Data Visualization

The grammar of graphics and ggplot2

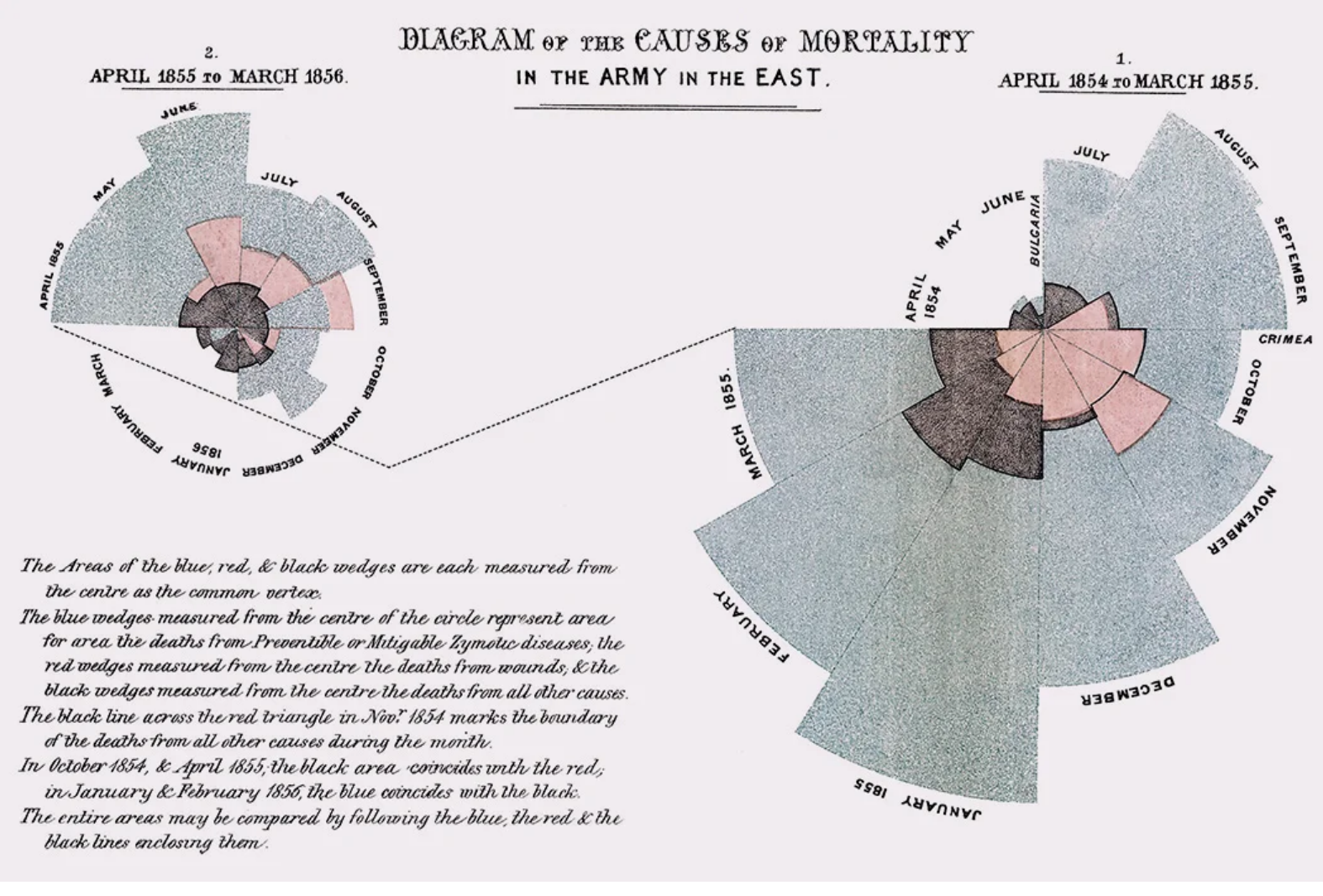
June 8th, 2022

Do these datasets have anything in common?

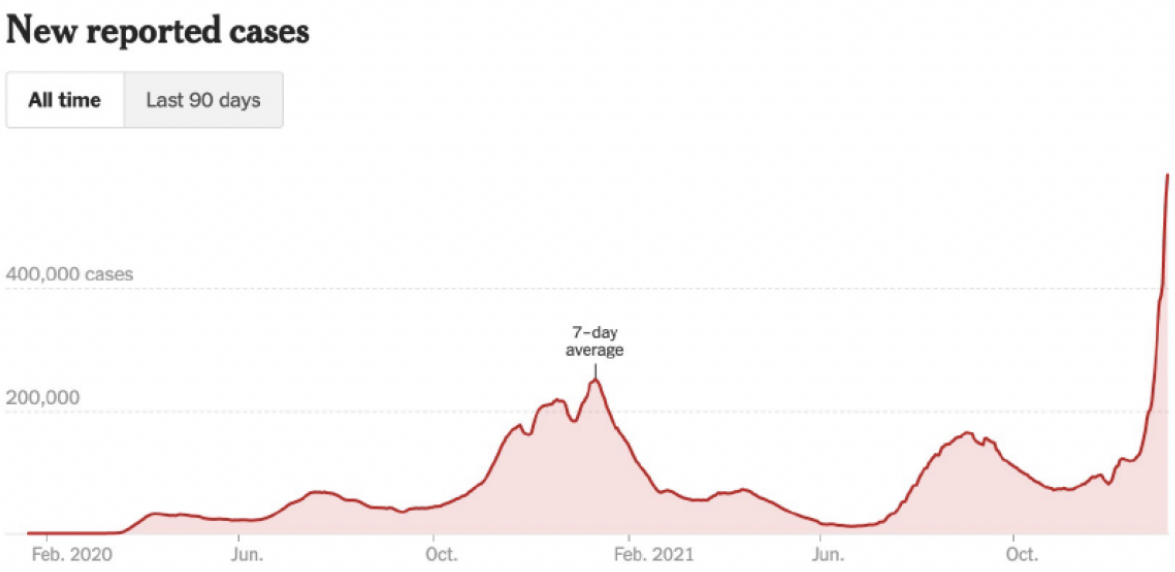
"Graphics reveal data" - Edward Tufte

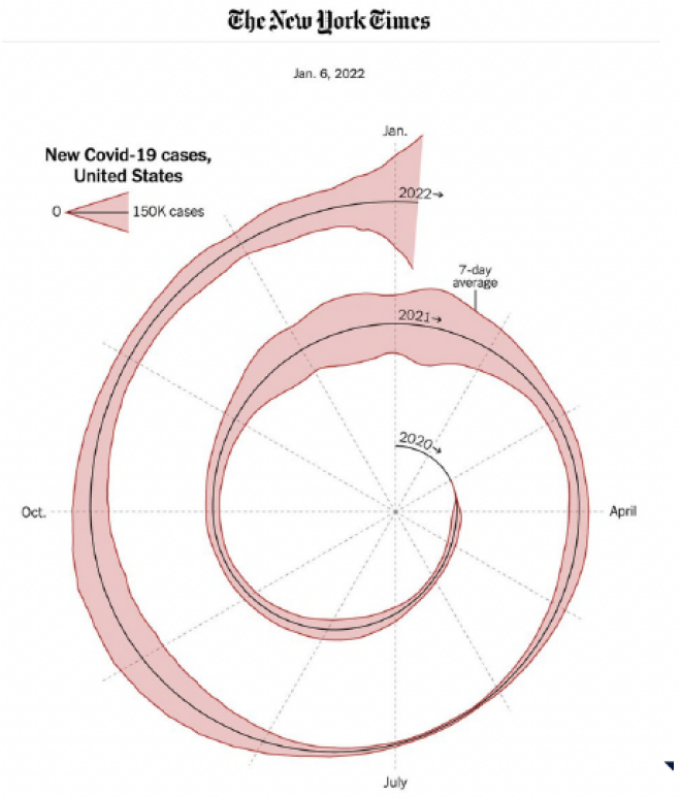
Always visualize your data before analyzing / modeling it

