

# Intro to Visualizations & String Data

---

Daniel Anderson

Week 2

# Agenda

---

- Visualizing distributions
  - histograms
  - density plots
  - Empirical cumulative density plots
  - QQ plots
- Visualizing amounts
  - bar plots
  - dot plots
  - heatmaps
- Working textual data
  - Splitting into words, n–grams
  - Visualizing word frequencies
- Cleaning string data
  - Substrings, basic transformations
  - Substitutions and quick pattern matching

A promise: You will have **at least 30 minutes** for lab, even if we have to pause this lecture early

# Learning Objectives

---

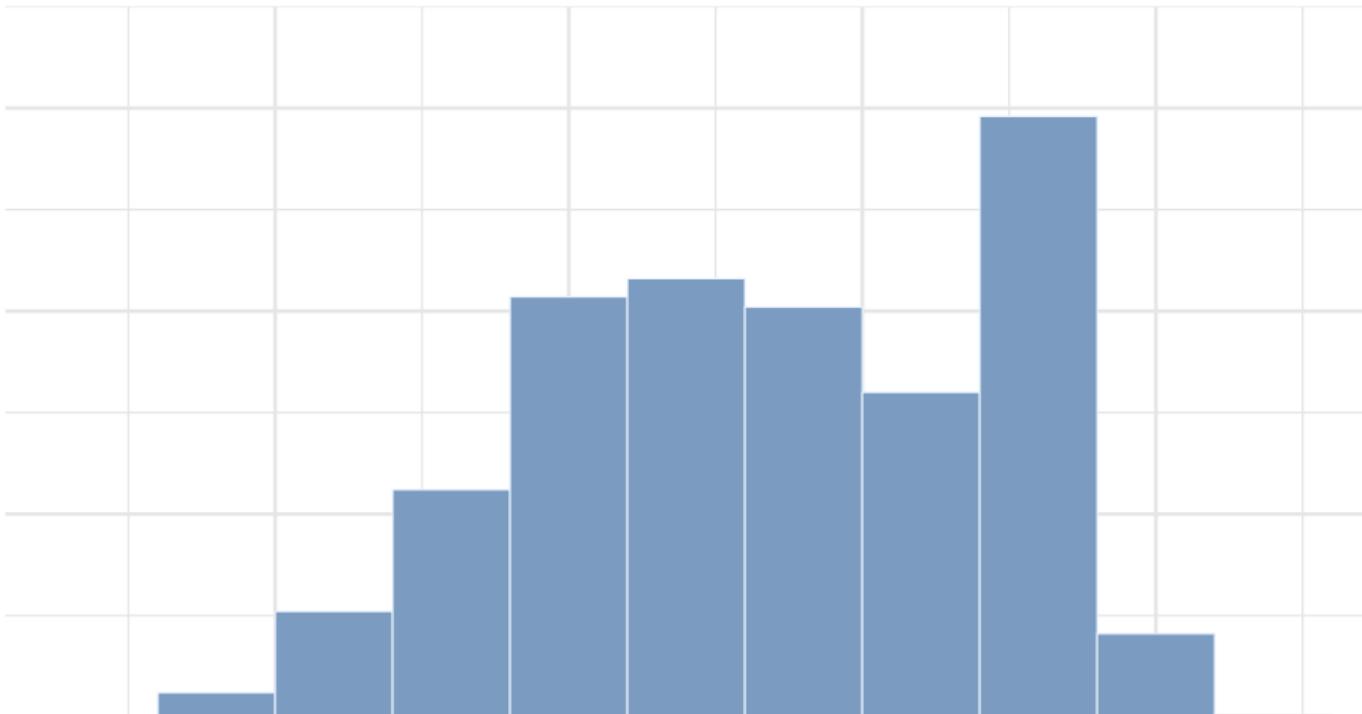
- Understand various ways the same underlying data can be displayed
- Think through pros/cons of each
- Understand the basic structure of the code to produce the various plots
- Create structured data from text and be able to visualize word frequencies
- Be able to replace patterns in strings and understand which character need to be "escaped"

One  
continuous  
variable

---

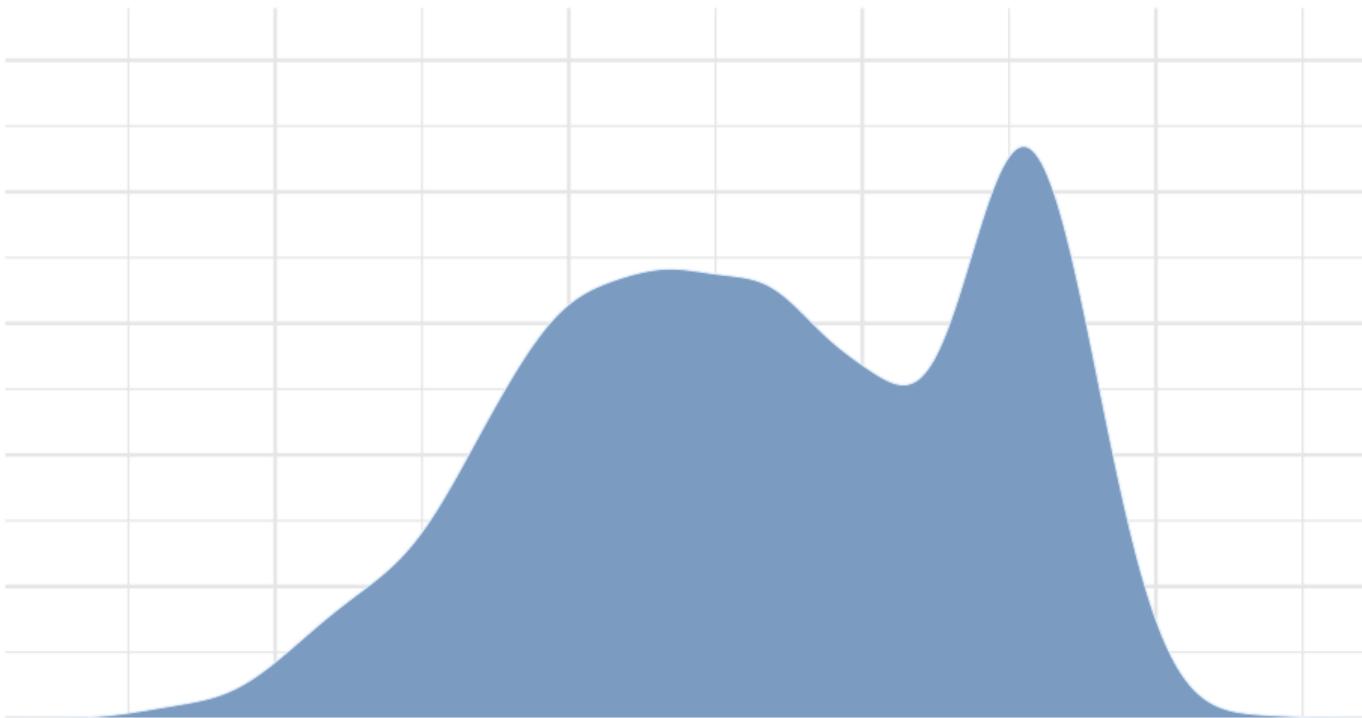
# Histogram

---



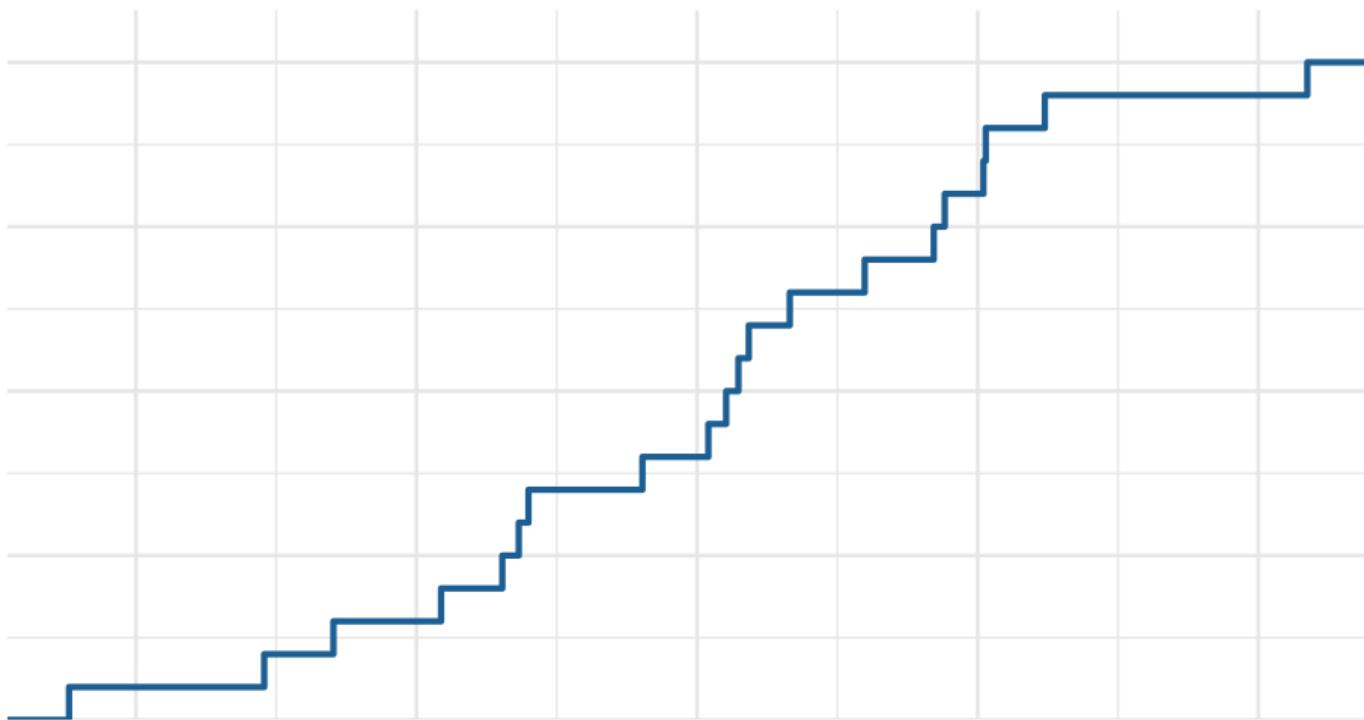
# Density plot

---



# (Empirical) Cumulative Density

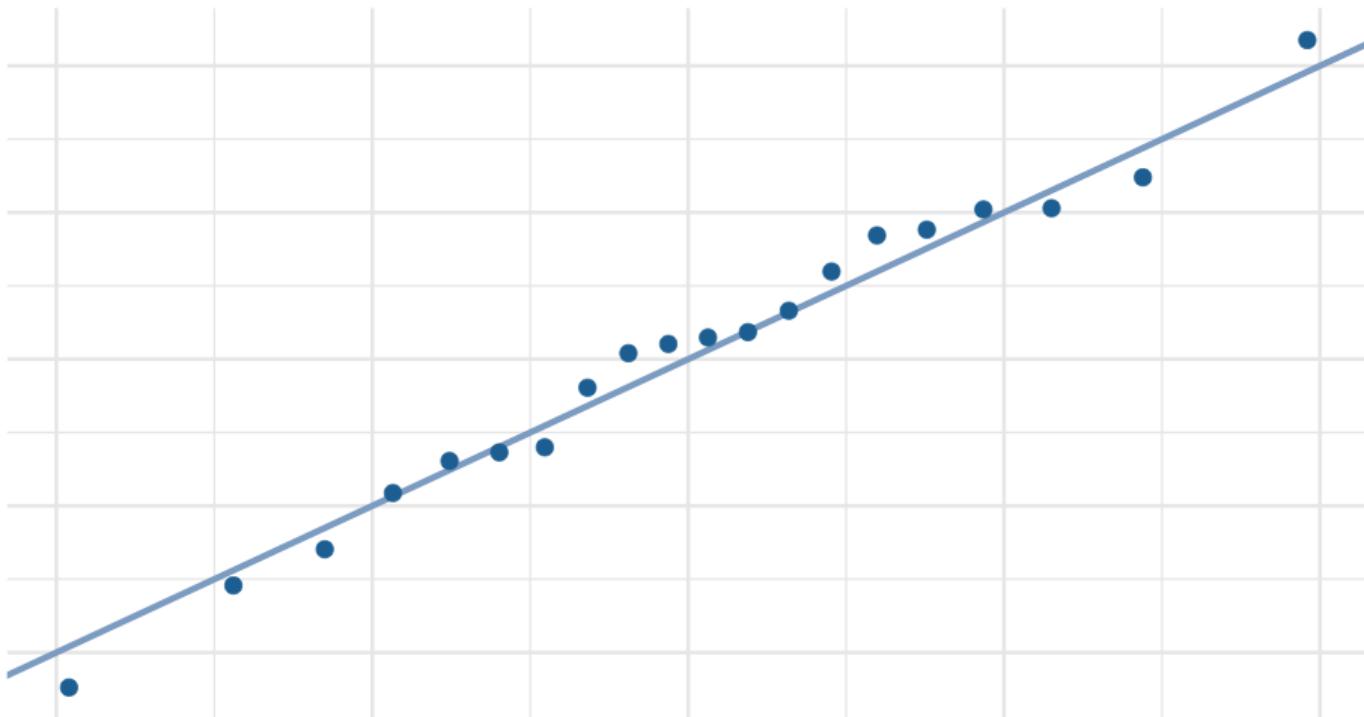
---



# QQ Plot

---

Compare to theoretical quantiles (for normality)



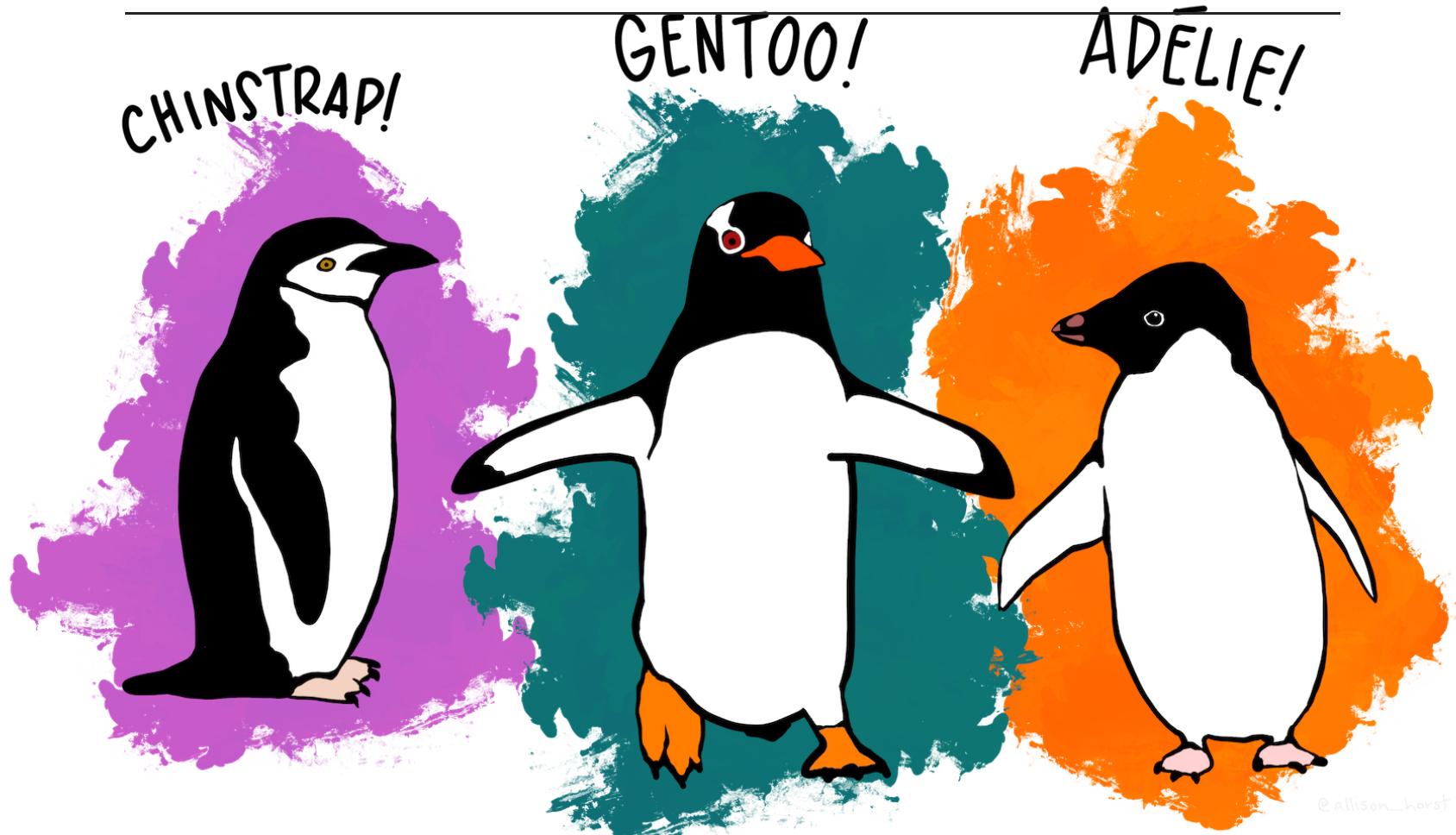
# Empirical examples

---

I'll move fast, but if you want to (try to) follow along, feel free.

# Penguin data

---



```
library(palmerpenguins)
penguins
```

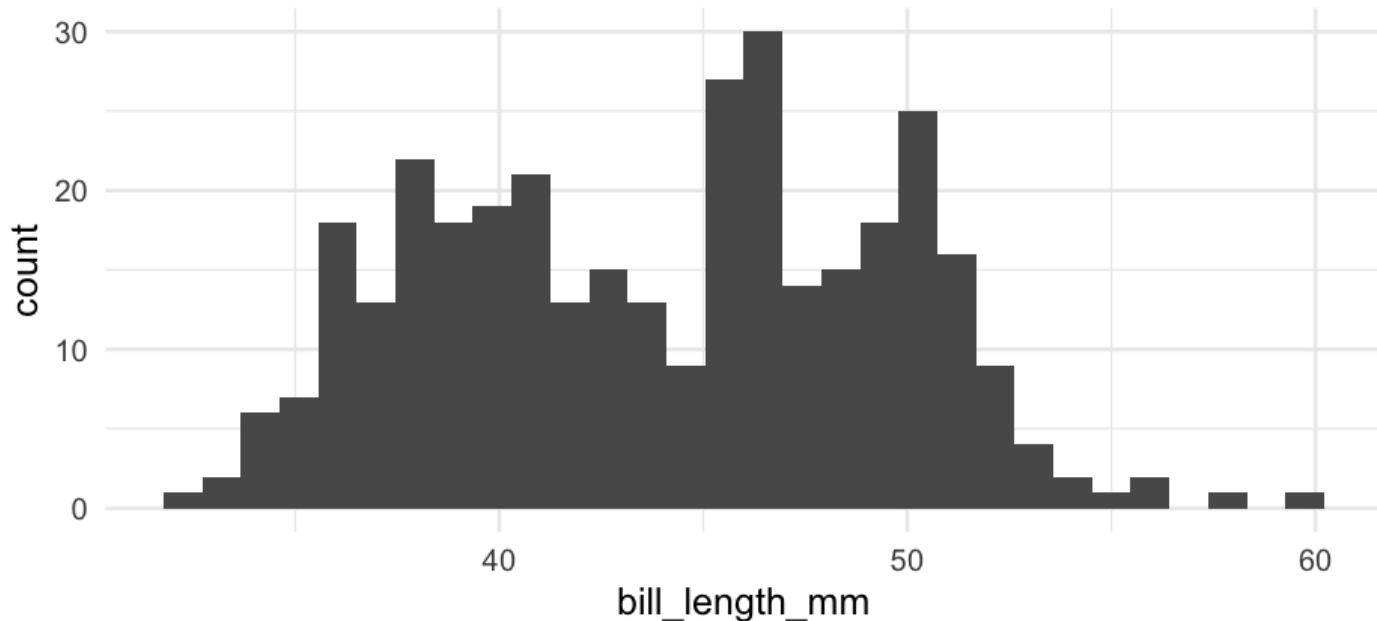
```
## # A tibble: 344 × 8
##   species     island   bill_length_mm   bill_depth_mm flipper_length_mm
##   <fct>      <fct>           <dbl>            <dbl>                <int>
## 1 Adelie    Torgersen       39.1             18.7                 181
## 2 Adelie    Torgersen       39.5             17.4                 186
## 3 Adelie    Torgersen       40.3              18                  195
## 4 Adelie    Torgersen        NA                NA                  NA
## 5 Adelie    Torgersen       36.7             19.3                 193
## 6 Adelie    Torgersen       39.3             20.6                 190
## 7 Adelie    Torgersen       38.9             17.8                 181
## 8 Adelie    Torgersen       39.2             19.6                 195
## 9 Adelie    Torgersen       34.1             18.1                 193
## 10 Adelie   Torgersen        42                20.2                190
## # ... with 334 more rows, and 3 more variables: body_mass_g <int>,
## #       sex <fct>, year <int>
```

# Basic histogram

---

```
ggplot(penguins, aes(x = bill_length_mm)) +  
  geom_histogram()
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

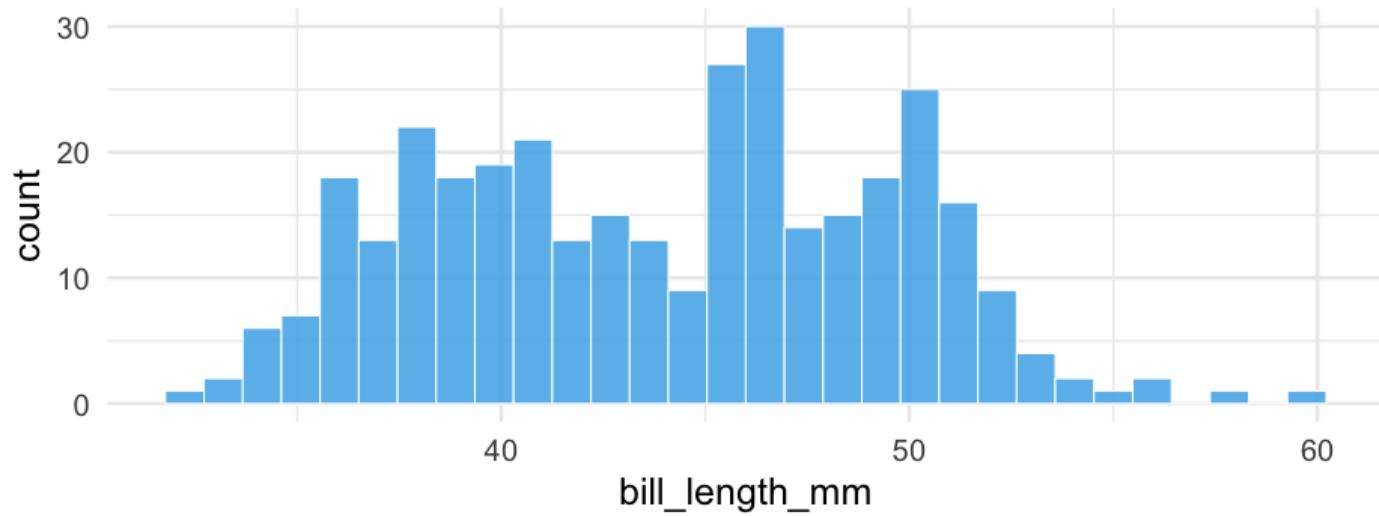


# Make it a little prettier

---

