Exercise 16 - Model diagnostics

Zoltan Kekecs

18 November 2019

Contents

1 Abstract

This exercise will show you how to check whether the assumptions of linear regression hold true for your model, what is the consenquence of the violation of the assumptions, and what to do in case of an assumption being violated.

The latest version of this document and the code the document refers to can be found in the GitHub repository of the class at: https://github.com/kekecsz/PSYP13_Data_analysis_class-2019

2 Data management and descriptive statistics

2.1 Loading packages

You will need the following packages for this exercise.

```
library(psych)  # for describe
library(car)  # for residualPlots, vif, pairs.panels, ncvTest
library(lmtest)  # bptest
library(sandwich)  # for coeftest vcovHC estimator
library(boot)  # for bootstrapping
library(tidyverse)  # for tidy code
```

2.2 Custom functions

We will use these custom functions to get bootstrapped confidence intervals.

```
# function to obtain regression coefficients source:
# https://www.statmethods.net/advstats/bootstrapping.html
bs_to_boot <- function(model, data, indices) {
    d <- data[indices, ]  # allows boot to select sample
    fit <- lm(formula(model), data = d)
    return(coef(fit))
}

# function to obtain adjusted R^2 source:
# https://www.statmethods.net/advstats/bootstrapping.html
# (partially modified)
adjR2_to_boot <- function(model, data, indices) {
    d <- data[indices, ]  # allows boot to select sample
    fit <- lm(formula(model), data = d)
    return(summary(fit)$adj.r.squared)
}


# Computing the booststrap BCa (bias-corrected and
# accelerated) bootstrap confidence intervals by Elfron
# (1987) This is useful if there is bias or skew in the
# residuals.

confint.boot <- function(model, data = NULL, R = 1000) {
    if (is.null(data)) {
        data = eval(parse(text = as.character(model$call[3])))
    }
    boot.ci_output_table = as.data.frame(matrix(NA, nrow = length(coef(model)),
        ncol = 2))
```

```
    row.names(boot.ci_output_table) = names(coef(model))
    names(boot.ci_output_table) = c("boot 2.5 %", "boot 97.5 %")
    results.boot = results <- boot(data = data, statistic = bs_to_boot,
        R = 1000, model = model)

    for (i in 1:length(coef(model))) {
        boot.ci_output_table[i, ] = unlist(unlist(boot.ci(results.boot,
            type = "bca", index = i))[c("bca4", "bca5")])
    }

    return(boot.ci_output_table)
}
```

2.3 Load data about housing prices in King County, USA

In this exercise we will predict the price of apartments and houses.

We use a dataset from Kaggle containing data about housing prices and variables that may be used to predict housing prices. This dataset contains house sale prices for King County, USA (Seattle and sorrounding area) which includes Seattle. It includes homes sold between May 2014 and May 2015. More info about the dataset here: https://www.kaggle.com/harlfoxem/housesalesprediction

We only use a portion of the full dataset now containing information about N = 200 accomodations.

You can load the data with the following code

```
data_house = read_csv("https://bit.ly/2DpwKOr")
```

2.4 Check the dataset

You should always get familiar with the dataset you are usung, and check for any inconsistencies that need to be corrected.

In the code below we convert the area metrics that are in square feet in the original dataset to square meters. We also specify that the variable has_basement is a factor.

```
data_house = data_house %>% mutate(sqm_living = sqft_living *
    0.09290304, sqm_lot = sqft_lot * 0.09290304, sqm_above = sqft_above *
    0.09290304, sqm_basement = sqft_basement * 0.09290304, sqm_living15 = sqft_living15 *
    0.09290304, sqm_lot15 = sqft_lot15 * 0.09290304, has_basement = factor(has_basement))

data_house %>% summary()
```

3 Model diagnostics

Whenever we arrive at a model that we use for statistical inference, we need to make sure that the assumptions of the linear regresion hold true for the model we are working with.

Thus, you have to do model diagnostics for all important models in your analyses. Usually we would do model diagnostics on the final model we arrive at in the end of model selection.

In some cases, if you do result-based post hoc model selection, it might be important to run model diagnostics on the interim steps as well, the models based on which you reached your final model composition.

If you make any adjustments to the data, or the model based on the model diagnostics, remember that you have to re-run model diagnostics on the new model/new dataset.

3.1 Build the model

Here we first build a model to predict the price of the apartment by using only sqm_living and grade as predictors. We are going to run model diagnostics on this model.

```
mod_house2 <- lm(price ~ sqm_living + grade, data = data_house)
```

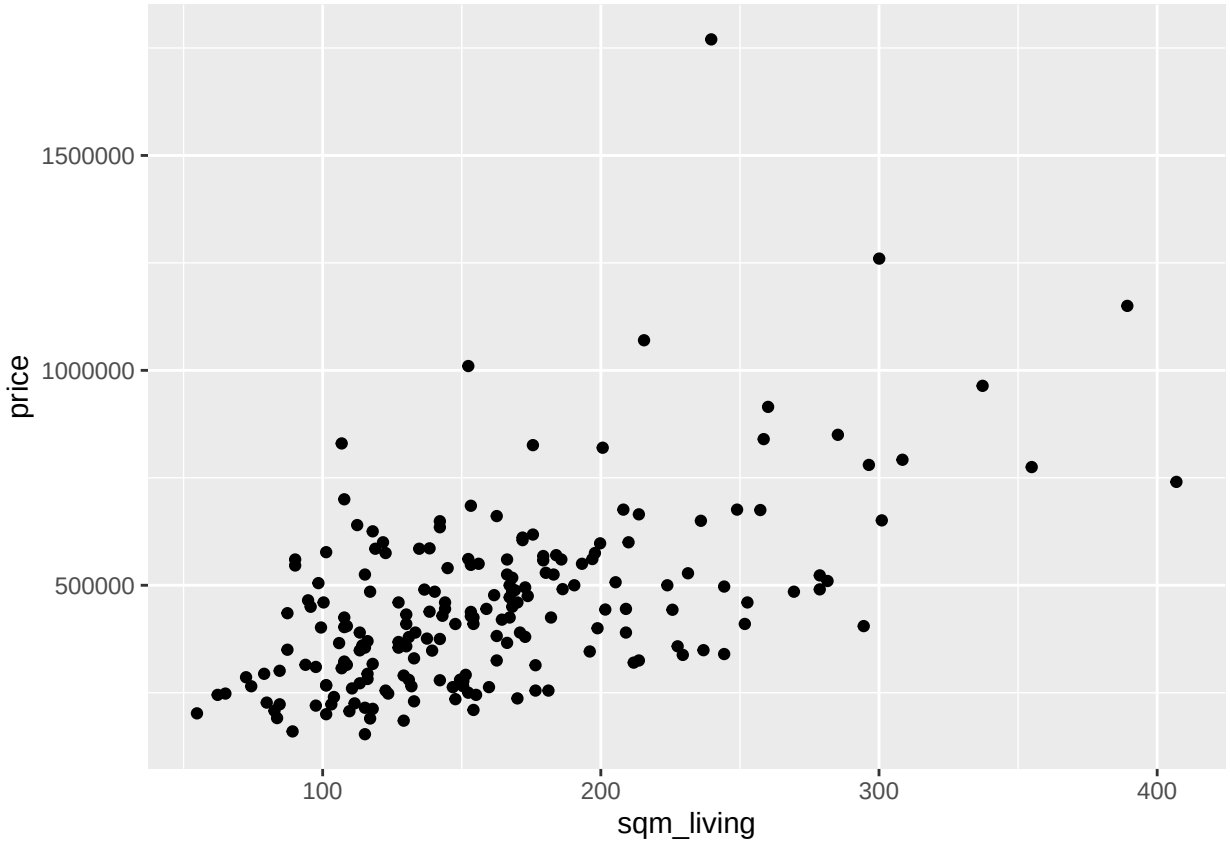
4 Dealing with outliers

4.1 Identifying extreme cases

Cases with extreme values can be identified by visualizing the data and the relationship of the outcome and the predictors one by one.

For example here we visualize price and sqarefootage data on a scatterplot.

```
data_house %>% ggplot() + aes(x = sqm_living, y = price) + geom_point()
```



Notice that most apartment sales had a final sales price below 1 million USD. However, there were a few cases where the price was higher than that. These might be considered extreme cases, especially the one with the sales price of 1.7 million USD.

