Statistical graphics with `ggplot2`

Programming for Statistical Science

Shawn Santo

Supplementary materials

Full video lecture available in Zoom Cloud Recordings

Additional resources

- Chapter 3, R for Data Science
- `ggplot2` Reference
- `ggplot2` cheat sheet
- color brewer 2

ggplot2

- ggplot2 is a plotting system for R, based on the grammar of graphics
    - using the good parts of base and lattice
- It takes care of many of the fiddly details that make plotting a hassle
    - such as drawing legends and faceting
    - particularly helpful for plotting multivariate data

Package ggplot2 is available in package tidyverse. Let's load that now.

```
library(tidyverse)
```

The Grammar of Graphics

- Visualization concept created by Leland Wilkinson (1999)
    - to define the basic elements of a statistical graphic
- Adapted for R by Wickham (2009)
    - consistent and compact syntax to describe statistical graphics
    - highly modular as it breaks up graphs into semantic components
- It is not meant as a guide to which graph to use and how to best convey your data (more on that later).

Today's data: MLB

```
teams <- read_csv("http://www2.stat.duke.edu/~sms185/data/mlb/teams.csv")
```

Object `teams` is a data frame that contains yearly statistics and standings for MLB teams from 2009 to 2018.

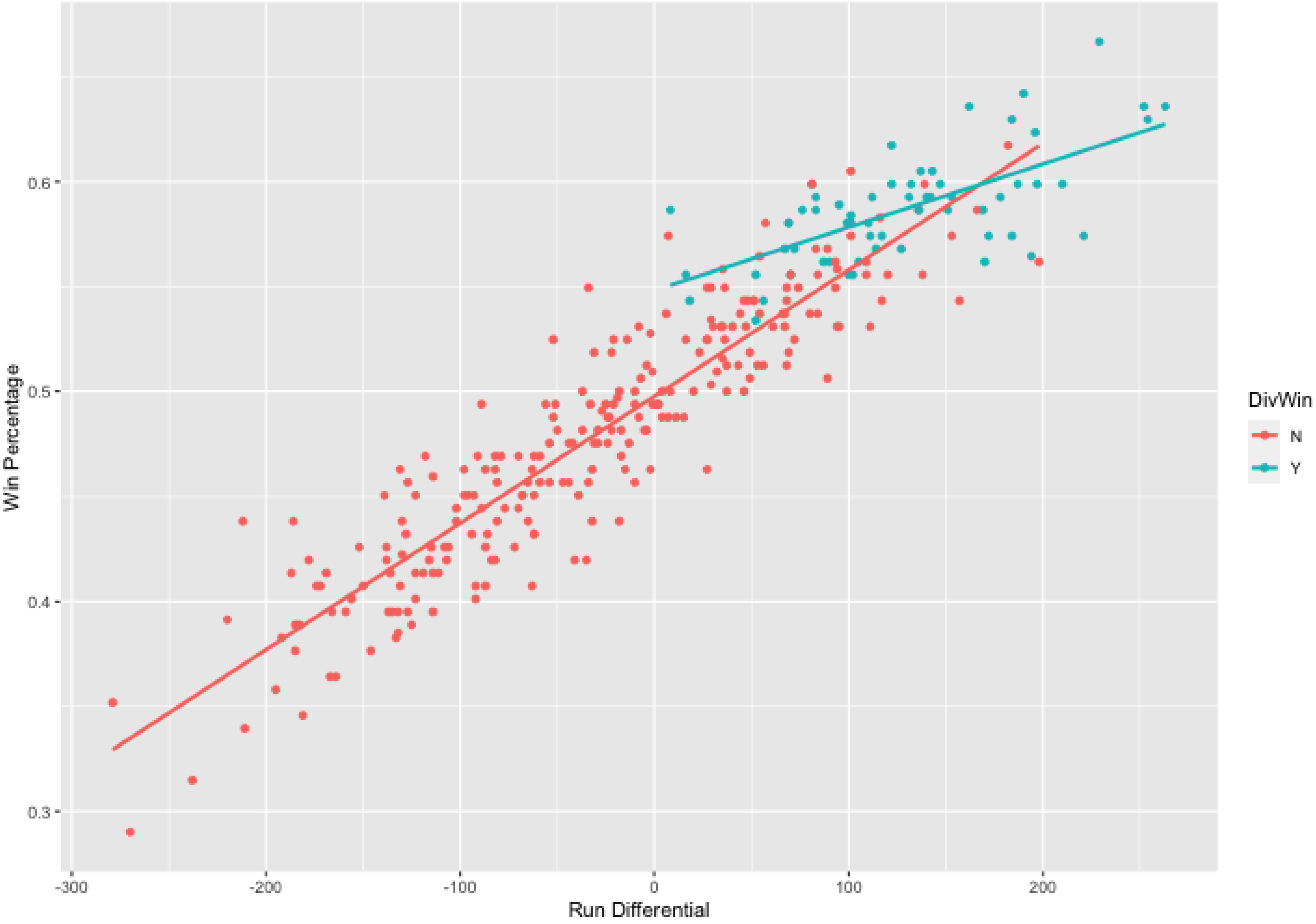
The data has 300 rows and 56 variables.

```r
teams
```

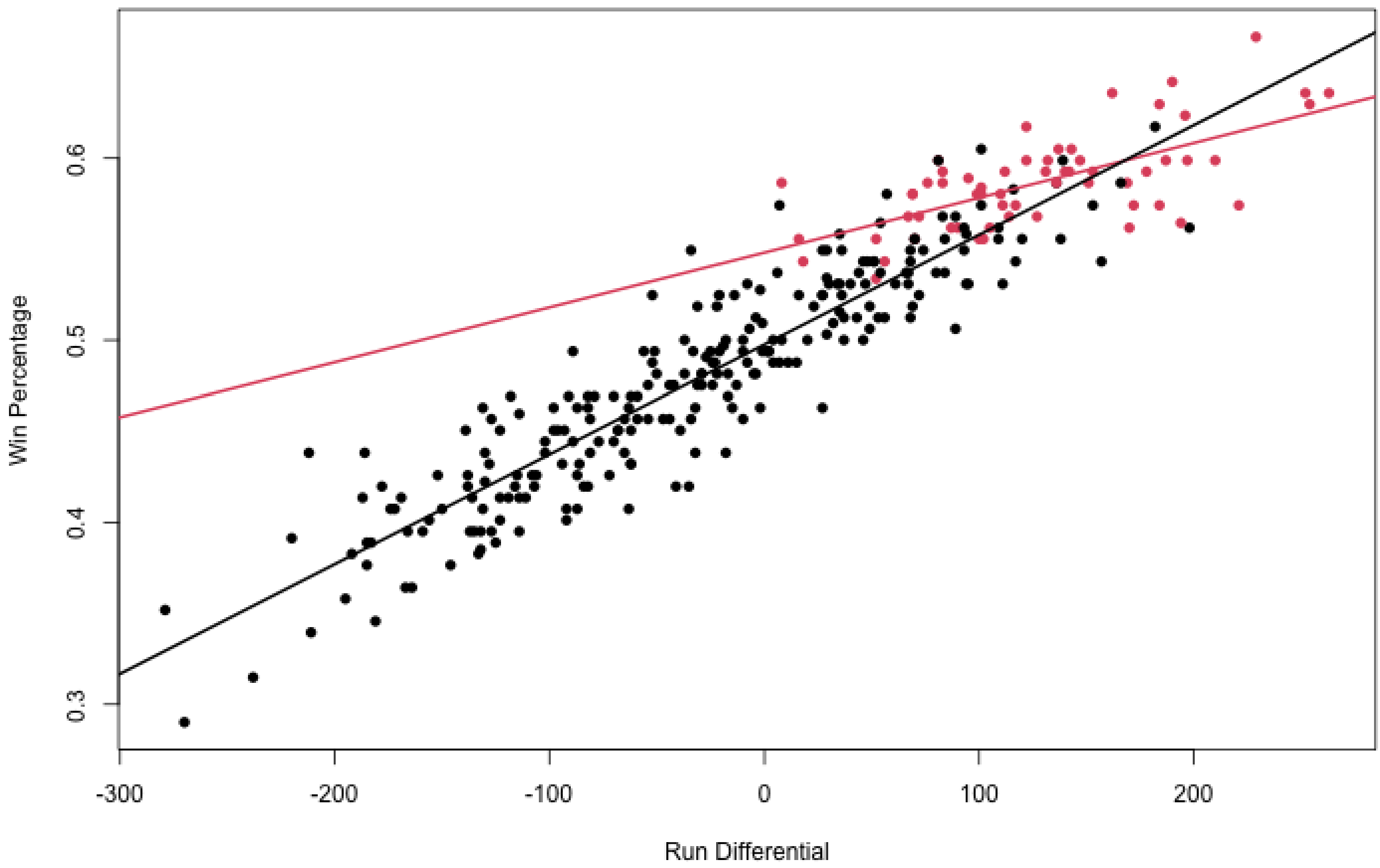
```
#> # A tibble: 300 x 56
#>    yearID lgID  teamID franchID divID  Rank     G Ghome     W     L DivWin WCWin
#>     <dbl> <chr> <chr>  <chr>    <chr> <dbl> <dbl> <dbl> <dbl> <dbl> <chr>  <chr>
#>  1   2009 NL    ARI    ARI      W         5   162    81    70    92 N      N
#>  2   2009 NL    ATL    ATL      E         3   162    81    86    76 N      N
#>  3   2009 AL    BAL    BAL      E         5   162    81    64    98 N      N
#>  4   2009 AL    BOS    BOS      E         2   162    81    95    67 N      Y
#>  5   2009 AL    CHA    CHW      C         3   162    81    79    83 N      N
#>  6   2009 NL    CHN    CHC      C         2   161    80    83    78 N      N
#>  7   2009 NL    CIN    CIN      C         4   162    81    78    84 N      N
#>  8   2009 AL    CLE    CLE      C         4   162    81    65    97 N      N
#>  9   2009 NL    COL    COL      W         2   162    81    92    70 N      Y
#> 10   2009 AL    DET    DET      C         2   163    81    86    77 N      N
#> # … with 290 more rows, and 44 more variables: LgWin <chr>, WSWin <chr>,
#> #   R <dbl>, AB <dbl>, H <dbl>, X2B <dbl>, X3B <dbl>, HR <dbl>, BB <dbl>,
#> #   SO <dbl>, SB <dbl>, CS <dbl>, HBP <dbl>, SF <dbl>, RA <dbl>, ER <dbl>,
#> #   ERA <dbl>, CG <dbl>, SHO <dbl>, SV <dbl>, IPouts <dbl>, HA <dbl>,
#> #   HRA <dbl>, BBA <dbl>, SOA <dbl>, E <dbl>, DP <dbl>, FP <dbl>, name <chr>,
#> #   park <chr>, attendance <dbl>, BPF <dbl>, PPF <dbl>, teamIDBR <chr>,
#> #   teamIDlahman45 <chr>, teamIDretro <chr>, TB <dbl>, WinPct <dbl>, rpg <dbl>,
#> #   hrpg <dbl>, tbpg <dbl>, kpg <dbl>, k2bb <dbl>, whip <dbl>
```

Plot comparison

Using `ggplot()`



Using `plot()`



Win Percentage
0.3
0.4
0.5
0.6
-300
-200
-100
0
100
200
Run Differential

Code comparison

Using `ggplot()`

```r
ggplot(teams, mapping = aes(x = R - RA, y = WinPct, color = DivWin)) +
  geom_point() +
  geom_smooth(method = "lm", se = FALSE) +
  labs(x = "Win Percentage", y = "Run Differential")
```

Using `plot()`

```r
teams$RD <- teams$R - teams$RA
teams_div <- teams[teams$DivWin == "Y", ]
teams_no_div <- teams[teams$DivWin == "N", ]

mod1 <- lm(WinPct ~ RD, data = teams_div)
mod2 <- lm(WinPct ~ RD, data = teams_no_div)

plot(x = (teams$R - teams$RA), y = teams$WinPct,
     col = adjustcolor(as.integer(factor(teams$DivWin))),
     pch = 16,
     xlab = "Run Differential",
     ylab = "Win Percentage")
abline(mod1, col = 2, lwd=2)
abline(mod2, col = 1, lwd=2)
```

What's in a `ggplot()`?

Terminology

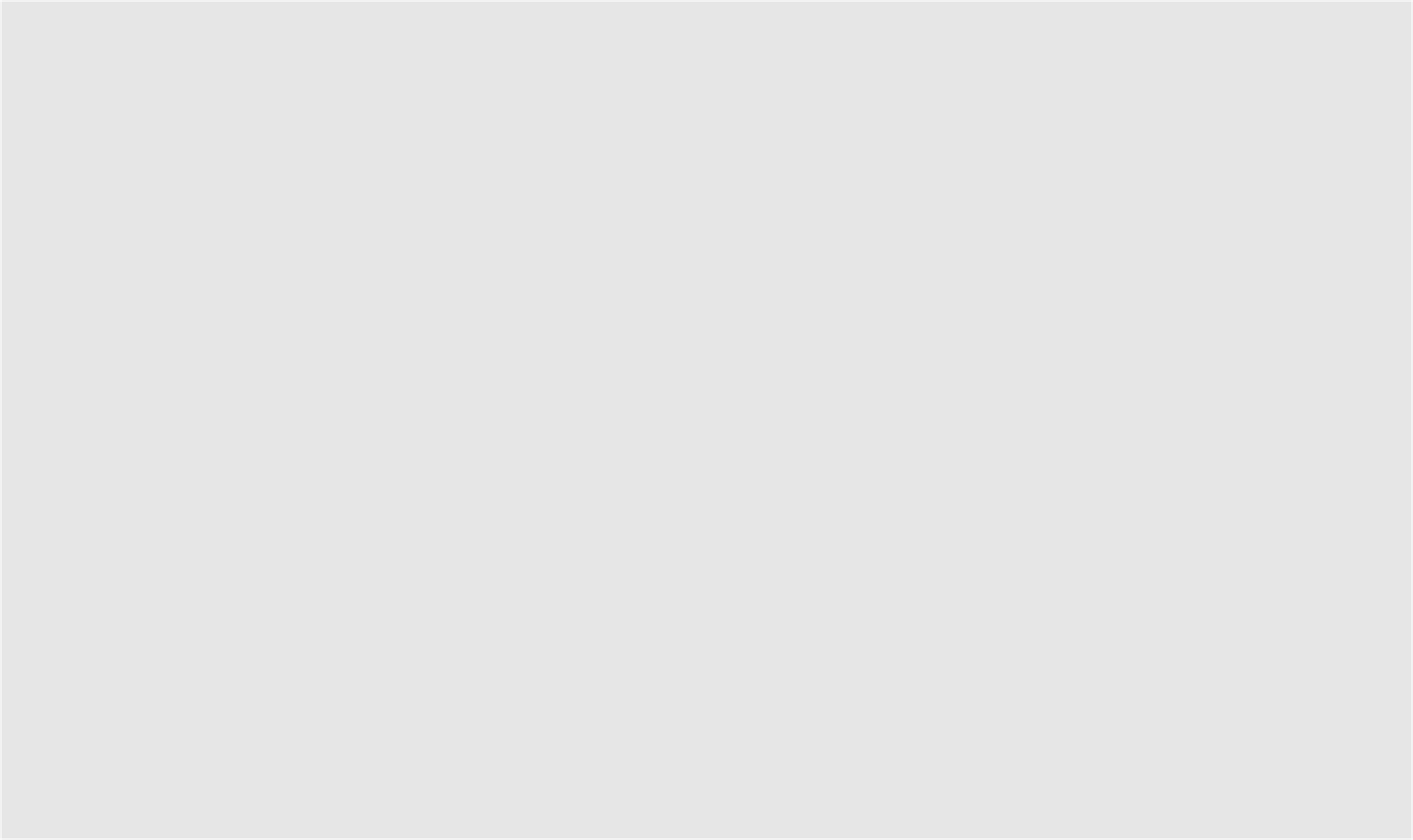
A statistical graphic is a...

- mapping of **data**
- which may be **statistically transformed** (summarized, log-transformed, etc.)
- to **aesthetic attributes** (color, size, xy-position, etc.)
- using **geometric objects** (points, lines, bars, etc.)
- and mapped onto a specific **facet** and **coordinate system.**

What do I "need"?

