Homework assignment 1

Data Analysis 4: Prediction Analytics with Introduction to Machine Learning 2017/2018 Winter

Peter Paziczki

2018 februĂ˘r 11

1. Prediction exercise for London

1.1 Loading the full London AirBnB dataset

I am loading the `airbnb_london_workfile.csv` data set, it has more than 50,000 observations and 74 variables. Now I need to do proper data cleaning and preparation.

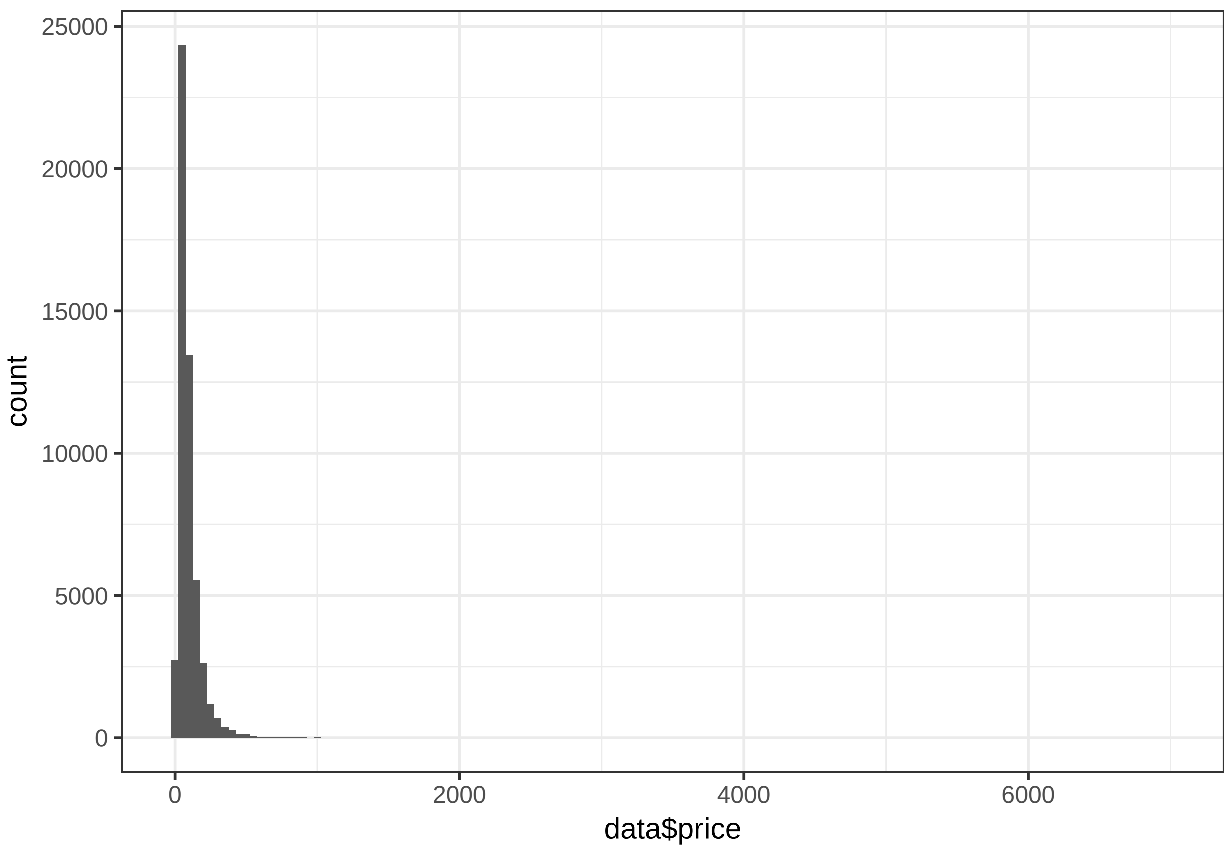
```
## Loading required package: lattice
```

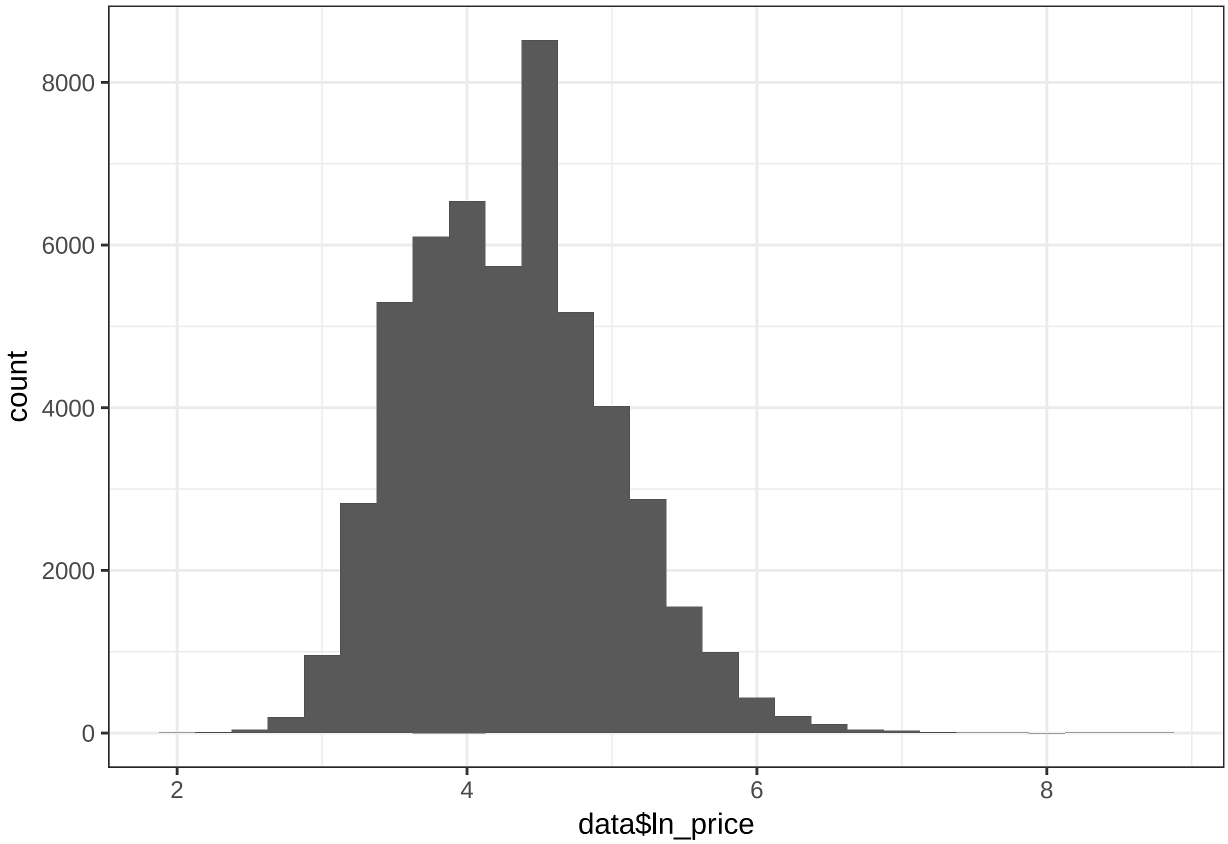
1.2 Data praparation

1.2.1 Price

The target of this exercise is to predict price, so first let's have a better understanding of price variable.