```
ggplot(penguins, aes(x = bill_length_mm)) +  
  geom_histogram(fill = "#56B4E9",  
                 color = "white",  
                 alpha = 0.9)
```

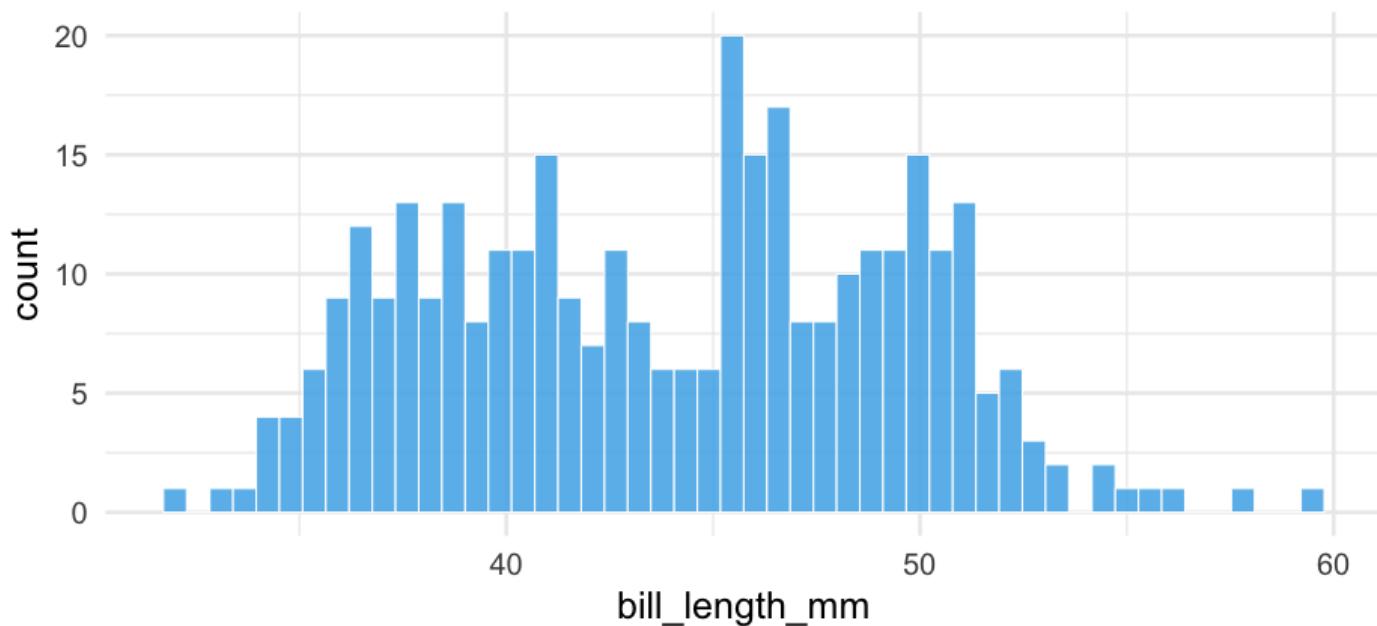
## `stat\_bin()` using `bins = 30`. Pick better value with `binwidth`.



# Change the number of bins

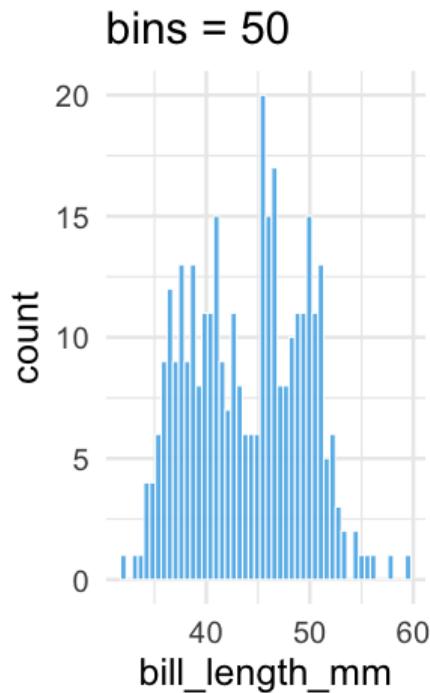
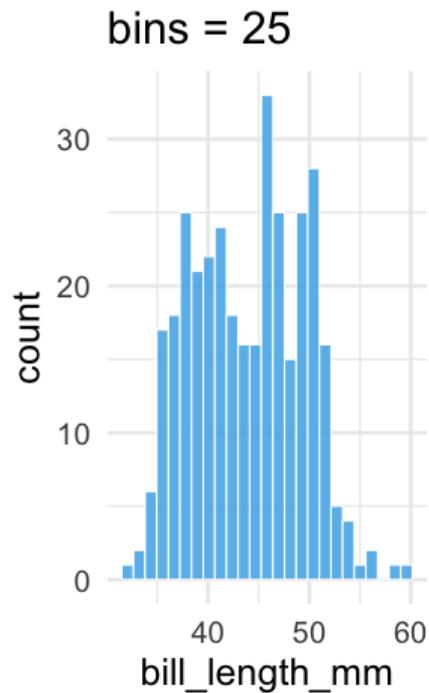
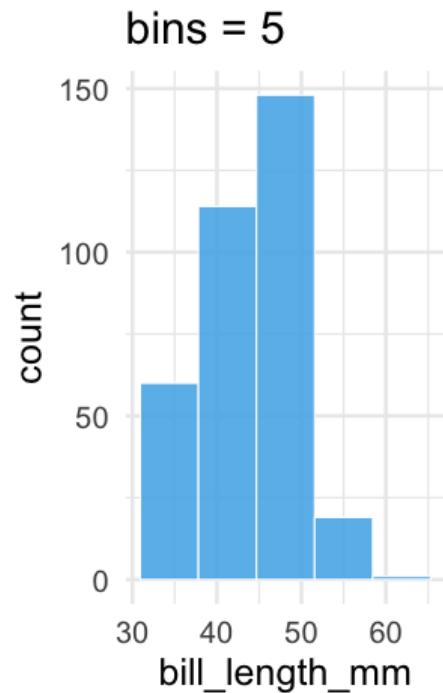
---

```
ggplot(penguins, aes(x = bill_length_mm)) +  
  geom_histogram(fill = "#56B4E9",  
                 color = "white",  
                 alpha = 0.9,  
                 bins = 50)
```



# Vary the number of bins

---

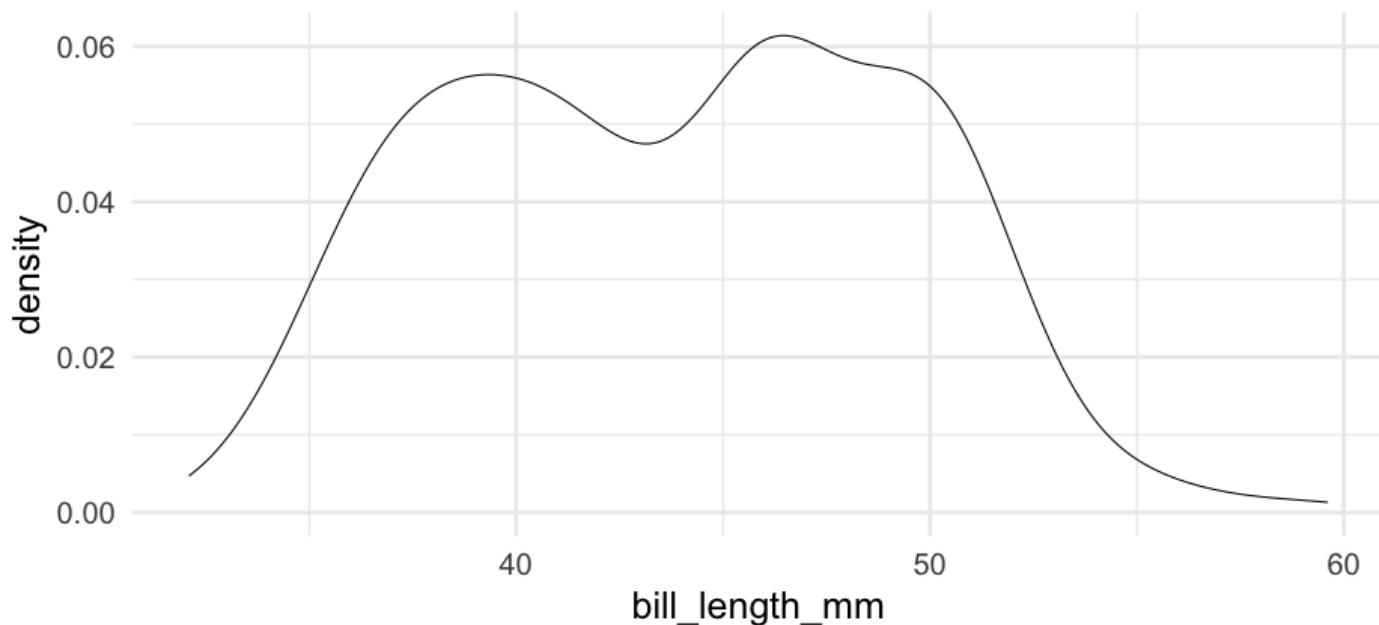


# Denisty plot

---

ugly 😞

```
ggplot(penguins, aes(bill_length_mm)) +  
  geom_density()
```

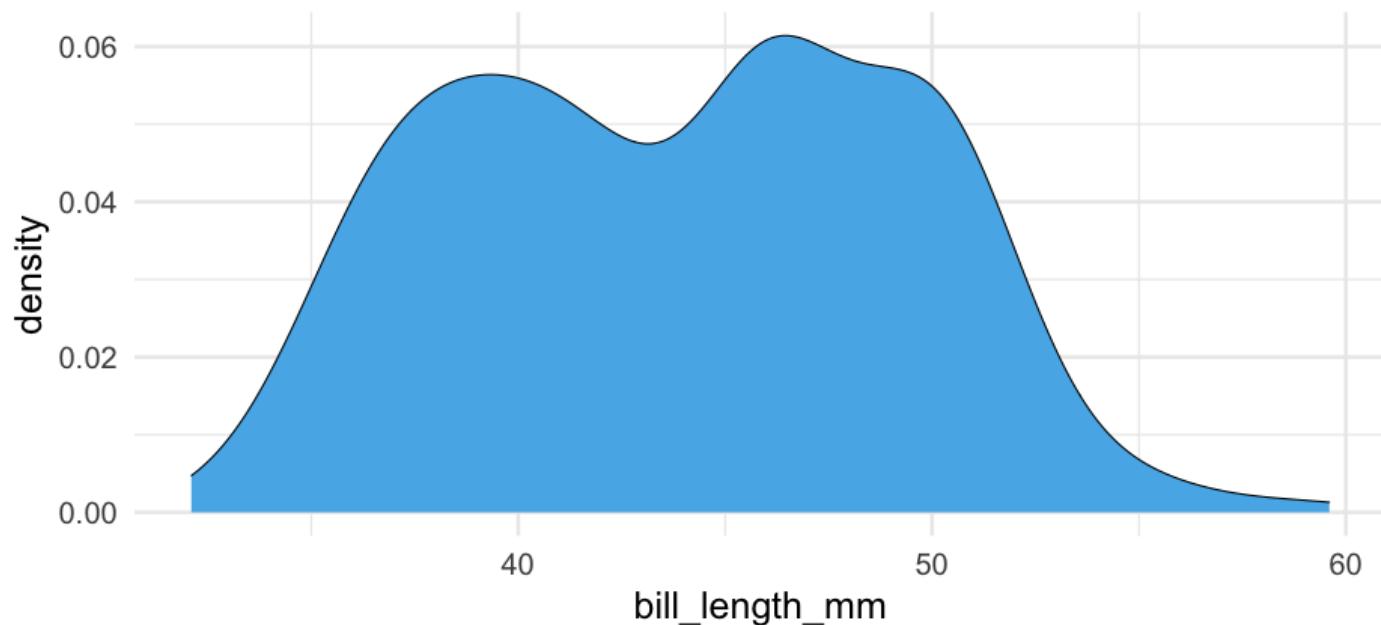


# Denisty plot

---

Change the fill 😊

```
ggplot(penguins, aes(bill_length_mm)) +  
  geom_density(fill = "#56B4E9")
```



# Density plot estimation

---

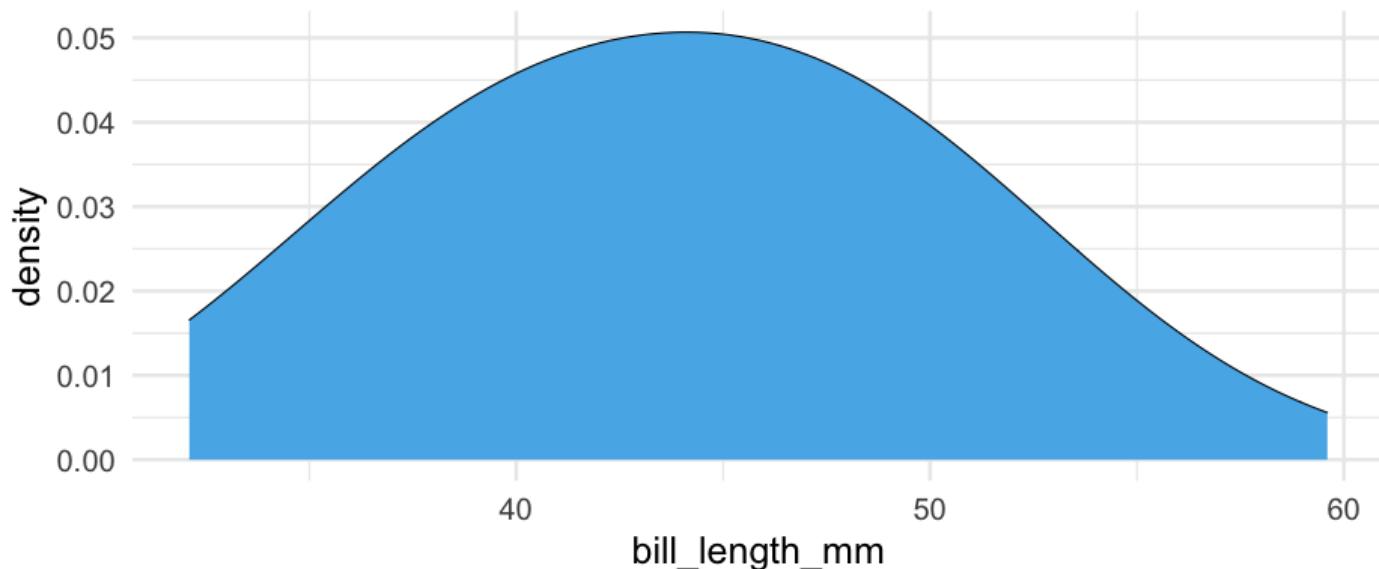
- Kernel density estimation
  - Different kernel shapes can be selected
  - Bandwidth matters most
  - Smaller bands = bend more to the data
- Approximation of the underlying continuous probability function
  - Integrates to 1.0 (y-axis is somewhat difficult to interpret)

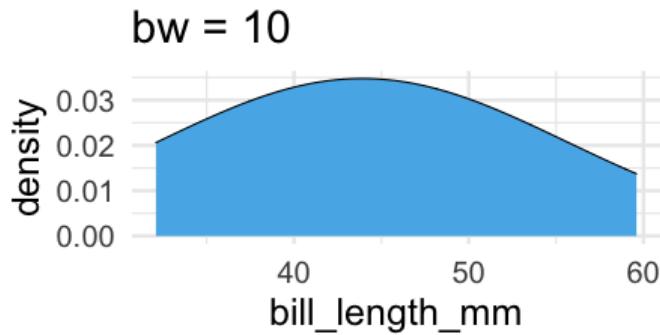
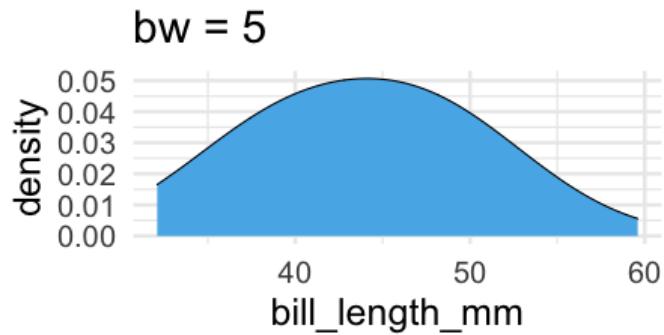
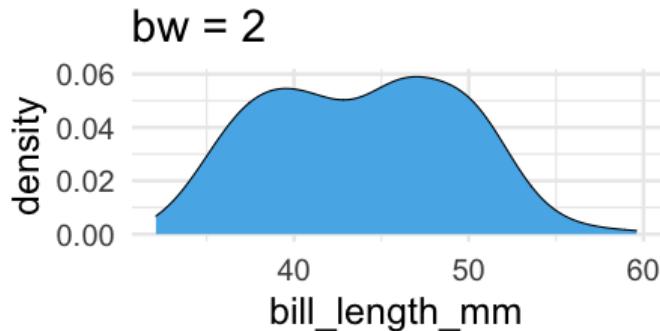
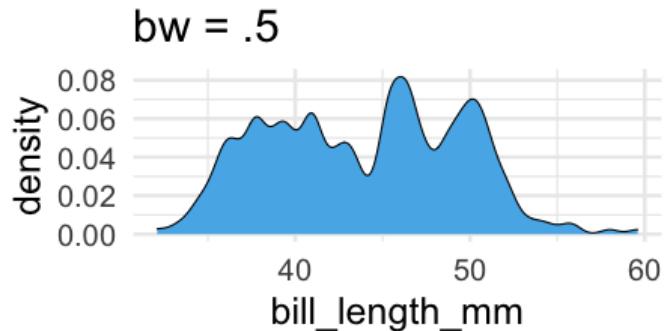
# Denisty plot

---

change the bandwidth

```
ggplot(penguins, aes(bill_length_mm)) +  
  geom_density(fill = "#56B4E9",  
              bw = 5)
```



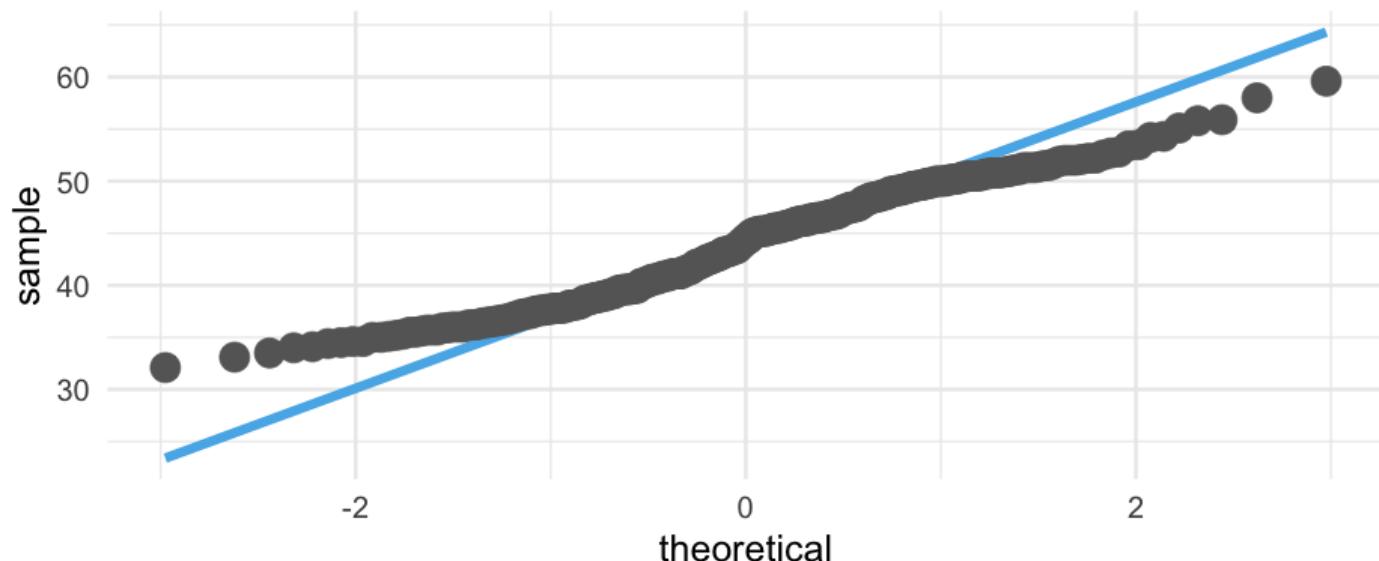


# Quickly

---

How well does it approximate a normal distribution?

