Datenvisualisierung 2

Darstellung der zusammenfassenden Statistik

Daniela Palleschi

2023-06-19

Inhaltsverzeichnis

Heutige Ziele

This week we will...

-

0.1 Lust auf mehr?

Section 2.5 Visualising relationsips in Wickham et al. (o. J.)

Ch. 4 Representing summary statistics in (**nordmann_data_2022?**)

Wiederholung

Letzte Woche haben wir…

- Maße der zentralen Tendenz (neu) kennengelernt
- (Wieder-)Erlernen von Streuungsmaßen
- gelernt, wie man die Funktion `summarise()` von `dplyr` benutzt
- gelernt, wie man Zusammenfassungen "nach" Gruppen erstellt

Heutige Ziele

Heute haben wir…

-

1 Einrichtung

```
pacman::p_load(tidyverse,
               here,
               palmerpenguins,
               ggthemes,
               patchwork,
               plotly)
```

```
df_penguins <- palmerpenguins::penguins
```

2 Visualising distributions

- we did this in week 3 using
  - histograms (1 numerical variable)
  - density plots (1 numerical variable)
  - scatterplots (2 numerical variables)
  - barplots (categorical variables)

2.1 Violin plots

- we can also use violin plots, which are pretty trendy at the moment
  - basically a double-sided/mirrored density plot

```
# fig_hist <-
  df_penguins %>%
  ggplot(aes(x = species, y = body_mass_g, fill = species)) +
  geom_violin(alpha = .2) +
  labs(
       x = "Body mass (g)",
       y = "Count",
    fill = "Species") +
  scale_color_colorblind() +
  scale_fill_colorblind() +
  theme_minimal()
```

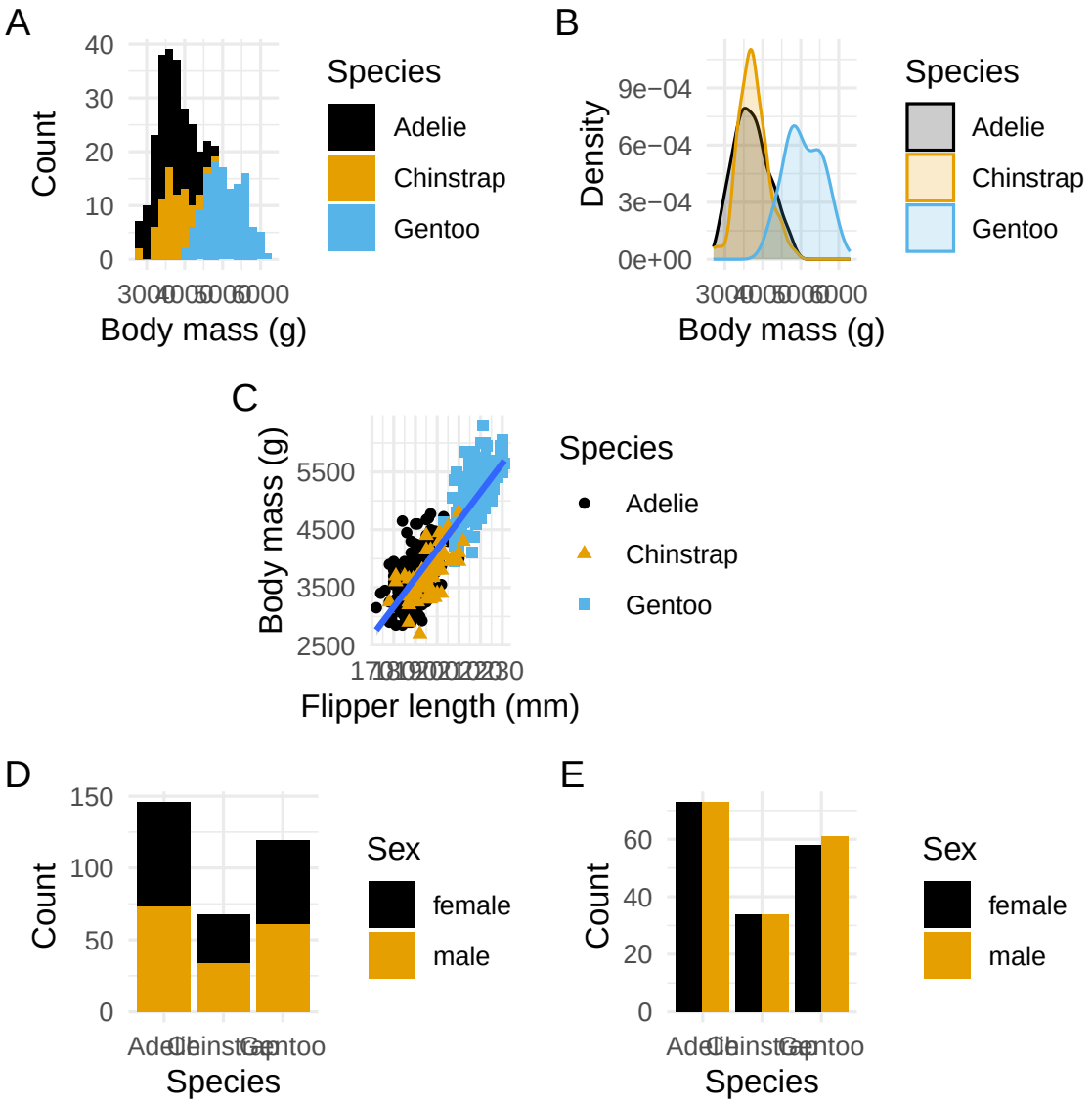


Abbildung 1: Different plots types to visualise distribution of raw data: histogram (A), density plot (B), scatterplot (C), stacked barplot (D), and dodged barplot (E)

