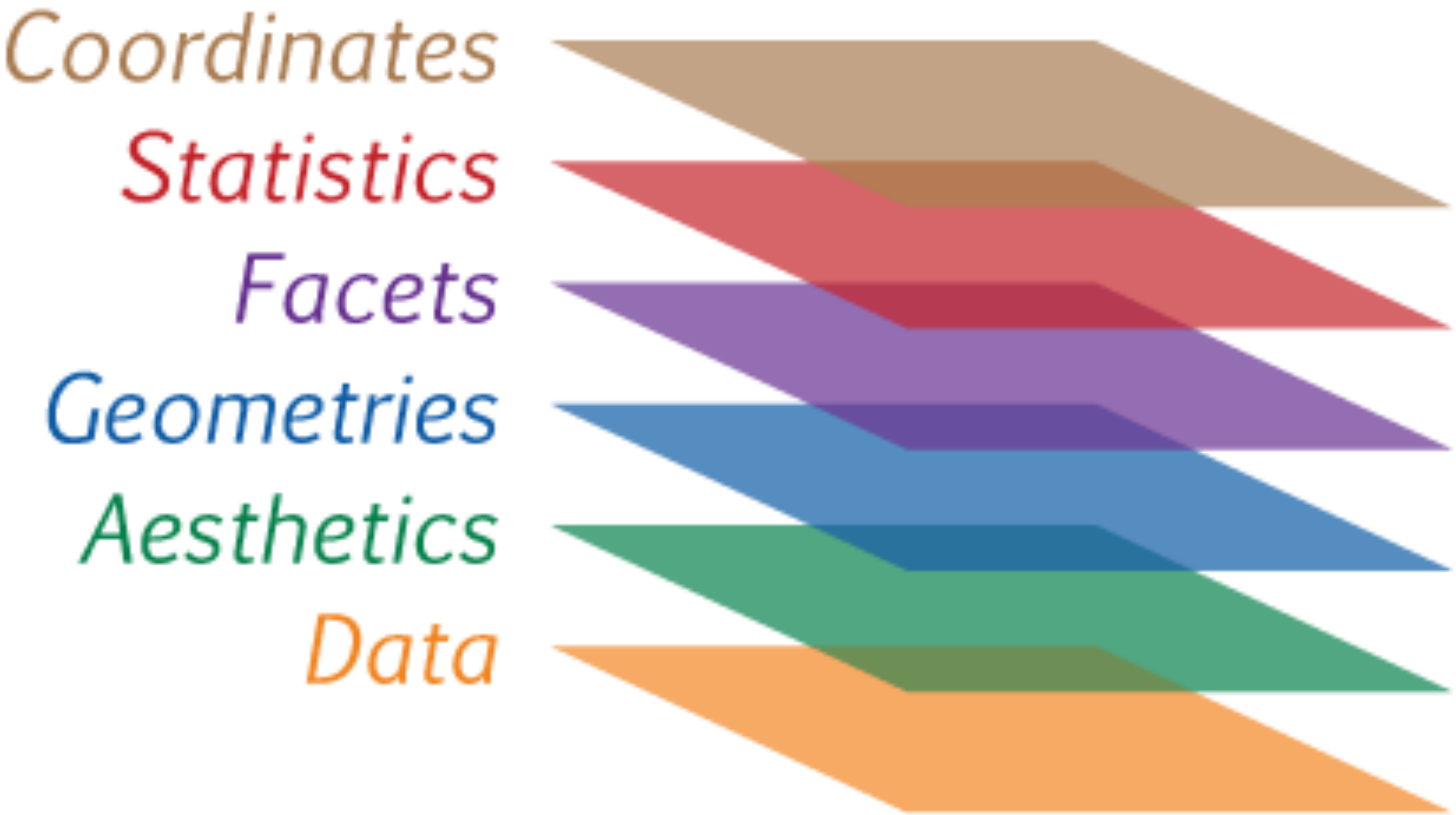
