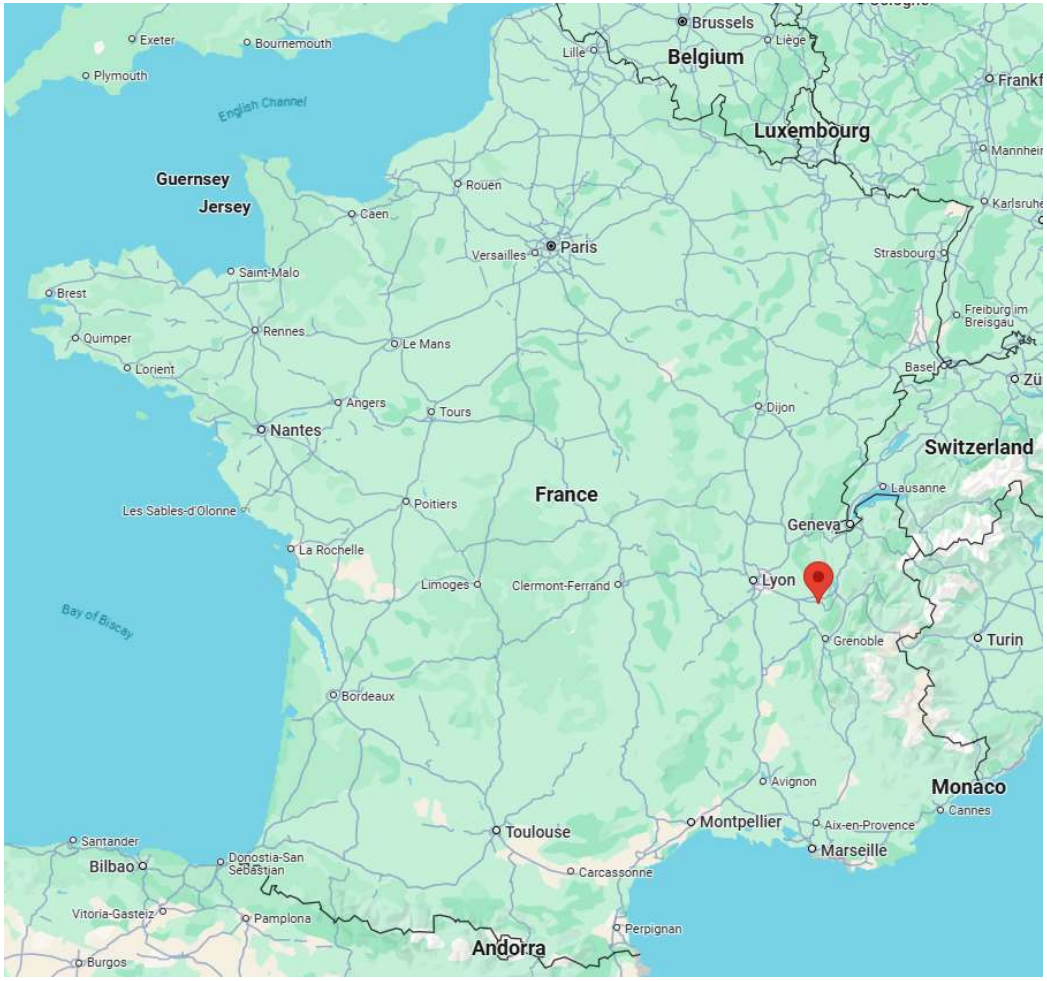
Reproducible analysis pipelines with {targets}

Olivia Angelin-Bonnet
The New Zealand Institute for Plant and Food Research Ltd

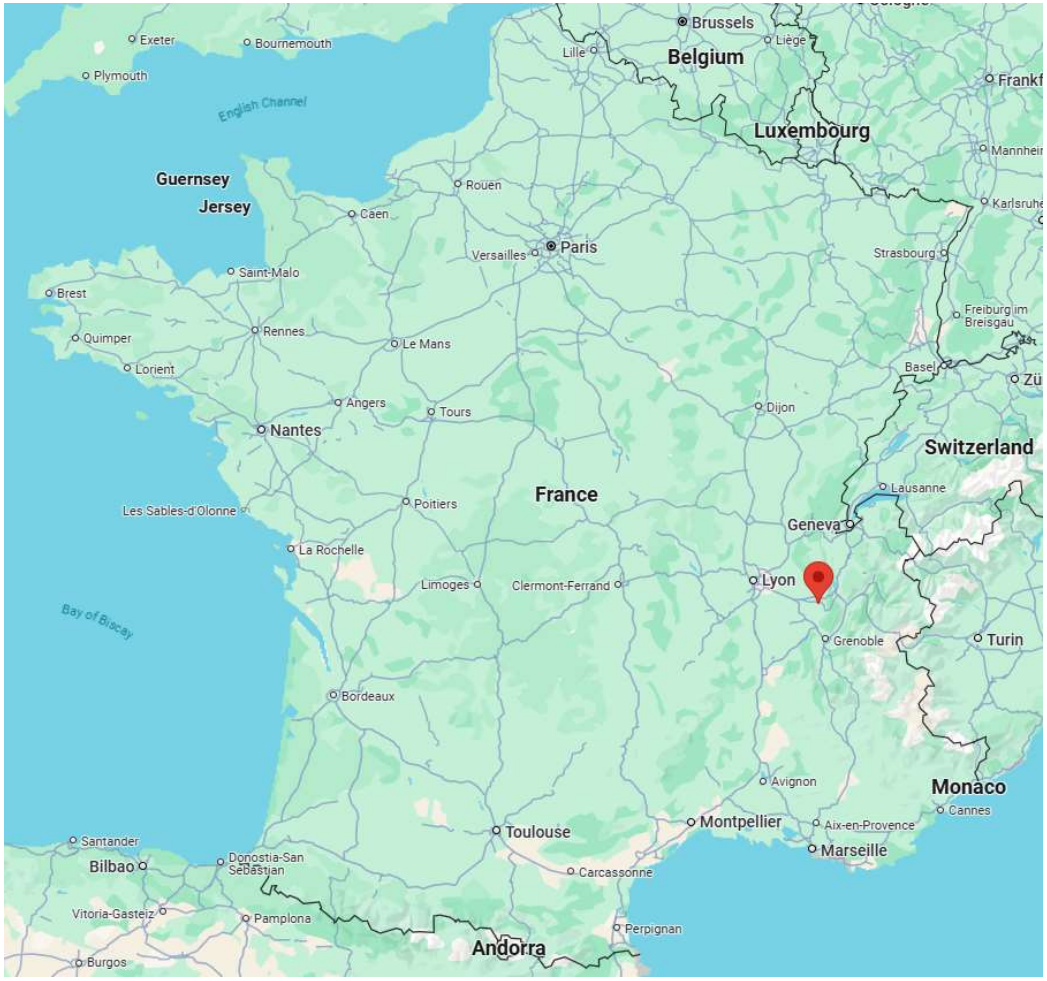
7 April 2025

Hi! I’m Olivia :)



Location of Le Pont-de-Beauvoisin on a map of France

Hi! I’m Olivia :)



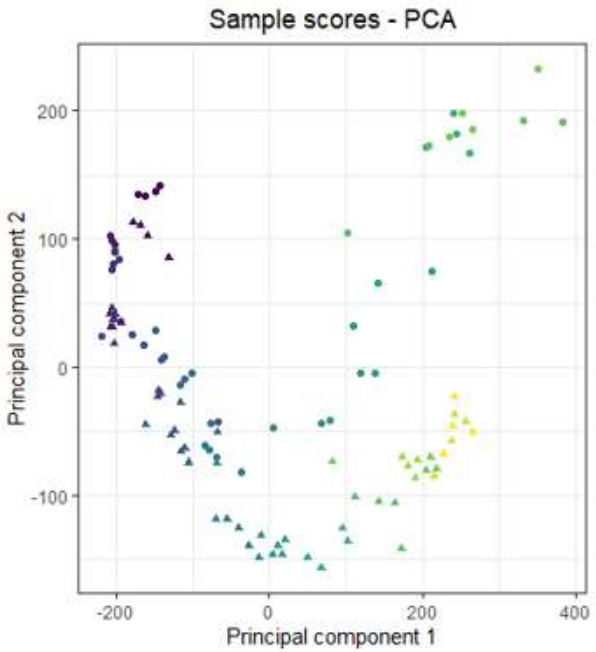
Location of Le Pont-de-Beauvoisin on a map of France

