Exercises

Natali Tckvitishvili

2024-08-18

Applied Statistics in R

Natali Tckvitishvili

Load libraries

```
#install.packages("extraDistr")
#install.packages('tinytex')
#install.packages('ggpubr')
#install.packages('tidyverse')
#install.packages('glmnet')
#install.packages('astsa')
#install.packages('surveillance')
#install.packages('JoSAE')

library(tidyverse)
library(ggplot2)
library(stats)
library(ggpubr)
library(extraDistr)
library(tinytex)
library(glmnet)
library(astsa)
library(surveillance)
library(JoSAE)
```

Exercise 1

a. load data & add new variable - good

```
wine <- read.csv("winequality-white.csv", sep = ";")
wine <- mutate(wine, good = ifelse(quality > 5, 1, 0))

head(wine)
```

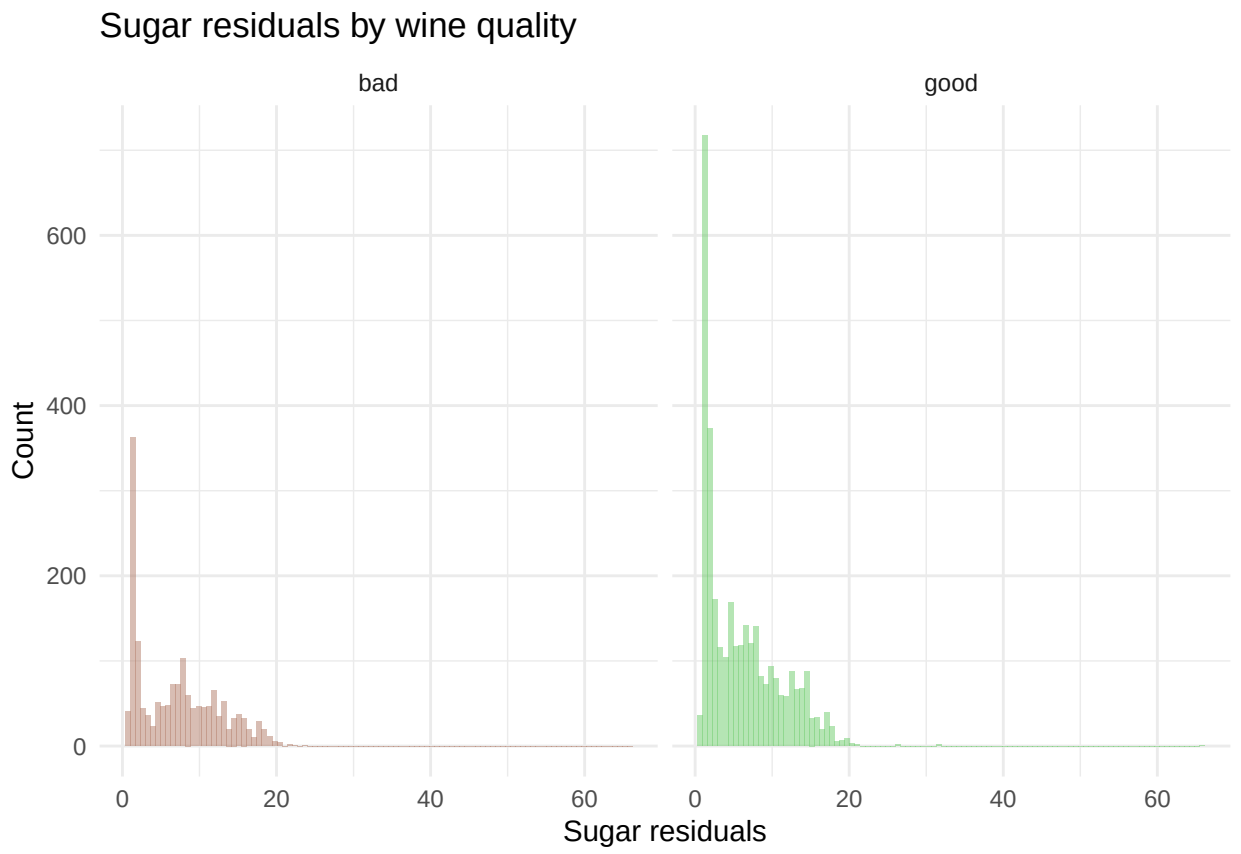
b. residual.sugar analysis

First I'd specify the analysed variable to add more flexibility to the further analysis, when we'll need to make the same calculation for another variable.

```
analysed_variable <- wine$residual.sugar
wine$analysed_variable <- analysed_variable
```

- histograms for good and bad quality wines

```
good_labels <- c("0" = "bad", "1" = "good")

ggplot(wine, aes(x = analysed_variable, fill = as.factor(good))) +
  geom_histogram(position = "identity", alpha = .5, bins = 100) +
  scale_fill_manual(values=c("#b37d69", "#6dcc6b")) +
  facet_wrap(vars(good), labeller=labeller(good = good_labels)) +
  theme_minimal() +
  theme(legend.position = "none") +
  labs (
    x = "Sugar residuals",
    y = "Count",
    title = "Sugar residuals by wine quality"
  )
```



According to the graphs, both bad and good quality wines have sugar residuals near zero, however, for good wines this number is higher than for bad ones. Moreover, sugar residuals of good quality wines have a smoother decrease in frequency, most of them have less sugar. Therefore, we can assume that sugar residuals may have negative correlation with wine quality.

- summary statistics

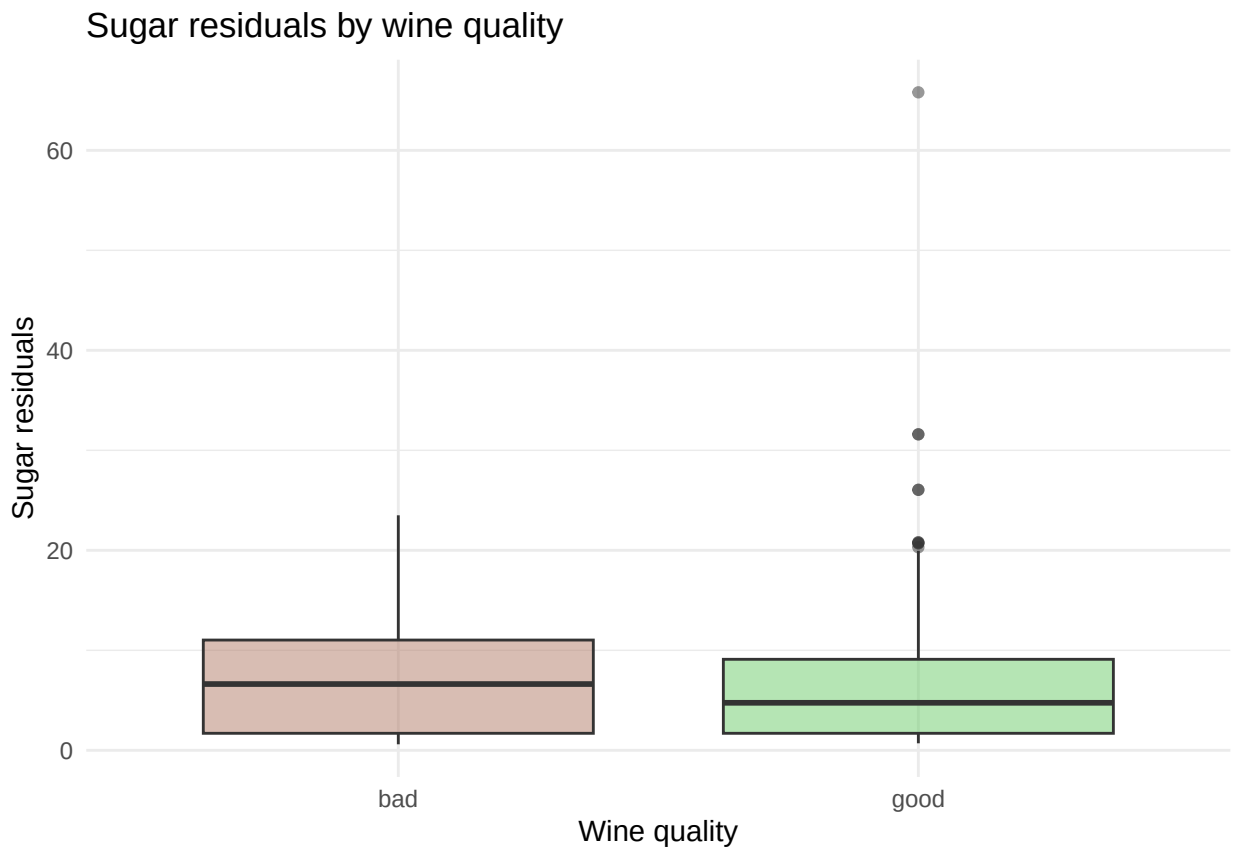
```
summary <- wine %>%
  group_by(good) %>%
  summarise(
    n = n(),
    mean = mean(analysed_variable),
    median = median(analysed_variable),
    sd = sd(analysed_variable),
    iqr = IQR(analysed_variable),
    max = max(analysed_variable),
    min = min(analysed_variable)
  )
data.frame(summary)
```

```
##   good    n     mean median       sd   iqr  max min
## 1    0 1640 7.054451  6.625 5.283594 9.325 23.5 0.6
## 2    1 3258 6.057658  4.750 4.929353 7.400 65.8 0.7
```

For the "bad" quality wines both mean and median of the sugar residuals are higher than for the "good" wines, which probably (I say probably here as we haven't checked the significance of this difference yet) means that bad wines on average contain more sugar than good ones. They also have a higher variance and range between the values which may mean that the variety of the bad wines is bigger than of the good ones. What is interesting, there is an observation of a good wine with 65.8 sugar residuals which is a huge number compared to the mean and median. This might be an outlier, we'll see if that's true drawing a boxplot. Additionally, for good wines the difference between the mean and median is quite big, so we can assume that there are more outliers that impact the mean.

- boxplots

```
ggplot(wine, aes(x = as.factor(good), y = analysed_variable, fill = as.factor(good))) +
    geom_boxplot(alpha = .5) +
    scale_fill_manual(values=c("#b37d69", "#6dcc6b")) +
    scale_x_discrete(labels = good_labels) +
    theme_minimal() +
    theme(legend.position = "none") +
    labs (
      x = "Wine quality",
      y = "Sugar residuals",
      title = "Sugar residuals by wine quality"
    )
```

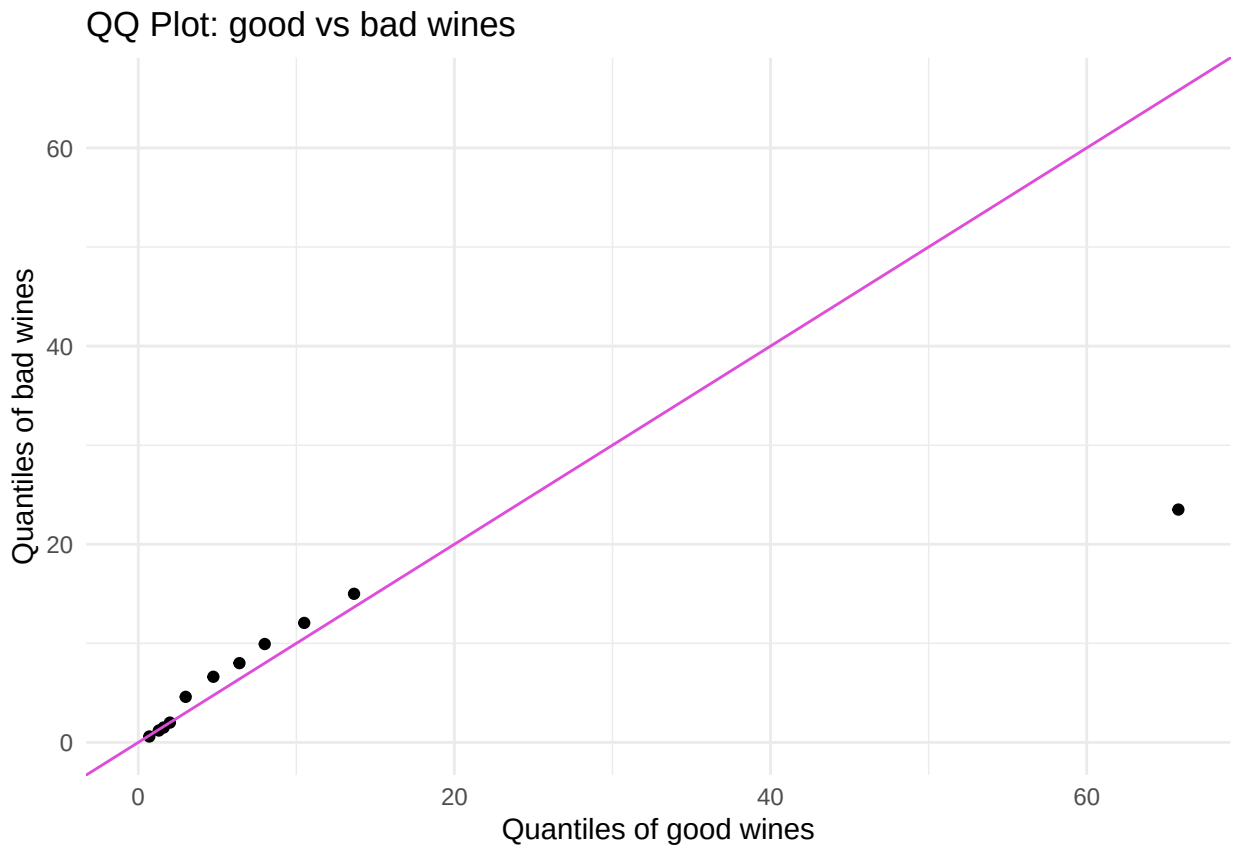


As stated above, the good wine with 65.8 sugar residuals must be an outlier, accoring to the boxplots. There are two more outliers, and all of them impact the mean. Assuming that better wines on average have less sugar, these observations might be a quality estimation error / human factor or there are sugary wines which are considered good in the modern somelier society.

- QQ plot to compare samples

```
good_wine <- wine %>%
  filter(good == 1)

bad_wine <- wine %>%
  filter(good == 0)

quantiles <- seq(0, 1, 0.1)
good_quantiles <- quantile(good_wine$analysed_variable, quantiles)
bad_quantiles <- quantile(bad_wine$analysed_variable, quantiles)

qq_wine <- data.frame(good_quantiles, bad_quantiles)

ggplot(qq_wine, aes(x = good_quantiles, y = bad_quantiles)) +
    geom_point() +
    geom_abline(slope = 1, intercept = 0, color = "#de4dd9") +
    xlim(0, max(good_quantiles, bad_quantiles)) +
    ylim(0, max(good_quantiles, bad_quantiles)) +
    theme_minimal() +
    labs(title = "QQ Plot: good vs bad wines",
```

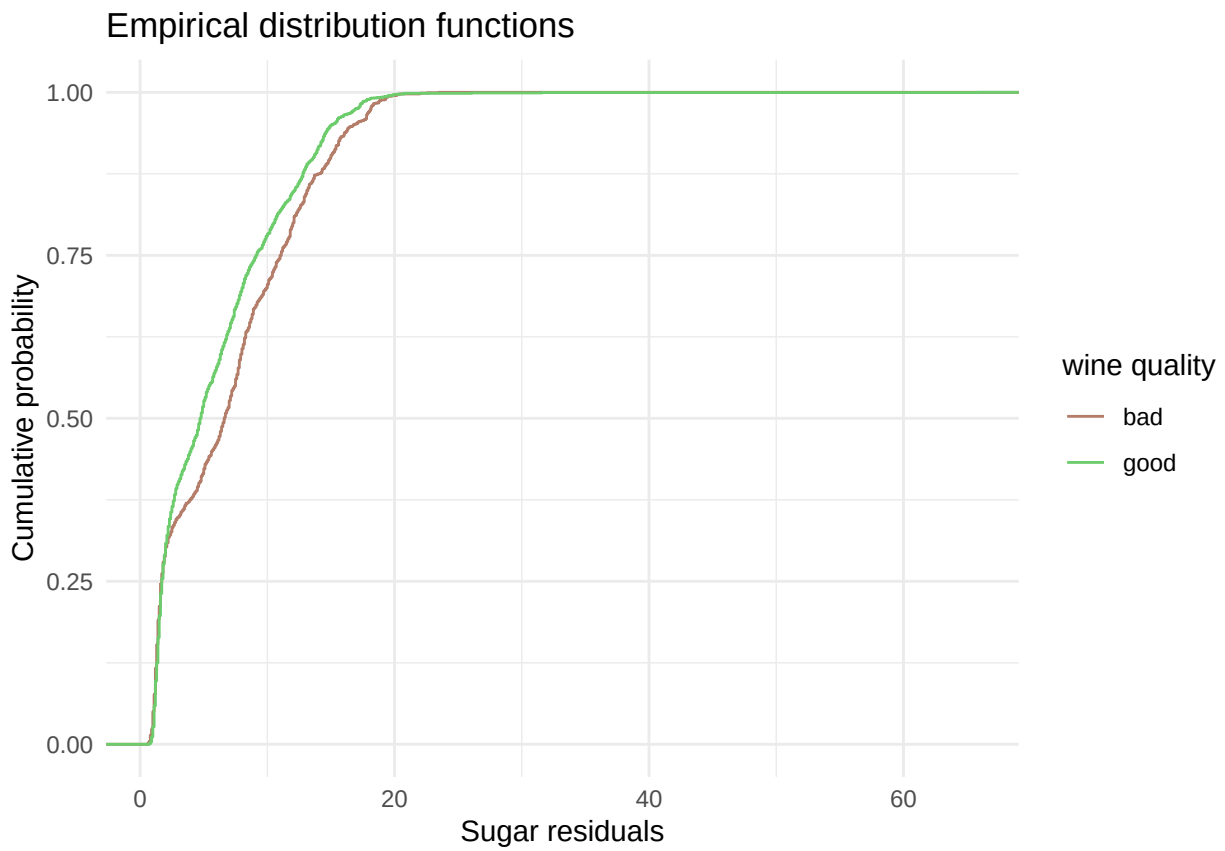
```
        x = "Quantiles of good wines",
        y = "Quantiles of bad wines")
```



The distribution of both samples seems to be similar and right-skewed (this can also be seen on the histograms above). We also see here the outlier.

- Empirical distribution functions

```
ggplot(wine, aes(x = analysed_variable, color = as.factor(good))) +
  stat_ecdf(geom = "step") +
  scale_color_manual(values=c("#b37d69", "#6dcc6b"), labels=good_labels, name="wine quality") +
  theme_minimal() +
  labs (
    x = "Sugar residuals",
    y = "Cumulative probability",
    title = "Empirical distribution functions"
  )
```



The distribution of sugar residuals in good wines is more concentrated around lower values than bad wines. Moreover, values are spread out (graphs are smooth).

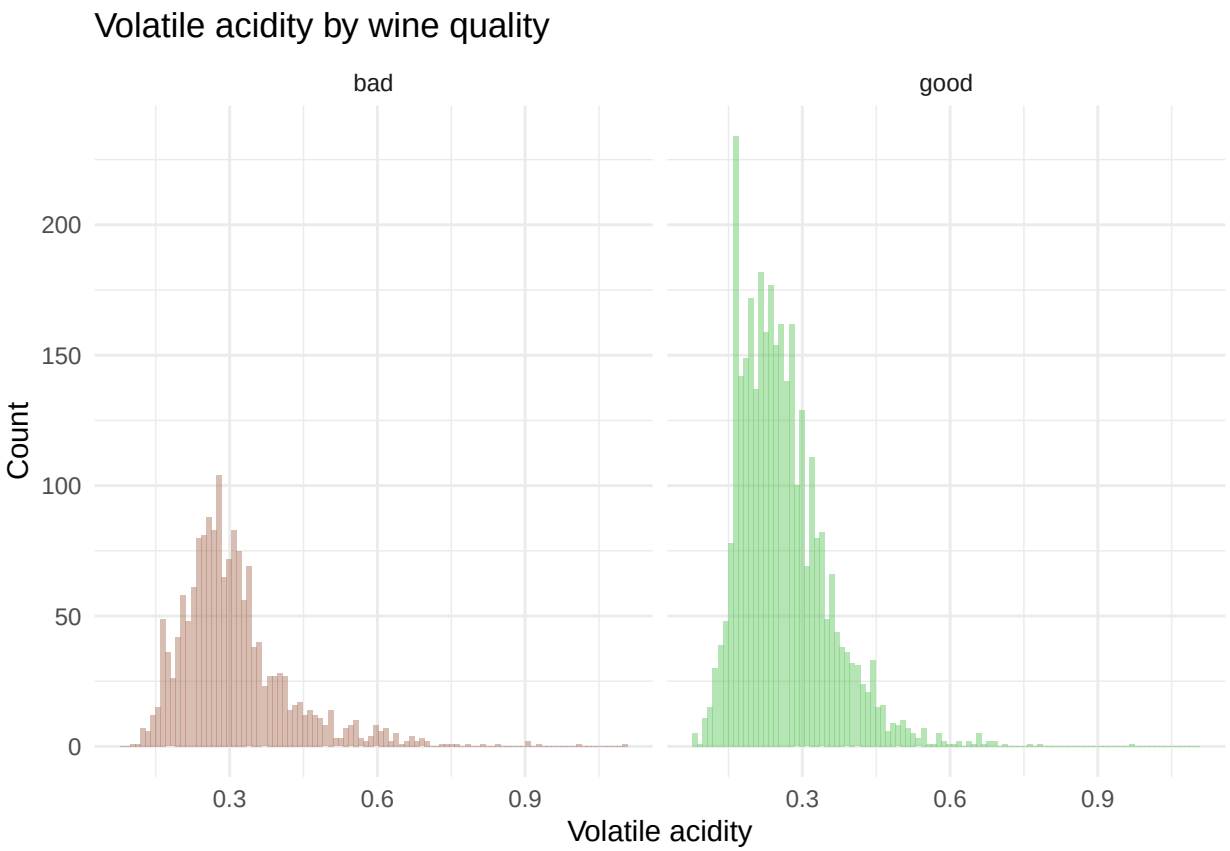
All of those graphs and summary statistics show the same: good wines in general have lower sugar residuals than bad ones.

c. volatile.acidity

```
analysed_variable <- wine$volatile.acidity
wine$analysed_variable <- analysed_variable
```

- histograms for good and bad quality wines

```
ggplot(wine, aes(x = analysed_variable, fill = as.factor(good))) +
  geom_histogram(position = "identity", alpha = .5, bins = 100) +
  scale_fill_manual(values=c("#b37d69", "#6dcc6b")) +
  facet_wrap(vars(good), labeller=labeller(good = good_labels)) +
  theme_minimal() +
  theme(legend.position = "none") +
  labs (
    x = "Volatile acidity",
    y = "Count",
    title = "Volatile acidity by wine quality"
  )
```



It can be said that means of volatile acidity for both good and bad wines do not seem do be significantly different, however, for good wines it may be a little less than for the bad ones. Both distributions are a little right-skewed as well.

- summary statistics

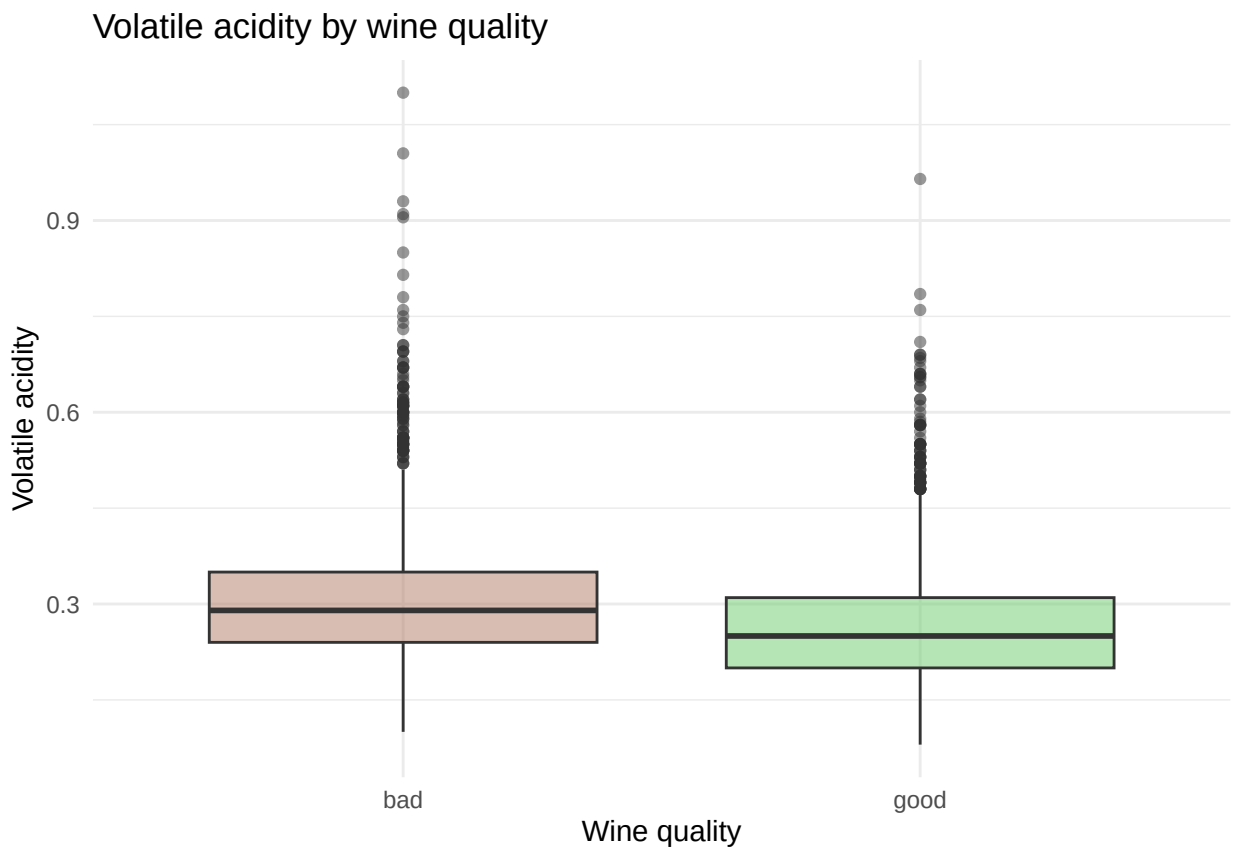
```
summary <- wine %>%
  group_by(good) %>%
  summarise(
    n = n(),
    mean = mean(analysed_variable),
    median = median(analysed_variable),
    sd = sd(analysed_variable),
    iqr = IQR(analysed_variable),
    max = max(analysed_variable),
    min = min(analysed_variable)
  )
data.frame(summary)
```

```
##   good    n      mean median        sd  iqr   max  min
## 1    0 1640 0.3102652   0.29 0.1125479 0.11 1.100 0.10
## 2    1 3258 0.2621209   0.25 0.0901360 0.11 0.965 0.08
```

Similarly to sugar residual, for the "bad" quality wines both mean and median of the volatile acidity are higher than for the "good" wines, although difference is not that huge.

- boxplots

```
ggplot(wine, aes(x = as.factor(good), y = analysed_variable, fill = as.factor(good))) +
    geom_boxplot(alpha = .5) +
    scale_fill_manual(values=c("#b37d69", "#6dcc6b")) +
    scale_x_discrete(labels = good_labels) +
    theme_minimal() +
    theme(legend.position = "none") +
    labs (
      x = "Wine quality",
      y = "Volatile acidity",
      title = "Volatile acidity by wine quality"
    )
```

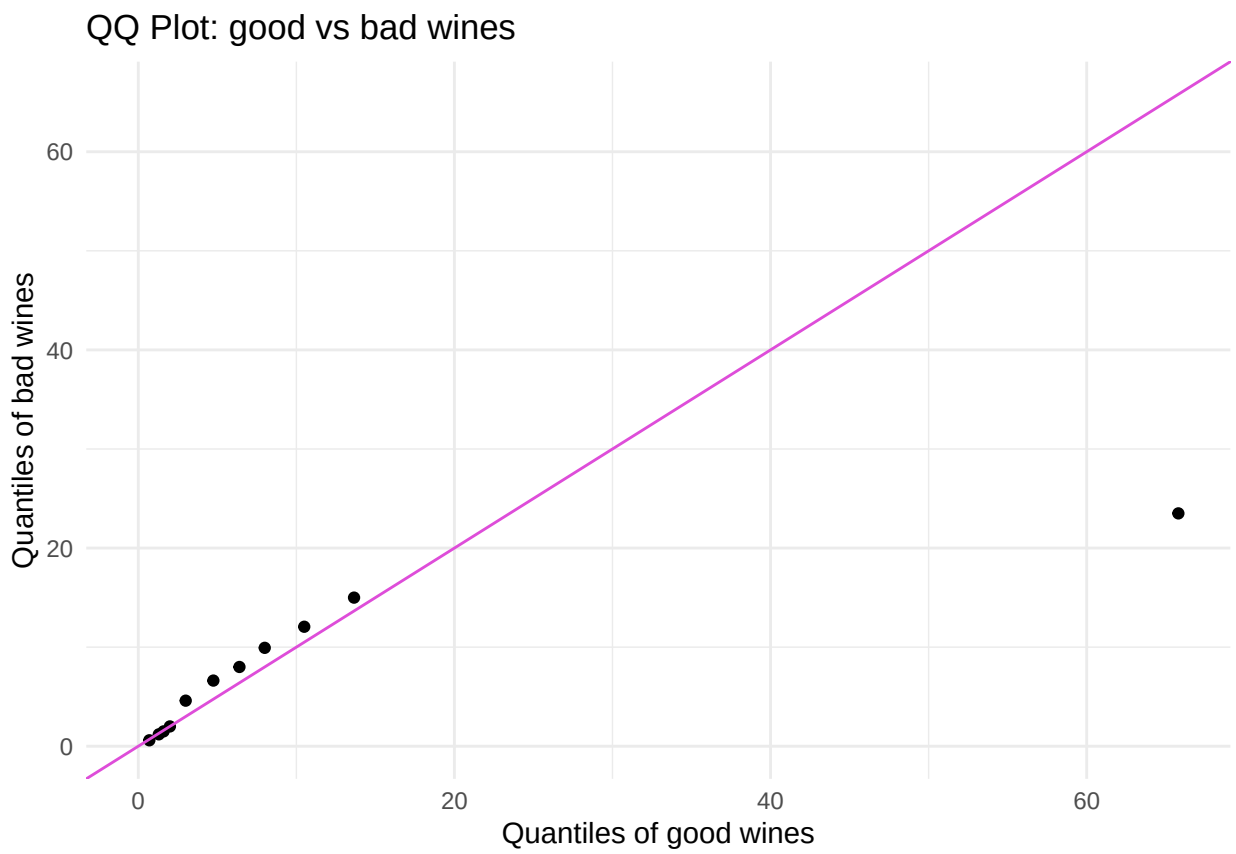


Looks like we have much more outliers here than in sugar residuals. In general, bad wines seem to have more acids (boxplot is located higher).

