict_14_with_grace_period  5.425e-02  1.058e-02   5.130
## cancellation_policysuper_strict_30          6.286e-01  8.748e-02   7.185
## cancellation_policysuper_strict_60          3.238e-01  9.066e-02   3.571
## instant_bookableTRUE                        3.462e-02  8.927e-03   3.878
## cleaning_fee                                3.721e-03  1.838e-04  20.248
## location_3waysModerate                      1.609e-01  9.550e-03  16.848
## location_3waysnear_centre                   3.456e-01  1.086e-02  31.810
## host_since_duration                        -1.770e-05  5.361e-06  -3.301
##                                            Pr(>|t|)
## (Intercept)                                 < 2e-16 ***
## review_scores_rating                       1.04e-10 ***
## host_is_superhost                          1.62e-15 ***
## host_listings_count                        4.69e-06 ***
## room_typeHotel room                        0.028095 *
## room_typePrivate room                       < 2e-16 ***
## room_typeShared room                       2.00e-11 ***
## bathrooms                                   < 2e-16 ***
## bedrooms                                    < 2e-16 ***
## minimum_nights                             0.244919
## number_of_reviews                          0.000113 ***
## cancellation_policymoderate                0.131084
## cancellation_policystrict_14_with_grace_period 2.96e-07 ***
## cancellation_policysuper_strict_30         7.17e-13 ***
## cancellation_policysuper_strict_60         0.000357 ***
## instant_bookableTRUE                       0.000106 ***
## cleaning_fee                                < 2e-16 ***
## location_3waysModerate                      < 2e-16 ***
## location_3waysnear_centre                   < 2e-16 ***
## host_since_duration                        0.000967 ***
```

```
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.3644 on 10492 degrees of freedom
## Multiple R-squared:  0.4259, Adjusted R-squared:  0.4248
## F-statistic: 409.6 on 19 and 10492 DF,  p-value: < 2.2e-16
```

```
#calculate MSE
#predictedvalues = predict(reg3, newdata = test)
#plot(predictedvalues, test$logprice)
#MSE1 = mean((predictedvalues-test$logprice)^2)
#the mse is around 0.143
##other variable selection method(not going to present in the report)
#step1 = stepAIC(reg1, direction = "both")
#summary(step1)
```

### Shrinkage

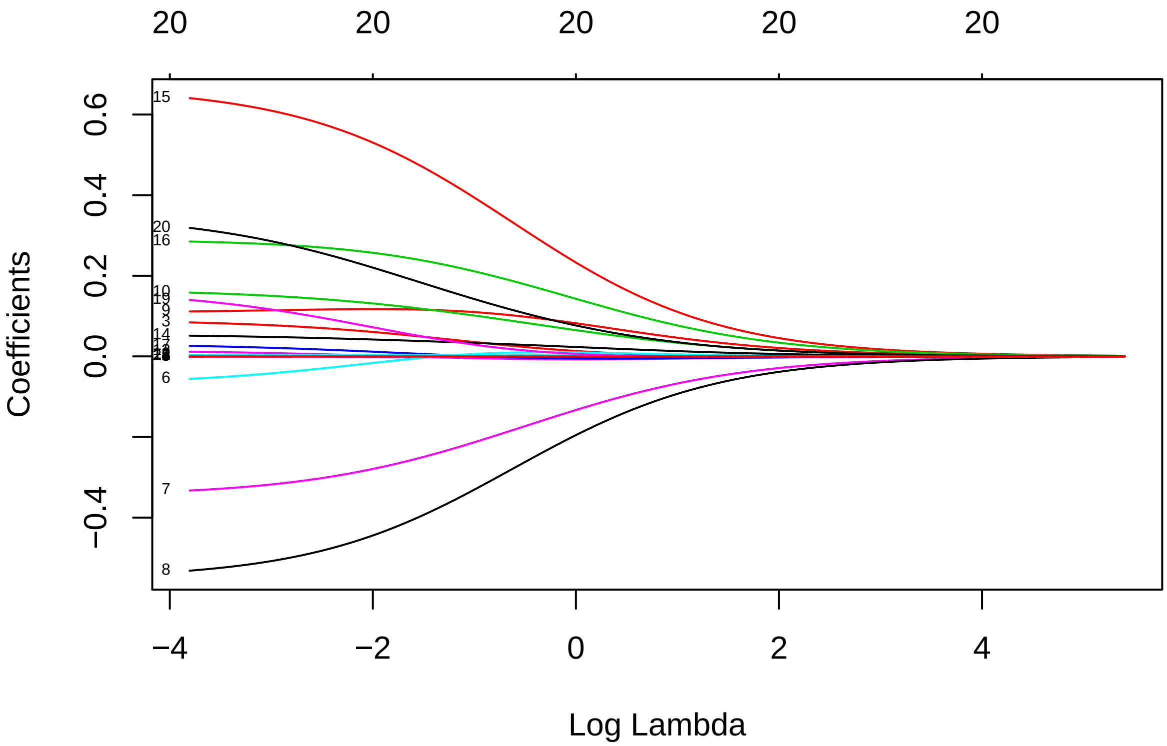
## Ridge

```
## Warning: Missing column names filled in: 'X1' [1]
```

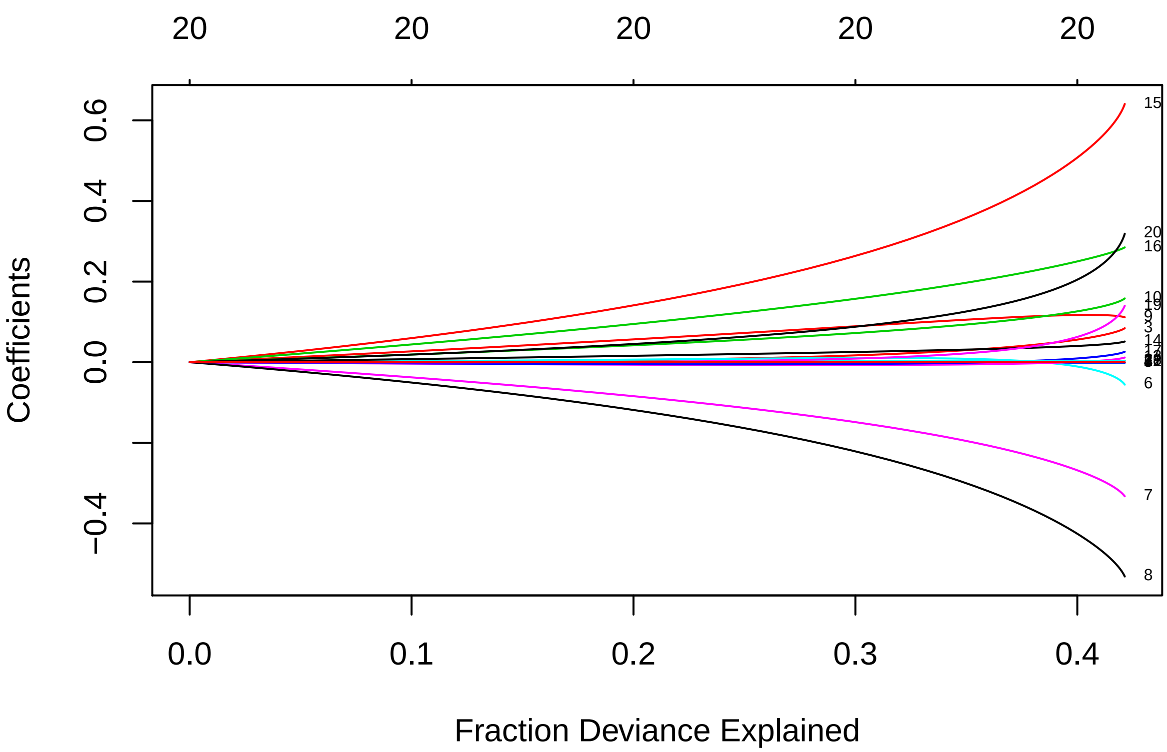
```
## Parsed with column specification:
## cols(
##   X1 = col_double(),
##   review_scores_rating = col_double(),
##   host_is_superhost = col_double(),
##   host_listings_count = col_double(),
##   host_identity_verified = col_double(),
##   room_type = col_character(),
##   bathrooms = col_double(),
##   bedrooms = col_double(),
##   minimum_nights = col_double(),
##   number_of_reviews = col_double(),
##   cancellation_policy = col_character(),
##   instant_bookable = col_logical(),
##   cleaning_fee = col_double(),
##   location_3ways = col_character(),
##   realprice = col_double(),
##   host_since_duration = col_double(),
##   logprice = col_double()
## )
```

