Daten rekodieren

Unit 5

BASEL LANDSCHAFT
FINANZ- UND KIRCHENDIREKTION
AMT FÜR DATEN UND STATISTIK

Ziele für heute

1. Datentypen und -klassen in einem Datensatz identifizieren und deren Bedeutung erklären
2. Spalten basierend auf Bedingungen umkodieren
3. Kategorien in Daten mit `{forcats}` umkodieren und sortieren

Datentypen

Warum sollten wir uns für Datentypen interessieren?

BASEL LANDSCHAFT
FINANZ- UND KIRCHENDIREKTION
AMT FÜR DATEN UND STATISTIK

Beispiel

```
1 cat_lovers <- read_csv("data/cat-lovers.csv")
```

name	number_of_cats	handedness
Bernice Warren	0	left
Woodrow Stone	0	left
Willie Bass	1	left
Tyrone Estrada	3	left
Alex Daniels	3	left
Jane Bates	2	left
Latoya Simpson	1	left
Darin Woods	1	left
Agnes Cobb	0	left

Durschnittliche Anzahl

```
cat_lovers |>
  summarise(mean_cats = mean(number_of_cats))
```

```
# A tibble: 1 × 1
  mean_cats
      <dbl>
1        NA
```

Warum funktioniert es nicht?!

```
cat_lovers |>
  summarise(mean_cats = mean(number_of_cats, na.rm = TRUE))
```

```
# A tibble: 1 × 1
  mean_cats
      <dbl>
1        NA
```

Warum funktioniert es immer noch nicht?!

Einatmen… und sich die Daten Anschauen

Welchen Typ hat die Variable `number_of_cats`?

```
cat_lovers |>
  glimpse()
```

```
Rows: 60
Columns: 3
$ name           <chr> "Bernice Warren", "Woodrow Stone", "Willie Bass", "Tyro…
$ number_of_cats <chr> "0", "0", "1", "3", "3", "2", "1", "1", "0", "0", "0", …
$ handedness     <chr> "left", "left", "left", "left", "left", "left", "left",…
```

Noch Einmal einen Blick darauf Werfen

```
cat_lovers |> count(number_of_cats)
```

```
# A tibble: 7 × 2
  number_of_cats                                        n
  <chr>                                             <int>
1 0                                                    32
2 1                                                    15
3 1.5 - honestly I think one of my cats is half human   1
4 2                                                     4
5 3                                                     6
6 4                                                     1
7 three                                                 1
```

Manchmal musst Du auf deine Befragten aufpassen

```
cat_lovers |>
  mutate(number_of_cats = case_when(
    number_of_cats == "1.5 - honestly I think one of my cats is half human" ~ 2,
    number_of_cats == "three" ~ 3,
    .default = as.numeric(number_of_cats)
  )) |>
  summarise(mean_cats = mean(number_of_cats))
```

```
Warning: There was 1 warning in `mutate()`.
ℹ In argument: `number_of_cats = case_when(...)`.
Caused by warning in `vec_case_when()`:
! NAs introduced by coercion
```

```
# A tibble: 1 × 1
  mean_cats
      <dbl>
1     0.833
```

Immer Datentypen Respektieren!

```r
cat_lovers |>
  mutate(
    number_of_cats = case_when(
      number_of_cats == "1.5 - honestly I think one of my cats is half human" ~ "2
      number_of_cats == "three" ~ "3",
      .default = number_of_cats
    ),
    number_of_cats = as.numeric(number_of_cats)
  ) |>
  summarise(mean_cats = mean(number_of_cats))
```

```
# A tibble: 1 × 1
  mean_cats
      <dbl>
1     0.833
```

Jetzt, wo wir wissen, was wir tun…

```
cat_lovers <- cat_lovers |>
  mutate(
    number_of_cats = case_when(
      number_of_cats == "1.5 - honestly I think one of my cats is half human" ~ "2
      number_of_cats == "three" ~ "3",
      .default = number_of_cats
    ),
    number_of_cats = as.numeric(number_of_cats)
  )
```

Moral der Geschichte

- Wenn sich Deine Daten nicht so verhalten, wie du es erwartest, könnte ein *type coercion* beim Einlesen der Daten die Ursache sein.
- Gehe hinein, untersuche deine Daten, wende den Fix an, speichere deine Daten und lebe glücklich bis ans Ende deiner Tage.

Datentypen in R

Atomic vectors

logical: `TRUE`, `FALSE`

character: “Hallo”, “a”, “TRUE”

integer: 2L, 34L, 0L

double: 1, 2.4, `pi`

Datentypen in R

`typeof()` → wie R das Objekt speichert

logical (`TRUE`/`FALSE`)

```
1 typeof(TRUE)
1 [1] "logical"
1 typeof(c(TRUE, FALSE))
1 [1] "logical"
```

character (Text)

```
1 typeof("Hallo")
1 [1] "character"
1 typeof(c("a", "aa"))
1 [1] "character"
```

double (*floating point*)

```
1 typeof(3.56)
1 [1] "double"
1 typeof(c(4, 3))
1 [1] "double"
```

integer (Ganzzahl)

```
1 typeof(4L)
1 [1] "integer"
1 typeof(1:4)
1 [1] "integer"
```

Expliziter vs. Impliziter Typenzwang

- ***Explicit coercion*** `as.logical()`, `as.numeric()`, `as.integer()`, `as.double()`, `as.character()`.

```
1 x <- c(TRUE, FALSE)
2 as.character(x)
```

```
1 [1] "TRUE"  "FALSE"
```

- ***Implicit coercion*** z. B. R konvertiert Variablen gemischter Typen in einen einzelnen Typ.

```
1 c(15, "Danke")
```

```
1 [1] "15"    "Danke"
```

```
1 c(3L, pi)
```

```
1 [1] 3.000000 3.141593
```

… und das ist nicht immer eine gute Sache!

Praktikum: *Type Coercion*

prak-05a-type-coercion.qmd

Welcher Typ sind die angegebenenen Vektoren?

