BIO144 Course Book (version 2026)

Owen Petchey

2025-11-22

Table of contents

Preface

⚠ Warning

Under construction: The 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 the version you have may not exactly match the lectures you attend. However, all of the required content will be the same, and any changes will be correcting typos or improving explanations.

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!

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

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

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.

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!

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:

Then type the following code:

```
[1] 2
```

```
[1] 7.389056
```

```
[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:

And then print the value of `c`:

```
[1] 15
```

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

We can also create vectors:

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

And do maths on vectors:

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

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

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

We can vectors of strings (text):

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

And can perform operations on strings:

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

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

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

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:

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.

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:

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.

> ℹ 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 `data.csv`, we can use the following code:

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.

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:

```
[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:

```
[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:

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:

```
  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()` func-

tion 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:

```
Attaching package: 'janitor'

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

    chisq.test, fisher.test
```

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

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:

Here is the same dataset with 100 rows:

Select columns [#select-columns]

We can select columns by name

```
# 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

```
# 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:

```
# 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":

```
# 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:

```
# 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:

```
# 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:

```
# 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 [#arrange-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:

```
# 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:

```
# 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:

```
# 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:

```
# 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 [#sec-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:

```
# 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:

Look at the structure of this new dataset:

```
# 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
```

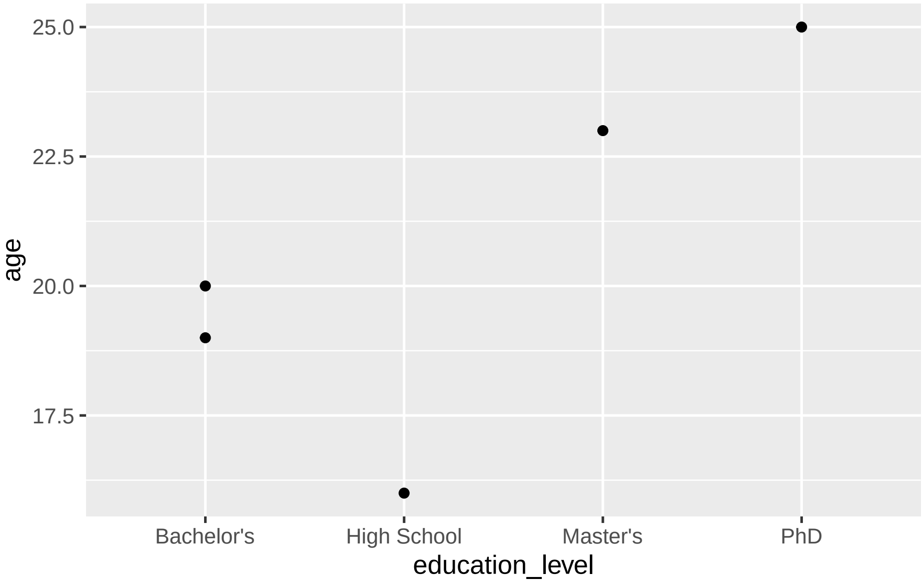
We can see that the `education_level` variable is of type `<chr>`, which is `character`.

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

```
# 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:



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".

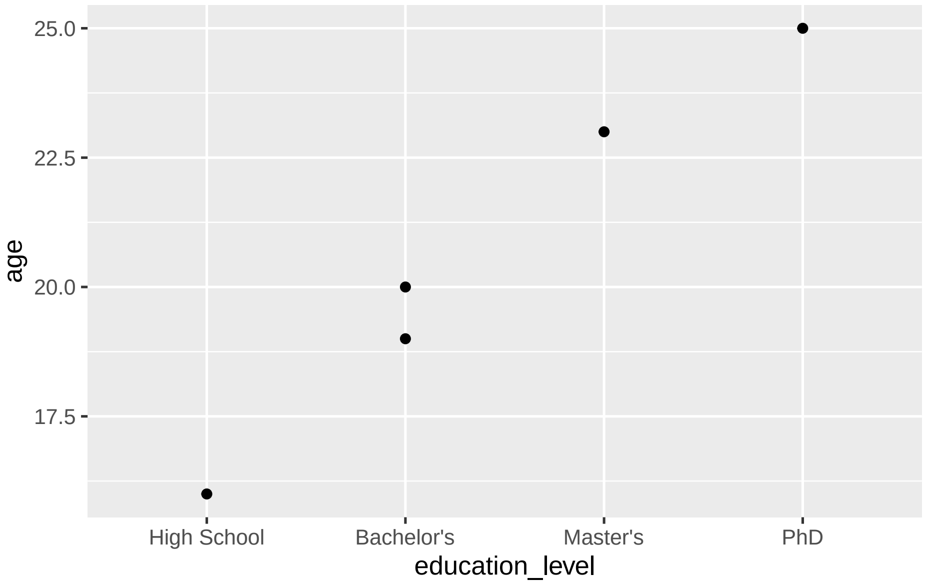
ℹ Note

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:

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



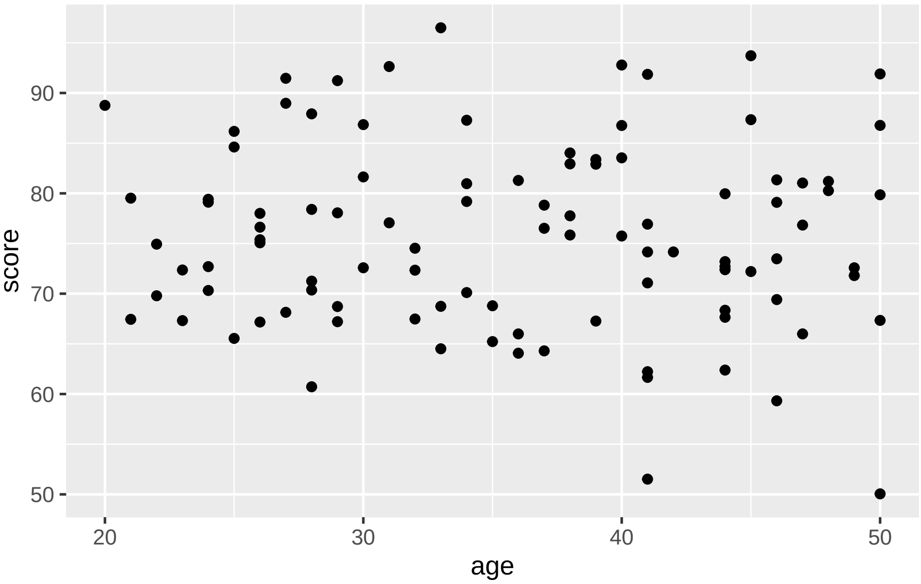
Another use case is when we are making a linear model and want to specify the reference level for a categorical variable. We will look at this when we get to linear models. If you want to skip ahead, you can see how this works in a section at the end of this chapter (Section **?@sec-ref-level**).

Visualisation

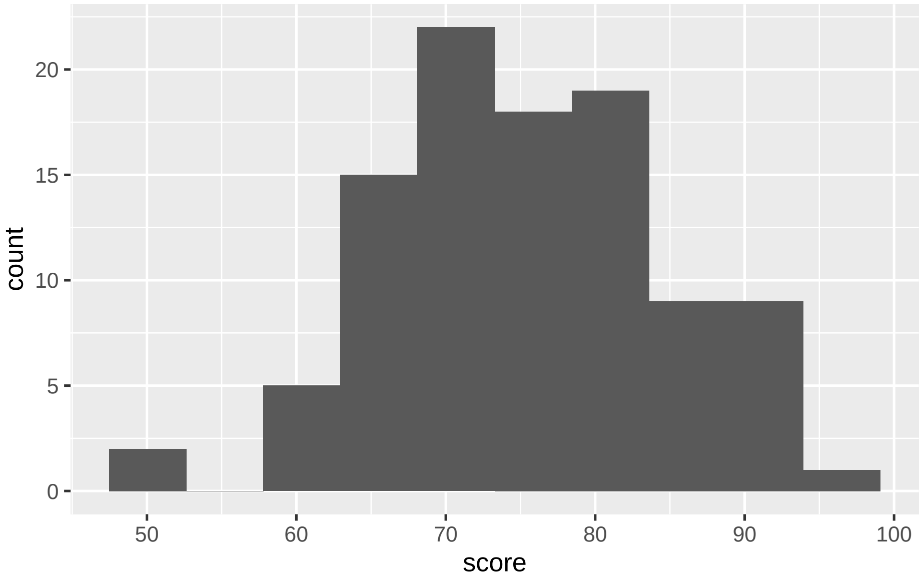
There are many many many types of data visualisation. We will not explore them all in this course! In fact, we will use only a few basic types of visualisation, but we will use them well and critically. The three types of visualisation we will focus on are scatter plots, histograms, and box and whisker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

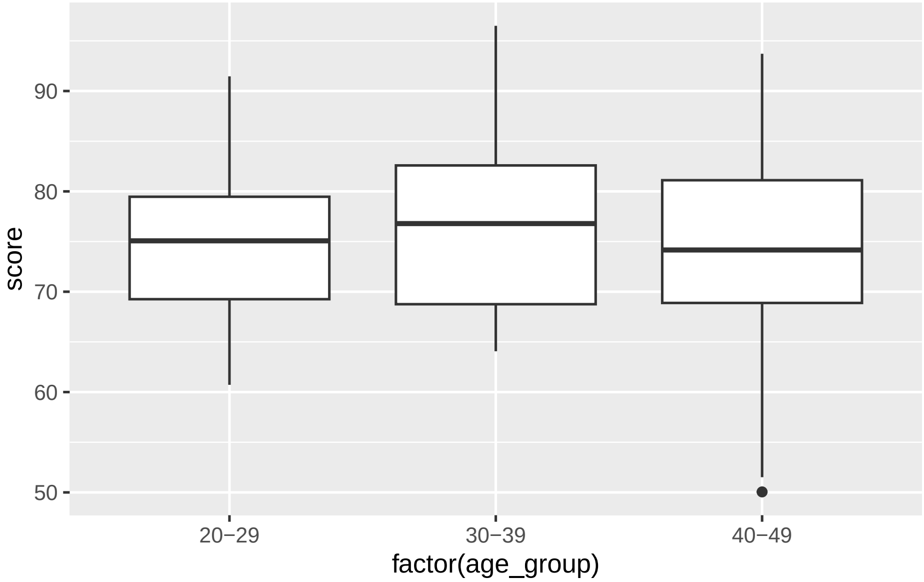


Histograms are used to visualise the distribution of a single continuous variable. The axiss are different to scatterplots: the x-axis is the variable being measured, and the y-axis is the count (or frequency) of observations in each bin. A bin is a range of values. Here is an esample of a histogram:



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:

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:



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.

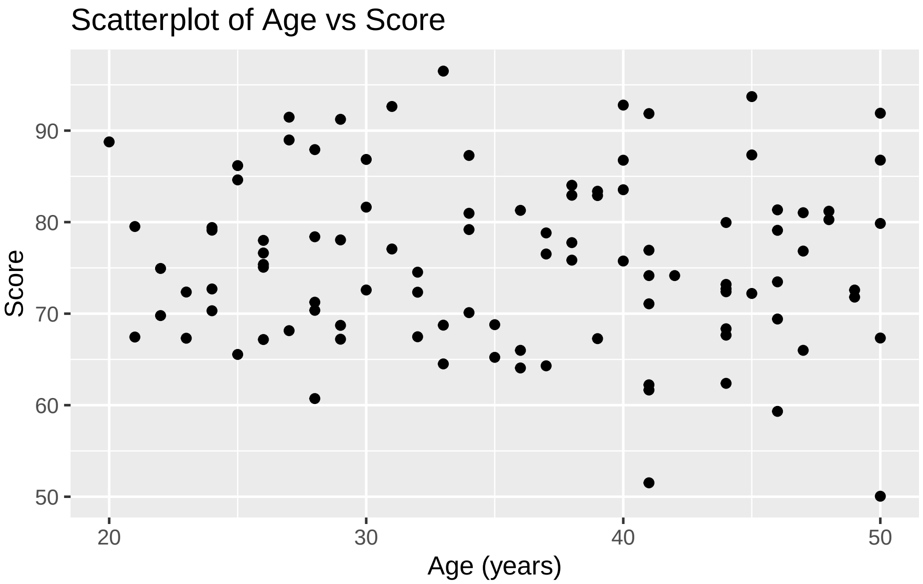
The first argument to the `ggplot()` function is the data frame that we want to visualise. The layers that we add to the plot each have two main components. The first component is the *aesthetic mappings*, which specify how the variables in the data frame are mapped to the visual properties of the plot (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)`.

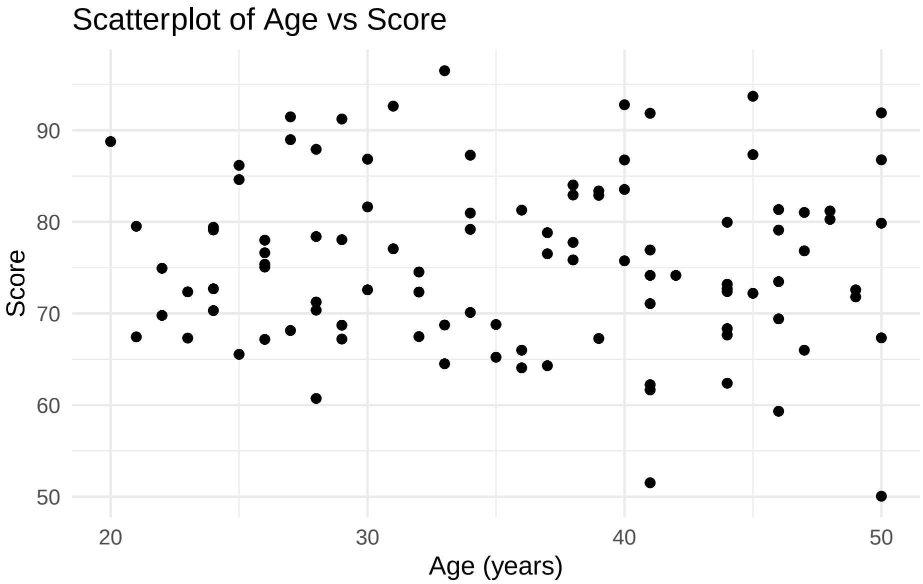
The *geometric object* is specified using functions that start with `geom_`, such as `geom_point()`, `geom_histogram()`, and `geom_boxplot()`.

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:



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:



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:

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

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:

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

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):

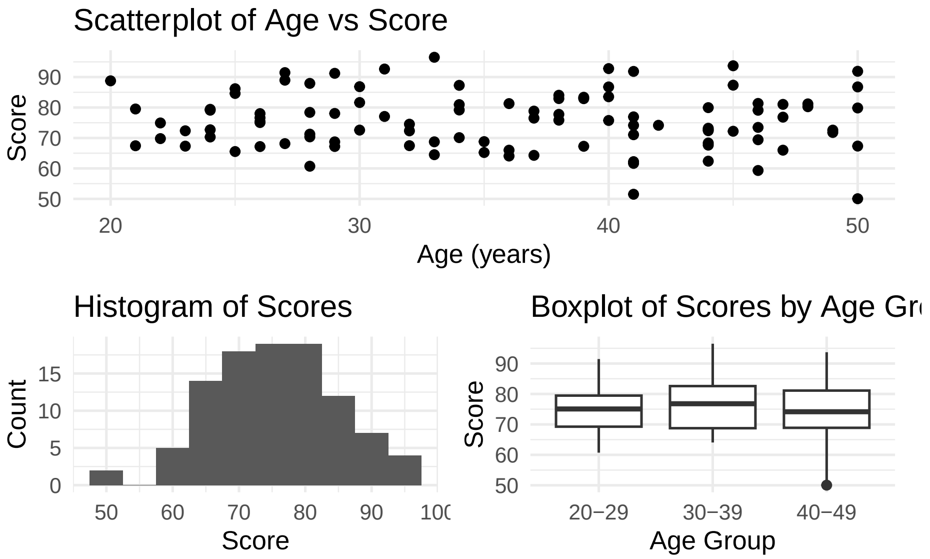
First, load the `patchwork` package:

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

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

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

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:



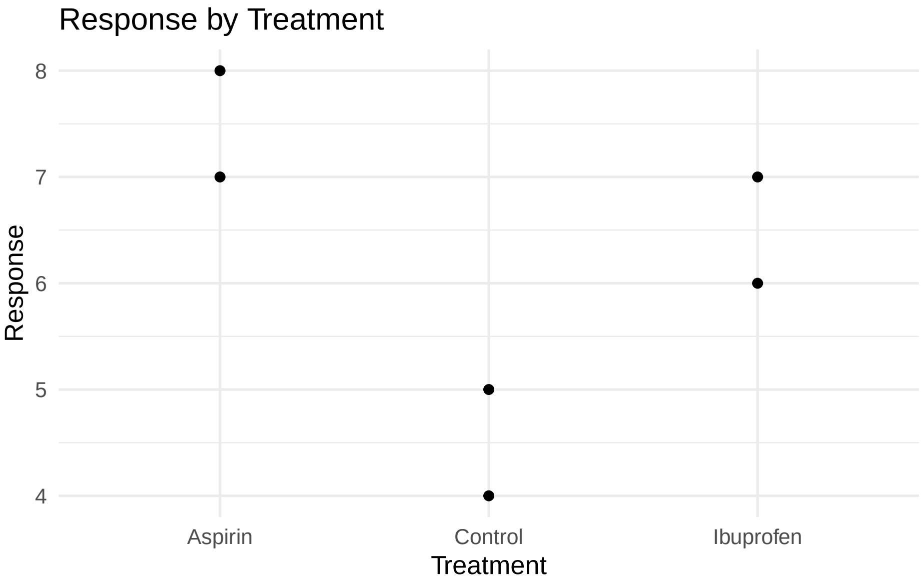
Amazing eh! OK, lets leave it there for now. We'll use ggplot2 throughout the course, and explore more features as we go along.

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:

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:



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

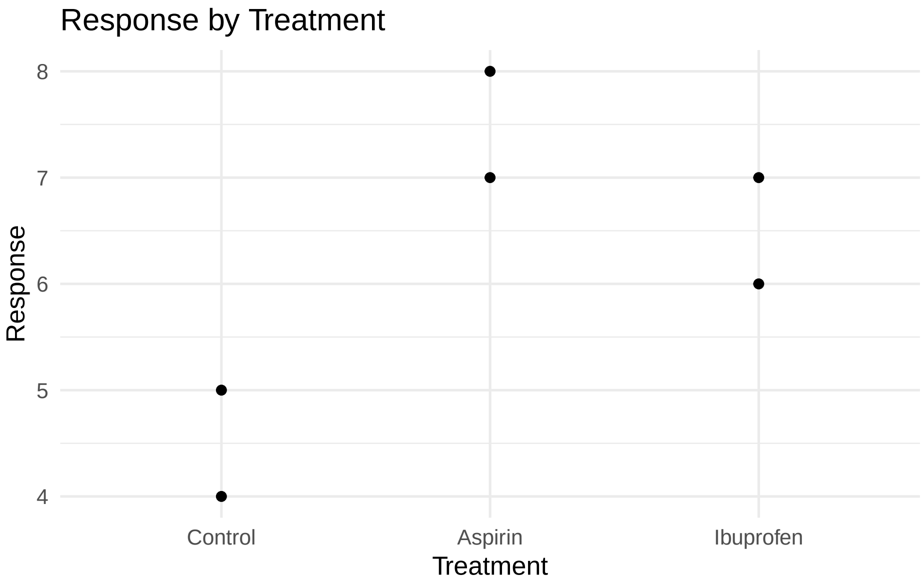
Likewise, when we make a linear model:

```

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

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()`:

Now when we visualise the data again:



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

And when we fit the linear model again:

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

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.

Study design (L3)

> ℹ Note
>
> This chapter in under development.

Different studies, different goals, different designs, different analyses

Paradigms of study design

Observational studies

Observational studies are used to observe and analyze the effects of interventions or exposures without manipulating the study environment.