```
suppressMessages(library(glmnet))

#amsterdam <- read_csv('st443_final_data')
#amsterdam <- amsterdam[,-c(1,15)]

# glmnet does not use formula language
x <- model.matrix(logprice ~ ., data = amsterdam)
y <- amsterdam$logprice

fit.ridge <-glmnet(x, y, alpha=0)
```

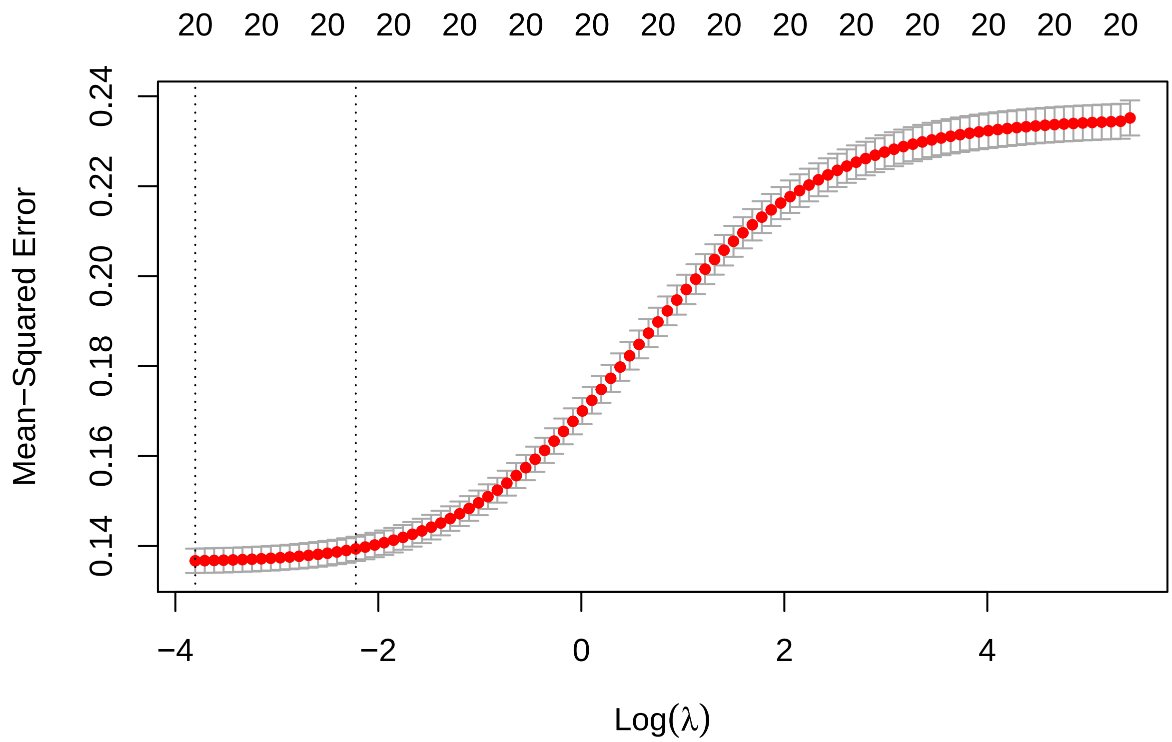
```r
# 8, 7, 15, 20, 16 most important vars
plot(fit.ridge, xvar="lambda", label= TRUE)
```




```r
plot(fit.ridge, xvar="dev", label= TRUE)
```



```r
cv.ridge <-cv.glmnet(x, y, alpha=0)

## Plot of CV mse vs log (lambda), small lambda is best
plot(cv.ridge)
```




```r
## Coeffecent vector corresponding to the mse which is within
# one standard error of the lowest mse using the best lambda.
coef(cv.ridge)
```

```
## 22 x 1 sparse Matrix of class "dgCMatrix"
##                                                          1
## (Intercept)                                   4.0718649364
## (Intercept)                                   .
## review_scores_rating                          0.0032651038
## host_is_superhost                             0.0652320883
## host_listings_count                          -0.0003663436
## host_identity_verified                       -0.0021231936
## room_typeHotel room                          -0.0222974072
## room_typePrivate room                        -0.2901882819
## room_typeShared room                         -0.4627803907
## bathrooms                                     0.1170837789
## bedrooms                                      0.1364474682
## minimum_nights                               -0.0001937531
## number_of_reviews                            -0.0003570440
## cancellation_policymoderate                   0.0035154889
## cancellation_policystrict_14_with_grace_period 0.0434824030
## cancellation_policysuper_strict_30            0.5529793443
## cancellation_policysuper_strict_60            0.2635415827
## instant_bookableTRUE                          0.0141006790
## cleaning_fee                                  0.0035043919
## location_3waysModerate                        0.0828561550
```

