Working with Data In Tidyverse

Background

A lot of your time working in R will likely be spent organizing, cleaning, and subsetting data to prepare it for plotting and statistical analysis. The tidyverse (https://www.tidyverse.org/) is a set of R packages that make this easier by establishing a shared set of standards for how to represent and work with data sets. It includes commonly used packages for managing data like dplyr (https://dplyr.tidyverse.org/) and tidyr (https://tidyr.tidyverse.org/), and the ggplot2 (https://ggplot2.tidyverse.org/) graphics package. This module focuses on data management with dplyr and tidyr. For more an introduction to ggplot2, check out this module (GettingStartedggplot2.md).

Objectives

The goal of this module is to introduce R users to the philosophy of tidyverse and illustrate how its features can make data management and analysis easier. More specifically, this module guides users through

1. Installing tidyverse
2. Subsetting data and chaining operations using pipes (%>%)
3. Calculating new variables from existing ones
4. Summarizing data using split-apply-combine approach
5. Applying statistical analyses “in the pipeline”
6. Plotting “in the pipeline”
7. Moving between long- and wide-format data

Installing Tidyverse

First thing’s first – let’s install tidyverse!

```
install.packages("tidyverse") # install the package
```

Note that installing a package just gets it onto our computer– not into R! To access a package in R, we have to load the package using the `library()` function.

```
library(tidyverse) # load it in the current R session.
```

```
## ── Attaching packages ─────────────────────────────────────── tidyverse 1.3.2 ──
## ✔ ggplot2 3.3.5      ✔ purrr   0.3.4
## ✔ tibble  3.1.8      ✔ dplyr   1.0.10
## ✔ tidyr   1.1.4      ✔ stringr 1.4.0
## ✔ readr   2.1.3      ✔ forcats 0.5.2
## ── Conflicts ────────────────────────────────────────── tidyverse_conflicts() ──
## ✖ dplyr::filter() masks stats::filter()
## ✖ dplyr::lag()    masks stats::lag()
```

You can see that loading tidyverse actually loads a set of other packages – *ggplot2*, *purrr*, *tibble*, etc. These are the workhorses of the tidyverse. You can also see a list of “conflicts”. These are cases where a package we just loaded (e.g., *dplyr*) has its own version of a base R function (e.g., filter, lag), so it’s good to be mindful of this if we use those functions.

Subsetting data using *dplyr*

The package *dplyr* within the *tidyverse* family has a lot of helpful functions for getting your data ready to visualize and analyze. I’ll focus on the most commonly used ones in this module, but there’s a great cheatsheet available here (https://github.com/rstudio/cheatsheets/blob/master/tidyr.pdf).

Commonly used functions

- select(): select columns
- pull(): select a column and turn to a vector
- filter(): filter rows matching some criteria
- mutate(): create new columns by applying functions to existing columns
- group_by(): split data into groups based on one or more variables
- summarize(): calculate summary statistics for a variable
- arrange(): sort rows by some criteria
- count(): count discrete values
- left_join(), right_join(), inner_join(), full_join(): merge data tables in various ways.

A little help from some penguins

Let’s try our hand at some “data wrangling” using some published data on Antarctic penguins. These data were collected by Kristen Gorman with the Palmer Station Long Term Ecological Research Program and later developed into an R package for educational uses by Allison Horst, and Alison Hill, and Kristen Gorman. The published manuscript focused on differences in sexual size dimorphism among 3 species of penguin and their relation to sex differences in foraging ecology. The full data set has really neat info on stable isotope blood measurements and reproductive success, but for the sake of brevity we will focus on the morphological measures used to quantify sexual size dimorphism.

First let’s install the *palmerpenguins* package.

```
install.packages("palmerpenguins") # install the package
```

And now let's load it.

```
library(palmerpenguins) # load it in current R session
```

The data we'll be using are now available in an object called `penguins` . The full data are in `penguins_raw` , which I encourage you to explore on your own later.

Let's take a look at our data set using the `print()` function in base R.

```
print(penguins)
```

```
## # A tibble: 344 × 8
##    species island    bill_length_mm bill_depth_mm flipper_…¹ body_…² sex    year
##    <fct>   <fct>              <dbl>         <dbl>      <int>   <int> <fct> <int>
##  1 Adelie  Torgersen           39.1          18.7        181    3750 male   2007
##  2 Adelie  Torgersen           39.5          17.4        186    3800 fema…  2007
##  3 Adelie  Torgersen           40.3          18          195    3250 fema…  2007
##  4 Adelie  Torgersen          NA            NA            NA      NA <NA>   2007
##  5 Adelie  Torgersen           36.7          19.3        193    3450 fema…  2007
##  6 Adelie  Torgersen           39.3          20.6        190    3650 male   2007
##  7 Adelie  Torgersen           38.9          17.8        181    3625 fema…  2007
##  8 Adelie  Torgersen           39.2          19.6        195    4675 male   2007
##  9 Adelie  Torgersen           34.1          18.1        193    3475 <NA>   2007
## 10 Adelie  Torgersen           42            20.2        190    4250 <NA>   2007
## # … with 334 more rows, and abbreviated variable names ¹flipper_length_mm,
## #   ²body_mass_g
```

Interesting! It tells us that the penguins dataset is saved in R as a `tibble` . *Tibbles are one of the main features of the tidyverse approach to data wrangling.* A `tibble` is like a `data.frame` from base R, but with a few important differences. For one, viewing and printing tibbles is generally tidier (get it?) than `data.frame` s. Notice how viewing our tibble didn't drown the console with the entire dataset at once?? Instead we get a nice summary. It also tells us what type of information is stored in each column – factors, integers, machine precision numbers ("dbl"), and so on. The authors of *palmerpenguins* like the tidyverse approach to data, so the data come prepackaged as a tibble.

