BIO144 Course Book (version 2026)

Owen Petchey

2026-01-06

Table of contents

Preface

Warning

Under construction: This 2026 version of this book is currently being prepared. Some chapters are not yet available, and some content may change before the course starts in February 2026.

This book contains the content of the course BIO144 Data Analysis in Biology at the University of Zurich. It is intended to be used as a companion to the lectures and practical exercises of the course. All of the required content of the course (i.e., what could be in the final exam) is included in this book. Additional content is included for those who want to learn more.

Beware that Owen sometimes makes updates to the book during the semester, so if you have downloaded a copy or taken screenshots, your copy may not exactly match the most current version. However, all of the required content will be the same, and any changes will be correcting typos or improving explanations. If content does change in a way that would change the answer to a question in the final exam, Owen will announce this in the lectures and on OLAT.

How to get a copy of this book

If you'd like a copy of this book for yourself, there are a few ways. But beware: if you take a local copy then it will not be updated when Owen makes changes to the online version!

- You can download a PDF version of the entire book: Download PDF
- You can download a complete local copy of the HTML version of the BIO144 course book from here: https://github.com/opetchey/BIO144_Course_Book/tree/main. The html files for the book are in the `_book` folder, and this is the only folder you need for your offline html copy of the book. You can open the `index.html` file in your web browser to read the book offline.

- You can get all of the source code for the book from the GitHub repository. However, you may find it a little complicated to do anything useful with it!

Datasets are not real

The datasets used in this book are not real datasets. They were created to illustrate the methods taught in the course. Any resemblance to real data is purely coincidental. Please do not use these datasets for any purpose other than learning the methods taught in this course. The patterns in the data may not reflect real-world patterns, and should not be used to draw any conclusions about real-world phenomena.

Getting the datasets

The datasets used in this book are available for download as a zip file here: course_book_datasets.zip. You can download this file and unzip it to get all of the datasets used in the book. The datasets are in CSV format, which can be opened in R or other statistical software.

Packages used in this book

This book uses a number of R add-on packages for data analysis and visualization. You will need to install these packages in order to run the code in the book. The required packages are listed in the `_common.r` file in the GitHub repository for the book. Here is a link to that file: _common.R. You can copy and paste the list of packages from that file into your R console to install them.

If you think you found a mistake in this book

If you think you have found a mistake in the book, please say. A really nice way is to submit an issue on the GitHub repository for the book: Issues page of the GitHub repository. You will need a GitHub account to do this, but they are free and easy to set up. Otherwise tell Owen in person sometime, or in the OLAT Forum, or by email.

When reporting a mistake, please be as specific as possible about where the mistake is. A screenshot works well. Or give the chapter and section number, and copy a chunk of text, as well as a description of the issue problem of course!

How this book was made

The book was written using a type of RMarkdown. It allows a script with a mix of normal text and R code to produce chapters and a book that has a mixture of text, R code, and R output. Rmarkdown is very useful for making reports, books, presentations, and even websites.

This book is a Quarto book. To learn more about Quarto books visit https://quarto.org/docs/books.

Acknowledgements

The content was based on lectures originally written by Dr Stefanie Muff.

The content of the book was written with the assistance of Github Copilot, an AI tool that helps write code and text.

Introduction (L1)

The first lecture of the course introduces it, gives some important information, and sets the stage for the rest of the course. Some of the time in the lecture will be used to create a dataset for use during the course. It also gives an opportunity to review some of the things about R and statistics that it is very useful to already know.

The lecture includes:

- Goals of the course
- Course organisation
- AI and the course
- Making a course dataset
- Using RStudio
- Reviewing what you should already know
- Learning objectives
- A general workflow for data analysis

Notation and some definitions

Throughout the course, we will use the following notation:

- x for a variable. Typically this variable contains a set of observations. These observations are said to represent a sample of all the possible observations that could be made of a *population*.
- $x_1, x_2, ...$ for the values of a variable
- x_i for the ith value of a scalar variable. This is often spoken as "x sub i" or the "i-th value of x".
- $x^{(1)}$ for variable 1, $x^{(2)}$ for variable 2, etc.
- The mean of the sample x is $\bar{x}$. This is usually spoken as "x-bar".
- The mean of x is calculated as $\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i$.
- n is the number of observations in a sample.
- The summation symbol $\sum$ is used to indicate that the values of x are summed over all values of i from 1 to n.

- The standard deviation of the sample is s. The standard deviation of the population is σ.
- The variance is s^2. The variance of the population is σ^2.
- The variance of the sample is calculated as $s^2 = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})^2$.
- The standard deviation of the sample is calculated as $s = \sqrt{s^2}$.
- y is usually used to represent a dependent / response variable.
- x is usually used to represent an independent / predictor / explanatory variable.
- β_0 is usually used to denote the intercept of a linear model.
- β_1, β_2, etc. are usually used to denote the coefficients of the independent variables in a linear model.
- Estimates are denoted with a hat, so $\hat{\beta}_0$ is the estimate of the intercept of a linear model.
- Hence, the estimated value of y_i in a linear regression model is $\hat{y}_i = \hat{\beta}_0 + \hat{\beta}_1 x_i^{(1)}$.
- e_i is the residual for the ith observation in a linear model. The residual is the difference between the observed value of y_i and the predicted value of y_i ($\hat{y}_i$).
- Often we assume errors are normally distributed with mean 0 and variance σ^2. This is written as $e_i \sim N(0, \sigma^2)$.
- SST is the total sum of squares. It is the sum of the squared differences between the observed values of y and the mean of y. It is calculated as $\sum_{i=1}^{n}(y_i - \bar{y})^2$.
- SSM is the model sum of squares. It is the sum of the squared differences between the predicted values of y and the mean of y. It is calculated as $\sum_{i=1}^{n}(\hat{y}_i - \bar{y})^2$.
- SSE is the error sum of squares. It is the sum of the squared differences between the observed values of y and the predicted values of y. It is calculated as $\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$.
- The variance of x can be written as $Var(x)$. The covariance between x and y can be written as $Cov(x, y)$.
- Covariance is calculated as $Cov(x, y) = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})$.
- H_0 is the null hypothesis.
- α is the significance level.
- df is the degrees of freedom.
- p is the p-value.

Data analysis workflow

A general workflow for data analysis is as follows:

1. **Define the question**: What are you trying to find out?
2. **Define the study**: How will you answer the question? What subjects, what observations, what measurements? What experimental design? What treatments? What graphics and analyses will you use?

3. **Collect the data**: Gather the necessary data to answer the question.
4. **Explore the data**: Use summary statistics and graphics to understand the data.
5. **Prepare the data**: Clean and format the data for analysis.
6. **Visualise the data**: Create plots to visualise patterns and relationships.
7. **Analyse the data**: Use appropriate statistical methods to analyse the data, including checking model assumptions.
8. **Interpret the results**: Draw conclusions from the analysis in the context of the original question.
9. **Be critical**: Consider limitations, alternative explanations, and the robustness of your conclusions.
10. **Communicate the results**: Present the findings in a clear and concise manner, using tables, figures, and written summaries.

Using Generative AI in R and Data Analysis: Guidance and Good Practice

! Important

For the final examination, you will use your own computer, but the test will run inside the Safe Exam Browser, which will be configured to block all access to generative AI tools, browser-based assistants, external software, online services, and any AI code copilots inside RStudio or other IDEs. This means **no form of generative AI will be available during the exam**. Because of this, please avoid becoming overly reliant on GenAI—such as ChatGPT, Claude, Gemini, Copilot, or similar tools for answering quiz questions, explaining results, fixing errors, guiding your analysis, or writing code. You must be able to perform all tasks independently. We also strongly recommend that you do not use RStudio with Copilot integration during the course, as it will not function in the exam environment and may leave you under prepared. Throughout the semester, be sure to practice writing your own R code, interpreting outputs yourself, and applying statistical reasoning without AI assistance, as your exam performance will depend entirely on your own knowledge and skills.

Generative AI (GenAI) tools can support learning, exploration, and coding in R. They can be powerful assistants, but they must be used with care. This section introduces the types of tools available, provides guidelines for responsible use, highlights red flags for problematic usage, and gives examples of good and poor practice.

Typical uses:

- asking conceptual questions

- summarizing methods
- generating example code
- explaining error messages

Strengths:

- flexible and conversational
- good for brainstorming
- can generate starter code

Limitations:

- often wrong in subtle ways
- may hallucinate functions
- cannot see your working R session

Guidelines for Good Use of Generative AI

Use GenAI as a Helper, Not a Source of Truth

Best uses:

- drafting
- explanation
- syntax reminders
- scaffolding

Not reliable for:

- model choice
- statistical inference
- interpreting coefficients
- designing analysis workflows
- checking assumptions

Always Verify AI-Generated Code and Explanations

Check:

- does the code run?
- do variable names match?
- is the model appropriate?
- are assumptions addressed?
- is the explanation logically correct?

Keep Human Judgement Central

GenAI cannot:

- understand scientific questions
- evaluate model assumptions
- know ecological/biological reasoning
- determine appropriate models

Provide Context Carefully

When asking GenAI:

- describe variables
- provide example data
- specify your goal
- show your existing code

Better context = better answers.

Use GenAI to Improve Understanding, Not Bypass It

Helpful:

- *"Explain logistic regression."*
- *"Why do residuals fan out?"*

Not helpful:

- *“Do my assignment for me.”*

Indicators of Problematic Usage

Code That Does Not Reflect Ability

Signs:

- unfamiliar advanced syntax
- unexplained packages
- inconsistent style

Hallucinated Functions or Nonsensical Code

Examples:

- `slope(x)` in mixed models
- missing arguments
- fabricated packages

Statistical Errors Typical of AI

Common issues:

- wrong model family
- wrong inference logic
- invented assumptions
- incorrect explanation of coefficients

Lack of Understanding

Indicators:

- cannot explain model
- inconsistent interpretations
- identical phrasing to AI output

Over-Reliance on AI

Signs:

- using AI for every step
- no debugging effort
- stagnation in skill development

Examples of Good and Problematic Use

Good Use Examples

A. Syntax help
“How do I specify a random slope in `lme4`?”

B. Clarification
“How does adding an interaction change interpretation?”

C. Debugging
“What does ‘object not found’ usually mean?”

D. Brainstorming
“How can I visualise a logistic regression?”

Problematic Use Examples

A. Blindly copying model code

```
lm(y ~ x1 + x2 * x3 * x1)
```

B. Incorrect statistical logic
AI code labelled as a bootstrap but is actually a permutation test.

C. Misleading interpretation
Claims that coefficients assume explanatory variable independence.

D. Presenting AI-generated plots without understanding

E. Outsourcing entire workflow
“Write a script that loads data, cleans it, runs models, interprets, and writes the report.”

Summary

Generative AI can:

- help learning
- support debugging
- provide code scaffolds
- explain concepts

But it can also:

- hallucinate
- produce incorrect models
- misinterpret statistics

Use GenAI as a supportive tool—never as an unquestioned authority. Good use of GenAI *supports* learning. Problematic use *replaces* it.

Common GenAI Errors in R and Statistical Modelling

Generative AI tools can be helpful for writing R code, exploring ideas, and learning syntax.
However, they sometimes produce *plausible but incorrect* code or explanations. This section provides real examples of typical GenAI mistakes, with correct solutions and learning points.

Why this matters: GenAI is a **pattern-matching system**, not a statistical reasoning engine. It does not understand assumptions, inference, or modelling logic.Therefore, students should **never accept code or explanations without checking them**.

Incorrect formula structure in `lm()`

Prompt: *Fit a linear model with main effects and a two-way interaction between `x2` and `x3`.*

Incorrect GenAI output:

```
lm(y ~ x1 + x2 * x3 * x1, data = df)
```

This includes an unintended **three-way interaction** and extra terms.

Correct:

```
lm(y ~ x1 + x2 * x3, data = df)
```

Learning point: Always check model formulas carefully. AI often adds or removes interactions.

Confusing bootstrap and permutation tests

Documented case: GenAI was asked for a *bootstrap t-test*.

Incorrect GenAI code (actually a permutation test):

```r
t_stats <- replicate(1000, {
  perm <- sample(df$group)
  t.test(df$value ~ perm)$statistic
})
```

Correct bootstrap approach:

```r
t_stats <- replicate(1000, {
  sample_df <- df[sample(nrow(df), replace = TRUE), ]
  t.test(value ~ group, data = sample_df)$statistic
})
```

Learning point: The logic of inference matters. Code that runs is not necessarily correct.

Incorrect explanation of linear-model coefficients

Incorrect claim: *"Coefficients assume independence among explanatory variable."*

This is false. Linear model coefficients describe **conditional effects within the model**, regardless of collinearity.

Learning point: Interpretations come from the model structure, not from simplistic assumptions GenAI sometimes invents.

Hallucinated functions in mixed models

Incorrect GenAI output:

```r
lmer(y ~ x + (slope(x) | group), data = df)
```

`slope()` does not exist.

Correct random-slope specification:

```r
lmer(y ~ x + (x | group), data = df)
```

Learning point: Always verify syntax in package documentation.

Wrong variable names

The dataset has variables `height` and `age`.

Incorrect GenAI output:

```
lm(Height ~ Age, data = df)
```

Correct:

```
lm(height ~ age, data = df)
```

Learning point: GenAI often guesses variable names. Check against your data.

Wrong model family for binary data

Incorrect GenAI output (linear regression):

```
lm(y ~ x, data = df)
```

Correct logistic regression:

```
glm(y ~ x, data = df, family = binomial)
```

Learning point: For binary response variables, specify the model family explicitly.

Incorrect explanation of random intercepts

Incorrect claim:
"Random intercepts eliminate correlation among repeated measures."

Incorrect — they **model** correlation, not eliminate it.

Learning point: Random effects structure determines the implied correlation. AI explanations are often vague or wrong here.

Omitting interaction terms in ANOVA

Prompt: *Two-way ANOVA with interaction.*

Incorrect:

```
aov(y ~ factor1 + factor2, data = df)
```

Correct:

```
aov(y ~ factor1 * factor2, data = df)
```

Learning point: Confirm that the model matches the experimental design.

Incorrect use of `predict()`

Prompt: *Predict for new x values.*

Incorrect GenAI output:

```r
predict(model)
```

This gives **in-sample fitted values**, not predictions for new data.

Correct:

```r
predict(model, newdata = data.frame(x = c(1, 2, 3)))
```

Learning point: Always specify `newdata` for predictions.

Poor explanations of multicollinearity

Incorrect GenAI claim:
"Multicollinearity is indicated when the model p-value is low but the individual explanatory variable p-values are high."

This is an unreliable and incomplete diagnostic.

Better diagnostics:

```r
car::vif(model)
cor(df)
model.matrix(model)
```

Learning point: AI often repeats common internet tropes rather than robust statistical principles.

GenAI Summary

GenAI can:

- write useful scaffolding code,
- provide quick reminders,
- help with simple tasks.

But it can also:

- hallucinate functions,
- give subtly incorrect models,

- invent statistical logic,
- provide plausible but wrong explanations.

Advice for students:
Use GenAI as a starting point, not an authority.
Always check:

- function names,
- model formulas,
- assumptions,
- interpretations,
- and logic.

In statistics, clarity of reasoning matters more than code that merely *runs*.

R and RStudio (L2)

Getting R and RStudio

R is a programming language and software environment for statistical computing and graphics. RStudio is an integrated development environment (IDE) for R. ***RStudio*** provides a user-friendly interface for working with R, including a console, a script editor, and tools for managing packages and projects.

We highly recommend using **RStudio** to work with **R**.

There are two ways to use **RStudio**:

1. **RStudio Desktop**: a standalone application that you can install on your computer. If you choose this option, you will need to install R first, and then install RStudio. This usually requires administrator privileges on your computer. If you have problems installing add-on packages, they will have to be fixed by you or with our help (rarely we cannot find a solution). Follow the instructions on this website about how to install R and RStudio: https://posit.co/download/rstudio-desktop/.

2. **Rstudio Cloud**: a web-based version of RStudio that you can use in your web browser. You don't need to install anything on your computer, and you can access your work from any computer with an internet connection. The Faculty of Science has a RStudio Cloud here that you can use (and will have to use during the final exam).

What do we recommend? Try the cloud first. If you like it then continue to use it.

> **!** Important
>
> Whether you use the RStudio application on your computer, or use RStudio on the Cloud, you are responsible for the safety and persistence of your files (data, code, etc.). Just because you're using RStudio on the Cloud does not mean your files are automatically saved forever. Make sure to download and back up your important files regularly!

Think–Pair–Share (#a_r_vs_rstudio_roles) What is one thing that R does, and one thing that RStudio does? Why is it useful that these are separate?

Getting to know the RStudio IDE

When you open RStudio, you will see a window with four main panes:

1. **Source pane**: where you can write and edit R scripts, R Markdown documents, and other files. This pane can have multiple tabs, so you can have several files open at the same time.
2. **Console pane**: where you can type and execute R commands directly. This pane has multiple tabs, including: **Console**, **Terminal**, and **Jobs**. During this course we will mostly use the **Console** tab.
3. **Environment pane**: where you can see the objects (data frames, vectors, etc.) that are currently in your R session. This pane has multiple tabs, including: **Environment**, **History**, **Connections**, and **Tutorial**. During this course we will mostly use the **Environment** tab.
4. **Files/Plots/Packages/Help pane**: where you can manage files, view plots, manage packages, and access help documentation. This pane has multiple tabs, including: **Files**, **Plots**, **Packages**, **Help**, and **Viewer**. During this course we will mostly use the **Files**, **Plots**, **Packages**, and **Help** tabs.

Our **Scripts** are in the Source pane tabs. The code / script we write in R is usually saved in a file with the extension `.R`. This file can be opened and edited in the Source pane. Creating a new R script: File > New File > R Script.

You can run code from the script by selecting the code and clicking the "Run" button, or by using the keyboard shortcut `Ctrl + Enter` (Windows) or `Cmd + Enter` (Mac).

There is so much more to learn about the RStudio IDE, but we will cover that as we go along in the course.

Getting to know R

In our newly opened script file, type the following code:

```r
# This is a comment. Comments are ignored by R.
# They are useful for explaining what your code does.
```

Then type the following code:

```r
1 + 1
```

```
[1] 2
```

```r
exp(2)
```

```
[1] 7.389056
```

```
sqrt(16)
```

```
[1] 4
```

Select all the code and run it (using the “Run” button or `Ctrl + Enter` / `Cmd + Enter`). You should see the results of the calculations in the Console pane.

Now try assigning values to named object:

```
a <- 5
b <- 10
c <- a + b
```

Note

In fact, just about everything in R is an object! These objects live in your R session (i.e., in the memory of your computer), and you can see them in the Environment pane. You can create objects to store data, functions, and other information. Your data files are something different. They are just like other files on your computer, such as documents, images, and music files. They live on your hard drive (or in the cloud), and you can import them into R when you need to work with them.

And then print the value of `c`:

```
c
```

```
[1] 15
```

You should see the value `15` printed in the Console pane.

We can also create vectors:

```
my_vector <- c(1, 2, 3, 4, 5)
my_vector
```

```
[1] 1 2 3 4 5
```

And do maths on vectors:

```
my_vector * 2
```

```
[1]  2  4  6  8 10
```

```
my_vector + 10
```

```
[1] 11 12 13 14 15
```

```
my_vector ^ 2
```

```
[1]  1  4  9 16 25
```

We can vectors of strings (text):

```r
my_strings <- c("apple", "banana", "cherry")
my_strings
```

```
[1] "apple"  "banana" "cherry"
```

And can perform operations on strings:

```r
paste("I like", my_strings)
```

```
[1] "I like apple"  "I like banana" "I like cherry"
```

```r
toupper(my_strings)
```

```
[1] "APPLE"  "BANANA" "CHERRY"
```

And we can create data frames, which are like tables of data

```r
my_data <- data.frame(
  Name = c("Alice", "Bob", "Charlie"),
  Age = c(25, 30, 35),
  Height = c(165, 180, 175)
)
my_data
```

```
     Name Age Height
1   Alice  25    165
2     Bob  30    180
3 Charlie  35    175
```

Above we have numerous examples of functions: `exp()`, `sqrt()`, `c()`, `paste()`, `toupper()`, and `data.frame()`. Functions are a fundamental part of R programming. They are used to perform specific tasks, such as calculations, data manipulation, and data analysis. All functions have a name and can take arguments (inputs) and return values (outputs). They are called by writing the function name followed by parentheses, with any arguments inside the parentheses.

You likely guessed that there is much much more to learn about R, but we will cover that as we go along in the course.

Think–Pair–Share (#tps_a_objects_not_files) When you type `a <- ...`, where does `x` live?

Getting help

R has a built-in help system that you can use to get information about functions, packages, and other topics. To access the help system, you can use the `?`

operator followed by the name of the function or topic you want to learn about. For example, to get help on the `mean()` function, you would type:

```
?mean
```

This will open the help documentation for the `mean()` function in the Help pane of RStudio. The documentation includes a description of the function, its arguments, and examples of how to use it. Some of the help documentation is very useful and accessible, other is less so. Over time you will learn which functions and packages have good documentation, and you will get better and better at understanding R help files.

Of course you can use any other resources to get help with R, including online forums, tutorials, and books. Some popular online resources for R help include:

- Stack Overflow
- RStudio Community
- R-bloggers
- The R Graph Gallery

You can also use search engines like Google to find answers to your R questions. Just be sure to include "R" in your search query to get relevant results.

AI assistants like ChatGPT can also be useful for getting help with R programming. You can ask specific questions about R code, functions, and packages, and get instant responses.

And of course there is always your course instructors and fellow students to help you out when you get stuck.

Think–Pair–Share (#a_errors_are_information) When we do something that R does not understand, it often gives us an error message. In red! What will you do when you get an error message?

Add-on packages

R has a vast ecosystem of add-on packages that extend its functionality. These packages are collections of functions, data, and documentation that can be installed and loaded into your R session. There are thousands of packages available on CRAN (the Comprehensive R Archive Network) and other repositories like Bioconductor and GitHub.

We will be using several packages throughout this course. To install a package, you can use the `install.packages()` function. For example, to install the `ggplot2` package, you would type:

```
install.packages("ggplot2")
```

You can also use the RStudio interface to install packages. Go to the "Packages" tab in the bottom right pane, click on "Install", type the name of the package you want to install, and click “Install”.

You can see which packages are currently installed by looking in the “Packages” tab.

Tip

You only need to install a package once. After it is installed, you can load it into your R session using the `library()` function. Do not install packages every time you want to use them; just load them with `library()`.

R Version and add-on package versions

(This section concerns the Desktop version of R and RStudio, and not so much the Cloud version, because version management is handled for you in the Cloud.)

R and its add-on packages are constantly being updated and improved. This can cause problems when trying to install or use packages that depend on specific versions of R or other packages.

Imagine that the online version of a package has been updated and now only works with the lastest version of R. If you are using an older version of R, you may not be able to install or use that package.

Or if a package depends on another package that has been updated, you may need to update that package as well to use the first package.

This sounds complicated, but there are some simple steps you can take to reduce the chances of running into version-related problems:

1. **Keep your R version up to date.** New versions of R are released every 6 months or so, and they often include important bug fixes and new features. You can check your current R version by typing `R.version.string` in the Console. To update R, you can download the latest version from the CRAN website.

2. **Keep your add-on packages up to date.** You can update all your installed packages by using the `update.packages()` function. This will check for updates for all installed packages and install the latest versions. You can also use the RStudio interface to update packages by going to the “Packages” tab, clicking on “Update”, selecting the packages you want to update, and clicking “Install Updates”.

3. **Do this well before critical deadlines or important events (e.g., exams).** Updating R and packages can sometimes lead to unexpected issues, so it’s best to do it well in advance of when you need everything to work perfectly.

Nevertheless, even with these precautions, you may still encounter version-related issues from time to time. When this happens, don't panic!

A common problem you might see is an error message when trying to install or load a package, indicating that the package requires a newer version of R or another package. The error / warning message might look like:

```
warning: package 'xyz' requires R version >= 4.2.0
```

```
Warning in install.packages: package 'XYZ' is not available (for
R version 4.2.0)
```

These messages indicate that the package you are trying to install or load requires a newer version of R than the one you currently have. To fix this, you will need to update your R installation to the required version or higher. Then its also a good idea to update your packages as well.

Note

RStudio is also regularly updated, with new version released every several months or so. Your version of RStudio is independent of your version of R, so you can update RStudio without changing your R version. Note that usually your version of RStudio is not as important as your version of R and the packages you are using. So updating RStudio is usually not a high priority and doesn't often help solve problems related to add on package versions.

R Projects

I always work within R Projects. R Projects help you to organise your work and keep all files related to a project in one place. They also make importing data a breeze.

But what is an R Project? An R Project is a directory (folder) that contains all the files related to a specific project. When you open an R Project, RStudio automatically sets the working directory to the project directory, so you don't have to worry about setting the working directory manually.

To see if you're working within an R Project, look at the top right of the RStudio window. If you see the name of your project there, you're good to go. If you see "Project: (None)", then you're not working within an R Project.

If you click on the project name, a dropdown menu will appear. From there, you can create a new project, open an existing project, or switch between projects.

Create a new R Project: File > New Project > New Directory or Existing Directory > New Project > Choose a name and location for your project > Create Project.

! Important

Get organised! Put all files for a project in one folder. For example, I made a folder called `BIO144_2026` and put all files related to this course in that folder. Within that folder, I have subfolders for `data`, `scripts`, and `results`. I then create an R Project in the `BIO144_2026` folder. This way, all files related to the course are in one place, and I can easily find them later.

Now, always open and ensure you're working within the R Project for your project. As mentioned, you can see the project name at the top right of the RStudio window. And if its not the correct project, click on the name to get the drop-down list of available projects from which you can switch to the correct one.

Importing data

First, get some data sets for us to work with. **XYZ** You can download them from the course website or use your own data sets. Save the data files in a folder called `data` within your R Project directory.

We will use the `readr` package to import data into R. The `readr` package provides functions to read data from various file formats, including CSV (comma-separated values) files, tab-separated values files, and others.

To read a CSV file, we can use the `read_csv()` function from the `readr` package. For example, to read a CSV file called `my_data_file.csv`, we can use the following code:

```r
library(readr)
my_data <- read_csv("datasets/my_data_file.csv")
```

This code will read the `data.csv` file from the `data` folder within the current working directory (which should be the R Project directory) and store it in a data frame called `data`.

Tip

Easily getting the file path In RStudio, you can easily get the file path by putting the cursor in the parentheses of the `read_csv()` function, the press the tab key. A drop-down menu will appear with options to navigate to the file. This way, you don't have to type the file path manually.

Viewing the data

Once you've imported your data, you can view it in several ways:

- Click on the data frame in the Environment tab in RStudio to open it in a new tab.
- Use the `View()` function to open the data frame in a new tab in RStudio.
- Use the `head()` function to view the first few rows of the data frame.
- Use the `str()` function to view the structure of the data frame, including the variable names and types.
- Use the `summary()` function to get a summary of the data frame, including basic statistics for each variable.

Another useful function is `glimpse()` from the `dplyr` package, which provides a quick overview of the data frame.

```
library(dplyr)
glimpse(my_data)
```

There are many checks you can do to ensure your data was imported correctly. For example checking if there are duplicated values in a variable when there shouldn't be:

```
any(duplicated(my_data$Name))
```

```
[1] FALSE
```

The function `any()` will return `TRUE` if there are any duplicated values in the `Name` variable, and `FALSE` otherwise. The function `duplicated()` returns a logical vector indicating which values are duplicates. We use the dollar sign `$` to access a specific variable (column) in the data frame. A logical vector is a vector that contains only `TRUE` or `FALSE` values:

```
duplicated(my_data$Name)
```

```
[1] FALSE FALSE FALSE
```

All three logicals are `FALSE`, meaning none of the three are duplicates. If there were duplicates, the corresponding positions in the logical vector would be `TRUE`. For example:

```
example_vector <- c("A", "B", "A", "C", "B")
```

What do you expect the output of `duplicated(example_vector)` to be?

A final check (though not the final one we could do–there are many others). Let us check for missing values and get a count of how many there are in each variable. We can do this with the following *tidyverse* code:

```
library(dplyr)
my_data |>
  summarise(across(everything(), ~ sum(is.na(.))))
  Name Age Height
1    0   0      0
```

Looks complicated eh! Well, that's because it is, for sure. But let's break it down:

- `summarise()` creates a new data frame with summary statistics.
- `across(everything(), ~ sum(is.na(.)))` applies the function `sum(is.na(.))` to every variable in the data frame.
- The `is.na()` function returns a logical vector indicating which values are missing (`NA`), and the `sum()` function counts the number of `TRUE` values in that vector (i.e., the number of missing values).

! Important

Let's assume your data was imported incorrectly. This means you have to inspect it carefully. Check that the variable names are correct, that the data types are correct (e.g., numeric, character, factor), that there are the correct number of rows and columns. If you find any issues, you need to find out what caused them, fix them, and re-import the data (see below).

Common data import problems:

- Incorrect delimiter: If your data file uses a different delimiter (e.g., tab, semicolon), you need to specify it in the `read_csv()` function using the `delim` argument (e.g., `read_delim("data.csv", delim = "\t")` for tab-delimited files).
- Missing values: If your data file uses a specific value to represent missing data (e.g., "NA", "-999"), you need to specify it in the `read_csv()` function using the `na` argument (e.g., `read_csv("data.csv", na = c("NA", "-999"))`).
- Only one column: If your data file has only one column, it may be because the delimiter is incorrect. Check the delimiter and re-import the data with the correct delimiter.
- You opened the downloaded file in Excel and then saved it: Excel may have changed the format of the file when you opened and saved it. Always work with the original downloaded file.
- Wrong path or file name: Make sure the file path and name are correct. Remember, when you work in an R Project, you can place the cursor in the parentheses of the `read_csv()` function and press the tab key to navigate to the file.

Data wrangling

Now we have our data imported and checked, and we're ready to start working with it. This process is called data wrangling, and it involves cleaning, transforming, and reshaping the data to make it suitable for visualisation and analysis.

Clean the variable names

The first thing I like to do is standardise and clean up the variable names. I like to use the `janitor` package for this:

```r
library(janitor)
```

```

Attaching package: 'janitor'

The following objects are masked from 'package:stats':

    chisq.test, fisher.test
```

```r
my_data <- my_data |>
  clean_names()
```

The `clean_names()` function from the `janitor` package will convert variable names to a consistent format (lowercase, spaces replaced by underscores, no special characters).

> **Note**
>
> When we ran the code `library(janitor)` we got the message: **Attaching package: 'janitor' The following objects are masked from 'package:stats': chisq.test, fisher.test** This sometimes happens when two packages have functions with the same name. In this case, the `janitor` package has functions called `chisq.test()` and `fisher.test()`, which are also in the base `stats` package. When we load the `janitor` package, it masks (hides) the functions from the `stats` package. This is usually not a problem, but if you want to use the functions from the `stats` package, you can specify the package name when calling the function, like this: `stats::chisq.test()`.

Manipulate the data frame

Functions in the `dplyr` package are used to manipulate data frames:

- `select()`: select columns by position, or by name, or by other methods
- `filter()`: select rows that meet a logical condition
- `slice()`: select rows by position

- `arrange()`: reorder rows
- `mutate()`: add new variables

The `dplyr` package also provides functions to group data frames and to summarize data:

- `group_by()`: add to a data frame a grouping structure
- `summarize()`: summarize data, respecting any grouping structure specified by `group_by()`

The pipe operator `|>` is used to chain together multiple operations on a data frame.

Tip

Note that you will often see another pipe operator `%>%` used in examples. The pipe operator `|>` is a newer version of `%>%` that is more efficient and easier to use. The pipe operator `|>` is available in R version 4.1.0 and later.

Lets work through some examples with a sample data frame:

```r
my_data1 <- tibble(
  name = c("Alice", "Bob", "Charlie", "David", "Eva"),
  age = c(25, 30, 35, 40, 45),
  score = c(90, 85, 95, 80, 70))
```

Here is the same dataset with 100 rows:

```r
set.seed(123)
my_data2 <- tibble(name = paste0("Person_", sprintf("%03d", 1:100)),
  age = sample(20:50, 100, replace = TRUE),
  score = rnorm(100, mean = 75, sd = 10))
```

Select columns

We can select columns by name

```r
my_data2 |>
  select(name, score)
```

```
# A tibble: 100 x 2
   name       score
   <chr>      <dbl>
 1 Person_001  91.9
 2 Person_002  87.3
 3 Person_003  77.8
 4 Person_004  64.5
 5 Person_005  69.8
 6 Person_006  91.2
 7 Person_007  64.3
 8 Person_008  91.9
 9 Person_009  72.6
10 Person_010  70.3
# i 90 more rows
```

We can select columns by position

```
my_data2 |>
  select(1, 3)
```

```
# A tibble: 100 x 2
   name       score
   <chr>      <dbl>
 1 Person_001  91.9
 2 Person_002  87.3
 3 Person_003  77.8
 4 Person_004  64.5
 5 Person_005  69.8
 6 Person_006  91.2
 7 Person_007  64.3
 8 Person_008  91.9
 9 Person_009  72.6
10 Person_010  70.3
# i 90 more rows
```

We can select columns by a condition, for example select only the numeric columns:

```
my_data2 |>
  select(where(is.numeric))
```

```
# A tibble: 100 x 2
     age score
   <int> <dbl>
 1    50  91.9
 2    34  87.3
 3    38  77.8
 4    33  64.5
 5    22  69.8
 6    29  91.2
 7    37  64.3
 8    41  91.9
 9    30  72.6
10    24  70.3
# i 90 more rows
```

We can select a column by pattern matching, using helper functions, for example select columns that contain the letter "a":

```
my_data2 |>
  select(contains("a"))
```

```
# A tibble: 100 x 2
   name         age
   <chr>      <int>
 1 Person_001    50
 2 Person_002    34
 3 Person_003    38
 4 Person_004    33
 5 Person_005    22
 6 Person_006    29
 7 Person_007    37
 8 Person_008    41
 9 Person_009    30
10 Person_010    24
# i 90 more rows
```

Other helpers include `starts_with()`, `ends_with()`, `matches()`, and `everything()`.

Filter: Getting particular rows of data [#filter-rows]

To get particular rows of data, we can use the `filter()` function. This function takes a *logical condition* as an argument and returns only the rows that meet that condition. For example, to get all rows where the Age is greater than 30:

```
my_data2 |>
  filter(age > 30)
```

```
# A tibble: 66 x 3
   name         age score
   <chr>      <int> <dbl>
 1 Person_001    50  91.9
 2 Person_002    34  87.3
 3 Person_003    38  77.8
 4 Person_004    33  64.5
 5 Person_007    37  64.3
 6 Person_008    41  91.9
 7 Person_011    39  67.3
 8 Person_012    33  96.5
 9 Person_013    41  61.7
10 Person_014    44  80.0
# i 56 more rows
```

Here, the logical condition is `age > 30`.

We can combine multiple conditions using the logical operators `&` (and), `|` (or), and `!` (not). For example, to get all rows where the Age is greater than 30 and the Score is less than 90:

```
my_data2 |>
  filter(age > 30 & score < 90)
```

```
# A tibble: 60 x 3
   name         age score
   <chr>      <int> <dbl>
 1 Person_002    34  87.3
 2 Person_003    38  77.8
 3 Person_004    33  64.5
 4 Person_007    37  64.3
 5 Person_011    39  67.3
 6 Person_013    41  61.7
 7 Person_014    44  80.0
 8 Person_015    45  87.3
 9 Person_016    46  81.3
10 Person_018    38  82.9
# i 50 more rows
```

Other logical operators include `==` (equal to), `!=` (not equal to), `<=` (less than or equal to), and `>=` (greater than or equal to).

Slice: Getting rows by position [#slice-rows]

The `slice()` function allows us to get rows by their position in the data frame. For example, to get the first two rows:

```
my_data2 |>
  slice(1:2)
```

```
# A tibble: 2 x 3
  name         age score
  <chr>      <int> <dbl>
1 Person_001    50  91.9
2 Person_002    34  87.3
```

I very rarely use this function, as I prefer to use `filter()` with logical conditions. I can't think of a good use case for this function right now! Perhaps you can?

Arrange: Reordering rows

The `arrange()` function allows us to reorder the rows of a data frame based on the values in one or more columns. For example, to reorder the rows by Age in ascending order:

```r
my_data2 |>
  arrange(age)
```

```
# A tibble: 100 x 3
   name          age score
   <chr>       <int> <dbl>
 1 Person_064     20  88.8
 2 Person_056     21  79.5
 3 Person_091     21  67.4
 4 Person_005     22  69.8
 5 Person_025     22  74.9
 6 Person_033     23  67.3
 7 Person_092     23  72.4
 8 Person_010     24  70.3
 9 Person_017     24  79.1
10 Person_058     24  72.7
# i 90 more rows
```

I f we want to reorder the rows by Age in descending order, we can use the `desc()` function:

```r
my_data2 |>
  arrange(desc(age))
```

```
# A tibble: 100 x 3
   name          age score
   <chr>       <int> <dbl>
 1 Person_001     50  91.9
 2 Person_061     50  79.9
 3 Person_075     50  67.3
 4 Person_085     50  50.1
 5 Person_096     50  86.8
 6 Person_031     49  71.8
 7 Person_078     49  72.6
 8 Person_024     48  81.2
 9 Person_057     48  80.3
10 Person_021     47  66.0
# i 90 more rows
```

It's unusual to need the rows of a dataset to be arranged in a specific order, but it can be useful when looking at the data directly.

Tip

Note that when you view the data in RStudio, it will always be arranged by the row number. In the viewer you can sort by clicking on the column headers.

Mutate: Adding new variables [#mutate-variables]

The `mutate()` function allows us to add new variables to a data frame. For example, to add a new variable called `Age_in_5_years` that is the Age plus 5:

```
my_data2 |>
  mutate(age_in_5_years = age + 5)
```

```
# A tibble: 100 x 4
   name         age score age_in_5_years
   <chr>      <int> <dbl>          <dbl>
 1 Person_001    50  91.9             55
 2 Person_002    34  87.3             39
 3 Person_003    38  77.8             43
 4 Person_004    33  64.5             38
 5 Person_005    22  69.8             27
 6 Person_006    29  91.2             34
 7 Person_007    37  64.3             42
 8 Person_008    41  91.9             46
 9 Person_009    30  72.6             35
10 Person_010    24  70.3             29
# i 90 more rows
```

We can add multiple new variables at once:

```
my_data2 |>
  mutate(
    age_in_5_years = age + 5,
    percentage_score = score / 100
  )
```

```
# A tibble: 100 x 5
   name         age score age_in_5_years percentage_score
   <chr>      <int> <dbl>          <dbl>            <dbl>
 1 Person_001    50  91.9             55            0.919
 2 Person_002    34  87.3             39            0.873
 3 Person_003    38  77.8             43            0.778
 4 Person_004    33  64.5             38            0.645
 5 Person_005    22  69.8             27            0.698
 6 Person_006    29  91.2             34            0.912
 7 Person_007    37  64.3             42            0.643
 8 Person_008    41  91.9             46            0.919
 9 Person_009    30  72.6             35            0.726
10 Person_010    24  70.3             29            0.703
# i 90 more rows
```

Working with categorical variables

Variables in a data frame in R have a *type*. The most common types of variables are numeric and categorical. Numeric variables are variables that take on numerical values, such as age or score. Categorical variables are variables that take on a limited number of values, often representing categories or groups. In R, categorical variables are typically have *type* `<chr>` which is `character`. Or they can be of type `<fct>` which is `factor`.

When we import data categorical variable is usually imported as a `character` variable. For example, the variable `name` in our example dataset is a categorical variable of type `character`. Look at the first few rows of the dataset again, and see that below the variable name it says `<chr>` for the `name` variable:

```
my_data2
```

```
# A tibble: 100 x 3
   name         age score
   <chr>      <int> <dbl>
 1 Person_001    50  91.9
 2 Person_002    34  87.3
 3 Person_003    38  77.8
 4 Person_004    33  64.5
 5 Person_005    22  69.8
 6 Person_006    29  91.2
 7 Person_007    37  64.3
 8 Person_008    41  91.9
 9 Person_009    30  72.6
10 Person_010    24  70.3
# i 90 more rows
```

This is all totally fine. There are, however, use cases where we might want to convert a `character` variable to a `factor` variable. Factors are useful when we have a categorical variable with a fixed number of levels, and we want to specify the order of those levels. For example, if we had a variable called `education_level` with the values "High School", "Bachelor's", "Master's", and "PhD", we might want to convert this variable to a factor and specify the order of the levels.

Let's make a new dataset to illustrate this:

```
my_data3 <- tibble(
  name = c("Alice", "Bob", "Charlie", "David", "Eve"),
  education_level = c("Bachelor's", "Master's", "PhD", "High School", "Bachelor's"),
  age = c(19, 23, 25, 16, 20)
)
```

Look at the structure of this new dataset:

```
my_data3
```

```
# A tibble: 5 x 3
  name    education_level   age
  <chr>   <chr>           <dbl>
1 Alice   Bachelor's         19
2 Bob     Master's           23
3 Charlie PhD                25
4 David   High School        16
5 Eve     Bachelor's         20
```

We can see that the `education_level` variable is of type `<chr>`, which is `character`.

We can convert the `education_level` variable to a factor:

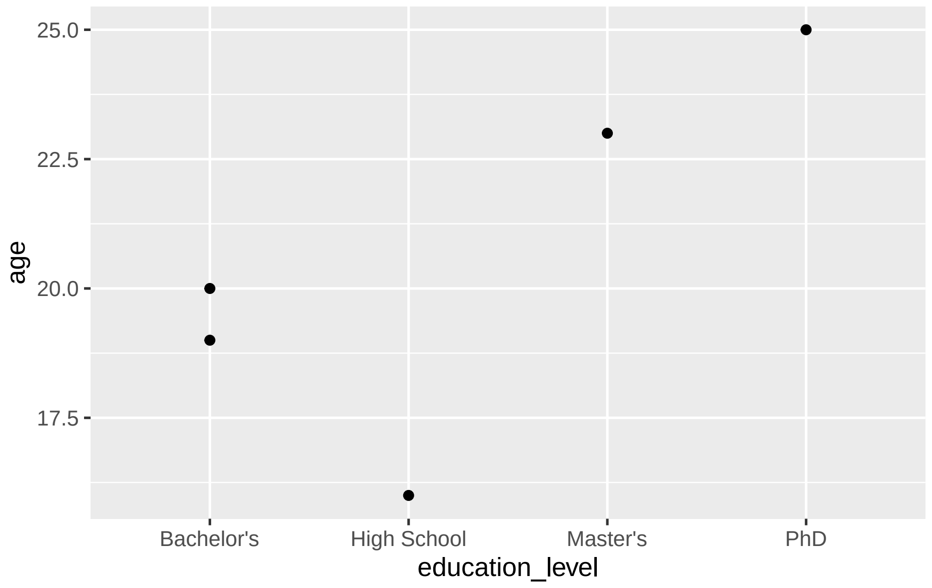
```
my_data3 <- my_data3 |>
  mutate(education_level = factor(education_level))
my_data3
```

```
# A tibble: 5 x 3
  name    education_level   age
  <chr>   <fct>           <dbl>
1 Alice   Bachelor's         19
2 Bob     Master's           23
3 Charlie PhD                25
4 David   High School        16
5 Eve     Bachelor's         20
```

Now, the `education_level` variable is of type `<fct>`, which is `factor`.

Here is a graph of age by education level:

```
ggplot(my_data3, aes(x = education_level, y = age)) +
  geom_point()
```



We have a problem here: the education levels are not in a sensible order. The first level is "Bachelor's", followed by "High School", "Master's", and "PhD".

> **Note**
>
> Why do you think the levels are in this order? We didn't tell R to order them like this! The answer is that R orders factor levels alphabetically by default. So when we convert a character variable to a factor without specifying the order of the levels, R will order them alphabetically.

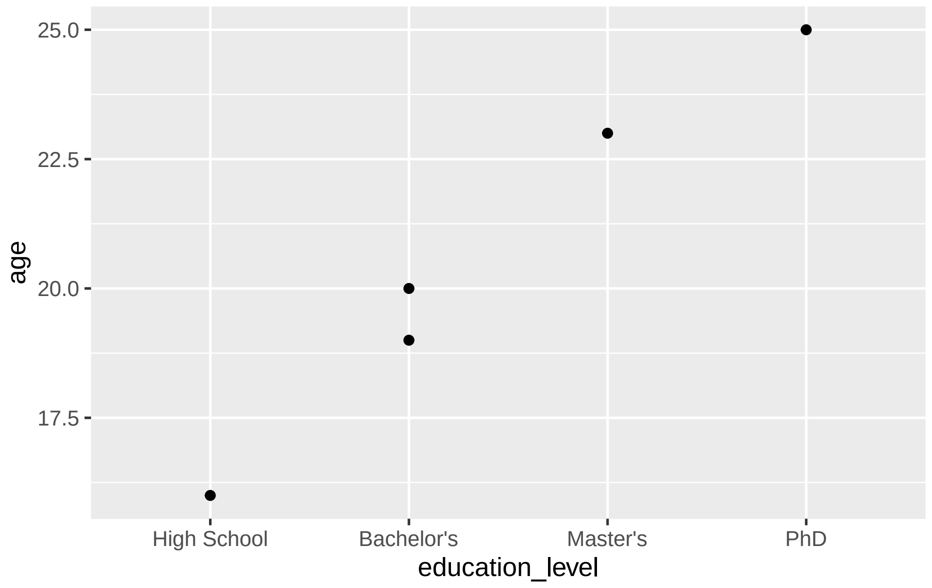
It would be much better to have the levels ordered as "High School", "Bachelor's", "Master's", and "PhD".

We can fix this by specifying the order of the levels when we convert the variable to a factor:

```
my_data3 <- my_data3 |>
  mutate(education_level = factor(education_level,
                                levels = c("High School", "Bachelor's", "Master's", "
```

Now when we plot the data again, the education levels are in the correct order:

```
ggplot(my_data3, aes(x = education_level, y = age)) +
  geom_point()
```



Another use case is when we are making a linear model and want to specify the reference level for a categorical variable. We will look at this when we get to linear models. If you want to skip ahead, you can see how this works in a section at the end of this chapter.

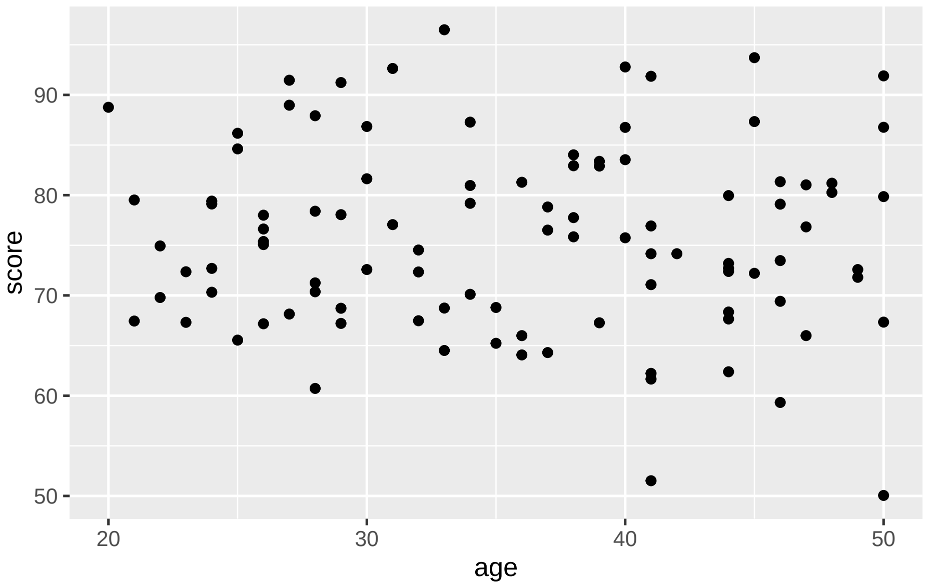
Visualisation

There are many many many types of data visualisation. We will not explore them all in this course! In fact, we will use only a few basic types of visualisation, but we will use them well and critically. The three types of visualisation we will focus on are scatter plots, histograms, and box and whisker plots.

Three basic types of visualisation [#three-basic-visualisations]

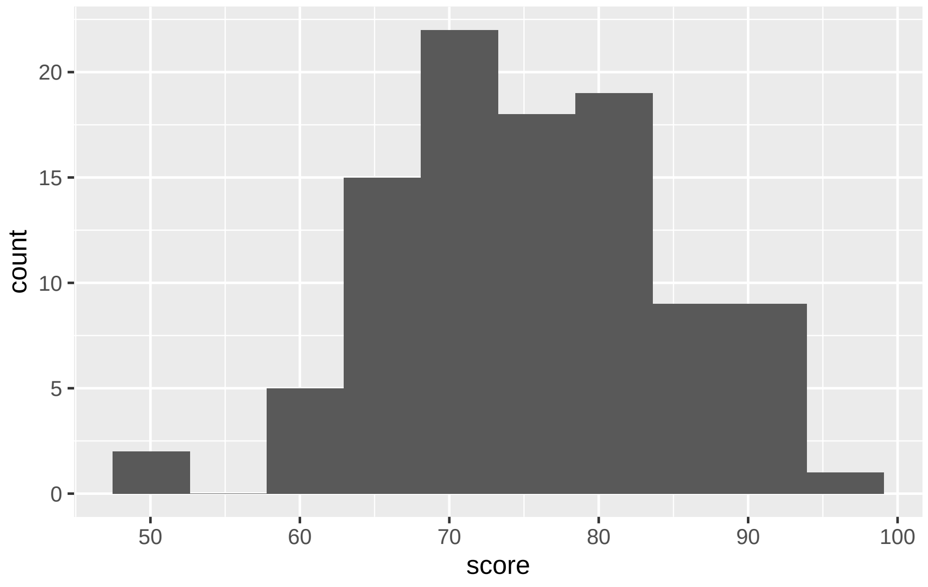
Scatterplots are used to visualise the relationship between two continuous variables. Here is an example of a scatterplot:

```
library(ggplot2)
ggplot(my_data2, aes(x = age, y = score)) +
  geom_point()
```



Histograms are used to visualise the distribution of a single continuous variable. The axiss are different to scatterplots: the x-axis is the variable being measured, and the y-axis is the count (or frequency) of observations in each bin. A bin is a range of values. Here is an esample of a histogram:

```
ggplot(my_data2, aes(x = score)) +
  geom_histogram(bins = 10)
```

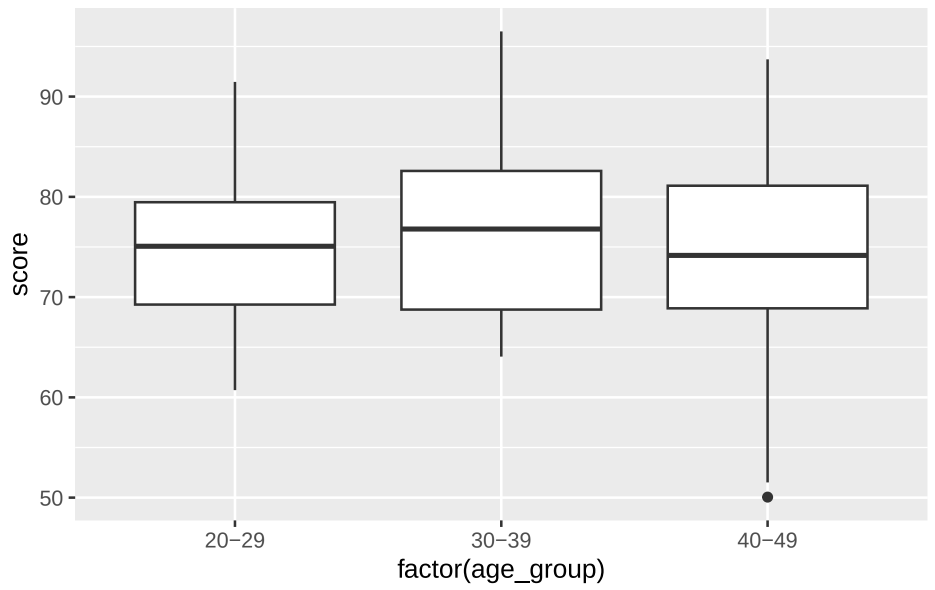


Box and whisker plots are used to visualise the distribution of a continuous variable across different categories. Here is an example of a box and whisker plot. First we add a new variable that is age group:

```r
my_data2 <- my_data2 |>
  mutate(age_group = case_when(
    age < 30 ~ "20-29",
    age >= 30 & age < 40 ~ "30-39",
    age >= 40 ~ "40-49"
  ))
```

The new variable `age_group` is a categorical variable with three levels: “20-29”, “30-39”, and “40-49”. We make this using the `case_when()` function. This function works by checking each condition (which are given as the arguments to the function) in turn, and assigning the corresponding value when the condition is true. Now we can make the box and whisker plot:

```r
ggplot(my_data2, aes(x = factor(age_group), y = score)) +
  geom_boxplot()
```



Understanding ggplot2 syntax [#understanding-ggplot2]

We have used the `ggplot2` package to create visualisations. The `ggplot2` package is based on the grammar of graphics, which provides a consistent way to create visualisations. It is amazing, and when it was created it revolutionised data visualisation in R.

You can see that for each of the three visualisations, we use the `ggplot()` function to create the base plot, and then we add layers to the plot using the `+` operator.

The first argument to the `ggplot()` function is the data frame that we want to visualise. The layers that we add to the plot each have two main components. The first component is the *aesthetic mappings*, which specify how the variables in the data frame are mapped to the visual properties of the plot (e.g., x-axis, y-axis, color, size). The second component is the *geometric object*, which defines how the data is represented in the plot (e.g., points, lines, bars).

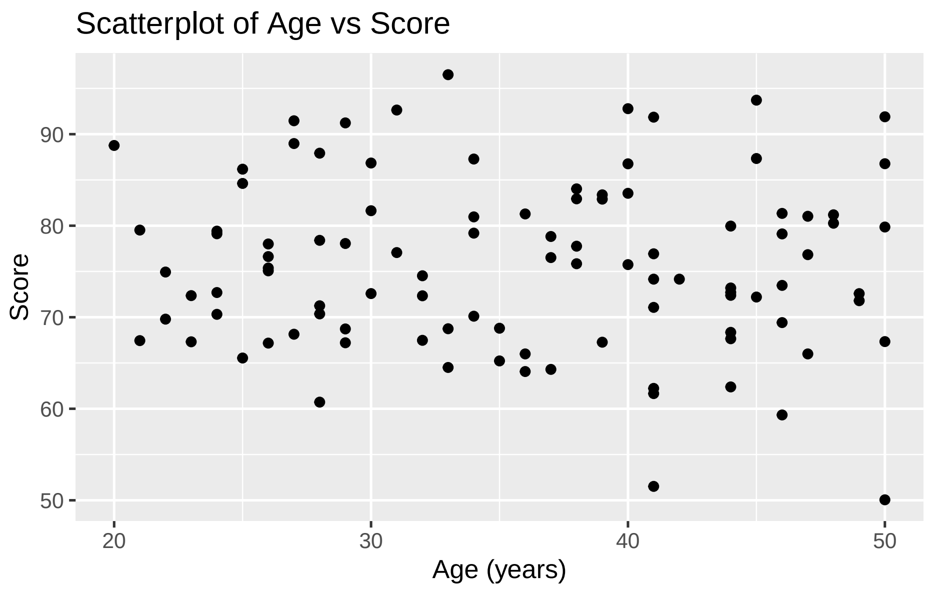
The *aesthetic mappings* are specified using the `aes()` function, which takes arguments that define the mappings. Inside the `aes()` function, we specify the variables from the data frame that we want to map to the visual properties of the plot. For example, in the scatterplot, we map the `age` variable to the x-axis and the `score` variable to the y-axis using `aes(x = age, y = score)`.

The *geometric object* is specified using functions that start with `geom_`, such as `geom_point()`, `geom_histogram()`, and `geom_boxplot()`.

You will notice that for the scatterplot and the box and whisker plot, we specify both an x- and a y-variable, but for the histogram we only specify an x-variable. This is because histograms only have one variable, which is the variable being measured. The y-axis is automatically calculated as the count (or frequency) of observations in each bin.

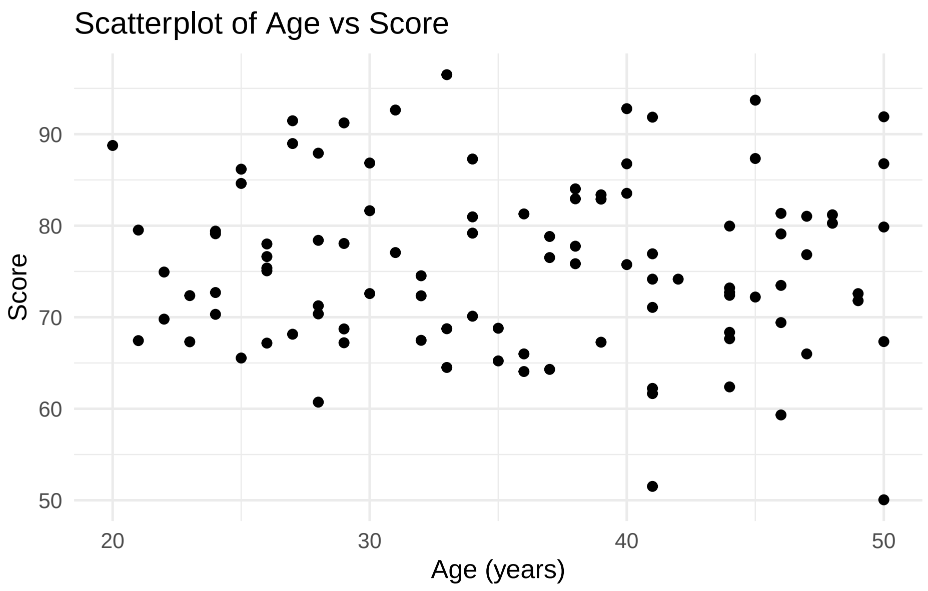
We can customise many features of the graph using additional arguments to the `ggplot()` function and the `geom_` functions. For example, we can add titles and labels to the axes using the `labs()` function:

```r
ggplot(my_data2, aes(x = age, y = score)) +
  geom_point() +
  labs(
    title = "Scatterplot of Age vs Score",
    x = "Age (years)",
    y = "Score"
  )
```



We can also change the theme of the plot using the `theme_` functions. For example, to use a minimal theme, and add it the customisations we already made:

```r
ggplot(my_data2, aes(x = age, y = score)) +
  geom_point() +
  labs(
    title = "Scatterplot of Age vs Score",
    x = "Age (years)",
    y = "Score"
  ) +
  theme_minimal()
```



There are a million and one ways to customise visualisations in `ggplot2`. We will explore many of them during the course in a rather ad-hoc way. In this course we do not *assess* your skill and competence in making clear and beautiful visualisations. We will, however, be very happy to help you make beautiful and effective visualisations for your assignments and projects. And please be sure that making beautiful and effective visualisations is a skill that is very highly valued in the workplace.

Saving ggplot visualisations [#saving-ggplot]

Another feature that is very useful is to save ggplot visualisations to objects and then save to a file (for example a pdf). First, here is how we save a ggplot to an object:

```
plot1 <- ggplot(my_data2, aes(x = age, y = score)) +
  geom_point() +
  labs(
    title = "Scatterplot of Age vs Score",
    x = "Age (years)",
    y = "Score"
  )
```

Now we can save the plot to a file using the `ggsave()` function:

```
ggsave("scatterplot_age_vs_score.pdf", plot = plot1, width = 8, height = 6)
```

Note two things about the `ggsave()` function. First, the first argument is the file name (including the file extension). The file extension determines the file type (e.g., pdf, png, jpeg). Second, we can specify the width and height of the plot in inches.

Also note that the file is saved to the current working directory. When you're working in an R project, this is usually the base directory of the project. If you want to save your plots in a folder named `plots` you would first need to create the folder (if it doesn't already exist) and then specify the path in the file name:

```
dir.create("plots")  # Create the folder if it doesn't exist
```

```
Warning in dir.create("plots"): 'plots' already exists
```

```
ggsave("plots/scatterplot_age_vs_score.pdf", plot = plot1, width = 8, height = 6)
```

Review

In this chapter we learned about using R and RStudio for data analysis. We covered the basics of R programming, including data types, variables, functions, and control structures. We also learned about importing data into R, cleaning and wrangling data using the `dplyr` package, and visualising data using the `ggplot2` package. A great starting point for your journey into data analysis with R!

Extras

Making reports directly using Quarto [#quarto-reports]

We don't explicitly ask you to make reports using Quarto in this course, but it is a very useful skill to have, and I highly recommend you explore it further in your own time. Here are a few basics to get you started.

One of the great features of R and RStudio is the ability to create reports that combine text, code, and visualisations. One of the most popular tools for this is Quarto (https://quarto.org/), which allows you to create documents in various formats (HTML, PDF, Word, etc.) using a combination of *Markdown* and R code.

**Why is this so great???* If you want to show someone your analysis and visualisation, say a team member or supervisor, it is often good to prepare a report that explains what you did, perhaps shows the code you used, and presents the results (including visualisations). One way to go about this is to prepare a powerpoint presentation or a word document, and then copy and paste code and visualisations into the document. Its what I used to do. It works. But it is tedious, error prone, and when you change something in your code or data, you have to remember to go back and update the powerpoint or word document.

With Quarto, you can create a report that automatically includes the code and visualisations directly from your R script. This way, if you change the code or data, you can simply re-render the report and everything is automatically updated. It takes away a lot of the tediousness and potential for errors. And it makes updating reports much easier.

If you'd like to get started with Quarto, check out the Quarto website (https://quarto.org/) and the RStudio Quarto documentation (https://quarto.org/docs/get-started/). There are also many tutorials and resources available online to help you learn how to use Quarto effectively.

If you have questions about Quarto, feel free to ask me or TAs during the practicals, though note that any particular TAs may or may not be experienced with Quarto themselves.

Combining ggplots with patchwork [#combining-ggplots]

We often make multiple ggplots in our analyses. Sometimes it is useful to combine multiple plots into a single figure for easier comparison or presentation. We can do with ggplots and the lovely add-on package called `patchwork`. The `patchwork` package allows us to combine multiple ggplots into a single plot layout. Here is an example of how to use `patchwork` to combine the three plots we made earlier (scatterplot, histogram, and boxplot):

First, load the `patchwork` package:

```r
library(patchwork)
```

Next make the first plot and assign it to an object:

```r
plot1 <- ggplot(my_data2, aes(x = age, y = score)) +
  geom_point() +
  labs(
    title = "Scatterplot of Age vs Score",
    x = "Age (years)",
    y = "Score"
  )
```

Now make the second plot and assign it to an object:

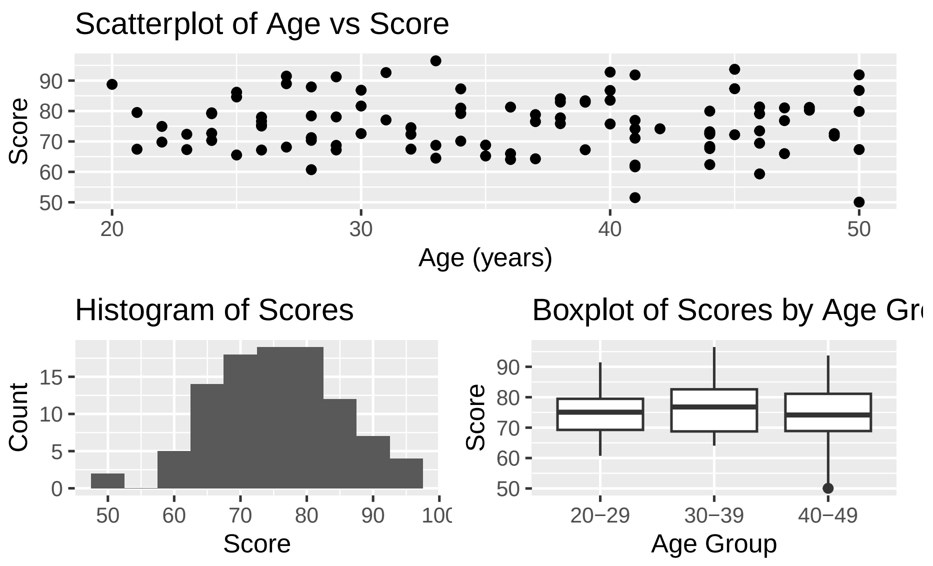
```r
plot2 <- ggplot(my_data2, aes(x = score)) +
  geom_histogram(binwidth = 5) +
  labs(
    title = "Histogram of Scores",
    x = "Score",
    y = "Count"
  )
```

Now make the third plot and assign it to an object:

```r
plot3 <- ggplot(my_data2, aes(x = factor(age_group), y = score)) +
  geom_boxplot() +
  labs(
    title = "Boxplot of Scores by Age Group",
    x = "Age Group",
    y = "Score"
  )
```

Now we can combine the three plots into a single layout using the `patchwork` syntax. Here, we arrange `plot1` on the top row, and `plot2` and `plot3` side by side on the bottom row:

```
combined_plot <- plot1 / (plot2 | plot3)
combined_plot
```



Amazing eh! OK, lets leave it there for now. We'll use ggplot2 throughout the course, and explore more features as we go along.

Setting a reference level in a linear model

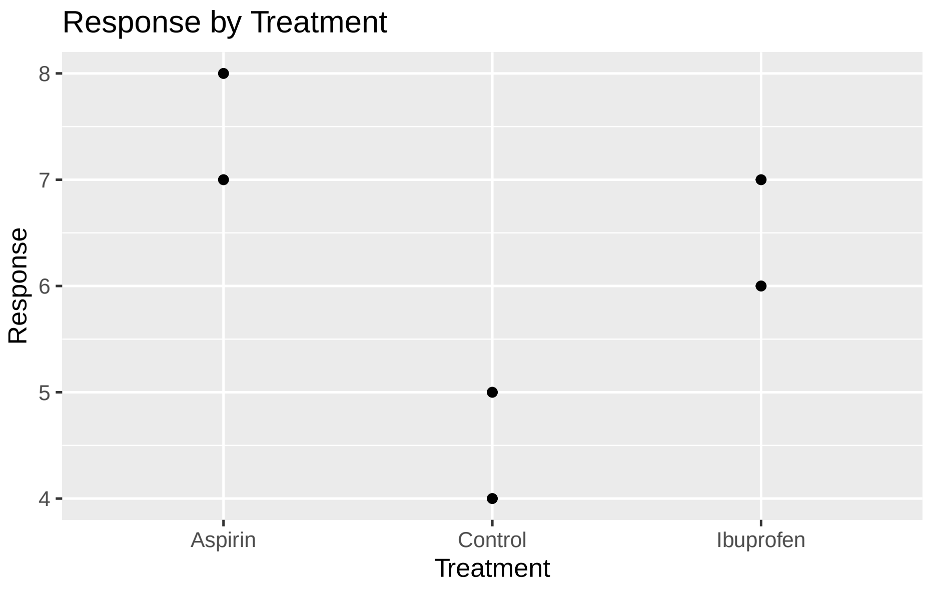
Sometimes when fitting linear models with categorical explanatory (independent) variables, it is useful to set a specific reference level for the categorical variable. This can help in interpreting the model coefficients. In R, we can set the reference level using the `relevel()` function or by using the `factor()` function with the `levels` argument.

First, let's create a simple dataset:

```
my_data4 <- tibble(
  treatment = factor(c("Control", "Aspirin", "Ibuprofen", "Control", "Aspirin", "Ibuprofen")),
  response = c(5, 7, 6, 4, 8, 7)
)
```

By default, R will set the first level of the factor (in alphabetical order) as the reference level. In this case, "Aspirin" would be the reference level. Therefore when we visualise the data:

```
ggplot(my_data4, aes(x = treatment, y = response)) +
  geom_point() +
  labs(
    title = "Response by Treatment",
    x = "Treatment",
    y = "Response"
  )
```



It would be nicer to have the "Control" group as the first level on the left of the x-axis.

Likewise, when we make a linear model:

```
model1 <- lm(response ~ treatment, data = my_data4)
summary(model1)
```

```

Call:
lm(formula = response ~ treatment, data = my_data4)

Residuals:
   1    2    3    4    5    6 
 0.5 -0.5 -0.5 -0.5  0.5  0.5 
```

```
Coefficients:
                   Estimate Std. Error t value Pr(>|t|)
(Intercept)          7.5000     0.5000  15.000 0.000643 ***
treatmentControl    -3.0000     0.7071  -4.243 0.023981 *
treatmentIbuprofen  -1.0000     0.7071  -1.414 0.252215
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.7071 on 3 degrees of freedom
Multiple R-squared:  0.8615,    Adjusted R-squared:  0.7692
F-statistic: 9.333 on 2 and 3 DF,  p-value: 0.05152
```

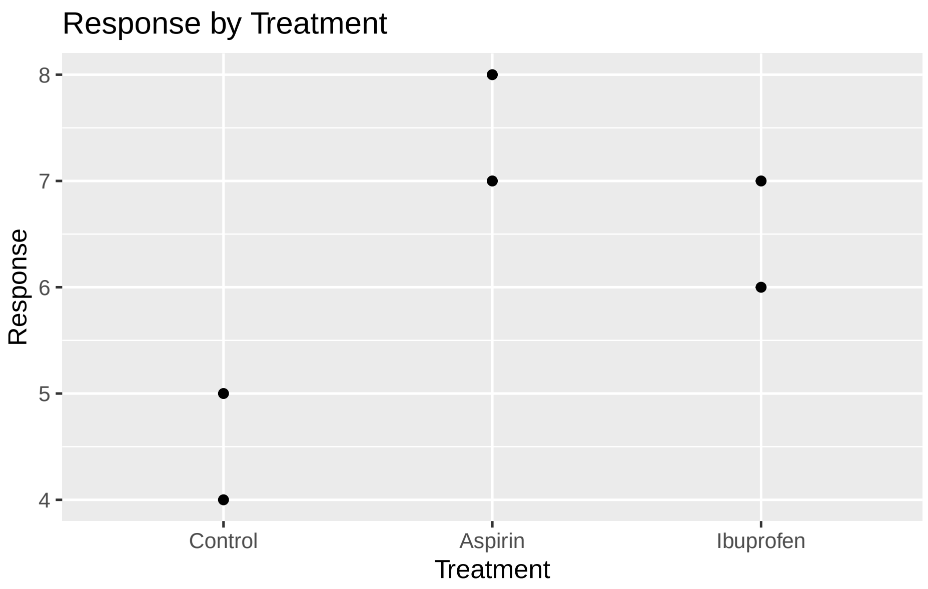
The (Intercept) term corresponds to the "Aspirin" group, and the coefficients for "Control" and "Ibuprofen" are relative to "Aspirin". *R* has done this because in the factor levels, "Aspirin" comes first alphabetically and was therefore set as the reference level when the factor variable was created.

If we want to set "Control" as the reference level, we can do so using `relevel()`:

```r
my_data4 <- my_data4 %>%
  mutate(treatment = relevel(treatment, ref = "Control"))
```

Now when we visualise the data again:

```r
ggplot(my_data4, aes(x = treatment, y = response)) +
  geom_point() +
  labs(
    title = "Response by Treatment",
    x = "Treatment",
    y = "Response"
  )
```



Magic! The "Control" group is now the first level on the left of the x-axis.

