Week 3: Data Visualization

{ggplot2}

Joe Nese

University of Oregon

Fall 2021

Data Visualization with

Week 3

Agenda

Finalize Groups

`{ggplot2}`

- syntax
- continuous data visualizations
- categorical data visualizations
- options
  - color/fill
  - transparency
  - labels
  - facets

Learning Objectives

- Understand the basic syntax requirements for `{ggplot2}`
- Recognize various options for displaying continuous and categorical data
- Familiarity with various `{ggplot2}` options
  - color/fill
  - transparency
  - labels
  - facets

Share!

{datapasta}

- Copy and paste data to and from R
- VERY handy!
- Good for reprex
    - posting questions on Rstudio Community or stackoverflow

[demo]

The Bus!
THEY SAID MY CAR'S TOTALED. I DISAGREE.
1107

TAXI

tidyverse

Providing grammar for:

- Graphics
  - `{ggplot2}`
- Data manipulations
  - `{dplyr}`
  - `{tidyr}`
- Expanding area of specialized topics
  - `{lubridate}`
  - `{glue}`
  - `{tidymodels}`
- Many more...

Providing grammar for:

- Graphics
    - `{ggplot2}`
- Data manipulations
    - `{dplyr}`
    - `{tidyr}`
- Expanding area of specialized topics
    - `{lubridate}`
    - `{glue}`
    - `{tidymodels}`
- Many more...

The {ggplot2} package

`gg` stands for the "grammar of graphics"

Resources

The `{ggplot2}` package is one of the most popular R packages, and there are many resources to learn the syntax

- ggplot2 book (email me for digital copy)
- RStudio cheat sheet
    - Can be helpful, perhaps more so after a little experience
- R Graphics Cookbook
- R Graph Gallery

Components

Every ggplot plot has three components

1. data
   - the `data` used to produce the plot
2. aesthetic mappings (`aes`)
   - between variables and visual properties
3. layer(s)
   - usually through the `geom_*()` function to produce geometric shapes to be rendered

`{ggplot}` always takes a data frame (tibble) as the first argument

Basic syntax

```
ggplot(data, aes(x = xvar, y = yvar ) ) +
        geom_function()
```

ggplot() = the function is `ggplot` and the package is `{ggplot2}`

data = the data to be plotted

aes(x = xvar, y = yvar) = the `aes`thetic mappings

geom_function() = the `geom`etrics of the plot; the "function" here represents any of the geom offerings

+ = note the + and <u>NOT</u> the %>%

{ggplot2} template

```
ggplot("data", aes("mappings")) +
         "geom_function"()
```

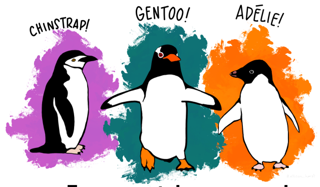
or

```
"data" %>%
  ggplot(aes("mappings")) +
  "geom_function"()
```

Some data for today

penguins from {palmerpenguins}

[run the following]

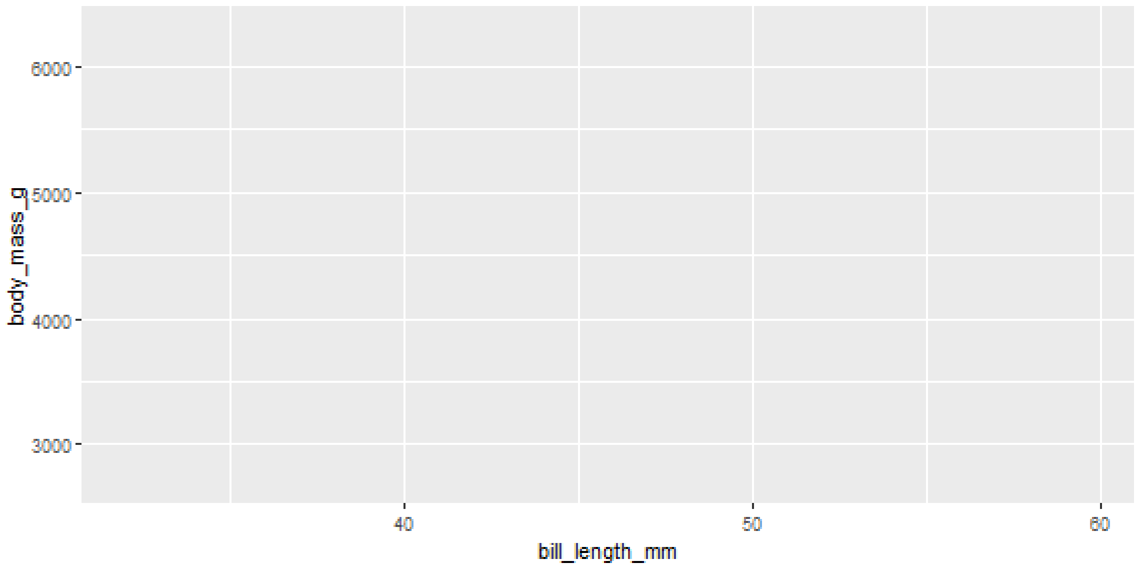


```
## # A tibble: 6 x 8
##   species island    bill_length_mm bill_depth_mm flipper_length_mm body_mas
##   <fct>   <fct>              <dbl>         <dbl>             <int>        <i
## 1 Adelie  Torgersen           39.1          18.7               181          3
## 2 Adelie  Torgersen           39.5          17.4               186          3
## 3 Adelie  Torgersen           40.3          18                 195          3
## 4 Adelie  Torgersen           NA            NA                  NA
## 5 Adelie  Torgersen           36.7          19.3               193          3
## 6 Adelie  Torgersen           39.3          20.6               190          3
```

Continuous Data

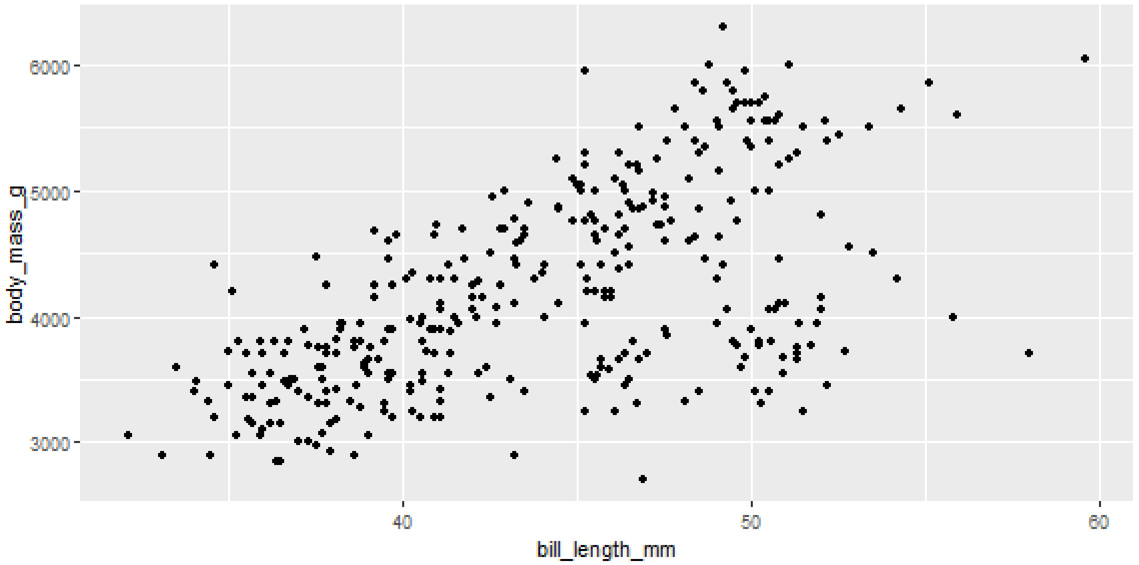
Setting up a plot

Run the following code. What do you see?



It's ready for you to add some geometric layers...what should we add?

How about points?



Adding layers

- In the previous slide, we added a layer of points
- The `geom_point()` layer is a function, complete with it's own arguments

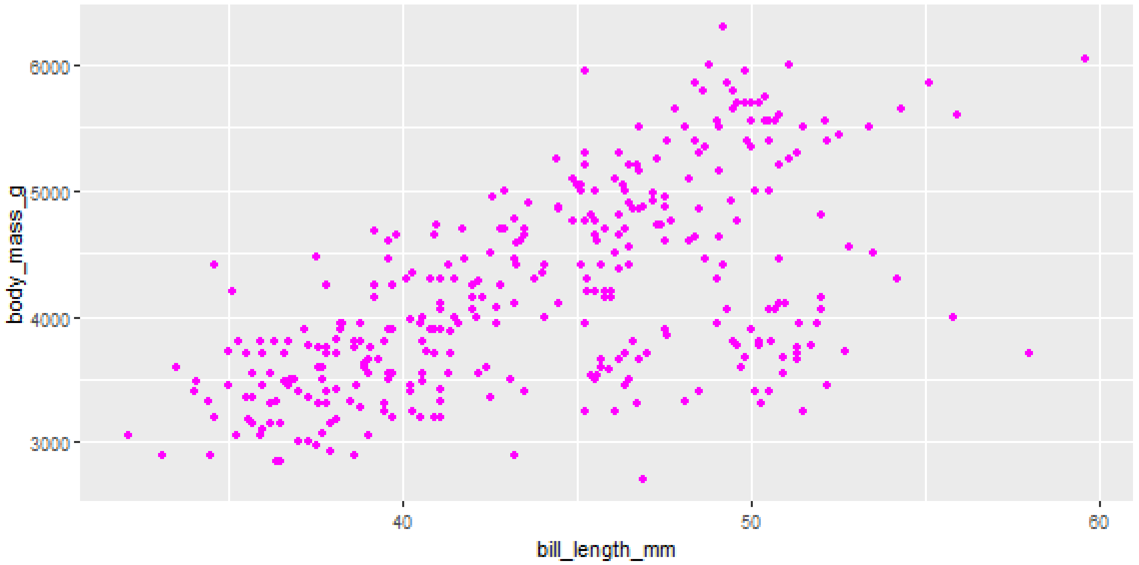
Let’s change the color of the points

How would you change the color of the points?

or

How would you find out about changing the color of the points?

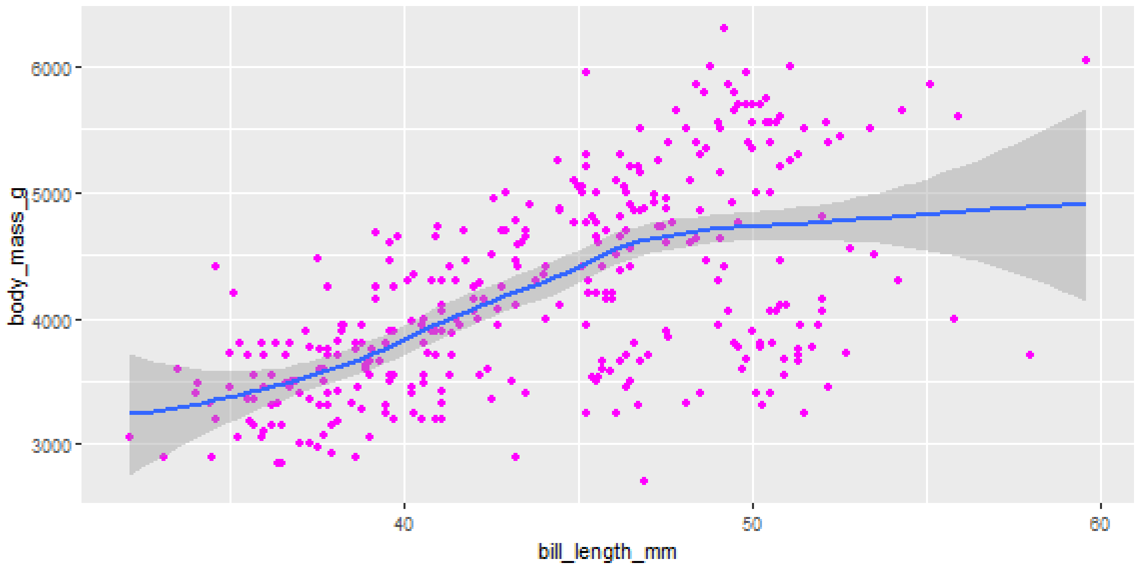
Color



]

Add another layer

Let's add a smoothed line with `geom_smooth()`



You try

You probably got the **message** below when you ran (defaults)

```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```

Change the `method` to `"lm"`

Let's do this together

Look at the help page – `?geom_smooth`

1. Remove the confidence interval around the line

2. Now change the *SE* band to reflect a 68% confidence interval

`color`: global vs. conditional

Prior examples changed colors globally

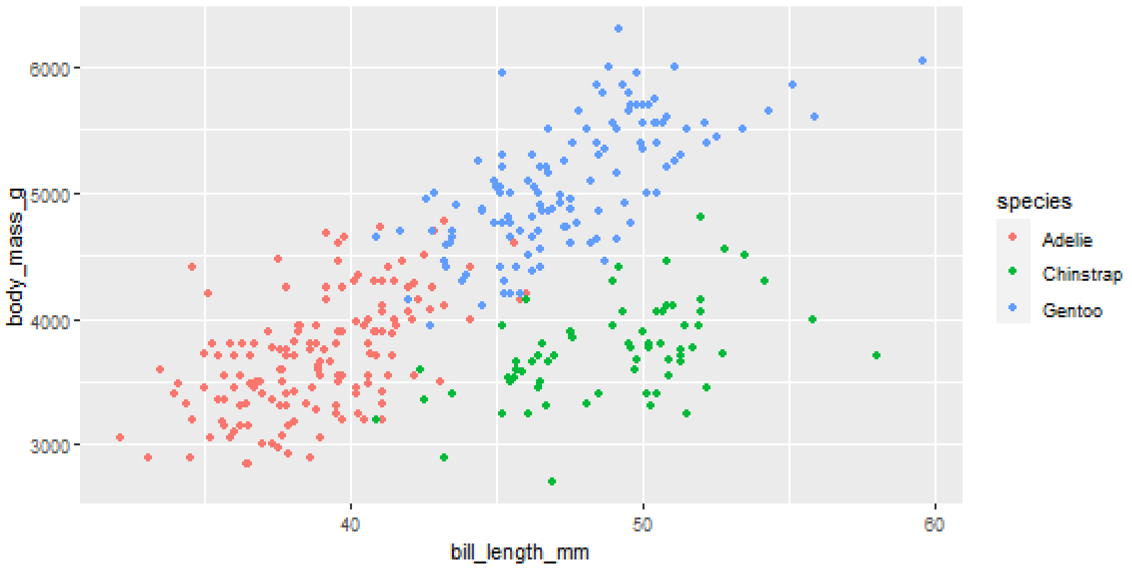
- `geom_point(color = "magenta")`

Use `aes()` to access variables, and color according to a specific variable

- We use variable names within `aes()`

Let’s check the data again (`head()`) and the "species" variable (`table()`)

[let's do this together]


body_mass_g
6000
5000
4000
3000
40
50
60
bill_length_mm
species
Adelie
Chinstrap
Gentoo

`color`: global vs. conditional

- When we did `geom_point(color = "magenta")` we put quotes around the color
- Why now is "species" not in quotes?
  - color names/hex codes are in quotes **NOT** in the `aes()`
  - variable names are in the `aes()` **NOT** in quotes
  - `aes()` is where you map to your data!

Conditional flow through layers

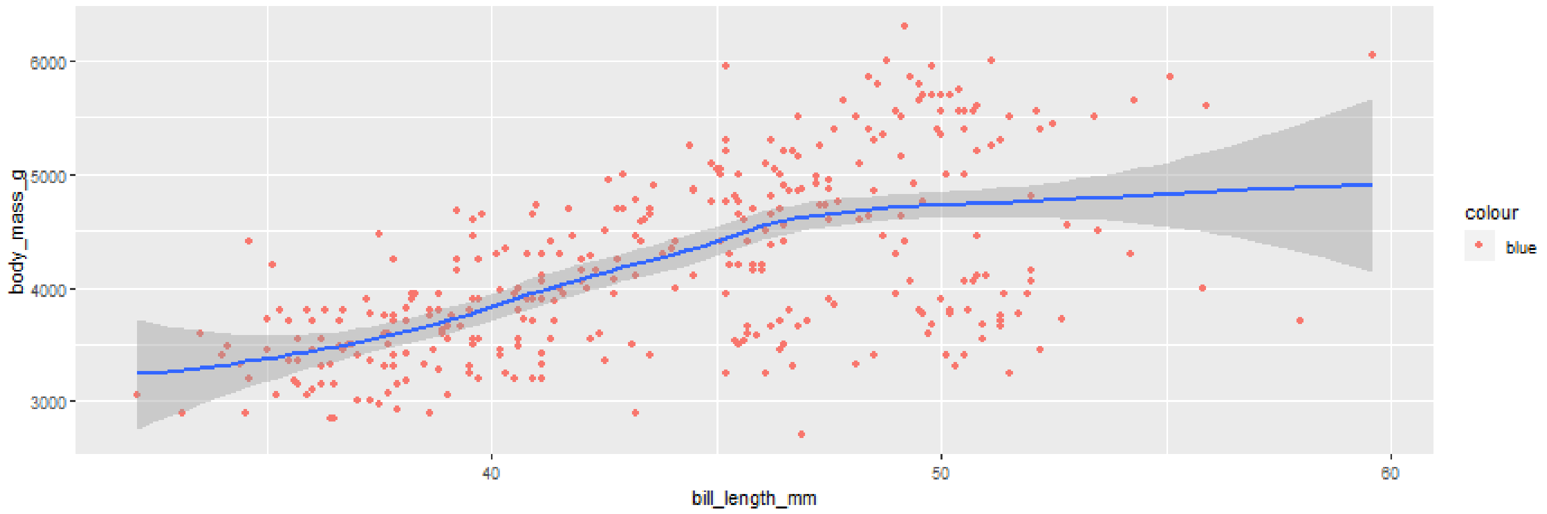
If we use something like `color = “x”` in the first `aes`thetic, it will carry on through all additional layers

- These two codes are the same:
- But these two are not…why? [run to find out]

Be mindful with `aes()`

Using `aes()` when you **don't** need it

What is happening here?



Be mindful with `aes()`

Not using `aes()` when you need it

What is happening here?