- QQ plot to compare samples

```
good_quantiles <- quantile(good_wine$analysed_variable, quantiles)
bad_quantiles <- quantile(bad_wine$analysed_variable, quantiles)

qq_wine <- data.frame(good_quantiles, bad_quantiles)

ggplot(qq_wine, aes(x = good_quantiles, y = bad_quantiles)) +
```

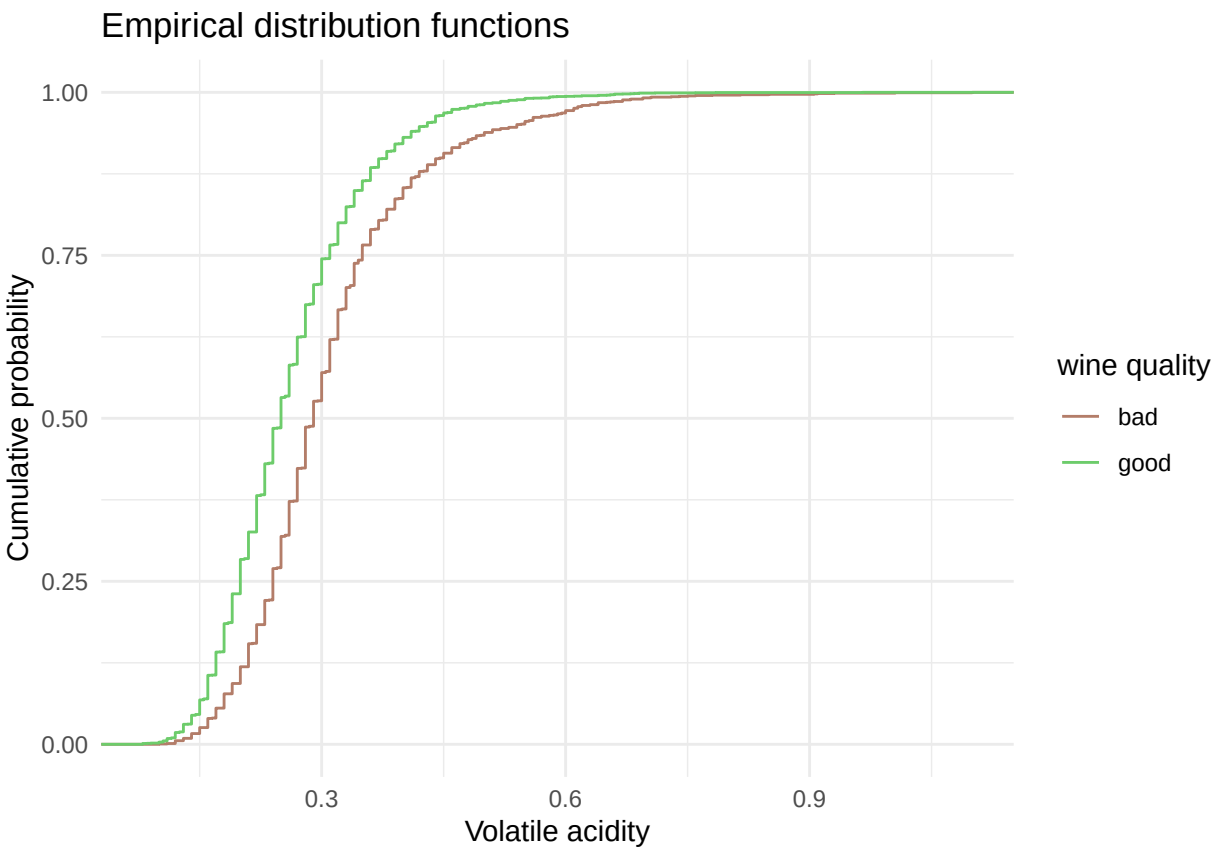
```
    geom_point() +
    geom_abline(slope = 1, intercept = 0, color = "#de4dd9") +
    xlim(0, max(good_quantiles, bad_quantiles)) +
    ylim(0, max(good_quantiles, bad_quantiles)) +
    theme_minimal() +
    labs(title = "QQ Plot: good vs bad wines",
         x = "Quantiles of good wines",
         y = "Quantiles of bad wines")
```



The distribution of both samples seems to be similar but with a difference in scal (variance) as dots do not fall on the y = x line, but still form the straight line.

- Empirical distribution functions

```
ggplot(wine, aes(x = analysed_variable, color = as.factor(good))) +
  stat_ecdf(geom = "step") +
  scale_color_manual(values=c("#b37d69", "#6dcc6b"), labels=good_labels, name="wine quality") +
  theme_minimal() +
  labs (
    x = "Volatile acidity",
    y = "Cumulative probability",
    title = "Empirical distribution functions"
  )
```



Good wines have generally lower values that bad ones; steepness demonstrates that a large number of observations is concentrated within a small range of values. Generally, all graphs incicate that good wines tend to have lower volatile acidity than bad wines.

Exercise 2

```
analysed_variable <- wine$pH
wine$analysed_variable <- analysed_variable
```

a. histogram

mean and standard deviation for plotting normal density

```
mean_pH <- mean(wine$analysed_variable)
sd_pH <- sd(wine$analysed_variable)

mean_pH
```

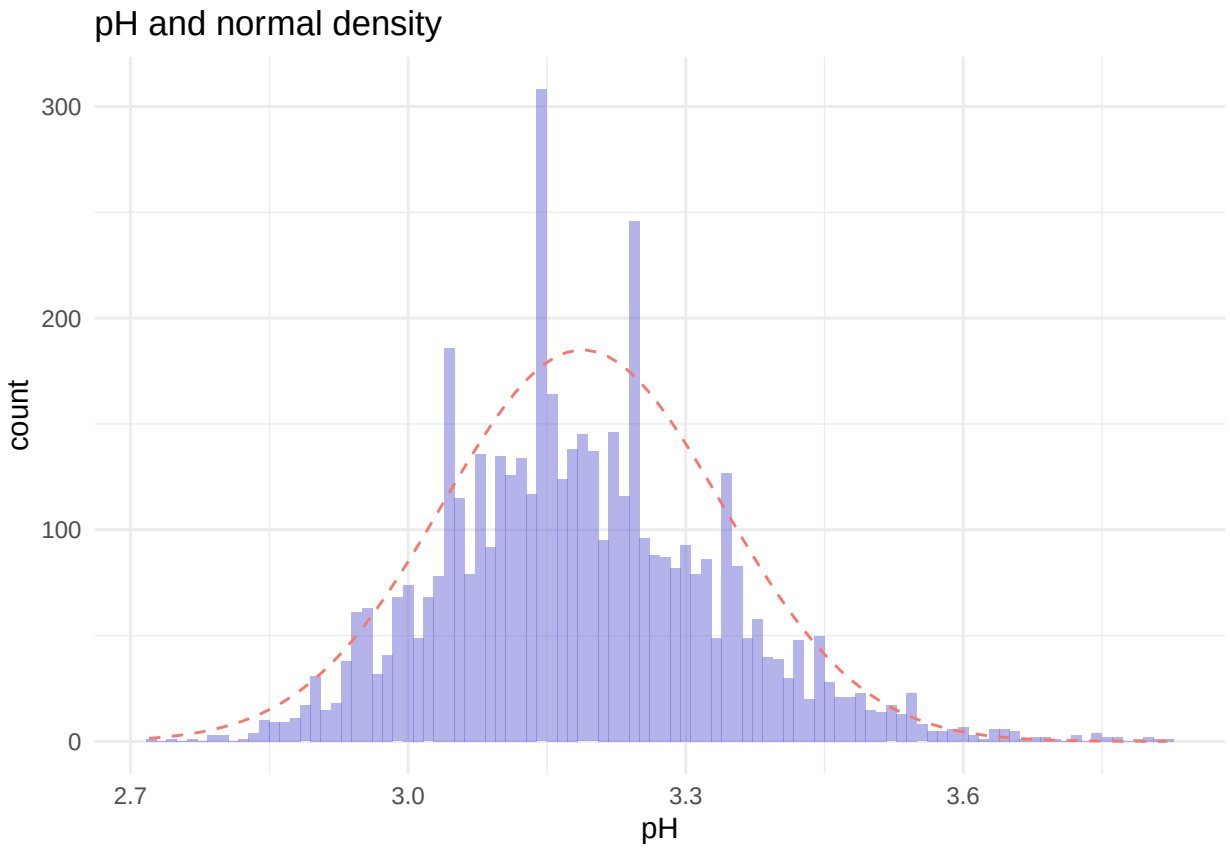
```
## [1] 3.188267
```

```
sd_pH
```

```
## [1] 0.1510006
```

- all wines

```
ggplot(wine, aes(x = analysed_variable)) +
  geom_histogram(position = "identity", alpha = .5, bins = 100, fill="#6a6ad9") +
  stat_function(fun = function(x) # scale normal density to be seen
    dnorm(x, mean = mean_pH, sd = sd_pH) * 70, aes(color = "#d27786"), linetype = "dashed") +
  theme_minimal() +
  theme(legend.position = "none") +
  labs (
    x = "pH",
    y = "count",
    title = "pH and normal density"
  )
```



- good and bad wines

```
good_wine <- wine %>%
  filter(good == 1)

bad_wine <- wine %>%
  filter(good == 0)

mean_pH_good <- mean(good_wine$analysed_variable)
sd_pH_good <- sd(good_wine$analysed_variable)
```

```r
mean_pH_bad <- mean(bad_wine$analysed_variable)
sd_pH_bad <- sd(bad_wine$analysed_variable)

mean_pH_good
```

```
## [1] 3.197231
```

```r
sd_pH_good
```

```
## [1] 0.1535172
```

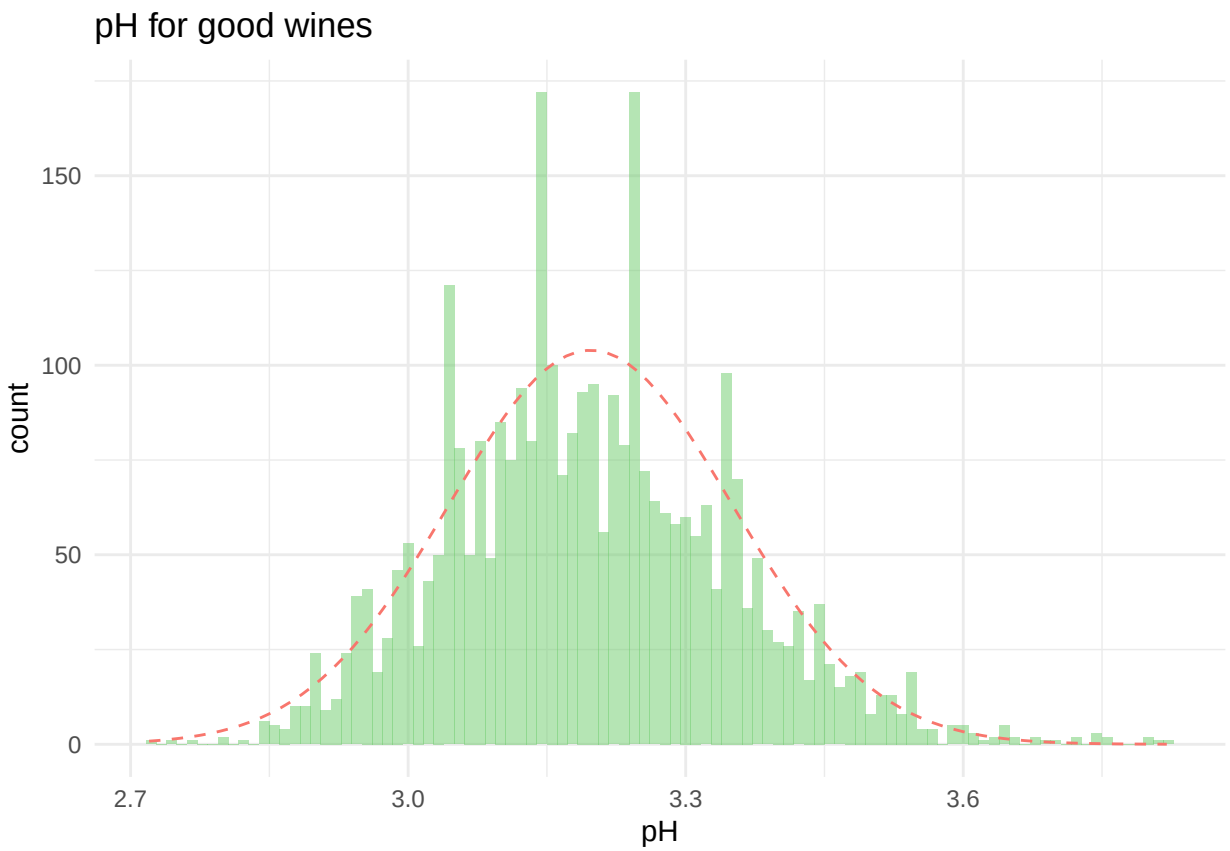
```r
mean_pH_bad
```

```
## [1] 3.170457
```

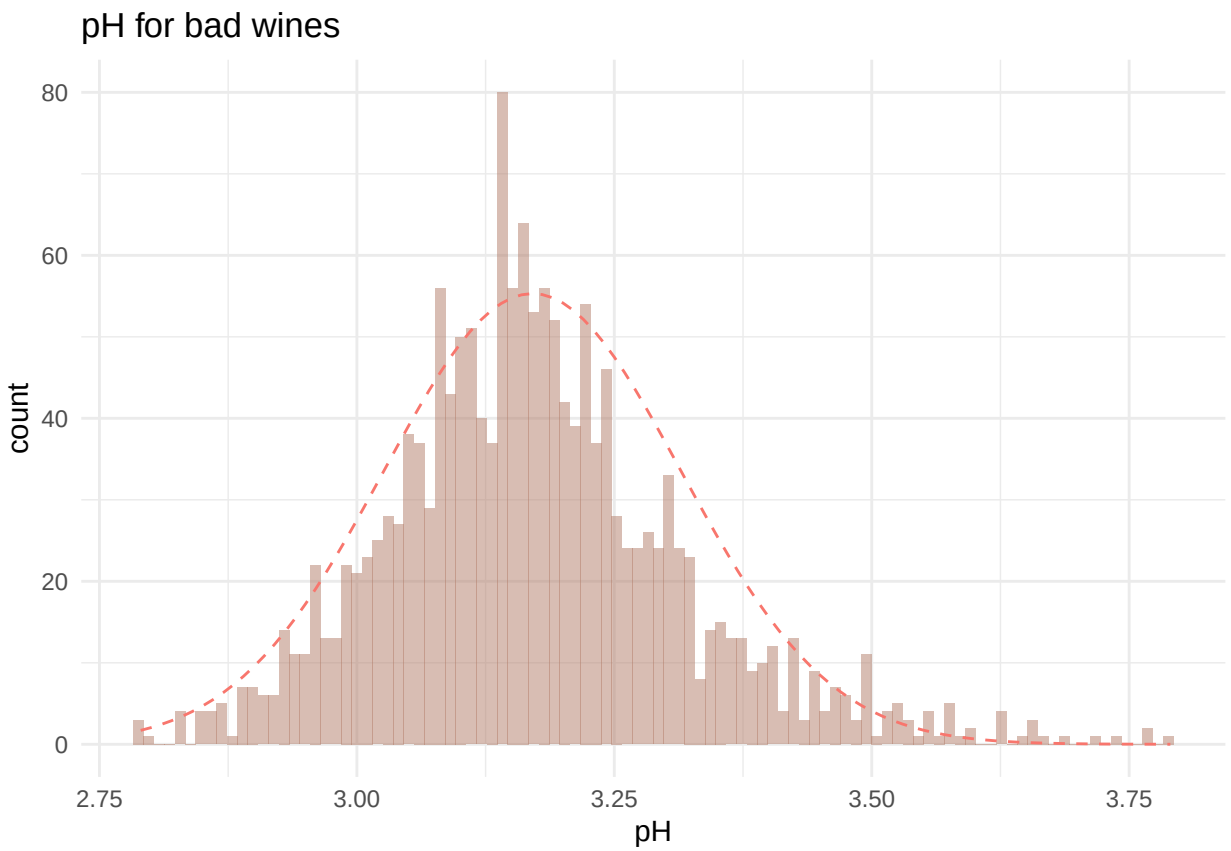
```r
sd_pH_bad
```

```
## [1] 0.1442744
```

```r
ggplot(good_wine, aes(x = analysed_variable)) +
  geom_histogram(position = "identity", alpha = .5, bins = 100, fill="#6dcc6b") +
  stat_function(fun = function(x)
    dnorm(x, mean = mean_pH_good, sd = sd_pH_good) * 40, aes(color = "#d27786"), linetype = "dashed") +
  theme_minimal() +
  theme(legend.position = "none") +
  labs (
    x = "pH",
    y = "count",
    title = "pH for good wines"
  )
```




```
ggplot(bad_wine, aes(x = analysed_variable)) +
  geom_histogram(position = "identity", alpha = .5, bins = 100, fill="#b37d69") +
  stat_function(fun = function(x)
    dnorm(x, mean = mean_pH_bad, sd = sd_pH_bad) * 20, aes(color = "#d27786"), linetype = "dashed") +
  theme_minimal() +
  theme(legend.position = "none") +
  labs (
    x = "pH",
    y = "count",
    title = "pH for bad wines"
  )
```



It can be said that pH follows the normal distribution for bad wines, but for good wines and wines in total the distribution seems to be bimodal with two peaks.

b. QQ plots

- all wines

```
qq_all <- ggplot(wine, aes(sample = pH)) +
  stat_qq() +
  stat_qq_line() +
  theme_minimal() +
  labs (
    x = "theoretical quantiles",
    y = "empirical quantiles"
  )
```

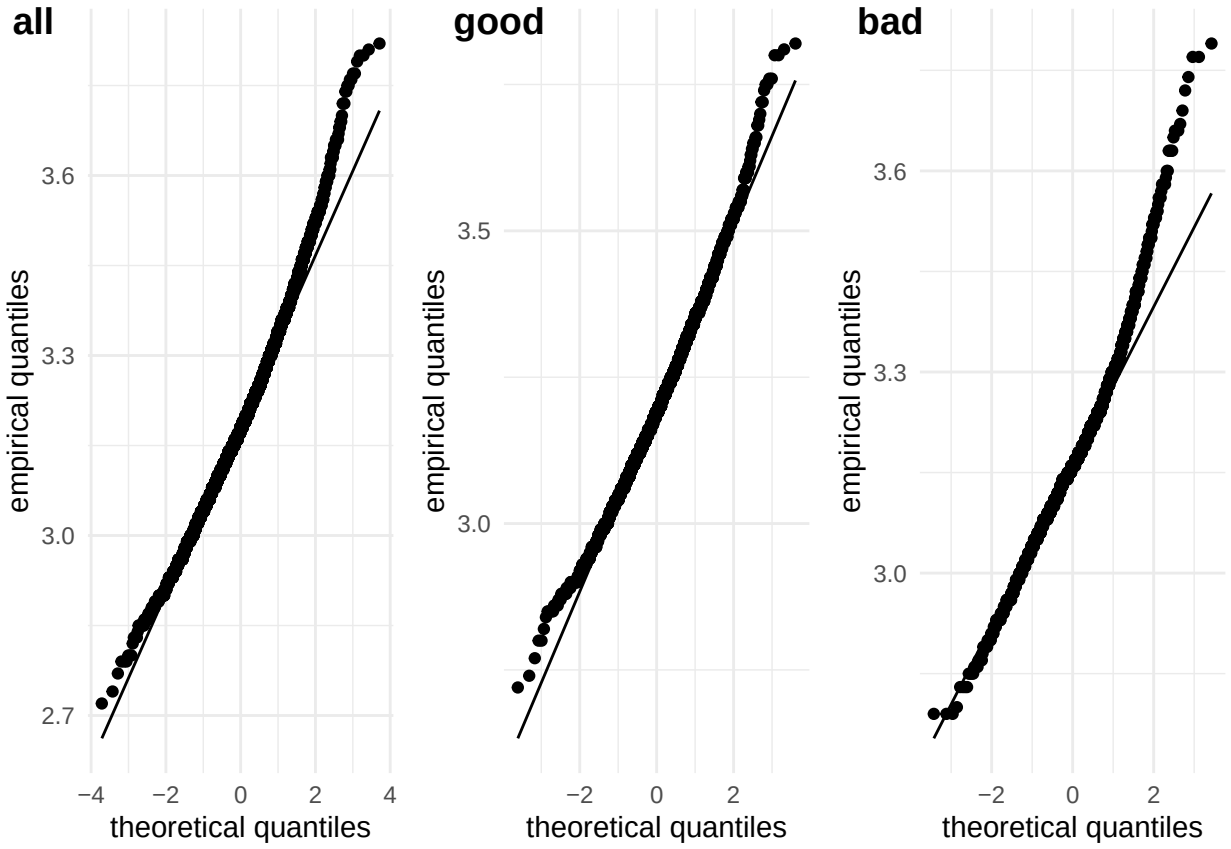
- good wines

```
qq_good <- ggplot(good_wine, aes(sample = pH)) +
  stat_qq() +
  stat_qq_line() +
  theme_minimal() +
  labs (
    x = "theoretical quantiles",
    y = "empirical quantiles"
  )
```

- bad wines

```
qq_bad <- ggplot(bad_wine, aes(sample = pH)) +
  stat_qq() +
  stat_qq_line() +
  theme_minimal() +
  labs (
    x = "theoretical quantiles",
    y = "empirical quantiles"
  )
```

```
ggarrange(qq_all, qq_good, qq_bad, labels = c("all", "good", "bad"), ncol = 3, nrow = 1)
```



b. PP plots

- all wines

```
pp_all <- ggplot(wine, aes(sample = pH)) +
  stat_qq(distribution = qnorm, dparams = list(mean = mean_pH, sd = sd_pH)) +
  stat_qq_line(distribution = qnorm, dparams = list(mean = mean_pH, sd = sd_pH)) +
  theme_minimal() +
  labs (
    x = "theoretical probabilities",
    y = "empirical probabilities"
  )
```

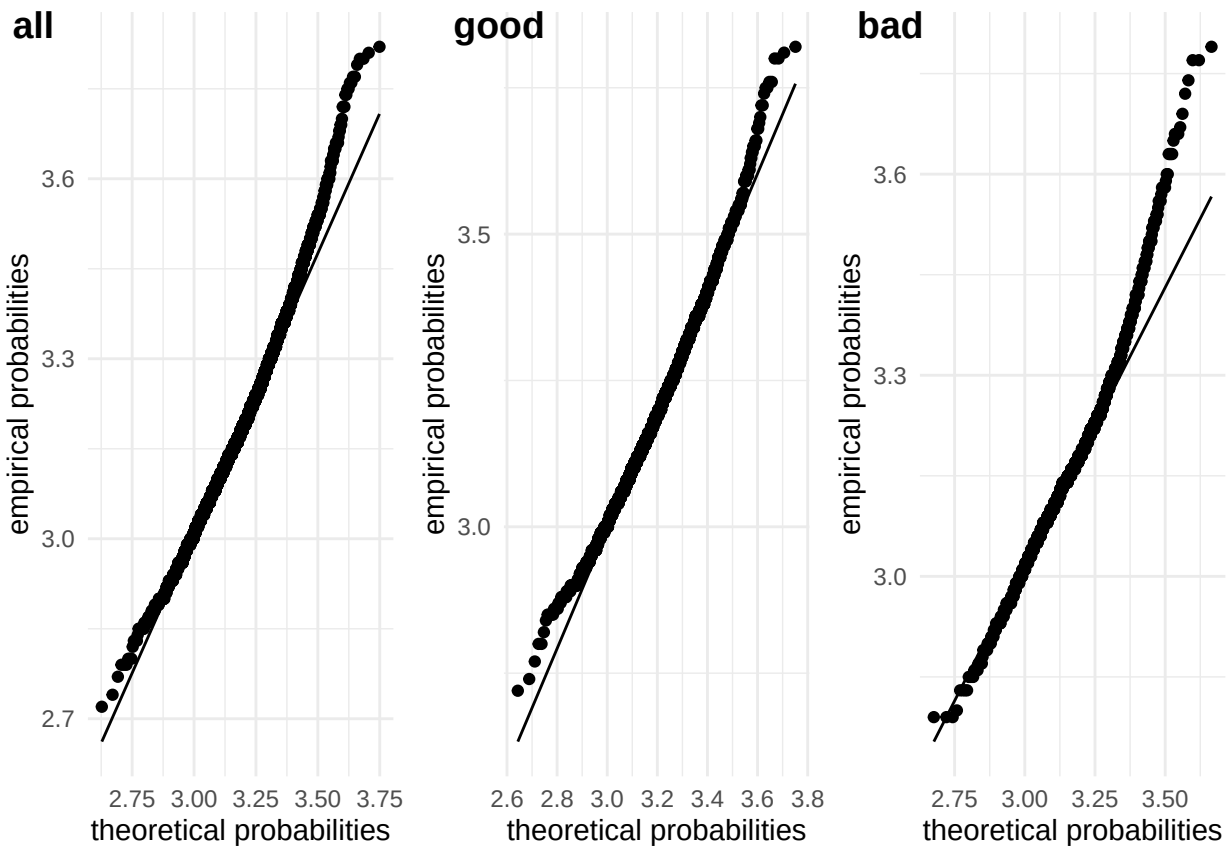
- good wines

```
pp_good <- ggplot(good_wine, aes(sample = pH)) +
  stat_qq(distribution = qnorm, dparams = list(mean = mean_pH_good, sd = sd_pH_good)) +
  stat_qq_line(distribution = qnorm, dparams = list(mean = mean_pH_good, sd = sd_pH_good)) +
  theme_minimal() +
  labs (
    x = "theoretical probabilities",
    y = "empirical probabilities"
  )
```

- bad wines

```
pp_bad <- ggplot(bad_wine, aes(sample = pH)) +
  stat_qq(distribution = qnorm, dparams = list(mean = mean_pH_bad, sd = sd_pH_bad)) +
  stat_qq_line(distribution = qnorm, dparams = list(mean = mean_pH_bad, sd = sd_pH_bad)) +
  theme_minimal() +
  labs (
    x = "theoretical probabilities",
    y = "empirical probabilities"
  )
```

```
ggarrange(pp_all, pp_good, pp_bad, labels = c("all", "good", "bad"), ncol = 3, nrow = 1)
```

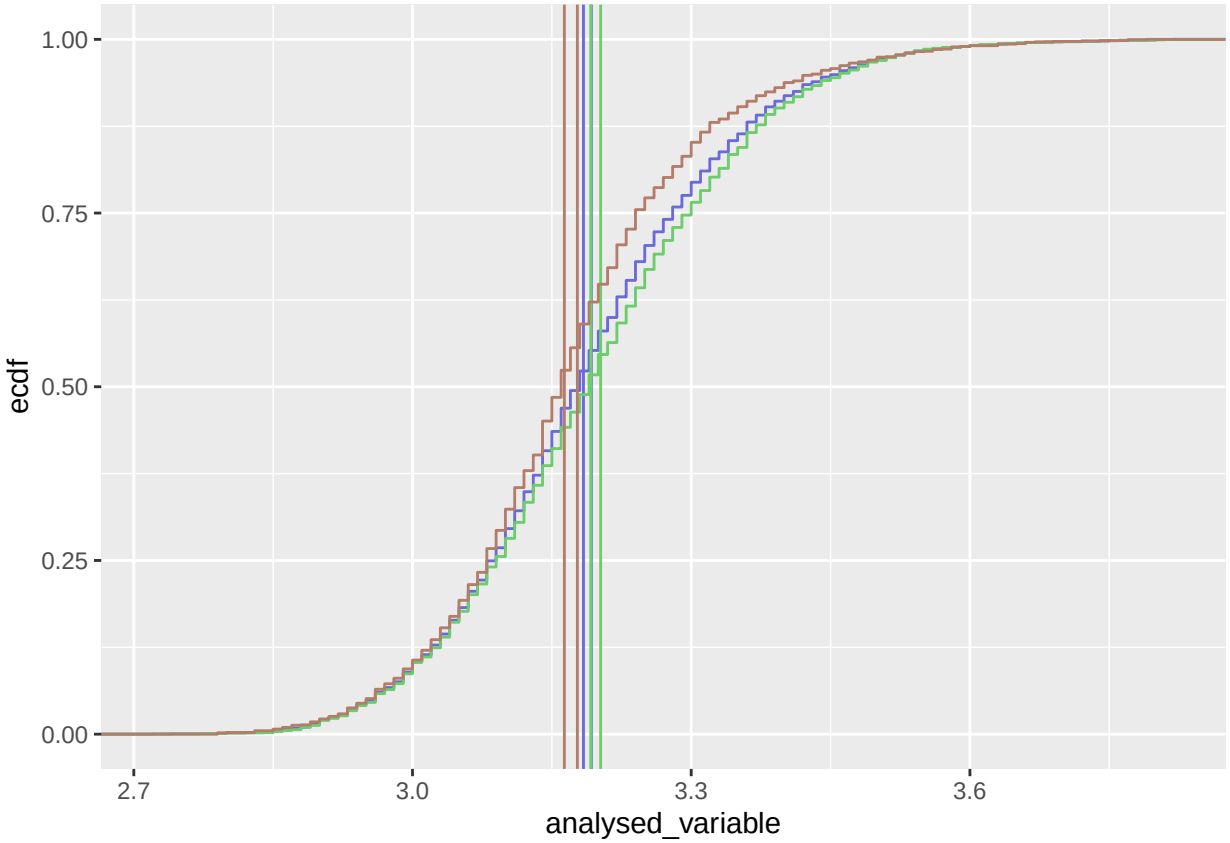


Samples probably do not follow normal distribution, as tails of QQ and PP plots do not lay on the line.

