# DATA2002

## Getting started with R


Garth Tarr
The University of Sydney

# DATA2002 and R

To successfully complete DATA2002 you will need to become reasonably proficient in R, an open source statistical computing language.

To get started:

1. Download and install R version **4.3.0** or higher from the Comprehensive R Archive Network (CRAN).
2. Download and install RStudio Desktop (Free Open Source Licence) version **2023.06.0** or higher.

> **Note**
>
> If you have previously installed R and/or RStudio for a previous unit, it's worth going through the above steps again to make sure you have the latest versions on your machine. Some features we will use are only available on the most recent versions.

# R packages

- While R by itself is very powerful, we will use many different **packages** over the semester.
- A **package** is a library of functions that add additional functionality.
- Most packages are available from CRAN and can be installed at the R console using `install.packages("package_name")`.
- R will also check if you have the required dependencies on your machine and if needed install any additional packages required.
- To use a function from an installed package, you load the package using `library("package_name")` and then you can refer to `function_name()` or you can explicitly use `package_name::function_name()`. To get started install the following packages:

```
install.packages("tidyverse")
install.packages("palmerpenguins")
```

# Palmer penguins

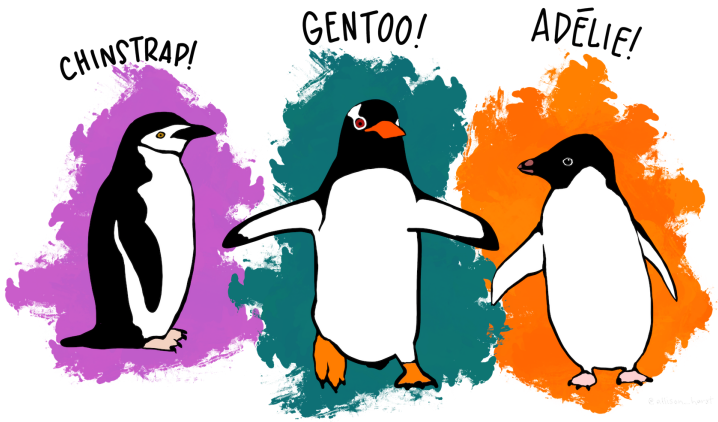
The `penguins` data set was collected and made available by Dr. Kristen Gorman and the Palmer Station, Antarctica LTER, a member of the Long Term Ecological Research Network (Horst et al., 2020).

It is available in the **palmerpenguins** package.

```
# install.packages("palmerpenguins")
library(palmerpenguins)
```

To find out more about the `penguins` data set:

```
help(penguins_raw, package = "palmerpenguins")
# or more simply
?penguins
```




Artwork by @allison_horst

# A quick look at the data

The `glimpse()` function from the **pillar** package (and re-exported by **dplyr**) gives us a quick overview of the data frame (Wickham et al., 2019).

```
library(tidyverse)
dplyr::glimpse(penguins_raw)
```

```
Rows: 344
Columns: 17
$ studyName             <chr> "PAL0708", "PAL0708", "PAL0708", "PAL0…
$ `Sample Number`       <dbl> 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12,…
$ Species               <chr> "Adelie Penguin (Pygoscelis adeliae)",…
$ Region                <chr> "Anvers", "Anvers", "Anvers", "Anvers"…
$ Island                <chr> "Torgersen", "Torgersen", "Torgersen",…
$ Stage                 <chr> "Adult, 1 Egg Stage", "Adult, 1 Egg St…
$ `Individual ID`       <chr> "N1A1", "N1A2", "N2A1", "N2A2", "N3A1"…
$ `Clutch Completion`   <chr> "Yes", "Yes", "Yes", "Yes", "Yes", "Ye…
$ `Date Egg`            <date> 2007-11-11, 2007-11-11, 2007-11-16, 2…
$ `Culmen Length (mm)`  <dbl> 39.1, 39.5, 40.3, NA, 36.7, 39.3, 38.9…
$ `Culmen Depth (mm)`   <dbl> 18.7, 17.4, 18.0, NA, 19.3, 20.6, 17.8…
$ `Flipper Length (mm)` <dbl> 181, 186, 195, NA, 193, 190, 181, 195,…
$ `Body Mass (g)`       <dbl> 3750, 3800, 3250, NA, 3450, 3650, 3625…
$ Sex                   <chr> "MALE", "FEMALE", "FEMALE", NA, "FEMAL…
```

# Cleaning the Palmer penguins column names

The **janitor** package has a bunch of functions to help with standardising and summarising data frames (Firke, 2021). Here we standardise the column names: all lower case and no spaces.