```
## Error in layer(data = data, mapping = mapping, stat = stat, geom = GeomPoin
```

Kind of helpful message here.

Themes

Let's talk themes

- The default is `theme_gray`
    - I don't like it
- But there are a lot of build-in alternative in `{ggplot2}`
    - `theme_minimal` is my favorite
- Check out the `{ggthemes}` package for a lot of alternatives
    - These days I nearly always use the `colorblind` theme for discrete values in my plots
- Check out the `{ggthemeassist}` add-in

More themes

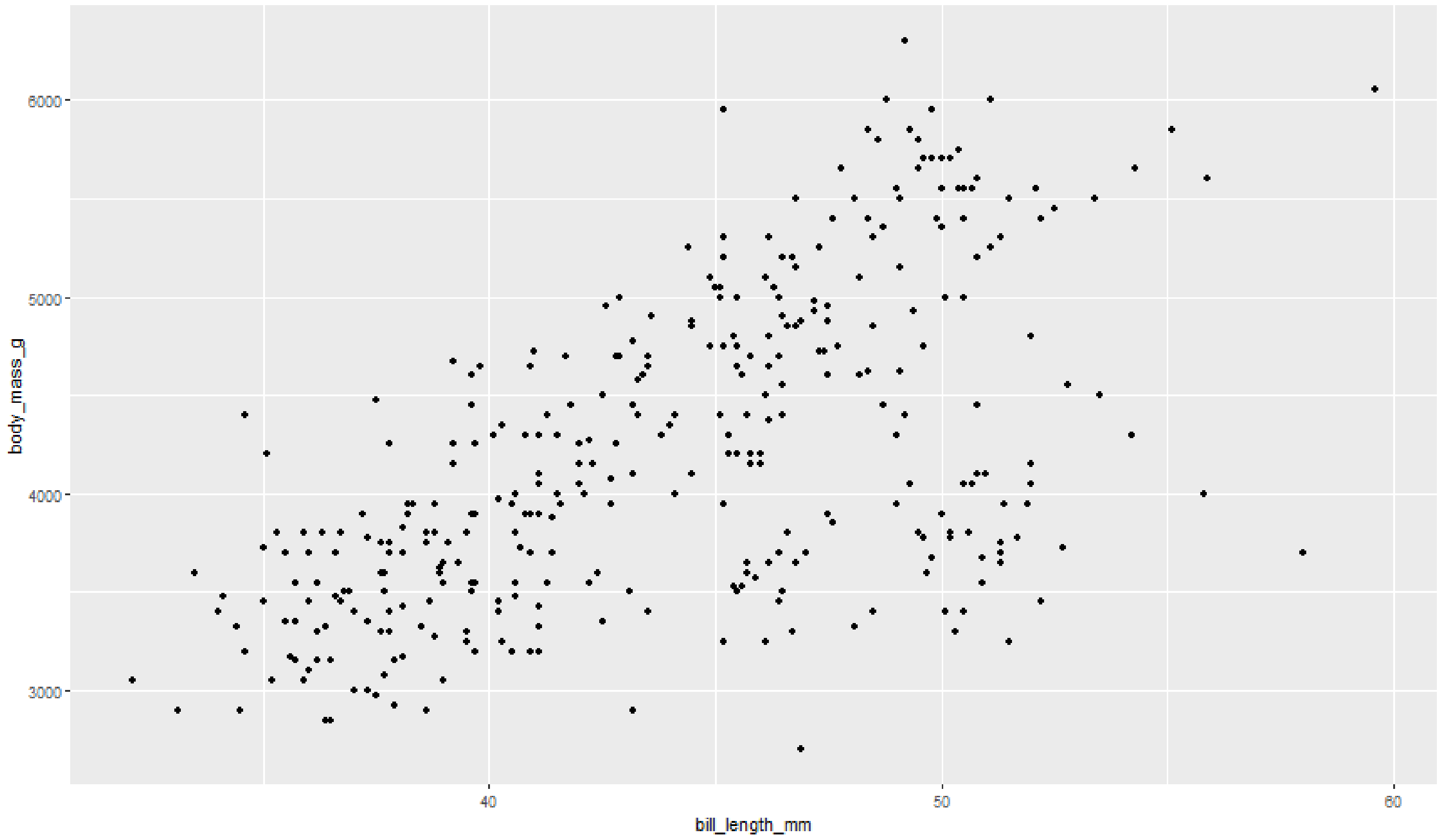
- The `{hrbrthemes}` are nice
- Consider building your own theme!
- Or Google around
- Set the theme globally
    - One of the first lines in your .Rmd file could be:

```
theme_set(theme_minimal())
```

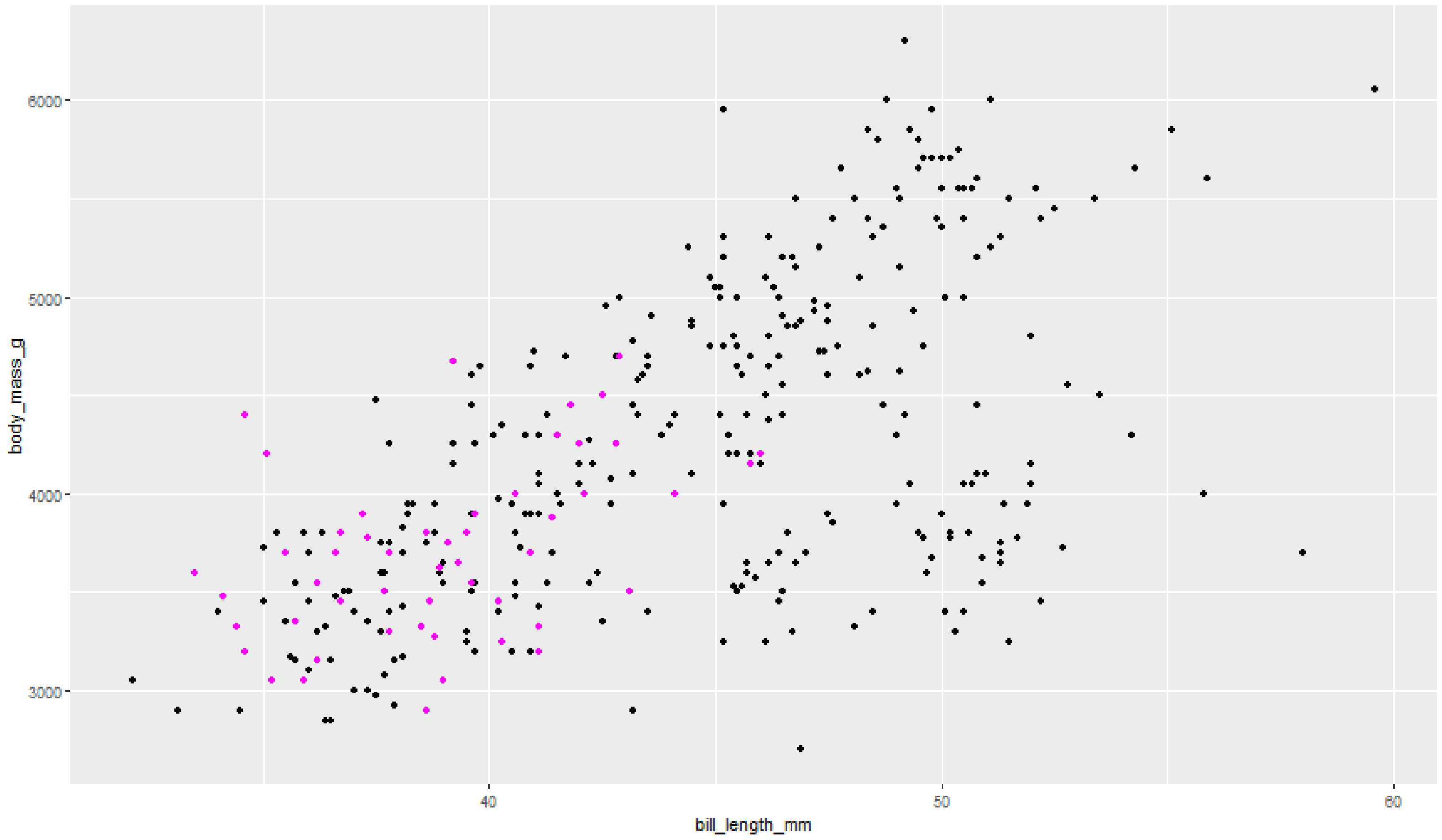
- I put this in the `setup` chunk

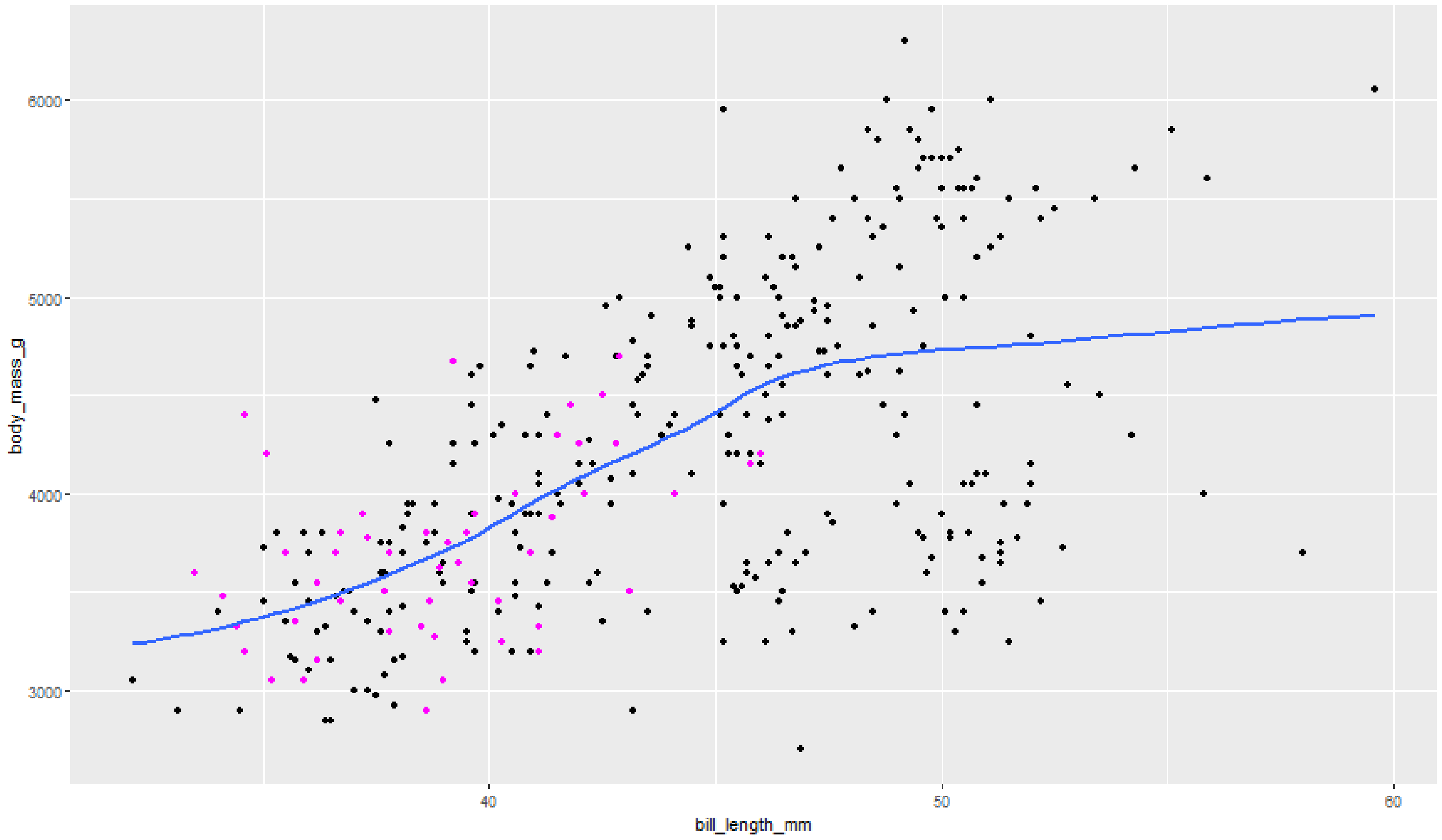
Get a little fancy

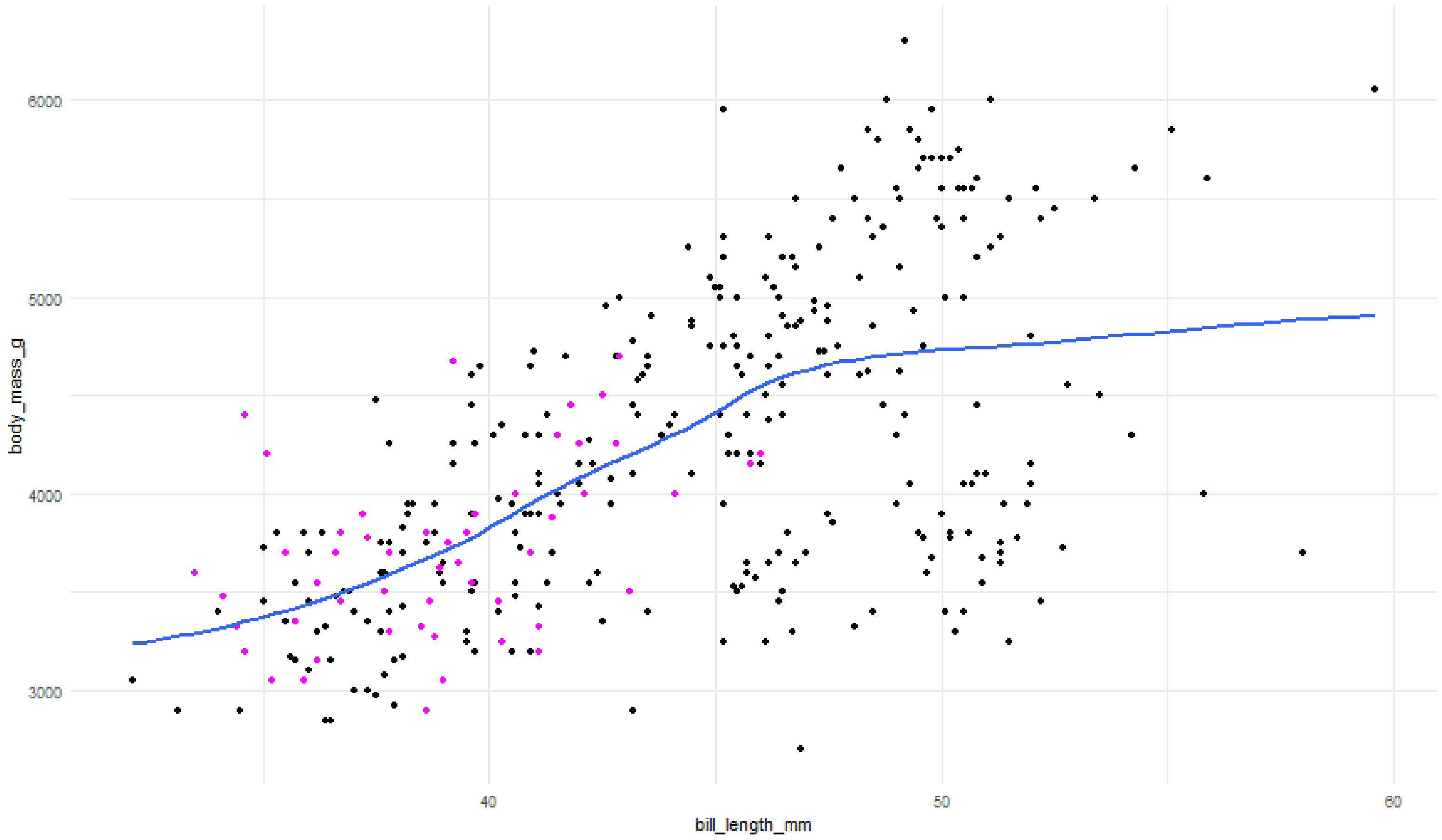
- You can use `geom_point()` for more than one layer
- You can also use a different data source on a layer
- Use these two properties to highlight points
    - How about penguins from Torgersen Island?



6000
5000
4000
3000
body_mass_g
40
50
60
bill_length_mm

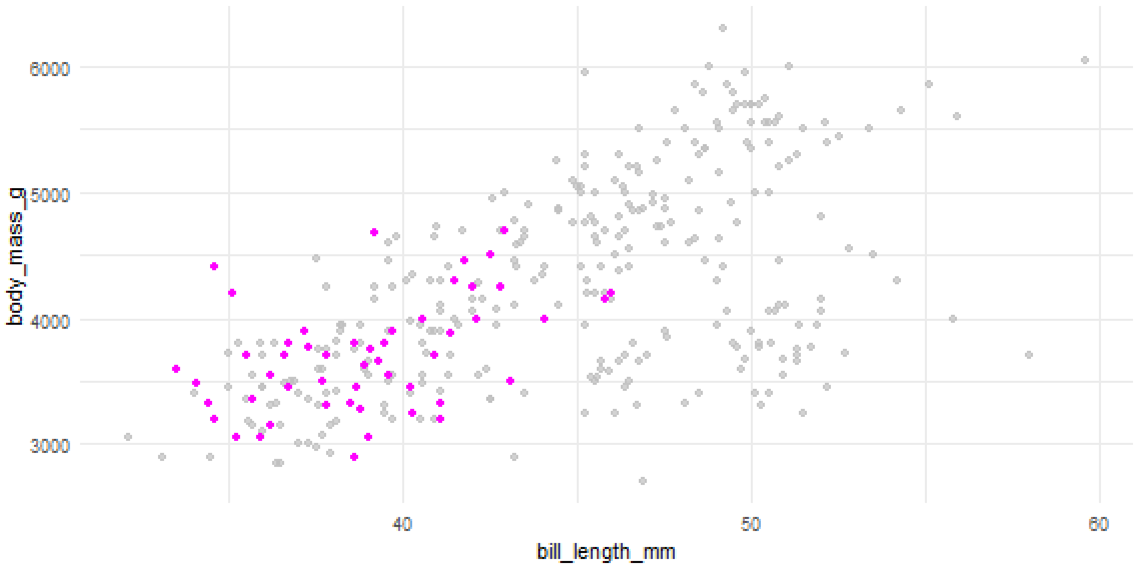

6000
5000
4000
3000
body_mass_g
40
50
60
bill_length_mm


6000
5000
4000
3000
body_mass_g
40
50
60
bill_length_mm


body_mass_g
6000
5000
4000
3000
40
50
60
bill_length_mm

Another option

`{gghighlight}` varying flexibility



Line plots

- When should you use line plots instead of smooths?
    - usually when time is involved
- What are some good candidate data for line plots?
    - observed versus model-implied (estimated)

`geom_line()`

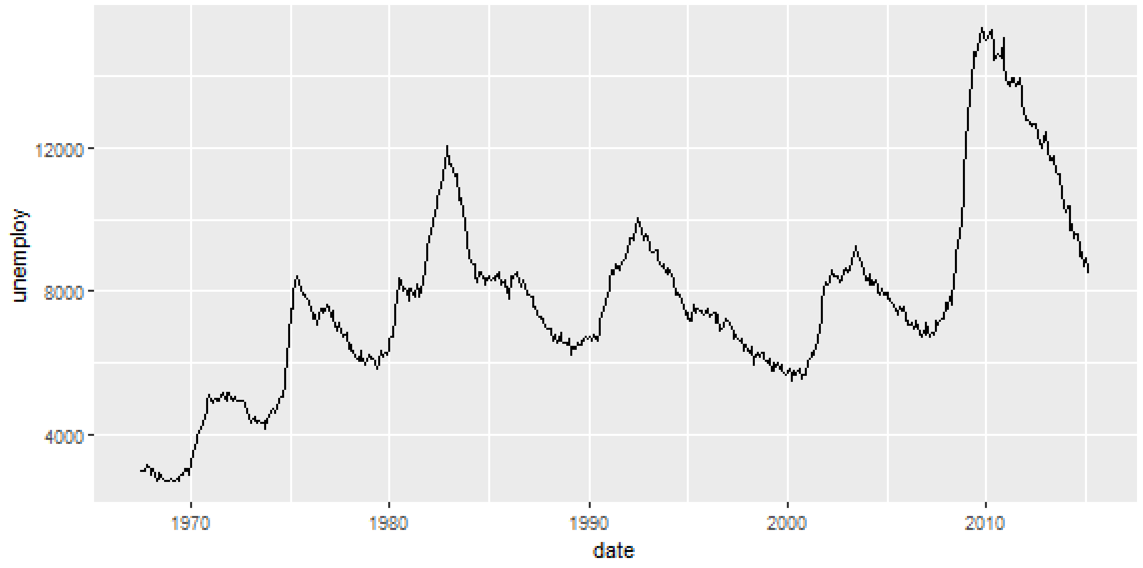
Classic time series example

economics data from `{ggplot2}`

