What do I mean by `tidy` data?

Data are often stored in **tabular** (or matrix) form:

```
1 library(palmerpenguins)
2 penguins |> slice(1:5)
```

```
# A tibble: 5 × 8
  species island    bill_length_mm bill_depth_mm flipper_length_mm body_mass_g
  <fct>   <fct>              <dbl>         <dbl>             <int>       <int>
1 Adelie  Torgersen           39.1          18.7               181        3750
2 Adelie  Torgersen           39.5          17.4               186        3800
3 Adelie  Torgersen           40.3          18                 195        3250
4 Adelie  Torgersen           NA            NA                  NA          NA
5 Adelie  Torgersen           36.7          19.3               193        3450
# ℹ 2 more variables: sex <fct>, year <int>
```

Each row == an observation

each column is variable / measurement

The Grammar of Graphics

Originally defined by Leland Wilkinson

1. **data**
2. **geometries**: type of geometric objects to represent data, e.g., points, lines
3. **aesthetics**: visual characteristics of geometric objects to represent data, e.g., position, size
4. **scales**: how each aesthetic is converted into values on the graph, e.g., color scales
5. **stats**: statistical transformations to summarize data, e.g., counts, means, regression lines
6. **facets**: split data and view as multiple graphs
7. **coordinate system**: 2D space the data are projected onto, e.g., Cartesian coordinates

Hadley Wickham created `ggplot2`

1. `data`
2. `geom_X` geom_bar geom_point
3. `aes`: mappings of columns to geometric objects
4. `scale`: one scale for each `aes` variable
5. `stat` scale_size_manual()
6. `facet`
7. `coord`
8. `labs`: labels/guides for each variable and other parts of the plot, e.g., title, subtitle, caption
9. `theme`: customization of plot layout

Bar charts

```
1 library(tidyverse)
2 penguins |>
3   ggplot(aes(x = species)) +
4   geom_bar()
```

Spine charts - height version

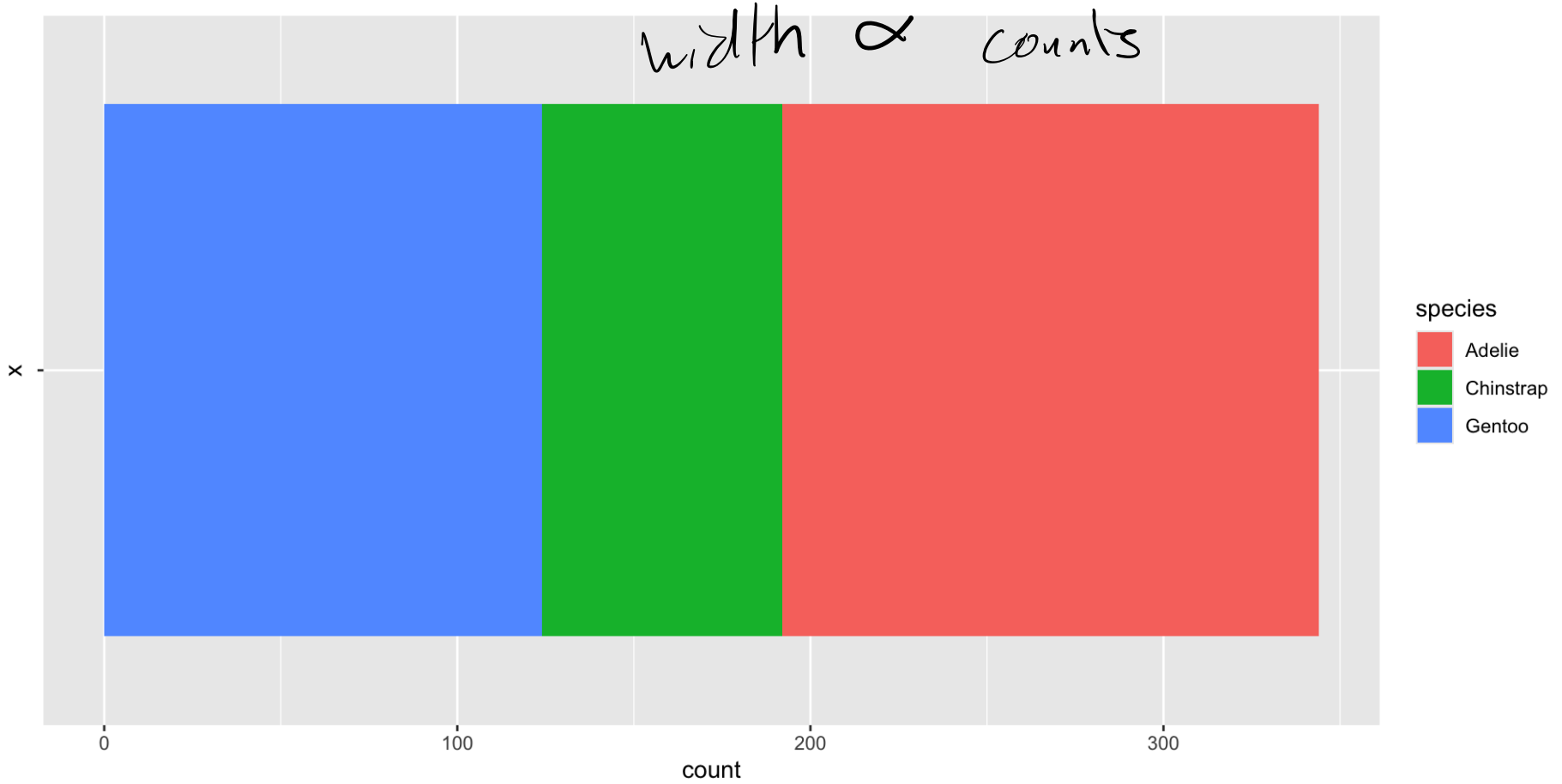
```
penguins |>
  ggplot(aes(fill = species, x = "")) +
  geom_bar()
```

height ∝ counts

width ∝ nothing

Spine charts - width version

```
penguins |>
  ggplot(aes(fill = species, x = "")) +
  geom_bar() +
  coord_flip()
```



So you want to make pie charts...

```
penguins |>
  ggplot(aes(fill = species, x = "")) +
  geom_bar(aes(y = after_stat(count))) +
  coord_polar(theta = "y") +
  theme_void()
```

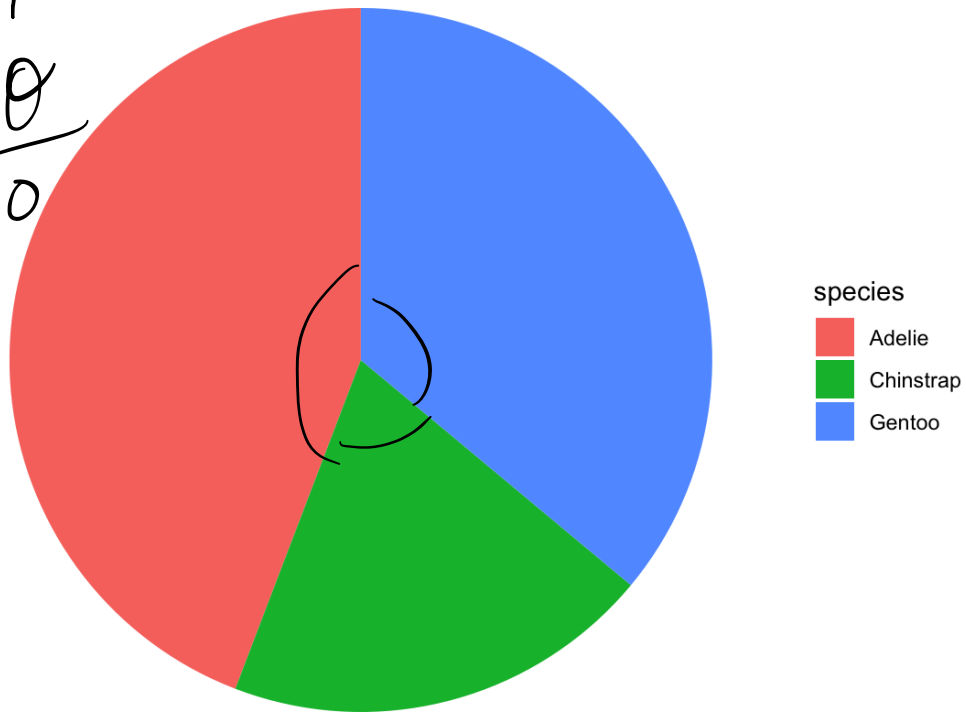
Total area = πr^2

slice area: $\frac{\pi r^2 \cdot \theta}{360}$

$\theta \propto$ counts

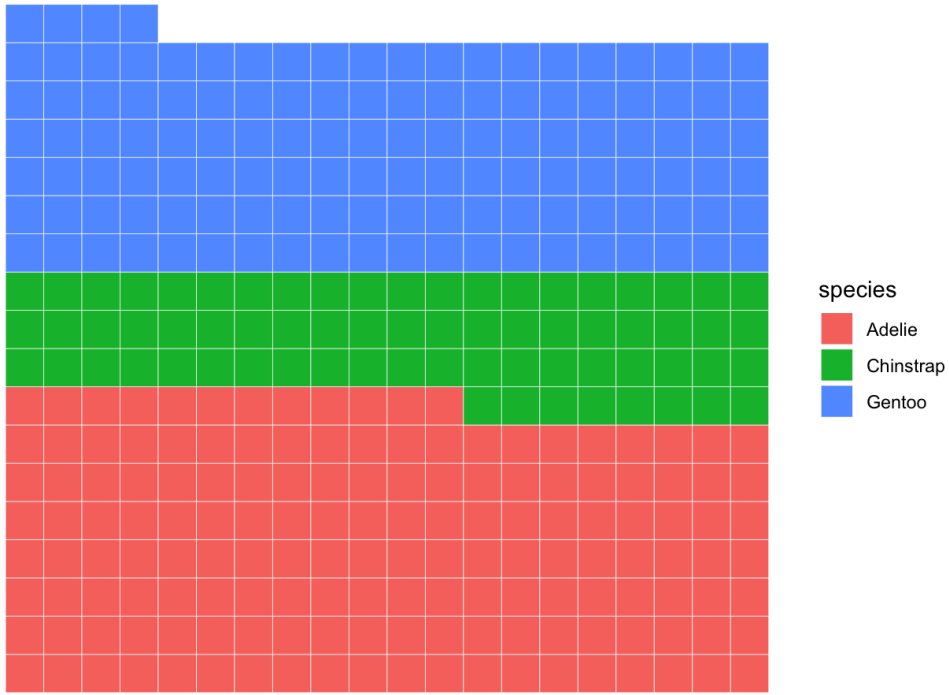
$r \propto 1$

radius



Waffle charts are cooler anyway...

```
library(waffle)
penguins |>
  group_by(species) |>
  summarize(count = n(), .groups = "drop") |>
  ggplot(aes(fill = species, values = count)) +
  geom_waffle(n_rows = 20, color = "white", flip = TRUE) +
  coord_equal() +
  theme_void()
```