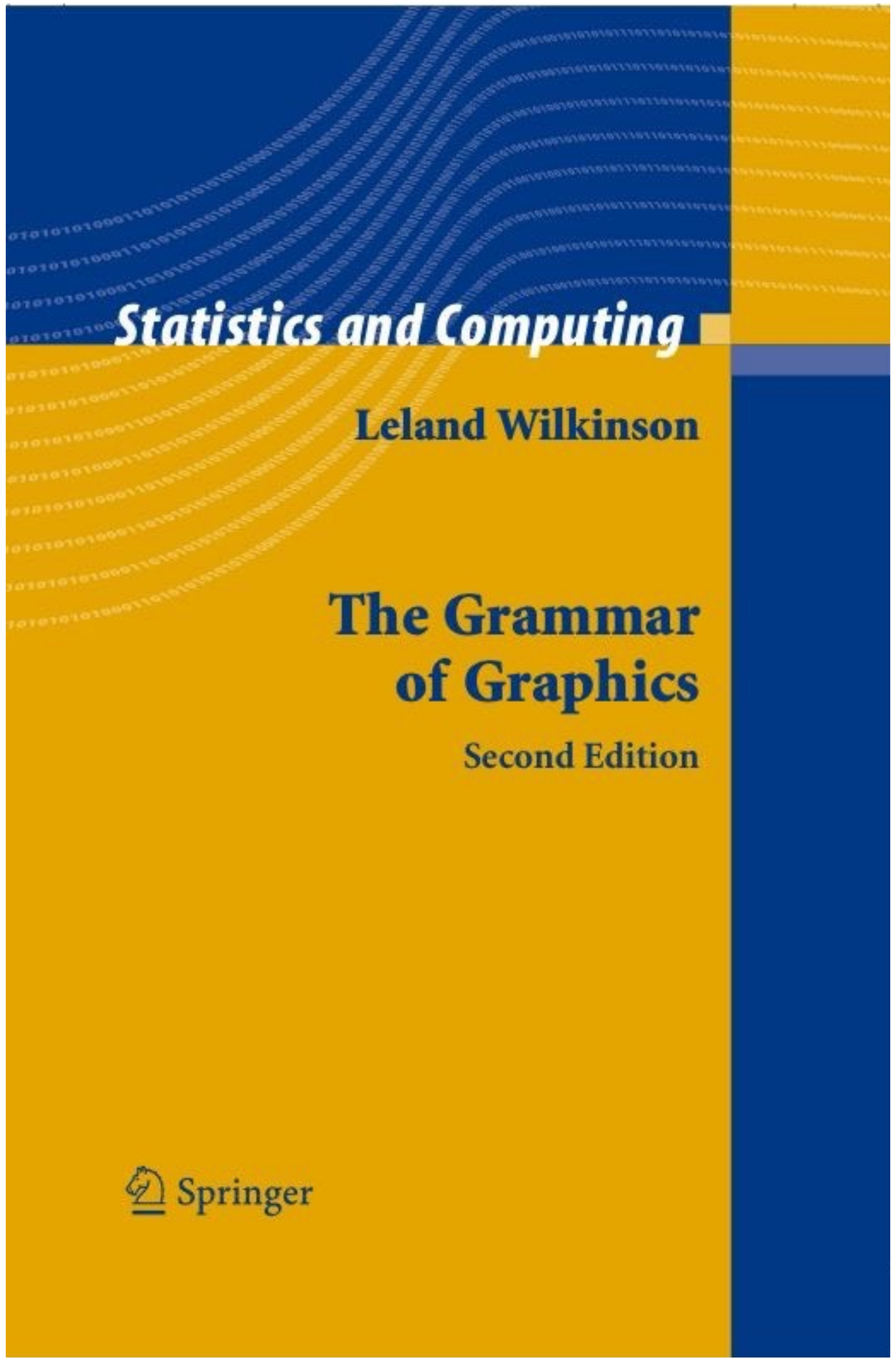
more ggplot2

