Creating Beautiful Data Visualizations in R:

a `ggplot2` Crash Course

Samantha Tyner, Ph.D.

July 21, 2020, 2:00-4:00pm EDT

How to watch this webinar

For optimum learning:

- Have both the webinar software and rstudio.cloud visible at all times
  - Material available at https://rstudio.cloud/project/1442337
- Follow along with the code (`project/code/03-follow-along.R`) or the slides (`project/slides/slides.Rmd`) and run the code as we go
- Use the ask question and chat features to communicate with TAs and ask questions
- Have fun!

Learning Goals

Upon completion of this tutorial, you will be able to:

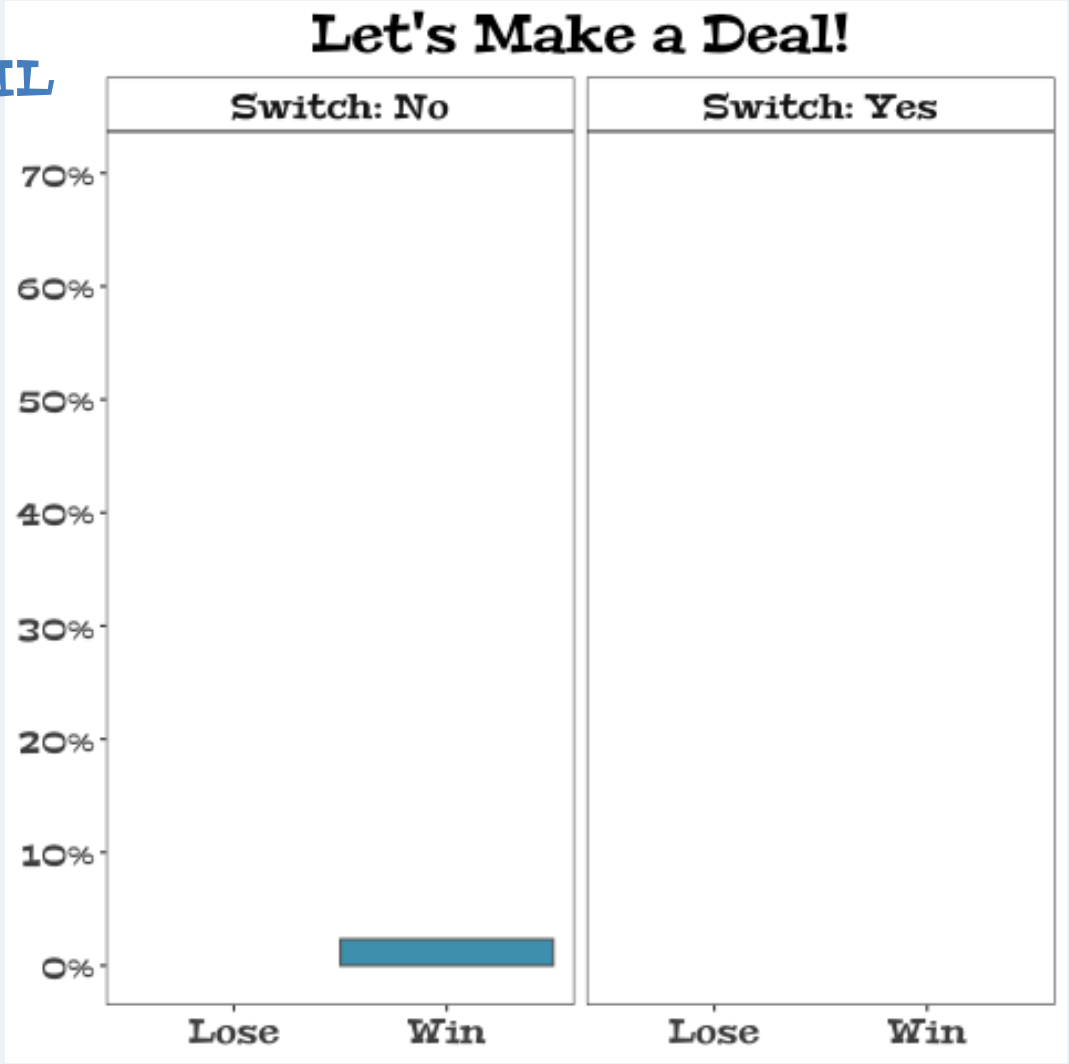
1. **identify** the appropriate plot types and corresponding `ggplot2` `geom`s to consider when visualizing your data;
2. **implement** the `ggplot2` grammar of graphics by using `ggplot()` and building up plots with the `+` operator;
3. **iterate** through multiple visualizations of your data by changing the aesthetic mappings, geometries, and other graph properties;
4. **incorporate** custom elements (colors, fonts, etc.) into your visualizations by adjusting `ggplot2` theme elements; and
5. **investigate** the world of `ggplot2` independently to expand upon the skills learned in the course.

Motivating the motivating example

Clip from *Brooklyn Nine-Nine*, Season 4, Episode 8:

Motivating example

GIF viewable on HTML version of slides available on rstudio.cloud.



ggplot2 and its Grammar of Graphics

Artwork by @allison_horst

What is the grammar of graphics?

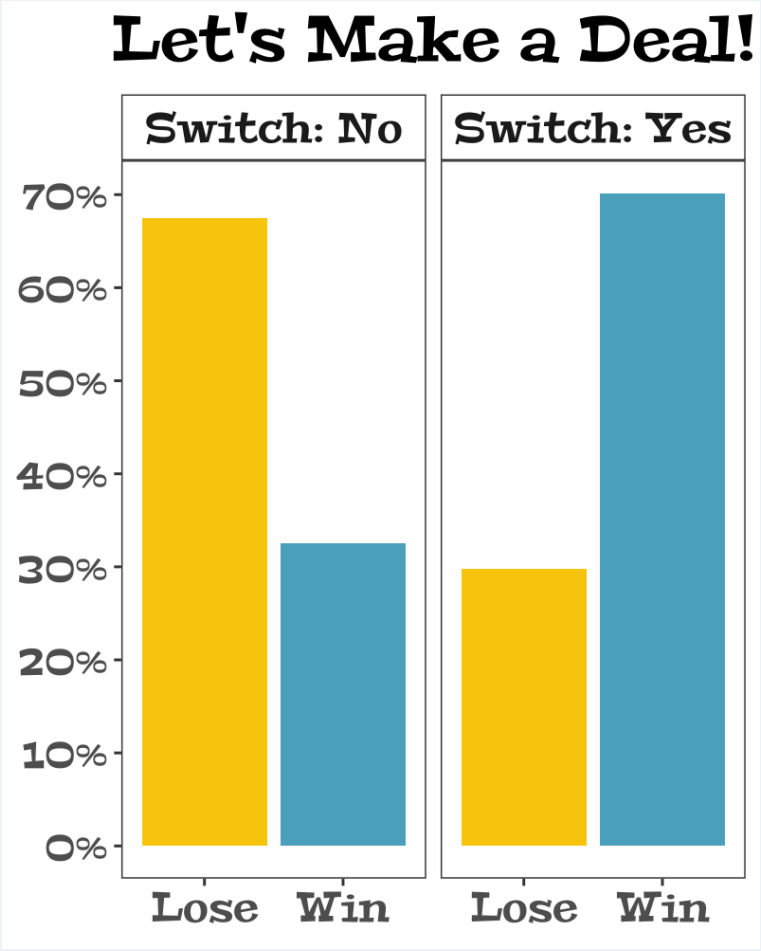
From a book, *The Grammar of Graphics* by Leland Wilkinson (1999)

> grammar (noun): (1) the study of the classes of words, their inflections, and their functions and relations in the sentence; (2) the principles or rules of an art, science, or technique
>
> grammar of graphics (noun): a set of principles for constructing data visualizations

grammar of language : sentence :: grammar of graphics : data visualization

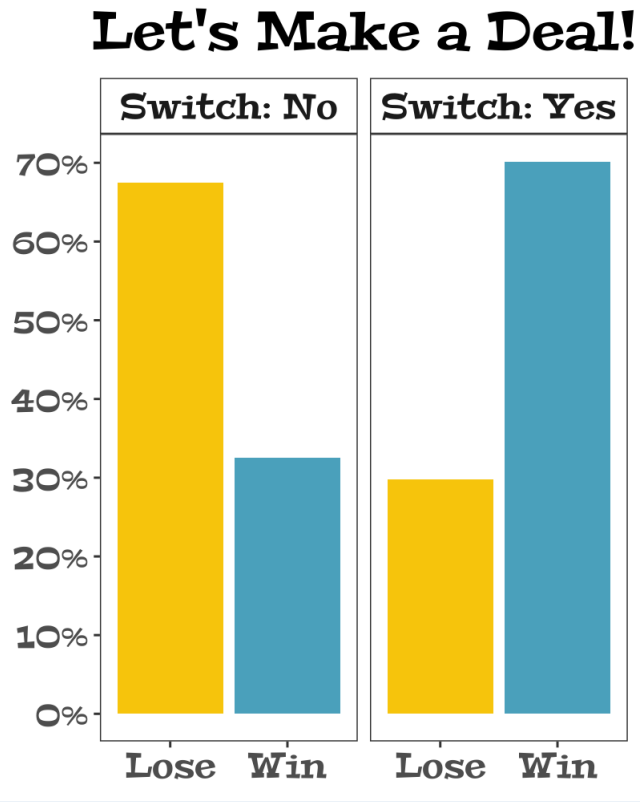
📢 I will try to use "data visualization" instead of "plot", "graph", "chart", or "graphic" because it is a more precise term.

A simple data viz



Data ↔ Noun

Data Visualization (Sentence)



Data (Noun)

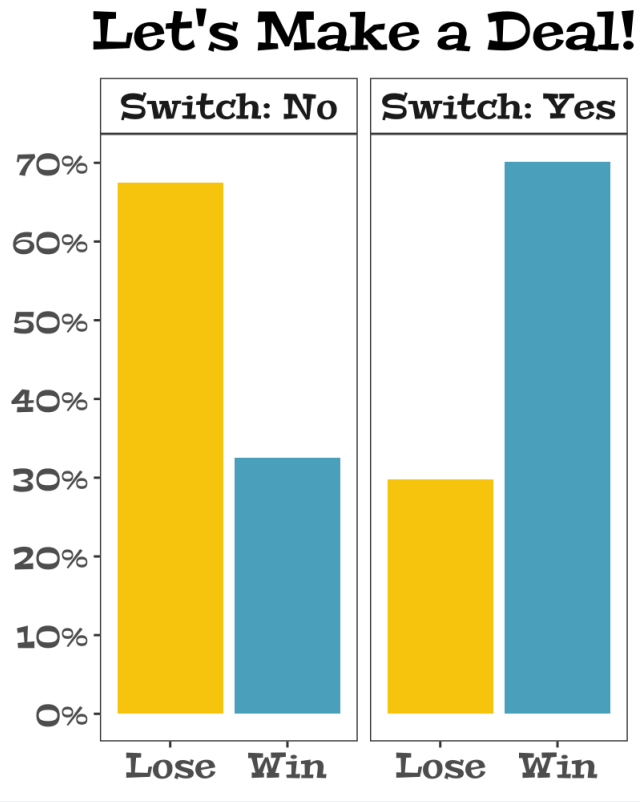
Switch	Win	n	perc
No	Lose	29	0.6744186
No	Win	14	0.3255814
Yes	Lose	17	0.2982456
Yes	Win	40	0.7017544

ggplot2 code

```
mh_sims ← readr::read_csv("dat/monty_hall.csv")
library(ggplot2)
ggplot(data = mh_sims)
```

Geom ↔ Verb

Data Visualization (Sentence)



Geom (Verb)

- bar chart a.k.a. column chart
- `geom_col()`

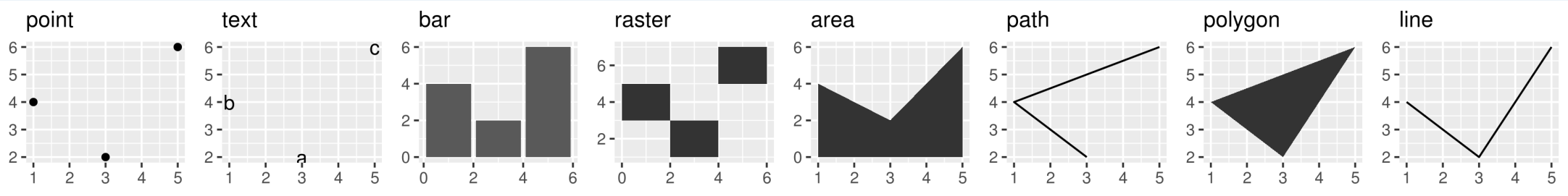
In ggplot2 code

We build up a data visualization in `ggplot2` with the `+` operator.

```
ggplot(data = mh_sims) +
  geom_col()
```

```
## Error: geom_col requires the following missing aesthetics: x, y
```

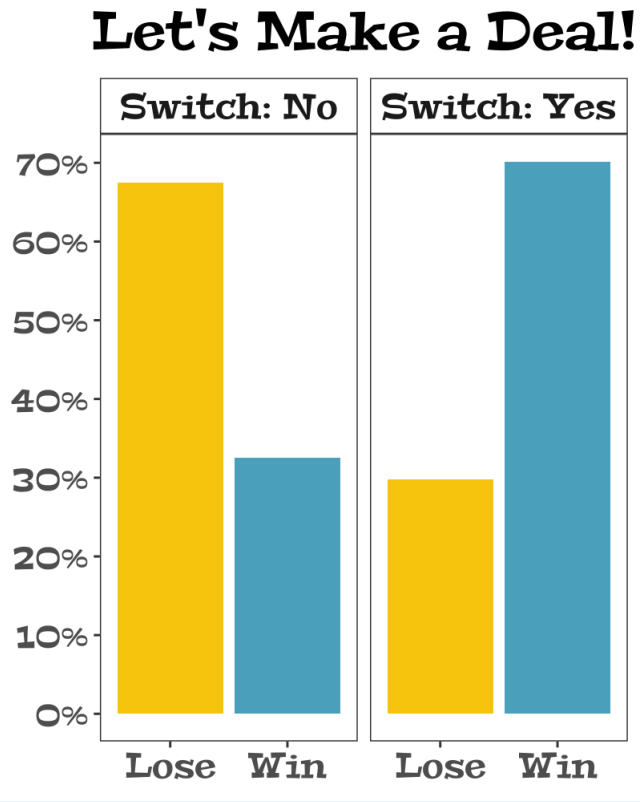
📢 The `geom_*()` suite of functions can take many arguments, which vary by the geom type



Source: ggplot2 book

aes mapping ↔ Pronouns

Data Visualization (Sentence)



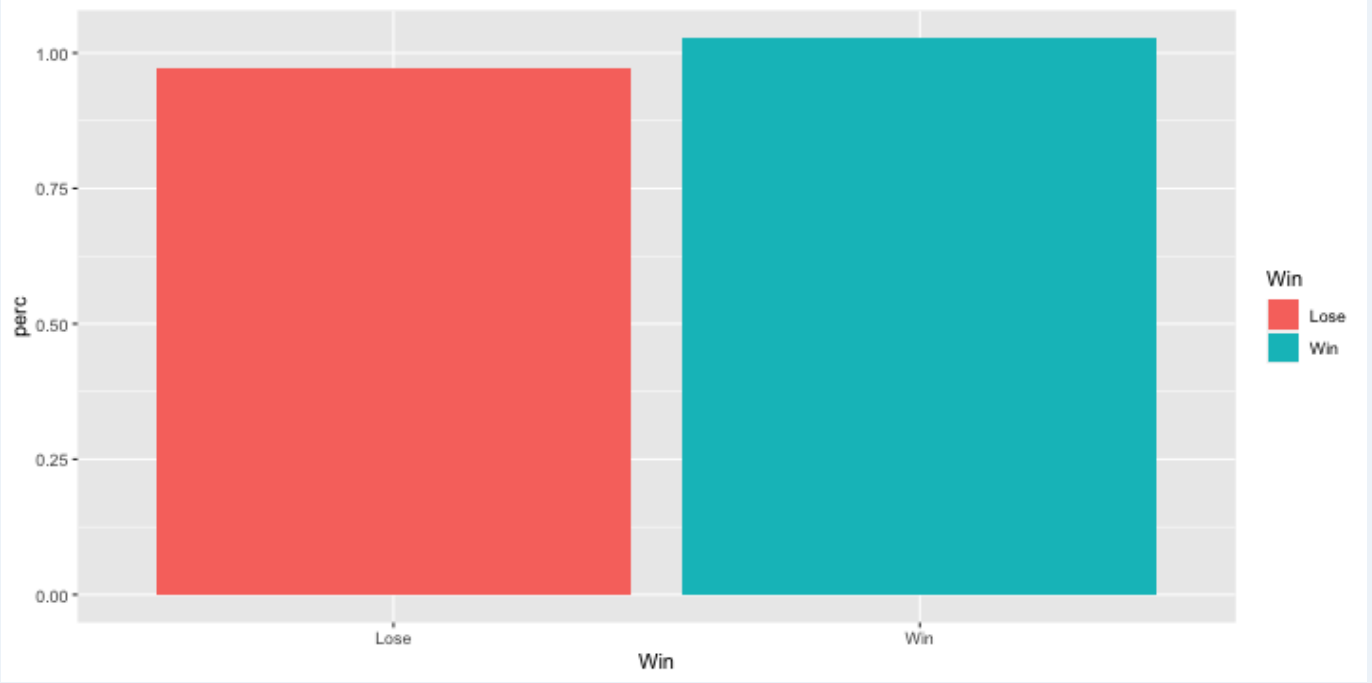
aes mapping (pronouns)

- x-axis: Outcome (win or lose)
- y-axis: % of outcomes in switch group
- Fill color: Outcome

In ggplot2 code

Use the `aes()` function inside `ggplot()`. (Can also use in `geom_col()`.)

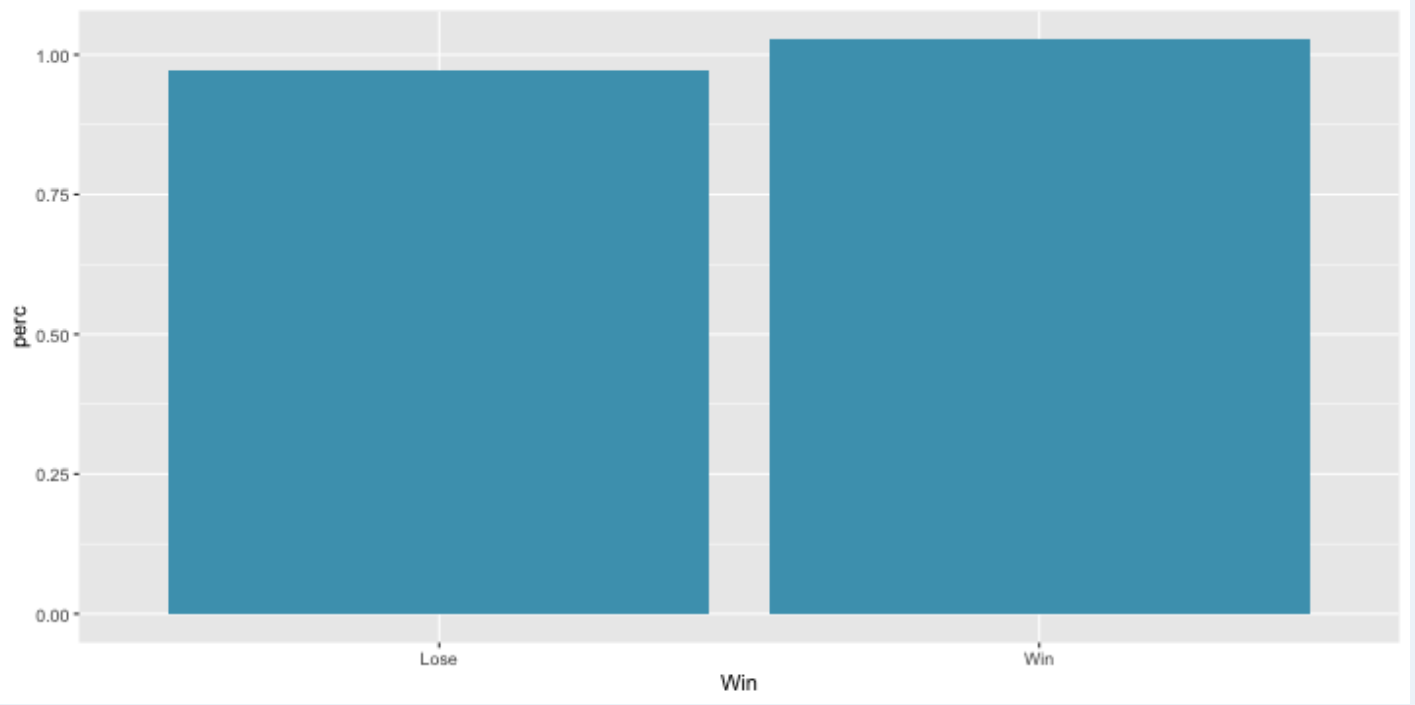
```
ggplot(data = mh_sims,
       aes(x = Win, y = perc, fill = Win)) +
  geom_col()
```



In ggplot2 code

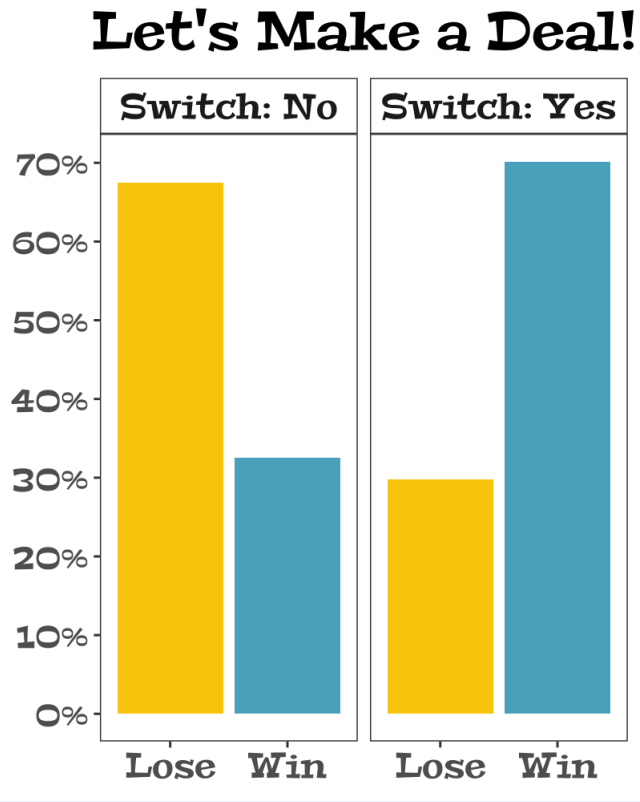
- Note: aesthetics / aes values do not have to be connected to data.
- To change an aes value for the entire plot, use the aes value *outside* of the `aes()` function.

```
ggplot(data = mh_sims, aes(x = Win, y = perc, fill = Win)) +
  geom_col(fill = "#4AA0BB")
```



Stat ↔ Adverb

Data Visualization (Sentence)