3 Visualising 3 or more variables

- as we know, we can incorporate more variables by mapping them onto aesthetics (e.g., `colour`, `fill`, or `shape`)
- Abbildung 1 did this by using `colour` (all plots) and `shape` (scatterplot) to visualise `species` or `sex` in addition to what was mapped along the x- and y-axes

---

- adding too many variables into a single plot can make it diffcult to read
- for example, how many variables are mapped in the following code?

```
df_penguins %>%
  drop_na(sex) %>%
  ggplot(aes(x = flipper_length_mm, y = body_mass_g)) +
  geom_point(aes(color = species, shape = island)) +
  labs(
    x = "Flipper length (mm)",
    y = "Body mass (g)",
    color = "Species",
    shape = "Island"
```

```
  ) +
  scale_color_colorblind() +
  theme_minimal()
```

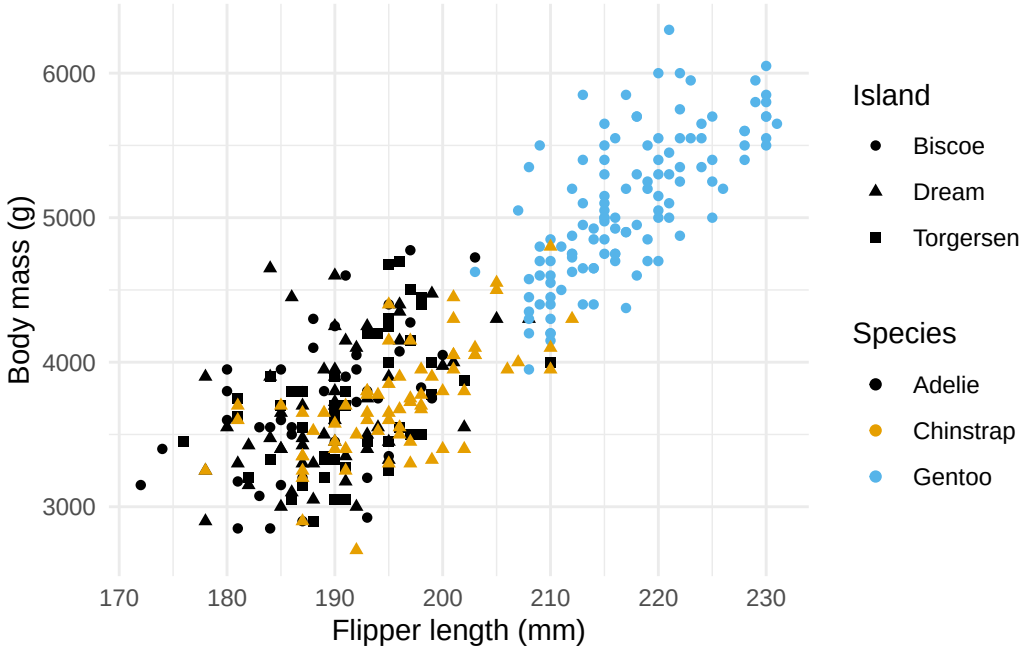


Abbildung 2: A cluttered scatterplot with 4 variables

- four: `flipper_length_mm` (x-axis), `body_mass_g` (y-axis), `species` (color), `island` (shape)
- this is a bit visually cluttered!

3.1 `facet_wrap()`

- a nice way to split our data into different plots is by using the `facet_wrap()`
  - can be used to split one cluttered plot into separate plots based on a categorical variable

---

- let's try using `facet_wrap()` to divide Abbildung 2 into three plots, by `island`

```r
df_penguins %>%
  drop_na(sex) %>%
  ggplot(aes(x = flipper_length_mm, y = body_mass_g)) +
  facet_wrap(~island) +
  geom_point(aes(color = species, shape = species)) +
  labs(
    x = "Flipper length (mm)",
    y = "Body mass (g)",
    color = "Species",
    shape = "Island"
  ) +
  scale_color_colorblind() +
  theme_bw()
```

