Introduction to Statistics using R

Nicolas Attalides

Who am I?

- **Name:** Nicolas Attalides
- **Coding in R since:** 2005
- **Profession:** Principal Data Scientist and trainer (9+ yrs.)
- **Education:** PhD in Statistical Science from UCL (2015)
- **R Status:** A never-ending evolving R dinosaur
- **Hobbies:** Tennis and coding (not at the same time)

Course agenda – Day 1: Morning

09:30 – 09:45 : Part 1 – Set up and introductions

09:45 – 10:45 : Part 2 – A reminder on R objects, loading data, data transformations, workflow and visualisations

10:45 – 11:00 : Data

15 min Break

11:15 – 11:45 : Part 3 – Summary statistics

11:45 – 12:30 : Part 4 – Univariate and Bivariate type plots

12:30 – 12:45 : Quiz challenge

12:45 – 13:45 : Lunch break

Course agenda – Day 1: Afternoon

13:45 – 14:00 : Part 5 – Populations, Samples and random variables

14:00 – 15:20 : Part 6 – Statistical distributions (Normal, Poisson and Binomial)

20 min Break

15:40 – 16:10 : Part 7 – Confidence intervals

16:10– 16:25 : Quiz challenge

16:25 – 16:30 : Summary / Q&A

Course agenda – Day 2: Morning

09:30 – 09:45 : Quick recap of Day 1 and Data + questions

09:45 – 10:00 : Part 8a – Defining a statistical test

10:00 – 11:00 : Part 8b – Statistical tests (t-test, paired t-test, proportion test)

20 min Break

11:20 – 12:15 Part 8b – Statistical tests (t-test, paired t-test, proportion test)

12:15 – 12:30 : Quiz challenge

12:30 – 13:30 : Lunch break

Course agenda – Day 2: Afternoon

13:30 – 14:40 : Part 8c – Statistical tests (KS, Mann Whitney U test and F test)

20 min Break

15:00 – 15:30 : Part 9 – ANOVA

15:30 – 16:00 : Part 10 – Simple/Multiple linear regression model

16:00 – 16:15 : Quiz challenge

16:15 – 16:30 : Summary / Q&A

SET UP AND INTRODUCTIONS

Course format

- **PowerPoint** slides for the course content
- **Live coding** with demos and exercises
- Online **quiz challenges**
- You are strongly advised to **actively participate** by sharing your screen
- **Ask questions** or ask for help with errors at any time

  (we've all created bugs!)

WARNING! Random animated GIFs of cats may appear during the course

Course objectives

1. Reference **material** for refreshing **key tasks** in
2. Learn about summary statistics and creating **visualisations** with **{ggplot2}**
3. Learn about **statistical distribution** and **confidence intervals**
4. Learn how to perform **statistical tests** such as t-tests or Kolmogorov Smirnov test (for one or two samples)
5. Learn how to create a single or multiple **linear regression** model
6. Learn how to carry out **ANOVA** in R

Who am I?

- **Name:**
- **Coding in R since:**
- **I am studying / working in … :**
- **I want to learn about statistics in R because …**

Systems check

- I have installed R

- I have installed RStudio

Systems check

```
install.packages(c("tidyverse", "ggcorrplot"))

# Include all dependencies. Select "NO" when asked to install from source

library(tidyverse) # Load tidyverse package
library(ggcorrplot) # Load ggcorrplot package

sessionInfo() # Check version information about R and OS
```

For this course I am using:

- R (version 4.3.3)
- RStudio (version 2024.12.1)
- **{tidyverse}** (version: 2.0.0)

Online Google Document

Please take a moment to check you have access to the Google document …

This allows for **real-time** collaboration to easily **share code**, exercise solutions, useful links etc.

R OBJECTS, LOADING DATA, DATA TRANSFORMATIONS, WORKFLOW AND VISUALISATIONS

Objectives

A quick reminder on:

1. R objects and how to **create** and **remove** them
2. How to perform some of the basic **data transformations** such as selecting columns, filtering rows and creating new columns
3. How to use the **pipe operator** to improve your code **workflow**
4. How to use the **{ggplot2}** package to create a simple **visualisation** with mapping **aesthetics** and adding **geometries**

R objects

An R object is the "thing" that we **create** during an **R session** and is used in R to perform statistical and graphical analysis.

Object	Description
vector	a 1-dimensional collection of elements of the same data type
matrix	a 2-dimensional collection of elements of the same data type
data frame	a 2-dimensional collection of elements of potentially different data types
list	an R object that contains other R objects

These are the most typical objects used in statistical analysis in R

R data types

Below is a table of the typical **data types** that the R objects, such as vectors, matrices and data frames can be/contain.

Data Type	Example
character (also known as string)	“A” or “red” or “The sky is blue”
numeric (real or decimal)	2 or 2.15
integer	5L or 6L (the L tells R that this is an integer)
logical	TRUE or FALSE

Creating R objects

It is important to note that when creating an R object the **name** you assign it to **must** start with a letter and contain only letters, numbers and underscore characters.

To create an object in R we need to use "**<-**" known as the **assign operator**. For example: Create an R object with the name "x" and assign the value 42, then print the object.

```
x <- 42

print(x)
```

Creating R objects

The **{base}** package provides a suite of functions that enables us to create R objects of specific types.

Object	Function	Arguments
vector	`c()`	Values to be combined separated by commas
matrix	`matrix()`	Data, number of rows and columns
data frame	`data.frame()`	Named variables separated by commas
list	`list()`	Variables separated by commas

Creating R objects of specific data types

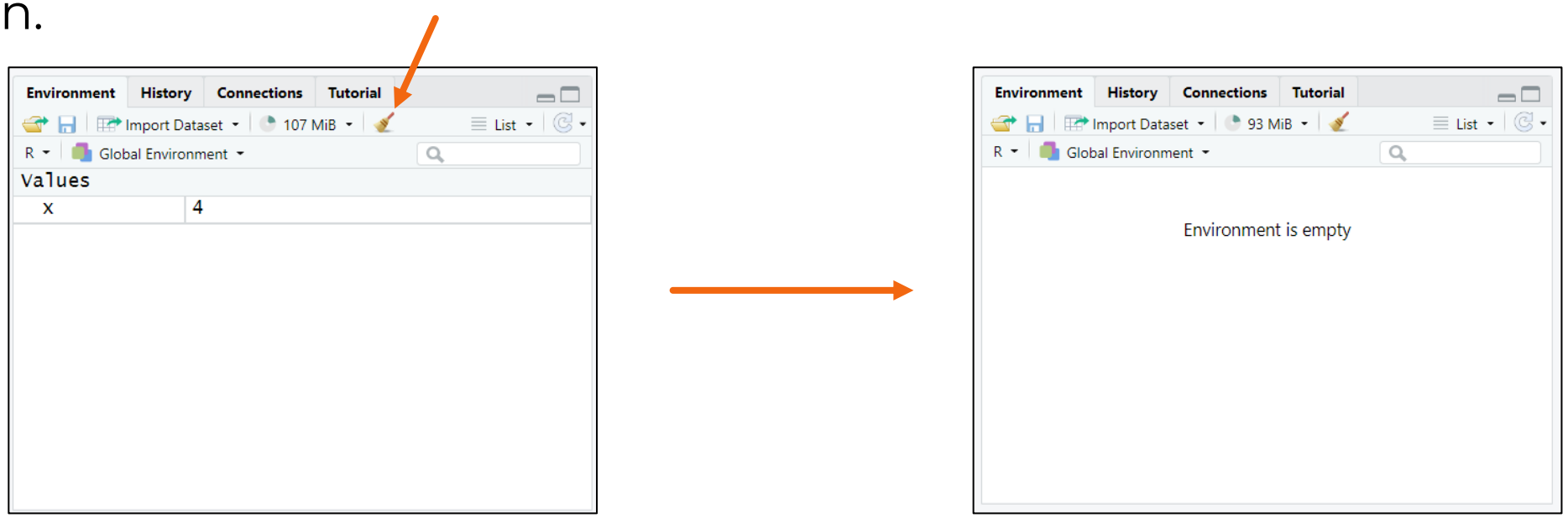
We saw earlier how we can create an R object such as a vector. Now that we have introduced the various **data types** in R we can combine the two concepts and therefore refer to an object by its data type. For example:

```
x <- c(1, 2, 3, 4)

y <- c(TRUE, FALSE)
```

We would call “x” a **numeric vector** and “y” a **logical vector**.

Clear your R workspace

When you start a new project, it is a good idea to start with an empty workspace. We would therefore want to **remove all objects** that we previously created in our workspace. This is easily done by pressing the  icon.



Remove an R object from the workspace

Alternatively we might want to **remove** just a few of the objects and keep the rest. This can be done using the `rm()` function from the **{base}** package.

```
rm(x) # Removes the object x from the workspace
```

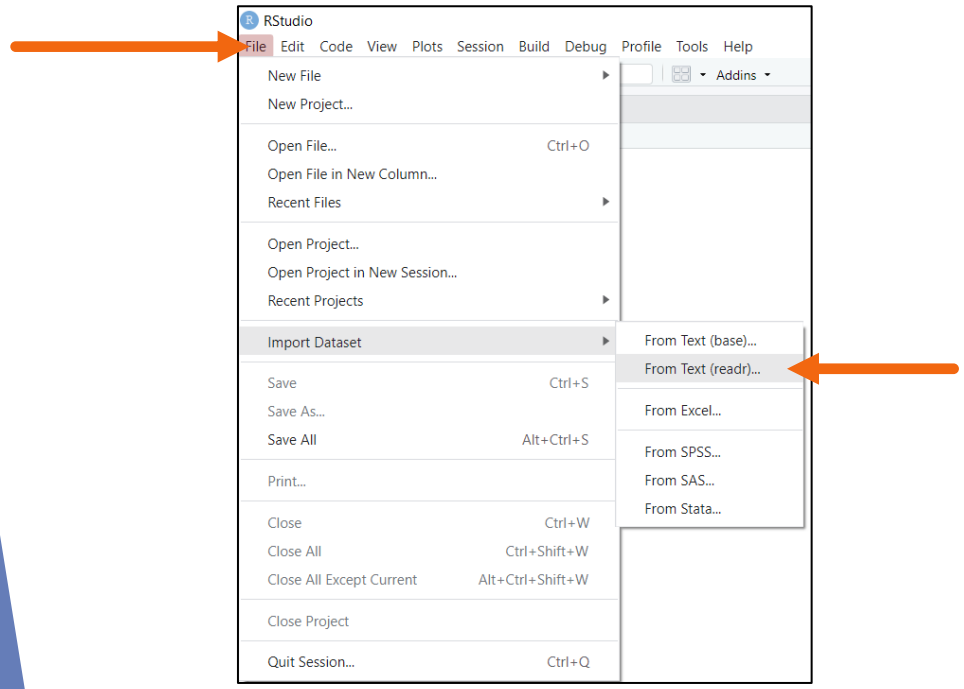
We can also use the `ls()` function and use it as the list argument of the `rm()` function in order to clear the entire workspace.

```
rm(list = ls()) # Removes all the objects from the workspace
```

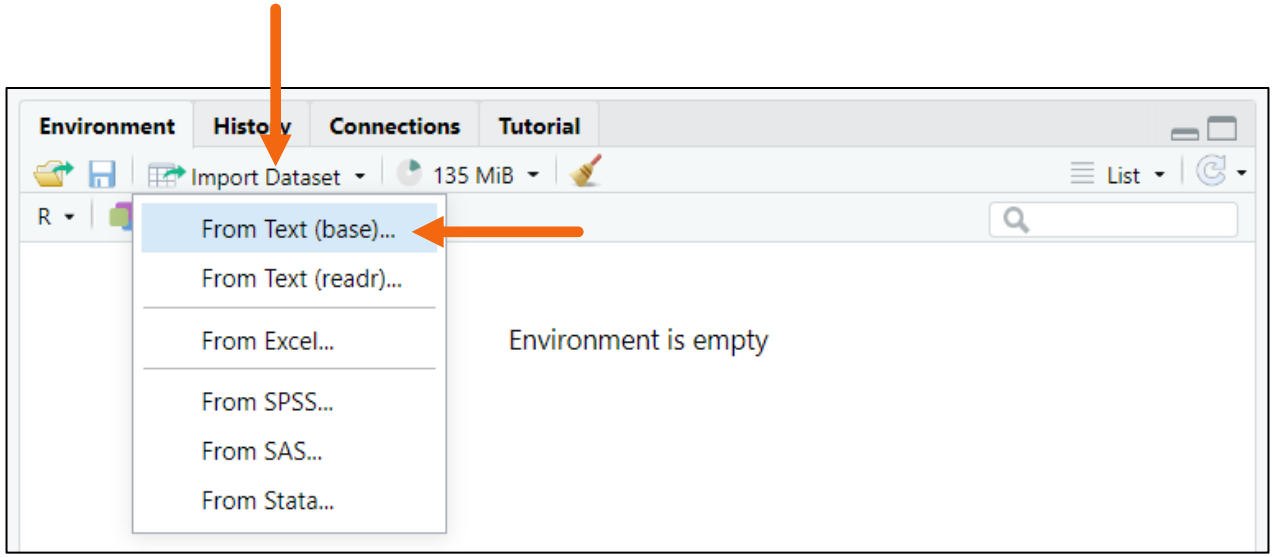
Load data into R using RStudio interface

First we will learn how to load data using the RStudio interface. Select:

File > Import Dataset > From Text (readr) ...

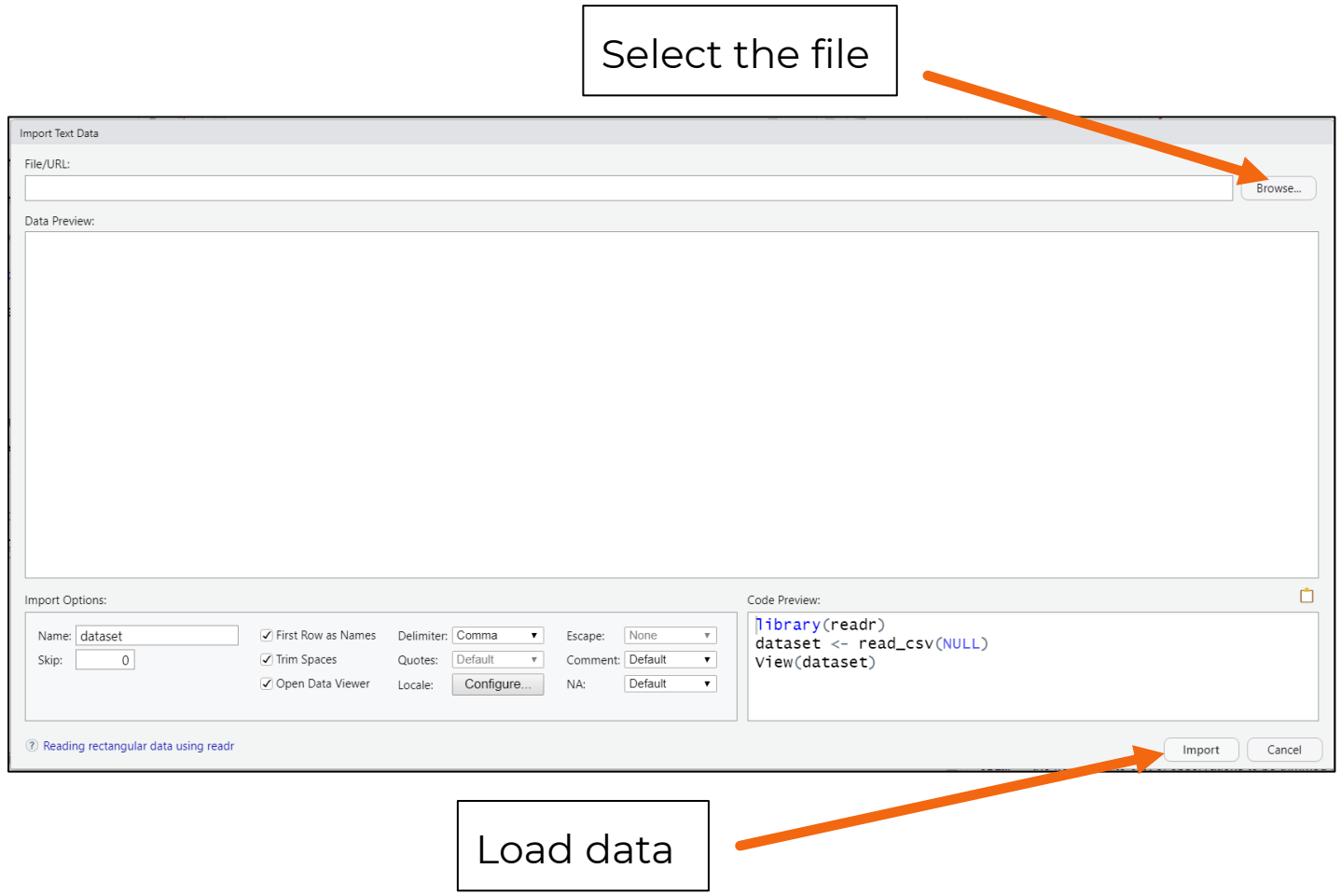


or



Load data into R using RStudio interface

Use the import wizard to browse and select the file to load and the click the “import” button to load the data.



Using an R project with a dedicated **data folder** is wise!

Load a CSV file into R using command line

It is very common for data to be found in **CSV** files (comma-separated values).

We use the `read_csv()` function from the **{readr}** package to load a CSV.

Don't forget that we need to specify the file path!

	A	B	C	D	E
1	name	height	weight	hair_color	skin_color
2	Luke Skywalker	172	77	blond	fair
3	C-3PO	167	75	NA	gold
4	R2-D2	96	32	NA	white, blue
5	Darth Vader	202	136	none	white
6	Leia Organa	150	49	brown	light
7	Owen Lars	178	120	brown, grey	light
8	Beru Whitesun lars	165	75	brown	light
9	R5-D4	97	32	NA	white, red
10	Biggs Darklighter	183	84	black	light
11	Obi-Wan Kenobi	182	77	auburn, white	fair

```
library(readr)
starwars <- read_csv("insert path to file")
```

Remember: In Windows the path needs to use either \\ or / as the directory separator

STARWARS DATA

Data

One of the datasets that we use during the course is:

Starwars characters: Various features of Starwars characters such as height, weight, hair colour etc.

DATASET CITATION: Dataset available from **{dplyr}** package and originally from SWAPI, the Star Wars API, https://swapi.dev/

Data Dictionary

- name
- height (in cm)
- weight (in kg)
- hair_color
- skin_color
- eye_color
- age (in years)
- gender (the gender role or gender identity of the character as determined by their personality or the way they were programmed (as in the case for Droids)
- homeworld
- species

Exercise 1 (5min)

1. Create the object “df”: A data frame with the data from the in-built `mtcars` data frame (from the **{datasets}** package)
2. Remove “df” with 1 command
3. Use the `read_csv()` function from the **{readr}** package to load the starwars CSV file
4. Extra: Use the `seq()` function to create object “date_vec” a Date vector of all the dates in the year 2023.

The “verb” functions of {dplyr}

One of the most popular R package to use for **data transformations** is **{dplyr}**. It is very powerful in handling tabular data such as **data frames** and is easy to use through “verb” functions. You can use **{dplyr}** to:

- **select** columns from your data
- **filter** your data to keep (or remove) rows that satisfy some conditions
- **mutate** your data to create new columns

The **first argument** of these functions is always the data!

Select columns from your data

The first “verb” function from the **{dplyr}** package to learn is the one called `select()`. This function helps us **select** the **columns** that we specify from our data frame. For example, from the data frame object called “starwars” I want to select the columns “name” and “height”…

```
library(tidyverse)

df <- select(starwars, name, height) # Select columns name and height
```

Note: Notice how we load the package **{tidyverse}**, in doing so, the other packages from this framework including **{dplyr}** are loaded!

Demo Example

1. Load the **{tidyverse}** package
2. Use the starwars object to select the columns: “name” and “height”

Group Exercise (5min)

1. What does the below mean when it is printed in our console when we load the **{tidyverse}** package?

```
> library(tidyverse)
── Attaching core tidyverse packages ──────────────────────────────────── tidyverse 2.0.0 ──
✓ dplyr     1.1.4     ✓ readr     2.1.5
✓ forcats   1.0.0     ✓ stringr   1.5.1
✓ ggplot2   3.5.1     ✓ tibble    3.2.1
✓ lubridate 1.9.3     ✓ tidyr     1.3.1
✓ purrr     1.0.2
```

2. Why do you think we get the below warning when we load the **{tidyverse}** package?

```
── Conflicts ──────────────────────────────────────────────────── tidyverse_conflicts() ──
✖ dplyr::filter() masks stats::filter()
✖ dplyr::lag()    masks stats::lag()
ℹ Use the conflicted package to force all conflicts to become errors
```

Exercise 2 (10min)

1. Load the **{tidyverse}** package
2. Load the starwars data (starwars.csv) in your R session and assign this to an R object called “starwars”
3. Select the columns: “name”, “height”, “eye_color” and “hair_color”
4. Extra: Remove the columns: “name”, “height” still using the `select()` function!
5. Extra: Use a selection helper (hint: look at help doc) to only select the columns that *end with* “color”

Filter rows from your data

The second “verb” function from the **{dplyr}** package to learn is the one called `filter()`. This function uses logical operators to **filter** the rows of the data frame. It **keeps** the rows that meet the logical conditions and **removes** those that do not. For example, from the data frame object called “starwars” I want to keep the rows such that the height of the character is greater than or equal to 175cm …

```
library(tidyverse)

df <- filter(starwars, height >= 175) # Filter rows according to condition
```

Demo Example

Use the starwars object to filter the dataset in order to keep the rows such that the height of the character is greater than or equal to 175cm

Exercise 3 (5min)

1. Using the same starwars data frame that you loaded in the previous exercise keep the rows that the weight of the character is greater than 50kg
2. What happens to the NA values?!
3. Now keep the rows that the weight is greater than 50kg and the height is greater than or equal to 175cm

Mutate your data

Very often you will want to **create** new columns from your existing data. The function **`mutate()`** from the **{dplyr}** package can be used to do exactly this task. For example, from the data frame object called "starwars" I want to create and add a new column called "height_m" that is the height of the character in meters.

```
library(tidyverse)
# Create a new column for height in metres
df <- mutate(starwars, height_m = height/100)
```

You can actually create multiple columns in a single function call.