Daten-Rekodierung

`if_else()`, `case_when()`

TRUE/FALSE: `if_else()`

Schnabellänge kategorisieren: “überdurchschnittlich”, “unterdurchschnittlich”

```
1 penguins |>
2   summarise(median_bill_length = median(bill_length_mm, na.rm = TRUE))
```

```
1 # A tibble: 1 × 1
2   median_bill_length
3                <dbl>
4 1               44.4
```

TRUE/FALSE: `if_else()`

`if_else(stimmt_das, das_passiert, sonst_das_passiert)`

```
penguins |>
  mutate(
    bl_cat = if_else(bill_length_mm < 44.45, "unterdurchschnittlich", "überdurchsc
  ) |>
  count(bl_cat)
```

```
# A tibble: 3 × 2
  bl_cat                 n
  <chr>              <int>
1 unterdurchschnittlich   171
2 überdurchschnittlich    171
3 <NA>                      2
```

TRUE/FALSE: `if_else()`

`if_else(stimmt_das, das_passiert, sonst_das_passiert, NA_so_behandeln)`

```
penguins |>
  mutate(
    bl_cat =
      if_else(
        bill_length_mm < 44.45, "unterdurchschnittlich", "überdurchschnittlich", m
      )
  ) |>
  count(bl_cat)
```

```
# A tibble: 3 × 2
  bl_cat                n
  <chr>             <int>
1 unbekannt             2
2 unterdurchschnittlich   171
3 überdurchschnittlich    171
```

Mehrere Bedingungen: `case_when()`

Schnabellänge kategorisieren: *short*, *medium*, *long*

```
penguins |>
  select(bill_length_mm) |>
  summary()
```

```
 bill_length_mm
 Min.   :32.10
 1st Qu.:39.23
 Median :44.45
 Mean   :43.92
 3rd Qu.:48.50
 Max.   :59.60
 NA's   :2
```

Mehrere Bedingungen: `case_when()`

```r
penguins |>
  mutate(
    bl_cat = case_when(
      is.na(bill_length_mm) ~ NA,
      bill_length_mm < 39.2 ~ "short",
      between(bill_length_mm, 39.2, 48.5) ~ "medium",
      .default = "long"
    )
  ) |>
  count(bl_cat)
```

```
# A tibble: 4 × 2
  bl_cat     n
  <chr>  <int>
1 long      84
2 medium   175
3 short     83
4 <NA>       2
```

Praktikum: Daten rekodieren

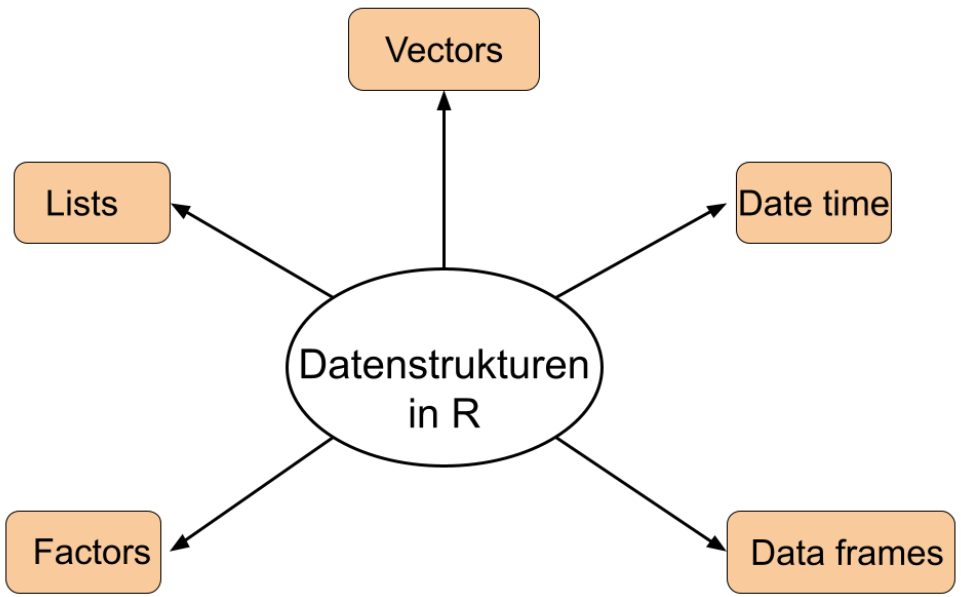
prak-05b-cond-mutate.qmd

Break ☕ 🍵 🍜

10:00

Datenstrukturen

Datenstrukturen



`class()` → wie sich das Objekt verhält

Factors

→ Kategoriale Variablen: Character + Ganzzahl

```
(x <- c("BS", "MS", "PhD", "MS"))
```

```
[1] "BS"  "MS"  "PhD" "MS"
```

```
typeof(x)
```

```
[1] "character"
```

```
class(x)
```

```
[1] "character"
```

```
(y <- factor(x))
```

```
[1] BS  MS  PhD MS
Levels: BS MS PhD
```

```
typeof(y)
```

```
[1] "integer"
```

```
class(y)
```

```
[1] "factor"
```

```
as.integer(y)
```

```
[1] 1 2 3 2
```

Dates

Ganzezahl = Anzahl Tage seit Ursprung

```
(y <- as.Date("1990-01-01"))
```

```
[1] "1990-01-01"
```

```
typeof(y)
```

```
[1] "double"
```

```
class(y)
```

```
[1] "Date"
```

```
as.integer(y)
```

```
[1] 7305
```

```
as.integer(y) / 365
```

```
[1] 20.0137
```

~ 20 Jahre nach dem `1970-01-01`

Lists

Generische Vektorcontainers: Vektoren jeglicher Typ und Länge

```r
l <- list(
  x = 1:4,
  y = c("Hallo", "hello", "salut"),
  z = c(TRUE, FALSE)
)
l
```

```
$x
[1] 1 2 3 4

$y
[1] "Hallo" "hello" "salut"

$z
[1]  TRUE FALSE
```

