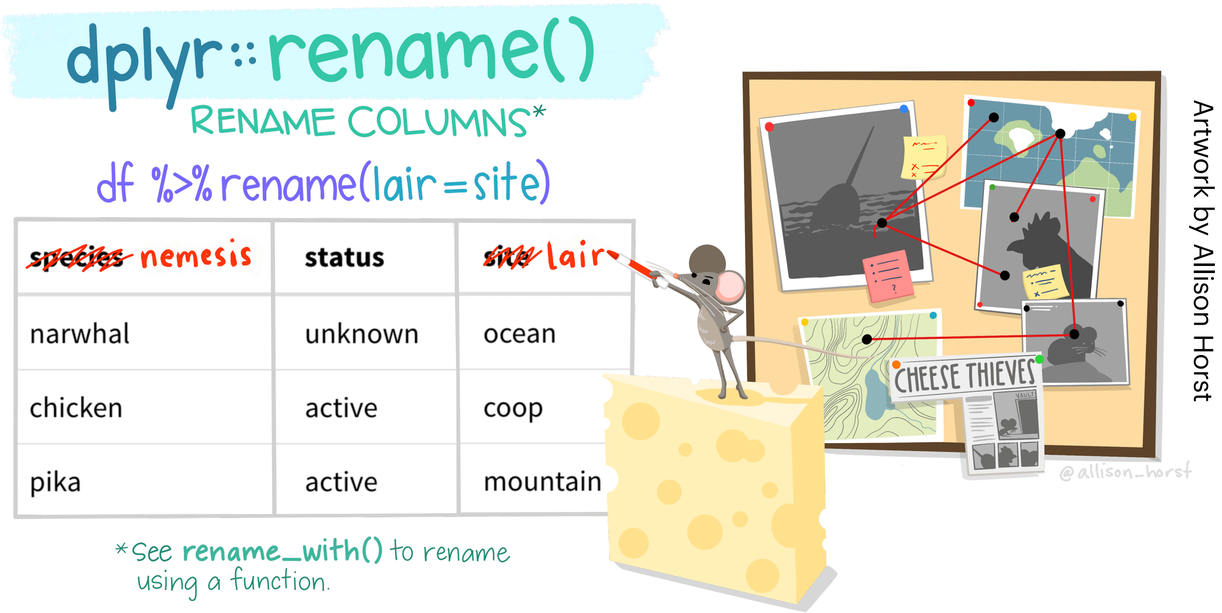
Data wrangling with `dplyr`

Week 5 (10/29/25)



Artwork by Allison Horst

Stepfanie M. Aguillon

Outline of today's class

- Introduce more `dplyr` functions
- Continue practicing with `ggplot2`

Continuing with `dplyr`

`tidyverse` package
for manipulating data

Review of last week's `dplyr` functions

1. `filter()`

filter observations by their values

2. `select()`

select variables by their column names

3. `mutate()`

mutate existing variables to create new ones

4. `arrange()`

arrange observations by their values

dplyr Cheatsheet

Data transformation with dplyr : : **CHEATSHEET**

dplyr functions work with pipes and expect **tidy data**. In tidy data:

Each **variable** is in its own **column**

Each **observation**, or **case**, is in its own **row**

pipes

x |> f(y) becomes **f(x, y)**

Summarize Cases

Apply **summary functions** to columns to create a new table of summary statistics. Summary functions take vectors as input and return one value (see back).

summary function

summarize(.data, …**)** Compute table of summaries.
mtcars |> summarize(avg = mean(mpg))

count(.data, …, wt = NULL, sort = FALSE, name = NULL**)** Count number of rows in each group defined by the variables in … Also **tally()**, **add_count()**, **add_tally()**.
mtcars |> count(cyl)

Group Cases

Use **group_by(**.data, …, .add = FALSE, .drop = TRUE**)** to create a "grouped" copy of a table grouped by columns in ... dplyr functions will manipulate each "group" separately and combine the results.

```
mtcars |>
  group_by(cyl) |>
  summarize(avg = mean(mpg))
```

Alternate grouping syntax with **.by** as an argument:

```
mtcars |>
  summarize(
    avg = mean(mpg), .by = cyl
  )
```

Use **rowwise(**.data, …**)** to group data into individual rows. dplyr functions will compute results for each row. Also apply functions to list-columns. See tidyr cheat sheet for list-column workflow.

```
starwars |>
  rowwise() |>
  mutate(film_count = length(films))
```

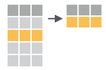
ungroup(x, …**)** Returns ungrouped copy of table.

```
g_mtcars <- mtcars |> group_by(cyl)
ungroup(g_mtcars)
```

Manipulate Cases

EXTRACT CASES

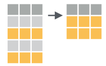
Row functions return a subset of rows as a new table.



filter(.data, …, .preserve = FALSE**)** Extract rows that meet logical criteria.
mtcars |> filter(mpg > 20)

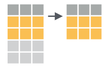
distinct(.data, …, .keep_all = FALSE**)** Remove rows with duplicate values.
mtcars |> distinct(gear)

slice(.data, …, .preserve = FALSE**)** Select rows by position.
mtcars |> slice(10:15)



slice_sample(.data, …, n, prop, weight_by = NULL, replace = FALSE**)** Randomly select rows. Use n to select a number of rows and prop to select a fraction of rows.
mtcars |> slice_sample(n = 5, replace = TRUE)

slice_min(.data, order_by, …, n, prop, with_ties = TRUE**)** and **slice_max()** Select rows with the lowest and highest values.
mtcars |> slice_min(mpg, prop = 0.25)



slice_head(.data, …, n, prop**)** and **slice_tail()** Select the first or last rows.
mtcars |> slice_head(n = 5)

Logical and boolean operators to use with filter()

==	<	<=	is.na()	%in%	|	xor()
!=	>	>=	!is.na()	!	&	

See **?base::Logic** and **?Comparison** for help.

ARRANGE CASES

arrange(.data, …, .by_group = FALSE**)** Order rows by values of a column or columns (low to high), use with **desc()** to order from high to low.
mtcars |> arrange(mpg)
mtcars |> arrange(desc(mpg))

ADD CASES

add_row(.data, …, .before = NULL, .after = NULL**)** Add one or more rows to a table.
cars |> add_row(speed = 1, dist = 1)

Manipulate Variables

EXTRACT VARIABLES

Column functions return a set of columns as a new vector or table.

pull(.data, var = -1, name = NULL, …**)** Extract column values as a vector, by name or index.
mtcars |> pull(wt)

select(.data, …**)** Extract columns as a table.
mtcars |> select(mpg, wt)

relocate(.data, …, .before = NULL, .after = NULL**)** Move columns to new position.
mtcars |> relocate(mpg, cyl, .after = last_col())

Use these helpers with select() and across()
e.g. mtcars |> select(mpg:cyl)

contains(match**)**	**num_range(**prefix, range**)**	**:**, e.g., mpg:cyl
ends_with(match**)**	**all_of(**x**)/any_of(**x, …, vars**)**	**!**, e.g., !gear
starts_with(match**)**	**matches(**match**)**	**everything()**

MANIPULATE MULTIPLE VARIABLES AT ONCE

```
df <- tibble(x_1 = c(1, 2), x_2 = c(3, 4), y = c(4, 5))
```

across(.cols, .funs, …, .names = NULL**)** Summarize or mutate multiple columns in the same way.
df |> summarize(across(everything(), mean))

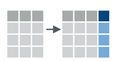
c_across(.cols**)** Compute across columns in row-wise data.

```
df |>
  rowwise() |>
  mutate(x_total = sum(c_across(1:2)))
```

MAKE NEW VARIABLES

Apply **vectorized functions** to columns. Vectorized functions take vectors as input and return vectors of the same length as output (see back).