```
## # A tibble: 574 x 6
##    date         pce    pop psavert uempmed unemploy
##    <date>     <dbl>  <dbl>   <dbl>   <dbl>    <dbl>
##  1 1967-07-01  507. 198712    12.6     4.5     2944
##  2 1967-08-01  510. 198911    12.6     4.7     2945
##  3 1967-09-01  516. 199113    11.9     4.6     2958
##  4 1967-10-01  512. 199311    12.9     4.9     3143
##  5 1967-11-01  517. 199498    12.8     4.7     3066
##  6 1967-12-01  525. 199657    11.8     4.8     3018
##  7 1968-01-01  531. 199808    11.7     5.1     2878
##  8 1968-02-01  534. 199920    12.3     4.5     3001
##  9 1968-03-01  544. 200056    11.7     4.1     2877
## 10 1968-04-01  544  200208    12.3     4.6     2709
## # ... with 564 more rows
```

Let's try it

How do you think we'd fit a line plot to these data, showing unemployment ("unemploy") over time?



Layers

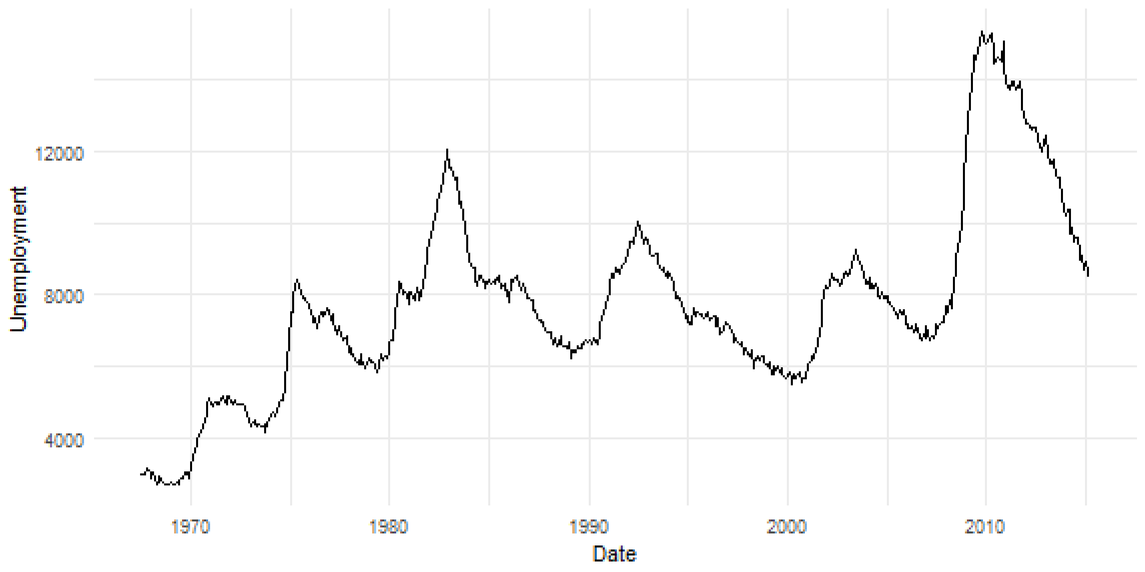
What happens when we layer geom_line and geom_point?

try it!

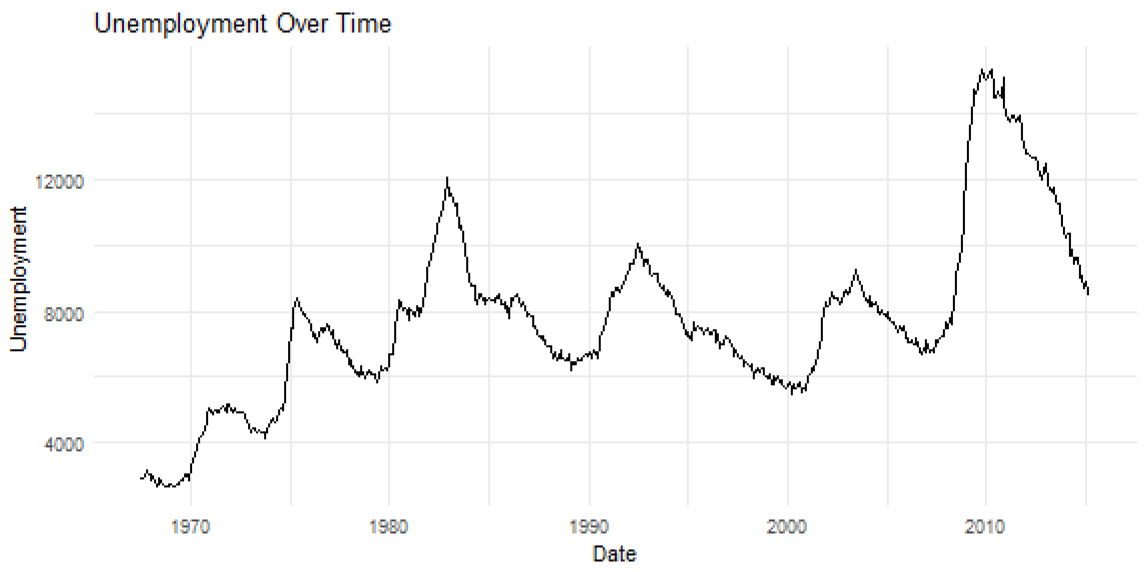
Not the best instance of this It would work better on a plot with fewer time points, but you get the idea

Labels

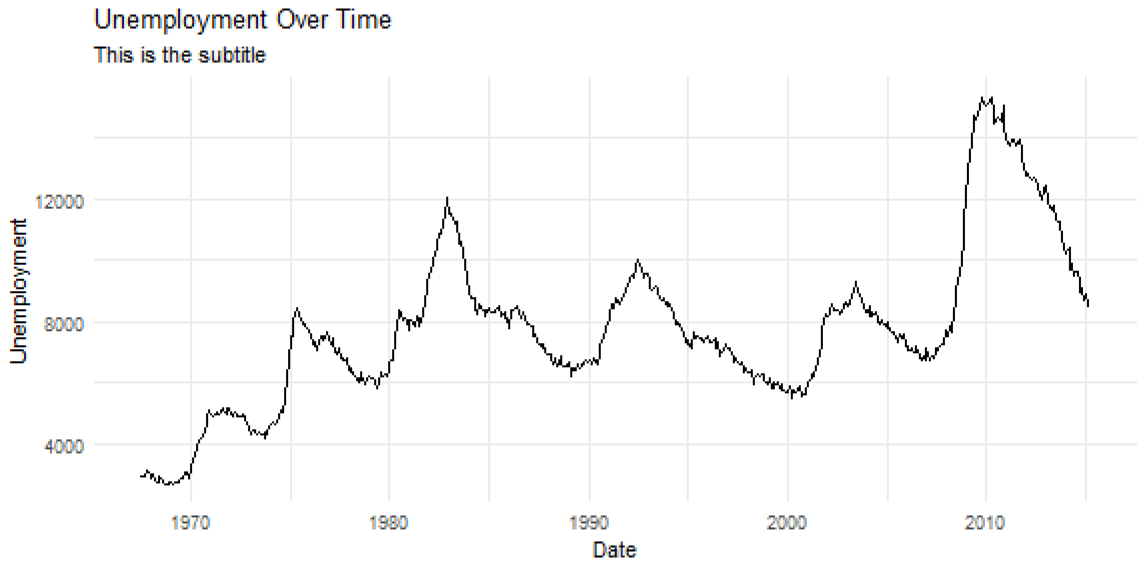
Axis Labels



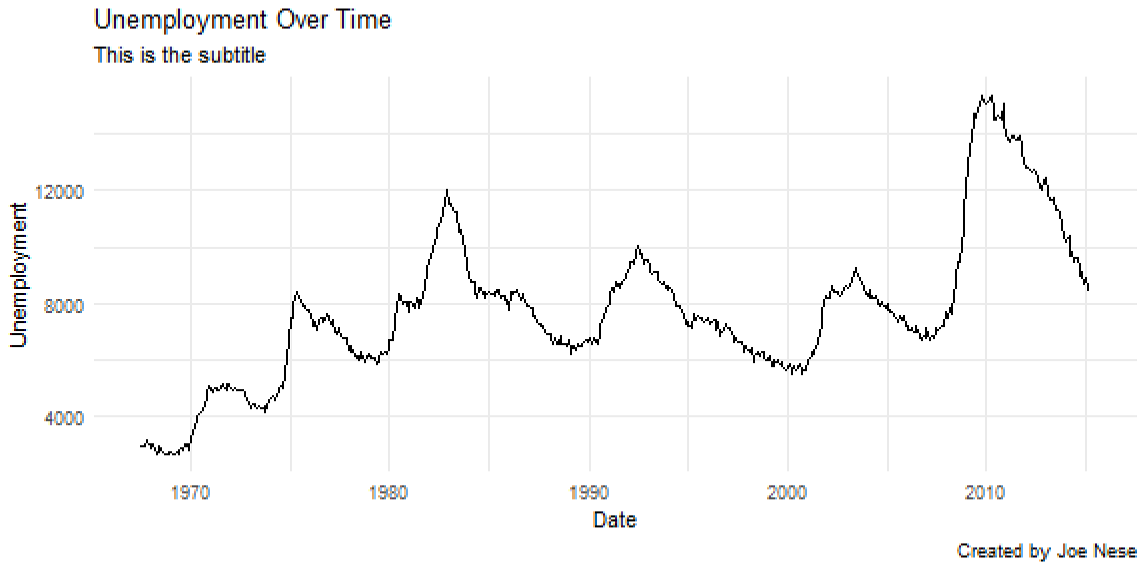
Title


Unemployment Over Time
Unemployment
Date
12000
8000
4000
1970
1980
1990
2000
2010

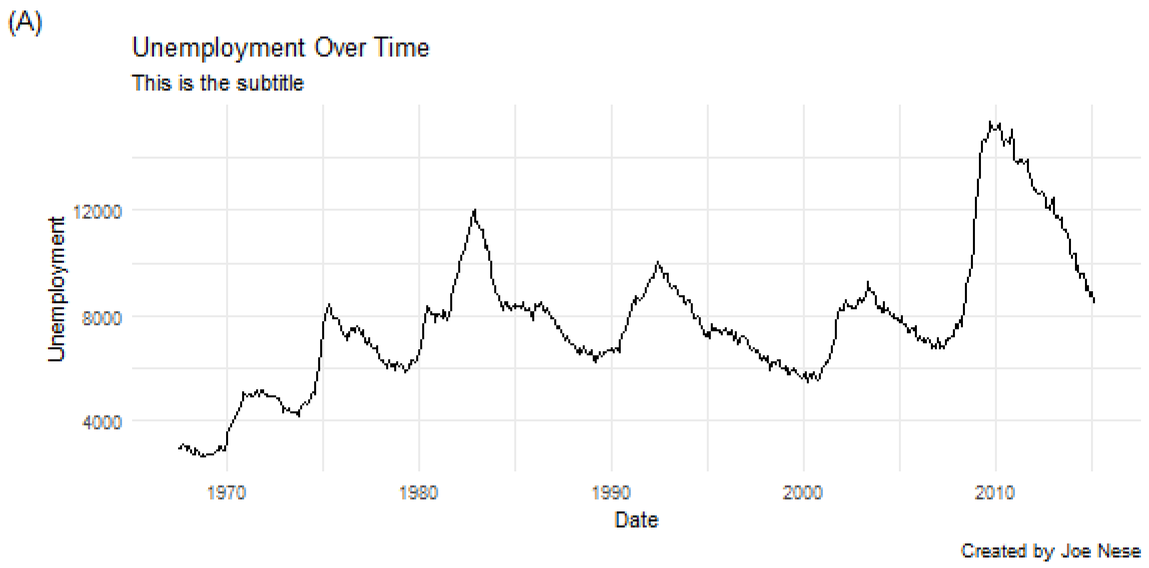
Subtitle


Unemployment Over Time
This is the subtitle
Unemployment
12000
8000
4000
1970
1980
1990
2000
2010
Date

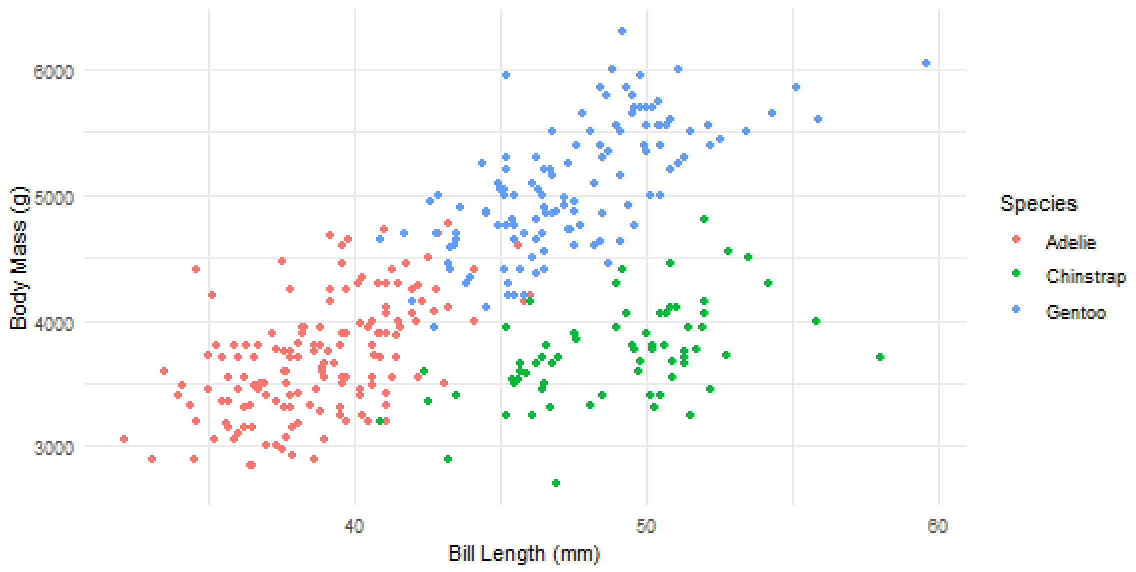
Caption


Unemployment Over Time
This is the subtitle
Unemployment
12000
8000
4000
1970
1980
1990
2000
2010
Date
Created by Joe Nese

Tag


(A)
Unemployment Over Time
This is the subtitle
Unemployment
12000
8000
4000
1970
1980
1990
2000
2010
Date
Created by Joe Nese

Legend (one way)

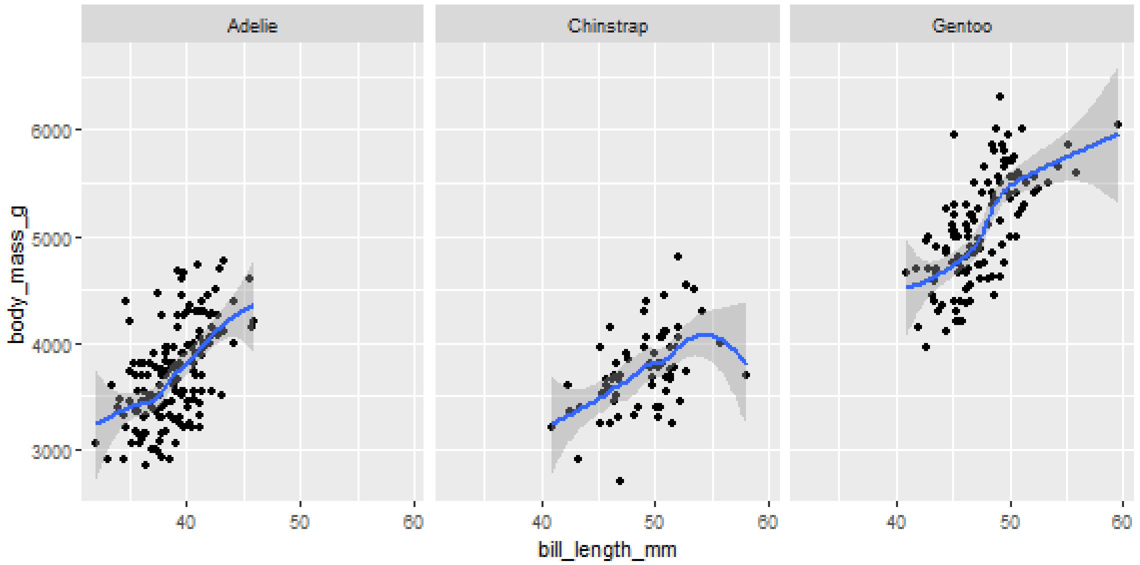


Facets

Faceting

- One of the most powerful features of `{ggplot}`
- Produce *n* plots by a specific variable
- `facet_wrap()`
    - wrap a sequence of panels into two dimensions
    - based on variables(s)

Faceting

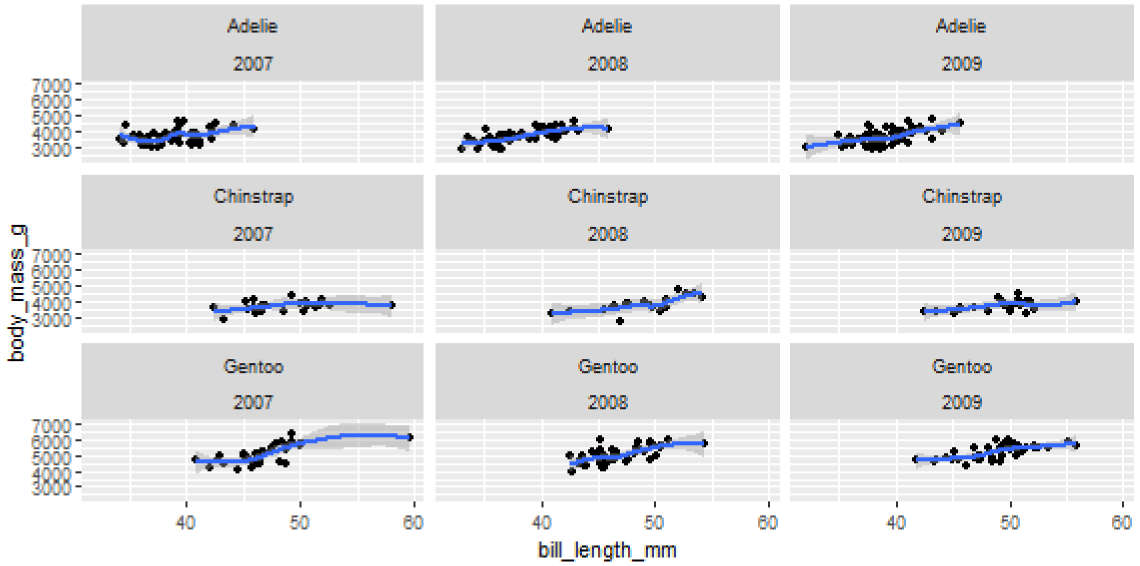

Adelie
Chinstrap
Gentoo
body_mass_g
6000
5000
4000
3000
40
50
60
bill_length_mm

Careful about the ~