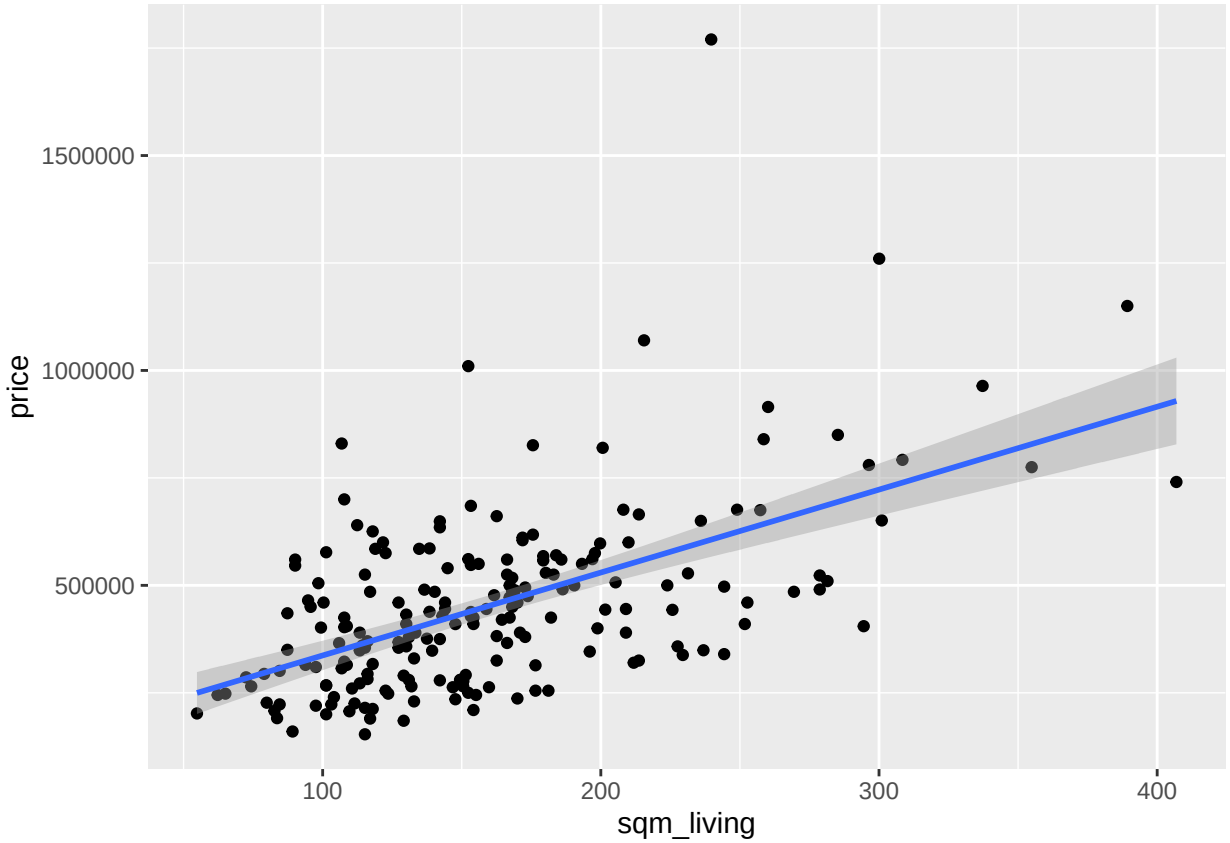
However, it might not be a huge problem if we have a few of these cases in the dataset if we have a lot of data to counteract their effects, especially if they don't have high leverage.

4.2 Identifying extreme cases with high leverage

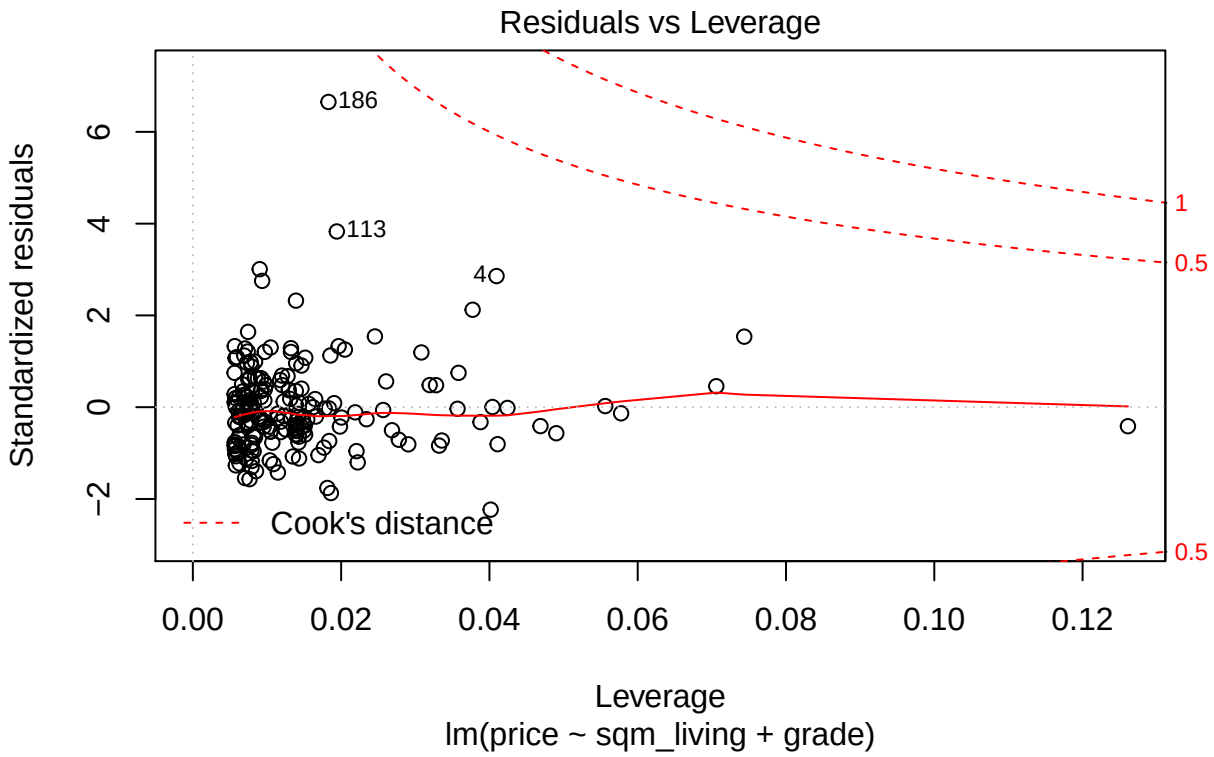
More problematic are cases that not only have very different values from the other data, but they have a high influence on the regression line as well. Highly influential cases can be identified by looking at the scatterplot

with the regression line, by plotting the residual - levarage plot, and by using Cook's distance.

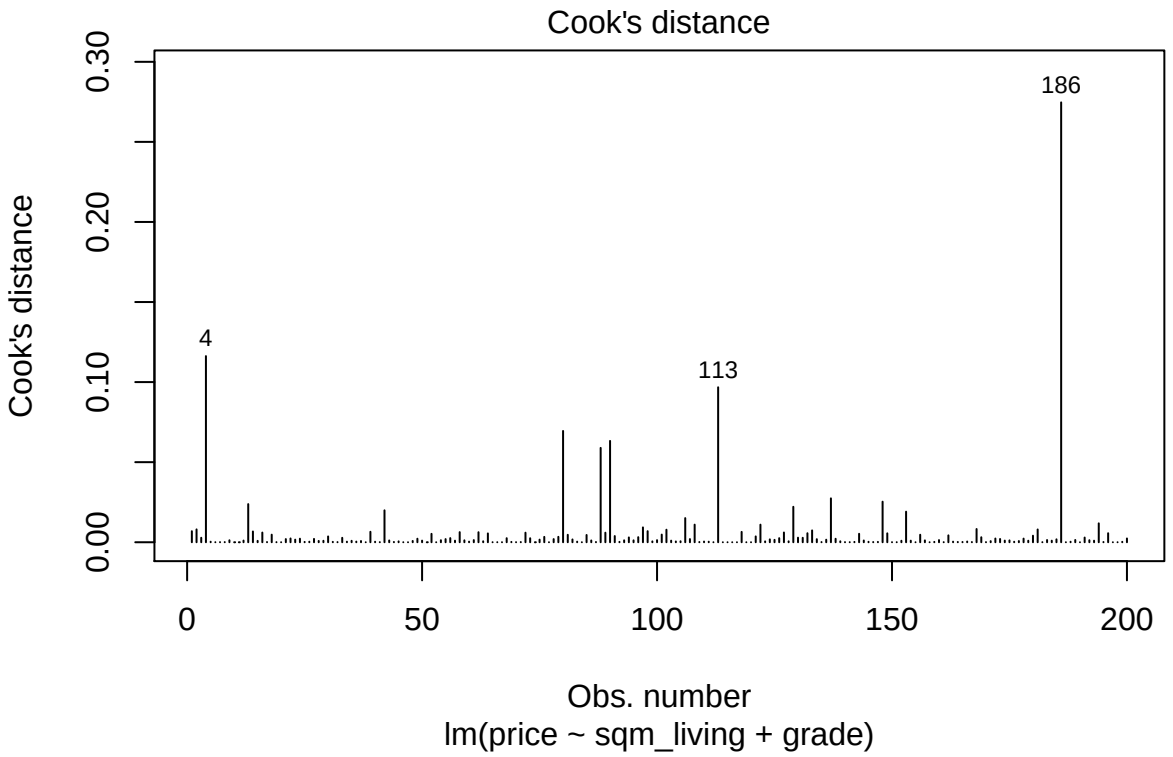
```
data_house %>% ggplot() + aes(x = sqm_living, y = price) + geom_point() +
    geom_smooth(method = "lm")
```




```
mod_house2 %>% plot(which = 5)
```




```
mod_house2 %>% plot(which = 4)
```



Cases that are close to the middle of the regression line on the scatterplot have less leverage while cases at the ends have more. Cases that have high residual error and high leverage are influential cases. Cook's distance is a number that quantifies this, taking into account the combination of "extremeness" and leverage.

It is not well established what should be considered a problematic case, but there are some rules of thumb. Some say that cases with Cook's distance > 1 are influential, while some use a threshold of Cook's distance $> 4/N$. (In our case this is $4/200 = 0.02$).

We have no cases that would have a Cook's distance higher than 1, but we have several where it is higher than 0.02. So we might have a problem with outliers according to one of the criteria.

Lets test whether the assumptions of multiple regression hold true, and determine if we need to do anything with these cases.

5 Assumptions of linear regression

- **Normality**: The residuals of the model must be normally distributed
- **Linearity**: Ther relationship between the outcome variable and the predictor(s) must be linear
- **Homoscedasticity**: The variance of the residuals are similar at all values of the predictor(s)
- **No Multicollinearity**: None of the predictors can be linearly determined by the other predictor(s)

5.1 Normality

The residuals of the model must be normally distributed.

