Quantitative Methods 2, ZHAW

Jürgen Degenfellner

2025-02-27

Contents

Chapter 1

Preamble

This script is a continuation of the first one for Quantitative Methods 1 at ZHAW.

In the first part, we learned about the basics of probability theory, descriptive statistics, Bayesian statistics, and hypothesis testing.

In this script, we will dive into the basics of statistical modeling - a world of aesthetic wonder and surprises.

This script is a first draft as you are the first group to be working with it.

Please feel free to send me suggestions for improvements or corrections.

This **should be a collaborative effort** and will (hopefully) never be finished as our insight grows over time.

The script can also be seen as a pointer to great sources which are suited to deepen your understanding of the topics. Knowledge is decentralized, and there are many great resources out there.

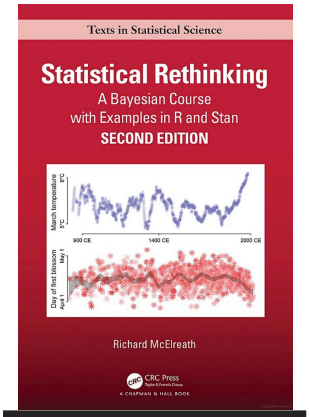
For the working setup with R, please see this and the following sections in the first script.

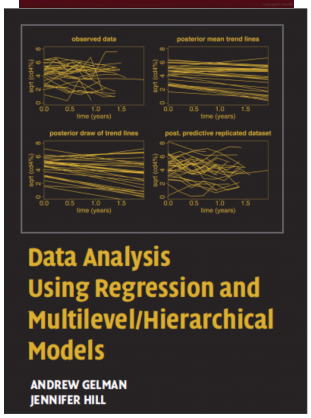
The complete code for this script can be found here.

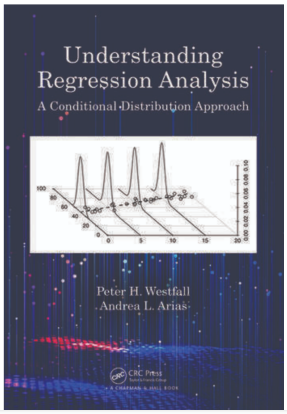
1.1 Books we will heavily borrow from are:

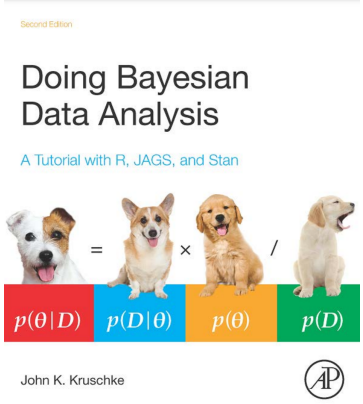
- (older online version is Free; current version in ZHAW Library) Statistical Rethinking, YouTube-Playlist: Statistical Rethinking 2023
- (Free) Understanding Regression Analysis: A Conditional Distribution Approach

- (Free online-access via ZHAW Library) Data Analysis Using Regression and Multilevel/Hierarchical Models
- (Free) Doing Bayesian Data Analysis









1.2 If you need a good reason to buy great books...

Think of the total costs of your education. You want to extract maximum benefit from it. In the US, an education costs a lot. In beautiful Switzerland, the tuition fees (if applicable) are nowhere near these figures. Costs you could consider are opportunity costs of not working. A comparison with both, a foreign education or opportunity costs, justifies the investment in good books. Or: The costs of all the good books of your education combined are probably less than an iPhone Pro.

Chapter 2

Introduction

2.1 What is statistical modeling and what do we need this for?

Typically, one simplifies the complex reality (and loses information) in order to make it better **understandable** (explainable), mathematically treatable and to make **predictions**.

Underlying our models, there are theories which should be falsifiable and testable. For instance, I would be really surprised if I pull up my multimeter and measure the voltage (V) and electric current (I) at a resistence (R) in a circuit and find that Ohm's law $V = IR$ is not true. This **law** can be tested over and over again and if one would find a single valid counterexample, the law would be falsified. It is also true that the law is probably not 100% accularate, but an extremely precise approximation of reality. Real-world measurements carry measurement errors and when plotting the data, one would see that the data points might not lie exactly on a straight line. This is not a problem.

A statistical model is a mathematical framework that represents the relationships between variables, helping us understand, infer, and predict patterns in data. It acts as a bridge between observed data and the real-world processes that generated them. In health research, where variability and uncertainty are inherent, statistical models are valuable tools for making sense of complex phenomena. You can watch this as short intro.

In QM1 we have already made testable predictions with respect to the probability of an event. In our 1000-researcher experiment we stated for instance, that the probability of obeserving 66 or more findings would be very unlikely. If such an event would occur (while not repeating the experiment many times), we would reconsider our model. Inexactly, we could have stated something like:

"We will not see more than 100 findings by chance." With respect to our multiple choice test at the end of QM1 we could predict: "We will not see a single person answering all questions correctly by chance in our lifetime (given the frequency of tests)." Note, that in this context, the word **predict** is used with respect to a future event (chance finding or chance passing of the test). As we will see, there does not necessarily have to be a temporal connection in order to *predict* something.

Depending on the task at hand, we would use different models. In any case, logical reasoning and critical thinking comes first, then comes the model. **It makes no sense to estimate statistical models just for the sake of it**.

All models are wrong, but some are useful. Or to quote George Box:

> "Since all models are wrong the scientist cannot obtain a 'correct' one by excessive elaboration. On the contrary following William of Occam he should seek an economical description of natural phenomena. Just as the ability to devise simple but evocative models is the signature of the great scientist so overelaboration and overparameterization is often the mark of mediocrity."

In my opinion, statistical modeling is an art form: difficult and beautiful.

One goal of this course is to improve interpretation and limitations of statistical models. They are not magical tools turning data into truth. Firstly, the rule gargabe in, garbage out (GABA) applies. Secondly, statistical models are based on data and their variability and have inherent limitations one cannot overcome even with the most sophisticated models. This is expressed for instance in the so-called bias-variance trade-off. You can't have it all.

2.1.1 Explanatory vs. Predictive Models

I can recommend reading this article by Shmueli et al. (2010) on this topic.

Statistical models serve different purposes depending on the research question. Two primary goals are **explanation** and **prediction**, and each requires a different approach:

Explanatory models (the harder of the two) focus on understanding causal relationships. These models aim to uncover mechanisms and answer **"why"** questions. For example:

- Does smoking increase the risk of lung cancer? **Yes**. (If you want to see what a large effect-size looks like, check out this study.)
- How large is the **effect** (causal) of smoking on lung cancer? **Large**.
- Does pain education and graded sensorimotor relearning improve disability (a question we ask in our Resolve Swiss project)?

Explanatory models are **theory-driven**, designed to test hypotheses. Here, one wants to understand the underlying mechanisms and the relationships between variables and hence often uses (parsimonious) models that are more interpretable, like linear regression.

Predictive models prioritize forecasting/predicting (future) outcomes based on patterns in the data. These models aim to answer **"what will happen?"** For instance:

- Gait analysis using Machine Learning (ML)?
- Skin cancer detection using neural networks?
- If I know the age, sex and weight of a person, can I predict his/her height? Can I predict the height better with the given covariates compared to just guessing by using the mean height of my sample for the next patient?

Predictive models are **data-driven**, often using complex algorithms to achieve high accuracy. Their success is measured using metrics like Root Means Square Error (RMSE), Area Unter the Curve (AUC), or **prediction error on new, unseen data**. Any amount of model complexity is allowed. One could for instance estimate a neural network ("just" another statistical model) with many hidden layers and neurons in order to improve prediction quality. Interpretability of the model weights is not a priority here.

While explanatory and predictive goals often complement each other, their differences highlight the importance of clearly defining the purpose of your analysis. In applied health research, explanatory models help identify causal mechanisms, while predictive models can guide real-world decisions by providing actionable forecasts. Together, they enhance both our understanding of phenomena and our ability to make informed decisions in complex environments.

2.1.2 Individual vs. Population Prediction

Another important distinction is between **individual vs. population** prediction. In the smoking example above, we can be very sure about the mean effects that smoking has on lung cancer. On an individual level, it is harder to predict the outcome. Nevertheless, individual predictions will be (notably) better than random guessing. We will discuss this in greater detail.

Statistical models are often much worse than one would naively expect, but they very often better than experts. If you are interested and want to boost your confidence in the predictive ability of statistical models, I recommend reading chapter 21 ("Intuitions vs. Formulas") of Daniel Kahneman's book "Thinking, Fast and Slow" (available in the ZHAW library).

2.1.3 Practical Use of Statistical Models

In my optinion, we should never be afraid to test our statistical models (as honestly as possible) against reality. We could for instance ask ourselves:

- "How much better does this model estimate an outcome than the arithmetic mean? (i.e., the linear model with just an intercept)"
- "How much better does this model classify than random guessing?"
- Is it worth the effort to collect data and estimate this model by using hundreds of hours of our time?

In some cases, these questions can be answered straightforwardly.

- In advertising (Google, Facebook, …), a couple of percentage points in prediction quality might make a difference of millions of dollars in revenue offsetting the statistitians salary by a large margin.
- Improved forecasts of a few percentage points in the stock market or just being slightly better than the average, will make you faboulously rich.
- Improved cancer forecasting might save lives, money and pain and is of course not only measured in financial gains.

2.1.4 Start at the beginning

What do we actually want to do in general? Very broadly speaking we want to: **describe** the association of variables to each other that carry variability. Hence, the relationship is not deterministic like

$$y = 2x + 3$$

but rather we need to "loosen up" the relationship to account for variability (in x and y). So, y and x are not fixed but aflicted with uncertainty. Depending on your philosophical view, you might say you want to find the "true" but unknown relationship (here, 2 and 3 are the true coefficients) between variables. This is what we do in simulation studies all the time: We know the true relationship, simulate data by adding variability and then try to estimate the true relationship we assumed in the first place. This is an **advantage** the pioneers of statistics did not have. We can simulate millions of lines of data at the click of a button. For some practical applications, we can get a really nice and complete answer to our question (for instance sample size for proportions).

So we are looking for a function f such that

$$\mathbf{Y} = f(\mathbf{X})$$

where

- $\mathbf{Y}$ is the "outcome", "dependent variable" or "response".
- $\mathbf{X}$ are the "predictors". $\mathbf{X}$ can be a single Variable x or many variables $x_1, x_2, \dots, x_p$.

It is important to be aware of the notation here: "Predict" does **not necessarily** mean that we can predict the value in the future. It merely means we estimate the value (or mean) of Y given X.

- This can be done at the same time points, known as **cross-sectional** analysis ("What is the maximum jumping height of a person given their age at a certain point in time, whereas both variables are measured at the same time?");
- or at different time points, known as **longitudinal analysis** ("What is the maximum jumping height of a person 10 years later (t_2) given their baseline health status at time t_1?").

The **simplest statistical model** would be the **mean model** where Y is "predicted" by a constant: $Y = c$ which (at least in the classical linear regression) turns out to be $c = \bar{x}$. This simple model is often surprisingly good, or, to put it in other words, models with more complexity are often not that much better.

2.2 A (simple) model for adult body heights in the Bayesian framework

As repetition, read the parts about Bayes statistics from QM1 again to refresh your memory about the Bayesian framework.

It's recommendable to read the beginning of the book Statistical rethinking (hint: the online-version of the book differs a bit from the paper-version) up until page 39 as well. We are not completely new to the topic of Bayes thanks to QM1. In the Bayesian setting we use (well argued for) prior knowledge about a parameter or effect and update this knowledge with new data.

We want to **start building our first model** right away.

Let's begin with the example in Statistical rethinking using data from the !Kung San people.

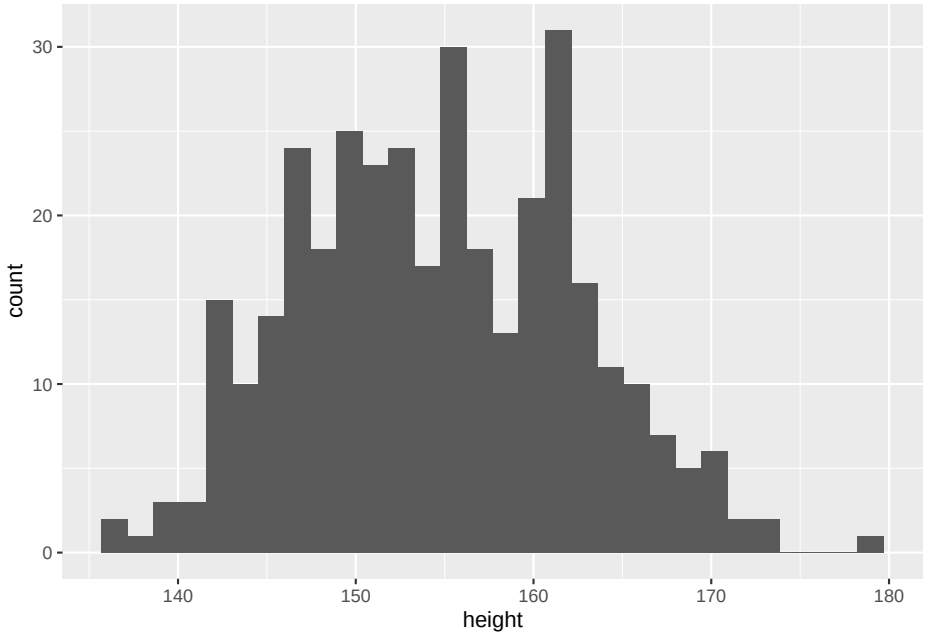
```
library(rethinking)
data("Howell1")
d <- Howell1
str(d)
```

```
## 'data.frame':    544 obs. of  4 variables:
##  $ height: num  152 140 137 157 145 ...
##  $ weight: num  47.8 36.5 31.9 53 41.3 ...
##  $ age   : num  63 63 65 41 51 35 32 27 19 54 ...
##  $ male  : int  1 0 0 1 0 1 0 1 0 1 ...
```

```
d2 <- d[d$age >= 18, ] # only adults
```

We want to model the adult height of the !Kun San people using prior knowledge (about the Swiss population) and data.

```
library(tidyverse)
d2 %>% ggplot(aes(x = height)) + geom_histogram()
```



Since we already have domain knowledge in this area, we can say that heights are usually normally distributed, or at least a mixture of normal distrubutions (female/male). We assume the following model:

$$h_i \sim \text{Normal}(\mu, \sigma)$$

As in QM1, we want to start with a Bayesian model and hence, we need some priors.

Since we are in Switzerland and just for fun, we use the mean of Swiss body heights as expected value for the **prior for the mean**. According to the link (Bundesamt für Statistik), the mean height of $n = 21,873$ people in the Swiss sample is 171.1 cm. We choose the same σ for the prior of the normal as in the book not to deviate too much from the example at hand.

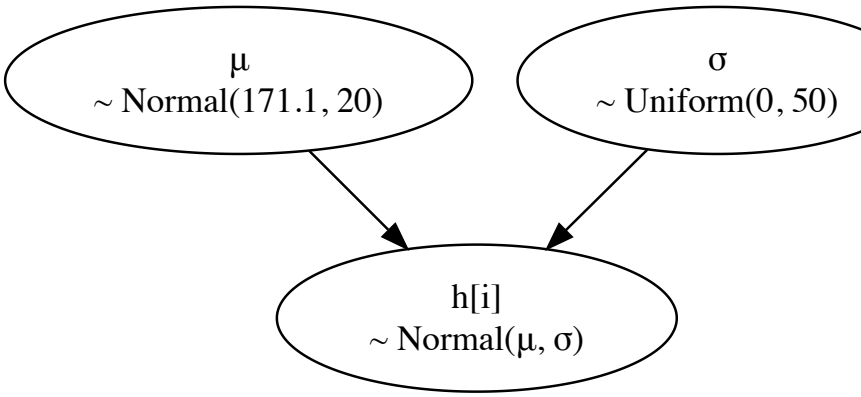
Next comes our **model definition in the Bayesian framework**, which I often find more intuitive than the Frequentist approach:

$$h_i \sim \text{Normal}(\mu, \sigma)$$

$$\mu \sim \text{Normal}(171.1, 20)$$

$$\sigma \sim \text{Uniform}(0, 50)$$

Description of the model definition: The heights are normally distributed with unknown mean and standard deviation. As our current knowledge about the mean height, we use a prior distribution for the mean (we do not know but want to estimate) by assuming the mean of a population we know and a standard deviation of 20 cm which allows are rather large range of possible values for μ (the unobserved population mean of the !Kung San people). σ (the unobserved standard deviation of the population of !Kun San people) is also unknown and a priori we restrict ourselves to values between 0 and 50 cm, whereas we assign equal plausibility to all values in this range (which can and should be critically discussed).



Vizualisation of the model structure:

Mind that there is a **conceptual difference** between the normal distribution of the heights and the normal prior distribution of the mean. The latter expresses our prior knowledge/insecurity about the unobserved mean of the normal distribution of the heights. The normal distribution of the heights says we expect the heights to be normally distributed but we do not know the parameters (μ

and σ) yet. We will estimate these parameters using prior knowledge and the data.

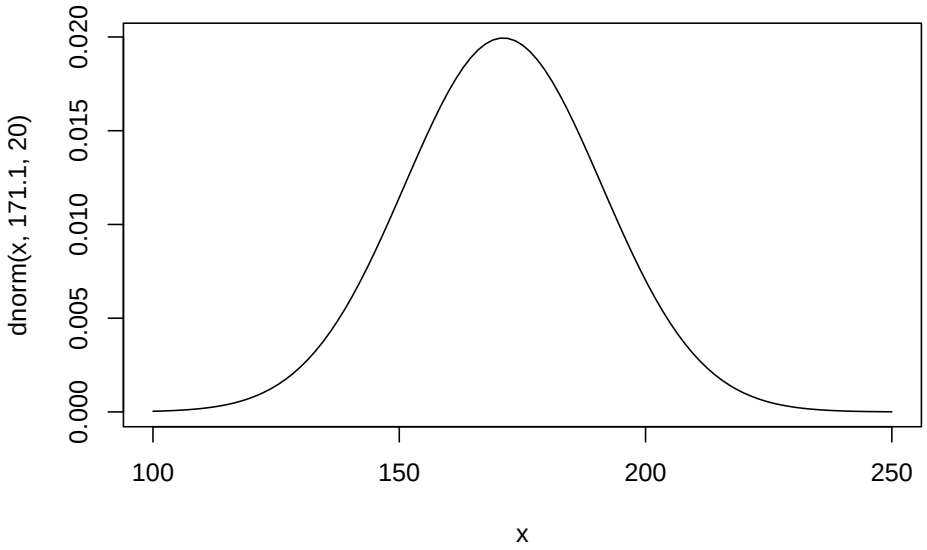
Of course we would not need the prior here due to the large sample size, but let's do it anyways for demonstration purposes. We are not completely uninformed about body heights and express our knowledge with the prior for μ. The 20 in the prior for the mean expresses our range of possible true mean values and aknowledge that there are a variety of different subpopulations with different means.

Using the Swiss data in the link one could estimate that the standard deviation of the heights from $21,873$ Swiss people is around is 25.67 cm (Exercise 1).

Remember, in the Baysian world, there is no **fixed but unknown** parameter, but instead we define a distribution over the unobserved parameter.

We **visualize the prior for** μ:

```
curve(dnorm(x, 171.1, 20), from = 100, to = 250)
```

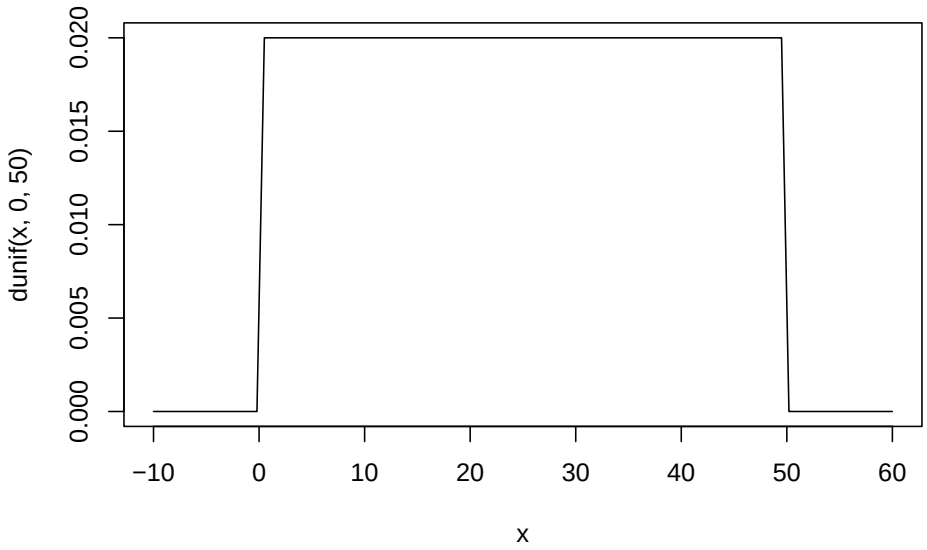


A wide range of population means is possible. Once could discuss this distribution and maybe further restrict it.

The **prior for** σ is uniform between 0 and 50 cm. This is a very wide prior and just constrains the values to be positive and below 50 cm. This could be stronger of course.

Visualization of the prior for σ:

```
curve(dunif(x, 0, 50), from = -10, to = 60)
```



Note, we didn't specify a prior probability distribution of heights directly, but once we've chosen priors for μ and σ, these imply a prior distribution of individual heights.

Without even having seen the **new data**, we can check what our prior (model) for heights would predict. This is important. If the prior already predicts impossible values, we should reconsider our priors and/or model.

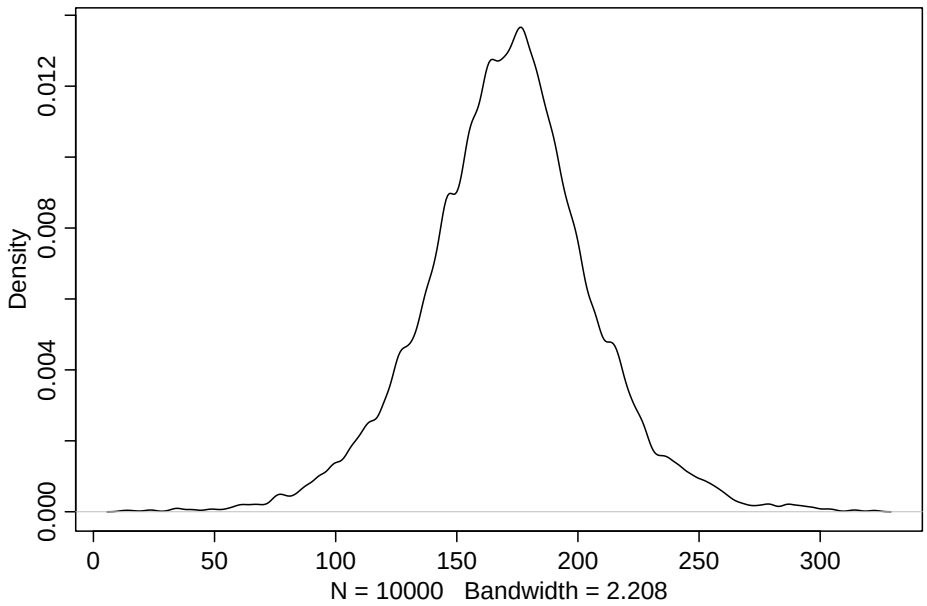
So, we simply draw μ and σ from the priors and then draw heights from the normal distribution using the drawn parameters.

Vizualisation of the prior for heights:

```
sample_mu <- rnorm(10^4, 171.1, 20)
sample_sigma <- runif(10^4, 0, 50)
prior_h <- rnorm(10^4, sample_mu, sample_sigma)
length(prior_h)
```

```
## [1] 10000
```

```
dens(prior_h)
```



The prior is not itself a Gaussian distribution, but a distribution of relative plausibilities of different heights, **before** seeing the data.

Note how we have created the model predictions for heights. We first drew μ and σ independently (there is no arrow between μ and σ) from the priors. Then we drew heights from the normal distribution using the drawn parameters. You just follow the model structure.

Now, there are a couple of different ways to estimtate the model incorporating the new data. For didactic reasons, grid approximation is often used (as in the books). For many parameters, this approach becomes more and more infeasible (due to combinatorial explosion).

We will skip that for now and use quadratic approximation instead which works well for many common procedures in applied statistics (like linear regression). Later, you'll probably use (or the software in the background) mostly Markov chain Monte Carlo (MCMC) sampling to get the posterior. Pages 39 and the following explain the 3 concepts grid approximation, quadratic approximation and MCMC.

In short, **quadratic approximation** assumes that our posterior distribution of body heights can be approximated well by a normal distribution, at least near the peak.

Please read the addendum to get a clearer picture of what a bivariate normal distribution is.

Using the `rethinking` package we can estimate the model using quadratic approximation. First, we define the model in the `rethinking` syntax (see R code 4.25 in the book).

```
library(rethinking)
flist <- alist(
  height ~ dnorm(mu, sigma),
  mu ~ dnorm(171.1, 20),
  sigma ~ dunif(0, 50)
)
```

Then we estimate/fit the model using quadratic approximation.

```
m_heights <- quap(flist, data = d2)
```

Now let's take a look at the fitted model: (Note: In the online-version of the book, they used the command `map` instead of `quap`.)

The `precis`function displays concise parameter estimate information (from the posterior) for an existing model fit.

```
precis(m_heights)
```

```
##             mean        sd       5.5%      94.5%
## mu    154.606914 0.4120045 153.948451 155.265377
## sigma   7.731518 0.2914035   7.265799   8.197237
```

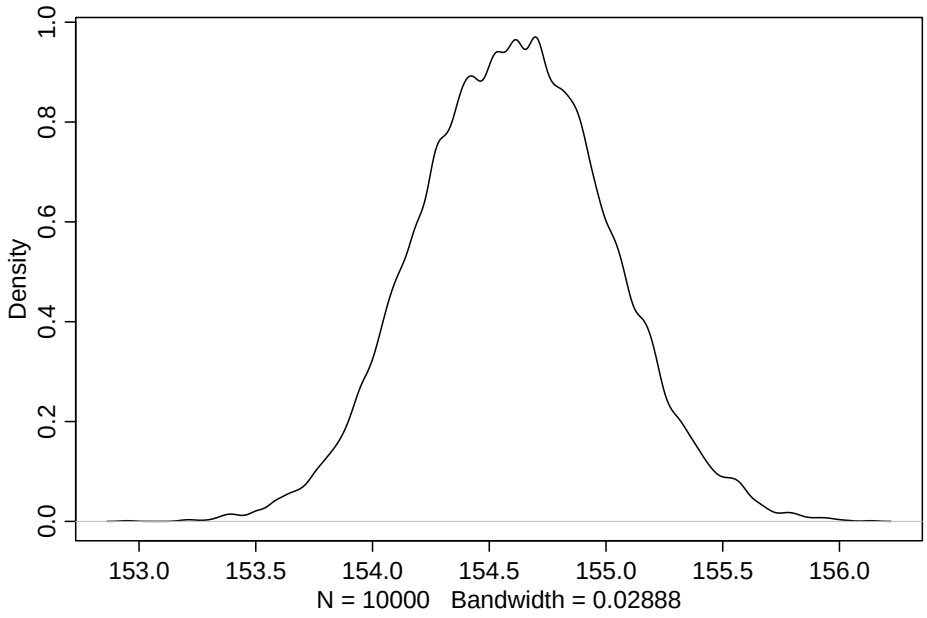
Above, we see the mean of the posterior for μ **and** σ; and a **89% credible interval** for those parameters. Note that these are rather tight credible intervals. We are rather confident that the mean is somewhere between 154 and 155 cm and the standard deviation is between 7 and 8 cm.

We can now plot the posterior distribution of the mean (μ) and the standard deviation (σ) separately by drawing from the posterior distribution.

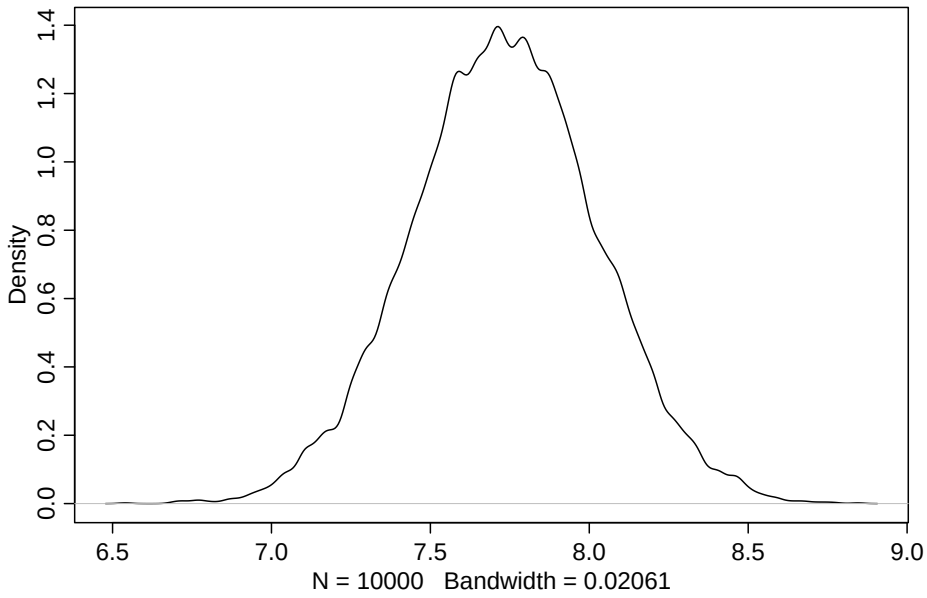
```
post <- extract.samples(m_heights, n = 10^4)
head(post)
```

```
##         mu    sigma
## 1 153.9637 8.072268
## 2 155.3763 7.439086
## 3 154.7934 8.019195
## 4 155.2723 7.596729
## 5 154.5083 7.871421
## 6 154.7321 8.115343
```

```
dens(post$mu)
```




```
dens(post$sigma)
```



Note, that **these samples come from a multi-dimensional posterior distribution**. In our case, we approximated the **joint** posterior distribution of μ

and σ with a bivariate normal distribution. They are not necessarily independent from each other, but in this case they are (see exercise 6). We know this from the prior definition above. μ and σ are both defined as normal respectively uniform distributions and by definition do not influence each other. This is also visible in the vizualisation of the model structure: There is no confounding variable or connection between those priors. One could think of a common variable Z that influences both μ and σ. This could be genetic similarity which could influence both μ and σ.

Let's verify that μ and σ are uncorrelated:

```
vcov(m_heights)
```

```
##                  mu        sigma
## mu    0.1697476878 0.0002156509
## sigma 0.0002156509 0.0849159731
```

gives you the variance-covariance matrix of the parameters of the posterior distribution. In the diagonal you see the variance of the parameters.

```
diag(vcov(m_heights))
```

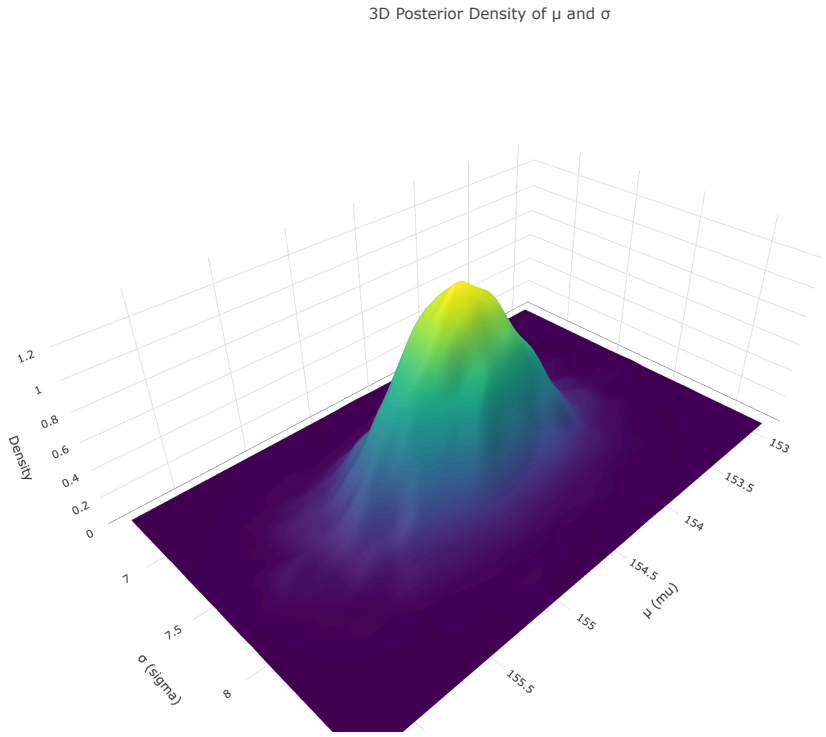
```
##         mu      sigma
## 0.16974769 0.08491597
```

And we can compute the correlation matrix easily:

```
cov2cor(vcov(m_heights))
```

```
##              mu     sigma
## mu    1.0000000 0.0017962
## sigma 0.0017962 1.0000000
```

Let's plot the posterior in 3D, because we **can**:



How beautiful ist that?

This shows how credible each combination of μ and σ is based on our priors and the data observed. The higher the mountain for a certain parameter combination, the more credible this combination is.

We see in the 3D plot, that the "mountain" is not rotated, indicating graphically that the parameters are independent from each other.

We also see in the correlation matrix, the correlation of the parameters is ~ 0. In the context of a joint normal distribution, this means that the parameters are also independent.

And, it is not an accident that the posterior looks like this. Using quadratic approximation, we used the bivariate normal distribution to **approximate** the posterior.

2.3 Classical approach for the simplest model

We have seen, how we could use prior knowledge to fit a very simple model for body heights of a population (!Kung San) in the Bayesian framework.

Now, let's start at the same point in the classical framework. Here, we do not use any prior knowledge, at least not that explicitely.

The classical approach to fit a regression line is the so-called **least squares method**.

There are hundreds of videos online explaining this method in great detail with animations. Maybe watch these videos later, when we add a predictor to the mean model, since most of instructional videos start at the simple linear regression using two parameters (intercept (β_0 or α) and slope (β_1)).

The **(simple mean-) model** is:

$$Y_i = height_i = c + \varepsilon_i$$

- for some $c \in \mathbb{R}$ and
- normally distributed errors $\varepsilon_i \sim \text{Normal}(0, \sigma)$.

The errors ε_i are on average zero and have a constant standard deviation of σ. So, we assume there is a fixed, but unknown, constant c that we want to estimate and we assume that there is a special sort of error in our model that is normally distributed. Sometimes there is a large deviation from the true c, sometimes there is a small deviation. On average, the deviations are zero and the **errors should also be independent from each other**:

$$\varepsilon_i \perp \varepsilon_j \text{ for } i \neq j$$

This means that just because I have just observed a large deviation from the true c does not mean, that the probability of a large deviation in the next observation is higher/lower. Note, that we cannot readily define different types of errors in the classical framework.

But what is c? We determine the shape of the model ourselves (constant model, or mean model) and then estimate the parameter c. By defining the shape of the model ourselves and imposing a distribution where we want to estimate the parameter of said distribution, we are in **parametric statistics**.

We choose the c which minimizes the sum of squared errors from the actual heights. This has the advantage that deviations upper and lower from the actual height are equally weighted. The larger the deviation the (quadratically) larger the penalty.

Why do we do that? Because, if the model assumptions (more on that later) are correct, the least squares estimator is a really good estimator. How good? Later…

We want to miminize the following function:

$$SSE \text{ (Sum of Squared Errors) } (c) = (height_1 - c)^2 + (height_2 - c)^2 + ... + (height_n - c)^2 =$$

$$= \sum_{i=1}^{n} (height_i - c)^2$$

The SSE is a function of c and we want to find the c that minimizes the function. Since it is a quadratic function, we can always find the minimum. We have learnt in school how to do this (hopefully): Take the derivative of the function and set it to zero. Solve for c and you have the c which yields the minimum of SSE(c).

Let's do that:

$$\frac{d}{dc} SSE(c) = 2(height_1 - c)(-1) + 2(height_2 - c)(-1) + ... + 2(height_n - c)(-1) =$$

$$= -2 \sum_{i=1}^{n} (height_i - c)$$

This should be zero for the minimum:

$$-2 \sum_{i=1}^{n} (height_i - c) = 0$$

$$\sum_{i=1}^{n} (height_i - c) = 0$$

$$\sum_{i=1}^{n} height_i - n \cdot c = 0$$

$$\hat{c} = \frac{1}{n} \sum_{i=1}^{n} height_i = \overline{height_i}$$

The hat over the c indicates that this is the estimated value of the true but unknown c. Everytime we estimate a parameter, we put a hat over it.

And voilà, we have estimated the parameter c of the model, which is just the sample mean of all the heights. In contrast to before, we did not put in a lot of prior knowledge, but just estimated the parameter from the data.

In R, we can do this easily:

```
mod <- lm(height ~ 1, data = d2)
summary(mod)
```

```
##
## Call:
## lm(formula = height ~ 1, data = d2)
##
```

```
## Residuals:
##      Min       1Q   Median       3Q      Max
## -18.0721  -6.0071  -0.2921   6.0579  24.4729
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 154.5971     0.4127   374.6   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 7.742 on 351 degrees of freedom
```

```
dim(d2)
```

```
## [1] 352   4
```

```
mean(d2$height) # same as the intercept
```

```
## [1] 154.5971
```

```
sd(d2$height) / sqrt(nrow(d2)) # standard error of the estimator
```

```
## [1] 0.4126677
```

```
# test-statistic for the intercept:
mean(d2$height) / (sd(d2$height) / sqrt(nrow(d2)))
```

```
## [1] 374.6285
```

```
# residual standard error:
sqrt(sum(mod$residuals^2) / (nrow(d2) - 1))
```

```
## [1] 7.742332
```

The `~1` means that there is just a so-called **intercept** in the model. There are **no covariates**, just the constant c. This is the simplest we can do. `lm` stands for linear model and with this base command in R we ask the software to do the least squares estimation for us.

Let's look at the **R-output** of the model estimation:

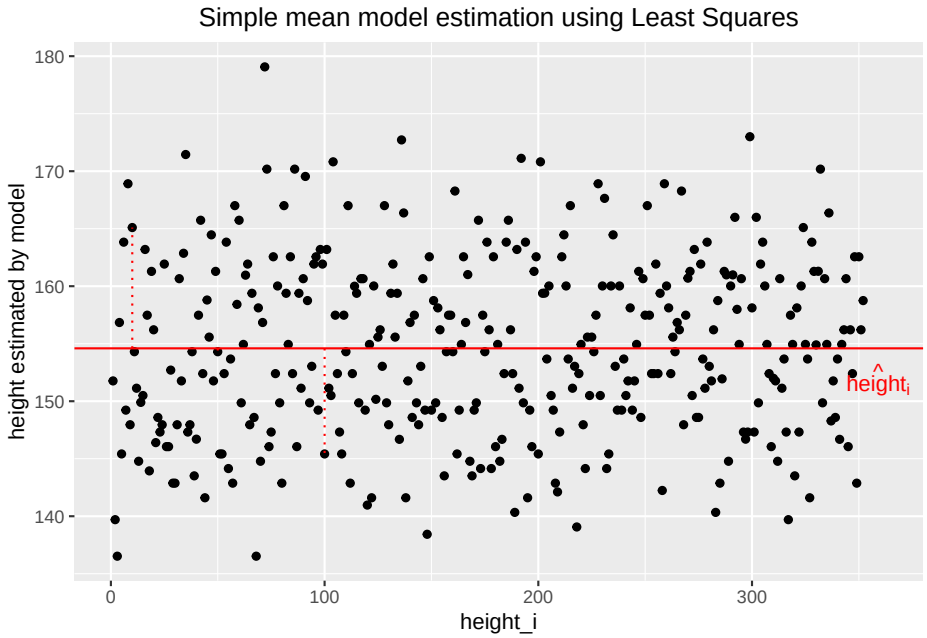
- `lm(formula = height ~ 1, data = d2)`: This is the model we estimated.

- `Residuals`: The difference between the actual height and the estimated height: $r_i = height_i - \hat{c}$. A univariate 5-point summary is given.
- `Coefficients`: The estimated coefficients of the model. In this case, there is just the intercept. We get the
  - `Std. Error` of the estimate, i.e. the standard error (SE) of the mean, which is (according to the Central Limit Theorem)

$$\frac{\sigma}{\sqrt{n}}$$

    and can be estimated by the sample standard deviation divided by the square root of the sample size. $\hat{SE} = \frac{s}{\sqrt{n}}$
  - the `t value` and the `Pr(>|t|)` which is the p-value of the (Wald-)test of the null hypothesis that the coefficient is zero (H_0 : intercept $= 0$). This is a perfect example of an absolutely useless t-test. Why? Because obviously (exercise 2) the population mean of body heights is not zero.
- `Residual standard error`: The standard deviation of the residuals $r_i = height_i - \hat{c}$. In this case identical with the sample standard deviation of heights (exercise 3). 351 degrees of freedom. There are 352 observations and 1 parameter estimated (intercept/mean). Hence, there are $352 - 1 = 351$ freely movable variables in the statistic of the sample standard deviation.

Let's look at the situation graphically:



Above, the heights are plotted against the index of the observation (the order does not matter). The variability of heights around the regression line (constant in this case) seems to stay constant, which is a good sign. We will call this **homoscedasticity** later. The dashed vertical red lines show two residuals (one > 0, the other < 0), the difference between the actual height and the estimated height. The model-estimated heights ($\widehat{heights}_i$) are all identical and nothing but the mean of all heights.

Peter Westfall explains in his excellent book a **conditional distribution approach** to regression, which is just what happens in the classical linear regression model. I **highly recommend** reading the first chapters.

What does this mean in this context? This means, that for every fixed value of the predictor (which we formally do not have), the distribution of the response is normal with mean $\hat{c}$ and standard deviation σ. Since, we do not have a predictor (apart from the intercept), we can say, that the distribution of the heights is normal with mean $\hat{c}$ and standard deviation σ. This is what we assumed in the model. It can also be directly seen in the formula:

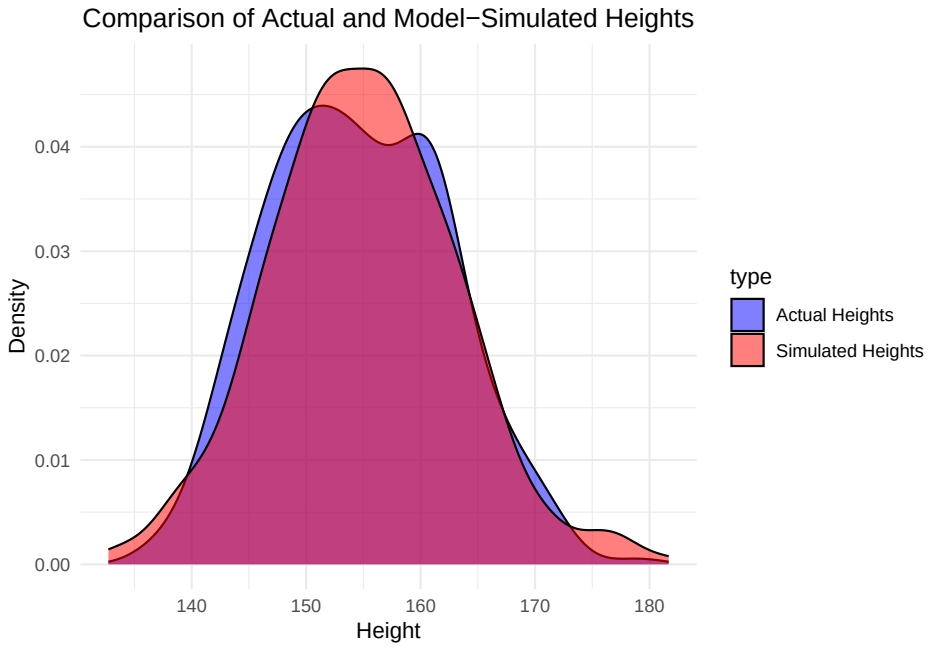
$$height_i = c + \varepsilon_i$$

If you add a normally distributed random variable (ε_i) to a constant (c), the result is a normally distributed. No surprise here.