A `data.frame` is not quite as friendly. However, you may sometimes need to turn a tibble to a data.frame to take advantage of functions in base R or other packages. Let's transform our penguin tibble into a data.frame and take a look at it.

```
penguins_df = as.data.frame(penguins) # creata a data.frame by applying the as.dat
a.frame() function to the penguin tibble
print(penguins_df)
```

```
##       species    island bill_length_mm bill_depth_mm flipper_length_mm
## 1      Adelie Torgersen           39.1          18.7               181
## 2      Adelie Torgersen           39.5          17.4               186
## 3      Adelie Torgersen           40.3          18.0               195
## 4      Adelie Torgersen             NA            NA                NA
## 5      Adelie Torgersen           36.7          19.3               193
## 6      Adelie Torgersen           39.3          20.6               190
## 7      Adelie Torgersen           38.9          17.8               181
## 8      Adelie Torgersen           39.2          19.6               195
## 9      Adelie Torgersen           34.1          18.1               193
## 10     Adelie Torgersen           42.0          20.2               190
## 11     Adelie Torgersen           37.8          17.1               186
## 12     Adelie Torgersen           37.8          17.3               180
## 13     Adelie Torgersen           41.1          17.6               182
## 14     Adelie Torgersen           38.6          21.2               191
## 15     Adelie Torgersen           34.6          21.1               198
## 16     Adelie Torgersen           36.6          17.8               185
## 17     Adelie Torgersen           38.7          19.0               195
## 18     Adelie Torgersen           42.5          20.7               197
## 19     Adelie Torgersen           34.4          18.4               184
## 20     Adelie Torgersen           46.0          21.5               194
## 21     Adelie    Biscoe           37.8          18.3               174
## 22     Adelie    Biscoe           37.7          18.7               180
## 23     Adelie    Biscoe           35.9          19.2               189
## 24     Adelie    Biscoe           38.2          18.1               185
## 25     Adelie    Biscoe           38.8          17.2               180
## 26     Adelie    Biscoe           35.3          18.9               187
## 27     Adelie    Biscoe           40.6          18.6               183
## 28     Adelie    Biscoe           40.5          17.9               187
## 29     Adelie    Biscoe           37.9          18.6               172
## 30     Adelie    Biscoe           40.5          18.9               180
## 31     Adelie     Dream           39.5          16.7               178
## 32     Adelie     Dream           37.2          18.1               178
## 33     Adelie     Dream           39.5          17.8               188
## 34     Adelie     Dream           40.9          18.9               184
## 35     Adelie     Dream           36.4          17.0               195
## 36     Adelie     Dream           39.2          21.1               196
## 37     Adelie     Dream           38.8          20.0               190
## 38     Adelie     Dream           42.2          18.5               180
## 39     Adelie     Dream           37.6          19.3               181
## 40     Adelie     Dream           39.8          19.1               184
## 41     Adelie     Dream           36.5          18.0               182
## 42     Adelie     Dream           40.8          18.4               195
## 43     Adelie     Dream           36.0          18.5               186
## 44     Adelie     Dream           44.1          19.7               196
## 45     Adelie     Dream           37.0          16.9               185
## 46     Adelie     Dream           39.6          18.8               190
## 47     Adelie     Dream           41.1          19.0               182
## 48     Adelie     Dream           37.5          18.9               179
## 49     Adelie     Dream           36.0          17.9               190
## 50     Adelie     Dream           42.3          21.2               191
## 51     Adelie    Biscoe           39.6          17.7               186
```

```
## 52     Adelie    Biscoe           40.1          18.9              188
## 53     Adelie    Biscoe           35.0          17.9              190
## 54     Adelie    Biscoe           42.0          19.5              200
## 55     Adelie    Biscoe           34.5          18.1              187
## 56     Adelie    Biscoe           41.4          18.6              191
## 57     Adelie    Biscoe           39.0          17.5              186
## 58     Adelie    Biscoe           40.6          18.8              193
## 59     Adelie    Biscoe           36.5          16.6              181
## 60     Adelie    Biscoe           37.6          19.1              194
## 61     Adelie    Biscoe           35.7          16.9              185
## 62     Adelie    Biscoe           41.3          21.1              195
## 63     Adelie    Biscoe           37.6          17.0              185
## 64     Adelie    Biscoe           41.1          18.2              192
## 65     Adelie    Biscoe           36.4          17.1              184
## 66     Adelie    Biscoe           41.6          18.0              192
## 67     Adelie    Biscoe           35.5          16.2              195
## 68     Adelie    Biscoe           41.1          19.1              188
## 69     Adelie Torgersen           35.9          16.6              190
## 70     Adelie Torgersen           41.8          19.4              198
## 71     Adelie Torgersen           33.5          19.0              190
## 72     Adelie Torgersen           39.7          18.4              190
## 73     Adelie Torgersen           39.6          17.2              196
## 74     Adelie Torgersen           45.8          18.9              197
## 75     Adelie Torgersen           35.5          17.5              190
## 76     Adelie Torgersen           42.8          18.5              195
## 77     Adelie Torgersen           40.9          16.8              191
## 78     Adelie Torgersen           37.2          19.4              184
## 79     Adelie Torgersen           36.2          16.1              187
## 80     Adelie Torgersen           42.1          19.1              195
## 81     Adelie Torgersen           34.6          17.2              189
## 82     Adelie Torgersen           42.9          17.6              196
## 83     Adelie Torgersen           36.7          18.8              187
## 84     Adelie Torgersen           35.1          19.4              193
## 85     Adelie     Dream           37.3          17.8              191
## 86     Adelie     Dream           41.3          20.3              194
## 87     Adelie     Dream           36.3          19.5              190
## 88     Adelie     Dream           36.9          18.6              189
## 89     Adelie     Dream           38.3          19.2              189
## 90     Adelie     Dream           38.9          18.8              190
## 91     Adelie     Dream           35.7          18.0              202
## 92     Adelie     Dream           41.1          18.1              205
## 93     Adelie     Dream           34.0          17.1              185
## 94     Adelie     Dream           39.6          18.1              186
## 95     Adelie     Dream           36.2          17.3              187
## 96     Adelie     Dream           40.8          18.9              208
## 97     Adelie     Dream           38.1          18.6              190
## 98     Adelie     Dream           40.3          18.5              196
## 99     Adelie     Dream           33.1          16.1              178
## 100    Adelie     Dream           43.2          18.5              192
## 101    Adelie    Biscoe           35.0          17.9              192
## 102    Adelie    Biscoe           41.0          20.0              203
## 103    Adelie    Biscoe           37.7          16.0              183
```

```
## 104    Adelie    Biscoe           37.8          20.0              190
## 105    Adelie    Biscoe           37.9          18.6              193
## 106    Adelie    Biscoe           39.7          18.9              184
## 107    Adelie    Biscoe           38.6          17.2              199
## 108    Adelie    Biscoe           38.2          20.0              190
## 109    Adelie    Biscoe           38.1          17.0              181
## 110    Adelie    Biscoe           43.2          19.0              197
## 111    Adelie    Biscoe           38.1          16.5              198
## 112    Adelie    Biscoe           45.6          20.3              191
## 113    Adelie    Biscoe           39.7          17.7              193
## 114    Adelie    Biscoe           42.2          19.5              197
## 115    Adelie    Biscoe           39.6          20.7              191
## 116    Adelie    Biscoe           42.7          18.3              196
## 117    Adelie Torgersen           38.6          17.0              188
## 118    Adelie Torgersen           37.3          20.5              199
## 119    Adelie Torgersen           35.7          17.0              189
## 120    Adelie Torgersen           41.1          18.6              189
## 121    Adelie Torgersen           36.2          17.2              187
## 122    Adelie Torgersen           37.7          19.8              198
## 123    Adelie Torgersen           40.2          17.0              176
## 124    Adelie Torgersen           41.4          18.5              202
## 125    Adelie Torgersen           35.2          15.9              186
## 126    Adelie Torgersen           40.6          19.0              199
## 127    Adelie Torgersen           38.8          17.6              191
## 128    Adelie Torgersen           41.5          18.3              195
## 129    Adelie Torgersen           39.0          17.1              191
## 130    Adelie Torgersen           44.1          18.0              210
## 131    Adelie Torgersen           38.5          17.9              190
## 132    Adelie Torgersen           43.1          19.2              197
## 133    Adelie     Dream           36.8          18.5              193
## 134    Adelie     Dream           37.5          18.5              199
## 135    Adelie     Dream           38.1          17.6              187
## 136    Adelie     Dream           41.1          17.5              190
## 137    Adelie     Dream           35.6          17.5              191
## 138    Adelie     Dream           40.2          20.1              200
## 139    Adelie     Dream           37.0          16.5              185
## 140    Adelie     Dream           39.7          17.9              193
## 141    Adelie     Dream           40.2          17.1              193
## 142    Adelie     Dream           40.6          17.2              187
## 143    Adelie     Dream           32.1          15.5              188
## 144    Adelie     Dream           40.7          17.0              190
## 145    Adelie     Dream           37.3          16.8              192
## 146    Adelie     Dream           39.0          18.7              185
## 147    Adelie     Dream           39.2          18.6              190
## 148    Adelie     Dream           36.6          18.4              184
## 149    Adelie     Dream           36.0          17.8              195
## 150    Adelie     Dream           37.8          18.1              193
## 151    Adelie     Dream           36.0          17.1              187
## 152    Adelie     Dream           41.5          18.5              201
## 153    Gentoo    Biscoe           46.1          13.2              211
## 154    Gentoo    Biscoe           50.0          16.3              230
## 155    Gentoo    Biscoe           48.7          14.1              210
```

```
## 156    Gentoo    Biscoe          50.0          15.2              218
## 157    Gentoo    Biscoe          47.6          14.5              215
## 158    Gentoo    Biscoe          46.5          13.5              210
## 159    Gentoo    Biscoe          45.4          14.6              211
## 160    Gentoo    Biscoe          46.7          15.3              219
## 161    Gentoo    Biscoe          43.3          13.4              209
## 162    Gentoo    Biscoe          46.8          15.4              215
## 163    Gentoo    Biscoe          40.9          13.7              214
## 164    Gentoo    Biscoe          49.0          16.1              216
## 165    Gentoo    Biscoe          45.5          13.7              214
## 166    Gentoo    Biscoe          48.4          14.6              213
## 167    Gentoo    Biscoe          45.8          14.6              210
## 168    Gentoo    Biscoe          49.3          15.7              217
## 169    Gentoo    Biscoe          42.0          13.5              210
## 170    Gentoo    Biscoe          49.2          15.2              221
## 171    Gentoo    Biscoe          46.2          14.5              209
## 172    Gentoo    Biscoe          48.7          15.1              222
## 173    Gentoo    Biscoe          50.2          14.3              218
## 174    Gentoo    Biscoe          45.1          14.5              215
## 175    Gentoo    Biscoe          46.5          14.5              213
## 176    Gentoo    Biscoe          46.3          15.8              215
## 177    Gentoo    Biscoe          42.9          13.1              215
## 178    Gentoo    Biscoe          46.1          15.1              215
## 179    Gentoo    Biscoe          44.5          14.3              216
## 180    Gentoo    Biscoe          47.8          15.0              215
## 181    Gentoo    Biscoe          48.2          14.3              210
## 182    Gentoo    Biscoe          50.0          15.3              220
## 183    Gentoo    Biscoe          47.3          15.3              222
## 184    Gentoo    Biscoe          42.8          14.2              209
## 185    Gentoo    Biscoe          45.1          14.5              207
## 186    Gentoo    Biscoe          59.6          17.0              230
## 187    Gentoo    Biscoe          49.1          14.8              220
## 188    Gentoo    Biscoe          48.4          16.3              220
## 189    Gentoo    Biscoe          42.6          13.7              213
## 190    Gentoo    Biscoe          44.4          17.3              219
## 191    Gentoo    Biscoe          44.0          13.6              208
## 192    Gentoo    Biscoe          48.7          15.7              208
## 193    Gentoo    Biscoe          42.7          13.7              208
## 194    Gentoo    Biscoe          49.6          16.0              225
## 195    Gentoo    Biscoe          45.3          13.7              210
## 196    Gentoo    Biscoe          49.6          15.0              216
## 197    Gentoo    Biscoe          50.5          15.9              222
## 198    Gentoo    Biscoe          43.6          13.9              217
## 199    Gentoo    Biscoe          45.5          13.9              210
## 200    Gentoo    Biscoe          50.5          15.9              225
## 201    Gentoo    Biscoe          44.9          13.3              213
## 202    Gentoo    Biscoe          45.2          15.8              215
## 203    Gentoo    Biscoe          46.6          14.2              210
## 204    Gentoo    Biscoe          48.5          14.1              220
## 205    Gentoo    Biscoe          45.1          14.4              210
## 206    Gentoo    Biscoe          50.1          15.0              225
## 207    Gentoo    Biscoe          46.5          14.4              217
```

```
## 208    Gentoo    Biscoe            45.0          15.4              220
## 209    Gentoo    Biscoe            43.8          13.9              208
## 210    Gentoo    Biscoe            45.5          15.0              220
## 211    Gentoo    Biscoe            43.2          14.5              208
## 212    Gentoo    Biscoe            50.4          15.3              224
## 213    Gentoo    Biscoe            45.3          13.8              208
## 214    Gentoo    Biscoe            46.2          14.9              221
## 215    Gentoo    Biscoe            45.7          13.9              214
## 216    Gentoo    Biscoe            54.3          15.7              231
## 217    Gentoo    Biscoe            45.8          14.2              219
## 218    Gentoo    Biscoe            49.8          16.8              230
## 219    Gentoo    Biscoe            46.2          14.4              214
## 220    Gentoo    Biscoe            49.5          16.2              229
## 221    Gentoo    Biscoe            43.5          14.2              220
## 222    Gentoo    Biscoe            50.7          15.0              223
## 223    Gentoo    Biscoe            47.7          15.0              216
## 224    Gentoo    Biscoe            46.4          15.6              221
## 225    Gentoo    Biscoe            48.2          15.6              221
## 226    Gentoo    Biscoe            46.5          14.8              217
## 227    Gentoo    Biscoe            46.4          15.0              216
## 228    Gentoo    Biscoe            48.6          16.0              230
## 229    Gentoo    Biscoe            47.5          14.2              209
## 230    Gentoo    Biscoe            51.1          16.3              220
## 231    Gentoo    Biscoe            45.2          13.8              215
## 232    Gentoo    Biscoe            45.2          16.4              223
## 233    Gentoo    Biscoe            49.1          14.5              212
## 234    Gentoo    Biscoe            52.5          15.6              221
## 235    Gentoo    Biscoe            47.4          14.6              212
## 236    Gentoo    Biscoe            50.0          15.9              224
## 237    Gentoo    Biscoe            44.9          13.8              212
## 238    Gentoo    Biscoe            50.8          17.3              228
## 239    Gentoo    Biscoe            43.4          14.4              218
## 240    Gentoo    Biscoe            51.3          14.2              218
## 241    Gentoo    Biscoe            47.5          14.0              212
## 242    Gentoo    Biscoe            52.1          17.0              230
## 243    Gentoo    Biscoe            47.5          15.0              218
## 244    Gentoo    Biscoe            52.2          17.1              228
## 245    Gentoo    Biscoe            45.5          14.5              212
## 246    Gentoo    Biscoe            49.5          16.1              224
## 247    Gentoo    Biscoe            44.5          14.7              214
## 248    Gentoo    Biscoe            50.8          15.7              226
## 249    Gentoo    Biscoe            49.4          15.8              216
## 250    Gentoo    Biscoe            46.9          14.6              222
## 251    Gentoo    Biscoe            48.4          14.4              203
## 252    Gentoo    Biscoe            51.1          16.5              225
## 253    Gentoo    Biscoe            48.5          15.0              219
## 254    Gentoo    Biscoe            55.9          17.0              228
## 255    Gentoo    Biscoe            47.2          15.5              215
## 256    Gentoo    Biscoe            49.1          15.0              228
## 257    Gentoo    Biscoe            47.3          13.8              216
## 258    Gentoo    Biscoe            46.8          16.1              215
## 259    Gentoo    Biscoe            41.7          14.7              210
```

