Exercises

Natali Tckvitishvili

2024-08-11

Applied Statistics in R

Natali Tckvitishvili

Load libraries

```
#install.packages("extraDistr")
#install.packages('tinytex')
#install.packages('ggpubr')
#install.packages('tidyverse')

library(tidyverse)
library(ggplot2)
library(stats)
library(ggpubr)
library(extraDistr)
library(tinytex)
```

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)
```

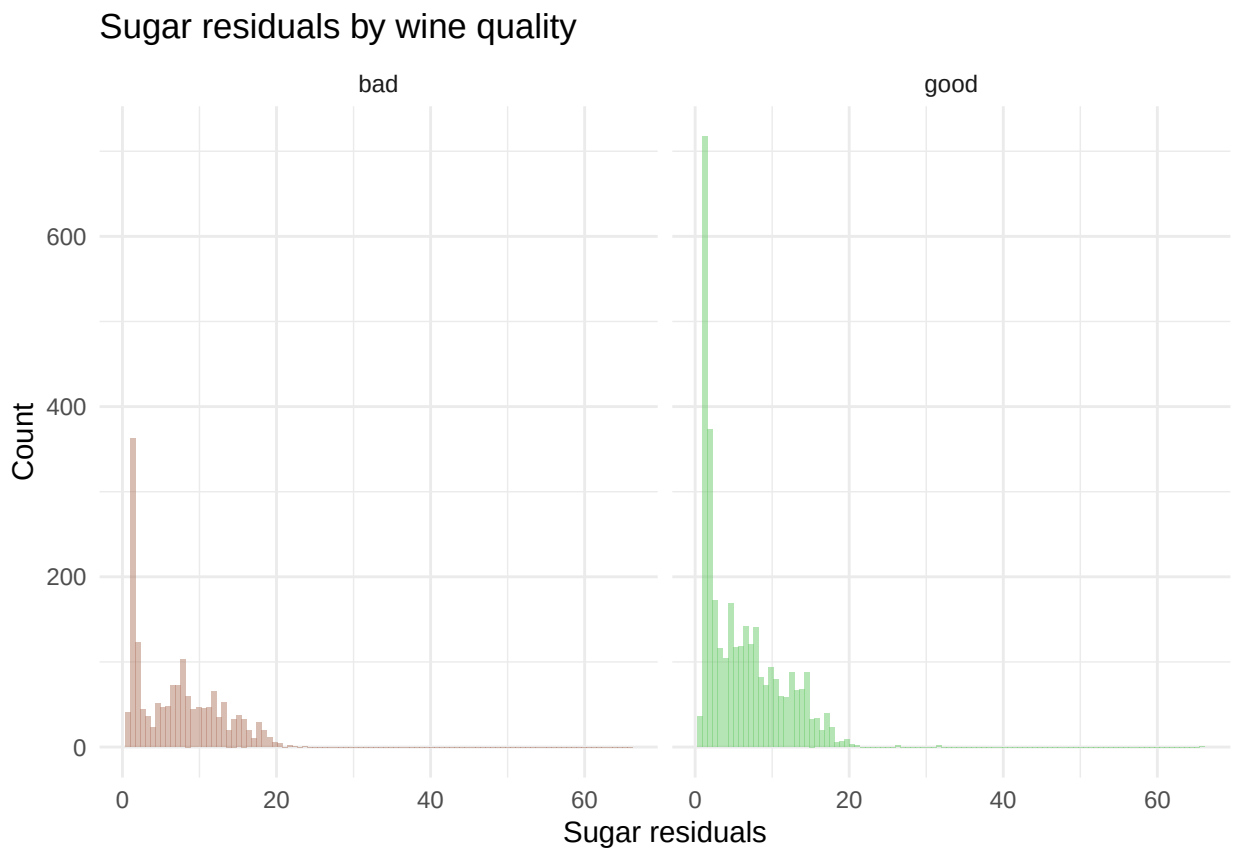
b. residual.sugar analysis

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

```r
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

```r
summary <- wine %>%
  group_by(good) %>%
  summarise(
    n = n(),
```

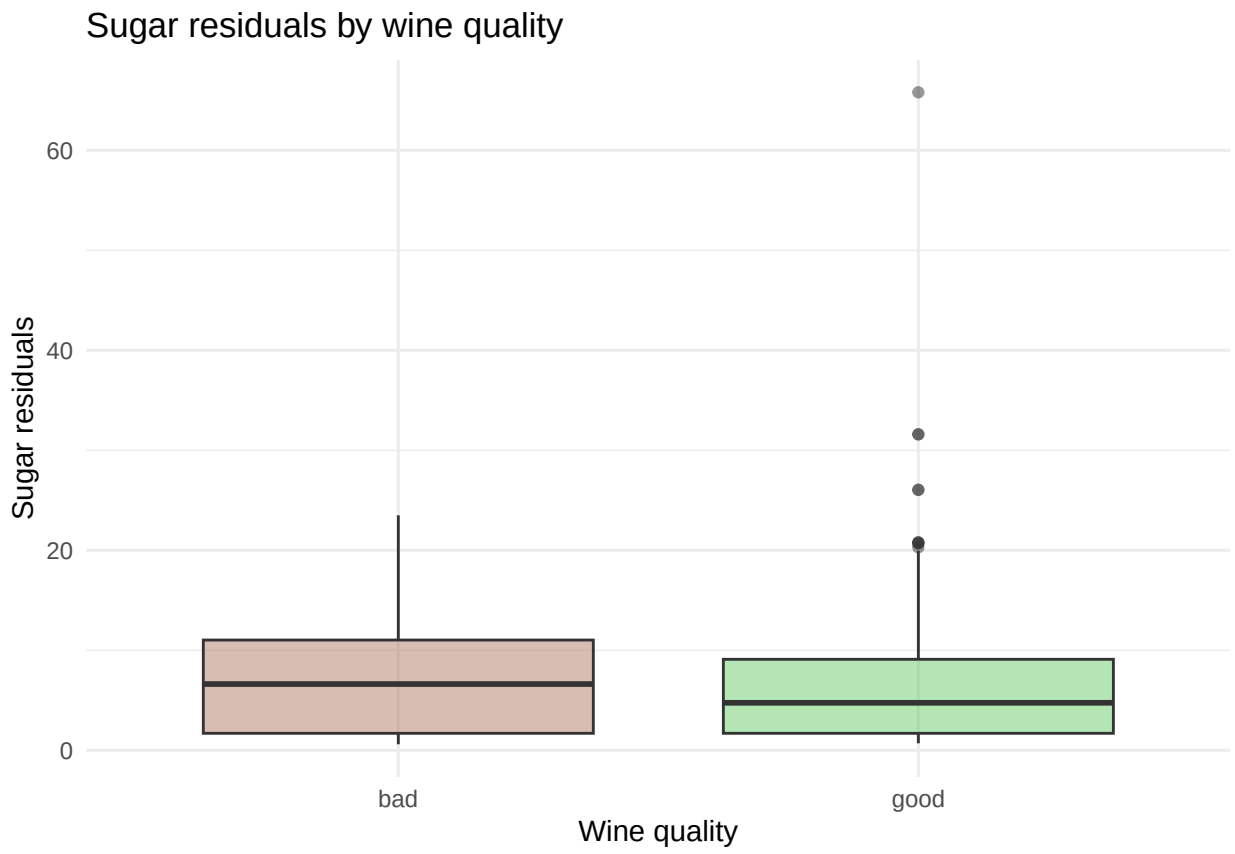
```
    mean = mean(analysed_variable),
    median = median(analysed_variable),
    sd = sd(analysed_variable),
    iqr = IQR(analysed_variable),
    max = max(analysed_variable),
    min = min(analysed_variable)
  )
data.frame(summary)
```

```
##   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"
    )
```

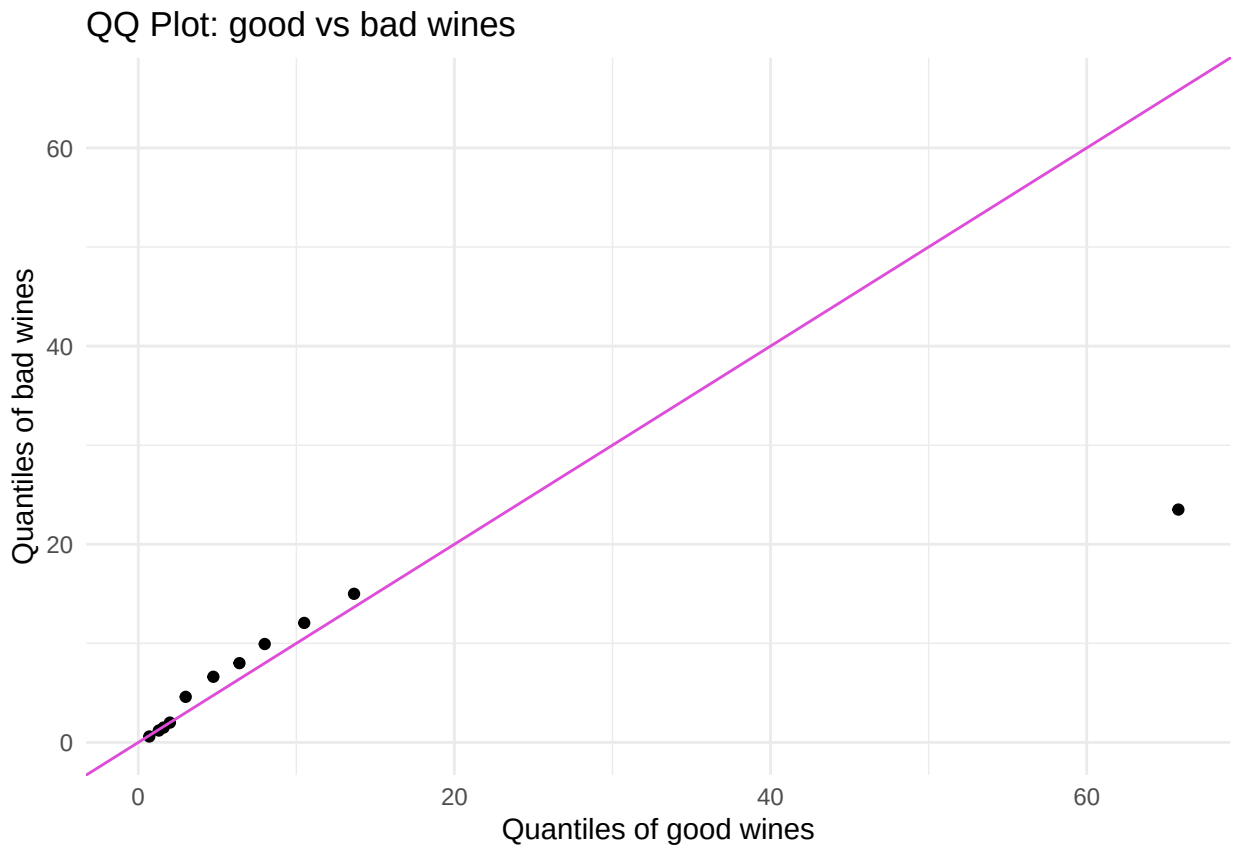


As stated above, the good wine with 65.8 sugar residuals must be an outlier, accoring to the boxplots. There are two more outliers, and all of them impact the mean. Assuming that better wines on average have less sugar, these observations might be a quality estimation error / human factor or there are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