We can also create new data (heights) from this model and compare the distributions of the actual data versus the model simulated data, since we have estimated the parameter c and the standard deviation σ.

```
c_hat <- mean(d2$height)
sigma_hat <- sqrt(sum(mod$residuals^2) / (nrow(d2) - 1))

# simulate heights from model
heights_sim <- rnorm(nrow(d2), c_hat, sigma_hat) # as many as in orig.

# Convert to data frames for plotting
df_actual <- data.frame(height = d2$height, type = "Actual Heights")
df_simulated <- data.frame(height = heights_sim, type = "Simulated Heights")

# Combine the datasets
df_combined <- rbind(df_actual, df_simulated)

ggplot(df_combined, aes(x = height, fill = type)) +
  geom_density(alpha = 0.5) +
  labs(title = "Comparison of Actual and Model-Simulated Heights",
       x = "Height",
       y = "Density") +
```

```
  scale_fill_manual(values = c("blue", "red")) +
  theme_minimal() +
  theme(plot.title = element_text(hjust = 0.5))
```



One can repeat the simulation and see how the distributions changes to get a feeling for the variability of the model/data.

2.4 Exercises

[E] Easy, [M] Medium, [H] Hard

(Some) solutions to exercises can be found in the git-repo here.

2.4.1 [E] Exercise 1

Use the Swiss body heights data to determine

- the 95% "Vertrauensintervall" for μ and
- calculate the standard deviation of the heights from $21,873$ Swiss people.
- Read the definition of the confidence interval in the footer of the table and explain why this is correct.

2.4.2 [E] Exercise 2

Why do we **not** need a hypothesis test to know that the population mean of body heights is not zero? Give 2 reasons.

2.4.3 [H] Exercise 3

Verify analytically that the `Residual standard error` is identical with the sample standard deviation of the heights in the simple mean model above.

2.4.4 [M] Exercise 4

Repeat the Bayesian and Frequentist estimation of the simple model using a different data set about chicken weights, which is included in R.

- Set useful priors for the mean and standard deviation of the model for the Baysian and the frequentist version considering your a priori knowledge about chicken weights.

2.4.5 [M] Exercise 5

- Try to understand/verify the code in the addendum below.
- Change the parameters $\mu = (\mu_X, \mu_Y)$ and Σ of the bivariate normal distribution and see how the 3D plot changes.
- Plot the scatter plot of the X and Y values and compare it to the 3D plot. Does this make sense?
- Why is the scatter plot elliptical?

2.4.6 [M] Exercise 6

Independence of two random variables X and Y implies that the correlation between them is zero. A correlation of zero does not necessarily imply independence.

- Verify this counterexample: For example, suppose the random variable X is symmetrically distributed about zero, and $Y = X^2$. Then Y is completely determined by X, so that X and Y are perfectly dependent, but their correlation is zero.

2.4.7 [H] Exercise 7

Go to the section aobut the bivariate normal distribution below.

- How do you get the correlation matrix from the covariance matrix (Σ)?

$$\Sigma = \begin{bmatrix} 0.75 & 0.5 \\ 0.5 & 0.75 \end{bmatrix}$$

  Hint: For the bivariate normal distribution, the correlation matrix is defined here.

2.5 Addendum

2.5.1 The bivariate normal distribution

As a refresher, you can look into the old QM1 script and read the chapter "4.7 Gemeinsame Verteilungen". Maybe this video also helps.

The bivariate normal distribution is a generalization of the normal distribution to two dimensions. Now, we look at the distribution of two random variables X and Y **at the same time**.

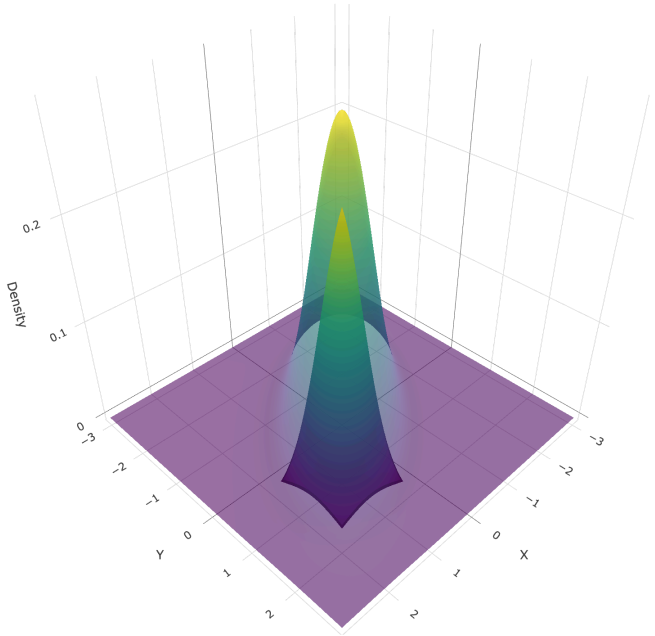
Instead of one Gaussian curve, we have a 3D curve. This curve defines how plausible different combinations of X and Y are.

Single points (like (3,6)) still have probability zero, because now the **volume** over a single point (x, y) is zero. The probability of a certain area is now the **volume** under the curve compared to the **area** under the density curve in the one-dimensional case.

Example: The following plot shows the density of a bivariate normal distribution of two variables X and Y with $\mu_X = 0$, $\mu_Y = 0$, $\sigma_X = 1$, $\sigma_Y = 1$ and $\rho = \frac{2}{3}$.

Below is the correlation matrix of the bivariate normal distribution.

```
##           [,1]      [,2]
## [1,] 1.0000000 0.6666667
## [2,] 0.6666667 1.0000000
```



If you move the plot around with your mouse, you see that there is a positive correlation between X and Y ($\rho = \frac{2}{3}$). This means that if X is above its mean, Y is also more likely to be above its mean. The variances of X and Y are both 1. That means, that if you cut through the plot in $X = 0$ or $Y = 0$, you see the same form of normal distribution. If you look at if from above, we have hihglighted the section on the surface over the area $X \in [0.5, 2]$ and $Y \in [0.5, 2]$. The volume over this area under the density curve is the probability of this area: $P(X \in [0.5, 2] \text{ and } Y \in [0.5, 2])$

Calculate with R this probability with R:

```r
# Load the mvtnorm package
library(mvtnorm)

# Define the parameters of the bivariate normal distribution
mu <- c(0, 0)                          # Mean
sigma <- matrix(c(0.75, 0.5, 0.5, 0.75), ncol = 2) # Covariance matrix

# Define the bounds of the square
highlight_x <- c(0.5, 2)
highlight_y <- c(0.5, 2)
# Calculate the probability using pmvnorm
pmvnorm(
```

```
  lower = c(highlight_x[1], highlight_y[1]),
  upper = c(highlight_x[2], highlight_y[2]),
  mean = mu,
  sigma = sigma
)
```

```
## [1] 0.1526031
## attr(,"error")
## [1] 1e-15
## attr(,"msg")
## [1] "Normal Completion"
```

Since we do not believe everything we are told, we rather check via simulation, if 0.1526 is a plausible value for the probability:

```
# Load necessary library
library(MASS)

# Define the parameters of the bivariate normal distribution
mu <- c(0, 0)                          # Mean
sigma <- matrix(c(0.75, 0.5, 0.5, 0.75), ncol = 2) # Covariance matrix
cov2cor(sigma)  # Convert covariance matrix to correlation matrix
```

```
##           [,1]      [,2]
## [1,] 1.0000000 0.6666667
## [2,] 0.6666667 1.0000000
```

```
# Define the bounds of the square
highlight_x <- c(0.5, 2)
highlight_y <- c(0.5, 2)

# Number of simulations
n_sim <- 10^4

set.seed(343434)
# Simulate bivariate normal samples
samples <- mvrnorm(n = n_sim, mu = mu, Sigma = sigma)

# Count how many samples fall within the square
inside_square <- sum(
  samples[, 1] >= highlight_x[1] & samples[, 1] <= highlight_x[2] &
  samples[, 2] >= highlight_y[1] & samples[, 2] <= highlight_y[2]
)
```

```
# Estimate the probability
inside_square / n_sim
```

```
## [1] 0.1557
```

Looks good.

Chapter 3

Simple Linear Regression

3.1 Simple Linear Regression in the Bayesian Framework

You can watch this video as primer.

We will now add one covariate/explanatory variable to the model. Refer to Statistical Rethinking "4.4 Linear prediction" or "4.4 Adding a predictor" as it's called in the online version of the book.

So far, our "regression" did not do much to be honest. The mean of a list of values was already calculated in the descriptive statistics section before and we have mentioned how great this statistic is as measure of location and where its weaknesses are.

Now, we want to model **how** body height and weight are **related**. Formally, one wants to *predict* body heights from body weights.

Here and in the Frequentist framework, we will see that it is **not the same** problem (and therefore results in a different statistical model) **to predict body weights from body heights or vice versa**.

We remember the following:

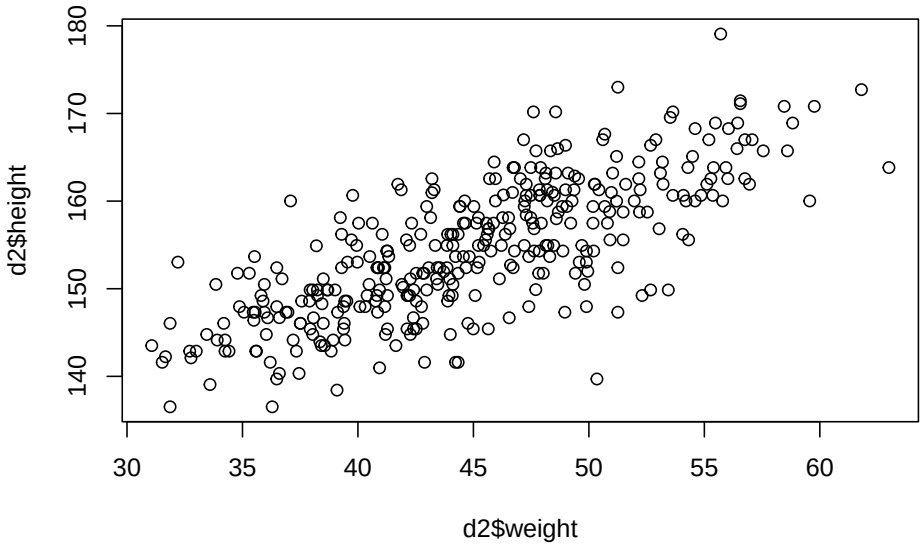
- Regress Y on X, which is equivalent to predict Y from X. We know X and want to predict Y.
- Regress X on Y, which is equivalent to predict X from Y. We know Y and want to predict X.

The word "predictor" is important here. It is a technical term and describes a variable that we know (in our case weight) and with which we want to "guess

as good as possible" the value of the dependent variable (in our case height). "As good as possible" means that we put a penalty on an error. The farer our prediction is away from the true value (y_i), the higher the penalty. And not only that, but if you are twice as far away from the true value, you should be penalized four times as much. This is the idea behind the squared error loss function and the core of the least squares method. What if we would punish differently, you ask? There are many loss functions one could use (for instance the Huber loss), maybe we will see some later. For now, we punish quadratically.

We **always** visualize the data first to improve our understanding. First comes descriptive statistics, then one can think about modeling.

```
plot(d2$height ~ d2$weight)
```



It's not often that you see such a clean plot. The scatterplot indicates a linear relationship between the two variables. The higher the weight, the higher the height; with some deviations of course and we decide that normally distributed errors are a good idea. This relationsip is neither causal, nor deterministic.

- It is not causal since an increase in weight does not necessarily lead to an increase in height, especially in grown-ups.
- It is not deterministic since there are deviations from the line. It if was deterministic, we would not need statistical modeling.

For simpler notation, we will call `d2$weight` x. $\bar{x}$ is the mean of x.

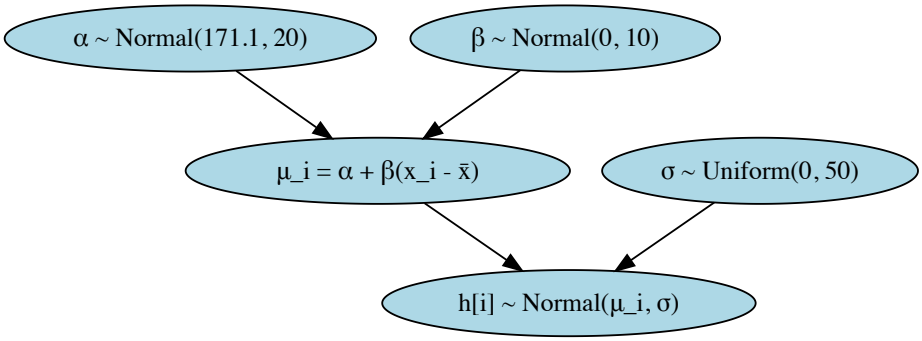
3.1.1 Model definition

Let's write down our **model** (again with the Swiss population prior mean):

$$
\begin{aligned}
h_i &\sim \text{Normal}(\mu_i, \sigma) \\
\mu_i &\sim \alpha + \beta(x_i - \bar{x}) \\
\alpha &\sim \text{Normal}(171.1, 20) \\
\beta &\sim \text{Normal}(0, 10) \\
\sigma &\sim \text{Uniform}(0, 50)
\end{aligned}
$$

Visualization of the **model structure**:

```
## file:////private/var/folders/pm/jd6n6gj10371_bml1gh8sc5w0000gn/T/Rtmp1MEKvh/file15e3f66b9858a/
```



There are now additional lines for the priors of α and β (compared to the simple mean model before). The model structure also shows the way to simulate from the prior. One starts at the top and ends up with the heights.

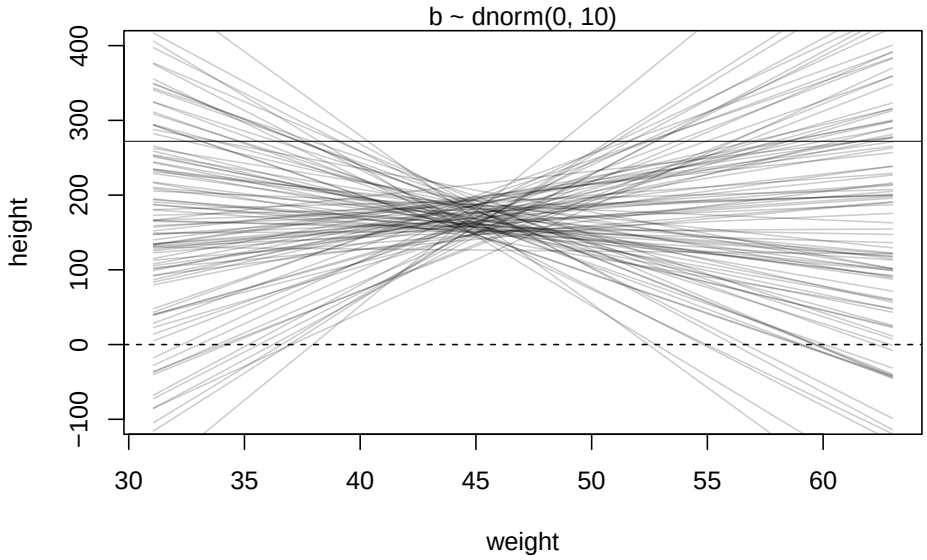
- h_i is the height of the i-th person and we assume it is normally distributed.
- μ_i is the mean of the height of the i-th person and we assume it is linearly dependent on the difference $x_i - \bar{x}$. Compared to the intercept model, a different mean is assumed for each person depending on his/her weight.
- α is the intercept and we use the same prior as before.
- β is the slope of the line and we use the normal distribution as prior for it, hence it can be positive or negative and how plausible each value is, is determined by that specific normal distribution. Note, that we could easily adapt the distribution to any distribution we like.
- The prior for σ is unchanged.
- $x_i - \bar{x}$ is the deviation of the weight from the mean weight, thereby **we center** the weight variable. This is a common practice in regression analysis. A value $x_i - \bar{x} > 0$ implies that the person is heavier than the average.

The linear model is quite popular in applied statistics and one reason is probably the rather straightforward interpretation of the coefficients: A one unit increase in weight is on average (in the mean) associated with a β unit increase/decrease (depending if β is > 0 or < 0) in height.

3.1.2 Priors

We want to plot our prior predictions to get a feeling **what the model would predict without seeing the data**. This is a kind of "sanity check" to see if the priors and the model definition are reasonable. Again, we just draw from the assumed distributions for α and β 100 times and draw the corresponding lines. Just as the model definition says.

```
set.seed(2971)
N <- 100  # 100 lines
a <- rnorm(N, 171.1, 20)
b <- rnorm(N, 0, 10)

xbar <- mean(d2$weight)

# start with empty plot
plot(NULL, xlim = range(d2$weight), ylim = c(-100, 400),
     xlab = "weight", ylab = "height")
abline(h = 0, lty = 2)  # horizontal line at 0
abline(h = 272, lty = 1, lwd = 0.5)  # horizontal line at 272
mtext("b ~ dnorm(0, 10)")

# Overlay the 100 lines
for (i in 1:N) {
  curve(a[i] + b[i] * (x - xbar),
        from = min(d2$weight), to = max(d2$weight),
        add = TRUE, col = col.alpha("black", 0.2))
}
```



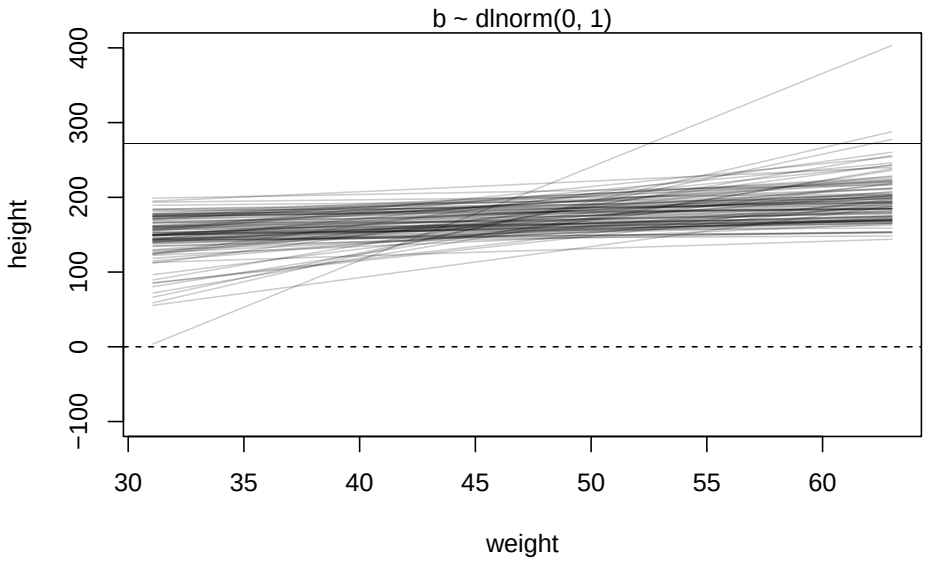
This linear relationship defined with the chosen priors seems rather non-restrictive. According to our priors, one could see very steeply rising or falling relationships between weight and expected heights. We could at least make the priors for the slope (β) non-negative. One possibility to do this is to use a log-normal distribution for the prior of β which can only take non-negative values.

$$\beta \sim \text{Log-Normal}(0, 1)$$

Lets plot the priors again.

```
set.seed(2971)
N <- 100  # 100 lines
a <- rnorm(N, 171.1, 20)
b <- rlnorm(N, 0, 1)

xbar <- mean(d2$weight)

plot(NULL, xlim = range(d2$weight), ylim = c(-100, 400),
     xlab = "weight", ylab = "height")
abline(h = 0, lty = 2)  # horizontal line at 0
abline(h = 272, lty = 1, lwd = 0.5)  # horizontal line at 272
mtext("b ~ dlnorm(0, 1)")

# Overlay the 100 lines
for (i in 1:N) {
  curve(a[i] + b[i] * (x - xbar),
```

```
        from = min(d2$weight), to = max(d2$weight),
        add = TRUE, col = col.alpha("black", 0.2))
}
```



This seems definitely more realistic. There is some sort of positive linear relationship between weight and expected height.

3.1.3 Fit model

Now, let's **estimate the posterior/fit the model** as before:

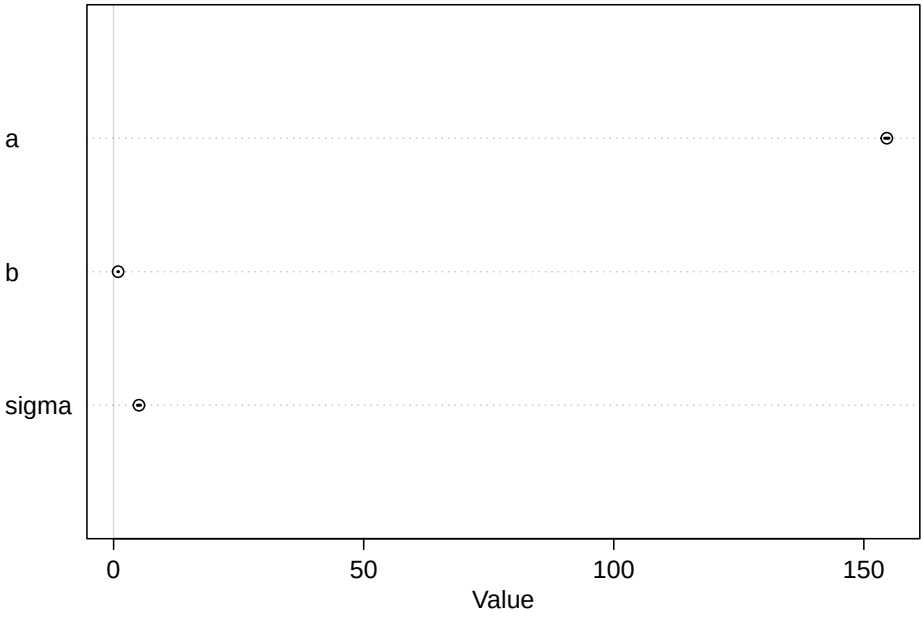
```
# load data again, since it's a long way back
library(rethinking)
data(Howell1)
d <- Howell1
d2 <- d[d$age >= 18, ]
xbar <- mean(d2$weight)
# fit model
mod <- quap(
    alist(
        height ~ dnorm(mu, sigma),
        mu <- a + b * (weight - xbar),
        a ~ dnorm(171.1, 100),
        b ~ dnorm(0, 10),
        sigma ~ dunif(0, 50)
    ) ,
data = d2)
```

Note that the model definition was now directly included in the `quap` function. Let's look at the **marginal distributions** of the parameters:

```
precis(mod)
```

```
##              mean         sd        5.5%       94.5%
## a     154.5972120 0.27033045 154.1651717 155.0292523
## b       0.9050131 0.04192754   0.8380048   0.9720214
## sigma   5.0718673 0.19115323   4.7663675   5.3773671
```

```
plot(mod)
```



Note, that the credible intervals in the plot of the coefficients are hardly noticable. The reason is that the intercept a is rather large, whereas b and sigma are comparativly small. The plot is scaled to the largest parameter.

We can improve this by centering the height variable as well. Or, we could standardize both weight and height. This would change the interpretation of the β parameter. It would then be the expected change in standard deviations when changing weight by one standard deviation. See exercise 12.

The analysis yields estimates for all our parameters of the model: α, β and σ. The estimates are the mean of the posterior distribution.

See exercise 2.

Interpretation of β: The mean of the posterior distribution of β is 0.9. A person with a weight of 1 kg more weight can be expected to be 0.9 cm taller.

A 89% credible interval for this estimate is $[0.83, 0.97]$. We can be quite sure that the slope is positive (of course we designed it that way too via the prior).

It might also be interesting to inspect the **variance-covariance matrix**, respectively the correlation between the parameters as we did before in the intercept model. Remember, these are the correlations of parameters in the multivariate (because three paremeters simulatenously) posterior distribution.

```
diag(vcov(mod))
```

```
##           a           b       sigma
## 0.073078550 0.001757918 0.036539558
```

```
round(cov2cor(vcov(mod)),2)
```

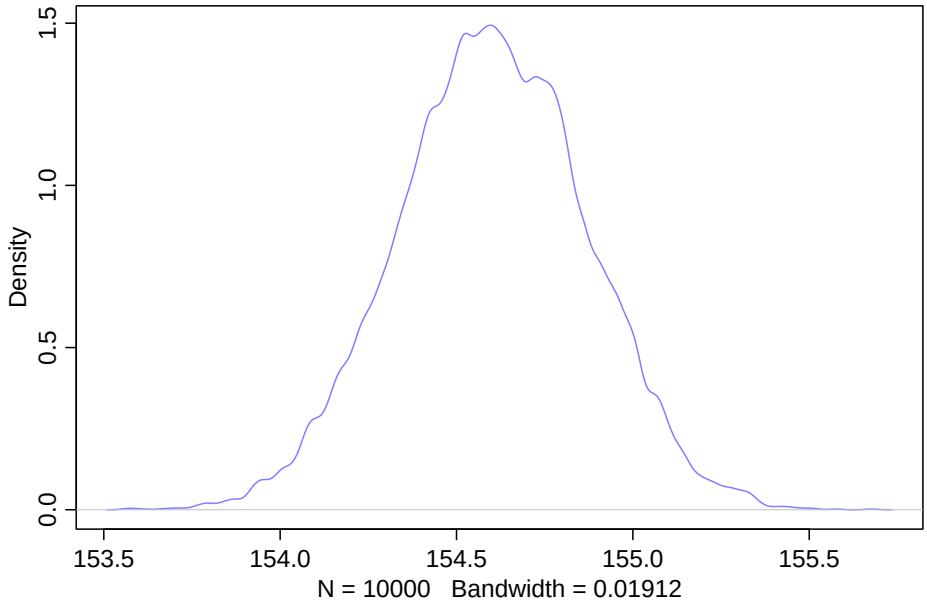
```
##       a b sigma
## a     1 0     0
## b     0 1     0
## sigma 0 0     1
```

- `diag(vcov(mod))` gives the variances of the parameters and
- `cov2cor(vcov(mod))`the correlations. As we can see the correlations are (near) zero. Compare to the graphical display of the model structure. There is no connection.

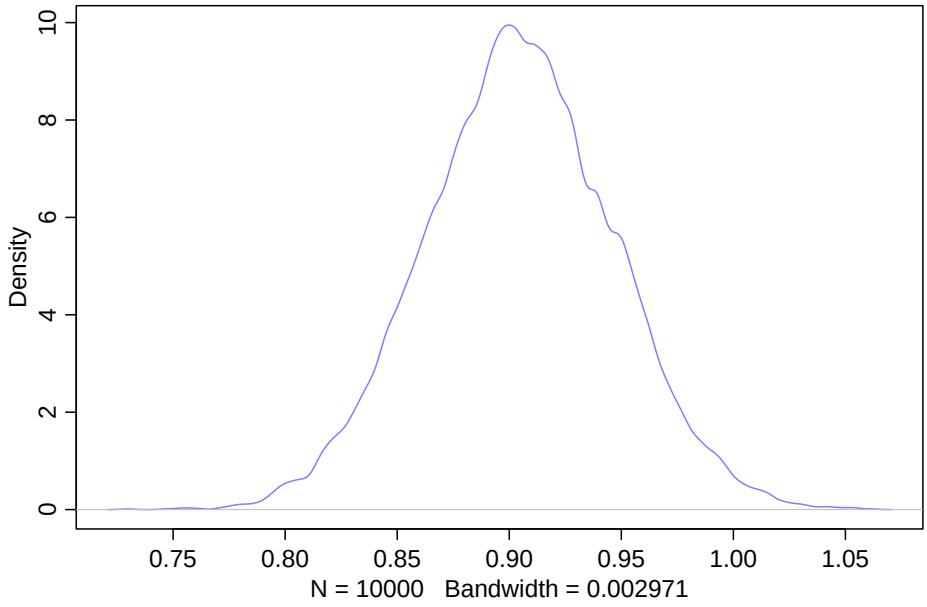
3.1.4 Result

Graphical end result of fitting the model: We plot the marginal posterior distributions of α and β, and also the raw data with the found regression line.

```
post <- extract.samples(mod)
dens(post$a, col = rangi2)
```

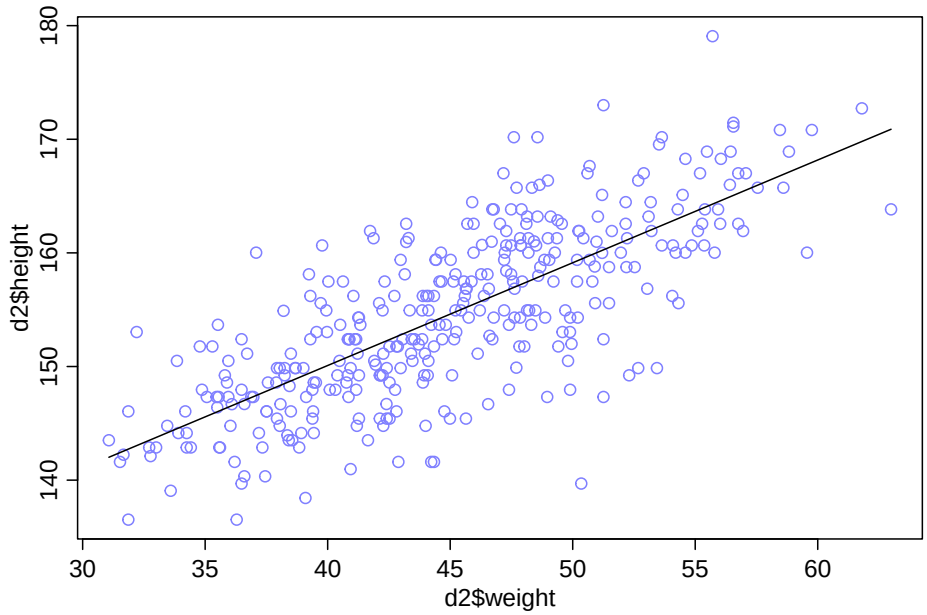



```
dens(post$b, col = rangi2)
```




```
# both posteror plots seem symmetric
# we use the mean as point estimate
# for a and b.
a_quap <- mean(post$a)
```

```
b_quap <- mean(post$b)

plot(d2$height ~ d2$weight, col = rangi2)
curve(a_quap + b_quap * (x - xbar), add = TRUE)
```



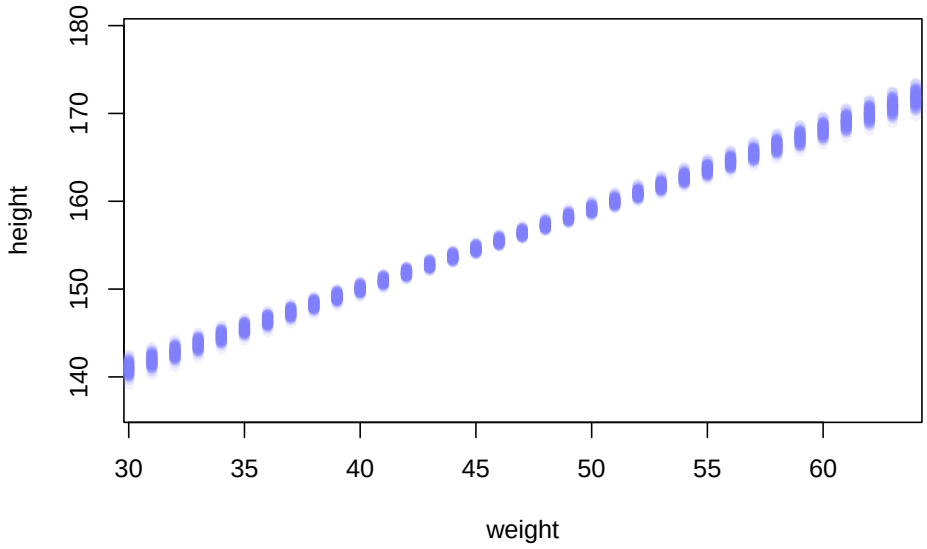
3.1.5 Credible bands

We could draw again and again from the posterior distribution and calculate the means like above. Plotting the regression lines with the respective parameters α, β would indicate the variability of the estimates. Note that we do not draw from the data (as one does in bootstrap resampling), but from the posterior distribution. The `link` function does this for us. It takes the posterior distribution we have just fit, samples α and β from it, calculates the mean and then samples from this normal distribution for the mean at a given weight. Refer to pages 98-106 in the current version of the book Statistical Rethinking for all details.

```
# Define a sequence of weights for predictions
weight.seq <- seq(from = 25, to = 70, by = 1)

# Use the model to compute mu for each weight
mu <- link(mod, data = data.frame(weight = weight.seq))
str(mu)
```

```
##  num [1:1000, 1:46] 138 136 137 136 137 ...
```

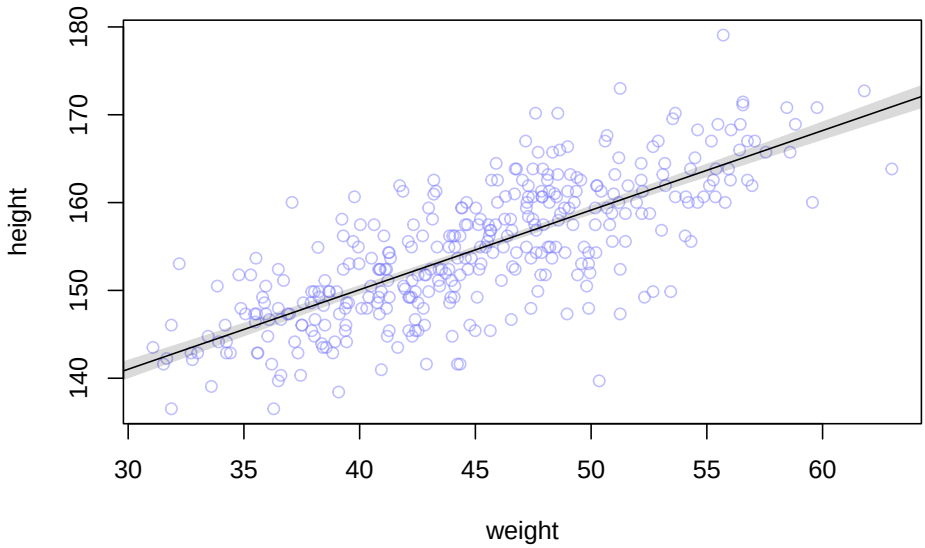
```
# Visualize the distribution of mu values
plot(height ~ weight, d2, type = "n")  # Hide raw data with type = "n"

# Loop over samples and plot each mu value
for (i in 1:100) {
  points(weight.seq, mu[i, ], pch = 16, col = col.alpha(rangi2, 0.1))
}
```



The `link` function fixes the weight at the values in `weight.seq` and draws samples from the posterior distribution of the parameters. We will do the analog thing in the Frequentist framework.

We can also draw a nice shade for the regression line:

```
# Summarize the distribution of mu
mu.mean <- apply(mu, 2, mean)
mu.PI <- apply(mu, 2, PI, prob = 0.89)
plot(height ~ weight, d2, col = col.alpha(rangi2, 0.5))
lines(weight.seq, mu.mean)
shade(mu.PI, weight.seq)
```



The function `PI` from the `rethinking` package calculates the 89% percentile interval for the mean of the height at a certain weight. The `apply` function apples this function to each columns (hence the `2`) of the matrix `mu`.

As we can see, we are pretty sure about the mean of height which we wanted to model in the first place.

Mean modeling is one thing, individual prediction is another. Given a certain weight of a person, what is the height of the same person? The first line in the model definition ($height_i \sim Normal(\mu_i, \sigma)$) tells us that a person's height is distributed *around* the mean (which linearly depends on weight) and is not necessary the mean itself.

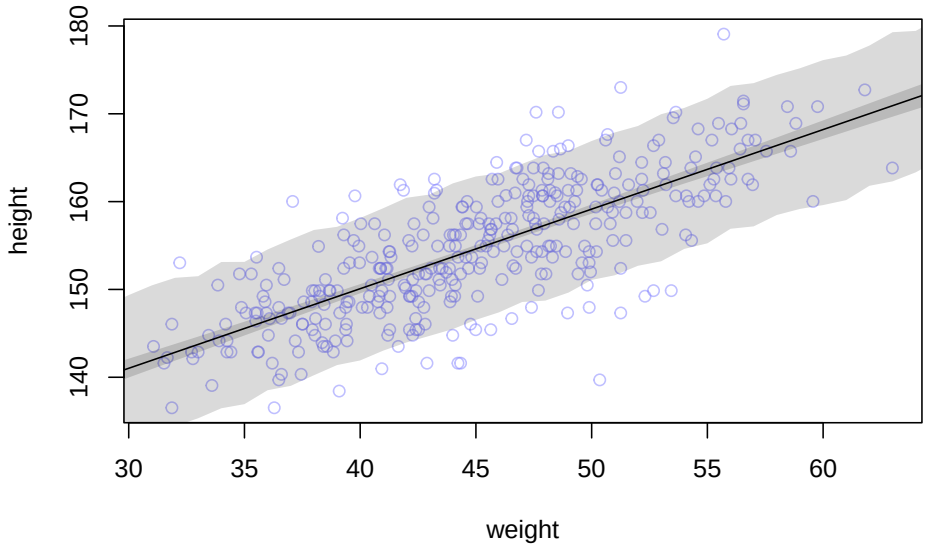
To get to an **individual prediction**, we need to consider the uncertainty of the parameter estimation *and* the uncertainty from the Gaussian distribution around the mean (at a certain weight). We do this with `sim`.

```
# Simulate heights from the posterior
sim.height <- sim(mod, data = list(weight = weight.seq))
str(sim.height)
```

```
##  num [1:1000, 1:46] 138 130 130 147 136 ...
```

```
# Compute the 89% prediction interval for simulated heights
height.PI <- apply(sim.height, 2, PI, prob = 0.89)

# Plot the raw data
plot(height ~ weight, d2, col = col.alpha(rangi2, 0.5))
```

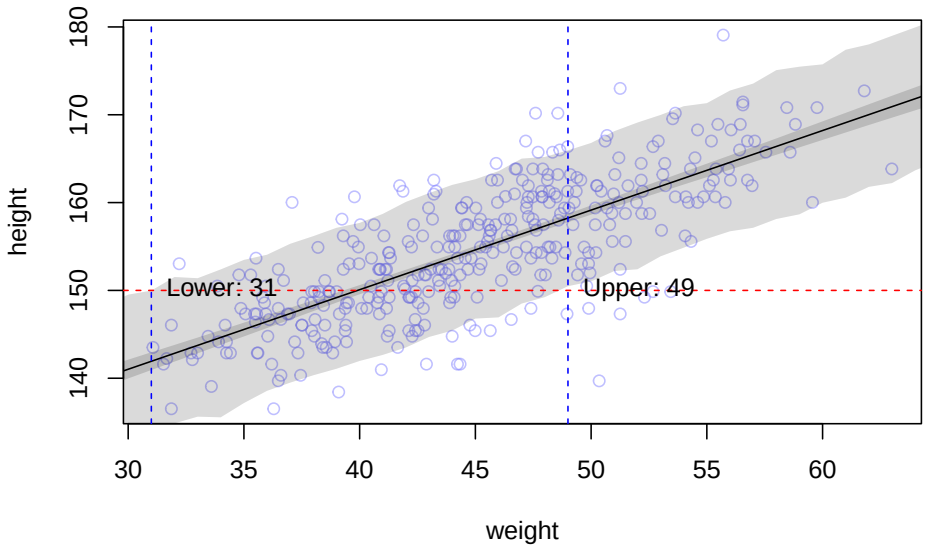
```
# Draw MAP (mean a posteriori) line
lines(weight.seq, mu.mean)

# Draw HPDI (highest posterior density interval) region for mu
shade(mu.PI, weight.seq)

# Draw PI (prediction interval) region for simulated heights
shade(height.PI, weight.seq)
```



Here, the `PI` function is applied to the simulated heights and calculates the 89% percentile interval for each weight.

The lighter and wider shaded region is where the model expects to find 89% of the heights of a person with a certain weight.

This part is **sometimes a bit desillusioning** when seen for the first time: Draw a horizontal line at 150 cm and see how many weights (according to the individual prediction) are compatible with this height. Weights from 30 to 50 kg are compatible with this height according to the 89% prediction interval:



The higher the credibility, the wider the interval, the wider the range of compatible weights. In our example, more than 60% of the weight-range are plausible to predict a height of 150 cm.

```
(50 - 30) / (range(d2$weight)[2] - range(d2$weight)[1])
```

```
## [1] 0.6265362
```

On the other hand: We did not model the relationship this way. We modeled **height depending on weight** and not the other way around. In the next chapter, we will regress weight on height (yes, this is the correct order) and see what changes.

3.1.6 Summary

- We have added a covariate (weight) to the simple mean model to predict height.
- We have centered the weight variable.
- We have defined and refined priors for the intercept and slope.
- We have estimated the posterior distribution of the parameters using quadratic approximation with `quap`.
- We have visualized the result.
- We have created credible bands for mean and individual predictions.

3.2 Simple Linear Regression in the Frequentist Framework

We will now do the same analysis in the Frequentist framework while introducing some foundational theory along the way. I recommend reading the first couple of chapters from Westfall.

3.2.1 Model definition

Our linear model is defined as:

$$h_i = \beta_0 + \beta_1 x_i + \varepsilon_i$$

where

- ε_i is the error term with $\varepsilon_i \sim N(0, \sigma^2), \forall i$
- β_0 is the unknown but fixed intercept
- β_1 is the unknown but fixed slope

3.2.1.1 Model Assumptions of the Classical Regression Model (Westfall, 1.7):

The first and **most important assumption** is that the data are produced probabilistically, which is specifically stated as

$$Y|X = x \sim p(y|x)$$

What does this mean?

- $Y|X = x$ is the random variable Y **conditional** on X being equal to x, i.e. the distribution of Y if we know the value of X (in our example the weight in kg). This is a nice image of what is meant here.
- $p(y|x)$ is the distribution of potentially observable Y given $X = x$. In our case above this was the normal distribution with mean μ_i and variance σ.

You can play with this shiny app to improve your understanding. It offers the option "Bedingte Verteilung anzeigen".

One always thinks about the so-called data generating process (Westfall, 1.2). How did the data come about? There is a process behind it and this process is attempted to be modeled.

Further assumptions:

- **Correct functional specification**: The conditional mean function $f(x) = \mathbb{E}(Y|X = x)$. In the case of the linear model, the assumption is $\mathbb{E}(Y|X = x) = \alpha + \beta x$. The **expectation** of Y (height) depends linearly on x (weight). This assumption is violated when the true relationship is not linear or the data at least suggest that it is not linear, like here.

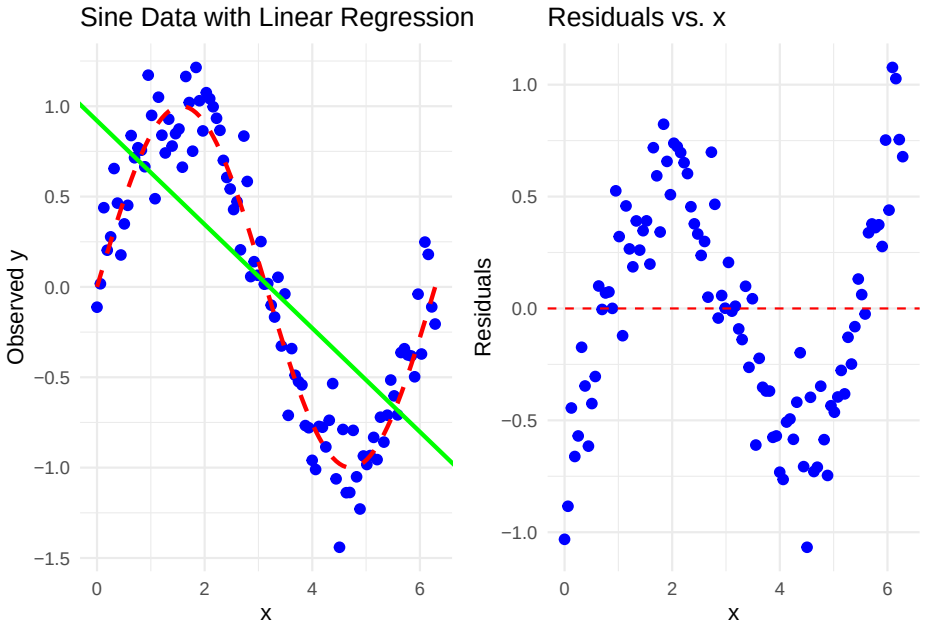
- **The errors are homoscedastic** (constant variance σ^2). This means the variances of all conditional distributions $p(y|x)$ are constant ($= \sigma^2$). This assumption is (for instance) violated if points are spreading out more and more around the regression line, indicating that the errors are getting larger.

