DATA2002

Data visualisation with ggplot

Garth Tarr

THE UNIVERSITY OF SYDNEY

Palmer penguins data set

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, Hill, and Gorman, 2020).

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

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

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

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

Taking a quick look at the data

```
library(dplyr)
dplyr::glimpse(penguins) # glimpse the structure of the penguins data frame
```

```
## Rows: 344
## Columns: 8
## $ species           <fct> Adelie, Adelie, Adelie, Adelie, …
## $ island            <fct> Torgersen, Torgersen, Torgersen,…
## $ bill_length_mm    <dbl> 39.1, 39.5, 40.3, NA, 36.7, 39.3…
## $ bill_depth_mm     <dbl> 18.7, 17.4, 18.0, NA, 19.3, 20.6…
## $ flipper_length_mm <int> 181, 186, 195, NA, 193, 190, 181…
## $ body_mass_g       <int> 3750, 3800, 3250, NA, 3450, 3650…
## $ sex               <fct> male, female, female, NA, female…
## $ year              <int> 2007, 2007, 2007, 2007, 2007, 20…
```

Could also use `str(penguins)`, short for `str`ucture, though in this case it prints out extra unnecessary information.

Palmer penguins data set

```
rmarkdown::paged_table(penguins)
```

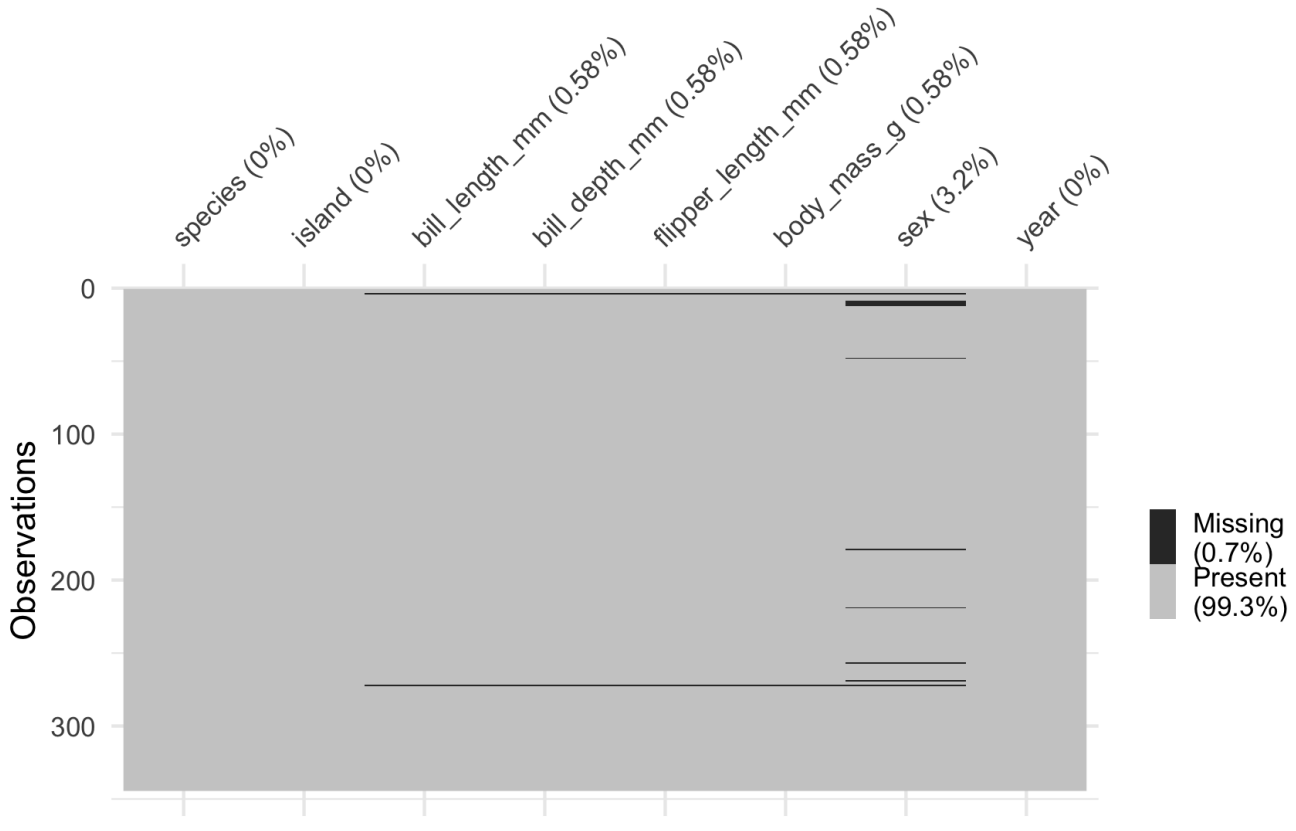
species <fct>	island <fct>	bill_length_mm <dbl>	bill_depth_mm <dbl>	flipper_length_mm <int>
Adelie	Torgersen	39.1	18.7	181
Adelie	Torgersen	39.5	17.4	186
Adelie	Torgersen	40.3	18.0	195
Adelie	Torgersen	*NA*	*NA*	*NA*
Adelie	Torgersen	36.7	19.3	193
Adelie	Torgersen	39.3	20.6	190
Adelie	Torgersen	38.9	17.8	181
Adelie	Torgersen	39.2	19.6	195
Adelie	Torgersen	34.1	18.1	193
Adelie	Torgersen	42.0	20.2	190