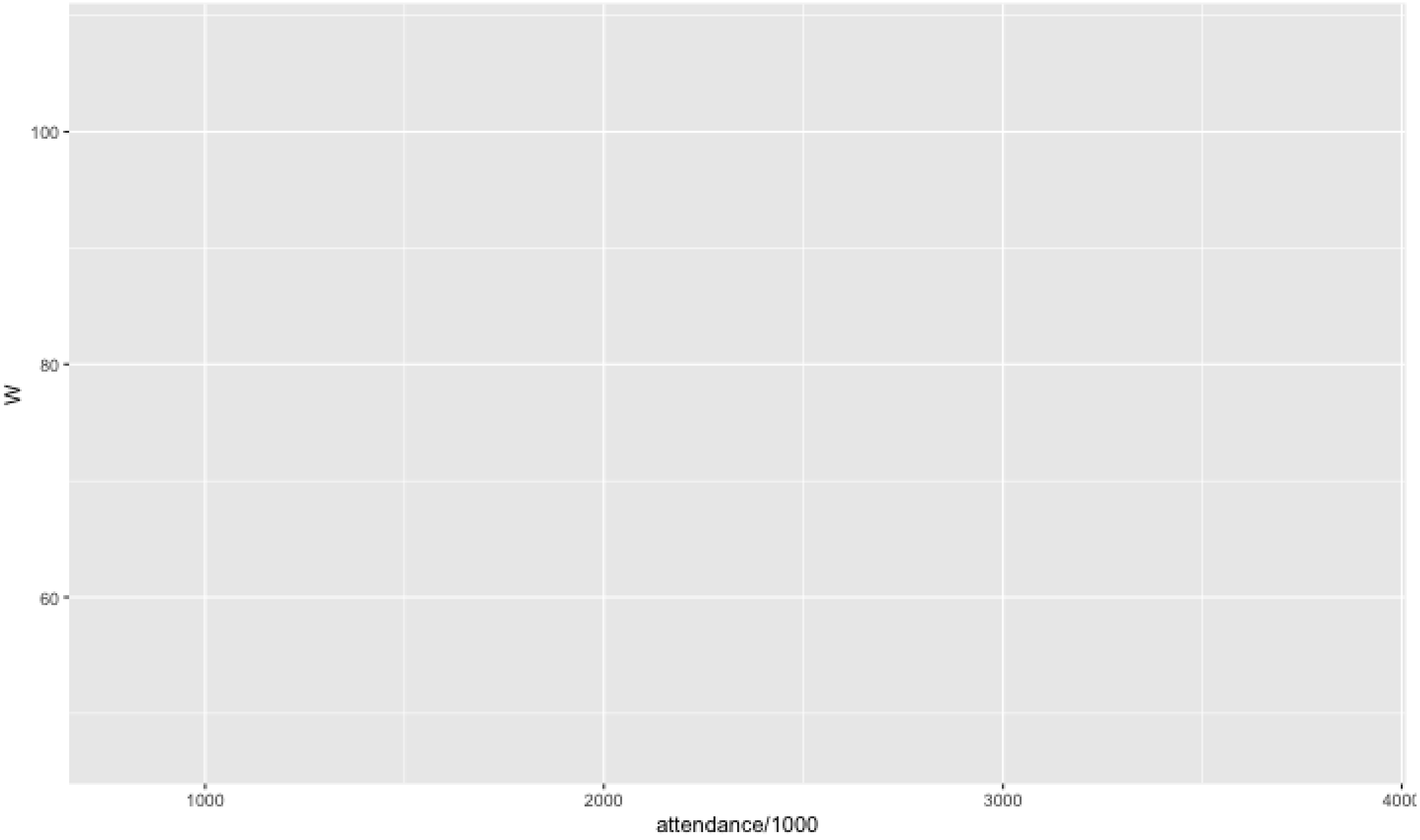
1) Some data (preferably in a data frame)

```
ggplot(data = teams)
```



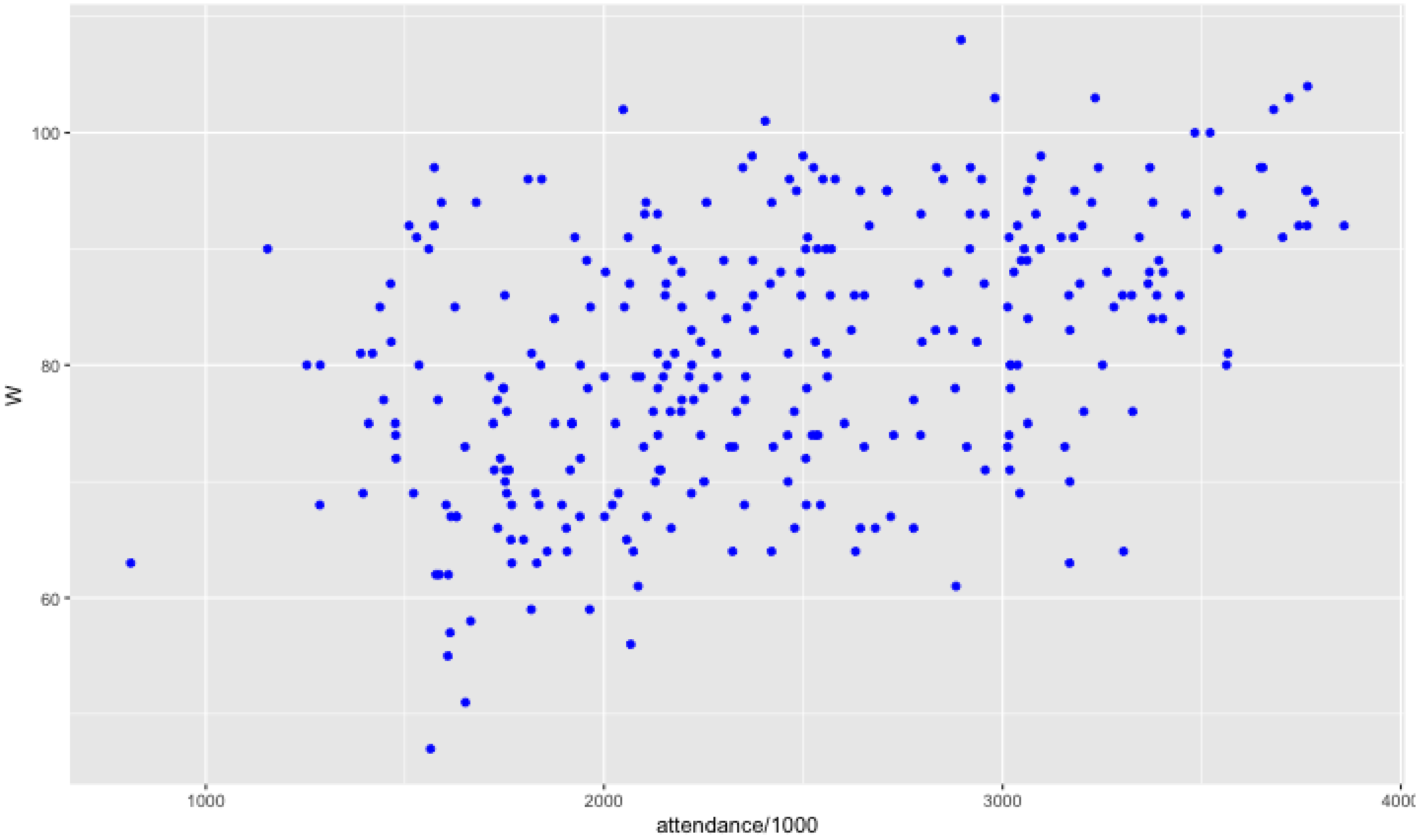
2) A set of variable mappings

```
ggplot(data = teams, mapping = aes(x = attendance / 1000, y = W))
```



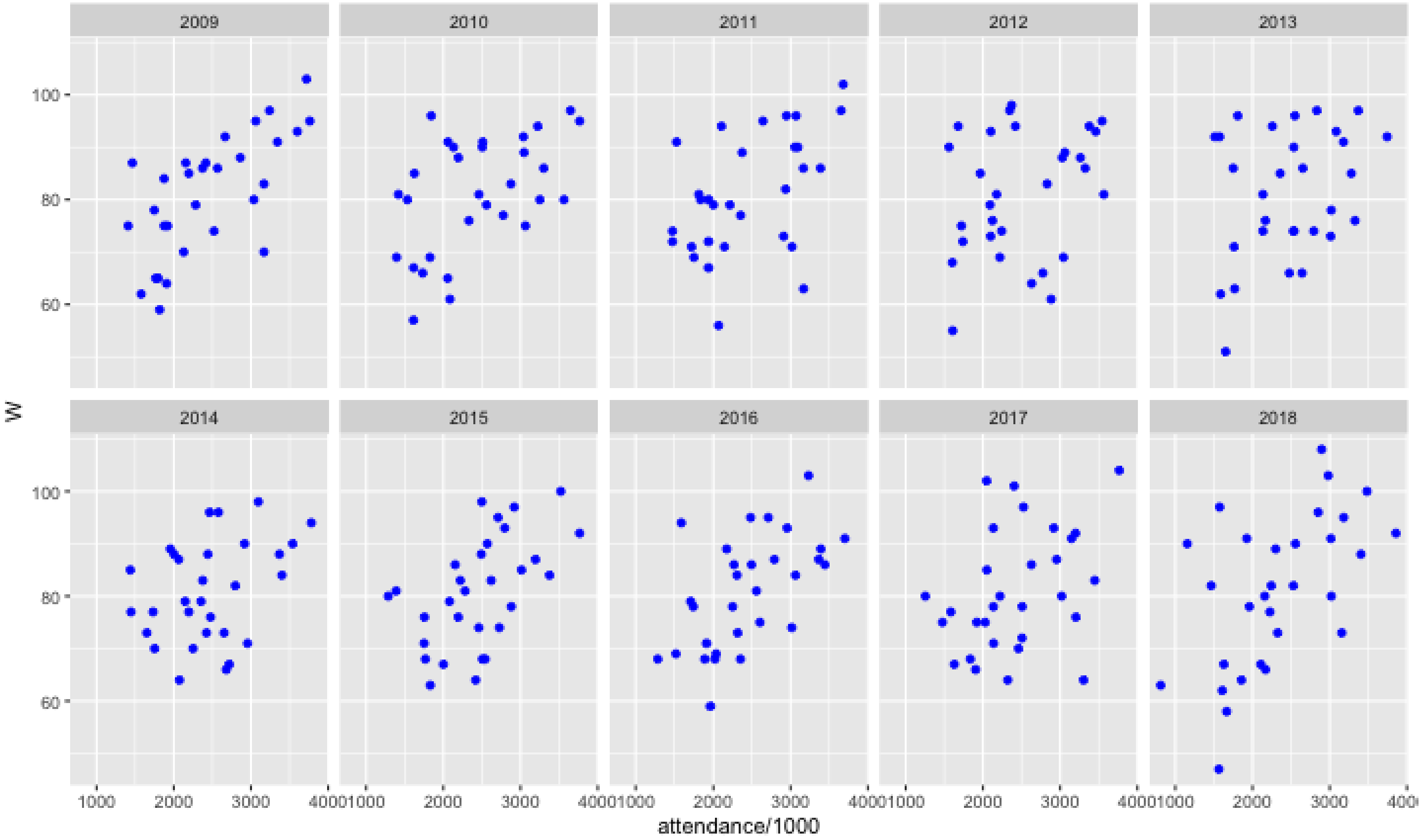
3) A geom with arguments, or multiple geoms with arguments connected by +

```
ggplot(data = teams, mapping = aes(x = attendance / 1000, y = W)) +
  geom_point(color = "blue")
```



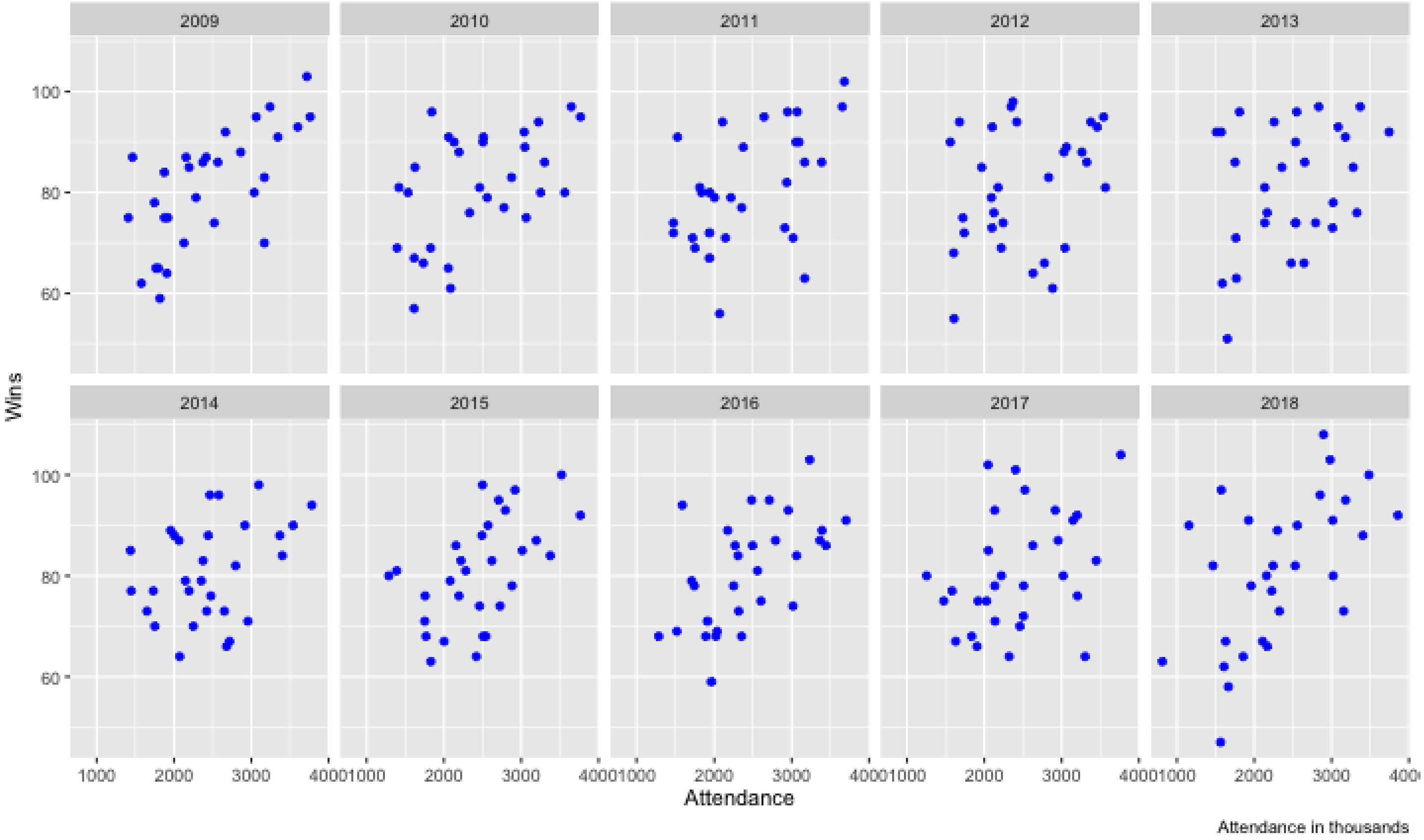
4) Some options on changing scales or adding facets

```
ggplot(data = teams, mapping = aes(x = attendance / 1000, y = W)) +
  geom_point(color = "blue") +
  facet_wrap(~yearID, nrow = 2)
```



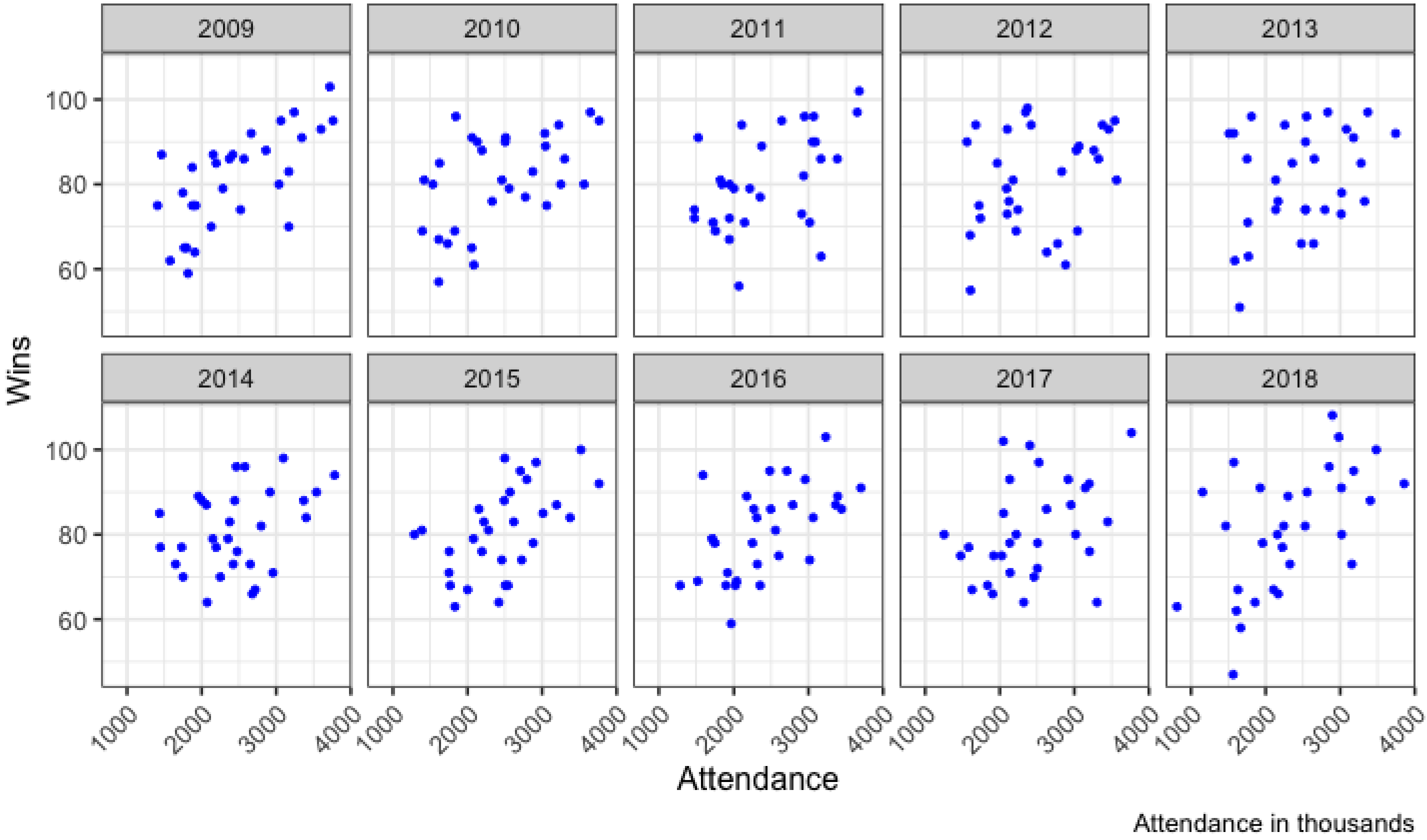
5) Some labels

```
ggplot(data = teams, mapping = aes(x = attendance / 1000, y = W)) +
  geom_point(color = "blue") +
  facet_wrap(~yearID, nrow = 2) +
  labs(x = "Attendance", y = "Wins", caption = "Attendance in thousands")
```