Modified from “Data Science in the tidyverse” materials. Major contributions by Garrett Grolemund, Amelia McNamara, Charlotte Wickham, and Hadley Wickham.

Review: Grammar of Graphics

The Grammar of Graphics




Statistics and Computing

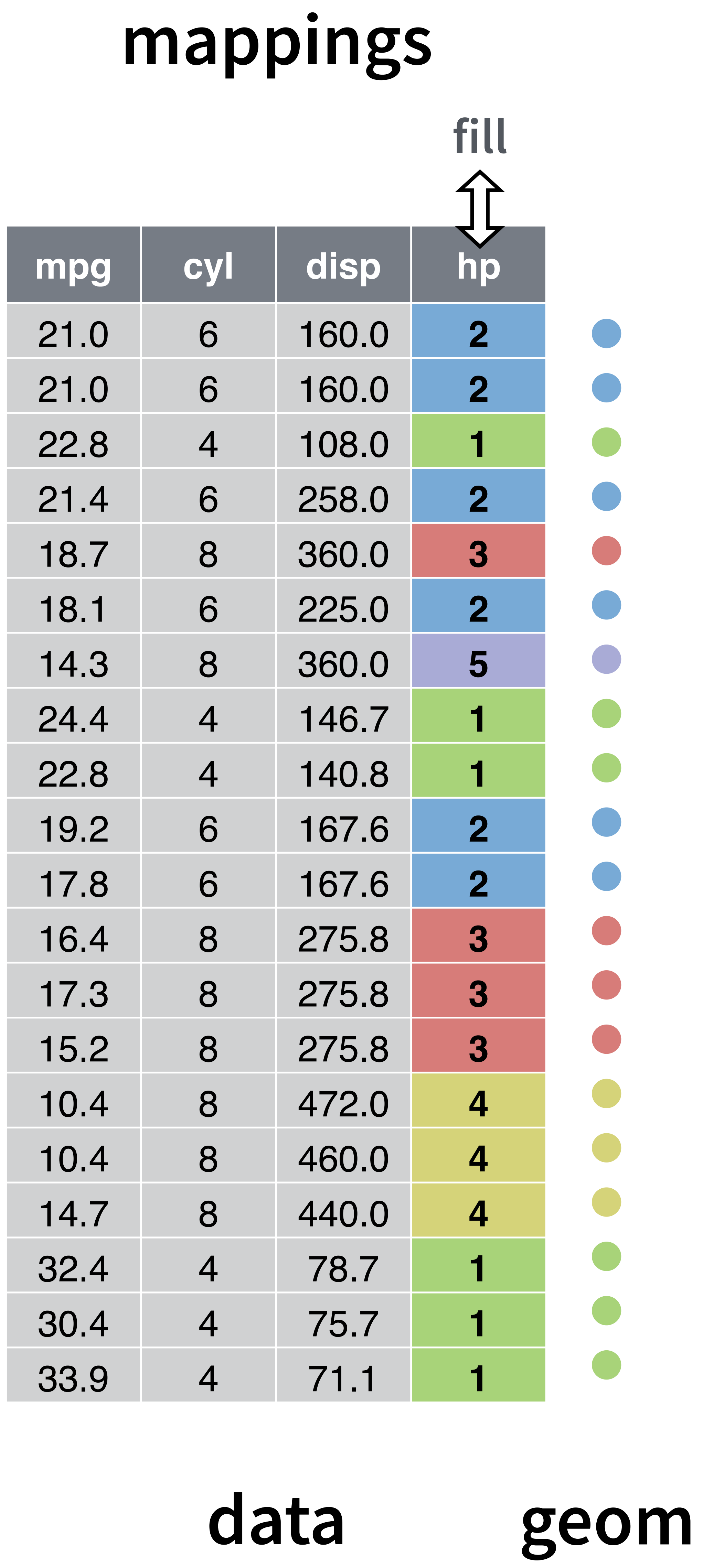
Leland Wilkinson

The Grammar of Graphics

Second Edition

Springer

To make a graph



mpg	cyl	disp	hp	
21.0	6	160.0	2	●
21.0	6	160.0	2	●
22.8	4	108.0	1	●
21.4	6	258.0	2	●
18.7	8	360.0	3	●
18.1	6	225.0	2	●
14.3	8	360.0	5	●
24.4	4	146.7	1	●
22.8	4	140.8	1	●
19.2	6	167.6	2	●
17.8	6	167.6	2	●
16.4	8	275.8	3	●
17.3	8	275.8	3	●
15.2	8	275.8	3	●
10.4	8	472.0	4	●
10.4	8	460.0	4	●
14.7	8	440.0	4	●
32.4	4	78.7	1	●
30.4	4	75.7	1	●
33.9	4	71.1	1	●