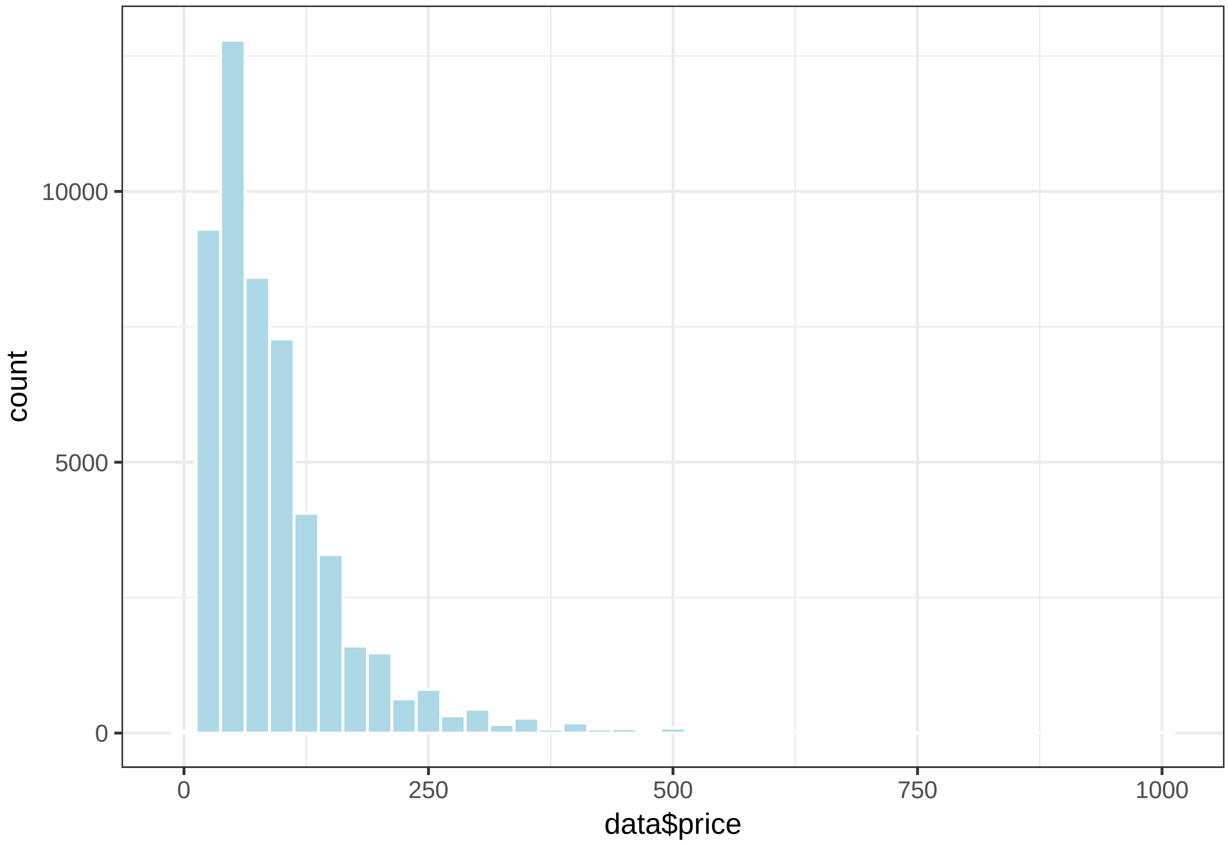
```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    8.00   43.00   75.00   96.67  120.00 7000.00
```



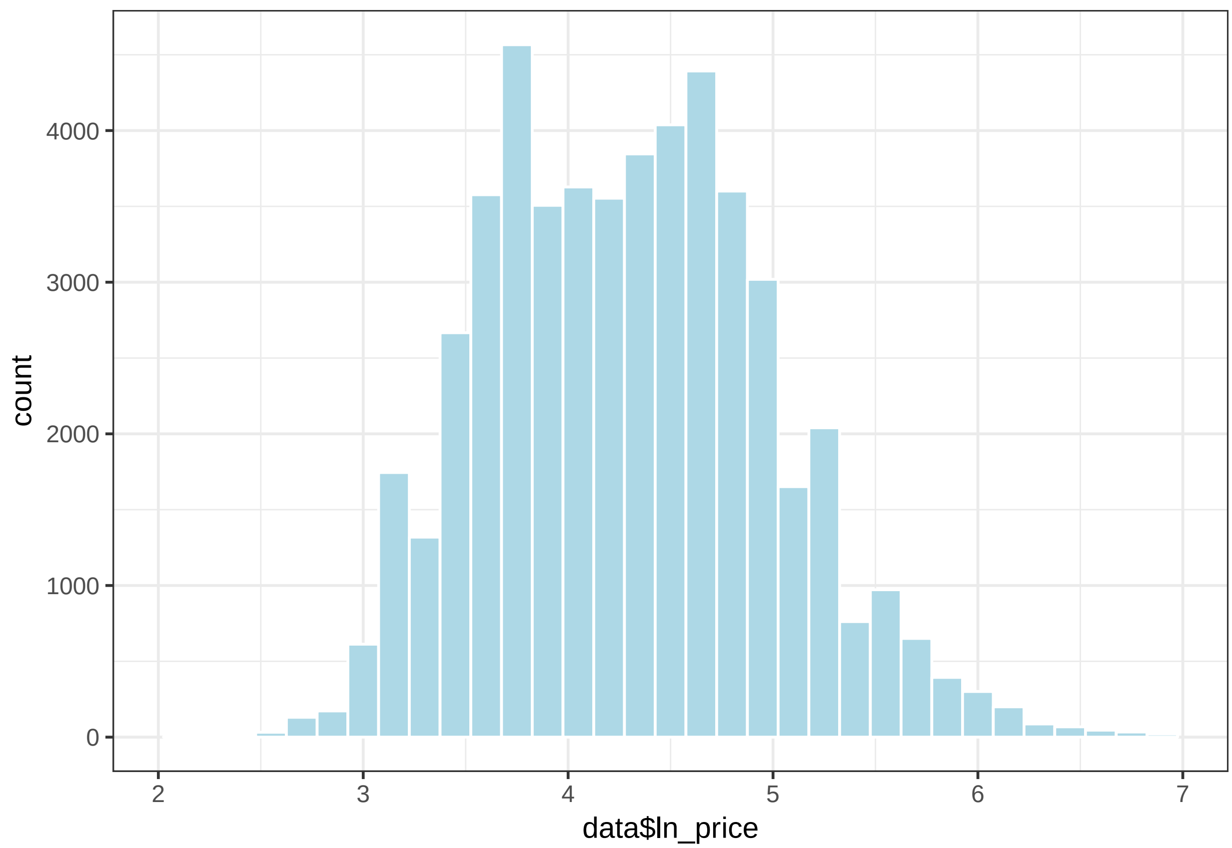


The mean and the median are far from each other, the histogram of price is skewed to the irght and there seem to be a few very large numbers. After taking the logarithm of price (`ln_price`) we got a normal like distribution, but dropping the observation with prices above would most likely yield a neater distribution.

I dropped the observation with prices above 1,000 ad the histogram of price variable is still skewed to the right, but it is much neater, so do the m dropping the observations that have a higher price than 1,000. The histograms of `price` and `ln_price` variables look better now in a sense of being less skewed and are being closer to a normal like distribution, repsectively.



