ANT 6973: DATA VISUALIZATION AND EXPLORATION

COLOR, OUTPUT, AND COMPOSING MULTIPLE PLOTS

TODAY'S TOPICS

- Using color effectively
- Saving your plots and image file formats
- Composing multiple plots
- Small Project 1

USING COLOR EFFECTIVELY

TYPES OF COLOR PALETTES

- There are three main types of color palettes:
  - Qualitative
  - Sequential
  - Diverging
- Each is appropriate for particular kinds of data.

QUALITATIVE COLOR PALETTES

- Use when color shows discrete categories that *don't have intrinsic order.*

- Limitations: unlikely to work well with more than ~12 categories (3–5 ideal) and hard to make colorblind safe

ggplot2 default

`Example: scale_color_hue()`

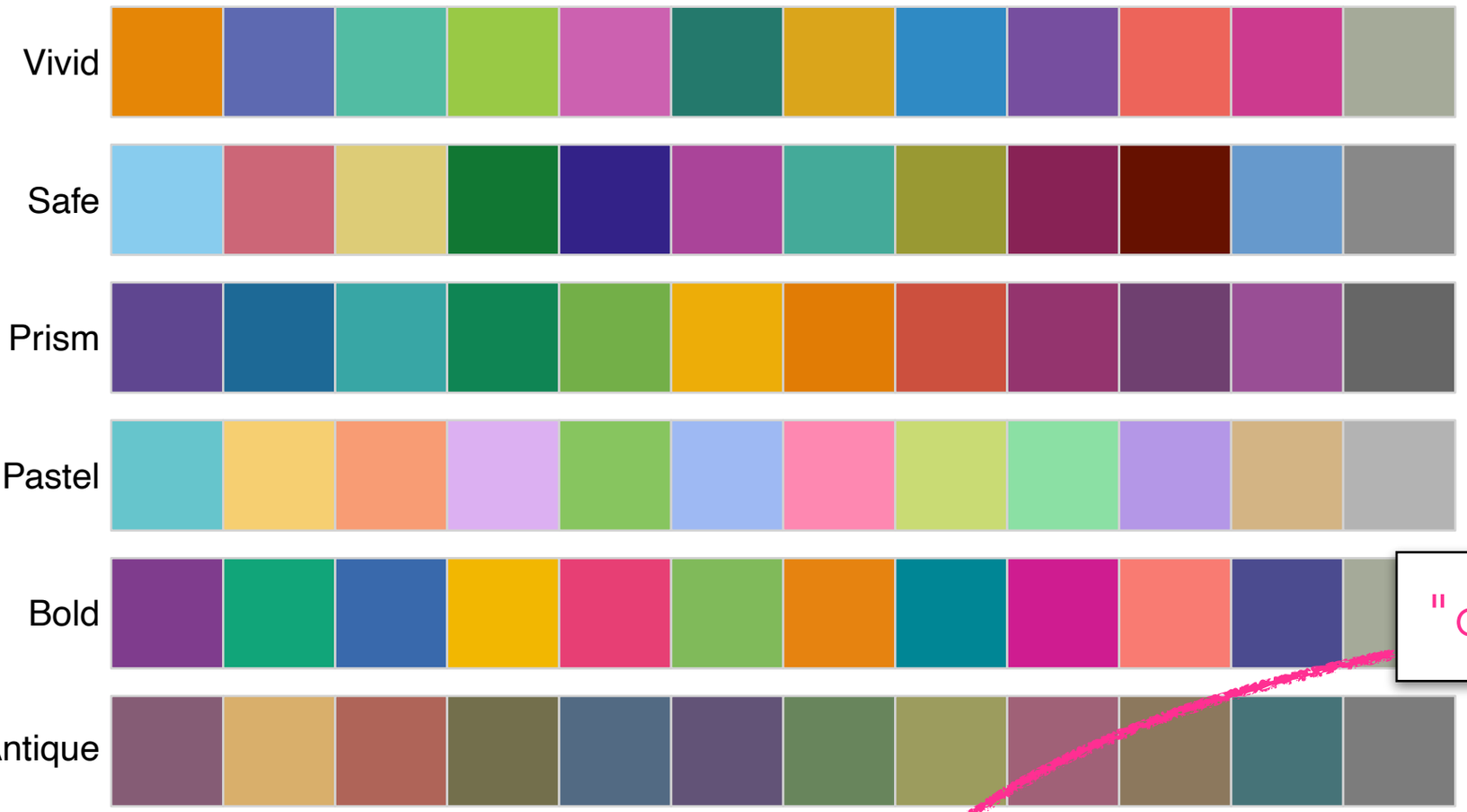
QUALITATIVE COLOR PALETTES

RColorBrewer


```
Example: scale_color_brewer(palette = "Set1")
```

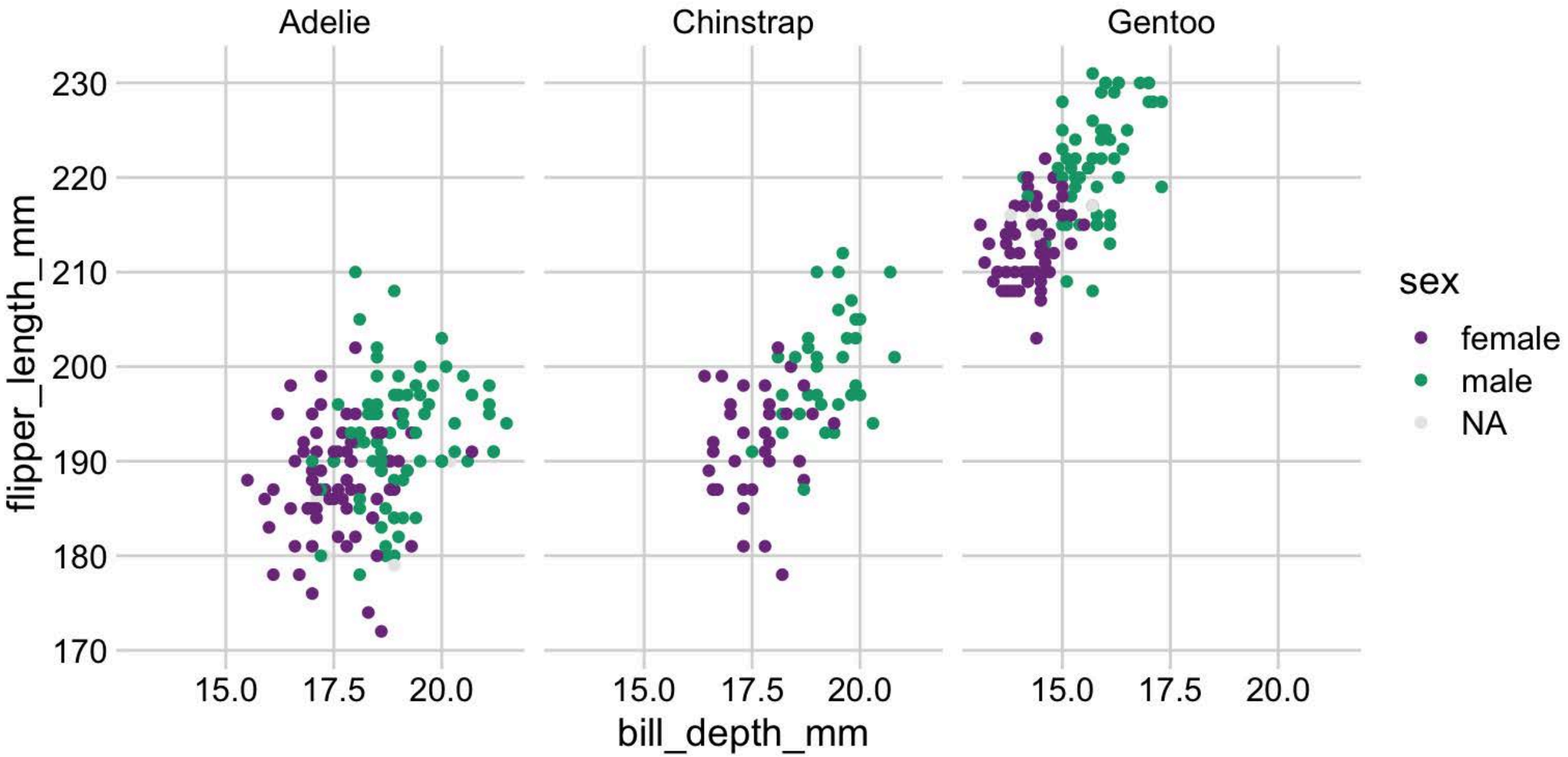
QUALITATIVE COLOR PALETTES

rcartocolor



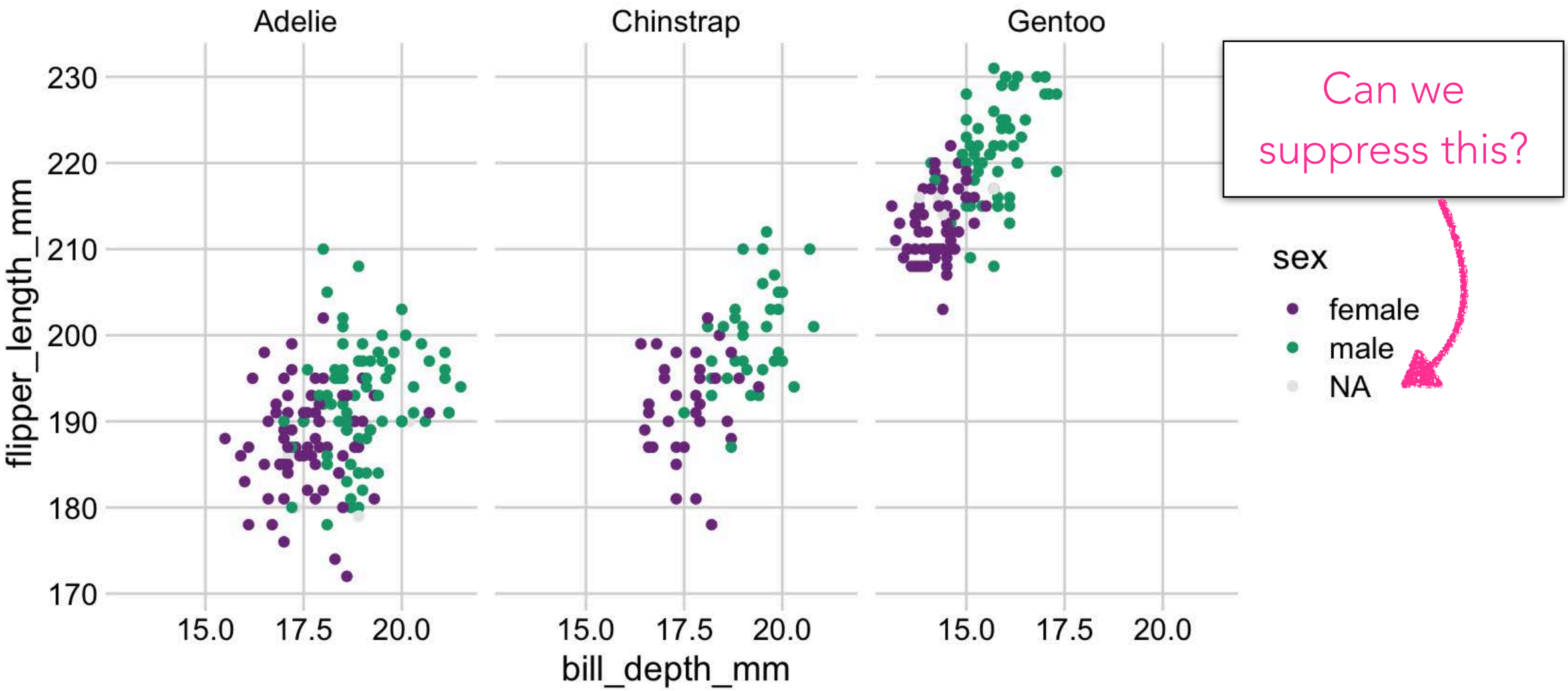

```
Example: scale_color_carto_d(palette = "Bold")
```

YOUR TURN



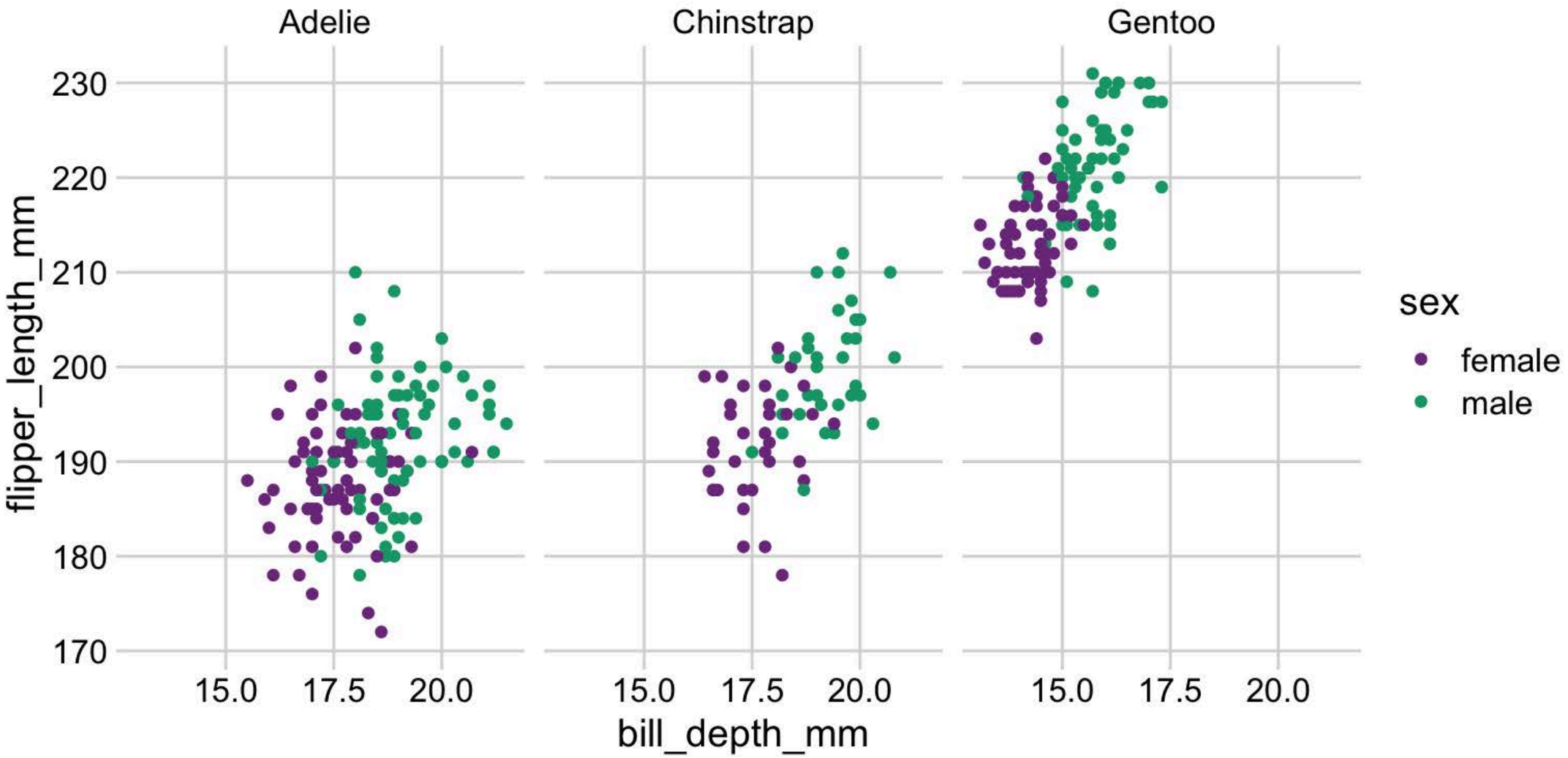
rcartocolor palette "Bold"

YOUR TURN




```
ggplot(penguins, aes(x = bill_depth_mm, y = flipper_length_mm, color = sex)) +
  geom_point() +
  scale_color_carto_d(palette = "Bold") +
  facet_wrap(vars(species))
```

YOUR TURN




```
ggplot(penguins, aes(x = bill_depth_mm, y = flipper_length_mm, color = sex)) +
  geom_point() +
  scale_color_carto_d(palette = "Bold", na.translate = FALSE) +
  facet_wrap(vars(species))
```

QUALITATIVE COLOR PALETTES

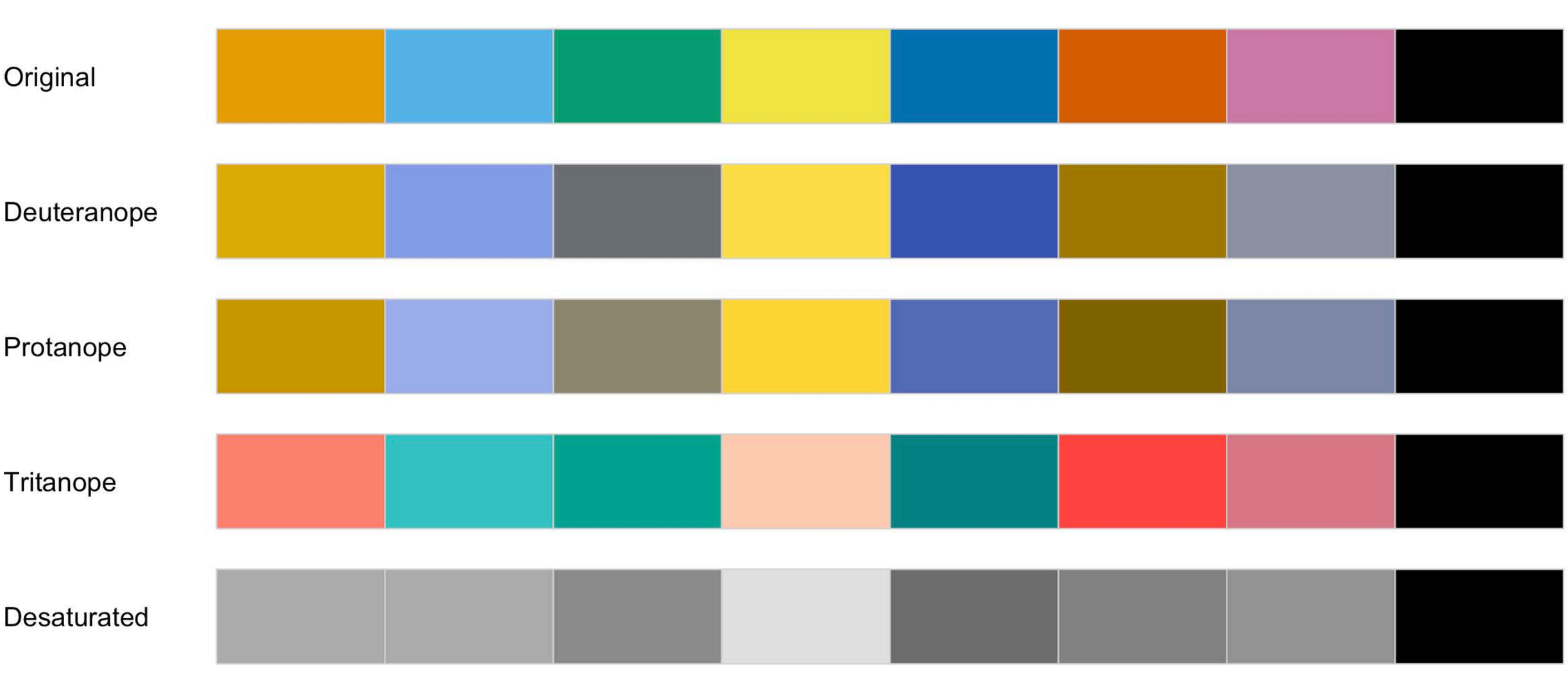
- Most qualitative color palettes are not colorblind safe
- But Ch. 19 in Wilke's book provides this palette by Okabe and Ito

```
okabe <- c("#E69F00", "#56B4E9", "#009E73", "#F0E442",
           "#0072B2", "#D55E00", "#CC79A7", "#000000")

scale_color_manual(values = okabe)
```

QUALITATIVE COLOR PALETTES

#E69F00	#56B4E9	#009E73	#F0E442	#0072B2	#D55E00	#CC79A7	#000000



SEQUENTIAL COLOR PALETTES

- Use when color represents data values that have a natural order.

