DA4 Term Project | Technical Appendix

Luca Keresztesi

20th April 2017

```
rm(list = ls())

# install.packages("data.table")
# install.packages("Matrix")
# install.packages("xgboost")
# install.packages("randomForest")
# install.packages("caret")
# install.packages("plyr")
# install.packages("moments")
# install.packages("ggplot2")
# install.packages("lmtest")
# install.packages("sandwich")
# install.packages("stargazer")
# install.packages("SparseM")
# install.packages("pROC")
# install.packages("stringr")
# install.packages("dummies")
# install.packages("stats")
# install.packages("randomForest")
# install.packages("ROCR")
# install.packages("pander")
# install.packages("h2o")
# install.packages("h2oEnsemble")

library(moments)
library(plyr)
library(data.table)
library(lmtest)
library(sandwich)
library(stargazer)
library(ggplot2)
library(SparseM)
library(caret)
library(stats)

setwd("C:/Users/Keresztesi Luca/Box Sync/CEU 2nd trimester/Data Analysis 4/Term Project/DA4_Term projec
source('../da_helper_functions.R')
getwd()

property_df <- read.csv("da4_property_aug.csv", header = TRUE)
```

Advertised price of flats available in the 13th district of Budapest, from August 2016. We predict future prices based on past data (6,747 observations) by using regression and RandomForest models. Target variable is price per square meter (psqm). We originally had 47 predictors such as number of floors, air conditioner, parking options, heating type, condition, etc.

Section 1: preparation

Do duplicates matter?

```
# Sort by these variables
property_df <- data.table(property_df)
property_df <- property_df[order(property_df[, 13], property_df[, 16], property_df[,
    2], property_df[, 39], property_df[, 26], property_df[, 25], property_df[,
    21], property_df[, 22], property_df[, 19], property_df[, 4], property_df[,
    38], property_df[, 40], property_df[, 41])]

# Transform the table to data.table Count the number of duplicates according
# to price, sqm, floor and heating and see statistics
property_df <- data.table(property_df)
property_df <- property_df[, `:=`(multiple, .N), by = .(p, sqm, floor, heating)]
count(property_df$multiple)

# Mark the duplicated observations from 1 to the number of duplicates
property_df <- data.table(property_df)
property_df <- property_df[, `:=`(temp, 1:.N), by = c("p", "sqm", "floor", "heating")]
# Assign a number to all the rows to indicate which duplicate it is. (1 =
# the first occurence/no duplicate)
property_df <- as.data.frame(property_df)
count(property_df$temp)

# Flag duplicated observations (flag all duplicated except for one)
property_df$duplicate1 <- as.numeric(property_df$temp != 1)
count(property_df$duplicate1)

# Drop the variable temp
property_df <- property_df[, -49]

# Count the number of duplicates according to all the variables in the list
# and see statistics
property_df <- data.table(property_df)
property_df <- property_df[, `:=`(multiple2, .N), by = .(p, sqm, floor, heating,
    lift_d, aircond_d, hasbalcony, balcony, concrete_blockflat_d, orientation,
    view, parking)]
count(property_df$multiple2)

# Count duplicates from 1 to the number of duplicates and show statistics
property_df <- property_df[, `:=`(temp, 1:.N), by = c("p", "sqm", "floor", "heating",
    "lift_d", "aircond_d", "hasbalcony", "balcony", "concrete_blockflat_d",
    "orientation", "view", "parking")]
property_df <- as.data.frame(property_df)
count(property_df$temp)

# Flag duplicated observations except for 1
property_df$duplicate2 <- as.numeric(property_df$temp != 1)
count(property_df$duplicate2)

# Drop the variable temp
property_df <- property_df[, -51]
```

```
# Create categories for duplicates: 0 if unique, 1 if only duplicated in p,
# sqm, floor and heating, 2 if duplicated all variables in the list
property_df$dup_m <- 0
property_df$dup_m[property_df$duplicate1 == 1 & property_df$duplicate2 == 0] <- 1
property_df$dup_m[property_df$duplicate2 == 1] <- 2
property_df$dup_m_labels <- property_df$dup_m
property_df$dup_m_labels[property_df$dup_m == 0] <- "unique"
property_df$dup_m_labels[property_df$dup_m == 1] <- "maybe duplicate"
property_df$dup_m_labels[property_df$dup_m == 2] <- "duplicate"
property_df$dup_m_labels <- as.factor(property_df$dup_m_labels)
count(property_df$dup_m_labels)

# Check the statistics of several variables by duplicated categories
ddply(property_df, .(dup_m_labels), summarize, freq = length(dup_m_labels),
    mean_sqm = round(mean(sqm), digits = 2), mean_psqm = round(mean(psqm), digits = 2),
    mean_floor2 = round(mean(floor2), digits = 2), mean_new_flat = round(mean(new_flat),
        digits = 2))

# Keep unique and maybe duplicated observations
property_df <- data.table(property_df)
property_df <- property_df[property_df$duplicate2 == 0]
```

Probably due to repeated advertisment, we had to filter out duplicates by only keeping observations with no or little duplication. After this step 6,005 observations were kept.

- Flagged as maybe duplicate if the price, sqm, floor and heating are the same: 742
- Flagged as duplicate if all the above match and also price, lift, airconditioner dummy, balcony dummy, balcony size, concrete blockflat dummy, orientation, view and parking are the same: 1,459
- No duplicates: 4,546

Section 2: Describe raw data

```
# summary statistics with stargazer
property_sum <- property_df[,c(
  2, 4, 13, 14, 15, 20, 21, 22, 23, 25, 26, 27, 47)]
stargazer(
  property_sum,
  header = FALSE,
  out = "summarystats.html",
  type = 'latex',
  title = "Descriptive Statistics for the numeric variables in the prediction")
```

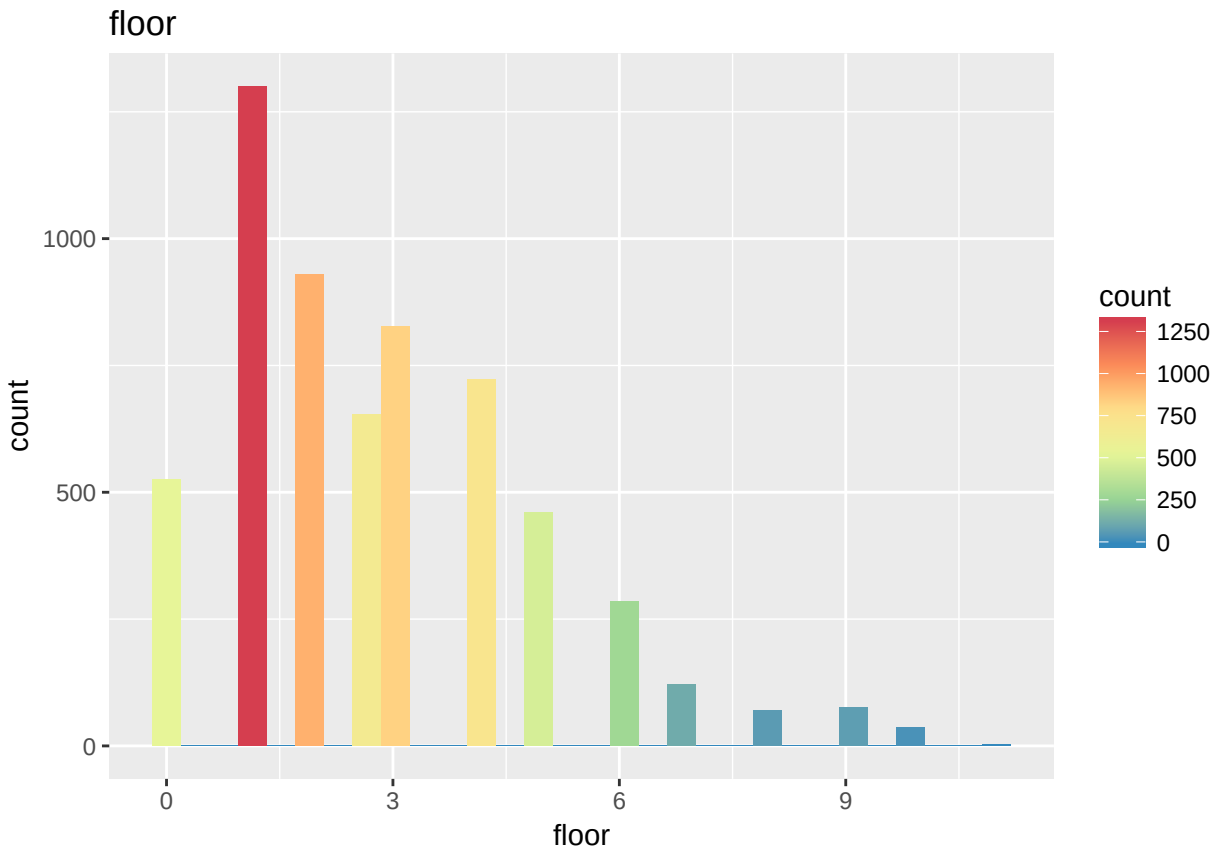
```
# find mean price per squared meter and number of observations by condition,
# heating, view, balcony, orientation and parking
ddply(property_df, .(condition), summarize, freq = length(condition), mean_psqm = round(mean(psqm),
    digits = 3))
ddply(property_df, .(heating), summarize, freq = length(heating), mean_psqm = round(mean(psqm),
    digits = 3))
ddply(property_df, .(view), summarize, freq = length(view), mean_psqm = round(mean(psqm),
    digits = 3))
ddply(property_df, .(orientation), summarize, freq = length(view), mean_psqm = round(mean(psqm),
    digits = 3))
ddply(property_df, .(parking), summarize, freq = length(view), mean_psqm = round(mean(psqm),
```

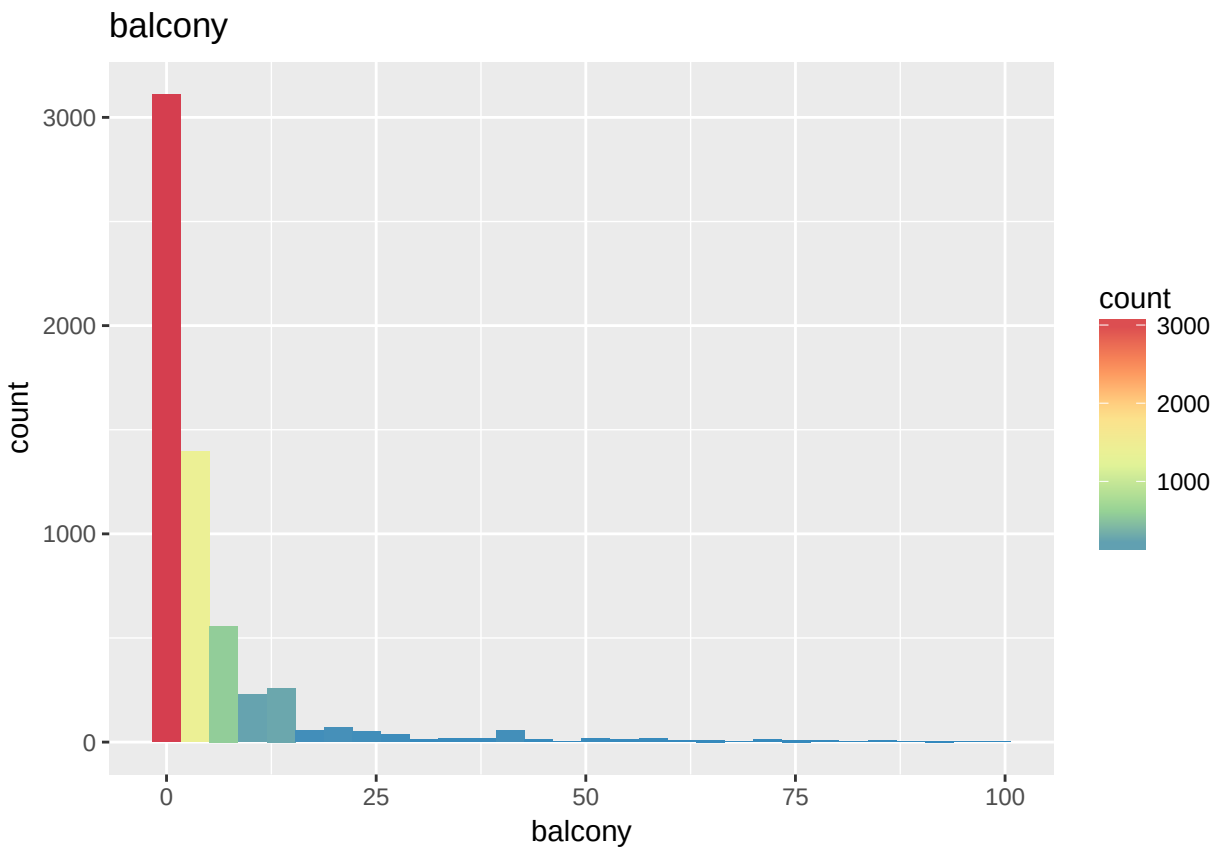
Table 1: Descriptive Statistics for the numeric variables in the prediction