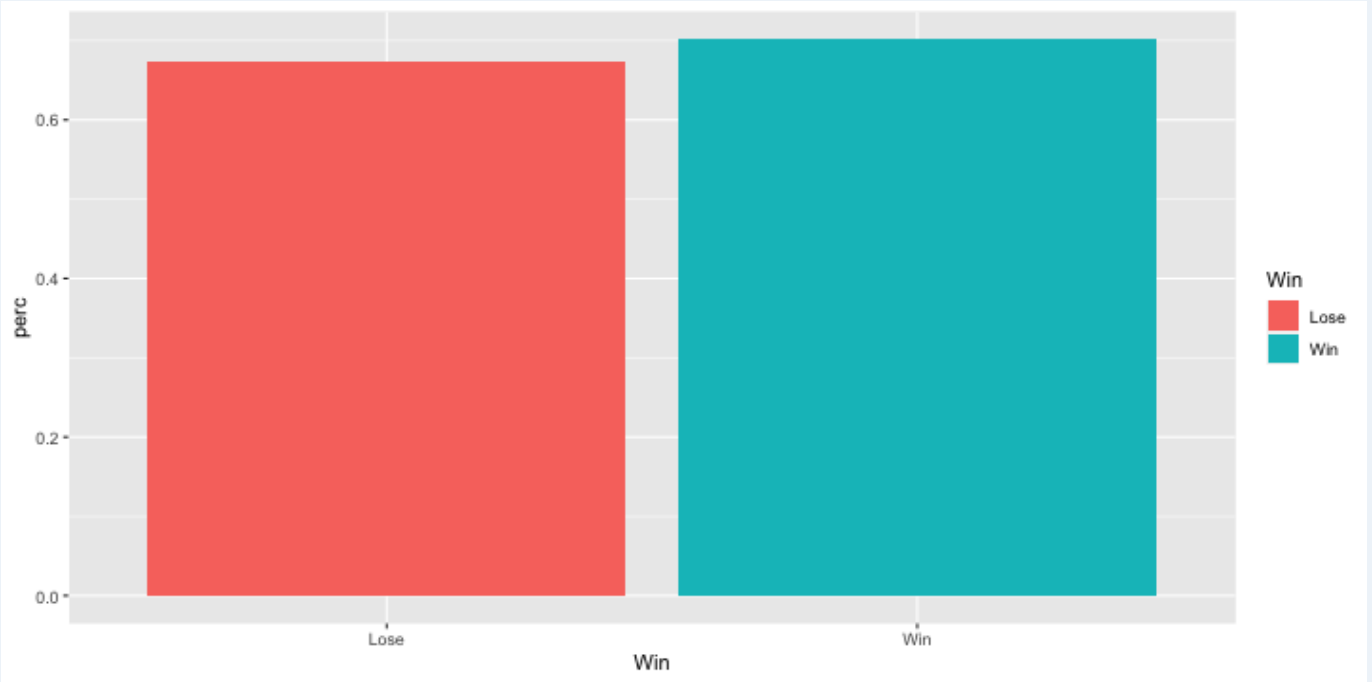
stat (adverb)

- Identity: The data are not altered in any way

Switch	Win	n	perc
No	Lose	29	0.6744186
No	Win	14	0.3255814
Yes	Lose	17	0.2982456
Yes	Win	40	0.7017544

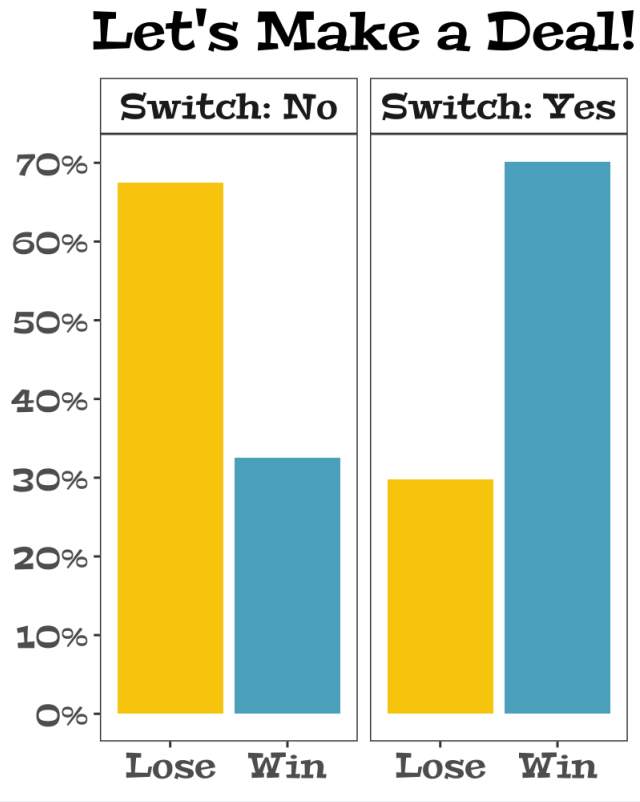
In ggplot2 code

```
ggplot(data = mh_sims,
       aes(x = Win, y = perc, fill = Win)) +
  stat_identity(geom="col")
```



Theme ↔ Adjective

Data Visualization (Sentence)



Theme (adjectives)

- white background
- no gridlines
- text font, size & face

In ggplot2 code

The `theme()` function can modify any non-data element of the plot.

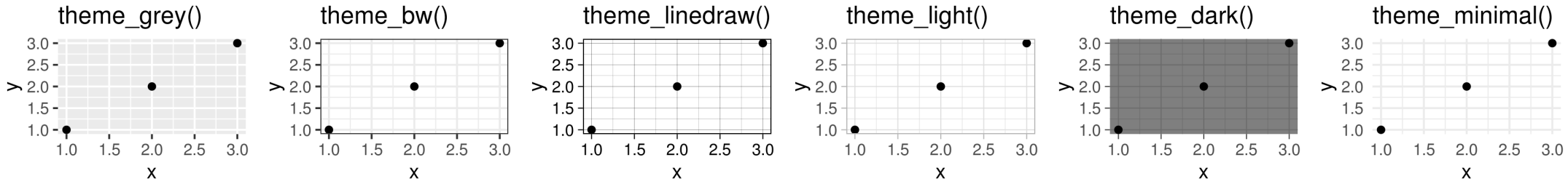
```
p ← ggplot(data = mh_sims,
       aes(x = Win, y = perc, fill = Win)) +
  geom_col()
```

📢 This is a common trick. Create an object `p` that is the `ggplot` to add to later on.

In ggplot2 code

Can use pre-made `theme_*()` functions and the `theme()` function to alter any non-data element of the plot.

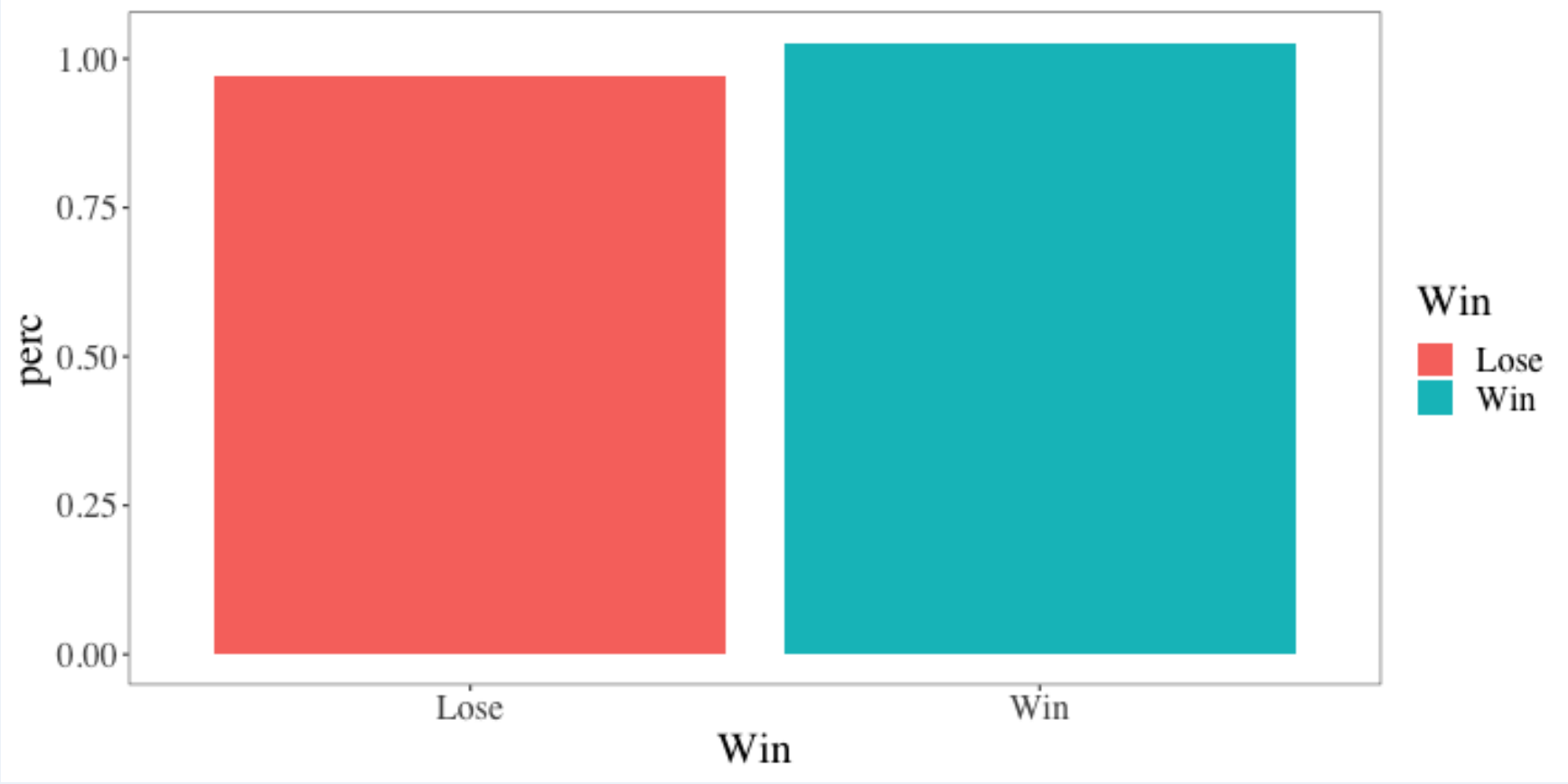
```
p2 ← p +
  theme_bw() +
  theme(text = element_text(family = "serif", size = 20),
        panel.grid = element_blank())
```



Source: ggplot2 book

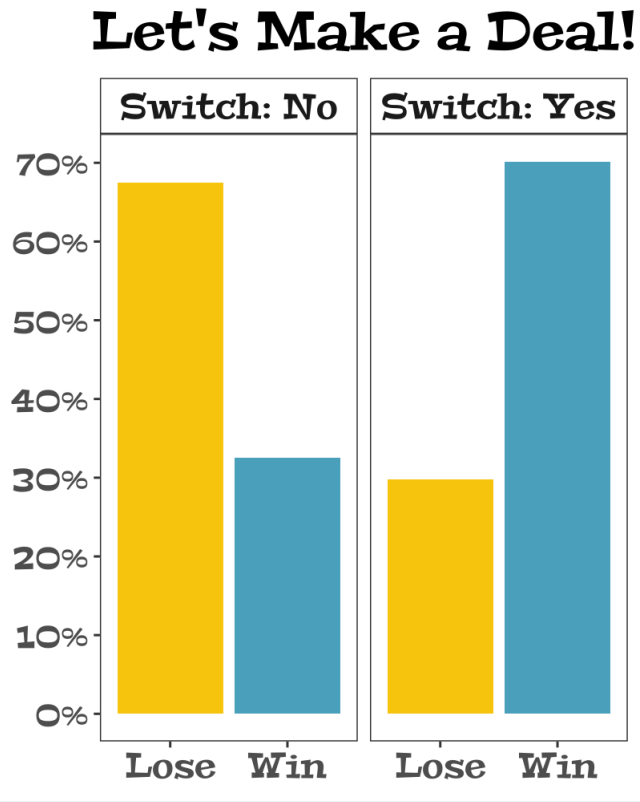
In ggplot2 code

```
p2
```



Guides ↔ Prepositions

Data Visualization (Sentence)



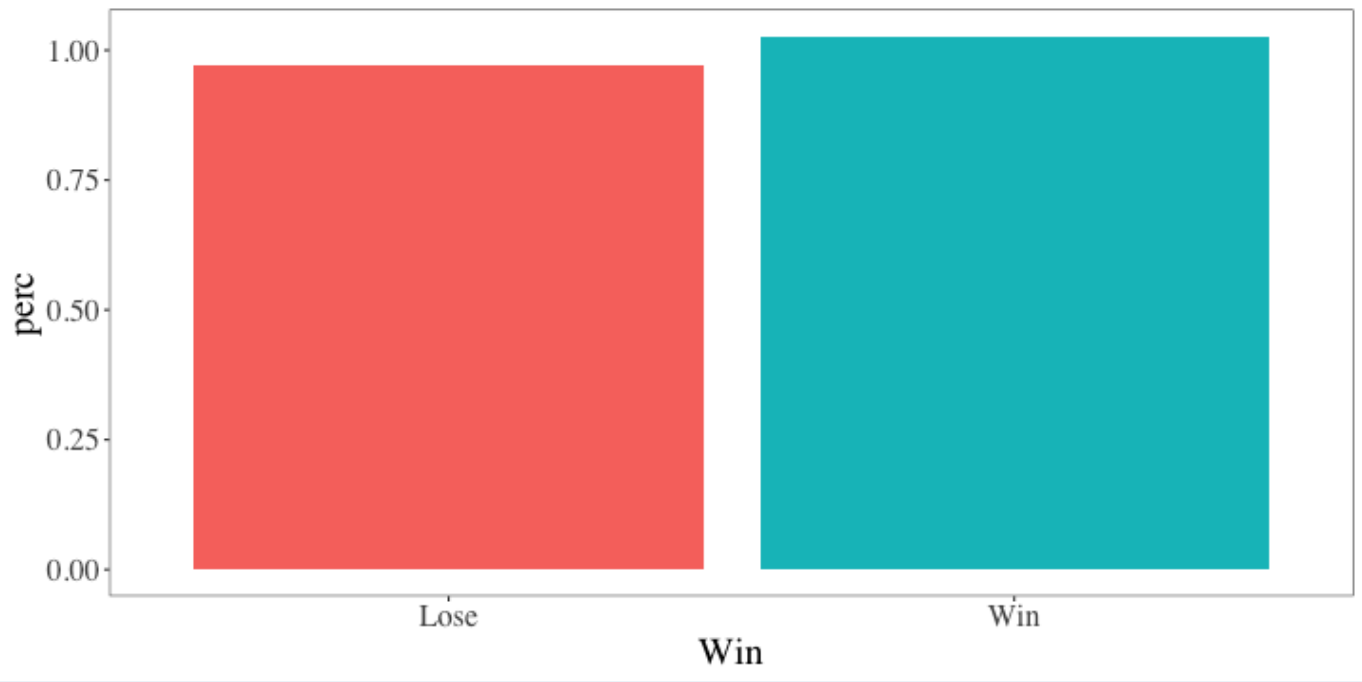
guide (preposition)

- There is no guide/legend.
- Not needed here, because color and x-axis are the same.

In ggplot2 code

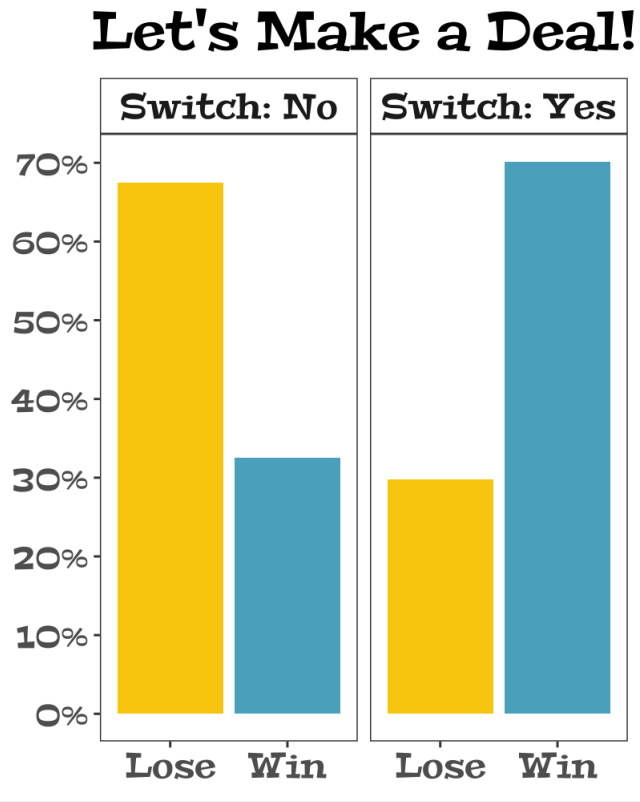
The `guides()` function controls all legends by connecting to the aes.

```
p3 ← p2 + guides(fill = "none")
p3
```



Facets ↔ Conjunctions

Data Visualization (Sentence)

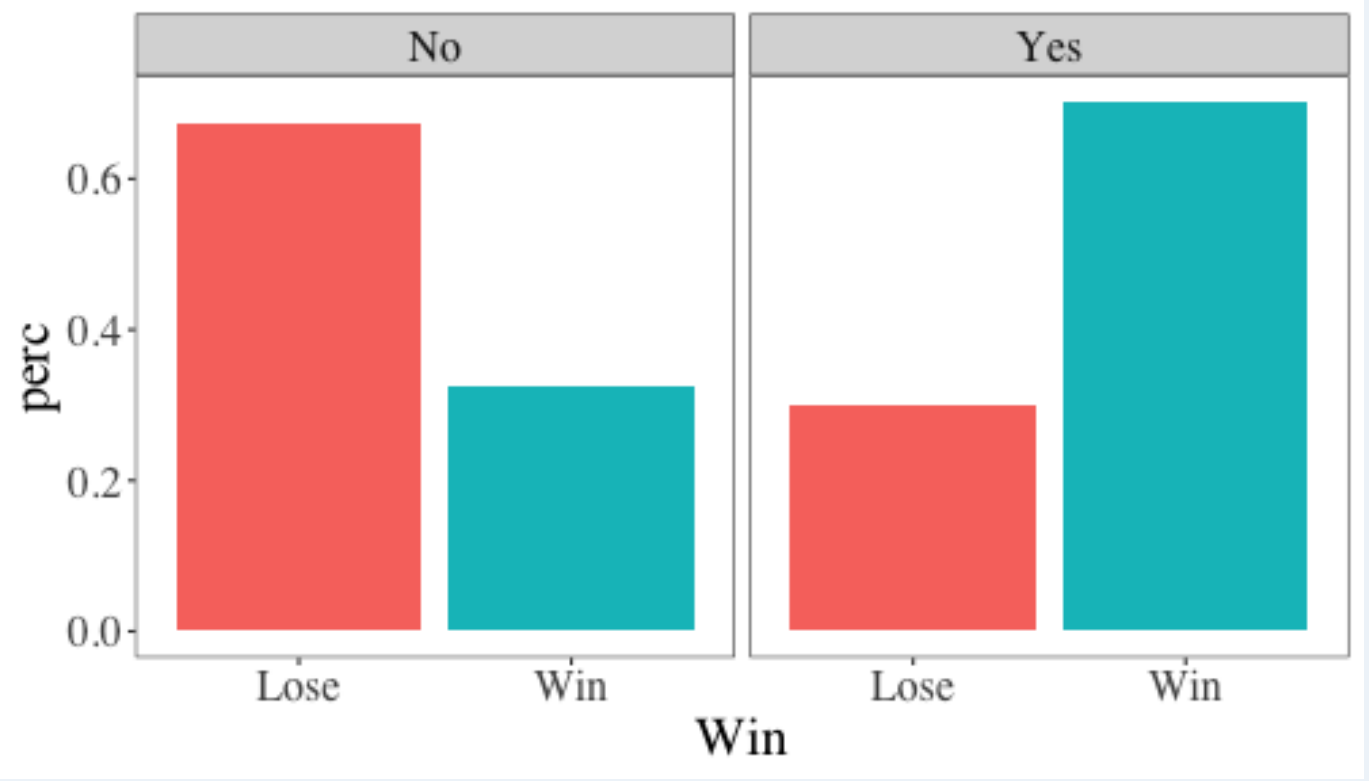


Facets (conjuction)

- Facet by the Switch variable

ggplot2 code

```
p4 ← p3 + facet_grid(cols =  vars(Switch))
p4
```



Other grammar elements

Scales: control how data are translated to visual properties (sentence structure)

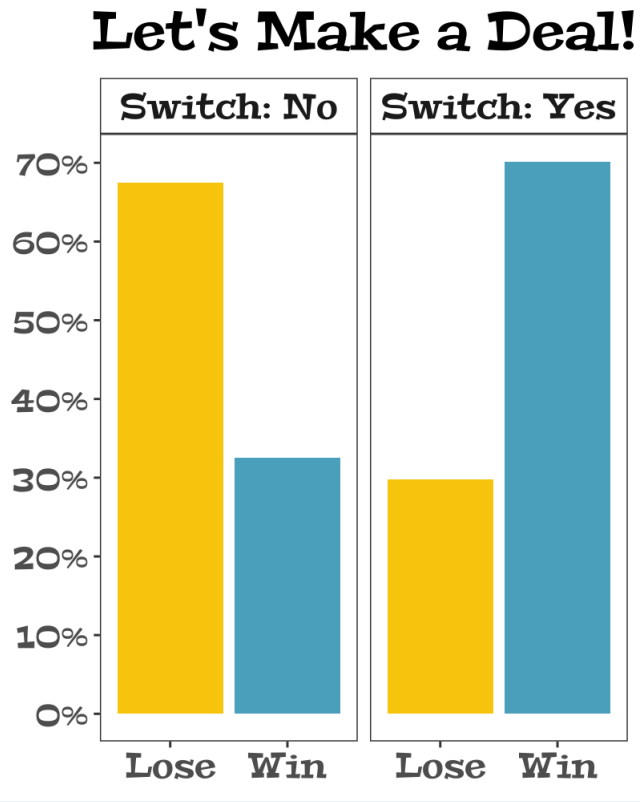
Coordinate system: how data are positioned in a 2D data visualization (verb tense)

Position: How to deal with overlap, if any (word order)

- "native speakers" don't have to think about these too much
- `ggplot2` has smart defaults here, less work for you
- Largely up to individual taste/style

Example

Data Visualization (Sentence)



Scales, coordinate systems, positions (sentence structure, verb tense, word order)

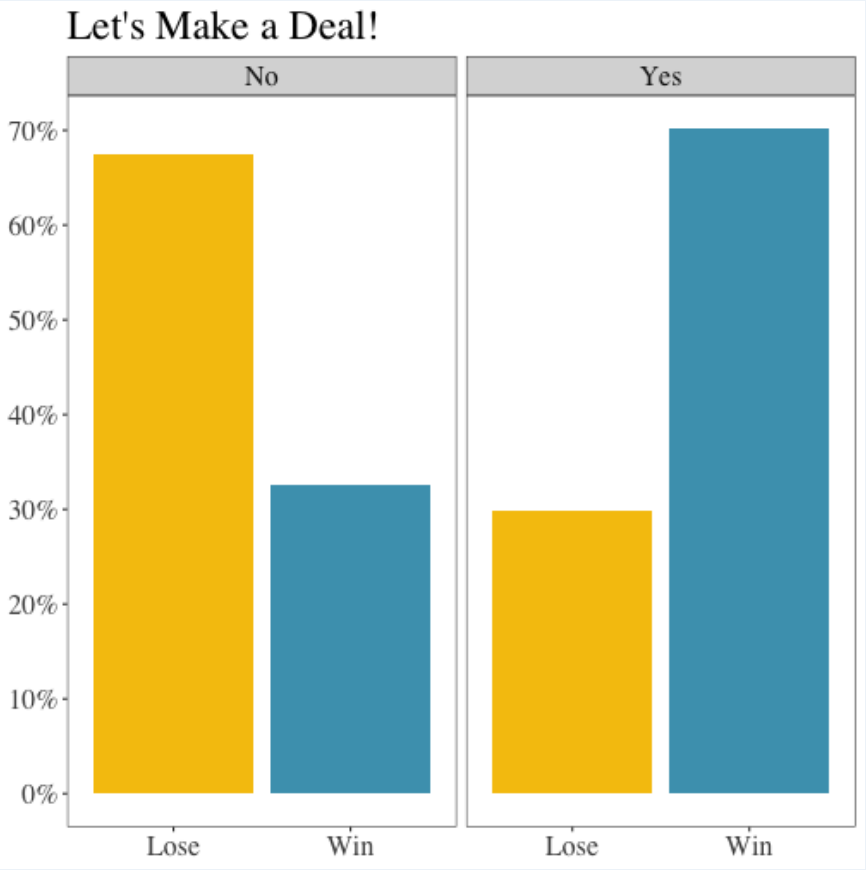
- Scale: y-axis labeled every 10%
- Scale: color of bars
- Scale: axes & plot titles
- Coordinate system: cartesian (automatic)
- Position: no position shift (identity)

ggplot2 code

There are no overlapping elements, so no position needed.

```
p4 +
  scale_y_continuous(name = NULL,
                     breaks = seq(0, .7, by = .1), label = scales::label_percent(accuracy = 1)) +
  scale_fill_manual(values =  c("#F6C40C", "#4AA0BB")) +
  labs(x = NULL, title = "Let's Make a Deal!")
```