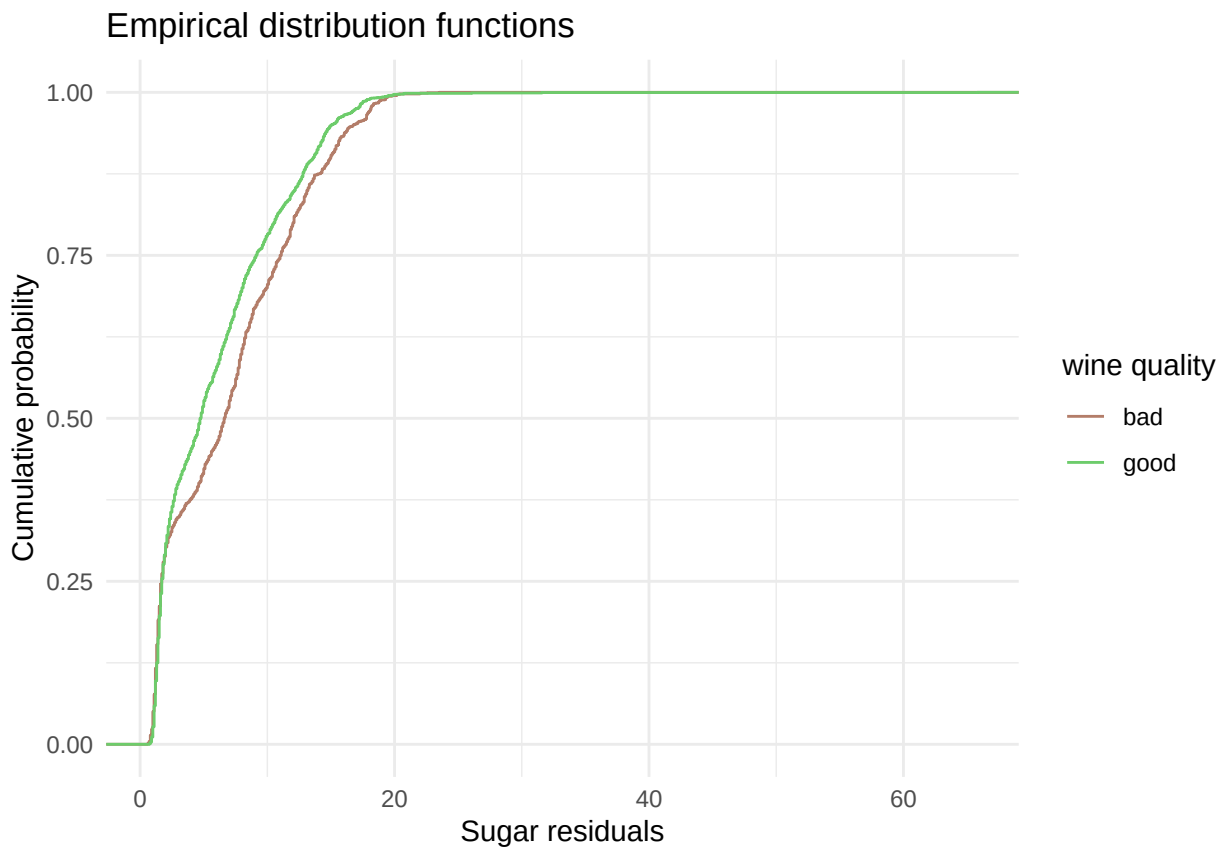
```
        x = "Quantiles of good wines",
        y = "Quantiles of bad wines")
```



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).

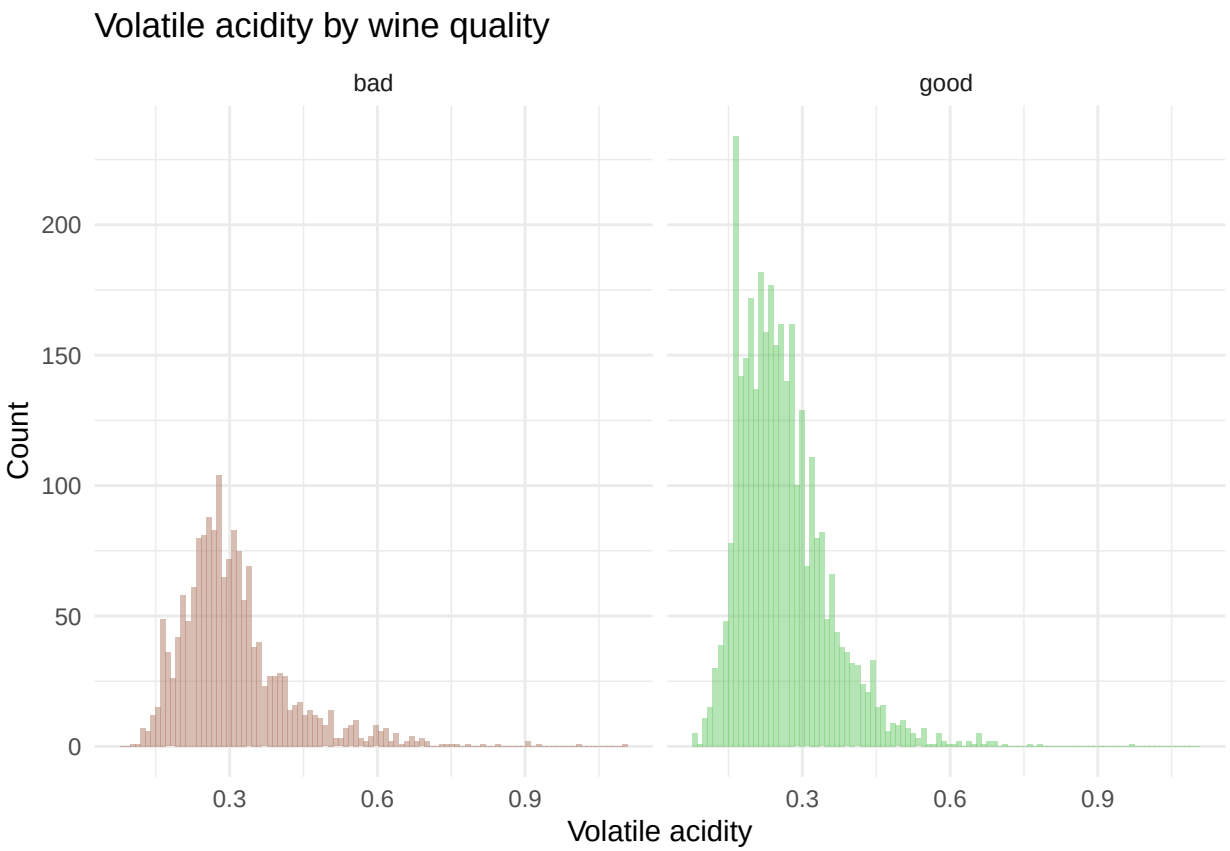
All of those graphs and summary statistics show the same: good wines in general have lower sugar residuals than bad ones.

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

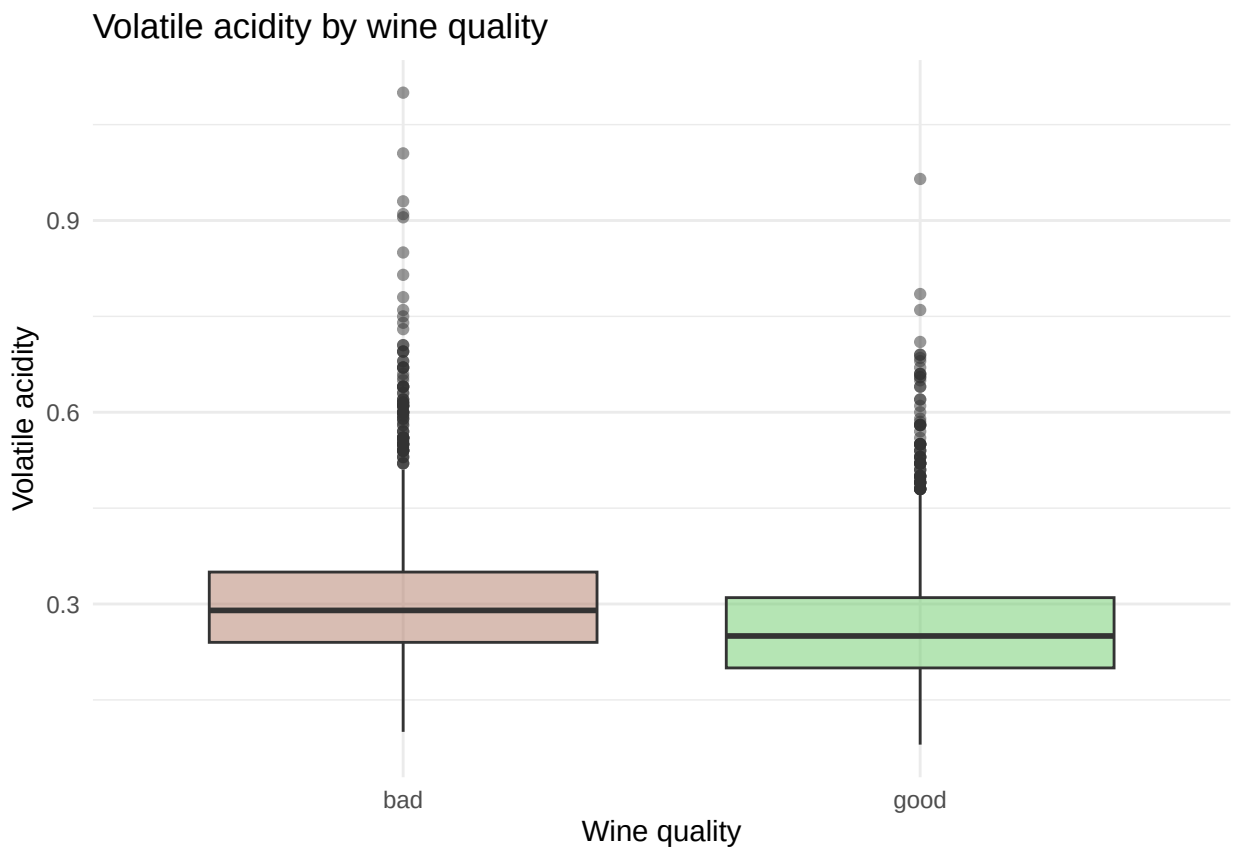
```
summary <- wine %>%
  group_by(good) %>%
  summarise(
    n = n(),
    mean = mean(analysed_variable),
    median = median(analysed_variable),
    sd = sd(analysed_variable),
    iqr = IQR(analysed_variable),
    max = max(analysed_variable),
    min = min(analysed_variable)
  )
data.frame(summary)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

