# Chapter 9 Linear regression

Adapted by Abigail Noyce from *Fundamentals of Quantitative Analysis*, James Bartlett and Wilhelmiina Toivo.[32]

Linear regression models test research questions and hypotheses between at least one continuous predictor (sometimes in combination with a categorical predictor) and and a continuous outcome. As you might have guessed, this is another application of the *general linear model*, the large family of tools for the prediction of continuous variables.

By *linear*, we mean that the model does not include any quadratic ($x^2$) terms, but it can include ( $x_1x_2$) terms, where two predictors are multiplied together.

## 9.1 Correlation

For practice, we'll look at data from Dawtry et al. (2005).[33] The authors investigated why people with more money tend to oppose wealth redistribution policies like higher taxes for higher incomes to decrease inequality in society. We are using data from Study 1A where 305 people completed measures on household income, predicted population income, their predicted social circle income, in addition to measures on support for wealth redistribution and fairness and satisfaction with the current system.

They predicted people with higher incomes have social circles with higher incomes, so they are more satisfied with the current system of wealth redistribution and less interested in changing it. In essence, poorer people and richer people have different experiences of how rich and equal their country is. In this chapter, we will explore the relationship between a range of these variables. Our main research question will be, "Is there a relationship between support for wealth redistribution and fairness and satisfaction with the current system?"

### 9.1.1 Initial exploration and descriptive statistics

Read in data, reverse-code two items, create composite variables, and select only relevant columns for the analysis.

```
datafile <- here("datasets/dawtry/Dawtry_2015.csv")

dawtry_data <- read.csv(datafile)

dawtry_clean <- dawtry_data %>%

  # reverse code, 6 to 1, 5 to 2, 4 to 3, etc
  mutate(redist2_R = 7 - redist2,
         redist4_R = 7 - redist4) %>%

  # group by rows to compute composite scores
  rowwise() %>%

  # mean across columns and ungroup
  mutate(fairness_satisfaction = mean(c(fairness, satisfaction)),
         redistribution = mean(c(redist1, redist2_R, redist3, redist4_R))
         ) %>%
  ungroup() %>%

  # select variables of interest
  select(PS,Household_Income:redistribution, -redist2_R, -redist4_R)

head(dawtry_clean)
```

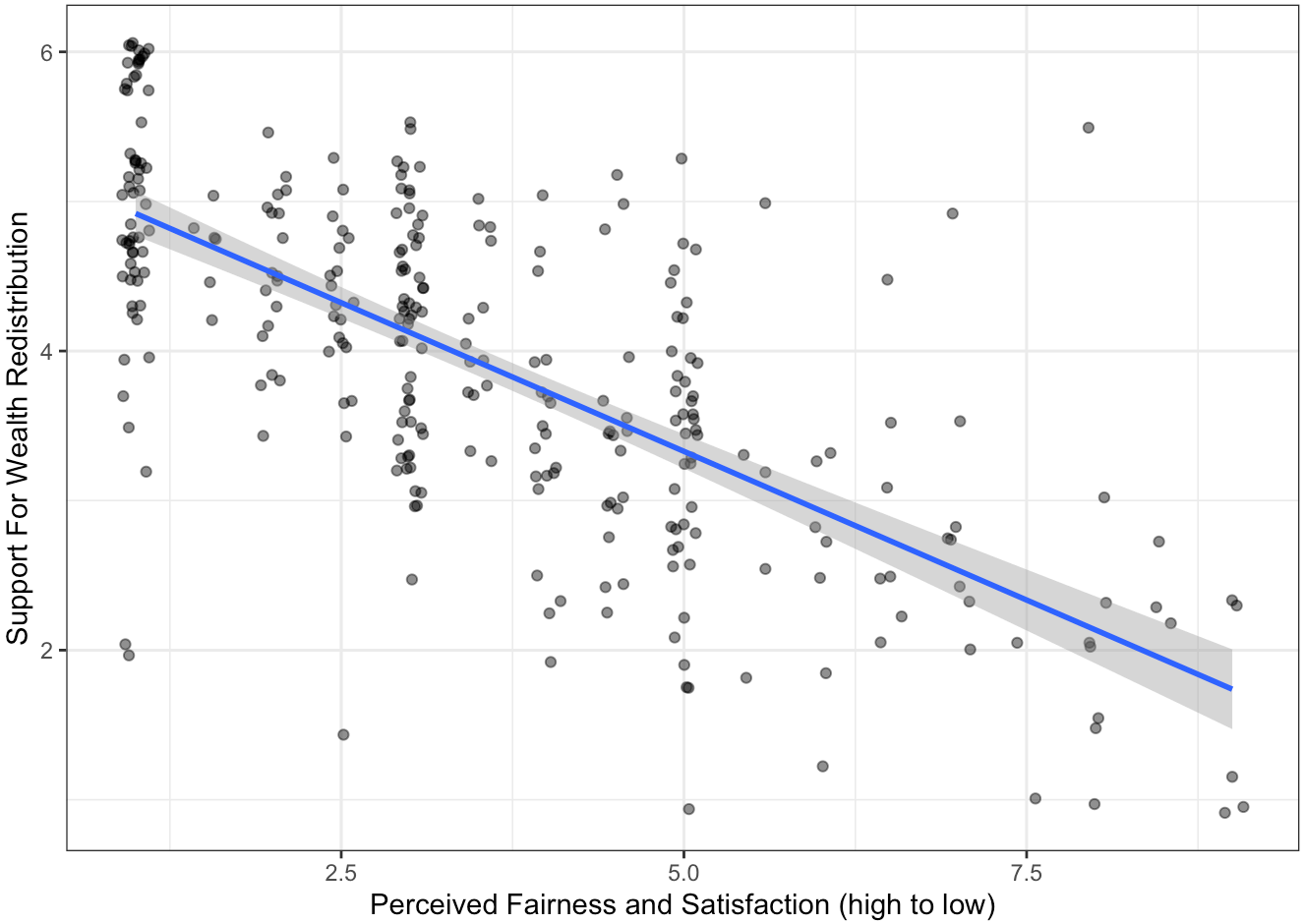
```
## # A tibble: 6 × 11
##      PS Household_Income Political_Preference   age gender
##   <int>            <dbl>                <int> <int>  <int>
## 1   233               NA                    5    40      2
## 2   157               20                    5    59      2
## 3   275              100                    5    41      2
## 4   111              150                    8    59      2
## 5    52              500                    5    35      1
## 6    11              600                    3    34      2
## # ℹ 6 more variables: Population_Inequality_Gini_Index <dbl>,
## #   Population_Mean_Income <int>, Social_Circle_Inequality_Gini_Index <dbl>,
## #   Social_Circle_Mean_Income <int>, fairness_satisfaction <dbl>,
## #   redistribution <dbl>
```

Before we cover regression as a more flexible framework for inferential statistics, we think it is useful to start with correlation to get a feel for how we can capture the relationship between two variables. As a reminder, correlations take the covariance between two variables and standardize it by their standard deviations, resulting in values that range from -1 (a perfect negative correlation) to 1 (a perfect positive correlation).

To explore the relationship between two variables, it is useful to create a scatterplot early for yourself, then provide a more professional looking version to help communicate your results. We plot these data with `fairness_satisfaction` on the x axis because this is the hypothesized predictor for our data; `redistribution` goes on the y axis as a hypothesized outcome

```
ggplot(data = dawtry_clean, aes(x = fairness_satisfaction, y = redistribution)) +
  geom_jitter(alpha = 0.5, width = 0.1, height = 0.1) +
  geom_smooth(method = "lm") +
  scale_x_continuous(name = "Perceived Fairness and Satisfaction (high to low)") +
  scale_y_continuous(name = "Support For Wealth Redistribution") +
  theme_bw()
```

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




The scatterplot shows a negative correlation between the two variables. You need to be careful interpreting fairness and satisfaction as it is coded a little counterintuitively. Higher values mean great *dissatisfaction*.

As support for wealth redistribution increases to be more positive, perceived fairness and satisfaction tends to decrease. This makes sense as people who are more dissatisfied with the current system think there should be more wealth redistribution strategies.

Visualising the relationship between two variables is great for our understanding, but it does not tell us anything about the inferential statistics for what we can learn from our sample in hypothesis testing and measures of effect size.

## 9.1.2 Using `correlation()`

The two most common versions of a correlation are:

- Pearson's product-moment correlation (often shortened to the Pearson correlation) and symbolised by $r$.

- Spearman's rank correlation coefficient (often shortened to the Spearman correlation) and symbolised by $r_s$ or sometimes the Greek letter $\rho$ (rho).

There is a function built into R ( `cor.test()` ) to calculate the correlation between two variables, but we tend to use the `correlation()` function from the **correlation** package as it has more consistent reporting features. The `correlation()` function requires:

- The name of the data set you are using.
- The name of the first variable you want to select for the correlation.
- The name of the second variable you want to select for the correlation.
- The type of correlation you want to run: e.g. `pearson` , `spearman` .

```
correlation(data = dawtry_clean,
            select = "fairness_satisfaction",
            select2 = "redistribution",
            method = "pearson",
            alternative = "two.sided")
```

```
## # Correlation Matrix (pearson-method)
##
## Parameter1            |     Parameter2 |     r |         95% CI | t(303) |
## ------------------------------------------------------------------------------
## fairness_satisfaction | redistribution | -0.70 | [-0.75, -0.64] | -17.08 | < .00
##
## p-value adjustment method: Holm (1979)
## Observations: 305
```

Your output will look a little different due to how the book renders tables, but you should get the same information. For the three key concepts of inferential statistics, we get

- **Hypothesis testing**: $p < .001$, suggesting we can reject the null hypothesis assuming $\alpha = .05$.
- **Effect size**: $r$ = -.70, suggesting a strong negative correlation.

- **Confidence interval**: [-0.75, -0.64], showing the precision around the effect size estimate.

To summarise: a Pearson correlation showed there was a strong, negative, statistically significant relationship between attitudes on wealth redistribution and perceived fairness and satisfaction, $r$ (303) = -0.70, $p < .001$, 95% CI = [-0.75, -0.64].

## 9.2 Linear regression with one continuous predictor

Now that you know how to calculate a correlation in R, we can turn to simple linear regression as a more flexible tool for modelling the relationship between variables. We'll start with just one predictor variable, before scaling up to more complicated models.