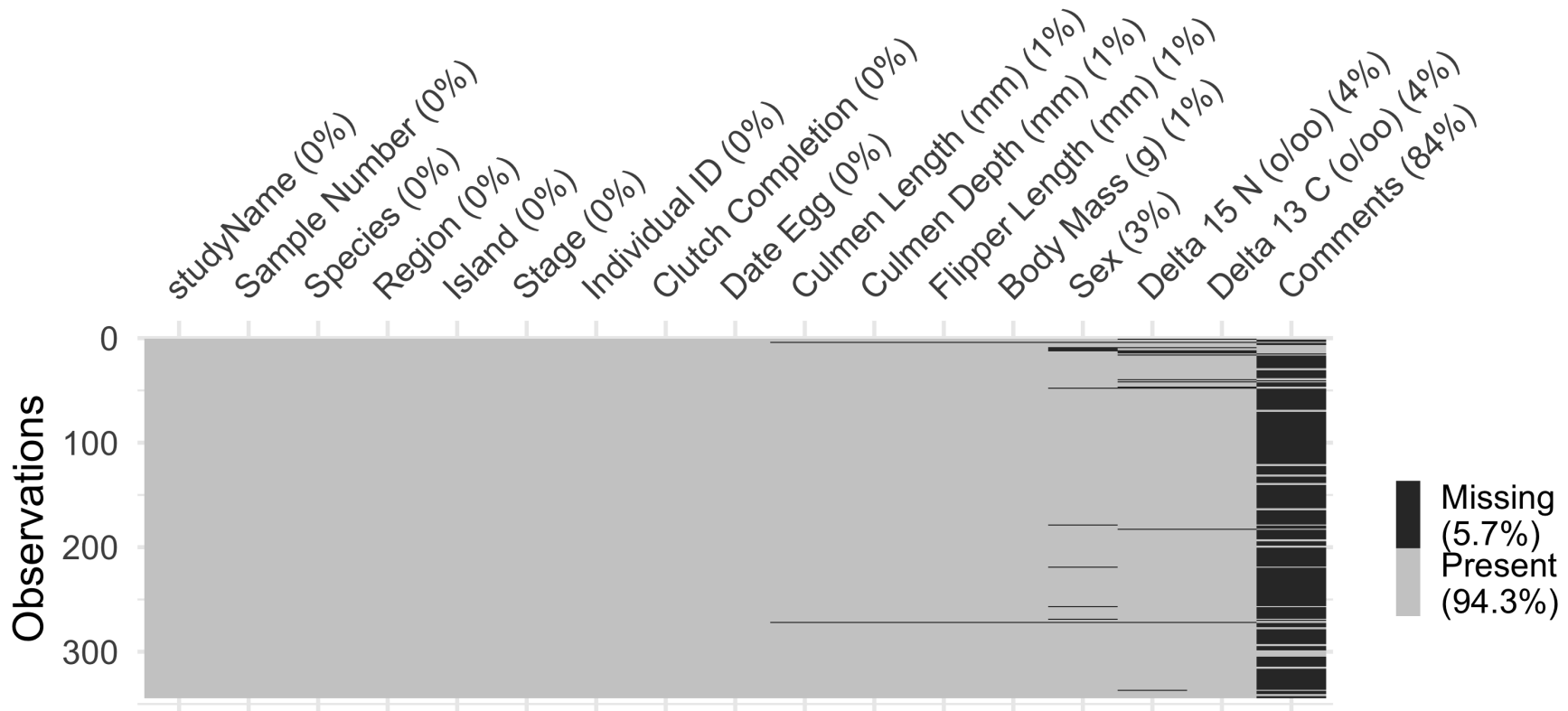
```
penguins_clean = janitor::clean_names(penguins_raw)
dplyr::glimpse(penguins_clean)
```

```
Rows: 344
Columns: 17
$ study_name        <chr> "PAL0708", "PAL0708", "PAL0708", "PAL0708"…
$ sample_number     <dbl> 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13,…
$ species           <chr> "Adelie Penguin (Pygoscelis adeliae)", "Ad…
$ region            <chr> "Anvers", "Anvers", "Anvers", "Anvers", "A…
$ island            <chr> "Torgersen", "Torgersen", "Torgersen", "To…
$ stage             <chr> "Adult, 1 Egg Stage", "Adult, 1 Egg Stage"…
$ individual_id     <chr> "N1A1", "N1A2", "N2A1", "N2A2", "N3A1", "N…
$ clutch_completion <chr> "Yes", "Yes", "Yes", "Yes", "Yes", "Yes", …
$ date_egg          <date> 2007-11-11, 2007-11-11, 2007-11-16, 2007-…
$ culmen_length_mm  <dbl> 39.1, 39.5, 40.3, NA, 36.7, 39.3, 38.9, 39…
$ culmen_depth_mm   <dbl> 18.7, 17.4, 18.0, NA, 19.3, 20.6, 17.8, 19…
$ flipper_length_mm <dbl> 181, 186, 195, NA, 193, 190, 181, 195, 193…
$ body_mass_g       <dbl> 3750, 3800, 3250, NA, 3450, 3650, 3625, 46…
$ sex               <chr> "MALE", "FEMALE", "FEMALE", NA, "FEMALE", …
```

