Datenvisualisierung 2

Darstellung der zusammenfassenden Statistik

Daniela Palleschi

2023-06-20

Inhaltsverzeichnis

Wiederholung

Letzte Woche haben wir…

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

Heutige Ziele

Diese Woche werden wir lernen, wie man…

- mehr als drei Variablen mit `facet_wrap()` darstellt
- zusammenfassende Statistiken visualisiert
- mehrteilige Diagramme erstellt (z. B. Verteilungen mit zusammenfassenden Statistiken)

Lust auf mehr?

- Section 2.5 (Visualising relationsips) in Wickham et al. (o. J.)
- Ch. 4 (Representing summary statistics) in Nordmann et al. (2022)

1 Einrichtung

```
# turn off scientific notation
options(scipen = 999)
```

```r
pacman::p_load(tidyverse,
               palmerpenguins,
               ggthemes,
               patchwork)
```

```r
df_penguins <- palmerpenguins::penguins %>%
  drop_na()
```

2 Rückblick: Visualisierung von Verteilungen

- Wir haben dies in Woche 3 anhand von
  - Histogramme (1 numerische Variable)
  - Dichteplots (1 numerische Variable)
  - Streudiagramme (2 numerische Variablen)
  - Balkendiagramme (kategorische Variablen)

- Wie viele Variablen werden in jeder Abbildung in Abbildung 1 dargestellt?
  - Welche *Typen* von Variablen werden in den einzelnen Diagrammtypen dargestellt?

2.1 Geigenplots

- Wir können auch Violinplots verwenden, die derzeit ziemlich angesagt sind
  - im Grunde ein doppelseitiges/gespiegeltes Dichte-Diagramm
- Violinplots gelten als einfacher zu lesen
  - wie wir später sehen werden, lassen sie sich auch leicht mit anderen Plots überlagern

```r
  df_penguins %>%
  ggplot(aes(x = species, y = body_mass_g, fill = species)) +
  geom_violin(alpha = .2) +
  labs(title = "Violin plot",
       x = "Body mass (g)",
       y = "Count",
    fill = "Species") +
  scale_color_colorblind() +
```

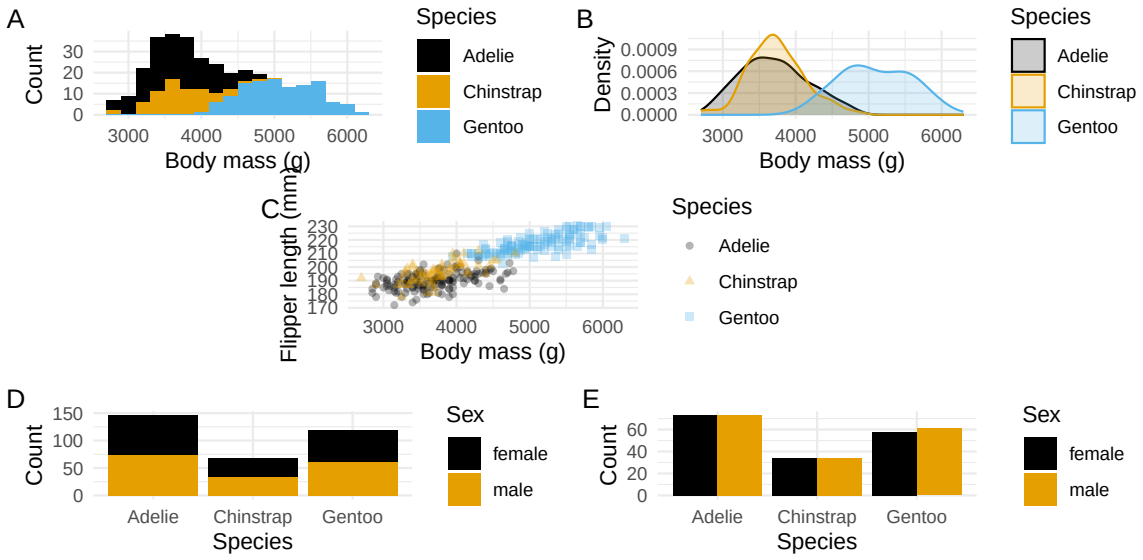


Abbildung 1: Different plots types to visualise distribution of raw data: histogram (A), density plot (B), scatterplot (C), stacked barplot (D), and dodged barplot (E)

```
    scale_fill_colorblind() +
    theme_minimal()
```

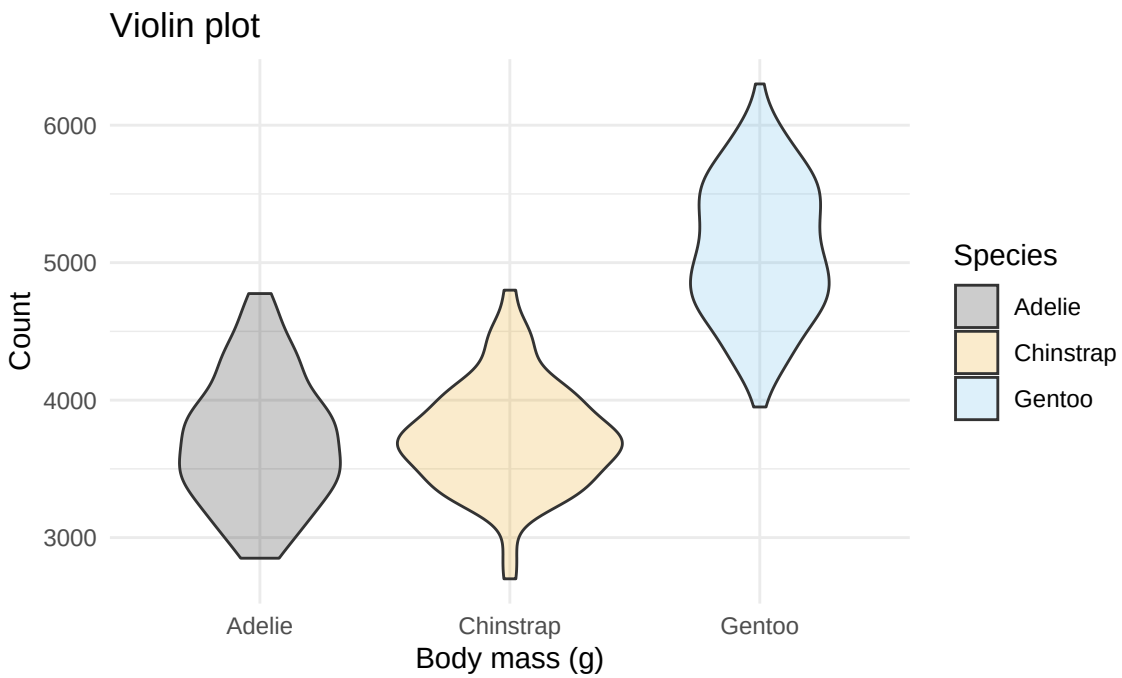


Abbildung 2: Violin plot: a mirrored density plot

Gespiegelte Dichtekurve

Was bedeutet "gespiegeltes" Dichteplot? Geigenplots sind buchstäblich nur ein doppelseitiger Dichteplot. Vergleichen Sie Abbildung 3 mit Abbildung 4. Sie zeigen dieselben Daten und dieselbe Verteilung, aber die Geigen-Darstellung ist einfach eine beidseitige Dichte-Darstellung, aber ohne die entlang der Achse gedruckten "Dichte"-Werte.

```
df_penguins %>%
ggplot(aes(y = body_mass_g, fill = species)) +
facet_grid(~species) +
geom_density(alpha = .2) +
labs(title = "Density plot"
     ) +
scale_fill_colorblind() +
theme_minimal() +
theme(axis.text.x = element_text(angle = 45, hjust=1))
```

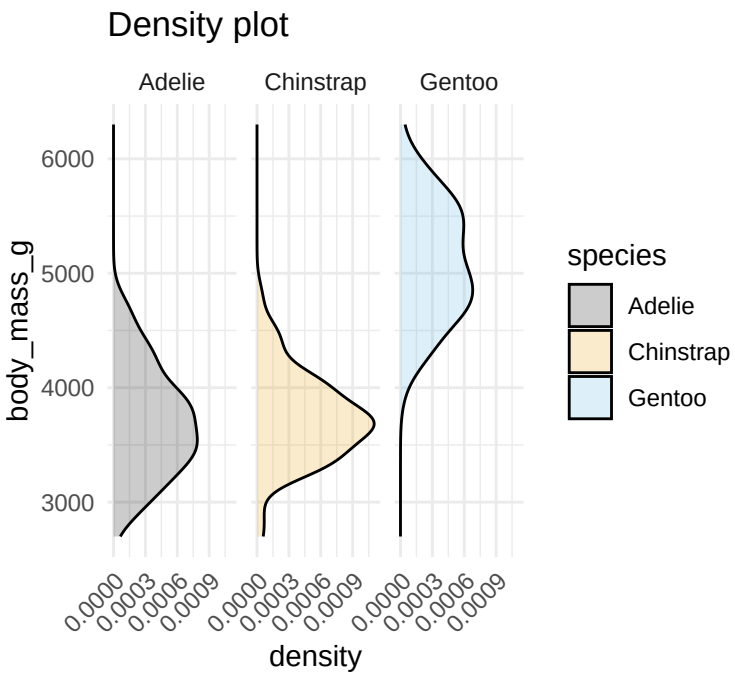


Abbildung 3: Density plot: same violin plot

```
df_penguins %>%
ggplot(aes(x = species, y = body_mass_g, fill = species)) +
geom_violin(alpha = .2) +
labs(title = "Violin plot",
     x = "Body mass (g)",
     y = "Count",
  fill = "Species") +
scale_fill_colorblind() +
theme_minimal()
```