c. empirical distribution function + confidence intervals

```
# calculate confidence bands

bands <- function(data, analysed_variable, alpha) {
  n <- length(data$analysed_variable)
  mean <- mean(data$analysed_variable)
  sd <- sd(data$analysed_variable)
  z_alpha <- qnorm(1 - alpha / 2)
  lower <- mean - z_alpha * sd / sqrt(n)
  upper <- mean + z_alpha * sd / sqrt(n)
  data.frame(lower, upper)
}

alpha <- 0.05
all_bands <- bands(wine, analysed_variable, alpha)
good_bands <- bands(good_wine, analysed_variable, alpha)
bad_bands <- bands(bad_wine, analysed_variable, alpha)
```

```
ggplot() +
  stat_ecdf(data = wine, aes(x = analysed_variable), color = "#6a6ad9") +
  geom_vline(xintercept = all_bands$lower, color = "#6a6ad9") +
  geom_vline(xintercept = all_bands$upper, color = "#6a6ad9") +
  stat_ecdf(data = good_wine, aes(x = analysed_variable), color = "#6dcc6b") +
  geom_vline(xintercept = good_bands$lower, color = "#6dcc6b") +
  geom_vline(xintercept = good_bands$upper, color = "#6dcc6b") +
  stat_ecdf(data = bad_wine, aes(x = analysed_variable), color = "#b37d69") +
  geom_vline(xintercept = bad_bands$lower, color = "#b37d69") +
  geom_vline(xintercept = bad_bands$upper, color = "#b37d69")
```



Exercise 3

a. MLE for mu

$$f(x;\mu,\sigma) = \frac{1}{2\sigma}\exp\left(-\frac{|x-\mu|}{\sigma}\right)$$

The log-likelihood function:

$$\ell(\mu,\sigma) = \sum_{i=1}^{n} \log\left(\frac{1}{2\sigma}\exp\left(-\frac{|X_i-\mu|}{\sigma}\right)\right) =$$

$$= \ell(\mu,\sigma) = -n\log(2\sigma) - \frac{1}{\sigma}\sum_{i=1}^{n}|X_i-\mu|$$

To find MLE for μ, we need to maximize $\ell(\mu,\sigma)$ with respect to μ This is equivalent to minimizing the sum of absolute deviations:

$$\text{minimize}\sum_{i=1}^{n}|X_i-\mu|$$

According to statistics, the sum of absolute deviations is minimal at the median

Having n even gives us two equally good estimators, but when number of observations is odd, the estimator will be unique.

b. quantile

- for 20 observations

```
set.seed(999)

n <- 20
mu <- 1
sigma <- 1
sample_20 <- rlaplace(n, mu, sigma)

real_median_20 <- median(sample_20)

mle_quantile_20 <- c()
diff_quantile_20 <- c()

for(type in 1:9) {
  mle_quantile_20[type] <- quantile(sample_20, 0.5, type = type)
  diff_quantile_20[type] <- mle_quantile_20[type] - real_median_20
}

diff_quantile_20
```

```
## [1] -0.01284831  0.00000000 -0.01284831 -0.01284831  0.00000000  0.00000000
## [7]  0.00000000  0.00000000  0.00000000
```

So, types 2, 5, 6, 7, 8, 9 of function quantile predict better than 1, 3 and 4 types Here, the choice of quantile type significantly affects the result because this sample may not be representative due to small number of observations

- for 1000 observations

```
set.seed(999)

n <- 1000
mu <- 1
sigma <- 1
sample_1000 <- rlaplace(n, mu, sigma)

real_median_1000 <- median(sample_1000)

mle_quantile_1000 <- c()
diff_quantile_1000 <- c()

for(type in 1:9) {
  mle_quantile_1000[type] <- quantile(sample_1000, 0.5, type = type)
  diff_quantile_1000[type] <- mle_quantile_1000[type] - real_median_1000
}

diff_quantile_1000
```

```
## [1] -0.0001569595  0.0000000000 -0.0001569595 -0.0001569595  0.0000000000
## [6]  0.0000000000  0.0000000000  0.0000000000  0.0000000000
```

For 1000 observations the best predictors are 2, 5, 6-9 types of fn quantile, similar as for 20 observations. It is noticable that the difference for other types of quantile fn is much smaller than for 20 observations, which means that increasing the number of observations estimator gets more confident and shows better results Looks like we just proved the law of large numbers :)

c. MLE function

```
mle_optimise <- function(data) {
  log_lik_laplace <- function(mu, data) {
    return(sum(abs(data - mu))) # from calculations in a
  }
  result <- optimise(
    log_lik_laplace,
    interval = c(min(data), max(data)),
    data = data)
  return(result$minimum)
}

mle_optimise_20 <- mle_optimise(sample_20)
mle_quantile_20 <- quantile(sample_20, 0.5, type = 2) # taking second as one of the best types

mle_optimise_1000 <- mle_optimise(sample_1000)
mle_quantile_1000 <- quantile(sample_1000, 0.5, type = 2)

results <- data.frame(
  optimise = c(mle_optimise_20, mle_optimise_1000),
  quantile = c(mle_quantile_20, mle_quantile_1000),
  real = c(real_median_20, real_median_1000),
  row.names = c("20", "1000")
)
print(results)
```

```
##      optimise quantile     real
## 20   1.066128 1.067329 1.067329
## 1000 1.028546 1.028643 1.028643
```

Function 'optimise' uses a combination of golden section search and successive parabolic interpolation to find the minimum or maximum in the selected interval. Golden section search uses golden ratio to narrow the range of values that potentially can be the extremums, while successive parabolic interpolation fits a quadratic function through three points, then the vertex is used for new fitting and so on.

The Newton-Raphson method is not suitable for Laplace function, as it requires continuously differentable function, but Laplace one is not smooth at the point $\mu = X_i$

Talking about the results, quantile estimates better, because, as I understood, it simply takes the median, knowing that it's a best estimator However, optimise function also gives a pretty close estimate especially when increasing the sample size.

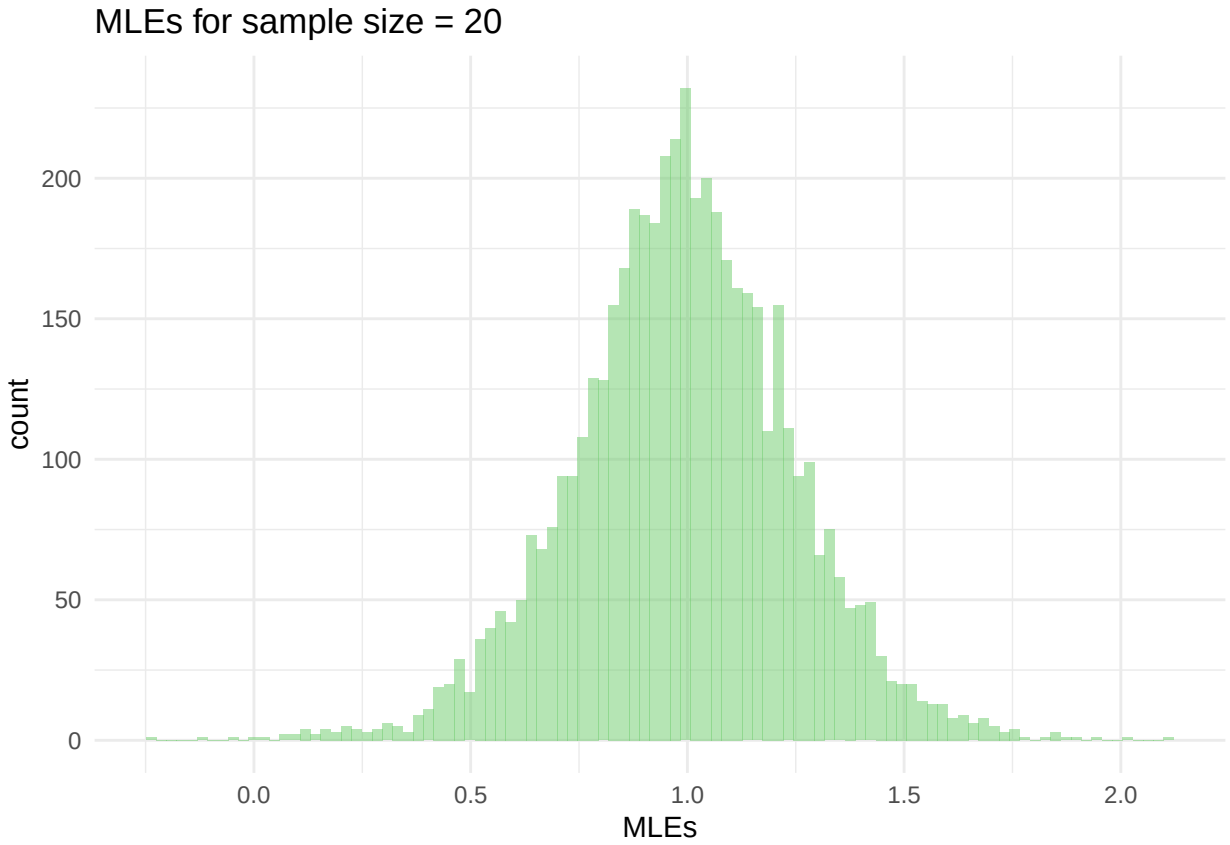
d. MLE distribution

- sample size = 20

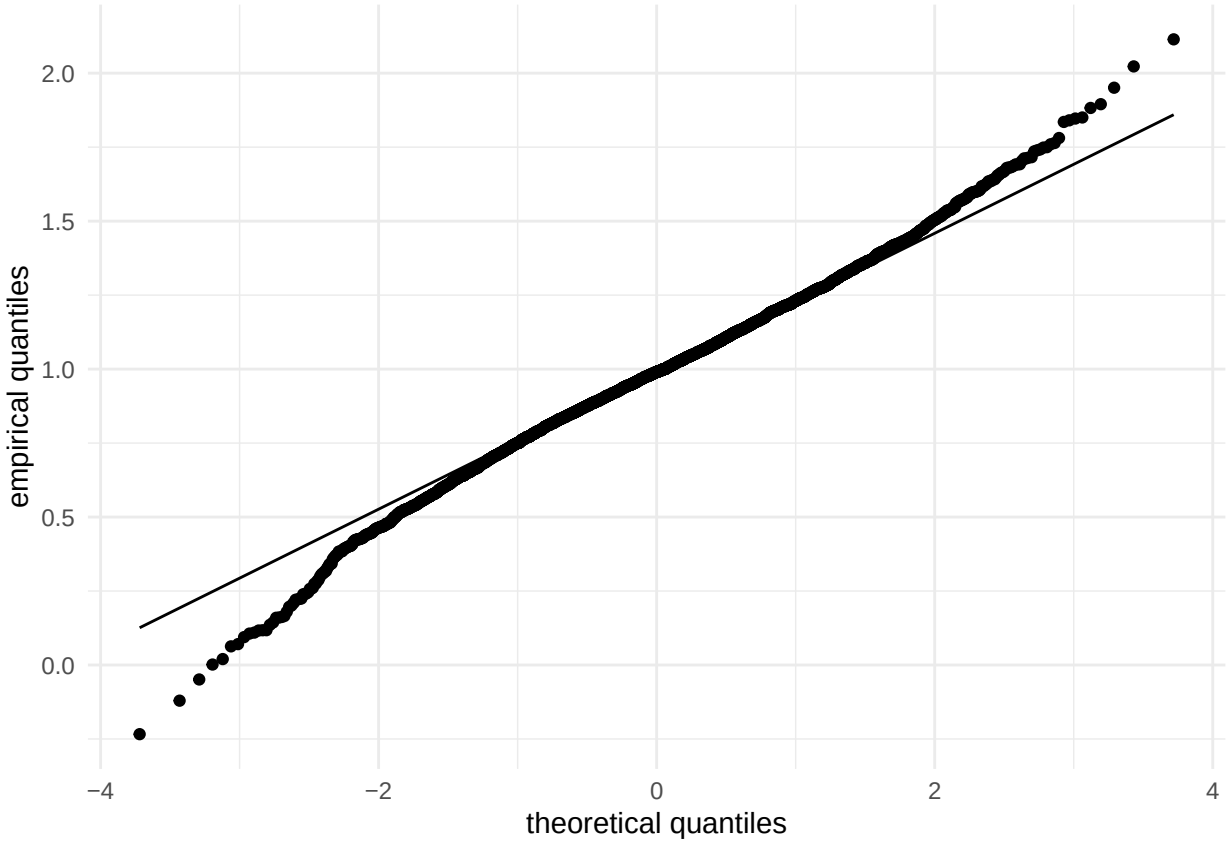
```r
set.seed(1000)

n = 20 # sample size
m = 5000 # num of MLEs
mu <- 1
sigma <- 1

mle_generator <- c()

for(i in 1:m) {
  mle_sample <- rlaplace(n, mu, sigma)
  mle_generator[i] <- mle_optimise(mle_sample)
}

mle_df_20 <- data.frame(mle_generator)
head(mle_df_20)
```

```
##   mle_generator
## 1     1.4299422
## 2     0.9766319
## 3     1.2763803
## 4     1.3678812
## 5     0.8346567
## 6     1.2527940
```

```r
ggplot(mle_df_20, aes(x = mle_generator)) +
  geom_histogram(position = "identity", alpha = .5, bins = 100, fill="#6dcc6b") +
  theme_minimal() +
  theme(legend.position = "none") +
  labs (
    x = "MLEs",
    y = "count",
    title = "MLEs for sample size = 20"
  )
```




```
ggplot(mle_df_20, aes(sample = mle_generator)) +
  stat_qq() +
  stat_qq_line() +
  theme_minimal() +
  labs (
    x = "theoretical quantiles",
    y = "empirical quantiles"
  )
```

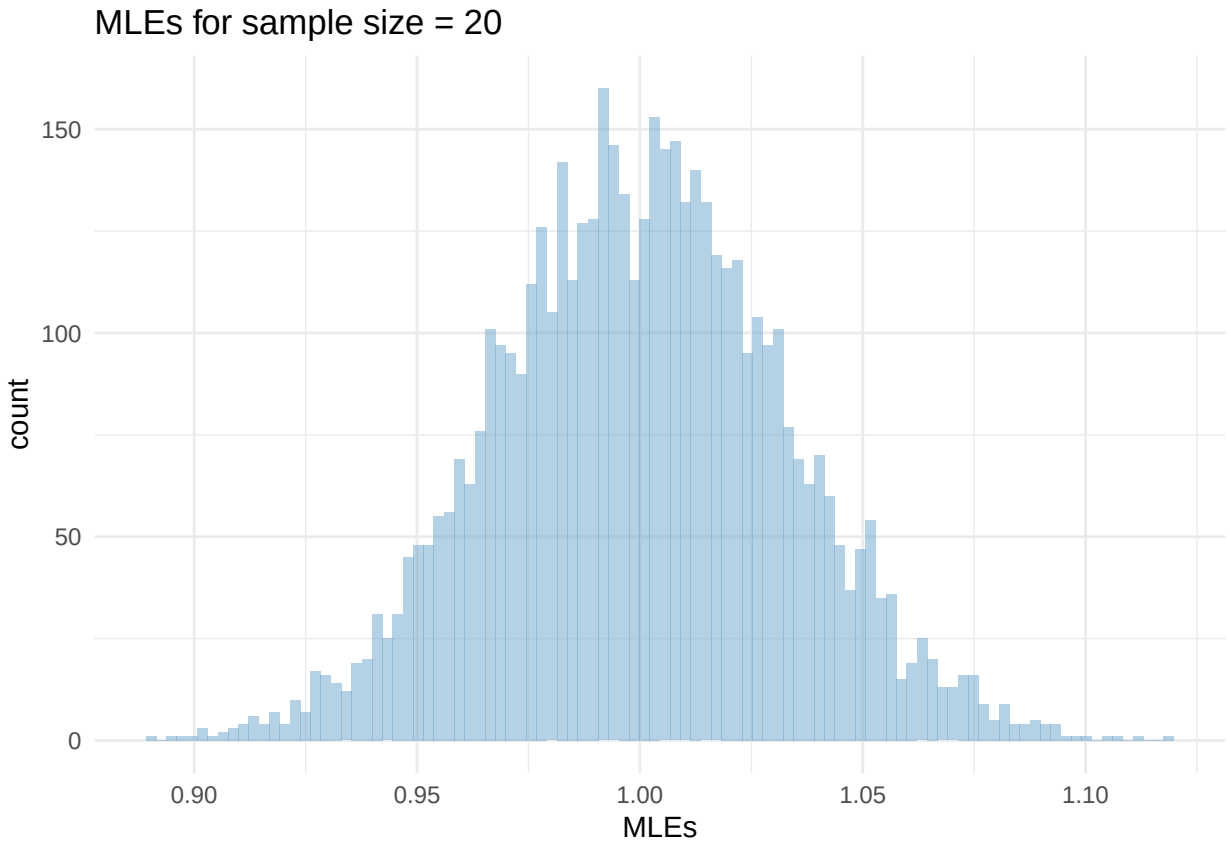


- sample size = 1000

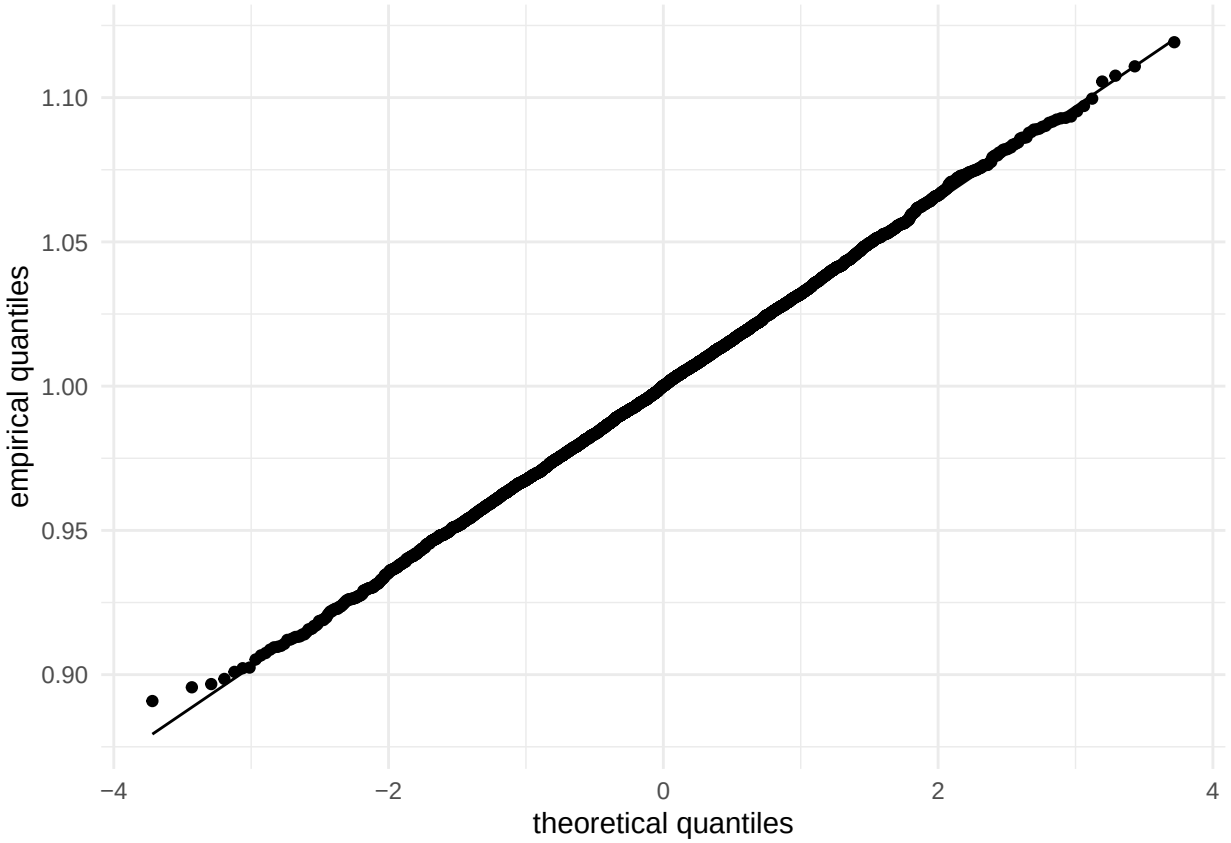
```
set.seed(1000)

n = 1000 # sample size
m = 5000 # num of MLEs
mu <- 1
sigma <- 1

mle_generator <- c()

for(i in 1:m) {
  mle_sample <- rlaplace(n, mu, sigma)
  mle_generator[i] <- mle_optimise(mle_sample)
}

mle_df_1000 <- data.frame(mle_generator)
head(mle_df_1000)
```

```
##   mle_generator
## 1     0.9860654
## 2     1.0242332
## 3     0.9770321
## 4     0.9992078
## 5     0.9645295
## 6     0.9656646
```

```r
ggplot(mle_df_1000, aes(x = mle_generator)) +
  geom_histogram(position = "identity", alpha = .5, bins = 100, fill="#6ba4cc") +
  theme_minimal() +
  theme(legend.position = "none") +
  labs (
    x = "MLEs",
    y = "count",
    title = "MLEs for sample size = 20"
  )
```




```r
ggplot(mle_df_1000, aes(sample = mle_generator)) +
  stat_qq() +
  stat_qq_line() +
  theme_minimal() +
  labs (
    x = "theoretical quantiles",
    y = "empirical quantiles"
  )
```



For samples of size 20 estimators visually followed the normal distribution, but QQ plot showed that it is skewed a little But, increasing the sample size to 1000 produced perfect normal distribution, giving us the representation of the central limit theorem.

- variances

```
var(mle_df_20)
```

```
##               mle_generator
## mle_generator    0.06470342
```

```
var(mle_df_1000)
```

```
##               mle_generator
## mle_generator   0.001056158
```

Increasing the sample size also highly decreases the variance, therefore, increases our confidence in predicting the true parameter.

Exercise 4

```r
house <- read.csv("kc_house_data.csv") %>%
  select(price, bedrooms, bathrooms,
  sqft_living, floors, view,
  condition, grade, yr_built)

head(house)
```

```
##     price bedrooms bathrooms sqft_living floors view condition grade yr_built
## 1  221900        3      1.00        1180      1    0         3     7     1955
## 2  538000        3      2.25        2570      2    0         3     7     1951
## 3  180000        2      1.00         770      1    0         3     6     1933
## 4  604000        4      3.00        1960      1    0         5     7     1965
## 5  510000        3      2.00        1680      1    0         3     8     1987
## 6 1225000        4      4.50        5420      1    0         3    11     2001
```

```r
summary(house)
```

```
##      price            bedrooms        bathrooms      sqft_living
##  Min.   :  75000   Min.   : 0.000   Min.   :0.000   Min.   :  290
##  1st Qu.: 321950   1st Qu.: 3.000   1st Qu.:1.750   1st Qu.: 1427
##  Median : 450000   Median : 3.000   Median :2.250   Median : 1910
##  Mean   : 540088   Mean   : 3.371   Mean   :2.115   Mean   : 2080
##  3rd Qu.: 645000   3rd Qu.: 4.000   3rd Qu.:2.500   3rd Qu.: 2550
##  Max.   :7700000   Max.   :33.000   Max.   :8.000   Max.   :13540
##      floors           view          condition         grade
##  Min.   :1.000   Min.   :0.0000   Min.   :1.000   Min.   : 1.000
##  1st Qu.:1.000   1st Qu.:0.0000   1st Qu.:3.000   1st Qu.: 7.000
##  Median :1.500   Median :0.0000   Median :3.000   Median : 7.000
##  Mean   :1.494   Mean   :0.2343   Mean   :3.409   Mean   : 7.657
##  3rd Qu.:2.000   3rd Qu.:0.0000   3rd Qu.:4.000   3rd Qu.: 8.000
##  Max.   :3.500   Max.   :4.0000   Max.   :5.000   Max.   :13.000
##     yr_built
##  Min.   :1900
##  1st Qu.:1951
##  Median :1975
##  Mean   :1971
##  3rd Qu.:1997
##  Max.   :2015
```

a. linear model for price

```r
model_price <- lm(price ~ bedrooms + bathrooms + sqft_living +
  floors + view + condition + grade + yr_built,
  data = house)

summary(model_price)
```

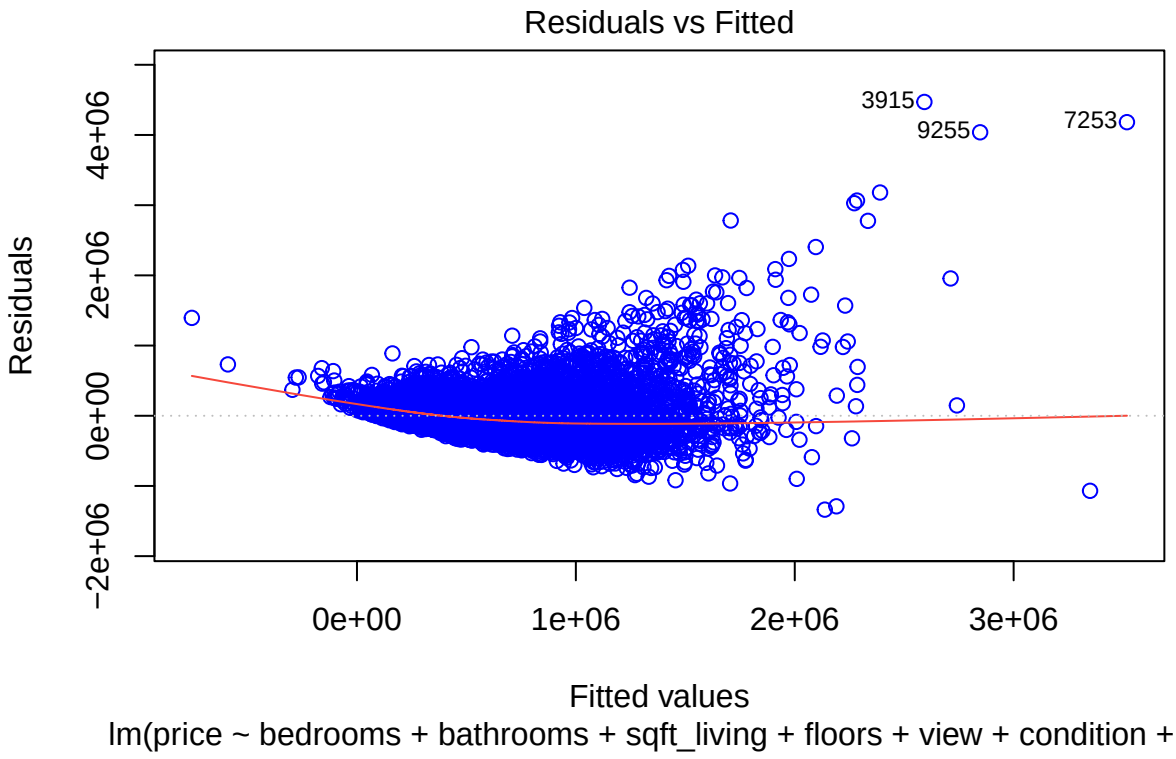
```
##
## Call:
## lm(formula = price ~ bedrooms + bathrooms + sqft_living + floors +
##     view + condition + grade + yr_built, data = house)
```

```
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1337280  -111873   -10359    90133  4470268
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  6.356e+06  1.326e+05  47.950  < 2e-16 ***
## bedrooms    -4.065e+04  2.066e+03 -19.671  < 2e-16 ***
## bathrooms    4.769e+04  3.487e+03  13.677  < 2e-16 ***
## sqft_living  1.693e+02  3.307e+00  51.199  < 2e-16 ***
## floors       2.832e+04  3.496e+03   8.101 5.72e-16 ***
## view         7.138e+04  2.107e+03  33.885  < 2e-16 ***
## condition    1.815e+04  2.519e+03   7.205 6.01e-13 ***
## grade        1.228e+05  2.187e+03  56.160  < 2e-16 ***
## yr_built    -3.650e+03  6.824e+01 -53.484  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 221600 on 21604 degrees of freedom
## Multiple R-squared:  0.6359, Adjusted R-squared:  0.6358
## F-statistic:  4717 on 8 and 21604 DF,  p-value: < 2.2e-16
```

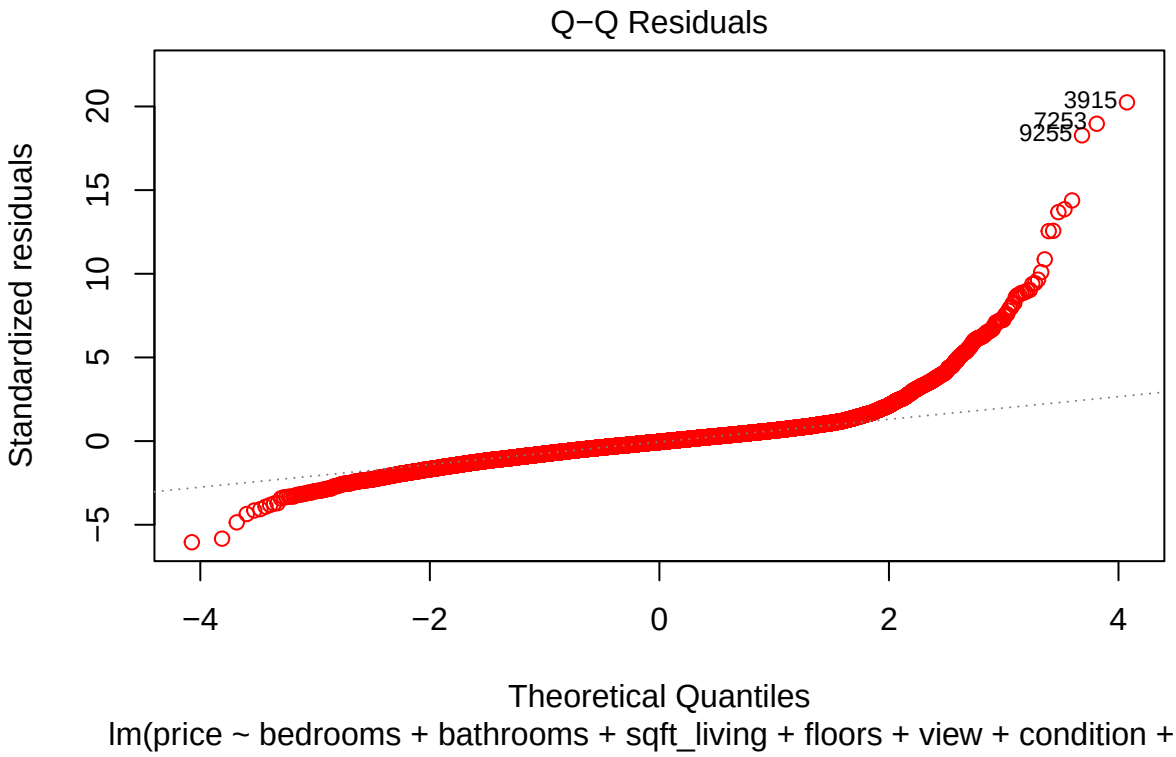
All variables are significant, R^2 equals to 0.6359 which means that $\sim 64\%$ of variance in the price is explained by those variables

