Data wrangling intro

Contents

1 Data

1.1 Data example

We use data about pengiuns from the R package palmerpenguins

```
pengu <- palmerpenguins::penguins
pengu
```

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

2 Data wrangling

2.1 The tidyverse package

We will use the `tidyverse` package extensively, and load it at the beginning of most sessions:

```
library(tidyverse)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.4     v readr     2.1.5
## v forcats   1.0.0     v stringr   1.5.1
## v ggplot2   3.5.1     v tibble    3.2.1
## v lubridate 1.9.3     v tidyr     1.3.1
## v purrr     1.0.2
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become error
```

2.2 Selecting columns/variables

- To subset columns of data use `select()` (automatically loaded from `dplyr` package by `tidyverse`):

```
bill_data <- select(pengu, bill_length_mm, bill_depth_mm,
                    species, sex, island, year)
```

- This particular subset can be written shorter by (overwriting the object we just created):

```
bill_data <- select(pengu, -flipper_length_mm, -body_mass_g)
```

- Special role of first argument gives rise to this "pipe" (`|>`) syntax:

```
bill_data <- pengu |> select(-flipper_length_mm, -body_mass_g)
```

- We read this as: first take the dataset `pengu` **and then** select all columns except `flipper_length_mm` and `body_mass_g`.
- The resulting dataset doesn't have flipper length and body mass:

```
names(bill_data)
```

```
## [1] "species"        "island"         "bill_length_mm" "bill_depth_mm"
## [5] "sex"            "year"
```

2.3 Renaming columns/variables

Either via `select()` (which can also use column position), but then you only keep the selected ones:

```
bill_data_small <- bill_data |>
  select(len = bill_length_mm, depth = bill_depth_mm, pengu_type = 1)
bill_data_small
```

```
## # A tibble: 344 x 3
##      len depth pengu_type
##    <dbl> <dbl> <fct>
##  1  39.1  18.7 Adelie
##  2  39.5  17.4 Adelie
##  3  40.3  18   Adelie
##  4  NA    NA   Adelie
##  5  36.7  19.3 Adelie
##  6  39.3  20.6 Adelie
##  7  38.9  17.8 Adelie
##  8  39.2  19.6 Adelie
##  9  34.1  18.1 Adelie
## 10  42    20.2 Adelie
## # i 334 more rows
```

Or via `rename()` where you keep everything in place and just rename the relevant columns:

```
bill_data |> rename(len = bill_length_mm, depth = bill_depth_mm, pengu_type = 1)
```

```
## # A tibble: 344 x 6
##    pengu_type island      len depth sex     year
##    <fct>      <fct>     <dbl> <dbl> <fct>  <int>
##  1 Adelie     Torgersen  39.1  18.7 male    2007
##  2 Adelie     Torgersen  39.5  17.4 female  2007
##  3 Adelie     Torgersen  40.3  18   female  2007
##  4 Adelie     Torgersen  NA    NA   <NA>    2007
##  5 Adelie     Torgersen  36.7  19.3 female  2007
##  6 Adelie     Torgersen  39.3  20.6 male    2007
##  7 Adelie     Torgersen  38.9  17.8 female  2007
##  8 Adelie     Torgersen  39.2  19.6 male    2007
##  9 Adelie     Torgersen  34.1  18.1 <NA>    2007
## 10 Adelie     Torgersen  42    20.2 <NA>    2007
## # i 334 more rows
```

2.4 Filtering rows/cases/observations

- We use `filter()` to subset rows/cases. E.g. all penguins from Biscoe islands:

```
pengu |> filter(island == "Biscoe")
```

```
## # A tibble: 168 x 8
##    species island bill_length_mm bill_depth_mm flipper_length_mm body_mass_g
##    <fct>   <fct>           <dbl>         <dbl>             <int>       <int>
##  1 Adelie  Biscoe           37.8          18.3               174        3400
##  2 Adelie  Biscoe           37.7          18.7               180        3600
##  3 Adelie  Biscoe           35.9          19.2               189        3800
##  4 Adelie  Biscoe           38.2          18.1               185        3950
##  5 Adelie  Biscoe           38.8          17.2               180        3800
##  6 Adelie  Biscoe           35.3          18.9               187        3800
##  7 Adelie  Biscoe           40.6          18.6               183        3550
##  8 Adelie  Biscoe           40.5          17.9               187        3200
##  9 Adelie  Biscoe           37.9          18.6               172        3150
## 10 Adelie  Biscoe           40.5          18.9               180        3950
## # i 158 more rows
## # i 2 more variables: sex <fct>, year <int>
```