Experimental studies

Experimental studies involve the manipulation of one or more variables to determine their effect on a response variable. The manipulation is typically done through random assignment of subjects to different treatment groups, which means that on average the only thing that differs among the subjects in the different groups is the treatment they receive. This helps to ensure that any differences in outcomes can be attributed to the treatment rather than other factors that might affect the subjects.

Key aspects of experimental studies:

- A treatment is applied to one or more groups of subjects.
- Subjects are randomly assigned to treatment groups, which helps to control for confounding variables.
- There are typically one or more control groups that do not receive the treatment, allowing for comparison of outcomes between treated and untreated groups.

- There is replication, meaning that there are multiple subjects in each treatment group to ensure that estimates of within group variability can be made.

Other terms

Unfortunately, the terms “observational” and “experimental” are not always used consistently. In particular, the term “experiment” is often used to refer to any study that involves a comparison of two or more groups, regardless of whether the groups were assigned randomly or not. This can lead to confusion, as some studies that are called “experiments” are actually observational studies. Some studies use the term “mensurative experiment” to refer to observational studies (e.g., Burdon *et al* 2018 *Global Change Biology*).

Regression Part 1

How Owen will structure the lecture time

The chapter content below is the reference for what students are expected to know. During the lecture time Owen will talk through and explain hopefully most of the content of this chapter. He will write on a tablet, show figures and other content of this chapter, and may live-code in RStudio. He will also present questions and ask students to *Think, Pair, Share.* The *Share* part will sometime be via clicker, sometimes by telling the class. The same will happen in lecture 3-6.

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!!! See **?@sec-kind-of-magic**.).

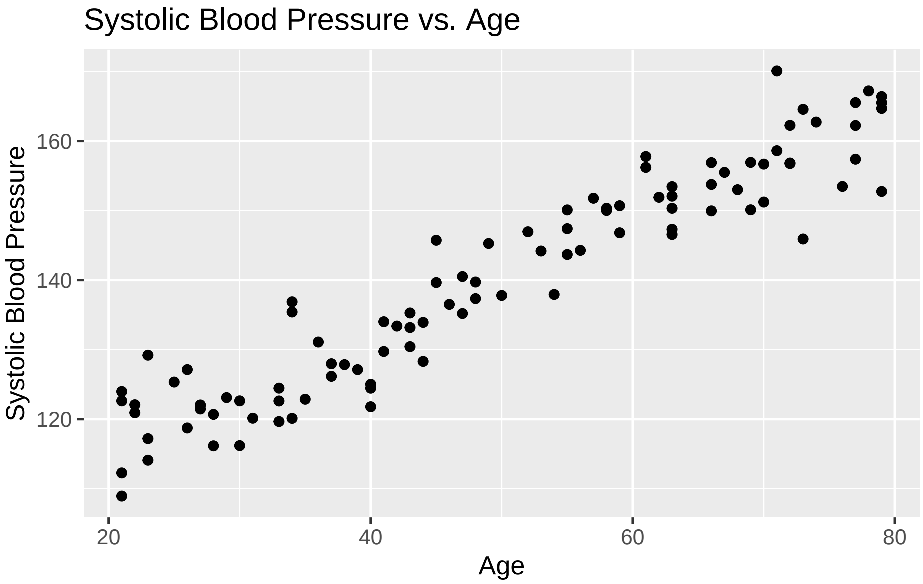
An example - blood pressure and age

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:



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 (#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.

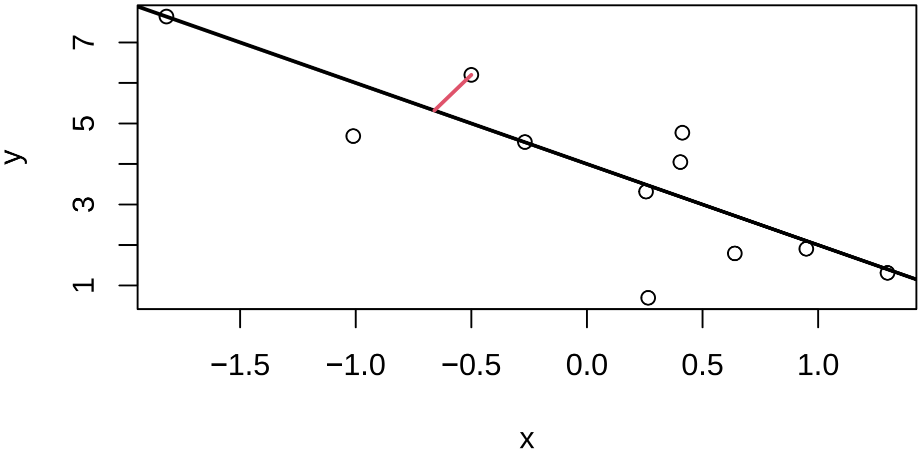
Finding the intercept and the slope

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.

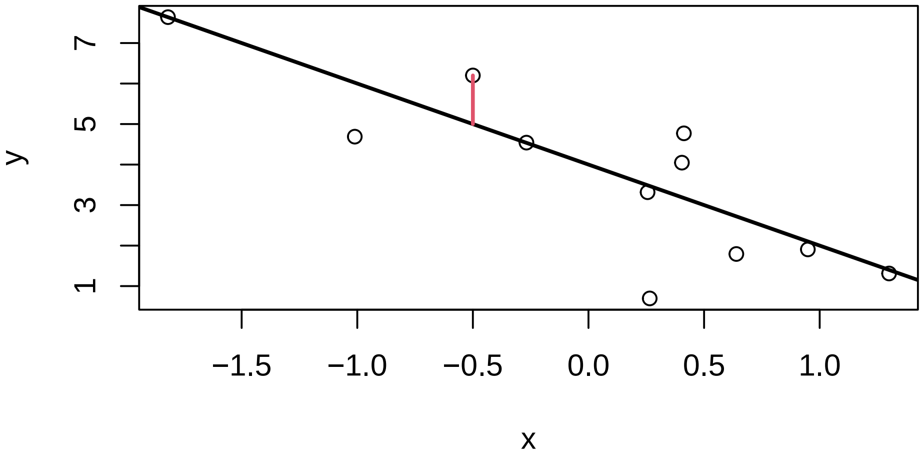
- **Problem**: For more than two points (x_i, y_i), $i = 1, \dots, n$, there is generally no perfectly fitting line.
- **Aim:** We want to estimate the parameters (β_0, β_1) of the **best fitting** line $Y = \beta_0 + \beta_1 x$.
- **Idea:** Find the **best fitting line** by minimizing the deviations between the data points (x_i, y_i) and the regression line. I.e., minimising the residuals.

But which deviations?

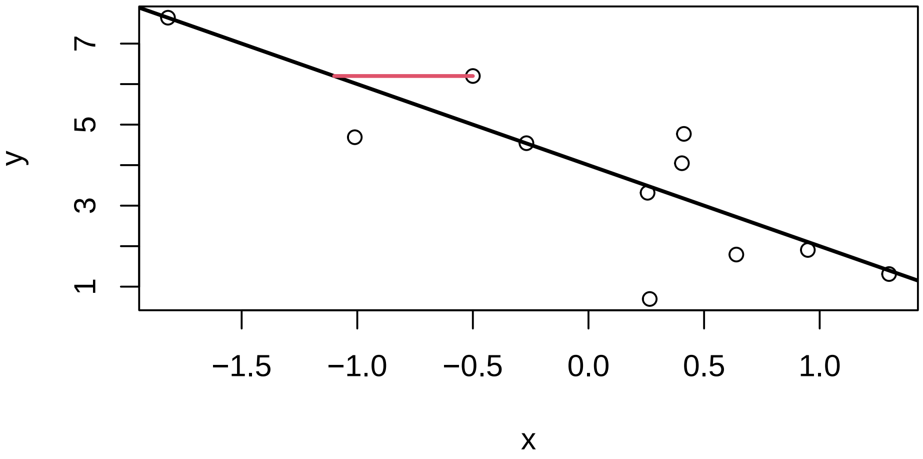
These ones?



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:

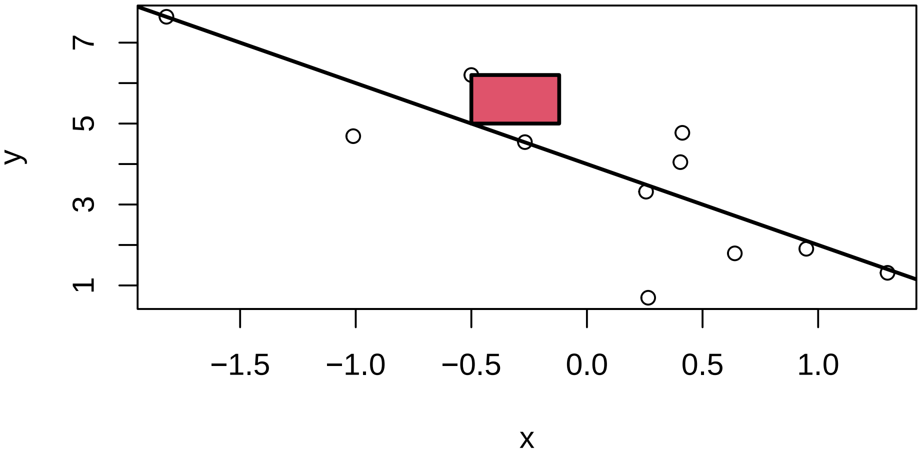
$$SSE = \sum_{i=1}^{n} e_i^2$$

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.



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.

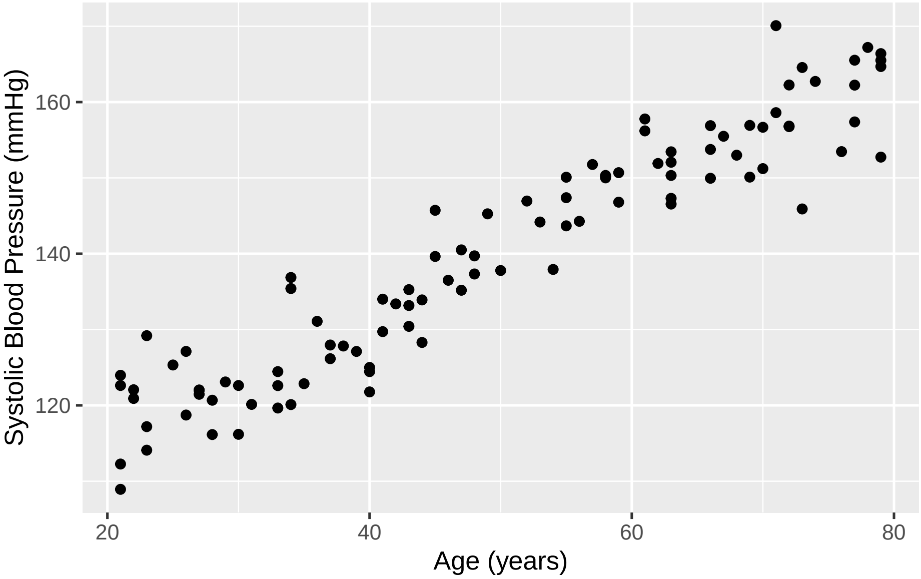
We'll look at degrees of freedom in more detail later, so don't worry if this is a bit confusing right now.

Let's do it in R

First we read in the dataset:

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

The we make a graph of the data:



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

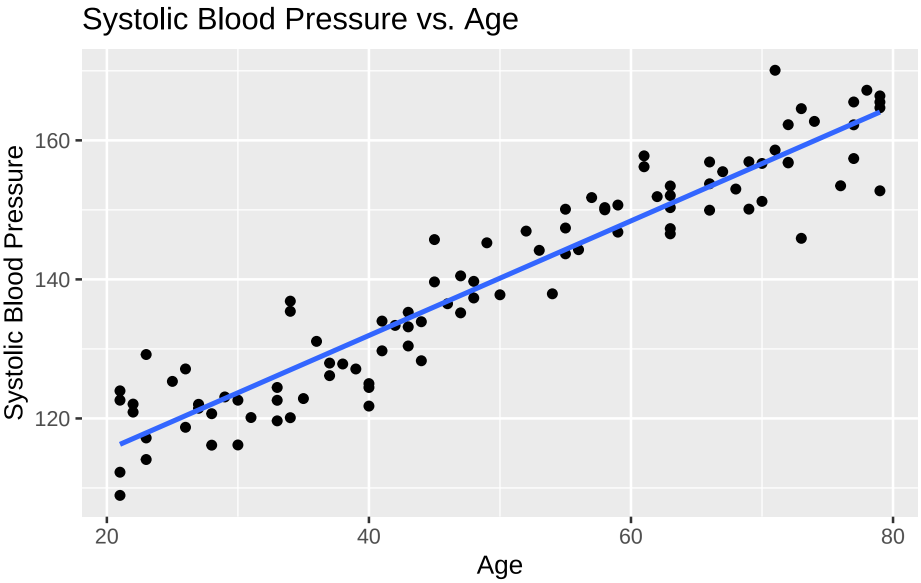
Then we can look at the summary of the model. It contains a lot of information, so can be a bit confusing at first.

```

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.

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

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

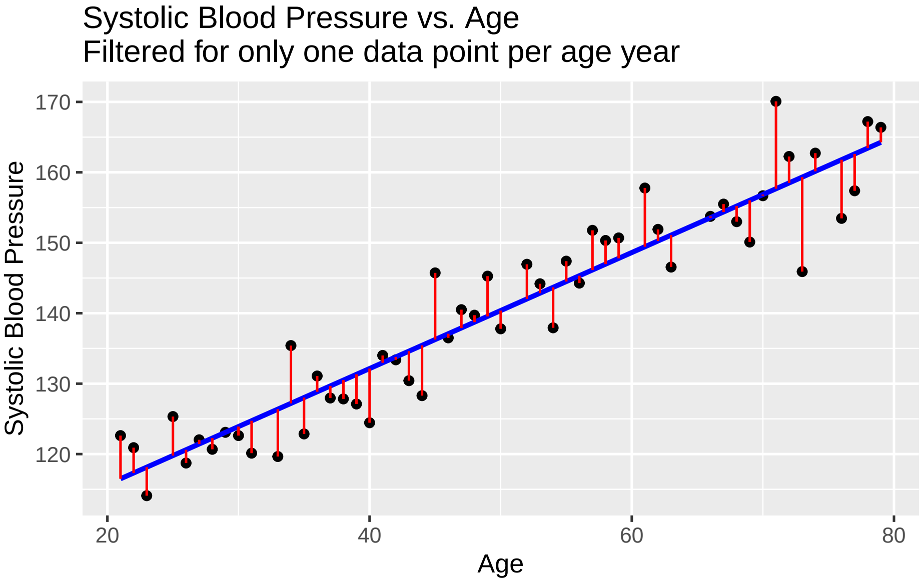
where

- 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.

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

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:

```
[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.

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

```
(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:

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

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

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 (#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.

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

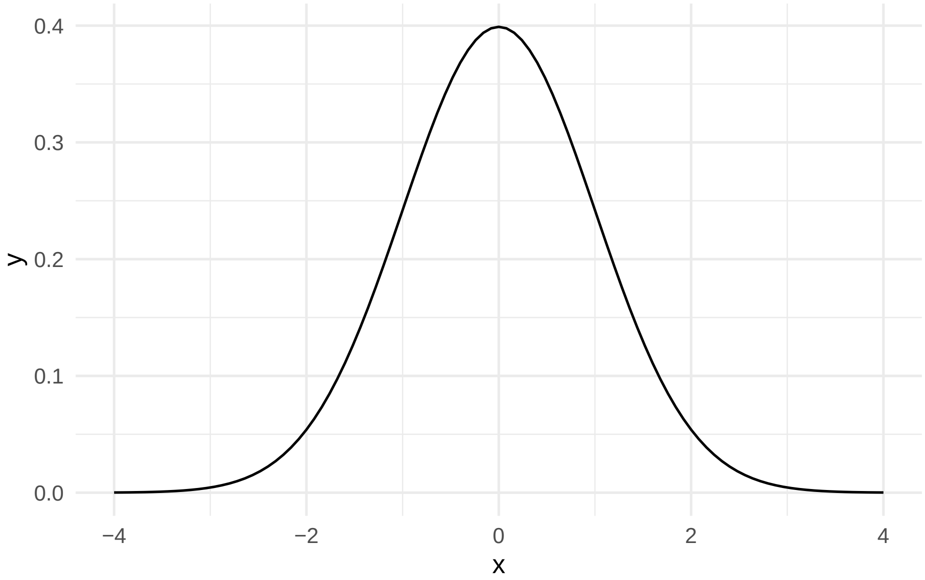
Recall that we make the assumption that the residuals are normally distributed with mean 0 and constant variance:

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

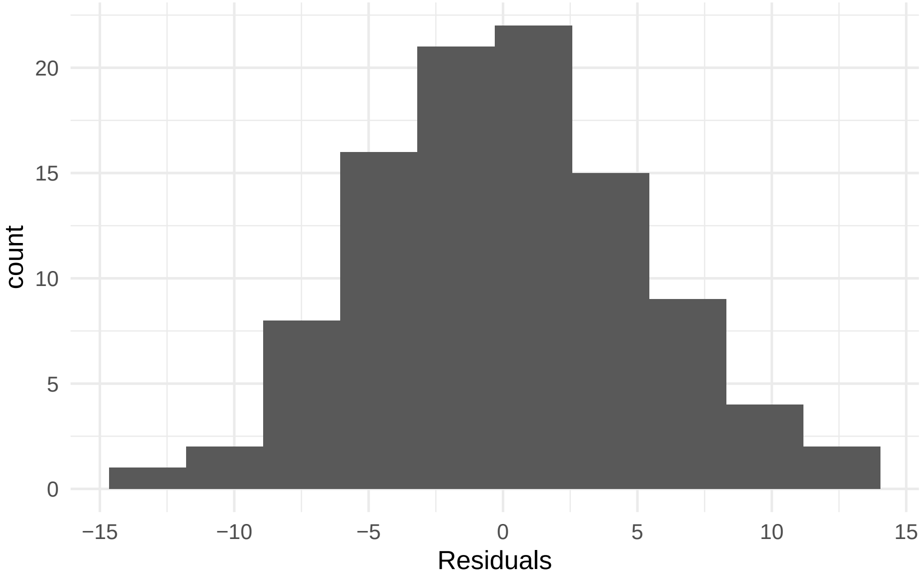
Here we are concerned with the first part of this assumption, that the residuals are normally distributed.

What does this mean? How can we check it?

A normal distribution is symmetric and bell-shaped…



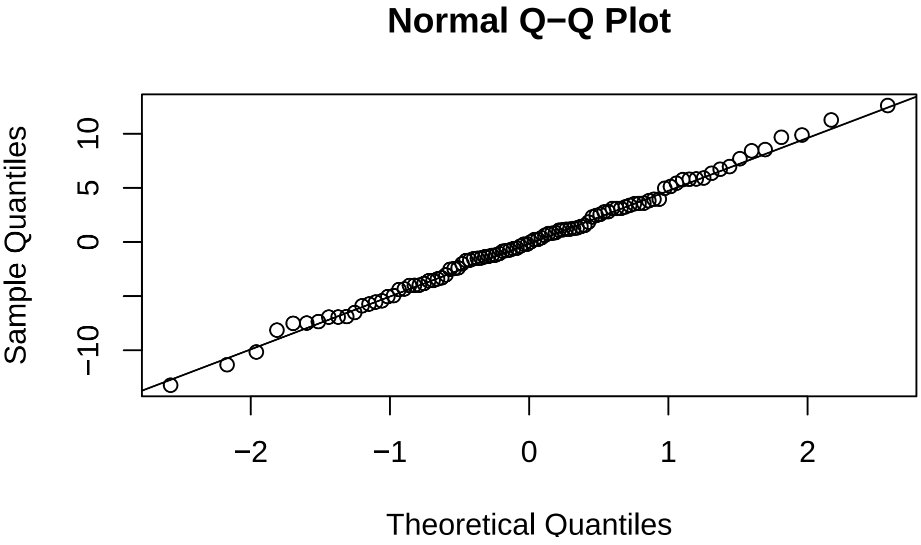
Lets look at the frequency distribution of the residuals of the linear regression of blood pressure and age:



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:

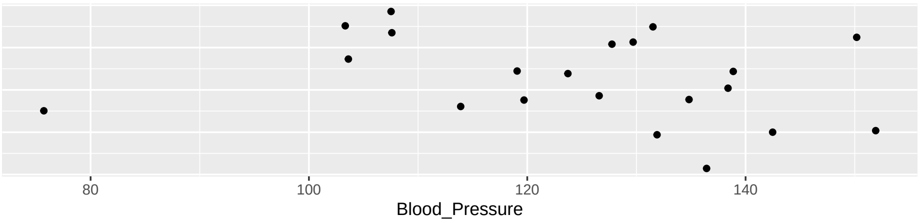


If the points lie approximately on a straight line, the data is fairly normally distributed.

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

But what on earth is a quantile???

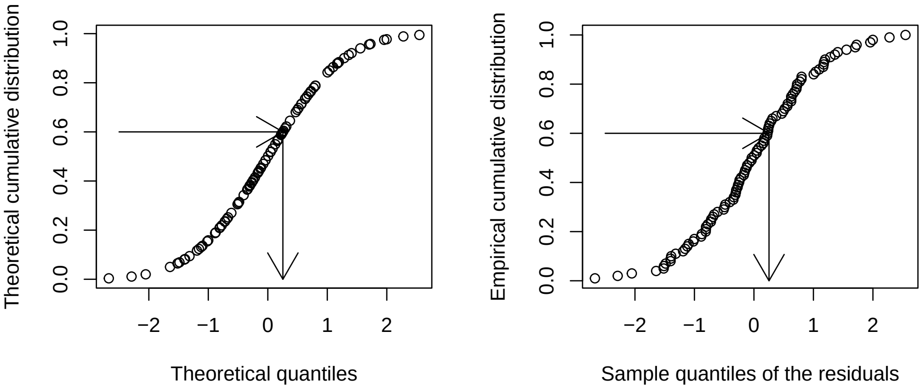
Imagine we make 21 measures of something, say 21 blood pressures:



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.