1-10 of 344 rows | 1-5 of 8 columns

Previous **1** 2 3 4 5 6 … 35 Next

Missing data?

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



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

```
penguins = penguins %>%
  tidyr::drop_na()
```

Palmer penguins breakdown

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

```
##    species female male Total
##     Adelie     73   73   146
##  Chinstrap     34   34    68
##     Gentoo     58   61   119
##      Total    165  168   333
```

Let's try to visualise this.

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

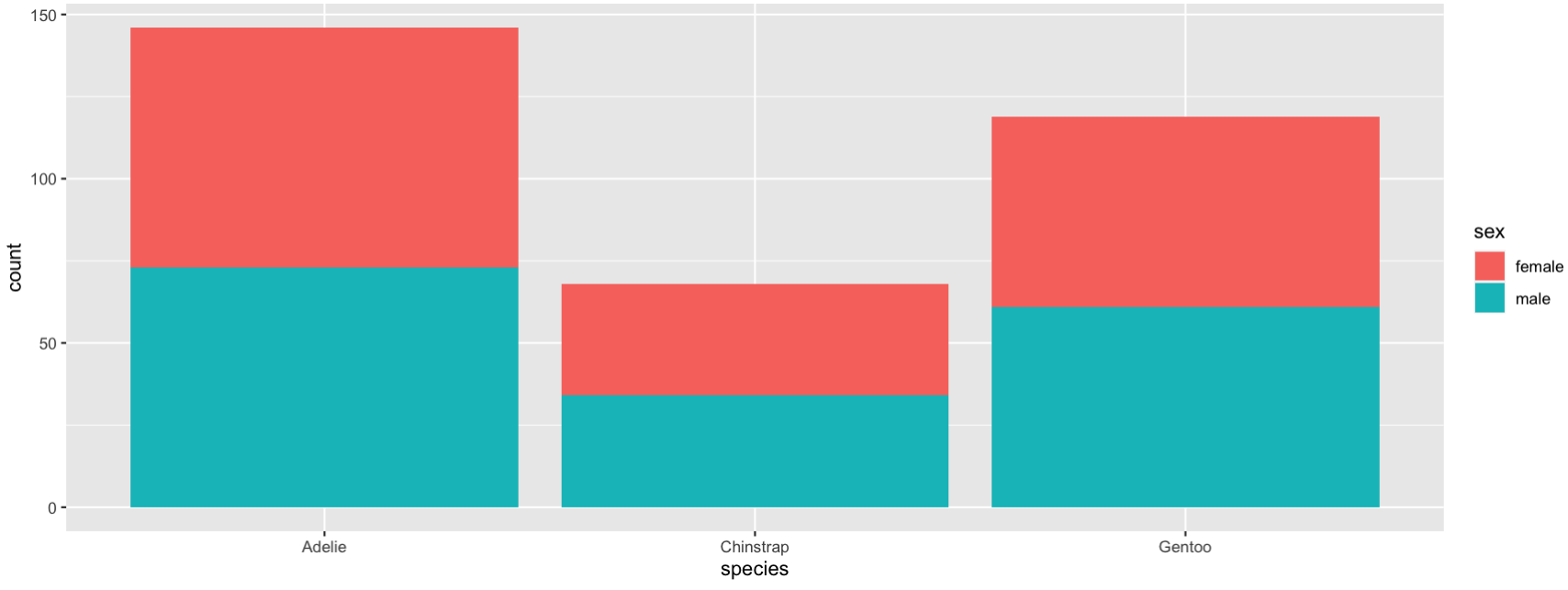
Finished product:

```
ggplot(data = penguins) + 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)
```

```r
library(ggplot2)
ggplot(data = penguins) + 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.

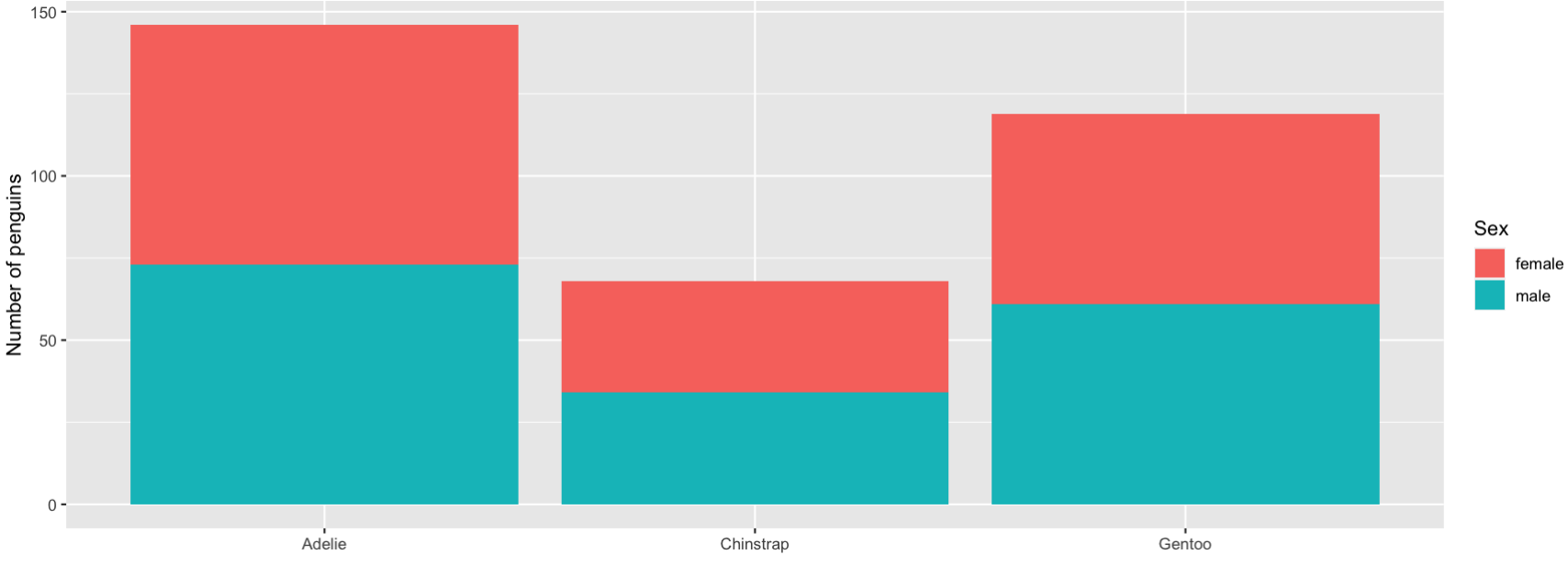
```
ggplot(data = penguins) + aes(x = species, fill = sex) +
  geom_bar()
```



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

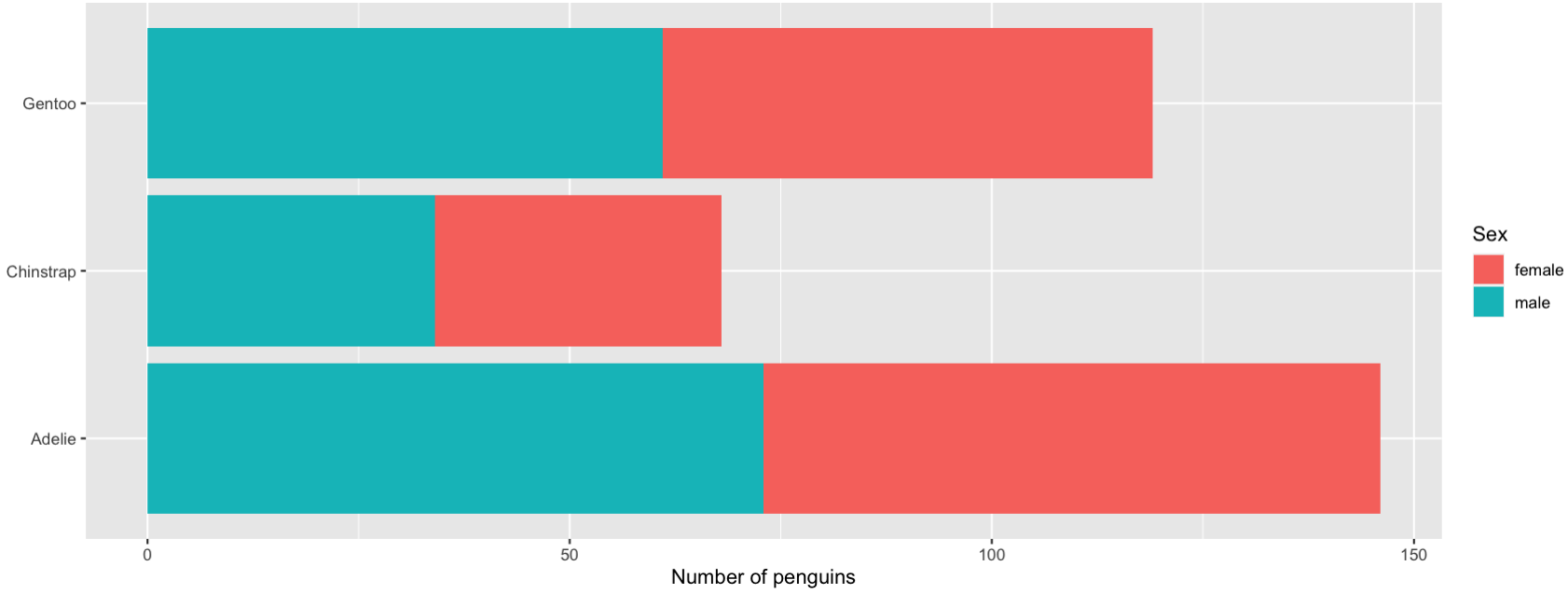
You will almost always need to tidy up the axis labels. This can be done with the `labs()` function.