10000
5000
0
count
0
250
500
750
1000
data$price



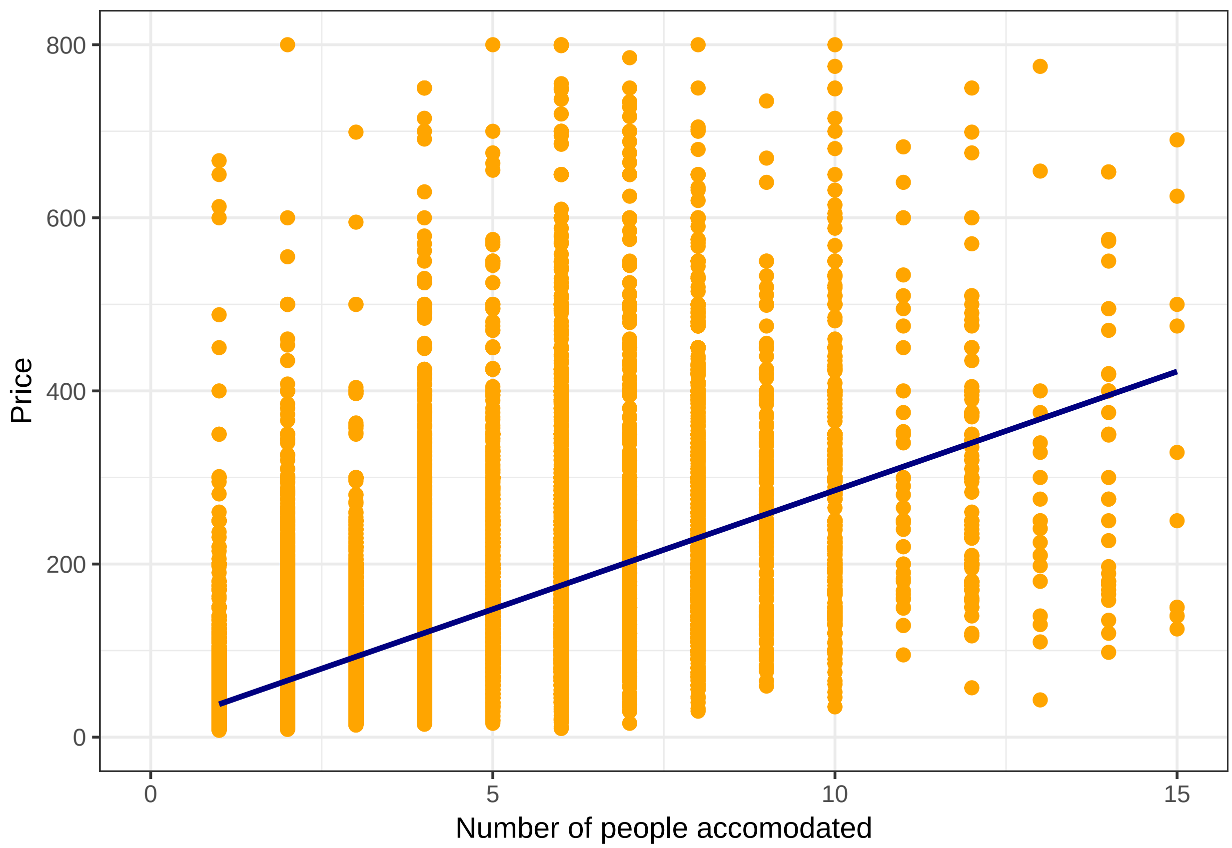
1.2.2 Number of people accomodated

Let's have some summary statistics about mean prices by number of people accomodated.

```
##     n_accommodates mean_price min_price max_price     n
##  1:              1   42.06018         8      1000  6265
##  2:              2   63.54453         9      1000 23668
##  3:              5  153.27265        16       900  2285
##  4:              4  122.65828        15      1000  9543
##  5:             10  295.17266        35       950   278
##  6:              3   90.95585        14      1000  3647
##  7:              7  221.34460        16       995   769
##  8:              8  231.31256        30       986  1107
##  9:              9  266.16337        59      1000   202
## 10:             13  281.75000        43       775    20
## 11:             11  300.83721        95       682    43
## 12:              6  178.39092        10      1000  3658
## 13:             16  353.27586        26       850    29
## 14:             12  326.58163        57      1000    98
## 15:             15  324.90909       125       690    11
## 16:             14  365.35294        98       901    34
```

```
## Warning: Removed 75 rows containing non-finite values (stat_smooth).
```

```
## Warning: Removed 75 rows containing missing values (geom_point).
```



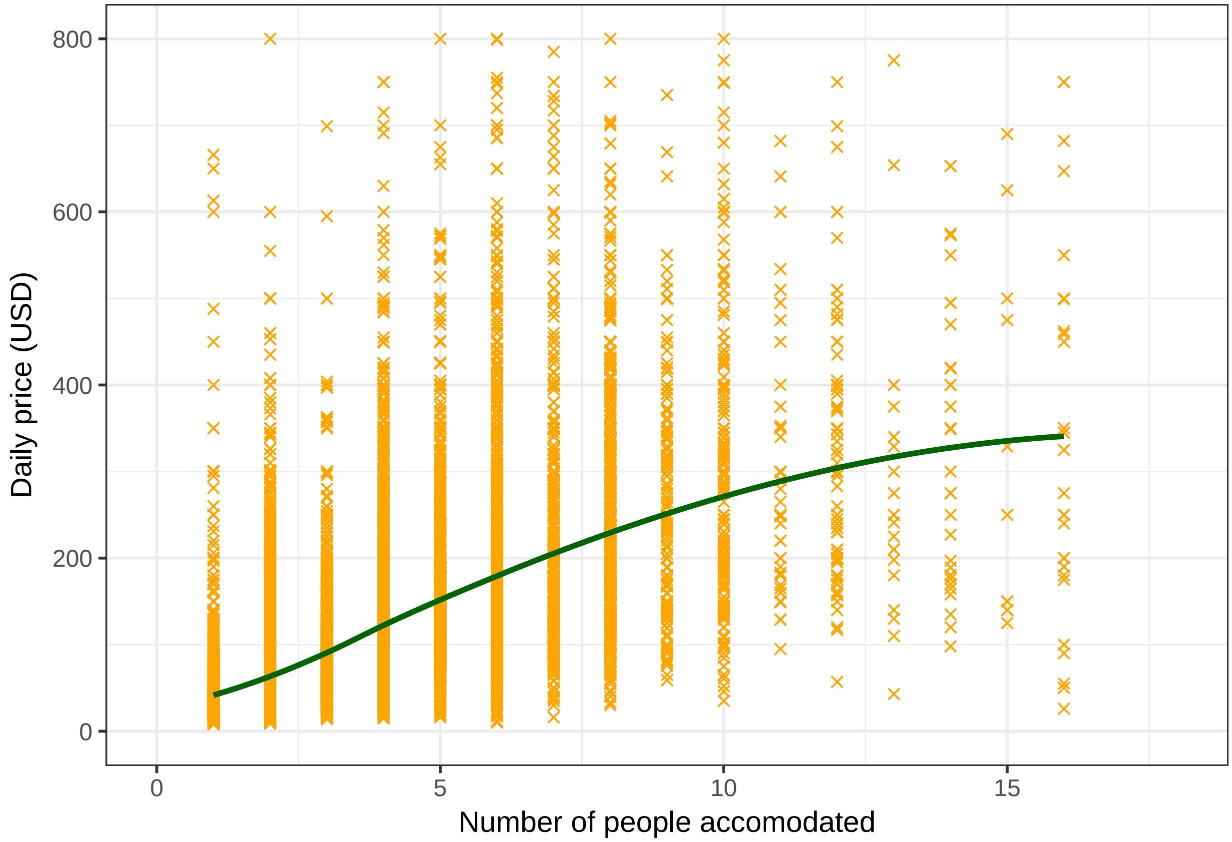
There seems to be a general positive relation between `price` and `n_accommodates` variables, the more guest accomodated, the higher the price is on average. Let's run a linear regression on `ln_price` and on `n_accommodates` or `ln_accommodates`.

```
##
## Call:
## lm(formula = ln_price ~ n_accommodates + n_accommodates2, data = data)
##
## Coefficients:
##     (Intercept)   n_accommodates  n_accommodates2
##         3.21813          0.44175         -0.02085
```

```
##
## Call:
## lm(formula = ln_price ~ ln_accommodates, data = data)
##
## Coefficients:
##     (Intercept)  ln_accommodates
##          3.4752           0.8627
```

```
##
## Call:
## lm(formula = ln_price ~ n_accommodates, data = data)
##
## Coefficients:
##    (Intercept)  n_accommodates
##         3.5423          0.2476
```