```
  50%
127.8
```

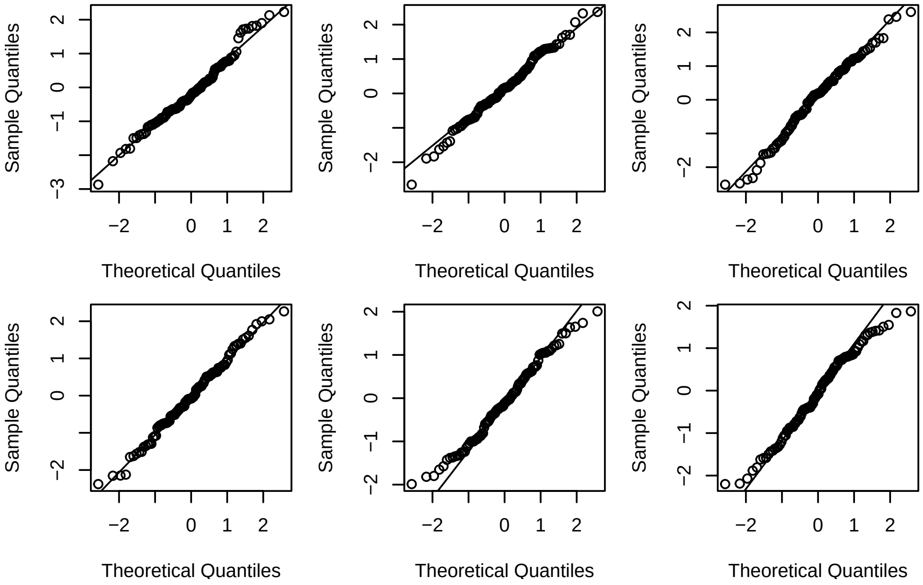
The *theoretical quantiles* come from the normal distribution. The *sample quantiles* come from the distribution of our residuals.



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:

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

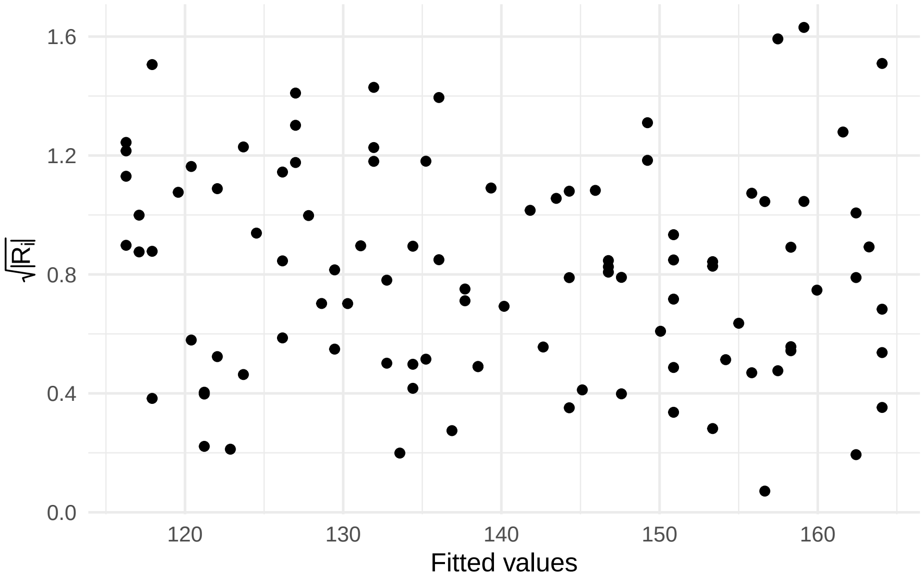
So that the full equation of the square root of the standardised residuals is:

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

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:

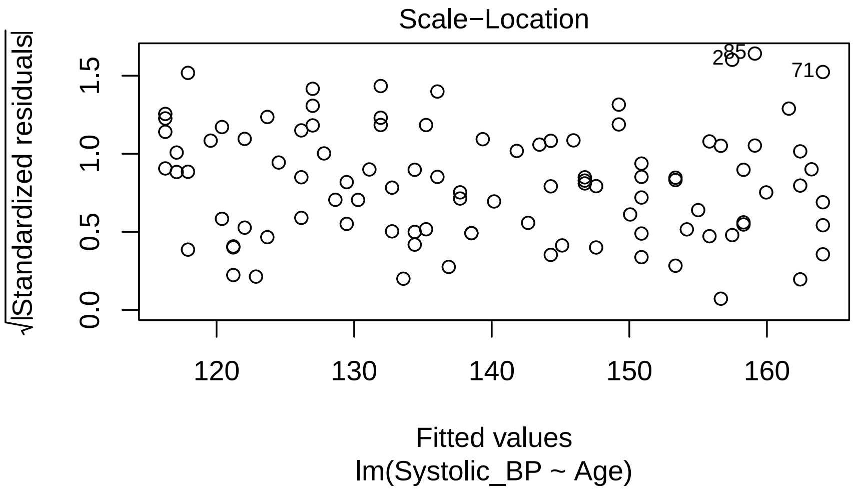
And now visualise the relationship between the fitted values and the size of the residuals:



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

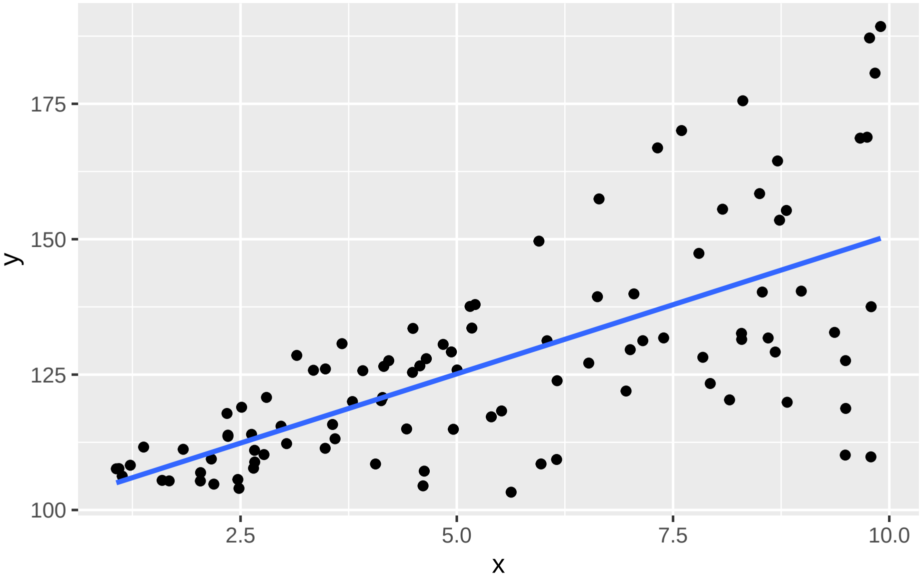
We can also use the built-in plot function for linear models to create this plot. It is the third plot in the set of diagnostic plots.



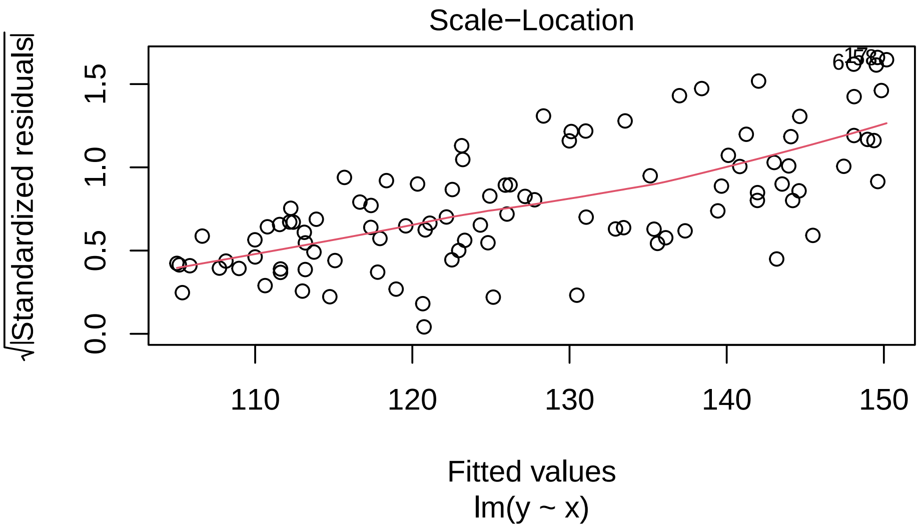
How it looks with the variance increasing with the fitted values

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:



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



(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:

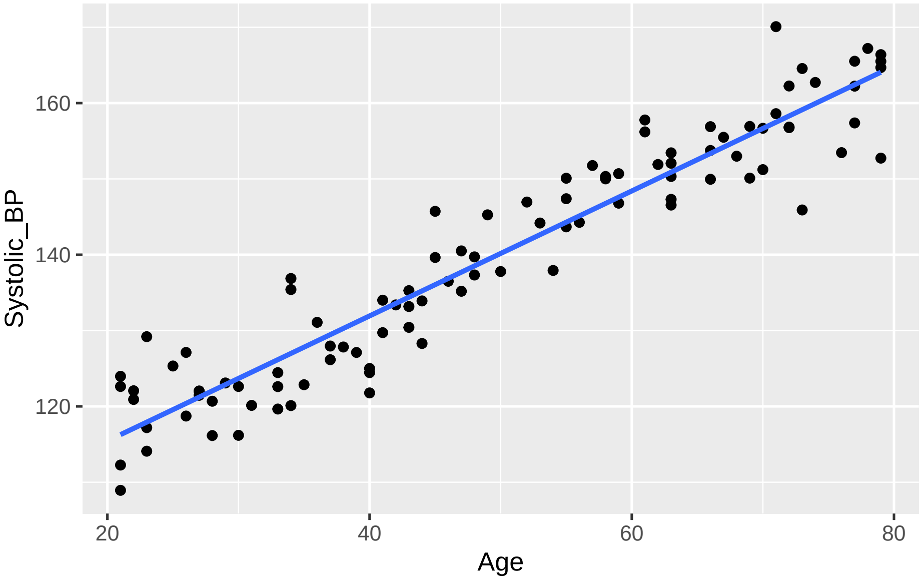
- Think carefully about the study design.
- Think carefully about the unit of observation in the data.
- Compare the number of observations to the number of units of observation.

(d) Linearity assumption

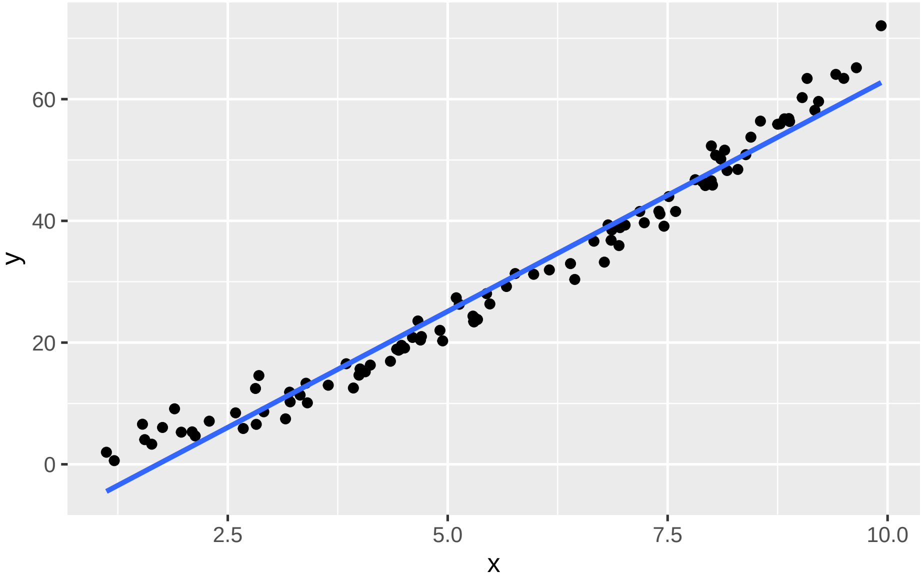
The linearity assumption states that the relationship between the independent variable and the dependent variable is linear. This means that the dependent variable changes by a constant amount for a one-unit change in the independent variable. And that this slope is does not change with the value of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

The blood pressure data seems to be linear:

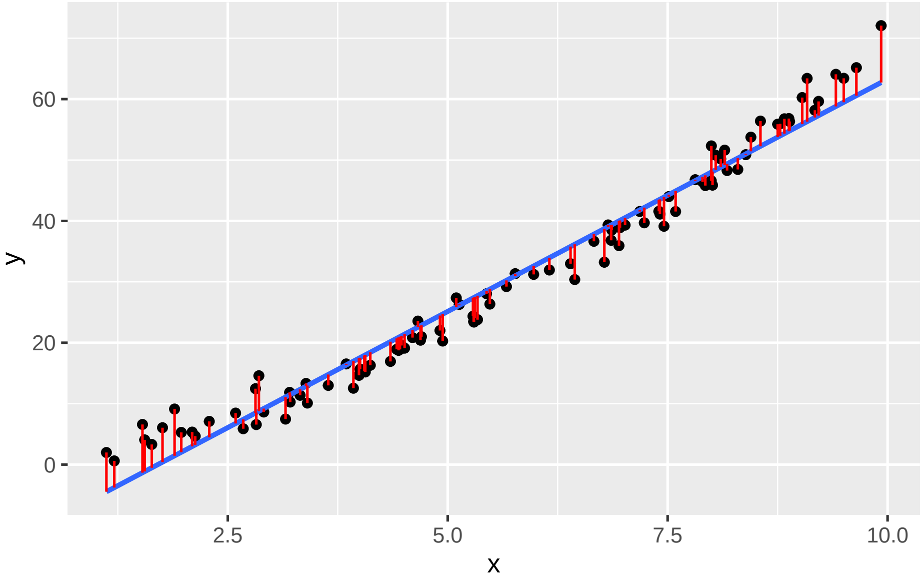
```
Warning: `fortify(<lm>)` was deprecated in ggplot2 3.6.0.
i Please use `broom::augment(<lm>)` instead.
i The deprecated feature was likely used in the ggplot2 package.
  Please report the issue at <https://github.com/tidyverse/ggplot2/issues>.
```



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



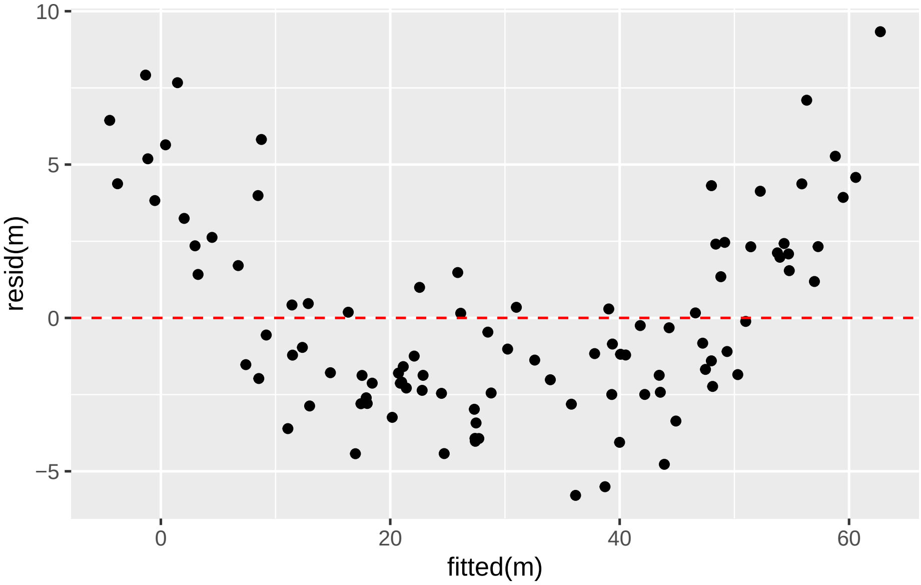
And with the residuals shown as red lines:



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.

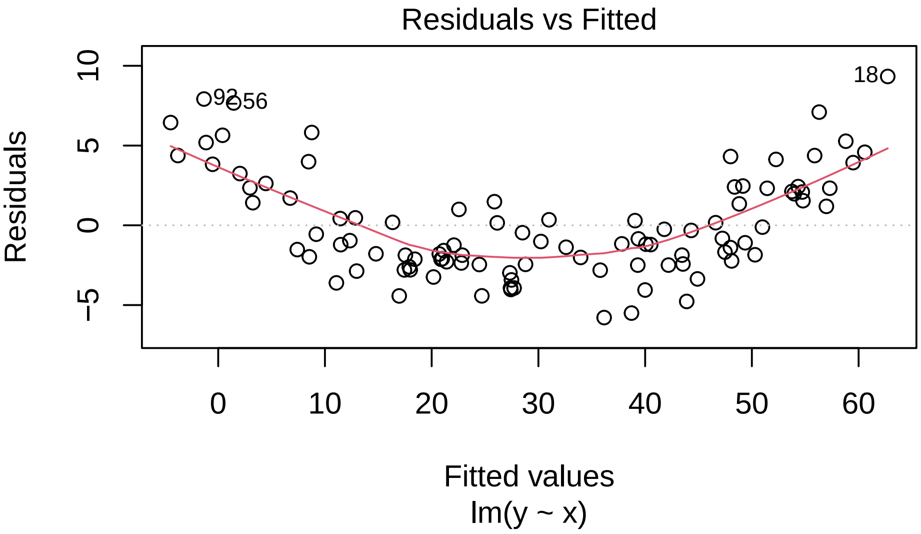
We can plot the value of the residuals against the y value directly, instead of looking at the pattern in the graph above. This is called a **Tukey-Anscombe**

plot. It is a graph of the residuals versus the fitted y values:

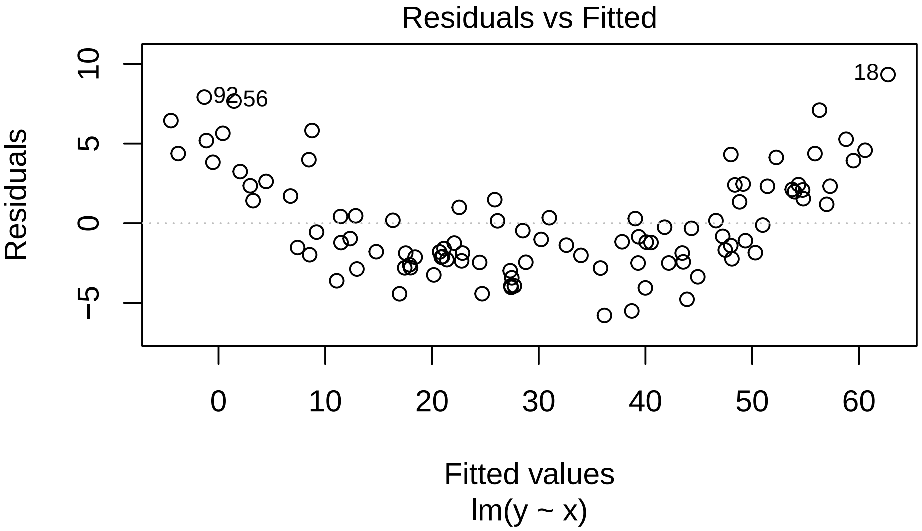


We can very clearly see pattern in the residuals in this Tukey-Anscombe plot. The residuals are positive, then negative, then positive, as the fitted y value gets larger.

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

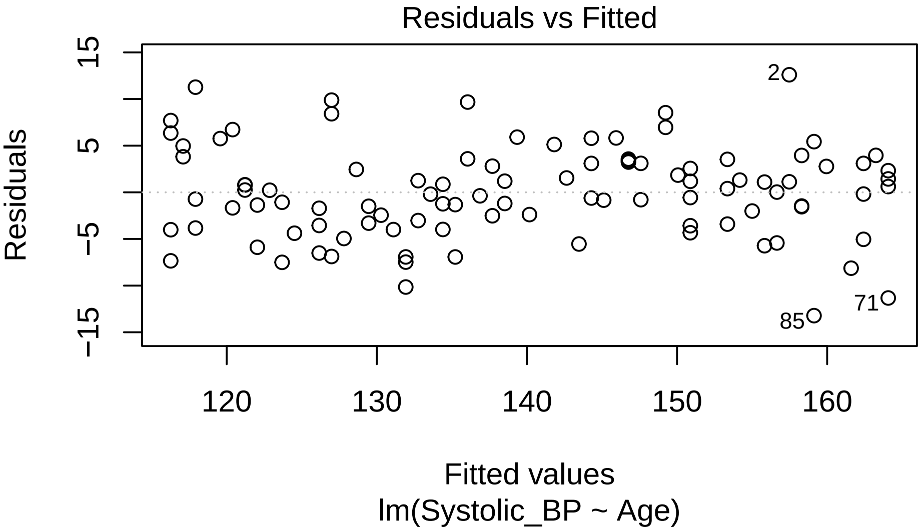


The red line in the Tukey-Anscombe plot is a loess smooth. It is automatically added to the plot. It is a way of estimating the pattern in the residuals. If the red line is not flat, then there is a pattern in the residuals. However, the loess smooth is not always reliable. It is a good idea to look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



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:

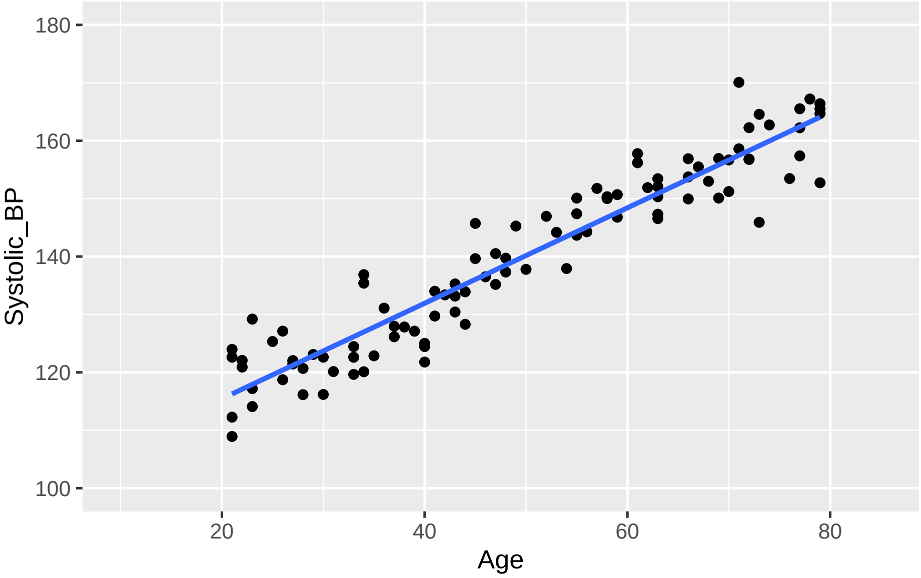


There is very little evidence of any pattern in the residuals. This data is simulated with a truly linear relationship, so we would not expect to see any pattern in the residuals.

(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:



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 (#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.

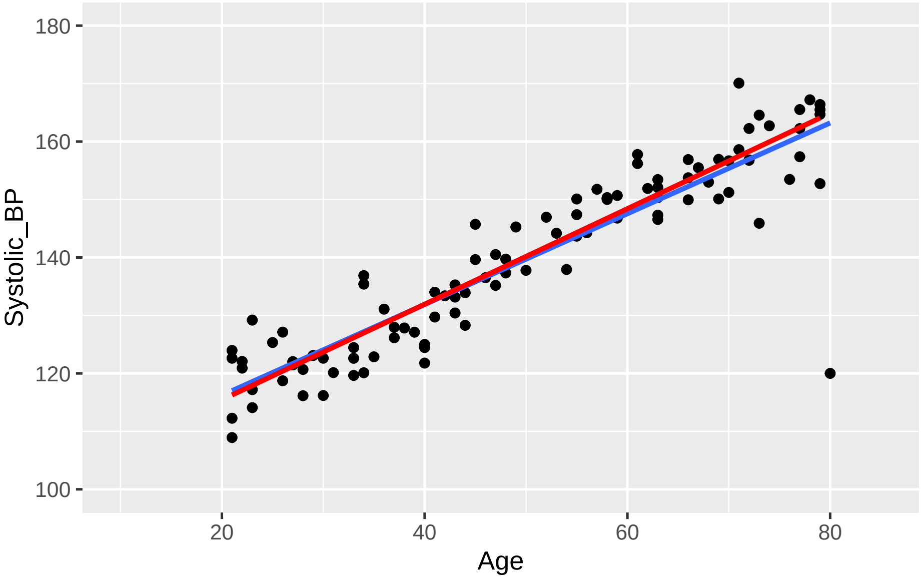
Think, Pair, Share (#like-weight)

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.

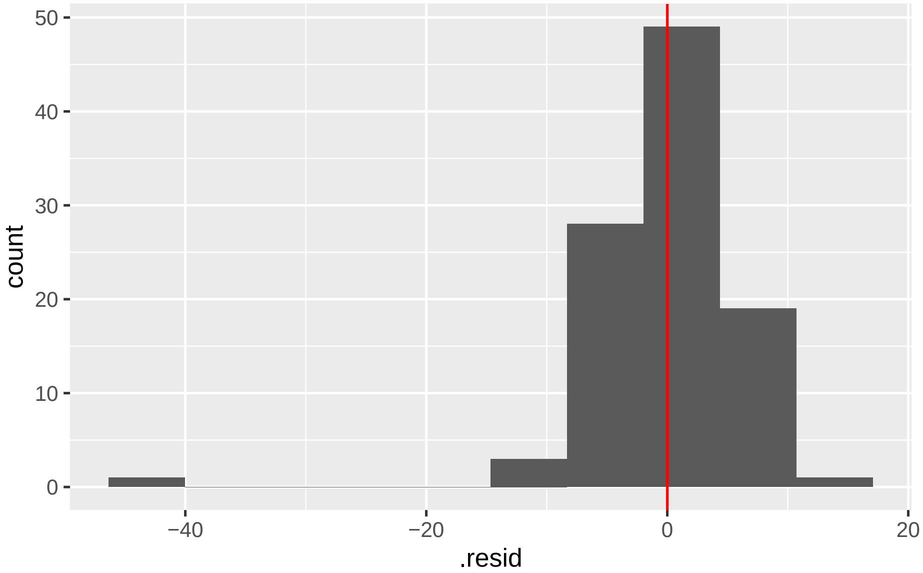
So the overall effect of a data point on the slope of the line of best fit is a combination of the leverage and the residual. This quantity is called the *influence* of a data point.

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:

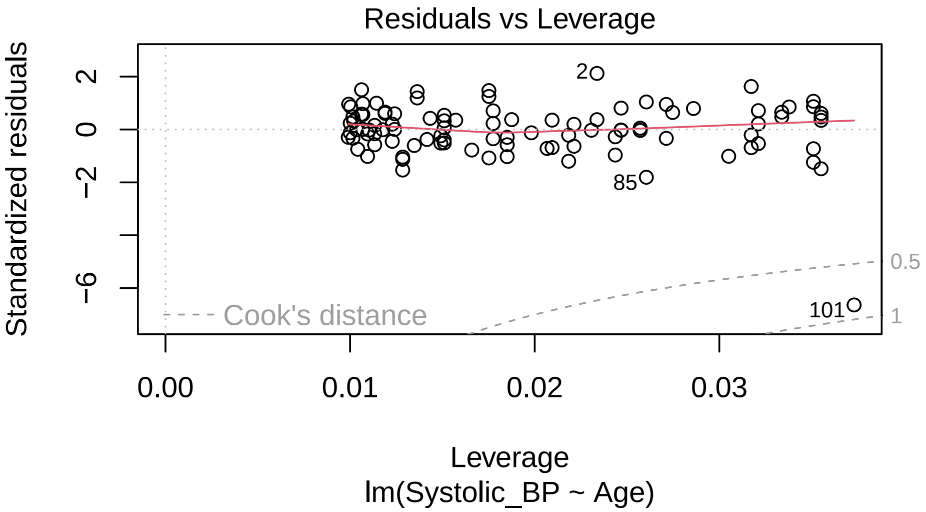


And we can see that the leverage is large.

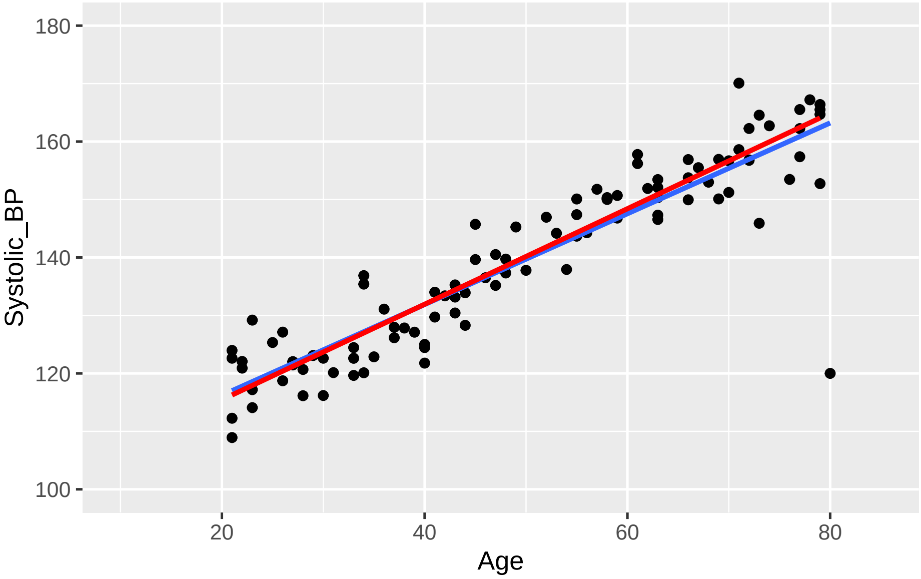
There is a graph that we can look at to see the influence of a data point. This is called a *Cook′s distance* plot. The Cook's distance of a data point is a measure of how much the slope of the line of best fit changes when that data point is removed. The Cook's distance of a data point is defined as

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

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.



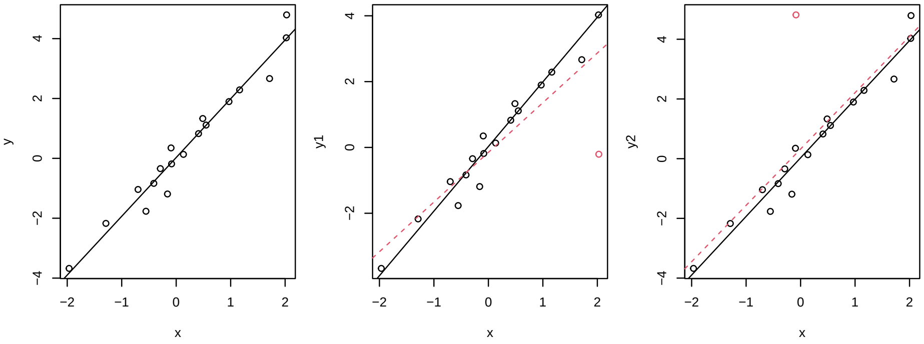
But does it have a large influence on the value of the slope? In the next graph we show the line of best fit with the outlier (blue line) and without the outlier (red line).



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

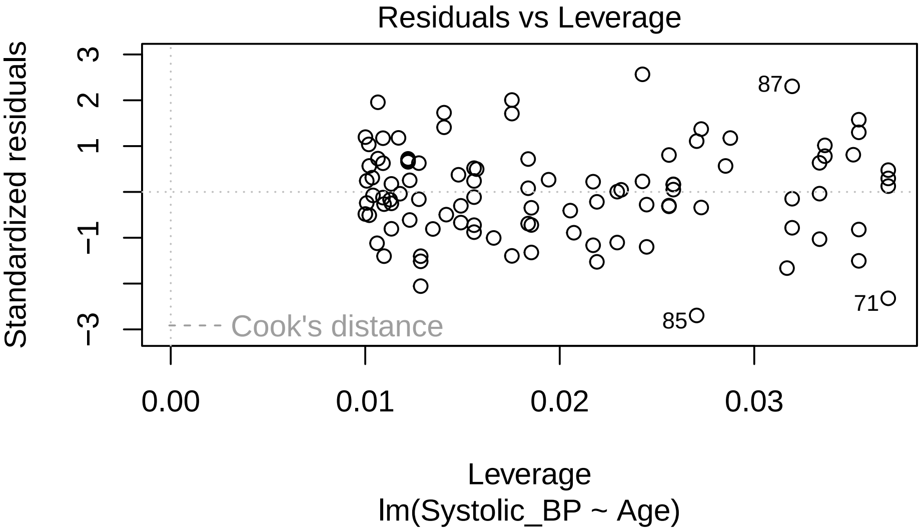
Data points with x_i values far from the mean have a stronger leverage effect than when $x_i \approx \overline{x}$:



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} :



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.

What can go "wrong" during the modeling process?

Answer: a lot of things!

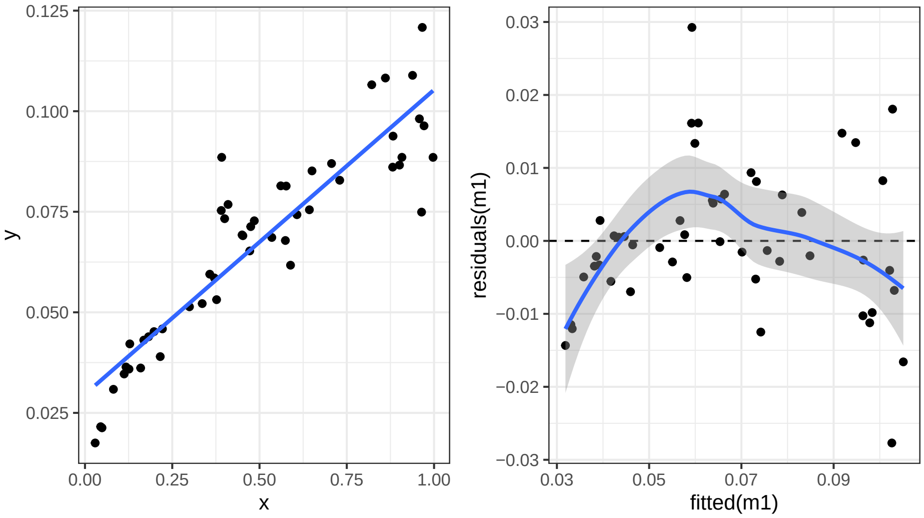
- Non-linearity. We assumed a linear relationship between the response and the explanatory variables. But this is not always the case in practice. We might find that the relationship is curved and not well représtented by a straight line.
- Non-normal distribution of residuals. The QQ-plot data might 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).

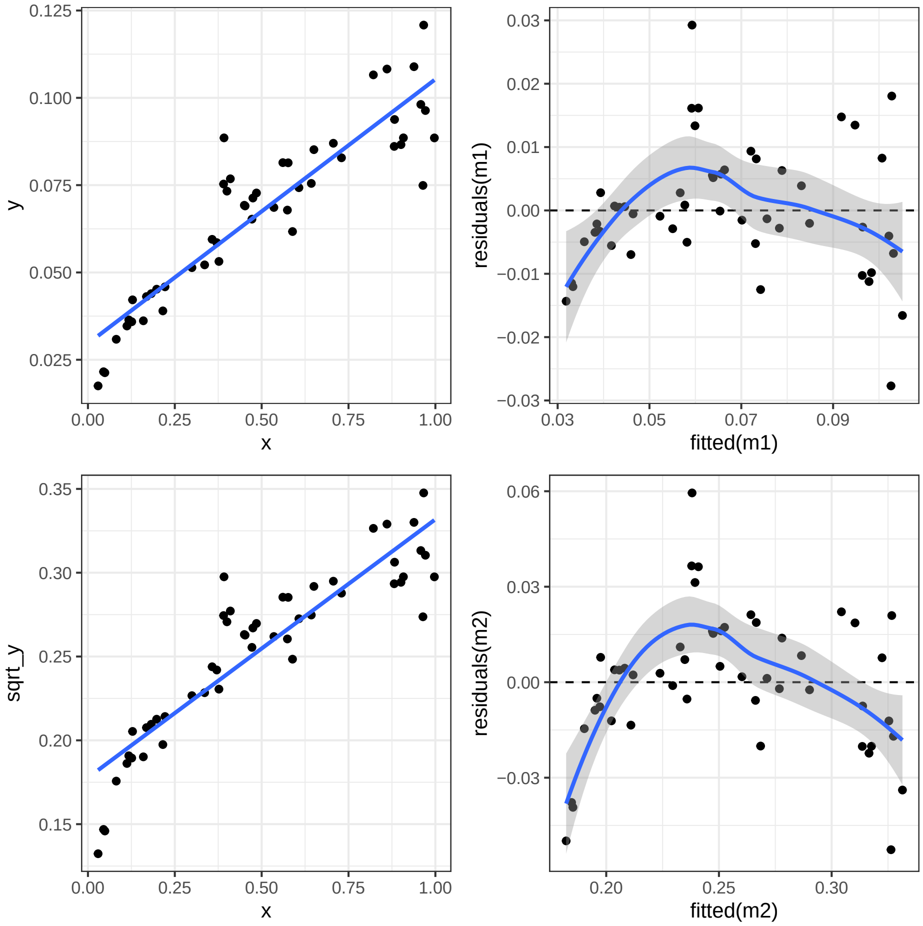
Dealing with non-linearity

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



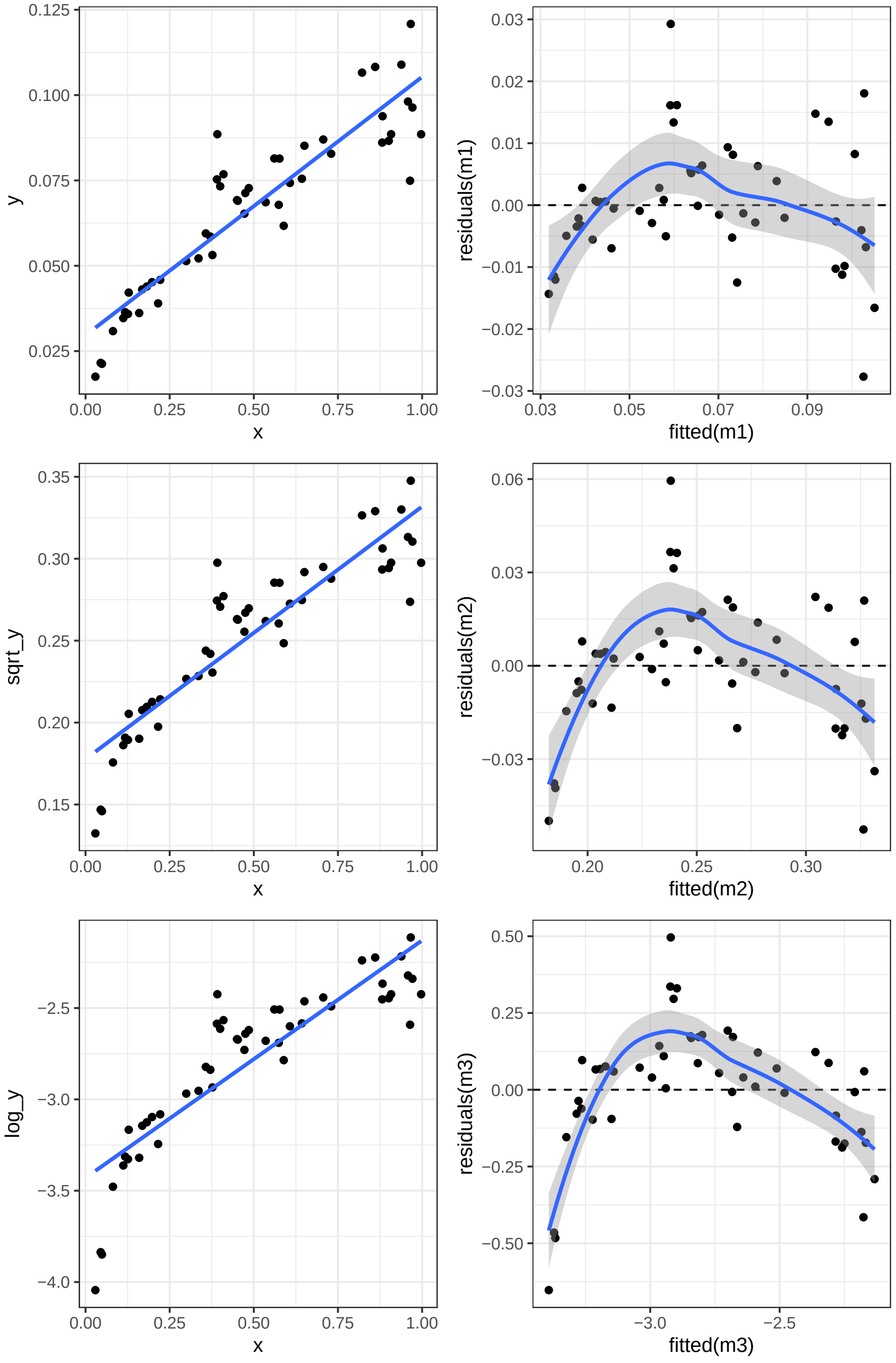
One way to deal with this is to transform the response variable Y. Here we try two different transformations: $\log_{10}(Y)$ and $\sqrt{Y}$.

Square root transform of the response variable Y:



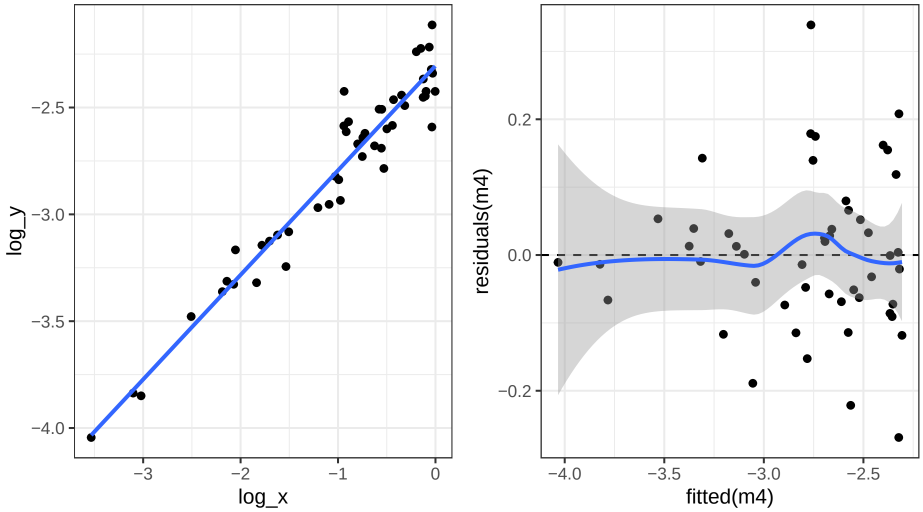
Not great.

Log transformation of the response variable Y:

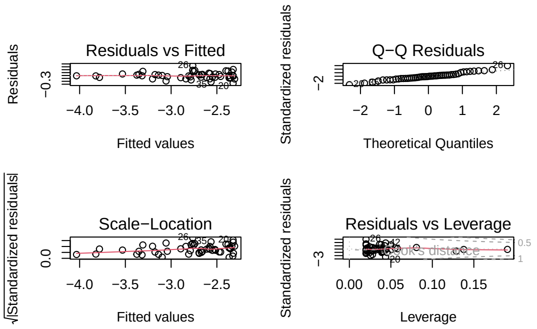


Nope. Still some evidence of non-linearity.

What about transforming the explanatory variable X as well?



Let’s look at the four diagnostic 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 diagnostic 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, $\beta_1, \beta_2, \dots, \beta_p$ are the coefficients of the independent variables, $x_1, x_2, \dots, x_p$ 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.

Regression Part 2

- r-squared
- Multiple r-squared
- Adjusted r-squared
- Conditional r-squared
- Marginal r-squared

How Owen will structure the lecture time

The chapter content below is the reference for what students are expected to know. During the lecture time Owen will talk through and explain hopefully most of the content of this chapter. He will write on a tablet, show figures and other content of this chapter, and may live-code in RStudio. He will also present questions and ask students to *Think, Pair, Share.* The *Share* part will sometime be via clicker, sometimes by telling the class. The same will happen in lecture 3-6.

Introduction

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

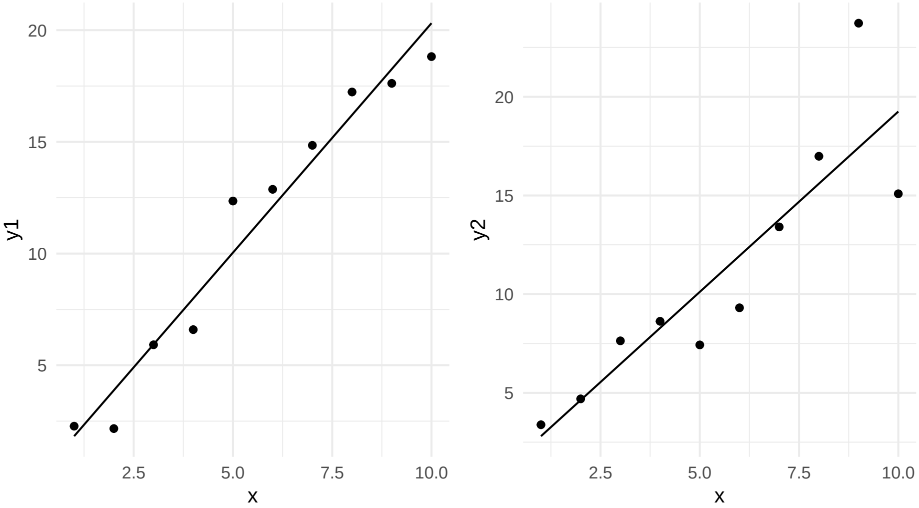
- How to measure how good is the regression (correlation and R^2).
- How to test if the parameter estimates are compatible with some specific value (t-test).
- How to find the range of parameters values are compatible with the data (confidence intervals).
- How to find the regression lines compatible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

Accompanying reading material

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 (#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

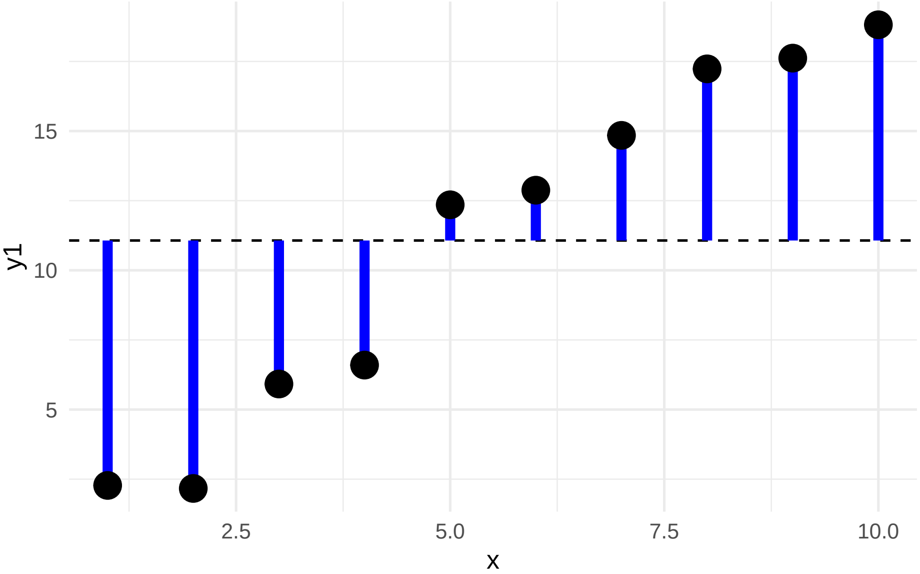
1. The total variability of the dependent variable is the sum of the squared differences between the dependent variable and its mean. This is called the total sum of squares (SST).

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

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.



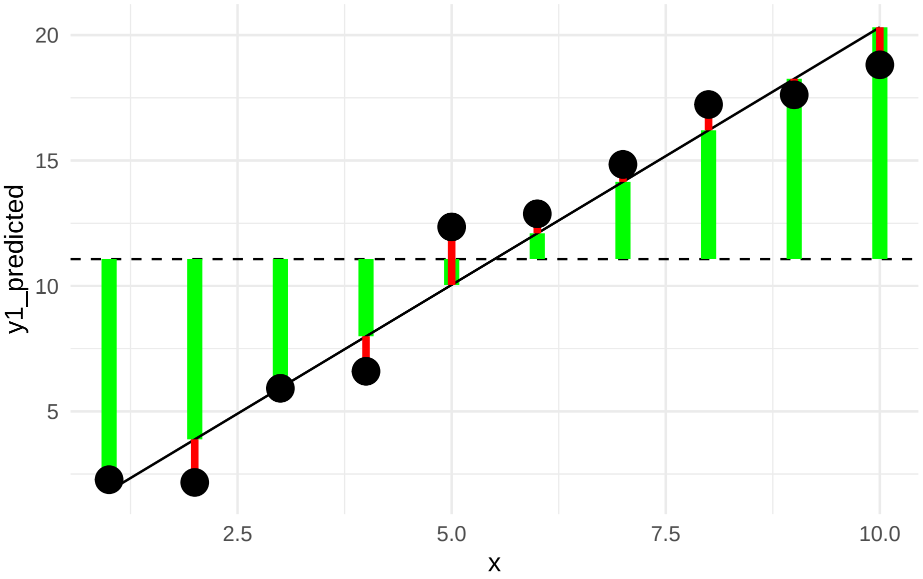
We can calculate this in R as follows:

```
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:

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

```
[1] 348.0177
```

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:

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

```
[1] 15.20883
```

Or…

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

```
[1] 15.20883
```

R^2

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

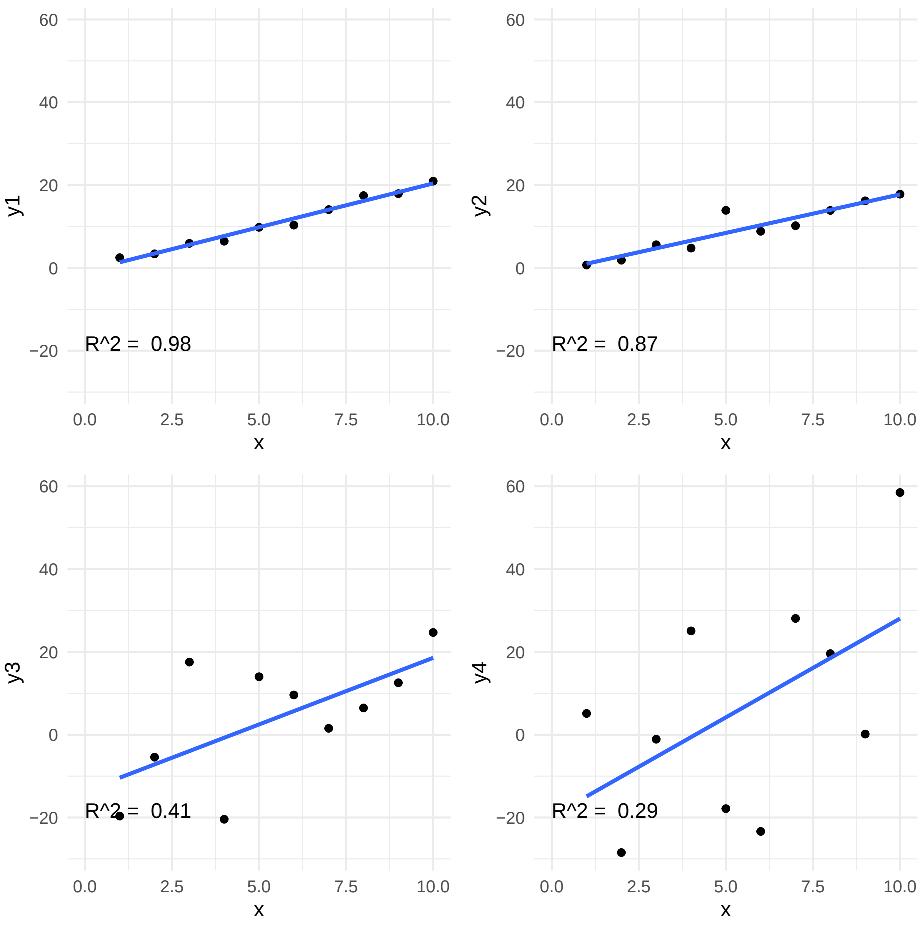
$$R^2 = \frac{SSM}{SST}$$

```
[1] 0.9581285
```

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.