```
ggplot(penguins, aes(sample = bill_length_mm)) +  
  stat_qq_line(color = "#56B4E9") +  
  geom_qq(color = "gray40")
```



# Grouped data

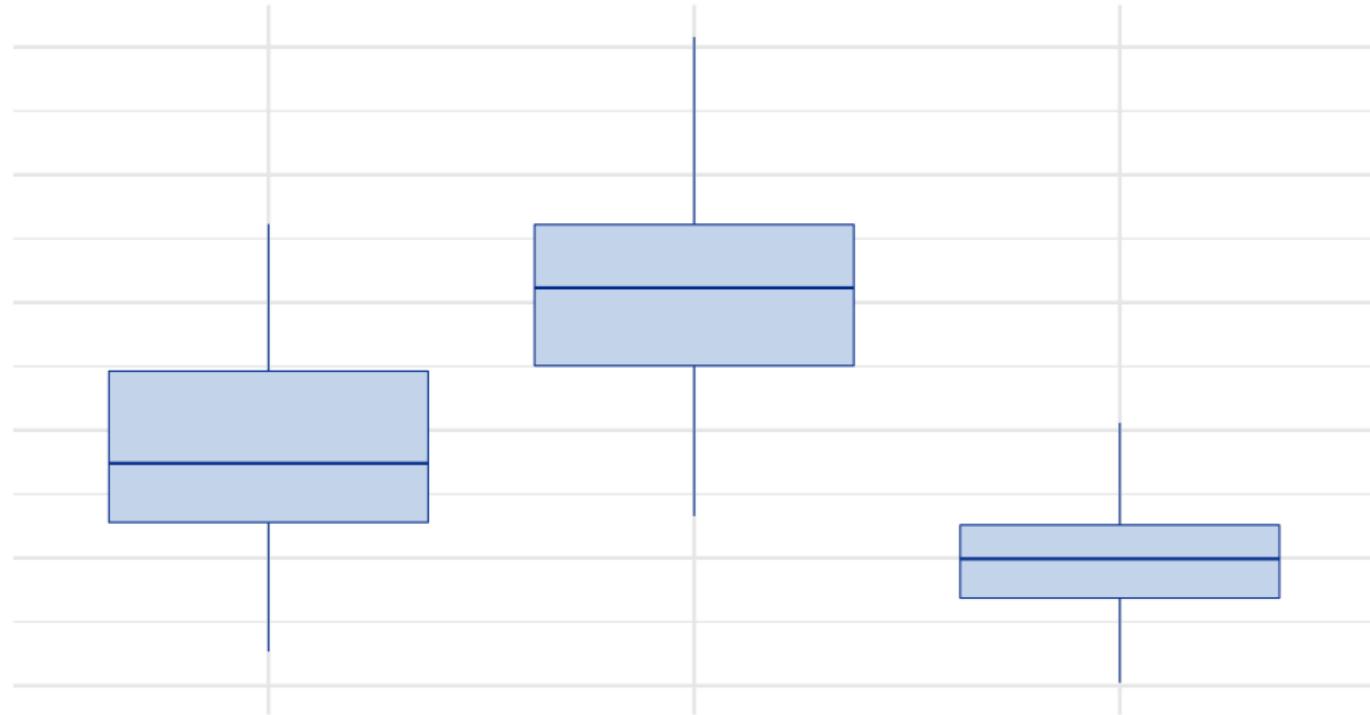
---

## Distributions

How do we display more than one distribution at a time?

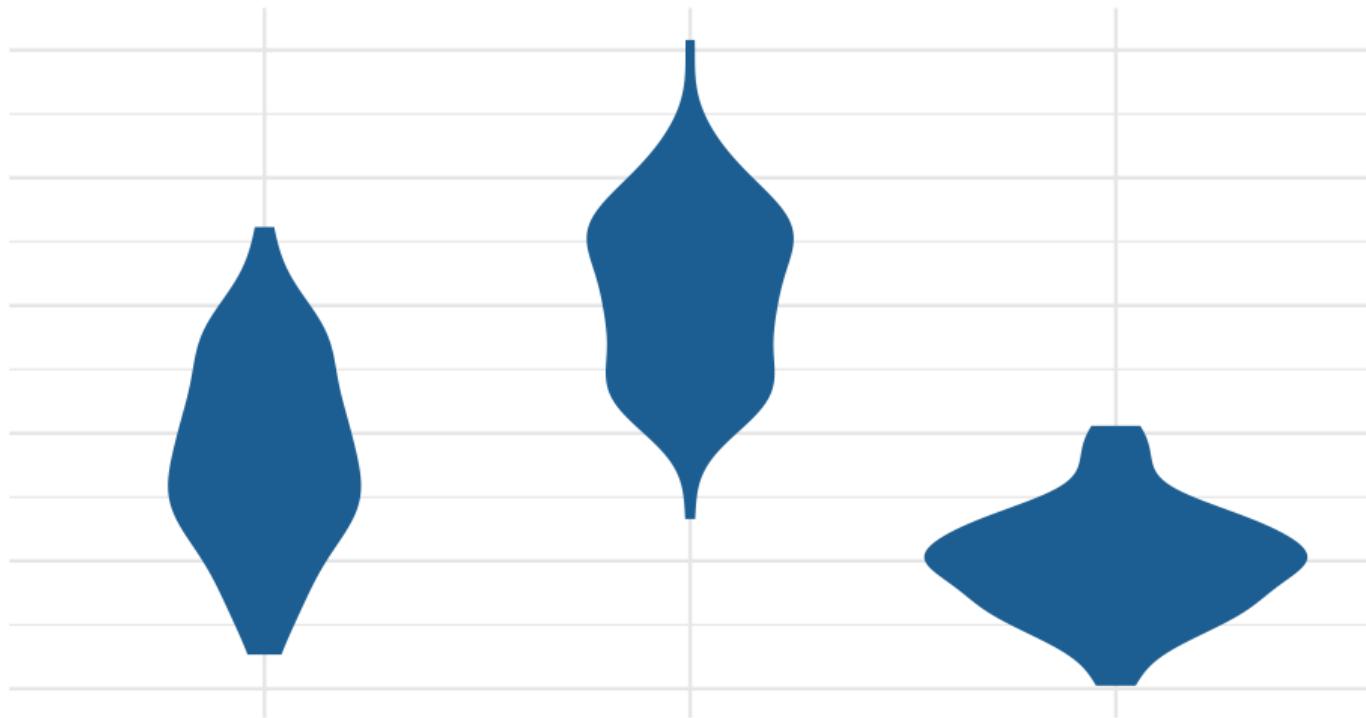
# Boxplots

---



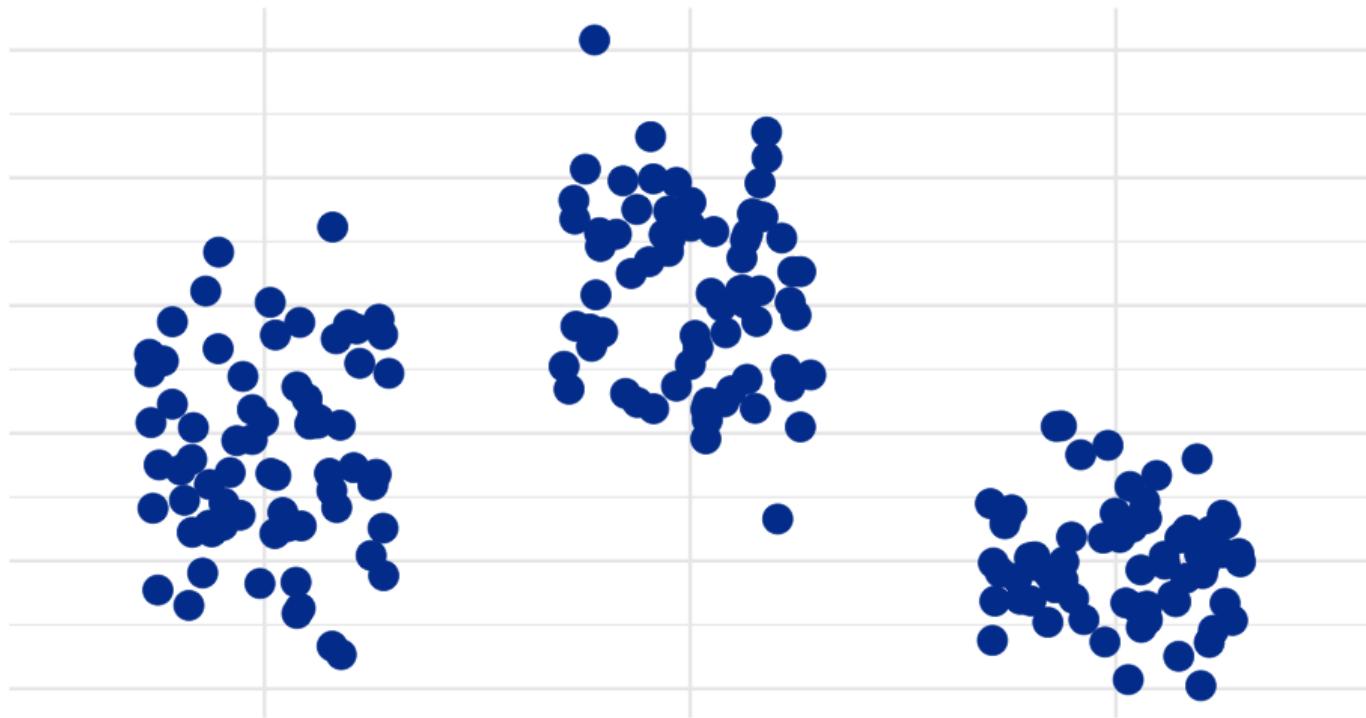
# Violin plots

---



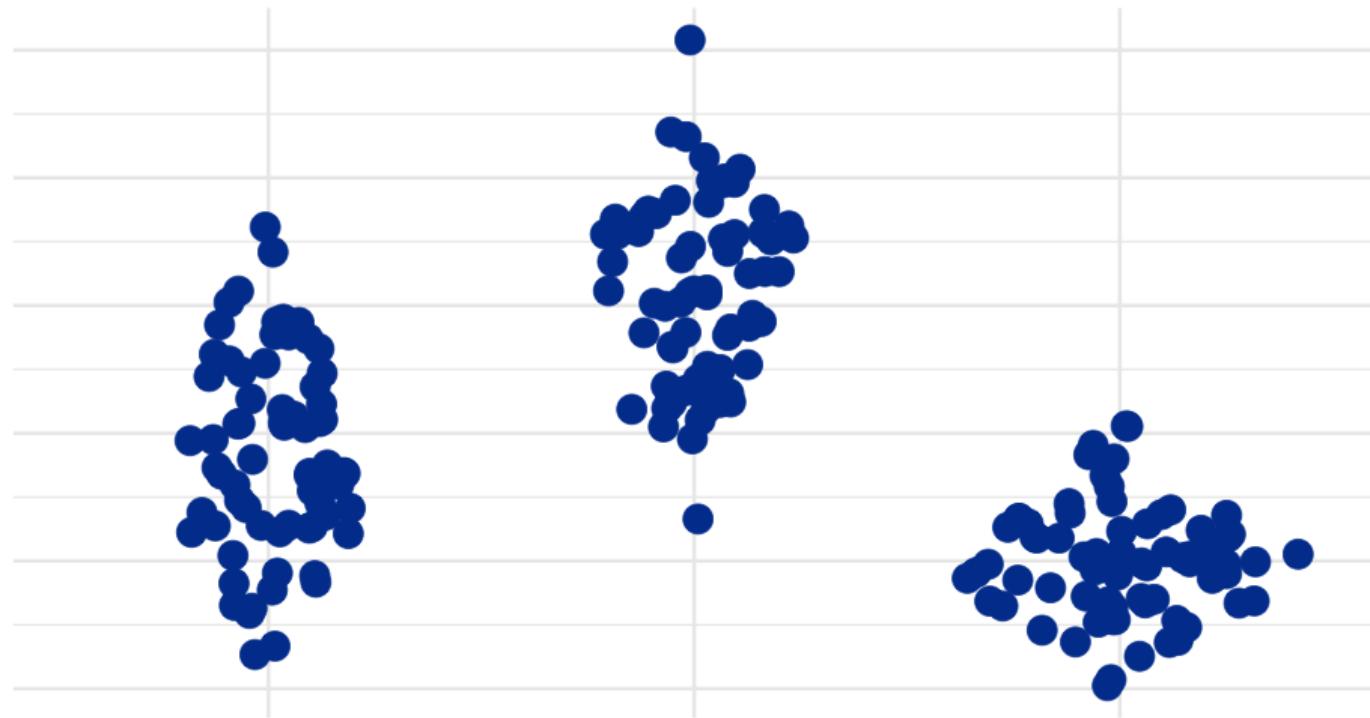
# Jittered points

---



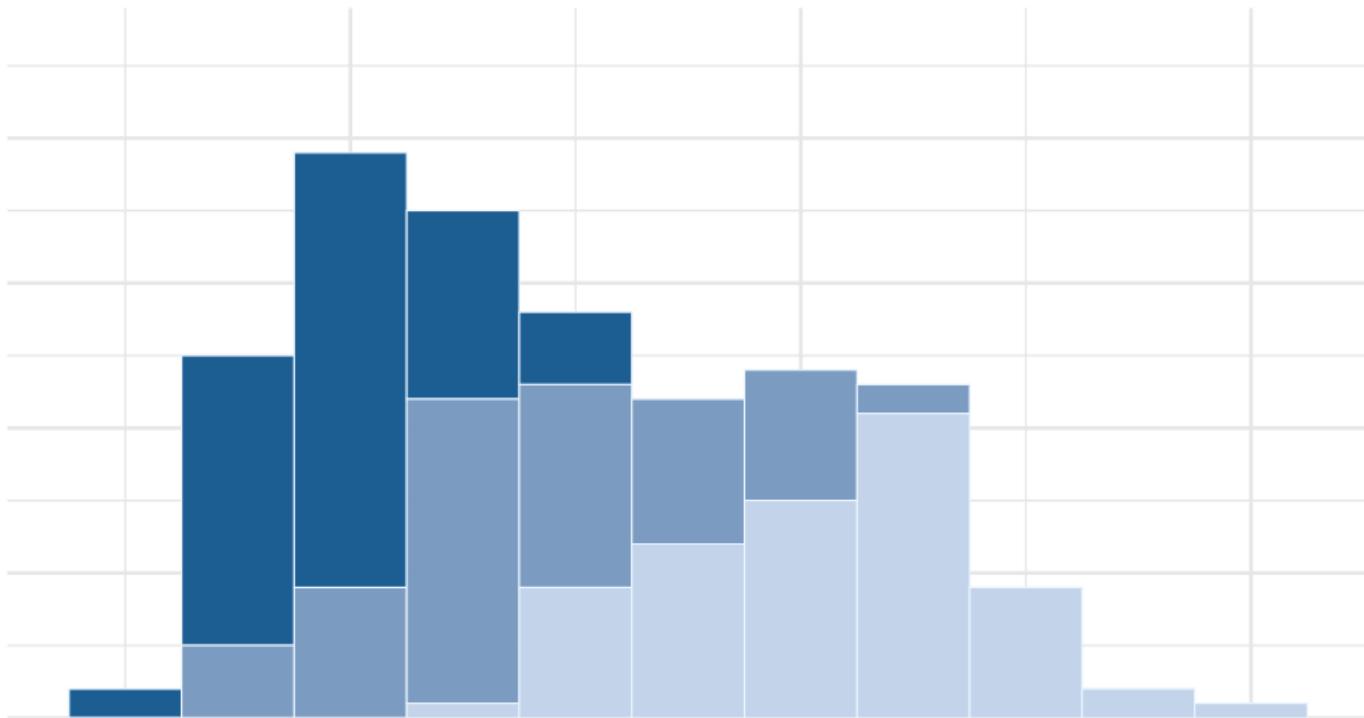
# Sina plots

---



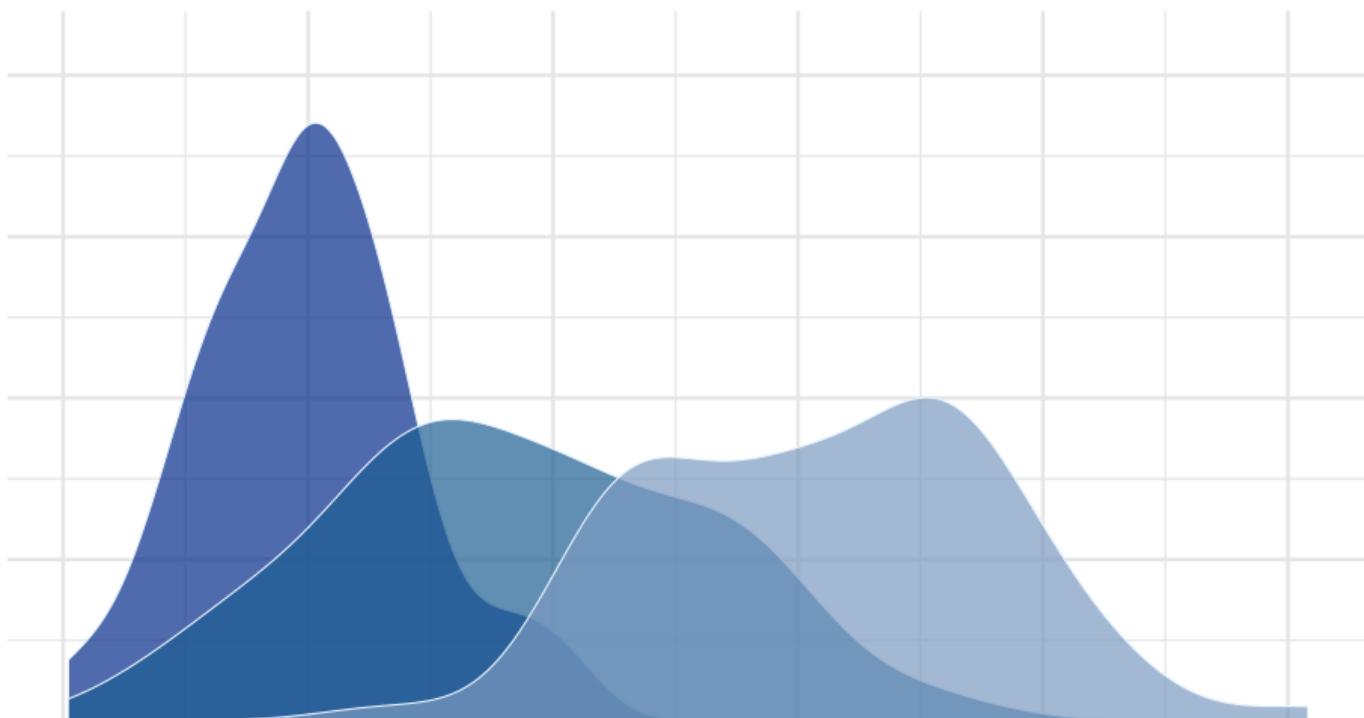
# Stacked histograms

---



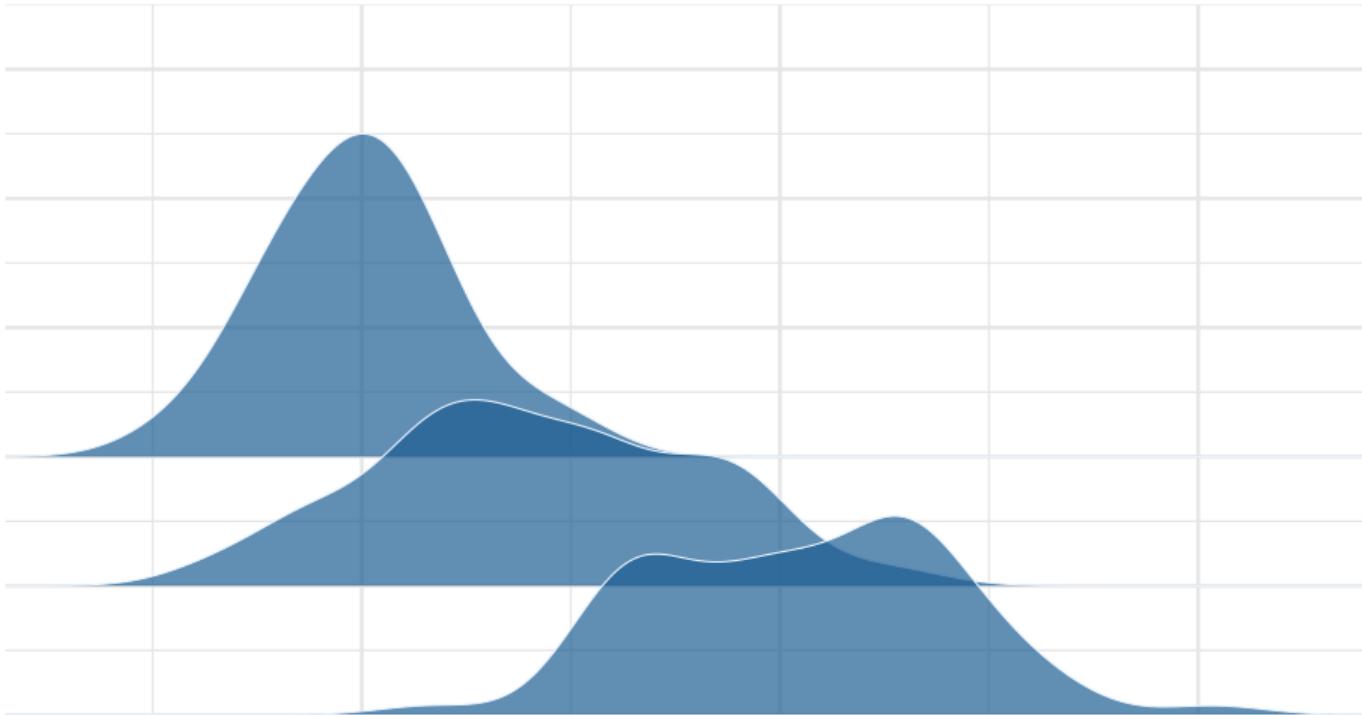
# Overlapping densities

---



# Ridgeline densities

---

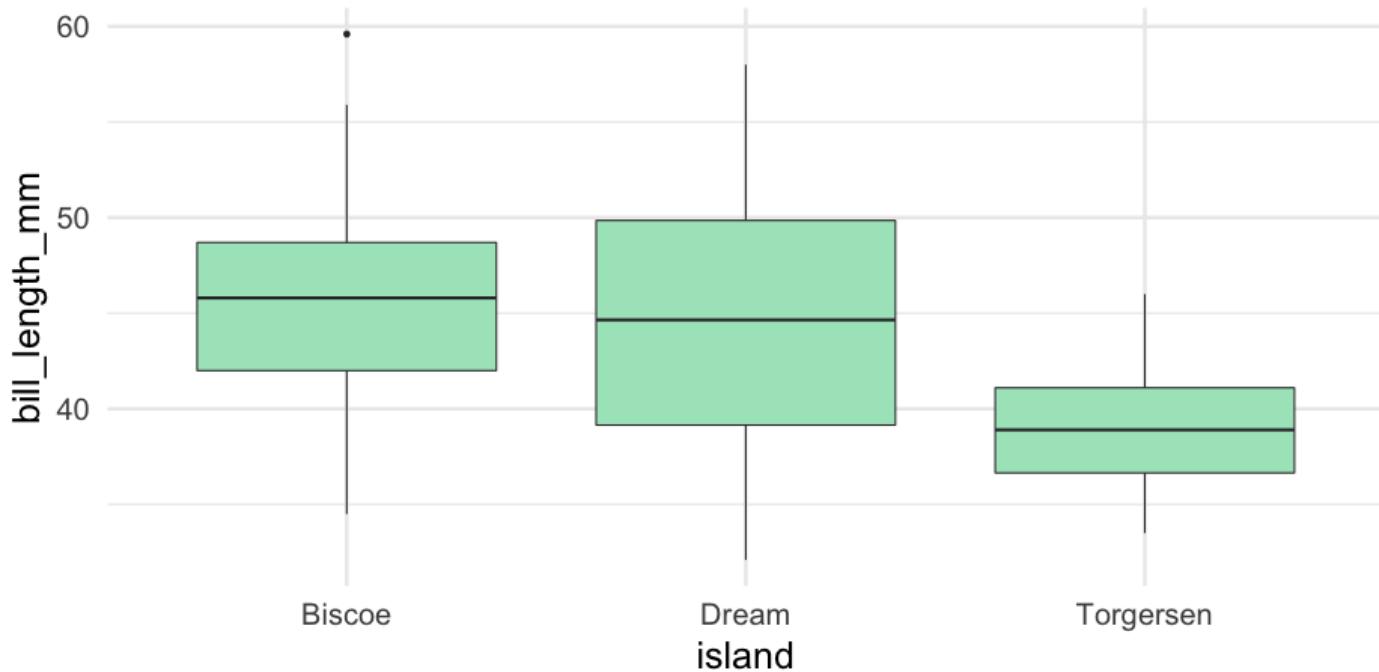




# Boxplots

---

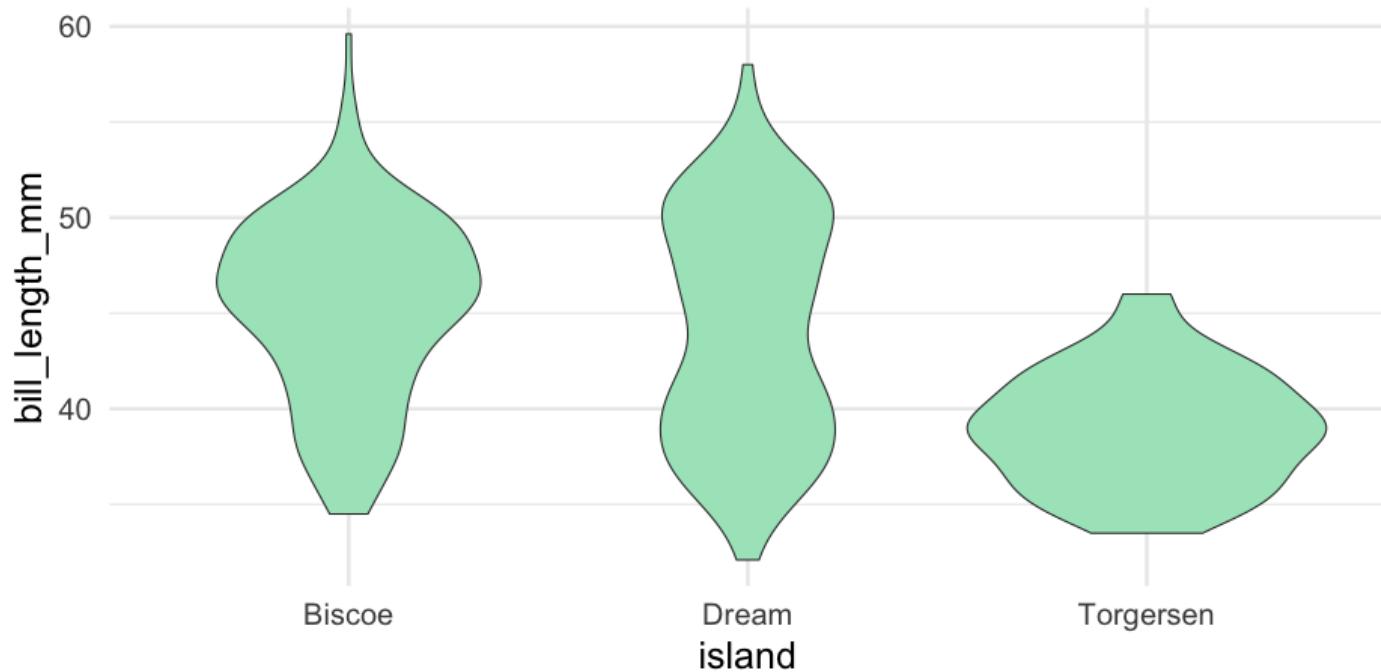
```
ggplot(penguins, aes(island, bill_length_mm)) +  
  geom_boxplot(fill = "#A9E5C5")
```



# Violin plots

---

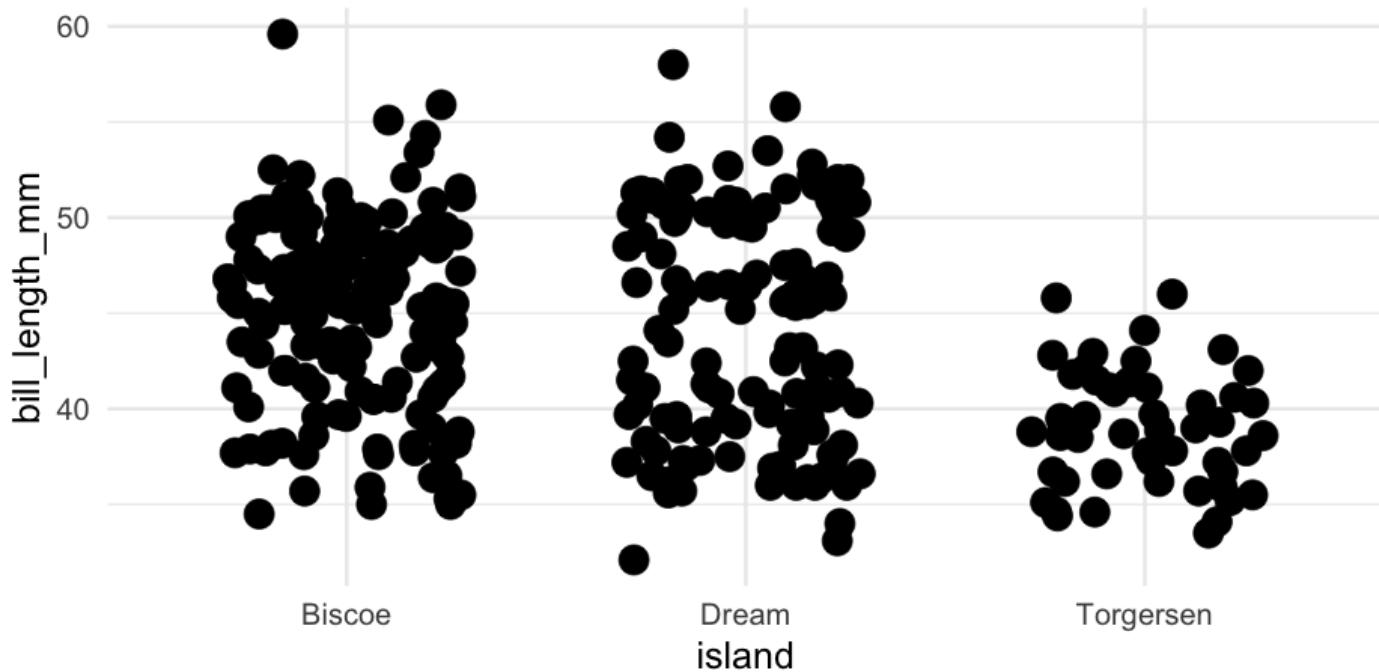
```
ggplot(penguins, aes(island, bill_length_mm)) +  
  geom_violin(fill = "#A9E5C5")
```



# Jittered point plots

---

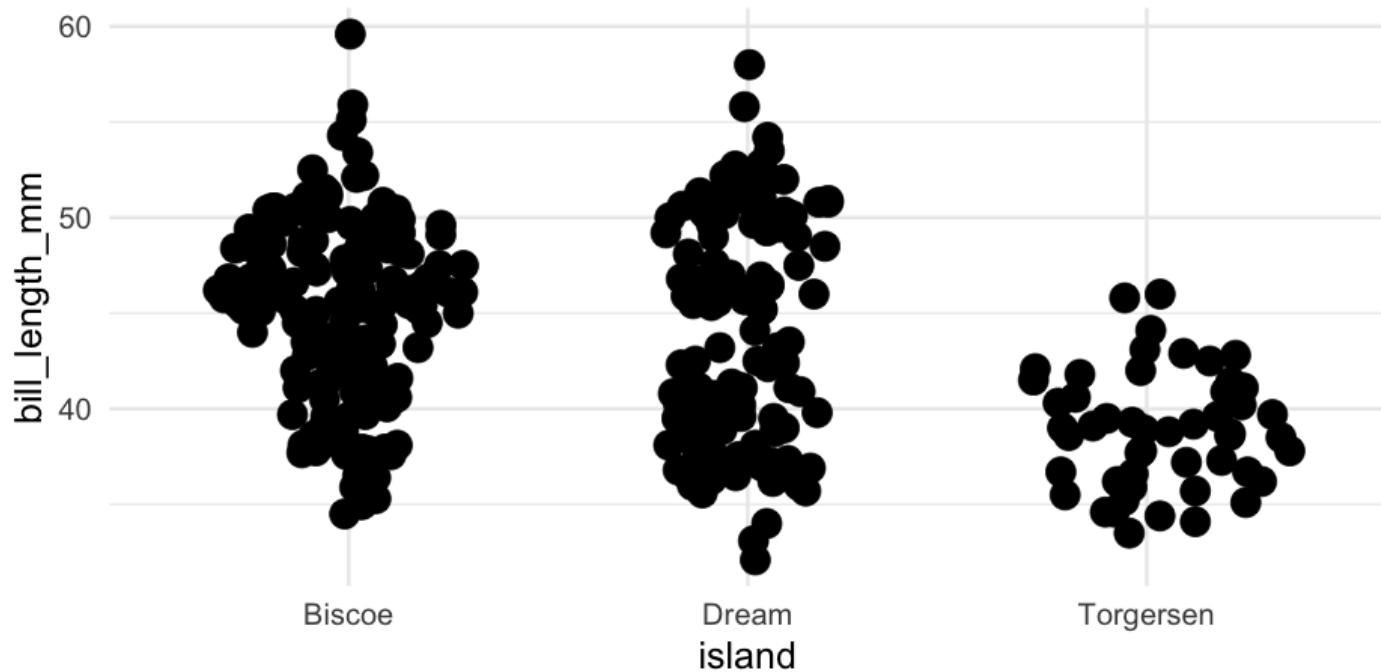
```
ggplot(penguins, aes(island, bill_length_mm)) +  
  geom_jitter(width = 0.3, height = 0)
```



# Sina plot

---

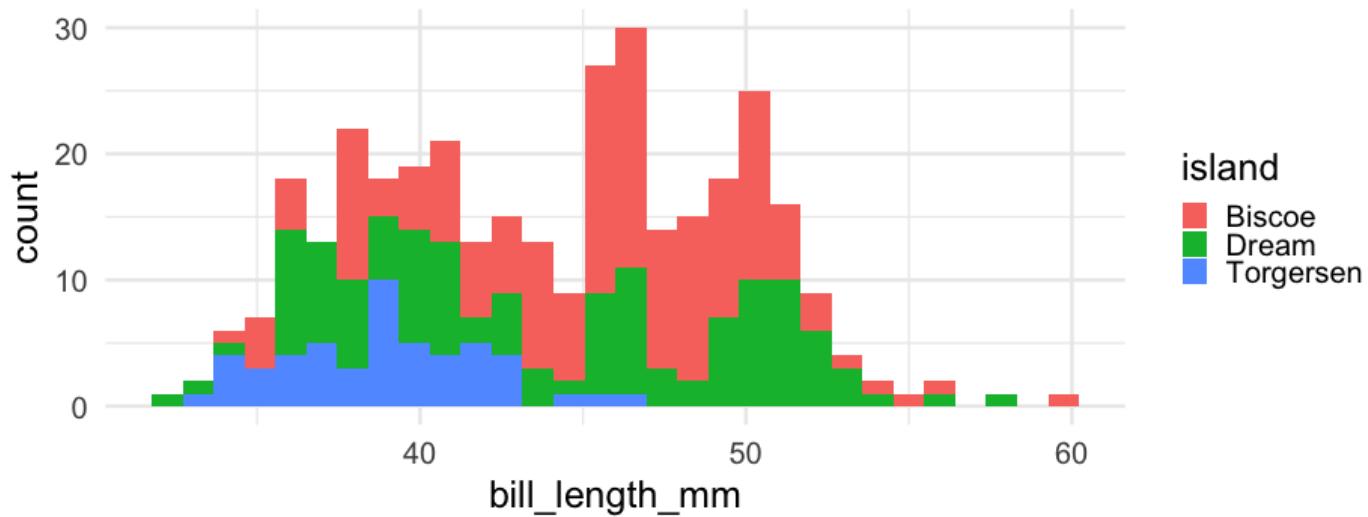
```
ggplot(penguins, aes(island, bill_length_mm)) +  
  ggforce::geom_sina()
```



# Stacked histogram

---

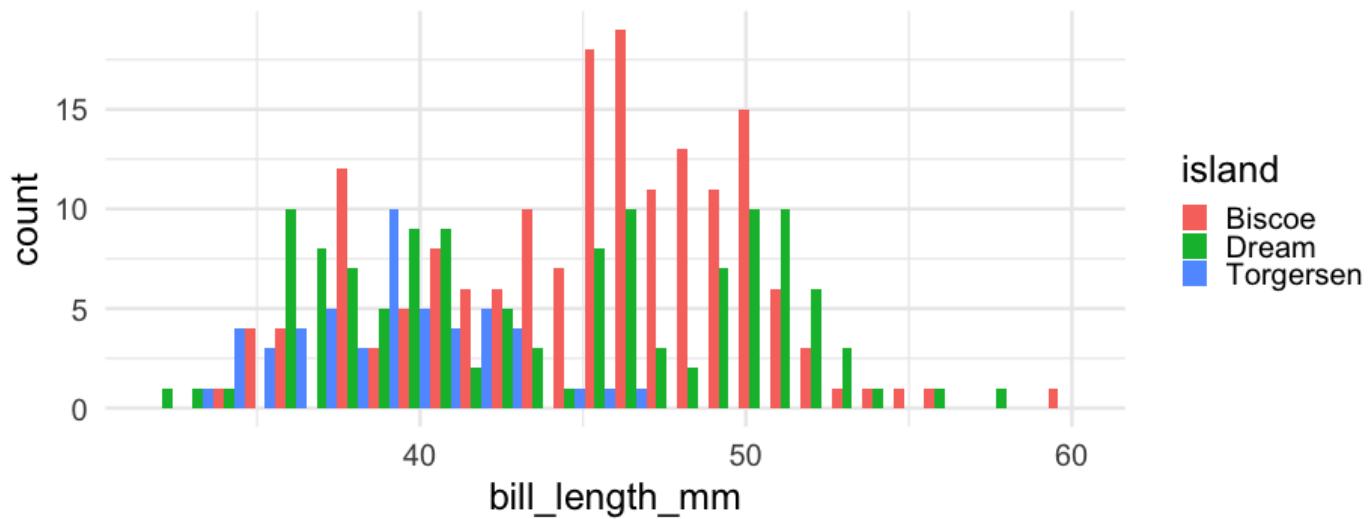
```
ggplot(penguins, aes(bill_length_mm)) +  
  geom_histogram(aes(fill = island))
```



# Dodged

---