Let's run a loess regression on `price` and on `n_accommodates`.

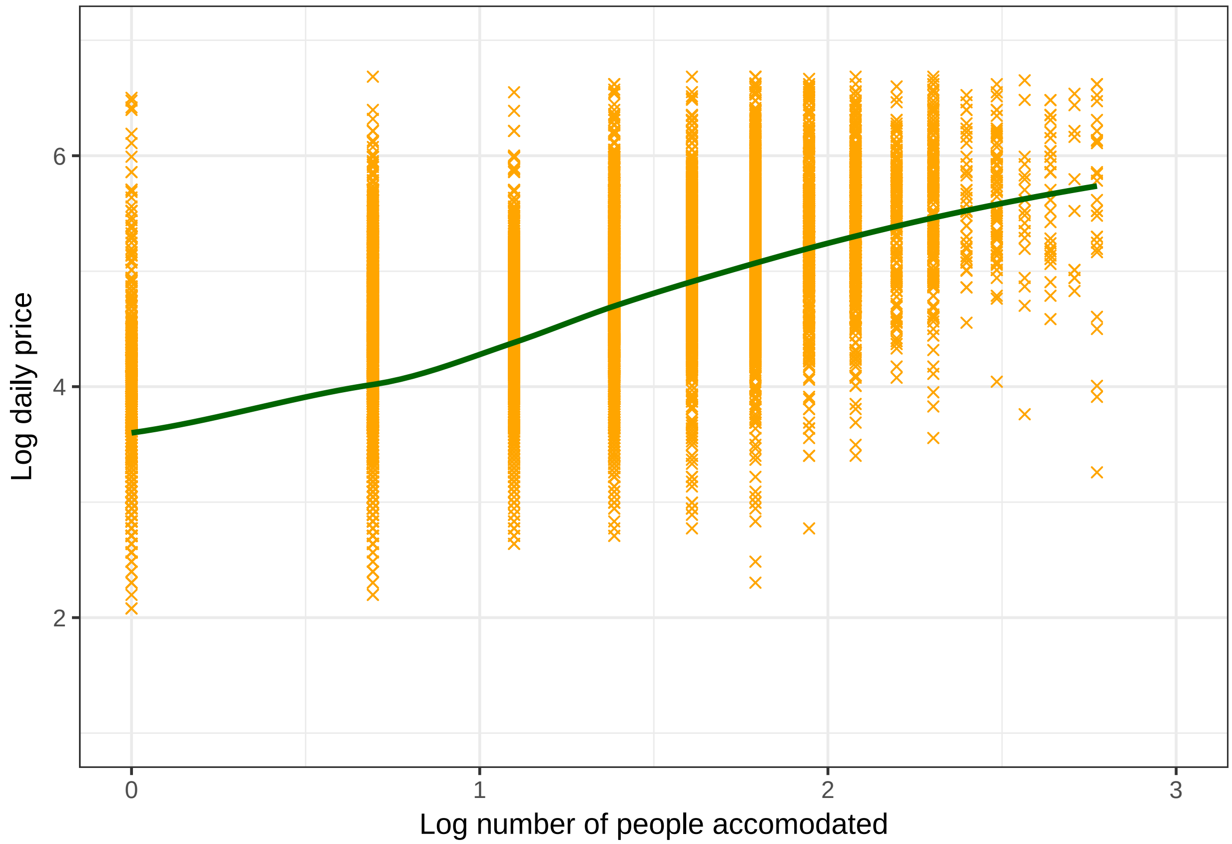


There is some non-linearity, let's run a loess regression on `ln_price` and on `ln_accommodates`.

```
## Warning in simpleLoess(y, x, w, span, degree = degree, parametric =
## parametric, : pseudoinverse used at 0.69315

## Warning in simpleLoess(y, x, w, span, degree = degree, parametric =
## parametric, : neighborhood radius 0.69315

## Warning in simpleLoess(y, x, w, span, degree = degree, parametric =
## parametric, : reciprocal condition number 1.2137e-014
```



This regression is much closer to being linear, it is better to work with, we just need to be careful when interpreting.

