R Take-Home Assignments — Answer Key

PhD Course — R Programming

2025-10-01

About these documents

This document contains the **worked-out answers** to all assignments.

- The text is **identical** to the student version: introductions, task descriptions, and story-like explanations are the same.
- The **only difference** is that the code blocks are **filled in with solutions** instead of hints.
- Outputs (tables, plots) are also shown so you can check your own results.

How to use the answers

We strongly recommend:

1. **First try the assignments yourself.**
   Use the hints and explanations in the student versions.
2. **Only consult these answers afterwards,** to check your solutions, compare approaches, or get unstuck.
3. Treat this document as a **learning aid, not a shortcut**. The real value comes from struggling with the exercises before reading the solutions.

Final remark

Keep both the assignments and the answer key. Together, they provide you with a **mini reference manual** for common R workflows and analyses that you can revisit later during your PhD.

R Take-Home Assignment 1: Gapminder (Wrangling & Visualization)

Your Name

2025-09-30

Overview

In this assignment you will work with the **Gapminder dataset**, which contains information on life expectancy, population size, and GDP per capita for 142 countries between 1952 and 2007.
You will practice importing data, exploring its structure, subsetting, summarising, and producing plots to investigate global health and economic patterns.

- https://github.com/resbaz/r-novice-gapminder-files/raw/master/data/gapminder-FiveYearData.csv
- **Skills:** loading CSVs, subsetting, summarising, plotting
- **Deliverable:** knitted **HTML** plus your completed `.Rmd`

Preparation

You will need the following R packages:

- `dplyr`
- `ggplot2`
- `readr`

```
library(dplyr)
library(ggplot2)
library(readr)
```

1. Load the data

The starting point of any analysis is importing the dataset. Here, you will use the URL provided above.

Task: Load the Gapminder dataset into R and store it as `gapminder`.

```
url <- "https://github.com/resbaz/r-novice-gapminder-files/raw/master/data/gapminder-FiveYearData.csv"
gapminder <- read_csv(url, show_col_types = FALSE)
head(gapminder)
```

```
## # A tibble: 6 x 6
##   country      year      pop continent lifeExp gdpPercap
##   <chr>       <dbl>    <dbl> <chr>       <dbl>     <dbl>
## 1 Afghanistan  1952  8425333 Asia         28.8      779.
```

```
## 2 Afghanistan  1957  9240934 Asia         30.3      821.
## 3 Afghanistan  1962 10267083 Asia         32.0      853.
## 4 Afghanistan  1967 11537966 Asia         34.0      836.
## 5 Afghanistan  1972 13079460 Asia         36.1      740.
## 6 Afghanistan  1977 14880372 Asia         38.4      786.
```

2. Explore the dataset

Before doing analysis, it is important to understand the dataset's scope. You should examine how many countries it contains, which years are covered, and what variables are available.

Task: Report the number of countries, the range of years, and the variable names.

```
n_distinct(gapminder$country)
```

```
## [1] 142
```

```
range(gapminder$year)
```

```
## [1] 1952 2007
```

```
names(gapminder)
```

```
## [1] "country"   "year"      "pop"       "continent" "lifeExp"   "gdpPercap"
```

Write your answers here:

```
Countries: ...
Years: ...
Variables: ...
```

3. Subset the data

Analyses are often focused on a specific region. For this step, you will work only with European countries.

Task: Create a subset containing only the rows where `continent` is `"Europe"`. Save this as `europe`.

```
europe <- gapminder %>% filter(continent == "Europe")
head(europe)
```

```
## # A tibble: 6 x 6
##   country  year     pop continent lifeExp gdpPercap
##   <chr>   <dbl>   <dbl> <chr>       <dbl>     <dbl>
## 1 Albania  1952 1282697 Europe       55.2     1601.
## 2 Albania  1957 1476505 Europe       59.3     1942.
## 3 Albania  1962 1728137 Europe       64.8     2313.
## 4 Albania  1967 1984060 Europe       66.2     2760.
## 5 Albania  1972 2263554 Europe       67.7     3313.
## 6 Albania  1977 2509048 Europe       68.9     3533.
```

4. Summarise life expectancy

To see general patterns across the world, it is helpful to calculate average values. By summarising life expectancy by continent and year, you can compare how regions developed over time.

Task: Calculate the mean life expectancy per continent per year. Save the result as `lifeexp_summary`.

```r
lifeexp_summary <- gapminder %>%
  group_by(continent, year) %>%
  summarise(mean_lifeExp = mean(lifeExp), .groups = "drop")
head(lifeexp_summary)
```

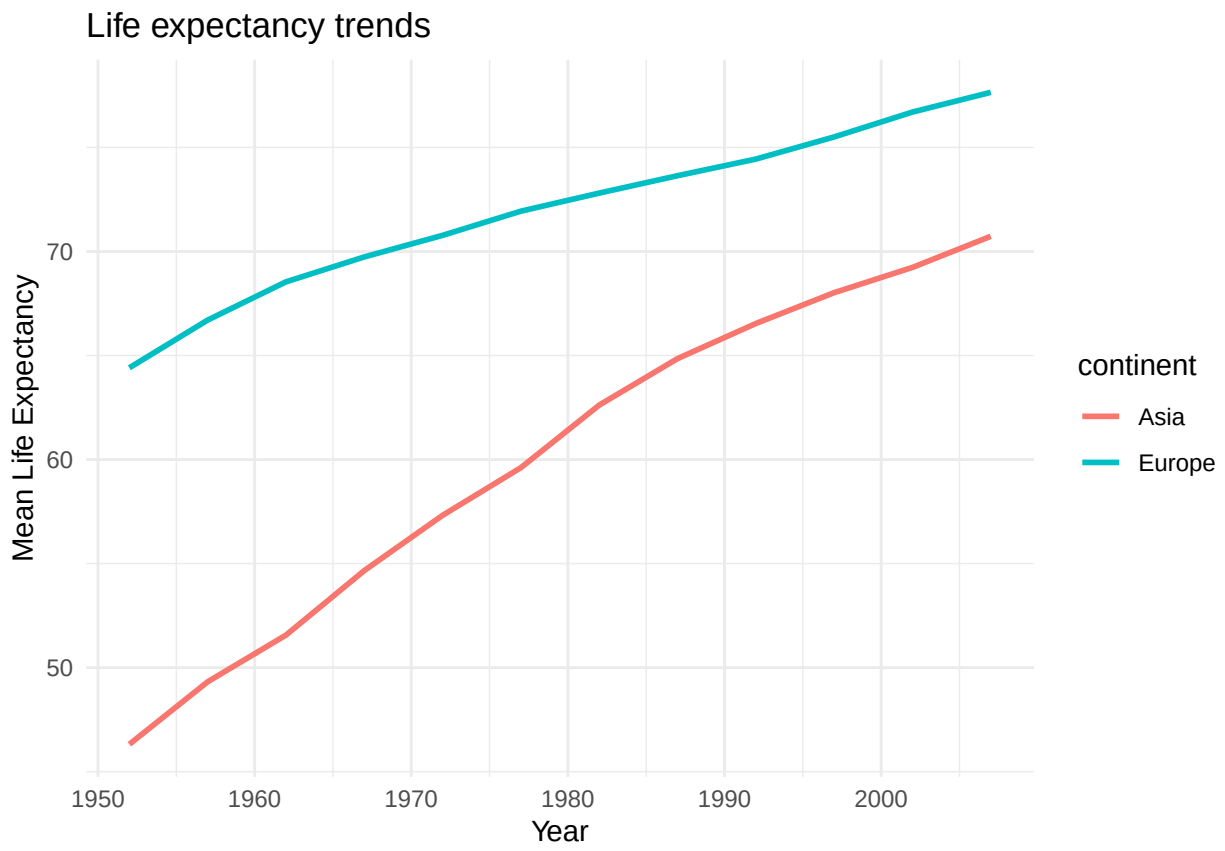
```
## # A tibble: 6 x 3
##   continent  year mean_lifeExp
##   <chr>     <dbl>        <dbl>
## 1 Africa     1952         39.1
## 2 Africa     1957         41.3
## 3 Africa     1962         43.3
## 4 Africa     1967         45.3
## 5 Africa     1972         47.5
## 6 Africa     1977         49.6
```