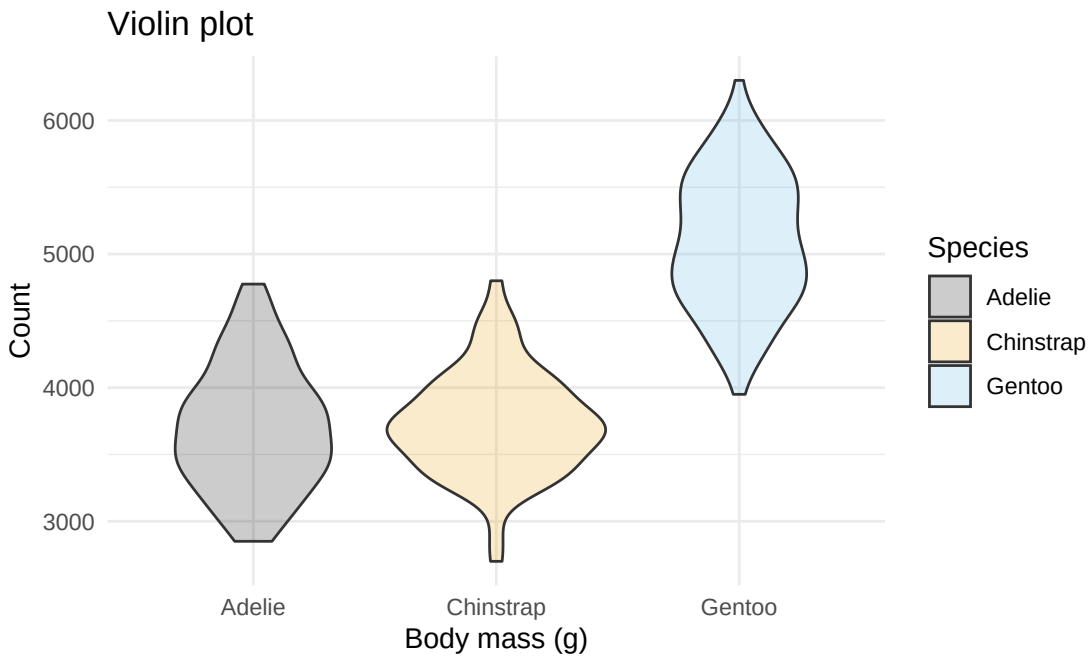


Abbildung 4: Violin plot: a mirrored density plot

3 Visualisierung von 3 oder mehr Variablen

- Wie wir wissen, können wir mehr Variablen einbeziehen, indem wir sie auf die Ästhetik abbilden (z. B. `colour`, `fill` oder `shape`)
- Abbildung 1 hat dies getan, indem es `colour` (alle Diagramme) und `shape` (Scatterplot) verwendet hat, um `species` oder `sex` zusätzlich zu dem zu visualisieren, was entlang der x- und y-Achse abgebildet wurde

---

- das Hinzufügen von zu vielen Variablen zu einer einzigen Darstellung kann die Lesbarkeit erschweren
- Wie viele Variablen werden zum Beispiel im folgenden Code abgebildet?

```
df_penguins %>%
  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()
```

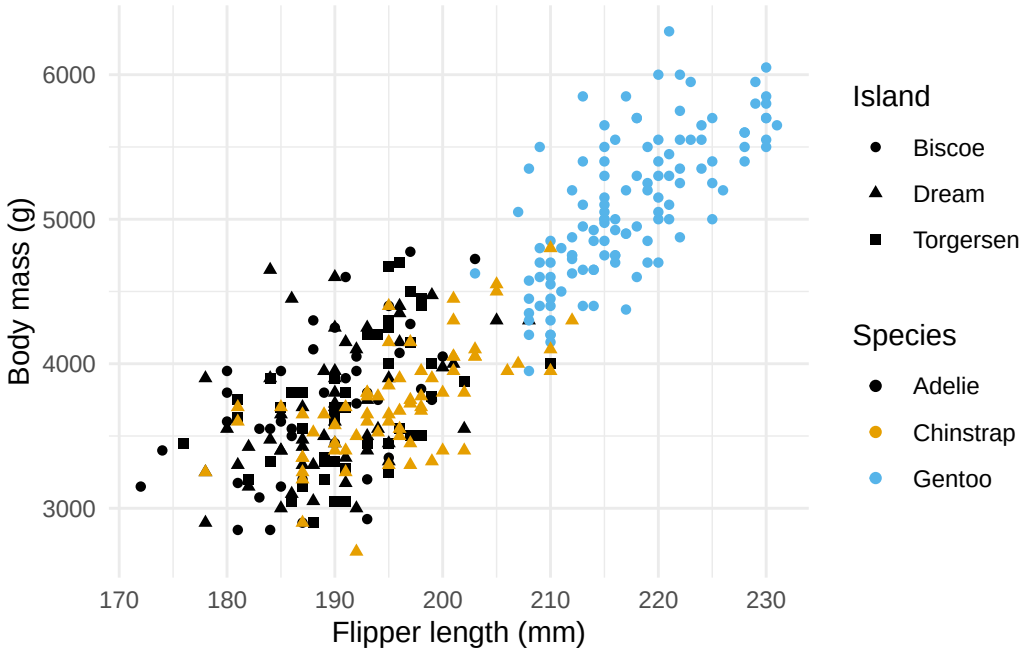


Abbildung 5: A cluttered scatterplot with 4 variables

- vier: `flipper_length_mm` (x-Achse), `body_mass_g` (y-Achse), `species` (Farbe), `island` (Form)
  - das ist optisch etwas unübersichtlich!

3.1 `facet_wrap()`

- eine gute Möglichkeit, unsere Daten in verschiedene Panels aufzuteilen, ist die Verwendung von `facet_wrap()`
  - kann verwendet werden, um ein unübersichtliches Diagramm in separate Panels zu unterteilen

- Versuchen wir, Abbildung 5 mit `facet_wrap()` in drei Panels zu unterteilen, und zwar nach `island`.

```
df_penguins %>%
  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()
```

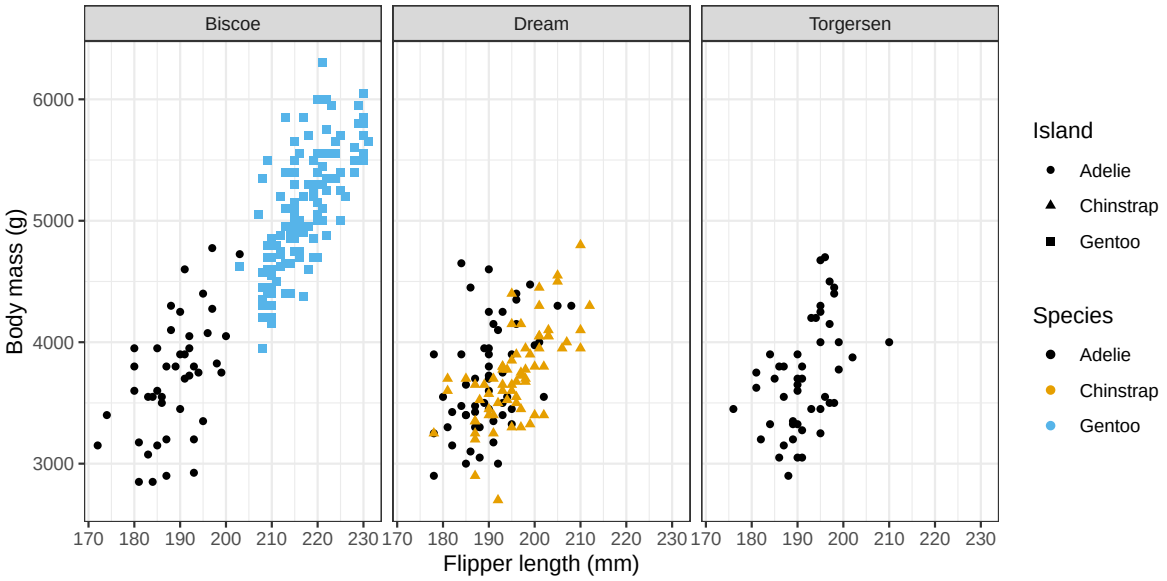


Abbildung 6: A cluttered scatterplot with 4 variables

- Welche Art von Variablen kann `facet_wrap()` als Argument(e) annehmen?
  - kategorisch! Jede 'Kategorie' erhält ihr eigenes Panel

`facet_grid()`

`facet_wrap()` ist verwandt mit `facet_grid()`, das zwei kategoriale Variablen, eine in Spalten und eine in Zeilen, annehmen kann. Das Argument für `facet_grid()` ist eine Gleichung: `row~column`. Wenn wir also `facet_grid(sex~island)` zu unserem Diagramm hinzufügen, sollten wir die Daten in Diagrammen sehen, die nach `sex` in Zeilen (eine Zeile für `female`, eine Zeile für `male`) und nach `island` in Spalten (eine Spalte für jedes `island`) gruppiert sind.

```
df_penguins %>%
  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()
```

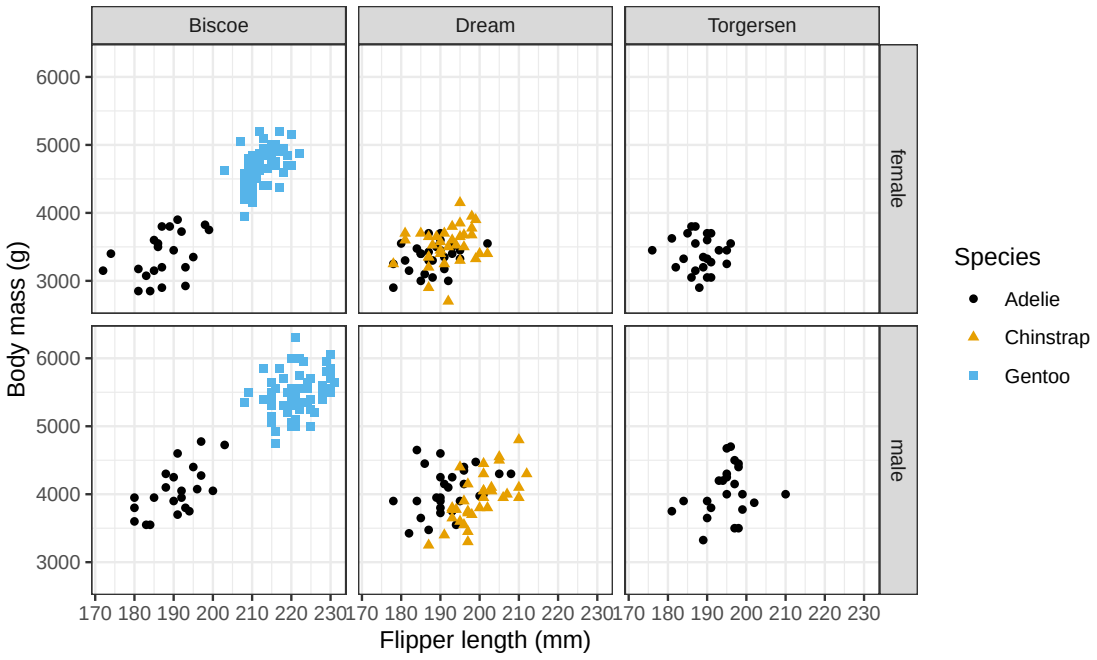


Abbildung 7: facet_grid(sex~species)