And when we fit the linear model again:

```r
model2 <- lm(response ~ treatment, data = my_data4)
summary(model2)
```

```

Call:
lm(formula = response ~ treatment, data = my_data4)

Residuals:
   1    2    3    4    5    6 
 0.5 -0.5 -0.5 -0.5  0.5  0.5 

Coefficients:
                   Estimate Std. Error t value Pr(>|t|)   
(Intercept)          4.5000     0.5000   9.000   0.0029 **
treatmentAspirin     3.0000     0.7071   4.243   0.0240 * 
treatmentIbuprofen   2.0000     0.7071   2.828   0.0663 . 
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.7071 on 3 degrees of freedom
Multiple R-squared:  0.8615,	Adjusted R-squared:  0.7692 
F-statistic: 9.333 on 2 and 3 DF,  p-value: 0.05152
```

The (Intercept) term now corresponds to the "Control" group, and the coefficients for "Aspirin" and "Ibuprofen" are relative to "Control". This makes interpretation of the model coefficients more intuitive.

Regression Part 1 (L3)

Introduction

Linear regression is a common statistical method that models the relationship between a dependent (response) variable and one or more independent (explanatory) variables. The relationship is modeled with the equation for a straight line ($y = a + bx$).

With linear regression we can answer questions such as:

- How does the dependent (response) variable change with respect to the independent (explanatory) variable?
- What amount of variation in the dependent variable can be explained by the independent variable?
- Is there a statistically significant relationship between the dependent variable and the independent variable?
- Does the linear model fit the data well?

In this chapter / lesson we will explore what is linear regression and how to use it to answer these questions. We'll cover the following topics:

- Why use linear regression?
- What is the linear regression model?
- Fitting the regression model (= finding the intercept and the slope).
- Is linear regression a good enough model to use?
- What do we do when things go wrong?
- Transformation of variables/the response.
- Identifying and handling odd data points (aka outliers).

In this chapter / lesson we will not discuss the statistical significance of the model. We will cover this topic in the next chapter / lesson.

Why use linear regression?

- It's a good starting point because it is a relatively simple model.
- Relationships are sometimes close enough to linear.
- It's easy to interpret.

- It’s easy to use.
- It’s actually quite flexible (e.g. can be used for non-linear relationships, e.g., a quadratic model is still a linear model!!!).

An example - blood pressure and age

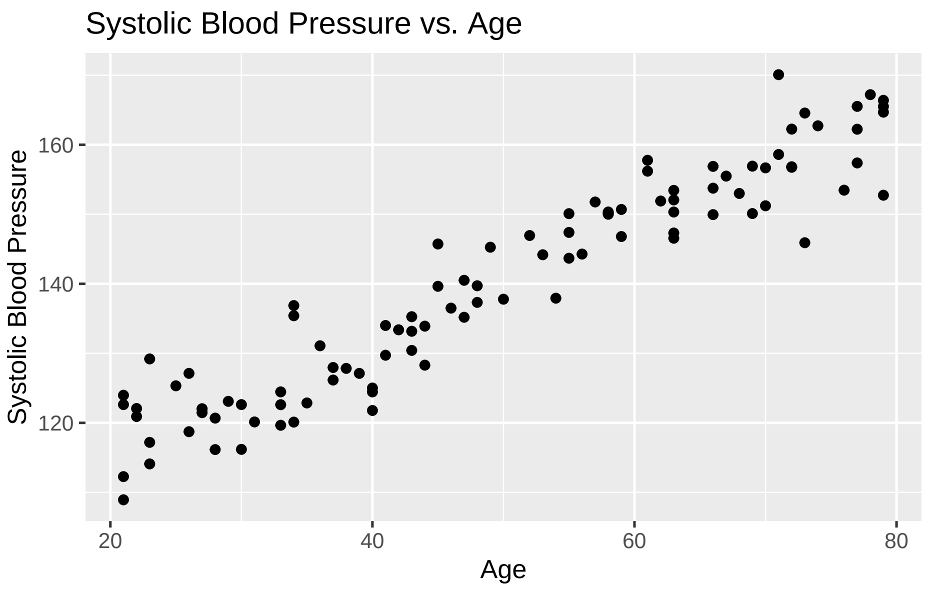
There are lots of situations in which linear regression can be useful. For example, consider hypertension. Hypertension is a condition in which the blood pressure in the arteries is persistently elevated. Hypertension is a major risk factor for heart disease, stroke, and kidney disease. It is estimated that hypertension affects about 1 billion people worldwide. Hypertension is a complex condition that is influenced by many factors, including age. In fact, it is well known that blood pressure increases with age. But how much does blood pressure increase with age? This is a question that can be answered using linear regression.

Here is an example of a study that used linear regression to answer this question: https://journals.lww.com/jhypertension/fulltext/2021/06000/association_of_age_and_blood_pressure

In this study, the authors used linear regression to model the relationship between age and blood pressure. They found that systolic blood pressure increased by 0.28–0.85 mmHg/year. This is a small increase, but it is statistically significant. This means that the observed relationship between age and blood pressure is unlikely to be due to chance.

Lets look at some simulated example data:

```r
# Load the data
bp_age_data <- read.csv("datasets/Simulated_Blood_Pressure_and_Age_Data.csv")

# How many data points with both age and blood pressure?
bp_age_data <- na.omit(bp_age_data)
# No rows with missing values

# Visualize the data
ggplot(bp_age_data, aes(x = Age, y = Systolic_BP)) +
  geom_point() +
  labs(title = "Systolic Blood Pressure vs. Age",
       x = "Age",
       y = "Systolic Blood Pressure")
```



Well, that is pretty conclusive. We hardly need statistics. There is a clear positive relationship between age and systolic blood pressure. But how can we quantify this relationship? And in less clear-cut cases what is the strength of evidence for a relationship? This is where linear regression comes in. Linear regression models the relationship between age and systolic blood pressure. With linear regression we can answer the following questions:

- What is the value of the intercept and slope of the relationship?
- Is the relationship different from what we would expect if there were no relationship?
- How well does the mathematical representation match the observed values?
- How much uncertainty is there in predictions?

Lets try to figure some of these out from the visualisation.

Think-Pair-Share (#tps-guess-params) Make a guess of the slope. Make a guess of the intercept (hint be careful, lots of people get this wrong).

Calculating the intercept and slope

Regression from a mathematical perspective

Given an **independent/explanatory variable** (X) and a **dependent/response variable** (Y) all points (x_i, y_i), $i = 1, \dots, n$, on a straight line follow the equation

$$y_i = \beta_0 + \beta_1 x_i \ .$$

- β_0 is the **intercept** - the value of Y when $x_i = 0$
- β_1 the **slope** of the line, also known as the regression coefficient of X.
- If $\beta_0 = 0$ the line goes through the origin $(x, y) = (0, 0)$.
- **Interpretation** of linear dependency: proportional increase in y with increase (decrease) in x.

Finding the intercept and the slope

In a regression analysis, one task is to estimate the intercept and the slope. These are known as the **regression coefficients** β_0, β_1.

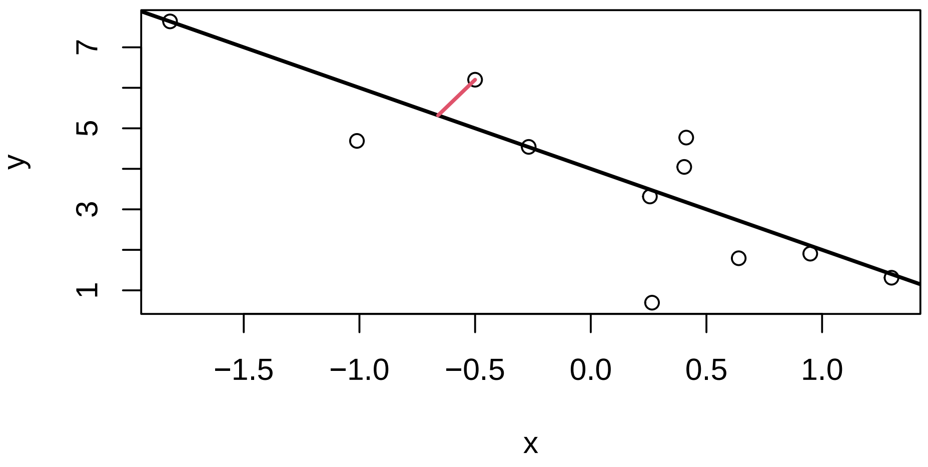
- **Problem**: For more than two points (x_i, y_i), $i = 1, \dots, n$, there is generally no perfectly fitting line.

- **Aim:** We want to estimate the parameters (β_0, β_1) of the **best fitting** line $Y = \beta_0 + \beta_1 x$.

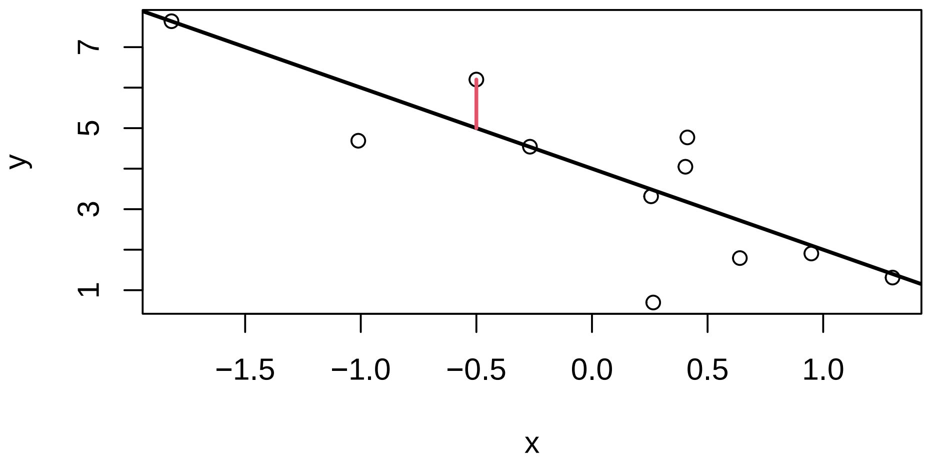
- **Idea:** Find the **best fitting line** by minimizing the deviations between the data points (x_i, y_i) and the regression line. I.e., minimising the residuals.

But which deviations?

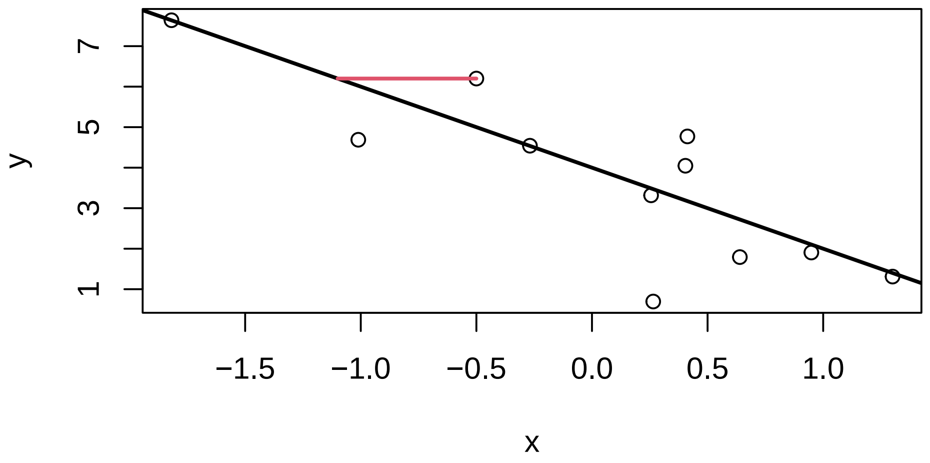
These ones?



Or these?



Or maybe even these?



Well, actually its none of these!!!

Least squares

For multiple reasons (theoretical aspects and mathematical convenience), the intercept and slope are estimated using the **least squares** approach. In this, yet something else is minimized:

The parameters β_0 and β_1 are estimated such that the **sum of squared vertical distances** (sum of squared residuals / errors) is minimised.

SSE means **S**um of **S**quared **E**rrors:

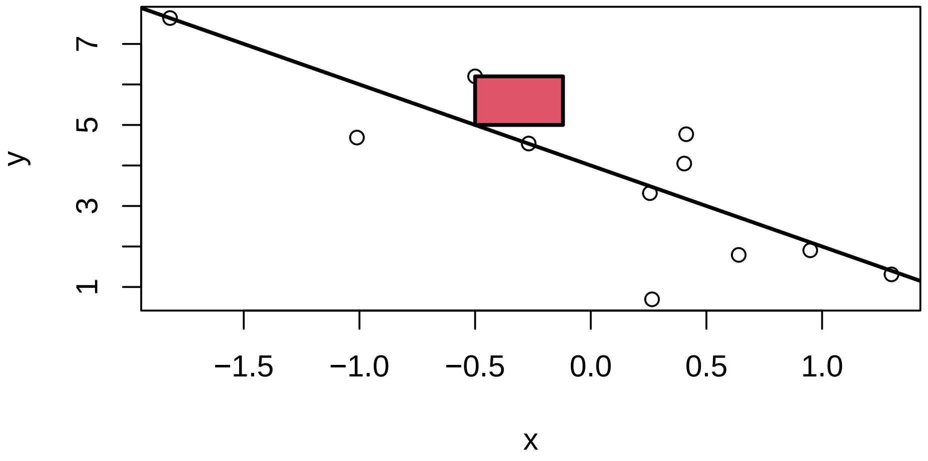
$$SSE = \sum_{i=1}^{n} e_i^2$$

where,

$$e_i = y_i - \underbrace{(\beta_0 + \beta_1 x_i)}_{=\hat{y}_i}$$

Note: $\hat{y}_i = \beta_0 + \beta_1 x_i$ are the *predicted values.*

In the graph just below, one of these squares is shown in red.



Note

Residuals are model-based quantities, not properties of the raw data.

Least squares estimates

With a linear model, we can calculate the least squares estimates of the parameters β_0 and β_1 directly using the following formulas.

For a given sample of data $(x_i, y_i), i = 1, .., n$, with mean values $\overline{x}$ and $\overline{y}$, the least squares estimates $\hat{\beta}_0$ and $\hat{\beta}_1$ are computed as

$$\hat{\beta}_1 = \frac{\sum_{i=1}^{n}(y_i - \overline{y})(x_i - \overline{x})}{\sum_{i=1}^{n}(x_i - \overline{x})^2} = \frac{cov(x, y)}{var(x)}$$

$$\hat{\beta}_0 = \overline{y} - \hat{\beta}_1 \overline{x}$$

Moreover,

$$\hat{\sigma}^2 = \frac{1}{n-2} \sum_{i=1}^{n} e_i^2 \quad \text{with residuals } e_i = y_i - (\hat{\beta}_0 + \hat{\beta}_1 x_i)$$

is an unbiased estimate of the residual variance σ^2.

(Derivations of the equations above are in the Stahel script 2.A b. Hint: differentiate, set to zero, solve.)

Why division by $n - 2$ ensures an unbiased estimator

When estimating parameters (β_0 and β_1), the square of the residuals is minimised. This fitting process inherently *uses up* two *degrees of freedom*, as the model forces the residuals to sum to zero and aligns the slope to best fit the data. I.e., one degree of freedom is lost due to the estimation of the intercept, and another due to the estimation of the slope.

The adjustment (division by $n - 2$ instead of n) compensates for the loss of variability due to parameter estimation, ensuring the estimator of the residual variance is unbiased. Mathematically, dividing by n - 2 adjusts for this loss and gives an accurate estimate of the population variance when working with sample data.

We'll look at degrees of freedom in more detail later, so don't worry if this is a bit confusing right now.

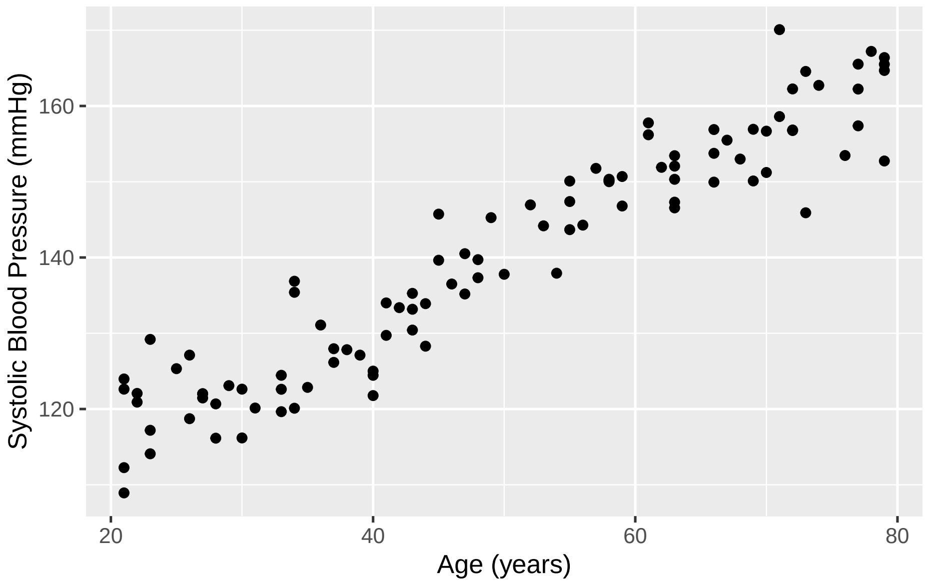
Let's do it in R

First we read in the dataset:

```r
bp_age_data <- read.csv("datasets/Simulated_Blood_Pressure_and_Age_Data.csv")
```

The we make a graph of the data:

```r
ggplot(bp_age_data, aes(x = Age, y = Systolic_BP)) +
  geom_point() +
  labs(x = "Age (years)", y = "Systolic Blood Pressure (mmHg)")
```



Then we make the linear model, using the `lm()` function:

```
bp_age_model <- lm(Systolic_BP ~ Age, data = bp_age_data)
```

Then we can look at the summary of the model. It contains a lot of information, so can be a bit confusing at first.

```
summary(bp_age_model)
```

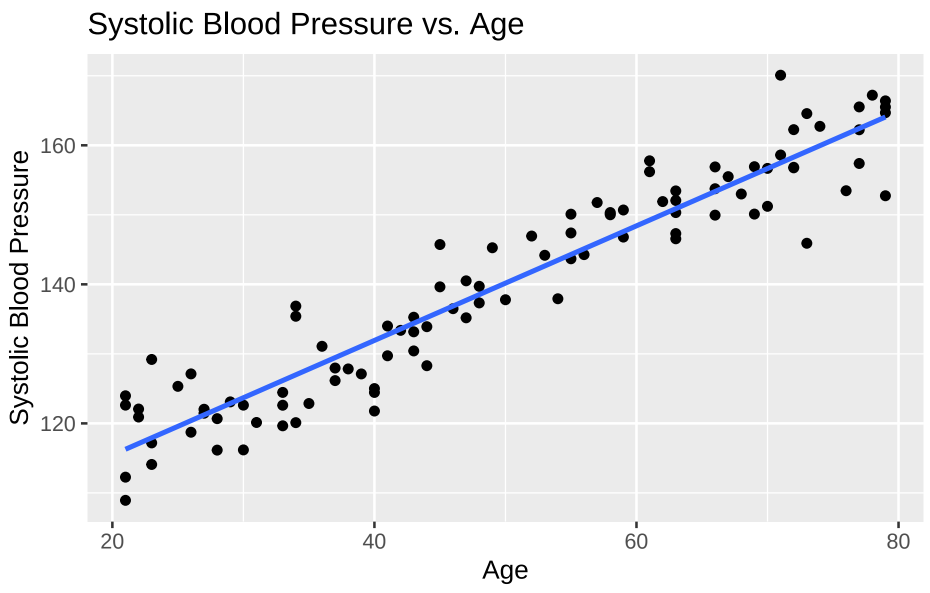
```

Call:
lm(formula = Systolic_BP ~ Age, data = bp_age_data)

Residuals:
     Min       1Q   Median       3Q      Max 
-13.2195  -3.4434  -0.0808   3.1383  12.6025 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept) 98.96874    1.46102   67.74   <2e-16 ***
Age          0.82407    0.02771   29.74   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 4.971 on 98 degrees of freedom
Multiple R-squared:  0.9002,	Adjusted R-squared:  0.8992 
F-statistic: 884.4 on 1 and 98 DF,  p-value: < 2.2e-16
```

How do our guesses of the intercept and slope compare to the guesses we made earlier?

Recal that the units of the *Age* coefficient are in mmHg per year. This means that for each additional year of age, the systolic blood pressure increases by ´r round(coef(bp_age_model)[2],2)´ mmHg.

Think–Pair–Share (#a_interpreting_slope) In words, what does the slope represent in this model? What does it *not* tell you?

Dealing with the error



The line is not a perfect fit to the data. There is scatter around the line.

Some of this scatter could be caused by other factors that influence blood pressure, such as diet, exercise, and genetics. Also, the there could be differences due to the measurement instrument (i.e., some measurement error).

These other factors are not included in the model (only age is in the model), so they create variation that can only appear in error term.

In the linear regression model the dependent variable Y is related to the independent variable x as

$$Y = \beta_0 + \beta_1 x + \epsilon$$

where

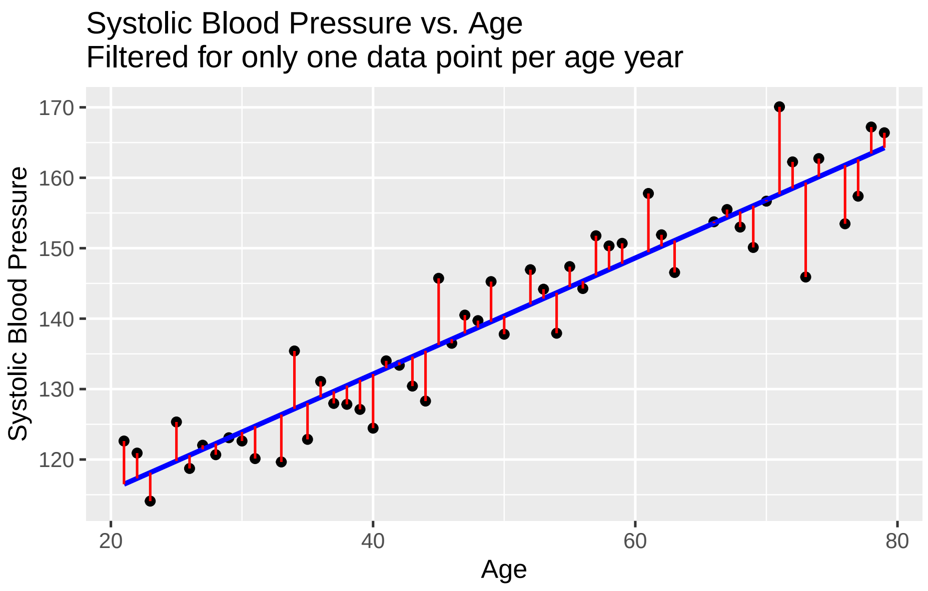
- ϵ is the error term
- β_0 is the intercept

- β_1 is the slope
- ϵ is the error term.

The error term captures the difference between the observed value of the dependent variable and the value predicted by the model. The error term includes the effects of other factors that influence the dependent variable, as well as measurement error.

$$Y = \underbrace{\text{expected value}}_{E(Y)=\beta_0+\beta_1 x} + \underbrace{\text{random error}}_{\epsilon}.$$

Graphically the error term is the vertical distance between the observed value of the dependent variable and the value predicted by the model.



The error term is also known as the residual. It is the variation that *resides* (is left over / is left unexplained) after accounting for the relationship between the dependent and independent variables.

Example in R

Let's look at observed values, expected (predicted) values, and residuals (error) in R.

The observed values of the response (dependent) variable are already in the dataset:

```
head(bp_age_data$Systolic_BP)
```

```
[1] 150.3277 170.0801 139.7174 135.4089 151.9041 122.0296
```

To get the expected values, we need to find the intercept and slope of the linear model. We can do this using the `lm()` function in R.

```
lm1 <- lm(Systolic_BP ~ Age, data = bp_age_data)
```

And we can get the intercept and slope using the `coef()` function:

```
coef(lm1)
```

```
(Intercept)         Age
 98.9687381   0.8240678
```

We can then use the mutate function from the dplyr package to add the expected values to the dataset:

```
bp_age_data <- bp_age_data %>%
  mutate(Expected_BP = coef(lm1)[1] + coef(lm1)[2] * Age)
```

And we can get the residuals by subtracting the expected values from the observed values:

```
bp_age_data <- bp_age_data %>%
  mutate(Residuals = Systolic_BP - Expected_BP)
```

Tip

We can also get the expected values and residuals directly from the `lm` object using the `fitted()` (or `predicted()`) and `residuals()` functions:

```
bp_age_data <- bp_age_data %>%
  mutate(Expected_BP = fitted(lm1),
         Residuals = residuals(lm1))
```

Now we have a model that gives the expected values (on the regression line) and that gives us a residual. Because the expected value plus the residual equals the observed value, if we use each of the residuals as the error for each respective data point, we end up with a perfect fit to the data. All we are doing is describing the observed data in a different way. This is known as over-fitting. In fact, we have gained very little by fitting the model. We have simply memorized / copied the data!!!

In order to avoid this, we need to assume something about the residuals – we need to *model* the residuals. The most common model for the residuals is a normal distribution with mean 0 and constant variance.

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

This is known as the normality assumption. The normality assumption is important because it allows us to make inferences about the *population parameters* based on the *sample data.*

The linear regression model then becomes:

$$Y = \beta_0 + \beta_1 x + N(0, \sigma^2)$$

where σ^2 is the variance of the error term. The variance of the error term is the amount of variation in the dependent variable that is not explained by the independent variable. The variance of the error term is also known as the residual variance.

An alternate and equivalent formulation is that Y is a random variable that follows a normal distribution with mean $\beta_0 + \beta_1 x$ and variance σ^2.

$$Y \sim N(\beta_0 + \beta_1 x, \sigma^2)$$

So, the answer to the question "how do we deal with the error term" is that we model the error term as normally distributed with mean 0 and constant variance. Put another way, the error term is assumed to be normally distributed with mean 0 and constant variance.

Back to blood pressure and age

The mathematical model in this case is:

$$SystolicBP = \beta_0 + \beta_1 \times Age + \epsilon$$

where: *SystolicBP* is the dependent (response) variable, β_0 is the intercept, β_1 is the coefficient of Age, *Age* is the independent (explanatory) variable, ϵ is the error term.

Let's ensure we understand this, by thinking about the units of the variables in this model. This can be very useful because it can help us to understand the model better and to check that the model makes sense.

Think-Pair-Share (#tps-what-units)

- What are the units of blood pressure?
- What are the units of age?
- What are the units of the intercept?
- What are the units of the coefficient of Age?
- What are the units of the error term?

Is the model good enough to use?

- All models are wrong, but is ours good enough to be useful?
- Are the assumption of the model justified?
- It would be very unwise to use the model before we know if it is good enough to use.
- *Don't jump out of an aeroplane until you know your parachute is good enough!*

What assumptions do we make?

We already heard about one. We assume that the residuals follow a $N(0, \sigma^2)$ distribution (that is, a Gaussian / Normal distrution with mean of zero and variance of σ^2). We make this assumption because it is often well enough met, and it gives great mathematical tractability.

This assumption implies that:

(a) The ϵ_i are normally distributed.
(b) ϵ_i has constant variance: $Var(\epsilon_i) = \sigma^2$.
(c) The ϵ_i are independent of each other.

Note

These assumptions are about the residuals, not the observed data!!!

Furthermore:

(d) we assumed a linear relationship.
(e) implies there are no outliers (implied by (a) above)

Lets go through each five assumptions.

(a) Normally distributed residuals

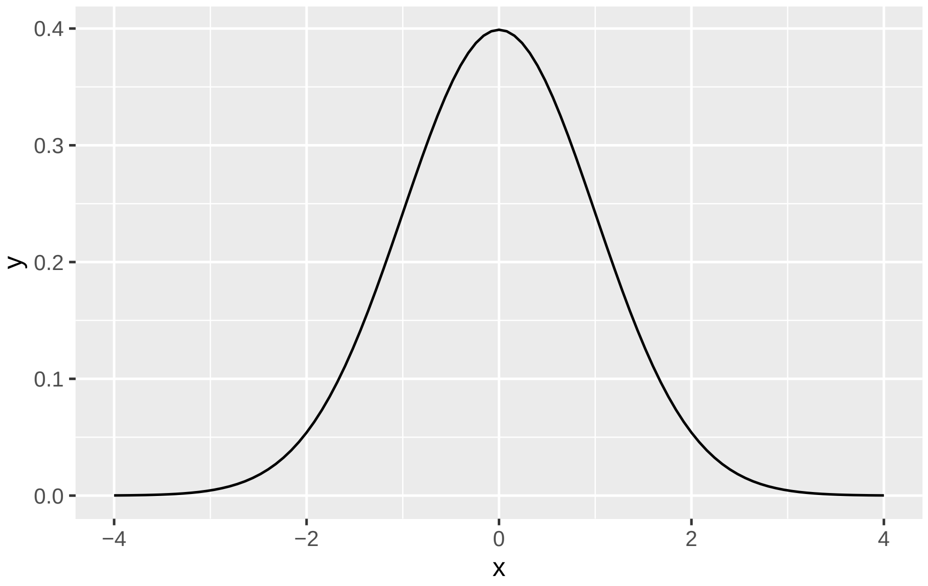
Recall that we make the assumption that the residuals are normally distributed with mean 0 and constant variance:

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

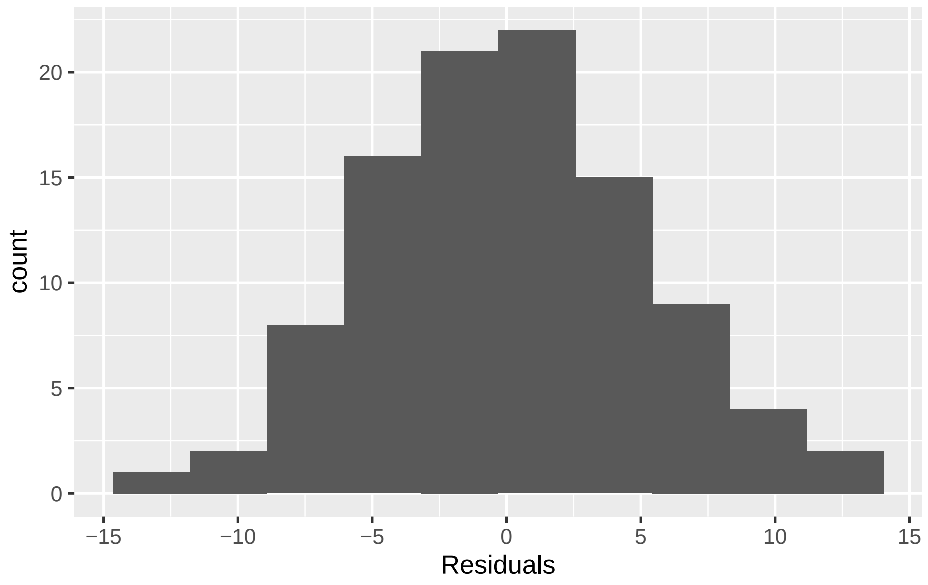
Here we are concerned with the first part of this assumption, that the residuals are normally distributed.

What does this mean? How can we check it?

A normal distribution is symmetric and bell-shaped…



Lets look at the frequency distribution of the residuals of the linear regression of blood pressure and age:

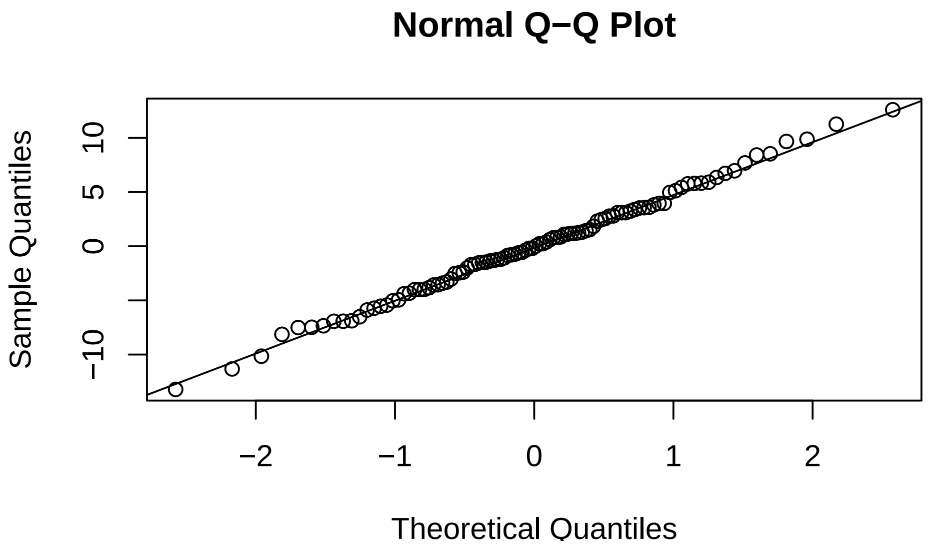


The normal distribution assumption (a) seems ok as well.

(a) Normally distributed residuals: The QQ-plot

Usually, not the histogram of the residuals is plotted, but the so-called **quantile-quantile** (QQ) plot. The quantiles of the observed distribution are plotted against the quantiles of the respective theoretical (normal) distribution:

```
qqnorm(residuals(bp_age_model))
qqline(residuals(bp_age_model))
```

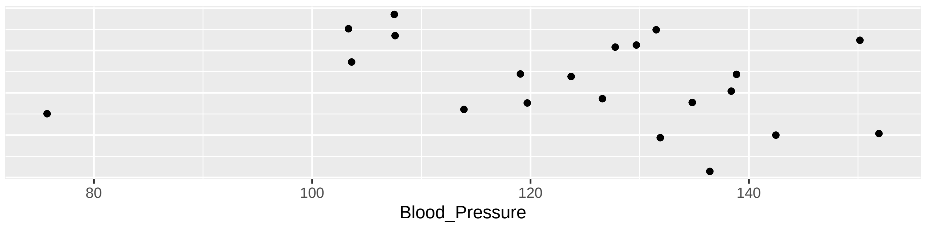


If the points lie approximately on a straight line, the data is fairly normally distributed.

This is often “tested” by eye, and needs some experience.

But what on earth is a quantile???

Imagine we make 21 measures of something, say 21 blood pressures:

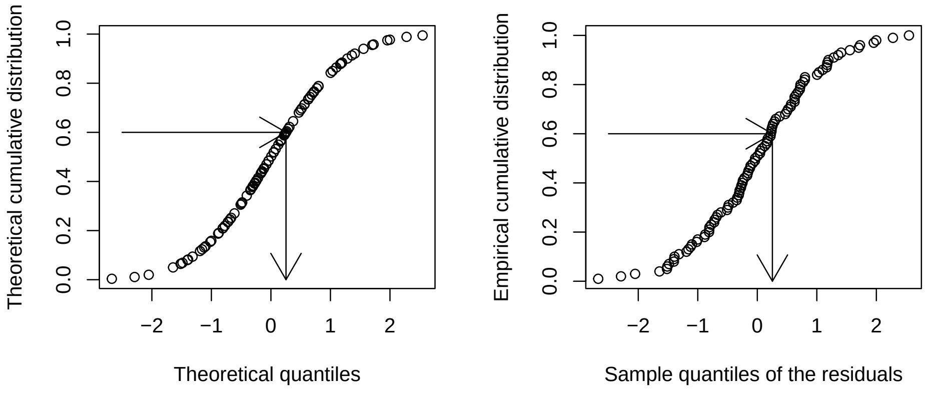


The median of these is 127.8. The median is the 50% or 0.5 quantile, because half the data points are above it, and half below.

```
quantile(dd$Blood_Pressure, 0.5)
```

```
  50%
127.8
```

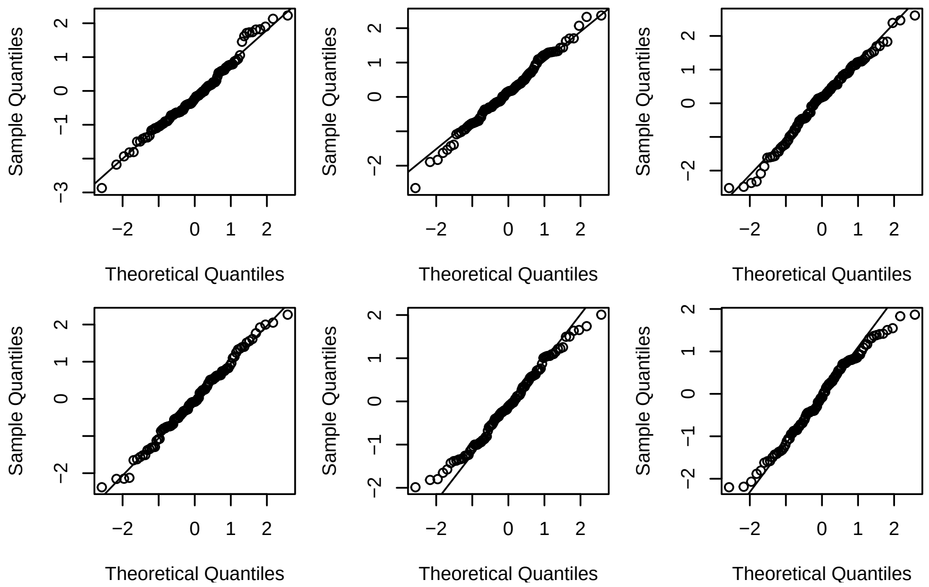
The *theoretical quantiles* come from the normal distribution. The *sample quantiles* come from the distribution of our residuals.



How do I know if a QQ-plot looks "good"?

There is **no quantitative rule** to answer this question. Instead experience is needed. You can gain this experience from simulations. To this end, we can generate the same number of data points of a normally distributed variable and compare this simulated qqplot to our observed one.

Example: Generate 100 points $\epsilon_i \sim N(0, 1)$ each time:



Each of the graphs above has data points that are randomly generated from a normal distribution. In all cases the data points are close to the line. This is what we would expect if the data were normally distributed. The amount of deviation from the line is what we would expect from random variation, and so seeing this amount of variation in a QQ-plot of your model should not be cause for concern.

(b) Constant error variance (homoscedasticity)

Recall that we assume the errors are normally distributed with constant variance σ^2:

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

Here we're concerned with the second part of this assumption, that the variance is constant.That is, variance of the residuals is a constant: $\text{Var}(\epsilon_i) = \sigma^2$. And not, for example $\text{Var}(\epsilon_i) = \sigma^2 \cdot x_i$.

Put another way, we're interested if the size of the residuals tends to show a pattern with the fitted values. By *size* of the residuals we mean the *absolute value* of the residuals. In fact, we often look at the square root of the absolute value of the standardized residuals:

$$R_i = \frac{\epsilon_i}{\hat{\sigma}}$$

Where $\hat{\sigma}$ is the estimated standard deviation of the residuals:

$$\hat{\sigma} = \sqrt{\frac{1}{n-2} \sum_{i=1}^{n} \epsilon_i^2}$$

So that the full equation of the square root of the standardised residuals is:

$$\sqrt{|R_i|} = \sqrt{\left|\frac{\epsilon_i}{\hat{\sigma}}\right|}$$

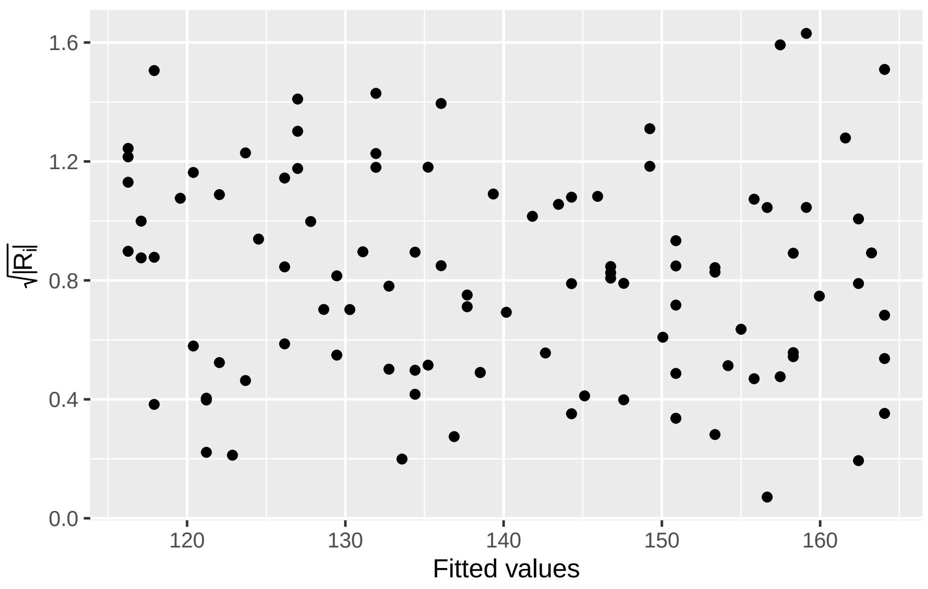
To look to see if the variance of the residuals is constant, we need to see if there is any relationship between the size of the residuals and the fitted values. A commonly used visualistion for this is a plot of the size of the residuals against the fitted values.

Lets first calculated the $\sqrt{|R_i|}$ values for our blood pressure model:

```r
bp_age_data <- bp_age_data %>%
  mutate(fitted = predict(bp_age_model),
         residuals = residuals(bp_age_model),
         sigma_hat = sqrt(sum(residuals^2)/(n()-2)),
         R_i = residuals / sigma_hat,
         sqrt_abs_R_i = sqrt(abs(R_i)))
```

And now visualise the relationship between the fitted values and the size of the residuals:

```r
ggplot(bp_age_data, aes(x = fitted, y = sqrt_abs_R_i)) +
  geom_point() +
  labs(x = "Fitted values", y = expression(sqrt(abs(R[i]))))
```

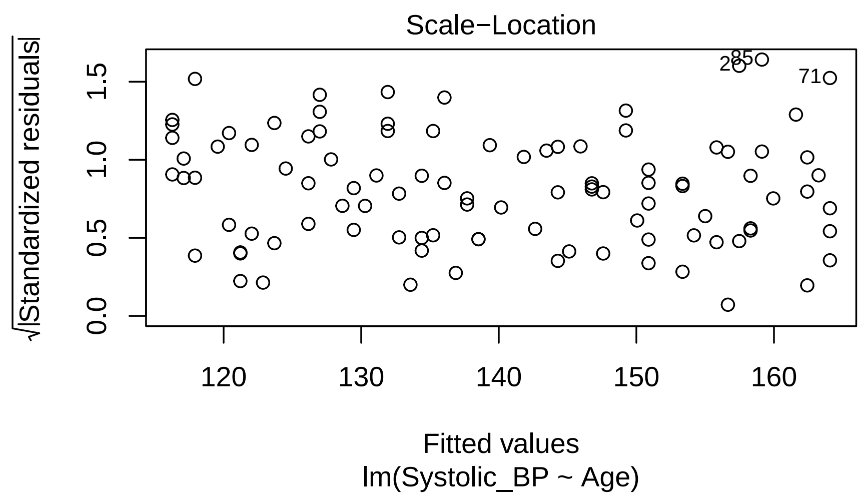


This graph is known as the scale-location plot. It is particularly suited to check the assumption of equal variances (**homoscedasticity / Homoskedastizität**). There should be **no trend** or pattern.

Tip

We can also use the built-in plot function for linear models to create this plot. It is the third plot in the set of model checking plots.

```r
plot(bp_age_model, which=3, add.smooth = FALSE)
```

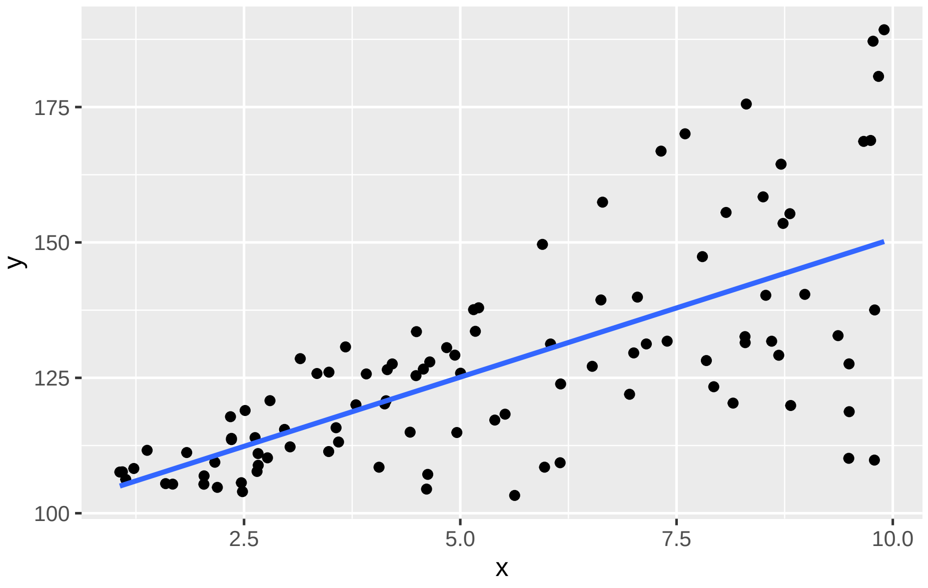


How it looks with the variance increasing with the fitted values

Here's a graphical example of how it would look if the variance of the residuals increases with the fitted values.

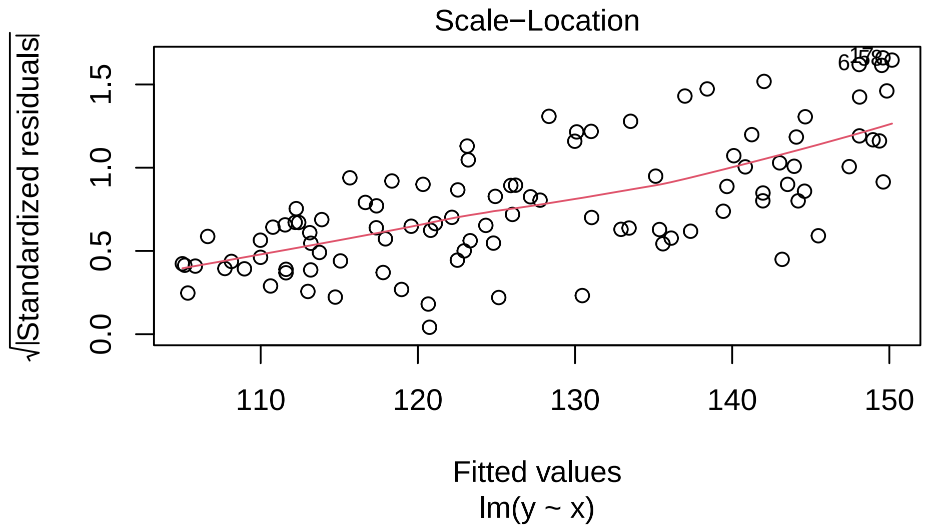
First here is a graph of the relationship:

```r
set.seed(2)
x <- runif(100, 1, 10)
y <- 100 + 5*x + 2*x*rnorm(100,0,1)
ggplot(data.frame(x=x,y=y), aes(x=x,y=y)) +
  geom_point() +
  geom_smooth(formula = y ~ x, method="lm", se=FALSE)
```



And here the scale-location plot for a linear model of that data:

```
m <- lm(y~x)
plot(m,which=3)
```



(c) Independence (residuals are independent of each other)

We assume that the residuals (ϵ_i) are independent of each other. This means that the value of one residual is not somehow related to the value of another.

The dataset about blood pressure we looked at contained 100 observations, each one made from a different person. In such a study design, we could be safe in the assumption that the people are independent, and therefore the assumption that the residuals are independent.

Imagine, however, if we had 100 observations of blood pressure collected from 50 people, because we measured the blood pressure of each person twice. In this case, the residuals would not be independent, because two measures of the blood pressure of the same person are likely to be similar. A person is likely to have a high blood pressure in both measurements, or a low blood pressure in both measurements. This would mean they have a high residual in both measurements, or a low residual in both measurements.

In this case, we would need to account for the fact that the residuals are not independent. We would need to use a more complex model, such as a mixed effects model, to account for the fact that the residuals are not independent. We will talk about this again in the last week of this course.

In general, you should always think about the study design when you are analysing data. You should always think about whether the residuals are likely to be independent of each other. If they are not, you should think about how you can account for this in your analysis.

A good way to assess if there could be dependencies in the residuals is to be critical about what is the unit of observation in the data. In the blood pressure example, the unit of observation is the person. Count the number of persons in the study. If there are fewer persons than observations, then at least some people must have been measured at least twice. Repeating measures on the same person is a common way to get dependent residuals.

So, to check the assumption of independence, you should:

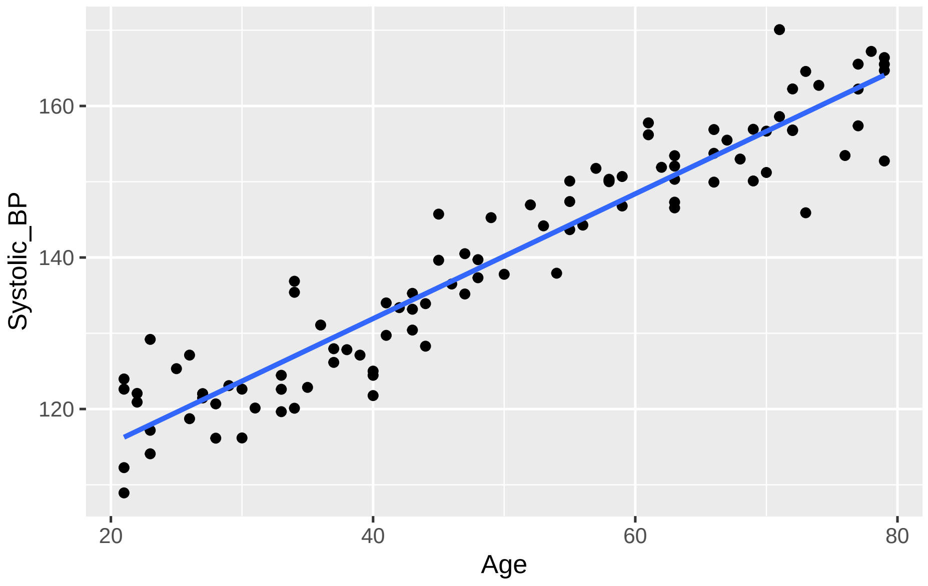
- Think carefully about the study design.
- Think carefully about the unit of observation in the data.
- Compare the number of observations to the number of units of observation.

(d) Linearity assumption

The linearity assumption states that the relationship between the independent variable and the dependent variable is linear. This means that the dependent variable changes by a constant amount for a one-unit change in the independent variable. And that this slope is does not change with the value of the independent variable.

The blood pressure data seems to be linear:

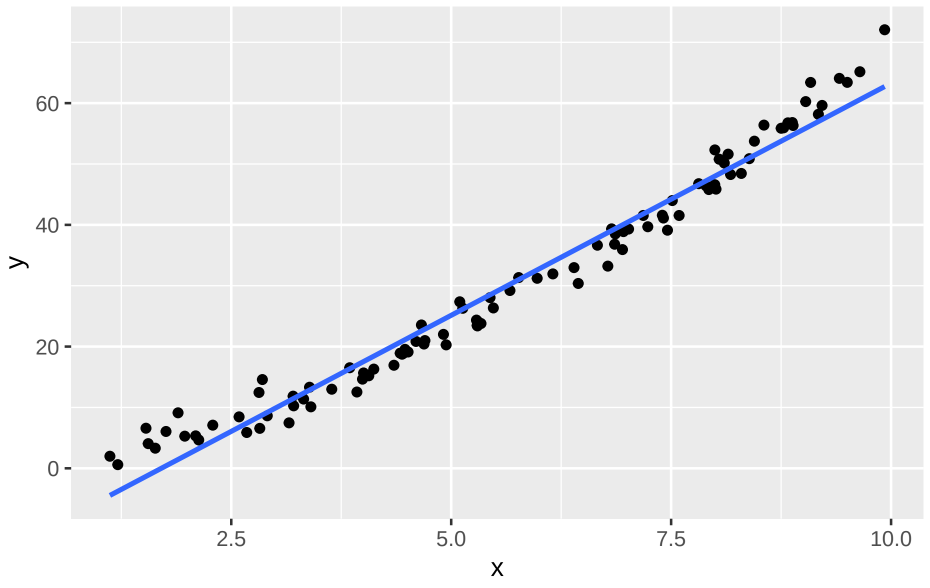
```
Warning: `fortify(<lm>)` was deprecated in ggplot2 3.6.0.
i Please use `broom::augment(<lm>)` instead.
i The deprecated feature was likely used in the ggplot2 package.
  Please report the issue at <https://github.com/tidyverse/ggplot2/issues>.
```



```
ggplot(bp_age_model, aes(x = Age, y=Systolic_BP)) +
  geom_point() +
  geom_smooth(formula = y ~ x, method="lm", se=FALSE)
```

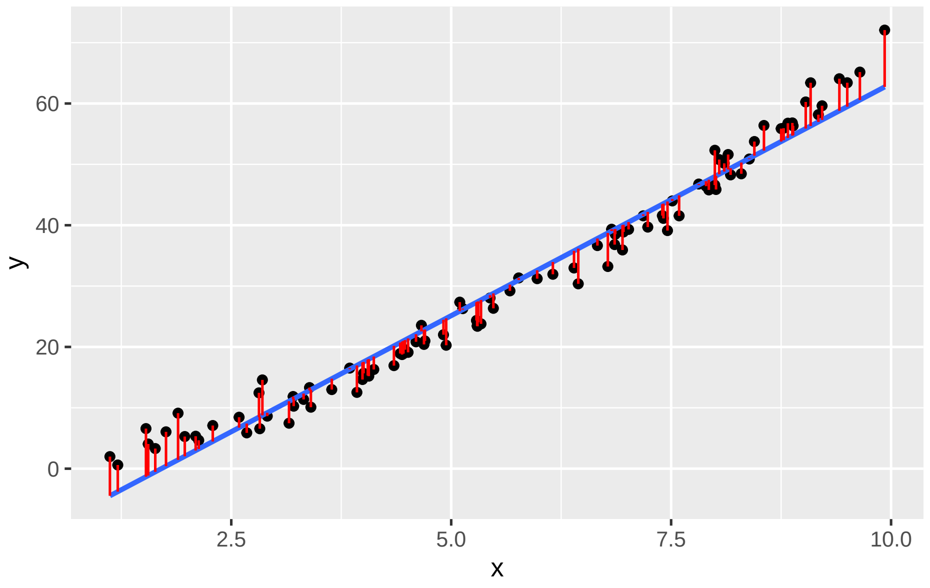
In contrast, look at this linear regression through data that appears non-linear:

```
set.seed(1)
x <- runif(100, 1, 10)
y <- 2*x + rnorm(100,0,2) + 0.5*x^2
m <- lm(y~x)
ggplot(data.frame(x=x,y=y), aes(x=x,y=y)) +
  geom_point() +
  geom_smooth(formula = y ~ x, method="lm", se=FALSE)
```



And with the residuals shown as red lines:

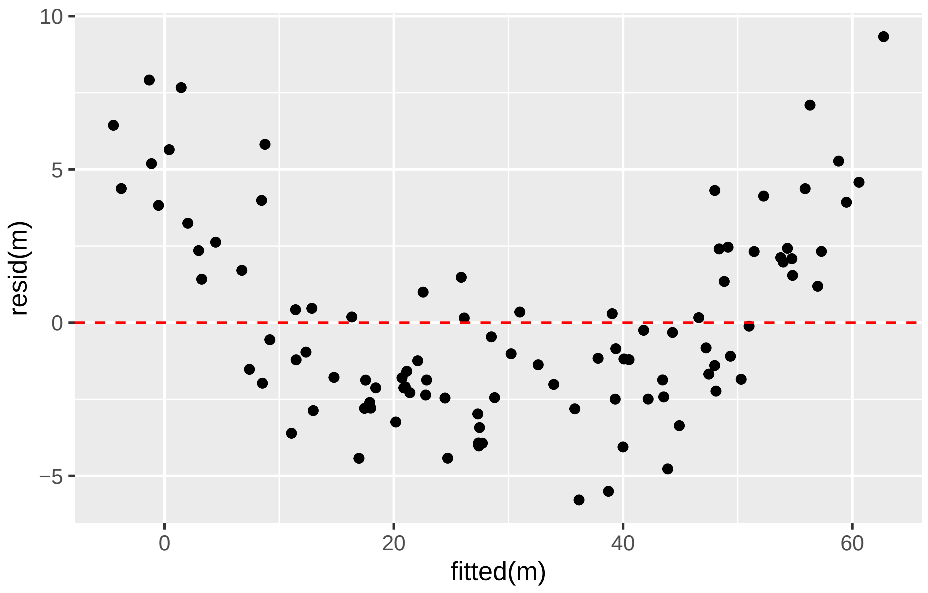
```
ggplot(data.frame(x=x,y=y), aes(x=x,y=y)) +
  geom_point() +
  geom_smooth(formula = y ~ x, method="lm", se=FALSE) +
  geom_segment(aes(xend=x, yend=m$fitted), color="red")
```



At low values of y, the residuals are positive, at intermediate values of y the residuals are negative, and at high values of y the residuals are positive. This pattern in the residuals is a sign that the relationship between x and y is not linear.

We can plot the value of the residuals against the y value directly, instead of looking at the pattern in the graph above. This is called a **Tukey-Anscombe plot**. It is a graph of the residuals versus the fitted y values:

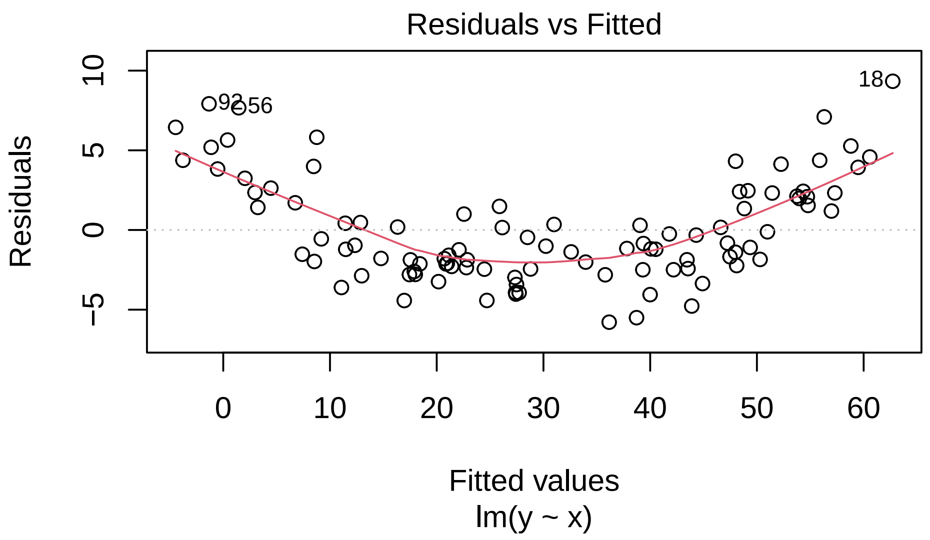
```r
ggplot(mapping = aes(x = fitted(m), y = resid(m))) +
  geom_point() +
  geom_hline(yintercept = 0, linetype="dashed", color="red")
```



We can very clearly see pattern in the residuals in this Tukey-Anscombe plot. The residuals are positive, then negative, then positive, as the fitted y value gets larger.

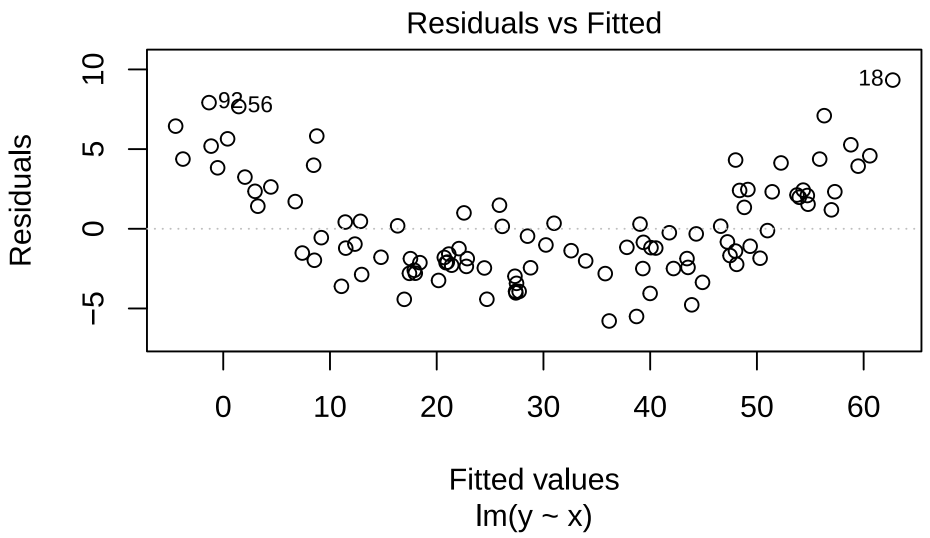
We can also make this Tukey-Anscombe plot using the built-in plot function for linear models in R:

```r
plot(m, which=1)
```



The red line in the Tukey-Anscombe plot is a loess smooth. It is automatically added to the plot. It is a way of estimating the pattern in the residuals. If the red line is not flat, then there is a pattern in the residuals. However, the loess smooth is not always reliable. It is a good idea to look at the residuals directly, without this smooth.

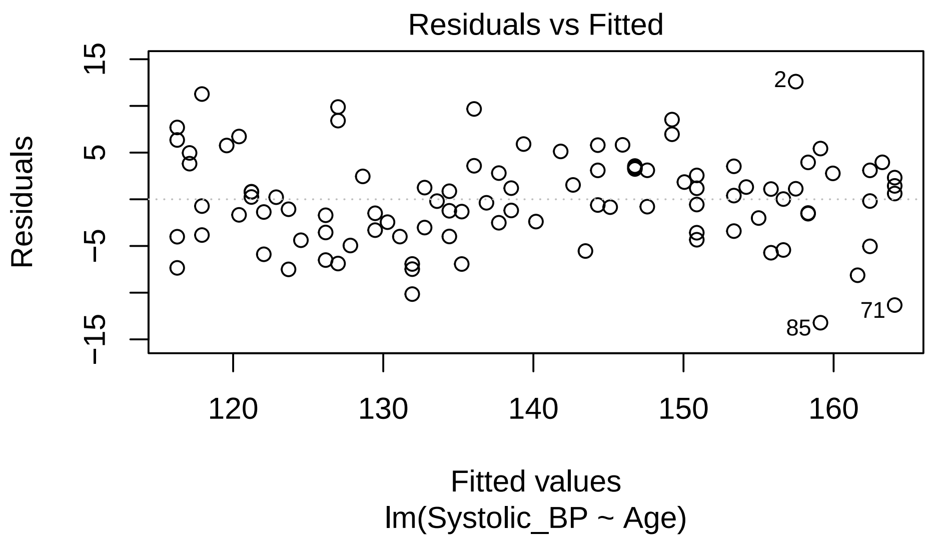
```
plot(m, which=1, add.smooth = FALSE)
```



The data here is simulated to show a very clear pattern in the residuals. In real data, the pattern might not be so clear. But if you suspect you see a pattern in the residuals, it could be a sign that the relationship between the independent and dependent variable is not linear.

Here is the Tukey-Anscombe plot for the blood pressure data:

```r
plot(bp_age_model, which=1, add.smooth = FALSE)
```



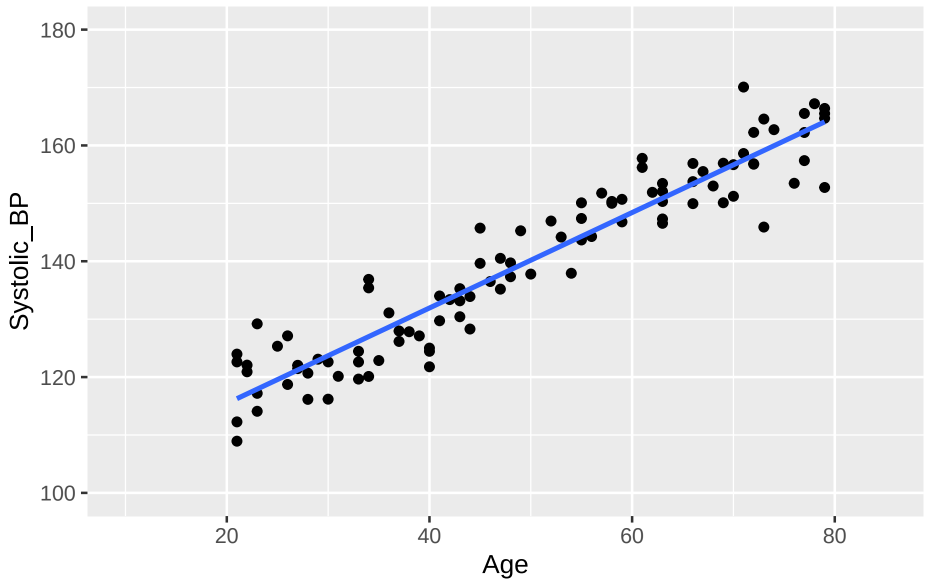
There is very little evidence of any pattern in the residuals. This data is simulated with a truly linear relationship, so we would not expect to see any pattern in the residuals.

(e) No outliers

An outlier is a data point that is very different from the other data points. Outliers can have a big effect on the results of a regression analysis. They can pull the line of best fit towards them, and make the line of best fit a poor representation of the data.

Lets again look at the blood pressure versus age data:

```r
ggplot(bp_age_model, aes(x = Age, y=Systolic_BP)) +
  geom_point() +
  geom_smooth(formula = y ~ x, method="lm", se=FALSE) +
  xlim(10, 85) +
  ylim(100, 180)
```



There are no obvious outliers in this data. The data points are all close to the line of best fit. This is a good sign that the line of best fit is a good representation of the data.

Think-Pair-Share (#tps-odd-data) Where on this graph would you expect to see particularly influential outliers? Influential in the sense that they would have a large effect on the slope of the line of best fit.

Data points that are far from the mean of the independent variable have a large effect on the value of the slope. These data points have a large leverage. They are data points that are far from the other data points in the x direction.

We can think of this with the analogy of a seesaw. The slope of the line of best fit is like the pivot point of a seesaw. Data points that are far from the pivot point have a large effect on the slope. Data points that are close to the pivot point have a small effect on the slope.

A measure of distance from the pivot point is called the *leverage* of a data point. In simple regression, the leverage of individual i is defined as

$h_i = (1/n) + (x_i - \overline{x})^2/SSX.$

where $SSX = \sum_{i=1}^{n}(x_i - \overline{x})^2$. (**S**um of **S**quares of X)

So, the leverage of a data point is inversely related to n (the number of data points). The leverage of a data point is also inversely related to the sum of the squares of the x values. The leverage of a data point is directly related to the square of the distance of the x value from the mean of the x values.

More intuitively perhaps, the leverage of a data point will be greater when the are fewer other data points. It will also be greater when the distance from the mean value of x is greater.

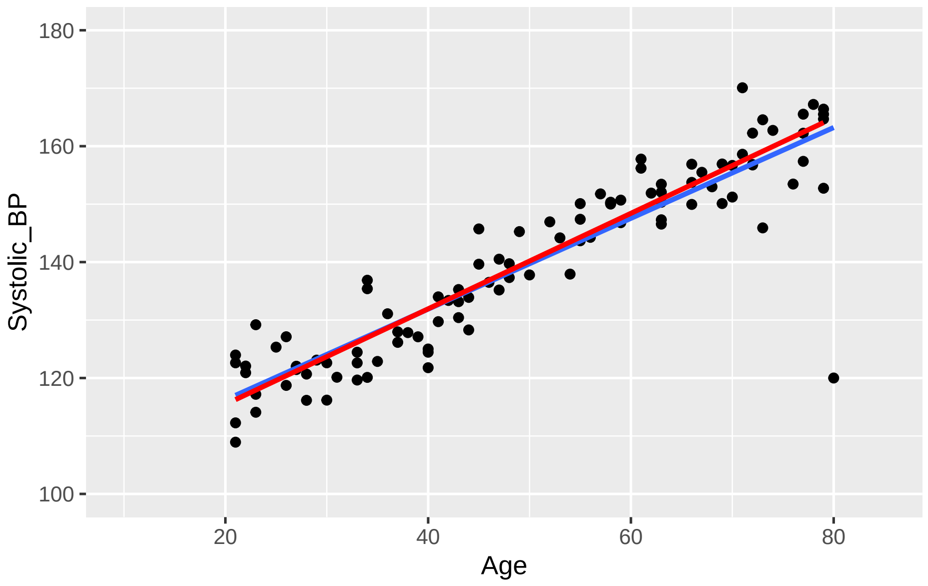
Going back to the analogy of a seesaw, with data points as children on the seesaw, the leverage of a data point is like the distance from the pivot a child sits. But we also have children of different weights. A lighter child will have less effect on the tilt of the seesaw. A heavier one will have a greater effect on the tilt. A heavier child sitting far from the pivot will have a very large effect.

Think-Pair-Share (#tps-like-weight) What quantity that we already experienced is like the weight of the child?

The size of the residuals are like the weight of the child. Data points with large residuals have a large effect on the slope of the line of best fit. Data points with small residuals have a small effect on the slope of the line of best fit.

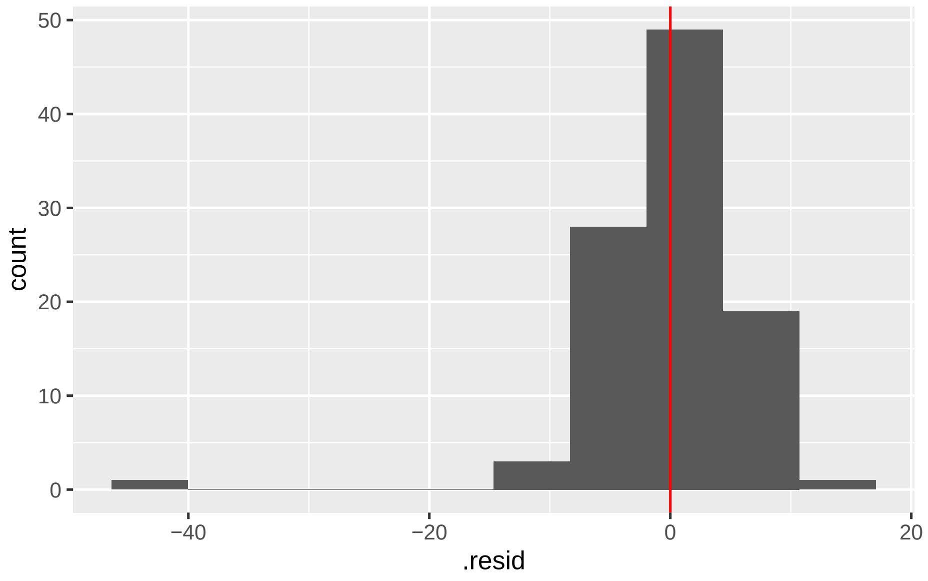
So the overall effect of a data point on the slope of the line of best fit is a combination of the leverage and the residual. This quantity is called the $influence$ of a data point.