Rose diagrams

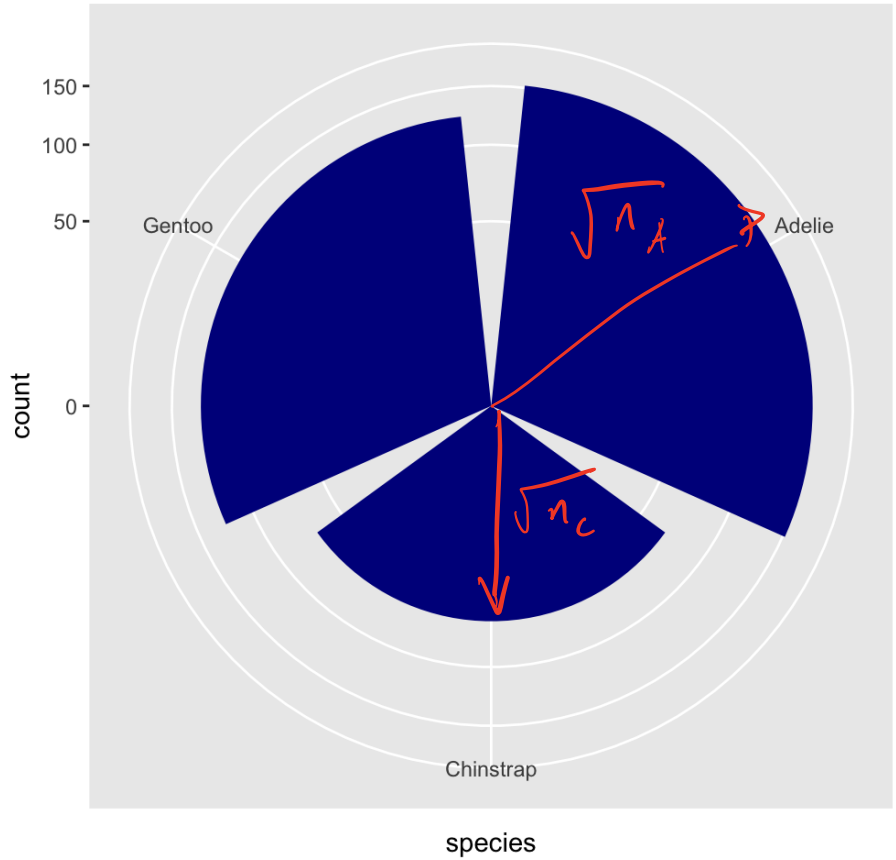
```
penguins |>
  ggplot(aes(x = species)) +
  geom_bar(fill = "darkblue") +
  coord_polar() +
  scale_y_sqrt()
```

$\theta \propto 1$

$\text{Area} \propto r^2 \cdot \theta \propto (\text{counts})^2$ if $r = \text{count}$

$r = \sqrt{\text{count}}$

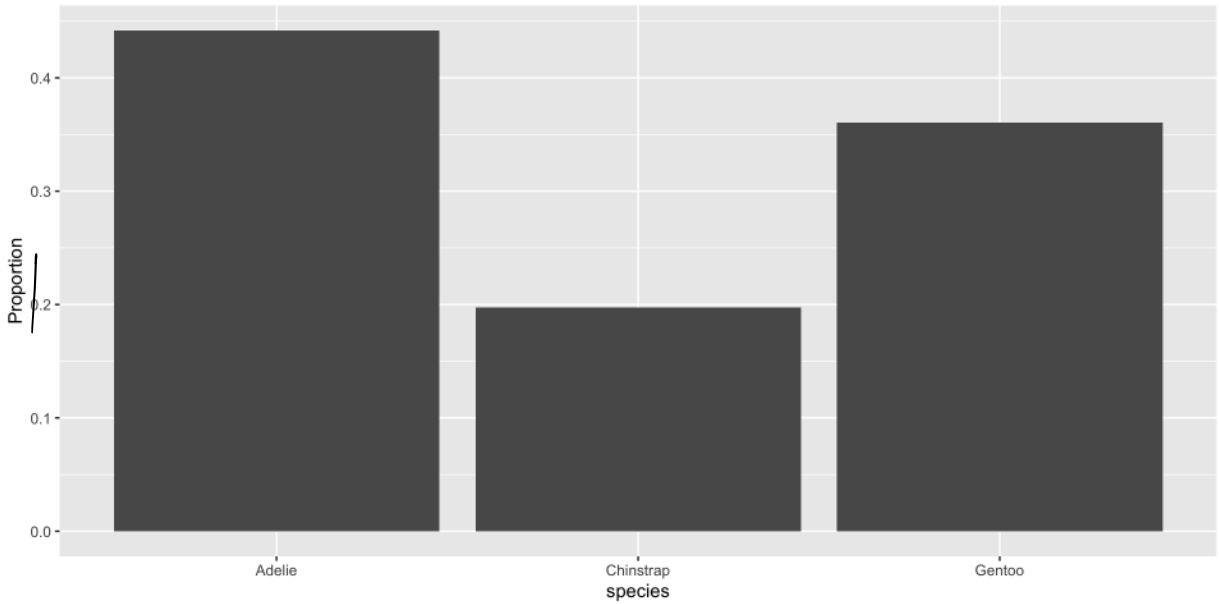
$\Rightarrow \text{Area} \propto \text{counts}$



Bar charts with proportions

- `after_stat()` indicates the aesthetic mapping is performed after statistical transformation
- Use `after_stat(count)` to access the `stat_count()` called by `geom_bar()`

```
1 penguins |>
2   ggplot(aes(x = species)) +
3   geom_bar(aes(y = after_stat(count) / sum(after_stat(count)))) +
4   labs(y = "Proportion")
```



Compute and display the proportions directly

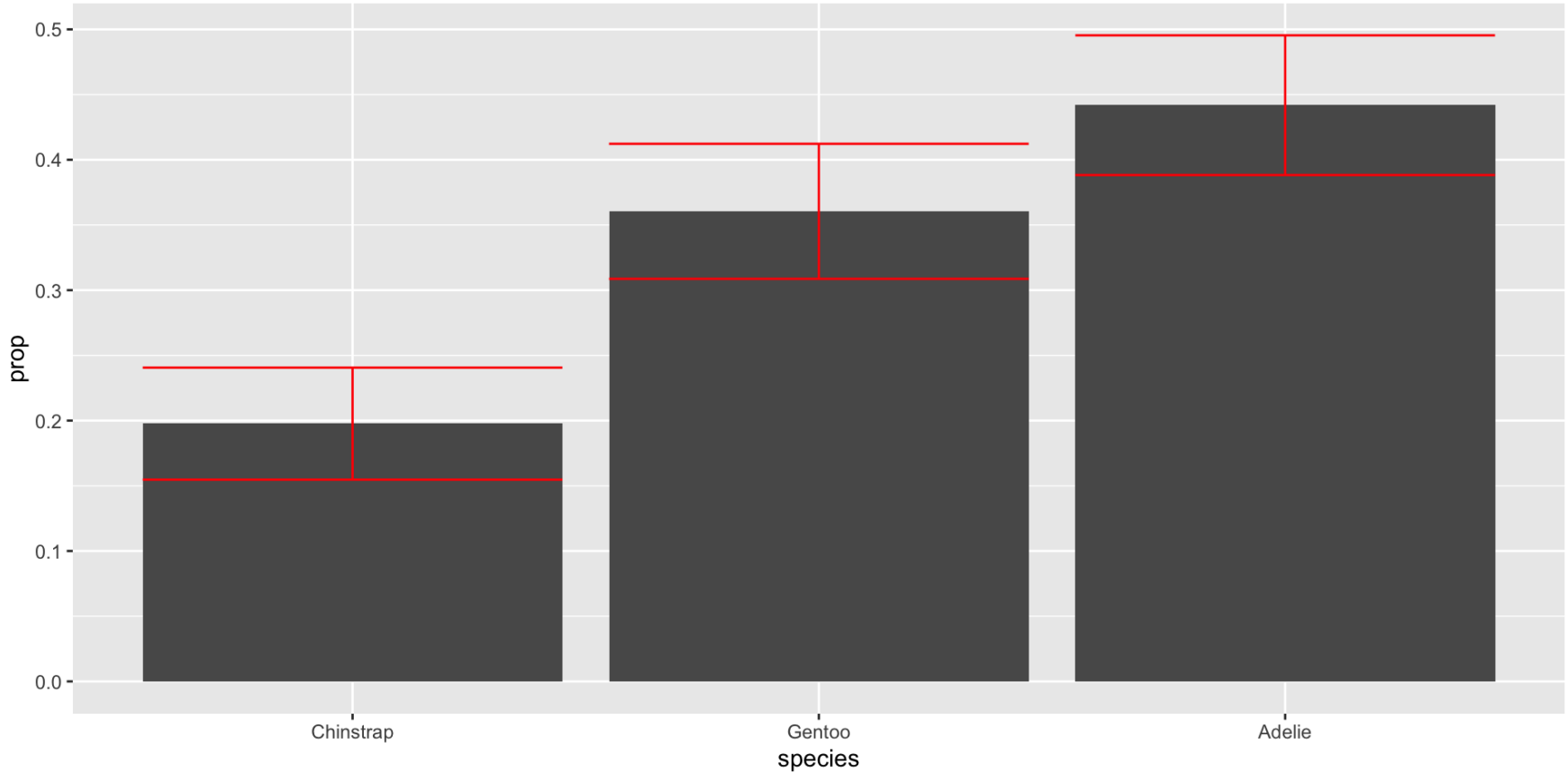
- Use `group_by()`, `summarize()`, and `mutate()` in a pipeline to compute then display the proportions directly
- Need to indicate we are displaying the `y` axis as given, i.e., the identity function

```
penguins |>
  group_by(species) |>
  summarize(count = n(), .groups = "drop") |>
  mutate(total = sum(count),
         prop = count / total) |>
  ggplot(aes(x = species)) +
  geom_bar(aes(y = prop), stat = "identity")
```