### 9.2.1 Calculate descriptive statistics

Let's get the mean and standard deviation of our variables, since we'll want them for our final report. If other types of descriptive statistic would be more suitable to your data, then you can just replace the functions you use within `summarise()`.

```
dawtry_descriptives <- dawtry_clean %>%

  # pivot longer for cleaner, less repetitive code
  pivot_longer(cols = fairness_satisfaction:redistribution,
               names_to = "Variable",
               values_to = "Value") %>%

  # group by Variable to get one value for each variable
  group_by(Variable) %>%

  # mean and SD, rounded to 2 decimal places
  summarize(mean_value = round(mean(Value),2),
            sd_value = round(sd(Value),2))

dawtry_descriptives
```

```
## # A tibble: 2 × 3
##   Variable              mean_value sd_value
##   <chr>                      <dbl>    <dbl>
## 1 fairness_satisfaction       3.54     2.02
## 2 redistribution              3.91     1.15
```

### 9.2.2 Using the `lm()` function

For our research question of "is there a relationship between support for wealth redistribution and fairness and satisfaction", we can address it with simple linear regression.

Instead of a standardised correlation coefficient, we can frame it as whether knowing fairness and satisfaction can predict values of support for wealth redistribution. The design is still correlational, so it does not tell us anything about a causal relationship in isolation. We use the word *predict* in the statistical sense, where we can ask whether knowing values of one variable help us predict values of another variable with a degree of error.

The first step is to create an object ( `lm_redistribution` ) for the linear model.

```
lm_redistribution <- lm(redistribution ~ fairness_satisfaction,
                        data = dawtry_clean)
```

The function `lm()` is built into R and is incredibly flexible for creating linear regression models.

- The first argument is to specify a formula which defines our model. The first component ( `redistribution` ) is our outcome variable for what we are interested in modelling.[34]
- The tilde ( `~` ) separates the equation, where everything on the right is your predictor variable(s). In simple linear regression, we just have one predictor, which is `fairness_satisfaction` in our model here. This is saying we want to predict `redistribution` as our outcome from `fairness_satisfaction` as our predictor.
- We then specify the data frame we want to use.

When you create this object, it stores a bunch of information, but does not really tell us all the statistics we expect. If you simply print the object in the console, it will tell you the intercept and coefficient(s), but none of the model fitting nor hypothesis testing values.

```
lm_redistribution
```

```
##
## Call:
## lm(formula = redistribution ~ fairness_satisfaction, data = dawtry_clean)
##
## Coefficients:
##           (Intercept)  fairness_satisfaction
##                5.3169                -0.3975
```

If you look at the object in the R environment, you will see it is a list containing several elements. It stores things like the model, the residuals, and other information you can use. To get that extra information, we need to call the `summary()` [35] function around the linear model object to explore its properties like estimates and model fit.

```
summary(lm_redistribution)
```

```
##
## Call:
## lm(formula = redistribution ~ fairness_satisfaction, data = dawtry_clean)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2.9193 -0.5279  0.0233  0.4782  3.3634
##
## Coefficients:
##                       Estimate Std. Error t value Pr(>|t|)
## (Intercept)            5.31686    0.09488   56.04   <2e-16 ***
## fairness_satisfaction -0.39754    0.02328  -17.08   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.8218 on 303 degrees of freedom
## Multiple R-squared:  0.4905, Adjusted R-squared:  0.4888
## F-statistic: 291.7 on 1 and 303 DF,  p-value: < 2.2e-16
```

To walk through the output, `Call:` summarises the model you specified. `Residuals:` provides a summary of the model residuals which we will come back to later. `Coefficients:` provides our model output, this time with inferential statistics. The two key lines are:

1. `(Intercept)` - This is the value of the outcome when our predictor is set to 0. For a fairness and satisfaction value of 0, we would expect a value of 5.32 for redistribution. You get a *p*-value for this, but in isolation it is not too useful. It just compares the intercept estimate to 0 which typically you are not interested in.

- `fairness_satisfaction` - This is the regression slope or coefficient. This is the change in the outcome for every 1 unit change in the predictor. So, for every 1 unit increase in fairness and satisfaction, we expect support for wealth redistribution to decrease (as we have a negative value) by 0.40 units. This is consistent with the correlation as we have a negative relationship between the two variables. Looking at the *p*-value, this is statistically significant ($p < .001$), suggesting we can reject the null hypothesis and conclude there is an effect here.

Does it matter if the slope is positive or negative? When you have a continuous predictor, the sign is important to keep in mind. A positive slope would mean an increase in the predictor is associated with increased values of your outcome. A negative slope would mean an increase in the predictor is associated with decreased values of your outcome. This is crucial for interpreting the coefficient.

At the bottom of the model output, you then get the fit statistics. Multiple $R^2$ tells you how much variance in your outcome your predictor(s) explain. Adjusted $R^2$ tends to be more conservative as it adjusts for the number of predictors in the model (something we will not cover until Chapter 14), but they will be very similar when you have one predictor. Adjusted $R^2$ is .49, suggesting fairness and satisfaction explains 49% of the variance in support for wealth redistribution.

Finally, we have the model fit statistics to tell us whether the model explains a significant amount of variance in the outcome. With one predictor, the *p*-value next to the coefficient and next to the model fit will be identical, as one predictor is the whole model. The F-statistic is 291.7, the model degrees of freedom is 1, the residual degrees of freedom is 303, and the *p*-value is $p < .001$.

Remember, if you want to "uncompress" a number in scientific notation, you can ask R like so:

```
format(2e-16, scientific = FALSE)
```

```
## [1] "0.0000000000000002"
```

### 9.2.3 Calculating confidence intervals

In the standard `lm()` and `summary()` output, we get most of the key values we need for our inferential statistics, but the one thing missing is confidence intervals around our estimates. Fortunately, R has a built-in function called `confint()` for calculating confidence intervals using your linear model object.

```
confint(lm_redistribution)
```

```
##                              2.5 %     97.5 %
## (Intercept)            5.1301581  5.5035664
## fairness_satisfaction -0.4433442 -0.3517332
```

Normally, you focus on the confidence interval around your slope estimate as the intercept is not usually super useful for interpreting your findings when you have a continuous predictor. Now, we can summarise the three key concepts of inferential statistics as:

- **Hypothesis testing**: $p < .001$, suggesting we can reject the null hypothesis assuming α = .05. Fairness and satisfaction is a significant predictor of support for wealth redistribution.
- **Effect size**: $b_1$ = -0.40, suggesting fairness and satisfaction is a negative predictor.
- **Confidence interval**: [-0.44, -0.35], showing the precision around the slope estimate.

## 9.2.4 Checking assumptions

As a reminder, the assumptions for simple linear regression are:

1. The outcome is interval/ratio level data.
2. The predictor variable is interval/ratio or categorical (with two levels at a time).
3. All values of the outcome variable are independent (i.e., each score should come from a different participant/observation).
4. The predictors have non-zero variance.
5. The relationship between the outcome and predictor is linear.
6. The residuals should be normally distributed.
7. There should be homoscedasticity.

Assumptions 1-4 are pretty straight forward as they relate to your understanding of the design or a simple check on the data for non-zero variance (the responses are not all the exact same value).

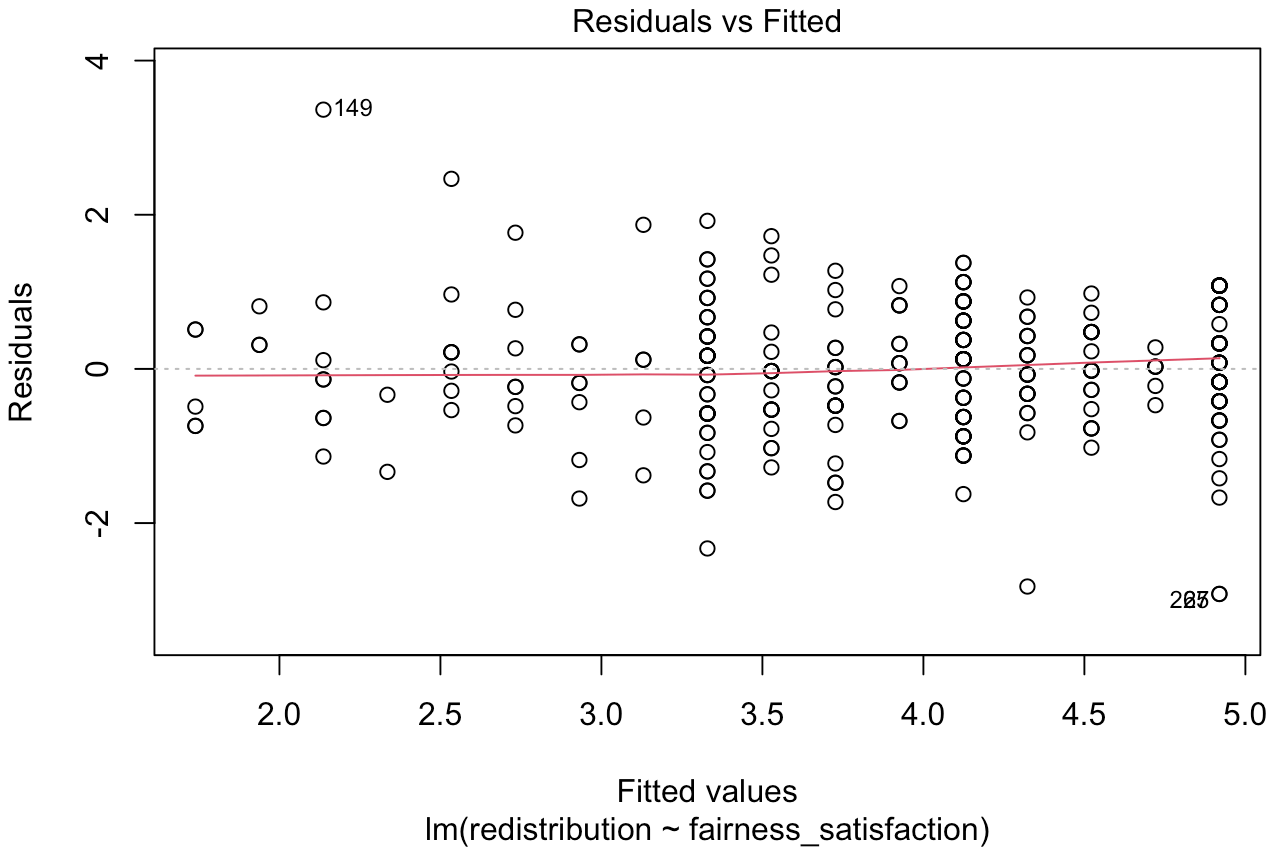
Assumptions 5-7 require diagnostic checks on the residuals from the model. The residuals are the difference between your observed values in the data and the values your model predicts given it's assumptions. If you remember back, we highlighted the model output mentioned `Residuals:` and they are saved within the model object.