```
ggplot(penguins, aes(bill_length_mm)) +  
  geom_histogram(aes(fill = island),  
                 position = "dodge")
```

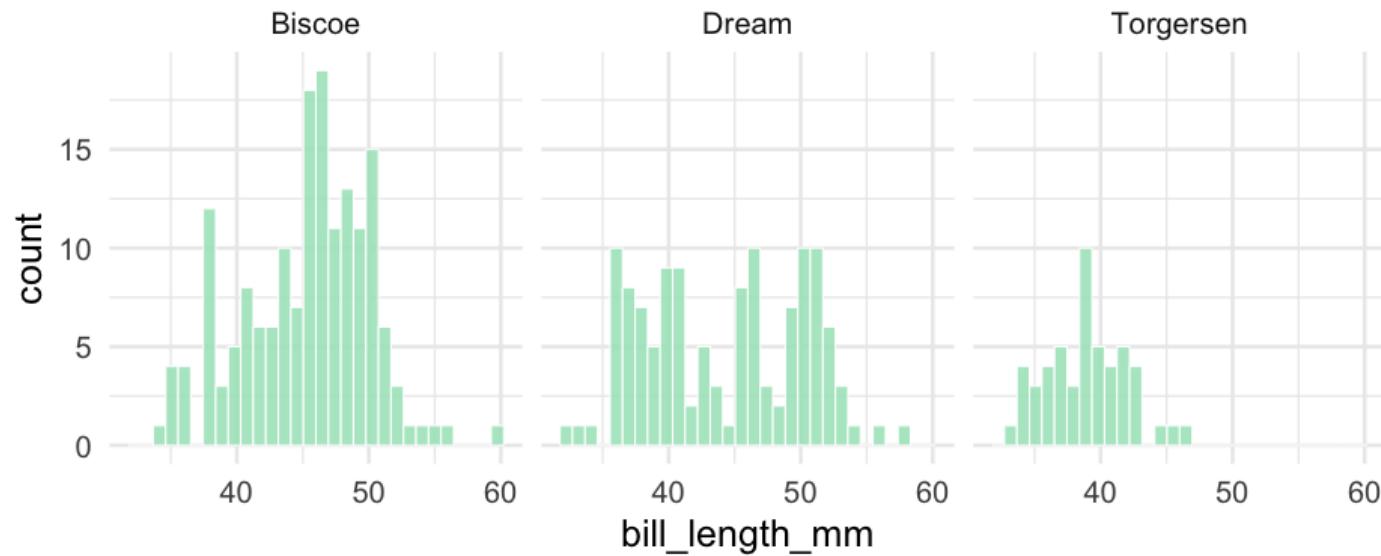


Note **position = "dodge"** does not go into **aes** (not accessing a variable in your dataset)

# Better

---

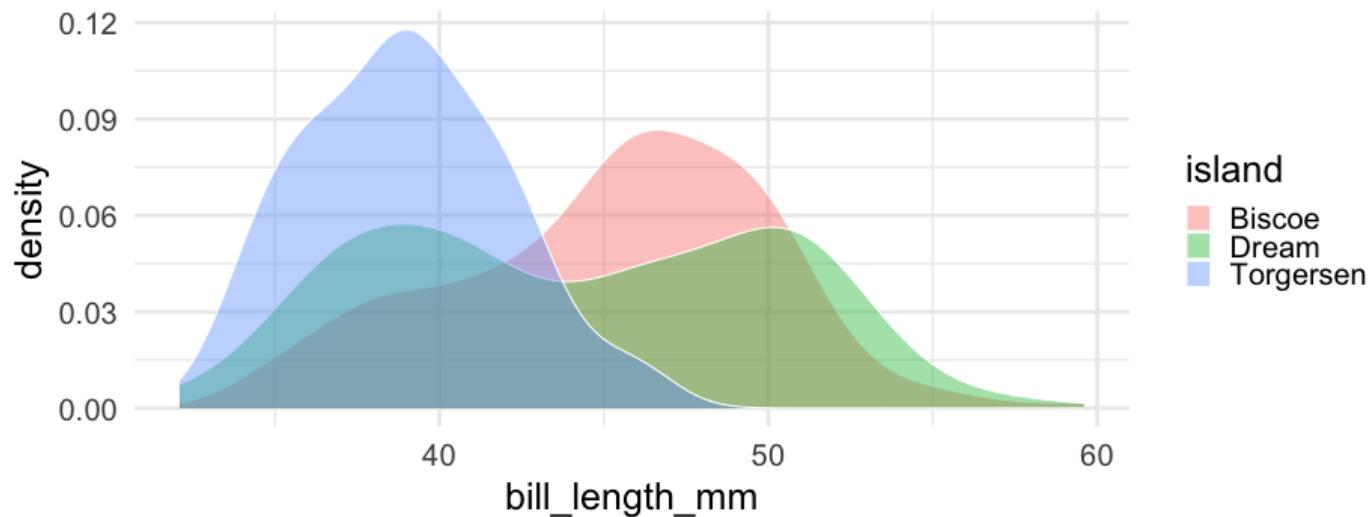
```
ggplot(penguins, aes(bill_length_mm)) +  
  geom_histogram(fill = "#A9E5C5",  
                 color = "white",  
                 alpha = 0.9,) +  
  facet_wrap(~island)
```



# Overlapping densities

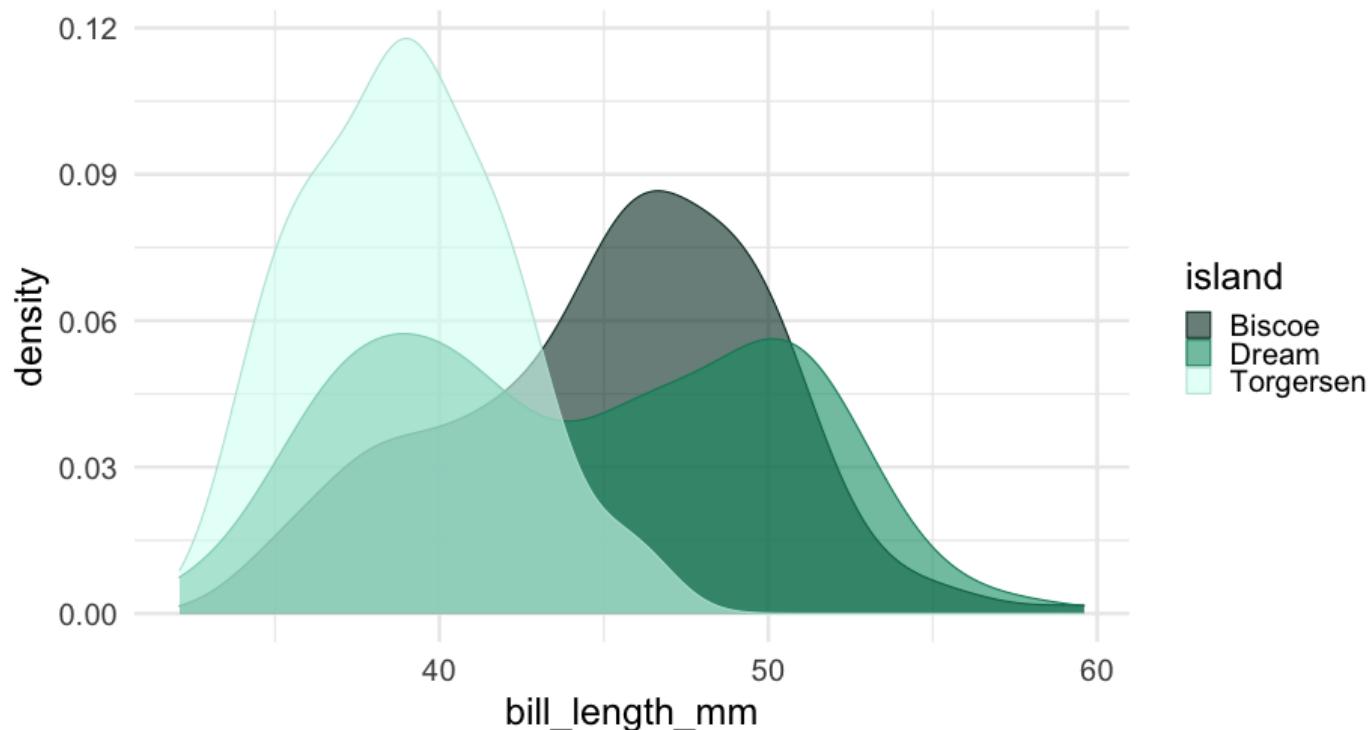
---

```
ggplot(penguins, aes(bill_length_mm)) +  
  geom_density(aes(fill = island),  
               color = "white",  
               alpha = 0.4)
```



Note the default colors generally don't work great in most of these

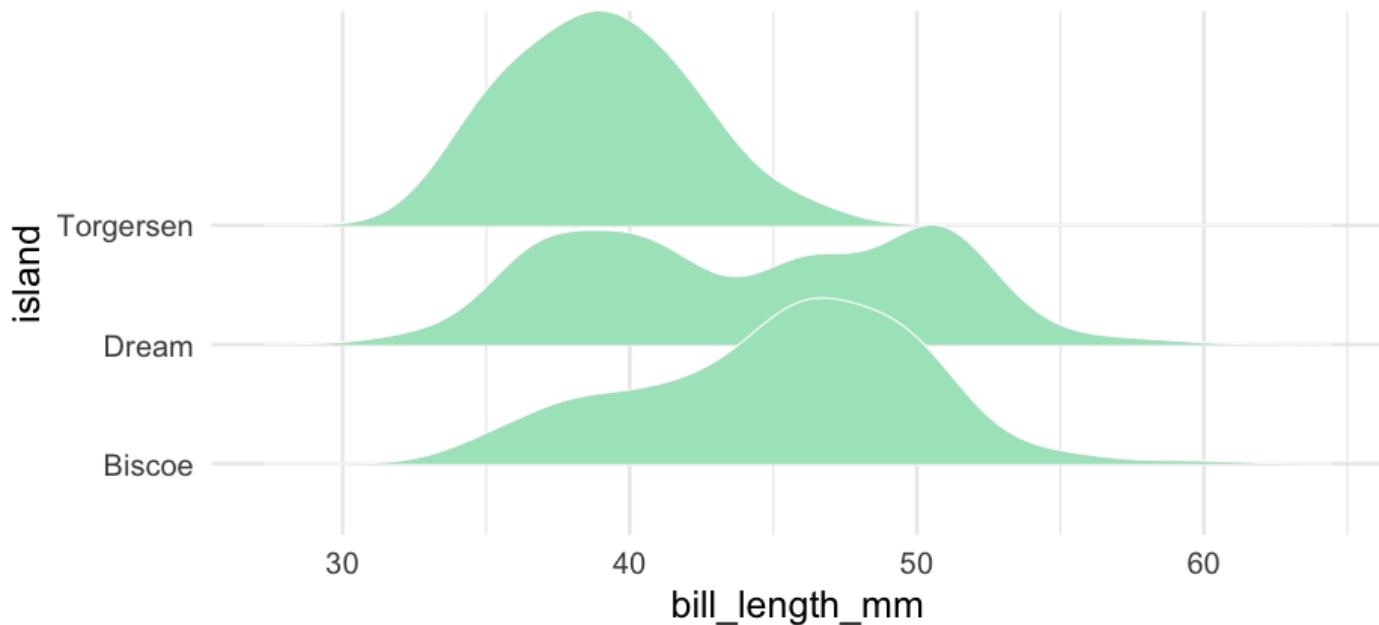
```
ggplot(penguins, aes(bill_length_mm)) +  
  geom_density(aes(color = island, fill = island),  
               alpha = 0.6) +  
  scale_fill_manual(values = c("#003326", "#009973", "#d6ffff"))  
  scale_color_manual(  
    values = darken(c("#003326", "#009973", "#d6ffff"), .1)  
)
```



# Ridgeline densities

---

```
ggplot(penguins, aes(bill_length_mm, island)) +  
  ggridges::geom_density_ridges(color = "white",  
                                 fill = "#A9E5C5")
```

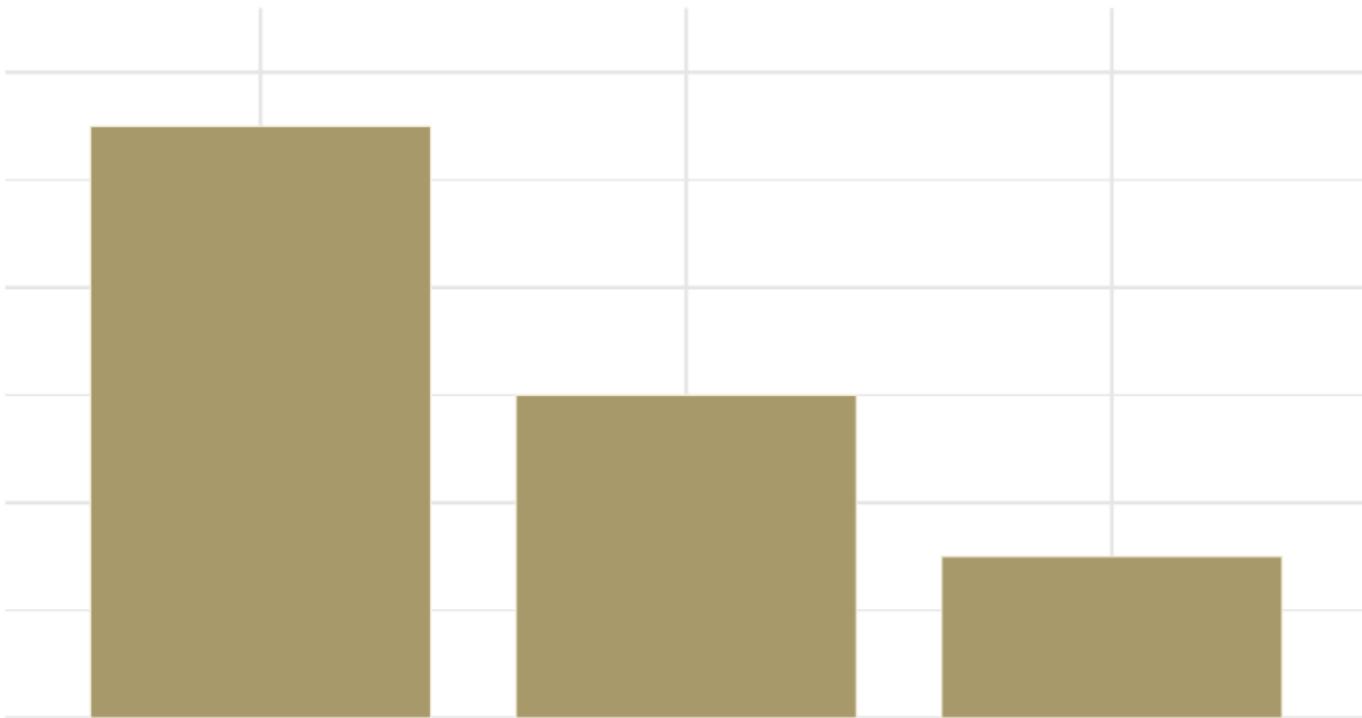


# Visualizing amounts

---

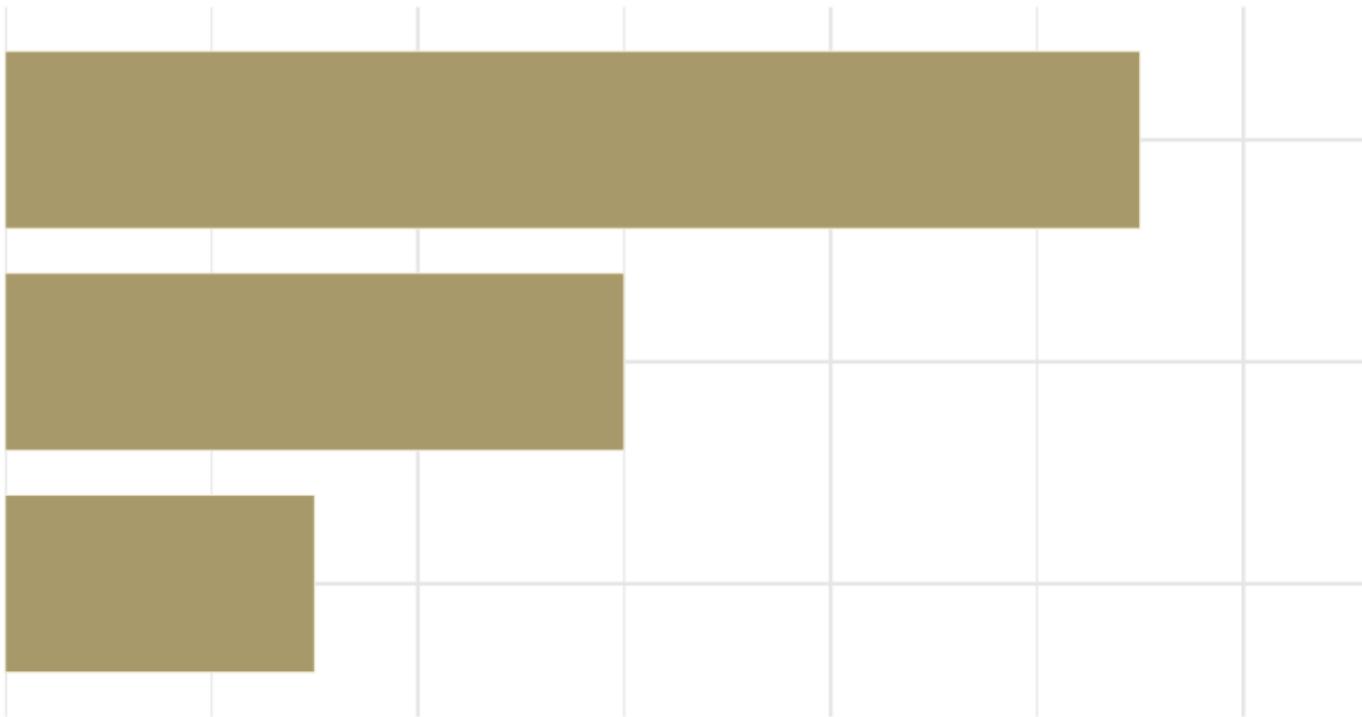
# Bar plots

---



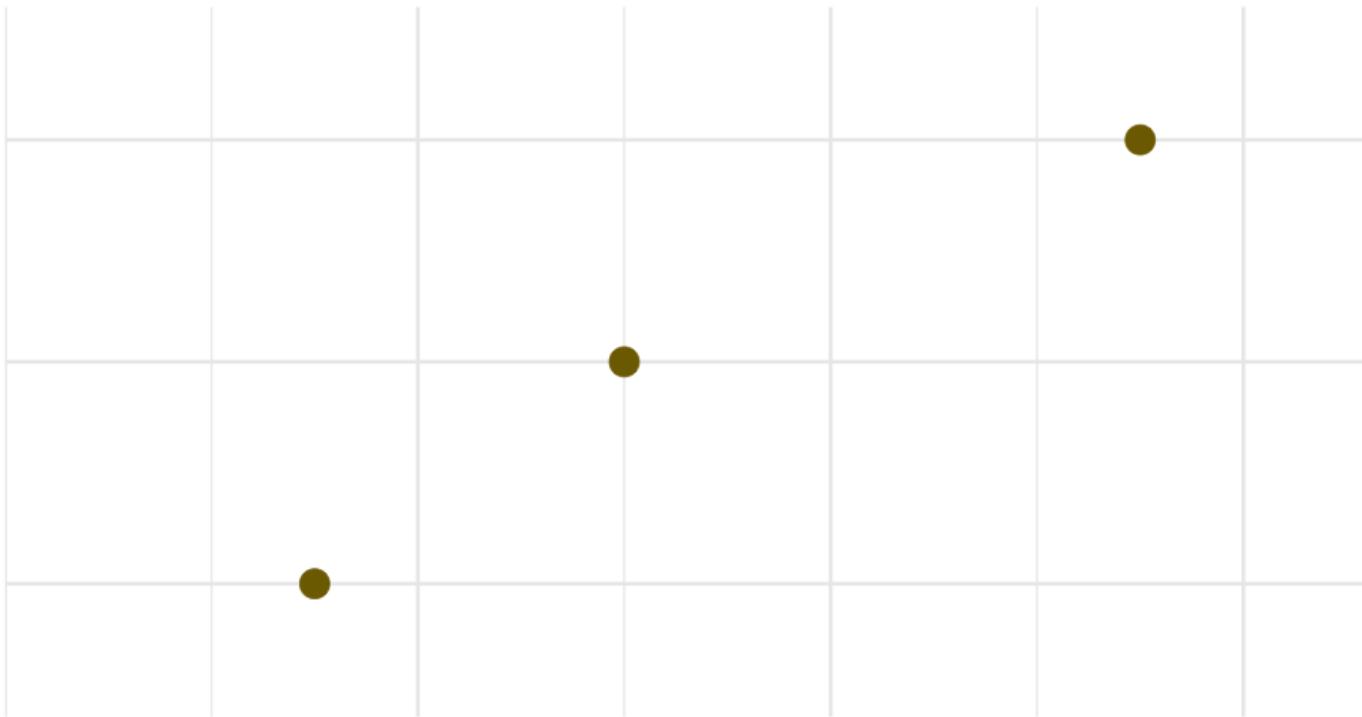
# Flipped bars

---



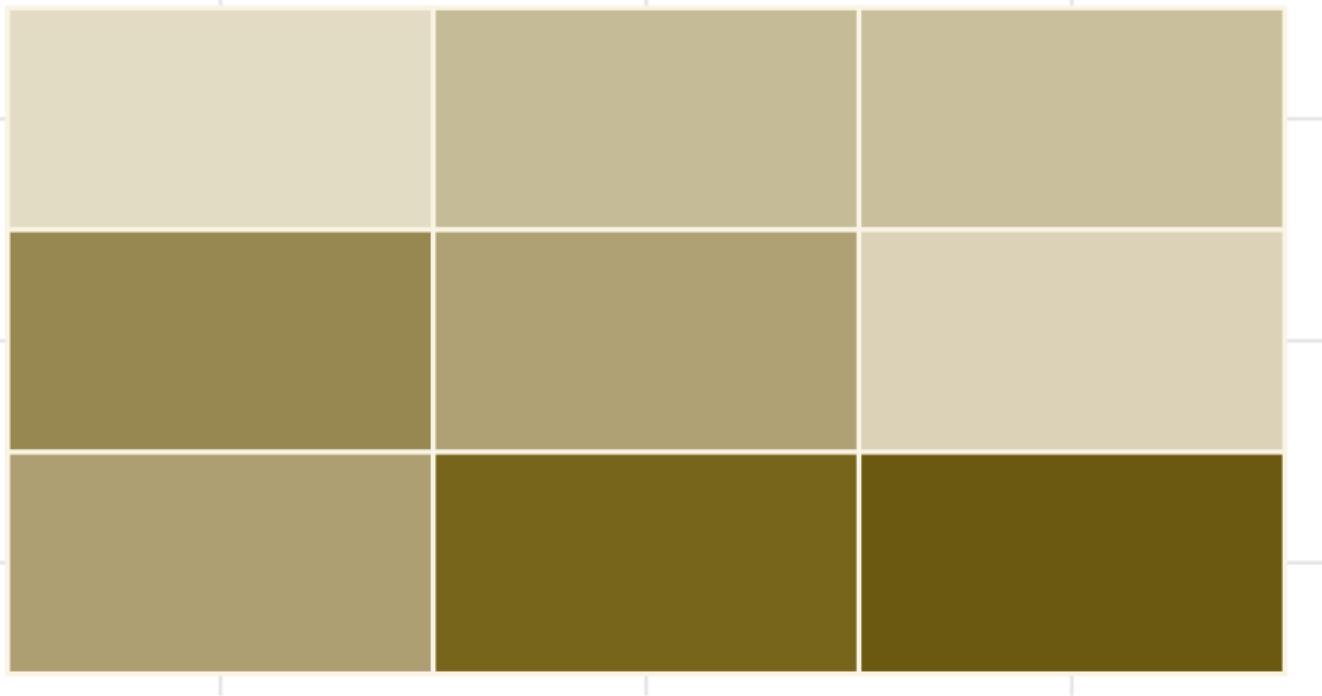
# Dotplot

---



# Heatmap

---



# Empirical examples

---

How much does college cost?

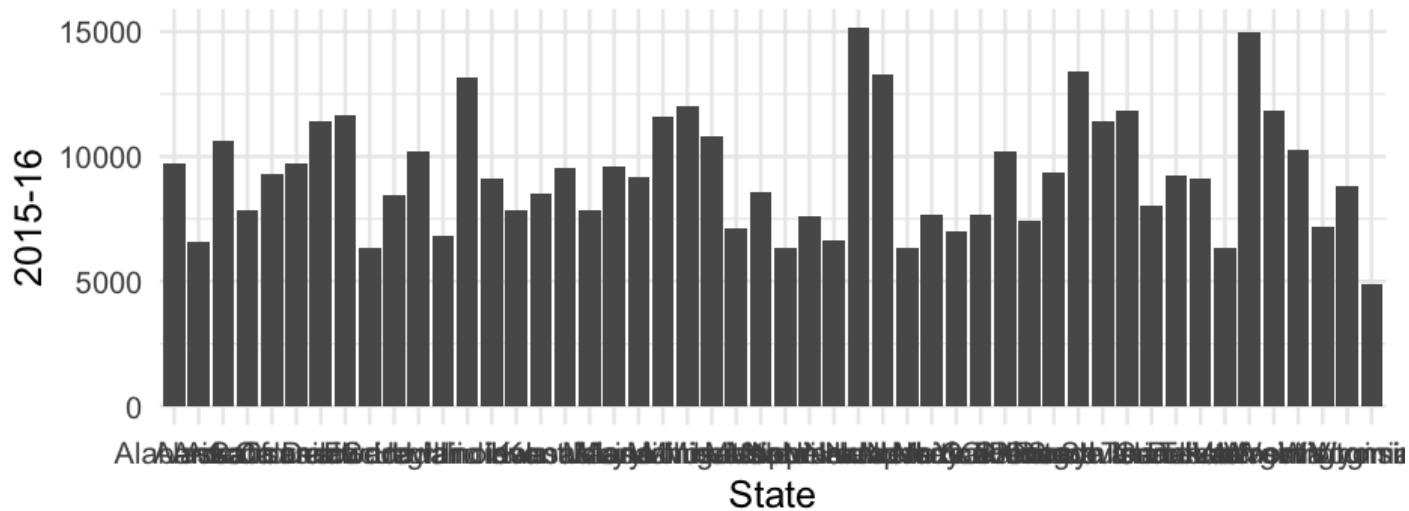
```
library(here)
library(rio)
tuition <- import(here("data", "us_avg_tuition.xlsx"),
                  setclass = "tbl_df")
head(tuition)
```