Let's add a rather extreme data point to the blood pressure versus age data:



This is a bit ridiculous, but it is a good example of an outlier. The data point is far from the other data points. It has a large residual. And it is a long way from the pivot (the middle of the x data) so has large leverage.

We can make a histogram of the residuals and see that the outlier has a large residual:



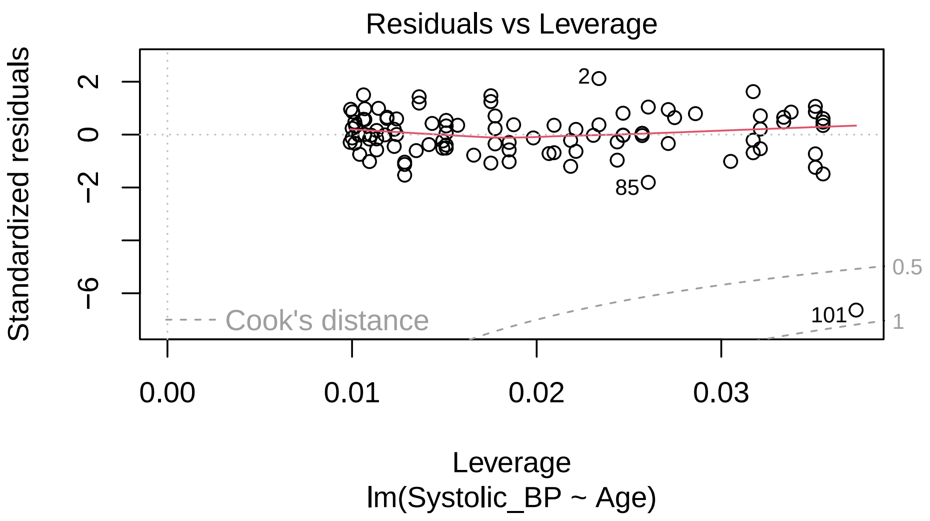
And we can see that the leverage is large.

There is a graph that we can look at to see the influence of a data point. This is called a $Cook's\ distance$ plot. The Cook's distance of a data point is a measure of how much the slope of the line of best fit changes when that data point is removed. The Cook's distance of a data point is defined as

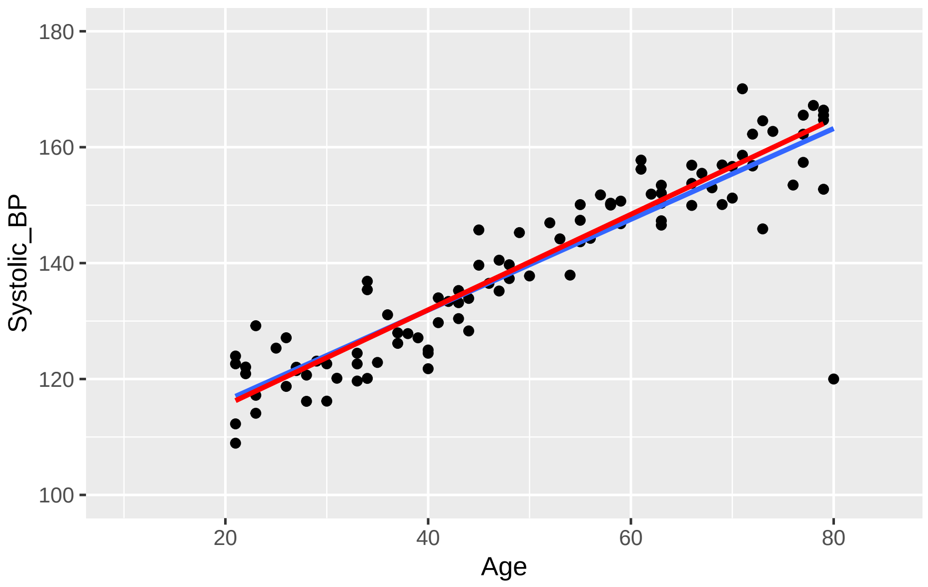
$D_i = \sum_{j=1}^{n}(\hat{y}_j - \hat{y}_{j(i)})^2/(p \times MSE)$.

where $\hat{y}_j$ is the predicted value of the dependent variable for data point j, $\hat{y}_{j(i)}$ is the predicted value of the dependent variable for data point j when data point i is removed, p is the number of parameters in the model (2 in this case), MSE is the mean squared error of the model.

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



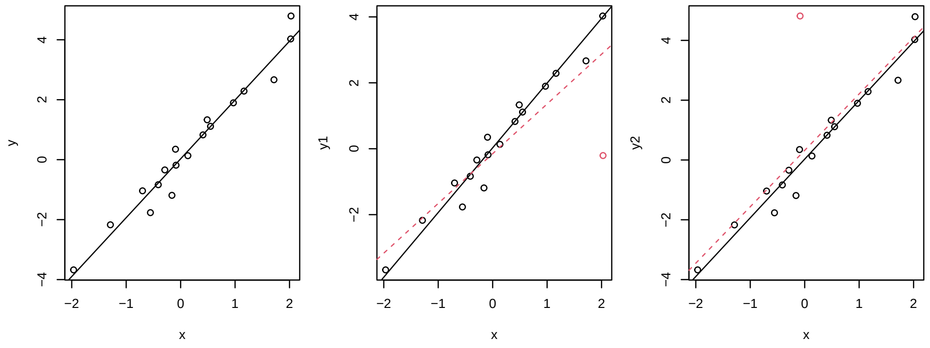
But does it have a large influence on the value of the slope? In the next graph we show the line of best fit with the outlier (blue line) and without the outlier (red line).



No, the outlier doesn't have much influence on the slope. The outlier has a large leverage. It is far from the pivot. But it does not have such a large effect (influence) on the slope. This is in large part because there are a lot data points (100) that are quite tightly arranged around the regression line.

Graphical illustration of the leverage effect

Data points with x_i values far from the mean have a stronger leverage effect than when $x_i \approx \overline{x}$:

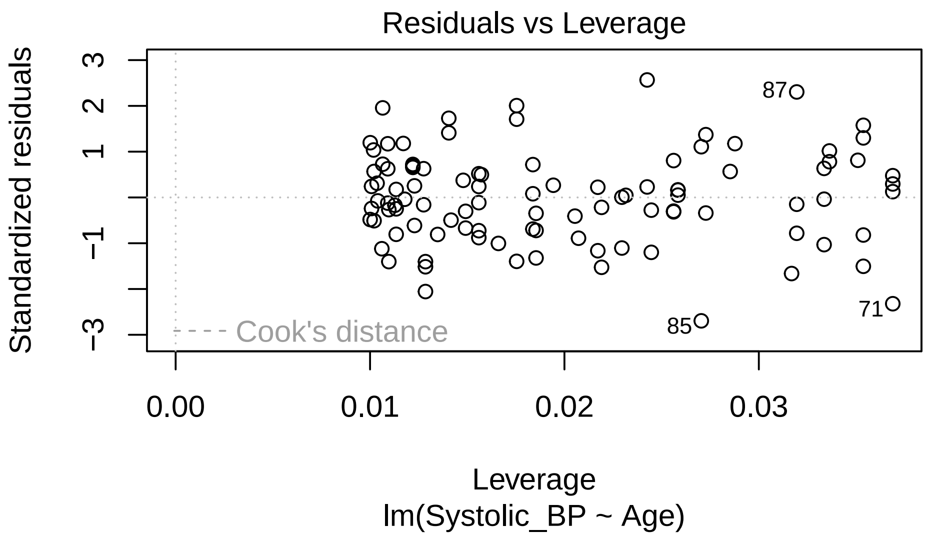


The outlier (red circle) in the middle plot “pulls” the regression line in its direction and has large influence on the slope. THe outlier (red circle) in the right plot has less influence on the slope because it is closer to the mean of x.

Leverage plot (Hebelarm-Diagramm)

In the leverage plot, (standardized) residuals $\tilde{R}_i$ are plotted against the leverage H_{ii} :

```
plot(bp_age_model, which=5, add.smooth = FALSE)
```



Critical ranges are the top and bottom right corners!!

Here, observations 71, 85, and 87 are labelled as potential outliers.

Some texts will give a rule of thumb that points with Cook's distances greater than 1 should be considered influential, while others claim a reasonable rule of thumb is $4/(n-p-1)$ where n is the sample size, and p is the number of *beta* parameters.

Think–Pair–Share (#a_linear_model_assumptions) Which assumption of linear regression feels most fragile in real biological data? Why?

What can go "wrong" during the modeling process?

Answer: a lot of things!

- Non-linearity. We assumed a linear relationship between the response and the explanatory variables. But this is not always the case in practice. We might find that the relationship is curved and not well representented by a straight line.
- Non-normal distribution of residuals. The QQ-plot data might deviate from the straight line so much that we get worried!
- Heteroscadisticity (non-constant variance). We assumed homoscadisticity, but the residuals might show a pattern.
- Data point with high influence. We might have a data point that has a large influence on the slope of the line of best fit.

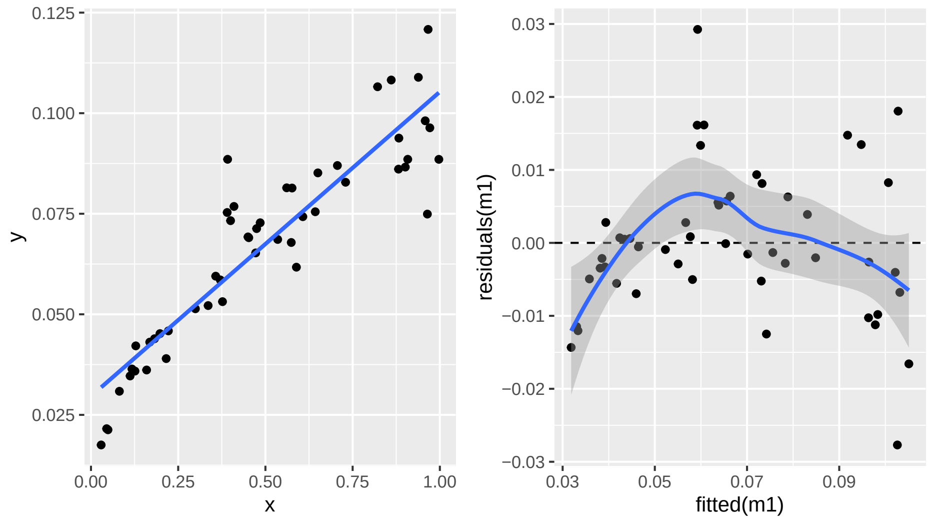
What to do when things "go wrong"?

1. Now: Transform the response and/or explanatory variables.
2. Now: Take care of outliers.
3. Later in the course: Improve the model, e.g., by adding additional terms or interactions.
4. Later in the course: Use another model family (generalized or nonlinear regression model).

Dealing with non-linearity

Here's another example of y and x that are not linearly related:

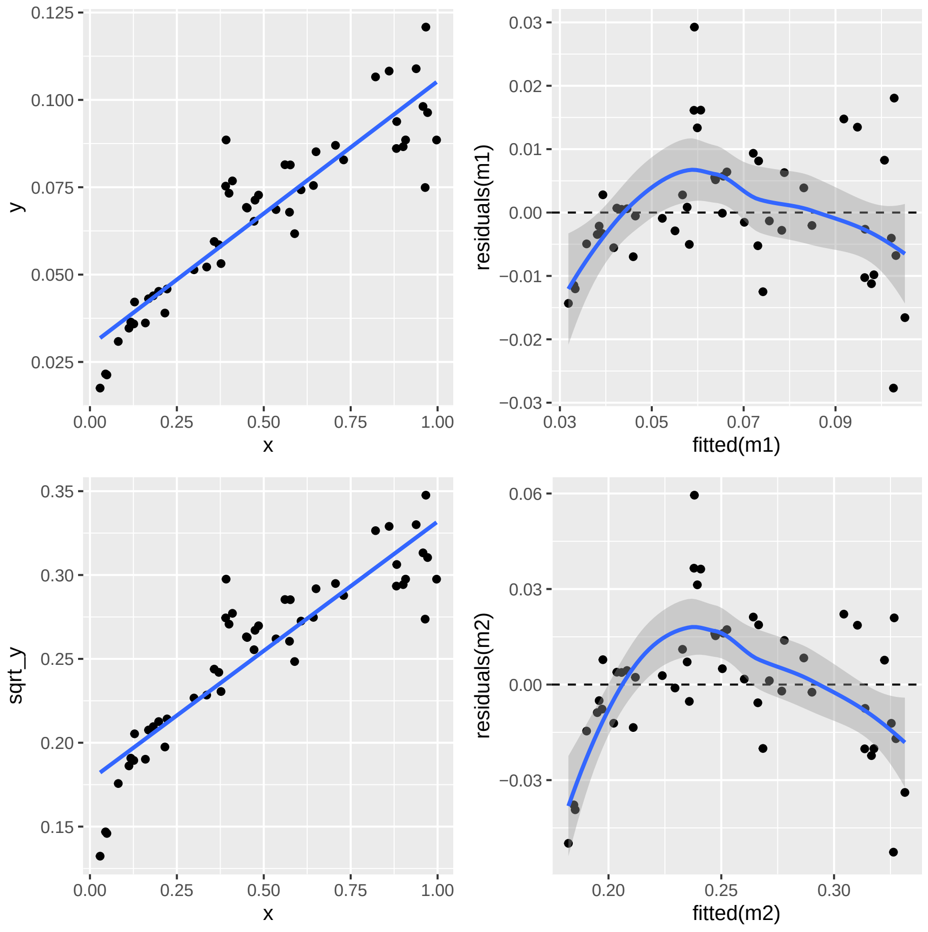
```r
eg_data <- tibble(x = runif(50)) %>%
  mutate(log_y = log(0.1) + 0.5* log(x) + rnorm(50,0,0.1),
         y = exp(log_y),
         log_y = log(y),
         log_x = log(x),
         sqrt_y = sqrt(y))

p1 <- ggplot(eg_data, aes(x = x, y = y)) +
  geom_point() +
  geom_smooth(formula = y ~ x, method = "lm", se = FALSE)

m1 <- lm(y ~ x, eg_data)

p2 <- ggplot(mapping = aes(x = fitted(m1), y = residuals(m1))) +
  geom_point() +
  geom_hline(yintercept = 0, linetype = "dashed") +
  geom_smooth(formula = y ~ x, method = "loess")

p1 + p2
```



One way to deal with this is to transform the response variable Y. Here we try two different transformations: $\log_{10}(Y)$ and $\sqrt{Y}$.

Square root transform of the response variable Y:

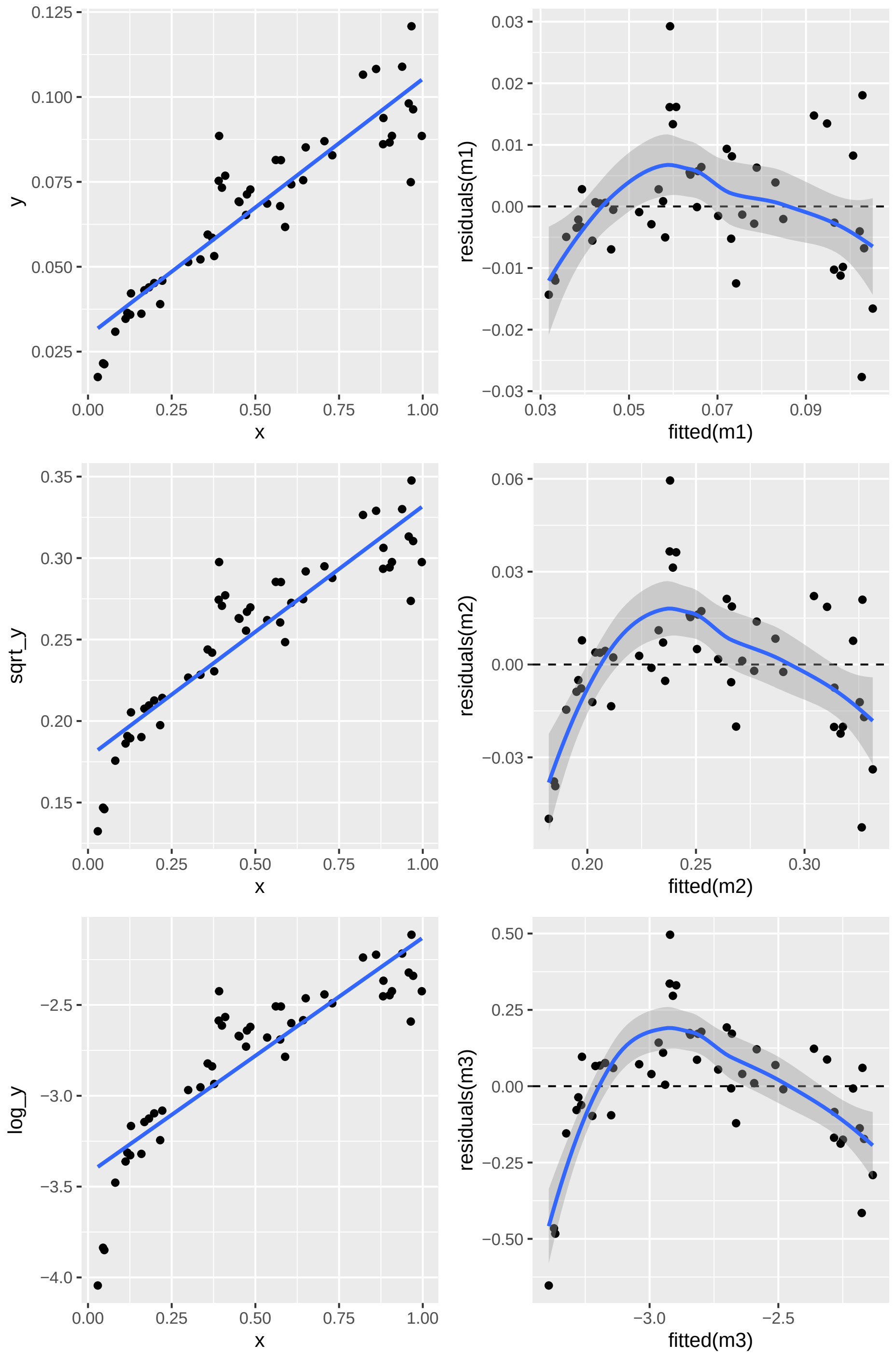
```
p3 <- ggplot(eg_data, aes(x = x, y = sqrt_y)) +
  geom_point() +
  geom_smooth(formula = y ~ x, method = "lm", se = FALSE)
m2 <- lm(sqrt_y ~ x, eg_data)
p4 <- ggplot(mapping = aes(x = fitted(m2), y = residuals(m2))) +
  geom_point() +
  geom_hline(yintercept = 0, linetype = "dashed") +
  geom_smooth(formula = y ~ x, method = "loess")

(p1 + p2) / (p3 + p4)
```



Not great.

Log transformation of the response variable Y:

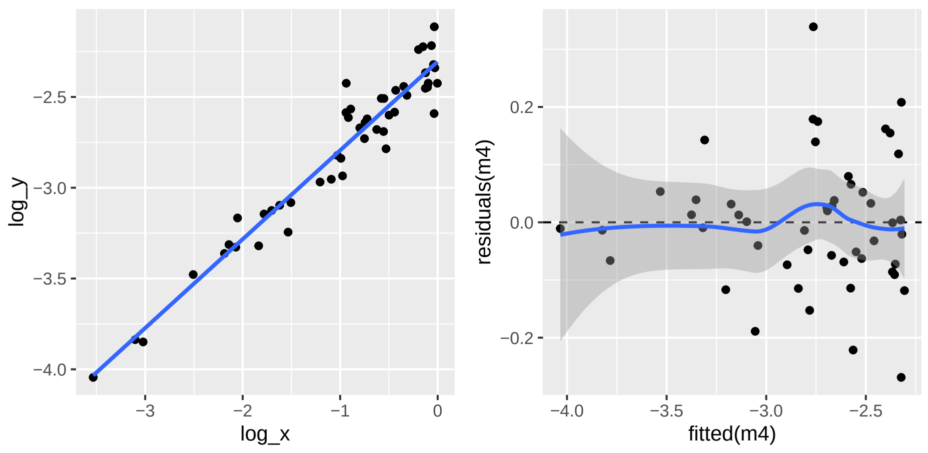
```
p5 <- ggplot(eg_data, aes(x = x, y = log_y)) +
  geom_point() +
  geom_smooth(formula = y ~ x, method = "lm", se = FALSE)
m3 <- lm(log_y ~ x, eg_data)
p6 <- ggplot(mapping = aes(x = fitted(m3), y = residuals(m3))) +
  geom_point() +
  geom_hline(yintercept = 0, linetype = "dashed") +
  geom_smooth(formula = y ~ x, method = "loess")

(p1 + p2) / (p3 + p4) / (p5 + p6)
```



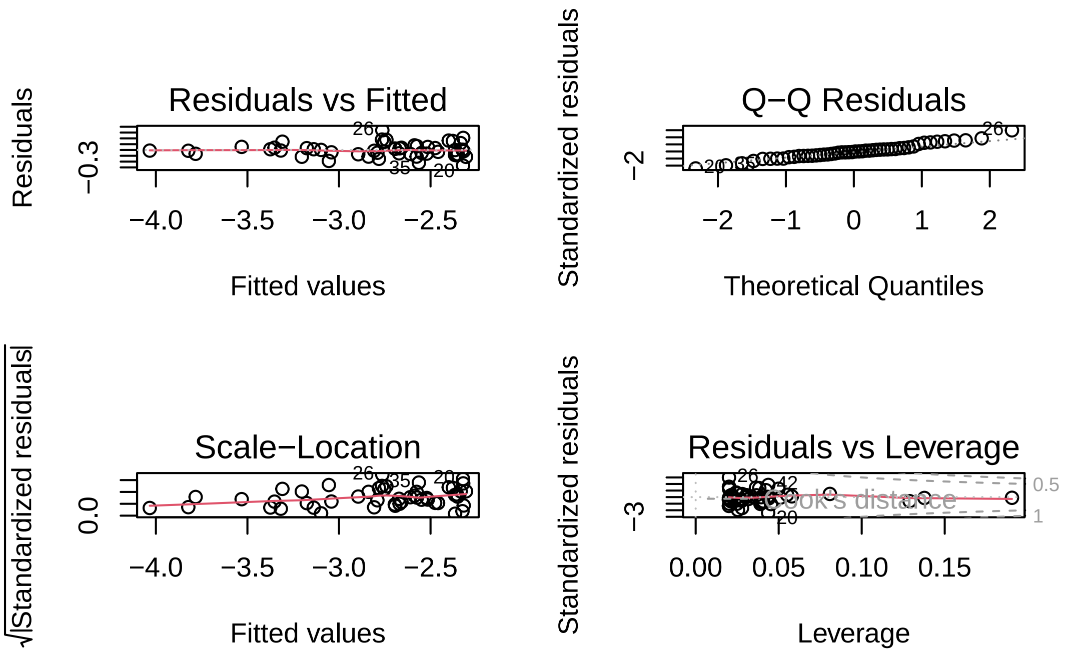
Nope. Still some evidence of non-linearity.

What about transforming the explanatory variable X as well?

```r
p7 <- ggplot(eg_data, aes(x = log_x, y = log_y)) +
  geom_point() +
  geom_smooth(formula = y ~ x, method = "lm", se = FALSE)
m4 <- lm(log_y ~ log_x, eg_data)
p8 <- ggplot(mapping = aes(x = fitted(m4), y = residuals(m4))) +
  geom_point() +
  geom_hline(yintercept = 0, linetype = "dashed") +
  geom_smooth(formula = y ~ x, method = "loess")

p7 + p8
```



Let's look at the four model checking plots for the log-log-transformed data:



All looks pretty good except for the scale-location plot, which shows a bit of a pattern. But overall, this looks much better than our original model.

But… how to know which transformation to use…? It’s a bit of trial and error. But we can use the model checking plots to help us.

Very very important is that we do this trial and error before we start using the model. E.g., we don’t want to jump from the aeroplane and then find out that our parachute is not working properly! And then try to fix the parachute while we are falling….

Likewise, we must not start using the model and then try to fix it. We need to make sure our model is in good working order before we start using it.

One of the traps we could fall into is called “p-hacking”. This is when we try different transformation until we find one that gives us the **result we want**, for example significant relationship. This is a big no-no in statistics. We need to decide on the model (including any transformations) before we start using it.

Common transformations

Which transformations could be considered? There is no simple answer. But some guidelines. E.g. if we see non-linearity and increasing variance with increasing fitted values, then a log transform may improve matter.

Some common and useful transformations are:

- The log transformation for concentrations and absolute values.

- The square-root transformation for count data.
- The arcsin square-root $\arcsin(\sqrt{\cdot})$ transformation for proportions/percentages.

Transformations can also be applied on explanatory variables, as we saw in the example above.

Outliers

What do we do when we identify the presence of one or more outliers?

1. Start by checking the "correctness" of the data. Is there a typo or a decimal point that was shifted by mistake? Check both the response and explanatory variables.
2. If not, ask whether the model could be improved. Do reasonable transformations of the response and/or explanatory variables eliminate the outlier? Do the residuals have a distribution with a long tail (which makes it more likely that extreme observations occur)?
3. Sometimes, an outlier may be the most interesting observation in a dataset! Was the outlier created by some interesting but different process from the other data points?
4. Consider that outliers can also occur just by chance!
5. Only if you decide to report the results of both scenario can you check if inclusion/exclusion changes the qualitative conclusion, and by how much it changes the quantitative conclusion.

Removing outliers

It might seem tempting to remove observations that apparently don't fit into the picture. However:

- Do this **only with greatest care** e.g., if an observation has extremely implausible values!

- Before deleting outliers, check points 1-5 above.
- When removing outliers, **you must mention this in your report**.

During the course we'll see many more examples of things going at least a bit wrong. And we'll do our best to improve the model, so we can be confident in it, and start to use it. Which we will start to do in the next lesson. But before we wrap up, some good news…

Its a kind of magic…

Above, we learned about linear regression, the equation for it, how to estimate the coefficients, and how to check the assumptions. There was a lot of information, and it might seem a bit overwhelming.

You might also be aware that there are quite a few other types of statistical model, such as multiple regression, t-test, ANOVA, two-way ANOVA, and ANCOVA. It could be worrying to think that you need to learn so much new information for each of these types of tests.

But this is where the kind-of-magic happens. The good news is that the linear regression model is a special case of what is called a *general linear model*, or just *linear model* for short. And that all the tests mentioned above are also types of *linear model*. So, once you have learned about linear regression, you have learned a lot about linear models, and therefore also a lot about all of these other tests as well.

Moreover, the same function in R 'lm' is used to make all those statistical models Awesome.

So what is a linear model?

A linear model is a model where the relationship between the dependent variable and the independent variables is linear. That is, the dependent variable can be expressed as a linear combination of the independent variables. An example of a linear model is:

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + ... + \beta_p x_p + \epsilon$$

where: y is the dependent variable, β_0 is the intercept, $\beta_1, \beta_2, ..., \beta_p$ are the coefficients of the independent variables, $x_1, x_2, ..., x_p$ are the independent variables, ϵ is the error term.

In contrast, a non-linear model is a model where the relationship between the dependent variable and the independent variables is non-linear. An example of a non-linear model is the exponential growth model:

$$y = \beta_0 + \beta_1 e^{\beta_2 x} + \epsilon$$

where: y is the dependent variable, β_0 is the intercept, β_1, β_2 are the coefficients of the independent variables, x is the independent variable, ϵ is the error term.

Keep in mind that a model with a quadratic term is still a linear model. For example:

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x^2 + \epsilon$$

is still a linear model. We can see this if we substitute x^2 with a new variable x_2, where $x_2 = x^2$. The model then becomes:

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \epsilon$$

This is clearly a linear model.

Review

In this chapter we learned about first steps of regression analysis. Specifically, we learned about to:

- Motivation for regression analysis.
- The simple linear regression model.
- Estimation of the coefficients (intercept and slope) using the least squares method.
- How to model the errors (residuals) and the assumptions of linear regression.
- How to check the assumptions of linear regression using model checking plots.
- What can go wrong during the modeling process, and how to deal with it.

With all that in place, we can be sure that when we use a linear regression model, we can trust the results it gives us. In the next chapter, we will start to use linear regression models to answer questions! This involves things such as hypothesis testing, confidence intervals, and prediction. Exciting times ahead!

Regression Part 2 (L4)

This chapter builds on the previous chapter on simple linear regression. There we learned how to fit a regression model to data, and how to check if the assumptions of the regression model are met. In this chapter we will learn how to interpret the results of a regression model, and how to use the model to make inferences about the relationship between the dependent and independent variables.

Introduction

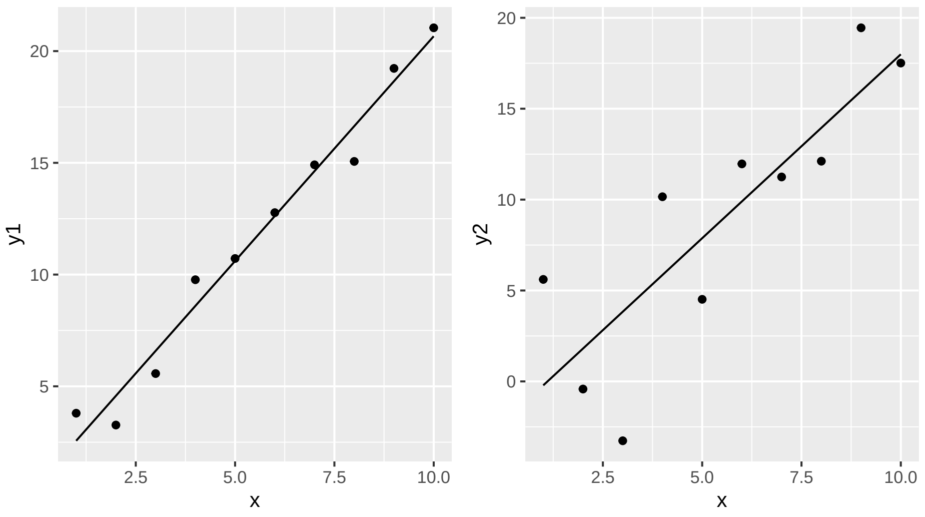
Now that we have a satisfactory model, we can start to use it. In the following material, you will learn:

- How to measure how good is the regression (correlation and R^2).
- How to test if the parameter estimates are compatible with some specific value (t-test).
- How to find the range of parameters values are compatible with the data (confidence intervals).
- How to find the regression lines compatible with the data (confidence band).
- How to calculate plausible values of newly collected data (prediction band).

How good is the regression model?

What would a good regression model look like? What would a bad one look like? One could say that a good regression model is one that explains the dependent variable well. But what could we mean by "explains the data well"?

Take these two examples.



Think-Pair-Share (#tps-better-model) In which of these two would you say the model is better, and in which is it worse?

The first model seems to fit the data well, while the second one does not. But how can we quantify this?

Let's say that we will measure the goodness of the model by the amount of variability of the dependent variable that is explained by the independent variable. To do this we need to do the following:

1. Measure the total variability of the dependent variable (total sum of squares, SST).
2. Measure the amount of variability of the dependent variable that is explained by the independent variable (model sum of squares, SSM).
3. Measure the variability of the dependent variable that is not explained by the independent variable (error sum of squares, SSE).
4. Calculate the proportion of variability of the dependent variable that is explained by the independent variable (R^2, pronounced as "r-squared") (also known as the coefficient of determination) ($R^2 = SSM/SST$).

Importantly, note that we will calculate SSM and SSE so that they sum up to SST. I.e., $SST = SSM + SSE$. That is, the total variability is the sum of what is explained by the model and what remains unexplained.

Let's take each in turn:

SST

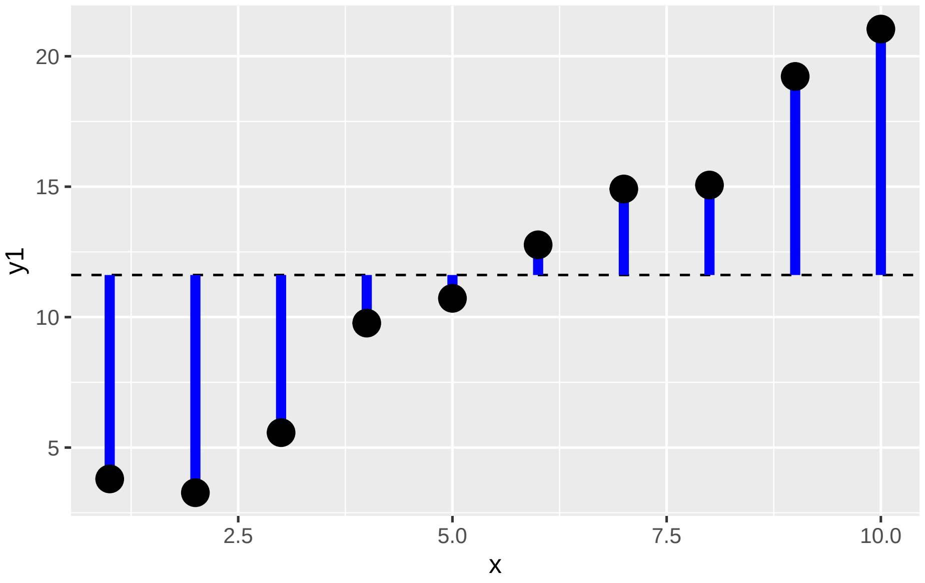
1. The total variability of the dependent variable is the sum of the squared differences between the dependent variable and its mean. This is called the total sum of squares (SST).

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

where: y_i is the dependent variable, $\bar{y}$ is the mean of the dependent variable, n is the number of observations.

Note that sometimes SST is referred to as SSY (sum of squares of y).

Graphically, this is the sum of the square of the blue residuals as shown in the following graph, where the horizontal dashed line is at the value of the mean of the dependent variable.



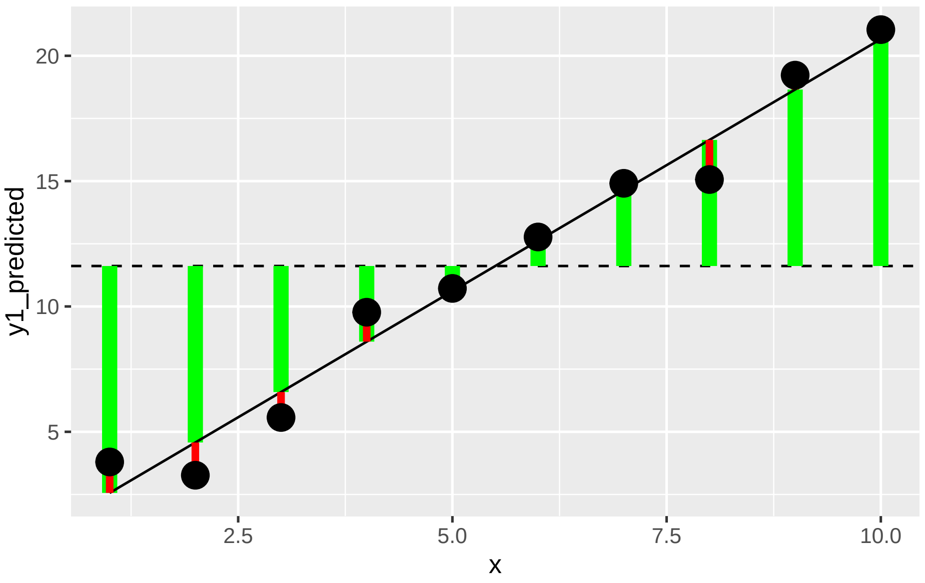
We can calculate this in R as follows:

```
SST <- sum((y1 - mean(y1))^2)
```

SSM and SSE

Now the next two steps, that is getting the model sum of squares (SSM) and the error sum of squares (SSE) are a bit more complicated. To do this we need to fit a regression model to the data. Let's see this graphically, and divide the data into the explained and unexplained parts.

Make a graph with vertical lines connecting the data to the mean of the data, but with each line two parts, one from the mean to the data, and one from the data to the predicted value.



In this graph, the square of the length of the green lines is the model sum of squares (SST). The square of the length of the red lines is the error sum of squares (SSE).

In a better model the length of the green lines will be **longer** (the square of these gives the SMM, the variability explained by the model). And the length of the red lines will be **shorter** (the square of these gives the SSE, the variability not explained by the model).

SSM

Next we will do the second step, that is calculate the model sum of squares (SSM).

2. The amount of variability of the dependent variable that is explained by the independent variable is called the model sum of squares (SSM**).**

This is the difference between the predicted value of the dependent variable and the mean of the dependent variable, squared and summed:

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

where: $\hat{y}_i$ is the predicted value of the dependent variable,

In R, we calculate this as follows:

```r
m1 <- lm(y1 ~ x)
y1_predicted <- predict(m1)
SSM <- sum((y1_predicted - mean(y1))^2)
SSM
```

```
[1] 333.762
```

SSE

Third, we calculate the error sum of squares (SSE) with either of two methods. We could calculate it as the sum of the squared residuals, or as the difference between the total sum of squares and the model sum of squares:

$$SSE = \sum_{i=1}^{n}(y_i - \hat{y}_i)^2 = SST - SSM$$

Let's calculate this in R uses both approaches:

```r
SSE <- sum((y1 - y1_predicted)^2)
SSE
```

```
[1] 8.713496
```

Or…

```r
SSE <- SST - SSM
SSE
```

```
[1] 8.713496
```

R^2

Finally, we calculate the proportion of variability of the dependent variable that is explained by the independent variable (R^2):

$$R^2 = \frac{SSM}{SST}$$

```r
R.squared <- SSM/SST
R.squared
```

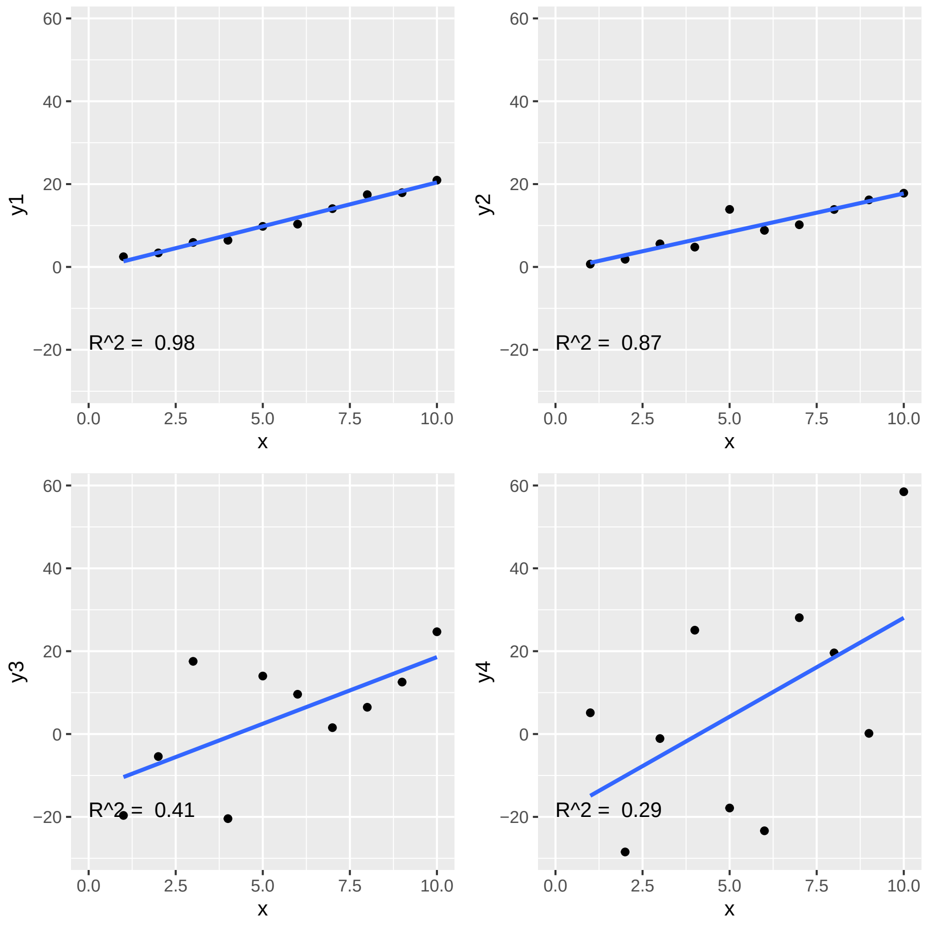
```
[1] 0.9745573
```

Is my R squared good?

What value of R^2 is considered good? In ecological research, R^2 values are often low (less than 0.3), because ecological systems are complex and many factors influence the dependent variable. However, in other fields, such as physiology,

R^2 values are often higher. Therefore, the answer of what values of R^2 are good depends on the field of research.

Here are the four examples and their r-squared.



Think-Pair-Share (#tps-what-minimised) What is minimised when we fit a regression model? And therefore what is maximised?

How unlikey is the observed data given the null hypothesis?

We often hear this expressed as "is the relationship significant?" And maybe we heard that the relationship is significant if the p-value is less than 0.05. But what does all this actually mean? In this section we'll figure all this out. The first step to is to formulate a null hypothesis.

What is a meaningful null hypothesis for a regression model?

As mentioned, often we're interested in whether there is a relationship between the dependent (response) and independent (explanatory) variable. Therefore, the null hypothesis is that there is no relationship between the dependent and independent variable. This means that the null hypothesis is that the slope of the regression line is zero.

Recall the regression model:

$$y = \beta_0 + \beta_1 x + \epsilon$$

Think-Pair-Share (#tps-null-hypothesis) Write down the null hypothesis of no relationship between x and y in terms of a β parameter.

The null hypothesis is that the slope of the regression line is zero:

$$H_0 : \beta_1 = 0$$

What is the alternative hypothesis?

$$H_1 : \beta_1 \neq 0$$

So, how do we test the null hypothesis? More precisely, we are going to calculate the probability of observing the data we have, given that the null hypothesis is true. If this probability is very low, then we can reject the null hypothesis.

Does that make sense? Does it seem a bit convoluted? It is a bit!!!

But this is how hypothesis testing works. We never prove the null hypothesis is true. Instead, we calculate the probability of observing our data given that the null hypothesis is true. If this probability is very low, we reject the null hypothesis.

To make the calculation we can use the fact that the slope of the regression line is an estimate of the true slope. This estimate has uncertainty associated with it. We can use this uncertainty to calculate the probability of observing the data we have, given the null hypothesis is true.

We can see that the slope estimate (the x row) has uncertainty by looking at the regression output:

```r
summary(m1)$coefficients[1:2, 1:2]
```

```
              Estimate Std. Error
(Intercept) -0.7638353  0.6652233
x            2.1161160  0.1072104
```

The estimate is the mean of the distribution of the parameter (slope) and the standard error is a measure of the uncertainty of the estimate.

The standard error is calculated as:

$$\sigma^{(\beta_1)} = \sqrt{\frac{\hat{\sigma}^2}{\sum_{i=1}^{n}(x_i - \bar{x})^2}}$$

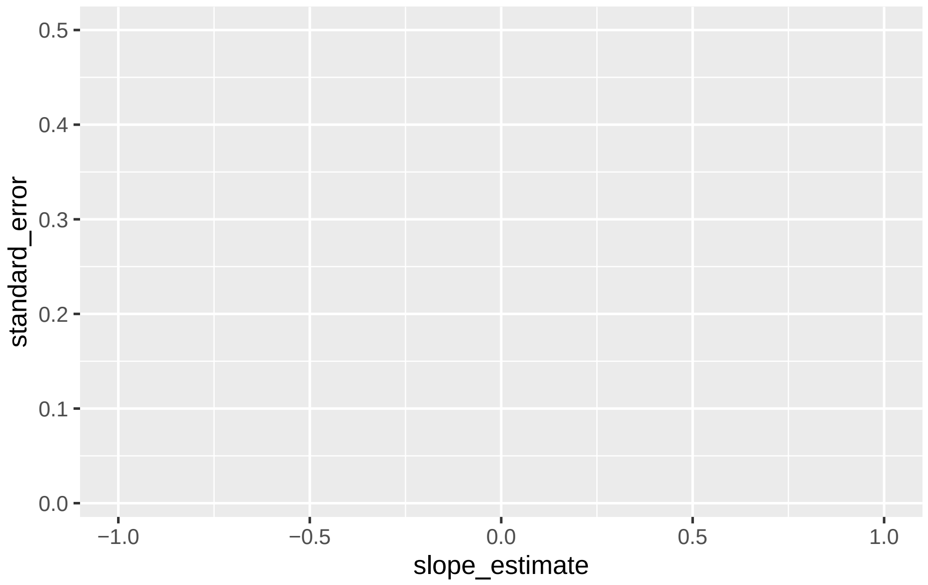
Where $\hat{\sigma}^2$ is the expected residual variance of the model. This is calculated as:

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

Where $\hat{y}_i$ is the predicted value of y_i from the regression model.

OK, let's take a look at this intuitively. We have the estimate of the slope and the standard error of the estimate.

Here is a graph of the value of the slope estimate versus the standard error of the estimate:



Think-Pair-Share (#tps-chance-area) In what areas of the graph is the slope estimate more likely to have been observed by chance? And what regions is it less likely to have been observed by chance? Think about this before you look at the end of this chapter for an answer.

When the slope estimate is larger, it is less likely to have been observed by chance. And when the standard error is larger, it is more likely to have been observed by chance. How can we put these together into a single measure?

If we divide the slope estimate by the standard error, we get a measure of how many standard errors the slope estimate is from the null hypothesis slope of zero. This is the t-statistic:

$$t = \frac{\hat{\beta}_1 - \beta_{1,H_0}}{\sigma^{(\beta_1)}}$$

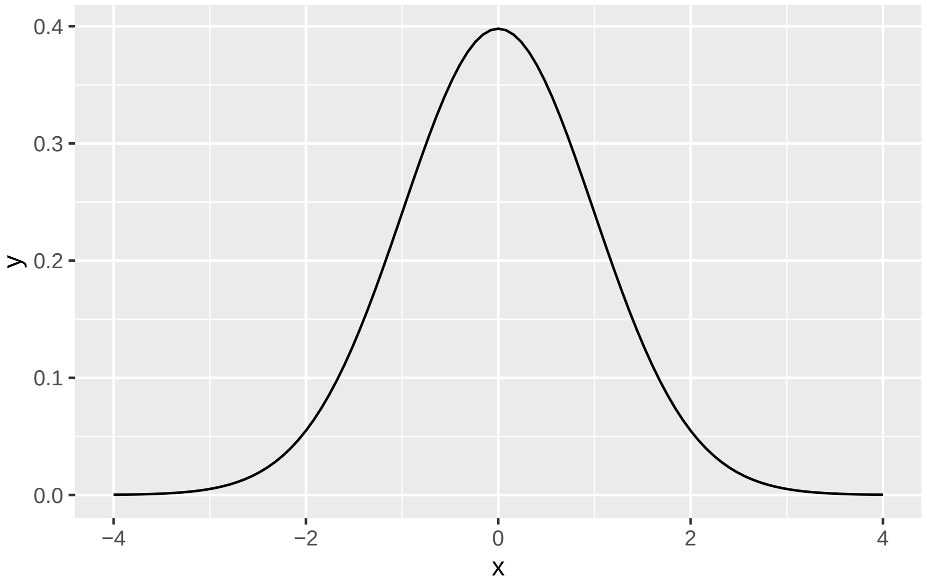
Where β_{1,H_0} is the null hypothesis value of the slope, usually zero, so that

$$t = \frac{\hat{\beta}_1}{\sigma^{(\beta_1)}}$$

The t-statistic is a measure of how many standard errors the slope estimate is from the null hypothesis value of the slope. The larger the t-statistic, the less likely the slope estimate was observed by chance.

How can we transform the value of a t-statistic into a p-value? We can use the ***t*-distribution**, which quantifies the probability of observing a value of the t-statistic under the null hypothesis.

But what is the t-distribution? It is a distribution of the t-statistic under the null hypothesis. It is a bell-shaped distribution that is centered on zero. The shape of the distribution is determined by the degrees of freedom, which is $n-2$ for a simple linear regression model.

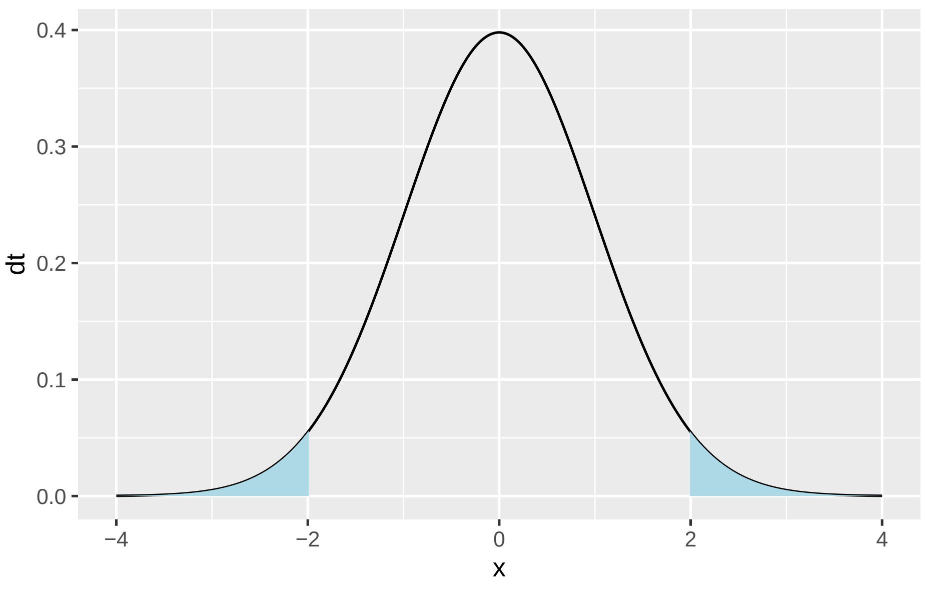


Tip

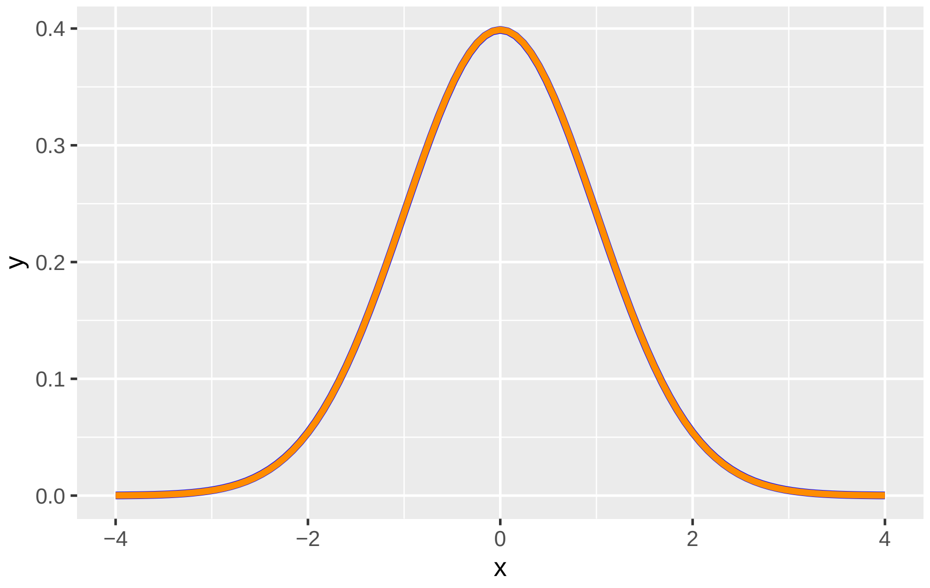
By the way, it is named the t-distribution by it's developer, William Sealy Gosset, who worked for the Guinness brewery in Dublin, Ireland. In his 1908 paper, Gosset introduced the t-distribution but he didn't explicitly explain his choice of the letter t. The choice of the letter t could be to indicate "Test", as the t-distribution was developed specifically for hypothesis testing.

Now, recall that the p-value is the probability of observing the value of the test statistic (so here the t-statistic) at least as extreme as the one we have, given the null hypothesis is true. We can calculate this probability by integrating the t-distribution from the observed t-statistic to the tails of the distribution.

Here is a graph of the t-distribution with 100 degrees of freedom with the tails of the distribution shaded so that the area of the shaded region is 0.05 (i.e., 5% of the total area).



And here's a graph of the t-distribution with 1000 degrees of freedom (blue line) and the normal distribution (green dashed line):



So, with a large number of observations, the *t*-distribution approaches the normal distribution. For the normal distribution, the 95% area is between -1.96 and 1.96.

```
## x value for 95% area of normal distribution
x_value <- qnorm(0.975)
x_value
```

```
[1] 1.959964
```

`qnorm` is a function that calculates the x value for a given quantile (probability) of the normal distribution. In simpler terms, it finds the value x at which the area under the normal curve (up to x) equals the given probability p (0.975 in the example immediately above here).

Let's go back to the age - blood pressure data and calculate the p-value for the slope estimate.

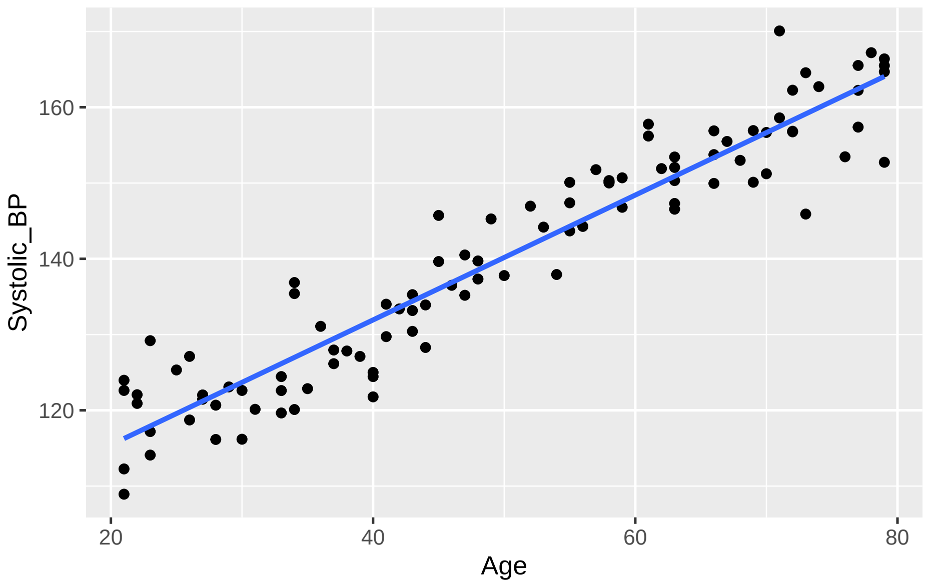
Read in the data:

```
bp_data <- read_csv(here::here("datasets/Simulated_Blood_Pressure_and_Age_Data.csv"))
```

```
Rows: 100 Columns: 2
-- Column specification --------------------------------------------------------
Delimiter: ","
dbl (2): Age, Systolic_BP

i Use `spec()` to retrieve the full column specification for this data.
i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

Make a graph:

```
ggplot(bp_data, aes(x = Age, y = Systolic_BP)) +
  geom_point() +
  geom_smooth(method = "lm", se = FALSE, formula = y ~ x)
```



Here's the model:

```
mod1 <- lm(Systolic_BP ~ Age, data = bp_data)
```

Here we calculate the *t*-statistic for the slope estimate:

```
t_stat <- mod1$coefficients[2] / summary(mod1)$coefficients[2, 2]
```

And here we calculate the one-tailed and two-tailed *p*-values:

```
one_tailed_p_value <- pt(-abs(t_stat), df = nrow(bp_data) - 2)
two_tailed_p_value <- 2 * one_tailed_p_value
one_tailed_p_value
```

```
         Age
3.746958e-51
```

```
two_tailed_p_value
```

```
         Age
7.493917e-51
```

We can get the *p*-value directly from the `summary` function:

```r
summary(mod1)$coefficients[2, ]
```

```
    Estimate   Std. Error      t value     Pr(>|t|)
8.240678e-01 2.770955e-02 2.973948e+01 7.493917e-51
```

Conclusion: there is **very strong evidence** that the blood pressure is associated with age, because the p-value is extremely small (thus it is very unlikely that the observed slope value or a large one would be seen if there was really no association). Thus, we can reject the null hypothesis that the slope is zero.

This basically answers question 1: "Are the parameters compatible with some specific value?"

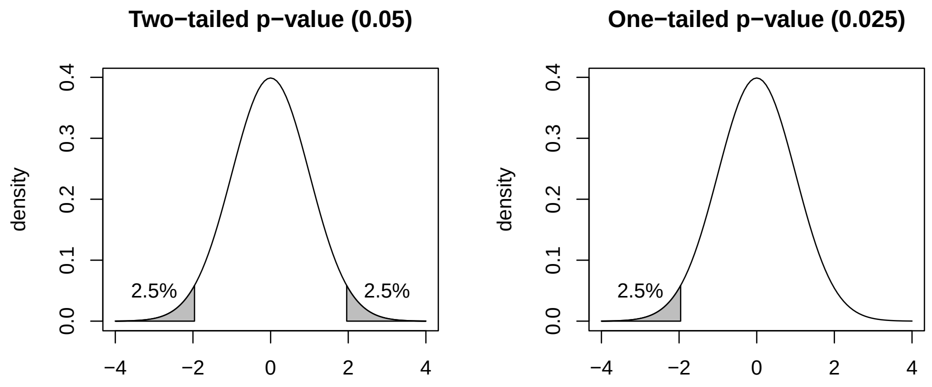
Recap: Formal definition of the p-value

The formal definition of p-value is the probability to observe a data summary (e.g., an average or a slope) that is at least as extreme as the one observed, given that the null hypothesis is correct.

Example (normal distribution): Assume that we calculated that t-value = -1.96

$\Rightarrow Pr(|t| \geq 1.96) = 0.05$ (two-tailed) and $Pr(t \leq -1.96) = 0.025$ (one-tailed).

And here is a graph showing this:



A cautionary note on the use of p-values

Maybe you have seen that in statistical testing, often the criterion $p \leq 0.05$ is used to test whether H_0 should be rejected. This is often done in a black-or-white manner. However, we will put a lot of attention to a more reasonable and cautionary interpretation of p-values in this course!

How strong is the relationship?

The actual value of the slope has practical meaning. The slope of the regression line tells us how much the dependent variable changes when the independent variable changes by one unit. The slope is one measure of the strength of the relationship between the two variables.

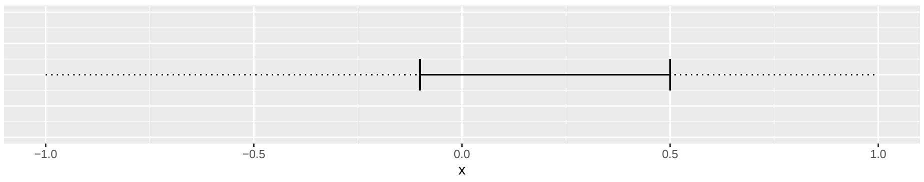
We can ask what values of a parameter estimate are compatible with the data (confidence intervals)? To answer this question, we can determine the confidence intervals of the regression parameters.

The confidence interval of a parameter estimate is defined as the interval that contains the true parameter value with a certain probability. So the 95% confidence interval of the slope is the interval that contains the true slope with a probability of 95%.

We can then imagine two cases. The 95% confidence interval of the slope includes 0:

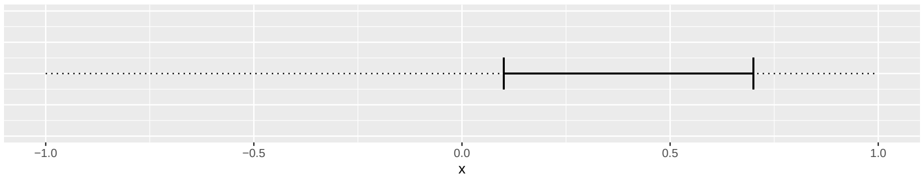
```
Warning: `geom_errobarh()` was deprecated in ggplot2 4.0.0.
i Please use the `orientation` argument of `geom_errorbar()` instead.
```

```
`height` was translated to `width`.
```



Or where the confidence interval does not include zero:

```
`height` was translated to `width`.
```



How do we calculate the lower and upper limits of the 95% confidence interval of the slope?

Recall that the t-value for a null hypothesis of slope of zero is defined as:

$$t = \frac{\hat{\beta}_1}{\hat{\sigma}^{(\beta_1)}}$$

The first step is to calculate the t-value that corresponds to a p-value of 0.05. This is the t-value that corresponds to the 97.5% quantile of the t-distribution with $n-2$ degrees of freedom.

$t_{0.975} = t_{0.025} = 1.96$, for large n.

The 95% confidence interval of the slope is then given by:

$$\hat{\beta}_1 \pm t_{0.975} \cdot \hat{\sigma}^{(\beta_1)}$$

In our blood pressure example the estimated slope is 0.8240678 and the standard error of the slope is 0.0277096. We can calculate the 95% confidence interval of the slope in *R* as follows:

```
n <- 100
t_0975 <- qt(0.975, df = n - 2)
half_interval <- t_0975 * summary(mod1)$coef[2,2]
lower_limit <- coef(mod1)[2] - half_interval
upper_limit <- coef(mod1)[2] + half_interval
ci_slope <- c(lower_limit, upper_limit)
slope <- coef(mod1)[2]
slope
```

```
      Age
0.8240678
```

```
ci_slope
```

```
      Age       Age
0.7690791 0.8790565
```

Or, using the `confint` function:

```
## 95% confidence interval of the slope of mod1
ci_slope_2 <- confint(mod1, level=c(0.95))[2,]
ci_slope_2
```

```
    2.5 %    97.5 %
0.7690791 0.8790565
```

Or we can do it using values from the coefficients table:

```
coefs <- summary(mod1)$coef
beta <- coefs[2,1]
sdbeta <- coefs[2,2]
beta + c(-1,1) * qt(0.975,241) * sdbeta
```

```
[1] 0.7694840 0.8786516
```

Interpretation: for an increase in the age by one year, roughly 0.82 mmHg increase in blood pressure is expected, and all true values for β_1 between 0.77 and 0.88 are compatible with the observed data.

Confidence and Prediction Bands

- Remember: If another sample from the same population was taken, the regression line would look slightly different.

- There are two questions to be asked:

1. Which other regression lines are compatible with the observed data? This leads to the *confidence band.*

2. Where do future observations (y) with a given x coordinate lie? This leads to the *prediction band.*

Note: The prediction band is much broader than the confidence band.

Calculation of the confidence band

Given a fixed value of x, say x_0. The question is:

Where does $\hat{y}_0 = \hat{\beta}_0 + \hat{\beta}_1 x_0$ lie with a certain confidence (i.e., 95%)?

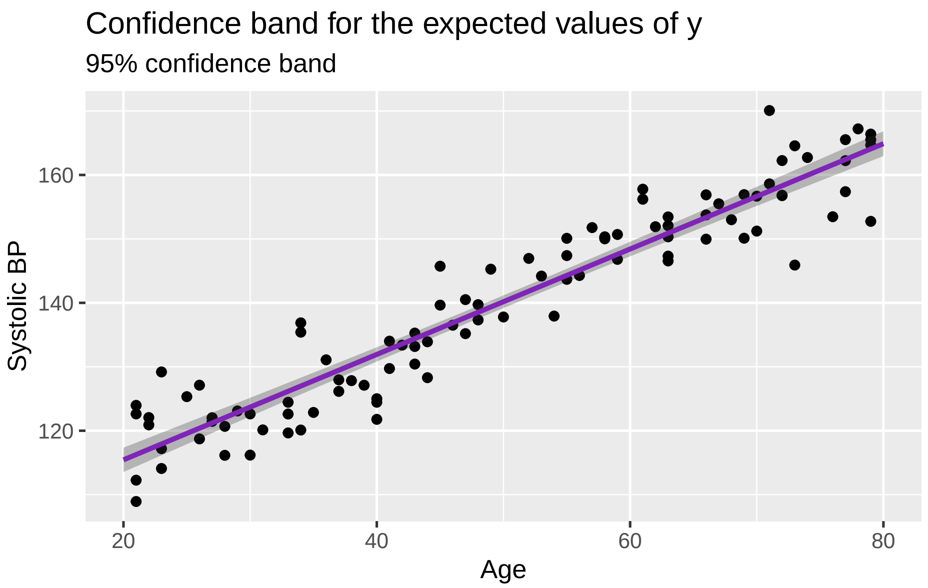
This question is not trivial, because both $\hat{\beta}_0$ and $\hat{\beta}_1$ are estimates from the data and contain uncertainty.

The details of the calculation are given in Stahel 2.4b.

Plotting the confidence interval around all $\hat{y}_0$ values one obtains the *confidence band* or *confidence band for the expected values* of y.

Note: For the confidence band, only the uncertainty in the estimates $\hat{\beta}_0$ and $\hat{\beta}_1$ matters.

Here is the confidence band for the blood pressure data:



Very narrow confidence bands indicate that the estimates are very precise. In this case the estimated intercept and slope are precise because the sample size is large and the data points are close to the regression line.

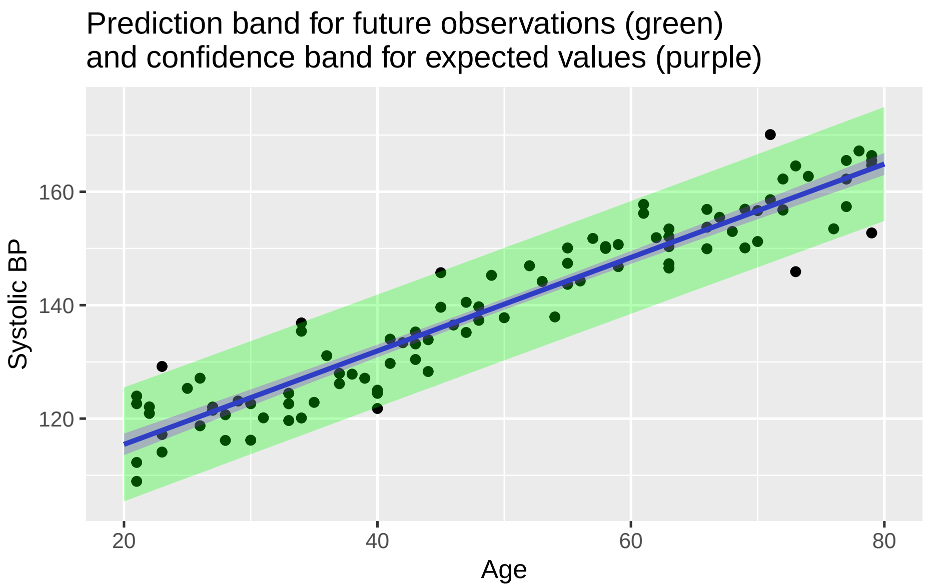
Calculations of the prediction band

We can easily predicted an expected value of y for a given x value. But we can also ask w where does a *future observation* lie with a certain confidence (i.e., 95%)?

To answer this question, we have to *consider not only the uncertainty in the predicted value caused by uncertainty in the parameter estimates* $\hat{y}_0 = \hat{\beta}_0 + \hat{\beta}_1 x_0$, but also the *error term* $\epsilon_i \sim N(0, \sigma^2)\}$.

This is the reason why the **prediction band** is wider than the confidence band.

Here's a graph showing the prediction band for the blood pressure data:



Another way to think of the 95% confidence band is that it is where we would expect 95% of the regression lines to lie if we were to collect many samples from the same population. The 95% prediction band is where we would expect 95% of the future observations to lie.

Think–Pair–Share (#a_prediction_vs_estimation) Which interval answers: "What is the mean response?" Which answers: "What might a new observation look like?"

Review

That is regression done (at least for our current purposes). Here is a summary of what we have covered:

Previous chapter:

- Why use (linear) regression?
- Fitting the line (= parameter estimation)
- Is linear regression good enough model to use?
- What to do when things go wrong?
- Transformation of variables/the response.
- Handling of outliers.

This chapter:

- Sums of squares: SST, SSM, SSE
- R^2 as a measure of goodness of fit
- Hypothesis testing in regression
- Null and alternative hypotheses

- t-statistic and p-values
- Tests and confidence intervals
- Confidence and prediction bands

Extras

Randomisation test for the slope of a regression line

Let's use randomisation as another method to understand how likely we are to observe the data we have, given the null hypothesis is true.

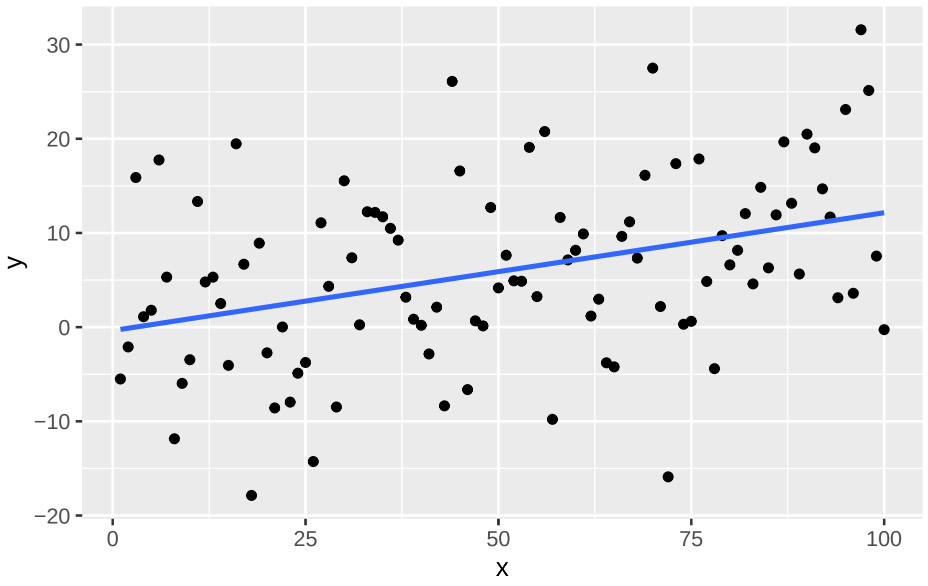
If the null hypothesis is true, we expect no relationship between x and y. Therefore, we can shuffle the y values and fit a regression model to the shuffled data. We can repeat this many times and calculate the slope of the regression line each time. This will give us a distribution of slopes we would expect to observe if the null hypothesis is true.

First, we'll make some data and get the slope of the regression line. Here is the observed slope and relationship:

```
set.seed(123)
n <- 100
x <- 1:n
y <- 0.1*x + rnorm(n, 0, 10)
m <- lm(y ~ x)
coef(m)[2]
```

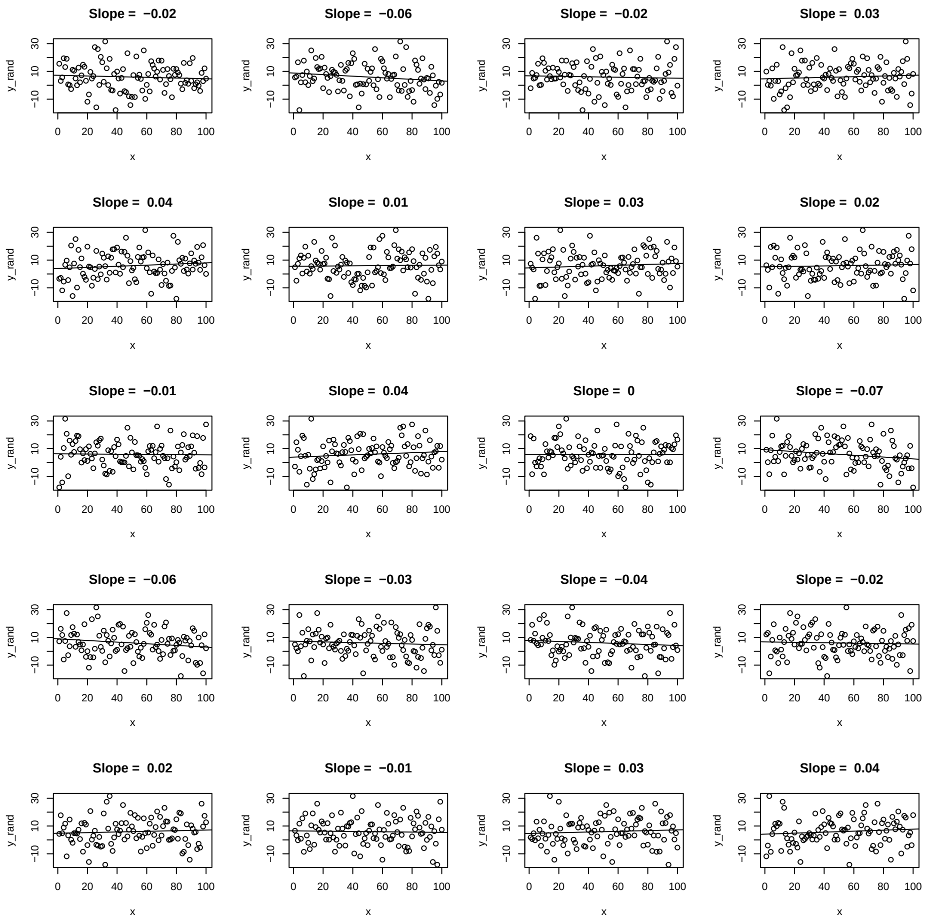
```
        x
0.1251108
```

```
ggplot(data.frame(x = x, y = y), aes(x = x, y = y)) +
  geom_point() +
  geom_smooth(method = "lm", se = FALSE, formula = y ~ x)
```

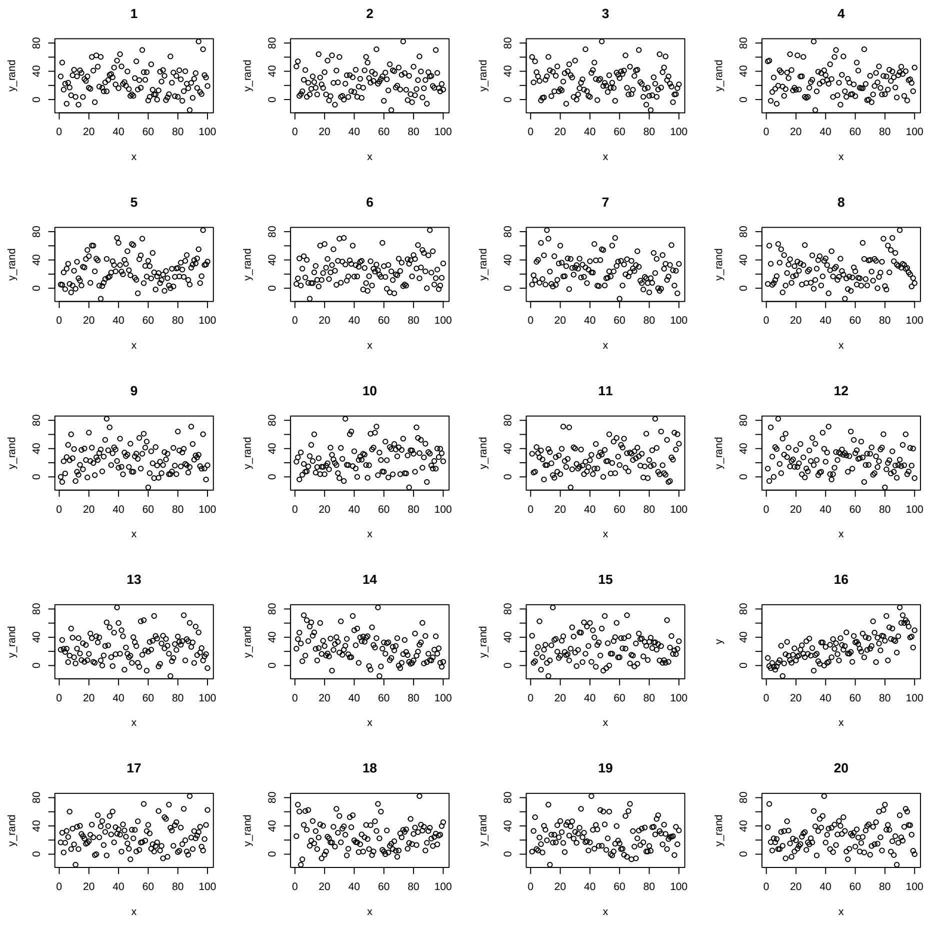


Now we'll use randomisation to test the null hypothesis. We can create lots of examples where the relationship is expected to have a slope of zero by shuffling randomly the y values. Here are 20:

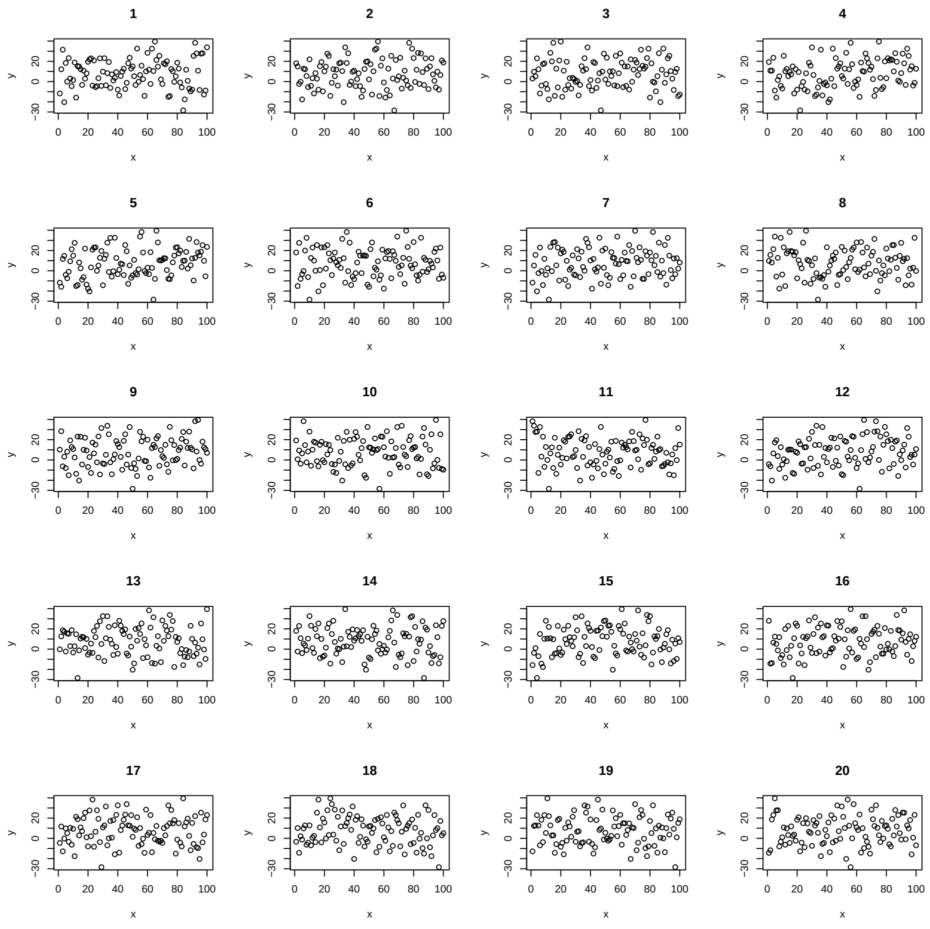
```r
par(mfrow = c(5, 4))
for (i in 1:20) {
  y_rand <- sample(y)
  m_rand <- lm(y_rand ~ x)
  plot(x, y_rand, main = paste("Slope = ", round(coef(m_rand)[2], 2)))
  abline(m_rand)
}
```



Now let's create 19 and put the real one in there somewhere random. Here's a case where the real data has a quite strong relationship:



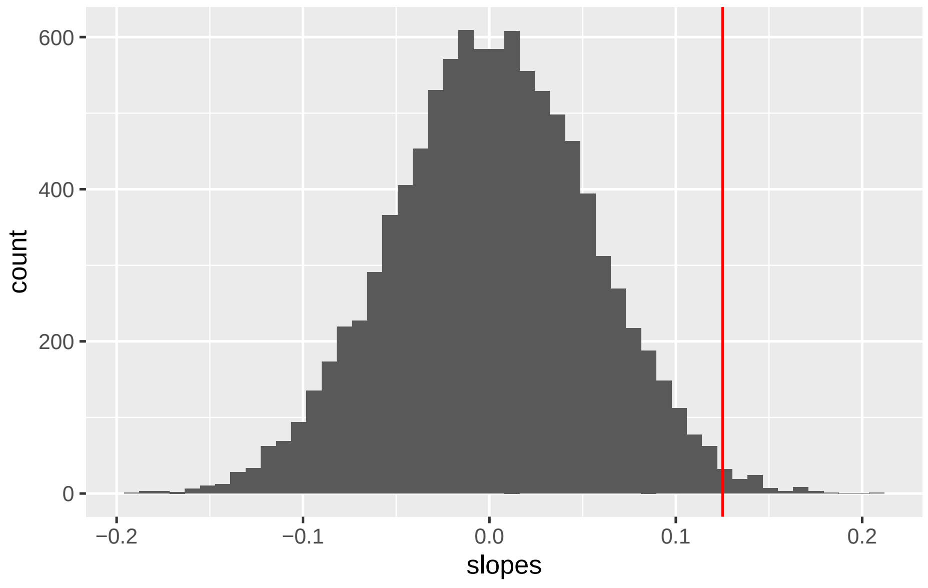
We can confidently find the real data amount the shuffled data. But what if the relationship is weaker?



Now its less clear which is the real data. We can use this idea to test the null hypothesis.

We do the same procedure of but instead of just looking at the graphs, we calculate the slope of the regression line each time. This gives us a distribution of slopes we would expect to observe if the null hypothesis is true. We can then see where the observed slope lies in this distribution of null hypothesis slopes.

```
# repeat 10000 time a randomisation test
y <- 0.15*x + rnorm(n, 0, 15)
rand_slopes <- replicate(10000, {
  y_rand <- sample(y)
  m_rand <- lm(y_rand ~ x)
  coef(m_rand)[2]
})

ggplot(data.frame(slopes = rand_slopes), aes(x = slopes)) +
  geom_histogram(bins = 50) +
  geom_vline(xintercept = coef(m)[2], color = "red")
```

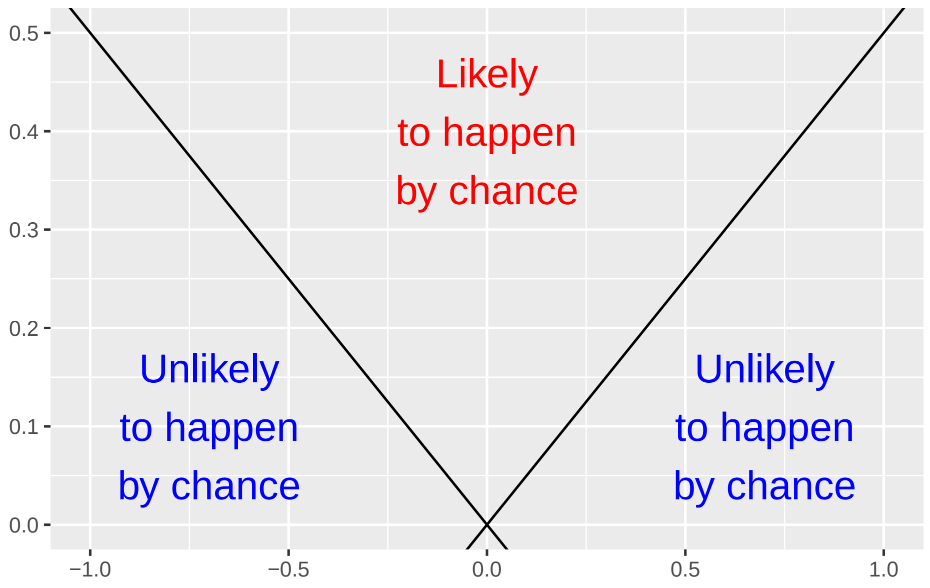


We can now calculate the probability of observing the data we have, given the null hypothesis is true.

```
p_value <- mean(abs(rand_slopes) >= abs(coef(m)[2]))
p_value
```

```
[1] 0.0172
```

Visualising p-values for regression slopes



Analysis of variance (L5)

The previous two chapters were about linear regression. *Linear regression* is a type of *linear model* – recall that in *R* we used the function `lm()` to make the regression model. In this chapter we will look at a different type of linear model: analysis of variance (ANOVA).

Introduction

Recall that linear regression is a linear model with one continuous explanatory (independent) variable. A continuous explanatory variable is a variable in which values can take any value within a range (e.g., height, weight, temperature).

In contrast, analysis of variance (ANOVA) is a linear model with one or more categorical explanatory variables. We will first look at a one-way ANOVA, which has one categorical explanatory variable. Later (in a following chapter) we will look at two-way ANOVA, which has two categorical explanatory variables.

What is a categorical variable? A categorical explanatory variable is a variable that contains values that fall into distinct groups or categories. For example, habitat type (e.g., forest, grassland, wetland), treatment group (e.g., control, low dose, high dose), or diet type (e.g., vegetarian, vegan, omnivore).

This means that each observation belongs to one of a limited number of categories or groups. For example, in a study of how blood pressure varies with diet type, diet type is a categorical variable with several levels (e.g., vegetarian, vegan, omnivore). A person can only belong to one diet type category.

Here are the first several rows of a dataset that contains blood pressure measurements for individuals following different diet types:

Reading in the dataset:

```
bp_data_diet <- read_csv("datasets/bp_data_diet.csv")
```

```
Rows: 50 Columns: 3
-- Column specification --------------------------------------------------------
Delimiter: ","
```

```
chr (2): diet, person_ID
dbl (1): bp

i Use `spec()` to retrieve the full column specification for this data.
i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
bp_data_diet <- select(bp_data_diet, bp, diet, person_ID)
head(bp_data_diet)
```

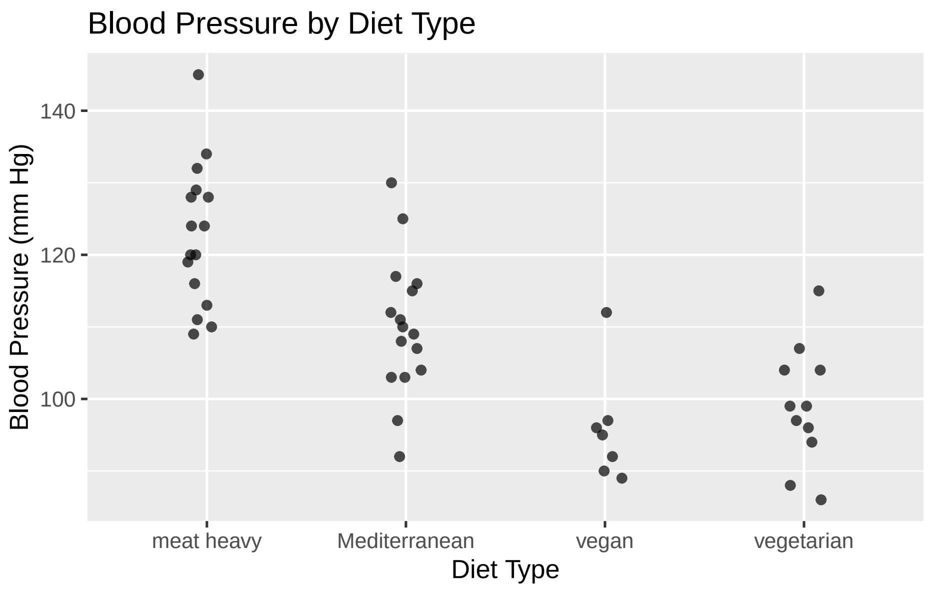
```
# A tibble: 6 x 3
     bp diet          person_ID
  <dbl> <chr>         <chr>
1   120 meat heavy    person_1
2    89 vegan         person_2
3    86 vegetarian    person_3
4   116 meat heavy    person_4
5   115 Mediterranean person_5
6   134 meat heavy    person_6
```

There are three variables: - `bp`: blood pressure (continuous response variable) - `diet`: diet type (categorical explanatory variable) - `person_ID`: unique identifier for each individual (not used in the analysis)

Note that the `diet` variable is of type `<chr>` which is short for `character`. In *R*, categorical variables are often represented as factors.

As usual, its a really good idea to visualise the data in as close to "raw" form as possible before doing any analysis. We'll make a scatterplot of blood pressure versus diet type.

```
ggplot(bp_data_diet, aes(x = diet, y = bp)) +
  geom_jitter(width = 0.1, height = 0, alpha = 0.7) +
  labs(title = "Blood Pressure by Diet Type",
       x = "Diet Type",
       y = "Blood Pressure (mm Hg)")
```



Tip

We just used `geom_jitter()` instead of `geom_point()` to make a scatter-plot. This is because `geom_jitter()` adds a small amount of random noise to the points, which helps to prevent overplotting when multiple points have the same value (which is common when the x-axis is categorical). When we use `geom_jitter()`, we can specify the amount of noise to add in the x and y directions using the `width` and `height` arguments, respectively. We must be very careful to not add noise to the y direction if we care about the actual y values (e.g., blood pressure). In this case, we only added noise in the x direction by setting `height = 0` to separate the points just enough, but not so much that we could get confused about which of the diets they belong to.

Looking at this graph it certainly looks like diet type has an effect on blood pressure. But is this effect statistically significant? In other words, are the differences in mean blood pressure between diet types larger than we would expect due to random variation alone?

Analysis of variance (ANOVA) is a statistical method that can help us answer this question, and also others.

How does it look like in R?

We can fit a one-way ANOVA model in *R* using the same `lm()` function that we used for linear regression. The only difference is that the explanatory variable is categorical.

```
anova_model <- lm(bp ~ diet, data = bp_data_diet)
```

Then instead of using `summary()` to look at the results, we use the `anova()` function.

```
anova(anova_model)
```

```
Analysis of Variance Table

Response: bp
          Df Sum Sq Mean Sq F value    Pr(>F)    
diet       3 5274.2 1758.08  20.728 1.214e-08 ***
Residuals 46 3901.5   84.82                      
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

This is an ANOVA table. It shows us the sources of variation in the data, along with their associated degrees of freedom (Df), sum of squares (Sum Sq), mean square (Mean Sq), F value, and p-value (Pr(>F)) associate with a getting a F value the same as or greater than the observed F value if the null hypothesis were true.

The challenge now is to understand what all of these values mean! Let's take it step by step.

What is ANOVA?

Analysis of variance is a method to compare whether the observations (e.g., of blood pressure) differ according to some grouping (e.g., diet) that the subjects (e.g., people) belong to.

We already know a lot about analysing variance: we compared the total sum of squares (SST), model sum of squares (SSM) and the residual sum of squares (SSE) in the context of linear regression. We used these to calculated the R^2 value. The R^2 value tells us how much of the total variance in the response variable (e.g., blood pressure) is explained by the explanatory variable (e.g., diet).

The same applies to analysis of variance (ANOVA) (as well as regression) because ANOVA is a special case of a linear model, just like regression is also a special case of a linear model.

The defining characteristic of ANOVA is that we are comparing the means of groups by analysing variances. Put another way, we will have a single categorical explanatory variable with two or more levels. We will test whether the means of the response variable are the same across all levels of the explanatory variable, and we test this by analysing the variances.

When we have only one categorical explanatory variable, we use a *one-way* ANOVA. When we have two categorical explanatory variables, we will use a *two-way* ANOVA (we'll look at this in a subsequent chapter).

ANOVA as a linear model

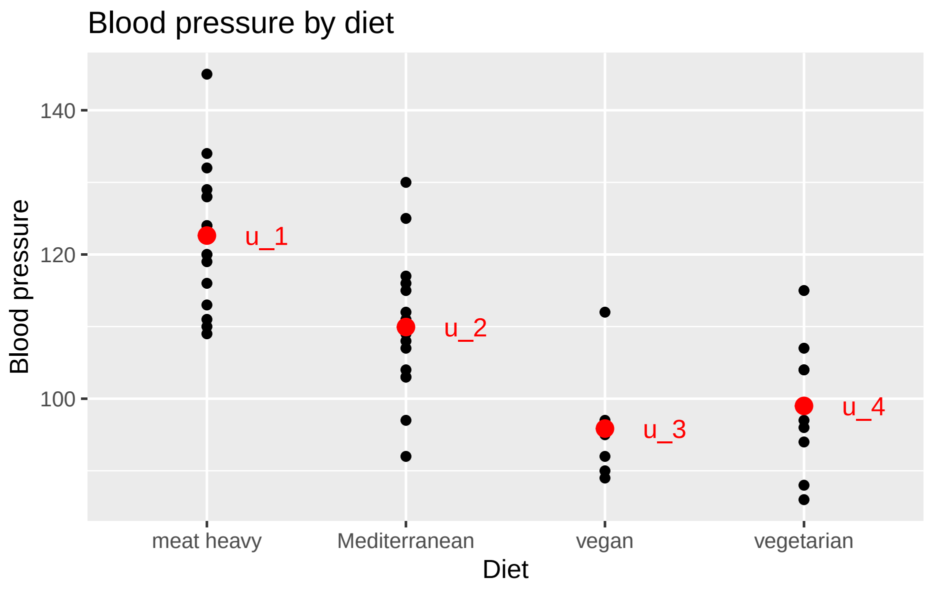
Just like linear regression, ANOVA can be expressed as a linear model. The key difference is that in ANOVA, the explanatory variable is categorical rather than continuous.We formulate the linear model as follows:

$$y_{ij} = \mu_j + \epsilon_i$$

where:

- y_{ij} = Blood pressure of individual i with diet j
- μ_i = Mean blood pressure of an individual with diet j
- $\epsilon_i \sim N(0, \sigma^2)$ is an independent error term.

Graphically, with the blood pressure and diet data, this looks like:



Note

There is lots of hidden code used to create the data used in the graph above, and to make the graph itself. You can see the code by going to the Github repository for this book.

Rewrite the model

We usually use a different formulation of the linear model for ANOVA. This is because we usually prefer to express the estimated parameters in terms of *differences between means* (rather than the means themselves). The reason for this is that then the null hypothesis can be that the differences are zero.

To proceed with this formulation, we define one of the groups as the reference group, and make the mean of that equal to the intercept of the model. For example, if we choose the "meat heavy" diet as the reference group, we can write:

$$\mu_{meat} = \beta_0$$

And then to express the other group means as deviations from the reference group mean:

$$\mu_{Med} = \beta_0 + \beta_1$$

$$\mu_{vegan} = \beta_0 + \beta_2$$

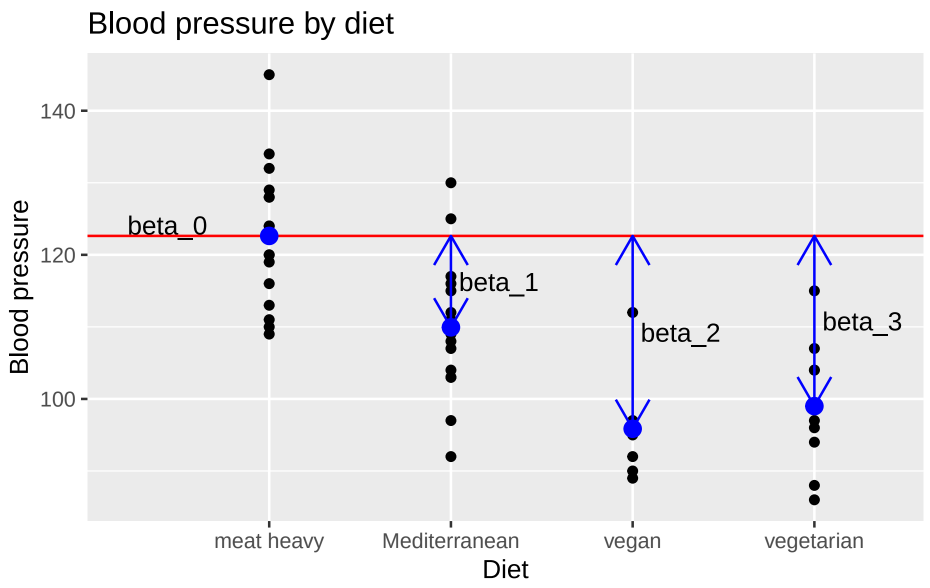
$$\mu_{veggi} = \beta_0 + \beta_3$$

When we write out the entire model, we get:

$$y_i = \beta_0 + \beta_1 x_i^1 + \beta_2 x_i^2 + \beta_3 x_i^3 + \epsilon_i$$

where: y_i is the blood pressure of individual i. x_i^1 is a binary variable indicating whether individual i is on the Mediterranean diet. x_i^2 is a binary variable indicating whether individual i is on the vegan diet. x_i^3 is a binary variable indicating whether individual i is on the vegetarian diet.

Graphically, the model now looks like this:



Here is something to warp you mind… we described one-way ANOVA as a linear model with one categorical explanatory variable. But as you can see above, we can also describe it as a linear model with multiple binary explanatory variables (one for each group except the reference group). And when we make a linear model in R it really does create multiple binary explanatory variables behind the scenes. So one-way ANOVA and multiple linear regression with multiple binary explanatory variables are really the same thing! And, even more mind-warping, one-way ANOVA and multiple regression (regression with multiple continuous explanatory variables) are also the same thing! So when we look at multiple regression later in the course, you can think of it as just an extension of one-way ANOVA.

The ANOVA test: The F-test

Aim of ANOVA: to test *globally* if the groups differ. That is we want to test the null hypothesis that all of the group means are equal:

$$H_0 : \mu_1 = \mu_2 = ... = \mu_g$$

This is equivalent to testing if all βs that belong to a categorical variable are = 0.

$$H_0 : \beta_1 = ... = \beta_{g-1} = 0$$

The alternate hypothesis is that H_1: The group means are not all the same.

A key point is that we are testing a null hypothesis that concerns all the groups. We are not testing if one group is different from another group (which we could do with a t-test on one of the non-intercept βs).

Because we are testing a null hypothesis that concerns all the groups, we need to use an F-test. It asks if the model with the group means is better than a model with just the overall mean.

Note

The F-test is called the "F-test" because it is based on the F-distribution, which was named after the statistician Sir Ronald A. Fisher. Fisher developed this statistical method as part of his pioneering work in analysis of variance (ANOVA) and other fields of experimental design and statistical inference.

Actually, the F-test does not directly test the null hypothesis that all the group means are equal. Instead, it tests whether the model that includes the *group means* explains significantly more variance in the data than a model that only includes the overall mean (i.e., without considering group differences).

The F-test does this by comparing two variance estimates: the variance explained by the group means (between-group variance) and the variance that remains unexplained within each group (within-group variance).

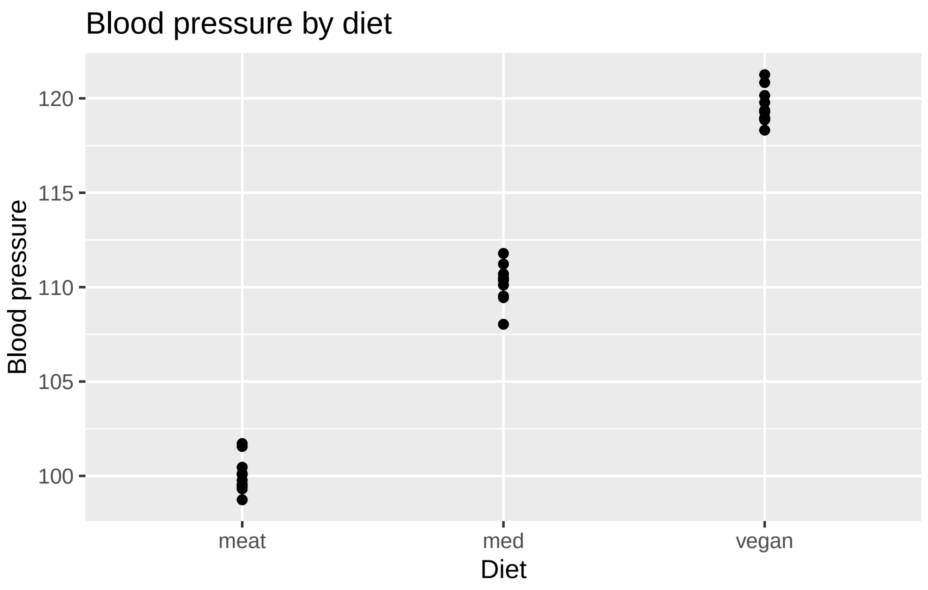
Think–Pair–Share (#a_what_f_test_answers) What question does the ANOVA F-test answer? What question does it *not* answer?

Interpretation of the F statistic

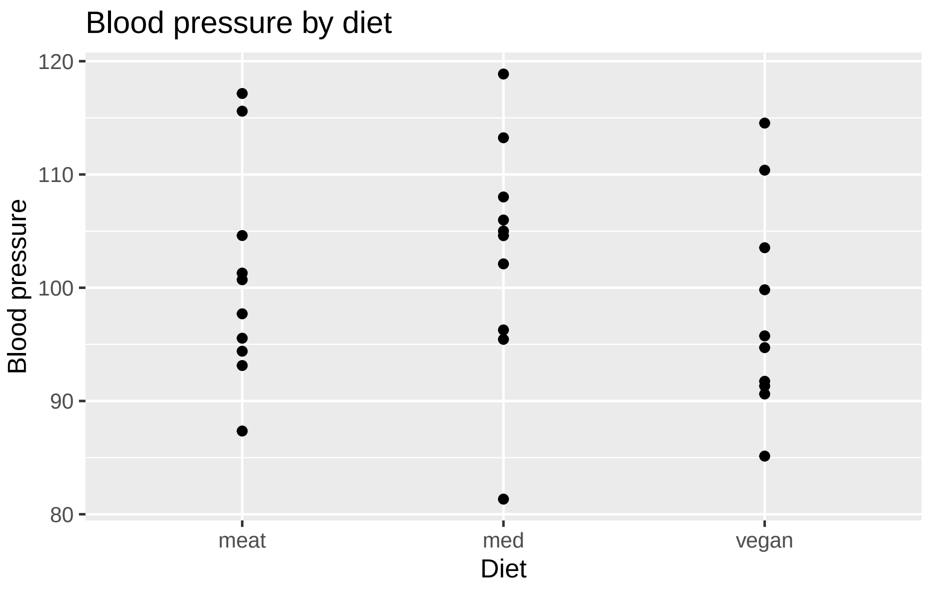
The F-test involves calculating from the observed data the value of the F statistic, and then computing if that value is large enough to reject the null hypothesis.

The F statistic is a ratio of two variances: the variance **between** groups, and the variance **within** groups.

Here is an example with very low within group variability, and high between group variability:



And here's an example with very high within group variability, and low between group variability:



So, when the ratio of between group variance to within group variance is large, the group means are very different compared to the variability within groups. This suggests that the groups are different.

When the ratio is small, the group means are similar compared to the variability within groups. This suggests that the groups are not different.

- F **increases**
  - when the group means become more different, or
  - when the variability within groups decreases.
- F **decreases**
  - when the group means become more similar, or
  - when the variability within groups increases.

$\rightarrow$ The larger F, the less likely are the data seen under H_0.

Calculating the F statistic

Recall that the F statistic is a ratio of two variances. Specifically, it is the ratio of two mean squares (MS):

- MS_{model}: the variability **between** groups.
- $MS_{residual}$: the variability **within** groups.

MS stands for Mean Square, and is a variance estimate.

The F statistic is calculated as:

$$F = \frac{MS_{model}}{MS_{residual}}$$

To find the mean squares, we need to calculate the within and the between group sums of squares, and the corresponding degrees of freedom. Let's go though this step by step.

Calculating the sums of squares

First we get the total sum of squares (SST), which quantifies the total variability in the data. This is then split into the explained variability (SSM), and the residual variability (SSE).

Total variability: $\text{SST} = \sum_{i=1}^{k} \sum_{j=1}^{n_i} (y_{ij} - \overline{y})^2$

where:

- y_{ij} is the blood pressure of individual j in group i
- $\overline{y}$ is the overall mean blood pressure
- n_i is the number of individuals in group i
- k is the number of groups

Explained variability (between group variability): $= \text{SSM} = \sum_{i=1}^{k} n_i (\overline{y}_i - \overline{y})^2$

where:

- $\overline{y}_i$ is the mean blood pressure of group i

Residual variability (within group variability): $= \text{SSE} = \sum_{i=1}^{k}\sum_{j=1}^{n_i}(y_{ij} - \overline{y}_i)^2$

Calculating the degrees of freedom

And now we need the degrees of freedom for each sum of squares:

SST degrees of freedom: $n-1$ (total degrees of freedom is number of observations n minus 1)

SSM degrees of freedom: $k-1$ (model degrees of freedom is number of groups k minus 1)

SSE degrees of freedom: $n-k$ (residual degrees of freedom is total degrees of freedom $n-1$ minus model degrees of freedom $k-1$)

Total degrees of freedom

The total degrees of freedom are the degrees of freedom associated with the total sum of squares (SST).

In order to calculate the SST, we need to calculate the mean of the response variable. This implies that we estimate one parameter (the mean of the response variable). As a consequence, we lose one degree of freedom and so there remain $n-1$ degrees of freedom associated with the total sum of squares (where n is the number of observations).

What do we mean by "lose one degree of freedom"? Imagine we have ten observations. We can calculate the mean of these ten observations. But if we know the mean and nine of the observations, we can calculate the tenth observation. So, in a sense, once we calculate the mean, the value of one of the ten observations is fixed. This is what we mean by "losing one degree of freedom". When we calculate and use the mean, one of the observations "loses its freedom".

For example, take the numbers 1, 3, 5, 7, 9. The mean is 5. The sum of the squared differences between the observations and the mean is $(1-5)^2 + (3-5)^2 + (5-5)^2 + (7-5)^2 + (9-5)^2 = 20$. This is the total sum of squares. The degrees of freedom are $5-1=4$.

The total degrees of freedom are the total number of observations minus one. That is, the total sum of squares is associated with $n-1$ degrees of freedom.

Another perspective in which to think about the total sum of squares and total degrees of freedom is to consider the intercept only model. The intercept only model is a model that only includes the intercept term. The equation of this model would be:

$$y_i = \beta_0 + \epsilon_i$$

The sum of the square of the residuals for this model is minimised when the predicted value of the response variable is the mean of the response variable. That is, the least squares estimate of β_0 is the mean of the response variable:

$$\hat{\beta}_0 = \bar{y}$$

Hence, the predicted value of the response variable is the mean of the response variable. The equation is:

$$\hat{y}_i = \bar{y} + \epsilon_i$$

The error term is therefore:

$$\epsilon_i = y_i - \bar{y}$$

And the total sum of squares is:

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

where $\hat{y}_i$ is the predicted value of the response variable for the ith observation, $\bar{y}$ is the mean of the response variable, and ϵ_i is the residual for the ith observation.

The intercept only model involves estimating only one parameter, so the total degrees of freedom are the total number of observations minus one $n-1$.

Therefore, the total degrees of freedom are the total number of observations minus one.

Bottom line: SST is the residual sum of squares when we fit the intercept only model. The total degrees of freedom are the total number of observations minus one.

Model degrees of freedom

The model degrees of freedom are the degrees of freedom associated with the model sum of squares (SSM).

In the case of the intercept only model, we estimated one parameter, the mean of the response variable.

In the case of a categorical variable with k groups, we need $k-1$ parameters (non intercept β parameters), so we lose $k-1$ degrees of freedom. Put another way, when we fit a model with a categorical explanatory variable with k groups, we estimate $k-1$ parameters in addition to the intercept. That is, we estimate the difference between each group and the reference group.

Each time we estimate a new parameter, we lose a degree of freedom.

Residual degrees of freedom

The residual degrees of freedom are the total degrees of freedom $(n-1)$ minus the model degrees of freedom $(k-1)$.

Therefore, the residual degrees of freedom are the degrees of freedom remaining after we estimate the intercept and the other β parameters. There is one intercept and $k-1$ other β parameters, so the residual degrees of freedom are $n-1-(k-1)=n-k$.

Calculating the mean square and F statistic

From these sums of squares and degrees of freedom we can calculate the mean squares and F-statistic:

$$MS_{model} = \frac{SS_{\text{between}}}{k-1} = \frac{SSM}{k-1}$$

$$MS_{residual} = \frac{SS_{\text{within}}}{n-k} = \frac{SSE}{n-k}$$

$$F = \frac{MS_{model}}{MS_{residual}}$$

Note

Why divide by the degrees of freedom? The more observations we have, the greater will be the total sum of squares. The more observations we have, the greater will be the residual sum of squares. So it is not very informative to compare totals. Rather, we need to compare the mean of the sums of squares. Except we don't calculate the mean by dividing by the number of observations. Rather we divide by the degrees of freedom. The total mean square is an estimate of the variance of the response variable. And the residual mean square is an estimate of the variance of the residuals.

SST, SSM, SSE, and degrees of freedom

Just a reminder and a summary of some of the material above:

- SST: degrees of freedom $= n-1$
- SSM: degrees of freedom $= k-1$
- SSE: degrees of freedom $= n-k$

The sum of squares add up:

$$SST = SSM + SSE$$

and the degrees of freedom add up

$$(n-1) = (k-1) + (n-k)$$

Source of variance table

Now we have nearly everything we need. We often express all of this (and a few more quantities) in a convenient table called the **sources of variance table** (or ANOVA table).

The **sources of variance table** is a table that conveniently and clearly gives all of the quantities mentioned above. It breaks down the total sum of squares into the sum of squares explained by the model and the sum of squares due to error. The source of variance table is used to calculate the F-statistic.

Table 1: Sources of variance table

Source	Sum of squares	Degrees of freedom	Mean square	F-statistic
Model	SSM	$k-1$	$MSE_{model} = SSM/k-1$	$\frac{MSE_{model}}{MSE_{error}}$
Error	SSE	$n-1-(k-1)$	$MSE_{error} = SSE/(n-1-(k-1))$	
Total	SST	$n-1$		

Is my F-statistic large or small?

OK, so we have calculated the F statistic. But how do we use it to test our hypothesis?

We can use the F statistic to calculate a p-value, which tells us how likely our data is under the null hypothesis.

Some key points:

1. F-Distribution: The test statistic of the F-test (that is, the F-statistic) follows the F-distribution under the null hypothesis. This distribution arises when comparing the ratio of two independent sample variances (or mean squares).
2. Ronald Fisher's Contribution: Fisher introduced the F-distribution in the early 20th century as a way to test hypotheses about the equality of variances and to analyze variance in regression and experimental designs. The "F" in F-distribution honours him.

3. Variance Ratio: The test statistic for the F-test is the ratio of two variances (termed mean squares in this case), making the F-distribution the natural choice for modeling this ratio when the null hypothesis is true.

The F-test is widely used, including when comparing variances, assessing the significance of multiple regression models (see later chapter), conducting ANOVA to test for differences among group means, and for comparing different models.

Recall that "The F-statistic is calculated as the ratio of the mean square error of the model to the mean square error of the residuals." And that a large F-statistic is evidence against the null hypothesis that the slopes of the explanatory variables are zero. And that a small F-statistic is evidence to not reject the null hypothesis that the slopes of the explanatory variables are zero.

But how big does the F-statistic need to be in order to confidently reject the null hypothesis?

The null hypothesis that the explained variance of the model is no greater than would be expected by chance. Here, "by chance" means that the slopes of the explanatory variables are zero.

$$H_0 : \beta_1 = \beta_2 = ... = \beta_p = 0$$

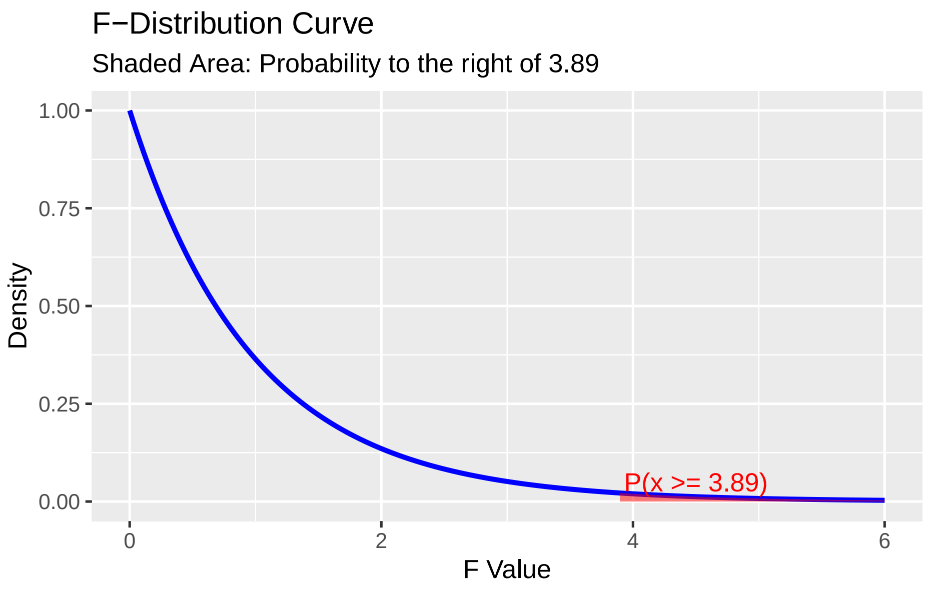
The alternative hypothesis is that the explained variance of the model is greater than would be expected by chance. This would occur if the slopes of some or all of the explanatory variables are not zero.

$$H_1 : \beta_1 \neq 0 \text{ or } \beta_2 \neq 0 \text{ or } \ ... \ \text{ or } \beta_p \neq 0$$

To test this hypothesis we are going to, as usual, calculate a p-value. The p-value is the probability of observing a test statistic as or more extreme as the one we observed, assuming the null hypothesis is true. To do this, we need to know the distribution of the test statistic under the null hypothesis. The distribution of the test statistic under the null hypothesis is known as the F-distribution.

The F-distribution has two degrees of freedom values associated with it: the degrees of freedom of the model and the degrees of freedom of the residuals. The degrees of freedom of the model are the number of parameters estimated by the model corresponding to the null hypothesis. The degrees of freedom of the residuals are the total degrees of freedom minus the degrees of freedom of the model.

Here is the F-distribution with 2 and 99 degrees of freedom:



The F-distribution is skewed to the right and has a long tail. The area to the right of 3.89 is shaded in red. This area represents the probability of observing an F-statistic as or more extreme as 3.89, assuming the null hypothesis is true. This probability is the p-value of the hypothesis test.

The F-statistic and F-test is briefly recaptured in 3.1.f) of the Stahel script, but see also Mat183 chapter 6.2.5. It uses the fact that

$$\frac{MSE_{model}}{MSE_{residual}} = \frac{SSM/p}{SSE/(n-1-p)} \sim F_{p,n-1-p}$$

follows an F-distribution with p and $(n-1-p)$ degrees of freedom, where p are the number of continuous variables, n the number of data points.

- $SSE = \sum_{i=1}^{n}(y_i - \hat{y}_i)^2$ is the residual sum of squares
- $SSM = SST - SSE$ is the sum of squares of the model
- $SST = \sum_{i=1}^{n}(y_i - \overline{y})^2$ is the total sum of squares
- n is the number of data points
- p is the number of explanatory variables in the regression model

Well, that is ANOVA conceptually. But how does it actually look like in R?

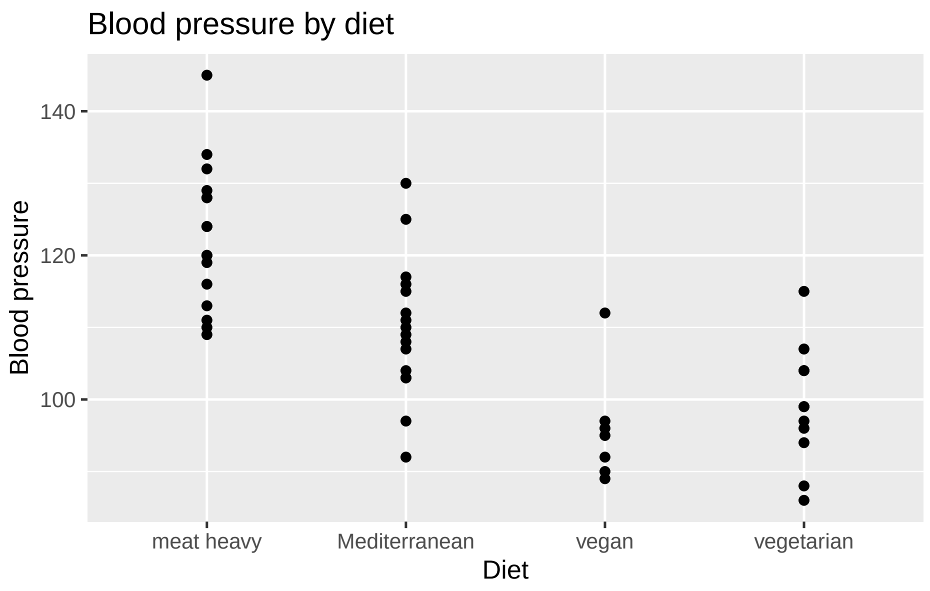
Doing ANOVA in R

Let's go back again the question of how diet effects blood pressure. Here is the data:

```r
bp_data_diet <- select(bp_data_diet, bp, diet, person_ID)
head(bp_data_diet)
```

```
# A tibble: 6 x 3
     bp diet          person_ID
  <dbl> <chr>         <chr>
1   120 meat heavy    person_1
2    89 vegan         person_2
3    86 vegetarian    person_3
4   116 meat heavy    person_4
5   115 Mediterranean person_5
6   134 meat heavy    person_6
```

```r
ggplot(bp_data_diet, aes(x = diet, y = bp)) +
  geom_point() +
  labs(title = "Blood pressure by diet",
       x = "Diet",
       y = "Blood pressure")
```



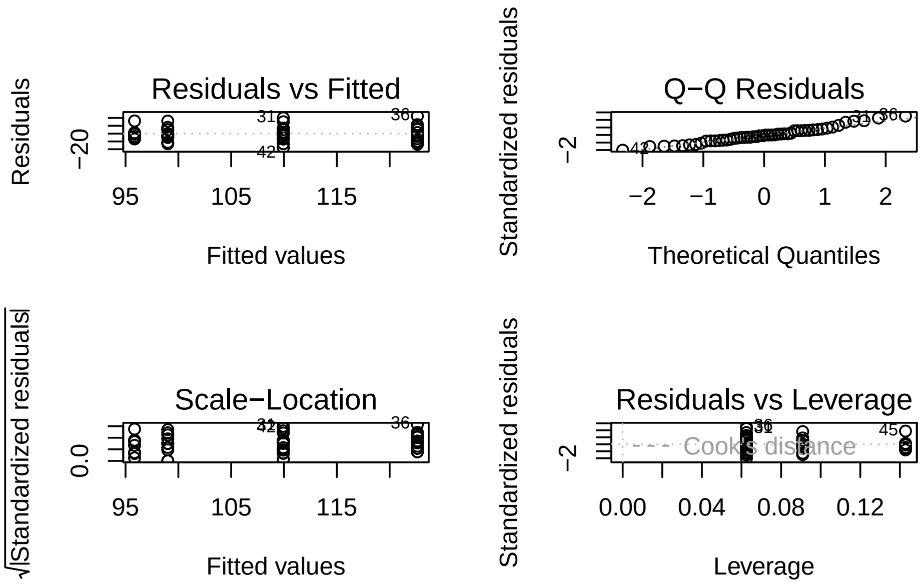
And here is how we fit a linear model to this data:

```r
fit <- lm(bp ~ diet, data = bp_data_diet)
```

IMPORTANT: Since ANOVA is a linear model, it is important to check the assumptions of linear models before interpreting the results. These are some of the same assumptions we checked for simple linear regression, including: independence of errors, normality of residuals, and homoscedasticity (constant variance of residuals).

As with linear regression, we check the assumptions are not too badly broken by looking at model checking plots:

```r
par(mfrow = c(2, 2))
plot(fit, add.smooth = FALSE)
```



Nothing looks too bad.

** Think-pair-share**: Which of the four plots above would you use to check each of the three assumptions listed above?

** Think-pair-share**: Before we look at the ANOVA table, lets figure out the total degrees of freedom, the degrees of freedom for the model, and the degrees of freedom for the residuals. Have a think-pair-share about each of these. Write you ideas down. Chat with you neighbour. Then share with the class.

Now we can look at the ANOVA table:

```r
anova(fit)
```

```
Analysis of Variance Table

Response: bp
          Df Sum Sq Mean Sq F value    Pr(>F)    
diet       3 5274.2 1758.08  20.728 1.214e-08 ***
Residuals 46 3901.5   84.82                      
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The ANOVA table shows the sum of squares, degrees of freedom, mean square, F value, and p-value for the model and residuals. As we know, the F value (F statistics) is calculated as the mean square of the model divided by the mean square of the residuals. The p-value is calculated based on the F-distribution with the appropriate degrees of freedom.

A suitable sentence to report our findings would be: "Diet has a significant effect on blood pressure ($F(2,27) = 20.7, p < 0.0001$)". This means that the probability of observing such a large F value under the null hypothesis is less than 0.01%.

Think-pair-share: You know that the R^2 value is a measure of how much variance in the response variable is explained by the model. How would you calculate the R^2 value from the ANOVA table above?

Difference between pairs of groups

"*ANOVA does not tell you which groups differ.*"

Recall that the F test is a global test. It tests the null hypothesis that all group means are equal. It does not tell us which groups are different from each other. It just tells us that at least one group mean is different. Sometimes researchers are interested in more specific questions such as:

1. finding the actual group(s) that deviate(s) from the others.
2. in estimates of the pairwise differences.

The summary table in R provides some of these comparison, specifically it contains the estimates for β_1, β_2, β_3 (while the reference was set to $\beta_0 = 0$).

For example, here is the summary table for our diet data:

```r
summary(fit)
```

```

Call:
lm(formula = bp ~ diet, data = bp_data_diet)

Residuals:
     Min       1Q   Median       3Q      Max 
-17.9375  -5.9174  -0.4286   5.2969  22.3750 

Coefficients:
                  Estimate Std. Error t value Pr(>|t|)    
(Intercept)        122.625      2.302  53.260  < 2e-16 ***
dietMediterranean  -12.688      3.256  -3.897 0.000314 ***
dietvegan          -26.768      4.173  -6.414 6.92e-08 ***
dietvegetarian     -23.625      3.607  -6.549 4.33e-08 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
Residual standard error: 9.21 on 46 degrees of freedom
Multiple R-squared:  0.5748,    Adjusted R-squared:  0.5471
F-statistic: 20.73 on 3 and 46 DF,  p-value: 1.214e-08
```

In this table we have the intercept (β_0) and the three β values for the diet groups: "dietMeat", "dietVegetarian", and "dietVegan". The Estimate column shows the estimated coefficients for each group. The intercept (β_0) represents the mean blood pressure for the reference group (in this case, the "meat" diet group). The other three coefficients represent *the difference* in mean blood pressure between each diet group and the reference group.

All well and good up to a point. But there are two issues with using the results from this table:

1. The greater the number of individual tests, the more likely one will be significant just by chance. This is called the problem of multiple comparisons. Many test can result in a type-I error: rejecting the null hypothesis when it is actually true. The more tests one does, the more likely one is to make a type-I error.

** Think-pair-share**: Imagine that when our threshold p-value each individual test is 0.05 (5%) so that if it is less than 0.05 we call it "significant" and if it is greater than 0.05 we call it "not significant" (this is the standard practice in many fields). When we make 20 hypothesis tests, how many would we expect to be "significant" just by chance (i.e., when we assume that all null hypotheses is true.)

2. The summary table does not provide all the possible pairwise comparisons. It does not, for example, provide the comparison between the "vegan" and the "vegetarian" group.

Several methods to circumvent the problem of too many "significant" test results (type-I error) have been proposed. The most prominent ones are:

- Bonferroni correction
- Tukey **H**onest **S**ignificant **D**ifferences (HSD) approach
- Fisher **L**east **S**ignificant **D**ifferences (LSD) approach

The second two when implemented in R also provide all possible pairwise comparisons.

Bonferroni correction

Idea: If a total of m tests are carried out, simply divide the type-I error level α_0 (often 5%) such that

$$\alpha = \alpha_0 / m \ .$$

But this still leaves the problem of how to efficiently get all of the possible pairwise comparisons. We can do this using the `pairwise.t.test` function in R:

```
pairwise.t.test(bp_data_diet$bp,
                bp_data_diet$diet,
                p.adjust.method = "bonferroni")
```

```

    Pairwise comparisons using t tests with pooled SD 

data:  bp_data_diet$bp and bp_data_diet$diet 

              meat heavy Mediterranean vegan 
Mediterranean 0.0019     -             -     
vegan         4.2e-07    0.0091        -     
vegetarian    2.6e-07    0.0239        1.0000

P value adjustment method: bonferroni 
```

Here we can see that all pairwise comparisons have a p-value less than 0.05, except for the comparison of vegan versus vegetarian, which has a p-value that rounds to 1.0000.

We also see in the output the note that "P value adjustment method: bonferroni", indicating that the Bonferroni correction has been applied to the p-values.

Tukey HSD approach

Idea: Take into account the distribution of *ranges* (max-min) and design a new test.

In R we can use the `multcomp` package to do Tukey HSD tests:

```
bp_data_diet <- bp_data_diet %>%
  mutate(diet = as.factor(diet))
fit <- lm(bp ~ diet, data = bp_data_diet)
suppressMessages(library(multcomp))
tukey_test <- glht(fit, linfct = mcp(diet = "Tukey"))
summary(tukey_test)
```

```

     Simultaneous Tests for General Linear Hypotheses

Multiple Comparisons of Means: Tukey Contrasts


Fit: lm(formula = bp ~ diet, data = bp_data_diet)
```

```
Linear Hypotheses:
                                Estimate Std. Error t value Pr(>|t|)
Mediterranean - meat heavy == 0  -12.688      3.256  -3.897  0.00168 **
vegan - meat heavy == 0          -26.768      4.173  -6.414  < 0.001 ***
vegetarian - meat heavy == 0     -23.625      3.607  -6.549  < 0.001 ***
vegan - Mediterranean == 0       -14.080      4.173  -3.374  0.00759 **
vegetarian - Mediterranean == 0  -10.938      3.607  -3.032  0.01951 *
vegetarian - vegan == 0            3.143      4.453   0.706  0.89305
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
(Adjusted p values reported -- single-step method)
```

We get all the pairwise comparisons, along with their estimates, standard errors, t-values, and p-values. We also get a note `Adjusted p values reported -- single-step method`, indicating that the Tukey HSD adjustment has been applied to the p-values.

Again, all pairwise comparisons have a p-value less than 0.05, except for the comparison of vegan versus vegetarian, which has a p-value of 0.89305.

Fisher's LSD approach

Idea: Adjust the idea of a two-sample test, but use a larger variance (namely the pooled variance of all groups).

Other contrasts

A contrast is a specific comparison between groups. So far we have only considered pairwise contrasts (i.e., comparing two groups at a time). But we can also design more complex contrasts. For example: are diets that contain meat different from diets that do not contain meat?

```
bp_data_diet <- mutate(bp_data_diet,
                      meat_or_no_meat = ifelse(diet == "meat" |
                                                 diet == "Mediterranean",
                                                 "meat", "no meat"))
head(bp_data_diet)
```

```
# A tibble: 6 x 4
     bp diet          person_ID meat_or_no_meat
  <dbl> <fct>         <chr>     <chr>
1   120 meat heavy    person_1  no meat
2    89 vegan         person_2  no meat
3    86 vegetarian    person_3  no meat
4   116 meat heavy    person_4  no meat
5   115 Mediterranean person_5  meat
6   134 meat heavy    person_6  no meat
```

Here we defined a new explanatory variable that groups the meat heavy and Mediterranean diet together into a single "meat" group and vegetarian and vegan into a single "no meat" group. We then fit a model with this explanatory variable:

```
fit_mnm <- lm(bp ~ meat_or_no_meat, data = bp_data_diet)
```

(We should not look at model checking plots here, before using the model. But let us continue as if the assumptions are sufficiently met.)

We now do something a bit more complicated: we compare the variance explained by the model with four diets to the model with two diets. This is done by comparing the two models using an F-test. We are testing the null hypothesis that the two models are equally good at explaining the data, in which case the two diet model will explain as much variance as the four diet model.

Let's look at the ANOVA table of the model comparison:

```
anova(fit, fit_mnm)
```

```
Analysis of Variance Table

Model 1: bp ~ diet
Model 2: bp ~ meat_or_no_meat
  Res.Df    RSS Df Sum of Sq      F    Pr(>F)
1     46 3901.5
2     48 9173.4 -2   -5271.9 31.078 2.886e-09 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

We see the residual sum of squares of the model with meat or no meat is over 9'000, while that of the four diet model is less than 4'000. That is, the four diet model explains much more variance in the data than the two diet model. The F-test is highly significant, so we reject the null hypothesis that the two models are equally good at explaining the data. And we conclude that its not just whether people eat meat or not, but rather what kind of diet they eat that affects their blood pressure.

Ideally we do not make a lot of contrasts after we have collected and looked at our data. Rather, we would specify the contrasts we are interested in before we collect the data. This is called a priori contrasts. But sometimes we do exploratory data analysis and then we can make post hoc contrasts. In this case we should be careful to adjust for multiple comparisons.

Choosing the reference category

Question: Why was the "heavy meat" diet chosen as the reference (intercept) category?

Answer: Because R orders the categories alphabetically and takes the first level alphabetically as reference category.

Sometimes we may want to override this, for example if we have a treatment that is experimentally the control, then it will usually be useful to set this as the reference / intercept level.

In R we can set the reference level using the `relevel` function:

```
bp_data_diet$diet <- relevel(factor(bp_data_diet$diet), ref = "vegan")
```

And now make the model and look at the estimated coefficients:

```
fit_vegan_ref <- lm(bp ~ diet, data = bp_data_diet)
summary(fit_vegan_ref)
```

```

Call:
lm(formula = bp ~ diet, data = bp_data_diet)

Residuals:
     Min       1Q   Median       3Q      Max 
-17.9375  -5.9174  -0.4286   5.2969  22.3750 

Coefficients:
                  Estimate Std. Error t value Pr(>|t|)    
(Intercept)         95.857      3.481  27.538  < 2e-16 ***
dietmeat heavy      26.768      4.173   6.414 6.92e-08 ***
dietMediterranean   14.080      4.173   3.374  0.00151 ** 
dietvegetarian       3.143      4.453   0.706  0.48386    
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 9.21 on 46 degrees of freedom
Multiple R-squared:  0.5748,	Adjusted R-squared:  0.5471 
F-statistic: 20.73 on 3 and 46 DF,  p-value: 1.214e-08
```

Now we see the estimated coefficients for all diets except the vegan diet. The intercept is the mean individuals with vegan diet.

Communicating the results of ANOVA

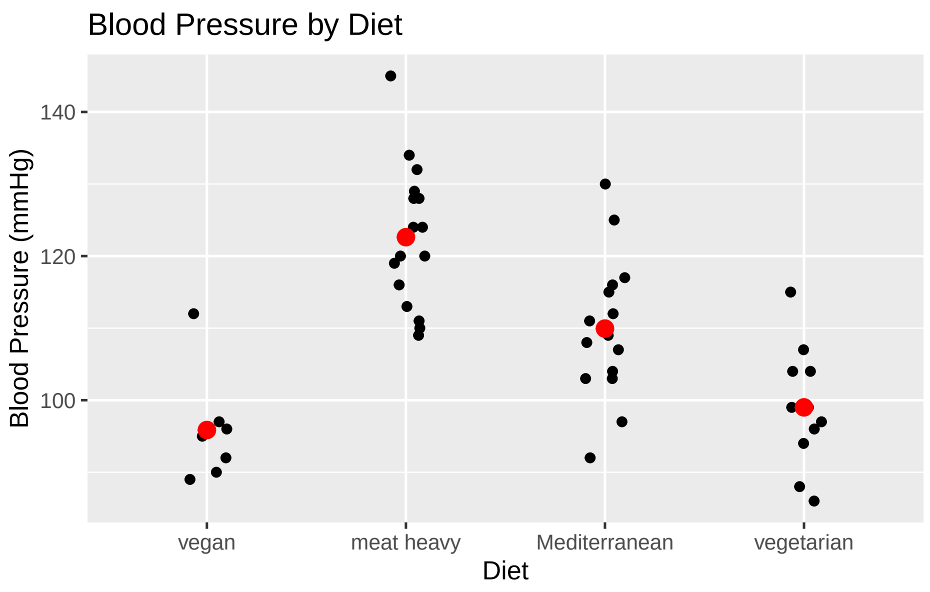
When communicating the results of an ANOVA, we usually report the F-statistic, the degrees of freedom of the numerator and denominator, and the p-value. For example, we could say:

> Blood pressure differed significantly between groups, with the mean of a meat heavy diet being 123 mmHg, while the mean blood pres-

sure of the vegan group was 27 mmHg lower (One-way ANOVA, $F(3,46) = 20.7$, $p < 0.0001$.

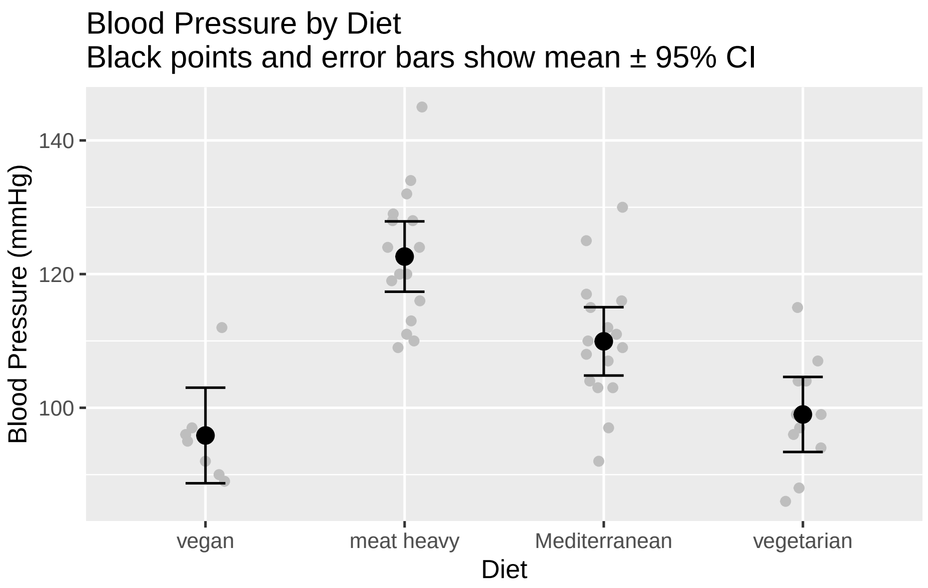
And we would make a nice graph, in this case showing each individual observation since there are not too many to cause overplotting. We can also add the estimated means of each group if we like:

```
ggplot(bp_data_diet, aes(x = diet, y = bp)) +
  geom_jitter(width = 0.1, height = 0) +
  stat_summary(fun = mean, geom = "point", color = "red", size = 3) +
  labs(title = "Blood Pressure by Diet",
       x = "Diet",
       y = "Blood Pressure (mmHg)")
```



Some people like to see error bars as well, for example showing the 95% confidence intervals of the means:

```
ggplot(bp_data_diet, aes(x = diet, y = bp)) +
  geom_jitter(width = 0.1, height = 0, col = "grey") +
  stat_summary(fun = mean, geom = "point", color = "black", size = 3) +
  stat_summary(fun.data = mean_cl_normal, geom = "errorbar", width = 0.2, color = "black") +
  labs(title = "Blood Pressure by Diet\nBlack points and error bars show mean ± 95% CI",
       x = "Diet",
       y = "Blood Pressure (mmHg)")
```



There are many many plotting styles and preferences. The important thing is to clearly communicate the results, and to not mislead the reader. I find that plotting the individual data points is often a good idea, especially when the sample size is not too large.

Review

Here are the key points from this chapter:

- ANOVA is just another linear model.
- It is used when we have categorical explanatory variables.
- We use F-tests to test the null hypothesis of no difference among the means of the groups (categories).
- We can use contrasts and post-hoc tests to test specific hypotheses about the means of the groups.

Think–Pair–Share (#a_anova_is_regression) What changes when we move from regression to ANOVA? What stays exactly the same?

Multiple regression (L6)

Introduction

In the previous chapters we covered simple linear regression and one-way analysis of variance. In both we had one response variable and one explanatory variable. In both cases we made a linear model to relate the response variable to the explanatory variable. The two cases differed in the type of explanatory variable. In the case of simple linear regression, the explanatory variable was continuous. In the case of one-way ANOVA, the explanatory variable was categorical.

We will now extend the linear model and analyses to cases with more than one (i.e., multiple) explanatory variables. The explanatory variables can be continuous or categorical, and can be a mixture of the two.

Some combinations of explanatory variables have special names:

- Multiple (more than one) continuous explanatory variables -> Multiple linear regression.
- Two categorical explanatory variables -> Two-way ANOVA.
- One continuous and one categorical explanatory variable -> Analysis of covariance (ANCOVA).

We will look at each of these, and start with multiple linear regression which is usually shortened to just multiple regression.

Multiple regression

We previously looked at whether blood pressure is associated with age. This is an important question, because blood pressure has many health implications. However, blood pressure is not only associated with age, but also with other factors, such as weight, height, and lifestyle. In this chapter, we will look at how to investigate the association between blood pressure and multiple explanatory variables.

When we have multiple explanatory variables, we are often interested in questions such as:

- Question 1: As an ensemble (i.e., all together), are the explanatory variables "useful"?
- Question 2: Are each of the explanatory variables associated with the response?
- Question 3: What proportion of variability is explained?
- Question 4: Are some explanatory variables more important than others?

An example dataset

Blood pressure is again the response variable, with age and lifestyle as two explanatory variables. Lifestyle is a continuous variable that is the number of minutes of exercise per week.