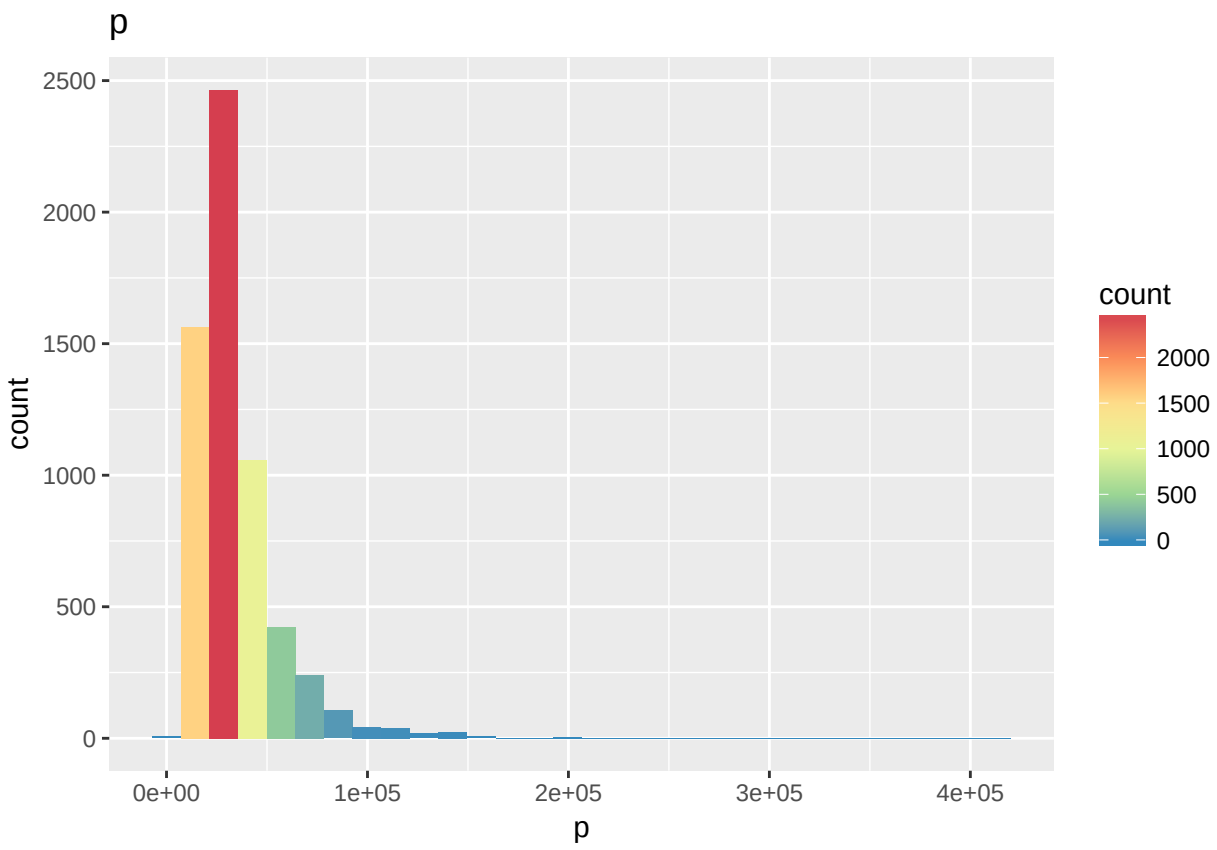
Statistic	N	Mean	St. Dev.	Min	Max
floor	6,005	2.800	2.002	0.000	11.000
balcony	6,005	5.280	11.455	0.000	99.000
p	6,005	34,282.180	22,190.910	5,885.000	419,000.000
nrooms	6,002	1.899	0.930	1	10
nhalfrooms	6,005	0.583	0.778	0	14
psqm	6,005	529.768	165.005	120.000	1,750.000
hasbalcony	6,005	0.493	0.500	0	1
concrete_blockflat_d	6,005	0.105	0.306	0	1
floor2	6,005	2.766	1.932	−1.000	8.000
aircond_d	6,005	0.140	0.347	0	1
lift_d	6,005	0.675	0.468	0	1
ln_psqm	6,005	6.230	0.284	4.787	7.467
new_flat	6,005	0.421	0.494	0	1

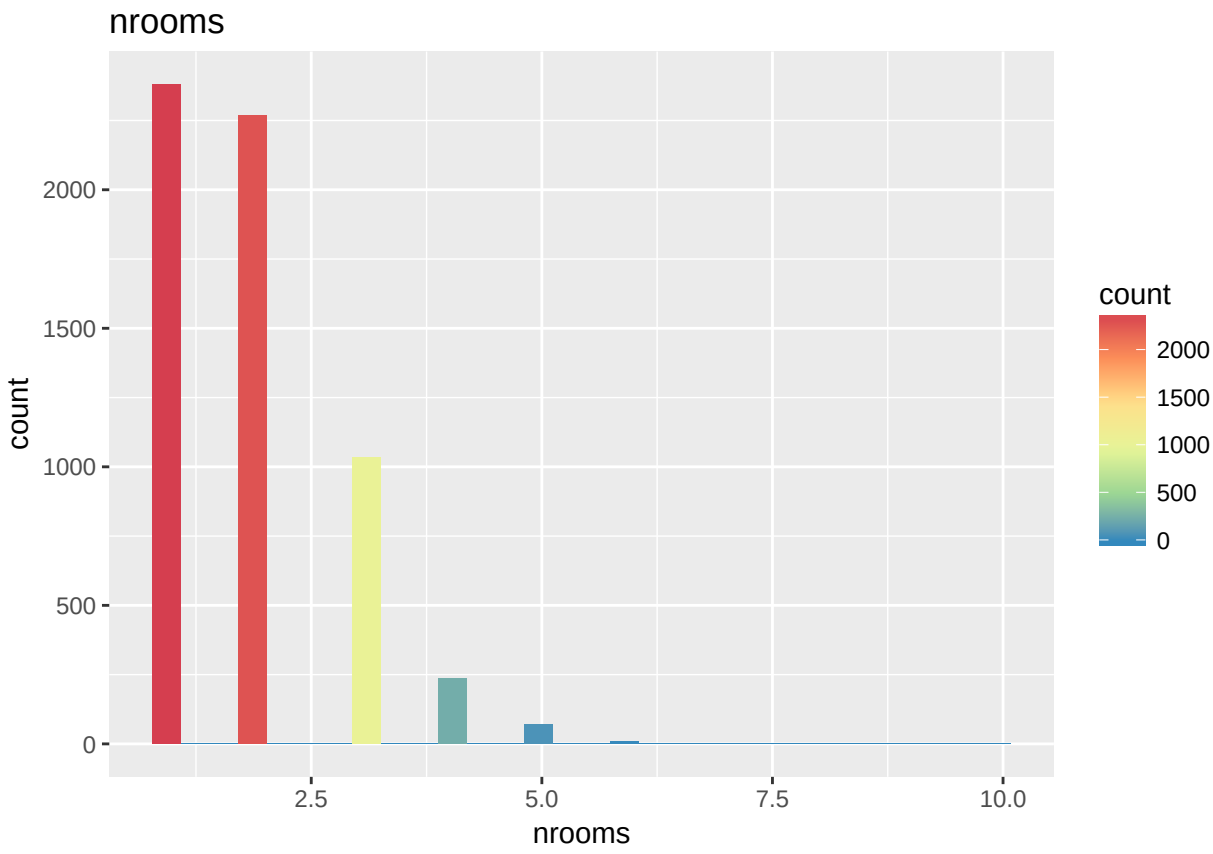
```
    digits = 3))
ddply(property_df, .(balcony), summarize, freq = length(view), mean_psqm = round(mean(psqm),
    digits = 3))
```

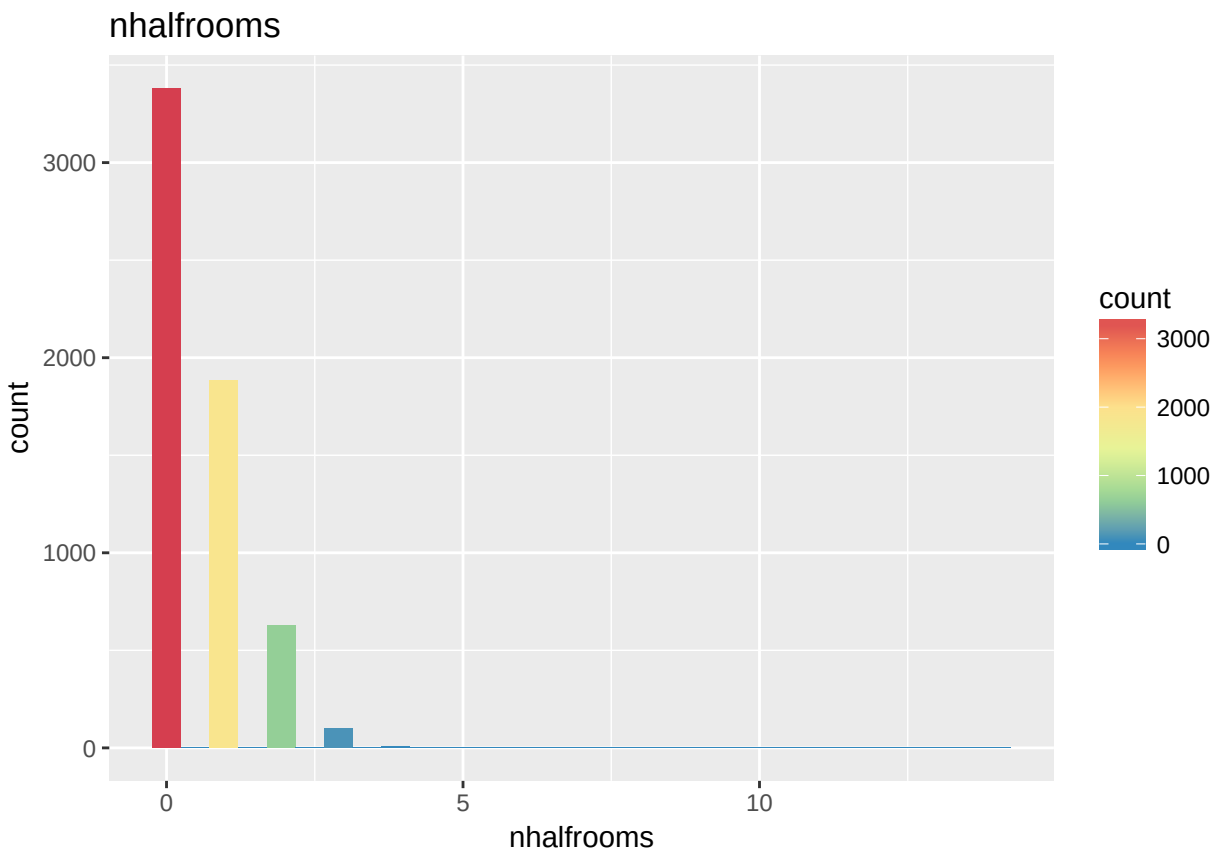
```
for (i in 1:length(property_sum)) {
    print(ggplot(property_sum, aes_string(x = colnames(property_sum)[i])) +
        geom_histogram(aes(fill = ..count..)) + ggtitle(colnames(property_sum)[i]) +
        scale_fill_distiller(palette = "Spectral"))
}
```