- Can be based on a single hue (e.g., from light blue to dark blue) or on multiple hues (e.g., from light yellow to dark red)

SEQUENTIAL COLOR PALETTES

- Two usage cases:

SEQUENTIAL COLOR PALETTES

- Two usage cases:
  - When you have ordered categories

ggplot2 default

`Example: scale_color_ordinal()`

SEQUENTIAL COLOR PALETTES

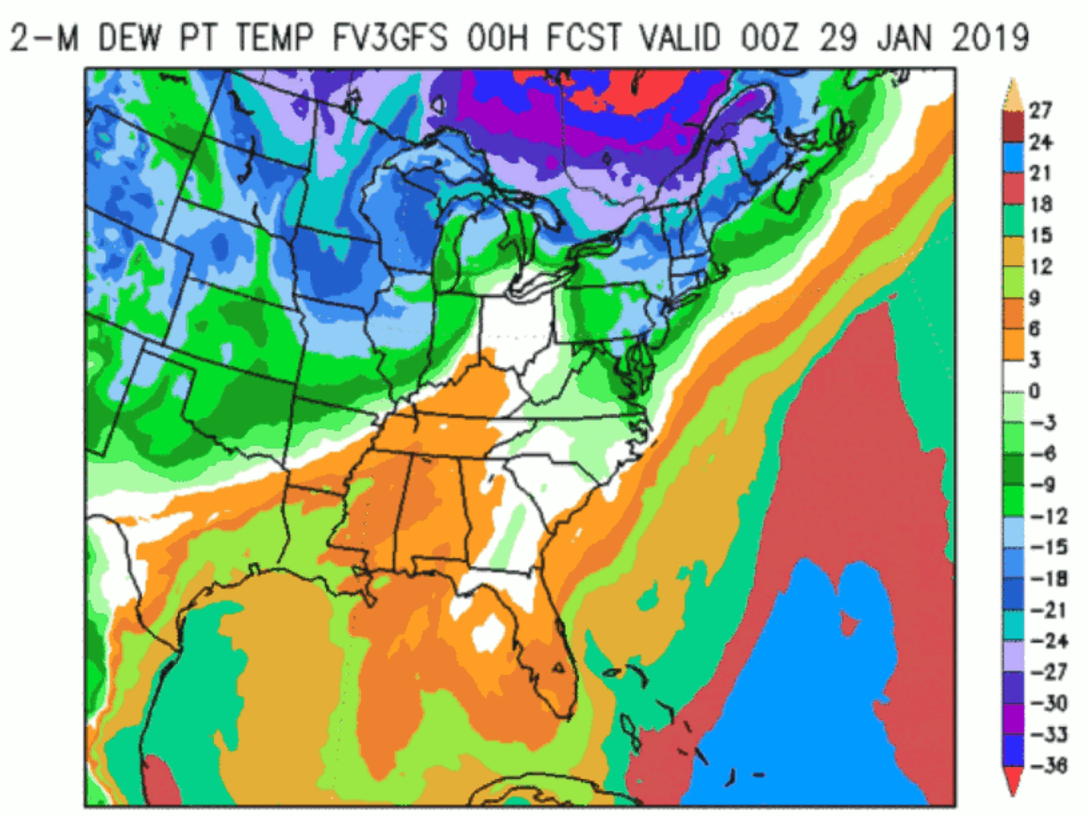
- Two usage cases:
  - When you have ordered categories
  - When you have continuously varying values

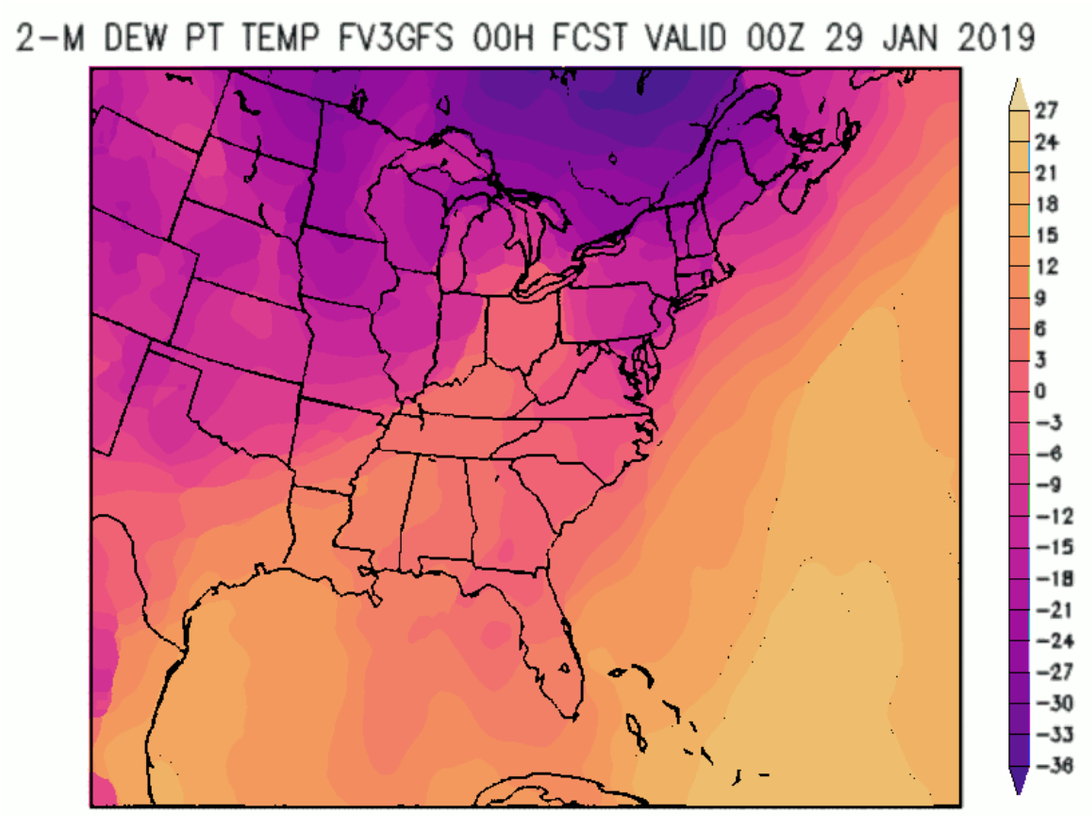
ggplot2 default

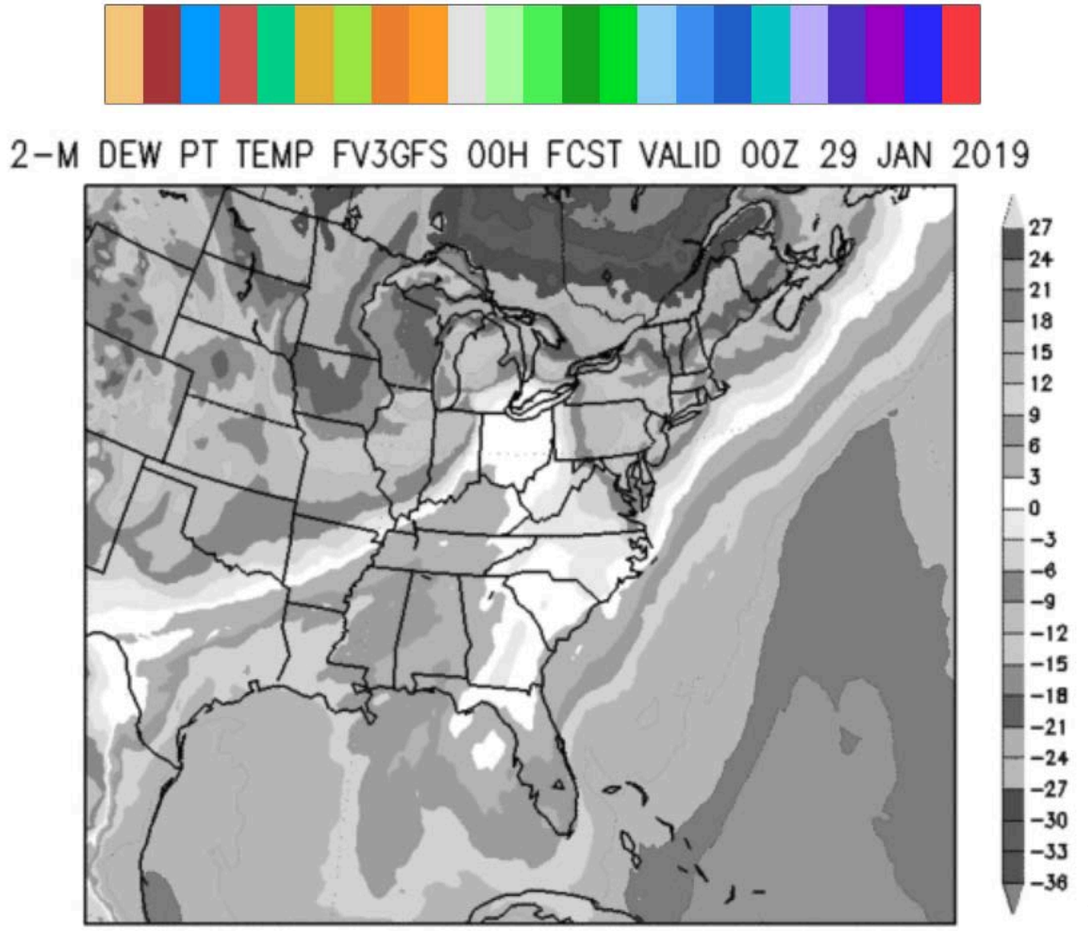
`Example: scale_color_continuous()`

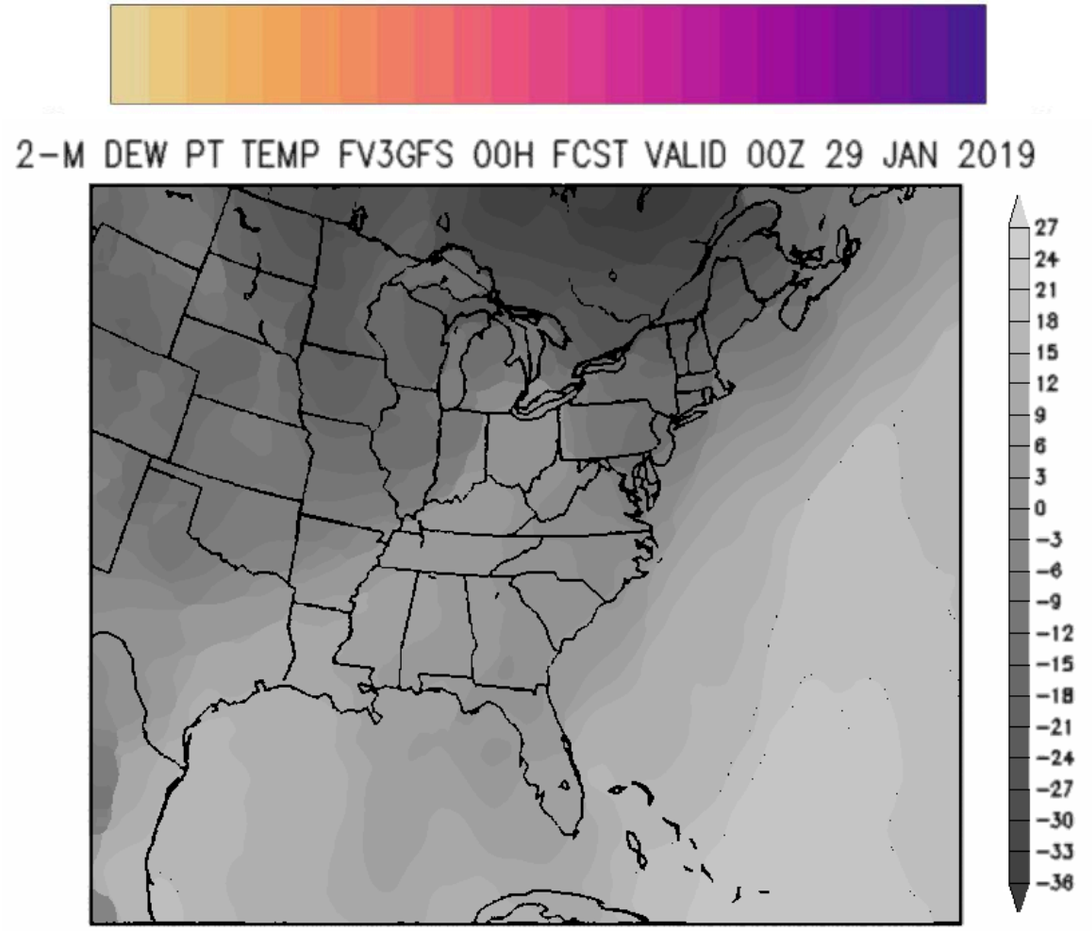
SEQUENTIAL COLOR PALETTES

- A good sequential color scale accurately conveys:
  - Which values are larger or smaller
  - How distant two values are from each other
- This means that it should be *perceptually uniform* across its entire range.
  - Ideally, smooth change from light to dark or vice versa
  - Non-circular, smooth change in lightness, no "jumps"









2-M DEW PT TEMP FV3GFS 00H FCST VALID 00Z 29 JAN 2019

2-M DEW PT TEMP FV3GFS 00H FCST VALID 00Z 29 JAN 2019

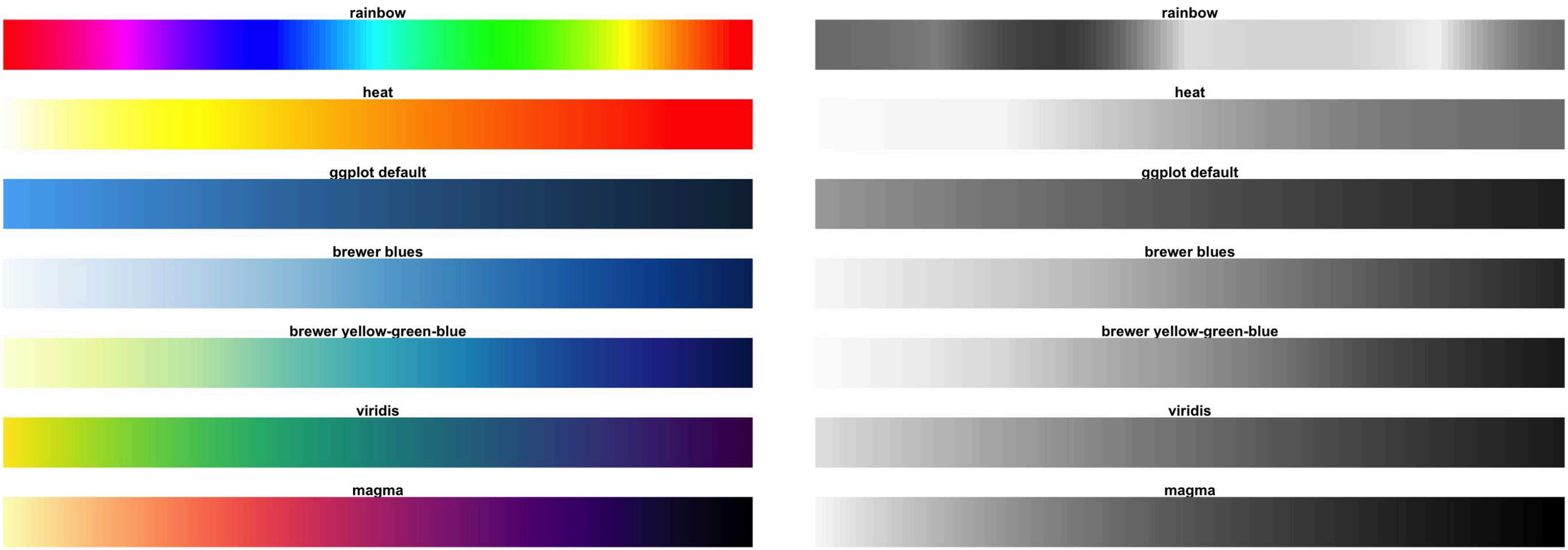
2-M DEW PT TEMP FV3GFS 00H FCST VALID 00Z 29 JAN 2019

, ~1

2-M DEW PT TEMP FV3GFS 00H FCST VALID 00Z 29 JAN 2019

SEQUENTIAL COLOR PALETTES

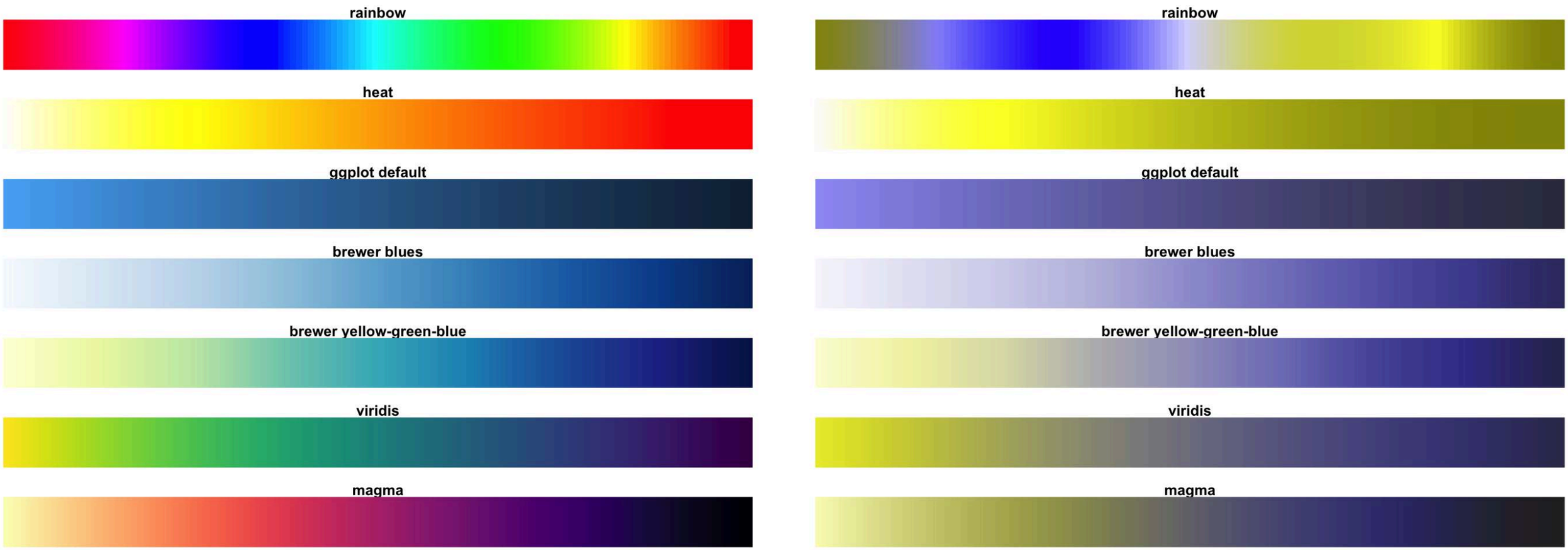
Desaturated



When figures are printed or rendered in grayscale

SEQUENTIAL COLOR PALETTES

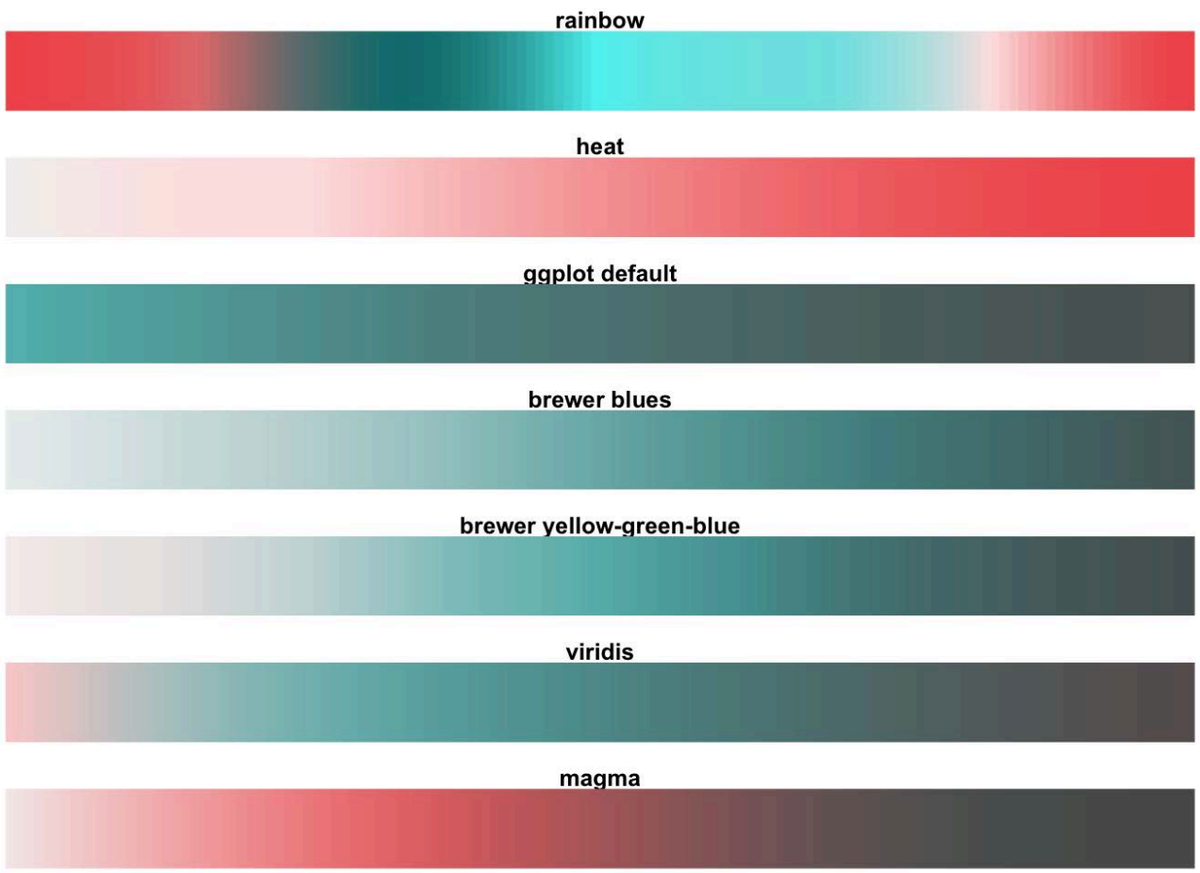
Red-green color blindness



~8% of men and 0.5% of women

SEQUENTIAL COLOR PALETTES

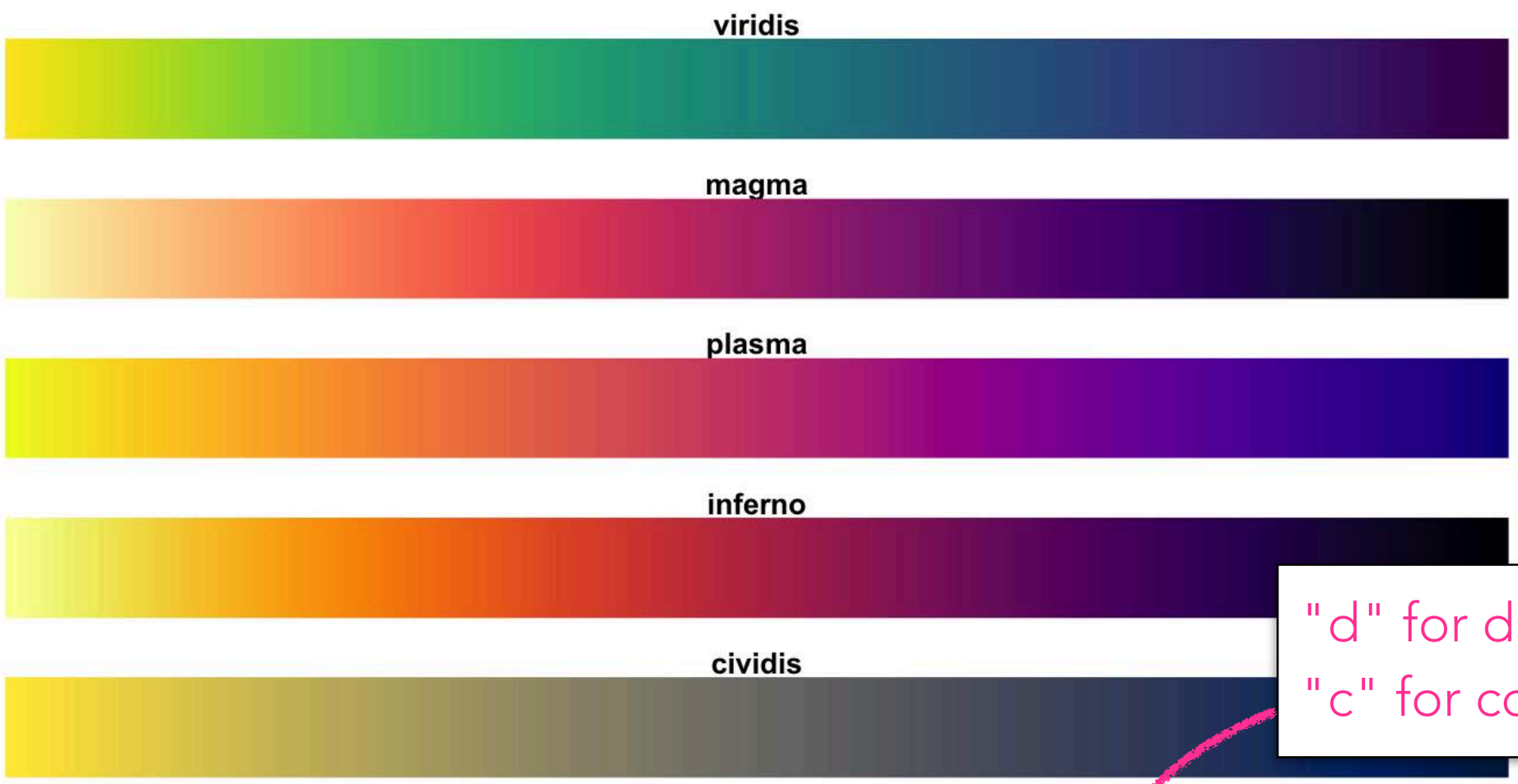
Blue-yellow color blindness



Very rare, ~1 in 40,000

SEQUENTIAL COLOR PALETTES

viridis

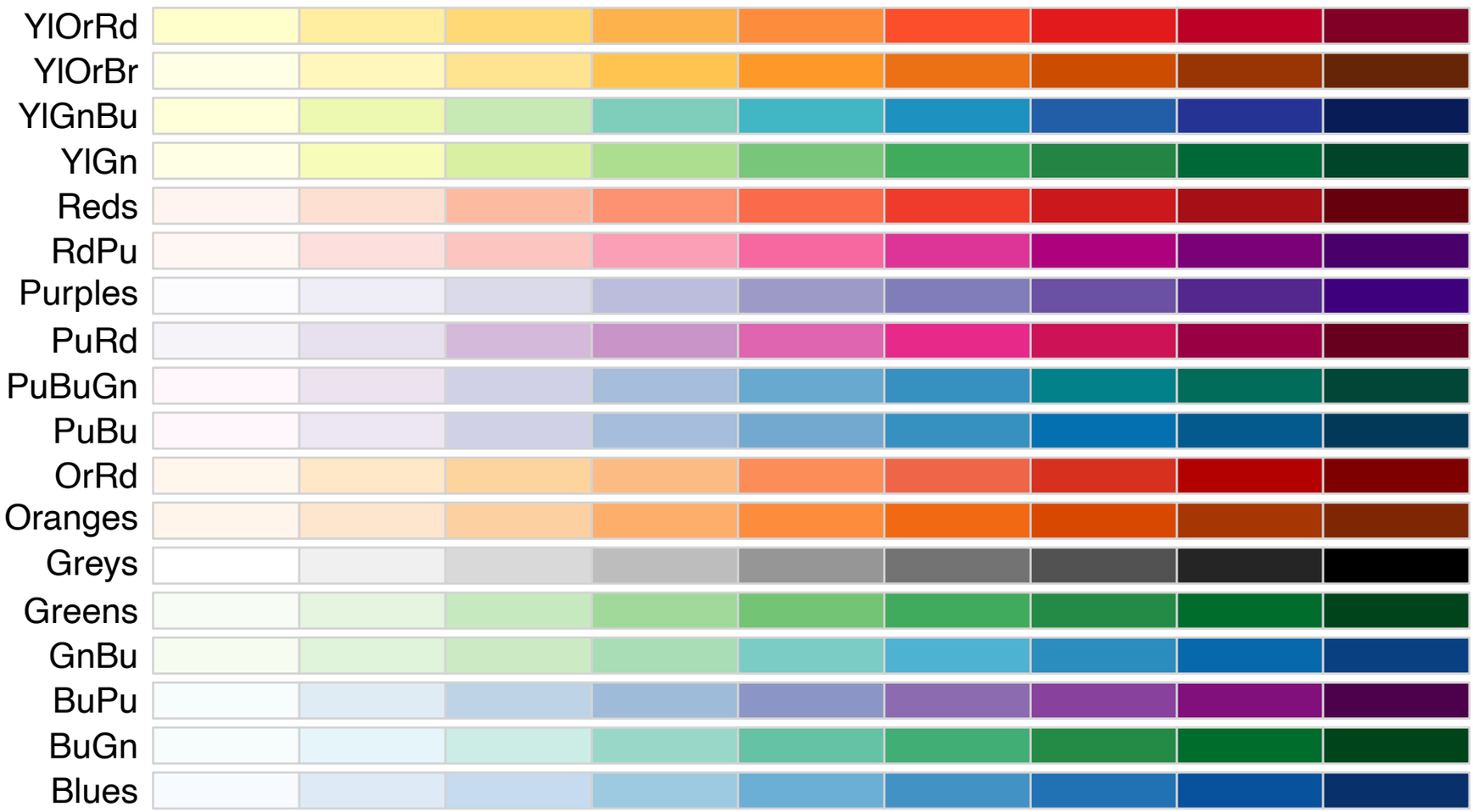


"d" for discrete
"c" for continuous

```
Ordered categories: scale_color_viridis_d(option = "magma")
Continuous values: scale_color_viridis_c(option = "magma")
```