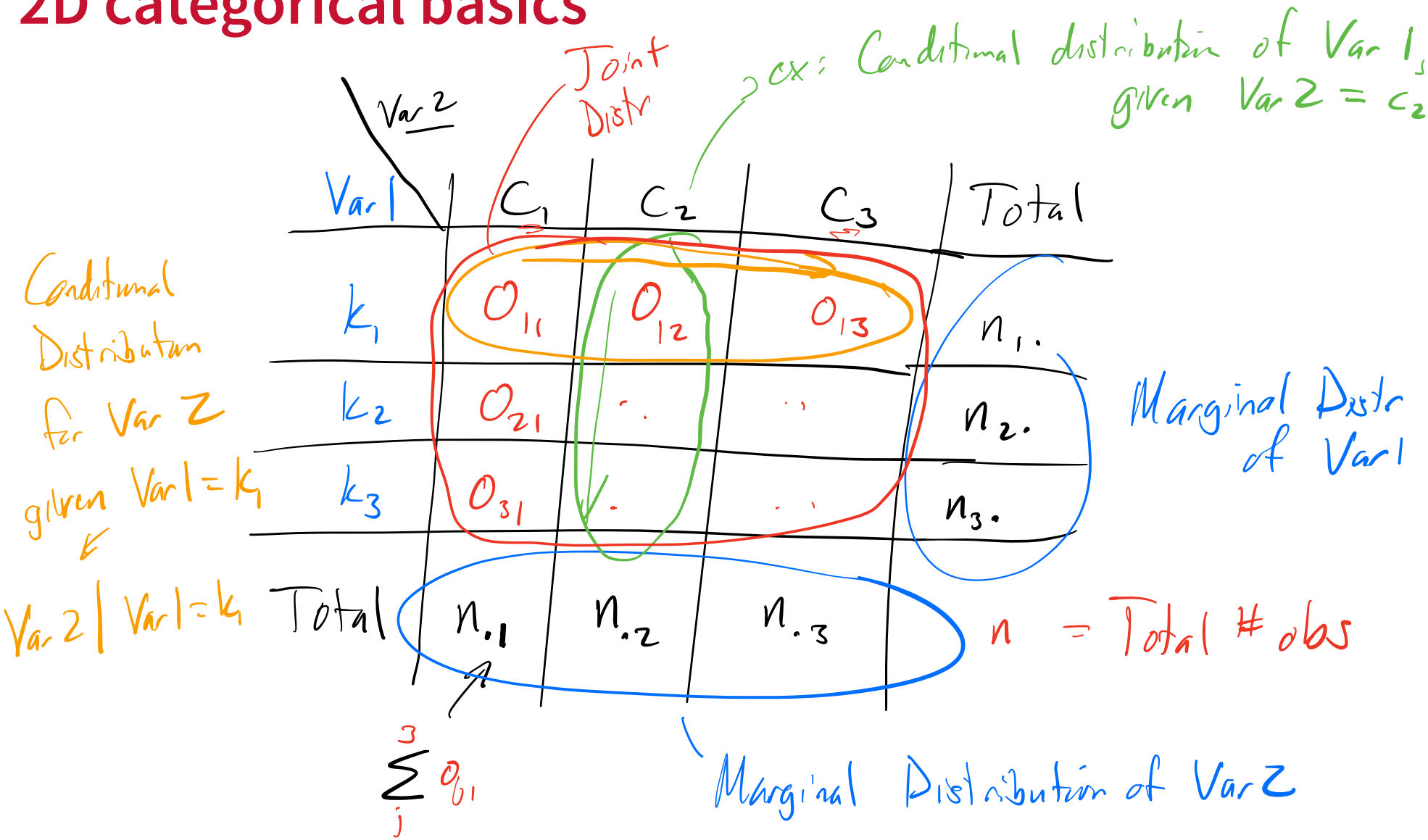
Useful to order categories by frequency with `forcats`

```
penguins |>
  group_by(species) |>
  summarize(count = n(), .groups = "drop") |>
  mutate(total = sum(count),
         prop = count / total,
         se = sqrt(prop * (1 - prop) / total),
         lower = prop - 2 * se,
         upper = prop + 2 * se,
         species = fct_reorder(species, prop)) |>
  ggplot(aes(x = species)) +
  geom_bar(aes(y = prop), stat = "identity") +
  geom_errorbar(aes(ymin = lower, ymax = upper),
                color = "red")
```

Useful to order categories by frequency with `forcats`



2D categorical basics



2D categorical basics

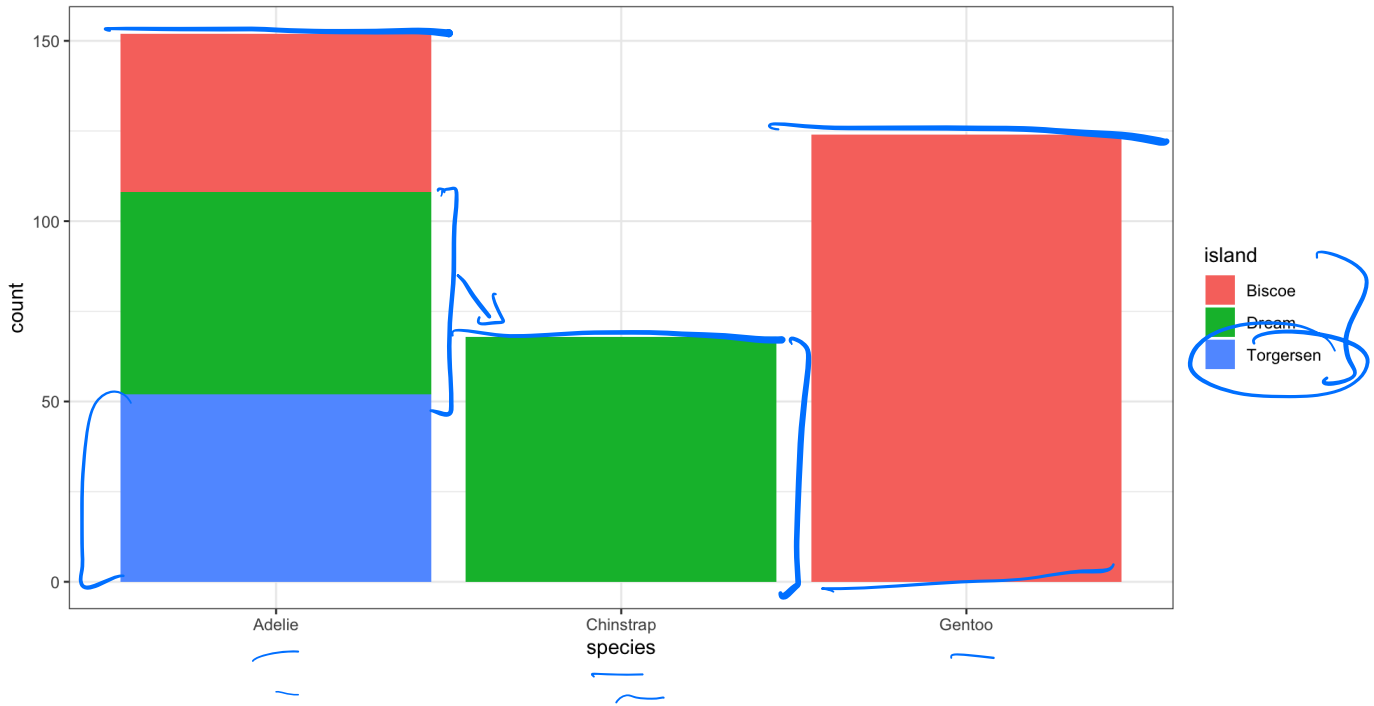
```
1 addmargins(table("Species" = penguins$species, "Island" = penguins$island))
```

```
           Island
Species     Biscoe Dream Torgersen Sum
  Adelie        44    56        52 152
  Chinstrap      0    68         0  68
  Gentoo       124     0         0 124
  Sum          168   124        52 344
```

- Column and row sums: marginal distributions
- Values within rows: conditional distribution for `Island` given `Species`
- Values within columns: conditional distribution for `Species` given `Island`
- Bottom right: total number of observations

Stacked bar charts - a bar chart of spine charts

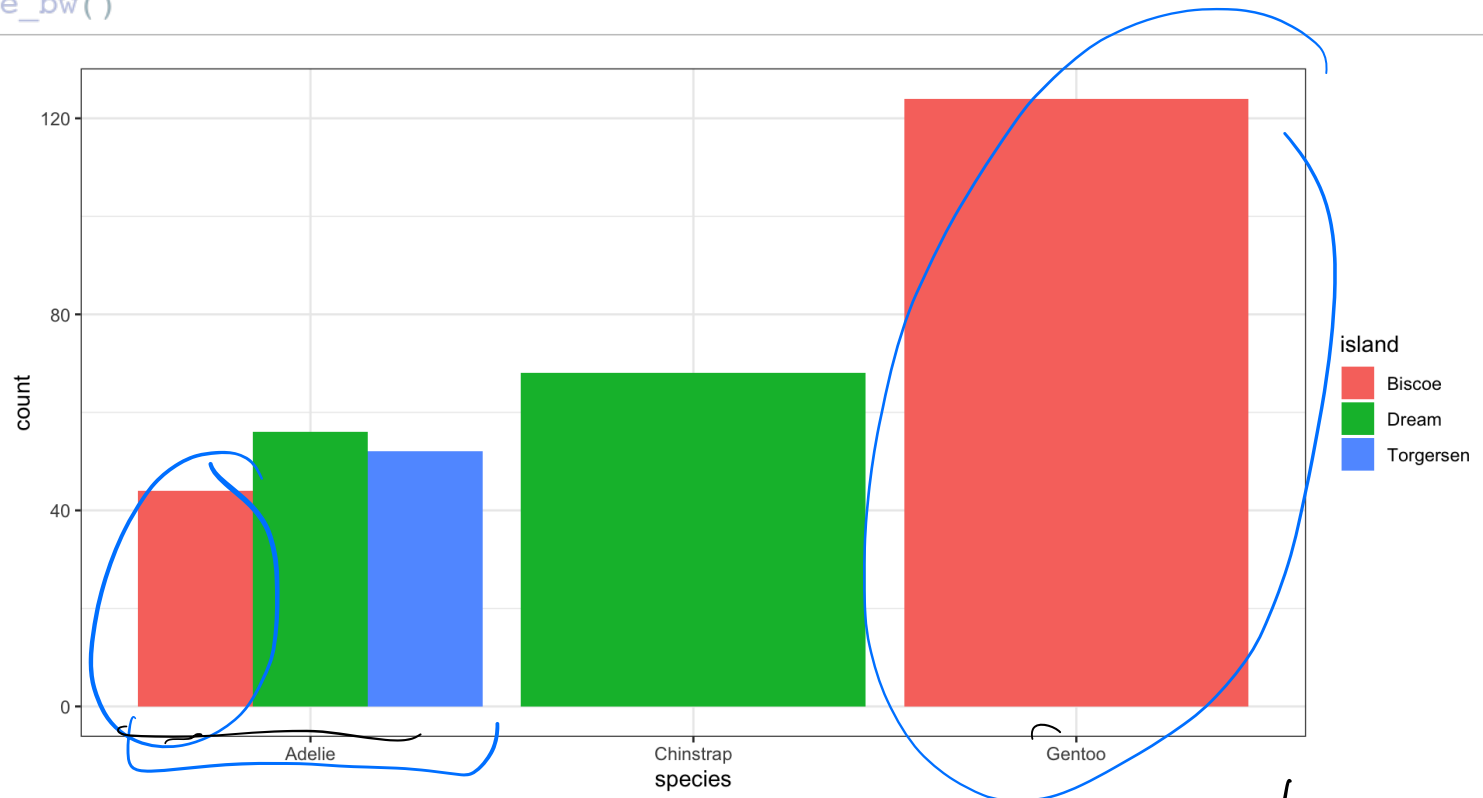
```
penguins |>
  ggplot(aes(x = species, fill = island)) +
  geom_bar() +
  theme_bw()
```



- Easy to see marginal of `species`, i.e., $P($ `x` $)$
- Can see conditional of `island` | `species`, i.e., $P($ `fill` $|$ `x` $)$
- Harder to see conditional of `species` | `island`, i.e., $P($ `x` $|$ `fill` $)$

Side-by-side bar charts

```
penguins |>
  ggplot(aes(x = species, fill = island)) +
  geom_bar(position = "dodge") +
  theme_bw()
```