Questions

> Think, Pair, Share (#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 (#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:

```
              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:

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

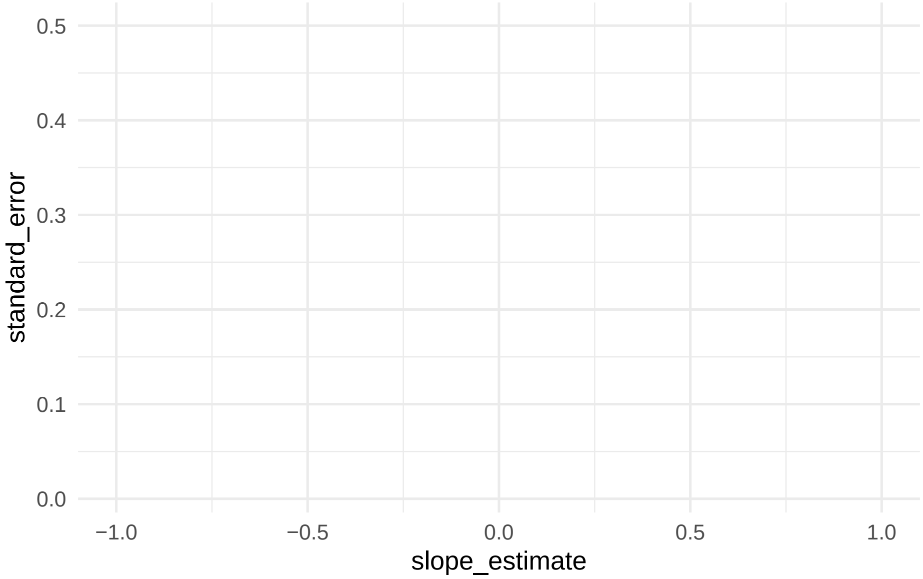
Where $\widehat{\sigma}^2$ is the expected residual variance of the model. This is calculated as:

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

Where $\widehat{y}_i$ 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 (#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 (Section **?@sec-visual-p-values-regress**).

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:

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

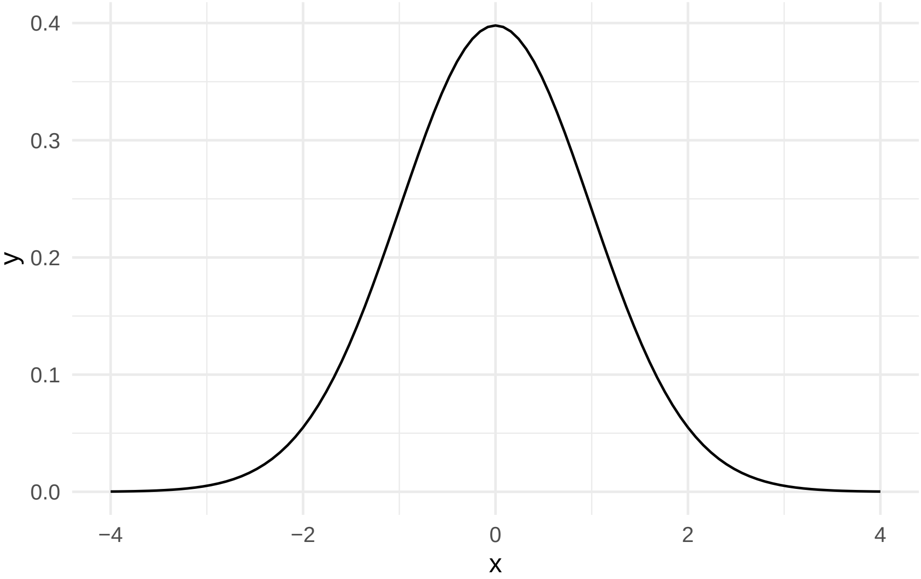
Where β_{1,H_0} is the null hypothesis value of the slope, usually zero, so that

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

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.

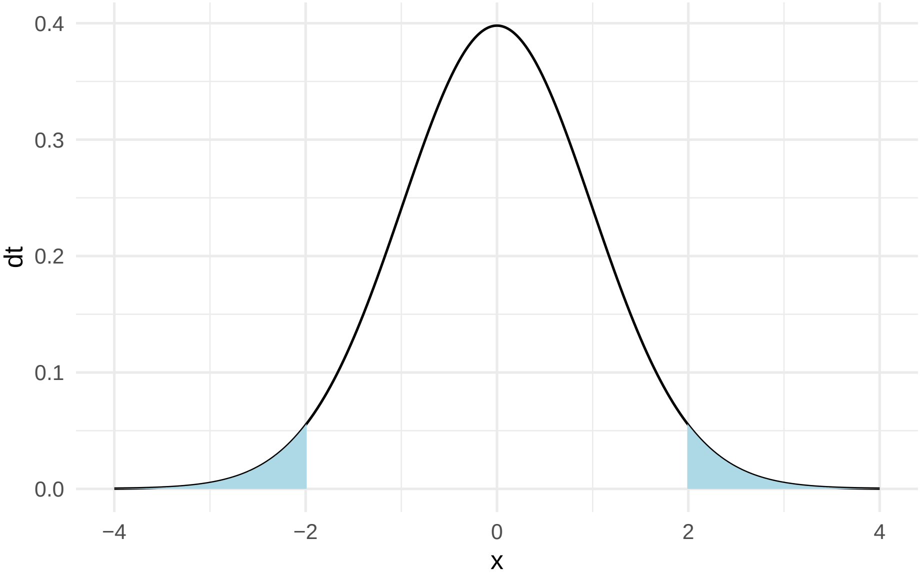


Tip

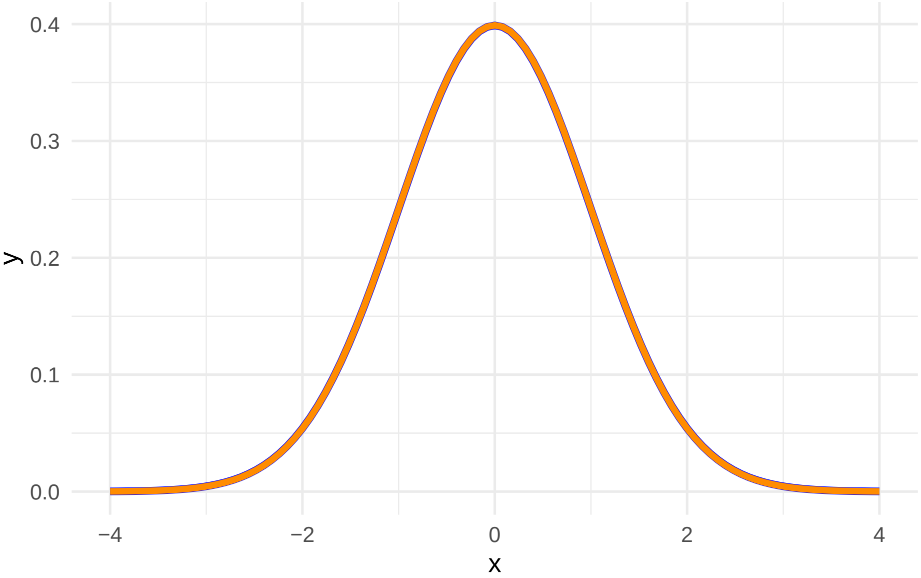
By the way, it is named the t-distribution by it's developer, William Sealy Gosset, who worked for the Guinness brewery in Dublin, Ireland. In his 1908 paper, Gosset introduced the t-distribution but he didn't explicitly explain his choice of the letter t. The choice of the letter t could be to indicate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



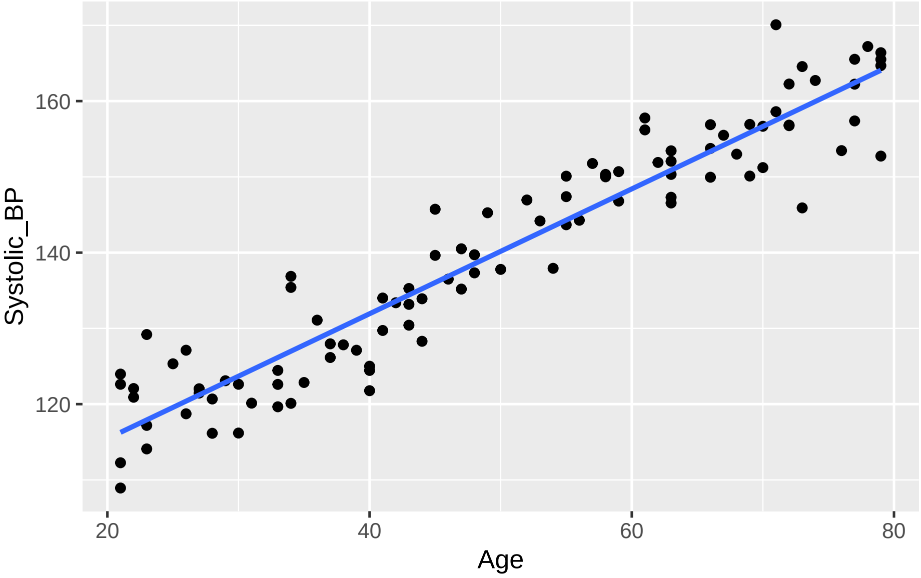
So, with a large number of observations, the *t*-distribution approaches the normal distribution. For the normal distribution, the 95% area is between -1.96 and 1.96.

```
## 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.



Here's the model:

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

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

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

```
         Age
3.746958e-51
```

```
         Age
7.493917e-51
```

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

```
    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?”

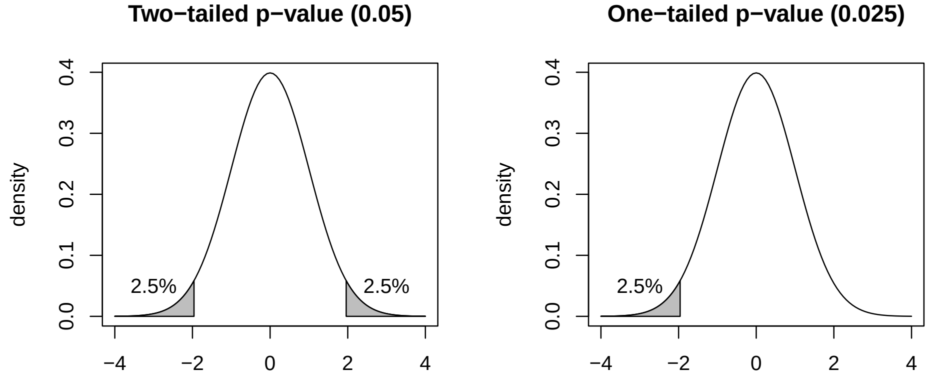
Recap: Formal definition of the p-value

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.

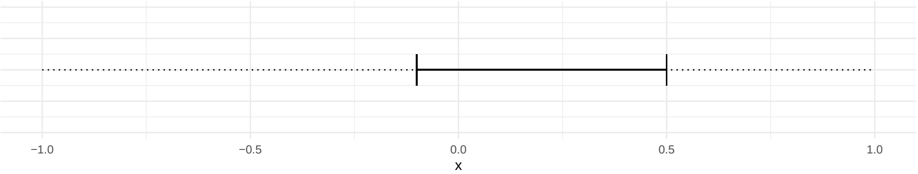
We can ask what values of a parameter estimate are compatible with the data (confidence intervals)? To answer this question, we can determine the confidence intervals of the regression parameters.

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:

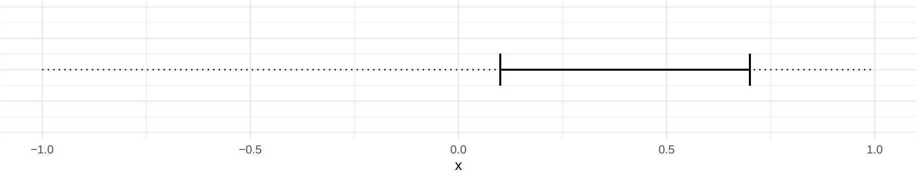
```
Warning: `geom_errobarh()` was deprecated in ggplot2 4.0.0.
i Please use the `orientation` argument of `geom_errorbar()` instead.
```

```
`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:

```
    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%)?

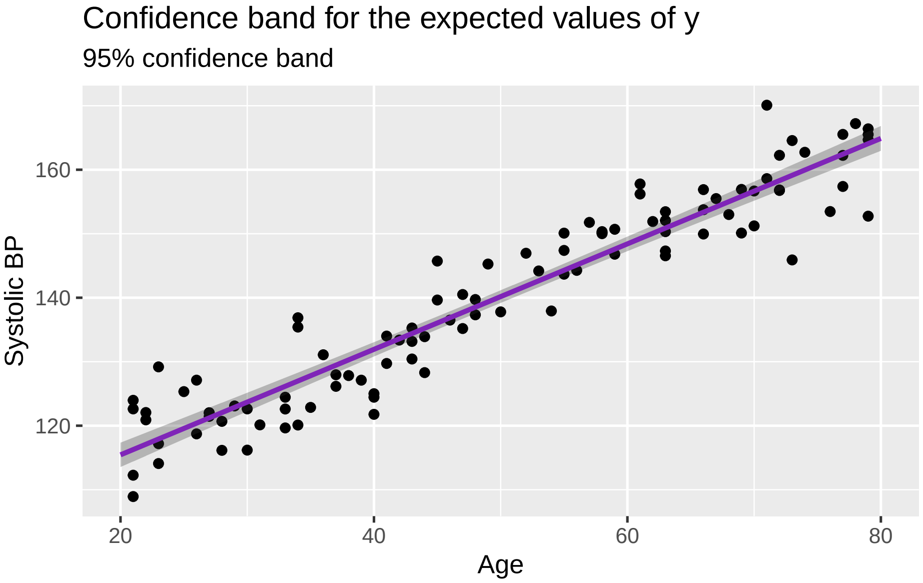
This question is not trivial, because both $\hat{\beta}_0$ and $\hat{\beta}_1$ are estimates from the data and contain uncertainty.

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.

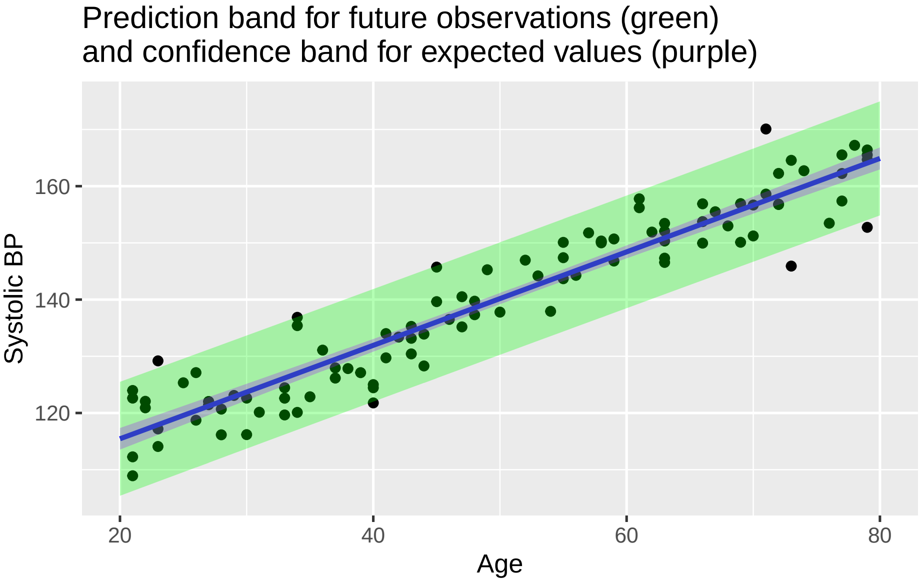
Calculations of the prediction band

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.

That is regression done (at least for our current purposes)

- 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.

- Goodness of the model: Correlation and R^2
- Tests and confidence intervals
- Confidence and prediction bands

Additional reading material

If you'd like another perspective and a deeper delve into some of the mathematical details, please look at Chapter 2 of *Lineare Regression*, p.7-20 (Stahel script), Chapters 3.1, 3.2a-q of *Lineare Regression*, and Chapters 4.1 4.2f, 4.3a-e of *Lineare Regression*

Extras

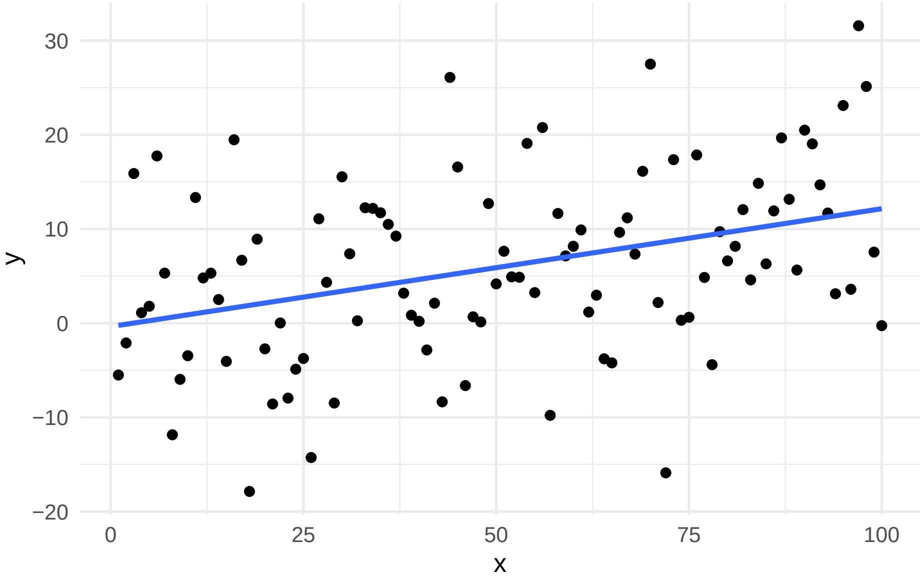
Randomisation test for the slope of a regression line

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.

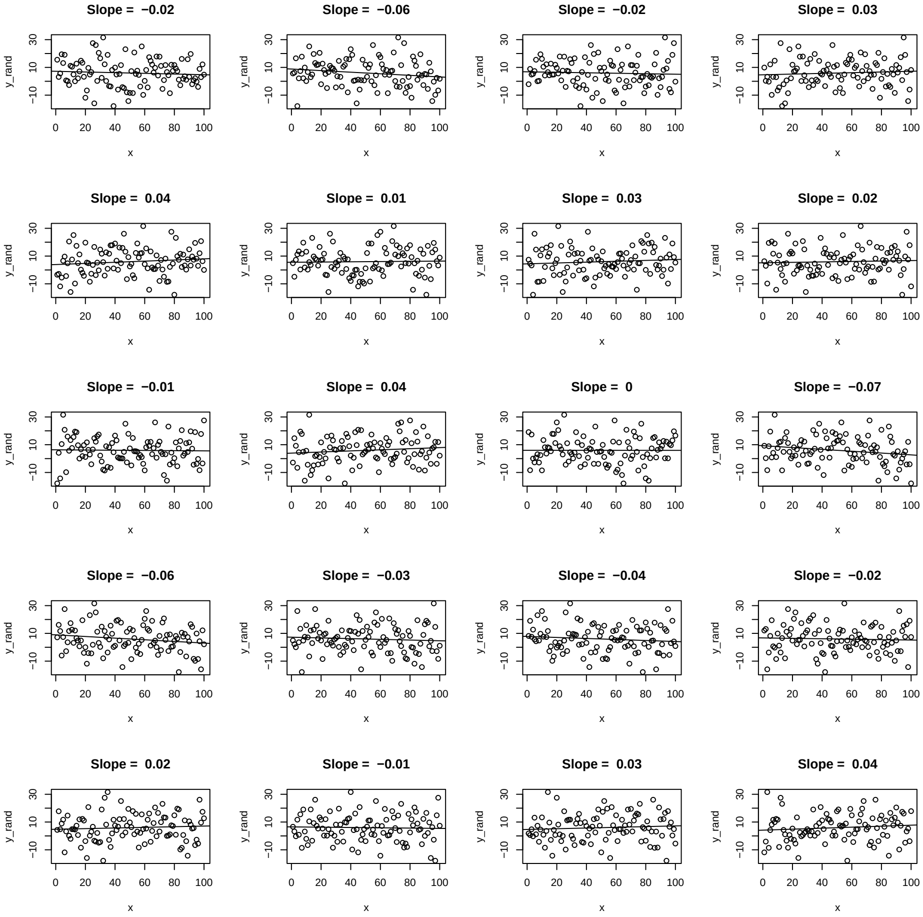
If the null hypothesis is true, we expect no relationship between x and y. Therefore, we can shuffle the y values and fit a regression model to the shuffled data. We can repeat this many times and calculate the slope of the regression line each time. This will give us a distribution of slopes we would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