SEQUENTIAL COLOR PALETTES

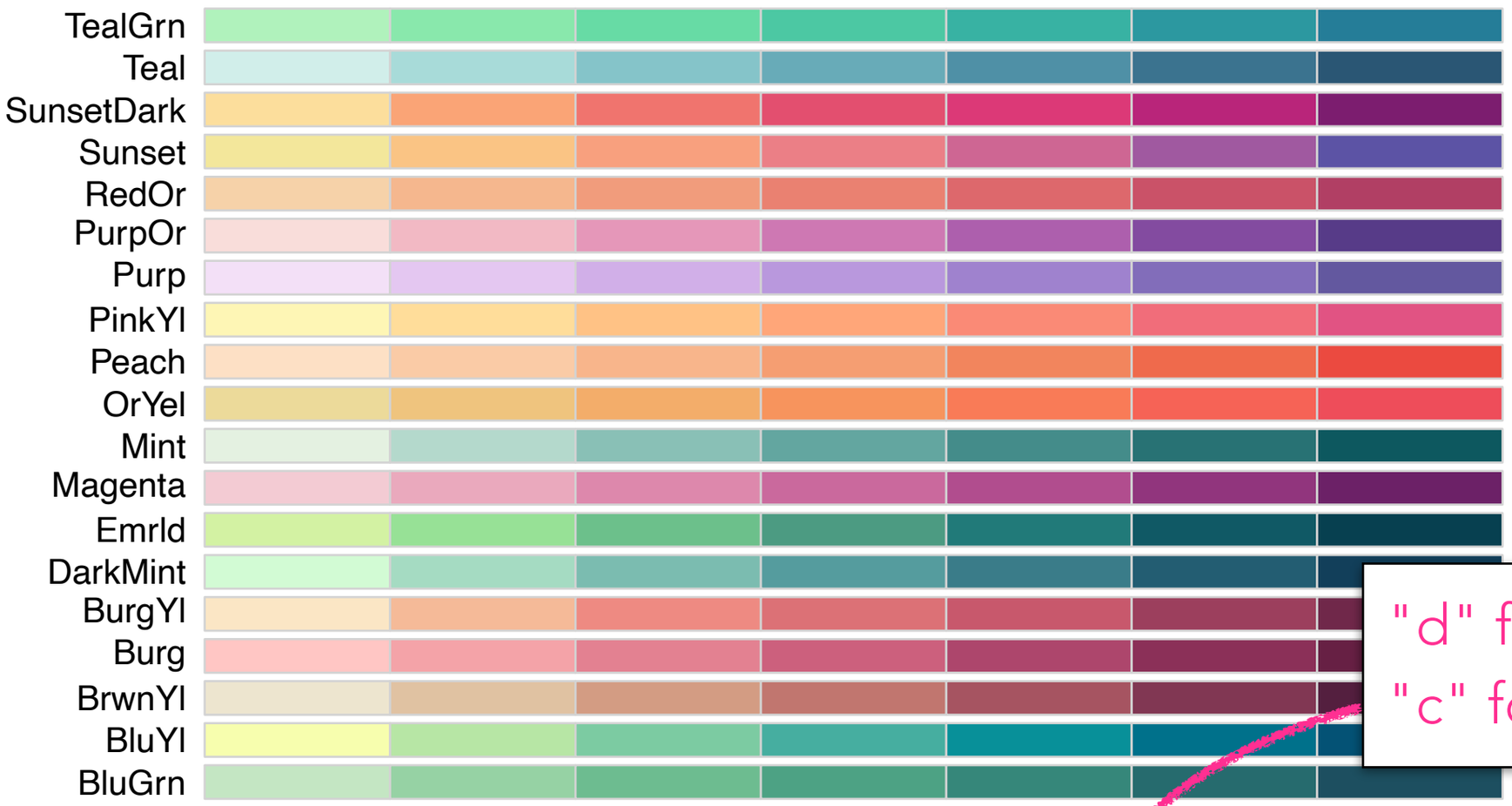
RColorBrewer




```
Ordered categories: scale_color_brewer(palette = "RdPu")
Continuous values: scale_color_distiller(palette = "RdPu")
```

SEQUENTIAL COLOR PALETTES

rcartocolor



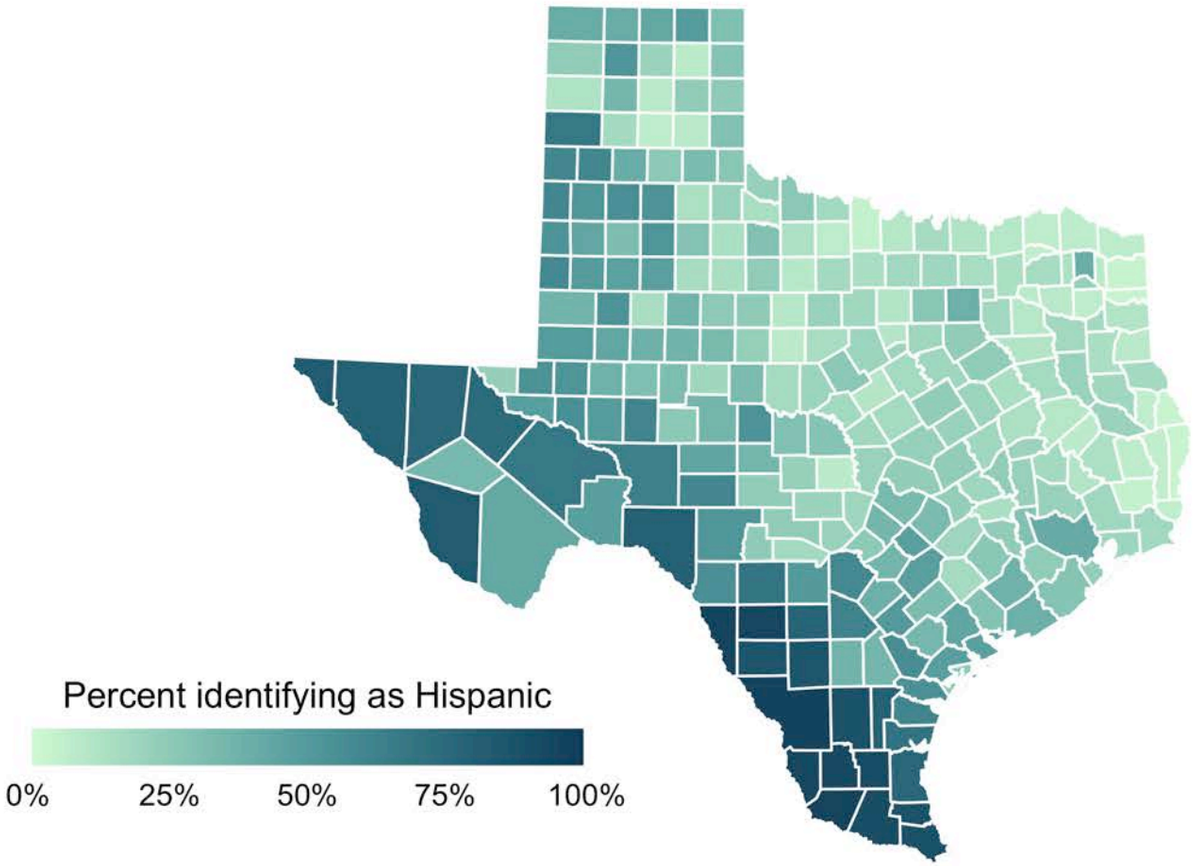

```
Ordered categories: scale_color_carto_d(palette = "Mint")
Continuous values: scale_color_carto_c(palette = "Mint")
```

SEQUENTIAL COLOR PALETTES

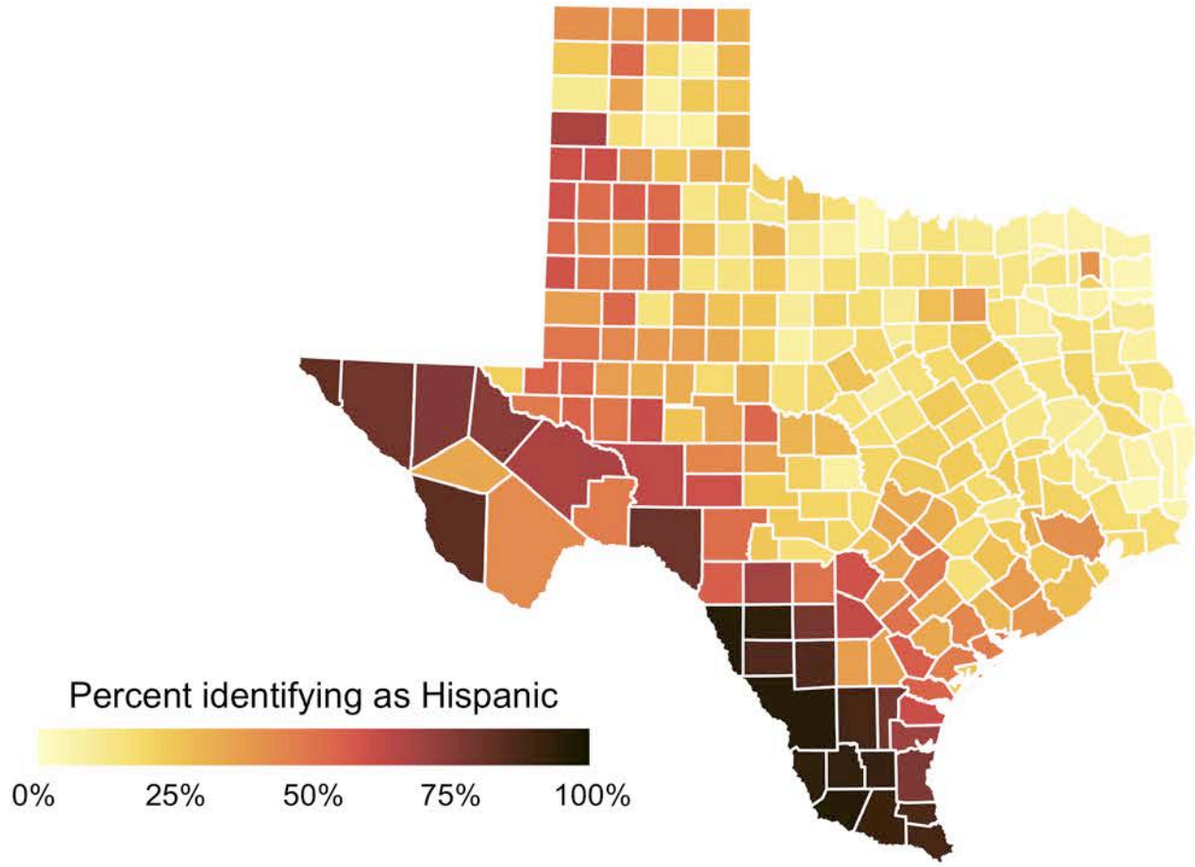
scico


```
Ordered categories: scale_color_scico_d(palette = "tokyo")
Continuous values: scale_color_scico(palette = "tokyo")
```

SEQUENTIAL COLOR PALETTES

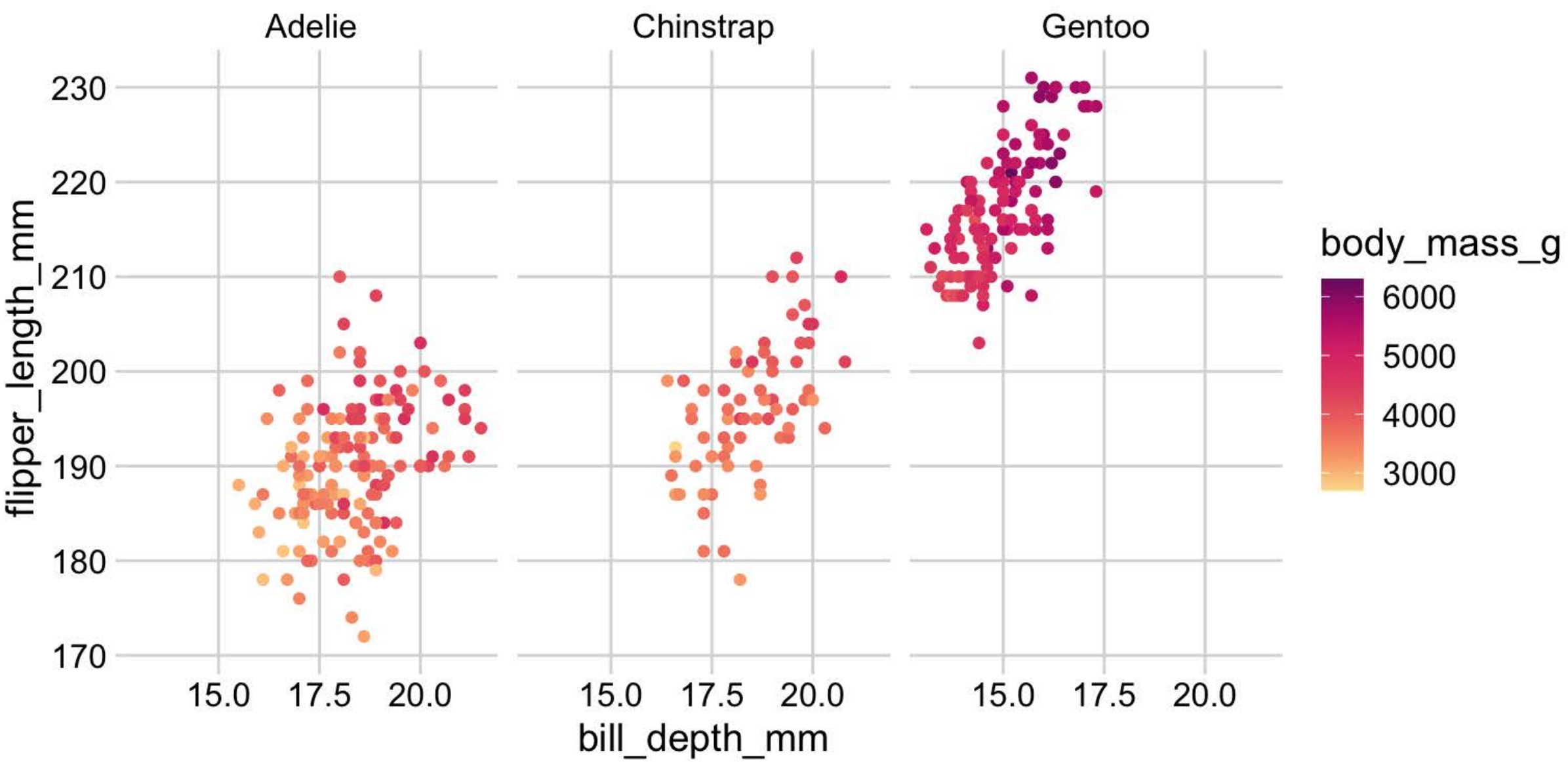


"DarkMint" from rcartocolor



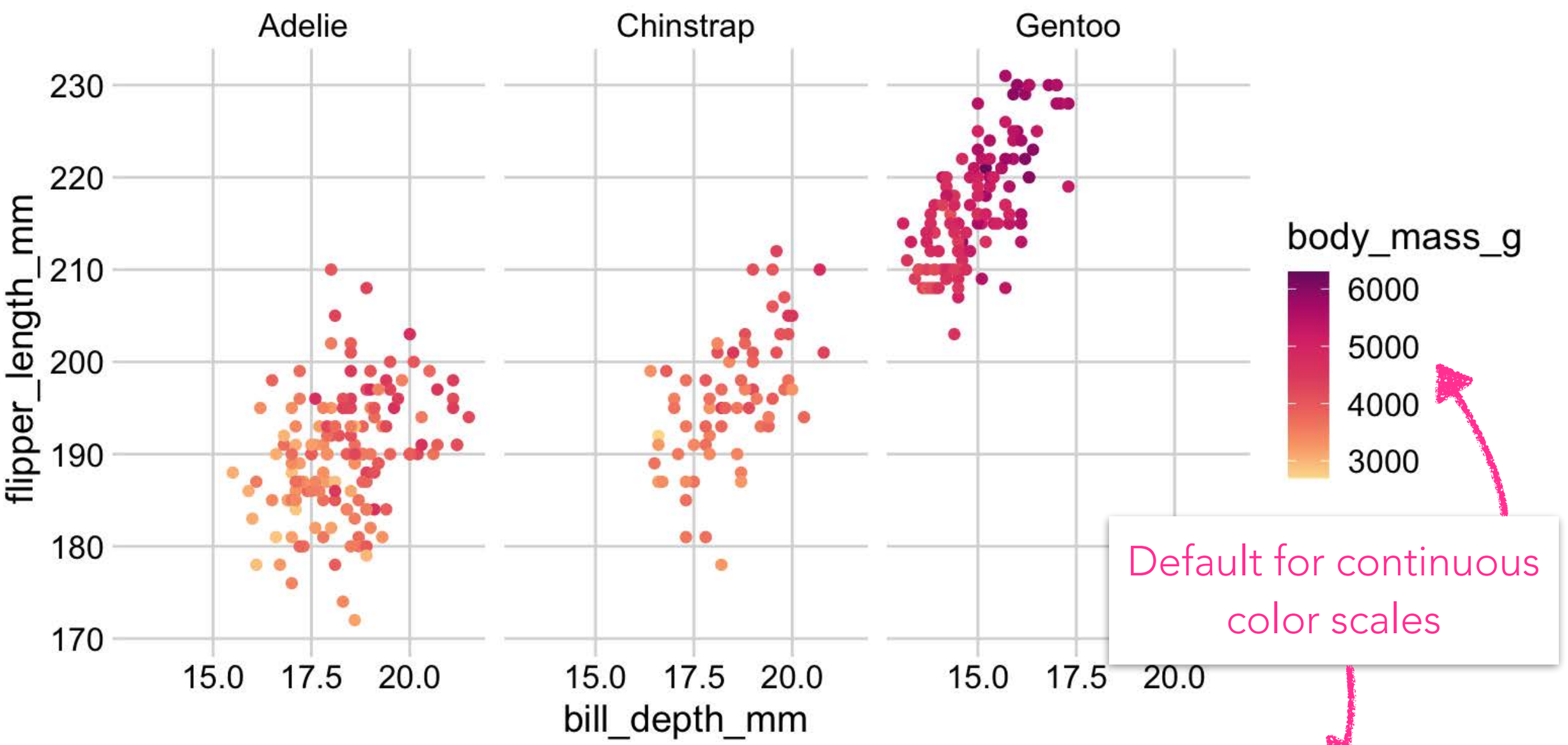
"lajolla" from scico

YOUR TURN



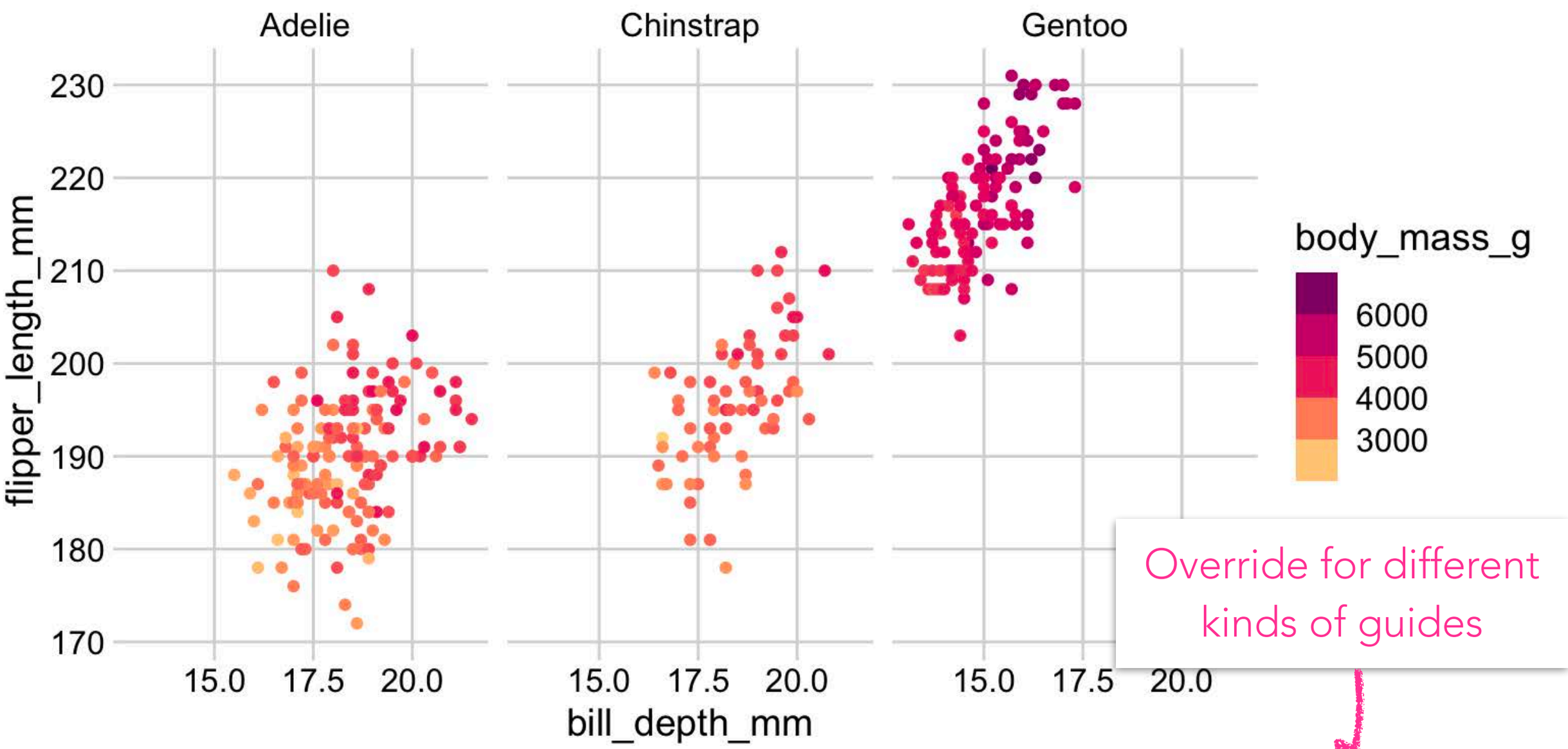

```
scale_color_carto_c(palette = "SunsetDark")
```