```
## location_3waysnear_centre                          0.2367923240
## host_since_duration                               -0.0000104894
```

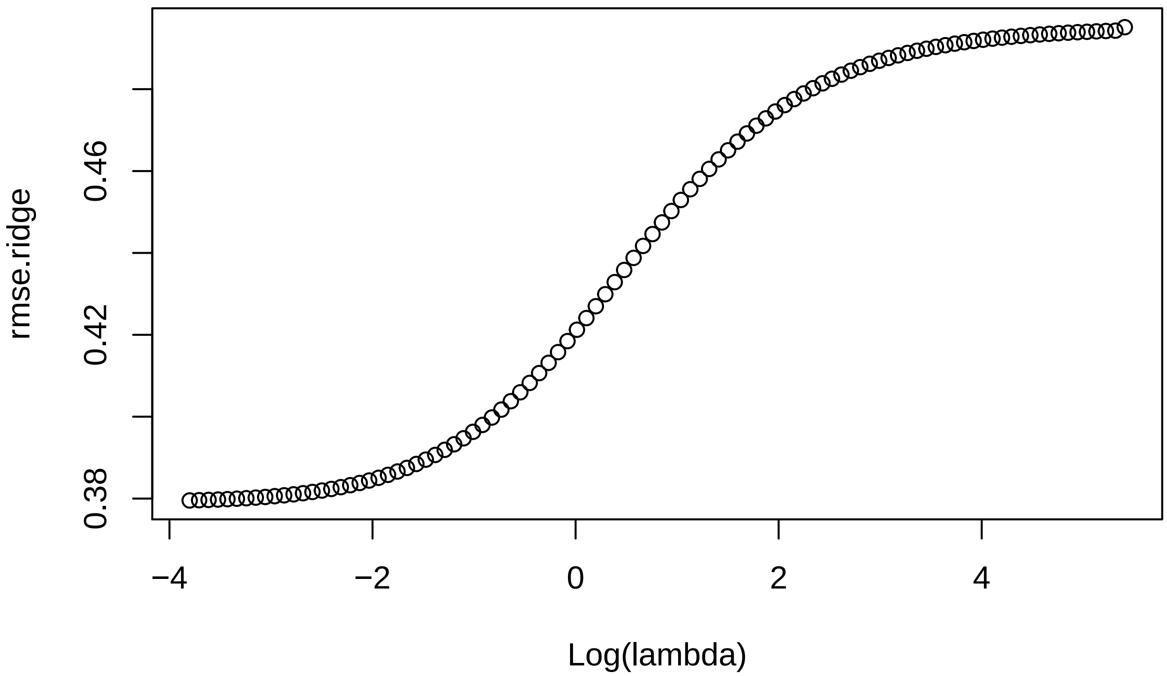
```r
## Coefficient vector corresponding to the lowest mse using the best lambda
coef(glmnet(x,y,alpha=0, lambda=cv.ridge$lambda.min))
```

```
## 22 x 1 sparse Matrix of class "dgCMatrix"
##                                                           s0
## (Intercept)                                     3.967646e+00
## (Intercept)                                     .
## review_scores_rating                            3.592352e-03
## host_is_superhost                               8.430644e-02
## host_listings_count                            -5.073438e-04
## host_identity_verified                         -1.472142e-03
## room_typeHotel room                            -5.567573e-02
## room_typePrivate room                          -3.328267e-01
## room_typeShared room                           -5.317271e-01
## bathrooms                                       1.115002e-01
## bedrooms                                        1.582633e-01
## minimum_nights                                 -2.600664e-04
## number_of_reviews                              -3.480520e-04
## cancellation_policymoderate                     1.165951e-02
## cancellation_policystrict_14_with_grace_period  5.136956e-02
## cancellation_policysuper_strict_30              6.407773e-01
## cancellation_policysuper_strict_60              2.849320e-01
## instant_bookableTRUE                            2.598437e-02
## cleaning_fee                                    3.518221e-03
## location_3waysModerate                          1.400964e-01
## location_3waysnear_centre                       3.188280e-01
## host_since_duration                            -1.652107e-05
```

```r
# finding MSE
traingsize = floor(0.7*nrow(amsterdam))
set.seed(123)
train = sample(seq_len(nrow(amsterdam)),size = traingsize)

ridge.train <-glmnet(x[train,], y[train], alpha = 0)
pred.test.ridge <-predict(ridge.train, x[-train,])
dim(pred.test.ridge)
```

```
## [1] 4506  100
```

```r
rmse.ridge <-sqrt(apply((y[-train]-pred.test.ridge)^2,2,mean))
plot(log(ridge.train$lambda), rmse.ridge, type="b", xlab="Log(lambda)")
```




```
lambda.best.ridge <- ridge.train$lambda[order(rmse.ridge)[1]]
lambda.best.ridge
```

```
## [1] 0.02234597
```

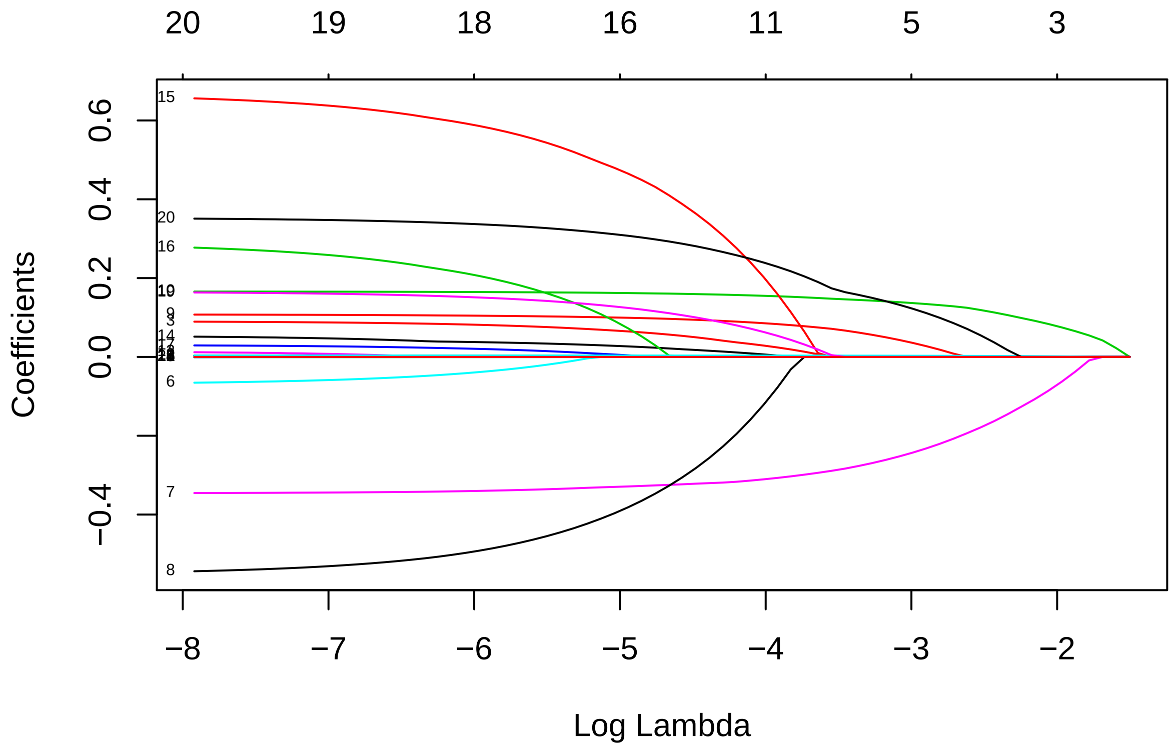
```
mseRidge <- min(rmse.ridge)
mseRidge
```