Florence Nightingale's rose diagram



New COVID-19 Cases (NYT, Jan 2022)





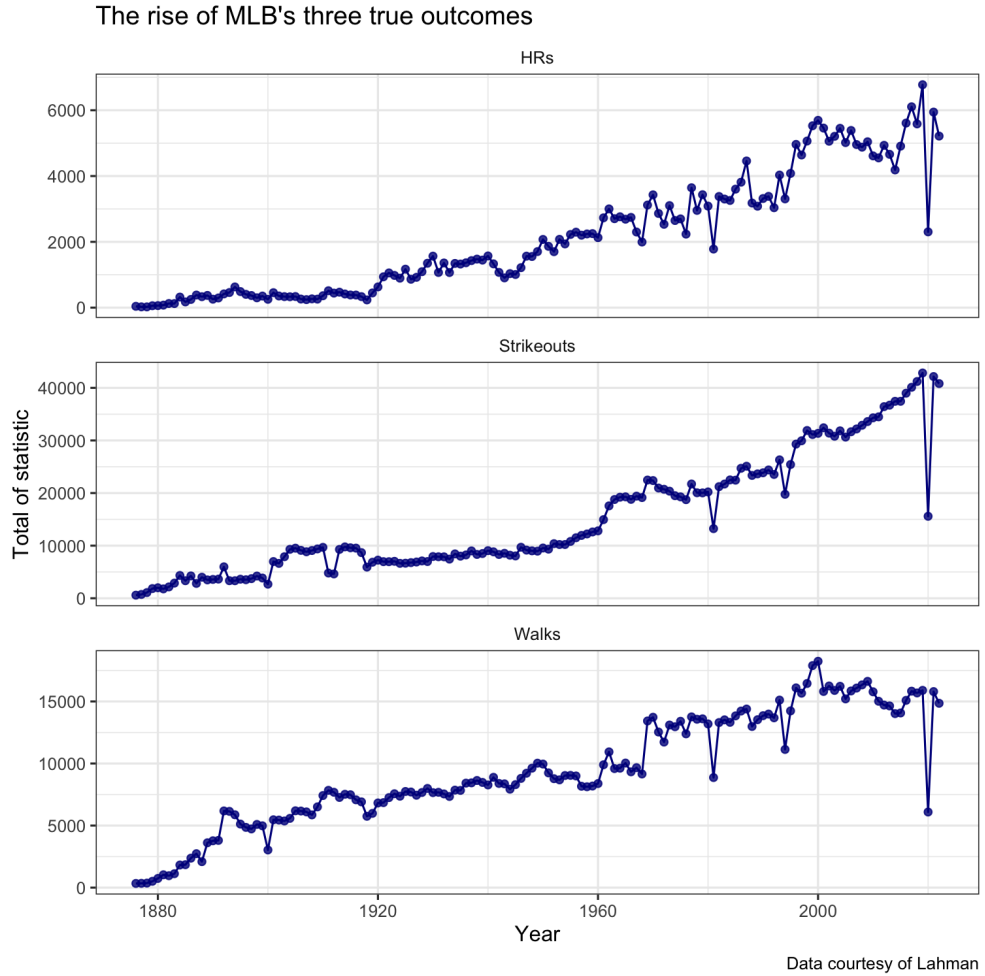
Quick review of yesterday's material

We are exploring MLB hitting data from `Lahman` using the `tidyverse`

```
library(tidyverse)
library(Lahman)
Batting <- as_tibble(Batting)
year_batting_summary <- Batting %>%
  filter(lgID %in% c("AL", "NL")) %>%
  group_by(yearID) %>%
  summarize_at(vars(H, HR, SO, BB, AB),
               sum, na.rm = TRUE) %>%
  mutate(batting_avg = H / AB)
```

But how do we make data visualizations in R?

What steps do we take to make this figure?



The Grammar of Graphics

Principled data visualization framework introduced by Leland Wilkinson

- **start with the raw data** -> visualize transformations, summaries, etc. of the data

Hadley Wickham expanded upon this foundation with a layered `R` implementation via `ggplot2`

1. **`data`** - one or more datasets (in tidy tabular format)
2. **`geom`** - one or more geometric objects to visually represent the data (e.g. points, lines, bars, etc.)
3. **`aes`** - mappings of columns / variables to visual properties (i.e. *aesthetics*) of the geometric objects
4. **`scale`** - one scale for each variable displayed (e.g. axis limits, log scale, colors, etc.)
5. **`facet`** - similar subplots (i.e. *facets*) for subsets of the same data using a categorical variable
6. **`stat`** - statistical transformations and summaries (e.g. identity, count, smooth, quantile, etc.)
7. **`coord`** - one or more coordinate systems (e.g. **cartesian**, polar, map projection)
8. **`labs`** - labels/guides for each variable and other parts of the plot (e.g. title, subtitle, caption, etc.)
9. **`theme`** - customization of plot layout (e.g. text size, alignment, legend position, etc.)

Start with the data...

```
ggplot(data = year_batting_summary)
```

or equivalently using the %>%

```
year_batting_summary %>%
  ggplot()
```

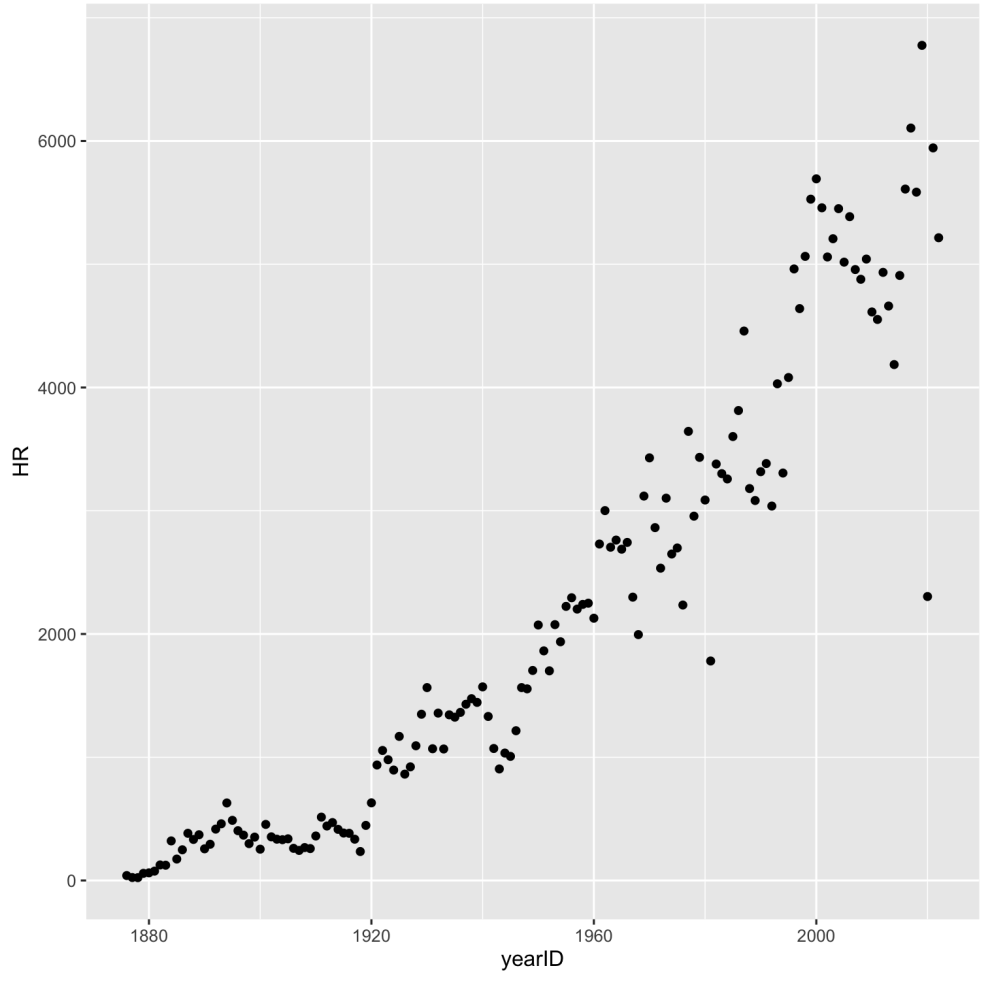
but nothing is displayed!

