Technical University of Munich
Department of Mathematics

Prof. Donna Ankerst, Stephan Haug

November 14, 2017

Grading: For each ✓ you should give 2, for each ✓ 1 and for each ✓ 0.5 points. For each part there exist also other approaches to solve the problem. Hence you have to check if solutions, which are different than the proposed one, are also correct.

Remark: If in the end the total number of points is not an integer (moodle only knows integers), you have to round up. So e.g. a result of 8.5 will be evaluated as 9 in moodle.

Problem H.2

Consider again the LA ozone data from problem H.1.

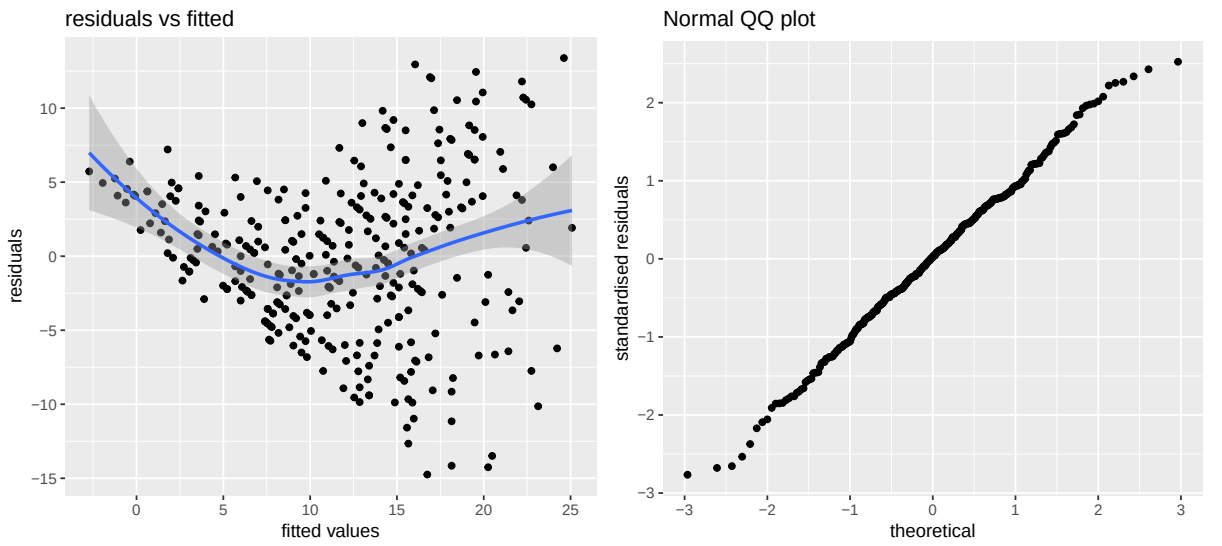
a) Fit two simple linear regression models to the data using `lm()`. Determine the variable, which has the highest correlation with the response `ozone`. Use this variable as predictor in the first model. The second model uses the variable `season`, defined in H.1 f) as predictor variable. Interpret the estimate b_1 for both models.

b) Perform a graphical residual analysis by plotting the two `lm`-objects from part a). Discuss in each case the first two plots. Which assumptions of the simple linear regression can you check using those two plots? Are these assumptions satisfied for the two models?

c) Which model do you prefer? Give reasons for your decision.

d) Now write a function `qplot_lm()`, which has as input a `lm`-object and the argument `cat_pred`, which indicates if the predictor variable of the model is categorical. Think about where to use this information. The output of the function is a plot of the form

```
qplot_lm(lm_model, cat_pred = FALSE)
```



Hence, it should produce one graphic showing the two plots side by side. Finally apply `qplot_lm()` to both models from part a).

Hint: Visit `https://www.statmethods.net/management/userfunctions.html` for information on how to write your own function. You might also want to use the functions `rstandard()`

and `gridExtra::grid.arrange()` (from the `gridExtra` package).

Remark: The notation `::` allows you to use `grid.arrange()` without loading in advance `gridExtra` with the `library()` command.