```
## # A tibble: 6 × 13
##   State    `2004-05` `2005-06` `2006-07` `2007-08` `2008-09` `2009-10` ...
##   <chr>     <dbl>     <dbl>     <dbl>     <dbl>     <dbl>     <dbl> ...
## 1 Alabama  5682.838  5840.550  5753.496  6008.169  6475.092  7188.954
## 2 Alaska   4328.281  4632.623  4918.501  5069.822  5075.482  5454.607
## 3 Arizona  5138.495  5415.516  5481.419  5681.638  6058.464  7263.204
## 4 Arkansas 5772.302  6082.379  6231.977  6414.900  6416.503  6627.092
## 5 California 5285.921  5527.881  5334.826  5672.472  5897.888  7258.771
## 6 Colorado  4703.777  5406.967  5596.348  6227.002  6284.137  6948.473
## # ... with 6 more variables: 2010-11 <dbl>, 2011-12 <dbl>, 2012-13 <dbl>,
## #   2013-14 <dbl>, 2014-15 <dbl>, 2015-16 <dbl>
```

# By state: 2015–16

---

```
ggplot(tuition, aes(State, `2015-16`)) +  
  geom_col()
```

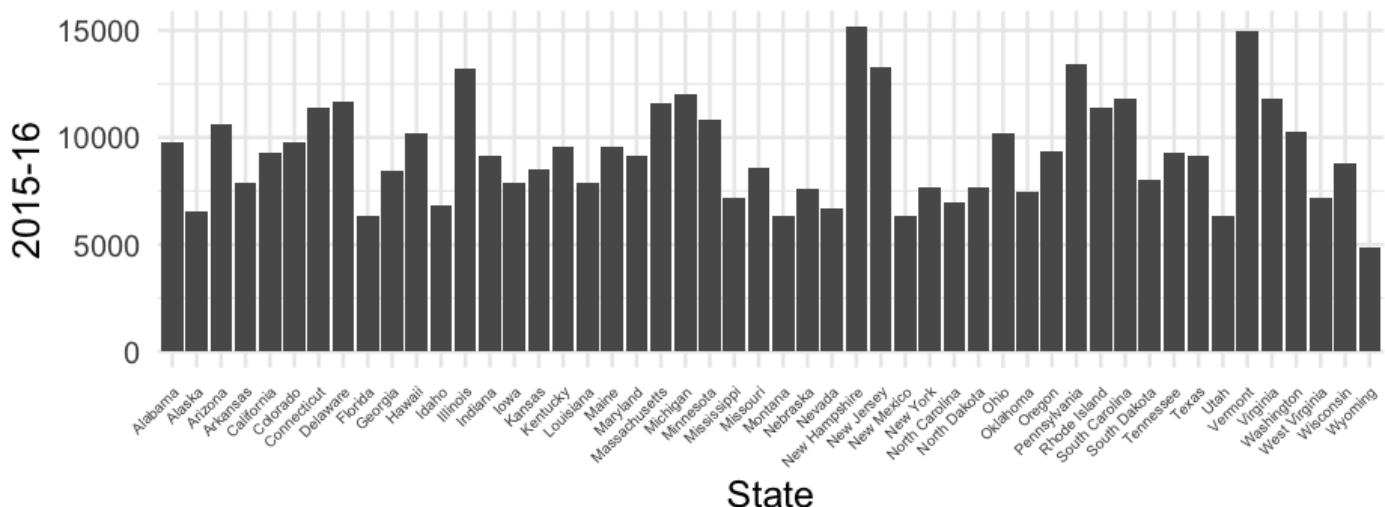


# Two puke emoji version

---



```
ggplot(tuition, aes(State, `2015-16`)) +  
  geom_col() +  
  theme(axis.text.x = element_text(angle = 45, hjust = 1, size =
```

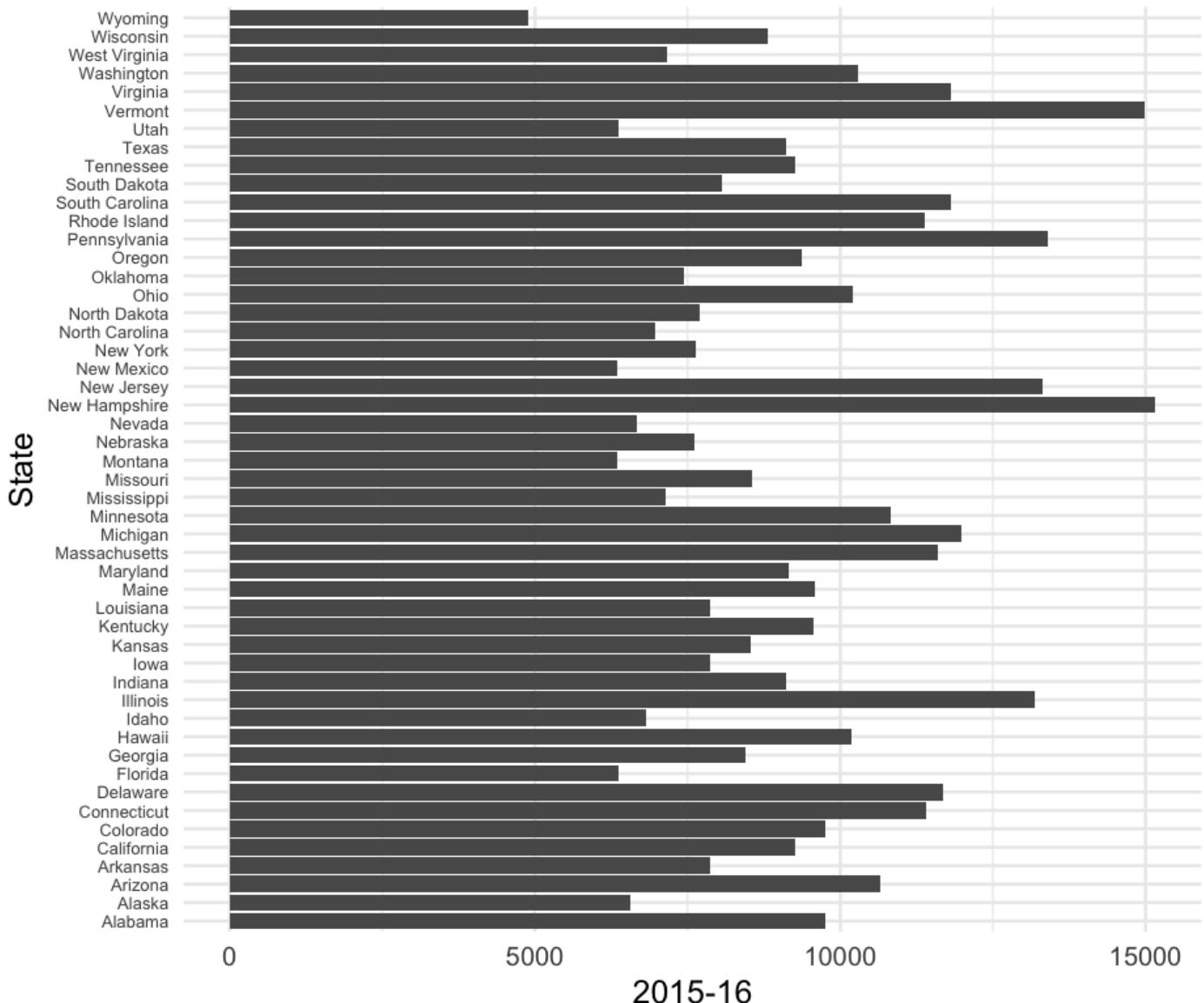


# One puke emoji version

---



```
ggplot(tuition, aes(State, `2015-16`)) +  
  geom_col() +  
  coord_flip()
```



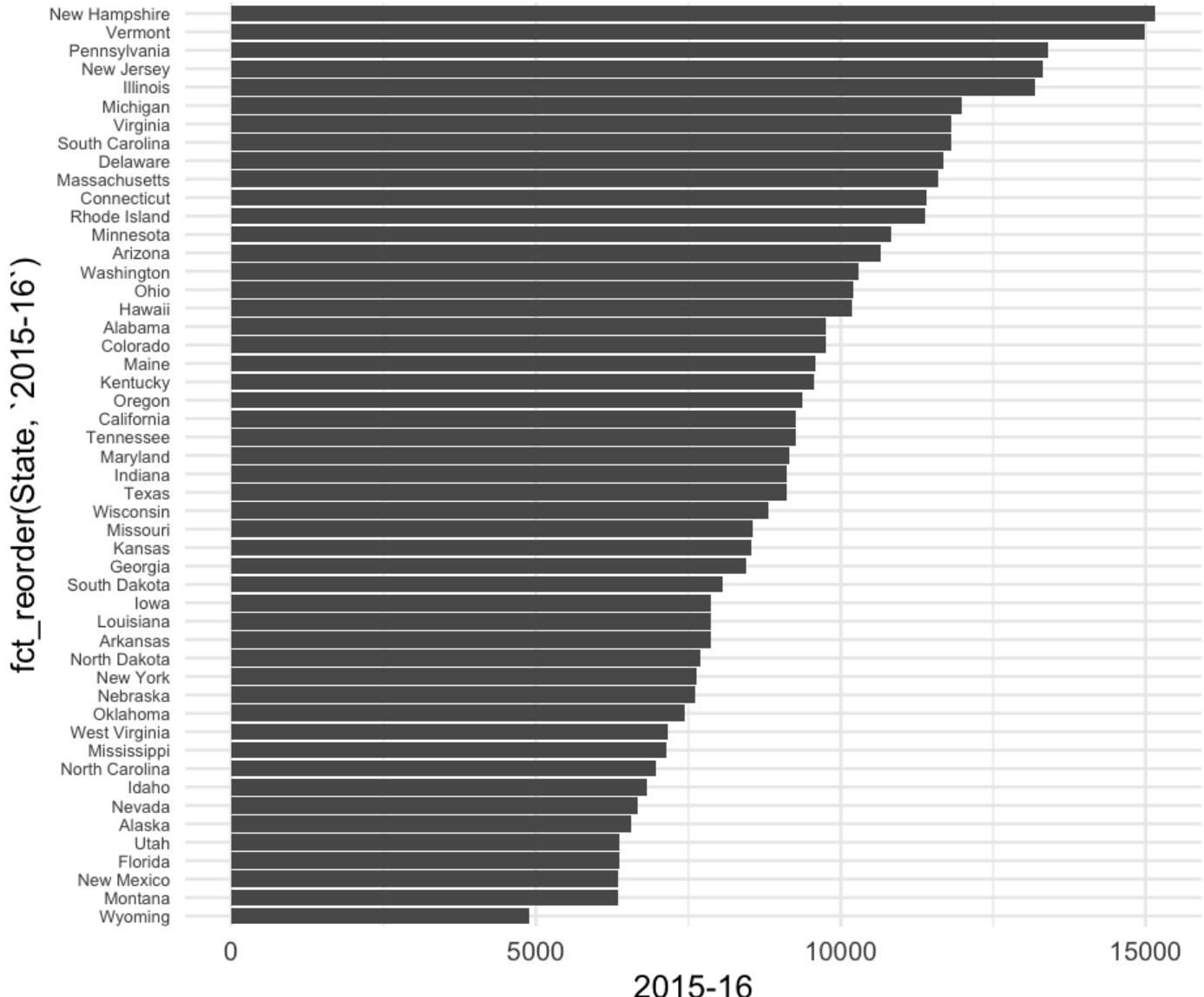
2015-16

# Kinda smiley version

---



```
ggplot(tuition, aes(fct_reorder(State, `2015-16`), `2015-16`)) +  
  geom_col() +  
  coord_flip()
```

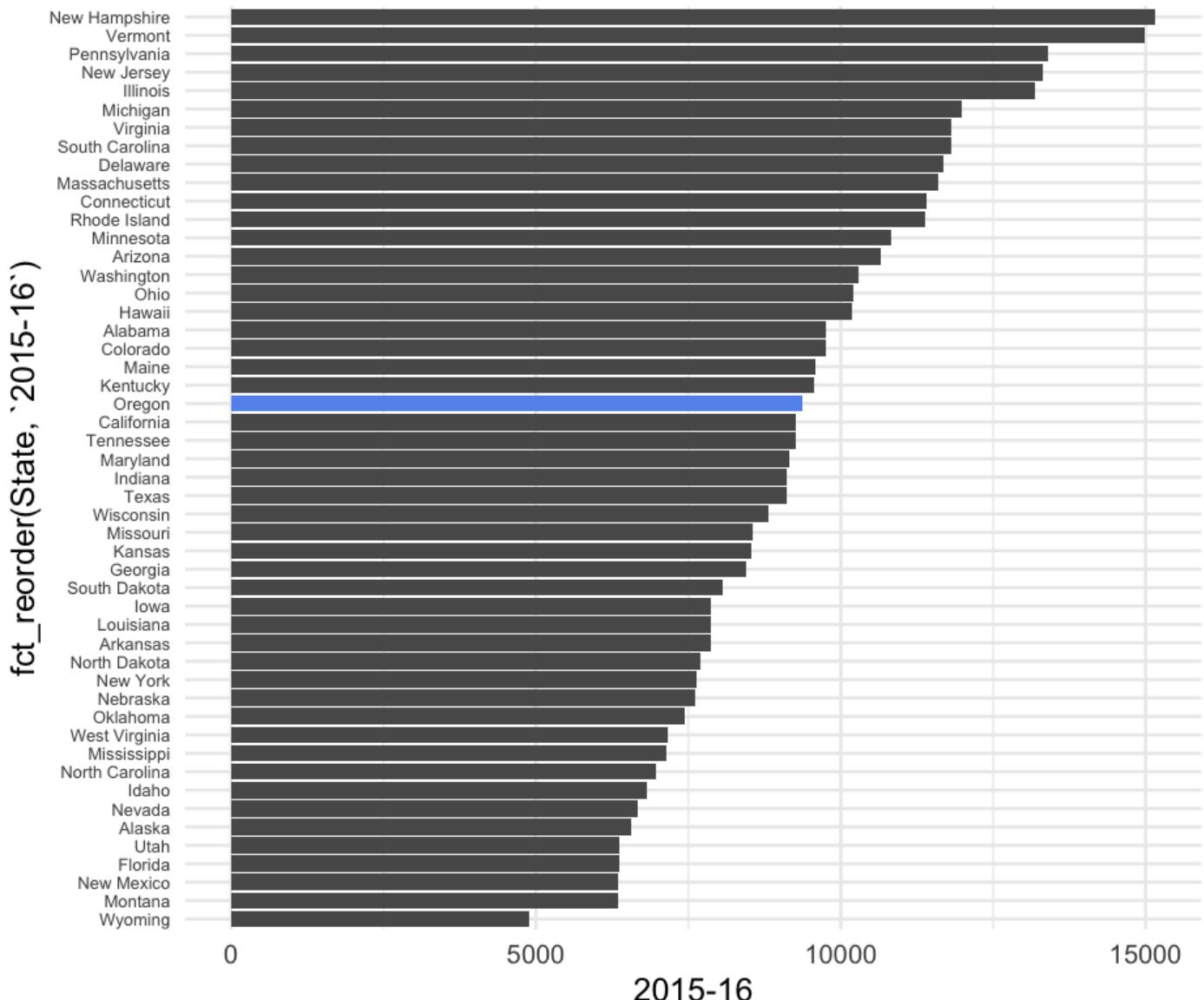


# Highlight Oregon

---

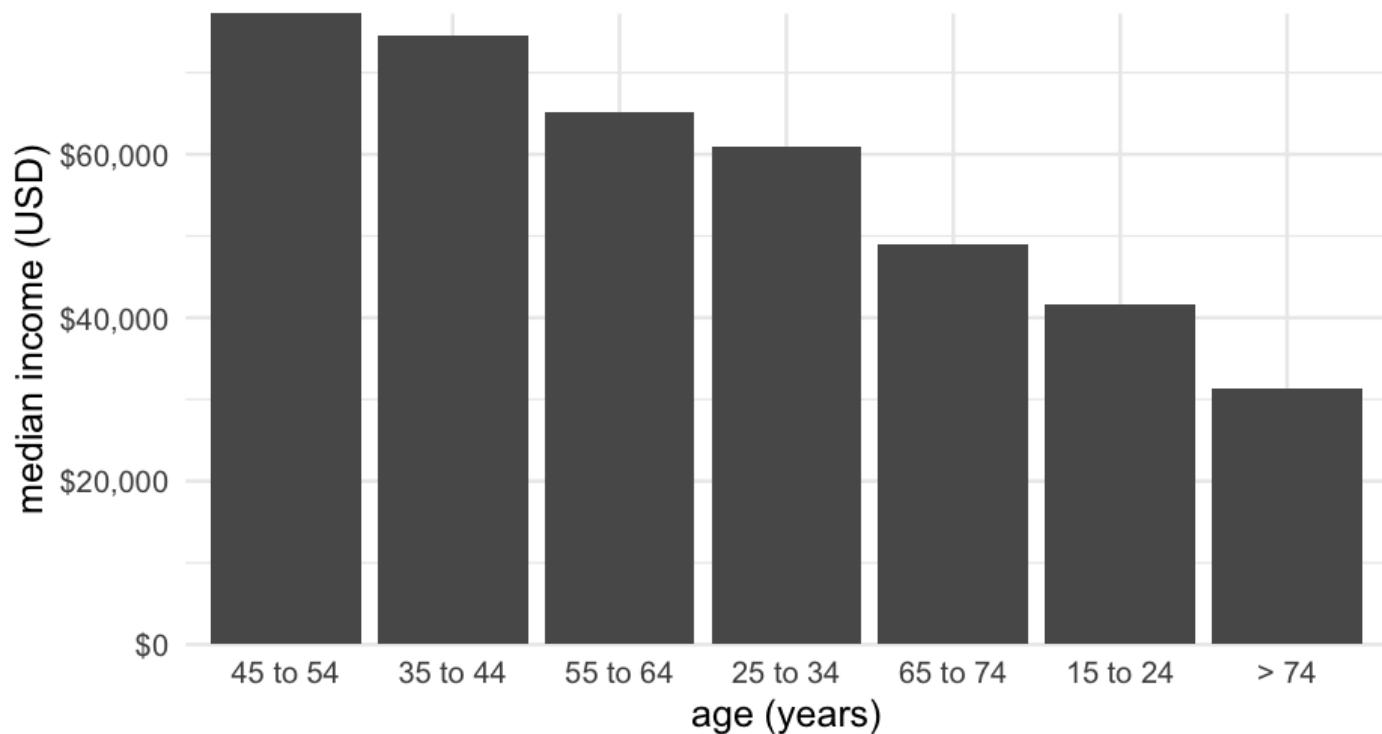


```
ggplot(tuition, aes(fct_reorder(State, `2015-16`), `2015-16`)) +  
  geom_col() +  
  geom_col(fill = "cornflowerblue",  
           data = filter(tuition, State == "Oregon")) +  
  coord_flip()
```



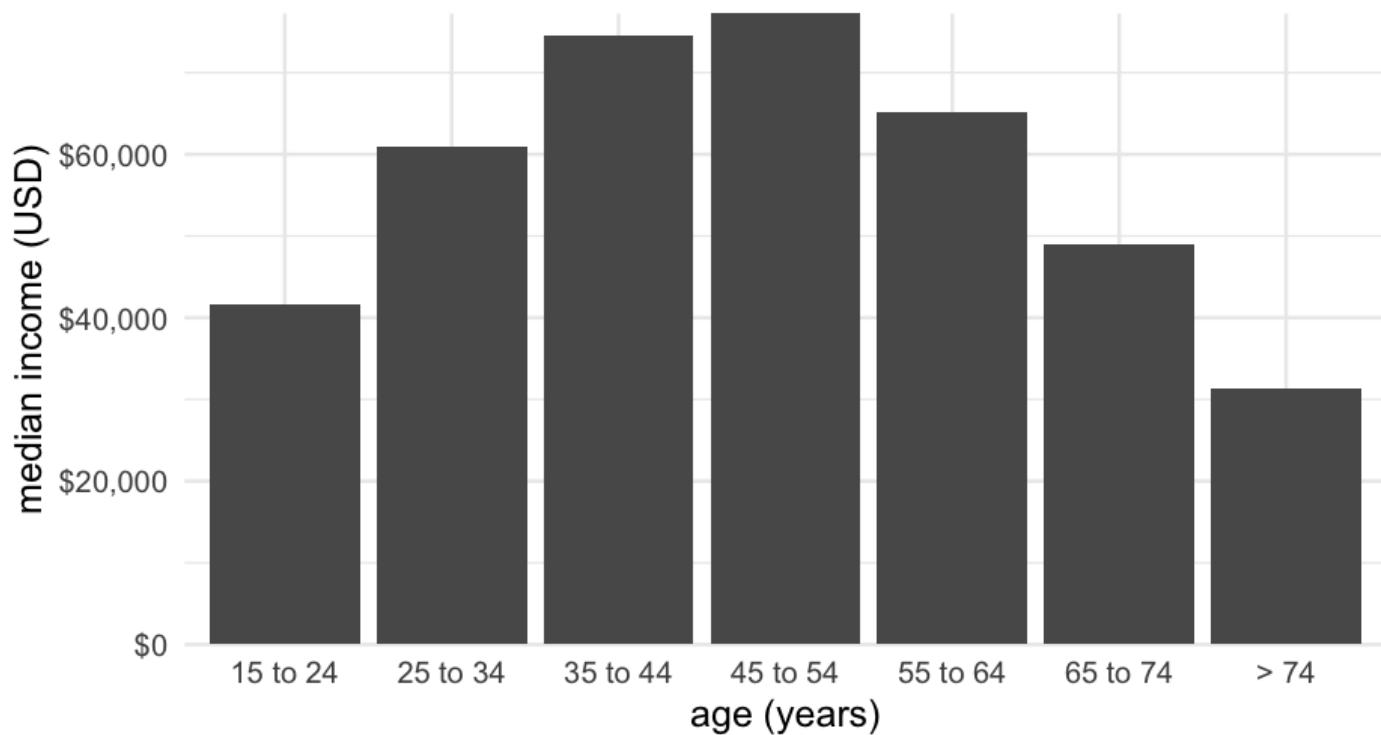
# Not always good to sort

---



# Much better

---



# Average tuition by year

---

How?