```
ggplot(data = penguins) + aes(x = species, fill = sex) +
  geom_bar() +
  labs(x = "", y = "Number of penguins", fill = "Sex")
```



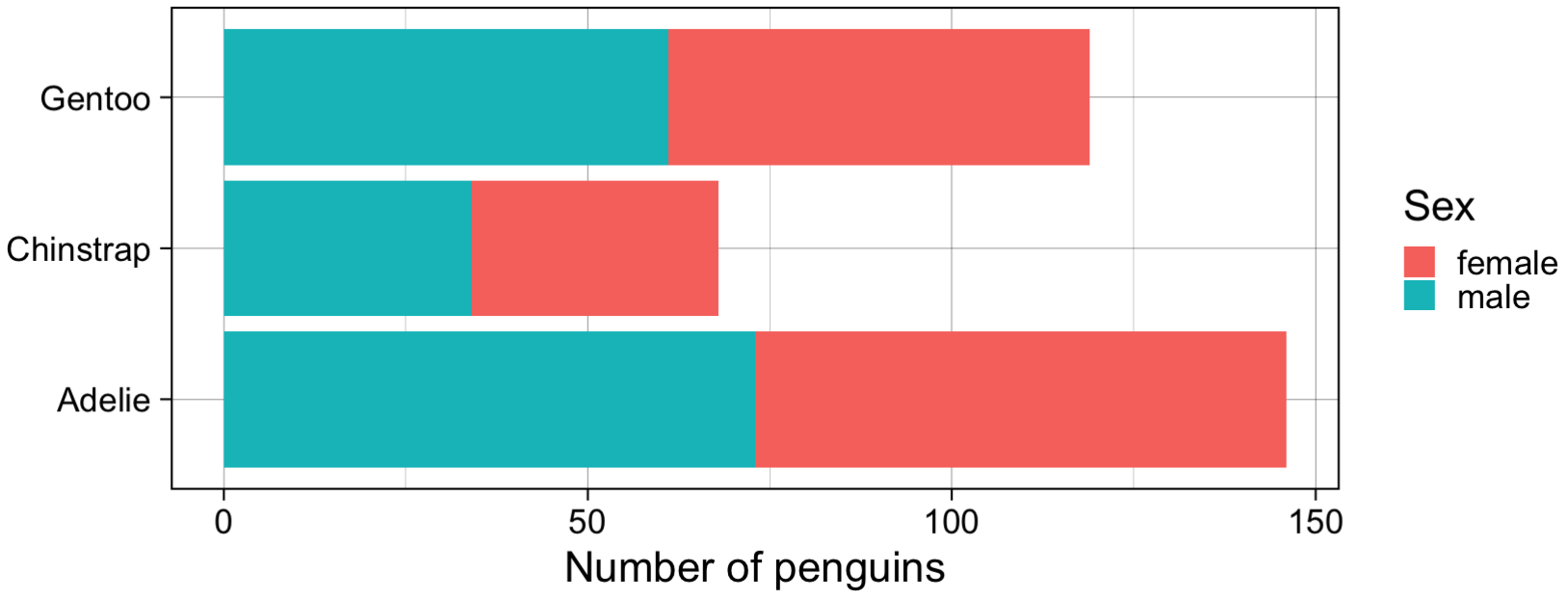
If you want to flip the axes around, you can use `coord_flip()`.