- residual analysis

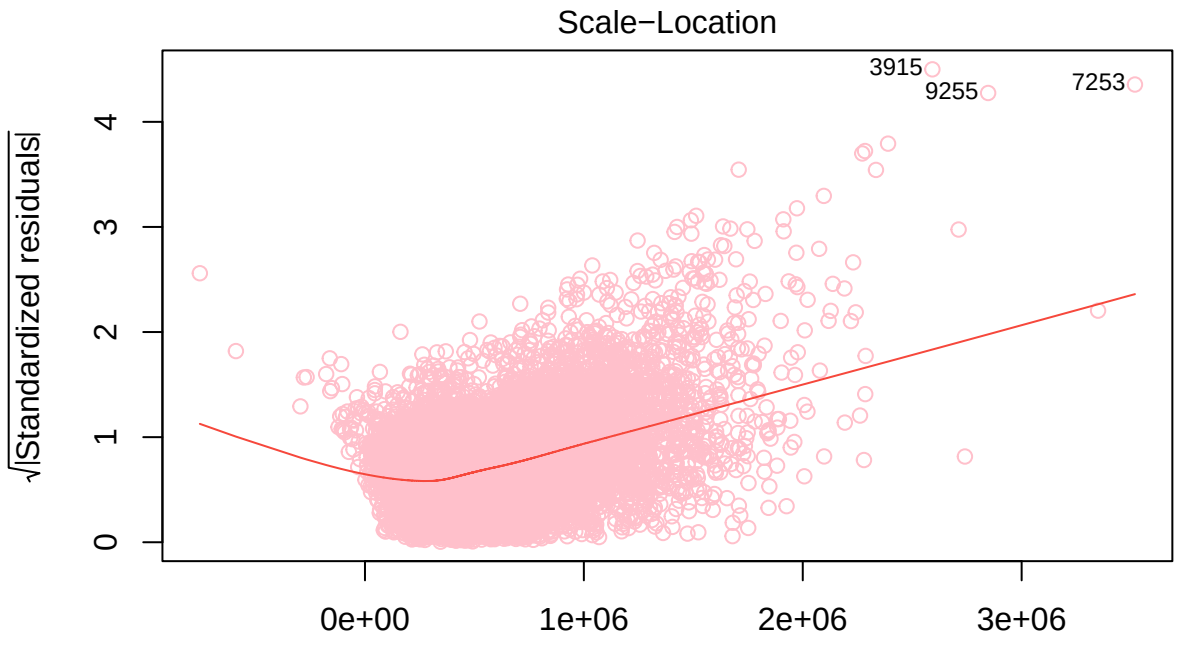
```
# residuals vs fitted
res_fit <- plot(model_price, which=1, col=c("blue"))
```




```
# q-q plot
qq <- plot(model_price, which=2, col=c("red"))
```




```
# scale-location
scale <- plot(model_price, which=3, col=c("pink"))
```




```
ggarrange(res_fit, qq, scale, ncol = 3, nrow = 1)
```

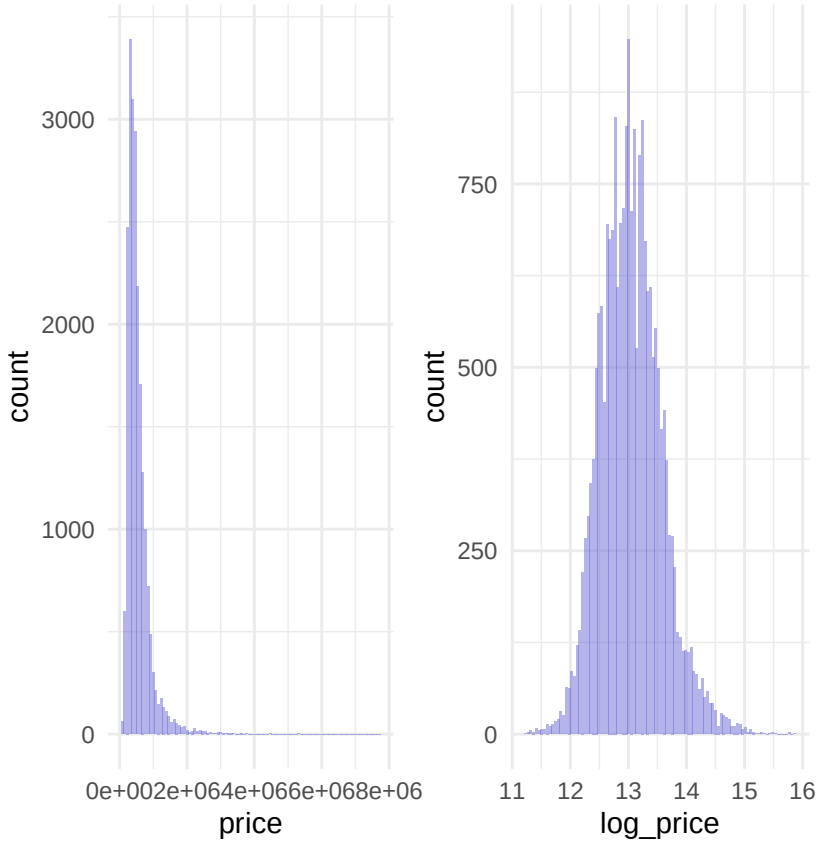
Residuals are not normally distributed and their variance is not the same, which means that our model violates normality and homoskedasticity of errors assumptions.

b. price vs log(price)

- histograms

```
price_hist <- ggplot(house, aes(x = price)) +
  geom_histogram(position = "identity", alpha = .5, bins = 100, fill="#6a6ad9") +
  theme_minimal() +
  theme(legend.position = "none")

log_price <- log(house$price)

log_price_hist <- ggplot(house, aes(x = log_price)) +
  geom_histogram(position = "identity", alpha = .5, bins = 100, fill="#6a6ad9") +
  theme_minimal() +
  theme(legend.position = "none")
```

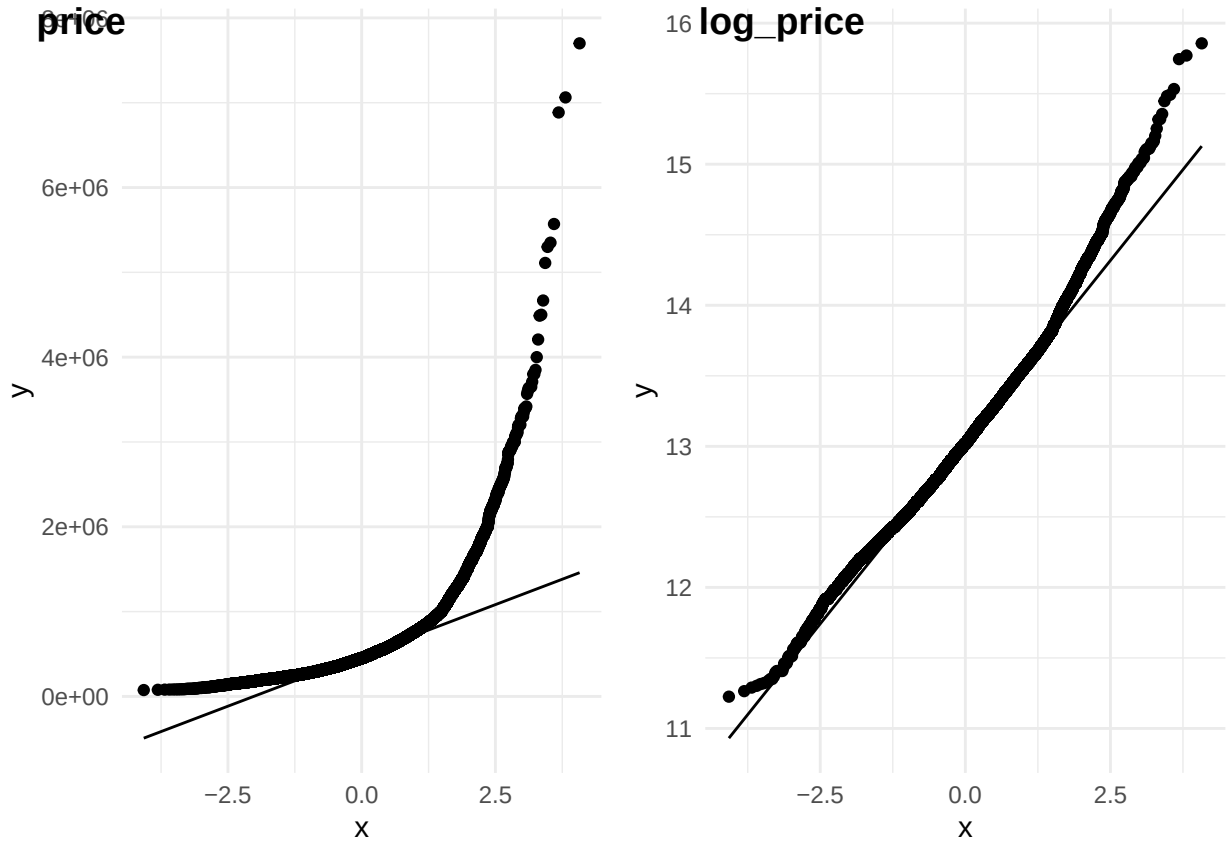
```
ggarrange(price_hist, log_price_hist, ncol = 3, nrow = 1)
```



- QQ plots

```
qq_price <- ggplot(house, aes(sample = price)) +
  stat_qq() +
  stat_qq_line() +
  theme_minimal()

qq_log_price <- ggplot(house, aes(sample = log_price)) +
  stat_qq() +
  stat_qq_line() +
  theme_minimal()
```

```
ggarrange(qq_price, qq_log_price, labels = c("price", "log_price"), ncol = 2, nrow = 1)
```



Log(price) looks closer to the normal distribution that price.

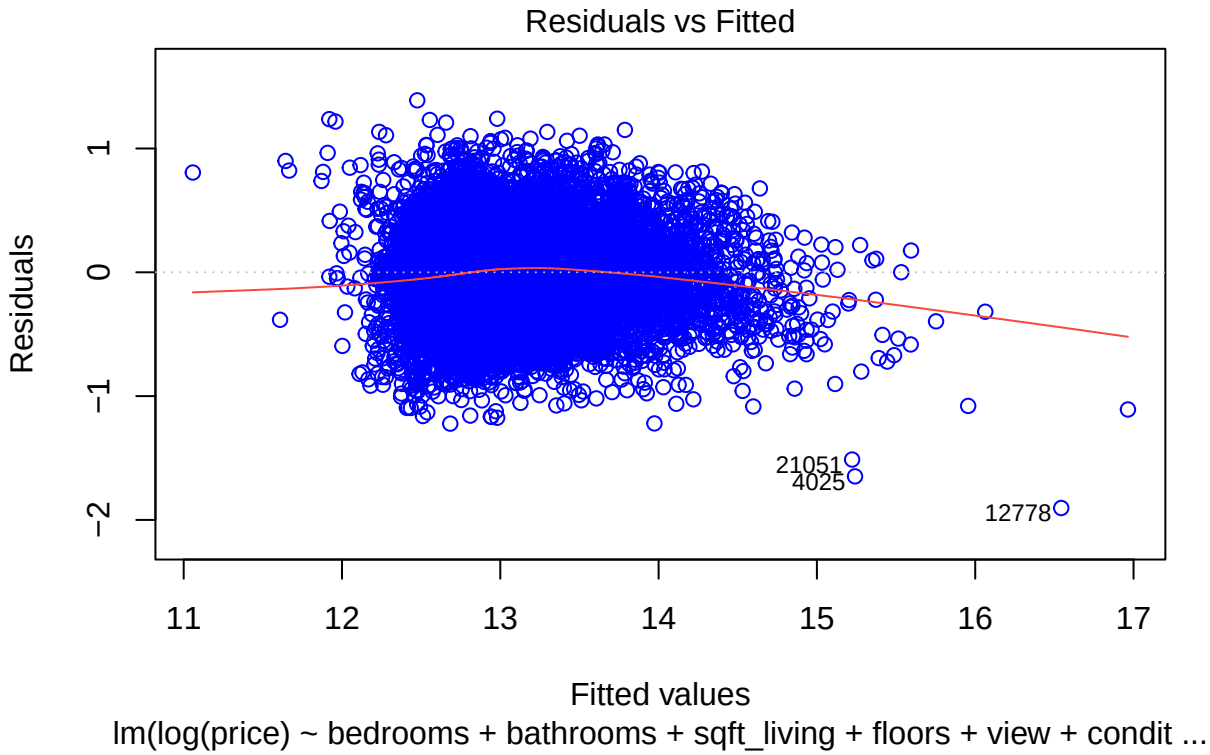
- model fit

```
model_log_price <- lm(log(price) ~ bedrooms + bathrooms + sqft_living +
  floors + view + condition + grade + yr_built,
  data = house)

summary(model_log_price)
```

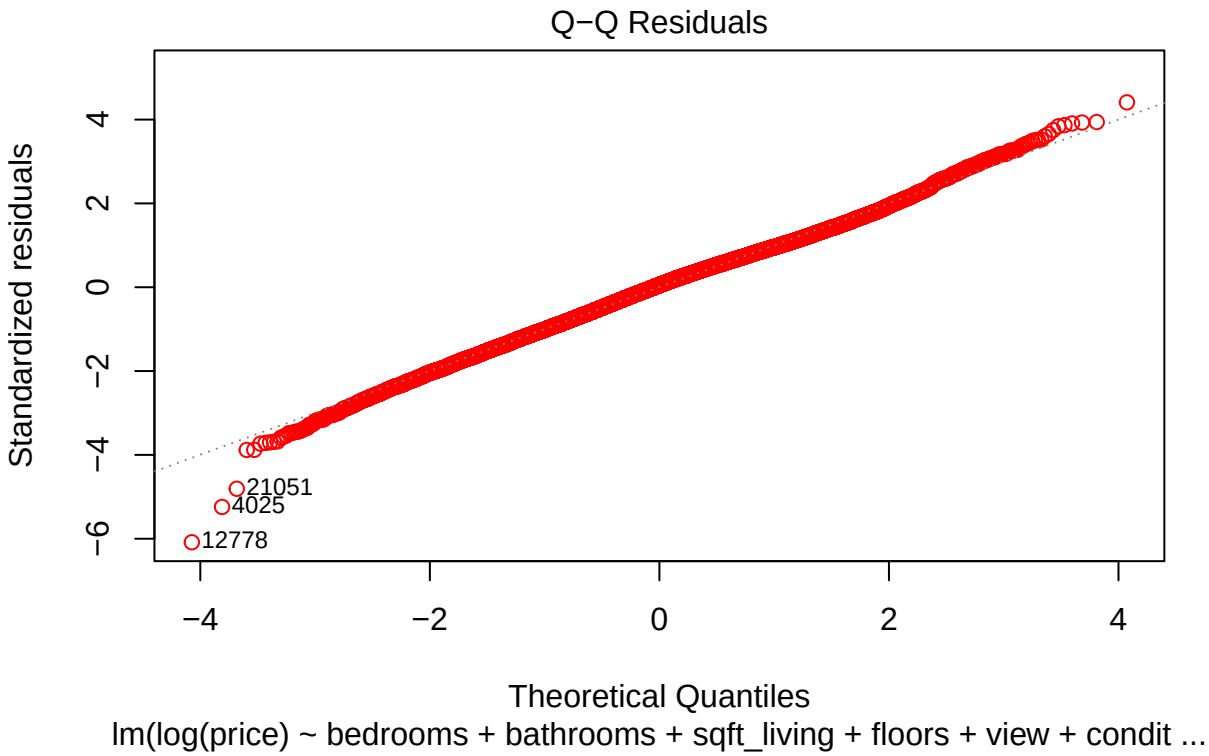
```
##
## Call:
## lm(formula = log(price) ~ bedrooms + bathrooms + sqft_living +
##     floors + view + condition + grade + yr_built, data = house)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.90374 -0.21157  0.01624  0.21288  1.38880
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  2.151e+01  1.884e-01 114.155  < 2e-16 ***
## bedrooms    -2.366e-02  2.937e-03  -8.056 8.28e-16 ***
## bathrooms    8.500e-02  4.956e-03  17.152  < 2e-16 ***
## sqft_living  1.664e-04  4.701e-06  35.402  < 2e-16 ***
```

```
## floors       8.569e-02  4.968e-03  17.246  < 2e-16 ***
## view         6.740e-02  2.994e-03  22.510  < 2e-16 ***
## condition    4.226e-02  3.580e-03  11.803  < 2e-16 ***
## grade        2.218e-01  3.108e-03  71.359  < 2e-16 ***
## yr_built    -5.526e-03  9.699e-05 -56.980  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.3149 on 21604 degrees of freedom
## Multiple R-squared:  0.6426, Adjusted R-squared:  0.6425
## F-statistic:  4856 on 8 and 21604 DF,  p-value: < 2.2e-16
```

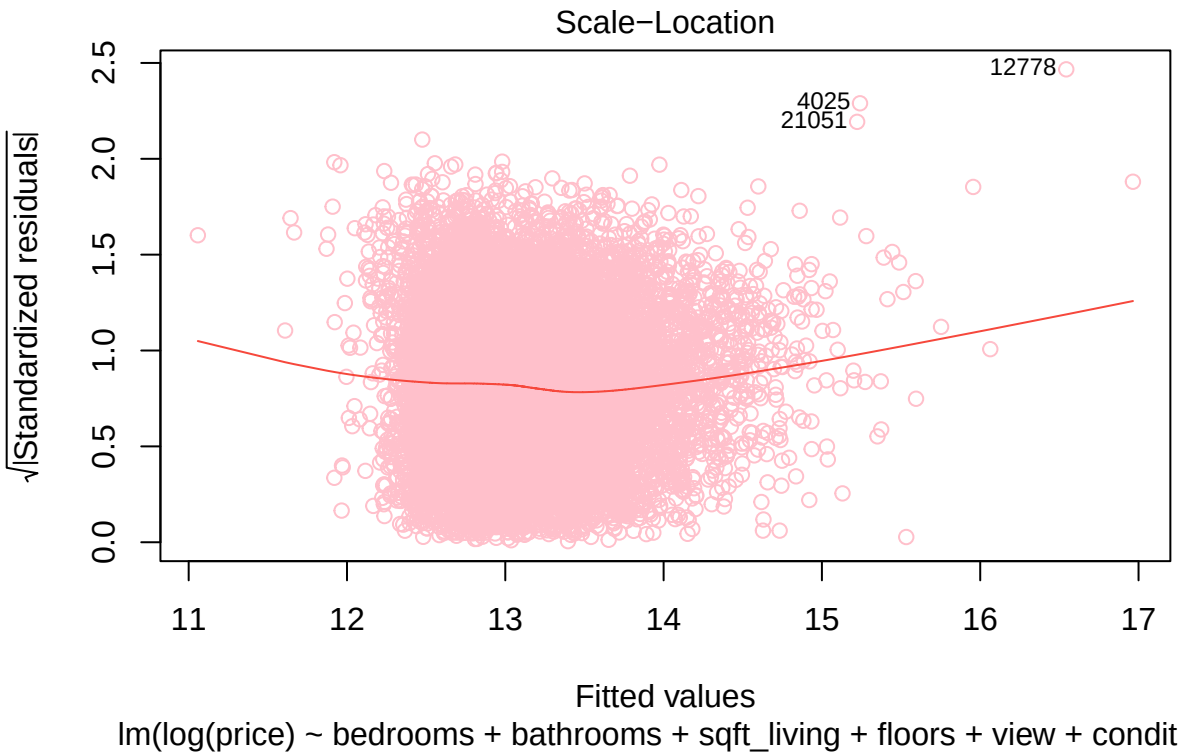
```
# residuals vs fitted
plot(model_log_price, which=1, col=c("blue"))
```




```
# q-q plot
plot(model_log_price, which=2, col=c("red"))
```




```
# scale-location
plot(model_log_price, which=3, col=c("pink"))
```



After changing price to log(price), all variables are still significant, R^2 is also around 64%, which means that predictive quality of the model didn't decrease Additionally, residuals look much better - they are normally distributed and variance seems constant for all values By looking at the covariates we can see the the percentage effect of each, which is much more convenient for further analysis than looking at absolute change.

c. effect of covariates

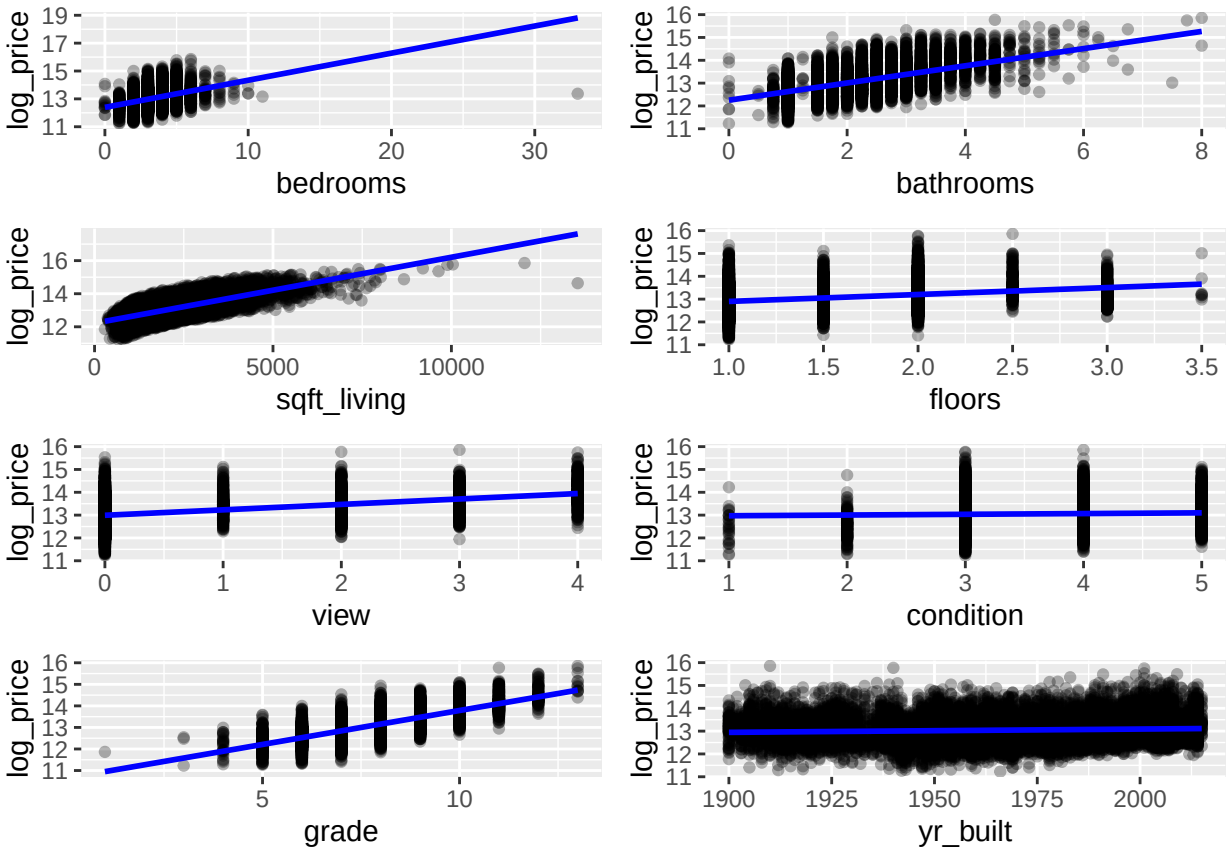
- intercept: expected value of log(price) is ~ 21.51 when all other parameters are zero
- bedrooms: each additional bedroom decreases the price by ~ 2.34% This, actually, doesn't make sense, by looking at the graph below, seems that the one outlier changes the sign of the coefficient, model fit, however, it should be positive.
- bathrooms: each additional bathroom increases the price by ~ 8.5%
- sqft_living: each additional square foot of living area increases the price by ~ 0.0166%
- floors: each additional floor in the house increases the price by ~ 8.57%
- view: each additional view increases the price by ~ 6.74%
- condition: each additional unit of condition rating increases the price by ~ 4.23%
- grade: each additional grade increases the price by ~ 22.18%
- yr_built: each additional year when house was built decreases the price by ~ 0.55%

```
bedrooms <- ggplot(house, aes(x = bedrooms, y = log_price)) +
  geom_point(alpha = 0.3) +
  geom_smooth(method = "lm", se = FALSE, color = "blue")

bathrooms <- ggplot(house, aes(x = bathrooms, y = log_price)) +
  geom_point(alpha = 0.3) +
```

```
  geom_smooth(method = "lm", se = FALSE, color = "blue")

sqft <- ggplot(house, aes(x = sqft_living, y = log_price)) +
  geom_point(alpha = 0.3) +
  geom_smooth(method = "lm", se = FALSE, color = "blue")

floors <- ggplot(house, aes(x = floors, y = log_price)) +
  geom_point(alpha = 0.3) +
  geom_smooth(method = "lm", se = FALSE, color = "blue")

views <- ggplot(house, aes(x = view, y = log_price)) +
  geom_point(alpha = 0.3) +
  geom_smooth(method = "lm", se = FALSE, color = "blue")

condition <- ggplot(house, aes(x = condition, y = log_price)) +
  geom_point(alpha = 0.3) +
  geom_smooth(method = "lm", se = FALSE, color = "blue")

grade <- ggplot(house, aes(x = grade, y = log_price)) +
  geom_point(alpha = 0.3) +
  geom_smooth(method = "lm", se = FALSE, color = "blue")

yr_built <- ggplot(house, aes(x = yr_built, y = log_price)) +
  geom_point(alpha = 0.3) +
  geom_smooth(method = "lm", se = FALSE, color = "blue")
```

```
ggarrange(bedrooms, bathrooms, sqft, floors, views, condition, grade, yr_built,
  ncol = 2, nrow = 4)
```

```
## `geom_smooth()` using formula = 'y ~ x'
## `geom_smooth()` using formula = 'y ~ x'
## `geom_smooth()` using formula = 'y ~ x'
## `geom_smooth()` using formula = 'y ~ x'
## `geom_smooth()` using formula = 'y ~ x'
## `geom_smooth()` using formula = 'y ~ x'
## `geom_smooth()` using formula = 'y ~ x'
## `geom_smooth()` using formula = 'y ~ x'
```



- adding squares

```
model_log_price_sq <- lm(log(price) ~ bedrooms + bathrooms + sqft_living  +
  floors + view + condition + grade + yr_built + I(sqft_living^2) + I(yr_built^2),
  data = house)

summary(model_log_price_sq)
```

```
##
## Call:
## lm(formula = log(price) ~ bedrooms + bathrooms + sqft_living +
##     floors + view + condition + grade + yr_built + I(sqft_living^2) +
##     I(yr_built^2), data = house)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -1.2644 -0.2113  0.0138  0.2107  1.4160
##
## Coefficients:
##                    Estimate Std. Error t value Pr(>|t|)
## (Intercept)       1.674e+02  1.082e+01  15.470   <2e-16 ***
## bedrooms         -2.978e-02  3.009e-03  -9.895   <2e-16 ***
## bathrooms         7.317e-02  4.959e-03  14.754   <2e-16 ***
## sqft_living       2.792e-04  8.690e-06  32.132   <2e-16 ***
## floors            4.733e-02  5.613e-03   8.432   <2e-16 ***
```

```
## view              7.222e-02  2.979e-03  24.247   <2e-16 ***
## condition         4.640e-02  3.591e-03  12.920   <2e-16 ***
## grade             2.176e-01  3.092e-03  70.355   <2e-16 ***
## yr_built         -1.545e-01  1.106e-02 -13.978   <2e-16 ***
## I(sqft_living^2) -1.782e-08  1.172e-09 -15.212   <2e-16 ***
## I(yr_built^2)     3.802e-05  2.823e-06  13.468   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.312 on 21602 degrees of freedom
## Multiple R-squared:  0.6492, Adjusted R-squared:  0.649
## F-statistic:  3998 on 10 and 21602 DF,  p-value: < 2.2e-16
```