5. Visualise trends

Summaries are useful, but plots reveal patterns more clearly. A line plot of life expectancy over time allows you to compare trajectories between regions.

Task: Produce a line plot of life expectancy over time for **two continents of your choice**. Label the axes and add a title.

```r
lifeexp_summary %>% filter(continent %in% c("Europe", "Asia")) %>%
  ggplot(aes(year, mean_lifeExp, color=continent)) +
  geom_line(linewidth=1) +
  labs(title="Life expectancy trends", x="Year", y="Mean Life Expectancy") +
  theme_minimal()
```



6. Reflection

Numbers and plots show differences, but interpretation is just as important. Take a moment to reflect on what you observe.

Task: Write 3–4 sentences describing the patterns you see across continents in life expectancy trends.

Write your reflection here...

Appendix

```
sessionInfo()
```

```
## R version 4.5.1 (2025-06-13)
## Platform: aarch64-apple-darwin24.4.0
## Running under: macOS Sequoia 15.6.1
##
## Matrix products: default
## BLAS:   /opt/homebrew/Cellar/openblas/0.3.30/lib/libopenblasp-r0.3.30.dylib
## LAPACK: /opt/homebrew/Cellar/r/4.5.1/lib/R/lib/libRlapack.dylib;  LAPACK version 3.12.1
##
## locale:
## [1] en_US.UTF-8/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8
##
## time zone: Europe/Amsterdam
```

```
## tzcode source: internal
##
## attached base packages:
## [1] stats     graphics  grDevices utils     datasets  methods   base
##
## other attached packages:
## [1] readr_2.1.5   ggplot2_3.5.2 dplyr_1.1.4
##
## loaded via a namespace (and not attached):
##  [1] crayon_1.5.3       vctrs_0.6.5        cli_3.6.5          knitr_1.50
##  [5] rlang_1.1.6        xfun_0.52          generics_0.1.4     labeling_0.4.3
##  [9] bit_4.6.0          glue_1.8.0         htmltools_0.5.8.1  hms_1.1.3
## [13] scales_1.4.0       rmarkdown_2.30     grid_4.5.1         evaluate_1.0.4
## [17] tibble_3.2.1       tzdb_0.5.0         fastmap_1.2.0      yaml_2.3.10
## [21] lifecycle_1.0.4    compiler_4.5.1     RColorBrewer_1.1-3 pkgconfig_2.0.3
## [25] farver_2.1.2       digest_0.6.37      R6_2.6.1           utf8_1.2.4
## [29] tidyselect_1.2.1   curl_6.2.2         parallel_4.5.1     vroom_1.6.5
## [33] pillar_1.10.2      magrittr_2.0.3     bit64_4.6.0-1      tools_4.5.1
## [37] withr_3.0.2        gtable_0.3.6
```

R Take-Home Assignment 2: Palmer Penguins (Functions & Visualization)

Your Name

2025-10-01

Overview

In this assignment you will explore the **Palmer Penguins dataset**, which contains measurements for three penguin species living in the Palmer Archipelago in Antarctica. Researchers collected data on their bill size, flipper length, body mass, sex, and island of origin. The dataset is a favourite in data science because it provides a simple but realistic setting to practise exploring biological variation.

You will use this dataset to practise creating functions in R and visualising data in meaningful ways. Along the way, you will think like a biologist: Do larger penguins weigh more? Do species differ in size? And can we summarise this variation in a reusable way?

Preparation

You will need: `dplyr`, `ggplot2`, `palmerpenguins`.

```
library(dplyr)
library(ggplot2)
library(palmerpenguins)
```

1. Load and inspect the data

Before beginning any analysis, the first step is always to bring the data into R and take a first look. By knowing how many rows we have, which species are included, and what variables are measured, we can plan the kinds of questions we might answer.

Task: Load the `penguins` dataset from the `palmerpenguins` package. Then:
- Report the number of rows in the dataset.
- List the unique species included.
- Identify which variables are numeric.

```
data(penguins)
nrow(penguins)
```

```
## [1] 344
```

```
unique(penguins$species)
```

```
## [1] Adelie    Gentoo    Chinstrap
## Levels: Adelie Chinstrap Gentoo
```

```
sapply(penguins, is.numeric)
```

```
##           species            island    bill_length_mm     bill_depth_mm
##             FALSE             FALSE              TRUE              TRUE
```