Demo Example

Use the starwars object to create and add a new column called “height_m” that is the height of the character in meters

Exercise 4 (5min)

Using the same starwars data frame that you loaded in the previous exercise create a new column called “weight_g” which represents the characters weight in grams

Improve your workflow with {magrittr}

A package that has revolutionised the way we write R code is called **{magrittr}**. It has significantly improved the **readability** and **workflow** of code by introducing the “pipe” operator. It acts as a “then” operation where we can pass data from one function to another function very easily.

Fun fact: The package name is inspired by the famous artist René Magritte.

The caption reads “this is not a pipe” inspired by one of his artworks, hence the pipe operator.

Demo Example

Use the pipe operator from the **{magrittr}** package and the starwars object in order to...

Select the columns: “name”, “height” **THEN filter** the rows to keep only those characters that are greater than or equal to 175cm **THEN mutate** the data to create a new column called “height_m” that is the height of the character in meters

Try:
CTRL+SHIFT+M (Windows)
CMD+SHIFT+M (macOS)
and see what happens

Demo Example

```
# Pipe each data manipulation operation to the next one
df <- starwars %>%
  select(name, height) %>%
  filter(height >= 175) %>%
  mutate(height_m = height/100)
```

Important: Notice that when we use the pipe operator we are passing the data from one function to the next so we do not need to provide the dataset as the first argument anymore!

Exercise 5 (5min)

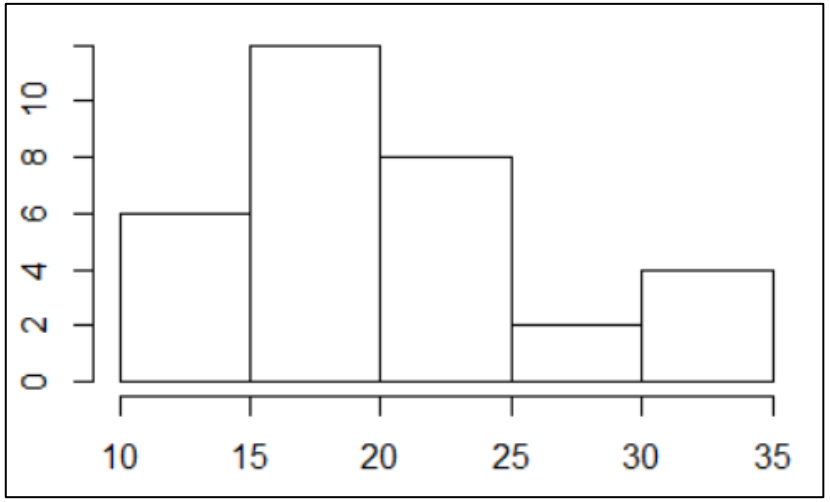
Use the starwars dataset and the pipe operator from the **{magrittr}** package in order to...

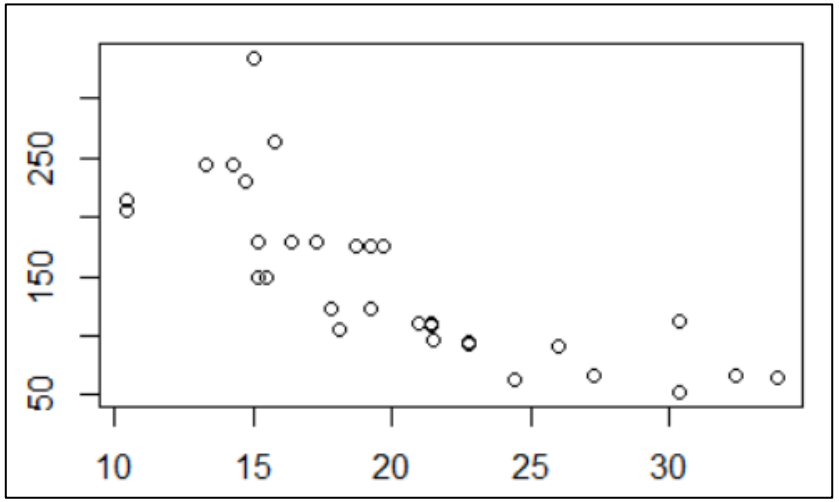
Select the columns: “name”, “height”, “eye_color” and “weight” **THEN filter** the rows to keep only those characters that have blue eyes **THEN mutate** the data to create a new column called “weight_g” which represents the characters weight in grams

Basic plots in R

By design R has in-built **graphical techniques** that allows us to quickly produce some of the most common plots like a histogram or a scatter plot.

However the process involved to refine and improve these plots can be tricky and sometimes not straightforward. Luckily, the **{ggplot2}** package offers a much more well-designed solution!





Introduction to {ggplot2}

is a graphical package that offers an **elegant** and **versatile** way of generating plots that follow the principles from the book "The Grammar of Graphics".

The main idea is to **break down a plot** into a collection of graphical components. Using **{ggplot2}** we create graphical objects by adding various **layers** such as **data** and **geometric objects** together. We can also specify other visual elements such as a title, legend and colours.

More information about **{ggplot2}** can be found here: https://ggplot2.tidyverse.org/

Build a simple plot

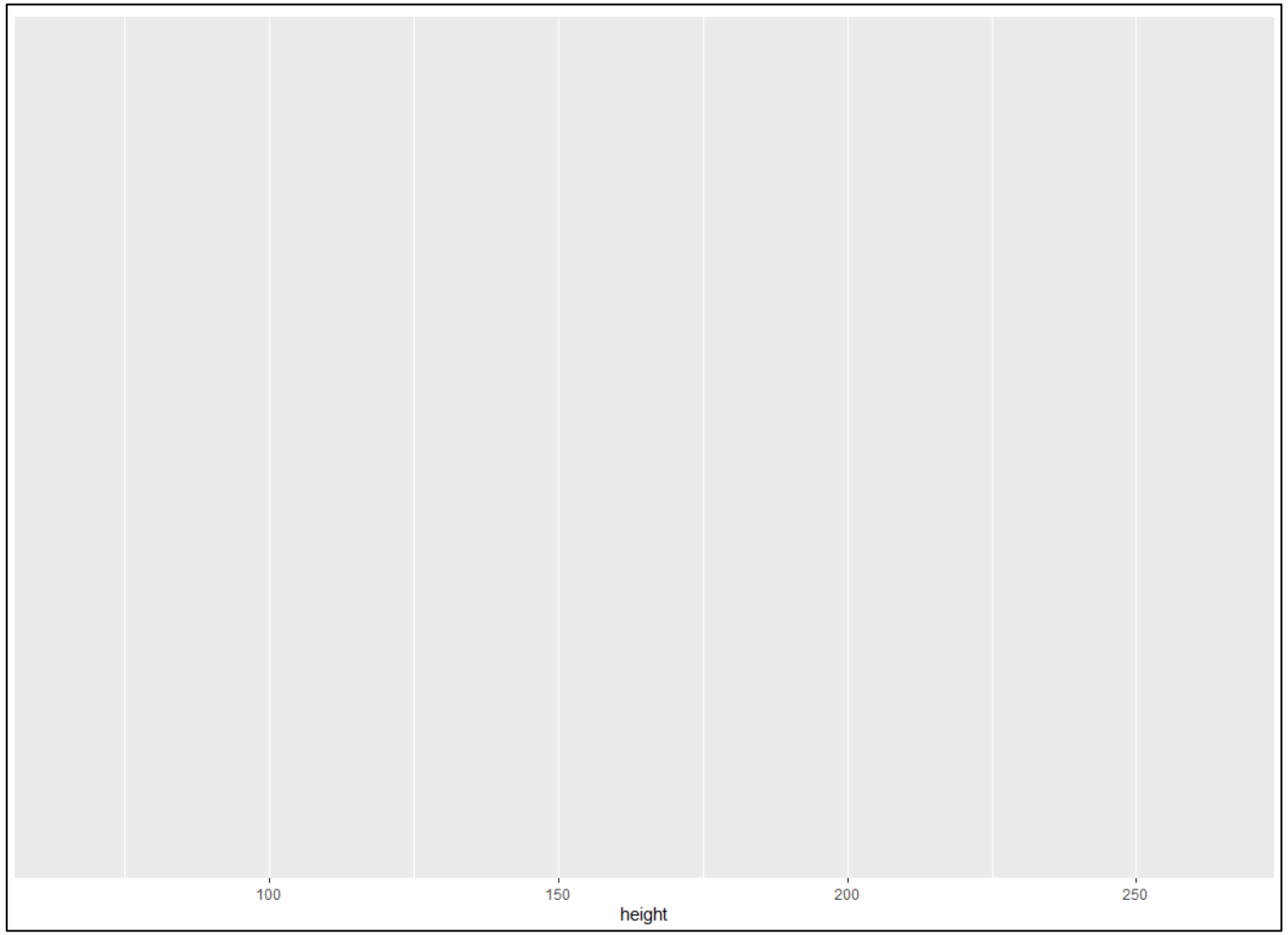
The minimum components required to *initiate* a **graphical object** using **{ggplot2}** is:

- **Data** in the form of a data frame (or tibble – a modern data frame)
- An **aesthetic mapping** that defines how the variables from the data are mapped to the visual space of the plot

```
# Create and print the graphical object
ggplot(data = starwars,
       mapping = aes(x = height))
```

You can also assign this to an object and then print the object to see the plot

Build a simple plot



Where is the plot?

Build a simple plot

The initial plot is blank because it does not have any geometric object added to it. Think of it as the initial layer being a **blank canvas** and we just simply need to "paint" a **geometry layer** on it.

To add any layer we use the "**+**" sign to the graphical object

There is a wide variety of geometry layers (referred to as "**geoms**") and can be accessed from the **{ggplot2}** package using the relevant `geom_*()` functions.

Geometry layers

Below is a curated list of some of the most commonly used geoms:

Geometry	Description
`geom_histogram()`	Adds a histogram layer
`geom_bar()`	Adds a bar chart layer
`geom_density()`	Adds a smooth density layer
`geom_area()`	Adds a shaded area layer
`geom_point()`	Adds points on the plot for a scatter plot layer
`geom_line()`	Adds lines on the plot connecting observations
`geom_boxplot()`	Adds a boxplot layer

For a more detailed list see: https://ggplot2.tidyverse.org/reference/#section-geoms

Build a simple plot continued

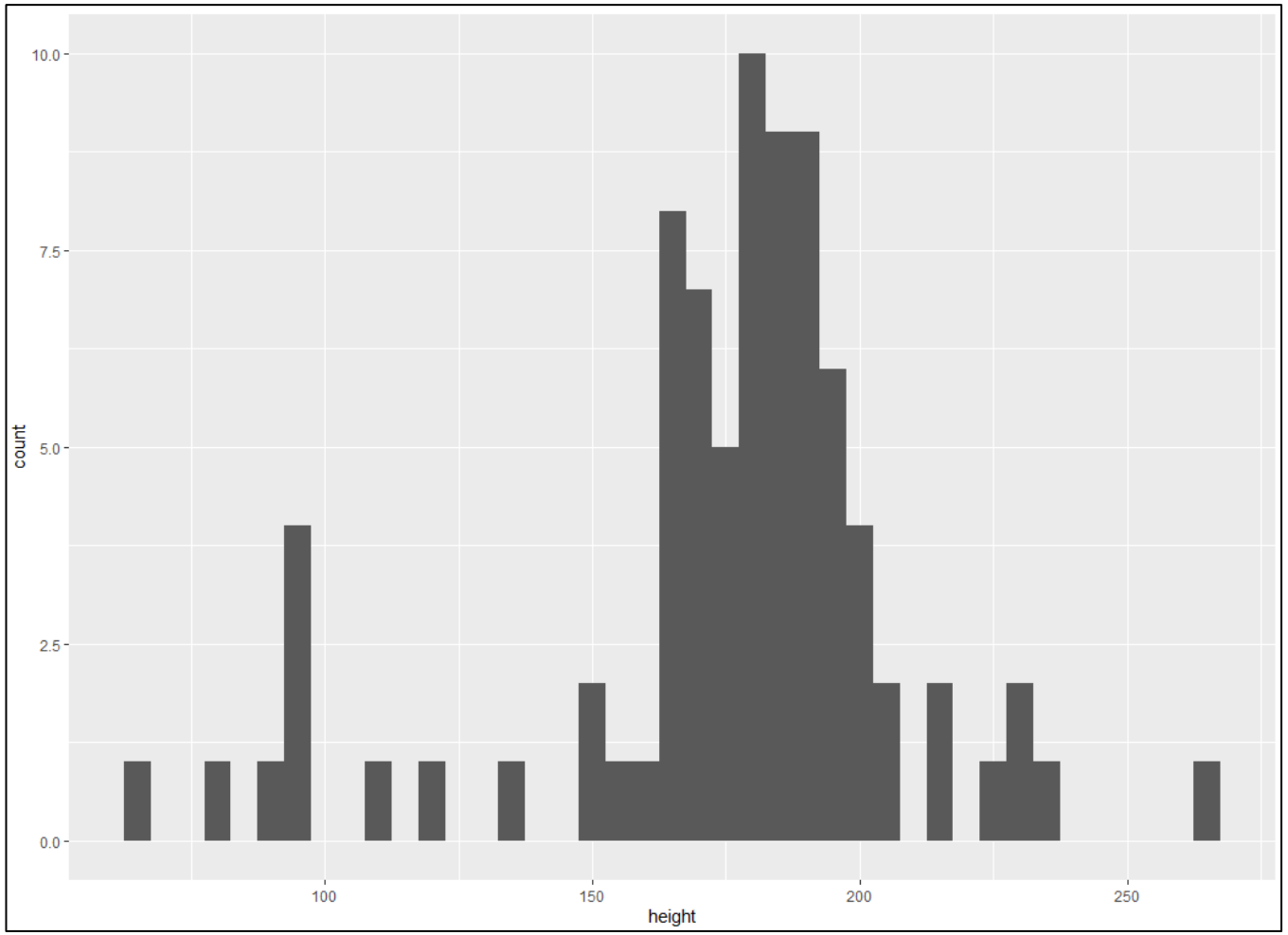
Lets add a histogram layer to our initial (blank) plot...

```
ggplot(data = starwars,
       mapping = aes(x = height)) +
  geom_histogram(binwidth = 5) # specify binwidth
```

Note that in this example there are 6 NAs in the “height” variable. When data is missing, **{ggplot2}** will throw a warning to inform us that those points were removed from the plot.

```
Warning message:
Removed 6 rows containing non-finite values (stat_bin).
```

Build a simple plot continued



Exercise 6 (5min)

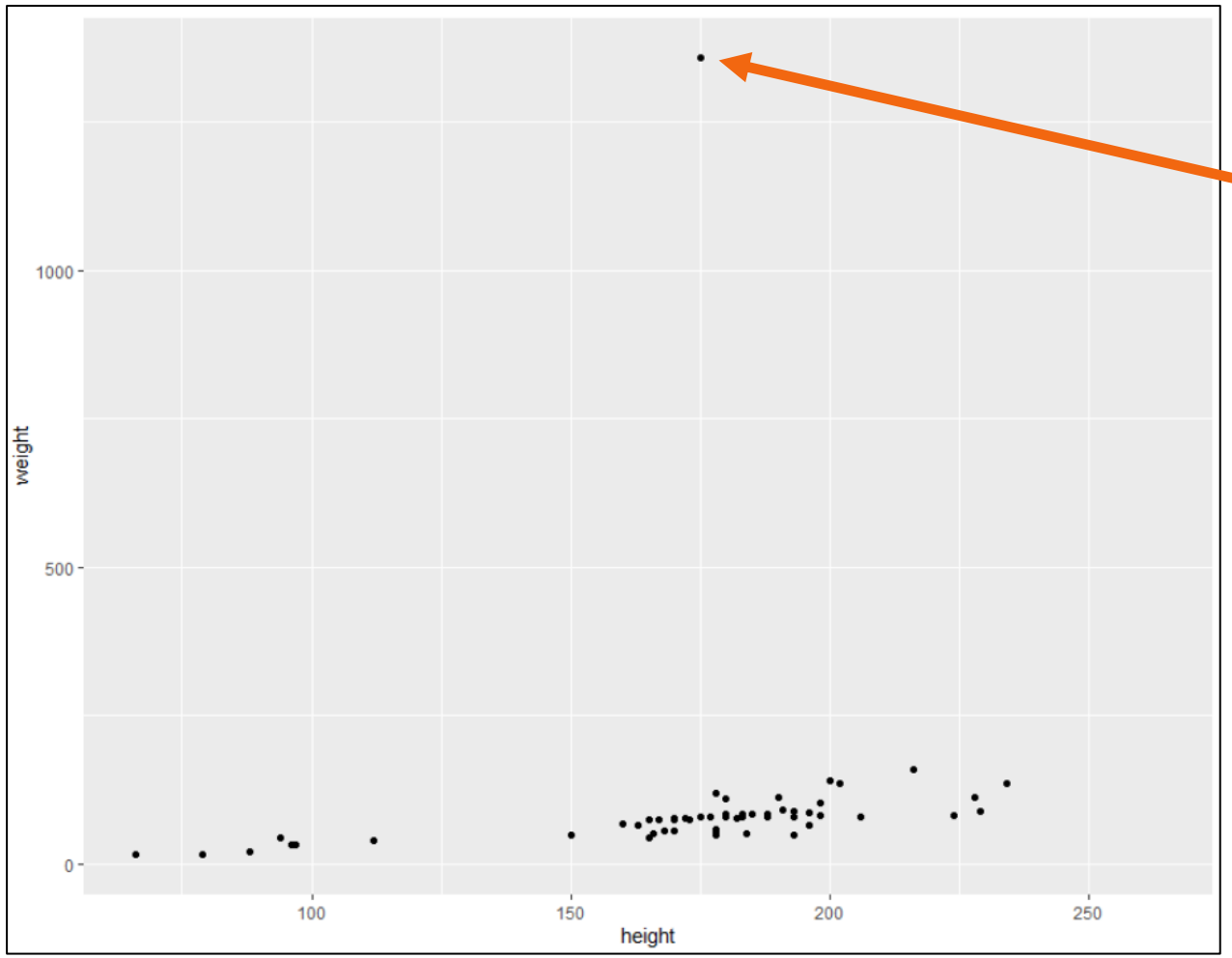
Use the starwars dataset to build a histogram plot of the `weight` variable (and a binwidth = 5)

Build a simple plot continued

Lets go back and re-define the **aesthetic mapping** of our data for variables to represent both **x** and **y**. Then we can use the `geom_point()` to represent our data as points, therefore adding the required layer to create a **scatter plot**.

```
ggplot(data = starwars,
       mapping = aes(x = height, y = weight)) +
  geom_point()
```

Build a simple plot continued



This point is messing up my plot!

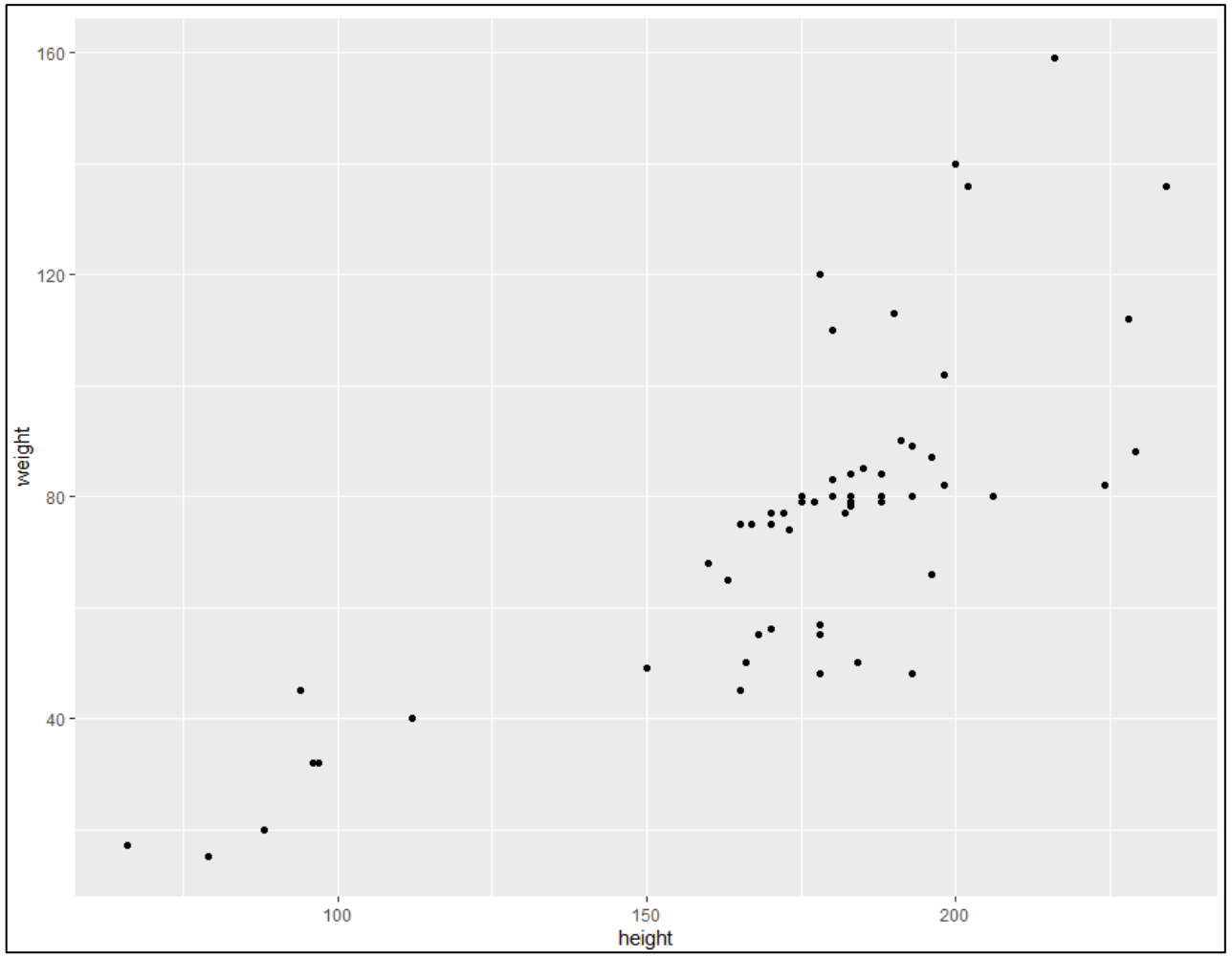
Build a simple plot continued

We can use **data manipulations** and **transformations** that we saw earlier to prepare our data and then use the **pipe operator** to pass it to the `ggplot()` function for plotting.

```
starwars %>%
  filter(weight < 200) %>% # Remove weight value point and pass to ggplot
  ggplot(mapping = aes(x = height, y = weight)) +
  geom_point()
```

Note: Use **%>%** for R code workflow – Use **+** for ggplot workflow

Build a simple plot continued



Exercise 7 (5min)

Replicate the plot that was demonstrated previously and try out some other data manipulations!