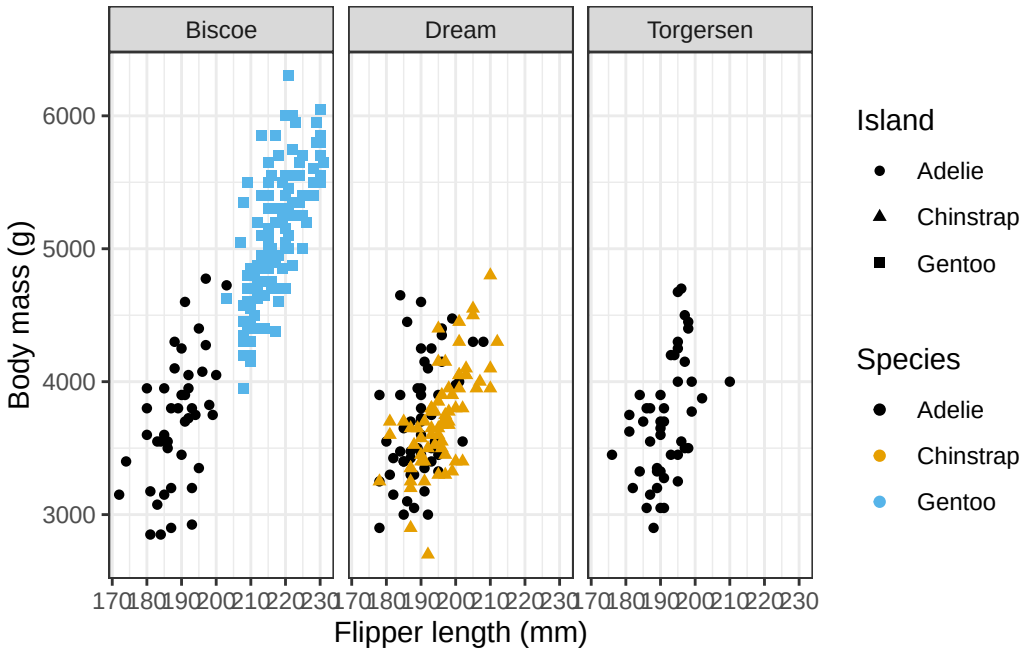


Abbildung 3: A cluttered scatterplot with 4 variables

`facet_grid()`

`facet_wrap()` is related to `facet_grid()`, which can take two categorical variables, one in columns and one in rows. The argument for `facet_grid()` is an equation: `row~column`. So, if we add `facet_grid(sex~island)` to our plot, we should see the data in plots grouped by `sex` in rows (one row for `female`, one row for `male`) and `island` in columns (one column for each `island`)

```
df_penguins %>%
  drop_na(sex) %>%
  ggplot(aes(x = flipper_length_mm, y = body_mass_g)) +
  facet_grid(sex~island) +
  geom_point(aes(color = species, shape = species)) +
  labs(
    x = "Flipper length (mm)",
    y = "Body mass (g)",
    color = "Species",
    shape = "Species"
  ) +
  scale_color_colorblind() +
  theme_bw()
```

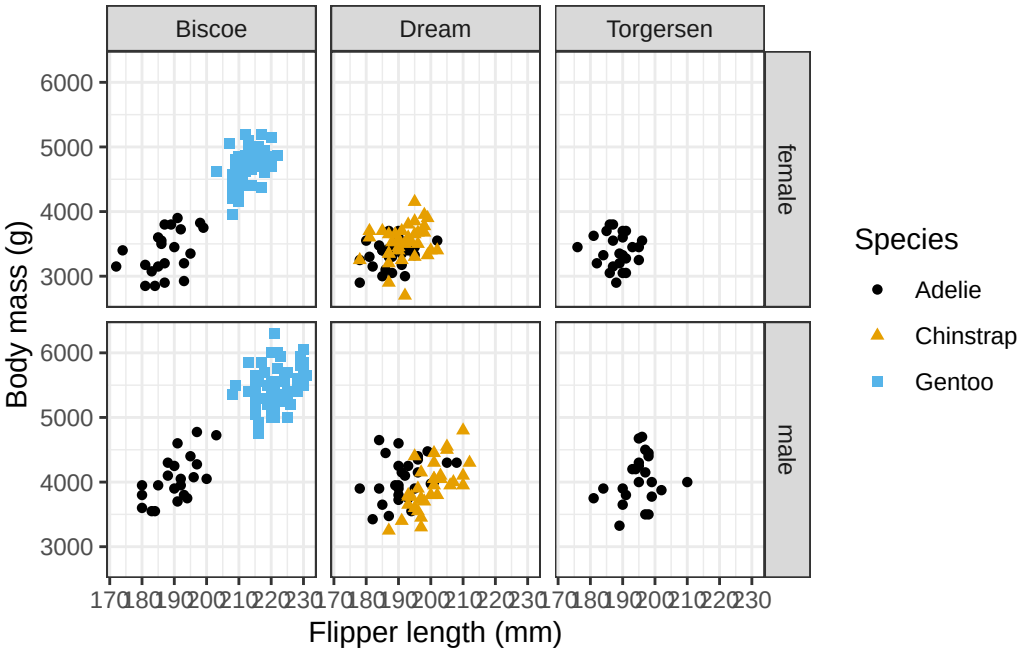


Abbildung 4: facet_grid(sex~species)

4 Representing summary statistics

- last week we talked about summary statistics
- now we will learn how to visualise some of these statistics

4.1 Boxplot

- boxplots visualise
  - **thick black line**: the median, also called Q2 (2nd quartile; the middle value above/below which 50% of the data lie)
  - **box**: the interquartile range (IQR; the range of values between the middle 50% of the data lie), with the boundaries:
    - Q1 (1st quartile, below which 25% of the data lie)
    - Q3 (3rd quartile, above which 25% of the data lie)
  - **whiskers**: 1.5*IQR from Q1 (lower whisker) or Q3 (upper whisker)
  - **dots**: outliers (outside the IQR)

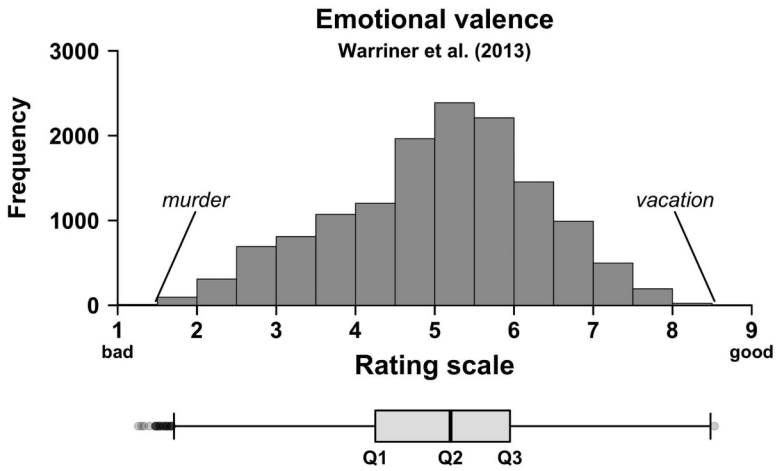


Figure 3.4. A histogram of the emotional valence rating data

Abbildung 5: Image source: (**winter_statistics_2019?**) (all rights reserved)