1.2.3 Beds

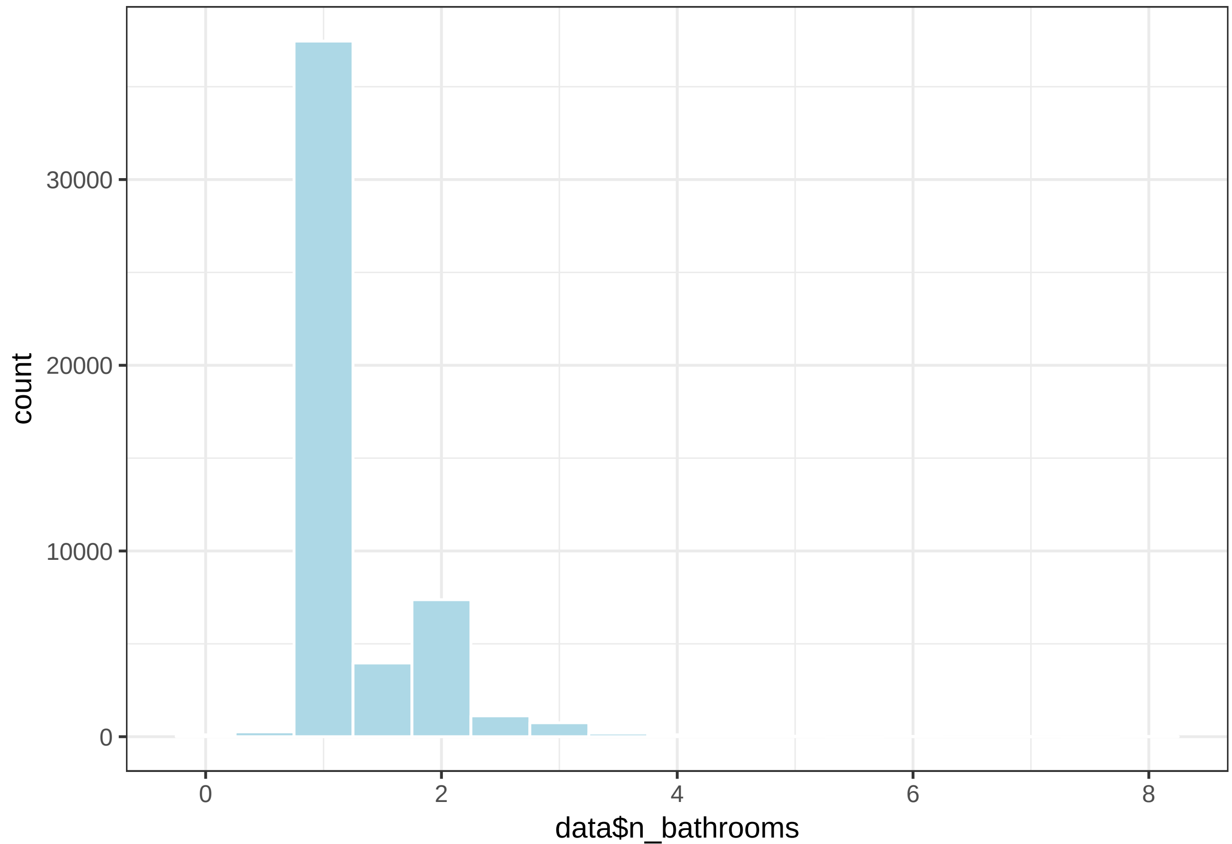
Let's have some summary statistics about mean prices by number of beds and take the logarithm of the number of beds.

```
##     n_beds mean_price min_price max_price     n
##  1:      1   62.73562         8      1000 30781
##  2:      2  114.79689         9      1000 12171
##  3:      4  189.95861         8       920  2223
##  4:      7  277.58721        16      1000   172
##  5:      5  225.24239        18       995   854
##  6:     NA   74.88623        15       600   167
##  7:     13  221.50000        43       400     2
##  8:      6  241.81182        12      1000   457
##  9:      3  160.31397        10       990  4631
## 10:      8  275.52137        25       850   117
## 11:     12  246.33333        30       450     9
## 12:      9  308.04545        45       775    22
## 13:     10  211.64516        17       625    31
## 14:     14  282.60000        26       750     5
## 15:      0   33.00000        33        33     1
## 16:     16  377.12500        50       750     8
## 17:     11  443.16667       110       899     6
```

1.2.4 Bathroom

Let's have a quick look at the histogram of number of bathrooms.

```
## Warning: Removed 237 rows containing non-finite values (stat_bin).
```



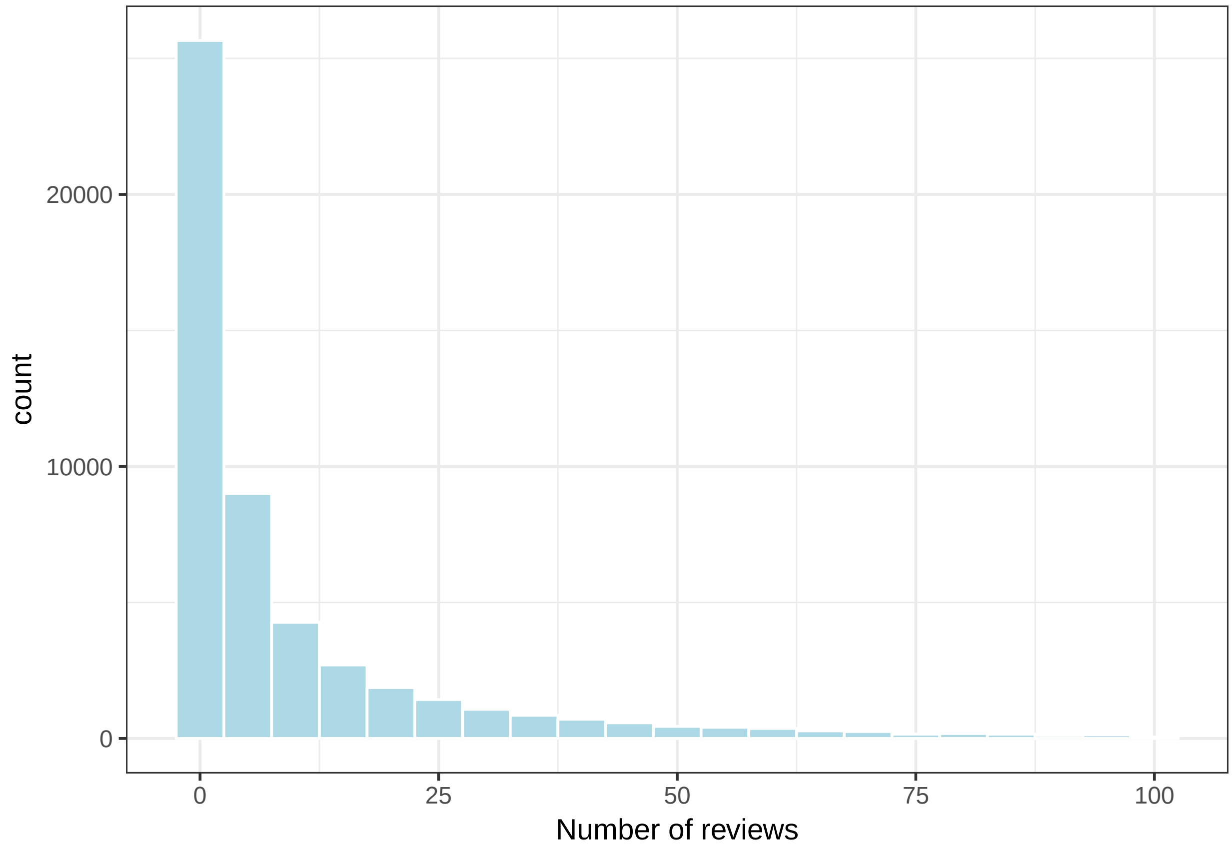
As in the majority of cases there is only one bathroom per accommodation, it would be wise to group the bathrooms into categorical variables based on their number per accommodation:

```
##    f_bathroom mean_price     n
## 1:          1   79.18116 41423
## 2:          2  165.24167  9633
## 3:          0   49.90385   364
## 4:         NA   90.98312   237
```

There are a few NAs among the observations, but it is a negligible fraction of our data. In approx. 80% of the cases there is one bathroom in the accommodation.