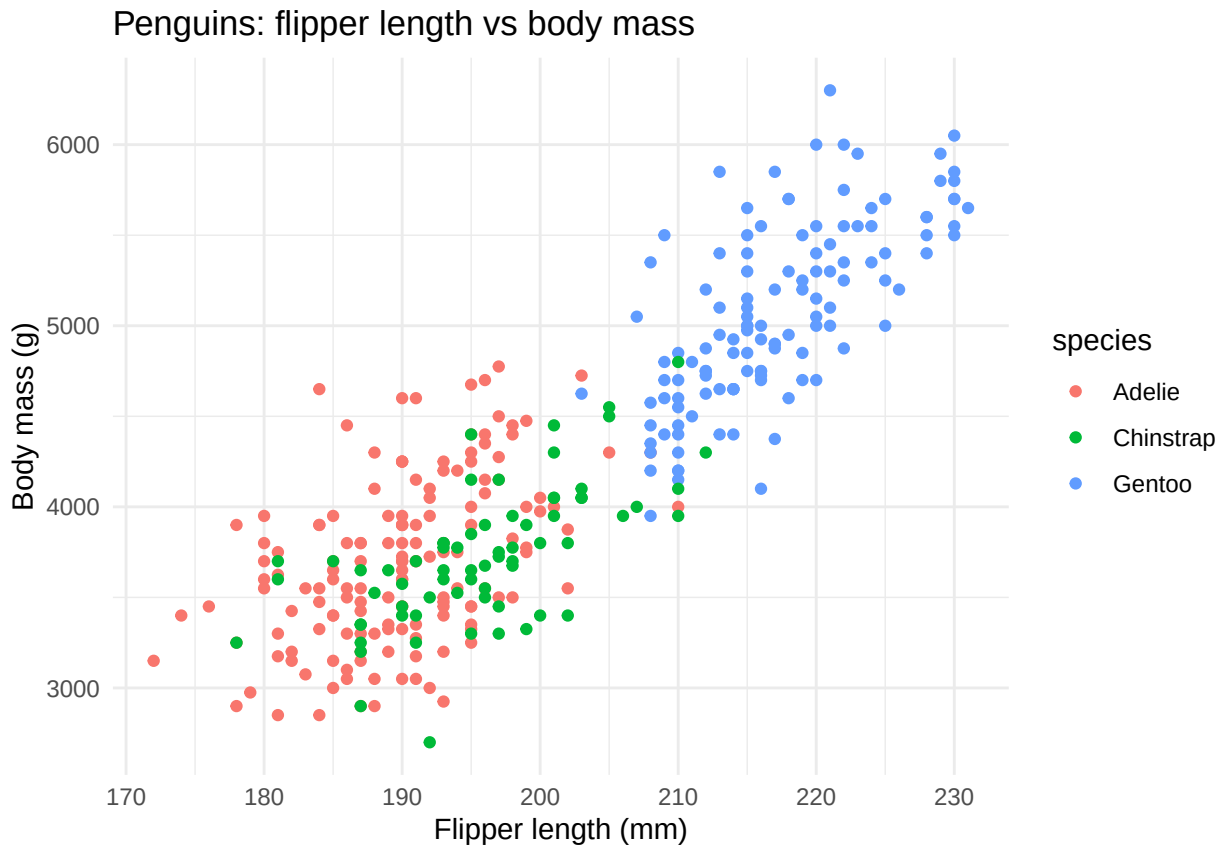
```
## flipper_length_mm       body_mass_g               sex              year
##              TRUE              TRUE             FALSE              TRUE
```

2. Plot flipper length vs body mass

Biologists often wonder whether certain traits scale together. For penguins, larger flippers might be expected to support a heavier body. If this is true, we should see a clear relationship between flipper length and body mass. By colouring the points by species, we can also see whether the relationship holds consistently across Adelie, Chinstrap, and Gentoo penguins.

Task: Make a scatterplot of `flipper_length_mm` (x-axis) against `body_mass_g` (y-axis). Colour the points by `species`. Add axis labels and a title.

```
ggplot(penguins, aes(x=flipper_length_mm, y=body_mass_g, color=species)) +
  geom_point() +
  labs(x="Flipper length (mm)", y="Body mass (g)", title="Penguins: flipper length vs body mass") +
  theme_minimal()
```



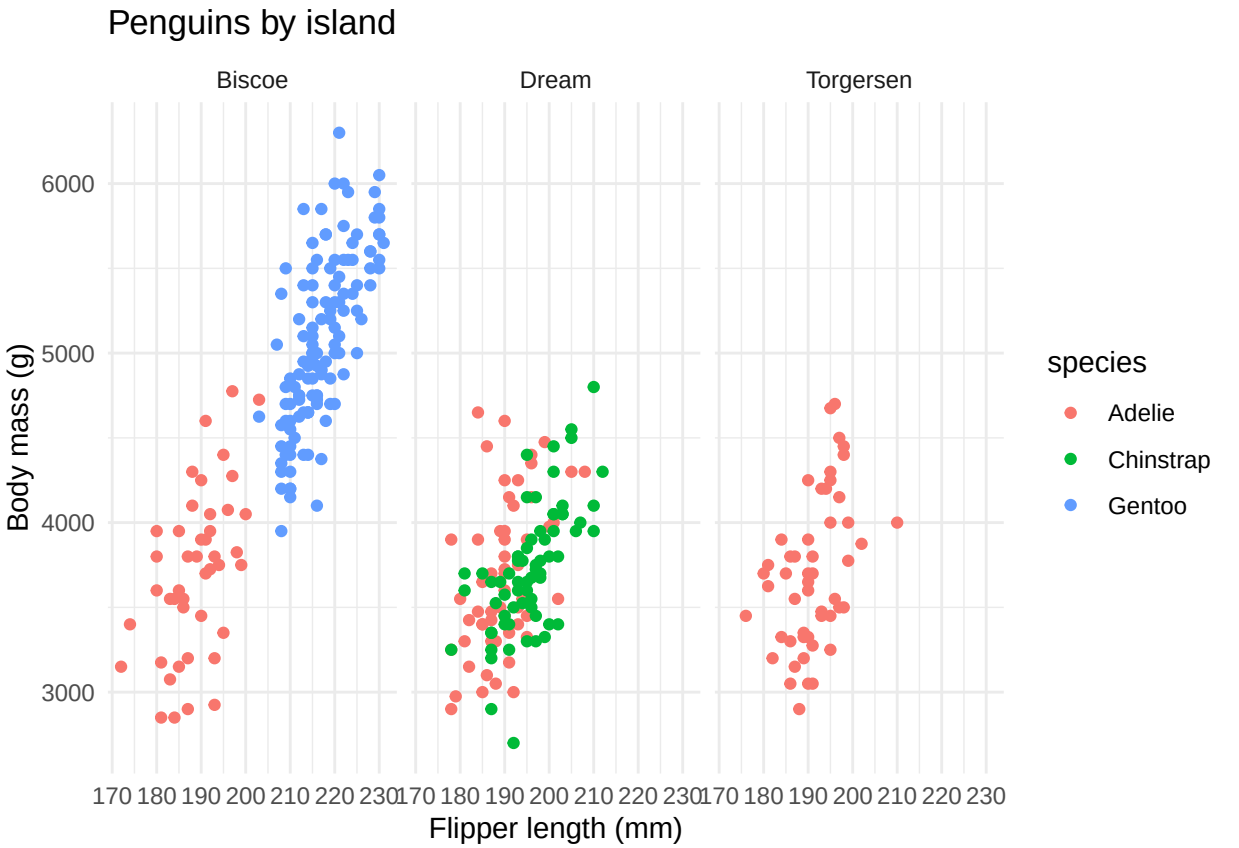
3. Facet by island

Species sometimes live on different islands, and environmental conditions (like food availability) can influence body size. To check whether penguins differ not just by species but also by location, we can break the plot into separate panels for each island.