- All male Gentoo penguins with over 220 mm flippers:

```
pengu |> filter(sex == "male") |>
  filter(species == "Gentoo") |>
  filter(flipper_length_mm>220)
```

```
## # A tibble: 34 x 8
##    species island bill_length_mm bill_depth_mm flipper_length_mm body_mass_g
##    <fct>   <fct>           <dbl>         <dbl>             <int>       <int>
##  1 Gentoo  Biscoe           50            16.3               230        5700
##  2 Gentoo  Biscoe           49.2          15.2               221        6300
##  3 Gentoo  Biscoe           48.7          15.1               222        5350
##  4 Gentoo  Biscoe           47.3          15.3               222        5250
##  5 Gentoo  Biscoe           59.6          17                 230        6050
##  6 Gentoo  Biscoe           49.6          16                 225        5700
##  7 Gentoo  Biscoe           50.5          15.9               222        5550
##  8 Gentoo  Biscoe           50.5          15.9               225        5400
##  9 Gentoo  Biscoe           50.1          15                 225        5000
## 10 Gentoo  Biscoe           50.4          15.3               224        5550
## # i 24 more rows
## # i 2 more variables: sex <fct>, year <int>
```

- This could also have been done with a single `filter()` command (output not shown):

```
pengu |> filter(sex == "male" & species == "Gentoo" & flipper_length_mm>220)
```

- All penguins of species Gentoo or Adelie:

```
pengu |> filter(species == "Gentoo" | species == "Adelie")
```

```
## # A tibble: 276 x 8
##    species island    bill_length_mm bill_depth_mm flipper_length_mm body_mass_g
##    <fct>   <fct>              <dbl>         <dbl>             <int>       <int>
##  1 Adelie  Torgersen           39.1          18.7               181        3750
##  2 Adelie  Torgersen           39.5          17.4               186        3800
##  3 Adelie  Torgersen           40.3          18                 195        3250
##  4 Adelie  Torgersen           NA            NA                  NA          NA
##  5 Adelie  Torgersen           36.7          19.3               193        3450
##  6 Adelie  Torgersen           39.3          20.6               190        3650
##  7 Adelie  Torgersen           38.9          17.8               181        3625
##  8 Adelie  Torgersen           39.2          19.6               195        4675
##  9 Adelie  Torgersen           34.1          18.1               193        3475
## 10 Adelie  Torgersen           42            20.2               190        4250
## # i 266 more rows
## # i 2 more variables: sex <fct>, year <int>
```

- This would be the same as penguins which are not Chinstrap (output not shown):

```
pengu |> filter(species != "Chinstrap")
```

2.5 Arranging rows/cases/observations

- We use `arrange()` to arrange the order of the rows/cases:

```
pengu |> filter(sex == "female") |> arrange(body_mass_g)
```

```
## # A tibble: 165 x 8
##    species   island   bill_length_mm bill_depth_mm flipper_length_mm body_mass_g
##    <fct>     <fct>             <dbl>         <dbl>             <int>       <int>
##  1 Chinstrap Dream              46.9          16.6               192        2700
##  2 Adelie    Biscoe             36.5          16.6               181        2850
##  3 Adelie    Biscoe             36.4          17.1               184        2850
##  4 Adelie    Biscoe             34.5          18.1               187        2900
##  5 Adelie    Dream              33.1          16.1               178        2900
##  6 Adelie    Torgers~           38.6          17                  188         2900
##  7 Chinstrap Dream              43.2          16.6                187         2900
##  8 Adelie    Biscoe             37.9          18.6                193         2925
##  9 Adelie    Dream              37            16.9                185         3000
## 10 Adelie    Dream              37.3          16.8                192         3000
## # i 155 more rows
## # i 2 more variables: sex <fct>, year <int>
```