- **Normality**. For the classical linear regression model all the conditional distributions $p(y|x)$ are normal distributions. It could well be, that the errors are not nicely normally distributed around the regression line, for instance if we have a lot of outliers upwards and the distribution is skewed, like here (Figure 5.20).

- The **errors are independent** of each other. The potentially observable $\varepsilon_i = Y_i - f(\mathbf{x_i},)$ is uncorrelated with $\varepsilon_j = Y_j - f(\mathbf{x_j},)$ for $i \neq j$. This assumption is violated if the errors are correlated, here is an example: The true data comes from a sine curve and we estimate a linear model (green), which does not fit the data well (left plot). The residuals plot shows clear patterns (right plot) and indicates that the errors are correlated. Specifically, the errors around $x = 2$ and $x = 4$ are negatively correlated (see exercise 6).

```
##
## Attaching package: 'patchwork'
```

```
## The following object is masked from 'package:MASS':
##
##     area
```



In the case above, the errors are **not conditionally independent**. If we condition on $X = 2$ and $X = 4.5$, the errors are correlated ($r \sim -0.3$), which they should not be.

These assumptions become clearer as we go along and should be checked for every model we fit. They are not connected, they can all be true or false. The question is not "Are the assumptions met?" since they never are exactly met. The question is **how** "badly" the assumptions are violated?

Remember, **all models are wrong, but some are useful**.

In full, the classical linear regression model can be written as:

$$Y_i|X_i = x_i \sim_{independent} N(\beta_0 + \beta_1 x_{i1} + ... \beta_k x_{ik}, \sigma^2)$$

for $i = 1, ..., n$.

3.2.2 Fit the model

Again, we fit the model using the least squares method. For a neat animated explanation, visit this video. There are literally hundreds of videos on the topic. Choose wisely. Not all are good. If in doubt, use our recommended books as reading materials. This is the most reliable source. A hint along the way: Be very sceptical if you ask GPT about information, although for this special case one has a good chance of getting a decent answer due to the vast amount of training data.

One has to minimize the sum of squared differences between the true heights and the model-predicted heights in order to find β_0 and β_1.

$$SSE(\beta_0, \beta_1) = \sum_{i=1}^{n} (y_i - (\beta_0 + \beta_1 x_i))^2$$

We omit the technical details (set derivative to zero and solve the system) and give the results for β_0 and β_1:

$$\hat{\beta}_0 = \bar{y} - (\hat{\beta}_1 \bar{x}),$$
$$\hat{\beta}_1 = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^{n}(x_i - \bar{x})^2} = \frac{s_{x,y}}{s_x^2} = r_{xy}\frac{s_y}{s_x}.$$

where:

- r_{xy} is the sample correlation coefficient between x and y
- s_x and s_y are the uncorrected sample standard deviations of x and y
- s_x^2 and s_{xy} are the sample variance and sample covariance, respectively

Interpretation of $\hat{\beta}_0$ and $\hat{\beta}_1$: see exercise 3.

Let's **use R again to solve the problem**:

```
library(rethinking)
data(Howell1)
d <- Howell1
d2 <- d[d$age >= 18, ]
mod <- lm(height ~ weight, data = d2)
summary(mod)
```

```
##
## Call:
## lm(formula = height ~ weight, data = d2)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -19.7464  -2.8835   0.0222   3.1424  14.7744
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 113.87939    1.91107   59.59   <2e-16 ***
## weight        0.90503    0.04205   21.52   <2e-16 ***
## ---
```

```
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 5.086 on 350 degrees of freedom
## Multiple R-squared:  0.5696, Adjusted R-squared:  0.5684
## F-statistic: 463.3 on 1 and 350 DF,  p-value: < 2.2e-16
```

Interpretation of R-output:

- `Call`: The model that was fitted.
- `Residuals`: $r_i = height_i - \widehat{height}_i$. Differences between true heights and model-predicted heights.
- `Coefficients`: Estimated for β_0 and β_1. We call them $\hat{\beta}_0$ and $\hat{\beta}_1$.
  - `Estimate`: The (least squares) estimated value of the coefficient.
  - `Std. Error`: The standard error of the estimate.
  - `t value`: The value of the t-statistic for the (Wald-) hypothesis test $H_0 : \beta_i = 0$.
  - `Pr(>|t|)`: The p-value of the hypothesis test.
- `Residual standard error`: The estimate of σ which is also a model parameter (as in the Bayesian framework).
- `Multiple R-squared`: The proportion of the variance explained by the model (we will explain this below).
- `Adjusted R-squared`: A corrected version of the R^2 which takes into account the number of predictors in the model.
- `F-statistic`: The value of the F-statistic for the hypothesis test: $H_0 : \beta_1 = \beta_2 = \cdots = \beta_k = 0$. Note, the alternative hypotheses to this test is that *any* of the β_i is not zero. If that is the case, the model explains more than the mean model with just β_0.

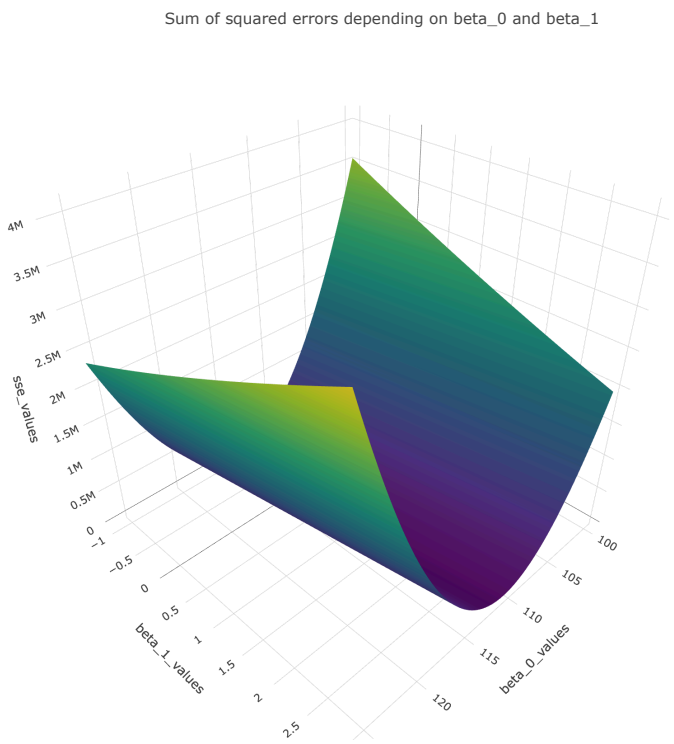
We could also **solve the least squares problem graphically**: We want to find the values of β_0 and β_1 that minimize the sum of squared differences which can be plotted as 3D function. Since the function is a sum of squared terms, we should expected a paraboloid form. All we have to do is to ask R which of the coordinates (β_0, β_1) minimizes the sum of squared errors. The result confirmes the results from the `lm` function. The dot in red marks the spot (Code is in the git repository):

—COMPILE CODE AT DEOPLOYMENT (takes a bit)—

```
## Intercept beta_0:  113.8794
```

```
## Slope beta_1:  0.9050291
```

```
## file:////private/var/folders/pm/jd6n6gj10371_bml1gh8sc5w0000gn/T/Rtmp1MEKvh/file15e3f286e871c/
```



3.2.3 Confidence Intervals of coefficients (Frequentist)

You can get CI's conveniently with the `confint` function:

```
confint(mod, level = 0.96)
```

```
##                    2 %        98 %
## (Intercept) 109.939864 117.8189232
## x             0.818351   0.9917072
```

Remember, these are Frequentist confidence intervals.

If one repeats the experiment many times, the true but unknown value of the parameter will be in the interval in 96% of the cases.

We can also use the simple bootstrap. The advantage of this technique is that we can basically always use it, no matter how compliated the estimator is. We do not need formulae. We simply

- create 1000 bootstrap samples,
- fit the model,

- store the coefficients.
- The 2% and 98% quantiles of the coefficients constitute the 96% bootstrap confidence interval.

```
set.seed(123)
n <- nrow(d2)
B <- 1000
boot_coefs <- matrix(NA, nrow = B, ncol = 2)
for (i in 1:B) {
  boot_idx <- sample(1:n, replace = TRUE)
  boot_mod <- lm(height ~ weight, data = d2[boot_idx, ])
  boot_coefs[i, ] <- coef(boot_mod)
}
#head(boot_coefs)
t(apply(boot_coefs, 2, quantile, c(0.02, 0.98)))
```

```
##               2%         98%
## [1,] 110.1862455 117.4516455
## [2,]   0.8229982   0.9859997
```

The CIs are quite similar to the ones from the `confint` function.

In the Bayesian setting, we used the centered weight variable. Let's to this here too for comparison and use 89% coverage probability.

```
d2$weight_centered <- d2$weight - mean(d2$weight)
mod_centered <- lm(height ~ weight_centered, data = d2)
#summary(mod_centered)
confint(mod_centered, level = 0.89)
```

```
##                       5.5 %      94.5 %
## (Intercept)     154.162715 155.0314698
## weight_centered   0.837658   0.9724002
```

Compare with `precis` from the Bayesian model:

```
##              mean         sd       5.5%       94.5%
## a     154.5972131 0.27033041 154.165173 155.0292533
## b       0.9050133 0.04192753   0.838005   0.9720216
## sigma   5.0718667 0.19115317   4.766367   5.3773663
```

We are glad to see that both analyses align really nicely.

3.2.4 ANOVA (Analysis of Variance)

A non-obvious and very useful finding is that the total variability (SST) in the data (our heights) can be **decomposed** (or analysed) into two parts:

- The variability explained by the model (the regression line): SSR
- The variability not explained by the model (the residuals): SSE

Sum of Squares in Total = Sum of Squares from Regression+Sum of Squared Errors

$$SST = SSR + SSE$$

$$\sum_{i=1}^{n}(y_i - \bar{y})^2 = \sum_{i=1}^{n}(\hat{y}_i - \bar{y})^2 + \sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

If you are interested in the details, check out this.

This video explains the concept nicely.

Let's visualize our regression result:

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

The blue dotted line is the distance from the mean to the point (total variance), the red dotted line is the distance from the mean to the regression line (explained variance) and the green dotted line is the distance from the regression line to the point (unexplained variance). We see that it adds up. I find this fact quite fascinating. One finds additivity by considering not determinisic values, but variances. Thank you Ronald Fisher.

See exercise 13.

3.2.5 R^2 - Coefficient of Determination

R^2 is the **amount of variance explained by the model**. You can also read Westfall 8.1.

As you can see above, the total variance (SST) of our outcome (height) can be decomposed into two parts: the variance explained by the model (SSR) and the variance not explained by the model (SSE).

Maybe the most intuitive definition of R^2 is:

$$R^2 = \frac{SSR}{SST} = \frac{SST - SSE}{SST} = 1 - \frac{SSE}{SST}$$

The value is between 0 and 1. The higher the value, the more variance is explained. But be cautious. Depending on the context, a really high R^2 is not necessarily a good thing. With the data we are working with, it could easily hint towards an error. If we are near 1, all points in the simple linear regression model are on the line. If we are near 0, the model does not explain much of the variance and we would see "noise with no slope" in the scatterplot (exercise 4). The normal R^2 can be found in the R output under `Multiple R-squared`.

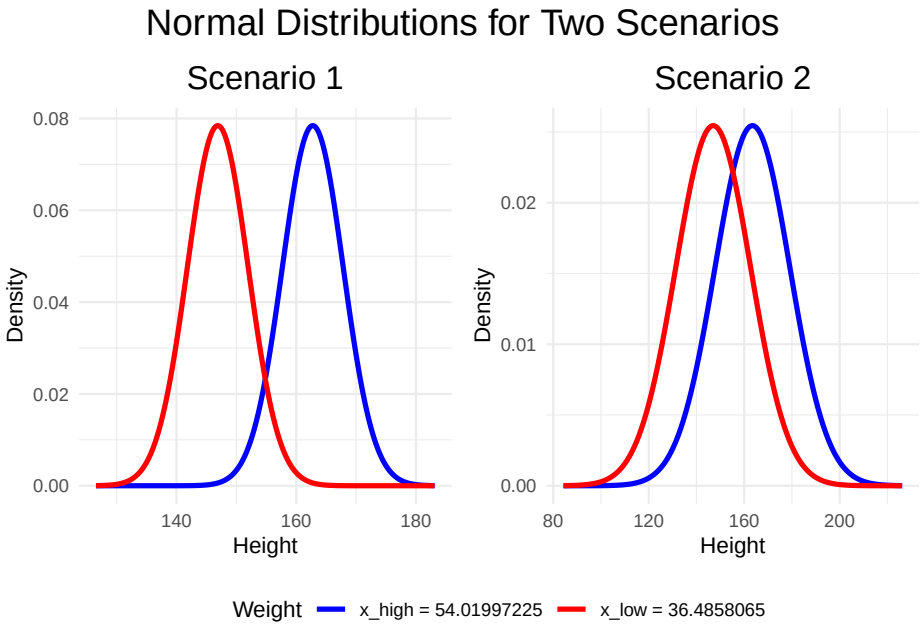
If you add a lot of variables to your regression model, you can get an R^2 of 1. The R^2 will never decrease when adding more variables. We will verify this when we have more than 2 explanatory variables. As a non-formal explanation for this: In the Sum of Squares Errors (SSE), if you add more covariates (β_2, β_3), you have more freedom to choose values that minimize the number that will be squared. Simple regression is just a special case of multiple (more than one predictor) regression with $\beta_2 = \beta_3 = \cdots = 0$. Hence, you will definitely not be worse off with regards to SSE when using more covariates. A smaller SSE implies a larger SSR (sum constraint; SST remains constant) and hence a larger R^2. If you have as many explanatory variables as data points, you can get an R^2 of 1. This is overfitting at its "best" (which we want to avoid). You would just get a value for each data point by setting all other β_i to zero and $\beta_i = \frac{y_i}{x_i}$. Since we want to find the unterlying process, we want to avoid this.

Although not perfect, one way to mitigate the influence of "too many" variables on R^2 is to use the adjusted R^2, which an also be found in the R output (`Adjusted R-squared`).

3.2.5.1 Seperating property of regression due to R^2:

Peter Westfall explains (in Figure 8.1 of the book) how R^2 influences the separation of distributions in our simple regression model.

In our regression of *height on weight* (the order is correct, that's how you say it), the R^2 is 0.5696. The following plot shows how "well" (i.e. precise) one can predict height if we use the 10% and 90% quantile of the weights (x_low and x_high). In both, you see the conditional distribution of height given the weight $X = x_{low}$ or $X = x_{high}$. Scenario 1 is the original model, scenario 2 is the same data with added noise (in Y-direction), which reduces R^2 to 0.13, much lower. In the right plot, the distributions have a large overlap and it is hard to distinguish between weights when seeing the height. With a very low R^2, the height prediction does not really change and we could just as well use the mean model.



In the left plot, given $X = x_{low}$ gives a a rather strongly shifted normal distribution of potentially observable heights for this weight compared to $X = x_{high}$. We would have a lower missclassification error when trying to distinguish heights of very light and very heavy people in the sample just by seeing the height. You can think of an even more extreme separation of these distributions, which would happen, when R^2 is very high or the true slope is much higher.

See also exercise 5.

An interesting way to look at R^2 is the following: Given, that one person is in the 90% quantile of the weight, the other is in the 10% quantile. What is the probability that the height of the person in the 90% quantile is higher

than the height of the other person? We could calculate this relatively easy using theorems about additivity of normal distributions. Since we are all about application, we simulate this:

```
library(rethinking)
data(Howell1)
d <- Howell1
d2 <- d[d$age >= 18, ]
# Simulate heights for the two quantiles
n_sims <- 10000
set.seed(123)
mod <- lm(height ~ weight, data = d2)
summary(mod)$r.squared
```

```
## [1] 0.5696444
```

```
x_low_high <- quantile(d2$weight, probs = c(0.1, 0.9))
x_low_high
```

```
##      10%      90%
## 36.48581 54.01997
```

```
mean1 <- 113.8793936 + 0.9050291 * x_low_high[1]
mean2 <- 113.8793936 + 0.9050291 * x_low_high[2]
sd <- 5.086  # Standard deviation for both distributions
simulated_heights <- tibble(
  # conditional normal distribution according to the model
  low_heights = rnorm(n_sims, mean = mean1, sd = sd),
  high_heights = rnorm(n_sims, mean =  mean2, sd = sd)
)

# Calculate the probability that the height of the person in the 90% quantile is higher
# than the height of the person in the 10% quantile
simulated_heights %>%
  mutate(higher_height = high_heights > low_heights) %>%
  dplyr::summarise(mean(higher_height))
```

```
## # A tibble: 1 x 1
##   `mean(higher_height)`
##                   <dbl>
## 1                 0.987
```

In other words, we can be almost sure, that a person in the 90% quantile of the weight is taller than a person in the 10% quantile of the weight given this data set. See also exercise 11.

3.2.6 Check regression assumptions

Everytime we fit a model, we should check the assumptions above. We do this for different reasons (which will become clearer over the course). The assumptions are independent of each other. They can all be true or false or some can be true and some false (Wesftall, p.21). Chapter 4 in the book is dedicated to this topic. It is important to know that the assumptions are usually not met exactly. The question is *how badly* they are violated, not *if* they are violated. Furthermore, the asssumptions refer to the data generating process, not the data itself. Thus, the evaluation of the assumptions should involve subject matter knowledge.

p-values to evaluate model assumptions are not a good idea. To quote the American Statistical Association (ASA): "By itself, a p-value does not provide a good measure of evidence regarding a model or hypothesis." Deicision-tree thinking might not be the best idea for statistical modeling.

We will not use hypothesis tests for assumptions, because

- They are never met exactly. You cannot "prove" them.
- With small sample sizes, the statistical *power* is often low.
- With large sample sizes, the smallest deviation will be "significant".

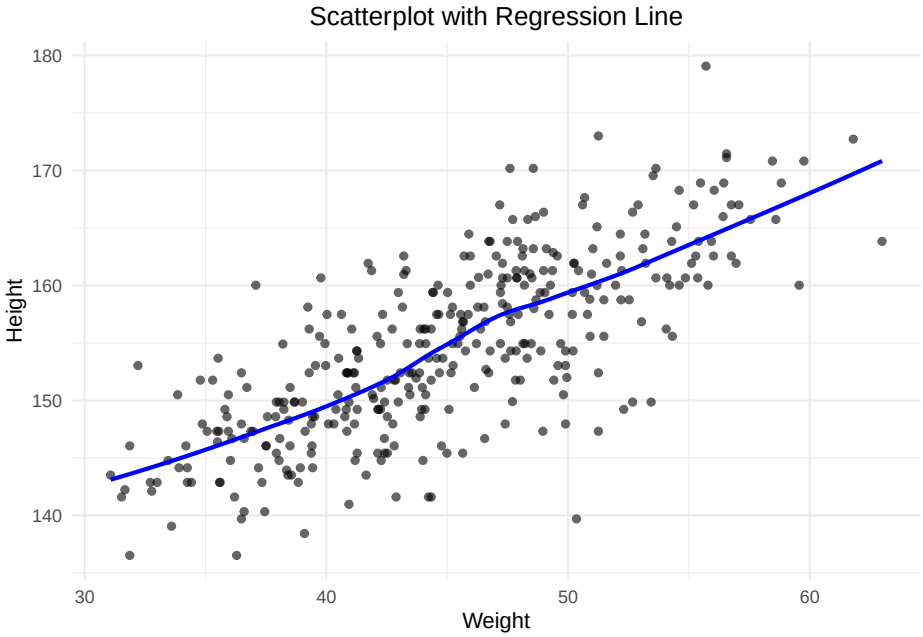
We will follow the book (chapter 4, pages 99ff) and use graphical and simulation methods to check the assumptions. As guidance, we could check them in the following order:

- Linearity
- Constant variance
- Independence
- Normality

3.2.6.1 Linearity

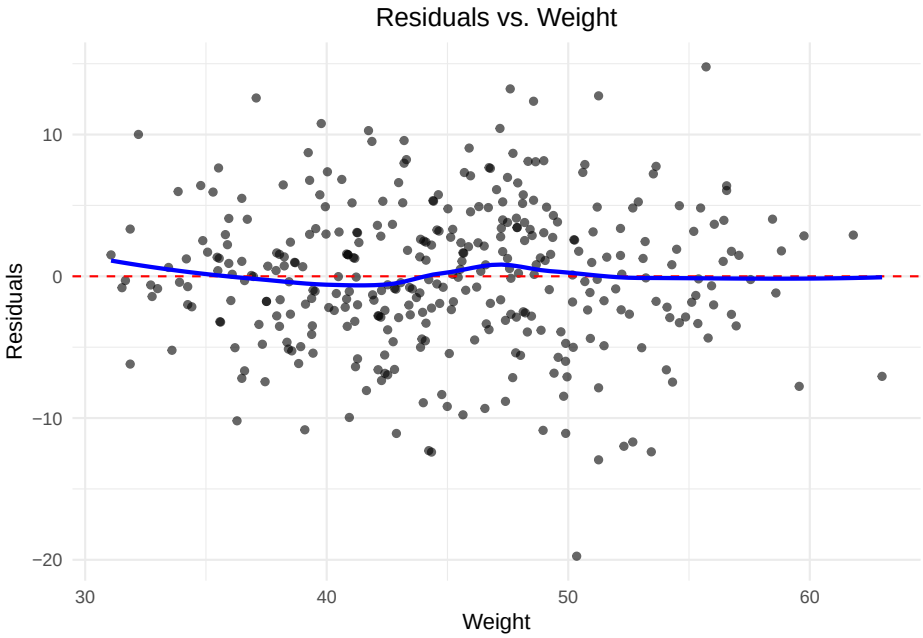
First, we **plot the data**, as we did already above. We can add a a smoothing line to the raw data as well:

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



It looks like this relationship is describable in a linear way. No apparent curvature or patches. A refined version of the scatterplot of the raw data is The **residual scatter plot** (x_i, e_i):

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

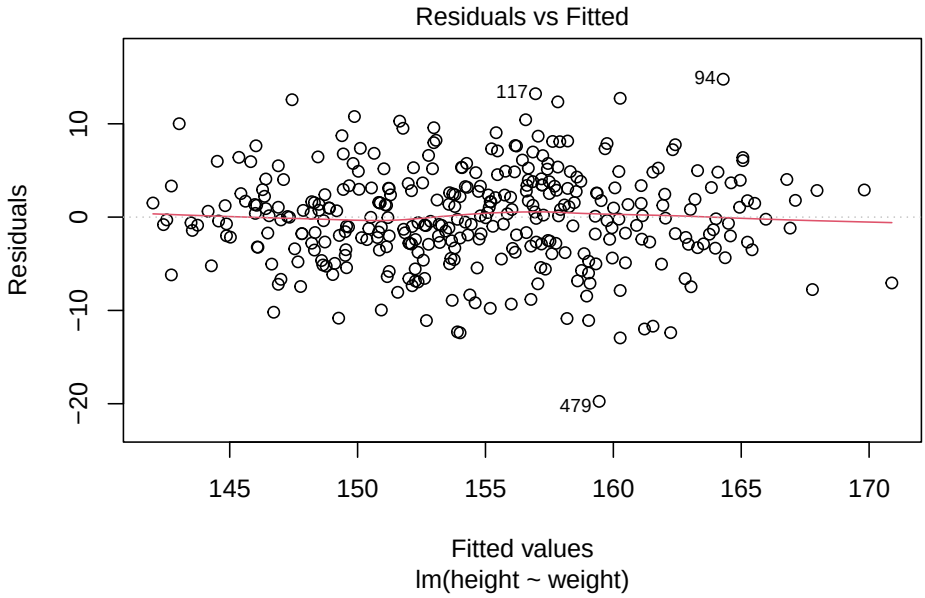


Compared to the scatterplot above, the residuals plot magnifies possible curvature. The reason is that the range of residuals is smaller than the range of the heights.

In multiple regression, we will use the $(\widehat{y}_i, e_i)$ plot, which is identical to the plot above in simple linear regression, but very helpful in multiple regression. You get the $(\widehat{y}_i, e_i)$ plot in R with `plot(mod, which = 1)`.

3.2.6.2 Constant variance

This assumption means that the variance of the residuals is constant. If it is violated, the spread around a hypothetical regression line is not constant. We look at the residual scatter plot again:



Look for a changes in patterns of vertical variability. Note, that you often do not have as many data points near the end of the range of the predictor. Here are some examples of heteroskedasticity: 1, 2, 3. The above looks homoscedastic. If it was heteroskedastic, this is not a problem, we just have to model it differently (later).

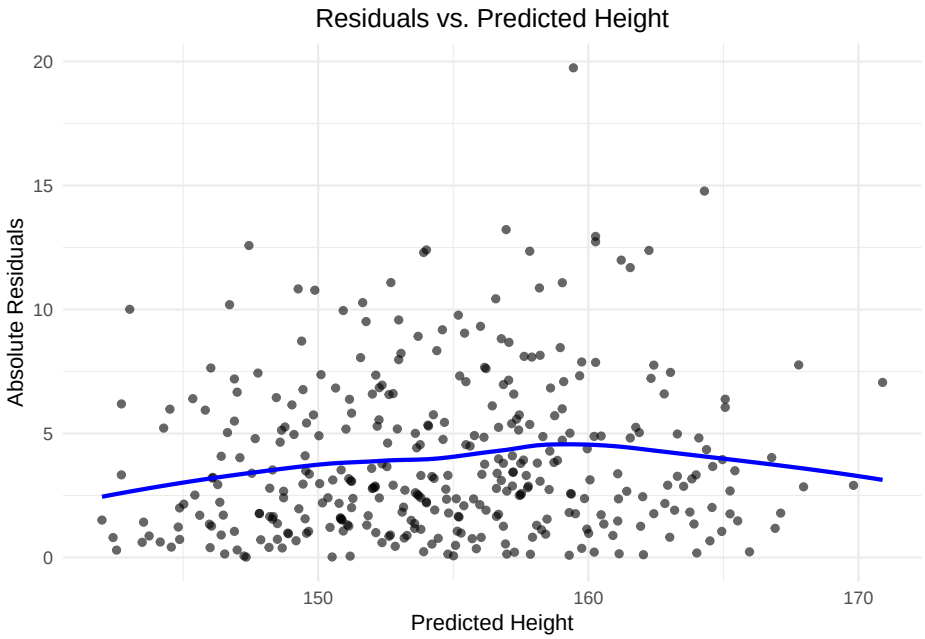
As always, it is a good idea to study the variability of these plots using simulation (exercise 7).

Better for detecting heteroscedasticity is the $(\widehat{y}_i, |e_i|)$ plot with a smoothing line:

```
## 'data.frame':    544 obs. of  4 variables:
##  $ height: num  152 140 137 157 145 ...
##  $ weight: num  47.8 36.5 31.9 53 41.3 ...
```

```
##  $ age   : num  63 63 65 41 51 35 32 27 19 54 ...
##  $ male  : int  1 0 0 1 0 1 0 1 0 1 ...
```

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



This will probably not be a perfectly horizontal line, since there are less points at the end of the range of the predictor. For instance, There are less people with extreme weights in the data set.

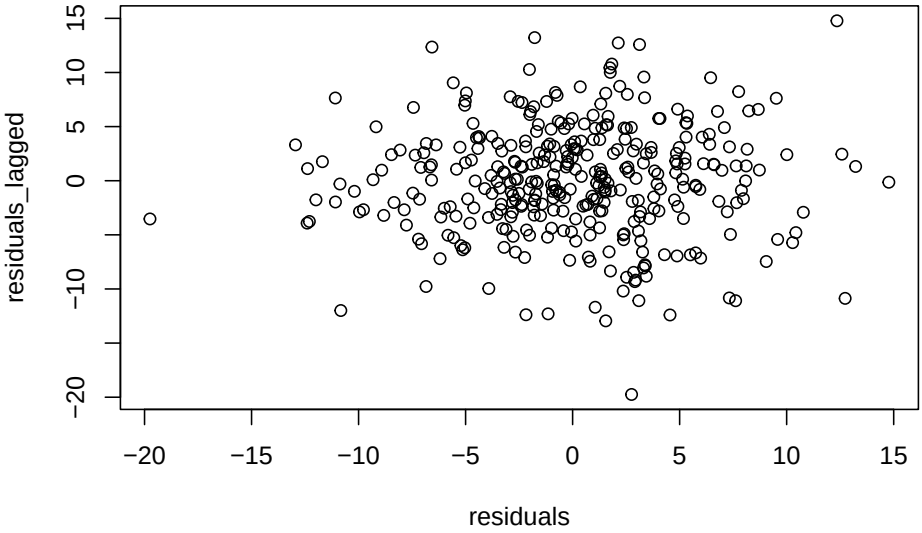
3.2.6.3 Independence, uncorrelated errors

We had an example above, where the errors were correlated. The sine curve could stem from time series data, where the x-variable is the time and the y-variable is a seasonally changing variable like temperature (purely hypothetical). In exercise 6, the values are autocorrelated: Correlated with previous values of the same variable. If we would track the body heights of persons over time, we would have an autocorrelated time series since the height of a person at time t is correlated with the height of the same person at time $t-1$, but a little less with the height at time $t-2$ and so on. If I am tall today, it is very likely that I was tall yesterday.

In our case of the !Kung San data, we do not have autocorrelated data.

But still, we could look at the correlations the residuals with lagged residuals. A $lag = 1$ means I compare the residuals with the ones right next to me and check if they are correlated.

```
## cor= 0.01858044
```



It least with a lag of 1, there so no large correlation between the residuals.

3.2.6.4 Normality

This assumptions states that the conditional distribution $Y|X = x$ is a normal distribution. We do not look at the normality of Y itself, since it is not a formal requirement, that Y is normally distributed. We need to assess the normality of the residuals

$$e_i = y_i - \hat{y}_i$$

I like doing this with a Q-Q plot from the R package `car` (command `qqPlot`):

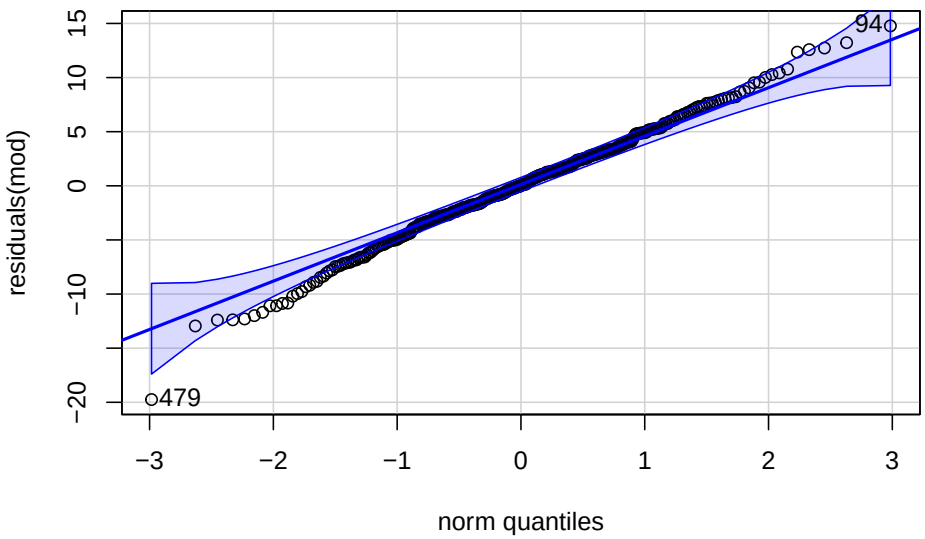
```
## Loading required package: carData
```

```
##
## Attaching package: 'car'
```

```
## The following object is masked from 'package:dplyr':
##
##     recode
```

```
## The following object is masked from 'package:purrr':
##
##     some
```

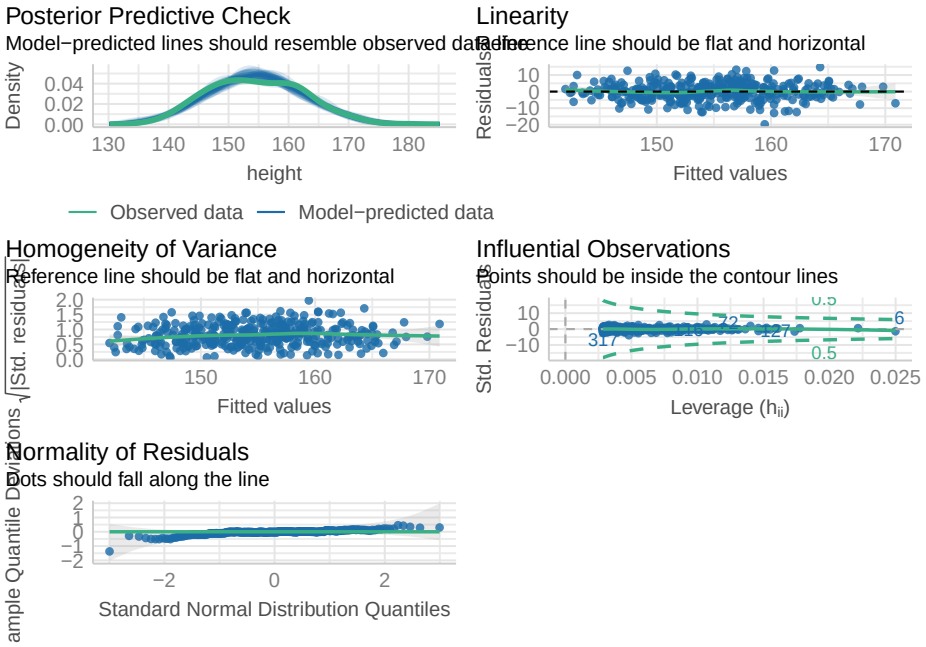
```
## The following object is masked from 'package:rethinking':
##
##     logit
```




```
## 479  94
## 317  72
```

With the 97% confidence envelope, we can see if the residuals are consistent with coming from a normal distribution. We could again use simulation to see how the Q-Q Plot changes to get a better feeling (see exercise 9).

Another convenient way to check model assumptions (for a wide class of models) is to use the `check_model` function from the `performance` package:



In this case, everything looks fine. Let's go through the plots and explain them one by one:

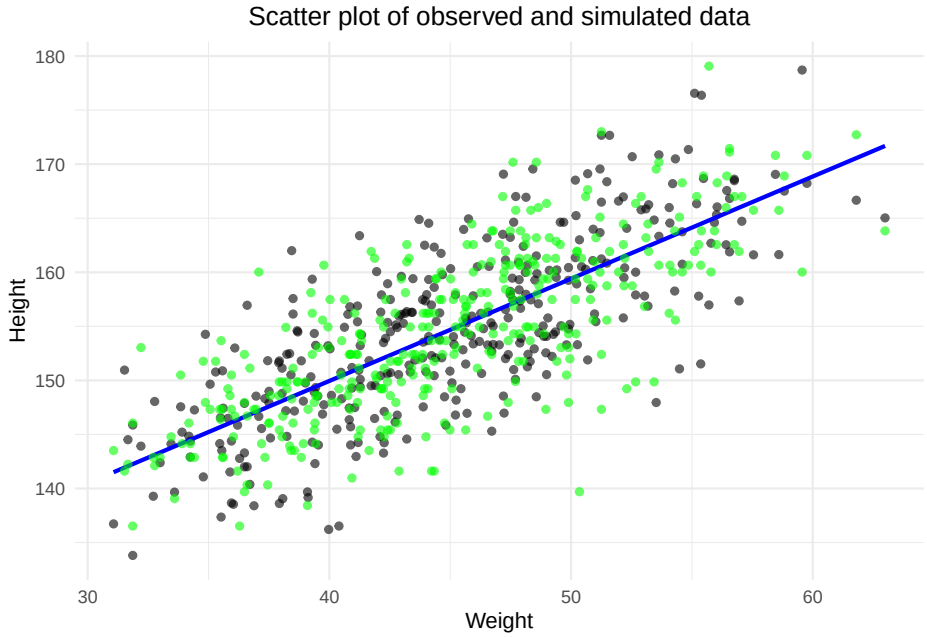
3.2.6.4.1 Posterior predictive check (upper left) Here, you create new data from the estimated model. We did that already in the Bayesian framework using `sim` from the `rethinking` package. You could also try the command `extract.samples()`from the `rethinking` package (Statistical Rethinking p. 196) to create new data from the Frequentist model. You can try this in our exercises sessions. Using least squares, the estimated model was:

$$height_i = 113.87939 + 0.90503 \cdot weight_i + Normal(0, 5.086)$$

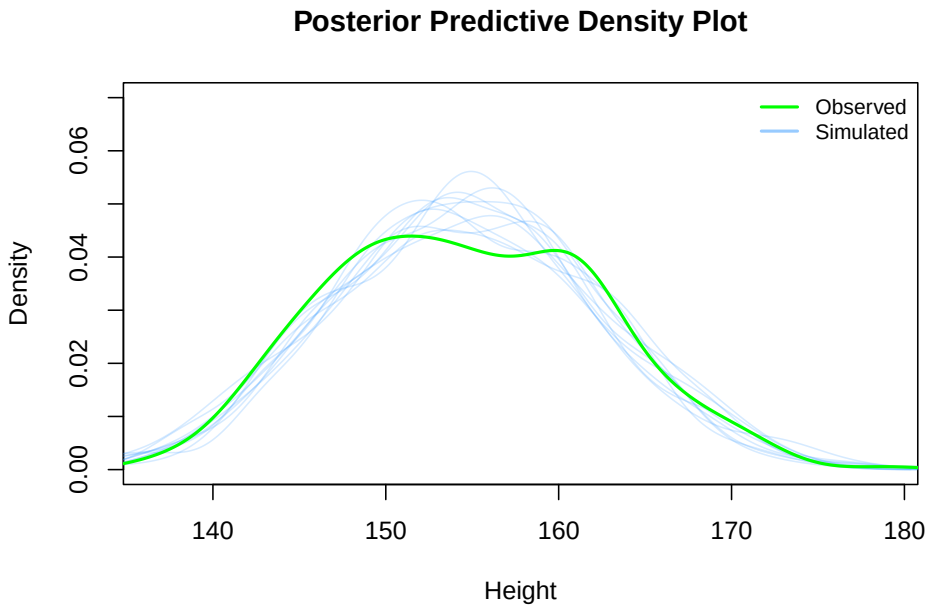
From this model, we can simulate new data (or the original weights, if we want to use fixed X) by plugging in different weights, and adding a random error from a normal distribution with mean 0 and standard deviation 5.086. Then, we can compare the distribution of the simulated data with the observed data. The blue lines in the graph are model predicted heights, the green line are the observed heights.

Let's try to replicate this plot. First, we want to simulate new data from the model. A scatter plot is also a nice way to check of the model predictions are in line with the observed data. One could repeat this process multiple times to get a feeling for the variability of the model predictions.

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



The simulated and original data (green) fit nicely together. This lends some credibility to the model - at least in terms of prediction. And now the density plots of the model created data (blue) and the observed data (green):



As you can see, the densities of the observed and simulated data are broadly similar. One could argue that heights in the range of 150-160 cm are a bit

overestimated by the model. Depending on the context, this might be a problem or not. Let's accept the model for now.

3.2.6.4.2 Linear fit (upper right)

This is the same plot as above: $(\hat{y}_i, e_i)$.

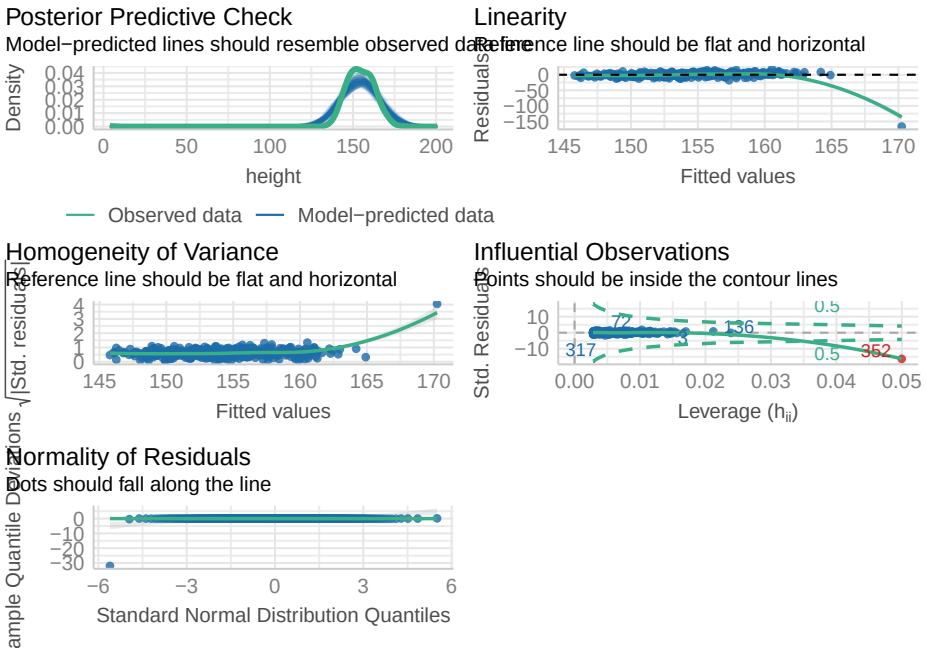
3.2.6.4.3 Homogeneity of variance (middle left)

This is the same plot as above: $(\hat{y}_i, |e_i|)$.

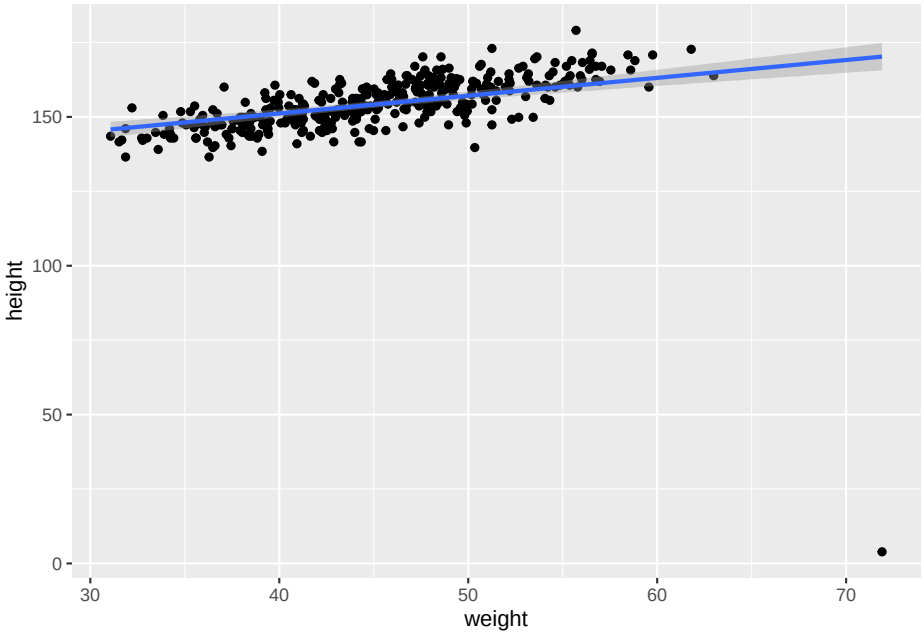
3.2.6.4.4 Influential observations (middle right)

The standard method to check for influential observations is to compute the so-called Cook's distance. This is a measure of how much leverage a single observation has on the model. We can extract the Cook's distance from the model using `cooks.distance()`. An ugly rule of thumb would be to look at observations with a Cook's distance greater than 1. In the plot, the leverage h_{ii} is on the x-axis, and the standardized residuals are on the y-axis. In short, a high leverage means that the estimated value $\hat{y}_i$ is potentially far away from the original value y_i. The contours (dotted green lines) are the Cook's distance, in this case at a Cook's distance of 0.5. There is formula that relates the leverage (h_{ii}), the Cook's distance and the residuals. Holding Cook's distant constant at 0.5, gives you the green dotted line in the plot.