Adding squares (which are significant predictors) slightly improves the model fit (0.6492 R^2 compared to 0.6426)

d. prediction

```
set.seed(1122)
sample_size <- 10806
train_indices <- sample(1:nrow(house), sample_size)
house_train <- house[train_indices, ]
house_test <- house[-train_indices, ]
summary(house_train)
```

```
##      price            bedrooms        bathrooms      sqft_living
##  Min.   :  78000   Min.   : 0.000   Min.   :0.000   Min.   :  380
##  1st Qu.: 320000   1st Qu.: 3.000   1st Qu.:1.500   1st Qu.: 1420
##  Median : 450000   Median : 3.000   Median :2.250   Median : 1900
##  Mean   : 537698   Mean   : 3.365   Mean   :2.106   Mean   : 2073
##  3rd Qu.: 642000   3rd Qu.: 4.000   3rd Qu.:2.500   3rd Qu.: 2538
##  Max.   :7700000   Max.   :10.000   Max.   :8.000   Max.   :13540
##      floors           view          condition         grade
##  Min.   :1.00   Min.   :0.0000   Min.   :1.000   Min.   : 3.000
##  1st Qu.:1.00   1st Qu.:0.0000   1st Qu.:3.000   1st Qu.: 7.000
##  Median :1.50   Median :0.0000   Median :3.000   Median : 7.000
##  Mean   :1.49   Mean   :0.2369   Mean   :3.415   Mean   : 7.646
##  3rd Qu.:2.00   3rd Qu.:0.0000   3rd Qu.:4.000   3rd Qu.: 8.000
##  Max.   :3.50   Max.   :4.0000   Max.   :5.000   Max.   :13.000
##     yr_built
##  Min.   :1900
##  1st Qu.:1951
##  Median :1974
##  Mean   :1971
##  3rd Qu.:1996
##  Max.   :2015
```

```
summary(house_test)
```

```
##      price            bedrooms        bathrooms      sqft_living
##  Min.   :  75000   Min.   : 0.000   Min.   :0.000   Min.   :  290
##  1st Qu.: 325000   1st Qu.: 3.000   1st Qu.:1.750   1st Qu.: 1430
```

```
##  Median : 451000   Median : 3.000   Median :2.250   Median : 1920
##  Mean   : 542478   Mean   : 3.376   Mean   :2.123   Mean   : 2086
##  3rd Qu.: 645000   3rd Qu.: 4.000   3rd Qu.:2.500   3rd Qu.: 2560
##  Max.   :7062500   Max.   :33.000   Max.   :7.750   Max.   :10040
##      floors           view          condition         grade
##  Min.   :1.000   Min.   :0.0000   Min.   :1.000   Min.   : 1.000
##  1st Qu.:1.000   1st Qu.:0.0000   1st Qu.:3.000   1st Qu.: 7.000
##  Median :1.500   Median :0.0000   Median :3.000   Median : 7.000
##  Mean   :1.499   Mean   :0.2317   Mean   :3.403   Mean   : 7.668
##  3rd Qu.:2.000   3rd Qu.:0.0000   3rd Qu.:4.000   3rd Qu.: 8.000
##  Max.   :3.500   Max.   :4.0000   Max.   :5.000   Max.   :13.000
##     yr_built
##  Min.   :1900
##  1st Qu.:1952
##  Median :1975
##  Mean   :1971
##  3rd Qu.:1997
##  Max.   :2015
```

- models b and c on training dataset

```
# model b
model_log_price <- lm(log(price) ~ bedrooms + bathrooms + sqft_living +
  floors + view + condition + grade + yr_built,
  data = house_train)

# model c
model_log_price_sq <- lm(log(price) ~ bedrooms + bathrooms + sqft_living  +
  floors + view + condition + grade + yr_built + I(sqft_living^2) + I(yr_built^2),
  data = house_train)
```

- predictions on test dataset

```
# predictions
pred_log_price <- predict(model_log_price, newdata = house_test)
pred_log_price_sq <- predict(model_log_price_sq, newdata = house_test)
```

- MSE for both models

```
mse_log_price <- mean((log(house_test$price) - pred_log_price)^2)
mse_log_price_sq <- mean((log(house_test$price) - pred_log_price_sq)^2)
mse_log_price
```

```
## [1] 0.09773623
```

```
mse_log_price_sq
```

```
## [1] 0.09654389
```

The second model gives a better prediction, as MSE there is smaller Let's improve the model by: * removing outlier with 33 bedrooms in training dataset * adding interaction bedrooms*bathrooms as they are correlated (bathrooms means bathroom per bedroom)* adding interaction condition*grade as they also seem correlated

```r
house_clean_train <- house_train %>%
  filter(bedrooms < 33)
```

```r
model_log_sqft <- lm(log(price) ~ bedrooms + bathrooms + sqft_living +
  floors + view + condition + grade + yr_built +
  I(sqft_living^2) + I(yr_built^2) +
  bedrooms*bathrooms + condition*grade,
  data = house_clean_train)

pred_log_sqft <- predict(model_log_sqft, newdata = house_test)
mean((log(house_test$price) - pred_log_sqft)^2) # new
```

```
## [1] 0.09646216
```

```r
mse_log_price_sq # old
```

```
## [1] 0.09654389
```

MSE in this case is a bit smaller which means that the new model predicts better, however, it still can be a random effect

Exercise 5

a. loading data

```r
hitters <- read.csv("Hitters.csv") %>%
  na.omit()

head(hitters)
```

```
##   AtBat Hits HmRun Runs RBI Walks Years CAtBat CHits CHmRun CRuns CRBI CWalks
## 2   315   81     7   24  38    39    14   3449   835     69   321  414    375
## 3   479  130    18   66  72    76     3   1624   457     63   224  266    263
## 4   496  141    20   65  78    37    11   5628  1575    225   828  838    354
## 5   321   87    10   39  42    30     2    396   101     12    48   46     33
## 6   594  169     4   74  51    35    11   4408  1133     19   501  336    194
## 7   185   37     1   23   8    21     2    214    42      1    30    9     24
##   League Division PutOuts Assists Errors Salary NewLeague
## 2      N        W     632      43     10  475.0         N
## 3      A        W     880      82     14  480.0         A
## 4      N        E     200      11      3  500.0         N
## 5      N        E     805      40      4   91.5         N
## 6      A        W     282     421     25  750.0         A
## 7      N        E      76     127      7   70.0         A
```

```r
summary(hitters)
```

```
##      AtBat            Hits           HmRun            Runs
##  Min.   : 19.0   Min.   :  1.0   Min.   : 0.00   Min.   :  0.00
##  1st Qu.:282.5   1st Qu.: 71.5   1st Qu.: 5.00   1st Qu.: 33.50
```

```
##  Median :413.0   Median :103.0   Median : 9.00   Median : 52.00
##  Mean   :403.6   Mean   :107.8   Mean   :11.62   Mean   : 54.75
##  3rd Qu.:526.0   3rd Qu.:141.5   3rd Qu.:18.00   3rd Qu.: 73.00
##  Max.   :687.0   Max.   :238.0   Max.   :40.00   Max.   :130.00
##       RBI             Walks            Years            CAtBat
##  Min.   :  0.00   Min.   :  0.00   Min.   : 1.000   Min.   :   19.0
##  1st Qu.: 30.00   1st Qu.: 23.00   1st Qu.: 4.000   1st Qu.:  842.5
##  Median : 47.00   Median : 37.00   Median : 6.000   Median : 1931.0
##  Mean   : 51.49   Mean   : 41.11   Mean   : 7.312   Mean   : 2657.5
##  3rd Qu.: 71.00   3rd Qu.: 57.00   3rd Qu.:10.000   3rd Qu.: 3890.5
##  Max.   :121.00   Max.   :105.00   Max.   :24.000   Max.   :14053.0
##      CHits            CHmRun           CRuns             CRBI
##  Min.   :   4.0   Min.   :  0.00   Min.   :   2.0   Min.   :   3.0
##  1st Qu.: 212.0   1st Qu.: 15.00   1st Qu.: 105.5   1st Qu.:  95.0
##  Median : 516.0   Median : 40.00   Median : 250.0   Median : 230.0
##  Mean   : 722.2   Mean   : 69.24   Mean   : 361.2   Mean   : 330.4
##  3rd Qu.:1054.0   3rd Qu.: 92.50   3rd Qu.: 497.5   3rd Qu.: 424.5
##  Max.   :4256.0   Max.   :548.00   Max.   :2165.0   Max.   :1659.0
##      CWalks          League            Division           PutOuts
##  Min.   :   1.0   Length:263         Length:263         Min.   :   0.0
##  1st Qu.:  71.0   Class :character   Class :character   1st Qu.: 113.5
##  Median : 174.0   Mode  :character   Mode  :character   Median : 224.0
##  Mean   : 260.3                                        Mean   : 290.7
##  3rd Qu.: 328.5                                        3rd Qu.: 322.5
##  Max.   :1566.0                                        Max.   :1377.0
##     Assists          Errors           Salary        NewLeague
##  Min.   :  0.0   Min.   : 0.000   Min.   :  67.5   Length:263
##  1st Qu.:  8.0   1st Qu.: 3.000   1st Qu.: 190.0   Class :character
##  Median : 45.0   Median : 7.000   Median : 425.0   Mode  :character
##  Mean   :118.8   Mean   : 8.593   Mean   : 535.9
##  3rd Qu.:192.0   3rd Qu.:13.000   3rd Qu.: 750.0
##  Max.   :492.0   Max.   :32.000   Max.   :2460.0
```

b. condition number

```
y <- hitters$Salary
X <- hitters %>%
  select(-Salary)

# create matrix and replace categorical variables with columns 1 or 0
X <- model.matrix(~ . -1, data = X) # -1 for removing intercept
head(X)
```

```
##   AtBat Hits HmRun Runs RBI Walks Years CAtBat CHits CHmRun CRuns CRBI CWalks
## 2   315   81     7   24  38    39    14   3449   835     69   321  414    375
## 3   479  130    18   66  72    76     3   1624   457     63   224  266    263
## 4   496  141    20   65  78    37    11   5628  1575    225   828  838    354
## 5   321   87    10   39  42    30     2    396   101     12    48   46     33
## 6   594  169     4   74  51    35    11   4408  1133     19   501  336    194
## 7   185   37     1   23   8    21     2    214    42      1    30    9     24
##   LeagueA LeagueN DivisionW PutOuts Assists Errors NewLeagueN
## 2       0       1         1     632      43     10          1
## 3       1       0         1     880      82     14          0
```

```
## 4       0       1         0     200      11      3          1
## 5       0       1         0     805      40      4          1
## 6       1       0         1     282     421     25          0
## 7       0       1         0      76     127      7          0
```

```
XtX <- t(X) %*% X

eigenvalues <- eigen(XtX)$values

condition_number <- max(eigenvalues) / min(eigenvalues)
condition_number
```

```
## [1] 703904352
```

The condition number is very high, which shows that there is a high multicollinearity among some variables

- standardization

```
X_stand <- scale(X)

XtX_stand <- t(X_stand) %*% X_stand

eigenvalues_stand <- eigen(XtX_stand)$values

condition_number_stand <- max(eigenvalues_stand) / min(eigenvalues_stand)
condition_number_stand
```

```
## [1] 8.980008e+16
```

Standardization didn't help, we even onbtained much higher number because there is a very small eigenvalue 2.131628e-14

c. models

- standard linear model

```
lm_hitters <- lm(y ~ X)
summary(lm_hitters)
```

```
##
## Call:
## lm(formula = y ~ X)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -907.62 -178.35  -31.11  139.09 1877.04
##
## Coefficients: (1 not defined because of singularities)
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  225.70301  109.89206   2.054 0.041059 *
```

```
## XAtBat        -1.97987    0.63398  -3.123 0.002008 **
## XHits          7.50077    2.37753   3.155 0.001808 **
## XHmRun         4.33088    6.20145   0.698 0.485616
## XRuns         -2.37621    2.98076  -0.797 0.426122
## XRBI          -1.04496    2.60088  -0.402 0.688204
## XWalks         6.23129    1.82850   3.408 0.000766 ***
## XYears        -3.48905   12.41219  -0.281 0.778874
## XCAtBat       -0.17134    0.13524  -1.267 0.206380
## XCHits         0.13399    0.67455   0.199 0.842713
## XCHmRun       -0.17286    1.61724  -0.107 0.914967
## XCRuns         1.45430    0.75046   1.938 0.053795 .
## XCRBI          0.80771    0.69262   1.166 0.244691
## XCWalks       -0.81157    0.32808  -2.474 0.014057 *
## XLeagueA     -62.59942   79.26140  -0.790 0.430424
## XLeagueN            NA         NA      NA       NA
## XDivisionW  -116.84925   40.36695  -2.895 0.004141 **
## XPutOuts       0.28189    0.07744   3.640 0.000333 ***
## XAssists       0.37107    0.22120   1.678 0.094723 .
## XErrors       -3.36076    4.39163  -0.765 0.444857
## XNewLeagueN  -24.76233   79.00263  -0.313 0.754218
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 315.6 on 243 degrees of freedom
## Multiple R-squared:  0.5461, Adjusted R-squared:  0.5106
## F-statistic: 15.39 on 19 and 243 DF,  p-value: < 2.2e-16
```

- ridge regression

```
lambda <- 70
ridge_hitters <- glmnet(X, y, alpha = 0, lambda = lambda)
as.data.frame(as.matrix(coef(ridge_hitters)))
```

```
##                        s0
## (Intercept)  5.647035e+01
## AtBat       -2.229055e-01
## Hits         1.646002e+00
## HmRun       -1.048269e+00
## Runs         1.179763e+00
## RBI          8.345634e-01
## Walks        2.432306e+00
## Years       -4.545099e+00
## CAtBat       8.238175e-03
## CHits        9.457440e-02
## CHmRun       5.890857e-01
## CRuns        1.881371e-01
## CRBI         1.937483e-01
## CWalks      -9.323679e-02
## LeagueA     -2.460078e+01
## LeagueN      2.460571e+01
## DivisionW   -1.142379e+02
## PutOuts      2.400898e-01
## Assists      9.822404e-02
```

```
## Errors      -3.021633e+00
## NewLeagueN  -1.031294e+01
```

- coefficients

```
lm_coefs <- as.vector(round(coef(lm_hitters), 4))
ridge_coefs <- as.vector(round(coef(ridge_hitters), 4))
variables <- c("Intercept", colnames(X))

coefs <- data.frame(
  variable = variables,
  lm_coefs = lm_coefs,
  ridge_coefs = ridge_coefs,
  diff = lm_coefs - ridge_coefs
)

coefs
```

```
##      variable  lm_coefs ridge_coefs     diff
## 1   Intercept  225.7030     56.4703 169.2327
## 2       AtBat   -1.9799     -0.2229  -1.7570
## 3        Hits    7.5008      1.6460   5.8548
## 4       HmRun    4.3309     -1.0483   5.3792
## 5        Runs   -2.3762      1.1798  -3.5560
## 6         RBI   -1.0450      0.8346  -1.8796
## 7       Walks    6.2313      2.4323   3.7990
## 8       Years   -3.4891     -4.5451   1.0560
## 9      CAtBat   -0.1713      0.0082  -0.1795
## 10      CHits    0.1340      0.0946   0.0394
## 11     CHmRun   -0.1729      0.5891  -0.7620
## 12      CRuns    1.4543      0.1881   1.2662
## 13       CRBI    0.8077      0.1937   0.6140
## 14     CWalks   -0.8116     -0.0932  -0.7184
## 15    LeagueA  -62.5994    -24.6008 -37.9986
## 16    LeagueN        NA     24.6057       NA
## 17  DivisionW -116.8492   -114.2379  -2.6113
## 18    PutOuts    0.2819      0.2401   0.0418
## 19    Assists    0.3711      0.0982   0.2729
## 20     Errors   -3.3608     -3.0216  -0.3392
## 21 NewLeagueN  -24.7623    -10.3129 -14.4494
```

I would say that ridge coefficients tend to be closer to zero and in general smaller, than standard linear model ones.

d. data split

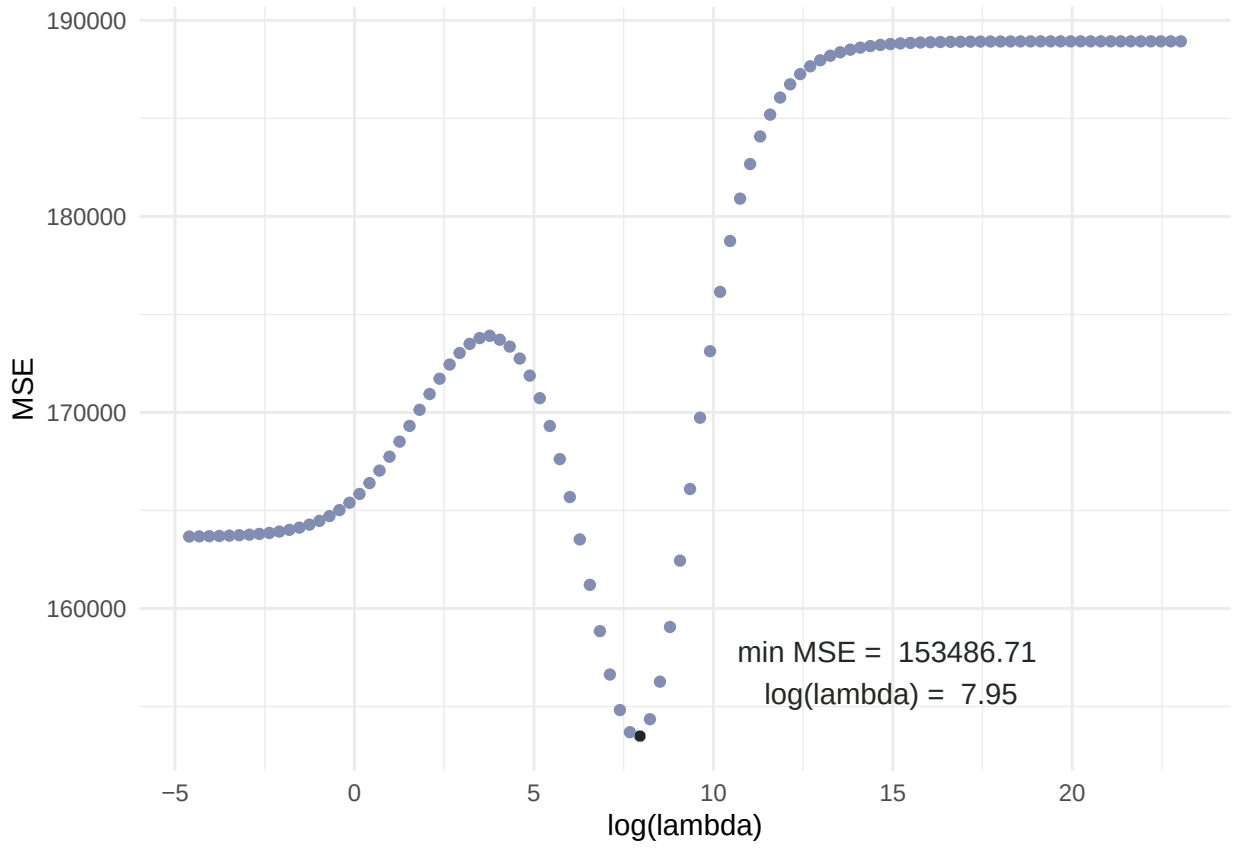
```
set.seed(1122)

# we'll take 80% train and 20% test
train_indices <- sample(1:nrow(hitters), size = 0.8 * nrow(hitters))
hitters_train <- hitters[train_indices, ]
hitters_test <- hitters[-train_indices, ]
```

```r
# design matrices
X_train <- model.matrix(Salary ~ ., data = hitters_train)
y_train <- hitters_train$Salary

X_test <- model.matrix(Salary ~ ., data = hitters_test)
y_test <- hitters_test$Salary
```

e. best lambda

```r
optimal_lambda <- function(lambda) {
  ridge_hitters_train <- glmnet(X_train, y_train, alpha = 0, lambda = lambda)
  pred_hitters <- predict(ridge_hitters_train, newx = X_test)
  mse <- colMeans((pred_hitters - y_test)^2)
  return(mse)
}
```

```r
potential_lambdas <- 10^seq(10, -2, length = 100)
mse_ridge <- sapply(potential_lambdas, optimal_lambda)

mse_ridge_df <- data.frame(
  log_lambda = log(potential_lambdas),
  mse = mse_ridge
)

# to display the needed point on the graph
best_lambda_ridge <- potential_lambdas[which.min(mse_ridge)]
min_mse_ridge <- min(mse_ridge)

ggplot(mse_ridge_df, aes(x = log_lambda, y = mse)) +
  geom_point(color = "#818db2") +
  geom_point(aes(x = log(best_lambda_ridge), y = min_mse_ridge), color = "#242824", size = 1) +
  labs(
    x = "log(lambda)",
    y = "MSE"
  ) +
  annotate("text", x = log(best_lambda_ridge) + 7, y = min_mse_ridge,
           label = paste("min MSE = ", round(min_mse_ridge, 2),
                         "\nlog(lambda) = ", round(log(best_lambda_ridge), 2)),
           vjust = -0.5, color = "#242824") +
  theme_minimal()
```



f. fitting with optimal lambda

```
lambda <- best_lambda_ridge
ridge_hitters <- glmnet(X, y, alpha = 0, lambda = lambda)
as.data.frame(as.matrix(coef(ridge_hitters)))
```

```
##                       s0
## (Intercept) 223.894005132
## AtBat         0.087821159
## Hits          0.359100399
## HmRun         1.158696741
## Runs          0.578919608
## RBI           0.578728301
## Walks         0.747276808
## Years         2.422873069
## CAtBat        0.007394323
## CHits         0.028411906
## CHmRun        0.211257547
## CRuns         0.056988201
## CRBI          0.058927048
## CWalks        0.057274196
## LeagueA      -2.658093515
## LeagueN       2.658241250
## DivisionW   -20.821649820
## PutOuts       0.050338927
```

```
## Assists       0.007143813
## Errors       -0.135505569
## NewLeagueN    2.484061051
```

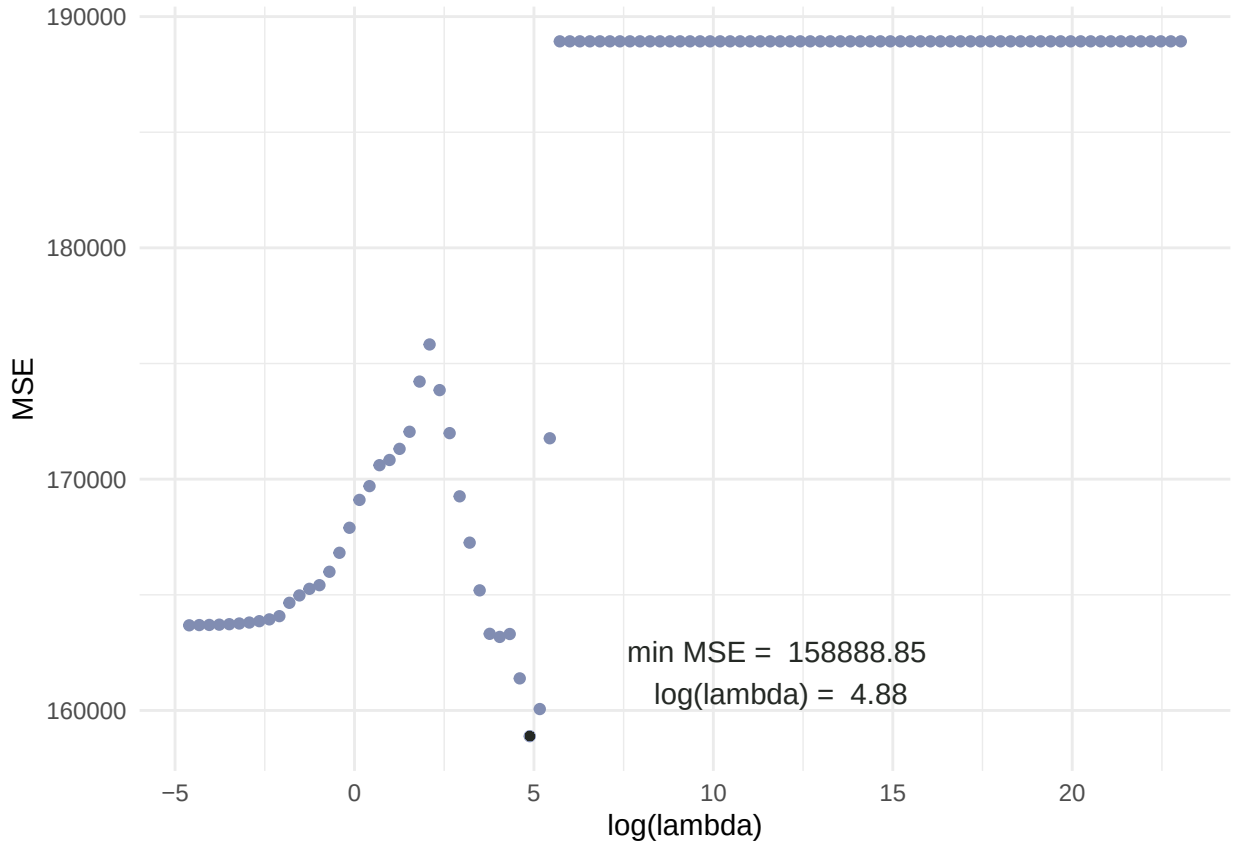
The most important variables are ones with bigger absolute values

- Division - players in the division W have much lower salaries than ones in division E
- League - being in league A is associated with lower salaries, while league N players in general have higher
- NewLeague - players in the New League N tend to have higher salaries
- Years - with the increase in the num of years in the major leagues the salary also increases

There are no coefficients that equal zero, as ridge regression doesn't make coefficients equal to zero.

g. lasso regression

```
optimal_lambda_lasso <- function(lambda) {
  lasso_hitters_train <- glmnet(X_train, y_train, alpha = 1, lambda = lambda)
  pred_hitters <- predict(lasso_hitters_train, newx = X_test)
  mse <- colMeans((pred_hitters - y_test)^2)
  return(mse)
}

mse_lasso <- sapply(potential_lambdas, optimal_lambda_lasso)

mse_lasso_df <- data.frame(
  log_lambda = log(potential_lambdas),
  mse = mse_lasso
)

best_lambda_lasso <- potential_lambdas[which.min(mse_lasso)]
min_mse_lasso <- min(mse_lasso)

ggplot(mse_lasso_df, aes(x = log_lambda, y = mse)) +
  geom_point(color = "#818db2") +
  geom_point(aes(x = log(best_lambda_lasso), y = min_mse_lasso), color = "#242824", size = 1) +
  labs(
    x = "log(lambda)",
    y = "MSE"
  ) +
  annotate("text", x = log(best_lambda_lasso) + 7, y = min_mse_lasso,
           label = paste("min MSE = ", round(min_mse_lasso, 2),
                         "\nlog(lambda) = ", round(log(best_lambda_lasso), 2)),
           vjust = -0.5, color = "#242824") +
  theme_minimal()
```




```
lambda <- best_lambda_lasso
lasso_hitters <- glmnet(X, y, alpha = 1, lambda = lambda)
as.data.frame(as.matrix(coef(lasso_hitters)))
```

```
##                       s0
## (Intercept) 316.02183375
## AtBat         0.00000000
## Hits          0.72112214
## HmRun         0.00000000
## Runs          0.00000000
## RBI           0.00000000
## Walks         0.56907852
## Years         0.00000000
## CAtBat        0.00000000
## CHits         0.00000000
## CHmRun        0.00000000
## CRuns         0.09666601
## CRBI          0.25371287
## CWalks        0.00000000
## LeagueA       0.00000000
## LeagueN       0.00000000
## DivisionW     0.00000000
## PutOuts       0.00000000
## Assists       0.00000000
## Errors        0.00000000
## NewLeagueN    0.00000000
```

The results are very different compared to the ridge regression. I suppose, this happened because lasso regression sets some coefficients to zero to ensure that no unsignificant variables are included in the model. However, the important variables still differ between these two models :(Maybe the most important conclusion we can get from this task is that we should be careful while using the regularization and choose the type correctly depending on the purpose of our research - prediction or feature selection.