Or, explained another way:

- we can produce boxplots with `geom_boxplot()`

```
df_penguins %>%
  drop_na(sex) %>%
  ggplot(aes(x = species, y = body_mass_g)) +
  geom_boxplot() +
  theme_bw()
```

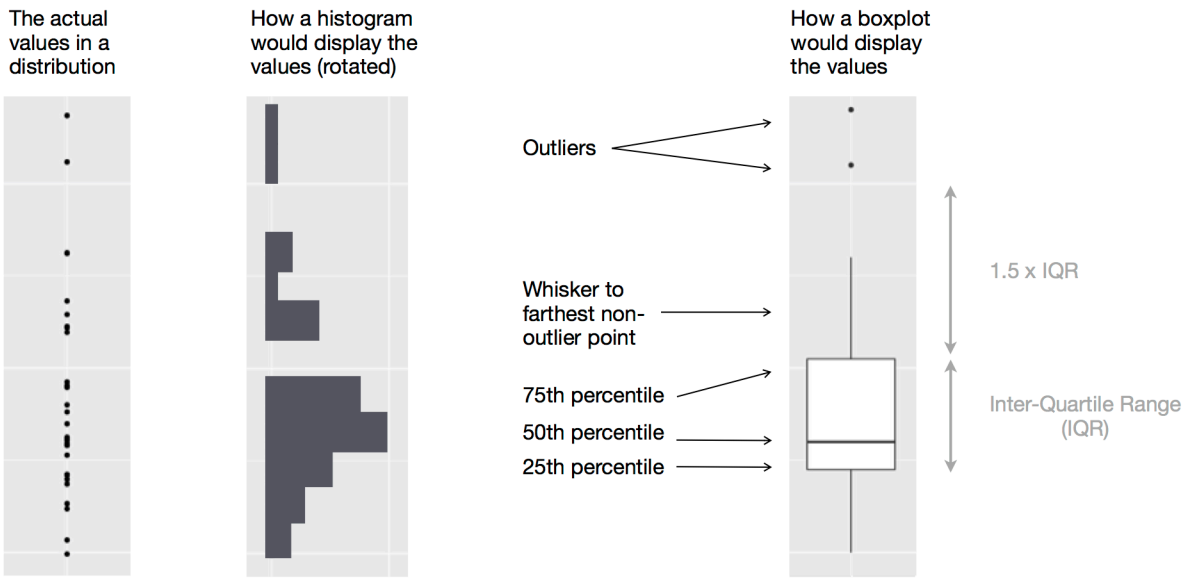
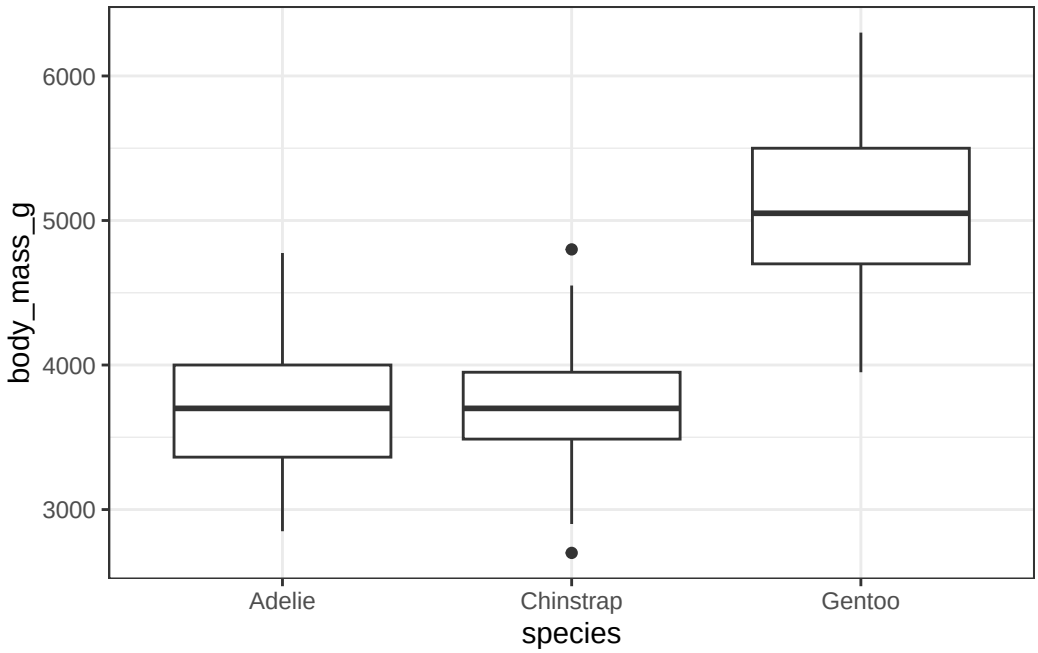