```
##   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"
    )
```

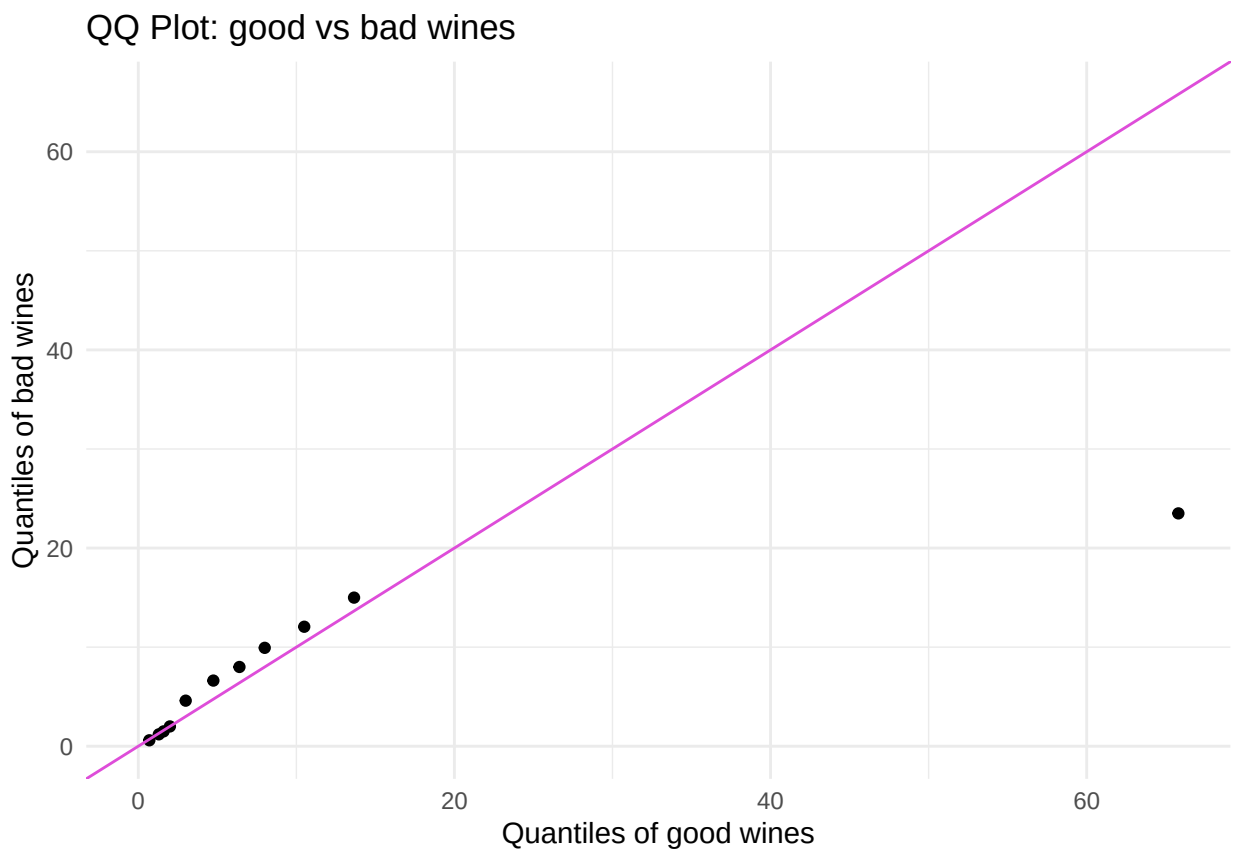


Looks like we have much more outliers here than in sugar residuals. In general, bad wines seem to have more acids (boxplot is located higher).

- 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)) +
```

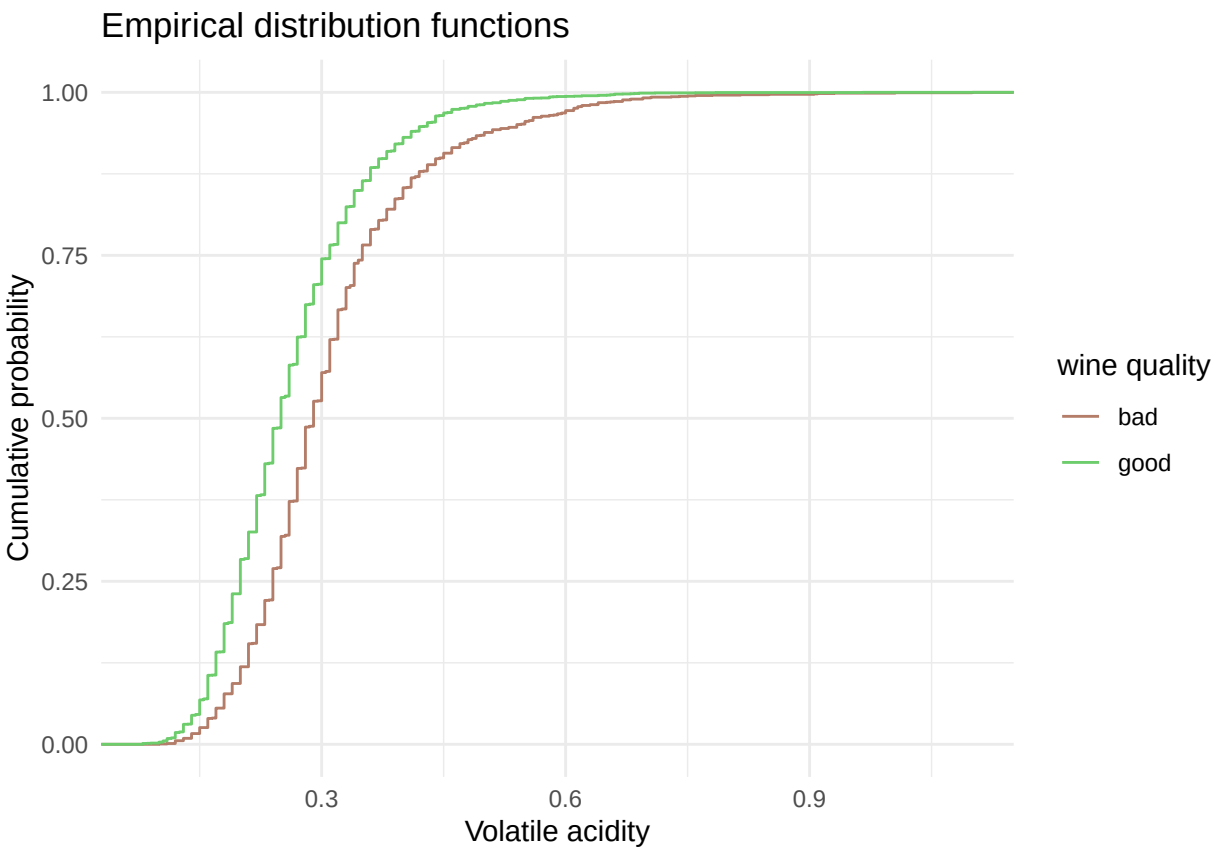
```
    geom_point() +
    geom_abline(slope = 1, intercept = 0, color = "#de4dd9") +
    xlim(0, max(good_quantiles, bad_quantiles)) +
    ylim(0, max(good_quantiles, bad_quantiles)) +
    theme_minimal() +
    labs(title = "QQ Plot: good vs bad wines",
         x = "Quantiles of good wines",
     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

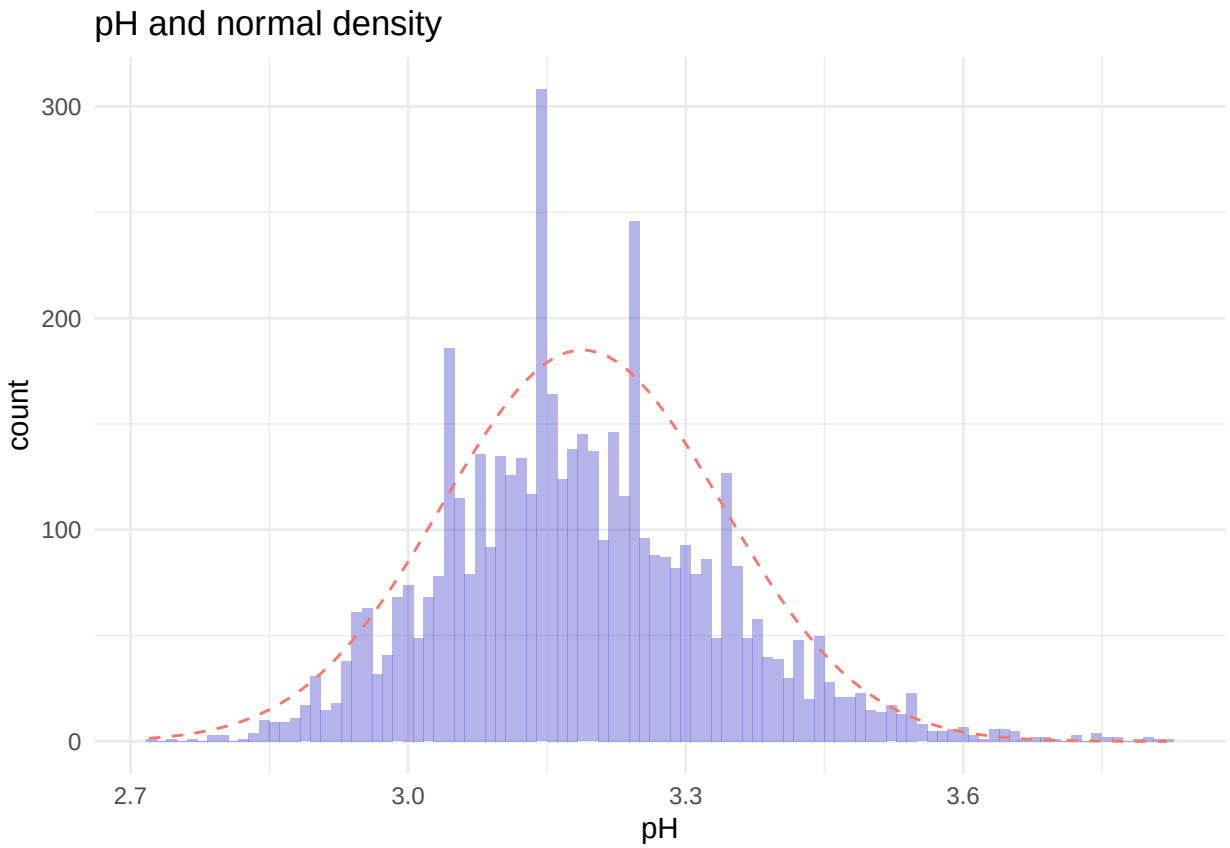
```
## [1] 3.188267
```

