# CompStat/R - Paper 1

*Group 2: Carlo Michaelis, Patrick Molligo, Lukas Ruff*

*11 May 2016*

## Part I

### 1. What are the atomic vector types in R? Explain which value they can take and give an example!

There are six *atomic* (or *basic*) vector types in R:

- **character**: Text, i.e. string variables.
- **numeric**: Real numbers, i.e. float variables.
- **integer**: Integers, i.e. values in $\mathbb{Z}$.
- **complex**: Complex numbers, i.e. a pair of values with a real and imaginary part.
- **logical**: Boolean variables, i.e. either 1 (`TRUE`) or 0 (`FALSE`).
- **raw**: A raw vector contains fixed-length sequences of bytes.

**Examples**

```
a <- c("blue", "red", "yellow")      ## character
b <- c(pi, exp(1), 0, 1)             ## numeric
c <- 1:10                            ## integer
d <- c(0+1i, 1+1i)                   ## complex
e <- c(TRUE, FALSE)                  ## logical
f <- raw(length = 3L)                ## raw
```

It is important to note, that a vector can only contain elements of the same type. We can check the type of an object using the `class`-function.

```
# verify types by using class function
lapply(list(a,b,c,d,e,f), class)
```

```
## [[1]]
## [1] "character"
##
## [[2]]
## [1] "numeric"
##
## [[3]]
## [1] "integer"
##
## [[4]]
## [1] "complex"
##
## [[5]]
## [1] "logical"
##
## [[6]]
## [1] "raw"
```

**2. What is the difference between generic and atomic vectors?**

- An *atomic vector* can only contains objects of the same class. An example would be a vector which contains only integers.
- A *generic vector* (in R representend as a `list`) can conatain objects of different classes. An example would be a vector which contains characters and numbers.

**3. Explain the following statement: "A data frame is a list, but not every list is a data frame."**

- A `list` is an object containing collections of objects. The types of the elements of the list can be different. It is for example allowed that a `list` contains a vector of real values (doubles) and a vector of characters. The length of the containing vectors can be *different*.
- A `data frame` is also an object containing colletions of objects. The types of the elements of the list can also be different. But the length of the containing vectors have to be *the same*. We can think of a `data frame` as a table or matrix, where each row is an observation and each column a different variable. The length of each element or column are the number of rows or observations.

In conclusion `list` and `data frame` are very similar, but the `data frame` has one more restriction (same length of all vectors). That is why a `data frame` is always a `list`, but a `list` is not always a `data frame`.

## Part II

The following code will perform a simulation of 100'000'000 samples from a $\mathcal{N}(5, 10)$ distribution, i.e. a normal distribution with mean $\mu = 5$ and standard deviation $\sigma = 10$. For reproducibility, we set a seed for the random number generator. In a second step, the cumulative sums of the first 100 samples are computed in two different ways, where the function `cumsum` returns a vector where element $i$ is the cumulative sum up to sample $i$. Finally, we check if the two ways of computing the cumulative sums up to sample 100 result in exactly equal vectors.

For random number generation R uses pseudo-random numbers. Starting from an initial state, called *seed state*, it will produce a deterministic sequence, which is used as random numbers. By choosing the same seed in every turn, we get the same results. To make the results of random numbers comparable, we first set the seed in a sepecific state, using `set.seed`.
After setting the seed, we define a vector with (pseudo-) random values. Using the `rnorm` function we create the $1 \cdot 10^8$ normal distributed random values and save them in a vector called `largeVector`.

```r
# Set the state of the random number generator (RNG) to 1
set.seed(1)

# Perform simulation of 1e8 samples from a normal distribution with mean 5
# and standard deviation 10
largeVector <- rnorm(1e6, mean=5, sd=10)
```

The function `cumsum`, which is used in the next code block, calculates the cumulative sum of the values of the vector. It takes all elements one by one and calculates for this element the sum of all elements before, including the current element. These values will be the new elements of the new vector. Consider following example:

$$\begin{pmatrix} 1 \\ 4 \\ 3 \end{pmatrix} \xrightarrow{\texttt{cumsum}} \begin{pmatrix} 1 \\ 5 \\ 8 \end{pmatrix}$$

In case `a` it is doing `cumsum` of the whole vector `largeVector`. Afterwards it just takes the first 100 elements and saves them in vector `a`. In case `b` it first takes the 100 first elements of `largeVector` and calculates the `cumsum` afterwards, with only those 100 elements. The result is saved in vector `b`. In the end the two vectors `a` and `b` are checked for exact equality, using `identical` function.

```
# Compute the cumulative sums for the whole "largeVector" and subset the
# first 100 elements
a <- cumsum(largeVector)[1:100]

# Compute the cumulative sums only for the first 100 elements of
# "largeVector"
b <- cumsum(largeVector[1:100])

# Check, whether both ways of computation are exactly identical
identical(a, b)
```

```
## [1] TRUE
```

Of course, both ways of computing the cumulative sums for the first 100 samples above have the same result and hence `identical(a, b)` returns `TRUE`, but computation `a` is very inefficient compared to computation `b` since we first apply `cumsum` to the whole `largeVector`, i.e. we compute the cumulative sums for $100'000'000$ elements and then only look at the first 100 elements. Computation `b` instead only computes the cumulative sums for the subset of the first 100 elements directly.