Data Frames

Spezielle Liste mit Vektoren gleicher Länge

```r
(df <- data.frame(x = 1:2, y = 3:4))
```

```
  x y
1 1 3
2 2 4
```

```r
class(df)
```

```
[1] "data.frame"
```

```r
df |>
  pull(y)
```

```
[1] 3 4
```

```r
df$y
```

```
[1] 3 4
```

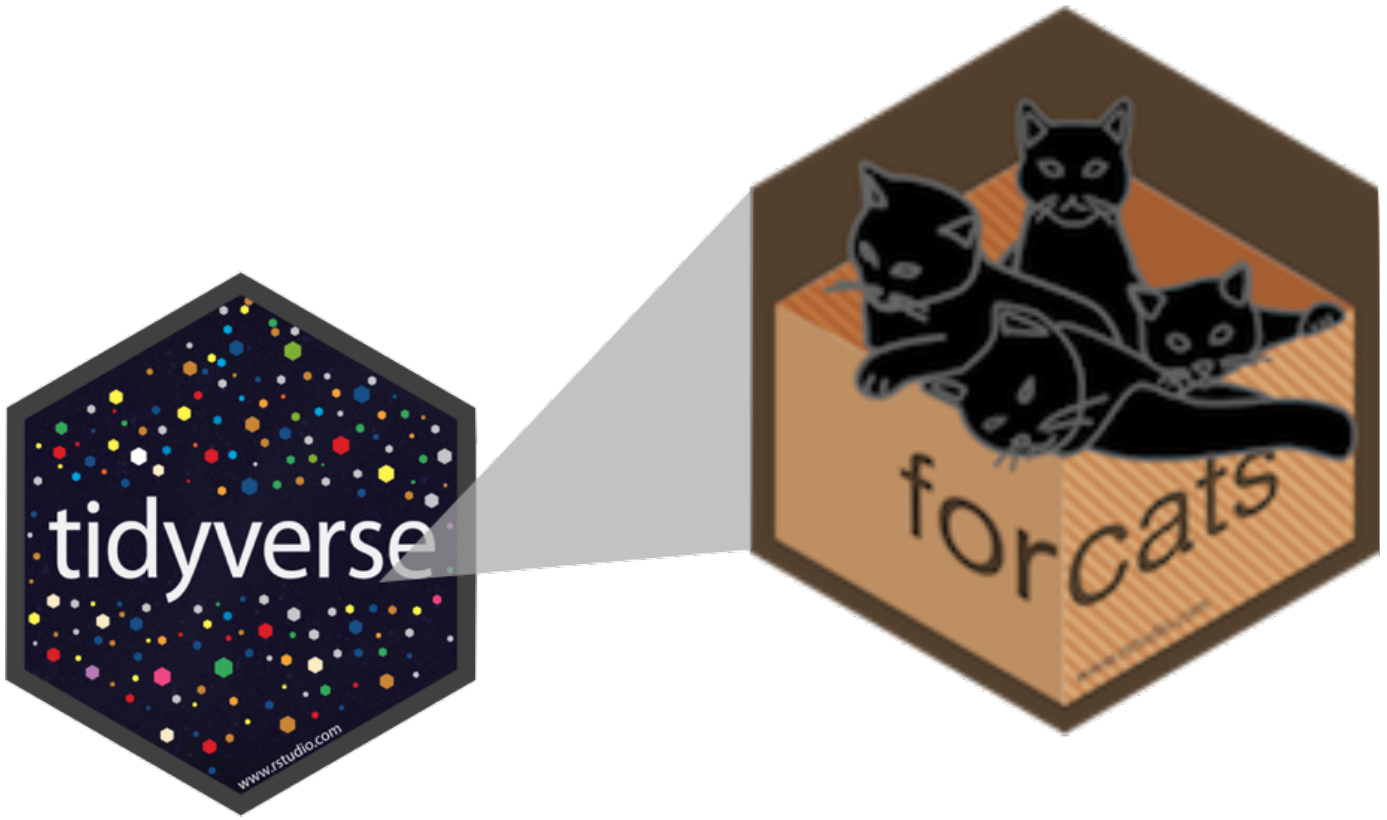
```r
(df <- tibble(x = 1:2, y = 3:4))
```

```
# A tibble: 2 × 2
      x     y
  <int> <int>
1     1     3
2     2     4
```

```r
class(df)
```

```
[1] "tbl_df"     "tbl"        "data.frame"
```