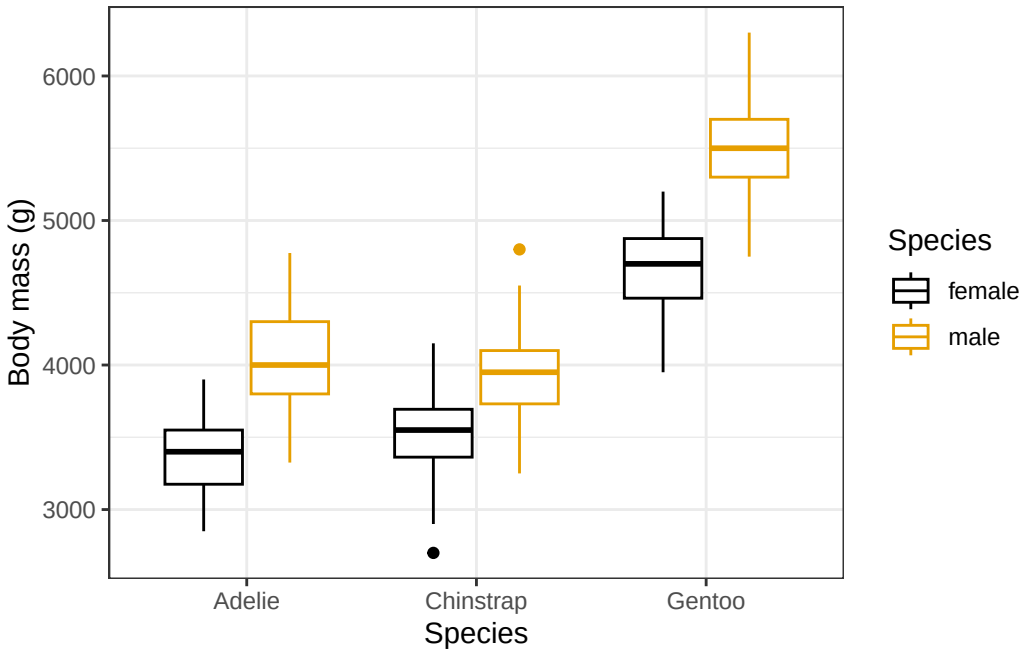
Abbildung 6: Image source: Wickham et al. (o. J.) (all rights reserved)



4.2 Grouped boxplot

- like a bargraph, we can produced grouped boxplots to visualise more variables
  - just add a `colour` or `fill` aesthetic

```
# fig_boxplot <-
df_penguins %>%
  drop_na(sex) %>%
  ggplot(aes(x = species, y = body_mass_g, colour = sex)) +
  geom_boxplot() +
  labs(
    x = "Species",
    y = "Body mass (g)",
    color = "Species",
    shape = "Species"
  ) +
  scale_colour_colorblind() +
  theme_bw()
```



4.2.1 Errorbar plots

- we can visualise the mean and standard deviation with errorbar plots
  - sometimes called interaction plots
- these plots have 2 parts:
  - the mean, visualised with `geom_point()`
  - the sd, visualised with `geom_errorbar()`

- we need to first calculate the mean and standard deviation, grouped by whatever variables we want to visualise
  - let's stick with `body_mass_g` and `species`

```
df_penguins %>%
  drop_na(body_mass_g, sex) %>%
  summarise(mean = mean(body_mass_g),
            sd = sd(body_mass_g),
            N = n(),
            .by = c(species,sex)) %>%
```

species	sex	mean	sd	N
Adelie	female	3368.836	269.3801	73
Adelie	male	4043.493	346.8116	73
Chinstrap	female	3527.206	285.3339	34
Chinstrap	male	3938.971	362.1376	34
Gentoo	female	4679.741	281.5783	58
Gentoo	male	5484.836	313.1586	61

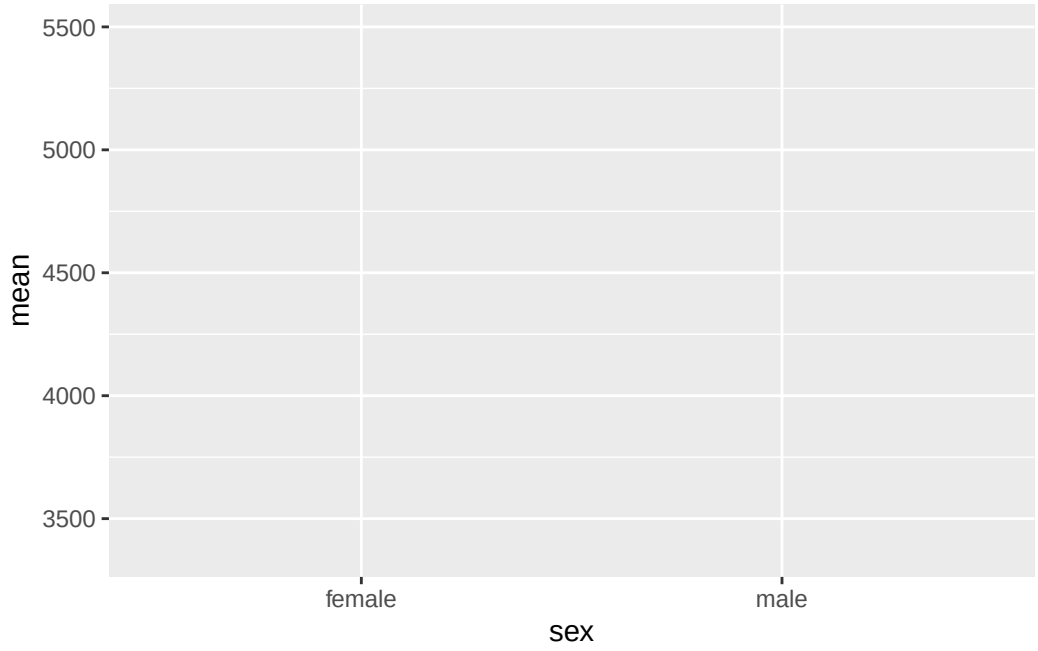
```
  arrange(species, sex) %>%
  knitr::kable() %>%
  kableExtra::kable_styling(font_size = 20)
```