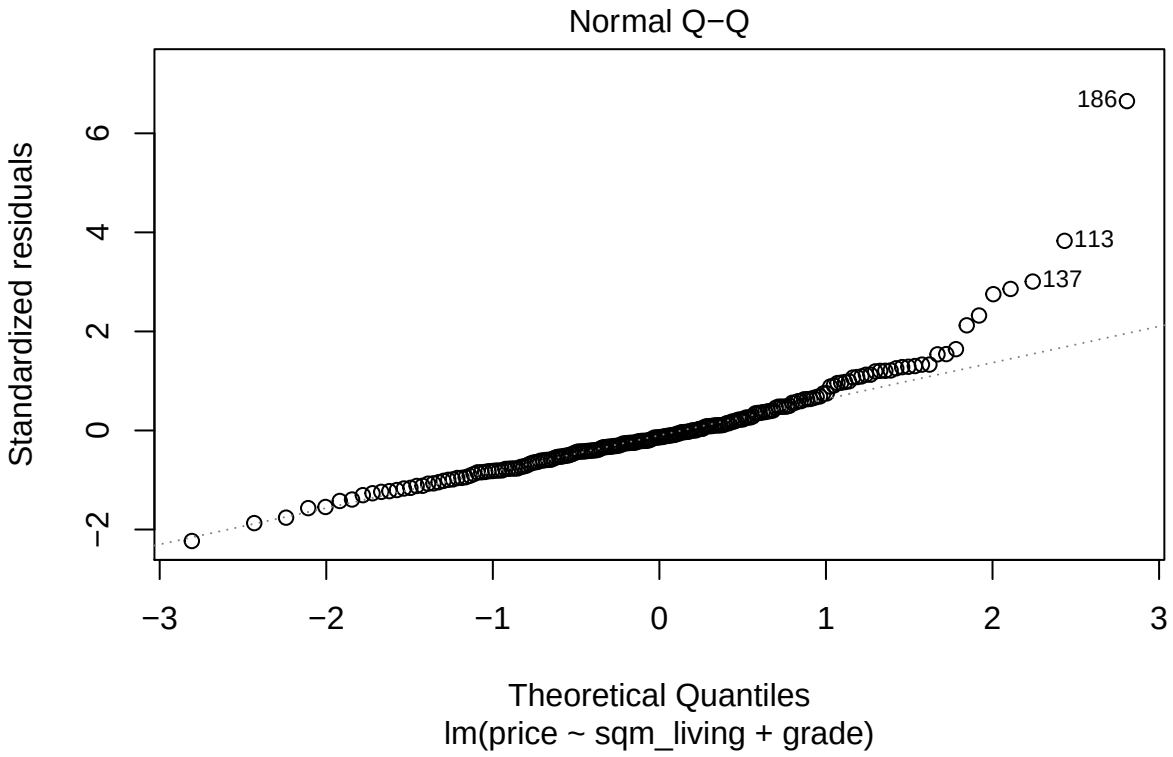
Note that here we talk about the distribution of the prediction error (residulas) of the model, and not the distribution of the individual predictors or the outcome.

We can check this assumption by plotting the QQ plot and checking if the cases are alligned with the theoretical diagonal. If the cases divert from the diagonal (the dashed line in the plot), the assumption of normality might be violated.

We should also look at the histogram of the residuals. We should see a roughly bell shaped (normal) distribution if the assumption of normality is met.

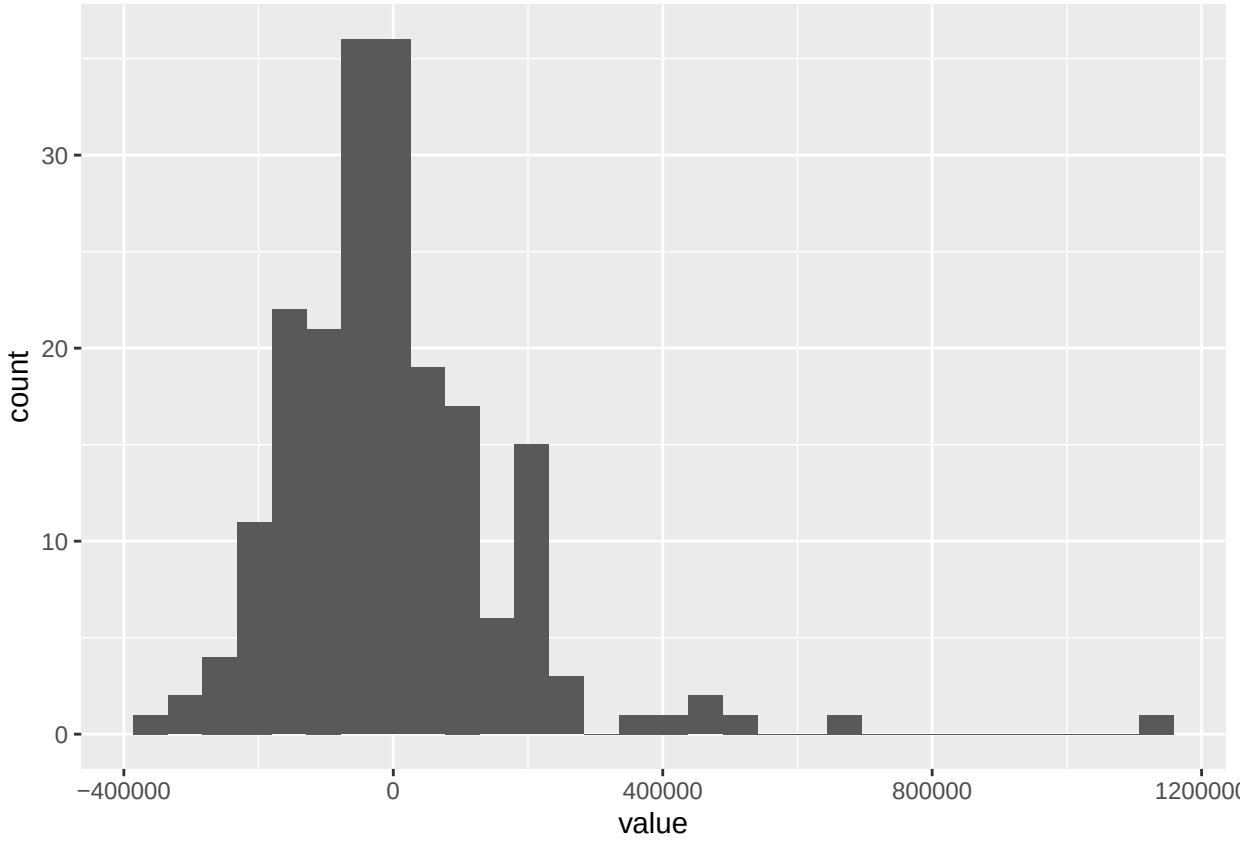
We can also ask for the skew and kurtosis statistics using the describe() function. Skew and kurtosis > 1 can indicate the violation of the assumption of normality.

```
# QQ plot
mod_house2 %>% plot(which = 2)
```




```
# histogram

residuals_mod_house2 = enframe(residuals(mod_house2))
residuals_mod_house2 %>% ggplot() + aes(x = value) + geom_histogram()
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```




```
# skew and kurtosis
describe(residuals(mod_house2))
```

```
##    vars   n mean       sd    median  trimmed      mad       min     max
## X1    1 200    0 169214.6 -23118.96 -14686.4 125887.3 -371917.1 1120748
##      range skew kurtosis       se
## X1 1492665 2.05     9.73 11965.28
```

Notice that the plots and the statistics indicate a slight deviation from the assumption of normality mainly driven by a few irregular cases.

5.1.1 What happens if the assumption of normality is violated?

The estimates and confidence intervals might be less accurate if the assumption of normality is violated. The effect of this depend on the sample size as well. For large sample sizes (N>500) the effect was very minor, while for smaller samples (N round 100) the effect is greater. For example in the study of Lumley, Diehr, Emerson and Chen (2002), the effect of extreme violation of normality (skewness = 8.8, kurtosis = 131) was that the 95% confidence intervals contained the true population mean in only 93.6% of cases instead 95% when N was 500, and 91.3% of the cases when N was 65.

The bottom line is that if the assumption of normality is violated, the confidence intervals and the p values are less reliable, but they are still informative.

Reference:

Lumley, T., Diehr, P., Emerson, S., & Chen, L. (2002). The importance of the normality assumption in large public health data sets. Annual review of public health, 23(1), 151-169.

5.1.2 What to do if the assumption of normality if violated?

1. you can interpret the results more cauciously, such as using 99% CI instead of 95% CI, or $p < 0.01$ as a threshold for significance
2. you can try transforming the predictors and/or the outcome variable to achieve more normally distributed residuals. But if you do this, be mindful when you interpret the unstandardized coefficients that they will be in the transformed units. The same goes for the error terms, so RSS of a model on transformed variables cannot be copared with that of one with untransformed variables. You can look up this data transformation guide at http://abacus.bates.edu/~ganderso/biology/bio270/homework_files/Data_Transformation.pdf (the author of this file is unknown to me, but the contents are accurate and it has nice references to the source of information).
3. if the deviation from normality is driven by a few cases, you might want to try to exclude the outliers. If you are doing formal hypothesis testing, exclusion of variables should not be based on the p-value. Rules for exclusion can be pre-registered, or a sensitivity analysis can be presented, meaning that you conduct the same analysis twice on data including and excluding the problematic cases, and compare the results of the two analysis to see the influence of exclusion of outliers.

In our case, since non-normality is the result of a few extreme cases, we can build a model with these cases excluded to see if this fixes the problem. We axclude cases 186 and 113 here because they showed up as extreme cases according to the Cook's distance, and they also contributed to the deviation from normality according to the QQ plot.

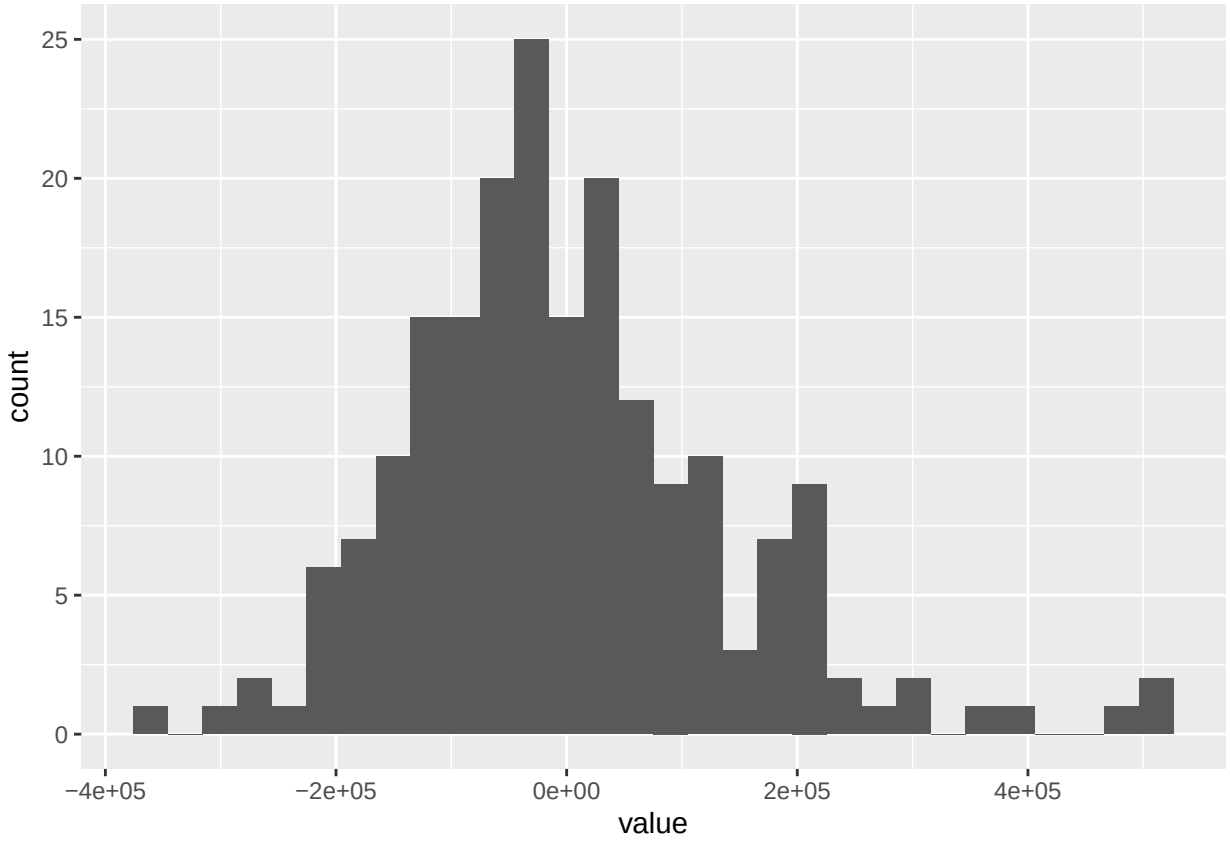
Here we refit the model without these cases two cases and recheck the assumption of normality. Notice that the residuals look much more normal in the model witout the outliers.