6) Other options

```
ggplot(data = teams, mapping = aes(x = attendance / 1000, y = W)) +
  geom_point(color = "blue") +
  facet_wrap(~yearID, nrow = 2) +
  labs(x = "Attendance", y = "Wins", caption = "Attendance in thousands")
  theme_bw(base_size = 16) +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```



Anatomy of a ggplot

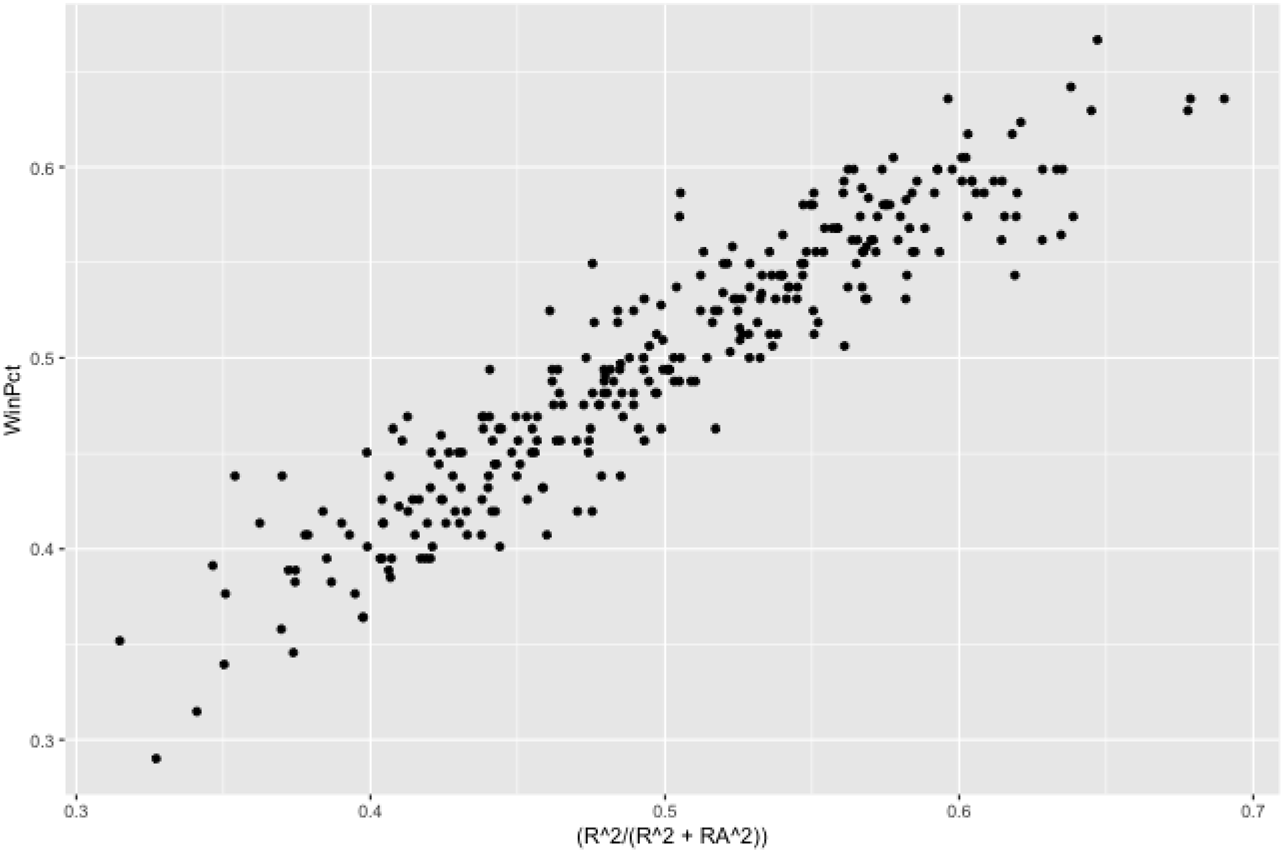
```
ggplot(
  data = [dataframe],

  aes(
    x = [var_x], y = [var_y],
    color = [var_for_color],
    fill  = [var_for_fill],
    shape = [var_for_shape],
    size  = [var_for_size],
    alpha = [var_for_alpha],
    ...#other aesthetics
  )
) +
  geom_<some_geom>([geom_arguments]) +
  ... # other geoms
  scale_<some_axis>_<some_scale>() +
  facet_<some_facet>([formula]) +
  ... # other options
```

To visualize multivariate relationships we can add variables to our visualization by specifying aesthetics: color, size, shape, linetype, alpha, or fill; we can also add facets based on variable levels.

Scatter plots

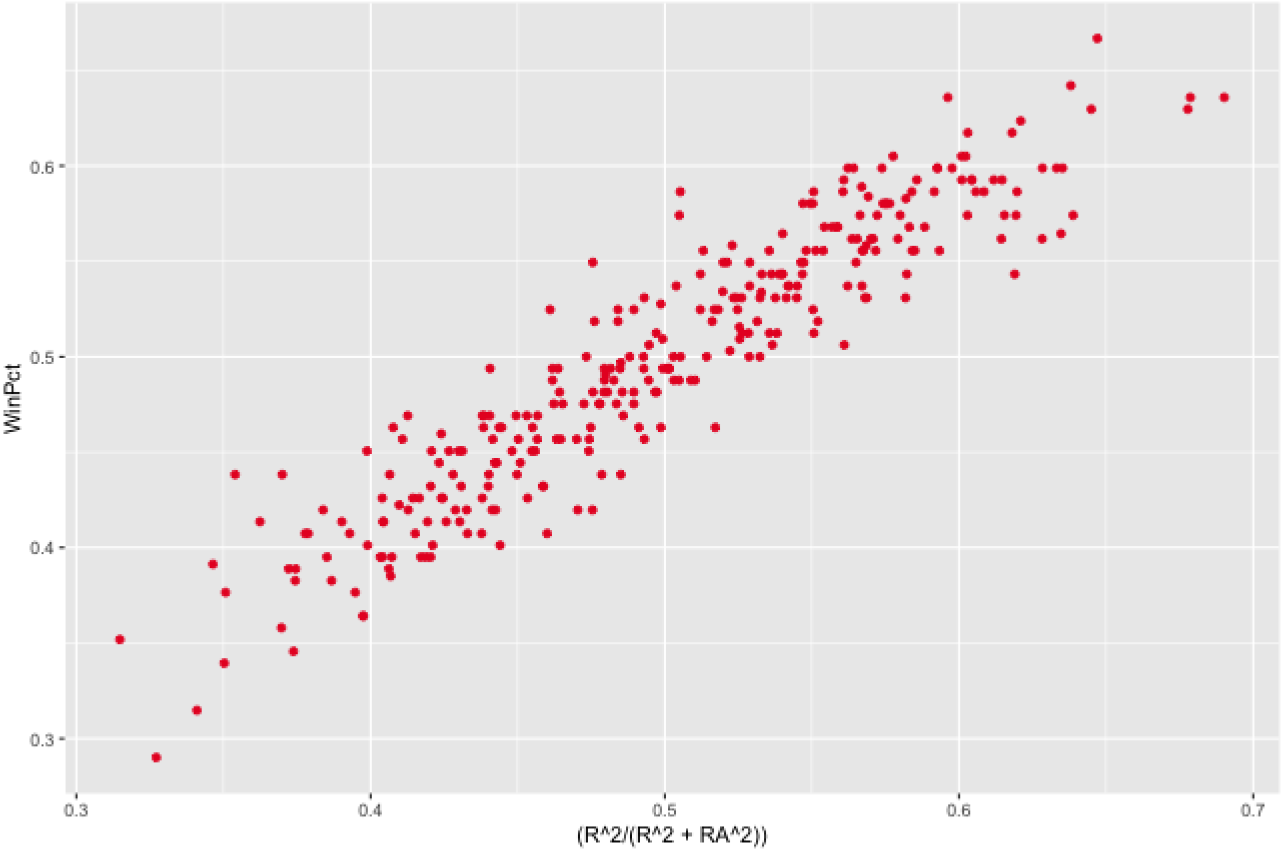
Base plot

```
ggplot(data = teams, mapping = aes(x = (R ^ 2 / (R ^ 2 + RA ^2 )), y = WinPct)) +
  geom_point()
```



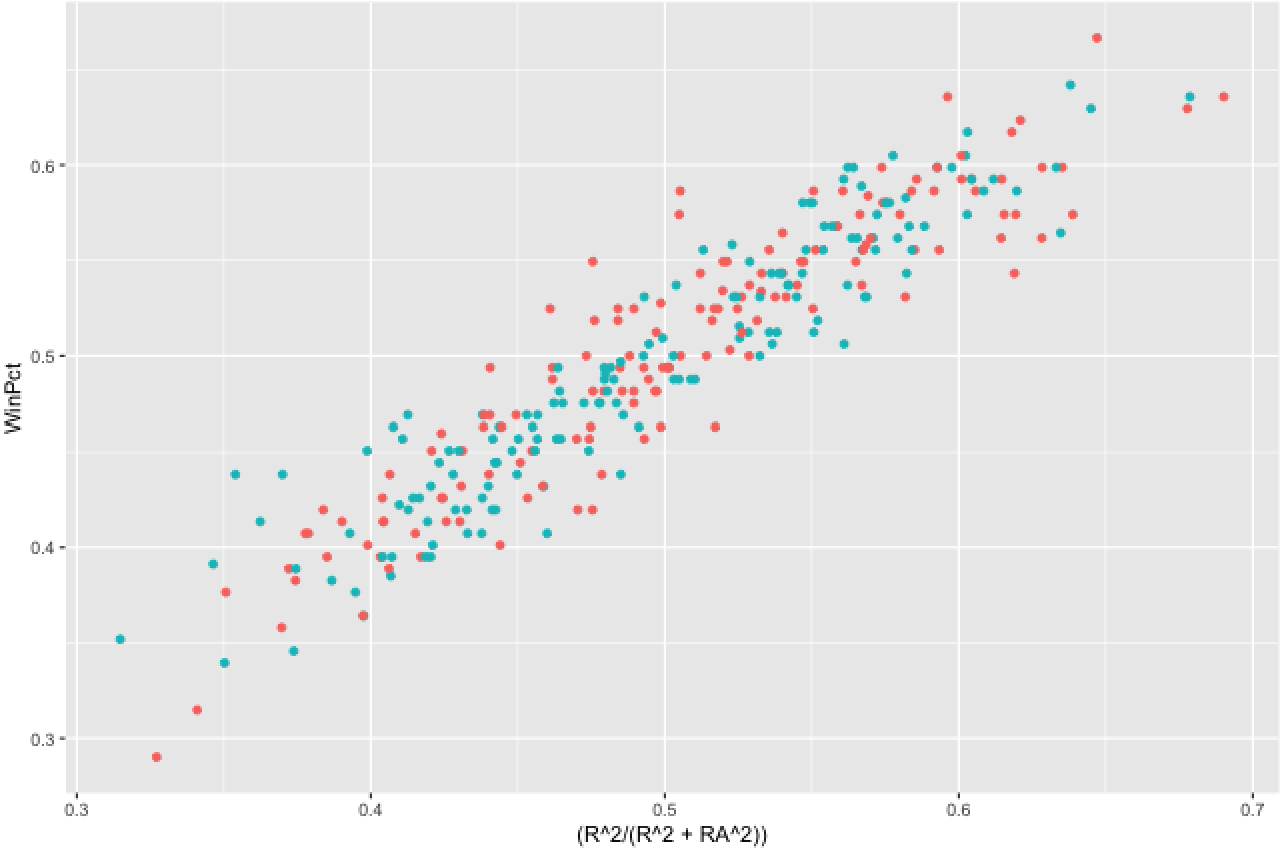
Altering aesthetic color

```
ggplot(data = teams, mapping = aes(x = (R ^ 2 / (R ^ 2 + RA ^2 )), y = WinPct)) +
  geom_point(color = "#E81828")
```



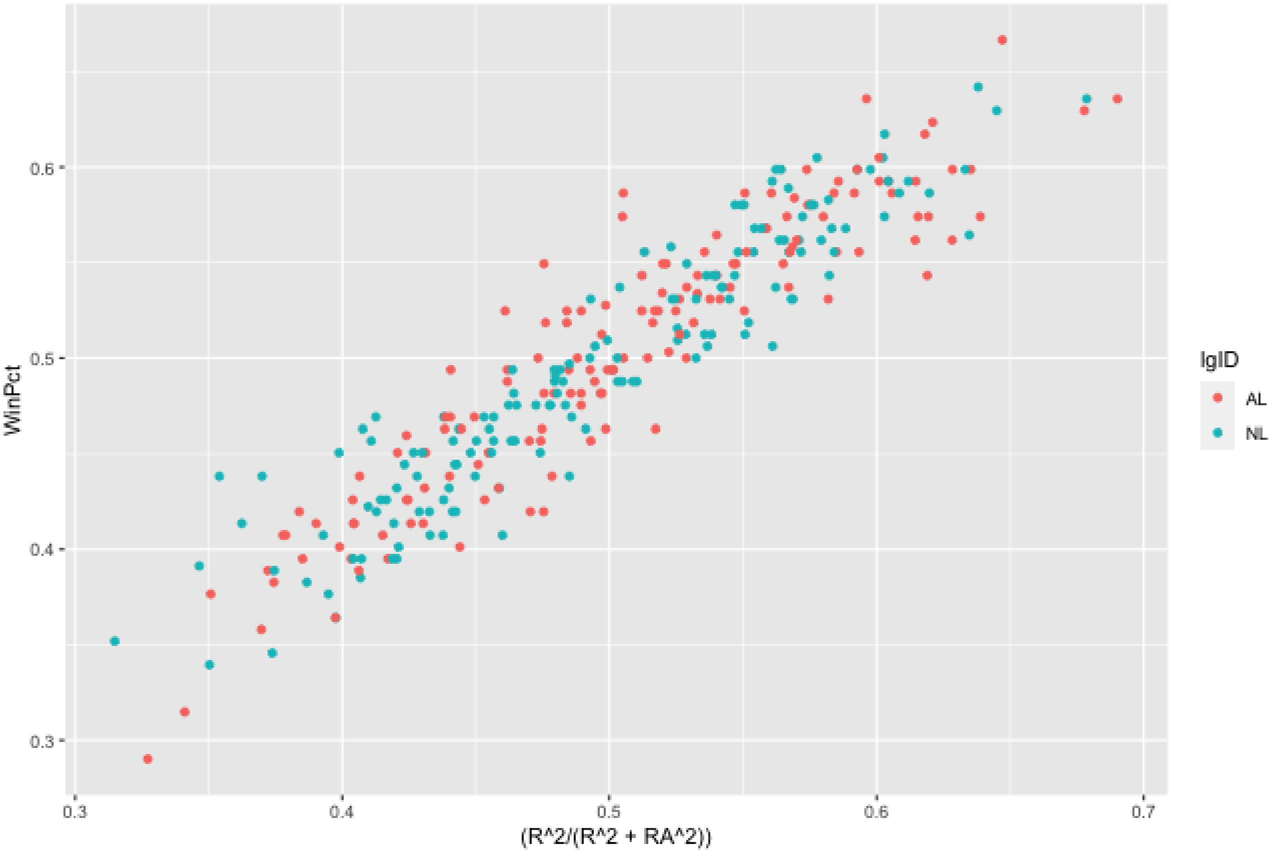
Altering aesthetic color

```
ggplot(data = teams, mapping = aes(x = (R ^ 2 / (R ^ 2 + RA ^2 )), y = WinPct, color = lgID)) +
  geom_point(show.legend = FALSE)
```



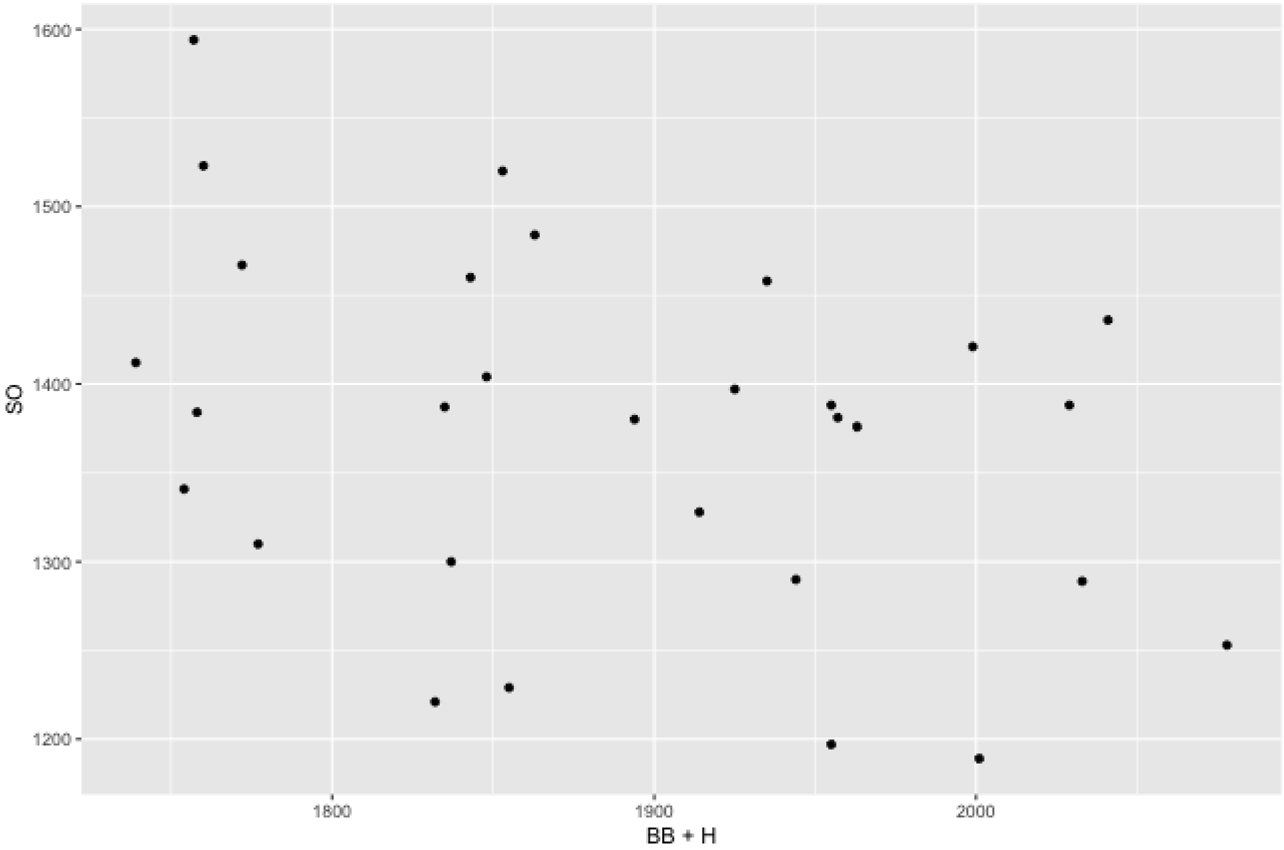
Altering aesthetic color