What variables and geometric object?

```
year_batting_summary %>%
  ggplot() +
  geom_point(aes(x = yearID, y = HR))
```

- adding (+) a geometric layer of points to the plot
- map `yearID` to the x-axis and `HR` to the y-axis via `aes()`
- implicitly using `coord_cartesian()`

```
year_batting_summary %>%
  ggplot() +
  geom_point(aes(x = yearID, y = HR)) +
  coord_cartesian()
```

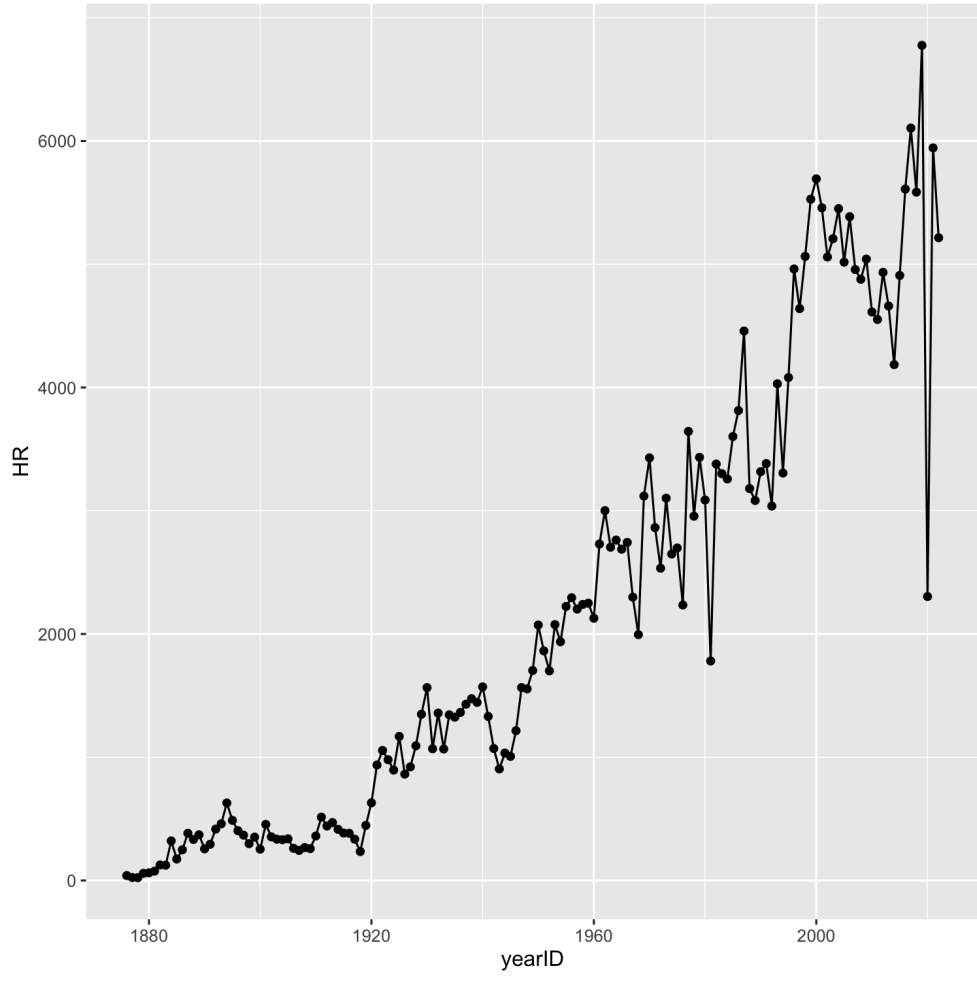


Can we add another geometric layer?

```
year_batting_summary %>%
  ggplot() +
  geom_point(aes(x = yearID, y = HR)) +
  geom_line(aes(x = yearID, y = HR))
```

- adding (+) a line geometric layer
- Include mappings shared across geometric layers inside `ggplot()`

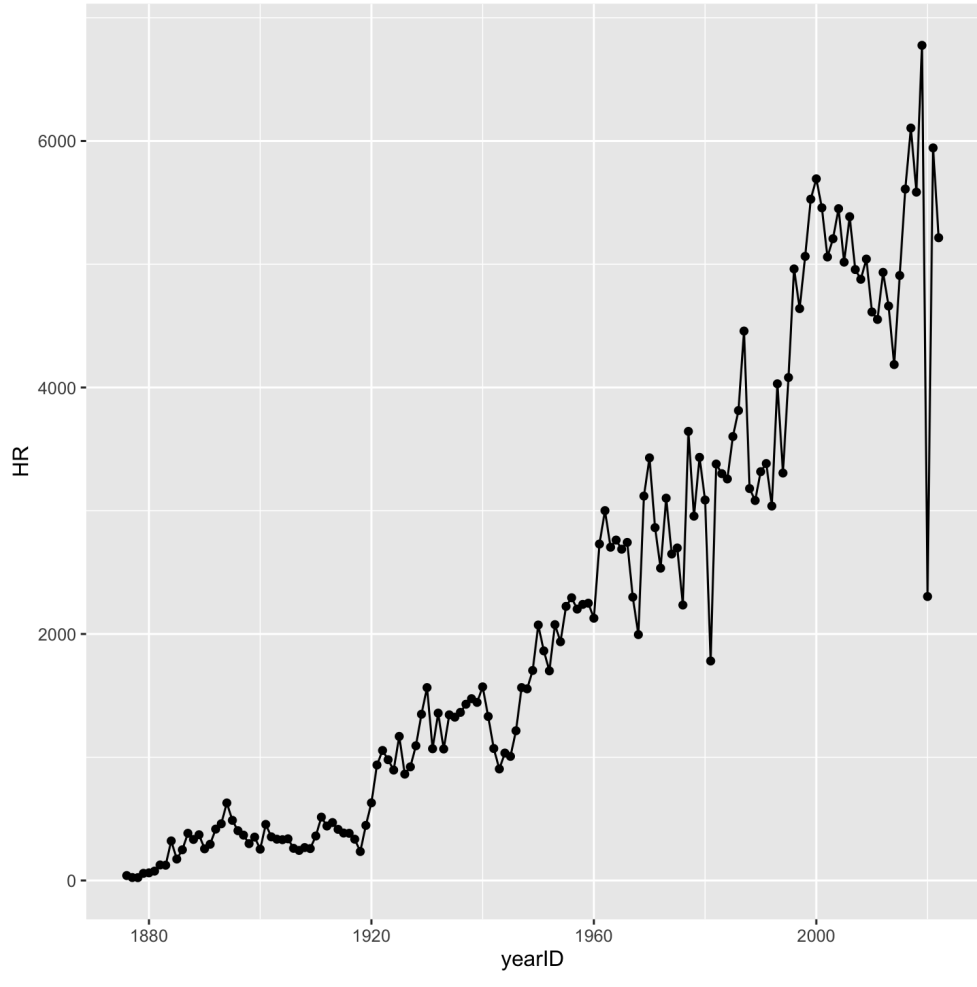
```
year_batting_summary %>%
  ggplot(aes(x = yearID, y = HR)) +
  geom_point() +
  geom_line()
```



What about the scales?