Let's create an outlier and see what happens:




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




```
## Cook's distance =  7.025207
```

One single outlier changes the diagnostic plots. Observation 352 is clearly identified as influential. The one point changes all diagnostic plots notably. The estimates of the regression coefficients are also affected:

```
## [1] "Original Model"
```

```
## (Intercept)      weight
## 113.8793936   0.9050291
```

```
## [1] "Model with outlier"
```

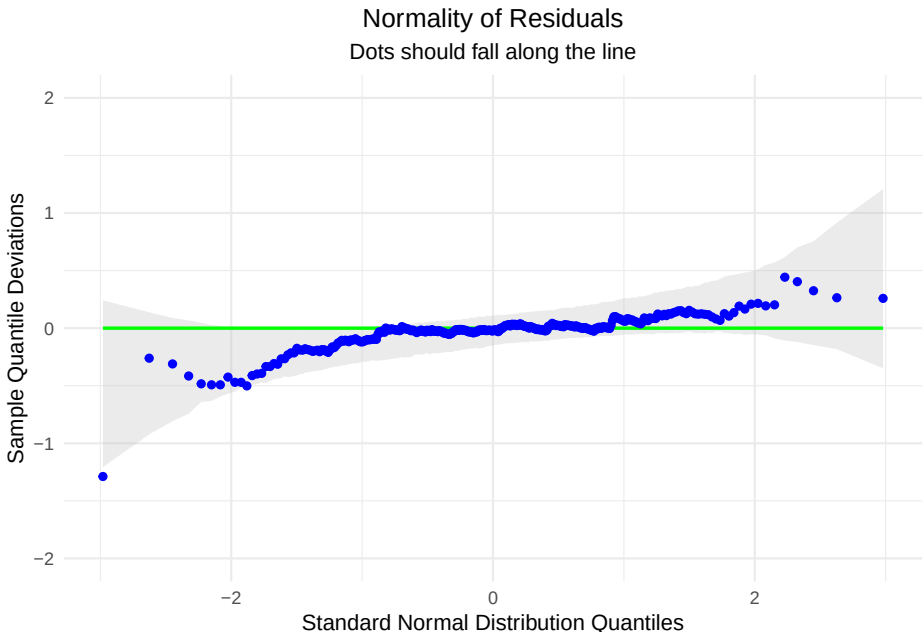
```
## (Intercept)      weight
##  127.172494    0.599054
```

Admitted, this is a somewhat artificial example.

3.2.6.4.5 Normality of residuals (lower left) This is basically the same plot as above, just detrended. Let's try to replicate it:

```
##
## Attaching package: 'qqplotr'
```

```
## The following objects are masked from 'package:ggplot2':
##
##     stat_qq_line, StatQqLine
```



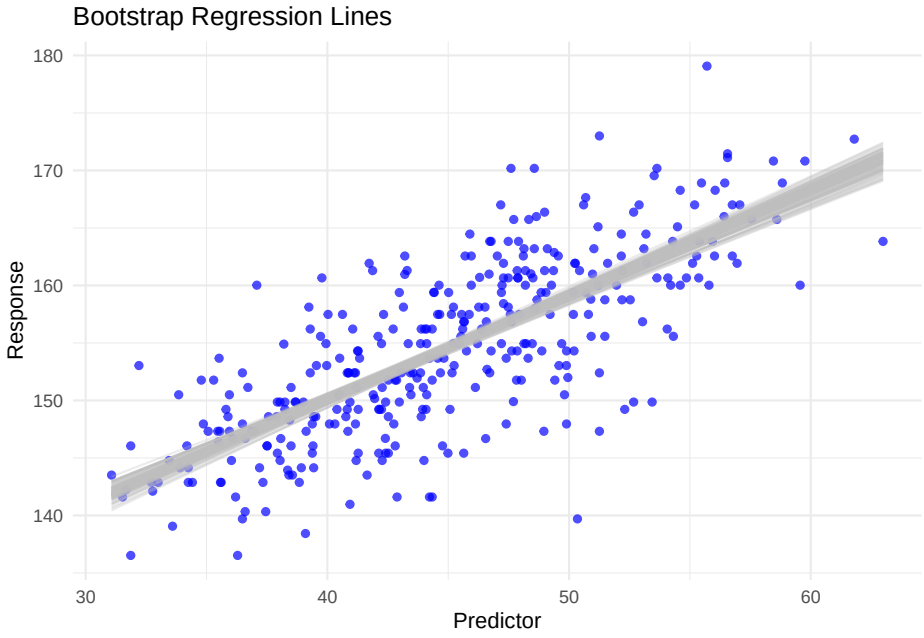
The normality assumption seems to hold. Note that in the above QQ plot, we detrended the residuals and used a bootrap confidence band. Depending on the band type, the confidence band can be wider or narrower and include or exclude points.

After having checked the assumptions for the classical regression model and we feel comfortable with the model, we get exact confidence intervals for the effect sizes (βs) using `confint()` (p. 74 in Westfall).

Heureka! We have a model that fits the data well.

3.2.7 Bootstrap fit

In order to get a feeling for the variability of the model with regards to the predictors as well, we can bootstrap the whole data set, fit the model and draw regression lines. We create 100 bootrap replicates of our data set `d2` by drawing with replication. For every bootstrap replicate, we fit the model and draw the regression line.



The results is very stable. Neither intercept nor slope change much.

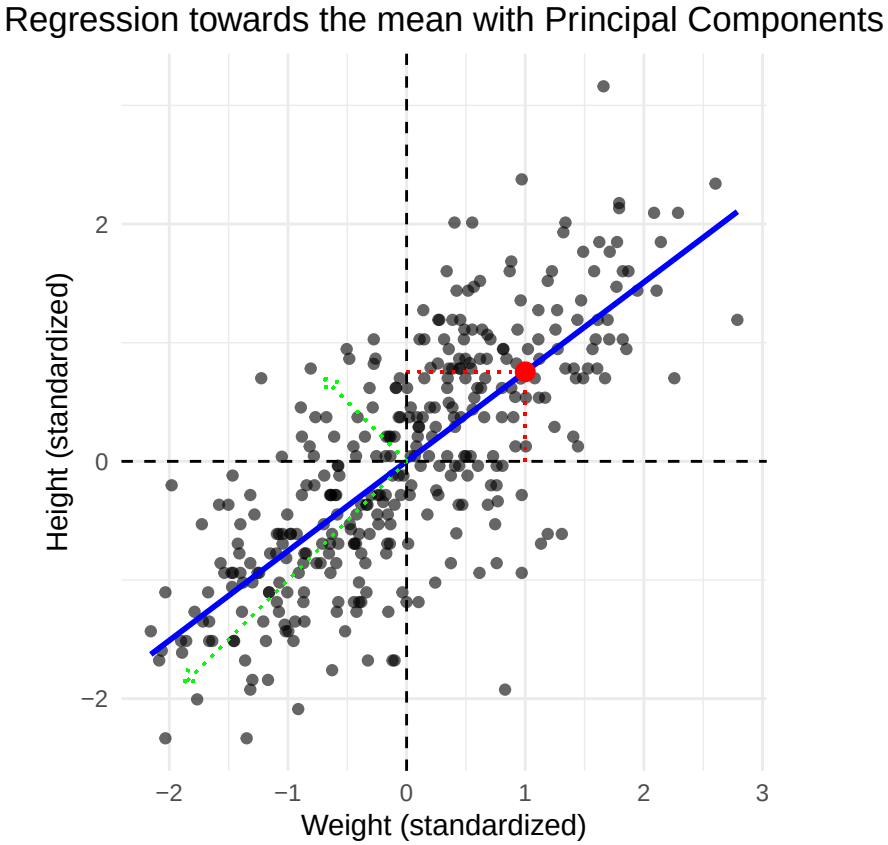
3.2.8 Regression towards the mean

There are great explanations for "regression towards the mean" in Gelman p.58 and Westfall p.36. This video might be interesting to watch.

It describes the phenomenon that the predicted value (y) is closer to its mean than the predictor (x) to its mean. In our case this means, if the weight of a person is 1 standard deviation above the mean of body weights, the (model-)predicted height is less than 1 standard deviation above the mean of body heights, but still larger than the average height. Let's verify:

```
## predicted height= 0.7547479
```

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



As Gelman points out in his book (Figure 4.2), there is a fine detail: The regression line is not the line most people would draw through the data. They would draw a line through the main directions of variability (directions are the eigenvectors of the covariance matrix). In dotted green, you can see these directions. The regression line is a solution to another problem (as we have seen): It minimizes the sum of squared residuals, which are defined via the **vertical** distances to the line. See exercise 8.

3.2.9 Random X vs. fixed X

A detail that is not mentioned often in introductory statistics courses is the question of whether the predictor variable X is random or fixed. In our case, we did not specify the weights of the people in the !Kung San data set. *Obervational* data was collected and we have no control over the weights. In an *experimental* setting, we could have controlled the weights of the people. We could have for instance only included people with weights at certain steps (50 kg, 60 kg, 70 kg). In the latter case, we could consider X as fixed. In the former case, we consider X as random. If we would draw another sample from the population,

we would get different weights again. Further reading: Westfall 1.5.

There are many more details we could look into, but we wanted to give a first practical introduction to regression analysis with less emphasis on theory and proofs behind it.

We want to make clear what the confidence intervals from `confint` of a model mean. For this see exercise 14.

3.3 Exercises

[E] Easy, [M] Medium, [H] Hard

(Some) solutions to exercises can be found in the git-repo here.

3.3.1 [E] Exercise 1

In the model from above:

$$\begin{aligned} h_i &\sim \text{Normal}(\mu_i, \sigma) \\ \mu_i &\sim \alpha + \beta(x_i - \bar{x}) \\ \alpha &\sim \text{Normal}(171.1, 20) \\ \beta &\sim \text{Normal}(0, 10) \\ \sigma &\sim \text{Uniform}(0, 50) \end{aligned}$$

- What ist the expected height when $x_i = \bar{x}$?
- What is the expected height when x_i changes by 1 unit?

3.3.2 [E] Exercise 2

Look at the marginal distrubutions of the parameters in the Bayesian model.

- Plot the posterior distribution of all 3 parameters.
- Include in the plot a 99% credible interval (HDI).

3.3.3 [M] Exercise 3

Go to the coefficient estimates in the simple linear regression setting above (Fit the model) in the classical framework.

- Create an R file to simulate the simple linear regression model.
- Change your input parameters and see how the estimates change.
- Does this make sense with respect to the estimates given, specifically with respect to β_1?

3.3.4 [H] Exercise 4

Verify the statement above in the text for high and low values of R^2.

3.3.5 [M] Exercise 5

Verify with simulation in R that the separation of the distributions in the simple linear regression model improves if the true (but usually unknown) slope increases.

3.3.6 [H] Exercise 6

Go to the model assumptions in the classical regression model (Model Assumptions). - Use the code from github to recreate the regression model with the sine-curve. - Check the independence assumption as described. Look at the residuals, when $X = 2$ and $X = 4.5$. You can get those by filtering residuals that are > 0.5 and < 0.5.

3.3.7 [M] Exercise 7

- Simulate data from the regression of heights on weights in our !Kung San data set.
- Draw the $\widehat{y}_i, e_i$ plot.
- Draw the $\widehat{y}_i, |e_i|$ plot.
- Repeat the simulation and look at the variability of the plot.

3.3.8 [E] Exercise 8

- Go to p.36 in Westfall's book and read Appendix A.
- Pay close attention to the explanation about regression toward the mean.

3.3.9 [M] Exercise 9

Go to the resuials above where we tested the normality assumption.

- Calcuate mean and standard deviation from the residuals of the model that regresses height on weight.
- Simulate from a normal distribution using these parameters.
- Get a feeling how the QQ plot changes by drawing the QQ plot repeatedly.

3.3.10 [M] Exercise 10

Using our !Kung San data,

- show that the regression of height on weight (`lm(height ~ weight)`) is not the same as the regression of weight on height (`lm(weight ~ height)`).
- Draw both regression lines in one diagram.
- Can you simulate data where the two regressions deliver (almost) identical results?
- Explain why the results differ and what consequences this would have for a research question. Which question do I answer with each?

3.3.11 [E] Exercise 11

Go back to the section about R^2 and the separation of the distributions. How would the probability that a person in the 90% quantile of the weight is taller than a person in the 10% quantile of the weight change if you change

- the true slope of the regression line
- the true σ, i.e. if you add more noise and have a lower R^2?

3.3.12 [M] Exercise 12

Go back to the Bayesian simple regression model for height above.

- Standardize weights and heights in the data set `d2`.
- Estimate the regression model as we did in the section using `quap`.
- Plot the posterior distribution using `plot(model)`.
- Interpret the regression coefficient β.

3.3.13 [M] Exercise 13

Go back to the ANOVA section.

- Calculate SSE, SST and SSR for the regression of height on weight.
- How many degrees of freedom does each term have?

3.3.14 [M] Exercise 14

When we fit a simple linear regression model (Frequentist), we get confidence intervals for the coefficients using the command `confint`. As stated previously, these are Frequentist CIs. That means, if we draw data from the true (but usually unknown) data generating process, we would expect that the true value of the coefficient (intercept and slope) lies in the CI in X% of the cases.

- Simulate data from the true data generating process of the simple linear regression model.
- Fit the model and get the confidence intervals for the coefficients.
- Repeat this process 1000 times and see how many times the true value of the coefficients lies in the CI.

Changing the distribution of the covariate x should not change the results, since we only get a different amount of data points agross the range of x.

- Verify this statement by changing the distribution of x.

3.4 eLearning 1

eLearning Assignments:

- Review all content from the script, up to and including section 3.1.
- Reading assignment on frequentist statistics (exam-relevant!): Chapter 1 (Introduction to Regression Models) from the book Understanding Regression Analysis - A Conditional Distribution Approach.
- Research task: Find at least two scientific articles in your field that apply regression (either simple regression with one predictor or multiple regression). We have already searched for journals relevant to your field in QM1. If you send me one of these studies in advance, we could also discuss the applied methods in more detail during the practice sessions.

Chapter 4

Multiple Linear Regression

So far, we have dealt with the simple mean model and the model with one predictor in the Bayesian and Frequentist framework. We will now add another predictor and subsequently an interaction term to the model. Finally, we will add more than two predictors to the model.

If you feel confused at any point: As Richard McElreath repeatedly says: This is normal, it means you are paying attention. I also refer to the great Richard Feynman.

4.1 Linear Regression with 2 predictors in the Bayesian Framework

4.1.1 Meaning of "linear"

What is a linear model? The term "linear" refers to the relationship of the predictors with the dependent variable (or outcome). The following model is also linear:

$$height_i = \beta_0 + \beta_1 x_i + \beta_2 x_i^2$$

The model is linear in the parameters $\beta_0, \beta_1, \beta_2$ but not in the predictors x_i. The term x_i^2 is ok, since the heights are just sums of multiples of the predictors (which can be nonlinear). This model is not a linear model anymore:

$$height_i = \beta_0 + \beta_1 x_i + e^{\beta_2 x_i^2}$$

β_2 is now is the exponent of e. It would also not be linear, if the coefficients are in a square root or in the denominator of a fraction, or in a sine or in a logarithm. You get the idea.

Here is an easy way to check if the model is linear: If I change the predictor-value (i.e.,the value of x_i, x_i^2 or whatever your predictor is) by one unit, the change in the (expected value of the) dependent variable is the coefficient in front of the predictor (β_i).

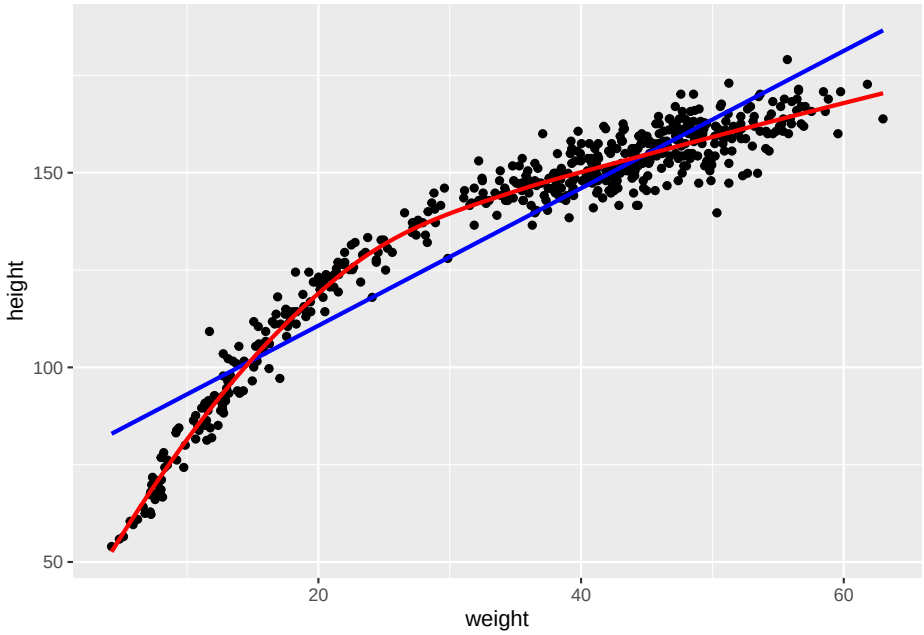
4.1.2 Adding a transformed predictor to the model

Around 4.5. in the book Statistical Rethinking there is are lineare regression using a quadratic term for weight. It is a principle, called the "**variable inclusion principle**", that we always include the lower order terms when fitting a model with higher order terms. See Westfall, p. 213. If we do not include the lower order terms, the coefficient does not measure what we want it to meausure (curvature in our case). For instance, if we want to model a quadratic relationship (parabola) between weight and height, we also have to include the linear term for weight (x_i). Since we do not assume the relationship between weight and height to be linear but quadratic (which is a polynomial of degree 2), we call this a polynomial regression. This video could be instructive. One has to be careful with fitting polynomials to data points since the regression coefficients can become quite large. Using a polynomial of high degree implies to have a lot of parameters to estimate. Increasing the degree of the polynomial increases R^2 but also the risk of overfitting. (see Statistical Rethinking p. 200). So this is - of course - not the final solution to regression problems.

This time, lets look at the whole age range of data from the !Kung San people.

```
library(rethinking)
library(tidyverse)
data(Howell1)
d <- Howell1
d %>% ggplot(aes(x = weight, y = height)) +
  geom_point() +
  geom_smooth(method = "lm", se = FALSE, color = "blue") +
  geom_smooth(method = "loess", se = FALSE, color = "red")
```

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



It would not be a good idea to fit a linear trend through this data, because we would not caupture the relationship adequately. The red line is a loess smothing line which is often used to capture non-linear relationships. The blue line is the usual line from classic linear regression (from the previous chapter). Which one describes the data more accurately? In this case it is obvious, a non-linear relationship is present and it might be a good idea to model it. Modeling the relationship with a linear trend leads to bad residuals with structure. We will demonstrate this in the freuqentist setting. Unfortunately, in more complex settings, with more predictors, it is not always so easy to see.

This time, we use the mean for the prior from the book ($178cm$). The model equations are (see exercise 2):

$$
\begin{aligned}
h_i &\sim \text{Normal}(\mu_i, \sigma) \\
\mu_i &= \alpha + \beta_1 x_i + \beta_2 x_i^2 \\
\alpha &\sim \text{Normal}(178, 20) \\
\beta_1 &\sim \text{Log-Normal}(0, 1) \\
\beta_2 &\sim \text{Normal}(0, 1) \\
\sigma &\sim \text{Uniform}(0, 50)
\end{aligned}
$$

The prior for β_1 is log-normal, because we can reasonably assume the the overall linear trend is positive. The prior for β_2 is normal, because we are not so sure about the sign yet. If we thought back to our school days to the topic of “curve

discussion" or parabolas, we could probably also assume that β_2 is negative. But, data will show.

How can we interpret the model equations? The model assumes that the **expected** height μ_i of a person i depends non-linearly (quadratically) on the (standardized) weight x_i of the person. We are in the business of mean-modeling. The prior for σ is uniform as before. The prior for α is normal with mean 178 and standard deviation 20 because this is what we can expect from body heights in our experience.

Let's **fit the model**:

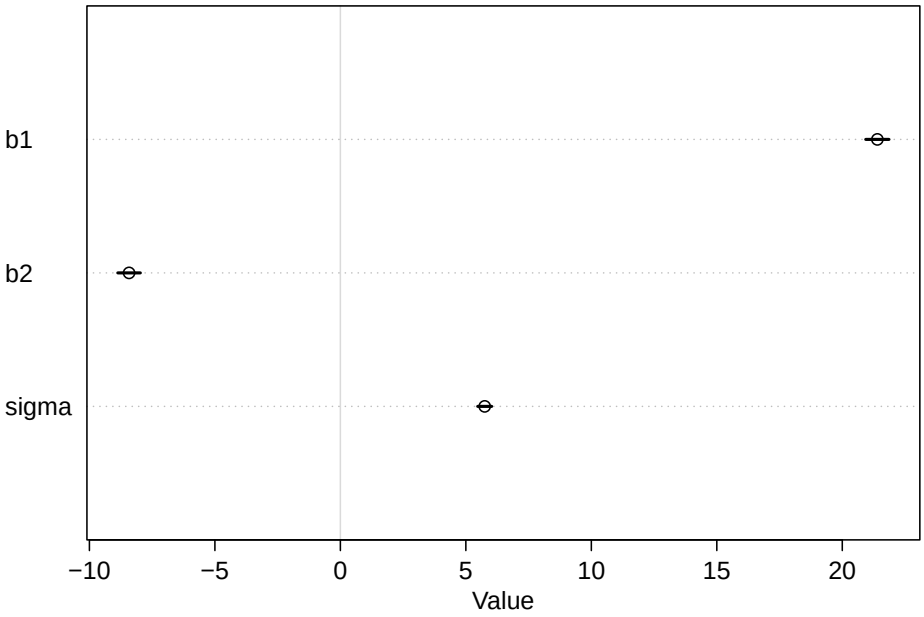
We standardize the weight again and add the squared weights to the data set. Standardizing the predictors is a good idea, especially in polynomial regression since squares and cubes of large numbers can get huge and cause numerical problems.

Let's fit the model with the quadratic term for weight:

```
# Standardize weight
d$weight_s <- (d$weight - mean(d$weight)) / sd(d$weight)
# Square of standardized weight
d$weight_s2 <- d$weight_s^2
m4.1 <- quap(
  alist(
    height ~ dnorm(mu, sigma),
    mu <- a + b1*weight_s + b2*weight_s^2,
    a ~ dnorm(178, 20),
    b1 ~ dnorm(0, 10),
    b2 ~ dnorm(0, 10),
    sigma ~ dunif(0, 50)
  ), data = d)
precis(m4.1)
```

```
##             mean         sd       5.5%      94.5%
## a     146.672739 0.3736465 146.075580 147.269898
## b1     21.397637 0.2898827  20.934348  21.860925
## b2     -8.419933 0.2813308  -8.869554  -7.970312
## sigma   5.750550 0.1743749   5.471865   6.029235
```

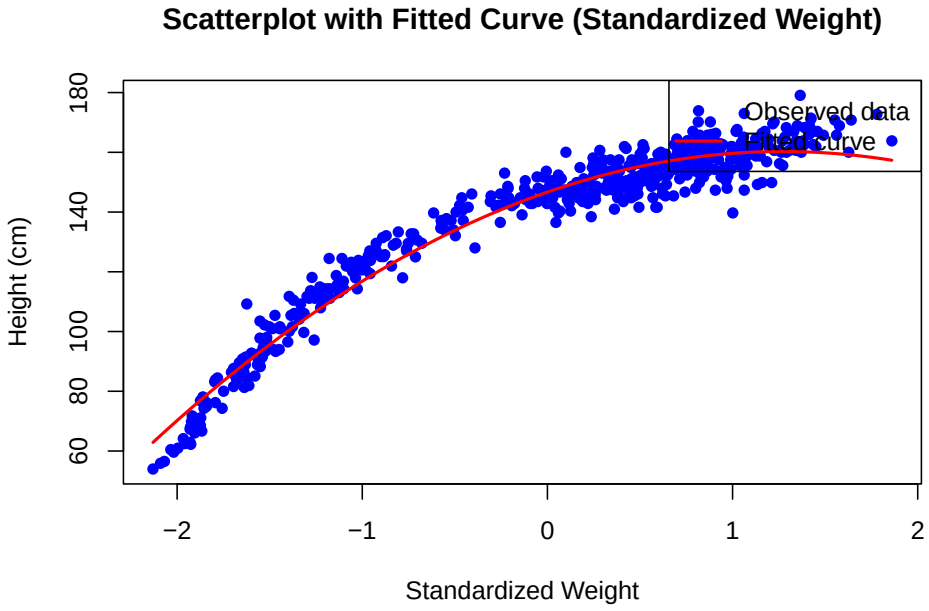
```
plot(precis(m4.1, pars = c("b1", "b2", "sigma")))
```



β_2 is indeed negative. In the estimate-plot above I have left out the intercept α to make the other coefficients more visible. As you can see, the credible intervals are very tight. Using prior knowledge an data, the model is very sure about the coefficients. We get our **joint distribution** of the **four model parameters**. Let's look at the fit using the mean estimates of the posterior distribution:

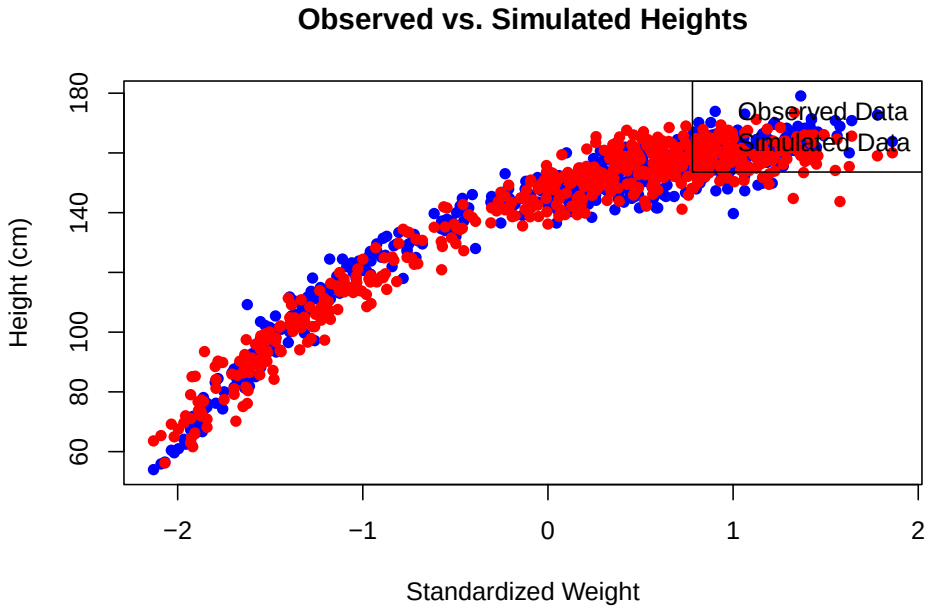
```r
# Summarize the model parameters
model_summary <- precis(m4.1)
params <- as.data.frame(model_summary)

# Extract parameter values
a <- params["a", "mean"]        # Intercept
b1 <- params["b1", "mean"]      # Coefficient for standardized weight
b2 <- params["b2", "mean"]      # Coefficient for squared standardized weight

# Generate a sequence of standardized weights for the fitted curve
weight_seq <- seq(min(d$weight_s), max(d$weight_s), length.out = 200)

# Calculate the fitted values using the quadratic equation
height_fitted <- a + b1 * weight_seq + b2 * weight_seq^2

# Plot the scatterplot
plot(d$weight_s, d$height, pch = 16, col = "blue",
     xlab = "Standardized Weight", ylab = "Height (cm)",
     main = "Scatterplot with Fitted Curve (Standardized Weight)")
```

```r
# Add the fitted curve
lines(weight_seq, height_fitted, col = "red", lwd = 2)

# Add a legend
legend("topright", legend = c("Observed data", "Fitted curve"),
       col = c("blue", "red"), pch = c(16, NA), lty = c(NA, 1), lwd = 2)
```




```r
        # ======== Simulate Heights from Posterior ========

# Prepare new data with the same number of rows as the original data
new_data <- data.frame(weight_s = d$weight_s, weight_s2 = d$weight_s2)

# Simulate height values from the posterior (same number as original data)
sim_heights <- sim(m4.1, data = new_data, n = nrow(d))  # Posterior samples

# Extract random samples from simulated heights
height_samples <- apply(sim_heights, 2, function(x) sample(x, 1))

# ======== Plot Observed vs. Simulated Heights ========

# Plot observed data
plot(d$weight_s, d$height, pch = 16, col = "blue",
     xlab = "Standardized Weight", ylab = "Height (cm)",
     main = "Observed vs. Simulated Heights")
```

```r
# Add simulated height points
points(d$weight_s, height_samples, pch = 16, col = "red")

# Add legend
legend("topright", legend = c("Observed Data", "Simulated Data"),
       col = c("blue", "red"), pch = c(16, 16))
```



This fits much better than the linear model without the quadratic term. In the book, there is also a polynomial regression with a cubic term for weight. The second plot shows the original heights and simulated heights from the posterior distribution in one plot. This fits quite well. Maybe this fits even better (see exercise 1).

4.1.3 Adding another predictor to the model

Since the !Kung San data set has already such a high R^2 with the quadratic term (and possibly higher with the cubic term), we will use the created data set from below in the frequentist setting to estimate the coefficients of the model with two predictors here as well. We *have* the true but (usually) unknown data generating mechnisms (for didactic reasons).

We use rather uniformative priors and fit the model using `quap`:

```r
library(rethinking)
set.seed(123)
```

```
n <- 100
X1 <- rnorm(n, 0, 5)
X2 <- rnorm(n, 0, 5)
Y <- 10 + 0.5 * X1 + 1 * X2 + rnorm(n, 0, 2) # true model
df <- data.frame(X1 = X1, X2 = X2, Y = Y)

# fit model
m4.2 <- quap(
  alist(
    Y ~ dnorm(mu, sigma),
    mu <- a + b1*X1 + b2*X2,
    a ~ dnorm(10, 10),
    b1 ~ dnorm(0, 10),
    b2 ~ dnorm(0, 10),
    sigma ~ dunif(0, 50)
  ), data = df)
precis(m4.2)
```

```
##             mean         sd      5.5%      94.5%
## a     10.2700227 0.18934442 9.9674137 10.5726316
## b1     0.4467278 0.04131434 0.3806995  0.5127561
## b2     1.0095082 0.03899992 0.9471788  1.0718377
## sigma  1.8738833 0.13250803 1.6621099  2.0856567
```

4.1.3.1 Checking model assumptions

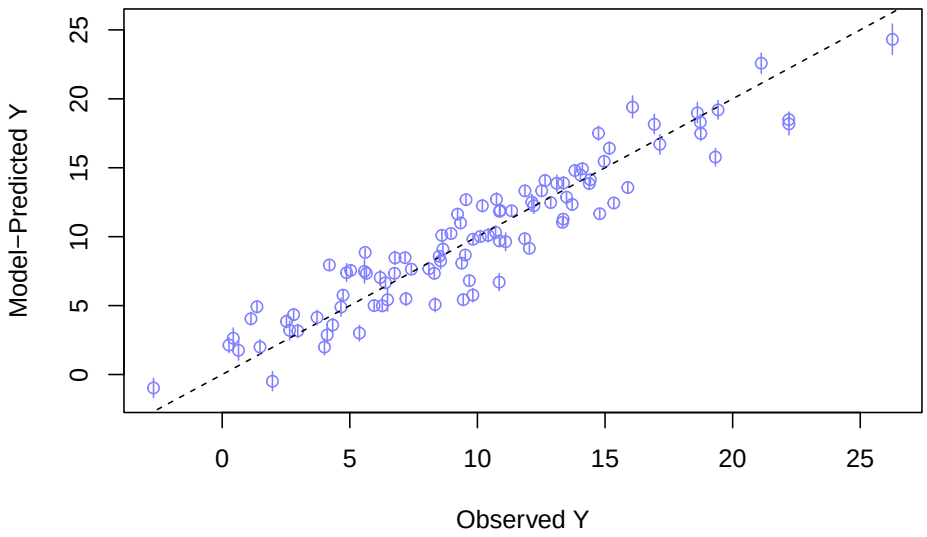
Andrew Gelman mentions in some of his talks (see here for more details) that many Bayesians he met do not check their models, since they reflect subjective probability. As I said in the introduction, one should not be afraid to check model predictions against the observed and probably new data. If a model for predicting BMI performs much worse on a new data set, we should adapt. We **do not ask** the question if a model is true or false, but if it is useful or how badly the model assumptions are violated.

For further, more detailed information on model checking, refer to chapter 6 of Gelman's book.

Anyhow, we plot two posterior predictive checks here. We test the model within the same data set. In order to do this, we create new observations by drawing from the posterior distribution and compare these with the acutally observed values. This is called **posterior predictive checks**.

First, we plot the observed Y values against the predicted Y values ($= \hat{Y}$) from the model (as in Statistical rethinking, Chapter 5). Although these practically never lie on the line $y = x$, they should be sufficiently close to it. We could also compare these two plots with the mean-model (see exercise 6).

```r
# 1) Posterior predictive checks Y vs Y_hat
# see Statstical Rethinking p 138.
# call link without specifying new data
# so it uses the original data
mu <- link(m4.2)

# summarize samples accross cases
mu_mean <- apply(mu, 2, mean)
mu_PI <- apply(mu, 2, PI, prob = 0.89)

# simulate observations
# again, no new data, so uses original data
D_sim <- sim(m4.2, n = 1e4)
D_PI <- apply(D_sim, 2, PI, prob = 0.89)

plot(mu_mean ~ df$Y, col = rangi2, ylim = range(mu_PI),
     xlab = "Observed Y", ylab = "Model-Predicted Y")
abline(a = 0, b = 1, lty = 2)
for(i in 1:nrow(df)) lines(rep(df$Y[i], 2), mu_PI[,i], col = rangi2)
```



As we can see, the model fits the data quite well. The points are close to the dashed line ($y = x$). No under- or overestimation is visible. The model seems to capture the relationship between the predictors X_1 and X_2 and the dependent variable Y quite well - at least in a predictive sense. If there were patches of data points above or below the dashed line, we would probably have to reconsider the model definition and think about why these points are not captured by the model.

Next, we plot the posterior predictive plots analog to the upper left in

the `check_model` output.

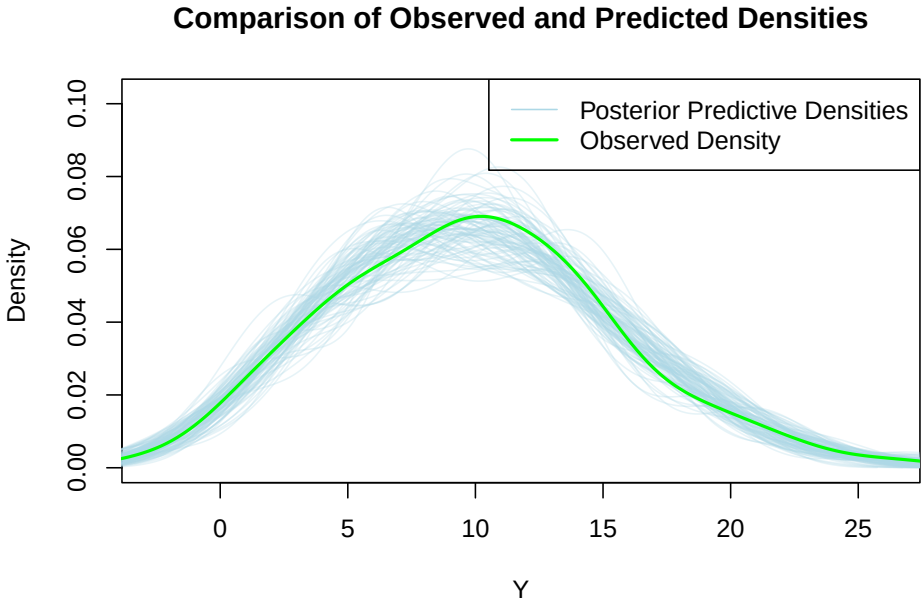
```
library(scales)  # For the alpha function to adjust transparency
```

```
##
## Attaching package: 'scales'
```

```
## The following object is masked from 'package:purrr':
##
##     discard
```

```
## The following object is masked from 'package:readr':
##
##     col_factor
```

```
# 2) Posterior predictive densities
# Simulate observations using the posterior predictive distribution
D_sim <- sim(m4.2, n = 1e4)  # Generate 10,000 simulated datasets

# Calculate densities for all samples
densities <- apply(D_sim, 1, density)

# Find the maximum density value for setting the y-axis limits
max_density <- max(sapply(densities, function(d) max(d$y)))

# Create the density plot with predefined ylim
plot(NULL, xlim = range(df$Y), ylim = c(0, max_density),
     xlab = "Y", ylab = "Density",
     main = "Comparison of Observed and Predicted Densities")

# Add 100 posterior predictive density lines
set.seed(42)  # For reproducibility
n_lines <- 100
samples <- sample(1:1e4, n_lines)  # Randomly sample 100 posterior predictive datasets
for (s in samples) {
  lines(density(D_sim[s, ]), col = alpha("lightblue", 0.3), lwd = 1)
}

# Add the density line for the observed Y values
obs_density <- density(df$Y)
lines(obs_density$x, obs_density$y, col = "green", lwd = 2)

# Add legend
legend("topright", legend = c("Posterior Predictive Densities", "Observed Density"),
       col = c("lightblue", "green"), lty = 1, lwd = c(1, 2))
```



The light blue lines show distributions of model predicted Y values. The green line shows the distribution of the observed Y values. As we can see, there seem to be no systematic differences between the observed and predicted values. The model seems to capture the relationship well. If we see systematic deviations here, we need to reconsider the model definition.

Example: If you want to predict pain (Y variable) and you have a lot of zeros (pain-free participants) you will probably see a discrepancy between the observed and predicted values in this plot. What could you do? You could use a two step process (model the probability that a person is pain-free and then model the pain intensity for the people who have pain) or use a different model (like a zero-inflated model).

Note that we did not explicitely assume normally distributed errors in the model definition above, so we won't check this here but in the Frequentist framework below.

4.2 Linear regression with 2 predictors in the Frequentist Framework

To reiterate from the last chapter: In full, the **classical linear regression model** can be written as (see p. 21-22 in Westfall):

$$Y_i|X_i = x_i \sim_{independent} N(\beta_0 + \beta_1 x_{i1} + ... \beta_k x_{ik}, \sigma^2)$$

for $i = 1, ..., n$.

The Y_i are independently normally distributed *conditioned* on the predictors having the values $X_i = x_i$. Each conditional distribution has an expected value (μ) that is a linear function of the predictors and a constant variance σ^2.

If the assumptions of the classical linear regression model are met, the least squares estimators (OLS) are the best (smallest variance) linear unbiased (on average correct) estimators - so-called: **BLUE** - of the parameters.

4.2.1 Adding a transformed predictor to the model

No, let's fit the same model as above in the Bayesian framework.

The model is:

$$height_i = \alpha + \beta_1 weight_i + \beta_2 weight_i^2 + \varepsilon_i$$

whereas

$$\varepsilon_i \sim N(0, \sigma)$$

And if you build the expectation on both sides for fixed $weight_i$, you get:

$$\mathbb{E}(height_i | weight_i) = \alpha + \beta_1 weight_i + \beta_2 weight_i^2$$

The last line means, the expected height of a person given a certain weight depends quadratically on the weight. The error term ε_i is on average zero, hence it goes away here. Remember the law of large numbers: The sample mean $\bar{\varepsilon_i}$ approaches the expected value $\mathbb{E}(\varepsilon_i) = 0$ as the sample size increases. If you drew many samples (from the true model) and average over the error terms, the average will approach zero. Think of this animated graph if you need a dynamic image of the regression model. The weights are considered fixed and therefore do not change when building the expectation.

We are looking for fixed, but unknown, parameters α, β_1, β_2 and σ. The fixed σ indicates that the observations *wiggle* around the expected value equally strong not matter which weight we have. This is called **homoscedasticity**.

Let's fit the model using the `lm` function in R which uses least squares to estimate the parameters. At this point I could torture you with matrix algebra and show you the normal equations for linear regression, but I will spare you for now. Note that the least squares algorithm for fitting the curve works for all kinds of functional forms. For example, we could also fit an exponential curve using the same technique (see exercise 9).

```
# scale weight
d$weight_s <- scale(d$weight)
# Fit the model
```

```
m4.2 <- lm(height ~ weight_s + I(weight_s^2), data = d)
summary(m4.2)
```

```
##
## Call:
## lm(formula = height ~ weight_s + I(weight_s^2), data = d)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -19.9689  -3.9794   0.2364   3.9262  19.5182
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)   146.6604     0.3748  391.30   <2e-16 ***
## weight_s       21.4149     0.2908   73.64   <2e-16 ***
## I(weight_s^2)  -8.4123     0.2822  -29.80   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 5.766 on 541 degrees of freedom
## Multiple R-squared:  0.9565, Adjusted R-squared:  0.9564
## F-statistic:  5952 on 2 and 541 DF,  p-value: < 2.2e-16
```

```
mean(d$height)
```

```
## [1] 138.2636
```

```
confint(m4.2, level = 0.94)
```

```
##                       3 %       97 %
## (Intercept)   145.954018 147.366836
## weight_s       20.866788  21.962979
## I(weight_s^2)  -8.944251  -7.880337
```

See `?I` in R. This command is used so that R knows that it should treat the "^2" as "square" and not as formula syntax. We could also create a new variable as before. Whatever you prefer.

4.2.1.1 Interpretation of output and coefficients

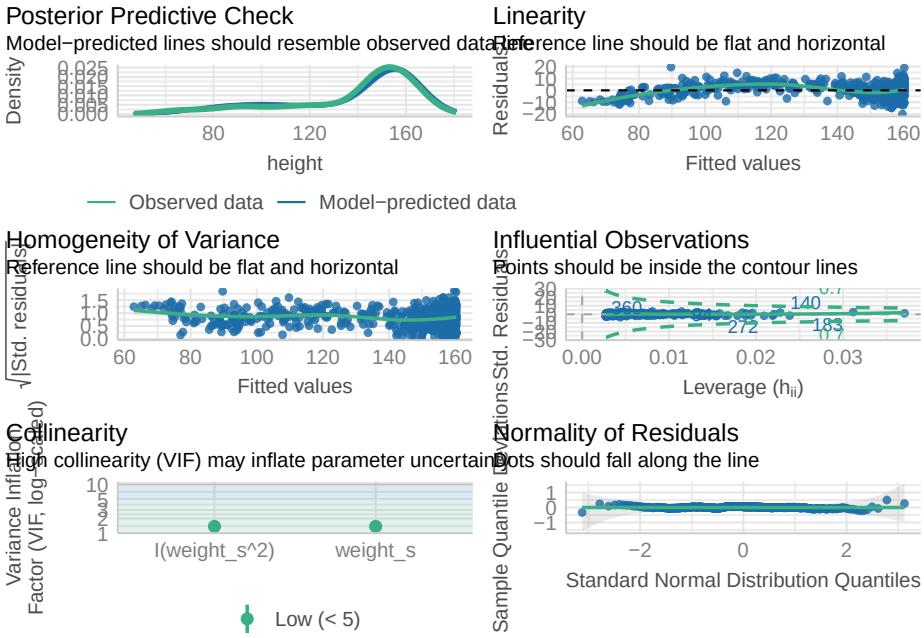
- The intercept α is the **model-predicted height** of a person of **average weight** ($weight_s = 0$ for a person of average weight). Note that this is not equal to the average height (138.2636 cm) of the people in the data set (see exercise 12).