Location of Palmerston North on a map of New Zealand

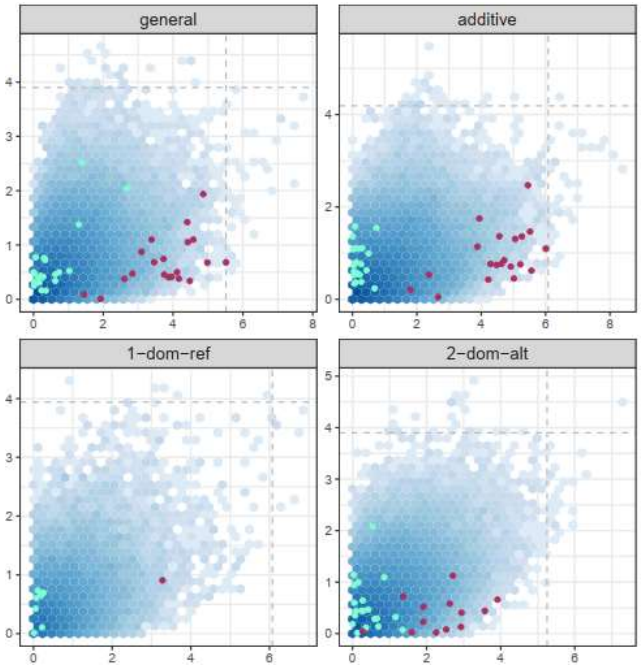
Hi! I'm Olivia :)

Statistical scientist at Plant & Food Research, working on omics analyses and multi-omics integration

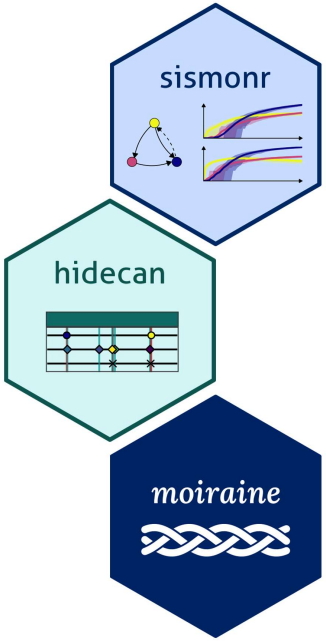
Multivariate analyses



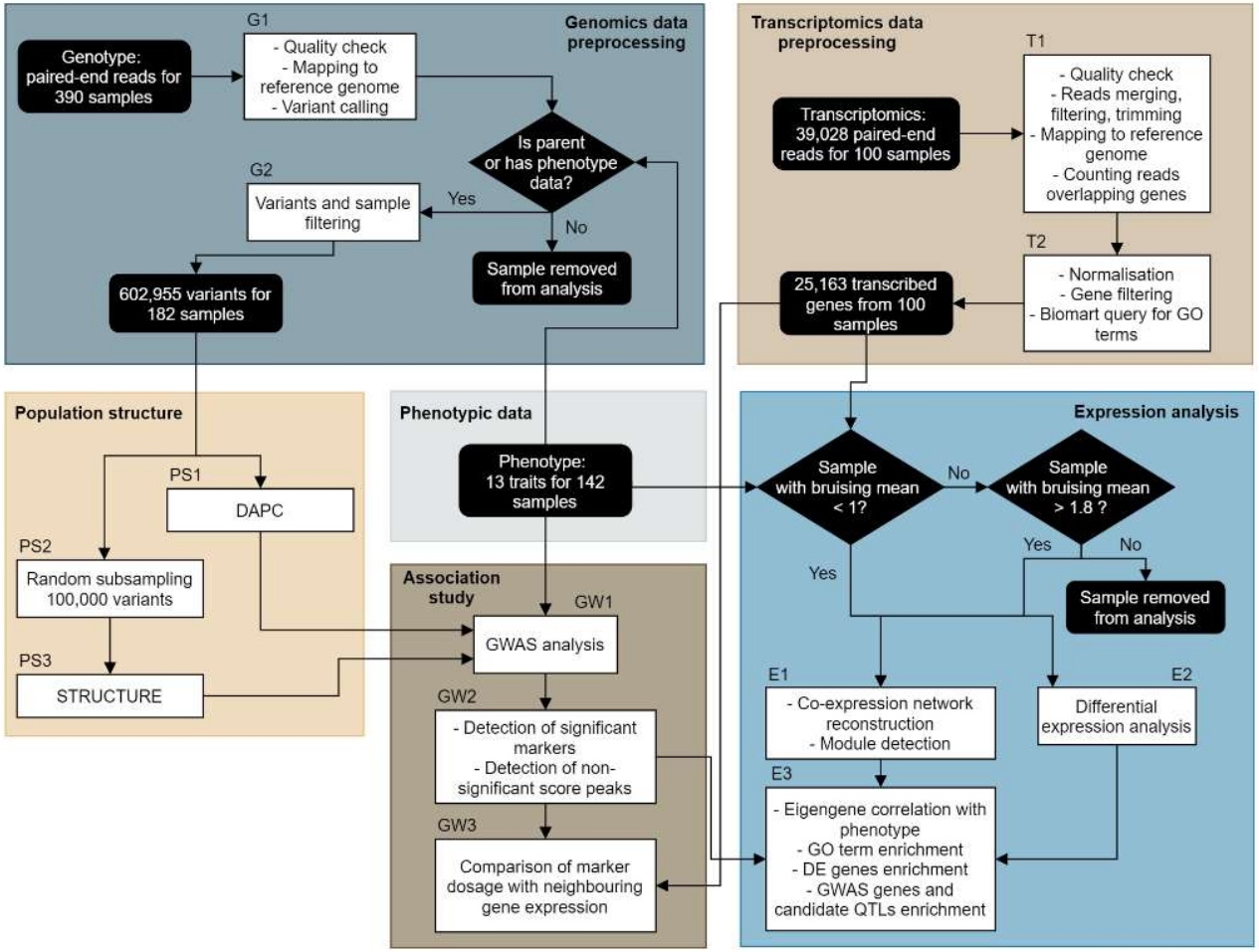
Visualisations



R packages



Developing complex analyses in R



Schema of the analysis for one of my thesis chapters

Developing complex analyses in R

Project folder:

```
thesis/chapter4_code
├── genomics_data_analysis
│   ├── 00_genomics_wrangling.Rmd
│   ├── 01_genomics_filtering.Rmd
│   └── 02_genomics_eda.Rmd
└── transcriptomics_data_analysis
|   ├── 00_transcriptomics_wrangling.Rmd
|   ├── 01_transcriptomics_normalisation.Rmd
|   ├── 02_transcriptomics_diff_expr.Rmd
|   └── 03_transcriptomics_wgcna.Rmd
...
```

Developing complex analyses in R

Project folder:

```
thesis/chapter4_code
├── genomics_data_analysis
│   ├── 00_genomics_wrangling.Rmd
│   ├── 01_genomics_filtering.Rmd
│   └── 02_genomics_eda.Rmd
└── transcriptomics_data_analysis
|   ├── 00_transcriptomics_wrangling.Rmd
|   ├── 01_transcriptomics_normalisation.Rmd
|   ├── 02_transcriptomics_diff_expr.Rmd
|   └── 03_transcriptomics_wgcna.Rmd
...
```

- Input data has changed; what do I need to re-run?

Developing complex analyses in R

Project folder:

```
thesis/chapter4_code
├── genomics_data_analysis
│   ├── 00_genomics_wrangling.Rmd
│   ├── 01_genomics_filtering.Rmd
│   └── 02_genomics_eda.Rmd
└── transcriptomics_data_analysis
|   ├── 00_transcriptomics_wrangling.Rmd
|   ├── 01_transcriptomics_normalisation.Rmd
|   ├── 02_transcriptomics_diff_expr.Rmd
|   └── 03_transcriptomics_wgcna.Rmd
...
```

- Input data has changed; what do I need to re-run?
- In which order do I need to run these scripts?

Developing complex analyses in R

Project folder:

```
thesis/chapter4_code
├── genomics_data_analysis
│   ├── 00_genomics_wrangling.Rmd
│   ├── 01_genomics_filtering.Rmd
│   └── 02_genomics_eda.Rmd
└── transcriptomics_data_analysis
|   ├── 00_transcriptomics_wrangling.Rmd
|   ├── 01_transcriptomics_normalisation.Rmd
|   ├── 02_transcriptomics_diff_expr.Rmd
|   └── 03_transcriptomics_wgcna.Rmd
...
```

- Input data has changed; what do I need to re-run?
- In which order do I need to run these scripts?
- I want to change one line of text in my report, how long will it take to re-generate the output?

Developing complex analyses in R

Solution: Turn your scripts into pipelines with `{targets}`!

Developing complex analyses in R

Solution: Turn your scripts into pipelines with `{targets}`!