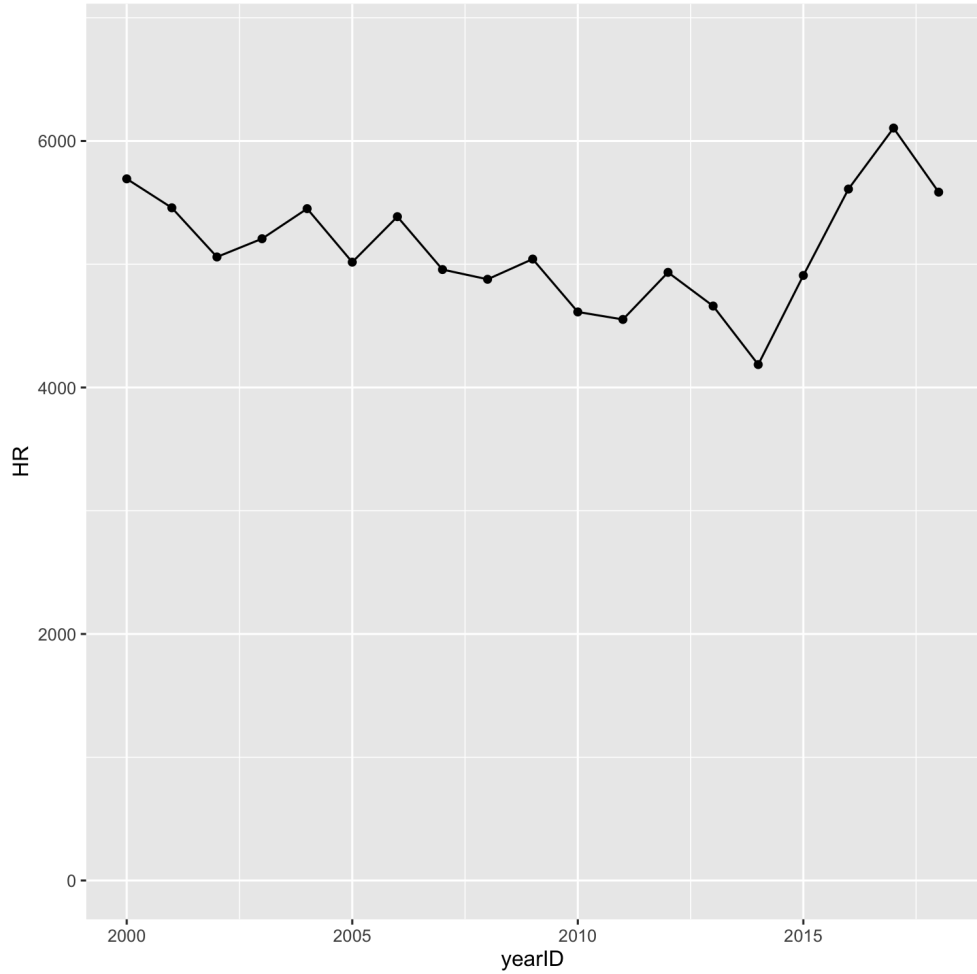
```
year_batting_summary %>%
  ggplot(aes(x = yearID, y = HR)) +
  geom_point() +
  geom_line() +
  scale_x_continuous() +
  scale_y_continuous()
```

- `yearID` and `HR` are continuous variables, resulting in continuous scales by default



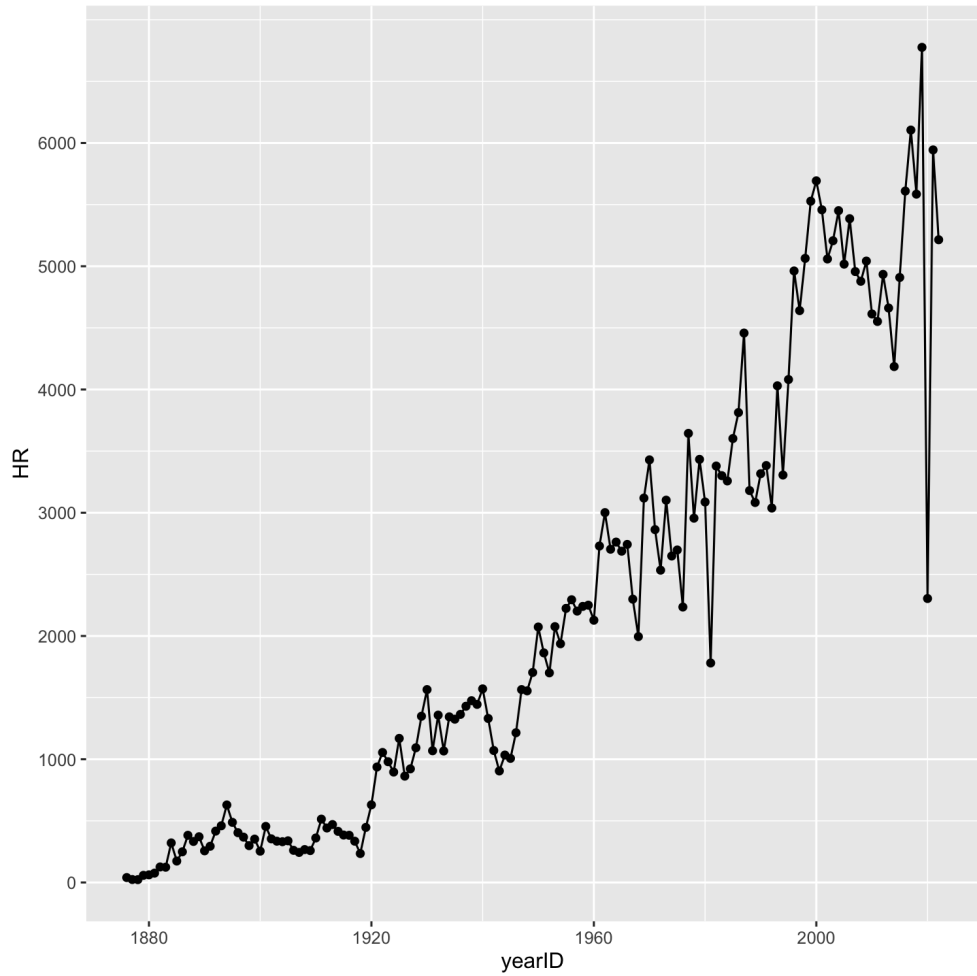
We can customize the scale limits

```
year_batting_summary %>%
  ggplot(aes(x = yearID, y = HR)) +
  geom_point() +
  geom_line() +
  scale_x_continuous(limits = c(2000, 2018))
```