- The residuals have range from -19.97 to 19.51. So, the model maximally overestimates the heights by 19.97 cm and underestimates by 19.51 cm. These numbers are plausible when you look at the scatterplot with the fitted curve.
- The coefficients β_1 and β_2 agree with the Bayes estimates. Specifically, β_2 is non-zero indicating curvature. You cannot directly interpret the coefficients as in the non-quadratic case since, for instance, you cannot change $weight^2$ by one unit and hold $weight$ constant at the same time. Refer to Peter Westfall's book section 9.1. for all the details.
- If you like p-values: All the hypotheses that the coefficients are zero are rejected. The p-values are very small. The values of the test statistics can not be explained by chance alone. On the other hand, for at least β_1 and and the global test this is not a surprise when you look at the scatterplot. After having fit many models, you would have guessed that all three parameters are solidly non-zero. The intercept is not zero since a person of average weight probably has non-zero height. β_1 is non-zero since you can easily imagine a linear trend line with positive slope going through the data, and β_2 is non-zero since there is clearly (non-trivial) curvature in the scatterplot.
- The R^2 is a whopping 0.96 which could be a sign of overfitting, but in this case we conclude that the true relationship is caputured rather well. Overfitting would occur if our curve would rather fit the noise in the data than the underying trend.

4.2.1.2 Checking model assumptions

```
check_model(m4.2)
```

```
## Some of the variables were in matrix-format - probably you used
##   `scale()` on your data?
##   If so, and you get an error, please try `datawizard::standardize()` to
##   standardize your data.
## Some of the variables were in matrix-format - probably you used
##   `scale()` on your data?
##   If so, and you get an error, please try `datawizard::standardize()` to
##   standardize your data.
```



If we want to be perfectionists, we could remark that (upper right plot) in the lower fitted values the residuals are more negative, meaning that the model overestimates the heights in this region. In the middle region the model underestimates a bit and we can see a positive tendency in the residuals. Apart from that, the diagnostic plots look excellent.

4.2.2 Adding another predictor to the model

Now, we add another predictor to the model. We use X_1 and X_2 **simultaneously** to predict Y. We are now in the lucky situation that we can still visualize the situation in 3D. The regression line from simple linear regression becomes a plane. The vertical distances between the data points and the plane are the residuals. See here or here at the end for examples. Minimizing the sum of the squared errors gives again the estimates for the coefficients.

For demonstration purposes, we can **create data ourselves** with known coefficients. This is the same as above. This is the true model, which we usually do not know:

$$Y_i = \beta_0 + \beta_1 X_{1i} + \beta_2 X_{2i} + \varepsilon_i$$

$$\varepsilon_i \sim N(0, \sigma^2)$$

$$\mathbb{E}(Y_i | X_1 = x_1; X_2 = x_2) = \beta_0 + \beta_1 x_1 + \beta_2 x_2$$

$$i = 1 \dots n$$

for example:

$$Y_i = 10 + 0.5 \cdot X_{1i} + 1 \cdot X_{2i} + \varepsilon_i$$
$$\varepsilon_i \sim N(0, 5)$$
$$\mathbb{E}(Y_i | X_1 = x_1; X_2 = x_2) = 10 + 0.5x_1 + 1x_2$$
$$i = 1 \dots n$$

According to the model, the conditional expected value of Y_i given $X_1 = x_1$ and $X_2 = x_2$ is a linear function of x_1 and x_2. Note, that small letters are realized values of random variables. Also note, that in the expectation the error term goes away, since $\mathbb{E}(\varepsilon_i) = 0$.

- If X_1 increases by one unit, Y increases by 0.5 units on average (in expectation).
- If X_2 increases by one unit, Y increases by 1 unit on average (in expectation).
- If X_1 and X_2 are zero, Y is 10 on average (in expectation).

Why in expectation? Because there is still the error term which makes the whole thing random! We can see that an increase in X_1 does not influence the relationship between X_2 and Y. Hence, there is **no interaction** between X_1 and X_2 with respect to Y.

Now lets's draw 100 points from this model, fit the model and add the plane:

```
library(plotly)

set.seed(123)
n <- 100
X1 <- rnorm(n, 0, 5)
X2 <- rnorm(n, 0, 5)
Y <- 10 + 0.5 * X1 + 1 * X2 + rnorm(n, 0, 2)
d <- data.frame(X1 = X1, X2 = X2, Y = Y)

# Fit the model
m4.3 <- lm(Y ~ X1 + X2, data = d)
summary(m4.3)
```

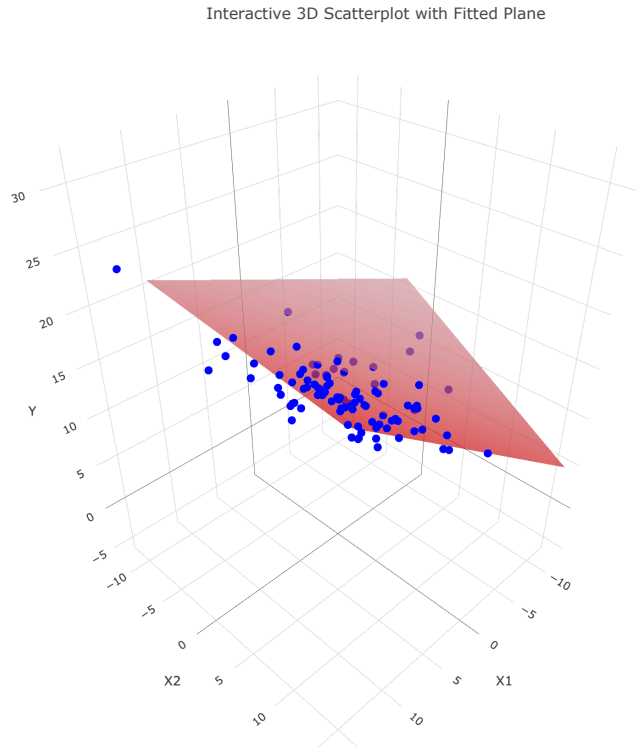
```
##
## Call:
## lm(formula = Y ~ X1 + X2, data = d)
##
## Residuals:
```

```
##     Min      1Q  Median      3Q     Max
## -3.7460 -1.3215 -0.2489  1.2427  4.1597
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 10.27013    0.19228   53.41   <2e-16 ***
## X1           0.44673    0.04195   10.65   <2e-16 ***
## X2           1.00952    0.03960   25.49   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.903 on 97 degrees of freedom
## Multiple R-squared:  0.8839, Adjusted R-squared:  0.8815
## F-statistic: 369.1 on 2 and 97 DF,  p-value: < 2.2e-16
```

```r
# Create a grid for the plane
X1_grid <- seq(min(d$X1), max(d$X1), length.out = 20)
X2_grid <- seq(min(d$X2), max(d$X2), length.out = 20)
grid <- expand.grid(X1 = X1_grid, X2 = X2_grid)

# Predict the values for the grid
grid$Y <- predict(m4.3, newdata = grid)

# Convert the grid into a matrix for the plane
plane_matrix <- matrix(grid$Y, nrow = length(X1_grid), ncol = length(X2_grid))

# Create the interactive 3D plot
plot_ly() %>%
  add_markers(
    x = d$X2, y = d$X1, z = d$Y,
    marker = list(color = "blue", size = 5),
    name = "Data Points"
  ) %>%
  add_surface(
    x = X1_grid, y = X2_grid, z = plane_matrix,
    colorscale = list(c(0, 1), c("red", "pink")),
    showscale = FALSE,
    opacity = 0.7,
    name = "Fitted Plane"
  ) %>%
  plotly::layout(
    scene = list(
      xaxis = list(title = "X1"),
      yaxis = list(title = "X2"),
      zaxis = list(title = "Y")
```

```
    ),
    title = "Interactive 3D Scatterplot with Fitted Plane"
  )
```

```
## file:////private/var/folders/pm/jd6n6gj10371_bml1gh8sc5w0000gn/T/Rtmp1MEKvh/file15e3
```



This is, of course, a very idealized situation. There is no curvature in the plane, no interaction, no outliers, no heteroscadasticity. It's the simplest case of multiple regression with 2 predictors. Reality is - usually - more complicated.

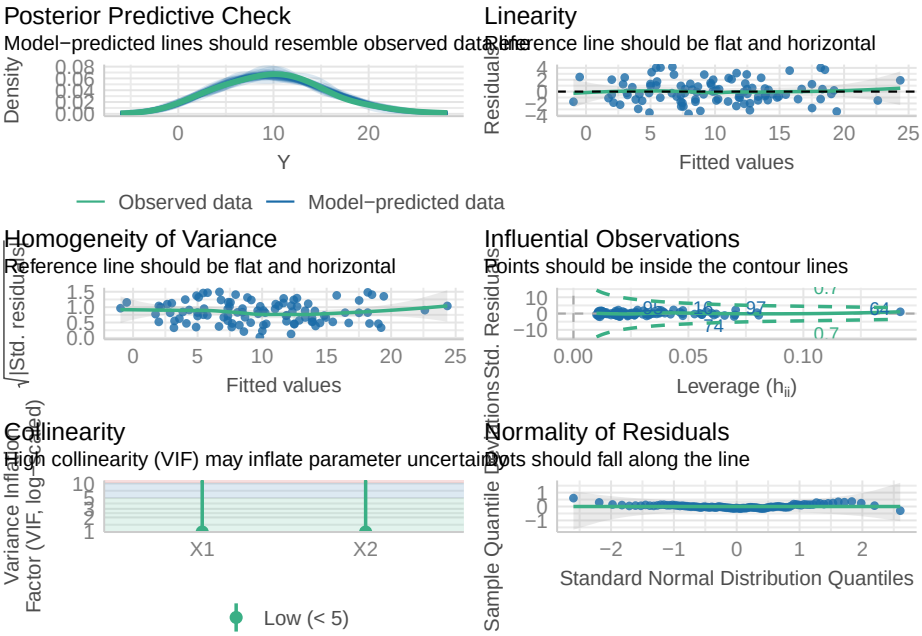
Let's look at the summary output and check model assumptions:

```
summary(m4.3)
```

```
##
## Call:
## lm(formula = Y ~ X1 + X2, data = d)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
```

```
## -3.7460 -1.3215 -0.2489  1.2427  4.1597
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 10.27013    0.19228   53.41   <2e-16 ***
## X1           0.44673    0.04195   10.65   <2e-16 ***
## X2           1.00952    0.03960   25.49   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.903 on 97 degrees of freedom
## Multiple R-squared:  0.8839, Adjusted R-squared:  0.8815
## F-statistic: 369.1 on 2 and 97 DF,  p-value: < 2.2e-16
```

```
check_model(m4.3)
```



We could repeat this simulation to get a feeling for the variability. The posterior predictive checks look nice. In this case, we *know* that the model is true.

4.2.2.1 Adding variables to the model and why

This is a very complex question. We will go into it in later chapters and the next course (Methodenvertiefung). At this point we can say this: We add variables to the model (and probably use other models apart from linear regression) *depending* on the goal at hand (prediction or explanation). Prediction seems to be

easier than explanation. For instance, within linear models and just a handful of predictors, one can even brute force the problem by searching through all subsets of predictors. If that is not possible, one could use clever algortithms, like best subest selection.

- What is **not** a good idea is to throw all variables into the model and hope for the best - especially if we want to learn the true relationships between the variables.
- What is also not a good idea is to select variables depending on the p-values of the coefficients (Westfall Chapter 11).
- **Leaving variables out**, that are important, can lead to biased estimates of the coefficients (omitted variable bias).
- Importantly, also **adding variables** can hurt conclusions from the model (see Statistical Rethinking 6.2).

4.2.3 Interaction Term $X_1 \times X_2$

I recommend reading the excellent explanations about interactions in John Kruschke's book Doing Bayesian Data Analysis, 15.2.2 und 15.2.3. Peter Westfall also has a nice explanation in his book in section 9.3.

Our statistical model is now:

$$
\begin{aligned}
Y_i &= \beta_0 + \beta_1 X_{1i} + \beta_2 X_{2i} + \ {}_{\mathbf{3}}\mathbf{X_{1i}} \times \mathbf{X_{2i}} + \varepsilon_i \\
\varepsilon_i &\sim N(0, \sigma^2) \\
\mathbb{E}(Y_i | X_1 = x_1; X_2 = x_2) &= \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_1 \times x_2 \\
i &= 1 \dots n
\end{aligned}
$$

for example:

$$
\begin{aligned}
Y_i &= 10 + 0.5 \cdot X_{1i} + 1 \cdot X_{2i} + 0.89 \cdot X_{1i} \times X_{2i} + \varepsilon_i \\
\varepsilon_i &\sim N(0, 5) \\
\mathbb{E}(Y_i | X_1 = x_1; X_2 = x_2) &= 10 + 0.5 x_1 + 1 x_2 + 0.89 x_1 \times x_2 \\
i &= 1 \dots n
\end{aligned}
$$

The second equation states that the conditional expectation of Y_i given $X_1 = x_1$ and $X_2 = x_2$ is a function of x_1 and x_2 and their interaction $x_1 \times x_2$ (i.e., the product). We are in a different situation now. Set for instance x_2 to a certain value, say $x_2 = 7$. Then the relationship (in expectation) between Y and X_1 is:

$$
\mathbb{E}(Y_i | X_1 = x_1; X_2 = 7) = 10 + 0.5 x_1 + 1 \cdot 7 + 0.89 x_1 \cdot 7
$$

$$\mathbb{E}(Y_i|X_1 = x_1; X_2 = 7) = 10 + (0.5 + 0.89 \cdot \mathbf{7}) \cdot x_1 + 1 \cdot 7$$

Depending on the value of x_2, the *effect* of X_1 on Y changes. Hence, X_2 **modifies** the relationship between X_1 and Y, or stated otherwise, X_1 and X_2 **interact** with respect to Y. Remember, the word *effect* is used in a strictly technical/statistical sense and **not in a causal** sense. It does not mean that if we *do* change X_1 by one unit, Y will also change in an experiment. We are purely describing the relationship in an associative way. We will probably touch causality in a later chapter. Bayesian statistics and causal inference are gaining popularity. Hence, we should try to keep up.

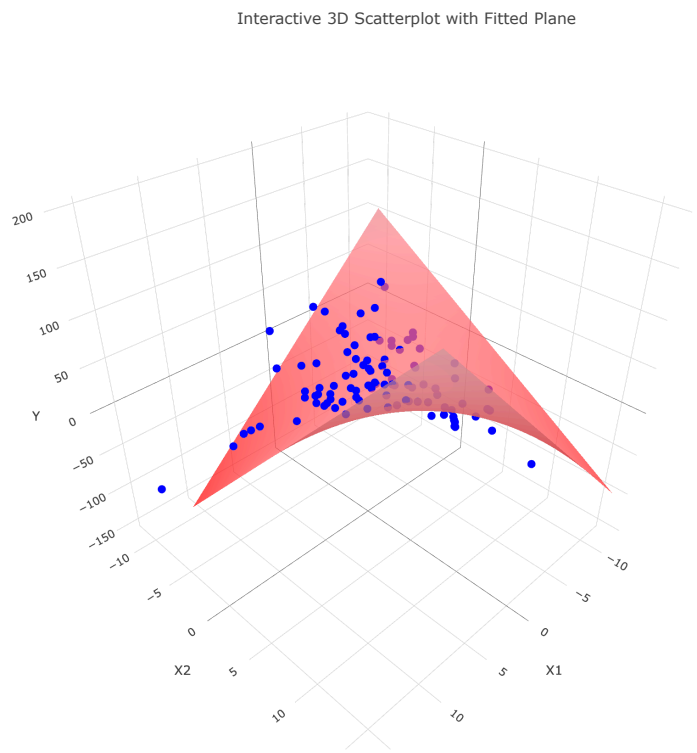
Let's draw 100 points from this model, fit the model and add the plane (see also exercise 4):

```
set.seed(123)
n <- 100
X1 <- rnorm(n, 0, 5)
X2 <- rnorm(n, 0, 5)
Y <- 10 + 0.5 * X1 + 1 * X2 + 0.89 * X1 * X2 + rnorm(n, 0, 5)
d <- data.frame(X1 = X1, X2 = X2, Y = Y)

# Fit the model
m4.4 <- lm(Y ~ X1 * X2, data = d)
summary(m4.4)
```

```
##
## Call:
## lm(formula = Y ~ X1 * X2, data = d)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -9.360 -3.389 -0.543  2.949 11.583
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 10.70491    0.47888  22.354  < 2e-16 ***
## X1           0.40719    0.10834   3.759 0.000293 ***
## X2           1.03434    0.09881  10.468  < 2e-16 ***
## X1:X2        0.92182    0.02290  40.257  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.734 on 96 degrees of freedom
## Multiple R-squared:  0.9476, Adjusted R-squared:  0.9459
## F-statistic: 578.1 on 3 and 96 DF,  p-value: < 2.2e-16
```

```r
# Create a grid for the plane
X1_grid <- seq(min(d$X1), max(d$X1), length.out = 20)
X2_grid <- seq(min(d$X2), max(d$X2), length.out = 20)
grid <- expand.grid(X1 = X1_grid, X2 = X2_grid)

# Predict the values for the grid
grid$Y <- predict(m4.4, newdata = grid)

# Convert the grid into a matrix for the plane
plane_matrix <- matrix(grid$Y, nrow = length(X1_grid), ncol = length(X2_grid))

# Create the interactive 3D plot
plot_ly() %>%
  add_markers(
    x = d$X2, y = d$X1, z = d$Y,
    marker = list(color = "blue", size = 5),
    name = "Data Points"
  ) %>%
  add_surface(
    x = X1_grid, y = X2_grid, z = plane_matrix,
    colorscale = list(c(0, 1), c("red", "pink")),
    showscale = FALSE,
    opacity = 0.7,
    name = "Fitted Plane"
  ) %>%
  plotly::layout(
    scene = list(
      xaxis = list(title = "X1"),
      yaxis = list(title = "X2"),
      zaxis = list(title = "Y")
    ),
    title = "Interactive 3D Scatterplot with Fitted Plane"
  )
```

```
## file:////private/var/folders/pm/jd6n6gj10371_bml1gh8sc5w0000gn/T/Rtmp1MEKvh/file15e3
```



The term `X1 * X2` is a shortcut for `X1 + X2 + X1:X2` where `X1:X2` is the interaction term. R automatically includes the main effects of the predictors when an interaction term is included (variable inclusion principle). The true but usually unknown βs are estimated quite precisely.

4.2.3.1 Formal test for interaction

We could apply a formal test for the interaction term by model comparison. The command `anova(., .)` would compare the two models and test if the change in the residual sum of squares is statistically interesting.

```
m4.5 <- lm(Y ~ X1 + X2, data = d) # without interaction
anova(m4.5, m4.4)
```

```
## Analysis of Variance Table
##
## Model 1: Y ~ X1 + X2
## Model 2: Y ~ X1 * X2
##   Res.Df   RSS Df Sum of Sq      F    Pr(>F)
## 1     97 38468
## 2     96  2151  1     36316 1620.6 < 2.2e-16 ***
```

```
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

One can show that the following test statistic is F distributed under the null hypothesis (that $\beta_3 = 0$):

$$F = \frac{\left(RSS_{\text{Model 1}} - RSS_{\text{Model 2}}\right) / \left(df_{\text{Model 1}} - df_{\text{Model 2}}\right)}{RSS_{\text{Model 2}} / df_{\text{Model 2}}}$$

where RSS is the residual sum of squares, df are the degrees of freedom of the residual sum of squares for both models.

The output of the `anova` command shows us the residual degress of freedom (`Res.Df`) of both models, the residual sum of squares errors of both models (`RSS`), the sum of squared errors between model 1 and model 2 (`Sum of Sq`), the value of the F-statistic and the p-value for the hypothesis, that the coefficient for the interaction term is zero ($\beta_3 = 0$). Model 1 RSS has 97 degrees of freedom, since we have 100 data points and 3 parameters to estimate ($\beta_0, \beta_1, \beta_2$). Model 2 has 96 degrees of freedom, since we have 100 data points and 4 parameters to estimate ($\beta_0, \beta_1, \beta_2, \beta_3$).

Let's verify the value of the F statistic:

```
RSS_model1 <- sum(residuals(m4.5)^2)
RSS_model2 <- sum(residuals(m4.4)^2)
df_model1 <- n - length(coef(m4.5))
df_model2 <- n - length(coef(m4.4))
F <- ((RSS_model1 - RSS_model2) / (df_model1 - df_model2)) / (RSS_model2 / df_model2)
F
```

```
## [1] 1620.606
```

```
# Sum of Sq
RSS_model1 - RSS_model2
```

```
## [1] 36316.28
```

In the numerator of the F statistic, we have the change in the residual sum of squares (from the small (model 1) model to the larger one (model 2), `Sum of Sq`) per additional parameter in the model (one additional parameter β_3).

In the denominator, we have the residual sum of squares per residual degree of freedom of the larger model (model 2). Hence, in the numerator we have the information on how much better we get with respect to the number of variables added, and in the denominator we have information on how good the full model is with respect to its degrees of freedom.

The p-value is the probability of observing a value of the F statistic as extreme or more extreme than the one we observed, given that the null hypothesis is true. Here, the p-value is extremely small. So, statistically we would see an improvement in RSS which is not explainable by chance alone. But **let's be careful with p-values** and especially with fixed cutoff values for α, which we will **never** use in this script. Even for a rather small effect β_3, we would reject the null hypothesis, if only the sample size is large enough. Since a very small effect relative to β_1 and β_2 would probably not be of practical interest, one should be careful with looking at p-values alone. For instance, in Richard McElreath's book Statistical Rethinking, there are no p-values at all. I like that.

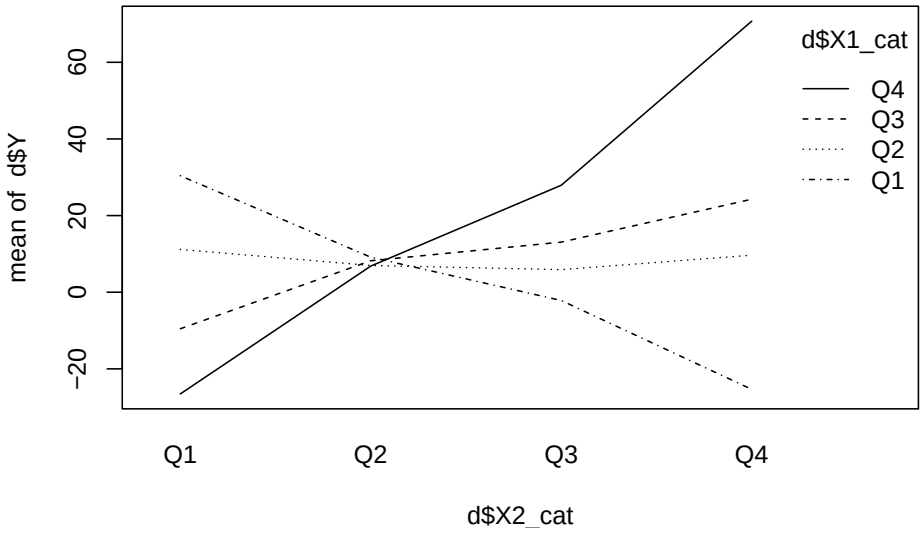
If you again look at the comparison of the RSS between the two models, you would immediately see that the model with the interaction term is better (at least with respect to this metric). The difference is huge. We have already mentioned in the context of R^2 not to overinterpret such metric, because RSS is monotonically descreasing with number of variables added and reaches zero when the number of variables equals the number of data points (see exercise 3).

4.2.4 Using an interaction plot to see a potential interaction

Chronologically before we include an interaction term in the model, we can use an interaction plot to see if there is a potential interaction between the predictors. We can just create a catorical predictor out of the continuous predictors. Just categorize the predictors into quartiles and plot the means of the dependent variable (Y). If the lines are parallel, there is probably no interaction. If the lines are not parallel, there might be an interaction.

```
n <- 100
X1 <- rnorm(n, 0, 5)
X2 <- rnorm(n, 0, 5)
Y <- 10 + 0.5 * X1 + 1 * X2 + 0.89 * X1 * X2 + rnorm(n, 0, 5)
d <- data.frame(X1 = X1, X2 = X2, Y = Y)

# Create categorical variables based on quartiles
d$X2_cat <- cut(d$X2,
                breaks = quantile(d$X2, probs = c(0, 0.25, 0.5, 0.75, 1), na.rm = TRUE),
                include.lowest = TRUE,
                labels = c("Q1", "Q2", "Q3", "Q4"))

d$X1_cat <- cut(d$X1,
                breaks = quantile(d$X1, probs = c(0, 0.25, 0.5, 0.75, 1), na.rm = TRUE),
                include.lowest = TRUE,
                labels = c("Q1", "Q2", "Q3", "Q4"))
```

```
# Create the interaction plot
interaction.plot(d$X2_cat, d$X1_cat, d$Y)
```



There seems to be an interaction of the predictors with respect to Y. The lines are not parallel. If there was no interaction, the change in Y with respect to X_2 would be the same for all levels of X_1. This seems not to be the case here. See exercise 5.

If we had one or both predictors already categorical, we would not have to discretize them before.

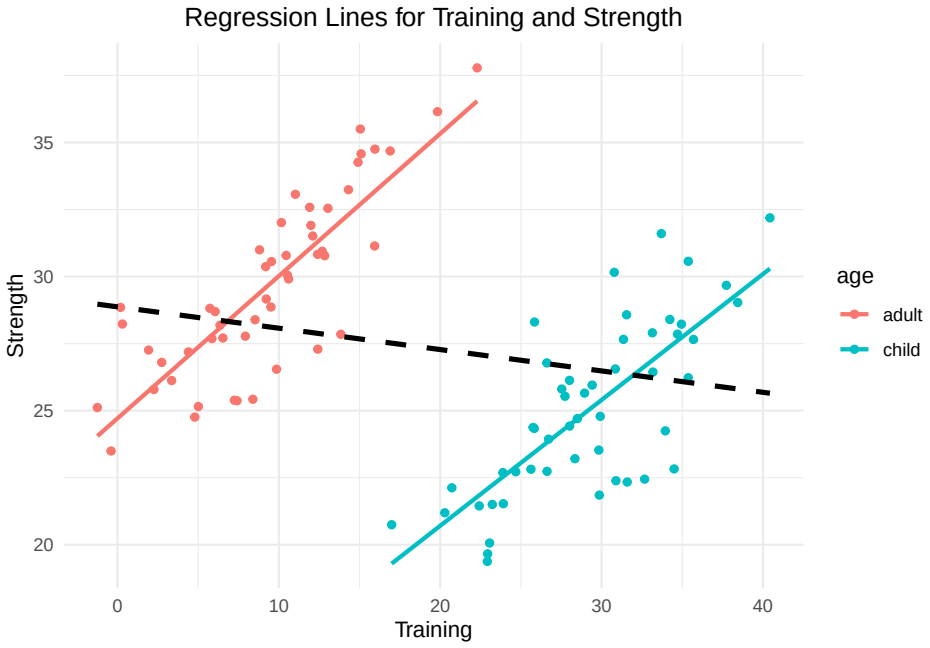
4.2.5 Simpsons Paradox

The Simpsons paradox is a phenomenon, in which a trend appears in several different groups of data but disappears or reverses when these groups are combined. I agree with the criticism that this is not really a paradox but a failure to consider confounding variables adequately. Let's quickly invent an example. We are interested in the relationship *hours of muscle training* and *strength* (not based on evidence) in children vs. adults. Within both groups there will be an increasing relationship. The more training, the more muscle strength. But if we combine the groups, we will see a decreasing relationship.

```
library(tidyverse)
n <- 100
age <- c(rep("child", n/2), rep("adult", n/2))
training <- c(rnorm(n/2, 0, 5) + 30, rnorm(n/2, 0, 5)+ 10)
strength <- c(
```

```
  10 + 0.5 * training[1:(n/2)] + rnorm(n/2, 0, 2), # For children
  25 + 0.5 * training[(n/2 + 1):n] + rnorm(n/2, 0, 2) # For adults
)

d <- data.frame(age = age, training = training, strength = strength)

ggplot(d, aes(x = training, y = strength, color = age)) +
  geom_point() +
  geom_smooth(method = "lm", se = FALSE) + # Group-specific regression lines
  geom_smooth(data = d, aes(x = training, y = strength),
              method = "lm", se = FALSE, color = "black", linetype = "dashed", linewidth = 1.2) +
  labs(title = "Regression Lines for Training and Strength",
       x = "Training",
       y = "Strength") +
  theme_minimal() +
  theme(plot.title = element_text(hjust = 0.5))
```

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



- **Group-Specific Trends**:
  - In the group of **children** (blue line), strength **increases** with training, as indicated by the positive slope of the regression line.

  - Similarly, in the group of **adults** (red line), strength also **increases** with training.

- **Overall Trend**:
  - When both groups are combined and the categorical variable *age* is neglected, the overall regression line (black, dashed) shows a **negative slope**, suggesting that **strength decreases** with training.
  - This overall trend is opposite to the trends observed within the individual groups.

- **Why Does This Happen?**
  - This paradox occurs because the relationship between the grouping variable (`age`) and the independent variable (`training`) creates a confounding effect.
  - In this case: Children tend to have higher training values overall, while adults tend to have lower training values. *Age* is associated with both *strength* and *training* and is therefore a *confounder* in the relationship between *training* and *strength*. Not adjusting (=including it in the regression model as predictor) for *age* leads to a misleading association.

This (fictitious) example shows that throwing variables into a multiple regression model without thinking about it, is not a good idea.

4.3 What happens when you just throw variables into multiple regression?

This sub-chapter is **important**. I can guarantee you that not too many applied scientists using regression models know about this.

A first taste of causality.

Richard McElreath has 3 cool examples on github that show what happens in the context of *explanation* if you include variables the wrong way and explains this in a video.

We will look at these below - **the pipe, the fork and the collider**. These are causal graphs showing the relationships between the variables. One is interested in the **effect** of X on Y. In this case, it is truly an *effect*, since we create the models in such a way that changing one variable, changes the other - which is indicated by an arrow in the graph. These graphs are called DAGs - directed acyclic graphs. *Directed* because of the arrows, *acyclic* because there are no cycles in the graph. One nice tool for drawing them is dagitty, which is also implemented in R. The online drawing tool is handy.

Note: Not only can it hurt to add variables to the model in an explanatory (causal) context, but also in a predictive context. The model can become unstable and the predictions can become worse. This is called overfitting.

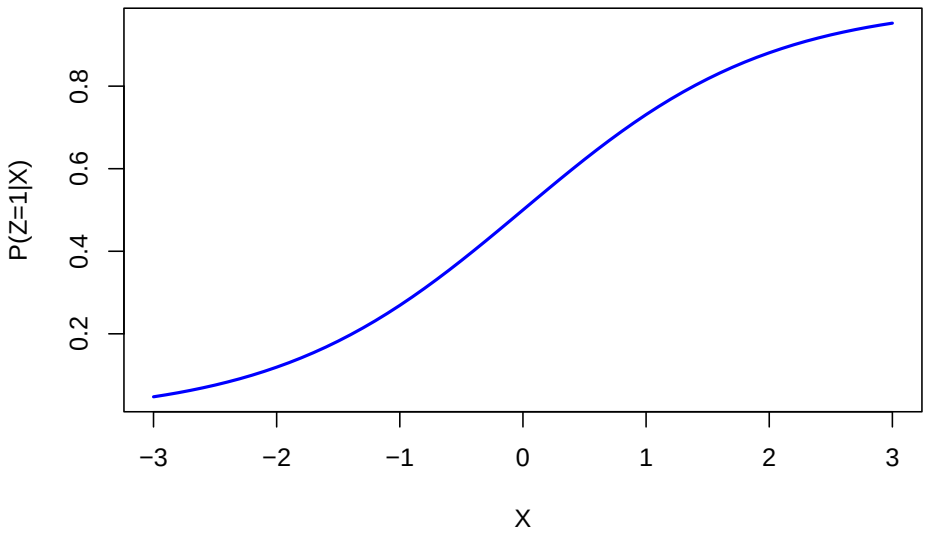
4.3.1 Pipe

In this setting X is associated with Z and Z is associated with Y. X and Y are not directly associated, but through Z (see graph below). If we condition on Z, the association between X and Y *disappears*. This means, if we **know** the value of Z, X does not give us any *additional* information about Y.

This can be seen in the scatterplot: Once we are within $Z = 0$ (black dots) or $Z = 1$ (red dots), X does not give us any information about Y, i.e., the point cloud is horizontal and there is no correlation.

The `inv_logit` function is the inverse of the logit function. It assigns higher probability of Z being 1 if X has a higher value. Let's plot this for understanding:

```
x <- seq(-3, 3, 0.1)
plot(x, inv_logit(x), type = "l", col = "blue", lwd = 2, xlab = "X", ylab = "P(Z=1|X)")
```



Higher X values lead to higher probabilities of $Z = 1$. As you can see, more red dots are on the right side of the scatterplot below.

Now to the pipe and a mini simulation for it:

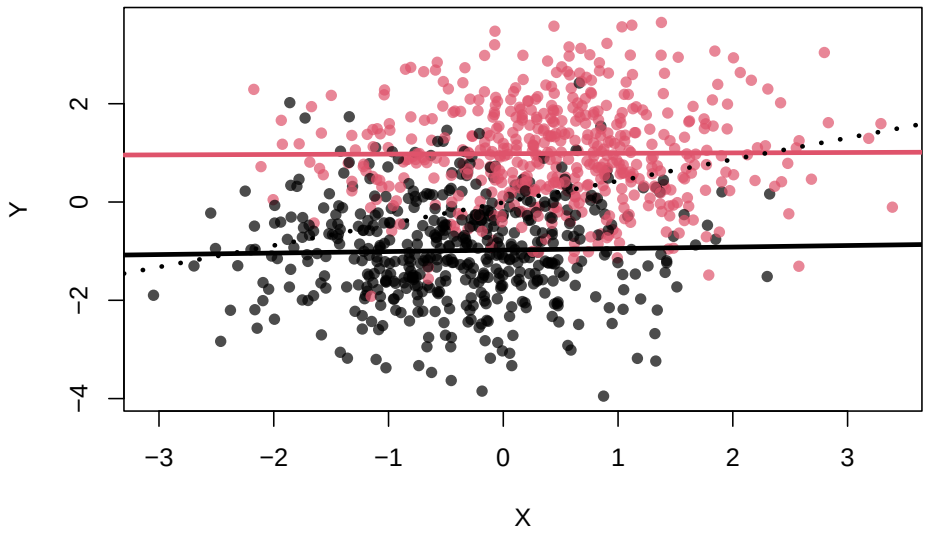
```
# pipe
library(dagitty)
library(tidyverse)
library(ggdag)
```

```
##
## Attaching package: 'ggdag'
```

```
## The following object is masked from 'package:stats':
##
##     filter
```

```
dag <- dagitty( 'dag {
  X -> Z -> Y
}' )

dagitty::coordinates( dag ) <-
  list( x=c(X=0, Y=2, Z=1),
        y=c(X=0, Y=0, Z=0) )

ggdag(dag) +
  theme_dag()
```


```
a <- 0.7
cols <- c( col.alpha(1,a) , col.alpha(2,a) )

# pipe
# X -> Z -> Y
N <- 1000
X <- rnorm(N)
Z <- rbern(N,inv_logit(X))
Y <- rnorm(N,(2*Z-1))

plot( X , Y , col=cols[Z+1] , pch=16 )
abline(lm(Y[Z==1]~X[Z==1]),col=2,lwd=3)
abline(lm(Y[Z==0]~X[Z==0]),col=1,lwd=3)
abline(lm(Y~X),lwd=3,lty=3)
```




```
cor(X[Z==1],Y[Z==1])
```

```
## [1] 0.007540748
```

```
cor(X[Z==0],Y[Z==0])
```

```
## [1] 0.02629344
```

Or in the framework of Simpsons paradox:

```
library(rethinking)

N <- 1000
X <- rnorm(N)
Z <- rbern(N,inv_logit(X))
Y <- rnorm(N,(2*Z-1))
mod1 <- lm(Y ~ X) # without conditioning on Z
summary(mod1)
```

```
##
## Call:
## lm(formula = Y ~ X)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -4.7717 -0.8634 -0.0037  0.9204  3.5437
```

```
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -0.00123    0.04210  -0.029    0.977
## X            0.46041    0.04296  10.718   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.331 on 998 degrees of freedom
## Multiple R-squared:  0.1032, Adjusted R-squared:  0.1023
## F-statistic: 114.9 on 1 and 998 DF,  p-value: < 2.2e-16
```

```
mod2 <- lm(Y ~ X + Z) # with conditioning on Z
summary(mod2)
```

```
##
## Call:
## lm(formula = Y ~ X + Z)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3.2979 -0.6399  0.0011  0.7047  2.8701
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -0.98781    0.04662 -21.190   <2e-16 ***
## X            0.02121    0.03542   0.599    0.549
## Z            1.97980    0.06941  28.523   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.9883 on 997 degrees of freedom
## Multiple R-squared:  0.5062, Adjusted R-squared:  0.5052
## F-statistic:   511 on 2 and 997 DF,  p-value: < 2.2e-16
```

Adding Z to the model (i.e., conditioning on Z) makes the coefficient for X disappear. Knowing Z means that X does not give us any additional information about Y. See exercise 7.

Z is also called a **mediator**.

We could easily verify that Z constitutes a mediator by using Baron and Kenny's 1986 approach (see Wiki):

```r
summary(lm(Y ~ X))# Step 1
```

```
##
## Call:
## lm(formula = Y ~ X)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -4.7717 -0.8634 -0.0037  0.9204  3.5437
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -0.00123    0.04210  -0.029    0.977
## X            0.46041    0.04296  10.718   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.331 on 998 degrees of freedom
## Multiple R-squared:  0.1032, Adjusted R-squared:  0.1023
## F-statistic: 114.9 on 1 and 998 DF,  p-value: < 2.2e-16
```

```r
summary(lm(Z ~ X))# Step 2
```

```
##
## Call:
## lm(formula = Z ~ X)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.97876 -0.40728 -0.00555  0.41502  1.09581
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.49832    0.01425   34.96   <2e-16 ***
## X            0.22184    0.01454   15.25   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.4507 on 998 degrees of freedom
## Multiple R-squared:  0.189,  Adjusted R-squared:  0.1882
## F-statistic: 232.6 on 1 and 998 DF,  p-value: < 2.2e-16
```

```
summary(lm(Y ~ X + Z))# Step 3
```

```
##
## Call:
## lm(formula = Y ~ X + Z)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3.2979 -0.6399  0.0011  0.7047  2.8701
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -0.98781    0.04662 -21.190   <2e-16 ***
## X            0.02121    0.03542   0.599    0.549
## Z            1.97980    0.06941  28.523   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.9883 on 997 degrees of freedom
## Multiple R-squared:  0.5062, Adjusted R-squared:  0.5052
## F-statistic:   511 on 2 and 997 DF,  p-value: < 2.2e-16
```

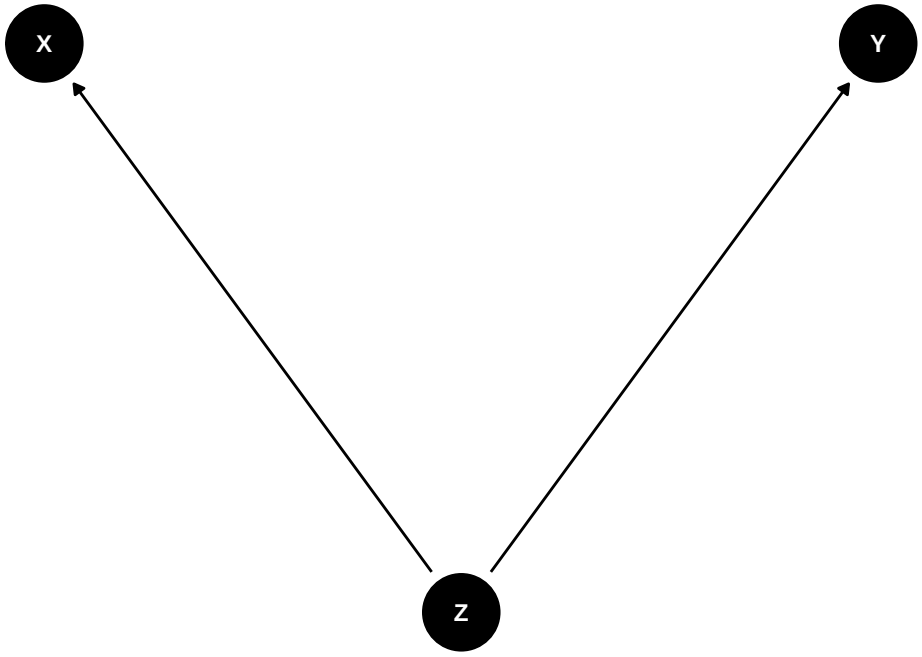
Both coefficients for X are "significant" in steps 1 and 2. The coefficients for Z in step 3 is "significant" and the coefficient for X is smaller compared to step 1. And not to forget: the very small p-values are strongly related to the very large smaple size and a strict cutoff ($\alpha = 0.05$) makes no sense.

Example: *Hours of studying* is associated with *exam performance*. This observation makes sense. A mediator in this context: *understanding/knowledge*. One could study for hours inefficiently with low levels of concentration and achieve low exam performance or one could study for a comparatively low number of hours and achieve good results. If we know that a person has gained the *understanding/knowledge*, the *hours of studying* do not give us any additional information with respect to *exam performance*.

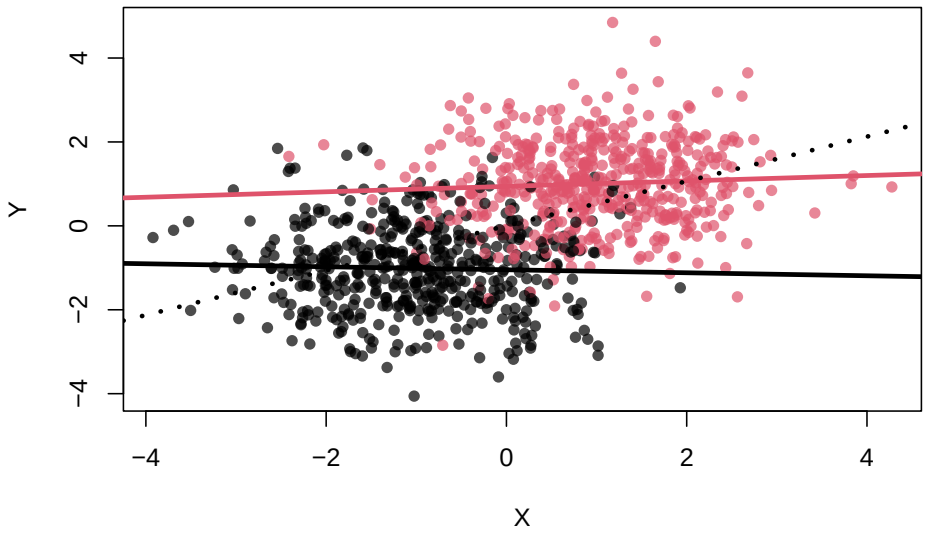
4.3.2 Fork

In health science, this is the classical confounder. Z is associated with X and Y. X and Y are *actually* not associated (no arrow) but an association is still shown: `cor(X,Y) = 0.5097437`. But if we condition on Z, the association between X and Y disappears. The assocation is spurious in this case and *adjusting* for the confounder yields the correct result.