In the following code, we stop the time for each of the two ways of computation using `system.time` function.

```
# Computation method a
system.time(cumsum(largeVector)[1:100])
```

```
##    user  system elapsed
##   0.010   0.001   0.012
```

```
# Computation method b
system.time(cumsum(largeVector[1:100]))
```

```
##    user  system elapsed
##   0.001   0.000   0.000
```

The *user* CPU time and the *system* CPU time is a technical distinction in time running the R code and time used in operating system kernel on behalf of the R code. The interesting time is the *elapsed* time, which is the sum of the *user* time and the *system* time. We can see that the first operation of taking the `cumsum` of the whole `largeVector` with its 100 million elements (and reducing the vector to 100 elements afterwards) takes a lot more CPU calucaltion time than taking the `cumsum` of the first 100 elements directly.

The results prove our reasoning above, the second method is much more efficient than the first method, because finally end we are only interested in the `cumsum` of the first 100 elements of the vector.

## Part III

In our regression analysis, we will analyze the rental prices in Munich from 2003 using the dataset "Münchner Mietspiegel 2003". The dataset contains 13 variables from 2053 apartments in Munich. The variables are the following:

- **nm**: net rent in EUR
- **nmqm**: net rent per $m^2$ in EUR
- **wfl**: living space in $m^2$
- **rooms**: number of rooms
- **bj**: year of construction
- **bez**: district
- **wohngut**: good residential area? (Y=1, N=0)

- **wohnbest**: best residential area? (Y=1, N=0)
- **ww0**: hot water supply? (Y=0, N=1)
- **zh0**: central heating? (Y=0, N=1)
- **badkach0**: tiled bathroom? (Y=0, N=1)
- **badextra**: optional extras in bathroom? (Y=1, N=0)
- **kueche**: luxury kitchen? (Y=1, N=0)

We would like to predict and explain the rental prices, i.e. the dependent variable of our regression analysis will be the net rent in EUR `nm`. All other variables are potential explanatory variables for our linear regression model.

### Data Import, Validation and Cleaning

First, we read the data into our global environment using the `load`-function and have a first look at it using `str` and `summary`:

```r
# Load data
load('miete.Rdata')
# Get a first overview
str(miete)
```

```
## 'data.frame':    2053 obs. of  13 variables:
##  $ nm      : num  741 716 528 554 698 ...
##  $ nmqm    : num  10.9 11.01 8.38 8.52 6.98 ...
##  $ wfl     : int  68 65 63 65 100 81 55 79 52 77 ...
##  $ rooms   : int  2 2 3 3 4 4 2 3 1 3 ...
##  $ bj      : num  1918 1995 1918 1983 1995 ...
##  $ bez     : Factor w/ 25 levels "1","2","3","4",..: 2 2 2 16 16 16 6 6 6 6 ...
##  $ wohngut : int  1 1 1 0 1 0 0 0 0 0 ...
##  $ wohnbest: int  0 0 0 0 0 0 0 0 0 0 ...
##  $ ww0     : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ zh0     : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ badkach0: int  0 0 0 0 0 0 0 0 0 0 ...
##  $ badextra: int  0 0 0 1 1 0 1 0 0 0 ...
##  $ kueche  : int  0 0 0 0 1 0 0 0 0 0 ...
```

```r
summary(miete)
```

```
##        nm               nmqm             wfl             rooms
##  Min.   :  77.31   Min.   : 1.470   Min.   : 17.0   Min.   :1.000
##  1st Qu.: 389.95   1st Qu.: 6.800   1st Qu.: 53.0   1st Qu.:2.000
##  Median : 534.30   Median : 8.470   Median : 67.0   Median :3.000
##  Mean   : 570.09   Mean   : 8.394   Mean   : 69.6   Mean   :2.598
##  3rd Qu.: 700.48   3rd Qu.:10.090   3rd Qu.: 83.0   3rd Qu.:3.000
##  Max.   :1789.55   Max.   :20.090   Max.   :185.0   Max.   :6.000
##
##        bj            bez          wohngut          wohnbest
##  Min.   :1918   9      : 177   Min.   :0.0000   Min.   :0.00000
##  1st Qu.:1948   2      : 161   1st Qu.:0.0000   1st Qu.:0.00000
##  Median :1960   5      : 139   Median :0.0000   Median :0.00000
##  Mean   :1958   4      : 137   Mean   :0.3911   Mean   :0.02192
##  3rd Qu.:1973   3      : 132   3rd Qu.:1.0000   3rd Qu.:0.00000
##  Max.   :2001   25     : 117   Max.   :1.0000   Max.   :1.00000
##                 (Other):1190
##       ww0              zh0              badkach0          badextra
```

```
##  Min.   :0.00000   Min.   :0.00000   Min.   :0.0000   Min.   :0.00000
##  1st Qu.:0.00000   1st Qu.:0.00000   1st Qu.:0.0000   1st Qu.:0.00000
##  Median :0.00000   Median :0.00000   Median :0.0000   Median :0.00000
##  Mean   :0.03507   Mean   :0.08524   Mean   :0.1851   Mean   :0.09303
##  3rd Qu.:0.00000   3rd Qu.:0.00000   3rd Qu.:0.0000   3rd Qu.:0.00000
##  Max.   :1.00000   Max.   :1.00000   Max.   :1.0000   Max.   :1.00000
##
##      kueche
##  Min.   :0.00000
##  1st Qu.:0.00000
##  Median :0.00000
##  Mean   :0.07306
##  3rd Qu.:0.00000
##  Max.   :1.00000
##
```