1. Pick a **data** set

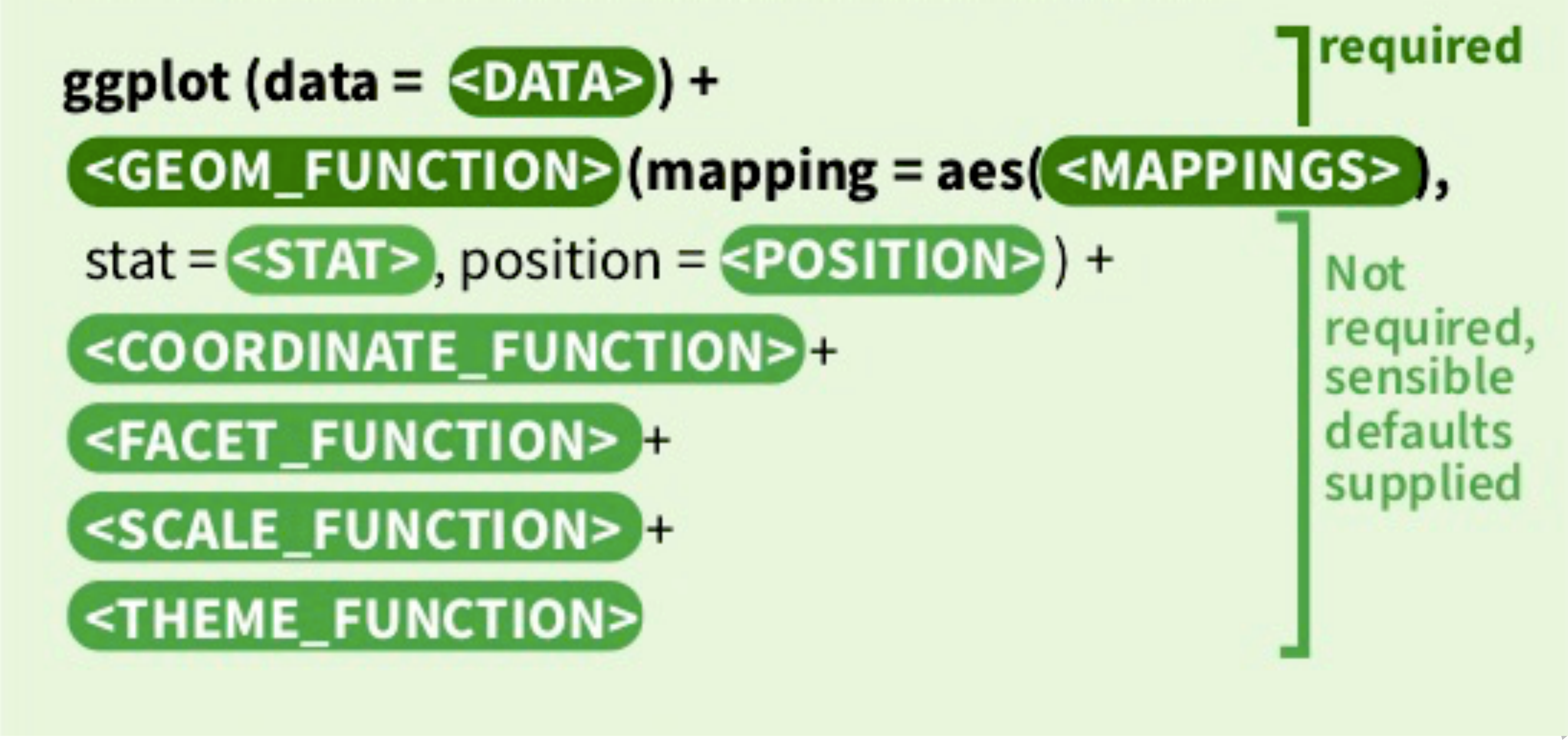
```
ggplot(data = <DATA>) +
  <GEOM_FUNCTION>(mapping = aes(<MAPPINGS>))
```