In your reading, you might see individual statistical tests to check these assumptions, but they have more limitations than benefits. The best way to check the assumptions is through diagnostic plots which express the model residuals in different ways. We want to walk through this longer way of checking assumptions to develop a solid understanding before showing you a shortcut to see them all below.

#### 9.2.4.1 Checking linearity

(You can get all four of these plots by calling `plot(lm_redistribution)`, but here we will look at them one at a time.)

```
plot(lm_redistribution,
     which=1)
```




Plot 1 is a residuals vs fitted plot and helps us check linearity by showing the residuals on the y-axis and the fitted (predicted) values on the x-axis.

Here, you are looking out for a roughly flat horizontal red line. Common patterns to look out for if there is a problem are when the line has an obvious curve to look like a hump or several bends to look like an S.

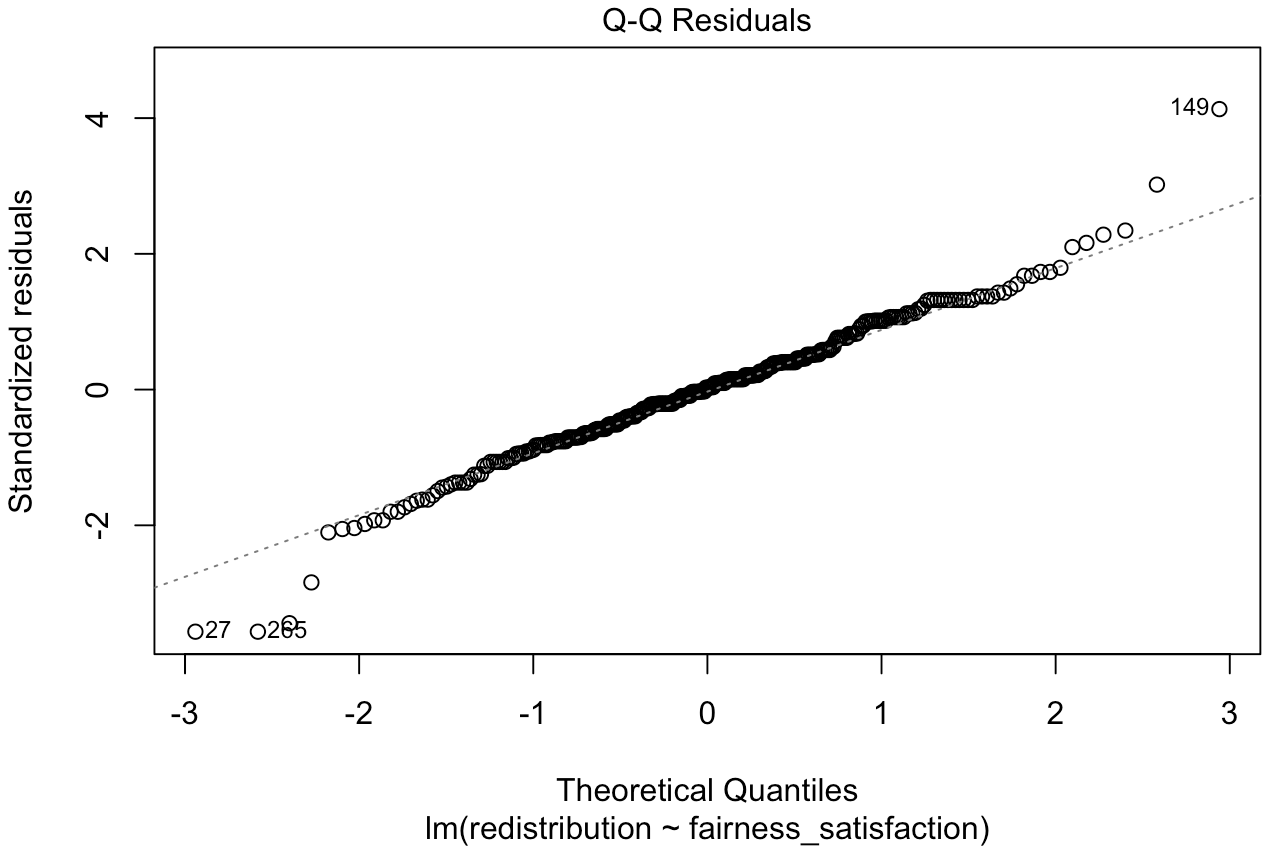
The only downside to using diagnostic plots is it takes experience to recognise when there is nothing wrong with a regression model compared to when it violates the assumptions. It is easy to see a little deviation and think there is some drastically wrong. Our advice is if you squint and it looks fine, it is probably fine. You are looking for clear and obvious deviations from what you expect.

### 9.2.4.2 Checking normality

Plot 2 is a qq-plot (quantile-quantile plot) and helps us check the normality of the model residuals by showing the standardised residuals on the y-axis and the theoretical quantiles on the x-axis. A common misconception is your variables should all be normally distributed, but it is

actually the model residuals which should be normal.

```
plot(lm_redistribution,
     which=2)
```




In this plot, you are looking for the data points to roughly follow the dashed line. The idea is there should be a linear relationship between the residuals and the values we would expect under a normal distribution.

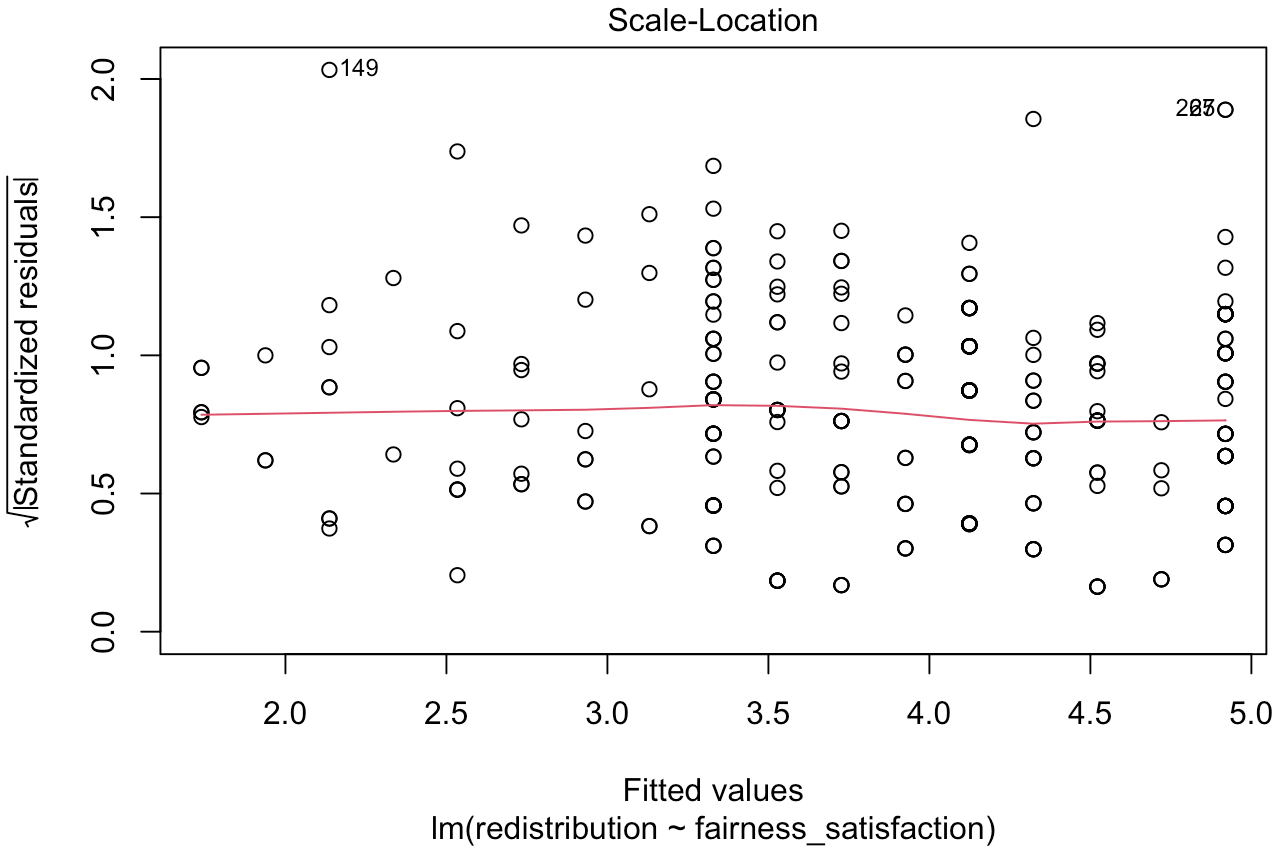
Like the other diagnostic plots, it is tempting to think there are problems where there are none. The vast majority of the points here follow the line nicely, but tail off a little at the extremes. It flags the points with the largest deviations but there do not appear to be any obvious problems.

When there are problems with normality, you are looking for obvious deviations, like the points curving around in a banana shape, or snaking around like an S.

### 9.2.4.3 Checking homoschedasticity (homogeneity of variance)

Plot 3 is a scale-location plot and helps us check homoscedasticity by showing the square root of the standardised residuals on the y-axis and the fitted values on the x-axis. Homoscedasticity is where the variance of the residuals is approximately equal across the range of your predictor(s).

```
plot(lm_redistribution,
     which = 3)
```




In this plot, you are looking out for a roughly random spread of points as you move from one end of the x-axis to the other. The red line should be roughly flat and horizontal.