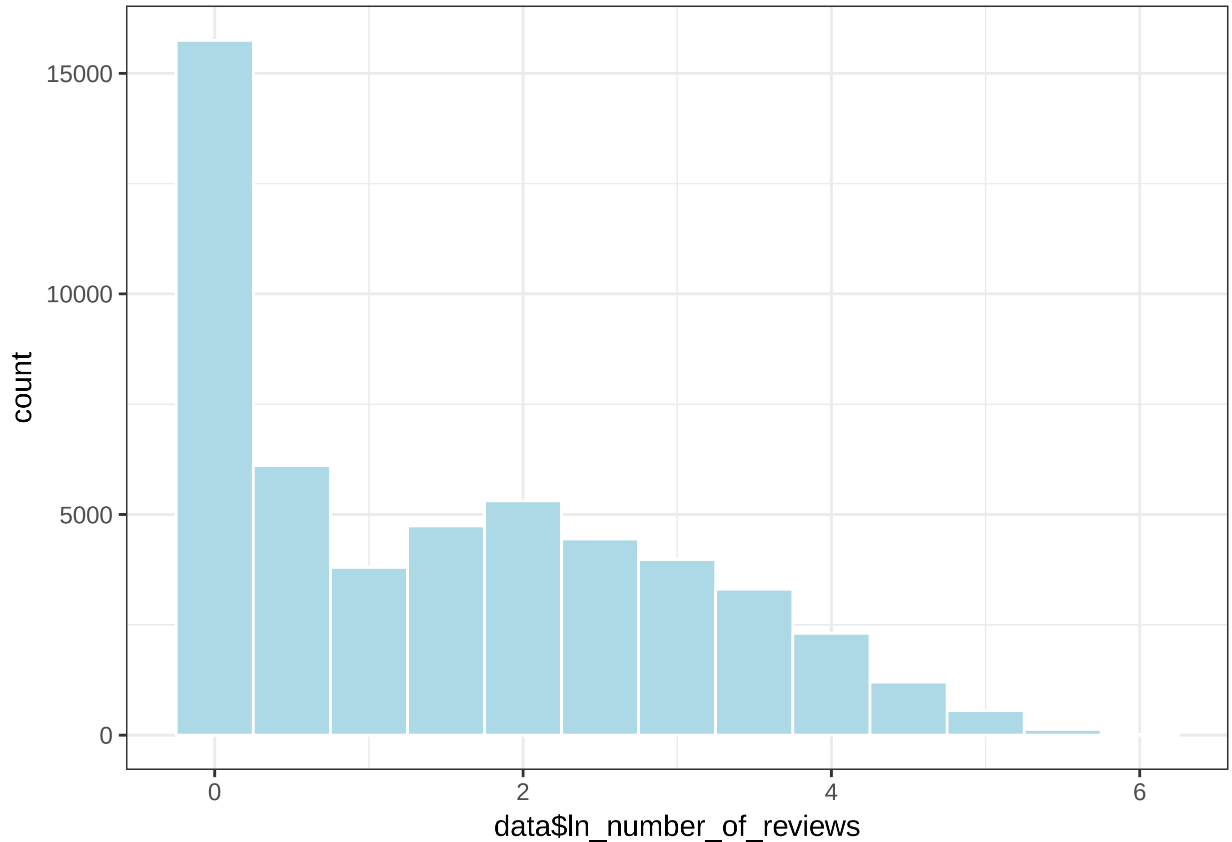
1.2.5 Reviews

Number of reviews can be a good predictor, let's have a look at the distribution:



It is highly skewed to the right, let's have the logarithm of it and check the distribution again:

```
# number of reviews: use logs as well
data[,ln_number_of_reviews:=log(n_number_of_reviews+1)]
qplot(data$ln_number_of_reviews,
      geom="histogram", binwidth=0.5, fill=I("lightblue"), col=I("white"))+
  theme_bw()
```




```
#ggsave("F14_h_ln_number_of_reviews.png")
## it is still exponential ... but not that high
```

It still seems to be exponential, but is is alreay less skewed. It probably makes sense to group the `n_number_of_reviews` into groups (factor variable) with such zero review, 1 - 51 reviews or more.

```
##    f_number_of_reviews median_price mean_price     n
## 1:                   1           75   92.61174 32687
## 2:                   0           70  101.68205 15748
## 3:                   2           75   87.81527  3221
## 4:                  NA           80   80.00000     1
```

Let's see if there is any relation between `ln_price` and the above created groups or the `ln_number_of_reviews`, which is the logairthm of the number of reviews.

```
##
## Call:
## lm(formula = ln_price ~ f_number_of_reviews, data = data)
##
## Coefficients:
##          (Intercept)  f_number_of_reviews1  f_number_of_reviews2
##              4.32914              -0.04370              -0.03215
```

```
##
## Call:
## lm(formula = ln_price ~ ln_number_of_reviews, data = data)
##
## Coefficients:
```

```
##          (Intercept)  ln_number_of_reviews
##             4.303371             -0.002558
```

The results of the first regression shows that there might be some relation between `ln_price` and `f_number_of_reviews`, not that strong but might be relevant. In case of `f_number_of_reviews` I did not observe a strong relation with `ln_price`.

1.2.6 Time since the first review

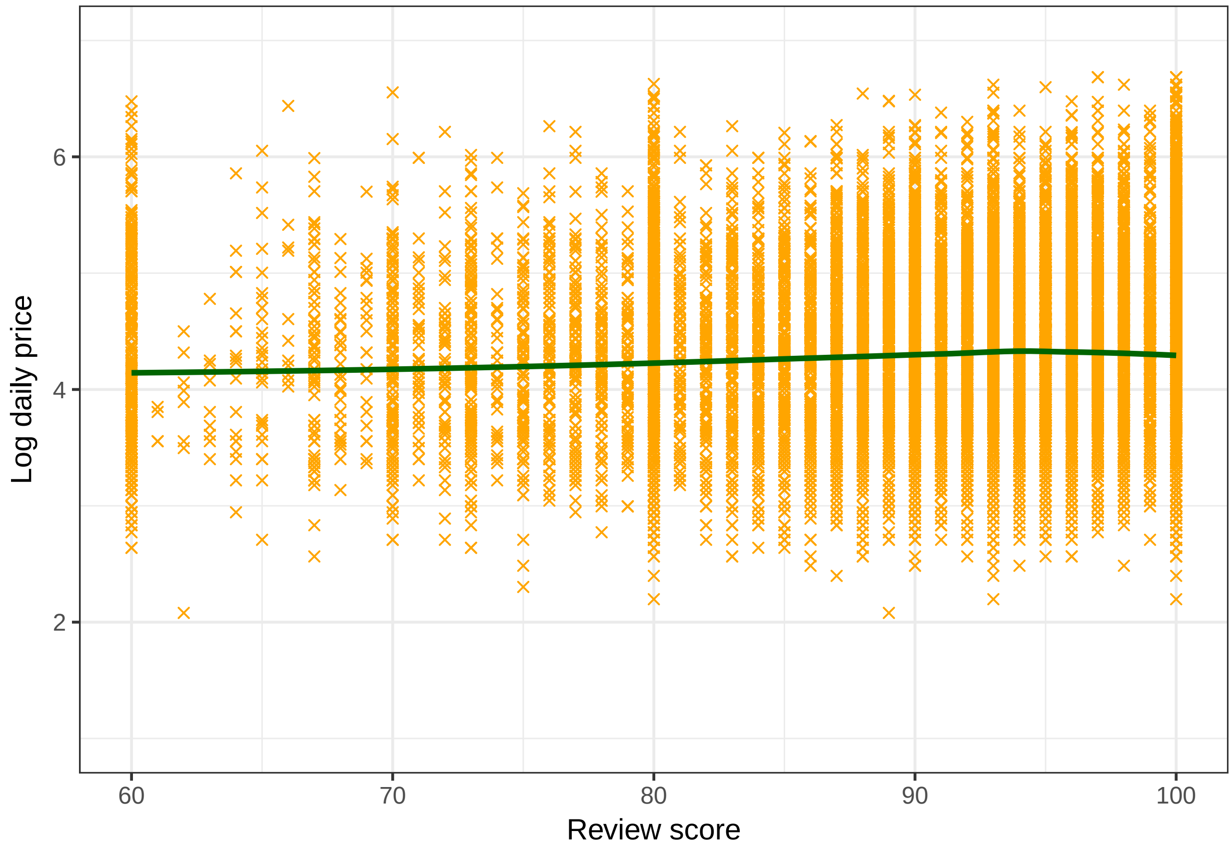
```
## Warning: Removed 2994 rows containing non-finite values (stat_smooth).
```

```
## Warning: Removed 2994 rows containing missing values (geom_point).
```

There might be some non-linear pattern in the data, we will use it as potential polinomial predictor.