WINE DATA (DEMO)

Data

Another dataset that we use during the course is:

Red and White wines: Physicochemical test results (such as PH) and quality assessment graded by experts: 0 (very bad) and 10 (very excellent)

DATASET CITATION: P. Cortez, A. Cerdeira, F. Almeida, T. Matos and J. Reis. Modeling wine preferences by data mining from physicochemical properties. In Decision Support Systems, Elsevier, 47(4):547-553. ISSN: 0167-9236. https://scinapse.io/papers/2103459159

Data Dictionary

- **wine type**: specifies if the wines is red or white
- **fixed acidity**: the amount of fixed or non-volatile acids (do not evaporate readily)
- **volatile acidity**: the amount of acetic acid (high levels can lead to vinegar like taste)
- **citric acid**: amount of citric acid (can add ‘freshness’ and flavour to wines)
- **residual sugar**: the amount of sugar remaining after fermentation stops (the higher the amount the sweeter the wine)
- **chlorides**: the amount of salt in the wine
- **free sulfur dioxide**: the amount of free form of SO_2 that exists (it prevents microbial growth and the oxidation of wine)

Data Dictionary

- **total sulfur dioxide**: amount of free and bound forms of SO_2
- **density**: the wine density (depends on the alcohol and sugar content)
- **pH**: shows how acidic or basic a wine is on a scale from 0 (very acidic) to 14 (very basic)
- **sulphates**: a wine additive which contributes to SO_2 levels (act as antimicrobial and antioxidant)
- **alcohol**: the alcohol content of the wine (percent)
- **quality**: wine quality score on a scale from 0 (very bad) to 10 (very excellent)

STROKE DATA (COURSE)

Data

Another dataset that we use during the course is:

Health dataset for the analysis of brain strokes. The dataset contains patient demographic and medical history data.

DATASET CITATION: Kaggle: https://www.kaggle.com/datasets/prosperchuks/health-dataset?select=stroke_data.csv

Data Dictionary

- **id –** unique identifier of patient
- **gender –** “male”, “female“
- **age –** age of the patient
- **hypertension –** 0 if the patient doesn't have hypertension, 1 if the patient has hypertension
- **heart_disease –** 0 if the patient doesn't have any heart diseases, 1 if the patient has a heart disease
- **ever_married –** “no” or “yes”

Couse Data Dictionary

- **work_type –** “govt_job”, “private” or “self-employed”
- **residence_type –** “rural” or “urban”
- **avg_glucose_level –** average glucose level in blood
- **bmi –** body mass index
- **smoking_status –** "formerly smoked”, “never smoked” or “smokes”
- **had_stroke –** 1 if the patient had a stroke or 0 if not

Group exercise (5min)

1. Load the stroke dataset (stroke.csv) in your R session and assign this to an R object for example “stroke”
2. View the data and perform some simple inspections using the functions `nrow()`, `ncol()`, `names()` and `summary()`

Group exercise (5min)

1. Using the pipe operator and the data manipulation functions that we saw earlier, remove all the rows that have NA
2. How many males over the age of 50 smokes?

SUMMARY STATISTICS

Objectives

1. What are **summary statistics** and why are they useful?
2. Learn how to calculate measures of **central tendency**
3. Learn how to calculate measures of **spread** or **variability**
4. Learn how to calculate **quantiles** in R

Summary Statistics

Summary statistics is a method of analysing data using a set of mathematical tools in order to **describe** the **data** in a **meaningful** and **concise** way. Typically, we use summary statistics to describe:

- The **central tendency** of the data, such as the mean, median and mode
- The **spread** or **variability** of the data such as the range, variance, standard deviation and interquartile range

Summary Statistics

Calculating the summary statistics is useful because it allows you to:

- **Condense** and **simplify** a large complex dataset into a few key values
- **Understand** what a “typical” value in the data is and how much values tend to deviate from the average
- **Compare** between different datasets or different parts of the same dataset
- **Identify outliers** or extreme values in the data
- Perform **hypothesis tests** to determine if there's a statistically significant difference between groups in the data

Five number summary

The five-number summary is a set of five descriptive statistics that provide a quick overview of the distribution of a dataset.

- minimum (smallest observation)
- lower quartile (or first quartile)
- median (the middle value or second quartile)
- upper quartile (or third quartile)
- maximum (largest observation)

Demo Example

1. Use the wine dataset and the column `alcohol` to calculate the five number summary statistics using one function
2. Show how to get the minimum and maximum of a numeric vector

Demo Example

```
library(tidyverse)

wine <- read_csv("data/wine.csv")

summary(wine$alcohol)

min(wine$alcohol)

max(wine$alcohol)
```

Exercise 8 (5min)

1. Use the stroke dataset and the column **age** to calculate the five number summary statistics
2. Calculate the minimum and the maximum of the column **age** ensuring that your answer is not NA!

Hint: Check out the arguments of the function…

Measures of central tendency

Measures of **central tendency** are statistical measures that provide information about the centre or typical value of a dataset. They help you understand where most of the data points tend to cluster.

```
library(tidyverse)

wine <- read_csv("data/wine.csv")

mean(wine$alcohol, na.rm = TRUE)

median(wine$alcohol, na.rm = TRUE)
```

A note on the mode

Unfortunately, in R there is no built-in function that directly calculates the **mode** of a dataset like there are functions for mean and median. We can however evaluate this in a number of ways...

```
wine %>%
  group_by(quality) %>% # Group by the variable in question
  summarise(counts = n()) %>% # Count the number of observations
  filter(counts == max(counts)) %>% # Keep the maximum count
  pull(quality) # Pull the variable name
```

Here I am using the functions **`group_by()`** and **`summarise()`** from the **{dplyr}** package

Exercise 9 (5min)

Use the stroke dataset and the column **age** to calculate the following measures of central tendency:

1. Mean
2. Median

Measures of spread or variability

Measures of spread, are statistical measures that provide information about how data points are **spread** out in a dataset. They provide insights into the **variability** and distribution of the data.

```
range(wine$alcohol, na.rm = TRUE) # range of variable

IQR(wine$alcohol, na.rm = TRUE) # interquartile range of variable

var(wine$alcohol, na.rm = TRUE) # variance of variable

sd(wine$alcohol, na.rm = TRUE) # standard deviation of variable
```

Exercise 10 (5min)

Use the stroke dataset and the column **age** to calculate the following measures of spread:

1. Range
2. Interquartile Range (IQR)
3. Variance
4. Standard Deviation

Quantiles

Quantiles are a way to divide a dataset into **equal-sized parts**, each containing a certain percentage of the data points.

- **Quartiles** are a specific type of quantile that divides a dataset into 4 equal parts
- **Percentiles** divide a dataset into 100 equal parts
- **Deciles** divide a dataset into 10 equal parts, representing 10% intervals

Quantiles

It is very easy to evaluate **quantiles** in R. We use the function `quantile()` from the **{stats}** package and specify the variable we want to study as well as the quantile probabilities. For example:

```
quantile(wine$alcohol, na.rm = TRUE) # Default: uses quartiles

# Specify a set of probabilities to calculate the quantiles
quantile(wine$alcohol, probs = seq(from = 0, to = 1, by = 0.25), na.rm = TRUE)
```

Exercise 11 (5min)

Use the stroke dataset and the column **age** to calculate:

- Quartiles
- Percentiles
- Deciles

UNIVARIATE AND BIVARIATE PLOTS

Objectives

1. Learn about **univariate** and **bivariate** analysis
2. Learn how to visualise **continuous** variables
3. Learn how to visualise **categorical** variables
4. Learn how to visualise two **continuous** variables
5. Learn how to visualise a continuous and a categorical variable
6. Learn how to visualise two **categorical** variables

Univariate analysis

Univariate analysis is one of the simplest methods of performing **statistical analysis** on data. As the name suggests, univariate analysis involves examining a **single variable** at a time that can be either **continuous** or **categorical**.

It is very common when working with a dataset to perform **Exploratory Data Analysis** (EDA) which typically involves univariate analysis.

The aim is to generate descriptive statistics for the single variable and is often easier to represent these findings in a graphical way.

Univariate analysis - continuous variable

Examples of a **continuous variable** are age, height, weight, speed and distance. Some of the plot types that we can use to create a graphical representation of the distribution are:

- **Histogram –** Displays the frequencies or counts of data points falling into specific intervals or "bins“
- **Box plot –** Gives a visual summary of key characteristics of the data, including the centre, spread, and presence of outliers
- **Density –** Provides a smoothed representation of the data's underlying probability density function

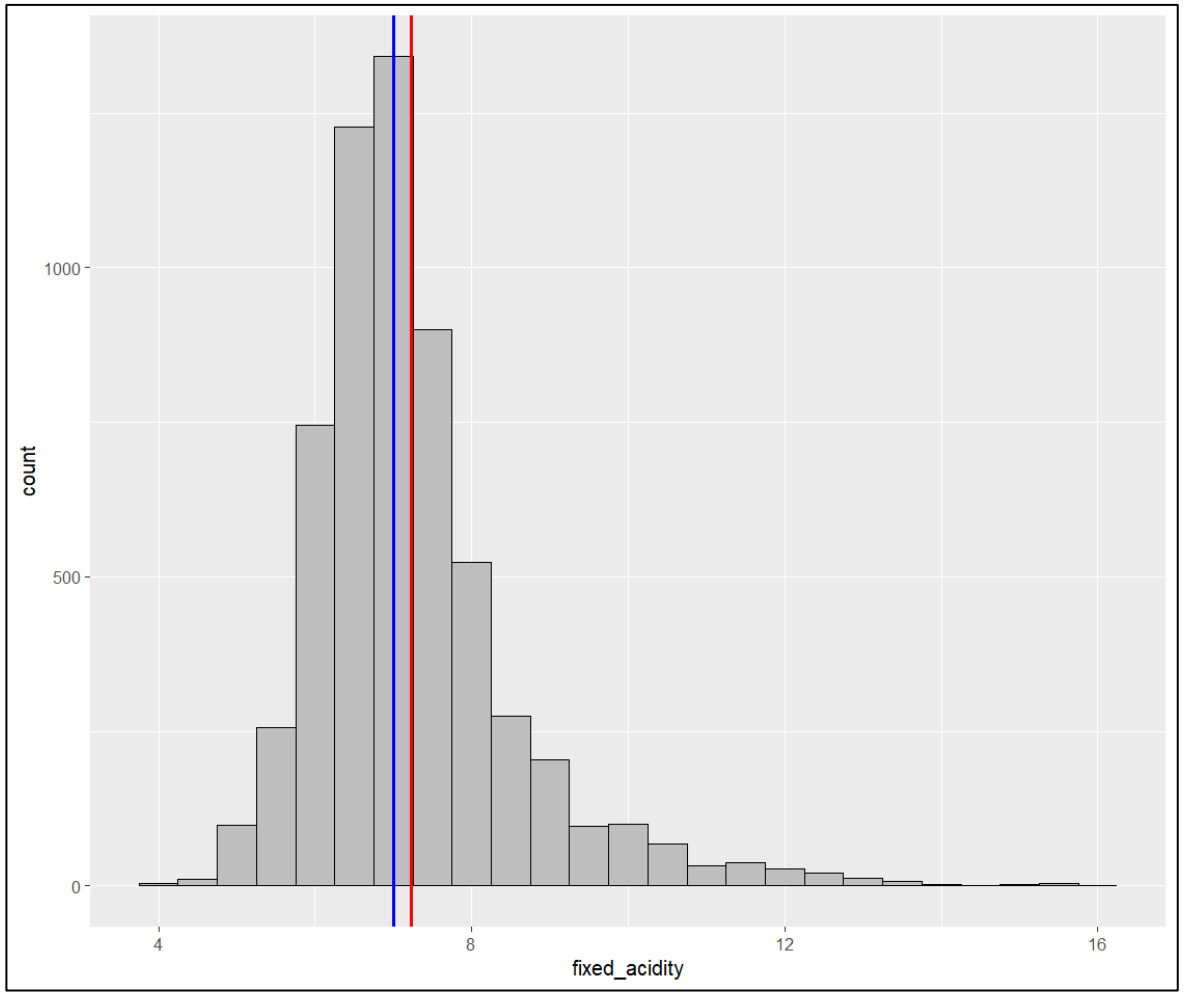
Demo Example - Histogram

1. Use the `fixed_acidity` variable from the wine dataset to plot a histogram with `binwidth = 0.5`
2. Calculate the mean and median values and add them to the plot as a vertical lines

Demo Example - Histogram

```
mean_value <- mean(wine$fixed_acidity)

median_value <- median(wine$fixed_acidity)

wine %>%
  select(fixed_acidity) %>%
  ggplot(mapping = aes(x = fixed_acidity)) + # specify x aesthetics
  # add histogram layer
  geom_histogram(binwidth = 0.5, colour = "black", fill = "grey") +
  # now add vertical line for mean
  geom_vline(xintercept = mean_value, lwd = 1, colour = "red") +
  # now add vertical line for median
  geom_vline(xintercept = median_value, lwd = 1, colour = "blue")
```

Demo Example - Histogram



Exercise 12 (10min)

1. Use the `avg_glucose_level` variable from the stroke dataset to plot a histogram with: `binwidth = 5`
2. Extra: Add a red vertical line representing the mean value and a blue vertical line representing the median value
3. Extra: Calculate the Lower and Upper quartiles and add them to the plot as green dashed lines

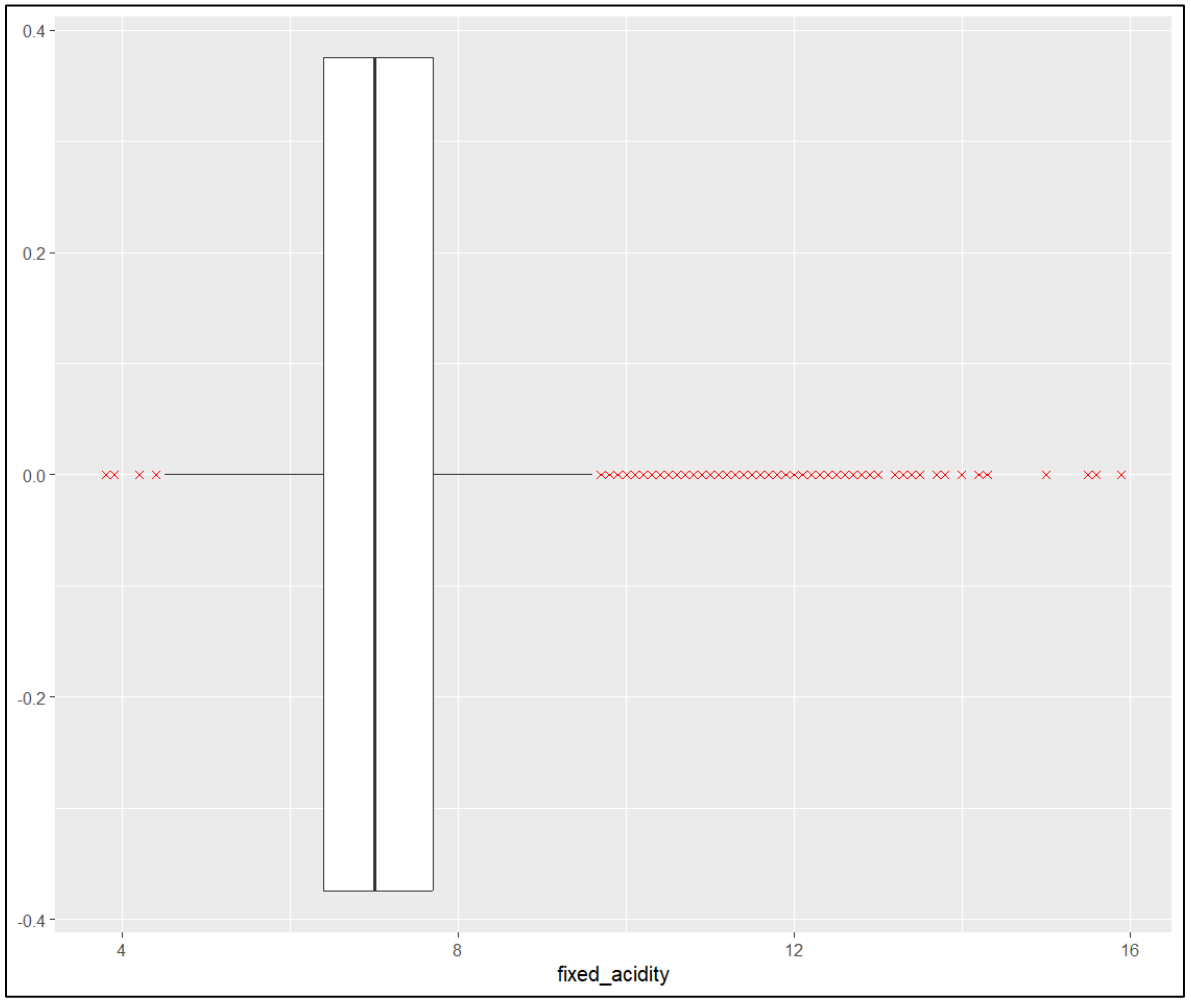
Demo Example - Boxplot

Repeat the previous demonstration but now using the boxplot geometry from the **{ggplot2}** package

Demo Example - Boxplot

```
wine %>%
  select(fixed_acidity) %>%
  ggplot(mapping = aes(x = fixed_acidity)) + # specify x aesthetics
  # add boxplot layer with specification about the outliers
  geom_boxplot(outlier.colour = "red", outlier.shape = 4)
```

Demo Example - Boxplot



Exercise 13 (5min)

Use the `avg_glucose_level` variable to repeat the previous exercise (without the vertical lines) but now using the boxplot geometry from the **{ggplot2}** package

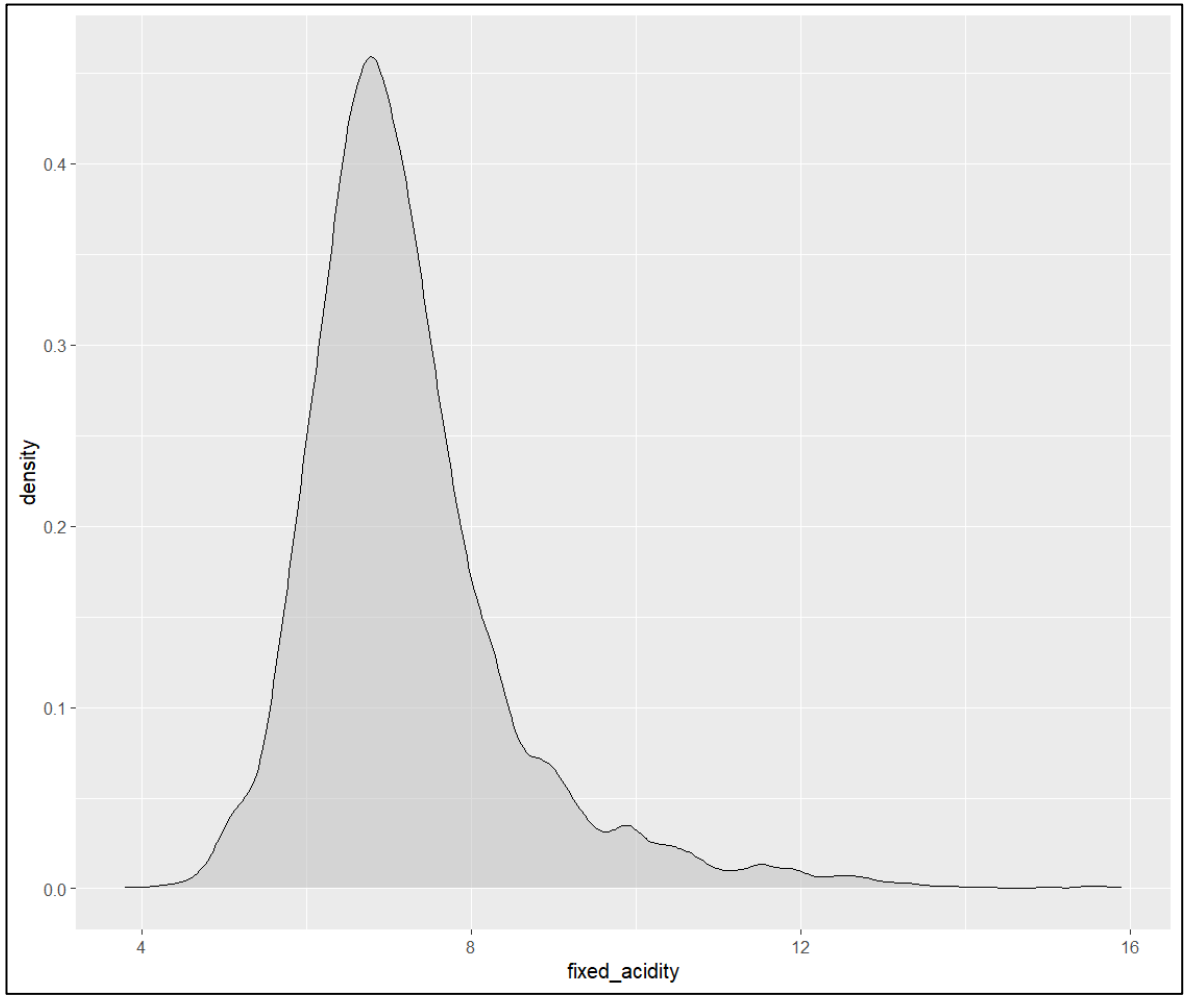
Demo Example – Density plot

Repeat the previous demonstration but now using the density geometry from the **{ggplot2}** package

Demo Example – Density plot

```
wine %>%
  select(fixed_acidity) %>%
  ggplot(mapping = aes(x = fixed_acidity)) + # specify x aesthetics
  # add density layer with specification about the fill colour & transparency
  geom_density(colour = "black", fill = "grey", alpha = 0.5)
```

Demo Example – Density plot



Exercise 17 (5min)

Use the `avg_glucose_level` variable to repeat the previous exercise (without the vertical lines) but now using the density geometry from the **{ggplot2}** package

Univariate analysis - categorical variable

Categorical variables can only take categories and represent quality aspects of our data. A category is a single value from a pool of possible categories. For example, gender, blood type, nationality etc.