```
## Error in validate_facets(x): object 'species' not found
```

Faceting

two variables (like a matrix)



Alternative specification (`vars()`)

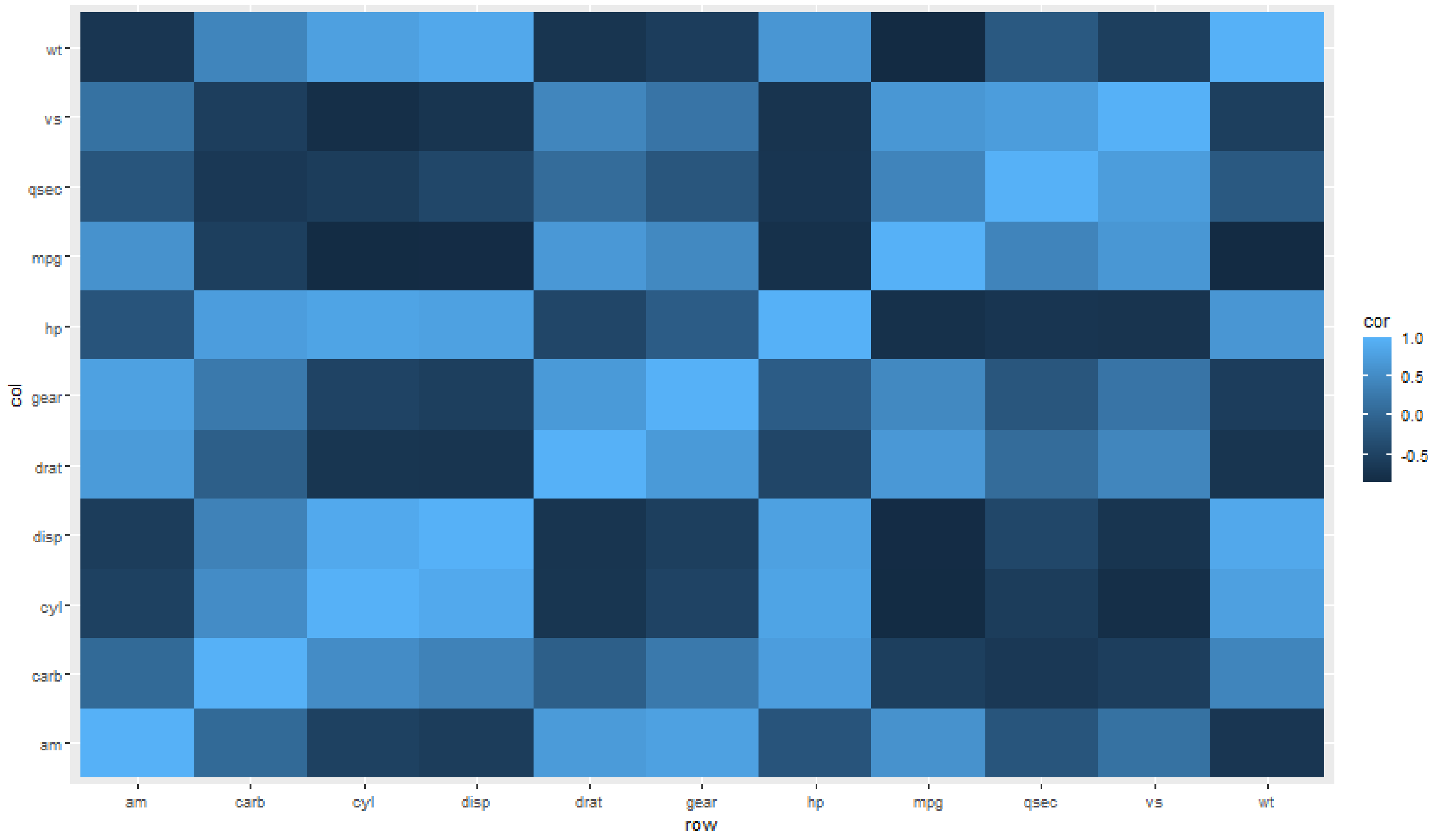
Heatmaps

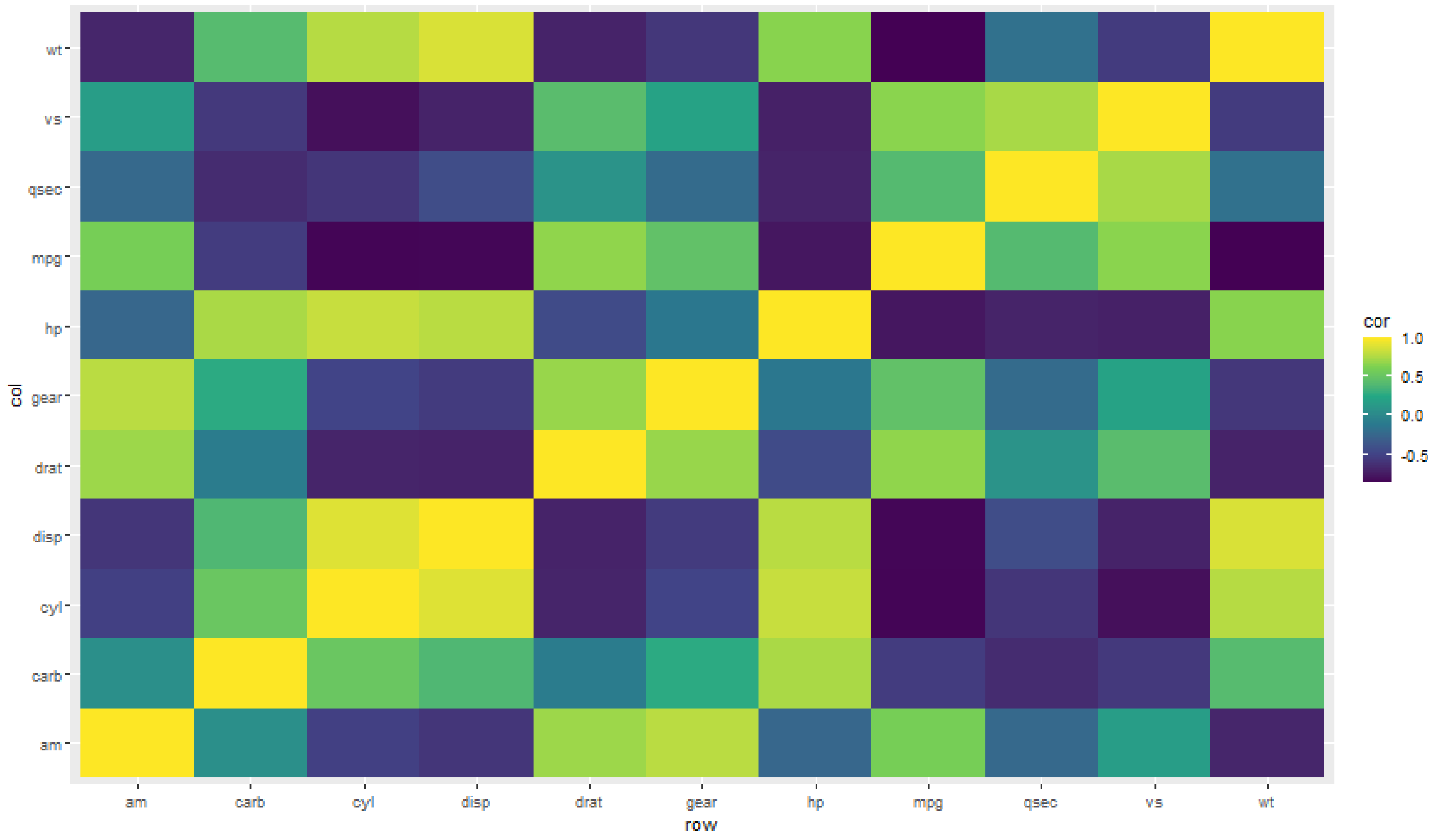
> A heatmap is a literal way of visualizing a table of numbers, where you substitute the numbers with colored cells. -- Nathan Yau

- Useful for finding highs and lows - and sometimes patterns
- They don't always work well

Example with correlations

```
## # A tibble: 6 x 3
##   row   col      cor
##   <chr> <chr>  <dbl>
## 1 mpg   mpg    1    
## 2 mpg   cyl   -0.852
## 3 mpg   disp  -0.848
## 4 mpg   hp    -0.776
## 5 mpg   drat   0.681
## 6 mpg   wt    -0.868
```


wt
vs
qsec
mpg
hp
gear
drat
disp
cyl
carb
am
col
am
carb
cyl
disp
drat
gear
hp
mpg
qsec
vs
wt
row
cor
1.0
0.5
0.0
-0.5


wt
vs
qsec
mpg
hp
gear
drat
disp
cyl
carb
am
col
am
carb
cyl
disp
drat
gear
hp
mpg
qsec
vs
wt
row
cor
1.0
0.5
0.0
-0.5

Categorical Data

Data

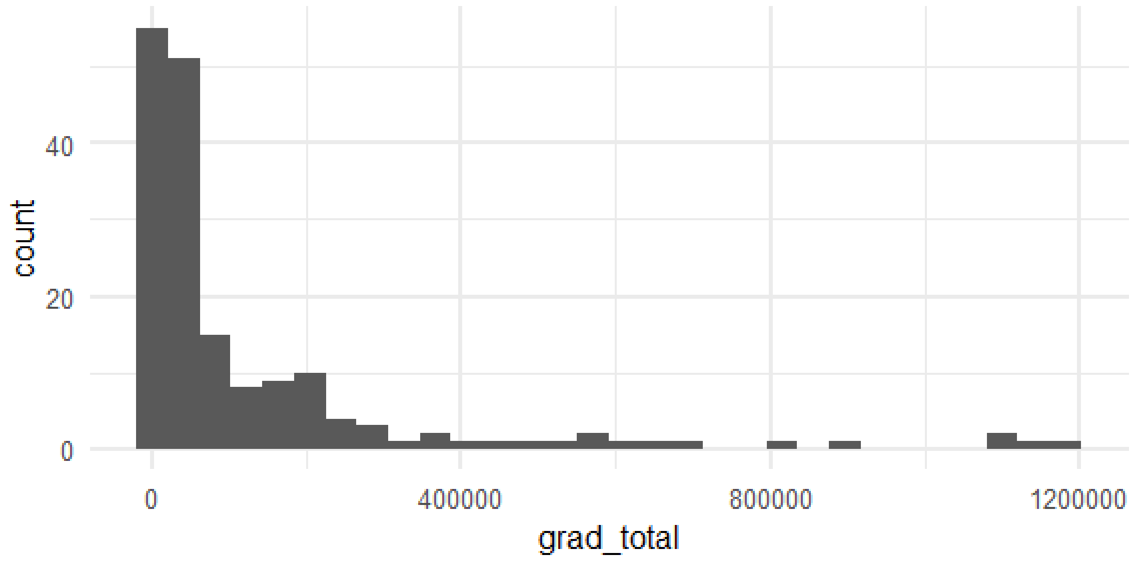
{fivethirtyeight} package

- *college_grad_students*