# Missing data?

We can use the **visdat** package to give a visual overview of the missingness in a data frame (Tierney, 2017).

```
# install.packages("visdat")
visdat::vis_miss(penguins_raw)
```



# Missing data?

For simplicity we'll remove any observations (rows) that have missing values.

```
penguins_clean = tidyr::drop_na(penguins_clean, sex)
```

Compare the number of observations between the original and the clean data frame:

```
nrow(penguins)
```

```
[1] 344
```

```
nrow(penguins_clean)
```

```
[1] 333
```

We have made two changes to `penguins_raw` to get to `penguins_clean`. We can summarise this in an easy to read **pipeline**:

```
penguins_clean = penguins_raw |>
  janitor::clean_names() |>
  tidyr::drop_na(sex)
```

We'll often use the pipe operator, `|>`. Read more about it in the Pipes section of Wickham et al. (2023).

# Palmer penguins cross tabulation

The **janitor** package has a range of functions that help with importing data and doing quick tabulations (Firke, 2021).

```
library(janitor)
penguins_clean |>
  janitor::tabyl(species, sex) |>
  janitor::adorn_totals(where = c("row", "col"))
```

| species | FEMALE | MALE | Total |
|---|---|---|---|
| Adelie Penguin (Pygoscelis adeliae) | 73 | 73 | 146 |
| Chinstrap penguin (Pygoscelis antarctica) | 34 | 34 | 68 |
| Gentoo penguin (Pygoscelis papua) | 58 | 61 | 119 |
| Total | 165 | 168 | 333 |

**Let's try to visualise the observed distribution of sex across species.**

# Visualising the palmer penguins data

We'll use the **ggplot2** package extensively this semester (Wickham, 2016).

Three key components:

- input a data frame
- mapping aesthetics `aes()` where you specify what goes on the axes, how to colour variables, what the groups are, etc.
- geometries `geom_****()` that you add to build up the plot

Finished product:

```
ggplot(data = penguins_clean) + aes(x = species, fill = sex) +
  geom_bar(position = "fill") +
  labs(x = "", y = "Proportion of penguins", fill = "Sex") +
  scale_y_continuous(labels = scales::percent_format()) +
  facet_grid(cols = vars(island), scales = "free_x", space = "free_x") +
  theme_linedraw(base_size = 22)
```

# Visualising the palmer penguins data

```
1 ggplot(data = penguins_clean) + aes(x = species, fill = sex)
```


- The `ggplot()` function knows about the data frame `penguins`.
- It knows what to map to the aesthetics: `species` is going to go on the x-axis and that the `fill` colour is going to be specified by the `sex` variable.

**It doesn't yet know what kind of plot to put on this blank canvas.**

# Aside: mutating a variable

The `species` variable is a bit long. For visualisation purposes we only really need the first word of the species variable. The code below adds to the **pipeline** we saw earlier by creating a new variable, `species_short`, that extracts the first word from the `species` variable.