```
ggplot(data = teams, mapping = aes(x = (R ^ 2 / (R ^ 2 + RA ^2 )), y = WinPct, color = lgID)) +
  geom_point()
```



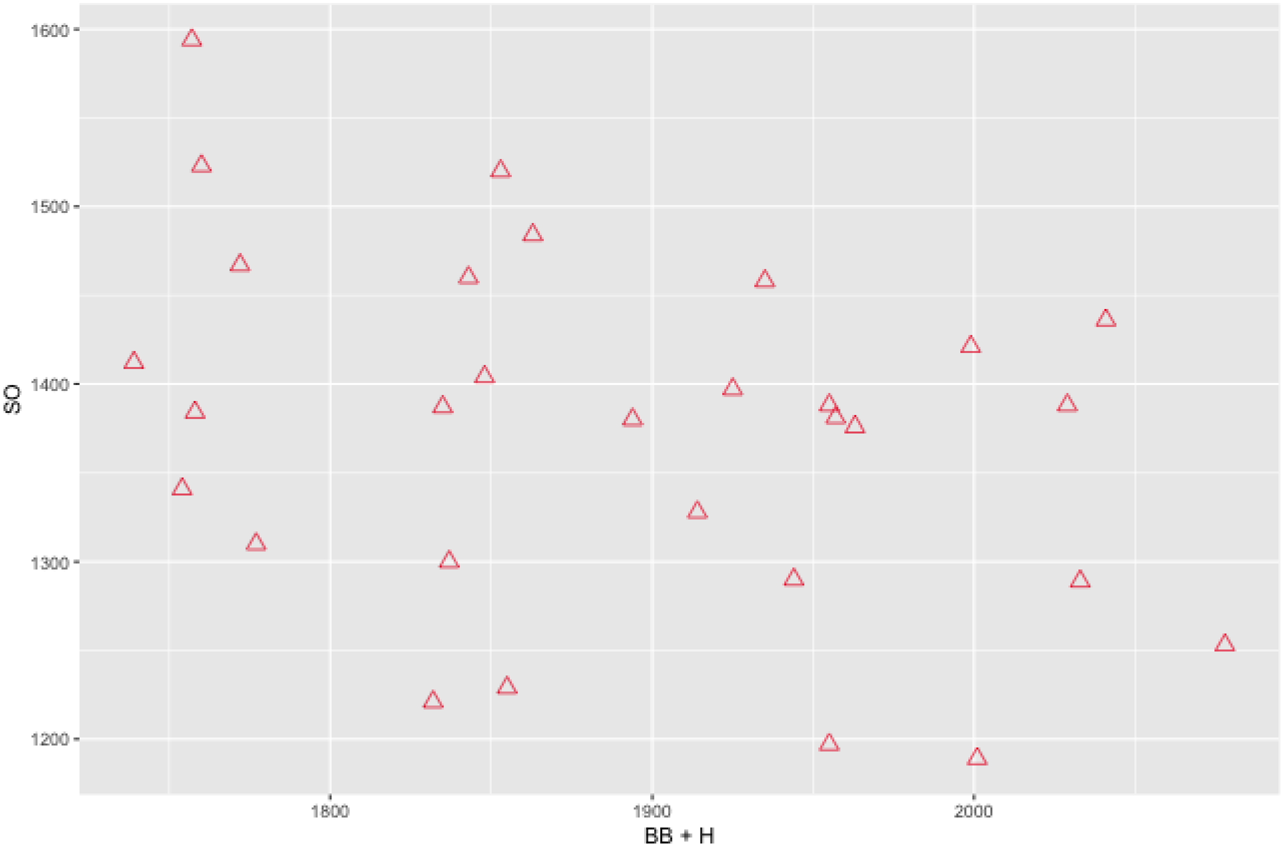
Base plot

```
ggplot(data = teams[teams$yearID == 2018, ], mapping = aes(x = BB + H, y = SO)) +
  geom_point()
```



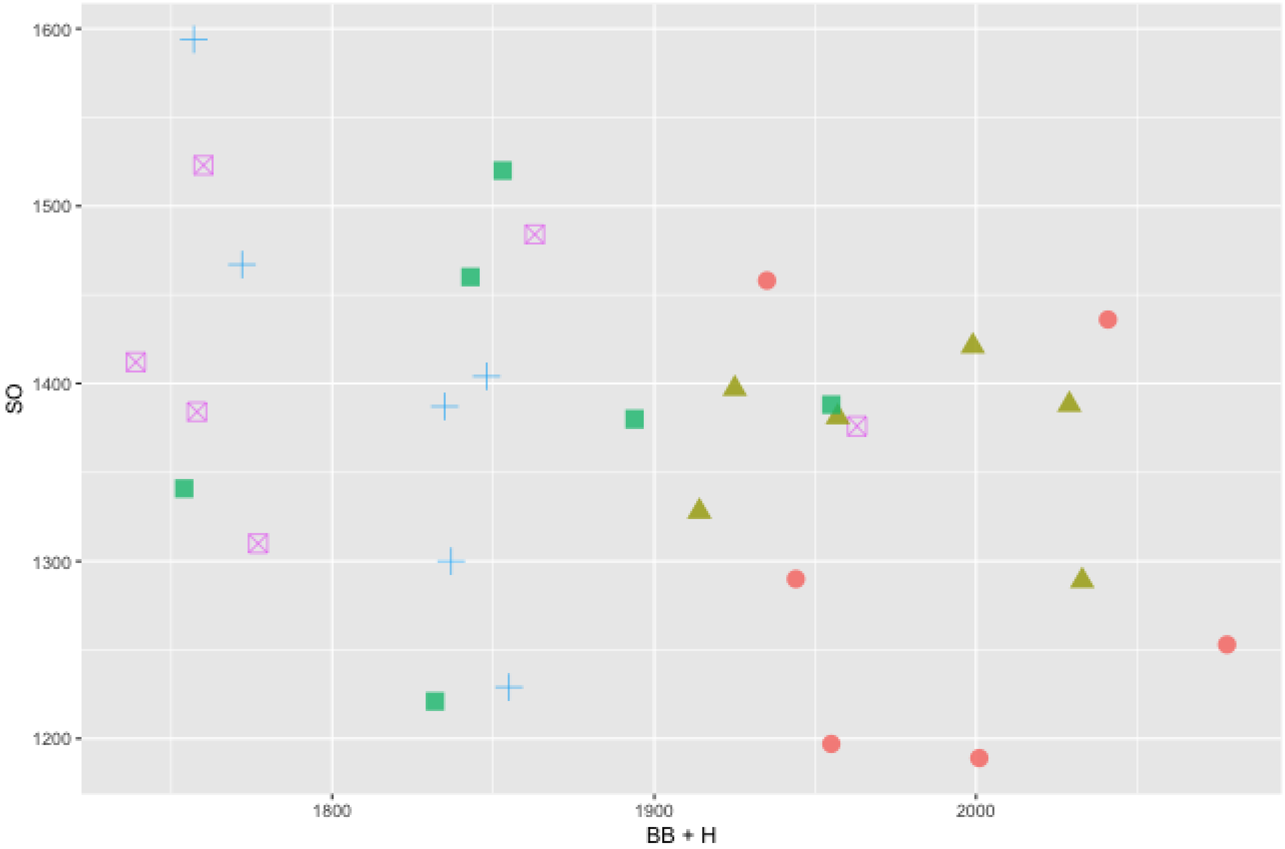
Altering multiple aesthetics

```
ggplot(data = teams[teams$yearID == 2018, ], mapping = aes(x = BB + H, y = SO)) +
  geom_point(size = 3, shape = 2, color = "#E81828")
```



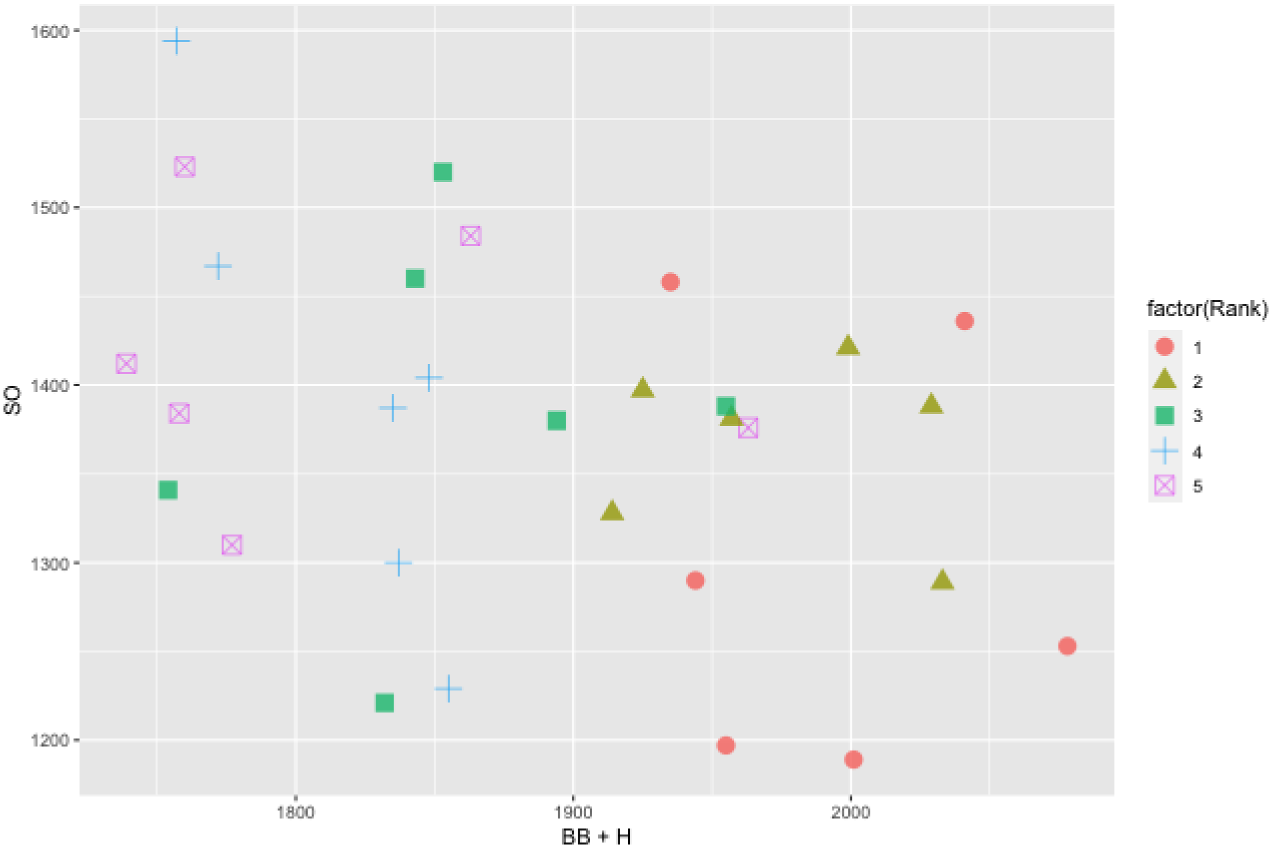
Altering multiple aesthetics

```
ggplot(data = teams[teams$yearID == 2018, ], mapping = aes(x = BB + H, y = SO,
                     color = factor(Rank), shape = factor(Rank))) +
  geom_point(size = 4, alpha = .8, show.legend = FALSE)
```



Altering multiple aesthetics

```
ggplot(data = teams[teams$yearID == 2018, ], mapping = aes(x = BB + H, y = SO,
                     color = factor(Rank), shape = factor(Rank)) +
  geom_point(size = 4, alpha = .8)
```



Inside or outside `aes()`?

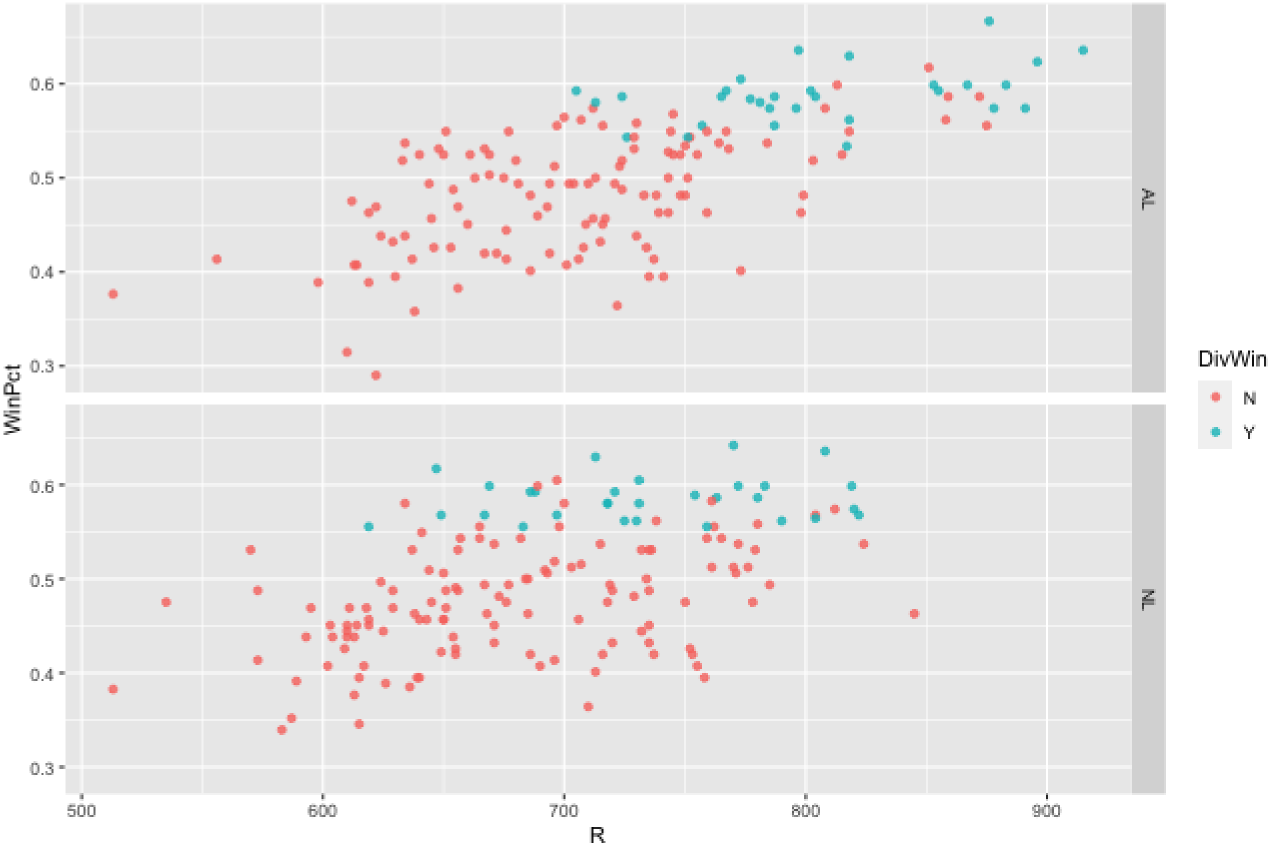
When does an aesthetic go inside function `aes()`?

- If you want an aesthetic to be reflective of a variable's values, it must go inside aes.
- If you want to set an aesthetic manually and not have it convey information about a variable, use the aesthetic's name outside of aes and set it to your desired value.

Aesthetics for continuous and discrete variables are measured on continuous and discrete scales, respectively.

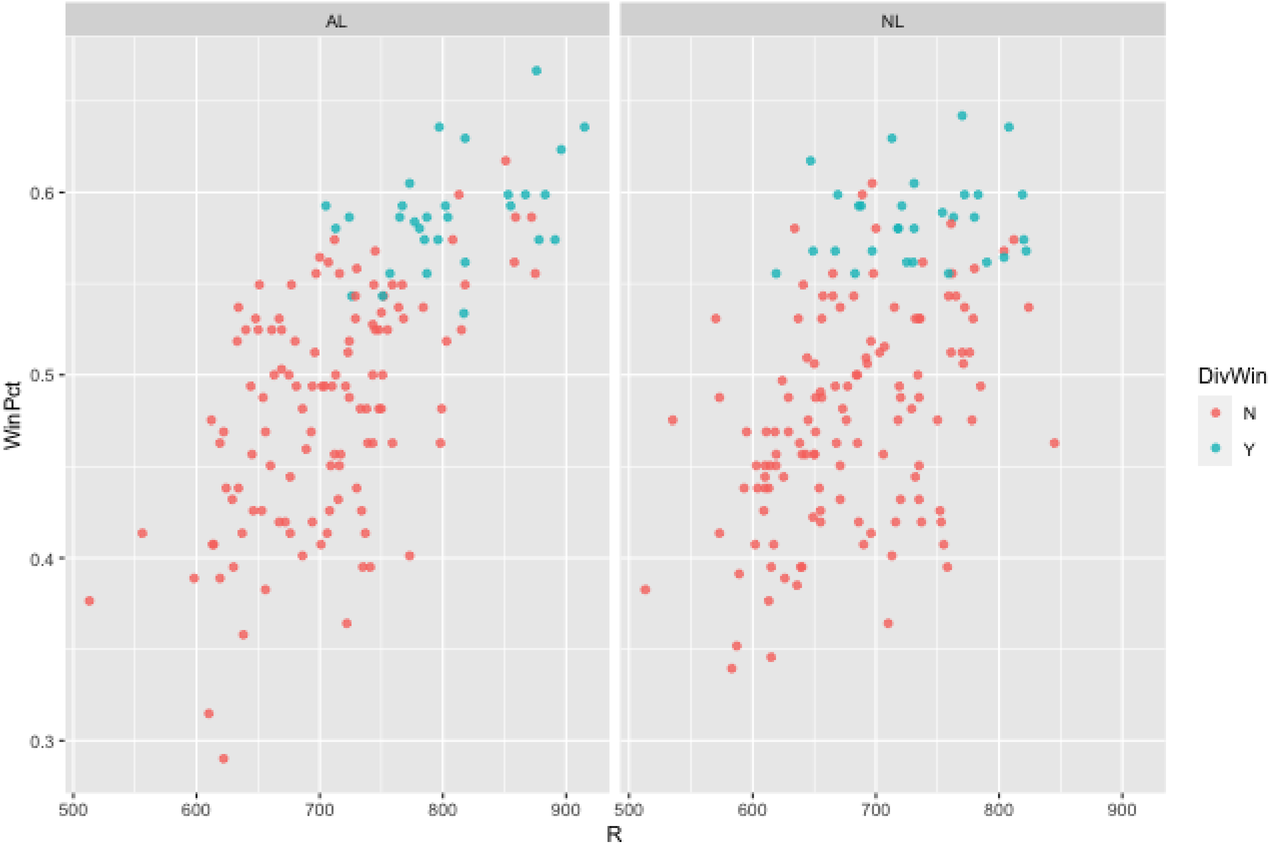
Faceting