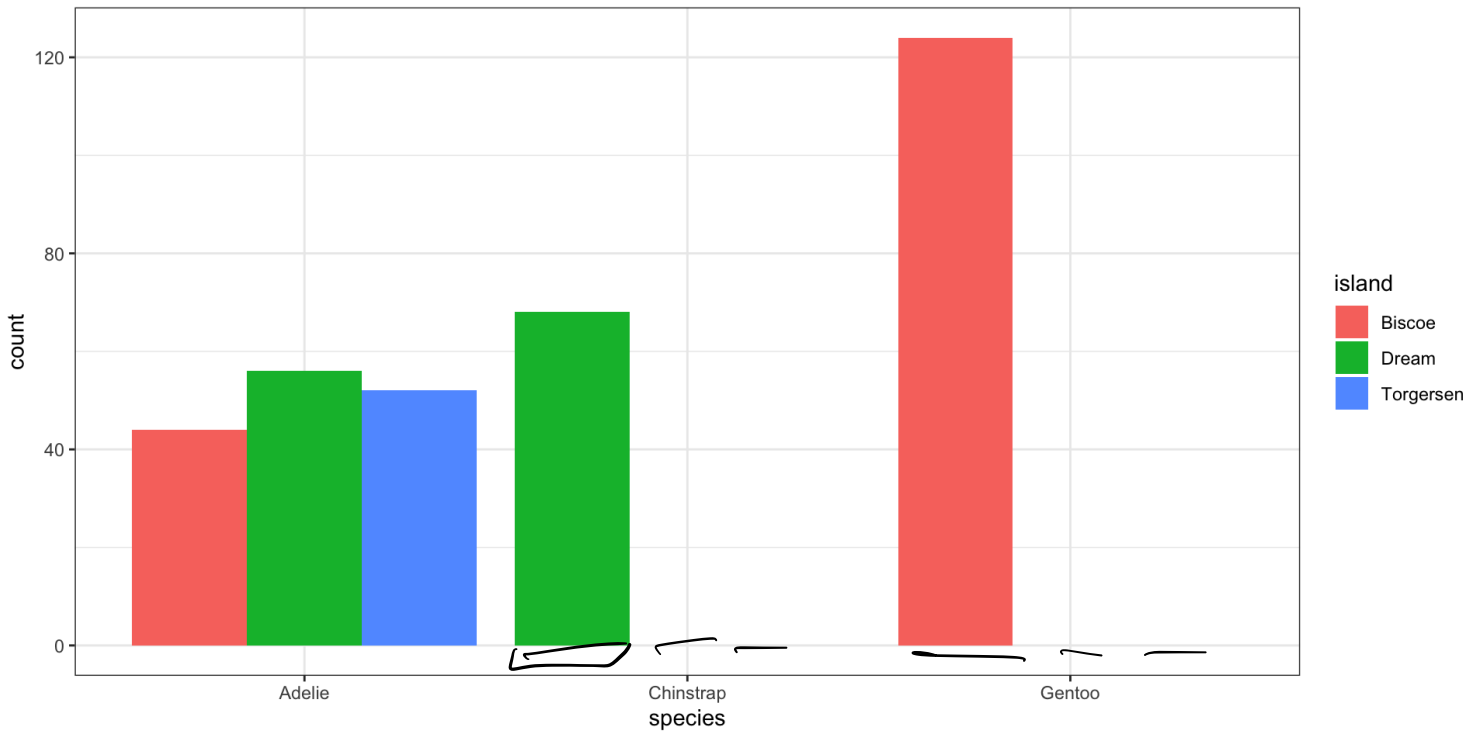
- Easy to see conditional of `island` | `species`, i.e., $P($ `fill` $|$ `x` $)$
- Can see conditional of `species` | `island`, i.e., $P($ `x` $|$ `fill` $)$

We can also see: joint

Hard to see: marginal

Side-by-side bar charts

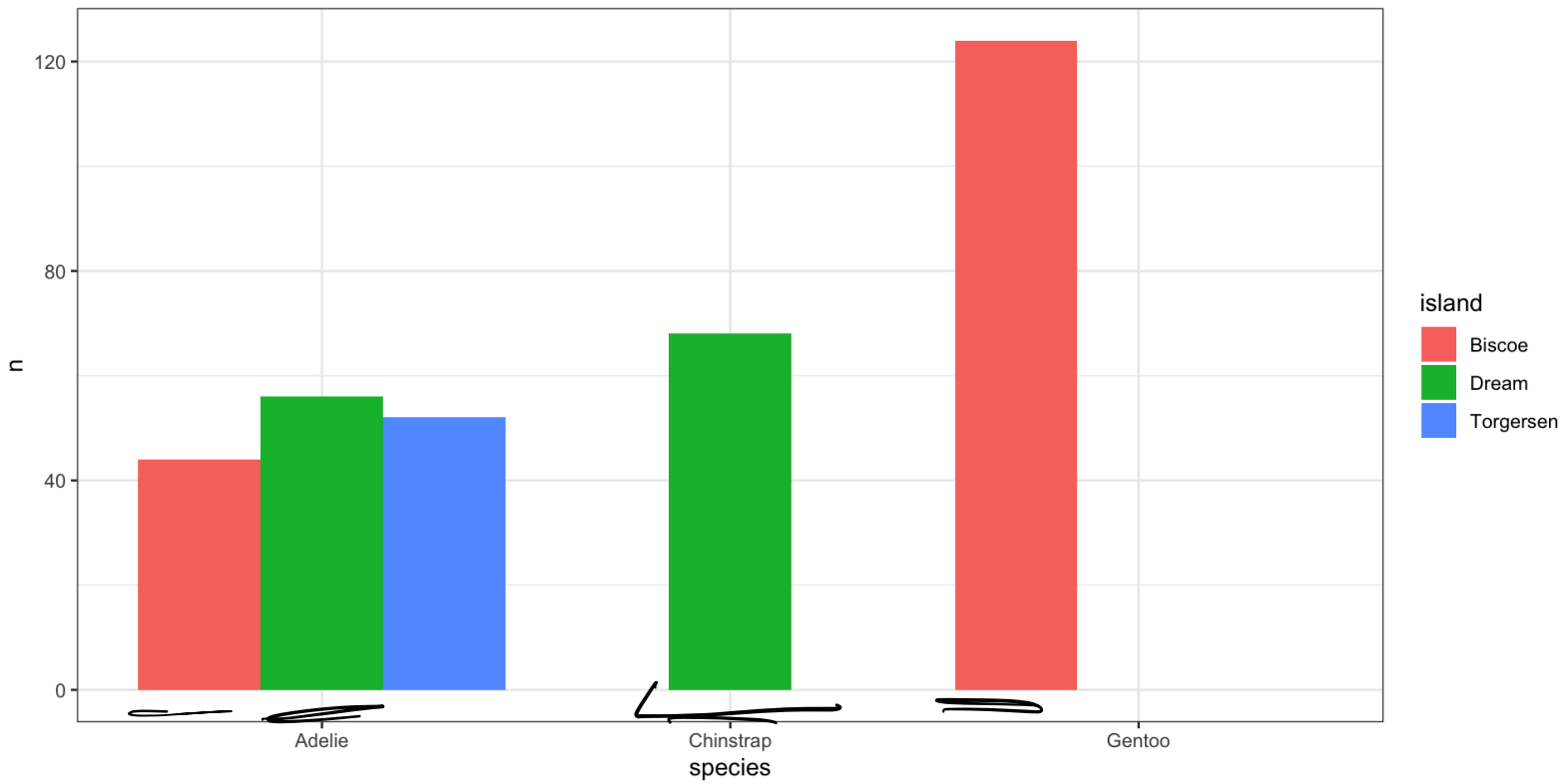
```
penguins |>
  ggplot(aes(x = species, fill = island)) +
  geom_bar(position = position_dodge(preserve = "single")) +
  theme_bw()
```



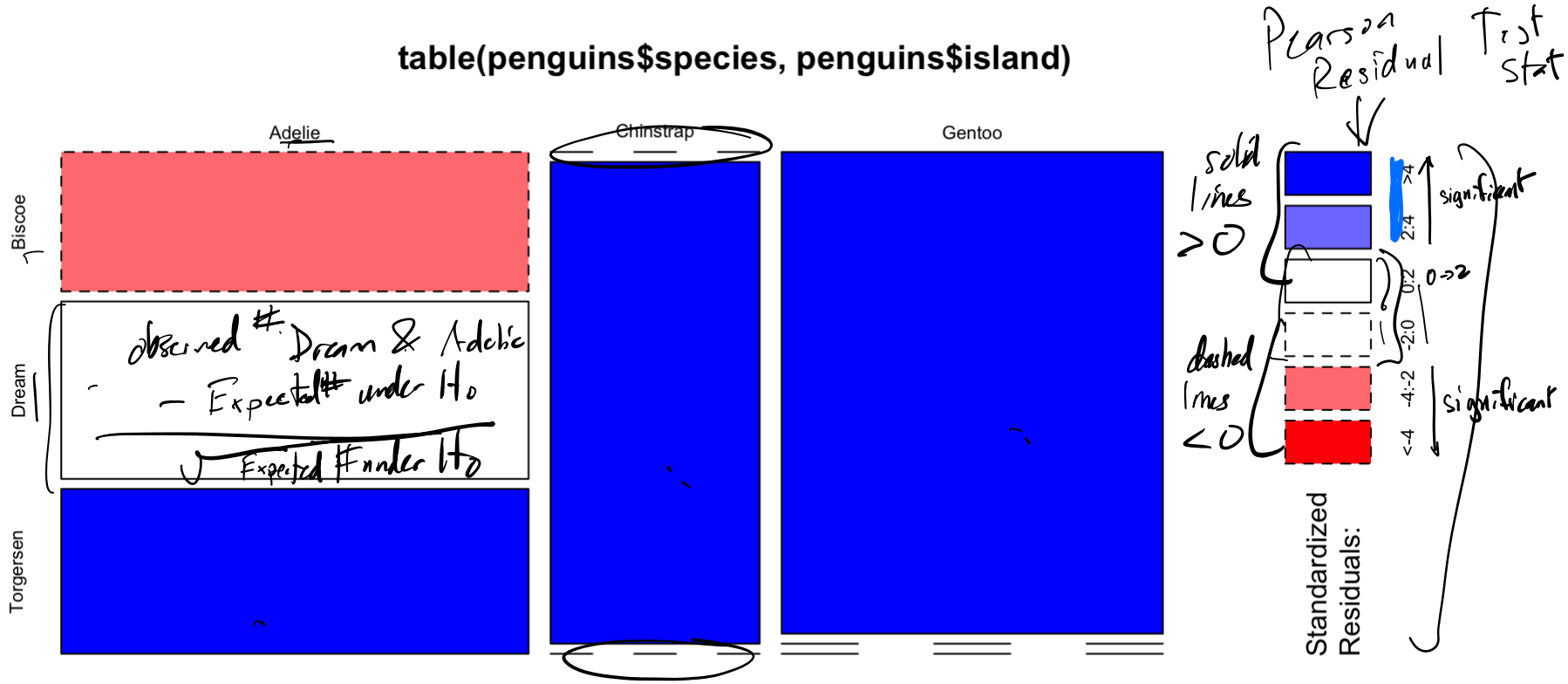
- Easy to see conditional of `island` | `species`, i.e., $P($ `fill` $|$ `x` $)$
- Can see conditional of `species` | `island`, i.e., $P($ `x` $|$ `fill` $)$