```
## 260    Gentoo    Biscoe           53.4          15.8              219
## 261    Gentoo    Biscoe           43.3          14.0              208
## 262    Gentoo    Biscoe           48.1          15.1              209
## 263    Gentoo    Biscoe           50.5          15.2              216
## 264    Gentoo    Biscoe           49.8          15.9              229
## 265    Gentoo    Biscoe           43.5          15.2              213
## 266    Gentoo    Biscoe           51.5          16.3              230
## 267    Gentoo    Biscoe           46.2          14.1              217
## 268    Gentoo    Biscoe           55.1          16.0              230
## 269    Gentoo    Biscoe           44.5          15.7              217
## 270    Gentoo    Biscoe           48.8          16.2              222
## 271    Gentoo    Biscoe           47.2          13.7              214
## 272    Gentoo    Biscoe             NA            NA               NA
## 273    Gentoo    Biscoe           46.8          14.3              215
## 274    Gentoo    Biscoe           50.4          15.7              222
## 275    Gentoo    Biscoe           45.2          14.8              212
## 276    Gentoo    Biscoe           49.9          16.1              213
## 277 Chinstrap     Dream           46.5          17.9              192
## 278 Chinstrap     Dream           50.0          19.5              196
## 279 Chinstrap     Dream           51.3          19.2              193
## 280 Chinstrap     Dream           45.4          18.7              188
## 281 Chinstrap     Dream           52.7          19.8              197
## 282 Chinstrap     Dream           45.2          17.8              198
## 283 Chinstrap     Dream           46.1          18.2              178
## 284 Chinstrap     Dream           51.3          18.2              197
## 285 Chinstrap     Dream           46.0          18.9              195
## 286 Chinstrap     Dream           51.3          19.9              198
## 287 Chinstrap     Dream           46.6          17.8              193
## 288 Chinstrap     Dream           51.7          20.3              194
## 289 Chinstrap     Dream           47.0          17.3              185
## 290 Chinstrap     Dream           52.0          18.1              201
## 291 Chinstrap     Dream           45.9          17.1              190
## 292 Chinstrap     Dream           50.5          19.6              201
## 293 Chinstrap     Dream           50.3          20.0              197
## 294 Chinstrap     Dream           58.0          17.8              181
## 295 Chinstrap     Dream           46.4          18.6              190
## 296 Chinstrap     Dream           49.2          18.2              195
## 297 Chinstrap     Dream           42.4          17.3              181
## 298 Chinstrap     Dream           48.5          17.5              191
## 299 Chinstrap     Dream           43.2          16.6              187
## 300 Chinstrap     Dream           50.6          19.4              193
## 301 Chinstrap     Dream           46.7          17.9              195
## 302 Chinstrap     Dream           52.0          19.0              197
## 303 Chinstrap     Dream           50.5          18.4              200
## 304 Chinstrap     Dream           49.5          19.0              200
## 305 Chinstrap     Dream           46.4          17.8              191
## 306 Chinstrap     Dream           52.8          20.0              205
## 307 Chinstrap     Dream           40.9          16.6              187
## 308 Chinstrap     Dream           54.2          20.8              201
## 309 Chinstrap     Dream           42.5          16.7              187
## 310 Chinstrap     Dream           51.0          18.8              203
## 311 Chinstrap     Dream           49.7          18.6              195
```

```
## 312 Chinstrap     Dream           47.5          16.8              199
## 313 Chinstrap     Dream           47.6          18.3              195
## 314 Chinstrap     Dream           52.0          20.7              210
## 315 Chinstrap     Dream           46.9          16.6              192
## 316 Chinstrap     Dream           53.5          19.9              205
## 317 Chinstrap     Dream           49.0          19.5              210
## 318 Chinstrap     Dream           46.2          17.5              187
## 319 Chinstrap     Dream           50.9          19.1              196
## 320 Chinstrap     Dream           45.5          17.0              196
## 321 Chinstrap     Dream           50.9          17.9              196
## 322 Chinstrap     Dream           50.8          18.5              201
## 323 Chinstrap     Dream           50.1          17.9              190
## 324 Chinstrap     Dream           49.0          19.6              212
## 325 Chinstrap     Dream           51.5          18.7              187
## 326 Chinstrap     Dream           49.8          17.3              198
## 327 Chinstrap     Dream           48.1          16.4              199
## 328 Chinstrap     Dream           51.4          19.0              201
## 329 Chinstrap     Dream           45.7          17.3              193
## 330 Chinstrap     Dream           50.7          19.7              203
## 331 Chinstrap     Dream           42.5          17.3              187
## 332 Chinstrap     Dream           52.2          18.8              197
## 333 Chinstrap     Dream           45.2          16.6              191
## 334 Chinstrap     Dream           49.3          19.9              203
## 335 Chinstrap     Dream           50.2          18.8              202
## 336 Chinstrap     Dream           45.6          19.4              194
## 337 Chinstrap     Dream           51.9          19.5              206
## 338 Chinstrap     Dream           46.8          16.5              189
## 339 Chinstrap     Dream           45.7          17.0              195
## 340 Chinstrap     Dream           55.8          19.8              207
## 341 Chinstrap     Dream           43.5          18.1              202
## 342 Chinstrap     Dream           49.6          18.2              193
## 343 Chinstrap     Dream           50.8          19.0              210
## 344 Chinstrap     Dream           50.2          18.7              198
##     body_mass_g    sex year
## 1          3750   male 2007
## 2          3800 female 2007
## 3          3250 female 2007
## 4            NA   <NA> 2007
## 5          3450 female 2007
## 6          3650   male 2007
## 7          3625 female 2007
## 8          4675   male 2007
## 9          3475   <NA> 2007
## 10         4250   <NA> 2007
## 11         3300   <NA> 2007
## 12         3700   <NA> 2007
## 13         3200 female 2007
## 14         3800   male 2007
## 15         4400   male 2007
## 16         3700 female 2007
## 17         3450 female 2007
## 18         4500   male 2007
```

```
## 19          3325 female 2007
## 20          4200   male 2007
## 21          3400 female 2007
## 22          3600   male 2007
## 23          3800 female 2007
## 24          3950   male 2007
## 25          3800   male 2007
## 26          3800 female 2007
## 27          3550   male 2007
## 28          3200 female 2007
## 29          3150 female 2007
## 30          3950   male 2007
## 31          3250 female 2007
## 32          3900   male 2007
## 33          3300 female 2007
## 34          3900   male 2007
## 35          3325 female 2007
## 36          4150   male 2007
## 37          3950   male 2007
## 38          3550 female 2007
## 39          3300 female 2007
## 40          4650   male 2007
## 41          3150 female 2007
## 42          3900   male 2007
## 43          3100 female 2007
## 44          4400   male 2007
## 45          3000 female 2007
## 46          4600   male 2007
## 47          3425   male 2007
## 48          2975   <NA> 2007
## 49          3450 female 2007
## 50          4150   male 2007
## 51          3500 female 2008
## 52          4300   male 2008
## 53          3450 female 2008
## 54          4050   male 2008
## 55          2900 female 2008
## 56          3700   male 2008
## 57          3550 female 2008
## 58          3800   male 2008
## 59          2850 female 2008
## 60          3750   male 2008
## 61          3150 female 2008
## 62          4400   male 2008
## 63          3600 female 2008
## 64          4050   male 2008
## 65          2850 female 2008
## 66          3950   male 2008
## 67          3350 female 2008
## 68          4100   male 2008
## 69          3050 female 2008
## 70          4450   male 2008
```

```
## 71         3600 female 2008
## 72         3900   male 2008
## 73         3550 female 2008
## 74         4150   male 2008
## 75         3700 female 2008
## 76         4250   male 2008
## 77         3700 female 2008
## 78         3900   male 2008
## 79         3550 female 2008
## 80         4000   male 2008
## 81         3200 female 2008
## 82         4700   male 2008
## 83         3800 female 2008
## 84         4200   male 2008
## 85         3350 female 2008
## 86         3550   male 2008
## 87         3800   male 2008
## 88         3500 female 2008
## 89         3950   male 2008
## 90         3600 female 2008
## 91         3550 female 2008
## 92         4300   male 2008
## 93         3400 female 2008
## 94         4450   male 2008
## 95         3300 female 2008
## 96         4300   male 2008
## 97         3700 female 2008
## 98         4350   male 2008
## 99         2900 female 2008
## 100        4100   male 2008
## 101        3725 female 2009
## 102        4725   male 2009
## 103        3075 female 2009
## 104        4250   male 2009
## 105        2925 female 2009
## 106        3550   male 2009
## 107        3750 female 2009
## 108        3900   male 2009
## 109        3175 female 2009
## 110        4775   male 2009
## 111        3825 female 2009
## 112        4600   male 2009
## 113        3200 female 2009
## 114        4275   male 2009
## 115        3900 female 2009
## 116        4075   male 2009
## 117        2900 female 2009
## 118        3775   male 2009
## 119        3350 female 2009
## 120        3325   male 2009
## 121        3150 female 2009
## 122        3500   male 2009
```

```
## 123        3450 female 2009
## 124        3875   male 2009
## 125        3050 female 2009
## 126        4000   male 2009
## 127        3275 female 2009
## 128        4300   male 2009
## 129        3050 female 2009
## 130        4000   male 2009
## 131        3325 female 2009
## 132        3500   male 2009
## 133        3500 female 2009
## 134        4475   male 2009
## 135        3425 female 2009
## 136        3900   male 2009
## 137        3175 female 2009
## 138        3975   male 2009
## 139        3400 female 2009
## 140        4250   male 2009
## 141        3400 female 2009
## 142        3475   male 2009
## 143        3050 female 2009
## 144        3725   male 2009
## 145        3000 female 2009
## 146        3650   male 2009
## 147        4250   male 2009
## 148        3475 female 2009
## 149        3450 female 2009
## 150        3750   male 2009
## 151        3700 female 2009
## 152        4000   male 2009
## 153        4500 female 2007
## 154        5700   male 2007
## 155        4450 female 2007
## 156        5700   male 2007
## 157        5400   male 2007
## 158        4550 female 2007
## 159        4800 female 2007
## 160        5200   male 2007
## 161        4400 female 2007
## 162        5150   male 2007
## 163        4650 female 2007
## 164        5550   male 2007
## 165        4650 female 2007
## 166        5850   male 2007
## 167        4200 female 2007
## 168        5850   male 2007
## 169        4150 female 2007
## 170        6300   male 2007
## 171        4800 female 2007
## 172        5350   male 2007
## 173        5700   male 2007
## 174        5000 female 2007
```

```
## 175         4400 female 2007
## 176         5050   male 2007
## 177         5000 female 2007
## 178         5100   male 2007
## 179         4100   <NA> 2007
## 180         5650   male 2007
## 181         4600 female 2007
## 182         5550   male 2007
## 183         5250   male 2007
## 184         4700 female 2007
## 185         5050 female 2007
## 186         6050   male 2007
## 187         5150 female 2008
## 188         5400   male 2008
## 189         4950 female 2008
## 190         5250   male 2008
## 191         4350 female 2008
## 192         5350   male 2008
## 193         3950 female 2008
## 194         5700   male 2008
## 195         4300 female 2008
## 196         4750   male 2008
## 197         5550   male 2008
## 198         4900 female 2008
## 199         4200 female 2008
## 200         5400   male 2008
## 201         5100 female 2008
## 202         5300   male 2008
## 203         4850 female 2008
## 204         5300   male 2008
## 205         4400 female 2008
## 206         5000   male 2008
## 207         4900 female 2008
## 208         5050   male 2008
## 209         4300 female 2008
## 210         5000   male 2008
## 211         4450 female 2008
## 212         5550   male 2008
## 213         4200 female 2008
## 214         5300   male 2008
## 215         4400 female 2008
## 216         5650   male 2008
## 217         4700 female 2008
## 218         5700   male 2008
## 219         4650   <NA> 2008
## 220         5800   male 2008
## 221         4700 female 2008
## 222         5550   male 2008
## 223         4750 female 2008
## 224         5000   male 2008
## 225         5100   male 2008
## 226         5200 female 2008
```

```
## 227        4700 female 2008
## 228        5800   male 2008
## 229        4600 female 2008
## 230        6000   male 2008
## 231        4750 female 2008
## 232        5950   male 2008
## 233        4625 female 2009
## 234        5450   male 2009
## 235        4725 female 2009
## 236        5350   male 2009
## 237        4750 female 2009
## 238        5600   male 2009
## 239        4600 female 2009
## 240        5300   male 2009
## 241        4875 female 2009
## 242        5550   male 2009
## 243        4950 female 2009
## 244        5400   male 2009
## 245        4750 female 2009
## 246        5650   male 2009
## 247        4850 female 2009
## 248        5200   male 2009
## 249        4925   male 2009
## 250        4875 female 2009
## 251        4625 female 2009
## 252        5250   male 2009
## 253        4850 female 2009
## 254        5600   male 2009
## 255        4975 female 2009
## 256        5500   male 2009
## 257        4725   <NA> 2009
## 258        5500   male 2009
## 259        4700 female 2009
## 260        5500   male 2009
## 261        4575 female 2009
## 262        5500   male 2009
## 263        5000 female 2009
## 264        5950   male 2009
## 265        4650 female 2009
## 266        5500   male 2009
## 267        4375 female 2009
## 268        5850   male 2009
## 269        4875   <NA> 2009
## 270        6000   male 2009
## 271        4925 female 2009
## 272          NA   <NA> 2009
## 273        4850 female 2009
## 274        5750   male 2009
## 275        5200 female 2009
## 276        5400   male 2009
## 277        3500 female 2007
## 278        3900   male 2007
```

```
## 279        3650   male 2007
## 280        3525 female 2007
## 281        3725   male 2007
## 282        3950 female 2007
## 283        3250 female 2007
## 284        3750   male 2007
## 285        4150 female 2007
## 286        3700   male 2007
## 287        3800 female 2007
## 288        3775   male 2007
## 289        3700 female 2007
## 290        4050   male 2007
## 291        3575 female 2007
## 292        4050   male 2007
## 293        3300   male 2007
## 294        3700 female 2007
## 295        3450 female 2007
## 296        4400   male 2007
## 297        3600 female 2007
## 298        3400   male 2007
## 299        2900 female 2007
## 300        3800   male 2007
## 301        3300 female 2007
## 302        4150   male 2007
## 303        3400 female 2008
## 304        3800   male 2008
## 305        3700 female 2008
## 306        4550   male 2008
## 307        3200 female 2008
## 308        4300   male 2008
## 309        3350 female 2008
## 310        4100   male 2008
## 311        3600   male 2008
## 312        3900 female 2008
## 313        3850 female 2008
## 314        4800   male 2008
## 315        2700 female 2008
## 316        4500   male 2008
## 317        3950   male 2008
## 318        3650 female 2008
## 319        3550   male 2008
## 320        3500 female 2008
## 321        3675 female 2009
## 322        4450   male 2009
## 323        3400 female 2009
## 324        4300   male 2009
## 325        3250   male 2009
## 326        3675 female 2009
## 327        3325 female 2009
## 328        3950   male 2009
## 329        3600 female 2009
## 330        4050   male 2009
```

```
## 331        3350 female 2009
## 332        3450   male 2009
## 333        3250 female 2009
## 334        4050   male 2009
## 335        3800   male 2009
## 336        3525 female 2009
## 337        3950   male 2009
## 338        3650 female 2009
## 339        3650 female 2009
## 340        4000   male 2009
## 341        3400 female 2009
## 342        3775   male 2009
## 343        4100   male 2009
## 344        3775 female 2009
```