Final Result (for now)



Complete ggplot2 code

```
ggplot(data = mh_sims,
       aes(x = Win, y = perc, fill = Win)) +
  geom_col() +
  facet_grid(cols =  vars(Switch)) +
  scale_y_continuous(name = NULL,
                     breaks = seq(0, .7, by = .1), label = scales::label_percent(accuracy = 1)) +
  scale_fill_manual(values =  c("#F6C40C", "#4AA0BB")) +
  labs(x = NULL, title = "Let's Make a Deal!") +
  theme_bw() +
  theme(text = element_text(family = "serif", size = 20),
        panel.grid = element_blank()) +
  guides(fill = "none")
```

Complete ggplot2 code

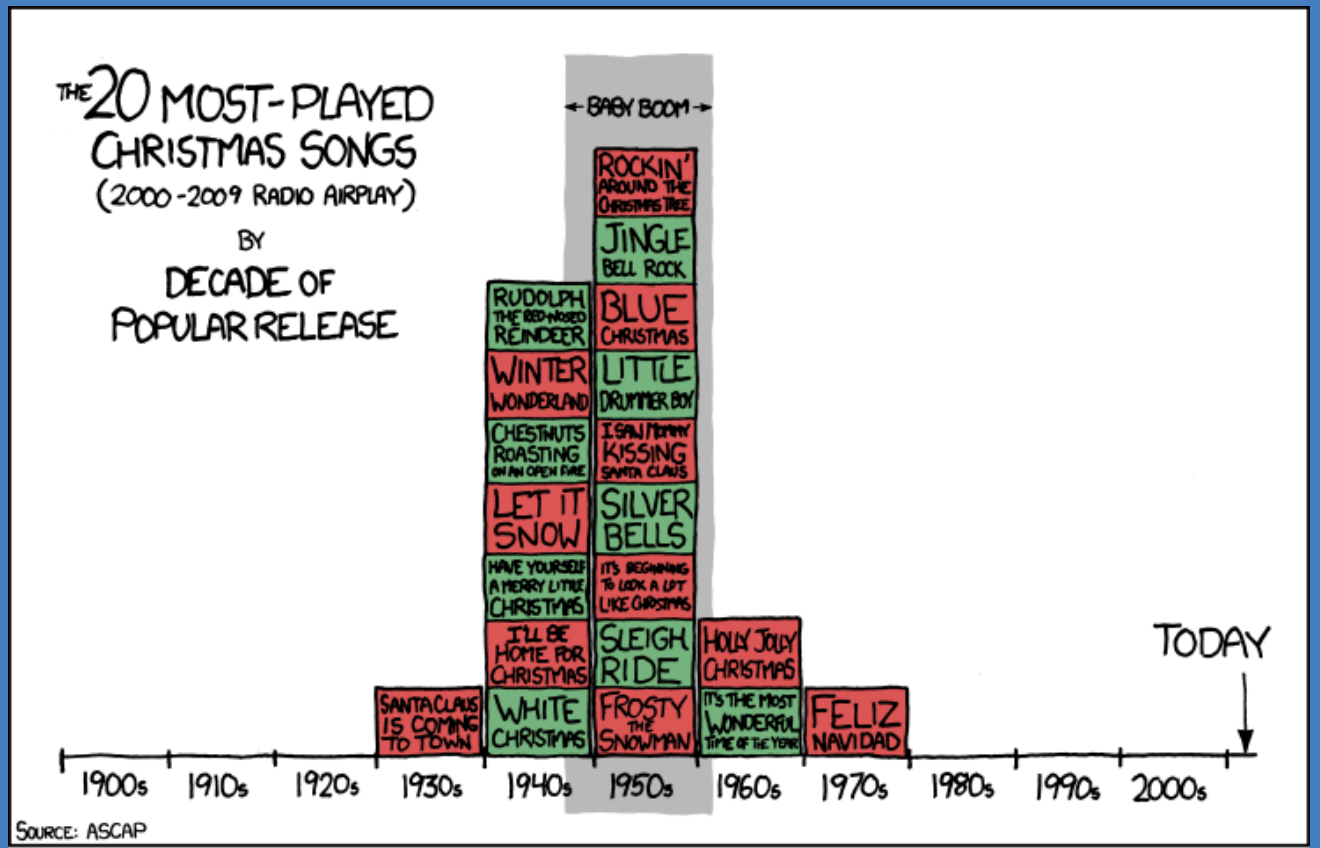
```
ggplot(data = mh_sims,
       aes(x = Win, y = perc, fill = Win)) +
  geom_col() +
  facet_grid(cols =  vars(Switch)) +
  scale_y_continuous(name = NULL,
                     breaks = seq(0, .7, by = .1), label = scales::label_percent(accuracy = 1)) +
  scale_fill_manual(values =  c("#F6C40C", "#4AA0BB")) +
  labs(x = NULL, title = "Let's Make a Deal!") +
  theme_bw() +
  theme(text = element_text(family = "serif", size = 20),
        panel.grid = element_blank()) +
  guides(fill = "none")
```

What's next?

More and more detail on the `ggplot2` universe:

- `geom`s for one- and two-variable visualization
- Including three or more variables in the visualization
  - Colors, sizes, shapes, linetypes
  - Grouping & faceting
  - Maps

One-variable visualization



EVERY YEAR, AMERICAN CULTURE EMBARKS ON A MASSIVE PROJECT TO CAREFULLY RECREATE THE CHRISTMASES OF BABY BOOMERS' CHILDHOODS.

xkcd.com/988/

Histogram

> A **histogram** approximates the distribution of a single numeric variable. It shows frequency of values in specified ranges.

`geom_histogram`

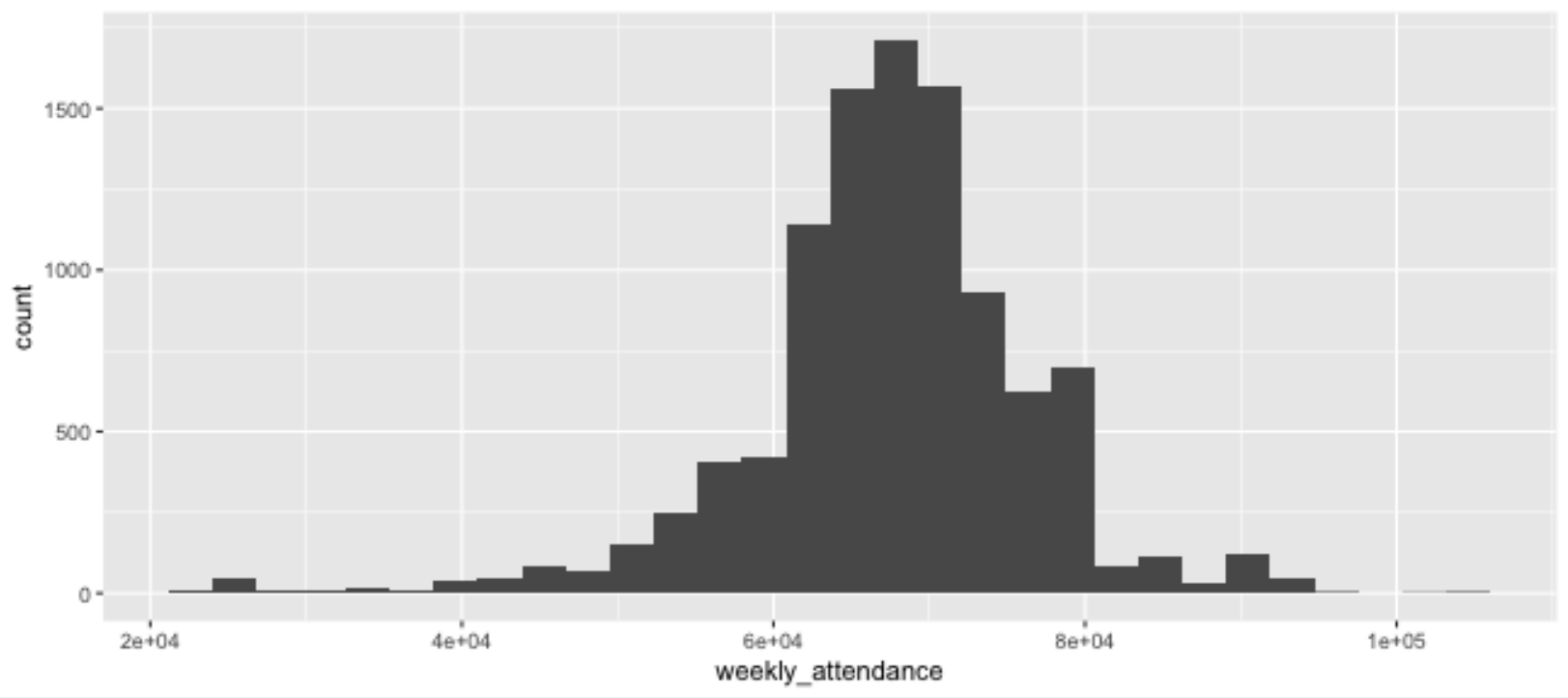
- requires the `x` aesthetic inside `aes()`
- Specify width of bars with the `bins` or `binwidth` argument
- Can change appearance of the bars with `color`, `fill`, `alpha` arguments

Your Turn

02:00

Complete the code below to recreate the histogram of weekly NFL attendance. (Source: Tidy Tuesday)

```
library(readr)
attendance ← read_csv("dat/nfl_attendance.csv")
ggplot(data = ???, aes(x = ???)) +
  geom_???()
```



Bar chart

> A **bar chart** displays counts of a categorical variable, and is the categorical equivalent of the histogram.

`geom_bar`

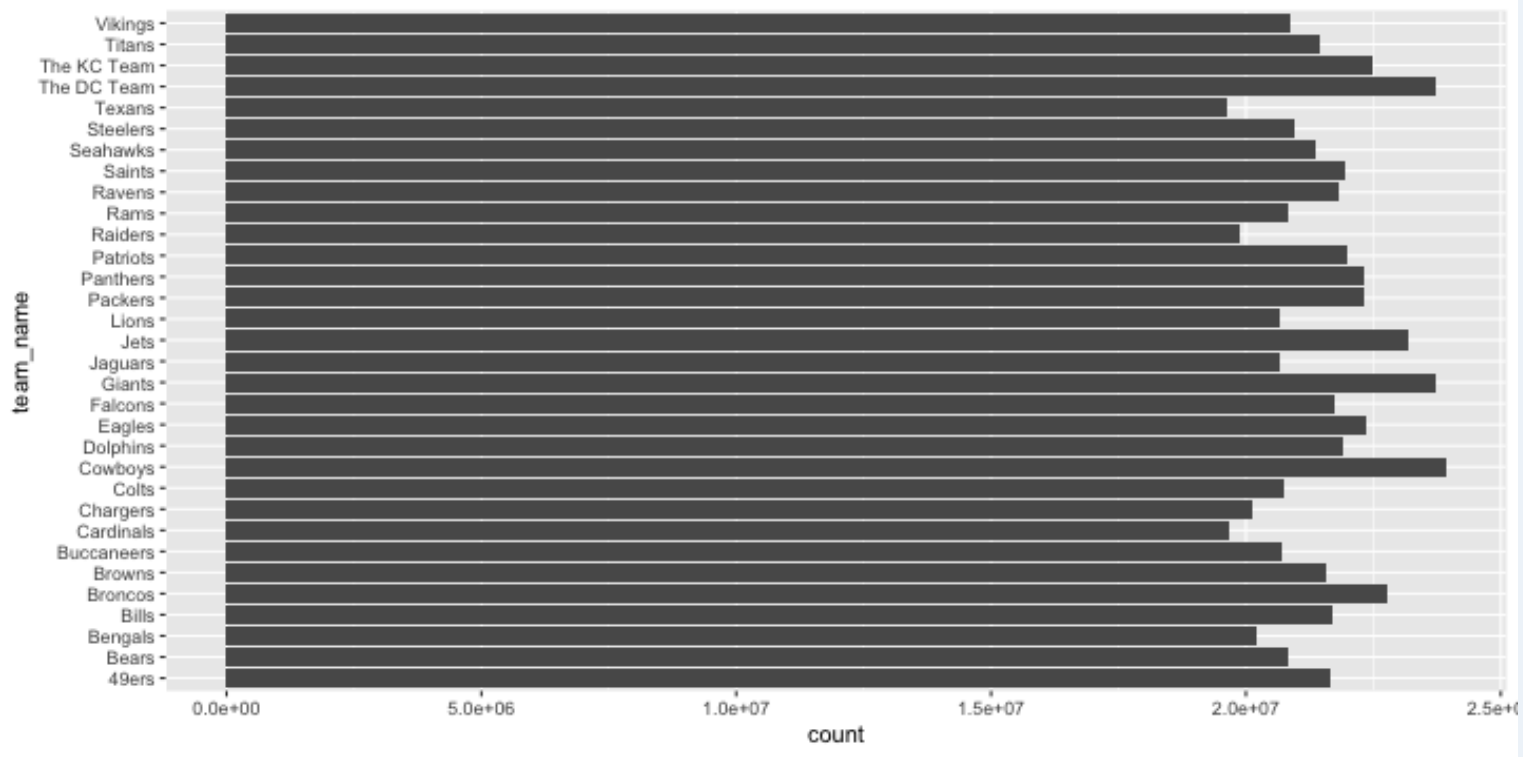
- requires the `x` aesthetic inside `aes()`
- Can change appearance of the bars with `color`, `fill`, `alpha` arguments

Your Turn

02:00

Complete the code below to display a bar chart of total NFL attendance from 2000-2019 by team.

```
ggplot(??? = attendance, ???(??? = ???, weight = weekly_attendance)) +
  geom_bar() +
  coord_flip()
```



Density plot

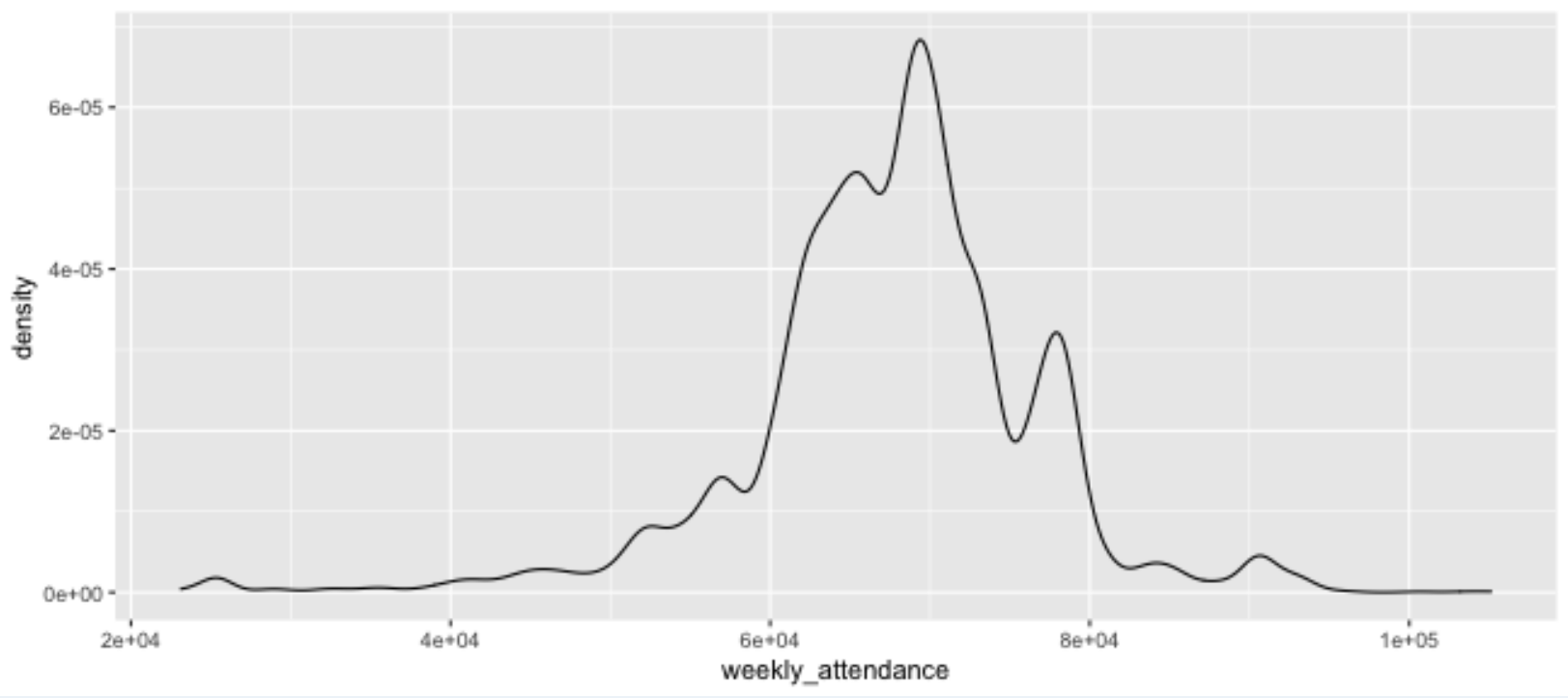
> A density estimate is a smoothed version of the histogram which is especially useful if the data come from a continuous distribution.

`geom_density`

- requires the `x` aesthetic inside `aes()`
- change the underlying kernel smoother with the `kernel` parameter
- change bandwidth with the `adjust` parameter
- Can change appearance of the bars with `color`, `fill`, `alpha` arguments

Example

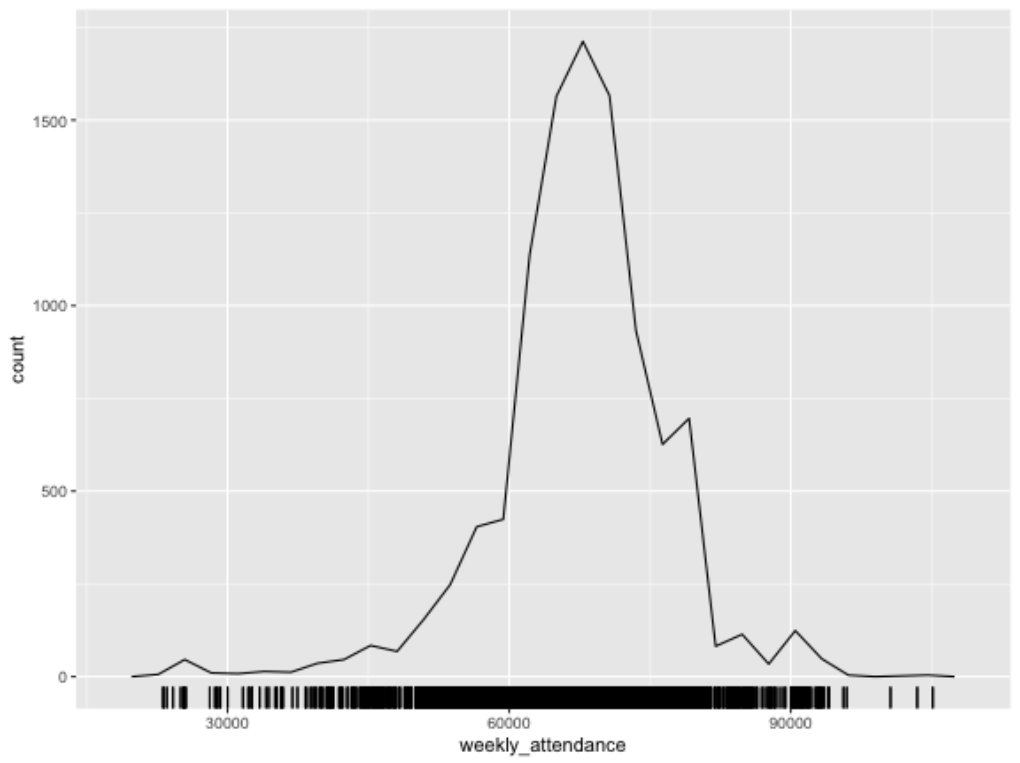
```
ggplot(data = attendance, aes(x = weekly_attendance)) +
  geom_density()
```



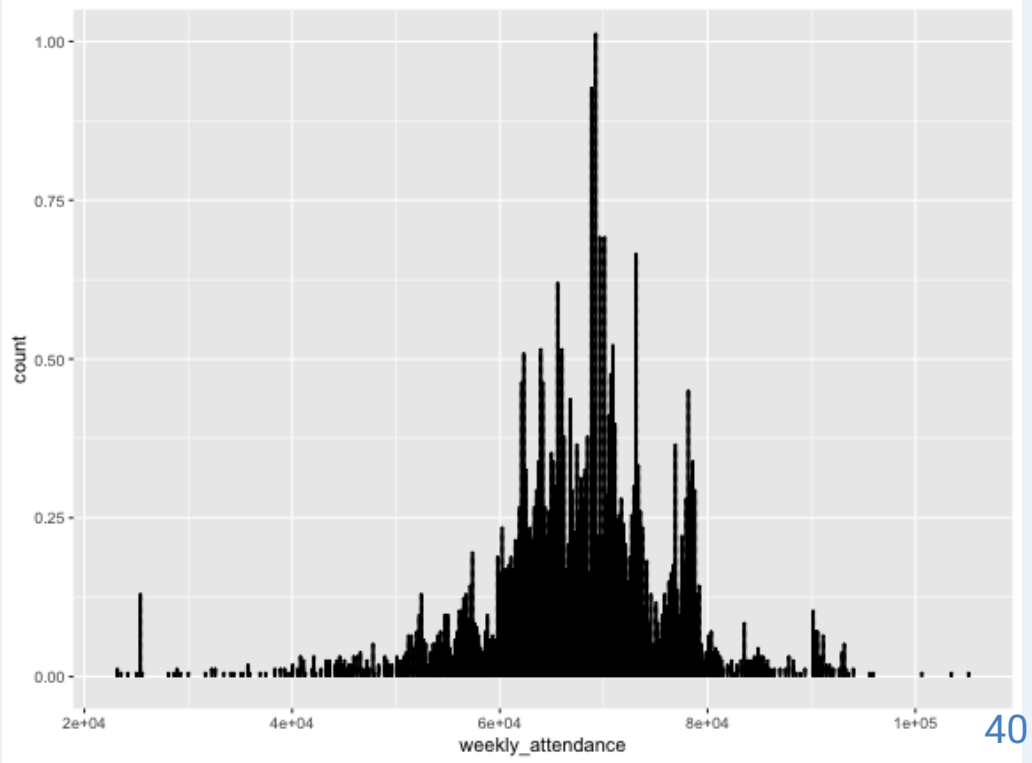
Other one-variable viz

- `geom_freqpoly`: behaves the same as `geom_histogram` but with connected lines instead of bars
- `geom_dotplot`: show values in bins as individual dots
- `geom_rug`: place lines along an axis for each observation