```
## # A tibble: 173 x 22
##    major_code major major_category grad_total grad_sample_size grad_employe
##         <int> <chr> <chr>               <int>            <int>          <int
##  1       5601 Cons~ Industrial Ar~       9173              200           709
##  2       6004 Comm~ Arts                53864              882          4049
##  3       6211 Hosp~ Business            24417              437          1836
##  4       2201 Cosm~ Industrial Ar~       5411               72           359
##  5       2001 Comm~ Computers & M~       9109              171           751
##  6       3201 Cour~ Law & Public ~       1542               22           100
##  7       6206 Mark~ Business           190996             3738         15157
##  8       1101 Agri~ Agriculture &~      17488              386          1310
##  9       2101 Comp~ Computers & M~       5611               98           471
## 10       1904 Adve~ Communication~      33928              688          2851
## # ... with 163 more rows, and 11 more variables: grad_p75th <int>, nongrad_
## #   nongrad_employed_fulltime_yearround <int>, nongrad_unemployed <int>, no
## #   nongrad_p75th <dbl>, grad_share <dbl>, grad_premium <dbl>
```

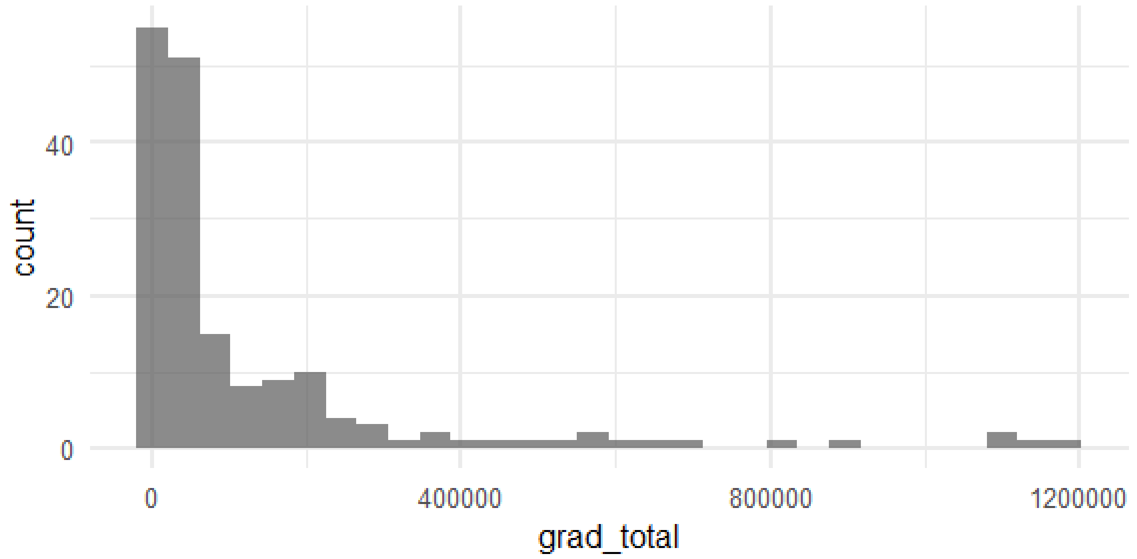
Histogram

Histogram of "grad_total"



Transparency - `alpha`

Add some transparency – perhaps this looks nicer



`color` vs. `fill`

In general

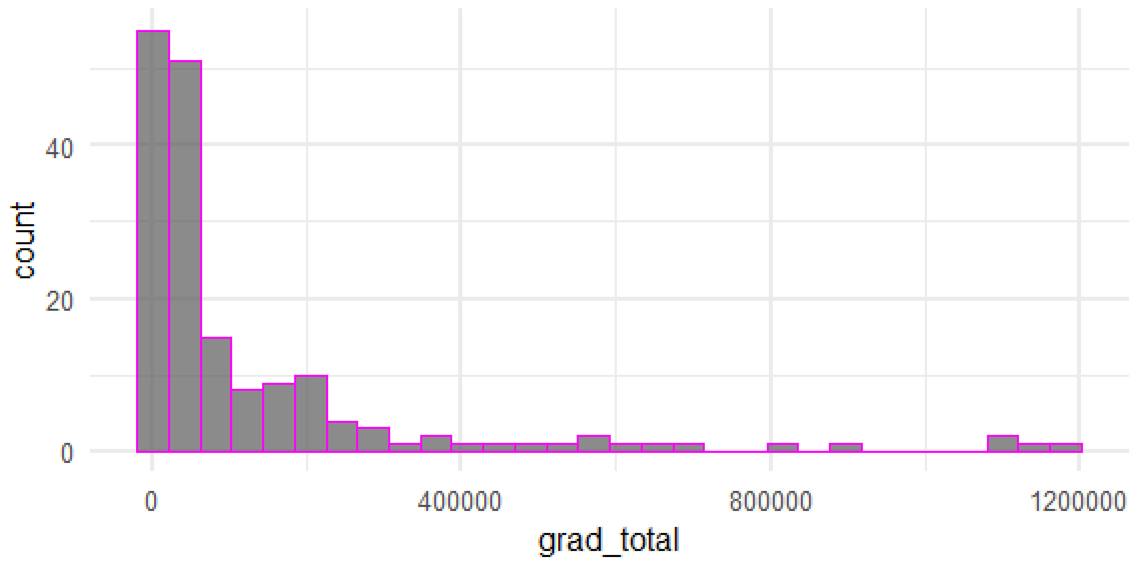
- `color` defines the color a geom is *outlined*
- `fill` defines the color a geom is *filled*

For example:

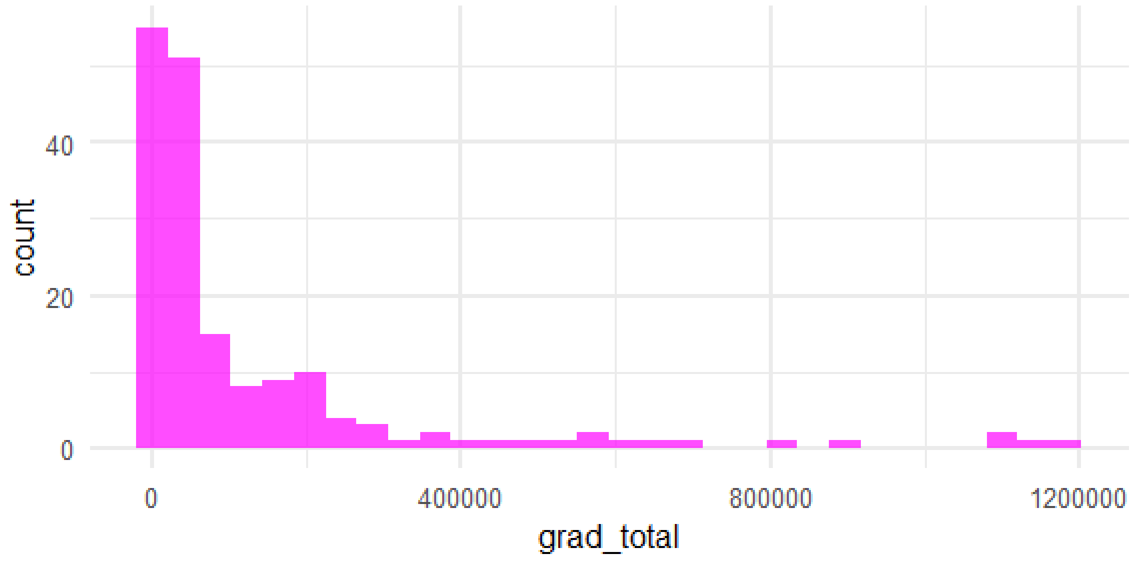
- `geom_point()` default has only has a color and **NO** fill because they're just points
- Point shapes 21–24 include both a color and a fill

How would we change the color of this plot?

How would we change the color of this plot?



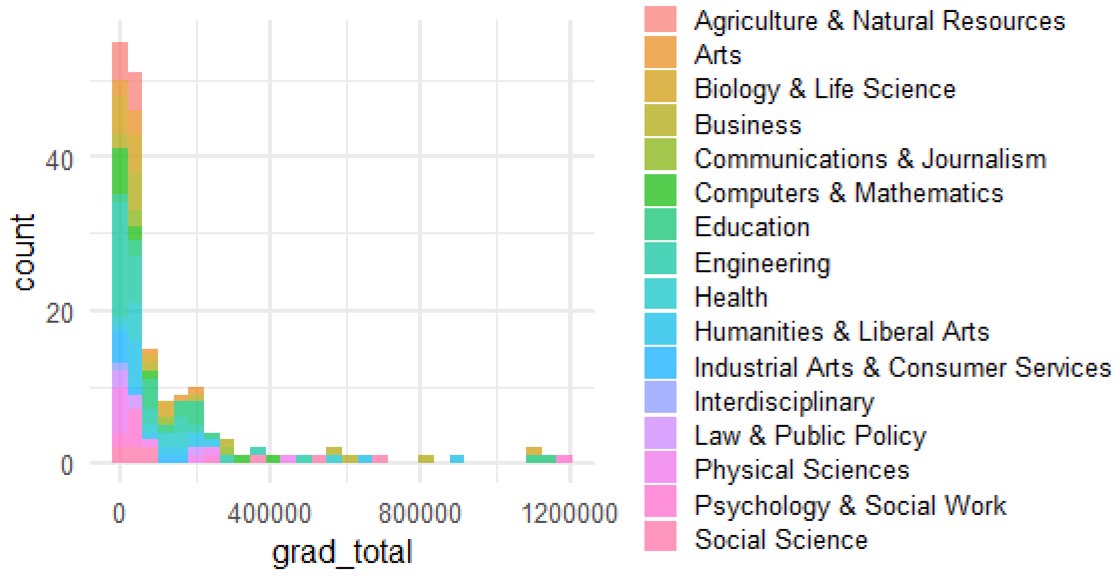
How would we change the ~~color~~ fill of this plot?



Color by variable

What if we wanted different colors by a variable

- *major_category*



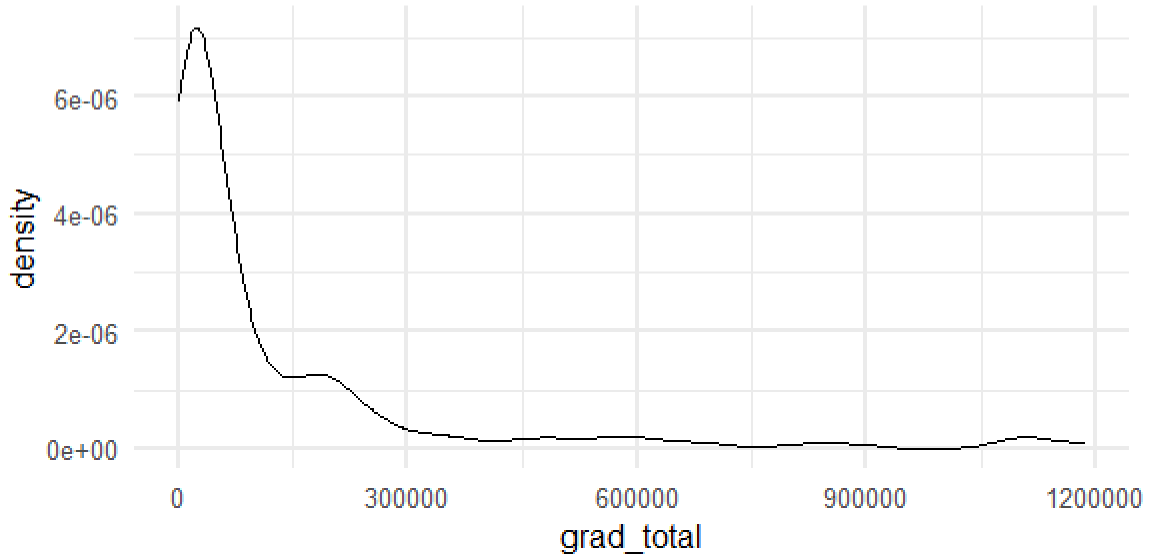
Density plot