We can customize the label breaks

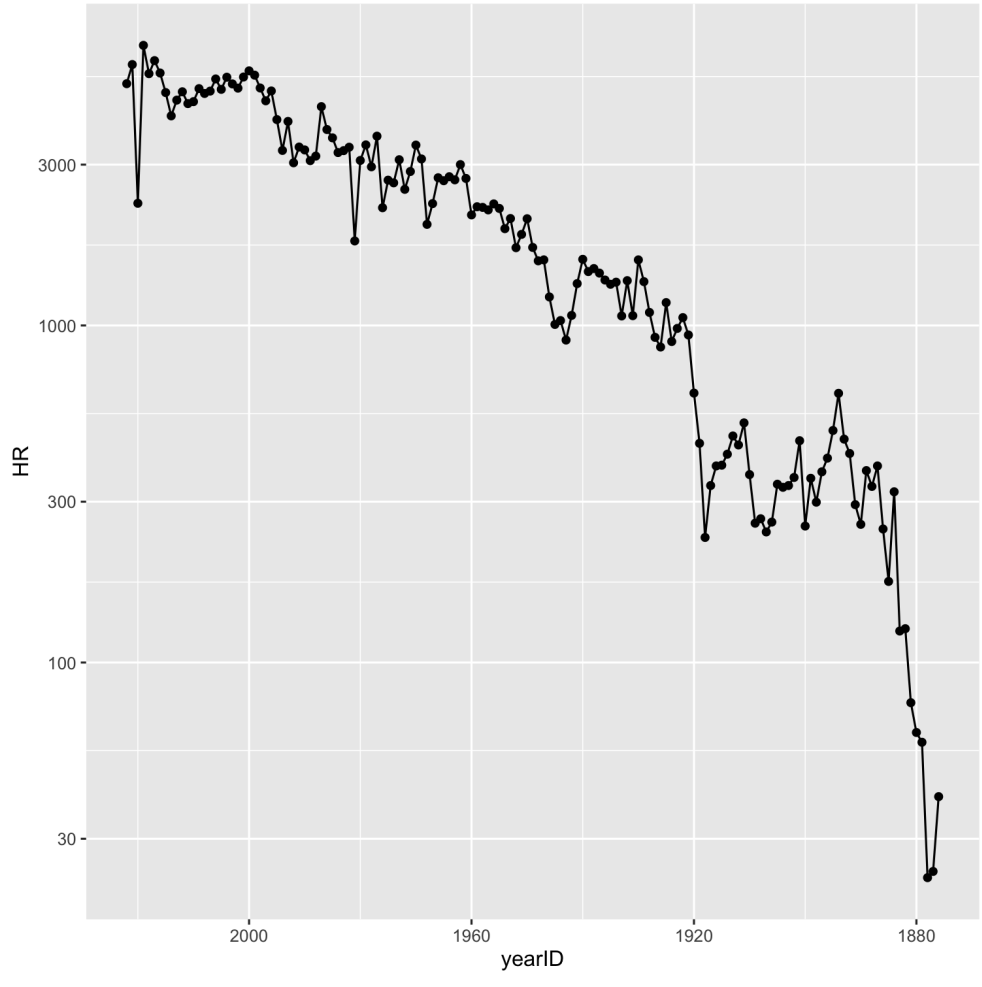
```
year_batting_summary %>%
  ggplot(aes(x = yearID, y = HR)) +
  geom_point() +
  geom_line() +
  scale_y_continuous(breaks = seq(0, 6000,
                                  by = 1000))
```



We can use different scales!

```
year_batting_summary %>%
  ggplot(aes(x = yearID, y = HR)) +
  geom_point() +
  geom_line() +
  scale_x_reverse() +
  scale_y_log10()
```

You can easily adjust variable scales without modifying the columns directly

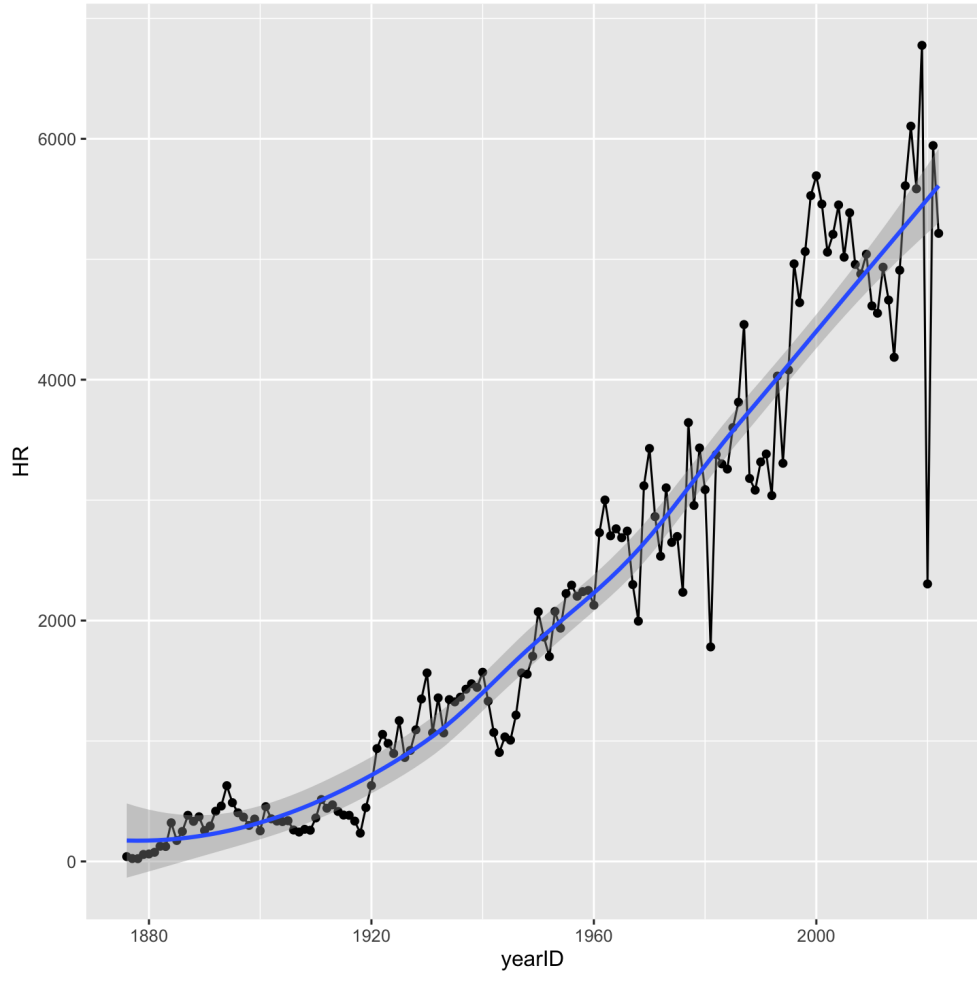


Add a statistical summary?

```
year_batting_summary %>%
  ggplot(aes(x = yearID, y = HR)) +
  geom_point() +
  geom_line() +
  stat_smooth()
```

- Smoothing regression summary (will cover later) using `yearID` and `HR`
- Geometric layers implicitly use a default statistical summary
- Technically we are already using `geom_point(stat = "identity")`

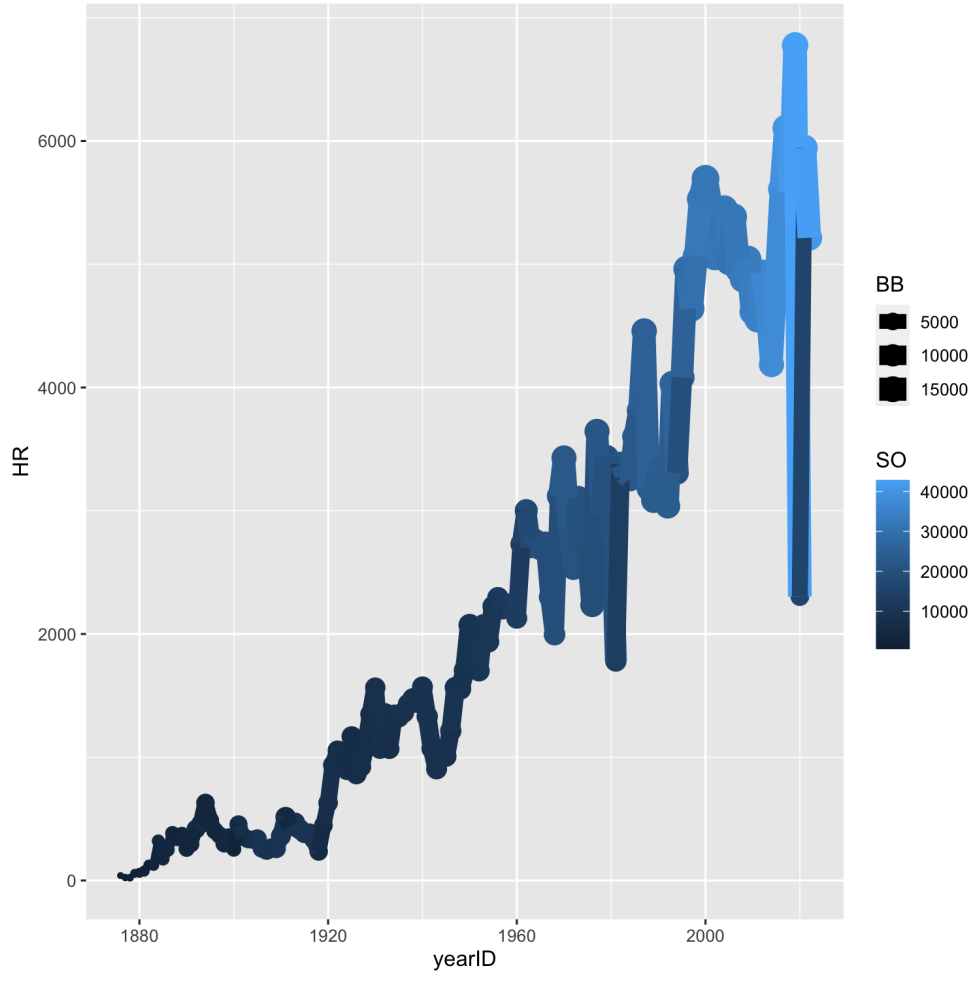
```
year_batting_summary %>%
  ggplot(aes(x = yearID, y = HR)) +
  geom_point() +
  geom_line() +
  geom_smooth()
```