```
# fork
# X <- Z -> Y

dag <- dagitty( 'dag {
  X <- Z -> Y
}' )

dagitty::coordinates( dag ) <-
  list( x=c(X=0, Y=2, Z=1),
        y=c(X=0.5, Y=0.5, Z=0) )

ggdag(dag) + theme_dag()
```




```
N <- 1000
Z <- rbern(N)
X <- rnorm(N,2*Z-1)
Y <- rnorm(N,(2*Z-1))

plot( X , Y , col=cols[Z+1] , pch=16 )
abline(lm(Y[Z==1]~X[Z==1]),col=2,lwd=3)
abline(lm(Y[Z==0]~X[Z==0]),col=1,lwd=3)
abline(lm(Y~X),lwd=3,lty=3)
```




```
cor(X[Z==1],Y[Z==1])
```

```
## [1] 0.05903899
```

```
cor(X[Z==0],Y[Z==0])
```

```
## [1] -0.03401368
```

```
cor(X,Y)
```

```
## [1] 0.5097437
```

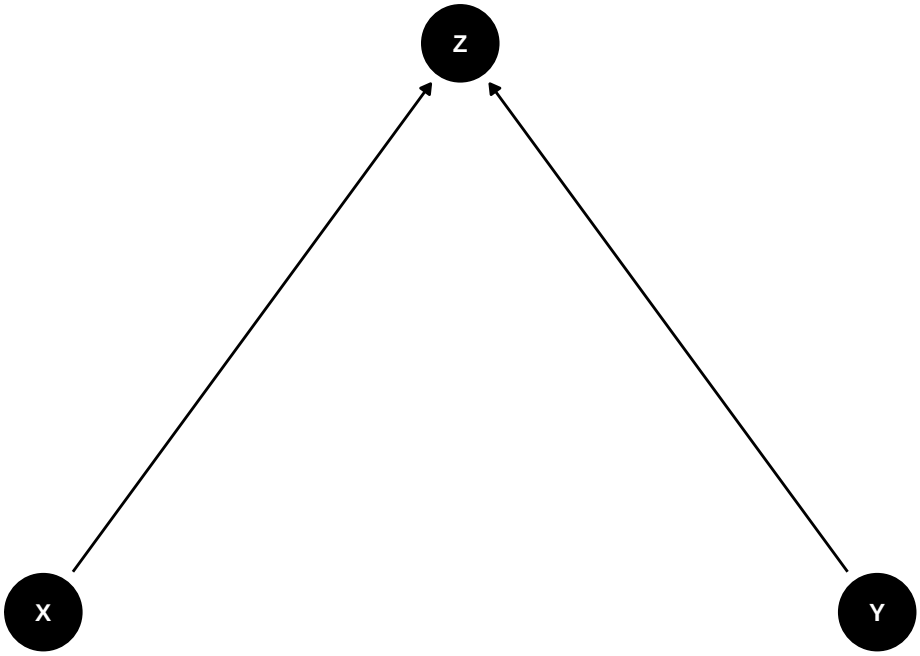
In the example, we know that Z is associated with both X and Y (per construction). As you can see in the definitions of X, Y and Z, X and Y would not be associated weren't it for Z.

Example: *Carrying lighters* is (spuriously) associated with *lung cancer*. Obviously, *carrying a lighter* does *not* cause lung cancer. *Carrying a lighter* is associated with *smoking*, since smokers need to light their cigarettes and how often do non-smokers carry a lighter just for fun? *Smoking* is (causally) associated with *lung cancer*. If you just look at the association between *carrying a ligher* and *lung cancer*, you would find an association. But if you condition on *smoking*, the association disappears. See exercise 13.

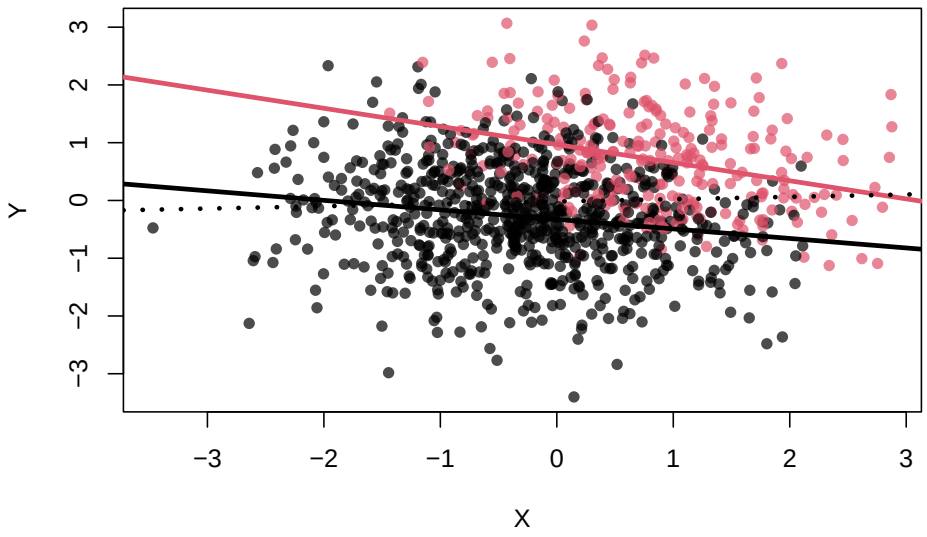
4.3.3 Collider

This is rather interesting. X and Y are independently associated with Z. This can be seen in the toy example below, where X and Y are used for the definition of Z. Z is defined as a **sum score** of X and Y. Sum scores are *very* often used in health sciences (and others). The higher the sum score, the higher the probability that $Z = 1$. Now, *if* we know the value of Z, X and Y are negatively associated (see graph). The reason for this association is that there is a compensatory effect. In order to get a high score, you can either have a high value of X or Y or both. This induces the negative correlation.

```
# collider
# X -> Z <- Y

#dag
dag <- dagitty( 'dag {
  X -> Z <- Y
}' )

dagitty::coordinates( dag ) <-
  list( x=c(X=0, Y=2, Z=1),
        y=c(X=0, Y=0, Z=1) )

ggdag(dag) + theme_dag()
```



```
N <- 1000
X <- rnorm(N)
Y <- rnorm(N)
Z <- rbern(N,inv_logit(2*X+2*Y-2))

plot( X , Y , col=cols[Z+1] , pch=16 )
abline(lm(Y[Z==1]~X[Z==1]),col=2,lwd=3)
abline(lm(Y[Z==0]~X[Z==0]),col=1,lwd=3)
abline(lm(Y~X),lwd=3,lty=3)
```




```
cor(Y[Z==1], X[Z==1])
```

```
## [1] -0.326671
```

```
cor(Y[Z==0], X[Z==0])
```

```
## [1] -0.1768203
```

```
summary(lm(Y ~ X)) # mod1
```

```
##
## Call:
## lm(formula = Y ~ X)
##
## Residuals:
```

```
##     Min      1Q  Median      3Q     Max
## -3.3892 -0.6079 -0.0269  0.6462  3.1038
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -0.01882    0.03055  -0.616    0.538
## X            0.04061    0.02948   1.378    0.169
##
## Residual standard error: 0.9662 on 998 degrees of freedom
## Multiple R-squared:  0.001898,   Adjusted R-squared:  0.0008978
## F-statistic: 1.898 on 1 and 998 DF,  p-value: 0.1686
```

```
summary(lm(Y ~ X + Z)) # mod2
```

```
##
## Call:
## lm(formula = Y ~ X + Z)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -3.03484 -0.55761  0.02022  0.55813  2.41336
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -0.33787    0.03215  -10.51  < 2e-16 ***
## X           -0.19963    0.02906   -6.87 1.13e-11 ***
## Z            1.21398    0.06840   17.75  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.8427 on 997 degrees of freedom
## Multiple R-squared:  0.2415, Adjusted R-squared:   0.24
## F-statistic: 158.7 on 2 and 997 DF,  p-value: < 2.2e-16
```

If you think of Simpsons paradox again, conditioning on Z *creates* an association between X and Y, which would otherwise be independent (by definition). So by learning the value of Z, we can learn something from X about Y. On the one hand, adding Z to the model creates an association where there is none, on the other hand, *prediction* of Y is better with Z in the model! In prediction, we largely do not care about the causal structure. We just want to predict Y as accurately as possible.

The sum score example is also called Berkson's paradox. Let's check if this also works for three variables in a sum score in exercise 10.

4.3.4 Multicollinearity

Westfall section 8.4 and McElreath section 6.1 cover this topic. Multicollinearity means that there is a strong correlation between two or more predictors. It is *perfect* multicollinearity when the correlation is 1. For example, if you accidentally include the same variable twice in the model, or when you include the elements of a sum score and the sum score itself as predictors in the model. R just gives you an *NA* if you do this in `lm`.

4.3.4.1 Example from McElreath 6.1

The code creates body heights and leg lengths (left and right) for 100 people.

`runif` draws a uniform random number between 40 and 50% of height as leg length, hence on average 45%. The slope in the linear regression should therefore be around the average height divided by the average leg length: $\frac{10}{0.45 \cdot 10} \sim 2.2$.

```
library(rethinking)

N <- 100
set.seed(909)
height <- rnorm(N, 10, 2)
leg_prop <- runif(N, 0.4, 0.5)
leg_left <- leg_prop * height + rnorm(N, 0, 0.02)
leg_right <- leg_prop * height + rnorm(N, 0, 0.02)

d <- data.frame(height = height, leg_left = leg_left, leg_right = leg_right)
cor(d$leg_left, d$leg_right)
```

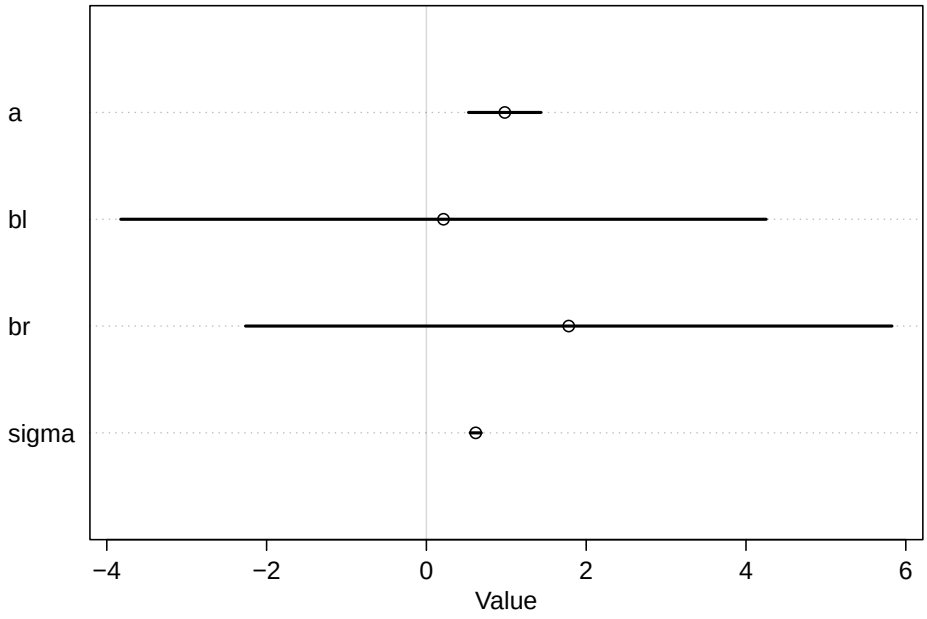
```
## [1] 0.9997458
```

```
m4.6 <- quap(
  alist(
    height ~ dnorm(mu, sigma),
    mu <- a + bl * leg_left + br * leg_right,
    a ~ dnorm(10, 100),
    bl ~ dnorm(2, 10),
    br ~ dnorm(2, 10),
    sigma ~ dexp(1)
  ),    data = d
)
precis(m4.6)
```

```
##            mean         sd       5.5%     94.5%
```

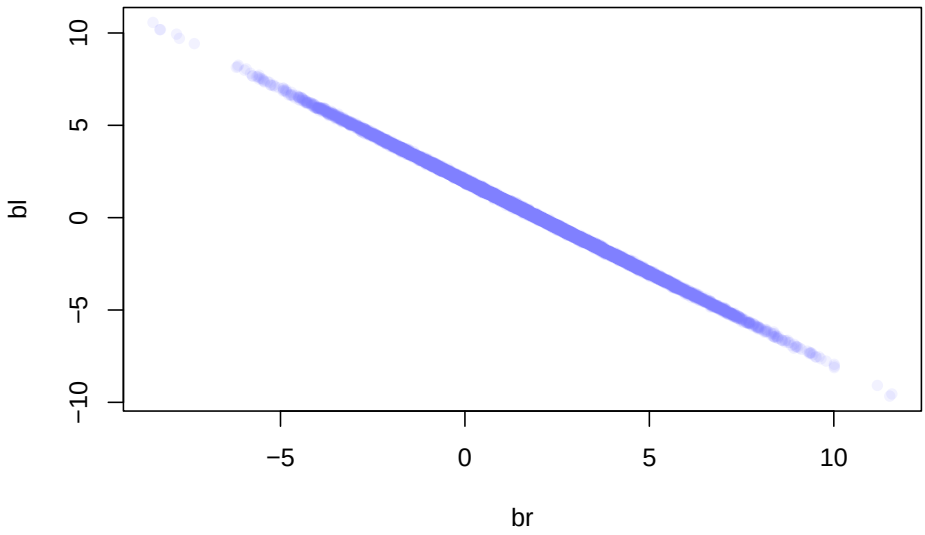
```
## a     0.9811938 0.28396068  0.5273698 1.4350178
## bl    0.2138475 2.52707954 -3.8249137 4.2526087
## br    1.7817046 2.53129314 -2.2637907 5.8271999
## sigma 0.6171141 0.04343629  0.5476945 0.6865337
```

```
plot(precis(m4.6))
```



With the given regression model, one asks the question: "What is the value of knowing each leg's length, after already knowing the other leg's length?" The answer is: "Not much.", since they are highly correlated. Both coefficients are not around the expected β and the credible intervals are wide and include the credible value 0.

```
post <- extract.samples(m4.6)
plot( bl ~ br, post, col=col.alpha(rangi2,0.1), pch = 16 )
```



Since the two coefficients are almost perfectly multicollinear (`cor(d$leg_left, d$leg_right)=0.9997458`), knowing one leg's length does not give us any additional information about the other leg's length. Leaving out the right leg length would give the correct result (exercise 11).

4.3.4.2 Example from Westfall 8.4

In the example below, the second prector $X2$ is a perfect linear function of the first predictor $X1$.

```
# Westfall 8.4.
set.seed(12345)
X1 = rnorm(100)
X2 = 2*X1 -1      # Perfect collinearity
Y = 1 + 2*X1 + 3*X2 + rnorm(100,0,1)
summary(lm(Y~X1+X2))
```

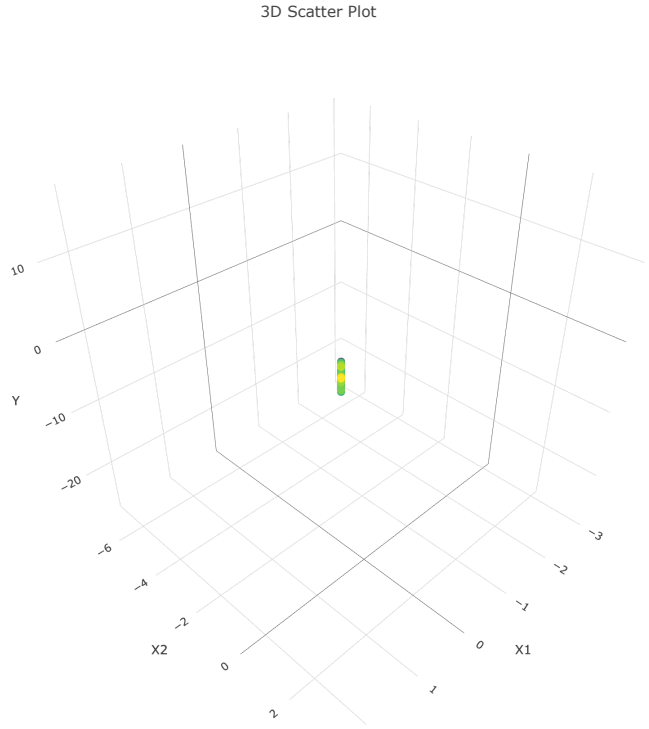
```
##
## Call:
## lm(formula = Y ~ X1 + X2)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2.20347 -0.60278 -0.01114  0.61898  2.60970
##
## Coefficients: (1 not defined because of singularities)
##             Estimate Std. Error t value Pr(>|t|)
```

```
## (Intercept) -1.97795    0.10353  -19.11   <2e-16 ***
## X1           8.09454    0.09114   88.82   <2e-16 ***
## X2                NA         NA      NA       NA
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.011 on 98 degrees of freedom
## Multiple R-squared:  0.9877, Adjusted R-squared:  0.9876
## F-statistic:  7888 on 1 and 98 DF,  p-value: < 2.2e-16
```

Let's look at a 3D plot:

```r
library(plotly)

# Generate data
set.seed(42)
X1 <- rnorm(100)
X2 <- 2 * X1 - 1  # Perfect collinearity
Y  <- 1 + 2 * X1 + 3 * X2 + rnorm(100, 0, 1)

# Create the 3D scatter plot with vertical lines
plot_ly() %>%
  add_markers(x = X1, y = X2, z = Y,
              marker = list(color = Y, colorscale = "Viridis", size = 5),
              name = "Data Points") %>%
  layout(title = "3D Scatter Plot",
         scene = list(xaxis = list(title = "X1"),
                      yaxis = list(title = "X2"),
                      zaxis = list(title = "Y")))
```

```
## file:////private/var/folders/pm/jd6n6gj10371_bml1gh8sc5w0000gn/T/Rtmp1MEKvh/file15e3f6e321546/
```




```
#VIF(lm(Y ~ X1 + X2)) # error
#check_model(lm(Y ~ X1 + X2)) # error
```

There is no unique solution for a plane in this case. Infinitely many planes can be defined using the "line" in space. The problem collapses into the simple linear regression problem. One can just plug in the formula for $X2$ into the model, which yields the identical result:

```
set.seed(12345)
X1 = rnorm(100)
# Y = 1 + 2*X1 + 3*(2*X1 -1) + rnorm(100,0,1) =
Y = -2 + 8*X1 + rnorm(100,0,1)
summary(lm(Y ~ X1))
```

```
##
## Call:
## lm(formula = Y ~ X1)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2.20347 -0.60278 -0.01114  0.61898  2.60970
```

```
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -1.97795    0.10353  -19.11   <2e-16 ***
## X1           8.09454    0.09114   88.82   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.011 on 98 degrees of freedom
## Multiple R-squared:  0.9877, Adjusted R-squared:  0.9876
## F-statistic:  7888 on 1 and 98 DF,  p-value: < 2.2e-16
```

In reality, you only have one parameter ($X1$ or $X2$) in the case of perfect multicollinearity.

4.4 More than 2 predictors

In practice, you often want more than two predictors in the model. If you listen closely to the research question raised by yourself or your colleagues, often it won't be clearly stated what goal you want to achieve with the regression model: prediction or explanation.

If we want to be rigorous about the true relationships between variables, we need to think about the causal structure of the variables, as Richard McElreath argues.

In order not to just throw in variables into our regression model and hope for the best, in order to have a fighting chance, we use the information provided in Chapter 5 in the Statistical Rethinking book.

Overall strategy for modeling:

- **Have a research question** and decide how this question can be answered: Quantitatively (experiment or observational study) or qualitatively. Remember: Statistical modeling is not a catch-all approach to turn data into truth. And: We never want to play *find-the-right-statistical-test-Bingo*. Statistical tests should make sense and not just thrown at data. What does the current literature say about the topic?
- **Define goal**: Prediction or explanation? *Prediction*: Use everything and every model type you can get your hands on to "best" predict the outcome (neural nets, random forests, etc.). *Explanation*: How do the variables relate to each other, how do they influence each other? Temporal relationships are important. And we want interpretable models.
- **Do** exploratory data analysis (**EDA**) to get a feeling for the data:

  - This includes (among others) scatterplots, histograms, boxplots, correlation matrices, etc.
  - How are the raw correlations/associations between the variables?
  - How are the variables distributed?
  - Are there outliers?
  - Are there missing values and why are the values missing?
  - Think about the data generating process. How did the data come about?
  - There is no shame in fitting exploratory models too.

- **Draw a DAG** (directed acyclic graph) of the hypthesized relationships between the variables, even though we will not do formal causal inference in this lecture yet (hopefully in the next). The drawn DAG has testable implications (more below). See 5.1.2 in Rethinking. Nicely enough, `dagitty`spits out:

  - The adjustment sets for the regression model based on a DAG. Which variables should I include as covariates in my model, *if* the DAG is correct?
  - The implied conditional dependencies coming from the DAG. These can be checked with the data. It is not a proof that we have the true relationships depicted by the DAG, but it is a good start.

- **Decide on a statistical model** (more on that later).

- **Define priors** for the parameters of the model: If we do not know much, we choose vague priors (i.e., wide range of plausible parameter values). In the best case, priors should be well argued for, especially in a *low-data* setting (if we do not have many observations).

- **Prior predictive checks**: Does the model produce outcomes (Y) that are at all plausible? If not, we might have to rethink the model and/or priors.

- **Fit and check the model**: If the models makes certain assumptions, check if these are met. For the classical regression model, see Chapter 4 in Westfall. Posterior predictive check: Check if the model produces new data that looks like the observed data.

- **Interpret and report** the results.

4.4.1 Example in NHANES data

We will now try to invent a not too exotic example with NHANES data. The National Health and Nutrition Examination Survey (NHANES) is a large, ongoing study conducted by the CDC to assess the health and nutritional status of

the U.S. population. It combines interviews, physical examinations, and laboratory tests to collect data on demographics, diet, chronic diseases, and physical activity. NHANES uses a complex, nationally representative sampling design, making it a valuable resource for public health research and policy development.

Open data is **great** and NHANES provides all the data to download for free without any restrictions or even a registration. For convenience, Randall Pruim created an R package called `NHANES` that contains a cleaned-up version of the NHANES data from 2009-2012. We will use this and ignore the complex sampling design for now.

```
#install.packages("NHANES")
library(pacman)
p_load(NHANES, tidyverse)
data(NHANES)
head(NHANES)
```

```
## # A tibble: 6 x 76
##      ID SurveyYr Gender   Age AgeDecade AgeMonths Race1 Race3 Education
##   <int> <fct>    <fct>  <int> <fct>         <int> <fct> <fct> <fct>
## 1 51624 2009_10  male      34 " 30-39"        409 White <NA>  High School
## 2 51624 2009_10  male      34 " 30-39"        409 White <NA>  High School
## 3 51624 2009_10  male      34 " 30-39"        409 White <NA>  High School
## 4 51625 2009_10  male       4 " 0-9"           49 Other <NA>  <NA>
## 5 51630 2009_10  female    49 " 40-49"        596 White <NA>  Some College
## 6 51638 2009_10  male       9 " 0-9"          115 White <NA>  <NA>
## # i 67 more variables: MaritalStatus <fct>, HHIncome <fct>, HHIncomeMid <int>,
## #   Poverty <dbl>, HomeRooms <int>, HomeOwn <fct>, Work <fct>, Weight <dbl>,
## #   Length <dbl>, HeadCirc <dbl>, Height <dbl>, BMI <dbl>,
## #   BMICatUnder20yrs <fct>, BMI_WHO <fct>, Pulse <int>, BPSysAve <int>,
## #   BPDiaAve <int>, BPSys1 <int>, BPDia1 <int>, BPSys2 <int>, BPDia2 <int>,
## #   BPSys3 <int>, BPDia3 <int>, Testosterone <dbl>, DirectChol <dbl>,
## #   TotChol <dbl>, UrineVol1 <int>, UrineFlow1 <dbl>, UrineVol2 <int>, ...
```

```
unique(NHANES$SurveyYr)
```

```
## [1] 2009_10 2011_12
## Levels: 2009_10 2011_12
```

```
colnames(NHANES)
```

```
##  [1] "ID"              "SurveyYr"        "Gender"          "Age"
##  [5] "AgeDecade"       "AgeMonths"       "Race1"           "Race3"
##  [9] "Education"       "MaritalStatus"   "HHIncome"        "HHIncomeMid"
```

```
## [13] "Poverty"          "HomeRooms"        "HomeOwn"          "Work"
## [17] "Weight"           "Length"           "HeadCirc"         "Height"
## [21] "BMI"              "BMICatUnder20yrs" "BMI_WHO"          "Pulse"
## [25] "BPSysAve"         "BPDiaAve"         "BPSys1"           "BPDia1"
## [29] "BPSys2"           "BPDia2"           "BPSys3"           "BPDia3"
## [33] "Testosterone"     "DirectChol"       "TotChol"          "UrineVol1"
## [37] "UrineFlow1"       "UrineVol2"        "UrineFlow2"       "Diabetes"
## [41] "DiabetesAge"      "HealthGen"        "DaysPhysHlthBad"  "DaysMentHlthBad"
## [45] "LittleInterest"   "Depressed"        "nPregnancies"     "nBabies"
## [49] "Age1stBaby"       "SleepHrsNight"    "SleepTrouble"     "PhysActive"
## [53] "PhysActiveDays"   "TVHrsDay"         "CompHrsDay"       "TVHrsDayChild"
## [57] "CompHrsDayChild"  "Alcohol12PlusYr"  "AlcoholDay"       "AlcoholYear"
## [61] "SmokeNow"         "Smoke100"         "Smoke100n"        "SmokeAge"
## [65] "Marijuana"        "AgeFirstMarij"    "RegularMarij"     "AgeRegMarij"
## [69] "HardDrugs"        "SexEver"          "SexAge"           "SexNumPartnLife"
## [73] "SexNumPartYear"   "SameSex"          "SexOrientation"   "PregnantNow"
```

The variable descriptions can be found here. As an exercise, please do exploratory data analysis; see exercise 14.

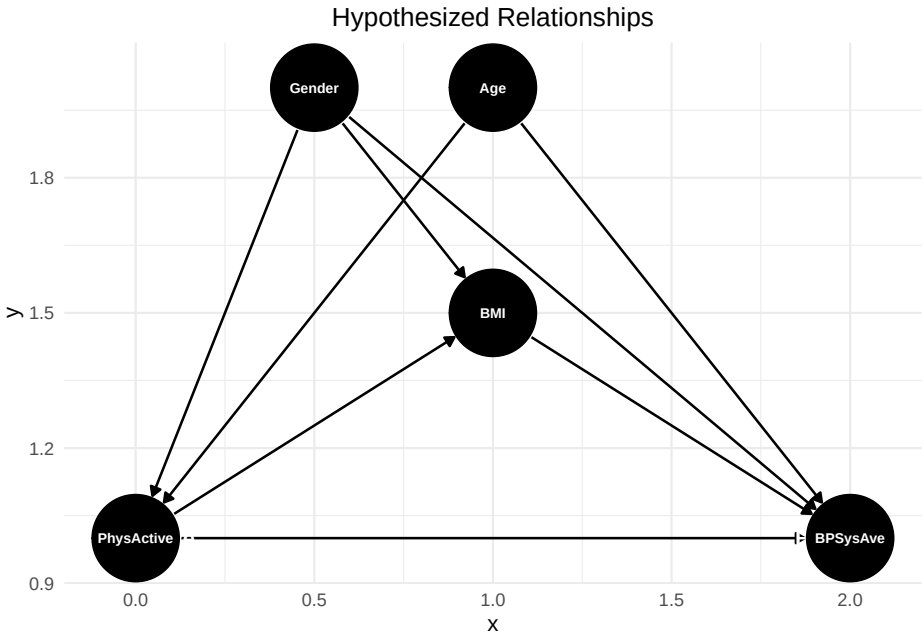
4.4.1.1 Research question

Does *Physical activity* (PhysActive) **influence** the *average systolic blood pressure* (BPSysAve) in adults ($\geq$ 20 years)?

We propose the following relationships among the variables (and limit ourselves to 4 covariates):

```
df <- NHANES # shorter

# Define the DAG
dag <- dagitty('dag {
  PhysActive -> BPSysAve
  Age -> PhysActive
  Age -> BPSysAve
  PhysActive -> BMI
  BMI -> BPSysAve
  Gender -> PhysActive
  Gender -> BMI
  Gender -> BPSysAve
}')

# Set node coordinates for a nice layout
dagitty::coordinates(dag) <- list(
  x = c(PhysActive = 0, BPSysAve = 2, Age = 1, BMI = 1, Gender = 0.5),
```

```
  y = c(PhysActive = 1, BPSysAve = 1, Age = 2, BMI = 1.5, Gender = 2)
)

# Plot the DAG with larger node labels and bubbles
ggdag(dag) +
  theme_minimal() +
  geom_dag_point(size = 20, color = "black") +  # Increase node size
  geom_dag_text(size = 2.5, color = "white") +    # Increase label size
  ggtitle("Hypothesized Relationships") +
  theme(plot.title = element_text(hjust = 0.5))
```



This DAG illustrates the hypothesized relationships between physical activity, age, BMI, gender, and blood pressure (BPSysAve). Physical activity potentially directly influences blood pressure, as regular exercise is hypothesized to improve cardiovascular health and lower blood pressure. Age has a direct effect on blood pressure, as arterial stiffness and other age-related physiological changes contribute to higher blood pressure over time. BMI also plays a role, with higher BMI being associated with increased blood pressure due to greater vascular resistance and metabolic factors. Additionally, gender affects blood pressure, as men and women often have different baseline levels due to hormonal and physiological differences.

There are also indirect pathways that contribute to these relationships. Age influences both physical activity and BMI, as older individuals tend to be less active and may experience weight gain due to metabolic changes (confounder).

BMI and physical activity are also interconnected, physical activity influences BMI, further influencing blood pressure (mediation). Gender plays a role in both BMI and physical activity, as men and women tend to have different average BMI distributions and physical activity levels due to both biological and societal factors (confounder).

Note, that we have a very large sample size (6,919 people). As exercise for later, we randomly select 50 to 100 participants and repeat the analysis. This would also give us the opportunity to study how well we can infer the relationships in the larger sample from the smaller one, which could be an eye-opener.

Which variables should be included in the model?

```
adjustmentSets(dag, exposure = "PhysActive", outcome = "BPSysAve")
```

```
## { Age, Gender }
```

```
impliedConditionalIndependencies(dag)
```

```
## Age _||_ BMI | Gndr, PhyA
## Age _||_ Gndr
```

The one adjustment set (using the command `adjustmentSets`) proposed by `dagitty` tells us that we should include age and gender as covariates in the model (in addition to *PhysActive*), *if* the DAG is correct. Note that **we do not include all three covariables in the model here**, just two of them.

If the DAG is correct, this would imply some conditional independencies (using the command `impliedConditionalIndependencies`) . We should probably check them to see how this is done:

First, we need to test, if age and BMI are independent, *if* we condition on gender and physical activity; i.e., just add the to the regression model. This is the case:

```
# Age _||_ BMI | Gnder, PhyA
summary(lm(Age ~ BMI + Gender + PhysActive, data = df %>% dplyr::filter(Age >= 20)))
```

```
##
## Call:
## lm(formula = Age ~ BMI + Gender + PhysActive, data = df %>% dplyr::filter(Age >=
##     20))
##
## Residuals:
##    Min     1Q Median     3Q    Max
```

```
## -31.53 -13.92  -0.97  12.23  36.67
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)   49.31700    0.94791  52.027  < 2e-16 ***
## BMI            0.04997    0.02982   1.676  0.09380 .
## Gendermale    -1.18815    0.39311  -3.022  0.00252 **
## PhysActiveYes -5.83325    0.39761 -14.671  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 16.62 on 7168 degrees of freedom
##   (63 observations deleted due to missingness)
## Multiple R-squared:  0.03282,    Adjusted R-squared:  0.03242
## F-statistic: 81.09 on 3 and 7168 DF,  p-value: < 2.2e-16
```

The implication can be confirmed.

Interpretation: If we know gender and physical activity (yes/no) of a person, BMI does not offer any *additional* information about age.

Second, I would go out on a limb and say that there is no relationship between age and gender, at least not in a statistically relevant sense.

4.4.1.2 Goal

The **goal** is to better understand (explain) the relationship between physical activity and blood pressure while considering age, BMI and gender. I would suggest that in most cases relevant for us, one wants to understand relationships rather to predict an outcome.

4.4.1.3 EDA

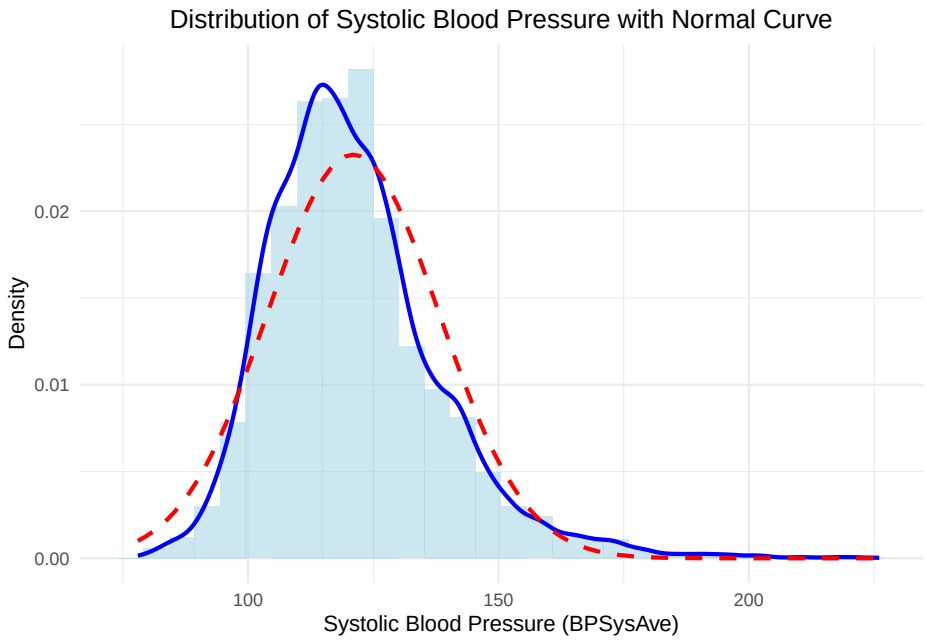
Exercise 14

4.4.1.4 Statistical model

The outcome is a continuous variable (blood pressure), the exposure is a binary variable (physical activity yes/no). Surprise, surprise, we could try a **multiple linear regression model** here.

As a first approximation, let's assume *BPSysAve* is (sufficiently) normally distributed, which can be critized. Let's quickly visualize the distribution of*BPSysAve* and overlay a normal density with paramaters estimated from the data (μ sample mean of *BPSysAve*, σ sample standard deviation of *BPSysAve*):

```r
df_age <- df %>% dplyr::filter(Age >= 20)
df_age %>%
  ggplot(aes(x = BPSysAve)) +
  geom_histogram(aes(y = after_stat(density)),
  bins = 30, fill = "lightblue", alpha = 0.6) +
  geom_density(color = "blue", linewidth = 1) +
  stat_function(
    fun = dnorm,
    args = list(mean = mean(df_age$BPSysAve, na.rm = TRUE),
    sd = sd(df_age$BPSysAve, na.rm = TRUE)),
    color = "red", linewidth = 1, linetype = "dashed"
  ) +  # Theoretical normal curve
  labs(
    x = "Systolic Blood Pressure (BPSysAve)",
    y = "Density",
    title = "Distribution of Systolic Blood Pressure with Normal Curve"
  ) +
  theme_minimal() +
  theme(plot.title = element_text(hjust = 0.5))
```



Not quite normal, but we will assume it for now.

4.4.1.5 Priors

Now, let's write down the statistical model. We choose vague priors since we have a vast data set. The advantage of the Bayesian approach remains (that we have a fully probabilistic model). Admitted, using Bayes is much more technical, at least at first glance.

We center age again, which makes the interpretation of the intercept easier.

- The intercept (β_0) is the expected value of *BPSysAve* for a person of average age, not physically active (reference level of the factor) and female gender (reference level of the factor).
- The coefficient for *PhysActive* (β_1) is the expected difference in *BPSysAve* between a physically active person and a non-physically active person, holding all other predictors constant. *BPSysAve* ranges from 78 to 226 mmHg. If we do not know enough, we can just take a very vague prior for now: $\beta_1 \sim \text{Normal}(0, 50)$. With 95% probability, the effect of physical activity on *BPSysAve* is between -100 and 100.
- $beta_2$: The expected change in *BPSysAve* for a one-year increase in age (above the mean age), holding all other predictors constant. $\beta_2 \sim \text{Normal}(0, 10)$. This effect should be comparatively small, since we are dealing with a one-year increase in age. We could further standardize age, then the effect would be in relation to one standard deviations increase in age.
- σ: The standard deviation of the residuals. $\sigma \sim \text{Uniform}(0, 50)$. We use a vague prior. 95% of the residuals are between -100 and 100. This seems plausible enough.

$$
\begin{aligned}
BPSysAve_i &\sim \text{Normal}(\mu_i, \sigma) \\
\mu_i &= \beta_0 + \beta_1 \cdot PhysActive + \beta_2 \cdot Age_{centered} + \beta_3 \cdot Gender \\
\beta_0 &\sim \text{Normal}(140, 20) \\
\beta_1 &\sim \text{Normal}(0, 50) \\
\beta_2 &\sim \text{Normal}(0, 10) \\
\beta_3 &\sim \text{Normal}(0, 10) \\
\sigma &\sim \text{Uniform}(0, 50)
\end{aligned}
$$

Note that age and gender are not influenced by each other.

4.4.1.6 Prior Predictive Checks

Now we simply draw from the priors and thereby generate systolic blood pressure values.

```
# Prior predictive checks
set.seed(123)
n_sims <- dim(df_age)[1] # 7235 participants
beta_0_vec <- rnorm(n_sims, 140, 20)
beta_1_vec <- rnorm(n_sims, 0, 50)
beta_2_vec <- rnorm(n_sims, 0, 10)
beta_3_vec <- rnorm(n_sims, 0, 10)
sigma_vec <- runif(n_sims, 0, 50)
BPSysAve_sim <- rnorm(n_sims, beta_0_vec + beta_1_vec + beta_2_vec + beta_3_vec, sigma
```

This yields also negative values (please verify) for* BPSysAve*, which is not plausible.

In exercise 15 we will play around with the priors until the prior predictive check producues plausible values. **Admitted**, one should probably not look at the distribution of data while doing this. On the other hand, we could have just Googled the distribution of blood pressure values. This is the result:

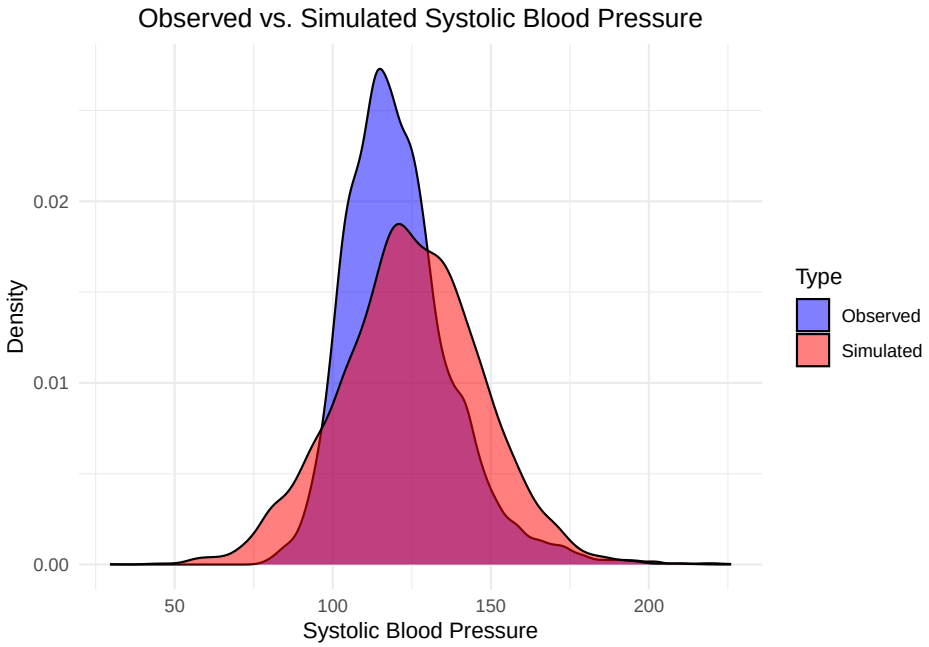
```
set.seed(123)
n_sims <- dim(df_age)[1] # 7235
beta_0_vec <- rnorm(n_sims, 120, 7)
beta_1_vec <- rnorm(n_sims, 0, 10)
beta_2_vec <- rnorm(n_sims, 5, 10)
beta_3_vec <- rnorm(n_sims, 0, 10)
sigma_vec <- runif(n_sims, 0, 20)
BPSysAve_sim <- rnorm(n_sims, beta_0_vec + beta_1_vec + beta_2_vec + beta_3_vec, sigma
length(BPSysAve_sim) # 7235
```

```
## [1] 7235
```

```
df_sim_vs_obs <- data.frame(
  BPSysAve_sim = BPSysAve_sim,
  BPSysAve_obs = df_age$BPSysAve
)

# Combine observed and simulated values into one long-format data frame
df_long <- df_sim_vs_obs %>%
  tidyr::pivot_longer(cols = everything(),
  names_to = "Type", values_to = "BPSysAve") %>%
  mutate(Type = factor(Type, levels = c("BPSysAve_obs", "BPSysAve_sim"),
```

```
  labels = c("Observed", "Simulated")))

# Plot densities of both observed and simulated values
ggplot(df_long, aes(x = BPSysAve, fill = Type)) +
  geom_density(alpha = 0.5) +  # Semi-transparent density curves
  labs(
    x = "Systolic Blood Pressure",
    y = "Density",
    title = "Observed vs. Simulated Systolic Blood Pressure"
  ) +
  scale_fill_manual(values = c("Observed" = "blue", "Simulated" = "red")) +
  theme_minimal() +
  theme(plot.title = element_text(hjust = 0.5))
```

```
## Warning: Removed 264 rows containing non-finite outside the scale range
## (`stat_density()`).
```



Looks decent. Not quite there yet. The blood pressure values have a smaller variances and are skewed. Let's nevertheless fit the model.

4.4.1.7 Fit and check model

We use `quap` for the model fitting.

