Workflow Linear Models I

Chris Howden Stanislaus Stadlmann

Table of contents

```r
set.seed(171974)
project <- "R Workshop - Linear Models"
version <- "1"
date <- format(Sys.Date(), "%d-%m-%Y")
title <- paste(project, " v", version, " ", date, sep = "")
```

The title of this project is R Workshop - Linear Models v1 23-05-2024. The slides for the presentation accompanying this workflow can be found here.

In this workflow we focus on practical data analysis by presenting statistical workflows applicable in any software for four of the most common univariate analyses: linear regression, ANOVA, ANCOVA, and repeated measures (a simple mixed model) – all assuming a normal (gaussian) residual. These workflows can be easily extended to more complex models. The R code used to create output is also included.

In order to re-run this `.qmd` file, either open it with RStudio and press "render" or, if you want both the `.pdf` and the `.html` file to be produced, run the following code with this `.qmd` file being present in your working directory: `quarto::quarto_render("linear_modelsI_workflow.qmd", output_format = "all")`

Libraries

Before we delve into the content, it is necessary to load certain libraries. If you want to use them, please install them beforehand with `install.packages("library_name")`.

```r
suppressPackageStartupMessages({
  library("tibble")
  library("magrittr")
  library("ggplot2")
  library("emmeans")
  library("lme4")
  library("lmerTest")
  library("gglm")
  library("patchwork")
  library("writexl")
})
```

```
Warning: package 'emmeans' was built under R version 4.3.2
```

```
Warning: package 'writexl' was built under R version 4.3.2
```

```r
# GGplot theme
theme_set(theme_bw())

# No warnings
options(warn = -1)
```

Simple Linear Models

For this part, we assume that we have a dataset called `dataset` with two variables:

- `response` which is the response or target variable, i.e. the 'variable of interest'.
- `predictor1` which is the predictor variable, i.e. the variable that we are using to explain the variance in `response`.

```r
predictor1 <- rnorm(100, 6, 0.25)
variance <- rnorm(100, 0, 0.05)
response <- 1 + 0.5 * predictor1 + variance
dataset <- data.frame(response, predictor1)
```

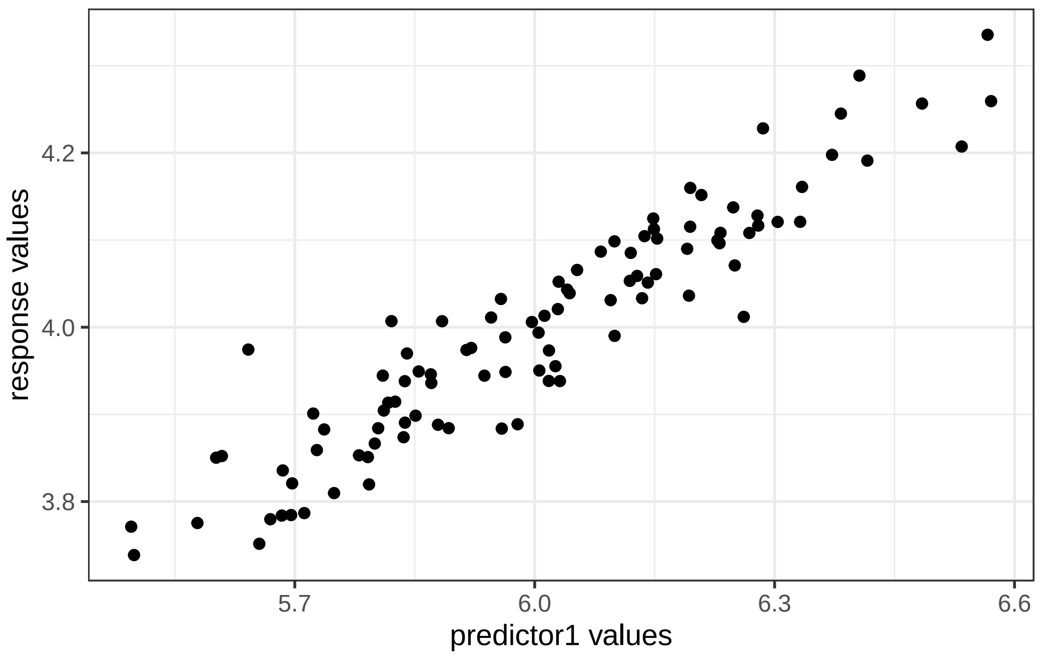
Step 1: Exploratory Data Analysis

In this section, we first check for validity of assumptions prior to the formal model fitting procedure using statistical diagnostics.

Linearity Assumption

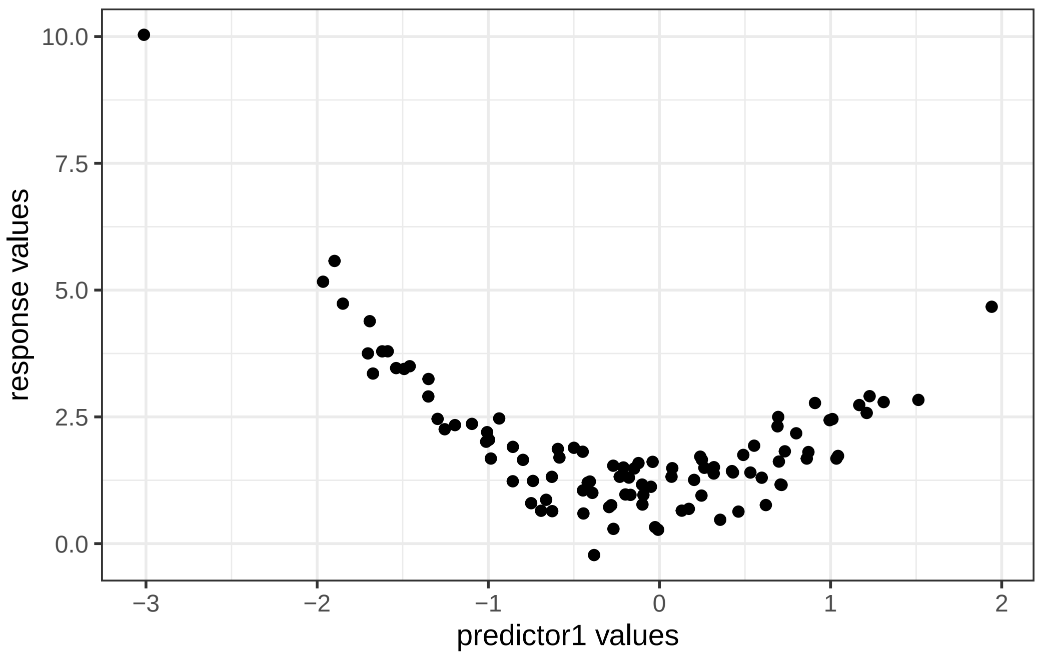
A simple linear model assumes a linear relationship between our predictor(s) and the target variable. In order to verify this, we create a scatterplot between `predictor1` and `response`:

```r
ggplot(data = dataset, aes(x = predictor1, y = response)) +
  geom_point() +
  labs(x = "predictor1 values", y = "response values")
```



As we can see, the `response` values rise nicely with increased values of `predictor1`, indicating a linear relationship. Have a look at a graph, in which linearity is clearly not present:

```r
tibble(x = rnorm(100), y = 1 + x^2 + rnorm(100, sd = 0.5)) %>%
  ggplot(data = ., aes(x = x, y = y)) +
  geom_point() +
  labs(x = "predictor1 values", y = "response values")
```

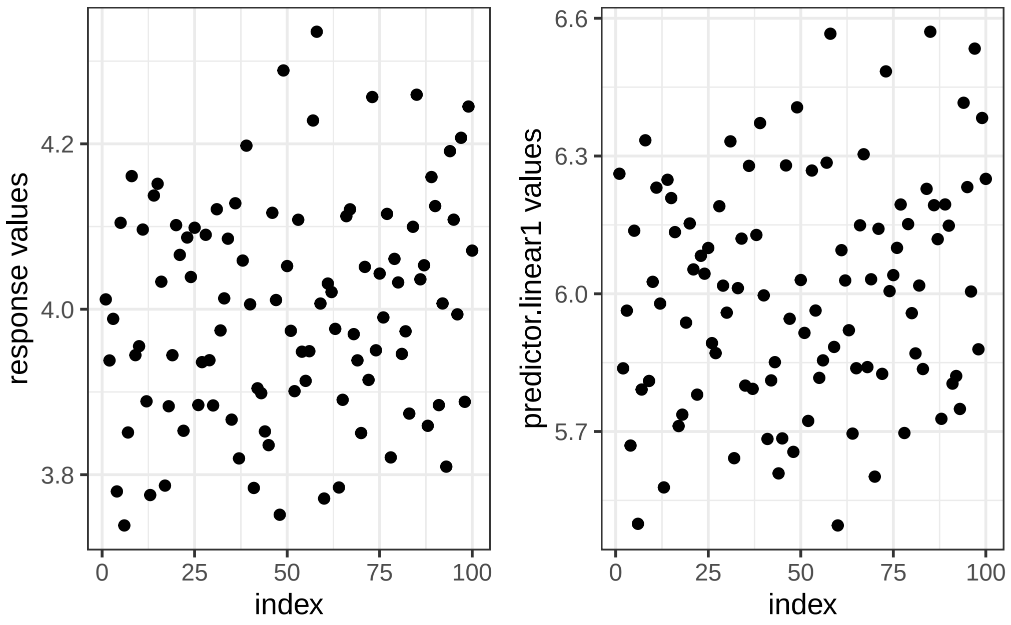


In this above graph, the response is actually dependent on x^2, not x.

Assumption of Independence

Another assumption of linear models is that the observations are not dependent on each other. One way to check this is to do a serial plot, which lines up all observations in order of appearance. If any pattern can be detected, they are likely to not be independent.

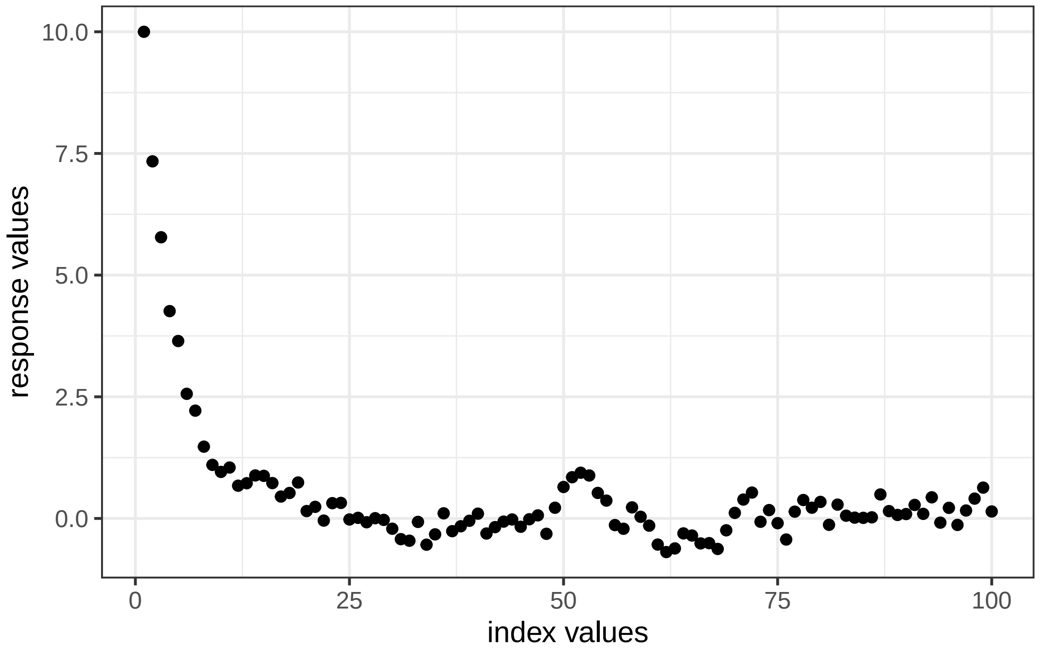
```
ggplot(data = dataset, aes(
  x = seq_along(response),
  y = response
)) +
  geom_point() +
  labs(x = "index", y = "response values") +
  ggplot(data = dataset, aes(
  x = seq_along(response),
  y = predictor1
)) +
  geom_point() +
  labs(x = "index", y = "predictor.linear1 values")
```



There are no visible patterns in the above graph. A problematic pattern could occur in time-dependent observations, like below:

```
y0 <- 10
y_1 <- stats::filter(c(y0, runif(99, -0.5, 0.5)), 0.75, method = "recursive")
qplot(seq_along(y_1), y_1, geom = "point") +
  labs(x = "index values", y = "response values")
```

```
Don't know how to automatically pick scale for object of type <ts>. Defaulting
to continuous.
```



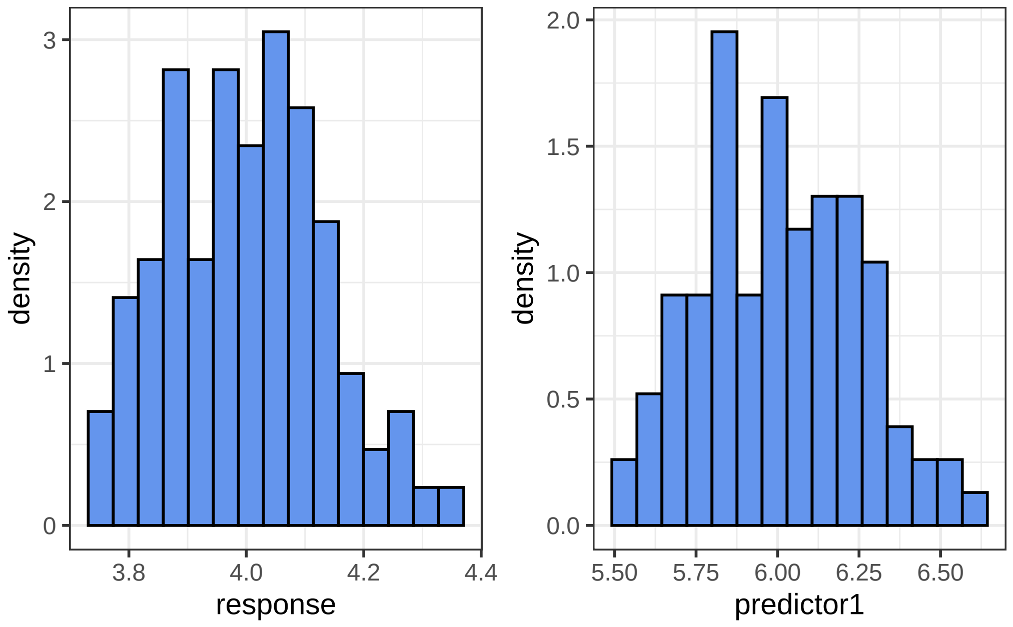
In the graph above, we can clearly see how each observations depend on the ones that came before.