4 Darstellung von zusammenfassenden Statistiken

- Letzte Woche haben wir über zusammenfassende Statistiken gesprochen
  - Maße der zentralen Tendenz und Maße der Streuung
- jetzt werden wir lernen, wie man einige dieser Statistiken visualisiert
  - und lernen einige neue Statistiken kennen

4.1 Boxplot

- Boxplots (manchmal auch Box-and-Whisker-Plots genannt) enthalten:
  - eine **Box** mit einer **Linie in der Mitte**
  - **Linien, die aus dem oberen und unteren Rand der Box herausragen** (die Whisker)
- die darstellen:

- **dicke Linie**: den Median, auch Q2 genannt (2. Quartil; der mittlere Wert, über/unter dem 50% der Daten liegen)
- **Box**: der Interquartilsbereich (IQR; der Wertebereich zwischen den mittleren 50% der Daten), mit den Grenzen:
  - Q1 (1. Quartil, unter dem 25% der Daten liegen)
  - Q3 (3. Quartil, oberhalb dessen 25% der Daten liegen)
- **Whiskers**: 1,5*IQR von Q1 (unterer Whisker) oder Q3 (oberer Whisker)
  - aber nur bis zur letzten Beobachtung in diesem Bereich
- **Punkte**: Ausreißer (außerhalb des IQR)

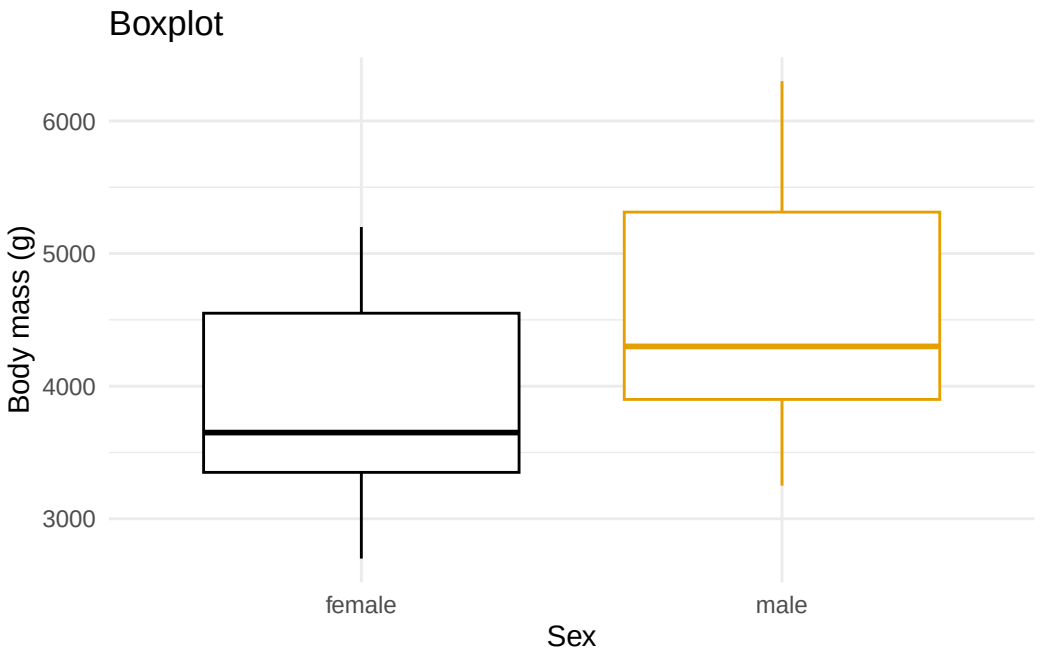


Abbildung 8: Boxplot of `df_penguins` (body mass by sex)

Whisker length

Die Whisker *können* so lang sein wie Q3+(IQR*1,5) und Q1-(IQR*1,5), enden aber an der letzten Beobachtung, die in diesen Bereich fällt. Betrachten wir beispielsweise Abbildung 9: Wenn die Whisker das 1,5-fache des IQR sein sollen, müssten sie länger sein als die Box, aber das scheinen sie nicht zu sein. Woran liegt das? Konzentrieren wir uns auf den unteren Whisker für weibliche Pinguine (den unteren linken Whisker).

Er sollte nur so weit nach unten reichen wie die niedrigste Beobachtung innerhalb des Bereichs von Q1 minus IQR*1,5, aber was sind diese Werte? Berechnen wir mit `R` Q1, Q3, IQR, den oberen und unteren Bereich, den die Whisker haben können (also die Höchst-/Minimalwerte, die die Whisker haben *können*), und schließlich den Mindestwert einer Beobachtung innerhalb dieses Bereichs (also die Mindestwerte, die der untere Whisker haben *wird*).

```
  df_penguins %>%
    filter(sex == "female") %>%
    select(body_mass_g) %>%
    summary()
```

```
  body_mass_g
 Min.   :2700
 1st Qu.:3350
 Median :3650
 Mean   :3862
 3rd Qu.:4550
 Max.   :5200
```

```
  q1 <- 3350
  q3 <- 4550
```

```
  iqr <-
    df_penguins %>%
    filter(sex == "female") %>%
    pull(body_mass_g) %>%
    IQR()
```

```
  q3 - q1
```

```
[1] 1200
```

```
  iqr
```

```
[1] 1200
```

```
  iqr*1.5
```

```
[1] 1800
```

```r
  upper <- q3 + iqr
  lower <- q1 - iqr
```

```r
  upper
```

```
[1] 5750
```

```r
  lower
```

```
[1] 2150
```

```r
  df_penguins %>%
    filter(body_mass_g < q1 &
             body_mass_g > lower &
             sex == "female") %>%
    select(body_mass_g) %>%
    min()
```

```
[1] 2700
```

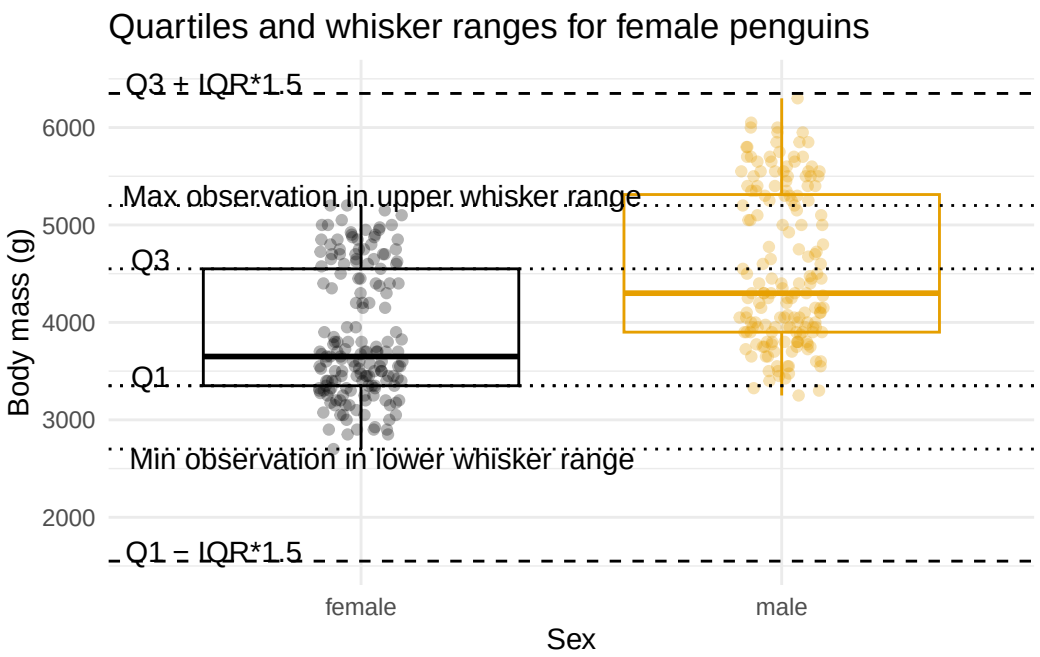


Abbildung 9: Boxplot of body mass (g) with horizontal lines representing quartiles and whisker ranges for female penguins, with observations (scatterplot).