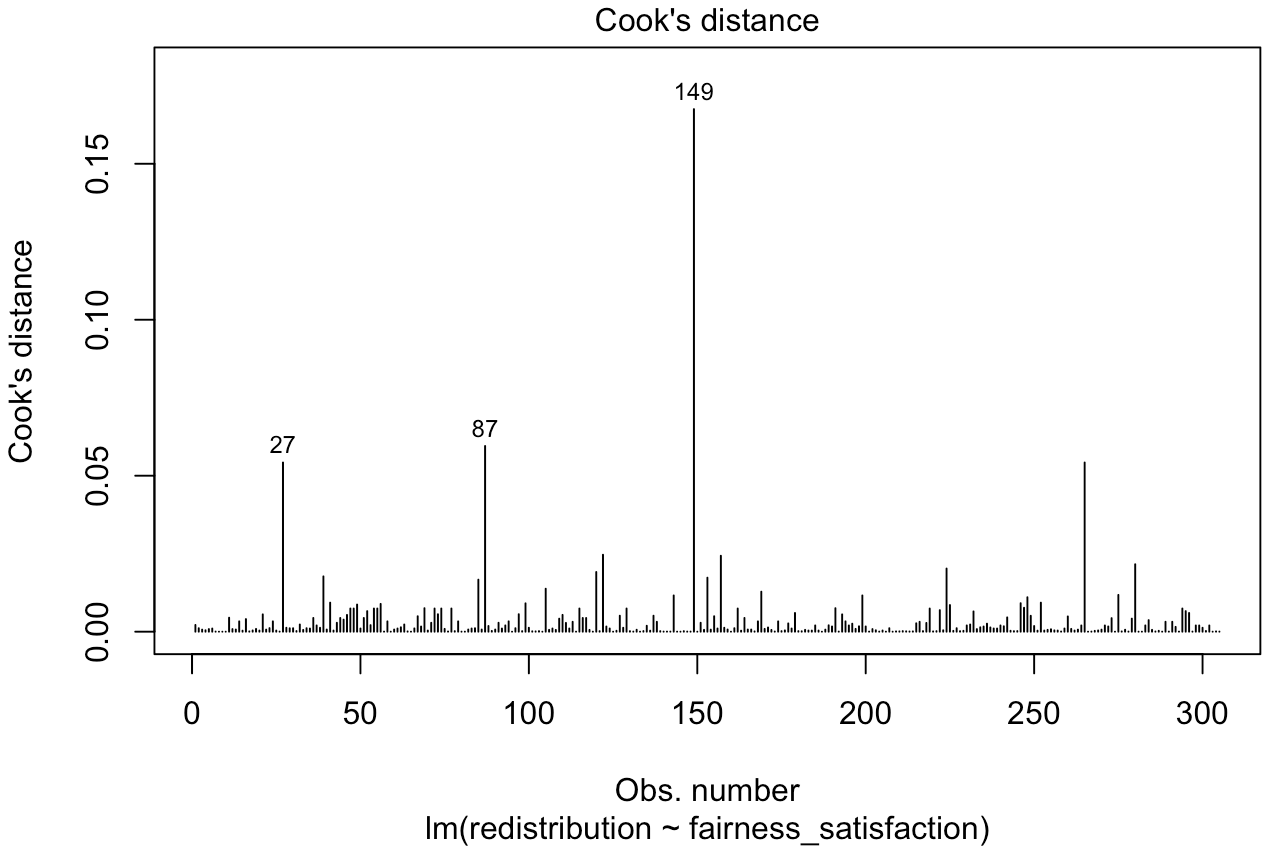
When there is *heteroscedasticity*, the characteristic patterns to look out for are a kind of arrow shape where there is a wide spread of points at one end and decreases to a denser range at the other end, or a bow tie where there is a wide spread of points at each end and dense in the middle.

### 9.2.4.4 Checking influential cases

Finally, there are two main plots to help us identify influential cases. You might have heard of the term outlier before and this is one way of classifying data points that are different enough to the rest of the data in a regression model. It is not strictly an assumption of linear regression, but it can affect the other assumptions.

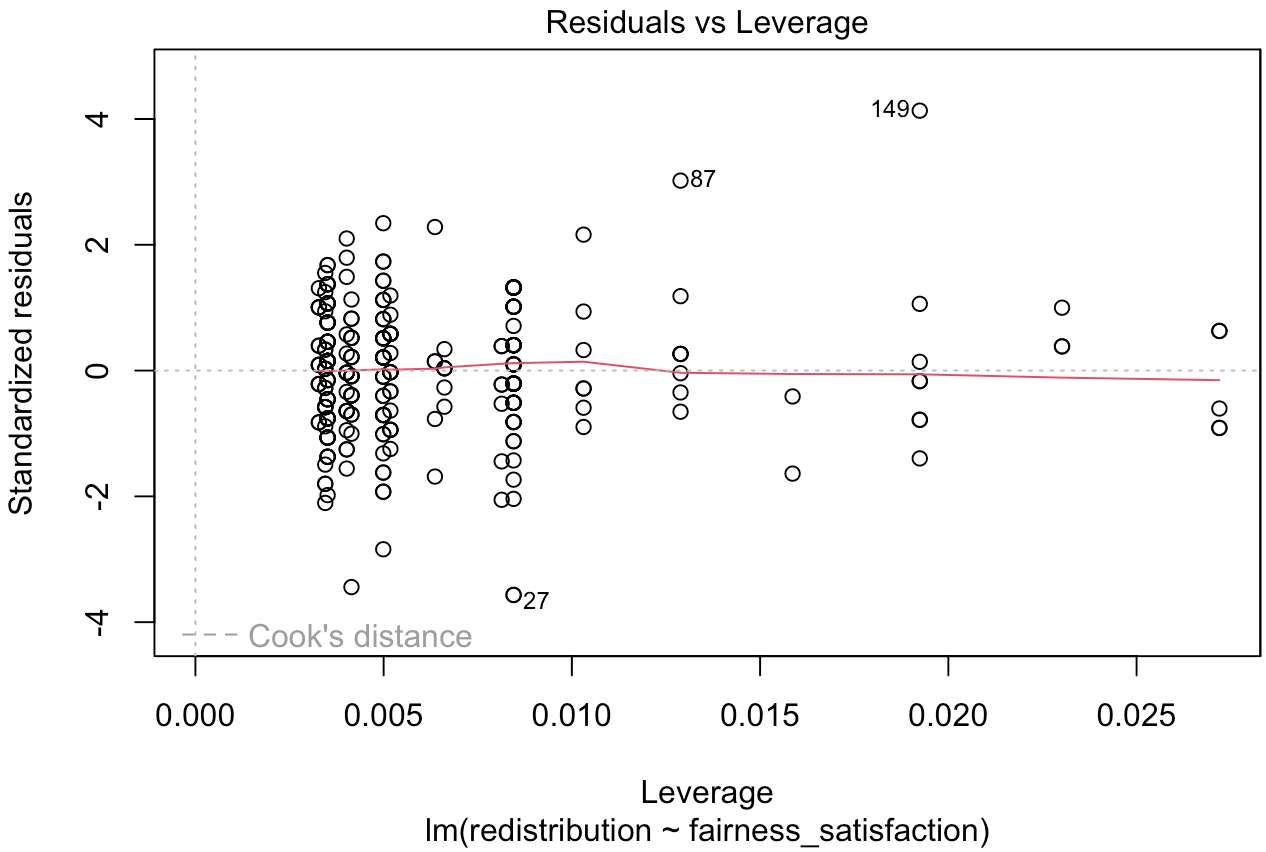
If you only run `plot(lm_redistribution)` and cycle through the plots, you do not see this version. This plot shows values of Cook's distance for each observation in your data along the x-axis. Cook's distance measures the influence of deleting a given observation, where higher values mean deleting that observation results in a larger change to the model estimates. There are different thresholds in the literature, but estimates range from 1, 0.5, to 4/n.

```
plot(lm_redistribution,
     which = 4)
```



Finally, we get a residuals vs leverage plot to show influential cases in a slightly different way. Instead of just the Cook's distance value of each observation, it plots the standardised residuals against leverage values.

```
plot(lm_redistribution,
     which = 5)
```



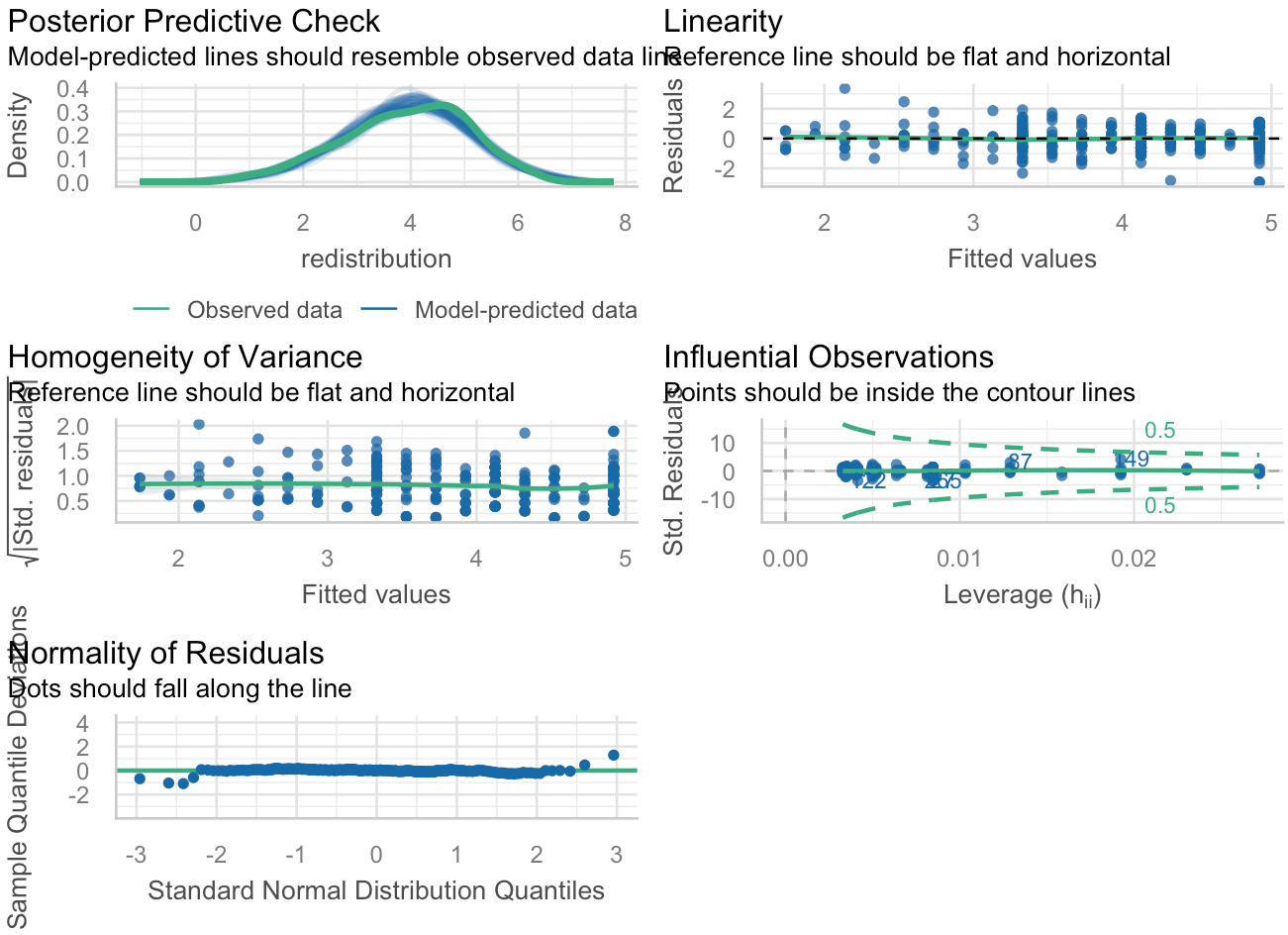

Influential points and potential outliers would have high leverage values and the plot will show a threshold of Cook's distance as red dashed lines. In this plot, they are not visible as there is no value with a big enough leverage value, but you would be looking for data points outside this threshold to identify influential values.