1.2.7 Review score

It can be meaningful and useful to not only look at the full population but subsamples too.




```
##
## Call:
## lm(formula = ln_price ~ n_review_scores_rating, data = data)
##
## Coefficients:
##            (Intercept)  n_review_scores_rating
##               4.001235                0.003131
```

```
##
## Call:
## lm(formula = ln_price ~ ln_review_scores_rating, data = data)
##
## Coefficients:
##             (Intercept)  ln_review_scores_rating
##                  3.3890                   0.1994
```

Having the log of review score seems to be good idea, we are going to use as a potential predictor.

1.2.8 Minimum nights

Let's run a linear regression on log of price and minimum nights.

```
##
## Call:
## lm(formula = ln_price ~ n_minimum_nights, data = data)
##
## Coefficients:
```

```
##      (Intercept)  n_minimum_nights
##          4.28689           0.00045
```

```
##
## Call:
## lm(formula = ln_price ~ f_minimum_nights, data = data)
##
## Coefficients:
##       (Intercept)  f_minimum_nights2  f_minimum_nights3
##            4.0613             0.2927             0.4153
```

There seems to be a very weak relationship between `ln_price` and `n_minimum_nights`, but it seems to be a good ide to group nights.

1.2.9 Categorical variables

These are the categorical variables we already have:

```
##    f_property_type mean_price     n
## 1:       Apartment   94.11897 26678
## 2:           House   86.38118  8471
##        f_room_type mean_price     n
## 1:    Private room   47.22327 15748
## 2: Entire home/apt  130.42737 19082
## 3:     Shared room   31.83072   319
##    f_cancellation_policy mean_price     n
## 1:              flexible   69.28579  9101
## 2:              moderate   87.19869  9648
## 3:                strict  107.97427 16400
##    f_bed_type mean_price     n
## 1:   Real Bed   93.04572 34403
## 2:      Couch   55.74933   746
```

1.3 Preparing model environment

1.3.1 Defining functions

We needed to define function to compute means squarred error for simple and log models. In case of log models a correction term was also needed to consider.

1.3.2 Defining models

The task is to create four different models, with the fourth being the most complex. I am starting with a very simple level model with only one predictor, `n_accommodates`, just ot have a good benchmark model. Then I am improving that prediction in a sense of having much mode predictor but still predicting price in level. Then I am switching to predict log price with one simple predictor, just to have a good benchmark for the log model. The fourth model, as requested, is the most complex one with the most predictors.

1.3.3 Creating training and test data sets

I am dividing the data into training (90%) and test (10%) sets.

1.3.4 Running linear models without CV

Please find a table below summarizing the MSE, RMSE and BIC scores for the level models I have previously defined.

```
## Warning in predict.lm(model, newdata = data_train): prediction from a rank-
## deficient fit may be misleading

## Warning in predict.lm(model, newdata = data_test): prediction from a rank-
## deficient fit may be misleading

##               modellev1    modellev2   modellog1   modellog2
## mse_train    3154.84854   2682.14126  3157.12287  2300.61492
## rmse_train     56.16804     51.78939    56.18828    47.96473
## mse_test     3073.05026   2586.87831  3095.85775  2221.32555
## rmse_test      55.43510     50.86136    55.64043    47.13094
## BIC        344670.00299 338468.02442 42216.87388 27797.75463
```

1.3.5 Running log models 10-fold CV

Please find a table below summarizing the MSE, RMSE and BIC scores for the level models I have previously defined.

```
## Warning in predict.lm(model, newdata = data_train): prediction from a rank-
## deficient fit may be misleading

## Warning in predict.lm(model, newdata = data_test): prediction from a rank-
## deficient fit may be misleading

## Warning in predict.lm(model, newdata = data_train): prediction from a rank-
## deficient fit may be misleading

## Warning in predict.lm(model, newdata = data_test): prediction from a rank-
## deficient fit may be misleading

## Warning in predict.lm(model, newdata = data_train): prediction from a rank-
## deficient fit may be misleading

## Warning in predict.lm(model, newdata = data_test): prediction from a rank-
## deficient fit may be misleading

## Warning in predict.lm(model, newdata = data_train): prediction from a rank-
## deficient fit may be misleading

## Warning in predict.lm(model, newdata = data_test): prediction from a rank-
## deficient fit may be misleading

## Warning in predict.lm(model, newdata = data_train): prediction from a rank-
## deficient fit may be misleading

## Warning in predict.lm(model, newdata = data_test): prediction from a rank-
## deficient fit may be misleading

## Warning in predict.lm(model, newdata = data_train): prediction from a rank-
## deficient fit may be misleading

## Warning in predict.lm(model, newdata = data_test): prediction from a rank-
## deficient fit may be misleading

## Warning in predict.lm(model, newdata = data_train): prediction from a rank-
## deficient fit may be misleading
```

```
## Warning in predict.lm(model, newdata = data_test): prediction from a rank-
## deficient fit may be misleading

## Warning in predict.lm(model, newdata = data_train): prediction from a rank-
## deficient fit may be misleading

## Warning in predict.lm(model, newdata = data_test): prediction from a rank-
## deficient fit may be misleading

## Warning in predict.lm(model, newdata = data_train): prediction from a rank-
## deficient fit may be misleading

## Warning in predict.lm(model, newdata = data_test): prediction from a rank-
## deficient fit may be misleading

## Warning in predict.lm(model, newdata = data_train): prediction from a rank-
## deficient fit may be misleading

## Warning in predict.lm(model, newdata = data_test): prediction from a rank-
## deficient fit may be misleading

##               modellev1    modellev2   modellog1   modellog2
## mse_train    3122.42007   2645.96473  3124.87632  2258.90324
## rmse_train     55.87862     51.43894    55.90059    47.52792
## mse_test     3364.57592   2913.50341    58.15214    51.00353
## rmse_test      58.00496     53.97688    53.97688    53.97688
## BIC        344354.04095 338093.54668 42318.86239 27730.91252
```

1.3.6 Running Lasso

The task is to run Lasso on the most complex model.

```
## Loading required package: Matrix

## Loading required package: foreach

## Loaded glmnet 2.0-13
```

The RMSE is 48.7314199.

1.3.7 Running a pcr model (Extra task)

I am trying to improve the simple linear model by using PCA for dimensionality reduction. I am centering and scaling variables and using `pcr` to conduct a search for the optimal number of principal components.

`pcr` is also a linear regression but with principal components as explanatory variables and its hyperparameter is the number of principal components to be used. Now I am doing a `pcr` with a 10-fold cross-validation with a sequence of hyperparameters 1 to 20.

```
## Principal Component Analysis
##
## 31635 samples
##    87 predictor
##
## Pre-processing: centered (156), scaled (156)
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 28471, 28471, 28472, 28471, 28473, 28472, ...
## Resampling results across tuning parameters:
##
```

```
##   ncomp  RMSE      Rsquared   MAE
##    5     48.98097  0.6045621  30.37572
##    6     48.78735  0.6076888  30.37583
##    7     48.25572  0.6162888  29.93562
##    8     48.10214  0.6187393  29.89997
##    9     47.74966  0.6242835  29.95126
##   10     47.74542  0.6243701  29.91837
##
## RMSE was used to select the optimal model using the smallest value.
## The final value used for the model was ncomp = 10.
```

The results show that having the most principal components does not yield a model with the lowest RMSE. In this case having 19 principal components provided the lowest RMSE.