Mit Factors Arbeiten: {forcats}


tidyverse
www.rstudio.com
forcats

Daten

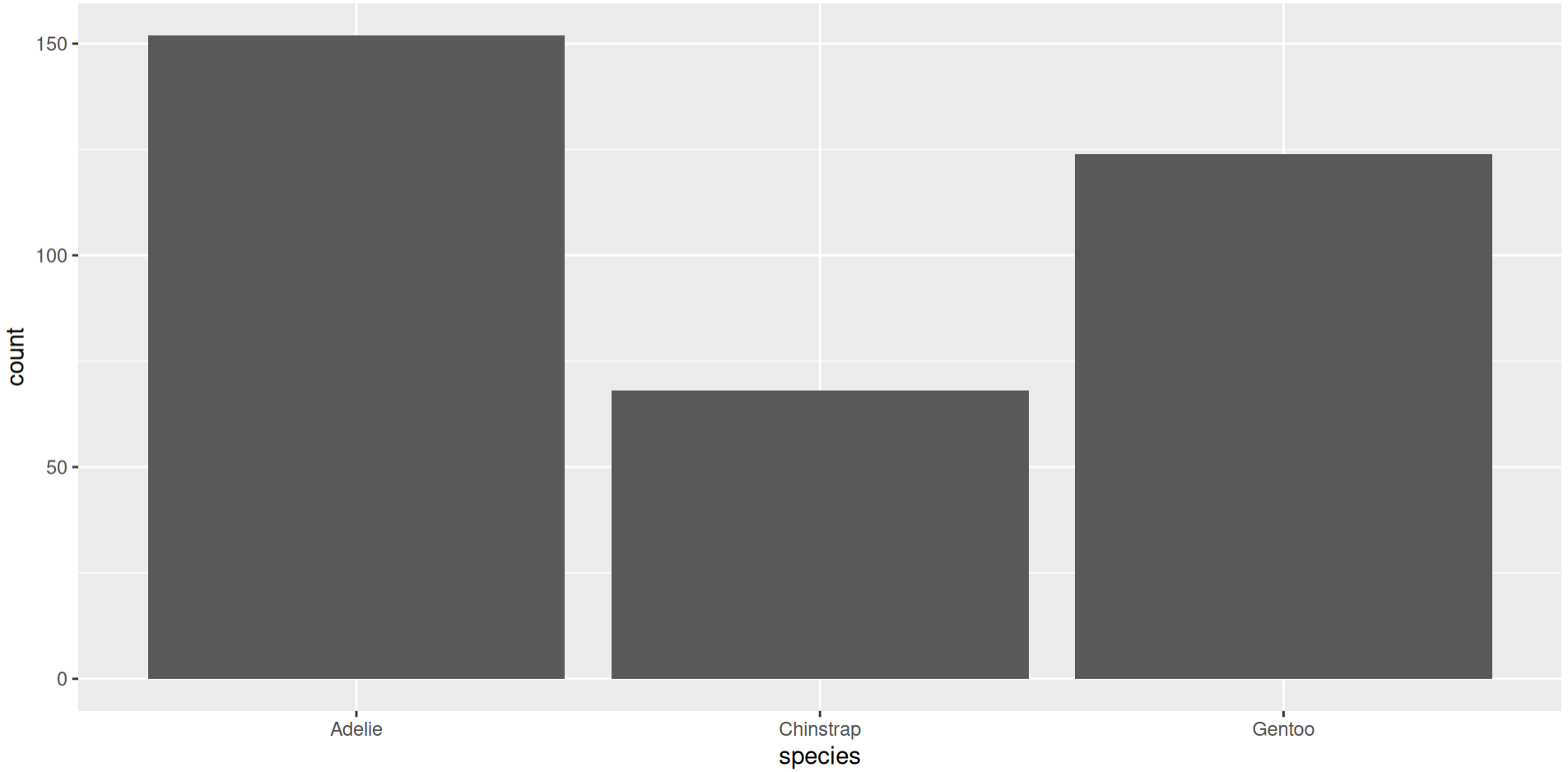
```
penguins |>
  glimpse()
```

```
Rows: 344
Columns: 8
$ species           <fct> Adelie, Adelie, Adelie, Adelie, Adelie, Adelie, Adel…
$ island            <fct> Torgersen, Torgersen, Torgersen, Torgersen, Torgerse…
$ bill_length_mm    <dbl> 39.1, 39.5, 40.3, NA, 36.7, 39.3, 38.9, 39.2, 34.1, …
$ bill_depth_mm     <dbl> 18.7, 17.4, 18.0, NA, 19.3, 20.6, 17.8, 19.6, 18.1, …
$ flipper_length_mm <int> 181, 186, 195, NA, 193, 190, 181, 195, 193, 190, 186…
$ body_mass_g       <int> 3750, 3800, 3250, NA, 3450, 3650, 3625, 4675, 3475, …
$ sex               <fct> male, female, female, NA, female, male, female, male…
$ year              <int> 2007, 2007, 2007, 2007, 2007, 2007, 2007, 2007, 2007…
```

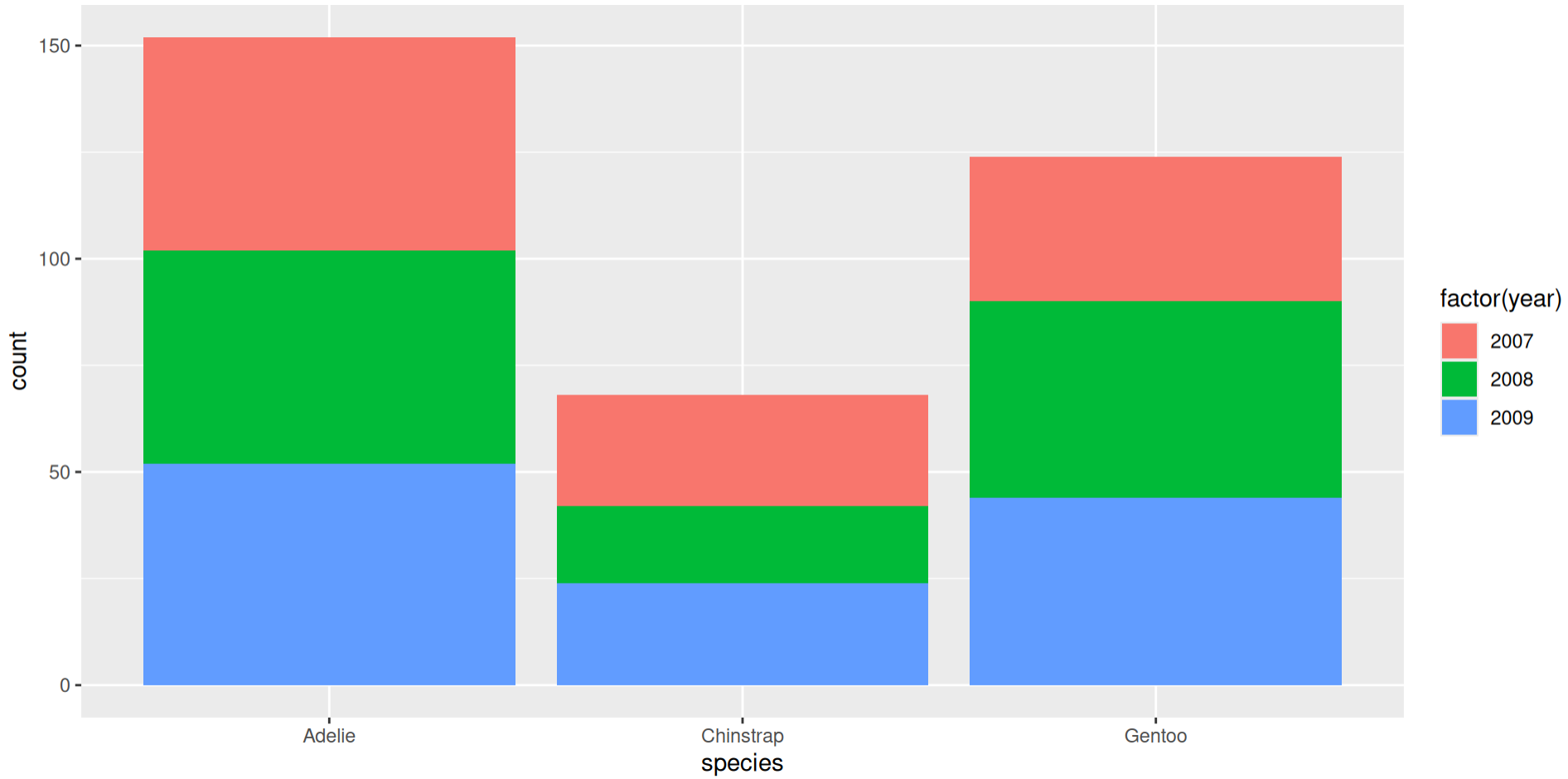
▼ Code

```
1 penguins |>
2   ggplot(aes(x = species, fill = year)) +
3   geom_bar()
```