### 9.2.4.5 Checking all the assumptions

Now we have covered the individual diagnostic plots, there is a handy function called `check_model()` from the **performance** package. This function reports all the diagnostic checks from `plot()`, but tidies up the presentation and has some useful reminders of what you are

looking for.

```
check_model(lm_redistribution)
```




The key difference is you get a posterior predictive check (essentially comparing values you observe compared to what your model predicts) and the qq-plot for normality of residuals looks a little different. Instead of a kind of angled line, the residuals are expressed as deviations instead, so the points should be close to a flat horizontal line. This version can make smaller deviations look worse, so keep in mind again you are looking for clear deviations in the overall pattern.

## 9.2.5 Reporting your results

Now we have some results to go with, there are a few recommendations on how to communicate that information. In psychology (and other disciplines), we tend to follow the American Psychological Association (APA) formatting guidelines as they provide a comprehensive

standardised style to make sure information is being reported as easily digestible and consistent as possible. Here are some key principles to ensure you provide your reader with enough information.

1. Explain to the reader what your linear regression model was. For example, what were your outcome and predictor variables?
2. Report descriptive statistics to help contextualise your findings. For example, the mean/standard deviation for your outcome and continuous predictor.
3. Provide an appropriate data visualisation to help communciate key patterns to the reader. For example, a scatterplot for the relationship between your outcome and predictor.
4. Report all three key inferential statistics concepts for the coefficient: the slope, the confidence interval around your slope, and the *p*-value for hypothesis testing. When you have one predictor in simple linear regression, you typically focus on the slope as your key effect size that helps address your research question and hypothesis. APA style rounds numbers to 2 decimal places when numbers can be bigger than 1, and 3 decimals with no leading zero when it cannot be bigger than 1. When you report the unstandardized slope, you use the symbol $b_1$ but for the standardized slope, you use Beta instead $\beta 1$.
5. Provide an overview of the model fit statistics for whether your model explained a significant amount of variance in your outcome. Remember: the *p*-value for your model will be the same as for the slope in simple linear regression.

For our main example, we could summarise the findings as:

> Our research question was: is there a relationship between support for wealth redistribution and fairness and satisfaction with the current system? To test this, we applied simple linear regression using fairness and satisfaction as a predictor and support for wealth redistribution as our outcome. Figure 1 shows a scatterplot of the relationship. Our model explained a statistically significant amount of variance in our outcome (adjusted $R^2$ = .489, F(1, 303) = 291.70, $p < .001$). Fairness and satisfaction was a negative predictor, where for every 1-unit increase we expect support for redistribution to decrease by 0.40 ( $b_1$ = –0.40, 95% CI = [-0.44, -0.35], $p < .001$).

## 9.3 Linear regression with multiple predictors

We can extend this approach to express more complicated designs with multiple predictors or interactions between predictors.

### 9.3.1 Introduction to the dataset

Here, we will use open data from Przybylski & Weinstein (2017).[36] This was a large-scale study that found support for the "Goldilocks" hypothesis among adolescents: that there is a "just right" amount of screen time, such that any amount more or less than this amount is associated with lower well-being. This was a huge survey study: the data contain responses from over 120,000 participants! In this chapter, we will look at whether the relationship between screen time and well-being is moderated by participants' (self-reported) gender.

The outcome/dependant variable used in the study was the Warwick-Edinburgh Mental Well-Being Scale (WEMWBS). This is a 14-item scale with 5 response categories, summed together to form a single score ranging from 14-70.

Przybylski & Weinstein (2017) looked at multiple measures of screen time, but we will be focusing on smartphone use. They found that decrements in well-being started to appear when respondents reported more than one hour of weekly smartphone use. Our research question is: Does the negative association between hours of use and well-being (beyond the one-hour point) differ for boys and girls?

```
pry_participants <- read.csv(here("datasets/przylbylski/Przybylski_2017_participant
pry_wellbeing <- read.csv(here("datasets/przylbylski/Przybylski_2017_wellbeing.csv"
pry_screen <- read.csv(here("datasets/przylbylski/Przybylski_2017_screentime.csv"))
```

The `pry_participants` data has information on each participant's background. The `wellbeing` data has info from the well-being questionnaire. The `screen` data has information about screentime use on weekends (variables ending with `we`) and weekdays (variables ending with `wk`) for four types of activities: (1) Using a computer (variables starting with `Comph`, Q10 on the survey); (2) Playing videogames (variables starting with `Comp`, Q9 on the survey); (3) Using a smartphone (variables starting with `Smart`, Q11 on the survey); (4) Watching TV (variables starting with `Watch`, Q8 on the survey).

For our final dependent variable, we want the WEMWBS score for each participant, which is simply the sum of all the participant scores. This should range from 14 to 70, which we check:

```
pry_wemwbs <- pry_wellbeing %>% pivot_longer(cols = starts_with("WB"),
                                             names_to = "Item",
                                             values_to = "Score") %>%
  group_by(Serial) %>%
  summarize(total_wellbeing = sum(Score))

summary(pry_wemwbs$total_wellbeing)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   14.00   42.00   48.00   47.52   54.00   70.00
```

## 9.3.2 Initial visualization

We can plot the relationship between wellbeing and hours of screentime, split across four different categories of technology. This plot is more complex than others we have made, so look carefully to understand what each line of code is doing.

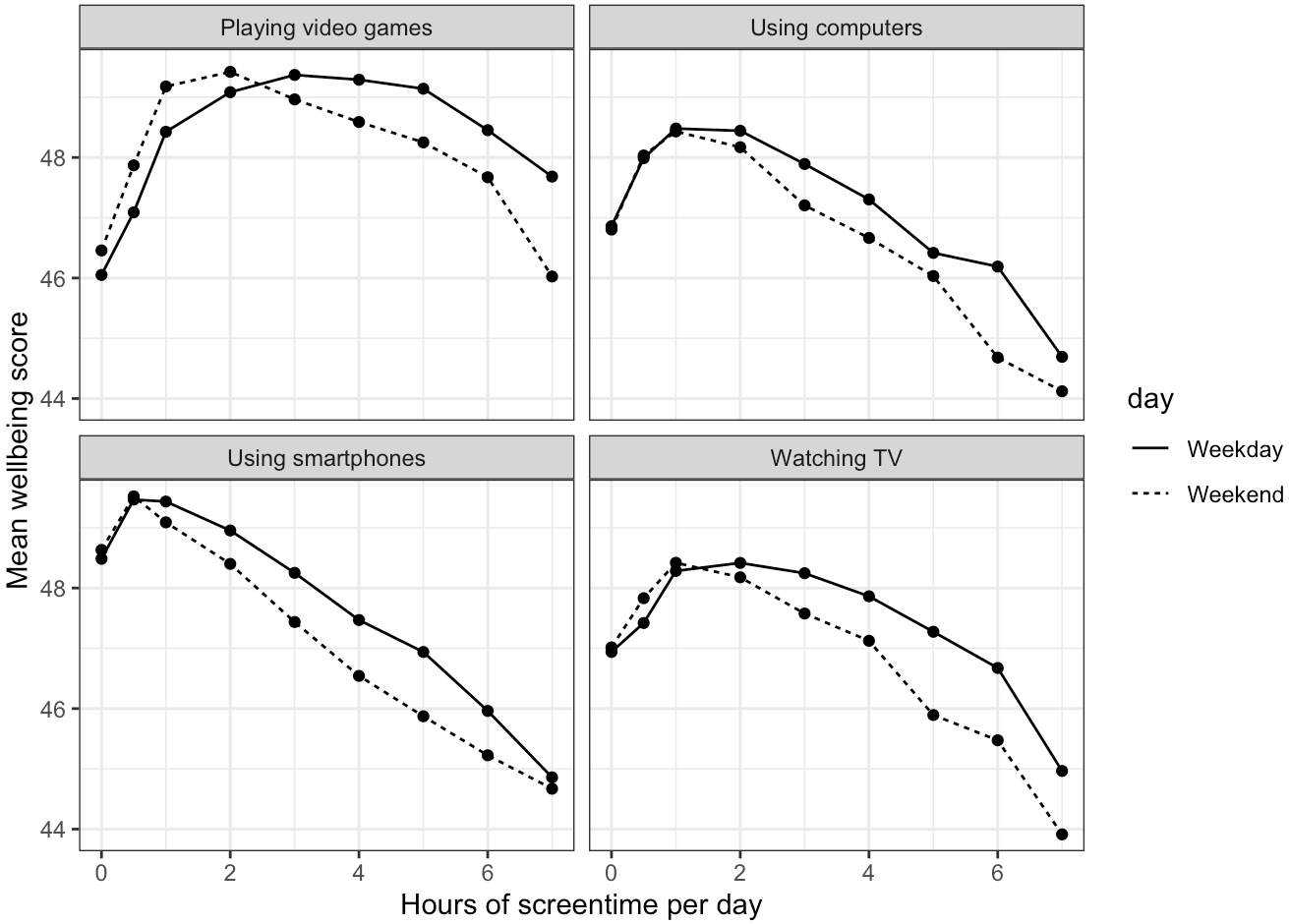
```r
# Wrangle the screentime data into long form with useful variable names

screen_long <- pry_screen %>%

  # pivot longer
  pivot_longer(cols = Comph_we:Watch_wk,
               names_to = "screen_type",
               values_to = "hours") %>%

  # split variable names into screen type and day type
  separate(col=screen_type,
           into = c("screen_type","day"),
           sep = "_") %>%

  # rename for readability.
  # note: mutate() uses "new = old" syntax; case_match uses "old ~ new" syntax
  mutate(screen_type = case_match(screen_type,
                                  "Watch" ~ "Watching TV",
                                  "Comp" ~ "Playing video games",
                                  "Comph" ~ "Using computers",
                                  "Smart" ~ "Using smartphones"),
         day = case_match(day,
                          "wk" ~ "Weekend",
                          "we" ~ "Weekday")

  )
```

```r
# Join the screentime data with the wellbeing data and create group-level means

pry_means <- pry_wemwbs %>%

  # join and combine by Serial (ID). Note inner_join keeps only those participants
  inner_join(screen_long,
             by = "Serial") %>%

  # specify grouping for all factors to keep separated
  group_by(screen_type, day, hours) %>%

  # compute group mean
  summarize(mean_wellbeing = mean(total_wellbeing))
```

```
## `summarise()` has grouped output by 'screen_type',
## 'day'. You can override using the `.groups`
## argument.
```