vectorized function



mutate(.data, …, .keep = "all", .before = NULL, .after = NULL**)** Compute new column(s). Also **add_column()**.
mtcars |> mutate(gpm = 1 / mpg)
mtcars |> mutate(gpm = 1 / mpg, .keep = "none")

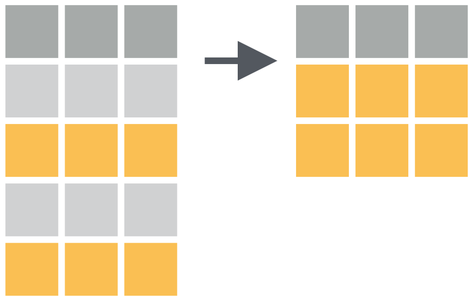
rename(.data, …**)** Rename columns. Use **rename_with()** to rename with a function.
mtcars |> rename(miles_per_gallon = mpg)

https://rstudio.github.io/cheatsheets/data-transformation.pdf

Introducing four more `dplyr` functions

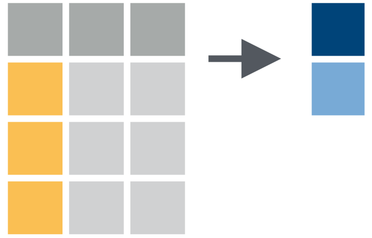
5 `rename()`

6 `slice_sample()`



7 `group_by()`

8 `summarize()`



Introducing four more `dplyr` functions

5 `rename()`

rename columns

6 `slice_sample()`

randomly choose ("**slice**") rows

7 `group_by()`

group your data by specific variables

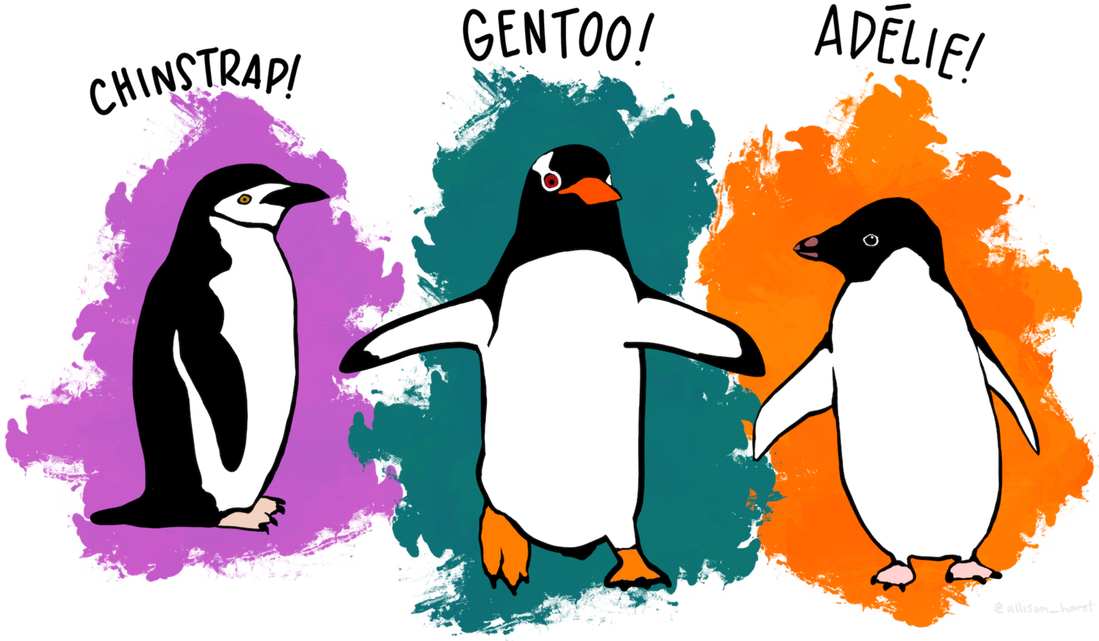
8 `summarize()`

generate **summary** statistics within groups

Practice, practice, practice...

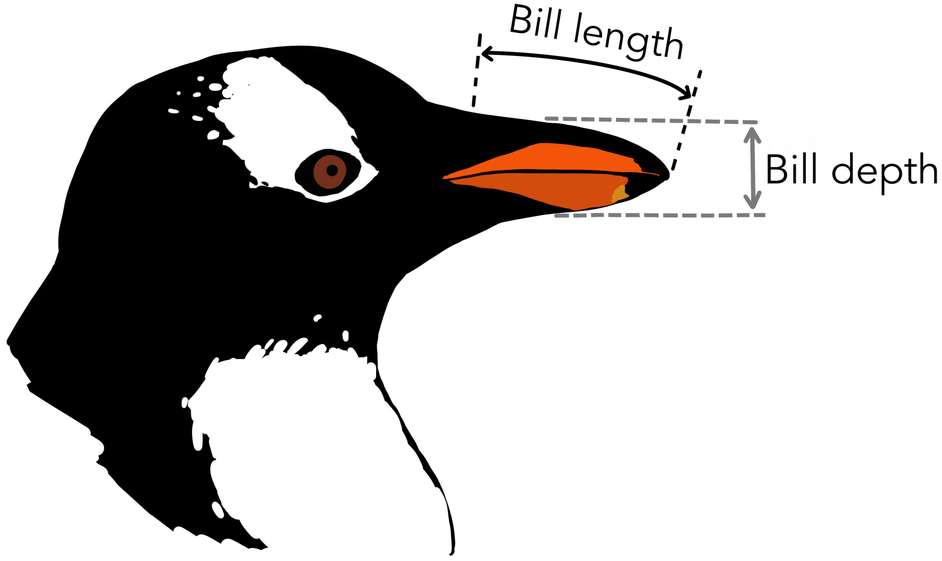
The `penguins` dataset

3 penguin species



Artwork by Allison Horst

Morphological measures



https://allisonhorst.github.io/palmerpenguins/

The `penguins` dataset

Snippet of the dataset:

species	island	bill_len	bill_dep	flipper_len	body_mass	sex	year
Adelie	Torgersen	39.1	18.7	181	3750	male	2007
Adelie	Torgersen	39.5	17.4	186	3800	female	2007
Adelie	Torgersen	40.3	18.0	195	3250	female	2007
Adelie	Torgersen	*NA*	*NA*	*NA*	*NA*	*NA*	2007
Adelie	Torgersen	36.7	19.3	193	3450	female	2007

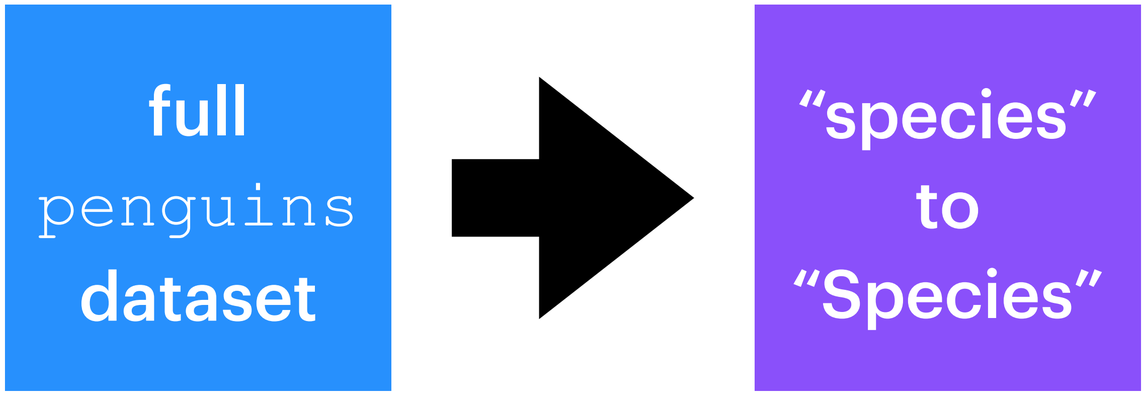
https://allisonhorst.github.io/palmerpenguins/