Developing complex analyses in R

Solution: Turn your scripts into pipelines with `{targets}`!

Developing complex analyses in R

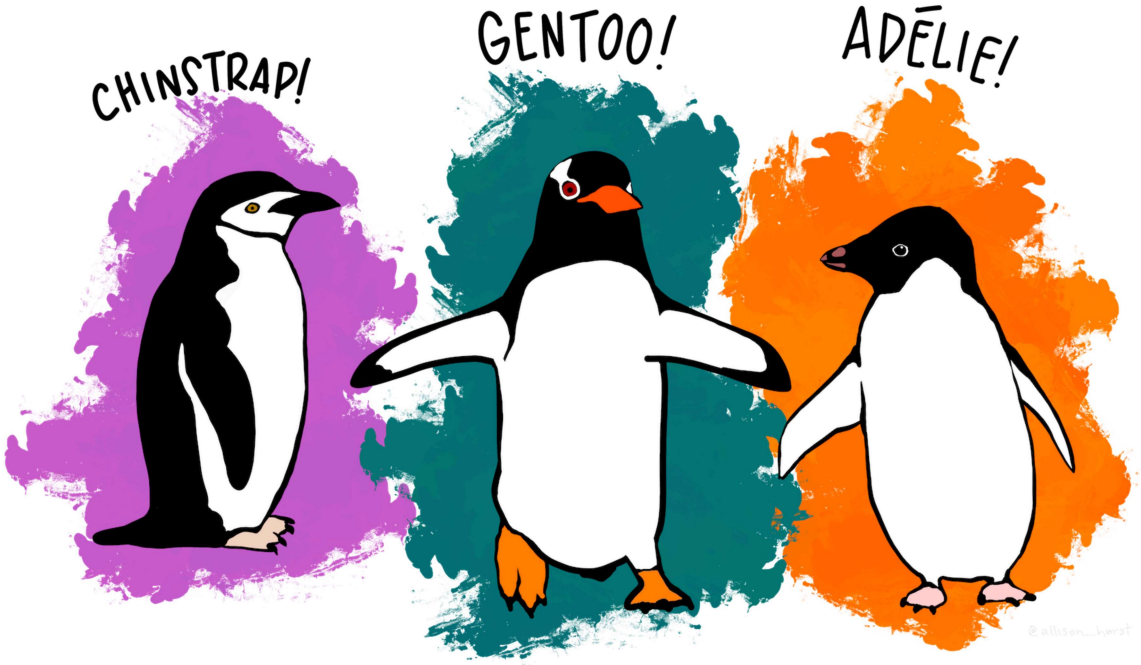
Solution: Turn your scripts into pipelines with `{targets}`!

Developing complex analyses in R

Solution: Turn your scripts into pipelines with `{targets}`!

Let's learn about `{targets}`

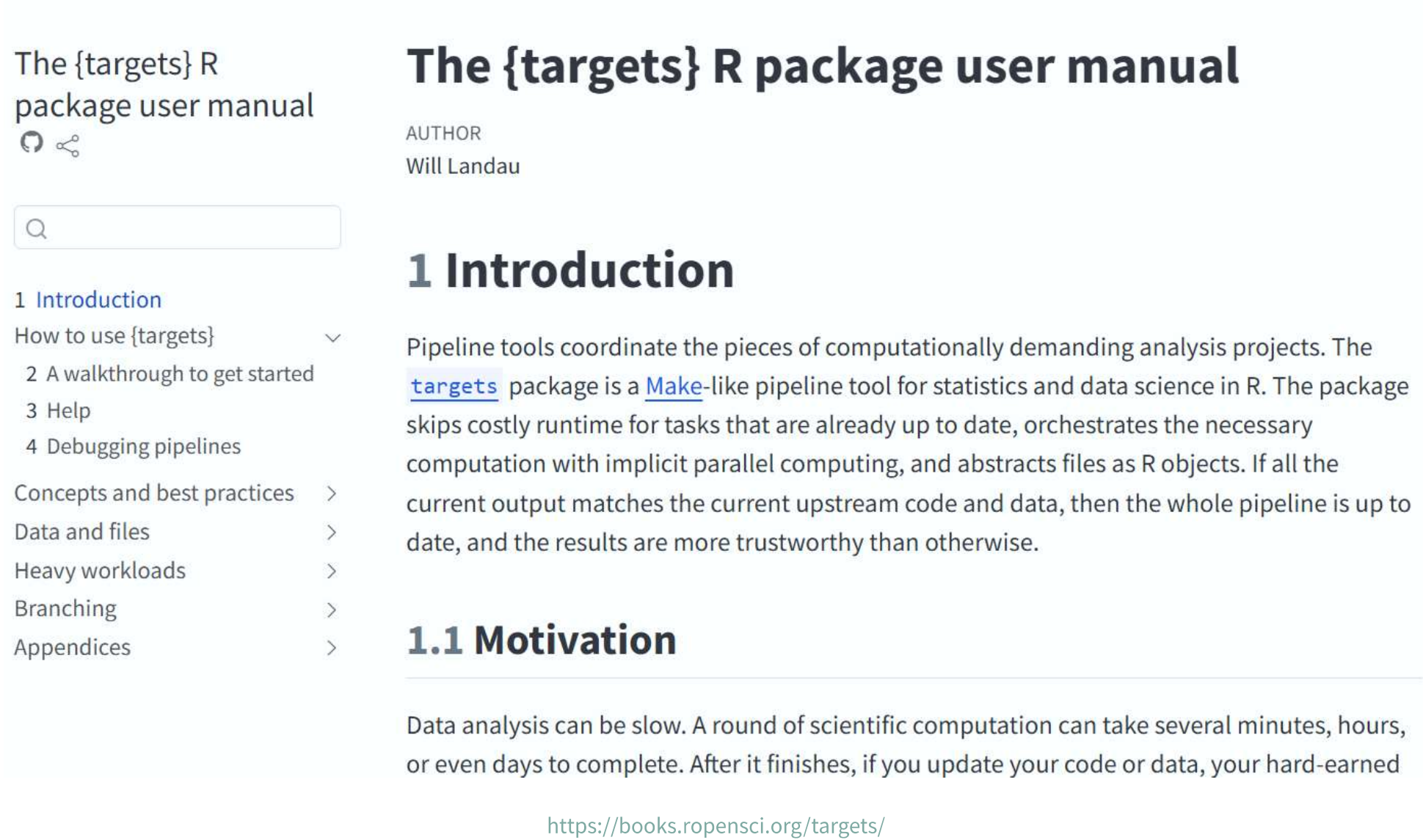
… through an example!



Artwork by @allison_horst

`palmerpenguins` dataset: size measurements for three penguin species observed in the Palmer Archipelago (Antarctica)

To go further



https://books.ropensci.org/targets/

Aside – more about setting up R projects

- previous talk: “Setting up a reproducible data analysis project in R–featuring GitHub, `{renv}`, `{targets}` and more”
- see the slides, recording, or blog post 😊

Project setup	• Creating a GitHub repository • Turning it into an R project • Initialising `{renv}` for the project • Setting up the directory structure • Populating the README
Tidying code	• Modularise code into functions • Turning code into a `{targets}` pipeline
Bonus	• Set up testing of input data with `{assertr}` • Automate report rendering with Quarto + `{targets}`

Let’s code!

I have written some code...

```
library(tidyverse)
library(janitor)
library(ggbeeswarm)
library(here)

# Reading and cleaning data
penguins_df <- read_csv(here("data/penguins_raw.csv"), show_col_types = FALSE) |>
  clean_names() |>
  mutate(
    species = word(species, 1),
    year = year(date_egg),
    sex = str_to_lower(sex),
    year = as.integer(year),
    body_mass_g = as.integer(body_mass_g),
    across(where(is.character), as.factor)
  ) |>
  select(
    species,
    island,
    year,
    sex,
    body_mass_g,
    bill_length_mm = culmen_length_mm,
```

I have written some code…

Good start, but often:

I have written some code…

Good start, but often:

- many more steps in the analysis → script becomes long and convoluted

I have written some code…

Good start, but often:

- many more steps in the analysis → script becomes long and convoluted
- harder to get an overview of the analysis, and to find things

I have written some code…

Good start, but often:

- many more steps in the analysis → script becomes long and convoluted
- harder to get an overview of the analysis, and to find things
- don’t want to re-run all the code everytime I make a change

I have written some code…

Good start, but often:

- many more steps in the analysis → script becomes long and convoluted
- harder to get an overview of the analysis, and to find things
- don’t want to re-run all the code everytime I make a change

Let’s turn it into a `{targets}` pipeline!