Plot in four separate quadrants for different screentime types.

```r
ggplot(data = pry_means, aes(x = hours, y = mean_wellbeing, linetype = day)) +
  geom_line() +
  geom_point() +
  facet_wrap(~ screen_type, nrow = 2) +
  labs(x = "Hours of screentime per day",
       y = "Mean wellbeing score") +
  theme_bw()
```




The graph shows that smartphone use of more than 1 hour per day is associated with increasingly negative well-being. We will focus on that observation for our multiple linear regression.

### 9.3.3 Final data wrangling

Now that we have explored the data, we can start preparing for the analysis. In this analysis we have:

- A continuous[37] outcome: well-being.
- One continuous[38] predictor: screen time.
- One categorical predictor: gender.

We want to estimate two slopes relating screen time to well-being, one for girls and one for boys, and then statistically compare these slopes. In some ways this problem seems simultaneously like a situation where you would run a regression (to estimate the slopes) but also one where you

would need a t-test (to compare two groups). This is the power of regression models as you can look at the interaction between a continuous and a categorical predictor, something that isn't possible in factorial ANOVA.

For this analysis, we are going to average weekday and weekend use of smartphones to create a single outcome value per participant, then filter the data to only include those subjects who use a smartphone for at least one hour per day. Finally, we'll combine it with the `male` variable from the participant data.

```
pry_cleandata <- screen_long %>%

  # filter for smartphone entries
  filter(screen_type == "Using smartphones") %>%

  # average weekdays and weekends
  group_by(Serial) %>%
  summarize(mean_hours = mean(hours)) %>%

  # filter for those using smartphones for at least 1h per day
  filter(mean_hours >= 1) %>%

  # join with well-being score
  inner_join(pry_wemwbs,
             by = "Serial") %>%

  inner_join(pry_participants %>% select(c(Serial, male)),
             by = "Serial")
```

When you have continuous variables in a regression model, it is often sensible to transform them by **mean centering**, and this can be particularly important in multiple linear regression. You mean center a predictor X using `X_centered = X - mean(X)` . This has two useful consequences.

1. The model intercept and the other model terms reflect the prediction for Y at the mean value of this predictor variable, rather than at the zero value of the unscaled variable.

2. If there are interactions in the model, lower-order effects can be treated as main effects rather than simple effects.

For categorical predictors with two levels, these become coded as -.5 and .5. This is known as *deviation coding*, and while it can only be used for binary predictors, it has two useful consequences:

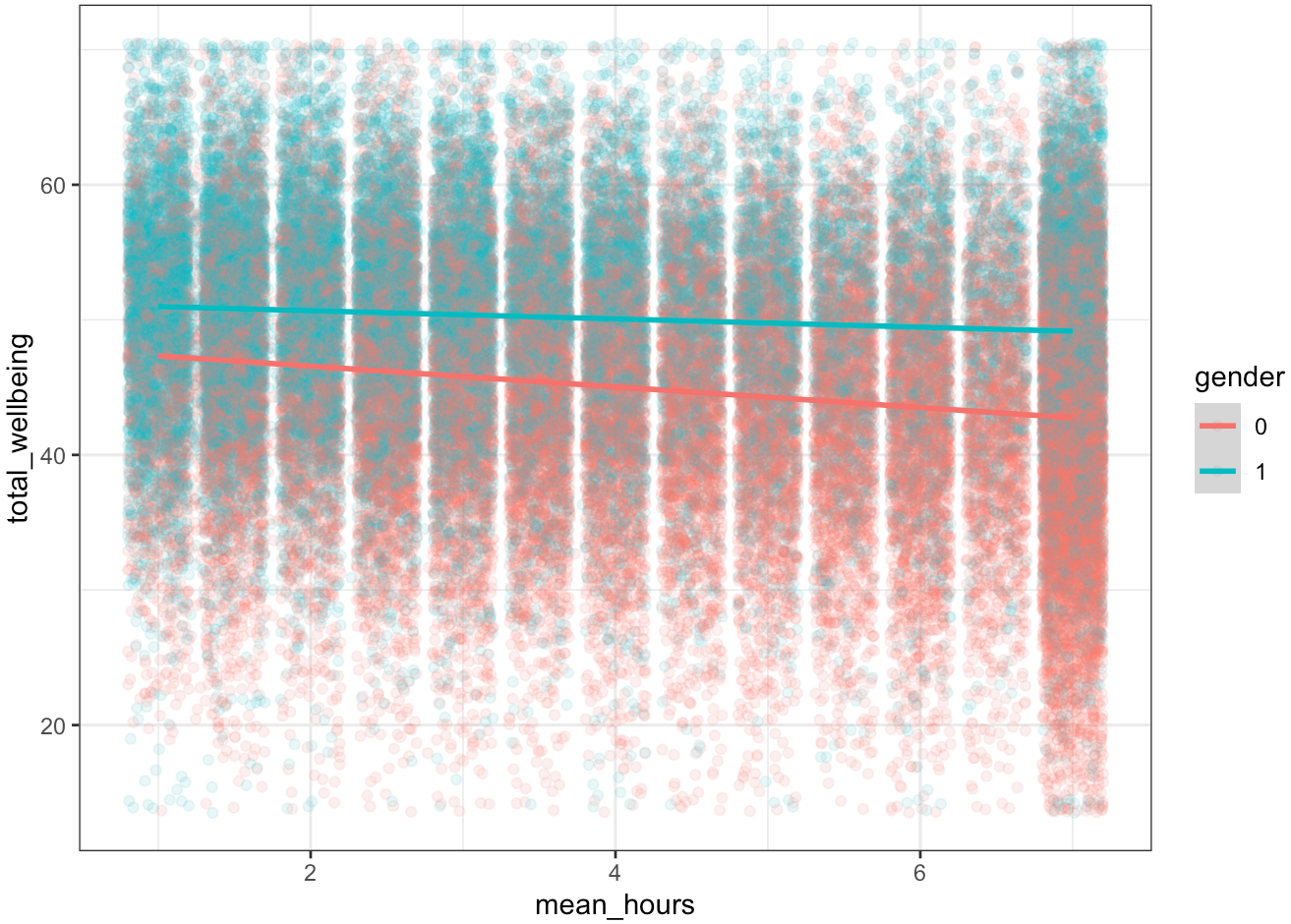
1. The model intercept and other model terms reflect the prediction for Y centered between the two values of this predictor variable, rather than at one or the other value of the variable.
2. The difference between the two levels remains 1, making the model terms for this variable easier to interpret.

```
pry_cleandata <- pry_cleandata %>%
  mutate(mean_hours_centered = mean_hours - mean(mean_hours),
         male_centered = male - 0.5)
```

Now that we have our clean dataframe, let's make a quick scatterplot that we can reference later to understand our model output.

```
ggplot(data = pry_cleandata, aes(x = mean_hours, y = total_wellbeing, color=as.fact
  geom_jitter(alpha = 0.1, width = .2, height=0.5) +
  geom_smooth(method = "lm") +
  labs(color="gender") +
  theme_bw()
```

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




it turns scatterplots are much harder to read when they have 76 thousand points plotted! Even with some jitter added and high transparency, it's hard to eyball any effects from the data points themselves. Adding gm_smooth plots the regression line on top, and we can see that the effect is negative and the slopes of the two lines are slightly different.

While this visualization does not convey the model as clearly, it does convey how large the variance in the dataset is relative to the model effects. While (as we will see below) the fitted model has an extremely significant p-value, it's still accounting for only a small portion of the variability in the data.

### 9.3.4 Running the regression using `lm()`

Our regression model has the form

$$Y_i = \beta_0 + \beta_1 X_{1i} + \beta_2 X_{2i} + \beta_3 X_{3i} + e_i$$

with the following data terms:

- $Y_i$ is the outcome, the well-being score of the $i$th participant

- $X_{1i}$ is the mean-centered smartphone use of the $i$th participant
- $X_{2i}$ is the gender of the $i$th participant
- $X_{3i}$ is the interaction between smartphone use and gender ($X_{1i} - X_{2i}$)

Again, we use `lm()` to specify and fit the model.

```
mod_wb <- lm(data = pry_cleandata,
             formula = total_wellbeing ~ mean_hours_centered + male_centered + mean
             )

summary(mod_wb)
```

```
##
## Call:
## lm(formula = total_wellbeing ~ mean_hours_centered + male_centered +
##     mean_hours_centered * male_centered, data = pry_cleandata)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -36.983  -5.678   0.429   6.169  27.255
##
## Coefficients:
##                                 Estimate Std. Error t value Pr(>|t|)
## (Intercept)                     47.54915    0.03415 1392.31   <2e-16 ***
## mean_hours_centered             -0.53517    0.01678  -31.89   <2e-16 ***
## male_centered                    5.03097    0.06830   73.66   <2e-16 ***
## mean_hours_centered:male_centered  0.45984    0.03356   13.70   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 9.104 on 76403 degrees of freedom
## Multiple R-squared:  0.09512,    Adjusted R-squared:  0.09508
## F-statistic:  2677 on 3 and 76403 DF,  p-value: < 2.2e-16
```

The output of `summary()` shows our model results for each predictor term. Because we used *deviation coding* for our categorical variable, we can interpret these values more cleanly.