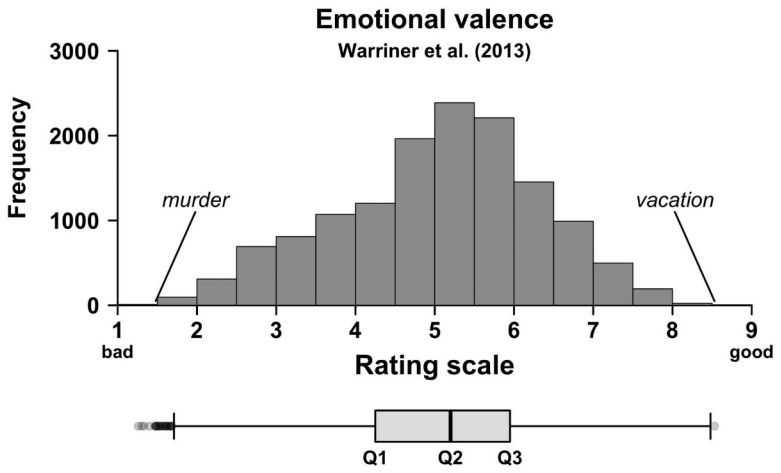


Figure 3.4. A histogram of the emotional valence rating data

Abbildung 10: Image source: Winter (2019) (all rights reserved)

Oder, anders ausgedrückt:

4.1.1 `geom_boxplot()`

- können wir Boxplots mit `geom_boxplot()` erstellen

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

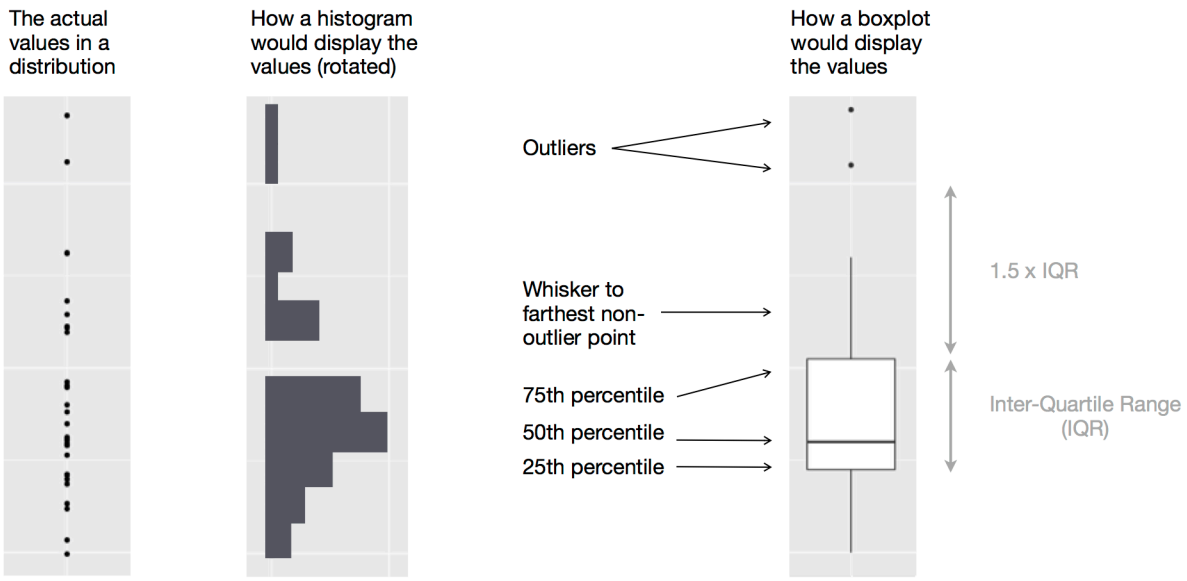


Abbildung 11: Image source: Wickham et al. (o. J.) (all rights reserved)

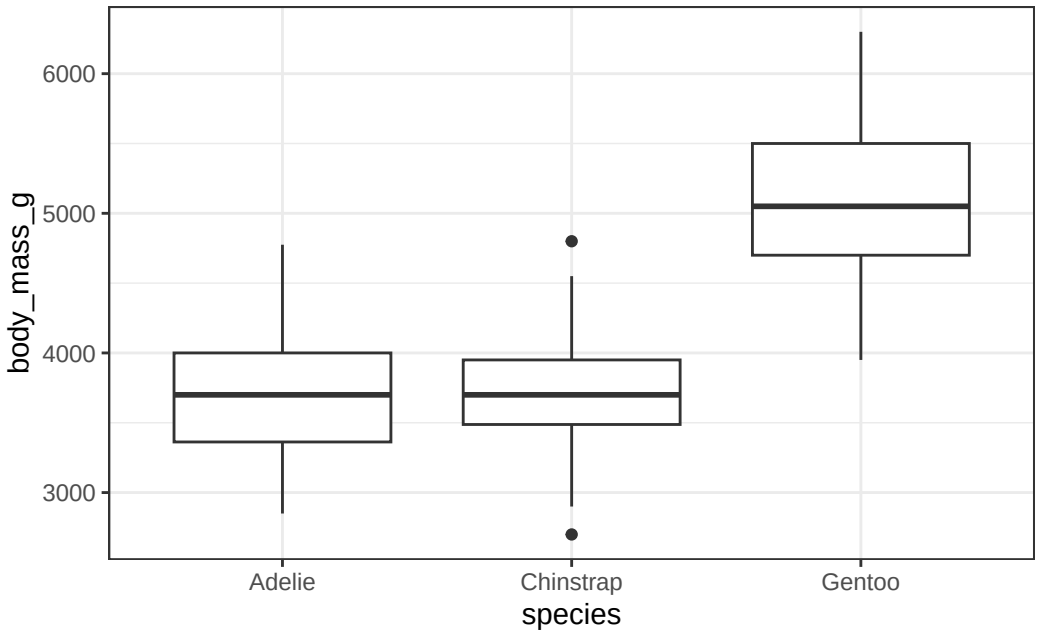


Abbildung 12: `geom_boxplot()`

- Wie viele/welche Variablentypen nimmt `geom_boxplot()`?
  - 2 Variablen: 1 kontinuierlich (`body_mass_g`), 1 kategorisch (`species`)

4.1.2 Gruppierter Boxplot

- Wie ein Balkendiagramm können wir gruppierte Boxplots erstellen, um mehr Variablen zu visualisieren
  - einfach eine neue Variable mit `colour` oder `fill` ästhetisch zuordnen

```
df_penguins %>%
  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()
```

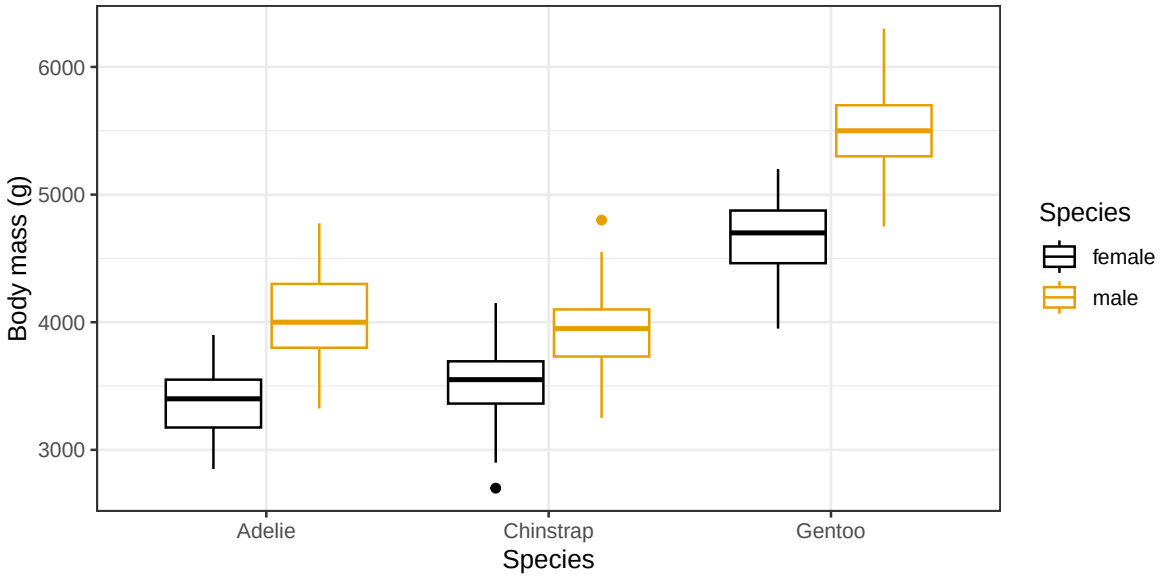


Abbildung 13: A grouped boxplot

5 Visualisierung des Mittelwerts

- Boxplots zeigen ein Maß für die zentrale Tendenz und mehrere Maße für die Streuung
  - Median, IQR (Q1 und Q3), 1,5*IQR (Whiskers) und Ausreißer (Punkte)
- In der Regel möchten wir jedoch den Mittelwert und die Standardabweichung beschreiben, wenn wir Schlussfolgerungen über Unterschiede zwischen Gruppen ziehen.
- Wie können wir dies tun?

5.1 Fehlerbalkenplots

- Wir können den Mittelwert und die Standardabweichung mit Fehlerbalkenplots visualisieren
  - manchmal auch Interaktionsdiagramme genannt
- Diese Darstellungen bestehen aus 2 Teilen:
  - den Mittelwert, visualisiert mit `geom_point()`
  - die Standardabweichung, visualisiert mit `geom_errorbar()`

- die Fehlerbalken stellen den Bereich von 1 Standardabweichung über und unter dem Mittelwert dar (Mittelwert +/- 1SD)

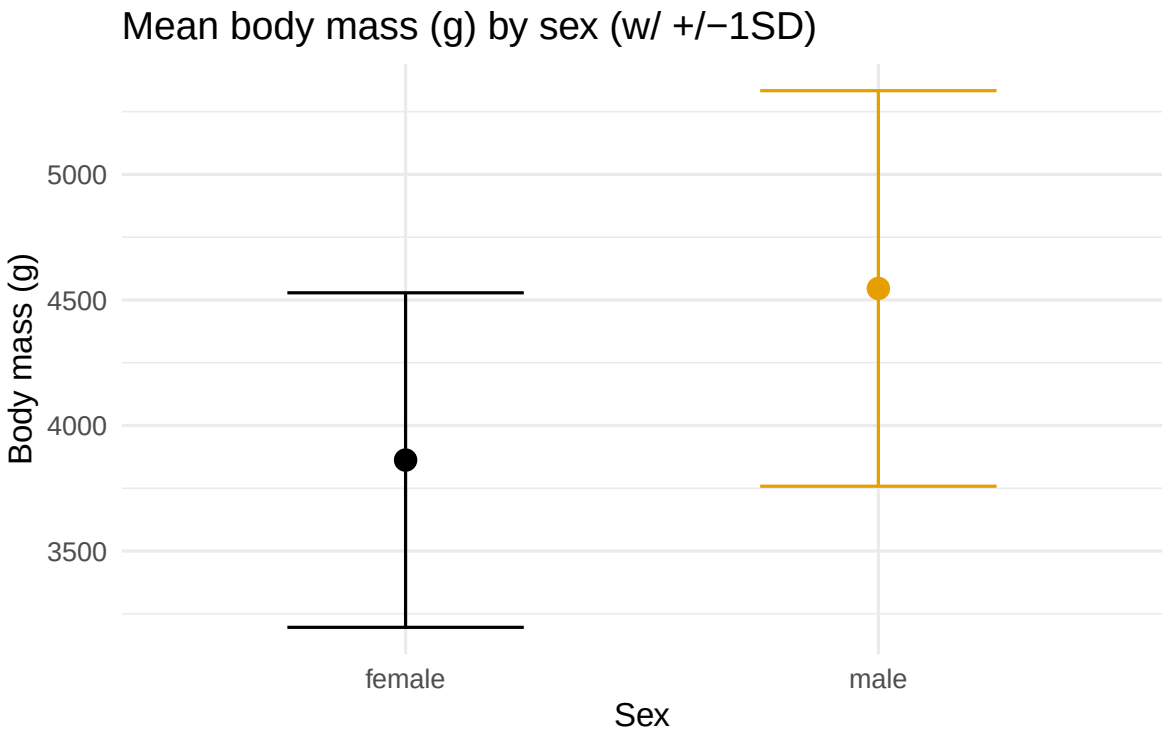


Abbildung 14: Errorbar plot of `df_penguins` (body mass by sex)