Task: Recreate the scatterplot from Question 2, but facet it by `island` so that each island has its own panel.

```
ggplot(penguins, aes(x=flipper_length_mm, y=body_mass_g, color=species)) +
  geom_point() +
```

```
  facet_wrap(~ island) +
  labs(x="Flipper length (mm)", y="Body mass (g)", title="Penguins by island") +
  theme_minimal()
```



4. Write a summary function

Analyses are more powerful when we can repeat them easily. Suppose we want to compare average bill length, flipper length, and body mass for each species. Instead of writing separate code each time, we can create a function that computes these summaries for any species we provide.

Task: Write a function called `species_summary(sp)` that takes a species name as input and returns:
- the mean bill length (mm),
- the mean flipper length (mm),
- the mean body mass (g).

```
species_summary <- function(sp){
  penguins %>%
    filter(species == sp) %>%
    summarise(mean_bill = mean(bill_length_mm, na.rm=TRUE),
      mean_flipper = mean(flipper_length_mm, na.rm=TRUE),
      mean_mass = mean(body_mass_g, na.rm=TRUE)
  ) %>% mutate(species = sp) %>% select(species, everything())
}
species_summary("Adelie")
```

```
## # A tibble: 1 x 4
##   species mean_bill mean_flipper mean_mass
```

```
##   <chr>       <dbl>        <dbl>     <dbl>
## 1 Adelie       38.8         190.     3701.
```

5. Apply your function

Now that you have written your function, you can quickly generate summaries for all species and combine them into one table. This gives a compact overview of how species differ.

Task: Apply your `species_summary()` function to all three species in the dataset. Combine the results into a single data frame with one row per species.

```
bind_rows(lapply(unique(penguins$species), species_summary))
```

```
## # A tibble: 3 x 4
##   species   mean_bill mean_flipper mean_mass
##   <fct>         <dbl>        <dbl>     <dbl>
## 1 Adelie         38.8         190.     3701.
## 2 Gentoo         47.5         217.     5076.
## 3 Chinstrap      48.8         196.     3733.
```

6. Reflection

Finally, think about what these analyses reveal. Did larger flippers usually go hand in hand with heavier bodies? Were differences between species clear? Did location appear to matter?

Task: Write 3–4 sentences describing the main species differences and any island effects you observed.

Write your reflection here...

R Take-Home Assignment 3: OWID COVID-19 (Loops & Conditionals)

Your Name

2025-10-01

Overview

In this assignment you will work with the **Our World in Data (OWID) COVID-19 dataset**, which tracks the spread and impact of the pandemic globally. We will focus on daily new cases for three countries — the Netherlands, Germany, and Italy — to explore how case numbers change over time.

Because daily counts can be noisy (for example, due to weekend reporting effects), you will use rolling averages and logical flags to highlight important peaks. This exercise gives practice with **loops and conditionals**, two fundamental programming tools that help automate repetitive tasks and make decisions based on data.

- **Source:** OWID compact CSV
- **Skills:** importing CSVs, subsetting, rolling averages, logical flags, time-series plots
- **Deliverable:** knitted **HTML** plus your completed `.Rmd`

Preparation

You will need: `dplyr`, `ggplot2`, `readr`, `zoo`.

```
library(dplyr)
library(ggplot2)
library(readr)
library(zoo)
```

1. Load and subset the data

The dataset is large and contains many countries. Since we want to compare trends in only three European countries, we first need to import the full dataset and then narrow it down.

Task: Load the OWID dataset from the provided URL. Subset it to the Netherlands, Germany, and Italy. Convert the `date` column to type `Date`.

```
url <- 'https://catalog.ourworldindata.org/garden/covid/latest/compact/compact.csv'
owid <- read_csv(url, show_col_types = FALSE)
owid_small <- owid %>% filter(country %in% c('Netherlands','Germany','Italy')) %>% mutate(date = as.Dat
head(owid_small)
```

```
## # A tibble: 6 x 61
##   country date       total_cases new_cases new_cases_smoothed
##   <chr>   <date>           <dbl>     <dbl>              <dbl>
```

```
## 1 Germany 2020-01-01          NA        NA                 NA
## 2 Germany 2020-01-02          NA        NA                 NA
## 3 Germany 2020-01-03          NA        NA                 NA
## 4 Germany 2020-01-04           1         1                 NA
## 5 Germany 2020-01-05           1         0                 NA
## 6 Germany 2020-01-06           1         0                 NA
## # i 56 more variables: total_cases_per_million <dbl>,
## #   new_cases_per_million <dbl>, new_cases_smoothed_per_million <dbl>,
## #   total_deaths <dbl>, new_deaths <dbl>, new_deaths_smoothed <dbl>,
## #   total_deaths_per_million <dbl>, new_deaths_per_million <dbl>,
## #   new_deaths_smoothed_per_million <dbl>, excess_mortality <dbl>,
## #   excess_mortality_cumulative <dbl>,
## #   excess_mortality_cumulative_absolute <dbl>, ...
```

2. Compute a 7-day moving average

Raw daily case counts jump up and down due to reporting schedules. A moving average smooths these fluctuations, making it easier to see real trends.