- The `(Intercept)` line gives the estimated well-being score at the mean number of smartphone hours, regardless of gender.
- The `mean_hours_centered` line gives the estimated change in well-being with every one-hour increase in smartphone use.
- The `male_centeredmale` line gives the difference in well-being between males and females, at the mean level of smartphone use.
- The interaction `mean_hours_centered:male_centeredmale` line gives the *difference in the slopes* for girls and boys.

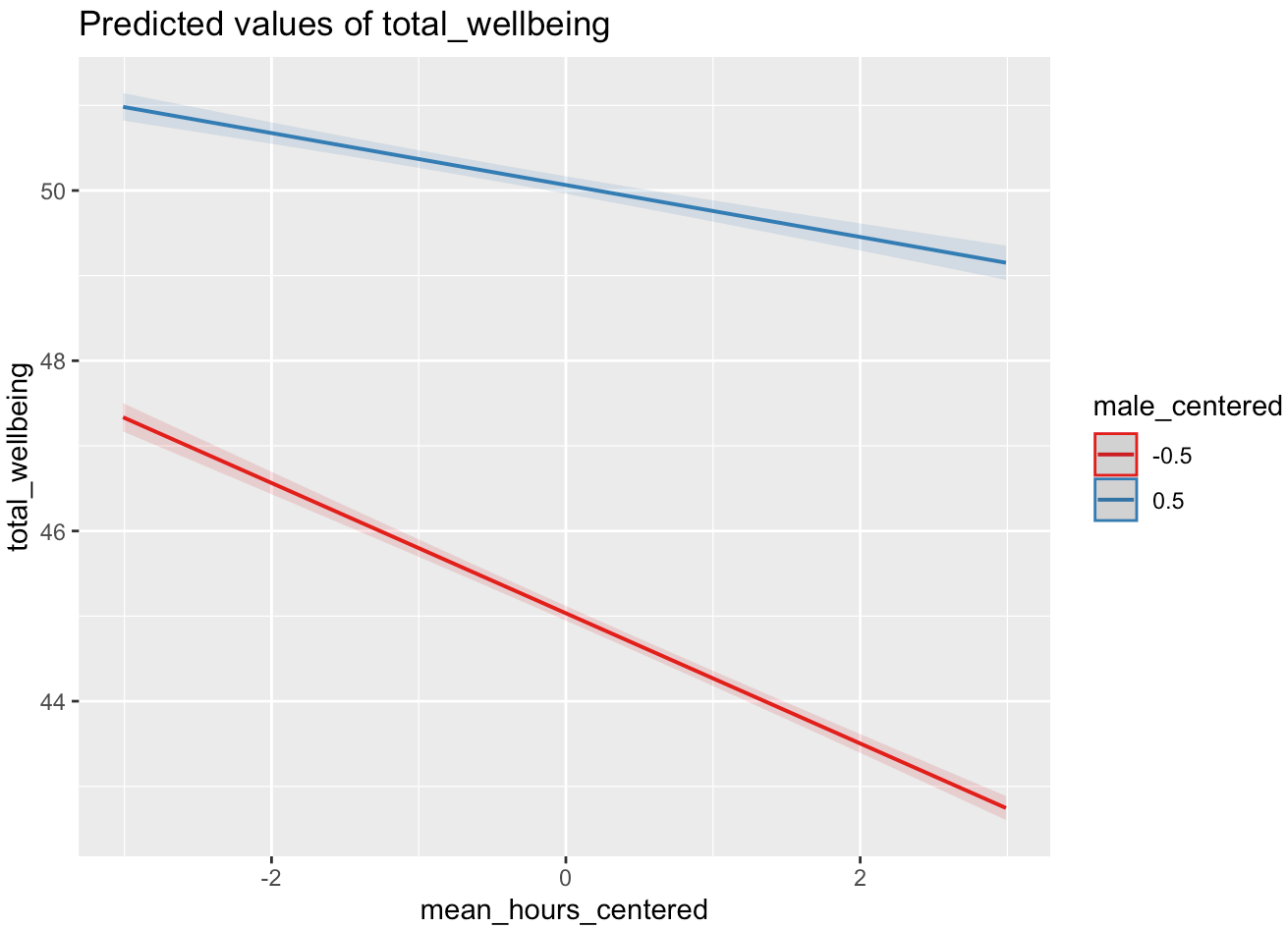
#### 9.3.4.1 Visualizing the interaction

It is very difficult to understand an interaction from the coefficient alone, so your best bet is visualising the interaction to help you understand the results and communicate your results to your readers.

There is a great package called **sjPlot** which takes regression models and helps you plot them in different ways. We will demonstrate one use here, but you can read the full documentation and vignettes online.

To plot the interaction, you need the model object, specify "pred" as the `type` (since we want to plot predicted values), and add the `terms` you want to include:

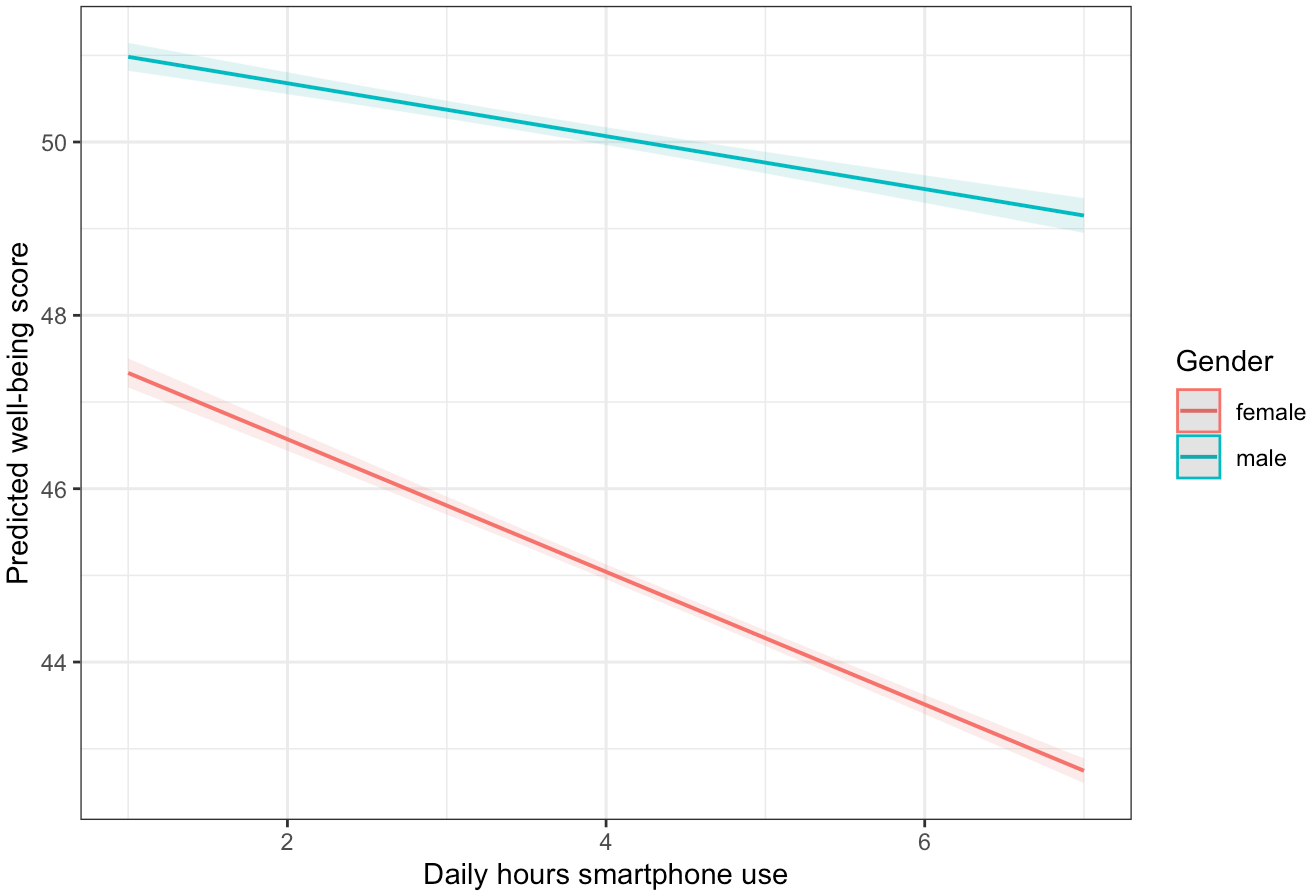
```
plot_model(mod_wb,
           type = "pred",
           terms = c("mean_hours_centered","male_centered"))
```




From this plot we can clearly see the main effect of smartphone usage (lines slope down), the main effect of gender (the line for male is higher than the line for female), and the interaction between them (the negative slope is steeper for female than for male).

This is fine for helping you to interpret your model, but you would need to customise it before adding it into a report. We recommend using the non-centered predictors for the plot in your report. Like `afex_plot()` we introduced you to previously, `plot_model()` uses **ggplot2** in the background, and you can add further customisation by adding layers after the initial function.

```r
mod_wb2 <- lm(data = pry_cleandata,
              formula = "total_wellbeing ~ mean_hours * male")

plot_model(mod_wb2,
           type = "pred",
           terms = c("mean_hours","male")) +
  labs(x = "Daily hours smartphone use",
       y = "Predicted well-being score",
       color  = "Gender",
       title = "") +
  scale_color_discrete(labels=c("female","male")) +
  theme_bw()


## Scale for colour is already present.
## Adding another scale for colour, which will replace
## the existing scale.
```




### 9.3.5 Checking assumptions

The assumptions for multiple regression are the same as simple regression but there is one additional assumption, that of multicollinearity. This is the idea that predictor variables should not be too highly correlated.