- Use `arrange(desc())` for descending values:

```
pengu |> filter(sex == "female") |> arrange(desc(body_mass_g))
```

```
## # A tibble: 165 x 8
##    species island bill_length_mm bill_depth_mm flipper_length_mm body_mass_g
##    <fct>   <fct>           <dbl>         <dbl>             <int>       <int>
##  1 Gentoo  Biscoe           46.5          14.8               217        5200
##  2 Gentoo  Biscoe           45.2          14.8               212        5200
##  3 Gentoo  Biscoe           49.1          14.8               220        5150
##  4 Gentoo  Biscoe           44.9          13.3               213        5100
##  5 Gentoo  Biscoe           45.1          14.5               207        5050
##  6 Gentoo  Biscoe           45.1          14.5               215        5000
##  7 Gentoo  Biscoe           42.9          13.1               215        5000
##  8 Gentoo  Biscoe           50.5          15.2               216        5000
##  9 Gentoo  Biscoe           47.2          15.5               215        4975
## 10 Gentoo  Biscoe           42.6          13.7               213        4950
## # i 155 more rows
## # i 2 more variables: sex <fct>, year <int>
```

2.6 Extracting rows/cases/observations

- The `arrange()` command reorders the rows, so the ones we are interested in are in the top or bottom. If we want to extract e.g. the top 5 heaviest female penguins we use `slice_max()`:

```
pengu |> filter(sex == "female") |> slice_max(body_mass_g, n = 5)
```

```
## # A tibble: 5 x 8
##   species island bill_length_mm bill_depth_mm flipper_length_mm body_mass_g
##   <fct>   <fct>           <dbl>         <dbl>             <int>       <int>
## 1 Gentoo  Biscoe           46.5          14.8               217        5200
## 2 Gentoo  Biscoe           45.2          14.8               212        5200
## 3 Gentoo  Biscoe           49.1          14.8               220        5150
## 4 Gentoo  Biscoe           44.9          13.3               213        5100
## 5 Gentoo  Biscoe           45.1          14.5               207        5050
## # i 2 more variables: sex <fct>, year <int>
```

- To extract rows by their rownumber we simply use `slice()` (note the five female penguins below are taken from the original order of the dataset and not according to some ordering variable):

```
pengu |> filter(sex == "female") |> slice(c(1, 3, 5, 7, 9))
```

```
## # A tibble: 5 x 8
##   species island    bill_length_mm bill_depth_mm flipper_length_mm body_mass_g
##   <fct>   <fct>              <dbl>         <dbl>             <int>       <int>
## 1 Adelie  Torgersen           39.5          17.4               186        3800
## 2 Adelie  Torgersen           36.7          19.3               193        3450
## 3 Adelie  Torgersen           41.1          17.6               182        3200
## 4 Adelie  Torgersen           38.7          19                 195        3450
## 5 Adelie  Biscoe              37.8          18.3               174        3400
## # i 2 more variables: sex <fct>, year <int>
```

- Other slicing command are: `slice_min()`, `slice_sample()`, `slice_head()` and `slice_tail()`.

2.7 Extending (mutating) data with new variables

```
bill_data_ratio <- bill_data_small |> mutate(ratio = len/depth)
bill_data_ratio
```

```
## # A tibble: 344 x 4
##      len depth pengu_type ratio
##    <dbl> <dbl> <fct>      <dbl>
##  1  39.1  18.7 Adelie      2.09
##  2  39.5  17.4 Adelie      2.27
##  3  40.3  18   Adelie      2.24
##  4  NA    NA   Adelie     NA
##  5  36.7  19.3 Adelie      1.90
##  6  39.3  20.6 Adelie      1.91
##  7  38.9  17.8 Adelie      2.19
##  8  39.2  19.6 Adelie      2
##  9  34.1  18.1 Adelie      1.88
## 10  42    20.2 Adelie      2.08
## # i 334 more rows
```