YOUR TURN




```
scale_color_carto_c(palette = "SunsetDark", guide = "colorbar")
```

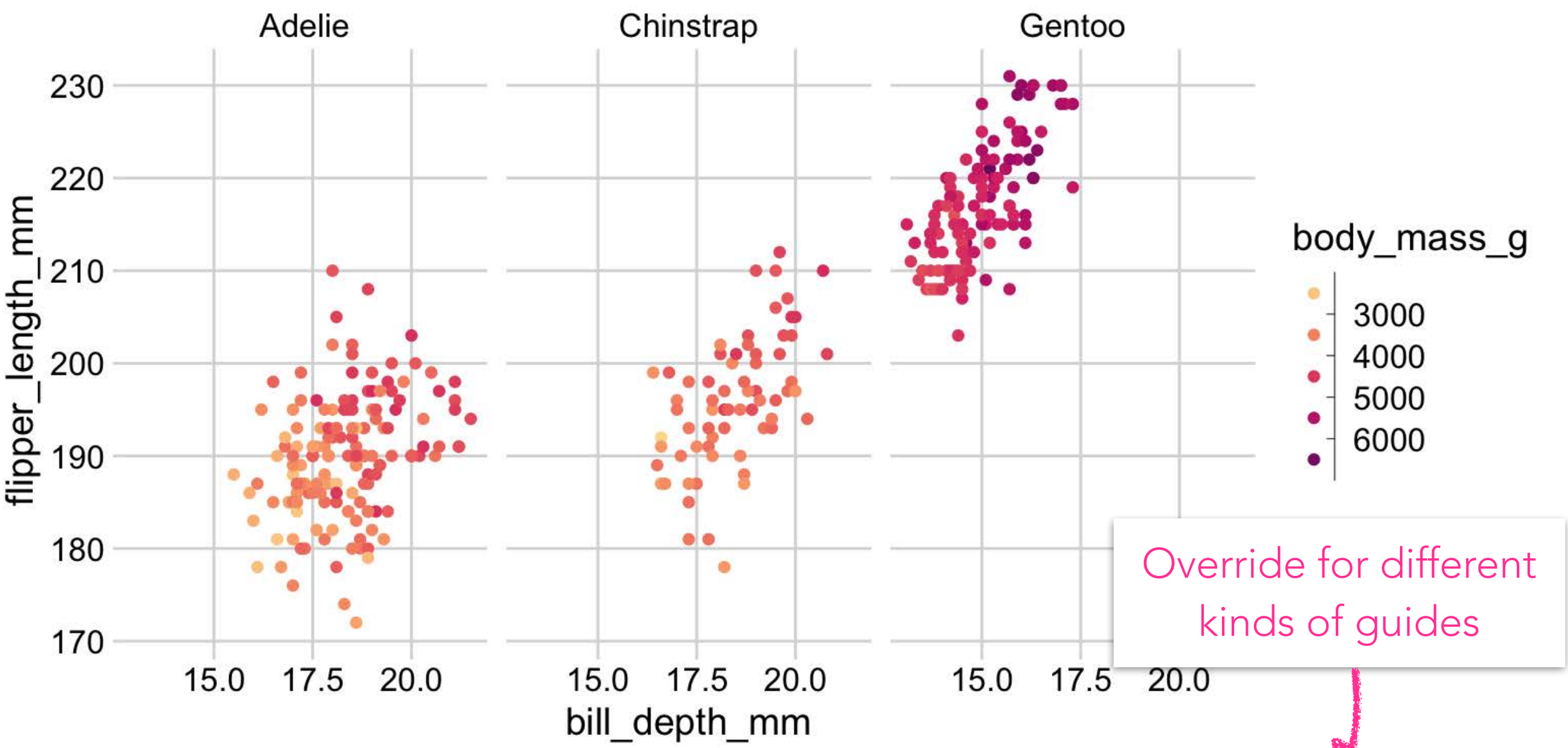
YOUR TURN



Override for different kinds of guides

```
scale_color_carto_c(palette = "SunsetDark", guide = "colorsteps")
```

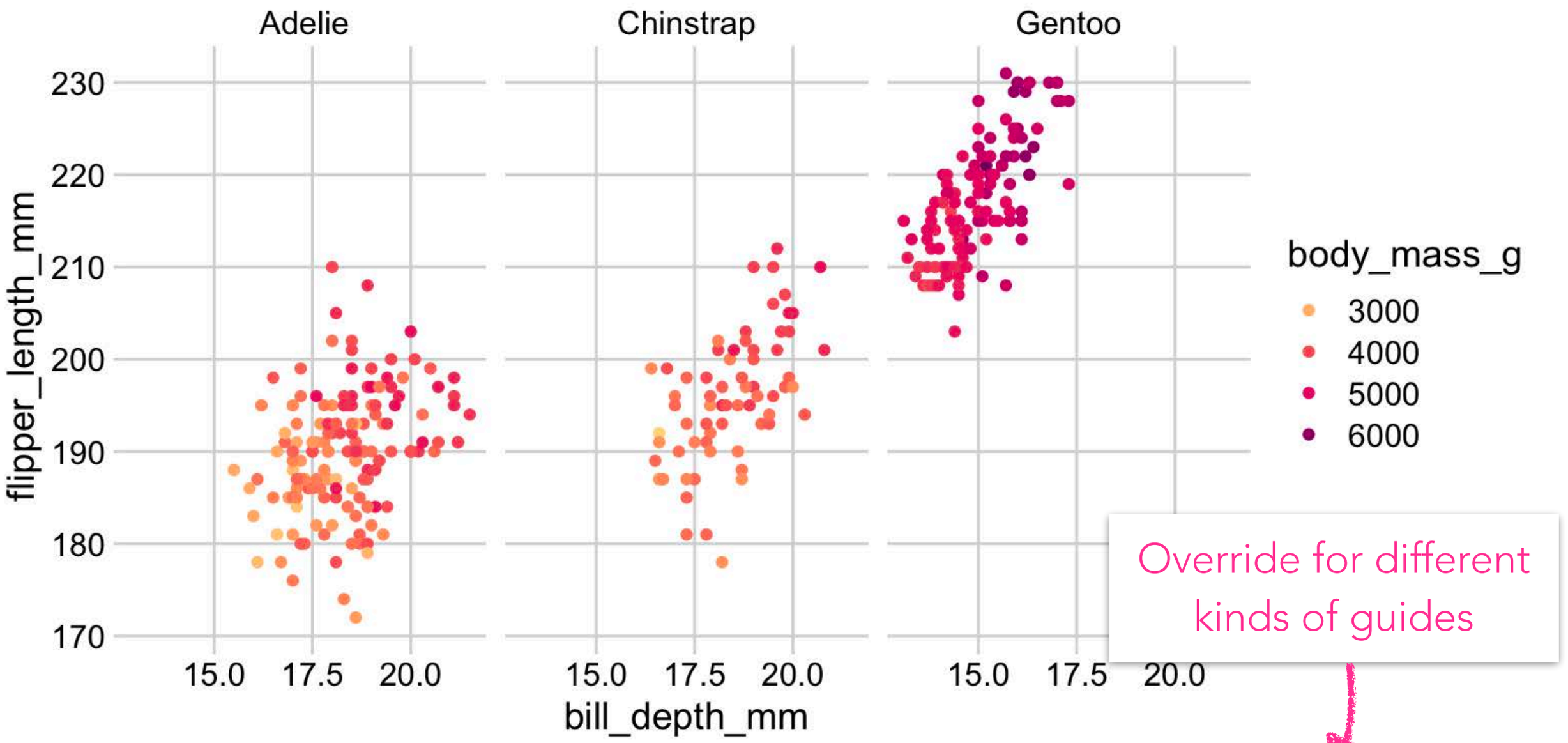
YOUR TURN



Override for different kinds of guides

```
scale_color_carto_c(palette = "SunsetDark", guide = "bins")
```

YOUR TURN

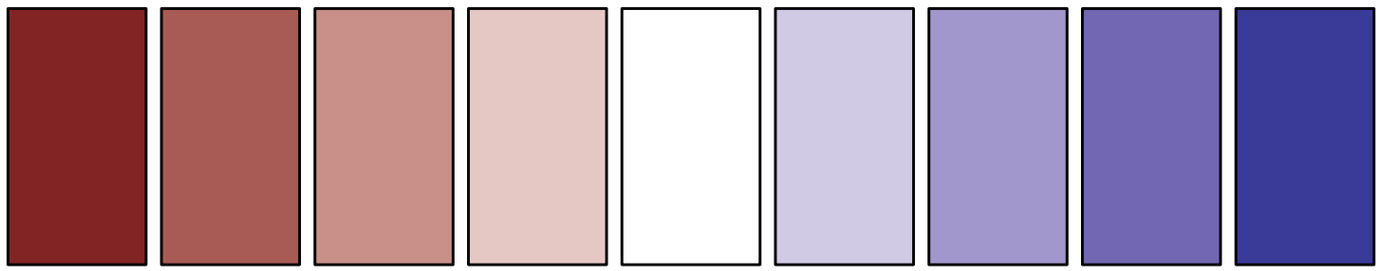



```
scale_color_carto_c(palette = "SunsetDark", guide = "legend")
```

DIVERGING COLOR PALETTES

- Use when data values diverge in two directions from a neutral midpoint (e.g., positive and negative numbers).
- Shows both how much and in which direction each value deviates from the midpoint.

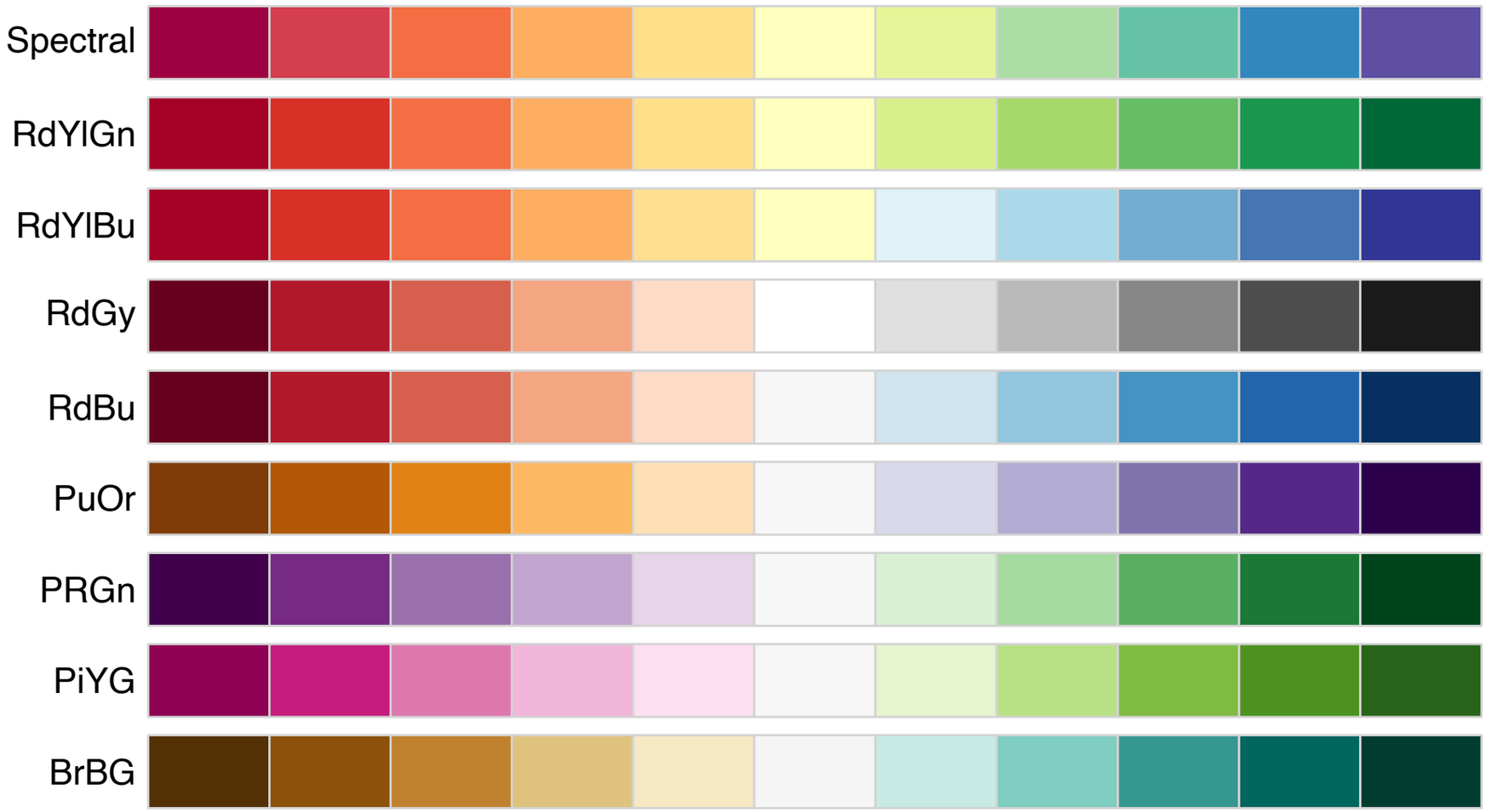
ggplot2 default



`Example: scale_color_gradient2()`

DIVERGING COLOR PALETTES

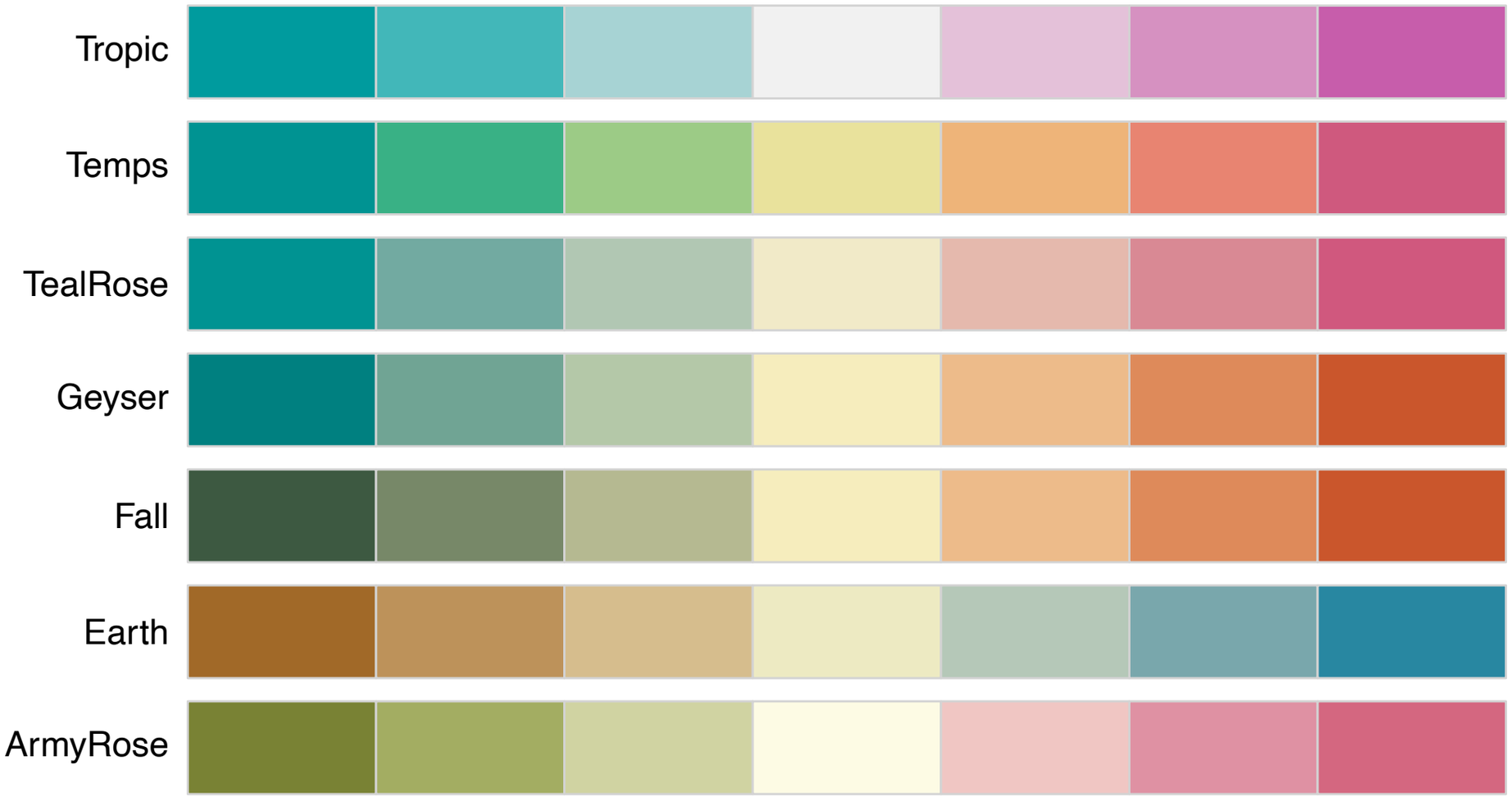
RColorBrewer




```
Ordered categories: scale_color_brewer(palette = "RdYlBu")
Continuous values: scale_color_distiller(palette = "RdYlBu")
```

DIVERGING COLOR PALETTES

rcartocolor




```
Ordered categories: scale_color_carto_d(palette = "Tropic")
Continuous values: scale_color_carto_c(palette = "Tropic")
```

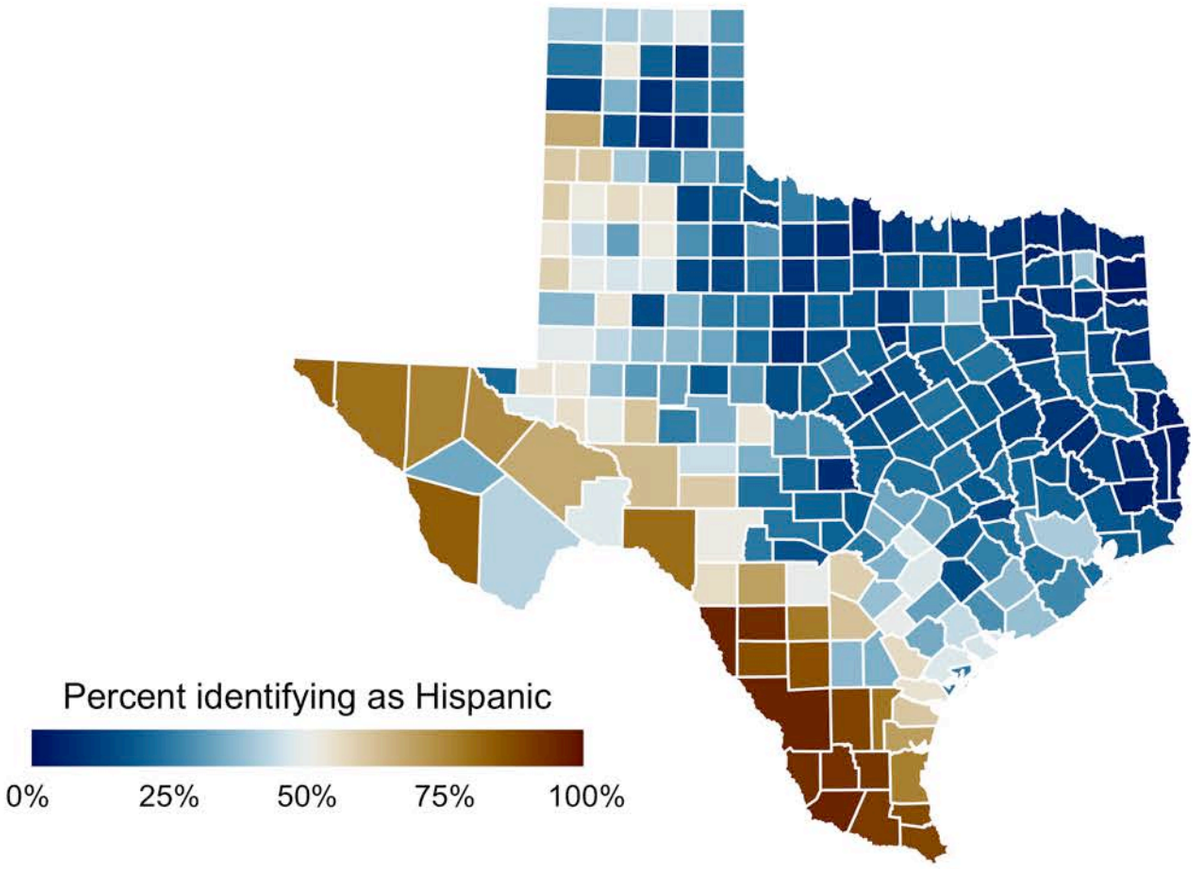
DIVERGING COLOR PALETTES

scico


```
Ordered categories: scale_color_scico_d(palette = "vik")
Continuous values: scale_color_scico(palette = "vik")
```

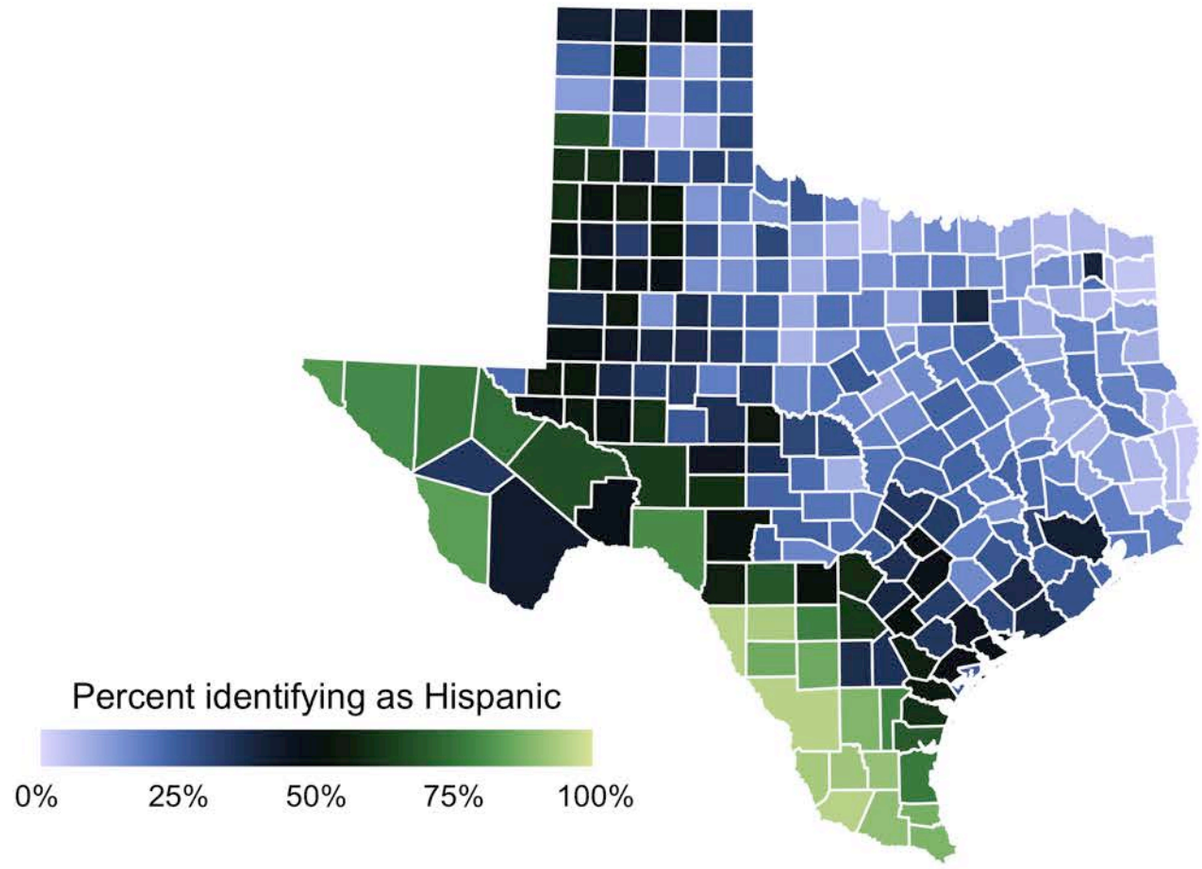
DIVERGING COLOR PALETTES

Light midpoint



"vik" from scico

Dark midpoint



"tofino" from scico

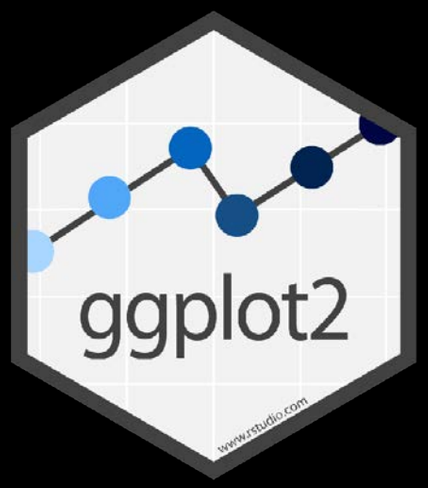
COMMON PITFALLS OF COLOR USE

- Too many colors for discrete categories
- Using coloring without clear purpose
- Color gradients with perceptual discontinuities (e.g., rainbows)
- Not considering conversion to grayscale and color-vision deficiency

SAVING YOUR PLOTS

TWO WAYS I RECOMMEND

Let knitr create the plots for you



`ggsave()` function

KNITR

- This should be familiar to you already.
- Figure height and width can be specified in chunk options.