2. Choose a **geom** to display cases

3. **Map** aesthetic properties to variables

A ggplot2 template

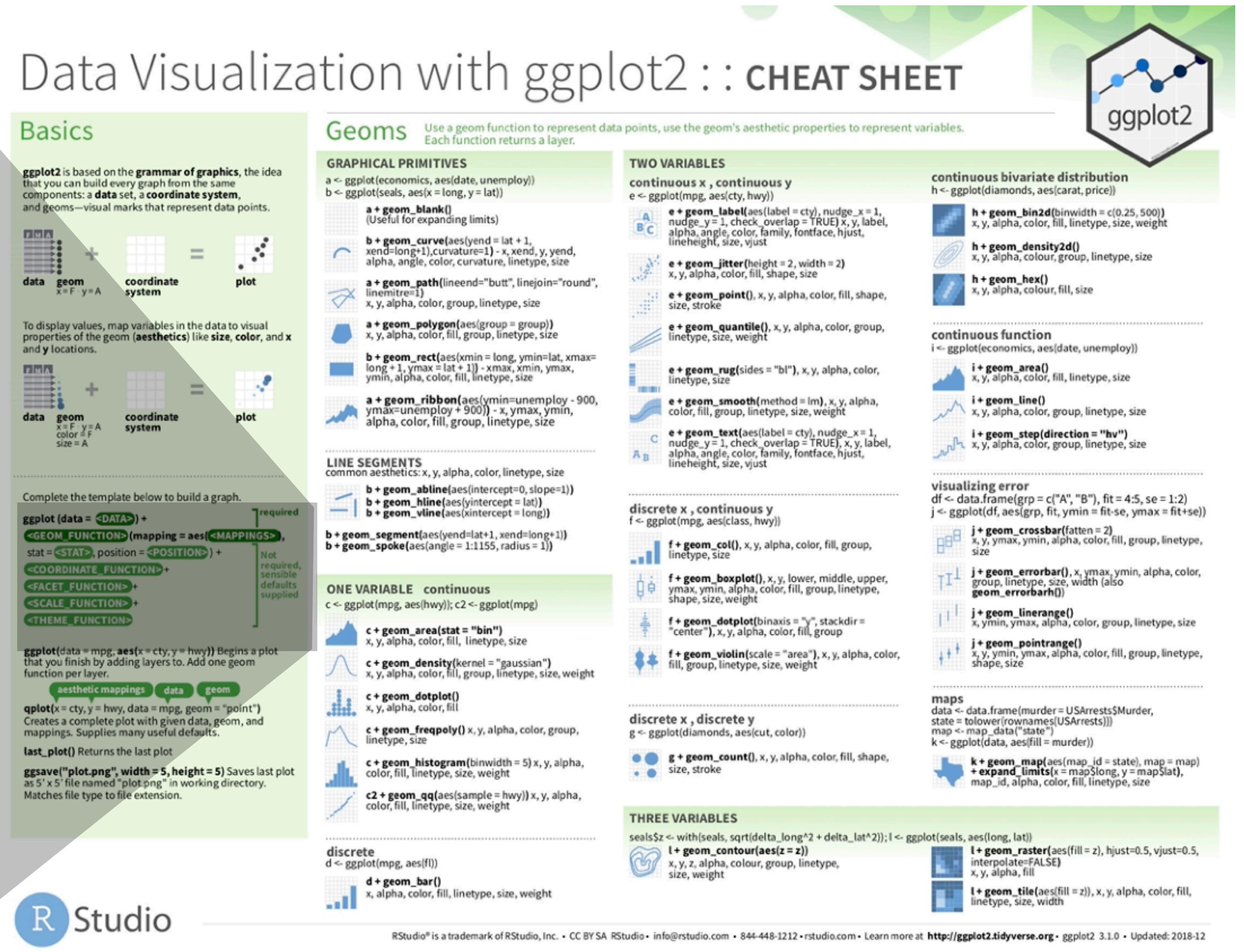
Make any plot by filling in the parameters of this template



```
ggplot (data = <DATA>) +
<GEOM_FUNCTION>(mapping = aes(<MAPPINGS>),
  stat = <STAT>, position = <POSITION>) +
<COORDINATE_FUNCTION> +
<FACET_FUNCTION> +
<SCALE_FUNCTION> +
<THEME_FUNCTION>
```