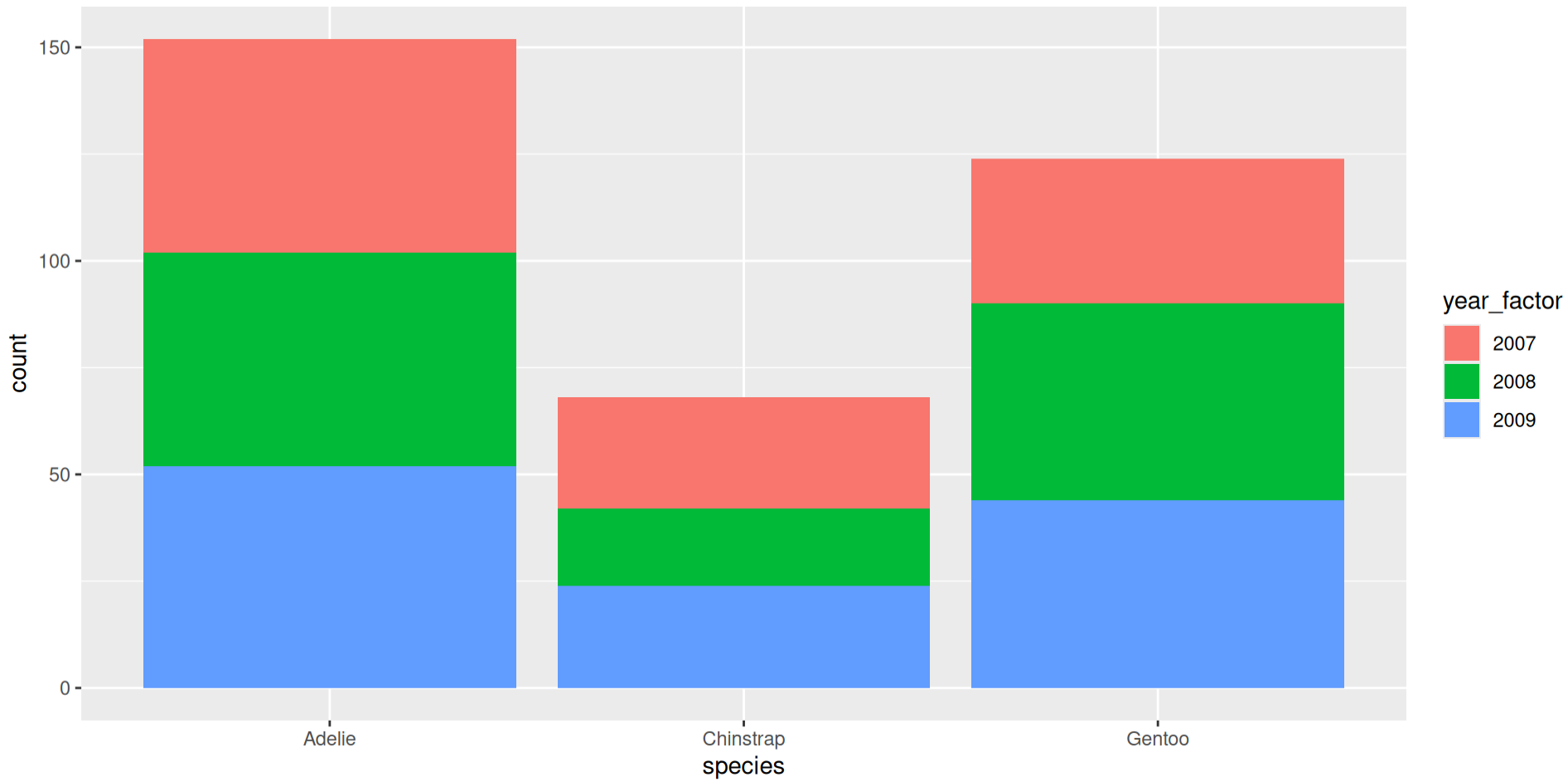
```
Warning: The following aesthetics were dropped during statistical transformation:
fill.
ℹ This can happen when ggplot fails to infer the correct grouping structure in
  the data.
ℹ Did you forget to specify a `group` aesthetic or to convert a numerical
  variable into a factor?
```