1.3.8 Evaluating pcr model on test set (Extra task)

```
## Warning in x - true_x: longer object length is not a multiple of shorter
## object length

## [1] 106.1168
```

2. Prediction exercise for a borough in London

```
large_boroughs <- data[, .N, by = "f_neighbourhood_cleansed"][N > 999][["f_neighbourhood_cleansed"]]
set.seed(20180210)
borough <- sample(large_boroughs, 1)
data <- data[neighbourhood_cleansed == borough]
```

I have created a small routine to randomy pick a borough that has at leasr a thousand observations. The routine picked Westminster.

2.1 Creating training and test data sets

I am dividing the data into training (90%) and test (10%) sets, as I did previously.

2.2 Running linear models without CV

Please find a table below summarizing the MSE, RMSE and BIC scores for the level models I have previously defined.

```
## Warning in predict.lm(model, newdata = data_train): prediction from a rank-
## deficient fit may be misleading

## Warning in predict.lm(model, newdata = data_test): prediction from a rank-
## deficient fit may be misleading

##              modellev1   modellev2  modellog1  modellog2
## mse_train   5751.86034  4695.67276 5758.50006 3932.44269
## rmse_train    75.84102    68.52498   75.88478   62.70919
## mse_test    5579.67435  4341.57673 5497.23675 4035.06441
## rmse_test     74.69722    65.89064   74.14335   63.52216
## BIC        39740.20563 39069.23972 4135.61859 4596.26510
```

2.3 Running log models 10-fold CV

Please find a table below summarizing the MSE, RMSE and BIC scores for the level models I have previously defined.

```
## Warning in predict.lm(model, newdata = data_train): prediction from a rank-
## deficient fit may be misleading

## Warning in predict.lm(model, newdata = data_test): prediction from a rank-
## deficient fit may be misleading

## Warning in predict.lm(model, newdata = data_train): prediction from a rank-
## deficient fit may be misleading

## Warning in predict.lm(model, newdata = data_test): prediction from a rank-
## deficient fit may be misleading

## Warning in predict.lm(model, newdata = data_train): prediction from a rank-
## deficient fit may be misleading

## Warning in predict.lm(model, newdata = data_test): prediction from a rank-
## deficient fit may be misleading

## Warning in predict.lm(model, newdata = data_train): prediction from a rank-
## deficient fit may be misleading

## Warning in predict.lm(model, newdata = data_test): prediction from a rank-
## deficient fit may be misleading

## Warning in predict.lm(model, newdata = data_train): prediction from a rank-
## deficient fit may be misleading

## Warning in predict.lm(model, newdata = data_test): prediction from a rank-
## deficient fit may be misleading

## Warning in predict.lm(model, newdata = data_train): prediction from a rank-
## deficient fit may be misleading

## Warning in predict.lm(model, newdata = data_test): prediction from a rank-
## deficient fit may be misleading

## Warning in predict.lm(model, newdata = data_train): prediction from a rank-
## deficient fit may be misleading

## Warning in predict.lm(model, newdata = data_test): prediction from a rank-
## deficient fit may be misleading

## Warning in predict.lm(model, newdata = data_train): prediction from a rank-
## deficient fit may be misleading

## Warning in predict.lm(model, newdata = data_test): prediction from a rank-
## deficient fit may be misleading

## Warning in predict.lm(model, newdata = data_train): prediction from a rank-
## deficient fit may be misleading

## Warning in predict.lm(model, newdata = data_test): prediction from a rank-
## deficient fit may be misleading

## Warning in predict.lm(model, newdata = data_train): prediction from a rank-
## deficient fit may be misleading

## Warning in predict.lm(model, newdata = data_test): prediction from a rank-
## deficient fit may be misleading
```

```
##             modellev1   modellev2  modellog1  modellog2
## mse_train   5819.26644  4742.98526 5828.86641 3928.13366
## rmse_train    76.28412    68.86933   76.34701   62.67482
## mse_test    4971.55152  3919.02135   69.89259   65.30562
## rmse_test     70.50923    62.60209   62.60209   62.60209
## BIC        39791.96706 39126.43043 4124.27714 4574.76926
```

2.4 Running Lasso

The task is to run Lasso on the most complex model.

The RMSE is 64.7916219.

2.5 Running a `pcr` model (Extra task)

I am trying to improve the simple linear model by using PCA for dimensionality reduction. I am centering and scaling variables and using `pcr` to conduct a search for the optimal number of principal components.

`pcr` is also a linear regression but with principal components as explanatory variables and its hyperparameter is the number of principal components to be used. Now I am doing a `pcr` with a 10-fold cross-validation with a sequence of hyperparameters 1 to 20.

```
## Principal Component Analysis
##
## 3456 samples
##   51 predictor
##
## Pre-processing: centered (62), scaled (62)
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 3110, 3110, 3111, 3111, 3110, 3110, ...
## Resampling results across tuning parameters:
##
##   ncomp  RMSE      Rsquared   MAE
##    5     70.75982  0.4719176  44.80693
##    6     70.73372  0.4722922  44.83034
##    7     70.57932  0.4746355  44.86905
##    8     70.58876  0.4744950  44.86823
##    9     70.58981  0.4742887  44.83285
##   10     70.61774  0.4738941  44.85409
##   11     70.49464  0.4758779  44.69748
##   12     70.44752  0.4767914  44.67686
##   13     70.30223  0.4789826  44.58633
##   14     70.31342  0.4787738  44.56473
##   15     70.18301  0.4803677  44.44036
##   16     70.00650  0.4832091  44.21360
##   17     70.00290  0.4831920  44.20429
##   18     69.73573  0.4871917  44.00414
##   19     69.67013  0.4883172  43.93040
##   20     69.59928  0.4894645  43.92551
##   21     69.25256  0.4944695  43.56977
##   22     69.27409  0.4941207  43.59826
##   23     69.25632  0.4942984  43.64092
##   24     69.21286  0.4951360  43.64458
##   25     69.22464  0.4950421  43.69939
```

```
##
## RMSE was used to select the optimal model using the smallest value.
## The final value used for the model was ncomp = 24.
```

```
## [1] " ~ ln_accommodates + ln_beds + f_property_type + f_room_type + ln_days_since + f_bathroom + f_ca
```

The results show that having the most principal components does not yield a model with the lowest RMSE. In this case having 19 principal components provided the lowest RMSE.

2.6 Evaluating pcr model on test set

```
## Warning in x - true_x: longer object length is not a multiple of shorter
## object length
```

```
## [1] 123.4911
```

3. Which model would you choose for your borough and which one for the London results?

Discuss (1 para) which model you would choose (and why?) to predict London prices in 2018 Budapest prices in 2018

4. Extra task for full London data set

We need to compare two modelling options: + predicting prices with the same set of variables, estimated borough by borough or + using a single model that has some interaction terms (such as borough X flat/house, or borough X n_accommodate)