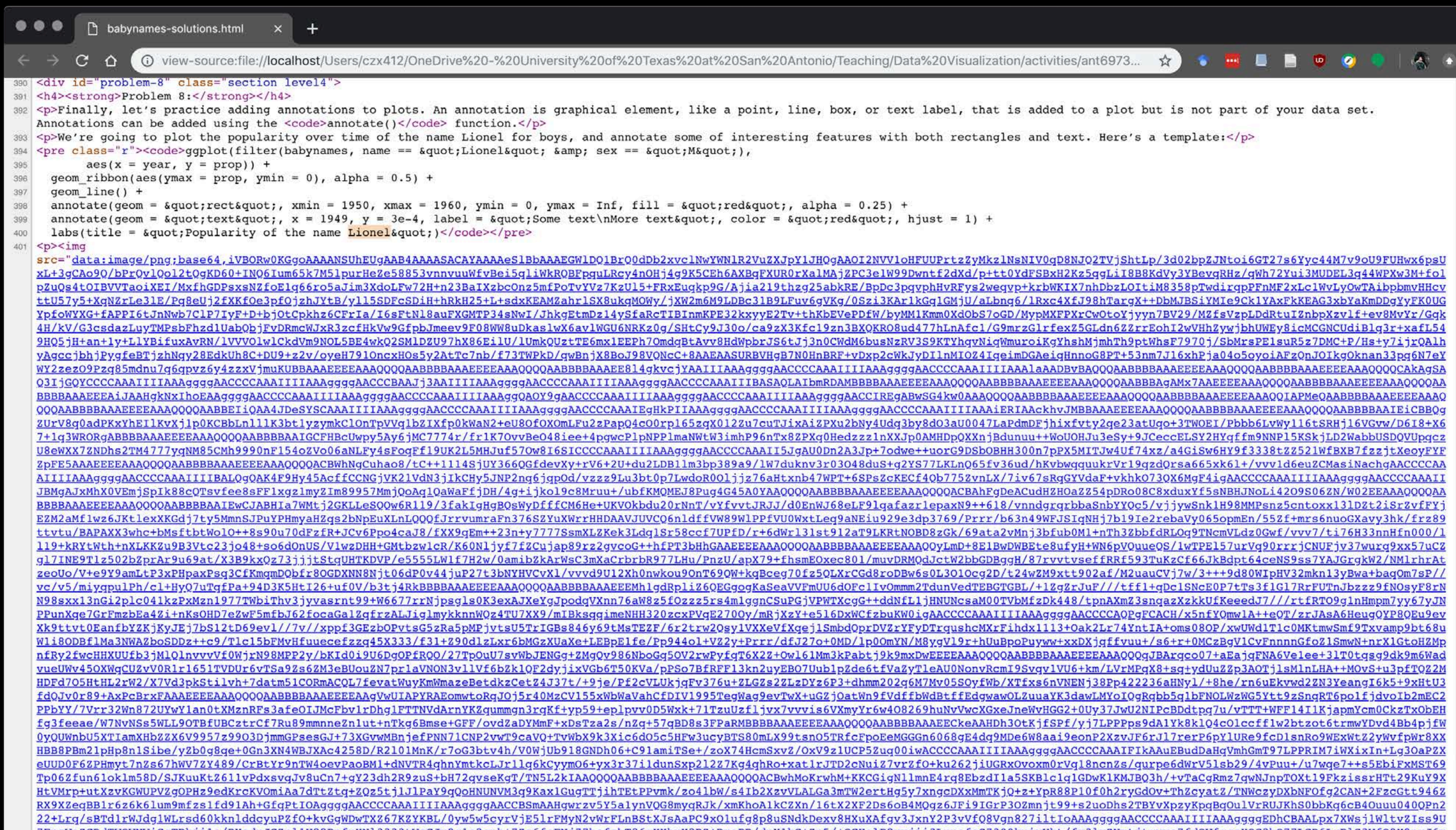
fig.height

fig.width

```
179
180 ```{r, fig.height = 6, fig.width = 10}
181 ggplot(filter(babynames, name == "Lionel" & sex == "M"),
182        aes(x = year, y = prop)) +
183   geom_ribbon(aes(ymax = prop, ymin = 0), alpha = 0.5) +
184   geom_line() +
185   annotate(geom = "rect", xmin = 1950, xmax = 1960, ymin = 0, ymax = Inf, fill = "red", alpha = 0.25) +
186   annotate(geom = "text", x = 1949, y = 3e-4, label = "Some text\nMore text", color = "red", hjust = 1) +
187   labs(title = "Popularity of the name Lionel")
188 ```
189
```

KNITR

- Where are the image files?



KNITR

- By default, knitr creates standalone HTML documents by encoding the image and embedding it inline with the document.

- That means that there is no separate image file sitting somewhere on your computer.

- This makes the HTML file portable!

- But if you need to *save* one of the images as a file, you can do so easily from your web browser.

KNITR

In-class Activity: Changing Ba

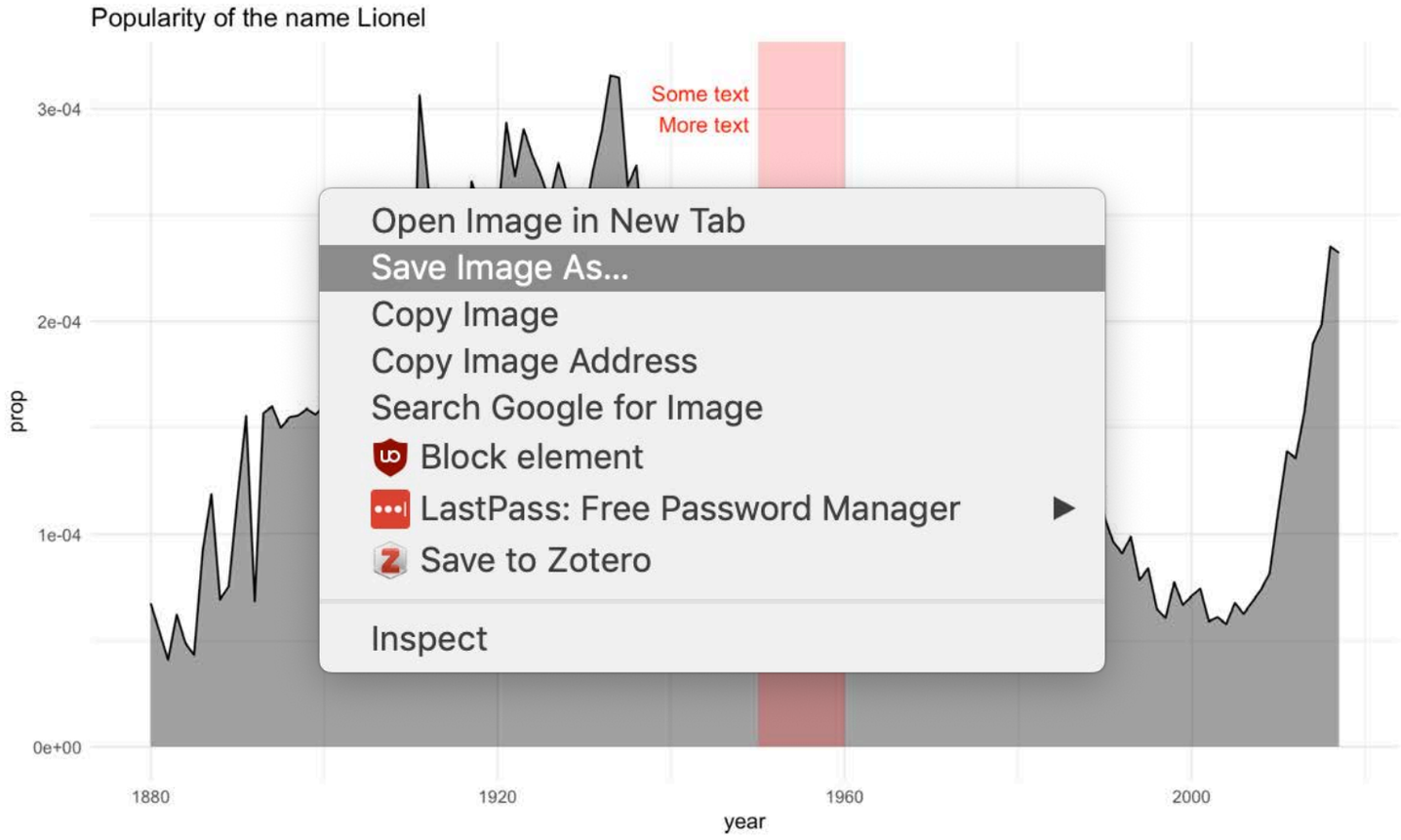
File | file://localhost/Users/czx412/OneDrive%20-%20University%20of%20Texas%20at%20San%20Antonio/Teaching/Data%20Visualization/ac...

Problem 8:

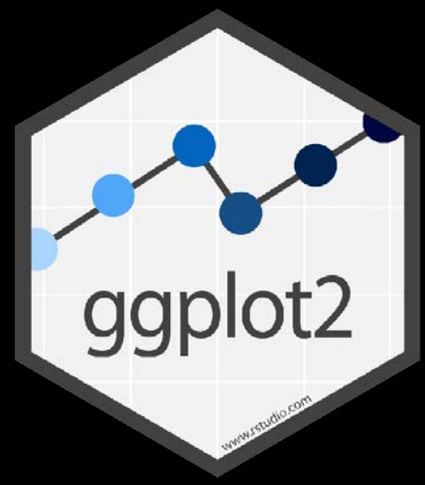
Finally, let's practice adding annotations to plots. An annotation is graphical element, like a point, line, box, or text label, that is added to a plot but is not part of your data set. Annotations can be added using the `annotate()` function.

We're going to plot the popularity over time of the name Lionel for boys, and annotate some of interesting features with both rectangles and text. Here's a template:

```
ggplot(filter(babynames, name == "Lionel" & sex == "M"),
       aes(x = year, y = prop)) +
  geom_ribbon(aes(ymax = prop, ymin = 0), alpha = 0.5) +
  geom_line() +
  annotate(geom = "rect", xmin = 1950, xmax = 1960, ymin = 0, ymax = Inf, fill = "red", alpha = 0.25) +
  annotate(geom = "text", x = 1949, y = 3e-4, label = "Some text\nMore text", color = "red", hjust = 1) +
  labs(title = "Popularity of the name Lionel")
```



GGSAVE



- Save a plot as an image file

```
ggsave(<filename>, <plot>, ...)
```

function

Name of file you want to create

You provide the path and file extension (e.g., ".png" or ".pdf")

Name of plot object in R (if any)

If nothing is specified here, it'll use the last plot that you created!

Options including width, height, and units

GGSAVE

- Example

```
ggplot(starwars, aes(x = eye_color)) +
  geom_bar()

ggsave("starwars-eye-color.pdf", width = 10, height = 5)
```

No path provided, so file will be saved in working directory

Function knows to create a PDF file

GGSAVE

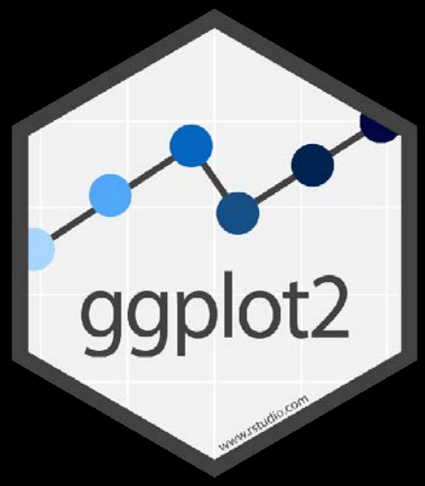
- Example

```
ggplot(starwars, aes(x = eye_color)) +
  geom_bar()

ggsave("07_viz-spatial/starwars-eye-color.pdf",
       width = 10, height = 5)
```

Provide a path to save file in subfolder of project home (note: folder must exist!)

GGSAVE



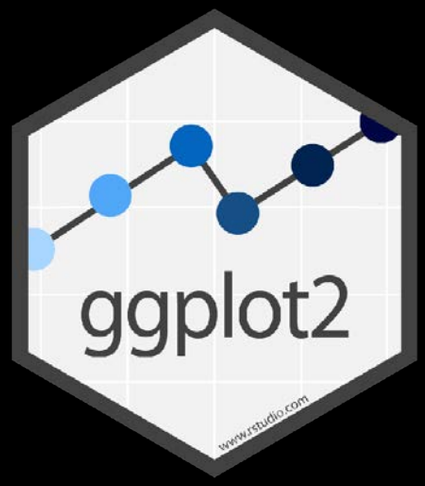
- Example

```
ggplot(starwars, aes(x = eye_color)) +
  geom_bar()

ggsave("~/Documents/Teaching/DataViz/activities/07_viz-
spatial/starwars-eye-color.pdf", width = 10, height = 5)
```

An absolute path will work but it is usually a sign of potential problems. It suggests you're not using a project-based workflow, and it will not work on other computers.

GGSAVE



- Note: `ggsave()` will not work for plots made using other plotting systems (e.g., base, lattice)

HOW DO I GET THE SIZE RIGHT?

- You can eventually develop an intuition about sizing image files…
- … but it still involves some trial and error.
- In R Studio, you can experiment using Export > Save as PDF… in the Plots pane.

HOW DO I GET THE SIZE RIGHT?

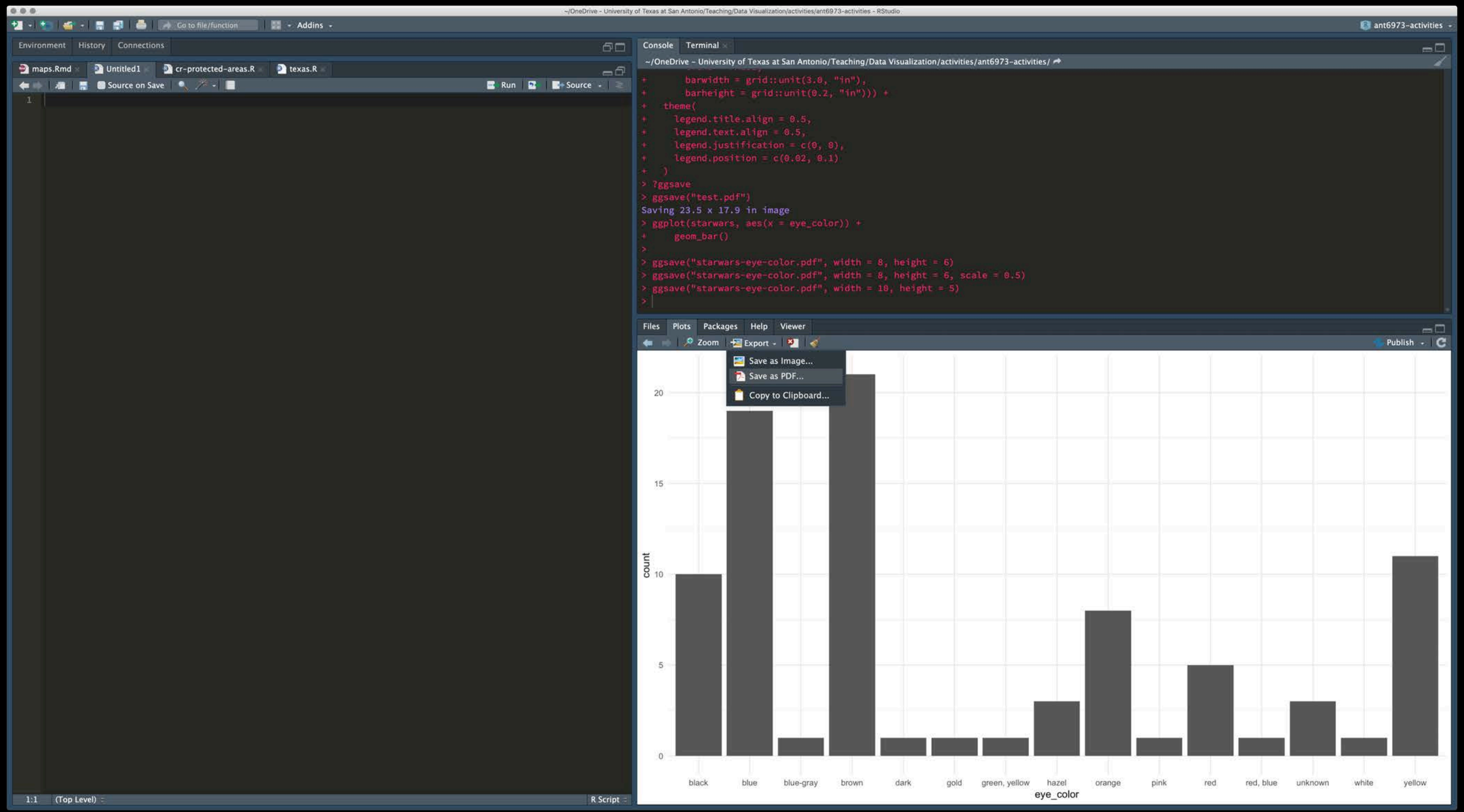


IMAGE FORMATS

- Refer to Wikle Ch. 27, "Understanding the most commonly used image file formats"

- Rules of thumb:

    1. Use PDF whenever possible, especially for creating high-quality, publication-ready files.

    2. Use PNG or TIFF for reports and web content, if required by journal, or for plots with large amounts of data (e.g., millions of points).

    3. Use JPG only for photographs or if PNG file is too large.

COMPOSING MULTIPLE PLOTS

FACETS VS COMPOUND FIGURES

- Facets: used for "small multiples," where each panel shows a subset of data and all panels use the same type of visualization.

- Compound figures: separate figures *composed* into one arrangement; the figures can use different types of visualizations or different data sets (but they should be related).

A

Adelie

Chinstrap

Gentoo

Bill depth (mm)

20.0
17.5
15.0

40
50
60

Bill length (mm)

B

Adelie

Chinstrap

Gentoo

Count

30
20
10
0

40
50
60

Bill length (mm)

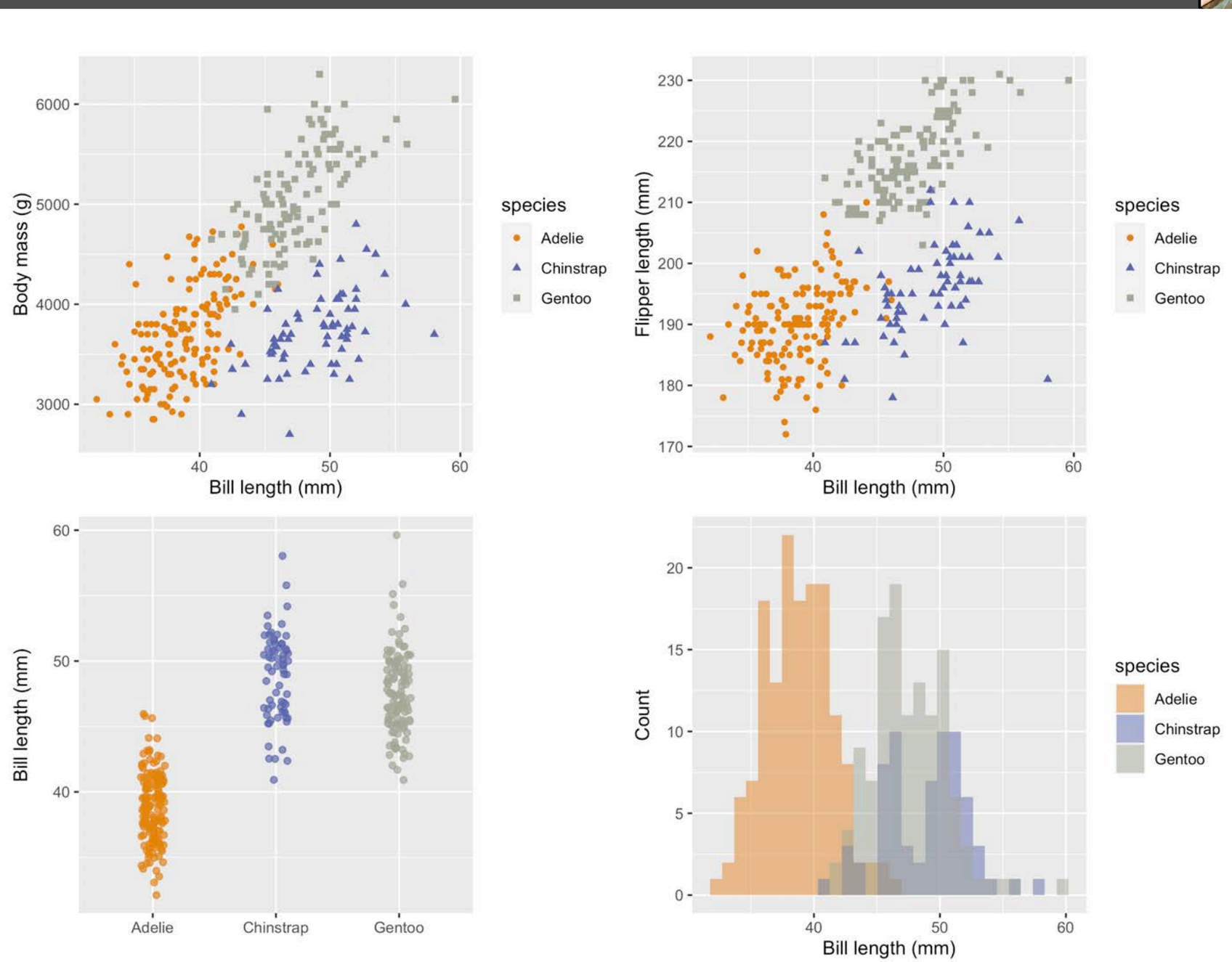
PATCHWORK

- "The goal of patchwork is to make it ridiculously simple to combine separate ggplots into the same graphic."