```
        x
0.1251108
```

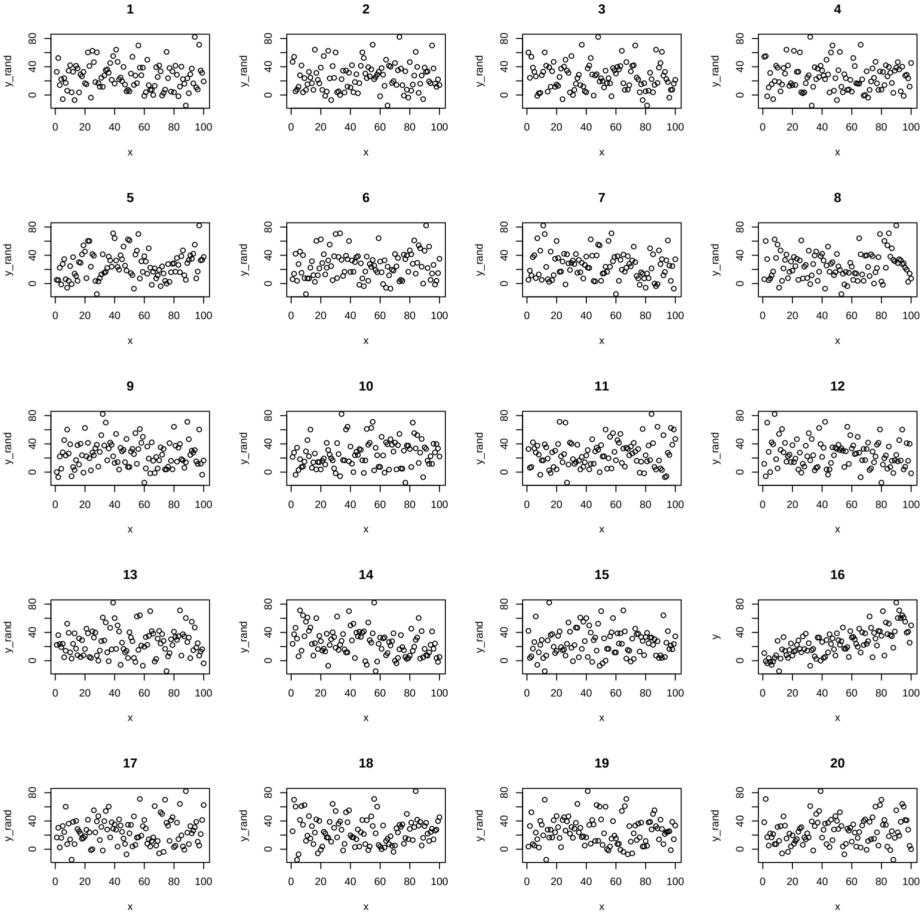


Now we'll use randomisation to test the null hypothesis. We can create lots of examples where the relationship is expected to have a slope of zero by shuffling randomly the y values. Here are 20:

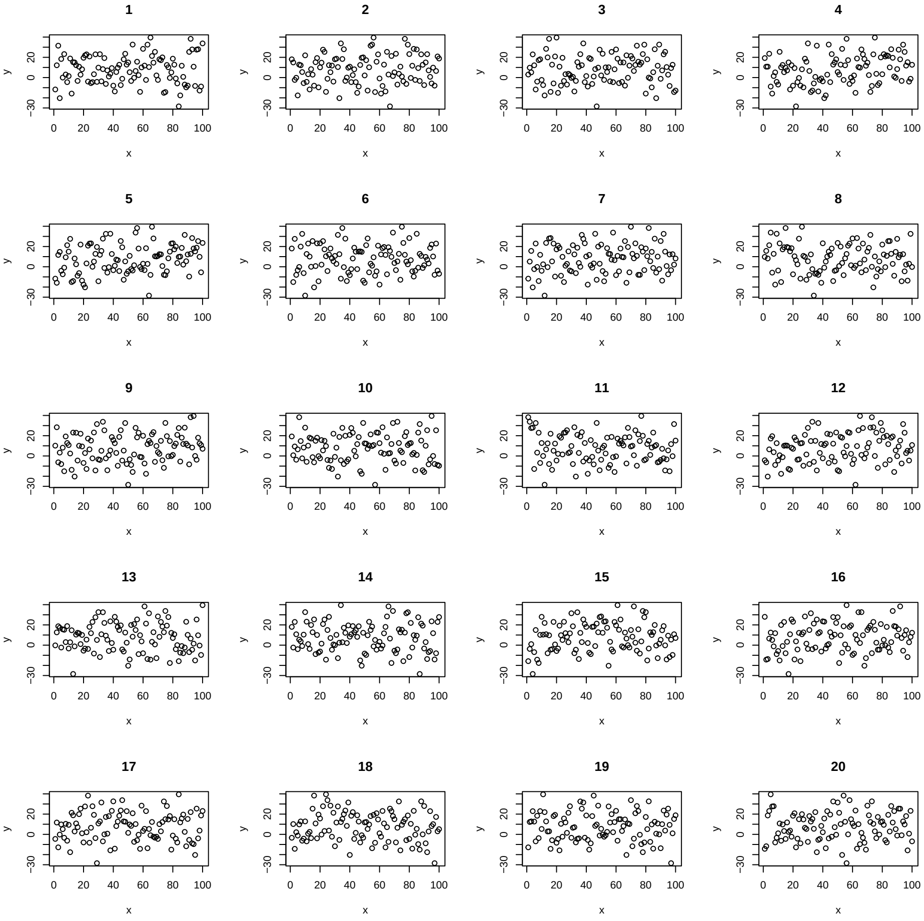


Slope = −0.02
Slope = −0.06
Slope = −0.02
Slope = 0.03
Slope = 0.04
Slope = 0.01
Slope = 0.03
Slope = 0.02
Slope = −0.01
Slope = 0.04
Slope = 0
Slope = −0.07
Slope = −0.06
Slope = −0.03
Slope = −0.04
Slope = −0.02
Slope = 0.02
Slope = −0.01
Slope = 0.03
Slope = 0.04

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:

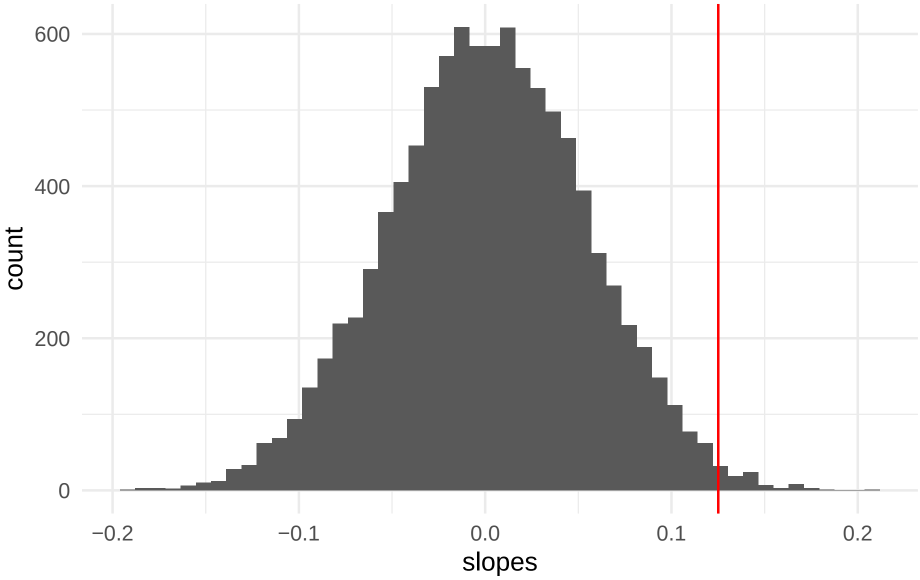


We can confidently find the real data amount the shuffled data. But what if the relationship is weaker?



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

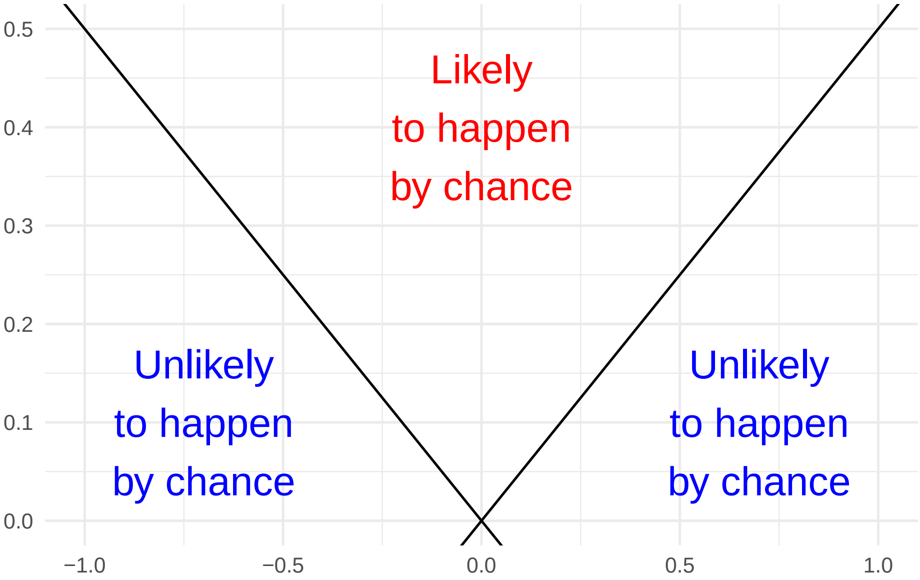
We do the same procedure of but instead of just looking at the graphs, we calculate the slope of the regression line each time. This gives us a distribution of slopes we would expect to observe if the null hypothesis is true. We can then see where the observed slope lies in this distribution of null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



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

```
[1] 0.0172
```

Visualising p-values for regression slopes



ANOVA (L6)

- One-way ANOVA
- Post-hoc tests and contrasts
- Two-way ANOVA

ANOVA = ANalysis Of VAriance (Varianzanalyse)

Introduction

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).

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:

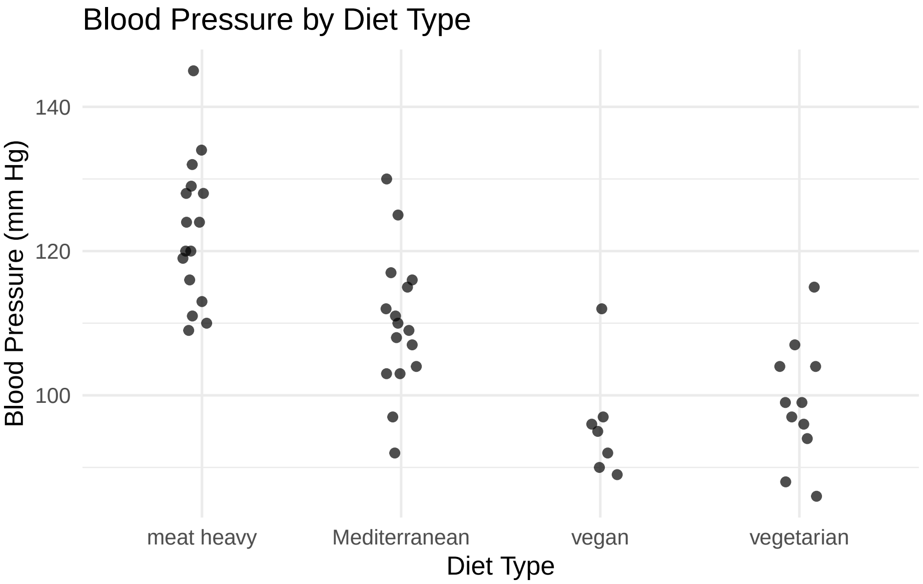
```
bp_data_diet <- select(bp_data_diet, bp, diet, person_ID)
bp_data_diet
```

```
# A tibble: 50 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 
 7    99 vegetarian    person_7 
 8   104 vegetarian    person_8 
 9   110 Mediterranean person_9 
10    97 Mediterranean person_10
# i 40 more rows
```

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.



> Tip
>
> We just used `geom_jitter()` instead of `geom_point()` to make a scatterplot. 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.

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

```
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:

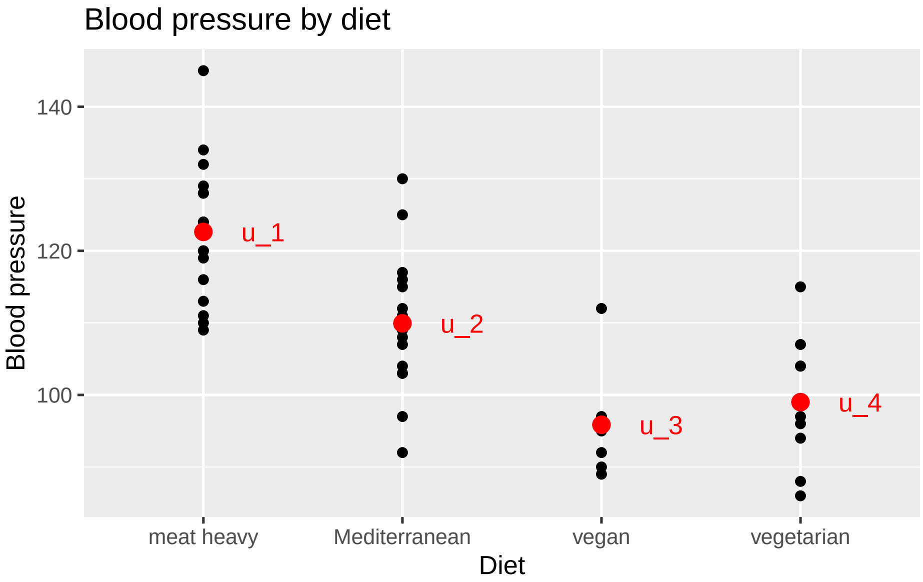
$$y_{ij} = \mu_j + \epsilon_i$$

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:

```
`summarise()` has grouped output by 'diet'. You can override using the
`.groups` argument.
```



::: {.callout-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:

$$\mu_{Med} = \beta_0 + \beta_1$$
$$\mu_{vegan} = \beta_0 + \beta_2$$
$$\mu_{veggi} = \beta_0 + \beta_3$$

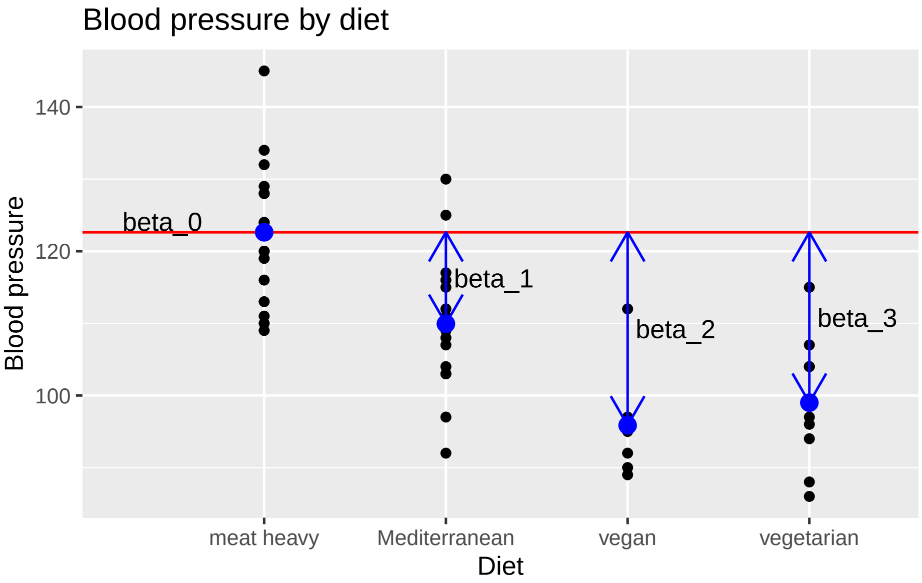
When we write out the entire model, we get:

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

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:

```
`summarise()` has grouped output by 'diet'. You can override using the
`.groups` argument.
```



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).

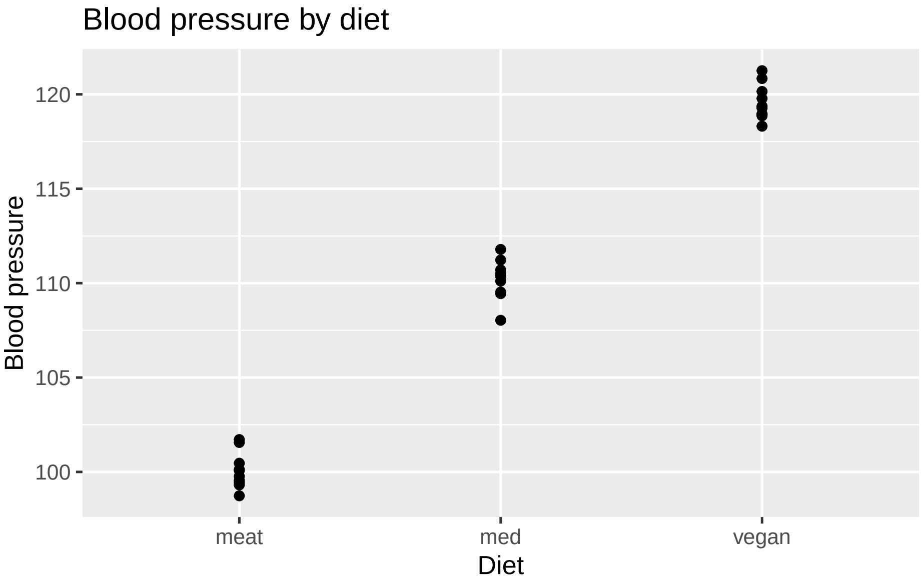
The F-test does this by comparing two variance estimates: the variance explained by the group means (between-group variance) and the variance that remains unexplained within each group (within-group variance).

Interpretation of the F statistic

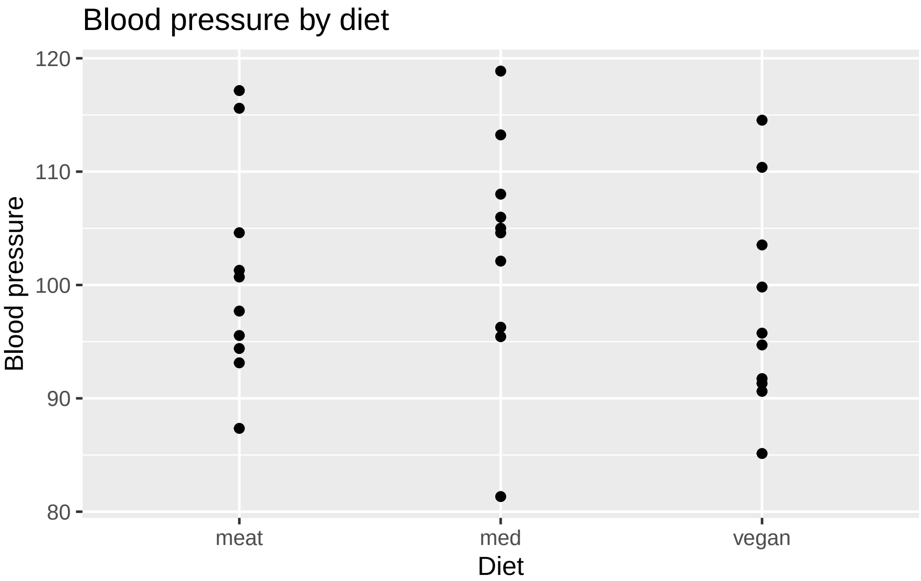
The F-test involves calculating from the observed data the value of the F statistic, and then computing if that value is large enough to reject the null hypothesis.

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.

→ 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: 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): == 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:

$$MS_{model} = \frac{SS_{\text{between}}}{k-1} = \frac{SSM}{k-1}$$

$$MS_{residual} = \frac{SS_{\text{within}}}{n-k} = \frac{SSE}{n-k}$$

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

ℹ 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	$n-1$		

Back to the F-test

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.

$$H_0 : \beta_1 = \beta_2 = ... = \beta_p = 0$$

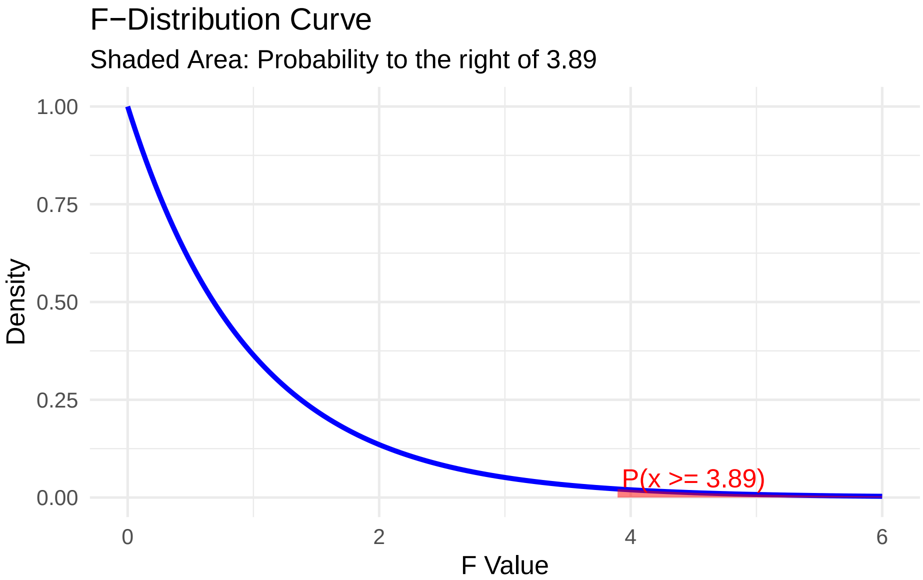
The alternative hypothesis is that the explained variance of the model is greater than would be expected by chance. This would occur if the slopes of some or all of the explanatory variables are not zero.

$$H_1 : \beta_1 \neq 0 \text{ or } \beta_2 \neq 0 \text{ or } ... \text{ or } \beta_p \neq 0$$

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

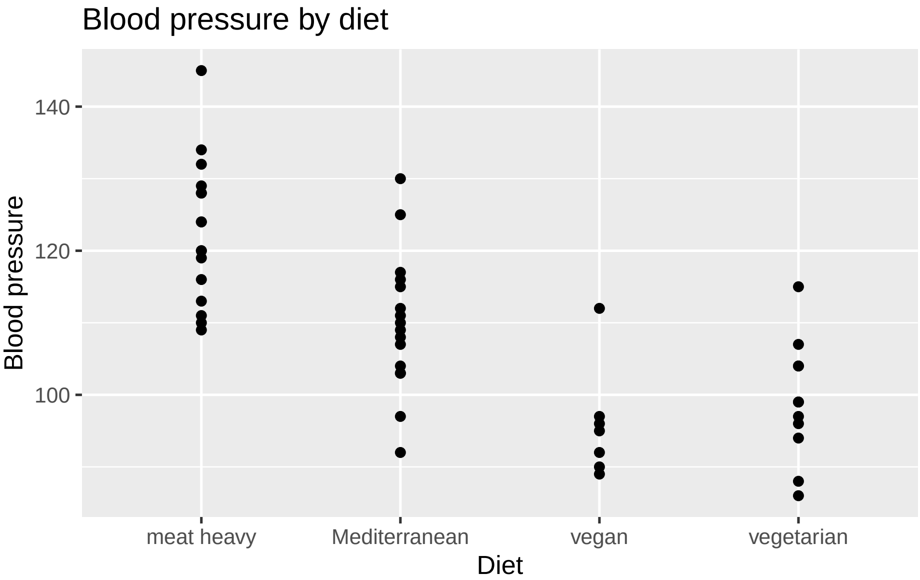
Well, that is ANOVA conceptually. But how does it actually look like in R?