Step 1: Turn your code into functions

From:

```
# Reading and cleaning data
penguins_df <- read_csv(
  here("data/penguins_raw.csv"),
  show_col_types = FALSE
) |>
  clean_names() |>
  mutate(
    species = word(species, 1),
    year = year(date_egg),
    sex = str_to_lower(sex),
    year = as.integer(year),
    body_mass_g = as.integer(body_mass_g),
    across(where(is.character), as.factor)
  ) |>
  select(
    ## all relevant columns
  ) |>
  drop_na()
```

Step 1: Turn your code into functions

From:

```
# Reading and cleaning data
penguins_df <- read_csv(
  here("data/penguins_raw.csv"),
  show_col_types = FALSE
) |>
  clean_names() |>
  mutate(
    species = word(species, 1),
    year = year(date_egg),
    sex = str_to_lower(sex),
    year = as.integer(year),
    body_mass_g = as.integer(body_mass_g),
    across(where(is.character), as.factor)
  ) |>
  select(
    ## all relevant columns
  ) |>
  drop_na()
```

To:

```
read_data <- function(file) {
  readr::read_csv(
    file,
    show_col_types = FALSE
  ) |>
  janitor::clean_names() |>
  dplyr::mutate(
    species = stringr::word(species, 1),
    year = lubridate::year(date_egg),
    sex = stringr::str_to_lower(sex),
    year = as.integer(year),
    body_mass_g = as.integer(body_mass_g),
    dplyr::across(
      dplyr::where(is.character),
      as.factor
    )
  ) |>
  dplyr::select(
    ## all relevant columns
  ) |>
  tidyr::drop_na()
}
```

Step 1: Turn your code into functions

From:

```
# Reading and cleaning data
penguins_df <- read_csv(
  here("data/penguins_raw.csv"),
  show_col_types = FALSE
) |>
  clean_names() |>
  mutate(
    species = word(species, 1),
    year = year(date_egg),
    sex = str_to_lower(sex),
    year = as.integer(year),
    body_mass_g = as.integer(body_mass_g),
    across(where(is.character), as.factor)
  ) |>
  select(
    ## all relevant columns
  ) |>
  drop_na()
```

To:

```
read_data <- function(file) {
  readr::read_csv(
    file,
    show_col_types = FALSE
  ) |>
  janitor::clean_names() |>
  dplyr::mutate(
    species = stringr::word(species, 1),
    year = lubridate::year(date_egg),
    sex = stringr::str_to_lower(sex),
    year = as.integer(year),
    body_mass_g = as.integer(body_mass_g),
    dplyr::across(
      dplyr::where(is.character),
      as.factor
    )
  ) |>
  dplyr::select(
    ## all relevant columns
  ) |>
  tidyr::drop_na()
}
```

Step 1: Turn your code into functions

From:

```
# Reading and cleaning data
penguins_df <- read_csv(
  here("data/penguins_raw.csv"),
  show_col_types = FALSE
) |>
  clean_names() |>
  mutate(
    species = word(species, 1),
    year = year(date_egg),
    sex = str_to_lower(sex),
    year = as.integer(year),
    body_mass_g = as.integer(body_mass_g),
    across(where(is.character), as.factor)
  ) |>
  select(
    ## all relevant columns
  ) |>
  drop_na()
```

To:

```
read_data <- function(file) {
  readr::read_csv(
    file,
    show_col_types = FALSE
  ) |>
  janitor::clean_names() |>
  dplyr::mutate(
    species = stringr::word(species, 1),
    year = lubridate::year(date_egg),
    sex = stringr::str_to_lower(sex),
    year = as.integer(year),
    body_mass_g = as.integer(body_mass_g),
    dplyr::across(
      dplyr::where(is.character),
      as.factor
    )
  ) |>
  dplyr::select(
    ## all relevant columns
  ) |>
  tidyr::drop_na()
}
```

Step 1: Turn your code into functions

Don’t forget to document your functions! (`{roxygen}`-style)

Step 1: Turn your code into functions

Don't forget to document your functions! ({roxygen}-style)

```
#' Read and clean data
#'
#' Reads in the penguins data, renames and selects relevant columns. The
#' following transformations are applied to the data:
#' * only keep species common name
#' * extract observation year
#' * remove rows with missing values
#'
#' @param file Character, path to the penguins data .csv file.
#' @returns A tibble.
read_data <- function(file) {
  readr::read_csv(file, show_col_types = FALSE) |>
  janitor::clean_names() |>
  dplyr::mutate(
    ## modifying columns
  ) |>
  dplyr::select(
    ## all relevant columns
  ) |>
  tidyr::drop_na()
}
```

Step 1: Turn your code into functions

Improved script:

R/helper_functions.R

```
#' Read and clean data
#'
#' ...
read_data <- function(file) { ... }

#' Violin plot of variable per species and s
#'
#' ...
violin_plot <- function(df, yvar) { ... }

#' Scatter plot of bill length vs depth
#'
#' ...
plot_bill_length_depth <- function(df) { ...
```

analysis/first_script.R

```
library(here)

source(here("R/helper_functions.R"))

penguins_df <- read_data(
  here("data/penguins_raw.csv")
)

body_mass_plot <- violin_plot(
  penguins_df,
  body_mass_g
)

flipper_length_plot <- violin_plot(
  penguins_df,
  flipper_length_mm
)

bill_scatterplot <- plot_bill_length_depth(
  penguins_df
)
```

Step 2: Turn your main script into a `{targets}` pipeline!

From:

```
library(here)

source(here("R/helper_functions.R"))

penguins_df <- read_data(here("data/penguins_raw.csv"))

body_mass_plot <- violin_plot(penguins_df, body_mass_g)

flipper_length_plot <- violin_plot(penguins_df, flipper_length_mm)

bill_scatterplot <- plot_bill_length_depth(penguins_df)
```

Step 2: Turn your main script into a `{targets}` pipeline!

To:

```
library(targets)
library(here)

source(here("R/helper_functions.R"))

list(
  tar_target(penguins_raw_file, here("data/penguins_raw.csv"), format = "file"),

  tar_target(penguins_df, read_data(penguins_raw_file)),

  tar_target(body_mass_plot, violin_plot(penguins_df, body_mass_g)),

  tar_target(flipper_length_plot, violin_plot(penguins_df, flipper_length_mm)),

  tar_target(bill_scatterplot, plot_bill_length_depth(penguins_df))
)
```

Step 2: Turn your main script into a `{targets}` pipeline!

To:

```
library(targets)
library(here)

source(here("R/helper_functions.R"))

list(
  tar_target(penguins_raw_file, here("data/penguins_raw.csv"), format = "file"),

  tar_target(penguins_df, read_data(penguins_raw_file)),

  tar_target(body_mass_plot, violin_plot(penguins_df, body_mass_g)),

  tar_target(flipper_length_plot, violin_plot(penguins_df, flipper_length_mm)),

  tar_target(bill_scatterplot, plot_bill_length_depth(penguins_df))
)
```

Step 2: Turn your main script into a `{targets}` pipeline!

To:

```
library(targets)
library(here)

source(here("R/helper_functions.R"))

list(
  tar_target(penguins_raw_file, here("data/penguins_raw.csv"), format = "file"),

  tar_target(penguins_df, read_data(penguins_raw_file)),

  tar_target(body_mass_plot, violin_plot(penguins_df, body_mass_g)),

  tar_target(flipper_length_plot, violin_plot(penguins_df, flipper_length_mm)),

  tar_target(bill_scatterplot, plot_bill_length_depth(penguins_df))
)
```

Step 2: Turn your main script into a `{targets}` pipeline!

To:

```
library(targets)
library(here)

source(here("R/helper_functions.R"))

list(
  tar_target(penguins_raw_file, here("data/penguins_raw.csv"), format = "file"),

  tar_target(penguins_df, read_data(penguins_raw_file)),

  tar_target(body_mass_plot, violin_plot(penguins_df, body_mass_g)),

  tar_target(flipper_length_plot, violin_plot(penguins_df, flipper_length_mm)),

  tar_target(bill_scatterplot, plot_bill_length_depth(penguins_df))
)
```

Step 2: Turn your main script into a `{targets}` pipeline!

To:

```
library(targets)
library(here)

source(here("R/helper_functions.R"))

list(
  tar_target(penguins_raw_file, here("data/penguins_raw.csv"), format = "file"),

  tar_target(penguins_df, read_data(penguins_raw_file)),

  tar_target(body_mass_plot, violin_plot(penguins_df, body_mass_g)),

  tar_target(flipper_length_plot, violin_plot(penguins_df, flipper_length_mm)),

  tar_target(bill_scatterplot, plot_bill_length_depth(penguins_df))
)
```

Step 2: Turn your main script into a `{targets}` pipeline!