2.8 Summarising data

```
bill_data_small |> summarise(avg_len = mean(len), avg_depth = mean(depth),
                              sd_len = sd(len), sd_depth = sd(depth),
                              cor_len_depth = cor(len, depth))
```

```
## # A tibble: 1 x 5
##   avg_len avg_depth sd_len sd_depth cor_len_depth
##     <dbl>     <dbl>  <dbl>    <dbl>         <dbl>
## 1      NA        NA     NA       NA            NA
```

By default R cannot calculate summaries of data with `NA`, so we can either tell each function to remove `NA` or drop all `NA`s from the data (we do the latter):

```
bill_data_small |> drop_na() |>
  summarise(avg_len = mean(len), avg_depth = mean(depth),
            sd_len = sd(len), sd_depth = sd(depth),
            cor_len_depth = cor(len, depth))
```

```
## # A tibble: 1 x 5
##   avg_len avg_depth sd_len sd_depth cor_len_depth
##     <dbl>     <dbl>  <dbl>    <dbl>         <dbl>
## 1    43.9      17.2   5.46     1.97        -0.235
```

2.9 Summarising data by group

The `group_by()` function simply annotates the data as being grouped:

```
bill_grp <- bill_data_small |> group_by(pengu_type)
bill_grp
```

```
## # A tibble: 344 x 3
## # Groups:   pengu_type [3]
##      len depth pengu_type
##    <dbl> <dbl> <fct>
##  1  39.1  18.7 Adelie
##  2  39.5  17.4 Adelie
##  3  40.3  18   Adelie
##  4  NA    NA   Adelie
##  5  36.7  19.3 Adelie
##  6  39.3  20.6 Adelie
##  7  38.9  17.8 Adelie
##  8  39.2  19.6 Adelie
##  9  34.1  18.1 Adelie
## 10  42    20.2 Adelie
## # i 334 more rows
```

Then summaries are done for each group:

```
bill_data_small |> na.omit() |> group_by(pengu_type) |>
  summarise(avg_len = mean(len), avg_depth = mean(depth),
            sd_len = sd(len), sd_depth = sd(depth),
            cor_len_depth = cor(len, depth))
```

```
## # A tibble: 3 x 6
##   pengu_type avg_len avg_depth sd_len sd_depth cor_len_depth
##   <fct>        <dbl>     <dbl>  <dbl>    <dbl>         <dbl>
## 1 Adelie        38.8      18.3   2.66    1.22          0.391
## 2 Chinstrap     48.8      18.4   3.34    1.14          0.654
## 3 Gentoo        47.5      15.0   3.08    0.981         0.643
```

2.10 Counting

Counting is a very common summary of data and it has a separate function/verb `count` which is effectively just shorthand for `group_by()` + `summarise(n = n())`:

```
pengu |> count(species)
```

```
## # A tibble: 3 x 2
##   species       n
##   <fct>     <int>
## 1 Adelie      152
## 2 Chinstrap    68
## 3 Gentoo      124
```

```
pengu |> group_by(species) |> summarise(n = n())
```