When analysing these type of variables, we are interested in describing:

- **Frequency** of observations by category

It is worth noting that we can **convert continuous variables** (such as age) into categorical variables by grouping. A special case of a categorical variable with only two categories is known as a **binary variable**.

Demo Example – Bar plot

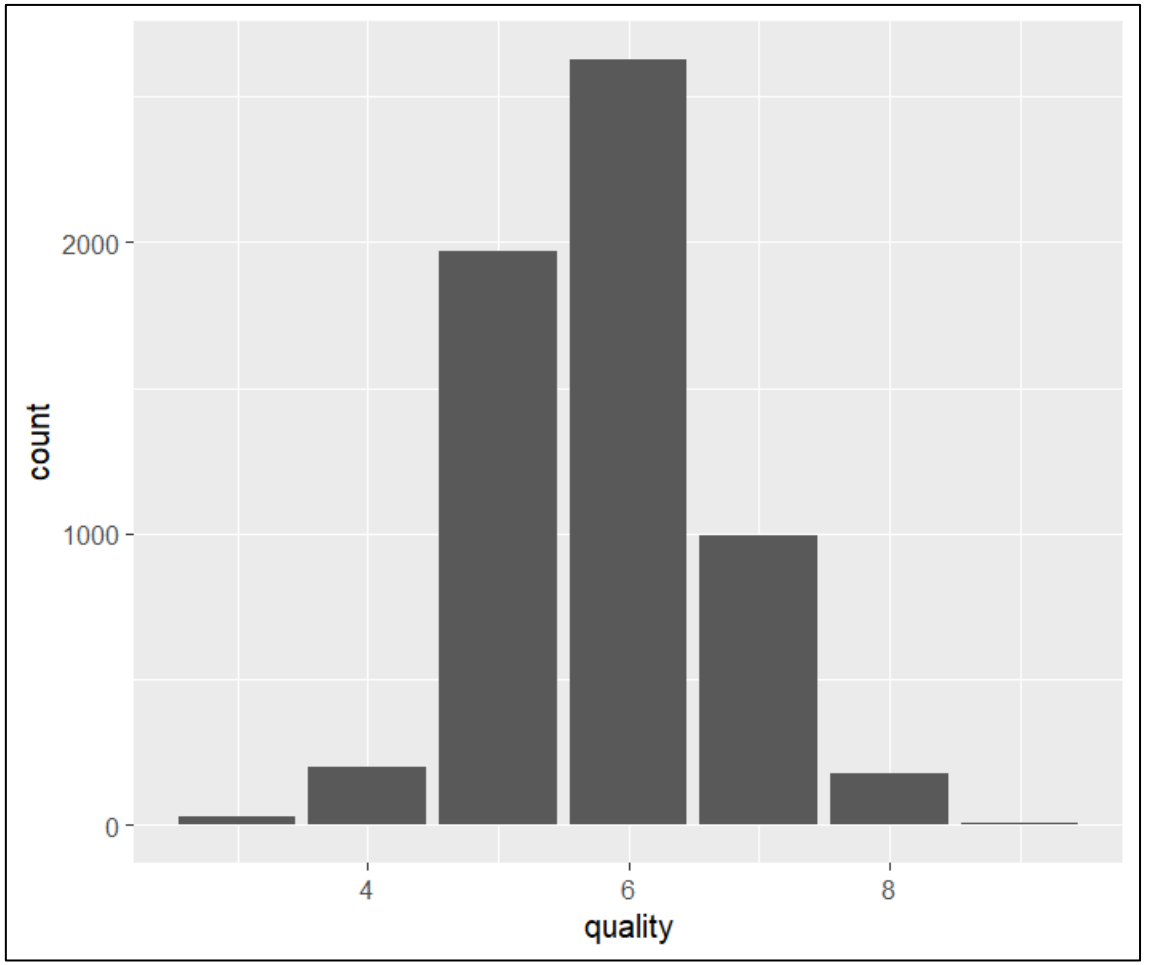
Create a bar plot for the `quality` variable from the wine dataset

Demo Example – Bar plot

```
wine %>%
select(quality) %>%
  # specify x aesthetics
  ggplot(mapping = aes(x = quality)) +
  # add bar plot layer
  geom_bar()
```

Demo Example – Bar plot

Something doesn't feel right with this bar plot... what do you think is the issue?

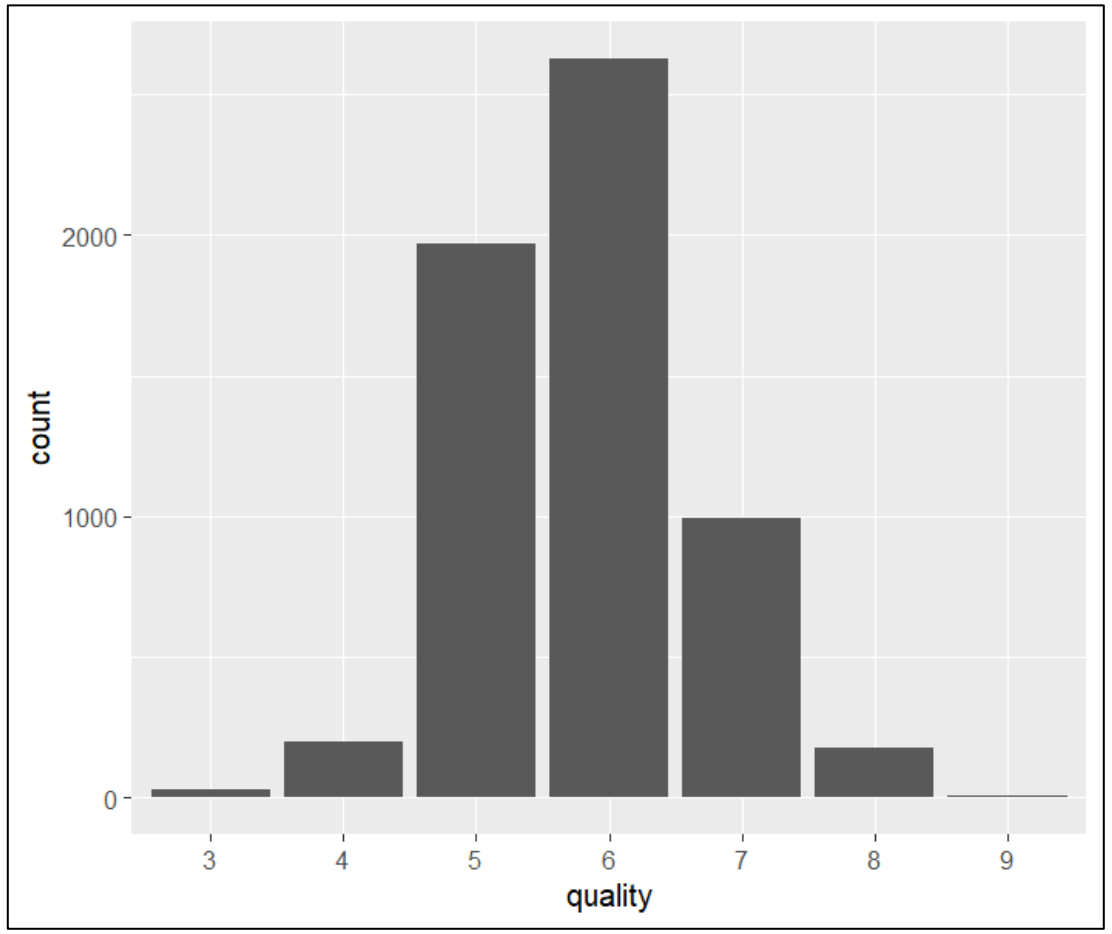


Demo Example – Bar plot

In closer inspection, the quality variable was loaded into R as a numeric data type. We therefore need to convert it to a **factor** to represent it as a **categorical variable**. It uses the values in the variable for the levels.

```
wine %>%
  select(quality) %>%
  # convert quality as a factor
  mutate(quality = factor(quality)) %>%
  # specify x aesthetics
  ggplot(mapping = aes(x = quality)) +
  # add bar plot layer with specification on how the bars are placed
  geom_bar(position = "dodge")
```

Demo Example – Bar plot



Exercise 15 (5min)

Continuing with the stroke dataset:

1. Convert the `smoking_status` variable to a factor
2. Create a bar plot for the `smoking_status` variable as in the example

Bivariate analysis

Bivariate analysis is another simple method of performing **statistical analysis** on data and can form part of an EDA.

You can think of it as an extension of univariate analysis because it involves examining **two variables** at a time that can be either **continuous** or **categorical** (or both).

The aim of bivariate analysis is to better understand the relationship between the two variables (or their correlation).

Remember: Correlation does not imply causation!

Bivariate analysis

During this course we examine the following pairs of variables:

- **Continuous – Continuous** (for example height vs weight) through a scatter plot
- **Continuous – Categorical** (for example height vs gender) through a box plot, histogram and density plot
- **Categorical – Categorical** (for example gender vs nationality) through a bar plot

Demo Example – Scatter plot

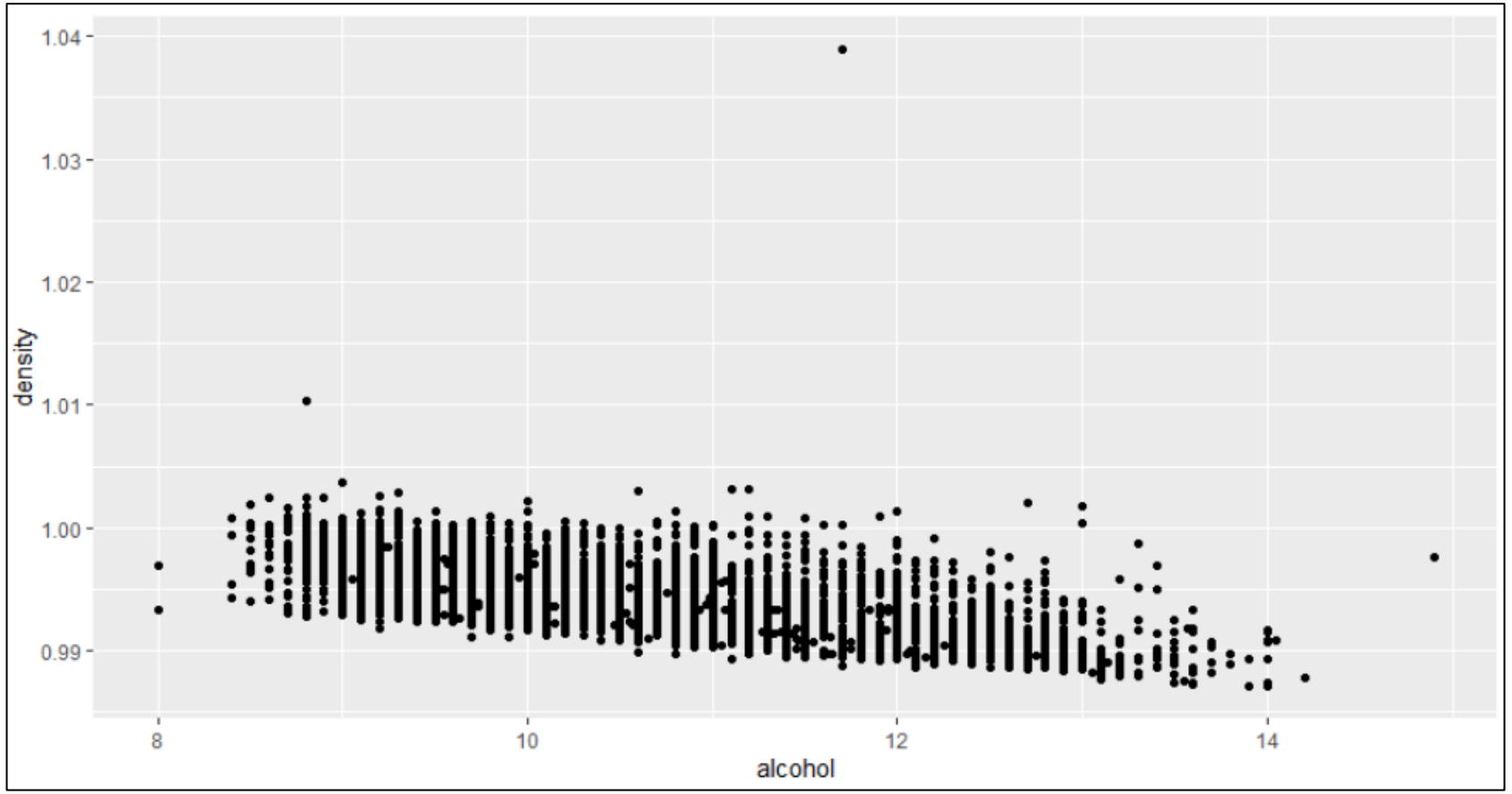
Using the wine dataset:

1. Prepare the plot data containing two categorical variables, namely the `alcohol` (x) and the `density` (y) variables
2. Create a scatter plot

Demo Example – Scatter plot

```
wine %>%
  select(alcohol, density) %>%
  # specify x and y aesthetics
  ggplot(mapping = aes(x = alcohol, y = density)) +
  # add points geometry layer
  geom_point()
```

Demo Example – Scatter plot



Exercise 16 (5min)

Using the stroke dataset:

1. Create a scatter plot of the two continuous variables, namely the `age` (x) and `avg_glucose_level` (y)
2. Add a `geom_smooth()` layer to the scatter plot. What do you think this is doing?

Demo Example – Box plot

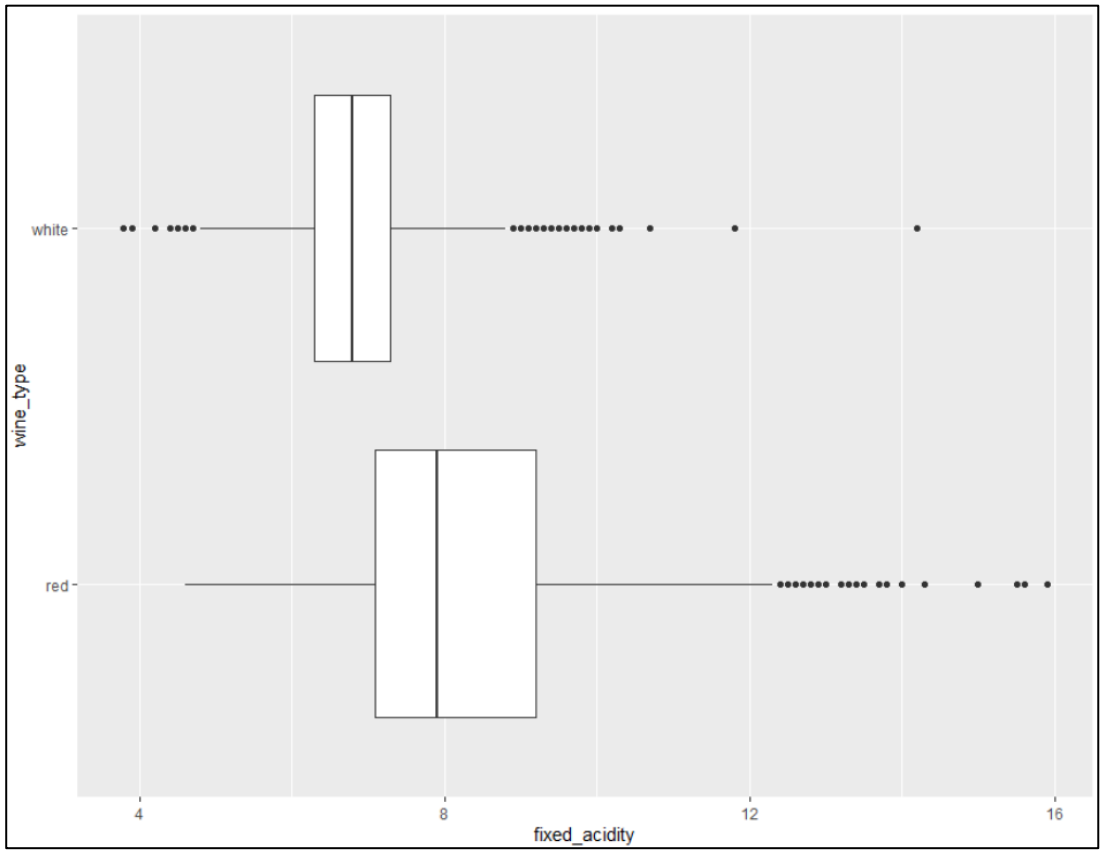
Use `wine_type` as the categorical variable and `fixed_acidity` as the continuous variable to:

1. Create a plain box plot of the continuous variable for each category
2. Repeat the above using the `fill` aesthetic

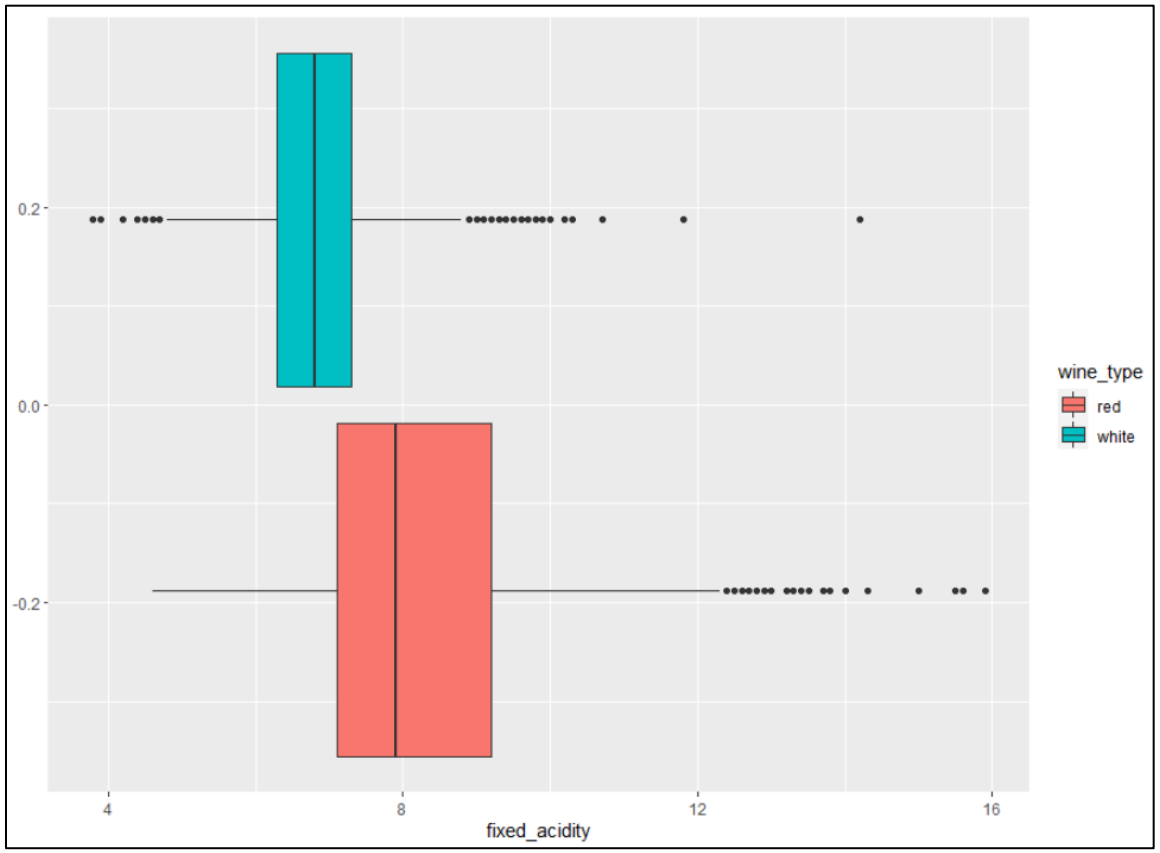
Demo Example – Box plot

```
wine %>%
  select(fixed_acidity, wine_type) %>%
  mutate(wine_type = factor(wine_type)) %>%  # convert to factor
  # specify x and y aesthetics
  ggplot(mapping = aes(x = fixed_acidity, y = wine_type)) +
  # add boxplot layer
  geom_boxplot()

wine %>%
  select(fixed_acidity, wine_type) %>%
  mutate(wine_type = factor(wine_type)) %>%  # convert to factor
  # specify x and y aesthetics
  ggplot(mapping = aes(x = fixed_acidity, fill = wine_type)) +
  # add boxplot layer
  geom_boxplot()
```

Demo Example – Box plot



With y aesthetic



With fill aesthetic

Exercise 17 (5min)

Use `had_stroke` as the categorical variable and `avg_glucose_level` as the continuous variable to create a density plot of the continuous variable for each category using the `fill` aesthetic