To:

```
library(targets)
library(here)

source(here("R/helper_functions.R"))

list(
  tar_target(penguins_raw_file, here("data/penguins_raw.csv"), format = "file"),

  tar_target(penguins_df, read_data(penguins_raw_file)),

  tar_target(body_mass_plot, violin_plot(penguins_df, body_mass_g)),

  tar_target(flipper_length_plot, violin_plot(penguins_df, flipper_length_mm)),

  tar_target(bill_scatterplot, plot_bill_length_depth(penguins_df))
)
```

Step 2: Turn your main script into a `{targets}` pipeline!

To:

```
library(targets)
library(here)

source(here("R/helper_functions.R"))

list(
  tar_target(penguins_raw_file, here("data/penguins_raw.csv"), format = "file"),

  tar_target(penguins_df, read_data(penguins_raw_file)),

  tar_target(body_mass_plot, violin_plot(penguins_df, body_mass_g)),

  tar_target(flipper_length_plot, violin_plot(penguins_df, flipper_length_mm)),

  tar_target(bill_scatterplot, plot_bill_length_depth(penguins_df))
)
```

Step 2: Turn your main script into a `{targets}` pipeline!

To:

```
library(targets)
library(here)

source(here("R/helper_functions.R"))

list(
  tar_target(penguins_raw_file, here("data/penguins_raw.csv"), format = "file"),

  tar_target(penguins_df, read_data(penguins_raw_file)),

  tar_target(body_mass_plot, violin_plot(penguins_df, body_mass_g)),

  tar_target(flipper_length_plot, violin_plot(penguins_df, flipper_length_mm)),

  tar_target(bill_scatterplot, plot_bill_length_depth(penguins_df))
)
```

Step 2: Turn your main script into a `{targets}` pipeline!

To:

```
library(targets)
library(here)

source(here("R/helper_functions.R"))

list(
  tar_target(penguins_raw_file, here("data/penguins_raw.csv"), format = "file"),

  tar_target(penguins_df, read_data(penguins_raw_file)),

  tar_target(body_mass_plot, violin_plot(penguins_df, body_mass_g)),

  tar_target(flipper_length_plot, violin_plot(penguins_df, flipper_length_mm)),

  tar_target(bill_scatterplot, plot_bill_length_depth(penguins_df))
)
```

Step 2: Turn your main script into a `{targets}` pipeline!

Aside: where should my targets script live?

Step 2: Turn your main script into a `{targets}` pipeline!

Aside: where should my targets script live?

- default would be in the `_targets.R` file in the main directory

Step 2: Turn your main script into a `{targets}` pipeline!

Aside: where should my targets script live?

- default would be in the `_targets.R` file in the main directory
- to choose a custom folder and file name, need to specify targets configuration:

Step 2: Turn your main script into a `{targets}` pipeline!

Aside: where should my targets script live?

- default would be in the `_targets.R` file in the main directory
- to choose a custom folder and file name, need to specify targets configuration:

In the console, run (from main directory):

```
targets::tar_config_set(script = "analysis/_targets.R", store = "analysis/_targets")
```

Step 2: Turn your main script into a `{targets}` pipeline!

Aside: where should my targets script live?

- default would be in the `_targets.R` file in the main directory
- to choose a custom folder and file name, need to specify targets configuration:

In the console, run (from main directory):

```
targets::tar_config_set(script = "analysis/_targets.R", store = "analysis/_targets")
```

Will create a `_targets.yaml` file:

_target.yaml

```
main:
  script: analysis/_targets.R
  store: analysis/_targets
```

Visualise your pipeline

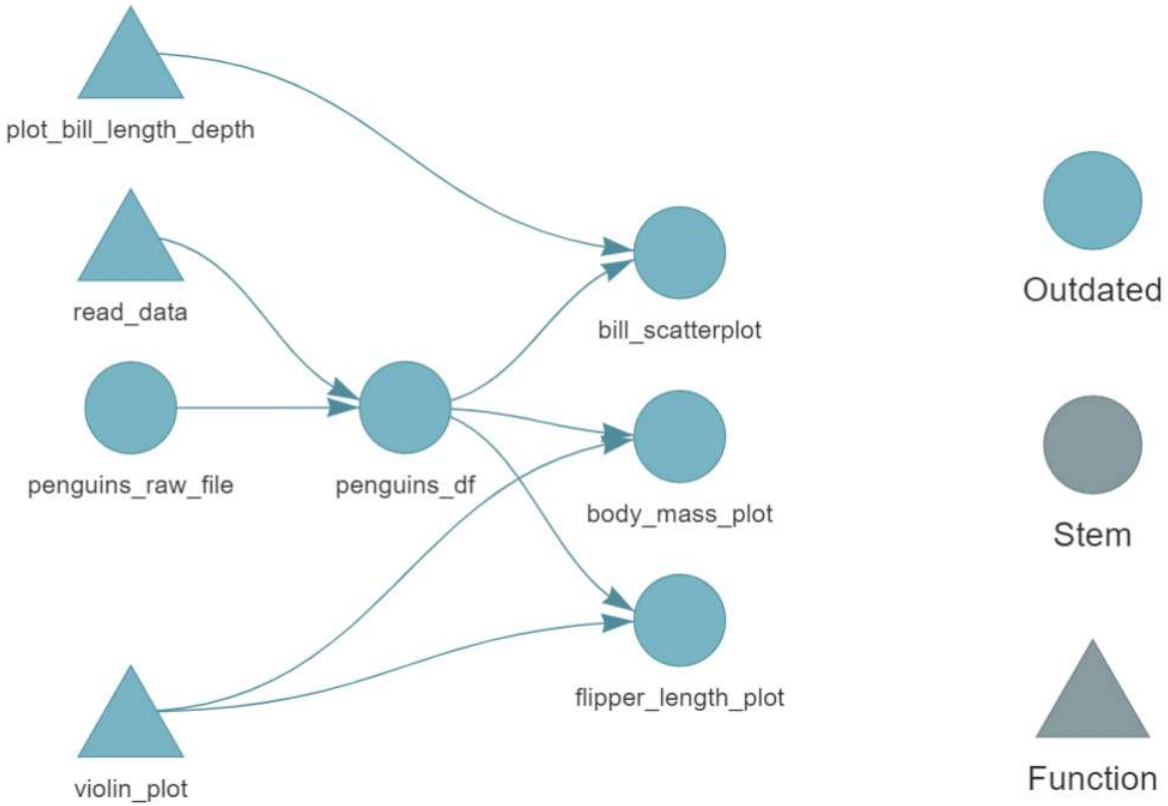
In the console, run:

```
targets::tar_visnetwork()
```