Normality Assumption

The assumption of normality is technically only relevant for residuals, which can only be obtained post-model fit. But it's still a good idea to check normality of the response variable , since they are related.

First, we create a histogram of all `response` observations, and have a look at the `predictor1` values as well:

```r
ggplot(data = dataset, aes(x = response, y = after_stat(density))) +
  geom_histogram(fill = "cornflowerblue", bins = 15, col = "black") +
  ggplot(data = dataset, aes(x = predictor1, y = after_stat(density))) +
  geom_histogram(fill = "cornflowerblue", bins = 15, col = "black")
```



What we're looking for here is a roughly symmetric distributions with a single peak.

Outlier checking

This is a also very poorly understood assumption. We want a model represent the bulk of the data. We don't want it biased towards 1 or 2 outlying influential points. Just like checking the normality assumption we can only test this for sure once we have fit a model. However, it is always worth looking at all our data to see if there are any outliers we might need to deal with. The best way to do this is via histograms, re-using the one created above.

Step 2: Model fitting

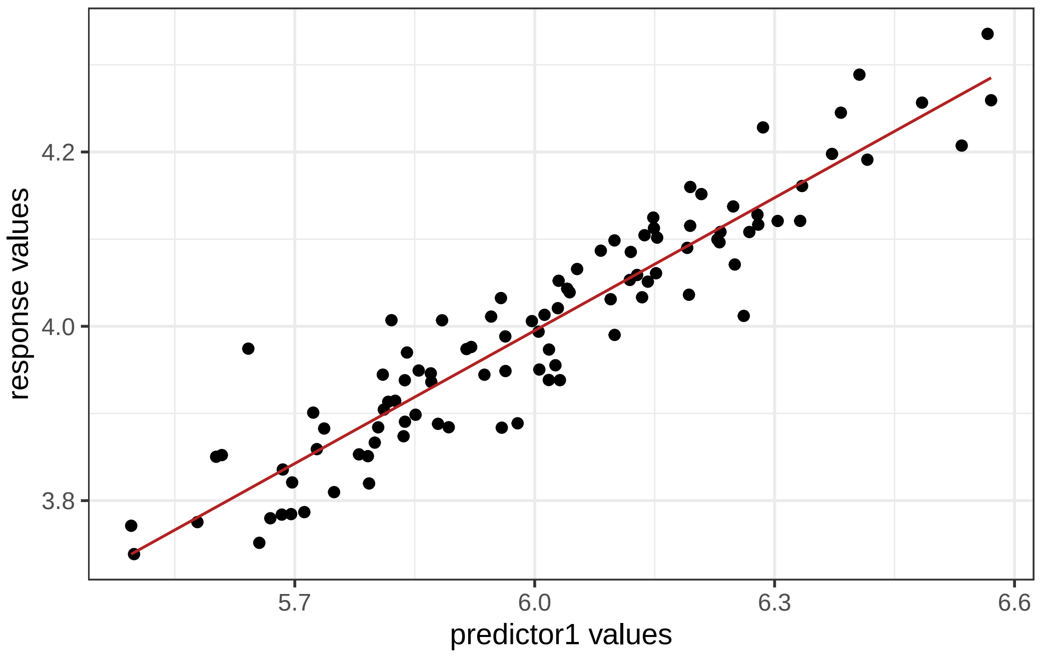
After a sucessful Exploratory Data Analysis, we use `lm()` to fit a linear model.

```r
model <- lm(response ~ predictor1, data = dataset)
```

Step 3: Model diagnostics

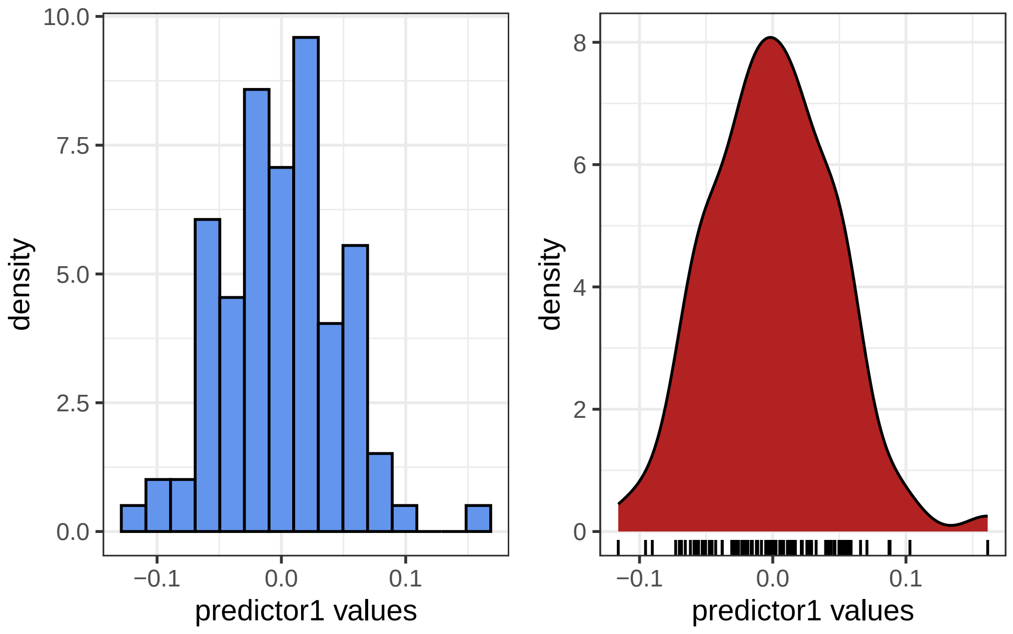
In order to gauge the model fit, we first visually display the residuals. Residuals are variability in our response that cannot be explained by the model. Visually, they appear as the distance between the fitted line and the observed datapoints. Have a look:

```r
ggplot(data = dataset, aes(x = predictor1, y = response)) +
  geom_point() +
  geom_line(data = NULL, aes(x = predictor1, y = fitted(model)), col = "firebrick") +
  labs(x = "predictor1 values", y = "response values")
```



We are looking for a few things here: Firstly, that the deviation from the line is roughly equal across the spread of `predictor1`. Secondly, that the residuals are normally distributed. We can also display the deviations from the line in a histogram including a kernel density graph, like so:

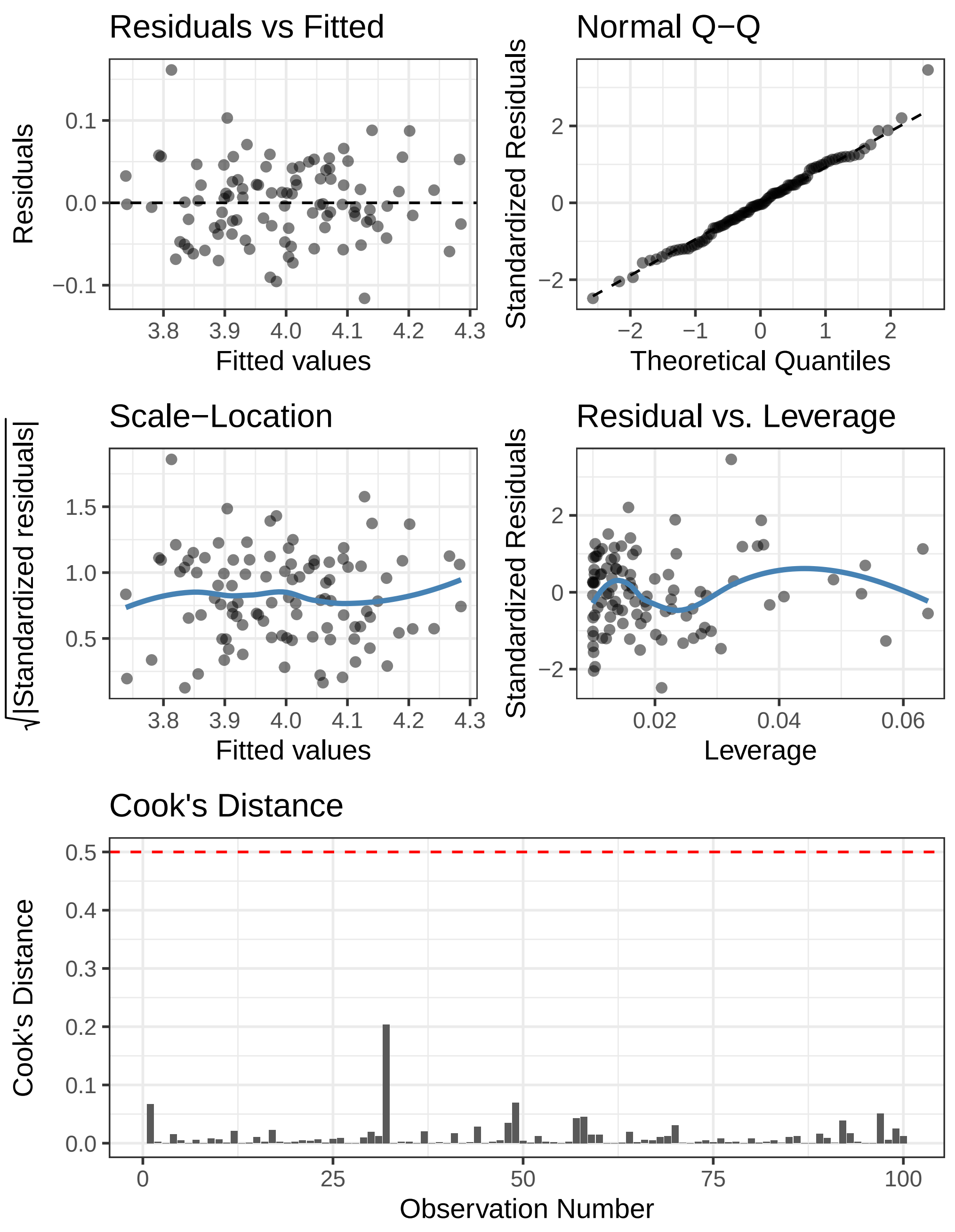
```r
model_res <- residuals(model)
ggplot(data = NULL, aes(x = model_res, y = after_stat(density))) +
  geom_histogram(fill = "cornflowerblue", bins = 15, col = "black") +
  labs(x = "predictor1 values", y = "density") +
  ggplot(data = NULL, aes(x = model_res, y = after_stat(density))) +
  geom_density(fill = "firebrick") +
  geom_rug(aes(y = NULL)) +
  labs(x = "predictor1 values", y = "density")
```



Using a histogram it is often easier to spot irregularities. In this case, using the graph above and below we can confirm that the residuals are normally and equally distributed across the line.

Other diagnostic plots can be produced with `plot(model_object_name)`. The `gglm` package produces the same plots but using the graphing package `ggplot2`, the plots of which look a little more up-to-date:

```
gglm(model, theme = theme_bw()) +
  ggplot(data = model) +
  stat_cooks_obs() +
  geom_hline(yintercept = 0.5, linetype = "dashed", col = "red") +
  plot_layout(nrow = 2, heights = c(2, 1))
```



The following plots are produced above:

1. Residuals vs Fitted: This plot displays the model residuals vs the fitted values. We are looking to confirm a linear relationships between predictor variables and the outcome variable here.
2. Normal Q-Q plot: This plot displays empirical normal distribution quantiles versus the theoretical ones. This plot seeks to confirm the normal distribution of residuals, as we

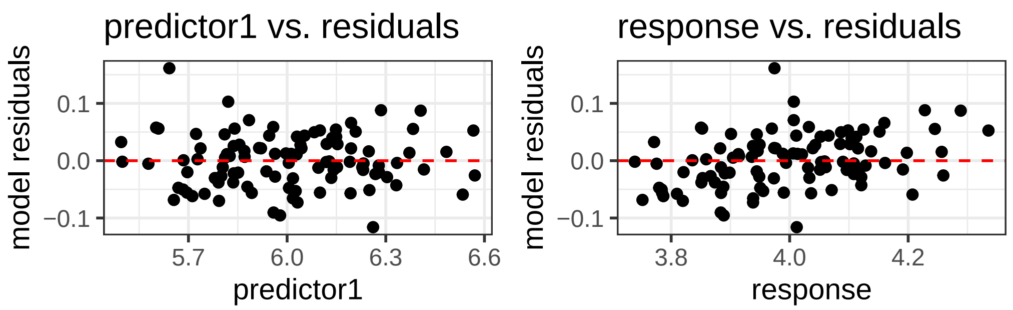
did before with the histogram.
3. Scale-Location: This plot shows whether residuals are equally spread along the range of predictors. A more or less horizontal line is what we're looking for here.
4. Residuals vs Leverage: In this plot, we're identifying influental observations. If one observation is unreasonably far to the right, it indicates an influental outlier.
5. Cook's distance plot: This plot shows Cook's distance numbers for each observation. R uses a cutoff of 0.5, so observations above that cut-off could indicate a problem.

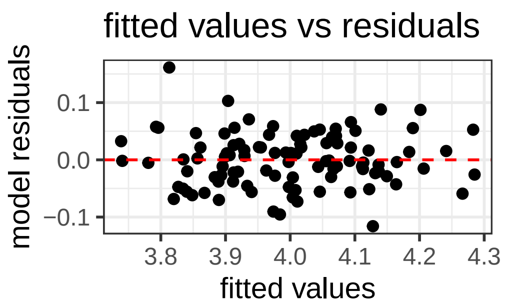
All of these plots look ideal.