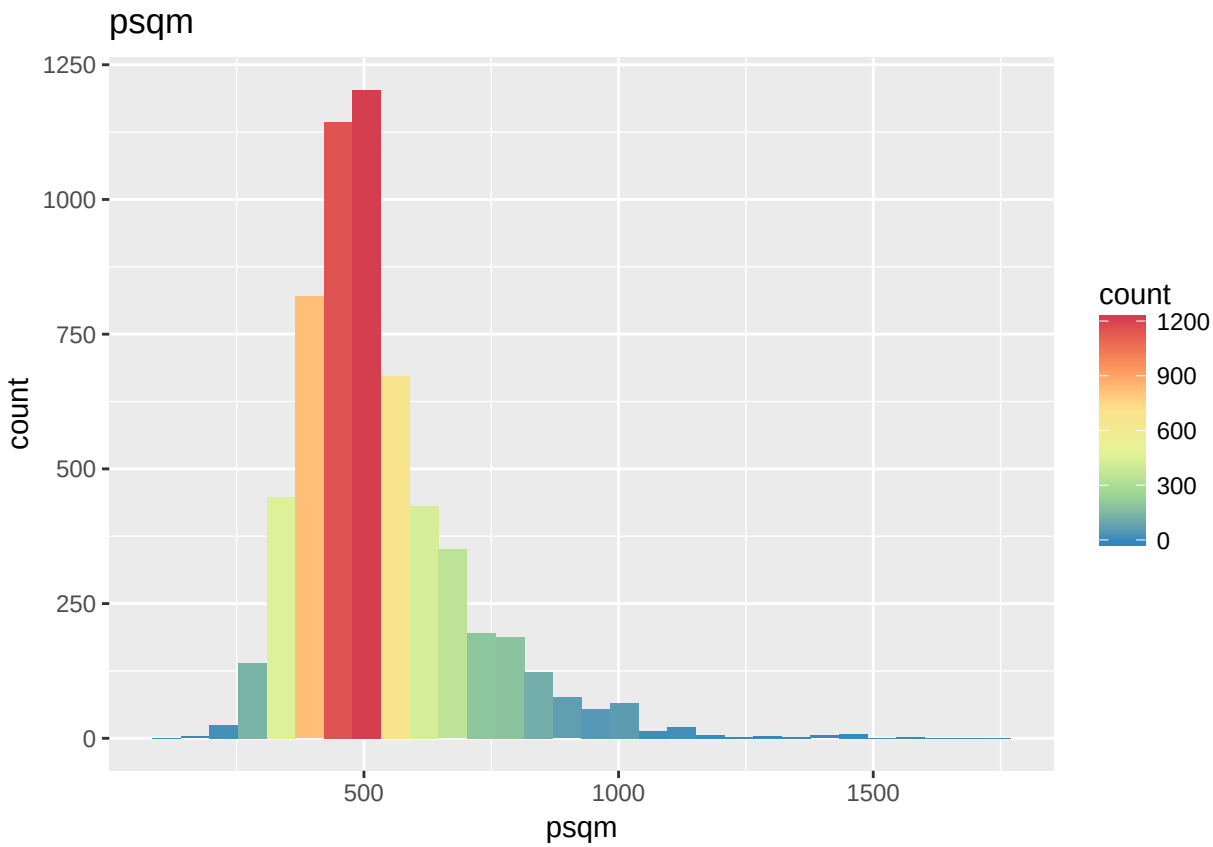

floor
1000
500
0
count
0
3
6
9
floor
count
1250
1000
750
500
250
0

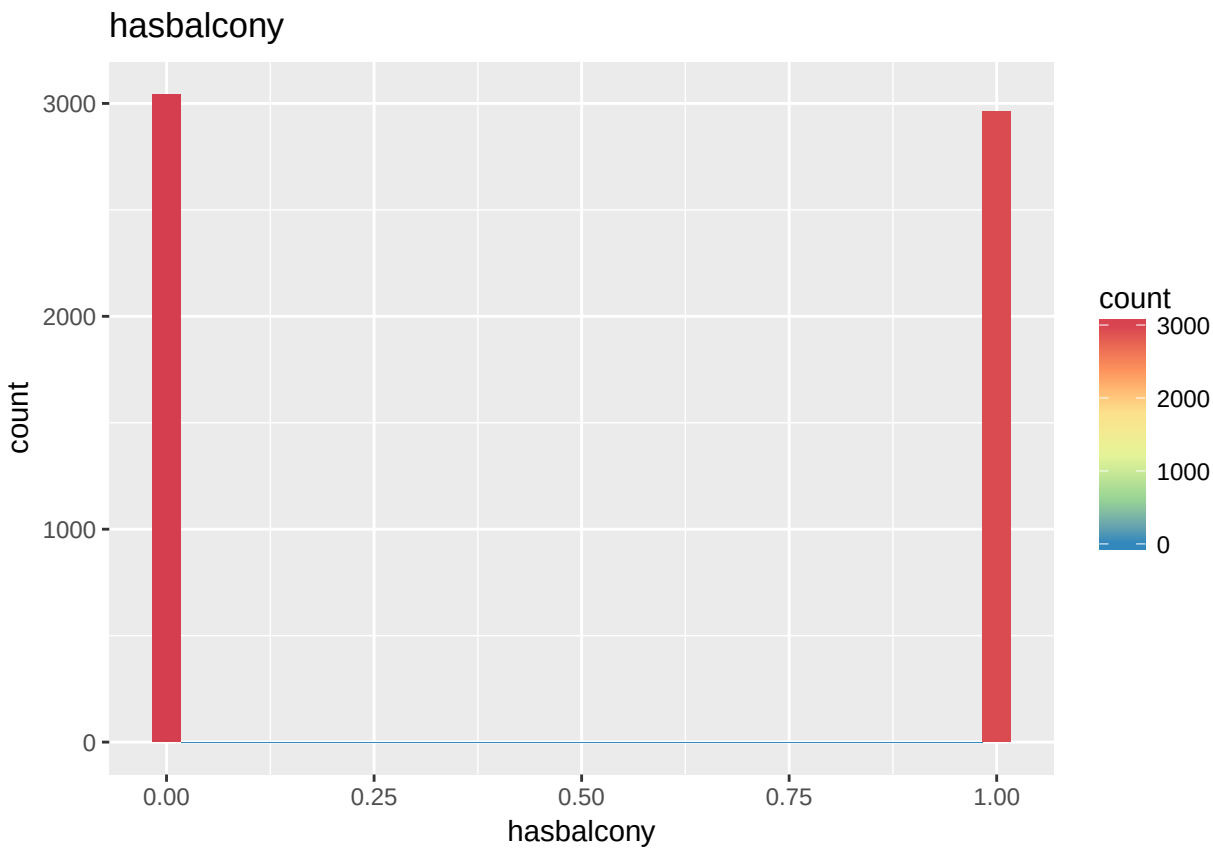

balcony
count
3000
2000
1000
0
0
25
50
75
100
balcony
count
3000
2000
1000

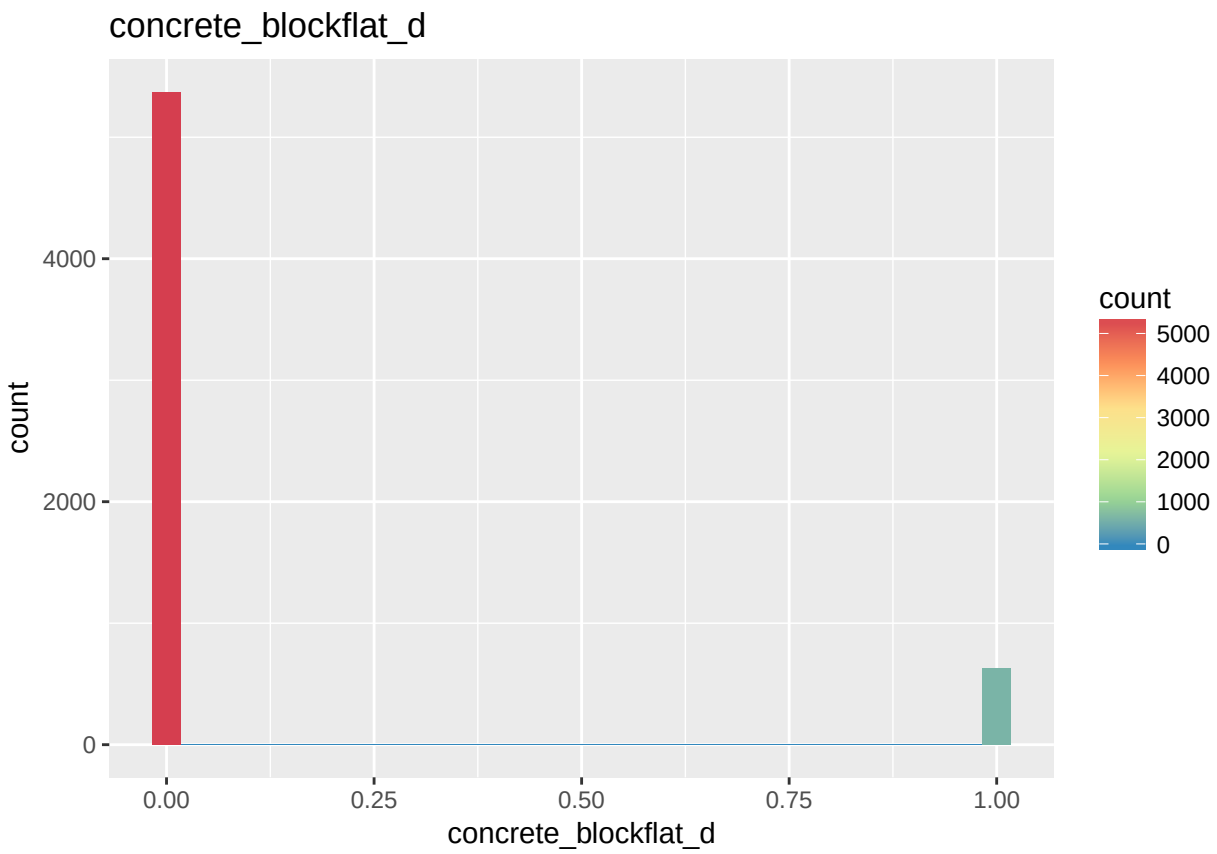

p
2500
2000
1500
1000
500
0
count
0e+00
1e+05
2e+05
3e+05
4e+05
p
count
2000
1500
1000
500
0

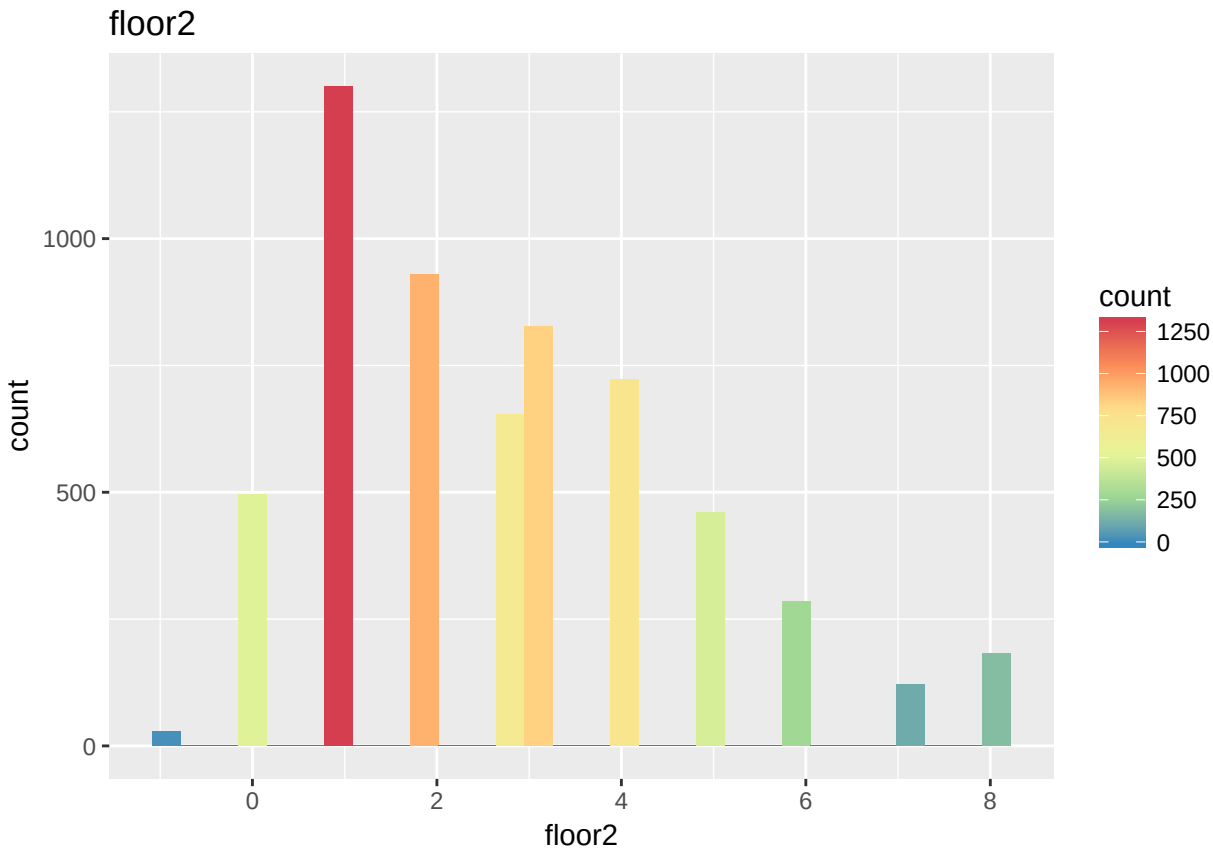

nrooms
count
2000
1500
1000
500
0
2.5
5.0
7.5
10.0
nrooms
count
2000
1500
1000
500
0