- we have to feed this summary into `ggplot2`
  - ***without*** the table formatting from `knitr` and `kableExtra`!!!!
  - we can do this by saving the summary as a new object, or by adding a pipe after the code

4.2.1.1 New object

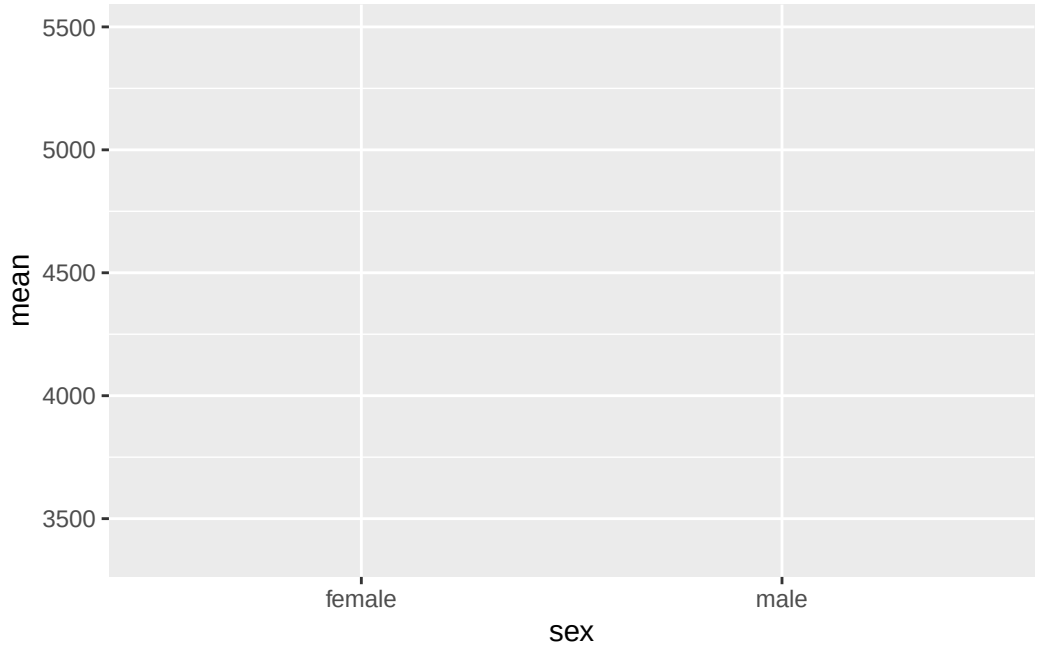
```
sum_penguins <- df_penguins %>%
  drop_na(body_mass_g, sex) %>%
  summarise(mean = mean(body_mass_g),
            sd = sd(body_mass_g),
            upper = mean+sd,
            lower = mean-sd,
            N = n(),
            .by = c(species,sex)) %>%
  arrange(species, sex)
```

```
sum_penguins %>%
  ggplot(aes(x = sex, y = mean, colour = species))
```



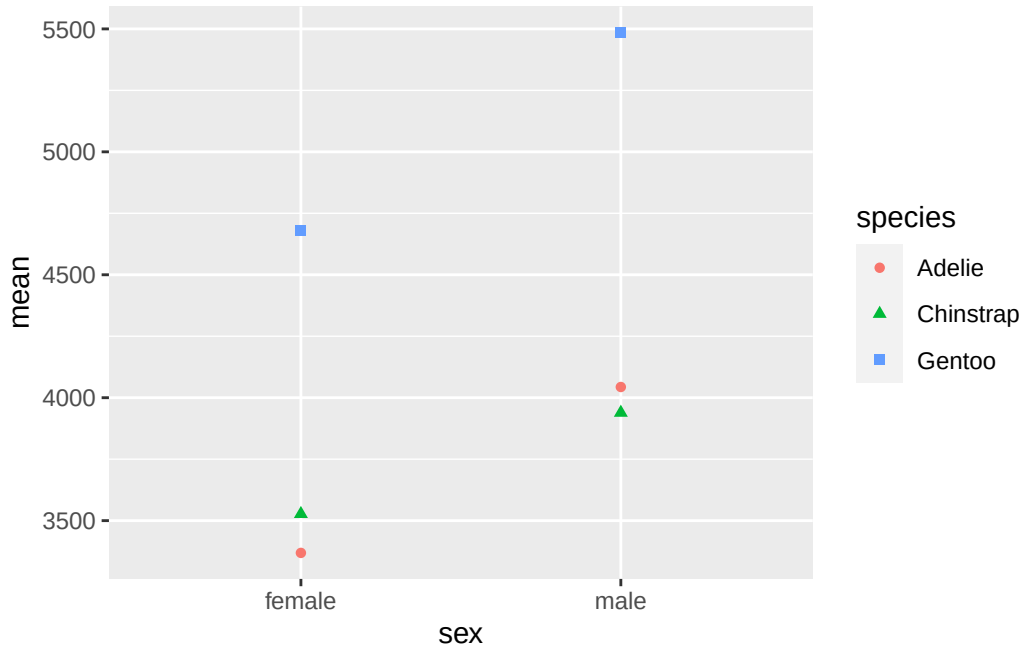
4.2.1.2 With a pipe

```
df_penguins %>%
  drop_na(body_mass_g, sex) %>%
  summarise(mean = mean(body_mass_g),
            sd = sd(body_mass_g),
            upper = mean+sd,
            lower = mean-sd,
            N = n(),
            .by = c(species,sex)) %>%
  arrange(species, sex) %>%
  ggplot(aes(x = sex, y = mean, colour = species))
```