Alternative representation of distribution

- Think of it as a smoothed histogram (uses kernel smoothing)
- The depiction of the distribution is **NOT** determined by the number of the bins you use, as are histograms

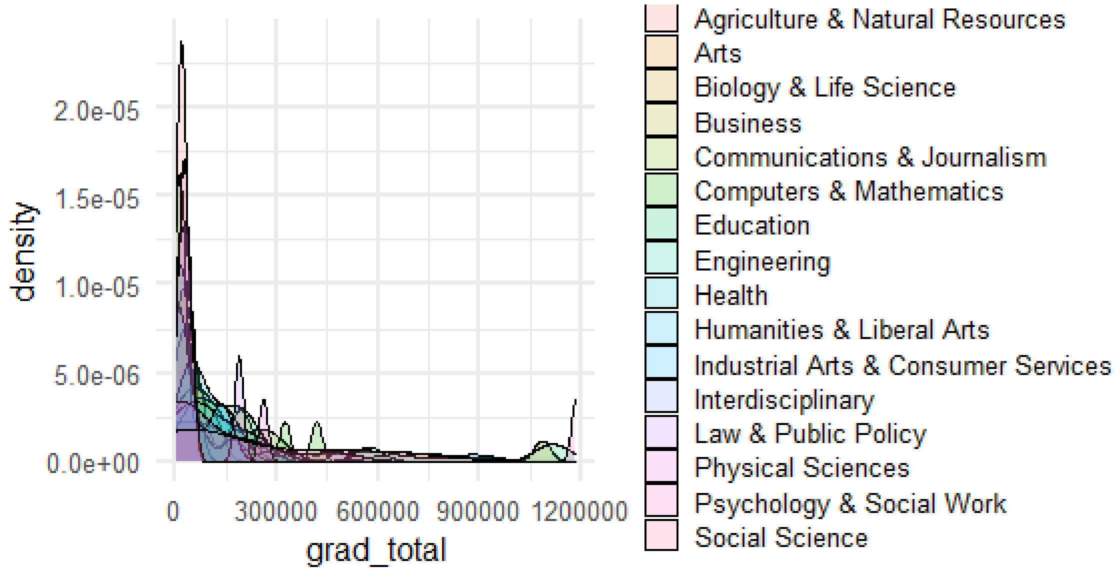
Density plot

`geom_density()`



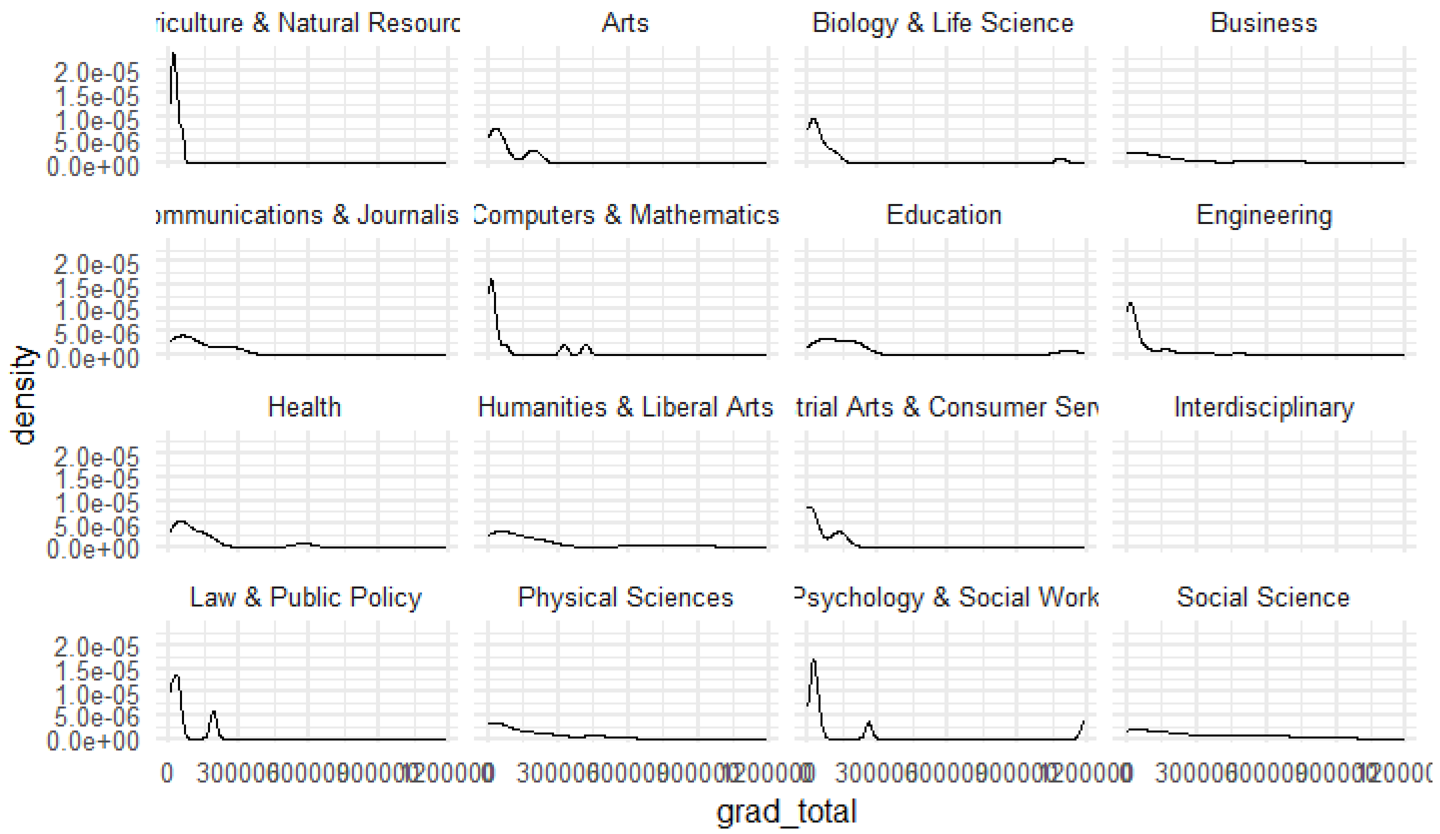
Density plot

Now let’s fill by *major_category*



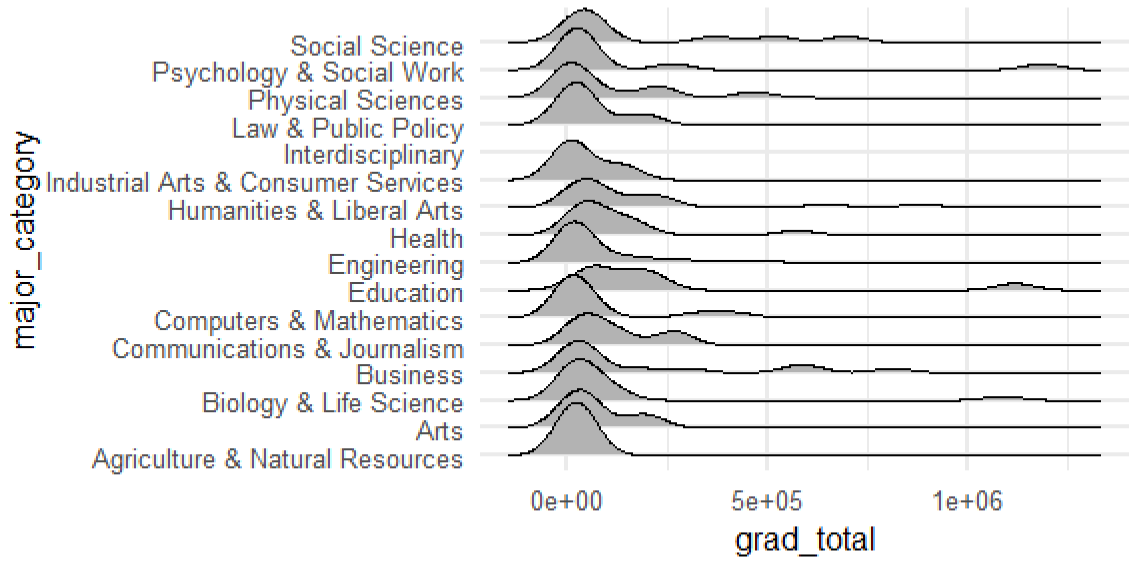
Not a good plot, just an example

Possible alternative? `facet_wrap`

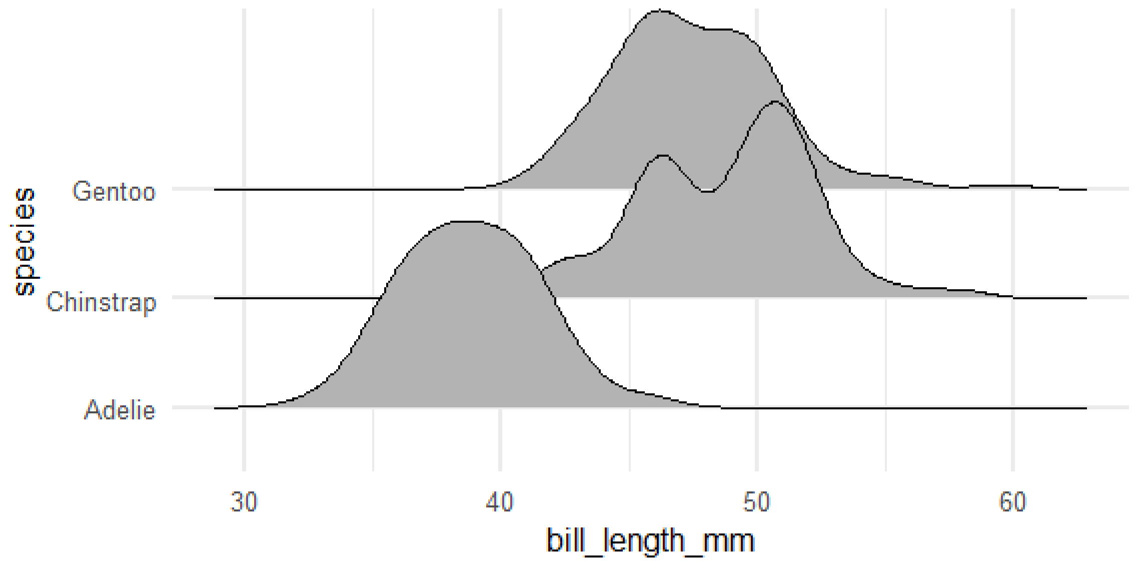


Even better

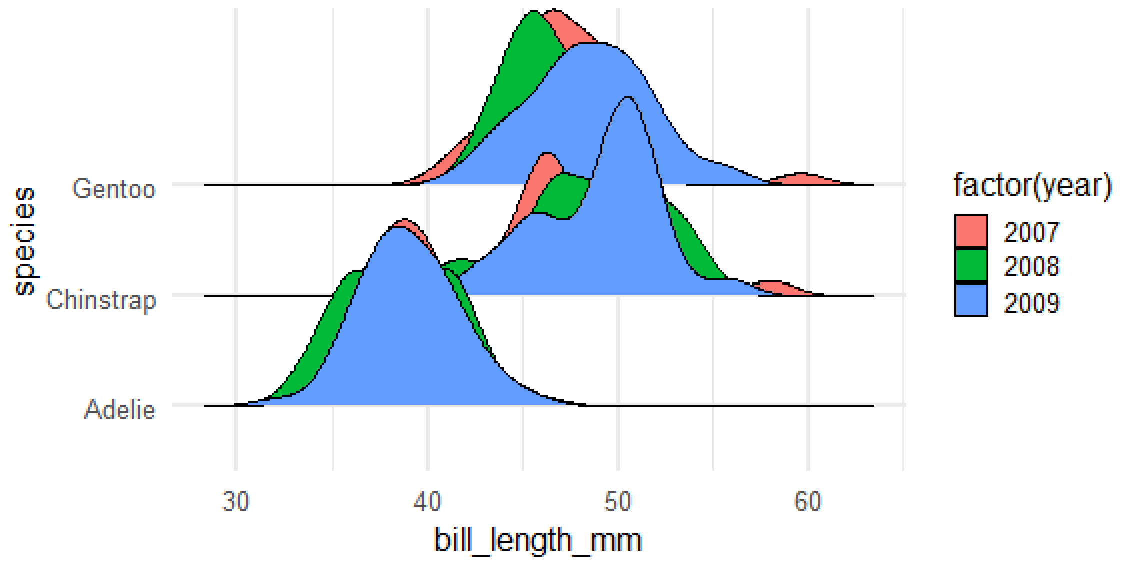
density ridges



Density Ridges

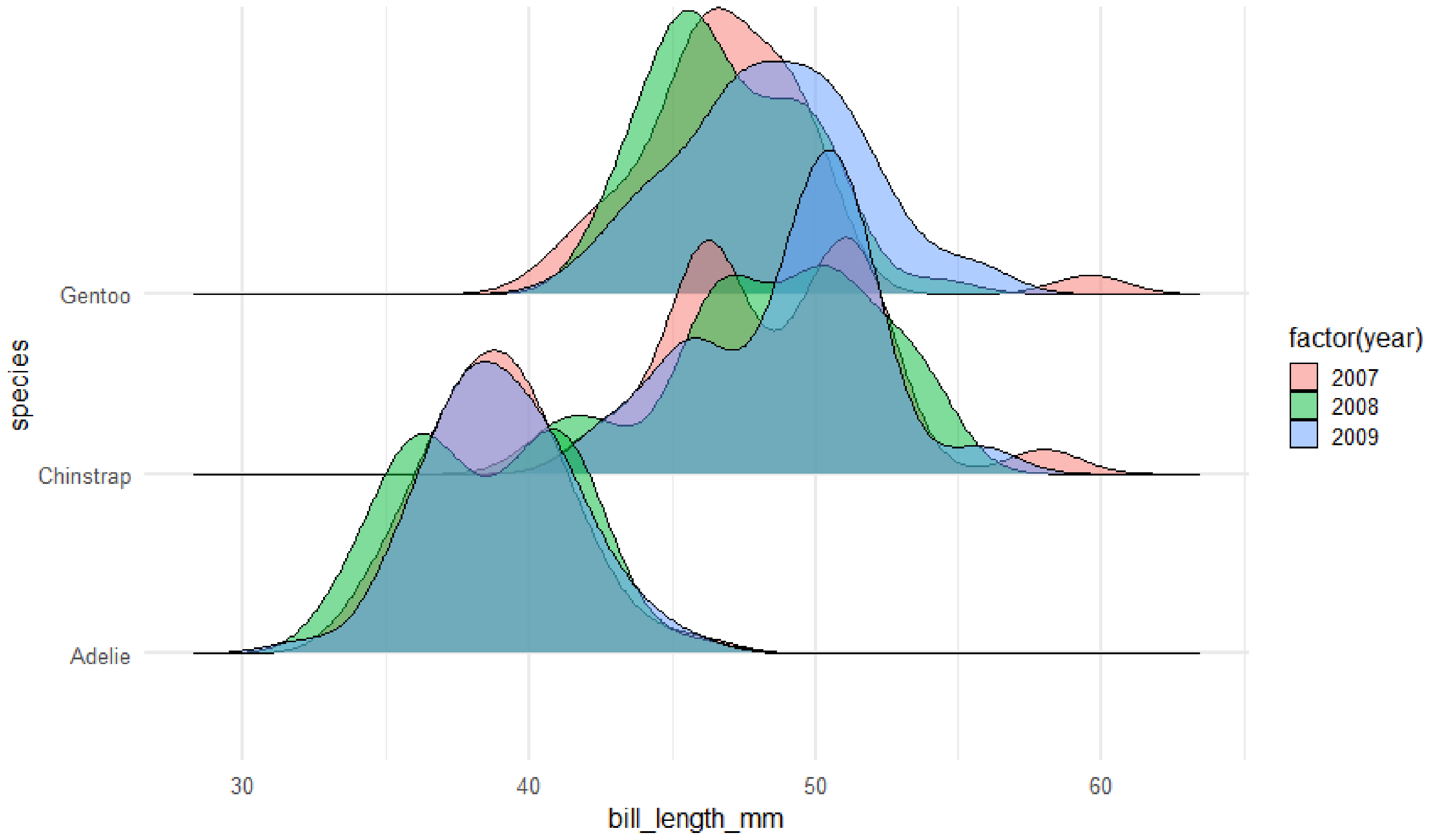


`fill`



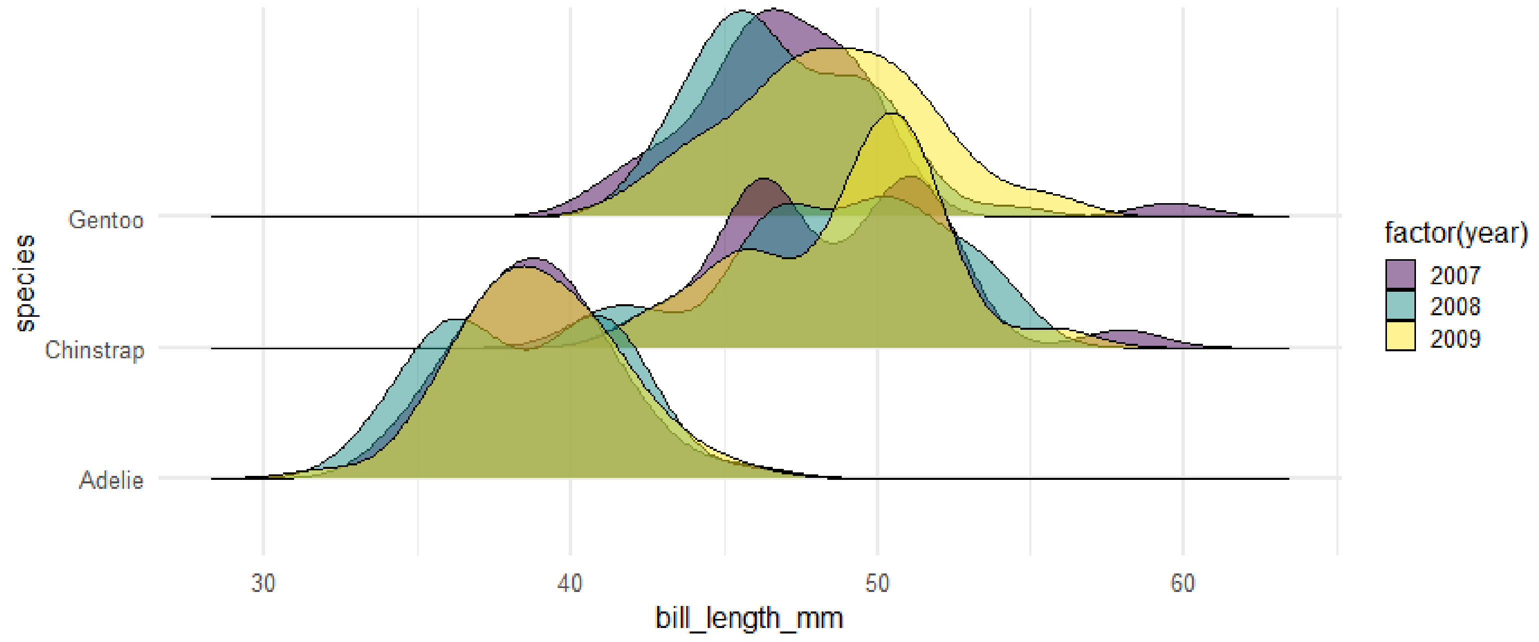
Gentoo
Chinstrap
Adelie
species
30
40
50
60
bill_length_mm
factor(year)
2007
2008
2009

Add transparency for clarity



Viridis

- easier to read by those with colorblindness
- prints well in gray scale

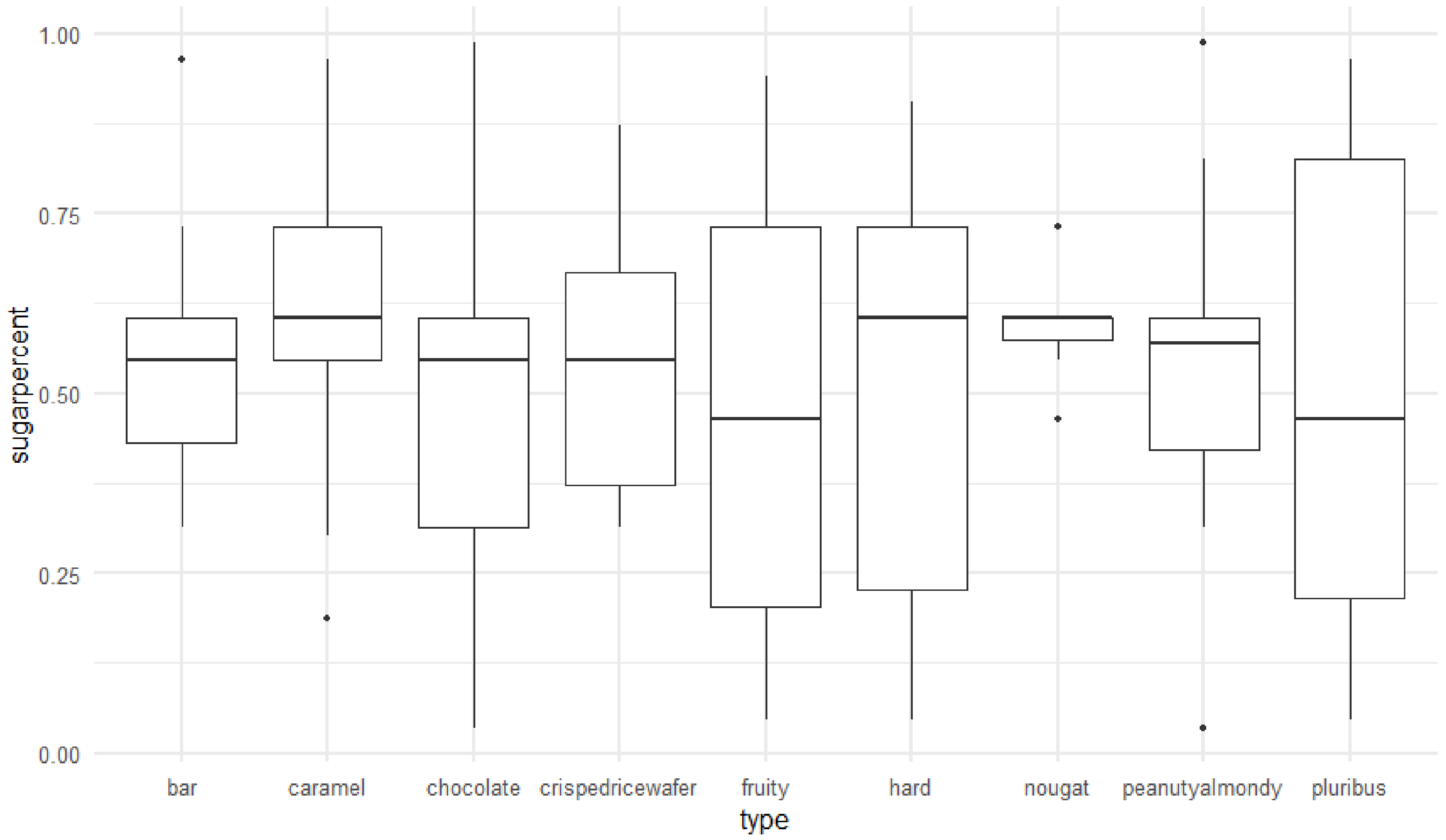


Same fill function, different option

Candy rankings {fivethirtyeight}

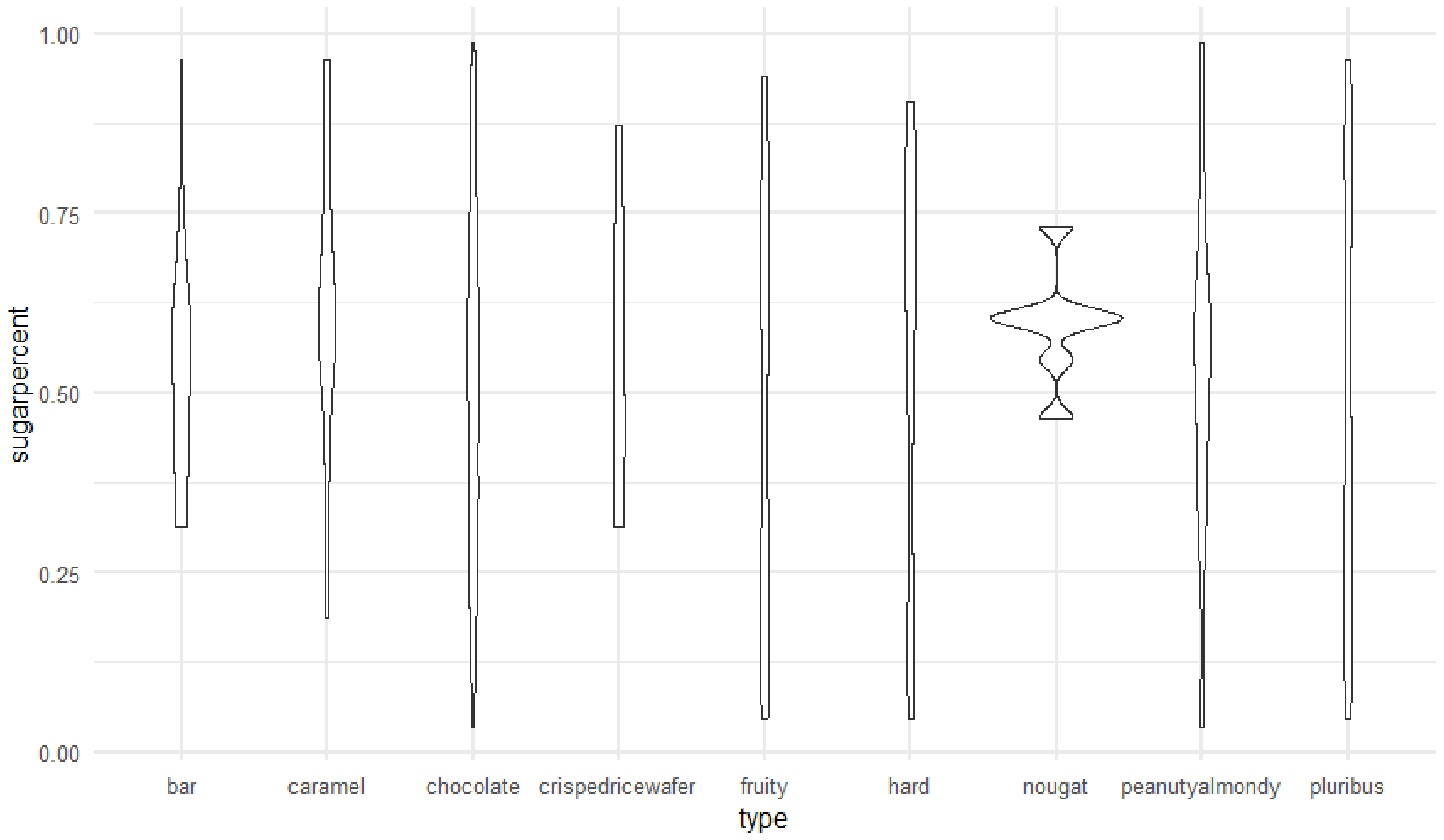
```
## # A tibble: 197 x 5
##    competitorname sugarpercent pricepercent winpercent type
##    <chr>                 <dbl>        <dbl>      <dbl> <chr>
##  1 100 Grand             0.732        0.860       67.0 chocolate
##  2 100 Grand             0.732        0.860       67.0 caramel
##  3 100 Grand             0.732        0.860       67.0 crispedricewafer
##  4 100 Grand             0.732        0.860       67.0 bar
##  5 3 Musketeers          0.604        0.511       67.6 chocolate
##  6 3 Musketeers          0.604        0.511       67.6 nougat
##  7 3 Musketeers          0.604        0.511       67.6 bar
##  8 Air Heads             0.906        0.511       52.3 fruity
##  9 Almond Joy            0.465        0.767       50.3 chocolate
## 10 Almond Joy            0.465        0.767       50.3 peanutyalmondy
## # ... with 187 more rows
```

Boxplot



sugarpercent
1.00
0.75
0.50
0.25
0.00
bar
caramel
chocolate
crispedricewafer
fruity
hard
nougat
peanutyalmondy
pluribus
type

Violin plots



Bar Charts