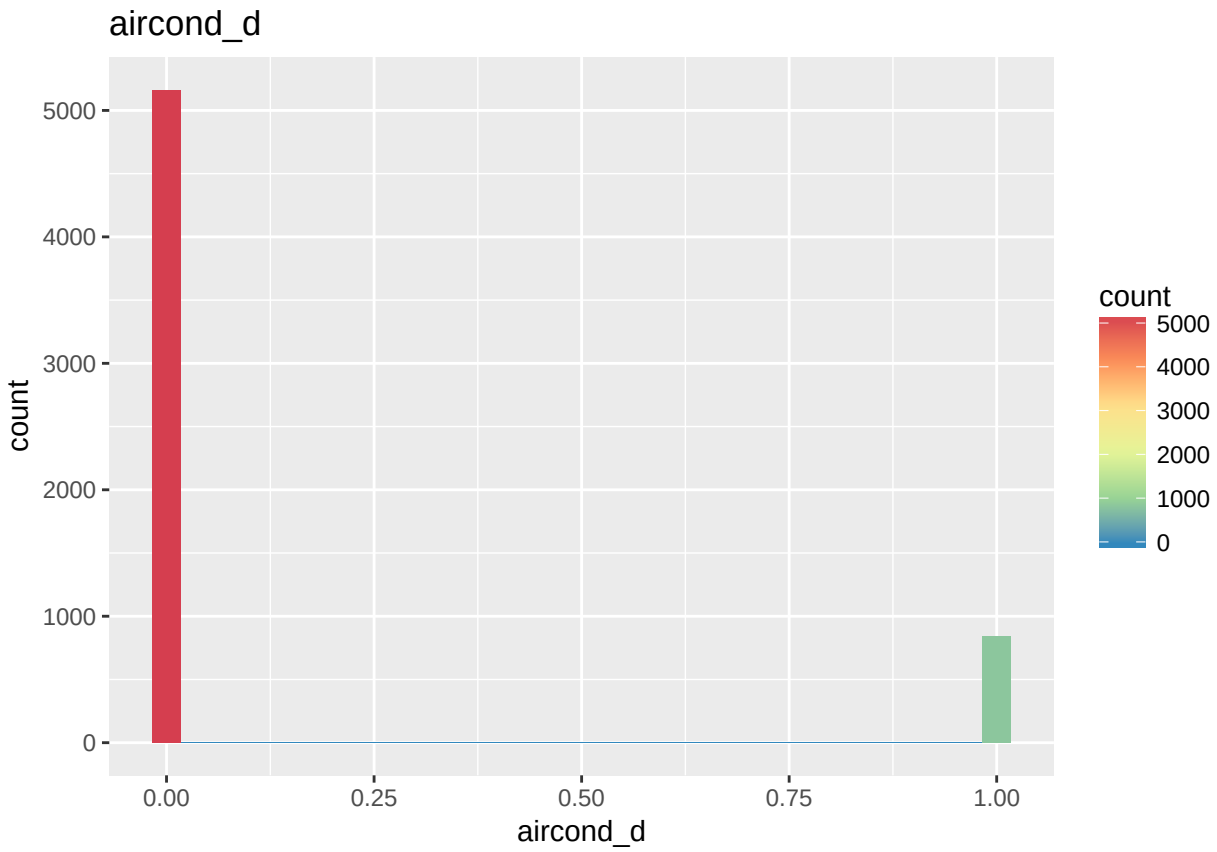

nhalfrooms
count
3000
2000
1000
0
0
5
10
nhalfrooms
count
3000
2000
1000
0

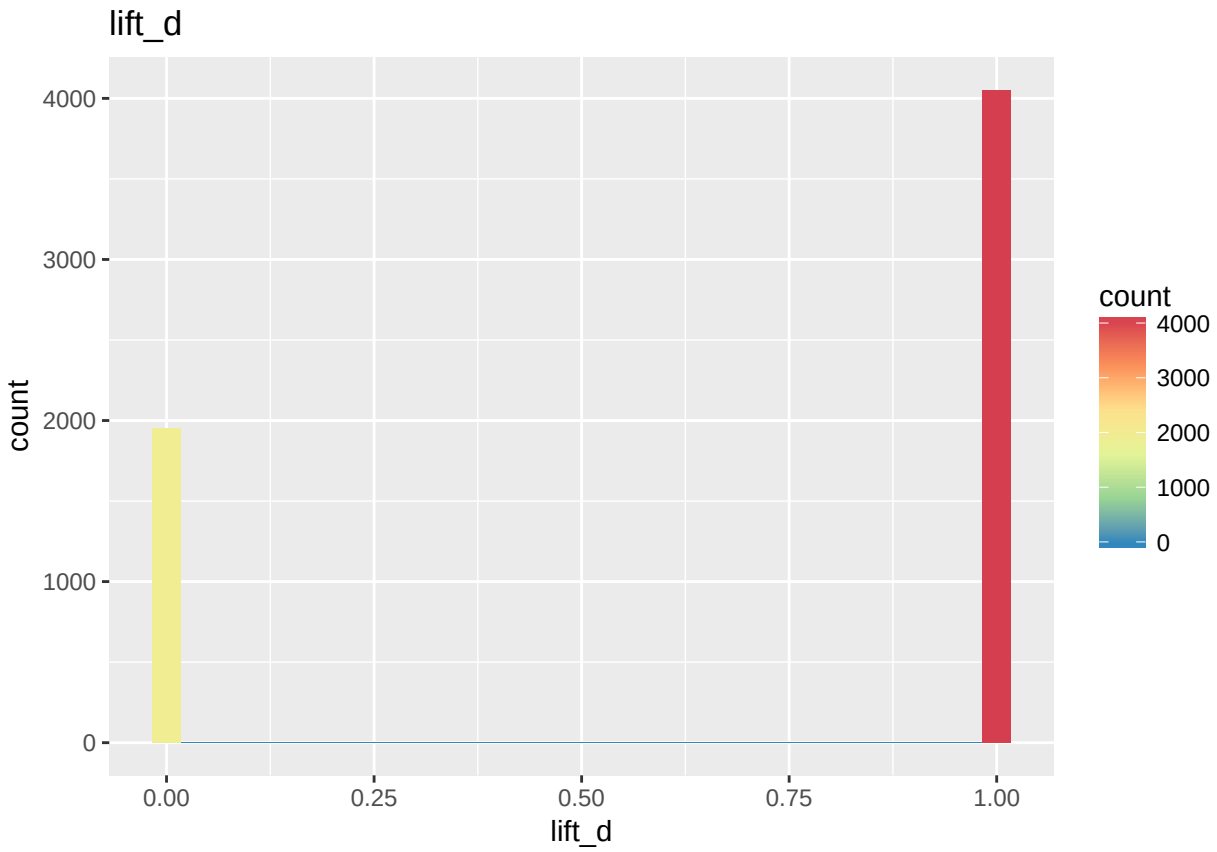

psqm
1250
1000
750
500
250
0
count
500
1000
1500
psqm
count
1200
900
600
300
0

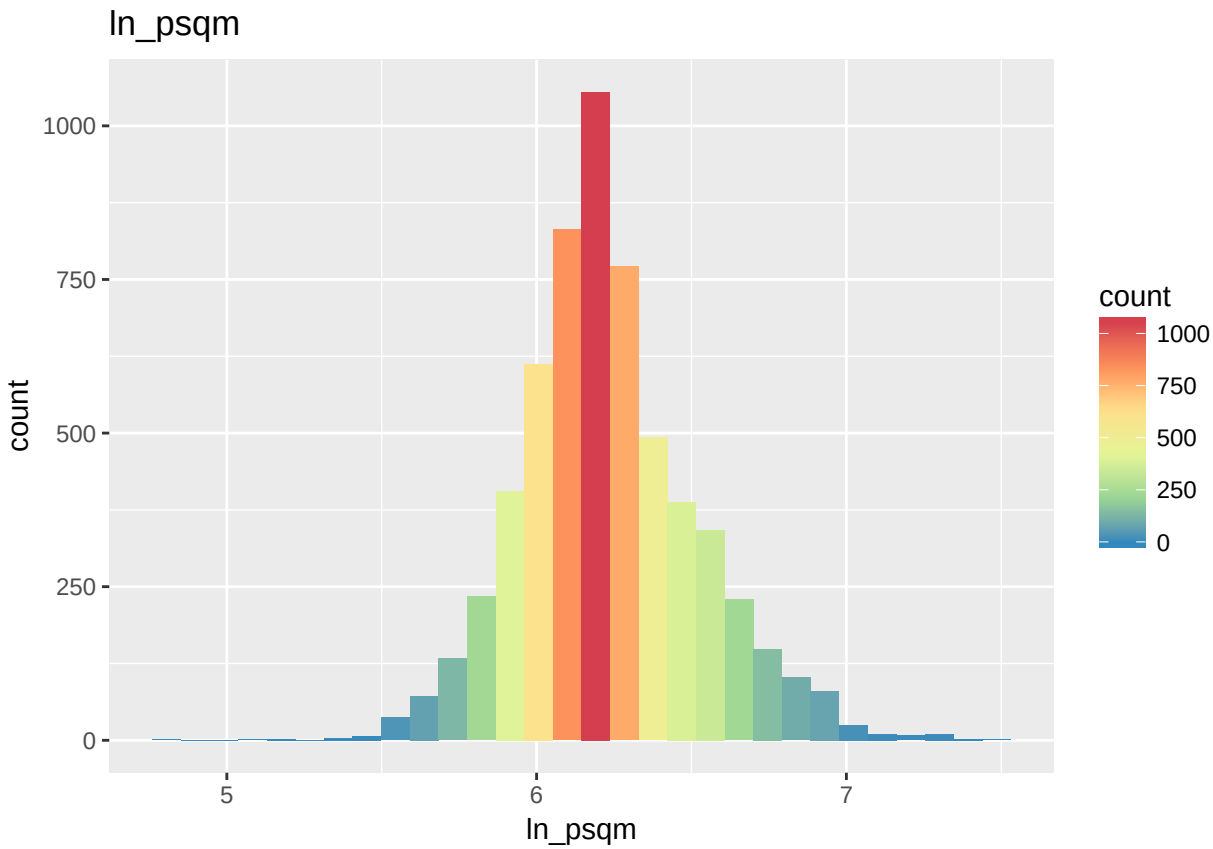

hasbalcony
3000
2000
1000
0
count
0.00
0.25
0.50
0.75
1.00
hasbalcony
count
3000
2000
1000
0

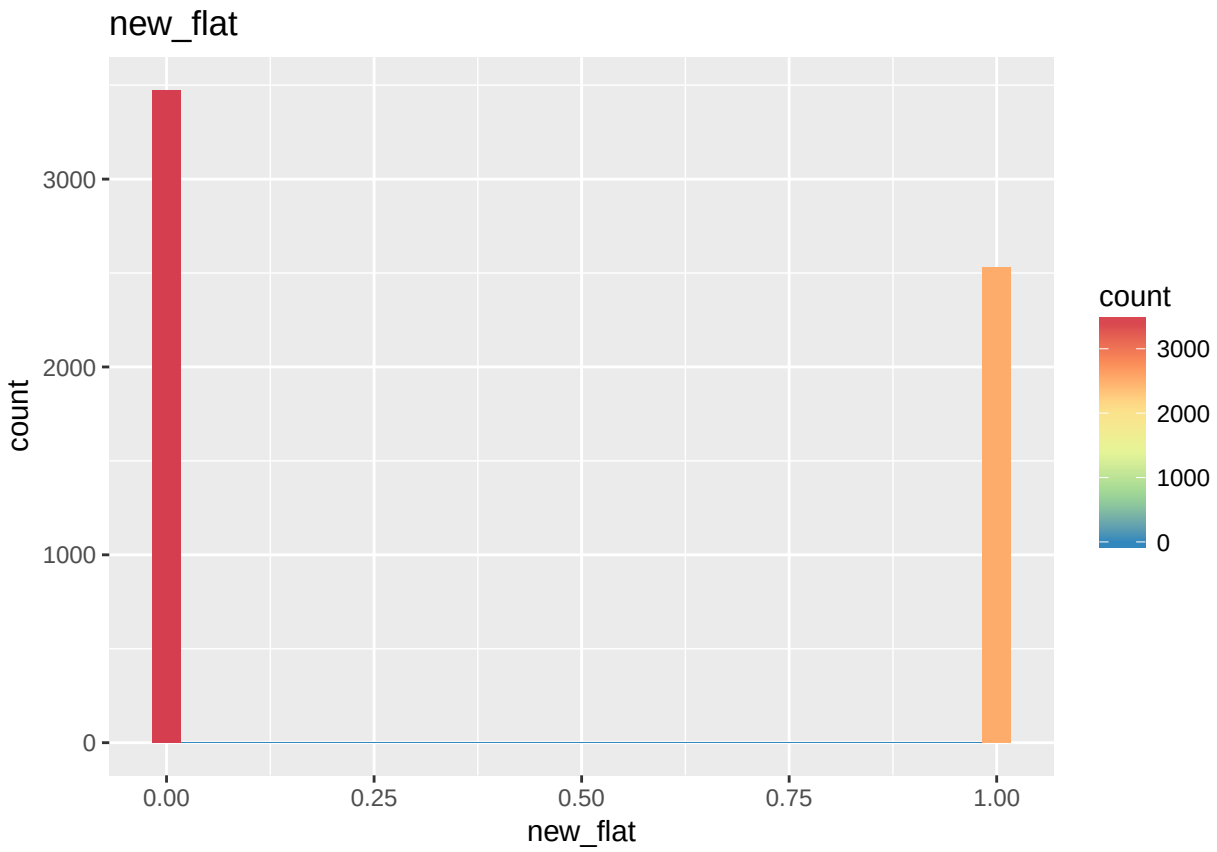

concrete_blockflat_d
count
4000
2000
0
0.00
0.25
0.50
0.75
1.00
concrete_blockflat_d
count
5000
4000
3000
2000
1000
0


floor2
count
1000
500
0
0
2
4
6
8
floor2
count
1250
1000
750
500
250
0


aircond_d
count
5000
4000
3000
2000
1000
0
0.00
0.25
0.50
0.75
1.00
aircond_d
count
5000
4000
3000
2000
1000
0