`rename()`

rename columns

```
penguins2 <- penguins %>%
  rename(Species = species)
```

Goal:

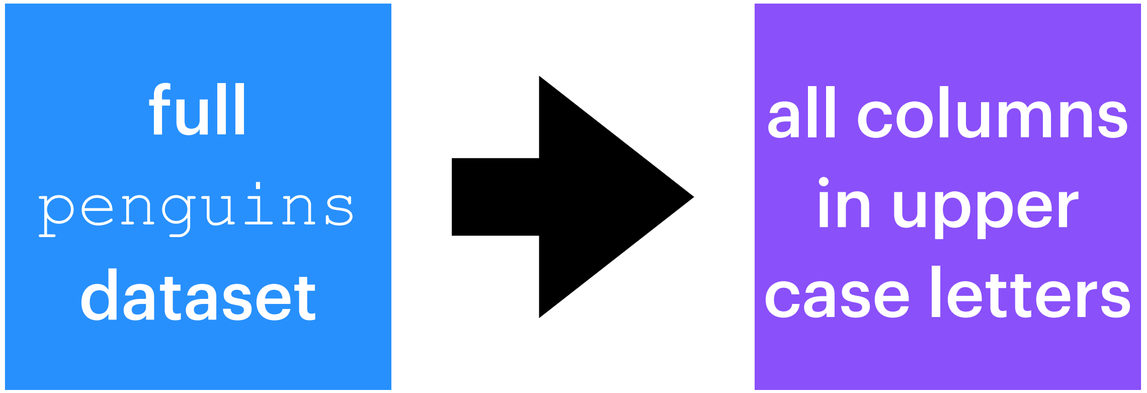


`rename_with()` 

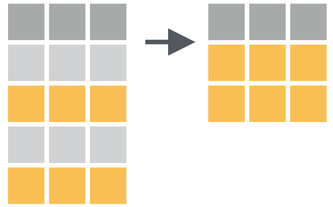
rename columns *with* a function

```
penguins2 <- penguins %>%
  rename_with(toupper)
```

Goal:



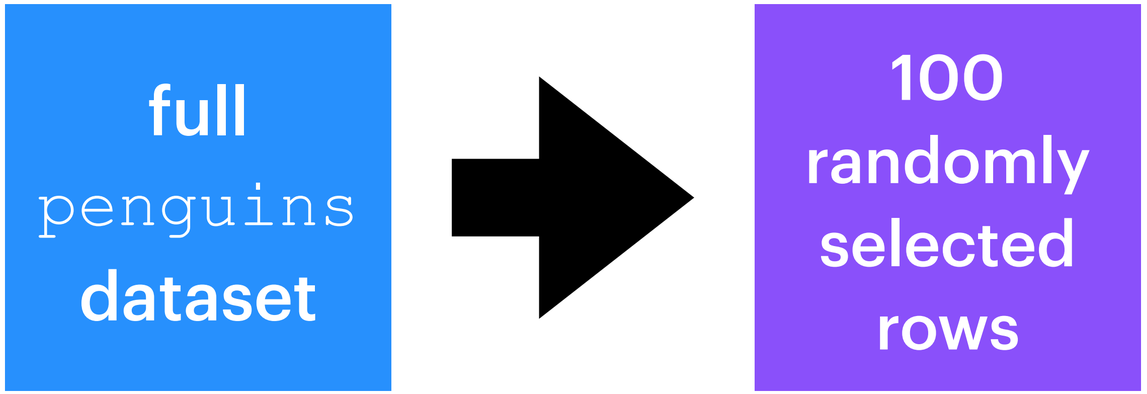
`slice_sample()`



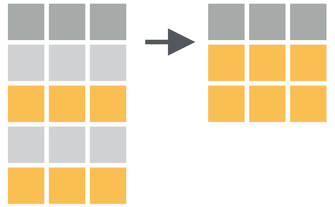
randomly choose (**“slice”**) rows

```
new_penguins <- penguins %>%
  slice_sample(n = 100)
```

Goal:



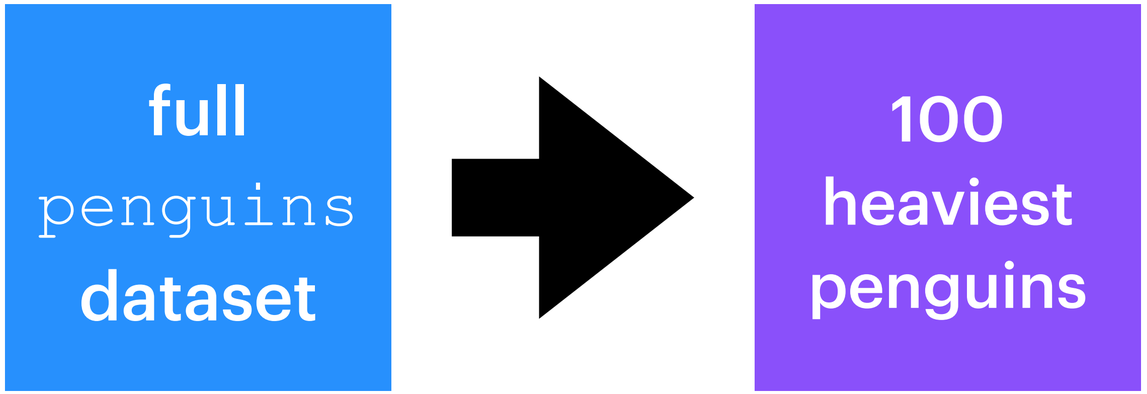
`slice_min()`, `slice_max()`



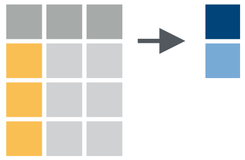
slice rows with the smallest or largest values of a variable

```
max_penguins <- penguins %>%
  slice_max(body_mass, n = 100)
```

Goal:



group_by() + summarize()



group your data by specific variables and generate **summary** statistics within groups

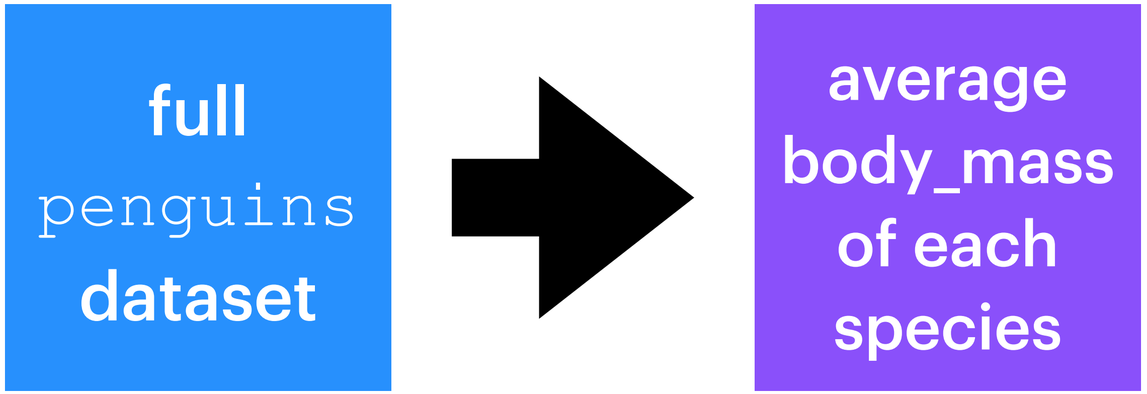
```
summary_df <- penguins %>%
  group_by(species) %>%
  summarize(avg_body_mass = mean(body_mass))
```

what variable(s) for your "group"

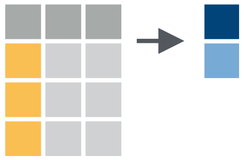
new summary variable

the calculation

Goal:



group_by() + summarize()



group your data by specific variables and generate **summary** statistics within groups

```
summary_df <- df %>%
 group_by(<VARIABLES>) %>%
 summarize(<SUMMARY VARIABLE> = <EQUATION>)
```