```
data_house_nooutliers = data_house %>% slice(-c(186, 113))

mod_house3 = lm(price ~ sqm_living + grade, data = data_house_nooutliers)

# recheck the assumption of normality of residuals
describe(residuals(mod_house3))
```

```
##    vars   n mean       sd    median  trimmed      mad       min      max
## X1    1 198    0 142511.8 -16998.56 -8569.97 121394.9 -347654.3 524234.8
##       range skew kurtosis       se
## X1 871889.1 0.79     1.35 10127.87
```

```
residuals_mod_house3 = enframe(residuals(mod_house3))

residuals_mod_house3 %>% ggplot() + aes(x = value) + geom_histogram()
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



When we compare the two models in a sensitivity analysis, the exclusion of the outlier did not change the statistical inference, the previously significant predictors are still significant, and the overall model F test is significant in both cases. The adjusted R^2 improved considerably because our regression line now fits the remaining data much better. However, model fit should be assessed on new data or a test set as well to get a more accurate picture of true prediction efficiency, because similar outliers might be present in new data as well, and our model will not be very good at predicting these cases.

```
# comparing the models on data with and without the outliers
summary(mod_house2)
```

```
##
## Call:
## lm(formula = price ~ sqm_living + grade, data = data_house)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -371917 -100605  -23119   66886 1120748
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -174389.9    95255.2  -1.831 0.068646 .
## sqm_living     1282.8      266.5   4.813 2.96e-06 ***
## grade         57352.8    16052.8   3.573 0.000444 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 170100 on 197 degrees of freedom
```

```
## Multiple R-squared:  0.358,  Adjusted R-squared:  0.3515
## F-statistic: 54.94 on 2 and 197 DF,  p-value: < 2.2e-16
```

```
summary(mod_house3)
```

```
##
## Call:
## lm(formula = price ~ sqm_living + grade, data = data_house_nooutliers)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -347654  -95907  -16999   73466  524235
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -164126.2    81165.6  -2.022   0.0445 *
## sqm_living     1172.7      225.2   5.208 4.82e-07 ***
## grade         57141.3    13653.2   4.185 4.31e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 143200 on 195 degrees of freedom
## Multiple R-squared:  0.413,  Adjusted R-squared:  0.407
## F-statistic: 68.59 on 2 and 195 DF,  p-value: < 2.2e-16
```

We are going to work with the new model without the outliers in the upcoming tests of the assumptions.

4. you can use bootstrapping to get a robust estimate of the confidence intervals.

5.1.3 Bootstrapping

In case you needed a robust esitmate of the confidence intervals, you can use bootstrapping to get them. Bootstrapping means that we randomly sample from our own observations and rebuild the model on this random sample. We repeat this many times (1000-10000 times), and drow conclusions about the confidence limits based on this multitude of results.

(You will need to run the custom functions in the top of the script for these to work.)

Compare the regular and bottstrapped confidence intervals of the model coefficients.

```
# regular confidence intervals for the model coefficients
confint(mod_house3)
```

```
##                   2.5 %    97.5 %
## (Intercept) -324201.403 -4051.032
## sqm_living      728.662  1616.770
## grade         30214.278 84068.235
```

```
# bootstrapped confidence intervals for the model
# coefficients
confint.boot(mod_house3)
```

```
##               boot 2.5 % boot 97.5 %
## (Intercept) -313465.3171  -16609.124
## sqm_living      713.5829    1696.022
## grade         32107.7570   82942.088
```

```
# regular adjusted R squared
summary(mod_house3)$adj.r.squared
```

```
## [1] 0.406965
```

```
# bootstrapping with 1000 replications
results.boot <- boot(data = data_house, statistic = adjR2_to_boot,
    R = 1000, model = mod_house3)

# get 95% confidence intervals for the adjusted R^2
boot.ci(results.boot, type = "bca", index = 1)
```

```
## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
## Based on 1000 bootstrap replicates
##
## CALL :
## boot.ci(boot.out = results.boot, type = "bca", index = 1)
##
## Intervals :
## Level       BCa
## 95%   ( 0.2330,  0.4833 )
## Calculations and Intervals on Original Scale
```

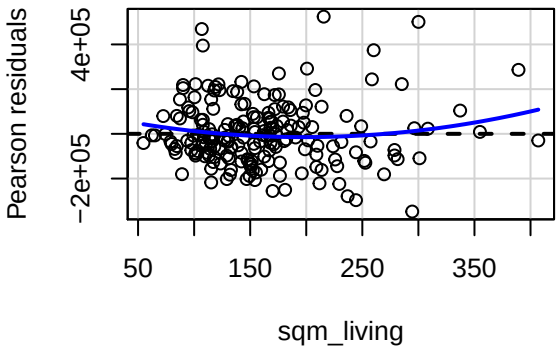
5.2 Linearity

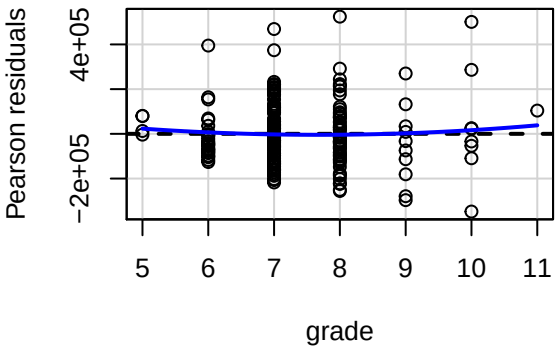
The relationship between the outcome variable and the predictor(s) must be linear.

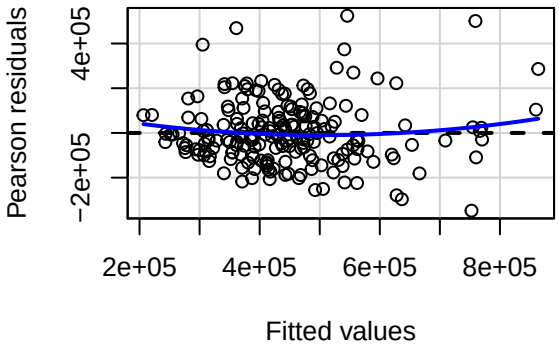
You can use the residualPlots() function for each predictor from the car package to explore linearity. It will return the scatterplot with a spline roughly indicating the relationship of the predictor and the outcome, and you will also see the residual-predicted value plot. In all of these plots you should see roughly flat lines if the assumption of linearity holds true.

Also, this function returns the results of a non-linearity test for each outcome-predictor pair. If the test is significant ($p < 0.05$), there might be a violation of the linearity assumption.

```
mod_house3 %>% residualPlots()
```








```
##            Test stat Pr(>|Test stat|)
## sqm_living    1.6218           0.1065
## grade         0.6485           0.5174
## Tukey test    1.2576           0.2085
```

In our case, even though there is minor curvature visible on the plots, the tests are all non significant, so the linearity assumption seems to hold true for our model.

5.2.1 What happens if the assumption of linearity is violated?

If there is nonlinear relationships among the outcome and the predictors, the predictions of the model may be off, resulting in lower prediction accuracy. Also, the model coefficients will be also unreliable when used for prediction, and even though the stnadardized coefficients and the t-test p-values for some predictors would indicate no significant added predictive value for a predictor, in reality the predictor might still be related to the outcome, just in a non-linear way.

5.2.2 What to do if the assumption of linearity if violated?

If the assumption of linearity is violated, you can try to account for the non-linearity in the relationship by making your model more flexible.