```
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
```

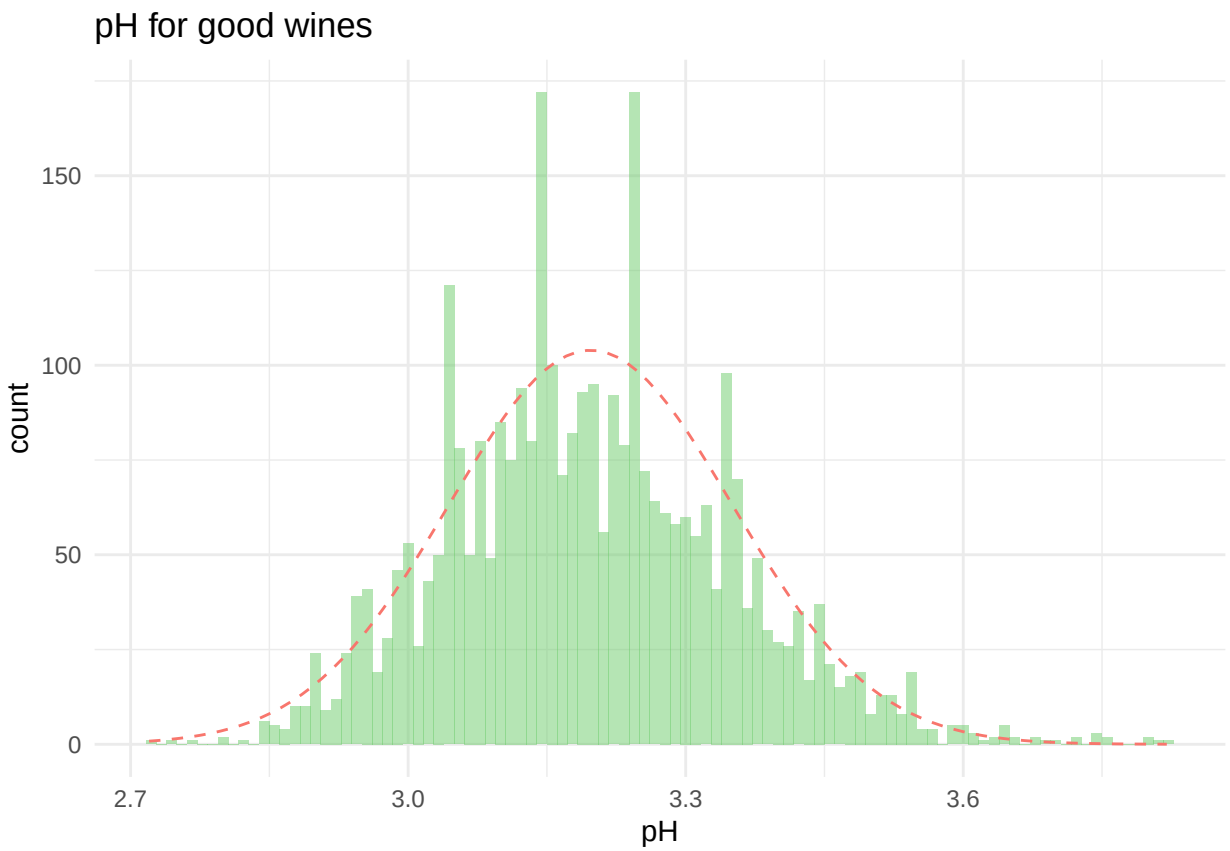
```r
mean_pH_bad
```

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

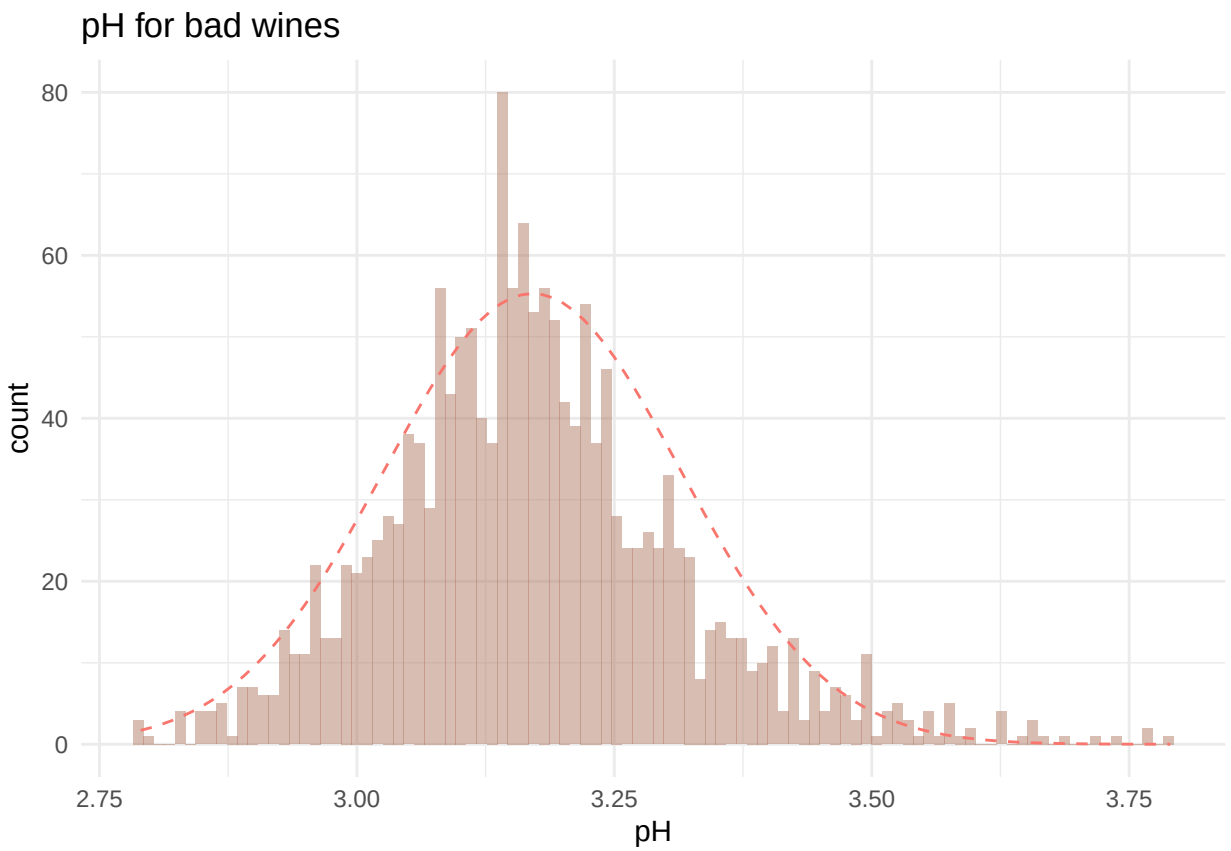
```r
sd_pH_bad
```

```
## [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"
  )
```

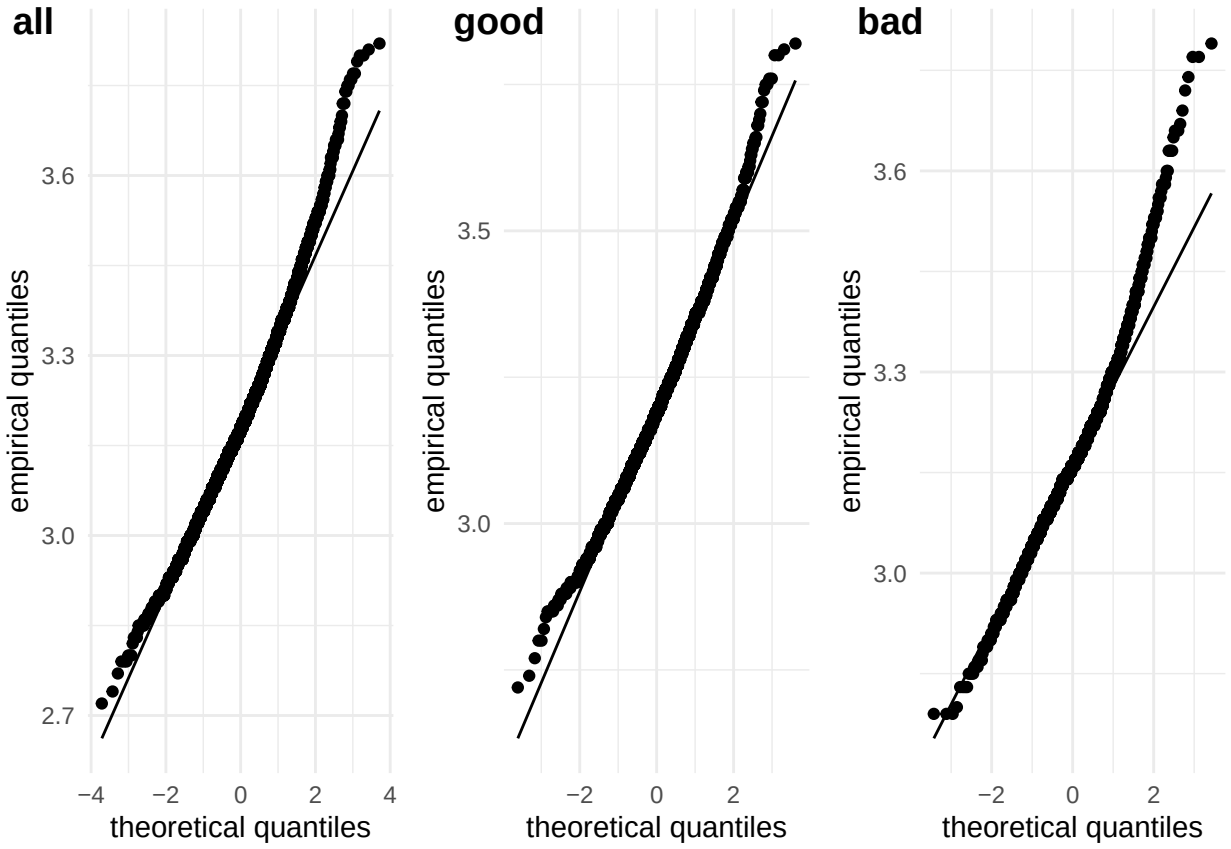
- good wines

```
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"
  )
```

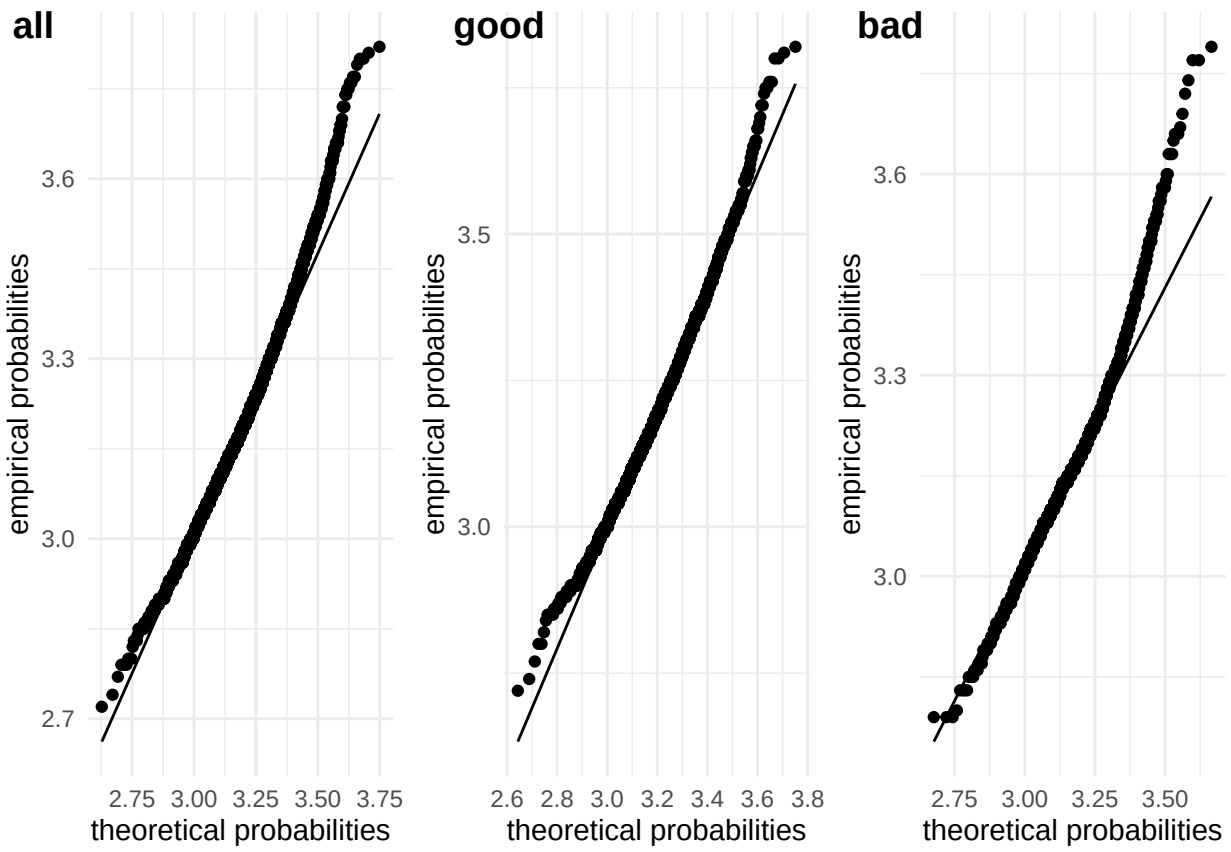
- good wines