Useful functions in `summarize()`

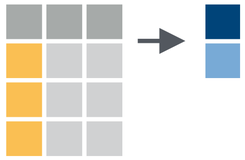
- Center: **`mean()`, `median()`**
- Spread: **`sd()`**, `IQR()`, `mad()`, `quantile()`
- Range: **`min()`, `max()`**
- Position: `first()`, `last()`, `nth()`
- Count: **`n()`**, `n_distinct()`
- Logical: **`any()`**, `all()`
- Missing data: **`is.na()`**, **`na.omit()`**
- Summation: **`sum()`**

https://dplyr.tidyverse.org/reference/summarise.html

Useful functions in `summarize()`

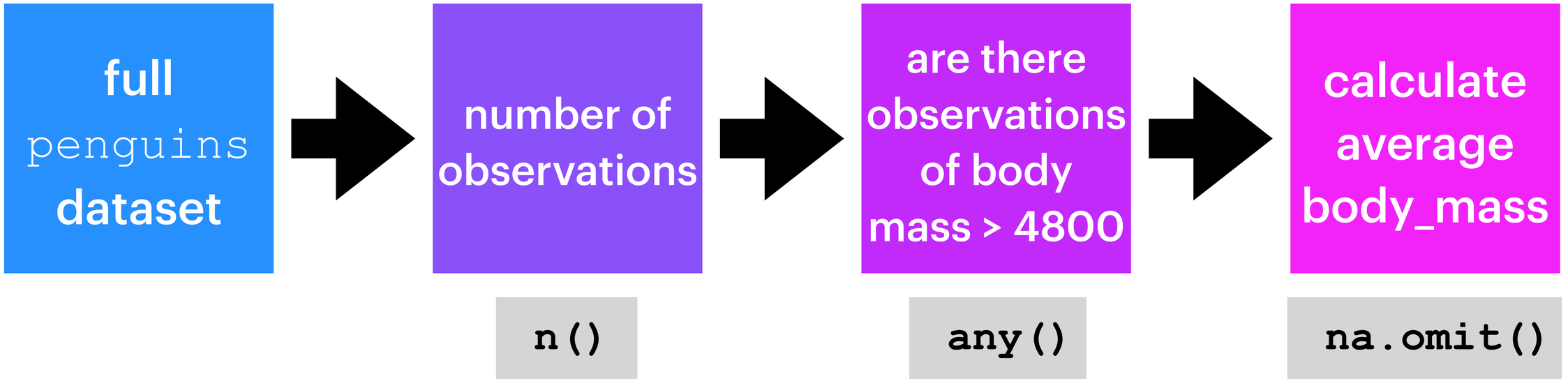
`n()`	outputs the group's size
`na.omit()`	removes NAs from your list/variable
`any()`	outputs a TRUE/FALSE based on a provided statement

`group_by()` + `summarize()`

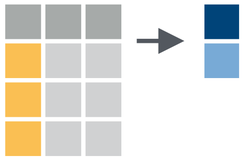


group your data by specific variables and generate **summary** statistics within groups

Goal:



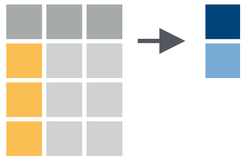
group_by() + summarize()



group your data by specific variables and generate **summary** statistics within groups

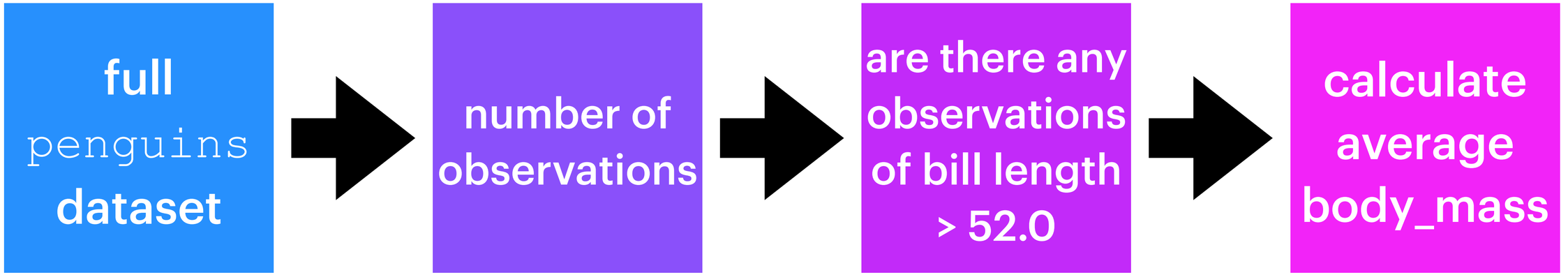
```
summary_df <- df %>%
 group_by(<VARIABLES>) %>%
 summarize(<SUMMARY VARIABLE> = <EQUATION>)
```

group_by() + summarize()

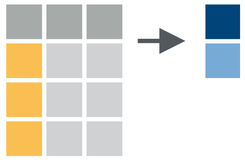


```
summary_df <- penguins %>%
 group_by(species) %>%
 summarize(count = n(),
  big_penguins = any(bill_len > 52.0),
  avg_body_mass = mean(na.omit(body_mass)))
```

Goal:



group_by() + summarize()

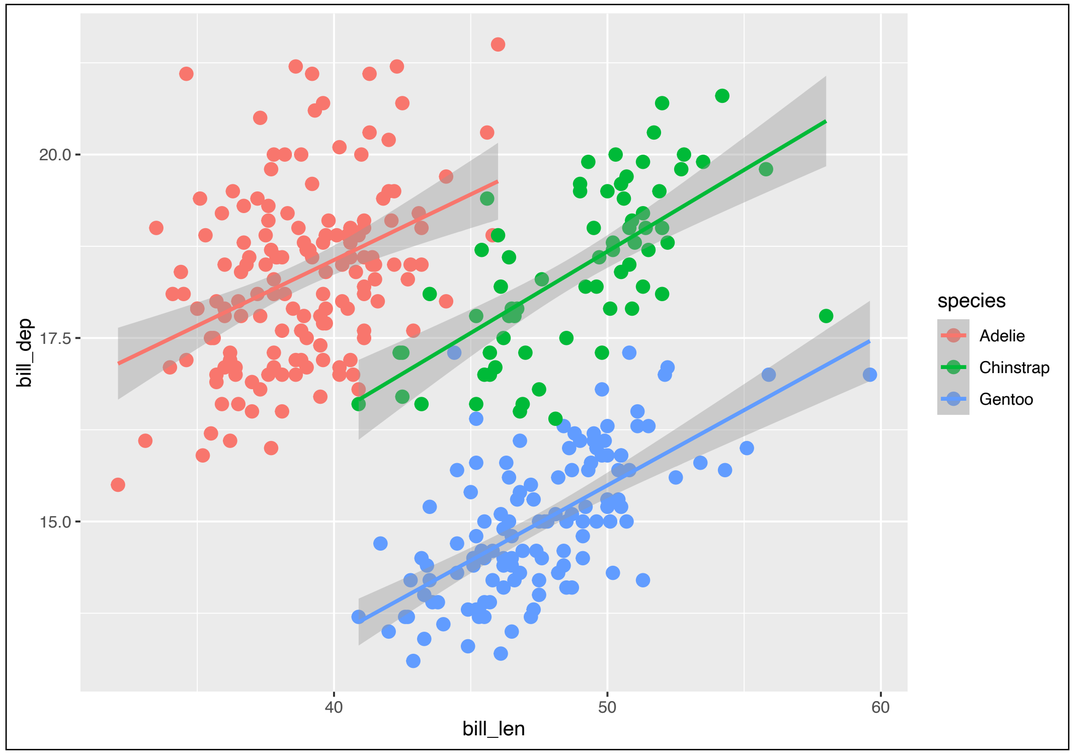


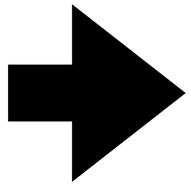
```
summary_df <- penguins %>%
 group_by(species, sex) %>%
 summarize(count = n(),
  big_penguins = any(bill_len > 52.0),
  avg_body_mass = mean(na.omit(body_mass)))
```