Before we go into the variables of our data in detail, let's do a quick check on missing values:

```r
# Check for NA's
sum(is.na(miete))
```

```
## [1] 0
```

There seem to be no missing values in our dataset. Now, let's think about plausibility and the data types of our variables. From the five-number summary (`Min.`, `1st Qu.`, `Median`, `3rd Qu.`, `Max,`) and `Mean` values shown by `summary`, we can see that `nm`, `nmqm`, `wfl`, and `rooms` are properly formated and within reasonable ranges. By definition of the variables, we should have that

$$\frac{\texttt{nm}}{\texttt{wfl}} = \texttt{nmqm} \tag{1}$$

Let's check whether this relationship holds by comparing the summary of `nmqm` and $\frac{\texttt{nm}}{\texttt{wfl}}$ and having a look at the sum of absolute errors (in relative terms):

```r
summary(miete$nmqm)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   1.470   6.800   8.470   8.394  10.090  20.090
```

```r
nmqm2 <- miete$nm / miete$wfl
summary(nmqm2)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   1.467   6.799   8.466   8.394  10.090  20.090
```

```r
sum(abs(miete$nmqm - nmqm2)) / sum(nmqm2)
```

```
## [1] 0.0002932959
```

There are only minor differences which are negligible and probably caused by rounding originally numeric values of `wfl` to integers. Since the year of construction, `bj`, contains values of years, we can convert it to integers:

```r
miete$bj <- as.integer(miete$bj)
```

The factor variable `bez`, indicating the district where the respective flat is located, has 25 levels. Let's have a closer look:

```r
table(miete$bez)
```

```
##
##   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18
##  43 161 132 137 139  66  69  62 177  58  70  78  98  60  43 115  67  82
##  19  20  21  22  23  24  25
## 106  50  56  24  14  29 117
```

The remaining variables (`wohngut`, `wohnbest`, `ww0`, `zh0`, `badkach0`, `badextra`, `kueche`) are all binary and valid which we can see from the summary above, since `Min.` is 0 and `Max.` is 1 for all those variables. Let's convert them properly:

```r
miete[7:13] <- lapply(miete[7:13], as.logical)
```

Now, we have a nice and tidy dataset and can proceed exploring our data.

```r
# library(psych)
# describe(miete)
```