```
library(data.table)
```

```
##
## Attaching package: 'data.table'
```

```
## The following objects are masked from 'package:lubridate':
##
##     hour, isoweek, mday, minute, month, quarter, second, wday, week,
##     yday, year
```

```
## The following objects are masked from 'package:dplyr':
##
##     between, first, last
```

```
## The following object is masked from 'package:purrr':
##
##     transpose
```

```
df <- NHANES
df <- as.data.table(df)

# Compute age mean for centering
Age_mean <- mean(df[Age >= 20,]$Age, na.rm = TRUE) # adults

# Filter age
df <- df %>% dplyr::filter(Age >= 20) %>%
  dplyr::select(BPSysAve, PhysActive, Age, Gender, BMI) %>%
  drop_na()
dim(df) # 6919    7
```

```
## [1] 6919    5
```

```
sum(is.na(df)) # 0
```

```
## [1] 0
```

```
# Fit model
m_NHANES <- quap(
  alist(
    BPSysAve ~ dnorm(mu, sigma),
    mu <- beta_0 + beta_1[PhysActive] + beta_2 * (Age - Age_mean) + beta_3[Gender],
    beta_0 ~ dnorm(120, 7),
```

```
    beta_1[PhysActive] ~ dnorm(0, 10),
    beta_2 ~ dnorm(5, 10),
    beta_3[Gender] ~ dnorm(0, 10),
    sigma ~ dunif(0, 20)
  ),
  data = df
)

precis(m_NHANES, depth = 2)
```

```
##                  mean         sd        5.5%       94.5%
## beta_0    120.4849312 5.73478808 111.3196322 129.6502302
## beta_1[1]   0.4483506 5.76628806  -8.7672915   9.6639926
## beta_1[2]  -0.2878748 5.76604721  -9.5031319   8.9273823
## beta_2      0.4194979 0.01115025   0.4016776   0.4373181
## beta_3[1]  -1.7776671 5.76694920 -10.9943658   7.4390315
## beta_3[2]   2.7074118 5.76700142  -6.5093703  11.9241939
## sigma      15.4150610 0.13103997  15.2056338  15.6244882
```

```
# We want to know the expected difference for the levels of PhysActive and Gender:
# (which is not directly visible in the summary)
post <- extract.samples(m_NHANES)
post$diff_PhysActive <- post$beta_1[,2] - post$beta_1[,1]
post$diff_G <- post$beta_3[,2] - post$beta_3[,1]

library(conflicted)
conflicts_prefer(posterior::sd)
```

```
## [conflicted] Will prefer posterior::sd over any other package.
```

```
precis(post, depth = 2)
```

```
##                       mean         sd        5.5%       94.5%   histogram
## beta_0          120.5532300 5.79551096 111.3020105 129.8048366
## beta_2            0.4196907 0.01127082   0.4016948   0.4374156
## sigma            15.4161590 0.13110866  15.2076867  15.6279605
## beta_1[1]         0.3487591 5.80968339  -8.9591156   9.5907618
## beta_1[2]        -0.3866548 5.81170379  -9.7471463   8.8316285
## beta_3[1]        -1.7463108 5.77133890 -11.1387559   7.3752075
## beta_3[2]         2.7422693 5.77545948  -6.6404321  11.8947750
## diff_PhysActive  -0.7354140 0.37500248  -1.3324591  -0.1257030
## diff_G            4.4885801 0.37263071   3.8813342   5.0880217
```

The summary tells us that `diff_PhysActive` is -0.74 with a large proportion of the posterior < 0. Please research if this is a clinically relevant difference. My guess is: no.

4.4.1.8 Fit model using `lm` (Frequentist approach)

We can also fit the model using the Frequentist approach. No priors. The results should be very similar due to the large sample size.

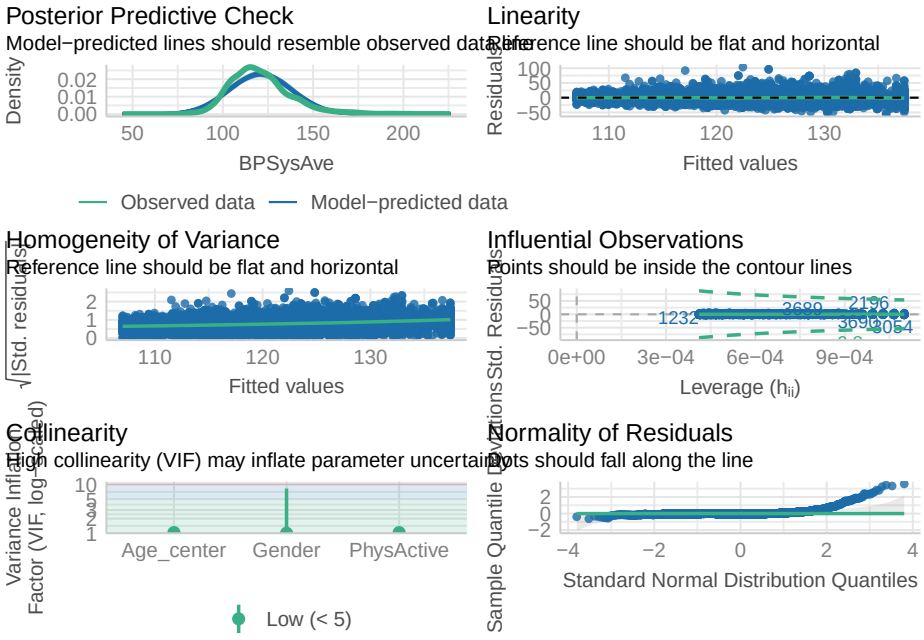
```
library(car)
dim(df)
```

```
## [1] 6919    5
```

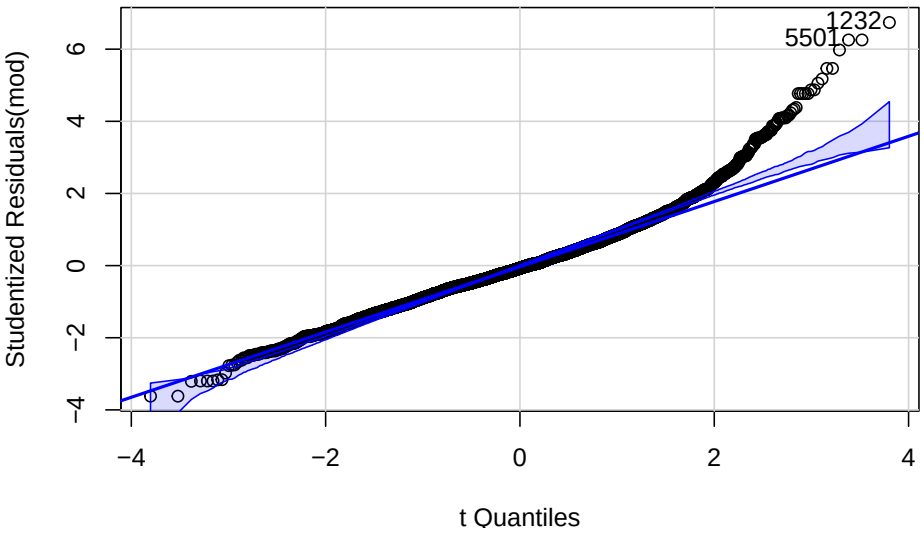
```
df <- df %>% mutate(Age_center = Age - mean(Age))
mod <- lm(BPSysAve ~ PhysActive + Age_center + Gender, data = df)
summary(mod)
```

```
##
## Call:
## lm(formula = BPSysAve ~ PhysActive + Age_center + Gender, data = df)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -55.736  -9.245  -1.160   8.254 103.561
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)   119.14860    0.32495 366.665   <2e-16 ***
## PhysActiveYes  -0.73693    0.37750  -1.952    0.051 .
## Age_center      0.41949    0.01115  37.610   <2e-16 ***
## Gendermale      4.48810    0.37134  12.086   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 15.42 on 6915 degrees of freedom
## Multiple R-squared:  0.1875, Adjusted R-squared:  0.1872
## F-statistic:   532 on 3 and 6915 DF,  p-value: < 2.2e-16
```

```
check_model(mod)
```




```
qqPlot(mod) # bad
```




```
## [1] 1232 5501
```

The results are similar to the Bayesian approach. In the summary output we see that the effect of physical activity is -0.73693 with a small p-value (0.051).

This is one example for the absurdity of strict cutoffs for p-values. The other coefficients including the estimate for σ are also similar. It's alsways a good idea to double-check.

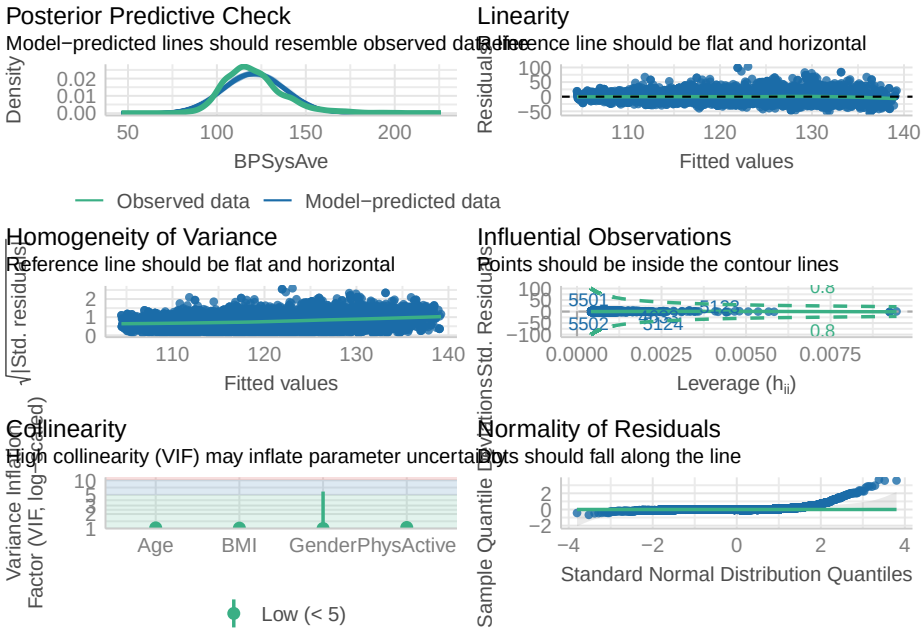
Model fit could be better, the model thinks the blood pressure is normally distributed. There also seems to be some heteroscadasticity and the residuals are not normally distributed.

Usually, one would fit this model by throwing in all variables into the model, including BMI. Let's see if this approach changes the results:

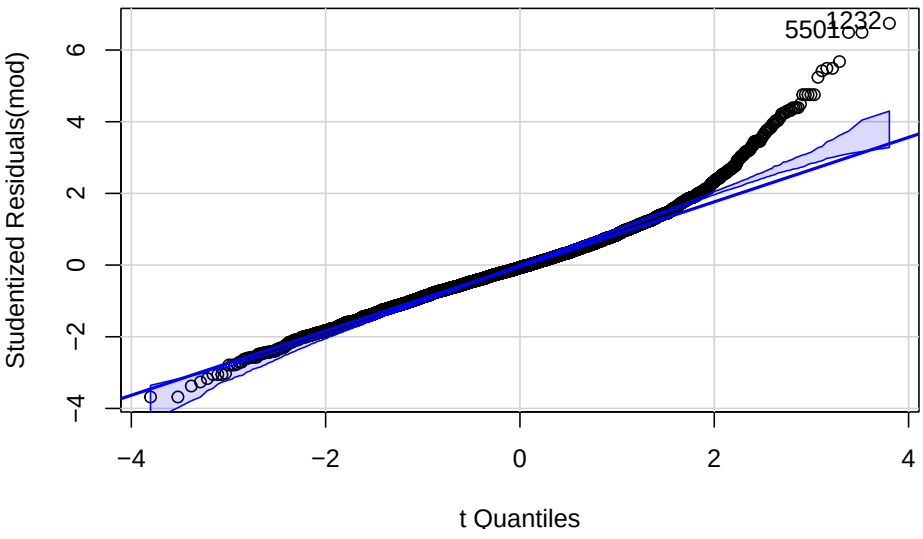
```
mod <- lm(BPSysAve ~ PhysActive + Age + Gender  + BMI, data = df) # add BMI
summary(mod)
```

```
##
## Call:
## lm(formula = BPSysAve ~ PhysActive + Age + Gender + BMI, data = df)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -56.295  -9.414  -1.043   8.004 102.952
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)   91.38776    1.04337  87.589   <2e-16 ***
## PhysActiveYes -0.22870    0.37854  -0.604    0.546
## Age            0.41728    0.01108  37.658   <2e-16 ***
## Gendermale     4.43779    0.36887  12.031   <2e-16 ***
## BMI            0.27237    0.02790   9.764   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 15.32 on 6914 degrees of freedom
## Multiple R-squared:  0.1986, Adjusted R-squared:  0.1981
## F-statistic: 428.3 on 4 and 6914 DF,  p-value: < 2.2e-16
```

```
check_model(mod)
```




```
qqPlot(mod)
```




```
## [1] 1232 5501
```

This changed the results notably. Before, the effect of physical activity was -0.73693 with a rather small p-value, now it is -0.22870 with a p-value ten times as large.

Remember this: The interpretation of a whole paper could change by falsely including an additional variable (in this case BMI).

4.4.1.9 Improve model

Now, that we have seen that the model fit is probably not sufficient, we could try to improve it.

First order of business is the **non-normality** of the blood pressure values (outcome). The model predicts normally distributed values, which is not the case. It seems that the blood pressure values are right-skewed.

Let's try a log-normal distribution for the outcome variable, which only takes positive values and is skewed.

We would **further** try to model σ as a linear function of μ. For higher expected values of *BPSysAve*, we would expect higher variability. Hence, we try to **explicitely model heteroscadasticity**.

Note, that we have no problem with adding parameters to the model because of the large sample size.

See exercise 16.

$$
\begin{aligned}
BPSysAve_i &\sim \text{Log-Normal}(\mu_i, \sigma_i) \\
\sigma_i &= \exp(\beta_4 + \beta_5 \cdot \mu_i) \\
\mu_i &= \beta_0 + \beta_1 \cdot PhysActive + \beta_2 \cdot Age_{centered} + \beta_3 \cdot Gender \\
\beta_0 &\sim \text{Normal}(140, 20) \\
\beta_1 &\sim \text{Normal}(0, 50) \\
\beta_2 &\sim \text{Normal}(0, 10) \\
\beta_3 &\sim \text{Normal}(0, 10) \\
\beta_4 &\sim \text{Normal}(0, 10) \\
\beta_5 &\sim \text{Uniform}(0, 50)
\end{aligned}
$$

We now allow the standard deviation σ_i to be different for different values of the mean. With `exp()` we ensure that σ_i is positive. This time, we assume: $log(BPSysAve_i) \sim \text{Normal}(\mu_i, \sigma_i)$. That's what it means to be log-normally distributed.

Fit improved model

```
sum(is.na(df)) # 0
```

```
## [1] 0
```

```
set.seed(122)
m_NHANES_lnorm <- quap(
  alist(
    BPSysAve ~ dlnorm(lmu, lsd),
    lsd <- exp(beta_4 + beta_5 * lmu),
    lmu <- beta_0 + beta_1[PhysActive] + beta_2 * (Age - Age_mean) + beta_3[Gender],
    beta_0 ~ dnorm(140, 10),  #
    beta_1[PhysActive] ~ dnorm(0, 10),  #
    beta_2 ~ dnorm(0, 10),
    beta_3[Gender] ~ dnorm(0, 10),  #
    beta_4 ~ dnorm(0, 10),  #
    beta_5 ~ dnorm(0, 10)  #
  ),
  data = df,
  start = list(beta_0 = 140, beta_1 = c(0.5, 0), beta_2 = 0, beta_3 = c(0.5, 0), beta_4 = 1, beta
)

precis(m_NHANES_lnorm, depth = 2)
```

```
##                    mean           sd          5.5%         94.5%
## beta_0     69.940815482 7.0688162635  58.643481822  81.238149141
## beta_1[1] -33.089453949 6.1218905701 -42.873417461 -23.305490436
## beta_1[2] -33.092785217 6.1218905660 -42.876748723 -23.308821711
## beta_2      0.003334919 0.0000845877   0.003199732   0.003470107
## beta_3[1] -32.083866696 6.1249525109 -41.872723781 -22.295009611
## beta_3[2] -32.042155145 6.1249525254 -41.831012254 -22.253298037
## beta_4    -14.545840087 0.7396301298 -15.727911886 -13.363768287
## beta_5      2.594743962 0.1546084882   2.347649737   2.841838187
```

```
# difference in levels of PhysActive and Gender (catorical variables):
post_lnorm <- extract.samples(m_NHANES_lnorm)
post_lnorm$diff_PhysActive <- post_lnorm$beta_1[,2] - post_lnorm$beta_1[,1]
post_lnorm$diff_G <- post_lnorm$beta_3[,2] - post_lnorm$beta_3[,1]
precis(post_lnorm, depth = 2)
```

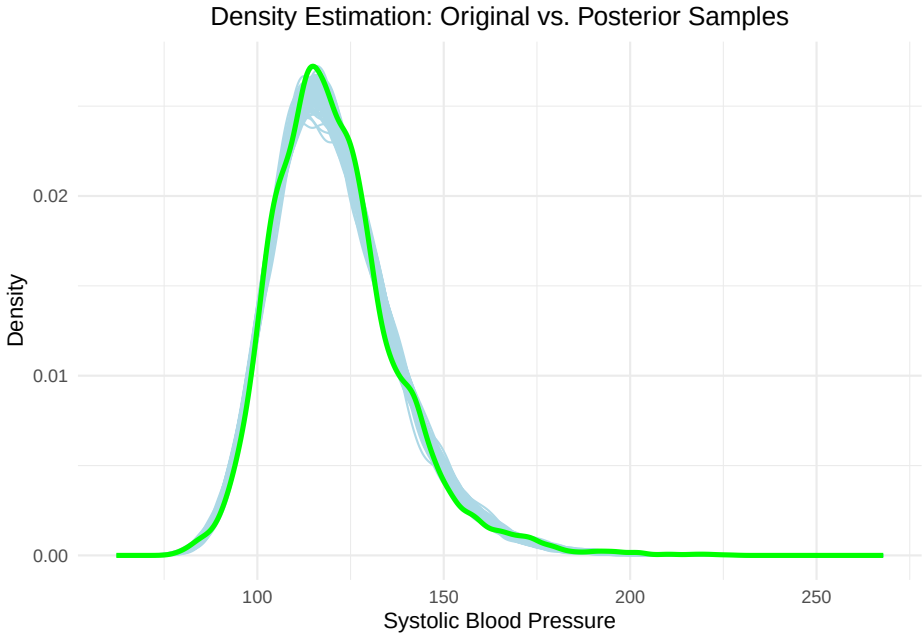
```
##                         mean           sd          5.5%         94.5%
## beta_0           69.802740693 7.085286e+00  58.450093685  8.100613e+01
## beta_2            0.003333900 8.420378e-05   0.003198976  3.469766e-03
## beta_4          -14.557397328 7.340126e-01 -15.730856478 -1.339366e+01
## beta_5            2.597170489 1.534331e-01   2.353734581  2.842822e+00
## beta_1[1]       -33.013184105 6.159037e+00 -42.880222934 -2.329850e+01
## beta_1[2]       -33.016534396 6.159079e+00 -42.884893477 -2.330076e+01
## beta_3[1]       -32.022042725 6.166508e+00 -41.656953623 -2.200235e+01
## beta_3[2]       -31.980336296 6.166495e+00 -41.611161302 -2.196006e+01
```

```
## diff_PhysActive  -0.003350291 2.668752e-03  -0.007636024  9.712922e-04
## diff_G            0.041706430 2.640623e-03   0.037460961  4.588615e-02
##                       histogram
## beta_0
## beta_2
## beta_4
## beta_5
## beta_1[1]
## beta_1[2]
## beta_3[1]
## beta_3[2]
## diff_PhysActive
## diff_G
```

`diff_PhysActive` is now -0.003350291 with a very tight credible interval containing 0 as a credible effect. The model is pretty sure, that the effect is practically zero.

Let's vizualize the posterior predictive distributions of 100 samples:

```
# posterior predictive checks
sample_BP <- sim(m_NHANES_lnorm, n = 1000)

# Convert the first 100 rows of the posterior samples into a long format for ggplot
df_posterior <- as.data.frame(t(sample_BP[1:100,])) %>%
  pivot_longer(cols = everything(), names_to = "Simulation", values_to = "BPSysAve_sim"

# Create the plot
ggplot() +
  geom_density(data = df_posterior, aes(x = BPSysAve_sim, group = Simulation),
               color = "lightblue", alpha = 0.05) +
  geom_density(data = df, aes(x = BPSysAve), color = "green", linewidth = 1.2) +
  labs(title = "Density Estimation: Original vs. Posterior Samples",
       x = "Systolic Blood Pressure",
       y = "Density") +
  theme_minimal() +
  theme(plot.title = element_text(hjust = 0.5))
```



With respect to the distribution of blood pressure, the model looks much better now. Note, that we did not assume that the residuals are normally distributed, neither did we assume homoscedasticity (on the contrary).

See exercise 17.

4.4.1.10 Interpret and report results

Summarizing the results, we find the expected direction of association in our first model. The effect of physical activity on blood pressure was negative. After improving the model fit, we concluded that the small but clinically irrelevant effect from the first model dissapeared. It would be interesting to see how other variables containing more information influence blood pressure, like *PhysActiveDays* (Number of days in a typical week that participant does moderate or vigorousintensity activity).

4.4.2 Concluding remarks

The NHANES example was a somewhat realistic instance of a multiple regression model which could easily be encountered in practice. In a Master thesis, we are often in a low-data setting with 20-150 observations. Here, the priors are much more important.

Exercise 18 asks you to repeat the analysis with a smaller sample size.

We saw that we can readily adapt the model, if assumptions of the classical linear regression model are violated. In our case, we had to model the outcome as log-normal in order to allow for the model to predict skewed blood pressure values. In addition, we explicitly modeled heteroscedasticity by allowing the standard deviation to grow with the mean. Theoretically, one could build arbitrarily complex models this way. Of course, syntax problems and convergence issues could arise. Even in this example some starting values were not fit for purpose, which is why I had to add the `set.seed()` command in the improved model.

4.5 Exercises

[E] Easy, [M] Medium, [H] Hard

(Some) solutions to exercises can be found in the git-repo here.

4.5.1 [M] Exercise 1

- Fit a model with a cubic term for weight and height of the !Kung San people.
- Add the prediction bands as seen in the book.
- Come up with an explanation for the functional form of this relationship.
- Could there be reasons to for taking a less complicated model (1, 2)?

4.5.2 [E] Exercise 2

Consider the model equations from above where we used polynomial regression to model the relationship between weight and height:

- Draw the model hierarchy for the model.

4.5.3 [H] Exercise 3

Invent a data set (or use the first 3 lines of a previous data set) with 3 observations of Y and X_1, X_2 and X_3. You have a data frame with 3 rows and 4 columns.

- Fit a model with Y as the dependent variable and X_1, X_2, X_3 as predictors.
- How big is R^2?
- Could you have calulated this without `lm`and R?

4.5.4 [E] Exercise 4

Go back to the section about the interaction term in the linear model.

- Use the code provided.
- Standardise the predictors. How are the βs changing and what is their interpration now?
- Change the relative sizes of the true but usually unknown βs. What happens to the estimates and the graph?
- What happens if you change the error term and increase or decrease its variance?

4.5.5 [E] Exercise 5

Draw the interaction plot from the section about the interaction plot for the case when there is no interaction, i.e. $\beta_3 = 0$.

4.5.6 [M] Exercise 6

Go back to the model assumptions checks above.

- Create the same two plots for the simple mean model without predictors, just with the intercept.
- Which model fits the data better according to these posterior predictive checks?

4.5.7 [E] Exercise 7

Go back to the Simpson's paradox section.

- Invent your own example for the pipe, fork and collider.

4.5.8 [M] Exercise 8

- Take a data set of your choosing with many columns, say 10 or so, either from the internet or from R (if available)
- Fit a model for an arbitrary outcome, add more and more variables to predict the outcome and verify that the R^2 increases.

4.5.9 [M] Exercise 9

Exponential curve fitting. Go back to the section about adding a transformed predictor in the Frequentist setting. Hint: You can use the `optim` function in R.

- Assume the relationship between weight and height looks like this: $height_i = \alpha + \beta_1 e^{\beta_2 weight_i}$.
- Use R and the least squares method to estimate the parameters α, β_1, β_2.
- Note that the sum of squared errors is this: $\sum_{i=1}^{n}(height_i - \alpha - \beta_1 e^{\beta_2 weight_i})^2$.
- What happens if you do not constrain the parameters β_1 and β_2 to be negative?
- Calculate the R^2 for this model.

4.5.10 [E] Exercise 10

Let's try to verify if Berkson's paradox (which we have mentioned in the collider-section) also works for three variables in a sum score.

- Now, we assume some college admits only applicants in the top 20% of a score consisting of the sum of three variables: W, X, Z (grade point average, math score, verbal score).
- All three scores are individually normally distributed with mean 100 and standard deviation 15.
- Calculate the correlation matrix `cor()` of all students and the admitted students.
- Are any of the three variables correlated?
- Plot a scatterplot of the math score and the verbal score und color the points according to being admitted or not. You can add trendlines for the two groups.

4.5.11 [E] Exercise 11

Go back to the multicollinearity section and the example from McElreath 6.1.

- Verify that the coefficient is correct when leaving out the right leg length from the model.

4.5.12 [M] Exercise 12

Go backt to the interpretation of the intercept in the quadratic height model above.

- Why is the intercept not equal to the sample mean of the heights?

4.5.13 [H] Exercise 13

Go back to the section about the fork and the example about smoking at the end.

- Draw a DAG for this example.
- Create data in R, where you assume that the probability of carrying a lighter is higher in smokers, the probability of lung cancer is higher in smokers.
- Show that the association between carrying a lighter and lung cancer disappears when conditioning on smoking.
- You may also invent an example relevant to the field of physiotherapy.

4.5.14 [H] Exercise 14

- Perform extensive EDA on the NHANES data set implemented in R. See above.

4.5.15 [H] Exercise 15

Go back to the NHANES model above.

- Play around with the priors until the prior predictive check produces plausible values.

4.5.16 [M] Exercise 16

Go to the improved model fit of our NHANES example above.

- Draw the model hierarchy for the model.

4.5.17 [H] Exercise 17

Go back to the improved model fit of our NHANES example above.

- Draw observed vs. model-predicted blood pressure values.
- add the $y = x$ line to the plot.
- Is the fit better compared to the first model?

4.5.18 [H] Exercise 18

Go back to the NHANES example above.

- Repeat the NHANES analysis with a smaller sample size.
- Draw 50-100 rows randomly from the adult NHANES data set we have used for the full analysis.
- What do you observe?

Chapter 5

Reliability and Validity

For this chapter we refer to the book Measurement in Medicine.

I invite you to read the introductory chapters 1 and 2 about concepts, theories and models, and types of measurement.

In general, when conducting a measurement of any sort (laboratory measurements, scores from questionnaires, etc.), we want to be reasonably sure

- that we actually **measure what we intend to measure**; (validity; chapter 6 in the book);
- that the measurement does **not change too much** if the underlying **conditions are the same** (reliability; chapter 5 in the book); and
- that we are able to detect a **change** if the underlying conditions change (responsiveness; chapter 7 in the book); and
- that we understand the meaning of a change in the measurement (interpretability; chapter 8 in the book).

In this video, Kai jump starts you on reliability and validity.

5.1 Reliability

You can watch this video to get started.

Imagine, you measure a patient (pick your favorite measurement), for example, the range of motion (ROM) of the shoulder.

- If you are interested in how similar your measurements are in comparison to your colleagues, you are trying to determine the so-called **inter-rater reliability**.

- If you are interested in how similar your measurements are when you measure the same patient twice, you are trying to determine the so-called **intra-rater reliability**.

Assuming there is a true (but unknown) underlying value (of ROM), it is clear that measurements will not be *exactly* the same. Possible influences (potentially) causing different results are:

- the measurement instrument itself (e.g., the goniometer),
- the patient (e.g., mood/motivation),
- the examiner (e.g., mood, influence on patient),
- the environment (e.g., the room temperature).

Note that the **true score** is defined in our context as the average of all measurements if we would measure repeat it an infinite number of times.

5.1.1 Peter and Mary's ROM measurements

The data can be found here. We randomly select 50 measurements from Peter and Mary in 50 different patients, plot their measurements and annotate the absolutely largest one.

```
library(pacman)
p_load(tidyverse, readxl)

# Read file
url <- "https://raw.githubusercontent.com/jdegenfellner/Script_QM2_ZHAW/main/data/chap
temp_file <- tempfile(fileext = ".xls")
download.file(url, temp_file, mode = "wb")  # mode="wb" is important for binary files
df <- read_excel(temp_file)

head(df)
```

```
## # A tibble: 6 x 5
##   patcode ROMnas.Mary ROMnas.Peter ROMas.Mary ROMas.Peter
##     <dbl>       <dbl>        <dbl>      <dbl>       <dbl>
## 1       1          90           92         88          95
## 2       2          82           88         82          90
## 3       3          82           88         57          59
## 4       4          89           89         82          81
## 5       5          80           82         48          40
## 6       6          90           96         99          85
```

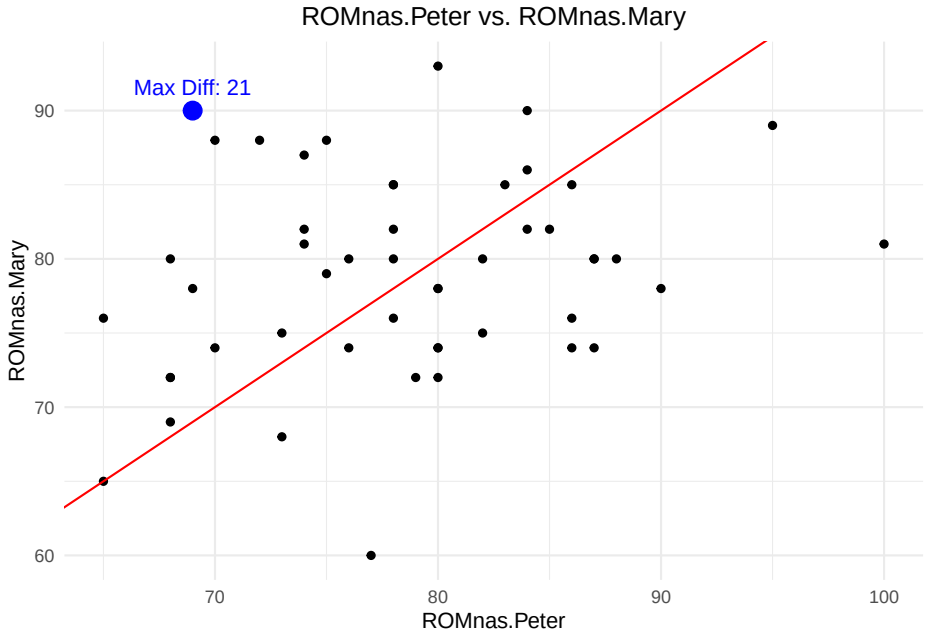
```r
dim(df)
```

```
## [1] 155   5
```

```r
# As in the book, let's randomly select 50 patients.
set.seed(123)
df <- df %>% sample_n(50)
dim(df)
```

```
## [1] 50  5
```

```r
# "as" = affected shoulder
# "nas" = not affected shoulder

df <- df %>%
  mutate(diff = abs(ROMnas.Peter - ROMnas.Mary))  # Compute absolute difference

max_diff_point <- df %>%
  dplyr::filter(diff == max(diff, na.rm = TRUE))  # Find the row with the max difference

df %>%
ggplot(aes(x = ROMnas.Peter, y = ROMnas.Mary)) +
  geom_point() +
  geom_point(data = max_diff_point, aes(x = ROMnas.Peter, y = ROMnas.Mary),
             color = "blue", size = 4) +  # Highlight max difference point
  geom_abline(intercept = 0, slope = 1, color = "red") +
  theme_minimal() +
  ggtitle("ROMnas.Peter vs. ROMnas.Mary") +
  theme(plot.title = element_text(hjust = 0.5)) +
  annotate("text", x = max_diff_point$ROMnas.Peter,
           y = max_diff_point$ROMnas.Mary,
           label = paste0("Max Diff: ", round(max_diff_point$diff, 2)),
           vjust = -1, color = "blue", size = 4)
```




```
# average abs. difference:
mean(df$diff, na.rm = TRUE) # 7.2
```

```
## [1] 7.2
```

```
cor(df$ROMnas.Peter, df$ROMnas.Mary, use = "complete.obs")
```

```
## [1] 0.2403213
```

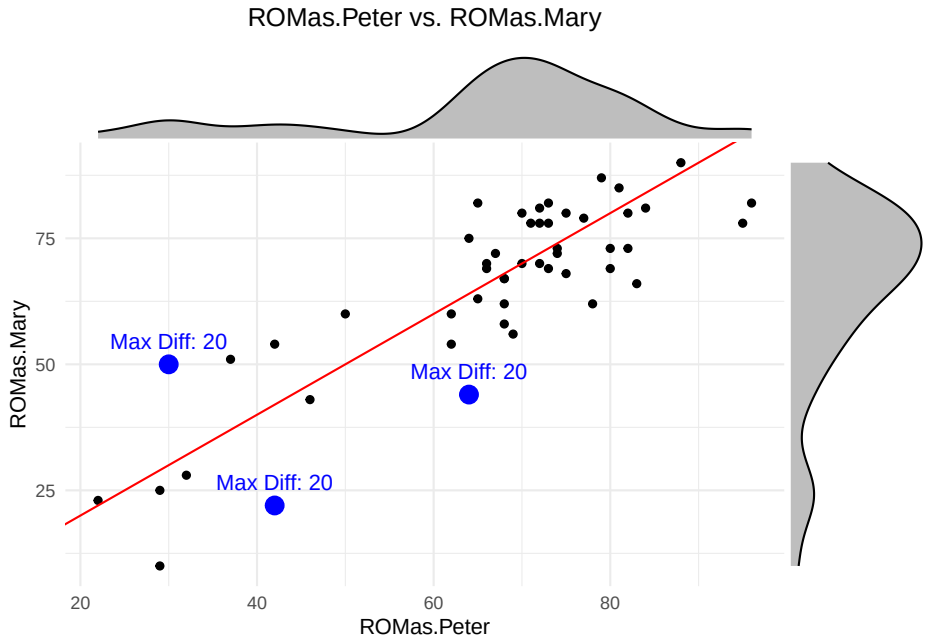
The red line represents the line of equality ($y = x$). If the measurements are exactly the same, all points would lie on this line. The blue point represents the largest difference in measured Range of Motion (ROM) values between Peter and Mary from the randomly chosen 50 people. Note that the maximum difference in all 155 patients is 35 degrees.

The first simple measure of agreement we could use is the correlation, which measures the strength and direction of a linear relationship between two variables. But correlation does not exactly measure what we want. If there was a bias (e.g., Mary systematically measures 5 degrees more than Peter), correlation would not notice this. (-> exercise later…). It actually is too optimistic about the agreement since it only cares about the linearity and not about a potential bias. $r = 0.2403213$ which indicates a weak positive correlation. Higher values of Peter's are associated with higher values of Mary's measurements. But: Knowing Peter's measurement does not help us to *predict* Mary's measurement

at such a low correlation (-> exercise later). So, on the *not affected shoulder* (nas), the agreement is really bad.

What about the affected shoulder (as)?

```
library(ggExtra)
df <- df %>%
  mutate(diff = abs(ROMas.Peter - ROMas.Mary))  # Compute absolute difference

max_diff_point <- df %>%
  dplyr::filter(diff == max(diff, na.rm = TRUE))  # Find the row with the max difference

p <- df %>%
ggplot(aes(x = ROMas.Peter, y = ROMas.Mary)) +
  geom_point() +
  geom_point(data = max_diff_point, aes(x = ROMas.Peter, y = ROMas.Mary),
             color = "blue", size = 4) +  # Highlight max difference point
  geom_abline(intercept = 0, slope = 1, color = "red") +
  theme_minimal() +
  ggtitle("ROMas.Peter vs. ROMas.Mary") +
  theme(plot.title = element_text(hjust = 0.5)) +
  annotate("text", x = max_diff_point$ROMas.Peter,
           y = max_diff_point$ROMas.Mary,
           label = paste0("Max Diff: ", round(max_diff_point$diff, 2)),
           vjust = -1, color = "blue", size = 4)
# Add marginal histograms
ggMarginal(p, type = "density", fill = "gray", color = "black")
```




```
# average abs. difference:
mean(df$diff, na.rm = TRUE) # 7.2
```

```
## [1] 7.78
```

```
cor(df$ROMas.Peter, df$ROMas.Mary, use = "complete.obs")
```

```
## [1] 0.8516653
```

```
# mean difference
mean(df$ROMas.Peter - df$ROMas.Mary, na.rm = TRUE)
```

```
## [1] 1.22
```

In the affected side, the average absolute difference is even larger (7.78) with a maximum absolute difference of 37 degrees, but the correlation is much higher ($r = 0.8516653$). See Figure 5.2 in the book.

Btw, this is an an example for using the correlation coefficient even though the marginal distributions are not normal: There are much more measurements in the higher values around 80 than below, say, 60. But the correlation coefficient makes sense for descriptive purposes.

In this case, knowing Peter's measurement *does* help us to predict Mary's measurement (-> exercise later).

5.1.2 Intraclass Correlation Coefficient (ICC)

One way to measure reliability is to use the intraclass correlation coefficient (ICC).

This measure is based on the idea that observed score Y_i consists of the true score (ROM) and a measurement error (for each person). The proportion of the true score variability to the total variability is the ICC.

In the background one thinks of a statistical model from the Classical Test Theory (CTT). There is

- a true underlying score η_i (for each patient i) and
- an error term $\varepsilon \sim N(0, \sigma_i)$ which is the difference between the true score and
- the observed score Y_i.

$$Y_i = \eta_i + \varepsilon_i$$

It is assumed that η_i and ε_i are independent: ($\mathbb{C}ov(\eta_i, \varepsilon_i) = 0$). This is a nice assumption because now we know (see here) that the variability of the observed score Y_i is just the sum of the variability of the true score η_i and the variability of the error term ε_i:

$$\mathbb{V}ar(Y_i) = \mathbb{V}ar(\eta_i) + \mathbb{V}ar(\varepsilon_i)$$
$$\sigma^2_{Y_i} = \sigma^2_{\eta_i} + \sigma^2_{\varepsilon_i}$$

We want most of the variability in our observed scores Y_i to be explained by the true but unobservable scores η_i. The measurement error ε_i should be be comparatively small. If it is large, we are mostly measuring noise or at least not what we want to measure.

If you either pull two people with the same true but unobservable score η out of the population or measure the same person twice and the score does not change in between, we can **define reliability as correlation between these two measurements**:

$$Y_1 = \eta + \varepsilon_1$$
$$Y_2 = \eta + \varepsilon_2$$

$$cor(Y_1, Y_2) = cor(\eta + \varepsilon_1, \eta + \varepsilon_2) = \frac{Cov(\eta + \varepsilon_1, \eta + \varepsilon_2)}{\sigma_{Y_1}\sigma_{Y_2}} =$$

If we use the properties of the covariance, and the fact that the errors ε_1 and ε_2 are independent, we get:

$$\frac{Cov(\eta,\eta) + Cov(\eta,\varepsilon_2) + Cov(\varepsilon_1,\eta) + Cov(\varepsilon_1,\varepsilon_2)}{\sigma_{Y_1}\sigma_{Y_2}} =$$
$$\frac{\sigma_\eta^2 + 0 + 0 + 0}{\sigma_{Y_1}\sigma_{Y_2}}$$

Since η is a random variable (we draw a person randomly from the population), it is well defined to talk about the variance of η (i.e., σ_η^2). I think this aspect may not come across in the book quite so clearly.

Furthermore, it does not matter if I call the measurement Y_1, Y_2 or more gerneral Y, since they have the same variance and true score:

$$\sigma_Y = \sigma_{Y_1} = \sigma_{Y_2}$$

Hence, it follows that:

$$cor(Y_1, Y_2) = \frac{\sigma_\eta^2}{\sigma_Y^2} = \frac{\sigma_\eta^2}{\sigma_\eta^2 + \sigma_\varepsilon^2}$$

This is the **intraclass correlation coefficient (ICC)**. It is the proportion of the true score variability to the total variability. The ICC is a number between 0 and 1 (think about why!).

Depending on how much deviation from the true but unknown η we throw into the error term ε, you get different versions of the ICC. We will probably stick with the simple versions $ICC_{agreement}$ and $ICC_{consistency}$ here and make sure we understand those.

Let's look again at the term for the ICC above and divide the numerator and the denominator by σ_η^2, which we can do, since it is a positive number:

$$\frac{\sigma_\eta^2}{\sigma_\eta^2 + \sigma_\varepsilon^2} = \frac{1}{1 + \frac{\sigma_\varepsilon^2}{\sigma_\eta^2}}$$

We could call the term $\frac{\sigma_\varepsilon^2}{\sigma_\eta^2}$ the noise-to-signal ratio. The higher this ratio, the lower the ICC. The lower the ratio, the higher the ICC.

- If you increase the noise (measurement error σ_ε^2) for fixed true score variability σ_η^2, the ICC decreases, because the denominator increases.
- If you increase the true score variability σ_η^2 for fixed noiseσ_ε^2, the ICC increases, since the denominator decreases.

Btw, we could also divide by σ_ε^2 and get the signal-to-noise ratio.

At first glance, the following statement seems wrong:

In a very **homogeneous population** (patients have very similar scores/measurements), the **ICC might be very low**. The reason is that the patient variability σ_η^2 is low and you probably have some measurement error σ_ε^2. Hence, if you look at the formula, ICC must be low (for a given measurement error).

On the other hand, if you have a very **heterogeneous population** (patients have rather different scores/measurements), the **ICC might be very high**. The reason is that the patient variability σ_η^2 is high and you probably have some measurement error σ_ε^2.

What matters is the ratio of the two, as can be seen from the formula above.

Let's try to calculate the ICC for our data using a statistical model. There are a couple of different R packages to do this. We will use the `irr` package.

```
library(irr)
```

```
## Loading required package: lpSolve
```

```
irr::icc(as.matrix(df[, c("ROMas.Peter", "ROMas.Mary")]),
    model = "oneway", type = "consistency")
```

```
##  Single Score Intraclass Correlation
##
##    Model: oneway
##    Type : consistency
##
##    Subjects = 50
##      Raters = 2
##      ICC(1) = 0.851
##
##  F-Test, H0: r0 = 0 ; H1: r0 > 0
##    F(49,50) = 12.4 , p = 7.31e-16
##
##  95%-Confidence Interval for ICC Population Values:
##   0.753 < ICC < 0.913
```

We get the result: $ICC(1) = 0.851$.

Since we are regression model experts, we would like to see if we can get the result using the Bayesian framework.

Below is the model structure.

Model details:

- ROM_i is the observed ROM-score for observation i. Every patient has two observations (one each from Mary and Peter). So, for instance $i = 1, 2$ could be patient $ID = 1$.
- μ_i is the expected value of the observed score for patient ID.
- σ_ε is the standard deviation of the measurement error.
- $\alpha[ID]$ is the patient-specific intercept. Since every patient has a different intercept and they come from a normal distribution, we have a **random intercepts model**.
- μ_α is the mean of the prior for the patient-specific intercepts. This is the overall mean of the scores.
- σ_α is the standard deviation of the patient-specific intercepts. **This is the patient variability**! The nice thing about presenting a model in this way is that it's easier to interpret. σ_α says how much the scores of the patients vary in relation to their respective level $\alpha[ID]$.

The ICC is then calculated as

$$\frac{\sigma_\alpha^2}{\sigma_\alpha^2 + \sigma_\varepsilon^2}$$

.

$$\begin{aligned} ROM_i &\sim N(\mu_i, \sigma_\varepsilon) \\ \mu_i &= \alpha[ID] \\ \alpha[ID] &\sim \text{Normal}(\mu_\alpha, \sigma_\alpha) \\ \mu_\alpha &\sim \text{Normal}(66, 20) \\ \sigma_\alpha &\sim \text{Uniform}(0, 20) \\ \sigma_\varepsilon &\sim \text{Uniform}(0, 20) \end{aligned}$$

We did not even notice it, but this was our first **multilevel regression model**. It is multilevel due to the extra layer of patient-specific intercepts. The **observations** are obviously **clustered within patients**, since observations from the **same patient** are **more similar than observations from different patients**.

Draw model structure … exercise..

This time we fire up the `rethinking` package and use the `ulam` function to fit the model. This uses Markov Chain Monte Carlo (MCMC) to sample from the posterior distribution of the parameters.

- The `chains` argument specifies how many chains we want to run. A *chain* is a sequence of points in a space with as many dimensions as there are parameters in the model. It jumps from one point to the next in this parameter space and in doing so, visits the points of the posterior exactly in the correct frequency. Here is an excellent visualization.

- The `cores` argument specifies how many CPU cores we want to use. For larger jobs, one can try to parallelize the chains, which saves some time.