Complete missing values to preserve location

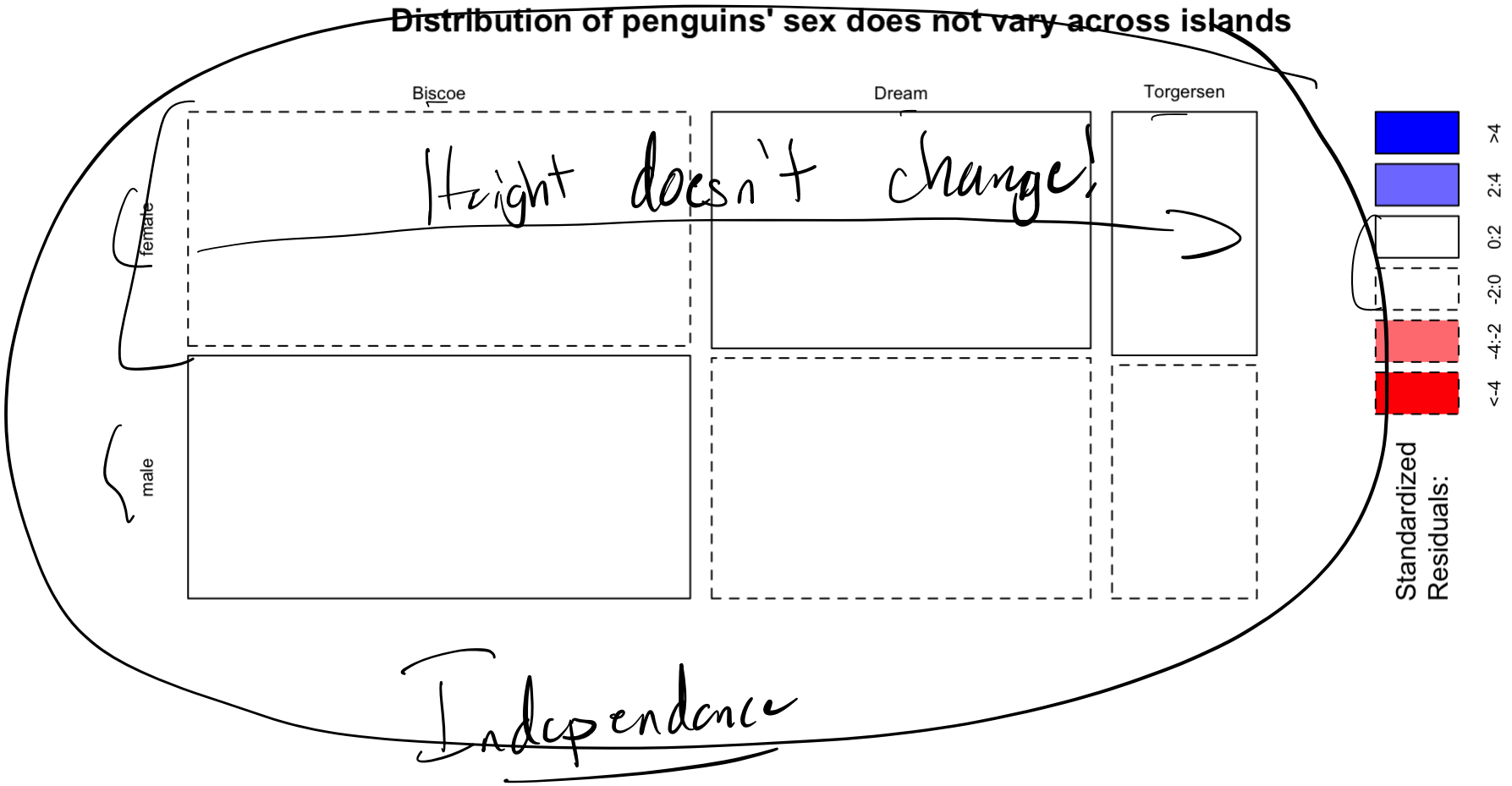
```
penguins |>
  count(species, island) |>
  complete(species = unique(species), island = unique(island),
           fill = list(n = 0)) |>
  ggplot(aes(x = species, y = n, fill = island)) +
  geom_bar(stat = "identity", position = "dodge") +
  theme_bw()
```



```r
mosaicplot(table(penguins$species, penguins$island), shade = TRUE)
```

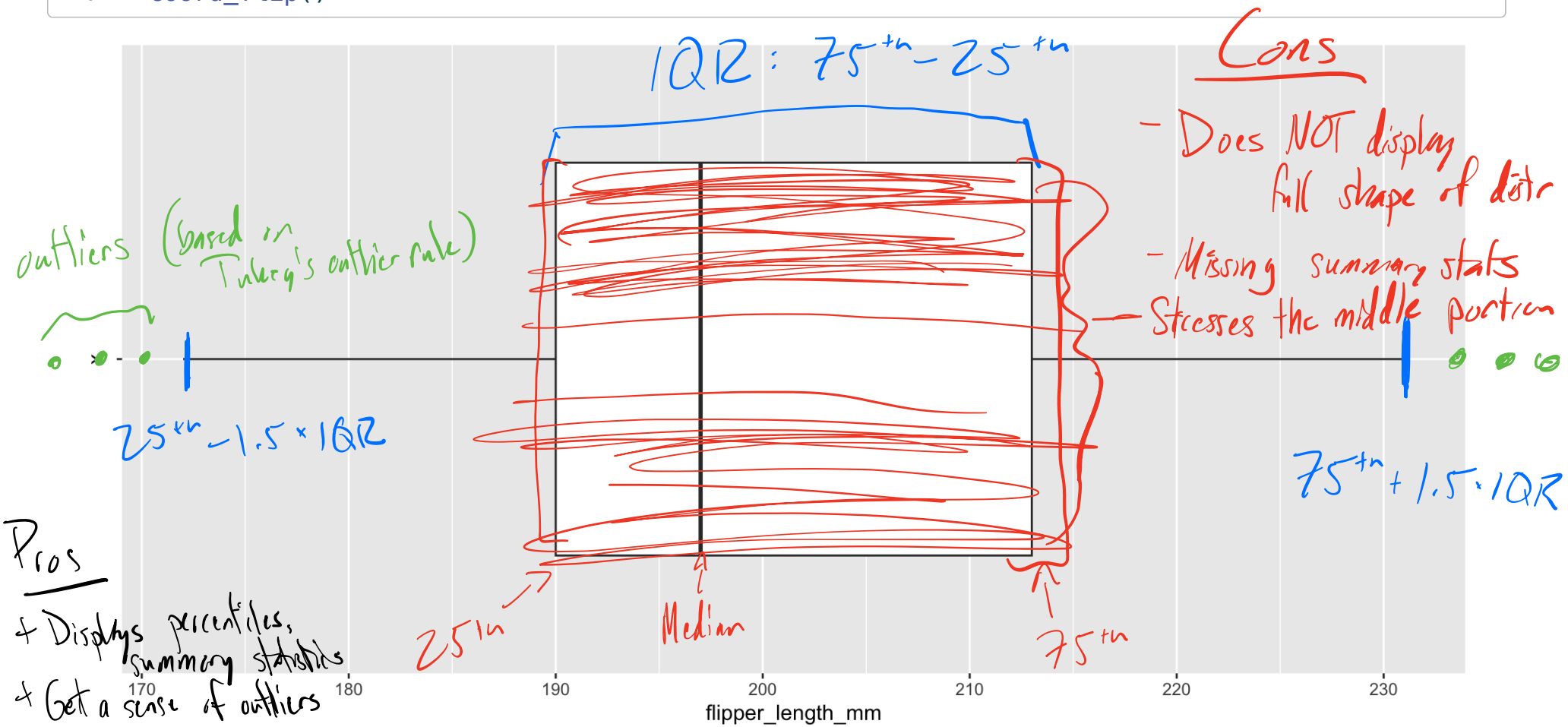


```
mosaicplot(table(penguins$island, penguins$sex), shade = TRUE,
           main = "Distribution of penguins' sex does not vary across islands")
```



Box plots visualize summary statistics

```
1 penguins |>
2   ggplot(aes(y = flipper_length_mm)) +
3   geom_boxplot(aes(x = "")) +
4   coord_flip()
```

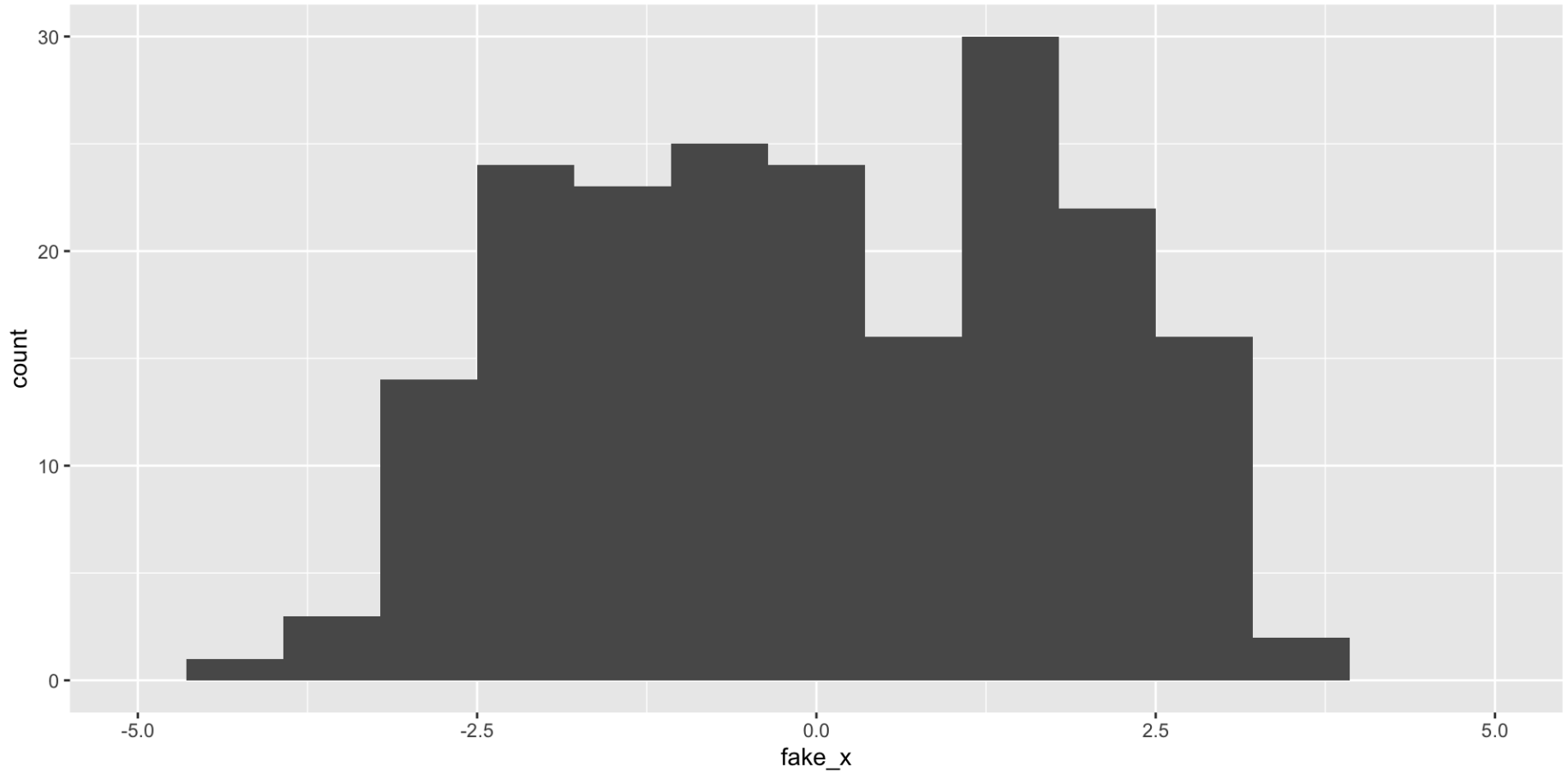


Pros

+ Displays percentiles, summary statistics
+ Get a sense of outliers
+ Useful for side-by-side

What happens as we change the number of bins?