Reading in the dataset:

```
bp_data_multreg <- read_csv("datasets/bp_data_multreg.csv")
```

```
Rows: 100 Columns: 3
-- Column specification --------------------------------------------------------
Delimiter: ","
dbl (3): age, mins_exercise, bp

i Use `spec()` to retrieve the full column specification for this data.
i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

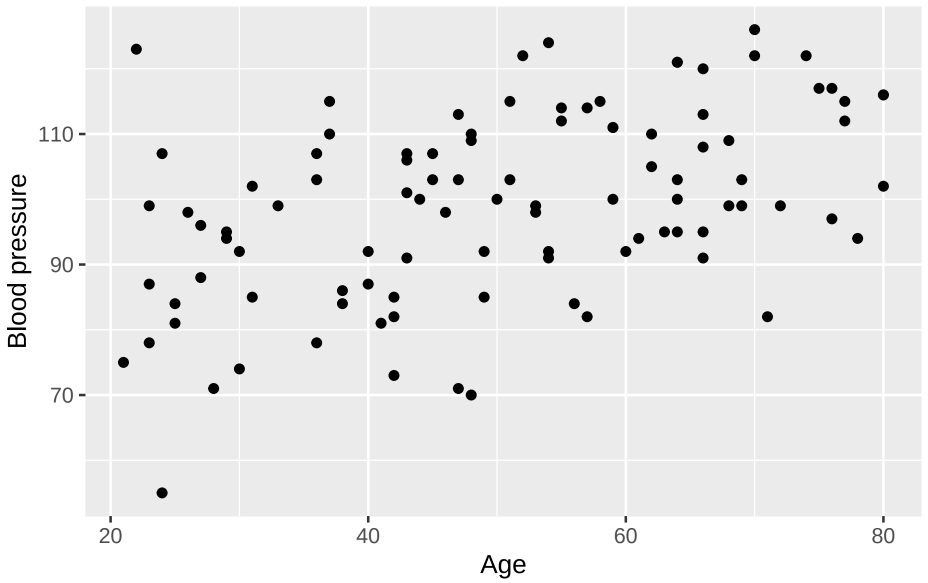
Here is a look at the dataset:

```
head(bp_data_multreg)
```

```
# A tibble: 6 x 3
    age mins_exercise    bp
  <dbl>         <dbl> <dbl>
1    59           148   100
2    80           213   116
3    64             3   121
4    80            26   116
5    54           251    91
6    59            59   111
```

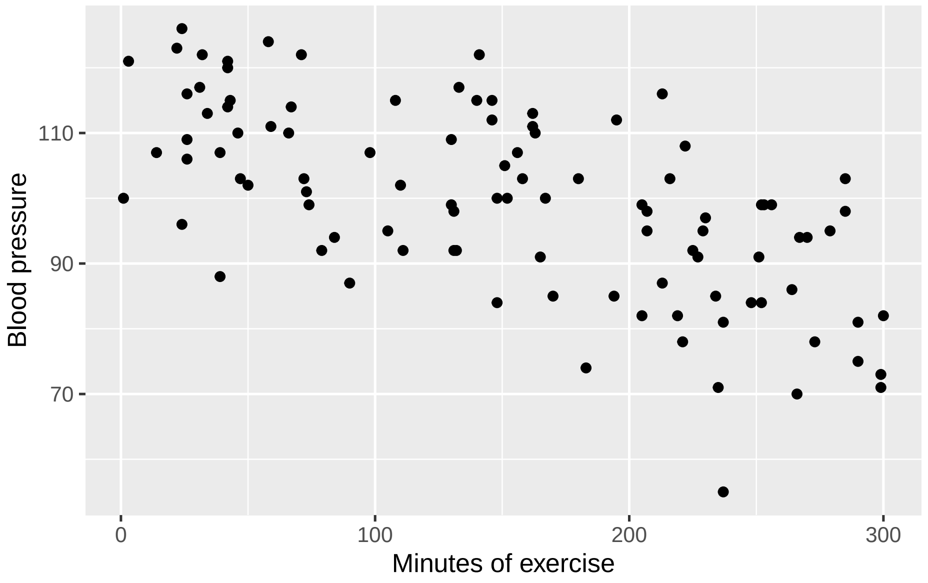
Since there are three variables, we can make three different scatter plots to visualise the relationships:

1. Age vs blood pressure. This is the graph of the response variable (blood pressure) against one of the explanatory variables (age). It looks like there is evidence of a positive relationship.



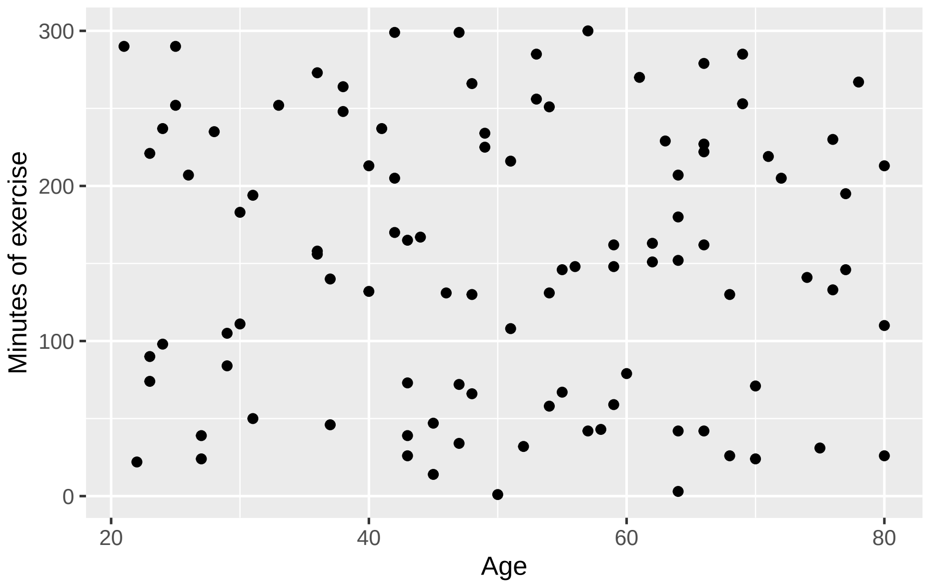
Here we see a positive relationship between age and blood pressure. Blood pressure tends to increase with age.

2. Minutes of exercise vs blood pressure. This is a graph of the response variable (blood pressure) against the other explanatory variable (minutes of exercise). It looks like there is evidence of a negative relationship.



Here we see a negative relationship between minutes of exercise and blood pressure. Blood pressure tends to decrease with more minutes of exercise.

3. Age vs minutes of exercise. This is a graph of the two explanatory variables against each other. It looks like there is no relationship.



And here we see no relationship between age and minutes of exercise. The two explanatory variables appear to be independent.

! Important

The lack of correlation between the two explanatory variables is very important. If the two explanatory variables were correlated, we would have a situation known as multicollinearity. Multicollinearity can greatly complicate the interpretation of the results of a multiple regression analysis. We will discuss multicollinearity later.

The multiple linear regression model

The multiple linear regression model is an extension of the simple linear regression model. Recall the simple linear regression model is:

$$y_i = \beta_0 + \beta_1 x_i + \epsilon_i$$

where:

- y_i is the response variable
- x_i is the explanatory variable
- β_0 is the intercept
- β_1 is the slope

- ϵ_i is the error term.

The multiple linear regression model with two explanatory variables is:

$$y_i = \beta_0 + \beta_1 x_i^{(1)} + \beta_2 x_i^{(2)} + \epsilon_i$$

where:

- $x_i^{(1)}$ and $x_i^{(2)}$ are the two explanatory variables
- β_0 is the intercept
- β_1 is the slope for the first explanatory variable
- β_2 is the slope for the second explanatory variable

Note that the intercept β_0 is the value of the response variable when all explanatory variables are zero. In this example, it would be the blood pressure for someone that is 0 years old and does 0 minutes of exercise per week. This is not a particularly useful scenario, but it is a necessary mathematical construct that helps us to build the model.

We can extend the multiple regression model to have an arbitrary number of explanatory variables:

$$y_i = \beta_0 + \beta_1 x_i^{(1)} + \beta_2 x_i^{(2)} + ... + \beta_p x_i^{(p)} + \epsilon_i$$

Where:

$x_i^{(1)}, x_i^{(2)}, ..., x_i^{(p)}$ are the p explanatory variables and all else is as before.

or with summation notation:

$$y_i = \beta_0 + \sum_{j=1}^{p} \beta_j x_i^{(j)} + \epsilon_i$$

Just like in simple linear regression, we can estimate the parameters $\beta_0, \beta_1, ..., \beta_p$ using the method of least squares. The least squares method minimizes the sum of the squared residuals:

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

where $\hat{y}_i$ is the predicted value of the response variable for the ith observation:

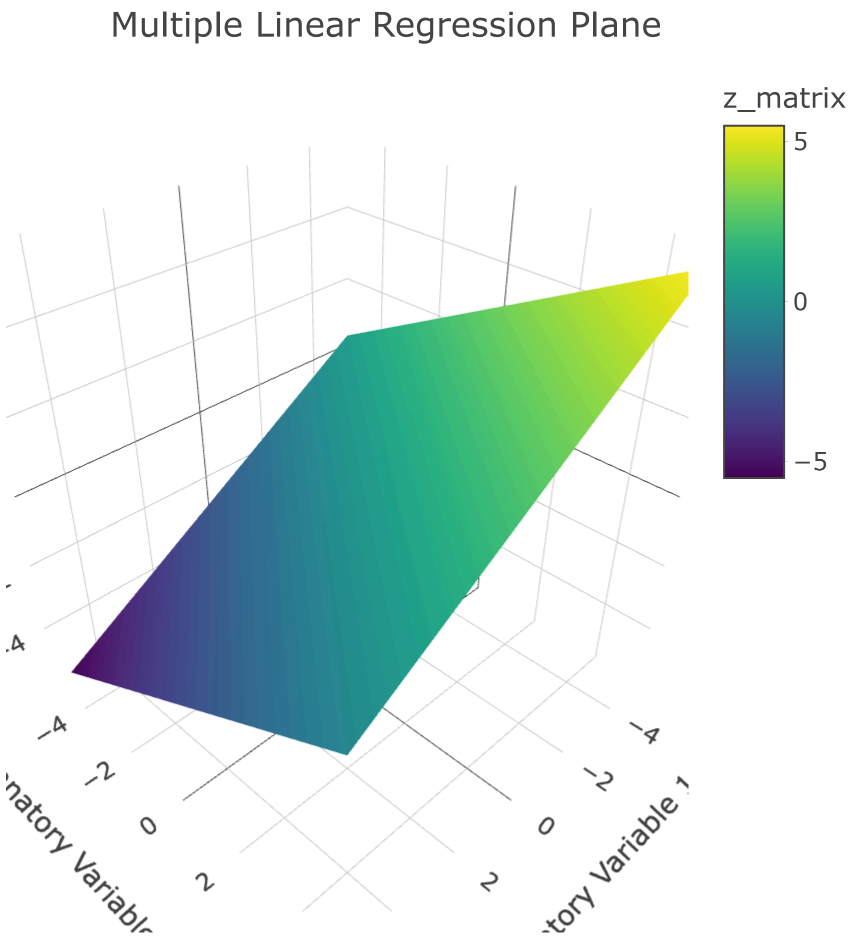
$$\hat{y}_i = \hat{\beta}_0 + \sum_{j=1}^{p} \hat{\beta}_j x_i^{(j)}$$

where:

$\hat{\beta}_0, \hat{\beta}_1, \dots, \hat{\beta}_p$ are the estimated parameters.

Think-Pair-Share (#tps-two-shape) Graphically, a linear regression with one explanatory variable is a line. What is a geometric representation of linear regression with two explanatory variables?

Here is a graph of the geometric representation of a multiple linear regression model with two explanatory variables (please note that this plot is best viewed in the HTML version of the book; in the PDF version, it will appear as a static image):



Let's write the equation for the blood pressure data:

$$bp_i = \beta_0 + \beta_1 \cdot age_i + \beta_2 \cdot mins\_exercise_i + \epsilon_i$$

where:

- bp_i is the blood pressure for the ith observation
- age_i is the age for the ith observation
- $mins\_exercise_i$ is the minutes of exercise for the ith observation
- β_0 is the intercept
- β_1 is the slope for age
- β_2 is the slope for minutes of exercise
- ϵ_i is the error term

and the error term is assumed to be normally distributed with mean 0 and constant variance, just as was the case for simple linear regression:

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

Seventh, we know how to make predictions using the model, and to make a prediction band.

What we don't know is how to answer the four questions already mentioned above:

- Question 1: As an ensemble (i.e., all together), are the explanatory variables "useful"?
- Question 2: Are each of the explanatory variables associated with the response?
- Question 3: What proportion of variability is explained?
- Question 4: Are some explanatory variables more important than others?

Let's answer these questions using the blood pressure example.

Fitting the model

We know how to estimate the parameters $\beta_0, \beta_1, \ldots, \beta_p$ using the method of least squares.

In R, we can fit a multiple linear regression model using the `lm()` function in a very similar way to the simple linear regression model. Here is the code for the blood pressure example. To fit two explanatory variables, we simply add the second variable to the formula using the `+` sign:

```r
m1 <- lm(bp ~ age + mins_exercise, data = bp_data_multreg)
```

Checking the assumptions

Great news –> the five assumptions of the multiple linear regression model are the same as for the simple linear regression model:

a) Normality of residuals.
b) Homoscedasticity = constant variance of residuals.
c) Independence of residuals.
d) Linearity.

e) No outliers.

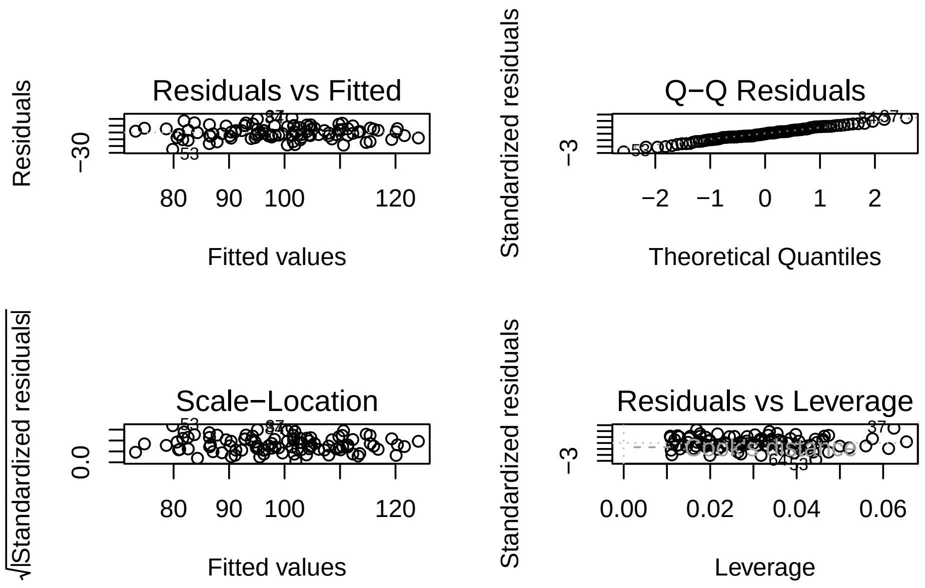
Think-Pair-Share (#tps-assump-match) Review how we can check these assumptions in the simple linear regression model. Match the following to the assumptions above:

1) Graph of size of residuals vs. fitted values.
2) QQ-plot.
3) Graph of residuals vs. fitted values.
4) Graph of leverage vs. standardized residuals.

And what is missing?

We can check the assumptions of the multiple linear regression model using the same methods as for the simple linear regression model. Here is the code for the blood pressure example:

```r
# Check the assumptions
par(mfrow=c(2,2))
plot(m1, which = c(1,2,3,5), add.smooth = FALSE)
```



We see that the assumptions are met for the blood pressure example:

a) Normality of residuals: The QQ-plot shows that the residuals are normally distributed.
b) Homoscedasticity: The scale-location plot shows that the residuals have constant variance.
c) Independence of residuals: No evidence of pattern or clustering. But also need to know about study design to properly assess independence.

d) Linearity: The residuals vs. fitted values plot shows no clear pattern in the residuals.
e) No outliers: No points with high leverage or high residuals.

Question 1: As an ensemble, are the explanatory variables useful?

Recall that when we learned about ANOVA we saw that a single categorical explanatory variable with multiple levels can be represented as multiple binary (0/1) explanatory variables. In that case, we used the F-test to test the null hypothesis of no effect / relationship for all binary variables together.

Likewise, when we have multiple continuous explanatory variables, we use the F-test to test the null hypothesis that **together** the explanatory variables have no association with the response variable. That is, we use the F-test to test the null hypothesis that the ensemble of explanatory variables is not associated with the response variable.

This corresponds to the same null hypothesis as we used in one-way ANOVA: The null hypothesis that the explained variance of the model is no greater than would be expected by chance. Here, "by chance" means that the slopes of the explanatory variables are zero:

$$H_0 : \beta_1 = \beta_2 = ... = \beta_p = 0$$

And the alternative hypothesis (just as in one-way ANOVA) is that the explained variance of the model is greater than would be expected by chance. This would occur if the slopes of some or all of the explanatory variables are not zero:

$$H_1 : \beta_1 \neq 0 \text{ or } \beta_2 \neq 0 \text{ or } ... \text{ or } \beta_p \neq 0$$

Recall that the F-test compares the variance explained by the model to the variance not explained by the model (i.e., the variance of the residuals). If the variance explained by the model is significantly greater than the variance not explained by the model, then we can conclude that the explanatory variables are associated with the response variable.

If we reject the null hypothesis, we can conclude that some combination of the explanatory variables is associated with the response variable. However, we cannot conclude which specific explanatory variables are associated with the response variable. To determine which specific explanatory variables are associated with the response variable, we need to perform individual t-tests for each explanatory variable. We will do this in the next section.

OK, back to the F-test.

We know a lot already from the ANOVA chapter. Let's review how we calculate the F-statistic.

The F-statistic is calculated as the ratio of two mean squares:

1. The mean square of the model (MSE_{model}).
2. The mean square of the residuals ($MSE_{residual}$).

Recall that a mean square is a sum of squares divided by the associated degrees of freedom. The formulas for these are the same as for ANOVA.

So, to calculate these two mean squares, we need to calculate three sums of squares:

1. The total sum of squares (SST).
2. The sum of squares of the model (SSM).
3. The sum of squares of the residuals (SSE).

We also need to calculate the degrees of freedom associated with each sum of squares.

1. The total degrees of freedom is $n-1$, where n is the number of observations.
2. The model degrees of freedom is p, where p is the number of explanatory variables. This is because for each explanatory variable we estimate one parameter (the slope), and each estimated parameter uses up one degree of freedom.
3. The residual degrees of freedom is $n - 1 - p$.

The F-statistic in R

We could do all these calculations ourselves (and you might be asked to in the exam), but also we can just ask R! For question 1 we need to know the F-statistic for the multiple linear regression model. We can easily get this from R using the `summary()` function, and by looking in the right place in the output:

```r
summary(m1)
```

```

Call:
lm(formula = bp ~ age + mins_exercise, data = bp_data_multreg)

Residuals:
     Min       1Q   Median       3Q      Max 
-24.8420  -4.7296  -0.1684   5.8146  21.6603 

Coefficients:
              Estimate Std. Error t value Pr(>|t|)    
(Intercept)   94.82139    3.30987  28.648  < 2e-16 ***
age            0.40000    0.05446   7.345 6.43e-11 ***
mins_exercise -0.10371    0.01024 -10.125  < 2e-16 ***
```

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

Residual standard error: 8.999 on 97 degrees of freedom
Multiple R-squared:  0.6222,    Adjusted R-squared:  0.6144 
F-statistic: 79.88 on 2 and 97 DF,  p-value: < 2.2e-16
```

In the final line of output we see "F-statistic: 79.88 on 2 and 97 DF, p-value: 3.14×10^{-21}".

The model degrees of freedom is 2 (because we have two explanatory variables), and the residual degrees of freedom is 97 (because we have 100 observations and 2 explanatory variables, so $100 - 1 - 2 = 97$).

So the `summary()` function gives us everything we need to answer question 1. It even gives us the p-value for the F-test.

How to report the result. We could write somthing like this: "The combination of age and minutes of exercise is significantly associated with blood pressure (F(2, 97) = 79.88, p = 3.14×10^{-21})." Note that this is rather an undesirable statement, because it focuses too much on the statistics and not enough on the science. Indeed, perhaps we care more about the association of each explanatory variable with blood pressure, which we will look at next.

Note

If you want to review how to calculate a p-value from an F-statistic, see the corresponding section of the one-way ANOVA chapter.

Question 2: Which variables are associated with the response?

As we did for simple linear regression, we can perform a t-test for one explanatory variable to determine if it is associated with the response. And we can do this for each of the explanatory variables. As before, the null hypothesis for each t-test is that the slope of the explanatory variable is zero. The alternative hypothesis is that the slope of the explanatory variable is not zero.

Here is the coefficients table, which includes the results of the t-tests for each explanatory variable:

```
summary(m1)$coef
```

```
                Estimate Std. Error    t value     Pr(>|t|)
(Intercept)   94.8213948 3.30987449  28.648033 3.966378e-49
age            0.3999966 0.05445864   7.344962 6.434757e-11
mins_exercise -0.1037103 0.01024311 -10.124887 7.207135e-17
```

And we can get the 95% CI for each slope estimate $\hat{\beta}_j$ as follows:

```r
confint(m1)
```

```
                    2.5 %       97.5 %
(Intercept)    88.2522102 101.39057945
age             0.2919113   0.50808197
mins_exercise -0.1240401  -0.08338059
```

Reminder: The 95% confidence interval is $[\hat{\beta} - c \cdot \sigma^{(\beta)}; \hat{\beta} + c \cdot \sigma^{(\beta)}]$, where c is the 97.5% quantile of the t-distribution with $n - p$ degrees of freedom).

! Important

However Please insert a note into your brain that we are dealing here with an ideal case of **uncorrelated explanatory variables**. You'll learn later in the course about what happens when explanatory variables are correlated. Hint: interpretation is difficult and unstable!

Think–Pair–Share (#a_partial_effects) What does "holding other variables constant" mean here? Is this always biologically plausible?

Question 3: What proportion of variability is explained?

Multiple R^2

We can calculate the R^2 value for the multiple linear regression model just like we already did for a simple linear regression model. The R^2 value is the proportion of variability in the response variable that is explained by the model. As before, the R^2 value ranges from 0 to 1, where 0 indicates that the model does not explain any variability in the response variable, and 1 indicates that the model explains all the variability in the response variable.

For multiple linear regression, we often use the term "multiple R^2" to distinguish it from the R^2 value for simple linear regression. The multiple R^2 is the proportion of variability in the response variable that is explained by the model, taking into account all the explanatory variables in the model.

As before, for simple linear regression, the multiple R^2 value is calculated as the sum of squares explained by the model divided by the total sum of squares:

$$R^2 = \frac{SSM}{SST}$$

```r
sss <- anova(m1)
SSM <- sss$`Sum Sq`[1] + sss$`Sum Sq`[2]
```

```r
SST <- sum(sss$`Sum Sq`)
R_squared <- SSM / SST
#R_squared
```

where SSM is the sum of squares explained by the model and SST is the total sum of squares, and $SSM = SST - SSE$.

For the blood pressure data:

```r
summary(m1)$r.squared
```

```
[1] 0.6222147
```

R^2 for multiple linear regression can also be calculated as the squared correlation between $(y_1, \ldots, y_n)$ and $(\hat{y}_1, \ldots, \hat{y}_n)$, where the $\hat{y}$ are the fitted values from the model. The fitted values are calculated as:

$$\hat{y}_i = \hat{\beta}_0 + \hat{\beta}_1 x^{(1)} + \ldots + \hat{\beta}_m x^{(m)}$$

In R:

```r
r_squared <- cor(m1$fitted.values, bp_data_multreg$bp)^2
r_squared
```

```
[1] 0.6222147
```

Or:

```r
sss <- anova(m1)
SSM <- sss$`Sum Sq`[1] + sss$`Sum Sq`[2]
SST <- sum(sss$`Sum Sq`)
R_squared <- SSM / SST
R_squared
```

```
[1] 0.6222147
```

Adjusted R^2

However, we have a little problem to address. The R^2 value increases as we add more explanatory variables to the model, even if the additional variables are not associated with the response. This is because the R^2 value is calculated as the proportion of variability in the response variable that is explained by the model. As we add more explanatory variables to the model, the model will always explain more variability in the response variable, even if the additional variables are not associated with the response. Some of the variance will be explained by chance.

Here is an example of this problem. First, here's the explanatory power of the model with only age and minutes of exercise as the explanatory variables:

```
m1 <- lm(bp ~ age + mins_exercise, data = bp_data_multreg)
summary(m1)$r.squared
```

```
[1] 0.6222147
```

Now, we can add a new explanatory variable to the blood pressure model that is not associated with the response:

```
bp_data_multreg$random_variable <- rnorm(nrow(bp_data_multreg))
m2 <- lm(bp ~ age + mins_exercise + random_variable, data = bp_data_multreg)
summary(m2)$r.squared
```

```
[1] 0.6224261
```

The R^2 value for the model with the random variable is higher than the R^2 value for the model without the random variable. This is because the model with the random variable explains more variability in the response variable, even though the random variable is not associated with the response.

To address this problem, we can use the adjusted R^2 value. The adjusted R^2 value is calculated as:

$$R^2_{\text{adj}} = 1 - \frac{SSE/(n-p-1)}{SST/(n-1)}$$

where * SSE is the sum of squared errors * SST is the total sum of squares * n is the number of observations * p is the number of explanatory variables in the model.

Or put another way:

$$R^2_{adj} = 1 - (1 - R^2)\frac{n-1}{n-p-1}$$

In this form, we can see that as p increases (as we add explanatory variables) the term $(n-1)/(n-p-1)$ increases, and the adjusted R^2 value will decrease if the additional variables are not associated with the response.

Take home: when we want to compare the explanatory power of models that differ in the number of explanatory variables, we should use the adjusted R^2 value.

Question 4: Are some explanatory variables more important than others?

We know how to test the significance of the parameters using the t-test. We also know how to calculate the confidence intervals for the parameters, and to make a confidence band.

How important are the explanatory variables and how important are they relative to each other?

Think-Pair-Share (#tps-variable-importance) How might we assess how important is each of the explanatory variables, and how important they are relative to each other?

The importance of an explanatory variable can be assessed by looking at the size of the coefficient for that variable. The larger the coefficient, the more important the variable is in explaining the response variable.

It is, however, important to remember that the size of the coefficient depends on the scale of the explanatory variable. If the explanatory variables are on different scales, then the coefficients will be on different scales and cannot be directly compared.

In our example, the age variable is measured in years, so the coefficient is in units mmHg (pressure) per year. The mins_exercise variable is measured in minutes, so the coefficient is in units mmHg per minute. The coefficients are on different scales and cannot be directly compared. Furthermore, the value of the coefficients would change if we measured age in months or minutes of exercise in hours.

There are other perspectives we can take when we're assessing importance. For example, we cannot change our age, but we can change the number of minutes of exercise. So, the practical importance of the two variables is quite different in that sense also.

To compare the importance of the explanatory variables that are measured on different scales, we can standardize the variables before fitting the model. This means that we subtract the mean of the variable and divide by the standard deviation. This puts all the variables on the same scale, so the coefficients can be directly compared. The coefficients are then in units of the response variable per standard deviation of the explanatory variable.

However, the coefficients are then not in the original units of the explanatory variables, so it is not always easy to interpret the coefficients. So while we can compare the coefficients, they have lost a bit of their original meaning and are not so easy to interpret.

One way to relate the coefficients in this case is to realise that to compensate for the blood pressure increase associated with one year of age, one would need to exercise for a certain number of minutes more.

Think-Pair-Share (#tps-exercise-age) How many minutes of exercise per week would we need to add to our fitness schedule to compensate for the blood pressure increase associated with one year of age?

```
extra_mins_exercise <- coef(m1)["age"] / -coef(m1)["mins_exercise"]
#extra_mins_exercise
```

Question 5: How do we make predictions?

We already made predictions from a simple linear regression model?

Recall that the equation for multiple linear regression is:

$$y_i = \beta_0 + \beta_1 x_i^{(1)} + \beta_2 x_i^{(2)} + \ldots + \beta_p x_i^{(p)} + \epsilon_i$$

Therefore to get a predicted value of y_i, we can use the estimated parameters $(\hat{\beta}_0, \hat{\beta}_1, \ldots, \hat{\beta}_p)$ and the values of the explanatory variables $(x_i^{(1)}, x_i^{(2)}, \ldots, x_i^{(p)})$:

$$\hat{y}_i = \hat{\beta}_0 + \hat{\beta}_1 x_i^{(1)} + \hat{\beta}_2 x_i^{(2)} + \ldots + \hat{\beta}_p x_i^{(p)}$$

In R, we can use the `predict()` function to make predictions from a multiple linear regression model. Here is an example of how to make predictions for the values of the explanatory variables in the original dataset:

```r
predictions <- predict(m1)
```

Where `m1` is the multiple linear regression model fitted earlier. The `predict` function automatically uses the original data because we did not provide any new data.

We can also make predictions for new values of the explanatory variables by providing a new data frame to the `predict()` function. Here is an example of how to make predictions for new values of age and minutes of exercise:

First we need to make some new values of age and minutes of exercise:

```r
new_data <- data.frame(age = c(30, 40, 50),
                       mins_exercise = c(50, 100, 150))
```

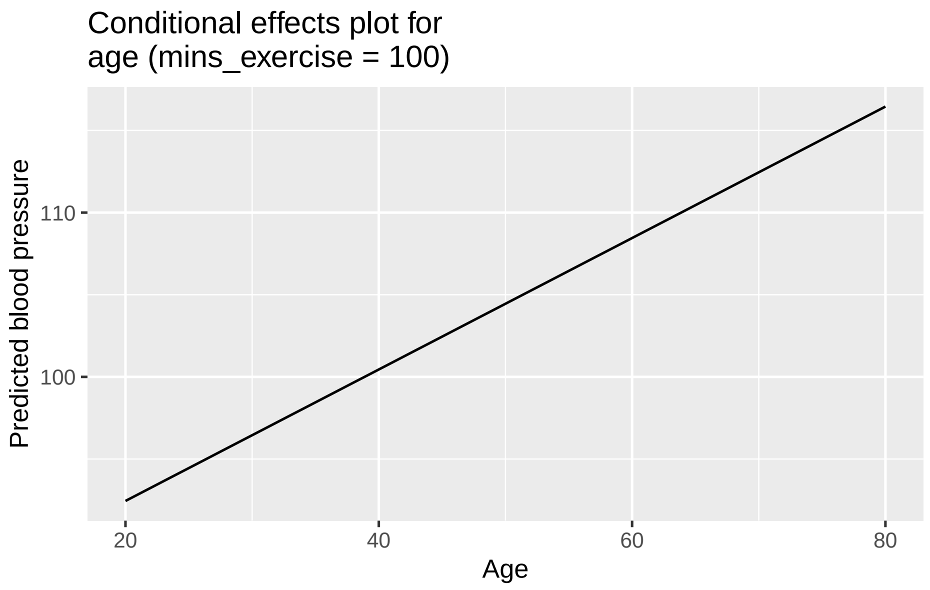
Then we can use the `predict()` function to make predictions for these new values:

```r
new_predictions <- predict(m1, newdata = new_data)
```

> **!** Important
>
> The new data frame must have the all the explanatory variables used in the model, and the variable names must match exactly those used in the model.

We can take this to the next level and make what is called a “conditional effects plot” or “effect plot”. This is a plot that shows the predicted values of the response variable for different values of one explanatory variable, while holding the other explanatory variables constant. Here is an example of how to make a conditional effects plot for age, while holding minutes of exercise constant at 100:

```
new_data_effects <- data.frame(age = seq(20, 80, by = 1),
                               mins_exercise = 100)
new_data_effects <- new_data_effects |>
  mutate(new_predictions_effects = predict(m1, newdata = new_data_effects))
ggplot(data = new_data_effects, aes(x = age, y = new_predictions_effects)) +
  geom_line() +
  labs(x = "Age", y = "Predicted blood pressure") +
  ggtitle("Conditional effects plot for\nage (mins_exercise = 100)")
```



And lets take this to the next level again and make a conditional effects plot for three different levels of `mins_exercise`:

```
new_data_effects_3 <- expand.grid(age = seq(20, 80, by = 1),
                                  mins_exercise = c(0, 150, 300))
```

A new function! And it's one of Owen's favourites: `expand.grid()`. This function creates a data frame from all combinations of the supplied vectors or factors. Here, we are creating a data frame with all combinations of age (from 20 to 80) and minutes of exercise (0, 150, and 300).

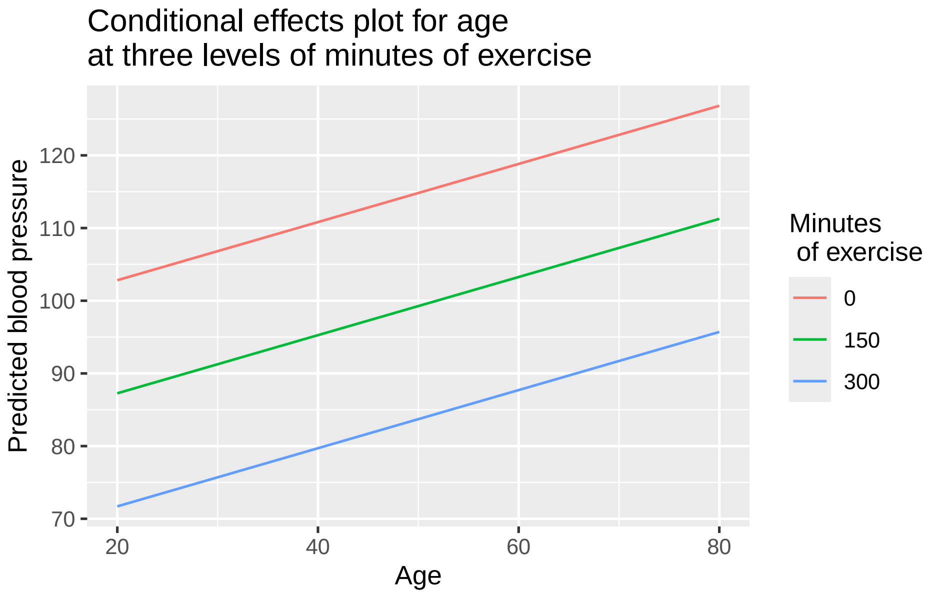
We then give the new data frame to the `predict()` function to get the predicted values for each combination of age and minutes of exercise:

```
new_data_effects_3 <- new_data_effects_3 |>
  mutate(prediction = predict(m1, newdata = new_data_effects_3),
         mins_exercise_fac = as.factor(mins_exercise))
```

(Note that we changed)

And make a graph:

```
ggplot(data = new_data_effects_3, aes(x = age, y = prediction,
                                                  col = mins_exercise_fac)) +
  geom_line() +
  labs(x = "Age", y = "Predicted blood pressure", color = "Minutes
 of exercise") +
  ggtitle("Conditional effects plot for age\nat three levels of minutes of exercise")
```



Collinearity

In the blood pressure example data used previously in this chapter there is no evidence of correlation between the two explanatory variables. However, in practice, it is common for explanatory variables to be correlated with each other. This is known as collinearity. It can be quite problematic for us!

Think-Pair-Share (#tps-collinearity1) Imagine if there was perfect correlation between age and minutes of exercise. This would mean that a graph of age vs minutes of exercise would show a perfect line. What would be the implications for the multiple linear regression model? For example, what would be the R^2 value for a model with one explanatory variable compared to a model with both explanatory variables?

Collinearity, specifically *harmful* collinearity, is extremely common in real dataset that result from observational studies. This is because in observational studies there are often numerous explanatory variables, and they are often correlated with each other. This is a situation that is ripe for collinearity problems. (Collinearity can also happen in data resulting from designed manipulative experiments, but is hopefully relatively rare there because a well-designed experiment will try to avoid collinearity by ensuring that the explanatory variables are independent.)

So what is collinearity, and why is it a problem?

Put simply, collinearity is when one explanatory variable is predictable from a linear combination of others.

This can happen due to strong correlation among pairs of explanatory variables, or due to more complex relationships involving three or more explanatory variables.

For example, if we have three explanatory variables, x_1, x_2, and x_3, and if x_3 can be predicted from a linear combination of x_1 and x_2, then we have collinearity. For example, if:

$$x_3 = 2 \cdot x_1 + 3 \cdot x_2 + \text{small random noise}$$

In this case, variable x_3 is a linear combination of x_1 and x_2, plus some small random noise. This means that if we know the values of x_1 and x_2, we can predict the value of x_3 quite accurately. Also, in this case we might not have strong correlation between any pair of the explanatory variables, but there is still collinearity because x_3 is predictable from x_1 and x_2. So lack of correlation between pairs of explanatory variables does not guarantee that there is no collinearity.

It is a problem because it makes the slope estimates unstable and therefore difficult to interpret.

Let us see this instability in practice. First let's look at the really extreme example of perfect collinearity. Here's a new version of the blood pressure data in which the minutes of exercise variable is perfectly predicted from age:

```r
set.seed(123)
n <- 100
age <- ceiling(runif(n, 20, 80))
```

Here is the line of code that makes the perfect correlation between age and minutes of exercise:

```r
mins_exercise <- 100 - age
```

Now we generate the blood pressure variable as before:

```r
bp <- 100 + 0.5 * age + rnorm(n, 0, 15)
bp_data_perfect <- data.frame(age = age,
                              mins_exercise = mins_exercise,
                              bp = bp)
```

Read in the data:

```
bp_data_perfect <- read_csv("datasets/bp_data_perfect.csv")
```

```
Rows: 100 Columns: 3
-- Column specification --------------------------------------------------------
Delimiter: ","
dbl (3): age, mins_exercise, bp

i Use `spec()` to retrieve the full column specification for this data.
i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

Now we fit a multiple linear regression model with only age:

```
m1_age <- lm(bp ~ age, data = bp_data_perfect)
summary(m1_age)
```

```


Call:
lm(formula = bp ~ age, data = bp_data_perfect)

Residuals:
    Min      1Q  Median      3Q     Max 
-33.546  -9.147  -0.327   8.938  33.213 

Coefficients:
             Estimate Std. Error t value Pr(>|t|)    
(Intercept) 100.43331    4.54751  22.085  < 2e-16 ***
age           0.47541    0.08549   5.561 2.33e-07 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 14.54 on 98 degrees of freedom
Multiple R-squared:  0.2399,	Adjusted R-squared:  0.2321 
F-statistic: 30.92 on 1 and 98 DF,  p-value: 2.325e-07
```

The estimated slope (coefficient) is 0.48, which is close to the true value of 0.5. The 95% confidence interval is 0.31 to 0.65, which includes the true value of 0.5. All good then.

Now we fit the multiple linear regression model with both age and minutes of exercise included:

```
m1_both <- lm(bp ~ mins_exercise + age, data = bp_data_perfect)
summary(m1_both)
```

```
Call:
lm(formula = bp ~ mins_exercise + age, data = bp_data_perfect)

Residuals:
    Min      1Q  Median      3Q     Max 
-33.546  -9.147  -0.327   8.938  33.213 

Coefficients: (1 not defined because of singularities)
               Estimate Std. Error t value Pr(>|t|)    
(Intercept)   147.97398    4.48276  33.010  < 2e-16 ***
mins_exercise  -0.47541    0.08549  -5.561 2.33e-07 ***
age                  NA         NA      NA       NA    
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 14.54 on 98 degrees of freedom
Multiple R-squared:  0.2399,    Adjusted R-squared:  0.2321 
F-statistic: 30.92 on 1 and 98 DF,  p-value: 2.325e-07
```

This output is a bit strange. The estimate for age is now `NA`. This is because the model cannot distinguish between the effects of age and minutes of exercise, since they are perfectly correlated. The model is unable to estimate the coefficient for age of exercise, so it returns `NA`.

This is an example of instability of coefficients due to collinearity. The coefficient for age is completely unstable, as it changes from a number to `NA` depending on whether minutes of exercise is included in the model or not.

Actually, you can see that the estimate for `mins_exercise` are the same (except the sign of the coefficient) as the estimate for age in the previous model. This is because minutes of exercise is perfectly correlated with age, so the model is essentially using minutes of exercise as a proxy for age. With perfect collinearity, the model cannot distinguish between the effects of the two variables… they are effectively identical.

You can also see that the R^2 value doesn't change when the second variable is added. That is, the model with only age included is identical to the model with both age and minutes of exercise included. This is because minutes of exercise is perfectly correlated age, so including minutes of exercise in the model does not add any new information.

That is a pretty extreme example of perfect collinearity. In practice, collinearity is often not perfect, but still strong enough to cause problems.

Let's look at the less extreme example of collinearity we had earlier, with three explanatory variables, x_1, x_2, and x_3, and where x_3 can be predicted from a linear combination of x_1 and x_2 plus some random noise:

```
set.seed(123)
n <- 100
x1 <- rnorm(n)
x2 <- rnorm(n)
x3 <- 2 * x1 + 3 * x2 + rnorm(n, 0, 5)
y <- 5 + 1.5 * x1 - 2 * x2 + 0.5 * x3 + rnorm(n, 0, 1)
data_collinear <- data.frame(x1 = x1, x2 = x2, x3 = x3, y = y)
cor(data_collinear)
```

```
            x1          x2        x3          y
x1  1.00000000 -0.04953215 0.1877534  0.6079515
x2 -0.04953215  1.00000000 0.5194100 -0.1488812
x3  0.18775342  0.51940999 1.0000000  0.6434430
y   0.60795153 -0.14888117 0.6434430  1.0000000
```

We see that there is not strong correlation between any pair of the explanatory variables, but there is still collinearity because x_3 is predictable from x_1 and x_2.

Let's look for evidence of instability of the coefficients. First, we fit the multiple linear regression model with all three explanatory variables included:

```
m_collinear_123 <- lm(y ~ x1 + x2 + x3, data = data_collinear)
summary(m_collinear_123)$coefficients
```

```
              Estimate Std. Error   t value     Pr(>|t|)
(Intercept)  4.9806739 0.10734366  46.39933 1.490805e-67
x1           1.4675042 0.11972274  12.25752 2.384482e-21
x2          -1.9193496 0.12990373 -14.77517 1.832330e-26
x3           0.4885226 0.02244616  21.76419 6.723265e-39
```

Do these change much if we fit the model with only x_1 and x_2 included?

```
m_collinear_12 <- lm(y ~ x1 + x2, data = data_collinear)
summary(m_collinear_12)$coefficients
```

```
              Estimate Std. Error   t value     Pr(>|t|)
(Intercept)  5.3105865  0.2575325 20.621034 3.113931e-37
x1           2.1192629  0.2809162  7.544111 2.464756e-11
x2          -0.3956201  0.2651792 -1.491897 1.389714e-01
```

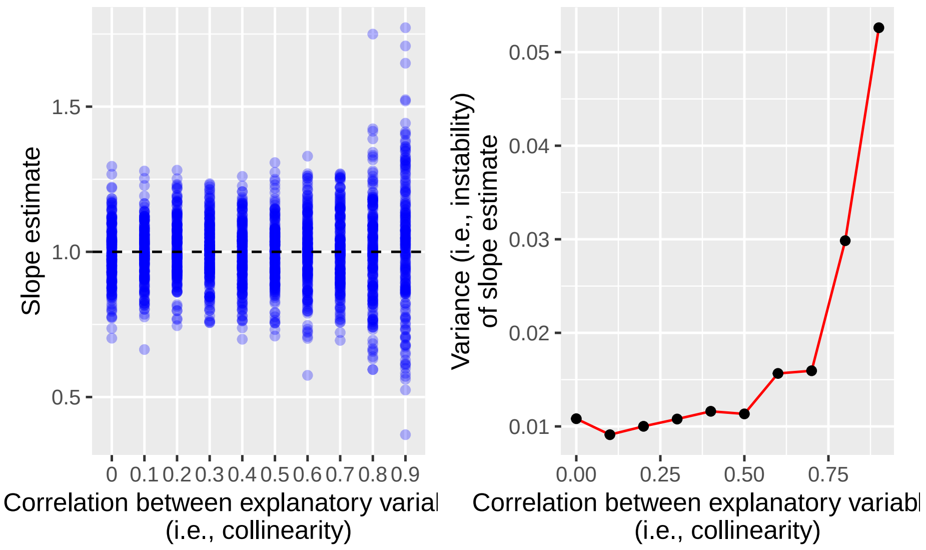
Yes, they do change quite a bit. The coefficients for x_1 and x_2 are quite different when x_3 is included in the model compared to when it is not included. This is because x_3 is predictable from x_1 and x_2, so including x_3 in the model changes the interpretation of the coefficients for x_1 and x_2.

Note

In a not so extreme case such as this one, the coefficients will not be completely unstable (i.e., they will not change from a number to NA) but they can still change quite a bit depending on which other collinear variables are included in the model.

The bottom line is that collinearity between explanatory variables complicates the interpretation of the model coefficients. If there is collinearity/correlation between the explanatory variables, then the model coefficients can be unstable and difficult to interpret.

Let's have one more look at this instability that is caused by collinearity. And make this demonstration a bit more general. We'll simulate data with two explanatory variables that are correlated with each other to varying degrees. We'll then fit multiple linear regression models with both explanatory variables included, and see how the stability of the coefficients changes as we change the correlation between the explanatory variables.



In the left panel we see the slope estimates for x_1 from the multiple linear regression model with both x_1 and x_2 included, for different levels of correlation between x_1 and x_2. When there is no correlation between x_1 and x_2 (i.e., no collinearity), the slope estimates are quite stable and close to the true value of 1.

As the correlation between x_1 and x_2 increases, the slope estimates become more variable (= less stable). At correlation of 0.9, the slope estimates are quite variable and can be very far from the true value of 1.

The right panel shows the variance of the slope estimates as a function of the correlation between x_1 and x_2. Variance here is a measure of the amount of vertical spread in the left panel.

We see that the variance of the slope estimates increases as the correlation between x_1 and x_2 increases. This shows that greater collinearity between explanatory variables leads to greater instability (variance) of the slope estimates.

This increase in variance caused by collinearity is known as the **variance inflation effect**. The variance inflation effect makes it difficult to interpret the coefficients of the model, because the coefficients can be quite unstable and can change a lot depending on which other collinear variables are included in the model.

Collinearity and interpretation of R^2

Collinearity also affects the interpretation of the R^2 values. Collinearity will cause the collinear explanatory variables to share some of the explained variance. The R^2 value of the multiple regression will then be less than the sum of the R^2 values of the individual regressions of the response variable on each of the explanatory variables separately.

R^2 of the age only model:

```
summary(m2_age)$r.squared
```

```
[1] 0.2229566
```

R^2 of the mins_exercise only model:

```
summary(m2_mins_exercise)$r.squared
```

```
[1] 0.4121019
```

R^2 of the model with both age and mins_exercise:

```
summary(m2_both)$r.squared
```

```
[1] 0.6222147
```

In this case the two explanatory variables are strongly correlated and so share a lot of the explained variance. The R^2 value of the model with both explanatory variables is much less than the sum of the R^2 values of the models with each explanatory variable separately. In fact, either of the models with only one explanatory variable is nearly as good as the model with both explanatory variables. We don't gain much from including another explanatory variable in the model when we already include one explanatory variable that is strongly correlated with the other.

Think–Pair–Share (#a_collinearity_intuition) Why can strong correlation among explanatory variables make estimates unstable even if the model fits the data well?

Do I have a problem (with collinearity)?

There are several ways to measuring collinearity between explanatory variables and to assess if it is a problematic. One way is to look at the correlation matrix of the explanatory variables. If there are strong correlations between any pair of explanatory variables, then there is likely to be collinearity.

But recall that collinearity can also occur without strong pairwise correlations. So another way to detect collinearity is to calculate the Variance Inflation Factor (VIF). This is so named because it measures how much the variance of the estimated regression coefficients is increased due to collinearity. (Recall that we saw the variance inflation effect in the simulation above.)

Recall the definition of collinearity: it is when an explanatory variable is predictable from a linear combination of others. To calculate the VIF for a specific explanatory variable, we fit a linear regression model with that explanatory variable as the response variable, and all the other explanatory variables as the explanatory variables. We then calculate the R^2 value for this model. The VIF is then calculated as:

$$VIF_j = \frac{1}{1 - R_j^2}$$

where R_j^2 is the R^2 value from the model with explanatory variable j as the response variable, and all other explanatory variables as the explanatory variables.

In the case where the explanatory variable j is not predictable from the other explanatory variables at all, then $R_j^2 = 0$, and the VIF is 1. This indicates that there is no collinearity.

In the case where the explanatory variable j is perfectly predictable from the other explanatory variables, then $R_j^2 = 1$, and the VIF is infinite. This indicates that there is perfect collinearity.

To get the VIF for a multiple linear regression model with multiple explanatory variables, we calculate the VIF for each explanatory variable separately.

The VIF measures how much the variance of the estimated regression coefficients is increased due to collinearity.

In R, we can calculate the VIF using the `vif()` function from the `car` package. Here is the code to calculate the VIF for the blood pressure model:

```r
vif(m1)
```

```
          age mins_exercise 
      1.00047       1.00047 
```

We see that the VIF values for both explanatory variables are close to 1, indicating that there is no collinearity between the explanatory variables.

For the previous example with collinearity among three explanatory variables, we can calculate the VIF as follows:

```r
vif(m_collinear_123)
```

```
      x1       x2       x3 
1.069365 1.412832 1.460863 
```

We see that the VIF values for all three explanatory variables are greater than 1, indicating that there is collinearity between the explanatory variables. The VIF for x_3 is highest, indicating that x_3 is predictable from x_1 and x_2. But none of the VIF values are extremely high, indicating that the collinearity is not severe.

A VIF value greater than 5 or 10 is often used as a rule of thumb to indicate that there is collinearity between the explanatory variables.

Let's make an example of three explanatory variables with more severe collinearity:

```r
set.seed(123)
n <- 100
x1 <- rnorm(n)
x2 <- rnorm(n)
x3 <- 2 * x1 + 3 * x2 + rnorm(n, 0, 1)  # less noise, more collinearity
y <- 5 + 1.5 * x1 - 2 * x2 + 0.5 * x3 + rnorm(n, 0, 1)
data_collinear_severe <- data.frame(x1 = x1, x2 = x2, x3 = x3, y = y)
m_collinear_severe <- lm(y ~ x1 + x2 + x3, data
 = data_collinear_severe)
vif(m_collinear_severe)
```

```
       x1        x2        x3 
 4.277416 10.644760 13.360572 
```

We see that the VIF values of x_2 and x_3 are greater than 10, indicating that there is strong collinearity between the explanatory variables. The coefficients for x_2 and x_3 will be quite unstable and difficult to interpret.

What to do about collinearity?

Imagine we see high VIFs, we can conclud that there is collinearity between explanatory variables, and that the coefficients for those variables are likely to be unstable and difficult to interpret. What can we do???

The answer is we should *think*!

- Are explanatory variables measuring the same concept?
- Is this collinearity expected from the study design?
- Do I care about interpretation, or am I only interested in prediction?

And then consider some possible remedies:

- Combine explanatory variables (index, PCA, biologically meaningful composite) (covered later in the course).
- Look at the explanatory power of a variable once all other variables are accounted for (see below).
- Remove one of a set of redundant explanatory variables (not covered in this course).
- Use regularization if prediction is the goal (not covered in this course).

Assessing the importance of an explanatory variables in the presence of collinearity

Imagine that we want to know the importance of an explanatory variable in the presence of collinearity. For example, we might want to know how important age is in explaining blood pressure. But we also know that we have collinearity with other explanatory variables, such as minutes of exercise. How can we assess the importance of a particular explanatory variable in the presence of collinearity? Let us call the explanatory variable of interest the "focal" explanatory variable.

One way to assess the importance of the focal explanatory variable is to compare two models. Both should have all other explanatory variables included, but one model should have the focal explanatory variable included, and the other model should have the focal explanatory variable removed. We can then make an F-test to compare the two models. The null hypothesis is that the focal explanatory variable does not explain any additional variance in the response variable, once all other explanatory variables are accounted for. The alternative hypothesis is that the focal explanatory variable does explain additional variance in the response variable, once all other explanatory variables are accounted for.

The question is essentially: does including the focal explanatory variable improve the model fit, once all other explanatory variables are already accounted for?

Here is an example of how to do this in R. Lets use the data with three explanatory variables, x_1, x_2, and x_3, where x_3 is predictable from a linear combination of x_1 and x_2 plus some random noise. We will assess the importance of x_1 in explaining the response variable y, once x_2 and x_3 are accounted for. First, we fit the full model with all three explanatory variables included:

```r
m_full <- lm(y ~ x1 + x2 + x3, data = data_collinear)
```

Then, we fit the reduced model with x_1 removed:

```r
m_reduced <- lm(y ~ x2 + x3, data = data_collinear)
```

Now, we can use the `anova()` function to compare the two models:

```r
anova(m_reduced, m_full)
```

```
Analysis of Variance Table
```

```
Model 1: y ~ x2 + x3
Model 2: y ~ x1 + x2 + x3
  Res.Df    RSS Df Sum of Sq      F    Pr(>F)
1     97 272.27
2     96 106.14  1    166.12 150.25 < 2.2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

In the R output we see information about two models.

- **Model 1** is the reduced model with x_1 removed.
- **Model 2** is the second model is the full model with all three explanatory variables included.

Then we have a table that looks somewhat like an ANOVA table. The first column has no title and contain 1 and 2. This indicates the two models being compared.

The second column is **Res.Df**, which is the residual degrees of freedom for each model. We can see that the residual degrees of freedom for Model 2 is one less than that for Model 1, because Model 2 has one more explanatory variable (i.e., x_1) than Model 1.

The third column is "RSS", which is the Residual Sum of Squares for each model. The residual sum of squares for Model 2 is less than that for Model 1, because Model 2 has one more explanatory variable (i.e., x_1) than Model 1, and because that explanatory variable helps to explain some of the variance in the response variable.

The fourth column is "Df", which is the difference in degrees of freedom between the two models. This is one in this case, because Model 2 has one more explanatory variable than Model 1.

The fifth column is "Sum of Sq", which is the difference in residual sum of squares between the two models. This the amount of variance in the response variable that is explained by the focal explanatory variable (i.e., x_1) once all other explanatory variables are already accounted for.

The sixth column is "F value", which is the F-statistic for the comparison of the two models. This is calculated as:

$$F = \frac{(RSS_1 - RSS_2)/(df_1 - df_2)}{RSS_2/df_2}$$

where RSS_1 and RSS_2 are the residual sum of squares for Model 1 and Model 2, respectively, and df_1 and df_2 are the residual degrees of freedom for Model 1 and Model 2, respectively.

Put another way, the numerator is the mean square difference between the two models, and the denominator is the mean square error for the full model.

In our example data, the numbers are: 272.3 - 106.1 = 166.1 divided by - 96 = 1 for the numerator.

And for the denominator: 106.1 divided by 96.

This gives an F value of 150.25. This is a very large F value, indicating that the focal explanatory variable (i.e., x_1) explains a significant amount of variance in the response variable, even once all other explanatory variables are already accounted for. This means that even though there is collinearity between the explanatory variables, we can still find that x_1 is important in explaining the response variable.

We can also find the partial R^2 value for the focal explanatory variable (i.e., x_1) from this analysis. The partial R^2 value is the proportion of variance in the response variable that is explained by the focal explanatory variable, as a fraction of the variance that is not explained by the other explanatory variables.

The partial R^2 is calculated as:

$$R^2_{partial} = \frac{RSS_1 - RSS_2}{RSS_1}$$

We can find this in R with:

```r
rss1 <- anova(m_reduced, m_full)$"RSS"[1]
rss2 <- anova(m_reduced, m_full)$"RSS"[2]
partial_r2 <- (rss1 - rss2) / rss1
partial_r2
```

```
[1] 0.6101473
```

Don't confuse this partial R^2 with what is sometimes known as the semi-partial R^2. The semi-partial R^2 is the proportion of variance in the response variable that is explained by the focal explanatory variable, as a fraction of the total variance in the response variable. The semi-partial R^2 is calculated as:

$$R^2_{semi-partial} = \frac{RSS_1 - RSS_2}{TSS}$$

where TSS is the total sum of squares of the response variable.

In R in our example this is:

```r
tss <- sum((data_collinear$y - mean(data_collinear$y))^2)
semi_partial_r2 <- (rss1 - rss2) / tss
semi_partial_r2
```

```
[1] 0.1625303
```

This means that about 16.25% of the total variance in the response variable is explained by the focal explanatory variable (i.e., x_1), even once all other explanatory variables are already accounted for. This is explanatory power that is unique to the focal explanatory variable.

Think–Pair–Share (#a_interpreting_coefficients_conditionally) How does the interpretation of a coefficient change when you add another explanatory variable to the model?

Review

Simple regression:

- How well does the model describe the data: Correlation and R^2
- Are the parameter estimates compatible with some specific value (t-test)?
- What range of parameters values are compatible with the data (confidence intervals)?
- What regression lines are compatible with the data (confidence band)?
- What are plausible values of other data (prediction band)?

Multiple regression:

- Multiple linear regression x_1, x_2, ..., x_m
- Checking assumptions.
- Question 1: As an ensemble, do the explanatory variables explain variation in the response variable? This is done with an overall F-test.
- Question 2: Which of the explanatory variables are important in explaining variation in the response variable? This is done by looking at the coefficients, t-tests, and confidence intervals for each explanatory variable.
- Question 3: How well does the model describe the data? This is done with R^2 and adjusted R^2.
- Question 4: Are some explanatory variables more important than others in explaining variation in the response variable? This is done by looking at the sizes of the coefficients, and comparing those for different explanatory variables. But caution is required.
- Question 5: How do we make predictions from the model? This is done with the `predict()` function, and often requires fixing the values of some explanatory variables to specific values.
- Collinearity between explanatory variables can make the coefficients unstable and difficult to interpret. This can be assessed with the Variance Inflation Factor (VIF). Remedies include combining explanatory variables, removing redundant predictors, or using regularization if prediction is the goal. The importance of an explanatory variable in the presence of collinearity can be assessed by comparing models with and without that variable, using an F-test.

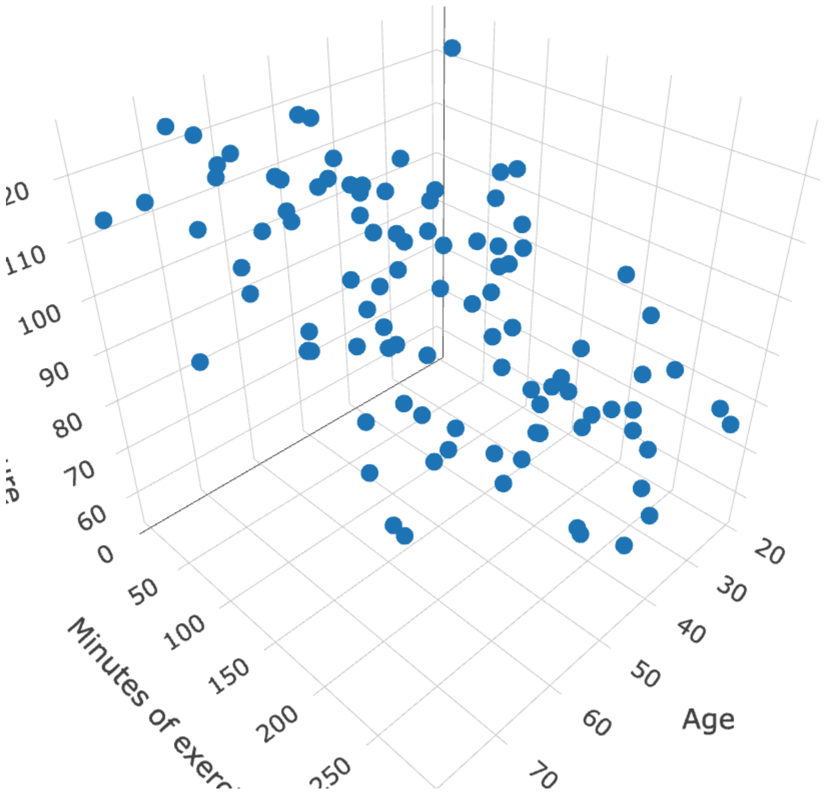
Extras

3D plot of multiple linear regression

3D plots can help us to visualise multiple linear regression models with two explanatory variables. They are also kind of cool. They can also be difficult to interpret. So use with caution!

Here is the code to make a 3D plot of the blood pressure data. The y-axis is blood pressure, the x-axis is age, and the z-axis is minutes of exercise. Here is a 3d plot that we can interactive with and rotate (please note that this plot is best viewed in the HTML version of the book; in the PDF version, it will appear as a static image):

```r
plotly::plot_ly(bp_data_multreg, x = ~age, y = ~mins_exercise, z = ~bp,
                type = "scatter3d", mode = "markers", marker = list(size = 5),
                width=500, height=500) |>
  plotly::layout(scene = list(xaxis = list(title = "Age"),
                             yaxis = list(title = "Minutes of exercise"),
                             zaxis = list(title = "Blood pressure")))
```



Publication ready table of results

Sometimes we need to put a table of coefficients and confidence intervals into a report or publication. One option is to make a table in Word, and to manually enter the coefficients and confidence intervals. This is tedious and error prone. A much better option is to use R to make the table for us. One way to do this is to use the `broom` package to tidy up the model output into a data frame, and then use the `knitr` package to make a table from the data frame.

```r
tidy_m1 <- tidy(m1, conf.int = TRUE)
kable(tidy_m1, digits = 3, caption = "Multiple linear regression results
for blood pressure data")
```

Table 2: Multiple linear regression results for blood pressure data

term	estimate	std.error	statistic	p.value	conf.low	conf.high
(Intercept)	94.821	3.310	28.648	0	88.252	101.391

Characteristic	**Beta**	**95% CI**[1]	**p-value**
age	0.40	0.29, 0.51	<0.001
mins_exercise	-0.10	-0.12, -0.08	<0.001

[1]CI = Confidence Interval

term	estimate	std.error	statistic	p.value	conf.low	conf.high
age	0.400	0.054	7.345	0	0.292	0.508
mins_exercise	-0.104	0.010	-10.125	0	-0.124	-0.083

Another approach is to use `tbl_regression` function within the `gtsummary` package to get a publication ready table of the coefficients and confidence intervals:

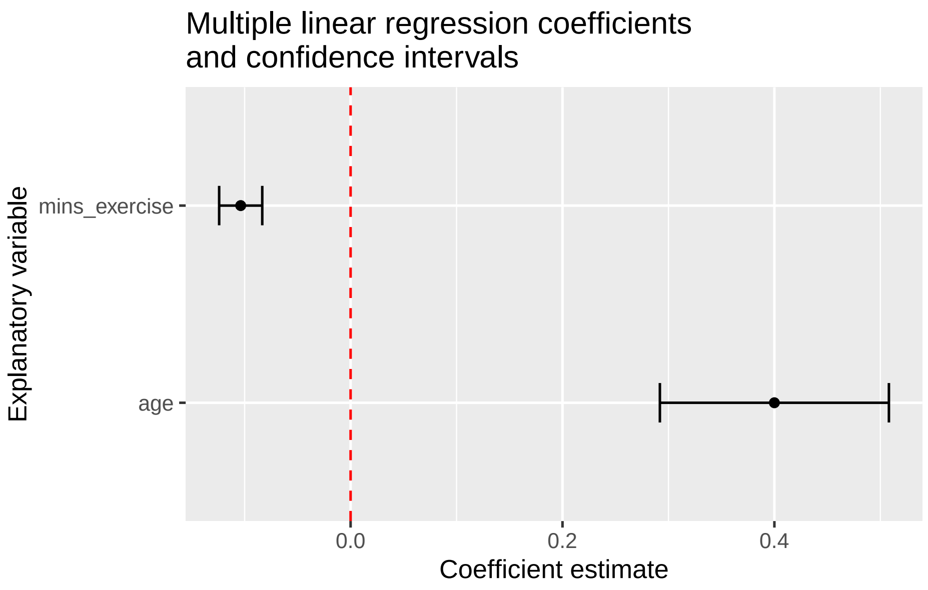
```
tbl_regression(m1)
```

To use either approach most efficiently you will, however, need to write your report using R Markdown or Quarto so that the table is automatically created in your report document.

A figure showing the coefficients and confidence intervals

As well as or rather than a table of coefficients, we could make a figure showing the coefficients and confidence intervals for each explanatory variable. This can be a nice way to visualise the results of the multiple linear regression model.

```
tidy_m1 <- tidy(m1, conf.int = TRUE)
tidy_m1 |>
  filter(term != "(Intercept)") |>
ggplot(aes(x = term, y = estimate)) +
  geom_point() +
  geom_errorbar(aes(ymin = conf.low, ymax = conf.high), width =
                0.2) +
  geom_hline(yintercept = 0, linetype = "dashed", color = "red") +
  labs(x = "Explanatory variable", y = "Coefficient estimate") +
  ggtitle("Multiple linear regression coefficients\nand confidence intervals") +
  coord_flip()
```


Multiple linear regression coefficients
and confidence intervals
Explanatory variable
mins_exercise
age
0.0
0.2
0.4
Coefficient estimate

Interactions (L7)

Thought for the week:

“*You should arrive at answers on your own, and not rely upon what you get from someone else. Answers from others are nothing more than stopgap measures, they’re of no value.*”

- Ichiro Kishimi & Fumitake Koga, The Courage to Be Disliked

They instead encourage answers gained from curiosity, observation, and dialogue. Answers you develop for yourself are more likely to stick with you, and to be meaningful.

When we mentor, ask questions that encourage others to think for themselves, rather than simply giving answers. This is hard, but worthwhile.

Introduction

In the previous chapters we have looked at models with a special type of simplicity: the effect of each explanatory variable on the response variable is independent of the other explanatory variables. This is special because it means that we can understand the effect of each explanatory variable on the response variable separately, without considering the other explanatory variables. Nevertheless, this is often not the case in real life. The effect of one explanatory variable on the response variable may depend on the value of another explanatory variable. This is called an interaction effect, or simply an interaction.

Interactions are some of the most interesting phenomena in science, including biology. We are not talking about interactions between species, like predation, though these are also very interesting. We are talking about effects of one thing, like diet, depending on another thing, like exercise. Let’s break that down a bit…

Imagine we make a study of the effect of exercise (minutes per week) on blood pressure for people with a meat heavy diet.

Read in the dataset:

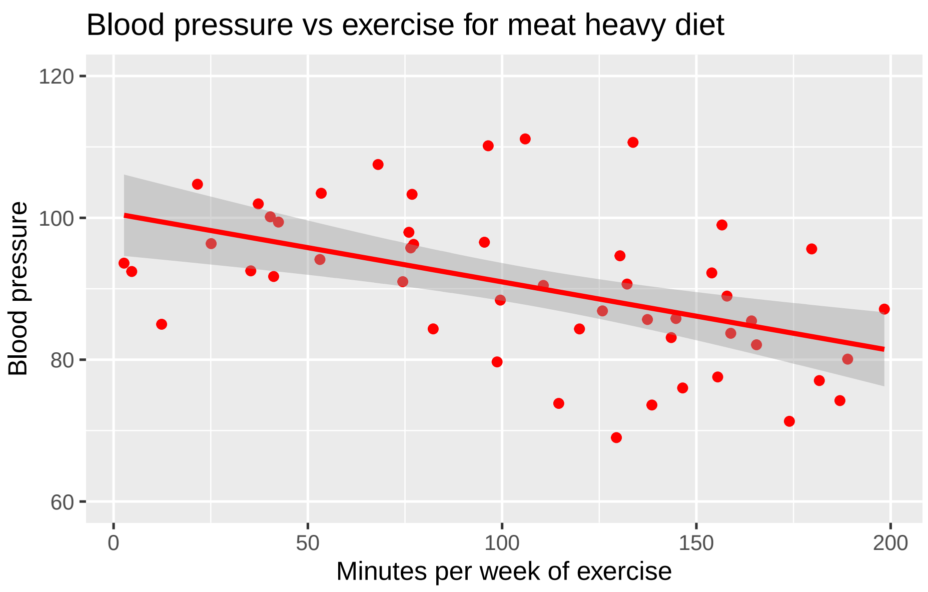
```
bp_meatheavy <- read.csv("datasets/bp_meatheavy.csv")
```

Here are the first few rows of the data:

```
head(bp_meatheavy)
```

```
         bp mins_per_week       diet
1  94.12854      53.10173 meat heavy
2  90.99957      74.42478 meat heavy
3  73.83541     114.57067 meat heavy
4  77.05434     181.64156 meat heavy
5 100.14578      40.33639 meat heavy
6  95.61900     179.67794 meat heavy
```

And here is a graph of the relationship between blood pressure and minutes of exercise for people with a meat heavy diet:



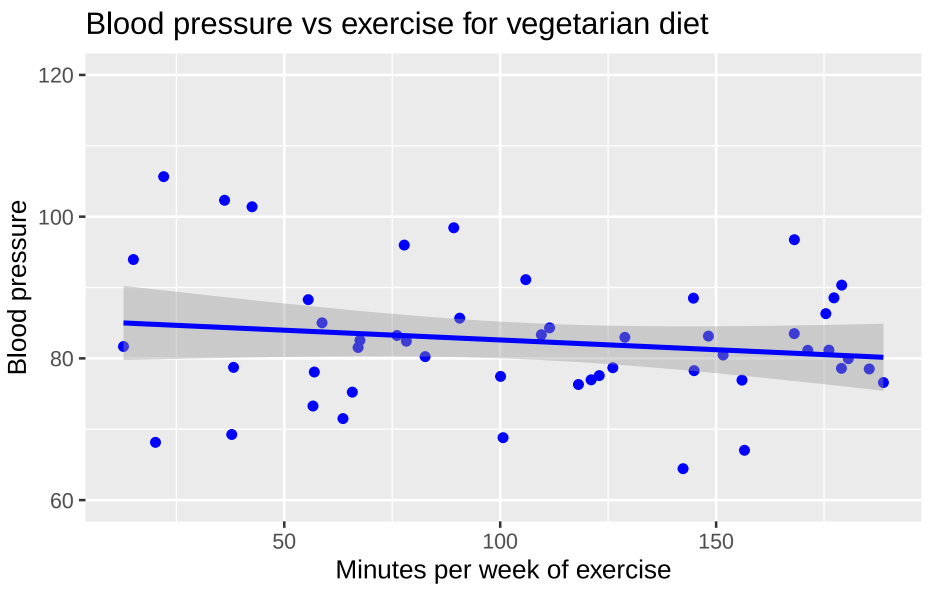
We see that exercise seems to lower blood pressure. But what if we look at the effect of exercise on blood pressure for people with a vegetarian diet? The relationship might look something like this:

```
         bp mins_per_week       diet
1  77.56704     122.92899 vegetarian
2  84.30684     111.43191 vegetarian
3  75.23323      65.75546 vegetarian
4  85.67402      90.62629 vegetarian
5  77.45327     100.08819 vegetarian
6 102.31114      36.17327 vegetarian
```

Read in the dataset:

```
bp_vegetarian <- read.csv("datasets/bp_vegetarian.csv")
```

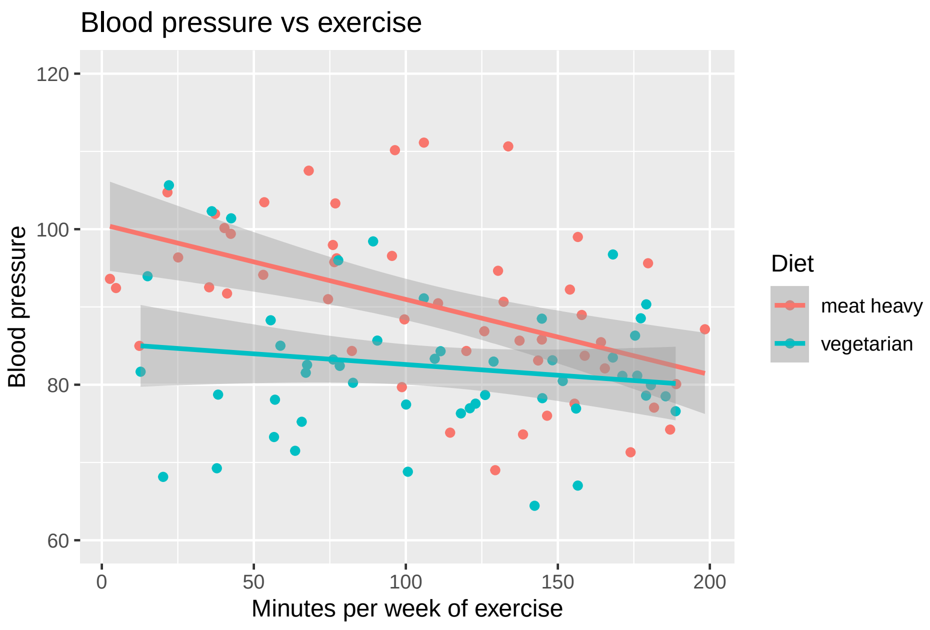
Here is a graph of the relationship between blood pressure and minutes of exercise for people with a vegetarian diet:



We see that exercise seems to lower blood pressure for vegetarians too, but the effect seems to be weaker.

To summarise this finding, we can say that the effect of exercise on blood pressure is stronger for people with a meat heavy diet than for people with a vegetarian diet. This means that the effect of exercise on blood pressure depends on diet.

This is very clear when we look at the both diets in the same graph:



(The dataset with both combined is in the file `bp_1cont1cat.csv`.)

Think-Pair-Share (#tps-general-diet) Can we say anything general about the effect of diet on blood pressure?

No, we can't. We cannot, for example, state that a vegetarian diet lowers blood pressure. We can say, however, that a vegetarian diet lowers blood pressure of people that do little exercise.

Think-Pair-Share (#tps-general-exercise) Can we say anything general about the effect of exercise on blood pressure?

Well, we can say that exercise lowers blood pressure, but we have to be careful. We have to say that exercise lowers blood pressure of people with a meat heavy diet more than it lowers blood pressure of people with a vegetarian diet.

I think it is clear that the interaction was easier to see when we plotted all the data in one graph… it is much easier to visually compare the slopes of the two regression lines when they are on the same graph.

As we will see later in this chapter, the same holds true for statistical tests of interactions: it is much easier to make a statistical test of the interaction when we make a single model with *an interaction term*.

It is harder and is not recommended to make a separate regression for each level of the second variable (diet) and then compare the slopes of the regression lines (it is possible, just not at all efficient).

Parallel and non-parallel effects

In the example above, the effect of exercise on blood pressure was stronger for people with a meat heavy diet than for people with a vegetarian diet. That is, the slope of the regression line was steeper for the meat heavy diet than for the vegetarian diet. Put another way, the regression lines are not parallel.

Parallel = no interaction: Parallel regression lines are evidence of no interaction. This means that the effect of one variable (exercise) is the same for all levels of another variable (diet).

Non-parallel = interaction: When the regression lines are not parallel, there is evidence of an interaction. This means that the effect of one variable (exercise) depends on the level of another variable (diet).