150
100
50
0
count
Adelie
Chinstrap
Gentoo
species

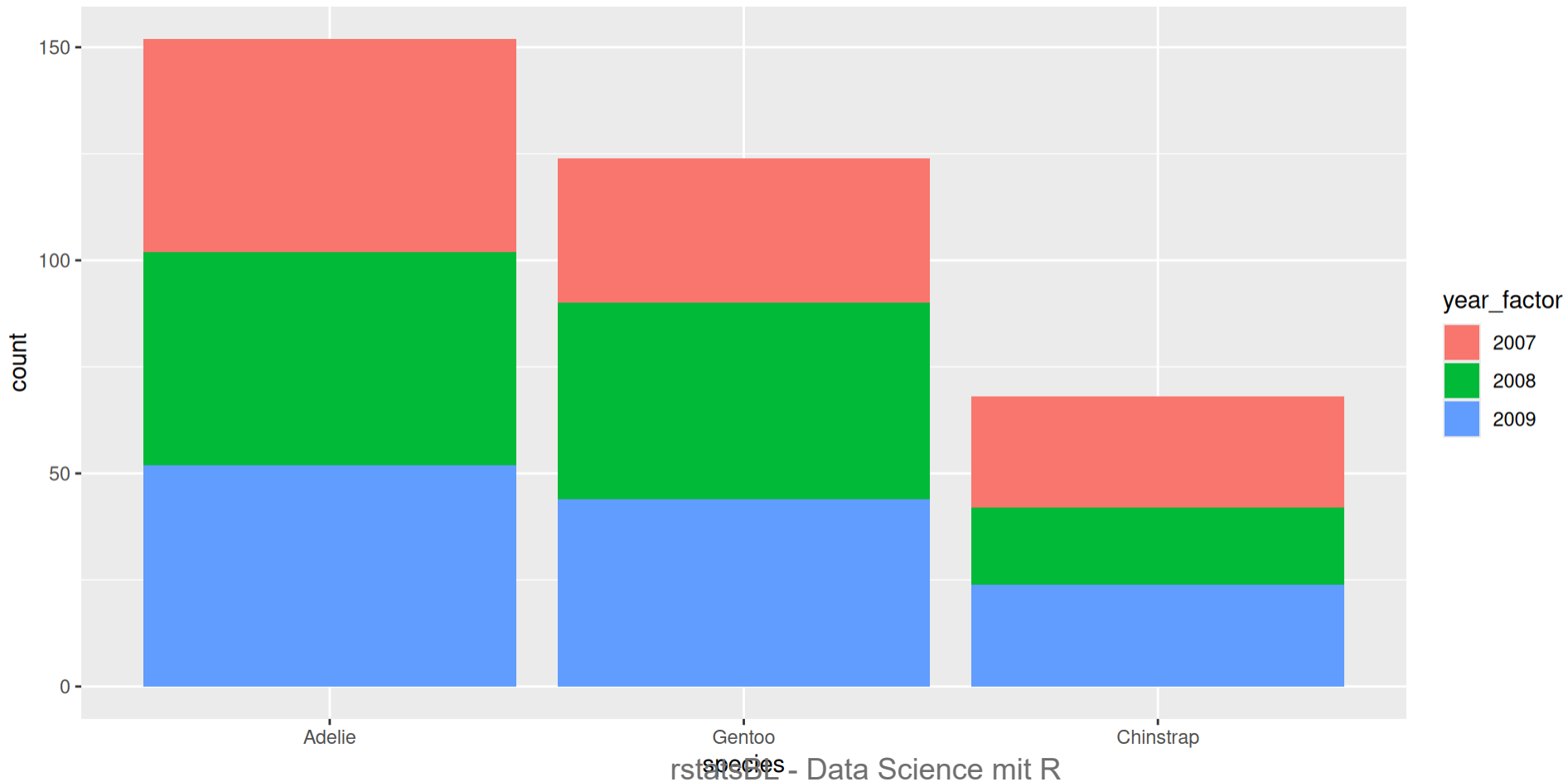
```r
penguins |>
  ggplot(aes(x = species, fill = factor(year))) +
  geom_bar()
```



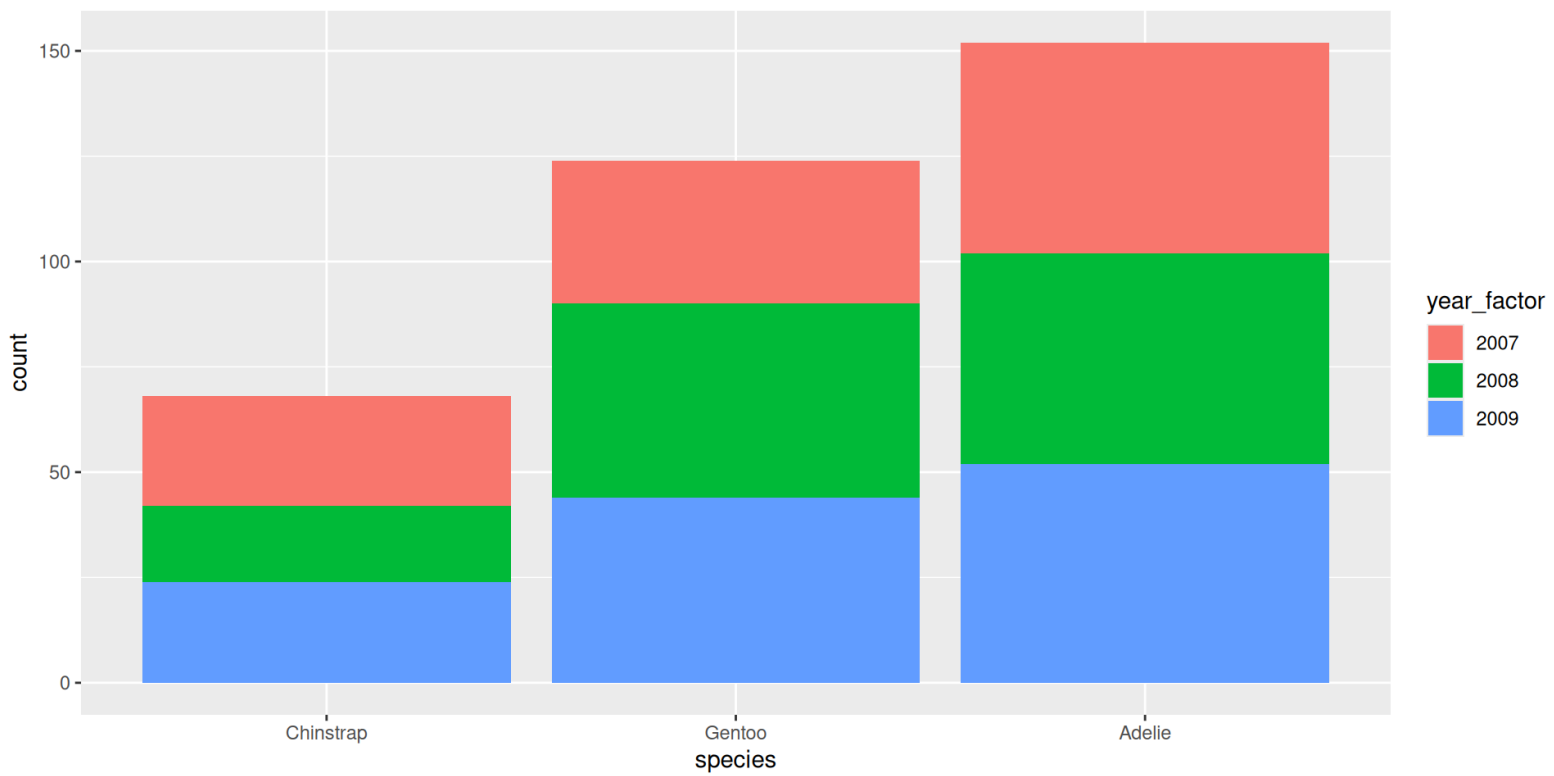
```
penguins |>
  mutate(year_factor = factor(year)) |>
  ggplot(aes(x = species, fill = year_factor)) +
  geom_bar()
```