Step 4: Goodness of fit

In this part, we want to check the goodness-of-fit of the model. Above, we already displayed the fitted values vs the residuals of the model. It also makes sense to compare the values of each predictor (explanatory variable), as well as the response with the residuals, to make sure the linearity assumption holds for each variable that we're predicting with. In our case, we only have one predictor variable.

```r
# Plot 1
ggplot(dataset, aes(x = predictor1, y = residuals(model))) +
  geom_point() +
  geom_hline(yintercept = 0, col = "red", linetype = "dashed") +
  labs(x = "predictor1", y = "model residuals", title = "predictor1 vs. residuals") +
# Plot 2
  ggplot(dataset, aes(x = response, y = residuals(model))) +
  geom_point() +
  geom_hline(yintercept = 0, col = "red", linetype = "dashed") +
  labs(x = "response", y = "model residuals", title = "response vs. residuals") +
# Plot 3
  ggplot(dataset, aes(x = predict(model), y = residuals(model))) +
  geom_point() +
  geom_hline(yintercept = 0, col = "red", linetype = "dashed") +
  labs(x = "fitted values", y = "model residuals", title = "fitted values vs residuals") +
  plot_layout(ncol = 2)
```

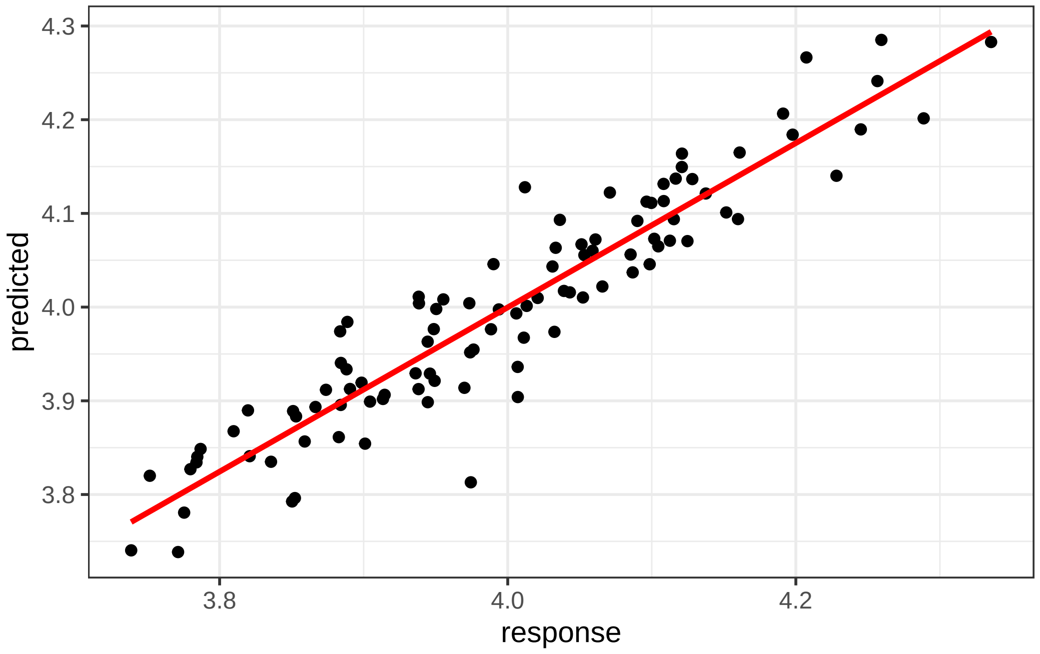




As we can see, the linearity assumption holds nicely. Let's also check the response vs predicted plot:

```
ggplot(dataset, aes(x = response, y = predict(model))) +
  geom_point() +
  geom_smooth(method = "lm", se = FALSE, col = "red") +
  labs(y = "predicted")
```

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



This plot is a good visual representation of model fit. If the response is being exactly predicted than we expect it to fall along the 1:1 line.

Now, we look at the summary output of our linear model:

```
summary(model)
```

```
Call:
lm(formula = response ~ predictor1, data = dataset)

Residuals:
      Min        1Q    Median        3Q       Max 
-0.115978 -0.030085 -0.001874  0.028932  0.161342 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept)  0.94554    0.11584   8.162 1.14e-12 ***
predictor1   0.50825    0.01927  26.370  < 2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.04691 on 98 degrees of freedom
Multiple R-squared:  0.8765,    Adjusted R-squared:  0.8752 
```

```
F-statistic: 695.4 on 1 and 98 DF,  p-value: < 2.2e-16
```

A model summary output has three parts:

1. Residual statistics (empirical quantiles)
2. Coefficient statistics
3. Model statistics

We already looked at the residuals a lot, so we're starting with the second section here. Each predictor and the intercept has one line in the table, which holds information on both the coefficient estimate and a significance test. A p value smaller than 0.05 indicates that we see a significant partial correlation between that predictor and the response variable. In this case, we have one predictor, the coefficient of which is significantly different from 0. It's also a good idea to look at the confidence interval of coefficients:

```
confint(model)
```

```
                2.5 %    97.5 %
(Intercept) 0.7156560 1.1754214
predictor1  0.4700031 0.5465012
```

In the third section of the summary output, we can see the R^2, which ranges from 0 to 1 and gives an indication on much variance of the response variable is explained by the predictor variables. The last line is important, as it tests for significance of the entire model.

Step 5: Interpret model parameters and reach conclusion

In this case, a p value of less than 0.05 indicates that the model is significantly different from 0 (no model). We can therefore conclude that the model is significant. Refer to the presentation here for more detail.

Step 6: Report overall conclusion

To interpret the paramters, we use the fitted regression coefficient. In our case, using the model table above, we can say (in this case we use "fake" predictor and response meanings):

> There is strong evidence to show that predictor1 influences response (p<2e-16), with each 1 unit of increase in predictor1 adding between 0.47-0.54 units of response (95% CI). This effect on weight has been estimated very accurately [as 95% CI is quite narrow].
>
> The model is a good fit to the data with an $R^2 = 88\%$. There were no outliers or unexplained structure. The error was normal.

Analysis of Variance (ANOVA)

In this section, we have a `data.frame` object with two variables:

- `treatment`: The treatment variable (binary)
- `response`: the response variable (metric)

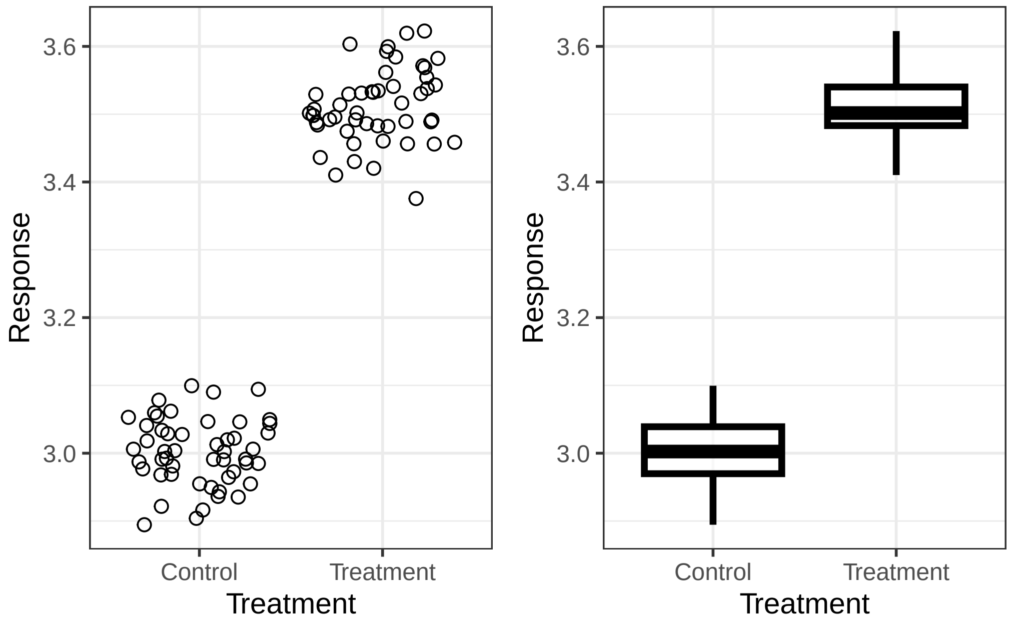
```r
set.seed(171975)
variance <- rnorm(100, 0, 0.05)
b0.control <- 3
b1.treatment <- 0.5

data1 <- tibble(
  treatment = factor(c(rep("Control", 50), rep("Treatment", 50))),
  response = b0.control + b1.treatment*ifelse(treatment=="Treatment", 1, 0) + variance
)
```

We assume that the response variable is dependent on the treatment. Due to the binary nature of `treatment`, we are fitting an Analysis of Variance (a subtype of linear model).

Step 1: Pick suitable model via EDA

First, let's visually display both variables using boxplots (as we see in the powerpoint):

```r
ggplot(data1, aes(x = treatment, y = response)) +
  # geom_violin(alpha=0.4, position = position_dodge(width = .75),size=1,color="black") +
  geom_point(
    shape = 21,
    size = 2,
    position = position_jitter(),
    color = "black",
    alpha = 1
  ) +
  labs(x = "Treatment", y = "Response") +
  ggplot(data1, aes(x = treatment, y = response)) +
  # geom_violin(alpha=0.4, position = position_dodge(width = .75),size=1,color="black") +
  geom_boxplot(
    notch = FALSE,
    outlier.size = -1,
    color = "black",
    lwd = 1.2,
    alpha = 0.7
  ) +
  labs(x = "Treatment", y = "Response")
```



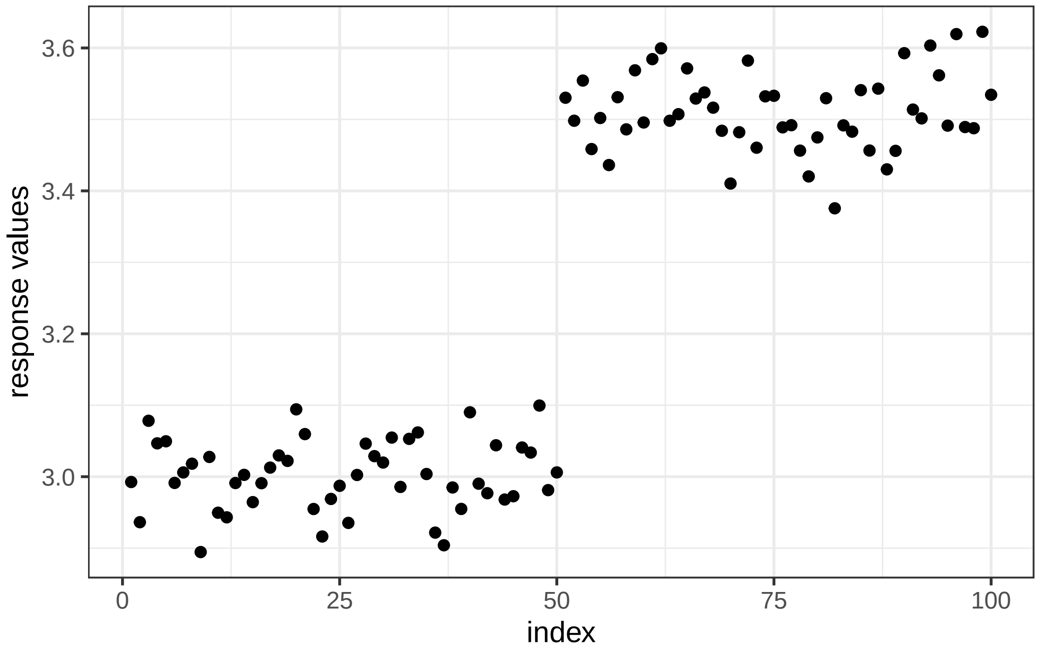
As we can see above, all observations are clearly separated, indicating a strong treatment effect.

Note: The "jitter" creates a nice visual scatter, but is purely for exploration. It should be removed in the final publication.

Check for assumptions

First, we check for independence using a serial plot:

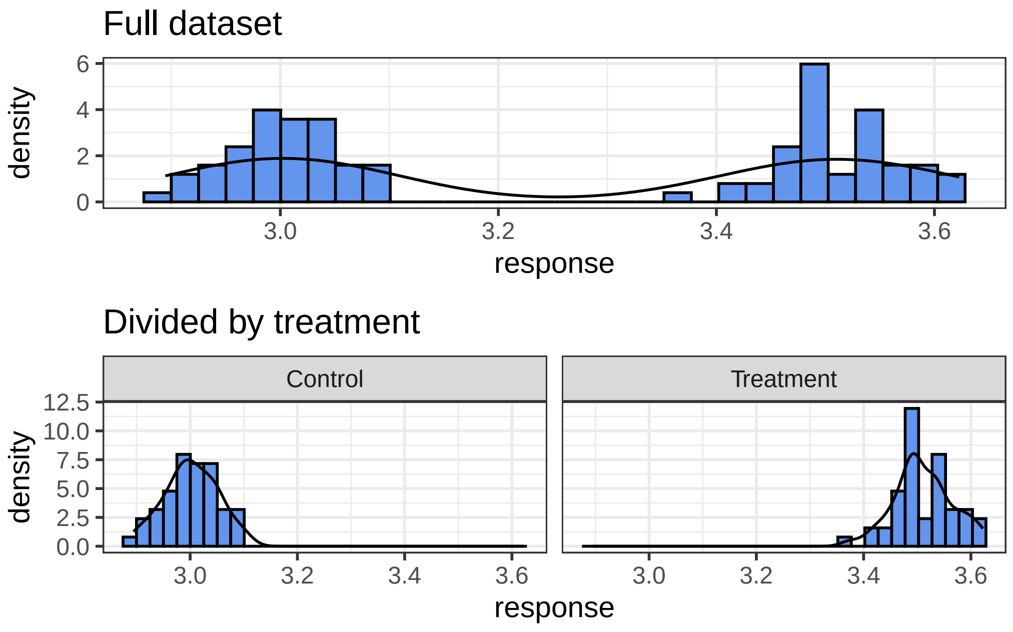
```
ggplot(data = data1, aes(
  x = seq_along(response),
  y = response
)) +
  geom_point() +
  labs(x = "index", y = "response values")
```



We see the separation of treatment values here, but within those there doesn't appear to be any (auto) correlation. Let's check normality next:

```
ggplot(data1, aes(
  x = response,
  y = after_stat(density)
)) +
  geom_histogram(fill = "cornflowerblue", col = "black") +
  geom_density(aes(fill = NULL)) +
  labs(title = "Full dataset") +
  ggplot(data1, aes(
  x = response,
  y = after_stat(density)
)) +
  facet_wrap(~ treatment) +
  geom_histogram(fill = "cornflowerblue", col = "black") +
  geom_density(aes(fill = NULL)) +
  labs(title = "Divided by treatment") +
  plot_layout(ncol = 1)
```

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



We can see clear separation here as well, otherwise normality. If we only looked in the above plot, we would have assumed non-normality. This is a great example of how we shouldn't worry about violations of normality in the dependent variable too much, since the residuals have to be normally distributed, not the outcome.

Outliers

The combined data (as seen above) exhibits a clear bimodal distribution and deviates significantly from normality. Thus, we need to address whether this poses a problem. However, the error distribution should be normal, not the response, and upon examination, the errors around the mean of each treatment appear to be approximately normal.