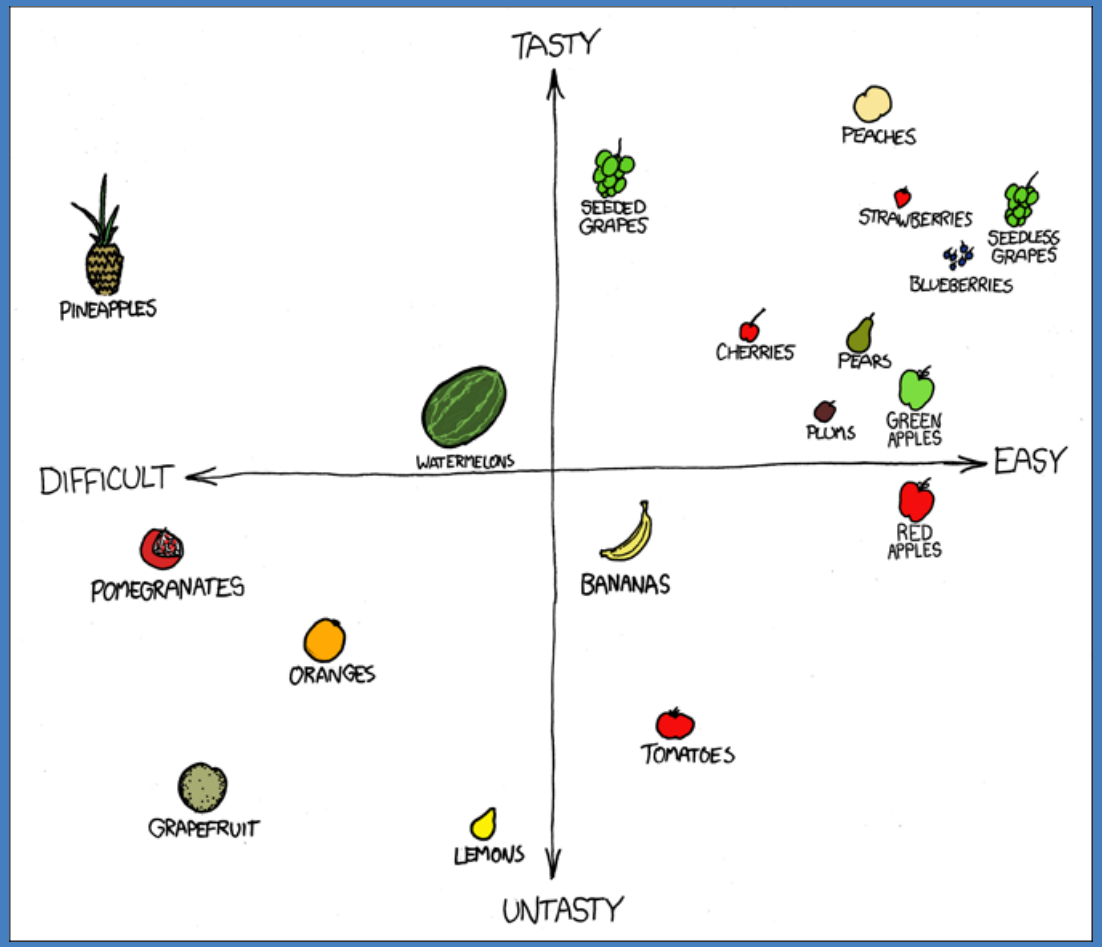
`geom_freqpoly` + `geom_rug`:



`geom_dotplot`:

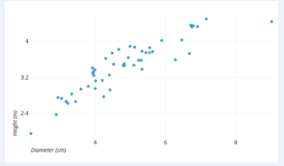
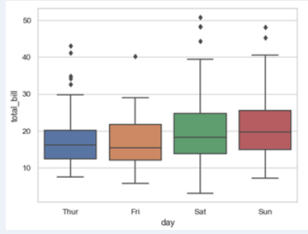
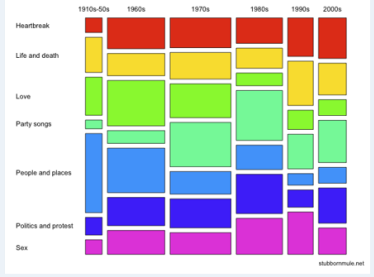


Two-variable visualization



xkcd.com/388

Which variables?

	Numerical	Categorical
Numerical	 • Scatter plot (point) • 2D binning (bin2d, hex) • Contour plot (density2d) • Quantiles (quantile, qq) • Lines (line, smooth) • Ribbons (ribbon, area)	 • Boxplot (boxplot, violin) • Counts (count, tile) • Error bars (errorbar) • Columns (col)
Categorical	***Which ggplot2 data viz is right for your data?*** (geoms in parentheses)	 • Mosaic (ggmosaic::geom_mosaic) • Counts (count, tile)

Two numeric variables

time, weekly_attendance

Using data from The DC Team.

```
library(dplyr)
dc ← attendance %>% filter(team_name == "The DC Team")
```

geom_line

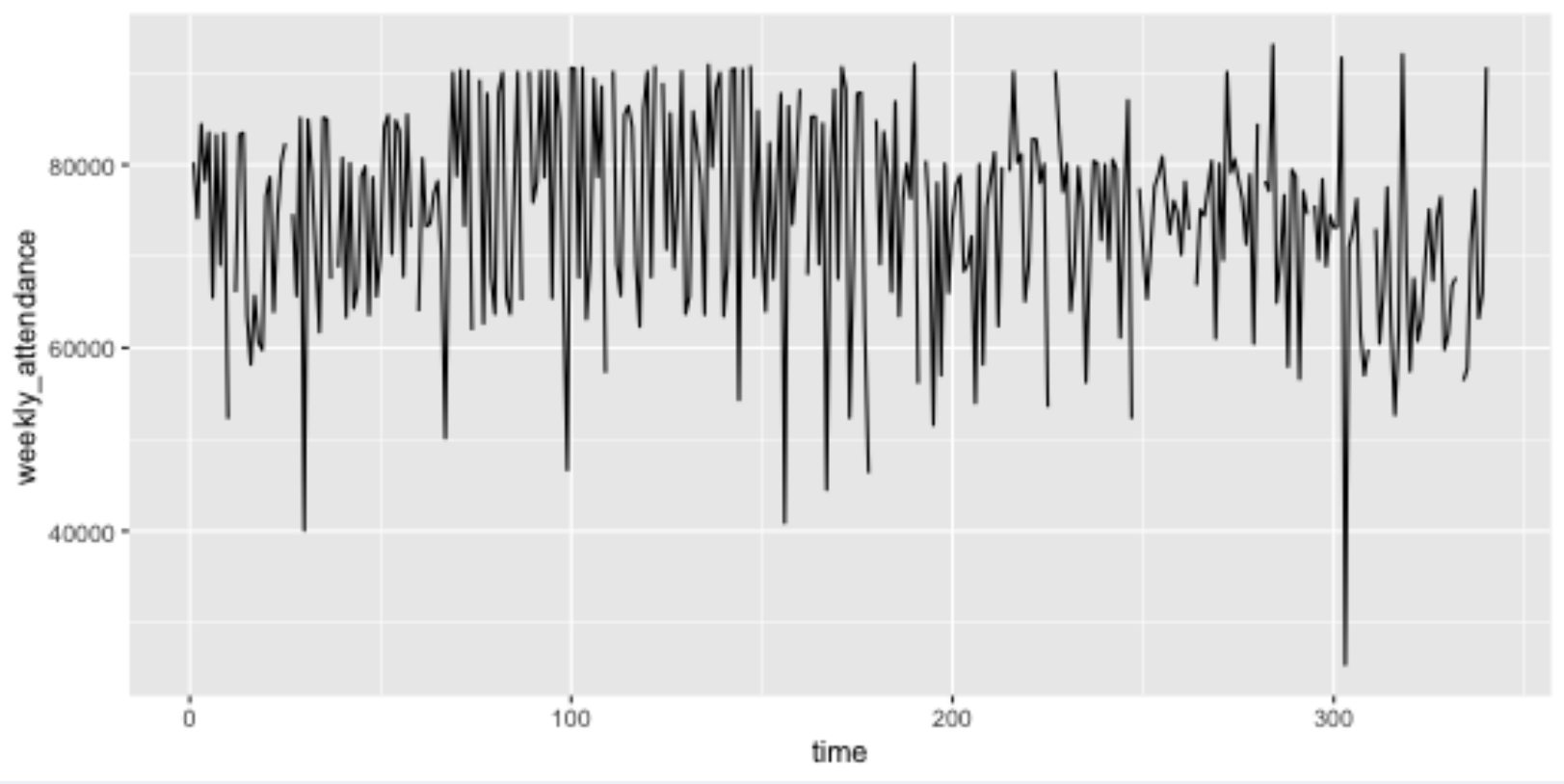
- requires the `x, y` aesthetics inside `aes()`
- Can change appearance of the lines with `color`, `linetype`, `alpha` arguments
- The `group aes` draws lines according to a grouping variable (later)

Your Turn

02:00

Complete the code to make the time series chart of weekly attendance from 2000-2019 for the DC Team.

```
???(data = ???, aes(x = ???, y = ???)) +
  geom_???()
```



One numeric, one categorical

`team, weekly_attendance`

Return to the full `attendance` dataset.

`geom_boxplot`

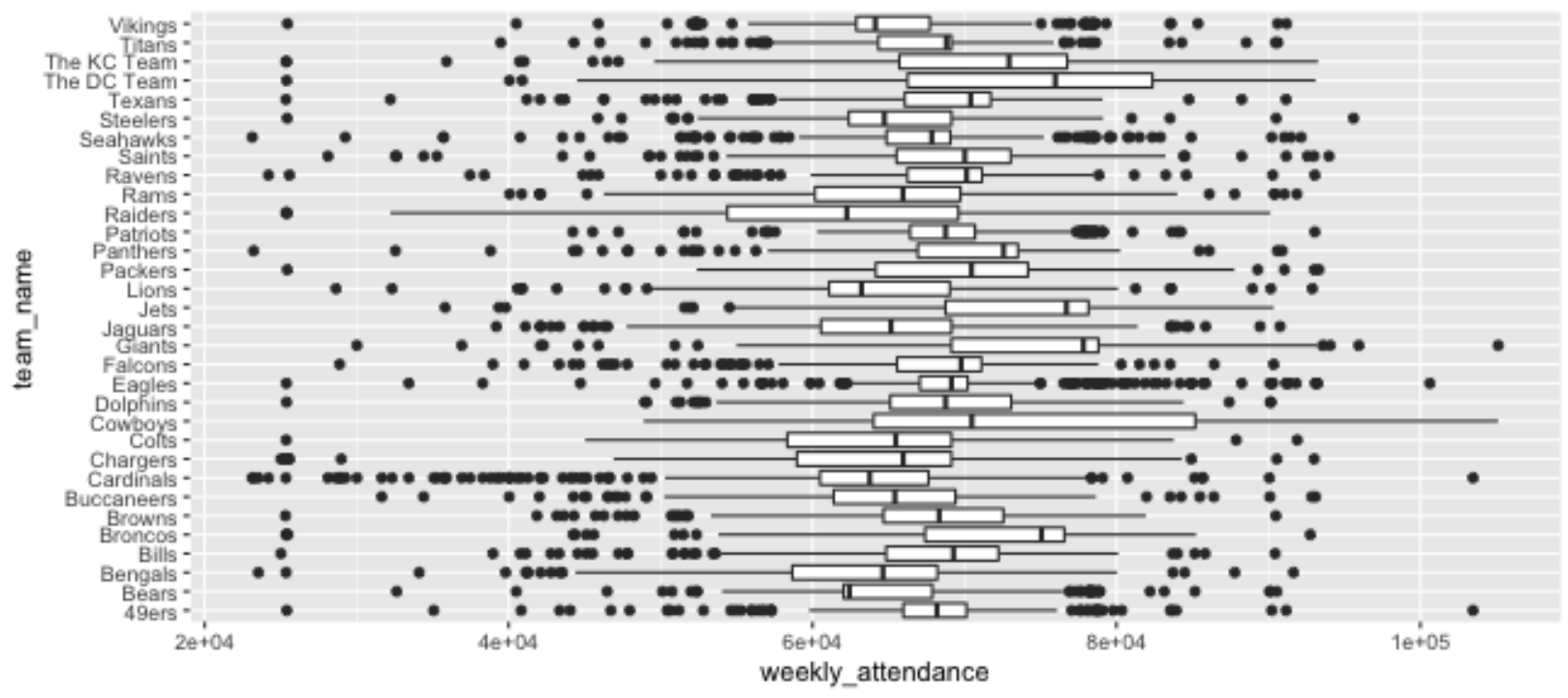
- requires the `x,y` aesthetics inside `aes()`
- Can change appearance of the boxes with `color`, `fill`, `alpha` arguments

Your Turn

02:00

Modify the code below to create box plots for weekly attendance by team name.

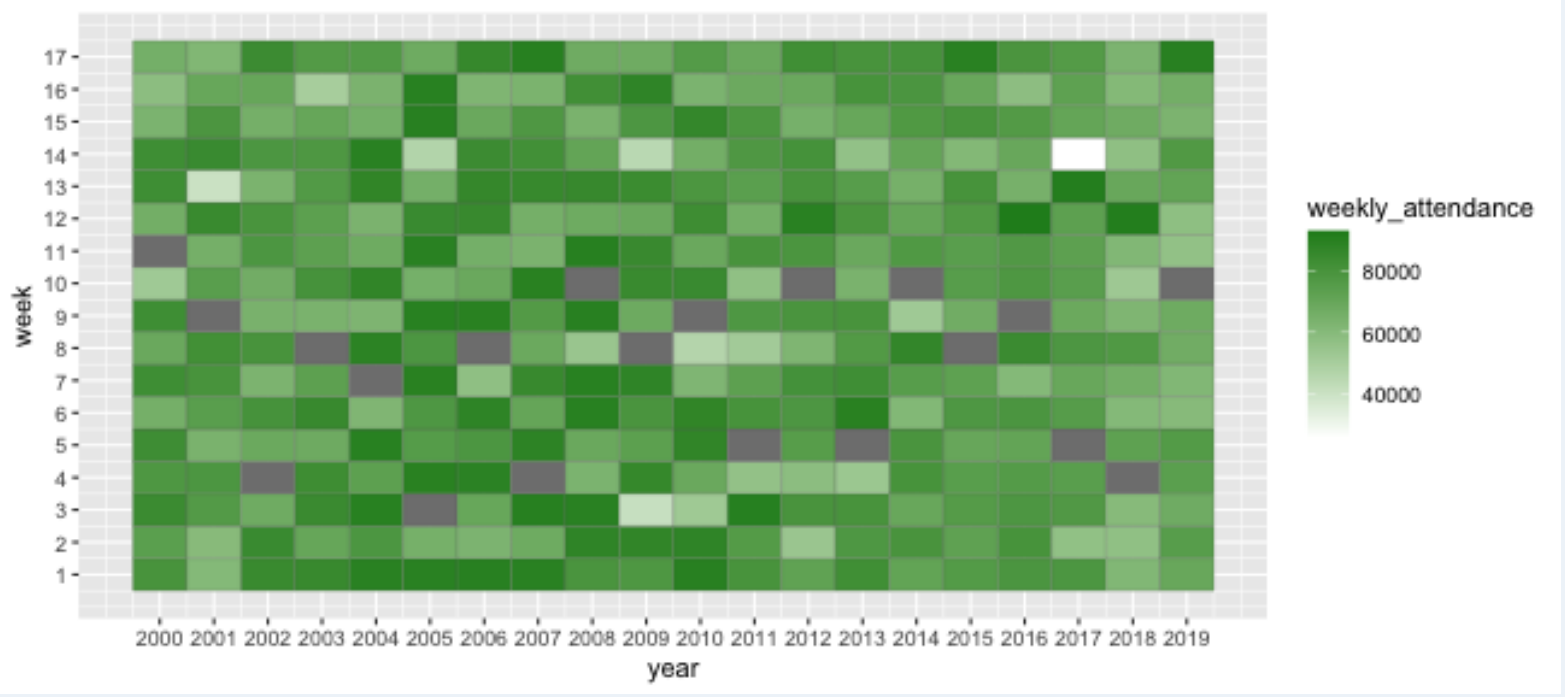
```
ggplot(data = ???, aes( ??? , ???)) +
  geom_???() +
  coord_flip()
```



Two categorical variables

year, week

```
ggplot(data = dc, aes(x = year, y = week, fill = weekly_attendance)) +
  geom_tile(color = "grey40") +
  scale_x_continuous(breaks = 2000:2019) + scale_y_continuous(breaks = 1:17) +
  scale_fill_gradient(low = "white", high = "forestgreen")
```



Three or more variable visualization

xkcd.com/880

The limits of `ggplot2`

`ggplot2` does not do 3D visualization

- (there are ways...but you won't see them here.)

`ggplot2` is not interactive

- See the `plotly` package and its `ggplotly()` function.

But, there are plenty of other ways to make informative graphics with `ggplot2`

Additional `aes()` mappings

Add a third variable to a plot with:

color : `color, fill`

size: `size, stroke`

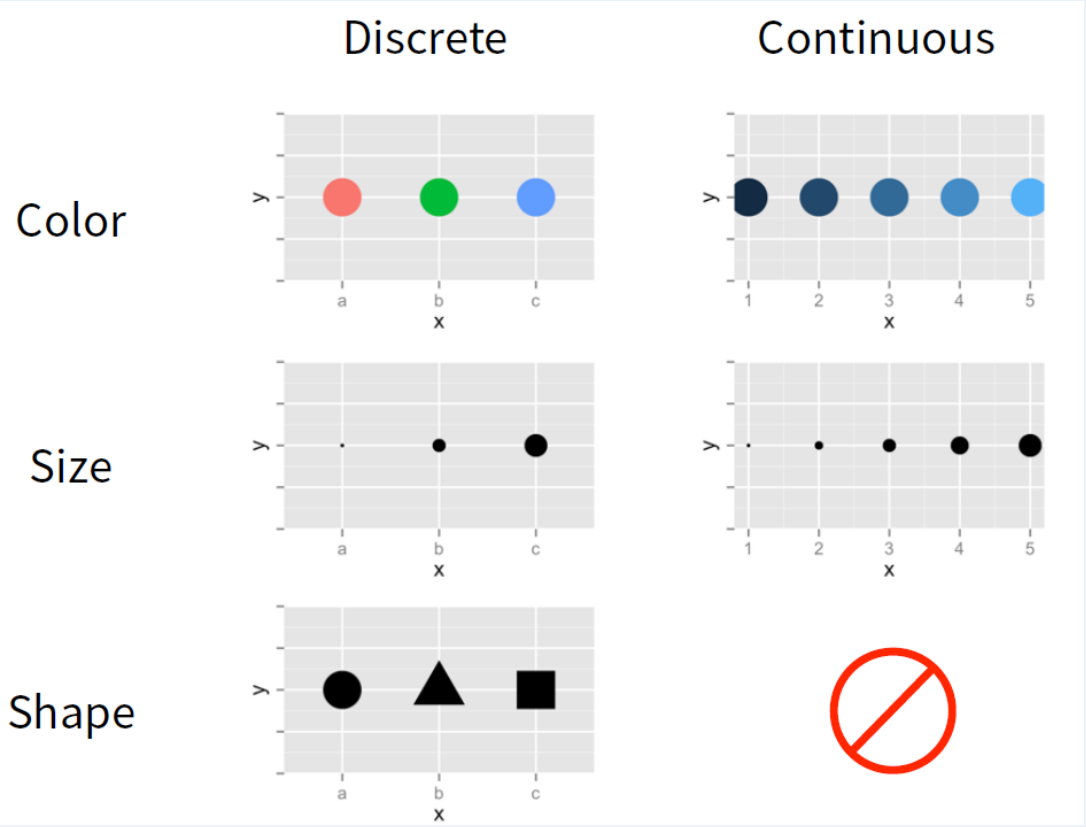
shape: `shape, linetype`

contour lines: `geom_contour(), geom_density_2d(), z`

bins: `geom_bin2d(), geom_hex(), geom_tile()`

Choosing a mapping

Image from Garrett Grolemund's 2019 JSM tidyverse tutorial.



Add a discrete/categorical variable

New York has two teams: the Giants and the Jets.

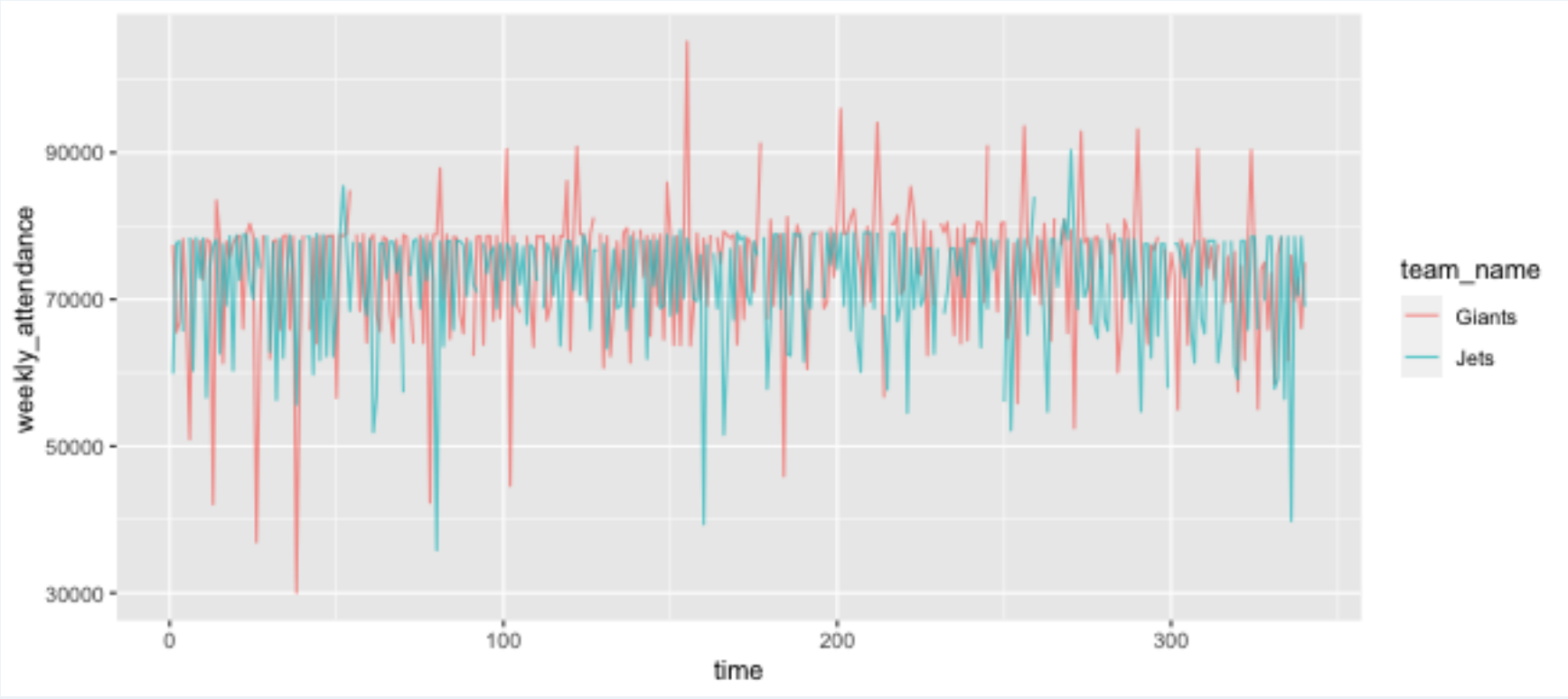
```
ny ← attendance %>% filter(team == "New York")
glimpse(ny)
```

```
## Rows: 680
## Columns: 11
## $ team              <chr> "New York", "New York", "New York", "New York", "New York", …
## $ team_name         <chr> "Giants", "Giants", "Giants", "Giants", "Giants", "Giants", …
## $ year              <dbl> 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, …
## $ total             <dbl> 1135455, 1135455, 1135455, 1135455, 1135455, 1135455, 113545…
## $ home              <dbl> 624085, 624085, 624085, 624085, 624085, 624085, 624085, 6240…
## $ away              <dbl> 511370, 511370, 511370, 511370, 511370, 511370, 511370, 5113…
## $ week              <dbl> 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 1…
## $ weekly_attendance <dbl> 77434, 65530, 66944, 78216, 68341, 50947, 78189, NA, 78087, …
## $ time              <dbl> 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 1…
## $ conference        <chr> "NFC", "NFC", "NFC", "NFC", "NFC", "NFC", "NFC", "NFC", "NFC…
## $ division          <chr> "East", "East", "East", "East", "East", "East", "East", "Eas…
```