lift_d
4000
3000
2000
1000
0
count
0.00
0.25
0.50
0.75
1.00
lift_d
count
4000
3000
2000
1000
0


ln_psqm
count
1000
750
500
250
0
5
6
7
ln_psqm
count
1000
750
500
250
0




```
# save a graph of regression line of sqm on price per sqm, scatter plot
png(filename = "Graph_sc1.png", res = 200, width = 1200, height = 800)
ggplot(data = property_df, aes(x = sqm, y = psqm)) + geom_point(size = 1.5,
    aes(col = psqm)) + geom_smooth(method = "lm", colour = "darkgrey", se = FALSE) +
    labs(x = "Size in square meter", y = "Price per square meter", title = "Linear relationship of size
    scale_color_distiller("Price per square meter", palette = "Spectral") +
    theme_bw()
dev.off()

# save a graph of regression line of log sqm on log price per sqm, scatter
# plot
png(filename = "Graph_sc2.png", res = 200, width = 1200, height = 800)
ggplot(data = property_df, aes(x = ln_sqm, y = ln_psqm)) + geom_point(size = 1.5,
    aes(col = psqm)) + geom_smooth(method = "lm", colour = "darkgrey", se = FALSE) +
    labs(x = "Log size in square meter", y = "Log price per square meter", title = "Linear relationship
    scale_color_distiller("Price per square meter", palette = "Spectral") +
    theme_bw()
dev.off()

# save plot to file: LOESS log price per square meters on log square meters
png(filename = "Graph_lp1.png", res = 200, width = 1200, height = 800)
ggplot(data = property_df, aes(x = ln_sqm, y = ln_psqm)) + geom_smooth(method = "loess") +
    labs(x = "Log size in square meter", y = "Log price per square meter", title = "Non-linear relations
    scale_color_distiller("Price per square meter", palette = "Spectral") +
    theme_bw()
dev.off()
```

```r
# save plot to file: LOESS level price per square meters on level square
# meters
png(filename = "Graph_lp2.png", res = 200, width = 1200, height = 800)
ggplot(data = property_df, aes(x = sqm, y = psqm)) + geom_smooth(method = "loess") +
    labs(x = "Size in square meter", y = "Price per square meter", title = "Non-linear relationship of s
    scale_color_distiller("Price per square meter", palette = "Spectral") +
    theme_bw()
dev.off()

# save plot to file: LOESS log price per square meters on log square meters
# for apartmans below 180 sqm
png(filename = "Graph_lp3.png", res = 200, width = 1200, height = 800)
ggplot(data = property_df[property_df$sqm < 180, ], aes(x = ln_sqm, y = ln_psqm)) +
    geom_smooth(method = "loess")
dev.off()

# save plot to file: LOESS level price per square meters on level square
# meters for apartmans below 180 sqm
png(filename = "Graph_lp4.png", res = 200, width = 1200, height = 800)
ggplot(data = property_df[property_df$sqm < 180, ], aes(x = sqm, y = psqm)) +
    geom_smooth(method = "loess")
dev.off()

property_df <- property_df[property_df$sqm <= 180, ]
```

Section 3: Features

Size in square meter

Based on the loess regression of Price per square meter on Size in square meter, until approx 60 m^2 the average relationship is negative, while above the knot the average relationship is positive and has a large confidence interval. Extreme high prices per square meter can be found for basically any size.

- The 32 observations above 180 m^2 were dropped.
- sqm_sp2060, a new variable was created to signal observations with sqm larger than 60 m^2. Also log form lnsqm_sp2060 was created.
- Log form of sqm variable was created with further variations.

Floor variable

Due to non-integer values, floor variable was transformed and decoded into floor2 variable. Flag variable was created for values larger than 5 and at floor 0.

Suter variable

Only observations with suter = 0 and no missing value were kept.

Condition and Heating variables

- Both variables are factors with 7 and 13 categories respectively and a high proportion of missing values (7.5% and 10.9%).

- The factor levels were decoded for both in two alternative ways to create cleaner categories: condition with 8 levels, condition_broad with 6 levels, heating with 14 levels and heating_broad with 6 levels were created.
- NA values were decoded into others for heating to avoid dropping observations automatically.

```
# recoding multiple floor variables into integer values
property_df$floor[property_df$floor > 2 & property_df$floor < 3] <- 3
property_df$floor2[property_df$floor2 > 2 & property_df$floor2 < 3] <- 3
property_df$number_of_floor[property_df$number_of_floor > 5 & property_df$number_of_floor <
    6] <- 5

# creating new variable: sqm_sp2060, for flats larger than 60m2, value is
# set as flat 60 creating new variable: sqm_sp60p, which signals flats
# larger than 60m2 (0 value)
property_df$sqm_sp2060 <- property_df$sqm
property_df$sqm_sp2060[property_df$sqm_sp2060 > 60] <- 60
property_df$sqm_sp60p <- property_df$sqm - property_df$sqm_sp2060
count(property_df$sqm_sp60p)

# creating new variable: lnsqm_sp2060, for flats larger than 60m2, value is
# set as 4.1 creating new variable: lnsqm_sp60p, which signals flats larger
# than 60m2 (0 value)
property_df$lnsqm_sp2060 <- property_df$ln_sqm
property_df$lnsqm_sp2060[property_df$lnsqm_sp2060 > 4.1] <- 4.1
property_df$lnsqm_sp60p <- property_df$ln_sqm - property_df$lnsqm_sp2060

# defining new variable, which is a categorical for sqm with differences of
# 6 (19-25) and 5 for the rest
property_df$sqmcut <- cut(property_df$sqm, breaks = c(19, 25, 30, 35, 40, 45,
    50, 55, 60, 65, 70, 75, 80, 85, 90, 95, 100, 105, 110, 115, 120, 125, 130,
    140, 150, 160, 170, 181), right = FALSE)

# decoding variable: multiplying it by 10 defining new variable, which is a
# chategorical for ln_sqm10 with differences of 2 (29-31) and 1 for the rest
property_df$ln_sqm10 <- property_df$ln_sqm * 10
property_df$lnsqmcut <- cut(property_df$ln_sqm10, breaks = c(29, 31, 32, 33,
    34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51,
    52), right = FALSE)

# aggregate multiple values by the frequency in floor2, getting back the
# rounded mean of psqm
ddply(property_df, .(floor2), summarize, freq = length(floor2), mean_psqm = round(mean(psqm),
    digits = 3))

# taking a subset by data.table, where suter = 0 and there are no missing
# values
property_df <- property_df[which(property_df$suter == 0 | is.na(property_df$suter)),
    ]

# decoding floor2 with 3-s creating new variable: floor_sp05 out of floor2,
# where values larger than 5 are 5 creating a flag for values larger than 5
# by taking the difference
property_df$floor2[is.na(property_df$floor2)] <- 3
property_df$floor_sp05 <- property_df$floor2
```

```
property_df$floor_sp05[property_df$floor_sp05 > 5] <- 5
property_df$floor_sp6p <- property_df$floor2 - property_df$floor_sp05

# creating a flag variable for flats at floor zero
property_df$floor0 <- as.numeric(property_df$floor == 0)

# aggregate multiple values by the frequency in condition, getting back the
# rounded mean of psqm
ddply(property_df, .(condition), summarize, freq = length(condition), mean_psqm = round(mean(psqm),
    digits = 3))

# decoding condition variable from factor into character to be able to
# recode into new categories, creating new variable: _broad for missing obs.
# adding NA decoding just opened and under construction into newly built
# decoding condition_broad into a factor with levels given in the list
property_df$condition_broad <- as.character(property_df$condition)
property_df$condition_broad[which(is.na(property_df$condition_broad))] <- "NA"
property_df$condition_broad[which(property_df$condition_broad == "just opened" |
    property_df$condition == "under construction")] <- "newly built"
property_df$condition_broad <- factor(property_df$condition_broad, levels = c("NA",
    "in good condition", "in medium condition", "newly built", "recently built",
    "renovated"))

# aggregate multiple values by the frequency in condition_broad, getting
# back the rounded mean of psqm
ddply(property_df, .(condition_broad), summarize, freq = length(condition_broad),
    mean_psqm = round(mean(psqm), digits = 3))

# decoding condition variable from factor into character to be able to
# recode into new categories for missing obs. adding NA decoding just opened
# and under construction into newly built decoding condition into a factor
# with levels given in the list - different from condition_broad
property_df$condition <- as.character(property_df$condition)
property_df$condition[which(is.na(property_df$condition))] <- "NA"
property_df$condition <- factor(property_df$condition, levels = c("NA", "in good condition",
    "in medium condition", "just opened", "newly built", "recently built", "renovated",
    "under construction"))

# aggregate multiple values by the frequency in heating, getting back the
# rounded mean of psqm
ddply(property_df, .(heating), summarize, freq = length(heating), mean_psqm = round(mean(psqm),
    digits = 3))

# decoding heating variable from factor into character to be able to recode
# into new categories, creating new variable _broad for missing obs. adding
# NA decoding fan coil, electric, NA, geothermal and stove into other
# decoding boiler values into gas convector, renaming district heating and
# central heating vaues decoding heating_broad into a factor with levels
# given in the list
property_df$heating_broad <- as.character(property_df$heating)
property_df$heating_broad[which(is.na(property_df$heating_broad))] <- "NA"
property_df$heating_broad[which(property_df$heating_broad == "fan coil " | property_df$heating_broad ==
    "electric" | property_df$heating_broad == "NA" | property_df$heating_broad ==
```

```
    "geothermal" | property_df$heating_broad == "stove")] <- "other"
property_df$heating_broad[which(property_df$heating_broad == "gas Héra boiler" |
    property_df$heating_broad == "gas boiler")] <- "gas convector heating"
property_df$heating_broad[which(property_df$heating_broad == "district heating with heat meter")] <- "di
property_df$heating_broad[which(property_df$heating_broad == "central heating with heat meter")] <- "cen
property_df$heating_broad <- factor(property_df$heating_broad)

# aggregate multiple values by the frequency in heating_broad, getting back
# the rounded mean of psqm
ddply(property_df, .(heating_broad), summarize, freq = length(heating_broad),
    mean_psqm = round(mean(psqm), digits = 3))

# decoding heating variable from factor into character to be able to recode
# into new categories for missing obs. adding NA decoding heating into a
# factor with levels given in the list, different from levels of
# heating_broad
property_df$heating <- as.character(property_df$heating)
property_df$heating[which(is.na(property_df$heating))] <- "NA"
property_df$heating <- factor(property_df$heating, levels = c("NA", "central heating",
    "central heating with heat meter", "district heating", "district heating with heat meter",
    "electric", "fan coil ", "gas boiler", "gas circo heating", "gas convector heating",
    "gas Héra boiler", "geothermal", "other", "stove"))

# aggregate multiple values by the frequency in view, getting back the
# rounded mean of psqm
ddply(property_df, .(view), summarize, freq = length(view), mean_psqm = round(mean(psqm),
    digits = 3))

# decoding view variable from factor into character to be able to recode
# into new categories for missing obs. adding NA decoding view into a factor
# with levels given in the list
property_df$view <- as.character(property_df$view)
property_df$view[which(is.na(property_df$view))] <- "NA"
property_df$view <- factor(property_df$view, levels = c("NA", "court view",
    "garden view", "panorama", "street view"))

# aggregate multiple values by the frequency in view, getting back the
# rounded mean of psqm
ddply(property_df, .(view), summarize, freq = length(view), mean_psqm = round(mean(psqm),
    digits = 3))
```

Section 4: Feature engineering new variables

View, Orientation

Missing values were decoded with NA as a category itself, stored into orient_new variable.

Parking

3078 missing values, which is 50% of observations., therefore missing values were decoded with NA as a category itself:

- Parking_1 variable was created with categories: garage for sale, garage included, street parking, outdoor parking.
- Parking_2 variable was created with categories : garage, outdoor.