That was an example of an interaction between a continuous explanatory variable (exercise) and a categorical explanatory variable (diet). Interactions can also occur between two categorical explanatory variables, or between two continuous explanatory variables. Let us look at some more examples.

Think–Pair–Share (#a_when_lines_cross) What does it mean biologically when lines cross in an interaction plot? What does it mean when they are parallel?

Two cats

Two categorical explanatory variables: diet (meat heavy or vegetarian) and exercise (low or high), and one continuous response variable (blood pressure).

Read in the dataset:

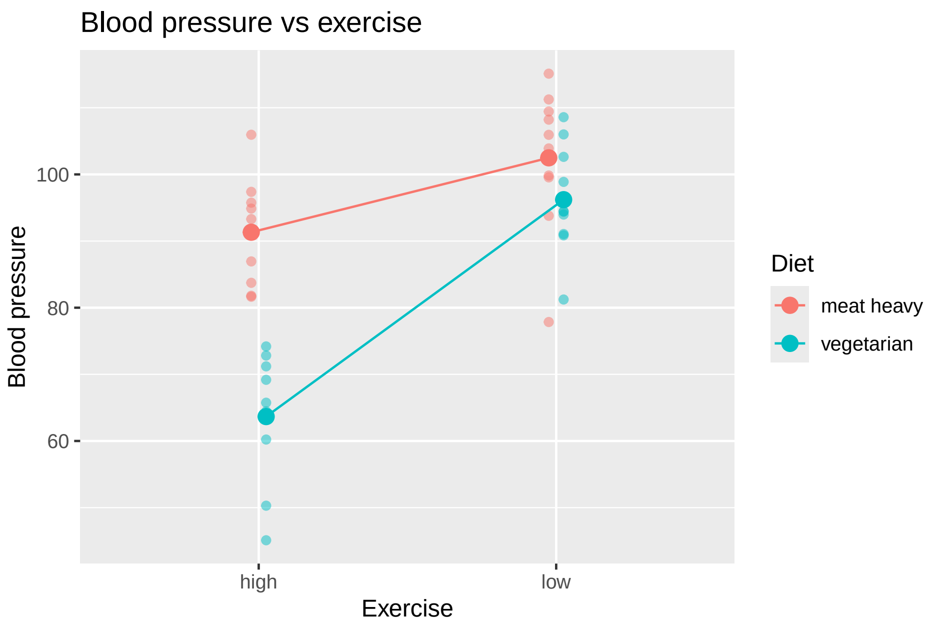
```r
bp_2cat <- read.csv("datasets/bp_2cat.csv")
```

Here are the first few rows of the data:

```r
head(bp_2cat)
```

```
         diet exercise reps        bp
1 meat heavy     high    1  83.73546
2 meat heavy      low    1 115.11781
3 vegetarian     high    1  74.18977
4 vegetarian      low    1 108.58680
5 meat heavy     high    2  91.83643
6 meat heavy      low    2 103.89843
```

And here is the data in a graph, where we show the individual data points as well as the group means connected by lines. The lines connecting the means are only to help visualize whether there is an interaction or not. If we see non-parallel lines connecting the means, we have evidence of an interaction.



Indeed, the lines connecting the means are not parallel, which is evidence of an interaction between diet and exercise on blood pressure.

Think-Pair-Share (#tps-express-int) Express the nature of the interaction in words.

Two continuous

Two continuous explanatory variables (age and exercise minutes) and one continuous response variable (blood pressure).

Read in the dataset:

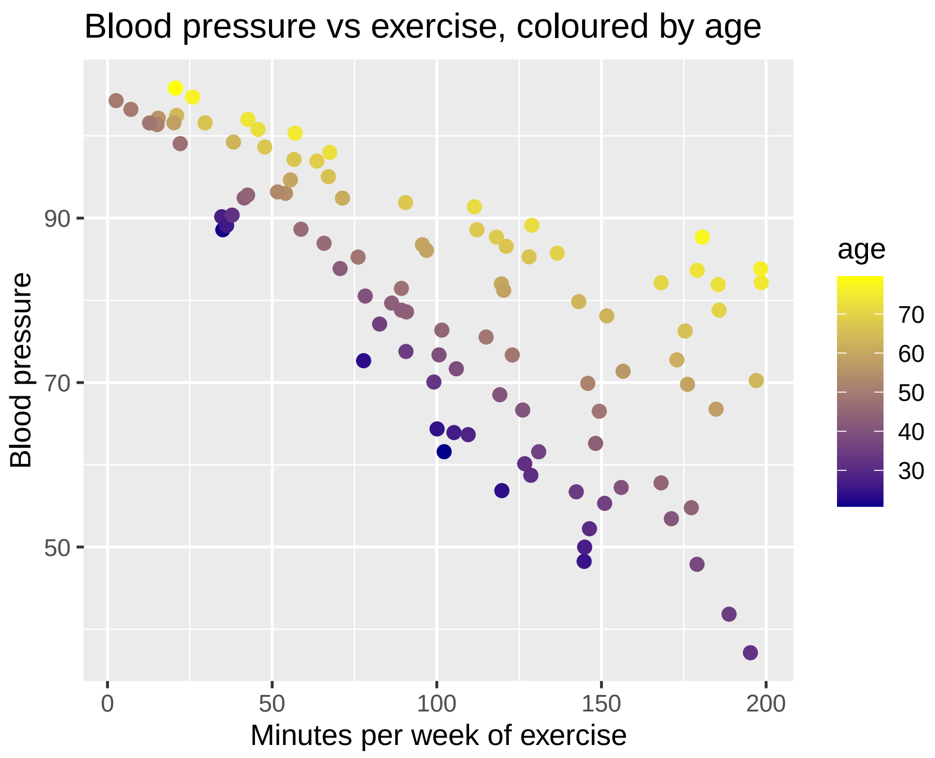
```r
bp_2cont <- read.csv("datasets/bp_2cont.csv")
```

Here is the first few rows of the data:

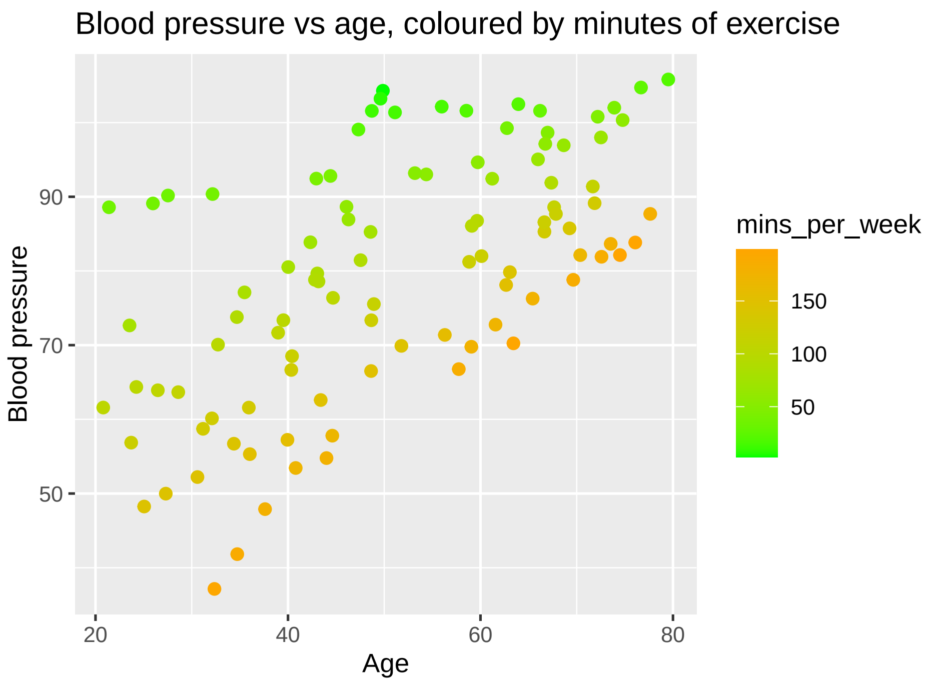
```r
head(bp_2cont)
```

```
         bp      age mins_per_week
1  61.58319 35.93052     130.94479
2  83.86716 42.32743      70.63945
3  93.01438 54.37120      54.05203
4  82.14413 74.49247     198.53681
5  60.13102 32.10092     126.69865
6 102.00306 73.90338      42.64163
```

Now we have a little challenge, namely that we have two continuous explanatory variables. This means that we need to use three dimensions to visualize the data. Here is a standard 2D scatter plot of blood pressure against minutes of exercise, though with age represented by color grading from young (dark blue) to old (yellow):



And we can make the complementary plot of blood pressure against age, with minutes of exercise represented by color grading from low (green) to high (orange):



What do we see? If we look at a set of points of the similar colour (i.e., similar number of minutes of exercised), we can see that the slope of the relationship between blood pressure and age depends on exercise. The slope is steeper for people that exercise more.

Think-Pair-Share (#tps-small-older) Another thing we could say is that the effect of exercise is smaller for older people. How is this shown in the graph?

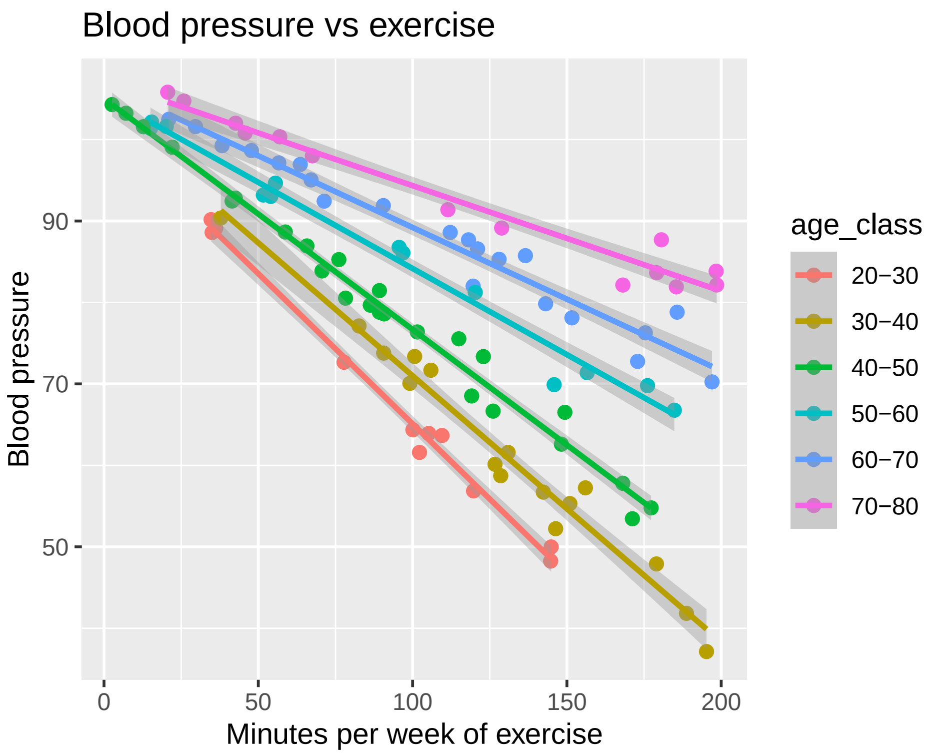
Yet another way to visualise this interaction is to create categorical versions of age and minutes of exercise, and then plot the data with these categorical variables:

Here is the first few rows of the modified data:

```
head(bp_2cont)
```

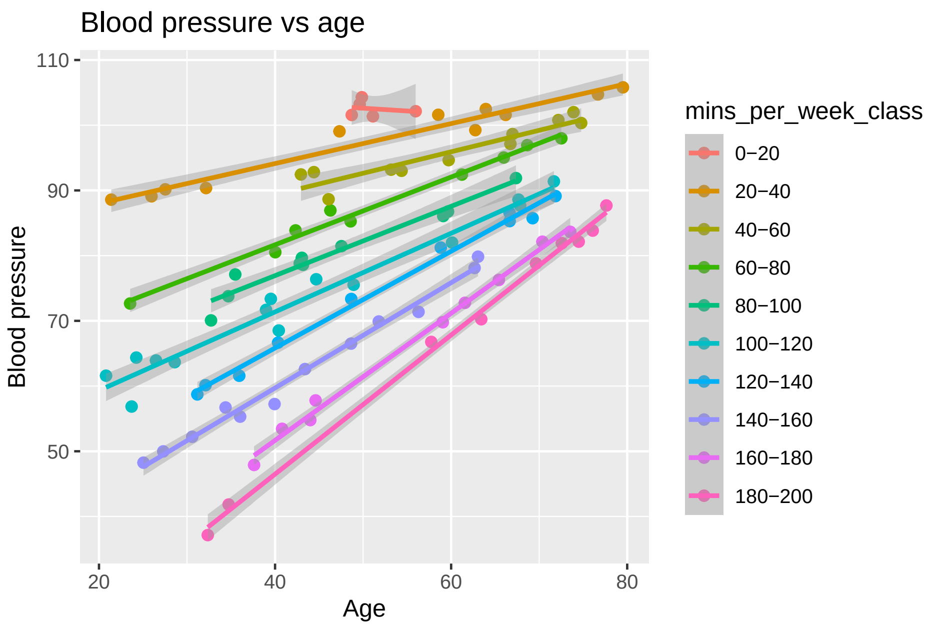
```
         bp      age mins_per_week age_class mins_per_week_class
1  61.58319 35.93052     130.94479     30-40             120-140
2  83.86716 42.32743      70.63945     40-50               60-80
3  93.01438 54.37120      54.05203     50-60               40-60
4  82.14413 74.49247     198.53681     70-80             180-200
5  60.13102 32.10092     126.69865     30-40             120-140
6 102.00306 73.90338      42.64163     70-80               40-60
```

And here is the plot of blood pressure against minutes of exercise, with age class represented by colour. Regression lines have been added, to help visualize any interaction:



It is as we saw in the previous version of the graph. The slope of the relationship between blood pressure and minutes of exercise depends on age. The slope is shallower for older people.

In the complementary graph of blood pressure against age, with minutes of exercise represented by colour, we see the same interaction:



! Important

There is only one interaction We made two graphs of the same data, one with age as a categorical variable and one with minutes of exercise as a categorical variable. In both graphs we see an interaction. This is not us seeing two separate interactions, however. There is only one interaction here, namely that the effect of age on blood pressure depends on minutes of exercise, and equivalently, that the effect of minutes of exercise on blood pressure depends on age. The two graphs just look at the same interaction from different perspectives.

Other perspectives

Interactions and additivity of effects

Another way of thinking about interactions is from the perspective of additivity or non-additivity of effects. Imagine we made two separate studies, one of the effect of diet on blood pressure, and one of the effect of exercise on blood pressure. In the first study we only varied diet, and in the second study we only varied exercise. In the first study we found an effect size of diet on blood pressure of say 10 mmHg (e.g., difference between meat heavy and vegetarian). And in the second study we found an effect size of exercise on blood pressure of 15 mmHg (e.g., difference between low and high exercise).

Think-Pair-Share (#tps-adding-effects) If the effects of diet and exercise were additive, what would we expect the effect size to be if we estimated the effect of diet and exercise on blood pressure in the same study?

Think-Pair-Share (#tps-comb-non-add) What would we expect the effect size of combined diet and exercise to be if the effects were non-additive? (Hint: this is a bit of a trick question.)

If the effects are non-additive, we would expect the effect size to be different from additive. For example, if we found the combined effect of diet and exercise on blood pressure to be 40, we would say that the effects are non-additive. Their combined effect is more than the sum of their individual effects. This example is of a synergistic interaction because the combined effect (40) is greater than the sum of the individual effects (25).

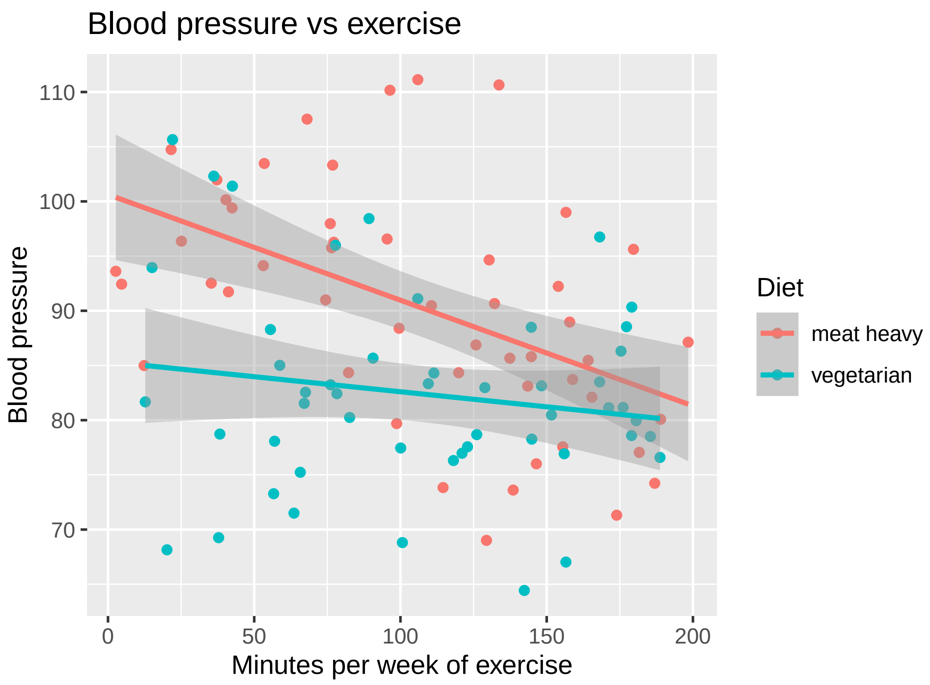
Drug interactions

When a doctor is considering giving us a particular medication, we are asked if we are taking any other medications. This is because the effects of drugs can interact. For example, if we take two drugs that both lower blood pressure and they interfere with each other, the combined effect might be less than the sum of their individual effects. This is an antagonistic interaction. It could be worse than that though, the interaction might actually be harmful, which is why doctors are so careful about drug interactions.

Think-Pair-Share (#tps-other-int-examples) Can you think of any other examples of interactions in biology or medicine?

The maths bit

Let us return to the example of the effects of number of minutes of exercise and diet on blood pressure:



We have one continuous explanatory variable (minutes of exercise) and one binary explanatory variable (diet) and one continuous response variable (blood pressure).

Think-Pair-Share (#tps-without-interaction) What would a linear model without an interaction term be?

$$y_i = \beta_0 + \beta_1 x_i^{(1)} + \beta_2 x_i^{(2)} + \epsilon_i$$

where:

- y_i is the blood pressure of the ith participant
- $x_i^{(1)} is the number of minutes of exercise of the ith participant
- $x_i^{(2)}$ is the diet of the ith participant
- β_0 is the intercept
- β_1 is the effect of exercise on blood pressure
- β_2 is the effect of diet on blood pressure
- ϵ_i is the error term for the ith participant.

This model is a multiple regression model in which one of the explanatory variables is binary.

Think-Pair-Share (#tps-with-interaction) What might the model look like if we wanted to include an interaction between diet and exercise?

$$y_i = \beta_0 + \beta_1 x_i^{(1)} + \beta_2 x_i^{(2)} + \beta_3 (x_i^{(1)} x_i^{(2)}) + \epsilon_i$$

where:

- $x_i^{(1)} x_i^{(2)}$ is the product of the number of minutes of exercise and the diet of the ith participant.
- β_3 is the coefficient of the interaction term between diet and exercise.

We could also write this model as:

$$y_i = \beta_0 + \beta_1 x_i^{(1)} + \beta_2 x_i^{(2)} + \beta_3 x_i^{(3)} + \epsilon_i$$

where:

- $x_i^{(3)} = x_i^{(1)} x_i^{(2)}$

This is again a multiple regression model, but now with three explanatory variables.

Think-Pair-Share (#tps-sketch-interaction) Make sketches of the possible relationships between diet, exercise and blood pressure. Make a sketch compatible with $\beta_3 = 0$. Make a sketch compatible with $\beta_3 \neq 0$.

Hypothesis testing

If we want to test whether the effect of minutes of exercise on blood pressure is different for people with different diets, we need a null hypothesis to test.

Think-Pair-Share (#tps-interaction-null) What is the null hypothesis in this case, verbally, and in terms of the coefficients of the model?

The null hypothesis is that the effect of minutes of exercise on blood pressure is the same for people with different diets. This is a null hypothesis of no interaction between diet and exercise. In terms of the coefficients of the model, the null hypothesis is that $\beta_3 = 0$.

If we reject the null hypothesis, we conclude that the effect of minutes of exercise on blood pressure is different for people with different diets. This is a non-additive effect.

Doing it in R

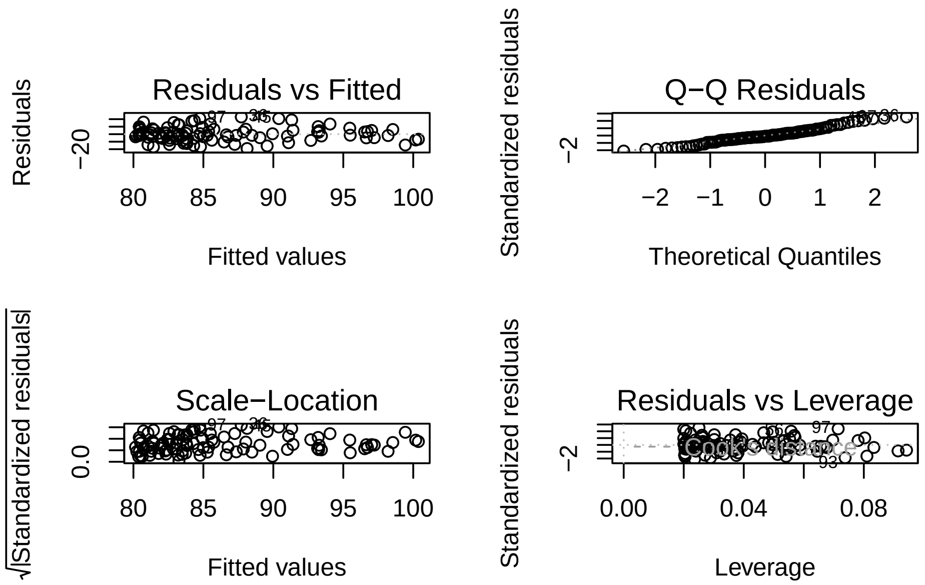
Let us fit the model with the interaction term in R. There are two methods to do this and they are equivalent:

```r
mod1 <- lm(bp ~ mins_per_week + diet + mins_per_week:diet, data=bp_1cont1cat)
mod2 <- lm(bp ~ mins_per_week * diet, data=bp_1cont1cat)
```

The second is a shorthand for the first. The `*` operator includes the main effects (main effects are terms in the model that don't include interactions) and the interaction term. The `:` operator includes only the interaction term.

Of course, we check the model assumptions before we interpret the results:

```
par(mfrow=c(2,2))
plot(mod2, add.smooth=FALSE)
```



All of the plots look good. Now we can do hypothesis testing of the interaction term using an F-test:

```
anova(mod2)
```

```
Analysis of Variance Table

Response: bp
                  Df Sum Sq Mean Sq F value    Pr(>F)
mins_per_week      1 1133.6 1133.55 13.4800 0.0003964 ***
diet               1 1560.8 1560.75 18.5601 3.979e-05 ***
mins_per_week:diet  1  340.4  340.36  4.0475 0.0470408 *
Residuals         96 8072.8   84.09
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

In the ANOVA table we see four rows. The first row is for the main effect of minutes of exercise, the second row is for the main effect of diet, the third row is for the interaction effect between diet and minutes of exercise, and the fourth row is for the residuals. As always, an interaction term in the R output is shown with a colon `:` between the two variables (here it looks like `mins_per_week:diet`).

In this example, the F-statistic for the interaction term is quite large (4.05), and the p-value is very small (0.047). This means that we reject the null hypothesis that there is no interaction between diet and minutes of exercise on blood pressure. We conclude that the effect of minutes of exercise on blood pressure is different for people with different diets.

If we like (and we don't have to), we can look at the coefficients of the model:

```r
summary(mod2)$coefficients
```

```
                                Estimate Std. Error   t value     Pr(>|t|)
(Intercept)                  100.62716134 2.87228930 35.033783 1.714199e-56
mins_per_week                 -0.09661054 0.02406014 -4.015376 1.177969e-04
dietvegetarian               -15.27591304 4.09882726 -3.726898 3.275630e-04
mins_per_week:dietvegetarian   0.06907583 0.03433487  2.011828 4.704076e-02
```

As expected, there are four coefficients.

The first is `(Intercept)`, which is the expected blood pressure for a person who does 0 minutes of exercise and is on diet "meat heavy".

The second is `mins_per_week`, which is the effect (slope) of minutes of exercise on blood pressure for a person on diet "meat heavy".

The third is `dietvegetarian`, which is the effect of being on a vegetarian diet on blood pressure for a person who does 0 minutes of exercise. This can be thought of as the change in the intercept for a person on a vegetarian diet compared to a person on a "meat heavy" diet.

The fourth is the interaction term `mins_per_week:dietvegetarian`, which is the difference in the effect (slope) of minutes of exercise on blood pressure for a person on a vegetarian diet compared to a person on a "meat heavy" diet.

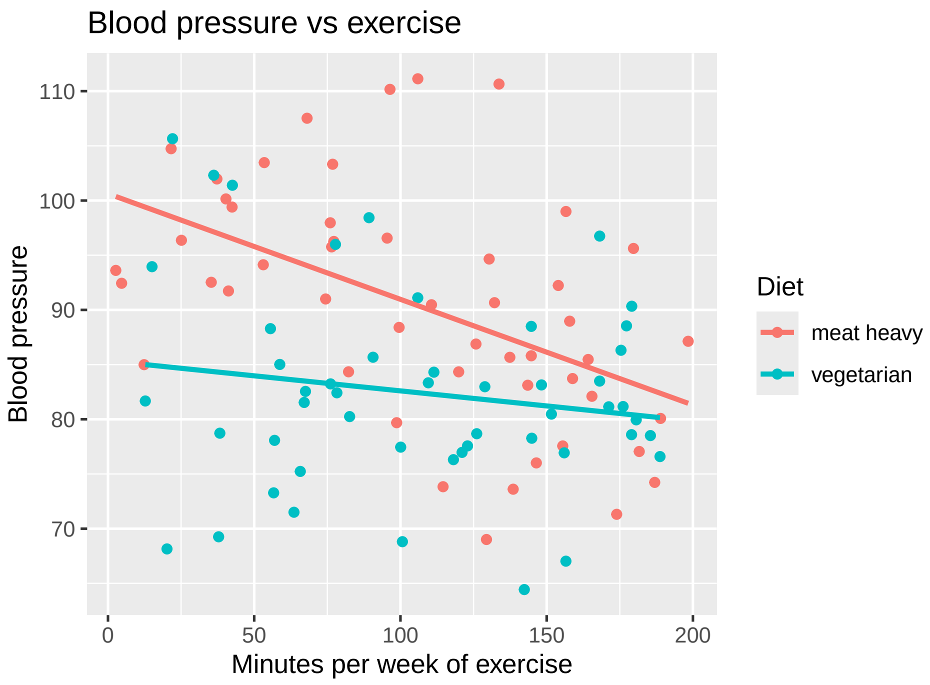
Think-Pair-Share (#tps-two-equations) Write two equations, one for each of the two diets. They would look something like this: $y_i = 0.1 - 0.1x_i^{(1)}$, but will have other numbers.

Reporting our findings

Of course a nice graph is always helpful. We already have quite a nice one:

	Estimate	Std. Error	t value	Pr(>|t|)
(Intercept)	100.6271613	2.8722893	35.033783	0.0000000
mins_per_week	-0.0966105	0.0240601	-4.015377	0.0001178
dietvegetarian	-15.2759130	4.0988273	-3.726898	0.0003276
mins_per_week:dietvegetarian	0.0690758	0.0343349	2.011828	0.0470408

	Df	Sum Sq	Mean Sq	F value	Pr(>F)
mins_per_week	1	1133.554	1133.55446	13.47998	0.0003964
diet	1	1560.753	1560.75255	18.56012	0.0000398
mins_per_week:diet	1	340.357	340.35702	4.04745	0.0470408
Residuals	96	8072.804	84.09171	NA	NA



We also might want some tables summarizing the model results. Here is a table of the coefficients:

We could also report the R^2 of the model:

```
summary(mod2)$r.squared
```

```
[1] 0.2732093
```

And also a table of the variances of the terms in the model:

We also might use a sentence like this to report the results: "The effect of minutes of exercise is generally negative, but the effect is stronger for people on a meat heavy diet than for people on a vegetarian diet (F-statistics of interaction term = 4.05, degrees of freedom = 1, degrees of freedom residuals = 96, p-value = 0.047).”

Think–Pair–Share (#a__interaction__vs__main__effect) How does an interaction change the meaning of a “main effect”? Can a main effect be misleading when an interaction is present?

Multiple regression vs. many single regressions

Question: Why not just fit a separate simple regression model and then test whether the slopes are the same (i.e., if they are parallel)? That is, why not fit the two models:

$$y_i = \beta_{0,veg} + \beta_{1,veg} x_i^{(1)} + \epsilon_i$$

$$y_i = \beta_{0,meat} + \beta_{1,meat} x_i^{(2)} + \epsilon_i$$

and compare the estimate of $\beta_{1,veg}$ to the estimate of $\beta_{1,meat}$?

Well, you could do that, and could probably find a way to test for whether the difference in the slopes is different from 0. This would be a test of the null hypothesis that the effect of minutes of exercise on blood pressure is the same for people with different diets. But, this would be a more complicated way to do it, and would not be as general as the model with the interaction term. The model with the interaction term is more general, more flexible, and more elegant.

Note

It is usually better model the whole dataset to test a single hypothesis, rather dividing up the dataset into smaller parts, fitting a model to each part, and then comparing the results of the models. The latter approach is less efficient and less elegant. **One hypothesis = one model.**

ANCOVA

The example we just worked through was with one continuous explanatory variable (minutes of exercise) and one categorical explanatory variable (diet). This is an example of an analysis of covariance (ANCOVA). ANCOVA is a type of linear model in which there are both continuous and categorical explanatory variables.

ANCOVA is often used in two main ways.

First, it can be used to account for covariates (continuous variables). In this case, the main interest is in comparing groups (as in ANOVA), while one or more continuous variables are included to explain additional variation in the response. These covariates are not the main focus of interpretation; instead, they help adjust group means and improve the precision of group comparisons.

Second, ANCOVA can be used to test whether covariate effects differ between groups. Here, the continuous variable is of real interest, and the question is whether the relationship between the covariate and the response is the same across groups. This is done by including a group $\times$ covariate interaction, which allows the slope of the relationship to differ between groups.

An important distinction is that the first use assumes the covariate has the same effect in all groups (parallel slopes), while the second explicitly tests whether this assumption is valid.

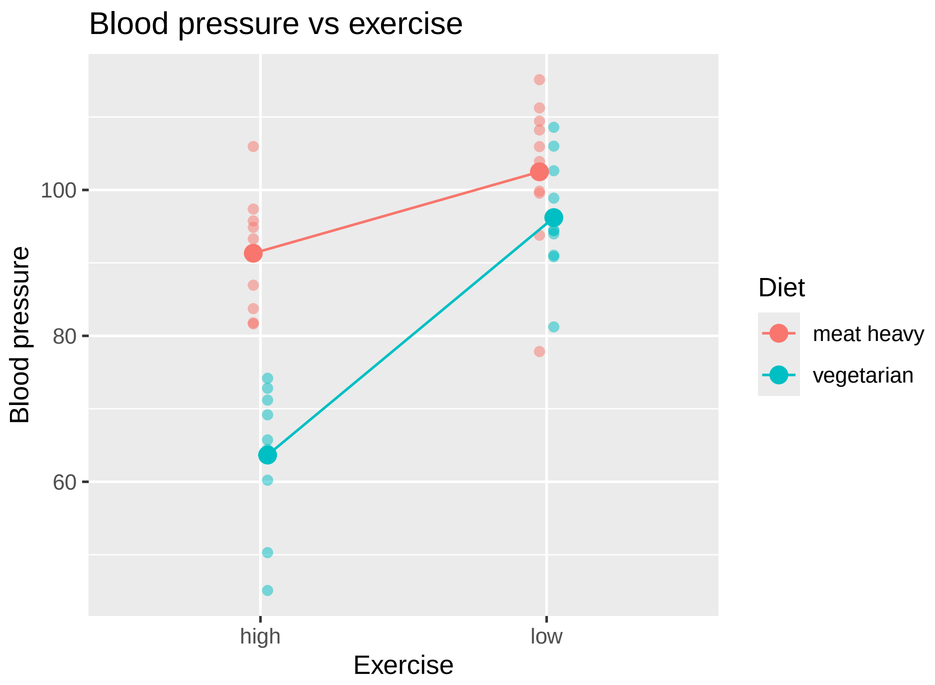
Two-way ANOVA

Above we had an example with two categorical explanatory variables (diet [levels: meat heavy or vegetarian] and exercise [levels: low or high]) and one continuous response variable (blood pressure). This is an example of a two-way ANOVA. Two-way ANOVA is used to test for effects of two categorical explanatory variables, as well as their interaction effect on a continuous response variable.

Here are the first few rows of the data again:

```
        diet exercise reps        bp
1 meat heavy     high    1  83.73546
2 meat heavy      low    1 115.11781
3 vegetarian     high    1  74.18977
4 vegetarian      low    1 108.58680
5 meat heavy     high    2  91.83643
6 meat heavy      low    2 103.89843
```

Here is the graph of the data again:



We can fit a two-way ANOVA model in R as follows:

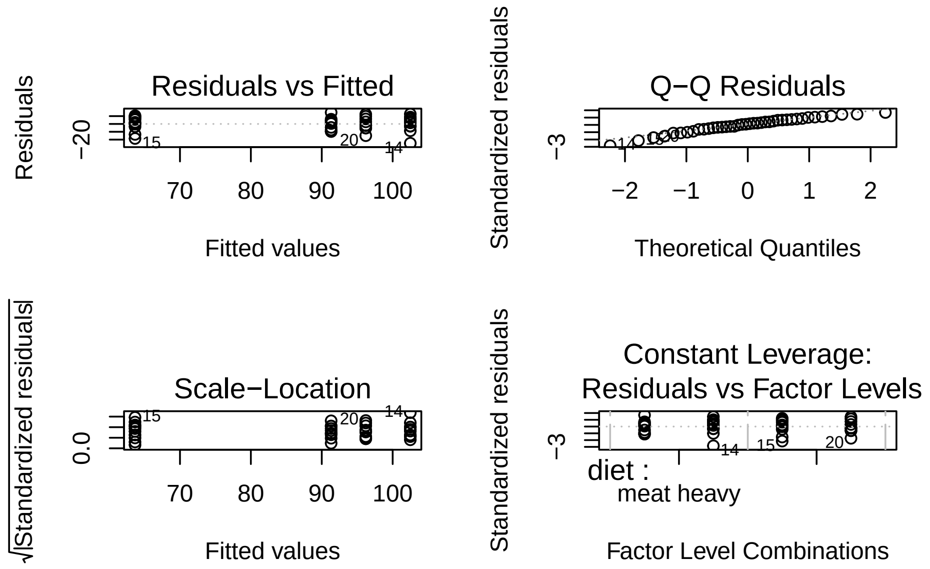
```
mod_2cat <- lm(bp ~ diet * exercise, data=bp_2cat)
```

Recall that this fits a model with main effects of diet and exercise, as well as their interaction effect. Recall that we could specify the model equivalently as:

```
mod_2cat <- lm(bp ~ diet + exercise + diet:exercise, data=bp_2cat)
```

We can check the model assumptions:

```
par(mfrow=c(2,2))
plot(mod_2cat, add.smooth=FALSE)
```



These look ok.

Hypothesis testing the interaction is just as before. We use an F-test on the interaction term to test the null hypothesis that there is no interaction between diet and exercise on blood pressure.

```
anova(mod_2cat)
```

```
Analysis of Variance Table

Response: bp
              Df Sum Sq Mean Sq F value    Pr(>F)    
diet           1 2879.8  2879.8  34.695 9.741e-07 ***
exercise       1 4776.5  4776.5  57.546 5.680e-09 ***
diet:exercise  1 1142.5  1142.5  13.765  0.000696 ***
Residuals     36 2988.1    83.0                      
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

In the ANOVA table we see four rows. The first row is for the main effect of diet, the second row is for the main effect of exercise, the third row is for the interaction effect between diet and exercise, and the fourth row is for the residuals. As always, an interaction term in the R output is shown with a colon `:` between the two variables (here it looks like `diet:exercise`).

In this example, the F-statistic for the interaction term is quite large (13.76 and the corresponding p-value is quite small (7×10^{-4}. Hence, we conclude that there is a strong evidence of an interaction between diet and exercise on blood pressure. The effect of exercise on blood pressure depends on diet.

Think-Pair-Share (#tps-two-way-report) Write a sentence reporting the results of the two-way ANOVA. Focus on the biological interpretation of the results, rather than the statistical details. Include statistics in parentheses, in support of your statements.

! Important

Reporting two-way ANOVA results When reporting the results of a two-way ANOVA, it is important to focus on the biological interpretation of the results, rather than the statistical details. Include statistics in parentheses, in support of your statements. For example, you might say something like: "The effect of exercise was different for people on different diets, with a stronger effect for those on a vegetarian diet compared to those on a meat heavy diet (F(1, 36) = 13.77, $p < 0.001$)."

More than two levels

What if we had a categorical explanatory variable with more than two levels? For example, what if diet had three levels: meat heavy, vegetarian, and vegan?

Here is an example dataset.

Read in the dataset:

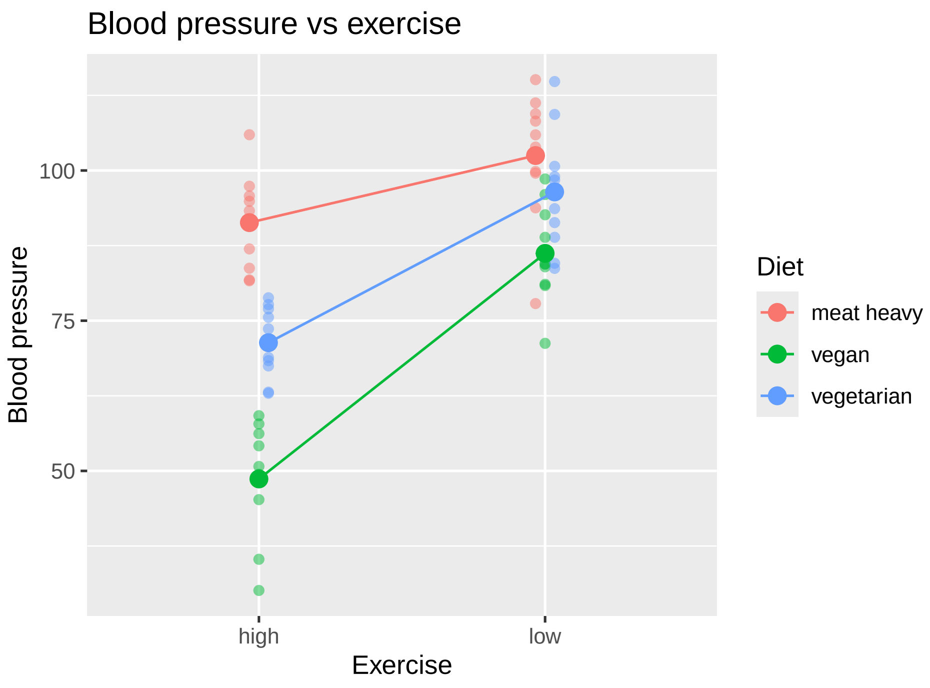
```r
bp_2cat_3levels <- read.csv("datasets/bp_3cat.csv")
```

Here are the first few rows of the data:

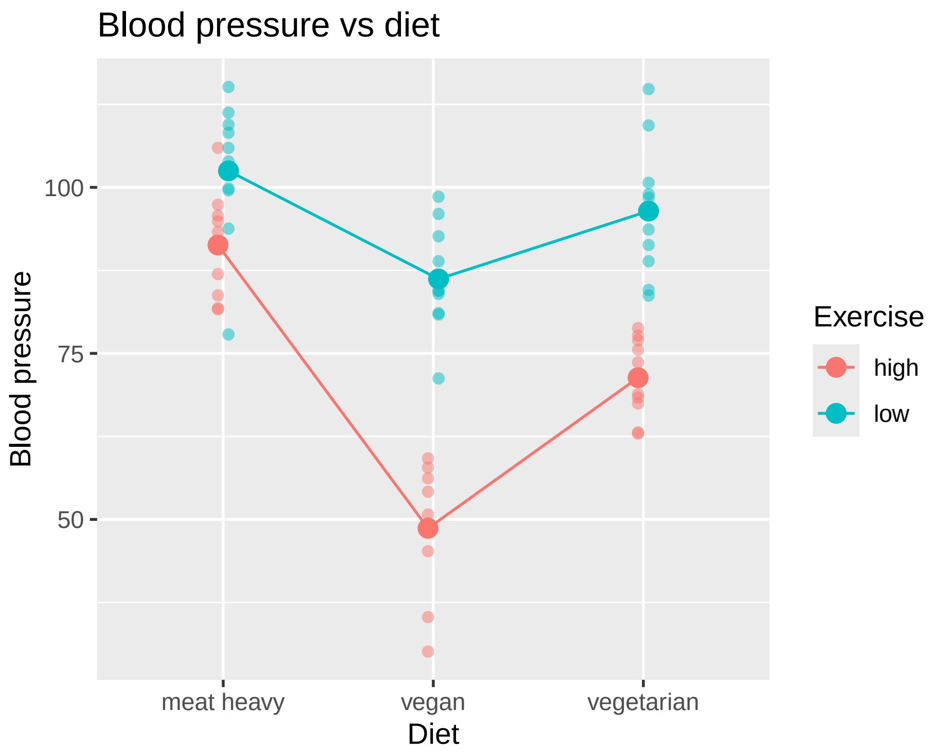
```r
head(bp_2cat_3levels)
```

```
        diet exercise reps        bp
1 meat heavy     high    1  83.73546
2 meat heavy      low    1 115.11781
3      vegan     high    1  59.18977
4      vegan      low    1  98.58680
5 vegetarian     high    1  68.35476
6 vegetarian      low    1  98.98106
```

Here is the graph of the data:



Or plotted differently:

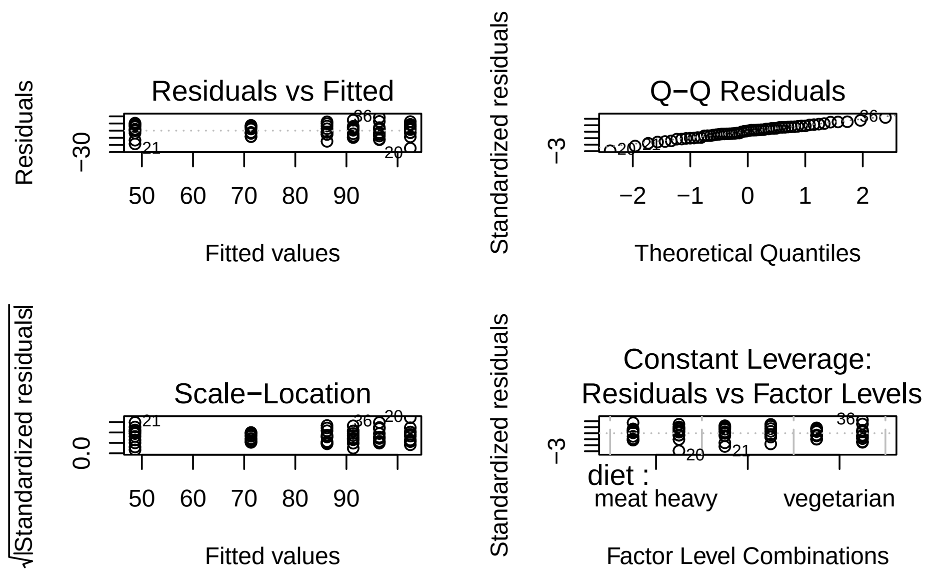


The hypothesis testing is the same as before. We fit the model with interaction term:

```r
mod_2cat_3levels <- lm(bp ~ diet * exercise, data=bp_2cat_3levels)
```

We check the model assumptions:

```r
par(mfrow=c(2,2))
plot(mod_2cat_3levels, add.smooth=FALSE)
```



These look ok.

Now we do the ANOVA:

```
anova(mod_2cat_3levels)
```

```
Analysis of Variance Table

Response: bp
              Df Sum Sq Mean Sq F value    Pr(>F)    
diet           2 8724.1  4362.1  55.636 7.644e-14 ***
exercise       1 9078.3  9078.3 115.789 4.791e-15 ***
diet:exercise  2 1741.3   870.6  11.104 9.130e-05 ***
Residuals     54 4233.8    78.4                      
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Important: Although we have a categorical variable with three rather than two levels, we still have only four rows in the ANOVA table. One for each main effect (diet and exercise), one for the interaction effect (diet:exercise), and one for the residuals. This is because the ANOVA table tests each effect as a whole, rather than testing each level of the categorical variable separately.

In this case, we see that there is strong evidence of main effects of diet and exercise on blood pressure, as well as strong evidence of an interaction effect between diet and exercise on blood pressure.

Reporting the patterns and statistics is similar to before, but now we have more levels to consider so the reporting is a bit more complex, and we have to be careful to not over-interpret the results. For example, when the hypothesis test is on the interaction term via an F-test, we can only say that there is evidence of an interaction between diet and exercise on blood pressure. We cannot say which diets have different effects of exercise on blood pressure. To do that, we would need to do post-hoc tests, such as pairwise comparisons between the levels of diet within each level of exercise.

Caution

Degrees of freedom Look at the ANOVA table and the degrees of freedom column. For diet, the degrees of freedom is 2, because there are three levels of diet (meat heavy, vegetarian, vegan), and the degrees of freedom is number of levels minus 1. For exercise, the degrees of freedom is 1, because there are two levels of exercise (low, high). For the interaction term diet:exercise, the degrees of freedom is 2, which is the product of the degrees of freedom for diet (2) and exercise (1).
Another way to think about this is how many parameters are estimated? The answer is as follows: six parameters are estimated in total, one for each of the combinations of diet and exercise (3 diets x 2 exercises = 6 combinations). Hence the residual degrees of freedom is total number of observations (60) minus 6.
Don't worry if this is a bit confusing at first. It will become clearer with practice, and you can ask for it to be explained again and in different ways.

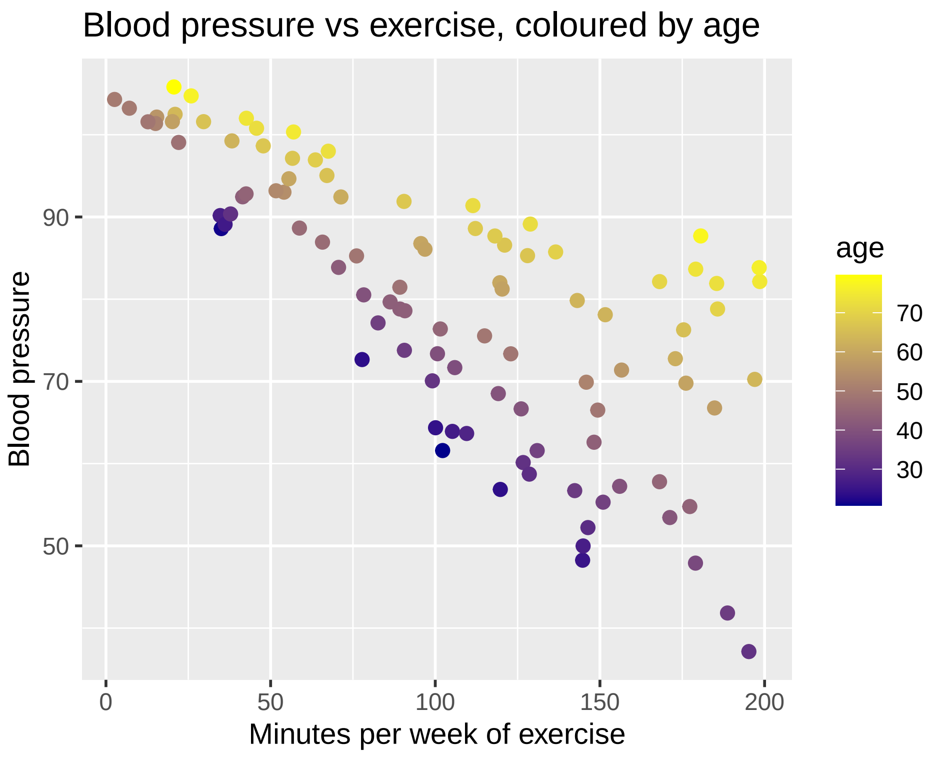
Multiple regression with interaction term

Above we had the example of two continuous explanatory variables (age and minutes of exercise) and one continuous response variable (blood pressure). We saw evidence of an interaction between age and minutes of exercise on blood pressure.

Here are the first few rows of the data again:

```
         bp      age mins_per_week age_class mins_per_week_class
1  61.58319 35.93052     130.94479     30-40             120-140
2  83.86716 42.32743      70.63945     40-50               60-80
3  93.01438 54.37120      54.05203     50-60               40-60
4  82.14413 74.49247     198.53681     70-80             180-200
5  60.13102 32.10092     126.69865     30-40             120-140
6 102.00306 73.90338      42.64163     70-80               40-60
```

Here is the data in a graph, with age represented by a colour gradient:

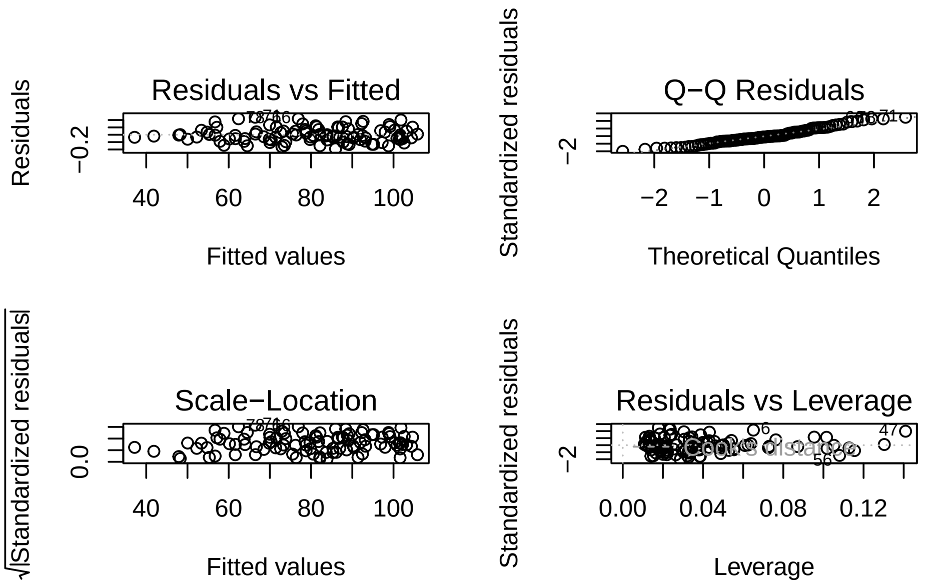


We can fit a multiple regression model with an interaction term in R as follows:

```r
mod_2cont <- lm(bp ~ mins_per_week * age, data=bp_2cont)
```

We check the model assumptions:

```r
par(mfrow=c(2,2))
plot(mod_2cont, add.smooth=FALSE)
```



These look ok.

Now we do the F-test for the interaction term:

```
anova(mod_2cont)
```

```
Analysis of Variance Table

Response: bp
                  Df  Sum Sq Mean Sq F value    Pr(>F)    
mins_per_week      1 14307.2 14307.2 1534142 < 2.2e-16 ***
age                1 10220.0 10220.0 1095879 < 2.2e-16 ***
mins_per_week:age  1  1764.8  1764.8  189242 < 2.2e-16 ***
Residuals         96     0.9     0.0                      
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

It looks the same as in the two-way ANOVA case! This is because we still have two variables and their interaction, so the ANOVA table has one row for each main effect (mins_per_week and age), one row for the interaction effect (mins_per_week:age), and one row for the residuals.

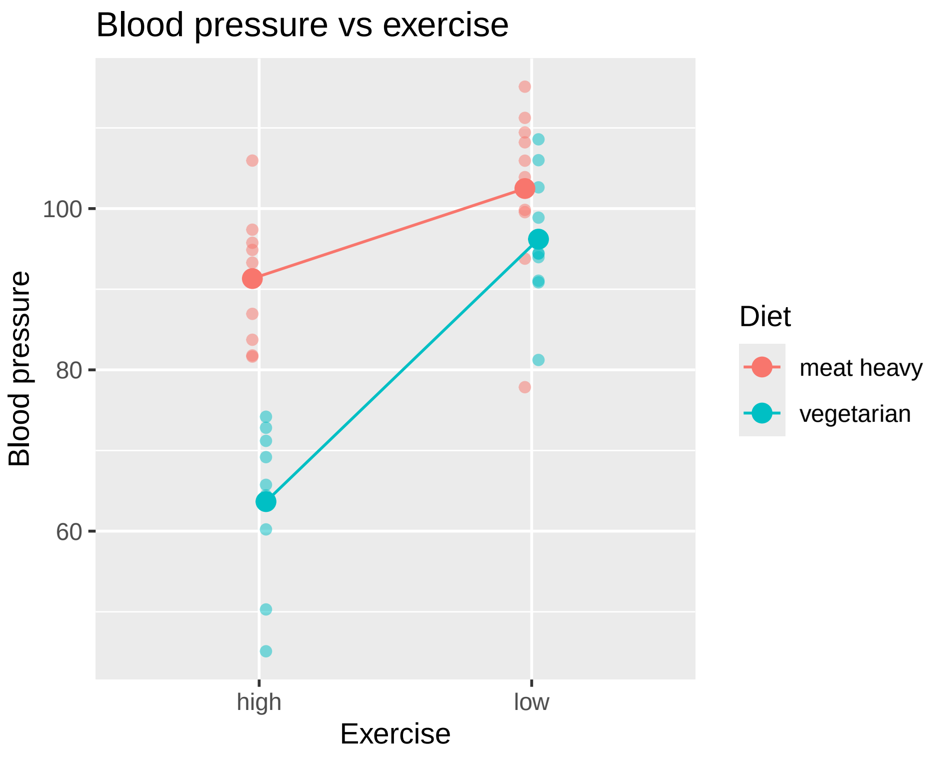
The F-statistic for the interaction term is very large (1.8924178×10^5) and the corresponding p-value is very small (5.6×10^{-160}). Hence, we conclude that there is very strong evidence of an interaction between minutes of exercise and age on blood pressure. The effect of minutes of exercise on blood pressure depends on age, with a stronger effect for younger people compared to older people.

Intepreting main effects and interaction effects

The term *main effect* refers to the individual effect of each categorical explanatory variable on the response variable, ignoring the other variable. For example, the main effect of diet would be the difference in blood pressure between meat heavy and vegetarian diets, averaged over both levels of exercise.

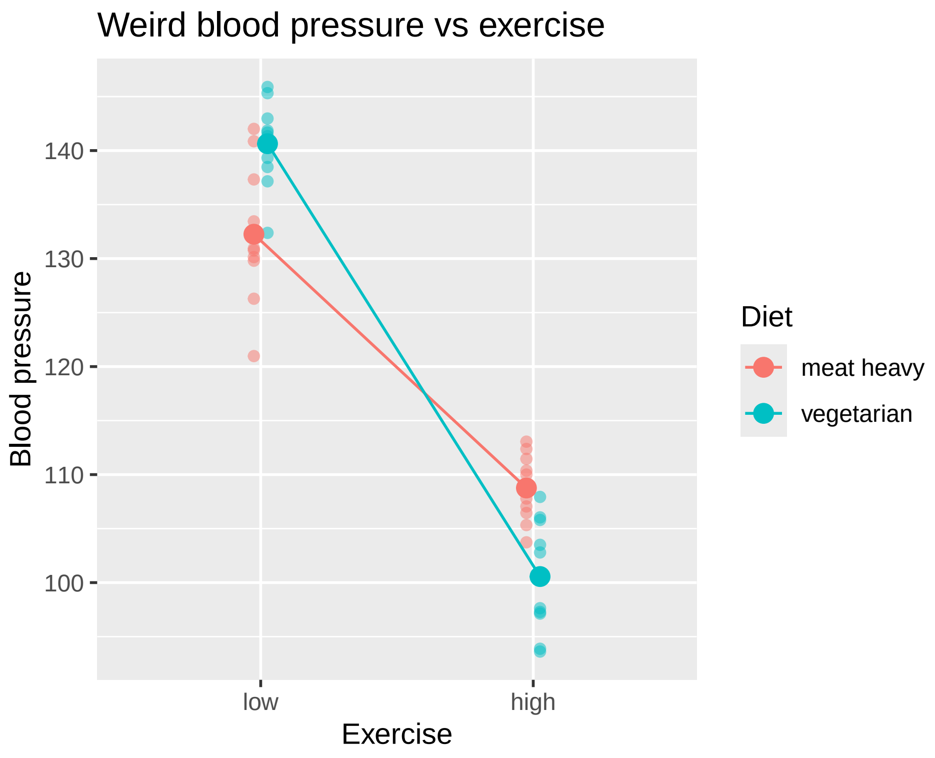
However, if there is an interaction between the two categorical explanatory variables, the main effects may not fully capture the relationship. The interaction effect indicates that the effect of one categorical variable on the response variable depends on the level of the other categorical variable. For example, the effect of diet on blood pressure may differ between low and high exercise groups.

We can see this in the example:



There is a main effect of diet on blood pressure, as well as a main effect of exercise on blood pressure. However, there is also an interaction effect between diet and exercise on blood pressure, as the effect of diet on blood pressure depends on the level of exercise.

It could look otherwise. For example, if we found a (albeit rather unlikely pattern) of higher blood pressure for vegetarians how exercise little, and lower blood pressure for vegetarians who exercise a lot, we would have a pattern as follows:



Here there is no main effect of diet on blood pressure, as the average blood pressure for meat heavy and vegetarian diets is the same when averaged over both levels of exercise. There is still a main effect of exercise on blood pressure, as blood pressure is lower for high exercise compared to low exercise. And there is still an interaction effect between diet and exercise on blood pressure, as the effect of diet on blood pressure depends on the level of exercise.

Take home: Interactions are very interesting, but also will require careful and nuanced interpretation.

More than two explanatory variables

All the examples above had two explanatory variables. What if we have more than two explanatory variables? The principles are the same, but the models and interpretations become more complex. For example, if we have three categorical explanatory variables (A, B, C), we can fit a model with main effects of A, B, and C, as well as all possible interaction effects (A:B, A:C, B:C, A:B:C). The ANOVA table will have one row for each main effect, one row for each two-way interaction effect, one row for the three-way interaction effect, and one row for the residuals.

Interpretation of three-way interactions can be quite complex, as it involves understanding how the effect of one variable on the response variable depends on the levels of the other two categorical variables. Very careful consideration and visualization of the data are often necessary to fully understand and communicate the results.

Review

- Interactions are some of the most interesting effects in biology and medicine. They occur when the effect of one explanatory variable on the response variable depends on the level of another explanatory variable.
- Interactions can occur between continuous and categorical explanatory variables, or between two categorical explanatory variables, or between two continuous explanatory variables.
- Visualization is key to understanding interactions. Use graphs to explore and communicate interactions.
- Hypothesis testing for interactions is done using F-tests on the interaction terms in linear models.
- The degrees of freedom for interaction terms depend on the levels of the categorical variables involved. Each categorical variable takes degrees of freedom equal to the number of levels minus one. Each continuous variable takes one degree of freedom. The degrees of freedom for the interaction term is the product of the degrees of freedom of the individual variables.
- Report interactions carefully, focusing on the biological interpretation and including relevant statistics (i.e., the F-statistic, degrees of freedom for the interaction term, degrees of freedom for error, and p-value).

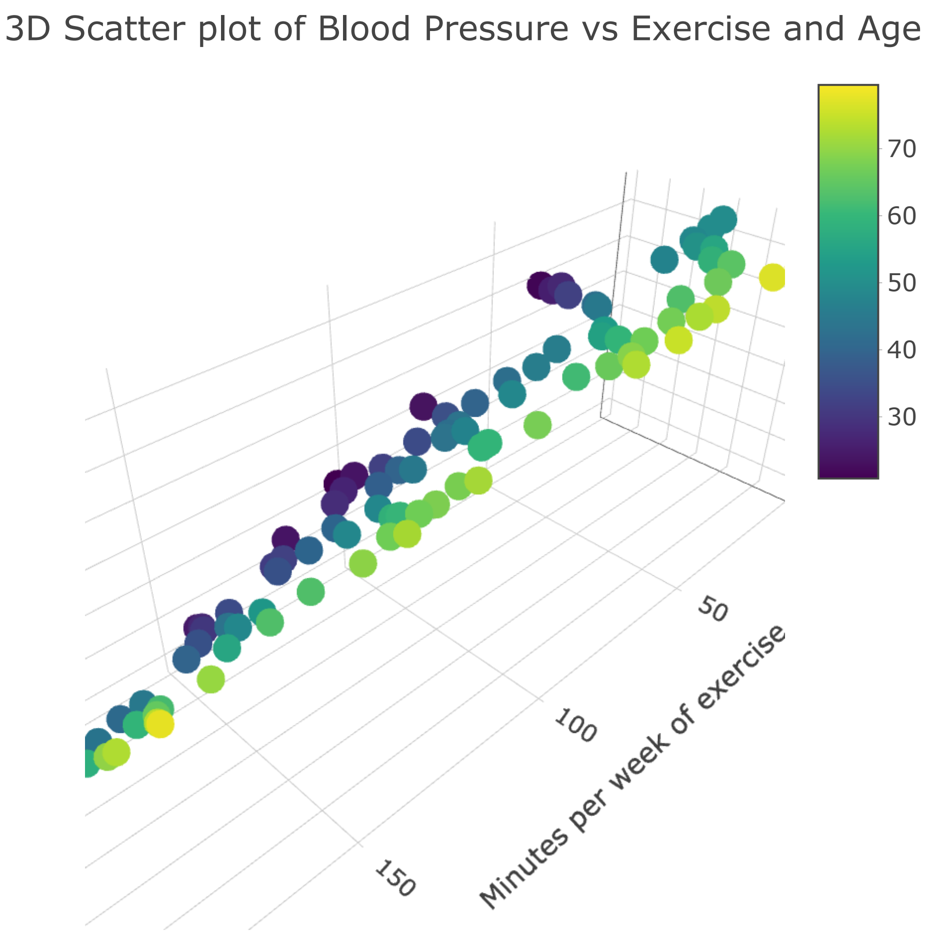
Extras

3d scatter plot

Here is a 3D scatter plot of the data with two continuous explanatory variables (age and minutes of exercise) and one continuous response variable (blood pressure). This plot helps to visualise the interaction between age and minutes of exercise on blood pressure.

(Please note that this plot is best viewed in the HTML version of the book; in the PDF version, it will appear as a static image.)

```
plotly::plot_ly(bp_2cont, x = ~mins_per_week, y = ~age, z = ~bp, type = "scatter3d", mo
  plotly::layout(title = "3D Scatter plot of Blood Pressure vs Exercise and Age",
         scene = list(xaxis = list(title = "Minutes per week of exercise"),
                      yaxis = list(title = "Age"),
                      zaxis = list(title = "Blood Pressure"))
         )
```



Types of sums of squares

❗ Important

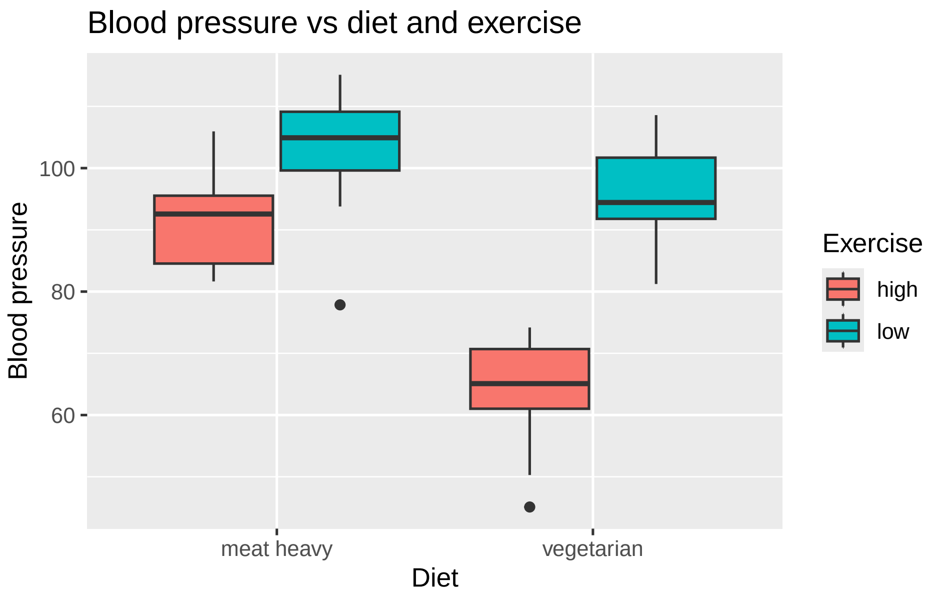
A note on anova() and "Type I" (sequential) sums of squares
In this course we often use `anova()` to test terms in a linear model. In base R, `anova()` for an `lm` uses Type I (sequential) sums of squares. That means the test for each term is done in the order the terms appear in the model formula: each term is tested after the terms before it have already been included.
In balanced designs (equal sample sizes in each group/combination), the order usually does not matter much. But in unbalanced designs (unequal sample sizes), the p-value for a " "main effect" or an interaction can change if you change the order of terms in the model formula.

This is related to collinearity between explanatory variables. When explanatory variables are correlated, the variance they explain in the response variable overlaps. In that case, the order of terms matters because earlier terms get to "claim" more of the shared variance.
So: when you see an ANOVA table from `anova()`, remember that it is testing terms sequentially, not "all at once".

Box and whisker plot for two cats

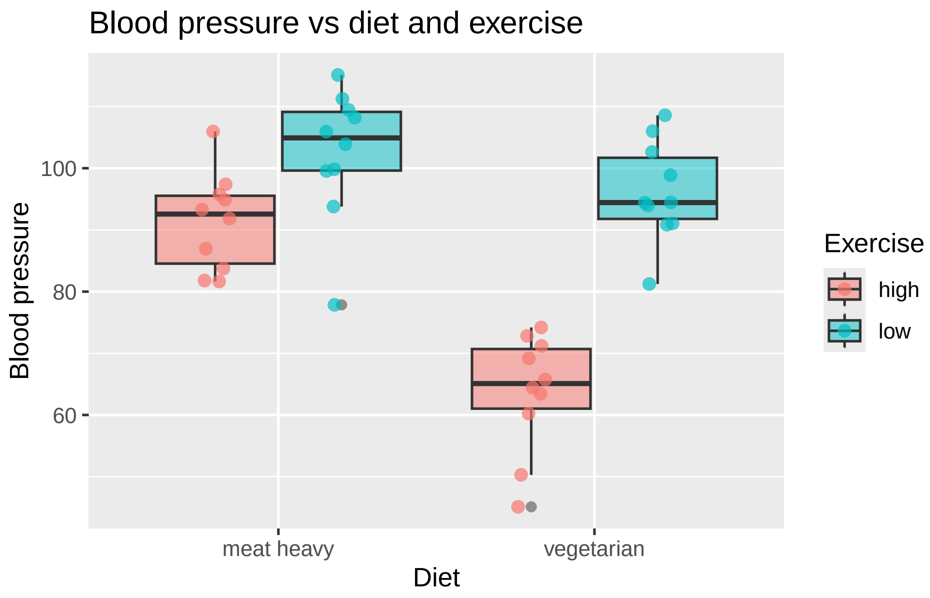
Often you will see box and whisker plots used to visualise data with two categorical explanatory variables. Here is how to make such a plot in R:

```
ggplot(bp_2cat, aes(x=diet, y=bp, fill=exercise)) +
  geom_boxplot(position=position_dodge(width=0.8)) +
  labs(title="Blood pressure vs diet and exercise",
       x="Diet",
       y="Blood pressure",
       fill="Exercise")
```



I (Owen) prefer to show the individual data points whenever possible, so here is a box and whisker plot with the individual data points overlaid:

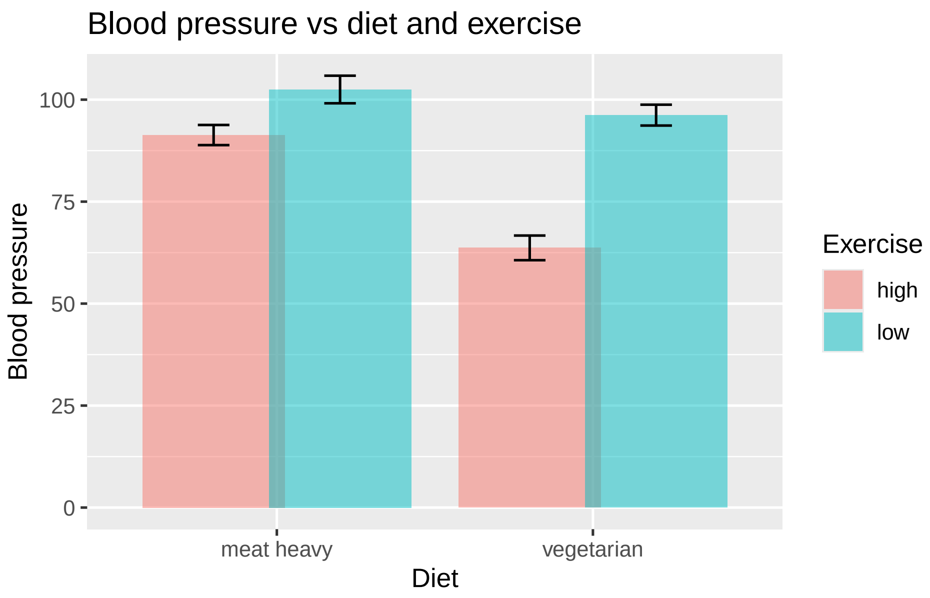
```
ggplot(bp_2cat, aes(x=diet, y=bp, fill=exercise)) +
  geom_boxplot(position=position_dodge(width=0.8), alpha = 0.5) +
  geom_jitter(aes(col=exercise), position=position_jitterdodge(jitter.width=0.2, dodge
  labs(title="Blood pressure vs diet and exercise",
       x="Diet",
       y="Blood pressure",
       fill="Exercise",
       col="Exercise")
```



I, and others, prefer to not use bar plots with error bars to visualise data with categorical explanatory variables. Bar plots hide the individual data points and can be misleading. Box and whisker plots are better, but still hide some of the data. Scatter plots with jittered points are often the best way to visualise such data.