Demo Example – Bar plot

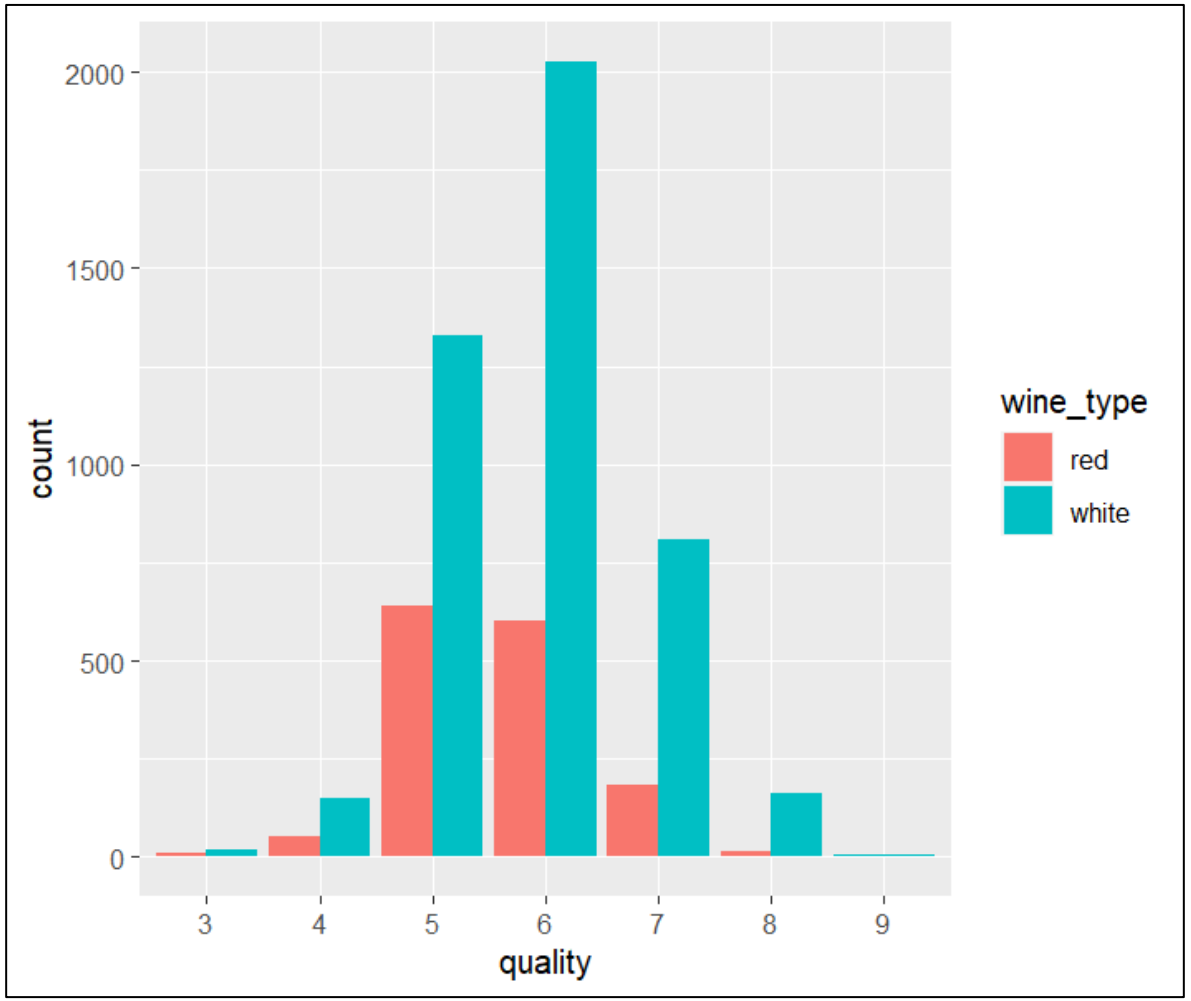
Use the `wine_type` and `quality` as the two categorical variables to:

1. Calculate the count for each unique combination of the pair of categorical variables
2. Use the data frame from above to create a bar plot (use the `geom_col()` function) of the two variables

Demo Example – Bar plot

```
plot_data <- wine %>%
  select(quality, wine_type) %>%
  mutate(quality = factor(quality)) %>%  # convert to factor
  mutate(wine_type = factor(wine_type)) %>%  # convert to factor
  group_by(quality, wine_type) %>%
  summarise(count = n()) %>%
  ungroup()

plot_data %>%
  # specify x aesthetics
  ggplot(mapping = aes(x = quality, y = count, fill = wine_type)) +
  # add bar plot layer
  geom_col(position = "dodge")
```

Demo Example – Bar plot



Demo Example – Correlation plot

1. Load the **{ggcorrplot}** package
2. Use the wine dataset and select the columns containing only the numerical variables

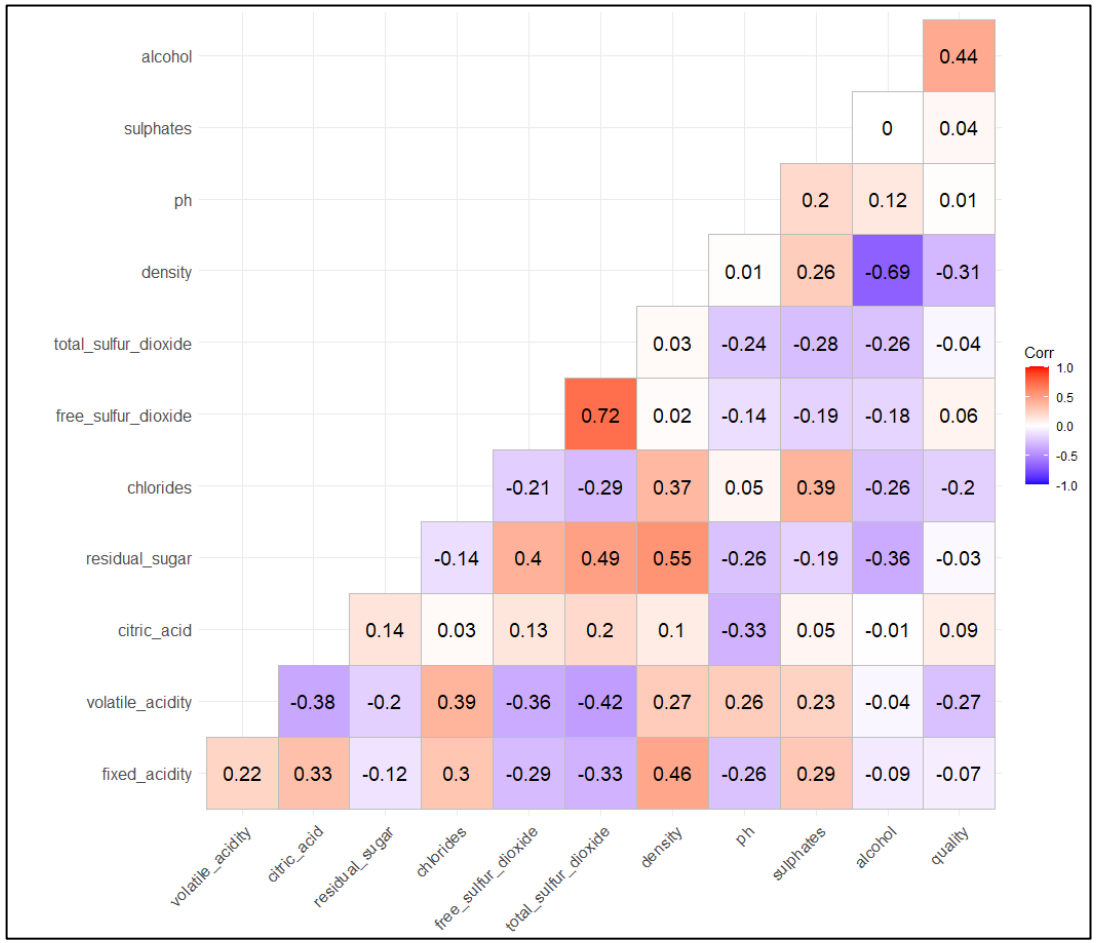
   Hint: Try using `select(is.numeric)` and see what happens…
3. Create a correlation plot using the `ggcorrplot()` function from the package **{ggcorrplot}**

This is an example of multivariate analysis

Demo Example – Correlation plot

```
library(ggcorrplot)

plot_data <- df %>%
  select(where(is.numeric)) %>% # select only numerical columns in data
  cor()

ggcorrplot(plot_data,
           type = "lower",
           lab = TRUE,
           lab_size = 5)
```

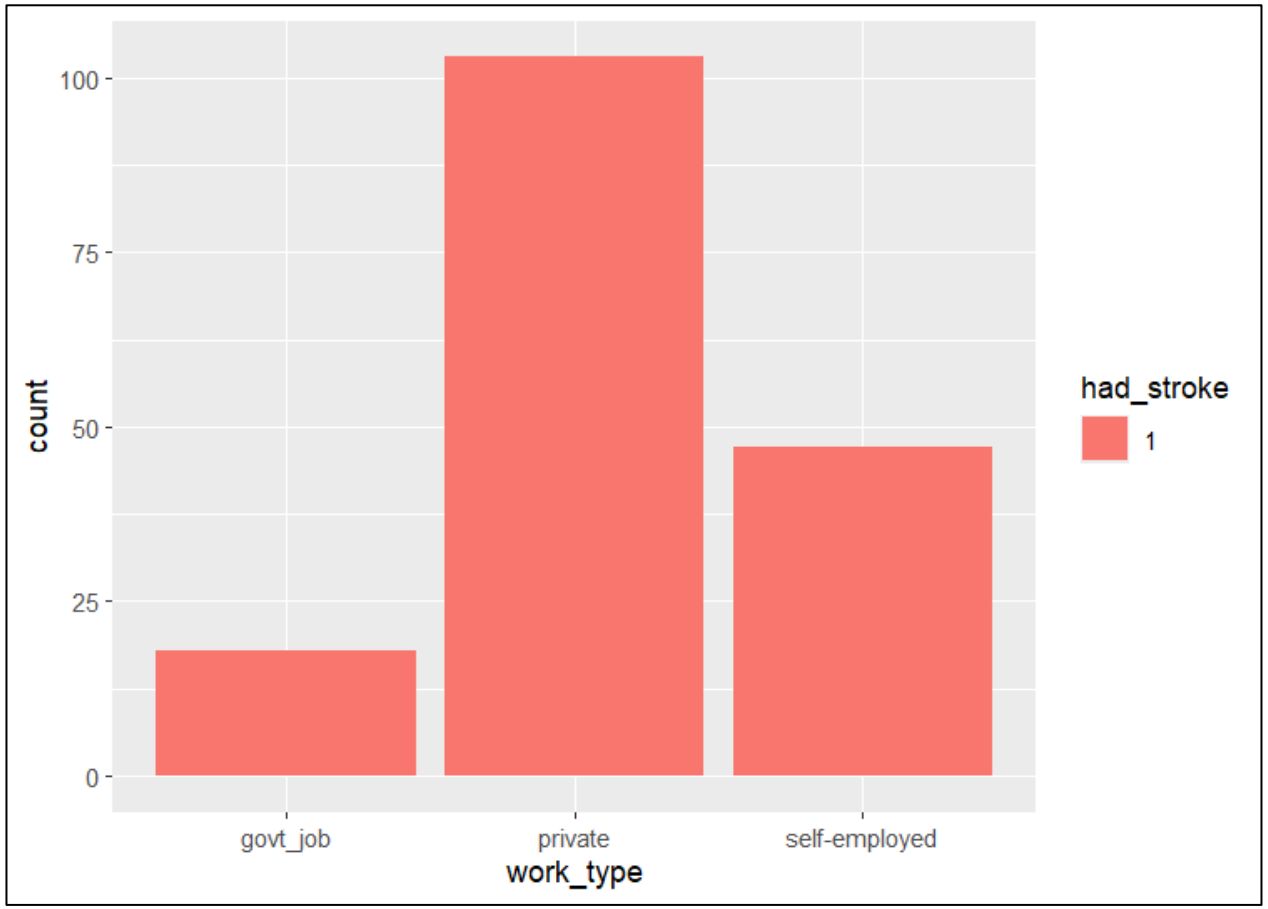
Demo Example – Correlation plot



Group Challenges

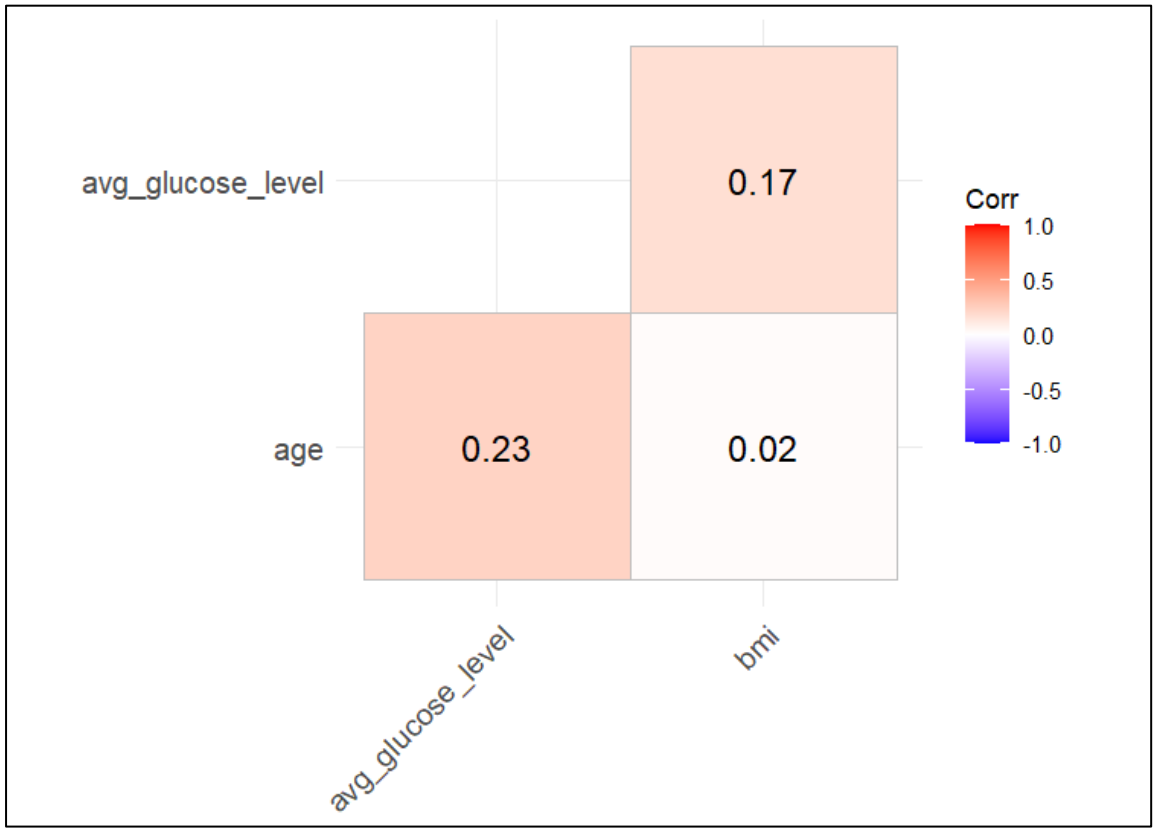
1. Use what we have learnt so far and any additional help documentation (or even searching online for tips) to generate the next 2 plots within the set time limit
2. Step 1: Load the **{tidyverse}** package and then load the stroke data
3. Step 2: Aim to prepare the dataset before you create the plot

Group Challenge 1 (10min)



Hint: Look at which variables are being used. Is there a new variable created? Does a variable type need to change?

Group Challenge 2 (10min)



Hint: Are missing values causing an issue?

QUIZ CHALLENGE

Get ready!

https://ahaslides.com/STATSINR1

POPULATIONS, SAMPLES AND RANDOM VARIABLES

Objectives

1. Learn about **populations** and **samples**
2. Learn about **random variables**

Populations vs Samples

We have seen in a previous section how to calculate summary statistics and visualise the **central tendency** or **spread** of a dataset.

When it comes to describing the **shape** and **distribution**, we first need to think about where our data is coming from and what it represents!

This is extremely important as it forms the basis for many statistical analyses, statistical modelling, statistical testing and research studies.

Populations vs Samples

- **Population –** In statistics, a "population" refers to the entire group of individuals, objects, or data points that are under study.

For example:

Let’s assume we want to understand and analyse the average income of all adults in a country. The population dataset would consist of every adult in that country.

One of the main disadvantages of collecting population data involves practical reasons such as cost and timings.

Populations vs Samples

- **Sample –** In statistics, a “sample" refers to a subset of the population that you select for study.

For example:

Using the same example as before. We can sample, 10 000 adults from the population to form our sample dataset and then use it to make inferences about the entire adult population's income.

The quality of the sample and the method of selection are crucial because they influence the validity of our study's findings.

Random Variables

In statistics, a **random variable** is a way to represent the uncertain outcome of a **random process**. It can take on various values with associated probabilities, allowing statisticians to model and analyse uncertainty in data.

- **Discrete Random Variable:** A discrete random variable is one that can take on a countable number of distinct values.

  For example: The number of heads when flipping a coin 5 times

- **Continuous Random Variable:** A continuous random variable is one that can take on any value within a specified range.

  For example: The height of individuals in a population

Group Exercise

1. How would you ensure that a sample is representative of a population?
2. Give an example of a situation where it would be impractical to study the entire population.
3. Is it discrete or continuous random variable?
   - The number of customers entering a store in a given hour
   - The time it takes for a car to travel a certain distance

STATISTICAL DISTRIBUTIONS

Objectives

1. Explore statistical distributions in R such as the **Normal**, **Poisson** and **Binomial** distributions
2. Learn how to generate **random numbers** from statistical distributions
3. Explore **visualisations** for statistical distributions
4. Learn about **approximations** of statistical distributions
5. Learn about **sampling distributions**

Empirical vs Theoretical distributions

We describe how the values of a random variable are distributed using a **statistical distribution**, also known as a probability distribution. We consider two types of statistical distributions:

- **Empirical Distribution –** This distribution is based on the actual observed data and represents the way the data is distributed
- **Theoretical Distribution –** This distribution is a mathematical model that describes how data should be distributed based on certain assumptions or properties

Random variables and distributions

Random variables and distributions are connected because the **distribution of a random variable** tells you how its values are spread out and how likely different values are.

For example: Let's assume that X is a random variable representing the number of "Heads" (outcome) you get when tossing a coin three times. You can use a statistical distribution to describe the probabilities of getting 0, 1, 2, or 3 "Heads".

Empirical vs Theoretical distributions

It is useful to highlight the differences in terms of representation and use between empirical and theoretical distributions:

- **Representation:** Empirical distributions provide a visual representation, while theoretical distributions provide a mathematical description of how data should be distributed
- **Use:** Empirical distributions describe the data at hand and are used for data exploration and understanding, while theoretical distributions are used for making predictions, statistical inference, and hypothesis testing

Empirical vs Theoretical distributions

As we discussed previously, very often the dataset we have available for study represents a sample of a wider population. It is common for the underlying assumption to be that the empirical distribution of our sample dataset is **“close enough”** to a theoretical distribution.

If this is true, then we can use the theoretical distribution to **make inferences** about the wider population. We now turn our focus on three distributions that are most widely used: a) **Normal** b) **Poisson** c) **Binomial**.

Each one is represented by a **mathematical function** that is defined by one or more **parameters**.

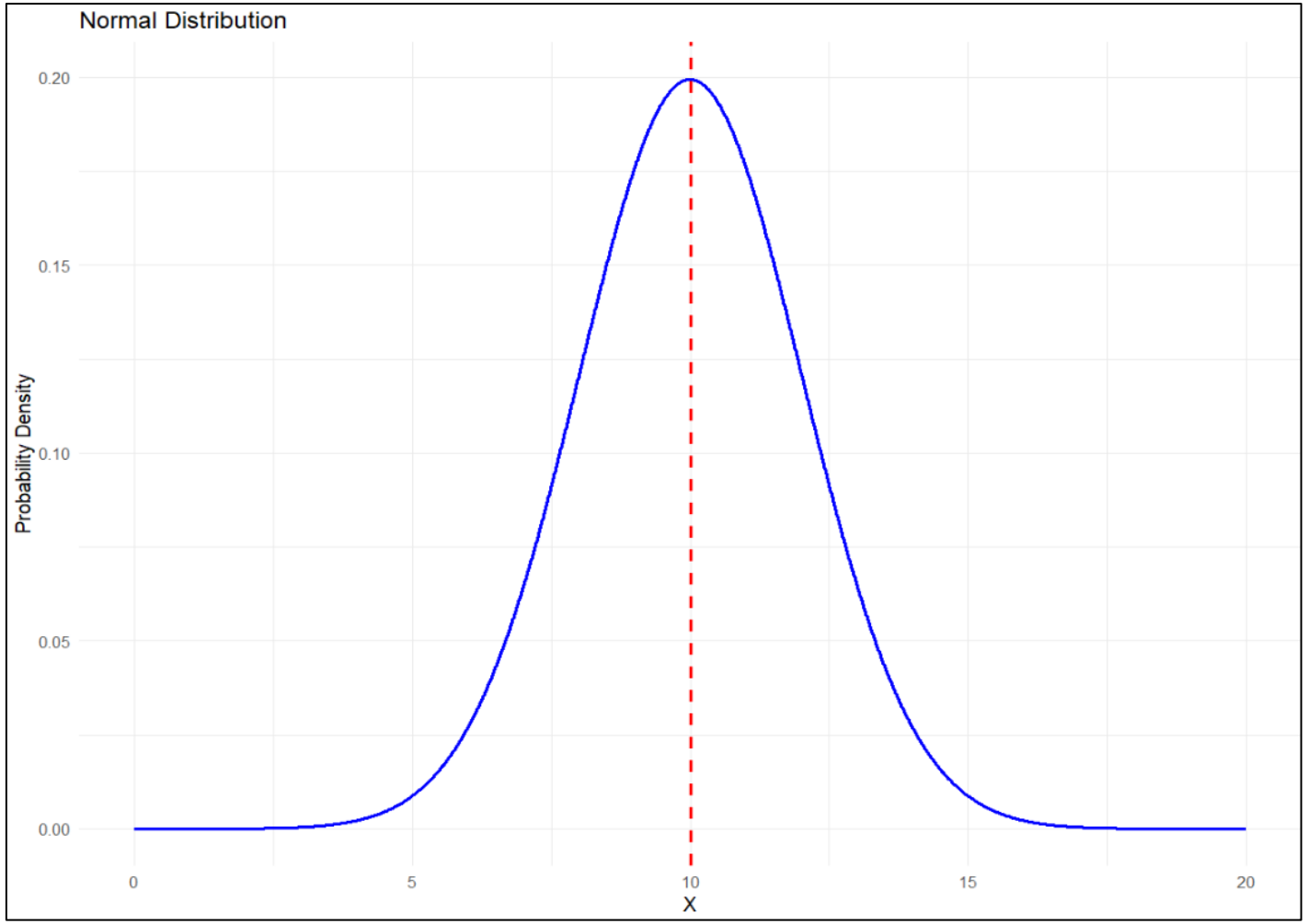
Normal distribution

The **Normal distribution** is one of the most widely used probability distributions in statistics. When plotted, this distribution has a **bell-shape curve** that is **symmetric** around its mean.