```
ggplot(data = penguins) + aes(x = species, fill = sex) +
  geom_bar() +
  labs(x = "", y = "Number of penguins", fill = "Sex") +
  coord_flip()
```



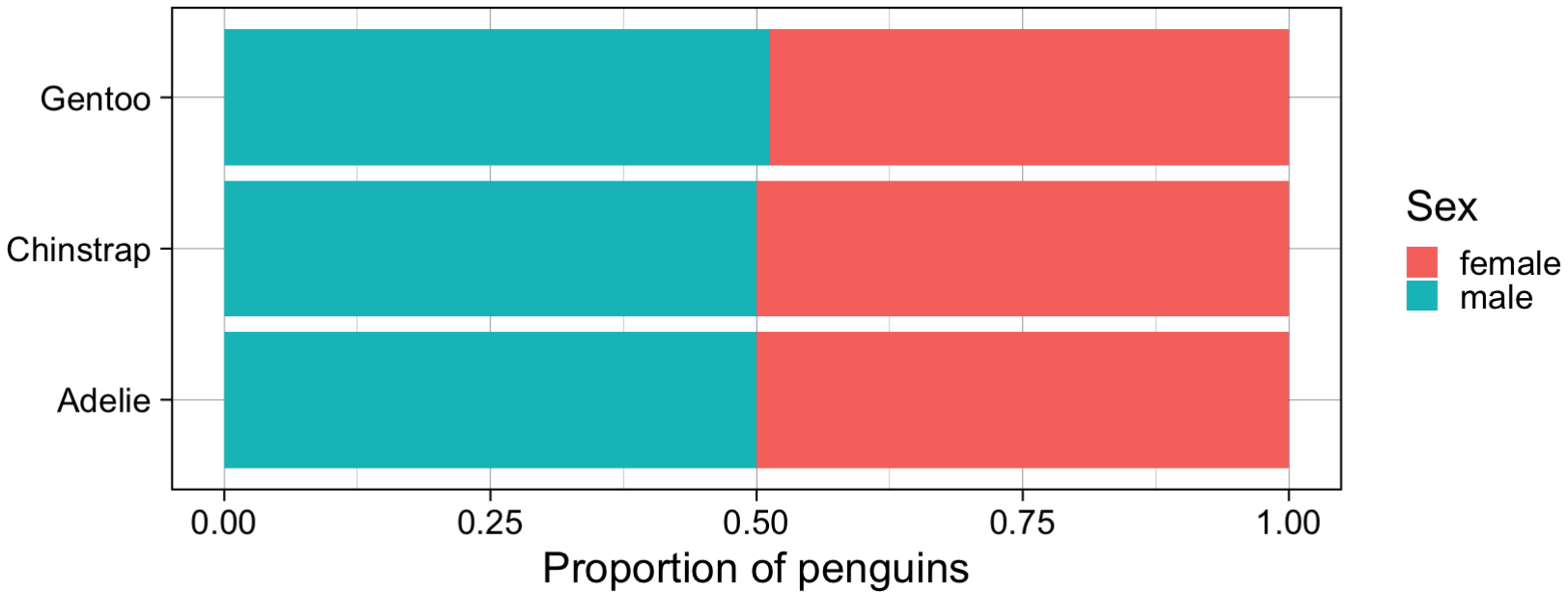
Increase the font size and change the theme.