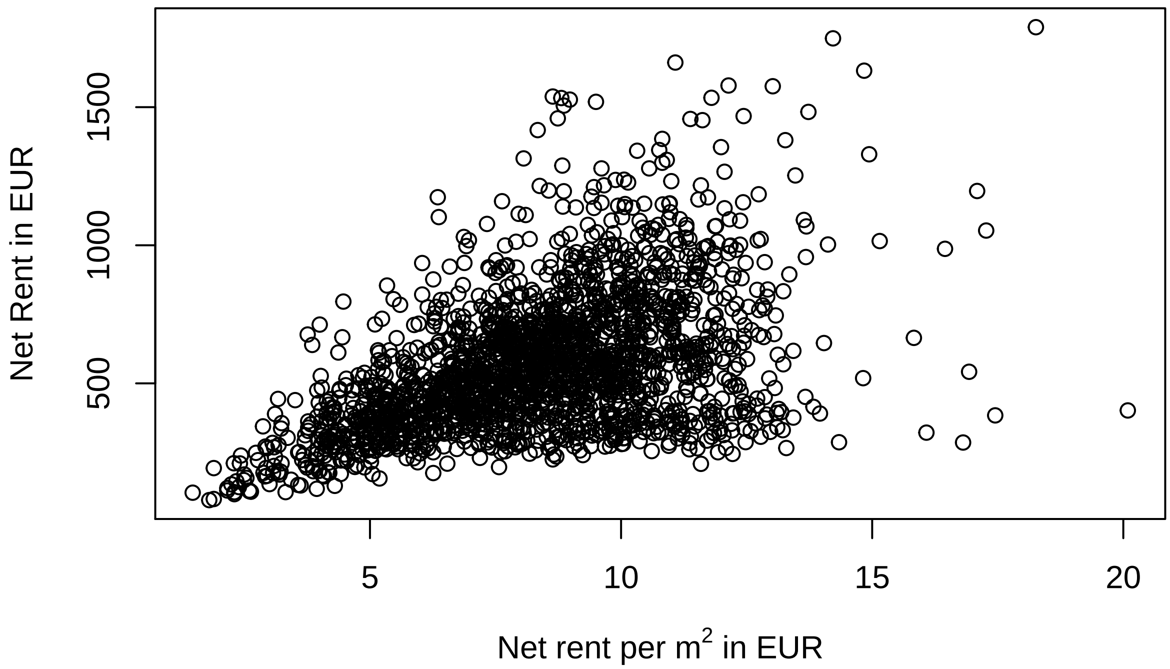
### Exploratory Analysis

Before building a model, we would like to better understand our data by using different plots and methods of analysis.

The dependent variable of our model will be `nm`. Therefore, it would be nice to have a look at some scatterplots with different regressors to get a first impression on the correlation between the dependent variable and the potential regressors.

Net rent per $m^2$ (`nmqm`) is the net rent (`nm`) per living space (`wfl`) as we have already seen above. Therefore, it is not appropriate to use `nmqm` as an explanatory variable because we would use rent pricing information to explain rent pricing information. Since we have living space `wfl` as a sepearte variable, `nmqm` is of no additional explanatory value. Hence, we would expect `nm` to be highly positively correlated with `nmqm`. Let's verify our reasoning with a scatterplot:

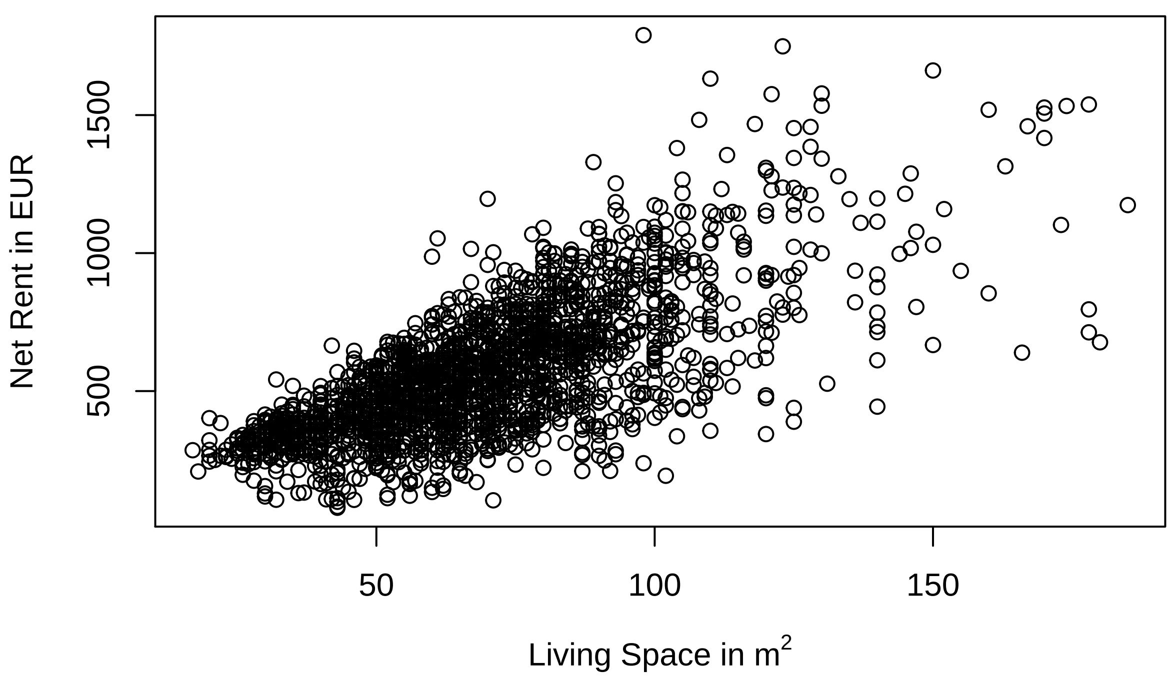
```r
plot(miete$nmqm,
     miete$nm,
     xlab = expression(paste("Net rent per m"^"2", " in EUR")),
     ylab = "Net Rent in EUR")
```




As expected, we can see a strong positive correlation.