Balcony

No missing values, originally a string variable with numbers, decoded into categories in two ways:

- Balcony_1 variable was created with categories: no balcony, french (0-1 m^2), small (1-5 m^2), medium (5-15 m^2), large (>15 m^2).
- Balcony_2 variable was created with categories : no balcony, normal (0-20 m^2), large (20-60 m^2), extra large (>60 m^2).

```
# VIEW already decoded

# ORIENTATION aggregate multiple values by the frequency in orientation,
# getting back the rounded mean of psqm
ddply(property_df, .(orientation), summarize, freq = length(orientation), mean_psqm = round(mean(psqm),
    digits = 3))

# for missing obs. adding NA
property_df$orient_new <- as.character(property_df$orientation)
property_df$orient_new[which(is.na(property_df$orient_new))] <- "NA"
property_df$orient_new <- factor(property_df$orient_new, levels = c("NA", "East",
    "North", "South", "West", "North-East", "North-West", "South-East", "South-West"))
ddply(property_df, .(orient_new), summarize, freq = length(orient_new), mean_psqm = round(mean(psqm),
    digits = 3))


# PARKING: creating two versions of a new variable aggregate multiple values
# by the frequency in parking, getting back the rounded mean of psqm
ddply(property_df, .(parking), summarize, freq = length(parking), mean_psqm = round(mean(psqm),
    digits = 3))

# decoding parking variable from factor into character to be able to recode
# into new categories creating parking_1 variable and categories: garage for
# sale, garage included, street parking, outdoor parking for missing obs.
# adding NA
property_df$parking_1 <- as.character(property_df$parking)
property_df$parking_1[which(is.na(property_df$parking_1))] <- "NA"
property_df$parking_1[which((property_df$parking_1 == "garage- for sale" | property_df$parking_1 ==
    "parking space in undergound garage- for sale"))] <- "garage for sale"
property_df$parking_1[which((property_df$parking_1 == "garage- included in the price" |
    property_df$parking_1 == "parking space in underground garage- included in the price"))] <- "garage
property_df$parking_1[which((property_df$parking_1 == "on street parking - for pay" |
    property_df$parking_1 == "on street parking -for free"))] <- "street parking"
property_df$parking_1[which((property_df$parking_1 == "outdoor parking spot- for sale" |
    property_df$parking_1 == "outdoor parking spot- included in the price"))] <- "outdoor parking"

ddply(property_df, .(parking_1), summarize, freq = length(parking_1), mean_psqm = round(mean(psqm),
    digits = 3))

# decoding parking variable from factor into character to be able to recode
# into new categories creating parking_2 variable and categories: garage,
# outdoor for missing obs. adding NA
```

```
property_df$parking_2 <- as.character(property_df$parking)
property_df$parking_2[which(is.na(property_df$parking_2))] <- "NA"
property_df$parking_2[which((property_df$parking_2 == "garage- for sale" | property_df$parking_2 ==
    "parking space in undergound garage- for sale" | property_df$parking_2 ==
    "garage- included in the price" | property_df$parking_2 == "parking space in underground garage- in
property_df$parking_2[which((property_df$parking_2 == "on street parking - for pay" |
    property_df$parking_2 == "on street parking -for free" | property_df$parking_2 ==
    "outdoor parking spot- for sale" | property_df$parking_2 == "outdoor parking spot- included in the

ddply(property_df, .(parking_2), summarize, freq = length(parking_2), mean_psqm = round(mean(psqm),
    digits = 3))


# BALCONY: creating two versions of a new variable aggregate multiple values
# by the frequency in balcony, getting back the rounded mean of psqm
ddply(property_df, .(balcony), summarize, freq = length(balcony), mean_psqm = round(mean(psqm),
    digits = 3))

# decoding balcony variable from factor into character to be able to recode
# into new categories creating balcony_1 variable and categories: no
# balcony, french, small, medium, large for missing obs. adding NA
property_df$balcony_0 <- as.numeric(paste(property_df$balcony))
summary(property_df$balcony_1)
property_df$balcony_1[which(is.na(property_df$balcony_0))] <- "NA"
property_df$balcony_1[property_df$balcony_0 == 0] <- "no balcony"
property_df$balcony_1[property_df$balcony_0 > 0 & property_df$balcony_0 <= 1] <- "french"
property_df$balcony_1[property_df$balcony_0 > 1 & property_df$balcony_0 <= 5] <- "small"
property_df$balcony_1[property_df$balcony_0 > 5 & property_df$balcony_0 <= 15] <- "medium"
property_df$balcony_1[property_df$balcony_0 > 15] <- "large"

ddply(property_df, .(balcony_1), summarize, freq = length(balcony_1), mean_psqm = round(mean(psqm),
    digits = 3))

# decoding balcony variable from factor into character to be able to recode
# into new categories creating balcony_1 variable and categories: for
# missing obs. adding NA
property_df$balcony_0 <- as.numeric(paste(property_df$balcony))
summary(property_df$balcony_2)
property_df$balcony_2[which(is.na(property_df$balcony_0))] <- "NA"
property_df$balcony_2[property_df$balcony_0 == 0] <- "no balcony"
property_df$balcony_2[property_df$balcony_0 > 0 & property_df$balcony_0 <= 20] <- "normal"
property_df$balcony_2[property_df$balcony_0 > 20 & property_df$balcony_0 <=
    60] <- "large"
property_df$balcony_2[property_df$balcony_0 > 60] <- "extra large"

ddply(property_df, .(balcony_2), summarize, freq = length(balcony_2), mean_psqm = round(mean(psqm),
    digits = 3))
```

Section 5: Create test and train samples

```
# create test and train samples (80% of observations in train sample)
smp_size <- floor(0.8 * nrow(property_df))
```

```
# it will make results reproducable
set.seed(201703)

# create IDs
train_ids <- sample(seq_len(nrow(property_df)), size = smp_size)

# subset our data on train and test sets
property_df$train <- 0
property_df$train[train_ids] <- 1
```

Section 6: Regressions

REGRESSION MODEL 1: TRAINED ON THE ENTIRE DATASET

```
# Create new function for RMSE calculation in levels (if RMSE from caret
# package doesn't work)
RMSE_Lev <- function(pred, obs, na.rm = FALSE) {
    sqrt(mean((obs - pred)^2, na.rm = na.rm))
}

# Create new function for RMSE calculation in logs (if RMSE from caret
# package doesn't work)
RMSE_Log <- function(pred, obs, na.rm = FALSE) {
    sqrt(mean((exp(obs) - exp(pred))^2, na.rm = na.rm))
}

# REGRESSION MODEL 1 Multiple regressions with no interactions trained on
# the entire dataset
reg_1 <- lm(data = property_df, psqm ~ sqm_sp2060 + sqm_sp60p + heating_broad +
    condition_broad + aircond_d + elevator + nrooms + nhalfrooms + hasbalcony +
    concrete_blockflat_d + floor2 + lift_d + new_flat + floor_sp05 + floor_sp6p)
reg_2 <- lm(data = property_df, ln_psqm ~ lnsqm_sp2060 + lnsqm_sp60p + heating_broad +
    condition_broad + aircond_d + elevator + nrooms + nhalfrooms + hasbalcony +
    concrete_blockflat_d + floor2 + lift_d + new_flat + floor_sp05 + floor_sp6p)

reg_3 <- lm(data = property_df, psqm ~ sqm_sp2060 + sqm_sp60p + heating_broad +
    condition_broad + aircond_d + elevator + nrooms + nhalfrooms + hasbalcony +
    concrete_blockflat_d + floor2 + lift_d + new_flat + floor_sp05 + floor_sp6p +
    view + orient_new + parking_1 + balcony_1)
reg_4 <- lm(data = property_df, ln_psqm ~ lnsqm_sp2060 + lnsqm_sp60p + heating_broad +
    condition_broad + aircond_d + elevator + nrooms + nhalfrooms + hasbalcony +
    concrete_blockflat_d + floor2 + lift_d + new_flat + floor_sp05 + floor_sp6p +
    view + orient_new + parking_1 + balcony_1)

reg_5 <- lm(data = property_df, psqm ~ sqm_sp2060 + sqm_sp60p + heating_broad +
    condition_broad + aircond_d + elevator + nrooms + nhalfrooms + hasbalcony +
    concrete_blockflat_d + floor2 + lift_d + new_flat + floor_sp05 + floor_sp6p +
    view + orient_new + parking_2 + balcony_2)
reg_6 <- lm(data = property_df, ln_psqm ~ lnsqm_sp2060 + lnsqm_sp60p + heating_broad +
    condition_broad + aircond_d + elevator + nrooms + nhalfrooms + hasbalcony +
    concrete_blockflat_d + floor2 + lift_d + new_flat + floor_sp05 + floor_sp6p +
```

```
    view + orient_new + parking_2 + balcony_2)

reg_models <- list(reg_1, reg_2, reg_3, reg_4, reg_5, reg_6)

# calculating RMSE with our predefined function calculating BIC
reg_1_rmse_test <- round(RMSE_Lev(pred = predict(reg_1, newdata = property_df[property_df$train ==
    0, ]), obs = property_df$psqm[property_df$train == 0], na.rm = TRUE), digits = 3)
reg_1_bic <- round(BIC(reg_1), digits = 3)

reg_2_rmse_test <- round(RMSE_Log(pred = predict(reg_2, newdata = property_df[property_df$train ==
    0, ]), obs = property_df$ln_psqm[property_df$train == 0], na.rm = TRUE),
    digits = 3)
reg_2_bic <- round(BIC(reg_2), digits = 3)

reg_3_rmse_test <- round(RMSE_Lev(pred = predict(reg_3, newdata = property_df[property_df$train ==
    0, ]), obs = property_df$psqm[property_df$train == 0], na.rm = TRUE), digits = 3)
reg_3_bic <- round(BIC(reg_3), digits = 3)

reg_4_rmse_test <- round(RMSE_Log(pred = predict(reg_4, newdata = property_df[property_df$train ==
    0, ]), obs = property_df$ln_psqm[property_df$train == 0], na.rm = TRUE),
    digits = 3)
reg_4_bic <- round(BIC(reg_4), digits = 3)

reg_5_rmse_test <- round(RMSE_Lev(pred = predict(reg_5, newdata = property_df[property_df$train ==
    0, ]), obs = property_df$psqm[property_df$train == 0], na.rm = TRUE), digits = 3)
reg_5_bic <- round(BIC(reg_5), digits = 3)

reg_6_rmse_test <- round(RMSE_Log(pred = predict(reg_6, newdata = property_df[property_df$train ==
    0, ]), obs = property_df$ln_psqm[property_df$train == 0], na.rm = TRUE),
    digits = 3)
reg_6_bic <- round(BIC(reg_6), digits = 3)
```

```
stargazer(
  title = "Multiple regression models trained on the entire dataset",
  type = "latex",
  column.labels = c("Reg 1", "Reg 2", "Reg 3", "Reg 4", "Reg 5", "Reg 6"),
  list(reg_models), digits = 2,
  model.numbers = FALSE,
  model.names = FALSE,
  omit.stat = c("adj.rsq", "f", "ser"),
  dep.var.caption = 'Dependent variable: Price per square meter',
  out = "IS.html",
  notes.align = "l",
  single.row = TRUE,
  header = FALSE,
  add.lines = list(
    c("New controls", "No", "No", "Yes", "Yes", "Yes", "Yes"),
    c("Scaling of new controls", "", "", "Complex", "Complex", "Simple", "Simple"),
    c("Dep. and expl. variables", "Levels", "Logs", "Levels", "Logs", "Levels", "Logs"),
    c("BIC", reg_1_bic, reg_2_bic, reg_3_bic, reg_4_bic, reg_5_bic, reg_6_bic),
    c("RMSE test", reg_1_rmse_test, reg_2_rmse_test, reg_3_rmse_test, reg_4_rmse_test, reg_5_rmse_test,
)
```

Reg_3 and Reg_4 provides best model performance (smallest RMSE), however log brings unnecessary model

complexity in Reg_4. For new variables complex scaling performs slightly better than simple.

REGRESSION MODEL 2: OUT-OF-SAMPLE

```
# Out of sample multiple regressions without interactions and
# cross-validation REG_3 and REG_4 provides best model performance (smallest
# RMSE), however log brings unnecessary model complexity For new variables:
# complex scaling performs slightly better

reg_7 <- lm(data = property_df[property_df$train == 1, ], psqm ~ sqm_sp2060 +
    sqm_sp60p + heating_broad + condition_broad + aircond_d + elevator + nrooms +
    nhalfrooms + hasbalcony + concrete_blockflat_d + floor2 + lift_d + new_flat +
    floor_sp05 + floor_sp6p + view + orient_new + parking_1 + balcony_1)

reg_7_rmse_train <- round(RMSE_Lev(pred = predict(reg_7, newdata = property_df[property_df$train ==
    1, ]), obs = property_df$psqm[property_df$train == 1], na.rm = TRUE), digits = 3)

reg_7_rmse_test <- round(RMSE_Lev(pred = predict(reg_7, newdata = property_df[property_df$train ==
    0, ]), obs = property_df$psqm[property_df$train == 0], na.rm = TRUE), digits = 3)

reg_7_bic <- round(BIC(reg_7), digits = 3)

# including all variables for each dimension using logs to provide the best
# prediction
reg_8 <- lm(data = property_df[property_df$train == 1, ], ln_psqm ~ lnsqm_sp2060 +
    lnsqm_sp60p + heating_broad + condition_broad + aircond_d + elevator + nrooms +
    nhalfrooms + hasbalcony + concrete_blockflat_d + floor2 + lift_d + new_flat +
    floor_sp05 + floor_sp6p + view + orient_new + parking_1 + balcony_1)

reg_8_rmse_train <- round(RMSE_Log(pred = predict(reg_8, newdata = property_df[property_df$train ==
    1, ]), obs = property_df$ln_psqm[property_df$train == 1], na.rm = TRUE),
    digits = 3)

reg_8_rmse_test <- round(RMSE_Log(pred = predict(reg_8, newdata = property_df[property_df$train ==
    0, ]), obs = property_df$ln_psqm[property_df$train == 0], na.rm = TRUE),
    digits = 3)

reg_8_bic <- round(BIC(reg_8), digits = 3)

reg_3_rmse_train <- round(RMSE_Lev(pred = predict(reg_3, newdata = property_df[property_df$train ==
    1, ]), obs = property_df$psqm[property_df$train == 1], na.rm = TRUE), digits = 3)
```

```
stargazer(
  title = "Levels and Logs Out-of-sample models",
  list(reg_3, reg_7, reg_8), digits = 2,
  model.numbers = FALSE,
  column.labels = c("Reg 3", "Reg 7", "Reg 8"),
  omit.stat = c("adj.rsq", "f", "ser"),
  dep.var.caption = 'Dependent variable: Price per square meter',
  out = "OS.html",
  notes.align = "l",
  single.row = TRUE,
  header = FALSE,
```

```
  type ='latex',
  add.lines = list(
    c("Dataset", "IS", "OS", "OS"),
    c("Scaling of new controls", "Complex", "Complex", "Complex"),
    c("Dep. and expl. variables", "Levels", "Levels", "Logs"),
    c("BIC", reg_3_bic, reg_7_bic, reg_8_bic),
    c("RMSE train", reg_3_rmse_train, reg_7_rmse_train, reg_8_rmse_train),
    c("RMSE test", reg_3_rmse_test, reg_7_rmse_test, reg_8_rmse_test))
)
```

Reg_7 provides better model performance (smallest RMSE), while log brings unnecessary model complexity in Reg_8. BIC values are not comparable across Level and Log.