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



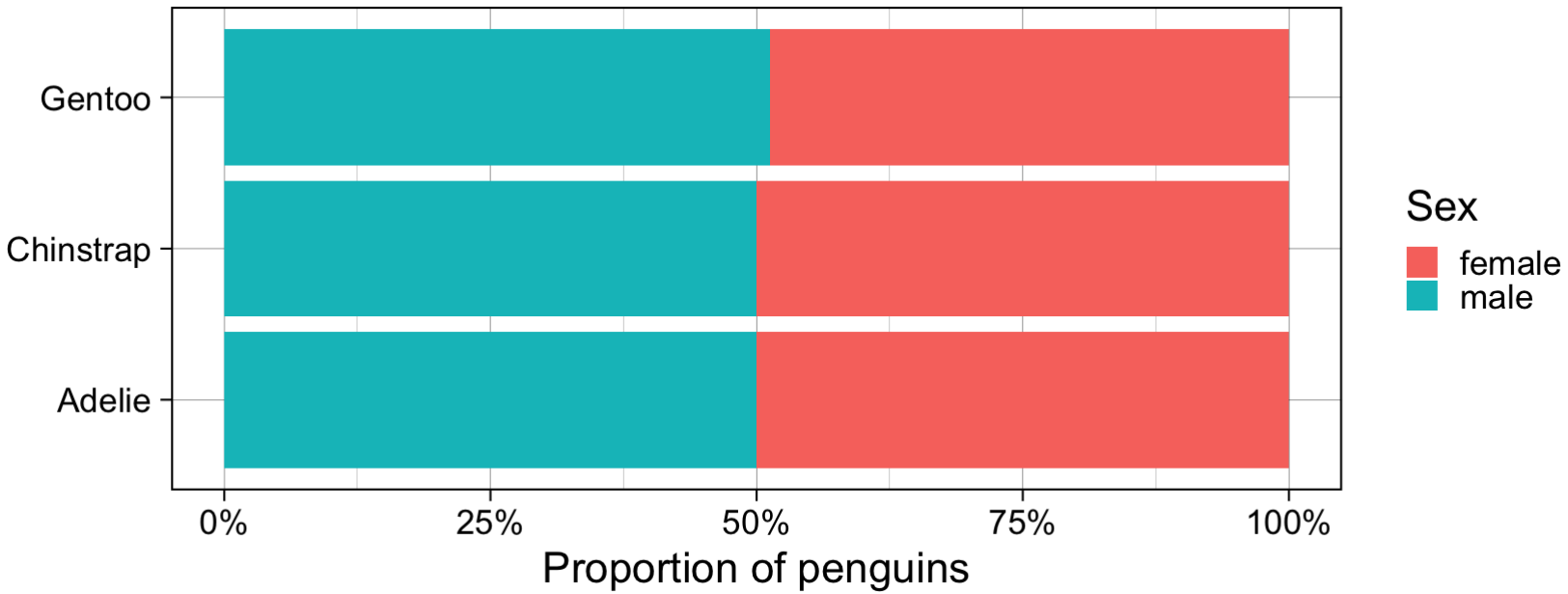
Make the bars represent proportions within each species.

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



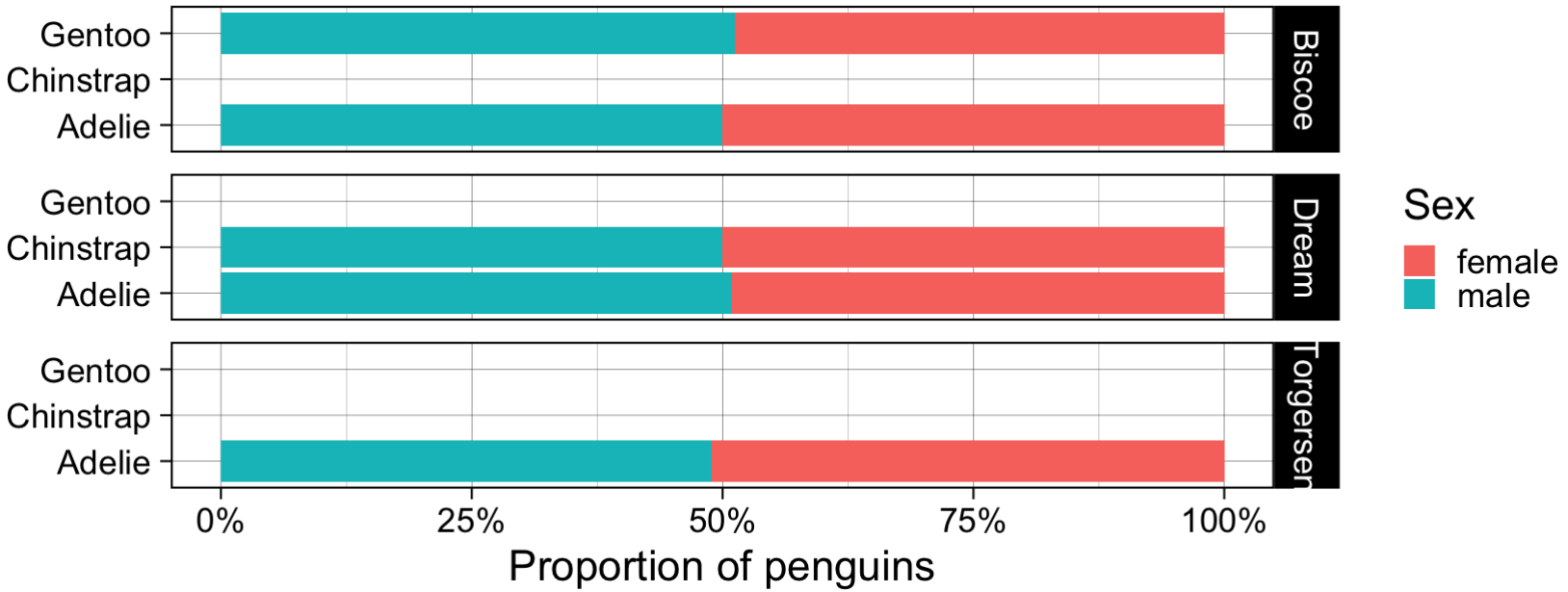
What if we want percents on the x-axis?