In case you must use bar plots with error bars, here is how to make one in R:

```
grouped_data <- group_by(bp_2cat, diet, exercise) %>%
  summarise(mean_bp = mean(bp), sd_bp = sd(bp), n = n(), se_bp = sd_bp / sqrt(n), .groups = "drop
ggplot(grouped_data, aes(x=diet, y=mean_bp, fill=exercise)) +
  geom_bar(stat="identity", position=position_dodge(width=0.8), alpha = 0.5) +
  geom_errorbar(aes(ymin=mean_bp - se_bp, ymax=mean_bp + se_bp), position=position_dodge(width=0.
  labs(title="Blood pressure vs diet and exercise",
       x="Diet",
       y="Blood pressure",
       fill="Exercise")
```


Blood pressure vs diet and exercise
Blood pressure
100
75
50
25
0
meat heavy
vegetarian
Diet
Exercise
high
low

Count data (L8)

In all the previous chapters, we have focused on **linear models** for continuous response variables. However, many biological response variables are counts (e.g., number of offspring, number of parasites). In this chapter, we introduce **Generalized Linear Models (GLMs)** for analysing count data, focusing on the Poisson distribution.

Introduction

So far in BIO144, we have focused on **linear models** fitted using `lm()`. These models assume:

- A **continuous response variable**
- Normally distributed residuals
- Constant variance (homoscedasticity)

Linear models are powerful, but these assumptions are often violated in biological data.

In this chapter we move beyond normal linear models to handle an important new type of response variable: **counts**.

We introduce **Generalized Linear Models (GLMs)**, which extend linear models by allowing the response variable to follow distributions other than the normal distribution.

By the end of this chapter, you should be able to:

- Recognise when linear regression is inappropriate
- Understand the core components of a GLM
- Fit and interpret Poisson regression models
- Diagnose common problems such as overdispersion and zero inflation

Think–Pair–Share (#tps-non-neg-kinds) What kinds of biological outcomes (response variables) can you think of that *cannot* be negative or continuous?

From LM to GLM

Linear models (`LM`) describe the **mean of a response variable** as a linear function of explanatory variables. For example:

$y_i = \beta_0 + \beta_1 x_i^{(1)} + \cdots + \beta_p x_i^{(p)} + \epsilon_i$

where the error term ϵ_i is normally distributed with mean 0 and constant variance. That is:

$\epsilon_i \sim N(0, \sigma^2)$.

This works well when the response variable is continuous and approximately normally distributed.

However, many biological response variables are:

- **Counts** (e.g. offspring, parasites, species)
- **Binary responses** (e.g. alive/dead)
- **Proportions**

In these cases, forcing the data into a linear model often leads to invalid predictions and misleading inference.

Generalized Linear Models (GLMs) solve this problem by:

- Keeping the familiar *linear predictor*, but
- Allowing different distributions for the response, and
- Linking the predictor to the mean response using a *link function*.

! Important

Key idea
GLMs are not a replacement for linear models — they are a *generalisation* of them. Linear regression is a special case of a GLM.

Think–Pair–Share (#tps-which-reg-assump) Which assumption of linear regression do you think is most problematic for count data?

Count data

Count data occur frequently in biology and medicine. Typical examples include:

- Numbers of animals, plants, or species
- Numbers of offspring
- Numbers of pathological structures (e.g. polyps)

Count data have four key properties:

1. They are **discrete**
2. They are **non-negative**
3. Their **variance often increases with the mean**

4. There is no known upper limit to the count.

These properties immediately suggest that standard linear regression may be inappropriate.

Think–Pair–Share (#tps-which-props-viol) Which of these properties is violated if we use a normal distribution for counts?

Example: Soay sheep

A feral population of Soay sheep on the island of Hirta (Scotland) has been studied extensively. Ecologists were interested in whether the **body mass of female sheep** influences their fitness, measured as **lifetime reproductive success** (number of offspring produced over a lifetime).

Question: Are heavier females fitter than lighter females?

Read in an example dataset:

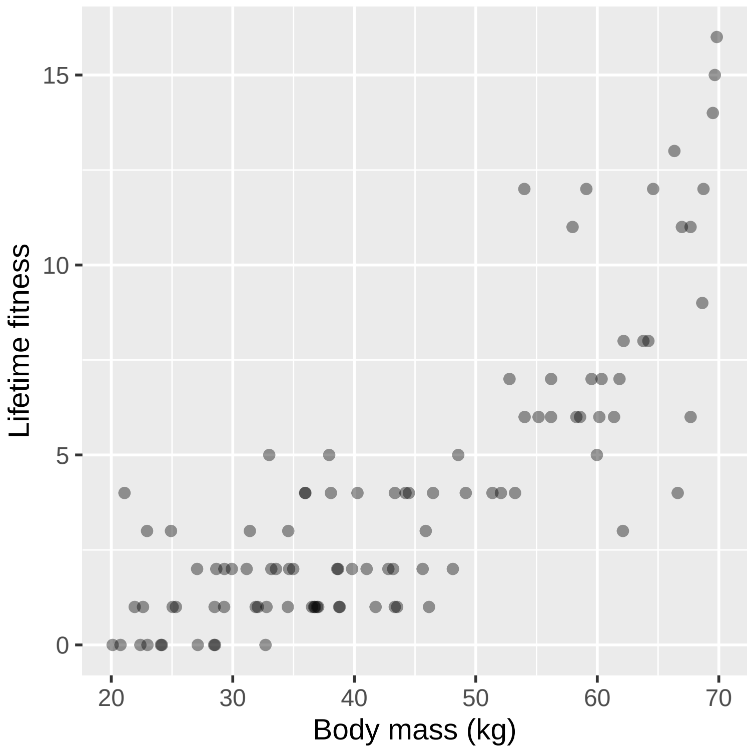
```
soay <- read.csv("datasets/soay_sheep.csv")
```

Here are the first few rows of the dataset:

```
head(soay)
```

```
  body.size fitness
1  53.99529      12
2  32.69467       0
3  33.55580       2
4  43.20078       2
5  43.34102       4
6  59.52711       7
```

As always, we start by exploring the data visually:



Think–Pair–Share (#tps-what-suspicious) What features of this plot might already make you suspicious about using linear regression?

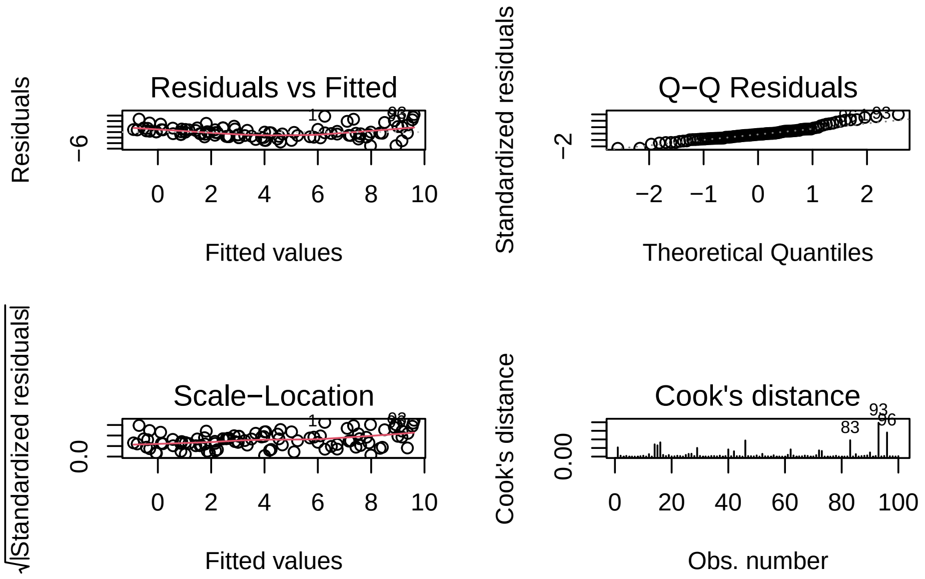
The wrong analysis

A common mistake is to analyse count data using linear regression, treating counts as if they were continuous.

```r
mod_soay_lm <- lm(fitness ~ body.size, data = soay)
```

The model checking plots show clear violations of linear regression assumptions:

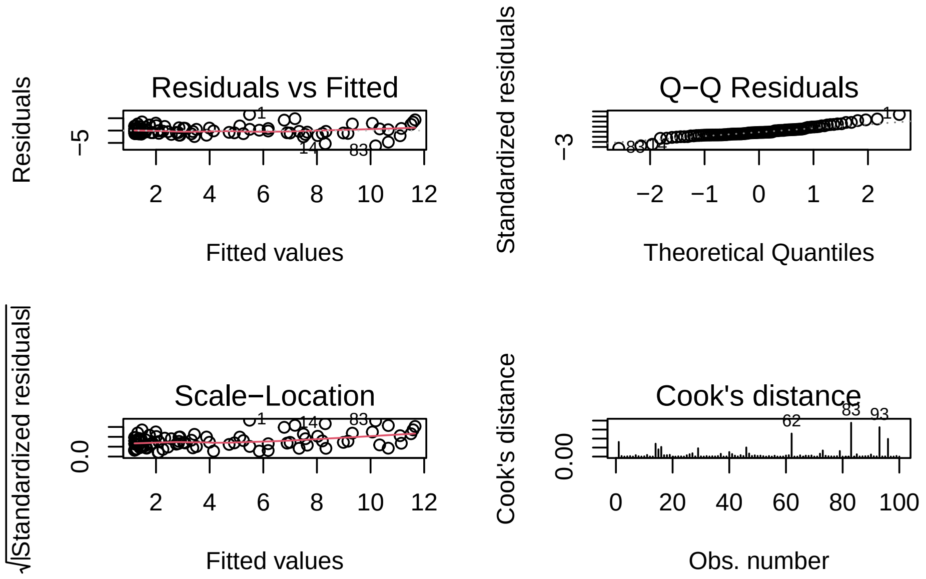
```r
par(mfrow = c(2, 2))
plot(mod_soay_lm, which = 1:4, add.smooth = TRUE)
```



The qq-plot looks fine. The scale-location plot shows that variance increases with fitted values, violating homoscedasticity. Also, there is a clear non-linear pattern in the residuals vs fitted plot, suggesting that the linear model is not capturing the relationship well.

Adding a quadratic term improves the fit slightly:

```
mod2_soay_lm <- lm(fitness ~ body.size + I(body.size^2), data = soay)
par(mfrow = c(2, 2))
plot(mod2_soay_lm, which = 1:4, add.smooth = TRUE)
```

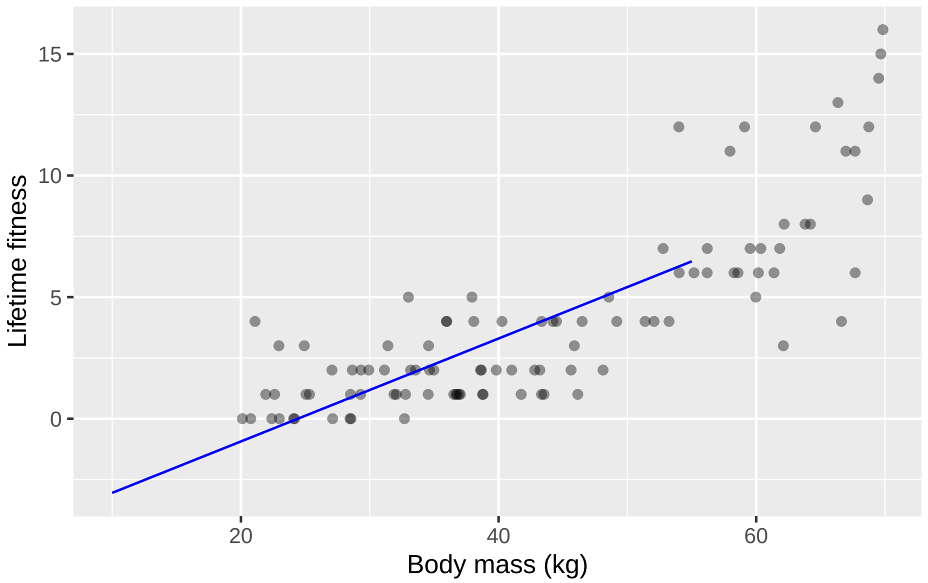


But problems remain: variance increases with fitted values.

Caution

Why this matters
Violating model assumptions can lead to biased estimates, incorrect standard errors, and misleading p-values.

Another issue remains, however: linear regression can predict negative counts, which are impossible. We can see this in a plot of the data and the fitted regression line:



Notice that for small body sizes the fitted line goes below 0 on the y-axis. That is, the moe predicts negative fitness values, which are biologically impossible.

Caution

We have extrapolated beyond the data range here, largely for illustrative purposes. In practice we should be very cautious about extrapolating beyond the range of observed data. This is because the model may not hold outside the observed range, leading to nonsensical predictions.

Why linear regression fails for count data

- The normal distribution is for continuous variables
- It allows negative values
- It assumes constant variance
- Count data are discrete, non-negative, and typically heteroscedastic

Think–Pair–Share (#a_why_poisson) Why is a normal distribution a poor choice for count data? Which assumptions can be expected to fail?

Poisson GLM

To deal with count data, we need a different probability (error) model. A common probability model for counts is the **Poisson distribution**. The Poisson distribution is often the default starting point for modelling counts because it is the simplest distribution that respects discreteness, non-negativity, and increasing variance.

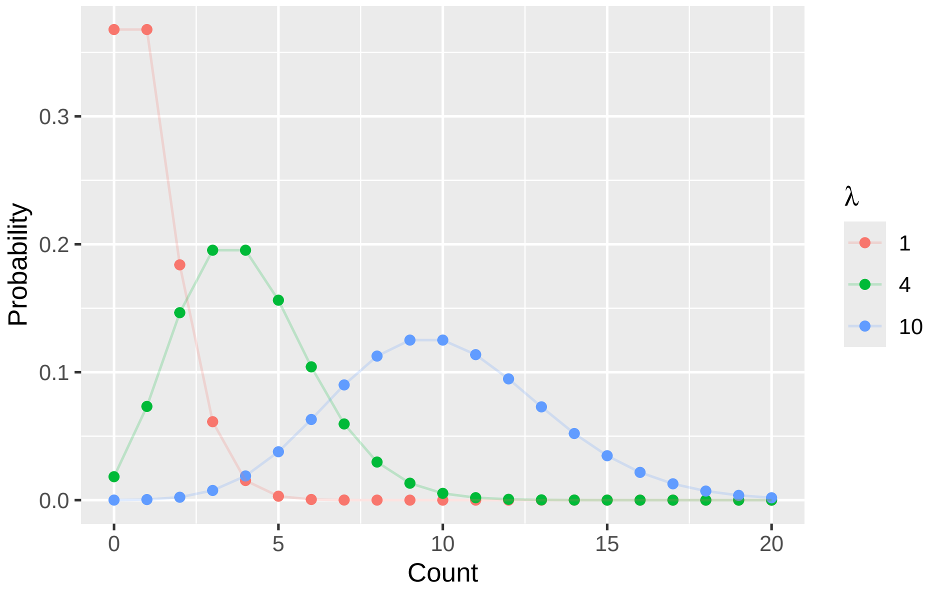
The Poisson distribution has the following probability mass function:

$P(Y = y) = \frac{\lambda^y e^{-\lambda}}{y!}$

where:

- $y = 0, 1, 2, ...$
- $\lambda > 0$ is the mean (and variance) of the distribution

Here are a couple of Poisson distributions with different means:



The Poisson distribution has two important properties:

1. It is defined only for **non-negative integers** (0, 1, 2, …)
2. The **mean and variance are equal**.

! Important

Mean–variance relationship
In a Poisson distribution, the mean and variance are equal. This captures an important feature of count data, but it will later lead to the concept of *overdispersion*.

Think–Pair–Share (#a_mean_variance_link) Why does increasing variance with the mean cause problems for linear regression?

Generalized Linear Models (GLMs)

GLMs extend linear models by combining three components:

1. **Linear predictor**.
2. **Link function**.
3. **Probability distribution (family)**.

Note

If you choose a normal family and an identity link, a GLM is mathematically identical to the linear model you fit using `lm()`.

The linear predictor

The linear predictor is the same as in linear regression:

$$\eta_i = \beta_0 + \beta_1 x_i^{(1)} + \cdots + \beta_p x_i^{(p)}$$

It is just the linear combination of explanatory variables and coefficients. Nothing new here.

The link function

In the linear models we already used (e.g., for linear regression) the link function is the **identity link**:

$$E(y_i) = \eta_i$$

Here, $E(y_i)$ refers to the expected value (mean) of many hypothetical observations, not the single observed value y_i.

Identity link: the expected value of the response $E(y_i)$ equals the linear predictor η_i.

That is:

$$E(y_i) = \eta_i = \beta_0 + \beta_1 x_i^{(1)} + \cdots + \beta_p x_i^{(p)}$$

For count data, this can lead to negative predictions.

The solution is a different **link function**. For Poisson regression, the standard choice is the **log link**. It relates the linear predictor to the expected value of the response as follows:

$$\eta_i = \log(E(y_i))$$

which implies:

$$E(y_i) = \exp(\eta_i) > 0$$

The log link ensures that the expected count is always positive, regardless of the values of the explanatory variables. Whatever the value of the linear predictor η_i, the exponential of it $\exp(\eta_i)$ is always positive.

A good link function enforces the natural constraints of the data.

Think–Pair–Share (#tps-what-wrong-id) What would go wrong if we used an identity link for count data?

The probability distribution (family)

The final component of a GLM is the **probability distribution** (also called the **family**). This specifies how the response variable is distributed. In Poisson regression, we assume that the response variable follows a Poisson distribution. Hence, we say we are fitting a **Poisson GLM**.

Mathematically, a Poisson GLM can be summarised as:

$y_i \sim \text{Poisson}(\lambda_i)$

And written out fully with the linear predictor and link function:

$y_i \sim \text{Poisson}(\lambda_i)$

where $\log(\lambda_i) = \beta_0 + \beta_1 x_i^{(1)} + \cdots + \beta_p x_i^{(p)}$

or equivalently:

$E(y_i) \sim \text{Poisson}(\exp(\beta_0 + \beta_1 x_i^{(1)} + \cdots + \beta_p x_i^{(p)}))$

R - Poisson GLM

When fitting a Poisson GLM in R, we use the `glm()` function, specifying the family as `poisson`:

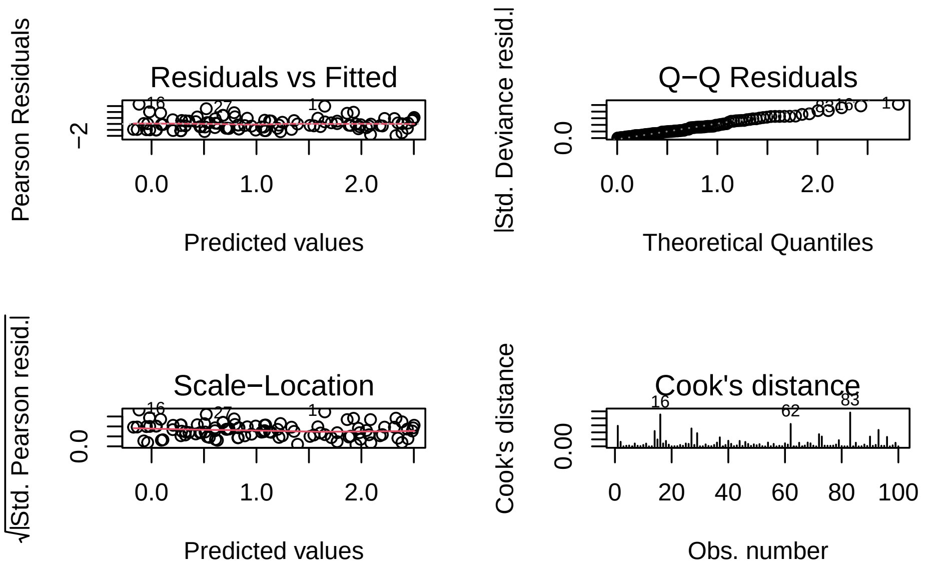
```r
soay_glm <- glm(fitness ~ body.size, data = soay, family = poisson)
```

Specifying `family = poisson` tells R to use the Poisson distribution with a log link function by default. We could specify the link explicitly as `family = poisson(link = "log")`, but this is unnecessary since the log link is the default for the Poisson family.

Checking model assumptions

As always, we should check model assumptions:

```r
par(mfrow = c(2, 2))
plot(soay_glm, which = 1:4, add.smooth = TRUE)
```



The QQ-plot is rather concerning. However, QQ-plots in GLMs are not testing normality of residuals in the same way as for linear models, so their interpretation differs. The deviance residuals should approximately follow a normal distribution if the model fits well. Here, there are some deviations from normality, which could be somewhat concerning, but for now we will focus on the other plots.

The other plots look much better than for the linear model. The residuals vs fitted plot shows no obvious pattern, and the scale-location plot shows more constant variance.

! Important

You must specify the **family** in `glm()`. If you omit the link function, R uses the default link for that family.

Interpreting coefficients

```
summary(soay_glm)
```

```

Call:
glm(formula = fitness ~ body.size, family = poisson, data = soay)

Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept) -1.253092   0.210186  -5.962 2.49e-09 ***
```

```
body.size    0.053781   0.003735  14.400  < 2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for poisson family taken to be 1)

    Null deviance: 333.105  on 99  degrees of freedom
Residual deviance:  97.783  on 98  degrees of freedom
AIC: 378.16

Number of Fisher Scoring iterations: 5
```

Coefficients are estimated on the **log scale**.

! Important

A one-unit increase in an explanatory variable multiplies the expected count by $\exp(\beta)$.

Think–Pair–Share (#tps-what-change-beta) If $\beta = 0.05$ what change in the expected count would be caused by an increase in body mass of 1 kg? As well as a numeric value, specify whether it is used additively or multiplicatively.

To interpret the coefficient for body size ($\beta_1 = 0.05$), we exponentiate it: $\exp(0.05) \approx 1.65$

This means that for each additional kilogram of body size, the expected count of fitness increases by a factor of approximately 1.65 (i.e., a 65% increase). We use the number multiplicatively because of the log link function.

To make a prediction of the expected count for a given body size, we back-transform the linear predictor:

$E(y) = \exp(\beta_0 + \beta_1 \times \text{body.size})$

For example, for a body size of 40 kg:

```r
body_size_example <- 40
linear_predictor <- coef(soay_glm)[1] + coef(soay_glm)[2] * body_size_example
expected_count <- exp(linear_predictor)
expected_count
```

```
(Intercept)
   2.455011
```

Note

Although counts can only be integers, the expected value from a Poisson model can be any positive real number. This is because the expected value is a mean over many possible counts.

Analysis of deviance

There is something technically different that we have glossed over until now: how model fit is assessed. In linear regression we estimate parameters by minimizing the sum of squared residuals. In GLMs, we use **maximum likelihood estimation (MLE)**. In this course we will not go into the mathematical details of MLE, but the key idea is that we find the parameter values that make the observed data most probable under the assumed model (just like in least squares). Instead of minimising sums of squares, we maximise the **likelihood** of the data given the model. Maximising the likelihood is equivalent to minimising the **deviance**, which is a measure of model fit based on likelihoods. Hence, when we fit a GLM in R, we get output including the **deviance** (and not sums of squares):

```r
anova(soay_glm, test = "Chisq")
```

```
Analysis of Deviance Table

Model: poisson, link: log

Response: fitness

Terms added sequentially (first to last)

          Df Deviance Resid. Df Resid. Dev  Pr(>Chi)
NULL                         99     333.11
body.size  1   235.32        98      97.78 < 2.2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

In the output we see the deviance for the null model and the fitted model, as well as the change in deviance when adding explanatory variables. We can use a chi-squared test to assess whether adding explanatory variables significantly improves model fit.

We use a chi-squared test here because, under certain regularity conditions, the change in deviance between nested models follows a chi-squared distribution with degrees of freedom equal to the difference in the number of parameters. If you are interested in what this means, you can read more about it in more advanced statistics textbooks.

Reporting

When reporting results from a Poisson GLM, we can make a graph and a sentence describing the pattern and related statistics.

When we wish to make a graph of the fitted relationship, we can either make it with the y-axis on the log scale (linear predictor scale) or back-transform to the original count scale. I usually prefer the original scale, as it is easier to interpret.

The first step is to create a new data frame with a sequence of body sizes for prediction, and errors for the 95% confidence intervals:

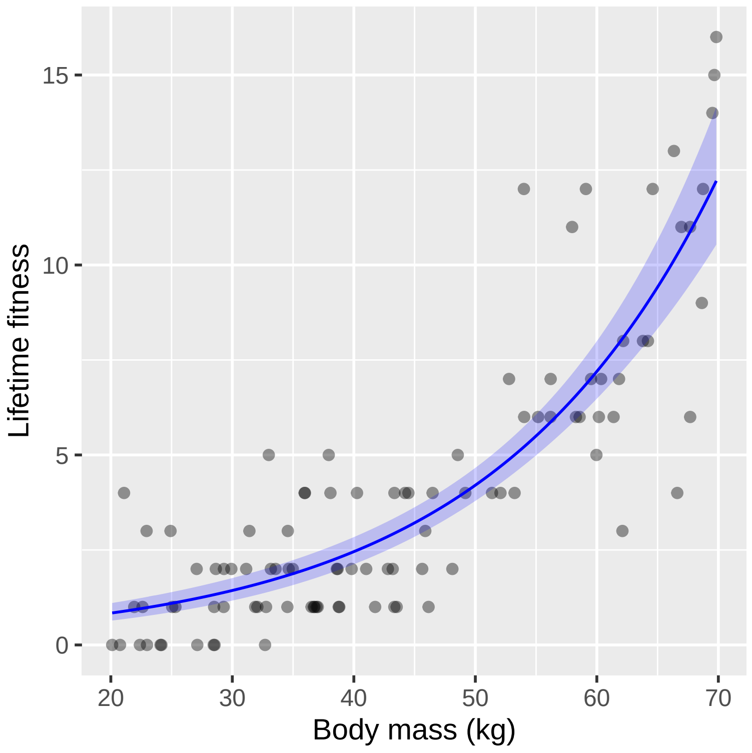
```r
new_data <- data.frame(body.size = seq(min(soay$body.size), max(soay$body.size), lengt
predictions <- predict(soay_glm, newdata = new_data, se.fit = TRUE)
new_data$fit <- exp(predictions$fit)
new_data$lower <- exp(predictions$fit - 1.96 * predictions$se.fit)
new_data$upper <- exp(predictions$fit + 1.96 * predictions$se.fit)
```

Note that we back-transform the predictions and confidence intervals using the exponential function, since the link function is the log. And note that we must calculate the confidence intervals on the log scale first, and then back-transform them.

Note

If we only wanted the fitted value (and not confidence intervals) we could have used `type = "response"` in the `predict()` function to get predictions on the original count scale (back-transformed from the log scale). When we don't specified this, we would get predictions on the log scale–this is the default behavior.

Now we can plot the data and the fitted relationship with confidence intervals:



Excellent! We can now write a nice sentence for our results:

Reproductive fitness (in terms of lifetime number of offspring) increased significantly with body mass, with a unit increase in body mass associated with a multiplicative increase in expected fitness of 1.06 (95% CI: 1.05, 1.06; $\chi^2 = 235.32$, $df = 1$, $p < 4.1 \times 10^{-53}$).

Well done!

We have made our first steps into the world of GLMs and Poisson regression for count data. You should now be able to:

- Recognise why linear regression is often inappropriate for count data.
- Understand the components of a GLM: linear predictor, link function, and family.
- Fit a Poisson GLM in R using `glm()`.
- Interpret coefficients from a Poisson GLM.

Next, we will look at common issues that arise when fitting Poisson models, such as overdispersion and zero inflation.

Overdispersion

In a Poisson model, mean and variance are assumed equal. In practice, variance often exceeds the mean, a phenomenon called **overdispersion**.

Common causes include:

- Unmeasured explanatory variables. This means important explanatory variables are missing from the model, leading to extra variability.
- Individual heterogeneity. This can arise when individuals differ in ways not captured by measured explanatory variables, leading to extra variability in counts.
- Correlated observations. This can occur when observations are not independent, such as repeated measures on the same individual.

A simple check for overdispersion is to compare the residual deviance to the residual degrees of freedom:

$\text{Dispersion} = \frac{\text{Residual Deviance}}{\text{Residual DF}}$

If this ratio is substantially greater than 1 (e.g., > 1.5 or 2), it indicates overdispersion.

Check this for our Soay sheep model:

```r
dispersion_soay <- deviance(soay_glm) / df.residual(soay_glm)
dispersion_soay
```

```
[1] 0.9977877
```

Here, the dispersion is quite close to 1 (it is 1), indicating little overdispersion. However, in many real datasets, overdispersion is common.

Caution

Ignoring overdispersion leads to **anti-conservative p-values** (too small). This would be a typical example of **Type I error inflation**. Type I errors occur when we incorrectly reject a true null hypothesis, leading to false positives. In the context of statistical modeling, ignoring overdispersion can result in underestimating standard errors, which in turn leads to smaller p-values. Consequently, we may conclude that an effect is statistically significant when it is not, thereby increasing the likelihood of Type I errors.

Think–Pair–Share (#tps-why-ind-diff) Why might individuals differ even after accounting for measured explanatory variables?

Quasi-Poisson and negative binomial models

One solution is the **quasi-Poisson** model, which estimates an additional dispersion parameter:

For this, we can use `family = quasipoisson` in `glm()`:

```r
soay_quasi <- glm(fitness ~ body.size, data = soay, family = quasipoisson)
summary(soay_quasi)
```

```

Call:
glm(formula = fitness ~ body.size, family = quasipoisson, data = soay)

Coefficients:
             Estimate Std. Error t value Pr(>|t|)    
(Intercept) -1.253092   0.209446  -5.983 3.59e-08 ***
body.size    0.053781   0.003722  14.450  < 2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for quasipoisson family taken to be 0.992977)

    Null deviance: 333.105  on 99  degrees of freedom
Residual deviance:  97.783  on 98  degrees of freedom
AIC: NA

Number of Fisher Scoring iterations: 5
```

Think–Pair–Share (#a_overdispersion_consequences) What happens to standard errors and p-values if overdispersion is ignored?

Zero inflation

A special case of overdispersion arises when there are **more zeros than expected** under a Poisson model. This is called **zero inflation**.

Examples include:

- Number of cigarettes smoked (many people do not smoke, more than can be modelled by a Poisson distribution)
- Parasite counts (many host individual have no parasites, more than can be modelled by a Poission distribution)

Note

Zero inflation often reflects **two processes**: whether an observation can be non-zero at all, and how large it is if it is. E.g., whether an individual smokes at all, and then how many cigarettes they smoke if they do.

Think–Pair–Share (#tps-zero-inf-examples) Can you think of an ecological, biological, or medical process that would generate many structural zeros?

In this course we will not describe in detail or practice fitting zero-inflated models, but they are an important tool for count data with many zeros. Examples of models include zero-inflated Poisson (ZIP) and zero-inflated negative binomial (ZINB) models. These models combine a count model (e.g., Poisson or negative binomial) with a separate model for the probability of being a structural zero. There is also a model called the **hurdle model**, which is similar but has a different interpretation.

Multiple explanatory variables

GLMs can include multiple explanatory variables, just like linear models. And they can include only categorical variables, only continuous variables, or a mix of both.

For example, we could include parasite load as an additional predictor of fitness in the Soay sheep data:

Read in the updated dataset:

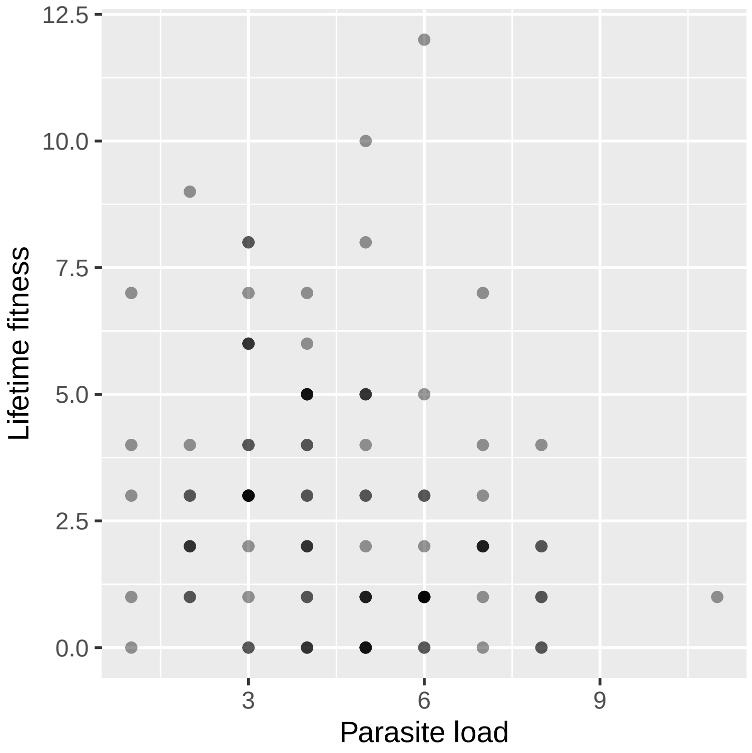
```
soay <- read.csv("datasets/soay_sheep_with_parasites.csv")
```

Here is the first few rows of the updated dataset:

```
head(soay)
```

```
  body.size fitness parasite.load
1  53.99529       1             2
2  32.69467       1             6
3  33.55580       0             5
4  43.20078       1             6
5  43.34102       0             7
6  59.52711       3             6
```

We can visualise the relationship between parasite load and fitness:



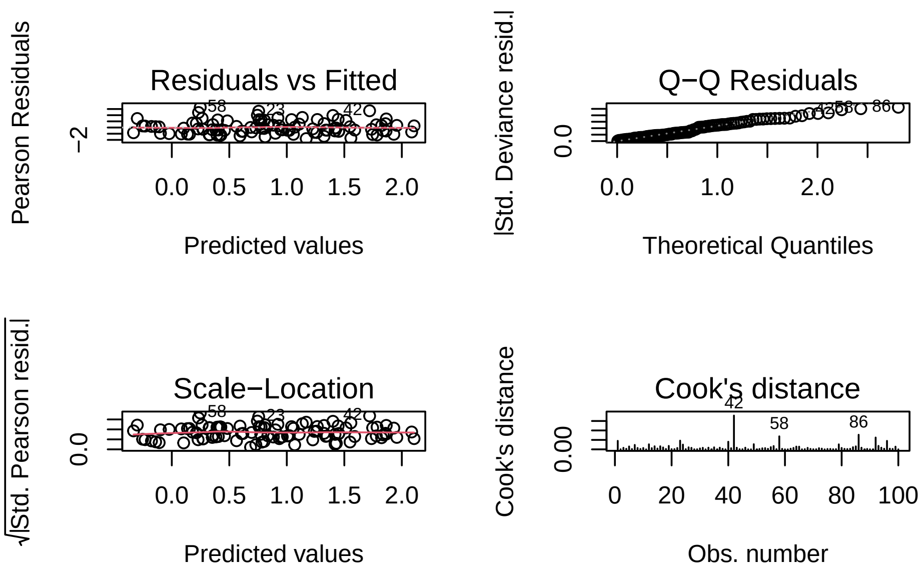
It looks like higher parasite loads are associated with lower fitness.

We can fit a Poisson GLM with both body size and parasite load as explanatory variables:

```
soay_glm2 <- glm(fitness ~ body.size + parasite.load, data = soay, family = poisson)
```

As always we check model assumptions:

```
par(mfrow = c(2, 2))
plot(soay_glm2, which = 1:4, add.smooth = TRUE)
```



The model checking plots look good. We can summarise the model:

```
anova(soay_glm2, test = "Chisq")
```

```
Analysis of Deviance Table

Model: poisson, link: log

Response: fitness

Terms added sequentially (first to last)

              Df Deviance Resid. Df Resid. Dev  Pr(>Chi)
NULL                              99     226.57
body.size      1   95.723        98     130.85 < 2.2e-16 ***
parasite.load  1   11.063        97     119.78 0.0008809 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

We see that both body size and parasite load significantly affect fitness. Body size has a positive effect, while parasite load has a negative effect. These effects are interpreted conditional on the other explanatory variable being held constant, just as in multiple linear regression.

Think–Pair–Share (#tps-how-vis-poisson) How would you visualise (make a graph) of the data and the modelled relationships? Make a sketch of what you could do.

Review

In this chapter we have introduced Generalized Linear Models (GLMs) for count data, focusing on Poisson regression. Key points include:

- Linear regression is often inappropriate for count data due to violations of assumptions.
- GLMs extend linear models by allowing different distributions and link functions.
- Poisson GLMs use the Poisson distribution with a log link to model counts.
- Coefficients are interpreted on the log scale, with exponentiation giving multiplicative effects.
- Overdispersion and zero inflation are common issues that need to be addressed.
- GLMs can include multiple explanatory variables, just like linear models.

With this foundation, you are now equipped to analyse count data using GLMs in R. In the next chapter, we will explore further extensions and applications of GLMs.

Binary data (L9)

Introduction

In the previous chapter we moved from linear models and normal errors to generalized linear models (GLMs) to handle **count responses** with a Poisson GLM. In this chapter we move to another common case: **binary responses**.

Binary response variables take only two values, often coded as 0 and 1:

- 1 = yes / success / present / dead
- 0 = no / failure / absent / alive

The central question becomes:

Which explanatory variables influence the probability $\pi_i = P(y_i = 1)$ of the response variable?

! Important

Bridge from LM to GLM
A GLM keeps the familiar *linear predictor* from linear models, but: 1) uses a distribution (family) that matches the response type, and
2) uses a link function so predictions respect the natural constraints.

- Count data: Poisson family + log link

- Binary data: binomial family + logit link

Think–Pair–Share (#tps-binary-examples) Name three response variables in biology, ecology, or medicine that are naturally binary (0/1). Why would a linear model be risky for each?

Overview

In this chapter we will cover:

- Contingency tables and the χ^2 test (a familiar starting point)

- Odds, odds ratios, and log-odds
- Logistic regression as a binomial GLM
- Interpreting coefficients (odds ratios and probabilities)
- Model assumptions, deviances, and (some) model checking
- Overdispersion in *aggregated* binomial data and the quasibinomial fix
- Practical complications for individual-level (non-aggregated) 0/1 data

A warm-up: the Chi-square test for a 2×2 table

Binary responses often first appear in a **contingency table**. For example:

- Explanatory variable x: hormonal contraception (yes/no)
- Response variable y: heart attack (yes/no)

The χ^2 (chi-squared) test asks whether the **proportion** (or frequency) of heart attacks is the same in the two contraception groups.

A concrete example dataset

We will reconstruct the classic 2×2 table from some example and store it as a dataset:

Read in the heart attack dataset:

```
heart_attack <- read_csv("datasets/heart_attack_2x2.csv")
```

```
Rows: 4 Columns: 3
-- Column specification --------------------------------------------------------
Delimiter: ","
chr (2): contraception, heart_attack
dbl (1): n

i Use `spec()` to retrieve the full column specification for this data.
i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

Here are the first few rows of the heart attack dataset:

```
head(heart_attack)
```

```
# A tibble: 4 x 3
  contraception heart_attack     n
  <chr>         <chr>        <dbl>
1 yes           yes             23
2 yes           no              34
3 no            yes             35
4 no            no             132
```

We can create the table and run the χ^2 test.

Here is the data ready for analysis:

```r
tab <- xtabs(n ~ contraception + heart_attack, data = heart_attack)
tab
```

```
             heart_attack
contraception  no yes
          no  132  35
          yes  34  23
```

And here is the χ^2 test:

```r
chisq.test(tab, correct = FALSE)
```

```

	Pearson's Chi-squared test

data:  tab
X-squared = 8.3288, df = 1, p-value = 0.003902
```

Note

Continuity correction
`chisq.test()` defaults to Yates' continuity correction for 2×2 tables. Here we set `correct = FALSE` to match the classic "hand-calculation" style example.

Think–Pair–Share (#tps-chisq-meaning) What is the null hypothesis of this χ^2 test, in words?

Quantifying an association: risk and odds

If π is the probability of "yes" (e.g. heart attack), then:

- **Risk (probability)** is π
- **Odds** is $\pi/(1-\pi)$

The **odds ratio (OR)** compares the odds in two groups:

Odds Ratio = OR = $\frac{\text{odds in group 1}}{\text{odds in group 2}}$.

The **log odds ratio** is $\log(OR)$. It is 0 when $OR = 1$.

Important

Odds are not probabilities
An odds of 3 means "3 to 1", which corresponds to a probability of $3/(3+1) = 0.75$. Odds and probabilities are linked mathematically, but they are not the same thing.

Think–Pair–Share (#tps-odds-intuition) Why might odds ratios be a convenient effect size in a regression model? (This is a difficult question—think carefully, and Pair!)

From tables to regression models

Contingency tables are a good start, but they are limited:

- they typically compare **groups** (few categorical explanatory variable),
- they don't easily handle **continuous explanatory variables** (e.g. dose),
- they don't naturally generalise to multiple explanatory variables.

When y is binary or binomial, the standard regression approach is **logistic regression**, a **binomial GLM**.

Working example: beetle mortality and insecticide dose (aggregated data)

Eight groups of beetles were exposed to an insecticide dose for 5 hours. For each dose level, we know how many beetles were tested and how many were killed. This is **binomial (aggregated)** data.

Read in the beetle dataset:

```
beetle <- read_csv("datasets/beetle_mortality.csv")
```

```
Rows: 8 Columns: 4
-- Column specification --------------------------------------------------------
Delimiter: ","
dbl (4): Dose, Number_tested, Number_killed, Mortality_rate

i Use `spec()` to retrieve the full column specification for this data.
i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

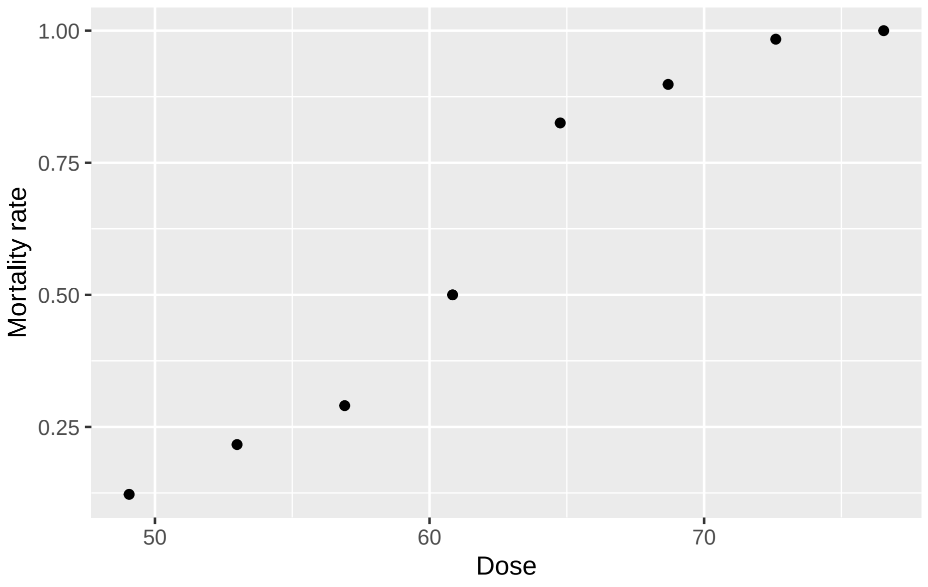
Here are the first few rows of the beetle dataset:

```
head(beetle)
```

```
# A tibble: 6 x 4
   Dose Number_tested Number_killed Mortality_rate
  <dbl>         <dbl>         <dbl>          <dbl>
1  49.1            49             6          0.122
2  53.0            60            13          0.217
3  56.9            62            18          0.290
4  60.8            56            28          0.5
5  64.8            63            52          0.825
6  68.7            59            53          0.898
```

It contains one explanatory variable (Dose) and two pieces of information about the binary response variable (Number_killed and Number_tested). The mortality rate is also included, and is the proportion killed at each dose (Number_killed / Number_tested).

As always, start with a graph.



Think–Pair–Share (#tps-why-lm-wrong) What goes wrong if we fit a linear regression to a probability (mortality rate)? Name at least two problems.

Two tempting but wrong analyses

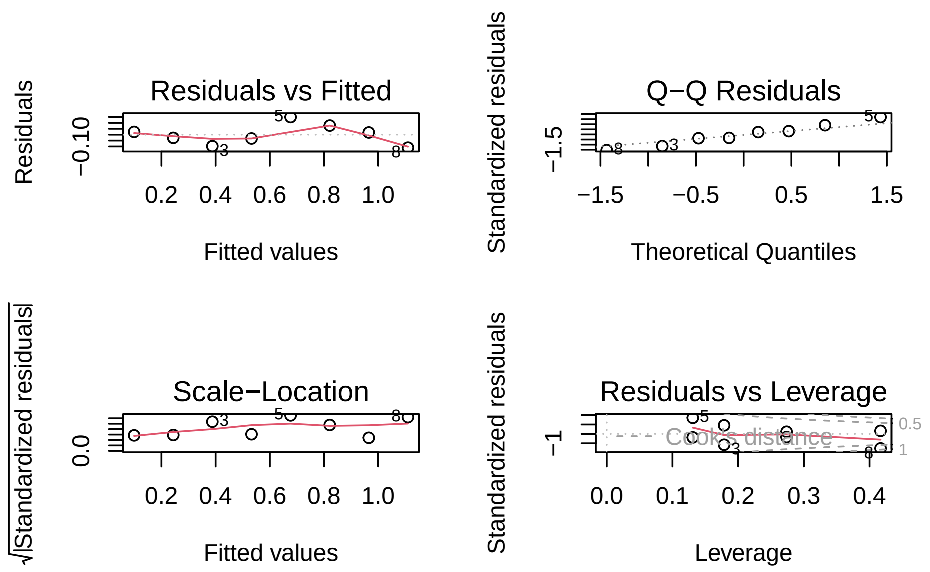
Wrong analysis 1: linear regression on mortality rate

```
mod_beetle_lm <- lm(Mortality_rate ~ Dose, data = beetle)
```

This can predict probabilities below 0 or above 1 and does not respect the mean–variance structure of binomial data.

Check out the model assumptions:

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



We don't have so many data points here, but the residuals vs fitted indicating non-linearity and the Q-Q plot indicating non-normality are both concerning. Also the scale-location plot suggests non-constant variance.

Wrong analysis 2: Poisson regression on counts killed

A Poisson model ignores the fact that deaths are bounded by the number tested at each dose.

```r
mod_beetle_pois <- glm(Number_killed ~ Dose, data = beetle, family = poisson)
summary(mod_beetle_pois)$coefficients
```

```
               Estimate  Std. Error   z value     Pr(>|z|)
(Intercept) -0.77418056 0.494639994 -1.565139 1.175502e-01
Dose         0.06678281 0.007240113  9.224002 2.862145e-20
```

```r
# Example of the "impossible" issue:
dose_example <- 76.54
pred_killed <- predict(mod_beetle_pois,
                       newdata = data.frame(Dose = dose_example),
                       type = "response")
pred_killed
```

```
       1 
76.50652 
```

```r
num_tested_example <- beetle$Number_tested[beetle$Dose == dose_example]
num_tested_example
```

```
[1] 60
```

At the highest dose (76.54), the model predicts about 76 deaths, but only 60 beetles were tested! Clearly this is an impossible prediction. We can't go on with the Poisson model either.

Caution

Poisson vs binomial, when to use each
Use a Poisson model when counts have *no known upper limit.*
Use a binomial model when counts are "k successes out of n trials". Use a binomial model when each observation is a 0/1 response.

Think–Pair–Share (#a__probability__not__linear) Why is it dangerous to model probabilities directly using a linear model?

The probability model: Bernoulli and binomial

Bernoulli (binary) data

For a single 0/1 observation:

$Y \sim \text{Bernoulli}(\pi), \quad P(Y=1)=\pi, \quad P(Y=0)=1-\pi$

In words, this means that the probability of success (Y=1) is π, and the probability of failure (Y=0) is $1-\pi$. And that this probability is Bernoulli distributed. Bernoulli is a special case of the binomial distribution with $n=1$.

Mean and variance are:

$E(Y)=\pi, \quad Var(Y)=\pi(1-\pi)$

Binomial (aggregated) data

For k successes out of n trials:

$Y \sim \text{Binomial}(n, \pi)$

This means that the number of successes Y in n independent Bernoulli trials, each with success probability π, follows a binomial distribution.

Mean and variance are:

$E(Y)=n\pi, \quad Var(Y)=n\pi(1-\pi)$

Note

Key pattern
For Bernoulli/binomial data, the variance is *determined by the mean.* This is one reason "constant variance" fails for linear models.

Think–Pair–Share (#tps-binom-variance) For fixed n, at what value of π is $Var(Y) = n\pi(1-\pi)$ largest? At what values of π is it smallest?

Logistic regression: the binomial GLM

We use the usual GLM structure:

1. **Linear predictor**

$\eta_i = \beta_0 + \beta_1 x_i^{(1)} + \cdots + \beta_p x_i^{(p)}$

2. **Family**: binomial (Bernoulli is the special case $n = 1$)
3. **Link**: logit

$\eta_i = \log\left(\frac{\pi_i}{1-\pi_i}\right)$

The inverse link (back-transformation) is the **logistic function**:

$\pi_i = \frac{\exp(\eta_i)}{1+\exp(\eta_i)}$

! Important

Why the logit link?
It maps probabilities $(0, 1)$ to the whole real line $(-\infty, \infty)$, so the linear predictor can take any value while π_i stays valid, i.e., [0,1].

Think–Pair–Share (#tps-identity-link-binary) What would go wrong if we used the identity link $E(y_i) = \eta_i$ for a binary response variable? And which link function is this, by the way?

Fitting logistic regression in R (aggregated binomial data)

For aggregated binomial data, we provide **successes and failures**:

```r
beetle <- beetle |>
  mutate(Number_survived = Number_tested - Number_killed)
head(beetle)
```

```
# A tibble: 6 x 5
   Dose Number_tested Number_killed Mortality_rate Number_survived
  <dbl>         <dbl>         <dbl>          <dbl>           <dbl>
1  49.1            49             6          0.122              43
2  53.0            60            13          0.217              47
3  56.9            62            18          0.290              44
4  60.8            56            28          0.5                28
5  64.8            63            52          0.825              11
```

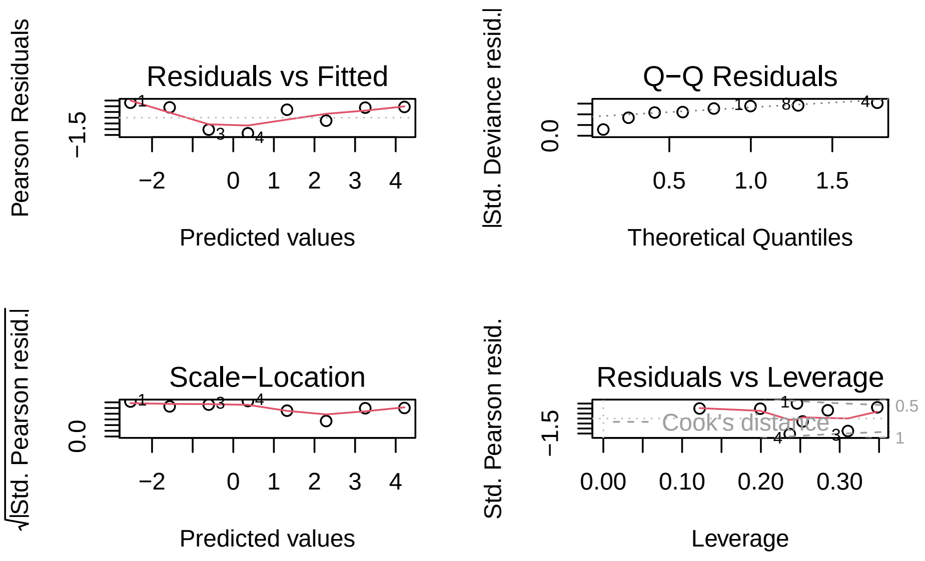
```
6  68.7            59              53            0.898             6
```

Now we can fit the logistic regression model, being careful to specify `family = binomial` and to use `cbind(successes, failures)` for the response. Because we specify the response this way, R knows we are working with aggregated binomial data. And because we specify `family = binomial`, R knows to use the logit link (the default).

```r
beetle_glm <- glm(cbind(Number_killed, Number_survived) ~ Dose,
                  data = beetle, family = binomial)
```

And as usual we can check the model assumptions:

```r
par(mfrow = c(2, 2))
plot(beetle_glm)
```



Not great, but again we have only 8 dose levels. Let us continue in any case and look at the ANOVA table and the coefficients.

! Important

The response in aggregated binomial data
Use `cbind(successes, failures)` in `glm(..., family = binomial)`.

- successes: number of 1s
- failures: number of 0s

Interpreting coefficients: log-odds, odds ratios, and probabilities**

In logistic regression:

- β_1 is a **log odds ratio** for a one-unit change in the linear predictor
- $\exp(\beta_1)$ is the **odds ratio** (multiplicative change in odds)

```
beta1 <- coef(beetle_glm)[["Dose"]]
exp(beta1)
```

```
[1] 1.278311
```

Interpretation: increasing dose by 1 unit multiplies the odds of death by $\exp(\beta_1)$.

Note

Odds vs probability
A constant odds ratio does *not* mean a constant change in probability. The probability change depends on where you start (e.g. $\pi = 0.05$ vs $\pi = 0.80$).

Think–Pair–Share (#tps-or-vs-probability) Why might a treatment have a "big" odds ratio, but only a "small" change in probability in one situation?

Think–Pair–Share (#a_odds_vs_probability) Why might odds be convenient mathematically,
even if probabilities feel more intuitive?

Analysis of deviance (likelihood-based ANOVA)

As for count-data GLMs, we use likelihood and deviance. Nested models can be compared with a χ^2 test.

```
anova(beetle_glm, test = "Chisq")
```

```
Analysis of Deviance Table

Model: binomial, link: logit

Response: cbind(Number_killed, Number_survived)

Terms added sequentially (first to last)

     Df Deviance Resid. Df Resid. Dev  Pr(>Chi)    
NULL                     7    267.662              
Dose  1   259.23         6      8.438 < 2.2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

As with the Poisson GLM, the model is fit using maximum likelihood estimation, and the deviance is used to assess goodness-of-fit. So we see a table quite similar to that for the Poisson GLM. In this case the Dose row shows a huge amount of the deviance is explained by dose, and so the p-value is very small.

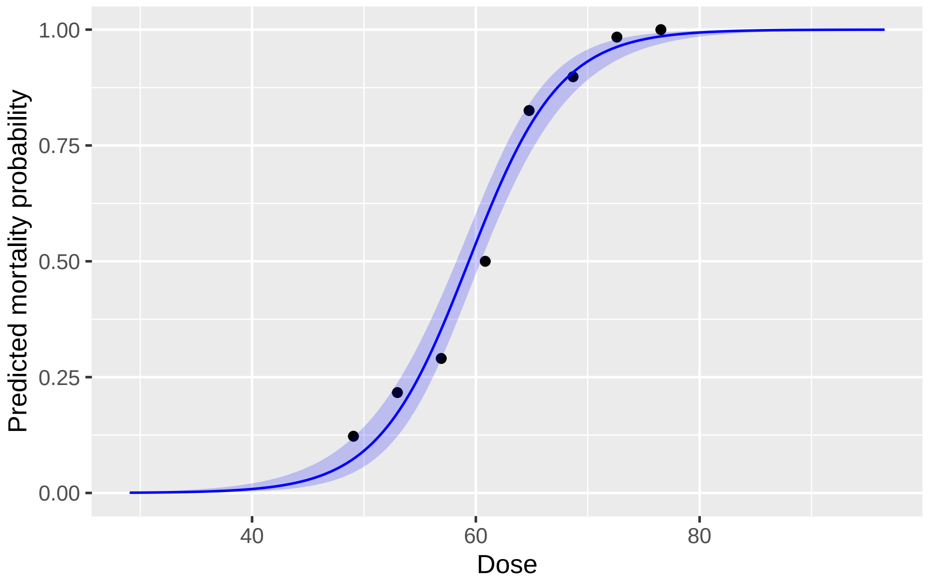
Plotting the fitted relationship

We will plot $P(Y = 1)$ (mortality probability) against dose using model predictions.

```
dose_grid <- tibble(Dose = seq(min(beetle$Dose)-20, max(beetle$Dose)+20, length.out = 400))
preds <- predict(beetle_glm, newdata = dose_grid, se.fit = TRUE) |>
  as_tibble() |>
         mutate(p_hat = exp(fit) / (1 + exp(fit)),
         p_hat_upper_2se = exp(fit + 2*se.fit) / (1 + exp(fit + 2*se.fit)),
         p_hat_lower_2se = exp(fit - 2*se.fit) / (1 + exp(fit - 2*se.fit))
       )
dose_grid <- bind_cols(dose_grid, preds)
```

And now the plot:

```
ggplot() +
  geom_point(data = beetle, aes(x = Dose, y = Mortality_rate)) +
  geom_line(data = dose_grid, aes(x = Dose, y = p_hat), color = "blue") +
  geom_ribbon(data = dose_grid,
              aes(x = Dose, ymin = p_hat_lower_2se, ymax = p_hat_upper_2se),
              alpha = 0.2, fill = "blue") +
  labs(x = "Dose", y = "Predicted mortality probability")
```



The plot is intentionally made to show what happens at very low and very high dose. At very low dose, the predicted mortality probability approaches 0, and at very high dose it approaches 1. This is a key feature of logistic regression: the predicted probabilities are always between 0 and 1. We also see that the standard errors are larger at intermediate doses, where the slope of the curve is steepest. And the standard errors are smaller at very low and very high doses, where the curve flattens out.

Reporting

When reporting results from a logistic regression, you might say something like:

"A logistic regression was used to model the probability of beetle mortality as a function of insecticide dose. The odds of mortality increased by a factor of 1.28 (95% CI: 1.23, 1.34 per unit increase in dose. This indicates that higher doses are associated with higher odds of mortality. The model predicts that at a dose of 50, the probability of mortality is approximately 0.09 (95% CI: -2.34, -2.26 transformed to probability scale). The model explained a significant amount of deviance (Deviance = 8.4, df = 6, χ^2 p < 0.001)."

Overdispersion in aggregated binomial data

Overdispersion means the variability is larger than the binomial model expects. A quick (rough) check for aggregated binomial data is:

Residual deviance $\approx$ df

Values much larger than 1 for the ratio $\frac{\text{Residual deviance}}{\text{df}}$ suggest overdispersion.

In practice we look for values above about 1.5 or 2.

In the beetle mortality example:

```r
deviance(beetle_glm)
```

```
[1] 8.437898
```

```r
df.residual(beetle_glm)
```

```
[1] 6
```

```r
deviance(beetle_glm) / df.residual(beetle_glm)
```

```
[1] 1.406316
```

Caution

Type I error inflation (binomial overdispersion)
If there is overdispersion and we ignore it, standard errors are usually too small and p-values can be too small (anti-conservative). That increases false positives (Type I errors).

Quasibinomial as a pragmatic fix

If we had found overdispersion, a simple fix is to use the `quasibinomial` family. This estimates an extra dispersion parameter from the data. For example:

```r
beetle_glm_q <- glm(cbind(Number_killed, Number_survived) ~ Dose,
                    data = beetle, family = quasibinomial)
```

Note

`quasibinomial` estimates an extra dispersion parameter from the data. It is often a good "first fix" when aggregated binomial data look overdispersed.

Individual-level binary data (non-aggregated)

In some datasets we record a single 0/1 observation per individual, rather than aggregated counts for many individuals. The same logistic regression model is used, but:

- simple plots of y against explanatory variables look uninformative (two bands)
- some assumption and dispersion checks need extra care

Example: blood screening (ESR)

Individuals with low ESR (ESR is erythrocyte sedimentation rate and is a measure of general inflammation and infection) are generally considered healthy; ESR > 20 mm/hr indicates possible disease. We will model the probability of high ESR using fibrinogen and globulin concentration.

Read in the plasma dataset:

```
plasma <- read_csv("datasets/plasma_esr_original_HSAUR3.csv")
```

```
Rows: 32 Columns: 4
-- Column specification --------------------------------------------------------
Delimiter: ","
chr (1): ESR
dbl (3): fibrinogen, globulin, y

i Use `spec()` to retrieve the full column specification for this data.
i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

Here are the first few rows of the data:

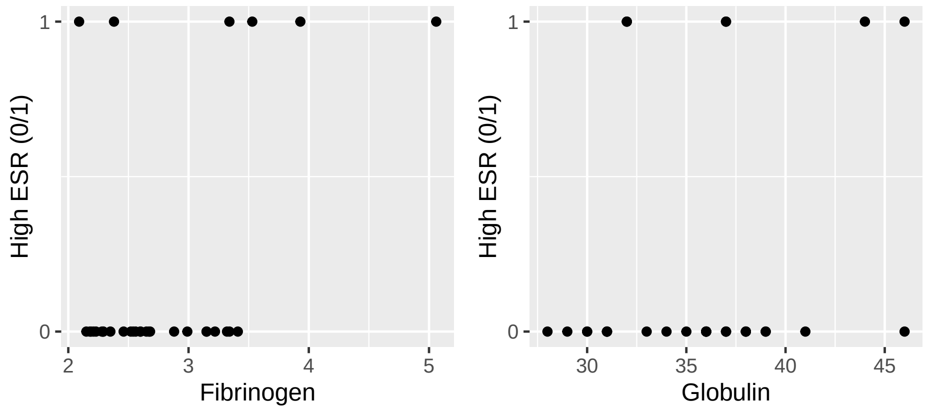
```
head(plasma)
```

```
# A tibble: 6 x 4
  fibrinogen globulin ESR          y
       <dbl>    <dbl> <chr>    <dbl>
1       2.52       38 ESR < 20     0
2       2.56       31 ESR < 20     0
3       2.19       33 ESR < 20     0
4       2.18       31 ESR < 20     0
5       3.41       37 ESR < 20     0
6       2.46       36 ESR < 20     0
```

The ESR response variable is in two variables. `ESR` is a factor with levels "low" and "high". It is also present as a numeric variable `y`, coded 0 (low) and 1 (high). The explanatory variables are `fibrinogen` and `globulin`, both continuous.

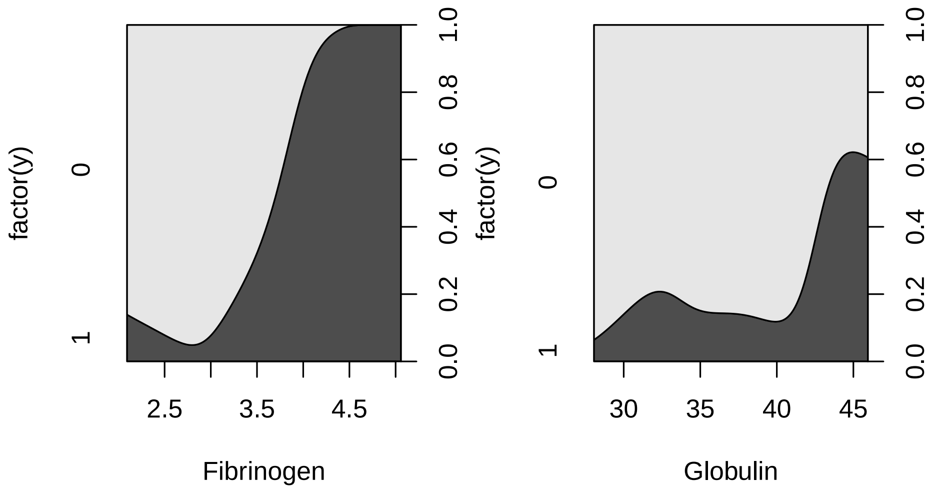
Complication 1: graphical description

Simple scatter plots of 0/1 data are not so informative:



A more informative plot is a conditional density plot. This shows the proportion of 0s and 1s at each value of the explanatory variable.

```r
par(mfrow = c(1, 2))
cdplot(factor(y) ~ fibrinogen, data = plasma, xlab = "Fibrinogen")
cdplot(factor(y) ~ globulin, data = plasma, xlab = "Globulin")
```



```r
par(mfrow = c(1, 1))
```

It looks like higher fibrinogen and higher globulin are associated with higher probability of high ESR. The pattern appears stronger for fibrinogen.

Fit the logistic regression model

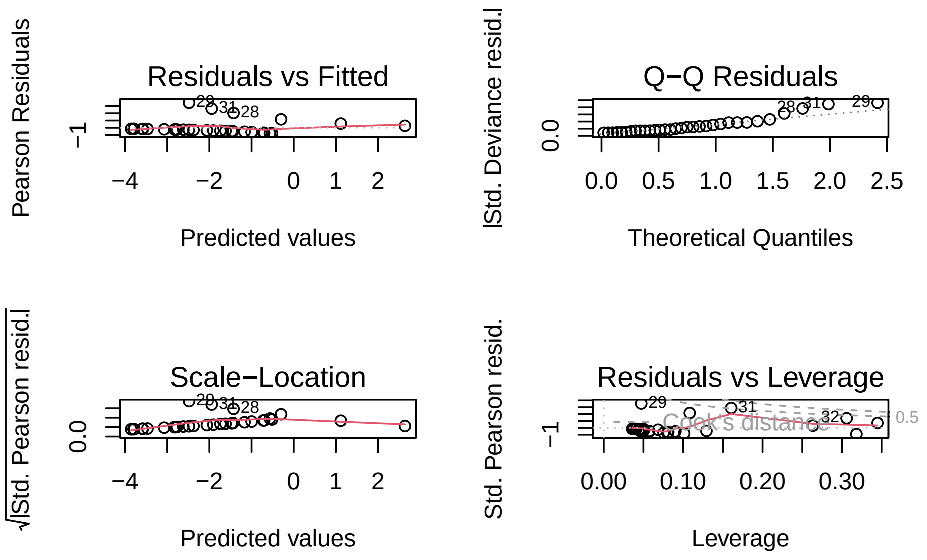
In the case of non-aggregated individual-level 0/1 data, we simply use `y` as the response variable. We have to remember to specify `family = binomial` in `glm()`.

```r
plasma_glm <- glm(y ~ fibrinogen + globulin, data = plasma, family = binomial)
```

Complication 2: model checking and dispersion

Residual plots exist, but can be hard to interpret for 0/1 data:

```r
par(mfrow = c(2, 2))
plot(plasma_glm)
```



❗ Important

Dispersion checks need caution for individual 0/1 data
The simple "residual deviance ≈ residual df" rule-of-thumb is most useful for *aggregated* binomial data. With individual-level 0/1 data, detecting overdispersion is more subtle and often requires additional structure (e.g. grouping, random effects).

Two common practical issues

1. Separation

Sometimes a explanatory variable (or combination of explanatory variables) perfectly predicts the response variable (all 0s on one side, all 1s on the other). This is called **(complete) separation** and can cause extremely large coefficient estimates and warnings.

Caution

Separation warning
If your logistic regression produces enormous standard errors or warnings about fitted probabilities being 0 or 1, check for separation. Remedies include more data, simpler models, or penalized methods (advanced topic).

2. Probability vs odds ratio in reporting

Odds ratios are the default "native" scale of logistic regression, but probabilities are often easier to interpret. In practice, you may report odds ratios **and** translate to probabilities at meaningful explanatory variable values.

Review

There is a lot more to logistic regression and binary data. We can of course make models to analyse more complex data and questions, including multiple explanatory variables, different types of explanatory variable (continuous and categorical), interactions, and so on. But the key ideas are all above. To summarize the main points:

- Data can be either aggregated binomial (successes out of trials) or individual-level 0/1.
- The key questions are about how explanatory variables influence the probability of "yes".
- The distribution family is binomial (Bernoulli for individual-level data).
- The link function is the logit (log-odds).
- If we find overdispersion in aggregated binomial data, `quasibinomial` is a pragmatic fix.
- With non-aggregated individual-level 0/1 data, plotting and checking model assumptions require more care.

Ordination (L10)

Introduction

In earlier chapters, we have analysed variability in **one response variable at a time**. But many biological questions are intrinsically **multivariate**:

- morphology: multiple measurements describe *shape* (not just size)
- communities: abundance of many species describes *composition*
- physiology / omics: many traits or genes change together

When we have **many response variables**, two practical problems appear:

1. How do we **visualize** patterns across many variables?
2. How do we **summarize** the dominant patterns without losing the biology?

Ordination is a family of methods that helps with both.

It can also help in cases where we have many explanatory variables (e.g. environmental gradients). This can be especially useful when predictors are collinear.

! Important

Core idea
Ordination finds a low-dimensional representation (often 2D) of *multivariate* data, so that points that are "similar" in the original multivariate space end up close together in the ordination plot.

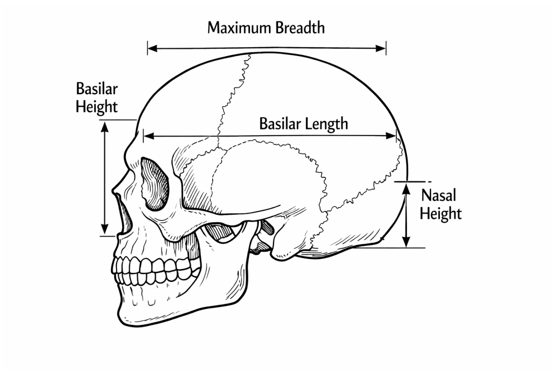
Think–Pair–Share (#tps-ordination-why) Name one dataset you work with (or could imagine working with) that has many response variables. What would you like to learn from it that a single-response analysis might miss?

Working example: skull shape through time

We will use a classic dataset of **human skull dimensions** measured in millimeters. The biological question is:

> Has skull **shape** changed through time?

Four measurements of skull were made:



We will treat the four skull measurements as multivariate responses, and time as an explanatory variable.