Solution

First we have to read in the data.

```
library(tidyverse)
LAozone <- read_csv("LAozone.csv")
```

a) To determine the predictor variable of the first model, we have to compute the empirical correlations between ozone and all the other variable. Hence, we use `cor()` to compute the empirical correlation matrix.

```
cor(LAozone)
```

```
##                ozone          vh        wind    humidity        temp
## ozone     1.00000000  0.60734379 -0.01341352  0.44922399  0.78070283
## vh        0.60734379  1.00000000 -0.24366363  0.07448508  0.80805889
## wind     -0.01341352 -0.24366363  1.00000000  0.21029305 -0.03207041
## humidity  0.44922399  0.07448508  0.21029305  1.00000000  0.34047421
## temp      0.78070283  0.80805889 -0.03207041  0.34047421  1.00000000
## ibh      -0.58953415 -0.50483503  0.20659647 -0.24232766 -0.53264472
## dpg       0.21404639 -0.14807054  0.33574699  0.64778894  0.18924192
## ibt       0.74557817  0.85202122 -0.17951744  0.20364788  0.86478656
## vis      -0.44098946 -0.36008025  0.14722694 -0.40100846 -0.38772104
## doy       0.06619978  0.33739672 -0.24644427  0.04070258  0.23800321
## id        0.07693707  0.34160255 -0.24344339  0.04606355  0.24840805
##                 ibh         dpg         ibt        vis         doy
## ozone    -0.58953415  0.21404639  0.74557817 -0.4409895  0.06619978
## vh       -0.50483503 -0.14807054  0.85202122 -0.3600802  0.33739672
## wind      0.20659647  0.33574699 -0.17951744  0.1472269 -0.24644427
## humidity -0.24232766  0.64778894  0.20364788 -0.4010085  0.04070258
## temp     -0.53264472  0.18924192  0.86478656 -0.3877210  0.23800321
## ibh       1.00000000  0.03707790 -0.77693259  0.3866858  0.04340027
## dpg       0.03707790  1.00000000 -0.09506021 -0.1258547 -0.15413843
## ibt      -0.77693259 -0.09506021  1.00000000 -0.4223720  0.21917185
## vis       0.38668582 -0.12585472 -0.42237201  1.0000000 -0.21697589
## doy       0.04340027 -0.15413843  0.21917185 -0.2169759  1.00000000
## id        0.03848732 -0.14410457  0.22626084 -0.2206765  0.99973012
##                  id
## ozone     0.07693707
## vh        0.34160255
## wind     -0.24344339
## humidity  0.04606355
## temp      0.24840805
## ibh       0.03848732
## dpg      -0.14410457
## ibt       0.22626084
## vis      -0.22067649
## doy       0.99973012
## id        1.00000000
```

In the above given matrix we just have to focus on the first row (or first column)

```
cor(LAozone)[1,]

##       ozone          vh        wind    humidity        temp         ibh
##  1.00000000  0.60734379 -0.01341352  0.44922399  0.78070283 -0.58953415
##         dpg         ibt         vis         doy          id
##  0.21404639  0.74557817 -0.44098946  0.06619978  0.07693707
```

There we see that `temp` has the highest correlation with ozone and will therefore be our predictor variable

```
model_1 <- lm(ozone ~ temp, data = LAozone)
```

✓ for `model_1` (which includes ✓ for selecting `temp` based on the result from `cor()`)

To fit the second model, we firs have to define the variable `season` (see solution to problem H.1)

```
LAozone$season <- factor(LAozone$doy <= 89 | LAozone$doy > 273,
                         levels = c(TRUE, FALSE), labels = c("winter", "summer"))
```

Now we can fit the second model

```
model_2 <- lm(ozone ~ season, data = LAozone)
```

✓ for `model_2`

Next we take a look at the estimated coefficients.