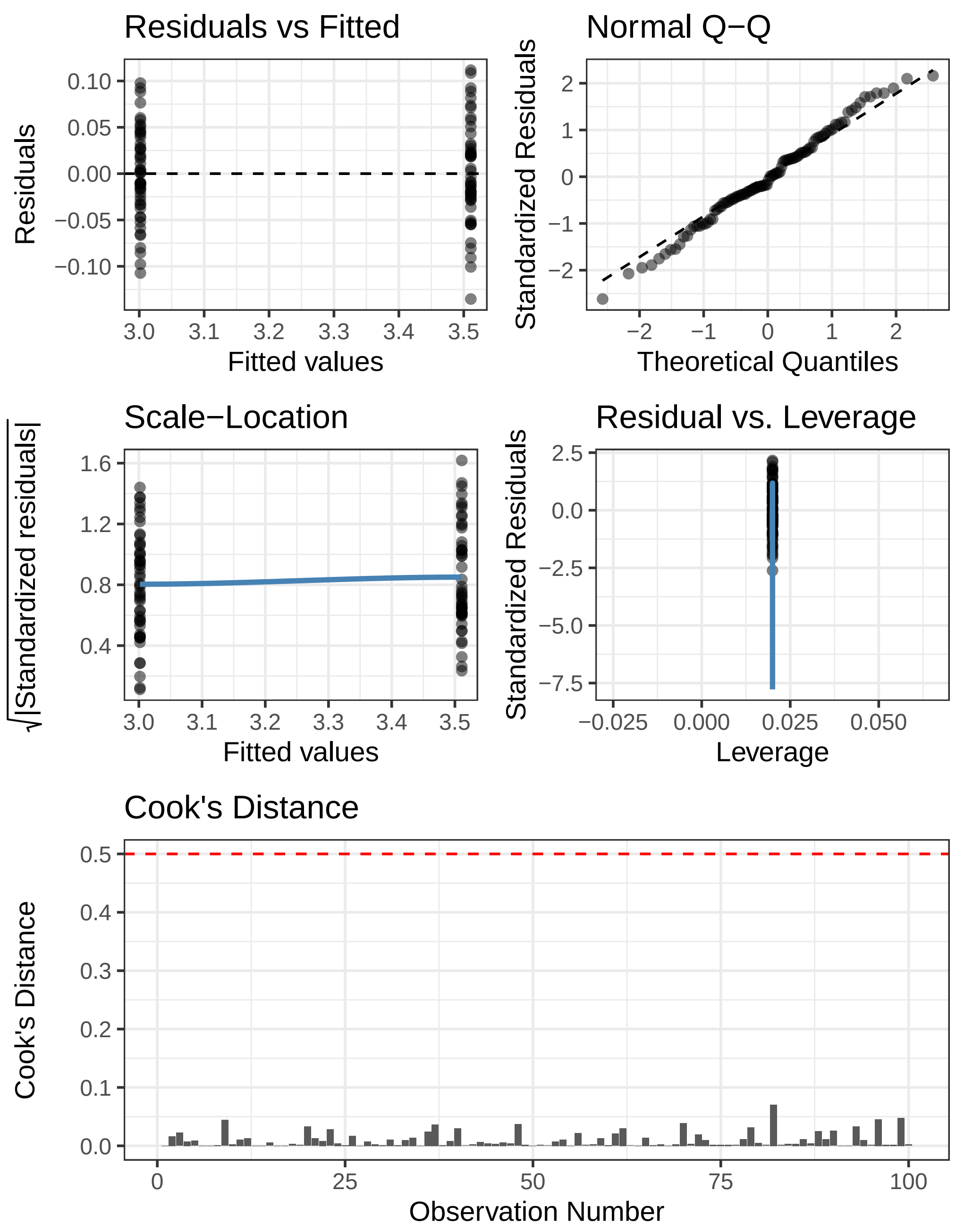
Step 2: Fit model

There are two ways to fit an ANOVA in R. You can either use the `lm()` function as above, or the `aov()` function. In this case, I'll use the `aov()` function, mainly because I prefer the summary output.

```r
anova_model <- aov(response ~ treatment, data = data1)
```

Step 3: Model diagnostics

```r
gglm(anova_model, theme = theme_bw()) +
  ggplot(data = anova_model) +
  stat_cooks_obs() +
  geom_hline(yintercept = 0.5, linetype = "dashed", col = "red") +
  plot_layout(nrow = 2, heights = c(2, 1))
```

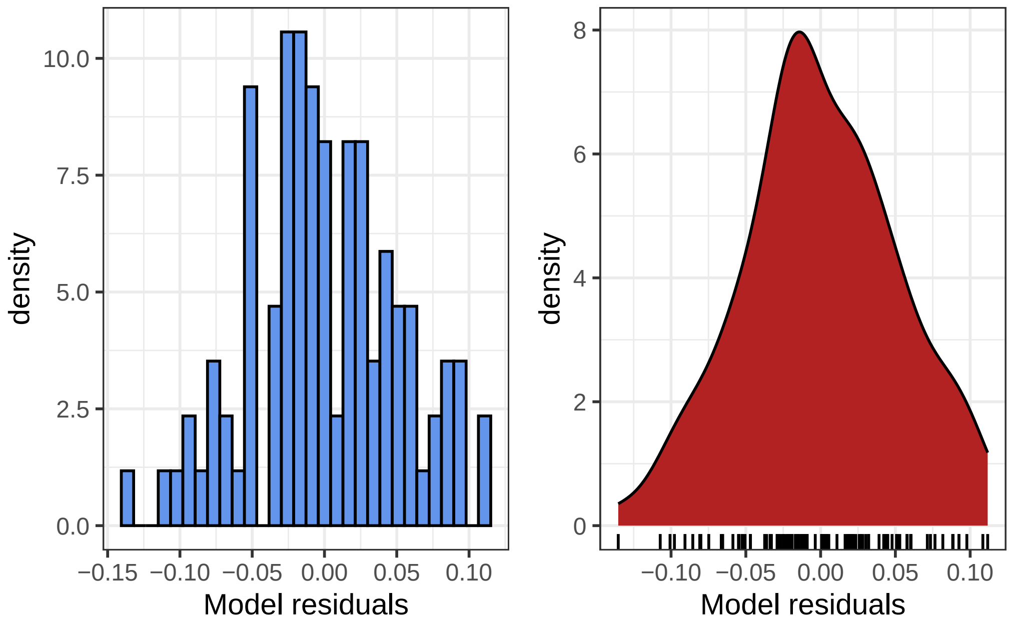


The model diagnostics plot is less informative than with the linear model, due to the binary nature of the `treatment` variable, but we can still observe interesting trends, like with the QQ plot for example.

Let's investigate the residuals a bit more:

```r
ggplot(data1, aes(
  x = residuals(anova_model),
  y = after_stat(density)
)) +
  labs(x = "Model residuals") +
  geom_histogram(fill = "cornflowerblue", col = "black") +
  ggplot(data1, aes(
  x = residuals(anova_model),
  y = after_stat(density)
)) +
  geom_density(fill = "firebrick") +
  geom_rug(aes(y = NULL)) +
  labs(x = "Model residuals")
```

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

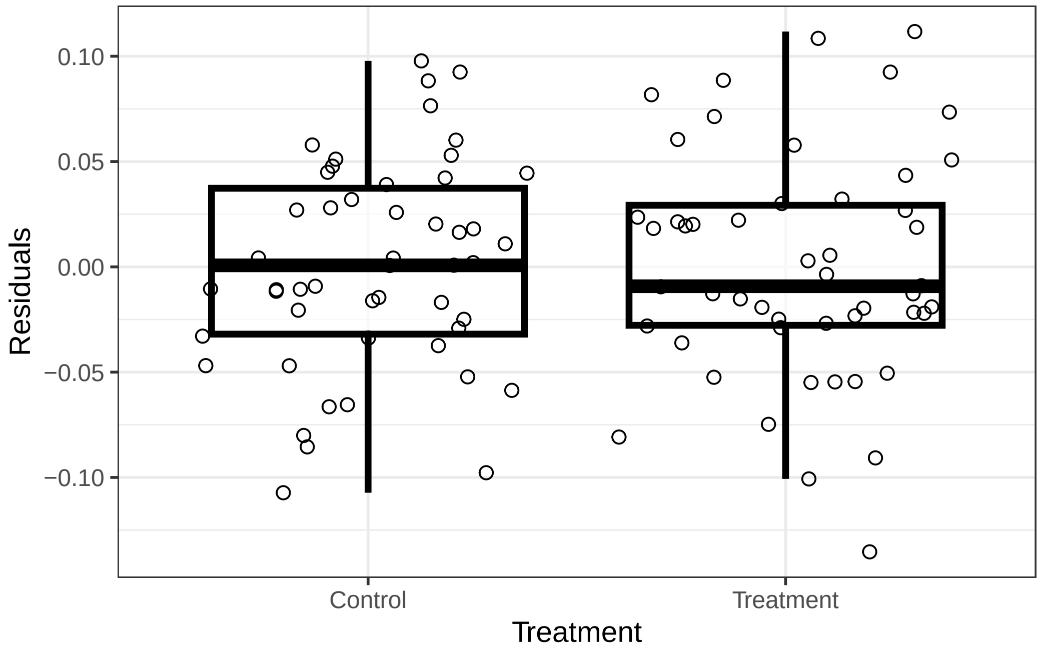


This time, we cannot observe large differences betwen the residuals of the both `treatment` effects, which indicates that the differences are nicely taken into account.

Step 4: Goodness of Fit

Let's have another look at the Residuals vs Fitted plot:

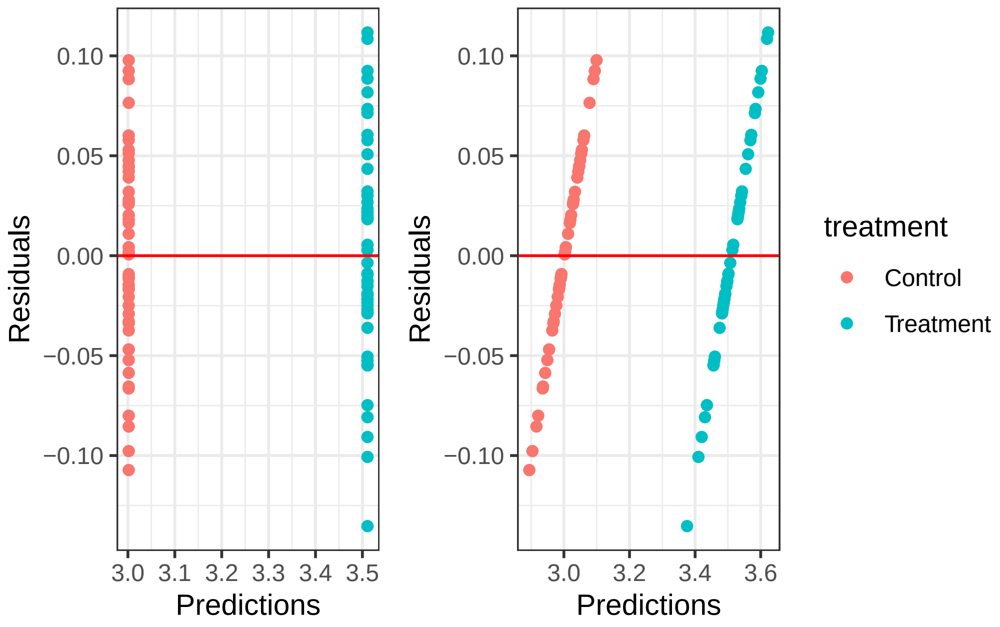
```r
ggplot(data1, aes(x = treatment, y = residuals(anova_model))) +
  # geom_violin(alpha=0.4, position = position_dodge(width = .75),size=1,color="black") +
  geom_boxplot(
    notch = FALSE,
    outlier.size = -1,
    color = "black",
    lwd = 1.2,
    alpha = 0.7
  ) +
  geom_point(
    shape = 21,
    size = 2,
    position = position_jitter(),
    color = "black",
    alpha = 1
  ) +
  labs(x = "Treatment", y = "Residuals")
```



This time we can again see that the residuals are nicely spread around 0 for both treatment and control.

Another great plot is Fitted vs Residuals as as well as Response vs Residuals :

```r
ggplot(data1, aes(
  x = predict(anova_model),
  y = residuals(anova_model),
  col = treatment)) +
  geom_point() +
  labs(x = "Predictions", y = "Residuals") +
  geom_hline(yintercept = 0, col = "red") +
  theme(legend.position = "none") +
  # Plot 2
  ggplot(data1, aes(
  x = response,
  y = residuals(anova_model),
  col = treatment)) +
  geom_point() +
  labs(x = "Predictions", y = "Residuals") +
  geom_hline(yintercept = 0, col = "red")
```

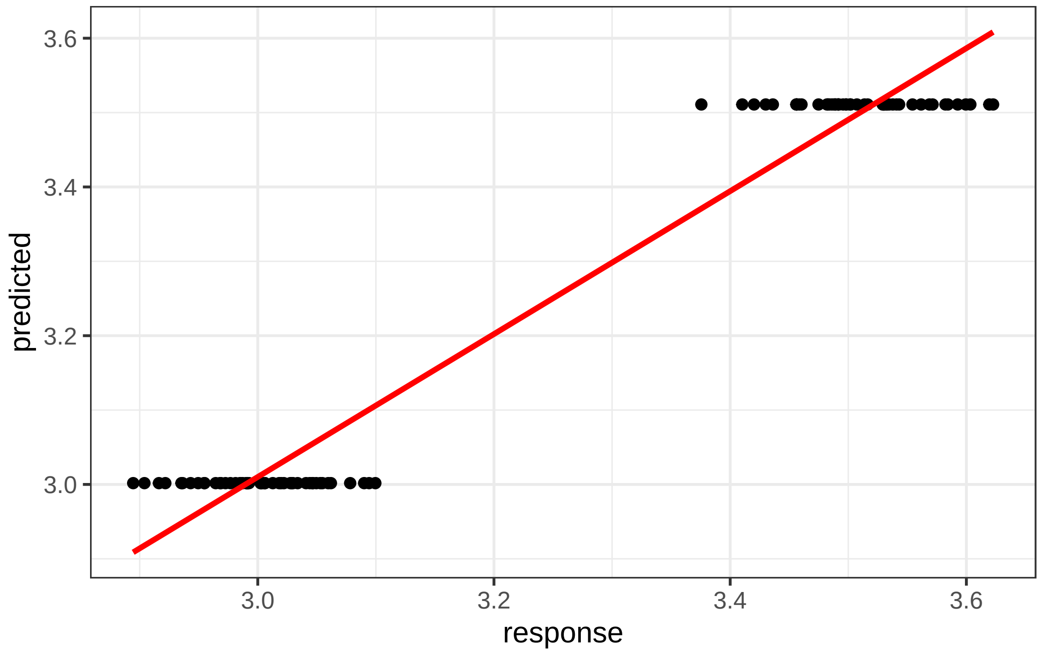


We expect the 'lines' of data rather than a random 'cloud' of data which we saw in the regression. This is because rather than a range of predictions for each different value of the predictor we only get 1 prediction for control and another for treatment, hence 2 vertical lines in the upper chart.