Doing ANOVA in R

Let's go back again the question of how diet effects blood pressure. Here is the data:

```
# A tibble: 6 x 3
```

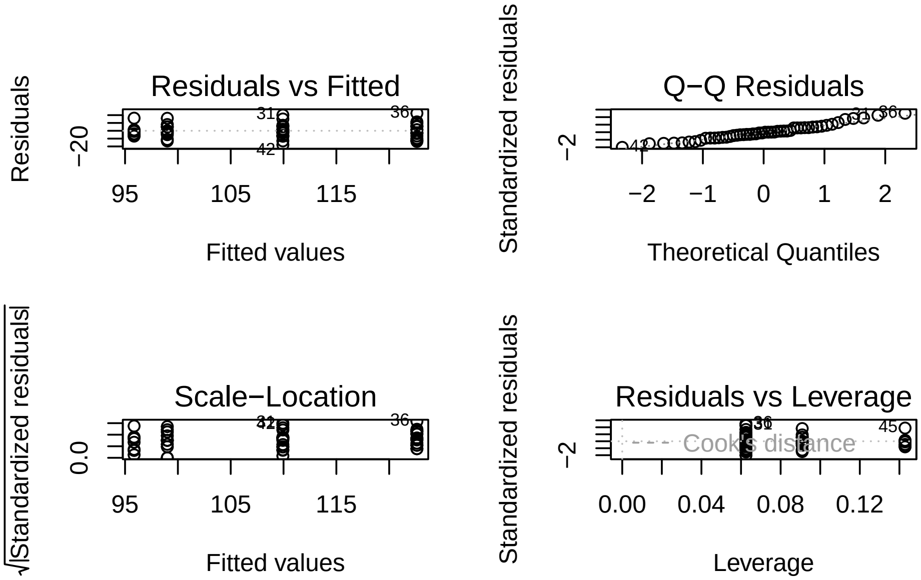
```
     bp dict          person_ID
  <dbl> <chr>         <chr>
1   120 meat heavy    person_1
2    89 vegan         person_2
3    86 vegetarian    person_3
4   116 meat heavy    person_4
5   115 Mediterranean person_5
6   134 meat heavy    person_6
```



And here is how we fit a linear model to this data:

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 diagnostic plots:



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:

```
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

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:

```

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:

```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)
library(multcomp)
```

```
Loading required package: mvtnorm

Loading required package: survival

Loading required package: TH.data

Loading required package: MASS


Attaching package: 'MASS'

The following object is masked from 'package:patchwork':

    area

The following object is masked from 'package:dplyr':

    select
```

```
Attaching package: 'TH.data'

The following object is masked from 'package:MASS':

    geyser
```

```
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?

```
# 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 diagnostics 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:

And now make the model and look at the estimated coefficients:

```

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.

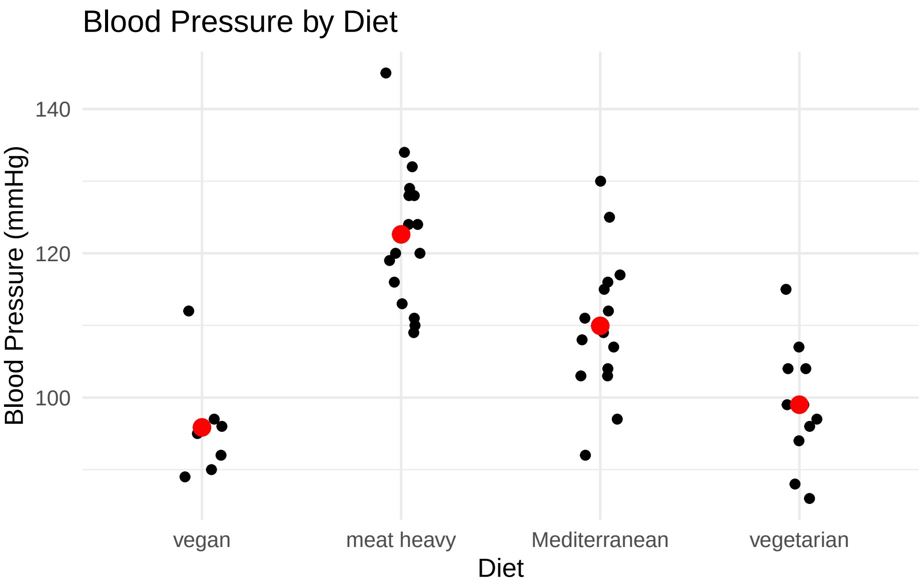
Communicating the results of ANOVA

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 pressure of the vegan group was 27 mmHg lower (One-way ANOVA, $F(3, 46) = 20.7$, $p < 0.0001$.

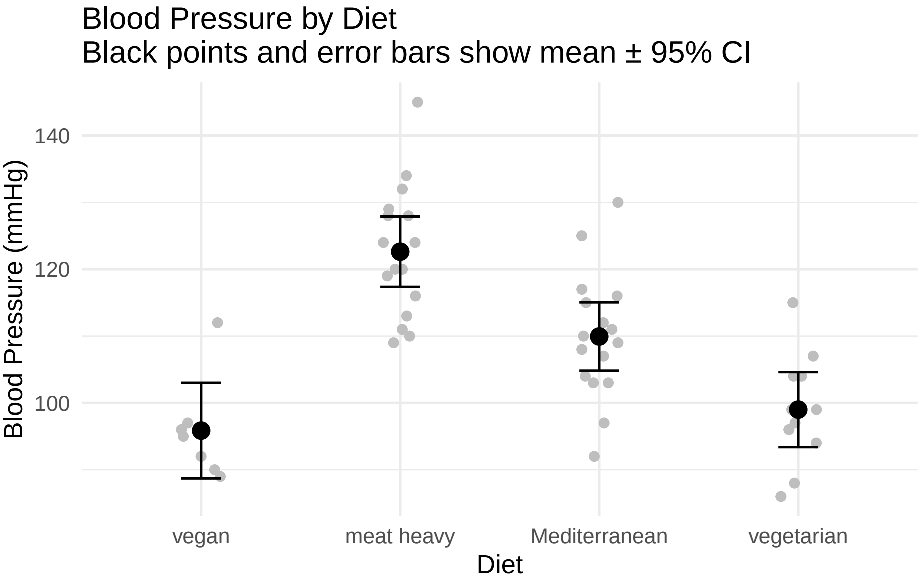
And we would make a nice graph, in this case showing each individual observation since there are not too many to cause overplotting. We can also add the estimated means of each group if we like:

```r
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)") +
  theme_minimal()
```



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)") +
  theme_minimal()
```



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.

Summary of what you have learned

Please referring to the learning objectives document and the learning objectives for this chapter.

end

end

Two-way ANOVA (Zweiweg-Varianzanalyse)

Two-way ANOVA is used to analyse a specific type of study design. When we have a study with two categorical treatments and all possible combinations of them, we can use a two-way ANOVA.

For example, take the question of how diet and exercise affect blood pressure. Let's say we can have three levels of diet: meat heavy, Mediterranean, and vegetarian. And that we have two levels of exercise: low and high. And that we have all possible combinations of these two treatments: i.e., we have a total of $3 \times 2 = 6$ **treatment combinations**.

We can also represent this study design in a table:

Exercise (G)

Diet (B)

Low (1)

High (2)

Meat heavy (1)

Mediterranean (2)

Vegetarian (3)

The six empty cells in the table represent the six treatment combinations.

This type of study, with all possible combinations, is known as a *factorial design*. The two treatments are called factors, and the levels of the factors are called factor levels. A *fully factorial design* is one where all possible combinations of the factor levels are present.

Let's look at example data:

```
# A tibble: 6 x 5
  diet          exercise  reps    bp  error
  <chr>         <fct>    <int> <dbl>  <dbl>
1 Mediterranean high         1  110.  4.59
2 Mediterranean low          1  119.  6.79
3 meat heavy    high         1  102. -3.13
4 meat heavy    low          1  128.  7.56
5 vegetarian    high         1  104. -0.823
6 vegetarian    low          1  111.  1.99
```

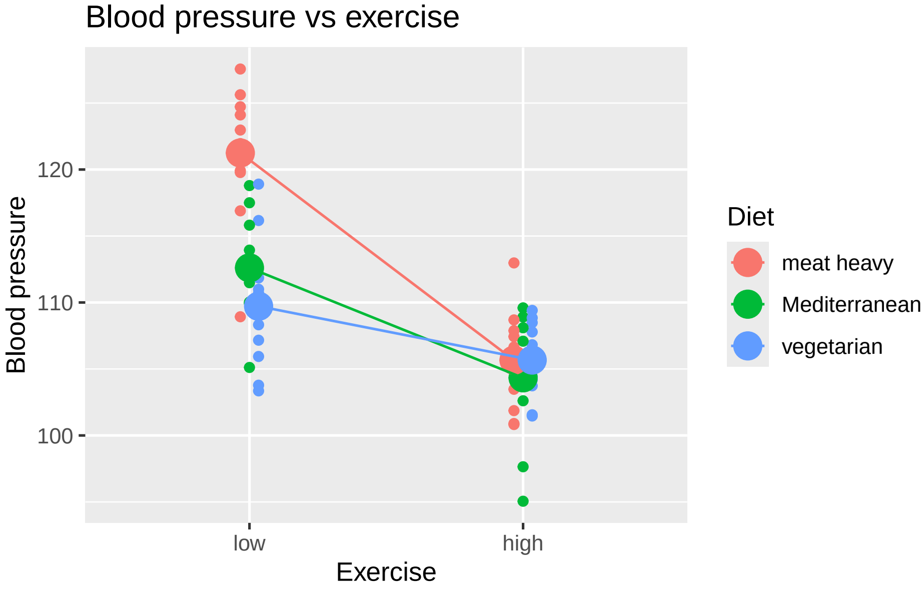
We can use the `xtabs` function to create a table of the data, by cross-tabulating the two treatments diet and exercise:

```
xtabs(~diet + exercise, data = bp_data_2cat)
```

```
               exercise
diet            low high
  meat heavy     10   10
  Mediterranean  10   10
  vegetarian     10   10
```

This tells us there are 10 replicates in each of the six treatment combinations.

And a visualisation of the data:



💡 Think, Pair, Share (#twoway-plot)

What do you conclude from this plot?

The model for 2-way ANOVA

Assume we have a factorial design with two treatments (factors), factor B and factor G.

And that we can label the levels of factor B as $i = 1, 2...$ and factor G as $j = 1, 2....$

Then we can denote a particular treatment combination as $B_i G_j$.

And let us set the one of the treatment combinations as the intercept of the model, and the let the intercept be equal to the mean of the observations in the treatment combination $B1G1$.

$$intercept = \frac{1}{n_{11}} \sum_{k=1}^{n_{11}} y_{1,1,k}$$

where:

- $y_{1,1,k}$ is the kth observation in the treatment combination $B1G1$
- n_{11} is the number of observations in the treatment combination $B1G1$.

And we will let all of the other treatment combinations be represented by the **effects** β_i and γ_j.

The resulting linear model is:

$$y_{ijk} = intercept + \beta_i + \gamma_j + (\beta\gamma)_{ij} + \epsilon_{ijk} \quad \text{with} \quad \epsilon_{ijk} \sim N(0, \sigma^2)$$

where

- y_{ijk} is the kth observation in the treatment combination of i and j.
- $(\beta\gamma)_{ij}$ is the interaction effect between the ith level of factor β and the jth level of factor γ.

In this model, we set $\beta_1 = \gamma_1 = 0$ and $(\beta\gamma)_{11} = 0$ because they are already included in the intercept.

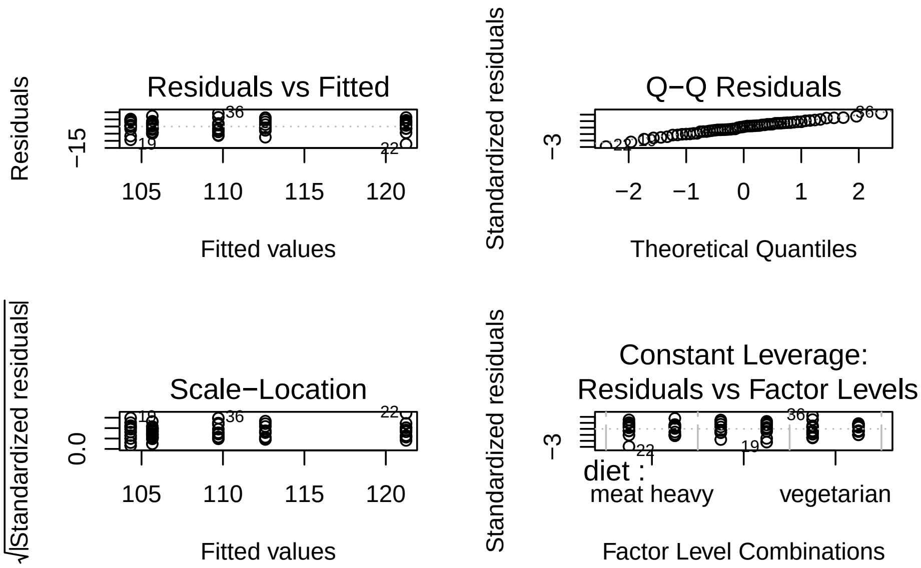
Using R for 2-way ANOVA

In R, a two-way ANOVA is as simple as one-way ANOVA, just add another variable:

```
mod1 <- lm(bp ~ diet * exercise, data = bp_data_2cat)
```

Note that, as we saw in the chapter about interactions, we include the main effects of diet and exercise and the interaction term with the short hand `diet * exercise`.

Of course we next check the model diagnostics:



No clear patterns: all is good.

Hypothesis testing

As is implied by the name "Analysis of variance" we analyse variances, here mean squares, to test hypotheses. And as before we use an F-test to do this. Remember that the F-test is a ratio of two mean squares (where mean squares are a kind of variance).

Think, Pair, Share (#full-degrees)

How many degrees of freedom for error will there be when we fit this model with both main effects and the interaction term? Hint: remember that the degrees of freedom for error is the number of observations minus the number of parameters estimated.

We can have have a null hypothesis of no effect for each of the two main effects and for the interaction. So we can do an F-test for each of these null hypotheses.

Here is the ANOVA table:

```
Analysis of Variance Table

Response: bp
              Df  Sum Sq Mean Sq F value    Pr(>F)
diet           2  390.74  195.37  9.9672 0.0002069 ***
exercise       1 1297.52 1297.52 66.1966 5.952e-11 ***
diet:exercise  2  340.67  170.33  8.6901 0.0005346 ***
```

```
Residuals     54 1058.46   19.60
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Interpretation: All three of the null hypotheses are rejected. Importantly, we see that the interaction term is significant, which means that the effect of one treatment is different depending on the level of the other treatment. This means that even though the main effects are significant, we cannot interpret them without considering the interaction term. I.e., we cannot say anything general about the effect of diet or exercise alone on blood pressure. We have to qualify any statement about the effect of diet or exercise with "depending on the level of the other treatment". Or state something like "the exercise reduces blood pressure greatly for people with a meat heavy diet, but reduces blood pressure only slightly for people with a vegetarian diet or Mediterranean diet".

Interpreting coefficients

We can look at the estimated coefficients to see the size of the effects, but be aware that it contains only a subset of the possible effects; it would contain different values if a different treatment combination were set to be the intercept of the model. Also be aware that often we are mostly interested in the F-test and their hypotheses test, and are less interest in the coefficients and their t-tests (unlike in a regression model where we are often interested in the coefficients and their t-tests).

Finally, be aware that it needs a bit of work to interpret the coefficients, because they are relative to the intercept. Let's try to figure out what that means. First, back to the table of the experimental design. This time we will put in the cells an expression for the mean of that treatment combination:

Exercise (G)

Diet (B)

Low (1)

High (2)

Meat heavy (1)

B_1G_1

B_1G_2

Mediterranean (2)

B_2G_1

B_2G_2

Vegetarian (3)

B_2G_1

B_3G_2

So, for example, the mean of the treatment combination “Meat heavy, Low” is B_1G_1. And the mean of the treatment combination “Mediterranean, High” is B_2G_2.

However, the coefficients in the summary table given by R are not like this. They are coefficients relative to an intercept / reference treatment combination. The reference treatment combination chosen by R is the first level of the first factor and the first level of the second factor. In this case, that is “**Meat heavy, Low**” – B_1G_1.

All of the other coefficients are about differences from this reference treatment combination.

So, for example, the coefficient for “High” in the “Exercise” factor (appearing as `exercisehigh` in the summary table) is the difference in mean blood pressure between the treatment combination “Meat heavy, High” (B_1G_2) and the treatment combination “Meat heavy, Low”. Put another way, $B_1G_2 = B_1G_1 + \gamma_2$ where γ_2 is the coefficient for “High” in the “Exercise” factor.

And the coefficient for “Mediterranean” in the “Diet” factor (appearing as `dietMediterranean` in the summary table) is the difference in mean blood pressure between the treatment combination “Mediterranean, Low” and the treatment combination “Meat heavy, Low”. Put another way, $B_2G_1 = B_1G_1 + \beta_2$ where β_2 is the coefficient for “Mediterranean” in the “Diet” factor.

Let us for a moment assume that the effects of diet and exercise are additive. If this is the case, then the mean for $B_2G_2 = B_1G_1 + \beta_2 + \gamma_2$. That is, the mean for “Mediterranean, High” is the mean for “Meat heavy, Low” plus the effect of “Mediterranean” plus the effect of “High”.

However, if the effects are not additive, then the mean for B_2G_2 is not $B_1G_1 + \beta_2 + \gamma_2$. Rather, it is $B_2G_2 = B_1G_1 + \beta_2 + \gamma_2 + (\beta\gamma)_{22}$. That is, the mean for “Mediterranean, High” is the mean for “Meat heavy, Low” B_1G_1 plus the effect of “Mediterranean” β_2 plus the effect of “High” γ_2 plus the non-additive effect between “Mediterranean” and “High” $(\beta\gamma)_{22}$.

Non-additivity implies an interaction, therefore the non-additive effect is the interaction effect. In the summary table these interaction effects are those that contain a colon (:), e.g., `dietMediterranean:exercisehigh`.

Here’s a graphical representation of how the coefficients in the summary table relate to the means of the treatment combinations:

Think, Pair, Share (#veghigh-estimate)

From the values in the coefficients table, calculate the estimated mean of the treatment combination “vegetarian, High”.

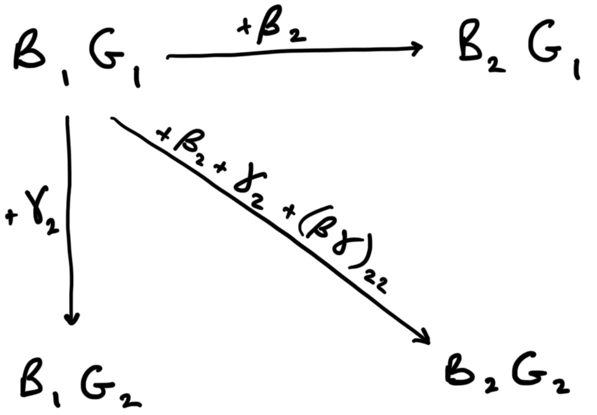


Figure 1: Understanding coefficients

Summing up

- 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.
- Two-way ANOVA is used when we have two categorical explanatory variables and can be used to test for interactions between them.

Additional reading

Please feel free to look at the follow resources for a slightly different perspective and some more information on ANOVA:

- Chapter 12 from Stahel book *Statistische Datenenalyse*
- *Getting Started with R* chapters 5.6 and 6.2