So that's a little more like R just vomited data into your console with no description of its size or the types of variables in it. To see that, we would need the str() function in base R.

```
str(penguins_df)
```

```
## 'data.frame':    344 obs. of  8 variables:
##  $ species          : Factor w/ 3 levels "Adelie","Chinstrap",..: 1 1 1 1 1 1 1
1 1 1 ...
##  $ island           : Factor w/ 3 levels "Biscoe","Dream",..: 3 3 3 3 3 3 3 3 3
3 ...
##  $ bill_length_mm   : num  39.1 39.5 40.3 NA 36.7 39.3 38.9 39.2 34.1 42 ...
##  $ bill_depth_mm    : num  18.7 17.4 18 NA 19.3 20.6 17.8 19.6 18.1 20.2 ...
##  $ flipper_length_mm: int  181 186 195 NA 193 190 181 195 193 190 ...
##  $ body_mass_g      : int  3750 3800 3250 NA 3450 3650 3625 4675 3475 4250 ...
##  $ sex              : Factor w/ 2 levels "female","male": 2 1 1 NA 1 2 1 2 NA NA
...
##  $ year             : int  2007 2007 2007 2007 2007 2007 2007 2007 2007 2007 ...
```

Tibbles streamline this for us a bit.

You can always get data into a tibble using `as_tibble()` . Let's try that with the penguins_df we just created and double-check that it looks like our original data set again.

```
penguins_tibble = as_tibble(penguins_df)
print(penguins_tibble)
```

```
## # A tibble: 344 × 8
##    species island    bill_length_mm bill_depth_mm flipper_…¹ body_…² sex    year
##    <fct>   <fct>              <dbl>         <dbl>      <int>   <int> <fct> <int>
##  1 Adelie  Torgersen           39.1          18.7        181    3750 male   2007
##  2 Adelie  Torgersen           39.5          17.4        186    3800 fema…  2007
##  3 Adelie  Torgersen           40.3          18          195    3250 fema…  2007
##  4 Adelie  Torgersen           NA            NA           NA      NA <NA>   2007
##  5 Adelie  Torgersen           36.7          19.3        193    3450 fema…  2007
##  6 Adelie  Torgersen           39.3          20.6        190    3650 male   2007
##  7 Adelie  Torgersen           38.9          17.8        181    3625 fema…  2007
##  8 Adelie  Torgersen           39.2          19.6        195    4675 male   2007
##  9 Adelie  Torgersen           34.1          18.1        193    3475 <NA>   2007
## 10 Adelie  Torgersen           42            20.2        190    4250 <NA>   2007
## # … with 334 more rows, and abbreviated variable names ¹flipper_length_mm,
## #   ²body_mass_g
```

Beautiful.

> - You can change a `tibble` to a `data.frame` using the `as.data.frame()` function.
> - You can change a `data.frame` to a `tibble` using the `as_tibble()` function.
> - You can create a `tibble` using the `tibble()` function [the syntax is largely similar to data.frame(), but with a few handy differences].

Getting back to the data, we see that we have variables for species, island, 4 different morphological variables, sex, and the year of observation. Let’s explore some of the tools that *dplyr* gives us for working with these data.

select() a column (or columns) of interest

If we’re interested in sexual dimorphism of penguins, we’ll certainly need to visualize and analyze some of those morphological variables. But how do we work with a single variable within this data set that has many variables? We can use the `select()` function to subset the data.

Let’s look at body mass first, since this seems to be the most direct measure of overall size.

```
select(penguins, body_mass_g) ## select(tibble, column)
```

```
## # A tibble: 344 × 1
##    body_mass_g
##          <int>
##  1        3750
##  2        3800
##  3        3250
##  4          NA
##  5        3450
##  6        3650
##  7        3625
##  8        4675
##  9        3475
## 10        4250
## # … with 334 more rows
```

Note that this is *still a tibble*, even though one could just as easily think of it as one string of numbers (what R calls a `vector`). Also, the data are still considered to be integer data. Using `select()` never changes the type of data. In the tidyverse, we have to be explicit about any changes we wish to make to the representation of our data. For example, if obtaining the variable as a `vector` really is the goal, we can accomplish that using the `pull()` function.

```
pull(penguins, body_mass_g)
```

```
##   [1] 3750 3800 3250   NA 3450 3650 3625 4675 3475 4250 3300 3700 3200 3800 4400
##  [16] 3700 3450 4500 3325 4200 3400 3600 3800 3950 3800 3800 3550 3200 3150 3950
##  [31] 3250 3900 3300 3900 3325 4150 3950 3550 3300 4650 3150 3900 3100 4400 3000
##  [46] 4600 3425 2975 3450 4150 3500 4300 3450 4050 2900 3700 3550 3800 2850 3750
##  [61] 3150 4400 3600 4050 2850 3950 3350 4100 3050 4450 3600 3900 3550 4150 3700
##  [76] 4250 3700 3900 3550 4000 3200 4700 3800 4200 3350 3550 3800 3500 3950 3600
##  [91] 3550 4300 3400 4450 3300 4300 3700 4350 2900 4100 3725 4725 3075 4250 2925
## [106] 3550 3750 3900 3175 4775 3825 4600 3200 4275 3900 4075 2900 3775 3350 3325
## [121] 3150 3500 3450 3875 3050 4000 3275 4300 3050 4000 3325 3500 3500 4475 3425
## [136] 3900 3175 3975 3400 4250 3400 3475 3050 3725 3000 3650 4250 3475 3450 3750
## [151] 3700 4000 4500 5700 4450 5700 5400 4550 4800 5200 4400 5150 4650 5550 4650
## [166] 5850 4200 5850 4150 6300 4800 5350 5700 5000 4400 5050 5000 5100 4100 5650
## [181] 4600 5550 5250 4700 5050 6050 5150 5400 4950 5250 4350 5350 3950 5700 4300
## [196] 4750 5550 4900 4200 5400 5100 5300 4850 5300 4400 5000 4900 5050 4300 5000
## [211] 4450 5550 4200 5300 4400 5650 4700 5700 4650 5800 4700 5550 4750 5000 5100
## [226] 5200 4700 5800 4600 6000 4750 5950 4625 5450 4725 5350 4750 5600 4600 5300
## [241] 4875 5550 4950 5400 4750 5650 4850 5200 4925 4875 4625 5250 4850 5600 4975
## [256] 5500 4725 5500 4700 5500 4575 5500 5000 5950 4650 5500 4375 5850 4875 6000
## [271] 4925   NA 4850 5750 5200 5400 3500 3900 3650 3525 3725 3950 3250 3750 4150
## [286] 3700 3800 3775 3700 4050 3575 4050 3300 3700 3450 4400 3600 3400 2900 3800
## [301] 3300 4150 3400 3800 3700 4550 3200 4300 3350 4100 3600 3900 3850 4800 2700
## [316] 4500 3950 3650 3550 3500 3675 4450 3400 4300 3250 3675 3325 3950 3600 4050
## [331] 3350 3450 3250 4050 3800 3525 3950 3650 3650 4000 3400 3775 4100 3775
```

How does this differ from base R? Well, let's use our data.frame we created a moment ago to see. We can select a single column from a data.frame in base R using the `$` operator.

```
penguins_df$body_mass_g ## subsetting by variable using base R's $ notation
```

```
##   [1] 3750 3800 3250   NA 3450 3650 3625 4675 3475 4250 3300 3700 3200 3800 4400
##  [16] 3700 3450 4500 3325 4200 3400 3600 3800 3950 3800 3800 3550 3200 3150 3950
##  [31] 3250 3900 3300 3900 3325 4150 3950 3550 3300 4650 3150 3900 3100 4400 3000
##  [46] 4600 3425 2975 3450 4150 3500 4300 3450 4050 2900 3700 3550 3800 2850 3750
##  [61] 3150 4400 3600 4050 2850 3950 3350 4100 3050 4450 3600 3900 3550 4150 3700
##  [76] 4250 3700 3900 3550 4000 3200 4700 3800 4200 3350 3550 3800 3500 3950 3600
##  [91] 3550 4300 3400 4450 3300 4300 3700 4350 2900 4100 3725 4725 3075 4250 2925
## [106] 3550 3750 3900 3175 4775 3825 4600 3200 4275 3900 4075 2900 3775 3350 3325
## [121] 3150 3500 3450 3875 3050 4000 3275 4300 3050 4000 3325 3500 3500 4475 3425
## [136] 3900 3175 3975 3400 4250 3400 3475 3050 3725 3000 3650 4250 3475 3450 3750
## [151] 3700 4000 4500 5700 4450 5700 5400 4550 4800 5200 4400 5150 4650 5550 4650
## [166] 5850 4200 5850 4150 6300 4800 5350 5700 5000 4400 5050 5000 5100 4100 5650
## [181] 4600 5550 5250 4700 5050 6050 5150 5400 4950 5250 4350 5350 3950 5700 4300
## [196] 4750 5550 4900 4200 5400 5100 5300 4850 5300 4400 5000 4900 5050 4300 5000
## [211] 4450 5550 4200 5300 4400 5650 4700 5700 4650 5800 4700 5550 4750 5000 5100
## [226] 5200 4700 5800 4600 6000 4750 5950 4625 5450 4725 5350 4750 5600 4600 5300
## [241] 4875 5550 4950 5400 4750 5650 4850 5200 4925 4875 4625 5250 4850 5600 4975
## [256] 5500 4725 5500 4700 5500 4575 5500 5000 5950 4650 5500 4375 5850 4875 6000
## [271] 4925   NA 4850 5750 5200 5400 3500 3900 3650 3525 3725 3950 3250 3750 4150
## [286] 3700 3800 3775 3700 4050 3575 4050 3300 3700 3450 4400 3600 3400 2900 3800
## [301] 3300 4150 3400 3800 3700 4550 3200 4300 3350 4100 3600 3900 3850 4800 2700
## [316] 4500 3950 3650 3550 3500 3675 4450 3400 4300 3250 3675 3325 3950 3600 4050
## [331] 3350 3450 3250 4050 3800 3525 3950 3650 3650 4000 3400 3775 4100 3775
```

Right, so base R would have transformed our data into a vector without us asking it to do so. This may seem trivial, but in more complicated situations these unintended changes to data upon subsetting can create a lot of problems.

Back to sexual dimorphism…

What we'd really like to do is to look at morphological measures like body mass *grouped by sex*. Can we get a `tibble` with both of those variables?

```
select(penguins, body_mass_g, sex) ## select(tibble, column)
```

```
## # A tibble: 344 × 2
##    body_mass_g sex
##          <int> <fct>
##  1        3750 male
##  2        3800 female
##  3        3250 female
##  4          NA <NA>
##  5        3450 female
##  6        3650 male
##  7        3625 female
##  8        4675 male
##  9        3475 <NA>
## 10        4250 <NA>
## # … with 334 more rows
```

Sure! Just add more of the variables in the original `tibble` to the `select()` function's arguments to retain them in the selection. There are lots of other interesting options for selecting and combining multiple variables that you can find in the help page for the select() function.

filter() rows based on some criteria

If we're interested in sexual dimorphism, we'll undoubtedly want to be able to look at the data for each sex separately at some point. We can do that with the `filter()` function, which will take only the rows of our data that meet some criterion. The criterion can be a lot of different things, but usually it's based on the value of one or more of the variables in the data set. For example, let's see if we can get the body mass measurements for only females (that is, rows of the data set for which the sex variable is equal to "female").