```
ggplot(data = penguins) + aes(x = species, fill = sex) +
  geom_bar(position = "fill") +
  labs(x = "", y = "Proportion of penguins", fill = "Sex") +
  scale_y_continuous(labels = scales::percent_format()) +
  coord_flip() +
  theme_linedraw(base_size = 22)
```

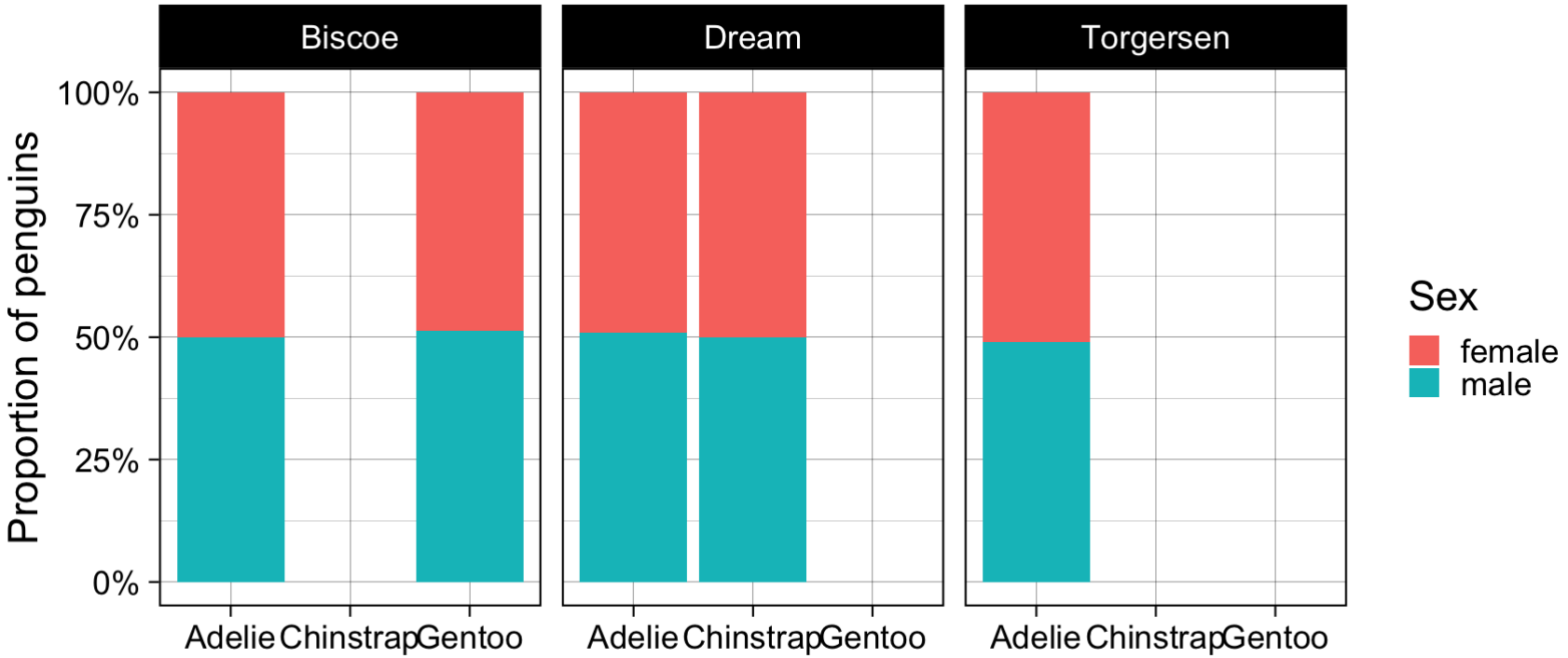


Is it the same across all islands?

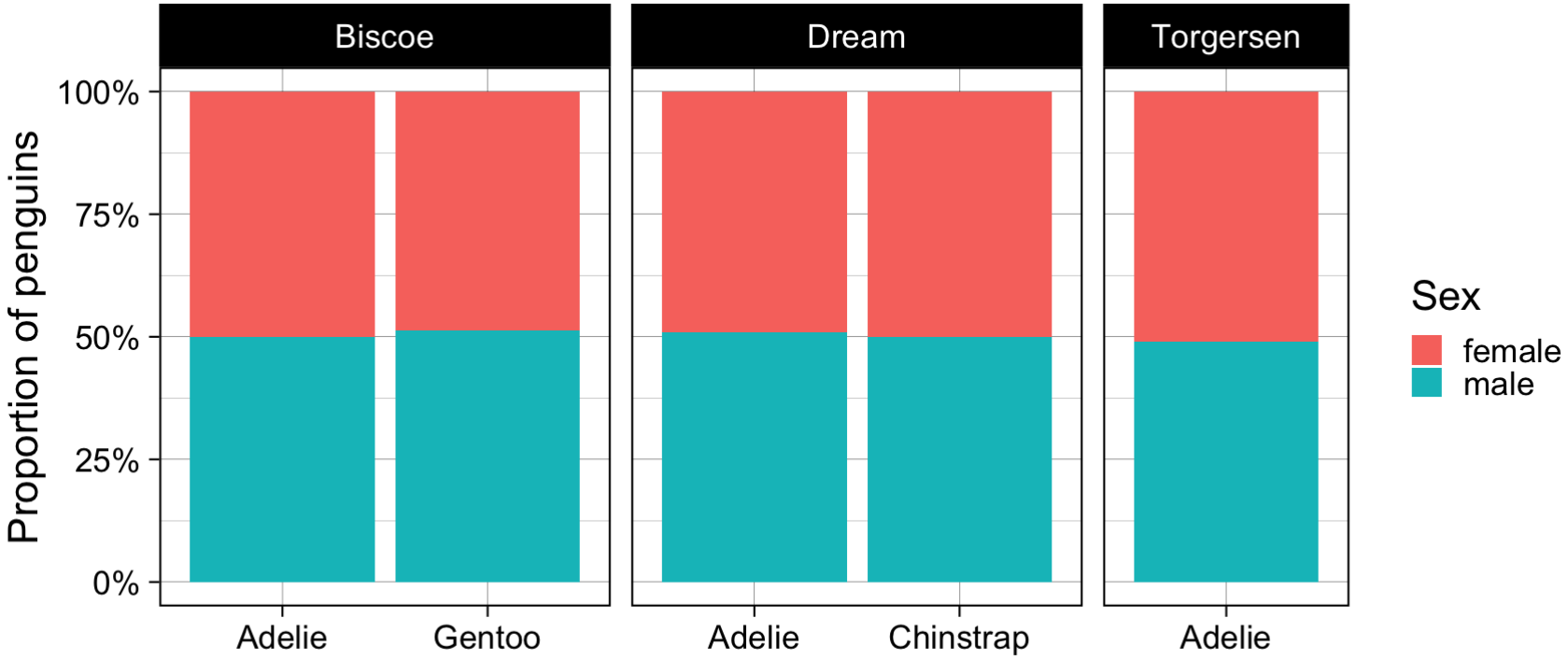
```
ggplot(data = penguins) + aes(x = species, fill = sex) +
  geom_bar(position = "fill") +
  labs(x = "", y = "Proportion of penguins", fill = "Sex") +
  scale_y_continuous(labels = scales::percent_format()) +
  coord_flip() +
  facet_grid(rows = vars(island)) +
  theme_linedraw(base_size = 22)
```