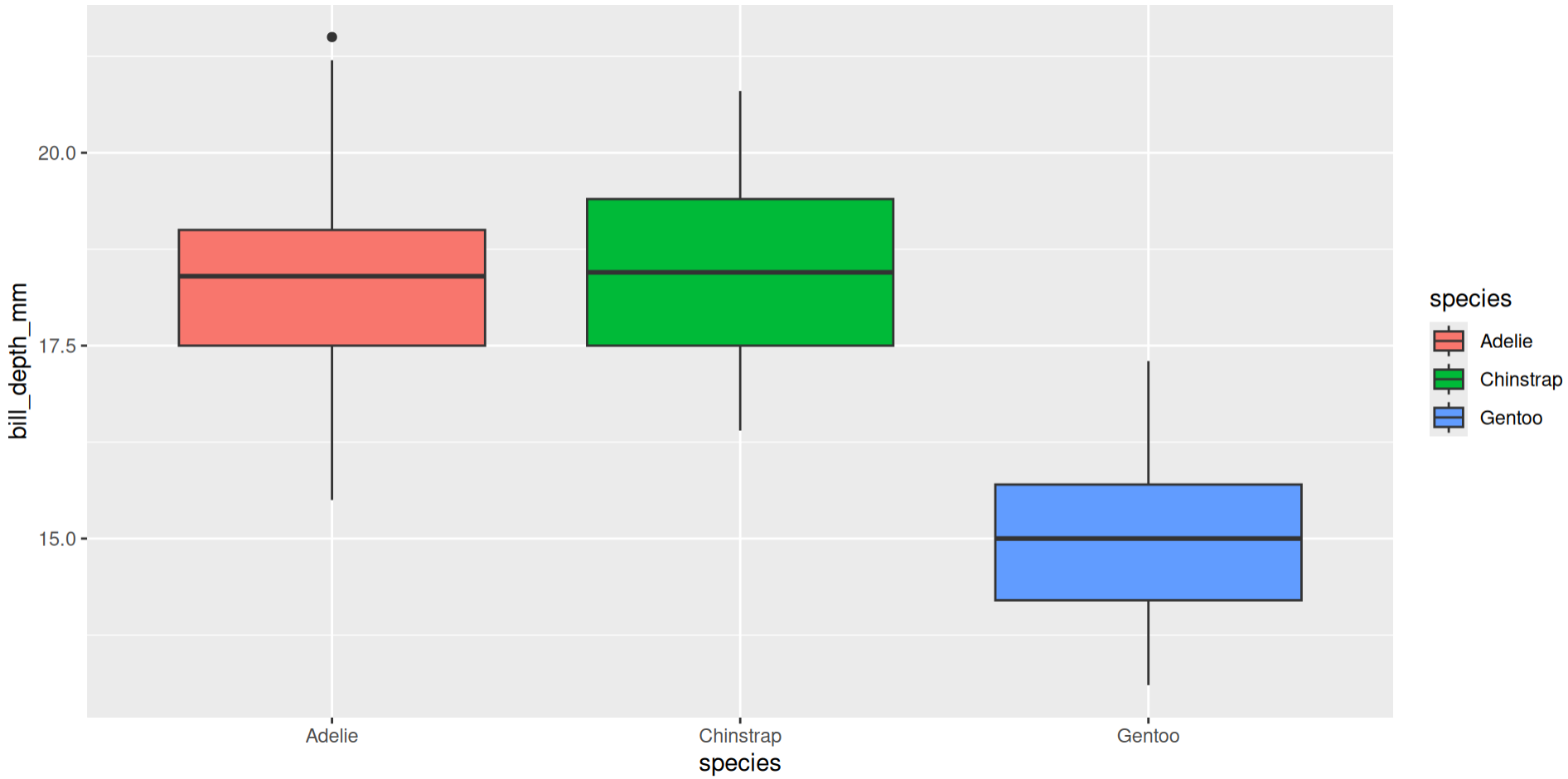
```r
penguins |>
  mutate(
    year_factor = factor(year),
    species = fct_infreq(species)
  ) |>
  ggplot(aes(x = species, fill = year_factor)) +
  geom_bar()
```