To do this, we need to select() the body mass and sex variables and then filter() the data for only the females. There are a couple of ways to package these two steps together.

1. intermediate steps – select data, save it, filter it, save again.
2. nesting steps – use select() as the data input argument inside the filter() function
3. piping – use the `%>%` operator to forward output from one operation to the next (tidyverse specific!)

The first option is easy to follow, but can quickly clutter your R environment with a lot of similarly named intermediate objects. The second option simply nests functions to avoid creating intermediate objects – this is very handy in moderation, but overzealous nesting brings many, many parentheses and much confusion. The last option is the tidyverse's solution to this tradeoff. Using the "pipe" operator `%>%`, we can simply push the output of one operation to the next in a "pipeline" that both removes intermediates and is easy to follow.

```
# Intermediate Steps #
bodyMassDat =  select(penguins, body_mass_g, sex) # select body mass and sex and sa
ve as a new tibble
filter(bodyMassDat, sex == "female") # filter bodyMassDat for only the rows for whi
ch sex is female
```

```
## # A tibble: 165 × 2
##    body_mass_g sex
##          <int> <fct>
##  1        3800 female
##  2        3250 female
##  3        3450 female
##  4        3625 female
##  5        3200 female
##  6        3700 female
##  7        3450 female
##  8        3325 female
##  9        3400 female
## 10        3800 female
## # … with 155 more rows
```

```
# Nested Steps #
filter(select(penguins, body_mass_g, sex), sex == "female") # use select() inside f
ilter() to select variables before filter
```

```
## # A tibble: 165 × 2
##    body_mass_g sex
##          <int> <fct>
##  1        3800 female
##  2        3250 female
##  3        3450 female
##  4        3625 female
##  5        3200 female
##  6        3700 female
##  7        3450 female
##  8        3325 female
##  9        3400 female
## 10        3800 female
## # … with 155 more rows
```

```
# Piping #
select(penguins, body_mass_g, sex) %>% filter(sex == "female") # use select, "pipe"
output forward to filter()
```

```
## # A tibble: 165 × 2
##    body_mass_g sex
##          <int> <fct>
##  1        3800 female
##  2        3250 female
##  3        3450 female
##  4        3625 female
##  5        3200 female
##  6        3700 female
##  7        3450 female
##  8        3325 female
##  9        3400 female
## 10        3800 female
## # … with 155 more rows
```

Here I used the `==` operator to specify the criterion that the variable `sex` must be equal to "female". There are lots of filter options available. We could filter rows for which a certain variable is less than or greater than a certain quantity using `<` and `<=` type operators, and we can find rows where a certain variable is missing using `is.na()`. This can be especially helpful with the complement operator `!`, for example to get the all the rows where a certian variable is *not* missing (try `filter(data, !is.na(variable))`).

You can also filter by multiple criteria as well using logical operators like and (`&`) and or (`|`). Maybe we only want to look at data from females on Torgersen Island, for example. We'd need to `select()` the island variable from our data set as well, and then `filter()` for the rows where sex == female *and* island == Torgersen.

```
select(penguins, island, body_mass_g, sex) %>% # use select to get island, body mas
s, sex from the penguins data set, pipe output forward
  filter(sex == "female" & island == "Torgersen") # then filter for rows where sex
is female and island is Torgersen
```

```
## # A tibble: 24 × 3
##    island    body_mass_g sex
##    <fct>           <int> <fct>
##  1 Torgersen        3800 female
##  2 Torgersen        3250 female
##  3 Torgersen        3450 female
##  4 Torgersen        3625 female
##  5 Torgersen        3200 female
##  6 Torgersen        3700 female
##  7 Torgersen        3450 female
##  8 Torgersen        3325 female
##  9 Torgersen        3050 female
## 10 Torgersen        3600 female
## # … with 14 more rows
```

Note that the "pipeline" starts to get too long for one line. Just make sure to place returns *after* a `%>%` and R will recognize that you are still mid-pipeline and continue to the next line.

You may have also noticed that when we used the intermediate steps or nested steps approaches, we had to specify the data used for the filter function. When we used a pipe, however, we only needed to specify the filtering criteria– the data is assumed to be the output of the preceding step. This is another handy feature of the “piping” approach.

mutate() to create new variables

Often, raw data do not have all of the information we would like to analyse. For example, I would be interested to know if female and male penguins differ not only in their size or mass, but in their body condition. We might imagine a “penguin plumpness” index, where individuals that have large values have greater mass for their length. One (rather crude) way of calculating such a measure might be to divide each penguin’s mass by its flipper length. Then, individuals with larger penguin plumpness indices (PPIs) will be those with more mass given (roughly) their size.

```
mutate(penguins, pengPlumpInd = body_mass_g / flipper_length_mm) # mutate(data, new
VariableName = ...)
```

```
## # A tibble: 344 × 9
##    species island    bill_length_mm bill_d…¹ flipp…² body_…³ sex    year pengP…⁴
##    <fct>   <fct>              <dbl>    <dbl>   <int>   <int> <fct> <int>   <dbl>
##  1 Adelie  Torgersen           39.1     18.7     181    3750 male   2007    20.7
##  2 Adelie  Torgersen           39.5     17.4     186    3800 fema…  2007    20.4
##  3 Adelie  Torgersen           40.3     18       195    3250 fema…  2007    16.7
##  4 Adelie  Torgersen           NA       NA        NA      NA <NA>   2007    NA
##  5 Adelie  Torgersen           36.7     19.3     193    3450 fema…  2007    17.9
##  6 Adelie  Torgersen           39.3     20.6     190    3650 male   2007    19.2
##  7 Adelie  Torgersen           38.9     17.8     181    3625 fema…  2007    20.0
##  8 Adelie  Torgersen           39.2     19.6     195    4675 male   2007    24.0
##  9 Adelie  Torgersen           34.1     18.1     193    3475 <NA>   2007    18.0
## 10 Adelie  Torgersen           42       20.2     190    4250 <NA>   2007    22.4
## # … with 334 more rows, and abbreviated variable names ¹bill_depth_mm,
## #   ²flipper_length_mm, ³body_mass_g, ⁴pengPlumpInd
```

Great, so the output of the mutate function is the original data, but with a new variable calculated from existing variables using some function (here, just simple division). Very handy. You can apply all manner of functions using `mutate()` , including ones you have written yourself.

If we want to make sure that we don’t calculate PPIs for penguins whose sex was unknown (there are a few in the data!), we could filter before mutating…

```
filter(penguins, !is.na(sex)) %>% mutate(pengPlumpInd = body_mass_g / flipper_lengt
h_mm)
```

```
## # A tibble: 333 × 9
##    species island    bill_length_mm bill_d…¹ flipp…² body_…³ sex    year pengP…⁴
##    <fct>   <fct>              <dbl>    <dbl>   <int>   <int> <fct> <int>   <dbl>
##  1 Adelie  Torgersen           39.1     18.7     181    3750 male   2007    20.7
##  2 Adelie  Torgersen           39.5     17.4     186    3800 fema…  2007    20.4
##  3 Adelie  Torgersen           40.3     18       195    3250 fema…  2007    16.7
##  4 Adelie  Torgersen           36.7     19.3     193    3450 fema…  2007    17.9
##  5 Adelie  Torgersen           39.3     20.6     190    3650 male   2007    19.2
##  6 Adelie  Torgersen           38.9     17.8     181    3625 fema…  2007    20.0
##  7 Adelie  Torgersen           39.2     19.6     195    4675 male   2007    24.0
##  8 Adelie  Torgersen           41.1     17.6     182    3200 fema…  2007    17.6
##  9 Adelie  Torgersen           38.6     21.2     191    3800 male   2007    19.9
## 10 Adelie  Torgersen           34.6     21.1     198    4400 male   2007    22.2
## # … with 323 more rows, and abbreviated variable names ¹bill_depth_mm,
## #   ²flipper_length_mm, ³body_mass_g, ⁴pengPlumpInd
```

And we get a slightly smaller tibble where we've not calculated PPI values for individuals whose sex is unknown anyway. Nice! This is the power of piping.

Note again that we can drop the data argument to `mutate()` because the pipe tells it to use the output from the `filter()` function that comes before.

Split-apply-combine approach to data analysis

Now that we know how to select and filter, we could get split our data up to get body masses and PPIs for females and males separately, apply various analyses to summarize the data for each group, and then combine that into a new tibble summarizing sexual size dimorphism. But then we'd probably want to look at each species separately, and it might be worth asking to what extent sexual size dimorphism varies among islands, or if there is variation among years, and whether these patterns are the same in the remaining morphological measures… whew! If all the required splitting and combining is starting to sound tedious, worry not! The tidyverse is here for you.

There are two functions that streamline this "split-apply-combine" approach to data analysis: `group_by()` and `summarize()`.

group_by()

The `group_by()` function tells R that any tidyverse functions that happen "downstream" in our pipeline should be applied at the level of the group, where the group corresponds to values of some variable (e.g., sex or island).

```
filter(penguins, !is.na(sex)) %>% group_by(sex) # filter for rows for which sex is
*not* NA, and then group the result by sex.
```

```
## # A tibble: 333 × 8
## # Groups:   sex [2]
##    species island    bill_length_mm bill_depth_mm flipper_…¹ body_…² sex    year
##    <fct>   <fct>              <dbl>         <dbl>      <int>   <int> <fct> <int>
##  1 Adelie  Torgersen           39.1          18.7        181    3750 male   2007
##  2 Adelie  Torgersen           39.5          17.4        186    3800 fema…  2007
##  3 Adelie  Torgersen           40.3          18          195    3250 fema…  2007
##  4 Adelie  Torgersen           36.7          19.3        193    3450 fema…  2007
##  5 Adelie  Torgersen           39.3          20.6        190    3650 male   2007
##  6 Adelie  Torgersen           38.9          17.8        181    3625 fema…  2007
##  7 Adelie  Torgersen           39.2          19.6        195    4675 male   2007
##  8 Adelie  Torgersen           41.1          17.6        182    3200 fema…  2007
##  9 Adelie  Torgersen           38.6          21.2        191    3800 male   2007
## 10 Adelie  Torgersen           34.6          21.1        198    4400 male   2007
## # … with 323 more rows, and abbreviated variable names ¹flipper_length_mm,
## #   ²body_mass_g
```

Note that this doesn't change how the data *looks* (other than the tibble now mentions that it is grouped), but it drastically changes how it interacts with subsequent operations. We can group by multiple variables, and even expressions based on variables. For example, let's create 6 groups, one for each sex on each island…

```
filter(penguins, !is.na(sex)) %>% group_by(sex, island) # filter for rows for which sex is *not* NA, and then group the result by sex and island.
```

```
## # A tibble: 333 × 8
## # Groups:   sex, island [6]
##    species island    bill_length_mm bill_depth_mm flipper_…¹ body_…² sex    year
##    <fct>   <fct>              <dbl>         <dbl>      <int>   <int> <fct> <int>
##  1 Adelie  Torgersen           39.1          18.7        181    3750 male   2007
##  2 Adelie  Torgersen           39.5          17.4        186    3800 fema…  2007
##  3 Adelie  Torgersen           40.3          18          195    3250 fema…  2007
##  4 Adelie  Torgersen           36.7          19.3        193    3450 fema…  2007
##  5 Adelie  Torgersen           39.3          20.6        190    3650 male   2007
##  6 Adelie  Torgersen           38.9          17.8        181    3625 fema…  2007
##  7 Adelie  Torgersen           39.2          19.6        195    4675 male   2007
##  8 Adelie  Torgersen           41.1          17.6        182    3200 fema…  2007
##  9 Adelie  Torgersen           38.6          21.2        191    3800 male   2007
## 10 Adelie  Torgersen           34.6          21.1        198    4400 male   2007
## # … with 323 more rows, and abbreviated variable names ¹flipper_length_mm,
## #   ²body_mass_g
```

Minor changes to this pipeline would now allow us to quickly break our data down by species, sex, island, etc. With these groups in hand, we'd like to calculate some summary statistics at the level of the groups. This is where the `summarize()` function comes in handy.

summarize()

The `summarize()` function creates a new data set with one row for each combination of the grouping variable(s) and one column for each summary statistic we specify. Let's try using that to get a data set summarizing the mean body mass for each sex.