Two time series, one plot

Use color to indicate team

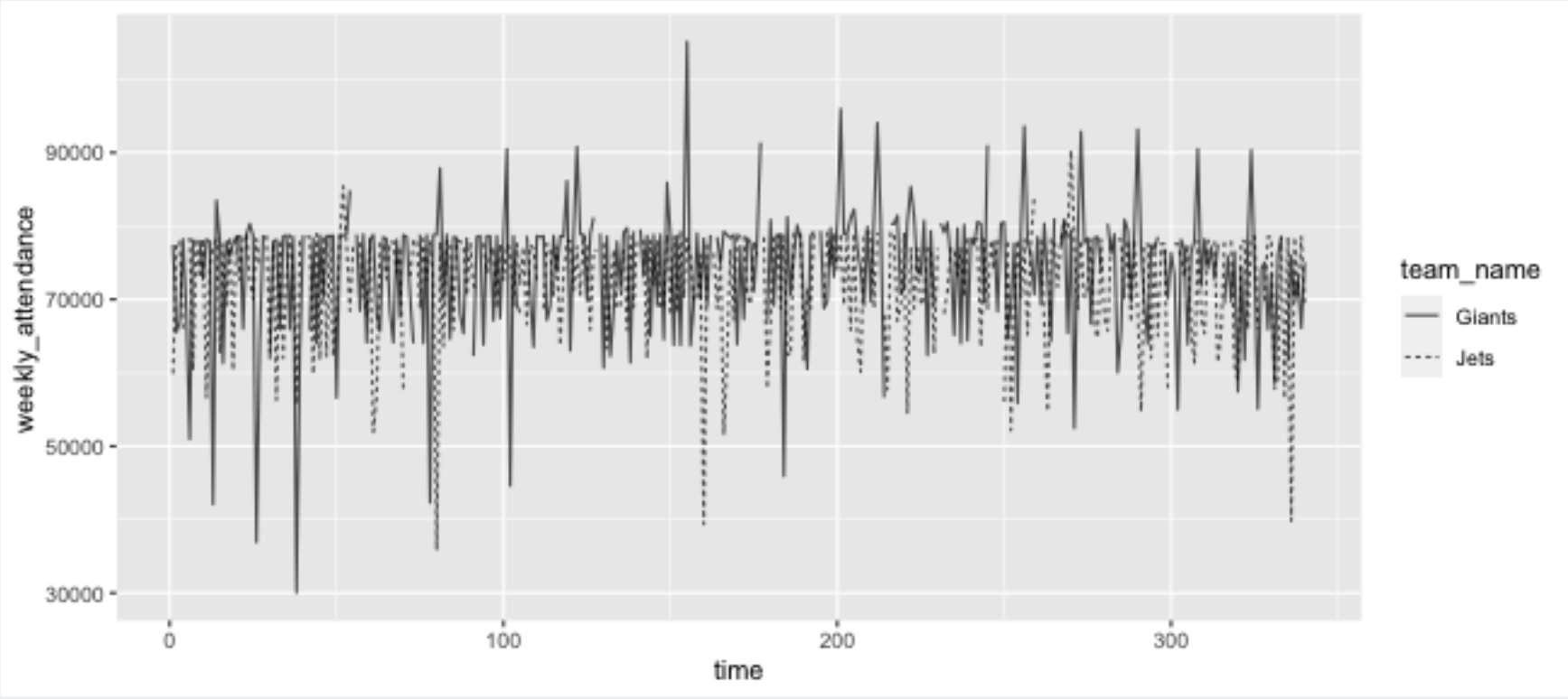
```
ggplot(data = ny, aes(x = time, y = weekly_attendance, color =team_name)) +
  geom_line(alpha = .7)
```



Two time series, one plot

Use linetype to indicate team

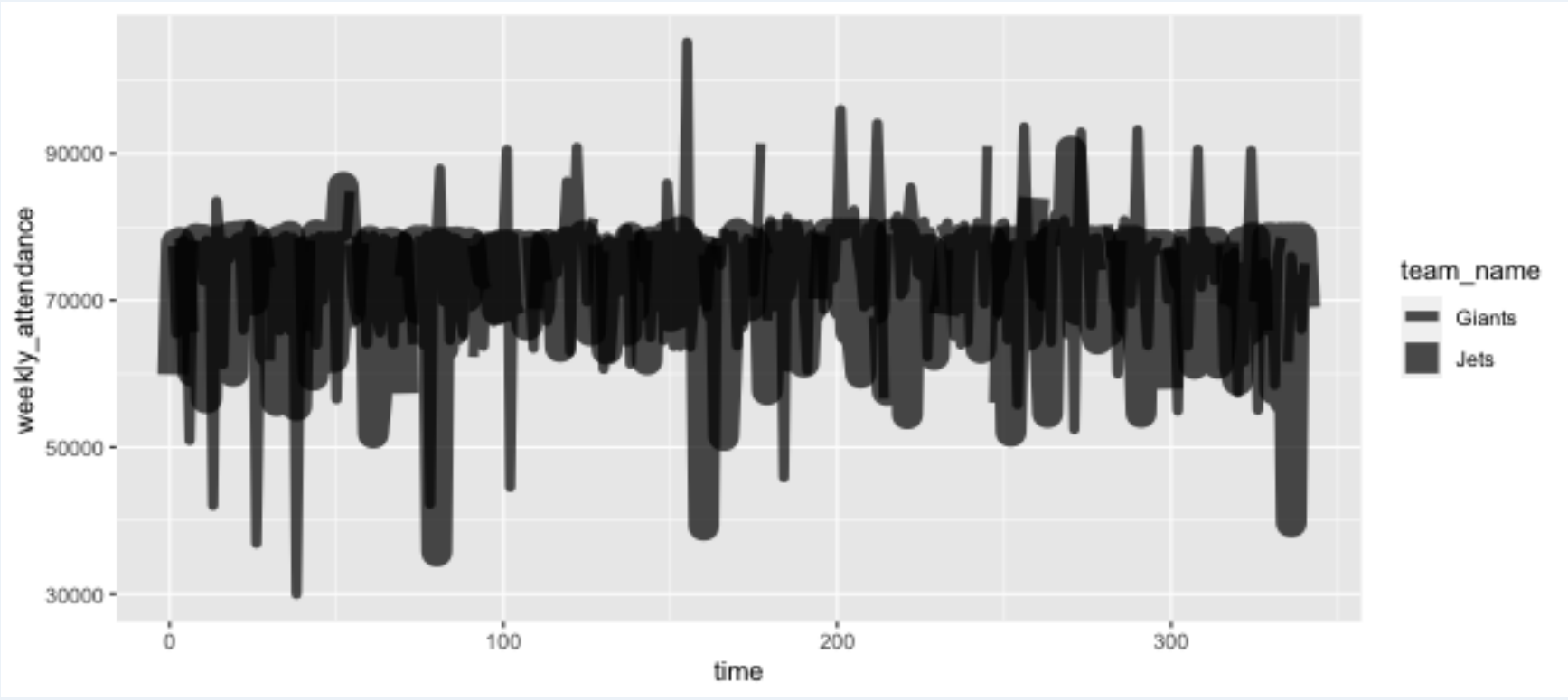
```
ggplot(data = ny, aes(x = time, y = weekly_attendance, linetype = team_name)) +
  geom_line(alpha = .7)
```



Two time series, one plot

Use size to indicate team. 📢 Note: I do not recommended this! Why?

```
ggplot(data = ny, aes(x = time, y = weekly_attendance, size =team_name)) +
  geom_line(alpha = .7)
```



Adding a continuous/numeric variable

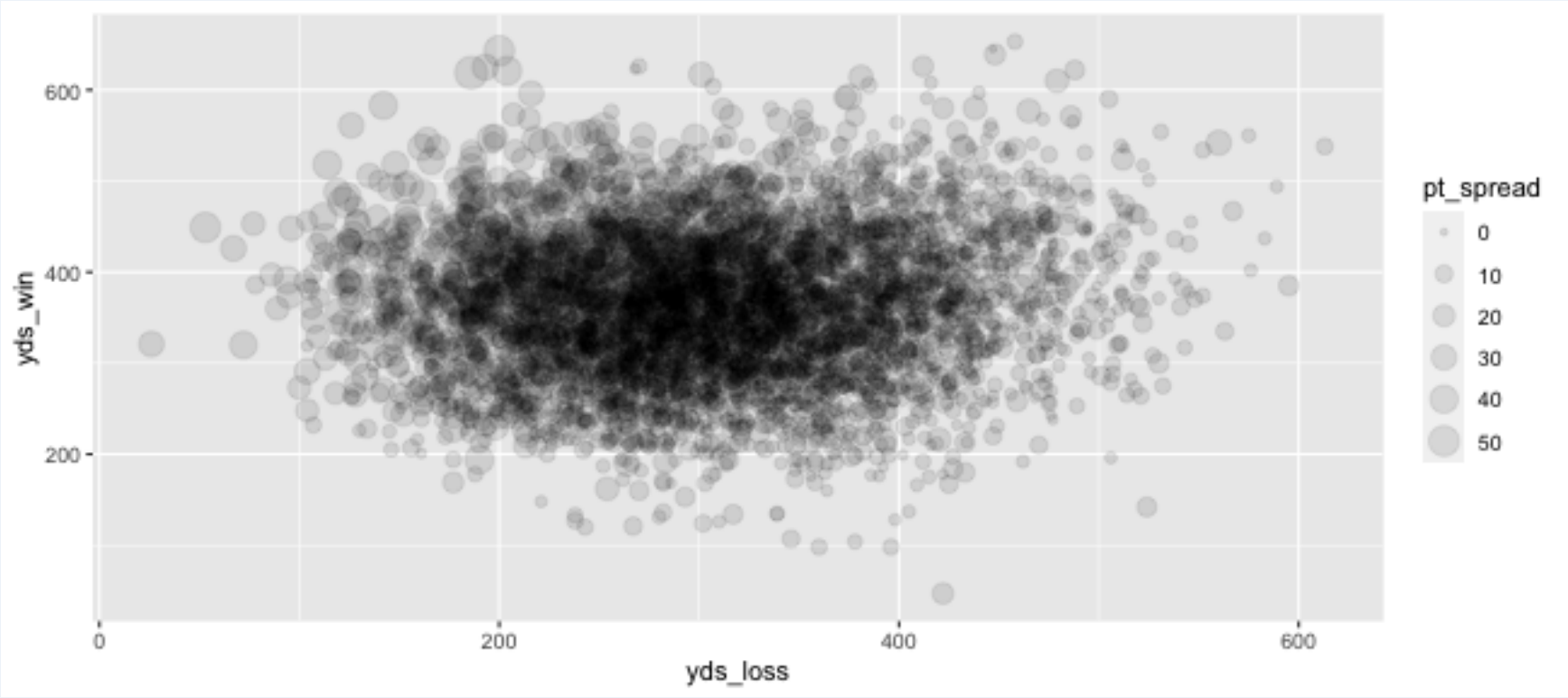
Using the `games` data from Tidy Tuesday:

```
games ← read_csv("dat/nfl_games.csv")
glimpse(games)
```

```
## Rows: 5,324
## Columns: 20
## $ year           <dbl> 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 200…
## $ week           <chr> "1", "1", "1", "1", "1", "1", "1", "1", "1", "1", "1", "1", "1"…
## $ home_team      <chr> "Minnesota Vikings", "Kansas City Team", "Washington Team", "At…
## $ away_team      <chr> "Chicago Bears", "Indianapolis Colts", "Carolina Panthers", "Sa…
## $ winner         <chr> "Minnesota Vikings", "Indianapolis Colts", "Washington Redskins…
## $ tie            <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,…
## $ day            <chr> "Sun", "Sun", "Sun", "Sun", "Sun", "Sun", "Sun", "Sun", "Sun", …
## $ date           <chr> "September 3", "September 3", "September 3", "September 3", "Se…
## $ time           <time> 13:00:00, 13:00:00, 13:01:00, 13:02:00, 13:02:00, 13:02:00, 13…
## $ pts_win        <dbl> 30, 27, 20, 36, 16, 27, 21, 14, 21, 41, 9, 23, 20, 16, 41, 13, …
## $ pts_loss       <dbl> 27, 14, 17, 28, 0, 7, 16, 10, 16, 14, 6, 0, 16, 13, 36, 7, 7, 3…
## $ yds_win        <dbl> 374, 386, 396, 359, 336, 398, 296, 187, 395, 425, 233, 308, 379…
## $ turnovers_win  <dbl> 1, 2, 0, 1, 0, 0, 1, 2, 2, 3, 0, 1, 1, 0, 3, 4, 1, 0, 0, 1, 2, …
## $ yds_loss       <dbl> 425, 280, 236, 339, 223, 249, 278, 252, 355, 167, 255, 143, 211…
## $ turnovers_loss <dbl> 1, 1, 1, 1, 1, 1, 1, 3, 4, 2, 4, 6, 2, 1, 0, 1, 3, 3, 4, 3, 4, …
## $ home_team_name <chr> "Vikings", "The KC Team", "The DC Team", "Falcons", "Steelers",…
```

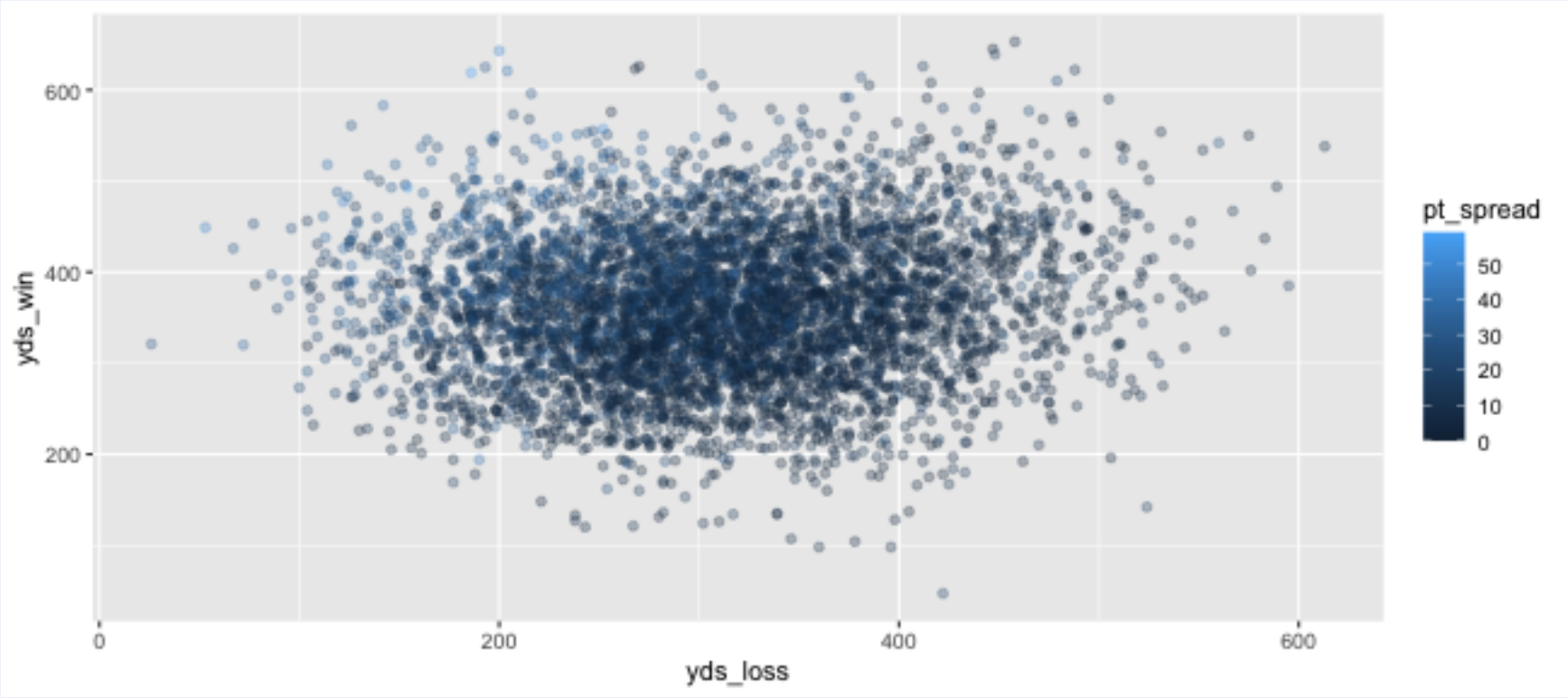
Size

```
ggplot(data = games, aes(x = yds_loss, y = yds_win, size = pt_spread)) +
  geom_point(alpha = .1)
```



Color

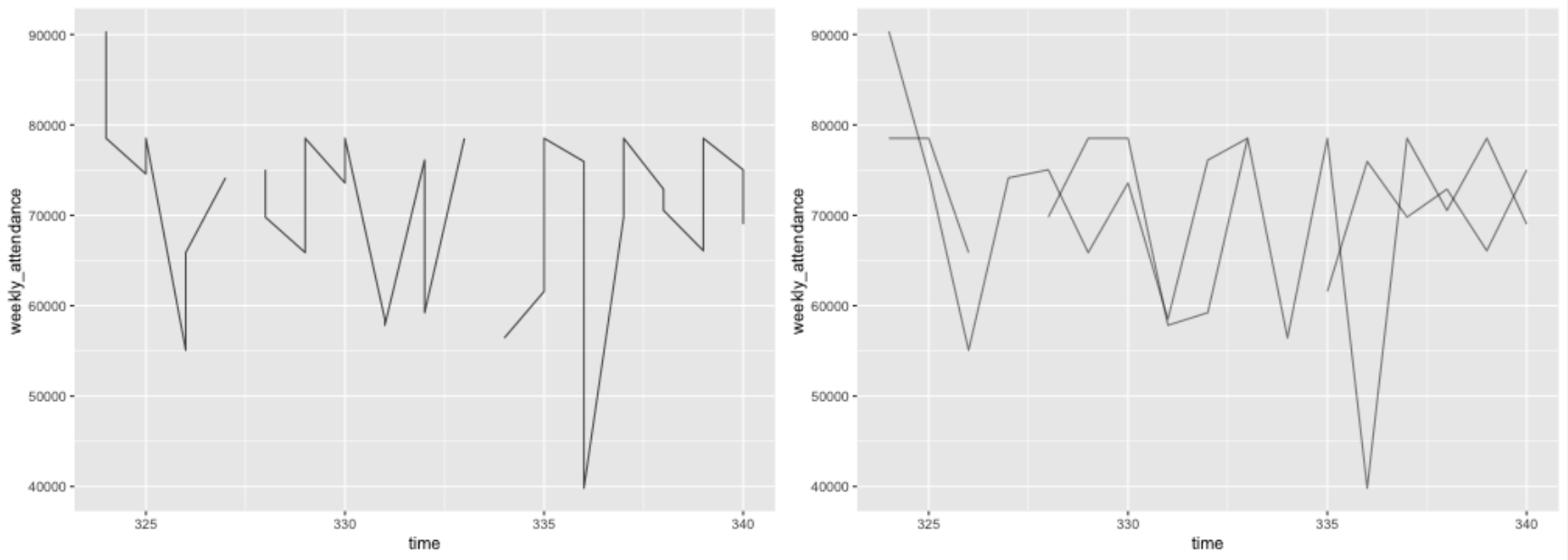
```
ggplot(data = games, aes(x = yds_loss, y = yds_win, color = pt_spread)) +
  geom_point(alpha = .3)
```



Grouping

> The `group` aesthetic partitions the data for plotting into groups

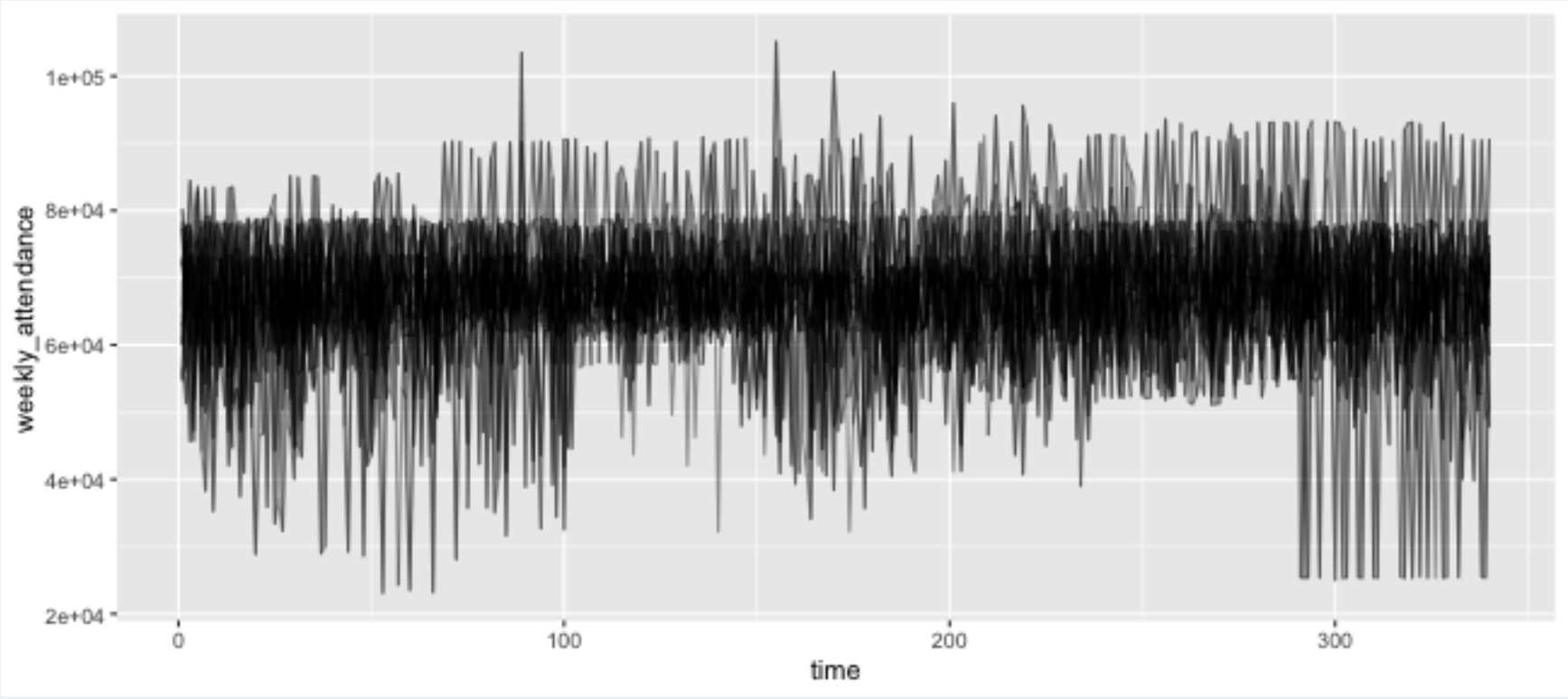
```
ny19 ← ny %>% filter(year == 2019)
ggplot(data = ny19, aes(x = time, y = weekly_attendance)) + geom_line(alpha = .7)
ggplot(data = ny19, aes(x = time, y = weekly_attendance, group =team_name)) + geom_line(alpha = .5)
```



Many groups

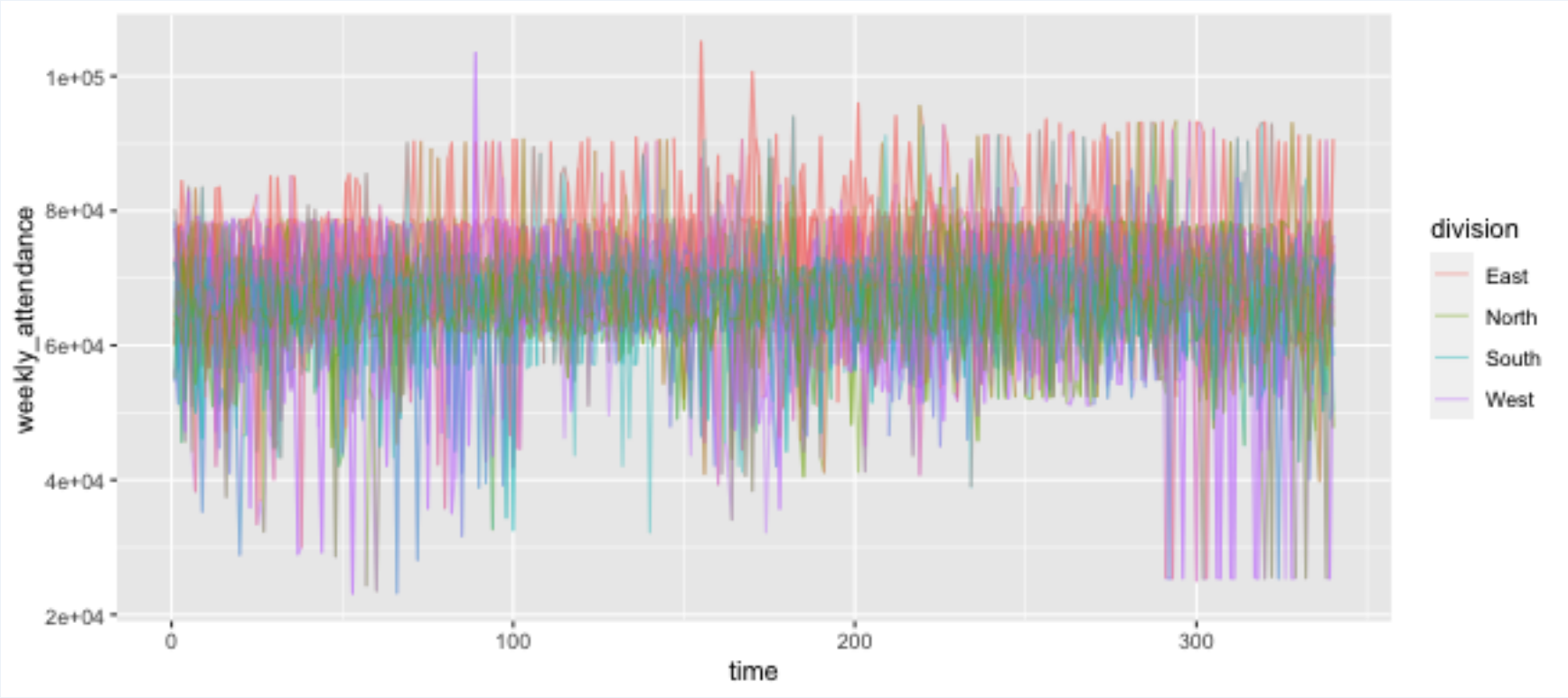
Look at all attendance for all teams

```
ggplot(data = attendance, aes(x = time, y = weekly_attendance, group =team_name))+
  geom_line(alpha = .4)
```



Add another variable

```
ggplot(data = attendance, aes(x = time, y = weekly_attendance, group =team_name, color = division))+
  geom_line(alpha = .4)
```



Facets

> Faceting generates small multiples, each displaying a different subset of the data. Facets are an alternative to aesthetics for displaying additional discrete variables.