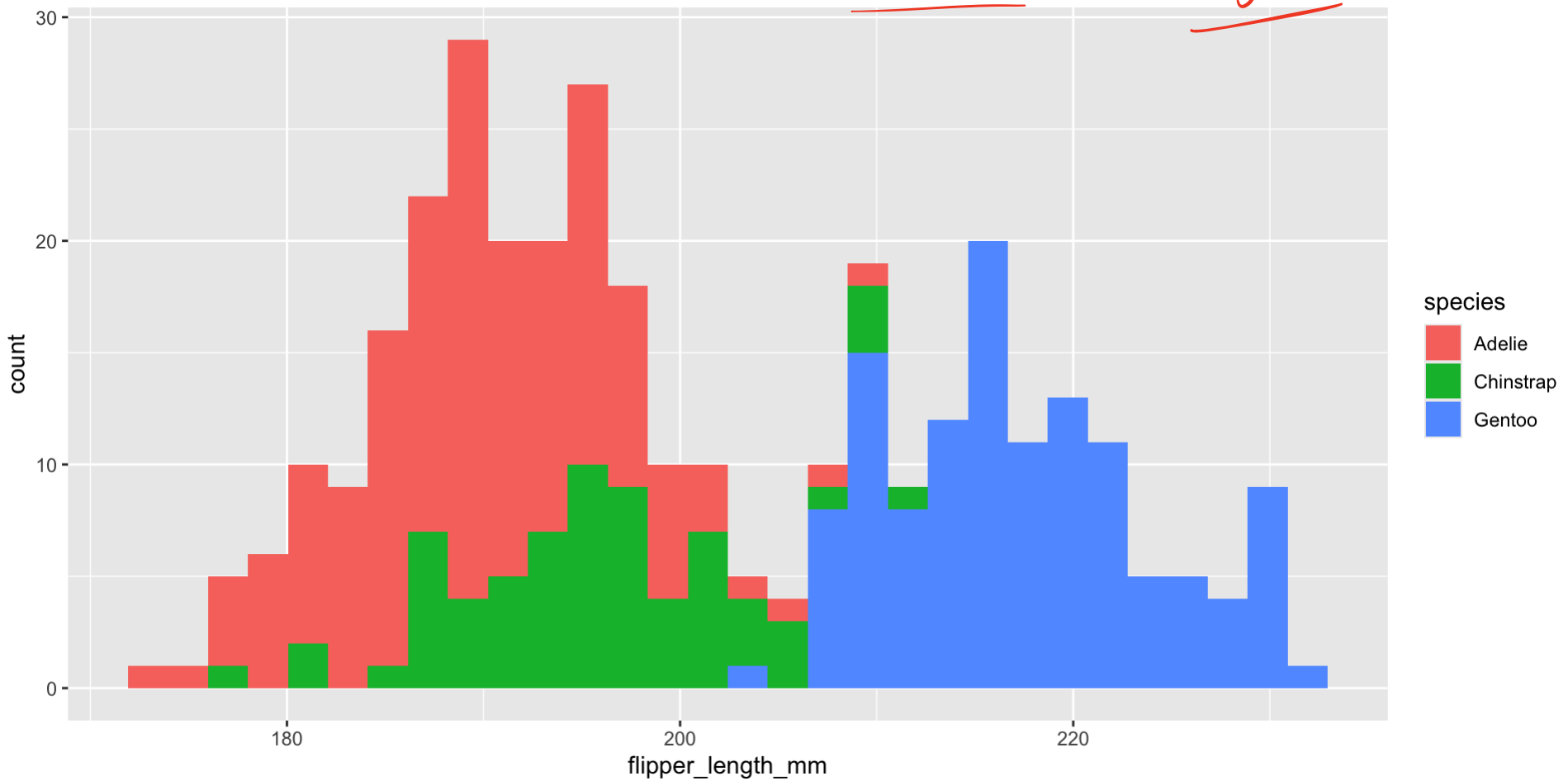
```
fake_data |>
  ggplot(aes(x = fake_x)) +
  geom_histogram(bins = 15) +
  scale_x_continuous(limits = c(-5, 5))
```



What about displaying conditional distributions?

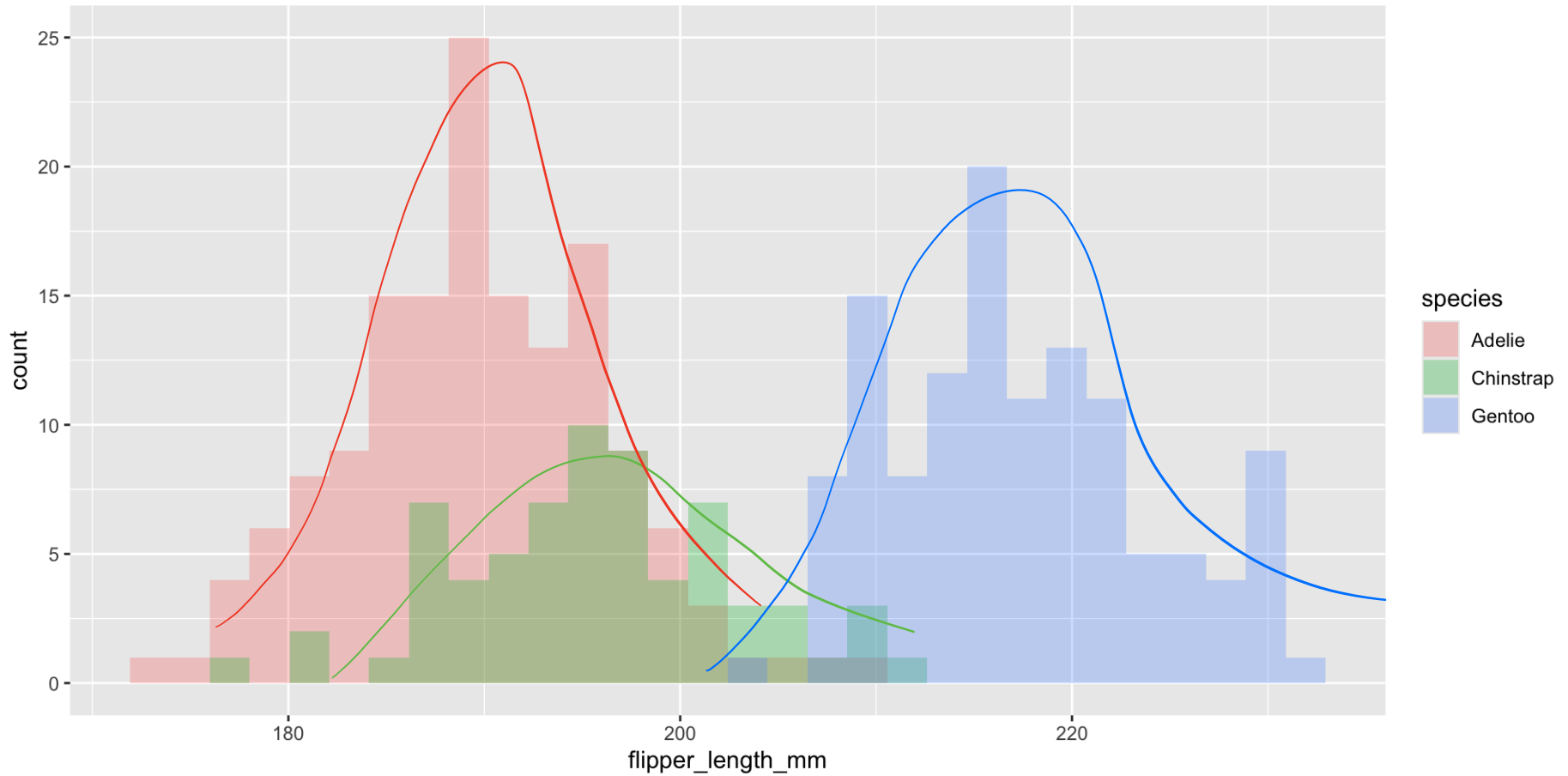
```
1 penguins |>
2   ggplot(aes(x = flipper_length_mm)) +
3   geom_histogram(aes(fill = species))
```

→ Stacking → Marginal



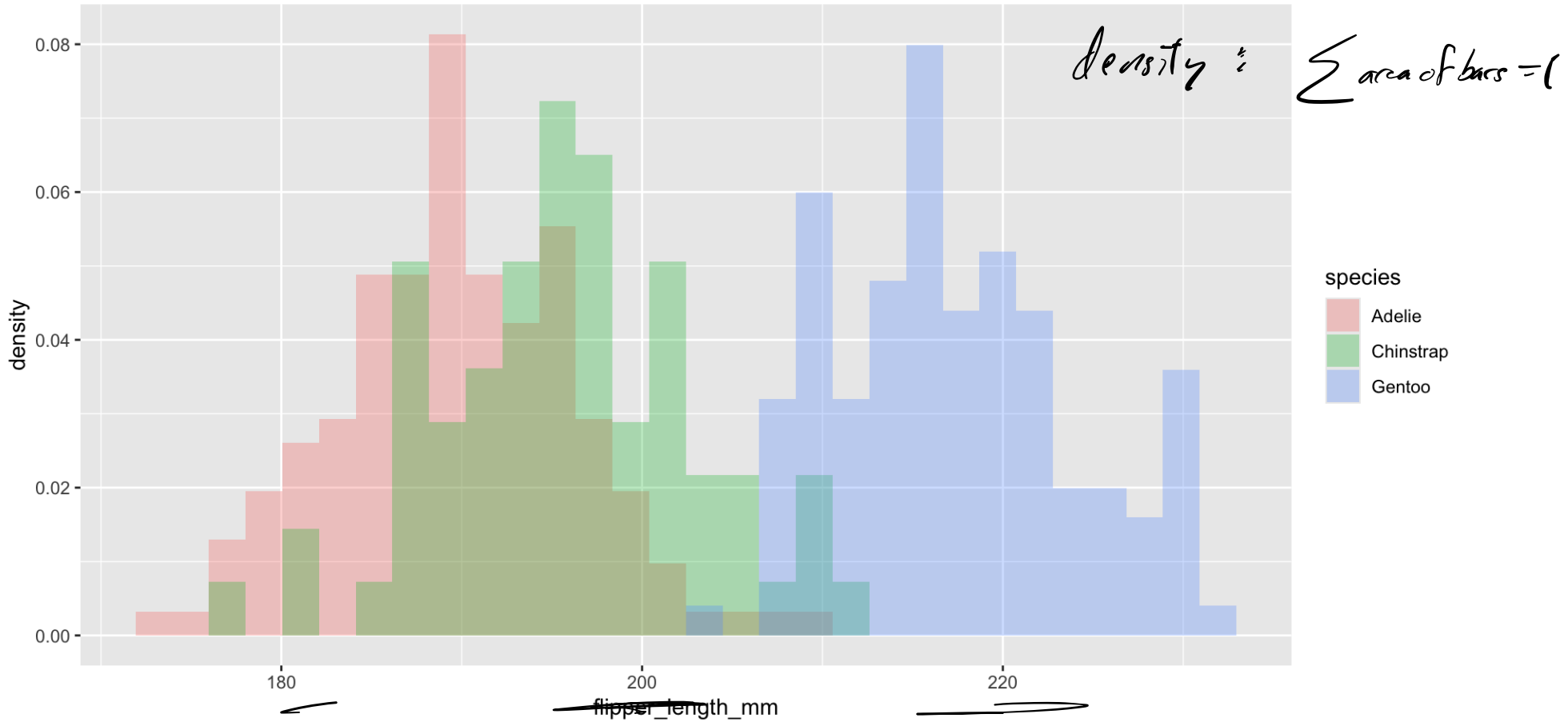
What about displaying conditional distributions?

```
penguins |>
  ggplot(aes(x = flipper_length_mm)) +
  geom_histogram(aes(fill = species),
                 position = "identity", alpha = 0.3)
```