Visualise your pipeline

In the console, run:

```
targets::tar_visnetwork()
```



Execute your pipeline

In the console, run:

```
targets::tar_make()
```

Execute your pipeline

In the console, run:

```
targets::tar_make()
```

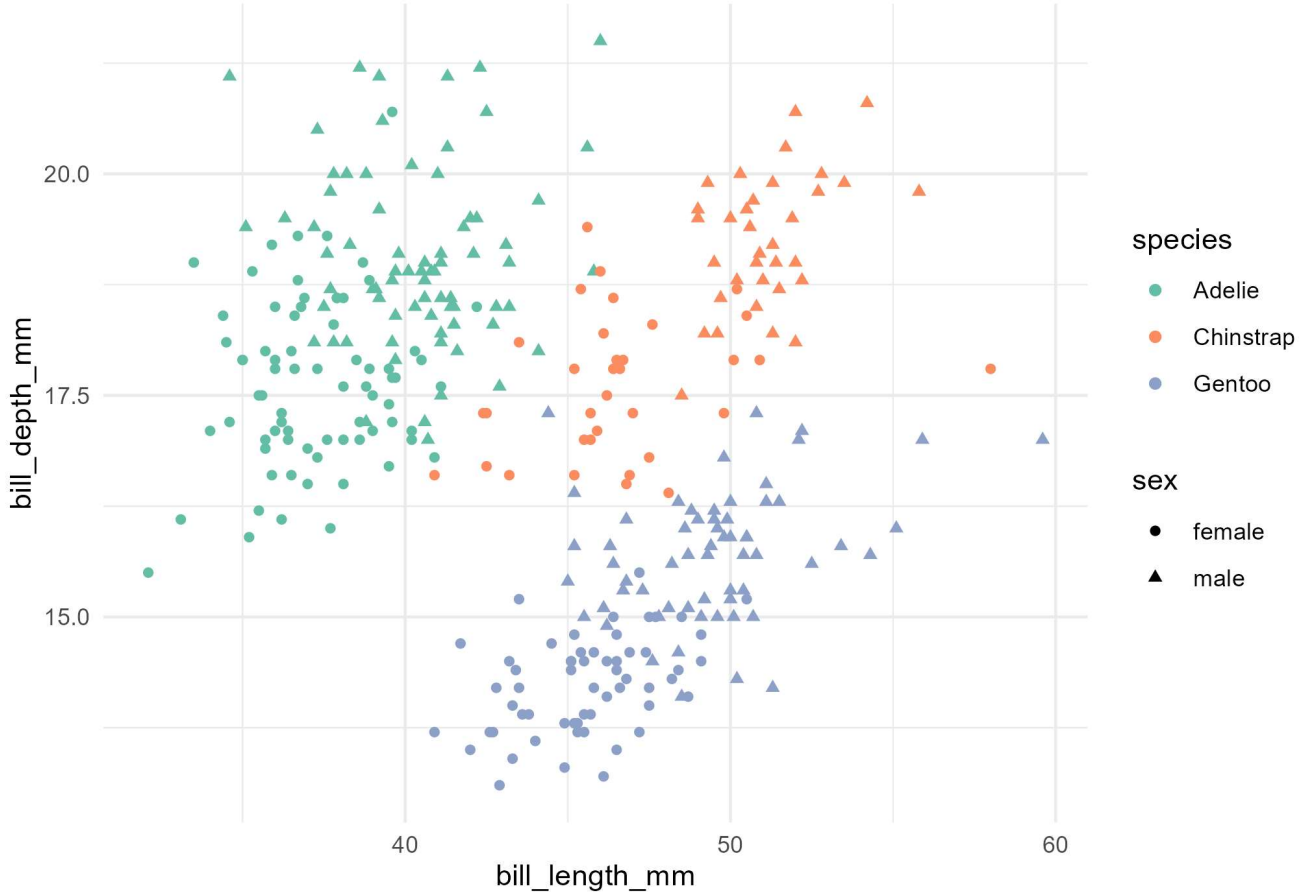
```
here() starts at C:/Users/hrpoab/Desktop/GitHub/palmerpenguins_analysis
> dispatched target penguins_raw_file
o completed target penguins_raw_file [0 seconds]
> dispatched target penguins_df
o completed target penguins_df [0.85 seconds]
> dispatched target body_mass_plot
o completed target body_mass_plot [0.16 seconds]
> dispatched target bill_scatterplot
o completed target bill_scatterplot [0.02 seconds]
> dispatched target flipper_length_plot
o completed target flipper_length_plot [0.02 seconds]
> ended pipeline [1.31 seconds]
```

Get the pipeline results

```
targets::tar_read(bill_scatterplot)
```

Get the pipeline results

```
targets::tar_read(bill_scatterplot)
```



Change in a step

> *Hi Olivia,*
>
> *Great work! Just a minor comment, could you change the colours in the bill length/depth scatter-plot? It's hard to see the difference between the species.*

Change in a step

> *Hi Olivia,*
>
> *Great work! Just a minor comment, could you change the colours in the bill length/depth scatter-plot? It's hard to see the difference between the species.*

R/helper_functions.R

```
plot_bill_length_depth <- function(df) {
  df |>
    ggplot2::ggplot(
      ggplot2::aes(
        x = bill_length_mm,
        y = bill_depth_mm,
        colour = species,
        shape = sex
        )
    ) +
    ggplot2::geom_point() +
    ggplot2::scale_colour_brewer(palette = "Set2") +
    ggplot2::theme_minimal()
}
```

Change in a step

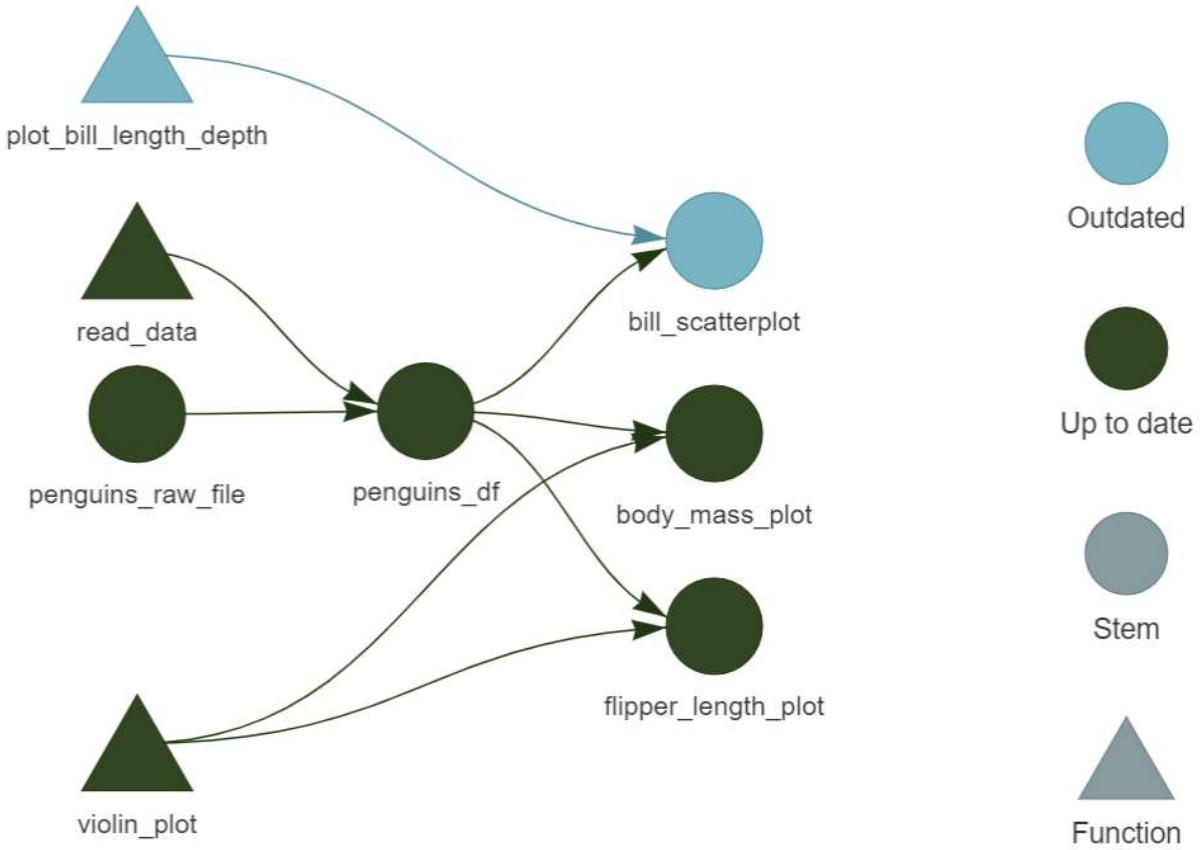
> *Hi Olivia,*
>
> *Great work! Just a minor comment, could you change the colours in the bill length/depth scatter-plot? It's hard to see the difference between the species.*

R/helper_functions.R

```r
plot_bill_length_depth <- function(df) {
  df |>
    ggplot2::ggplot(
      ggplot2::aes(
        x = bill_length_mm,
        y = bill_depth_mm,
        colour = species,
        shape = sex
        )
    ) +
    ggplot2::geom_point() +
    ggplot2::scale_colour_brewer(palette = "Set1") +
    ggplot2::theme_minimal()
}
```