PATCHWORK OPERATORS

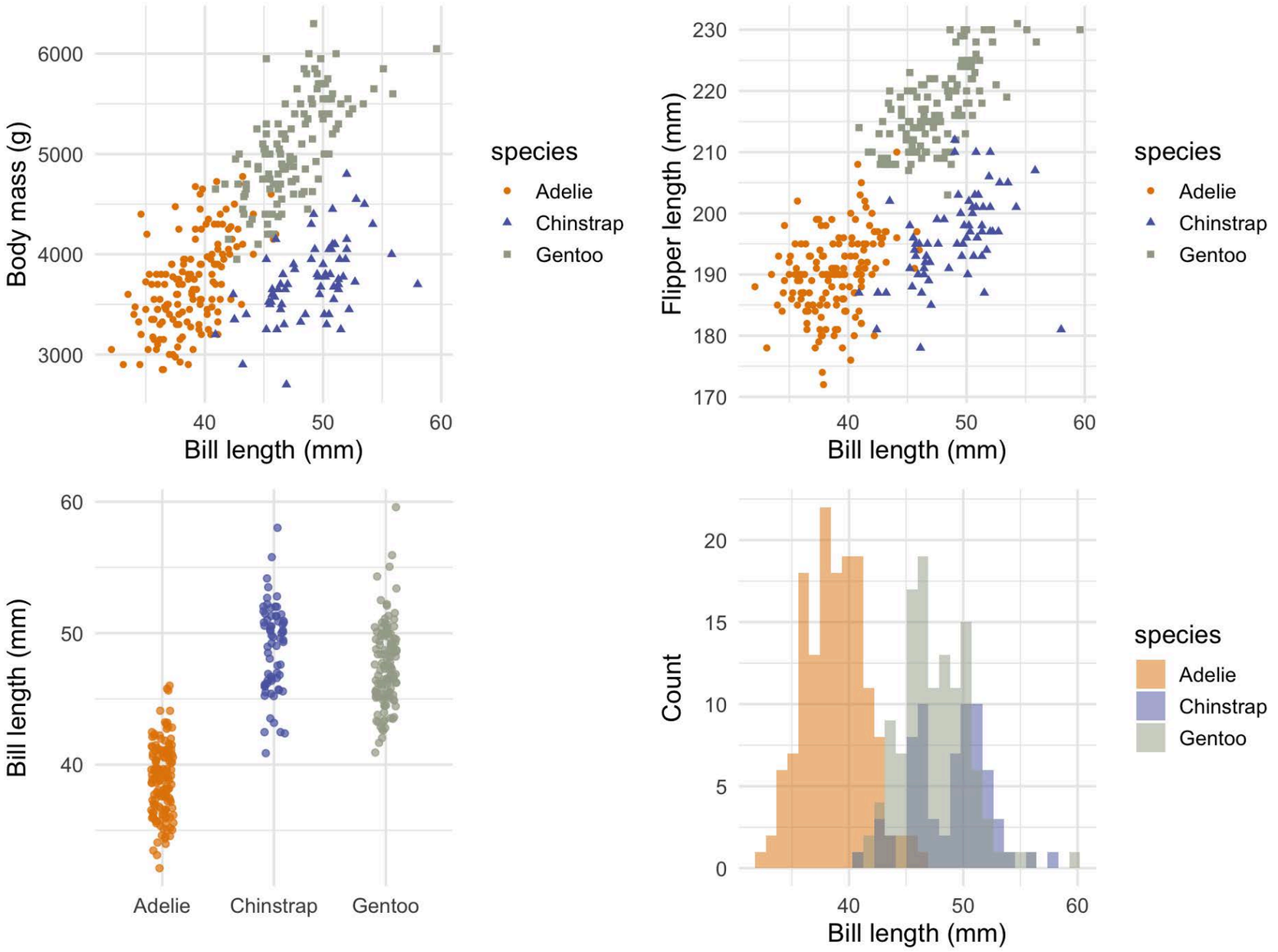
- + combines the plots into some kind of grid, without information about layout
- | puts the first thing beside the second thing
- / puts the first thing on top of the second thing
- & applies something (e.g., style) to the entire patchwork
- ( ) are used for nesting

YOUR TURN




```
p1 + p2 + p3 + p4
```

YOUR TURN




```
p1 + p2 + p3 + p4 & theme_minimal(14)
```

CONTROLLING LAYOUTS

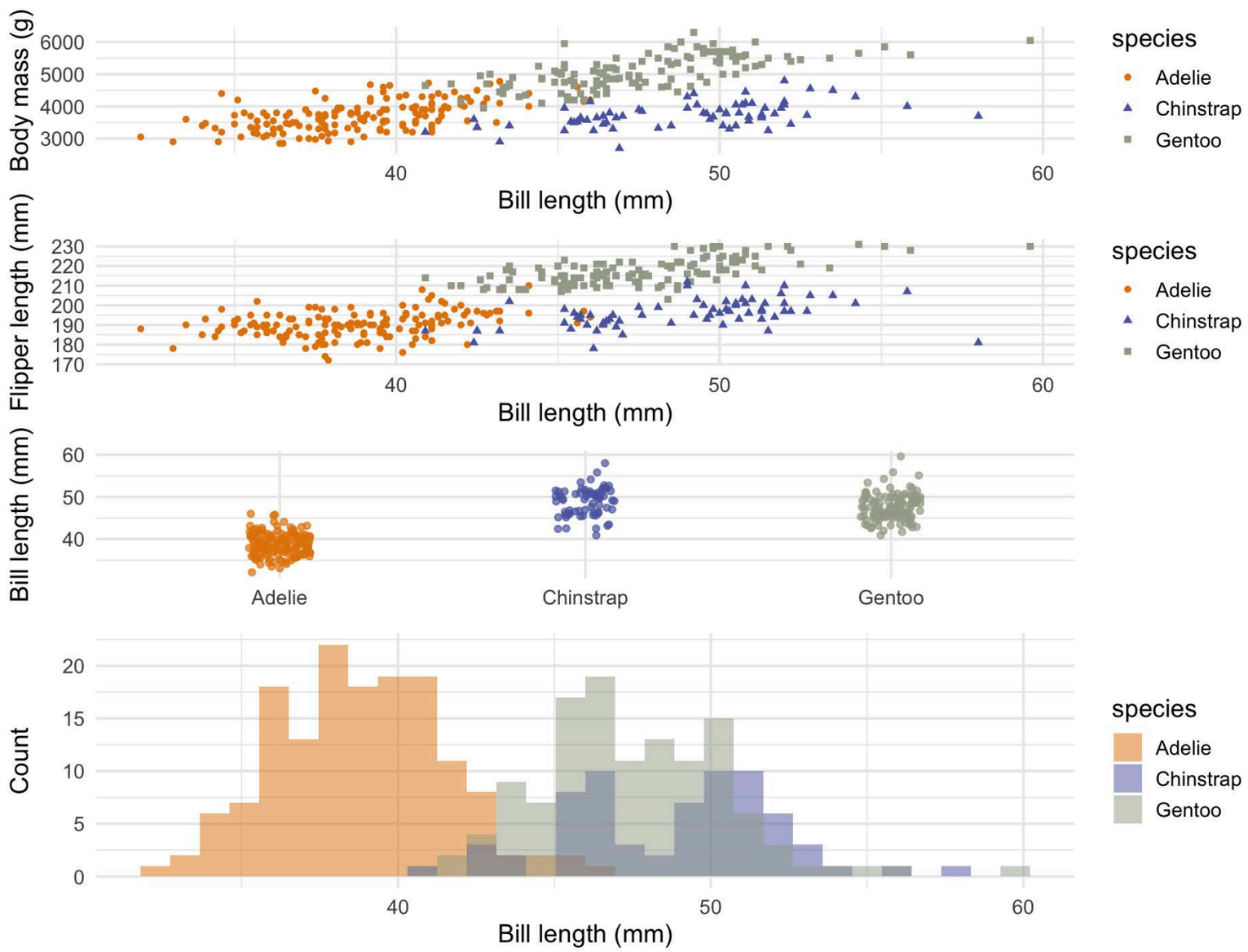
- `plot_layout()` can be added (using + operator) to the patchwork to control layout options

  - Specify number of columns or rows using arguments `ncol` or `nrow`

  - Collect all legends using the argument `guides = "collect"`

  - Relative widths and heights of each element can be specified using the arguments `widths` and `heights`

YOUR TURN




```
p1 + p2 + p3 + p4 +
  plot_layout(ncol = 1, heights = c(1, 1, 1, 2)) &
  theme_minimal(14)
```

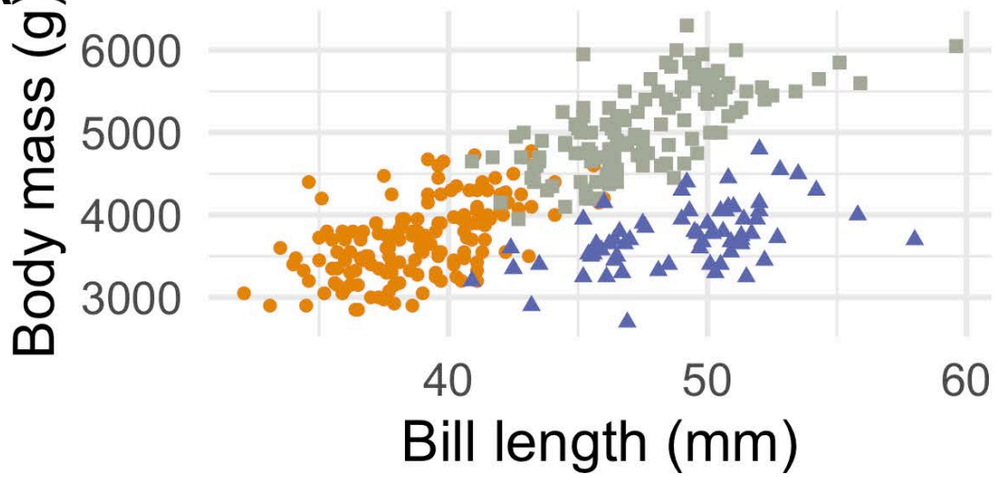
TAGS AND ANNOTATIONS

- `plot_annotation()` can be added (using + operator) to the patchwork to add tags (A, B, ...), titles, and captions

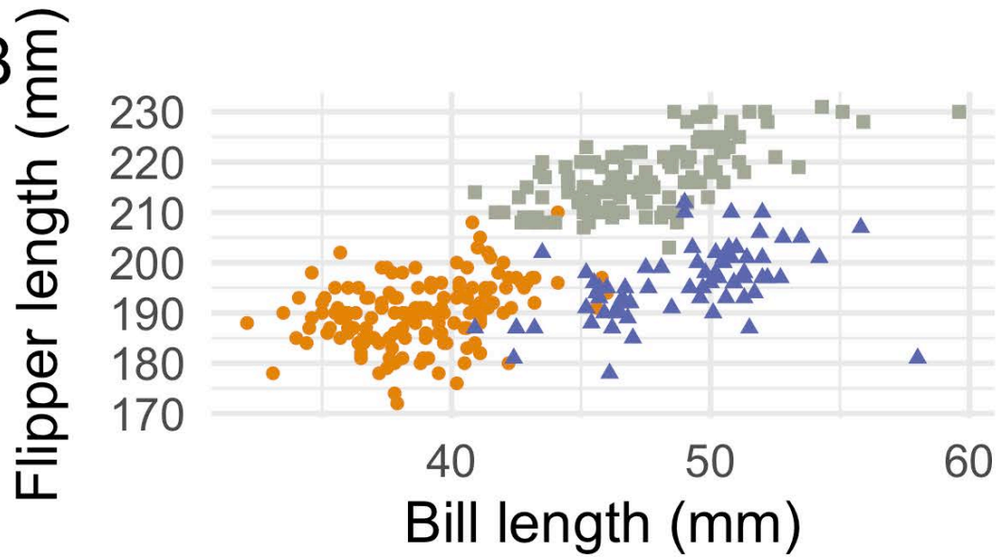
- `tag_levels` argument specifies how to enumerate the plots; possible values are `"a"`, `"A"`, `"1"`, `"i"`, and `"I"`.

- `title`, `subtitle`, and `caption` can be set for the entire patchwork.

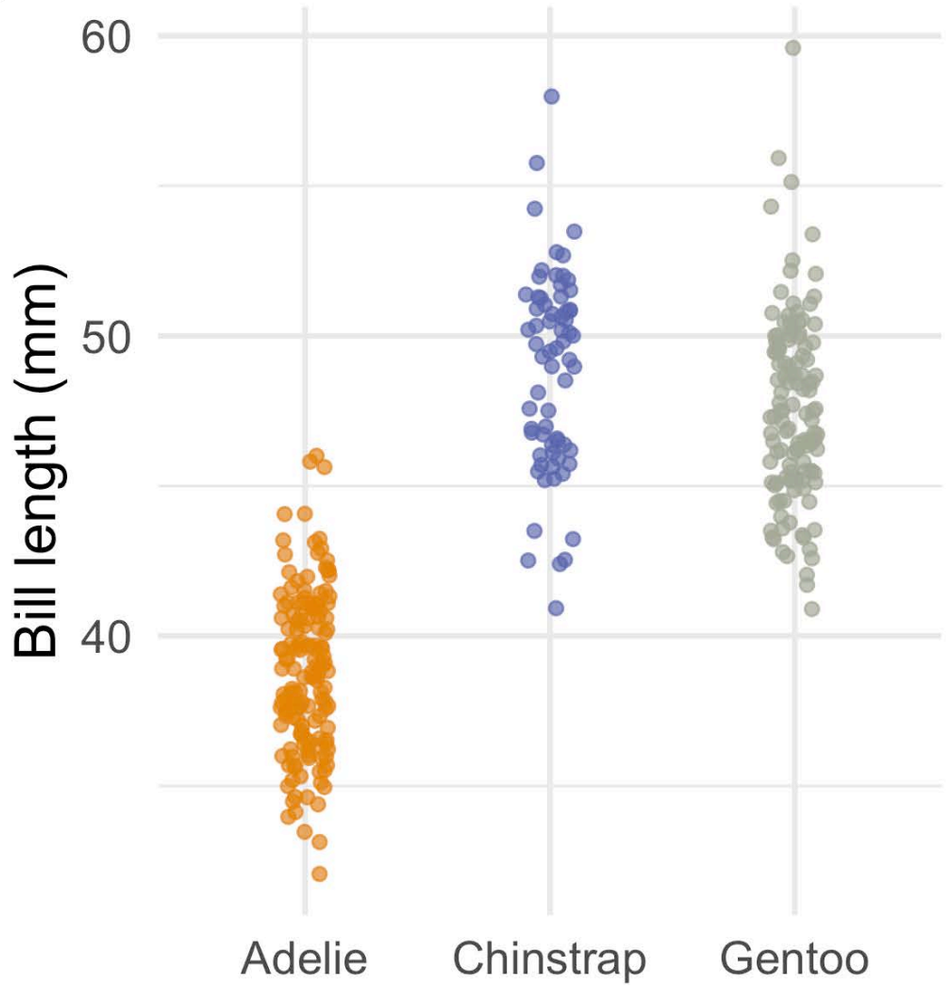
A


Body mass (g)
6000
5000
4000
3000
40
50
60
Bill length (mm)

B

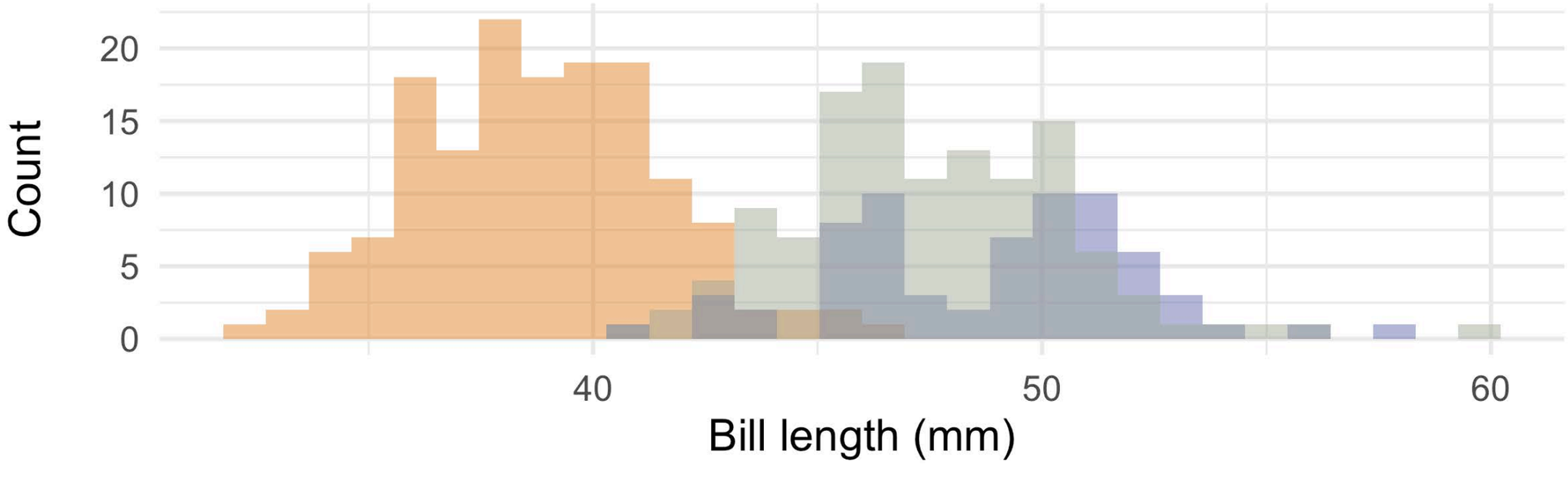

Flipper length (mm)
230
220
210
200
190
180
170
40
50
60
Bill length (mm)

C


Bill length (mm)
60
50
40
Adelie
Chinstrap
Gentoo


species
Adelie
Chinstrap
Gentoo
species
Adelie
Chinstrap
Gentoo

D

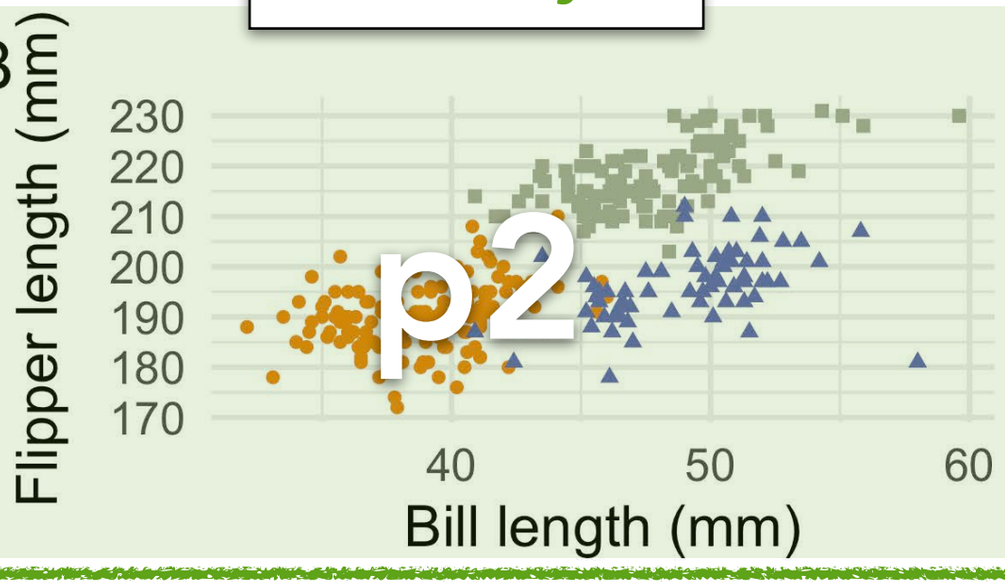

Count
20
15
10
5
0
40
50
60
Bill length (mm)