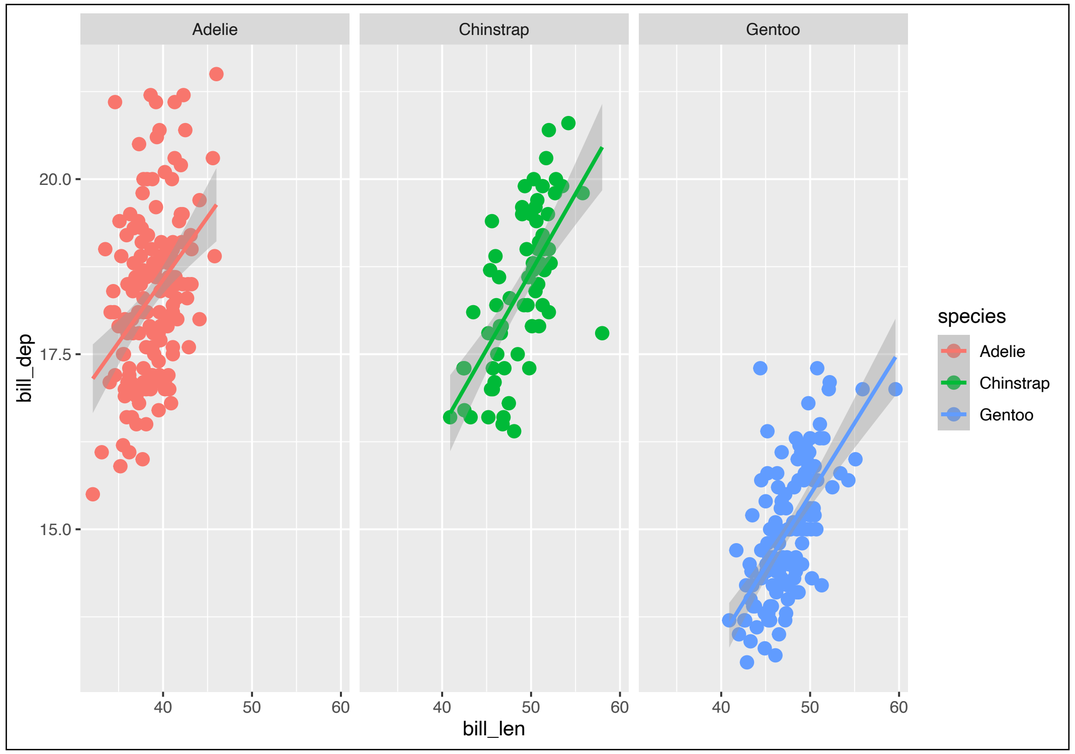
A few new `ggplot2` concepts

Plotting with `facet_wrap()`

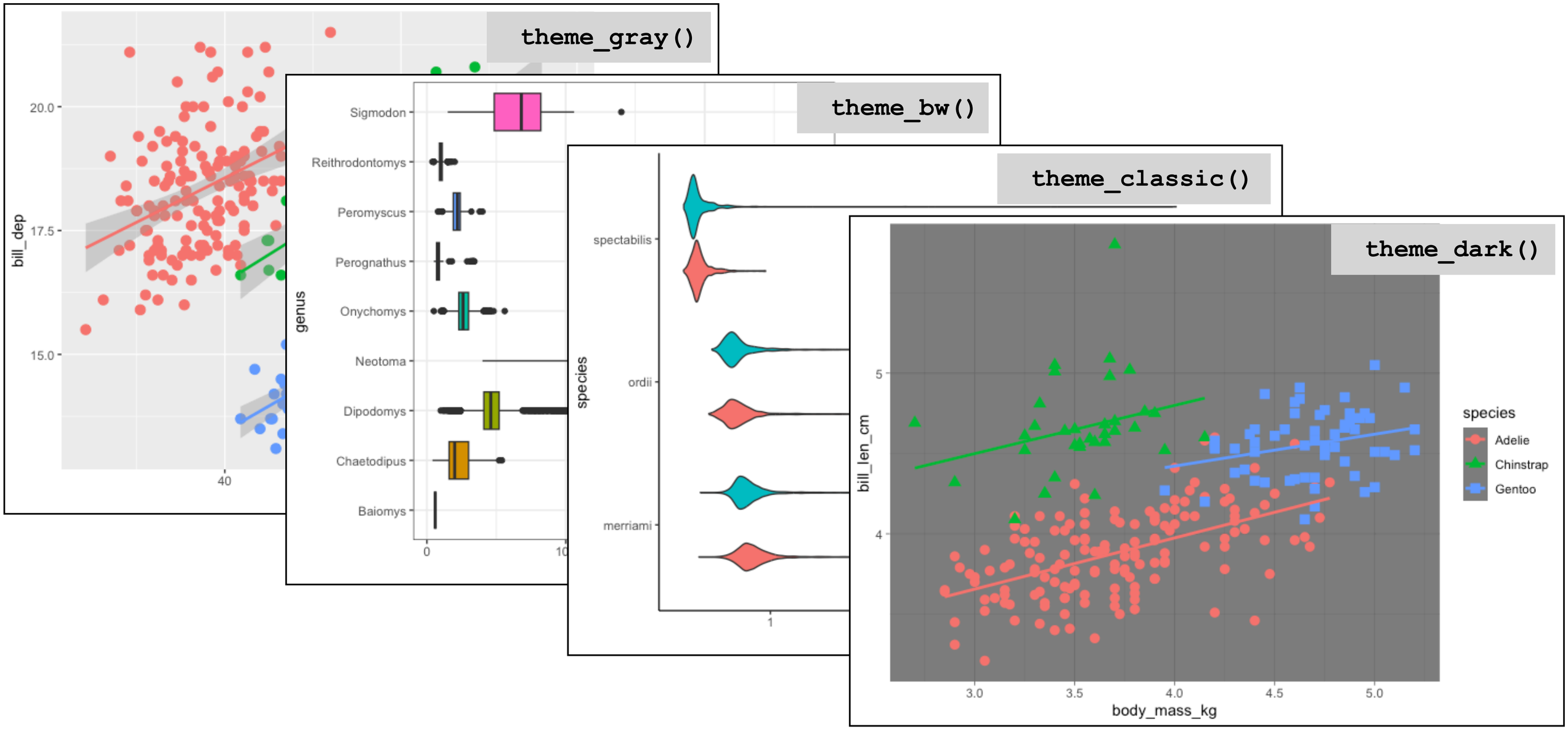
```
ggplot(data = penguins, mapping = aes(x = bill_len, y = bill_dep, color = species)) +
  geom_point(size = 3) +
  geom_smooth(method = "lm") +
  facet_wrap(~ species)
```