```
## [1] 0.3795437
```

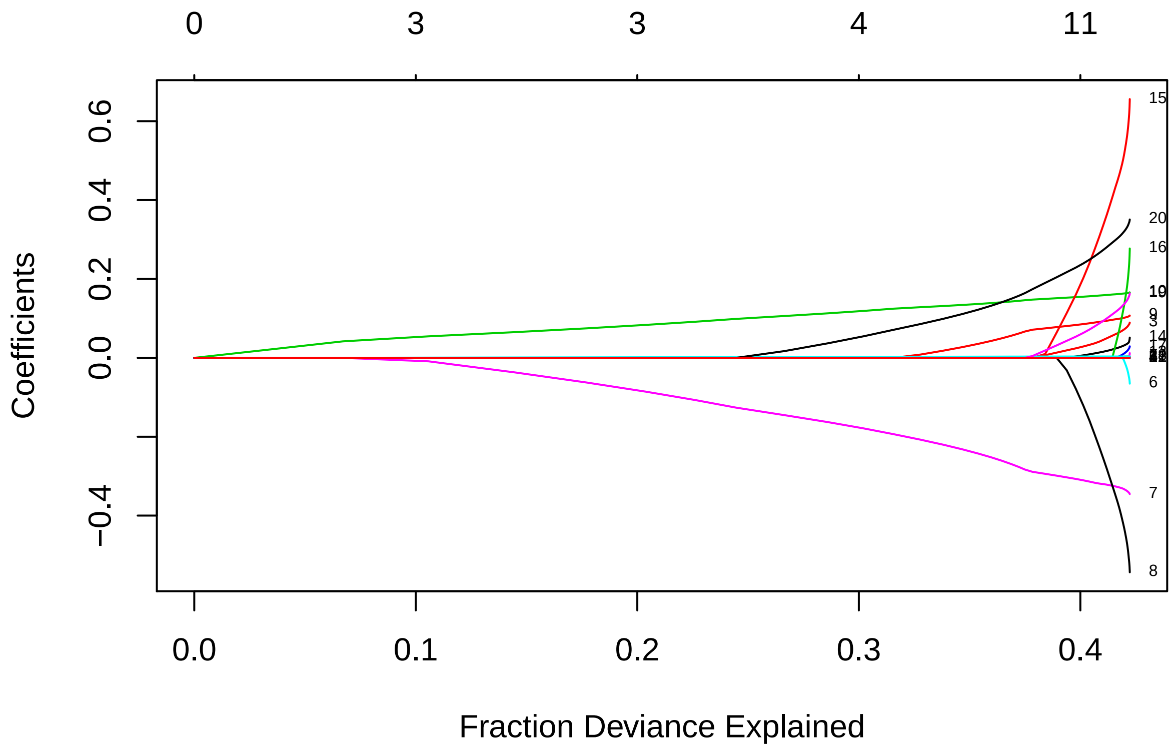
```
#the mse for rigde is 0.144
```

## Lasso

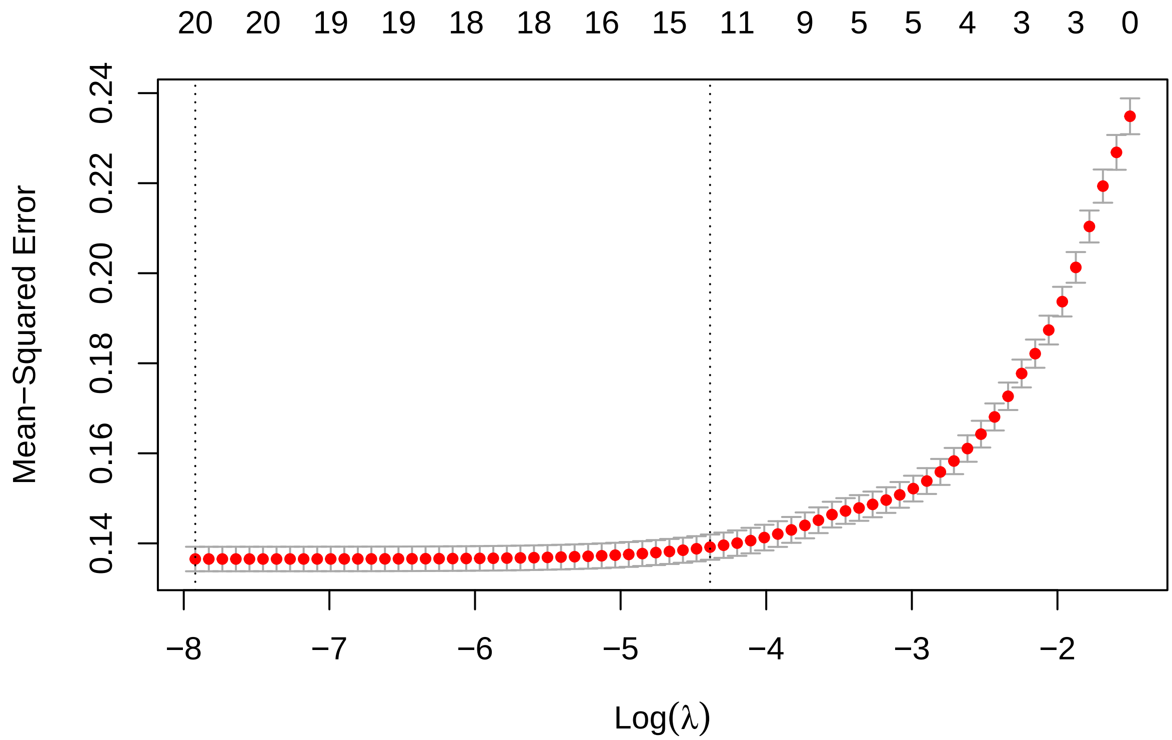
```
fit.lasso <- glmnet(x,y)
plot(fit.lasso, xvar="lambda", label= TRUE)
```




```
plot(fit.lasso, xvar="dev", label= TRUE)
```




```
cv.lasso <-cv.glmnet(x, y)
# Again, 8, 15, 7, 20.

plot(cv.lasso)
```




```r
# Use very small lambda, again
## coefficent vector corresponding to the mse which is within
# one standard error of the lowest mse using the best lambda.
coef(cv.lasso)
```