A

p1

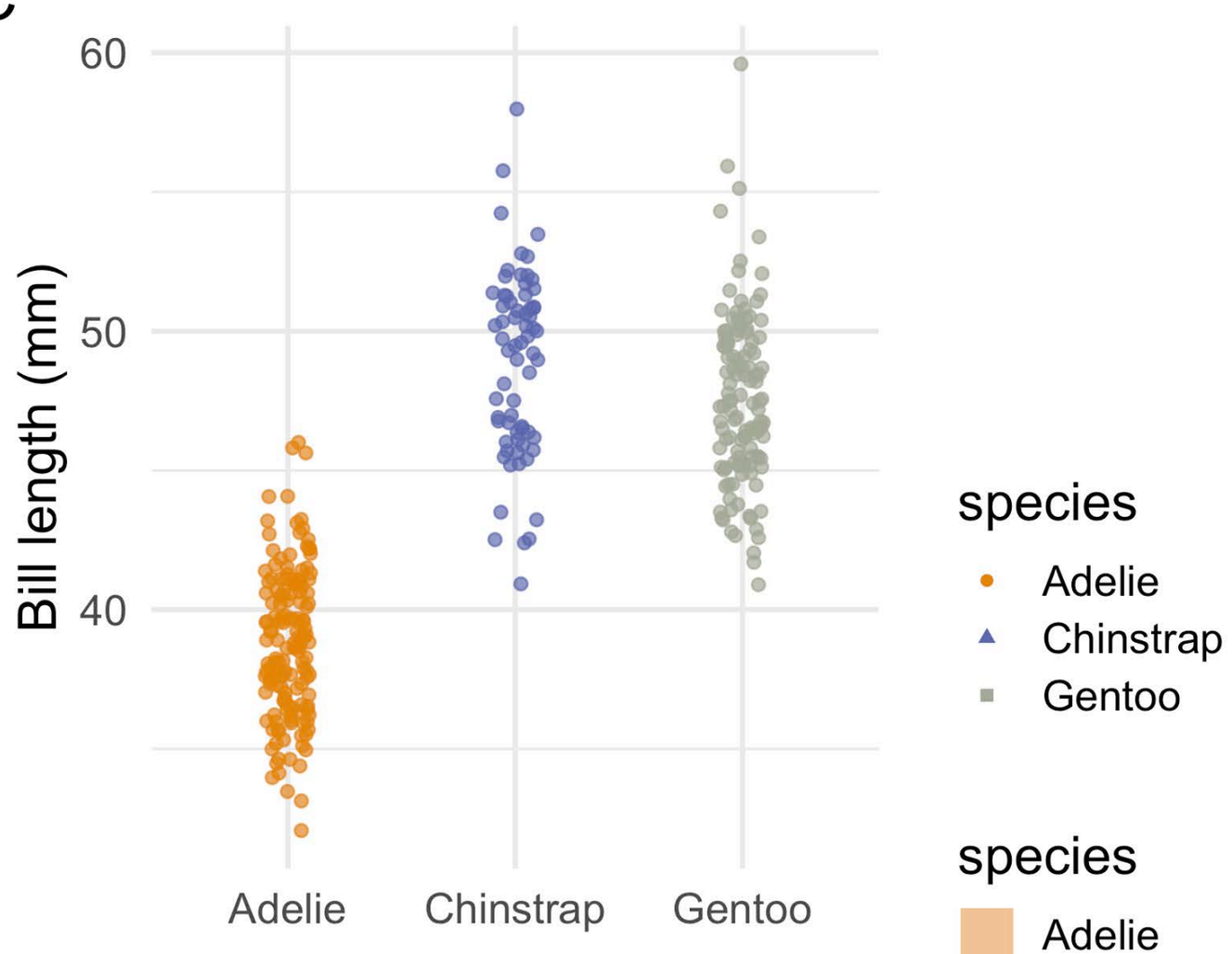
First layer

B

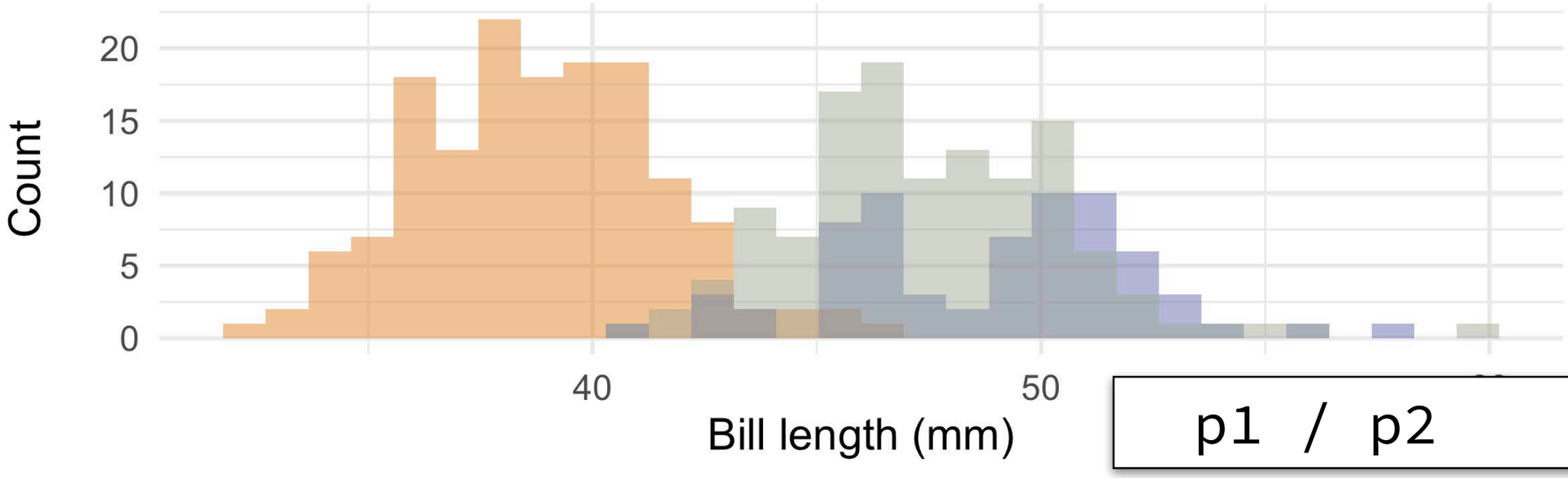


p2

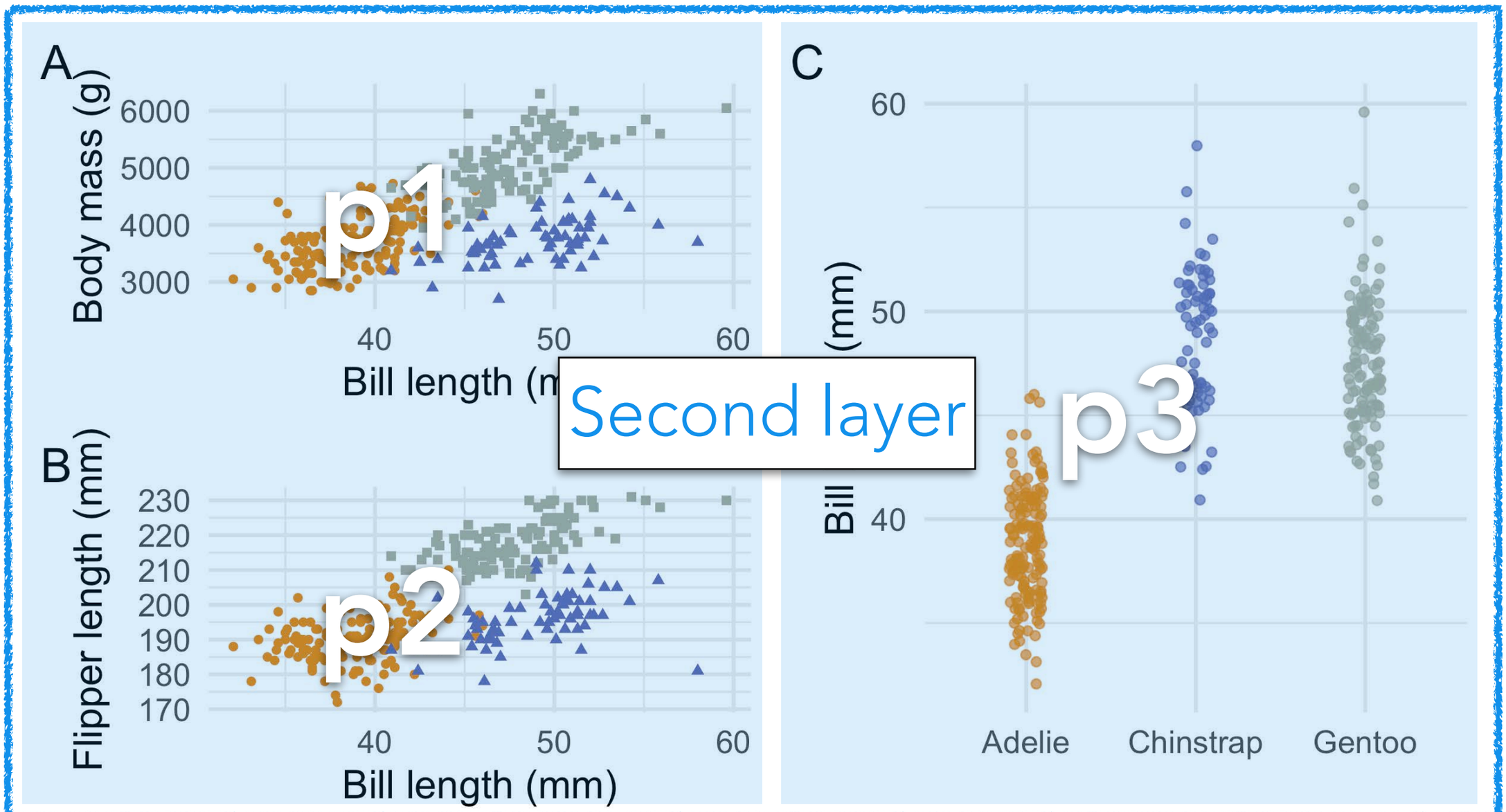
C

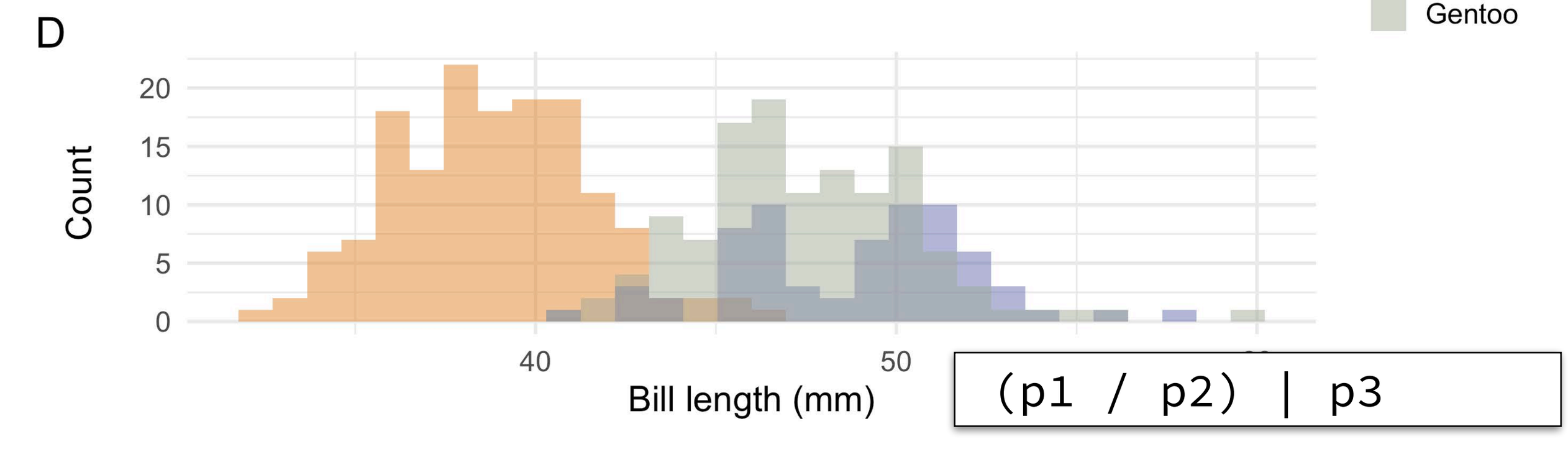


D



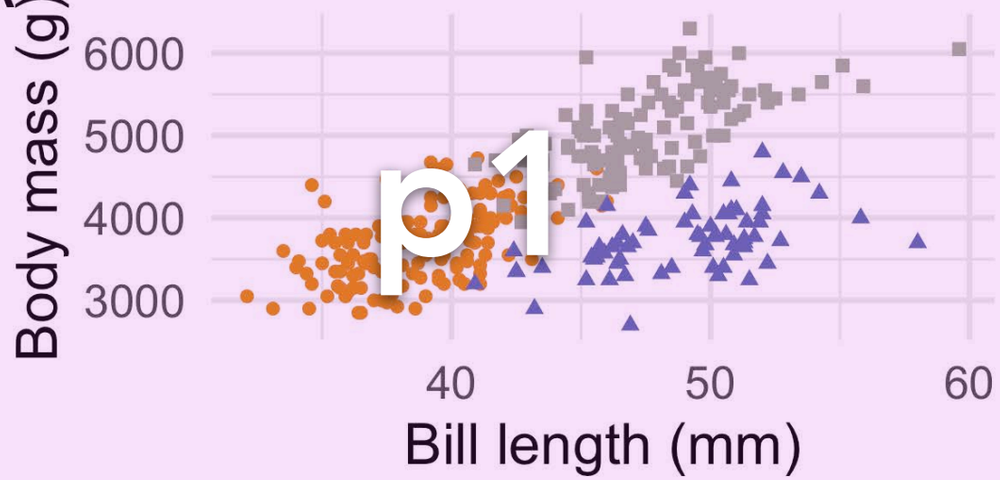

```
p1 / p2
```


A
Body mass (g)
6000
5000
4000
3000
40
50
60
Bill length (m
p1
Second layer
B
Flipper length (mm)
230
220
210
200
190
180
170
40
50
60
Bill length (mm)
p2
C
60
(mm)
50
Bill
40
p3
Adelie
Chinstrap
Gentoo
species
Adelie
Chinstrap
Gentoo
species
Adelie
Chinstrap
Gentoo