The Normal distribution is particularly important because it naturally arises in many **real-world** phenomena (such as height, stock market returns etc.) and is central to various statistical techniques and approximations.

The Normal distribution is sometimes referred to as the Gaussian distribution.

Normal distribution



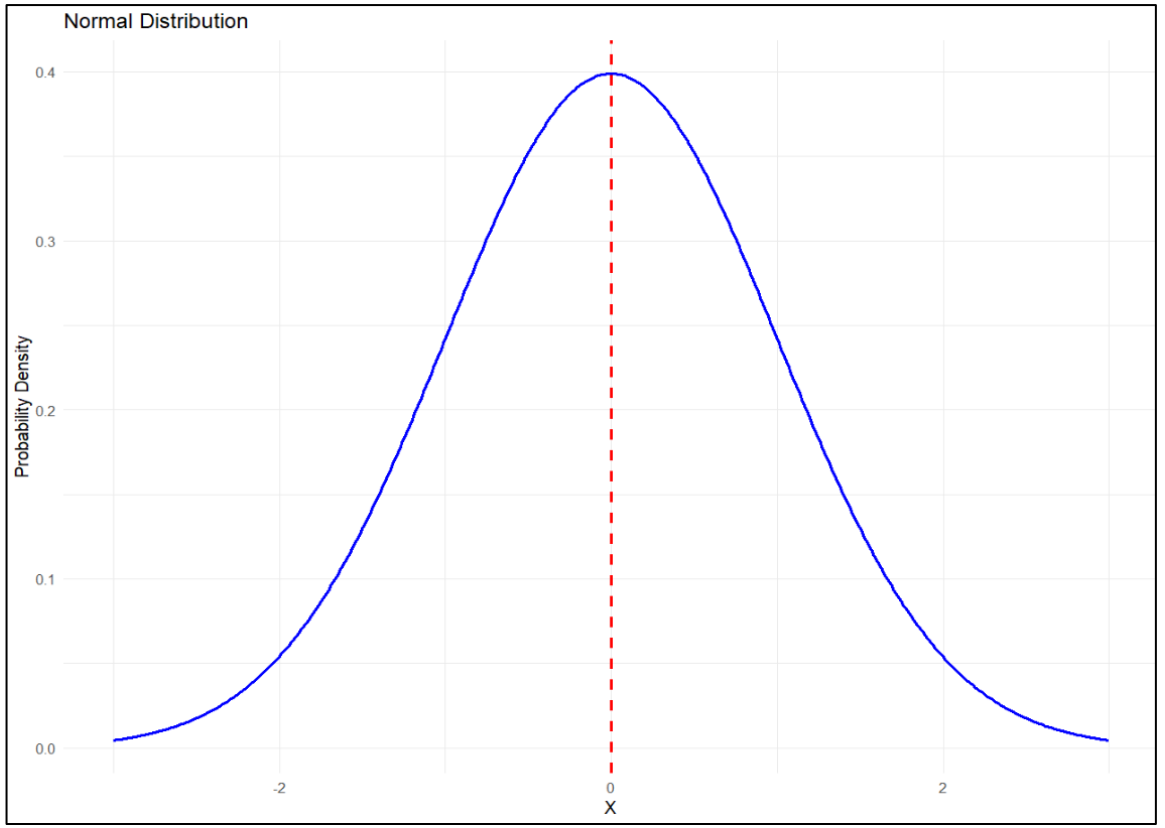
Normal distribution parameters

The **Normal distribution** is a theoretical distribution and can be described using a mathematical model with two parameters, namely its **mean (μ)** and its **standard deviation (σ)**. Let's assume that the random variable X is normally distributed then we denote this as: $X \sim \mathcal{N}(\mu, \sigma^2)$

Parameter	Description
Mean (μ)	The centre of the distribution indicating the highest point on the curve
Standard Deviation (σ)	A measure of how spread out the data is. **Note:** Smaller σ values result in narrower, taller curves, while larger σ values produce wider, flatter curves

The Standard Normal distribution

The **Standard Normal Distribution**, also known as the Z-distribution is a special case of the Normal distribution with a **mean of 0** and a **standard deviation of 1**. This is denoted as $X \sim \mathcal{N}(0,1)$. It is an important distribution because it is used as a reference distribution for many statistical tests.



Random numbers from a Normal distribution

Did you know that we can **simulate random numbers** from a statistical distribution in R?

We can use the function `rnorm()` from the **{stats}** package to generate random numbers from a Normal distribution. We need to specify the parameters, i.e. the mean and standard deviation.

```
# Generate 10 random Normal distribution variables
# from a Normal distribution with mean = 10 and standard deviation = 2
random_normal <- rnorm(10, mean = 10, sd = 2)

print(random_normal)
```

Random numbers from a Normal distribution

When a programming step in R involves **randomisation**, it is good practice to **set the random seed** using the function `set.seed()`. This ensures that your random process is in fact **reproducible**.

The example below shows how to create a reproducible output from a simulation of 10 numbers coming from the Normal Distribution.

```
set.seed(42) # set the random seed for reproducibility
random_normal <- rnorm(10, mean = 10, sd = 2)

print(random_normal)
```

Exercise 18 (10 min)

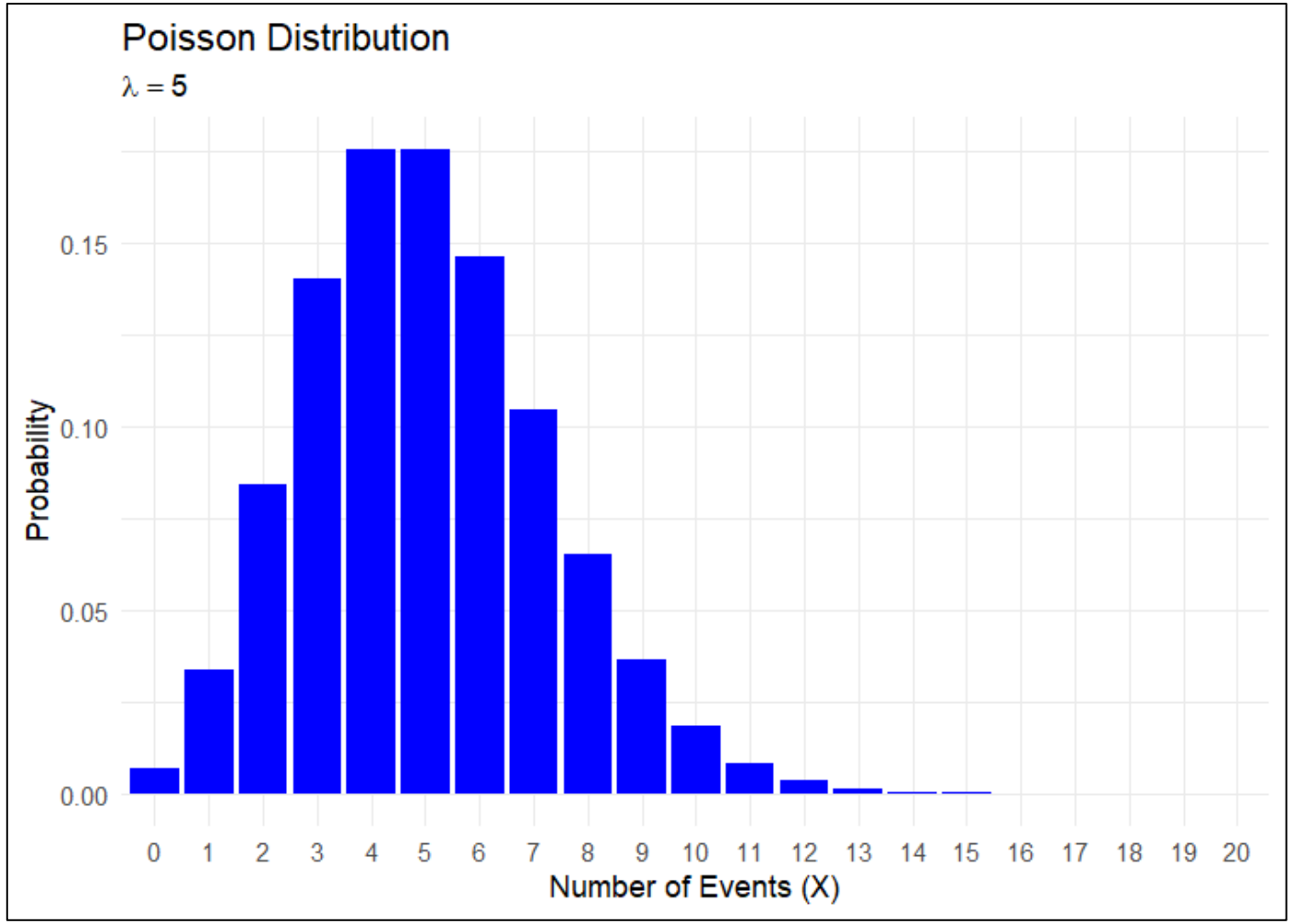
1. Simulate 10,000 random numbers from the Normal distribution that has mean = 0 and standard deviation = 1
2. Now make this sample reproducible by setting the seed
3. What visualisation do you think is appropriate to represent the Normal distribution?
4. Use **{ggplot2}** and what you have learned earlier in the course to visualise the 10,000 Standard Normal distribution random numbers

Poisson Distribution

Another widely used probability distribution in statistics is called the **Poisson distribution**.

This is a **discrete** probability distribution that models the **number of events** that occur within a fixed interval of time when these events happen at a constant average rate and independently of the time since the last event.

Poisson Distribution



Poisson distribution parameter

The Poisson distribution is described by one parameter: λ **(lambda)**. This is also equal to the mean and variance of the distribution.

Let's assume that the random variable X has a Poisson distribution then we denote this as: $X \sim Pois(\lambda)$

Parameter	Description
λ **(lambda)**	Represents the average rate of events occurring within the given interval. λ is also equal to the mean and variance of the distribution.

Random numbers from a Poisson distribution

We can use the function `rpois()` from the **{stats}** package to generate random numbers from a Poisson distribution. We need to specify the distribution parameter λ (lambda).

```
# Generate 10 random Poisson distribution variables
# from a Poisson distribution with lambda = 5

set.seed(42) # set the random seed for reproducibility
random_poisson <- rpois(10, lambda = 5)

print(random_poisson)
```

Exercise 19 (10 min)

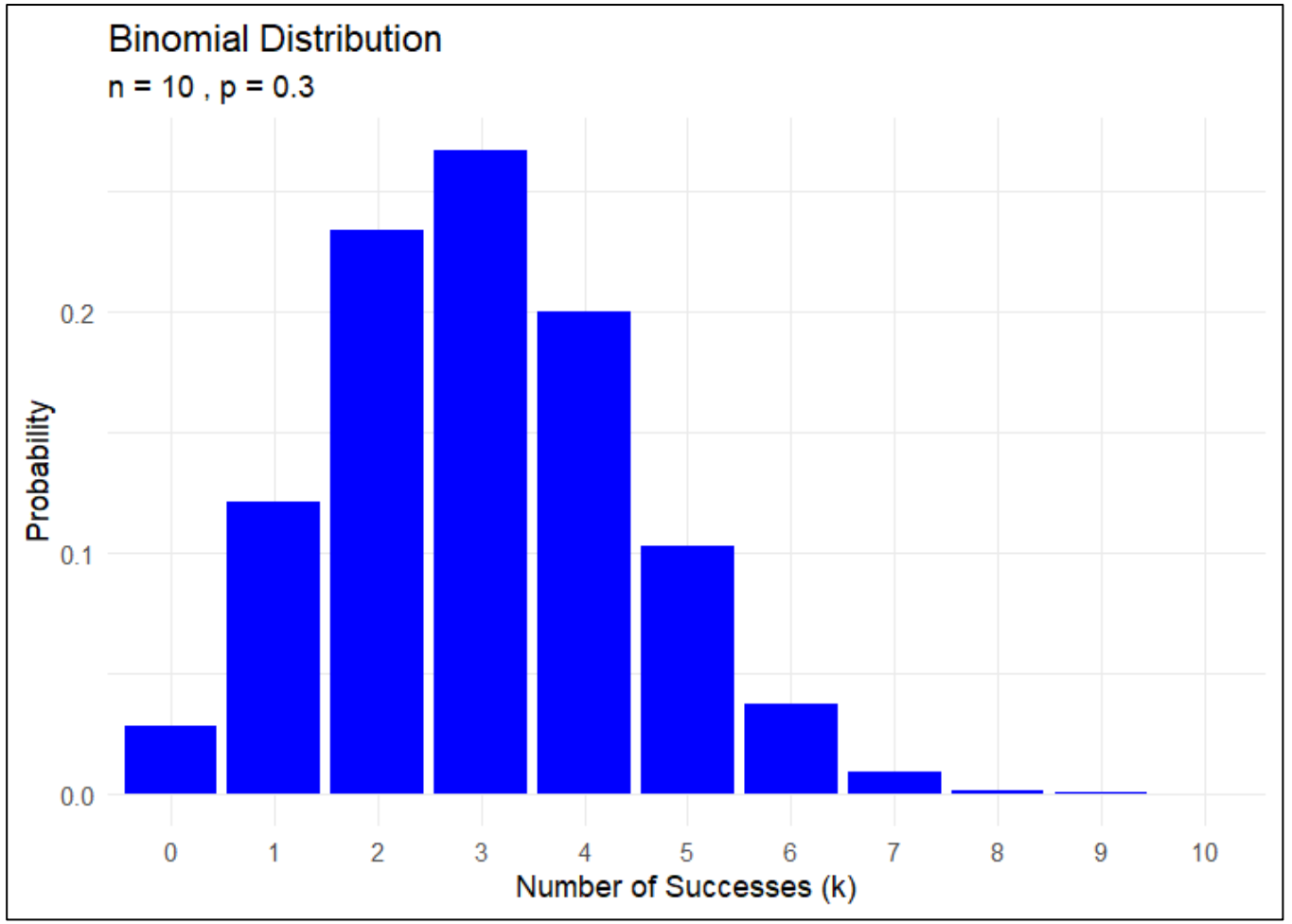
1. Create a reproducible example of 1,000 Poisson distribution random numbers with lambda = 2
2. What visualisation do you think is appropriate to represent this distribution?
3. Use **{ggplot2}** and what you have learned earlier in the course to visualise the 1,000 Poisson distribution random numbers (Hint: factor)

Binomial distribution

Another widely used probability distribution in statistics is called the **Binomial distribution**.

Like the Poisson distribution, this is also a **discrete** probability distribution that models the **number of successes** in a fixed number of independent and identical experiments (known as Bernoulli trials). Each trial can result in one of two possible outcomes: **success** or **failure**.

Binomial distribution



Binomial distribution parameters

The Binomial distribution is described by two parameters, namely **n** (size) and **p** (probability of success). The mean of this distribution can be calculated by $\mathrm{n} \times \mathrm{p}$ and the variance by $\mathrm{n} \times \mathrm{p} \times (1 - \mathrm{p})$.

Let's assume that the random variable X has a Binomial distribution then we denote this as: $\mathrm{X} \sim B(n, p)$

Parameter	Description
n	The total number of trials or experiments (size)
p	The probability of success on each trial (prob)

Random numbers from a Binomial distribution

Similarly to the other distributions that we have seen so far, we can use the function **`rbinom()`** from the **{stats}** package to generate random numbers from a Binomial distribution. We need to specify the distribution parameters **n** and **p**.

```
# Generate 10 random Binomial distribution variables
# from a Binomial distribution with size = 10 and p = 0.5

set.seed(42) # set the random seed for reproducibility
random_binomial <- rbinom(10, size = 10, prob = 0.5)

print(random_binomial)
```

This is the number of random numbers to generate

Exercise 20 (5 min)

1. Create a reproducible example of 1,000 Binomial distribution random numbers of size 10 and p = 0.1
2. What visualisation do you think is appropriate to represent this distribution?
3. Use **{ggplot2}** and what you have learned earlier in the course to visualise the 1,000 Binomial distribution random numbers

Statistical Distribution approximations

It is worth pointing out that both the Poisson and the Binomial distributions could be **approximated** to a Normal distribution.

The Poisson distribution can be approximated by a Normal distribution when the **mean (λ) is sufficiently large**, typically greater than or equal to 20.

As the value of λ increases, the Poisson distribution becomes more symmetric and bell-shaped, resembling the Normal distribution.

It is advisable to check that the conditions are met for a distribution to be approximated to the Normal distribution

Statistical Distribution approximations

The Binomial distribution can also be approximated by a Normal distribution when the **sample size (n) is sufficiently large**.

A common rule of thumb is that both n times the probability of success ($n \times p$) and n times the probability of failure ($n \times (1 - p)$) should be greater than or equal to 5.

In both the Poisson and the Binomial approximations to the Normal distribution, a **continuity correction** is often applied as we are approximating a discrete to a continuous distribution.

Exercise 21 (10 min)

1. Repeat the exercise from the random values from a Poisson distribution, trying different values for lambda with the aim of visualising the approximation to a Normal distribution
2. Repeat the exercise from the random values from a Binomial distribution, trying different values for n and p with the aim of visualising the approximation to a Normal distribution

Sampling distributions

A **sampling distribution** is a concept in statistics that represents the distribution of a statistic (such as the mean or proportion). This distribution is calculated from **multiple random samples** drawn from the same population and helps us understand the properties of the statistic.

Importance: Sampling distributions are critical for making inferences about population parameters, constructing confidence intervals, and performing hypothesis tests.

Central Limit Theorem

The **Central Limit Theorem (CLT)** states that if you have a sufficiently large sample from a population (typically greater than or equal to 30) then the **sampling distribution of the mean** will follow an approximately Normal distribution even if the population isn't normally distributed.

This is denoted by $\overline{X} \approx \mathcal{N}(\mu, \frac{\sigma^2}{n})$ Note: $\frac{\sigma}{\sqrt{n}}$ is called the **standard error**.

Importance: We will see later that in fact the CLT is crucial for hypothesis testing, where we compare sample statistics to population parameters and assess the likelihood of our results occurring by chance.

n represents the sample size

Exercise 22 (15 min) – Group Exercise

1. Let's suppose that we asked 50 people to toss a coin 20 times and record the number of heads they observed.
2. Let's suppose that we asked another 50 people to repeat the same experiment and so on until we repeated this exercise a total of 1000 times.
3. What do we consider as 1 sample? What is that sample size?
4. According to CLT, what is the underlying theoretical distribution and its parameters?
5. Write the code to carry out this experiment to show that the CLT applies for the sample mean.
6. Plot a histogram to visualise the outcome… does it look like a Normal distribution with the parameters you expected?

CONFIDENCE INTERVALS

Objectives

1. Learn about **confidence intervals**
2. Learn how to **construct** confidence intervals in R
3. Learn how to **interpret** confidence intervals and what factors affect them

Confidence Intervals

- **Definition:** A **Confidence Interval** (CI) is a statistical **range** that provides an estimate of the true **population parameter** and is based on a sample of data. It quantifies the uncertainty surrounding our point estimate
- **Importance:** CIs are essential for making informed decisions in statistics and data analysis. They provide a way to express the precision and reliability of our estimates

Confidence Level of a confidence interval

The **confidence level** in a CI represents the probability that the true population parameter falls within the CI. Commonly used confidence levels include **90%**, **95%**, and **99%**.

For example: Let's suppose that you want to construct a **95% Confidence Interval** for a population parameter, such as the population mean.

This means that if you were to take many random samples from the same population and calculate an interval for each sample for the unknown value, you would expect approximately 95% of these intervals to contain the true population parameter.

Constructing a Confidence Interval

Constructing a Confidence Interval is quite straightforward. In fact, in its simplest form the formula for a CI is:

Point Estimate ± Margin of Error

Let's assume that we are interested in constructing the 95% CI for the population mean (μ). This means that our margin of error is 2.5% on either side of the interval. Given a sample of data, we can use the sampling distribution of the mean and the approximated **Normal distribution** (from CLT) to identify the **critical value** for evaluating the margin of error.

Constructing a Confidence Interval

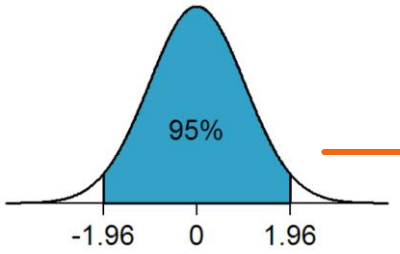
We can describe the construction of the 95% Confidence Interval as:

The point estimate of the mean from the sample data

$$(\bar{x} - 1.96 \times \frac{\sigma}{\sqrt{n}}, \bar{x} + 1.96 \times \frac{\sigma}{\sqrt{n}})$$

The standard deviation of the population – however this is often unknown and can be replaced by the sample standard deviation

Remember:
We need sufficiently large (typically n ≥ 30) sample sizes!



The critical values come from the standard Normal distribution and reflect the 2.5% and 97.5% percentiles on the distribution

The sample size

A note on the critical value

We can use the `qnorm()` function from the **{stats}** package to get the critical value from the Standard Normal distribution needed for creating the CI. However, this is true when the **population variance** is **known**.

When the population variance is unknown – as in most cases where we have a sample - we must use a similar distribution, namely, the **Student's t-distribution** with **n-1 degrees of freedom** to determine the critical value.

A note on the critical value

We can use the `qt()` function from the **{stats}** package to get the critical value from the **Student's t-distribution**.

It is important to note that as n (the sample size) gets larger the two distributions become practically identical. You will often see cases when **n > 30** that the Normal distribution is used for the critical value giving a similar outcome. We will examine both cases however strictly speaking when the population variance is unknown it is more accurate to use the Student's t-distribution for the critical value.

Demo Example

1. Use the wine dataset to calculate the 95% Confidence Interval for the column `volatile_acidity`
2. I have intentionally made an error in this demonstration! Can you think what this error is? Hint: Wine colour…
3. Why is this error important that could cause my CI to be unreliable?