Errorbars: standard deviation

In wissenschaftlichen Veröffentlichungen stehen sie in der Regel für Konfidenzintervalle (EN: confience intervals), Standardfehler (EN: standard error) oder glaubwürdige Intervalle (in der Bayes-Statistik; EN: credible interval), die eine *Inferenzstatistik* darstellen, die, einfach ausgedrückt, versucht, Ergebnisse aus einer Stichprobenpopulation auf die Population als Ganzes zu verallgemeinern. In diesem Kurs werden wir uns auf die Standardabweichung beschränken, da wir nur versuchen, etwas über die uns vorliegenden Daten zu sagen.
Aus diesem Grund sollten wir immer angeben, was unsere Fehlerbalken darstellen, z. B. indem wir "(w/ +/-1SD)" in den Titel oder die Beschriftung der Grafik aufnehmen.

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

5.1.0.1 Berechnung der zusammenfassenden Statistik

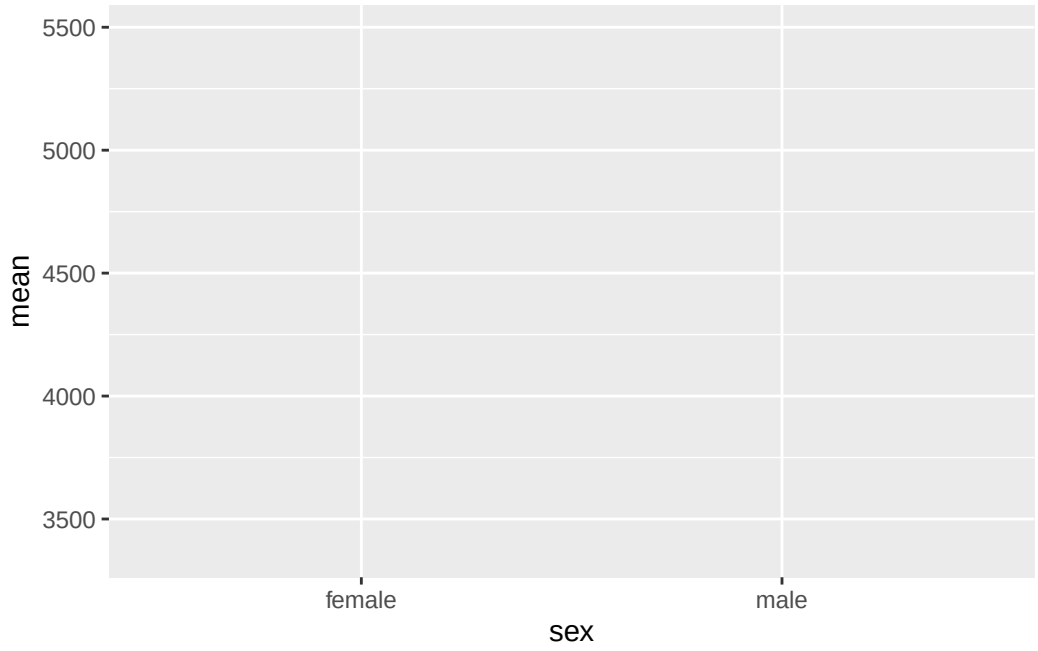
- Zunächst müssen wir den Mittelwert und die Standardabweichung berechnen, gruppiert nach den Variablen, die wir visualisieren wollen
  - Bleiben wir bei `body_mass_g` nach `species` und `sex`
  - Wie können wir den Mittelwert und die Standardabweichung von "body_mass_g" nach `species` und `sex` berechnen?

```
df_penguins %>%
  summarise(mean = mean(body_mass_g),
            sd = sd(body_mass_g),
            N = n(),
            .by = c(species,sex)) %>%
  arrange(species, sex) %>%
  knitr::kable() %>%
  kableExtra::kable_styling(font_size = 30)
```

- wir müssen diese Zusammenfassung in `ggplot2` einspeisen
  - ***ohne*** die Tabellenformatierung von `knitr` und `kableExtra`!!!!
  - Wir können dies tun, indem wir die Zusammenfassung als neues Objekt speichern, oder die Zusammenfassung direkt mit einer Pipe in ggplot einspeisen

5.1.0.2 Neues Objekt

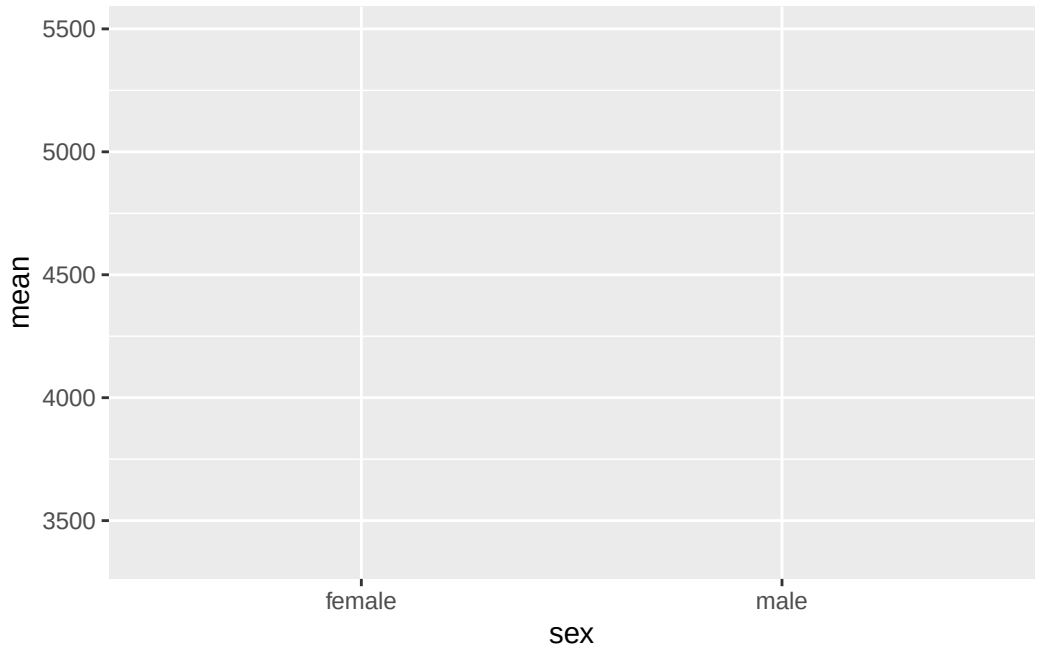
```
# Create new object with summaries
sum_penguins <- df_penguins %>%
  summarise(mean = mean(body_mass_g),
            sd = sd(body_mass_g),
            upper = mean+sd,
            lower = mean-sd,
            N = n(),
            .by = c(species,sex)) %>%
  arrange(species, sex)

# Feed new object into ggplot
sum_penguins %>%
  ggplot(aes(x = sex, y = mean, colour = species))
```



5.1.0.3 Mit einem Pipe

```
df_penguins %>%
  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))
```

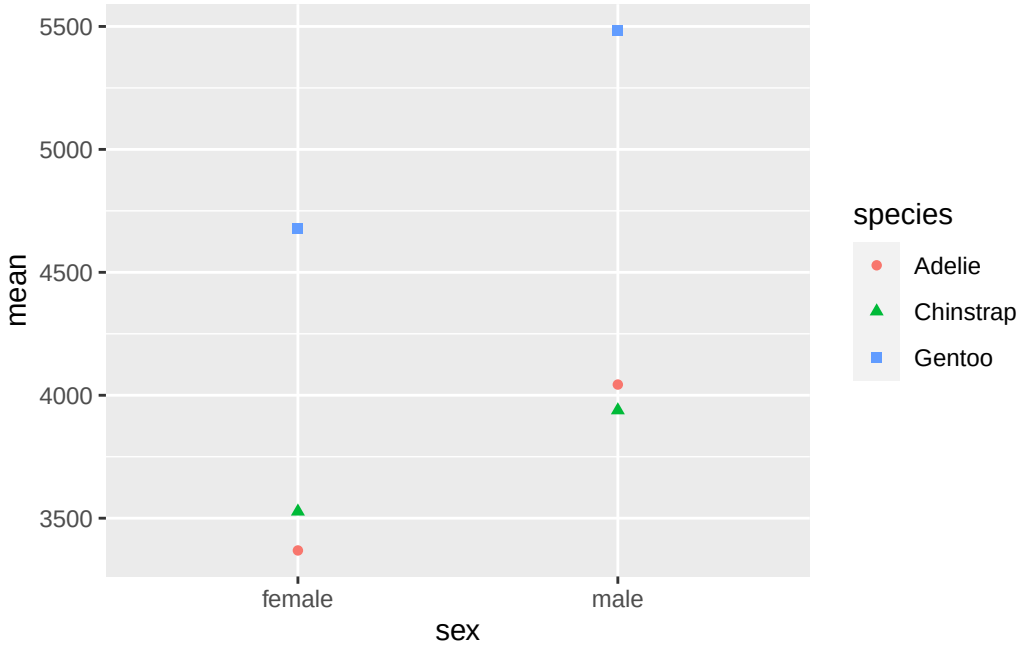


5.1.0.4 Mittelwert aufzeichnen

- wir tun dies mit `geom_point()`

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

species	sex	mean	sd	upper	lower	N
Adelie	female	3368.836	269.3801	3638.216	3099.456	73
Adelie	male	4043.493	346.8116	4390.305	3696.682	73
Chinstrap	female	3527.206	285.3339	3812.540	3241.872	34
Chinstrap	male	3938.971	362.1376	4301.108	3576.833	34
Gentoo	female	4679.741	281.5783	4961.320	4398.163	58
Gentoo	male	5484.836	313.1586	5797.995	5171.677	61