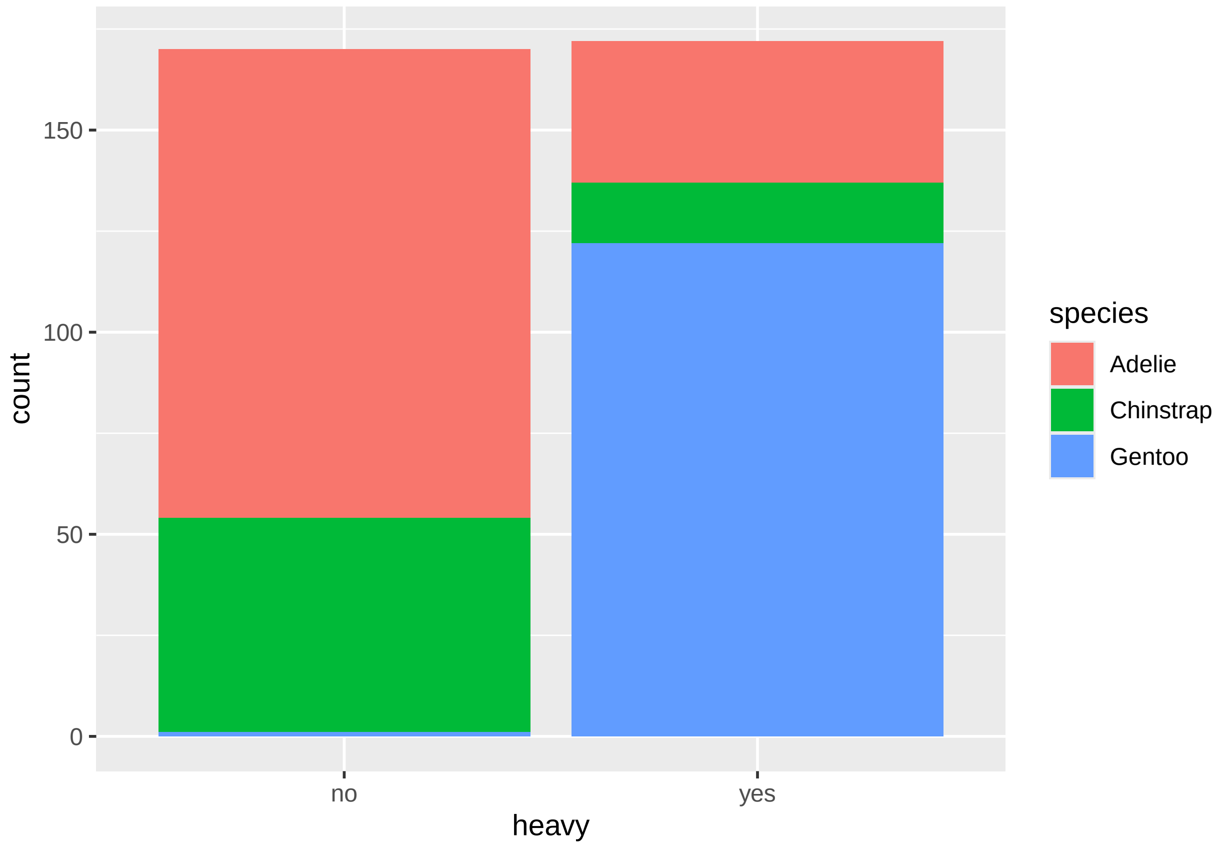
```
## # A tibble: 3 x 2
##   species       n
##   <fct>     <int>
## 1 Adelie      152
## 2 Chinstrap    68
## 3 Gentoo      124
```

2.11 Bar graphs

To make a bar graph from the “raw” data where we didn’t count things first use `geom_bar()`. Before making the graph we make a new variable **heavy** indicating whether the penguin mass is above 4000 g (the command `ifelse()` checks the condition given as first input argument and if the condition is `TRUE` it returns the one option and if it is false it returns the other (these options are given as second and third input arguments, respectively):

```
pengu3 <- pengu |>
  drop_na(body_mass_g) |>
  mutate(heavy = ifelse(body_mass_g > 4000, "yes", "no"))
```

```
ggplot(pengu3, aes(x = heavy, fill = species)) +
  geom_bar()
```