1. you can include a higher order term of the predictor that seems to have a non-linear relationship with the outcome. (See the exercise on special predictors on how to do this.) Usually, including the second and third order term should be enough to address curved relationship. You should avoid adding too much flexibility to reduce overfitting.
2. If higher order terms do not capture the relationship well, you might have to experiment with non-linear regression. You can learn more about it for example from this book: James, G., Witten, D., Hastie,

T., & Tibshirani, R. (2013). An introduction to statistical learning. New York: springer. It is freely available at: http://www-bcf.usc.edu/~gareth/ISL/

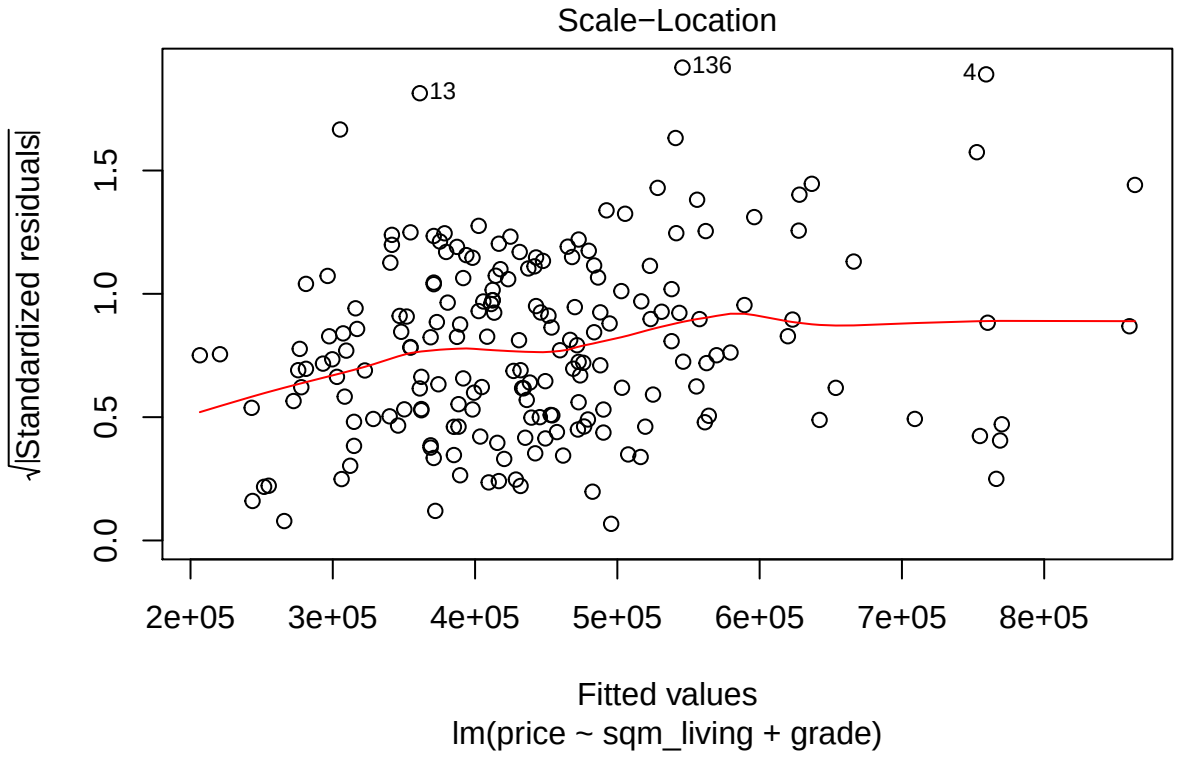
5.3 Homoscedasticty

In regression we asume that the standard deviations of the error terms (residuals) are constant and do not depend on the value of the predictors. So for example the variance of the residuals should be the same for 600 sqft apartments and 3000 sqft apartments.

We can check this assumption by looking at the plot of the standardized residuals and the predicted values, where we should see roughly equal variation at all values of the predicted values. There are also good statistical tests to check this assumption. The Breush-Pagan test is initiated with the bptest() function from the lmtest package, and the NCV test is performed by the ncvTest() funtion from base R (no package needed). A p-value < 0.05 in these tests would indicate a violation of the assumption of homoscedasticity, so it would mean that there is significant heteroscedasticity.

These test indicate that there is significant heteroscedasticity in mod_house3, so we need to investigate and treat this if possible.

```
mod_house3 %>% plot(which = 3)
```




```
mod_house3 %>% ncvTest()  # NCV test
```

```
## Non-constant Variance Score Test
## Variance formula: ~ fitted.values
## Chisquare = 17.01078, Df = 1, p = 3.7168e-05
```

```
mod_house3 %>% bptest()  # Breush-Pagan test
```

```
##
##  studentized Breusch-Pagan test
##
## data:  .
## BP = 10.028, df = 2, p-value = 0.006645
```

5.3.1 What happens if the assumption of homoscedasticity is violated?

If there is heteroscedasticity, the prediction can be off, meaning that the model overall would be less efficient in predicting the outcome. This being said, we can still use the model for prediction, it will be just less good predicting new data.

More importantly, the model coefficients and their confidence intervals are no longer accurate.

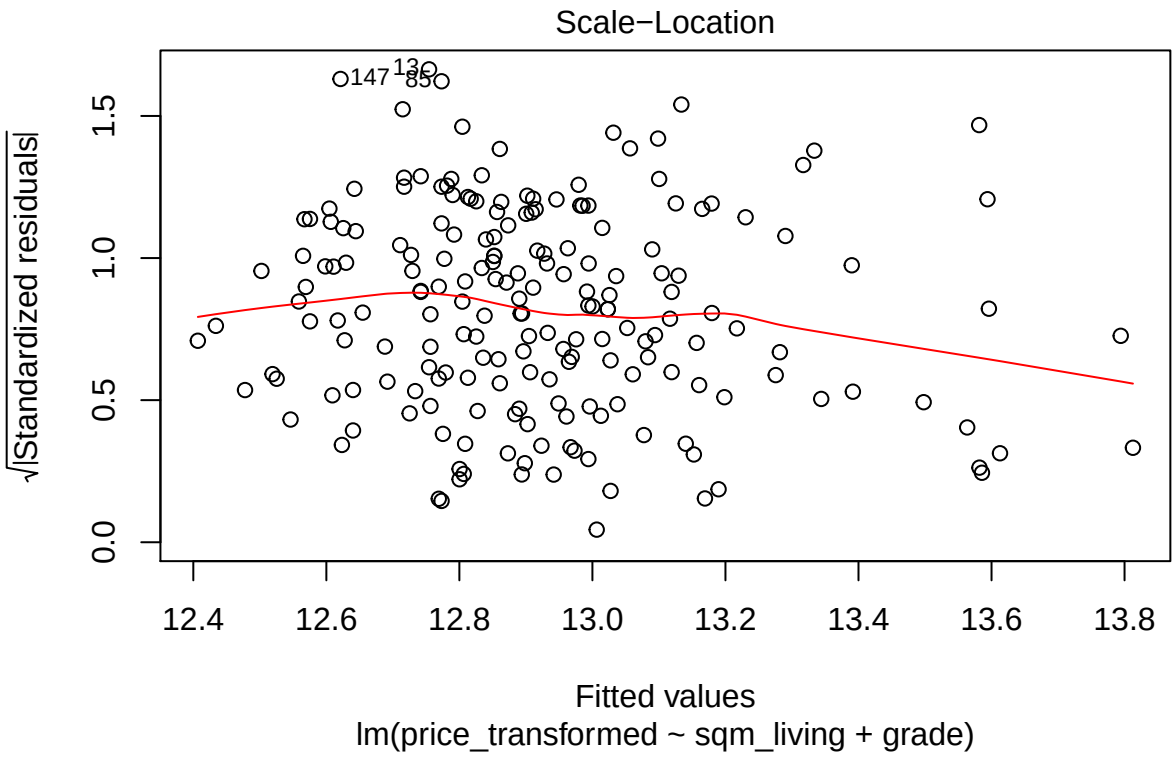
So if we are interested in predicting the outcome, we could still do that with some success even if there is heteroscedasticity. However, interpreting the individual coefficients and our confidence in their added predictive value is no longer possible.

5.3.2 What to do if the assumption of homoscedasticity if violated?

We have a couple of options to deal with heteroscedasticity.

1. **Transformation**. If we are mainly interested in more accurate predictions, we can do some transformation on the outcome and/or the predictors to make them more normally distributed, and thus, homogenize the variability in some parts of the dataset. For example in the example below we apply log() transformation to the outcome, price, and refit the model that way. The tests now show that the assumption is no longer violated. However in this case, we have to keep in mind when we interpret the model coefficient and the predicted (fitted) values that we are no longer predicting price, the predictions are about log(price). So the predicted values will have to be transformed back with the exponential function "exp()" to be interpretable on the original scale of the outcome, and the model coefficents now also have to be interpreted by keeping in mind that they refer to changes in log(price).

```
data_house_nooutliers = data_house_nooutliers %>% mutate(price_transformed = log(price))

mod_house4 = lm(price_transformed ~ sqm_living + grade, data = data_house_nooutliers)

mod_house4 %>% plot(which = 3)
```




```
mod_house4 %>% ncvTest()  # NCV test
```

```
## Non-constant Variance Score Test
## Variance formula: ~ fitted.values
## Chisquare = 0.5274885, Df = 1, p = 0.46766
```

```
mod_house4 %>% bptest()  # Breush-Pagan test
```

```
##
##  studentized Breusch-Pagan test
##
## data:  .
## BP = 0.97249, df = 2, p-value = 0.6149
```

You can get the predictions for price, transformed back from log() to its regular scale with the exp() function

```
exp(predict(mod_house4))
```

2. **Using robust estimators**. If it is important to keep the outcome on its original scale to make the interpretation of the model coefficients more intuitive, we can use robust estimators to establish the Heteroscedasticity-consistent (HC) standard errors and use them to compute confidence intervals, and corrected p-values of the model coefficients. These new statistics are robust to the violation of homoscedasticity, so we call them robust estimates. In the example below we use the Huber-White Sandwich Estimator.