We'll filter out the NAs again, group our data by sex, and then create a summary data frame with one row for each sex and one column called "meanBodyMassg" containing the average body mass measurement for individuals in that group (sex).

```
# filter penguins to remove rows for which sex is unknown, then group the data by s
ex, and then calculate mean body mass for each group.
filter(penguins, !is.na(sex)) %>% group_by(sex) %>% summarize(meanBodyMassg = mean
(body_mass_g))
```

```
## # A tibble: 2 × 2
##   sex    meanBodyMassg
##   <fct>          <dbl>
## 1 female         3862.
## 2 male           4546.
```

Cool! So with one small line of code, we can get rid of NAs, group our data by sex, calculate summary statistics for each group, and make a nice tibble of the results. How delightful. Also, I know now that male penguins are, on average, much more massive than females. This is very different than the spiders I am used to studying!

Sex differences in mean body mass are central to our question of sexual size dimorphism, but we'd like to know whether this difference is small or large relative to the natural range of penguin mass. Let's add a measure of variance to our summary. Simply head back to the pipeline and add the standard deviation in body mass to the `summarize()` function's arguments.

```
filter(penguins, !is.na(sex)) %>% group_by(sex) %>% summarize(meanBodyMassg = mean
(body_mass_g), sdBodyMassg = sd(body_mass_g))
```

```
## # A tibble: 2 × 3
##   sex    meanBodyMassg sdBodyMassg
##   <fct>          <dbl>       <dbl>
## 1 female         3862.        666.
## 2 male           4546.        788.
```

So while there's a roughly 700 g difference in the mean mass of females and males, the average penguin differs from their sex-specific average by about that much (666 g for females, 788 g for males.). So while females and males are, on average, pretty different in mass, there's a lot of overlap. I probably wouldn't want to go out and start weighing penguins to determine their sex. That's helpful to know.

We can add as many summary measures as we would like. Let's go ahead and add the mean and standard deviation for PPI as well.

NOTE: we have to add our PPI calculation back into our pipeline to accomplish this. This is because our pipelines don't alter the original data set. They simply call on the data set, perform some operation, and then pass the output down the pipeline. Of course, we could save the output data as an object (e.g., a new tibble) at any time. However, the fact that we can accomplish a lot without necessarily needing to do this is part of what makes piping so nice.

```
# filter() to remove rows where sex is NA,
filter(penguins, !is.na(sex)) %>%
  # mutate() to calculate penguin plumpness index from body mass and flipper length
  mutate(pengPlumpInd = body_mass_g / flipper_length_mm) %>%
  # group data by sex
  group_by(sex) %>%
  # summarize groups in terms of mean body manss, sd in body mass, mean PPI, and sd
in PPI
  summarize(meanBodyMassg = mean(body_mass_g),
            sdBodyMassg = sd(body_mass_g),
            meanPPI = mean(pengPlumpInd),
            sdPPI = sd(pengPlumpInd))
```

```
## # A tibble: 2 × 5
##   sex    meanBodyMassg sdBodyMassg meanPPI sdPPI
##   <fct>          <dbl>       <dbl>   <dbl> <dbl>
## 1 female         3862.        666.    19.5  2.30
## 2 male           4546.        788.    22.1  2.57
```

We can even calculate summary measures based on other summary measures in the same line of code, provided we make sure that we ask R to calculate them in order. For example, we might want to know the coefficient of variation (CV) in body mass for each sex (i.e., the standard deviation in body size divided by the mean body size). As long as we ask to calculate this *after* we calculate the mean and standard deviation, `summarize()` can handle this for us. In this way, we can sort of sneak a "mutate" step into our summarizing.

```
# filter() to remove rows where sex is NA,
filter(penguins, !is.na(sex)) %>%
  # mutate() to calculate penguin plumpness index from body mass and flipper length
  mutate(pengPlumpInd = body_mass_g / flipper_length_mm) %>%
  # group data by sex
  group_by(sex) %>%
  # summarize groups in terms of mean body manss, sd in body mass, mean PPI, and sd
in PPI
  summarize(meanBodyMassg = mean(body_mass_g),
            sdBodyMassg = sd(body_mass_g),
            CVBodyMassg = sdBodyMassg / meanBodyMassg,
            meanPPI = mean(pengPlumpInd),
            sdPPI = sd(pengPlumpInd))
```

```
## # A tibble: 2 × 6
##   sex    meanBodyMassg sdBodyMassg CVBodyMassg meanPPI sdPPI
##   <fct>          <dbl>       <dbl>       <dbl>   <dbl> <dbl>
## 1 female         3862.        666.       0.172    19.5  2.30
## 2 male           4546.        788.       0.173    22.1  2.57
```

Now we can see that while males are more variable in body mass than females (i.e., their `sdBodyMassg` is larger), the variation is really pretty similar once we consider that males are just larger overall (i.e., their `CVBodyMassg` is almost identical).

Let's take this all the way to its limit with these data! We have 3 species, 3 islands, and two sexes we'd like to compare for each. This seems like a logical way to organize the data and summarize them. Let's also get the sample size for each of these sub-groups – we can do this including the `n()` function as one of our summary measures.

```
# filter() to remove rows where sex is NA,
filter(penguins, !is.na(sex)) %>%
  # mutate() to calculate penguin plumpness index from body mass and flipper length
  mutate(pengPlumpInd = body_mass_g / flipper_length_mm) %>%
  # group data by sex
  group_by(species, island, sex) %>%
  # summarize groups in terms of mean body manss, sd in body mass, mean PPI, and sd
in PPI
  summarize(meanBodyMassg = mean(body_mass_g),
            sdBodyMassg = sd(body_mass_g),
            CVBodyMassg = sdBodyMassg / meanBodyMassg,
            meanPPI = mean(pengPlumpInd),
            sdPPI = sd(pengPlumpInd),
            sampleSize = n())
```

```
## `summarise()` has grouped output by 'species', 'island'. You can override using
## the `.groups` argument.
```

```
## # A tibble: 10 × 9
## # Groups:   species, island [5]
##    species   island    sex    meanBodyMa…¹ sdBod…² CVBod…³ meanPPI sdPPI sampl…⁴
##    <fct>     <fct>     <fct>         <dbl>   <dbl>   <dbl>   <dbl> <dbl>   <int>
##  1 Adelie    Biscoe    female        3369.    343.  0.102     18.0  1.67      22
##  2 Adelie    Biscoe    male          4050     356.  0.0878    21.3  1.52      22
##  3 Adelie    Dream     female        3344.    212.  0.0634    17.8  1.09      27
##  4 Adelie    Dream     male          4046.    331.  0.0817    21.1  1.68      28
##  5 Adelie    Torgersen female        3396.    259.  0.0763    18.0  1.48      24
##  6 Adelie    Torgersen male          4035.    372.  0.0923    20.7  1.85      23
##  7 Chinstrap Dream     female        3527.    285.  0.0809    18.4  1.46      34
##  8 Chinstrap Dream     male          3939.    362.  0.0919    19.7  1.49      34
##  9 Gentoo    Biscoe    female        4680.    282.  0.0602    22.0  1.19      58
## 10 Gentoo    Biscoe    male          5485.    313.  0.0571    24.8  1.35      61
## # … with abbreviated variable names ¹meanBodyMassg, ²sdBodyMassg, ³CVBodyMassg,
## #   ⁴sampleSize
```

In some cases it might be useful to sort this summary by some variable of interest. For example, if we were interested in identifying the combination of species-island-sex with the biggest penguins, we could use the `arrange` function to sort the output in order of descending body mass.

```
# filter() to remove rows where sex is NA,
filter(penguins, !is.na(sex)) %>%
  # mutate() to calculate penguin plumpness index from body mass and flipper length
  mutate(pengPlumpInd = body_mass_g / flipper_length_mm) %>%
  # group data by sex
  group_by(species, island, sex) %>%
  # summarize groups in terms of mean body manss, sd in body mass, mean PPI, and sd
in PPI
  summarize(meanBodyMassg = mean(body_mass_g),
            sdBodyMassg = sd(body_mass_g),
            CVBodyMassg = sdBodyMassg / meanBodyMassg,
            meanPPI = mean(pengPlumpInd),
            sdPPI = sd(pengPlumpInd),
            sampleSize = n()) %>%
  # sort by descending body mass to get largest at top -- careful that we sort on t
he new summary variables, not theoriginal variables!
  arrange(desc(meanBodyMassg))
```

```
## `summarise()` has grouped output by 'species', 'island'. You can override using
## the `.groups` argument.
```

```
## # A tibble: 10 × 9
## # Groups:   species, island [5]
##    species   island    sex    meanBodyMa…¹ sdBod…² CVBod…³ meanPPI sdPPI sampl…⁴
##    <fct>     <fct>     <fct>         <dbl>   <dbl>   <dbl>   <dbl> <dbl>   <int>
##  1 Gentoo    Biscoe    male          5485.    313.  0.0571    24.8  1.35      61
##  2 Gentoo    Biscoe    female        4680.    282.  0.0602    22.0  1.19      58
##  3 Adelie    Biscoe    male          4050     356.  0.0878    21.3  1.52      22
##  4 Adelie    Dream     male          4046.    331.  0.0817    21.1  1.68      28
##  5 Adelie    Torgersen male          4035.    372.  0.0923    20.7  1.85      23
##  6 Chinstrap Dream     male          3939.    362.  0.0919    19.7  1.49      34
##  7 Chinstrap Dream     female        3527.    285.  0.0809    18.4  1.46      34
##  8 Adelie    Torgersen female        3396.    259.  0.0763    18.0  1.48      24
##  9 Adelie    Biscoe    female        3369.    343.  0.102     18.0  1.67      22
## 10 Adelie    Dream     female        3344.    212.  0.0634    17.8  1.09      27
## # … with abbreviated variable names ¹meanBodyMassg, ²sdBodyMassg, ³CVBodyMassg,
## #   ⁴sampleSize
```

A someone with absolutely no penguin-related knowledge, this is pretty neat to see! I didn't know that Gentoo penguins are generally more massive and plumper for their body size than Adelie and Chinstrap penguins. And I certainly would've never guessed that Adelie penguins can be bigger *or* smaller than Chinstrap penguins, depending on whether the Adelie in question is male (bigger) or female (smaller). It's also interesting to me that for Adelie penguins, the one species that occurs across all 3 islands, the island with the biggest females (Torgersen) is *not* the same island with the biggest males (Biscoe). That said, the differences in mean body masses and plumpnesses are very small relative to the their standard deviations, so penguins are virtually identical in mass across the islands.

Also, it's interesting to note that the most massive penguins also tend to be more massive for their size (our "PPI"). This probably makes sense given that a small change in a penguin's length produces a larger change in its volume.

Statistical analysis "in the pipeline"

If you're like me, you're probably itching for a statistical test of whether these differences in size are greater than one would expect by chance. While I won't get into the details here of how to choose and implement statistical analyses, it's helpful to know the general approach for working one into your data analysis pipeline.