```
head(tuition)
```

```
## # A tibble: 6 × 13
##   State    `2004-05` `2005-06` `2006-07` `2007-08` `2008-09` `2009-10`
##   <chr>     <dbl>     <dbl>     <dbl>     <dbl>     <dbl>     <dbl>
## 1 Alabama  5682.838  5840.550  5753.496  6008.169  6475.092  7188.954
## 2 Alaska   4328.281  4632.623  4918.501  5069.822  5075.482  5454.607
## 3 Arizona  5138.495  5415.516  5481.419  5681.638  6058.464  7263.204
## 4 Arkansas 5772.302  6082.379  6231.977  6414.900  6416.503  6627.092
## 5 California 5285.921  5527.881  5334.826  5672.472  5897.888  7258.771
## 6 Colorado  4703.777  5406.967  5596.348  6227.002  6284.137  6948.473
## # ... with 6 more variables: 2010-11 <dbl>, 2011-12 <dbl>, 2012-13 <dbl>,
## #   2013-14 <dbl>, 2014-15 <dbl>, 2015-16 <dbl>
```

# Rearrange

---

```
tuition %>%
  pivot_longer(`2004-05`:`2015-16`,
              names_to = "year",
              values_to = "avg_tuition")
```

```
## # A tibble: 600 × 3
##       State   year   avg_tuition
##       <chr>   <chr>     <dbl>
## 1 Alabama 2004-05    5682.838
## 2 Alabama 2005-06    5840.550
## 3 Alabama 2006-07    5753.496
## 4 Alabama 2007-08    6008.169
## 5 Alabama 2008-09    6475.092
## 6 Alabama 2009-10    7188.954
## 7 Alabama 2010-11    8071.134
## 8 Alabama 2011-12    8451.902
## 9 Alabama 2012-13    9098.069
## 10 Alabama 2013-14   9358.929
## # ... with 590 more rows
```

# Compute summaries

---

```
annual_means <- tuition %>%
  pivot_longer(`2004-05`:`2015-16`,
              names_to = "year",
              values_to = "avg_tuition") %>%
  group_by(year) %>%
  summarize(mean_tuition = mean(avg_tuition))

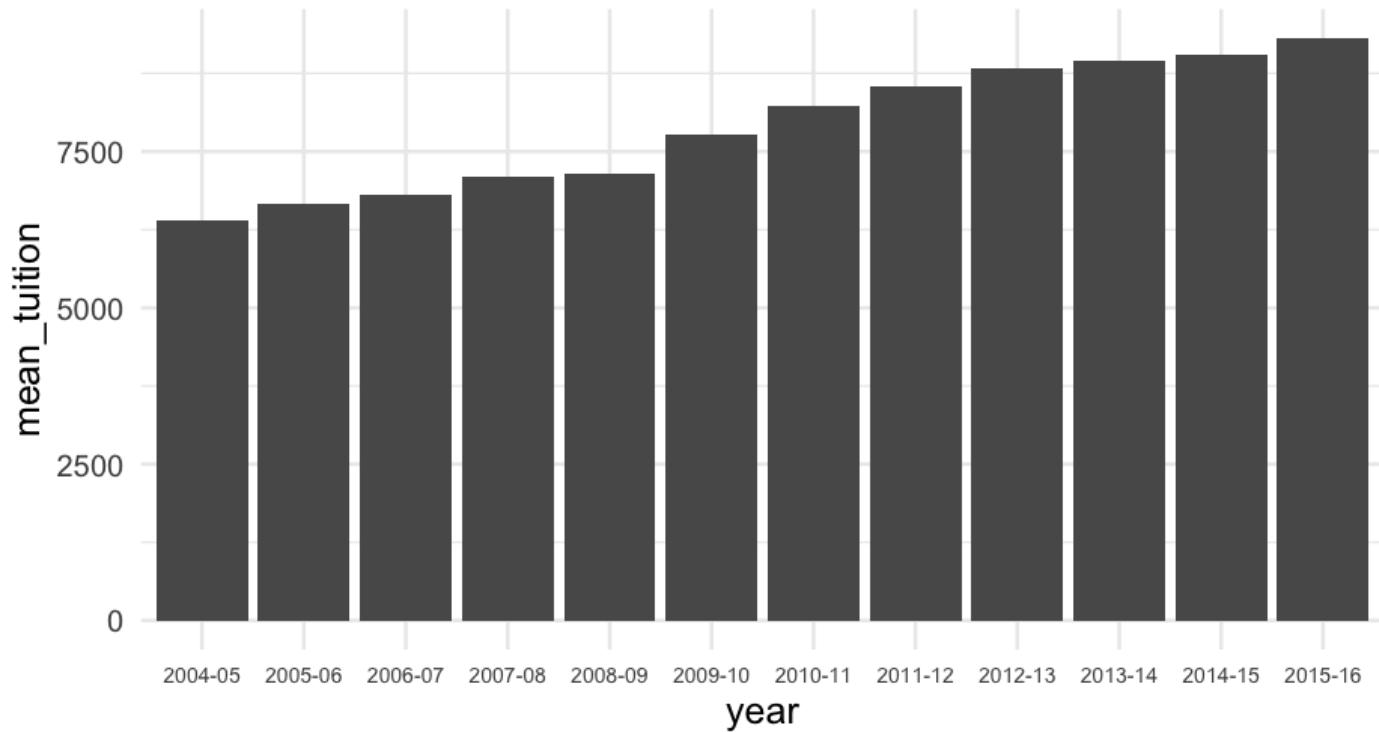
annual_means
```

```
## # A tibble: 12 × 2
##       year   mean_tuition
##   <chr>     <dbl>
## 1 2004-05    6409.564
## 2 2005-06    6654.177
## 3 2006-07    6809.914
## 4 2007-08    7085.881
## 5 2008-09    7156.560
## 6 2009-10    7761.810
## 7 2010-11    8228.834
## 8 2011-12    8539.115
## 9 2012-13    8842.357
## 10 2013-14   8947.938
## 11 2014-15   9037.357
## 12 2015-16   9317.633
```

# Good

---

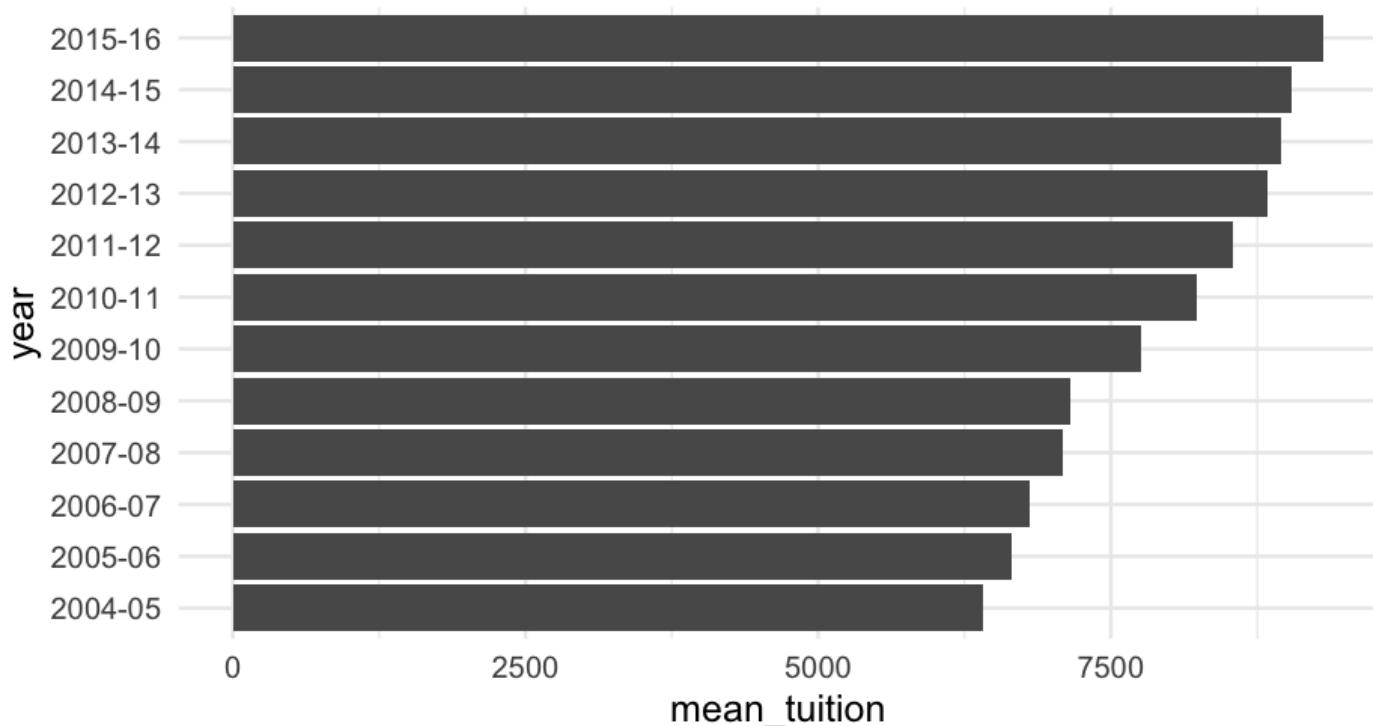
```
ggplot(annual_means, aes(year, mean_tuition)) +  
  geom_col()
```



# Better?

---

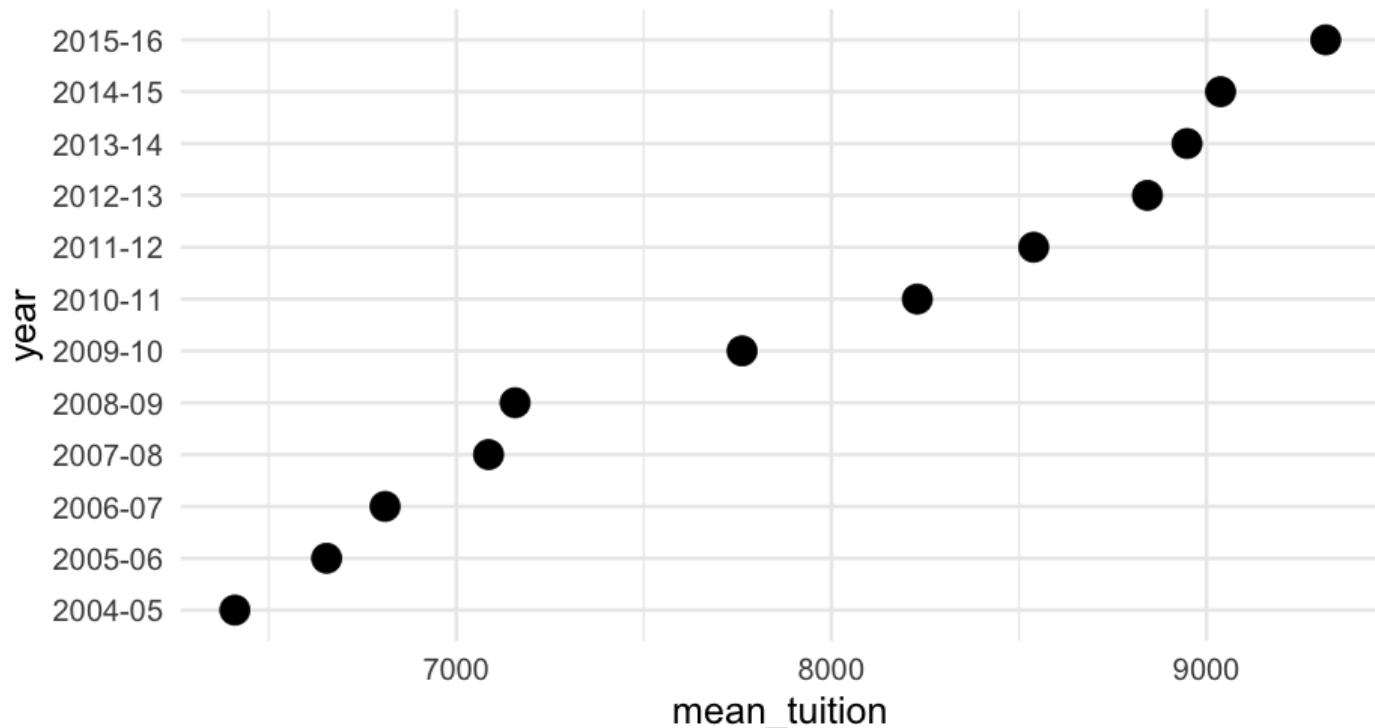
```
ggplot(annual_means, aes(year, mean_tuition)) +  
  geom_col() +  
  coord_flip()
```



# Better still?

---

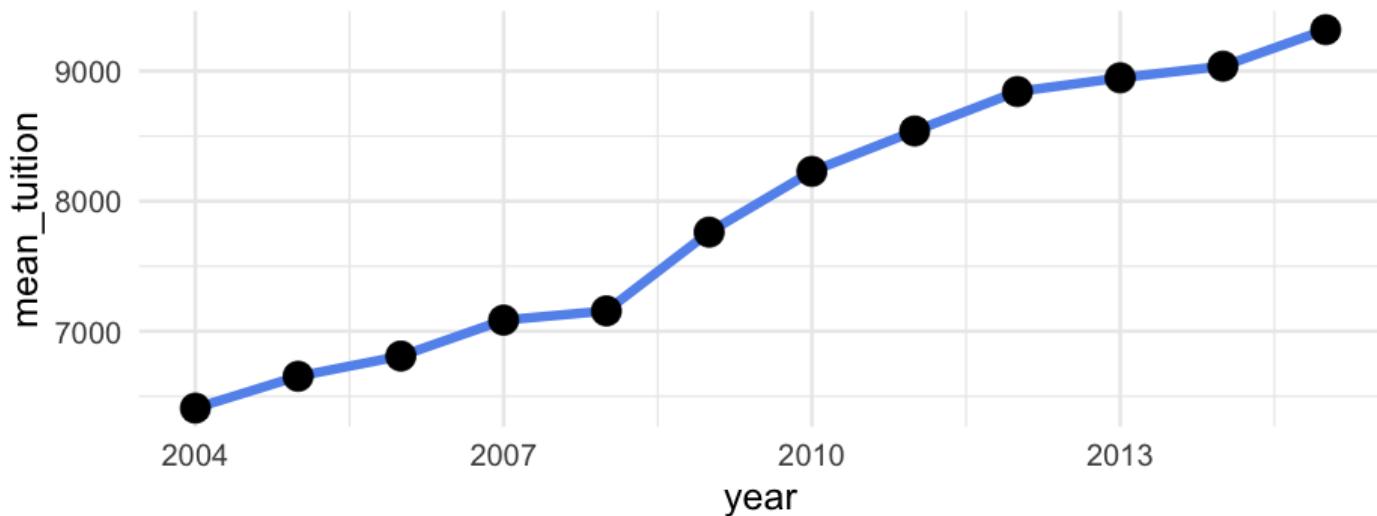
```
ggplot(annual_means, aes(year, mean_tuition)) +  
  geom_point() +  
  coord_flip()
```



# Even better

---

```
annual_means %>%
  mutate(year = readr::parse_number(year)) %>%
  ggplot(aes(year, mean_tuition)) +
  geom_line(color = "cornflowerblue") +
  geom_point()
```



Treat time (year) as a continuous variable

# Grouped points

---

Show change in tuition from 05–06 to 2015–16

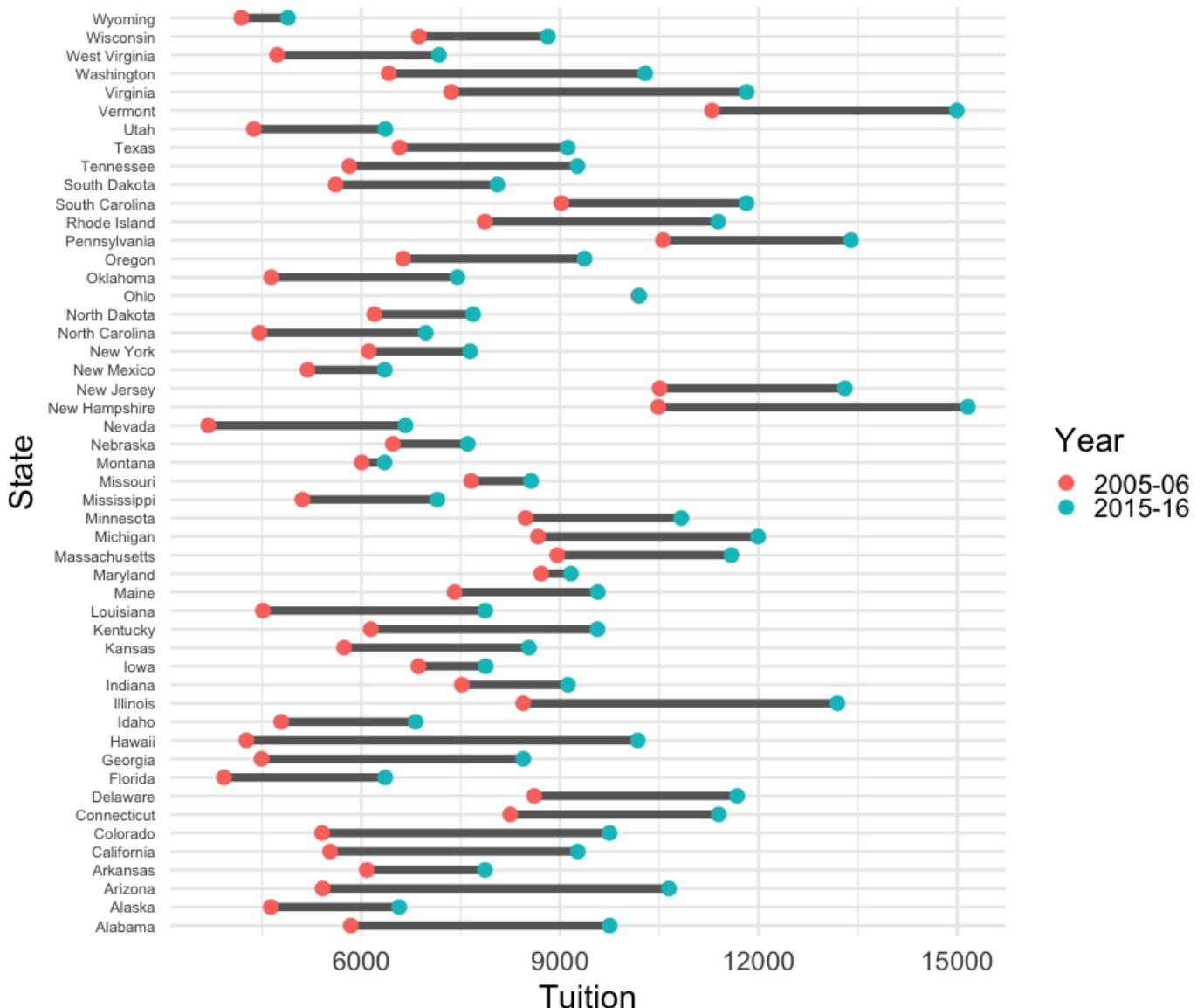
```
tuition %>%
  select(State, `2005-06`, `2015-16`)
```

```
## # A tibble: 50 × 3
##       State   `2005-06` `2015-16`
##       <chr>     <dbl>     <dbl>
## 1 Alabama    5840.550  9751.101
## 2 Alaska     4632.623  6571.340
## 3 Arizona    5415.516 10646.28 
## 4 Arkansas   6082.379  7867.297
## 5 California 5527.881  9269.844
## 6 Colorado    5406.967  9748.188
## 7 Connecticut 8249.074 11397.34 
## 8 Delaware   8610.597 11676.22 
## 9 Florida    3924.234  6360.159
## 10 Georgia   4492.167  8446.961
## # ... with 40 more rows
```

```
lt <- tuition %>%
  select(State, `2005-06`, `2015-16`) %>%
  pivot_longer(`2005-06`:`2015-16`,
              names_to = "Year",
              values_to = "Tuition")
lt
```

```
## # A tibble: 100 × 3
##       State     Year   Tuition
##       <chr>    <chr>    <dbl>
## 1 Alabama 2005-06 5840.550
## 2 Alabama 2015-16 9751.101
## 3 Alaska   2005-06 4632.623
## 4 Alaska   2015-16 6571.340
## 5 Arizona  2005-06 5415.516
## 6 Arizona  2015-16 10646.28
## 7 Arkansas 2005-06 6082.379
## 8 Arkansas 2015-16 7867.297
## 9 California 2005-06 5527.881
## 10 California 2015-16 9269.844
## # ... with 90 more rows
```

```
ggplot(lt, aes(State, Tuition)) +  
  geom_line(aes(group = State), color = "gray40") +  
  geom_point(aes(color = Year)) +  
  coord_flip()
```



# Extensions

---

We need to move on to other things, but we definitely would want to keep going here:

- Order states according to something more meaningful (starting tuition, ending tuition, or difference in tuition)
- Meaningful title, e.g., "Change in average tuition over a decade"
- Consider better color scheme for points
- Potentially color the difference line by magnitude

# Let's back up a bit

---

- Lets go back to our full data, but in a format that we can have a **year** variable.

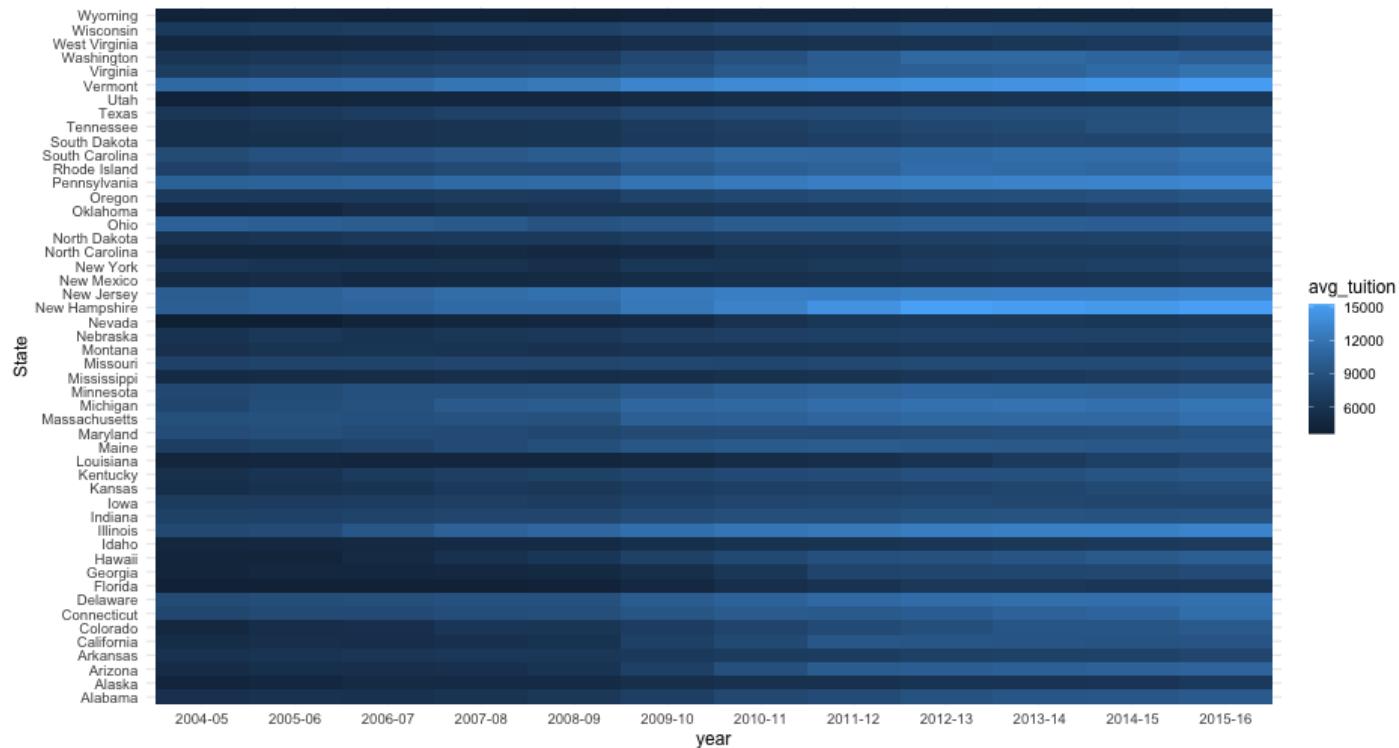
```
tuition_l <- tuition %>%
  pivot_longer(-State,
               names_to = "year",
               values_to = "avg_tuition")  
  
tuition_l
```

```
## # A tibble: 600 × 3
##       State   year   avg_tuition
##       <chr>  <chr>     <dbl>
## 1 Alabama 2004-05    5682.838
## 2 Alabama 2005-06    5840.550
## 3 Alabama 2006-07    5753.496
## 4 Alabama 2007-08    6008.169
## 5 Alabama 2008-09    6475.092
## 6 Alabama 2009-10    7188.954
## 7 Alabama 2010-11    8071.134
## 8 Alabama 2011-12    8451.902
## 9 Alabama 2012-13    9098.069
## 10 Alabama 2013-14   9358.929
```

# Heatmap

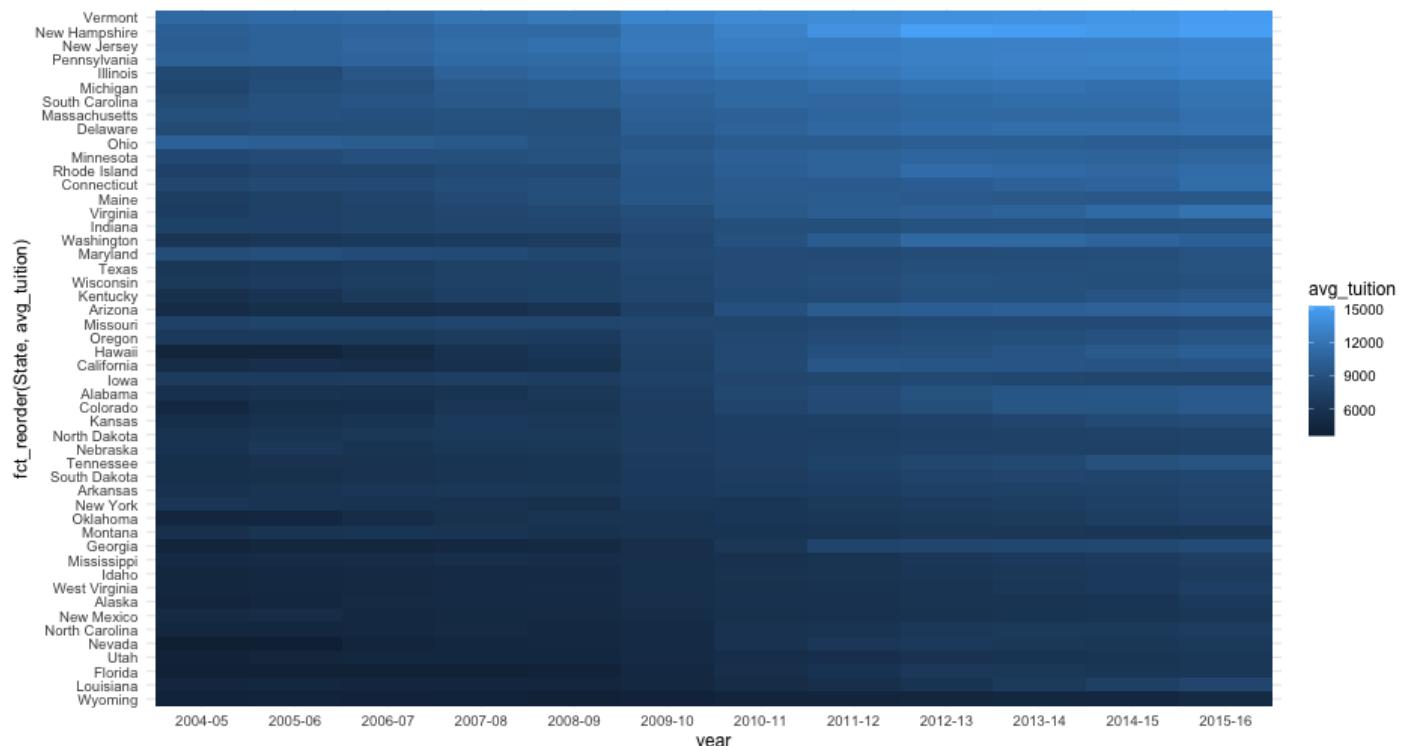
---