Exercise 6

a. GLM for donation with frequency and amount

```
donations <- read.csv('transfusion.data')
names(donations) <- c('recency', 'frequency', 'amount', 'time', 'donation')
head(donations)
```

```
##   recency frequency amount time donation
## 1       2        50  12500   98        1
## 2       0        13   3250   28        1
## 3       1        16   4000   35        1
## 4       2        20   5000   45        1
## 5       1        24   6000   77        0
## 6       4         4   1000    4        0
```

```
summary(donations)
```

```
##     recency         frequency          amount           time
##  Min.   : 0.000   Min.   : 1.000   Min.   :  250   Min.   : 2.00
##  1st Qu.: 2.750   1st Qu.: 2.000   1st Qu.:  500   1st Qu.:16.00
##  Median : 7.000   Median : 4.000   Median : 1000   Median :28.00
##  Mean   : 9.507   Mean   : 5.515   Mean   : 1379   Mean   :34.28
##  3rd Qu.:14.000   3rd Qu.: 7.000   3rd Qu.: 1750   3rd Qu.:50.00
##  Max.   :74.000   Max.   :50.000   Max.   :12500   Max.   :98.00
##     donation
##  Min.   :0.000
##  1st Qu.:0.000
##  Median :0.000
##  Mean   :0.238
##  3rd Qu.:0.000
##  Max.   :1.000
```

We will use the binomial family in the model, as we want to predict the binary outcome

```
# frequency GLM model
freq_model <- glm(donation ~ frequency, data = donations, family = binomial)
summary(freq_model)
```

```
##
## Call:
## glm(formula = donation ~ frequency, family = binomial, data = donations)
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
```

```
## (Intercept) -1.63796    0.12822 -12.774  < 2e-16 ***
## frequency    0.07937    0.01514   5.243 1.58e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 820.89  on 747  degrees of freedom
## Residual deviance: 789.12  on 746  degrees of freedom
## AIC: 793.12
##
## Number of Fisher Scoring iterations: 4
```

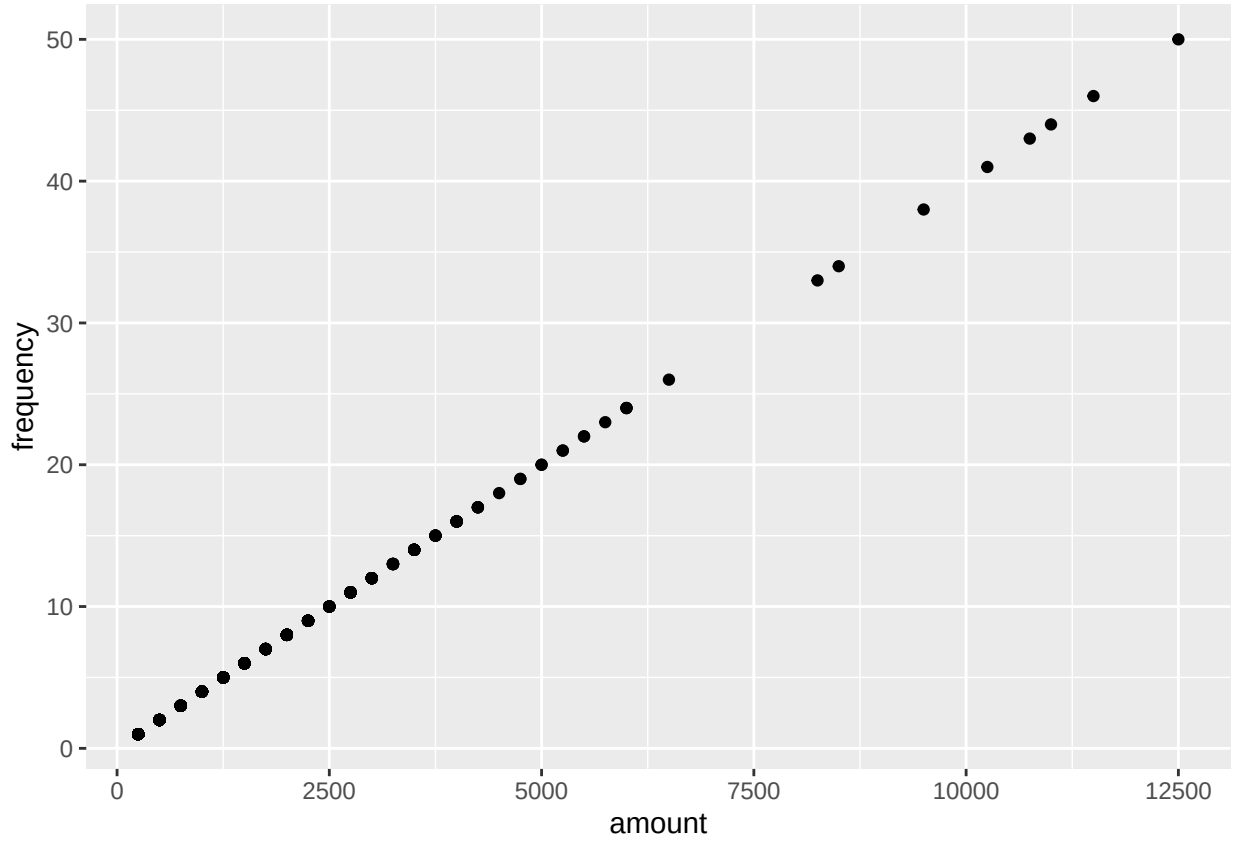
```
# amount GLM model
amount_model <- glm(donation ~ amount, data = donations, family = binomial)
summary(amount_model)
```

```
##
## Call:
## glm(formula = donation ~ amount, family = binomial, data = donations)
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept) -1.638e+00  1.282e-01 -12.774  < 2e-16 ***
## amount       3.175e-04  6.056e-05   5.243 1.58e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 820.89  on 747  degrees of freedom
## Residual deviance: 789.12  on 746  degrees of freedom
## AIC: 793.12
##
## Number of Fisher Scoring iterations: 4
```

```
# comparing using AIC
AIC(freq_model, amount_model)
```

```
##              df      AIC
## freq_model    2 793.1162
## amount_model  2 793.1162
```

```
ggplot(donations, aes(x = amount, y = frequency)) +
  geom_point() +
  labs(
    x = "amount",
    y = "frequency"
    )
```



Frequency and amount variables are highly correlated, which is logical as the amount of blood taken is fixed and its sum depends only on the frequency on donation We don't need both variables in our model, as multicollinearity decreases its accuracy

b. different link functions

```
recency_logit <- glm(donation ~ recency, data = donations, family = binomial(link = "logit"))
summary(recency_logit)
```

```
##
## Call:
## glm(formula = donation ~ recency, family = binomial(link = "logit"),
##     data = donations)
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  -0.2032     0.1363  -1.492    0.136
## recency      -0.1250     0.0165  -7.573 3.64e-14 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 820.89  on 747  degrees of freedom
## Residual deviance: 743.55  on 746  degrees of freedom
```

```
## AIC: 747.55
##
## Number of Fisher Scoring iterations: 5
```

```
recency_probit <- glm(donation ~ recency, data = donations, family = binomial(link = "probit"))
summary(recency_probit)
```

```
##
## Call:
## glm(formula = donation ~ recency, family = binomial(link = "probit"),
##     data = donations)
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept) -0.159627   0.081631  -1.955   0.0505 .
## recency     -0.069216   0.008766  -7.896 2.89e-15 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 820.89  on 747  degrees of freedom
## Residual deviance: 744.33  on 746  degrees of freedom
## AIC: 748.33
##
## Number of Fisher Scoring iterations: 5
```

```
recency_cauchit <- glm(donation ~ recency, data = donations, family = binomial(link = "cauchit"))
summary(recency_probit)
```

```
##
## Call:
## glm(formula = donation ~ recency, family = binomial(link = "probit"),
##     data = donations)
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept) -0.159627   0.081631  -1.955   0.0505 .
## recency     -0.069216   0.008766  -7.896 2.89e-15 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 820.89  on 747  degrees of freedom
## Residual deviance: 744.33  on 746  degrees of freedom
## AIC: 748.33
##
## Number of Fisher Scoring iterations: 5
```

```
recency_log <- glm(donation ~ recency, data = donations, family = binomial(link = "log"))
summary(recency_probit)
```

```
##
## Call:
## glm(formula = donation ~ recency, family = binomial(link = "probit"),
##     data = donations)
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept) -0.159627   0.081631  -1.955   0.0505 .
## recency     -0.069216   0.008766  -7.896 2.89e-15 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 820.89  on 747  degrees of freedom
## Residual deviance: 744.33  on 746  degrees of freedom
## AIC: 748.33
##
## Number of Fisher Scoring iterations: 5
```

```
recency_cloglog <- glm(donation ~ recency, data = donations, family = binomial(link = "cloglog"))
summary(recency_cloglog)
```

```
##
## Call:
## glm(formula = donation ~ recency, family = binomial(link = "cloglog"),
##     data = donations)
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept) -0.47661    0.11101  -4.293 1.76e-05 ***
## recency     -0.11089    0.01482  -7.483 7.27e-14 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 820.89  on 747  degrees of freedom
## Residual deviance: 743.47  on 746  degrees of freedom
## AIC: 747.47
##
## Number of Fisher Scoring iterations: 5
```

```
AIC(recency_logit, recency_probit, recency_log, recency_cauchit, recency_cloglog)
```

```
##                 df      AIC
## recency_logit    2 747.5547
## recency_probit   2 748.3307
## recency_log      2 747.6188
## recency_cauchit  2 749.4959
## recency_cloglog  2 747.4671
```

While comparing using AIC, the lowest result is obtained by complementary log-log link, however, the results are not really different from each other.

c. best prediction

- train and test data

```
set.seed(1122)
sample_size <- 374
train_indices <- sample(1:nrow(donations), sample_size)
donations_train <- donations[train_indices, ]
donations_test <- donations[-train_indices, ]
```

- model fit

We will not include amount there, as it is highly correlated with frequency

```
donations_model <- glm(donation ~ recency + frequency + time,
  data = donations_train,
  family = binomial)
summary(donations_model)
```

```
##
## Call:
## glm(formula = donation ~ recency + frequency + time, family = binomial,
##     data = donations_train)
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept) -0.351257   0.249155  -1.410 0.158600
## recency     -0.094233   0.024035  -3.921 8.83e-05 ***
## frequency    0.131617   0.037421   3.517 0.000436 ***
## time        -0.025462   0.008412  -3.027 0.002471 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 415.04  on 373  degrees of freedom
## Residual deviance: 356.99  on 370  degrees of freedom
## AIC: 364.99
##
## Number of Fisher Scoring iterations: 5
```

```
pred_donations <- predict(donations_model, newx = donations_test)
```

- classification

```
treshhold <- 0.5
pred_donations_class <- ifelse(pred_donations < treshhold, 0, 1)
real_donations_class <- donations_test$donation
CE <- mean(abs(real_donations_class - pred_donations_class))
CE
```

```
## [1] 0.2566845
```

Exercise 7

```
student <- read.csv('student-mat.csv')
head(student)
```

```
##   school sex age address famsize Pstatus Medu Fedu     Mjob     Fjob     reason
## 1     GP   F  18       U     GT3       A    4    4  at_home  teacher     course
## 2     GP   F  17       U     GT3       T    1    1  at_home    other     course
## 3     GP   F  15       U     LE3       T    1    1  at_home    other      other
## 4     GP   F  15       U     GT3       T    4    2   health services       home
## 5     GP   F  16       U     GT3       T    3    3    other    other       home
## 6     GP   M  16       U     LE3       T    4    3 services    other reputation
##   guardian traveltime studytime failures schoolsup famsup paid activities
## 1   mother          2         2        0       yes     no   no         no
## 2   father          1         2        0        no    yes   no         no
## 3   mother          1         2        3       yes     no  yes         no
## 4   mother          1         3        0        no    yes  yes        yes
## 5   father          1         2        0        no    yes  yes         no
## 6   mother          1         2        0        no    yes  yes        yes
##   nursery higher internet romantic famrel freetime goout Dalc Walc health
## 1     yes    yes       no       no      4        3     4    1    1      3
## 2      no    yes      yes       no      5        3     3    1    1      3
## 3     yes    yes      yes       no      4        3     2    2    3      3
## 4     yes    yes      yes      yes      3        2     2    1    1      5
## 5     yes    yes       no       no      4        3     2    1    2      5
## 6     yes    yes      yes       no      5        4     2    1    2      5
##   absences G1 G2 G3
## 1        6  5  6  6
## 2        4  5  5  6
## 3       10  7  8 10
## 4        2 15 14 15
## 5        4  6 10 10
## 6       10 15 15 15
```

```
summary(student)
```

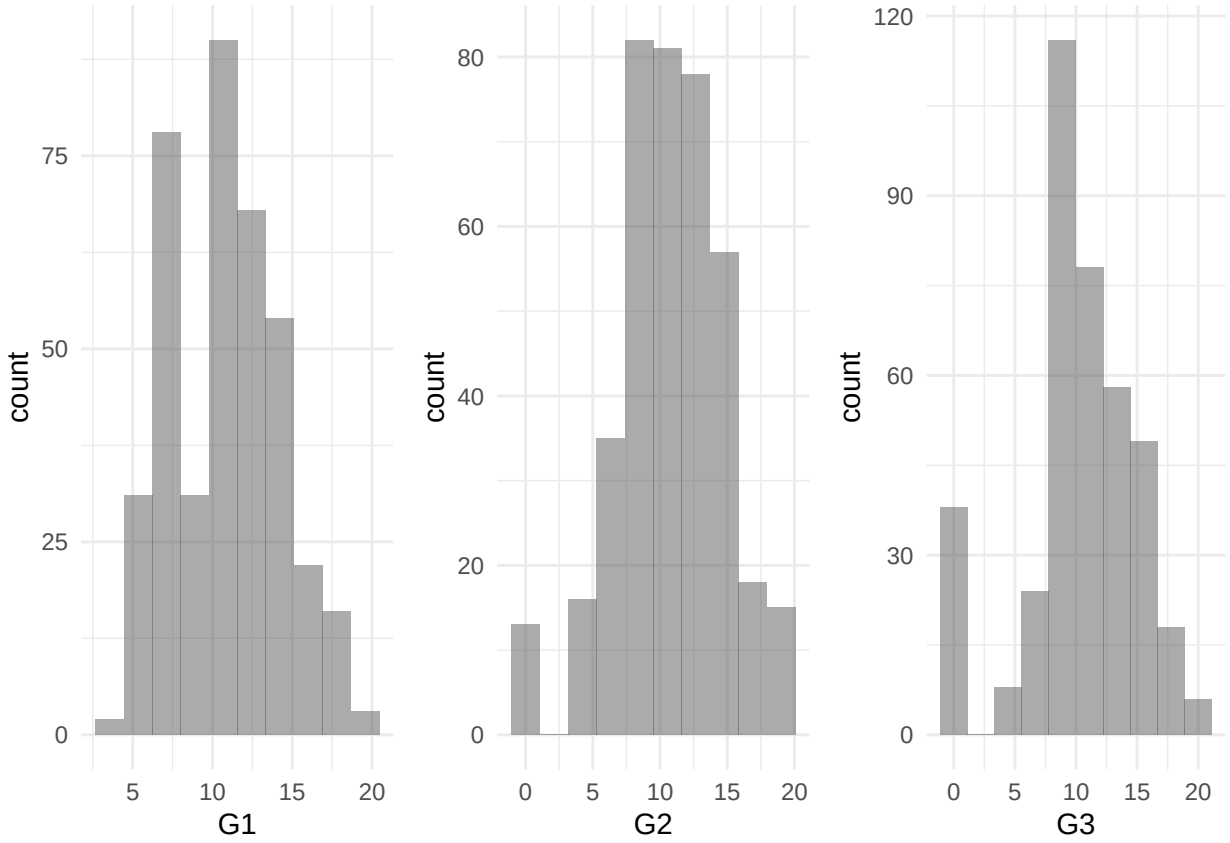
```
##     school              sex                 age         address
##  Length:395         Length:395         Min.   :15.0   Length:395
##  Class :character   Class :character   1st Qu.:16.0   Class :character
##  Mode  :character   Mode  :character   Median :17.0   Mode  :character
##                                        Mean   :16.7
##                                        3rd Qu.:18.0
##                                        Max.   :22.0
##    famsize            Pstatus              Medu            Fedu
##  Length:395         Length:395         Min.   :0.000   Min.   :0.000
##  Class :character   Class :character   1st Qu.:2.000   1st Qu.:2.000
##  Mode  :character   Mode  :character   Median :3.000   Median :2.000
##                                        Mean   :2.749   Mean   :2.522
##                                        3rd Qu.:4.000   3rd Qu.:3.000
##                                        Max.   :4.000   Max.   :4.000
##      Mjob               Fjob              reason            guardian
##  Length:395         Length:395         Length:395         Length:395
```

```
##  Class :character   Class :character   Class :character   Class :character
##  Mode  :character   Mode  :character   Mode  :character   Mode  :character
##
##
##
##    traveltime      studytime        failures       schoolsup
##  Min.   :1.000   Min.   :1.000   Min.   :0.0000   Length:395
##  1st Qu.:1.000   1st Qu.:1.000   1st Qu.:0.0000   Class :character
##  Median :1.000   Median :2.000   Median :0.0000   Mode  :character
##  Mean   :1.448   Mean   :2.035   Mean   :0.3342
##  3rd Qu.:2.000   3rd Qu.:2.000   3rd Qu.:0.0000
##  Max.   :4.000   Max.   :4.000   Max.   :3.0000
##     famsup              paid            activities          nursery
##  Length:395         Length:395         Length:395         Length:395
##  Class :character   Class :character   Class :character   Class :character
##  Mode  :character   Mode  :character   Mode  :character   Mode  :character
##
##
##
##     higher            internet           romantic             famrel
##  Length:395         Length:395         Length:395         Min.   :1.000
##  Class :character   Class :character   Class :character   1st Qu.:4.000
##  Mode  :character   Mode  :character   Mode  :character   Median :4.000
##                                                           Mean   :3.944
##                                                           3rd Qu.:5.000
##                                                           Max.   :5.000
##     freetime         goout            Dalc            Walc
##  Min.   :1.000   Min.   :1.000   Min.   :1.000   Min.   :1.000
##  1st Qu.:3.000   1st Qu.:2.000   1st Qu.:1.000   1st Qu.:1.000
##  Median :3.000   Median :3.000   Median :1.000   Median :2.000
##  Mean   :3.235   Mean   :3.109   Mean   :1.481   Mean   :2.291
##  3rd Qu.:4.000   3rd Qu.:4.000   3rd Qu.:2.000   3rd Qu.:3.000
##  Max.   :5.000   Max.   :5.000   Max.   :5.000   Max.   :5.000
##      health         absences            G1              G2
##  Min.   :1.000   Min.   : 0.000   Min.   : 3.00   Min.   : 0.00
##  1st Qu.:3.000   1st Qu.: 0.000   1st Qu.: 8.00   1st Qu.: 9.00
##  Median :4.000   Median : 4.000   Median :11.00   Median :11.00
##  Mean   :3.554   Mean   : 5.709   Mean   :10.91   Mean   :10.71
##  3rd Qu.:5.000   3rd Qu.: 8.000   3rd Qu.:13.00   3rd Qu.:13.00
##  Max.   :5.000   Max.   :75.000   Max.   :19.00   Max.   :19.00
##        G3
##  Min.   : 0.00
##  1st Qu.: 8.00
##  Median :11.00
##  Mean   :10.42
##  3rd Qu.:14.00
##  Max.   :20.00
```

a. G1, G2, G3 distributions

- histograms

```
G1_hist <- ggplot(student, aes(x = G1)) +
  geom_histogram(position = "identity", alpha = .5, bins = 10) +
  theme_minimal()

G2_hist <- ggplot(student, aes(x = G2)) +
  geom_histogram(position = "identity", alpha = .5, bins = 10) +
  theme_minimal()

G3_hist <- ggplot(student, aes(x = G3)) +
  geom_histogram(position = "identity", alpha = .5, bins = 10) +
  theme_minimal()

ggarrange(G1_hist, G2_hist, G3_hist, ncol = 3, nrow = 1)
```

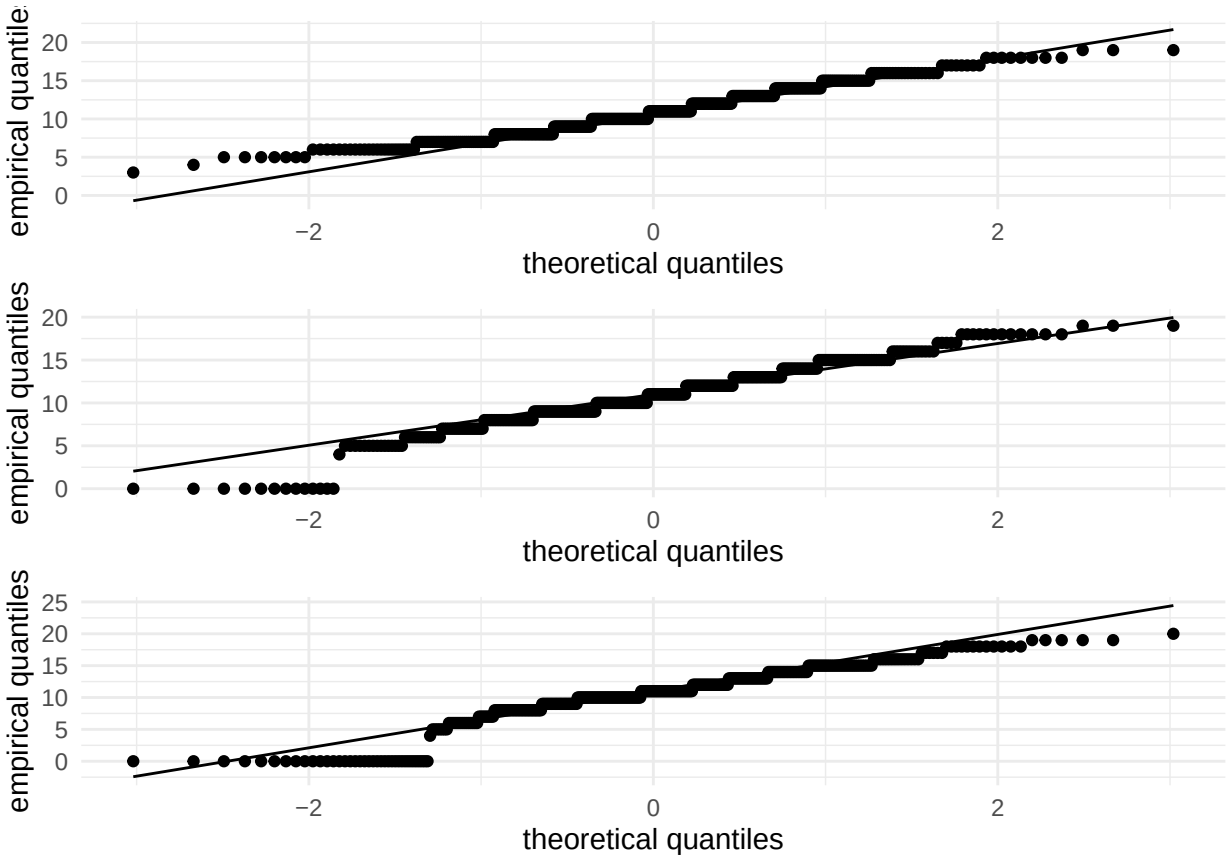


The histograms do not seem to follow the normal distribution We can check that using QQ-plots as well

- QQ plots

```
G1_qq <- ggplot(student, aes(sample = G1)) +
  stat_qq() +
  stat_qq_line() +
  theme_minimal() +
  labs (
    x = "theoretical quantiles",
    y = "empirical quantiles"
```

```
  )

G2_qq <- ggplot(student, aes(sample = G2)) +
  stat_qq() +
  stat_qq_line() +
  theme_minimal() +
  labs (
    x = "theoretical quantiles",
    y = "empirical quantiles"
  )

G3_qq <- ggplot(student, aes(sample = G3)) +
  stat_qq() +
  stat_qq_line() +
  theme_minimal() +
  labs (
    x = "theoretical quantiles",
    y = "empirical quantiles"
  )

ggarrange(G1_qq, G2_qq, G3_qq, ncol = 1, nrow = 3)
```



G1, G2, G3 do not seem like they follow the normal distribution

- Poisson ?

To check whether they may follow the Poisson distribution we can check if there is a significant difference between mean and variance:

```
var(student$G1) / mean(student$G1)
```

```
## [1] 1.009918
```

```
var(student$G2) / mean(student$G2)
```

```
## [1] 1.32061
```

```
var(student$G3) / mean(student$G3)
```

```
## [1] 2.015289
```

Looks like G1 and G2 may follow Poisson distribution (they're close to be equal) while G3 probably is not However, we cannot state that there is overdispersion in the distribution of these variables, as variance is not highly bigger than the mean I also do not think that there are any anomalies in their distributions, as looking at the histograms they seem pretty close to Poisson

b. glm

Fitting the model without G2 and G3 - these grades cannot influence the first period grade, as they haven't happened yet

```
model_1 <- glm(G1 ~. -G2 -G3, data = student)
summary(model_1)
```

```
##
## Call:
## glm(formula = G1 ~ . - G2 - G3, data = student)
##
## Coefficients:
##                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)      11.375064   3.113004   3.654 0.000297 ***
## schoolMS          0.009965   0.549925   0.018 0.985553
## sexM              0.894290   0.347385   2.574 0.010448 *
## age              -0.070082   0.150905  -0.464 0.642639
## addressU          0.150710   0.405805   0.371 0.710571
## famsizeLE3        0.429175   0.339195   1.265 0.206602
## PstatusT          0.154297   0.502913   0.307 0.759170
## Medu              0.117943   0.224515   0.525 0.599688
## Fedu              0.143774   0.192870   0.745 0.456496
## Mjobhealth        0.926137   0.776837   1.192 0.233983
## Mjobother        -0.782287   0.495455  -1.579 0.115244
## Mjobservices      0.466532   0.554282   0.842 0.400529
## Mjobteacher      -0.922790   0.721274  -1.279 0.201596
## Fjobhealth       -0.553377   0.998994  -0.554 0.579973
## Fjobother        -1.134849   0.710736  -1.597 0.111217
## Fjobservices     -0.994008   0.734310  -1.354 0.176705
## Fjobteacher       1.187017   0.900744   1.318 0.188414
```

```
## reasonhome        0.165602   0.384744   0.430 0.667150
## reasonother      -0.181207   0.567991  -0.319 0.749891
## reasonreputation  0.444004   0.400557   1.108 0.268411
## guardianmother    0.050219   0.379042   0.132 0.894673
## guardianother     0.866380   0.694357   1.248 0.212947
## traveltime       -0.025119   0.235489  -0.107 0.915112
## studytime         0.604725   0.199842   3.026 0.002659 **
## failures         -1.314183   0.231280  -5.682 2.77e-08 ***
## schoolsupyes     -2.155394   0.463335  -4.652 4.65e-06 ***
## famsupyes        -0.978681   0.332560  -2.943 0.003466 **
## paidyes          -0.102389   0.331906  -0.308 0.757892
## activitiesyes    -0.052728   0.309114  -0.171 0.864652
## nurseryyes        0.029587   0.381623   0.078 0.938245
## higheryes         1.140610   0.748777   1.523 0.128575
## internetyes       0.255412   0.430423   0.593 0.553293
## romanticyes      -0.211223   0.326001  -0.648 0.517455
## famrel            0.025733   0.170852   0.151 0.880363
## freetime          0.254817   0.164896   1.545 0.123161
## goout            -0.413594   0.155971  -2.652 0.008367 **
## Dalc             -0.063146   0.229869  -0.275 0.783703
## Walc             -0.025339   0.172300  -0.147 0.883164
## health           -0.167531   0.111859  -1.498 0.135102
## absences          0.012277   0.020124   0.610 0.542204
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for gaussian family taken to be 8.144982)
##
##     Null deviance: 4340.7  on 394  degrees of freedom
## Residual deviance: 2891.5  on 355  degrees of freedom
## AIC: 1989.3
##
## Number of Fisher Scoring iterations: 2
```

At the significance level 0.05 the following variables are significant for G1 - first period grade:

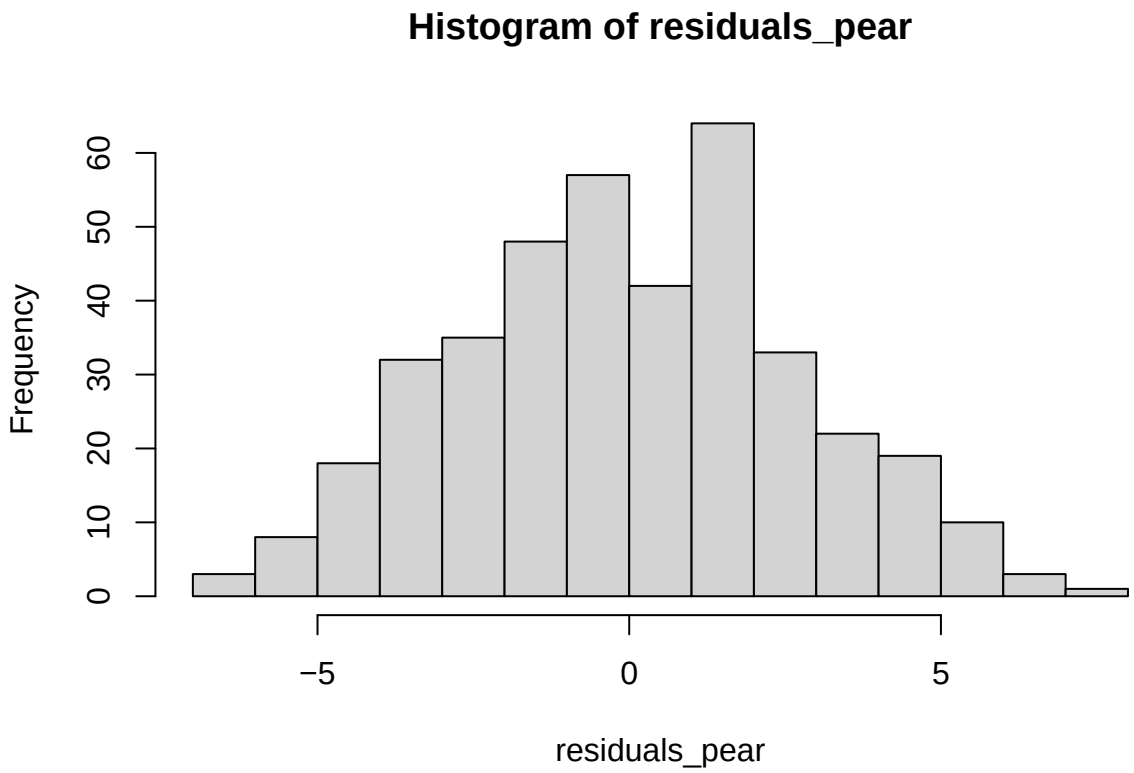
- sexM - being a male increases the chances of getting higher grade (that's sad)
- studytime - with more time put into studies, the grade gets higher
- failures - the number of failures has negative impact on the grade
- schoolsup - having extra educational support has strong negative influence
- famsup - having extra family support has negative influence

side note: I guess here we see the opposite causaition: usually children who are not very successful in the education require additional support

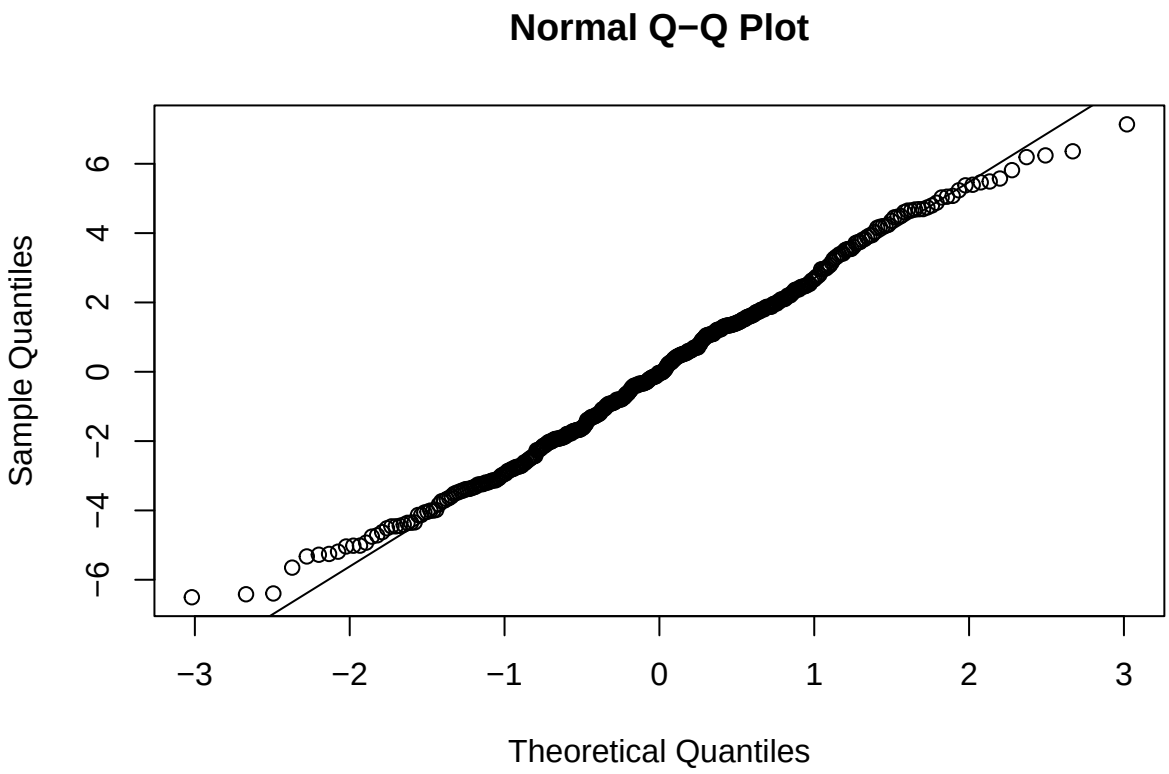
- goout - with increasing frequency of going out the grade tends to decrease