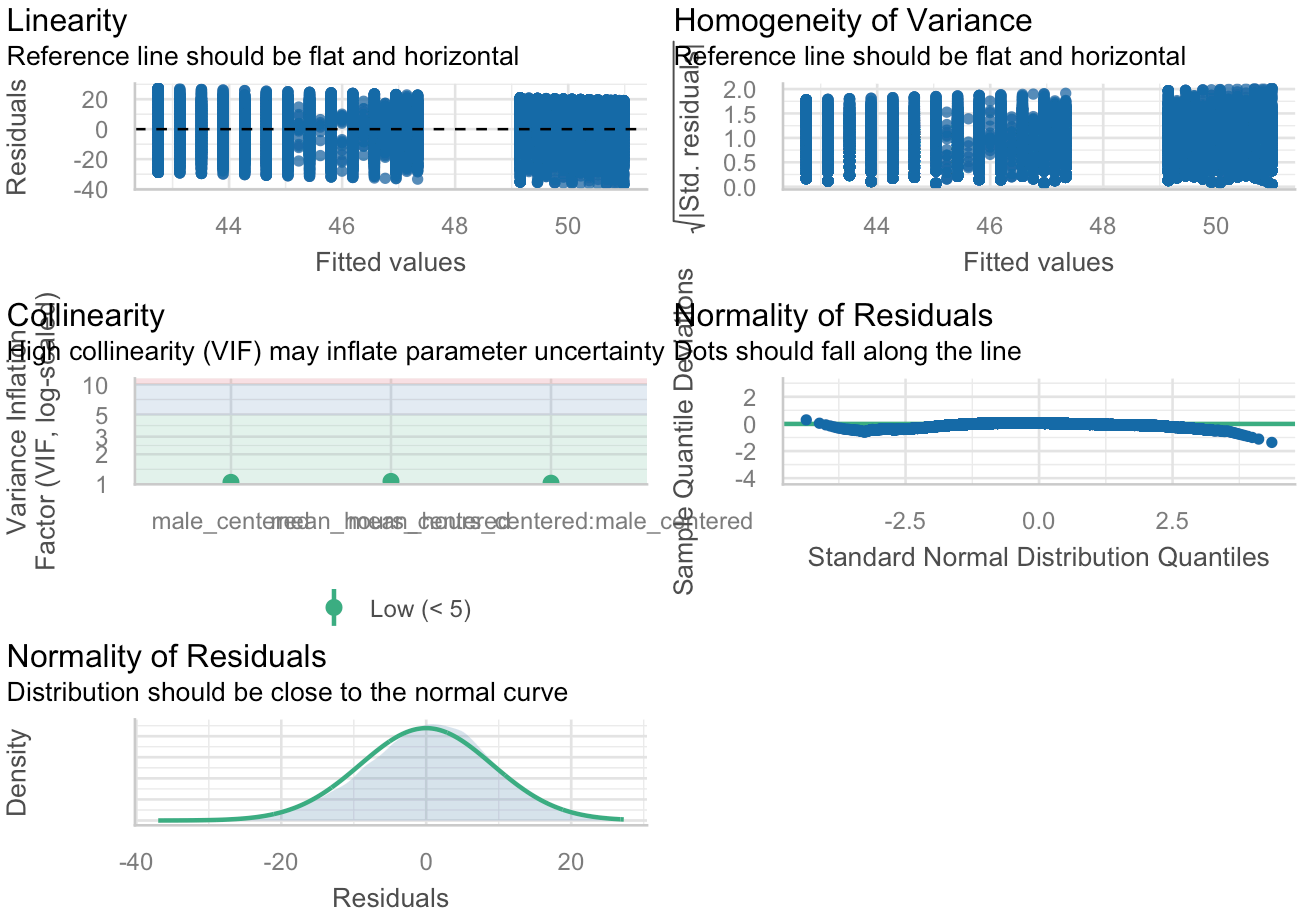
1. The outcome/DV is a interval/ratio level data.
2. The predictor variable is interval/ratio or categorical (with two levels).
3. All values of the outcome variable are independent (i.e., each score should come from a different participant).
4. The predictors have non-zero variance.
5. The relationship between outcome and predictor is linear.
6. The residuals should be normally distributed.
7. There should be homoscedasticity (homogeneity of variance, but for the residuals).

8. Multicollinearity: predictor variables should not be too highly correlated.

We know that we meet assumptions 1 - 4, and we will use check_model from the **performance** package.

One difference from when we used `check_model()` previously is that rather than just letting it run all the tests it wants, we are going to specify which tests to stop it throwing an error. A word of warning - these assumption tests will take longer than usual to run because it's such a big data set. The first line of code will run the assumption tests and save it to an object, calling the object name will then display the plots.

```
assumption_checks <- check_model(mod_wb,
                                 check = c("vif","qq","normality","linearity","homo

assumption_checks
```



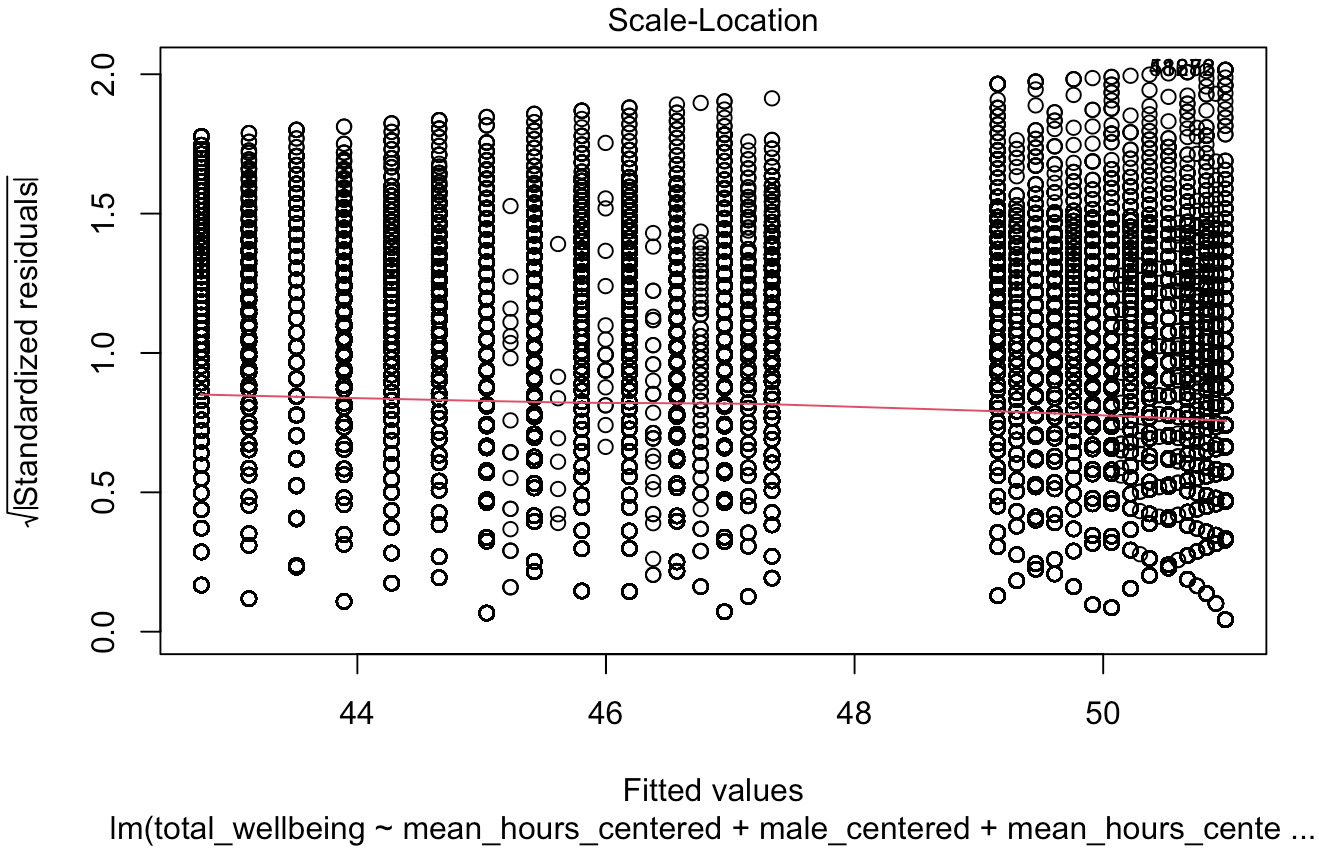
For assumption 5, linearity, we suspected from the group means that the relationship is linear, but the residual plot also confirms this.

For assumption 6, normality of residuals, the residuals look good in both plots and this provides an excellent example of why it's often better to visualise than rely on statistics. With a sample size this large, any statistical diagnostic tests will be highly significant as they are sensitive to sample size.

For assumption 8, multicollinearity, the plot also indicates no issues.

For assumption 7, homoscedasticity, the plot is missing the reference line. Fun fact, this took us [Bartlett & Toivo] several days of our lives and asking for help on social media to figure out. The reason the line is not there is because the data set is so large that is creates a memory issue. However, if you use the `plot()` version, it does show the reference line.

```
plot(mod_wb,
     which = 3)
```



It is not perfect, but the reference line is roughly flat to suggest there are no serious issues with homoscedasticity.

### 9.3.6 Reporting results

To report the outcomes of this test, we might write:

> Continuous predictors were mean-centered; binary predictors were deviation coded. The results of the regression indicated that the model significantly predicted well-being scores (F(3, 76403) = 2677, p < .001, adjusted $R^2$ = 0.10) accounting for approximately 10% of the variance. Girls had signifiantly lower well-being scores than boys (b = 5.03, p < .001), and total screen time was a significant negative predictor of well-being scores (b = -0.54, p < .001). Importantly, there was a significant interaction between screentime and gender (b = 0.46, p < .001); smartphone use was more negatively associated with wellbeing for girls than for boys.

32. https://psyteachr.github.io/quant-fun-v3/↩

33. Dawtry, R. J., Sutton, R. M., & Sibley, C. G. (2015). Why Wealthier People Think People Are Wealthier, and Why It Matters: From Social Sampling to Attitudes to Redistribution. *Psychological Science*, *26*(9), 1389–1400. https://doi.org/10.1177/0956797615586560↩

34. In some resources, you might see people enter the same model as `redistribution ~ 1 + fairness_satisfaction`. The `1 +` component explicitly tells R to fit an intercept, plus a slope from `fairness_satisfaction`. R includes an intercept by default, so you do not need to add it, but some people like to include it for clarity.↩

35. Note: not `summarize()!` ↩

36. Przybylski, A. K., & Weinstein, N. (2017). A Large-Scale Test of the Goldilocks Hypothesis: Quantifying the Relations Between Digital-Screen Use and the Mental Well-Being of Adolescents. *Psychological Science*, *28*(2), 204–215. https://doi.org/10.1177/0956797616678438↩

37. Quasi-continuous, the usual ordinal dilemma for Likert scale data.↩

38. Quasi-continuous, as only discrete values are possible. But there are a sufficient number of those discrete values that we can treat them as continuous.↩