For small samples (N around 50) you might have to use some other method to correct the standard error, for example Bell-McCaffrey estimates. For more details see the paper by Imbens and Kolesar (2016), and their R function at: https://github.com/kolesarm/Robust-Small-Sample-Standard-Errors

Reference: Imbens, G. W., & Kolesar, M. (2016). Robust standard errors in small samples: Some practical advice. Review of Economics and Statistics, 98(4), 701-712.

```
# compute robust SE and p-values
mod_house3_sandwich_test = coeftest(mod_house3, vcov = vcovHC,
    type = "HC")
mod_house3_sandwich_test
```

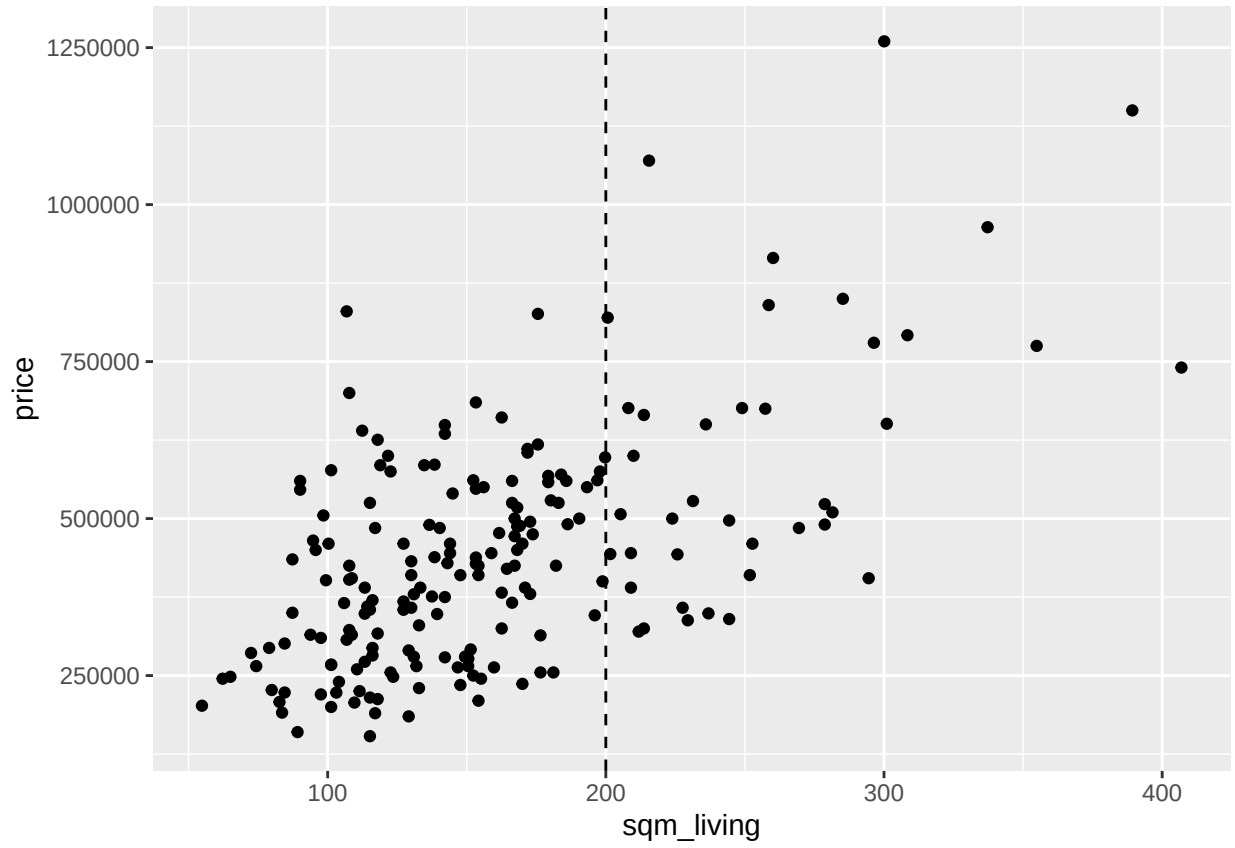
```
##
## t test of coefficients:
##
##               Estimate Std. Error t value  Pr(>|t|)
## (Intercept) -164126.22   81784.56 -2.0068   0.04615 *
## sqm_living     1172.72     228.46  5.1332 6.864e-07 ***
## grade         57141.26   13001.05  4.3951 1.817e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
mod_house3_sandwich_se = unclass(mod_house3_sandwich_test)[,
    2]

# compute robust confidence intervals
CI95_lb_robust = coef(mod_house3) - 1.96 * mod_house3_sandwich_se
CI95_ub_robust = coef(mod_house3) + 1.96 * mod_house3_sandwich_se

cbind(mod_house3_sandwich_test, CI95_lb_robust, CI95_ub_robust)
```

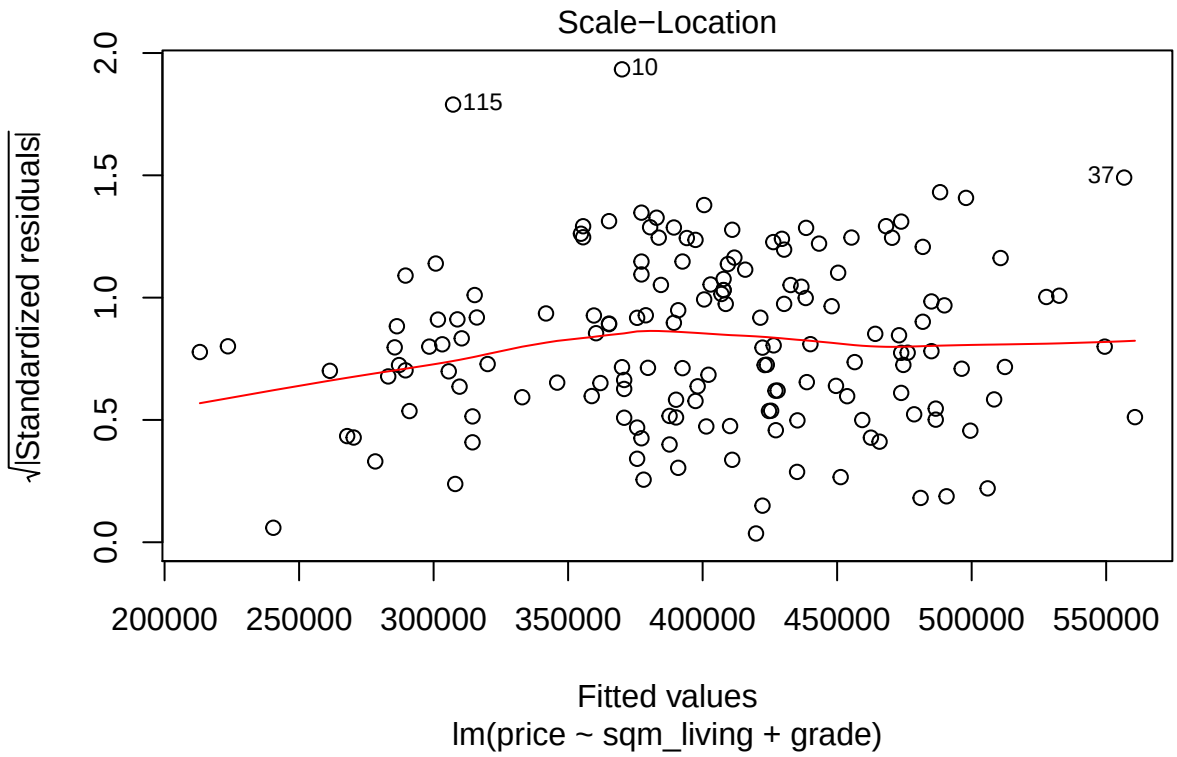
```
##                Estimate Std. Error   t value     Pr(>|t|) CI95_lb_robust
## (Intercept) -164126.218 81784.5648 -2.006812 4.615088e-02   -324423.9650
## sqm_living     1172.716   228.4552  5.133242 6.864072e-07       724.9436
## grade         57141.257 13001.0546  4.395125 1.816739e-05     31659.1895
##             CI95_ub_robust
## (Intercept)      -3828.471
## sqm_living        1620.488
## grade            82623.324
```

3. **Separate models for different parts of the dataset**. Another approach you might take is to visually inspect the plots of the relationships, to see if you can identify clusters with different variance. For example, in our case you might notice that the variances start to be larger at around 200 sqm apartment size. Here, we create two separate datasets for the part of the dataset containing 200 sqm or smaller apartments (data_house_small), and apartments larger than that (data_house_large), and fit separate models on these. So now if you would like to predict the price of apartments 200 sqm or smaller, you would use mod_house3_small, and for larger apartments we would use mod_house3_large. The homogeneity of variance tests indicate no heteroscedasticity in these two models.

```
data_house_nooutliers %>% ggplot() + aes(x = sqm_living, y = price) +
    geom_point() + geom_vline(xintercept = 200, lty = "dashed")
```




```
data_house_small = data_house_nooutliers %>% filter(sqm_living <=
    200)

data_house_large = data_house_nooutliers %>% filter(sqm_living >
    200)

mod_house3_small = lm(price ~ sqm_living + grade, data = data_house_small)
mod_house3_large = lm(price ~ sqm_living + grade, data = data_house_large)


mod_house3_small %>% plot(which = 3)
```