Map additional variables?

```
year_batting_summary %>%
  ggplot(aes(x = yearID, y = HR,
             color = SO,
             size = BB)) +
  geom_point() +
  geom_line()
```

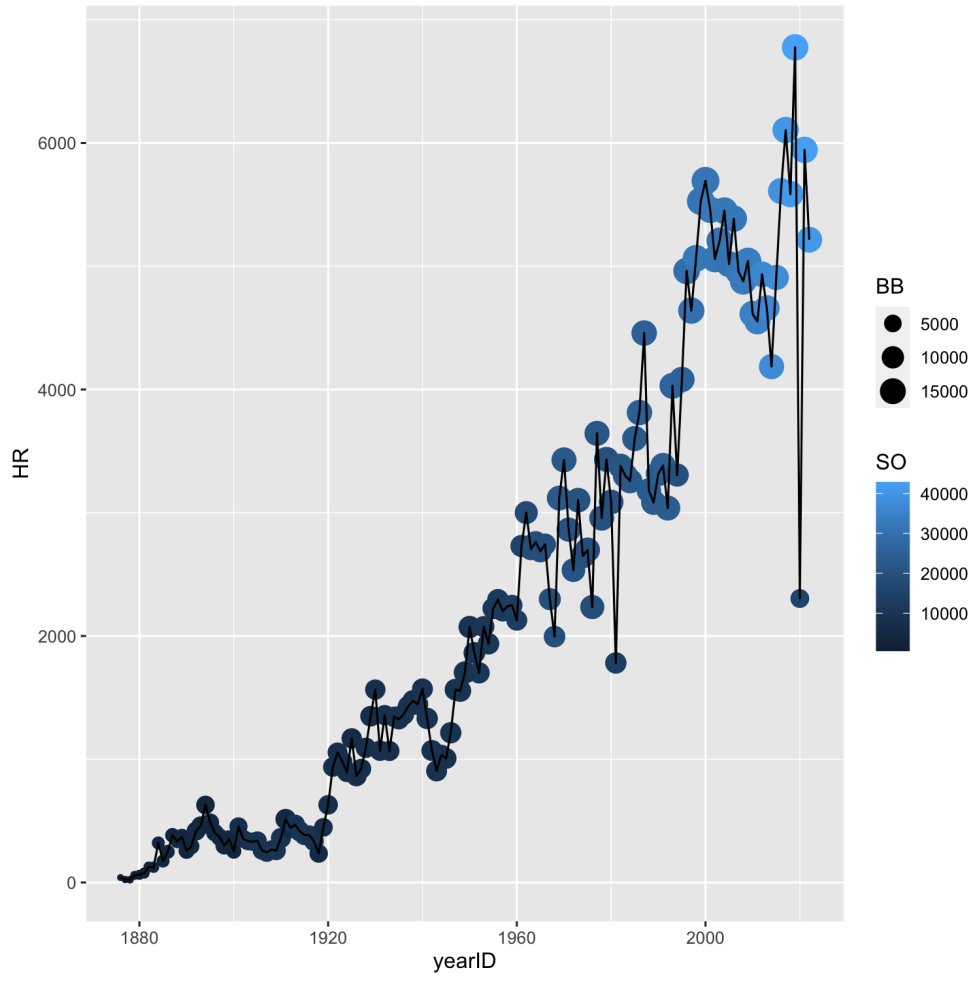
- HR, SO, and BB are all displayed!
- color and size are being shared across layers
- This is a bit odd to look at...



Customize mappings by layer

```
year_batting_summary %>%
  ggplot(aes(x = yearID, y = HR)) +
  geom_point(aes(color = SO,
                 size = BB)) +
  geom_line()
```

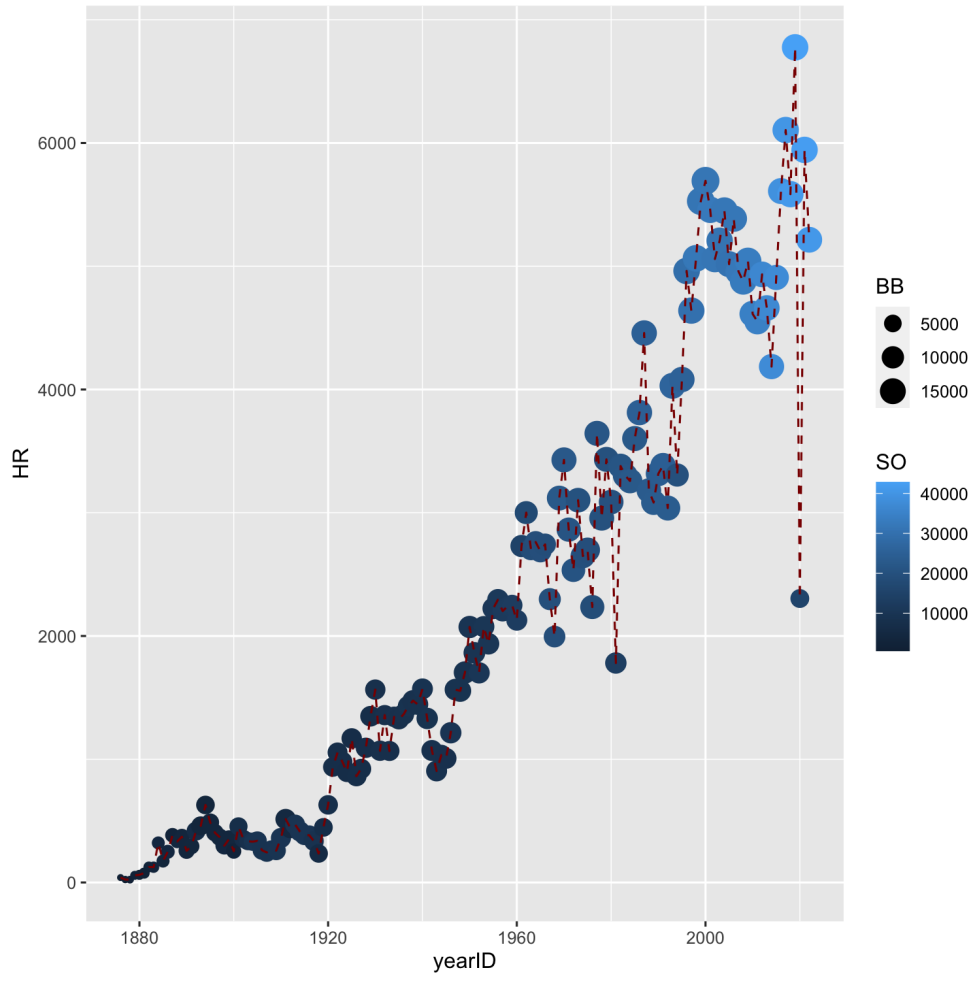
- Now mapping `SO` and `BB` to `color` and `size` of **only** the point layer