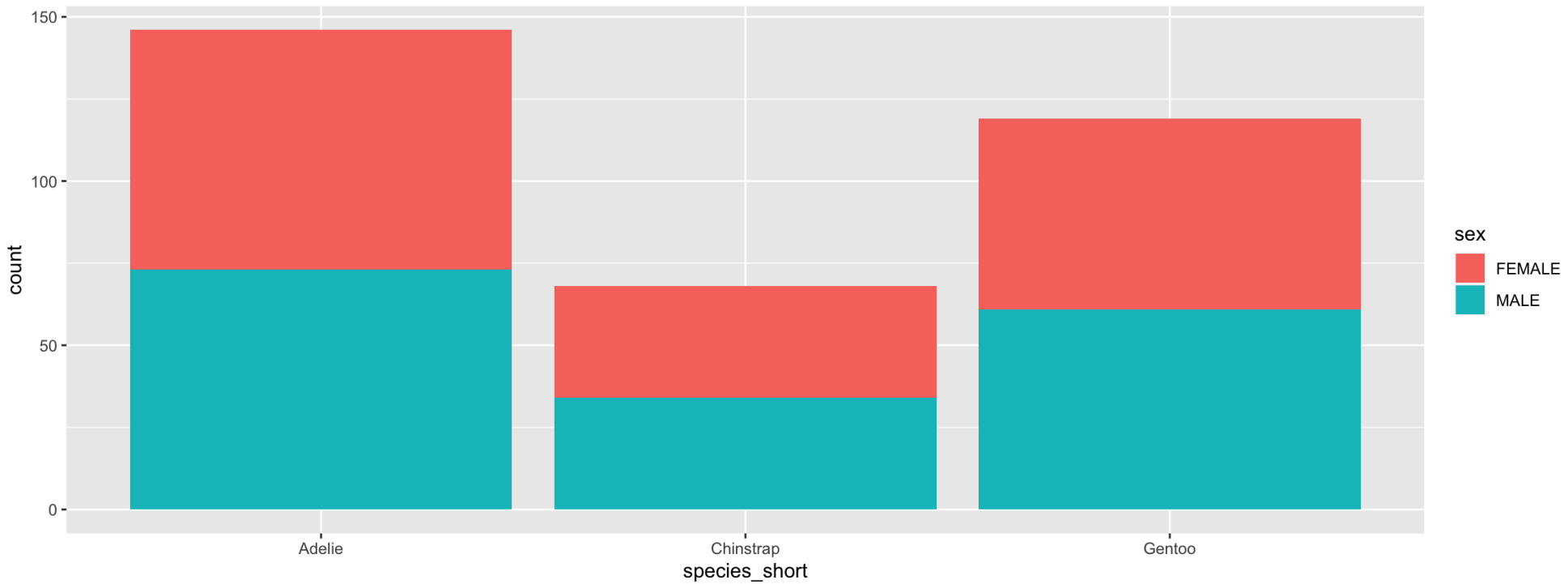
```
penguins_clean = penguins_raw |>
  janitor::clean_names() |>
  tidyr::drop_na(sex) |>
  dplyr::mutate(
    species_short = stringr::word(species, start = 1, end = 1)
  )
penguins_clean |>
  dplyr::select(species, species_short) |>
  dplyr::distinct()
```

```
# A tibble: 3 × 2
  species                                   species_short
  <chr>                                     <chr>
1 Adelie Penguin (Pygoscelis adeliae)       Adelie
2 Gentoo penguin (Pygoscelis papua)         Gentoo
3 Chinstrap penguin (Pygoscelis antarctica) Chinstrap
```

Learn more: “tibbles” vs “data frames” here.

# Visualising the palmer penguins data

```
1 ggplot(data = penguins_clean) + aes(x = species_short, fill = sex) +
2   geom_bar()
```

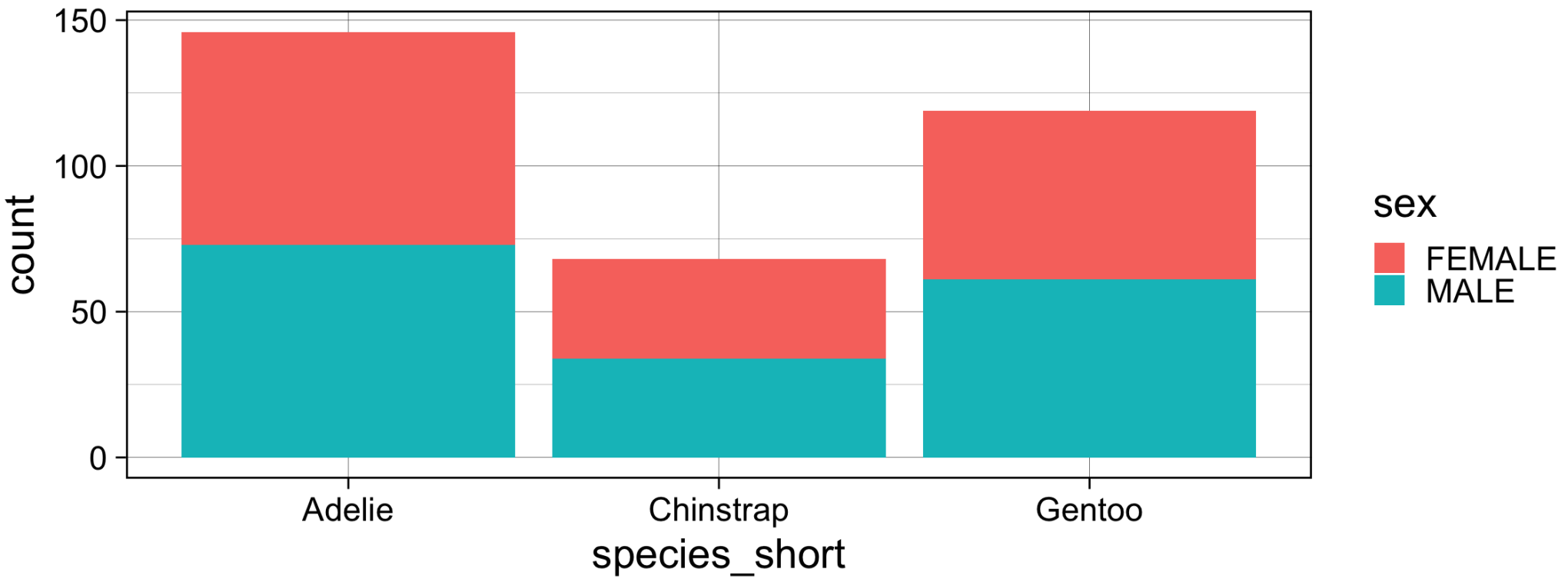



- We add (+) the bar chart geometry, `geom_bar()`, to our blank canvas.
- The bars represent counts in each species.
- The `fill` colours each bar (species) by sex.