Task: For each of the three countries, calculate a 7-day rolling average of `new_cases`. Store it in a new column `ma7`.

```
owid_small <- owid_small %>% arrange(country, date) %>% group_by(country) %>% mutate(ma7 = rollmean(new
head(owid_small)
```

```
## # A tibble: 6 x 62
##   country date       total_cases new_cases new_cases_smoothed
##   <chr>   <date>           <dbl>     <dbl>              <dbl>
## 1 Germany 2020-01-01          NA        NA                 NA
## 2 Germany 2020-01-02          NA        NA                 NA
## 3 Germany 2020-01-03          NA        NA                 NA
## 4 Germany 2020-01-04           1         1                 NA
## 5 Germany 2020-01-05           1         0                 NA
## 6 Germany 2020-01-06           1         0                 NA
## # i 57 more variables: total_cases_per_million <dbl>,
## #   new_cases_per_million <dbl>, new_cases_smoothed_per_million <dbl>,
## #   total_deaths <dbl>, new_deaths <dbl>, new_deaths_smoothed <dbl>,
## #   total_deaths_per_million <dbl>, new_deaths_per_million <dbl>,
## #   new_deaths_smoothed_per_million <dbl>, excess_mortality <dbl>,
## #   excess_mortality_cumulative <dbl>,
## #   excess_mortality_cumulative_absolute <dbl>, ...
```

3. Flag large values

Public health authorities often monitor when case numbers cross certain thresholds. Here, we will flag days with especially high case numbers.

Task: Add a logical column `flag_high` that is `TRUE` if `ma7 > 10000` and `FALSE` otherwise.

```
owid_small <- owid_small %>% mutate(flag_high = ma7 > 10000)
head(owid_small)
```

```
## # A tibble: 6 x 63
##   country date       total_cases new_cases new_cases_smoothed
##   <chr>   <date>           <dbl>     <dbl>              <dbl>
```

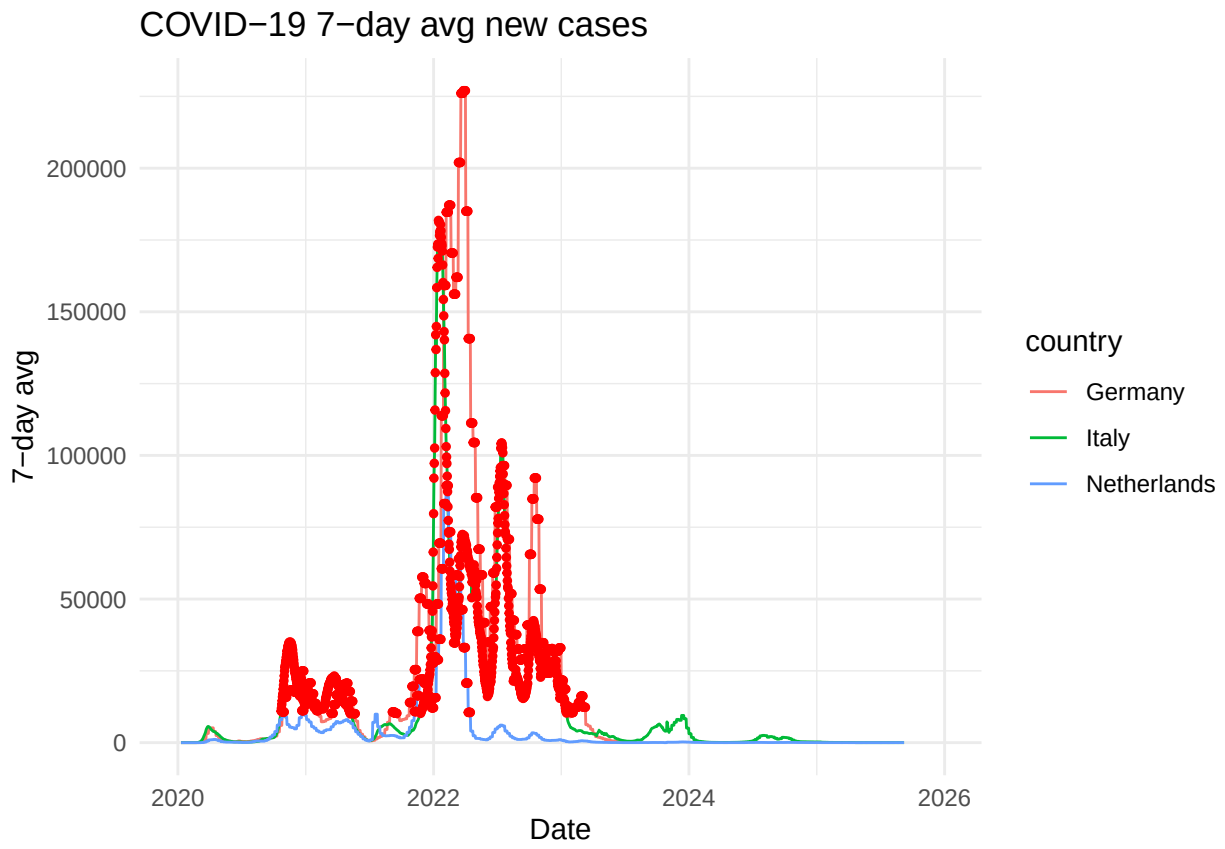
```
## 1 Germany 2020-01-01          NA        NA                 NA
## 2 Germany 2020-01-02          NA        NA                 NA
## 3 Germany 2020-01-03          NA        NA                 NA
## 4 Germany 2020-01-04           1         1                 NA
## 5 Germany 2020-01-05           1         0                 NA
## 6 Germany 2020-01-06           1         0                 NA
## # i 58 more variables: total_cases_per_million <dbl>,
## #   new_cases_per_million <dbl>, new_cases_smoothed_per_million <dbl>,
## #   total_deaths <dbl>, new_deaths <dbl>, new_deaths_smoothed <dbl>,
## #   total_deaths_per_million <dbl>, new_deaths_per_million <dbl>,
## #   new_deaths_smoothed_per_million <dbl>, excess_mortality <dbl>,
## #   excess_mortality_cumulative <dbl>,
## #   excess_mortality_cumulative_absolute <dbl>, ...
```

4. Plot with flagged points

A visualisation makes it easy to compare the three countries. Highlighting the flagged points will help you see when major peaks occurred.

Task: Create a line plot of the 7-day averages (`ma7`) over time for each country. Use different colours for the countries, and mark flagged points with an additional symbol (e.g. points in red).

```
ggplot(owid_small, aes(date, ma7, color=country)) + geom_line() +
  geom_point(data=subset(owid_small, flag_high), size=1, color='red') +
  labs(title='COVID-19 7-day avg new cases', x='Date', y='7-day avg') + theme_minimal()
```



5. Reflection

Finally, think about the patterns. Which country had the highest peaks? Did they occur at the same time? Did the threshold capture the waves you would expect?

Task: Write 3–4 sentences comparing the trends across the three countries and reflecting on how loops and conditionals helped structure the analysis.

Write your reflection here...