Demo Example

```r
library(tidyverse)

wine <- read_csv("data/wine.csv")

sample_mean <- mean(wine$volatile_acidity) # sample mean
sample_n <- length(wine$volatile_acidity) #  sample size
sample_sd <- sd(wine$volatile_acidity) # sample standard deviation
sample_se <- sample_sd/sqrt(sample_n) # sample standard error
confidence_level <- 0.95 # confidence level
critical_value <- qnorm((1-confidence_level)/2, lower.tail = FALSE) # critical value
margin_error <- critical_value*sample_se # margin of error

lower_bound <- sample_mean - margin_error # CI lower bound
upper_bound <- sample_mean + margin_error # CI upper bound

print(c(lower_bound, upper_bound))
```

Demo Example

```
critical_value <- qnorm((1-confidence_level)/2, lower.tail = FALSE) # critical value
```

The above will return a value of 1.959964

```
critical_value <- qt((1-confidence_level)/2, lower.tail = FALSE) # critical value
```

The above will return a value of 1.96036

Interpreting Confidence Intervals

Let's assume that we have now constructed the 95% CI for the population mean (μ). How would we **interpret** this result?

In statistics, a confidence interval is ***informally*** described as a range of values that we expect to include the parameter being estimated.

More on Confidence Intervals

Width of a Confidence Interval: The width of a confidence interval depends on the **sample size** and the chosen **confidence level**

$$(\bar{x} - 1.96 \times \frac{\sigma}{\sqrt{n}}, \bar{x} + 1.96 \times \frac{\sigma}{\sqrt{n}})$$

A larger sample size leads to a narrower CI and vice versa!

A higher Confidence Level leads to a larger critical value which then leads to a wider CI and vice versa!

Group Exercise 1 (10min)

1. Use the `avg_glucose_level` variable from the stroke dataset to construct a 99% Confidence Interval for the population mean.
2. How would you interpret your results?

Group Exercise 2 (20min)

1. We have seen so far how to construct the Confidence Interval for the population mean. Find the formula that constructs the Confidence Interval of a **proportion**
2. Use the stroke dataset and calculate the 95% CI for the proportion of stroke events for patients that have hypertension and separately for those that do not have hypertension
3. Are there any interesting results?

QUIZ CHALLENGE

Get ready!

https://ahaslides.com/STATSINR2

WINE DATA

Demo Data

Another dataset that we use during the course is:

Red and White wine quality of the Portuguese "Vinho Verde“

Physicochemical test results (such as PH) and quality assessment graded by experts - 0 (very bad) and 10 (very excellent)

DATASET CITATION: P. Cortez, A. Cerdeira, F. Almeida, T. Matos and J. Reis. Modeling wine preferences by data mining from physicochemical properties. In Decision Support Systems, Elsevier, 47(4):547-553. ISSN: 0167-9236. https://scinapse.io/papers/2103459159

Demo/Couse Data Dictionary

- **wine type**: specifies if the wines is red or white
- **fixed acidity**: the amount of fixed or non-volatile acids (do not evaporate readily)
- **volatile acidity**: the amount of acetic acid (high levels can lead to vinegar like taste)
- **citric acid**: amount of citric acid (can add ‘freshness’ and flavour to wines)
- **residual sugar**: the amount of sugar remaining after fermentation stops (the higher the amount the sweeter the wine)
- **chlorides**: the amount of salt in the wine
- **free sulfur dioxide**: the amount of free form of SO_2 that exists (it prevents microbial growth and the oxidation of wine)

Demo/Couse Data Dictionary

- **total sulfur dioxide**: amount of free and bound forms of SO_2
- **density**: the wine density (depends on the alcohol and sugar content)
- **pH**: shows how acidic or basic a wine is on a scale from 0 (very acidic) to 14 (very basic)
- **sulphates**: a wine additive which contributes to SO_2 levels (act as antimicrobial and antioxidant)
- **alcohol**: the alcohol content of the wine (percent)
- **quality**: wine quality score on a scale from 0 (very bad) to 10 (very excellent)

STROKE DATA

Course Data

Another dataset that we use during the course is:

Health dataset for the analysis of brain strokes. The dataset contains patient demographic and medical history data

DATASET CITATION: Kaggle: https://www.kaggle.com/datasets/prosperchuks/health-dataset?select=stroke_data.csv

Couse Data Dictionary

- **id –** unique identifier of patient
- **gender –** “male”, “female“
- **age –** age of the patient
- **hypertension –** 0 if the patient doesn't have hypertension, 1 if the patient has hypertension
- **heart_disease –** 0 if the patient doesn't have any heart diseases, 1 if the patient has a heart disease
- **ever_married –** “no” or “yes”

Couse Data Dictionary

- **work_type –** “govt_job”, “private” or “self-employed”
- **residence_type –** “rural” or “urban”
- **avg_glucose_level –** average glucose level in blood
- **bmi –** body mass index
- **smoking_status –** "formerly smoked”, “never smoked” or “smokes”
- **stroke –** 1 if the patient had a stroke or 0 if not

DEFINING A STATISTICAL TEST

Objectives

1. Learn about **hypothesis testing** and how to **define** a test
2. Learn about the **Null** and the **Alternative** Hypotheses
3. Learn about the **significance level**
4. Learn about the **p-value**
5. Learn about the **power** of the test

Hypothesis testing

Hypothesis testing is an important statistical methodology that helps us determine whether any observed differences (from our sample) are **statistically significant** from the underlying population measure or simply due to **randomness**.

For example: There is a proposal that the mean of a variable has a certain value. We can use a hypothesis test to determine whether this is consistent with what we have observed in our sample.

Null vs Alternative hypotheses

The first step in performing a hypothesis test is by formulating two hypotheses:

- **Null hypothesis (H_0)** – This represents the underlying assumption about the population parameter. For example:

  $H_0: \mu = \mu_0$ → "The population mean (μ) is equal to a specific value (μ_0)"

- **Alternative hypothesis (H_1)** – This represents a statement that contradicts the null hypothesis. For example:

  $H_1: \mu \neq \mu_0$ → "The population mean (μ) is **not** equal to a specific value (μ_0)"

Significance Level

Let' assume that we have established the Null and the Alternative hypotheses. We've also collected our sample data and calculated the measure we want to test (such as the mean).

- We need to specify a certain **threshold** by which we can decide as to which hypothesis our sample data supports.

This threshold is called the **significance level** (usually denoted as α). It represents the maximum acceptable probability of rejecting the Null hypothesis when in fact it was true (also known as **Type I** error).

Common choices for the significance level are 0.05 (5%), 0.01 (1%), and 0.10 (10%)

Hypothesis testing - continued

The definition of the Alternative hypothesis is actually quite important! We saw earlier the example of:

$H_1: \mu \neq \mu_0$ → "The population mean (μ) is **not** equal to a specific value (μ_0)"

This would be categorised as a **two-sided** (or two-tailed) test since we are interested to find out whether μ is either larger or smaller than the specific value (μ_0). The other two options for Alternative hypotheses are:

$H_1: \mu > \mu_0$ or $H_1: \mu < \mu_0$ in which case we would categorise as **one-sided** (or one-tailed) tests.

In the case of a two-sided test, we use half the significance level ($\frac{\alpha}{2}$) as the decision rule threshold

Type I and Type II errors

- **Type I Error** – this error occurs when you **reject the Null** hypothesis when it is **actually true**. The probability of committing a Type I error is denoted as α and is the significance level you set for your hypothesis test
- **Type II Error** – this error occurs when you **fail to reject the Null** hypothesis when it is **actually false**. The probability of committing a Type II error is denoted as β

There is a **trade-off** between Type I and Type II errors. A lower α reduces the chance of Type I error but increases the chance of Type II error, and vice versa.

p-value

The p-value is a quantitative measure of the strength of **evidence against the Null** hypothesis. Assuming the Null hypothesis is true, the p-value quantifies how **unusual** or **extreme** the observed data is.

- A **small p-value** (typically $< \alpha$) suggests **strong evidence against** the Null hypothesis and supports the Alternative hypothesis
- A **large p-value** (typically $> \alpha$) suggests **weak evidence against** the Null hypothesis and fails to support the Alternative hypothesis

Power of the Test

The power of a statistical test, represented as 1- β, measures the test's **ability to correctly reject a Null hypothesis** when it is actually false.

- A **high power** indicates a high probability of correctly rejecting a false Null hypothesis (lowering the risk of a Type II error)
- A **low power** indicates a high probability of failing to reject a false Null hypothesis (increasing the risk of a Type II error)

The power of a test depends on various factors such as: the significance level, the sample size, the effect size (the magnitude of the difference you're trying to detect) and the variability of the data (standard deviation).

STATISTICAL TESTS
Part 1

Objectives

1. Learn what is a **test statistic**
2. Learn how to perform a **one-sample t-test**
3. Learn how to perform a **two-sample t-test**
4. Learn how to perform a **paired t-test**
5. Learn how to perform a **proportion test**

What is a test statistic

A test statistic is a numerical value that we calculate from the sample data when performing a statistical hypothesis test. It **provides a way to quantify the evidence** against the Null hypothesis and decide whether to reject it or not.

A key property of a test statistic is that, under the Null hypothesis, its sampling distribution must be calculable.

Test statistic for the mean (one-sample)

We know from the Central Limit Theorem that the sampling distribution for the mean is approximately distributed as: $\mathcal{N}(\mu, \frac{\sigma^2}{n})$.

For a one-sample test of size (n) and a specific value (μ_0) for the mean, we can formulate the test statistic for the mean as: $\mathrm{z} = \frac{\bar{x} - \mu_0}{(\sigma/\sqrt{n})}$ where z comes from a **standard Normal distribution**. However, as the population variance is unknown, we can replace σ^2 with the sample variance s^2 but in doing so we need to use the **t-distribution** and therefore the test statistic for the mean becomes:

$$t = \frac{\bar{x} - \mu_0}{(s/\sqrt{n})}$$ with n-1 degrees of freedom

The `t.test()` function in R

When carrying out hypothesis testing in R, one of the most useful functions that we can use from the **{stats}** package is **`t.test()`**. This function allows us to perform several of the tests that we cover in the following sections!

t.test {stats} R Documentation

Student's t-Test

Description

Performs one and two sample t-tests on vectors of data.

Usage

```
t.test(x, ...)

## Default S3 method:
t.test(x, y = NULL,
       alternative = c("two.sided", "less", "greater"),
       mu = 0, paired = FALSE, var.equal = FALSE,
       conf.level = 0.95, ...)

## S3 method for class 'formula'
t.test(formula, data, subset, na.action, ...)
```

Note: The 3 dots (`...`) in R are called ellipsis and allow you to pass other named or unnamed arguments to a function

Demo Example

Using the wine dataset:

1. Let's assume that the white wine manufacturer claims that the mean for the alcohol content of their wine is 10.5%
2. Perform one sample t-test to test this hypothesis using a 5% significance level (two-sided)

Demo Example

```
library(tidyverse)

wine <- read_csv("data/wine.csv")

white_wine <- wine %>%
  filter(wine_type == "white") # Keep only the white wine rows

# Two-sided t-test result
result <- t.test(x = white_wine$alcohol,
                 mu = 10.5)

# View result
print(result)
```

Demo Example

```
> print(result)

        One Sample t-test

data:  white_wine$alcohol
t = 0.853, df = 4499, p-value = 0.3937
alternative hypothesis: true mean is not equal to 10.5
95 percent confidence interval:
 10.47967 10.55165
sample estimates:
mean of x
 10.51566
```

degrees of freedom

p-value to compare against significance level

test statistic

alternative hypothesis

95% confidence interval

value of point estimate (mean)

Exercise 23 (5 min)

1. Use the stroke dataset and the column `avg_glucose_level` to perform a one-sample t-test where the Null hypothesis is that the population mean is 100 mg/dL against the Alternative hypothesis that the mean is not equal to this value.
2. How would you interpret your results?

Note: Use the `drop_na()` function to remove any rows that have missing values.

One sample t-test for the mean

It is worth noting that the `t.test()` function allows us to modify the hypothesis test that we want to perform. For example, the argument `alternative` enables us to do a one- or two-sided test.

```
t.test(x = white_wine$alcohol,
       alternative = "greater",
       mu = 10.5)
```

The argument `conf.level` enables us to specify the confidence level of the Confidence Interval that is also evaluated.

```
t.test(x = white_wine$alcohol,
       mu = 10.5,
       conf.level = 0.90)
```

Exercise 24 (5 min)

1. Repeat the previous exercise and change some of the function arguments to see what the output returns.
2. Let's discuss the “strange” output when you use the “less” option… (does the test make sense … should we think of 1 minus the p value?

Test statistic for the mean (two sample)

Let's assume that we have **two independent samples** and we want to test whether the population **means are equal**.

- For the sake of simplicity, we won't cover the formula behind the test statistic for this two-sample t-test. We can simply use the `t.test()` function that we saw earlier.

However, it is important to highlight one **additional condition** when performing such a test… we need to specify whether the two populations have **equal variance or not**!

Demo Example

Using the wine dataset:

1. Let's assume that the wine manufacturer claims that the mean alcohol content for the white wine is equal to the red wine but some experts want to test whether the white wine is greater than the red
2. It is believed that the variance of the alcohol content of the two wines are the same, test the claim of the manufacturer

Demo Example

```r
library(tidyverse)

wine <- read_csv("data/wine.csv")

white_wine <- wine %>%
  filter(wine_type == "white") # Keep only the white wine rows

red_wine <- wine %>%
  filter(wine_type == "red") # Keep only the white wine rows

result <- t.test(x = white_wine$alcohol, y = red_wine$alcohol,
                 alternative = "greater",
                 var.equal = TRUE)

print(result)
```

Demo Example

```
> print(result)

        Two Sample t-test

data:  white_wine$alcohol and red_wine$alcohol
t = 2.751, df = 5998, p-value = 0.00298
alternative hypothesis: true difference in means is greater than 0
95 percent confidence interval:
 0.03924408        Inf
sample estimates:
mean of x mean of y
 10.51566  10.41803
```

Alternative hypothesis: x is greater than y

One-sided confidence interval

values of point estimates (mean)

Exercise 25 (5 min)

1. Use the stroke dataset to test whether the mean of the `avg_glucose_level` for the two genders is different (you can assume equal variances)

Paired t-test

A **paired t-test** is a statistical hypothesis test used to compare the mean of a variable from two sets of **data** that are **related**, i.e. taken from the same individual, object, or related unit.

For example:

Let's assume that you are conducting a medical study to determine whether a new medication has a statistically significant effect of lowering a patient's blood pressure. A paired t-test is needed to compare patients' blood pressure **before** and **after** they receive the new medication.

Paired Data

The dataset that we use for demonstrating a paired t-test is:

Reaction times (in seconds) of a task before and after sleep deprivation for a group of patients

Demo Example

Use the before/after dataset to test whether there is any statistically significant difference (at 5% significance level) between the means of the patients' reaction times

Demo Example

```
library(tidyverse)

before_after <- read_csv("data/before_after.csv")

result <- t.test(x = before_after$before,
                 y = before_after$after,
                 paired = TRUE) # This argument is needed for paired test

print(result)
```

Demo Example

```
> print(result)

        Paired t-test

data:  before_after$before and before_after$after
t = -3.1056, df = 29, p-value = 0.004218
alternative hypothesis: true mean difference is not equal to 0
95 percent confidence interval:
 -0.7286605 -0.1500062
sample estimates:
mean difference
     -0.4393333
```

Alternative hypothesis: mean of differences is not equal to 0

Mean of differences

Think that the paired difference is calculated as: x - y

Exercise 26 (5 min)

1. Repeat the demonstrated paired t-test example but this time specifying the alternative hypothesis that you would expect.
2. How do you interpret your results?

Proportion test

When we have categorical data, we are often more interested in understanding **proportions**. For such scenarios that deal with **binary data** (yes/no, success/failure), we look to using a **proportion test**.

For example:

Let's say you want to know if the proportion of customers who buy a particular product (success) is different from the historical average (known proportion of successes) in your store.

Proportion test

The `prop.test()` function from the **{stats}** package can be used to test the Null hypothesis that the proportion (probability of success) of a sample is a specified value. Just like with the previous tests, we can specify whether we want to perform a one/two-sided test depending on how we specify the alternative hypothesis.

This function requires us to submit as arguments the number of successes, the number of trials and the specified value.

Demo Example

Using the wine dataset:

1. Let's assume that the white wine manufacturer claims that the proportion of bad quality white wines (4 or less) is 1%
2. Perform a one sample proportion test to test this hypothesis

Demo Example

```r
library(tidyverse)

wine <- read_csv("data/wine.csv")

white_wine <- wine %>%
  filter(wine_type == "white") # Keep only the white wine rows

successes <- sum(white_wine$quality <= 4) # Number of successes

total_trials <- nrow(white_wine) # Total number of trials

hypothesized_proportion <- 0.01 # Hypothesized proportion

# Perform the one-sample proportion test
result <- prop.test(x = successes,
                    n = total_trials,
                    p = hypothesized_proportion)

print(result)
```

Demo Example

```
> print(result)

        1-sample proportions test with continuity correction

data:  successes out of total_trials, null probability hypothesized_proportion
X-squared = 336.84, df = 1, p-value < 2.2e-16
alternative hypothesis: true p is not equal to 0.01
95 percent confidence interval:
 0.0320742 0.0433975
sample estimates:
          p
0.03733333
```

Point estimate of proportion

Alternative hypothesis:

proportion is not equal to 1%

Exercise 27 (5 min)

1. Use the stroke data and the results from the male patients to test the hypothesis that 1 in 20 of male patients have a stroke.
2. How do you interpret your results?

A note on assumptions...

It is important to consider what are the key **assumptions** that we make when performing these tests. In the following section we examine and test the following assumptions:

1. **Normality:** This is a necessary assumption when performing an unpaired (one/two sample) or paired t-test for the mean
2. **Equal Variance (Homoscedasticity):** This is a necessary assumption when performing a two-sample t-test

QUIZ CHALLENGE

Get ready!

https://ahaslides.com/STATSINR3

STATISTICAL TESTS
Part 2

Objectives

1. Learn how to perform a **one sample Kolmogorov-Smirnov (KS) test**
2. Learn how to perform a **two sample Kolmogorov-Smirnov (KS) test**
3. Learn how to perform a **Mann-Whitney U test**
4. Learn how to perform an **F-test**

The Kolmogorov-Smirnov (KS) Test

The **Kolmogorov-Smirnov (KS)** test is an important and versatile statistical test that helps us determine **goodness of fit** in two cases:

1. For **one sample** – we can test whether that sample follows a particular theoretical distribution, for example a Normal distribution. This is what is often called a **Normality test**
2. For **two** (independent) **samples** – we can test if they are drawn from the **same underlying distribution** (not necessarily from a Normal distribution)

The Kolmogorov-Smirnov (KS) Test

In summary, the KS test uses the **vertical distances** between:

- The empirical cumulative distribution function for one sample and the theoretical cumulative distribution function of the specified distribution
  - **Null hypothesis:** Sample follows the specified theoretical distribution
  - **Alternative hypothesis:** Sample does not follow the specified theoretical distribution
- The two empirical cumulative distribution functions of the two samples
  - **Null hypothesis:** Two samples come from the same underlying distribution
  - **Alternative hypothesis:** Two samples come from different distributions

The `ks.test()` function in R

The **{stats}** package makes it very easy for us to perform a Kolmogorov-Smirnov (KS) test.

We can use the function **`ks.test()`** for both one or two sample cases that we described earlier.

ks.test {stats} R Documentation

Kolmogorov-Smirnov Tests

Description

Perform a one- or two-sample Kolmogorov-Smirnov test.

Usage

```
ks.test(x, ...)
## Default S3 method:
ks.test(x, y, ...,
        alternative = c("two.sided", "less", "greater"),
        exact = NULL, simulate.p.value = FALSE, B = 2000)
## S3 method for class 'formula'
ks.test(formula, data, subset, na.action, ...)
```

The KS Test in R

Let's generate an artificial dataset coming from the Normal distribution.

```
set.seed(42)
random_normal <- rnorm(100, mean = 10, sd = 2)
```

Let's use the KS test to check for Normality for this dataset.

```
ks.test(random_normal, "pnorm",
        mean = 10, sd = 2)
```

Use the help functionality in R to find out more about **pnorm**

Demo Example

Using the wine dataset and only the white wine rows , test whether the `alcohol` variable is Normally distributed.

Note: If we do not specify the mean and standard deviation in the test, then the default Standard Normal distribution is assumed.

Demo Example

```r
white_wine <- wine %>%
  filter(wine_type == "white") # Keep only the white wine rows

# Testing for Normality with a specified mean and standard deviation
ks.test(white_wine$alcohol, "pnorm",
        mean = mean(white_wine$alcohol), sd = sd(white_wine$alcohol))

# Testing for Normality using the Standard Normal distribution
ks.test(white_wine$alcohol, "pnorm")
```

Demo Example

Test statistic

p-value

```
        Asymptotic one-sample Kolmogorov-Smirnov test

data:  white_wine$alcohol
D = 0.0901, p-value < 2.2e-16
alternative hypothesis: two-sided

Warning message:
In ks.test.default(white_wine$alcohol, "pnorm", mean = mean(white_wine$alcohol),  :
  ties should not be present for the Kolmogorov-Smirnov test
```

The KS test is for continuous variables and therefore should strictly not have any repeated values

A note on the KS Test in R

- The KS test is quite **sensitive** to the **sample size** which can cause either too “lenient” or too “harsh” results.
- It requires us to have a **specified distribution** in mind. Using the sample data to specify the distribution’s mean and standard deviation can be considered as “cheating” or **bad practice**.
- It is important to note that there are other tests for Normality that we can perform in R, for example, the **Shapiro–Wilk test** or **Lilliefor’s test**.

Exercise 28 (10 min)

1. Use the stroke dataset to test whether the `avg_glucose_level` variable follows a Normal distribution
2. Use the before and after dataset to test whether the differences follow a Normal distribution

The KS Test in R

We can also use the **`ks.test()`** function for comparing **two independent samples** and the hypothesis that they come from the **same underlying distribution**. Let's use some artificial data to perform this test.

```
set.seed(42)
random_normal <- rnorm(100, mean = 10, sd = 2) # Sample from a Normal distribution
random_poisson <- rpois(100, lambda = 5) # Sample from a Poisson distribution

# Do they come from the same underlying distribution?
ks.test(random_normal, random_poisson)
```

Demo Example

Using the wine dataset test the hypothesis that the underlying distribution of the `alcohol` variable is the same for the white and red wines.