# Visualising the palmer penguins data

Increase the font size and change the theme.

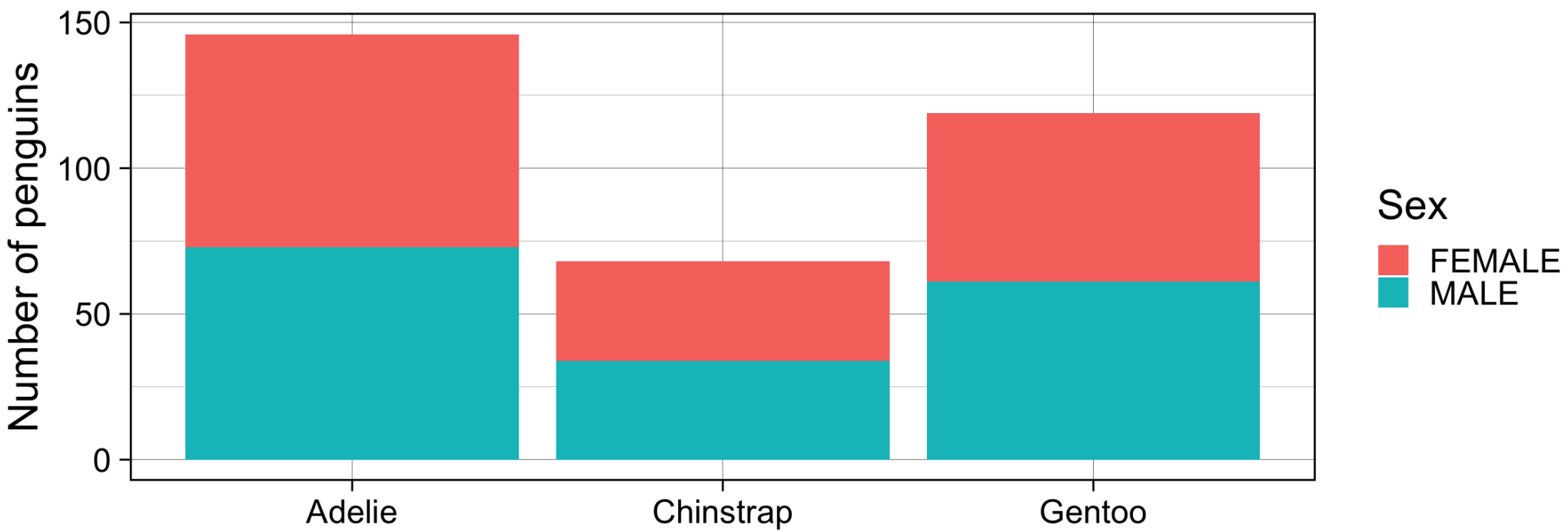
```
ggplot(data = penguins_clean) + aes(x = species_short, fill = sex) +
  geom_bar() +
  theme_linedraw(base_size = 22)
```



# Visualising the palmer penguins data

Tidy up the axis labels.