```
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)
```

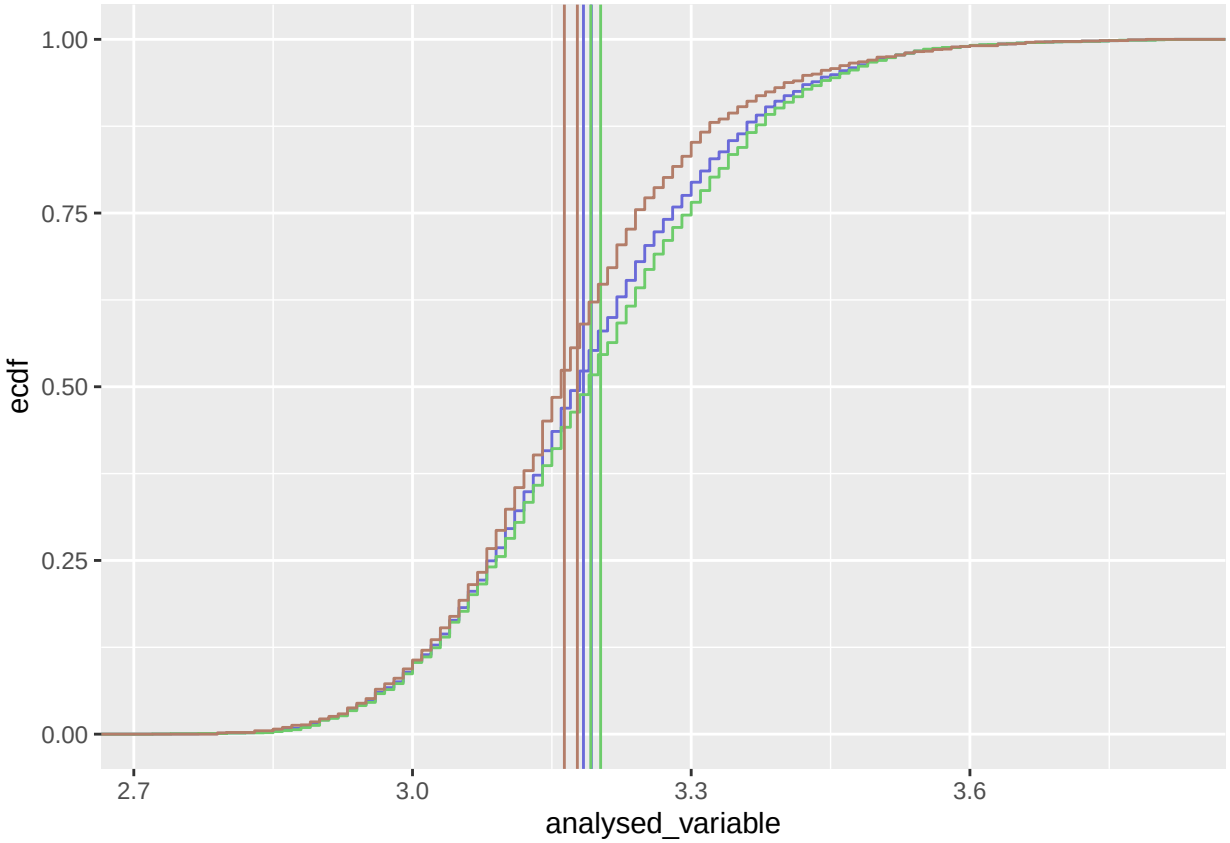


Samples probably do not follow normal distribution, as tails of QQ and PP plots do not lay on the line.

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")
```



d. EDF + uniform confidence bands

e. EDFs for good and bad wines

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()
```

```r
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

```r
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.

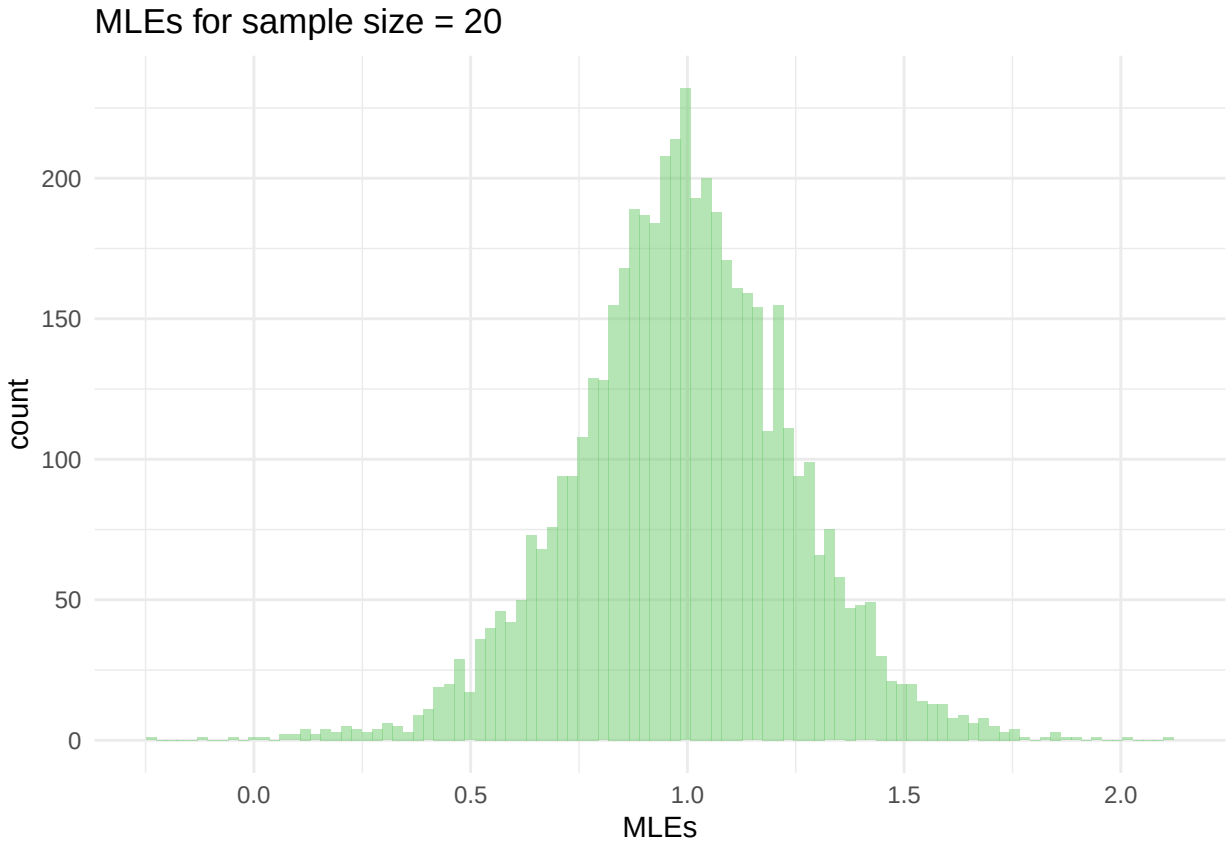
d. MLE distribution

- sample size = 20

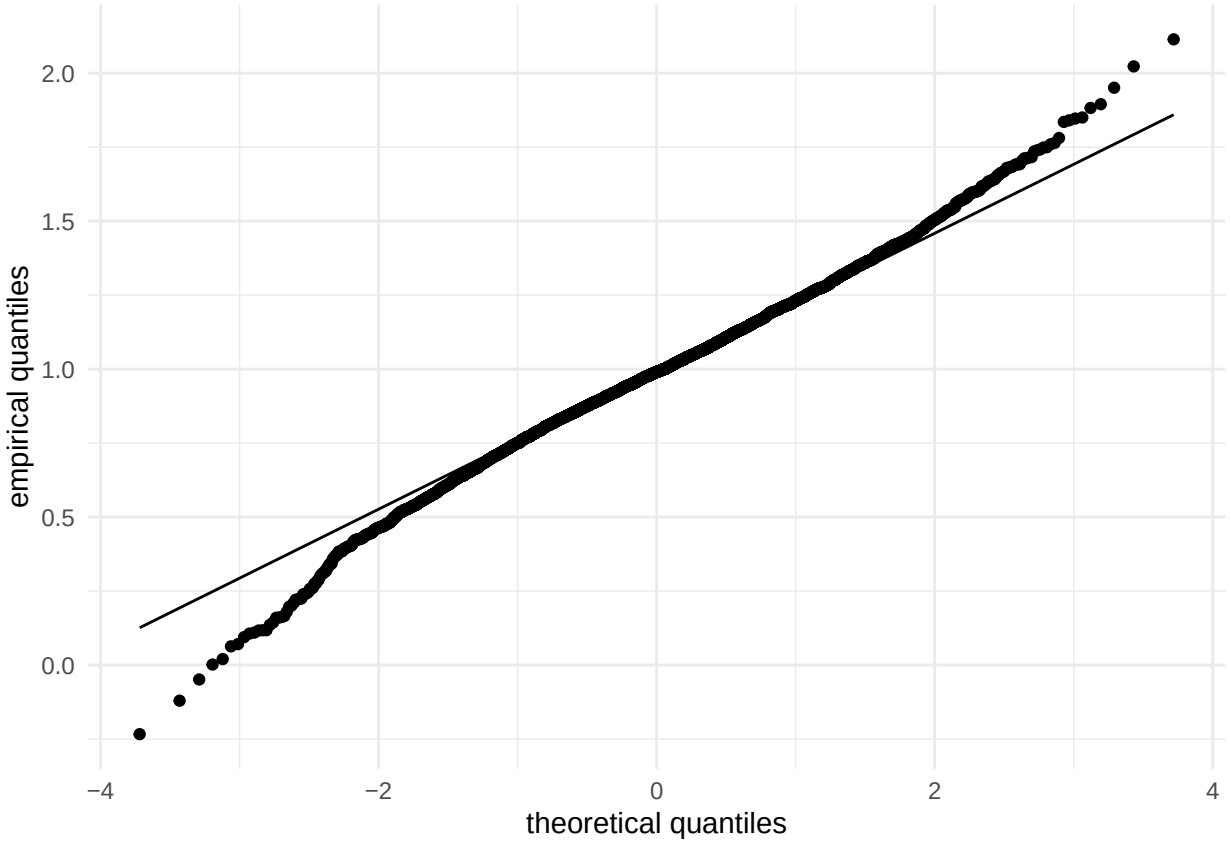
```
set.seed(1000)