```
## # A tibble: 6 x 11
##   manufacturer model displ  year   cyl trans      drv     cty   hwy fl    cl
##   <chr>        <chr> <dbl> <int> <int> <chr>      <chr> <int> <int> <chr> <
## 1 audi         a4      1.8  1999     4 auto(l5)   f        18    29 p     c
## 2 audi         a4      1.8  1999     4 manual(m5) f        21    29 p     c
## 3 audi         a4      2    2008     4 manual(m6) f        20    31 p     c
## 4 audi         a4      2    2008     4 auto(av)   f        21    30 p     c
## 5 audi         a4      2.8  1999     6 auto(l5)   f        16    26 p     c
## 6 audi         a4      2.8  1999     6 manual(m5) f        18    26 p     c
```

Bar Charts

```
## # A tibble: 6 x 2
## # Groups:   class [6]
##   class          n
##   <chr>      <int>
## 1 2seater        5
## 2 compact       47
## 3 midsize       41
## 4 minivan       11
## 5 pickup        33
## 6 subcompact    35
```

Bar/Col Charts

`geom_bar()`

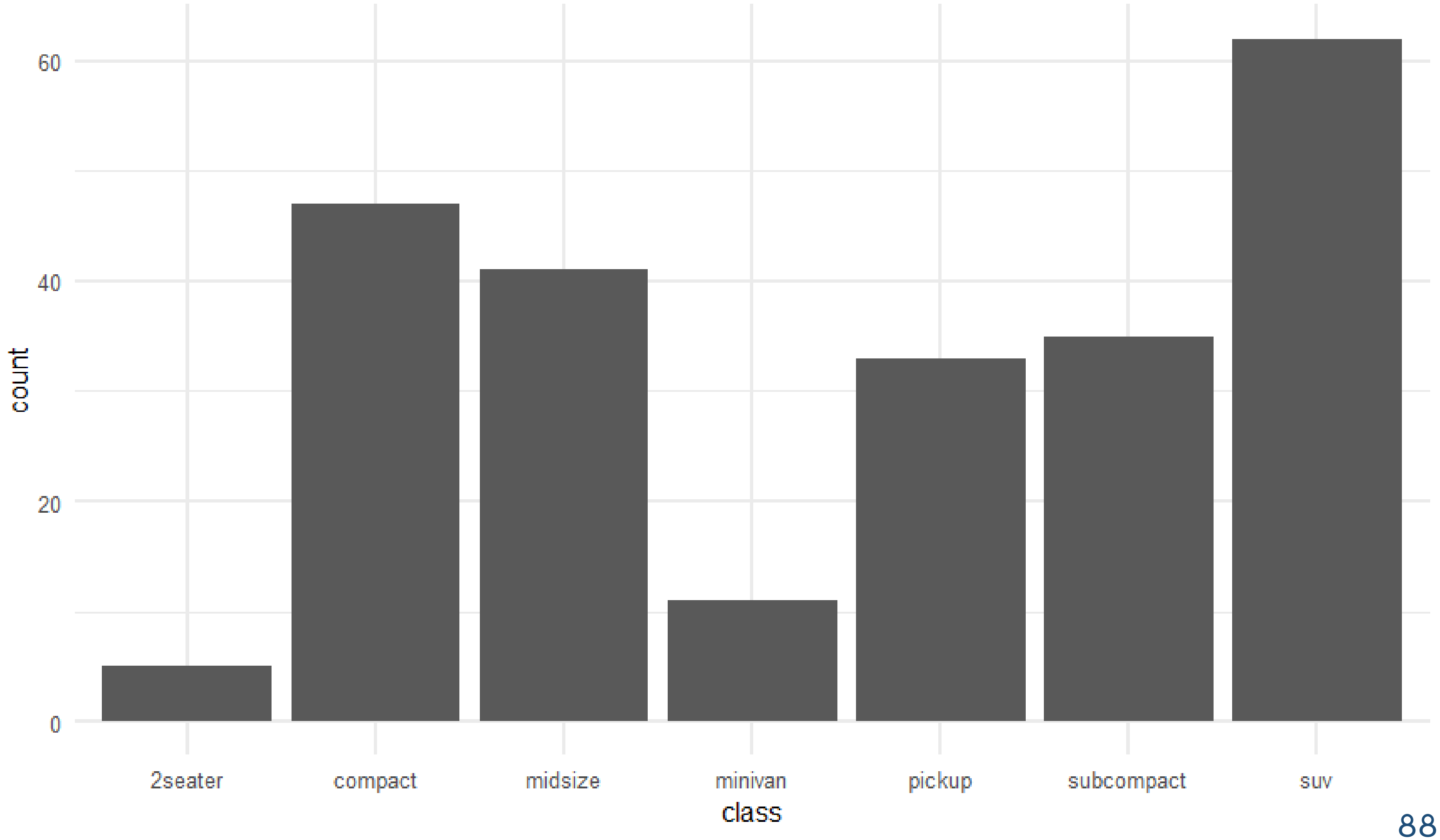
- expects *x* **OR** *y*
- counts rows
- if you want to count the number of cases at each *x* or *y* position
- makes the height of the bar proportional to the number of cases in each group
- uses `stat_count()` by default

`geom_col()`

- expects *x* **AND** *y*
- expects numbers in your data
- if you want the heights of the bars to represent values in the data
- leaves the data as is
- uses `stat_identity()` by default

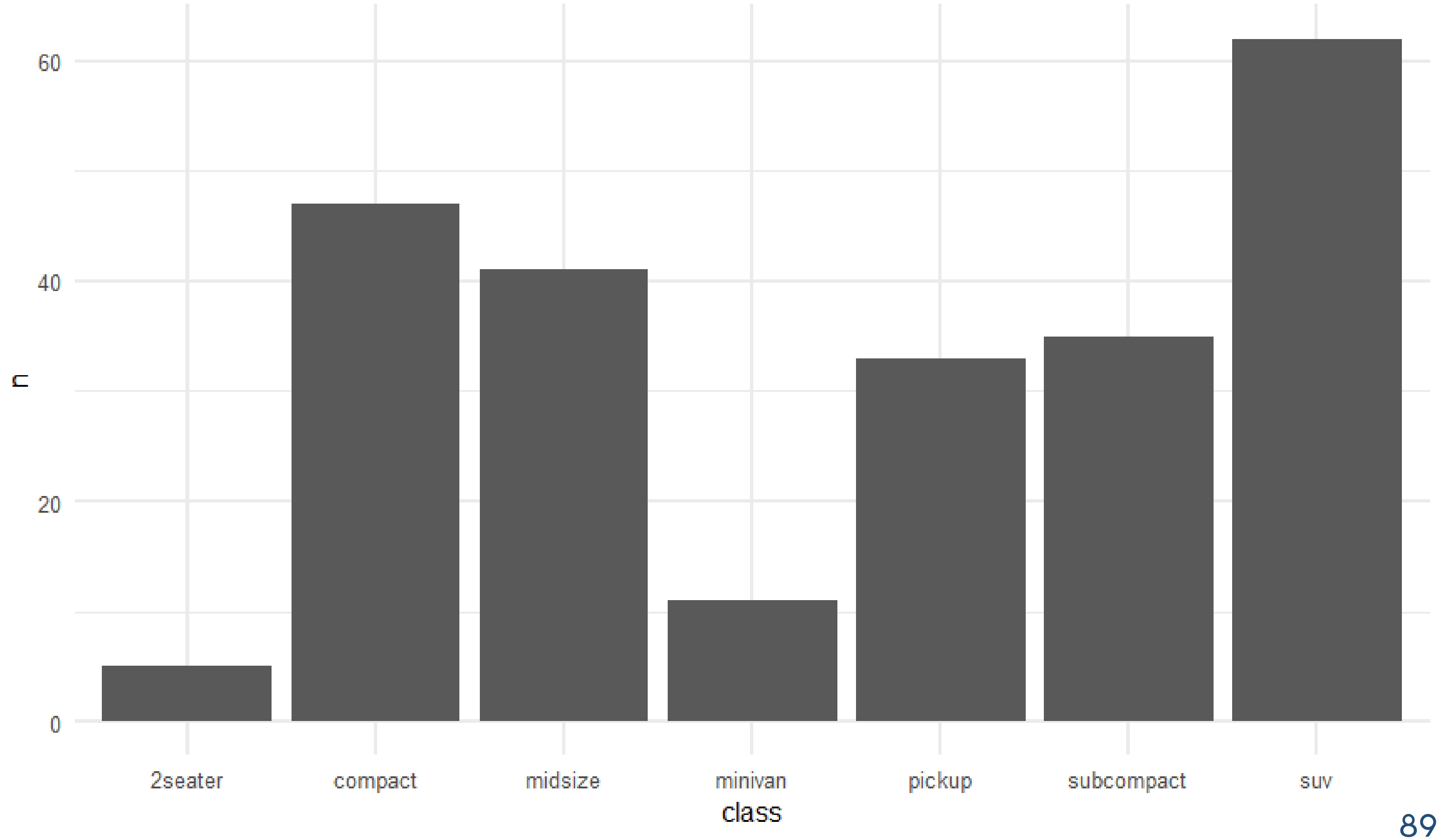
`geom_bar()`

mpg data

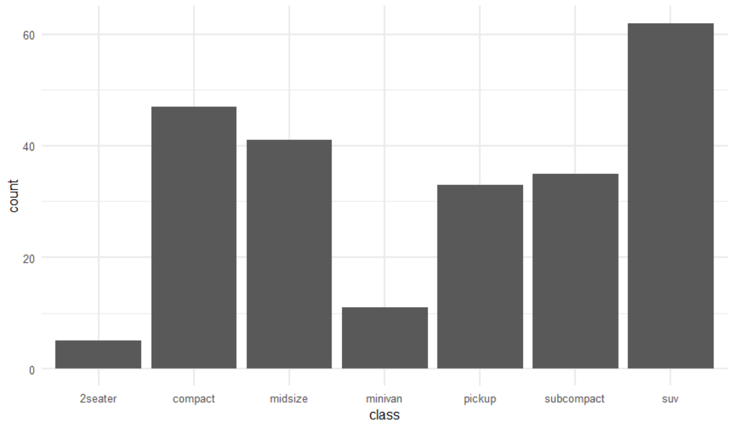


`geom_col()`

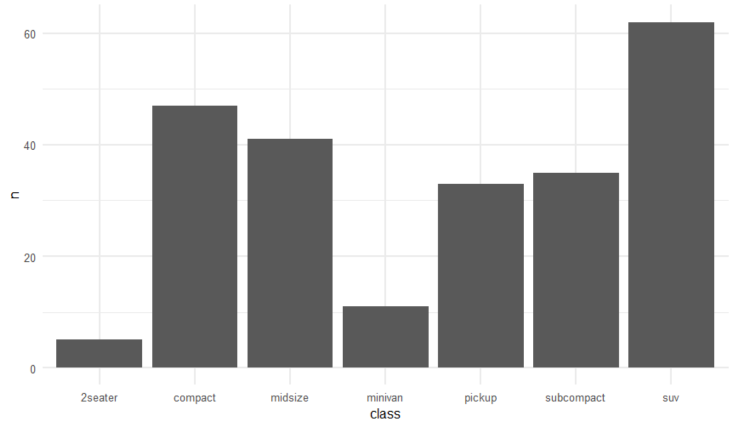
summarized_mpg data



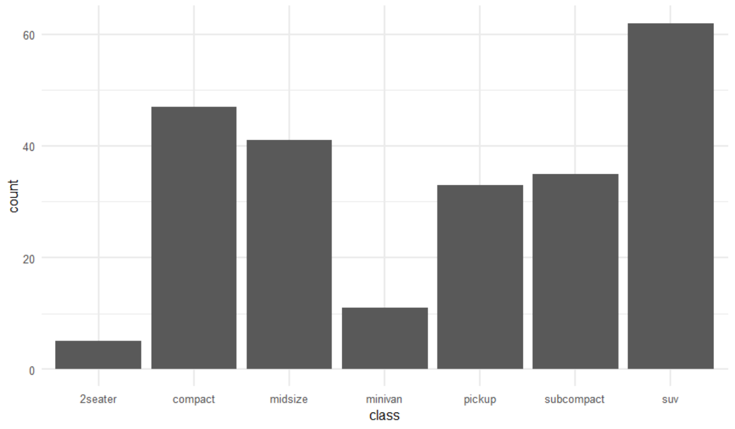
`geom_bar()` default



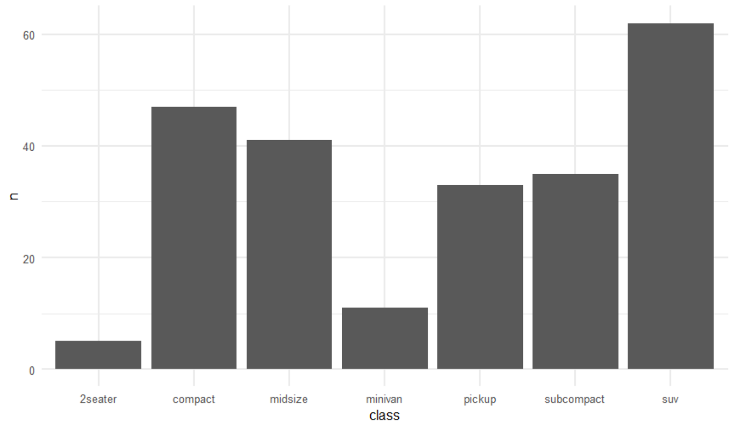
`geom_bar(stat = "identity")`



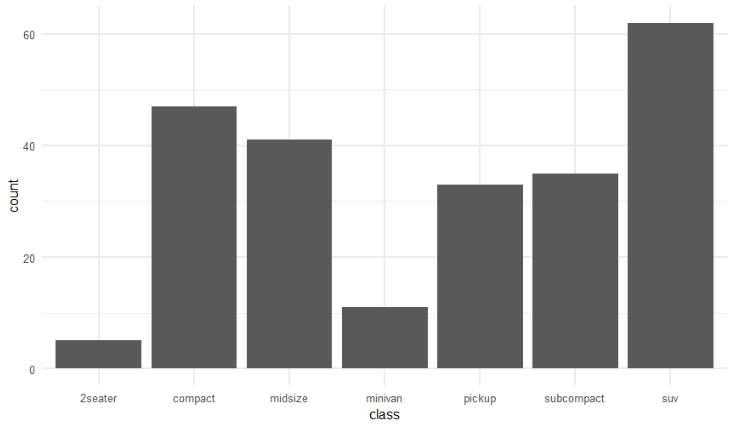
`geom_bar()` **default**



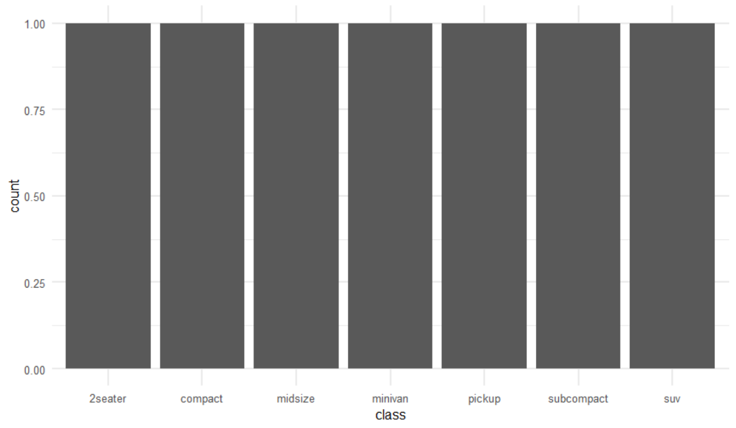
`geom_col()` **default**



`geom_bar()` default



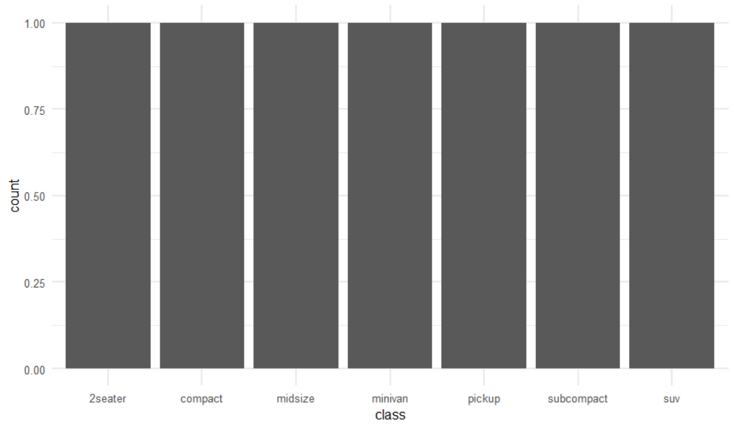
`geom_bar()` uh-oh



What happened?

Let's look at our data again

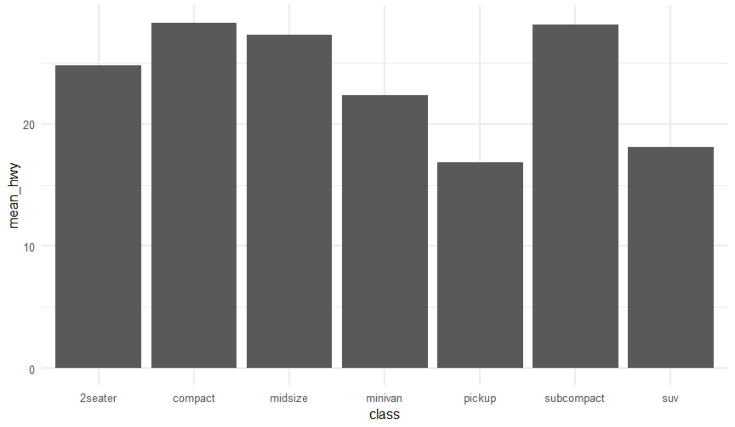
```
## # A tibble: 7 x 2
## # Groups:   class [7]
##   class          n
##   <chr>      <int>
## 1 2seater        5
## 2 compact       47
## 3 midsize       41
## 4 minivan       11
## 5 pickup        33
## 6 subcompact    35
## 7 suv           62
```

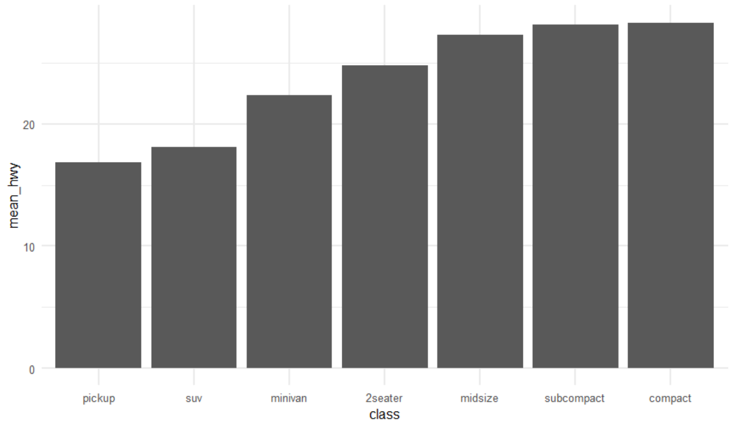


Change the order of the *x*-axis

- We'll talk about automated methods more later
- Basically - define *x*-axis variable as a factor with levels in the order you want