```
library(rethinking)
library(tictoc)

df_long <- df %>%
  mutate(ID = row_number()) %>%
  dplyr::select(ID,ROMas.Peter, ROMas.Mary) %>%
  pivot_longer(cols = c(ROMas.Peter, ROMas.Mary),
               names_to = "Rater", values_to = "ROM") %>%
  mutate(Rater = factor(Rater))

tic()
m5.1 <- ulam(
  alist(
    # Likelihood
    ROM ~ dnorm(mu, sigma),

    # Patient-specific intercepts (random effects)
    mu <- a[ID],
    a[ID] ~ dnorm(mu_a, sigma_ID),  # Hierarchical structure for patients

    # Priors for hyperparameters
    mu_a ~ dnorm(66, 20),  # Population-level mean
    sigma_ID ~ dunif(0,20),  # Between-patient standard deviation
    sigma ~ dunif(0,20)  # Residual standard deviation
  ),
  data = df_long,
  chains = 8, cores = 4
)
```

```
## Running MCMC with 8 chains, at most 4 in parallel, with 1 thread(s) per chain...
##
## Chain 1 Iteration:   1 / 1000 [  0%]  (Warmup)
## Chain 1 Iteration: 100 / 1000 [ 10%]  (Warmup)
## Chain 1 Iteration: 200 / 1000 [ 20%]  (Warmup)
## Chain 1 Iteration: 300 / 1000 [ 30%]  (Warmup)
## Chain 1 Iteration: 400 / 1000 [ 40%]  (Warmup)
## Chain 1 Iteration: 500 / 1000 [ 50%]  (Warmup)
## Chain 1 Iteration: 501 / 1000 [ 50%]  (Sampling)
## Chain 1 Iteration: 600 / 1000 [ 60%]  (Sampling)
## Chain 1 Iteration: 700 / 1000 [ 70%]  (Sampling)
## Chain 1 Iteration: 800 / 1000 [ 80%]  (Sampling)
## Chain 1 Iteration: 900 / 1000 [ 90%]  (Sampling)
```

```
## Chain 1 Iteration: 1000 / 1000 [100%]  (Sampling)

## Chain 1 Informational Message: The current Metropolis proposal is about to be reject

## Chain 1 Exception: normal_lpdf: Scale parameter is 0, but must be positive! (in '/va

## Chain 1 If this warning occurs sporadically, such as for highly constrained variable

## Chain 1 but if this warning occurs often then your model may be either severely ill-

## Chain 1

## Chain 2 Iteration:   1 / 1000 [  0%]  (Warmup)
## Chain 2 Iteration: 100 / 1000 [ 10%]  (Warmup)
## Chain 2 Iteration: 200 / 1000 [ 20%]  (Warmup)
## Chain 2 Iteration: 300 / 1000 [ 30%]  (Warmup)
## Chain 2 Iteration: 400 / 1000 [ 40%]  (Warmup)
## Chain 2 Iteration: 500 / 1000 [ 50%]  (Warmup)
## Chain 2 Iteration: 501 / 1000 [ 50%]  (Sampling)
## Chain 2 Iteration: 600 / 1000 [ 60%]  (Sampling)
## Chain 2 Iteration: 700 / 1000 [ 70%]  (Sampling)
## Chain 2 Iteration: 800 / 1000 [ 80%]  (Sampling)
## Chain 2 Iteration: 900 / 1000 [ 90%]  (Sampling)
## Chain 2 Iteration: 1000 / 1000 [100%]  (Sampling)

## Chain 2 Informational Message: The current Metropolis proposal is about to be reject

## Chain 2 Exception: normal_lpdf: Scale parameter is 0, but must be positive! (in '/va

## Chain 2 If this warning occurs sporadically, such as for highly constrained variable

## Chain 2 but if this warning occurs often then your model may be either severely ill-

## Chain 2

## Chain 3 Iteration:   1 / 1000 [  0%]  (Warmup)
## Chain 3 Iteration: 100 / 1000 [ 10%]  (Warmup)
## Chain 3 Iteration: 200 / 1000 [ 20%]  (Warmup)
## Chain 3 Iteration: 300 / 1000 [ 30%]  (Warmup)
## Chain 3 Iteration: 400 / 1000 [ 40%]  (Warmup)
## Chain 3 Iteration: 500 / 1000 [ 50%]  (Warmup)
## Chain 3 Iteration: 501 / 1000 [ 50%]  (Sampling)
## Chain 3 Iteration: 600 / 1000 [ 60%]  (Sampling)
```

```
## Chain 3 Iteration: 700 / 1000 [ 70%]  (Sampling)
## Chain 3 Iteration: 800 / 1000 [ 80%]  (Sampling)
## Chain 3 Iteration: 900 / 1000 [ 90%]  (Sampling)
## Chain 3 Iteration: 1000 / 1000 [100%]  (Sampling)
## Chain 4 Iteration:   1 / 1000 [  0%]  (Warmup)
## Chain 4 Iteration: 100 / 1000 [ 10%]  (Warmup)
## Chain 4 Iteration: 200 / 1000 [ 20%]  (Warmup)
## Chain 4 Iteration: 300 / 1000 [ 30%]  (Warmup)
## Chain 4 Iteration: 400 / 1000 [ 40%]  (Warmup)
## Chain 4 Iteration: 500 / 1000 [ 50%]  (Warmup)
## Chain 4 Iteration: 501 / 1000 [ 50%]  (Sampling)
## Chain 4 Iteration: 600 / 1000 [ 60%]  (Sampling)
## Chain 4 Iteration: 700 / 1000 [ 70%]  (Sampling)
## Chain 4 Iteration: 800 / 1000 [ 80%]  (Sampling)
## Chain 4 Iteration: 900 / 1000 [ 90%]  (Sampling)
## Chain 4 Iteration: 1000 / 1000 [100%]  (Sampling)

## Chain 4 Informational Message: The current Metropolis proposal is about to be rejected because

## Chain 4 Exception: normal_lpdf: Scale parameter is 0, but must be positive! (in '/var/folders/

## Chain 4 If this warning occurs sporadically, such as for highly constrained variable types lik

## Chain 4 but if this warning occurs often then your model may be either severely ill-conditione

## Chain 4

## Chain 1 finished in 0.1 seconds.
## Chain 2 finished in 0.2 seconds.
## Chain 3 finished in 0.1 seconds.
## Chain 4 finished in 0.1 seconds.
## Chain 5 Iteration:   1 / 1000 [  0%]  (Warmup)
## Chain 5 Iteration: 100 / 1000 [ 10%]  (Warmup)
## Chain 5 Iteration: 200 / 1000 [ 20%]  (Warmup)
## Chain 5 Iteration: 300 / 1000 [ 30%]  (Warmup)
## Chain 5 Iteration: 400 / 1000 [ 40%]  (Warmup)
## Chain 5 Iteration: 500 / 1000 [ 50%]  (Warmup)
## Chain 5 Iteration: 501 / 1000 [ 50%]  (Sampling)
## Chain 5 Iteration: 600 / 1000 [ 60%]  (Sampling)
## Chain 5 Iteration: 700 / 1000 [ 70%]  (Sampling)
## Chain 5 Iteration: 800 / 1000 [ 80%]  (Sampling)
## Chain 5 Iteration: 900 / 1000 [ 90%]  (Sampling)
## Chain 5 Iteration: 1000 / 1000 [100%]  (Sampling)

## Chain 5 Informational Message: The current Metropolis proposal is about to be rejected because
```

```
## Chain 5 Exception: normal_lpdf: Scale parameter is 0, but must be positive! (in '/va

## Chain 5 If this warning occurs sporadically, such as for highly constrained variable

## Chain 5 but if this warning occurs often then your model may be either severely ill-

## Chain 5

## Chain 6 Iteration:   1 / 1000 [  0%]  (Warmup)
## Chain 6 Iteration: 100 / 1000 [ 10%]  (Warmup)
## Chain 6 Iteration: 200 / 1000 [ 20%]  (Warmup)
## Chain 6 Iteration: 300 / 1000 [ 30%]  (Warmup)
## Chain 6 Iteration: 400 / 1000 [ 40%]  (Warmup)
## Chain 6 Iteration: 500 / 1000 [ 50%]  (Warmup)
## Chain 6 Iteration: 501 / 1000 [ 50%]  (Sampling)
## Chain 6 Iteration: 600 / 1000 [ 60%]  (Sampling)
## Chain 6 Iteration: 700 / 1000 [ 70%]  (Sampling)
## Chain 6 Iteration: 800 / 1000 [ 80%]  (Sampling)
## Chain 6 Iteration: 900 / 1000 [ 90%]  (Sampling)
## Chain 6 Iteration: 1000 / 1000 [100%]  (Sampling)

## Chain 6 Informational Message: The current Metropolis proposal is about to be reject

## Chain 6 Exception: normal_lpdf: Scale parameter is 0, but must be positive! (in '/va

## Chain 6 If this warning occurs sporadically, such as for highly constrained variable

## Chain 6 but if this warning occurs often then your model may be either severely ill-

## Chain 6

## Chain 7 Iteration:   1 / 1000 [  0%]  (Warmup)
## Chain 7 Iteration: 100 / 1000 [ 10%]  (Warmup)
## Chain 7 Iteration: 200 / 1000 [ 20%]  (Warmup)
## Chain 7 Iteration: 300 / 1000 [ 30%]  (Warmup)
## Chain 7 Iteration: 400 / 1000 [ 40%]  (Warmup)
## Chain 7 Iteration: 500 / 1000 [ 50%]  (Warmup)
## Chain 7 Iteration: 501 / 1000 [ 50%]  (Sampling)
## Chain 7 Iteration: 600 / 1000 [ 60%]  (Sampling)
## Chain 7 Iteration: 700 / 1000 [ 70%]  (Sampling)
## Chain 7 Iteration: 800 / 1000 [ 80%]  (Sampling)
## Chain 7 Iteration: 900 / 1000 [ 90%]  (Sampling)
## Chain 7 Iteration: 1000 / 1000 [100%]  (Sampling)
```

```
## Chain 8 Iteration:   1 / 1000 [  0%]  (Warmup)
## Chain 8 Iteration: 100 / 1000 [ 10%]  (Warmup)
## Chain 8 Iteration: 200 / 1000 [ 20%]  (Warmup)
## Chain 8 Iteration: 300 / 1000 [ 30%]  (Warmup)
## Chain 8 Iteration: 400 / 1000 [ 40%]  (Warmup)
## Chain 8 Iteration: 500 / 1000 [ 50%]  (Warmup)
## Chain 8 Iteration: 501 / 1000 [ 50%]  (Sampling)
## Chain 8 Iteration: 600 / 1000 [ 60%]  (Sampling)
## Chain 8 Iteration: 700 / 1000 [ 70%]  (Sampling)
## Chain 5 finished in 0.1 seconds.
## Chain 6 finished in 0.1 seconds.
## Chain 7 finished in 0.1 seconds.
## Chain 8 Iteration: 800 / 1000 [ 80%]  (Sampling)
## Chain 8 Iteration: 900 / 1000 [ 90%]  (Sampling)
## Chain 8 Iteration: 1000 / 1000 [100%]  (Sampling)
## Chain 8 finished in 0.1 seconds.
##
## All 8 chains finished successfully.
## Mean chain execution time: 0.1 seconds.
## Total execution time: 0.7 seconds.
```

```
toc() # 7s
```

```
## 7.06 sec elapsed
```

```
precis(m5.1, depth = 2)
```

```
##              mean        sd      5.5%     94.5%      rhat ess_bulk
## a[1]     67.722244 4.9090800 59.884240 75.424230 1.0011883 6619.262
## a[2]     64.111654 4.7840605 56.528447 71.776798 1.0034783 6191.276
## a[3]     86.987105 4.8248501 79.310842 94.721499 1.0021836 6533.691
## a[4]     76.488312 4.8772869 68.688592 84.377506 1.0004241 6088.506
## a[5]     58.652587 4.9146668 50.800481 66.376233 1.0057245 6894.371
## a[6]     69.286318 4.9001887 61.446915 77.135439 1.0023860 6440.228
## a[7]     69.630175 4.7214488 62.093989 77.274188 1.0042853 6102.224
## a[8]     67.364715 4.8073624 59.760551 74.940179 1.0004233 6926.255
## a[9]     70.982913 4.8029119 63.473921 78.838160 1.0021779 6096.437
## a[10]    81.123297 4.6829024 73.463408 88.601603 1.0000833 5882.855
## a[11]    70.525946 4.7789205 62.916758 78.015190 1.0027020 6057.395
## a[12]    42.158971 4.9255574 34.330380 50.142874 1.0052952 5608.430
## a[13]    86.981146 4.8049263 79.173926 94.829309 1.0006401 5697.280
## a[14]    74.665706 4.6060512 67.261235 81.970150 1.0042568 6217.659
## a[15]    76.383198 4.8629608 68.375808 84.099854 1.0019507 5393.146
## a[16]    34.941636 4.9924846 27.056167 43.018477 1.0015714 6091.685
```

```
## a[17]    84.803446 4.7374711 77.144124 92.406944 0.9993908 4742.513
## a[18]    64.997487 4.9643186 56.943773 72.939016 1.0035173 6046.231
## a[19]    63.285164 4.9025730 55.533948 71.178041 1.0013061 5093.825
## a[20]    79.685208 5.0087033 71.658232 87.571054 1.0025818 5732.622
## a[21]    30.284397 4.9534314 22.471539 38.175759 1.0029669 4993.642
## a[22]    62.717165 4.8429270 55.056809 70.508550 1.0042888 5186.047
## a[23]    67.369138 4.7935870 59.691637 75.021504 1.0015468 5712.044
## a[24]    81.364895 4.6500222 73.951702 88.744605 1.0010820 5840.154
## a[25]    54.960936 4.6723859 47.304353 62.472893 1.0031483 6486.986
## a[26]    67.484558 4.8629666 59.694329 75.237344 1.0024718 5631.800
## a[27]    75.662864 4.6822039 68.289372 83.149005 1.0015293 5142.724
## a[28]    81.458669 4.7261292 73.931357 89.044628 1.0007963 6429.453
## a[29]    46.319123 4.9454046 38.515295 54.285881 1.0011816 5884.281
## a[30]    61.257367 4.7772157 53.562560 68.978274 1.0023302 5689.914
## a[31]    23.509215 4.9033830 15.777376 31.510781 1.0033955 4020.937
## a[32]    74.135907 4.9851734 66.120570 81.958539 1.0022234 6227.534
## a[33]    70.503386 4.7654367 62.891725 78.072130 1.0023102 5314.875
## a[34]    76.514259 4.7673813 69.060676 84.004384 1.0056486 6295.756
## a[35]    75.552330 4.7650873 68.063417 83.238239 1.0059815 6458.817
## a[36]    69.318470 4.8028313 61.733604 76.842796 1.0019923 5624.692
## a[37]    49.567778 4.8517137 42.000534 57.348616 1.0001428 5345.948
## a[38]    73.807516 4.8704217 66.012632 81.610347 1.0014698 5593.351
## a[39]    72.757513 4.8058223 65.331446 80.543411 0.9996477 6046.893
## a[40]    45.817525 4.8801249 38.090789 53.659197 1.0017618 5810.945
## a[41]    73.683223 4.8798317 65.754844 81.586076 0.9997154 5788.226
## a[42]    26.096569 5.0020037 18.151000 34.085055 1.0035820 5645.769
## a[43]    32.929828 4.9839276 25.060965 40.934064 1.0009925 4969.622
## a[44]    74.142579 4.8767249 66.482373 81.844996 1.0027673 5518.837
## a[45]    76.914062 4.8315152 69.190795 84.479135 1.0021492 5799.564
## a[46]    73.639372 4.7714194 66.006190 81.297628 1.0014867 5772.895
## a[47]    55.859204 4.8616217 48.241338 63.875677 1.0018038 5633.257
## a[48]    69.603460 4.7802544 61.990560 77.160122 1.0024286 5065.542
## a[49]    72.402414 4.6825454 65.012180 79.784921 1.0015697 5911.021
## a[50]    72.770000 4.8603442 65.014248 80.455220 1.0029222 7919.236
## mu_a     65.555333 2.4626739 61.598423 69.494927 1.0017301 4186.431
## sigma_ID 16.613184 1.6346644 14.010696 19.244560 1.0024463 1915.575
## sigma     7.081393 0.7331837  6.011486  8.331337 1.0004015 2066.538
```

```
post <- extract.samples(m5.1)
var_patients <- mean(post$sigma_ID^2)  # Between-patient variance
var_residual <- mean(post$sigma^2)     # Residual variance
var_patients / (var_patients + var_residual) # ICC
```

```
## [1] 0.8461117
```

```
# 0.846
# not too bad; very close to the result from the irr package
```

In the output from `precis(m5.1, depth = 2)` above we see

- all 50 intercept estimates for each patient: `a[ID]`
- `mu_a`is the overall intercept.
- `sigma_ID` is the **patient variability**.
- `sigma` is the **residual variability**.

We just square the sigmas to get the variances.

The trick to do these calculations by "hand" is to get the variance decomposition correct. We stumbled upon variance decomposition in the context of ANOVA, where we decomposed the total variance into the regression variance and the residual variance. Here, we decompose the total variance into the between-patient variance and the residual variance.

Remember: In the background, there is just a statistical model to predict the outcome. Depending on the predictors, we get different models and probably different ICCs

We can also estimate a **random intercept model** with the `lme4` package using the command `lmer`in the Frequentist framework. No priors.

```
library(lme4)
```

```
## Loading required package: Matrix
```

```
##
## Attaching package: 'Matrix'
```

```
## The following objects are masked from 'package:tidyr':
##
##     expand, pack, unpack
```

```
m5.2 <- lmer(ROM ~ (1|ID), data = df_long)
summary(m5.2)
```

```
## Linear mixed model fit by REML ['lmerMod']
## Formula: ROM ~ (1 | ID)
##    Data: df_long
##
## REML criterion at convergence: 791
```

```
##
## Scaled residuals:
##      Min       1Q   Median       3Q      Max
## -1.91875 -0.44821  0.00964  0.51325  1.47941
##
## Random effects:
##  Groups   Name        Variance Std.Dev.
##  ID       (Intercept) 270.99   16.462
##  Residual              47.35    6.881
## Number of obs: 100, groups:  ID, 50
##
## Fixed effects:
##             Estimate Std. Error t value
## (Intercept)   65.590      2.428   27.02
```

```
print(VarCorr(m5.2), comp = "Variance")
```

```
##  Groups   Name        Variance
##  ID       (Intercept) 270.99
##  Residual              47.35
```

```
# ICC =
270.99 / (270.99 + 47.35) #
```

```
## [1] 0.8512597
```

```
# 0.8512597
# -> exactly the same result as the irr package
```

So far, we have only looked at the $ICC_{consistency}$ (see also page 106 in the book). There, we have not yet explicitely considered a bias (=systematic difference between the raters) that the raters could introduce. In the book, they introduce a bias of 5 degrees (Mary measures 5 degrees more than Peter on average).

There is also the $ICC_{agreement}$, which explicitely considers this difference that could occur between the raters. This results in an extra term in the demoniator of the ICC, an additional variance component:

$$ICC_{agreement} = \frac{\sigma_\alpha^2}{\sigma_\alpha^2 + \sigma_\varepsilon^2 + \sigma_{rater}^2}$$

where σ_{rater}^2 is the variance due to systematic rater differences.

We will now introduce the 5 degree bias and use our Bayesian framework to estimate the $ICC_{agreement}$. By introducing a bias, we should see a lower ICC

(agreement). Note, that the prediction quality of Mary's scores given Peter's scores should not change, since we would only shift Mary's scores down by 5 degrees, which would not disturb the linear regression model. We can always move around the points to where we want them to be. We do that for instance when we scale or standardize the data.

Anyhow, let's try to give the model equations for the new model considering the introduced bias:

$$
\begin{aligned}
ROM_i &\sim N(\mu_i, \sigma_\varepsilon) \\
\mu_i &= \alpha[ID] + \beta[Rater] \\
\alpha[ID] &\sim \text{Normal}(\mu_\alpha, \sigma_\alpha) \\
\beta[Rater] &\sim \text{Normal}(0, \sigma_\beta) \\
\mu_\alpha &\sim \text{Normal}(66, 20) \\
\sigma_\alpha &\sim \text{Exp}(0.5) \\
\sigma_\beta &\sim \text{Exp}(1) \\
\sigma_\varepsilon &\sim \text{Exp}(1)
\end{aligned}
$$

Draw model structure … exercise..

```
df_long_bias <- df_long %>%
  mutate(ROM = ROM + ifelse(Rater == "ROMas.Mary", 5, 0))
head(df_long_bias)
```

```
## # A tibble: 6 x 3
##      ID Rater         ROM
##   <int> <fct>       <dbl>
## 1     1 ROMas.Peter    66
## 2     1 ROMas.Mary     75
## 3     2 ROMas.Peter    65
## 4     2 ROMas.Mary     68
## 5     3 ROMas.Peter    96
## 6     3 ROMas.Mary     87
```

```
library(rethinking)
set.seed(123)
m5.2 <- ulam(
  alist(
    # Likelihood
    ROM ~ dnorm(mu, sigma_eps),

    # Model for mean ROM with patient and rater effects
    mu <- alpha[ID] + beta[Rater],

    # Patient-specific random effects
```

```
    alpha[ID] ~ dnorm(mu_alpha, sigma_alpha),

    # Rater effect (Peter/Mary)
    beta[Rater] ~ dnorm(0, sigma_beta),

    # Priors for hyperparameters
    mu_alpha ~ dnorm(66, 10),  # Population mean ROM
    sigma_alpha ~ dexp(0.5),  # Between-patient SD (less aggressive shrinkage)
    sigma_beta ~ dexp(1),   # Rater SD (better regularization)
    sigma_eps ~ dexp(1)     # Residual SD (prevents over-shrinkage)
  ),
  data = df_long_bias,
  chains = 8, cores = 4
)
```

```
## Running MCMC with 8 chains, at most 4 in parallel, with 1 thread(s) per chain...
##
## Chain 1 Iteration:   1 / 1000 [  0%]  (Warmup)
## Chain 1 Iteration: 100 / 1000 [ 10%]  (Warmup)
## Chain 1 Iteration: 200 / 1000 [ 20%]  (Warmup)
## Chain 1 Iteration: 300 / 1000 [ 30%]  (Warmup)
## Chain 1 Iteration: 400 / 1000 [ 40%]  (Warmup)
## Chain 1 Iteration: 500 / 1000 [ 50%]  (Warmup)
## Chain 1 Iteration: 501 / 1000 [ 50%]  (Sampling)
## Chain 1 Iteration: 600 / 1000 [ 60%]  (Sampling)

## Chain 1 Informational Message: The current Metropolis proposal is about to be reject

## Chain 1 Exception: normal_lpdf: Scale parameter is 0, but must be positive! (in '/va

## Chain 1 If this warning occurs sporadically, such as for highly constrained variable

## Chain 1 but if this warning occurs often then your model may be either severely ill-

## Chain 1

## Chain 2 Iteration:   1 / 1000 [  0%]  (Warmup)
## Chain 2 Iteration: 100 / 1000 [ 10%]  (Warmup)
## Chain 2 Iteration: 200 / 1000 [ 20%]  (Warmup)

## Chain 2 Informational Message: The current Metropolis proposal is about to be reject

## Chain 2 Exception: normal_lpdf: Scale parameter is 0, but must be positive! (in '/va
```

```
## Chain 2 If this warning occurs sporadically, such as for highly constrained variable types lik

## Chain 2 but if this warning occurs often then your model may be either severely ill-conditione

## Chain 2

## Chain 3 Iteration:   1 / 1000 [  0%]  (Warmup)
## Chain 3 Iteration: 100 / 1000 [ 10%]  (Warmup)
## Chain 3 Iteration: 200 / 1000 [ 20%]  (Warmup)
## Chain 4 Iteration:   1 / 1000 [  0%]  (Warmup)
## Chain 4 Iteration: 100 / 1000 [ 10%]  (Warmup)
## Chain 4 Iteration: 200 / 1000 [ 20%]  (Warmup)
## Chain 4 Iteration: 300 / 1000 [ 30%]  (Warmup)

## Chain 4 Informational Message: The current Metropolis proposal is about to be rejected because

## Chain 4 Exception: normal_lpdf: Scale parameter is 0, but must be positive! (in '/var/folders/

## Chain 4 If this warning occurs sporadically, such as for highly constrained variable types lik

## Chain 4 but if this warning occurs often then your model may be either severely ill-conditione

## Chain 4

## Chain 1 Iteration: 700 / 1000 [ 70%]  (Sampling)
## Chain 1 Iteration: 800 / 1000 [ 80%]  (Sampling)
## Chain 1 Iteration: 900 / 1000 [ 90%]  (Sampling)
## Chain 1 Iteration: 1000 / 1000 [100%]  (Sampling)
## Chain 2 Iteration: 300 / 1000 [ 30%]  (Warmup)
## Chain 2 Iteration: 400 / 1000 [ 40%]  (Warmup)
## Chain 2 Iteration: 500 / 1000 [ 50%]  (Warmup)
## Chain 2 Iteration: 501 / 1000 [ 50%]  (Sampling)
## Chain 2 Iteration: 600 / 1000 [ 60%]  (Sampling)
## Chain 2 Iteration: 700 / 1000 [ 70%]  (Sampling)
## Chain 2 Iteration: 800 / 1000 [ 80%]  (Sampling)
## Chain 2 Iteration: 900 / 1000 [ 90%]  (Sampling)
## Chain 2 Iteration: 1000 / 1000 [100%]  (Sampling)
## Chain 3 Iteration: 300 / 1000 [ 30%]  (Warmup)
## Chain 3 Iteration: 400 / 1000 [ 40%]  (Warmup)
## Chain 3 Iteration: 500 / 1000 [ 50%]  (Warmup)
## Chain 3 Iteration: 501 / 1000 [ 50%]  (Sampling)
## Chain 3 Iteration: 600 / 1000 [ 60%]  (Sampling)
## Chain 3 Iteration: 700 / 1000 [ 70%]  (Sampling)
```

```
## Chain 3 Iteration: 800 / 1000 [ 80%]  (Sampling)
## Chain 3 Iteration: 900 / 1000 [ 90%]  (Sampling)
## Chain 3 Iteration: 1000 / 1000 [100%]  (Sampling)
## Chain 4 Iteration: 400 / 1000 [ 40%]  (Warmup)
## Chain 4 Iteration: 500 / 1000 [ 50%]  (Warmup)
## Chain 4 Iteration: 501 / 1000 [ 50%]  (Sampling)
## Chain 4 Iteration: 600 / 1000 [ 60%]  (Sampling)
## Chain 4 Iteration: 700 / 1000 [ 70%]  (Sampling)
## Chain 4 Iteration: 800 / 1000 [ 80%]  (Sampling)
## Chain 4 Iteration: 900 / 1000 [ 90%]  (Sampling)
## Chain 4 Iteration: 1000 / 1000 [100%]  (Sampling)
## Chain 1 finished in 0.2 seconds.
## Chain 2 finished in 0.3 seconds.
## Chain 3 finished in 0.2 seconds.
## Chain 4 finished in 0.2 seconds.
## Chain 5 Iteration:   1 / 1000 [  0%]  (Warmup)
## Chain 5 Iteration: 100 / 1000 [ 10%]  (Warmup)
## Chain 5 Iteration: 200 / 1000 [ 20%]  (Warmup)
## Chain 5 Iteration: 300 / 1000 [ 30%]  (Warmup)
## Chain 5 Iteration: 400 / 1000 [ 40%]  (Warmup)
## Chain 5 Iteration: 500 / 1000 [ 50%]  (Warmup)
## Chain 5 Iteration: 501 / 1000 [ 50%]  (Sampling)
## Chain 5 Iteration: 600 / 1000 [ 60%]  (Sampling)
## Chain 5 Iteration: 700 / 1000 [ 70%]  (Sampling)
## Chain 5 Iteration: 800 / 1000 [ 80%]  (Sampling)
## Chain 5 Iteration: 900 / 1000 [ 90%]  (Sampling)
## Chain 5 Iteration: 1000 / 1000 [100%]  (Sampling)

## Chain 5 Informational Message: The current Metropolis proposal is about to be reject

## Chain 5 Exception: normal_lpdf: Scale parameter is 0, but must be positive! (in '/va

## Chain 5 If this warning occurs sporadically, such as for highly constrained variable

## Chain 5 but if this warning occurs often then your model may be either severely ill-

## Chain 5

## Chain 6 Iteration:   1 / 1000 [  0%]  (Warmup)
## Chain 6 Iteration: 100 / 1000 [ 10%]  (Warmup)
## Chain 6 Iteration: 200 / 1000 [ 20%]  (Warmup)
## Chain 6 Iteration: 300 / 1000 [ 30%]  (Warmup)
## Chain 6 Iteration: 400 / 1000 [ 40%]  (Warmup)
## Chain 6 Iteration: 500 / 1000 [ 50%]  (Warmup)
```

```
## Chain 6 Iteration: 501 / 1000 [ 50%]  (Sampling)
## Chain 6 Iteration: 600 / 1000 [ 60%]  (Sampling)
## Chain 6 Iteration: 700 / 1000 [ 70%]  (Sampling)
## Chain 6 Iteration: 800 / 1000 [ 80%]  (Sampling)
## Chain 7 Iteration:   1 / 1000 [  0%]  (Warmup)
## Chain 7 Iteration: 100 / 1000 [ 10%]  (Warmup)
## Chain 7 Iteration: 200 / 1000 [ 20%]  (Warmup)
## Chain 7 Iteration: 300 / 1000 [ 30%]  (Warmup)
## Chain 7 Iteration: 400 / 1000 [ 40%]  (Warmup)
## Chain 7 Iteration: 500 / 1000 [ 50%]  (Warmup)
## Chain 7 Iteration: 501 / 1000 [ 50%]  (Sampling)

## Chain 7 Informational Message: The current Metropolis proposal is about to be rejected because

## Chain 7 Exception: normal_lpdf: Scale parameter is 0, but must be positive! (in '/var/folders/

## Chain 7 If this warning occurs sporadically, such as for highly constrained variable types lik

## Chain 7 but if this warning occurs often then your model may be either severely ill-conditione

## Chain 7

## Chain 8 Iteration:   1 / 1000 [  0%]  (Warmup)
## Chain 8 Iteration: 100 / 1000 [ 10%]  (Warmup)
## Chain 5 finished in 0.2 seconds.
## Chain 6 Iteration: 900 / 1000 [ 90%]  (Sampling)
## Chain 6 Iteration: 1000 / 1000 [100%]  (Sampling)
## Chain 6 finished in 0.2 seconds.
## Chain 7 Iteration: 600 / 1000 [ 60%]  (Sampling)
## Chain 7 Iteration: 700 / 1000 [ 70%]  (Sampling)
## Chain 7 Iteration: 800 / 1000 [ 80%]  (Sampling)
## Chain 7 Iteration: 900 / 1000 [ 90%]  (Sampling)
## Chain 7 Iteration: 1000 / 1000 [100%]  (Sampling)
## Chain 8 Iteration: 200 / 1000 [ 20%]  (Warmup)
## Chain 8 Iteration: 300 / 1000 [ 30%]  (Warmup)
## Chain 8 Iteration: 400 / 1000 [ 40%]  (Warmup)
## Chain 8 Iteration: 500 / 1000 [ 50%]  (Warmup)
## Chain 8 Iteration: 501 / 1000 [ 50%]  (Sampling)
## Chain 8 Iteration: 600 / 1000 [ 60%]  (Sampling)
## Chain 8 Iteration: 700 / 1000 [ 70%]  (Sampling)
## Chain 8 Iteration: 800 / 1000 [ 80%]  (Sampling)
## Chain 8 Iteration: 900 / 1000 [ 90%]  (Sampling)
## Chain 7 finished in 0.2 seconds.
## Chain 8 Iteration: 1000 / 1000 [100%]  (Sampling)
```

```
## Chain 8 finished in 0.2 seconds.
##
## All 8 chains finished successfully.
## Mean chain execution time: 0.2 seconds.
## Total execution time: 0.8 seconds.

## Warning: 27 of 4000 (1.0%) transitions ended with a divergence.
## See https://mc-stan.org/misc/warnings for details.
```

```
precis(m5.2, depth = 2)
```

```
##                mean        sd       5.5%      94.5%     rhat  ess_bulk
## alpha[1]    70.221230 4.6957078 62.9122250 77.6751330 1.003921 3956.8010
## alpha[2]    66.637192 4.7935979 59.2921655 74.3219425 1.000703 4048.1323
## alpha[3]    89.391211 4.8460529 81.8305935 96.9301420 1.001484 3581.3337
## alpha[4]    79.068129 4.7090695 71.7267400 86.7491720 1.002625 4088.4617
## alpha[5]    61.248545 4.6824417 53.8962900 68.7503010 1.002206 3886.9310
## alpha[6]    71.711300 4.5314894 64.4525160 79.0005630 1.001403 3806.2065
## alpha[7]    72.121798 4.7349544 64.6055805 79.6941110 1.001856 3482.5502
## alpha[8]    69.834714 4.7037178 62.3043635 77.3545565 1.000583 4054.8858
## alpha[9]    73.526363 4.7369985 66.0854635 81.0876520 1.001352 3708.7551
## alpha[10]   83.497318 4.7172575 76.0272720 91.0849000 1.001394 3871.4522
## alpha[11]   73.114282 4.6041905 65.8139360 80.4195110 1.002261 3330.7789
## alpha[12]   44.827995 4.8132889 37.2627160 52.4649335 1.003939 3774.3717
## alpha[13]   89.401919 4.7509654 81.8033940 96.8927025 1.001166 3733.9925
## alpha[14]   77.155674 4.7663782 69.3451470 84.8327385 1.002594 3611.6344
## alpha[15]   78.928227 4.8283770 71.2416295 86.6386025 1.005149 3462.8282
## alpha[16]   37.537016 4.8929538 29.6438890 45.4816100 1.000967 3802.3178
## alpha[17]   87.205983 4.6947001 79.8024680 94.7725850 1.004141 3205.6678
## alpha[18]   67.618180 4.6490608 60.4495525 74.8906485 1.000691 3565.7257
## alpha[19]   65.734938 4.7038246 58.2243970 73.1307710 1.003272 3021.3632
## alpha[20]   82.148125 4.6634709 74.8132215 89.4844660 1.001809 3525.2622
## alpha[21]   32.959556 4.7365812 25.6872720 40.6902550 1.000230 3896.3144
## alpha[22]   65.312545 4.7691775 57.6339360 72.9769385 1.001701 3888.3018
## alpha[23]   69.922455 4.6557324 62.4829305 77.3353785 1.004454 3949.3830
## alpha[24]   84.028041 4.8666565 76.4453060 91.8557125 1.002520 4582.8271
## alpha[25]   57.576782 4.6027933 50.1384940 65.0507070 1.003696 3883.0937
## alpha[26]   69.844403 4.5423390 62.4724325 77.0730070 1.002214 3588.0981
## alpha[27]   78.181829 4.6438020 70.9217780 85.6579100 1.000977 3573.7990
## alpha[28]   83.976099 4.7519956 76.3497625 91.4832750 1.000891 3836.6036
## alpha[29]   48.888864 4.6981990 41.3078680 56.4819905 1.001334 4153.8714
## alpha[30]   63.812328 4.7111640 56.4071725 71.2587825 1.003047 3935.6824
## alpha[31]   26.098770 4.8818137 18.3637725 33.9304330 1.003445 4068.8549
## alpha[32]   76.676408 4.7365322 69.0211035 83.9915570 1.002364 3823.5694
## alpha[33]   73.055353 4.6273537 65.9705020 80.5753430 1.002063 4039.5757
```

```
## alpha[34]   78.963166 4.7676436 71.3900570 86.3736705 1.001748 3403.3517
## alpha[35]   78.075230 4.7744829 70.5443095 85.7926805 1.001281 4222.4407
## alpha[36]   71.652651 4.7601329 64.0808175 79.2662110 1.000708 4471.9540
## alpha[37]   52.084764 4.7084537 44.8942315 59.7080135 1.002348 3439.9837
## alpha[38]   76.121385 4.8822293 68.3599335 83.7502705 1.002290 4006.6888
## alpha[39]   75.451394 4.7905587 67.7880930 82.8390550 1.000887 3454.6782
## alpha[40]   48.408307 4.7026185 40.9218930 56.1687365 1.003060 3999.9664
## alpha[41]   76.176540 4.6262898 68.7745580 83.5991630 1.004537 3963.8804
## alpha[42]   28.812970 4.7775508 21.1928615 36.4243750 1.003746 3855.7962
## alpha[43]   35.646945 4.6876761 28.1427865 43.0427785 1.000564 4114.5370
## alpha[44]   76.736915 4.6819911 69.0549725 83.9337520 1.003016 3594.7244
## alpha[45]   79.377066 4.6099939 71.8918425 86.7122600 1.001080 4016.1217
## alpha[46]   76.266121 4.6936374 68.9066090 83.8324250 1.000442 4007.8677
## alpha[47]   58.467633 4.5906174 51.1480830 65.6998370 1.002588 3396.4658
## alpha[48]   72.114214 4.7995591 64.4240875 79.7356615 1.005283 3715.8719
## alpha[49]   74.771082 4.6137731 67.3251525 82.0455495 1.001963 3457.0385
## alpha[50]   75.414102 4.7692631 67.7589025 83.1030340 1.001202 4211.2151
## beta[1]      1.196482 1.4699703 -0.7591575  3.6725132 1.005792  932.0193
## beta[2]     -1.260339 1.5058841 -3.8538230  0.6600649 1.007244  857.8581
## mu_alpha    67.979401 2.5706977 63.9069175 72.0885400 1.003032 1842.7425
## sigma_alpha 15.457249 1.6142987 13.1380645 18.2104485 1.000383 4277.2078
## sigma_beta   1.685571 1.0527940  0.3547305  3.5587972 1.001191 1581.1771
## sigma_eps    6.713486 0.6434269  5.7619659  7.7933864 1.004713 2175.7641
```

```
precis(m5.2)
```

```
## 52 vector or matrix parameters hidden. Use depth=2 to show them.
```

```
##                  mean        sd       5.5%     94.5%     rhat ess_bulk
## mu_alpha    67.979401 2.5706977 63.9069175 72.088540 1.003032 1842.742
## sigma_alpha 15.457249 1.6142987 13.1380645 18.210449 1.000383 4277.208
## sigma_beta   1.685571 1.0527940  0.3547305  3.558797 1.001191 1581.177
## sigma_eps    6.713486 0.6434269  5.7619659  7.793386 1.004713 2175.764
```

```
# check systematic difference for rater in posterior
post <- extract.samples(m5.2)
mean(post$beta[,1] - post$beta[,2])
```

```
## [1] 2.456821
```

```
# ICC agreement:
post <- extract.samples(m5.2)
(var_patients <- mean(post$sigma_alpha^2))  # Between-patient variance
```

```
## [1] 241.5319
```

```
(var_raters <- mean(post$sigma_beta^2))     # Rater variance
```

```
## [1] 3.949248
```

```
(var_residual <- mean(post$sigma_eps^2))    # Residual variance
```

```
## [1] 45.48479
```

```
# ICC_agreement =
var_patients / (var_patients + var_raters + var_residual)
```

```
## [1] 0.8301037
```

```
# 0.8033613 (sigma_alpha ~ dexp(1))
# 0.83 (sigma_alpha ~ dexp(0.5))

# ICC (Single_fixed_raters) = ICC3 in psych output =
var_patients / (var_patients + var_residual)
```

```
## [1] 0.8415256
```

```
# 0.8415256
```

It should be noted that this ICC is very sensitive to the choice of the prior. If you choose too agressive priors for the standard deviations $\sigma_\alpha, \sigma_\beta, \sigma_\varepsilon$, you will get a too low ICC.

We will proably talk about this in the next lecture (Methodenvertiefung) in greater detail. I have played around a little with the parameters in the exponential priors to get the desired result which compares nicely to the two alternative methods below: using the `psych` package and with the `lmer` package. Both use a Frequentist random intercept model in the background. Using a package like `psych` does just gives a more convenient interface to elicit the ICC.

`psych` package:

```
library(psych)
# needs wide format
conflicts_prefer(dplyr::select)
```

```
## [conflicted] Will prefer dplyr::select over any other package.
```

```
df_wide <- df_long_bias %>%
  pivot_wider(names_from = Rater, values_from = ROM)
df_wide_values <- df_wide %>% select(-ID)
psych::ICC(df_wide_values) # ICC1 = 0.83
```

```
## Call: psych::ICC(x = df_wide_values)
##
## Intraclass correlation coefficients
##                          type  ICC  F df1 df2       p lower bound upper bound
## Single_raters_absolute   ICC1 0.83 11  49  50 1.1e-14        0.72        0.90
## Single_random_raters     ICC2 0.83 12  49  49 1.4e-15        0.71        0.91
## Single_fixed_raters      ICC3 0.85 12  49  49 1.4e-15        0.75        0.91
## Average_raters_absolute ICC1k 0.91 11  49  50 1.1e-14        0.84        0.95
## Average_random_raters   ICC2k 0.91 12  49  49 1.4e-15        0.83        0.95
## Average_fixed_raters    ICC3k 0.92 12  49  49 1.4e-15        0.86        0.95
##
##  Number of subjects = 50     Number of Judges =  2
## See the help file for a discussion of the other 4 McGraw and Wong estimates,
```

lmer package:

```
# _lmer------
m5.3 <- lmer(ROM ~ (1 | ID) + (1 | Rater), data = df_long_bias)
summary(m5.3)
```

```
## Linear mixed model fit by REML ['lmerMod']
## Formula: ROM ~ (1 | ID) + (1 | Rater)
##    Data: df_long_bias
##
## REML criterion at convergence: 793.2
##
## Scaled residuals:
##      Min       1Q   Median       3Q      Max
## -1.87448 -0.46270  0.00272  0.57820  1.45008
##
## Random effects:
##  Groups   Name        Variance Std.Dev.
##  ID       (Intercept) 270.882  16.458
##  Rater    (Intercept)   6.193   2.489
##  Residual              47.557   6.896
## Number of obs: 100, groups:  ID, 50; Rater, 2
##
## Fixed effects:
##             Estimate Std. Error t value
## (Intercept)   68.090      2.998   22.71
```

```
print(VarCorr(m5.3), comp = "Variance")
```

```
##  Groups   Name        Variance
##  ID       (Intercept) 270.882
##  Rater    (Intercept)   6.193
##  Residual              47.557
```

```
# Groups   Name        Variance
# ID       (Intercept) 270.882
# Rater    (Intercept)   6.193
# Residual              47.557


# ICC (Single_random_raters) = ICC2 in psych output
270.882 / (270.882 + 6.193 + 47.557) #
```

```
## [1] 0.8344279
```

```
# 0.8344279
# how can this be higher?

# ICC (Single_fixed_raters) = ICC3 in psych output
270.882 / (270.882 + 47.557) #
```

```
## [1] 0.8506559
```

```
# 0.85
```

5.1.3 Explanation of ICCs in the `psych` output

…

5.1.4 Difference between correlation and ICC

If we do not introduce a bias in the data, the correlation coefficient is the same as the ICC (as seen above). On page 110, Figure 5.3, the authors show nicely what the difference is between the correlation coefficient and the ICC. We note:

- The $ICC_{agreement}$ measures how tightly the two measurements are clustered around the line of equality $(y = x)$…… ……..

Let's verify Figure 5.3 with a little simulation:

….

5.1.5 Standard Error of Measurement (SEM)

...

5.1.6 Bland-Altman Plot

...

5.2 Validity

...

5.3 TODOS

- mention missing values, missingness mechanisms -> Methodenvertiefung
- Logistic Regression, Poisson, -> Methodenvertiefung
- Exercise: Show by simulation what Gelman talks about with significant p values. So I scan the data for significant p values and then simulate data with the same effect size and see how often I get significant p values. Especially the next effect would be probably smaller, especially, if one did p-hacking! Calculate a priori probability for replication (def?).
- Chapter: Sample size calculations for multivariate regression, Proportions, ICCs, t.test
- Chapter about Reliability, Validity and ICCs (incl. simulation of what an ICC of 0.9 or so means), but maybe reduced
- Angenommen man hat ein masking eines Effekts und der Model fit ist aber gut (keine Voraussetzung verletzt), ist diese Situation möglich?
- What about papers? -> eLearning
- AIC, BIC, cross-validation, Model selection (best subset, leaps....), Variable selection
- More on bias variance tradeoff, show for polynomial regression?
- include eLearning tasks in script.