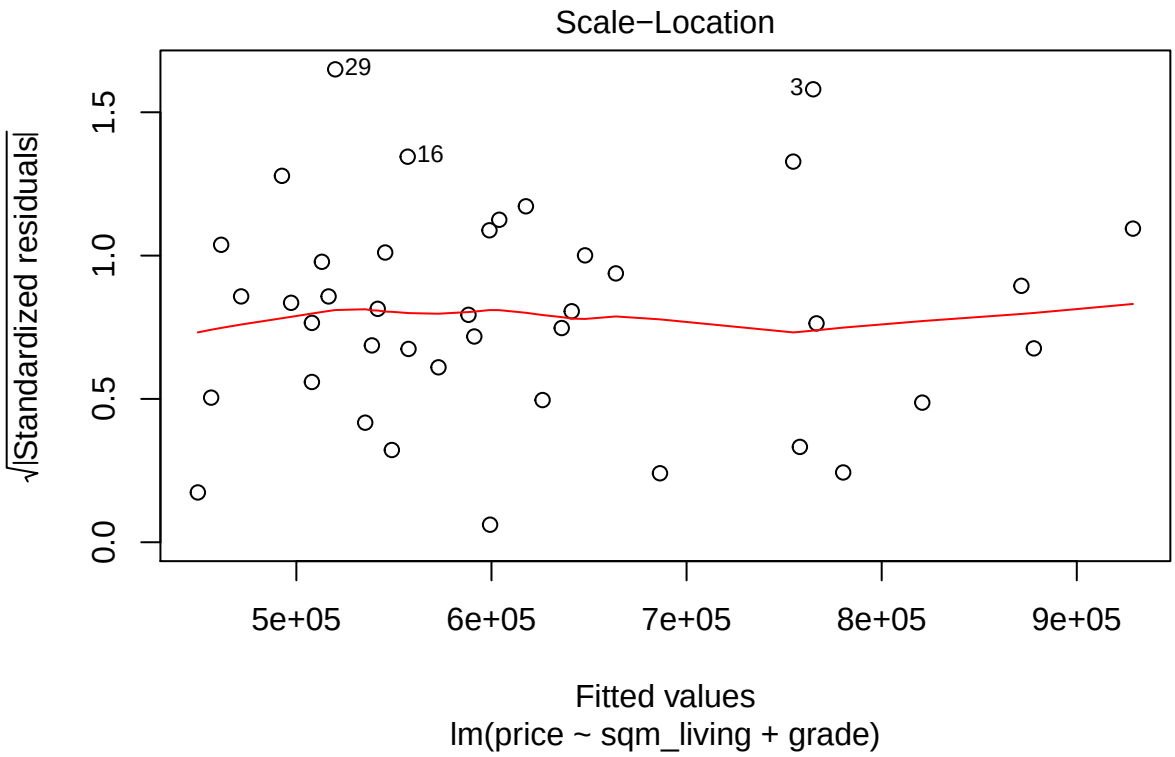

```
mod_house3_small %>% ncvTest()  # NCV test
```

```
## Non-constant Variance Score Test
## Variance formula: ~ fitted.values
## Chisquare = 0.2195388, Df = 1, p = 0.63939
```

```
mod_house3_small %>% bptest()  # Breush-Pagan test
```

```
##
##  studentized Breusch-Pagan test
##
## data:  .
## BP = 0.95326, df = 2, p-value = 0.6209
```

```
mod_house3_large %>% plot(which = 3)
```




```
mod_house3_large %>% ncvTest()  # NCV test
```

```
## Non-constant Variance Score Test
## Variance formula: ~ fitted.values
## Chisquare = 0.005016941, Df = 1, p = 0.94353
```

```
mod_house3_large %>% bptest()  # Breush-Pagan test
```

```
##
##  studentized Breusch-Pagan test
##
## data:  .
## BP = 0.35366, df = 2, p-value = 0.8379
```

5.4 No multicollinearity

We asume that none of the predictors can be linearly determined by the other predictor(s). This usually mean that the predictors should not be too highly correlated with each other, but there are some more subtle cases where even though the predictors are not super highly correlated pair-by-pair, the combination of multiple predictors can still accurately determin another predictor.

So to evaluate whether this assumption holds true, we compute the variance inflation factor for each predictor using the vif() function in the car package.

There is no real consensus about what vif values indicates problematic multicollinearity, some advocate for vif values of 10 or larger are problematic (for example Montgomery and Peck ,1992). However, a more conservative approach is to explore and treat multicollinearity if VIF is above 3 (an approach suggested by Zuur, Ieno, and Elphick, 2010). This is the approach that you should stick to in your home assignment as

well.

Reference: Montgomery, D.C. & Peck, E.A. (1992) Introduction to Linear Regression Analysis. Wiley, New York. Zuur, A. F., Ieno, E. N., & Elphick, C. S. (2010). A protocol for data exploration to avoid common statistical problems. Methods in ecology and evolution, 1(1), 3-14.

```
mod_house3 %>% vif()
```

```
## sqm_living      grade
##   1.849221   1.849221
```

In the example above multicollinearity is not problematic.

5.4.1 What happens in the case of multicollinearity?

In regression we would often like to know the main effect of each predictor, and whether this effect (or in other words, whether the added predictive value of a certain predictor) is statistically significant or not. The model coefficient indicates the size and direction of the effect of increasing the value of the predictor by one point, when all other predictors are held constant. However, if correlation is high among the predictors, there are fewer cases in the dataset where one predictor has high values, while the others are low. This makes it difficult for the model to estimate the independent effect of the predictors that are highly correlated.

The bottom line is, that in the case of multicollinearity, the model coefficients and the t-test of their individual predictive value will become less reliable. Futhermore, the model coefficients will be very unstable, and will show big changes and even sign changes (a positive effect would turn into a negative effect) if we change the model a bit, such as by adding or removing a new predictor.

Fortunately, multicollinearity does not affect prediction accuracy. So if prediction accuracy is our only concern, we might not have to deal with multicollinearity. But if we are interested in the coefficients as well and drawing inference about the individual predictors and their effect and predictive value, we have to address it.

5.4.2 What to do in the case of multicollinearity?

There are two types of multicollinearity:

Structural multicollinearity, where multicollinearity is a result of entering a predictor into the model that is derived from one or more of the other predictors. For example higher order terms (e.g. grade and grade^2), or interaction terms (e.g. long*lat).

Data multicollinearity, where multicollinearity is present in the data itself rather than being an artifact of our model.

To demonstrating the two types of multicollinearity and how to address them, we will build some new models.

5.4.2.1 Example of handling structural multicollinearity

First, lets build a model where in addition to the sqm_living and grade, we also enter geolocation information into the model as predictors.

In the first model we only add the main effect of these predictors (no interaction term). Notice that longitude has a has a negative coefficient indicating that further east we go the lower the price of the apartment which makes sense if you look at the map of King County with Seattle being in the west of the county. Latitude has a positive coefficient, indicating that the further north we go the higher the price. The added predicitive value of latitude is significant in this model. Also notice that there is no multicollinearity in the model according to vif.

```
mod_house_geolocation = lm(price ~ sqm_living + grade + long +
    lat, data = data_house_nooutliers)
summary(mod_house_geolocation)
```

```
##
## Call:
## lm(formula = price ~ sqm_living + grade + long + lat, data = data_house_nooutliers)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -239590  -79244  -14858   55982  483474
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -3.478e+07  6.672e+06  -5.214 4.74e-07 ***
## sqm_living   1.292e+03  1.919e+02   6.730 1.89e-10 ***
## grade        5.267e+04  1.156e+04   4.555 9.29e-06 ***
## long        -7.518e+04  5.262e+04  -1.429    0.155
## lat          5.350e+05  6.257e+04   8.551 3.72e-15 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 120900 on 193 degrees of freedom
## Multiple R-squared:  0.5858, Adjusted R-squared:  0.5772
## F-statistic: 68.25 on 4 and 193 DF,  p-value: < 2.2e-16
```

```
mod_house_geolocation %>% vif()
```

```
## sqm_living      grade       long        lat
##   1.884575   1.860764   1.041486   1.024952
```

Now, we add an interaction term as well (using the * sign instead of +), where we include the interaction of longitude and latitude in the model. Notice that coefficients and the signs undervent a violent shift. Longitude now has a positive coefficient, latiutude has a negative coefficient, and latitud's added predicitve value no longer seems significant. However, the vif values of long, lat and long:lat are extremely high indicating severe multicollinearity. The big change in the results is because of the coefficient's instability due to multicollinearity. Even though the coefficients undervent violent change, the model R^2 only chaged a little bit.

```
mod_house_geolocation_inter = lm(price ~ sqm_living + grade +
    long * lat, data = data_house_nooutliers)

summary(mod_house_geolocation_inter)
```

```
##
## Call:
## lm(formula = price ~ sqm_living + grade + long * lat, data = data_house_nooutliers)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -223574  -75687  -16930   58851  479323
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  4.974e+09  2.656e+09   1.873   0.0626 .
## sqm_living   1.298e+03  1.907e+02   6.809 1.23e-10 ***
## grade        5.436e+04  1.152e+04   4.717 4.59e-06 ***
## long         4.091e+07  2.173e+07   1.882   0.0613 .
## lat         -1.048e+08  5.585e+07  -1.876   0.0621 .
## long:lat    -8.618e+05  4.570e+05  -1.886   0.0608 .
## ---
```

```
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 120100 on 192 degrees of freedom
## Multiple R-squared:  0.5934, Adjusted R-squared:  0.5828
## F-statistic: 56.03 on 5 and 192 DF,  p-value: < 2.2e-16
```

```
mod_house_geolocation_inter %>% vif()
```

```
##   sqm_living        grade         long          lat     long:lat
## 1.885276e+00 1.872074e+00 1.799568e+05 8.275735e+05 1.125795e+06
```

Structural multicollinearity in this case is due to the fact that the interaction term long:lat is derived from both long and lat predictors, so they are now highly correlated.

This can be fixed by standardizing the variables. A good method to use here is "centering", that is, subtracting the mean of the variable from the values of the variable. By doing this, we can still preserv the original scale of the variables, and thus, the coefficients will mean the same thing as before centering. (We could use Z transformation as well, but that would change the interpretation of the coefficients, because the predictors would be transformed to a scale between -1 to +1.)