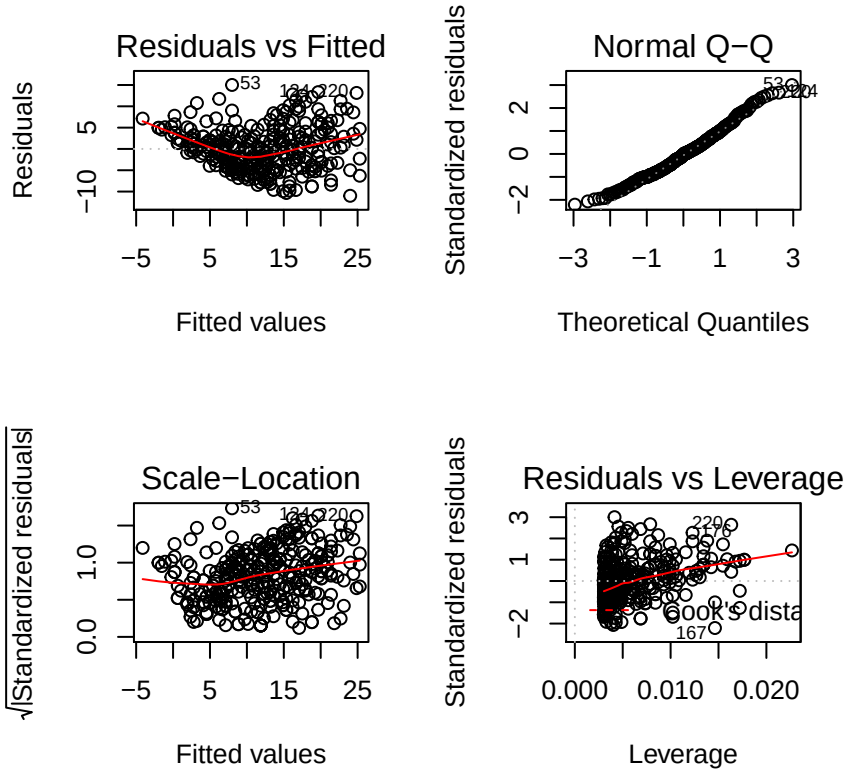
```
coef(model_1)

## (Intercept)        temp
## -14.9374539   0.4325708

coef(model_2)

##  (Intercept) seasonsummer
##     7.772455     8.104846
```

In the first model, b_1 is the estimated slope. So, the model predicts e.g. in case of an increase of $10°F$ an increase in ozone of 4.3257077 percentage points. ✓

Since `season` is a categorical predictor variable, b_1 is not a slope in the second model. Instead it is the mean difference in ozone concentration between the two levels `winter` and `summer`. From the above R output we can see, that `winter` is the reference category. Hence, 8.1048455 is the increase in the mean ozone concentration for summer observations compared to winter observations. ✓

b) We start the residual analysis with the first model.

```
par(mfrow = c(2,2)) # splits the graphic device in a 2 x 2 grid
plot(model_1)
```