R Take-Home Assignment 4: World Development Data (Multiple Indicators)

Your Name

2025-10-01

Overview

In this assignment you will explore a dataset combining three important development indicators for almost all countries in the world between 1990 and 2020:

- **GDP per capita** – a measure of economic output per person
- **Life expectancy** – an indicator of population health
- **CO_2 emissions per capita** – a measure of environmental impact

The dataset `world_data.csv` has already been prepared for you. It allows you to study how wealth, health, and environmental outcomes are related across countries and over time.

- **File:** `world_data.csv` (stored in the same folder as this `.Rmd`)
- **Skills:** reading local CSVs, subsetting, cleaning, plotting, interpreting relationships
- **Deliverable:** knitted **HTML** plus your completed `.Rmd`

Preparation

You will need: `dplyr`, `ggplot2`, `readr`.

```
library(dplyr)
library(ggplot2)
library(readr)
```

1. Load the dataset

Before we can do any analysis, the dataset needs to be loaded into R. Since this file is stored locally, you'll also practise one of the most common first steps in data analysis: reading data from disk.

Task: Load `world_data.csv` into R and store it as `world_data`. Print the number of rows and columns, and look at the first few lines of the data.

```
world_data <- read_csv("world_data.csv", show_col_types = FALSE)
print(dim(world_data))
```

```
## [1] 4322    5
```

```
head(world_data)
```

```
## # A tibble: 6 x 5
```

```
##   country      year gdp_per_capita co2_per_capita life_expectancy
##   <chr>       <dbl>          <dbl>          <dbl>           <dbl>
## 1 Afghanistan  1990          1085.          0.168            51.6
## 2 Afghanistan  1991           984.          0.156            51.3
## 3 Afghanistan  1992           955.          0.112            51.4
## 4 Afghanistan  1993           658.          0.1              51.3
## 5 Afghanistan  1994           487.          0.089            50.7
## 6 Afghanistan  1995           721.          0.083            51.0
```

2. Subset the dataset

Instead of analysing every year, it's common to take a snapshot of one year for cross-country comparisons. Here we'll use 2015, as it provides recent data but avoids issues with missing values in the latest years.

Task: Subset the dataset to the year 2015. Show the resulting table (a few rows is enough).

```
subset_data <- world_data %>% filter(year == 2015)
head(subset_data)
```

```
## # A tibble: 6 x 5
##   country      year gdp_per_capita co2_per_capita life_expectancy
##   <chr>       <dbl>          <dbl>          <dbl>           <dbl>
## 1 Afghanistan  2015          1856.          0.286            57.8
## 2 Albania      2015         10085.          1.63             77.4
## 3 Algeria      2015         13781.          4                77.3
## 4 Angola       2015          6014.          0.977            64.0
## 5 Argentina    2015         19559.          4.41             76.6
## 6 Armenia      2015         10087.          1.87             75.6
```

3. Explore summary statistics

Before making plots, it's useful to see some descriptive statistics. Averages and ranges can reveal whether values look reasonable, and whether there is a lot of variation between countries.

Task: For all countries in 2015, calculate the mean and range for GDP per capita, life expectancy, and CO_2 emissions per capita.

```
subset_data %>% summarise(
  mean_gdp = mean(gdp_per_capita), range_gdp = paste(range(gdp_per_capita), collapse='-'),
  mean_life = mean(life_expectancy), range_life = paste(range(life_expectancy), collapse='-'),
  mean_co2 = mean(co2_per_capita), range_co2 = paste(range(co2_per_capita), collapse='-')
)
```

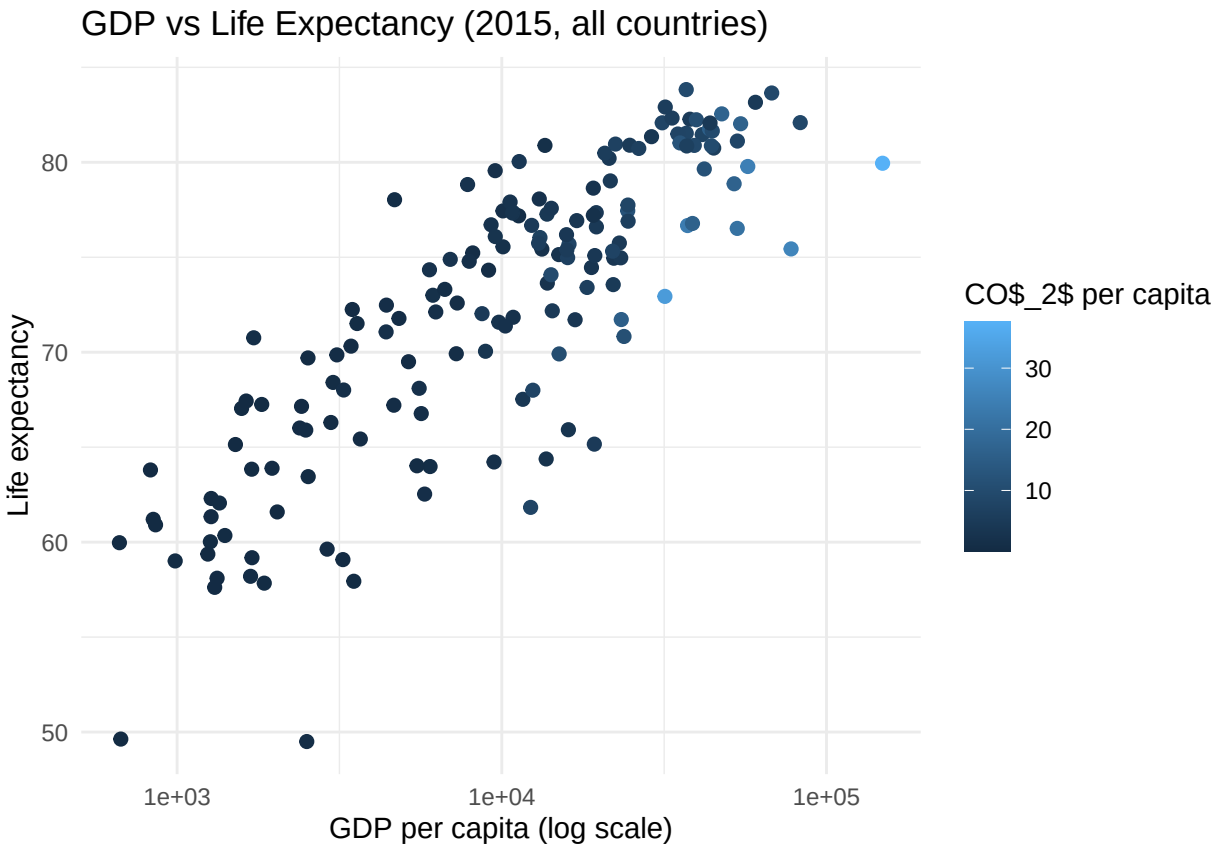
```
## # A tibble: 1 x 6
##   mean_gdp range_gdp                      mean_life range_life mean_co2 range_co2
##      <dbl> <chr>                              <dbl> <chr>         <dbl> <chr>
## 1   17638. 664.438548170731-148822.6354~      72.5 49.5-83.8~     4.77 0.032-37~
```

4. Plot GDP vs life expectancy

A classic question in development studies is whether economic prosperity translates into better health outcomes. Plotting GDP per capita against life expectancy lets us explore this visually. Adding CO_2 emissions as colour allows us to consider whether higher prosperity comes with environmental costs.