Change in a step

```
targets::tar_visnetwork()
```



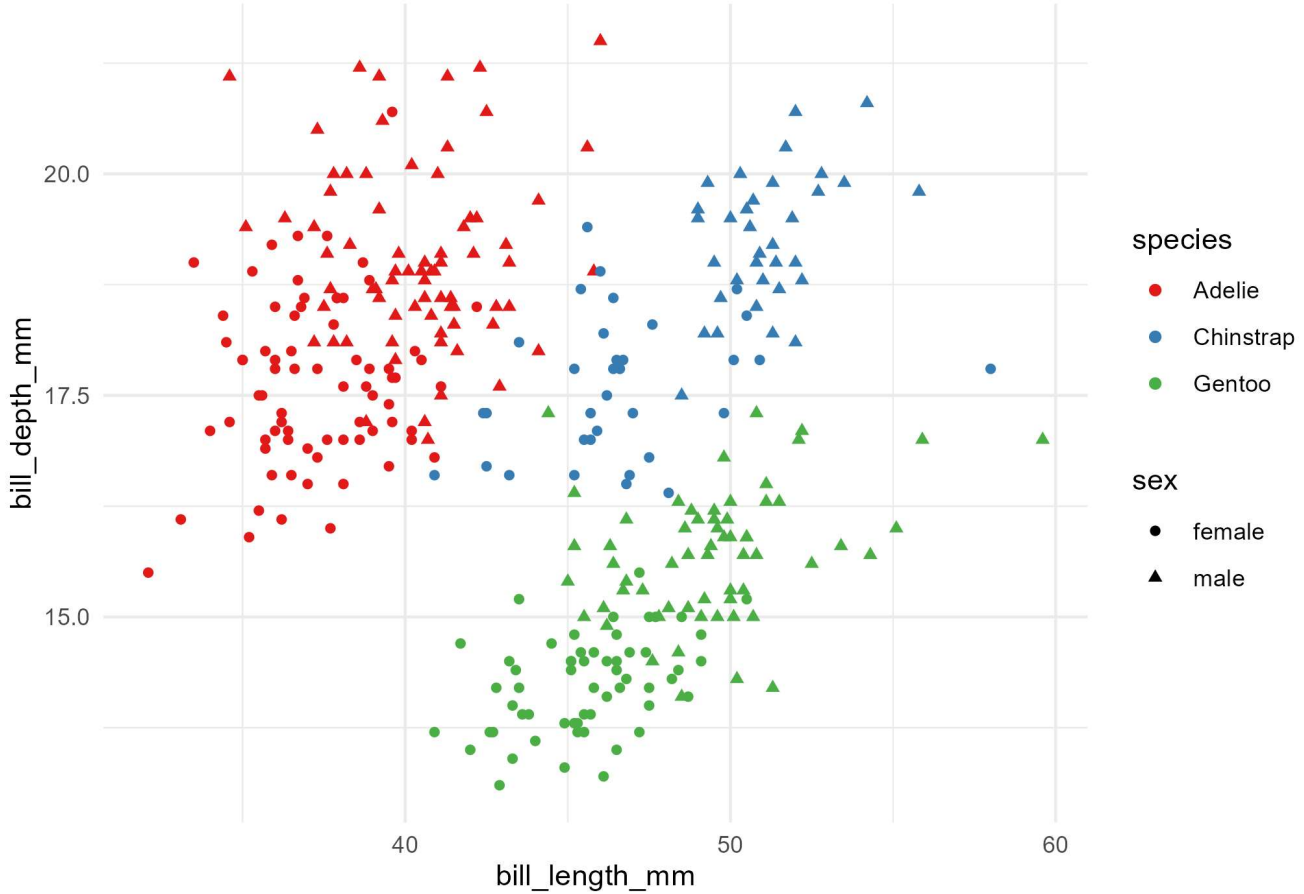
Change in a step

```
targets::tar_make()
```

```
here() starts at C:/Users/hrpoab/Desktop/GitHub/palmerpenguins_analysis
v skipped target penguins_raw_file
v skipped target penguins_df
v skipped target body_mass_plot
> dispatched target bill_scatterplot
o completed target bill_scatterplot [0.35 seconds]
v skipped target flipper_length_plot
> ended pipeline [0.53 seconds]
```

Change in a step

```
targets::tar_read(bill_scatterplot)
```



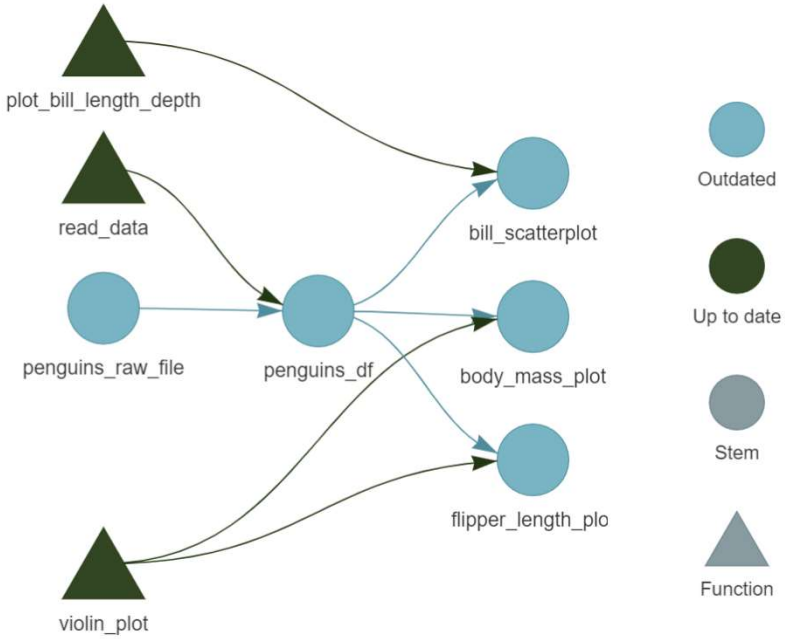
Change in the data

> *Hi Olivia,*
>
> *Oopsie! We realised there was a mistake in the original data file. Here is the updated spreadsheet, could you re-run the analysis with this version?*

Change in the data

> *Hi Olivia,*
>
> *Oopsie! We realised there was a mistake in the original data file. Here is the updated spreadsheet, could you re-run the analysis with this version?*

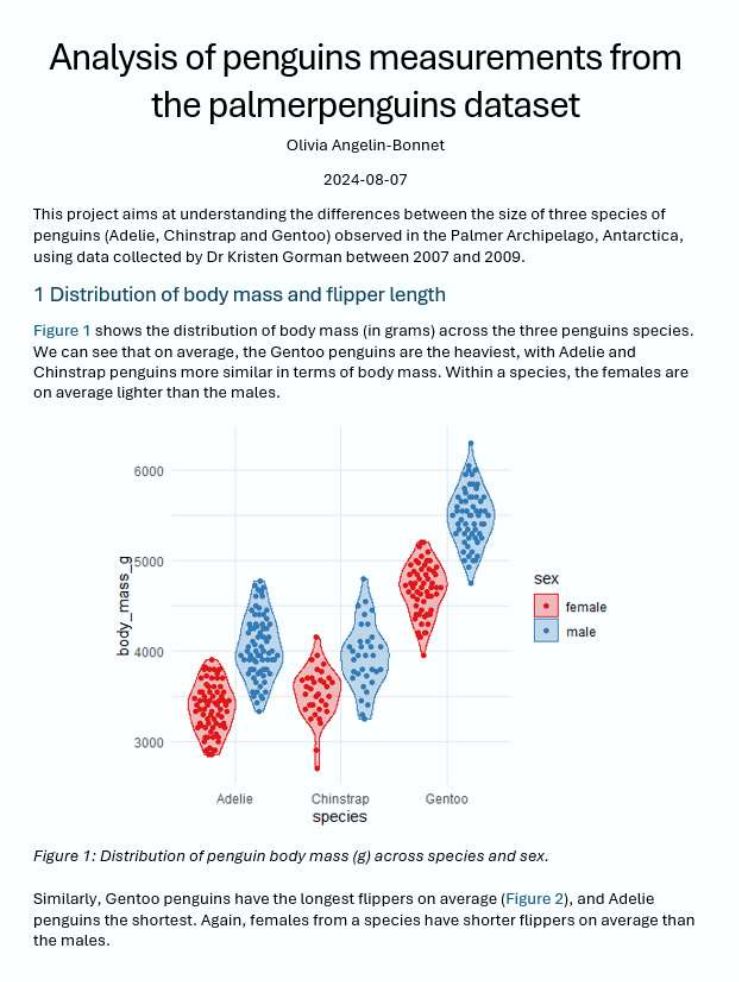
```
targets::tar_visnetwork()
```



Writing a report with Quarto

reports/palmerpenguins_report.qmd

````
---
title: |
  Analysis of penguins measurements from the
  palmerpenguins dataset
author: "Olivia Angelin-Bonnet"
date: today
format:
  docx:
    number-sections: true
---

```{r setup}
#| include: false

library(knitr)
opts_chunk$set(echo = FALSE)
```

This project aims at understanding the differences betwee
the size of three species of penguins (Adelie, Chinstrap
and Gentoo) observed in the Palmer Archipelago, Antarctic
using data collected by Dr Kristen Gorman between 2007 an
2009.
````



Analysis of penguins measurements from the palmerpenguins dataset

Olivia Angelin-Bonnet

2024-08-07

This project aims at understanding the differences between the size of three species of penguins (Adelie, Chinstrap and Gentoo) observed in the Palmer Archipelago, Antarctica, using data collected by Dr Kristen Gorman between 2007 and 2009.