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
```

```
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"
  )
```

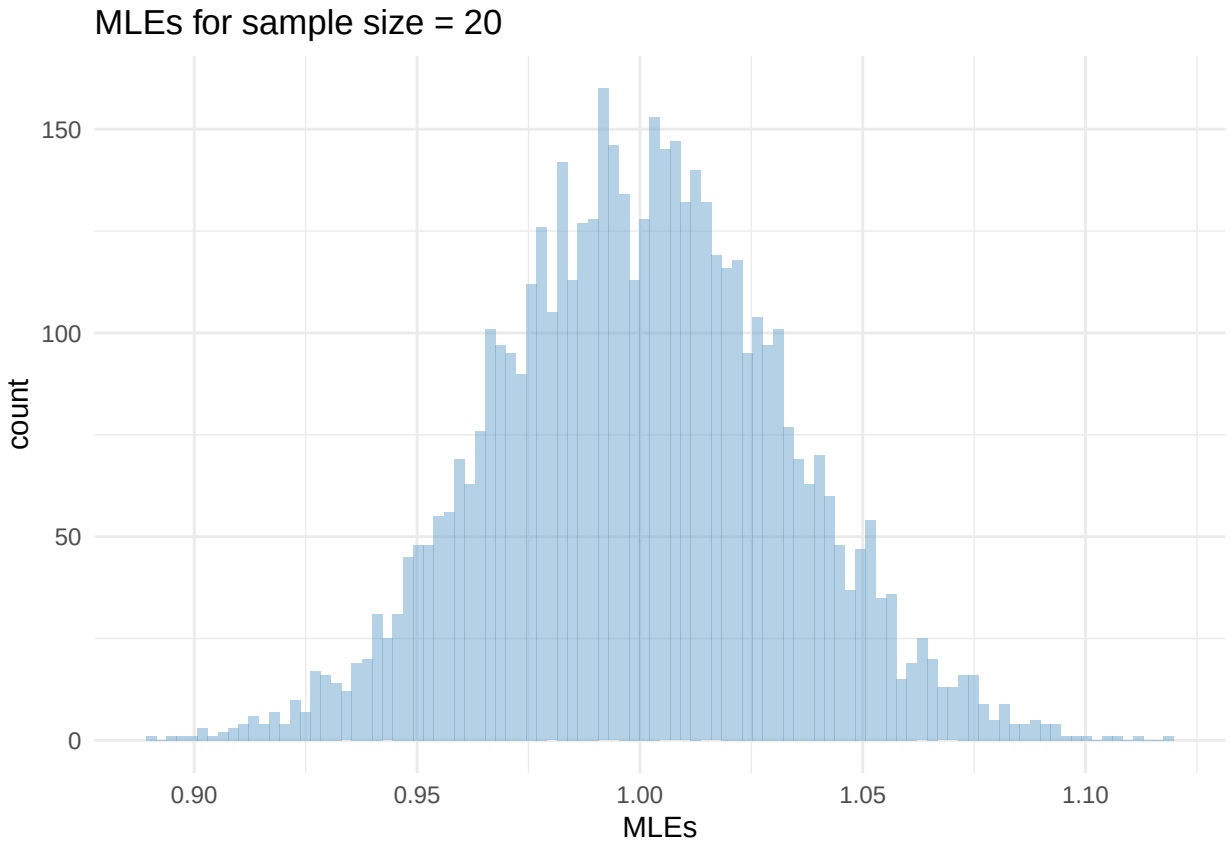


- sample size = 1000

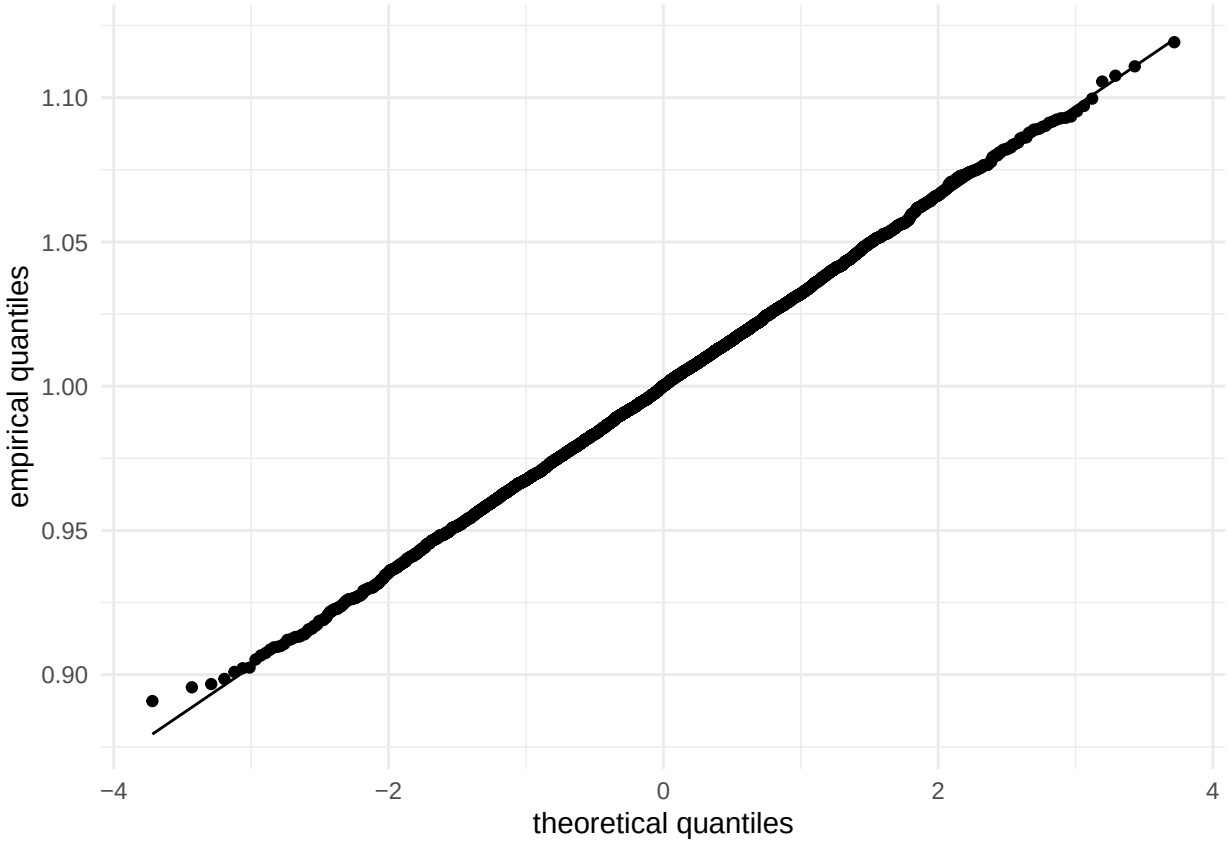
```
set.seed(1000)

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)
```

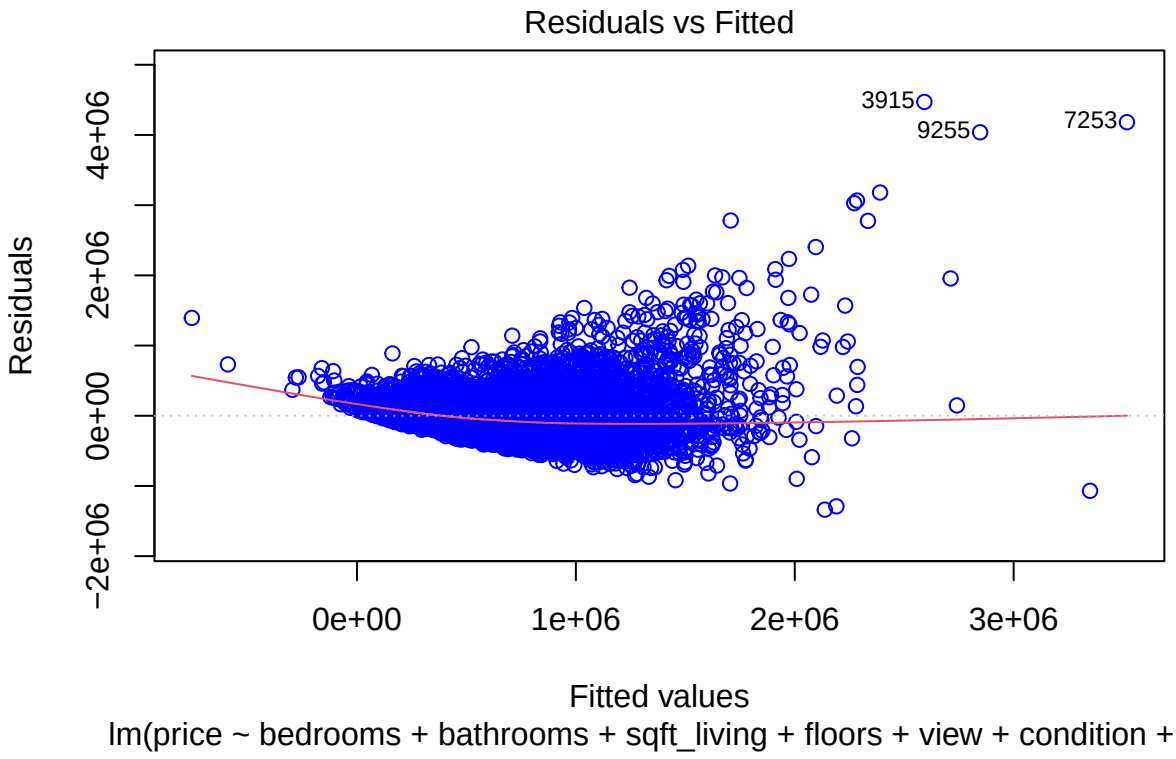
```
##
## Call:
## lm(formula = price ~ bedrooms + bathrooms + sqft_living + floors +
##     view + condition + grade + yr_built, data = house)
```