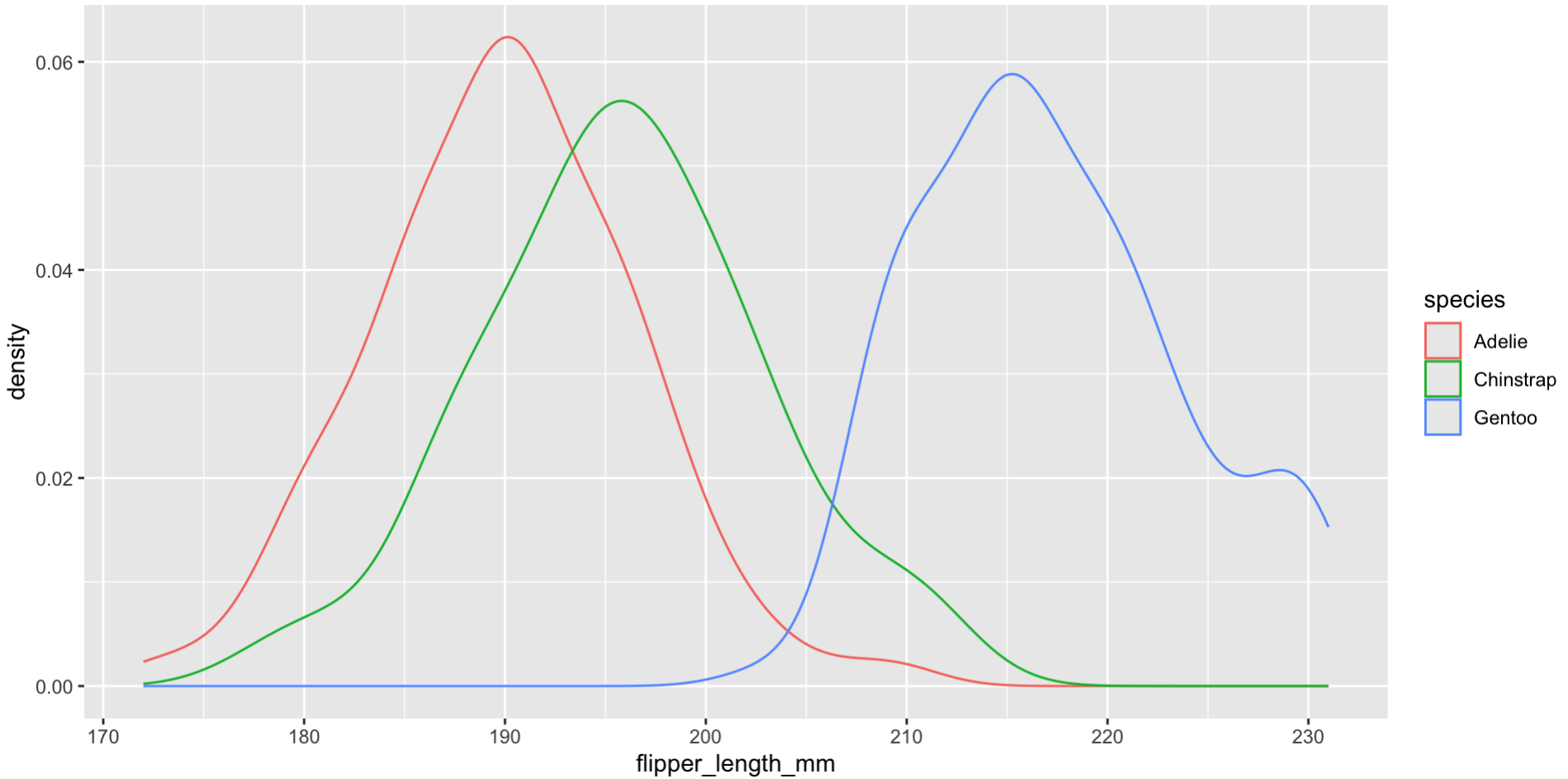
Normalize histogram frequencies with density

```
penguins |>
  ggplot(aes(x = flipper_length_mm)) +
  geom_histogram(aes(y = after_stat(density), fill = species),
                 position = "identity", alpha = 0.3)
```



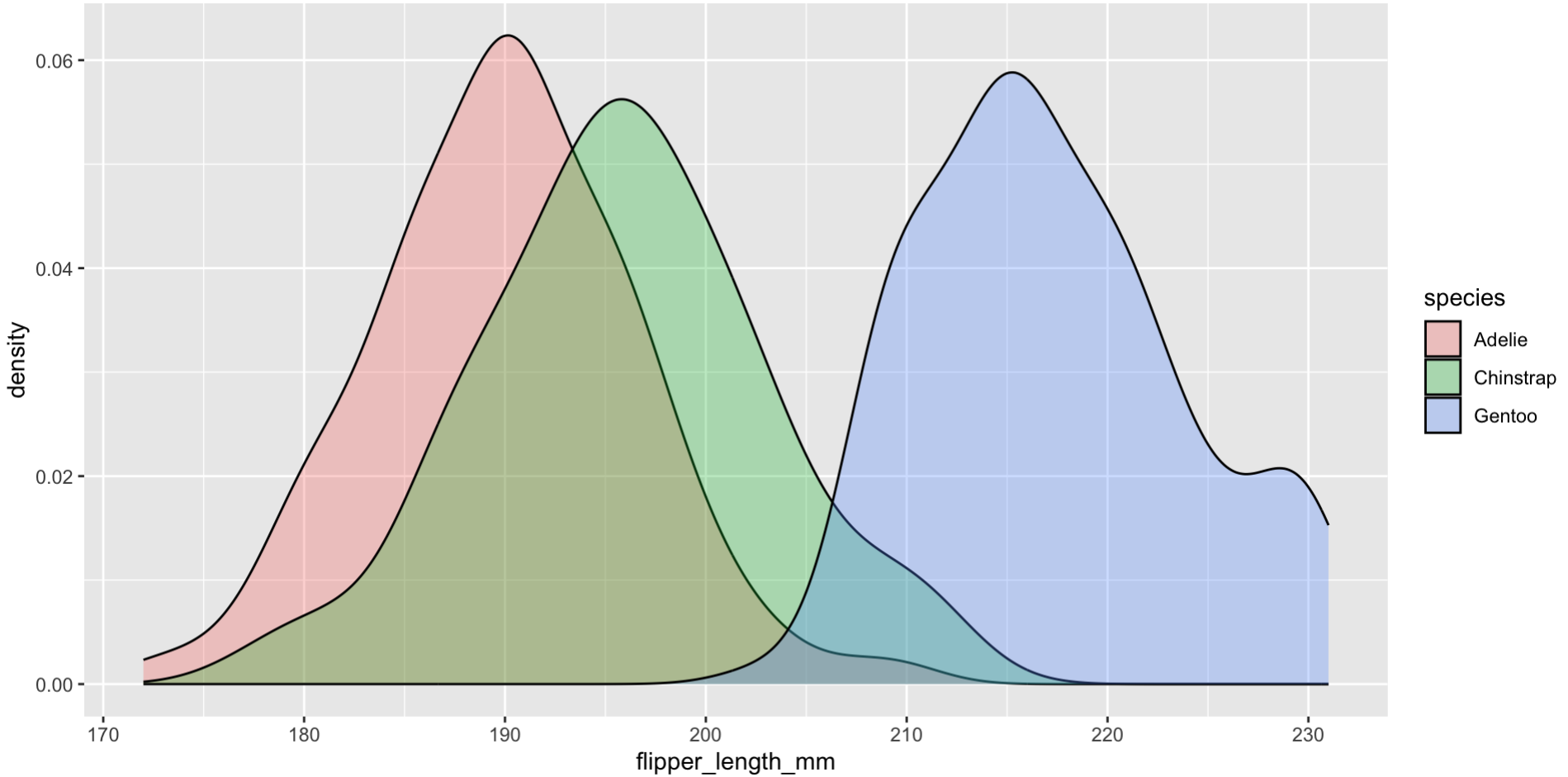
Can use density curves instead

```
1 penguins |>
2   ggplot(aes(x = flipper_length_mm)) +
3   geom_density(aes(color = species))
```



We should NOT fill the density curves

```
1 penguins |>
2   ggplot(aes(x = flipper_length_mm)) +
3   geom_density(aes(fill = species), alpha = .3)
```