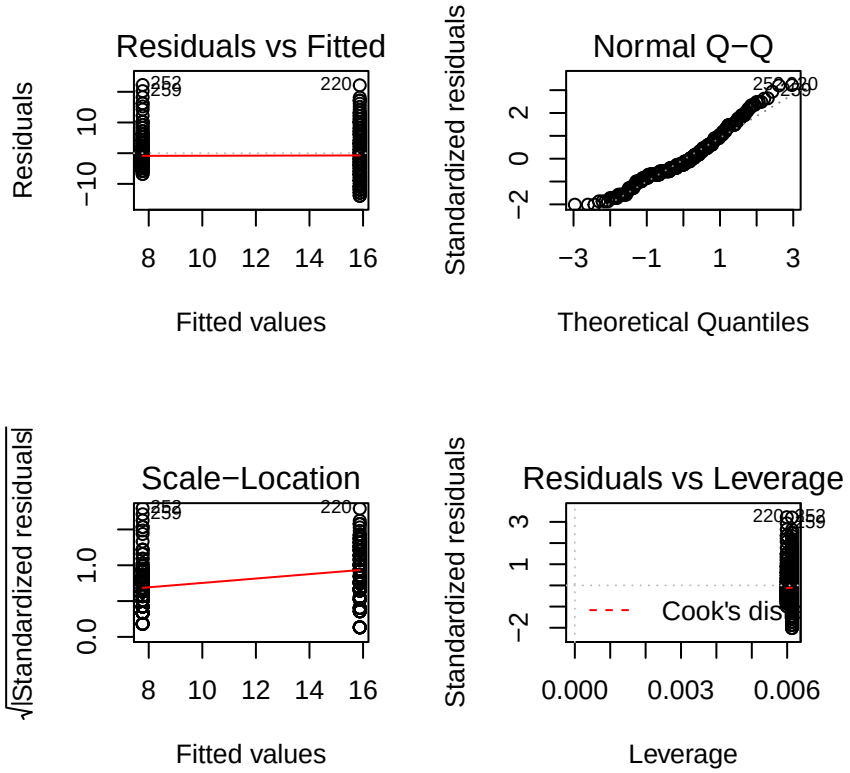


Using the first two plots we can check the assumptions 2.) to 4.) on page 13 of Lecture 1a (*Remark: The third plot is more suitable to check assumption3.), but we can also use the first one*). ✓

The first plot shows a scatterplot of the fitted values $\widehat{y}_i = b_0 + b_1 x_i$ against the residuals $y_i - \widehat{y}_i = y_i - (b_0 + b_1 x_i)$. So if the response depends on the predictor in a linear way, then the residuals should not depend on the fitted values in a systematic way. The red line shows a non-parametric fit to the relation between the two variables, which indicates in this case a slight quadratic dependence. ✓ Further, the plot shows an increased variability for growing fitted values, i.e. that assumption 3.) seems not fulfilled. ✓ The forth assumption can be checked with the Normal Q-Q (Probability) plot. The points follow more or less a straight line, which indicates that the empirical distribution of the residuals is close to normal distribution. ✓

Now we take a look at the residuals of the second model

```
par(mfrow = c(2,2))
plot(model_2)
```



The first plot looks strange compared to the first model. But of course, this time we can just separate the data into two groups (winter vs. summer) by using the predictor `season`. Hence, we are just able to predict two different ozone values. Within those two groups the variation seems similar (slightly larger for summer). ✓ Since `season` is categorical, there is obviously no violation of assumption 2.). ✓ . The Normal Q-Q plot indicates that assumption 4.) is also fulfilled. ✓

c) To answer the question, which model would be preferred, we take a look at the summary of each model. For `model_1` we get

```
summary(model_1)

##
## Call:
## lm(formula = ozone ~ temp, data = LAozone)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -10.9939  -3.8202  -0.1796   3.1951  15.0112
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -14.93745    1.21247  -12.32   <2e-16 ***
## temp          0.43257    0.01912   22.63   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
```

```
## Residual standard error: 5.014 on 328 degrees of freedom
## Multiple R-squared:  0.6095,Adjusted R-squared:  0.6083
## F-statistic: 511.9 on 1 and 328 DF,  p-value: < 2.2e-16
```

We detect a $r^2 (= R^2)$ value of 0.6094969 and a highly significant influence of `temp` (compare also problem C.2). This means, that the model can explain roughly 61 % of the variation in `ozone`.

The summary of `model_2` is

```
summary(model_2)

##
## Call:
## lm(formula = ozone ~ season, data = LAozone)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -13.877  -4.772  -1.772   4.123  22.227
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)    7.7725     0.5353   14.52   <2e-16 ***
## seasonsummer   8.1048     0.7617   10.64   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 6.918 on 328 degrees of freedom
## Multiple R-squared:  0.2566,Adjusted R-squared:  0.2543
## F-statistic: 113.2 on 1 and 328 DF,  p-value: < 2.2e-16
```

and yields that `season` has a highly significant influence on `ozone` and that the model can explain roughly 26 % of the variation in `ozone`.

Possible conclusions are

- One prefers `model_1` due to the larger r^2 value compared to model `model_2`.
- One prefers `model_2`, since not all assumptions are satisfied for `model_1`.