- plot subgroups separately
- can be arranged by rows, columns, or both
- can be "wrapped" for many subgroups
- great for exploratory analyses

Faceting functions

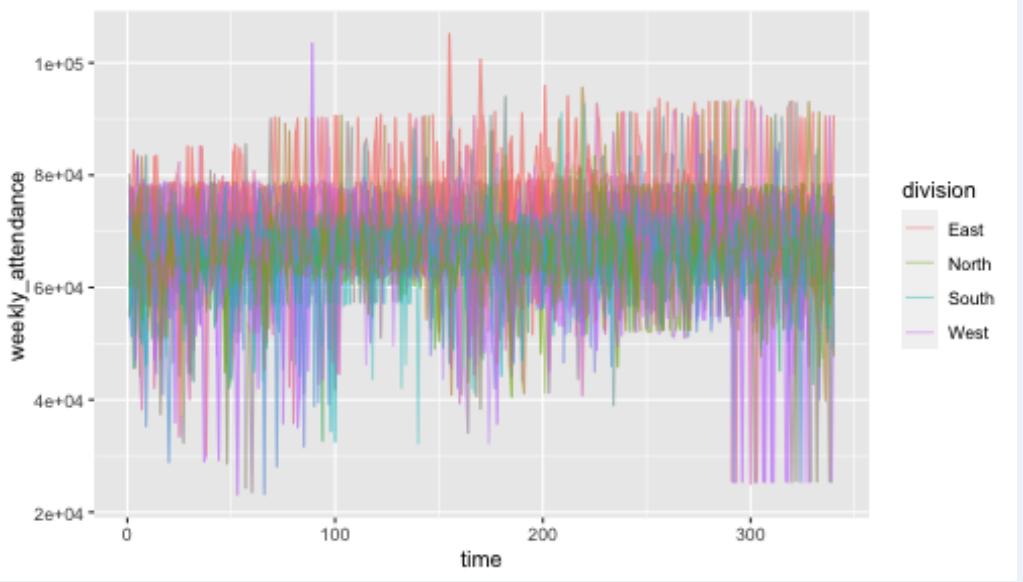
facet_grid()

- create a grid of graphs, by rows and columns
- use `vars()` to call on the variables
- adjust scales with `scales = "free"`

facet_wrap()

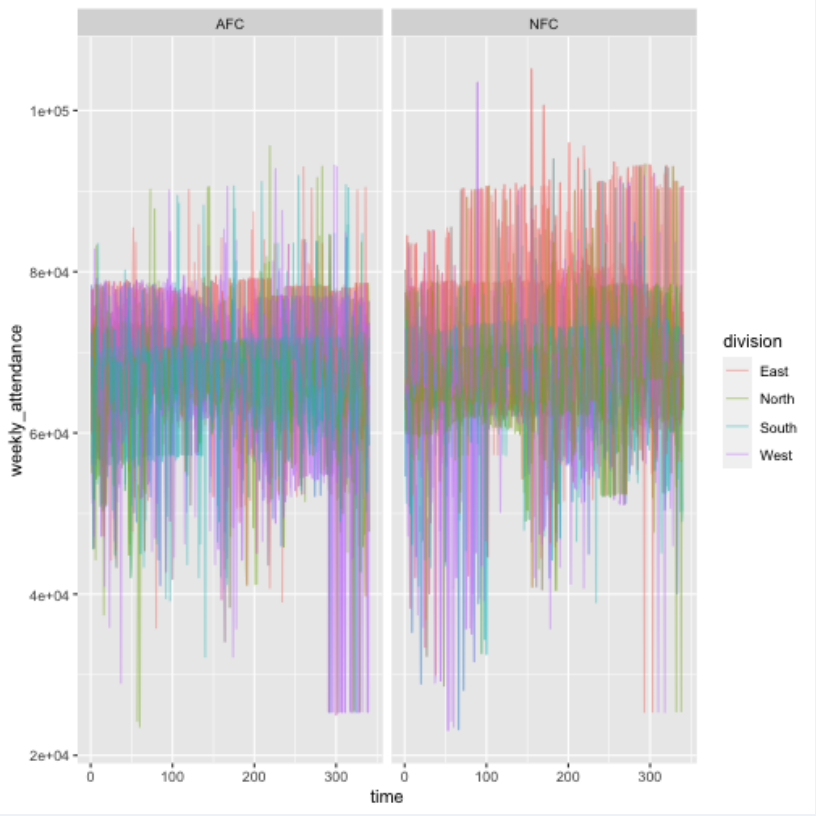
- create small multiples by "wrapping" a series of plots
- use `vars()` to call on the variables
- `nrow` and `ncol` arguments for dictating shape of grid

```
p ← ggplot(data = attendance, aes(x = time, y = we
  geom_line(alpha = .4)
p
```

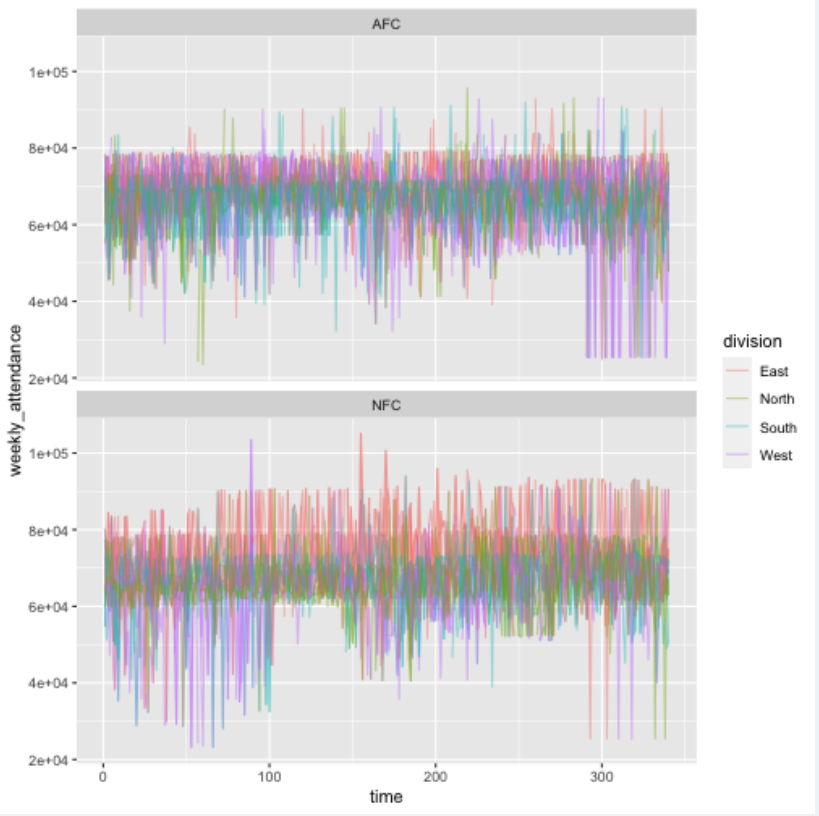


Faceting example

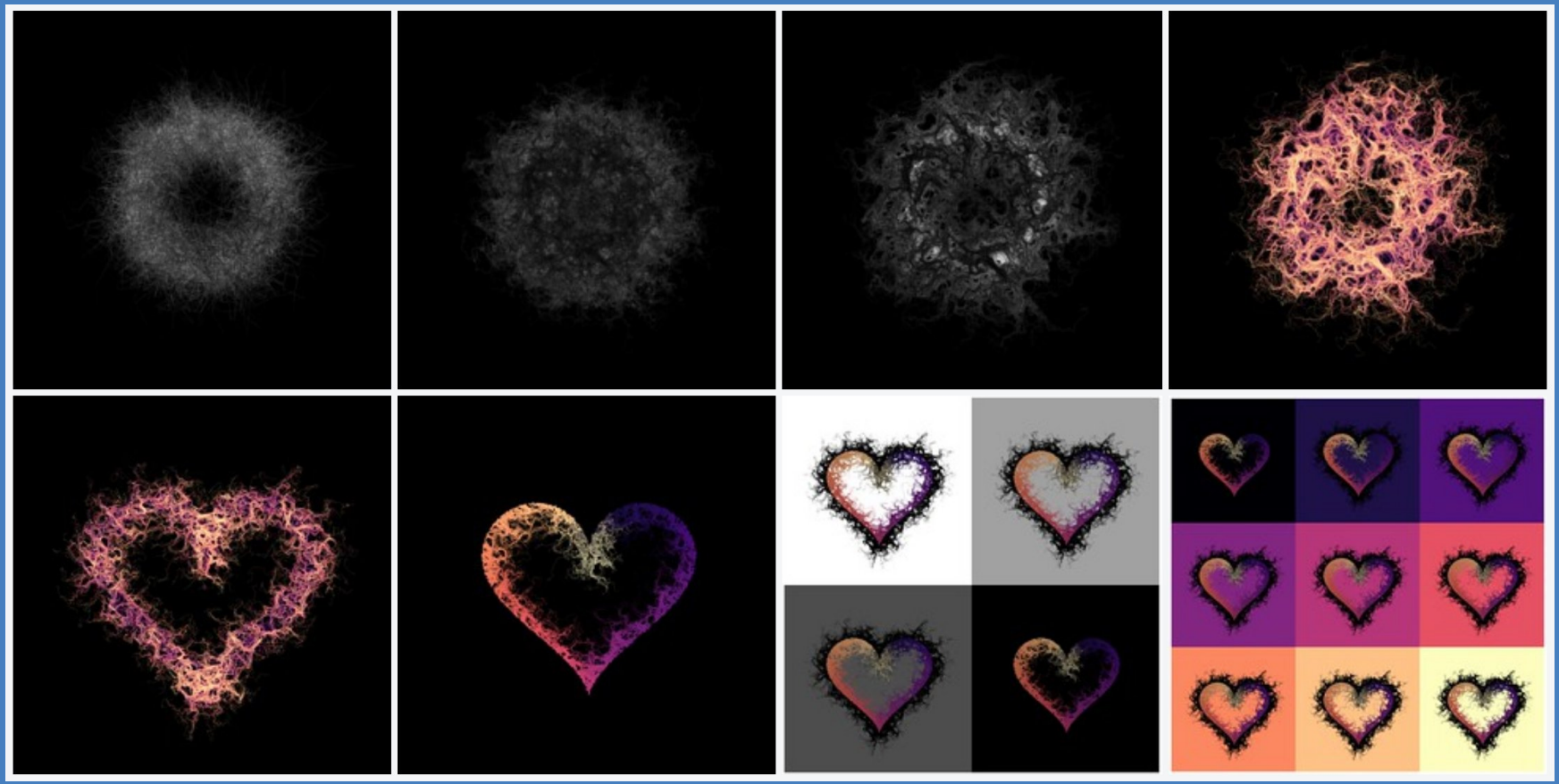
```
p + facet_grid(cols = vars(conference))
```




```
p + facet_wrap(vars(conference), nrow = 2)
```



Part 2: Advanced customization



Source @djnavarro

Combining layers

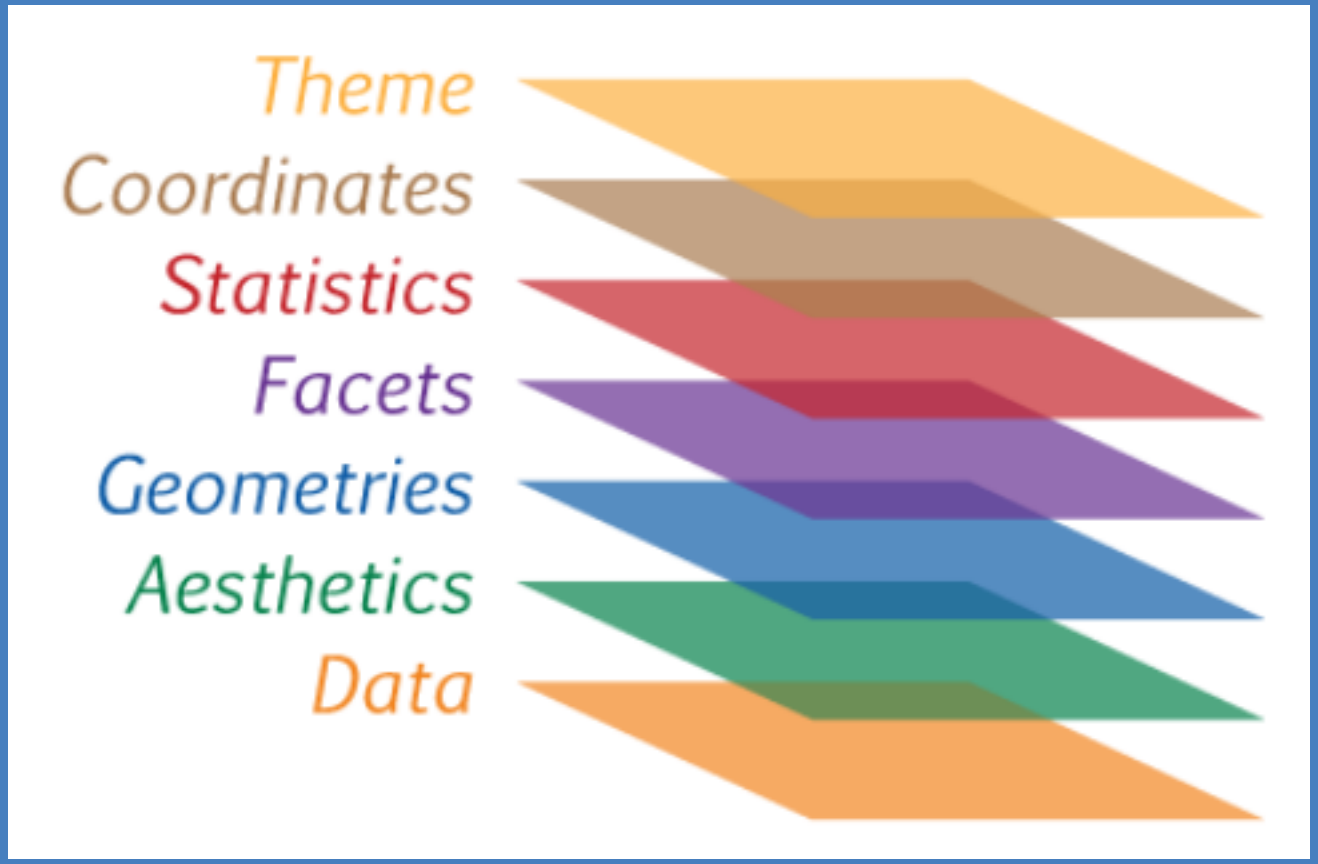
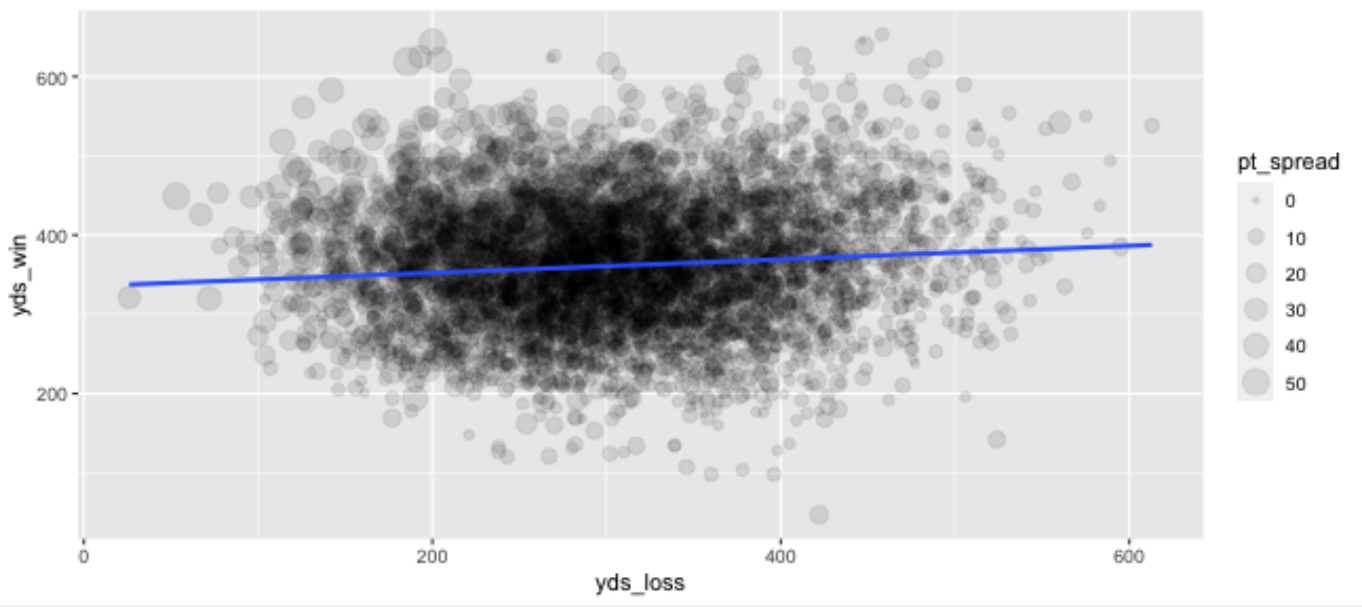


Image from skillgaze.com

Using the same data

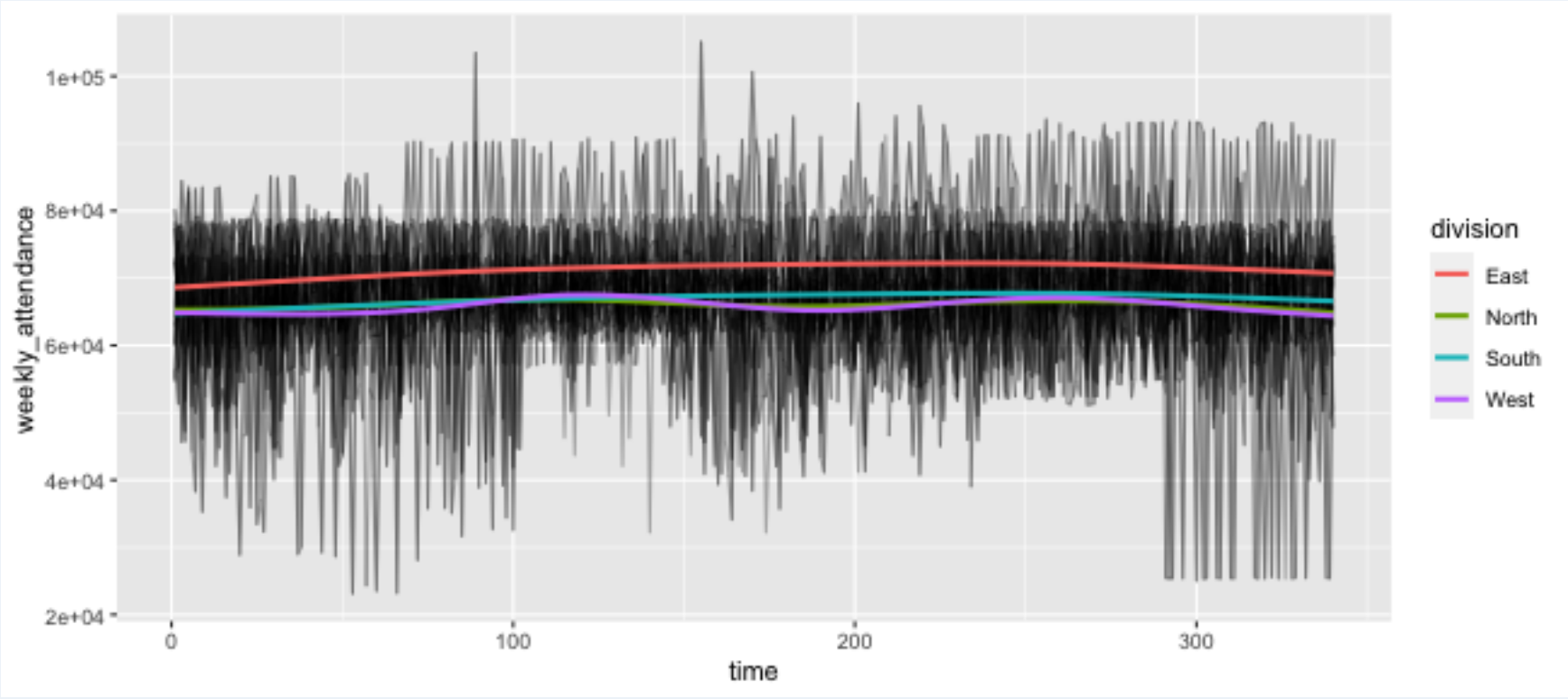
Add another layer by adding a different geom

```
ggplot(data = games, aes(x = yds_loss, y = yds_win)) +
  geom_point(aes(size = pt_spread), alpha = .1) +
  geom_smooth(se = FALSE, method = "lm")
```