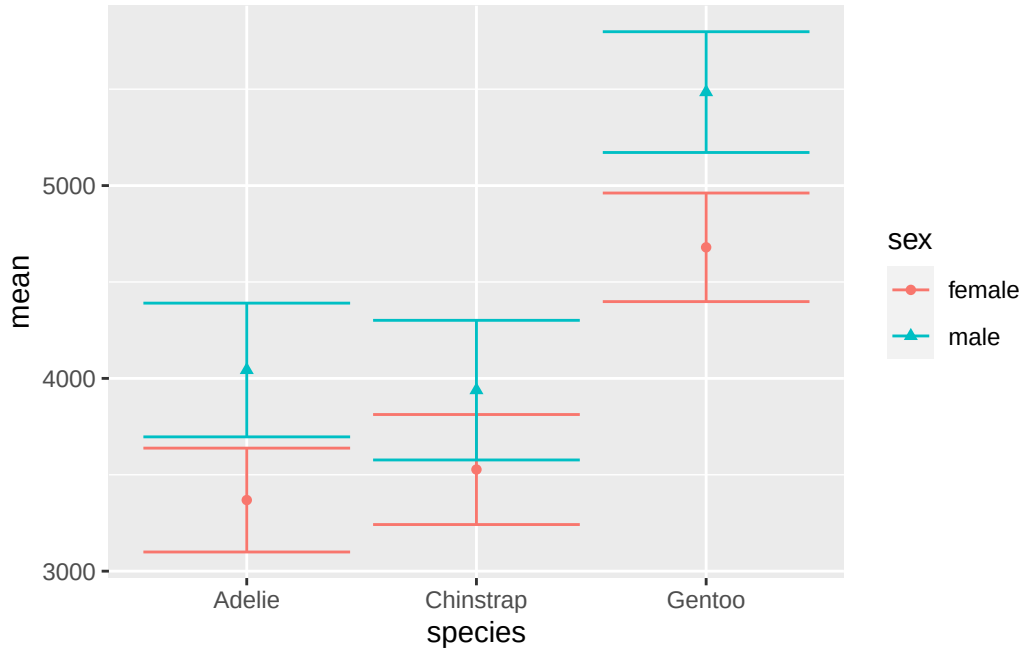


5.1.0.5 Hinzufügen von Fehlerbalken

- wir tun dies mit `geom_errorbar()`
- wir müssen die Abbildungsästhetik für die oberen und unteren Grenzen des Fehlerbalkens hinzufügen
  - `geom_errorbar(aes(ymin = mean-sd, ymax = mean+sd))`
  - Wir haben summarise benutzt, um `mean-sd` (`lower`) und `mean+sd` (`upper`) für jede Gruppe zu berechnen.

```
sum_penguins %>%
  knitr::kable() %>%
  kableExtra::kable_styling()
```

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



Balkendiagramm des Mittelwerts: Finger weg!

Ich flehe Sie an, *nicht* Mittelwerte mit Fehlerbalken darzustellen! Sie werden sehr oft Balkendiagramme von Mittelwerten sehen, und andere unterrichten dies vielleicht sogar in anderen Kursen, aber es gibt viele Gründe, warum dies eine schlechte Idee ist!!! Erstens können sie sehr irreführend sein. Sie beginnen bei 0 und vermitteln den Eindruck, dass die Daten beim Mittelwert enden, obwohl etwa die Hälfte der Daten (normalerweise) über dem Mittelwert liegt.

- Erinnern Sie sich an das Paket `datasauRus`, das Datensätze mit ähnlichen Mittelwerten, Standardabweichungen und Anzahl der Beobachtungen enthält
  - aber *sehr* unterschiedliche Verteilungen
- Abbildung 15 zeigt die Verteilung von 5 dieser Datensätze (oberste Zeile), und den

Mittelwert, die Standardabweichung und die Anzahl der Beobachtungen für die Variablen `x` und `y`

  - Sie werden sehen, dass die Verteilungen sehr unterschiedlich aussehen.

- Aus diesem Grund ist es ein guter Grund, die Rohdatenpunkte *immer* zu visualisieren, unabhängig davon, welche zusammenfassende Darstellung Sie erstellen (z. B. verbergen Fehlerbalken-Diagramme auch eine Menge Daten)

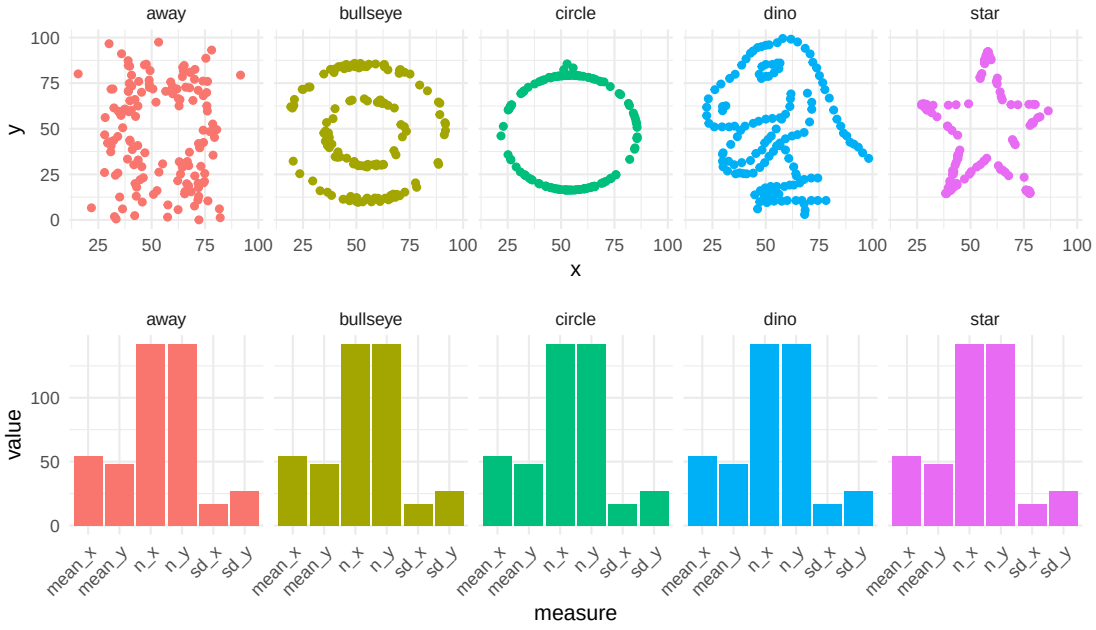


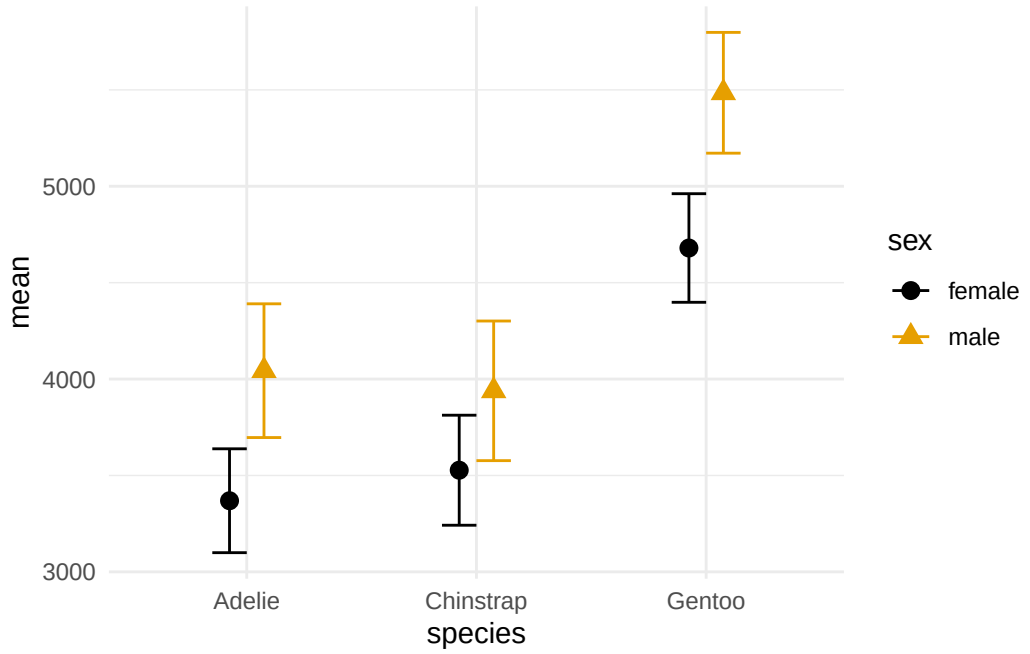
Abbildung 15: Datasets with the same means, sds, and Ns, but very different distributions

5.1.1 Anpassen von

- Welche Anpassungen sehen Sie im Code und in der Darstellung?

```
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()
```



- `position = posiiton_dodge(0.3)` sagt `ggplot2` wie Objekte zu positionieren sind
  - `position_dodge()` bedeutet: überlappende Objekte horizontal verschieben
  - Wichtig ist, dass Sie `position_dodge()` für *jedes* `Geom_` verwenden, das sich an der gleichen Stelle und mit dem gleichen Wert befinden soll, sonst werden sie nicht ausgerichtet
- `geom_point(size = 3)`: passt die Größe der Punkte an
- `geom_errorbar(width = .3)`: passt die Breite der Fehlerbalken an
  - Tipp: Ich gebe immer den gleichen Wert für `position_dodge()` und `geom_errorbar(width = )` ein, so dass die Fehlerbalken immer die 'mittlere' Linie berühren (probieren Sie, beide Werte zu ändern, um zu sehen, was ich meine)
- `scale_colour_colorblind()`: verwendet ein farbenblindenfreundliches Farbschema
- `theme_minimal()`: räumt die Darstellung auf (wir haben auch `theme_bw()` gesehen, mehr über Themes hier)