Task: Create a scatterplot of GDP per capita (x-axis, log scale) vs life expectancy (y-axis). Colour the points by CO_2 emissions per capita. Add informative axis labels and a title.

```r
ggplot(subset_data, aes(x=gdp_per_capita, y=life_expectancy, color=co2_per_capita)) +
  geom_point(size=2) +
  scale_x_log10() +
  labs(x='GDP per capita (log scale)', y='Life expectancy', color='CO$_2$ per capita',
       title='GDP vs Life Expectancy (2015, all countries)') +
  theme_minimal()
```



5. Reflection

The final step in any analysis is to connect numbers and plots back to real-world meaning. Does wealth always mean health? Are there exceptions? Do you notice patterns in CO_2 emissions?

Task: Write 3–4 sentences interpreting your findings. Mention at least one interesting or surprising pattern.

Write your reflection here...

R Take-Home Assignment 5 (Instructor Key): Statistics with Penguins

Instructor

2025-10-01

Overview

In this assignment you will practice simple statistical tests in R using the **palmerpenguins** dataset. We will ask biological questions about penguin body mass, learn to run standard tests in R, and interpret the results. Each section starts with a short story to explain why we are doing this step.

Source

`palmerpenguins` package

Skills

`t-tests, ANOVA, post-hoc tests, effect sizes, diagnostics`

1. Load the dataset

Before we can analyze penguins, we need to load the dataset. Real-world data often contain missing values, which can cause errors in analysis, so we will remove them first.

```
data(penguins)
penguins <- penguins %>% drop_na()
glimpse(penguins)
```

```
## Rows: 333
## Columns: 8
## $ species           <fct> Adelie, Adelie, Adelie, Adelie, Adelie, Adelie, Adel~
## $ island            <fct> Torgersen, Torgersen, Torgersen, Torgersen, Torgerse~
## $ bill_length_mm    <dbl> 39.1, 39.5, 40.3, 36.7, 39.3, 38.9, 39.2, 41.1, 38.6~
## $ bill_depth_mm     <dbl> 18.7, 17.4, 18.0, 19.3, 20.6, 17.8, 19.6, 17.6, 21.2~
## $ flipper_length_mm <int> 181, 186, 195, 193, 190, 181, 195, 182, 191, 198, 18~
## $ body_mass_g       <int> 3750, 3800, 3250, 3450, 3650, 3625, 4675, 3200, 3800~
## $ sex               <fct> male, female, female, female, male, female, male, fe~
## $ year              <int> 2007, 2007, 2007, 2007, 2007, 2007, 2007, 2007, 2007~
```

2. Two-sample t-test

Imagine you are a biologist asking: *Do male and female penguins of the same species differ in body mass?* The **t-test** compares the means of two groups. It answers whether the difference is large enough that it is unlikely to be due to chance.

```
adelie <- penguins %>% filter(species == "Adelie")
t.test(body_mass_g ~ sex, data = adelie)
```

```
##
##  Welch Two Sample t-test
##
## data:  body_mass_g by sex
## t = -13.126, df = 135.69, p-value < 2.2e-16
## alternative hypothesis: true difference in means between group female and group male is not equal to
## 95 percent confidence interval:
##  -776.3012 -573.0139
## sample estimates:
## mean in group female   mean in group male
##             3368.836             4043.493
```

```
cohens_d(body_mass_g ~ sex, data = adelie)
```

```
## Cohen's d |         95% CI
## -------------------------
## -2.17     | [-2.58, -1.76]
##
## - Estimated using pooled SD.
```

3. ANOVA

Now suppose we want to compare *all three species at once*. Do Gentoo, Adelie, and Chinstrap penguins differ in body mass?
An **ANOVA (Analysis of Variance)** extends the t-test to more than two groups.

```
anova_model <- aov(body_mass_g ~ species, data = penguins)
summary(anova_model)
```

```
##              Df    Sum Sq  Mean Sq F value Pr(>F)
## species       2 145190219 72595110   341.9 <2e-16 ***
## Residuals   330  70069447   212332
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

4. Post-hoc tests

ANOVA tells us if there is *some* difference between groups, but not *which groups differ*. For that, we run a **Tukey HSD post-hoc test**.

```
TukeyHSD(anova_model)
```

```
##   Tukey multiple comparisons of means
##     95% family-wise confidence level
##
## Fit: aov(formula = body_mass_g ~ species, data = penguins)
##
## $species
##                         diff       lwr       upr    p adj
## Chinstrap-Adelie   26.92385 -132.3528  186.2005 0.916431
## Gentoo-Adelie    1386.27259 1252.2897 1520.2554 0.000000
## Gentoo-Chinstrap 1359.34874 1194.4304 1524.2671 0.000000
```

5. Effect sizes

Statistical significance is not enough: we also want to know *how large the difference is*.
- For t-tests, we use **Cohen's d** (small 0.2, medium 0.5, large 0.8).

- For ANOVA, we use ^{2} **(eta squared)** (small .01, medium .06, large .14).