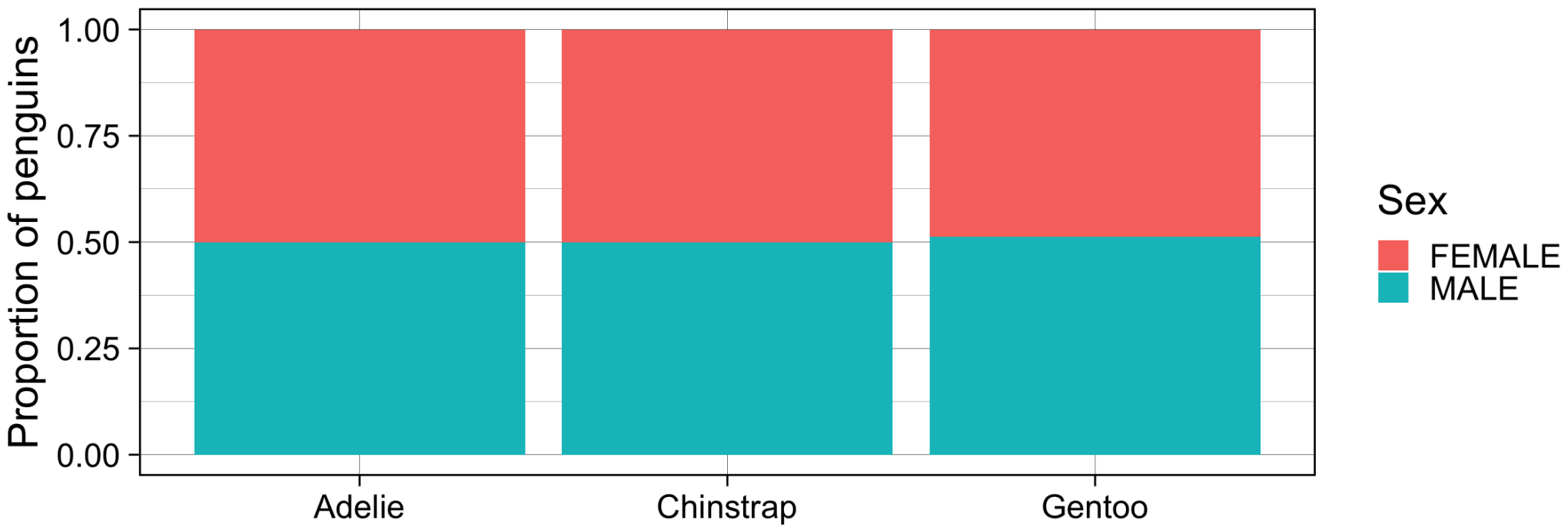
```
ggplot(data = penguins_clean) + aes(x = species_short, fill = sex) +
  geom_bar() +
  theme_linedraw(base_size = 22) +
  labs(x = "", y = "Number of penguins", fill = "Sex")
```



# Visualising the palmer penguins data

Make the bars represent **proportions** within each species:

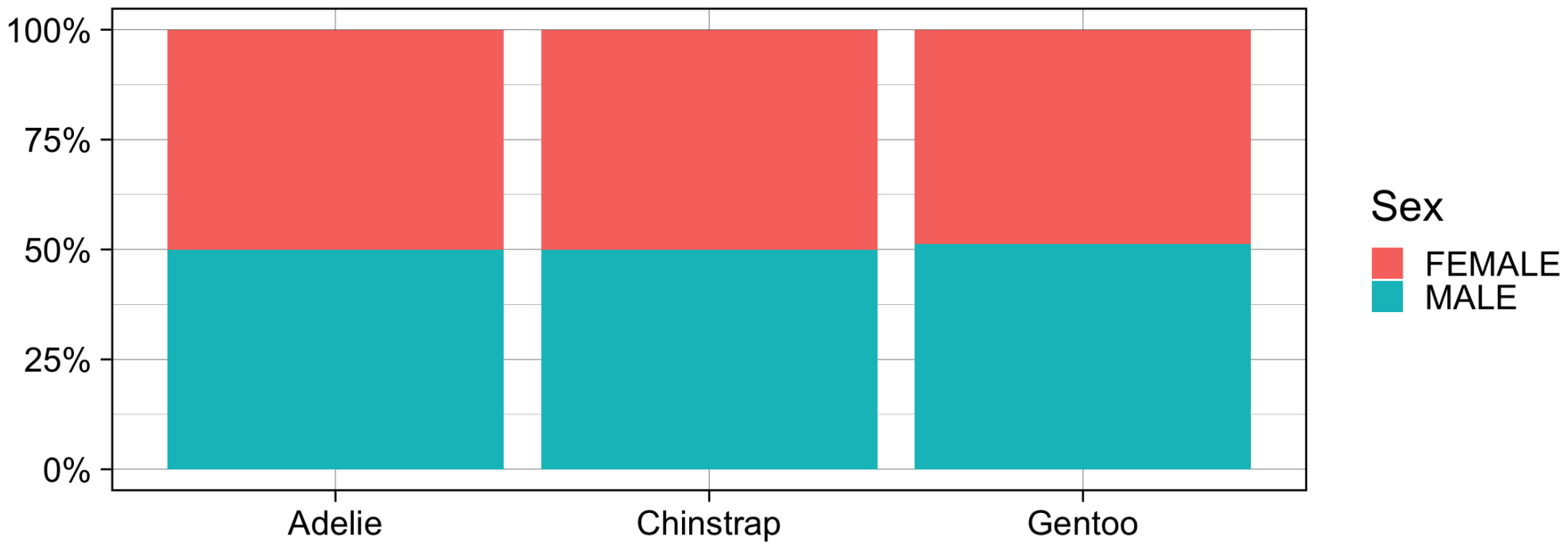
```
ggplot(data = penguins_clean) + aes(x = species_short, fill = sex) +
  geom_bar(position = "fill") +
  theme_linedraw(base_size = 22) +
  labs(x = "", y = "Proportion of penguins", fill = "Sex")
```



# Visualising the palmer penguins data

What if we want percent on the y-axis?