```
ggplot(data = teams, mapping = aes(x = R, y = WinPct, color = DivWin)) +
  geom_point(alpha = .8) +
  facet_grid(lgID~ .)
```



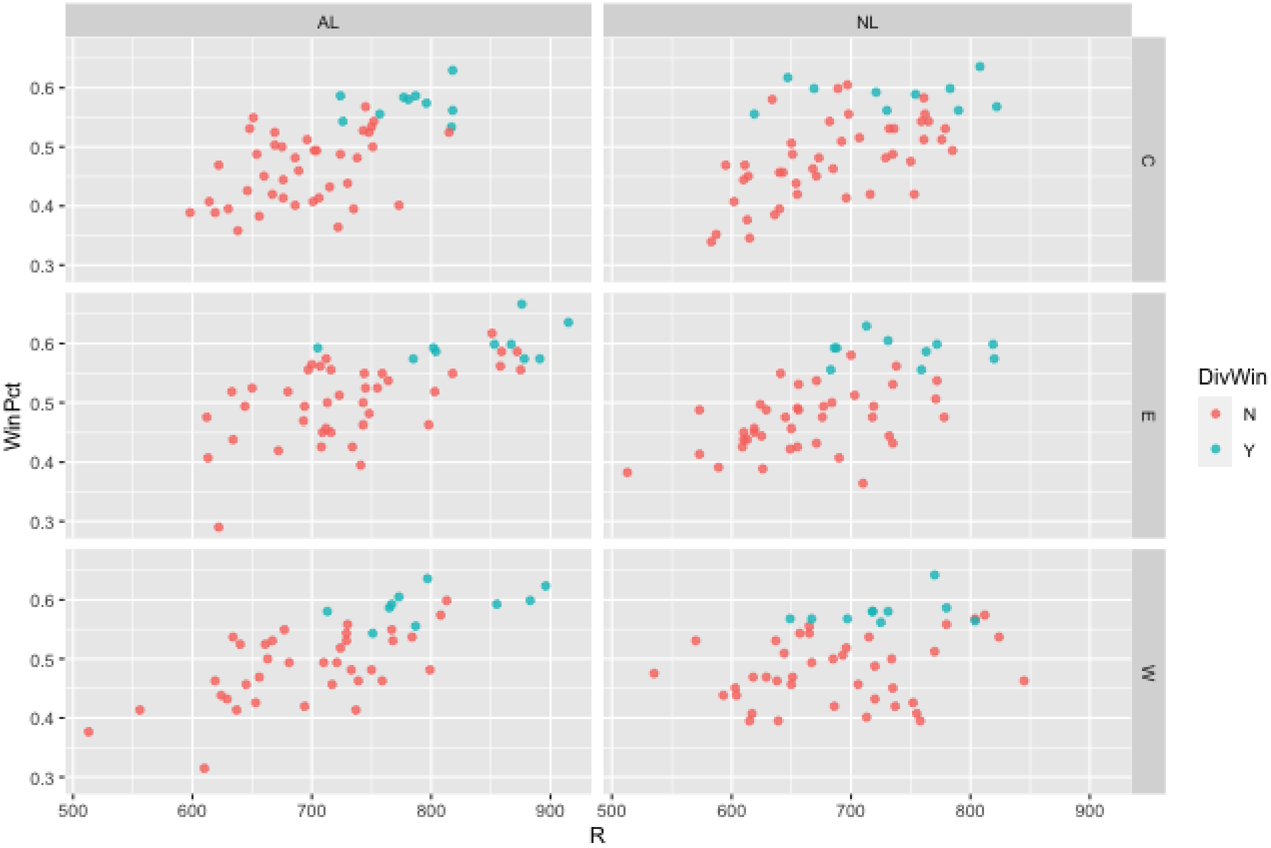
Faceting

```
ggplot(data = teams, mapping = aes(x = R, y = WinPct, color = DivWin)) +
  geom_point(alpha = .8) +
  facet_grid(. ~lgID)
```



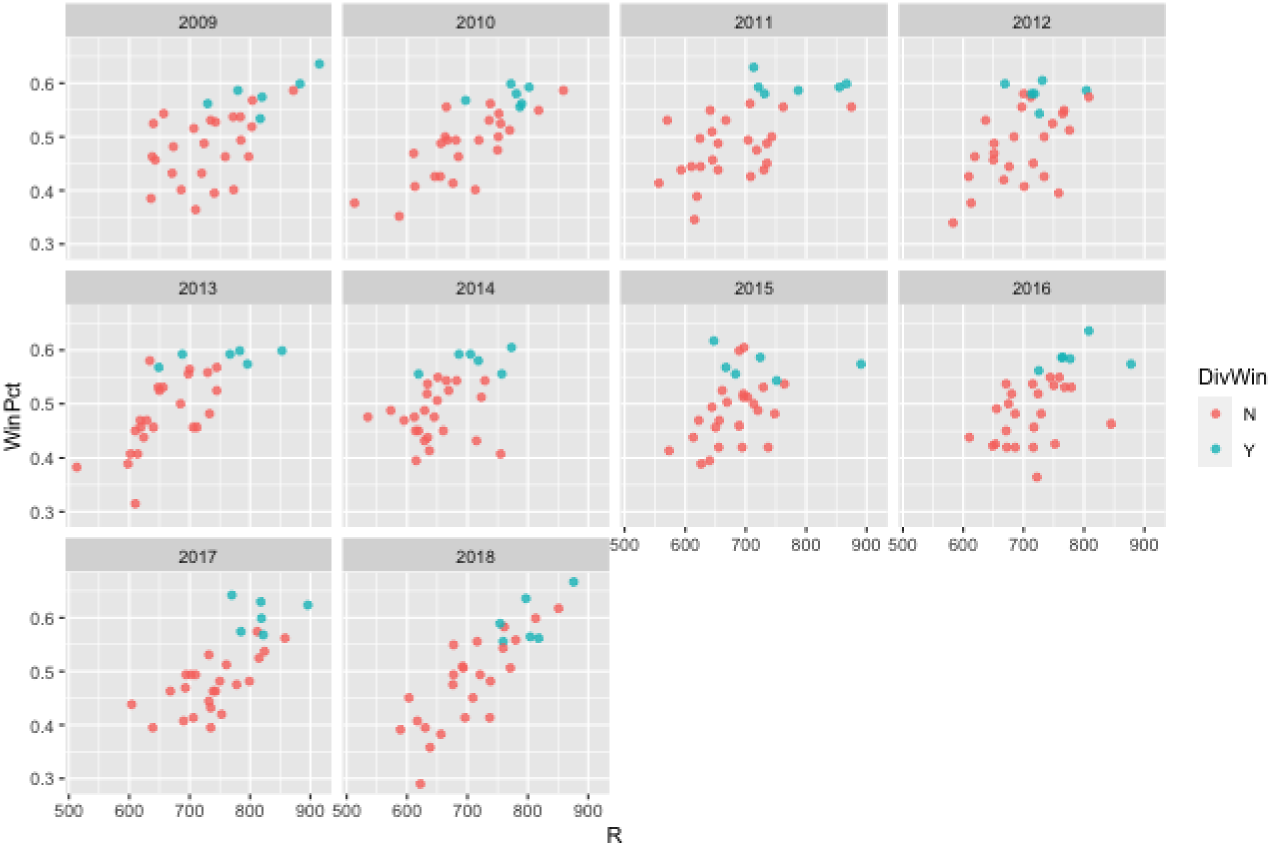
Faceting

```
ggplot(data = teams, mapping = aes(x = R, y = WinPct, color = DivWin)) +
  geom_point(alpha = .8) +
  facet_grid(divID~lgID)
```



Faceting

```
ggplot(data = teams, mapping = aes(x = R, y = WinPct, color = DivWin)) +
  geom_point(alpha = .8) +
  facet_wrap(~yearID)
```

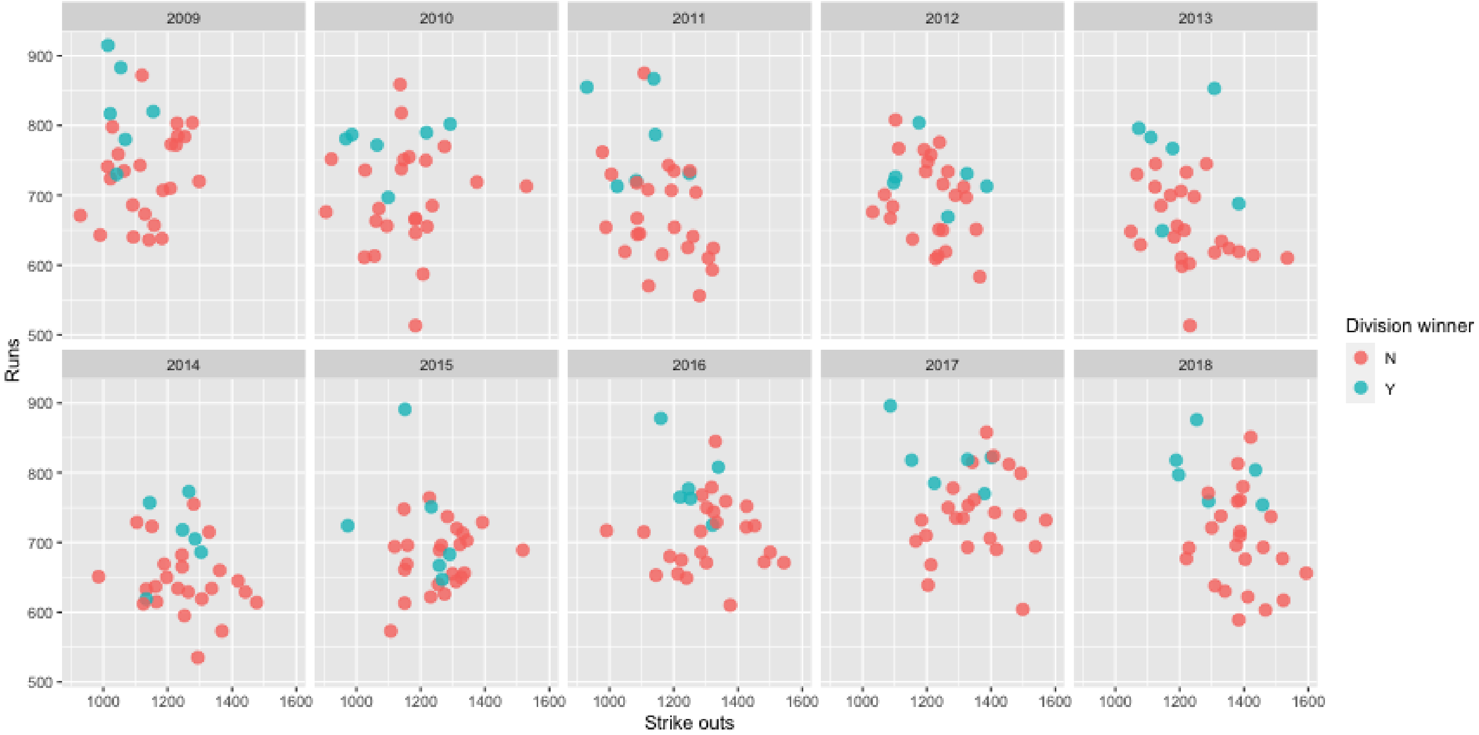


Facet grid or wrap?

- Use `facet_wrap()` to wrap a one dimensional sequence into two dimensional panels.
- Use `facet_grid()` when you have two discrete variables and you want panels of plots to represent all possible combinations.

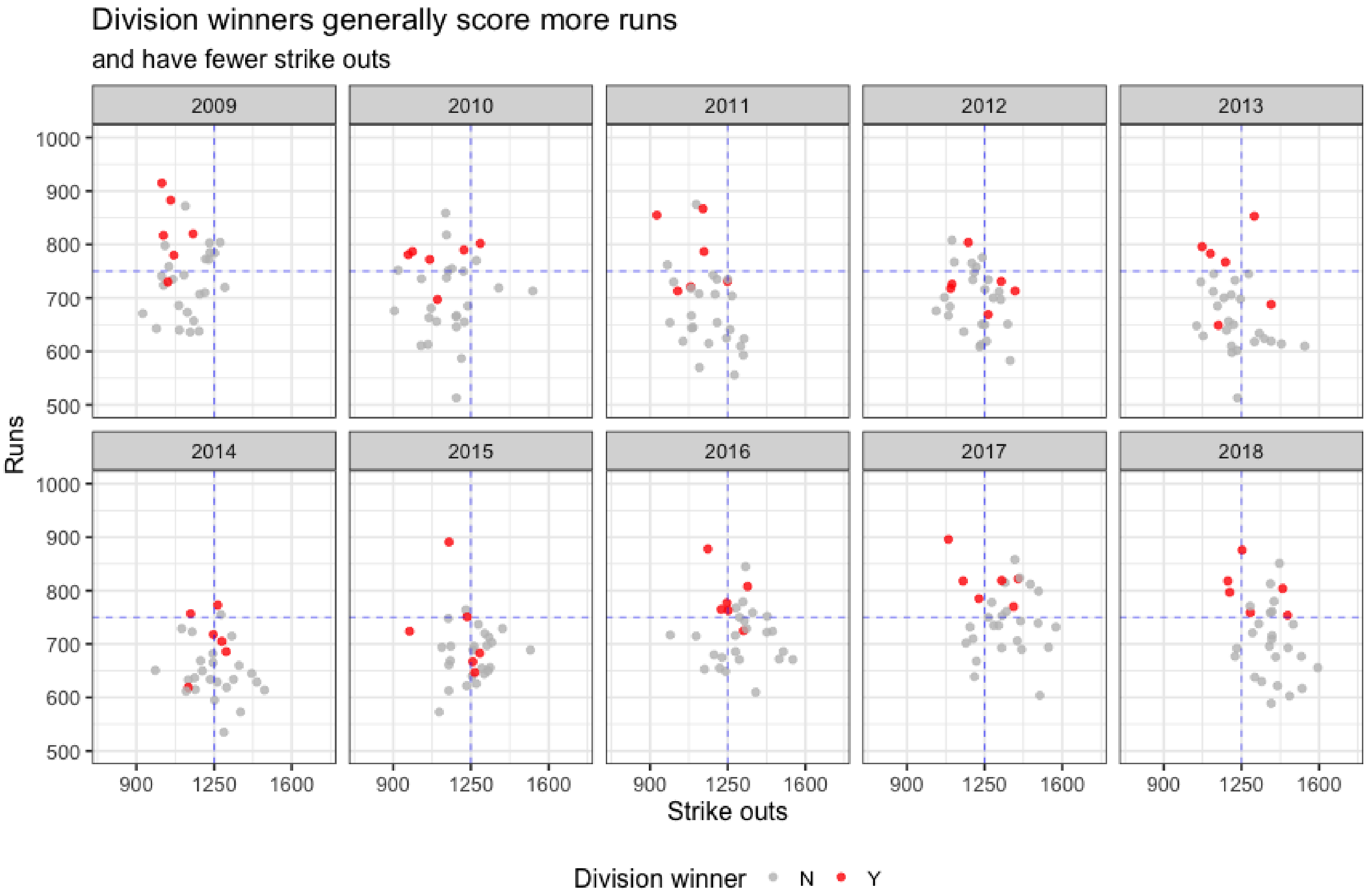
Exercise

Let's explore the relationship between runs and strikeouts for division winners and non-division winners. Use tibble `teams` to re-create the plot below.



How can we improve this visualization?

A more effective visualization



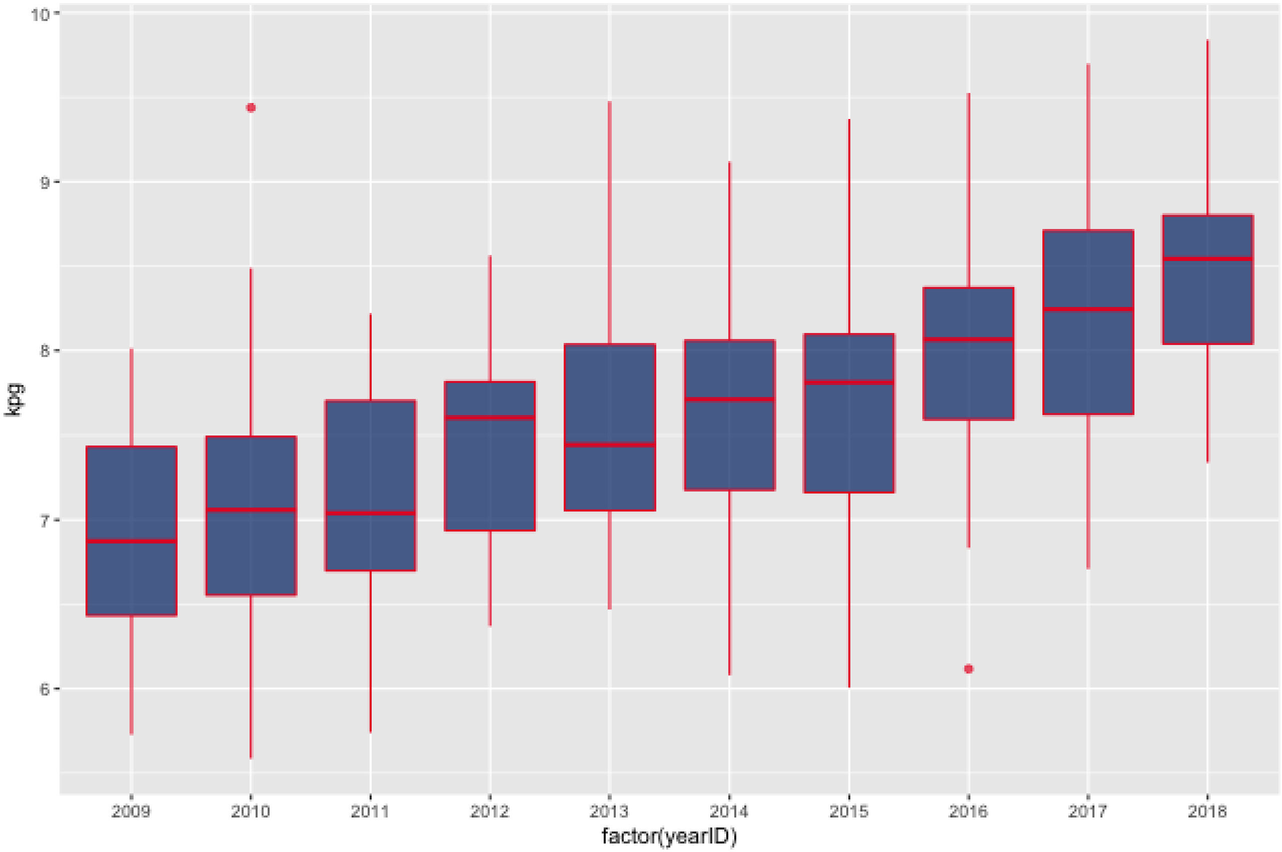
Other geoms

Caution

- The following plots are not well-polished. They are designed to demonstrate the various geoms and options that exist within `ggplot2`.
- You should always have a well-labelled and polished visualization if it will be seen by an outside audience.

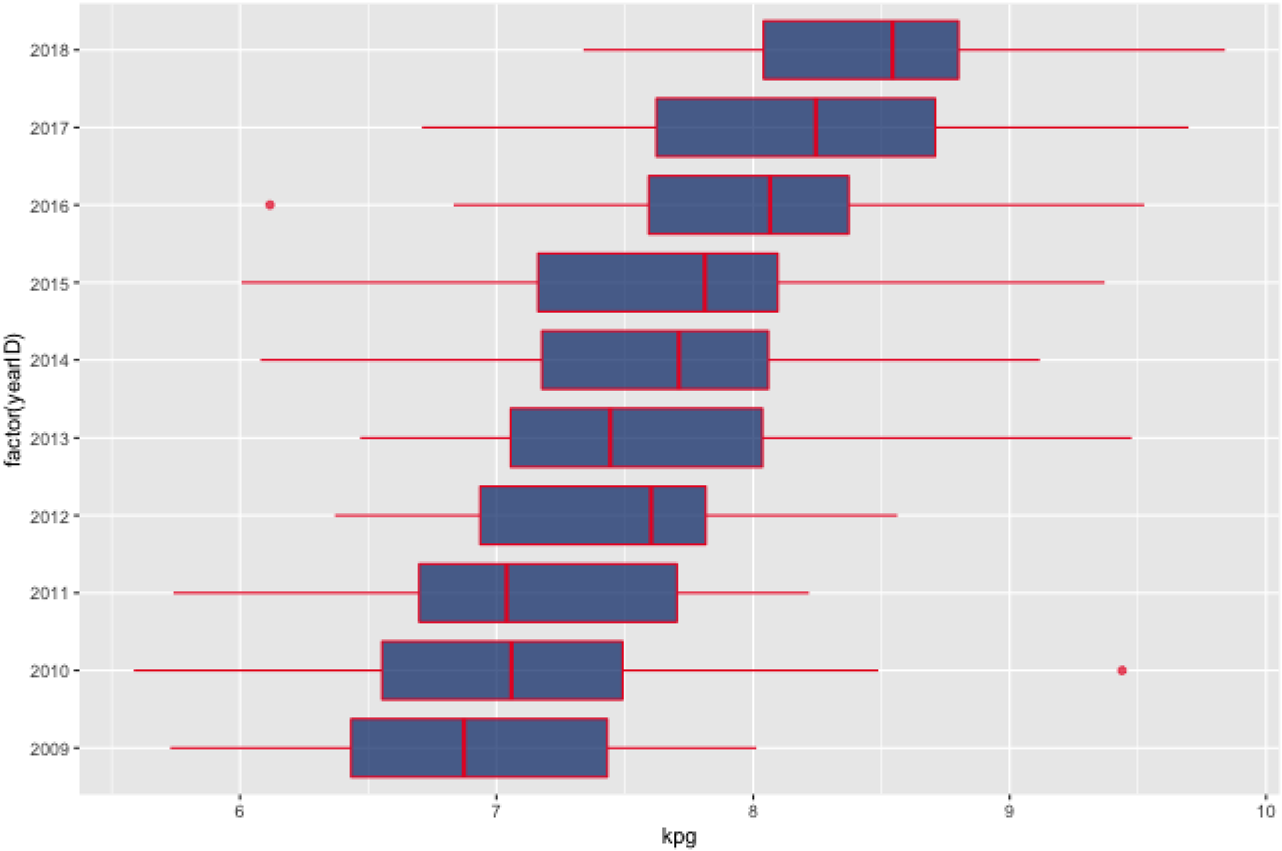
Box plots