Let's also check the response vs. predictions:

```r
ggplot(data1, aes(x = response, y = predict(anova_model))) +
  geom_point() +
  geom_smooth(method = "lm", se = FALSE, col = "red") +
  labs(y = "predicted")
```

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



To check the goodness-of-fit, we have a look at the model summary output:

```r
summary(anova_model)
```

```
            Df Sum Sq Mean Sq F value Pr(>F)
treatment    1  6.481   6.481    2400 <2e-16 ***
Residuals   98  0.265   0.003
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

In this case, we purely observe the significance of our treatment effect, which is seen in the first line of the Anova-Output. A p value of less than 0.05 indicates a significant difference between the categories of the `treatment` variable (of which there are only two).

Step 5: Interpret Model Parameters and reach a conclusion

In order to find out how large the difference is, we need to convert the anova object back to a linear model like so:

```r
coef(lm(anova_model))
```

```
       (Intercept) treatmentTreatment 
         3.0017546          0.5091524 
```

```r
confint(lm(anova_model))
```

```
                       2.5 %    97.5 %
(Intercept)        2.9871714 3.0163378
treatmentTreatment 0.4885287 0.5297761
```

We can now see that the average difference of `response` between the treatment and control effect is 0.509. Since our effect is significantly different from 0, we can now conclude that our treatment is successful.

Refer to the presentation here for more detail.

Step 6: Report overall conclusion

You can use the following sentence (we again assume a "fake" meaning):

> There is strong evidence to show that the treatment influences response (p<2e-16). It increases the response by between 0.49-0.53 units (95% CI), from an average of approximately 3 (95% CI=2.98-3.01). This effect on the response has been estimated very accurately [as 95% CI is quite narrow].
>
> The model is a good fit to the data with an R2=97%. There were no outliers or unexplained structure. The error was normal"

Analysis of Covariance (ANCOVA)

In ANCOVA model situations, we are interested in a treatment effect like in ANOVA, but we also want to account for other covariates. This could, for example, be the age of clinical trial candidates.

Let's create some data first (I collapsed this element because it's a lot of code):

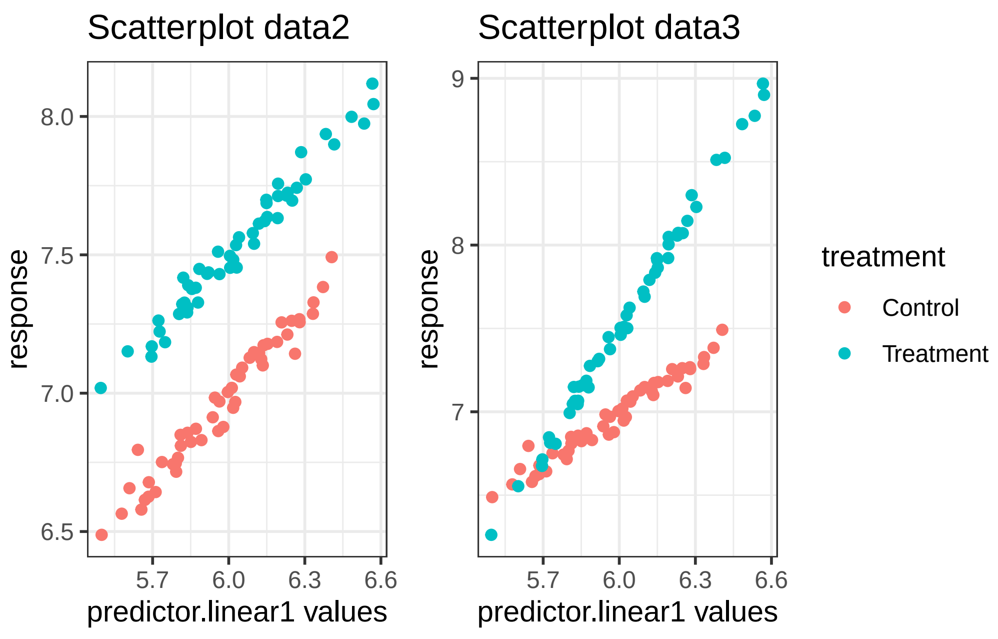
```r
set.seed(171974)
predictor.linear1 <- rnorm(100, 6, 0.25)
variance <- rnorm(100, 0, 0.05)

treatment <- factor(c(rep("Control", 50), rep("Treatment", 50)))

# Model 1) if no difference between groups i.e. lines are parrallel
b0.control <- 1
b0.treatment <- 0.5
b1.control <- 1
b1.treatment <- 0

data2 <- data.frame()[1:100, ]
data2$treatment <- treatment
data2$predictor.linear1 <- predictor.linear1
data2$response <- b0.control +
    b1.control*predictor.linear1 +
    b0.treatment*ifelse(data2$treatment=="Treatment", 1, 0) +
    b1.treatment*ifelse(data2$treatment=="Treatment", 1, 0)*predictor.linear1 +
    variance
row.names(data2) <- NULL


# Model 2) if  difference between groups i.e. lines aren't parallel
b0.control <- 1
b0.treatment <- -8.5
b1.control <- 1
b1.treatment <- 1.5

data3 <- data.frame()[1:100, ]
data3$treatment <- treatment
data3$predictor.linear1 <- predictor.linear1
data3$response <- b0.control +
    b1.control*predictor.linear1 +
    b0.treatment*ifelse(data3$treatment=="Treatment", 1, 0) +
    b1.treatment*ifelse(data3$treatment=="Treatment", 1, 0)*predictor.linear1 +
    variance
row.names(data3) <- NULL
```

We now have three covariates in a `data.frame` object called `data2`, and the same in a different object called `data3`:

- `treatment`: A categorical variable depicting the treatment/control groups
- `predictor.linear1`: Our numeric predictor
- `response`: The numeric response

Let's look at our datasets graphically:

```
d2graph <- ggplot(data2, aes(x = predictor.linear1, y = response, col = treatment)) +
  geom_point() +
  guides(col = "none") +
  labs(x = "predictor.linear1 values", title = "Scatterplot data2")
d3graph <- ggplot(data3, aes(x = predictor.linear1, y = response, col = treatment)) +
  geom_point() +
  labs(x = "predictor.linear1 values", title = "Scatterplot data3")
d2graph + d3graph
```



We can now see the relationship between `predictor.linear1` and `response` in two different scenarios (`data2` vs `data3`). The difference is that on the left graph, the differently coloured points have a parallel increase, whereas on the right graph the slopes of the increase between the points is also different. We will be further using `data3`, because it has a more complex (and interesting) model structure.

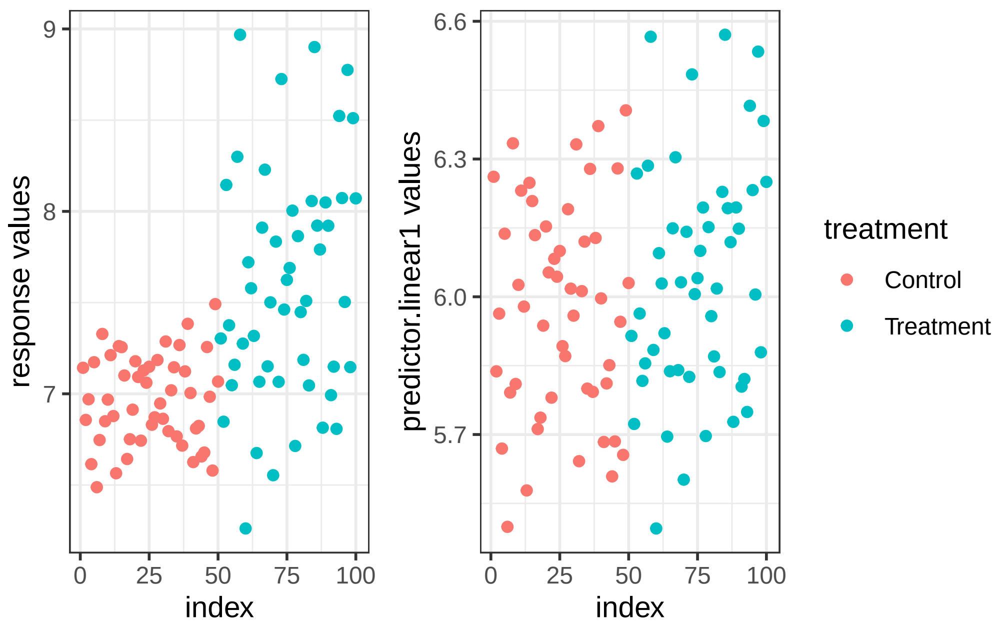
Step 1: Pick suitable model using EDA

From the (right) graph above, we can see that there is a binary treatment effect as well as a linear predictor called `predictor.linear1`. This is a classic ANCOVA scenario.

Model assumptions

First, let's check whether the response observations seem to be (auto) correlated using a serial plot:

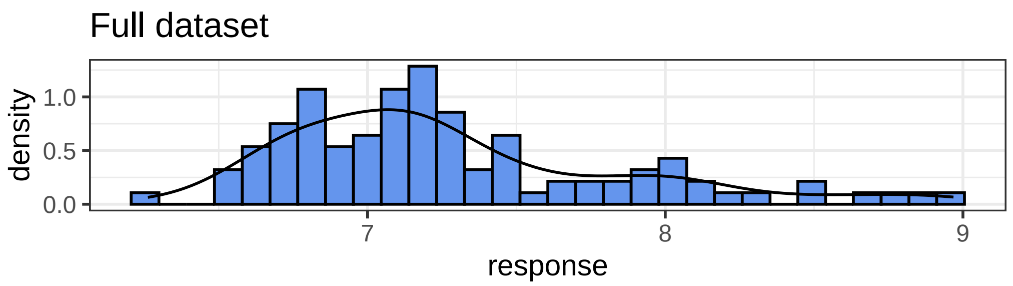
```r
ggplot(data = data3, aes(
  x = seq_along(response),
  y = response,
  col = treatment
)) +
  geom_point() +
  labs(x = "index", y = "response values") +
  ggplot(data = data3, aes(
  x = seq_along(response),
  y = predictor.linear1,
  col = treatment
)) +
  geom_point() +
  labs(x = "index", y = "predictor.linear1 values") +
  plot_layout(guides = "collect")
```

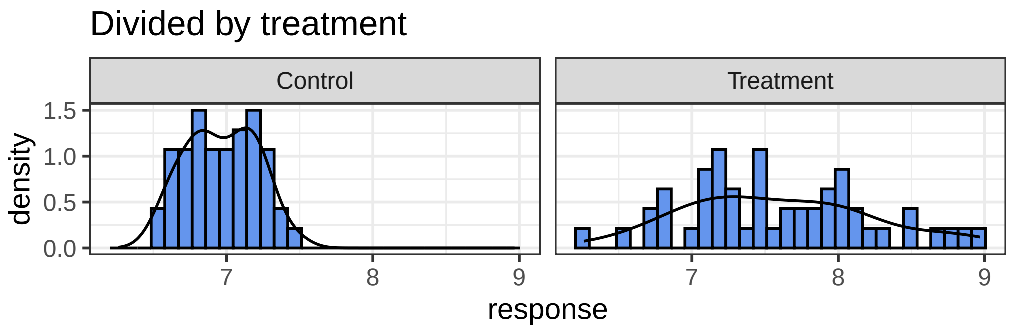


No pattern visible, except the one created by the treatment effect, which we will later account for in the model. Next, we check for normality and look for apparent outliers.

```
ggplot(data3, aes(
  x = response,
  y = after_stat(density)
)) +
  geom_histogram(fill = "cornflowerblue", col = "black") +
  geom_density(aes(fill = NULL)) +
  labs(title = "Full dataset") +
  ggplot(data3, aes(
  x = response,
  y = after_stat(density)
)) +
  facet_wrap(~ treatment) +
  geom_histogram(fill = "cornflowerblue", col = "black") +
  geom_density(aes(fill = NULL)) +
  labs(title = "Divided by treatment") +
  plot_layout(ncol = 1)
```

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





We can see that there are visible differences in the response variable broken up by the treatment effect, but this is not concerning, both histograms still appear mostly normal, just with differences in the variance and location of the distribution. This is a great example of how we shouldn't worry about violations of normality in the dependent variable too much, since the residuals have to be normally distributed, not the outcome.

Step 2: Fit model

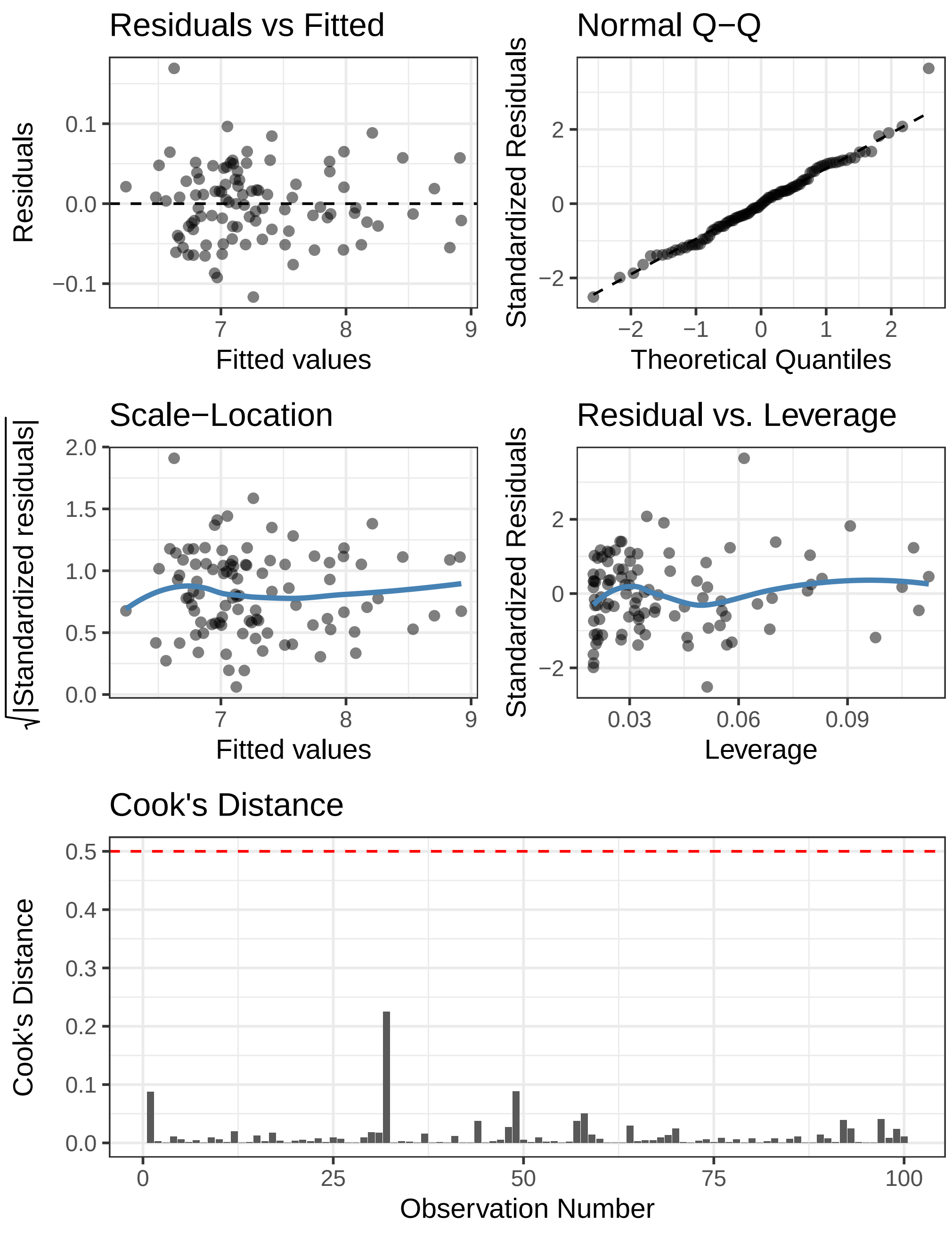
Let's fit the model next. Due to the different slopes in the predictor variable in different treatment effects, it is required that we fit an interaction between `treatment` and `predictor.linear1`:

```
ancova_model <- aov(
  response ~ treatment + predictor.linear1 +
  treatment * predictor.linear1,
  data = data3
)
```

Step 3: Model diagnostics

Our classic five diagnostic plots can be created first:

```r
gglm(ancova_model, theme = theme_bw()) +
  ggplot(data = ancova_model) +
  stat_cooks_obs() +
  geom_hline(yintercept = 0.5, linetype = "dashed", col = "red") +
  plot_layout(nrow = 2, heights = c(2, 1))
```

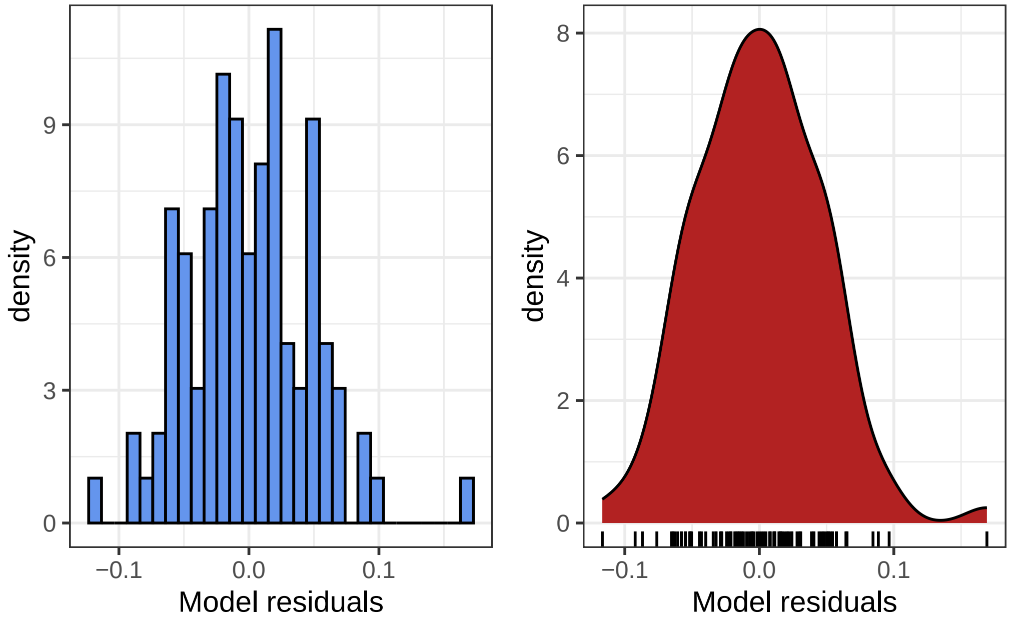


All of these plots look very well behaved. We can't detect any visible pattern in the plots, neither can we find any observations that are too influental.

Next, let's have a look at the residuals, and whether there is still a visible pattern:

```
ggplot(data3, aes(
  x = residuals(ancova_model),
  y = after_stat(density)
)) +
  labs(x = "Model residuals") +
  geom_histogram(col = "black", fill = "cornflowerblue") +
  ggplot(data3, aes(
  x = residuals(ancova_model),
  y = after_stat(density)
)) +
  geom_density(fill = "firebrick") +
  geom_rug(aes(y = NULL)) +
  labs(x = "Model residuals")
```

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



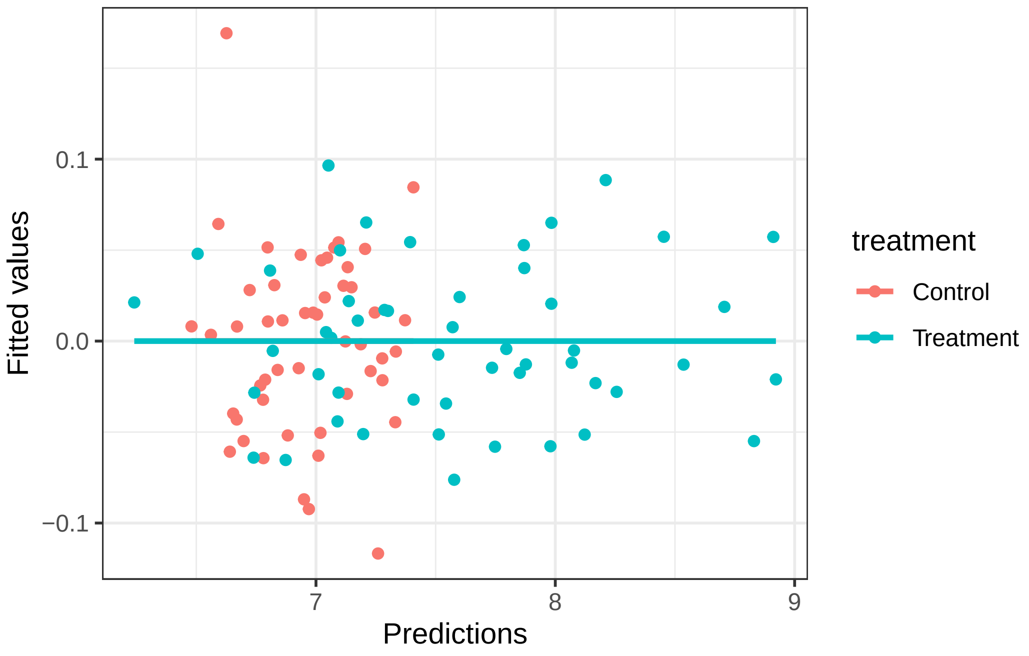
The residuals look very nicely behaved, no non-normality can be detected.

Step 4: Goodness-of-fit

Let's have another look at the Residuals vs Fitted plot, as well as Residuals vs response:

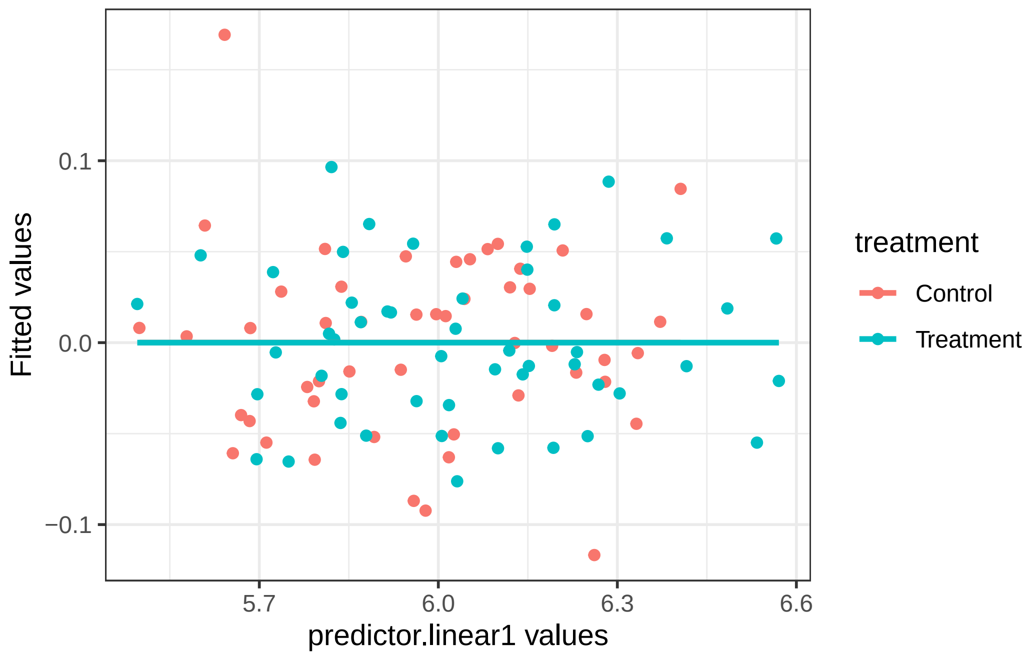
```r
ggplot(data3, aes(
  x = predict(ancova_model),
  y = residuals(ancova_model),
  col = treatment)) +
  geom_point() +
  geom_smooth(se = FALSE, method = "lm") +
  labs(x = "Predictions", y = "Fitted values")
```

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



```r
  ggplot(data3, aes(
  x = predictor.linear1,
  y = residuals(ancova_model),
  col = treatment)) +
  geom_point() +
  geom_smooth(se = FALSE, method = "lm") +
  labs(x = "predictor.linear1 values", y = "Fitted values")
```

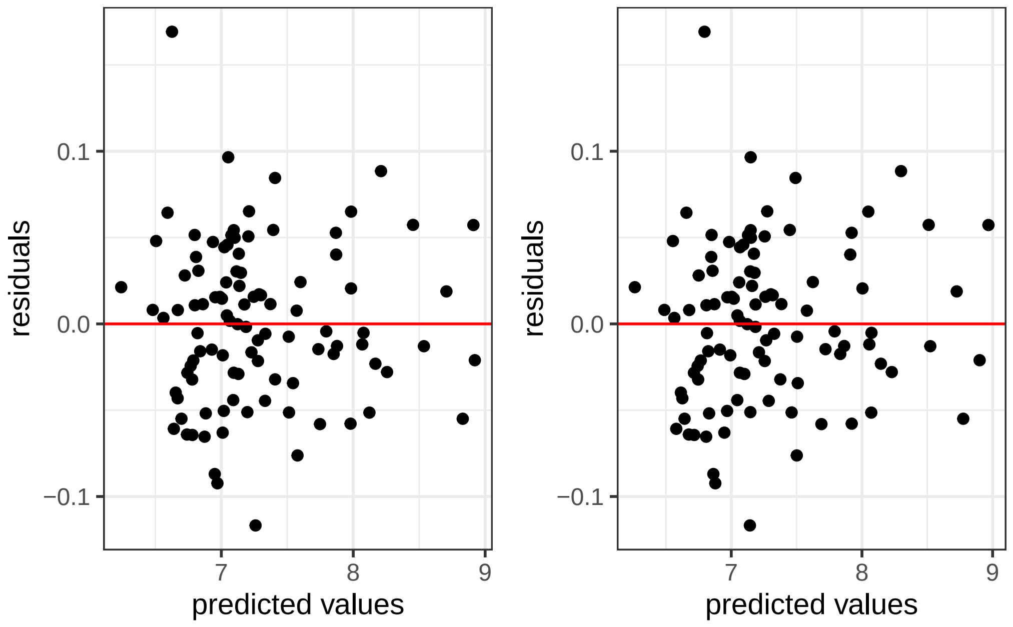
```
`geom_smooth()` using formula = 'y ~ x'
```



Even separated by the treatment effect, no pattern is apparent. Had we forgotten to fit an interaction effect, we would still see some remaining slopes or strange deviations from the line.

Let's dig deeper into the residuals:

```
ggplot(data3, aes(x = predict(ancova_model), y = residuals(ancova_model))) +
  geom_point() +
  geom_hline(yintercept = 0, col = "red") +
  labs(x = "predicted values", y = "residuals") +
  ggplot(data3, aes(x = response, y = residuals(ancova_model))) +
  geom_point() +
  geom_hline(yintercept = 0, col = "red") +
  labs(x = "predicted values", y = "residuals")
```

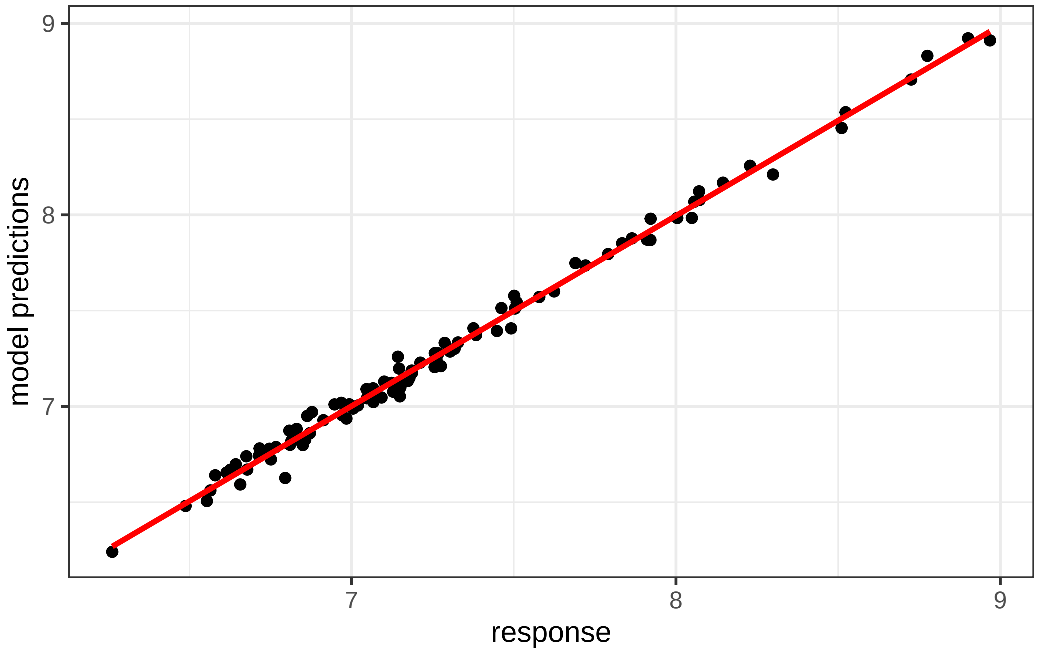


No evidence of outliers, or unexplained structure or non linearity.

We can also compare the predictions vs. the response:

```
ggplot(data3, aes(x = response, y = predict(ancova_model))) +
  geom_point() +
  geom_smooth(method = "lm", se = FALSE, col = "red") +
  labs(x = "response", y = "model predictions")
```

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



Let’s check the summary output next:

```
summary(ancova_model)
```

```
                             Df Sum Sq Mean Sq F value Pr(>F)    
treatment                     1  9.809   9.809    4412 <2e-16 ***
predictor.linear1             1 19.388  19.388    8721 <2e-16 ***
treatment:predictor.linear1   1  3.125   3.125    1406 <2e-16 ***
Residuals                    96  0.213   0.002                   
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

As we can see, all three fitted effects are significantly different from zero, indicating a successful treatment effect.