So lets center longitude and latitude, and re-build the model. Notice that mutlicollinearity is gone, and the coefficients are now back to their original signs and roughly the same effect size as before entering the interaction. Latitude has a significant added predictive value again. But also notice that the R^2 was not at all effected by the removal of multicollinearity! So the prediction efficientcy is the same, but the coefficients are more reliable now and their interpretation is more straightforward.

```
data_house_nooutliers = data_house_nooutliers %>% mutate(long_centered = long -
    mean(long), lat_centered = lat - mean(lat))


mod_house_geolocation_inter_centered = lm(price ~ sqm_living +
    grade + long_centered * lat_centered, data = data_house_nooutliers)

mod_house_geolocation_inter_centered %>% vif()
```

```
##                 sqm_living                      grade
##                   1.885276                   1.872074
##              long_centered               lat_centered
##                   1.058954                   1.049776
## long_centered:lat_centered
##                   1.050759
```

```
summary(mod_house_geolocation_inter_centered)
```

```
##
## Call:
## lm(formula = price ~ sqm_living + grade + long_centered * lat_centered,
##     data = data_house_nooutliers)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -223574  -75687  -16930   58851  479323
##
## Coefficients:
##                             Estimate Std. Error t value Pr(>|t|)
## (Intercept)               -166830.4    68692.4  -2.429   0.0161 *
## sqm_living                   1298.4      190.7   6.809 1.23e-10 ***
## grade                       54356.8    11522.7   4.717 4.59e-06 ***
```

```
## long_centered               -87943.0    52713.8  -1.668   0.0969 .
## lat_centered                516751.6    62905.4   8.215 3.08e-14 ***
## long_centered:lat_centered -861819.8   456978.6  -1.886   0.0608 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 120100 on 192 degrees of freedom
## Multiple R-squared:  0.5934, Adjusted R-squared:  0.5828
## F-statistic: 56.03 on 5 and 192 DF,  p-value: < 2.2e-16
```

5.4.2.2 Example of handling data multicollinearity

Now lets build a model where in addition to the sqm_living and grade, we also use sqm_above as a predictor (the squarefootage of the area of the apartment above ground level).

The vif is now above 3.

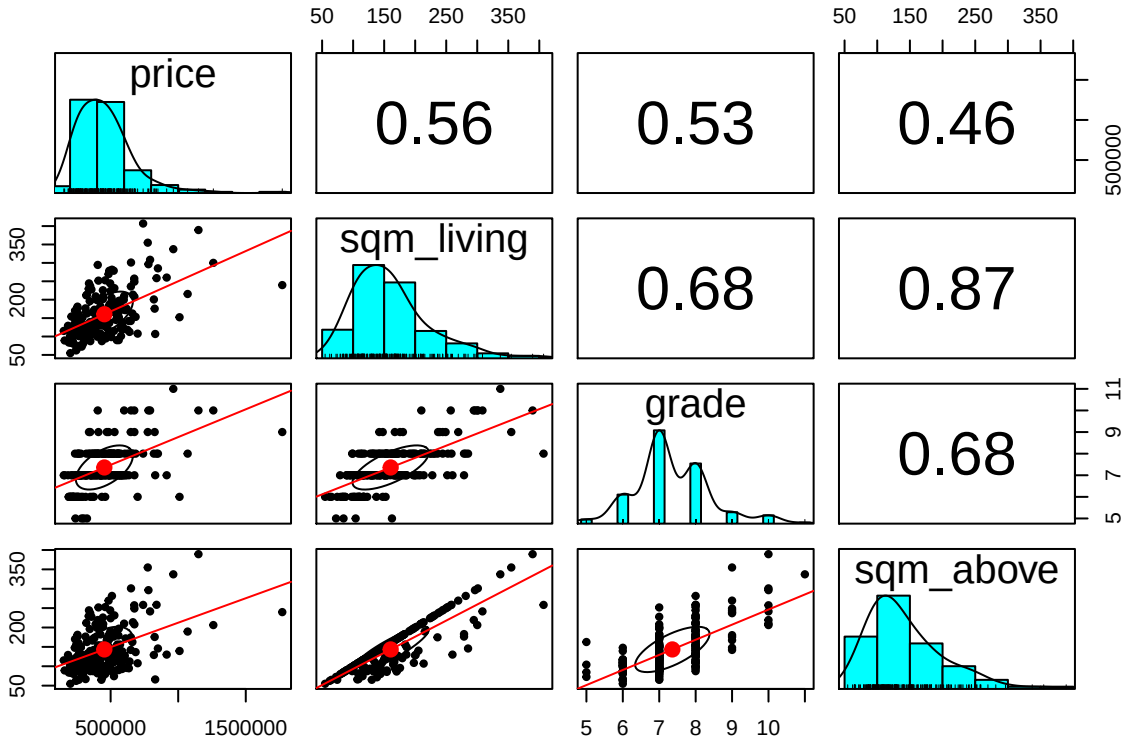
```
mod_house5 = lm(price ~ sqm_living + grade + sqm_above, data = data_house)

vif(mod_house5)
```

```
## sqm_living      grade  sqm_above
##   4.505215   1.972816   4.532763
```

We can explore the reason for this by looking at the correlation matrix of the predictors. The pairs.panels() function returns a series of very useful diagnostic plots where we can not only see the correlation of the variables, but we can also see the distribution of the variables and the scatterplot of their pairvise relationships.

The correlation matrix clearly indicates that the correlation of sqm_living and sqm_above is very high.

```
data_house %>% select(price, sqm_living, grade, sqm_above) %>%
    pairs.panels(col = "red", lm = T)
```



Compare the coefficients in the model summary of the two models. What can you observe? Do the coefficients in the new model make sense?

```
summary(mod_house5)
```

```
##
## Call:
## lm(formula = price ~ sqm_living + grade + sqm_above, data = data_house)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -417412  -90760  -25309   71154 1144030
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -215411.1    95961.5  -2.245   0.0259 *
## sqm_living     2001.5      411.2   4.868 2.32e-06 ***
## grade         66522.7    16386.5   4.060 7.10e-05 ***
## sqm_above      -989.0      434.0  -2.279   0.0238 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 168300 on 196 degrees of freedom
## Multiple R-squared:  0.3746, Adjusted R-squared:  0.365
## F-statistic: 39.13 on 3 and 196 DF,  p-value: < 2.2e-16
```

```r
summary(mod_house3)
```

```
##
## Call:
## lm(formula = price ~ sqm_living + grade, data = data_house_nooutliers)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -347654  -95907  -16999   73466  524235
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -164126.2    81165.6  -2.022   0.0445 *
## sqm_living     1172.7      225.2   5.208 4.82e-07 ***
## grade         57141.3    13653.2   4.185 4.31e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 143200 on 195 degrees of freedom
## Multiple R-squared:  0.413,  Adjusted R-squared:  0.407
## F-statistic: 68.59 on 2 and 195 DF,  p-value: < 2.2e-16
```

```r
names(data_house)
```

```
##  [1] "id"            "date"          "price"         "bedrooms"
##  [5] "bathrooms"     "sqft_living"   "sqft_lot"      "floors"
##  [9] "waterfront"    "view"          "condition"     "grade"
## [13] "sqft_above"    "sqft_basement" "yr_built"      "yr_renovated"
## [17] "zipcode"       "lat"           "long"          "sqft_living15"
## [21] "sqft_lot15"    "has_basement"  "sqm_living"    "sqm_lot"
## [25] "sqm_above"     "sqm_basement"  "sqm_living15"  "sqm_lot15"
```

```r
summary(lm(price ~ sqm_living + sqm_living15 + yr_built, data = data_house))
```

```
##
## Call:
## lm(formula = price ~ sqm_living + sqm_living15 + yr_built, data = data_house)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -333498 -110508  -11684   81129 1113445
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  2603678.8   818948.0   3.179  0.00172 **
## sqm_living      1461.5      287.3   5.087 8.48e-07 ***
## sqm_living15    1142.0      366.7   3.114  0.00212 **
## yr_built       -1308.6      420.9  -3.109  0.00216 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 168200 on 196 degrees of freedom
## Multiple R-squared:  0.3754, Adjusted R-squared:  0.3659
## F-statistic: 39.27 on 3 and 196 DF,  p-value: < 2.2e-16
```

Because of the multicollinearity, we cannot trust the coefficients or the t-tests of the predictors.

You have a number of options to deal with mutlicollinearity:

1. removing highly correlated predictors
2. linearly combining predictors, such as using the mean of the two values for each observation (e.g. for case 3, sqm_living is 1060, sqm_above is 960, so the mean for this case would be 1000). However, it is not clear how we would interpret this value in this particular case of predicting housing prices.
3. you could use some other statistical method entirely (e.g. partial least squares regression or principal components analysis)

Here I suggest using method 1., relaizing that there is hardly any difference in the two variables sqm_living and sqm_above, and when they are different it is due to the apartment having a basement. So we should just settle with using one of these variables, the one that makes more sense for the types of apartments the price of which we want to be able to predict. In this case I personally would keep sqm_living in the model, because I have a theory that the size of the living area is what ultimately influences the price, whether the living area is above or below ground, and the information about the apartment having a basement can be added by using the has_basement variable in a later model. However, those more familiar with the literature of predicting housing price might use previous research results to make this decision. Or the decision might be made based on practical grounds, for example the fact that size of the living area is more accessable information than size of the area above ground, so we would be able to use the model practically in more cases for prediction, if we chose sqm_living as a predictor.

Notice that I did not use model comparison to make a decision about which varialbe to exclude! This decision, like all model selection decisions, should be made based on theoretical grounds, or based on previous research results or practical issues.

You can find some additional resources about multicollinearity on these links:

https://statisticalhorizons.com/multicollinearity

http://blog.minitab.com/blog/understanding-statistics/handling-multicollinearity-in-regression-analysis

http://statisticsbyjim.com/regression/multicollinearity-in-regression-analysis/

_____Practice_____*

Using the techniques that you learned today, run model diagnostics on a new linear model where you predict price by “sqm_living” “sqm_living15”, and “yr_built”" as predictors.

_____*