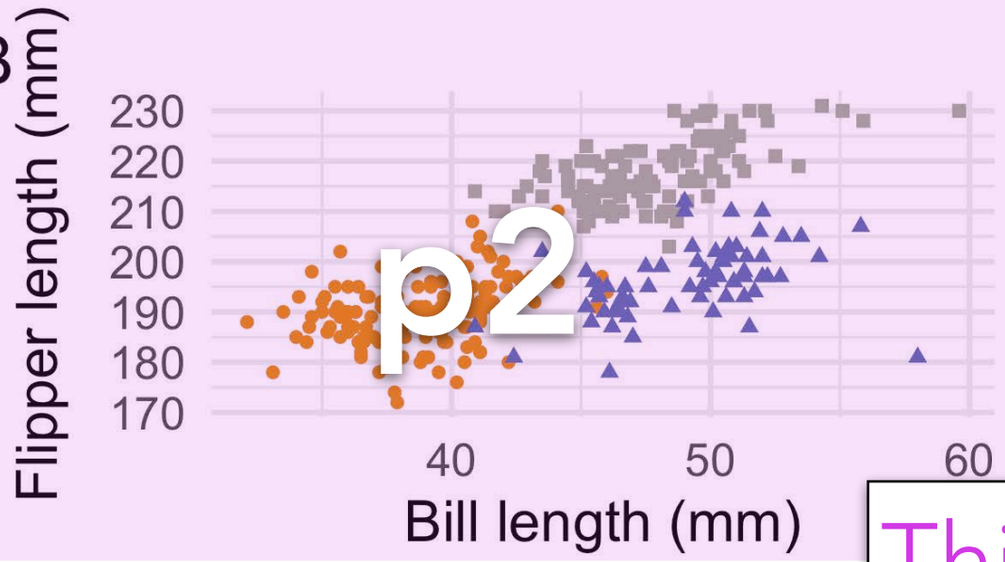

D
Count
20
15
10
5
0
40
50
Bill length (mm)

```
(p1 / p2) | p3
```

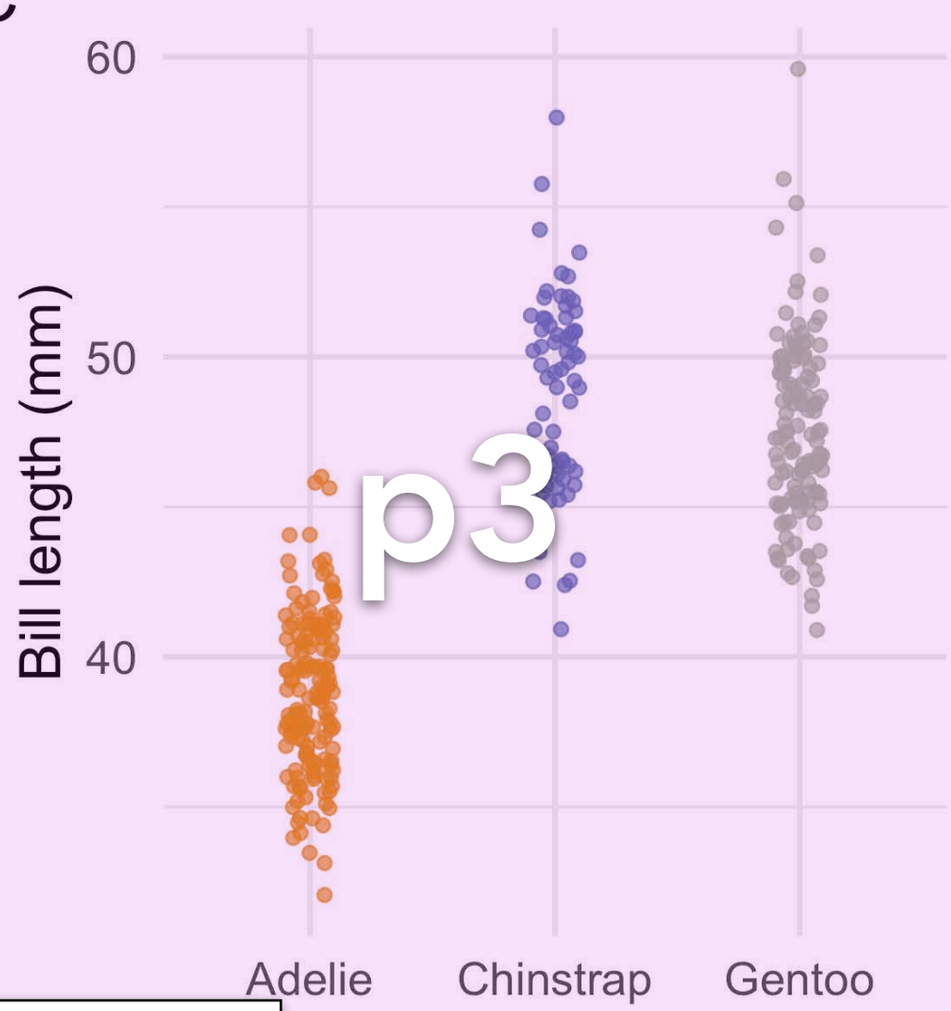
A



B

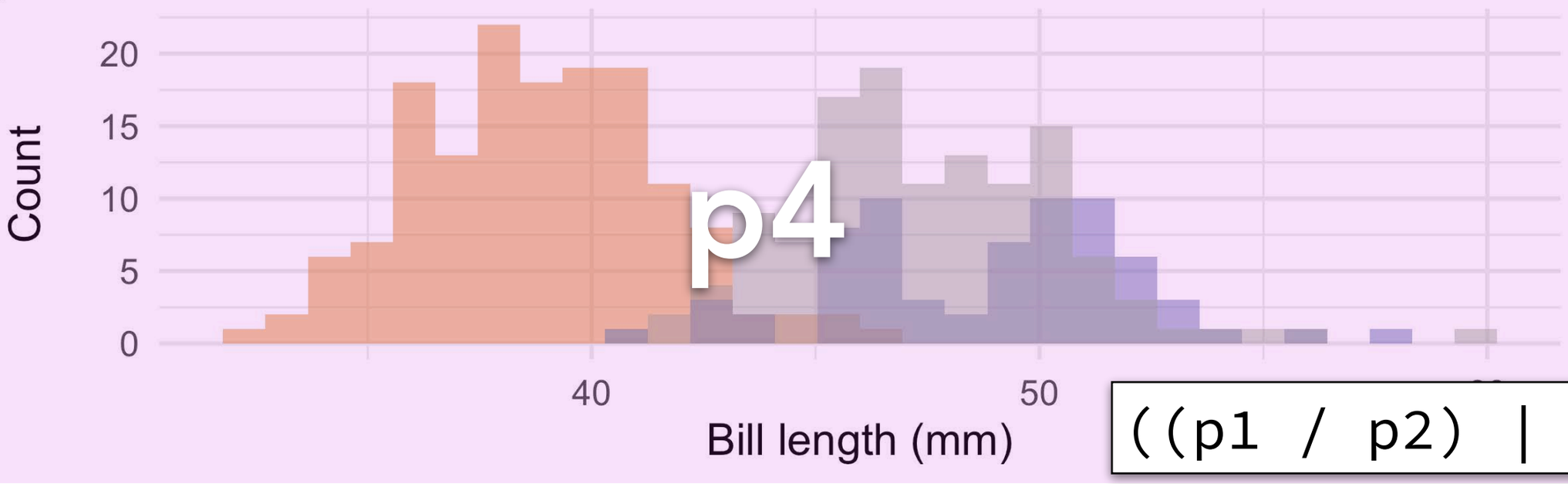


C

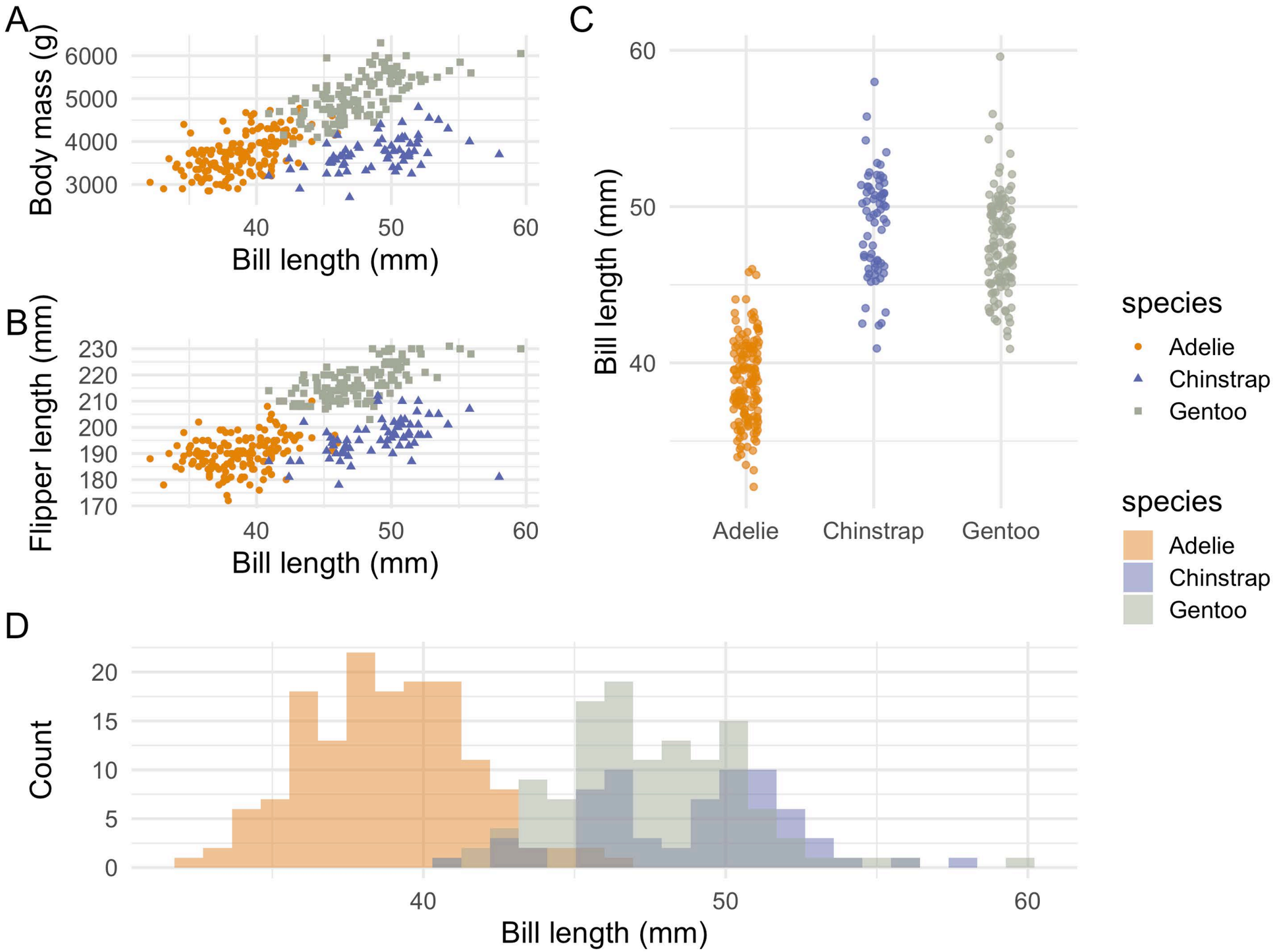


Third layer

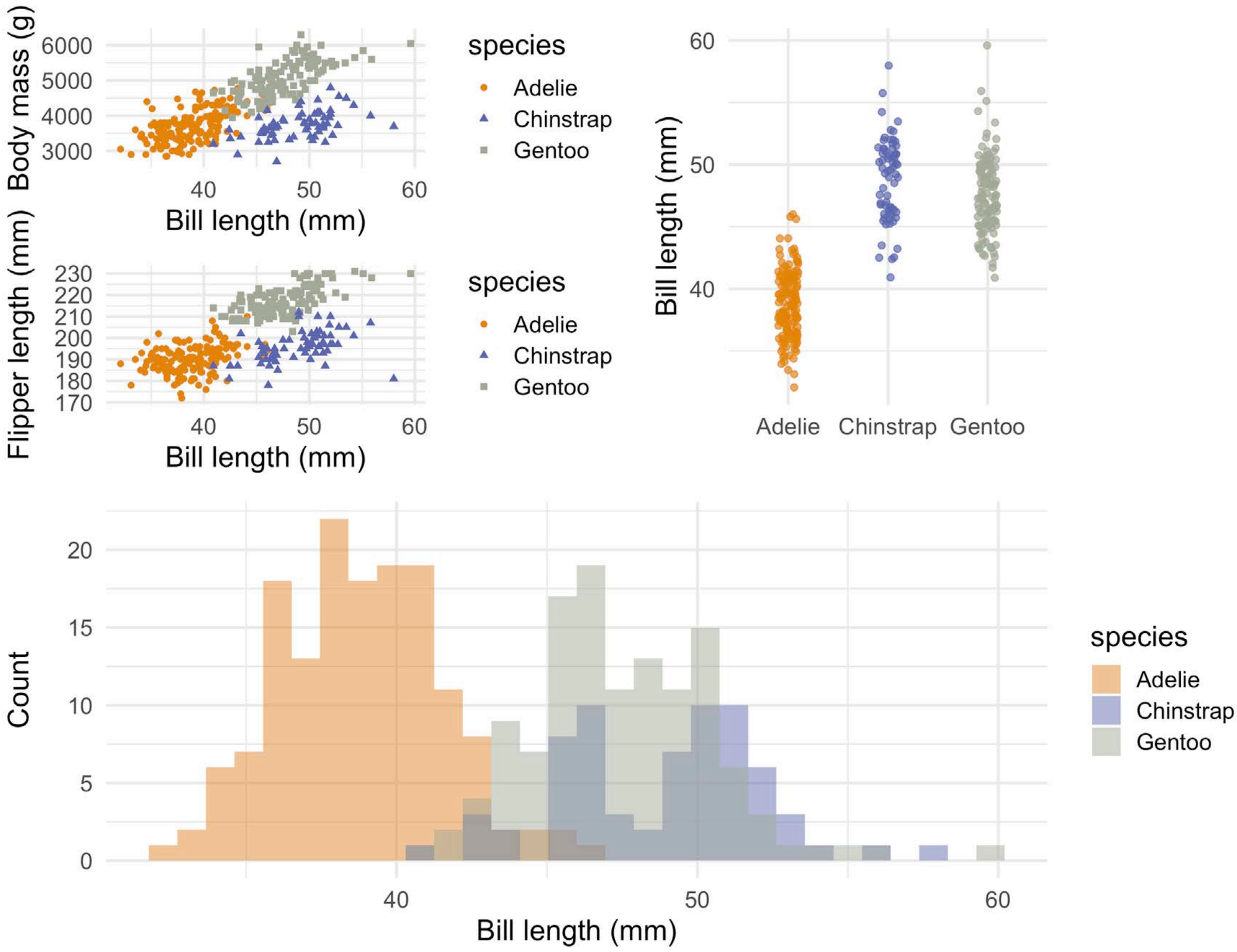
D



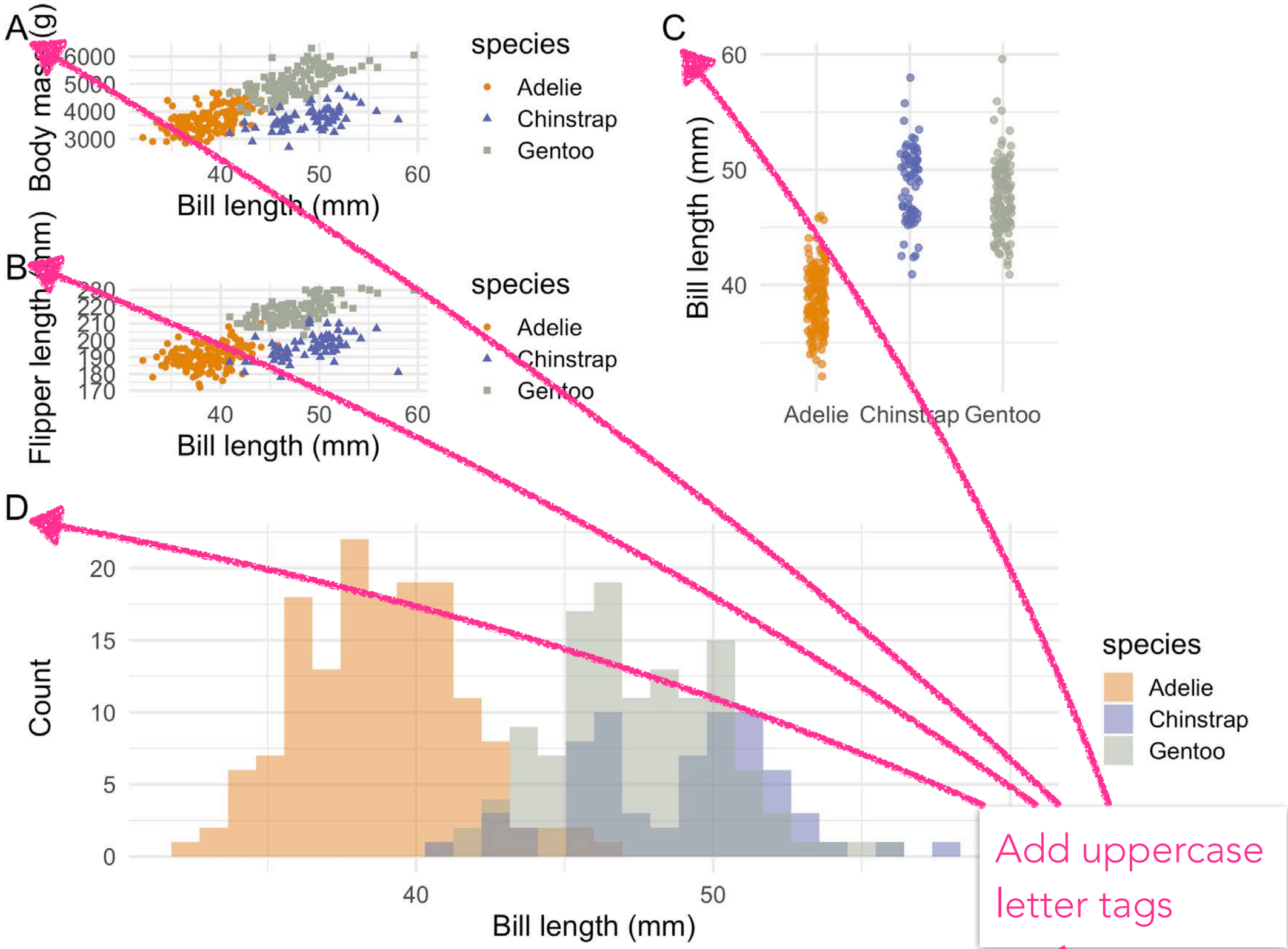

```
((p1 / p2) | p3) / p4
```


A
Body mass (g)
6000
5000
4000
3000
40
50
60
Bill length (mm)
B
Flipper length (mm)
230
220
210
200
190
180
170
40
50
60
Bill length (mm)
C
Bill length (mm)
60
50
40
Adelie
Chinstrap
Gentoo
species
Adelie
Chinstrap
Gentoo
species
Adelie
Chinstrap
Gentoo
D
Count
20
15
10
5
0
40
50
60
Bill length (mm)

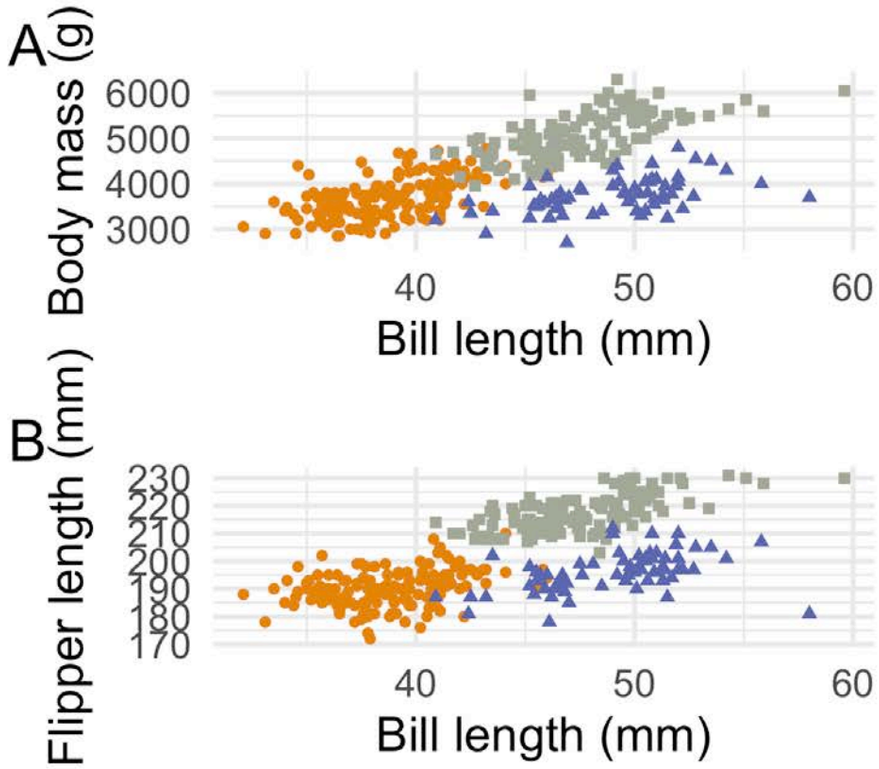
```
((p1 / p2) | p3) / p4 +
...
```

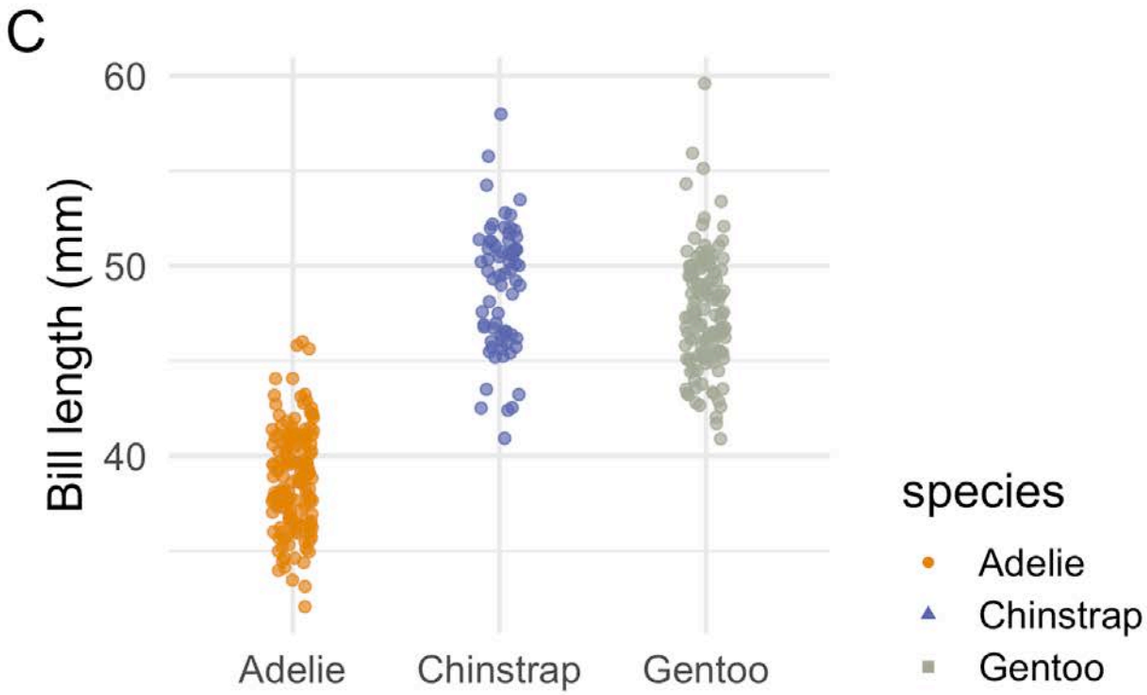

Body mass (g)
6000
5000
4000
3000
40
50
60
Bill length (mm)
species
Adelie
Chinstrap
Gentoo
Flipper length (mm)
230
220
210
200
190
180
170
40
50
60
Bill length (mm)
species
Adelie
Chinstrap
Gentoo
Bill length (mm)
60
50
40
Adelie
Chinstrap
Gentoo
Count
20
15
10
5
0
40
50
60
Bill length (mm)
species
Adelie
Chinstrap
Gentoo

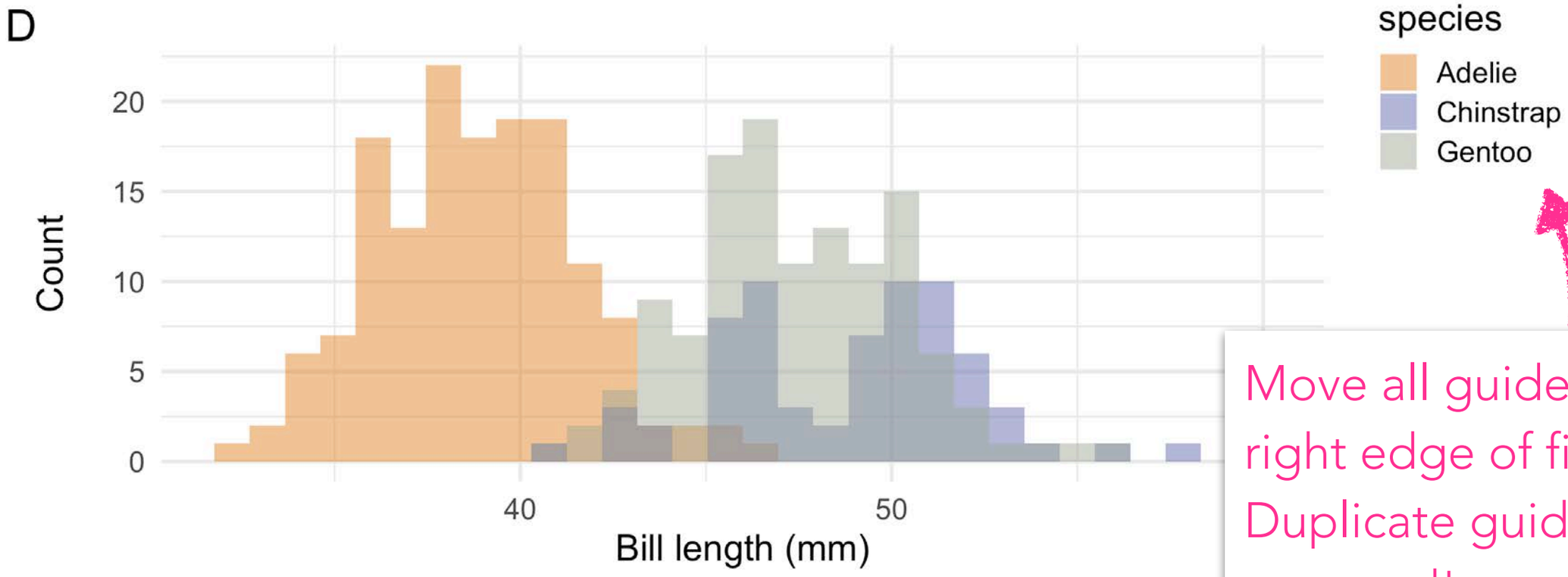
```
((p1 / p2) | p3) / p4
```




```
((p1 / p2) | p3) / p4
  plot_annotation(tag_levels = 'A')
```

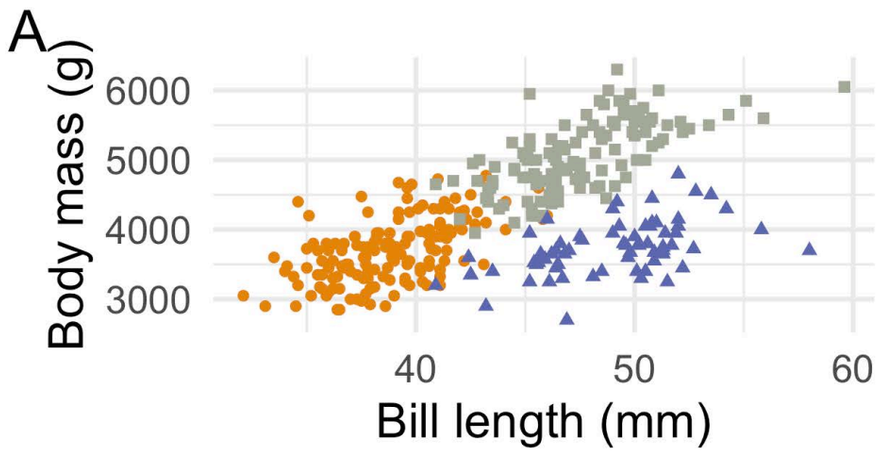


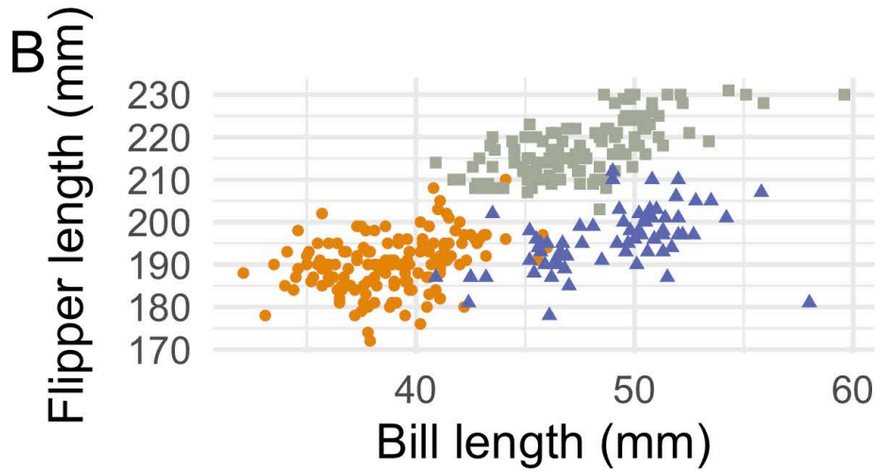


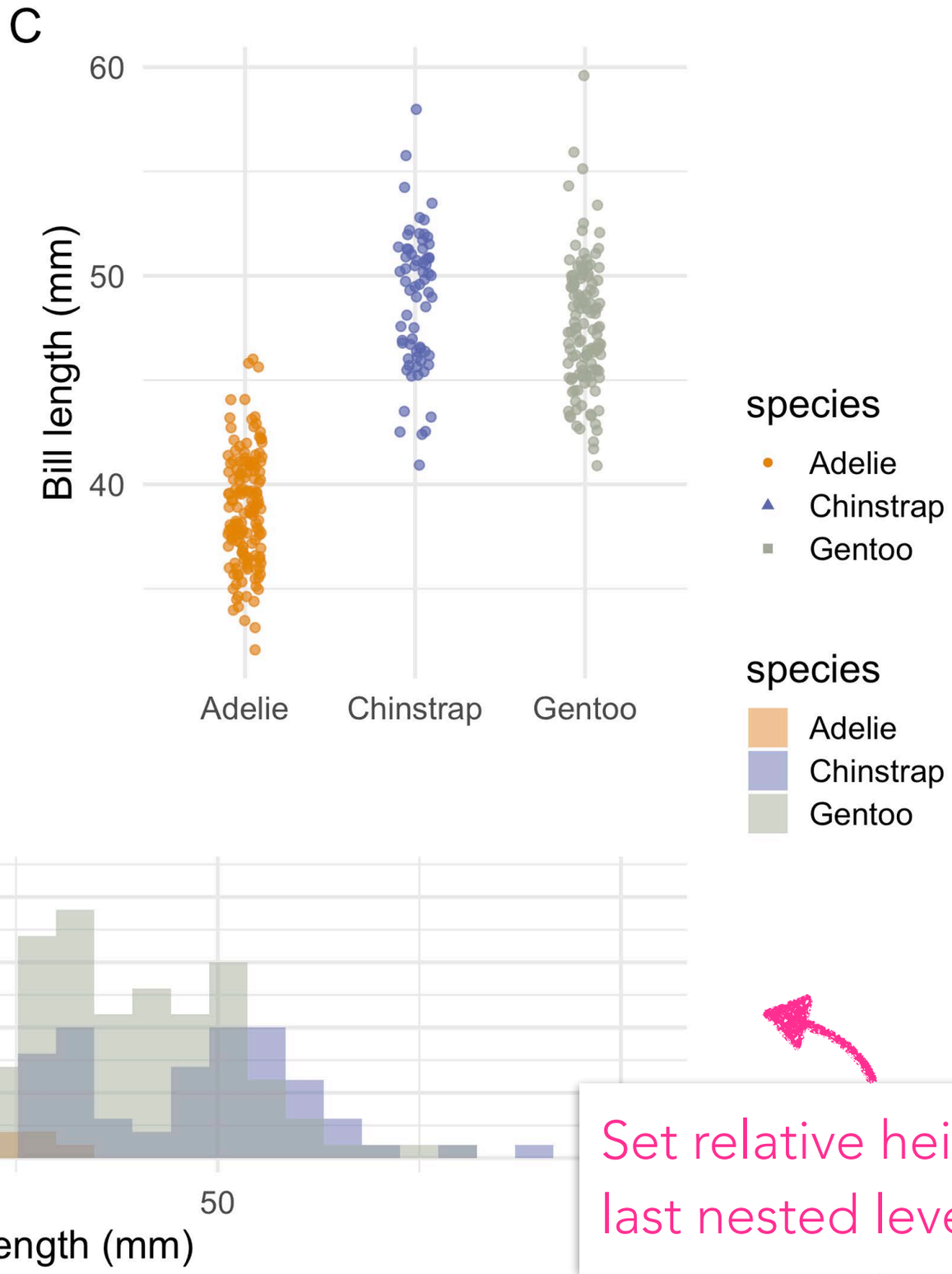


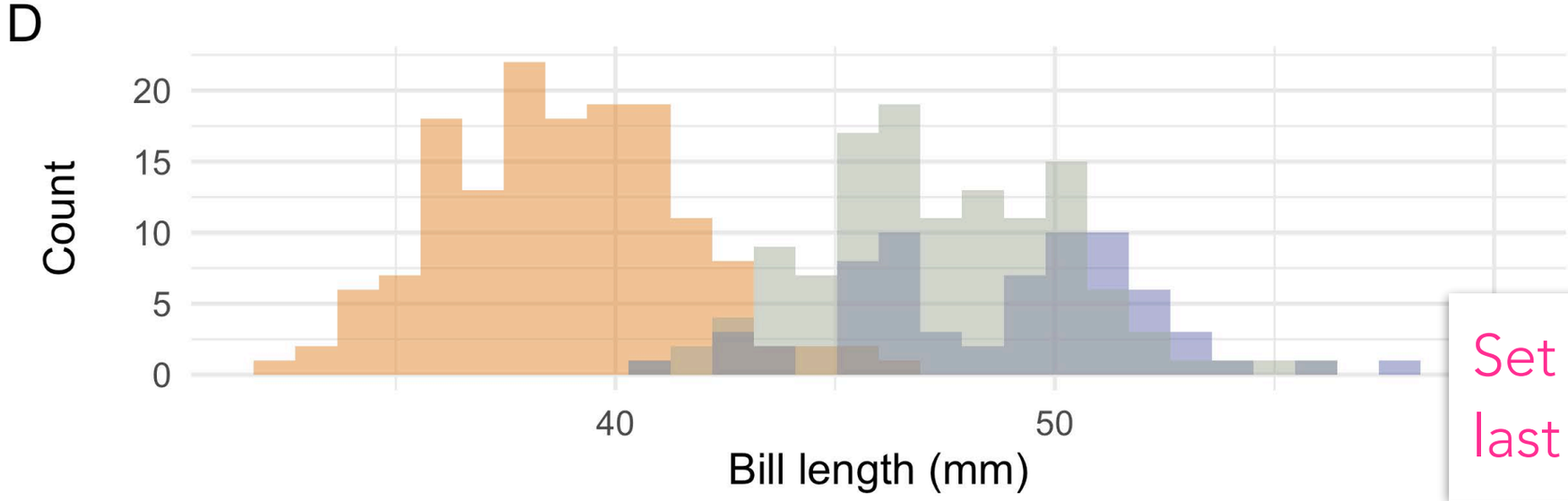
Move all guides to the right edge of figure. Duplicate guide removed!

```
((p1 / p2) | p3) / p4
  plot_annotation(tag_levels = 'A') +
  plot_layout(guides = "collect")
```









Set relative heights for last nested level

```
((p1 / p2) | p3) / p4
  plot_annotation(tag_levels = 'A') +
  plot_layout(guides = "collect", heights = c(2, 1))
```

SMALL PROJECT 1