The residual deviance is smaller than null deviance which suggests that the model is better than the default one However, the deviance number is still large which means we do not explain the data with our model perfectly well, and it could be pred_donations_improved

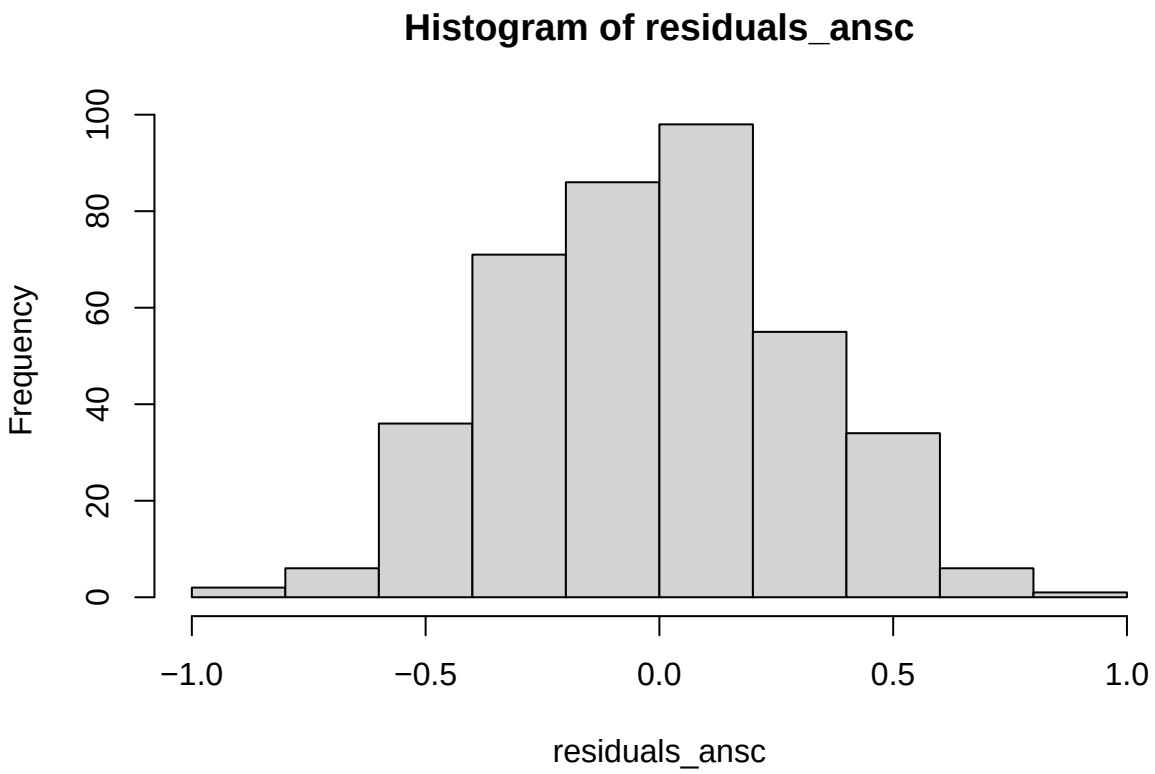
```r
# Pearson residuals
residuals_pear <- residuals(model_1, type = "pearson")
hist(residuals_pear)
```



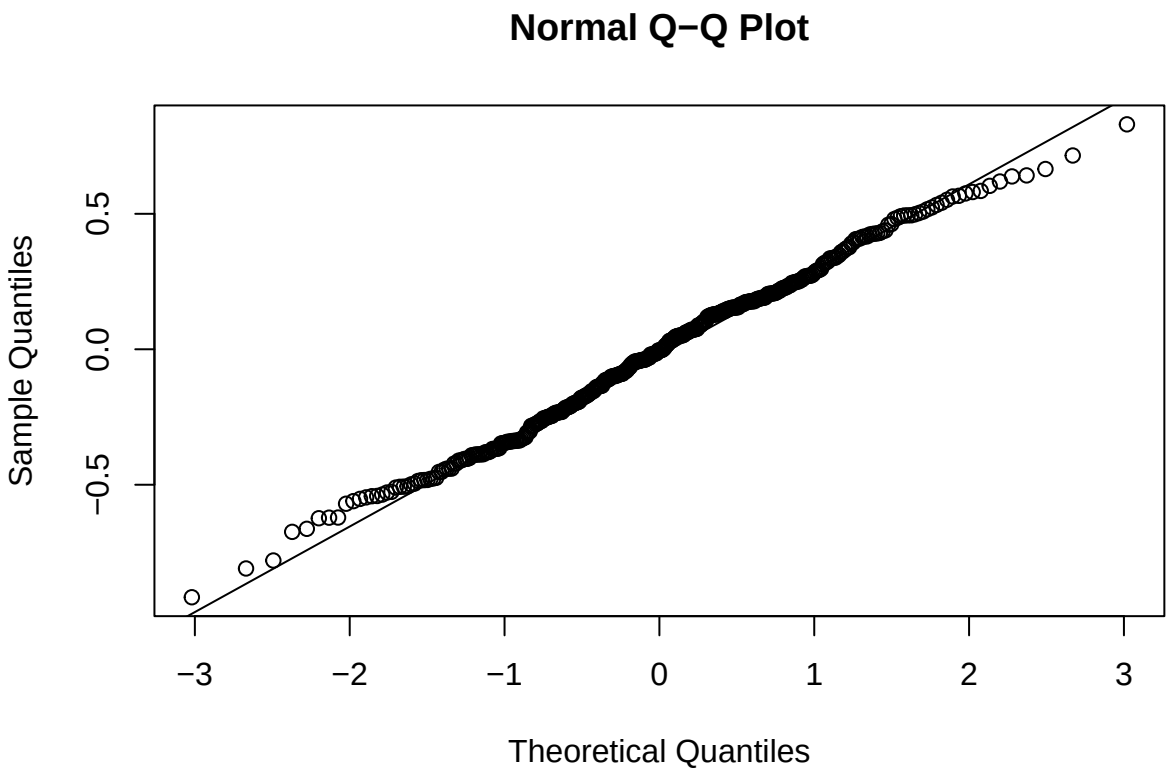

```r
qqnorm(residuals_pear)
qqline(residuals_pear)
```




```
# Anscombe residuals

phi <- deviance(model_1) / df.residual(model_1) # estimated over-dispersion

residuals_ansc <- anscombe.residuals(model_1, phi)
hist(residuals_ansc)
```




```
qqnorm(residuals_ansc)
qqline(residuals_ansc)
```



These residuals do not seem to follow the normal distribution, so we probably didn't capture all the influencing variables

c. less covariates

```
model_2 <- glm(G1 ~ sex + Fedu + studytime + failures + schoolsup + famsup + goout, data = student)
summary(model_2)
```

```
##
## Call:
## glm(formula = G1 ~ sex + Fedu + studytime + failures + schoolsup +
##     famsup + goout, data = student)
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)   10.4184     0.7355  14.165  < 2e-16 ***
## sexM           0.7544     0.3149   2.396 0.017068 *
## Fedu           0.4613     0.1432   3.221 0.001387 **
## studytime      0.6476     0.1883   3.439 0.000649 ***
## failures      -1.2677     0.2099  -6.039 3.65e-09 ***
## schoolsupyes  -1.9928     0.4450  -4.478 9.93e-06 ***
## famsupyes     -0.7874     0.3142  -2.506 0.012618 *
## goout         -0.3811     0.1341  -2.841 0.004739 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
##
## (Dispersion parameter for gaussian family taken to be 8.547329)
##
##     Null deviance: 4340.7  on 394  degrees of freedom
## Residual deviance: 3307.8  on 387  degrees of freedom
## AIC: 1978.4
##
## Number of Fisher Scoring iterations: 2
```

All the covariates are significant on alpha = 0.05

- sexM - being a male increases the chances of getting higher grade
- Fedu - with higher level of education of father the grades also increase
- studytime - with more time put into studies, the grade gets higher
- failures - the number of failures has negative impact on the grade
- schoolsup - having extra educational support has strong negative influence
- famsup - having extra family support has negative influence side note here too: I guess here we see the opposite causaition: usually children who are not very successful in the education require additional support
- goout - with increasing frequency of going out the grade tends to decrease

Again, the residual deviance is smaller than null, so model performs better than null one, but not really much better

- deviance test: model_1 and model_2

```
anova(model_1, model_2)
```

```
## Analysis of Deviance Table
##
## Model 1: G1 ~ (school + sex + age + address + famsize + Pstatus + Medu +
##     Fedu + Mjob + Fjob + reason + guardian + traveltime + studytime +
##     failures + schoolsup + famsup + paid + activities + nursery +
##     higher + internet + romantic + famrel + freetime + goout +
##     Dalc + Walc + health + absences + G2 + G3) - G2 - G3
## Model 2: G1 ~ sex + Fedu + studytime + failures + schoolsup + famsup +
##     goout
##   Resid. Df Resid. Dev  Df Deviance      F  Pr(>F)
## 1       355     2891.5
## 2       387     3307.8 -32  -416.35 1.5974 0.02364 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The significant p-value (0.02364) suggests that Model 1 fits the data better than Model 2. Which means that the variables we excluded still explain a lot of variance and we should not omit them

- goout -> Walc

```
model_3 <- glm(G1 ~ sex + Fedu + studytime + failures + schoolsup + famsup + Walc, data = student)
summary(model_3)
```

```
##
## Call:
## glm(formula = G1 ~ sex + Fedu + studytime + failures + schoolsup +
##     famsup + Walc, data = student)
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)   10.0375     0.7016  14.306  < 2e-16 ***
## sexM           0.8575     0.3223   2.661  0.00813 **
## Fedu           0.4344     0.1433   3.031  0.00260 **
## studytime      0.5860     0.1916   3.059  0.00238 **
## failures      -1.2951     0.2100  -6.166 1.76e-09 ***
## schoolsupyes  -2.0073     0.4470  -4.490 9.40e-06 ***
## famsupyes     -0.7968     0.3154  -2.526  0.01193 *
## Walc          -0.2805     0.1224  -2.292  0.02242 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for gaussian family taken to be 8.608663)
##
##     Null deviance: 4340.7  on 394  degrees of freedom
## Residual deviance: 3331.6  on 387  degrees of freedom
## AIC: 1981.2
##
## Number of Fisher Scoring iterations: 2
```

```
anova(model_2, model_3)
```

```
## Analysis of Deviance Table
##
## Model 1: G1 ~ sex + Fedu + studytime + failures + schoolsup + famsup +
##     goout
## Model 2: G1 ~ sex + Fedu + studytime + failures + schoolsup + famsup +
##     Walc
##   Resid. Df Resid. Dev Df Deviance F Pr(>F)
## 1       387     3307.8
## 2       387     3331.6  0  -23.736
```

Weekend alcohol consumption is also a significant predictor, however, it doesn't imrpove the fit compared to the model 2 - it even makes it worse (higher residual deviance)

Exercise 9

```
data(iris)
iris <- iris
head(iris)
```

```
##   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
## 1          5.1         3.5          1.4         0.2  setosa
## 2          4.9         3.0          1.4         0.2  setosa
## 3          4.7         3.2          1.3         0.2  setosa
```

```
## 4          4.6         3.1          1.5         0.2  setosa
## 5          5.0         3.6          1.4         0.2  setosa
## 6          5.4         3.9          1.7         0.4  setosa
```

```
summary(iris)
```

```
##   Sepal.Length    Sepal.Width     Petal.Length    Petal.Width
##  Min.   :4.300   Min.   :2.000   Min.   :1.000   Min.   :0.100
##  1st Qu.:5.100   1st Qu.:2.800   1st Qu.:1.600   1st Qu.:0.300
##  Median :5.800   Median :3.000   Median :4.350   Median :1.300
##  Mean   :5.843   Mean   :3.057   Mean   :3.758   Mean   :1.199
##  3rd Qu.:6.400   3rd Qu.:3.300   3rd Qu.:5.100   3rd Qu.:1.800
##  Max.   :7.900   Max.   :4.400   Max.   :6.900   Max.   :2.500
##        Species
##  setosa    :50
##  versicolor:50
##  virginica :50
##
##
##
```

a. reduced dataset

```
iris_cut <- iris %>%
  select(., -Species) %>%
  as.matrix()
pca <- prcomp(iris_cut, scale. = FALSE)
summary(pca)
```

```
## Importance of components:
##                           PC1     PC2    PC3     PC4
## Standard deviation     2.0563 0.49262 0.2797 0.15439
## Proportion of Variance 0.9246 0.05307 0.0171 0.00521
## Cumulative Proportion  0.9246 0.97769 0.9948 1.00000
```

According to the summary, PC1 and PC2 explain $92.46\% + 5.307\% = 97.767\%$ proportion of the variance

- loadings

Loadings are the eigenvectors of the covariance matrix

```
cov_iris <- cov(iris_cut) # covariance matrix
eigen_decomposition <- eigen(cov_iris)
loadings <- eigen_decomposition$vectors
loadings
```

```
##             [,1]        [,2]        [,3]       [,4]
## [1,]  0.36138659 -0.65658877  0.58202985  0.3154872
## [2,] -0.08452251 -0.73016143 -0.59791083 -0.3197231
## [3,]  0.85667061  0.17337266 -0.07623608 -0.4798390
## [4,]  0.35828920  0.07548102 -0.54583143  0.7536574
```

- scores

Scores are projections onto loadings

```
scores <- iris_cut %*% loadings
scores
```

```
##            [,1]      [,2]        [,3]         [,4]
##   [1,] 2.818240 -5.646350  0.65976754 -0.031089276
##   [2,] 2.788223 -5.149951  0.84231699  0.065674837
##   [3,] 2.613375 -5.182003  0.61395246 -0.013383323
##   [4,] 2.757022 -5.008654  0.60029334 -0.108927529
##   [5,] 2.773649 -5.653707  0.54177348 -0.094610305
##   [6,] 3.221505 -6.068283  0.46317506 -0.057552570
##   [7,] 2.681827 -5.237491  0.37396056 -0.081494819
##   [8,] 2.876220 -5.490338  0.65373203 -0.078649583
##   [9,] 2.615982 -4.748641  0.61109315 -0.060096449
##  [10,] 2.829609 -5.213178  0.82948544 -0.089647114
##  [11,] 2.995418 -5.972021  0.70717073 -0.048371637
##  [12,] 2.889610 -5.341683  0.52970246 -0.189730921
##  [13,] 2.716256 -5.091841  0.83869715 -0.041239624
##  [14,] 2.278561 -4.815558  0.57055304 -0.055031525
##  [15,] 2.857615 -6.505717  0.78348024  0.125858005
##  [16,] 3.116326 -6.665015  0.35407581 -0.026800167
##  [17,] 2.878837 -6.137632  0.49366949  0.134383024
##  [18,] 2.854068 -5.638802  0.60518440  0.044276467
##  [19,] 3.302545 -6.199792  0.75215824 -0.006299845
##  [20,] 2.914379 -5.840513  0.41818754 -0.099624363
##  [21,] 3.192109 -5.718299  0.87129676 -0.048422504
##  [22,] 2.958660 -5.759949  0.42339548  0.007713690
##  [23,] 2.286426 -5.460421  0.33945597 -0.028869588
##  [24,] 3.199632 -5.425661  0.59272946  0.115000876
##  [25,] 3.146611 -5.289671  0.50683163 -0.333682617
##  [26,] 2.995696 -5.180936  0.88527276  0.001255759
##  [27,] 3.033545 -5.457904  0.53694214  0.024098003
##  [28,] 2.940045 -5.694671  0.71034692 -0.047524455
##  [29,] 2.862830 -5.638993  0.77776161  0.032431754
##  [30,] 2.870376 -5.129991  0.59108164 -0.157335019
##  [31,] 2.914967 -5.122634  0.70907571 -0.093813990
##  [32,] 3.092433 -5.737877  0.77737769  0.198276779
##  [33,] 2.853503 -6.140316  0.40618357 -0.314724060
##  [34,] 2.903628 -6.420098  0.47404190 -0.128700571
##  [35,] 2.865438 -5.205630  0.77490230 -0.014281372
##  [36,] 2.636123 -5.396317  0.79618502  0.129246733
##  [37,] 2.877127 -5.926323  0.90020309  0.143089500
##  [38,] 2.701681 -5.595596  0.53815363 -0.201524767
##  [39,] 2.521863 -4.838994  0.55892567 -0.044084860
##  [40,] 2.912359 -5.555996  0.71193502 -0.047100864
##  [41,] 2.732263 -5.590480  0.55460502  0.060711646
##  [42,] 2.652996 -4.385992  0.98108309  0.286635774
##  [43,] 2.504959 -4.985027  0.43934350 -0.108029481
##  [44,] 3.096751 -5.515824  0.36798477  0.142857178
##  [45,] 3.292876 -5.763616  0.33310997 -0.216194215
##  [46,] 2.787914 -5.076744  0.72953086  0.109491861
```

```
##  [47,] 2.964217 -5.830724  0.46514708 -0.222974004
##  [48,] 2.662903 -5.099007  0.54812587 -0.092915941
##  [49,] 2.959279 -5.906363  0.64896774 -0.079920357
##  [50,] 2.799005 -5.434659  0.72114672  0.001306626
##  [51,] 6.787191 -6.012113  1.03842074 -0.014826425
##  [52,] 6.434854 -5.645286  0.64986690 -0.032785201
##  [53,] 6.966667 -5.831215  0.97017848 -0.035004889
##  [54,] 5.685683 -4.498994  0.81144411  0.060215127
##  [55,] 6.590468 -5.401543  0.93961061  0.078668861
##  [56,] 6.144034 -4.908706  0.59077663 -0.276468479
##  [57,] 6.597426 -5.610421  0.46204248 -0.116908285
##  [58,] 4.753242 -4.322062  0.61954980 -0.051259435
##  [59,] 6.546497 -5.555314  1.04718880 -0.072486215
##  [60,] 5.493620 -4.603871  0.35071128 -0.038970631
##  [61,] 4.994524 -4.060981  0.90166990  0.012210728
##  [62,] 6.014064 -5.222971  0.50130497  0.017367520
##  [63,] 5.767342 -4.776916  1.32599955  0.023833807
##  [64,] 6.487300 -5.202135  0.69396712 -0.202847967
##  [65,] 5.328440 -5.072098  0.54139503  0.091865579
##  [66,] 6.430226 -5.794132  0.94647369  0.066451424
##  [67,] 6.162649 -4.973983  0.30382519 -0.221230334
##  [68,] 5.738470 -4.993342  0.90301455 -0.247109082
##  [69,] 6.447099 -4.783807  1.13137176  0.223840464
##  [70,] 5.547592 -4.743118  0.86685482 -0.074928360
##  [71,] 6.618648 -5.242336  0.17223172 -0.108383266
##  [72,] 5.860254 -5.258028  0.86170660  0.089645891
##  [73,] 6.800549 -4.999165  0.97970707 -0.032463342
##  [74,] 6.424094 -5.144215  0.86292449 -0.321607142
##  [75,] 6.217218 -5.476009  0.95365365  0.008368043
##  [76,] 6.402540 -5.655457  0.94806179  0.066875015
##  [77,] 6.834390 -5.571393  1.15355550  0.001981479
##  [78,] 7.060167 -5.594448  0.79677370  0.036617569
##  [79,] 6.315656 -5.163602  0.59642821 -0.063063147
##  [80,] 5.196781 -4.958690  0.95034430  0.041217901
##  [81,] 5.434239 -4.621780  0.87606653 -0.026520871
##  [82,] 5.312743 -4.646666  0.93827328 -0.053902714
##  [83,] 5.638794 -5.012920  0.80909548 -0.000409800
##  [84,] 6.882392 -4.905998  0.61568559 -0.211656176
##  [85,] 6.090372 -4.842665  0.18741922 -0.284327773
##  [86,] 6.309223 -5.521135  0.24288965 -0.147558956
##  [87,] 6.723056 -5.734572  0.86901973 -0.002134530
##  [88,] 6.317460 -4.954916  1.24657356  0.120669287
##  [89,] 5.748323 -5.058428  0.44348590 -0.180026225
##  [90,] 5.668778 -4.645026  0.69186194 -0.003729493
##  [91,] 5.967165 -4.656241  0.65615957 -0.303003141
##  [92,] 6.393180 -5.292488  0.64179965 -0.186836379
##  [93,] 5.732913 -4.922567  0.86126296 -0.016421388
##  [94,] 4.797833 -4.314704  0.73754387  0.012261594
##  [95,] 5.859347 -4.822042  0.61523555 -0.132093192
##  [96,] 5.834300 -5.114298  0.54864843 -0.271827147
##  [97,] 5.878581 -5.033734  0.55385637 -0.164489094
##  [98,] 6.144941 -5.344691  0.83724768 -0.054729396
##  [99,] 4.595895 -4.570859  0.64445236  0.199183131
## [100,] 5.801366 -4.978055  0.62127106 -0.084532885
```

```
## [101,] 8.033558 -5.317103 -0.12831271 -0.062407286
## [102,] 6.917601 -4.752036  0.33553019 -0.048656387
## [103,] 8.119041 -5.670856  0.74264060  0.032420328
## [104,] 7.473896 -5.147225  0.52342805 -0.270142647
## [105,] 7.852371 -5.286692  0.34646315 -0.033522346
## [106,] 8.899404 -5.877789  0.98029027 -0.145723366
## [107,] 6.023597 -4.134194  0.08619342 -0.131478332
## [108,] 8.434952 -5.682453  1.05209265 -0.290542745
## [109,] 7.823594 -5.083121  0.98015711 -0.112026326
## [110,] 8.419116 -6.109745  0.20851730  0.077630358
## [111,] 7.164139 -5.569181  0.38941253  0.087688839
## [112,] 7.305767 -5.111315  0.66950088  0.044668132
## [113,] 7.667957 -5.543228  0.59852607  0.129709765
## [114,] 6.848529 -4.550134  0.34994983  0.107089156
## [115,] 7.088293 -4.787312  0.00282339  0.296200016
## [116,] 7.406822 -5.446203  0.15221290  0.186269550
## [117,] 7.452054 -5.368896  0.58766655 -0.191033620
## [118,] 8.989420 -6.502692  0.49795784 -0.342571286
## [119,] 9.298011 -5.584276  1.14562048  0.020494384
## [120,] 6.803157 -4.565803  0.97684775 -0.079176468
## [121,] 7.930183 -5.705149  0.41273339  0.152077552
## [122,] 6.701366 -4.720861  0.11999721  0.027607404
## [123,] 9.002285 -5.787627  1.20503496 -0.173579667
## [124,] 6.891131 -5.122553  0.69637547  0.129689265
## [125,] 7.777796 -5.661943  0.34570262 -0.093723682
## [126,] 8.116456 -5.887854  0.83738724 -0.274056699
## [127,] 6.760873 -5.147248  0.58600501  0.114152134
## [128,] 6.793497 -5.210284  0.40059625 -0.029325105
## [129,] 7.625974 -5.117223  0.47767269  0.019475610
## [130,] 7.890368 -5.791592  1.08138291 -0.264875766
## [131,] 8.344038 -5.702222  1.13075079 -0.055688175
## [132,] 8.733039 -6.701118  0.74640092 -0.286253636
## [133,] 7.661803 -5.109675  0.42308955  0.094841353
## [134,] 6.946526 -5.183539  0.78508660 -0.224348071
## [135,] 7.283660 -4.827051  0.80472791 -0.538786125
## [136,] 8.578865 -6.015038  0.96744501  0.276476332
## [137,] 7.646608 -5.467017 -0.10302622  0.022190256
## [138,] 7.407463 -5.376253  0.46967248 -0.254554650
## [139,] 6.671691 -5.161962  0.35001688 -0.012889926
## [140,] 7.609976 -5.699240  0.60456158  0.177270073
## [141,] 7.816520 -5.510604  0.30915897  0.244302065
## [142,] 7.424633 -5.736156  0.51826612  0.471953254
## [143,] 6.917601 -4.752036  0.33553019 -0.048656387
## [144,] 8.065379 -5.604815  0.33928319  0.024561035
## [145,] 7.921111 -5.631751  0.12737005  0.207739288
## [146,] 7.446475 -5.514485  0.45402763  0.392844227
## [147,] 7.029532 -4.951636  0.75375089  0.221015729
## [148,] 7.266711 -5.405811  0.50137108  0.103649561
## [149,] 7.403307 -5.443581 -0.09139885  0.011243592
## [150,] 6.892554 -5.044292  0.26894307 -0.188390341
```

Order of the columns:

- Sepal.Length

- Sepal.Width
- Petal.Length
- Petal.Width

Looking at the loadings, PC1 is mostly influenced by Petal Length and a bit less of Petal Width, while PC2 is negatively associated by Sepal Length and Width

b. correlation matrix

```
cor_iris <- cor(iris_cut) # correlation matrix
eigen_decomposition_cor <- eigen(cor_iris)
loadings_cor <- eigen_decomposition_cor$vectors
loadings_cor
```

```
##            [,1]        [,2]       [,3]       [,4]
## [1,]  0.5210659 -0.37741762  0.7195664  0.2612863
## [2,] -0.2693474 -0.92329566 -0.2443818 -0.1235096
## [3,]  0.5804131 -0.02449161 -0.1421264 -0.8014492
## [4,]  0.5648565 -0.06694199 -0.6342727  0.5235971
```

```
scores_cor <- iris_cut %*% loadings_cor
scores_cor
```

```
##            [,1]      [,2]     [,3]         [,4]
##   [1,] 2.640270 -5.204041 2.488621 -0.117033159
##   [2,] 2.670730 -4.666910 2.466898 -0.107535605
##   [3,] 2.454606 -4.773636 2.288321 -0.104349860
##   [4,] 2.545517 -4.648463 2.212378 -0.278417376
##   [5,] 2.561228 -5.258629 2.392226 -0.155512749
##   [6,] 2.975946 -5.707321 2.437245 -0.223766470
##   [7,] 2.463157 -4.929697 2.089848 -0.182965623
##   [8,] 2.673139 -5.076419 2.426890 -0.210955749
##   [9,] 2.437132 -4.385872 2.131553 -0.225827783
##  [10,] 2.645351 -4.754994 2.491675 -0.252391205
##  [11,] 2.800761 -5.504375 2.641402 -0.143494123
##  [12,] 2.626967 -5.003385 2.268764 -0.343357930
##  [13,] 2.562138 -4.622474 2.458369 -0.186023946
##  [14,] 2.127481 -4.426418 2.141224 -0.076232312
##  [15,] 2.754260 -5.924983 2.898552  0.164402277
##  [16,] 2.881509 -6.277296 2.559350 -0.046845546
##  [17,] 2.743781 -5.697524 2.494096  0.096813229
##  [18,] 2.696755 -5.210735 2.425193 -0.064673445
##  [19,] 3.102715 -5.721522 2.740981 -0.185389337
##  [20,] 2.673992 -5.490173 2.337666 -0.181871256
##  [21,] 2.997648 -5.232284 2.686291 -0.266731087
##  [22,] 2.757413 -5.404538 2.298677 -0.117160580
##  [23,] 2.120637 -5.097865 2.161250  0.060552438
##  [24,] 3.037720 -5.046812 2.304577 -0.175686868
##  [25,] 2.801091 -5.010732 2.226126 -0.583792704
##  [26,] 2.838920 -4.709550 2.510430 -0.241696826
##  [27,] 2.844152 -5.092257 2.285822 -0.186381247
```