```
## 22 x 1 sparse Matrix of class "dgCMatrix"
##                                                         1
## (Intercept)                                  4.165290e+00
## (Intercept)                                  .
## review_scores_rating                         2.067228e-03
## host_is_superhost                            4.570986e-02
## host_listings_count                          .
## host_identity_verified                       .
## room_typeHotel room                          .
## room_typePrivate room                       -3.202302e-01
## room_typeShared room                        -2.559487e-01
## bathrooms                                    9.280449e-02
## bedrooms                                     1.586917e-01
## minimum_nights                               .
## number_of_reviews                           -5.087404e-05
## cancellation_policymoderate                  .
## cancellation_policystrict_14_with_grace_period 1.563968e-02
## cancellation_policysuper_strict_30           3.370446e-01
## cancellation_policysuper_strict_60           .
## instant_bookableTRUE                         .
## cleaning_fee                                 3.411724e-03
## location_3waysModerate                       9.394693e-02
## location_3waysnear_centre                    2.738160e-01
## host_since_duration                          .
```

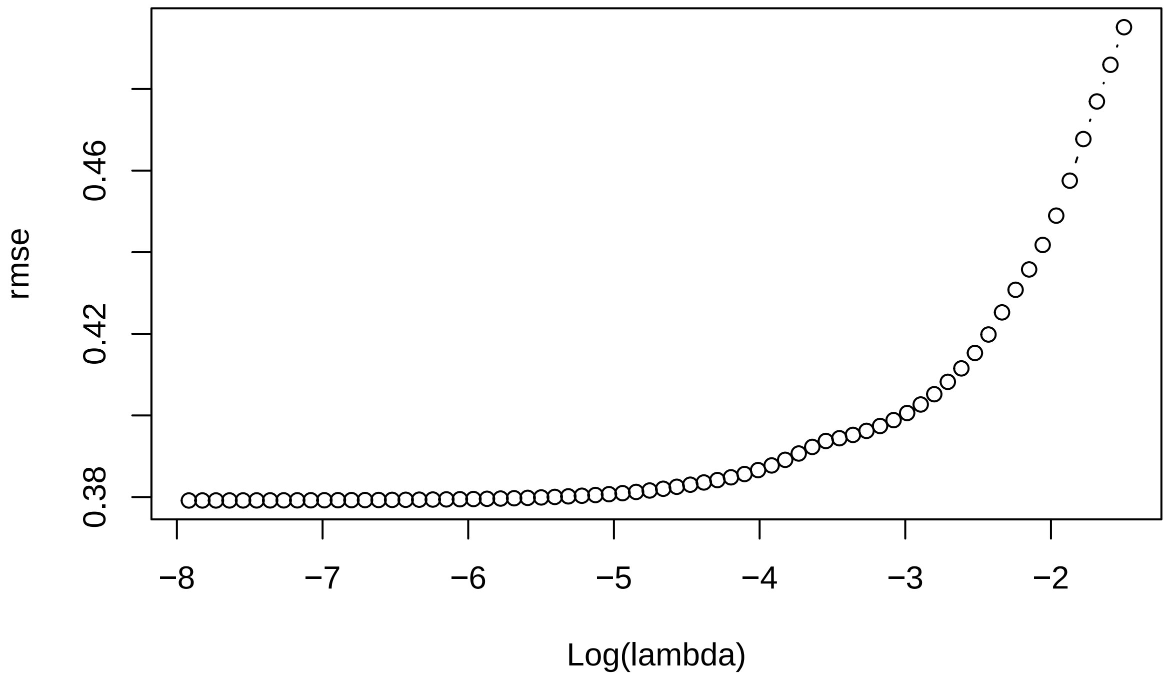
```r
## coefficient vector corresponding to the lowest mse using the best lambda
coef(glmnet(x,y, lambda=cv.lasso$lambda.min))
```

```
## 22 x 1 sparse Matrix of class "dgCMatrix"
##                                                        s0
## (Intercept)                                  3.940950e+00
## (Intercept)                                  .
## review_scores_rating                         3.632109e-03
## host_is_superhost                            8.941803e-02
## host_listings_count                         -5.363794e-04
## host_identity_verified                      -5.333925e-04
## room_typeHotel room                         -6.555163e-02
## room_typePrivate room                       -3.453835e-01
## room_typeShared room                        -5.437931e-01
## bathrooms                                    1.074298e-01
## bedrooms                                     1.658676e-01
## minimum_nights                              -2.592541e-04
## number_of_reviews                           -3.362666e-04
## cancellation_policymoderate                  1.237059e-02
## cancellation_policystrict_14_with_grace_period  5.173575e-02
## cancellation_policysuper_strict_30           6.563447e-01
## cancellation_policysuper_strict_60           2.775788e-01
## instant_bookableTRUE                         2.921495e-02
## cleaning_fee                                 3.470366e-03
## location_3waysModerate                       1.636307e-01
## location_3waysnear_centre                    3.509282e-01
## host_since_duration                         -1.841013e-05
```

```r
## test MSE
lasso.train <-glmnet(x[train,], y[train])
pred.test <-predict(lasso.train, x[-train,])
dim(pred.test)
```

```
## [1] 4506   70
```

```r
rmse <-sqrt(apply((y[-train]-pred.test)^2,2,mean))
plot(log(lasso.train$lambda), rmse, type="b", xlab="Log(lambda)")
```




```r
lambda.best <-lasso.train$lambda[order(rmse)[1]]
lambda.best
```

```
## [1] 0.0003641836
```

```r
mseLasso <- min(rmse)
mseLasso
```

```
## [1] 0.3791684
```

```r
#the mse is 0.143
```

### Trees

Codes: Generate training and testing set

```r
set.seed(123)
trainingsize <- floor(0.7 * nrow(amsterdam))
trainindex <- sample(seq_len(nrow(amsterdam)), size = trainingsize)
levels(amsterdam$room_type)
```

```
## NULL
```

```r
train_df <- amsterdam[trainindex,]
test_df <- amsterdam[-trainindex,]
```

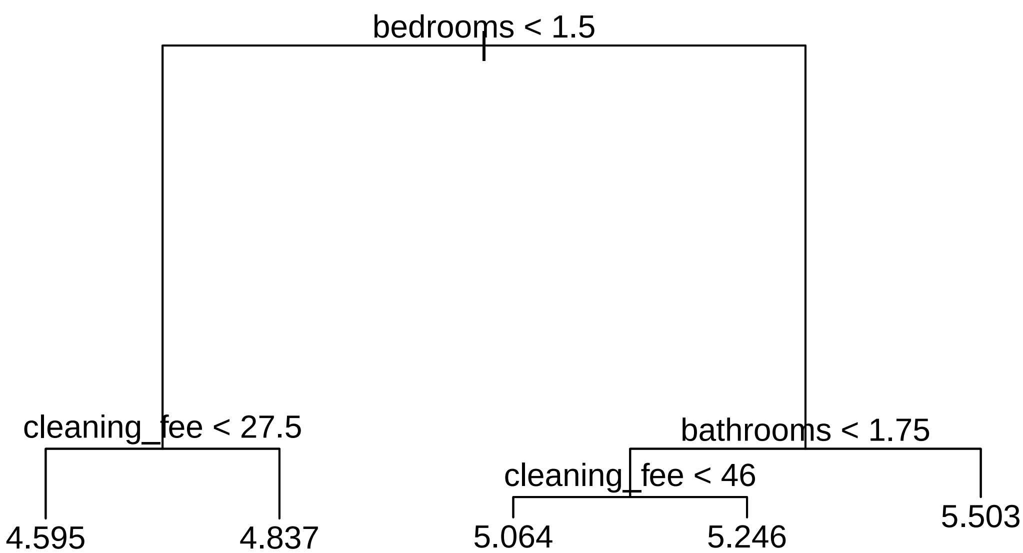
Codes: Decision tree - base model and plots 7 terminal nodes, bedrooms/roomtype+bathroom/location in order of tree hierachy