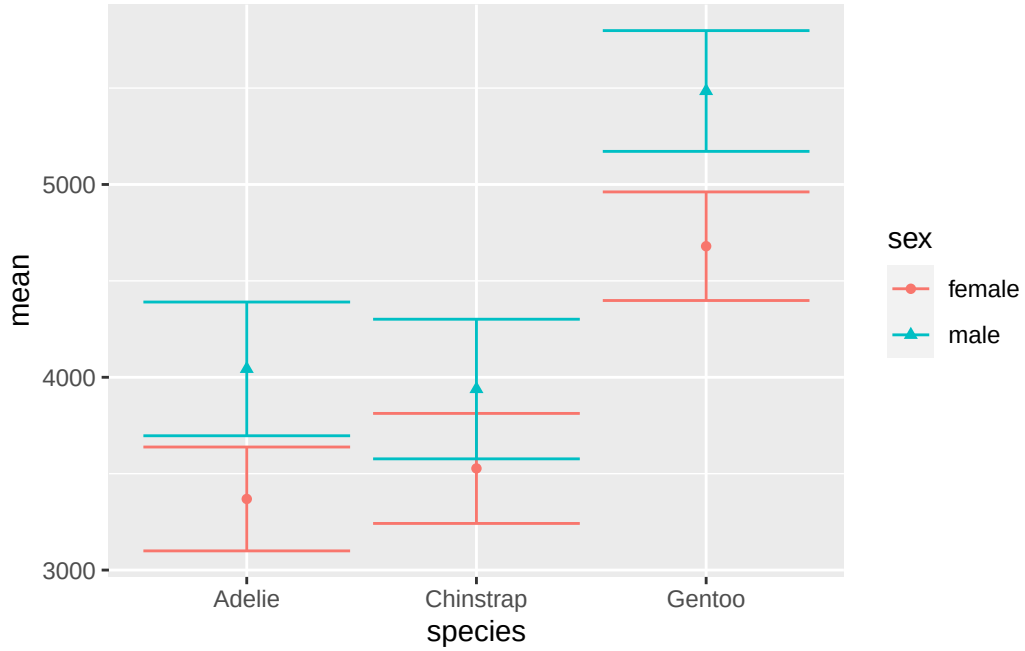
4.2.1.3 Adding mean

```
sum_penguins %>%
  ggplot(aes(x = sex, y = mean,
             colour = species, shape = species)) +
  geom_point()
```



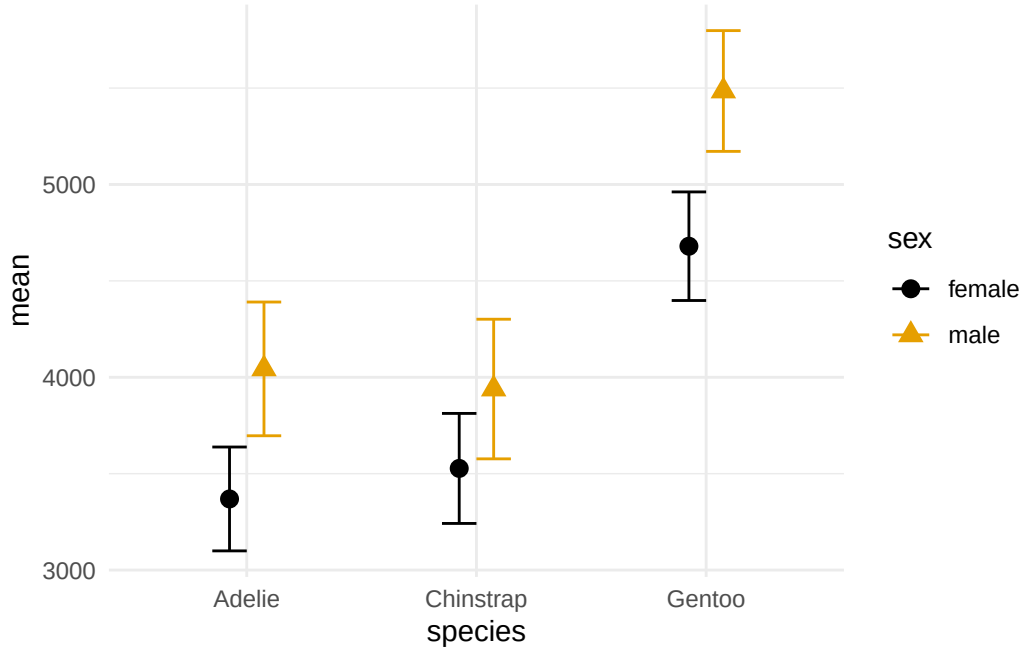
4.2.1.4 Adding errorbars

```
sum_penguins %>%
  ggplot(aes(x = species, y = mean,
             colour = sex, shape = sex)) +
  geom_point() +
  geom_errorbar(aes(ymin=lower,ymax=upper))
```