"Complete" themes in ggplot

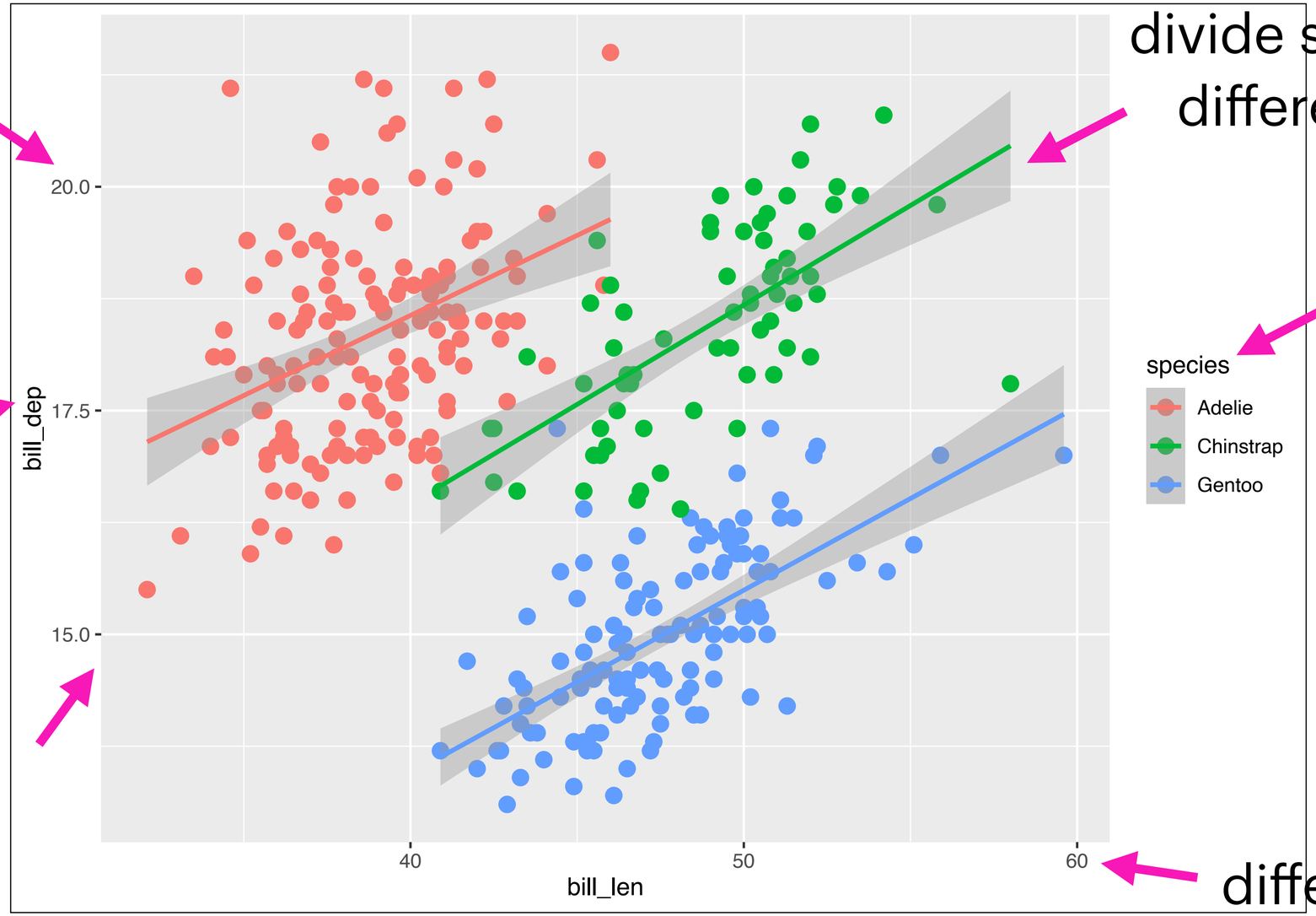


What about modifying other theme aspects?

different axis limits?

different axis labels?

different distribution of axis tick marks?



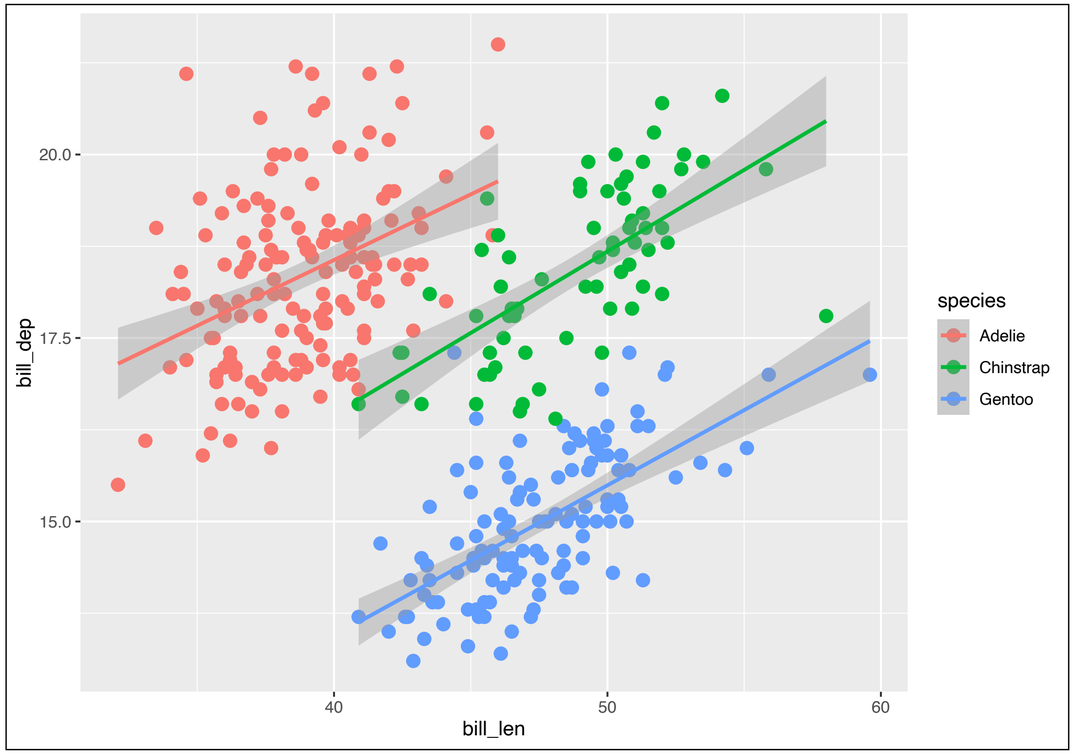
divide species into different plots?

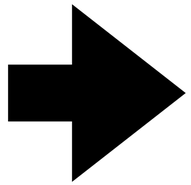
different legend title?

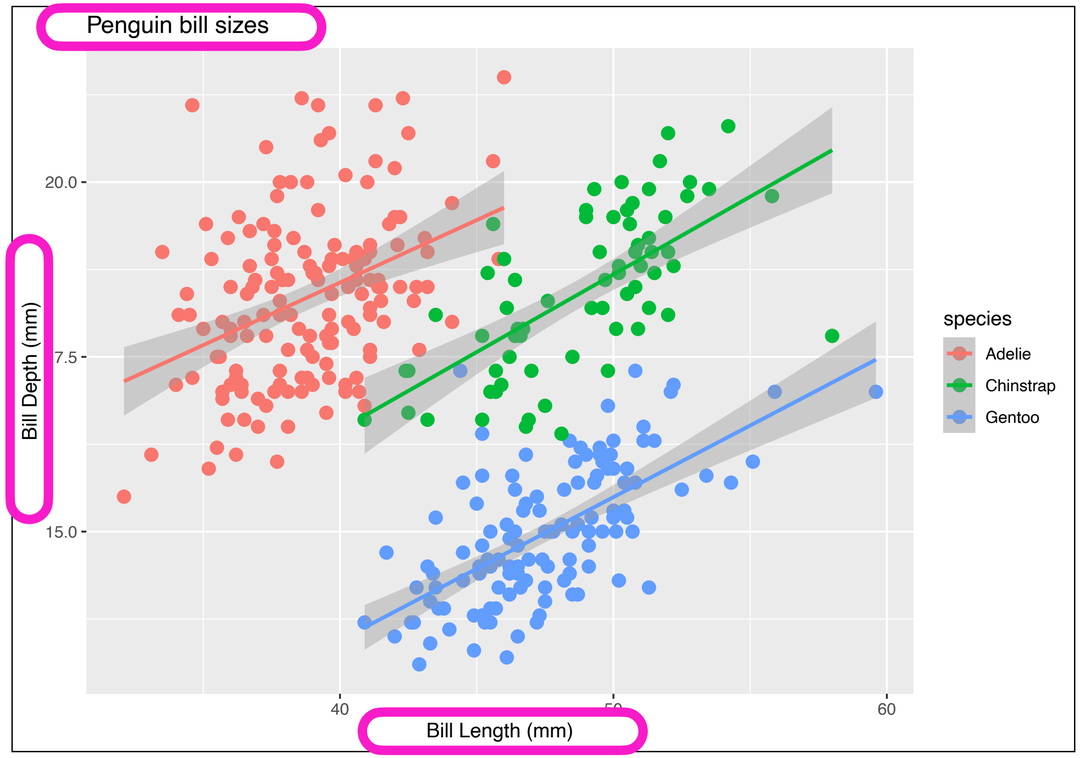
different font size?