6 Mehrteilige Diagramme

- Wir können verschiedene Arten von Diagrammen kombinieren, um unsere Daten zusammenzufassen, aber auch die Verteilung darzustellen.
  - Dies ist am einfachsten, wenn sie die gleichen zugrunde liegenden Daten verwenden, wie z. B. Violinplots und Boxplots.

```
df_penguins %>%
  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()
```

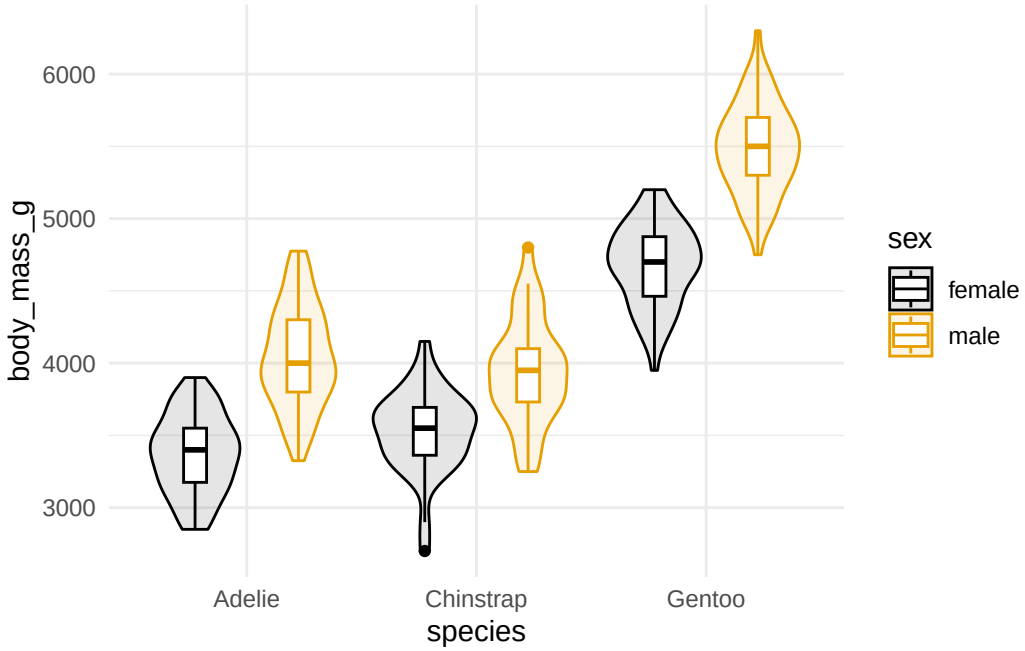


Abbildung 16: A violin-boxplot

6.1 Plotten verschiedener Daten

- dies ist schwieriger, wenn wir Zusammenfassungen (wie Fehlerbalken) *und* Verteilungen darstellen wollen
  - Fehlerbalkenplots nehmen Datenzusammenfassungen (Mittelwert, Sd)
  - Violine, Boxplot und Scatterplots nehmen alle die Rohdaten auf (jede Zeile = Beobachtung)

- Lassen Sie uns versuchen, ein Streudiagramm zu unserem Fehlerbalkenplot hinzuzufügen
  - Dies kann auf verschiedene Weise geschehen, z.B.,
    - man nimmt ein Streudiagramm und fügt den Mittelwert und den Fehlerbalken `Geom` hinzu
    - oder man nimmt die Fehlerbalkengrafik und fügt eine Streuungsgrafik "Geom" hinzu
- Letzteres ist etwas einfacher, also probieren wir das aus

6.1.1 Streudiagramm zur Fehlerleiste hinzufügen

- `geom_point()` mit den erforderlichen `data` und `aes()` verwenden

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

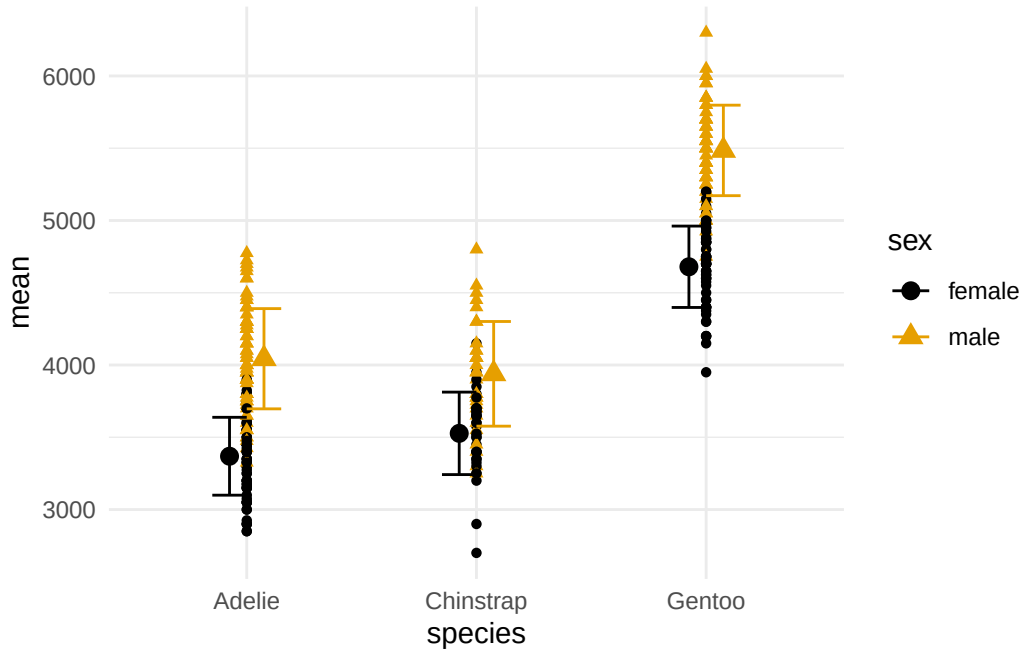


Abbildung 17: Scatterplot with errorbar

6.1.2 Streudiagramm anpassen

- mit `position_dodge()`

```
sum_penguins %>%
  ggplot(aes(x = species, y = mean,
             colour = sex, shape = sex)) +
  geom_point(data = df_penguins,
             aes(x = species, y = body_mass_g),
             position = position_dodge(0.3)) +
  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()
```

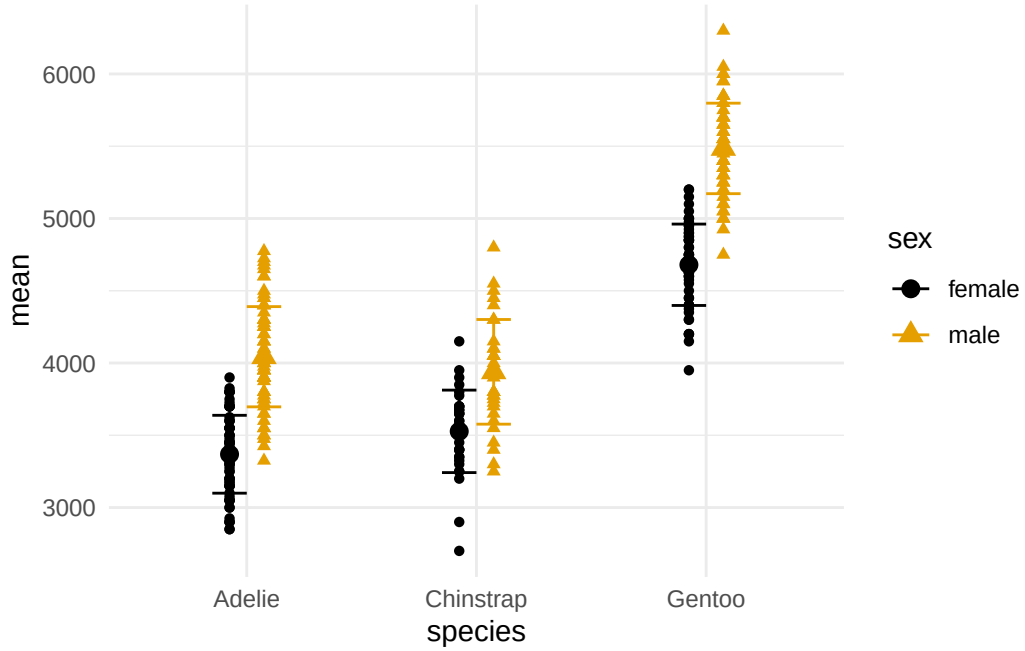


Abbildung 18: Scatterplot with errorbar

6.1.3 Wert `alpha` hinzufügen

- damit wir überlappende Werte unterscheiden können

```
sum_penguins %>%
  ggplot(aes(x = species, y = mean,
             colour = sex, shape = sex)) +
  geom_point(data = df_penguins,
             aes(x = species, y = body_mass_g),
             position = position_dodge(0.3),
             alpha = .4) +
  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()
```