```
eta_squared(anova_model)
```

```
## # Effect Size for ANOVA
##
## Parameter | Eta2 |       95% CI
## ------------------------------
## species   | 0.67 | [0.63, 1.00]
##
## - One-sided CIs: upper bound fixed at [1.00].
```

6. Diagnostics (checking assumptions)

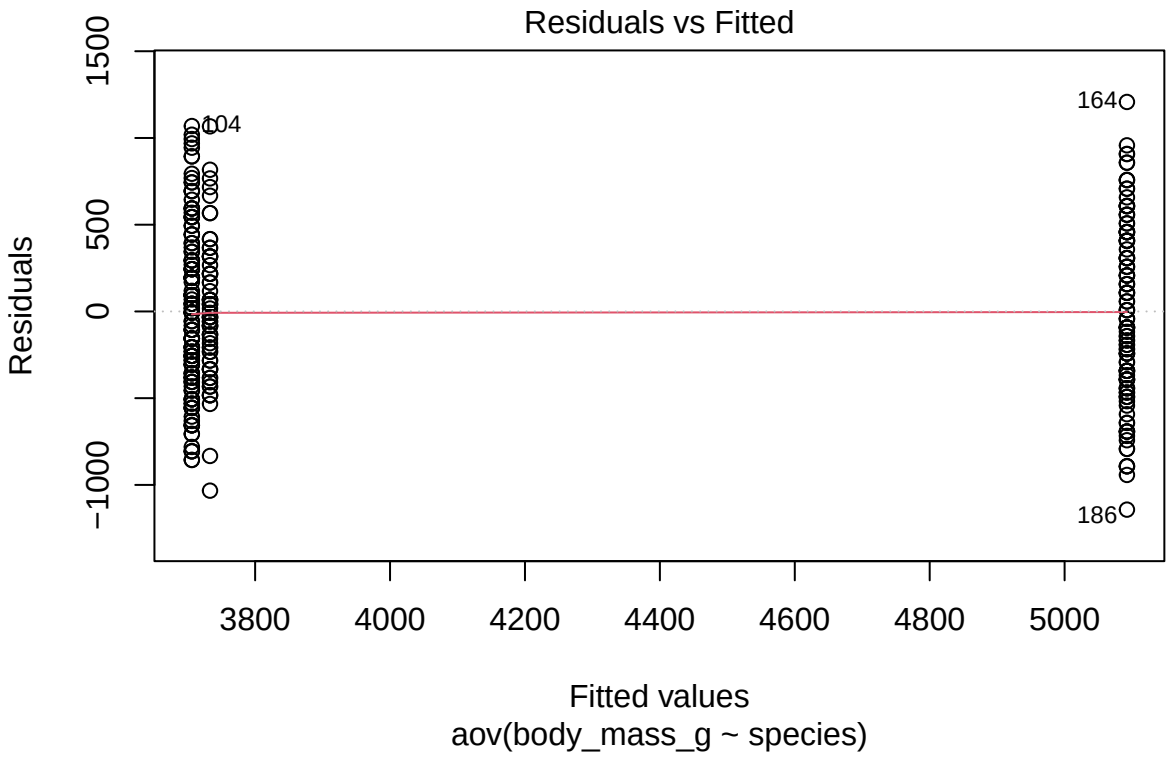
Every statistical test makes assumptions. For ANOVA, the two most important are:

1. **Equal spread of errors (homoscedasticity):**
   The variation in the data should be roughly the same across all groups.
2. **Normally distributed errors:**
   The differences between the observed data and the model's predictions (residuals) should follow a bell-shaped curve.

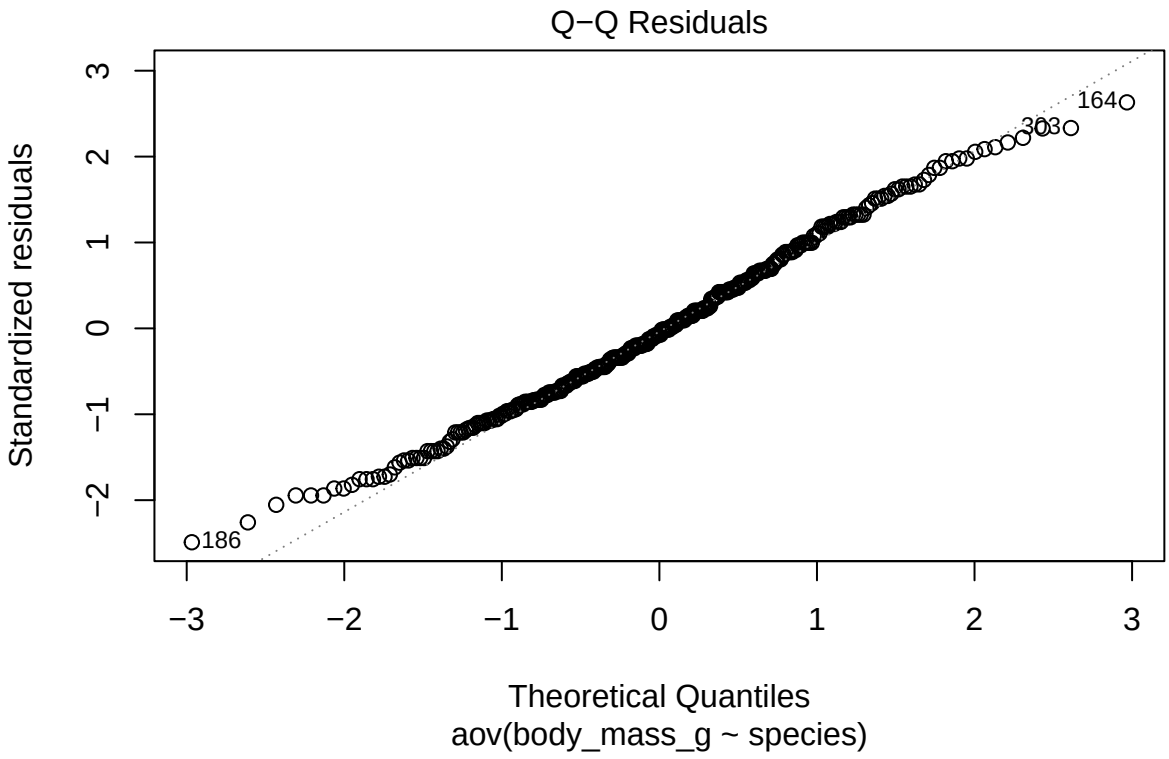
We can check these assumptions using two standard diagnostic plots:

- **Residuals vs Fitted plot**
  - Each dot is one observation.
  - The x-axis shows what the model predicts, the y-axis shows the "error" (residual).
  - What to look for: the dots should look like random noise.
    - If you see a curve → the relationship might not be captured well.
    - If you see a funnel shape → group variances may not be equal.
- **Normal QQ plot**
  - This compares your residuals to what would be expected if they were perfectly normal.
  - What to look for: the dots should fall roughly along the diagonal line.
    - If they bend away strongly → residuals may not be normally distributed.

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




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



7. Reflection

Statistics are not just numbers: they help answer real questions. Think like a biologist preparing a short report.

Across species, body mass differs significantly, with Gentoo penguins being much heavier. Within a species (e.g. Adelie), males are significantly heavier than females. Effect sizes (Cohen's d and ^{2}) show these are large effects. Diagnostics are usually acceptable; residuals look roughly normal and variance is fairly equal.