Step 5: Interpret Model Parameters and reach a conclusion

If we want to know how large our coefficiencts are, we need to convert the model back to a “linear model”:

```
coef(lm(ancova_model))
```

```
              (Intercept)                      treatmentTreatment
                0.8589649                              -8.3203390
        predictor.linear1 treatmentTreatment:predictor.linear1
                1.0222017                               1.4711692
```

```r
confint(lm(ancova_model))
```

```
                                          2.5 %    97.5 %
(Intercept)                           0.5150596  1.202870
treatmentTreatment                   -8.7882616 -7.852416
predictor.linear1                     0.9646711  1.079732
treatmentTreatment:predictor.linear1  1.3932757  1.549063
```

The treatment effect seems to be negative in this case, but it can only be interpreted jointly with the interaction. As such, while the treatment effect by itself is negative, the slope of it's interaction with predictor.linear1 is positive, so the values do rise above the control group quite quickly. This becomes apparent in the scatterplot shown in the first part of this section (EDA).

This does indicate that the treatment effect strongly interacts with the `predictor.linear1` variable.

Refer to the presentation here for more detail.

Step 6: Report overall conclusion

You can again use the following sentences (assuming a fake application):

> There is strong evidence to show that predictor.linear1 impacts the response (p<2e-16), with each unit of predictor.linear1 adding between 0.96-1.08 units of response (95% CI).
>
> There is strong evidence that the treatment have a positive effect on the impact of predictor.linear1 (p<2e-16), increasing its effect by between 1.39-1.55 (95% CI), for a total average effect of 2.5 units of response increase for each unit increase of predictor.linear1.
>
> This effect of predictor.linear1 on response has been estimated very accurately [as 95% CI is quite narrow].
>
> The model is a good fit to the data with an R2=99%. There were no outliers or unexplained structure. The error was normal

Linear Mixed Models

We use Mixed Models when we cannot assume that all observations are independently distributed, due to an error term structure that we need to take into account. This could be due to repeated measures where we take a measurement multiple times for each unit of observation (and therefore each unit's measurements are correlated) or a "nested structure", like students in a classroom.

Let's create some data first to illustrate the example:

```r
set.seed(171974)
variance <- rnorm(100, 0, 250) # as within person variance is 100

id <- rep(c(1:10), 2, each=5)
treatment <- factor(c(rep("Control", 50), rep("Treatment", 50)))

b0.control.average <- 4000
b0.control.0 <- rnorm(10, b0.control.average, 1500) # as range 1000-9000
b0.control <- rep(b0.control.0, 2, each=5)
b1.treatment <- 500

data6 <- data.frame()[1:100, ]
data6$id <- as.factor(id)
data6$treatment <- treatment
data6$response <- b0.control + b1.treatment*ifelse(data6$treatment=="Treatment", 1, 0) + vari
data6 <- data6[order(data6$id, data6$treatment),]
row.names(data6) <- NULL
data6$b0.control <- b0.control

writexl::write_xlsx(x = data6, path = "lm1_dataset_linear_mixed_models.xlsx")
```

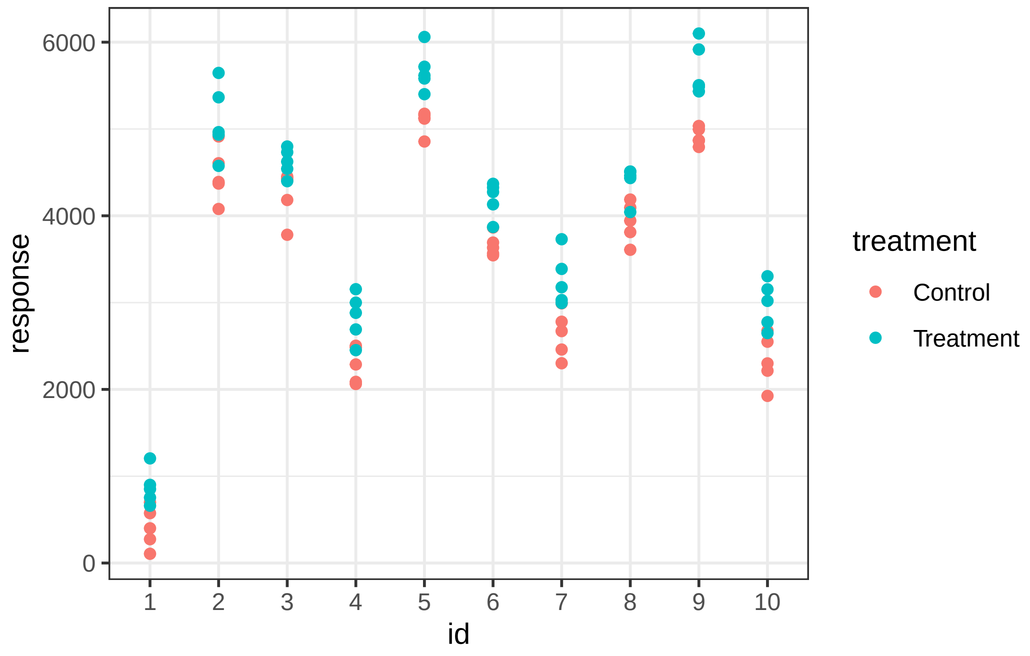
In our new `data.frame`, we now have a new variable `id`, which specifies the unit of observation. This is "repeated", as we take each unit of observation multiple times. Our variables are thus:

- `id`: ID of the unit of observation
- `b0.control`: This is our numeric predictor variable, which is unique for each unit of observation but not for each repeated measure.
- `treatment`: This contains information about the treatment/control groups
- `response`: Our target variable

Step 1: Pick a suitable model to fit to the data via EDA

Let's first graphically display the dataset:

```r
ggplot(data6, aes(
  x = id,
  y = response,
  col = treatment
  )) +
  geom_point() +
  labs(x = "id")
```

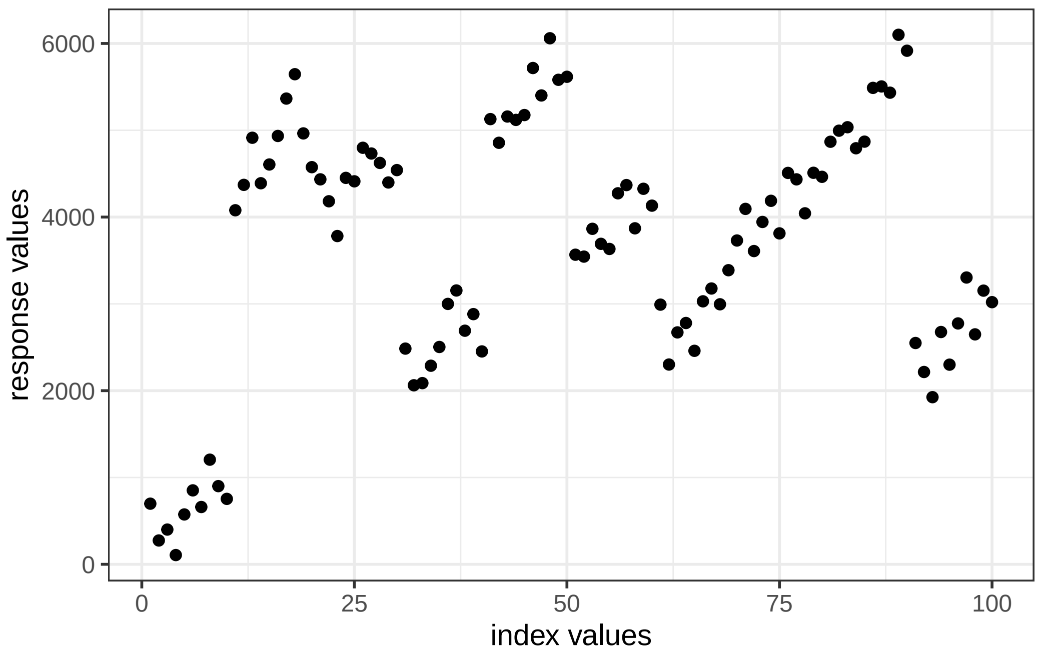


In this graph, we can both see a horizontal trend between `b0.control` and `response` as well as the multiple repetitions scattered vertically. This is a perfect mixed model scenario. We can see that the observations are correlated within the `id` variable.

Check assumptions

The first assumption we'll check is the we already know will be broken: independence. A serial plot helps to see this:

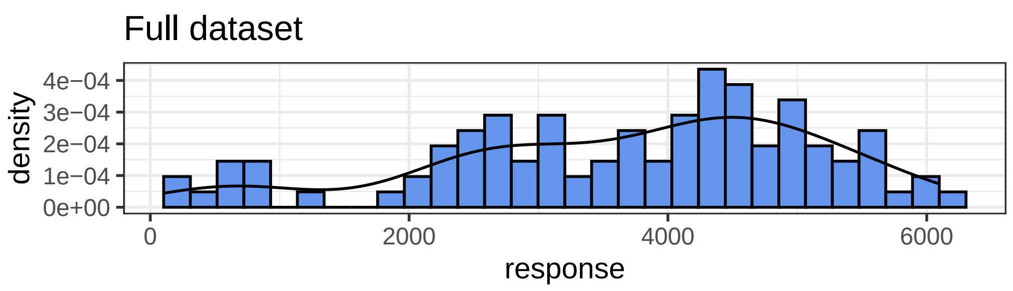
```r
ggplot(data = data6, aes(
  x = seq_along(response),
  y = response
)) +
  geom_point() +
  labs(x = "index values", y = "response values")
```

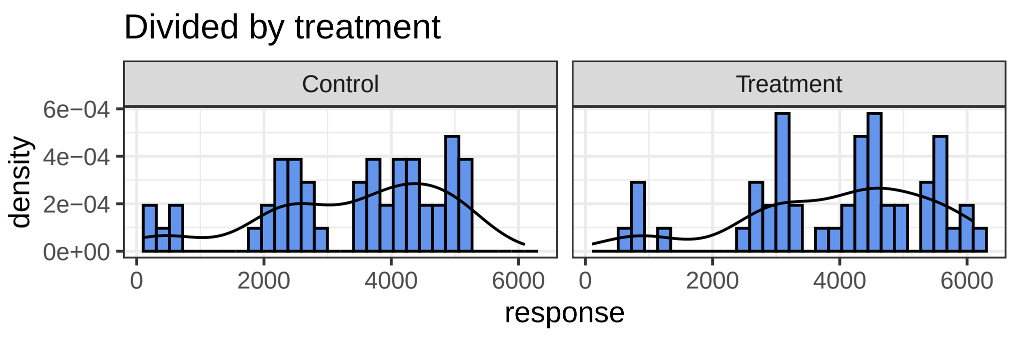


We can definitely see the repetition pattern here. Let's check normality next:

```
ggplot(data6, aes(
  x = response,
  y = after_stat(density)
)) +
  geom_histogram(fill = "cornflowerblue", col = "black") +
  geom_density(aes(fill = NULL)) +
  labs(title = "Full dataset") +
  ggplot(data6, aes(
  x = response,
  y = after_stat(density)
)) +
  facet_wrap(~ treatment) +
  geom_histogram(fill = "cornflowerblue", col = "black") +
  geom_density(aes(fill = NULL)) +
  labs(title = "Divided by treatment") +
  plot_layout(ncol = 1)
```

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





We see a slight right-skew in both treatment and control response groups. This is nothing to worry about yet, however, since the residuals need to be normally distributed, not the response. This is a great example of how we shouldn't worry about violations of normality in the dependent variable too much, since the residuals have to be normally distributed, not the outcome.

Step 2: Fit the Model

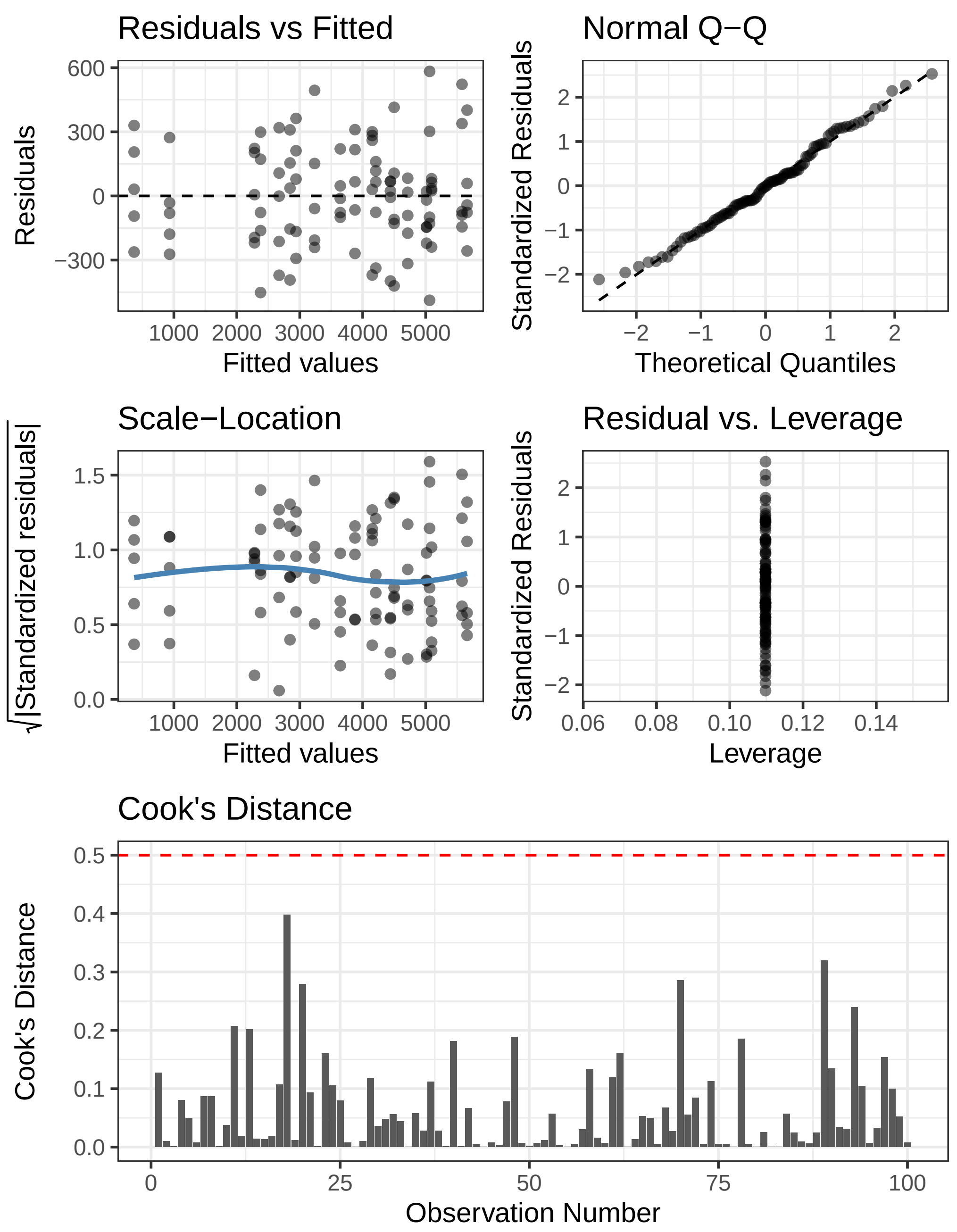
Now, we fit our mixed model. In general, mixed models have a slightly different notation as linear "unmixed" models, since we have to distinguish between "fixed" (uncorrelated) and "random" (correlated) effects. The variable which "nests" the observation is our random effect.

```r
mixed_model <- lmer(response ~ treatment + (1 | id), data = data6)
```