```
##
## 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
```

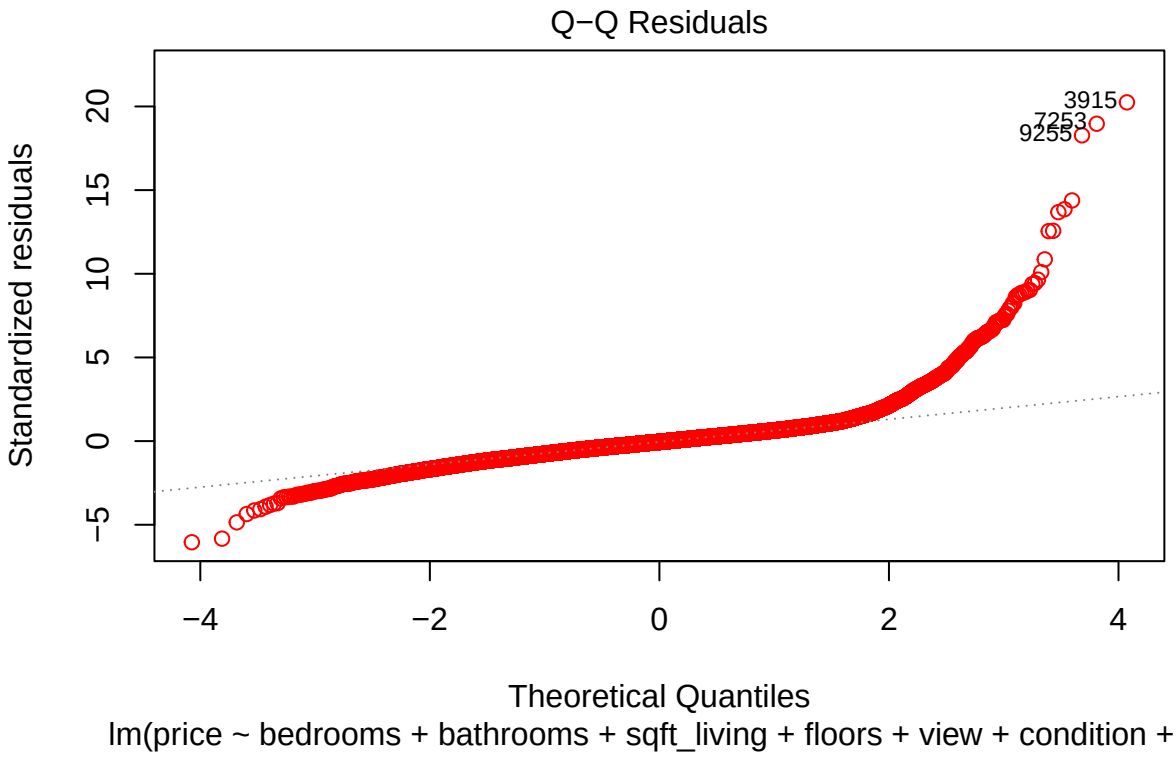
All variables are significant, R^2 equals to 0.6359 which means that $\sim 64\%$ of variance in the price is explained by those variables

- residual analysis

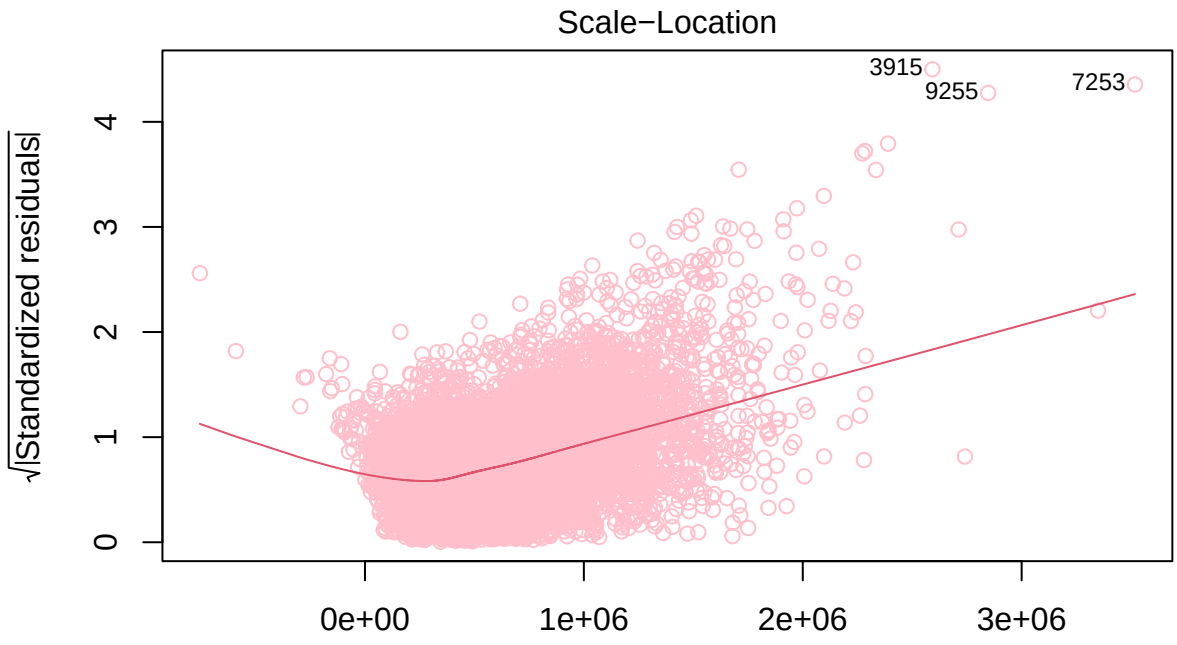
```
# residuals vs fitted
res_fit <- plot(model_price, which=1, col=c("blue"))
```




```
# 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)
```

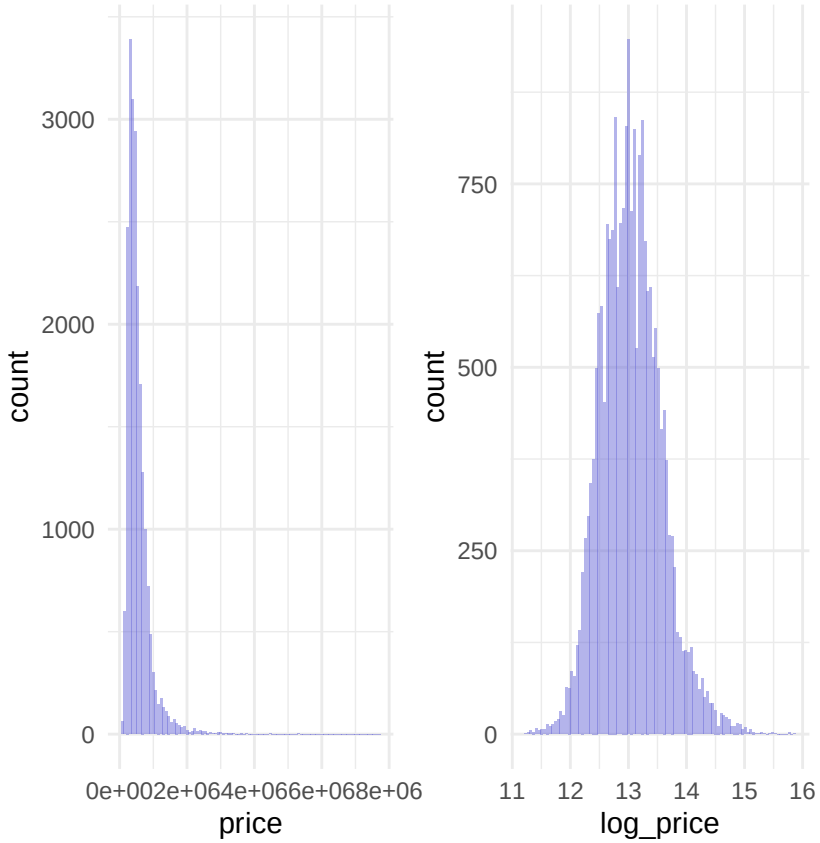
Residuals are not normally distributed and their variance is not the same, which means that our model violates normality and homoskedasticity of errors assumptions.

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")
```

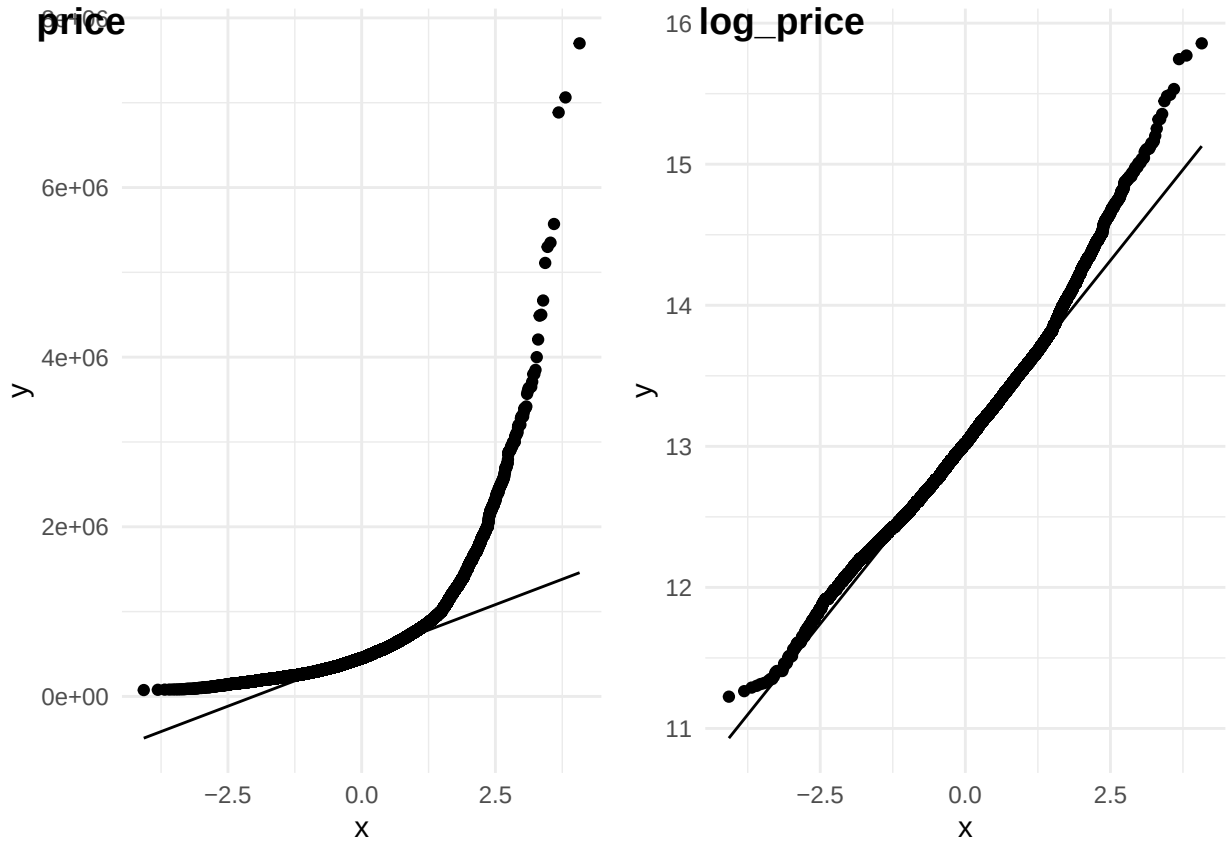
```
ggarrange(price_hist, log_price_hist, ncol = 3, nrow = 1)
```



- 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.