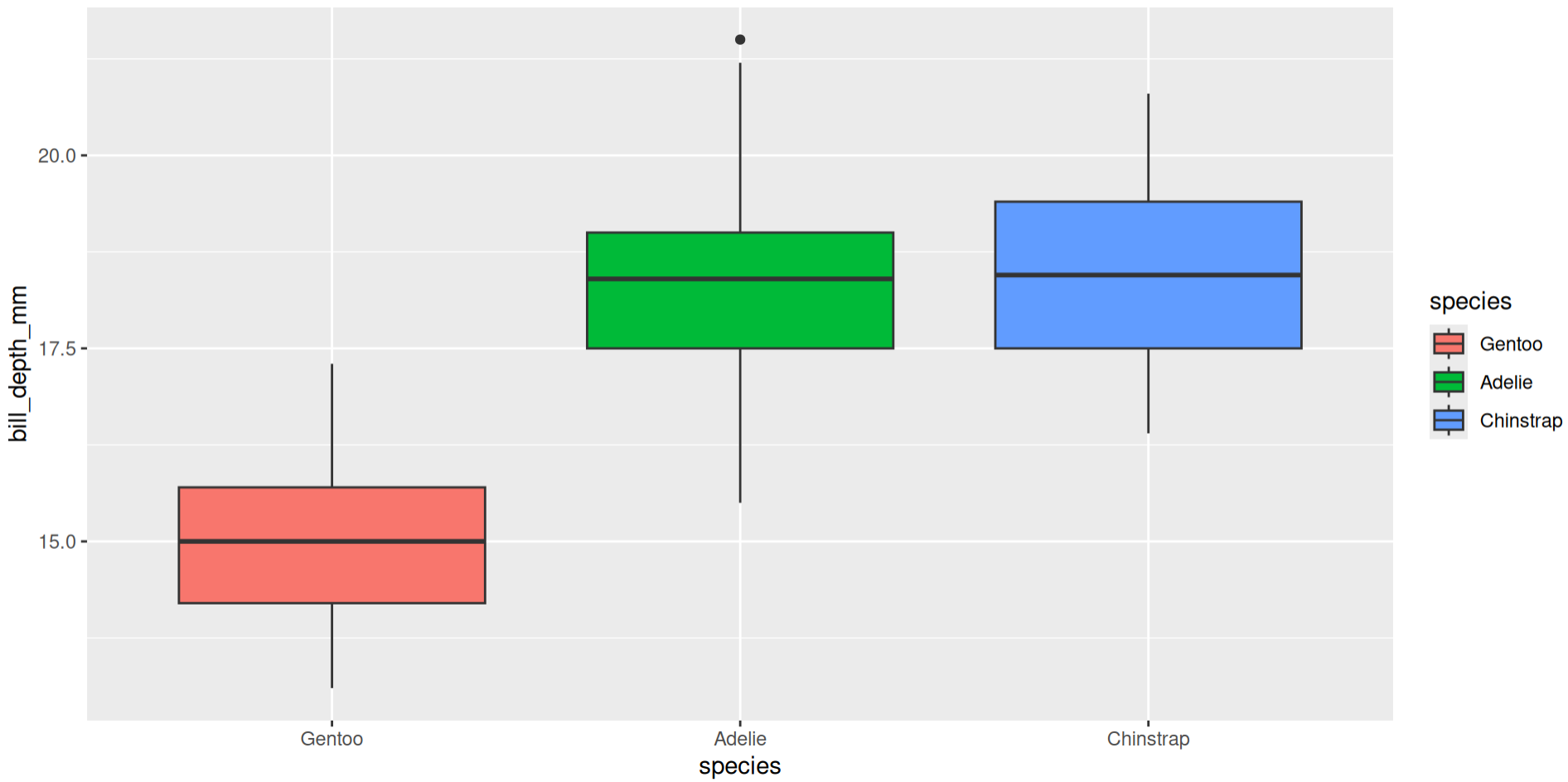
```r
penguins |>
  mutate(
    year_factor = factor(year),
    species = fct_infreq(species),
    species = fct_rev(species)
  ) |>
  ggplot(aes(x = species, fill = year_factor)) +
  geom_bar()
```



```
penguins |>
  ggplot(aes(x = species, y = bill_depth_mm, fill = species)) +
  geom_boxplot()
```



```r
penguins |>
  mutate(
    species = fct_reorder(species, bill_depth_mm)
  ) |>
  ggplot(aes(x = species, y = bill_depth_mm, fill = species)) +
  geom_boxplot()
```




STAR WARS
STAR WARS
freevector.com

```r
starwars |> count(species, sort = TRUE)
```

```
# A tibble: 38 × 2
   species      n
   <chr>    <int>
 1 Human       35
 2 Droid        6
 3 <NA>         4
 4 Gungan       3
 5 Kaminoan     2
 6 Mirialan     2
 7 Twi'lek      2
 8 Wookiee      2
 9 Zabrak       2
10 Aleena       1
# ℹ 28 more rows
```

```r
starwars |>
  mutate(species = fct_lump(species, n = 2)) |>
  count(species)
```

```
# A tibble: 4 × 2
  species     n
  <fct>   <int>
1 Droid       6
2 Human      35
3 Other      42
4 <NA>        4
```

```r
starwars |>
  mutate(
    species = fct_lump(species, n = 2),
    species = fct_relevel(species, "Human")
  ) |>
  count(species)
```

```
# A tibble: 4 × 2
  species     n
  <fct>   <int>
1 Human      35
2 Droid       6
3 Other      42
4 <NA>        4
```

```r
starwars |>
  mutate(
    species = fct_lump(species, n = 2),
    species = fct_relevel(species, "Human"),
    species = fct_recode(species, "Mensch" = "Human", "Anders" = "Other")
  ) |>
  count(species)
```

```
# A tibble: 4 × 2
  species     n
  <fct>   <int>
1 Mensch     35
2 Droid       6
3 Anders     42
4 <NA>        4
```

Praktikum: `{forcats}`

prak-05c-forcats-firmen.qmd

20:00

Break ☕ 🍵 🍜

10:00

Praktikum: `if_else()`, `case_when()`, `{forcats}`

prak-05d-cond-mutate-forcats.qmd

30:00

Danke! 

Slides created via revealjs and Quarto.

Access slides as PDF.