Step 3: Model diagnostics

Let's do our classic 5-graph model diagnostics first:

```r
gglm(mixed_model, theme = theme_bw()) +
  ggplot(data = mixed_model) +
  stat_cooks_obs() +
  geom_hline(yintercept = 0.5, linetype = "dashed", col = "red") +
  plot_layout(nrow = 2, heights = c(2, 1))
```

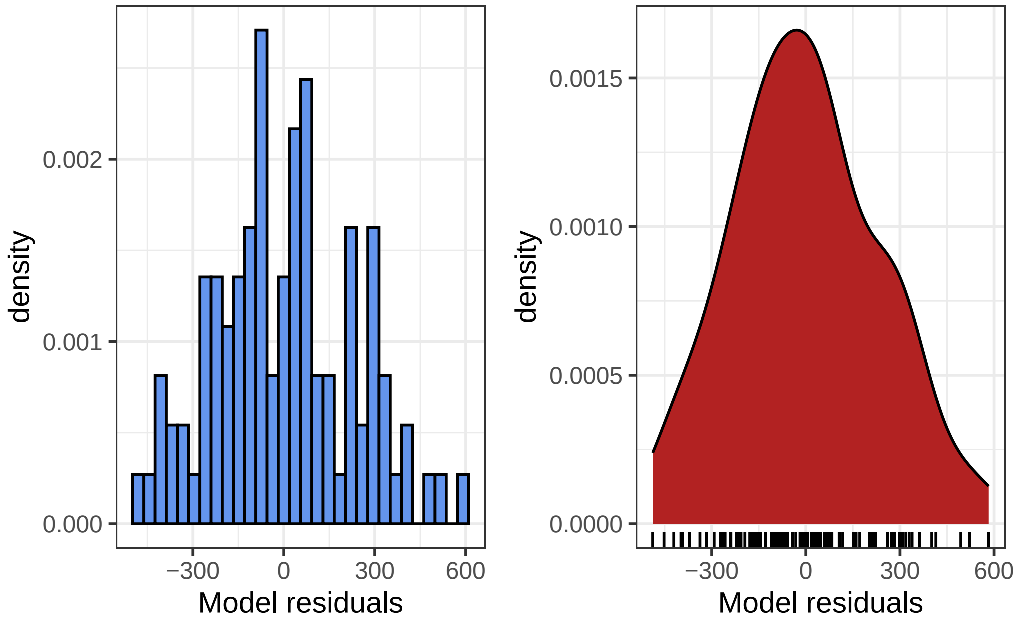


Our graphs look pretty good. There are no influental observations that we can see.

Let's have a closer look at the residuals, to see whether there is any pattern left:

```r
ggplot(data6, aes(
  x = residuals(mixed_model),
  y = after_stat(density)
)) +
  labs(x = "Model residuals") +
  geom_histogram(col = "black", fill = "cornflowerblue") +
  ggplot(data6, aes(
  x = residuals(mixed_model),
  y = after_stat(density)
)) +
  geom_density(fill = "firebrick") +
  geom_rug(aes(y = NULL)) +
  labs(x = "Model residuals")
```

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



The residuals seem to be greatly behaved, and pretty normal now (especially in comparison to the response graph from before).

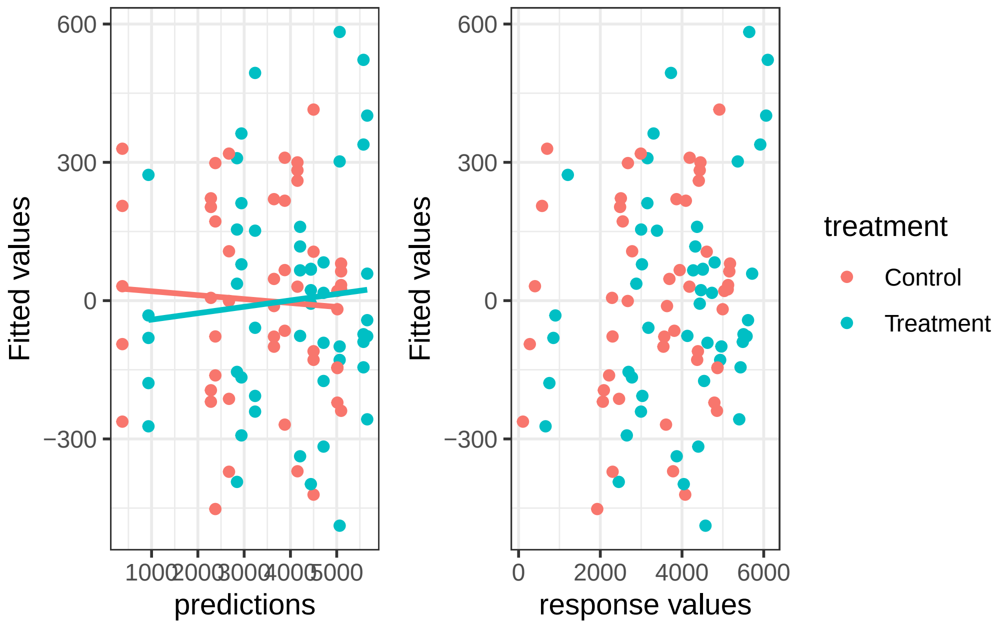
Step 4: Goodness of fit

Outliers and unexpected structure or nonlinearity

Let's have another look at the residuals vs fitted plot as well as response vs residuals.

```
ggplot(data6, aes(
  x = predict(mixed_model),
  y = residuals(mixed_model),
  col = treatment)) +
  geom_point() +
  geom_smooth(se = FALSE, method = "lm") +
  labs(x = "predictions", y = "Fitted values") +
  theme(legend.position = "none") +
  ggplot(data6, aes(
  x = response,
  y = residuals(mixed_model),
  col = treatment)) +
  geom_point() +
  labs(x = "response values", y = "Fitted values")
```

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



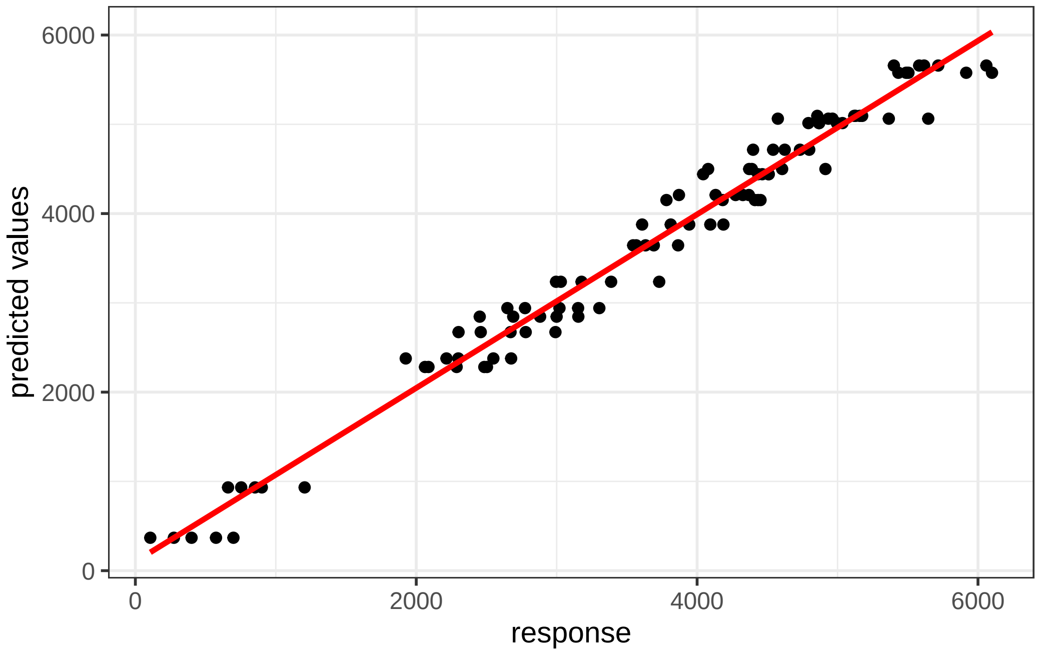
There is some remaining structure in the control and treatment slopes, however this is likely due to this graph not having accounted for the individual repetitions.

As we can see in the plot above, there is no evidence of outliers, unexplained structure or non linearity. Notice the predicted scores are falling out into 20 discrete vertical patterns of 5 points. This is expected since we had 5 repeated measures for 10 patients over 2 treatments.

Let's also have a look at the response vs predicted graph:

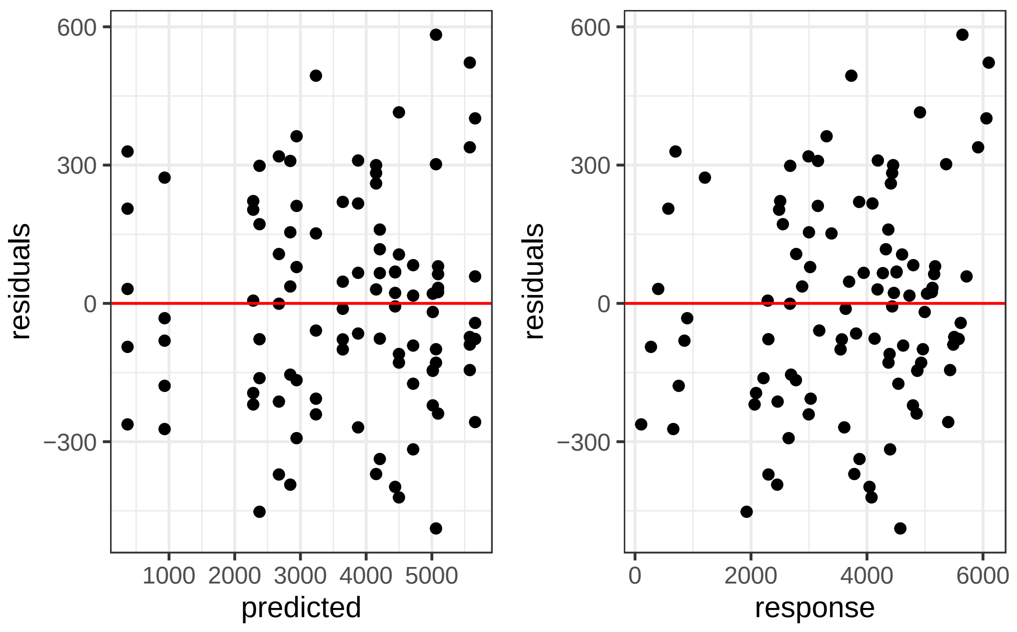
```
ggplot(data6, aes(x = response, y = predict(mixed_model)), col = id) +
  geom_point() +
  labs(y = "predicted values") +
  geom_smooth(method = "lm", se = FALSE, col = "red")
```

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



We should also check response vs residuals as well as predict vs. residuals:

```
ggplot(data6, aes(x = predict(mixed_model), y = residuals(mixed_model))) +
  geom_point() +
  geom_hline(yintercept = 0, col = "red") +
  labs(y = "residuals", x = "predicted") +
  ggplot(data6, aes(x = response, y = residuals(mixed_model))) +
  geom_point() +
  geom_hline(yintercept = 0, col = "red") +
  labs(y = "residuals") +
  plot_layout(guides = "collect")
```



After assessing the model diagnostics, let's look at the summary output.

Step 5: Model interpretation and conclusion

```
summary(mixed_model)
```

```
Linear mixed model fit by REML. t-tests use Satterthwaite's method [
lmerModLmerTest]
Formula: response ~ treatment + (1 | id)
   Data: data6

REML criterion at convergence: 1415.9

Scaled residuals:
     Min       1Q   Median       3Q      Max
-2.00798 -0.64360 -0.01528  0.64054  2.39706

Random effects:
 Groups   Name        Variance Std.Dev.
 id       (Intercept) 2178377  1475.9
 Residual               59105   243.1
Number of obs: 100, groups:  id, 10
```

```
Fixed effects:
                   Estimate Std. Error       df t value Pr(>|t|)    
(Intercept)        3398.202    467.996    9.049   7.261 4.63e-05 ***
treatmentTreatment  563.738     48.623   89.000  11.594  < 2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Correlation of Fixed Effects:
            (Intr)
trtmntTrtmn -0.052
```

As we can see from the above output, we have a significant treatment effect. The control variable `b0.control`, though, is not significantly different from 0.

Refer to the presentation here for more detail.

Step 6: Report overall conclusion

With the model output above, we can now make a conclusion. In order to do so, let's first also calculate the confidence intervals of each estimates:

```r
confint(mixed_model)
```

```
Computing profile confidence intervals ...
```

```
                       2.5 %    97.5 %
.sig01               953.2940 2340.1219
.sigma               210.3224  281.8879
(Intercept)         2437.6856 4358.7182
treatmentTreatment   467.9490  659.5266
```

The following statement can now be made:

> There is strong evidence to show that the treatment influences the response (p<2e-16). It increases # of the response by 0.32 to 0.45 (95% CI). This effect has been estimated fairly accurately [as 95% CI isn't too wide].

Was it worth fitting the more complex model?

Let's compare the random (output above) to the fixed model:

```r
fixed_effects_anova <- lm(response ~ treatment, data = data6)
summary(fixed_effects_anova)
```

```

Call:
lm(formula = response ~ treatment, data = data6)

Residuals:
    Min      1Q  Median      3Q     Max 
-3302.0  -939.6   373.6  1040.5  2138.0 

Coefficients:
                   Estimate Std. Error t value Pr(>|t|)    
(Intercept)          3398.2      203.0  16.743   <2e-16 ***
treatmentTreatment    563.7      287.0   1.964   0.0524 .  
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 1435 on 98 degrees of freedom
Multiple R-squared:  0.03787,	Adjusted R-squared:  0.02805 
F-statistic: 3.857 on 1 and 98 DF,  p-value: 0.05236
```

If we fit a simple ANOVA model like we did previously it shows marginal support that the treatment has an impact (treatment p=0.0524) while the random model has strong support ($p < 2e\text{-}16$). This is because the effect of treatment has been hidden by the noise in the data set (residual=1435), while the residual for the random model is much smaller (222) meaning it has more power. This is because the differences between subjects is included in the fixed effects residual, but is partitioned out in the random effects as the id-intercept SD (1580).

> There was much larger variation between patients (sd=1.0151) than within (sd=0.1678), meaning it was worthwhile partitioning it out for a more accurate model.