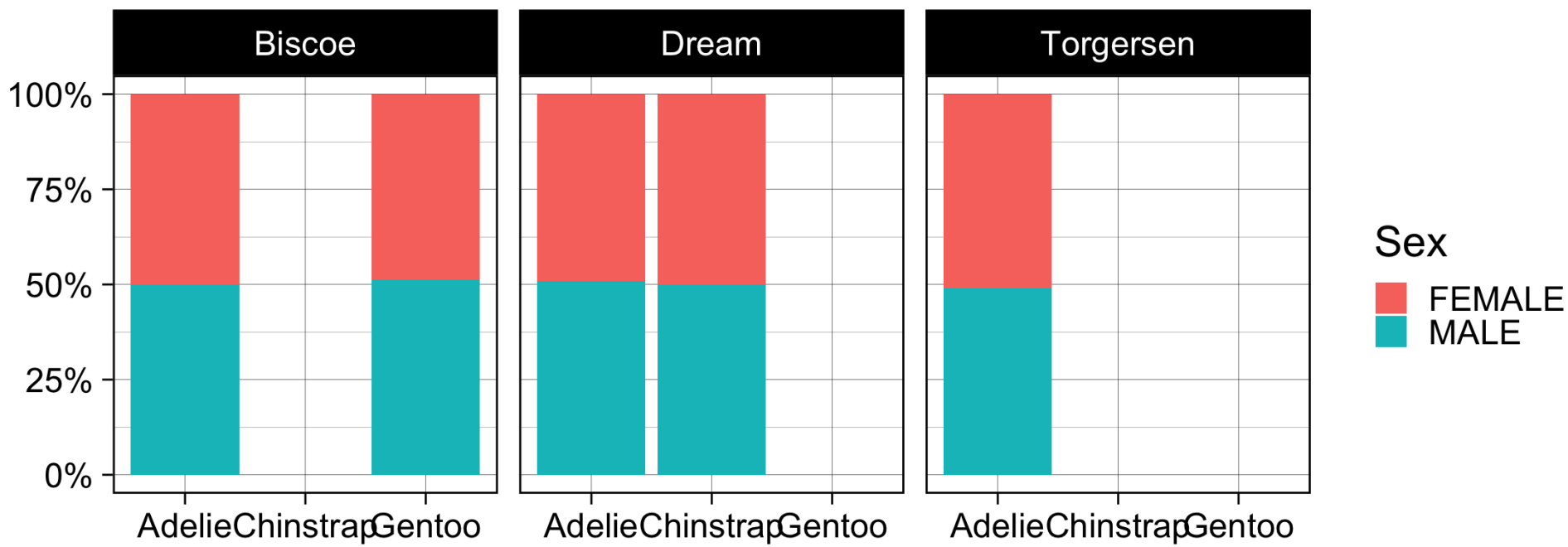
```
ggplot(data = penguins_clean) + aes(x = species_short, fill = sex) +
  geom_bar(position = "fill") +
  theme_linedraw(base_size = 22) +
  labs(x = "", y = "", fill = "Sex") +
  scale_y_continuous(labels = scales::percent_format())
```



# Visualising the palmer penguins data

Is it the same across all islands?