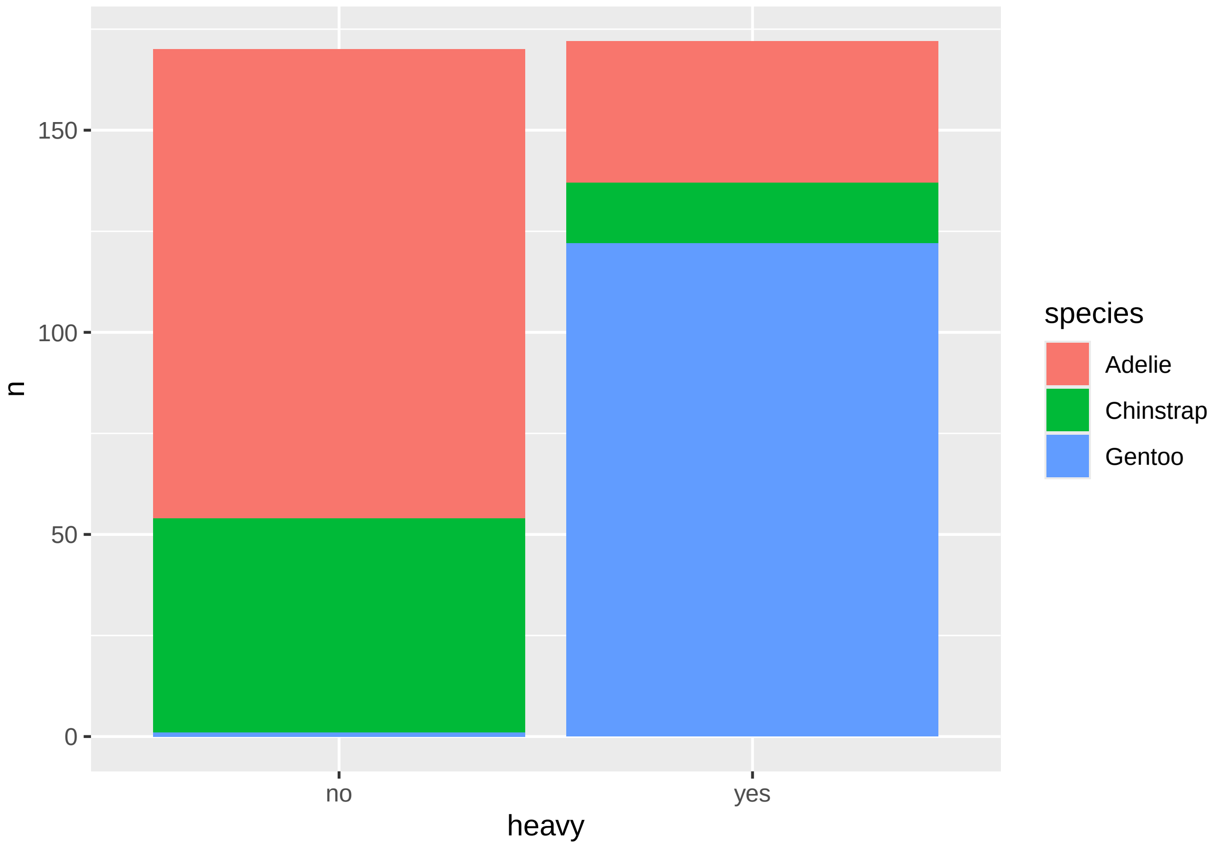


To make a bar graph from the aggregated data where we have the counts we want to plot use `geom_col()` (for column):

```
pengu3count <- pengu3 |>
  count(heavy, species)
pengu3count
```

```
## # A tibble: 6 x 3
##   heavy species       n
##   <chr> <fct>     <int>
## 1 no    Adelie      116
## 2 no    Chinstrap    53
## 3 no    Gentoo        1
## 4 yes   Adelie       35
## 5 yes   Chinstrap    15
## 6 yes   Gentoo      122
```

```r
ggplot(pengu3count, aes(x = heavy, y = n, fill = species)) +
  geom_col()
```



2.12 Exercises

- First work on the qmd exercise on Moodle.
- If you have extra time confront Chapter 3 of R4DS and work on (you will need to read about the NYC flights data to understand the exercises):
  - Exercises 3.2.5: https://r4ds.hadley.nz/data-transform#exercises
  - Exercises 3.3.5: https://r4ds.hadley.nz/data-transform#exercises-1
  - Exercises 3.5.7: https://r4ds.hadley.nz/data-transform#exercises-2