Next, we will have a look at living space `wfl`. Naturally, one would assume prices to be higher for larger spaces. Let's have a look:

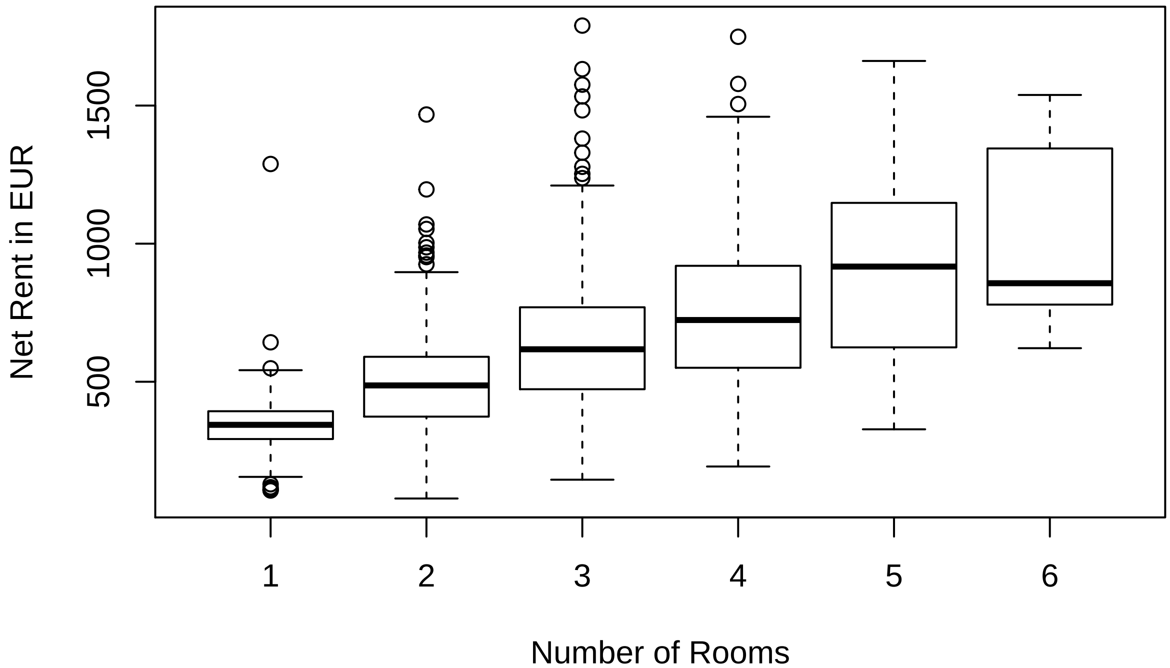
```
plot(miete$wfl,
     miete$nm,
     xlab = expression(paste("Living Space in m"^"2")),
     ylab = "Net Rent in EUR")
```




Indeed, there seems to be a positive correlation and therefore we expect living space to be a significant regressor later in our model.

A further potential regressor are the number of rooms (`rooms`) available in a flat. Number of rooms ranges from 1 to 6 rooms at most. Therefore, a boxplot would be nice to get a first impression on how net rent varies with room number:

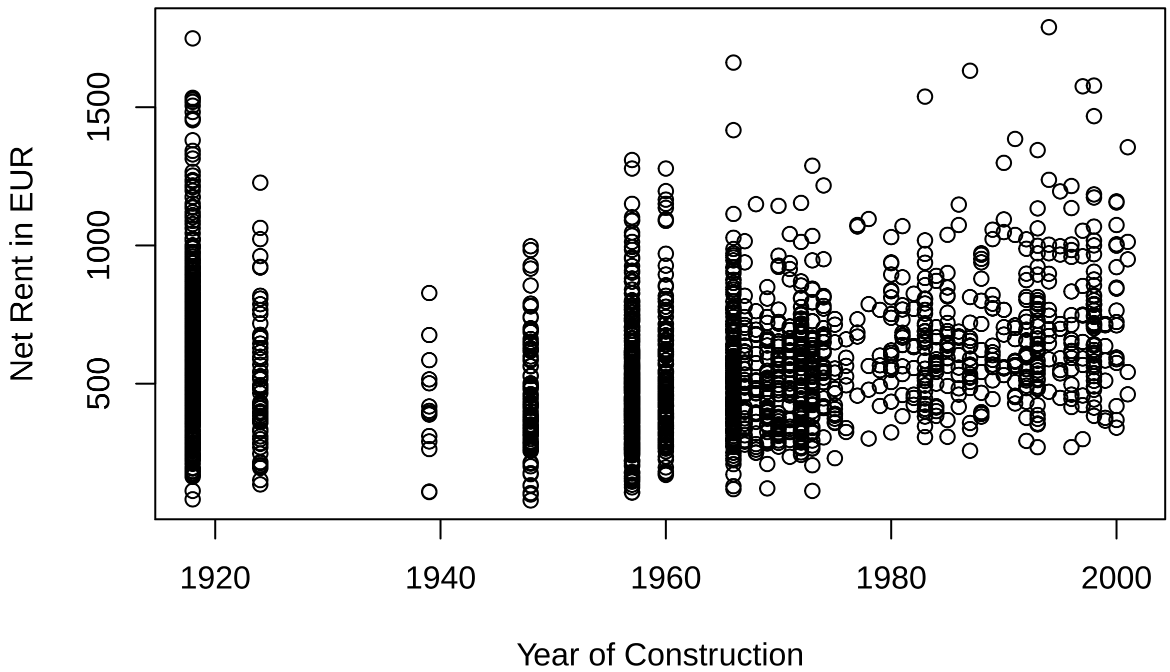
```
boxplot(nm ~ as.factor(rooms), data = miete,
        xlab = "Number of Rooms",
        ylab = "Net Rent in EUR")
```




From the boxplot, we can observe higher net rents for flats with more rooms (although from 5 to 6 rooms there doesn't seem to be a significant difference). But we have to be careful with our reasoning. Since more rooms most likely mean larger living space (or the other way round), this positive relationship in the plot could already be explained by `wfl`. For example, if people generally prefer more open rooms for some fixed living space, i.e. fewer rooms per space, and are willing to pay more for this kind of architechture, then there could even be a reducing effect of more rooms on renting prices, when pure living space has already explained a higher renting price level.

For the effect of the year of construction on net rents, we do not have a clear intuition, since very old, but renovated buildings could also be of high value. Let's look at the scatterplot:

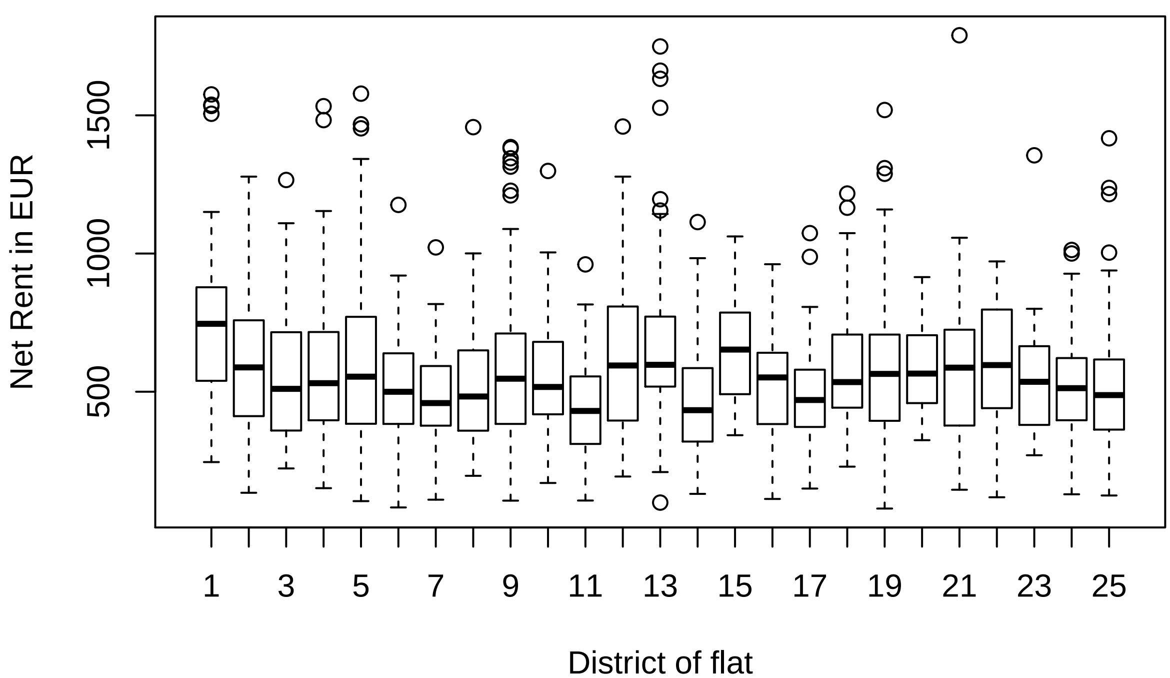
```
plot(miete$bj,
     miete$nm,
     xlab = "Year of Construction",
     ylab = "Net Rent in EUR")
```




From the plot, there doesn't seem to be a significant relationship.