```
## Registered S3 method overwritten by 'tree':
##   method     from
##   print.tree cli
```

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

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

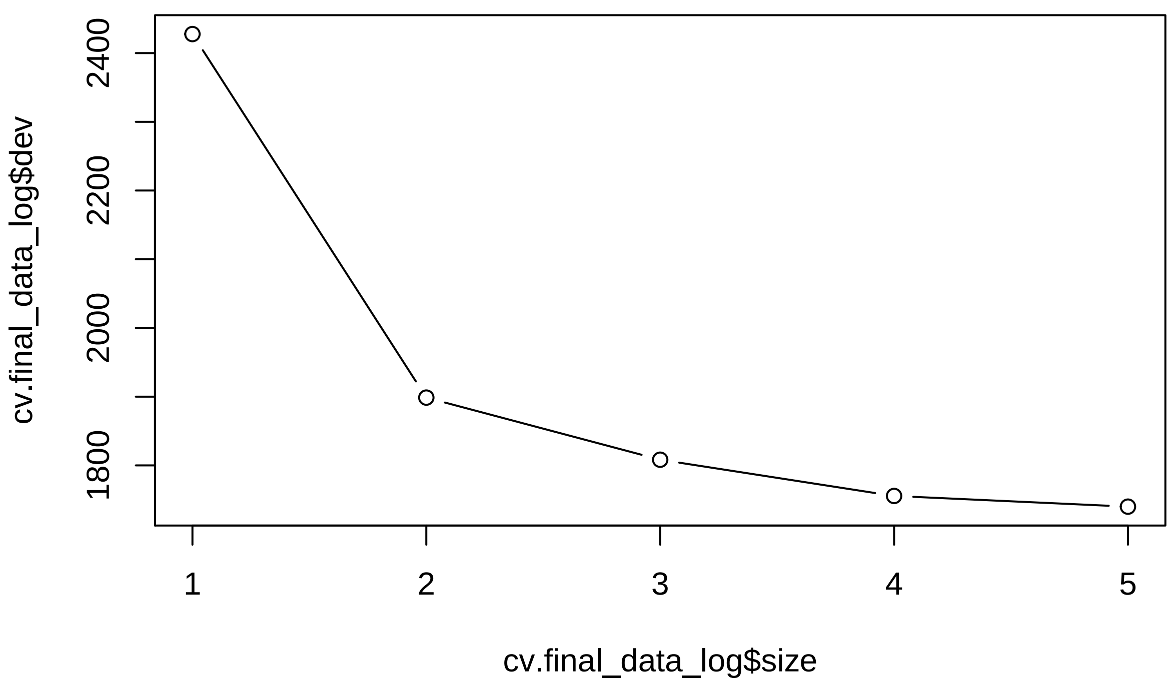
```
##
## Attaching package: 'randomForest'

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

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

## Warning in tree(logprice ~ review_scores_rating + host_is_superhost +
## host_listings_count + : NAs introduced by coercion

##
## Regression tree:
## tree(formula = logprice ~ review_scores_rating + host_is_superhost +
##     host_listings_count + host_identity_verified + room_type +
##     bathrooms + bedrooms + minimum_nights + number_of_reviews +
##     cancellation_policy + instant_bookable + host_since_duration +
##     location_3ways + cleaning_fee, data = train_df)
## Variables actually used in tree construction:
## [1] "bedrooms"     "cleaning_fee" "bathrooms"
## Number of terminal nodes:  5
## Residual mean deviance:  0.1634 = 1717 / 10510
## Distribution of residuals:
##      Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
## -2.649000 -0.242000 -0.009072  0.000000  0.233600  3.412000
```




Codes: Cross-validation on base decision tree Choose 3 terminal nodes as the decrease in deviation from 3 nodes onwards is minimal.

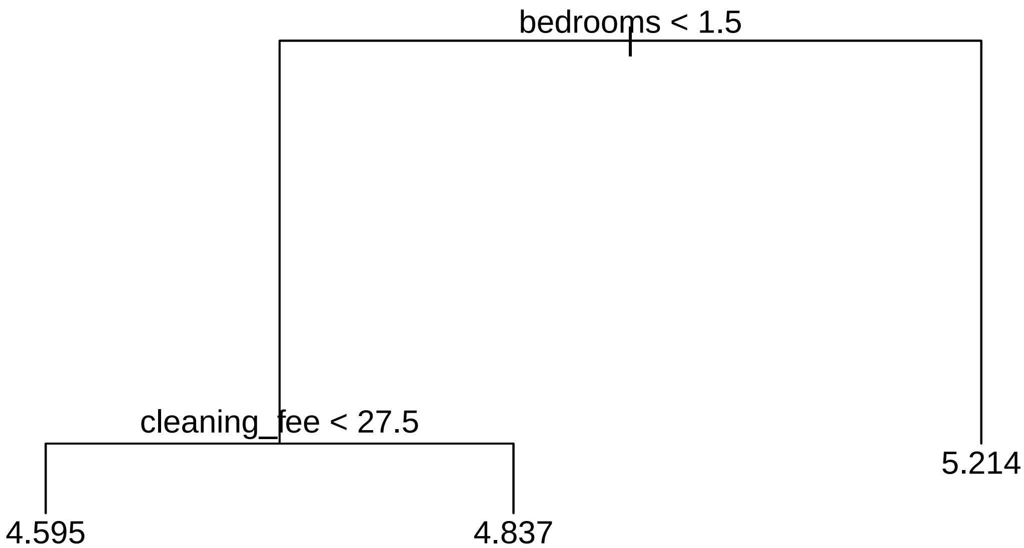
```
## Warning in tree(model = m[rand != i, , drop = FALSE]): NAs introduced by
## coercion

## Warning in pred1.tree(tree, tree.matrix(nd)): NAs introduced by coercion

## Warning in tree(model = m[rand != i, , drop = FALSE]): NAs introduced by
## coercion

## Warning in pred1.tree(tree, tree.matrix(nd)): NAs introduced by coercion
```

```
## Warning in tree(model = m[rand != i, , drop = FALSE]): NAs introduced by
## coercion

## Warning in pred1.tree(tree, tree.matrix(nd)): NAs introduced by coercion

## Warning in tree(model = m[rand != i, , drop = FALSE]): NAs introduced by
## coercion

## Warning in pred1.tree(tree, tree.matrix(nd)): NAs introduced by coercion

## Warning in tree(model = m[rand != i, , drop = FALSE]): NAs introduced by
## coercion

## Warning in pred1.tree(tree, tree.matrix(nd)): NAs introduced by coercion

## Warning in tree(model = m[rand != i, , drop = FALSE]): NAs introduced by
## coercion

## Warning in pred1.tree(tree, tree.matrix(nd)): NAs introduced by coercion

## Warning in tree(model = m[rand != i, , drop = FALSE]): NAs introduced by
## coercion

## Warning in pred1.tree(tree, tree.matrix(nd)): NAs introduced by coercion

## Warning in tree(model = m[rand != i, , drop = FALSE]): NAs introduced by
## coercion

## Warning in pred1.tree(tree, tree.matrix(nd)): NAs introduced by coercion

## Warning in tree(model = m[rand != i, , drop = FALSE]): NAs introduced by
## coercion

## Warning in pred1.tree(tree, tree.matrix(nd)): NAs introduced by coercion

## Warning in tree(model = m[rand != i, , drop = FALSE]): NAs introduced by
## coercion

## Warning in pred1.tree(tree, tree.matrix(nd)): NAs introduced by coercion
```



Codes: Plot of Prune tree

```
##
## Regression tree:
## snip.tree(tree = tree.final_data_log, nodes = 3L)
## Variables actually used in tree construction:
## [1] "bedrooms"     "cleaning_fee"
## Number of terminal nodes:  3
## Residual mean deviance:  0.1719 = 1807 / 10510
## Distribution of residuals:
##      Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
## -2.649000 -0.242300 -0.009072  0.000000  0.234500  3.412000
```




Codes: Generate predicted value of log price on test_df and calculate MSE (0.1748343)

```
yhat_log <- predict(prune.tree_final_data_log, newdata = test_df)
```

```
## Warning in pred1.tree(object, tree.matrix(newdata)): NAs introduced by coercion
```