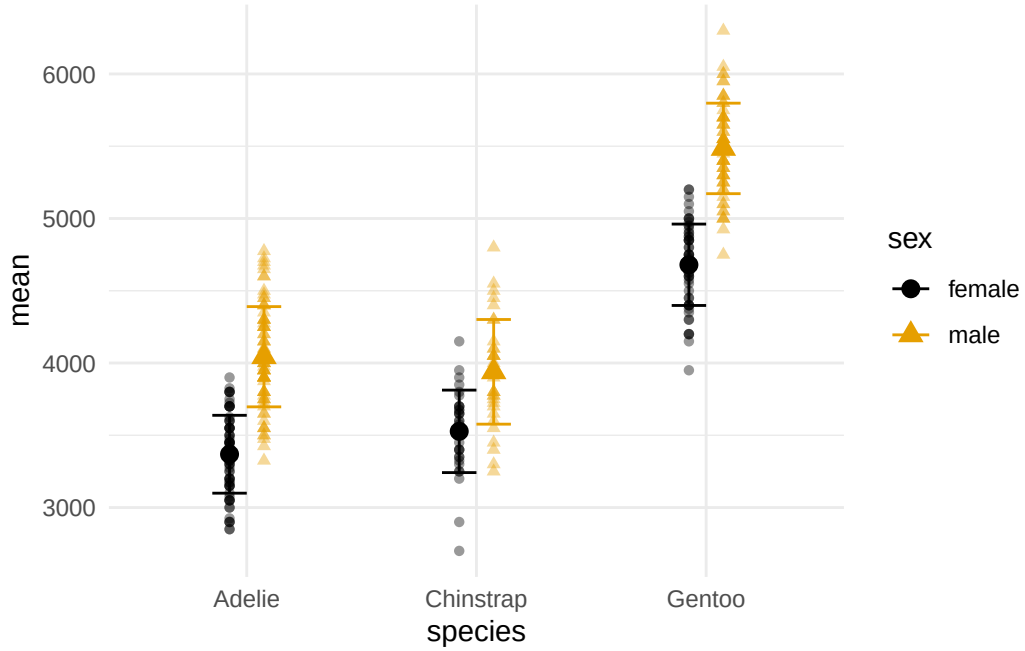


Abbildung 19: Scatterplot with errorbar

6.1.4 Position ändern

- `position_jitterdodge()` verschiebt Objekte so, dass sie sich nicht überlappen
  - wir können `dodge.width = .3` setzen, um `position_dodge()` von errorbars zu entsprechen
  - und `jitter.width =` um anzugeben, wie stark die Punkte zittern sollen
- und `geom_errorbar(size = 1)` macht die Linien der Fehlerbalken dicker

```
sum_penguins %>%
  ggplot(aes(x = species, y = mean,
             colour = sex, shape = sex)) +
  geom_point(data = df_penguins,
             aes(y = body_mass_g),
             position = position_jitterdodge(dodge.width = .3,
                                             jitter.width = 0.3),
             alpha = .4) +
  geom_point(position = position_dodge(width =0.3),
             size = 3) +
  geom_errorbar(aes(ymin=lower,ymax=upper),
```

```
                  position = position_dodge(0.3),
                  width = .3,
                  size = 1) +
  scale_colour_colorblind() +
  theme_minimal()
```

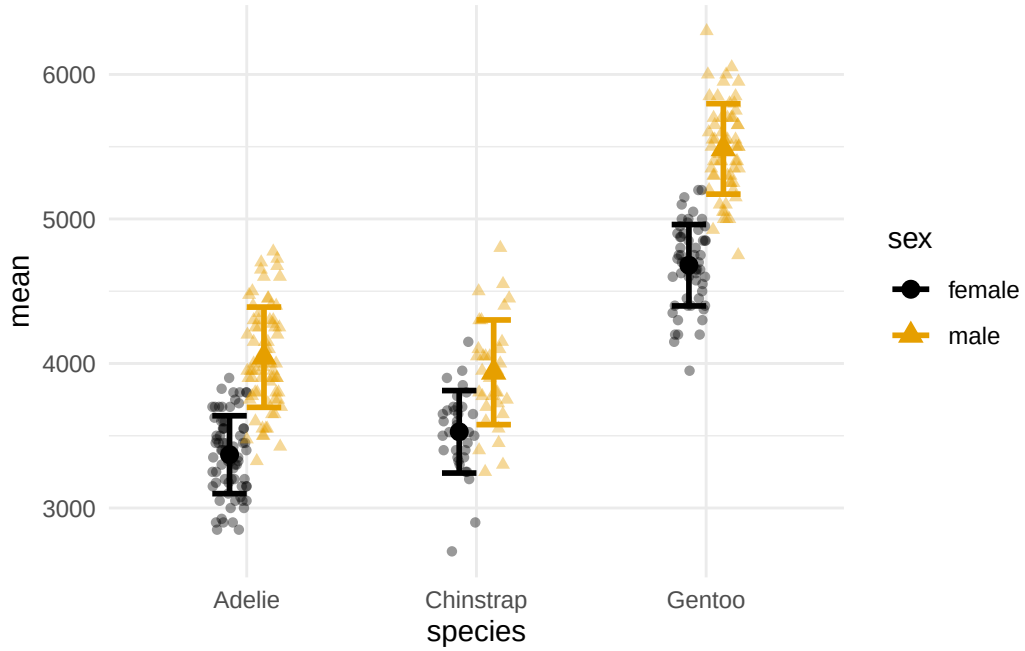


Abbildung 20: Scatterplot with errorbar

Heutige Ziele

Heute haben wir gelernt, wie man…

- mehr als drei Variablen mit `facet_wrap()` darstellt
- zusammenfassende Statistiken visualisiert
- mehrteilige Diagramme erstellt (z. B. Verteilungen mit zusammenfassenden Statistiken)

7 Aufgaben

7.1 Boxplot with facet

1. Erstelle einen Boxplot der `df_penguins` Daten, mit:
    - Geschlecht" auf der "x"-Achse und mit "Farbe" *oder* "Füllung" (wähle eine)
    - Flipper_Länge_mm" auf der y-Achse aufgetragen
    - Insel" in drei Feldern unter Verwendung von "facet_wrap()" aufgetragen
    - die von Ihnen gewählte `Theme_`-Einstellung (z.B. `theme_bw()`; für weitere Optionen siehe hier)

7.2 Code chunk options

2. Fügen Sie der Abbildung eine Beschriftung (`fig-...`) und eine Überschrift (`fig-cap:`) hinzu. Beschreiben Sie die Grafik kurz und verwenden Sie einen Querverweis (*`@fig-...` zeigt, dass...*).

7.3 Multi-layered plot

3. Versuchen Sie, Abbildung 21 zu reproduzieren. Hinweis: Sie müssen Abbildung 16 ein `geom_` und einige Beschriftungen hinzufügen (und ein paar Formatierungsanpassungen, wenn Sie wollen, dass es genau so aussieht).

7.4 Patchwork

4. Verwenden Sie das Paket `patchwork` (siehe Anmerkungen zu Woche 3) und stellen Sie Ihren Boxplot und Ihre Fehlerbalken/Violin-Plots nebeneinander dar. Es sollte ungefähr so aussehen wie Abbildung 22.
    - Hinweis: Wenn Sie die "Tag-Ebenen" ("A" und "B") hinzufügen möchten, müssen Sie `+ plot_annotation(tag_level = "A")` aus `patchwork` hinzufügen.

Session Info

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

```
print(sessionInfo(),locale = F)
```

Abbildung 21: A multi-layered plot

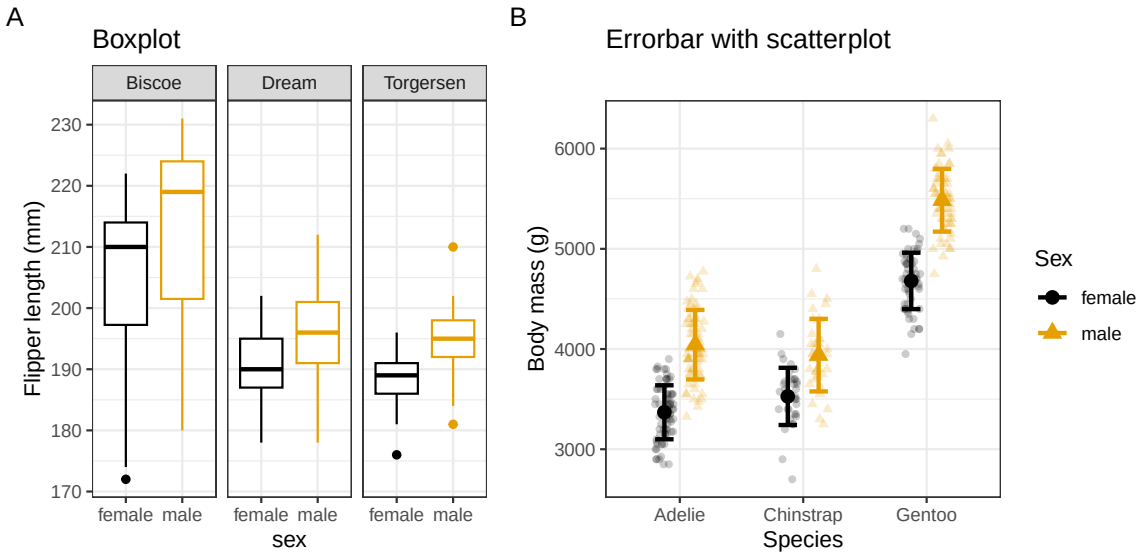


Abbildung 22: Combined plots with `patchwork`

```
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;  L

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

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

loaded via a namespace (and not attached):
 [1] utf8_1.2.3            generics_0.1.3        xml2_1.3.4
 [4] stringi_1.7.12        hms_1.1.3             digest_0.6.31
 [7] magrittr_2.0.3        evaluate_0.21         grid_4.3.0
[10] timechange_0.2.0      fastmap_1.1.1         rprojroot_2.0.3
[13] jsonlite_1.8.5        httr_1.4.6            rvest_1.0.3
[16] fansi_1.0.4           viridisLite_0.4.2     scales_1.2.1
[19] cli_3.6.1             rlang_1.1.1           munsell_0.5.0
[22] withr_2.5.0           yaml_2.3.7            tools_4.3.0
[25] tzdb_0.4.0            colorspace_2.1-0      webshot_0.5.4
[28] pacman_0.5.1          here_1.0.1            kableExtra_1.3.4.9000
[31] png_0.1-8             vctrs_0.6.2           R6_2.5.1
[34] lifecycle_1.0.3       pkgconfig_2.0.3       pillar_1.9.0
[37] gtable_0.3.3          glue_1.6.2            Rcpp_1.0.10
[40] systemfonts_1.0.4     xfun_0.39             tidyselect_1.2.0
[43] rstudioapi_0.14       knitr_1.43            farver_2.1.1
[46] datasauRus_0.1.6      htmltools_0.5.5       svglite_2.1.1
[49] rmarkdown_2.22        labeling_0.4.2        compiler_4.3.0
```

Literaturverzeichnis

Nordmann, E., McAleer, P., Toivo, W., Paterson, H., & DeBruine, L. M. (2022). Data Visualization Using R for Researchers Who Do Not Use R. *Advances in Methods and Practices in Psychological Science*, *5*(2), 251524592210746. https:

//doi.org/10.1177/25152459221074654
Wickham, H., Çetinkaya-Rundel, M., & Grolemund, G. (o. J.). *R for Data Science* (2. Aufl.). https://r4ds.hadley.nz/
Winter, B. (2019). *Statistics for Linguists: An Introduction Using R*. Routledge. https://doi.org/10.4324/9781315165547