Another candidate for providing explanatory value on rent levels is the district, where the respective property is located (`bez`). For example, one would expect higher levels in districts close to the center of Munich. Overall, the observations in our dataset are located in 25 different districts. A complete list of all districts of Munich for example can be found at en.wikipedia.org/wiki/Boroughs_of_Munich. Munich has 25 districts in total, i.e. the dataset contains flats from all districts. Let's consider a boxplot:

```
boxplot(nm ~ bez, data = miete,
        xlab = "District of flat",
        ylab = "Net Rent in EUR")
```



From the boxplot we can see, that rental prices in district 1 are relatively high. This is the "Altstadt-Lehel"-district, which is the center of Munich. Since we will incorporate the factor variable `bez` as a dummy-variables in our linear regression, the "Altstadt-Lehel"-district will be our reference-district (zero encoded). Hence, we expect from looking at the boxplot, that different districts will have a decreasing effect on rental prices when compared to the benchmark "Altstadt-Lehel". For example, lower rental prices could be expected in district 11 ("Milbertshofen-Am Hart") or district 14 ("Berg am Laim").

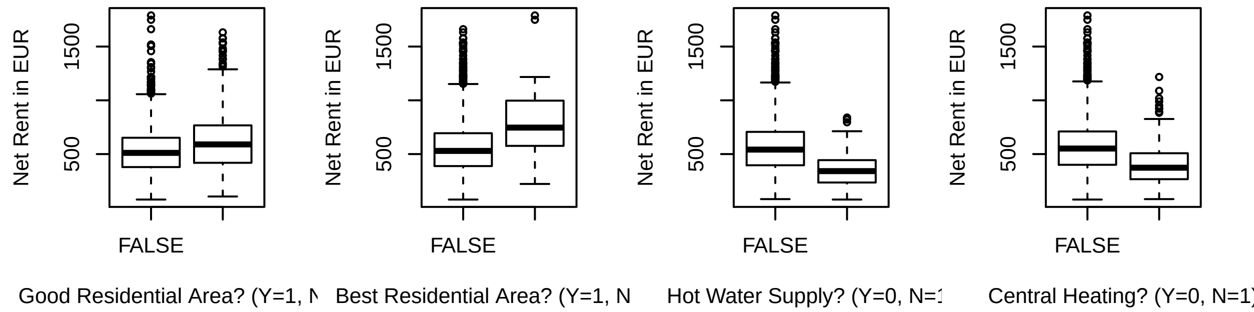
To complete our exploratory analysis, let's consider a further plot, showing boxplots of all binary variables:

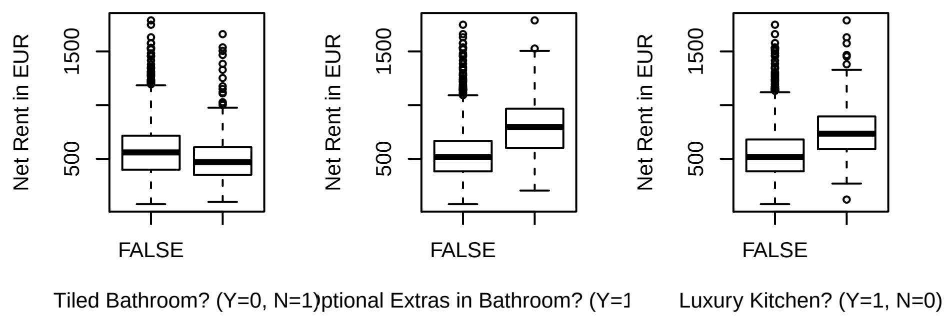
```r
# Prepare for multiple base plots
par(mfrow = c(2,4))

# Labels
nmLab <- "Net Rent in EUR"
BinLab <- c("Good Residential Area? (Y=1, N=0)",
            "Best Residential Area? (Y=1, N=0)",
            "Hot Water Supply? (Y=0, N=1)",
            "Central Heating? (Y=0, N=1)",
            "Tiled Bathroom? (Y=0, N=1)",
            "Optional Extras in Bathroom? (Y=1, N=0)",
            "Luxury Kitchen? (Y=1, N=0)")

# Plot
for (i in 7:13){
    boxplot(formula(paste("nm ~ ", names(miete)[i])),
            data = miete,
            xlab = BinLab[i-6],
            ylab = nmLab)
}

# Reset to single base plot
par(mfrow = c(1,1))
```




```r
# Maybe convert binary variables to factor variables with proper Yes/No
# labels... Also clean up xlabs...
```

```r
# Maybe plot covariance matrix of regressors to see relationships between regressors?
```