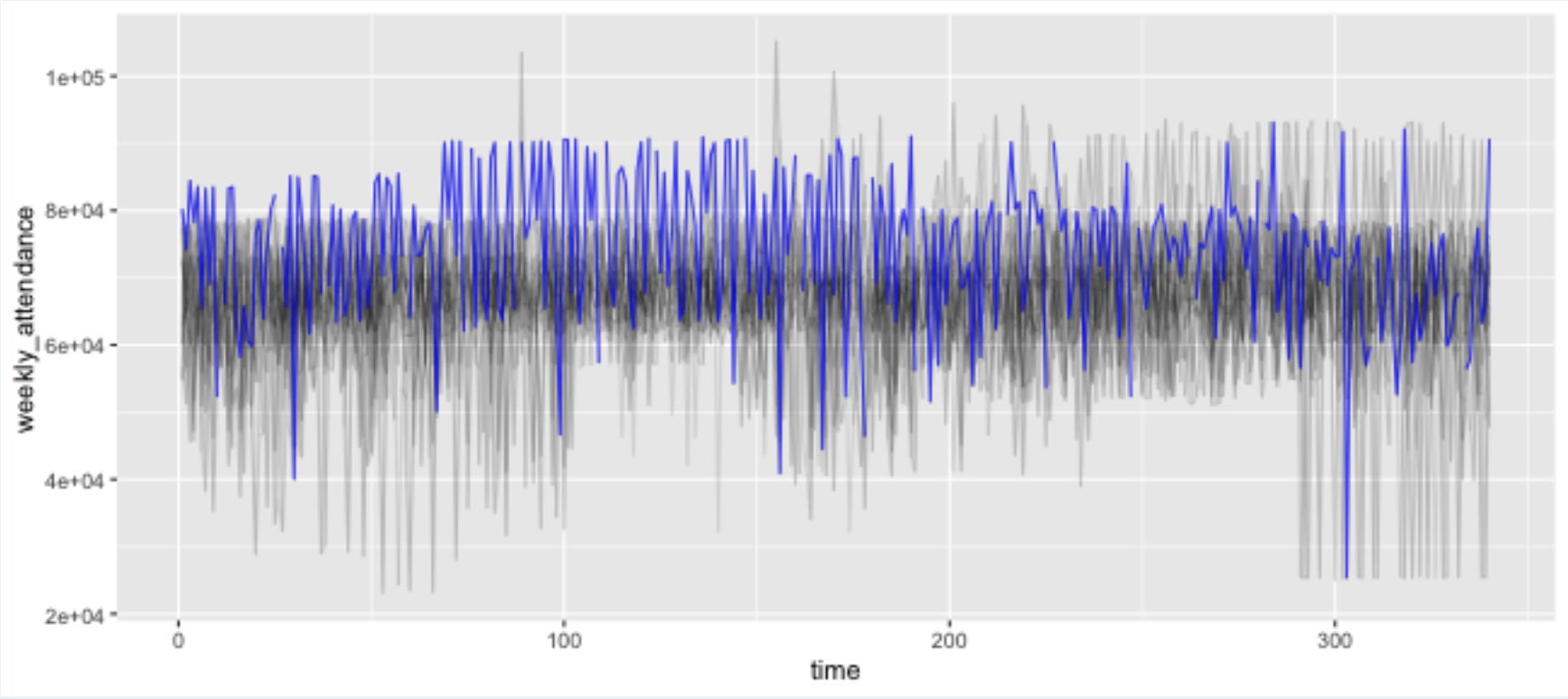
Your Turn

03:00



Using different data

```
ggplot() +
 geom_line(data = attendance, aes(x = time, y = weekly_attendance, group = team_name), alpha = .1) +
geom_line(data = dc, aes(x = time, y = weekly_attendance), alpha = .7, color = "blue")
```



Graph appearance

Image by @w_r_chase

Scales

> "Scales control the details of how data values are translated to visual properties."

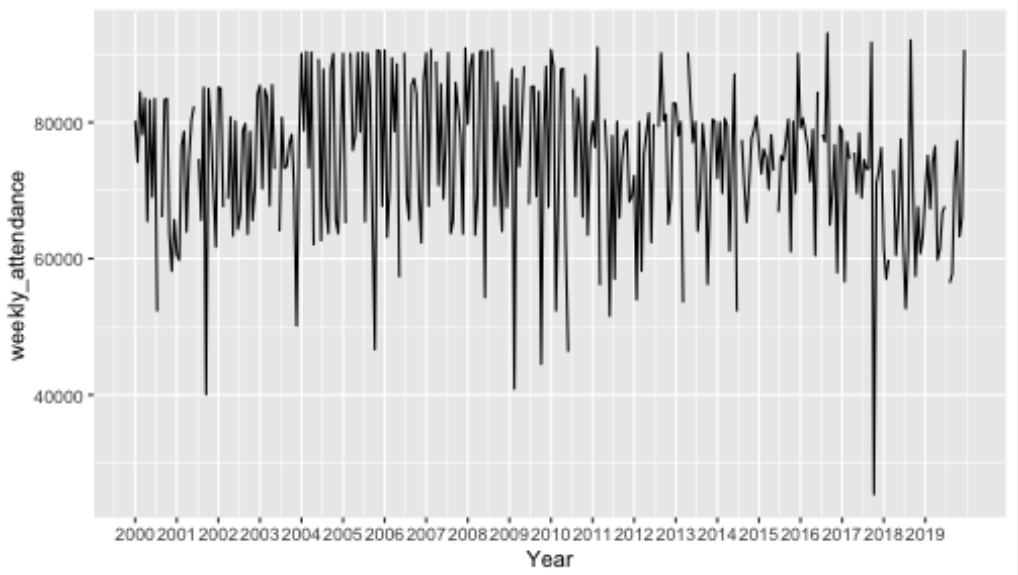
- Every aes value has a corresponding family of scales functions
- Of the form `scale_{aes}_*()`, e.g. `scale_x_continuous()`
- Values of the * depend on the aes
- Possible scale function arguments:
  - `name`: label of the axis/legend
  - `breaks`: numeric positions of breaks on axes/legends
  - `labels`: labels of the breaks on axes/legends
  - `limits`: continuous axis limits
  - `expand`: padding around data
  - `na.value`: what to do with missings
  - `trans`: continuous transformations of data
  - `guide`: function to create guide/legend
  - `date_breaks`: breaks for date variables

Scales for axes

`scale_x_*()`, `scale_y_*()`

- continuous
- discrete
- binned
- log10
- sqrt
- date
- datetime
- reverse

```
ggplot(dc, aes(x = time, y = weekly_attendance)) +
  geom_line() +
  scale_x_continuous(breaks = seq(1, 340, by = 17),
                     labels = 2000:2019, name = "Ye
```

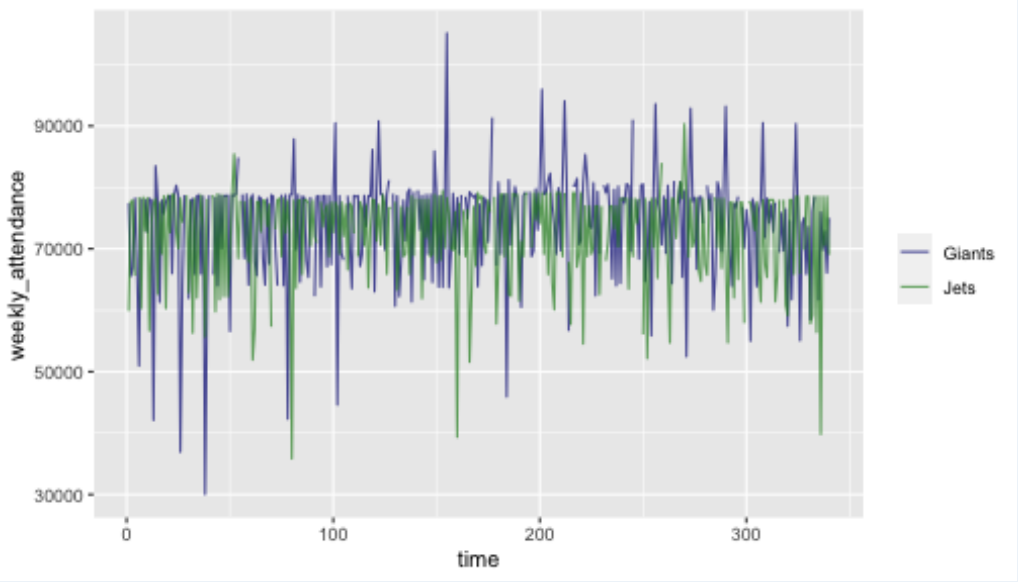


Scales for color

`scale_color_*()`, `scale_fill_*()`

- manual
- continuous
- brewer/distiller/fermenter
- gradient/gradient2/gradientn
- steps
- viridis

```
ggplot(ny, aes(x = time, y = weekly_attendance,
  color = team_name)) + geom_line(alpha = .7) +
  scale_color_manual(name = NULL,
    values = c("navy", "forestgreen"))
```

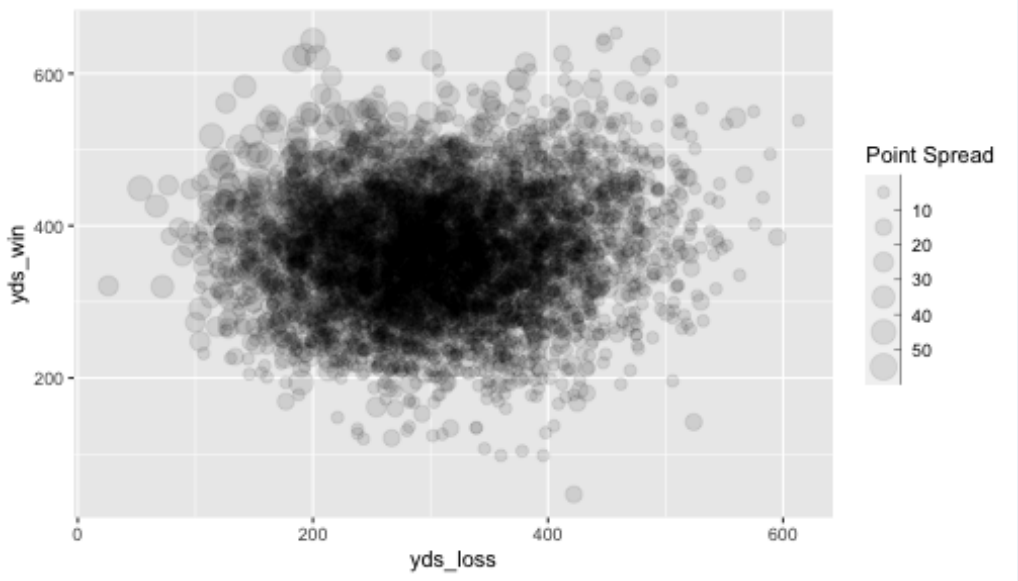


Scales for size

scale_*()

- size
- radius
- size_binned
- size_area
- size_binned_area

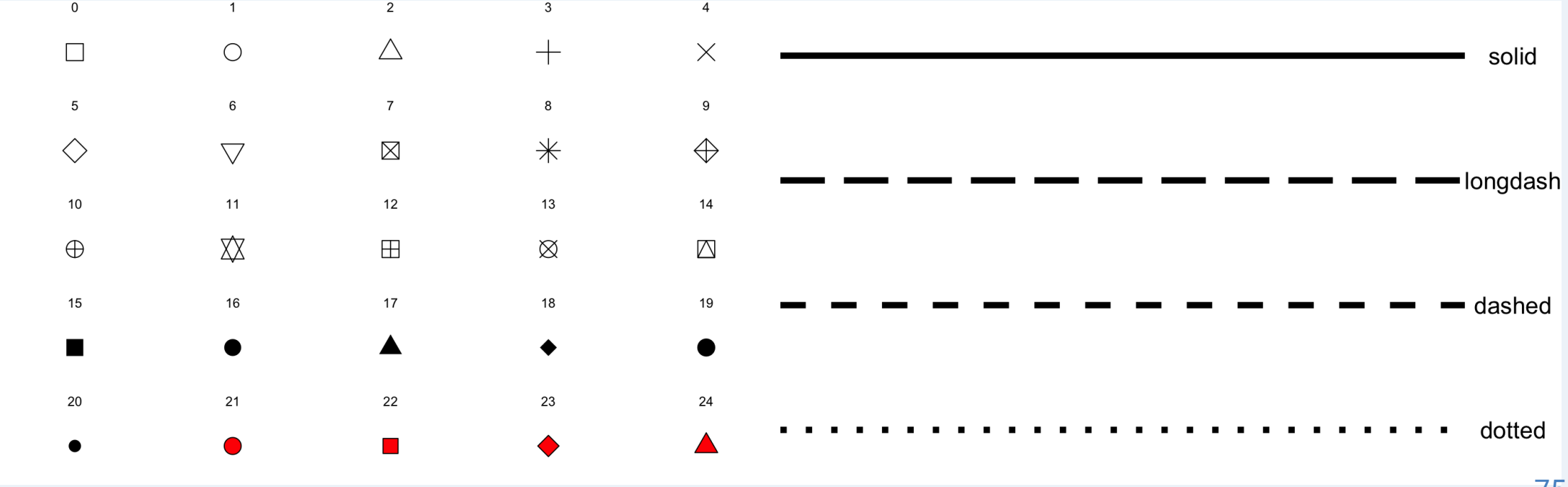
```
ggplot(data=games, aes(x = yds_loss,
  y = yds_win, size = pt_spread)) +
  geom_point(alpha = .1) +
  scale_size_binned("Point Spread",
                    n.breaks = 5)
```



Other scale functions

- `scale_alpha_*()`: for mapping a variable to transparency
- `scale_linetype_*()`: for mapping a variable to linetype (`geom_line`)
- `scale_shape_*()`: for mapping a variable to shape (`geom_point`)

📢 Note that linetype and shape have relatively few possible values by default, and are best for variables with only a few levels.



Labels

Labels are also scale elements

- `ggtitle(main, subtitle)`: plot title & subtitle
- `xlab()`, `ylab()`: axes titles
- `labs()`: all of the above plus captions, tags, and other aes values
  - e.g. `color = "My variable"` names the legend "My variable"
  - `title`: plot title
  - `subtitle`: plot subtitle
  - `caption`: text for bottom right corner of plot area e.g. to ID data source
  - `tag`: text for top left corner of plot area, e.g. A, B, C when combining many plots together

Coordinates

The `coord_*()` family of functions dictate position aesthetics (e.g. `x`, `y`):

- Controls the "canvas" the data are "drawn" on
- Especially useful for maps

Function examples:

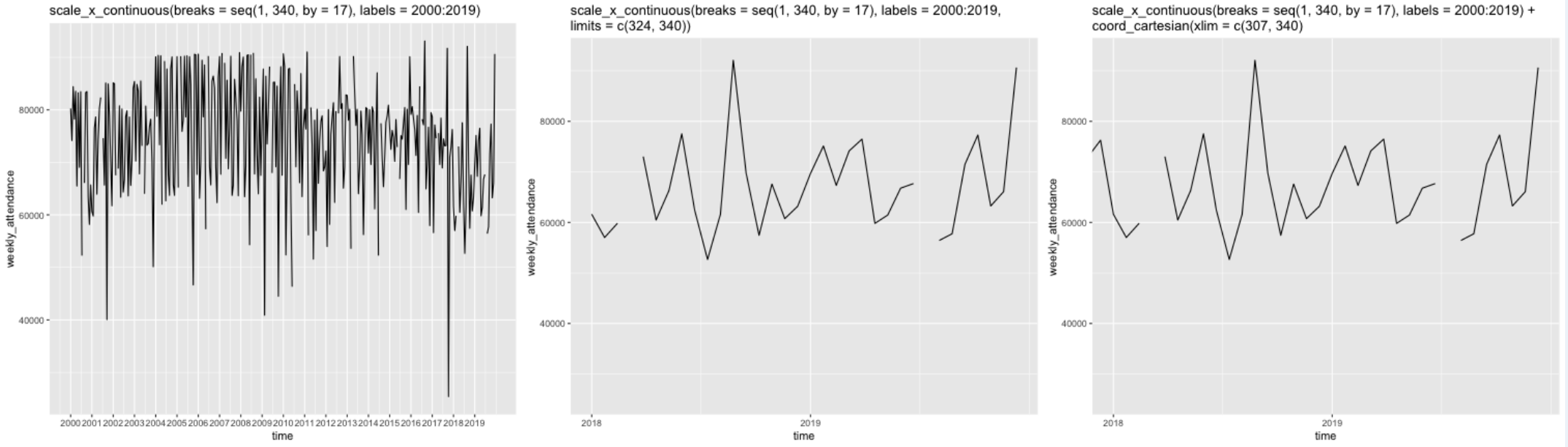
- `coord_cartesian()`: the default. x, y axes
- `coord_polar()`: x becomes radius, y becomes angle

📢 You can apply limits and transformations to axes in scales or coordinates (e.g. `xlim()`, `ylim()`) but using coordinates is probably what you want.

Example

All three visualizations below begin with the same plot code:

```
ggplot(dc, aes(x = time, y = weekly_attendance)) + geom_line() +
```



Themes

Specific themes:

- `theme_grey()`: default
- `theme_bw()`: white background, gray gridlines
- `theme_classic()`: looks more like base R plots
- `theme_void()`: removes all background elements, all axes elements, keeps legends

General `theme` function for advanced customization:

- `theme()`
  - adjust the appearance every "non-data element" of the viz
  - fonts, background, text positioning, legend appearance, facet appearance, etc.
- 📢 Rule of thumb: when changing an element that shows data, use aes() and scales. Otherwise, use themes.

Theme elements

Every theme element is either a line, a rect, or text. See documentation for more.

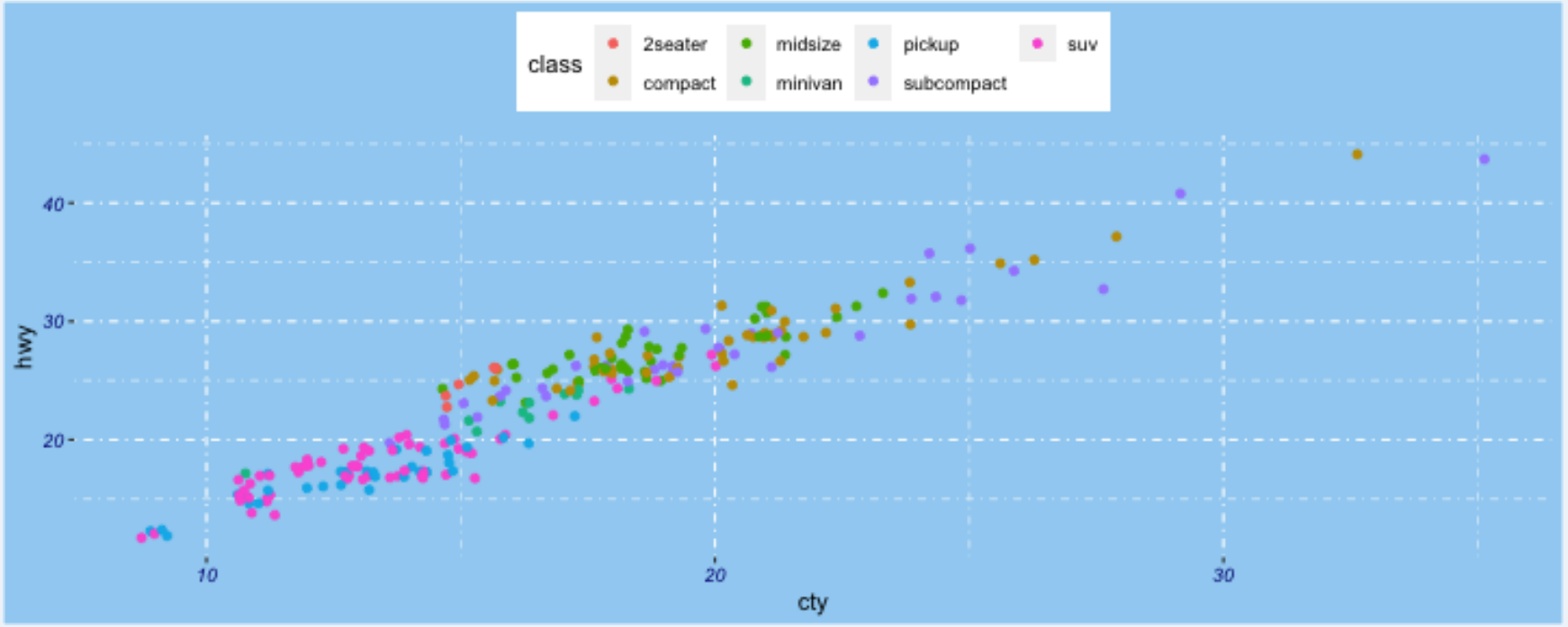
To modify a theme element, use:

- `element_line()`: change lines' appearance (color, linetype, size, etc.)
- `element_rect()`: change rectangles' appearance (fill, border lines, etc.)
- `element_text()`: change text elements' appearance (family, face, color, etc.)
- `element_blank()`: draw nothing. Use to remove a theme element.

📢 Note: there are 92 possible arguments used to modify a ggplot theme. Usually, we will only need to call on a handful.