```
## # A tibble: 7 x 2
##   class      mean_hwy
##   <chr>         <dbl>
## 1 2seater        24.8
## 2 compact        28.3
## 3 midsize        27.3
## 4 minivan        22.4
## 5 pickup         16.9
## 6 subcompact     28.1
## 7 suv            18.1
```


mean_hwy
20
10
0
2seater
compact
midsize
minivan
pickup
subcompact
suv
class


mean_hwy
20
10
0
pickup
suv
minivan
2seater
midsize
subcompact
compact
class

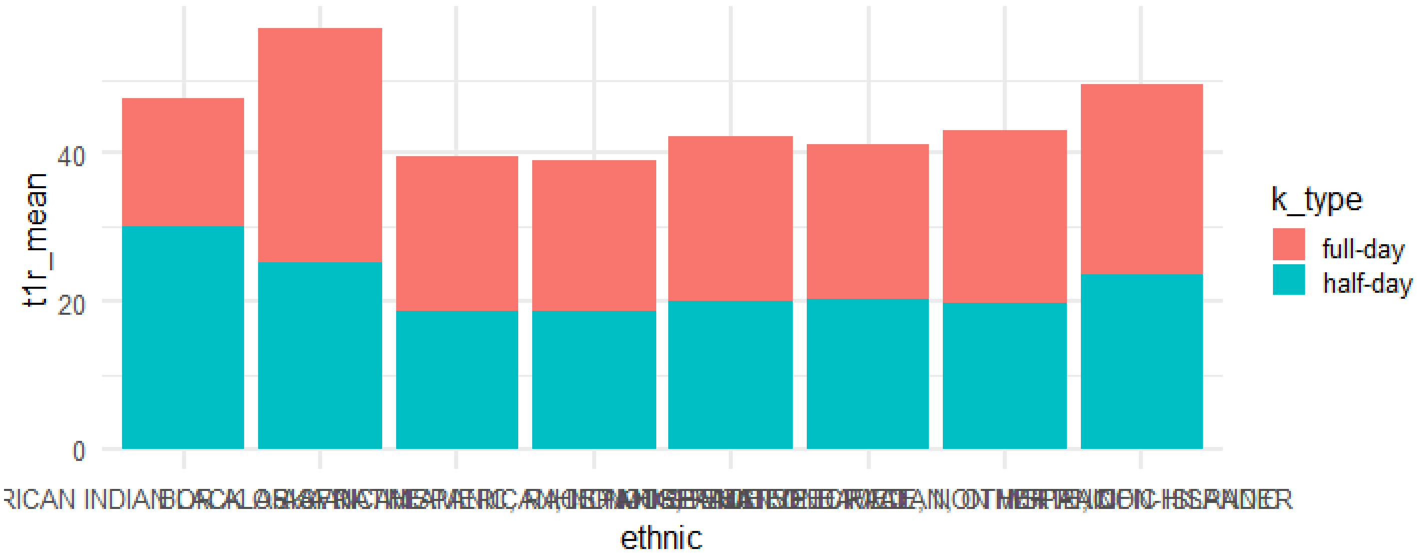
More bar plot options

Data

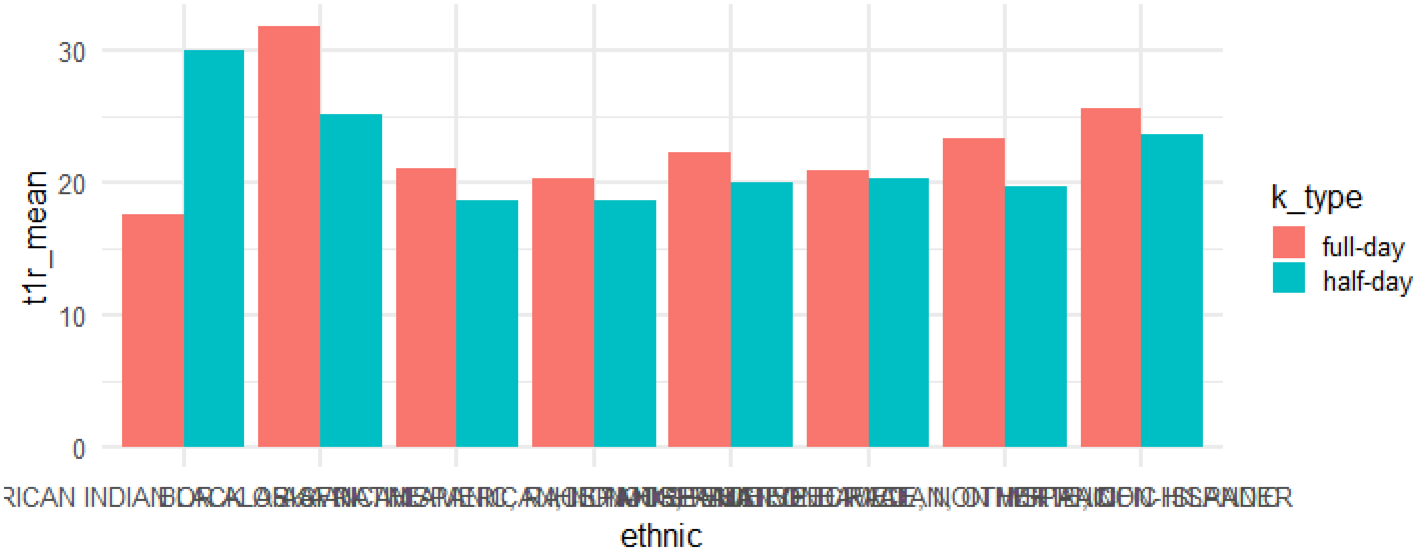
```
## # A tibble: 16 x 3
## # Groups:   k_type [2]
##    k_type   ethnic                                     t1r_mean
##    <chr>    <chr>                                         <dbl>
##  1 full-day AMERICAN INDIAN OR ALASKA NATIVE               17.5
##  2 full-day ASIAN                                          31.8
##  3 full-day BLACK OR AFRICAN AMERICAN, NON-HISPANIC        21.1
##  4 full-day HISPANIC, RACE NOT SPECIFIED                   20.3
##  5 full-day HISPANIC, RACE SPECIFIED                       22.3
##  6 full-day MORE THAN ONE RACE, NON HISPANIC               20.8
##  7 full-day NATIVE HAWAIIAN, OTHER PACIFIC ISLANDER        23.2
##  8 full-day WHITE, NON-HISPANIC                            25.5
##  9 half-day AMERICAN INDIAN OR ALASKA NATIVE               30.0
## 10 half-day ASIAN                                          25.1
## 11 half-day BLACK OR AFRICAN AMERICAN, NON-HISPANIC        18.6
## 12 half-day HISPANIC, RACE NOT SPECIFIED                   18.7
## 13 half-day HISPANIC, RACE SPECIFIED                       20.0
## 14 half-day MORE THAN ONE RACE, NON HISPANIC               20.3
## 15 half-day NATIVE HAWAIIAN, OTHER PACIFIC ISLANDER        19.7
## 16 half-day WHITE, NON-HISPANIC                            23.7
```

Stacked bar plot

Look for effects in "ethnicity" by "k_type" (full/half day K)

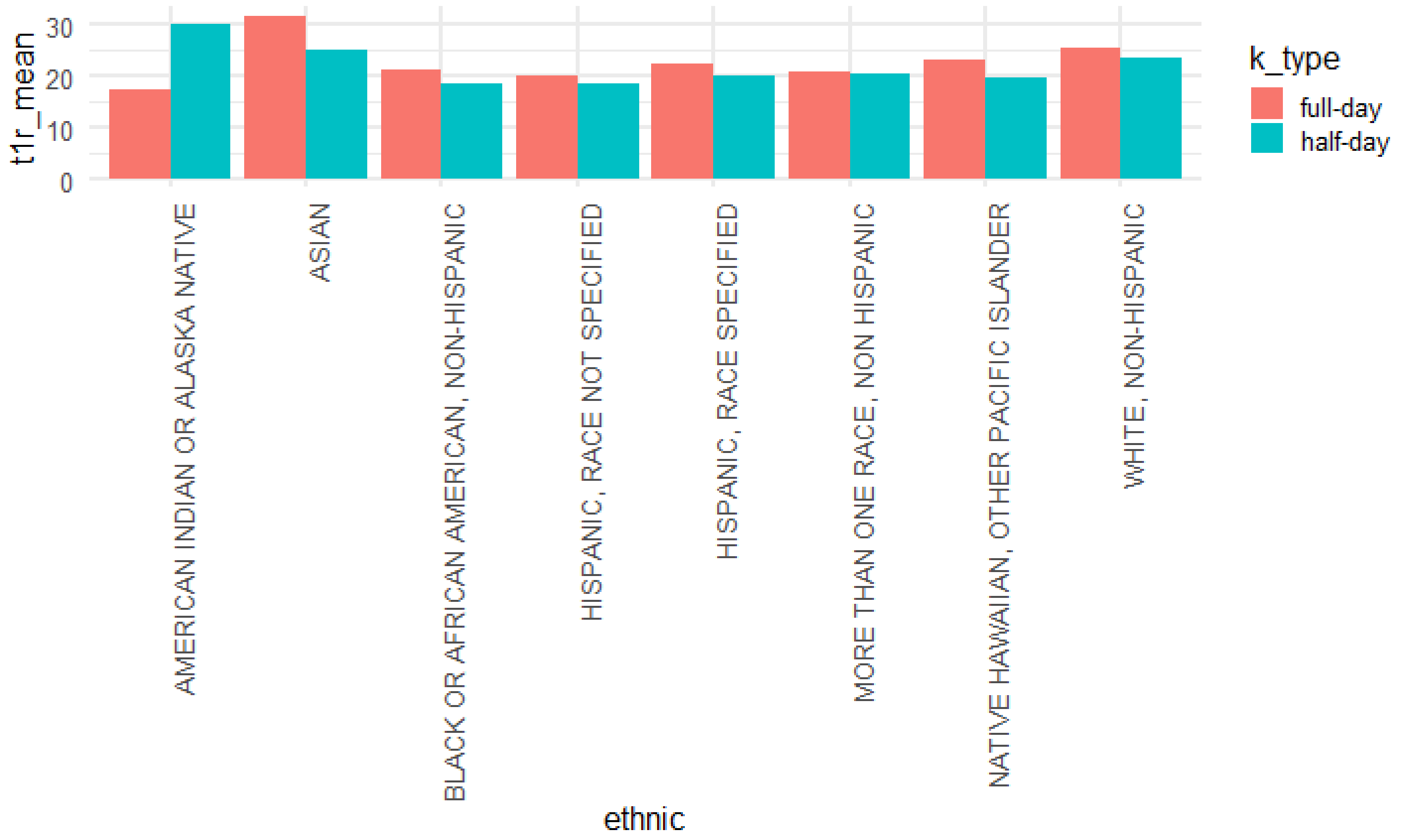


Grouped bar plot



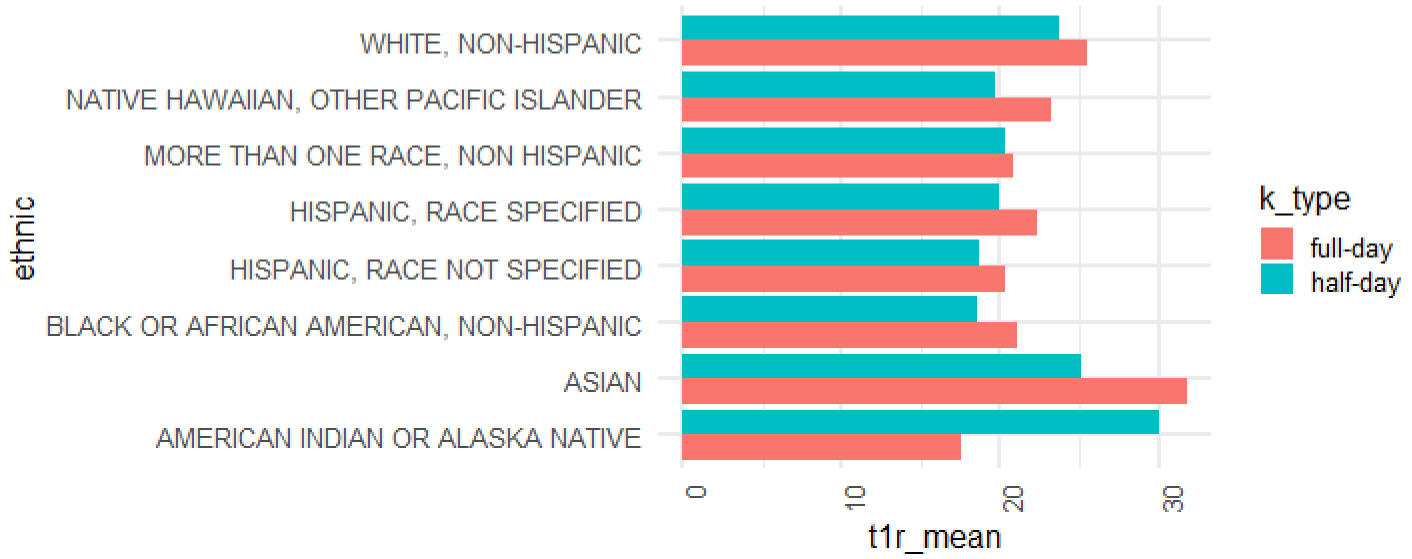
Rotating Labels

I have to look this up *every* time



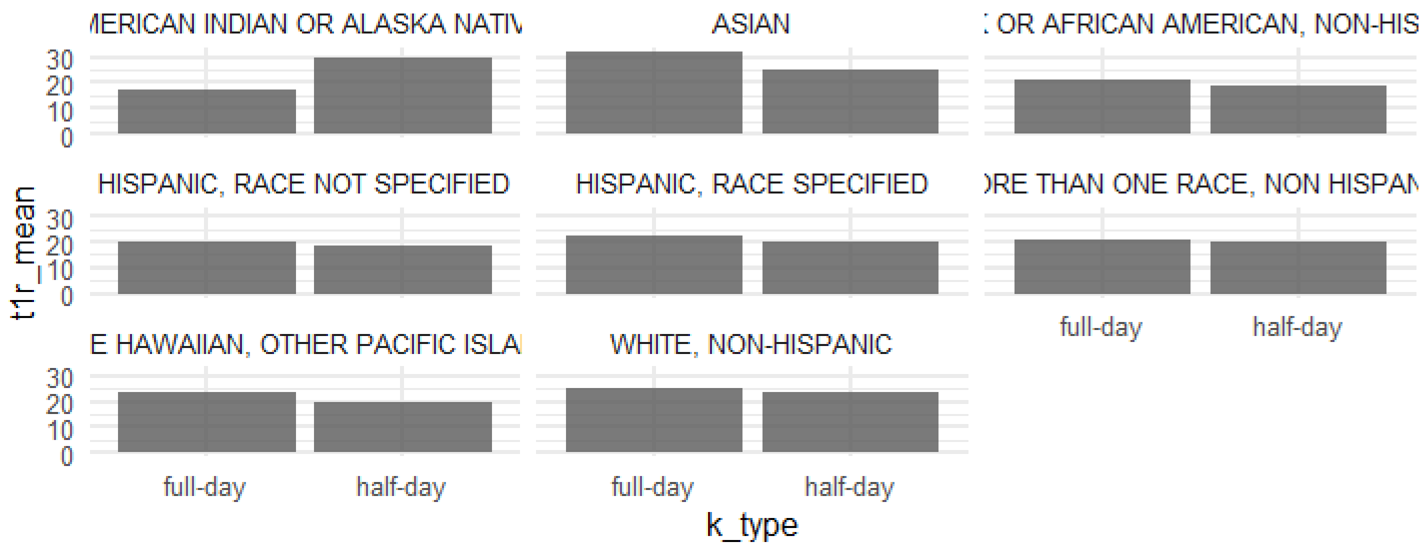
Flip the coordinates

`coord_flip()`

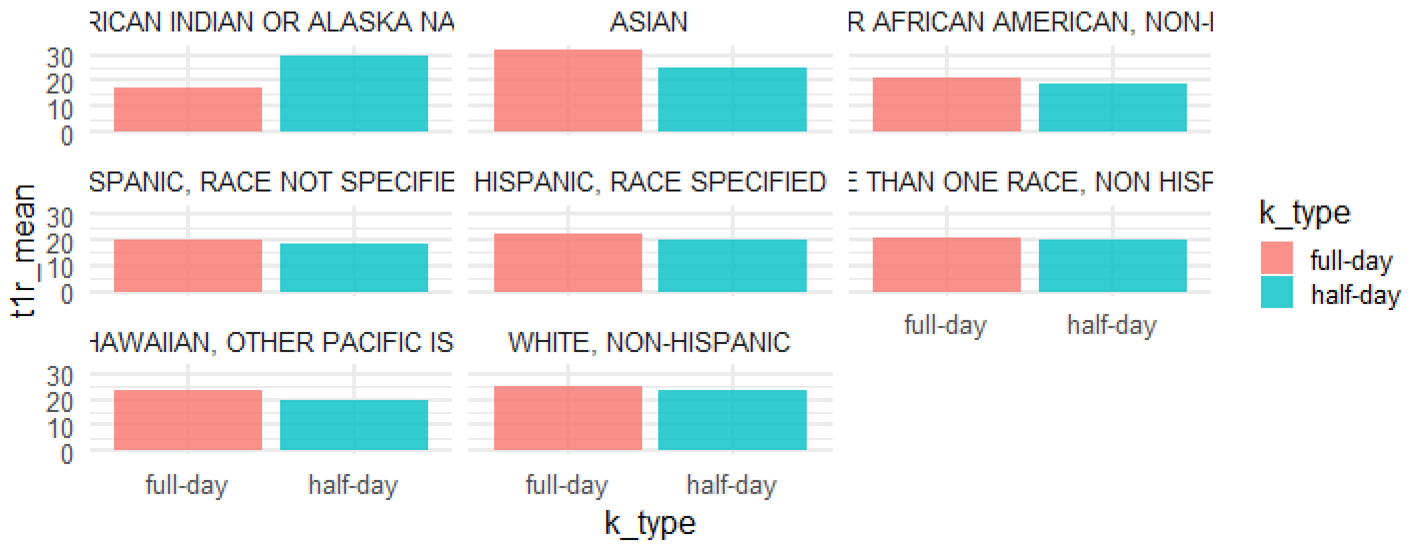


Alternatively

`facet_wrap()`



Sometimes some redundancy works well



`geom_*()` Review

- `geom_point()`
- `geom_smooth()`
- `geom_line()`
- `geom_tile()`
- `geom_histogram()`
- `geom_density()`
- `ggridges::geom_density_ridges()`
- `geom_boxplot()`
- `geom_violin()`
- `geom_bar()`
- `geom_col()`

Challenge

- Start a new R project
- Create a new script, save it as "practice-plots.R"
- Load the `{tidyverse}`
- Print the **msleep** dataset to see it's structure (it's from `{ggplot2}`)

For each of the following, produce a separate plot

1. Plot the relation between "sleep_total" and "brainwt" (with "brainwt" as the DV) - scatter plot
2. Overlay a smooth on the previous plot
3. Color the points by "vore", but fit a single smooth
4. Fit separate smooths by "vore", but with all points being gray
5. Omit the standard error of the smooths
6. Use `ylim()` as an additional layer to restrict the *y*-axis to range from `0` to 5

Next time

Before next class

- Reading
  - R4DS 5
- Supplemental Learning
  - RStudio Primer: Working with Tibbles
  - Rbootcamp: Ch 3
  - Codecademy: Aggregates in R
- Homework
  - **Homework 2**
  - **Homework 3**