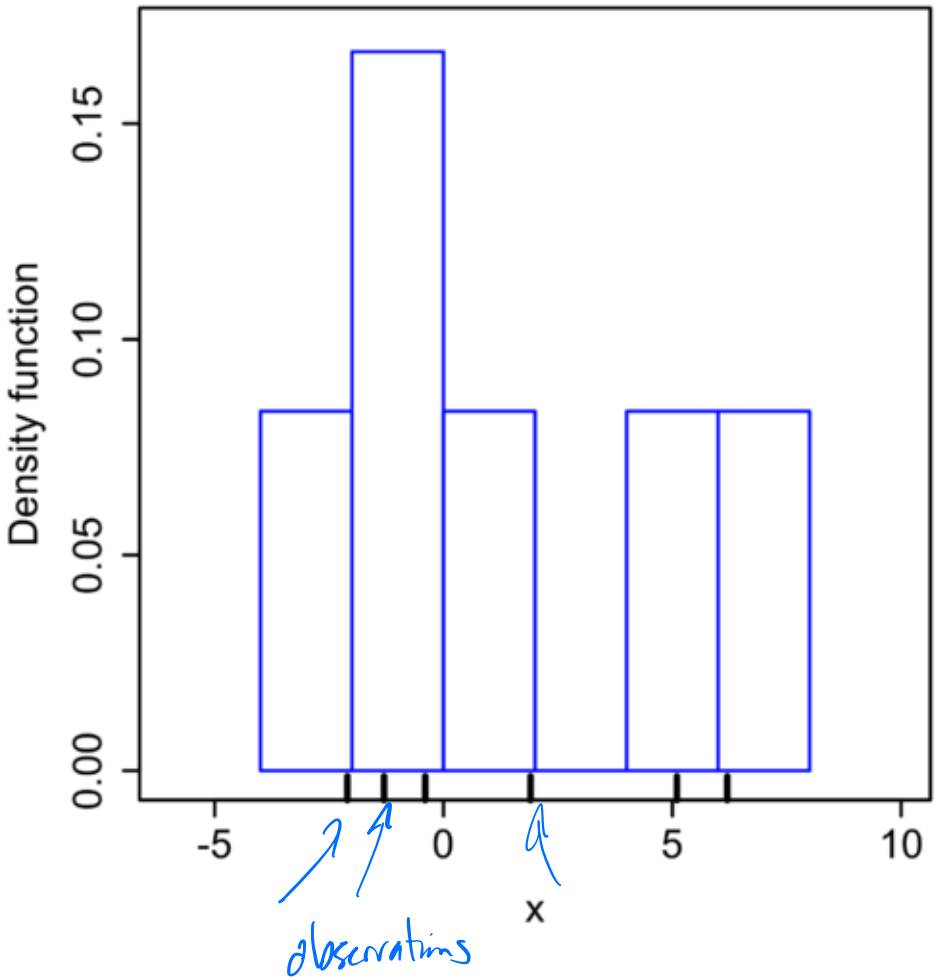
Kernel density estimation

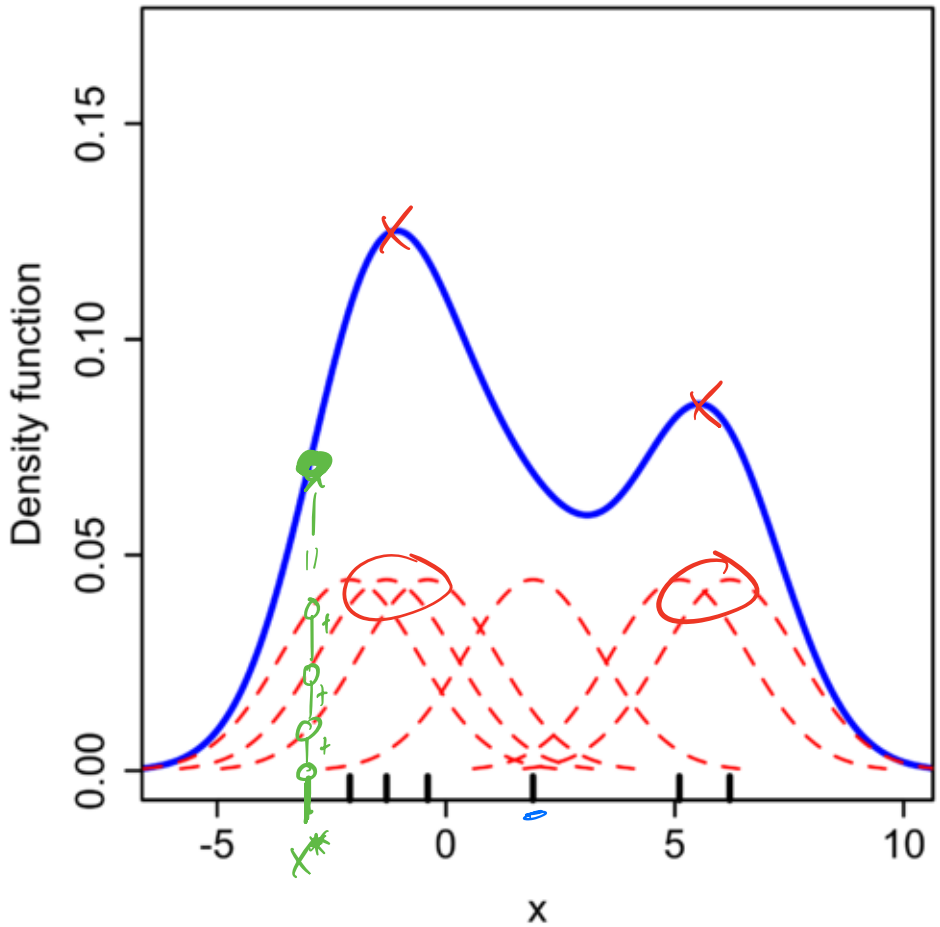
Goal: estimate PDF $f(x)$ for all possible values (assuming it is continuous & smooth)

$$\text{Kernel density estimate: } \hat{f}(x) = \frac{1}{n}\sum_{i=1}^{n}\frac{1}{h}K_h(x - x_i)$$

- $n =$ sample size, $x =$ new point to estimate $f(x)$ (does NOT have to be in dataset!)
- $h =$ **bandwidth**, analogous to histogram bin width, ensures $\hat{f}(x)$ integrates to 1
- $x_i = i$th observation in dataset
- $K_h(x - x_i)$ is the **Kernel** function, creates **weight** given distance of ith observation from new point
  - as $|x - x_i| \to \infty$ then $K_h(x - x_i) \to 0$, i.e. further apart ith row is from x, smaller the weight
  - as **bandwidth** $h \uparrow$ weights are more evenly spread out (as $h \downarrow$ more concentrated around x)
  - typically use **Gaussian** / Normal kernel: $\propto e^{-(x-x_i)^2/2h^2}$
  - $K_h(x - x_i)$ is large when x_i is close to x

Wikipedia example





We display kernel density estimates with `geom_density()`

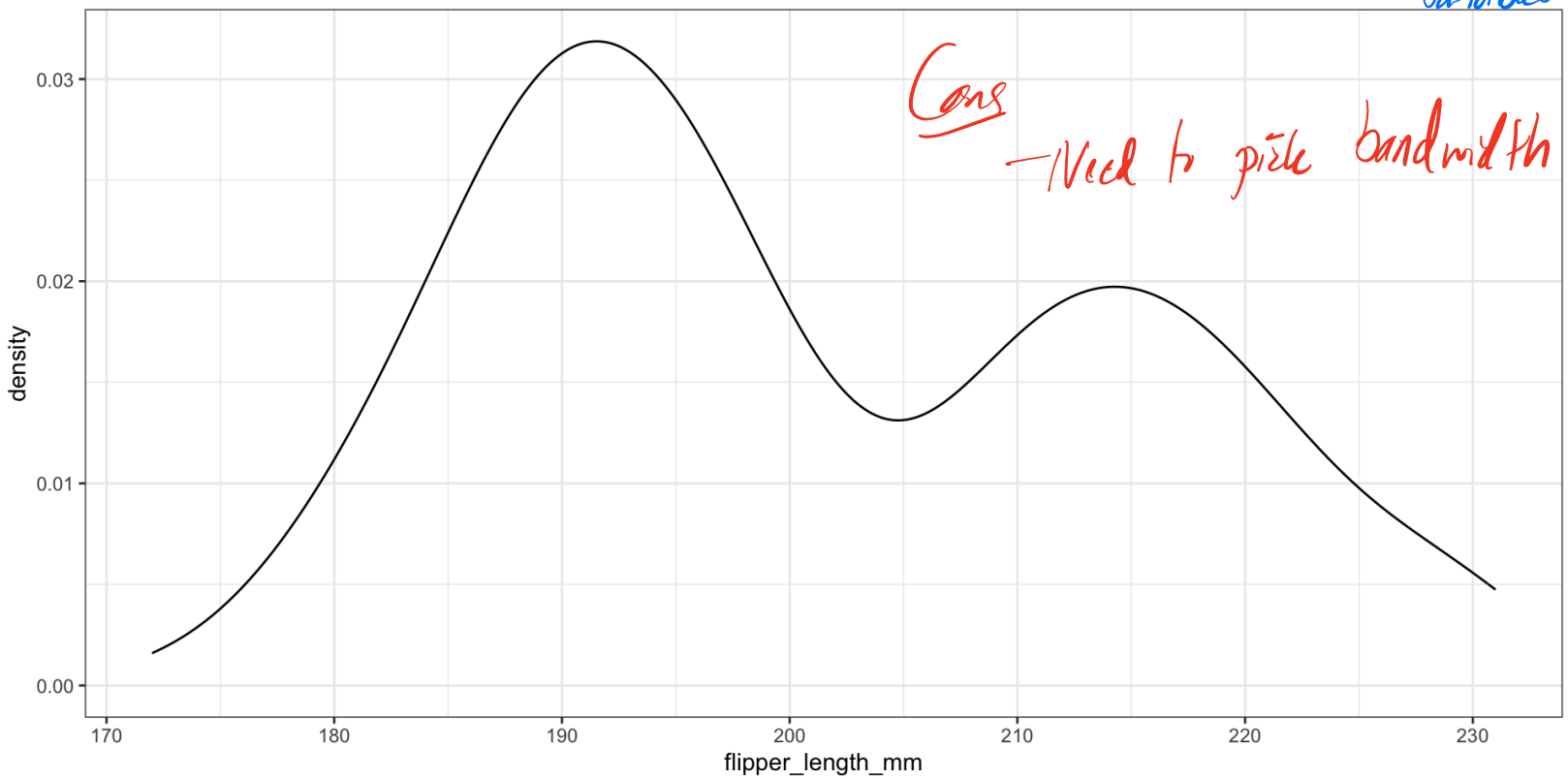
```
penguins |>
  ggplot(aes(x = flipper_length_mm)) +
  geom_density() +
  theme_bw()
```

Pros

+ Display full shape of distribution
+ Easily layer, add categorical variables w/ color

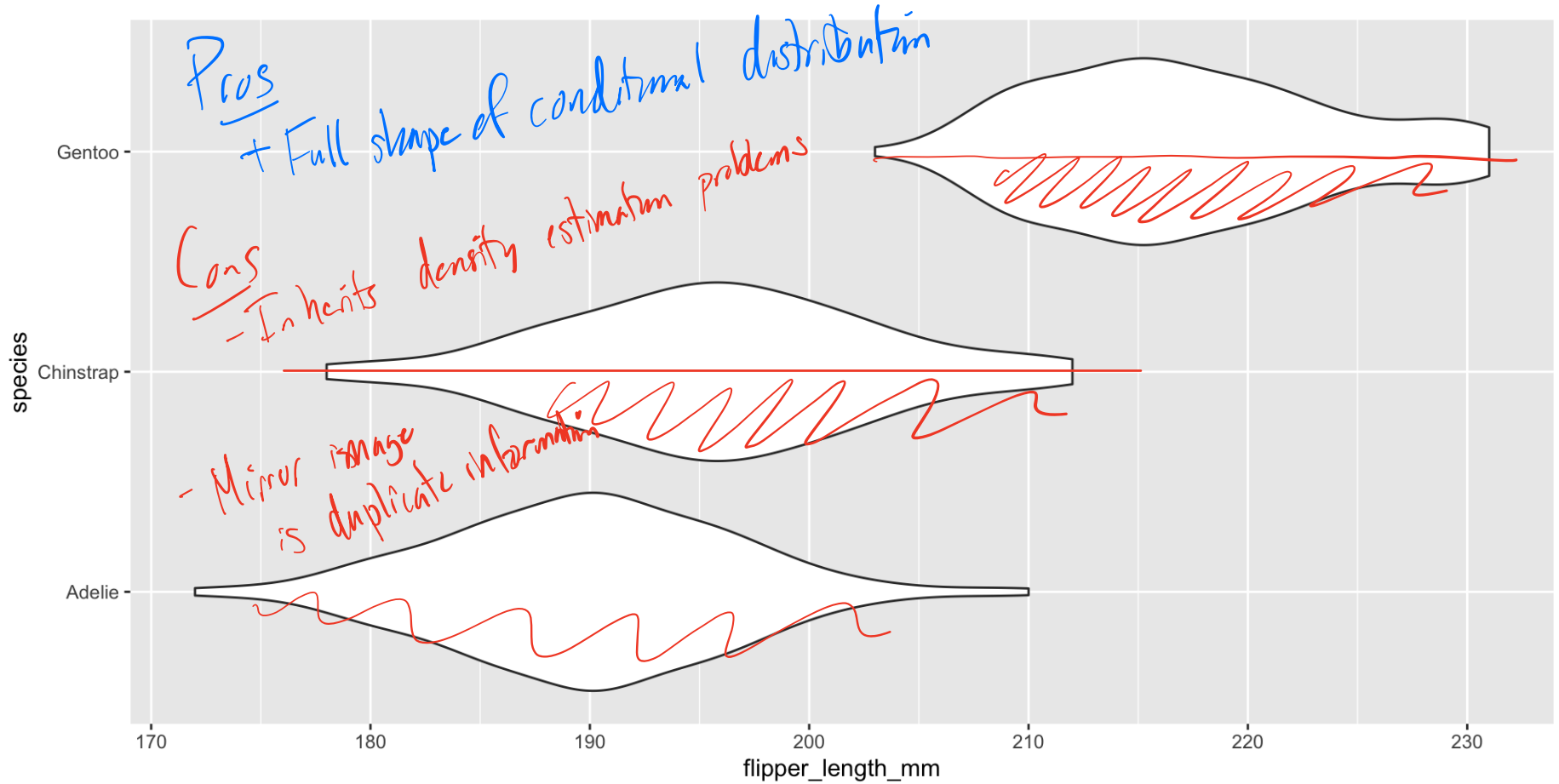
Cons

- Need to pick bandwidth & kernel



Visualizing conditional distributions: violin plots

```
penguins |>
  ggplot(aes(x = species, y = flipper_length_mm)) +
  geom_violin() +
  coord_flip()
```



Visualizing conditional distributions: ggbeeswarm

```
1 library(ggbeeswarm)
2 penguins |>
3   ggplot(aes(x = flipper_length_mm, y = species)) +
4   geom_beeswarm(cex = 1.5) +
5   theme_bw()
```