```
tree_final_data_log.test <- test_df[,"logprice"]

## Compute the test MSE
mean((yhat_log - tree_final_data_log.test)^2)
```

```
## Warning in mean.default((yhat_log - tree_final_data_log.test)^2): argument is
## not numeric or logical: returning NA
```

```
## [1] NA
```

Codes: Use of bagging, m=14 Compare MSE of bagged tree (0.1318838), lower than base decision tree of 0.1748343 Var Imp Plot shows that bedrooms, room type, cleaning fee, locations (in priorities) are the main factors [Note: bedroom is >100%]

```
bag.final_data_log <- randomForest(logprice ~ review_scores_rating + host_is_superhost +
                                 host_listings_count + host_identity_verified +
                                 room_type + bathrooms + bedrooms +
                                 minimum_nights + number_of_reviews + cancellation_policy +
                                 instant_bookable + host_since_duration + location_3ways +
                                 cleaning_fee,  data = train_df, mtry=14, importance=TRUE)
bag.final_data_log
```

```
yhat_log.bag <- predict(bag.final_data_log, newdata = test_df)

## Compute the test MSE
mean((yhat_log.bag - tree_final_data_log.test)^2)
# 0.1318838 MSE
importance(bag.final_data_log)
varImpPlot(bag.final_data_log)
```

Codes: Random forest with n.tree = 5000. With 14 features, 3 different random forest models with varying "m" are run ==> m=sqrt(14), m=7 (14/2), and m = 4 (14/3)

m = sqrt(14): MSE = 0.1290447 m = 7 (14/2): MSE = 0.1305521 m = 4 (14/3): MSE = 0.1290447

```
set.seed(123)
forest.final_data_m1 <- randomForest(logprice ~ review_scores_rating + host_is_superhost +
                                     host_listings_count + host_identity_verified +
                                     room_type + bathrooms + bedrooms +
                                     minimum_nights + number_of_reviews + cancellation_policy +
                                     instant_bookable + host_since_duration + location_3ways +
                                     cleaning_fee,  data = train_df, mtry=sqrt(14), importance=TRUE,
                                     n.tree = 5000)
forest.final_data_m1

## Predicted values on the testing data
yhat.forest_m1 <-predict(forest.final_data_m1, newdata=test_df)

## Compute the test MSE
mean((yhat.forest_m1 - tree_final_data_log.test)^2)
# MSE of 0.1290447
```

```
set.seed(123)
forest.final_data_m2 <- randomForest(logprice ~ review_scores_rating + host_is_superhost +
                                     host_listings_count + host_identity_verified +
                                     room_type + bathrooms + bedrooms +
                                     minimum_nights + number_of_reviews + cancellation_policy +
                                     instant_bookable + host_since_duration + location_3ways +
                                     cleaning_fee,  data = train_df, mtry=7, importance=TRUE,
                                     n.tree = 5000)
forest.final_data_m2

## Predicted values on the testing data
yhat.forest_m2 <-predict(forest.final_data_m2, newdata=test_df)

## Compute the test MSE
mean((yhat.forest_m2 - tree_final_data_log.test)^2)
# MSE of 0.1305521
```

```
set.seed(123)
forest.final_data_m3 <- randomForest(logprice ~ review_scores_rating + host_is_superhost +
                                     host_listings_count + host_identity_verified +
                                     room_type + bathrooms + bedrooms +
                                     minimum_nights + number_of_reviews + cancellation_policy +
                                     instant_bookable + host_since_duration + location_3ways +
                                     cleaning_fee,  data = train_df, mtry=4, importance=TRUE,
                                     n.tree = 5000)
```

```r
forest.final_data_m3

## Predicted values on the testing data
yhat.forest_m3 <-predict(forest.final_data_m3, newdata=test_df)

## Compute the test MSE
mean((yhat.forest_m3 - tree_final_data_log.test)^2)
# MSE of 0.1290447
```

Codes: Boosting with n.tree = 5000. 2 different boosting models with varying depth -> depth=4 and depth =6

Boosting depth = 4: MSE: 0.1383925 ==> relative influence of host_since_duration followed by bedrooms are the highest Boosting depth = 6: MSE: 0.1435316 ==> relative influence of host_since_duration followed by bedrooms remains the highest

```r
library(gbm)
set.seed (123)
train_df$instant_bookable <- factor(train_df$instant_bookable)
boost.log1 <- gbm( logprice ~ review_scores_rating + host_is_superhost +
                      host_listings_count + host_identity_verified +
                      room_type + bathrooms + bedrooms +
                      minimum_nights + number_of_reviews + cancellation_policy +
                      instant_bookable + host_since_duration + location_3ways +
                      cleaning_fee, data = train_df, distribution = "gaussian",
                    n.trees = 5000, interaction.depth = 4)
summary(boost.log1)

## Predicted values on the testing data
yhat.boost1 <- predict(boost.log1, newdata = test_df, n.trees = 5000)

## Compute the test MSE
mean((yhat.boost1 - tree_final_data_log.test) ^ 2)
#MSE of 0.1383925
```

```r
set.seed (123)
train_df$instant_bookable <- factor(train_df$instant_bookable)
boost.log2 <- gbm( logprice ~ review_scores_rating + host_is_superhost +
                    host_listings_count + host_identity_verified +
                    room_type + bathrooms + bedrooms +
                    minimum_nights + number_of_reviews + cancellation_policy +
                    instant_bookable + host_since_duration + location_3ways +
                    cleaning_fee, data = train_df, distribution = "gaussian",
                  n.trees = 5000, interaction.depth = 6)
summary(boost.log2)

## Predicted values on the testing data
yhat.boost2 <- predict(boost.log2, newdata = test_df, n.trees = 5000)

## Compute the test MSE
mean((yhat.boost2 - tree_final_data_log.test) ^ 2)
#MSE of 0.1435316
```

```r
set.seed (123)
train_df$instant_bookable <- factor(train_df$instant_bookable)
```

```
boost.log3 <- gbm( logprice ~ review_scores_rating + host_is_superhost +
                    host_listings_count + host_identity_verified +
                    room_type + bathrooms + bedrooms +
                    minimum_nights + number_of_reviews + cancellation_policy +
                    instant_bookable + host_since_duration + location_3ways +
                    cleaning_fee, data = train_df, distribution = "gaussian",
                  n.trees = 5000, interaction.depth = 2)
summary(boost.log3)

## Predicted values on the testing data
yhat.boost3 <- predict(boost.log3, newdata = test_df, n.trees = 5000)

## Compute the test MSE
mean((yhat.boost3 - tree_final_data_log.test) ^ 2)
#MSE of 0.1318706
```

## GAM

Code below.

```
library("gam")
#Check if non-linearity exists and see what degree freedom is the best using scatter plots and ANOVA
poly1 = lm(logprice~poly(bedrooms,4), data = airbnb1)
summary(poly1)
poly2 = lm(logprice~poly(bathrooms,3),data = airbnb1)
summary(poly2)
poly3 = lm(logprice ~ poly(number_of_reviews,4),data = airbnb1)
summary(poly3)
plot(bathrooms, logprice)