```
##  [28,] 2.750418 -5.244232 2.546365 -0.171049455
##  [29,] 2.719311 -5.149453 2.585016 -0.078553569
##  [30,] 2.628730 -4.780984 2.245683 -0.344784634
##  [31,] 2.707772 -4.726396 2.342078 -0.306305044
##  [32,] 2.994537 -5.240775 2.587862 -0.001721811
##  [33,] 2.532324 -5.791515 2.463163 -0.297514941
##  [34,] 2.660153 -6.001315 2.605380 -0.098975380
##  [35,] 2.701837 -4.761689 2.428248 -0.200031492
##  [36,] 2.552885 -4.884413 2.518404  0.054180948
##  [37,] 2.790655 -5.352559 2.790660  0.067626278
##  [38,] 2.452636 -5.214193 2.383697 -0.234001090
##  [39,] 2.352156 -4.475752 2.121328 -0.158033820
##  [40,] 2.725246 -5.114161 2.498846 -0.184827121
##  [41,] 2.586608 -5.170545 2.367449 -0.010657149
##  [42,] 2.649291 -3.873881 2.300924  0.006911255
##  [43,] 2.298287 -4.660411 2.072451 -0.182735744
##  [44,] 2.930188 -5.197975 2.134530 -0.094012782
##  [45,] 2.962643 -5.506664 2.217388 -0.450091241
##  [46,] 2.675109 -4.635862 2.331514 -0.081304519
##  [47,] 2.675548 -5.485928 2.386881 -0.314375894
##  [48,] 2.460541 -4.738344 2.202152 -0.210623413
##  [49,] 2.748655 -5.466633 2.569445 -0.169622751
##  [50,] 2.642033 -4.981640 2.465540 -0.118459863
##  [51,] 6.304290 -5.805299 2.698967 -1.600002292
##  [52,] 5.932054 -5.580644 2.232225 -1.544124498
##  [53,] 6.451687 -5.686820 2.559596 -1.721710094
##  [54,] 5.302329 -4.384368 2.002477 -1.372118296
##  [55,] 6.149941 -5.251517 2.387722 -1.548736946
##  [56,] 5.562075 -4.933745 1.953136 -1.782340473
##  [57,] 6.025581 -5.646824 2.043978 -1.690534223
##  [58,] 4.387009 -4.213020 1.836069 -1.137305694
##  [59,] 6.062141 -5.368200 2.562095 -1.639678707
##  [60,] 5.036715 -4.644706 1.639640 -1.367403389
##  [61,] 4.662937 -3.886342 1.977353 -1.222063067
##  [62,] 5.551266 -5.199929 1.963956 -1.409630940
##  [63,] 5.420340 -4.460665 2.576980 -1.386203334
##  [64,] 5.916135 -5.188634 2.124672 -1.798107058
##  [65,] 4.960662 -4.966290 1.984655 -1.099515741
##  [66,] 6.000781 -5.592396 2.550173 -1.425602441
##  [67,] 5.569071 -5.094051 1.705448 -1.728451598
##  [68,] 5.239494 -4.849278 2.296663 -1.580360325
##  [69,] 6.097188 -4.581865 2.332694 -1.472872134
##  [70,] 5.129554 -4.590931 2.166624 -1.395266094
##  [71,] 6.015102 -5.419365 1.639522 -1.758123271
##  [72,] 5.480295 -5.072466 2.312026 -1.277101338
##  [73,] 6.300656 -4.906392 2.274485 -1.804376090
##  [74,] 5.830099 -5.082916 2.275965 -1.890475523
##  [75,] 5.783804 -5.285369 2.460820 -1.451501189
##  [76,] 5.975609 -5.462325 2.502655 -1.439380107
##  [77,] 6.365857 -5.362946 2.638594 -1.683000625
##  [78,] 6.545421 -5.534844 2.299054 -1.737041886
##  [79,] 5.804432 -5.152688 2.017713 -1.611586124
##  [80,] 4.866075 -4.704512 2.334421 -1.113268443
##  [81,] 5.046341 -4.458411 2.133318 -1.328898835
```

```
##  [82,] 4.931814 -4.449267 2.210958 -1.301113624
##  [83,] 5.236383 -4.857768 2.198234 -1.315351048
##  [84,] 6.263035 -4.989418 1.917886 -2.015394034
##  [85,] 5.464857 -5.018567 1.561535 -1.780708854
##  [86,] 5.726244 -5.621030 1.832095 -1.620981220
##  [87,] 6.231391 -5.606438 2.444108 -1.613677501
##  [88,] 5.951347 -4.696099 2.521279 -1.483668970
##  [89,] 5.223934 -5.070866 1.889154 -1.512591326
##  [90,] 5.248460 -4.569027 1.953600 -1.396820220
##  [91,] 5.397205 -4.664459 1.935739 -1.782110594
##  [92,] 5.831159 -5.278515 2.114446 -1.730313096
##  [93,] 5.321359 -4.767888 2.208459 -1.383145011
##  [94,] 4.466050 -4.158432 1.932464 -1.098826104
##  [95,] 5.362780 -4.796326 1.948255 -1.555683365
##  [96,] 5.277596 -5.104363 2.010325 -1.618967336
##  [97,] 5.361017 -5.018727 1.971336 -1.554256661
##  [98,] 5.679591 -5.209885 2.316906 -1.503758445
##  [99,] 4.346649 -4.380180 1.934755 -0.804604912
## [100,] 5.329910 -4.923948 2.009987 -1.461760774
## [101,] 7.288489 -5.738911 1.288368 -2.261180823
## [102,] 6.328278 -4.934017 1.583691 -1.910572150
## [103,] 7.502162 -5.734631 2.205257 -2.144392842
## [104,] 6.768663 -5.312937 1.886962 -2.257715270
## [105,] 7.187966 -5.512425 1.724303 -2.168659972
## [106,] 8.168984 -5.940484 2.465552 -2.574764174
## [107,] 5.451969 -4.381599 1.197088 -1.744877757
## [108,] 7.696018 -5.707499 2.507040 -2.557443463
## [109,] 7.200911 -5.099484 2.244116 -2.264086760
## [110,] 7.734685 -6.358025 1.848451 -2.143220981
## [111,] 6.614836 -5.666552 1.901770 -1.737066850
## [112,] 6.757001 -5.165366 1.987006 -1.914090231
## [113,] 7.113677 -5.611609 2.046238 -1.902199027
## [114,] 6.328486 -4.715862 1.511396 -1.779494216
## [115,] 6.583772 -5.059818 1.242117 -1.661124544
## [116,] 6.848269 -5.653791 1.611106 -1.766406187
## [117,] 6.787900 -5.478301 2.020650 -2.137664052
## [118,] 8.120139 -6.726005 2.264363 -2.675228453
## [119,] 8.615925 -5.629643 2.465769 -2.634947046
## [120,] 6.283181 -4.518627 2.117717 -1.925854013
## [121,] 7.340968 -5.852296 1.914039 -1.956342746
## [122,] 6.137534 -4.952659 1.380338 -1.762530805
## [123,] 8.276516 -5.789321 2.635600 -2.656438261
## [124,] 6.416243 -5.111134 2.035327 -1.671998874
## [125,] 7.096848 -5.855754 1.872542 -2.125670390
## [126,] 7.388983 -5.939398 2.404407 -2.380190203
## [127,] 6.279160 -5.163272 1.953145 -1.630333539
## [128,] 6.231226 -5.312639 1.818099 -1.761309016
## [129,] 7.017161 -5.278432 1.793075 -2.062156540
## [130,] 7.213799 -5.736452 2.608563 -2.299917857
## [131,] 7.715462 -5.654707 2.568433 -2.306314310
## [132,] 7.937257 -6.780753 2.577769 -2.487255850
## [133,] 7.073647 -5.285126 1.729648 -2.009796827
## [134,] 6.335934 -5.188279 2.172745 -2.001718826
## [135,] 6.519311 -4.933688 2.070073 -2.482358494
```

```
## [136,] 8.043855 -5.979368 2.481717 -2.043191496
## [137,] 6.972903 -5.814750 1.384208 -2.005311799
## [138,] 6.708859 -5.532889 1.924255 -2.176143642
## [139,] 6.121078 -5.272448 1.760355 -1.707292719
## [140,] 7.080807 -5.739231 2.107969 -1.808276437
## [141,] 7.262134 -5.688728 1.745349 -1.863744402
## [142,] 7.019655 -5.745272 2.023753 -1.463122236
## [143,] 6.328278 -4.934017 1.583691 -1.910572150
## [144,] 7.404944 -5.819453 1.813657 -2.142761223
## [145,] 7.322791 -5.882531 1.618833 -1.916231537
## [146,] 7.000417 -5.579908 1.890065 -1.583173455
## [147,] 6.584640 -4.935618 2.006564 -1.675082161
## [148,] 6.726747 -5.484342 1.936433 -1.792509851
## [149,] 6.748228 -5.765416 1.404104 -1.923510292
## [150,] 6.243095 -5.242054 1.645761 -1.973856121
```

While using correlation matrix for PCA, components have equal weight and emportance, therefore, loadings for all principal components are more equally distributed than ones calculated using covariance matrix.

c. millimeters

```
iris_cut_mil <- iris_cut
iris_cut_mil[, "Petal.Length"] <- iris_cut[, "Petal.Length"] * 10
head(iris_cut_mil)
```

```
##      Sepal.Length Sepal.Width Petal.Length Petal.Width
## [1,]          5.1         3.5           14         0.2
## [2,]          4.9         3.0           14         0.2
## [3,]          4.7         3.2           13         0.2
## [4,]          4.6         3.1           15         0.2
## [5,]          5.0         3.6           14         0.2
## [6,]          5.4         3.9           17         0.4
```

```
pca_mil <- prcomp(iris_cut_mil, scale. = FALSE)
summary(pca_mil)
```

```
## Importance of components:
##                             PC1     PC2     PC3    PC4
## Standard deviation     17.6840 0.50221 0.28081 0.1754
## Proportion of Variance  0.9988 0.00081 0.00025 0.0001
## Cumulative Proportion   0.9988 0.99965 0.99990 1.0000
```

The proportion of variance has highly changed to having more weight on the first PC (99.88%) and leaving just small proportion of variance for the last three components

```
cov_iris_mil <- cov(iris_cut_mil) # covariance matrix
eigen_decomposition_mil <- eigen(cov_iris_mil)
loadings_mil <- eigen_decomposition_mil$vectors
loadings_mil
```

```
##             [,1]         [,2]        [,3]        [,4]
```

```
## [1,]  0.04083715  0.723859293  0.60196083  0.33466881
## [2,] -0.01055112  0.689576596 -0.63020980 -0.35666286
## [3,]  0.99824759 -0.022442542 -0.01090193 -0.05365844
## [4,]  0.04150602  0.002859015 -0.49026515  0.87057979
```

Petal.Length accounts for the highest influence on the first principal component - which is logical, as its measure is now different and model consideres it as 10 times increase, not as another measure

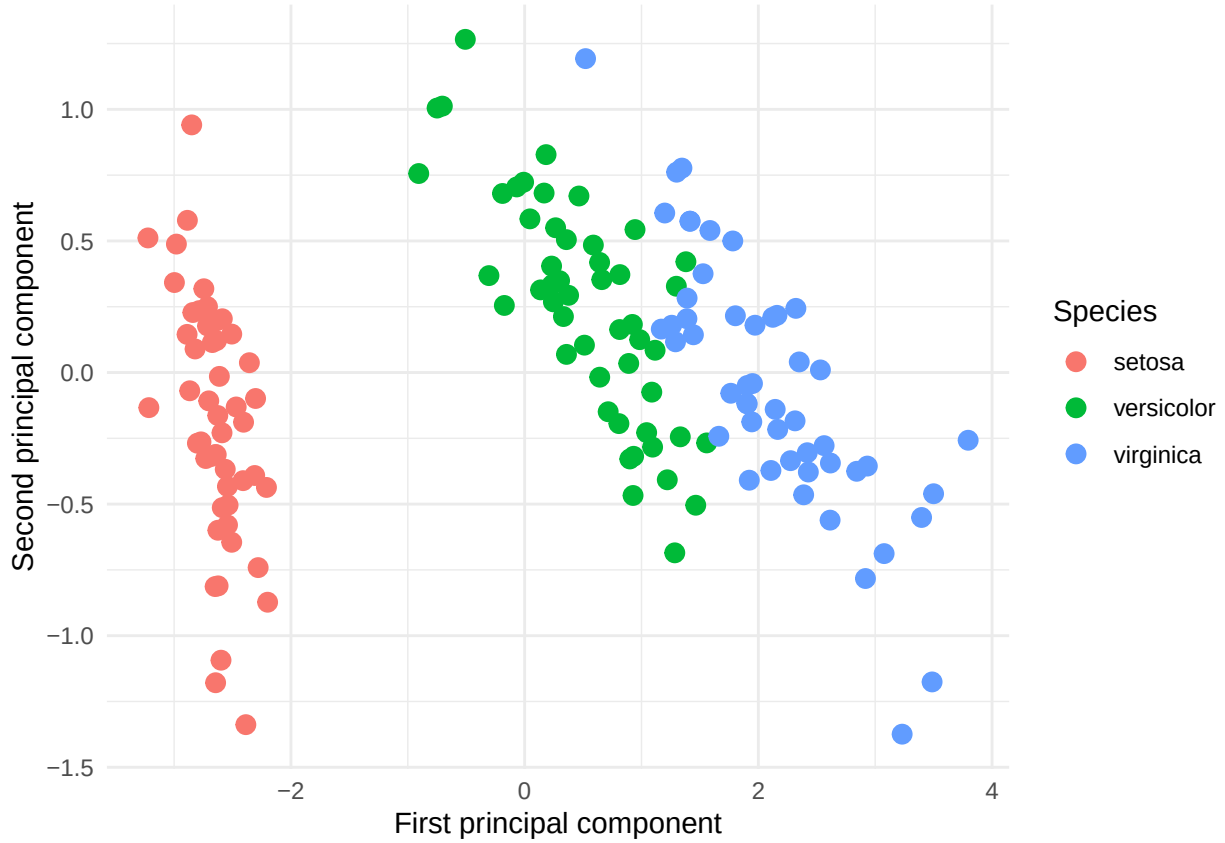
```
cor_iris_mil <- cor(iris_cut_mil) # correlation matrix
eigen_decomposition_cor_mil <- eigen(cor_iris_mil)
loadings_cor_mil <- eigen_decomposition_cor_mil$vectors
loadings_cor_mil
```

```
##            [,1]        [,2]       [,3]       [,4]
## [1,]  0.5210659 -0.37741762  0.7195664  0.2612863
## [2,] -0.2693474 -0.92329566 -0.2443818 -0.1235096
## [3,]  0.5804131 -0.02449161 -0.1421264 -0.8014492
## [4,]  0.5648565 -0.06694199 -0.6342727  0.5235971
```

However, calculations based on correlation matrix are not influenced by this change at all, which is also logical, considering that prior correlation hasn't changed by changing the measumement This leads us to the conclusion that having equal weights (correlation matrix) can be useful while modeling different types of variables, while using covariation matrix helps to understand the actual weight of each variable

d. PC plot

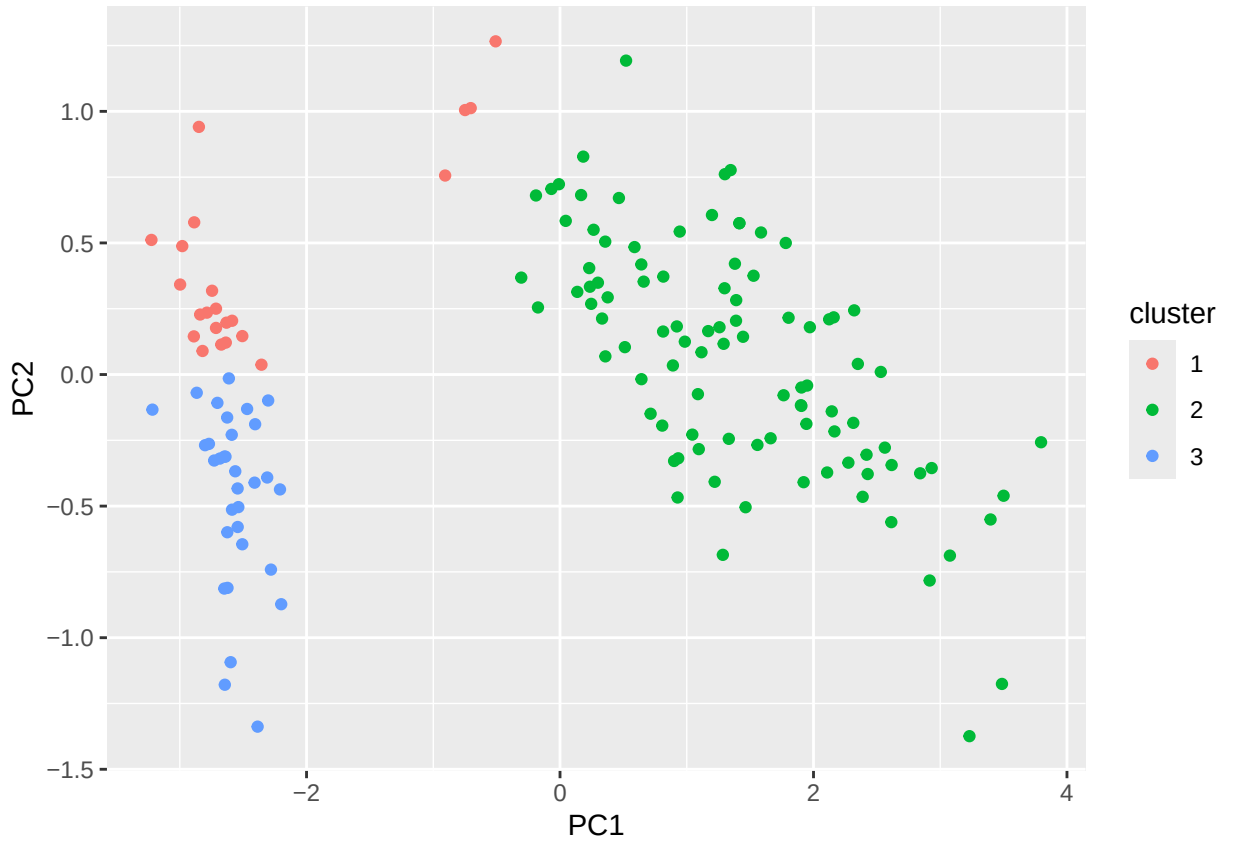
```
pca_components <- data.frame(pca$x[, 1:2])
pca_components$Species <- iris$Species
ggplot(pca_components, aes(x = PC1, y = PC2, color = Species)) +
  geom_point(size = 3) +
  labs(
       x = "First principal component",
       y = "Second principal component"
       ) +
  theme_minimal()
```



According to the plot, all three species are quite well separated I think, this two-dimensional plot represents the relative positions quite well, as first two components account for the most variance of the data.

e. K-means

```
set.seed(47)
k = 3
kmeans_iris <- kmeans(pca_components[, 1:2], centers = k)
# take cluster
pca_components$cluster <- as.factor(kmeans_iris$cluster)
ggplot(pca_components, aes(x = PC1, y = PC2, color = cluster)) +
  geom_point()
```




```
table(pca_components$cluster, pca_components$Species)
```

```
##
##     setosa versicolor virginica
##   1     18          4         0
##   2      0         46        50
##   3     32          0         0
```

cluster numbers:

- 1 - setosa
- 2 - versicolor
- 3 - virginica

Looks like the model predicts setosa and versicolor species good, but very bad predicts virginica, messing it up with setosa

- classification error (reduced)

```
species <- as.numeric(factor(iris$Species))
CE_cut <- sum(pca_components$cluster != species) / length(species)
CE_cut
```

```
## [1] 0.5733333
```

- classification error (full)

```
set.seed(47)
kmeans_iris_full <- kmeans(iris_cut, centers = k)

CE_full <- sum(kmeans_iris_full$cluster != species) / length(species)
CE_full
```

```
## [1] 0.58
```

The classification error is even larger while using the full dataset And still very big, that's probably mainly because of incorrect classification of virginica

Exercise 10

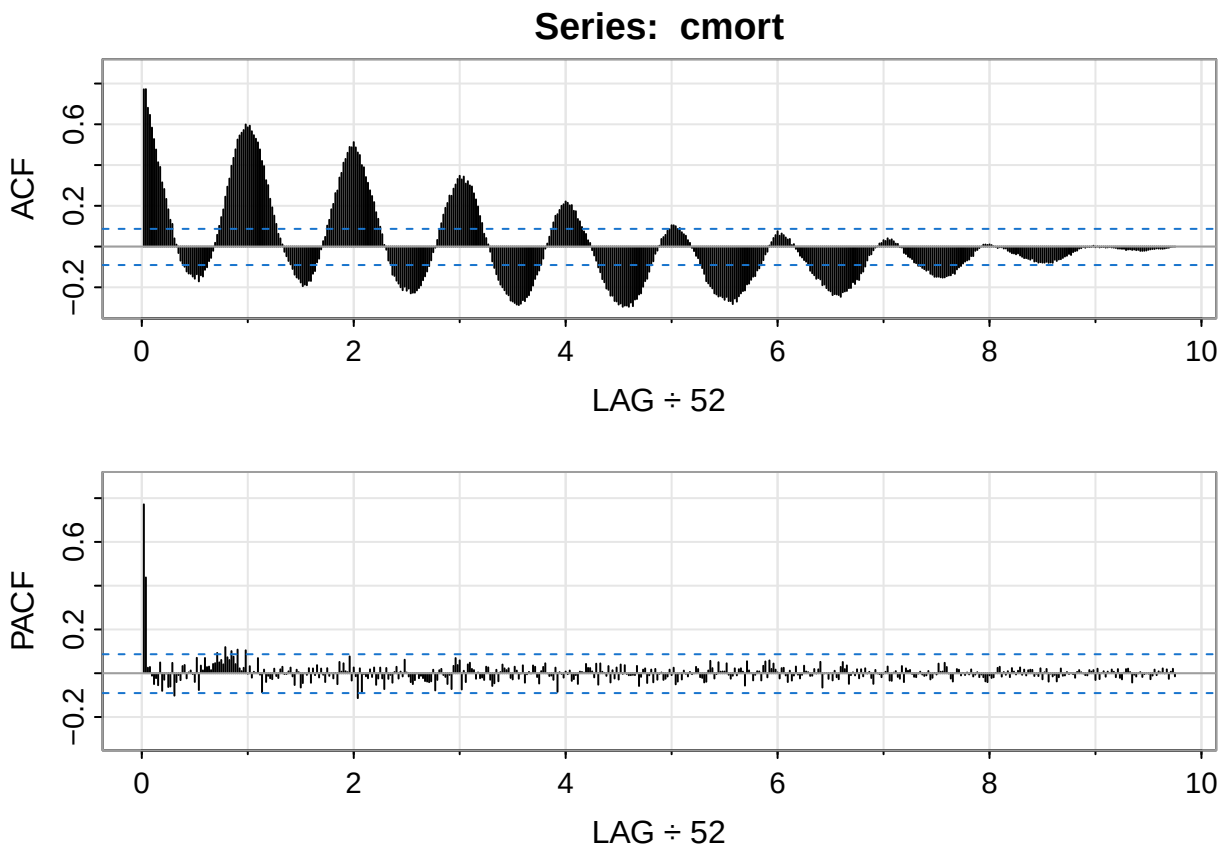
a. AR(2) fit

```
data(cmort)
cmort <- cmort
head(cmort)
```

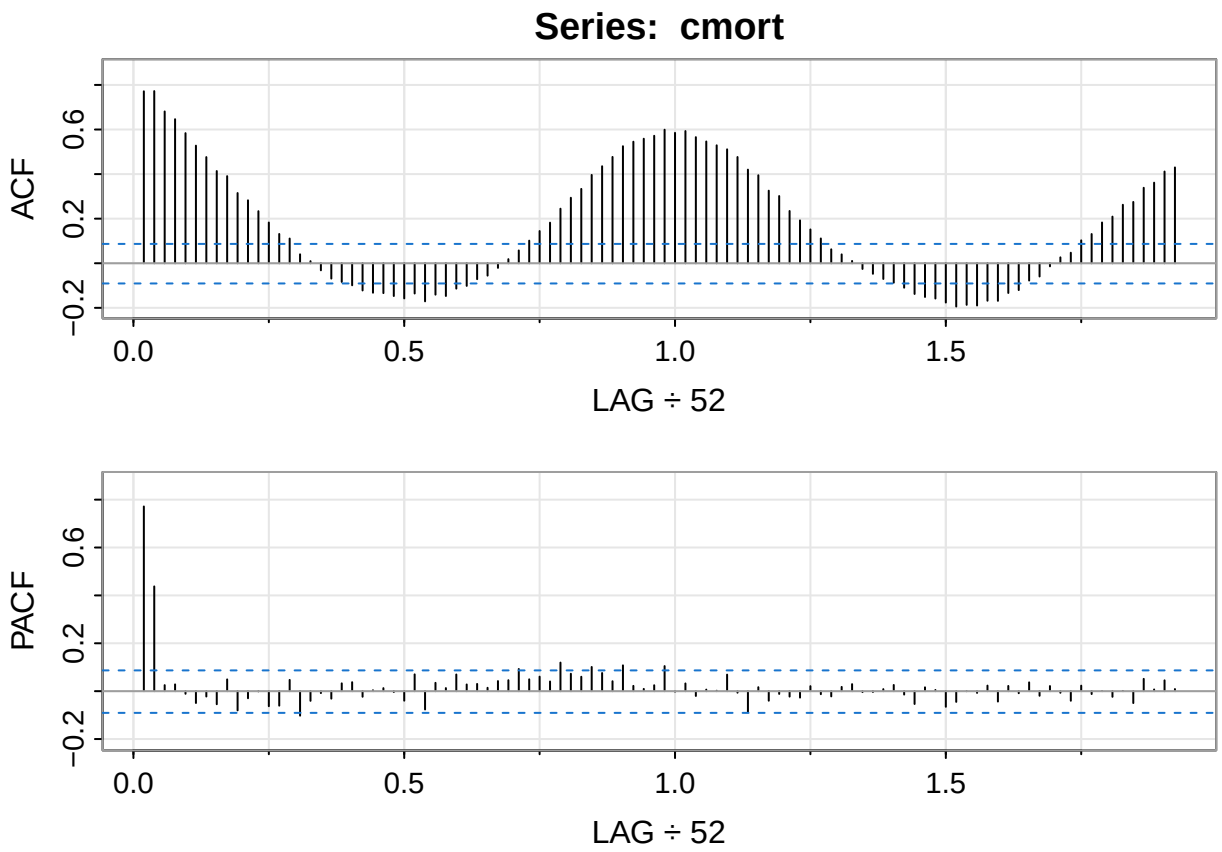
```
## [1]  97.85 104.64  94.36  98.05  95.85  95.98
```

Let's look at the data:

```
n <- length(cmort)
invisible(acf2(cmort, max.lag = n - 1))
```




```
invisible(acf2(cmort, max.lag = 100))
```



There is definitely some time trend in the data Drop after second lag in PACF indicates that AR(2) may be a good model to fit

a. AR model

```
cmort_lags <- data.frame(
  x = cmort[3:n],
  x_1 = cmort[2:(n - 1)],
  x_2 = cmort[1:(n - 2)]
)
cmort_ar2 <- lm(x ~ x_1 + x_2, data = cmort_lags)
```

b. forecast AR(2)

```
n_weeks <- 4
forecast_ar <- numeric(n_weeks)
se_ar <- numeric(n_weeks)

x_1 <- cmort_lags$x[nrow(cmort_lags)]
x_2 <- cmort_lags$x_1[nrow(cmort_lags)]

# predictions
for (i in 1:n_weeks) {
  forecast_ar[i] <- coef(cmort_ar2)["(Intercept)"] +
                    coef(cmort_ar2)["x_1"] * x_1 +
```

```
                    coef(cmort_ar2)["x_2"] * x_2
  x_2 <- x_1
  x_1 <- forecast_ar[i]

  se_ar[i] <- sqrt(sum(residuals(cmort_ar2)^2) / (nrow(cmort_lags) - 2))
}

lower_bound_ar <- forecast_ar - 1.96 * se_ar
upper_bound_ar <- forecast_ar + 1.96 * se_ar

results_ar2 <- data.frame(
  ar.pred = forecast_ar,
  ar.se = se_ar,
  lower_95_CI = lower_bound_ar,
  upper_95_CI = upper_bound_ar
)

print(results_ar2)
```

```
##    ar.pred    ar.se lower_95_CI upper_95_CI
## 1 87.59986 5.696116    76.43548    98.76425
## 2 86.76349 5.696116    75.59910    97.92788
## 3 87.33714 5.696116    76.17275    98.50153
## 4 87.21350 5.696116    76.04911    98.37789
```

Based on the predictions, the mortality rate in the following four weeks will be around 75 - 98.

c. Yule-Walker method

```
cmort_yw <- ar.yw(cmort, order.max = 2)
cmort_yw$ar
```

```
## [1] 0.4339481 0.4375768
```

```
sqrt(diag(cmort_yw$asy.var.coef))
```

```
## [1] 0.04001303 0.04001303
```

```
summary(cmort_ar2)
```

```
##
## Call:
## lm(formula = x ~ x_1 + x_2, data = cmort_lags)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -17.8192  -4.0339  -0.2112   3.4219  22.1840
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
```

```
## (Intercept) 11.45061    2.40080   4.769 2.42e-06 ***
## x_1          0.42859    0.03991  10.738  < 2e-16 ***
## x_2          0.44179    0.03988  11.078  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 5.702 on 503 degrees of freedom
## Multiple R-squared:  0.6752, Adjusted R-squared:  0.6739
## F-statistic: 522.8 on 2 and 503 DF,  p-value: < 2.2e-16
```

Yule-Walker method suggests similar coefficients for both lags and a bit larger standard error than AR(2) model This is because of the different approach: OLS assumes having independent error, while Yule-Walker accounts for autocorrelation (which is typical for time series data) All in all, results do not differ much, so we can rely on both

```
forecast_yw <- predict(cmort_yw, n.ahead = n_weeks, se.fit = TRUE)

lower_bound_yw <- forecast_yw$pred - 1.96 * forecast_yw$se
upper_bound_yw <- forecast_yw$pred + 1.96 * forecast_yw$se

results_total <- data.frame(
  forecast_ar = forecast_ar,
  lower_95_CI = lower_bound_ar,
  upper_95_CI = upper_bound_ar,
  forecast_yw = forecast_yw,
  lower_95_CI = lower_bound_yw,
  upper_95_CI = upper_bound_yw
)

print(results_total)
```

```
##   forecast_ar lower_95_CI upper_95_CI forecast_yw.pred forecast_yw.se
## 1    87.59986    76.43548    98.76425         87.62631       5.730669
## 2    86.76349    75.59910    97.92788         86.82931       6.246984
## 3    87.33714    76.17275    98.50153         87.41825       7.203445
## 4    87.21350    76.04911    98.37789         87.32507       7.673573
##   lower_95_CI.1 upper_95_CI.1
## 1      76.39420      98.85842
## 2      74.58522      99.07340
## 3      73.29950     101.53701
## 4      72.28487     102.36528
```

The confidence intervals using Yule-Walker get wider with for third and fourth weeks because the uncertainty increases In total, the predictions are quite similar among both models

f. ARMA model

For the comparison I will recalculate AR(2) model using arima function

```
cmort_ar2 <- arima(cmort, order = c(2, 0, 0))
cmort_arma22 <- arima(cmort, order = c(2, 0, 2))
AIC(cmort_ar2)
```

```
## [1] 3217.429
```

```
AIC(cmort_arma22)
```

```
## [1] 3220.77
```

```
BIC(cmort_ar2)
```

```
## [1] 3234.351
```

```
BIC(cmort_arma22)
```

```
## [1] 3246.153
```

AIC and BIC information criterions suggest that AR model performs better than ARMA, so there is no need in adding additional complexity for the model