Can change aesthetics without mapping variables

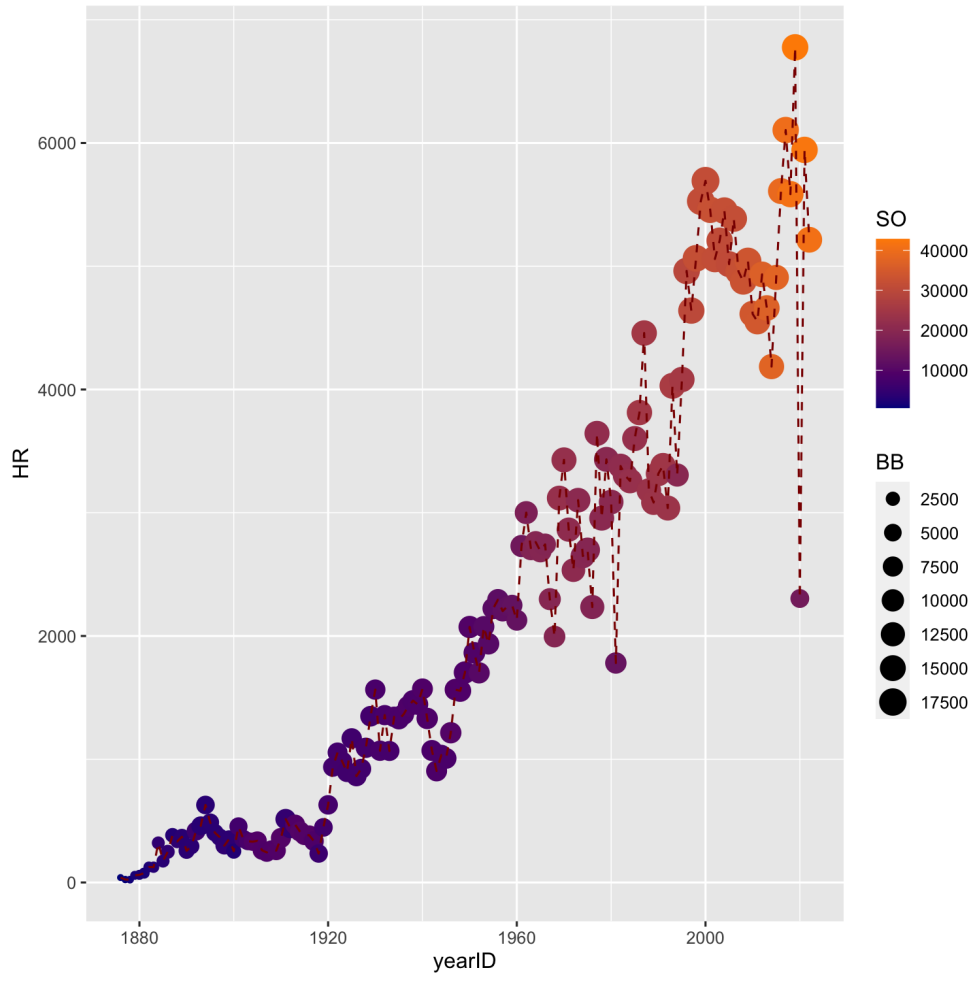
```
year_batting_summary %>%
  ggplot(aes(x = yearID, y = HR)) +
  geom_point(aes(color = SO,
                 size = BB)) +
  geom_line(color = "darkred",
            linetype = "dashed")
```

- Set manual values to the `color` and `linetype` of the line layer



Remember: one scale for each mapped variable

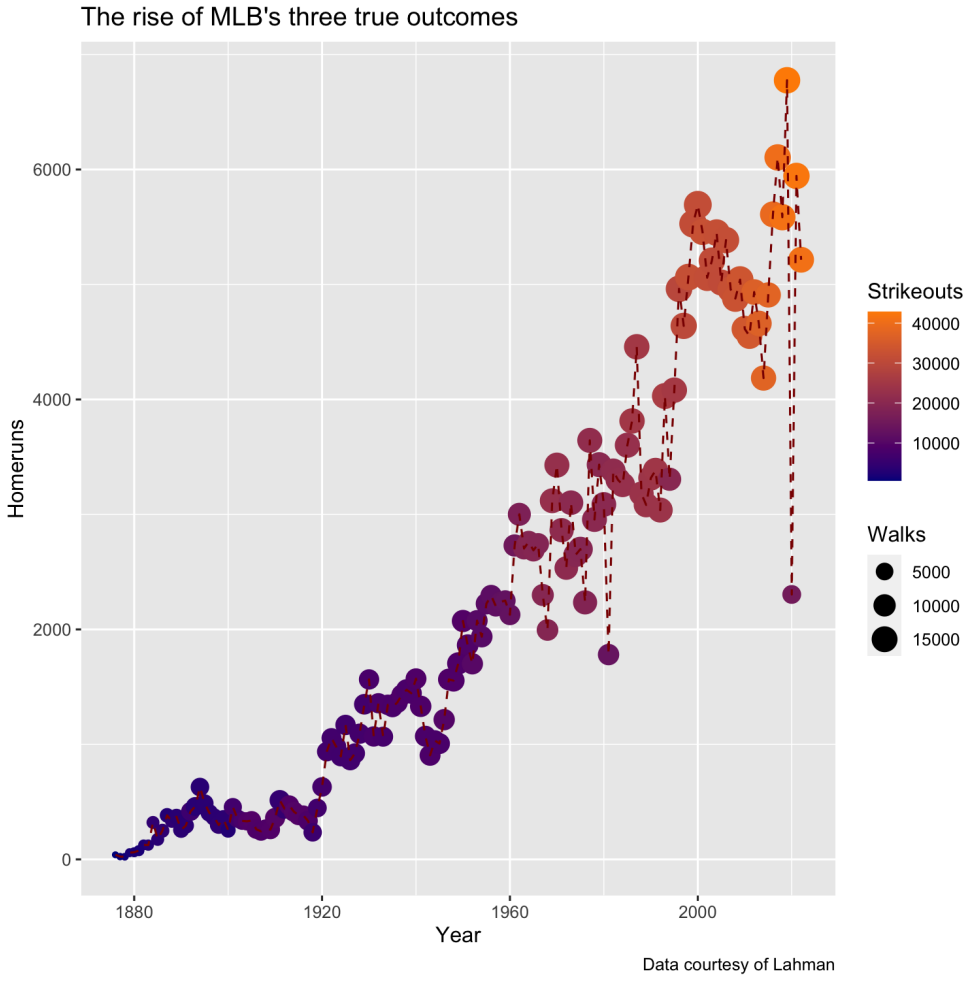
```
year_batting_summary %>%
  ggplot(aes(x = yearID, y = HR)) +
  geom_point(aes(color = SO,
                 size = BB)) +
  geom_line(color = "darkred",
            linetype = "dashed") +
  scale_color_gradient(low = "darkblue",
                       high = "darkorange") +
  scale_size_continuous(breaks =
                          seq(0, 20000,
                              2500))
```



You MUST label your plots!

```
year_batting_summary %>%
  ggplot(aes(x = yearID, y = HR)) +
  geom_point(aes(color = SO,
                 size = BB)) +
  geom_line(color = "darkred",
            linetype = "dashed") +
  scale_color_gradient(low = "darkblue",
                       high = "darkorange") +
  labs(x = "Year", y = "Homeruns",
       color = "Strikeouts",
       size = "Walks",
       title = "The rise of MLB's three true
       caption = "Data courtesy of Lahman")
```