```
ggplot(teams, mapping = aes(x = factor(yearID), y = kpg)) +
  geom_boxplot(color = "#E81828", fill = "#002D72", alpha = .7)
```



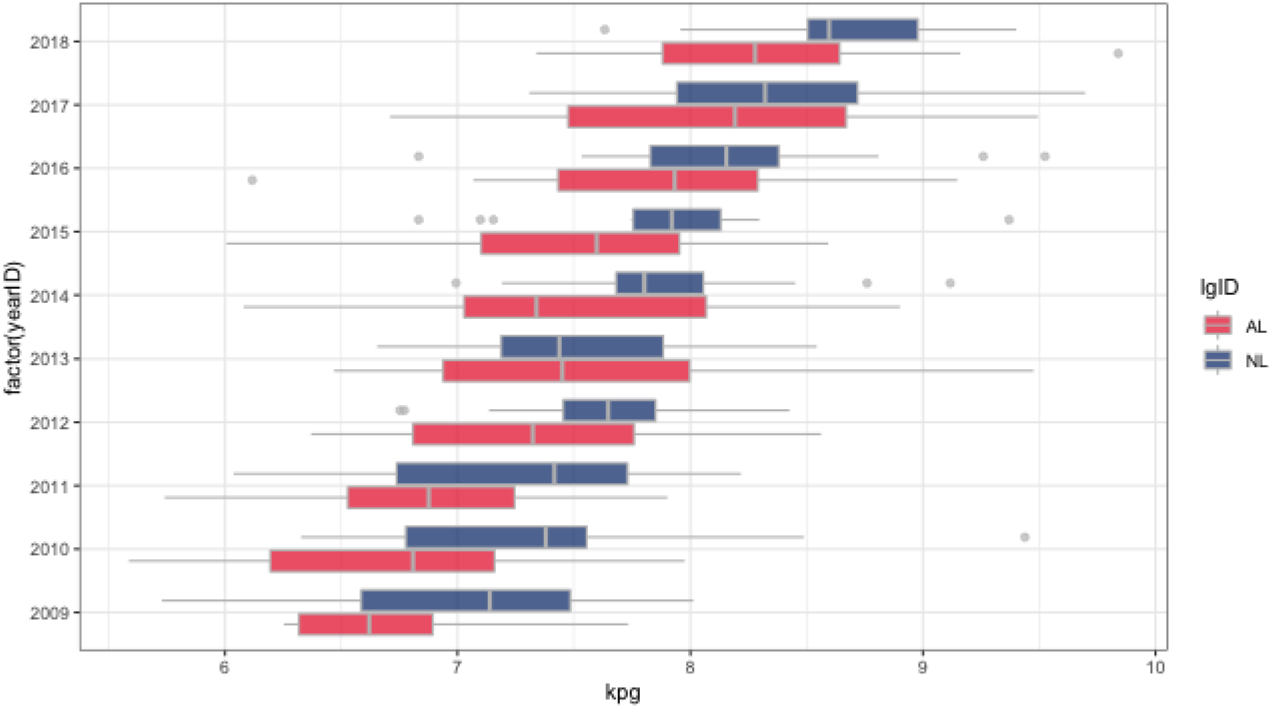
Box plots: flipped coordinates

```
ggplot(teams, mapping = aes(x = factor(yearID), y = kpg)) +
  geom_boxplot(color = "#E81828", fill = "#002D72", alpha = .7) +
  coord_flip()
```



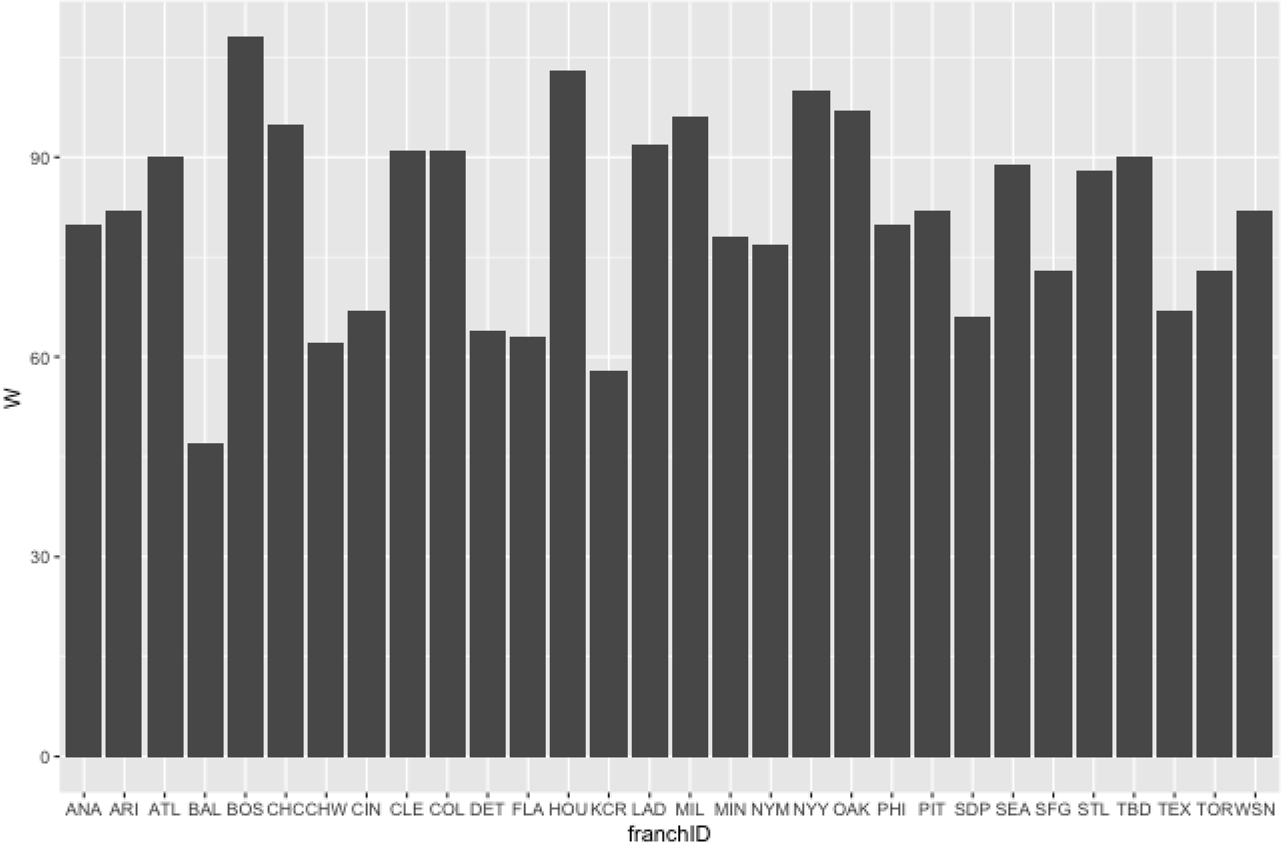
Box plots: custom colors

```
ggplot(teams, mapping = aes(x = factor(yearID), y = kpg, fill = lgID)) +
  geom_boxplot(color = "grey", alpha = .7) +
  scale_fill_manual(values = c("#E81828", "#002D72")) +
  coord_flip() +
  theme_bw()
```



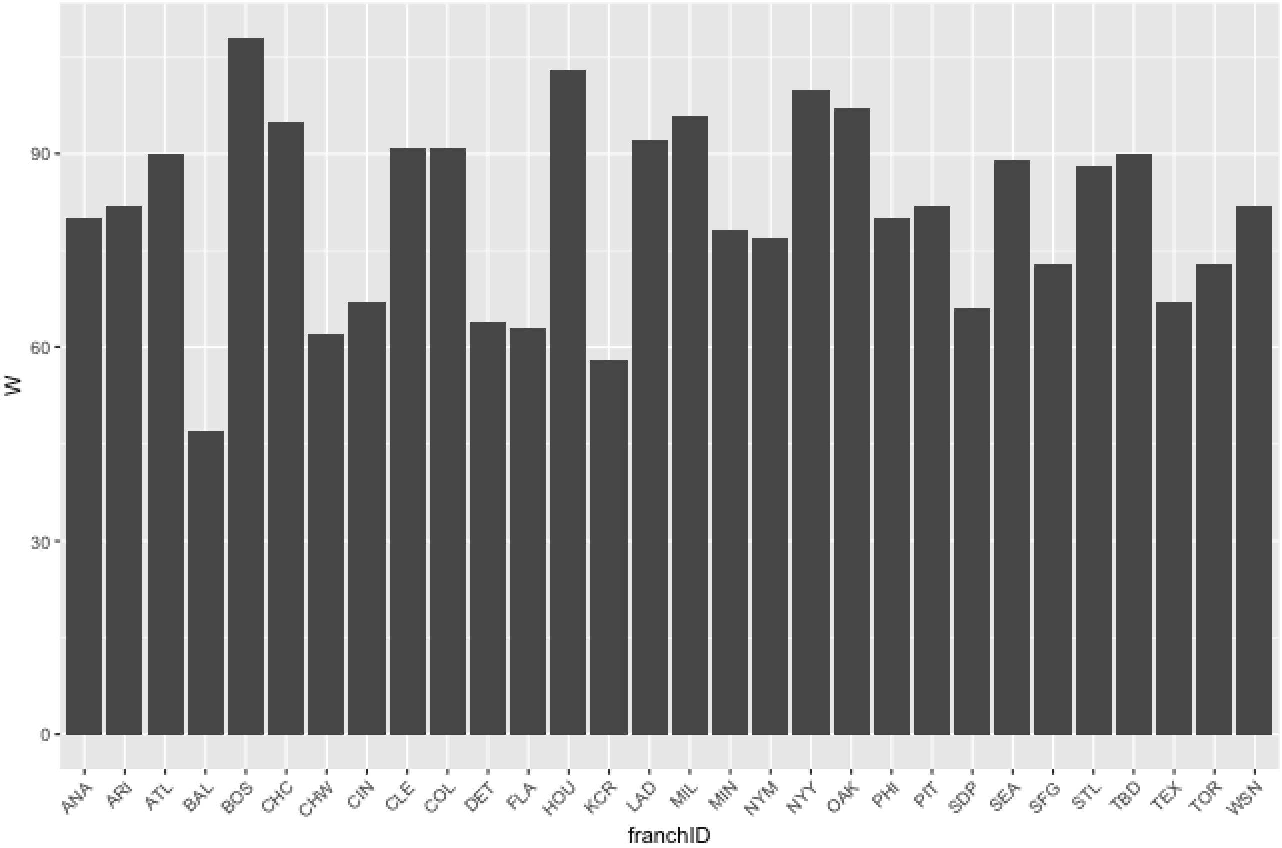
Bar plots

```
ggplot(teams[teams$yearID == 2018, ], mapping = aes(y = W, x = franchID)) +
  geom_bar(stat = "identity")
```



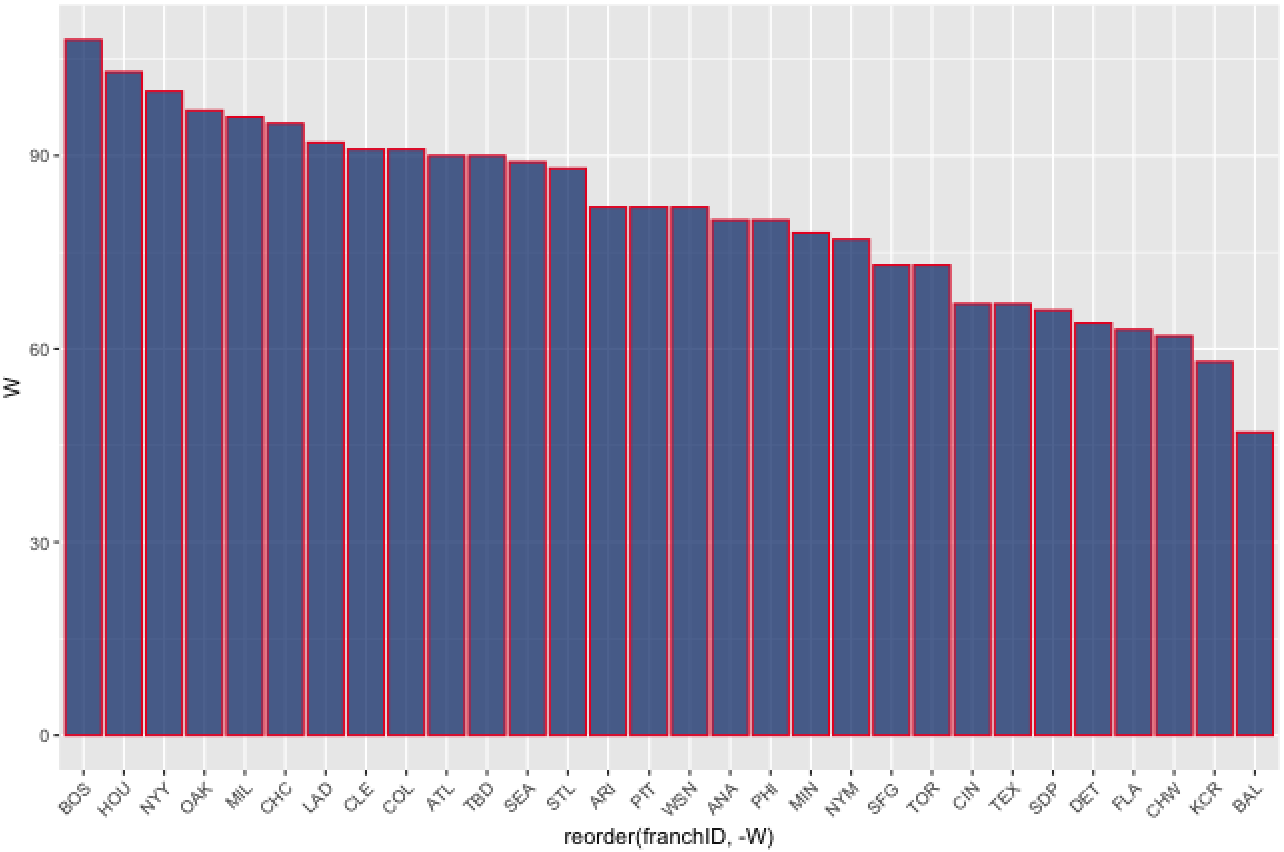
Bar plots: angled text