For one of the above conclusions with a similar justification ✓

Remark: The violation of assumption 3.) is not so severe, but this is probably hard to judge at the start.

d) In this part we have to write a function `qplot_lm()`, which draws the given graph. In general a R function has the following structure

```
myfunction <- function(arg1, arg2, ... ){
  statements
  return(object)
}
```

In our case, the function should not return anything. It should just produce the residual vs. fitted and the Normal Q-Q plot. Since the input is just a `lm`-object, we have to define

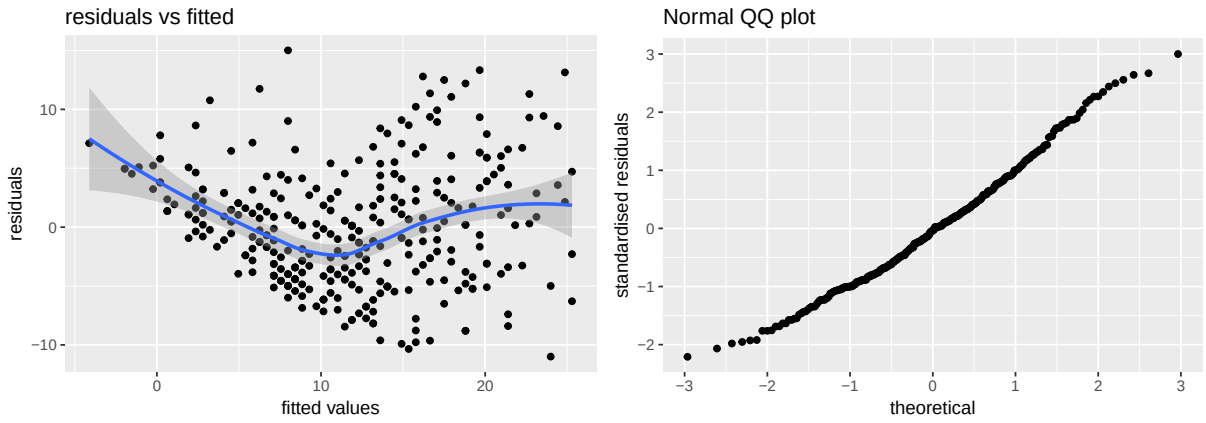
the residuals, standardized residuals and fitted values in the body of the function. Therefore we will use the functions `residuals()`, `rstandard()` and `fitted.values()`. Afterwards we combine those vectors in a data frame (remember that the input to `ggplot()` always has to be data frame). Afterwards we use `geom_point()` and `geom_smooth()` for the residual vs. fitted plot and `stat_qq()` for the Normal Q-Q plot. In the last one, we do not need to specify any arguments, since drawing a Normal Q-Q plot is the default option. Finally, both plots are arranged side by side with the function `gridExtra::grid.arrange()`.

A function, which does the job, is the following

```
qplot_lm <- function(model, cat_pred = FALSE){


  fitted <- fitted.values(model)
  residuals <- residuals(model)
  stand_res <- rstandard(model)

  res_analysis <- data.frame(fitted, residuals, stand_res)

  if(cat_pred == FALSE){
    plot1 <- ggplot(res_analysis, aes(x = fitted, y = residuals)) + geom_point() +
    geom_smooth() + xlab("fitted values") + ggtitle("residuals vs fitted")

    plot2 <- ggplot(res_analysis, aes(sample = stand_res)) + stat_qq() +
    ylab("standardised residuals") + ggtitle("Normal QQ plot")
  } else {
    plot1 <- ggplot(res_analysis, aes(x = fitted, y = residuals)) + geom_point() +
    geom_smooth(method = "lm") + xlab("fitted values") +
      ggtitle("residuals vs fitted")

    plot2 <- ggplot(res_analysis, aes(sample = stand_res)) + stat_qq() +
    ylab("standardised residuals") + ggtitle("Normal QQ plot")
  }

  gridExtra::grid.arrange(plot1, plot2, ncol = 2)
}
```