#gam1 is trying natural spline
gam1 = lm(logprice ~ ns(bedrooms,4)+ns(bathrooms,2)+review_scores_rating+host_is_superhost+host_listings
summary(gam1)
#gam1p is trying smooth spline
gam1p = lm(logprice ~ s(bedrooms,4)+s(bathrooms,2)+review_scores_rating+host_is_superhost+host_listings_
summary(gam1p)
#Using best subset selection to determine what variables to remain
bestgam = regsubsets(logprice ~ ns(bedrooms,4)+ns(bathrooms,2)+review_scores_rating+host_is_superhost+ho
plot(bestgam, scale = "adjr2")

gam2 = lm(logprice ~ ns(bedrooms,4)+ns(bathrooms,2)+review_scores_rating+host_is_superhost+host_listing
summary(gam2)
#calculate MSE(gam)
predictedvalues1 = predict(gam2, newdata = test)
plot(predictedvalues1, test$logprice)
MSE2 = mean((predictedvalues1-test$logprice)^2)
#We can see that the MSE is around 0.137.
```

## Neural Networks

```r
# NN - Part 1: Data transformation
#amsterdam <- read.csv("st445_final_data", header = T)

amsterdam <- amsterdam[,-1]
#dummify the data
amsterdam <- mutate(amsterdam,
                    instant_bookable = ifelse(instant_bookable == TRUE, 1, 0))
#output_vector = amsterdam[,'logprice']
amsterdam <- fastDummies::dummy_cols(amsterdam)
#amsterdam <- amsterdam[,-c(5,10,13,14,22,26)]
amsterdam <- amsterdam[,-c(5,10,13,14)]

# Set training and testing dataset
set.seed(123)
trainingsize <- floor(0.7 * nrow(amsterdam))
trainindex <- sample(seq_len(nrow(amsterdam)), size = trainingsize)

train_df <- amsterdam[trainindex,]
test_df <- amsterdam[-trainindex,]

# split up train features(x) and train targets(y)
train_data <- as.matrix(train_df[,-12])
train_targets <- as.array(train_df[,12])

# split up test features(x) and test targets(y)
test_data <- as.matrix(test_df[,-12])
test_targets <- as.array(test_df[,12])

#Scale the data so that all variables are between 0 and 1

mean <- apply(train_data, 2, mean)
std <- apply(train_data, 2, sd)
train_data <- scale(train_data, center = mean, scale = std)
test_data <- scale(test_data, center = mean, scale = std)
```

```r
# NN - PArt 2: Build neural network model
build_model <- function() {
  model <- keras_model_sequential() %>%
    layer_dense(units = 16, activation = "relu",
                input_shape = dim(train_data)[[2]]) %>%
    layer_dense(units = 16, activation = "relu") %>%
    layer_dense(units = 1) # single node because it is a regression ML

  model %>% compile(
    optimizer = "rmsprop",
    loss = "mse",
    metrics = c("mae")
  )
}
```

```r
# NN - Part 3 - change the number of learning iterations, i.e "num_epochs", with 10 and 50
all_scores <- c()
num_epochs <- 50
```

```r
# Build the Keras model (already compiled)
model <- build_model()
summary(model)
# Train the model (in silent mode, verbose=0)
model %>% fit(train_data, train_targets,
                epochs = num_epochs, batch_size = 1, verbose = 0)

# Evaluate the model on the validation data
results <- model %>% evaluate(test_data, test_targets, verbose = 0)
#all_scores <- c(all_scores, results$mean_absolute_error)
all_scores <- c(all_scores, results$loss)
#MSE: 0.149141778 using num_epoch = 10, batch size=1
#MSE: 0.15116772 using num_epoch = 50, batch size=1
```

## XGBoost

```r
library(dplyr)
library(xgboost)
```

```
##
## Attaching package: 'xgboost'

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

```r
library(stringr)
library(caret)
```

```
## Loading required package: lattice

##
## Attaching package: 'caret'

## The following object is masked from 'package:purrr':
##
##     lift
```

```r
library(car)
library(fastDummies)
library(ModelMetrics)
```

```
##
## Attaching package: 'ModelMetrics'

## The following objects are masked from 'package:caret':
##
##     confusionMatrix, precision, recall, sensitivity, specificity

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

```r
#amsterdam <- read_csv('st443_final_data')
#amsterdam <- amsterdam[,-c(1,15)]
```

```r
amsterdam <- mutate(amsterdam,
                    instant_bookable = ifelse(instant_bookable == TRUE, 1, 0))

amsterdam <- fastDummies::dummy_cols(amsterdam)
amsterdam <- amsterdam[,-c(5,10,13,16,22,26)]

#Set training and testing dataset
traingsize = floor(0.7*nrow(amsterdam))
set.seed(123)
trainindex = sample(seq_len(nrow(amsterdam)),size = traingsize)

train_df <- amsterdam[trainindex,]
test_df <- amsterdam[-trainindex,]

trainmatrix <- as.matrix(train_df, rownames.force = NA)
testmatrix <- as.matrix(test_df, rownames.force = NA)
dtrain <- as(trainmatrix, "sparseMatrix")
dtest <- as(testmatrix, "sparseMatrix")

train_data <- xgb.DMatrix(data = dtrain[,-12], label = dtrain[,"logprice"])
test_data <- xgb.DMatrix(data = dtest[,-12])
#tune parameters using a full grid search
xgb_grid = expand.grid(
  nrounds = 1000,
  eta = c(0.1, 0.05, 0.01),
  max_depth = c(2, 3, 4, 5, 6),
  gamma = 0,
  colsample_bytree=1,
  min_child_weight=c(1, 2, 3, 4 ,5),
  subsample=1
)
#Find the best hyperparameter values using 5 fold cross validation
my_control <-trainControl(method="cv", number=5)

# Not run, takes ages
#xgb_caret <- train(x = train_df[-12], y = train_df$logprice,
#                    method='xgbTree', trControl= my_control,
#                    tuneGrid = xgb_grid)
#xgb_caret$bestTune
# nrounds = 1000, max_depth = 5, eta = 0.01, min_child_weight = 1

#xgb_tune <-train(logprice ~.,
#                  data = train_df,
#                  method="xgbLinear",
#                  metric = "RMSE",
#                  trControl = cv.ctrl,
#                  tuneGrid = xgb.grid
#)

default_param <- list(
  objective = "reg:linear",
  booster = "gbtree",
  eta=0.01, #default = 0.3
```

```
  gamma=0,
  max_depth=5, #default=6
  min_child_weight=1, #default=1
  subsample=1,
  colsample_bytree=1
)
##use cross validation to determine the optimal number of rounds
#xgbcv <- xgb.cv( params = default_param,
#                  data = dtrain, nrounds = 2000,
#                  nfold = 5,
#                  showsd = T,
#                  stratified = T,
#                  print_every_n = 40,
#                  early_stopping_rounds = 10,
#                  maximize = F,
#                  label = dtrain[,"logprice"])

#Calculate the mse
xgb_mod <- xgb.train(data = train_data, params = default_param, nrounds = 1300)
XGBpred <- predict(xgb_mod, test_data)
rmse <- rmse(test_df$logprice,XGBpred)

library(Ckmeans.1d.dp) #required for ggplot clustering
mat <- xgb.importance(feature_names = colnames(train_df[-12]),model = xgb_mod)
xgb.ggplot.importance(importance_matrix = mat[1:20], rel_to_first = TRUE)
```