As an example, let's see if we can `filter()` the data to exclude the rows for which sex is unknown, `mutate` the data to calculate PPI, and then use `lm()` from base R to fit a linear model predicting PPI as a function of species and sex (we'll ignore different islands for now, for simplicity). After fitting the model, we need to test the significance of the estimated effects. We'll use an Analyis of Variance (ANOVA) for this, which is available using the `anova()` function in base R. That's right, we can use base R functions within our pipelines too! If you're keeping count, doing this without pipes would involve either (1) creating 3 intermediate objects or (2) 3 instances of nesting a function inside another function. Pipes are nice for cases like this.

```
# filter() to remove rows where sex is NA,
filter(penguins, !is.na(sex)) %>%
  # mutate() to calculate penguin plumpness index from body mass and flipper length
  mutate(pengPlumpInd = body_mass_g / flipper_length_mm) %>%
  # fit a linear model where pengPlumpInd is the response variable, and sex, specie
s, and their interaction is the predictor
  lm(formula = pengPlumpInd ~ species + sex + species:sex) %>%
  # ask r to analyse that linear model using an ANalysis Of VAriance (ANOVA) using
the anova() function in base R.
  anova()
```

```
## Analysis of Variance Table
##
## Response: pengPlumpInd
##              Df  Sum Sq Mean Sq  F value    Pr(>F)
## species       2 1277.20  638.60 308.2048 < 2.2e-16 ***
## sex           1  561.69  561.69 271.0833 < 2.2e-16 ***
## species:sex   2   38.45   19.23   9.2787 0.0001204 ***
## Residuals   327  677.54    2.07
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The output kindly reminds us what our response variable was (pengPlumpInd) and provides us with a table containing rows for each predictor variable or interaction term and columns for the various parameters calculated during the ANOVA on the linear model. These are all important in their own right, but you'll most often find yourself reporting the the degrees of freedom (*df*), the test stastic (an *F*-value, in this case), and the *p*-value (Pr(>F)).

It seems like there are, on average, significant species and sex differences in PPI (note that the *p*-values associated with these terms fall below the conventional cut-off of 0.05). However, there is also a significant interaction between species and sex. This tells us that the difference between species depends on which sex you consider, and vice versa. This is what we inferred from the summary table we made, but it's good to know that a preliminary statistical analysis supports this intuition.

Plotting “in the pipeline”

Finally, we can also run pipelines into plots. There are really neat tools for doing this that are part of the tidyverse, largely within the *ggplot2* package. Although I won’t get deep into ggplot here, it’s useful to see the general approach to implementing plotting in a pipeline.

Let’s make sure ggplot2 is installed.

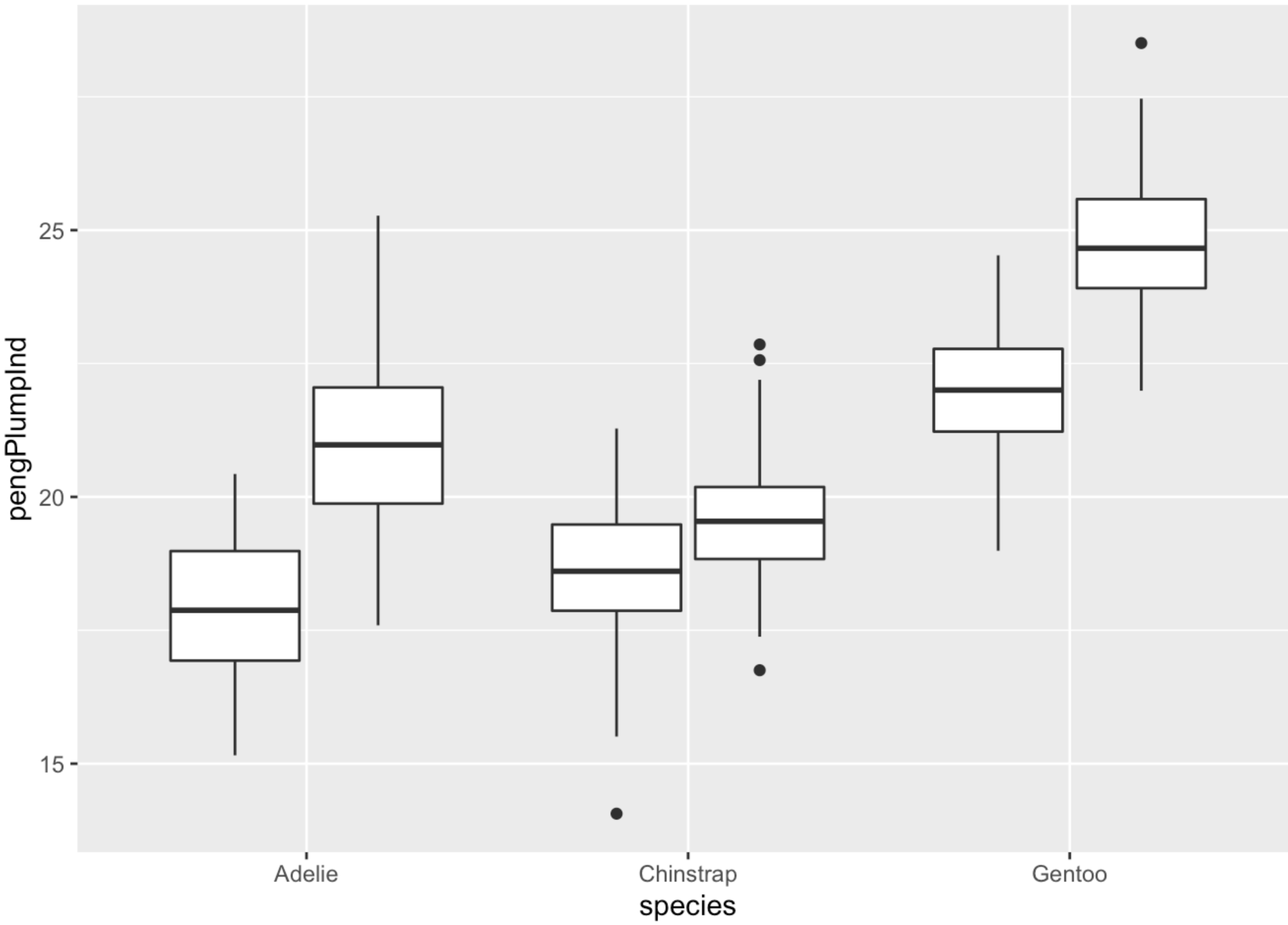
```
install.packages("ggplot2") # install the package
```

And now we need to load it.

```
library(ggplot2) # load it in the current R session.
```

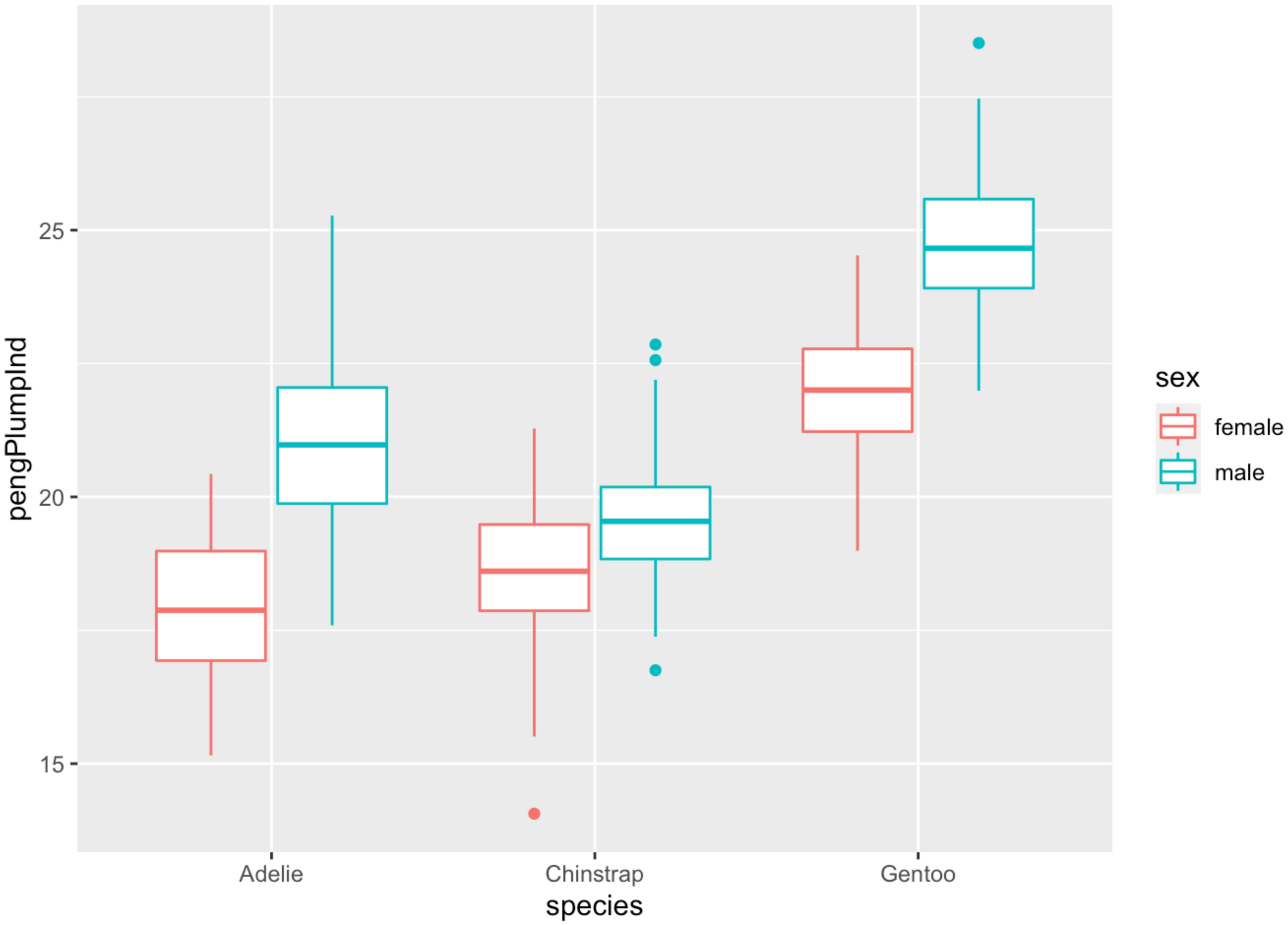
Let’s try to visualize the result from our ANOVA on the PPI of different species and sexes of penguins above. Ideally, we’d like to have a plot that has our response variable on the y-axis and the predictor on the x-axis. Our response variable is PPI, but we have two predictors, sex and species. Since we set out to look at sexual dimorphism, we probably want the male and female PPIs of a given species right next to each other to emphasize sex differences, and then replicate that visual comparison across the three species. We can do with a grouped boxplot, where we plot PPI for each species, but group the data in each species into that for males and that for females.