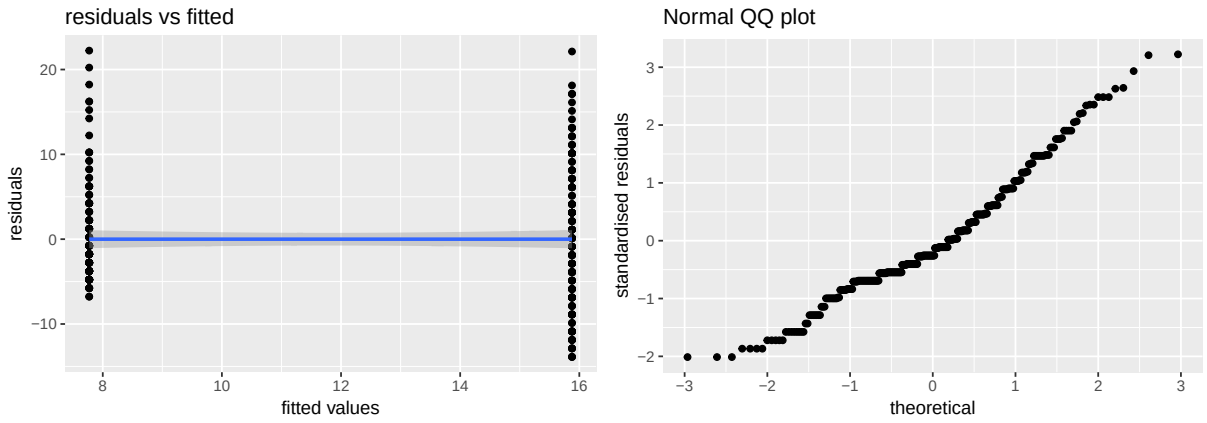
Now we can apply `plot_lm()` to `model_1`

```
qplot_lm(model_1)
```



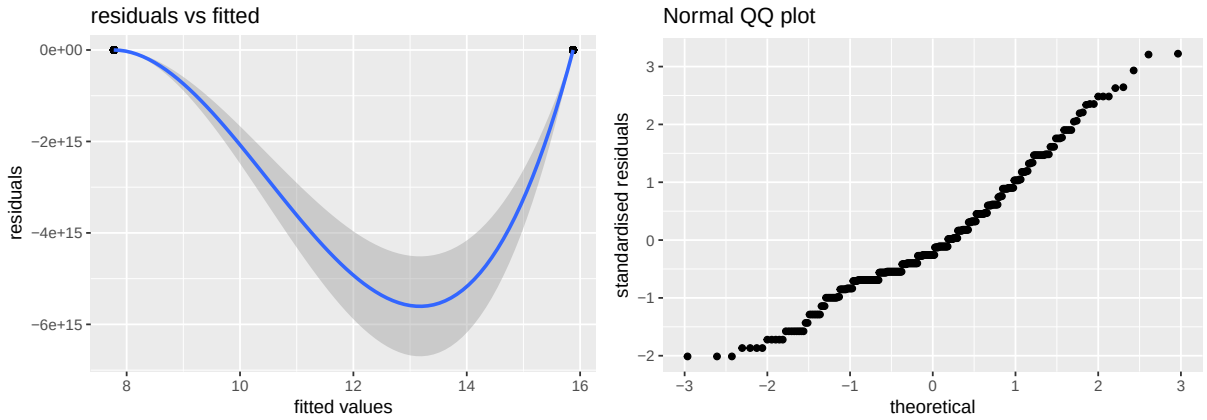
and `model_2`

```
qplot_lm(model_2, cat_pred = TRUE)
```



Remark: The argument `cat_pred` was needed, since the function `loess()` used as default in `geom_smooth()` can't handle this kind of data, as you see here

```
qplot_lm(model_2)
```