Modifying theme components with `labs()`

```
ggplot(data = penguins, mapping = aes(x = bill_len, y = bill_dep, color = species)) +
  geom_point(size = 3) +
  geom_smooth(method = "lm") +
  labs(x = "Bill Length (mm)", y = "Bill Depth (mm)", title = "Penguin bill sizes")
```







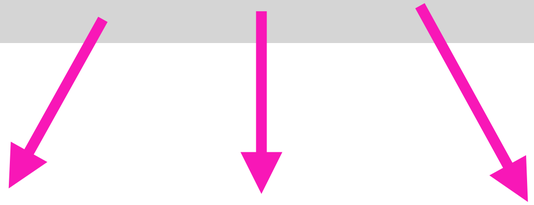
Modifying other `theme()` components

```
theme(
  ...,
  line,
  rect,
  text,
  title,
  point,
  polygon,
  geom,
  spacing,
  margins,
  aspect.ratio,
  axis.title,
  axis.title.x,
  axis.title.x.top,
  axis.title.x.bottom,
  axis.title.y,
  axis.title.y.left,
  axis.title.y.right,
  axis.text,
  axis.text.x,
  axis.text.x.top,
  axis.text.x.bottom,
  axis.text.y,
  axis.text.y.left,
  axis.text.y.right,
  axis.text.theta,
  axis.text.r,
  axis.ticks,
  axis.ticks.x,
  axis.ticks.x.top,
  axis.ticks.x.bottom,
  axis.ticks.y,
  axis.ticks.y.left,
  axis.ticks.y.right,
  axis.ticks.theta,
  axis.ticks.r,
  axis.minor.ticks.x.top,
  axis.minor.ticks.x.bottom,
  axis.minor.ticks.y.left,
  axis.minor.ticks.y.right,
  axis.minor.ticks.theta,
  axis.minor.ticks.r,
  axis.ticks.length,
  axis.ticks.length.x,
  axis.ticks.length.x.top,
  axis.ticks.length.x.bottom,
  axis.ticks.length.y,
  axis.ticks.length.y.left,
  axis.ticks.length.y.right,
  axis.ticks.length.theta,
  axis.ticks.length.r,
  axis.minor.ticks.length,
  axis.minor.ticks.length.x,
  axis.minor.ticks.length.x.top,
  axis.minor.ticks.length.x.bottom,
  axis.minor.ticks.length.y,
  axis.minor.ticks.length.y.left,
  axis.minor.ticks.length.y.right,
  axis.minor.ticks.length.theta,
  axis.minor.ticks.length.r,
  axis.line,
  axis.line.x,
  axis.line.x.top,
  axis.line.x.bottom,
  axis.line.y,
  axis.line.y.left,
  axis.line.y.right,
  axis.line.theta,
  axis.line.r,
  legend.background,
  legend.margin,
  legend.spacing,
  legend.spacing.x,
  legend.spacing.y,
  legend.key,
  legend.key.size,
  legend.key.height,
  legend.key.width,
  legend.key.spacing,
  legend.key.spacing.x,
  legend.key.spacing.y,
  legend.key.justification,
  legend.frame,
  legend.ticks,
  legend.ticks.length,
  legend.axis.line,
  legend.text,
  legend.text.position,
  legend.title,
  legend.title.position,
  legend.position,
  legend.position.inside,
  legend.direction,
  legend.byrow,
  legend.justification,
  legend.justification.top,
  legend.justification.bottom,
  legend.justification.left,
  legend.justification.right,
  legend.justification.inside,
  legend.location,
  legend.box,
  legend.box.just,
  legend.box.margin,
  legend.box.background,
  legend.box.spacing,
  panel.background,
  panel.border,
  panel.spacing,
  panel.spacing.x,
  panel.spacing.y,
  panel.grid,
  panel.grid.major,
  panel.grid.minor,
  panel.grid.major.x,
  panel.grid.major.y,
  panel.grid.minor.x,
  panel.grid.minor.y,
  panel.ontop,
  panel.widths,
  panel.heights,
  plot.background,
  plot.title,
  plot.title.position,
  plot.subtitle,
  plot.caption,
  plot.caption.position,
  plot.tag,
  plot.tag.position,
  plot.tag.location,
  plot.margin,
  strip.background,
  strip.background.x,
  strip.background.y,
  strip.clip,
  strip.placement,
  strip.text,
  strip.text.x,
  strip.text.x.bottom,
  strip.text.x.top,
  strip.text.y,
  strip.text.y.left,
  strip.text.y.right,
  strip.switch.pad.grid,
  strip.switch.pad.wrap,
  complete = FALSE,
  validate = TRUE
)
```

https://ggplot2.tidyverse.org/reference/theme.html

Modifying other `theme()` components

```
ggplot(<...>) + geom_<...>(<...>)+
  theme(<options>)
```