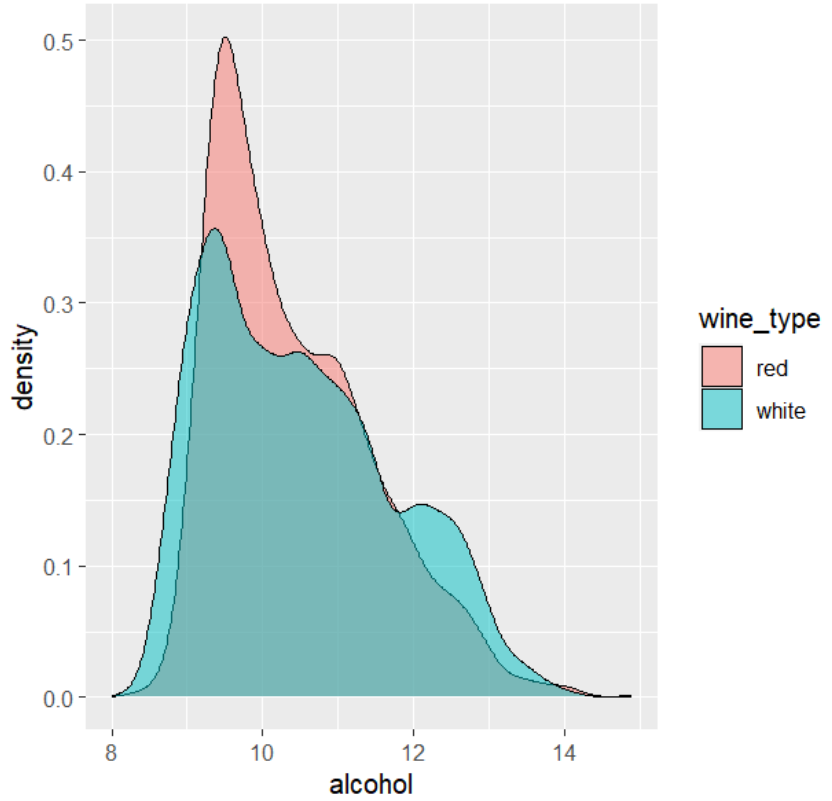
Demo Example

```r
# White wine data
white_wine <- wine %>%
  filter(wine_type == "white") # Keep only the white wine rows

# Red wine data
red_wine <- wine %>%
  filter(wine_type == "red") # Keep only the white wine rows

# Does alcohol have the same underlying distribution?
ks.test(white_wine$alcohol, red_wine$alcohol)

# Visualise the data
wine %>%
  ggplot(mapping = aes(x = alcohol, fill = wine_type)) +
  geom_density(alpha = 0.5)
```

Demo Example

```
        Asymptotic two-sample Kolmogorov-Smirnov test

data:  white_wine$alcohol and red_wine$alcohol
D = 0.094444, p-value = 3.846e-09
alternative hypothesis: two-sided
```

Reject the Null hypothesis



For two sample tests, the alternative hypothesis can be used to test if one sample is stochastically larger/smaller than the other

Exercise 29 (10 min)

1. Use the stroke dataset to test whether the `avg_glucose_level` variable of the male and of the female patients come from the same distribution.
2. Repeat question 1 but now using the “stroke” and “non-stroke” patients.

Extra: Visualise the distributions

Mann-Whitney U test

The **Mann-Whitney U test**, also known as the Wilcoxon rank-sum test, is a non-parametric statistical test used to determine whether there is a significant difference between the distributions of **two independent groups**.

Non-Parametric Nature: The Mann-Whitney U test does not assume that the data are normally distributed, making it useful for a wide range of situations.

Mann-Whitney U test

Below are some of the key points for the Mann-Whitney U test:

- **Ranking Data:** The test involves ranking all the data points from both groups together. The ranks are used to calculate the test statistic.
- **Assumptions:** The Mann-Whitney U test does not assume equal variances or that the data are Normally distributed. However, it does assume that the groups have similar shapes.
- **Use Cases:** Often used in biology, social sciences, and medical research when comparing two groups for differences in variables that are not Normally distributed or are ordinal in nature.

Mann-Whitney U test

In summary, we can specify the hypotheses for the Mann-Whitney U test as:

- **Null hypothesis:** There is no significant difference between the distributions of the two groups. *Focus is on the medians
- **Alternative hypothesis:** There is a significant difference between the distributions of the two groups.

We are testing whether the ranks of observations in one group are randomly distributed among the observations of the other group.

The `wilcox.test()` function in R

Once again, the **{stats}** package makes it very easy for us to perform a Mann-Whitney U test by using the function **`wilcox.test()`**. As a reminder we can use the command **`?wilcox.test`** to get the R help documentation about the function.

wilcox.test {stats} R Documentation

Wilcoxon Rank Sum and Signed Rank Tests

Description

Performs one- and two-sample Wilcoxon tests on vectors of data; the latter is also known as 'Mann-Whitney' test.

Usage

```
wilcox.test(x, ...)

## Default S3 method:
wilcox.test(x, y = NULL,
            alternative = c("two.sided", "less", "greater"
            mu = 0, paired = FALSE, exact = NULL, correct
            conf.int = FALSE, conf.level = 0.95,
            tol.root = 1e-4, digits.rank = Inf, ...)
```

Demo Example

Using the wine dataset test the hypothesis that the underlying distribution of the `alcohol` variable is the same for the white and red wines.

Demo Example

```r
# White wine data
white_wine <- wine %>%
  filter(wine_type == "white") # Keep only the white wine rows

# Red wine data
red_wine <- wine %>%
  filter(wine_type == "red") # Keep only the white wine rows

# Does alcohol have the same underlying distribution?
wilcox.test(white_wine$alcohol, red_wine$alcohol)
```

Demo Example

Test statistic

```
        Wilcoxon rank sum test with continuity correction

data:  white_wine$alcohol and red_wine$alcohol
W = 3457328, p-value = 0.1563
alternative hypothesis: true location shift is not equal to 0
```

Note: The KS test is sensitive to differences in both location and shape of the distributions, whereas Mann-Whitney U test focuses on differences in medians.

Exercise 30 (5 min)

1. Repeat the previous exercise using the Mann-Whitney U test
2. How would you interpret your results?

F-test

The F-test is used to determine if there is a significant difference in **variances** between two groups. We can specify the hypotheses for the F-test as

- **Null hypothesis:** The variances of the two groups are equal, this is also known as **homoscedasticity**
- **Alternative hypothesis:** The variances of the two groups are not equal, this is also known as **heteroscedasticity**

F-test

Below are some of the key points for the F-test:

- **Assumptions:** The F-test assumes that the data are normally distributed and that the samples are independent.
- **Use Cases:** Common applications include comparing the variances of experimental and control groups to ensure they are comparable. In analysis of variance (ANOVA) it is used to check the assumption of equal variances among different groups.

The `var.test()` function in R

The **{stats}** package makes it very easy for us to perform the F-test by using the function **`var.test()`**.

Note: The argument **`ratio`** is used to specify the hypothesised ratio of the population variances of the two samples. The default is 1, i.e. equality.

var.test {stats} R Documentation

F Test to Compare Two Variances

Description

Performs an F test to compare the variances of two samples from normal populations.

Usage

```
var.test(x, ...)

## Default S3 method:
var.test(x, y, ratio = 1,
         alternative = c("two.sided", "less", "greater"),
         conf.level = 0.95, ...)
```

Demo Example

Using the wine dataset test the hypothesis that the variance of the `alcohol` variable is the same for the white and red wines.

Demo Example

```
library(tidyverse)

wine <- read_csv("data/wine.csv")

# White wine data
white_wine <- wine %>%
  filter(wine_type == "white") # Keep only the white wine rows

# Red wine data
red_wine <- wine %>%
  filter(wine_type == "red") # Keep only the white wine rows

var.test(white_wine$alcohol, red_wine$alcohol)
```

Demo Example

Test statistic

```
        F test to compare two variances

data:  white_wine$alcohol and red_wine$alcohol
F = 1.3566, num df = 4499, denom df = 1499, p-value = 2.004e-12
alternative hypothesis: true ratio of variances is not equal to 1
95 percent confidence interval:
 1.247824 1.472225
sample estimates:
ratio of variances
          1.356563
```

Point estimate of variance ratio

The variances of the two groups are not equal

Exercise 31 (5 min)

1. Test whether the variance of `avg_glucose_level` variable is the same for the “stroke” and “non-stroke” patients
2. How would you interpret your results?

ANOVA

Objectives

1. Learn about **one-factor analysis of variance (ANOVA)**
2. Learn how to perform **ANOVA** in R

One-factor ANOVA

The **One-Factor ANOVA** (Analysis of Variance) is a statistical technique used to compare the means of **three or more groups** to determine if there are significant differences between them.

If the **between-group** variation is much larger than the **within-group** variation, it suggests that the groups are different from each other, indicating that at least one group mean differs significantly from the others.

One-factor ANOVA

A **factor** in ANOVA refers to the **independent variable** that categorises the data into **distinct** groups.

For example:

- In a study analysing the effects of different types of fertilizers on plant growth, the type of fertilizer (A, B, C, etc.) is the one factor.
- It can answer questions like: Does different teaching methods significantly affect student scores across multiple classes?

One-factor ANOVA

One-factor ANOVA compares means, identifying if there are differences among multiple groups. We can specify the hypotheses for ANOVA as

- **Null hypothesis:** The means of all groups are equal
- **Alternative hypothesis:** At least one group's mean is different from the others

One-factor ANOVA assumptions

Below are the assumptions for a one-factor ANOVA:

- **Normality:** Each group should follow a normal distribution
- **Homoscedasticity:** Variances within groups should be equal
- **Independence:** Observations within each group must be independent of each other

The aov() function in R

The **{stats}** package makes it very easy for us to perform a one-factor ANOVA by using the function **aov()**.

Note: This function requires us to provide a formula to specify the model.

aov {stats} R Documentation

Fit an Analysis of Variance Model

Description

Fit an analysis of variance model by a call to `lm` for each stratum.

Usage

```
aov(formula, data = NULL, projections = FALSE, qr = TRUE,
    contrasts = NULL, ...)
```

Arguments

`formula`	A formula specifying the model.
`data`	A data frame in which the variables specified in the formula will be found. If missing, the variables are searched for in the standard way.

A note about the formula notation in R

In R, **formula notation** is a convenient way to specify statistical models, especially in functions that perform linear modelling, regression analysis, and other statistical analyses such as ANOVA. The basic structure of a formula in R is

```
response_variable ~ predictor_variables
```

In one-factor ANOVA and the `aov()` function, we use the measure we want to compare to be the **response variable** and the distinct grouping to be the **predictor variable**.

Demo Example

Use the modified wine dataset (“wine_anova.csv”) to demonstrate a one-factor ANOVA and test the hypothesis that the mean alcohol for the 3 wine types is the same.

Demo Example

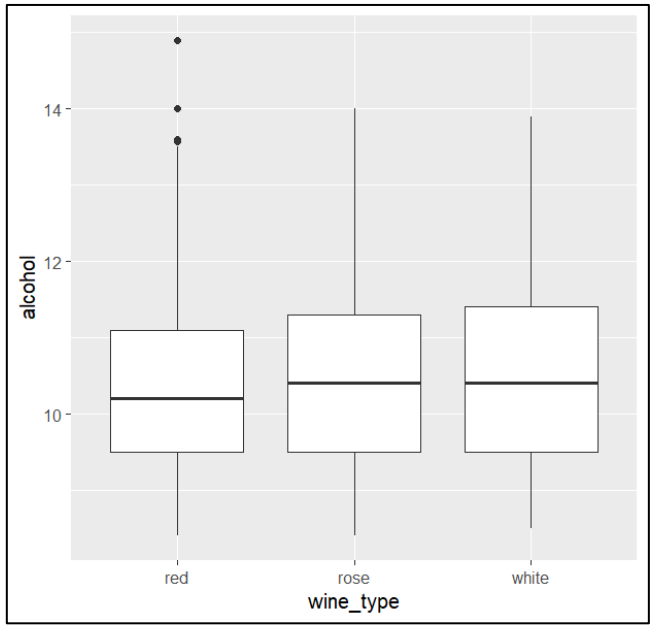
```r
library(tidyverse)

# Load the modified wine dataset containing 3 groups
wine_anova <- read_csv("data/wine_anova.csv")

# Perform one-factor ANOVA
result <- aov(alcohol ~ wine_type, data = wine_anova)

# Get results from analysis
summary(result)
```

Demo Example

Test statistic

p-value

```
> summary(result)
             Df Sum Sq Mean Sq F value Pr(>F)
wine_type     2     10   4.868   3.606 0.0273 *
Residuals  2997   4046   1.350
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```



We can conclude that there is evidence to reject the Null hypothesis in support that at least one group mean is different from the others.

SIMPLE/MULTIPLE LINEAR REGRESSION MODEL

Objectives

1. Learn about **linear models**
2. Learn how to fit a **simple** and a **multiple** linear model
3. Learn how to **interpret** the results

What is a simple linear model?

A statistical model (such a linear regression) is a **mathematical representation** that simplifies complex real-world phenomena. It allows us to analyse and **understand relationships** within data.

A **simple linear model** is a fundamental statistical model which assumes a linear relationship between **two variables**:

- independent variable (predictor)
- dependent variable (response)

Fitting a simple linear model

Let's assume that we believe that there is a **relationship** between **two variables**. Typically, the first step is to visualise that relationship via a **scatter plot**.

To describe this relationship, we calculate a **straight line** that **best fits** the data. This line represents a **simple linear model**, capturing the underlying pattern within the data. We can add this to a plot using the `geom_smooth()` function and specifying the method as **"lm"** for linear model.

You can specify more complex smoothing methods by changing the `method` argument of the `geom_smooth()` function

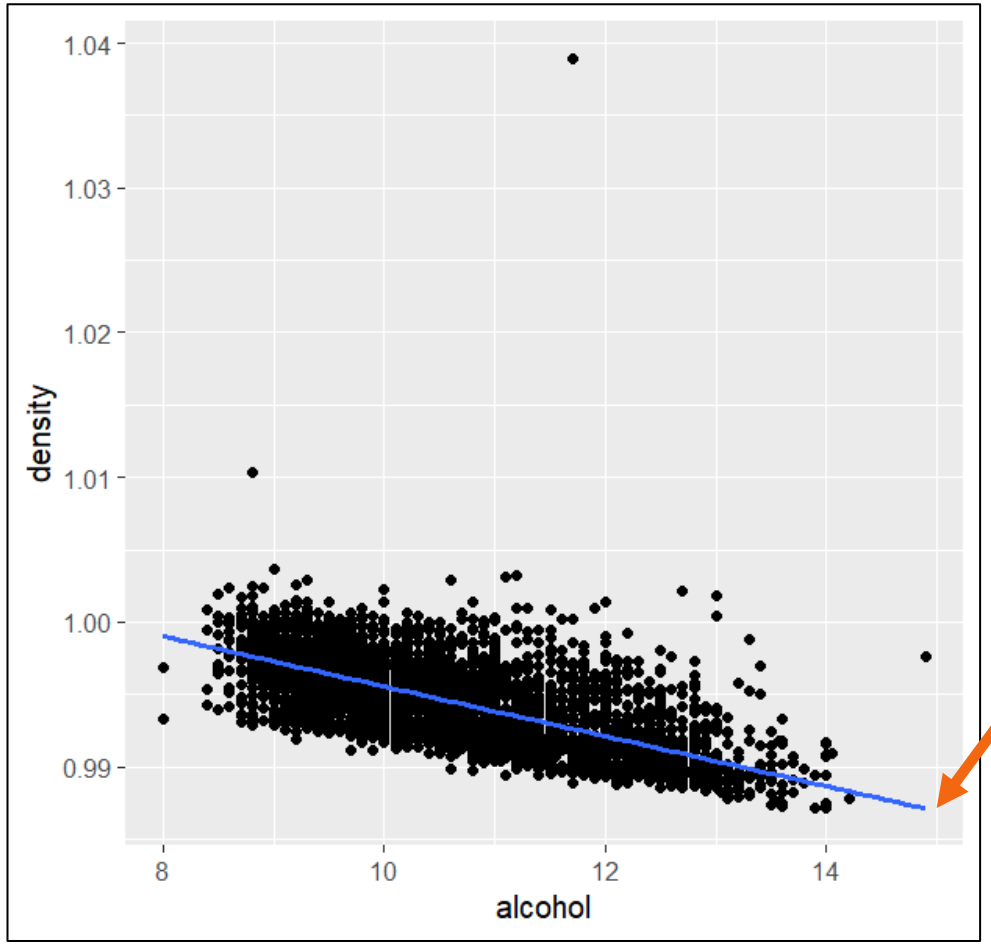
Demo Example

1. Use the wine dataset to create a scatter plot of the `alcohol` (x) and `density` (y) variables
2. Add a straight line of best fit

Demo Example

```
library(tidyverse)

wine <- read_csv("data/wine.csv")

wine %>%
  ggplot(mapping = aes(x = alcohol, y = density)) +
  geom_point() +
  # Fit a linear model without the standard error band
  geom_smooth(method = "lm", se = FALSE)
```

Demo Example



Notice the formula notation used

straight line of best fit

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

Simple Linear Regression Model

The simple linear regression model can be expressed as follows:

$$y = a + bx + \epsilon \qquad \text{where:}$$

	Description
$\boldsymbol{y}$	represents the dependent variable (response) that you want to predict
$\boldsymbol{a}$	represents the $\boldsymbol{y}$-intercept of the regression line, indicating the value of $\boldsymbol{y}$ when $\boldsymbol{x}$ is 0
$\boldsymbol{b}$	represents the slope of the regression line, indicating how much $\boldsymbol{y}$ is expected to change when $\boldsymbol{x}$ changes by one unit
$\boldsymbol{x}$	represents the independent variable (predictor) that you are using to make predictions
$\boldsymbol{\epsilon}$	represents the error term that is used to account for variation in the model

Fitting a simple linear model

When we fit a simple linear model, we are effectively determining the **best possible values** for a and b in the Linear Regression equation.

This involves identifying the straight line that **minimises** the total squared distances between each data point and the line. Fortunately, in R we can achieve this by using the `lm()` function from the **{stats}** package.

Demo Example

1. Use the wine dataset to fit a simple linear regression model using `alcohol` (x) and `density` (y) variables
2. Print the resulting model
3. Show the summary of the model and explain the key outputs

Demo Example

```
# Fit a linear regression model: y ~ x
result <- lm(density ~ alcohol, data = wine)

# Print model
print(result)

# Summary of model
summary(result)
```

Demo Example

```
> print(result)

Call:
lm(formula = density ~ alcohol, data = wine)

Coefficients:
(Intercept)      alcohol
   1.012808    -0.001725
```

Intercept value

Coefficient value of predictor variable

Note: You can change the **`scipen`** option to a high value to prevent R from using scientific notation:

```
options(scipen = 999)
```

Demo Example

```
> summary(result)

Call:
lm(formula = density ~ alcohol, data = wine)

Residuals:
      Min        1Q    Median        3Q       Max
-0.005688 -0.001596 -0.000136  0.001365  0.046354

Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept)  1.013e+00  2.495e-04    4060   <2e-16 ***
alcohol     -1.725e-03  2.363e-05     -73   <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.00218 on 5998 degrees of freedom
Multiple R-squared:  0.4705,    Adjusted R-squared:  0.4704
F-statistic:  5330 on 1 and 5998 DF,  p-value: < 2.2e-16
```

Indicates if model parameters are significant

Indicates the proportion of variance in 'density’ (y) explained by 'alcohol’ (x)

Tests the overall significance of the model

Is it a good model?

We can use the **R^2** (the higher the better) and **F-statistic** (test model significance) from the model summary as the first step to assess whether the model is significant and offers some **predictive ability**.

Even though we do not cover it in this course, it is common to further investigate the **distribution of the residual values** (actual minus fitted values) to be evenly spread around 0 and be close to a theoretical Normal distribution – usually through the visual of a Q-Q plot.

Multiple Regression

In reality, models often involve **more than one predictor variables** because there are many factors affecting the underlying process. To do this, we expand our linear regression formula to include more variables and their corresponding coefficients:

$$y = a + b_1 x_1 + b_2 x_2 + \cdots + b_n x_n + \epsilon$$

In R, our formula notation simply expands to reflect this:

```
# Formula for a multiple linear regression model (with 3 variables)
y ~ x1 + x2 + x3
```

Some notes on linear regression models

Variables: It is important to note that we can supply both **numerical** and **categorical** variables to a linear regression model. You must remember to convert the categorical variables into **factors**. One of the factors is used as the **baseline** represented by the intercept.

ANOVA: We can use this method to compare models (and select the best one) as long as the models are nested. A nested model is a model that is a subset of a more complex model.

QUIZ CHALLENGE

Get ready!

https://ahaslides.com/STATSINR4

Q & A

Thank you and happy coding!

DIET DATA

Course Data

This is a dataset that can be used for the analysis of weight loss from diet.

The dataset contains information about participants from 3 different diet plans. A participant's weight is measured before a 6-week diet plan and measured after the diet plan finishes.

DATASET CITATION: Kaggle: https://www.kaggle.com/datasets/

Couse Data Dictionary

- **id –** unique identifier of participants
- **gender –** “male”, “female“
- **age –** age of the patient
- **height –** the height of the participant in centimetres
- **diet –** indicates the diet plan followed by the participant, one of 3 different diet plans (1, 2 or 3)
- **pre_diet_weight –** the weight of the participant in Kg before diet
- **post_diet_weight –** the weight of the participant in Kg after diet

Group exercise - (30-45min)

Use the diet dataset or your own dataset to:

1) Formulate some hypotheses
2) Create visualisations
3) Perform any of the tests that we have covered so far to test the hypotheses

Exercise 32 (10 min)

1. Clear your environment and load the diet dataset
2. Perform a one-factor ANOVA to test the hypothesis that the mean weight loss for the 3 types of diets is the same
3. Create a boxplot to visualise the results
4. How would you interpret your results?

Exercise 33 (15 min)

1. Use the diet dataset to perform a simple linear regression model to predict the weight loss using a single predictor variable
2. Repeat the exercise for a multiple linear regression model using a selection of predictor variables