The proposed solution also just works for a factor variable with two categories. But this is sufficient for the current problem. A solution, which works in general, would look like this

```
qplot_lm <- function(model, cat_pred = FALSE){


  fitted <- fitted.values(model)
  residuals <- residuals(model)
  stand_res <- rstandard(model)
  # we need the predictor
  predictor <- model$model[,2]

  res_analysis <- data.frame(fitted, residuals, stand_res, predictor)

  if(cat_pred == FALSE){
    plot1 <- ggplot(res_analysis, aes(x = fitted, y = residuals)) + geom_point() +
    geom_smooth() + xlab("fitted values") + ggtitle("residuals vs fitted")

    plot2 <- ggplot(res_analysis, aes(sample = stand_res)) + stat_qq() +
    ylab("standardised residuals") + ggtitle("Normal QQ plot")
```

```
  } else {
    # the scatterplot is the same as before
    plot1 <- ggplot(res_analysis, aes(x = fitted, y = residuals)) + geom_point()
    # but now we need the mean of the residuals for each factor level; here we
    # use the dplyr package
    mean_res <- res_analysis %>%
      group_by(predictor) %>%
      summarise(mean_values = mean(residuals))
    # the corresponding values on the x-axis are
    mean_res$fitted <- unique(round(fitted,7))

    plot1 <- plot1 + geom_line(data = mean_res, aes(x = fitted, y = mean_values),
                               colour = "blue")+ xlab("fitted values") +
      ggtitle("residuals vs fitted")

    plot2 <- ggplot(res_analysis, aes(sample = stand_res)) + stat_qq() +
    ylab("standardised residuals") + ggtitle("Normal QQ plot")
  }

  gridExtra::grid.arrange(plot1, plot2, ncol = 2)
}
```

But such a solution was **not** expected. The above given one is fine.

Grading for part d):

✓ for a solution, which produces plots like the first two plots in this part (they do not need to include the confidence bands, although they are shown in the problem).

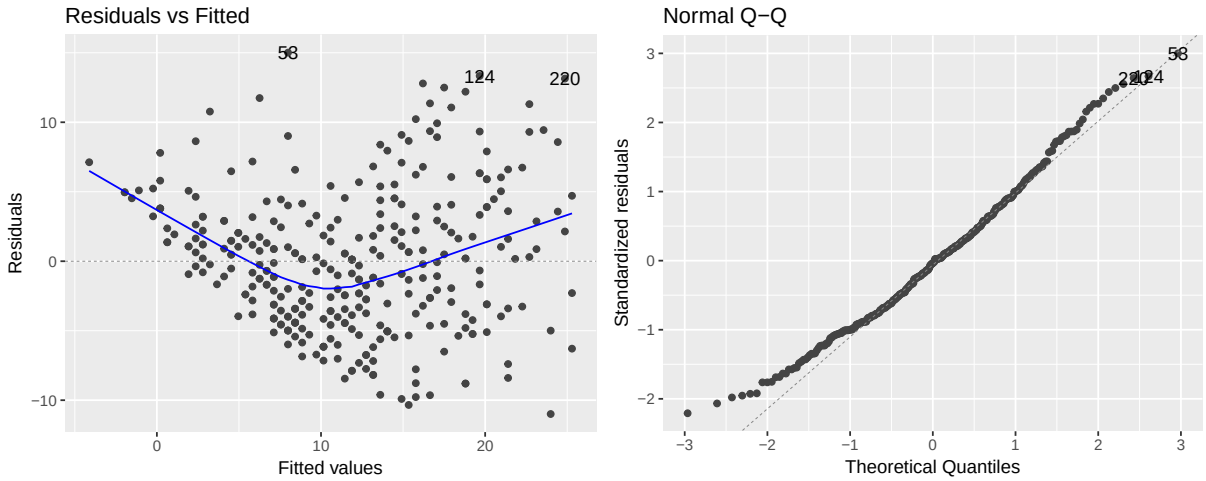
✓ ✓ for a solution, which produces for `model_2` a plot as in the remark, but works fine for `model_1`

✓ if the residuals and not the standardised residuals are used in the Normal Q-Q plot or if the two plots are not arranged in one graphic

✓ if the proposed function hasn't been applied to the two models, but one can guess from the code that the function works

Remark: One aspect of the last part was to write a function, which is a task needed from time to time, since the existing functions do not always fit. In this case however, there exists of course a function, which does a much better job then the proposed solution.

```
library(ggfortify)
autoplot(model_1, which = 1:2)
```



Hence

```
qplot_lm <- function(model){
  autoplot(model, which = 1:2)
}
```

would also be a solution, which obviously doesn't need the argument `cat_pred`, but depends on the package `ggfortify`.

```
qplot_lm(model_2)
```