```
ggplot(tuition_l, aes(year, State)) +  
  geom_tile(aes(fill = avg_tuition))
```



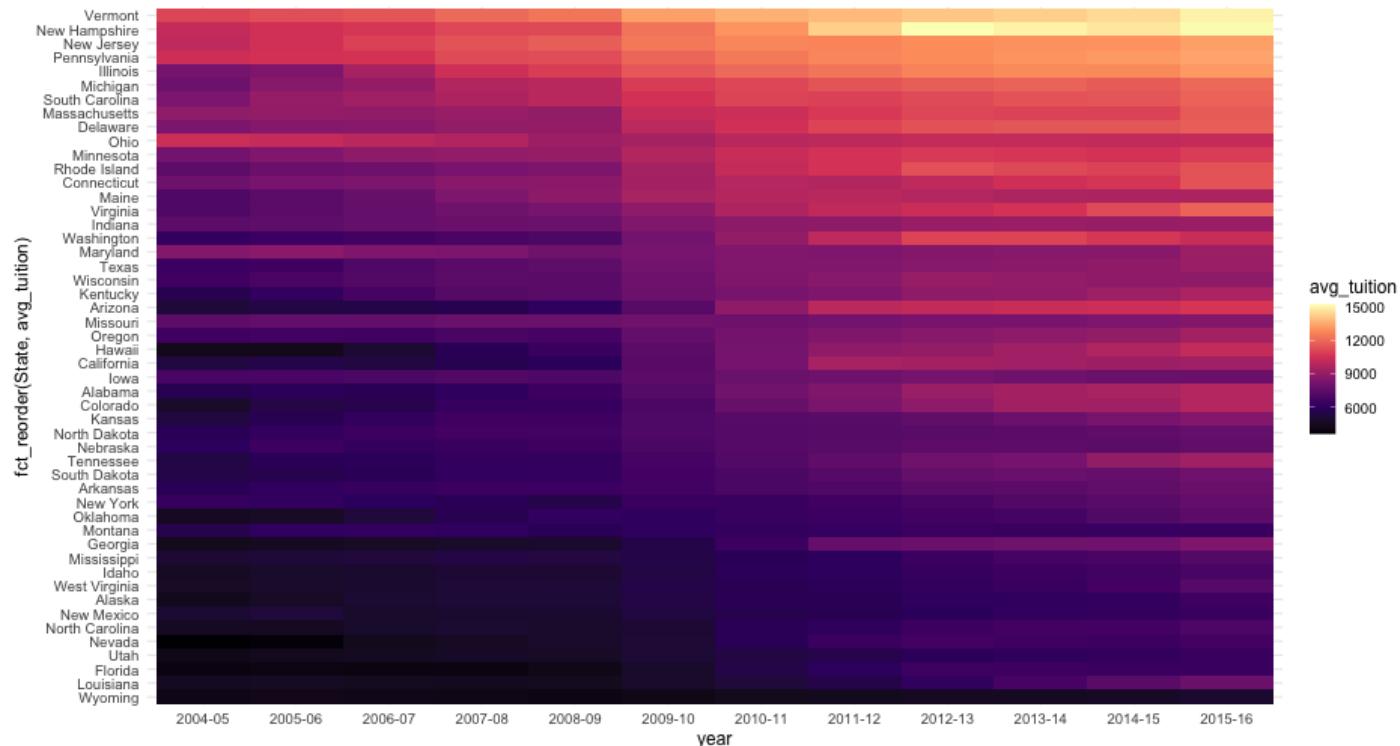
# Better heatmap

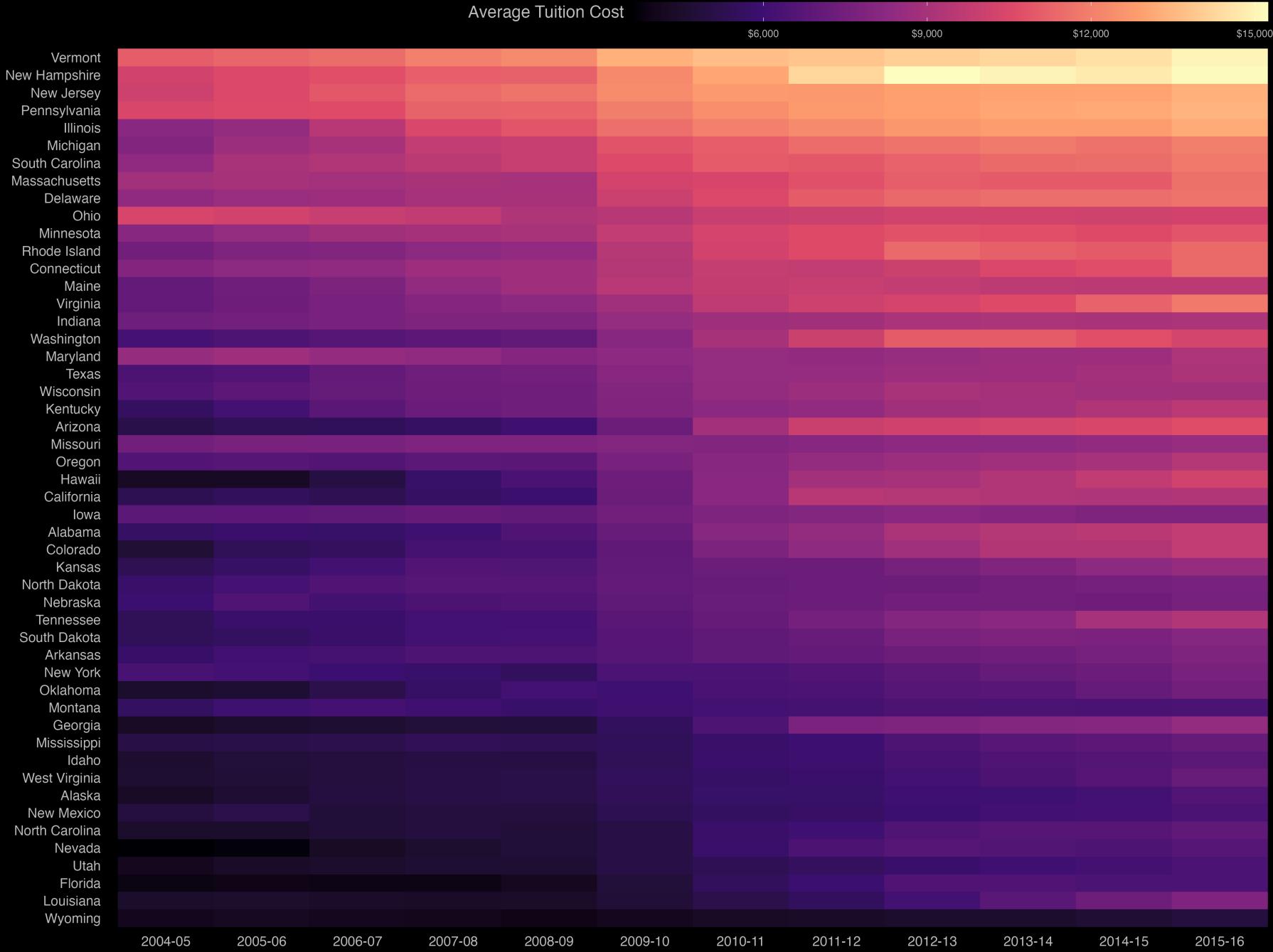
```
ggplot(tuition_l, aes(year, fct_reorder(State, avg_tuition))) +  
  geom_tile(aes(fill = avg_tuition))
```



# Even better heatmap

```
ggplot(tuition_l, aes(year, fct_reorder(State, avg_tuition))) +  
  geom_tile(aes(fill = avg_tuition)) +  
  scale_fill_viridis_c(option = "magma")
```





# Quick aside

---

- Think about the data you have
- Given that these are state–level data, they have a geographic component

```
#install.packages("maps")
state_data <- map_data("state") %>% # ggplot2::map_data
  rename(State = region)
```

# Join it

---

Obviously we'll talk more about joins later

```
tuition <- tuition %>%
  mutate(State = tolower(State))
states <- left_join(state_data, tuition)
head(states)
```

```
##           long      lat group order    State subregion 2004-05 2005-06
## 1 -87.46201 30.38968     1     1 alabama       <NA> 5682.838 5840.55
## 2 -87.48493 30.37249     1     2 alabama       <NA> 5682.838 5840.55
## 3 -87.52503 30.37249     1     3 alabama       <NA> 5682.838 5840.55
## 4 -87.53076 30.33239     1     4 alabama       <NA> 5682.838 5840.55
## 5 -87.57087 30.32665     1     5 alabama       <NA> 5682.838 5840.55
## 6 -87.58806 30.32665     1     6 alabama       <NA> 5682.838 5840.55
##           2006-07 2007-08 2008-09 2009-10 2010-11 2011-12 2012-13
## 1 5753.496 6008.169 6475.092 7188.954 8071.134 8451.902 9098.069
## 2 5753.496 6008.169 6475.092 7188.954 8071.134 8451.902 9098.069
## 3 5753.496 6008.169 6475.092 7188.954 8071.134 8451.902 9098.069
## 4 5753.496 6008.169 6475.092 7188.954 8071.134 8451.902 9098.069
## 5 5753.496 6008.169 6475.092 7188.954 8071.134 8451.902 9098.069
## 6 5753.496 6008.169 6475.092 7188.954 8071.134 8451.902 9098.069
##           2013-14 2014-15 2015-16
## 1 9358.929 9496.084 9751.101
## 2 9358.929 9496.084 9751.101
```

# Rearrange

---

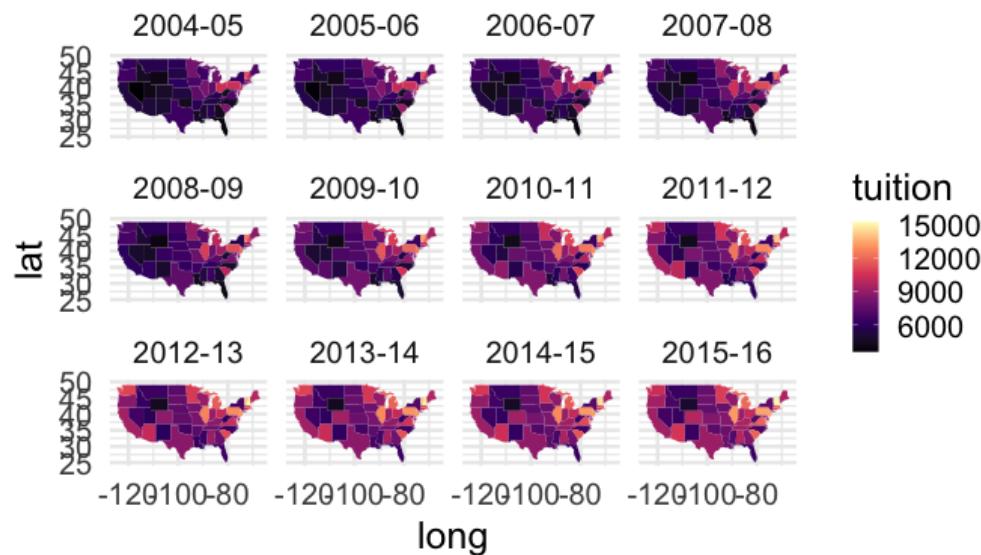
```
states <- states %>%
  gather(year, tuition, `2004-05`:`2015-16`)
head(states)
```

```
##      long      lat group order    State subregion     year tuition
## 1 -87.46201 30.38968     1     1 alabama      <NA> 2004-05 5682.838
## 2 -87.48493 30.37249     1     2 alabama      <NA> 2004-05 5682.838
## 3 -87.52503 30.37249     1     3 alabama      <NA> 2004-05 5682.838
## 4 -87.53076 30.33239     1     4 alabama      <NA> 2004-05 5682.838
## 5 -87.57087 30.32665     1     5 alabama      <NA> 2004-05 5682.838
## 6 -87.58806 30.32665     1     6 alabama      <NA> 2004-05 5682.838
```

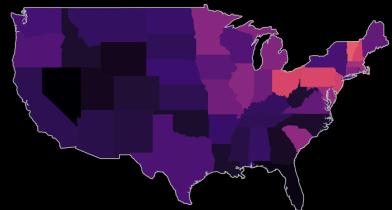
# Plot

---

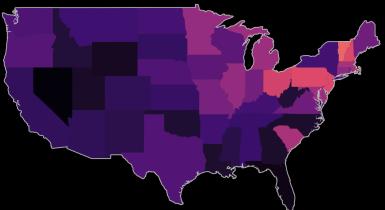
```
ggplot(states) +  
  geom_polygon(aes(long, lat, group = group, fill = tuition)) +  
  coord_fixed(1.3) +  
  scale_fill_viridis_c(option = "magma") +  
  facet_wrap(~year)
```



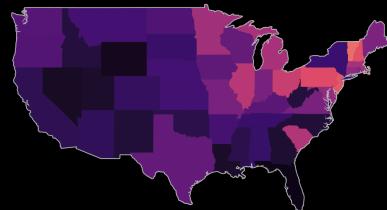
2004-05



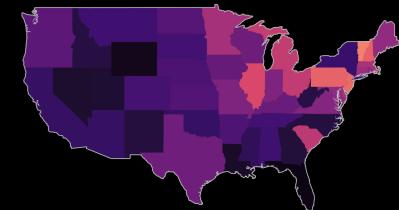
2005-06



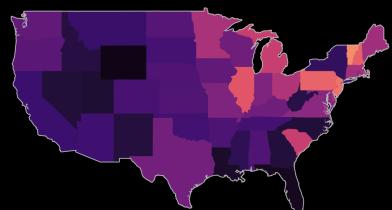
2006-07



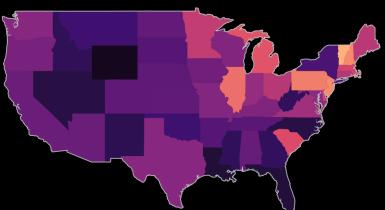
2007-08



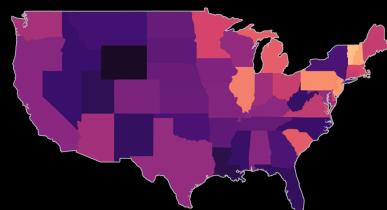
2008-09



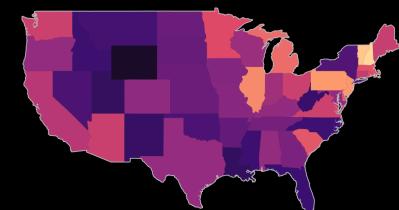
2009-10



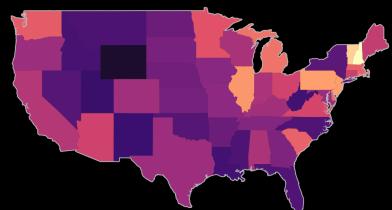
2010-11



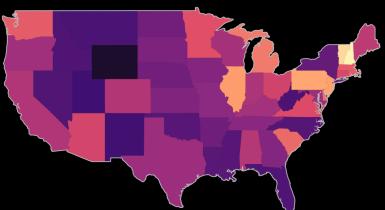
2011-12



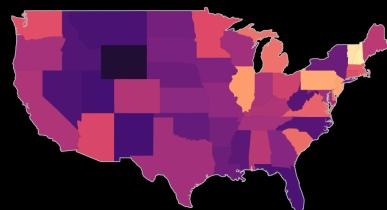
2012-13



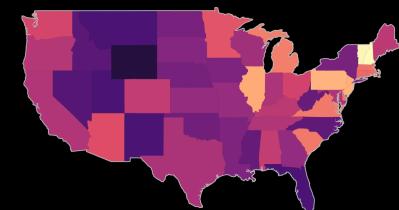
2013-14



2014-15



2015-16



# Intro to textual data

---

## Disclaimer:

This is going to be relatively fast, with not much depth. It's an intro, and I can work with you more individually if you end up working with text data.

# Structured vs unstructured

---

- Most every dataset you've ever worked with is what is referred to as a **structured** dataset – it has rows and columns.
- But there is an incredible amount of data out there that is **unstructured** – it just sort of exists
- Most text data is unstructured. How would you analyze the contents of a book? No rows or columns there

# Getting text data

---

There are **many** ways to get text data. Any digital text could potentially be used as textual data.

How about Wikipedia?

Anything that lives on the web is a common use case. Social media data being perhaps primary among them.

# "Screen" scraping

---

Short foray into web scraping. It's not expected you fully follow this. More about "exposure" and less about building competencies.

Use the `rvest` package to scrape the data you see "on the screen".

Let's read in the Wikipedia page on Eugene

```
library(rvest)
eugene <- read_html("https://en.wikipedia.org/wiki/Eugene%2C_Oreg
```



WIKIPEDIA  
The Free Encyclopedia

Main page  
Contents  
Current events  
Random article  
About Wikipedia  
Contact us  
Donate

Contribute  
Help  
Learn to edit  
Community portal  
Recent changes  
Upload file

Tools  
What links here  
Related changes  
Special pages  
Permanent link  
Page information  
Cite this page  
Wikidata item

Print/export  
Download as PDF  
Printable version

In other projects

Article Talk

Read Edit View history

Search Wikipedia



Not logged in Talk Contributions Create account Log in

# Eugene, Oregon

From Wikipedia, the free encyclopedia

Coordinates: 44°03'07"N 123°05'12"W

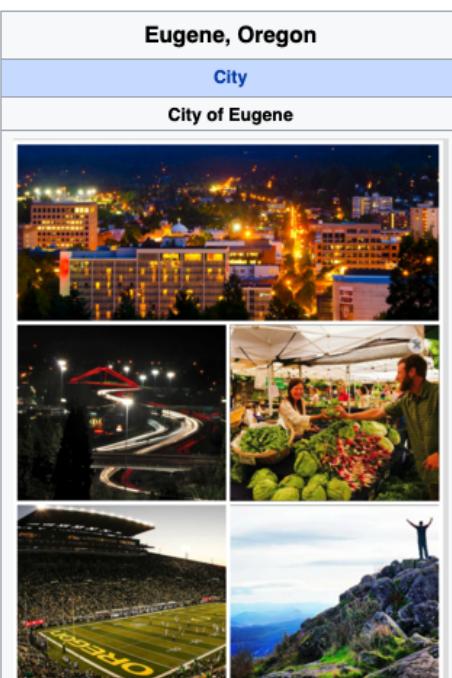
**Eugene** (/juːdʒɪn/ *yoo-JEEN*) is a city in the U.S. state of Oregon, in the Pacific Northwest. It is at the southern end of the Willamette Valley, near the confluence of the McKenzie and Willamette rivers, about 50 miles (80 km) east of the Oregon Coast.<sup>[7]</sup>

As of the 2010 census, Eugene had a population of 156,185; it is the county seat of Lane County and the state's third most populous city after Portland and Salem, though recent state estimates suggest its population may have surpassed Salem's.<sup>[8][9]</sup> The Eugene-Springfield, Oregon metropolitan statistical area (MSA) is the 146th largest metropolitan statistical area in the US and the third-largest in the state, behind the Portland Metropolitan Area and the Salem Metropolitan Area.<sup>[10]</sup> The city's population for 2019 was estimated to be 172,622 by the U.S. Census.<sup>[11]</sup>

Eugene is home to the University of Oregon, Bushnell University, and Lane Community College.<sup>[12][13][14]</sup> The city is noted for its natural environment, recreational opportunities (especially bicycling, running/jogging, rafting, and kayaking), and focus on the arts, along with its history of civil unrest, protests, and green activism. Eugene's official slogan is "A Great City for the Arts and Outdoors".<sup>[15]</sup> It is also referred to as the "Emerald City" and as "Track Town, USA".<sup>[16]</sup> The Nike corporation had its beginnings in Eugene.<sup>[17]</sup> In 2022, the city will host the 18th Track and Field World Championships.<sup>[18]</sup>

## Contents [hide]

- 1 History
  - 1.1 Indigenous presence
  - 1.2 Settlement and impact
  - 1.3 Educational institutions
  - 1.4 Twentieth century
  - 1.5 History of civil unrest
    - 1.5.1 1960s: Counterculture and campus protests



Clockwise: Downtown Eugene from Skinner Butte, Lane County Farmers' Market, Hiking on Spencer Butte, University of Oregon Autzen Stadium, Delta Ponds pedestrian bridge

# Grab paragraphs

---

The "#mw-content-text > div.mw-parser-output > p" is the CSS selector that I pulled from the website

```
paragraphs <- eugene %>%  
  html_elements("#mw-content-text > div.mw-parser-output > p") %>  
  html_text2()
```

The first paragraph is just an empty line, so they are numbered p + 1

Print the first paragraph

```
cat(stringr::str_wrap(paragraphs[2], 50))
```

```
## Eugene (/ju:'dʒi:n/ yoo-JEEN) is a city in the  
## U.S. state of Oregon, in the Pacific Northwest. It  
## is at the southern end of the Willamette Valley,  
## near the confluence of the McKenzie and Willamette  
## rivers, about 50 miles (80 km) east of the Oregon
```

# Print the fourth paragraph

---

```
cat(stringr::str_wrap(paragraphs[5], 50))
```

```
## The first people to settle in the Eugene area were
## known as the Kalapuyans, also written Calapooia
## or Calapooya. They made "seasonal rounds," moving
## around the countryside to collect and preserve
## local foods, including acorns, the bulbs of the
## wapato and camas plants, and berries. They stored
## these foods in their permanent winter village.
## When crop activities waned, they returned to their
## winter villages and took up hunting, fishing, and
## trading.[19][20] They were known as the Chifin
## Kalapuyans and called the Eugene area where they
## lived "Chifin", sometimes recorded as "Chafin" or
## "Chiffin".[21][22]
```

# Analysis

---

How do we analyze the text? What we we even analyze?

First, let's structure it! Turn the text into a simple data frame.