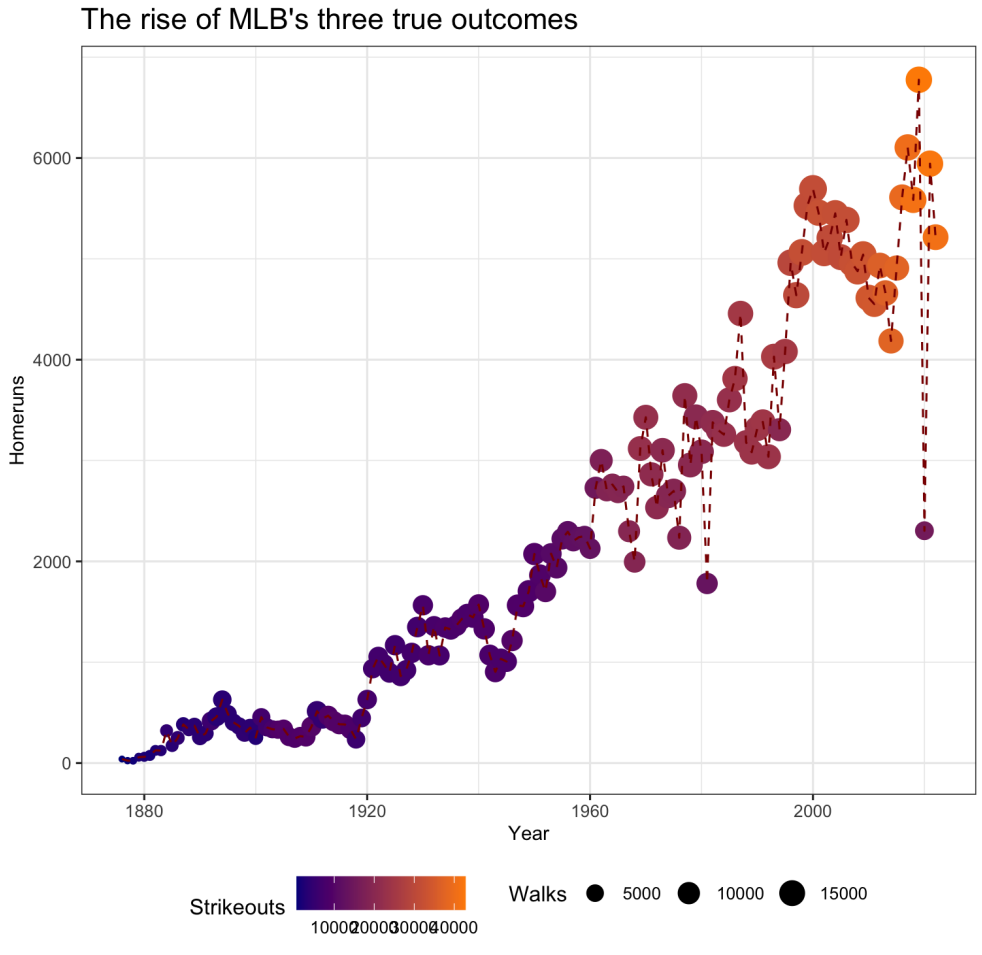
- Each mapped aesthetic can be labelled (*what happened to the legend order?*)



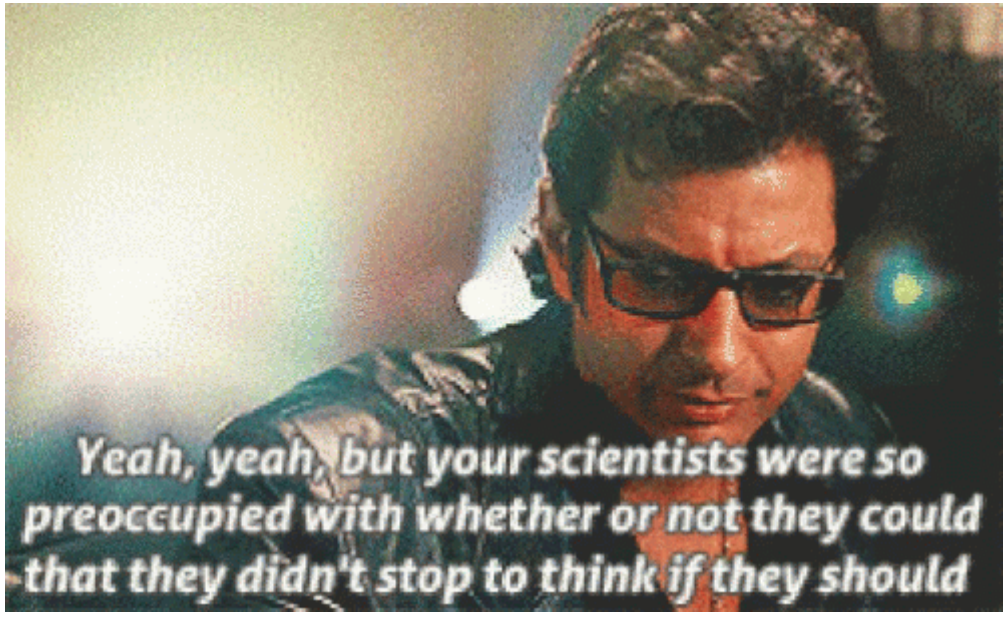
Custom theme

```
year_batting_summary %>%
  ggplot(aes(x = yearID, y = HR)) +
  geom_point(aes(color = SO,
                 size = BB)) +
  geom_line(color = "darkred",
            linetype = "dashed") +
  scale_color_gradient(low = "darkblue",
                       high = "darkorange") +
  labs(x = "Year", y = "Homeruns",
       color = "Strikeouts",
       size = "Walks",
       title = "The rise of MLB's three true
       caption = "Data courtesy of Lahman") +
  theme_bw() +
  theme(legend.position = "bottom",
        plot.title = element_text(size = 15),
        axis.title = element_text(size = 10))
```

- `theme_bw()` is a popular default, but check out `ggthemes` for more, ex: `theme_fivethirtyeight()`



A lesson about data visualization...



Simpler is better - instead create three separate plots for `HR`, `SO`, and `BB` with each mapped to `y`

How do we do this without repeating the same code three times?

Pivot the data!

Remember: we need the data in tidy format

Within the `tidyverse`, the `tidyr` package has functions that allow us to easily *realign* our data

- `pivot_longer` - gathers information spread out across variables, increase `nrow()` but decrease `ncol()`
- `pivot_wider` - spreads information out from observations, decrease `nrow()` but increase `ncol()`

```
year_batting_summary %>%
  select(yearID, HR, SO, BB) %>%
  rename(HRs = HR, Strikeouts = SO,
         Walks = BB) %>%
  pivot_longer(HRs:Walks,
               names_to = "stat",
               values_to = "value")
```

```
## # A tibble: 6 × 3
##   yearID stat       value
##    <int> <chr>      <int>
## 1   1876 HRs           40
## 2   1876 Strikeouts   589
## 3   1876 Walks        336
## 4   1877 HRs           24
## 5   1877 Strikeouts   726
## 6   1877 Walks        345
```

We have now created a new variable `stat`

Use facets to create subplots with categorical variables

```
year_batting_summary %>%
  select(yearID, HR, SO, BB) %>%
  rename(HRs = HR, Strikeouts = SO,
         Walks = BB) %>%
  pivot_longer(HRs:Walks,
               names_to = "stat",
               values_to = "value") %>%
  ggplot(aes(x = yearID, y = value)) +
  geom_line(color = "darkblue") +
  geom_point(color = "darkblue") +
  facet_wrap(~ stat,
             scales = "free_y", ncol = 1) +
  labs(x = "Year", y = "Total of statistic",
       title = "The rise of MLB's three true
       caption = "Data courtesy of Lahman") +
  theme_bw() +
  theme(strip.background = element_blank())
```