```
  # filter() to remove rows where sex is NA,
  filter(penguins, !is.na(sex)) %>%
  # mutate() to calculate penguin plumpness index from body mass and flipper length
  mutate(pengPlumpInd = body_mass_g / flipper_length_mm) %>%
  # use ggplot() to plot PPI for each species, grouped by sex.
  # the first term (ggplot()) just sets up the plot object by saying what the x, y,
and grouping variables will be.
  # the second term (geom_boxplot()) specifies a "layer" of the plot's appearance--
in this case, a boxplot.
  # Note that we don't have to specify our data set for either argument because it
is inherited from the pipe.
  # Also note that we don't have to re-specify our x, y, and grouping variable in g
eom_boxplot because it inherits
  # these from the first ggplot term (unless we tell it otherwise).
  ggplot(aes(x = species, y = pengPlumpInd, by = sex)) + # use plus signs to add on
additional ggplot "layers"
  geom_boxplot()
```



We can add an `aes(color = sex)` argument to the boxplot function to get the sexes in different colors…

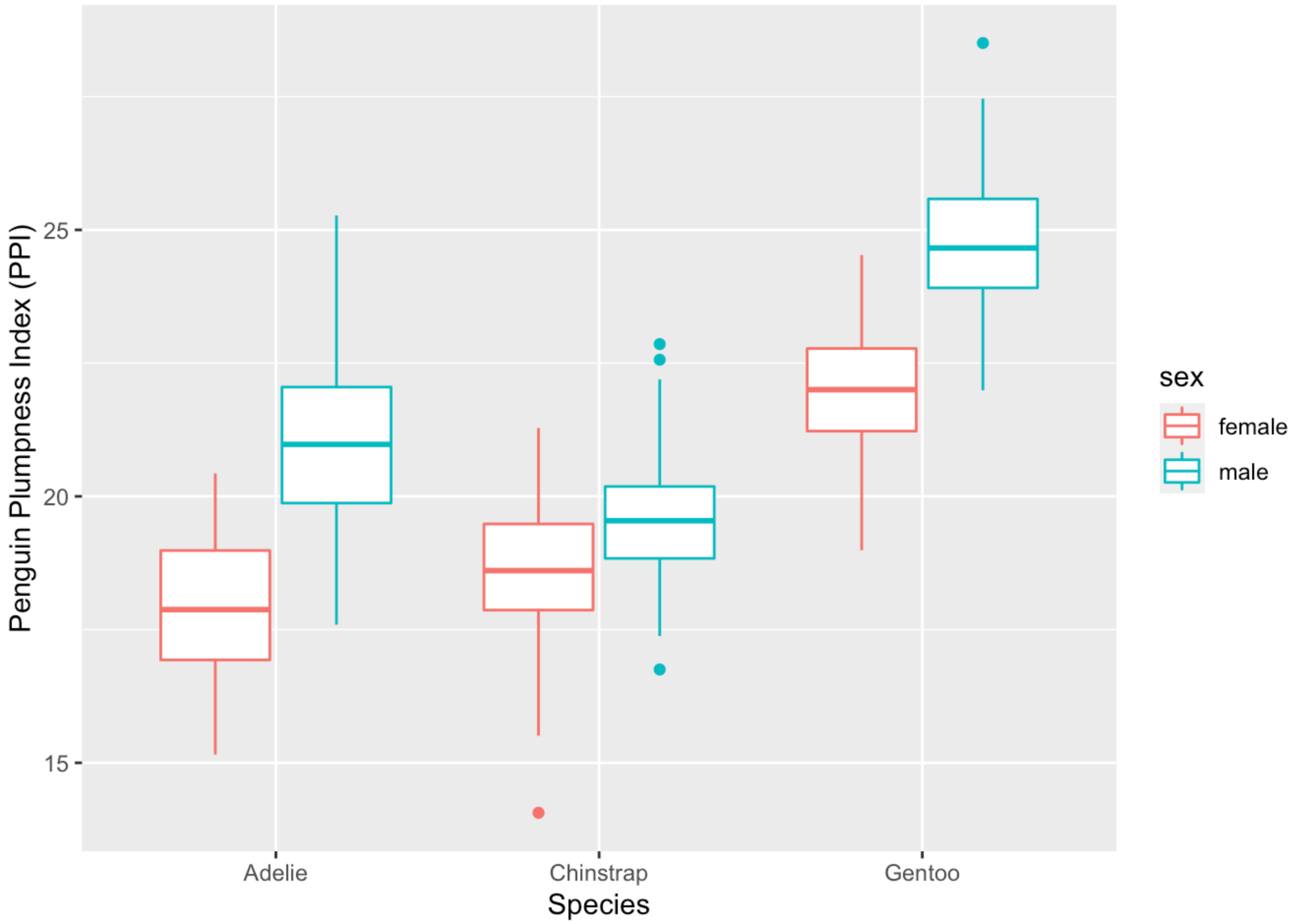
```
# filter() to remove rows where sex is NA,
filter(penguins, !is.na(sex)) %>%
# mutate() to calculate penguin plumpness index from body mass and flipper length
mutate(pengPlumpInd = body_mass_g / flipper_length_mm) %>%
# use ggplot() to plot PPI for each species, grouped by sex.
# This time, specify that boxplots should differ in color depending on sex.
ggplot(aes(x = species, y = pengPlumpInd, by = sex)) +
geom_boxplot(aes(color = sex))
```



Note that if we specify how colors are to be applied by groups in `geom_boxplot()`, we can omit the "by" argument from the first term because applying the color by sex groups the data by sex anyway.

While we're at it, let's add nice axis labels using `xlab()` and `ylab()`.

```
  # filter() to remove rows where sex is NA,
  filter(penguins, !is.na(sex)) %>%
  # mutate() to calculate penguin plumpness index from body mass and flipper length
  mutate(pengPlumpInd = body_mass_g / flipper_length_mm) %>%
  # use ggplot() to plot PPI for each species, grouped by sex.
  # Can omit the "by" argument in ggplot() because we apply colors by the same grou
ping structure in geom_boxplot.
  ggplot(aes(x = species, y = pengPlumpInd)) +
  geom_boxplot(aes(color = sex)) +
  xlab("Species") +
  ylab("Penguin Plumpness Index (PPI)")
```



The tools available through *ggplot2* are pretty staggering and certainly exceed the scope of what I can cover here. The takeaway here is that these tools are developed to work with piping and the tidyverse approach in general.

Moving between long- and wide-format data

It's occasionally necessary to change the format of the data to work with a particular function or workflow in R. Most of the time, R functions will use data that are in "long format". Long format data are data where each variable observed appears in only one column, and there is therefore only one value for a given variable in any row. Let's look at our summary table of the `penguins` data as an example.

```
# filter() to remove rows where sex is NA,
filter(penguins, !is.na(sex)) %>%
  # mutate() to calculate penguin plumpness index from body mass and flipper length
  mutate(pengPlumpInd = body_mass_g / flipper_length_mm) %>%
  # group data by sex
  group_by(species, island, sex) %>%
  # summarize groups in terms of mean body mass, sd in body mass, mean PPI, and sd
in PPI
  summarize(meanBodyMassg = mean(body_mass_g),
            sdBodyMassg = sd(body_mass_g),
            CVBodyMassg = sdBodyMassg / meanBodyMassg,
            meanPPI = mean(pengPlumpInd),
            sdPPI = sd(pengPlumpInd),
            sampleSize = n())
```

```
## `summarise()` has grouped output by 'species', 'island'. You can override using
## the `.groups` argument.
```

```
## # A tibble: 10 × 9
## # Groups:   species, island [5]
##    species   island    sex    meanBodyMa…¹ sdBod…² CVBod…³ meanPPI sdPPI sampl…⁴
##    <fct>     <fct>     <fct>         <dbl>   <dbl>   <dbl>   <dbl> <dbl>   <int>
##  1 Adelie    Biscoe    female        3369.    343.  0.102     18.0  1.67      22
##  2 Adelie    Biscoe    male          4050     356.  0.0878    21.3  1.52      22
##  3 Adelie    Dream     female        3344.    212.  0.0634    17.8  1.09      27
##  4 Adelie    Dream     male          4046.    331.  0.0817    21.1  1.68      28
##  5 Adelie    Torgersen female        3396.    259.  0.0763    18.0  1.48      24
##  6 Adelie    Torgersen male          4035.    372.  0.0923    20.7  1.85      23
##  7 Chinstrap Dream     female        3527.    285.  0.0809    18.4  1.46      34
##  8 Chinstrap Dream     male          3939.    362.  0.0919    19.7  1.49      34
##  9 Gentoo    Biscoe    female        4680.    282.  0.0602    22.0  1.19      58
## 10 Gentoo    Biscoe    male          5485.    313.  0.0571    24.8  1.35      61
## # … with abbreviated variable names ¹meanBodyMassg, ²sdBodyMassg, ³CVBodyMassg,
## #   ⁴sampleSize
```

We can see that each variable appears in only one column, and was therefore measured only once per row. For example, there is only one column containing measures for mean body mass, so there is only one such measurement for each row. Thus, this is long-format data. Adding more observations to the data set (e.g., more species, islands, sexes, or combinations thereof) means adding more rows (lengthening the data), each with a single set of values for each variable.

pivot_wider

To see how this differs from wide-format data, let's change it to wide format using `pivot_wider()`. This function requires the following arguments:

- data: the data set you're transforming. Can be forwarded by the %>%
- names_from: The column that you want to expand into different columns. Each of these new columns will take its name from one of the unique values in the original column

- values_from: The column containing the values that you want to see for each “name” from the original column used for names_from.

Let’s try taking names from the variable `island` and values from the variable `meanBodyMassg`.

```
# filter() to remove rows where sex is NA,
filter(penguins, !is.na(sex)) %>%
  # mutate() to calculate penguin plumpness index from body mass and flipper length
  mutate(pengPlumpInd = body_mass_g / flipper_length_mm) %>%
  # group data by sex
  group_by(species, island, sex) %>%
  # summarize groups in terms of mean body manss, sd in body mass, mean PPI, and sd
in PPI
  summarize(meanBodyMassg = mean(body_mass_g),
            sdBodyMassg = sd(body_mass_g),
            CVBodyMassg = sdBodyMassg / meanBodyMassg,
            meanPPI = mean(pengPlumpInd),
            sdPPI = sd(pengPlumpInd),
            sampleSize = n()) %>%
  # be sure to select *only* the variables you need in your wider table, else thing
s will get messy.
  select(island, sex, meanBodyMassg) %>%
  # use pivot_wider() to turn values of island into their own columns, with the cor
responding value of mean body mass contained in each.
  pivot_wider(names_from = island, values_from = meanBodyMassg)
```

```
## `summarise()` has grouped output by 'species', 'island'. You can override using
## the `.groups` argument.
## Adding missing grouping variables: `species`
```

```
## # A tibble: 6 × 5
## # Groups:   species [3]
##   species   sex    Biscoe Dream Torgersen
##   <fct>     <fct>   <dbl> <dbl>     <dbl>
## 1 Adelie    female  3369. 3344.     3396.
## 2 Adelie    male    4050  4046.     4035.
## 3 Chinstrap female    NA  3527.       NA
## 4 Chinstrap male      NA  3939.       NA
## 5 Gentoo    female  4680.   NA        NA
## 6 Gentoo    male    5485.   NA        NA
```

Now we can see that “island” is no longer a variable that exists in a single column. Instead, every value of “island” is given its own column, and for each combination of species and sex (row) we simply put the mean body mass for that particular island (if any) in the appropriate column. Each row therefore corresponds to a single combination of species and sex but contains multiple observations of mean body mass (one for each island). Adding more islands to this data set would mean adding columns, or “widening” the data.

pivot_longer()

More often, you will find data in wide format and need to transform it to long format for compatibility with R's functions. Let's see if we can get our data back the way we had it! This requires the following arguments to the `pivot_longer()` function:

- cols: The columns that we want to collapse into a single column. Here, we want all the columns with island names, as these contain the repeated measures of body mass for each combination of species and sex (row). One way to do this is by putting all the column names together with `c()`. Another way would be to exclude all the other columns using `-c(species, sex)`.
- names_to: The name of a new column that will contain names of the columns that you collapsed. We'll call it "island".
- values_to: The name of a new column that will have the values for combination of species, sex, and now island. For our use, this is "meanBodyMassg".

```
# filter() to remove rows where sex is NA,
filter(penguins, !is.na(sex)) %>%
  # mutate() to calculate penguin plumpness index from body mass and flipper length
  mutate(pengPlumpInd = body_mass_g / flipper_length_mm) %>%
  # group data by sex
  group_by(species, island, sex) %>%
  # summarize groups in terms of mean body manss, sd in body mass, mean PPI, and sd
in PPI
  summarize(meanBodyMassg = mean(body_mass_g),
            sdBodyMassg = sd(body_mass_g),
            CVBodyMassg = sdBodyMassg / meanBodyMassg,
            meanPPI = mean(pengPlumpInd),
            sdPPI = sd(pengPlumpInd),
            sampleSize = n()) %>%
  # be sure to select *only* the variables you need in your wider table, else thing
s will get messy.
  select(island, sex, meanBodyMassg) %>%
  # use pivot_wider() to turn values of island into their own columns, with the cor
responding value of mean body mass contained in each.
  pivot_wider(names_from = island, values_from = meanBodyMassg) %>%
  # use pivot_longer() to collapse values for different islands into a single colum
n and create a new column with the body mass measure for
  # each island.
  pivot_longer(cols = c(Biscoe, Dream, Torgersen), names_to = "island", values_to =
"meanBodyMassg")
```

```
## `summarise()` has grouped output by 'species', 'island'. You can override using
## the `.groups` argument.
## Adding missing grouping variables: `species`
```

```
## # A tibble: 18 × 4
## # Groups:   species [3]
##    species   sex    island    meanBodyMassg
##    <fct>     <fct>  <chr>             <dbl>
##  1 Adelie    female Biscoe            3369.
##  2 Adelie    female Dream             3344.
##  3 Adelie    female Torgersen         3396.
##  4 Adelie    male   Biscoe            4050 
##  5 Adelie    male   Dream             4046.
##  6 Adelie    male   Torgersen         4035.
##  7 Chinstrap female Biscoe              NA 
##  8 Chinstrap female Dream             3527.
##  9 Chinstrap female Torgersen           NA 
## 10 Chinstrap male   Biscoe              NA 
## 11 Chinstrap male   Dream             3939.
## 12 Chinstrap male   Torgersen           NA 
## 13 Gentoo    female Biscoe            4680.
## 14 Gentoo    female Dream               NA 
## 15 Gentoo    female Torgersen           NA 
## 16 Gentoo    male   Biscoe            5485.
## 17 Gentoo    male   Dream               NA 
## 18 Gentoo    male   Torgersen           NA
```

Great, now we are back to one column for each variable, and one row for each observation. This is obviously a really silly pipeline at this point, but it illustrates the point.