### Identification of relevant regressors and model fit

To idenfity relevant regressors we can apply `lm()`, which calculates a linear model, to all variables. The first argument of the function is the formular. In our case we want to do a regression of the rent in EUR (`miete\$nm`) on all other variables (we can use the `.` to include all variables). In the second argument we set our dataset.

```r
regrel <- lm(miete$nm ~ ., data = miete)
```

We can omit all variables which have no significant slope. To get the slope we can have a look to the `summary` of the result of the linear regression.

```r
summary(regrel)
```

```
##
## Call:
## lm(formula = miete$nm ~ ., data = miete)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -511.19  -19.26    7.26   27.60  328.92
##
## Coefficients:
##                 Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -903.67622  141.25860  -6.397 1.96e-10 ***
## nmqm           65.33069    0.71104  91.880  < 2e-16 ***
```

```
## wfl             8.29758    0.11435  72.563  < 2e-16 ***
## rooms           2.52110    2.82157   0.894  0.37169
## bj              0.16275    0.07243   2.247  0.02475 *
## bez2           16.40563   11.36719   1.443  0.14911
## bez3           19.32616   11.70560   1.651  0.09889 .
## bez4           13.71455   11.57493   1.185  0.23622
## bez5           13.97488   11.53420   1.212  0.22581
## bez6           19.08483   13.22698   1.443  0.14921
## bez7           15.46011   13.22919   1.169  0.24269
## bez8           25.69554   13.43483   1.913  0.05594 .
## bez9           24.60532   11.30832   2.176  0.02968 *
## bez10          16.46769   13.67104   1.205  0.22851
## bez11          27.77102   13.31002   2.086  0.03706 *
## bez12          19.89723   12.55427   1.585  0.11315
## bez13          24.86314   12.36727   2.010  0.04452 *
## bez14          26.41531   13.58446   1.945  0.05197 .
## bez15          21.80586   14.57358   1.496  0.13474
## bez16          24.94937   12.21742   2.042  0.04127 *
## bez17          22.44613   13.25114   1.694  0.09044 .
## bez18          14.62044   12.74381   1.147  0.25141
## bez19          25.02123   12.25559   2.042  0.04132 *
## bez20          15.82627   13.92585   1.136  0.25590
## bez21          30.21343   13.55179   2.229  0.02589 *
## bez22          27.75977   17.20238   1.614  0.10675
## bez23          20.26342   20.57135   0.985  0.32473
## bez24          29.69837   16.27283   1.825  0.06814 .
## bez25          26.62443   12.09441   2.201  0.02782 *
## wohngutTRUE    -3.52914    3.73042  -0.946  0.34424
## wohnbestTRUE   27.20112   10.44388   2.605  0.00927 **
## ww0TRUE       -45.99457    9.42129  -4.882 1.13e-06 ***
## zh0TRUE        11.53547    6.44480   1.790  0.07362 .
## badkach0TRUE    4.52387    3.84480   1.177  0.23949
## badextraTRUE    7.25510    5.31839   1.364  0.17267
## kuecheTRUE     27.29273    5.84519   4.669 3.22e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 65.64 on 2017 degrees of freedom
## Multiple R-squared:  0.9297, Adjusted R-squared:  0.9285
## F-statistic:   762 on 35 and 2017 DF,  p-value: < 2.2e-16
```

We would suggest to include all variables which are significant on a 99% level (* or **). With the relevant variables we can fit the regression.

```
summary(lm(miete$nm ~ miete$nmqm + miete$wfl + miete$wohnbest +
             miete$ww0 + miete$kueche, data = miete))
```

```
##
## Call:
## lm(formula = miete$nm ~ miete$nmqm + miete$wfl + miete$wohnbest +
##     miete$ww0 + miete$kueche, data = miete)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -511.22  -18.90   10.02   26.26  326.83
```

```
##
## Coefficients:
##                     Estimate Std. Error t value Pr(>|t|)
## (Intercept)        -554.0541     7.7059 -71.900  < 2e-16 ***
## miete$nmqm           64.7417     0.6463 100.166  < 2e-16 ***
## miete$wfl             8.3237     0.0600 138.723  < 2e-16 ***
## miete$wohnbestTRUE   31.9400    10.0179   3.188  0.00145 **
## miete$ww0TRUE       -44.2141     8.2233  -5.377 8.45e-08 ***
## miete$kuecheTRUE     31.1426     5.7326   5.433 6.22e-08 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 65.75 on 2047 degrees of freedom
## Multiple R-squared:  0.9284, Adjusted R-squared:  0.9282
## F-statistic:  5308 on 5 and 2047 DF,  p-value: < 2.2e-16
```

### Discussion of Model Fit and Interpretation of the Model

## References