`element_line()` `element_point()`

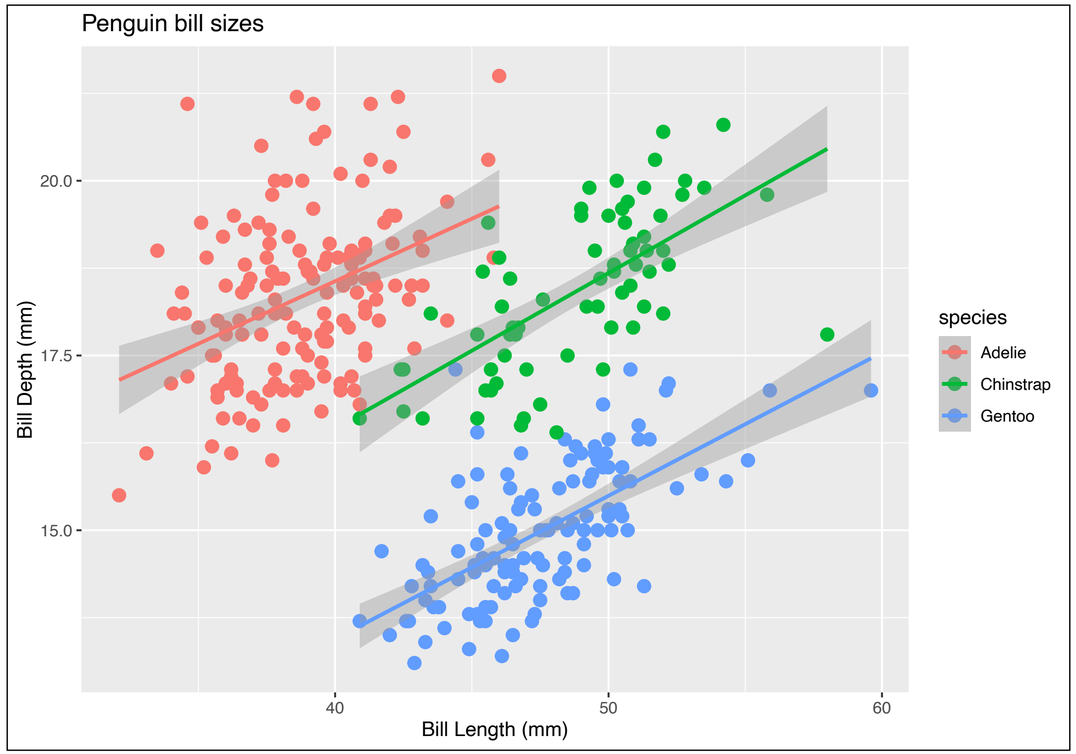
`element_rect()` `element_polygon()`

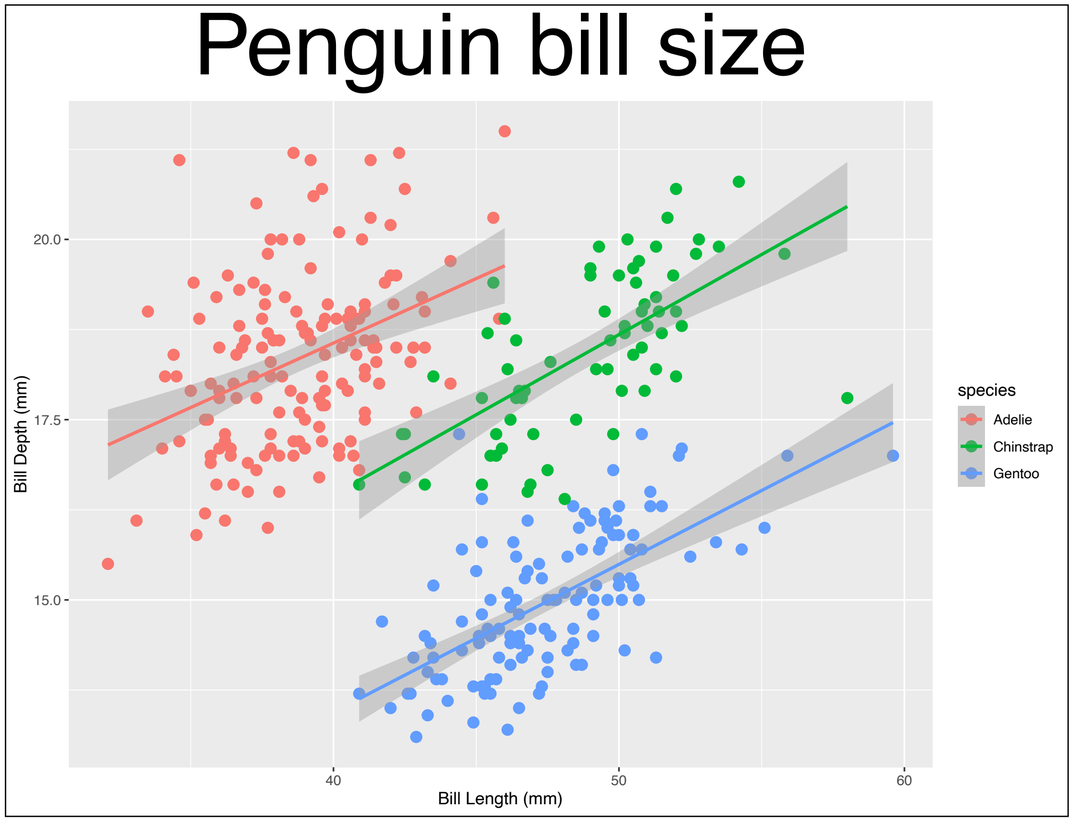
`element_text()` `element_geom()`

https://ggplot2.tidyverse.org/reference/theme.html

Modifying other `theme()` components

```
ggplot(<...>) + geom_<...>(<...>)+
  theme(plot.title = element_text(size = 55, hjust = 0.5))
```



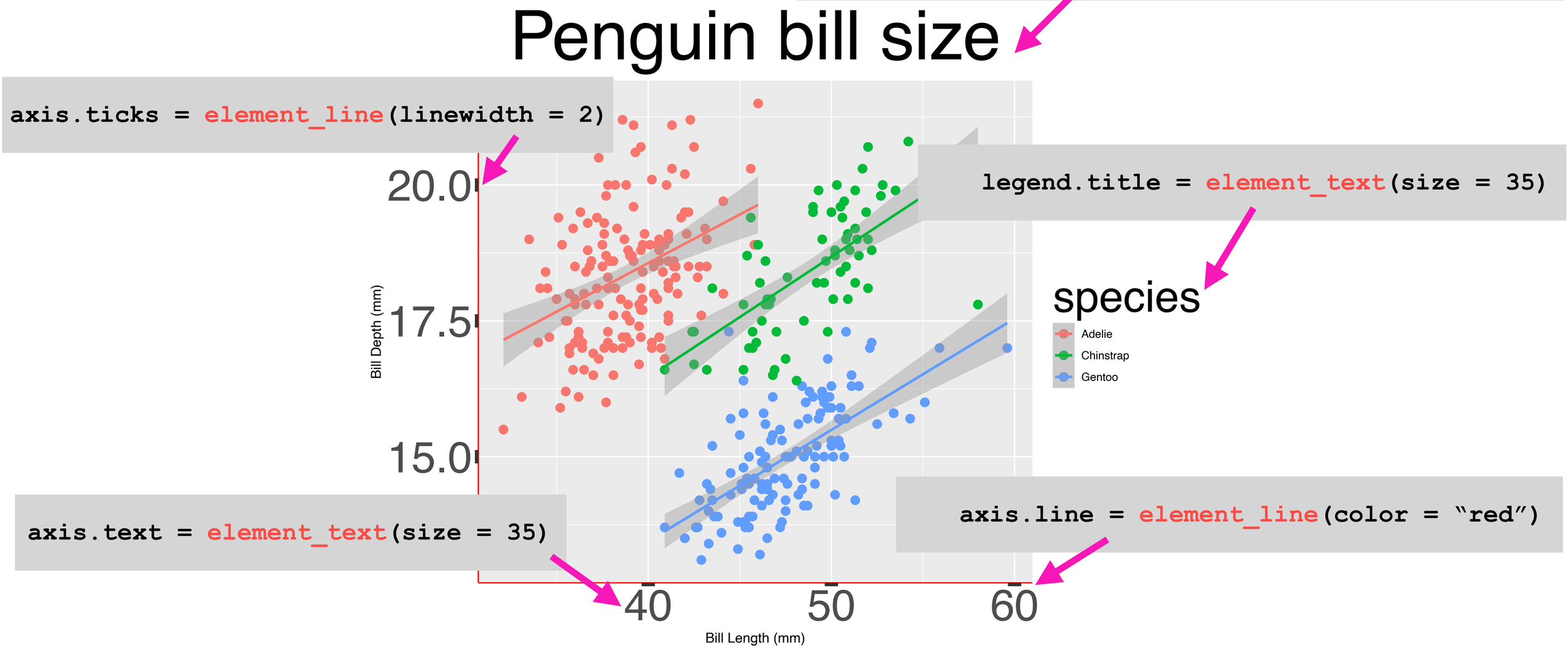


Modifying other `theme()` components

```
ggplot(<…>) + geom_<…>(<…>)+
  theme(plot.title = element_text(size = 55, hjust = 0.5),
    axis.text = element_text(size = 35),
    axis.ticks = element_line(linewidth = 2),
    axis.line = element_line(color = “red”),
    legend.title = element_text(size = 35))
```

Modifying other `theme()` components

```
plot.title = element_text(size = 55, hjust = 0.5)
```



Further reading

- Data Transformation with `dplyr` Cheatsheet
- Boehmke 2016, *Data Wrangling with R*
- Alston & Rick 2021. A Beginner's Guide to Conducting Reproducible Research. *The Bulletin of the Ecological Society of America*