REGRESSION MODEL 3: INTERACTIONS

```
# Out of sample multiple regressions with interactions and without
# cross-validation REG_3 provides best model performance (smallest RMSE)
# without unnecessary model complexity REG_9: Levels, interactions with 1)
# nrooms, 2) new_flat, 3) concrete_blockflat_d only those variables are
# interacted per each which result in a statistically significant
# coefficient

reg_9 <- lm(data = property_df[property_df$train == 1, ], psqm ~ sqm_sp2060 +
    sqm_sp60p + heating_broad + condition_broad + aircond_d + elevator + nrooms +
    nhalfrooms + hasbalcony + concrete_blockflat_d + floor2 + lift_d + new_flat +
    floor_sp05 + floor_sp6p + view + orient_new + parking_1 + balcony_1 + sqm_sp60p *
    nrooms + heating_broad * nrooms + elevator * nrooms + concrete_blockflat_d *
    nrooms + floor2 * nrooms + floor_sp05 * nrooms + view * nrooms + orient_new *
    nrooms + heating_broad * new_flat + elevator * new_flat + hasbalcony * new_flat +
    view * new_flat + orient_new * new_flat + parking_1 * new_flat + aircond_d *
    concrete_blockflat_d + elevator * concrete_blockflat_d + floor2 * concrete_blockflat_d +
    parking_1 * concrete_blockflat_d)

reg_9_rmse_train <- round(RMSE_Lev(pred = predict(reg_9, newdata = property_df[property_df$train ==
    1, ]), obs = property_df$psqm[property_df$train == 1], na.rm = TRUE), digits = 3)

reg_9_rmse_test <- round(RMSE_Lev(pred = predict(reg_9, newdata = property_df[property_df$train ==
    0, ]), obs = property_df$psqm[property_df$train == 0], na.rm = TRUE), digits = 3)

reg_9_bic <- round(BIC(reg_9), digits = 3)

# REG_10: Logs, interactions with condition_broad for the rest of the
# variables
reg_10 <- lm(data = property_df[property_df$train == 1, ], ln_psqm ~ lnsqm_sp2060 +
    lnsqm_sp60p + heating_broad + condition_broad + aircond_d + elevator + nrooms +
    nhalfrooms + hasbalcony + concrete_blockflat_d + floor2 + lift_d + new_flat +
    floor_sp05 + floor_sp6p + view + orient_new + parking_1 + balcony_1 + lnsqm_sp60p *
    nrooms + heating_broad * nrooms + elevator * nrooms + concrete_blockflat_d *
    nrooms + floor2 * nrooms + floor_sp05 * nrooms + view * nrooms + orient_new *
    nrooms + heating_broad * new_flat + elevator * new_flat + hasbalcony * new_flat +
    view * new_flat + orient_new * new_flat + parking_1 * new_flat + aircond_d *
    concrete_blockflat_d + elevator * concrete_blockflat_d + floor2 * concrete_blockflat_d +
```

```
    parking_1 * concrete_blockflat_d + lnsqm_sp2060 * condition_broad + nhalfrooms *
    condition_broad + hasbalcony * condition_broad + lift_d * condition_broad +
    floor_sp6p * condition_broad + balcony_1 * condition_broad)

reg_10_rmse_train <- round(RMSE_Log(pred = predict(reg_10, newdata = property_df[property_df$train ==
    1, ]), obs = property_df$ln_psqm[property_df$train == 1], na.rm = TRUE),
    digits = 3)

reg_10_rmse_test <- round(RMSE_Log(pred = predict(reg_10, newdata = property_df[property_df$train ==
    0, ]), obs = property_df$ln_psqm[property_df$train == 0], na.rm = TRUE),
    digits = 3)

reg_10_bic <- round(BIC(reg_10), digits = 3)

# calculating the number of coefficents
reg_3_n <- length(which(is.na(coefficients(reg_3)) == FALSE)) - 1
reg_7_n <- length(which(is.na(coefficients(reg_7)) == FALSE)) - 1
reg_8_n <- length(which(is.na(coefficients(reg_8)) == FALSE)) - 1
reg_9_n <- length(which(is.na(coefficients(reg_9)) == FALSE)) - 1
reg_10_n <- length(which(is.na(coefficients(reg_10)) == FALSE)) - 1
```

```
stargazer(
  title = "Levels and Logs Out-of-sample models with interactions",
  type ='latex',
  list(reg_3, reg_7, reg_8, reg_9, reg_10), digits = 2,
  model.numbers = FALSE,
  column.labels = c("Reg 3", "Reg 7", "Reg 8", "Reg 9", "Reg 10"),
  omit.stat = c("adj.rsq", "f", "ser"),
  dep.var.caption = 'Dependent variable: Price per square meter',
  out = "OS_int.html",
  notes.align = "l",
  single.row = TRUE,
  header = FALSE,
  add.lines = list(
    c("Dataset", "IS", "OS", "OS", "OS", "OS"),
    c("Scaling of new controls", "Complex", "Complex", "Complex", "Complex", "Complex"),
    c("Dep. and expl. variables", "Levels", "Levels", "Logs", "Levels", "Logs"),
    c("BIC", reg_3_bic, reg_7_bic, reg_8_bic, reg_9_bic, reg_10_bic),
    c("RMSE train", reg_3_rmse_train, reg_7_rmse_train, reg_8_rmse_train, reg_9_rmse_train, reg_10_rmse
    c("RMSE test", reg_3_rmse_test, reg_7_rmse_test, reg_8_rmse_test, reg_9_rmse_test, reg_10_rmse_test
    c("Number of controls", reg_3_n, reg_7_n, reg_8_n, reg_9_n, reg_10_n))
)
```

Reg_9 provides better model performance (smaller RMSE) but has a larger BIC. This means that neither Logs do not improve our prediction a lot, nor the additional interactions with conditions_broad make our prediction better.

REGRESSION MODEL 4: CROSS-VALIDATION

```
# Out of sample multiple regressions with interactions and with
# cross-validation Using reg_7, reg_9 and reg_10 (best performers, all in
# levels)
```

```r
# Setup 5-fold cross-validation
set.seed(201703)
k <- 5

# Assign fold number randomly to rows
folds <- sample(rep(1:k, nrow(property_df)/k))
folds
table(folds)

# Create empty vectors for RMSEs and BICs
cv_reg_7_rmse_train <- NULL
cv_reg_9_rmse_train <- NULL
cv_reg_10_rmse_train <- NULL
cv_reg_7_rmse_test <- NULL
cv_reg_9_rmse_test <- NULL
cv_reg_10_rmse_test <- NULL
cv_reg_7_bic <- NULL
cv_reg_9_bic <- NULL
cv_reg_10_bic <- NULL

# Run the loop k times (the number of folds), k = 5
for (i in 1:k) {
    # Creating test and train sets selecting the i-th fold as test set
    cv_property_df_train <- property_df[rep(folds != i, length = .N)]
    cv_property_df_test <- property_df[rep(folds == i, length = .N)]

    cv_reg_7 <- lm(data = cv_property_df_train, psqm ~ sqm_sp2060 + sqm_sp60p +
        heating_broad + condition_broad + aircond_d + elevator + nrooms + nhalfrooms +
        hasbalcony + concrete_blockflat_d + floor2 + lift_d + new_flat + floor_sp05 +
        floor_sp6p + view + orient_new + parking_1 + balcony_1)

    cv_reg_7_rmse_train[i] <- round(RMSE_Lev(pred = predict(cv_reg_7, newdata = cv_property_df_train),
        obs = cv_property_df_train$psqm, na.rm = TRUE), digits = 3)

    cv_reg_7_rmse_test[i] <- round(RMSE_Lev(pred = predict(cv_reg_7, newdata = cv_property_df_test),
        obs = cv_property_df_test$psqm, na.rm = TRUE), digits = 3)

    cv_reg_7_bic <- round(BIC(cv_reg_7), digits = 3)


    cv_reg_9 <- lm(data = cv_property_df_train, psqm ~ sqm_sp2060 + sqm_sp60p +
        heating_broad + condition_broad + aircond_d + elevator + nrooms + nhalfrooms +
        hasbalcony + concrete_blockflat_d + floor2 + lift_d + new_flat + floor_sp05 +
        floor_sp6p + view + orient_new + parking_1 + balcony_1 + sqm_sp60p *
        nrooms + heating_broad * nrooms + elevator * nrooms + concrete_blockflat_d *
        nrooms + floor2 * nrooms + floor_sp05 * nrooms + view * nrooms + orient_new *
        nrooms + heating_broad * new_flat + elevator * new_flat + hasbalcony *
        new_flat + view * new_flat + orient_new * new_flat + parking_1 * new_flat +
        aircond_d * concrete_blockflat_d + elevator * concrete_blockflat_d +
        floor2 * concrete_blockflat_d + parking_1 * concrete_blockflat_d)

    cv_reg_9_rmse_train[i] <- round(RMSE_Lev(pred = predict(cv_reg_9, newdata = cv_property_df_train),
        obs = cv_property_df_train$psqm, na.rm = TRUE), digits = 3)
```

```
    cv_reg_9_rmse_test[i] <- round(RMSE_Lev(pred = predict(cv_reg_9, newdata = cv_property_df_test),
        obs = cv_property_df_test$psqm, na.rm = TRUE), digits = 3)

    cv_reg_9_bic <- round(BIC(cv_reg_9), digits = 3)


    cv_reg_10 <- lm(data = cv_property_df_train, psqm ~ sqm_sp2060 + sqm_sp60p +
        heating_broad + condition_broad + aircond_d + elevator + nrooms + nhalfrooms +
        hasbalcony + concrete_blockflat_d + floor2 + lift_d + new_flat + floor_sp05 +
        floor_sp6p + view + orient_new + parking_1 + balcony_1 + sqm_sp60p *
        nrooms + heating_broad * nrooms + elevator * nrooms + concrete_blockflat_d *
        nrooms + floor2 * nrooms + floor_sp05 * nrooms + view * nrooms + orient_new *
        nrooms + heating_broad * new_flat + elevator * new_flat + hasbalcony *
        new_flat + view * new_flat + orient_new * new_flat + parking_1 * new_flat +
        aircond_d * concrete_blockflat_d + elevator * concrete_blockflat_d +
        floor2 * concrete_blockflat_d + parking_1 * concrete_blockflat_d + lnsqm_sp2060 *
        condition_broad + nhalfrooms * condition_broad + hasbalcony * condition_broad +
        lift_d * condition_broad + floor_sp6p * condition_broad + balcony_1 *
        condition_broad)

    cv_reg_10_rmse_train[i] <- round(RMSE_Lev(pred = predict(cv_reg_10, newdata = cv_property_df_train)
        obs = cv_property_df_train$psqm, na.rm = TRUE), digits = 3)

    cv_reg_10_rmse_test[i] <- round(RMSE_Lev(pred = predict(cv_reg_10, newdata = cv_property_df_test),
        obs = cv_property_df_test$psqm, na.rm = TRUE), digits = 3)

    cv_reg_10_bic <- round(BIC(cv_reg_10), digits = 3)

}
```

```
stargazer(
  title = "Out-of-sample models with interactions and cross-validation",
  list(reg_7, cv_reg_7, reg_9, cv_reg_9, reg_10, cv_reg_10), digits = 2,
  model.numbers = FALSE,
  column.labels = c("Reg 7", "Reg 7 CV", "Reg 9", "Reg 9 CV","Reg 10", "Reg 10 CV"),
  omit.stat = c("adj.rsq", "f", "ser"),
  dep.var.caption = 'Dependent variable: Price per square meter',
  out = "CV.html",
  notes.align = "l",
  single.row = TRUE,
  header = FALSE,
  type ='latex',
  add.lines = list(
    c("Dataset", "OS", "CV", "OS", "CV", "OS", "CV"),
    c("BIC", reg_7_bic, cv_reg_7_bic, reg_9_bic, cv_reg_9_bic, reg_10_bic, cv_reg_10_bic),
    c("RMSE train", reg_7_rmse_train, cv_reg_7_rmse_train, reg_9_rmse_train, cv_reg_9_rmse_train, reg_1
    c("RMSE test", reg_7_rmse_test, cv_reg_7_rmse_test, reg_9_rmse_test, cv_reg_9_rmse_test, reg_10_rms
)
```

```
# Model statistics summary
models_rmse <- data.frame(matrix(nrow = 5, ncol = 3))
colnames(models_rmse) <- c("Model 7", "Model 9", "Model 10")
rownames(models_rmse) <- c("BIC", "RMSE train", "RMSE test", "RMSE min", "RMSE max")
```

```
# Calculating RMSE and BIC from all 3 models
models_rmse[, 1] <- c(mean(cv_reg_7_bic), mean(cv_reg_7_rmse_train), mean(cv_reg_7_rmse_test),
    min(cv_reg_7_rmse_test), max(cv_reg_7_rmse_test))
models_rmse[, 2] <- c(mean(cv_reg_9_bic), mean(cv_reg_9_rmse_train), mean(cv_reg_9_rmse_test),
    min(cv_reg_9_rmse_test), max(cv_reg_9_rmse_test))
models_rmse[, 3] <- c(mean(cv_reg_10_bic), mean(cv_reg_10_rmse_train), mean(cv_reg_10_rmse_test),
    min(cv_reg_10_rmse_test), max(cv_reg_10_rmse_test))

# Display table
models_rmse
```

Cross-validated datasets provided lower RMSE and larger BIC in case of Reg 10, while for Reg 7 and Reg 9 RMSE from the model run on the original train and test set performed better. Overall, Reg 9 and Reg 10 CV provided nearly the same results: RMSE is lower for Reg 9 while BIC is larger for Reg 10.