```
ggplot(teams[teams$yearID == 2018, ], mapping = aes(y = W, x = franchID)) +
  geom_bar(stat = "identity") +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```



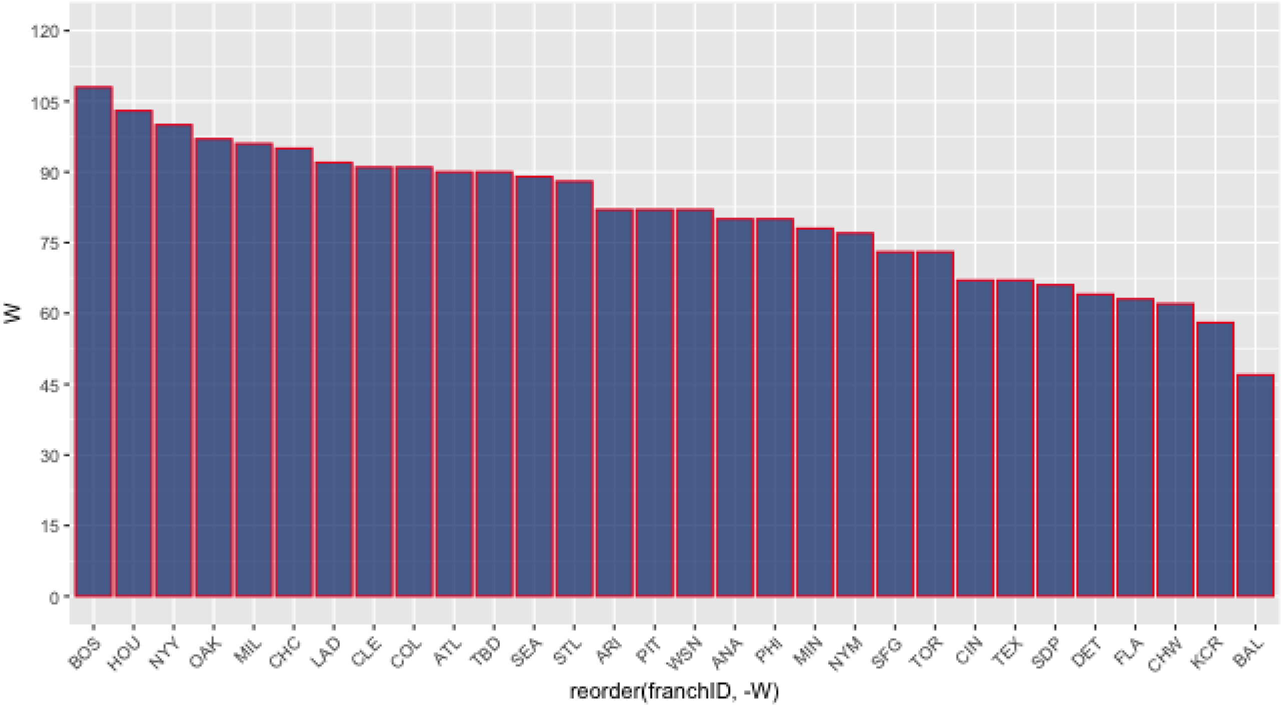
Bar plots: sorted

```
ggplot(teams[teams$yearID == 2018, ], mapping = aes(y = W, x = reorder(franchID, -W))) +
  geom_bar(stat = "identity", color = "#E81828", fill = "#002D72", alpha = .7) +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```



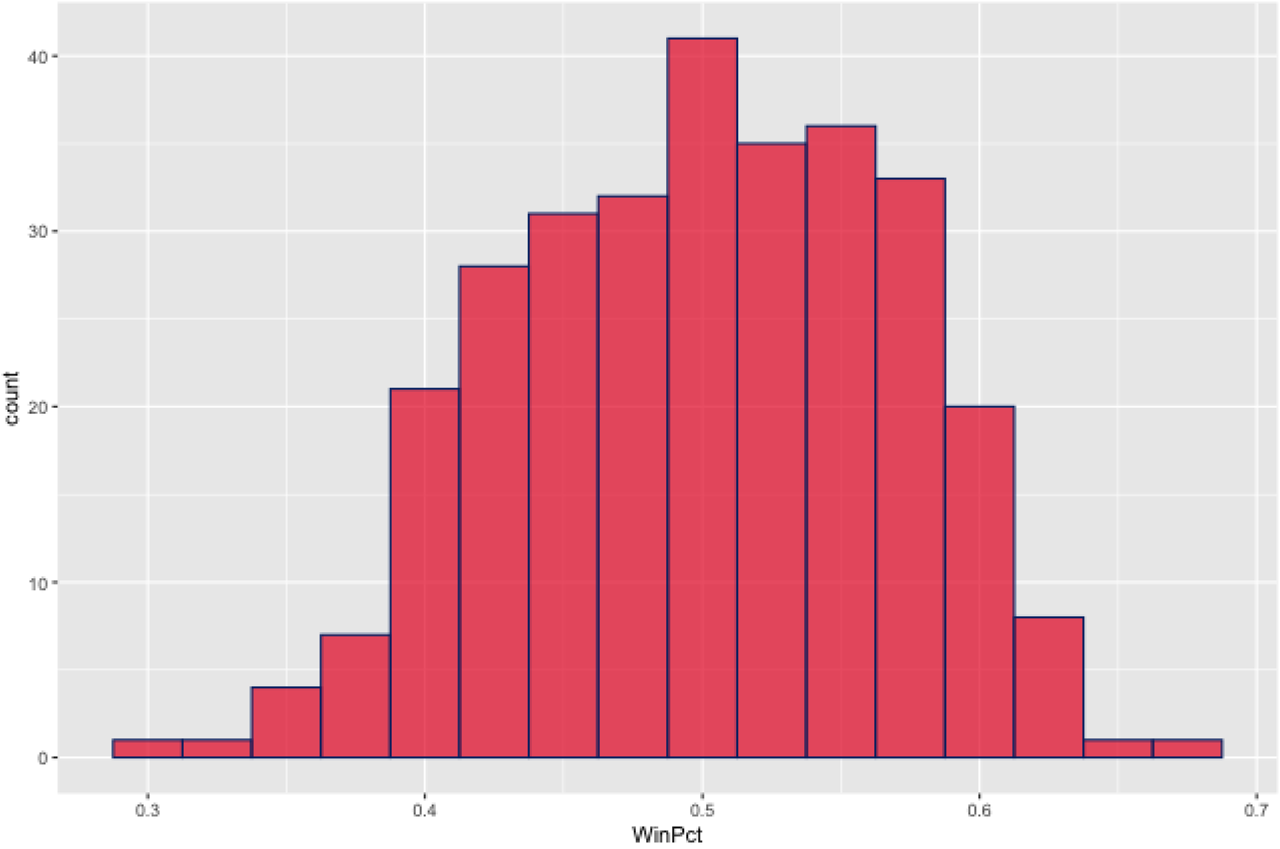
Bar plots: granular scale

```
ggplot(teams[teams$yearID == 2018, ], mapping = aes(y = W, x = reorder(franchID, -W))) +
  geom_bar(stat = "identity", color = "#E81828", fill = "#002D72", alpha = .7) +
  scale_y_continuous(breaks = seq(0, 120, 15), labels = seq(0, 120, 15), limits = c(0, 120)) +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```



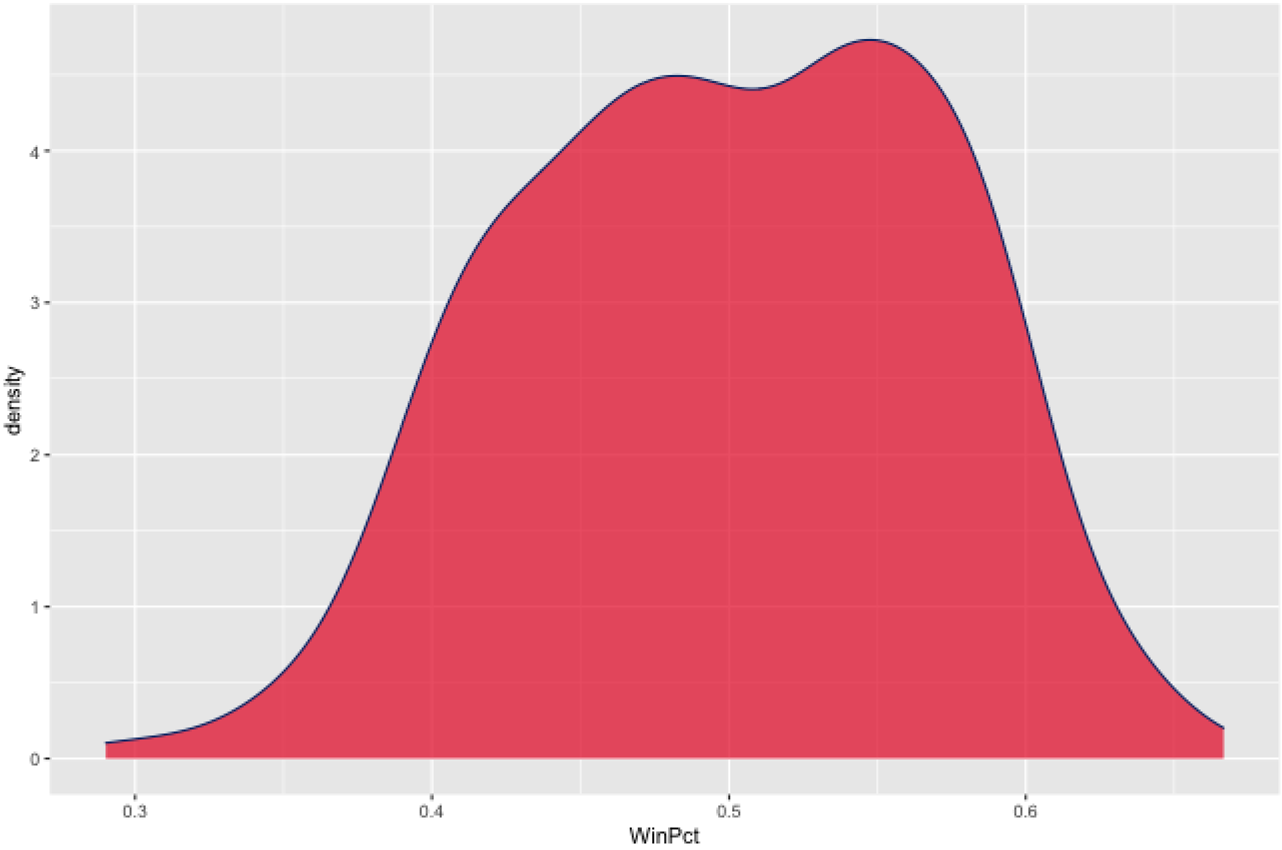
Histograms

```
ggplot(teams, mapping = aes(x = WinPct)) +
  geom_histogram(binwidth = .025, fill = "#E81828", color = "#002D72", alpha = .7)
```



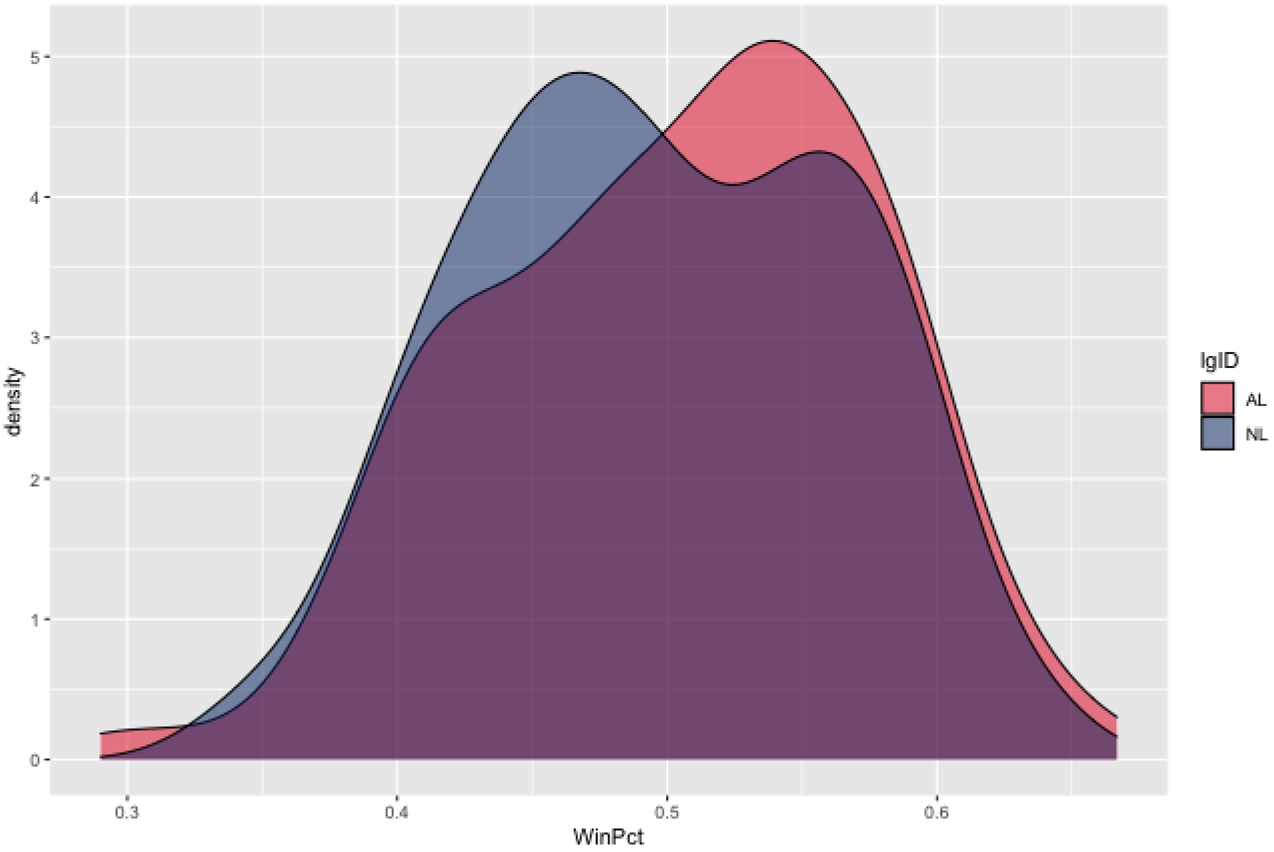
Density plots

```
ggplot(teams, mapping = aes(x = WinPct)) +
  geom_density(fill = "#E81828", color = "#002D72", alpha = .7)
```



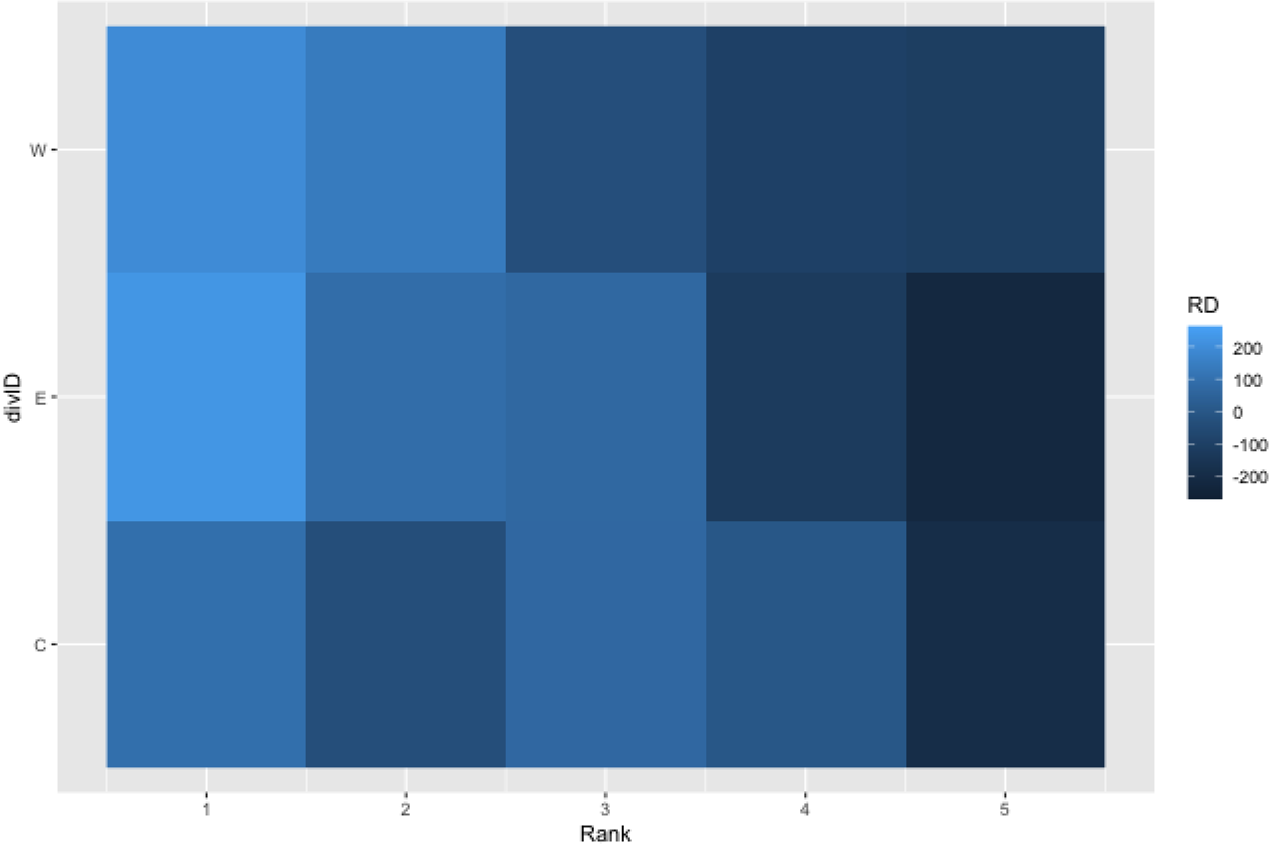
Density plots: custom colors

```
ggplot(teams, mapping = aes(x = WinPct, fill = lgID)) +
  geom_density(alpha = .5) +
  scale_fill_manual(values = c("#E81828", "#002D72"))
```