```
library(tidyverse)
eugene_df <- tibble(
  paragraph = seq_along(paragraphs),
  description = paragraphs
)
eugene_df
```

```
## # A tibble: 149 × 2
##       paragraph
##           <int>
## 1             1
## 2             2
## 3             3
## 4             4
## 5             5
## 6             6
## 7             7
## 8             8
## 9             9
```

# Can we analyze it now?

---

Not really... what would we analyze?

Words!

Let's break it into words. This is where the `tidytext` package comes into play.

# The `unnest_tokens()` function

---

Just like most functions in the tidyverse, we pipe our data to `unnest_tokens()`

- First argument is the name of the new column we want in our data
- Second argument is the text data to process
- Third argument is how the text should processed. The default is "`words`", meaning the text will be broken into words.

# Example

---

```
library(tidytext)
eugene_tidy_words <- eugene_df %>%
  unnest_tokens(word, description)
eugene_tidy_words
```

```
## # A tibble: 9,620 × 2
##       paragraph word
##           <int> <chr>
## 1             2 eugene
## 2             2 ju:'dʒi:n
## 3             2 yoo
## 4             2 jeen
## 5             2 is
## 6             2 a
## 7             2 city
## 8             2 in
## 9             2 the
## 10            2 u.s
## # ... with 9,610 more rows
```

Not perfect, but pretty good

# What to do now?

---

Let's count some words!

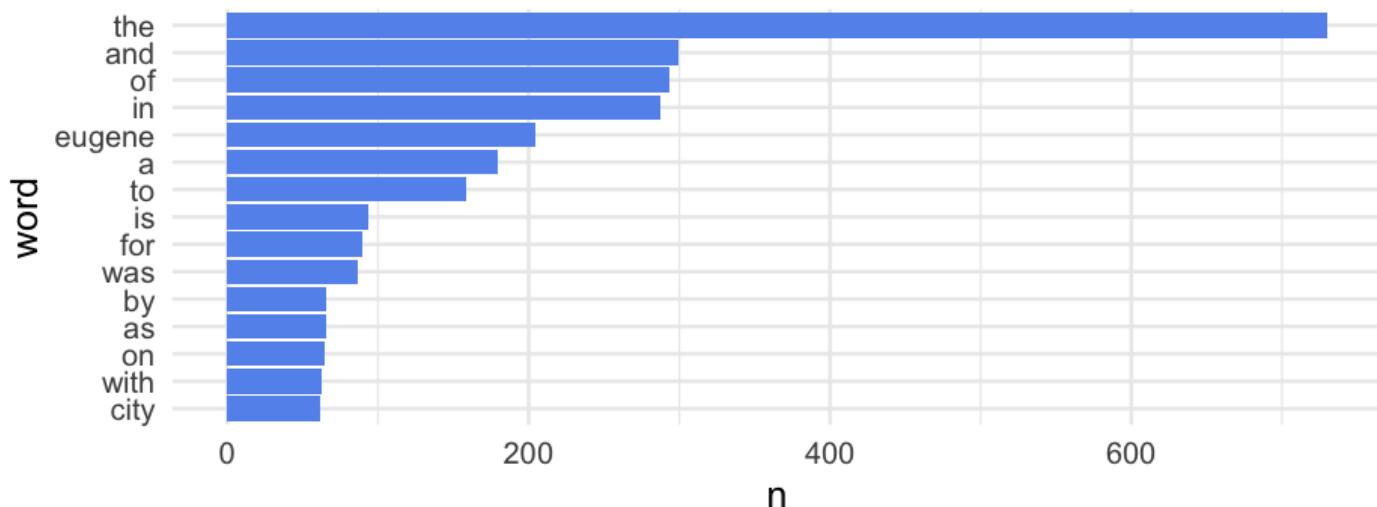
```
eugene_tidy_words %>%  
  count(word, sort = TRUE)
```

```
## # A tibble: 2,875 × 2  
##   word      n  
##   <chr>    <int>  
## 1 the      730  
## 2 and      299  
## 3 of       293  
## 4 in       287  
## 5 eugene   204  
## 6 a        180  
## 7 to       159  
## 8 is        94  
## 9 for      90  
## 10 was     87  
## # ... with 2,865 more rows
```

# Plot the top 15 words

---

```
eugene_tidy_words %>%
  count(word, sort = TRUE) %>%
  mutate(word = fct_reorder(word, n)) %>%
  slice(1:15) # select only the first 15 rows
ggplot(aes(n, word)) +
  geom_col(fill = "cornflowerblue")
```



# Not very informative

---

Why?

Most of the words are common words like "the", "and", "of"  
(top three words)

These are referred to as "stop words".

Luckily, **tidytext** provides us with a dictionary of stop words.  
We can use an `anti_join()` with this dictionary to remove  
these words.

# Quick refresher

---

A `semi_join()` works just like an `inner_join()`, but without adding any columns. A `semi_join()` works by **keeping** only rows that are in common with the two datasets.

An `anti_join()` does basically the opposite, by **removing** any rows that are in common between the two datasets.

# Look at the stop words

---

This dataset is available to you as soon as you load **tidytext**.

There are three lexicons – I usually use all three.

## stop\_words

```
## # A tibble: 1,149 × 2
##   word      lexicon
##   <chr>     <chr>
## 1 a        SMART
## 2 a's      SMART
## 3 able     SMART
## 4 about    SMART
## 5 above    SMART
## 6 according SMART
## 7 accordingly SMART
## 8 across   SMART
## 9 actually SMART
## 10 after   SMART
## # ... with 1,139 more rows
```

# Count

---

Let's try counting again without the stop words included.

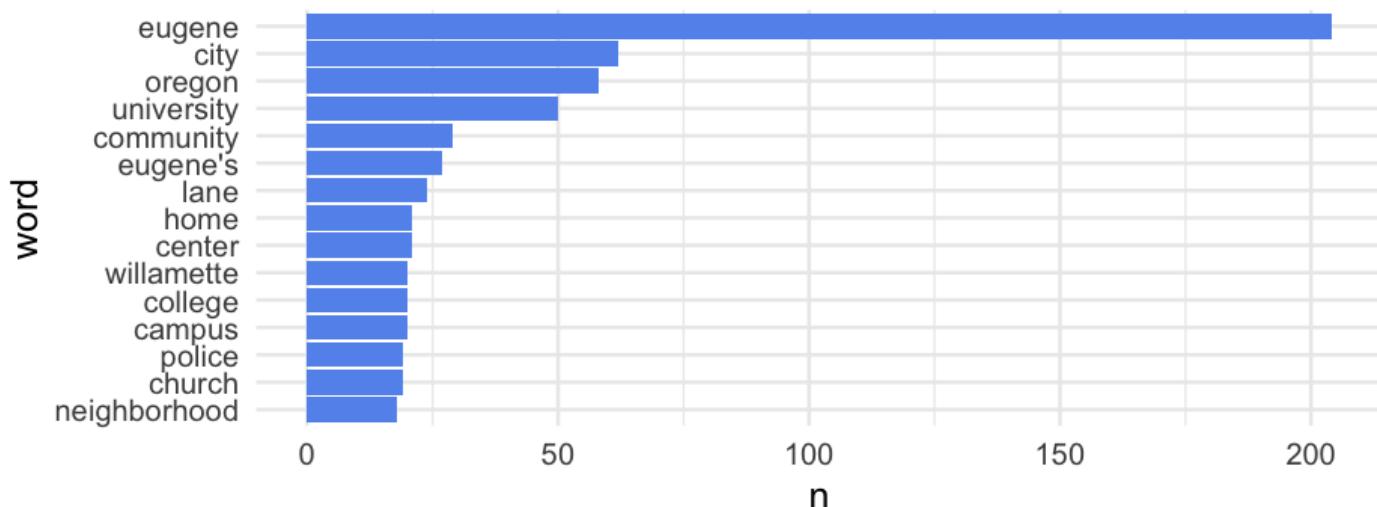
```
eugene_tidy_words %>%  
  anti_join(stop_words) %>%  
  count(word, sort = TRUE)
```

```
## # A tibble: 2,581 × 2  
##   word          n  
##   <chr>     <int>  
## 1 eugene      204  
## 2 city        62  
## 3 oregon      58  
## 4 university  50  
## 5 community   29  
## 6 eugene's    27  
## 7 lane        24  
## 8 center      21  
## 9 home        21  
## 10 campus     20  
## # ... with 2,571 more rows
```

So much more informative!

# Plot the top 15 words

```
eugene_tidy_words %>%
  anti_join(stop_words) %>%
  count(word, sort = TRUE) %>%
  mutate(word = reorder(word, n)) %>% # make y-axis ordered by n
  slice(1:15) %>% # select only the first 15 rows
  ggplot(aes(n, word)) +
  geom_col(fill = "cornflowerblue")
```



# Add the headers in

---

I hid the code here because it's weird and inefficient but the HTML structure made it difficult. You can look at the source if you want. The new dataframe is called

[eugene\\_tidy\\_words2](#).

```
## # A tibble: 149 × 3
##       paragraph header
##   <int> <chr>
## 1     1 Intro
## 2     2 Intro
## 3     3 Intro
## 4     4 History
## 5     5 History
## 6     6 History
## 7     7 History
## 8     8 History
## 9     9 History
## 10    10 History
## # ... with 139 more rows, and 1 more variable: description <chr>
```

# Count words by header

---

Not surprisingly, "eugene" appears to be the most common among multiple categories.

We might want to remove "eugene" as well.

```
eugene_tidy_words2 %>%
  unnest_tokens(word, description) %>%
  count(header, word, sort = TRUE) %>%
  anti_join(stop_words)
```

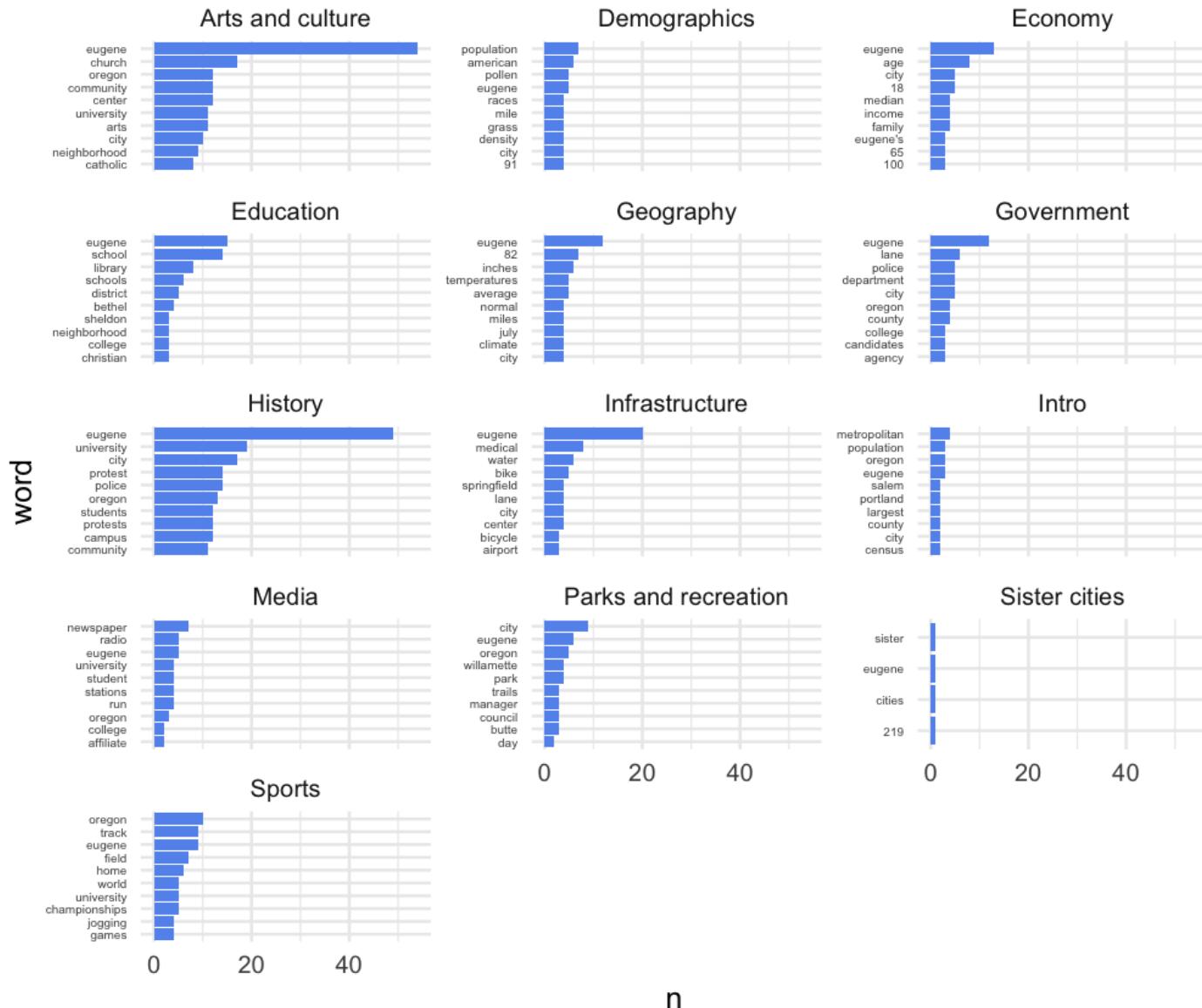
```
## # A tibble: 3,584 × 3
##   header      word     n
##   <chr>      <chr> <int>
## 1 Arts and culture eugene    54
## 2 History       eugene    49
## 3 Infrastructure eugene    20
## 4 History       university 19
## 5 Arts and culture church   17
## 6 History       city      17
## 7 Education     eugene    15
## 8 Education     school    14
## 9 History       police    14
```

# Plot

---

Top 10 words by header

```
p <- eugene_tidy_words2 %>%
  unnest_tokens(word, description) %>%
  count(header, word, sort = TRUE) %>%
  anti_join(stop_words) %>%
  group_by(header) %>%
  slice(1:10) %>%
  mutate(word = reorder_within(word, n, header)) %>%
  ggplot(aes(n, word)) +
  geom_col(fill = "cornflowerblue") +
  scale_y_reordered() +
  facet_wrap(~header, scales = "free_y", ncol = 3)
```



# String manipulations

---

# Base vs tidyverse

---

The tidyverse package is `{stringr}`

It is more consistent than base functions and occasionally faster

However, I tend to prefer the base functions, and they are still more commonly seen "in the wild" than **stringr**.

We'll therefore briefly cover both.

# Inconsistencies

---

Super common example

```
set.seed(123)
ex <- tibble(
  gender = sample(
    c("f", "F", "Fem", "Female", "FEMALE",
      "m", "M", "Male", "MALE",
      "nb", "NB", "non-binary",
      "agender", "AGENDER", "Agender",
      "gender-fluid", "fluid",
      "No response"),
    500,
    replace = TRUE),
  score = rnorm(500)
)
```

# Have vs want

---

Have

```
## # A tibble: 18 × 2
##   gender      n
##   <chr>     <int>
## 1 agender     25
## 2 Agender     26
## 3 AGENDER     29
## 4 f            24
## 5 F            24
## 6 Fem          27
## 7 Female        27
## 8 FEMALE       25
## 9 fluid         28
## 10 gender-fluid 28
## 11 m            29
## 12 M            34
## 13 Male          37
## 14 MALE         27
## 15 nb            36
## 16 NB            27
## 17 No response  21
## 18 non-binary    26
```

Want

```
## # A tibble: 6 × 2
##   gender      n
##   <chr>     <int>
## 1 agender     80
## 2 female       127
## 3 fluid         56
## 4 male          127
## 5 no response  21
## 6 non-binary    89
```

# Walkthrough

---

Getting to what we want takes a few steps. Let's do it together!

# Consistent case

---

- The first thing we might want to do is change everything to uppercase or lowercase. This will fix many of our inconsistencies.
- Options are `stringr::str_to_upper()`, `stringr::str_to_lower()`, `base::toupper()` or `base::tolower()`

# Consistent case

---

## Original

```
ex %>%  
  count(gender)
```

```
## # A tibble: 18 × 2  
##   gender      n  
##   <chr>     <int>  
## 1 agender     25  
## 2 Agender     26  
## 3 AGENDER     29  
## 4 f            24  
## 5 F            24  
## 6 Fem          27  
## 7 Female       27  
## 8 FEMALE       25  
## 9 fluid         28  
## 10 gender-fluid 28  
## 11 m            29  
## 12 M            34  
## 13 Male         37  
## 14 MALE         27  
## 15 nb           36
```

## Modified

```
ex %>%  
  mutate(gender = tolower(gender))  
  count(gender)
```

```
## # A tibble: 11 × 2  
##   gender      n  
##   <chr>     <int>  
## 1 agender     80  
## 2 f            48  
## 3 fem          27  
## 4 female       52  
## 5 fluid         28  
## 6 gender-fluid 28  
## 7 m            63  
## 8 male         64  
## 9 nb           63  
## 10 no response 21  
## 11 non-binary   26
```

# What next?

---

Collapse the genders that have "fluid"?

Use `grepl()` (global regular expression parser `logical`) with `ifelse()` to replace *if* a pattern is found

You could also use `stringr::str_detect()` instead. The arguments are just in reversed order

```
grepl("fluid", gender)
str_detect(gender, "fluid")
```

```
ex %>%  
  mutate(  
    gender = tolower(gender),  
    gender = ifelse(grepl("fluid", gender), "gender-fluid", gender)  
  ) %>%  
  count(gender)
```

```
## # A tibble: 10 × 2  
##   gender     n  
##   <chr>     <int>  
## 1 agender     80  
## 2 f            48  
## 3 fem          27  
## 4 female       52  
## 5 gender-fluid 56  
## 6 m            63  
## 7 male         64  
## 8 nb           63  
## 9 no response  21  
## 10 non-binary  26
```

# stringr

---

```
library(stringr)
ex %>%
  mutate(
    gender = tolower(gender),
    gender = ifelse(str_detect(gender, "fluid"), "gender-fluid",
  ) %>%
  count(gender)
```

```
## # A tibble: 10 × 2
##   gender      n
##   <chr>     <int>
## 1 agender     80
## 2 f           48
## 3 fem          27
## 4 female       52
## 5 gender-fluid 56
## 6 m           63
## 7 male         64
## 8 nb           63
## 9 no response 21
## 10 non-binary 26
```

# What next?

---

How about – if it starts with "m", then "Male"?

Use "**"^"** to denote "starts with"

```
ex %>%  
  mutate(  
    gender = tolower(gender),  
    gender = ifelse(grepl("fluid", gender), "gender-fluid", gender),  
    gender = ifelse(grepl("^m", gender), "male", gender)  
) %>%  
  count(gender)
```

```
## # A tibble: 9 × 2  
##   gender      n  
##   <chr>     <int>  
## 1 agender     80  
## 2 f           48  
## 3 fem          27  
## 4 female       52  
## 5 gender-fluid  56  
## 6 male         127  
## 7 nb            63  
## 8 no response   21  
## 9 non-binary    26
```

# Again

Replicate the same thing, but this time with "female". Note that we couldn't do this initially, but we can now because "**fluid**" has been collapsed with "**gender-fluid**".

You try first

```
ex %>%  
  mutate(  
    gender = tolower(gender),  
    gender = ifelse(grepl("fluid", gender), "gender-fluid", gender),  
    gender = ifelse(grepl("^m", gender), "male", gender),  
    gender = ifelse(grepl("^f", gender), "female", gender)  
) %>%  
  count(gender)
```

```
## # A tibble: 7 × 2  
##   gender      n  
##   <chr>     <int>  
## 1 agender     80  
## 2 female      127  
## 3 gender-fluid  56  
## 4 male        127  
## 5 nb          63  
## 6 no response  21  
## 7 non-binary   26
```

# Again?

---

Can we do the same thing with "`"^n"` for non-binary?

NO!

Little bit more complicated – use Boolean logic.

```
ex %>%  
  mutate(  
    gender = tolower(gender),  
    gender = ifelse(grepl("fluid", gender), "gender-fluid", gender),  
    gender = ifelse(grepl("^m", gender), "male", gender),  
    gender = ifelse(grepl("^f", gender), "female", gender),  
    gender = ifelse(  
      grepl("^n", gender) & gender != "no response",  
      "non-binary",  
      gender  
    )  
  ) %>%  
  count(gender)
```

```
## # A tibble: 6 × 2
##   gender      n
##   <chr>     <int>
## 1 agender     80
## 2 female      127
## 3 gender-fluid 56
## 4 male        127
## 5 no response 21
## 6 non-binary   89
```

# stringr version

---

```
ex %>%
  mutate(
    gender = tolower(gender),
    gender = ifelse(str_detect(gender, "fluid"), "gender-fluid",
    gender = ifelse(str_detect(gender, "^m"), "male", gender),
    gender = ifelse(str_detect(gender, "^f"), "female", gender),
    gender = ifelse(
      str_detect(gender, "^n") & gender != "no response",
      "non-binary",
      gender
    )
  ) %>%
  count(gender)
```

# Special characters

---

- `^`: Anchor – matches the start of a string
- `$`: Anchor – matches the end of a string
- `*`: Matches the preceding character **zero or more** times
- `?`: Matches the preceding character **zero or one** times
- `+`: Matches the preceding character **one or more** times
- `{`: Used to specify number of matches, `a{n}`, `a{n,}`, and `a{n, m}`
- `.`: Wildcard – matches any character
- `|`: OR operator
- `[`: Alternates, also used for character matching (e.g., `[:digit:]`)
- `(`: Used for backreferencing, look aheads, or groups
- `\`: Used to escape special characters

# More detail

---

Both of the below are good places to get more comprehensive information

- RStudio cheatsheet
- Regular expression vignette

# One more quick example

---

```
library(edld652)
d <- get_data("EDFacts_acgr_lea_2011_2019")
```

```
## | 0%
| |
| |
| = | 1%
| = | 1%
| |
| == | 2%
| |
| == | 3%
| |
| == | 4%
| === | 4%
| |
| === | 5%
| |
| ===== | 6%
| |
| ===== | 7%
```

# Do three things:

---

- Create a new variable that identifies if the LEA is associated with a city or a county
- Drop "City" or "County" from the LEA name (e.g., "Albertville City" would be "Albertville")
- Replace all `.` with **--DOT--** in **FILEURL** (so they are not actual links)

# City or county

---

Ideas?

```
d <- d %>%
  mutate(county = grepl("county$", tolower(LEANM)))
```

# Quick Check

---

```
d %>%
  select(LEANM, county) %>%
  print(n = 15)
```

```
## # A tibble: 11,326 × 2
##   LEANM           county
##   <chr>          <lgl>
## 1 Albertville City FALSE
## 2 Marshall County TRUE
## 3 Hoover City    FALSE
## 4 Madison City   FALSE
## 5 Leeds City     FALSE
## 6 Boaz City      FALSE
## 7 Trussville City FALSE
## 8 Alexander City FALSE
## 9 Andalusia City FALSE
## 10 Anniston City FALSE
## 11 Arab City     FALSE
## 12 Athens City   FALSE
## 13 Attalla City  FALSE
## 14 Auburn City   FALSE
## 15 Autauga County TRUE
## # ... with 11,311 more rows
```

# Remove city/county

---

- Lots of ways to do this. Use `base::gsub()` or `stringr::str_replace_all()`
- Replace everything after the space with nothing

```
d %>%
  select(LEANM) %>%
  mutate(new_name = gsub(" .+", "", LEANM))
```

```
## # A tibble: 11,326 × 2
##   LEANM           new_name
##   <chr>          <chr>
## 1 Albertville City Albertville
## 2 Marshall County Marshall
## 3 Hoover City    Hoover
## 4 Madison City   Madison
## 5 Leeds City     Leeds
## 6 Boaz City      Boaz
## 7 Trussville City Trussville
## 8 Alexander City Alexander
## 9 Andalusia City Andalusia
## 10 Anniston City Anniston
```

# Another way

---

There are other ways too, of course

```
d %>%
  select(LEANM) %>%
  mutate(new_name = gsub(" City$| County$", "", LEANM))
```

```
## # A tibble: 11,326 × 2
##   LEANM      new_name
##   <chr>      <chr>
## 1 Albertville City Albertville
## 2 Marshall County Marshall
## 3 Hoover City    Hoover
## 4 Madison City   Madison
## 5 Leeds City     Leeds
## 6 Boaz City      Boaz
## 7 Trussville City Trussville
## 8 Alexander City Alexander
## 9 Andalusia City Andalusia
## 10 Anniston City Anniston
## # ... with 11,316 more rows
```

# Final step

---

Handling the URLs. This is a bit artificial, but it illustrates escaping, which is important.

- Remember `.` is a special character, so needs to be escaped
- `\` itself is a special character so it needs to be escaped. Functionally, then, you escape special characters with `\\\`, not `\`.

```
d %>%  
  select(FILEURL)
```

```
## # A tibble: 11,326 × 1  
## # ... with 11,316 more rows, and 1 more variable: FILEURL <chr>
```

```
d %>%  
  select(FILEURL) %>%  
  mutate(FILEURL = gsub("\\.", "--DOT--", FILEURL)) %>%  
  as.data.frame()
```

```
##  
## 1 https://www2--DOT--ed--DOT--gov/about/inites/ed/edfacts/data-files/a  
## 2 https://www2--DOT--ed--DOT--gov/about/inites/ed/edfacts/data-files/a  
## 3 https://www2--DOT--ed--DOT--gov/about/inites/ed/edfacts/data-files/a  
## 4 https://www2--DOT--ed--DOT--gov/about/inites/ed/edfacts/data-files/a  
## 5 https://www2--DOT--ed--DOT--gov/about/inites/ed/edfacts/data-files/a  
## 6 https://www2--DOT--ed--DOT--gov/about/inites/ed/edfacts/data-files/a  
## 7 https://www2--DOT--ed--DOT--gov/about/inites/ed/edfacts/data-files/a  
## 8 https://www2--DOT--ed--DOT--gov/about/inites/ed/edfacts/data-files/a  
## 9 https://www2--DOT--ed--DOT--gov/about/inites/ed/edfacts/data-files/a  
## 10 https://www2--DOT--ed--DOT--gov/about/inites/ed/edfacts/data-files/a  
## 11 https://www2--DOT--ed--DOT--gov/about/inites/ed/edfacts/data-files/a  
## 12 https://www2--DOT--ed--DOT--gov/about/inites/ed/edfacts/data-files/a  
## 13 https://www2--DOT--ed--DOT--gov/about/inites/ed/edfacts/data-files/a  
## 14 https://www2--DOT--ed--DOT--gov/about/inites/ed/edfacts/data-files/a  
## 15 https://www2--DOT--ed--DOT--gov/about/inites/ed/edfacts/data-files/a  
## 16 https://www2--DOT--ed--DOT--gov/about/inites/ed/edfacts/data-files/a  
## 17 https://www2--DOT--ed--DOT--gov/about/inites/ed/edfacts/data-files/a  
## 18 https://www2--DOT--ed--DOT--gov/about/inites/ed/edfacts/data-files/a  
## 19 https://www2--DOT--ed--DOT--gov/about/inites/ed/edfacts/data-files/a  
## 20 https://www2--DOT--ed--DOT--gov/about/inites/ed/edfacts/data-files/a  
## 21 https://www2--DOT--ed--DOT--gov/about/inites/ed/edfacts/data-files/a  
## 22 https://www2--DOT--ed--DOT--gov/about/inites/ed/edfacts/data-files/a  
## 23 https://www2--DOT--ed--DOT--gov/about/inites/ed/edfacts/data-files/a
```

# Wrapping up

This was pretty quick, still a lot we didn't get to

I'll try to embed more opportunities for you to practice these skills throughout the term

# Next time

---

No class – MLK Jr. Day. Please do have your lab in before then.

# Lab 1

---