Section 7: RANDOM FOREST MODELS

```
library(randomForest)
library(ROCR)
library(pander)

rf <- property_df
rf$description <- NULL
rf$parking_1 <- as.factor(rf$parking_1)
rf$balcony_1 <- as.factor(rf$balcony_1)
rf_d <- rf[, c(9, 13, 14, 19, 20, 21, 22, 24, 25, 39, 46, 53, 54, 60, 61, 63,
    64, 65, 66, 69)]
rf_d <- rf_d[!is.na(rf_d$psqm), ]

rf_d$rnd <- runif(nrow(rf_d))
rf_d <- rf_d[order(rf_d$rnd), ]
rf_d$rnd <- NULL

# Splitting the training and the test set: 60% - 40%
training_d <- rf_d[0:round(nrow(rf_d) * 0.6), ]
test_d <- rf_d[(round(nrow(rf_d) * 0.6) + 1):nrow(rf_d), ]

# Number of predictors: tried models with 4, 8 and default = sqrt(p)
# predictors per split In all forests 100 trees were fitted Seed is set, so
# the results are reproducible

# 8 predictors
set.seed(201703)
rf_8 <- randomForest(psqm ~ ., data = training_d, ntree = 100, mtry = 8, importance = TRUE,
    na.action = na.exclude)
pander(rf_8)
round(importance(rf_8), 2)
pred <- predict(rf_8, test_d)

rf_8_rmse <- RMSE_Lev(pred = predict(rf_8, test_d), obs = test_d$psqm, na.rm = TRUE)
```

```r
# 4 predictors
set.seed(201712)
rf_4 <- randomForest(psqm ~ ., data = training_d, ntree = 500, mtry = 4, importance = TRUE,
    na.action = na.exclude)
pander(rf_4)
round(importance(rf_4), 2)
pred <- predict(rf_4, test_d)

rf_4_rmse <- RMSE_Lev(pred = predict(rf_4, test_d), obs = test_d$psqm, na.rm = TRUE)


# sqrt(p) predictors
set.seed(201711)
rf_def <- randomForest(psqm ~ ., data = training_d, ntree = 500, importance = TRUE,
    na.action = na.exclude)
pander(rf_def)
round(importance(rf_def), 2)
pred <- predict(rf_def, test_d)

rf_def_rmse <- RMSE_Lev(pred = predict(rf_def, test_d), obs = test_d$psqm, na.rm = TRUE)
```

The RF model with 8 predictors at each split provided the lowest error rate and could explain most of the variance. The most important variables based on the % of MSE reduction are the concrete_blockflat dummy, the size in square meter spline and heating - the best predictors of price per square meter of flats in the dataset.

SECTION 8: Visualizing fit and error rate

```r
# REG_9
act <- property_df[property_df$train == 0, ]$psqm
# act_mat <- matrix(act, nrow = length(act), ncol = 1)
p_9 <- predict(reg_9, newdata = property_df[property_df$train == 0, ])
# p_9_mat <- matrix(p_9, nrow = length(p_9), ncol = 1)
err_9 <- (p_9 - act)
# err_9_mat <- matrix(err_9, nrow = length(err_9), ncol = 1)

# fit_9 <- cbind(act_mat, p_9_mat, err_9_mat)
fit_9 <- cbind(act, p_9, err_9)
colnames(fit_9) <- c("actual", "prediction", "error")
fit_9 <- data.table(fit_9)

# Plotting actual and predicted values, coloured with the error
png(filename = "Graph_fit_9.png", res = 200, width = 1200, height = 800)
ggplot(data = fit_9, aes(x = act, y = p_9)) + geom_point(size = 0.5, aes(col = err_9)) +
    geom_abline(aes(slope = 1, intercept = 0), size = 0.5, col = "blue") + geom_smooth(method = "lm",
    size = 0.5, colour = "red", se = FALSE) + labs(x = "Actual values of Price per square meter",
    y = "Predicted values of Price per square meter", title = "Out-of-sample model fit for Regression 9
    scale_color_distiller("Error", palette = "Spectral") + theme_bw()
dev.off()

# Plotting predictions and error
png(filename = "Graph_error_9.png", res = 200, width = 1200, height = 800)
```

```r
ggplot(data = fit_9, aes(x = p_9, y = err_9)) + geom_point(size = 0.5, aes(col = err_9)) +
    geom_smooth(method = "loess", size = 0.5, colour = "darkgrey", se = TRUE) +
    labs(x = "Predicted values of Price per square meter", y = "Error of prediction",
        title = "Out-of-sample model error for Regression 9") + scale_color_distiller("Error",
    palette = "Spectral") + theme_bw()
dev.off()


# REG_7
act <- property_df[property_df$train == 0, ]$psqm
# act_mat <- matrix(act, nrow = length(act), ncol = 1)
p_7 <- predict(reg_7, newdata = property_df[property_df$train == 0, ])
# p_7_mat <- matrix(p_7, nrow = length(p_7), ncol = 1)
err_7 <- (p_7 - act)
# err_7_mat <- matrix(err_7, nrow = length(err_7), ncol = 1)

# fit_7 <- cbind(act_mat, p_7_mat, err_7_mat)
fit_7 <- cbind(act, p_7, err_7)
colnames(fit_7) <- c("actual", "prediction", "error")
fit_7 <- data.table(fit_7)

# Plotting actual and predicted values, coloured with the error
png(filename = "Graph_fit_7.png", res = 200, width = 1200, height = 800)
ggplot(data = fit_7, aes(x = act, y = p_7)) + geom_point(size = 0.5, aes(col = err_7)) +
    geom_abline(aes(slope = 1, intercept = 0), size = 0.5, col = "blue") + geom_smooth(method = "lm",
    size = 0.5, colour = "red", se = FALSE) + labs(x = "Actual values of Price per square meter",
    y = "Predicted values of Price per square meter", title = "Out-of-sample model fit for Regression 7
    scale_color_distiller("Error", palette = "Spectral") + theme_bw()
dev.off()

# Plotting predictions and error
png(filename = "Graph_error_7.png", res = 200, width = 1200, height = 800)
ggplot(data = fit_7, aes(x = p_7, y = err_7)) + geom_point(size = 0.5, aes(col = err_7)) +
    geom_smooth(method = "loess", size = 0.5, colour = "darkgrey", se = TRUE) +
    labs(x = "Predicted values of Price per square meter", y = "Error of prediction",
        title = "Out-of-sample model error for Regression 7") + scale_color_distiller("Error",
    palette = "Spectral") + theme_bw()
dev.off()


# REG_10
act <- property_df[property_df$train == 0, ]$psqm
# act_mat <- matrix(act, nrow = length(act), ncol = 1)
p_10 <- predict(reg_10, newdata = property_df[property_df$train == 0, ])
# p_10_mat <- matrix(p_10, nrow = length(p_10), ncol = 1)
err_10 <- (p_10 - act)
# err_10_mat <- matrix(err_10, nrow = length(err_10), ncol = 1)

# fit_10 <- cbind(act_mat, p_10_mat, err_10_mat)
fit_10 <- cbind(act, p_10, err_10)
colnames(fit_10) <- c("actual", "prediction", "error")
fit_10 <- data.table(fit_10)
```

```r
# Plotting actual and predicted values, coloured with the error
png(filename = "Graph_fit_10.png", res = 200, width = 1200, height = 800)
ggplot(data = fit_10, aes(x = log(act), y = p_10)) + geom_point(size = 0.5,
    aes(col = err_10)) + geom_abline(aes(slope = 1, intercept = 0), size = 0.5,
    col = "blue") + geom_smooth(method = "lm", size = 0.5, colour = "red", se = FALSE) +
    labs(x = "Actual values of Price per square meter", y = "Predicted values of Price per square meter
        title = "Out-of-sample model fit for Regression 10") + scale_color_distiller("Error",
    palette = "Spectral") + theme_bw()
dev.off()

# Plotting predictions and error
png(filename = "Graph_error_10.png", res = 200, width = 1200, height = 800)
ggplot(data = fit_10, aes(x = p_10, y = err_10)) + geom_point(size = 0.5, aes(col = err_10)) +
    geom_smooth(method = "loess", size = 0.5, colour = "darkgrey", se = TRUE) +
    labs(x = "Predicted values of Price per square meter", y = "Error of prediction",
        title = "Out-of-sample model error for Regression 10") + scale_color_distiller("Error",
    palette = "Spectral") + theme_bw()
dev.off()


# RF_8
act_rf <- test_d$psqm
# act_mat_rf <- matrix(act, nrow = length(act_rf), ncol = 1)
p_rf_8 <- predict(rf_8, newdata = test_d)
# p_rf_8_mat <- matrix(p_rf_8, nrow = length(p_rf_8), ncol = 1)
err_rf_8 <- (p_rf_8 - act_rf)
# err_rf_8_mat <- matrix(err_rf_8, nrow = length(err_rf_8), ncol = 1)

# fit_9 <- cbind(act_mat, p_9_mat, err_9_mat)
fit_rf_8 <- cbind(act_rf, p_rf_8, err_rf_8)
colnames(fit_rf_8) <- c("actual", "prediction", "error")
fit_rf_8 <- data.table(fit_rf_8)

# Plotting actual and predicted values, coloured with the error
png(filename = "Graph_fit_rf_8.png", res = 200, width = 1200, height = 800)
ggplot(data = fit_rf_8, aes(x = act_rf, y = p_rf_8)) + geom_point(size = 0.5,
    aes(col = err_rf_8)) + geom_abline(aes(slope = 1, intercept = 0), size = 0.5,
    col = "blue") + geom_smooth(method = "lm", size = 0.5, colour = "red", se = FALSE) +
    labs(x = "Actual values of Price per square meter", y = "Predicted values of Price per square meter
        title = "Random Forest model fit for 8 predictors") + scale_color_distiller("Error",
    palette = "Spectral") + theme_bw()
dev.off()

# Plotting predictions and error
png(filename = "Graph_error_rf_8.png", res = 200, width = 1200, height = 800)
ggplot(data = fit_rf_8, aes(x = p_rf_8, y = err_rf_8)) + geom_point(size = 0.5,
    aes(col = err_rf_8)) + geom_smooth(method = "loess", size = 0.5, colour = "darkgrey",
    se = TRUE) + labs(x = "Predicted values of Price per square meter", y = "Error of prediction",
    title = "Random Forest model error for 8 predictors") + scale_color_distiller("Error",
    palette = "Spectral") + theme_bw()
dev.off()
```

For new variables added (view, orientation, bacony, parking) complex scaling performed slightly better than simple scales, therefore it is advised to use more complex scaling for these factor variables. In regressions

with no interactions Log models performed slightly better than levels, however, they brought unnecessary model complexity and exposure to massive overfitting. In regressions with interactions Log models did not really improve our prediction.

From BIC perspective, Reg 10 with cross-validation is the best because of its predictive power. From the RMSE perspective, Reg 9 without cross-validation provided the lowest error rate, however, both models are rather complex. If our aim is to provide the best prediction possible, we will have to use one of the more complex models.

Out of the models listed, the preferred one is Reg 9, because it performs the best without cross-validation as well and has the lowest error (125.39) and a high-enough BIC (60,148). A Random Forest model was also estimated with an error rate of 111.50, which is significantly lower than all of the regression models above and also provided meaningful insight into variable importance. Therefore, the preferred method of choice for the prediction would be Random Forest. The most important variables based on error reduction are the concrete_blockflat dummy, the size in square meter spline and heating - the best predictors of price per square meter of flats in the dataset.