```
ggplot(data = penguins) + 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)) +
  theme_linedraw(base_size = 22)
```



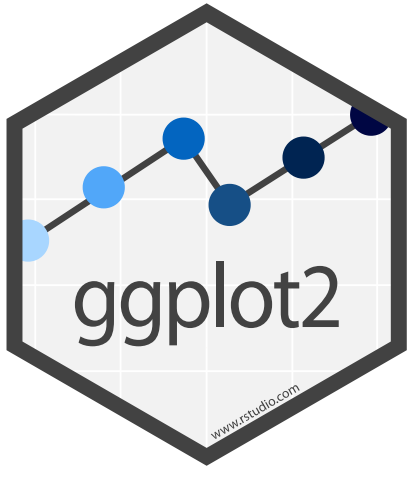
```
ggplot(data = penguins) + aes(x = species, fill = sex) +
  geom_bar(position = "fill") +
  labs(x = "", y = "Proportion of penguins", fill = "Sex") +
  scale_y_continuous(labels = scales::percent_format()) +
  facet_grid(cols = vars(island), scales = "free_x", space = "free_x") +
  theme_linedraw(base_size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



R packages and functions

- `readr::read_csv()` for reading in csv files
- `dplyr::glimpse()` or `str()` for inspecting the structure of objects
- `visdat::vis_miss()` to check visually for missing data
- `tidyr::drop_na()` to remove any observations with missingness
- `janitor::tabyl()` to produce a cross-tabulation
- `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** features. You can find out more about it here: http://ggplot2.tidyverse.org



References

Firke, S. (2021). *janitor: Simple Tools for Examining and Cleaning Dirty Data*. R package version 2.1.0. URL: https://CRAN.R-project.org/package=janitor.

Horst, A. M., A. P. Hill, and K. B. Gorman (2020). *palmerpenguins: Palmer Archipelago (Antarctica) penguin data*. R package version 0.1.0. URL: https://allisonhorst.github.io/palmerpenguins/.

Tierney, N. (2017). "visdat: Visualising Whole Data Frames". In: *JOSS* 2.16, p. 355. DOI: 10.21105/joss.00355. URL: http://dx.doi.org/10.21105/joss.00355.

Wickham, H. (2016). *ggplot2: Elegant Graphics for Data Analysis*. New York, NY: Springer-Verlag. ISBN: 978-3-319-24277-4. URL: http://ggplot2.tidyverse.org/.

Wickham, H., M. Averick, J. Bryan, et al. (2019). "Welcome to the tidyverse". In: *Journal of Open Source Software* 4.43, p. 1686. DOI: 10.21105/joss.01686.

Wickham, H., R. François, L. Henry, et al. (2018). *dplyr: A Grammar of Data Manipulation*. R package version 0.7.5. URL: https://CRAN.R-project.org/package=dplyr.