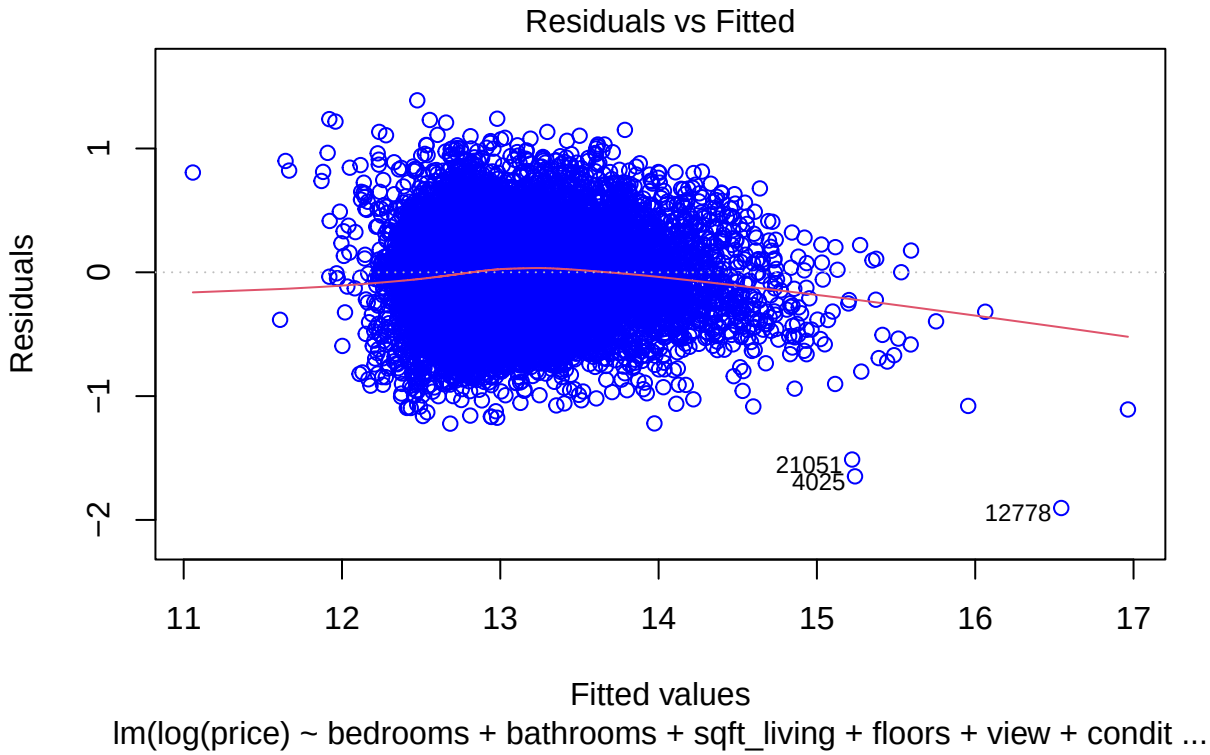
- model fit

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

summary(model_log_price)
```

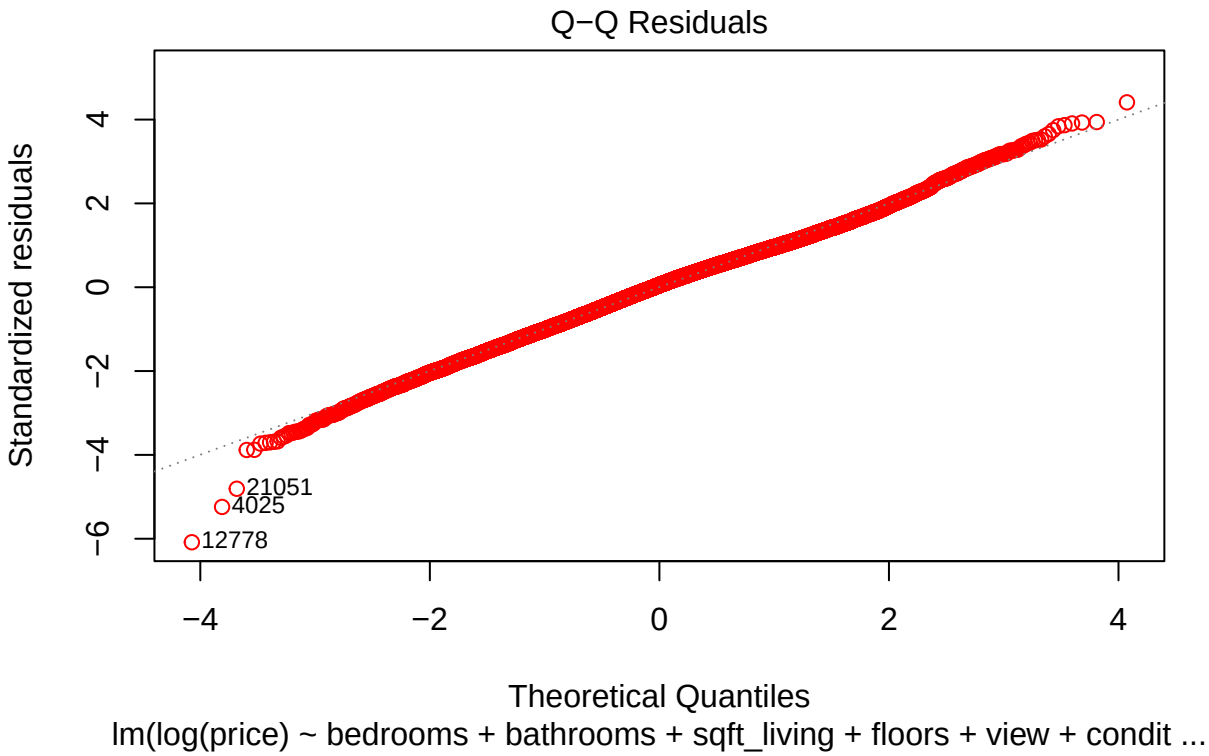
```
##
## Call:
## lm(formula = log(price) ~ bedrooms + bathrooms + sqft_living +
##     floors + view + condition + grade + yr_built, data = house)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.90374 -0.21157  0.01624  0.21288  1.38880
##
## Coefficients:
##               Estimate 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

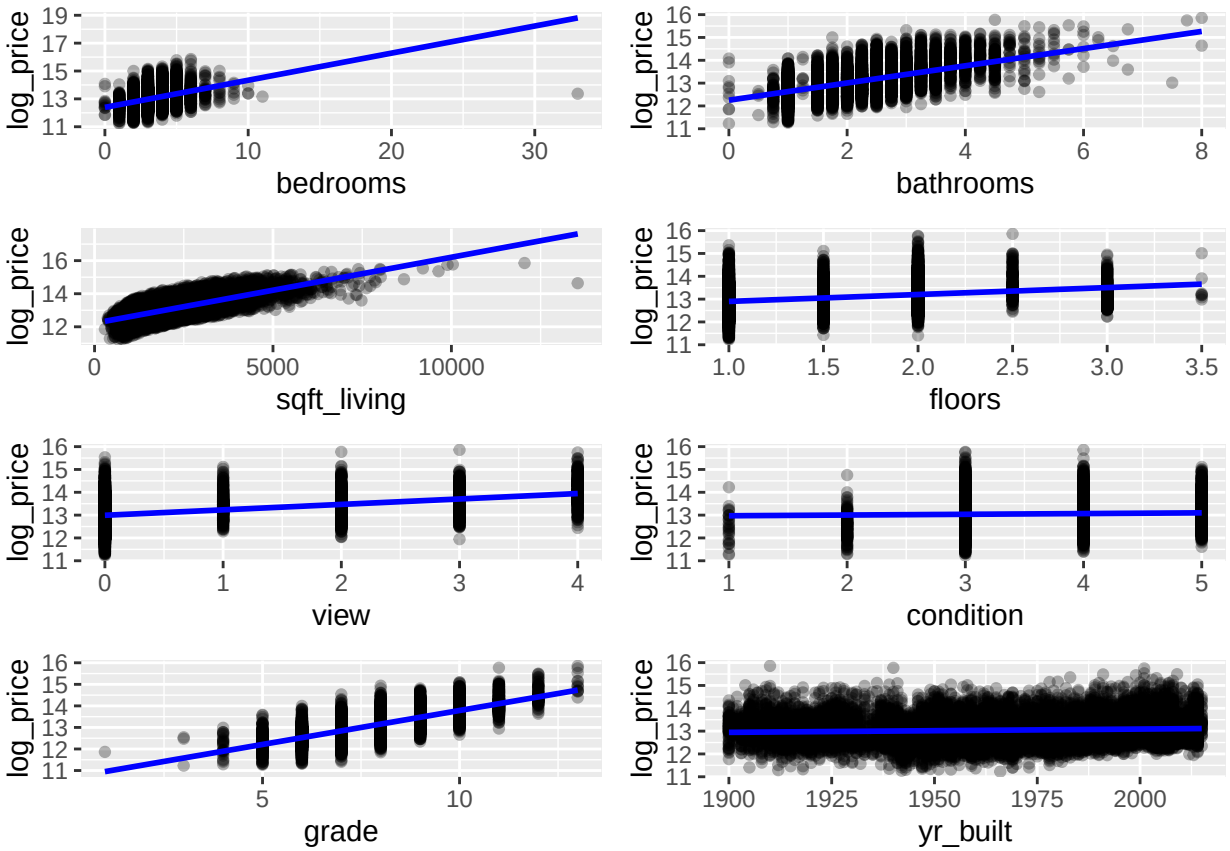
- intercept: expected value of log(price) is ~ 21.51 when all other parameters are zero
- bedrooms: each additional bedroom decreases the price by ~ 2.34% This, actually, doesn't make sense, by looking at the graph below, seems that the one outlier changes the sign of the coefficient, model fit, however, 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")
#  labs(title = "Log(price) vs num of bedrooms", x = "bedrooms", y = "log(price)")

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

```
  geom_point(alpha = 0.3) +
  geom_smooth(method = "lm", se = FALSE, color = "blue")
#  labs(title = "Log(price) vs num of bathrooms", x = "bathrooms", y = "log(price)")

sqft <- ggplot(house, aes(x = sqft_living, y = log_price)) +
  geom_point(alpha = 0.3) +
  geom_smooth(method = "lm", se = FALSE, color = "blue")
#  labs(title = "Log(price) vs sqft", x = "sqft", y = "log(price)")

floors <- ggplot(house, aes(x = floors, y = log_price)) +
  geom_point(alpha = 0.3) +
  geom_smooth(method = "lm", se = FALSE, color = "blue")
#  labs(title = "Log(price) vs num of floors", x = "floors", y = "log(price)")

views <- ggplot(house, aes(x = view, y = log_price)) +
  geom_point(alpha = 0.3) +
  geom_smooth(method = "lm", se = FALSE, color = "blue")
#  labs(title = "Log(price) vs num of views", x = "views", y = "log(price)")

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

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

yr_built <- ggplot(house, aes(x = yr_built, y = log_price)) +
  geom_point(alpha = 0.3) +
  geom_smooth(method = "lm", se = FALSE, color = "blue")
#  labs(title = "Log(price) vs year built", x = "year built", y = "Log(price)")
```

```
ggarrange(bedrooms, bathrooms, sqft, floors, views, condition, grade, yr_built,
#  labels = c("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

And I also removed an outlier in bedrooms here to change the sign of the coefficient

```
house_clean <- house %>%
  filter(bedrooms < 33)

summary(house_clean)
```

```
##      price            bedrooms        bathrooms      sqft_living
##  Min.   :  75000   Min.   : 0.000   Min.   :0.000   Min.   :  290
##  1st Qu.: 321838   1st Qu.: 3.000   1st Qu.:1.750   1st Qu.: 1426
##  Median : 450000   Median : 3.000   Median :2.250   Median : 1910
##  Mean   : 540084   Mean   : 3.369   Mean   :2.115   Mean   : 2080
##  3rd Qu.: 645000   3rd Qu.: 4.000   3rd Qu.:2.500   3rd Qu.: 2550
##  Max.   :7700000   Max.   :11.000   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
```

```
log_price_clean <- log(house_clean$price)
```

```
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
```

I don't know why the coefficient for bedrooms is still negative, while on the graph the slope is for sure positive

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: 1. adding interaction bedrooms*bathrooms as they are correlated (bathrooms means bathroom per bedroom) 1. adding interaction condition*grade as they also seem correlated

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

```
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)
```

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

```
mse_log_price_sq
```

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

MSE in this case is a bit smaller