4.2.2 Customising

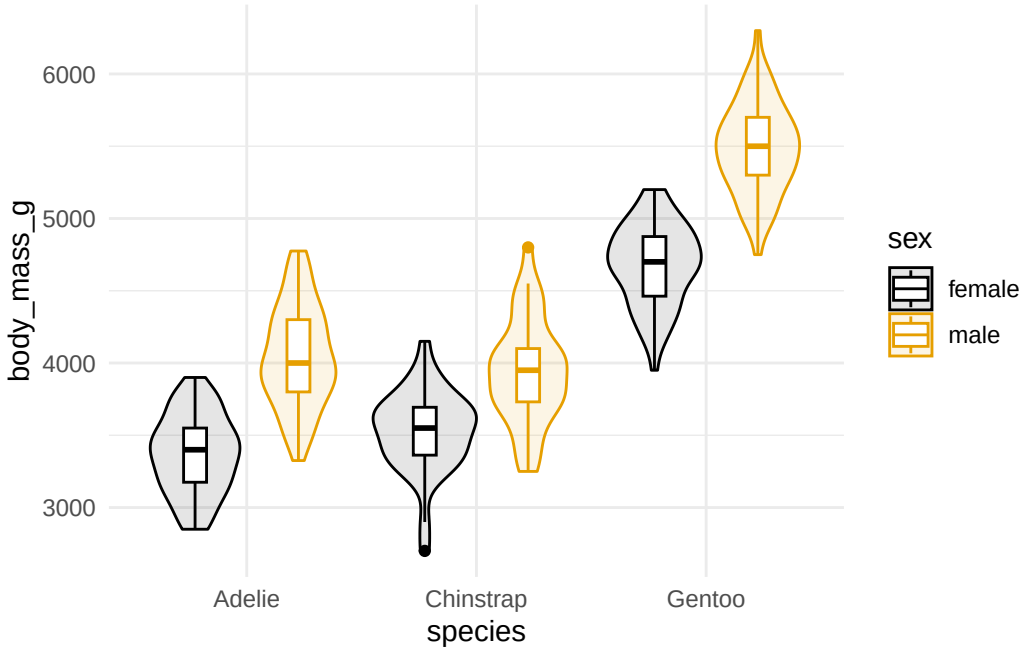
```
sum_penguins %>%
  ggplot(aes(x = species, y = mean,
             colour = sex, shape = sex)) +
  geom_point(position = position_dodge(0.3),
             size = 3) +
  geom_errorbar(aes(ymin=lower,ymax=upper),
                position = position_dodge(0.3),
                width = .3) +
  scale_colour_colorblind() +
  theme_minimal()
```



5 Multi-part plots

- we can combine various types of plots to summarise our data but also provide the distribution
  - this is easiest when they use the same underlying data

```
df_penguins %>%
  drop_na(sex) %>%
  ggplot(aes(x = species, y = body_mass_g,
             colour = sex, shape = sex)) +
  geom_violin(aes(fill = sex), alpha = .1, position = position_dodge(.9)) +
  geom_boxplot(width = .2, position = position_dodge(.9)) +
  scale_colour_colorblind() +
  scale_fill_colorblind() +
  theme_minimal()
```



Session Info

Hergestellt mit R version 4.3.0 (2023-04-21) (Already Tomorrow) und RStudioversion 2023.3.0.386 (Cherry Blossom).

```
sessionInfo()
```

```
R version 4.3.0 (2023-04-21)
Platform: aarch64-apple-darwin20 (64-bit)
Running under: macOS Ventura 13.2.1

Matrix products: default
BLAS:   /Library/Frameworks/R.framework/Versions/4.3-arm64/Resources/lib/libRblas.0.dylib
LAPACK: /Library/Frameworks/R.framework/Versions/4.3-arm64/Resources/lib/libRlapack.dylib;

locale:
[1] en_US.UTF-8/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8

time zone: Europe/Berlin
tzcode source: internal
```

```
attached base packages:
[1] stats     graphics  grDevices utils     datasets  methods   base

other attached packages:
 [1] magick_2.7.4         plotly_4.10.2        patchwork_1.1.2
 [4] ggthemes_4.2.4       palmerpenguins_0.1.1 here_1.0.1
 [7] lubridate_1.9.2      forcats_1.0.0        stringr_1.5.0
[10] dplyr_1.1.2          purrr_1.0.1          readr_2.1.4
[13] tidyr_1.3.0          tibble_3.2.1         ggplot2_3.4.2
[16] tidyverse_2.0.0

loaded via a namespace (and not attached):
 [1] gtable_0.3.3          xfun_0.39             htmlwidgets_1.6.2
 [4] lattice_0.21-8        tzdb_0.4.0            vctrs_0.6.2
 [7] tools_4.3.0           generics_0.1.3        fansi_1.0.4
[10] pacman_0.5.1          pkgconfig_2.0.3       Matrix_1.5-4
[13] data.table_1.14.8     webshot_0.5.4         lifecycle_1.0.3
[16] compiler_4.3.0        farver_2.1.1          munsell_0.5.0
[19] htmltools_0.5.5       yaml_2.3.7            lazyeval_0.2.2
[22] pillar_1.9.0          nlme_3.1-162          tidyselect_1.2.0
[25] rvest_1.0.3           digest_0.6.31         stringi_1.7.12
[28] labeling_0.4.2        splines_4.3.0         rprojroot_2.0.3
[31] fastmap_1.1.1         grid_4.3.0            colorspace_2.1-0
[34] cli_3.6.1             magrittr_2.0.3        utf8_1.2.3
[37] withr_2.5.0           scales_1.2.1          timechange_0.2.0
[40] rmarkdown_2.22        httr_1.4.6            png_0.1-8
[43] hms_1.1.3             kableExtra_1.3.4.9000 evaluate_0.21
[46] knitr_1.43            viridisLite_0.4.2     mgcv_1.8-42
[49] rlang_1.1.1           Rcpp_1.0.10           glue_1.6.2
[52] xml2_1.3.4            svglite_2.1.1         rstudioapi_0.14
[55] jsonlite_1.8.5        R6_2.5.1              systemfonts_1.0.4
```

Literaturverzeichnis

Wickham, H., Çetinkaya-Rundel, M., & Grolemund, G. (o. J.). *R for Data Science* (2. Aufl.). https://r4ds.hadley.nz/