First read the dataset from the csv file:

```r
skull <- read_csv(here("datasets", "skull_shape_time.csv"))
```

```
Rows: 150 Columns: 6
-- Column specification --------------------------------------------------------
Delimiter: ","
chr (1): id
dbl (5): max.breadth, basi.height, basi.length, nasal.height, thousand.years

i Use `spec()` to retrieve the full column specification for this data.
i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

Wide vs long format (and why you should care)

Multivariate data often come in **wide format**:

- one row per object (skull)
- one column per variable (measurement)

That's great for many ordination functions.

But long format is often easier for: - grouped summaries - plotting multiple variables with a single ggplot call

We can easily make a long format version with `pivot_longer()`:

```r
skull_long <- skull |>
  pivot_longer(
    cols = c(max.breadth, basi.height, basi.length, nasal.height),
    names_to = "dimension",
    values_to = "value"
  )
```

This will make some of the following exploration easier.

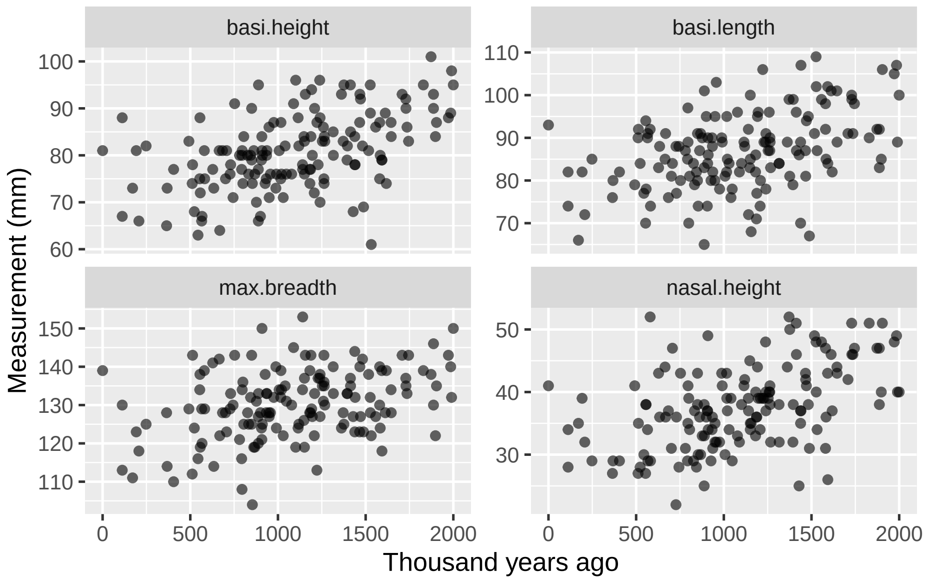
Explore first: univariate views of a multivariate problem

A good habit:

1) visualize each variable,
2) look for patterns and surprises,
3) then use ordination to *summarize*.

Four scatterplots (measurement vs time)

```r
p_vars <- skull_long |>
  ggplot(aes(x = thousand.years, y = value)) +
  geom_point(alpha = 0.6) +
  facet_wrap(~ dimension, scales = "free_y", ncol = 2) +
  labs(x = "Thousand years ago", y = "Measurement (mm)")
p_vars
```



Interpretation (qualitative):
If all dimensions change similarly through time, that's mostly a **size** change. If some change differently, that suggests **shape** change.

Think–Pair–Share (#tps-size-shape) How would you distinguish "size change" from "shape change" using these plots?

Why not four separate regressions?

Fitting separate regressions for each measurement has two problems: 1. **Multiple testing**: testing four times increases the chance of false positives. 2. **Ignoring covariance**: measurements may be correlated, so analyzing them separately misses joint patterns

Think–Pair–Share (#a_why_ordination) What information is lost when you analyze multivariate data one variable at a time? Why could this be important?

Ordination I: PCA (Principal Components Analysis)

PCA is the most common entry point.

! Important

What PCA does
PCA finds new axes (PC1, PC2, …) that are: - linear combinations of the original variables - orthogonal (uncorrelated) - ordered so that PC1 captures the most variance, PC2 the next most, etc.

Intuitive picture of PCA

Imagine that each skull is a point in a space with one axis per measurement (four dimensions here). Together, the skulls form a cloud of points.

PCA asks from which direction does this cloud show the greatest spread?

To answer this, PCA rotates the coordinate system so that:

- PC1 points in the direction of greatest variation,
- PC2 points in the next greatest direction, orthogonal to PC1,
- and so on.

You can think of PCA as turning the data cloud until you find the view where the points are most spread out. That view becomes PC1. Then PCA finds the best second view at right angles to it (PC2).

The key idea is that PCA does not invent new information: it simply re-expresses the same data using new axes that make dominant patterns easier to see.

Note

Under the hood, PCA is based on the covariance (or correlation) matrix of the variables. The principal components are eigenvectors of this matrix, and the amount of variance they capture is given by the corresponding eigenvalues. You do not need to compute these by hand, but this explains why PCA is fundamentally about variance and correlation.

Centering and scaling

Before PCA, we usually **center** and **scale** the data, or at least think carefully if we should.

- **Centering** subtracts the mean of each variable (so mean = 0).
- **Scaling** divides by the SD of each variable (so SD = 1).

Scaling matters if variables are on different scales or if you care about *relative* variation.

Note

Note that above it is written that PCA does not change the relative positions of the points. However, if we scale the data before performing PCA, this can affect the relative positions of the points in the PCA space. Scaling ensures that all variables contribute equally to the analysis, which can be important when variables are measured on different scales. If we do not scale the data, variables with larger variances can dominate the PCA results, potentially distorting the relative positions of the points in the PCA space.

We can make these transformed variables ourself, or use the built-in options in `prcomp()`.

So, let us do our first PCA. We must be sure to only do the PCA on the four skull measurement variables. To do this we will select only those columns from the original data frame, and pipe these into the `prcomp()` function:

```
skull_pca <- skull |>
  dplyr::select(max.breadth, basi.height, basi.length, nasal.height) |>
  prcomp(center = TRUE, scale. = TRUE)
```

What is this new dataset? Part of it is the new coordinates of each skull in the PC space:

```
head(skull_pca$x)
```

```
           PC1         PC2        PC3        PC4
[1,] -3.621758  0.87923353  0.2413718 -0.2421541
```

```
[2,] -3.201426 -0.38205353  0.3833647 -0.3500371
[3,] -3.004179 -1.00242513  1.5640064 -0.2348435
[4,] -2.683398  0.08615036  1.6069216 -1.0534579
[5,] -3.001132 -0.46314398 -0.2188543 -0.7276811
[6,] -2.774762  0.09654899  0.2109694 -0.8941411
```

Variance represented

Each of the new PC axes captures some of the variance in the original data. We can summarize this with:

```
summary(skull_pca)
```

```
Importance of components:
                          PC1    PC2    PC3     PC4
Standard deviation     1.4916 0.8981 0.7643 0.62007
Proportion of Variance 0.5562 0.2016 0.1460 0.09612
Cumulative Proportion  0.5562 0.7578 0.9039 1.00000
```

This shows the standard deviation of each PC axis, the proportion of variance explained by each PC, and the cumulative proportion of variance explained.

The first PC axis captures 55.6% of the variance in the original data. The first two PC axes together capture 75.8% of the variance.

This is quite a lot, so we can visualize the data well with two PC axes. When we do so, we ignore the remaining axes, which capture less variance (24.2%).

PCA scores plot (PC1 vs PC2)

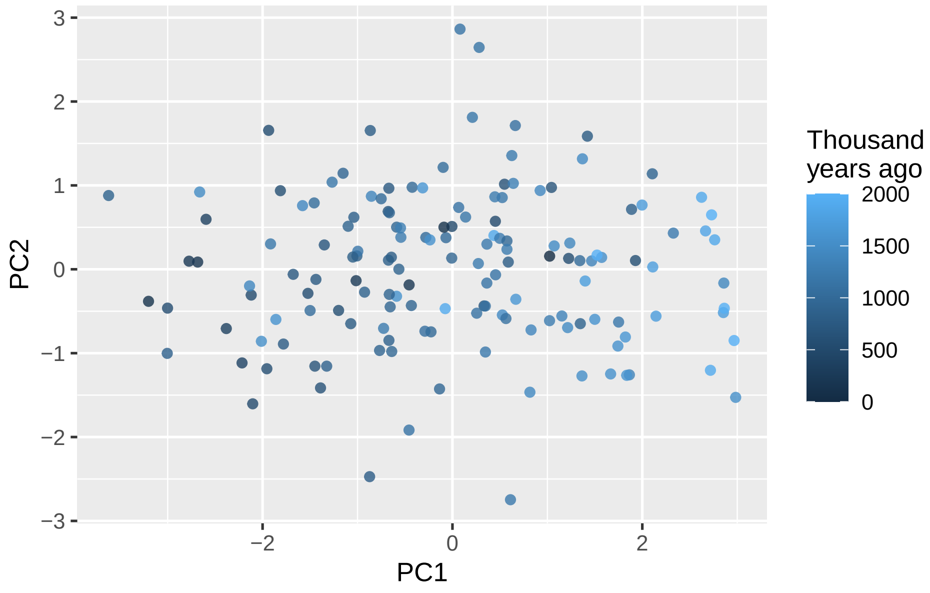
The **scores** are the coordinates of each skull in the new PC space. We do a bit of data wrangling to combine these with the original data (id and time):

```
scores <- as_tibble(skull_pca$x) |>
  mutate(id = skull$id, thousand.years = skull$thousand.years)
scores |> select(id, PC1, PC2, thousand.years) |> slice_head(n = 5)
```

```
# A tibble: 5 x 4
  id      PC1     PC2 thousand.years
  <chr> <dbl>   <dbl>          <dbl>
1 S1    -3.62  0.879             888
2 S2    -3.20 -0.382             112
3 S3    -3.00 -1.00              854
4 S4    -2.68  0.0862            171
5 S5    -3.00 -0.463             544
```

And then make the graph:

```r
ggplot(scores, aes(x = PC1, y = PC2, color = thousand.years)) +
  geom_point(alpha = 0.8) +
  labs(color = "Thousand\nyears ago")
```



ℹ Note

Interpretation tip
If points separate along PC1 as time increases, that suggests the dominant multivariate trend is associated with time.

What do the axes mean?

The **loadings** tell us how each original variable contributes to each PC. In fact, each PC is a linear combination of the original variables, weighted by the loadings.

```r
loadings <- as_tibble(skull_pca$rotation, rownames = "variable")
loadings
```

```
# A tibble: 4 x 5
  variable       PC1     PC2    PC3    PC4
  <chr>        <dbl>   <dbl>  <dbl>  <dbl>
1 max.breadth  0.436  0.689  -0.568  0.107
2 basi.height  0.536  0.108   0.643  0.537
3 basi.length  0.440 -0.712  -0.474  0.272
4 nasal.height 0.573 -0.0783  0.196 -0.792
```

These can help us interpret the PCs. For example, we see a strong positive loading of all four skull measurements on PC1. This suggests that PC1 represents overall size, as all measurements increase together. On the second PC axis, we see a mix of positive and negative loadings, indicating that PC2 captures shape differences where some measurements increase while others decrease. The positive PC2 loadings for head breadth and nasal height, combined with negative loadings for basi height and basi length, suggest that PC2 reflects a shape change where skulls become wider and taller in the nasal region while becoming shorter in the base dimensions.

Another way to interpret PCs is to look at the correlations between the original variables and the PCs. This can provide insights into how each original variable relates to the new PC axes. We can calculate these correlations as follows:

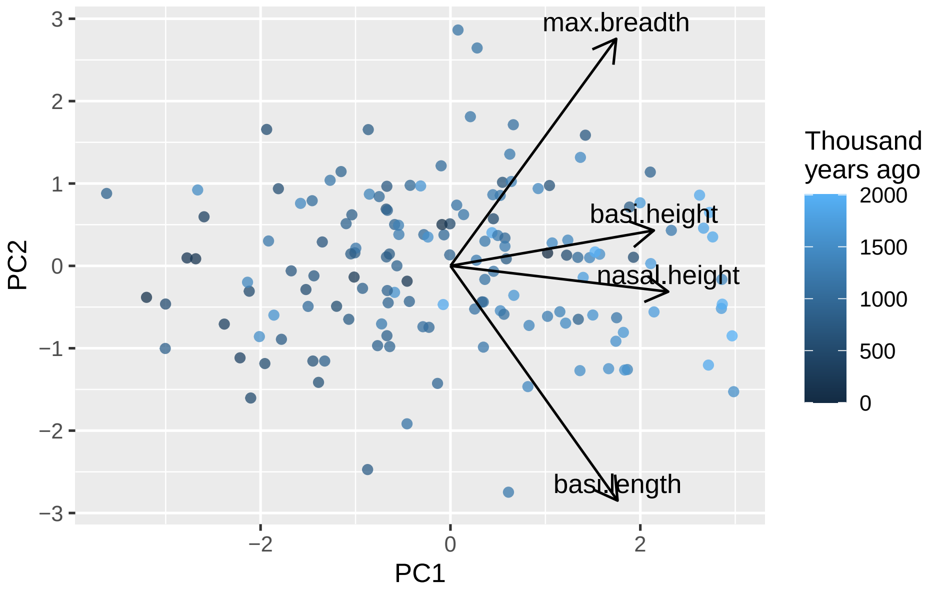
```
X <- skull |>
  dplyr::select(max.breadth, basi.height, basi.length, nasal.height)
round(cor(X, skull_pca$x), 2)
```

```
              PC1   PC2   PC3   PC4
max.breadth  0.65  0.62 -0.43  0.07
basi.height  0.80  0.10  0.49  0.33
basi.length  0.66 -0.64 -0.36  0.17
nasal.height 0.86 -0.07  0.15 -0.49
```

Think–Pair–Share (#tps-pc-meaning) If PC1 is positively correlated with all four skull measurements, what biological interpretation is most natural?

A simple biplot (scores + loadings)

Below is a lightweight biplot-style plot. (There are fancy versions in packages like **factoextra**, but we keep it minimal.)



In this graph we see all arrow point in the same horizontal direction, indicating that PC1 represents overall size. An increase in any of the four measurements will increase PC1.

In contrast, the arrows for PC2 point in different directions, indicating that PC2 represents shape differences where some measurements increase while others decrease. The two longest arrows are for basi.length and max.breadth, suggesting that these measurements contribute most strongly to shape variation captured by PC2.

Think–Pair–Share (#tps-biplot-interpretation) Make a sketch of how skull size / shape changes along the two PC axis.

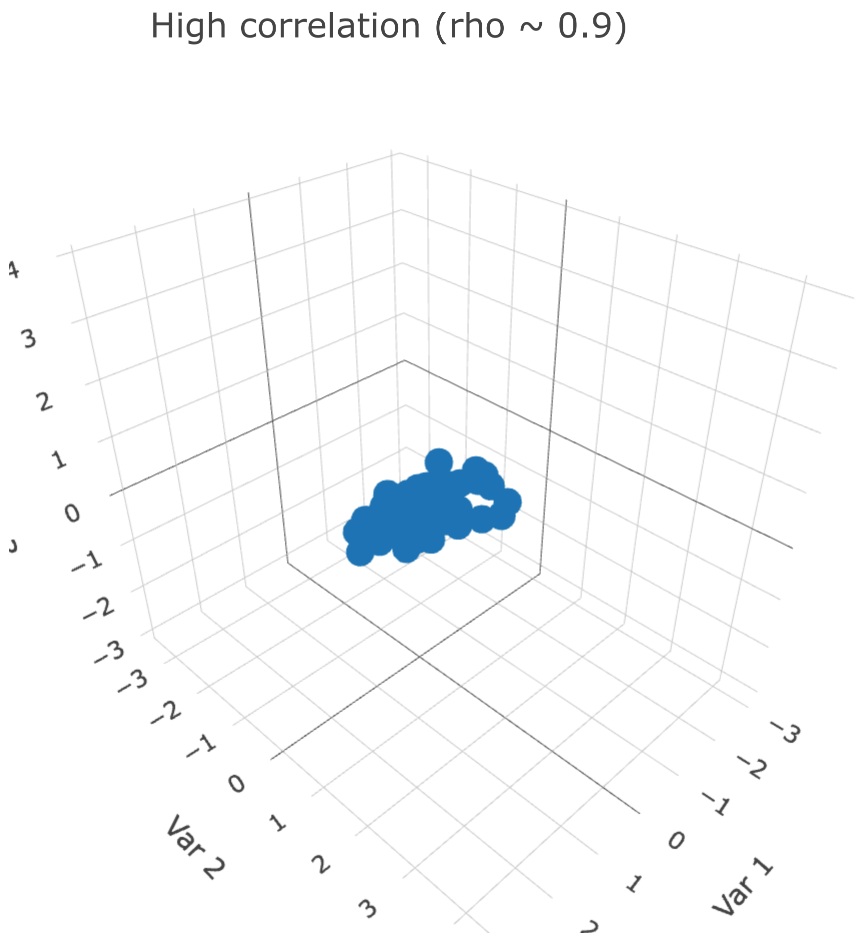
Think–Pair–Share (#a_size_vs_shape) What pattern would indicate pure size change? What pattern would indicate shape change?

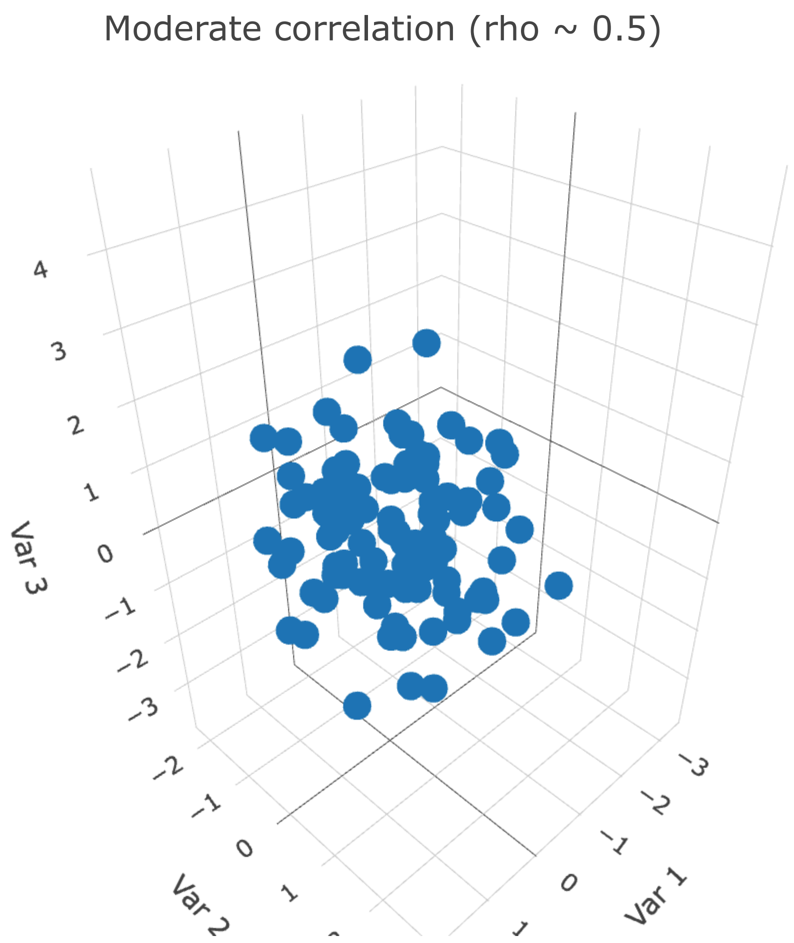
When ordination helps

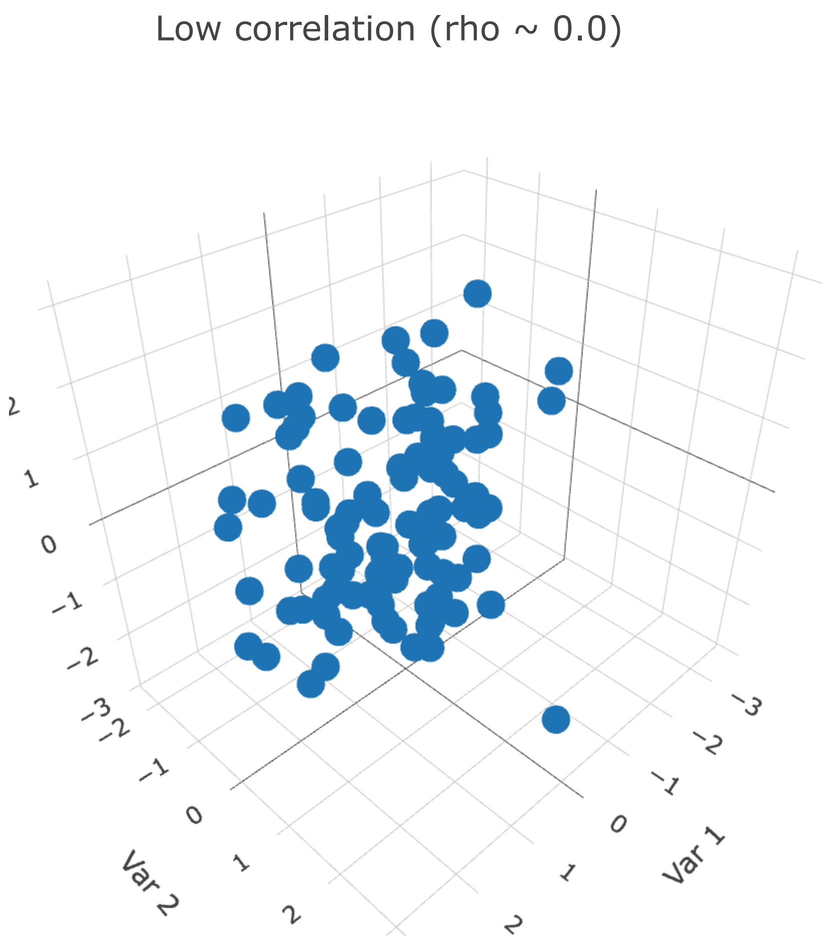
Think–Pair–Share (#tps-when-ordination) In what situations would ordination methods like PCA be especially helpful? When would they be less useful? Think about the nature of the data, and, if you like, the biological questions.

We might think that ordination is more useful when we have a greater number of variables, because then the reducing down to two or three dimensions is more helpful. But the key factor is actually the **correlation structure** among the variables. If many variables are correlated, then ordination can capture most of the variance in a few dimensions. If variables are uncorrelated, then ordination may not help much since most of the variance is spread evenly across many dimensions. Let's have a look a this with some simulated data. We will simulate three datasets with three variables each, but with different correlation structures: high correlation, moderate correlation, and low correlation.

Now a 3D plot of the simulated data with different correlation structures (please note that these plots are best viewed in the HTML version of the book; in the PDF version, they will appear as static images):




Moderate correlation (rho ~ 0.5)
Var 3
4
3
2
1
0
−1
−2
−3
Var 2
−2
−1
0
1
Var 1
−3
−2
−1
0



The first plot shows data with high correlation among variables, where the points are clustered along a diagonal line, indicating that the variables are strongly related. The second plot shows moderate correlation, where the points are more spread out but still show some clustering. The third plot shows low correlation, where the points are scattered randomly in space, indicating that the variables are largely independent.

Think–Pair–Share (#tps-ordination-correlation) What output of the PCA would you expect to differ among these three datasets?

We expect that in the high correlation case, the first principal component will capture a large proportion of the variance, while in the low correlation case, the variance will be more evenly distributed across all principal components. Let's check this by performing PCA on each dataset and examining the variance explained by each principal component.

```
[1] 0.925377783 0.065681354 0.008940863
```

```
[1] 0.6571611 0.1978438 0.1449951
```

```
[1] 0.3874361 0.3586825 0.2538815
```

It turns out that in the high correlation case, the first principal component captures a large proportion of the variance (92.5%). This is a large amount of variance explained by a single component. It is caused because we made a dataset where the variables are strongly correlated. There is only one axis along which the data vary strongly.

In contrast, in the zero correlation case, the variance is more evenly distributed across all principal components, with each component capturing around 38.7%, 35.9%, and 25.4% of the variance respectively. This indicates that there is no single dominant direction of variance in the data, as the variables are largely independent.

In the intermediate correlation case, the variance explained by each principal component is more balanced, with the first component capturing around 65.7% of the variance, and the remaining components capturing significant portions as well. This reflects the moderate correlation structure in the data.

Ordination is most useful when it can reduce the dimensionality of the data while retaining most of the variance. This means that ordination methods like PCA will be more effective in summarizing the data in the high correlation case, as most of the information can be captured in just one or two dimensions. In contrast, in the low correlation case, ordination may not provide much dimensionality reduction, as each variable contributes independently to the overall variance.

What determines if we have variables with high correlation in real datasets? Often, it is biological or physical relationships among the variables. For example, in morphological datasets, measurements of different body parts may be correlated due to overall size or shape factors. In ecological datasets, species abundances may be correlated due to shared environmental preferences or interactions. Understanding the underlying biology can help us anticipate when ordination methods will be most useful.

Ordination II: NMDS (Non-metric Multidimensional Scaling)

PCA is powerful, but it is a **linear** method and it relies on Euclidean geometry in the original variable space.

NMDS is often used when:

- you want ordination based on **distances/dissimilarities**
- the relationships are not well represented by a linear method
- you want flexibility in the choice of distance (e.g. Bray–Curtis, Gower, …)

! Important

What NMDS does (conceptually)
1) compute pairwise distances among objects
2) place points in a low-dimensional space (usually 2D)
3) try to preserve the **rank order** of distances (non-metric)
4) report **stress**: lower is better (roughly: mismatch between original dissimilarities and ones in the lower-dimensional space)

Step 1: Choose a distance measure

Here our variables are numeric, so Euclidean distance is a reasonable default.

We will scale first (so variables contribute comparably), then compute distances:

```
X_scaled <- scale(X)
D <- dist(X_scaled, method = "euclidean")
#D
```

Step 2: Fit NMDS with multiple random starts

In NMDS, we need to choose some random starting configuration of points. And we want to make sure that we choose a good solution, not just a local optimum. Hence, we try multiple random starts and pick the best one.

In practice, use `metaMDS()` (from **vegan**) rather than calling the low-level optimizer directly. It tries multiple starting configurations and does useful housekeeping.

When calling `metaMDS()`, we specify the following:

- `k = 2` for a 2D solution.
- `trymax = 50` to try up to 50 random starts.
- `autotransform = FALSE` because we already scaled the data ourselves.
- `trace = FALSE` to suppress output during fitting.

```
set.seed(1)
nmds <- metaMDS(D, k = 2, trymax = 50, autotransform = FALSE, trace = FALSE)
nmds
```

```

Call:
metaMDS(comm = D, k = 2, trymax = 50, autotransform = FALSE,      trace = FALSE)

global Multidimensional Scaling using monoMDS

Data:     D
Distance: euclidean
```

```
Dimensions: 2
Stress:     0.1467223
Stress type 1, weak ties
Best solution was repeated 1 time in 20 tries
The best solution was from try 16 (random start)
Scaling: centring, PC rotation
Species: scores missing
```

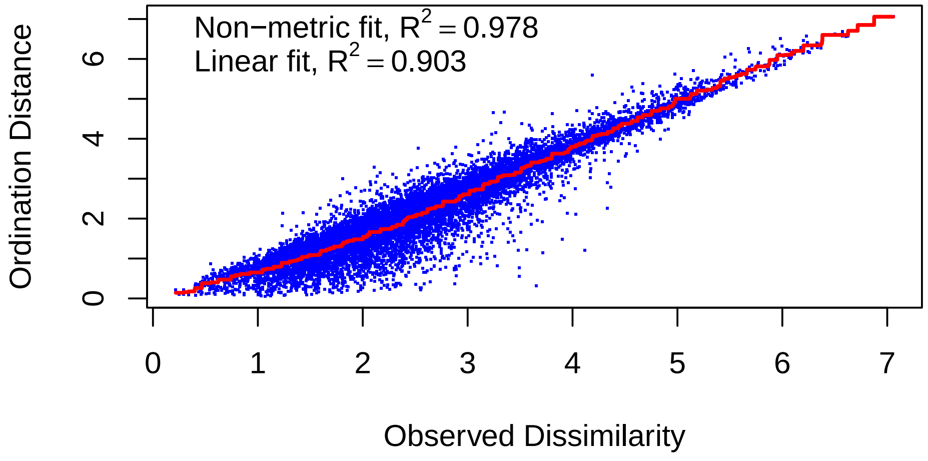
Lots of information here, for example we have:

- the call used
- the type of distance
- the number of dimensions
- the number of random starts tried
- the final stress value
- the scaling of the data

Step 3: Assess NMDS fit with stress

The stress of an NMDS solution quantifies how well the low-dimensional configuration preserves the rank order of the original dissimilarities. That is, it is a measure of mismatch between the distances in the original high-dimensional space and the distances in the reduced low-dimensional space. Here is the stress plot for our NMDS solution:

```
stressplot(nmds)
```



We see a fairly good match between the original distances and the NMDS distances. The correlations are reasonably high, indicating that the NMDS solution captures the rank order of dissimilarities well.

The stress value is another measure of fit. Lower stress values indicate a better fit. A common rule of thumb is that stress < 0.1 is a good fit, stress between 0.1 and 0.2 is acceptable, and stress > 0.2 indicates a poor fit. A poor fit suggests that the data may not be well represented in two dimensions, and a higher-dimensional solution may be needed.

```
nmds$stress
```

```
[1] 0.1467223
```

We are in the range of okay fit in two dimensions.

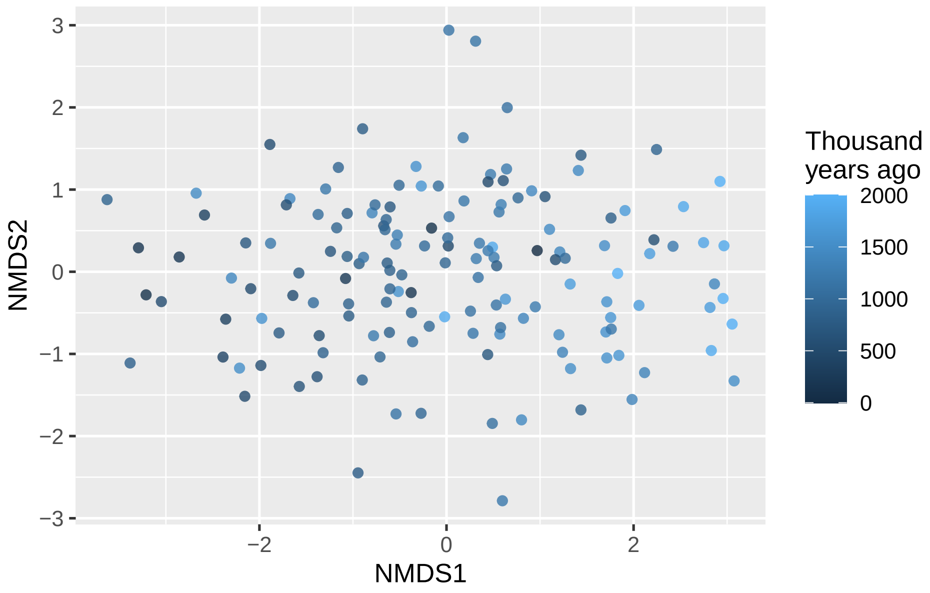
What improvement in stress do we get if we go to three dimensions?

```
set.seed(1)
nmds_3d <- metaMDS(D, k = 3, trymax = 50, autotransform = FALSE, trace = FALSE)
nmds_3d$stress
```

```
[1] 0.07009879
```

The stress decreases when we move to three dimensions, indicating a better fit. However, the improvement may not be substantial enough to justify the added complexity of a three-dimensional solution. In practice, we often prefer two-dimensional solutions for ease of visualization and interpretation, unless the stress reduction is very large.

We can plot the NMDS scores, colored by time:



Note

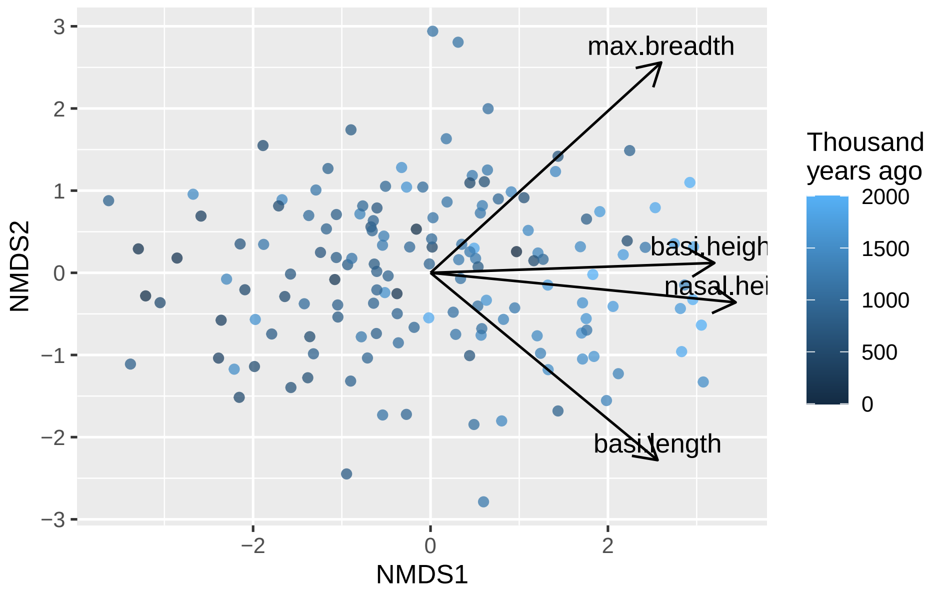
Axis directions are arbitrary
Flipping or rotating an NMDS solution (or a PCA) does not change its meaning. Only relative distances among points matter.

If we wanted to know the biological interpretation of the NMDS axes, we would need to look at correlations between the original variables and the NMDS axes. This is just the same as we did for PCA. We could then plot the correlations as arrows on the NMDS plot to help interpret the axes.

```
cor_nmds <- round(cor(X, nmds_scores |> select(NMDS1, NMDS2)), 2)
cor_nmds
```

```
             NMDS1 NMDS2
max.breadth   0.65  0.64
basi.height   0.80  0.03
basi.length   0.64 -0.57
nasal.height  0.86 -0.09
```

And plot these as arrows:



Note

This NMDS graph looks a lot like the PCA biplot. This is because our data are fairly linear and Euclidean distances are appropriate. In more complex datasets, NMDS can reveal patterns that PCA might miss.

Think–Pair–Share (#a__linear__vs__nonlinear__ordination) When might a linear method like PCA be misleading? What does NMDS relax?

Hypothesis testing: avoid "four separate regressions"

If you test time against each skull measurement separately, you face a **multiple testing** problem.

A better match to the question ("do skulls change through time in multivariate space?") is a **multivariate test**.

Here are four common approaches demonstrated but not deeply explained:

1. MANOVA (parametric, multivariate normality assumptions)

2. PERMANOVA (distance-based, permutation test)

3. Dispersion checks (are groups equally variable?)
4. Fitting time onto an ordination (envfit, ordisurf)

1) MANOVA (parametric)

Multivariate ANOVA (MANOVA) tests whether group **centroids** differ in multivariate space, assuming multivariate normality. The response is a matrix of multiple variables, and the explanatory variables can be categorical or continuous.

```
man_mod <- manova(cbind(max.breadth, basi.height, basi.length, nasal.height) ~ thousand
                  data = skull)
summary(man_mod, test = "Pillai")
```

```
                Df  Pillai approx F num Df den Df    Pr(>F)    
thousand.years   1 0.36751   21.064      4    145 1.042e-13 ***
Residuals      148                                             
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

If we bin years into categories, MANOVA becomes closer to multivariate ANOVA:

```
skull <- skull |>
  mutate(year_class = cut(thousand.years, breaks = 8) |> fct_inorder())
man_mod_cat <- manova(cbind(max.breadth, basi.height, basi.length, nasal.height) ~ thou
                      data = skull)
summary(man_mod_cat, test = "Pillai")
```

```
                Df  Pillai approx F num Df den Df    Pr(>F)
```

```
thousand.years   1 0.36751   21.064      4    145 1.042e-13 ***
Residuals      148
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Caution

MANOVA assumptions matter
MANOVA relies on multivariate normality and homogeneity of covariance matrices. In many biological datasets, these are imperfect—so distance-based alternatives are common.

2) PERMANOVA (distance-based)

PERMANOVA tests whether group **centroids** differ in multivariate space, using permutations. In vegan, use `adonis2()`.

```r
# Use the same distance matrix D we used for NMDS
adonis2(D ~ year_class, data = skull, permutations = 999)
```

```
Permutation test for adonis under reduced model
Permutation: free
Number of permutations: 999

adonis2(formula = D ~ year_class, data = skull, permutations = 999)
          Df SumOfSqs      R2      F Pr(>F)
Model      7   138.47 0.23234 6.1396  0.001 ***
Residual 142   457.53 0.76766
Total    149   596.00 1.00000
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

3) PERMDISP / betadisper: are groups equally variable?

A common pitfall:

- PERMANOVA can be significant either because **centroids differ**
- or because **dispersion differs** (groups have different spread)

So it's good practice to check dispersion when comparing groups.

```r
bd <- betadisper(D, skull$year_class)
#bd
anova(bd)
```

```
Analysis of Variance Table

Response: Distances
```

```
           Df Sum Sq Mean Sq F value Pr(>F)
Groups      7  0.929 0.13274  0.2284  0.978
Residuals 142 82.529 0.58119
```

```
permutest(bd, permutations = 999)
```

```
Permutation test for homogeneity of multivariate dispersions
Permutation: free
Number of permutations: 999

Response: Distances
           Df Sum Sq Mean Sq      F N.Perm Pr(>F)
Groups      7  0.929 0.13274 0.2284    999   0.98
Residuals 142 82.529 0.58119
```

4) Fitting time onto an ordination: envfit and ordisurf

Even when you use an unconstrained ordination (PCA/NMDS), you may want to show how an explanatory variable aligns with it.

envfit: linear fit + permutation test

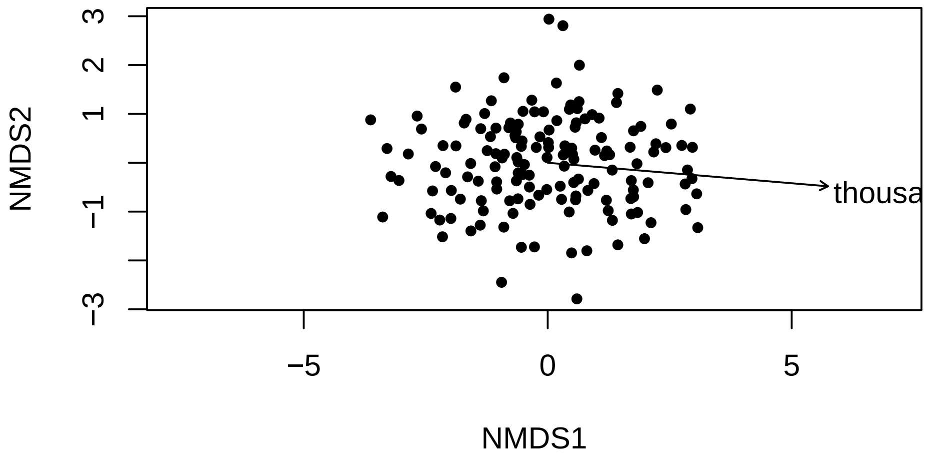
```
fit_years <- envfit(nmds ~ thousand.years, data = skull, permutations = 999)
fit_years
```

```
***VECTORS

                   NMDS1     NMDS2     r2 Pr(>r)
thousand.years  0.996500 -0.083553 0.3643  0.001 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
Permutation: free
Number of permutations: 999
```

```
plot(nmds, type = "n")
```

```
species scores not available
```

```
points(nmds, display = "sites", pch = 16, cex = 0.8)
plot(fit_years, col = "black")  # vector direction and significance
```

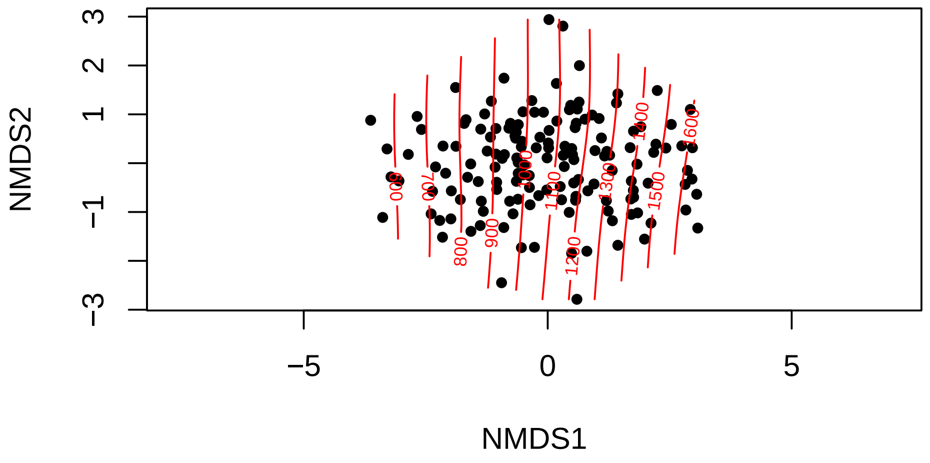


ordisurf: non-linear surface (GAM)

```r
plot(nmds, type = "n")
```

```
species scores not available
```

```r
points(nmds, display = "sites", pch = 16, cex = 0.8)
ordisurf(nmds ~ thousand.years, data = skull, add = TRUE)
```



```
Family: gaussian
Link function: identity

Formula:
y ~ s(x1, x2, k = 10, bs = "tp", fx = FALSE)

Estimated degrees of freedom:
```

```
2.98  total = 3.98

REML score: 1092.962
```

PCA in multiple regression

PCA can also be used to reduce multiple explanatory variable into a few uncorrelated components, which can then be used in multiple regression models. This is particularly useful when the predictors are highly correlated, as it helps to avoid multicollinearity issues.

The steps for this would be:

1. Perform PCA on the explanatory variables to obtain principal components.
2. Make a linear regression model using the selected principal components as explanatory variables.
3. Interpret the results in terms of the original variables, if needed.

The "win" here is that the principal components are uncorrelated, which simplifies the regression analysis and interpretation. The "lose" is that the principal components may not have a straightforward biological interpretation, so care is needed when explaining the results.

Review

Ordination is useful when we have multivariate data and when variables within that data are correlated. The correlation structure means that we can represent the data in a lower-dimensional space without losing too much information. If there is a lot of correlation among variables, then ordination becomes very useful. If there is little correlation, then ordination may not help much since most of the variance is spread evenly across many dimensions.

Summary:

- PCA is a linear ordination method that finds orthogonal axes capturing the most variance.
- NMDS is a flexible, distance-based ordination method that preserves rank order of dissimilarities.
- Both methods help visualize and summarize multivariate data.
- Multivariate hypothesis tests (MANOVA, PERMANOVA) assess group differences in multivariate space.
- Checking dispersion (PERMDISP) is important to interpret PERMANOVA results.
- Fitting explanatory variables onto ordinations (envfit, ordisurf) helps interpret patterns.

Mixed models (L11-1)

Introduction

So far in BIO144 we mostly used models where **each observation is assumed independent**:

- linear models (`lm()`)
- generalized linear models (`glm()`), e.g. Poisson and binomial.

By independent we mean that the value of one observation does not give us any information about the value of another observation.

But many biological datasets violate independence because observations come in **groups**:

- repeated measures on the same individual (before/after, time series, multiple tissues)
- multiple individuals from the same plot / site / stream / lake / cage / family
- students within classes, patients within hospitals, samples within batches

In these cases, treating all rows as independent often leads to **false confidence** (too-small standard errors, too-small p-values). Mixed models are one standard way to handle this.

! Important

Core idea A *mixed model* extends regression by adding **random effects** that represent grouping structure (clusters) in the data.

- **fixed effects**: effects you want to estimate explicitly (treatments, temperature, time, …)
- **random effects**: variation among groups (individuals, sites, years, …) that induces correlation within groups

Think–Pair–Share (#tps-why-mixed) Name one example in biology where multiple measurements come from the same "unit" (individual, plot, lake, strain, batch…). What goes wrong if we pretend those measurements are independent?

Why not just average?

A common workaround is to average repeated measurements to a single value per group and then use `lm()`.

Sometimes this is OK — but often it throws away information.

Reasons *not* to average include:

1. **Imbalanced sampling** (groups have different numbers of observations): averaging changes the weighting.
2. **Which average?** (mean, median, mode): different choices answer different questions.
3. **False confidence**: pretending "n rows" are independent can make uncertainty look too small.
4. **You may want to study variation among groups** (some individuals respond more strongly than others).
5. **"Sharing information" across groups**: mixed models partially pool group estimates toward the overall mean.
6. **Keeping information**: you can use all observations without collapsing the design.

Note

A mixed model gives you a principled compromise between:
- analysing each observation as a independent one, and
- averaging everything (too coarse).

This is sometimes called **partial pooling**.

The problem: non-independence and pseudoreplication

Imagine we measure the same individual multiple times.

If we fit a simple linear model, the residuals are assumed independent:

$$\varepsilon_i \sim \text{Normal}(0, \sigma^2), \quad \text{independent across } i$$

But repeated measures induce correlation:

- measurements from the same individual tend to be more similar
- so residuals are not independent

This is one common form of **pseudoreplication**: treating repeated measurements as if they were separate independent replicates.

Random intercept models

Random intercept idea

Suppose we measure a response y (e.g. growth) for individuals j at observations i. And that we measure each individual more than once. And that we have some treatment, such as temperature x.

This means that our data have a **grouping structure**: observations are grouped by individual. If we want to calculate the mean growth, we should account for this grouping.

Let's make an example dataset to illustrate:

Read in the data:

```r
dat_example <- read_csv("datasets/mixed_model_example.csv")
```

```
Rows: 50 Columns: 3
-- Column specification --------------------------------------------------------
Delimiter: ","
chr (1): individual
dbl (2): temperature, growth

i Use `spec()` to retrieve the full column specification for this data.
i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

Here are the first few rows:

```r
head(dat_example)
```

```
# A tibble: 6 x 3
  individual temperature growth
  <chr>            <dbl>  <dbl>
1 Ind01             9.15  16.1
2 Ind01             9.37  17.4
3 Ind01             2.86   7.58
4 Ind01             8.30  14.2
5 Ind01             6.42  13.8
6 Ind02             5.19  12.5
```

In this data we have two sources of variation in `growth`:

1. **fixed effect** of `temperature` (same for all individuals)
2. **random effect** of `individual` (different baseline growth for each individual)

A fixed effect is something we want to estimate explicitly (e.g. how growth changes with temperature). Fixed effects are variables in which the values have specific meaning (e.g. temperature = 5°C).

A random effect is something that varies among groups (individuals here), but where we are not interested in the specific values for each group. Instead, we want to estimate the amount of variation among groups. Random effects are variables where the specific values are not of interest, but rather the variation among them (e.g. individual identity). I.e., the value "Ind01" has no specific meaning; we just want to know how much individuals differ from each other on average. That value could be anything.

A random effect is often containing levels that are considered a random sample from a larger population. We did not, for example, choose specific individuals for a reason; they are just a sample of all possible individuals.

An appropriate model for this data is a **random-intercept mixed model**. It is a model in which each group (individual) has its own intercept (baseline), but the intercepts are assumed to come from a common distribution.

A random-intercept mixed model is:

$$y_{ij} = \beta_0 + \beta_1 x_{ij} + b_{0j} + \varepsilon_{ij}$$

where:

- β_0, β_1 are **fixed effects**
- x_{ij} is the explanatory variable for observation i in group j. Also called a **fixed effect**.
- b_{0j} is a **random intercept** for group j, typically $b_{0j} \sim \text{Normal}(0, \sigma_b^2)$
- ε_{ij} is residual error, $\varepsilon_{ij} \sim \text{Normal}(0, \sigma^2)$

Interpretation:

- each group gets its own intercept (baseline), but
- those intercepts are assumed to come from a common distribution

Random intercept syntax in R

To make a mixed model in R we can no longer use `lm()` or `glm()`. Instead, we use the `lmer()` function from the `lme4` package.

In the `lmer` function we have to specify the random effects in a special way. For example, a random-intercept model for `y` with fixed effect `x` and random intercept by `group` is specified as:

```
y ~ x + (1 | group)
```

Read it as: "a model for `y` with fixed effect `x` and a random intercept by `group`".

Random slope models

Sometimes groups differ not only in baseline level, but also in how they respond to an explanatory variable.

A random-slope model:

$$y_{ij} = \beta_0 + \beta_1 x_{ij} + b_{0j} + b_{1j} x_{ij} + \varepsilon_{ij}$$

In R:

```r
y ~ x + (1 + x | group)
```

Here each group has its own intercept **and** its own slope, and these can be correlated.

Caution

Random slopes are powerful but can be hard to estimate with small datasets. If the model struggles to fit (singular fit warnings), consider simplifying.

Nested and crossed random effects

Nested

Nested means one grouping factor is contained within another.

Example: measurements within plants within plots:

```r
y ~ treatment + (1 | plot/plant)
```

This expands to `(1 | plot) + (1 | plot:plant)`.

Crossed

Crossed means groups are not nested.

Example: repeated measures with multiple observers (each observer measures many individuals; each individual is measured by many observers):

```r
y ~ x + (1 | individual) + (1 | observer)
```

Hands-on example: plant growth with repeated measures

Biological story

You are studying how temperature affects plant growth.

- 30 plants are grown at one of three temperatures: 10°C, 15°C, 20°C.
- Each plant is measured weekly for 8 weeks.
- Response: plant height (cm).

Because we repeatedly measure the **same plant**, the observations are not independent. We will compare:

1. a naive linear model (wrong independence assumption),
2. a mixed model with plant as a random effect.

An example dataset

Read in the data:

```r
dat_example <- read_csv("datasets/plant_growth_repeated.csv")
```

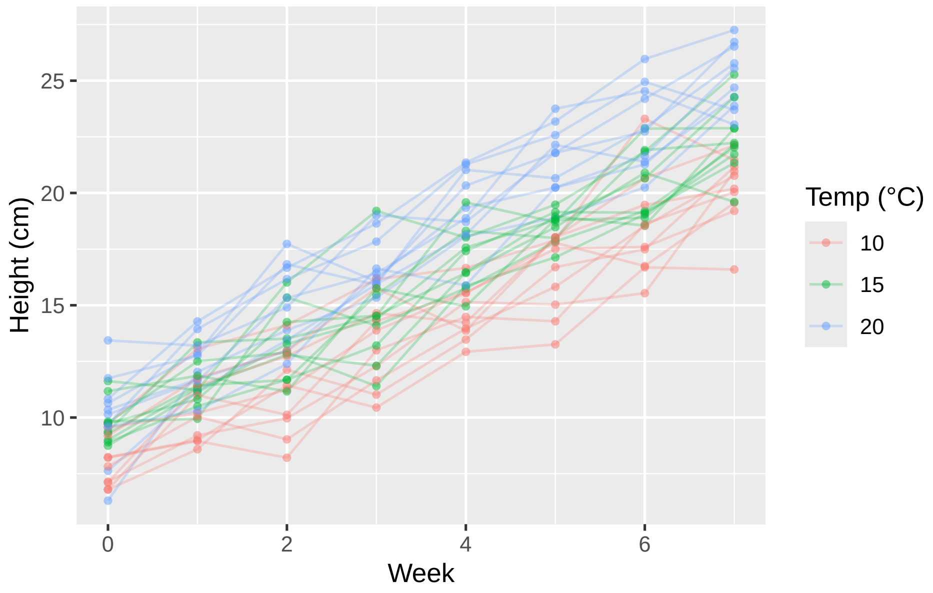
```
Rows: 240 Columns: 4
-- Column specification --------------------------------------------------------
Delimiter: ","
chr (1): plant_id
dbl (3): temp, week, height

i Use `spec()` to retrieve the full column specification for this data.
i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```r
head(dat)
```

```
# A tibble: 6 x 4
  plant_id temp   week height
  <fct>    <fct> <int>  <dbl>
1 P01      10        0   6.79
2 P01      10        1   8.59
3 P01      10        2  12.1
4 P01      10        3  11.0
5 P01      10        4  13.5
6 P01      10        5  16.7
```

Explore the data

First we make a plot of the data:



We can see data from each individual plant because it is connected by a line. We also see that plants at higher temperature tend to be taller. And we see that plants differ in their baseline height (week 0).

There are two challenges here: 1. We have repeated measures on the same plants (non-independence). 2. We have variation among plants in baseline height.

A mixed model can handle both of these.

Wrong model: treat all rows as independent

```r
m_lm <- lm(height ~ week * temp, data = dat)
anova(m_lm)
```

```
Analysis of Variance Table

Response: height
           Df Sum Sq Mean Sq   F value    Pr(>F)    
week        1 4478.8  4478.8 1888.8551 < 2.2e-16 ***
temp        2  586.9   293.5  123.7678 < 2.2e-16 ***
week:temp   2   46.9    23.4    9.8838 7.572e-05 ***
Residuals 234  554.8     2.4                        
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

This model pretends there are $30 \times 8 = 240$ independent data points. But most of that information is repeated measures on the same plants.

We can see that this is pseudoreplication because there are many more residual degrees of freedom (236) than plants (30). This is a classic sign of pseudoreplication. We really cannot trust the p-values or confidence intervals from this model.

Mixed model: random intercept for plant

Now let's fit a mixed model with plant identity as a random effect:

```
m_lmm1 <- lmer(height ~ week * temp + (1 | plant_id), data = dat)
anova(m_lmm1)
```

```
Type III Analysis of Variance Table with Satterthwaite's method
          Sum Sq Mean Sq NumDF  DenDF   F value    Pr(>F)    
week      4478.8  4478.8     1 207.00 3264.1929 < 2.2e-16 ***
temp        21.6    10.8     2  46.37    7.8666  0.001144 ** 
week:temp   46.9    23.4     2 207.00   17.0806 1.362e-07 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

In the anova table, note that there are no degrees of freedom and no p-values for fixed effects. This is because calculating these in mixed models is complicated and there are multiple methods. We will look at this more later in this chapter.

Interpretation:

- fixed effects describe the **average** relationship between height, week, and temperature
- the random intercept captures baseline differences among plants

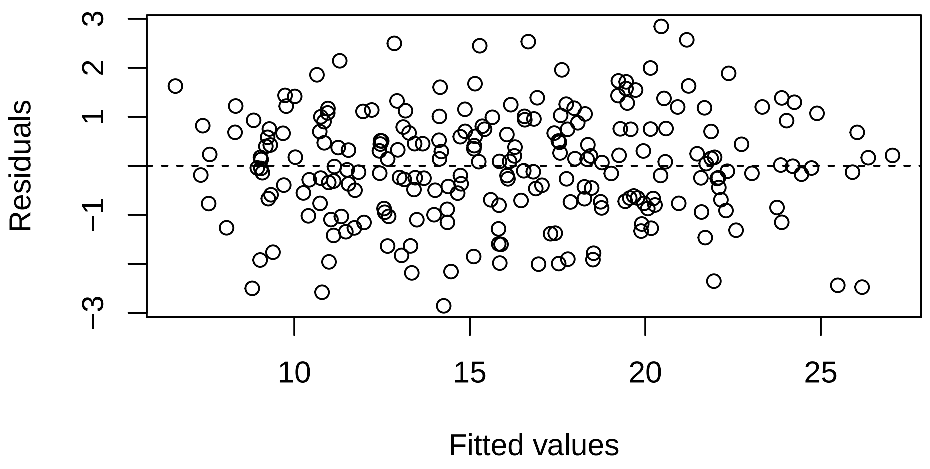
model checking

Mixed-model diagnostics (i.e., model checking) can be more involved than `lm()`, but you can still start with:

- residual vs fitted plot (nonlinearity / heteroscedasticity)
- normal Q–Q of residuals (approximate)
- check random effect estimates for extreme outliers

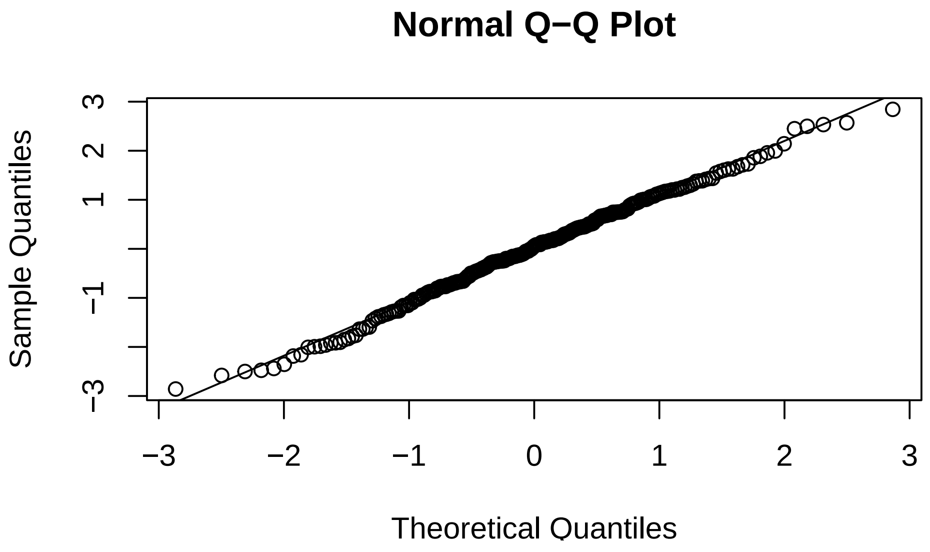
1) Residuals vs fitted

```
plot(fitted(m_lmm1), resid(m_lmm1),
     xlab = "Fitted values", ylab = "Residuals")
abline(h = 0, lty = 2)
```



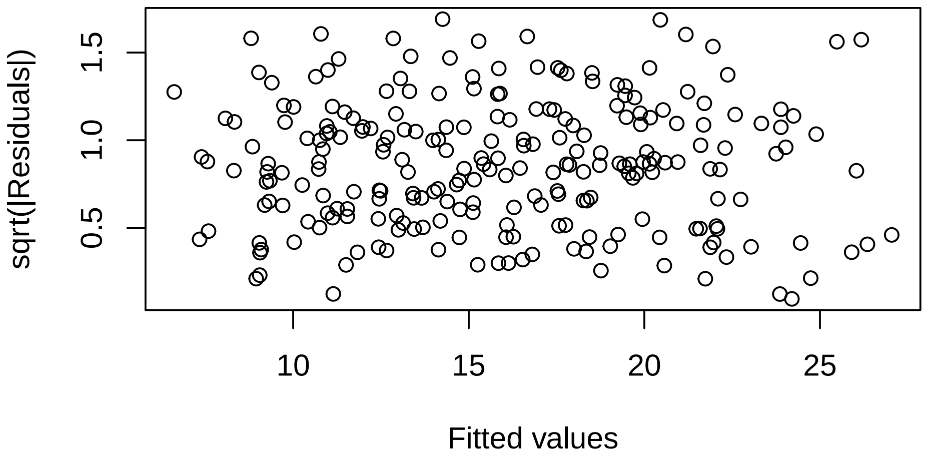
2) Normal Q–Q of residuals

```
qqnorm(resid(m_lmm1)); qqline(resid(m_lmm1))
```



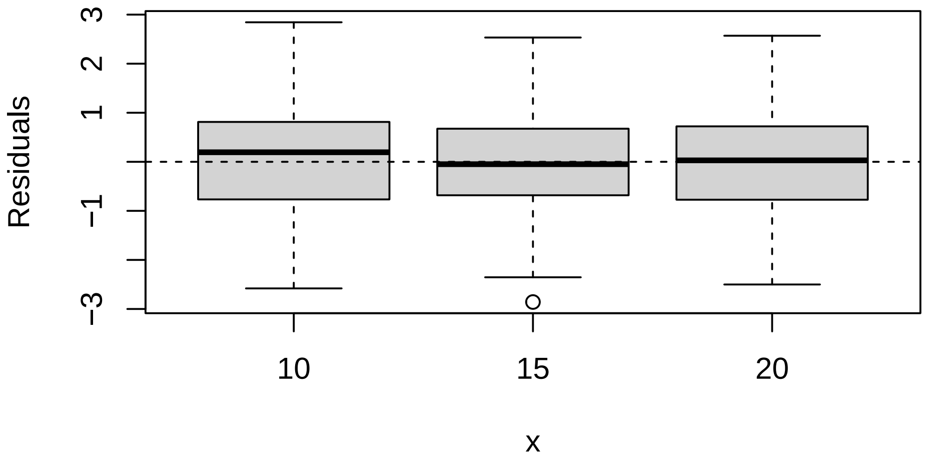
3) Scale–location (sqrt(|resid|) vs fitted)

```
plot(fitted(m_lmm1), sqrt(abs(resid(m_lmm1))),
     xlab = "Fitted values", ylab = "sqrt(|Residuals|)")
```



4) Residuals vs explanatory variable (helpful for nonlinearity with a continuous explanatory variable)

```r
plot(dat$temp, resid(m_lmm1), xlab = "x", ylab = "Residuals")
abline(h = 0, lty = 2)
```



Caution

Don't over-interpret p-values
In mixed models, inference depends on how you handle degrees of freedom and uncertainty. In BIO144, focus on: - correct model structure (what must be random?) - effect sizes and uncertainty (CIs) - sensible plots and biological interpretation

Extension: random slopes for week

If you believe plants differ in growth rate (not only baseline), add a random slope:

```r
m_lmm2 <- lmer(height ~ week * temp + (1 + week | plant_id), data = dat)
```

```
boundary (singular) fit: see help('isSingular')
```

```r
summary(m_lmm2)
```

```
Linear mixed model fit by REML. t-tests use Satterthwaite's method [
lmerModLmerTest]
Formula: height ~ week * temp + (1 + week | plant_id)
   Data: dat

REML criterion at convergence: 822.6

Scaled residuals:
     Min       1Q   Median       3Q      Max
-2.43220 -0.64862  0.04439  0.63469  2.31852

Random effects:
 Groups   Name        Variance  Std.Dev. Corr
 plant_id (Intercept) 0.9012460 0.94934
          week        0.0006804 0.02608  1.00
 Residual             1.3682645 1.16973
Number of obs: 240, groups:  plant_id, 30

Fixed effects:
             Estimate Std. Error        df t value Pr(>|t|)
(Intercept)   8.12308    0.38358  29.45344  21.177  < 2e-16 ***
week          1.69948    0.05767 158.78156  29.469  < 2e-16 ***
temp15        1.45872    0.54247  29.45344   2.689 0.011676 *
temp20        2.24843    0.54247  29.45344   4.145 0.000263 ***
week:temp15   0.10592    0.08156 158.78156   1.299 0.195916
week:temp20   0.45169    0.08156 158.78156   5.538 1.24e-07 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Correlation of Fixed Effects:
            (Intr) week   temp15 temp20 wk:t15
week        -0.404
temp15      -0.707  0.285
temp20      -0.707  0.285  0.500
week:temp15  0.285 -0.707 -0.404 -0.202
week:temp20  0.285 -0.707 -0.202 -0.404  0.500
```

```
optimizer (nloptwrap) convergence code: 0 (OK)
boundary (singular) fit: see help('isSingular')
```

Compare the two mixed models:

```
anova(m_lmm1, m_lmm2)
```

```
refitting model(s) with ML (instead of REML)
```

```
Data: dat
Models:
m_lmm1: height ~ week * temp + (1 | plant_id)
m_lmm2: height ~ week * temp + (1 + week | plant_id)
       npar    AIC    BIC  logLik deviance  Chisq Df Pr(>Chisq)
m_lmm1    8 826.51 854.35 -405.25   810.51
m_lmm2   10 830.01 864.82 -405.00   810.01 0.4974  2     0.7798
```

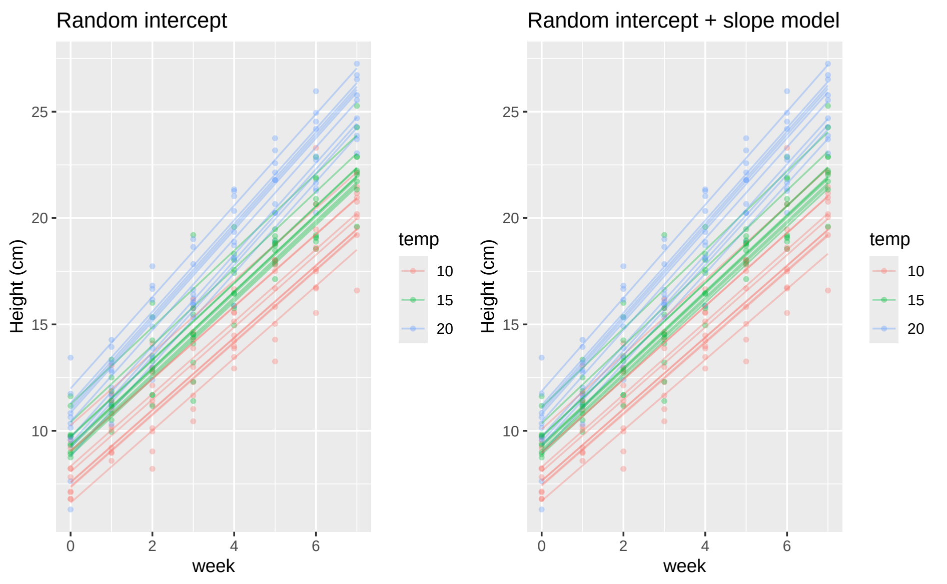
There is little evidence that adding random slopes improves the model here (p = 0.77). But in other datasets it might.

Note

This likelihood ratio test compares nested models. It is widely used, but has subtleties for random effects. In BIO144 you can treat it as a reasonable practical tool, while noting that "testing random effects" is an advanced topic.

Visualising fitted values

We can visualise the fitted values from both mixed models:



There is not much difference between the two models here, but in other datasets random slopes can make a big difference.

Significance testing for fixed effects

Calculating p-values for fixed effects in mixed models is complicated, and there are multiple methods (Satterthwaite, Kenward-Roger, likelihood ratio tests, bootstrapping, Bayesian credible intervals). It is difficult because the degrees of freedom depend on the random effects structure and the data. There is no clear and objective method to get the degrees of freedom.

Neverthless, we can get p-values for terms using the `lmerTest` package (optional). This changes the `lmer()` function to provide p-values using Satterthwaite's method for degrees of freedom.

```r
m_lmm1 <- lmer(height ~ week * temp + (1 | plant_id), data = dat)
anova(m_lmm1)
```

```
Type III Analysis of Variance Table with Satterthwaite's method
          Sum Sq Mean Sq NumDF  DenDF   F value    Pr(>F)
week      4478.8  4478.8     1 207.00 3264.1929 < 2.2e-16 ***
temp        21.6    10.8     2  46.37    7.8666  0.001144 **
week:temp   46.9    23.4     2 207.00   17.0806 1.362e-07 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Reporting (template)

- We modelled plant height as a function of week, temperature, and their interaction using a linear mixed model with plant identity as a random effect.

- Height increased with week (fixed effect of week), and plants at higher temperature were taller on average (fixed effect of temperature).
- All fixed effects had a p-value < 0.05, calculate using Satterthwaite's method for degrees of freedom (lmerTest package).
- Including plant as a random effect accounted for non-independence due to repeated measures.

Review

- Mixed models are used when observations are **grouped** (non-independent).
- They combine **fixed effects** (average relationships) with **random effects** (group-to-group variation).
- Random intercepts model different baselines among groups; random slopes allow different responses.
- Nested and crossed random effects reflect study design.
- Mixed models let you keep all observations while avoiding pseudoreplication and false confidence.

Think–Pair–Share (#a_why_random_effects) What problem do random effects solve that fixed effects alone cannot?

Think–Pair–Share (#a_partial_pooling_intuition) How does partial pooling differ from complete pooling or no pooling at all? Hint: pooling is sharing of information.

What next (L11-2)

During this course, you have learned a variety of data analysis techniques using R, including data manipulation, visualization, and statistical modeling. You have gained a solid foundation in using R for data analysis, you can analyse data that meets the assumptions of linear models, and some types of data that do not (e.g., count data and binary data using GLMs).

Of course, there is much more to learn! There are many opportunities for you to further develop your skills in R and data analysis. What is the next step after this course? What are the options to further improve your skills in data analysis in R? What other types of analyses could you learn about, and when might you need them?

Here are a list of other types of problem / question that we might have, and types of analysis that could be relevant. The list is by no means exhaustive, but it should give you some ideas of what to explore next, and what to explore when you encounter specific types of data or research questions.

- **Time series analysis**: If your data are collected over time (e.g., daily, monthly, yearly), you might need to learn about time series analysis techniques such as ARIMA models, seasonal decomposition, and forecasting methods. A key feature of time series data is that observations are not independent, which violates assumptions of many standard statistical methods. This needs to be carefully handled in the analysis.

- **Spatial analysis**: If your data have a spatial component (e.g., locations, regions), you might need to learn about spatial statistics, geostatistics, and spatial modeling techniques. This could include methods such as kriging, spatial autocorrelation analysis, and spatial regression models. Again, spatial data often violate independence assumptions, requiring specialized methods.

- **Non-linear regression**: If the relationship between your explanatory variables and response variable is not linear, you might need to learn about non-linear regression techniques. These can estimate the parameters of specific non-linear functions, and to assess the goodness of fit.

- **Breakpoint analysis**: If you suspect that there are changes in the relationship between variables at certain points (e.g., before and after an intervention), you might need to learn about breakpoint analysis techniques, such as piecewise regression or change point detection methods.
- **Generalized Additive Models (GAMs)**: If you want to model complex, non-linear relationships between explanatory variables and response variables while maintaining some interpretability, you might need to learn about GAMs. These models use smooth functions to capture non-linear effects. They are rather elegant!
- **Structural Equation Modeling (SEM)**: If you want to analyze complex relationships among multiple variables, including latent variables, you might need to learn about SEM techniques. SEM allows for the modeling of direct and indirect effects, as well as measurement error. Effectively, we can build and test complex causal models. Variables can be both explanatory variables and responses at the same time.
- **Machine Learning**: If you want to make predictions or classify data based on patterns, you might need to learn about machine learning techniques such as decision trees, random forests, support vector machines, and neural networks. These methods can handle large datasets and complex relationships but may sacrifice some interpretability.
- **Meta-analysis**: If you want to synthesize results from multiple studies to draw broader conclusions, you might need to learn about meta-analysis techniques. This involves combining effect sizes from different studies and assessing heterogeneity among them.
- **Survival analysis**: If your data involve time-to-event response variables (e.g., time until failure, time until death), you might need to learn about survival analysis techniques such as Kaplan-Meier estimation, Cox proportional hazards models, and parametric survival models.
- **Non-parametric methods**: If your data do not meet the assumptions of parametric tests (e.g., normality, homoscedasticity), and you really can't figure out how to make a parametric model (e.g., LM or GLM) you might need to learn about non-parametric methods such as rank-based tests, bootstrapping, and permutation tests.
- **Power analysis and sample size estimation**: If you want to design studies with adequate statistical power, you might need to learn about power analysis techniques. This involves calculating the required sample size based on effect sizes, significance levels, and desired power.
- **Bayesian statistics**: If you want to incorporate prior knowledge and uncertainty into your analyses, you might need to learn about Bayesian statistical methods. This involves using Bayes' theorem to update prior beliefs based on observed data.

These are just a few examples of the many types of analyses that you might encounter in your data analysis journey. The choice of which techniques to learn next will depend on your specific research questions, data characteristics, and goals. Ideally you will plan your analyses when you design your study, so that you can collect the right type of data to answer your questions. When you don't or when something changes, you can then explore and discussion with experts which techniques are most appropriate.

A final word of advice… try to not be driven by techniques. Instead, be driven by your research questions. After all, we are not doing data analysis for its own sake, but to answer questions about the world around us. Let your questions guide your learning journey!

Review (L12)

You need to prepare in advance for this lecture.

In the final lecture of the course, we will review and repeat any of the content that you find challenging. Please write in the Forum which topics you would like to revisit, so that we can focus on those during the session. There may also be chances to ask during the lecture time questions on topics that you find difficult.

So the preparation for this lecture is to think about which topics you would like to discuss again, and to post these in the Forum.

Here are some ideas of topics that you might want to revisit:

- Types of study designs in biology.
- Framing research questions
- Observational studies
- Experimental studies

- Key modeling concepts: interactions, overfitting, degrees of freedom, variance explained, hypothesis testing, null hypothesis

- Parametric vs non-parametric approaches: what they mean and when to use them