Example

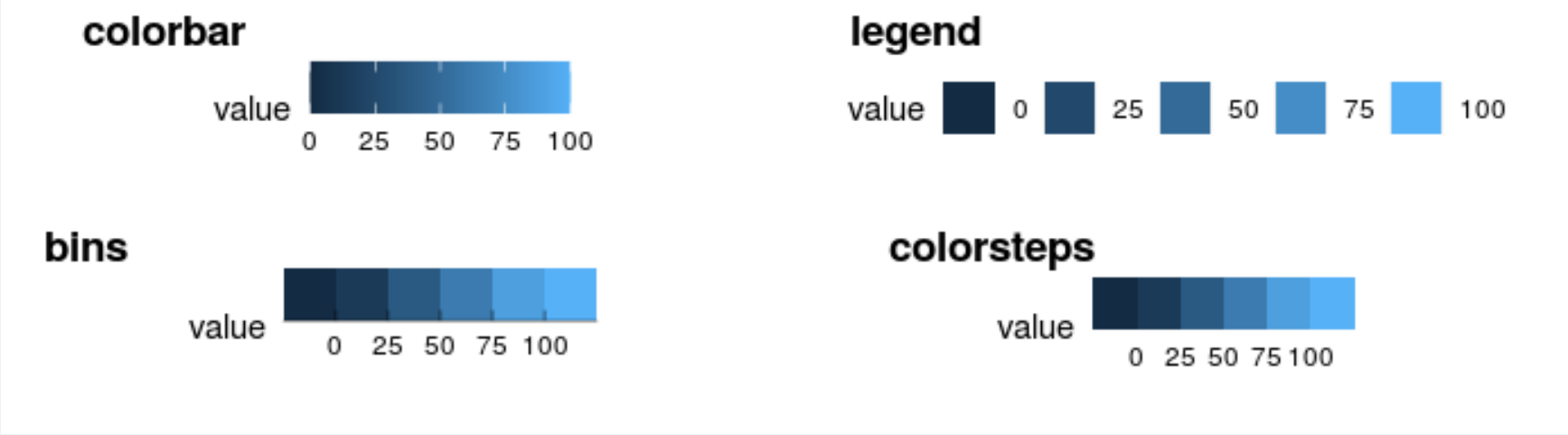
```
mytheme ← theme(legend.position =  "top",
        axis.text = element_text(face = "italic", color = "navy"),
        plot.background = element_rect(fill = "#a0d1f2"),
        panel.background = element_blank(),
        panel.grid = element_line(linetype = "dotdash"))
ggplot(data = mpg) + geom_jitter(aes(x = cty, y = hwy, color = class)) + mytheme
```



Legends

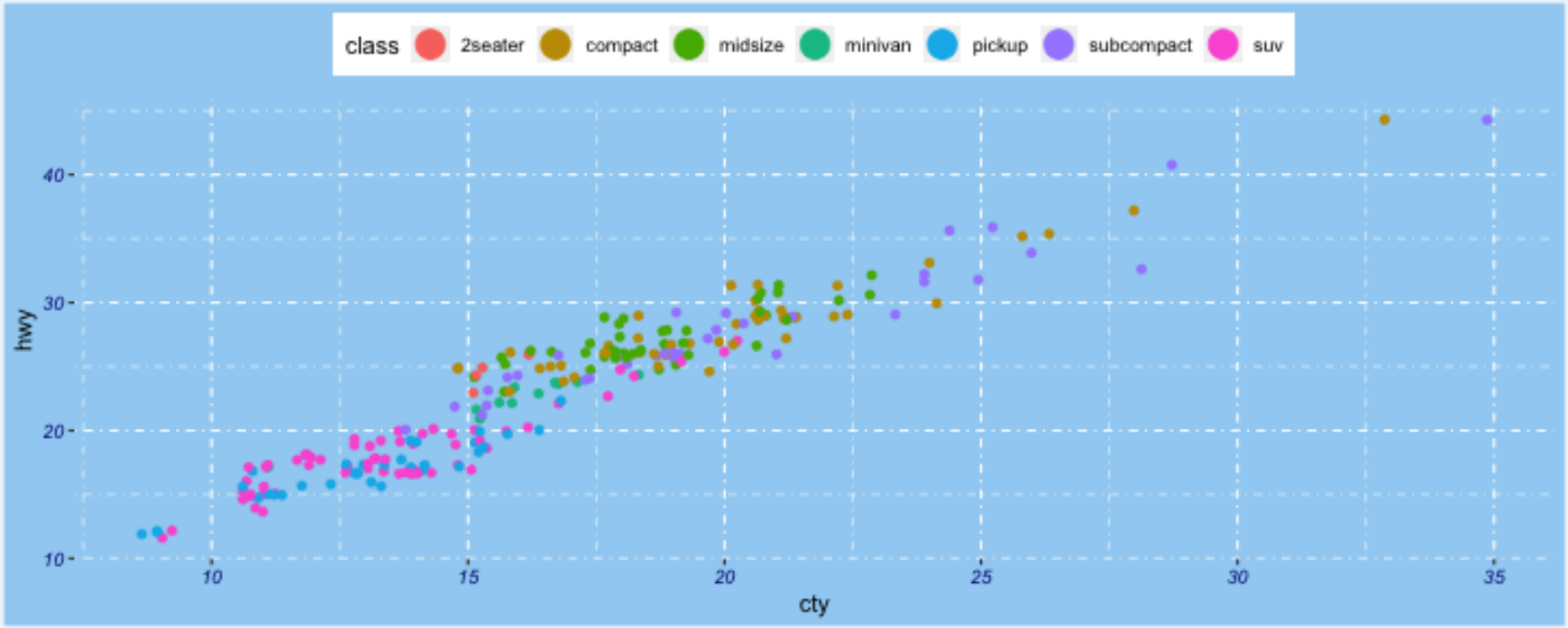
The `guides()` family of functions control legends' appearance

- `guide_colorbar()`: continuous colors
- `guide_legend()`: discrete values (shapes, colors)
- `guide_axis()`: control axis text/spacing, add a secondary axis
- `guide_bins()`: creates "bins" of values in the legend
- `guide_colorsteps()`: makes colorbar discrete



Example

```
ggplot(data = mpg) +
  geom_jitter(aes(x = cty, y = hwy, color = class),key_glyph = draw_key_pointrange) +
  mytheme +
  guides(color = guide_legend(nrow = 1))
```



Fonts

To change fonts in a `ggplot2` viz:

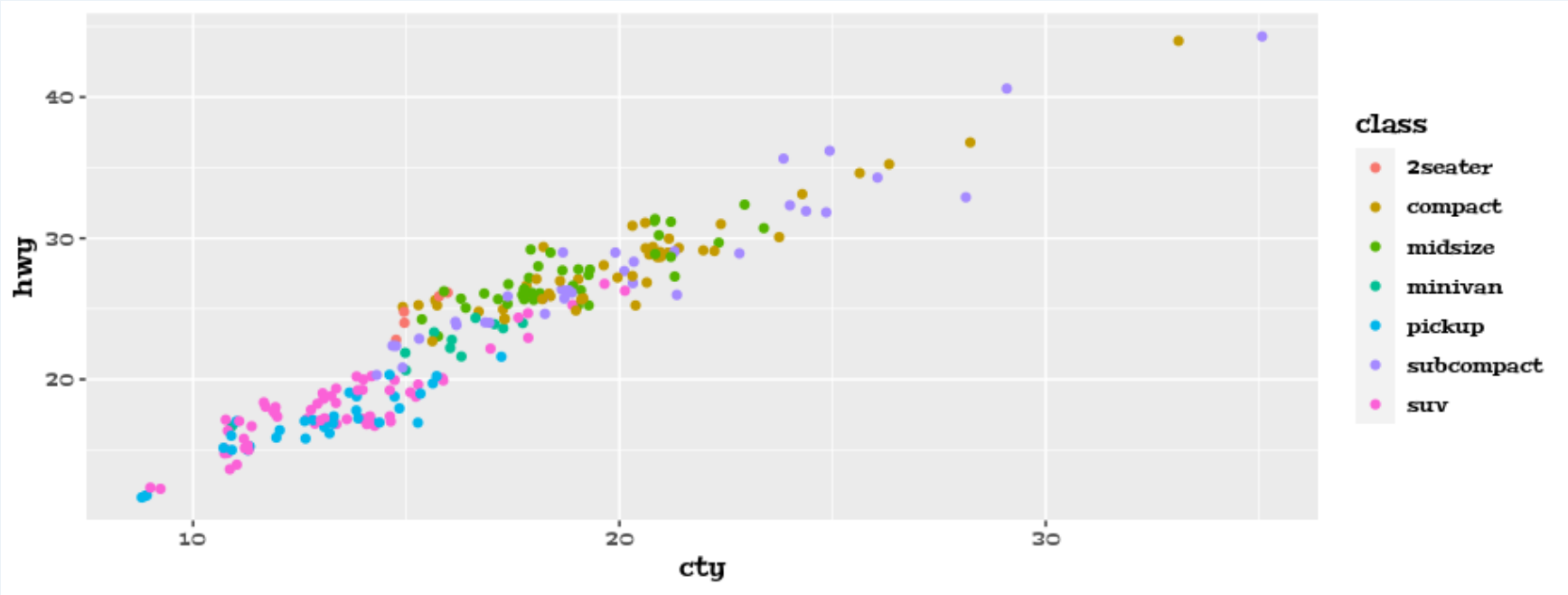
- Use the `element_text()` function inside of `theme()`
  - `family`: font family
  - `face` : bold, italic, bold.italic, plain
  - `color`, `size`, `angle`, etc.
- Include additional fonts with the `extrafont` package:

```
library(extrafont)
font_import() # will take 2-3 minutes. Only need to run once
loadfonts()
fonts()
fonttable()
```

📢 Restart R after importing the fonts for the first time, and load `extrafont` and `loadfonts` BEFORE loading `ggplot2`.

Example

```
ggplot(data = mpg) +
  geom_jitter(aes(x = cty, y = hwy, color = class)) +
  theme(text = element_text(family = "Peralta"))
```

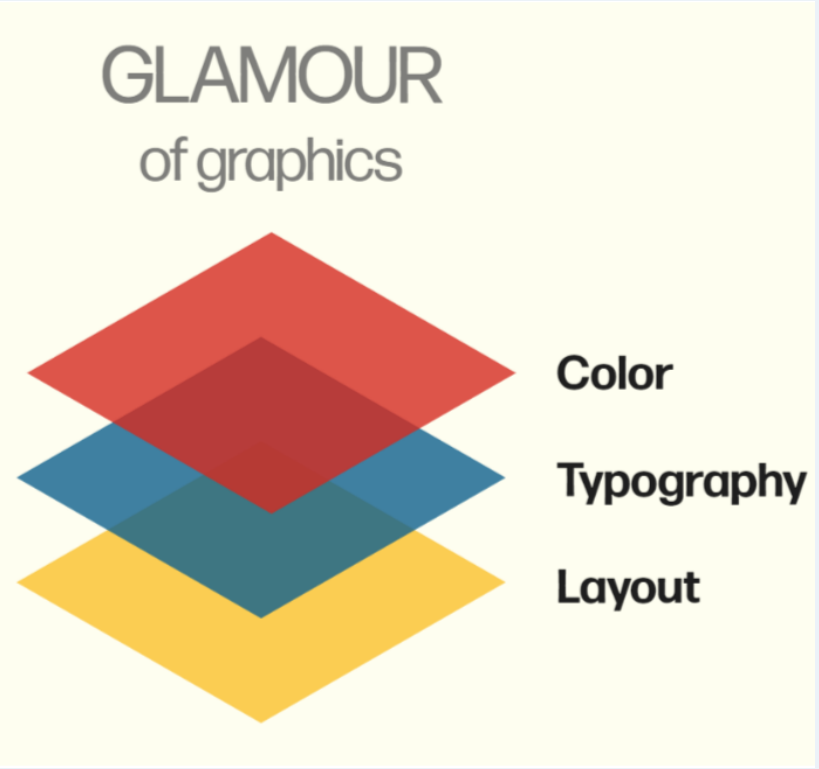


Design principles

Advice on data visualization design from Will Chase.

- Left-align text
- Don't make people tilt their heads!
- Remove borders & gridlines
- Directly label in place of legends
- White space is your friend
- Use simple, clear fonts
- Color is hard. (Stick to the available palettes)

Watch Will's talk here.

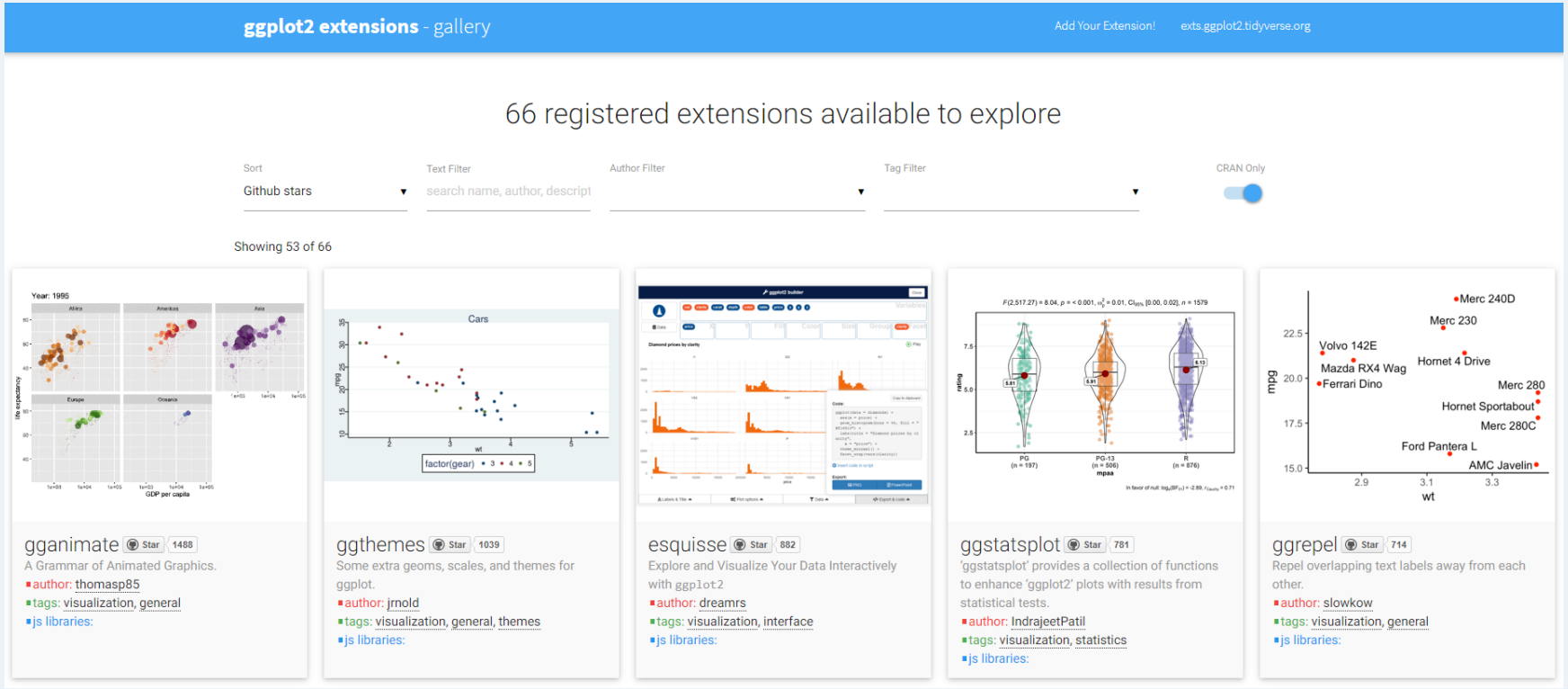


ggplot2 extensions

Image by @allison_horst

Where to find them

Maintainers of packages can put their `ggplot2` extension on exts.ggplot2.tidyverse.org/gallery



Animation

`gganimate`: a grammar of animated graphics

From the documentation, here are the `gganimate` function families:

- `transition_*()`: defines how the data should be spread out and how it relates to itself across time.
- `view_*()`: defines how the positional scales should change along the animation.
- `shadow_*()`: defines how data from other points in time should be presented in the given point in time.
- `enter_*()/exit_*()`: defines how new data should appear and how old data should disappear during the course of the animation.
- `ease_aes()`: defines how different aesthetics should be eased during transitions.

📢 For optimum performance, use the `animate()` and `anim_save()` functions to create and save animations.

Example (data)

Full simulated data:

```
## Rows: 100
## Columns: 9
## $ sim      <dbl> 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20…
## $ switch   <dbl> 1, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 1, 1, 1, 0, 0, 0, 0, 1, 1, …
## $ win      <dbl> 1, 0, 0, 1, 1, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0, …
## $ Switch   <chr> "Yes", "No", "No", "No", "No", "No", "No", "Yes", "Yes", "No", "No", …
## $ Win      <chr> "Win", "Lose", "Lose", "Win", "Win", "Lose", "Lose", "Win", "Lose", "…
## $ count    <dbl> 1, 1, 2, 1, 2, 3, 4, 2, 1, 5, 3, 6, 3, 7, 2, 3, 4, 4, 8, 9, 10, 5, 4,…
## $ n_switch <dbl> 1, 1, 2, 3, 4, 5, 6, 2, 3, 7, 8, 9, 4, 10, 5, 6, 7, 11, 12, 13, 14, 8…
## $ perc     <dbl> 0.02127660, 0.01886792, 0.03773585, 0.01886792, 0.03773585, 0.0566037…
## $ perc2    <dbl> 0.00000000, 0.00000000, 0.01886792, 0.00000000, 0.01886792, 0.0377358…
```

Example (code)

```
library(gganimate)
ggplot(data = sims) +
  geom_col(aes(x = Win, y = perc, fill = Win, group = seq_along(sim))) +
  facet_grid(cols = vars(Switch), labeller = label_both) +
  scale_fill_manual(values = c("#F6C40C", "#4AA0BB")) +
  theme_bw() +
  theme(legend.position = "none",
        text = element_text(family = "Peralta", size = 20),
        panel.grid = element_blank(),
        strip.background = element_rect(fill = NA),
        plot.title =  element_text(hjust = .5),
        plot.background = element_rect(fill = NA),
        panel.background = element_rect(fill = NA))  +
  labs(title = "Let's Make a Deal!", subtitle = "Sim: {frame_along}") +
  transition_reveal(seq_along(sim))
```

Example (animation)

GIF viewable on HTML
version of slides
available on
rstudio.cloud.

Your Turn

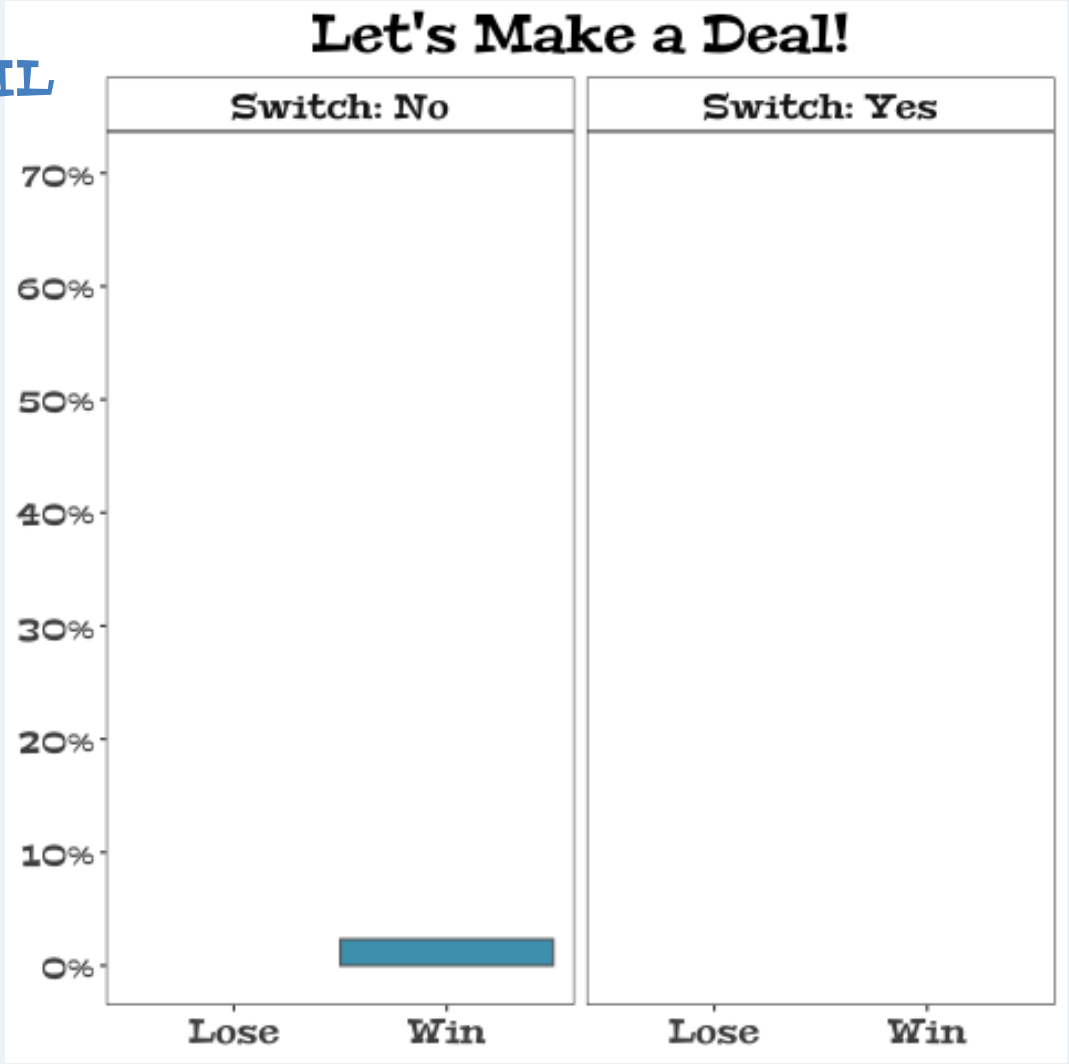
06:00

Complete the code to recreate the GIF from the motivating example. (GIF on next slide)

```
ggplot(data = sims) +
  geom_???(aes(xmin = win - .5, xmax = win + .5, ymin = perc2, ymax = perc, fill = Win,
               group = seq_along(sim)),
           color = "grey40") +
  facet_grid(cols = vars(Switch), labeller = label_both) +
  scale_x_continuous(breaks = c(0,1), labels = ??? ) +
  scale_y_continuous(breaks = ???, labels = scales::label_percent(accuracy = 1)) +
  scale_fill_manual(values = c("#F6C40C", "#4AA0BB")) +
  theme_bw() +
  theme(legend.position = "none",
        text = element_text(family = "Peralta", size = 20),
        panel.grid = element_blank(),
        strip.background = element_rect(fill = NA),
        plot.title =  element_text(hjust = .5),
        plot.background = element_rect(fill = NA),
        panel.background = element_rect(fill = NA))  +
  labs(title = "Let's Make a Deal!")  +
  transition_???(sim, transition_length = 10, state_length = 5) +
  shadow_???(color = NA)
```

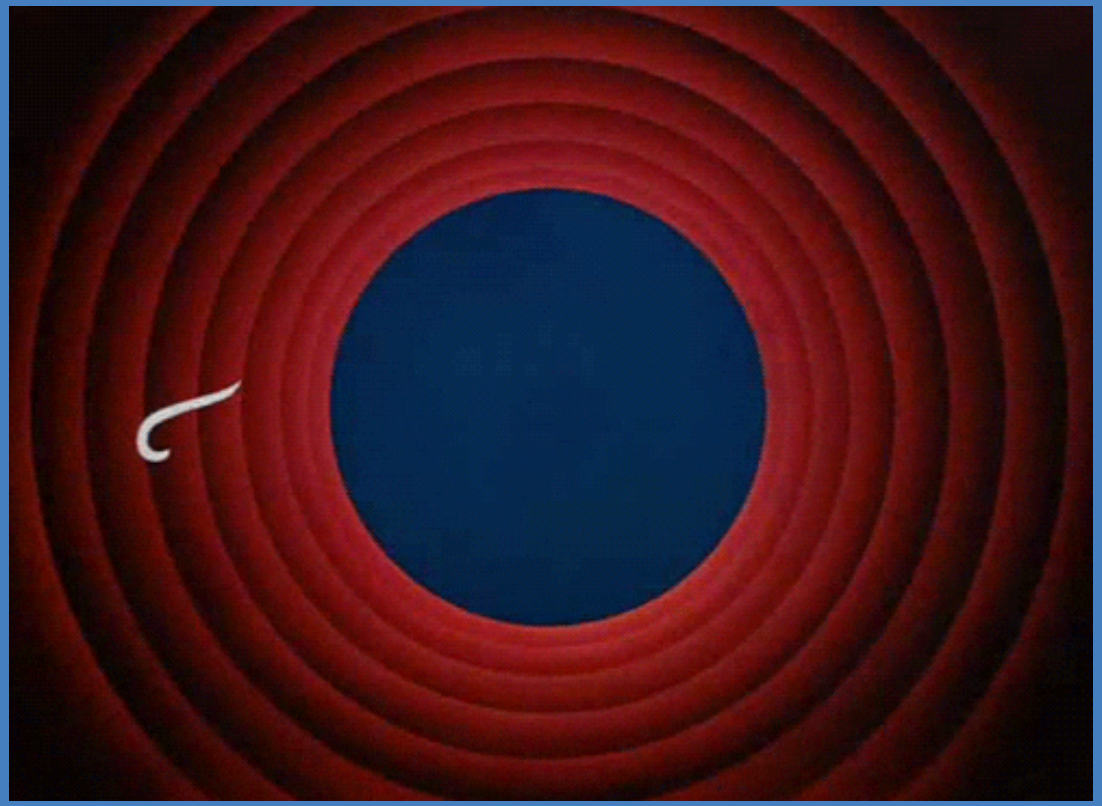
Motivating example (again)

GIF viewable on HTML version of slides available on rstudio.cloud.



GIF viewable on HTML version of slides available on rstudio.cloud.

Conclusion



Additional resources

- ggplot2 book
- plotly book
- R for Data Science book
- Tidy Tuesday
- My advice for getting help in R
- Thomas Lin Pedersen's ggplot2 webinar: part 1 and part 2
- RStudio Cheat Sheets
- Will Chase's Design Talk at rstudio::conf
- The 4-hour version of this workshop
- More in the `resources/` folder

Many of these and more are in the rstudio.cloud project