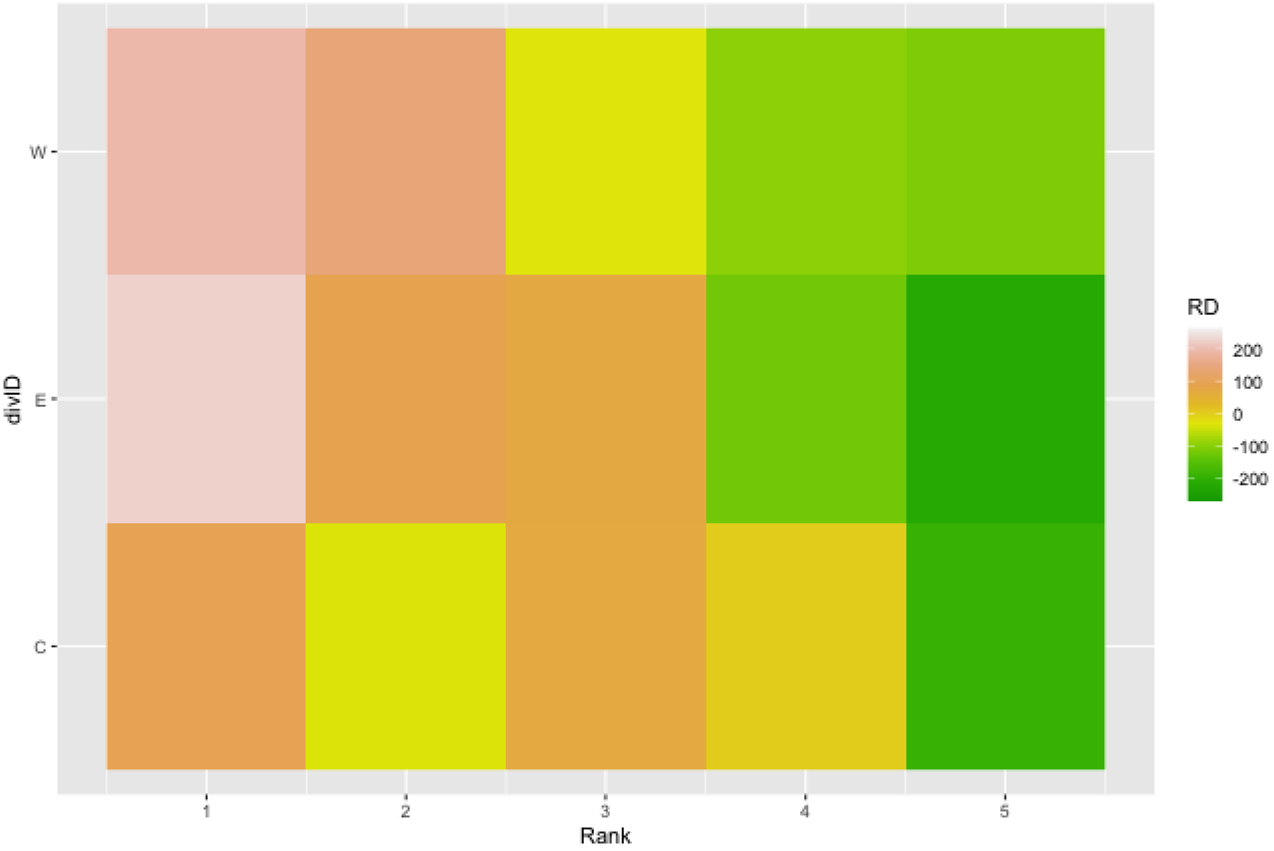
Heat maps

```
ggplot(teams[teams$yearID == 2018, ], mapping = aes(x = Rank, y = divID, fill = RD)) +
  geom_raster()
```



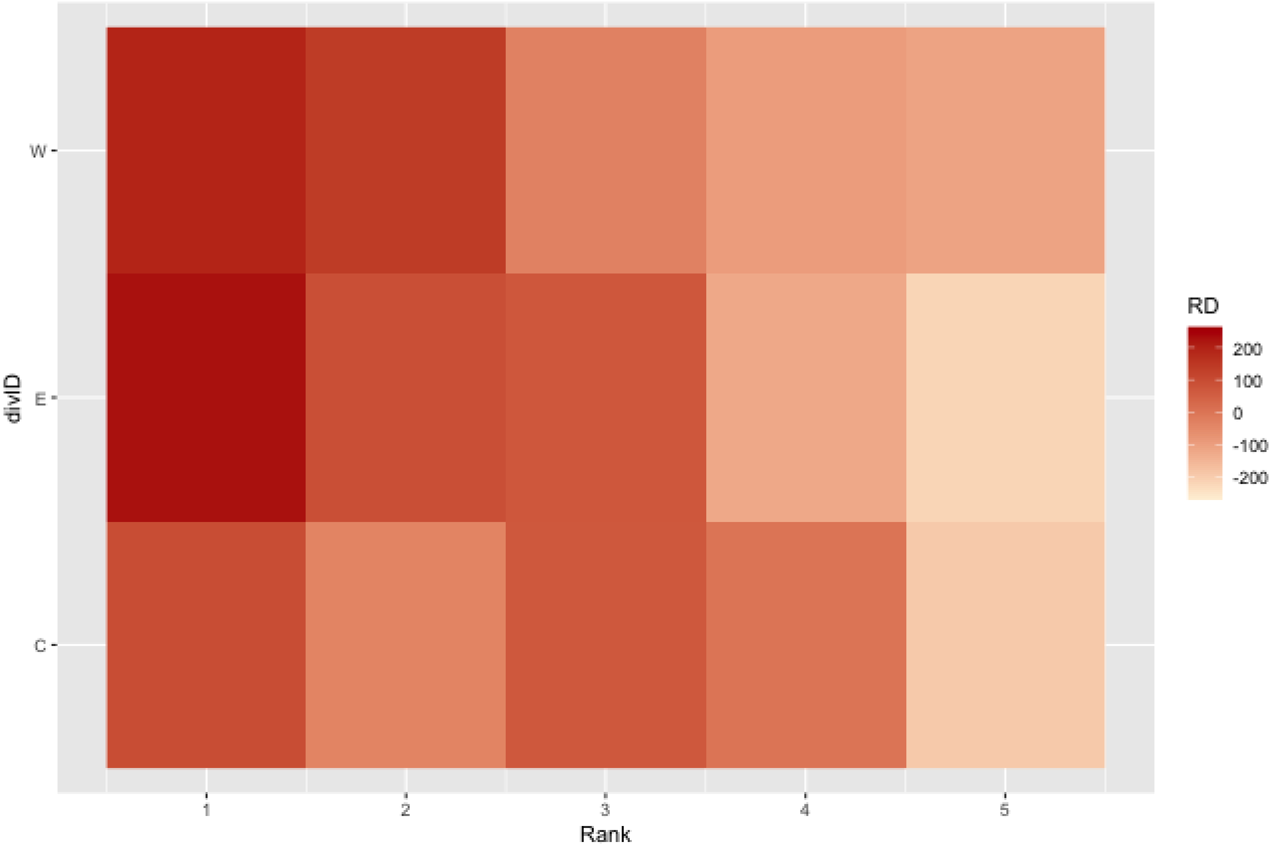
Heat maps: color palette

```
ggplot(teams[teams$yearID == 2018, ], mapping = aes(x = Rank, y = divID, fill = RD)) +
  geom_raster() +
  scale_fill_gradientn(colours = terrain.colors(10))
```



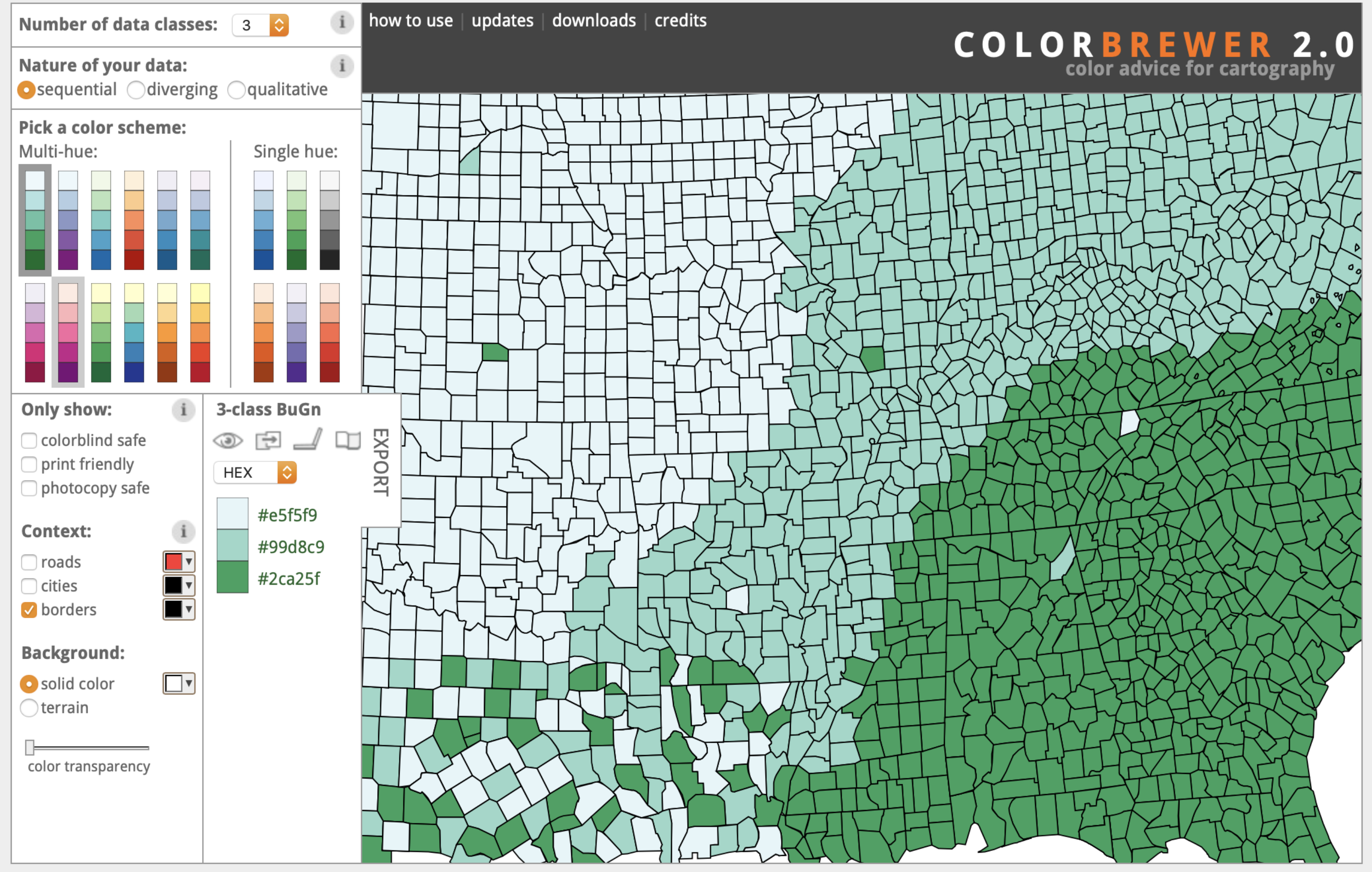
Heat maps: color palette

```
ggplot(teams[teams$yearID == 2018, ], mapping = aes(x = Rank, y = divID, fill = RD)) +
  geom_raster() +
  scale_fill_gradient(low = "#fef0d9", high = "#b30000")
```



Choosing colors

Color Brewer 2



how to use | updates | downloads | credits
COLORBREWER 2.0
color advice for cartography
EXPORT

Number of data classes: 3

Nature of your data:
● sequential ○ diverging ○ qualitative

Pick a color scheme:
Multi-hue:
Single hue:

Only show:
□ colorblind safe
□ print friendly
□ photocopy safe

Context:
□ roads
□ cities
☑ borders

Background:
● solid color
○ terrain

color transparency

3-class BuGn

HEX

#e5f5f9
#99d8c9
#2ca25f

Effective visualization tips

- Provide a title that tells a story.
- Strive to have your visualization function in a closed environment.
- Be mindful of color and scale choices.
- Generally, color is better than shape to make things "pop".
- Not everything has to have a color, shape, transparency, etc.
- Add labels and annotation.
- Use your visualization to support your story.
- Use chunk options `fig.width`, `fig.height`, `fig.align`, and `fig.show` to manipulate your plot's size and placement.

Exercise

Energy data

```r
energy <- read_csv("http://www2.stat.duke.edu/~sms185/data/energy/energy.csv")
```

```r
energy
```

```
#> # A tibble: 105 x 6
#>    MWhperDay name            type  location    note                          boe
#>        <dbl> <chr>           <chr> <chr>       <chr>                       <dbl>
#>  1         3 Chernobyl Solar Solar Ukraine     "On the site of the former…     0
#>  2       637 Solarpark Meuro Solar Germany      <NA>                          55
#>  3       920 Tesla's propos… Solar South Aust… "50,000 homes with solar p…    79
#>  4      1280 Quaid-e-Azam    Solar Pakistan    "Named in honor of Quaid-e…   110
#>  5      1760 Topaz           Solar USA          <NA>                         152
#>  6      2025 Agua Caliente   Solar USA         "Arizona"                     175
#>  7      2466 Kamuthi         Solar India       "\"150,000\" homes"           213
#>  8      2720 Longyangxia     Solar China        <NA>                         234
#>  9      3840 Kurnool         Solar India        <NA>                         331
#> 10      4950 Tengger Desert  Solar China       "Covers 3.2% of the land a…   427
#> # … with 95 more rows
```

Data dictionary

The power sources represent the amount of energy a power source generates each day as represented in daily MWh.

- `MWhperDay`: MWh of energy generated per day
- `name`: energy source name
- `type`: type of energy source
- `location`: country of energy source
- `note`: more details on energy source
- `boe`: barrel of oil equivalent

- **Daily megawatt hour (MWh)** is a measure of energy output.
- **1 MWh** is, on average, enough power for 28 people in the USA

Objective

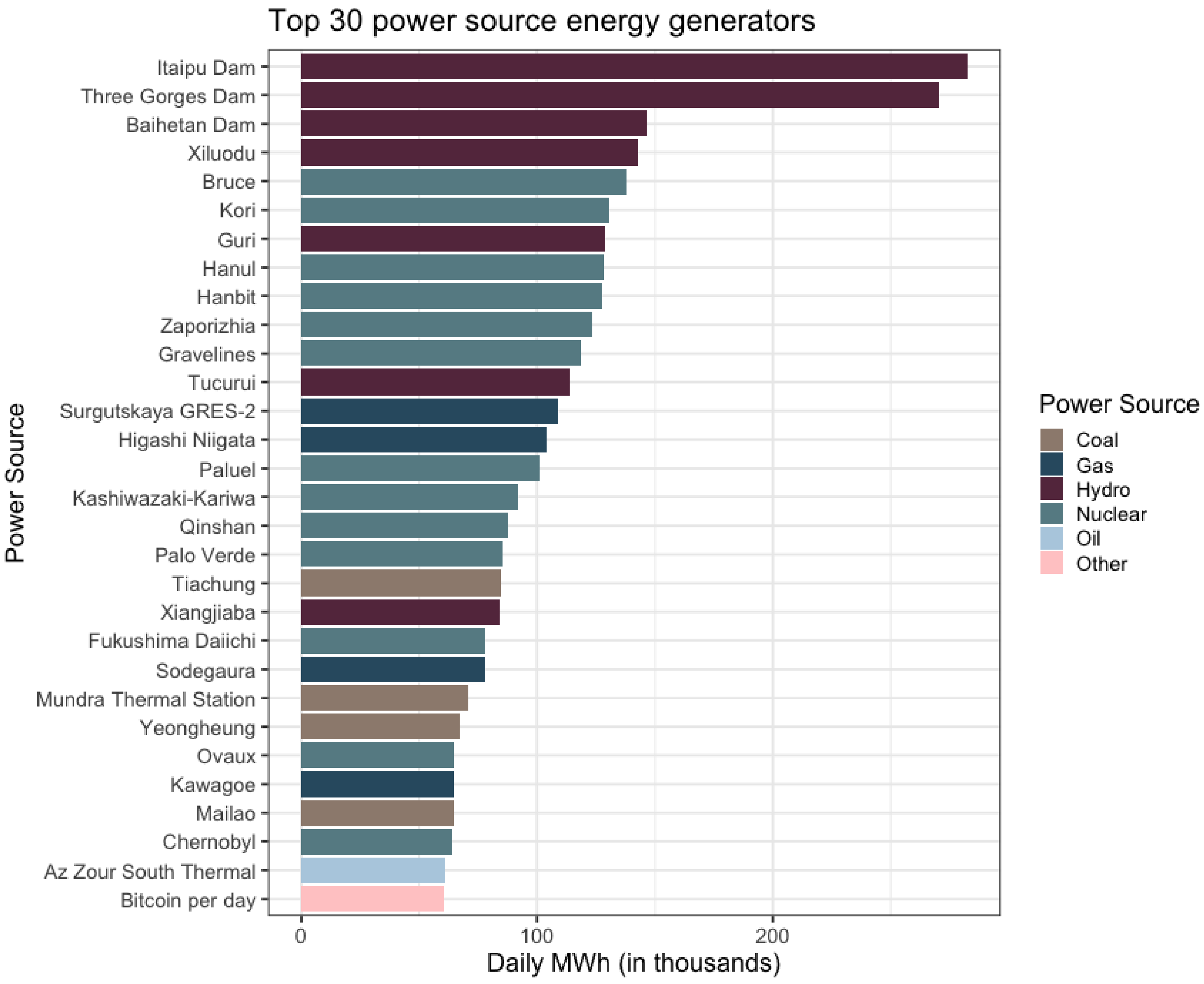
Re-create the plot on the following slide.

A few notes:

- base font size is 18
- hex colors: `c("#9d8b7e", "#315a70", "#66344c", "#678b93", "#b5cfe1", "#ffcccc")`
- use function `order()` to help get the top 30

Starter code:

```
energy_top_30 <- energy[order(energy$MWhperDay, decreasing = T)[1:30], ]
```



1 MWh is, on average, enough power for 28 people in the USA

References

1. Grolemund, G., & Wickham, H. (2019). R for Data Science. R4ds.had.co.nz. https://r4ds.had.co.nz/data-visualisation.html

2. https://ggplot2.tidyverse.org/reference/

3. Lahman, S. (2019) Lahman's Baseball Database, 1871-2018, Main page, http://www.seanlahman.com/baseball-archive/statistics/

4. https://www.visualcapitalist.com/worlds-largest-energy-sources/