1 Distribution of body mass and flipper length

Figure 1 shows the distribution of body mass (in grams) across the three penguins species. We can see that on average, the Gentoo penguins are the heaviest, with Adelie and Chinstrap penguins more similar in terms of body mass. Within a species, the females are on average lighter than the males.

Figure 1: Distribution of penguin body mass (g) across species and sex.

Similarly, Gentoo penguins have the longest flippers on average (Figure 2), and Adelie penguins the shortest. Again, females from a species have shorter flippers on average than the males.

Quarto + `{targets}`

Two advantages of using a Quarto document alongside `{targets}`:

Quarto + `{targets}`

Two advantages of using a Quarto document alongside `{targets}`:

- can read in results from targets pipeline inside the report → no computation done during report generation

Quarto + `{targets}`

Two advantages of using a Quarto document alongside `{targets}`:

- can read in results from targets pipeline inside the report → no computation done during report generation
- can add the rendering of the report as a step in the pipeline → ensures that the report is always up-to-date

Quarto + {targets}

Two steps to use the {targets} pipeline results in a Quarto document:

reports/palmerpenguins_report.qmd

````
```{r setup}
#| include: false

library(knitr)

opts_chunk$set(echo = FALSE)
```

@fig-body-mass shows...

```{r fig-body-mass}
#| fig-cap: "Distribution ..."

# code for plot
```
````

Quarto + {targets}

Two steps to use the {targets} pipeline results in a Quarto document:

- Make sure the report 'sees' the project root directory

reports/palmerpenguins_report.qmd

````
```{r setup}
#| include: false

library(knitr)
library(here)

opts_chunk$set(echo = FALSE)
opts_knit$set(root.dir = here())
```

@fig-body-mass shows...

```{r fig-body-mass}
#| fig-cap: "Distribution ..."

# code for plot
```
````

Quarto + {targets}

Two steps to use the {targets} pipeline results in a Quarto document:

- Make sure the report 'sees' the project root directory
- Read targets objects with `targets::tar_read()`

reports/palmerpenguins_report.qmd

````
```{r setup}
#| include: false

library(knitr)
library(here)
library(targets)

opts_chunk$set(echo = FALSE)
opts_knit$set(root.dir = here())
```

@fig-body-mass shows...

```{r fig-body-mass}
#| fig-cap: "Distribution ..."

tar_read(body_mass_plot)
```
````

Quarto + {targets}

Adding the Quarto report as a step in the pipeline (need {tarchetypes} and {quarto} packages installed):

analysis/_targets.R

```
list(
  tar_target(penguins_raw_file, here("data/penguins_raw.csv"), format = "file"),

  tar_target(penguins_df, read_data(penguins_raw_file)),

  tar_target(body_mass_plot, violin_plot(penguins_df, body_mass_g)),

  tar_target(flipper_length_plot, violin_plot(penguins_df, flipper_length_mm)),

  tar_target(bill_scatterplot, plot_bill_length_depth(penguins_df))
)
```

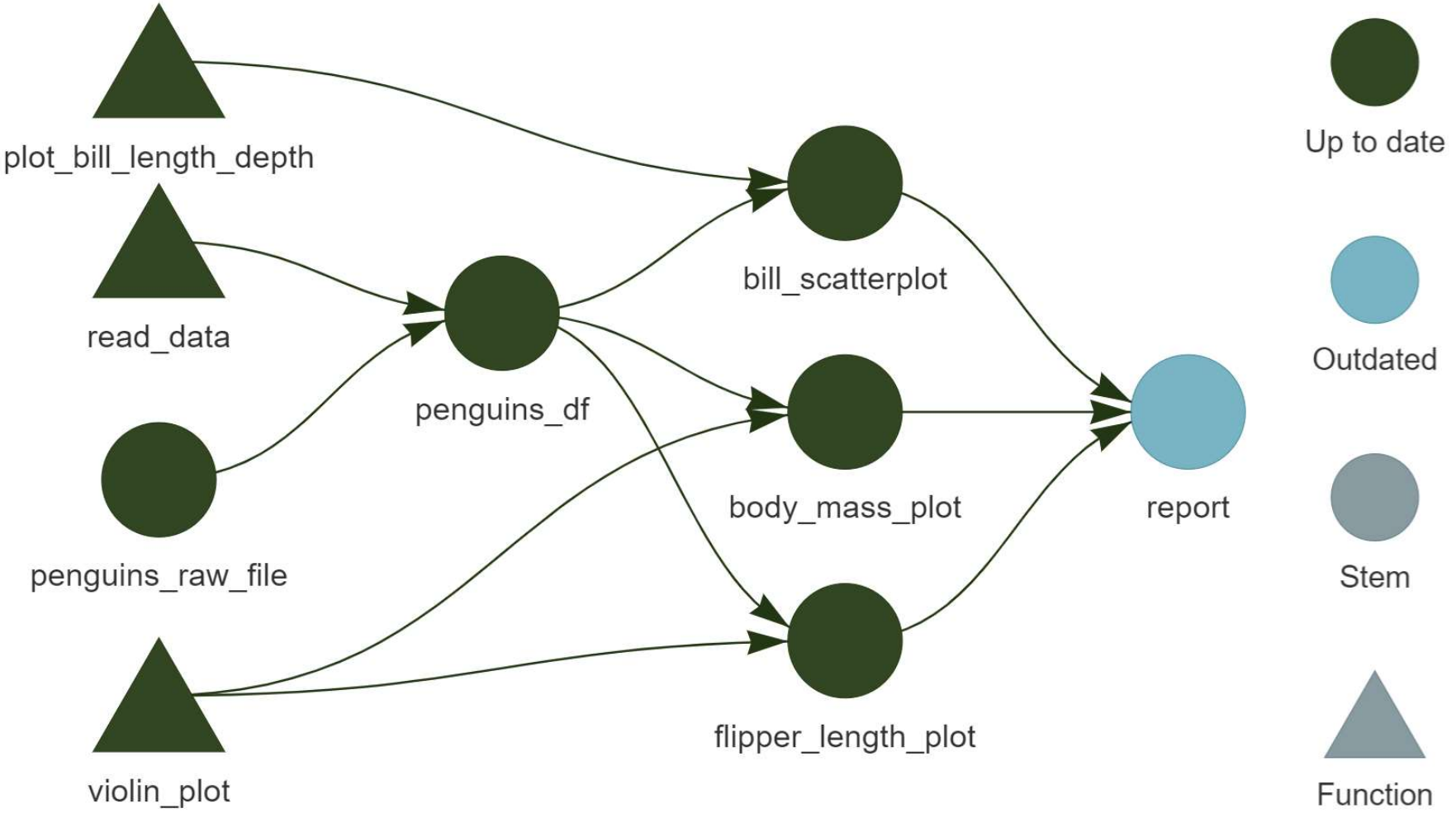
Quarto + {targets}

Adding the Quarto report as a step in the pipeline (need {tarchetypes} and {quarto} packages installed):

analysis/_targets.R

```
list(
  tar_target(penguins_raw_file, here("data/penguins_raw.csv"), format = "file"),

  tar_target(penguins_df, read_data(penguins_raw_file)),

  tar_target(body_mass_plot, violin_plot(penguins_df, body_mass_g)),

  tar_target(flipper_length_plot, violin_plot(penguins_df, flipper_length_mm)),

  tar_target(bill_scatterplot, plot_bill_length_depth(penguins_df)),

  tar_quarto(report, here("reports/palmerpenguins_report.qmd"))
)
```

Quarto + {targets}



Further reading

- The blog post version of this talk 😄
- The targets user manual manual
- Bruno Rodrigues' book, Building reproducible analytical pipelines with R
- The Carpentries Incubator's Introduction to targets workshop

Thank you for your attention!

olivia.angelin-bonnet@plantandfood.co.nz

Presentation disclaimer

Presentation for
Rladies Sydney, online, 7 April 2025

Publication data:
Angelin-Bonnet O. April 2025. Reproducible analysis pipelines with `{targets}`. A Plant & Food Research presentation. SPTS No. 26935.

Presentation prepared by:
Olivia Angelin-Bonnet
Statistical Scientist, Data Science
April 2025

Presentation approved by:
Mark Wohlers
Science Group Leader, Data Science
April 2025

For more information contact:
Olivia Angelin-Bonnet
DDI: +64 6 355 6156
Email: olivia.angelin-bonnet@plantandfood.co.nz

This report has been prepared by The New Zealand Institute for Plant and Food Research Limited (Plant & Food Research).
Head Office: 120 Mt Albert Road, Sandringham, Auckland 1025, New Zealand, Tel: +64 9 925 7000, Fax: +64 9 925 7001.
www.plantandfood.co.nz