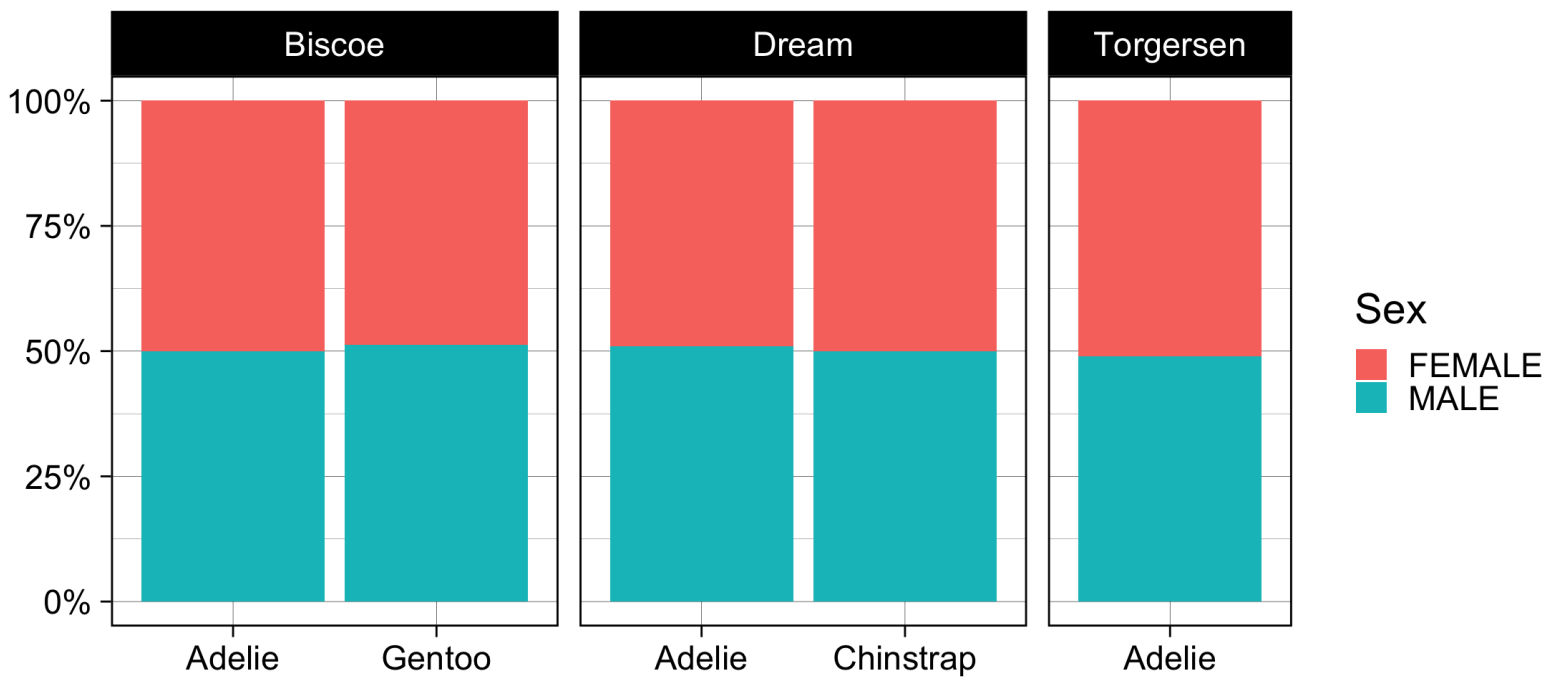
```
ggplot(data = penguins_clean) + aes(x = species_short, fill = sex) +
  geom_bar(position = "fill") +
  theme_linedraw(base_size = 22) +
  labs(x = "", y = "", fill = "Sex") +
  scale_y_continuous(labels = scales::percent_format()) +
  facet_grid(cols = vars(island))
```



# Visualising the palmer penguins data

Remove empty categories from the x axis.

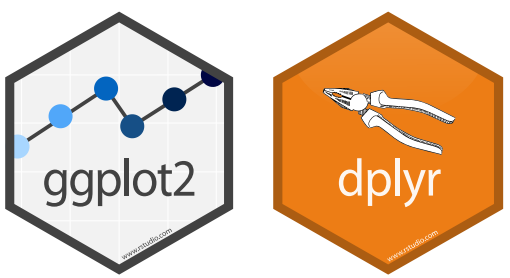
```
ggplot(data = penguins_clean) + aes(x = species_short, fill = sex) +
  geom_bar(position = "fill") +
  theme_linedraw(base_size = 22) +
  labs(x = "", y = "", fill = "Sex") +
  scale_y_continuous(labels = scales::percent_format()) +
  facet_grid(cols = vars(island), scales = "free_x", space = "free_x")
```



# R packages and functions

- `dplyr::glimpse()` or `str()` for inspecting the structure of objects
- `dplyr::mutate()` for creating new variables
- `visdat::vis_miss()` to check visually for missing data (`NA` values)
- `tidyr::drop_na()` to remove observations with missing values
- `janitor::clean_names()` and `janitor::tabyl()` to tidy column names and generate cross tabulations
- `ggplot2::ggplot()` and associated functions from the **ggplot2** package `aes()`, `geom_bar()`, `labs()`, `coord_flip()`, `facet_grid()` and `theme_linedraw()`

Over the semester we'll explore various **ggplot2** and **dplyr** features. You can learn more about this now in the data visualisation and data transformation chapters of R for Data Science (2nd Ed) by Wickham et al. (2023).



# References


Firke, S. (2021). *Janitor: Simple tools for examining and cleaning dirty data*. https://CRAN.R-project.org/package=janitor

Horst, A.M., Hill, A.P., & Gorman, K.B. (2020). *Palmerpenguins: Palmer archipelago (antarctica) penguin data*. https://allisonhorst.github.io/palmerpenguins/

Tierney, N. (2017). visdat: Visualising whole data frames. *JOSS*, 2(16), 355. DOI: 10.21105/joss.00355

Wickham, H. (2016). *ggplot2: Elegant graphics for data analysis*. New York, NY: Springer-Verlag. http://ggplot2.tidyverse.org/

Wickham, H., Averick, M., Bryan, J., Chang, W., McGowan, L.D., François, R., … Yutani, H. (2019). Welcome to the tidyverse. *Journal of Open Source Software*, 4(43), 1686. DOI: 10.21105/joss.01686

Wickham, H., Çetinkaya-Rundel, M., & Garrett, G. (2023). *R for data science* (2nd ed.). O'Reilly Media. https://r4ds.hadley.nz/