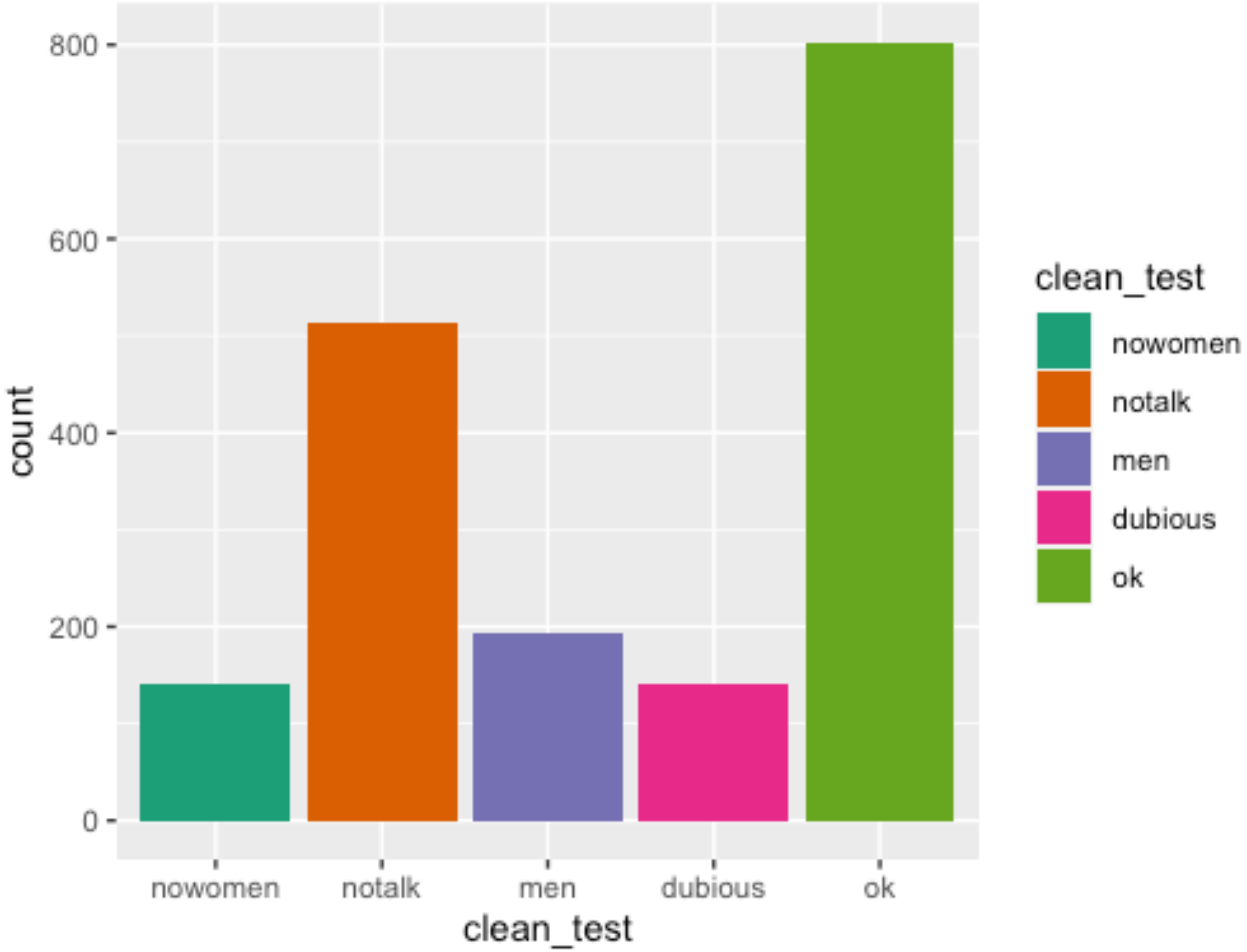
required

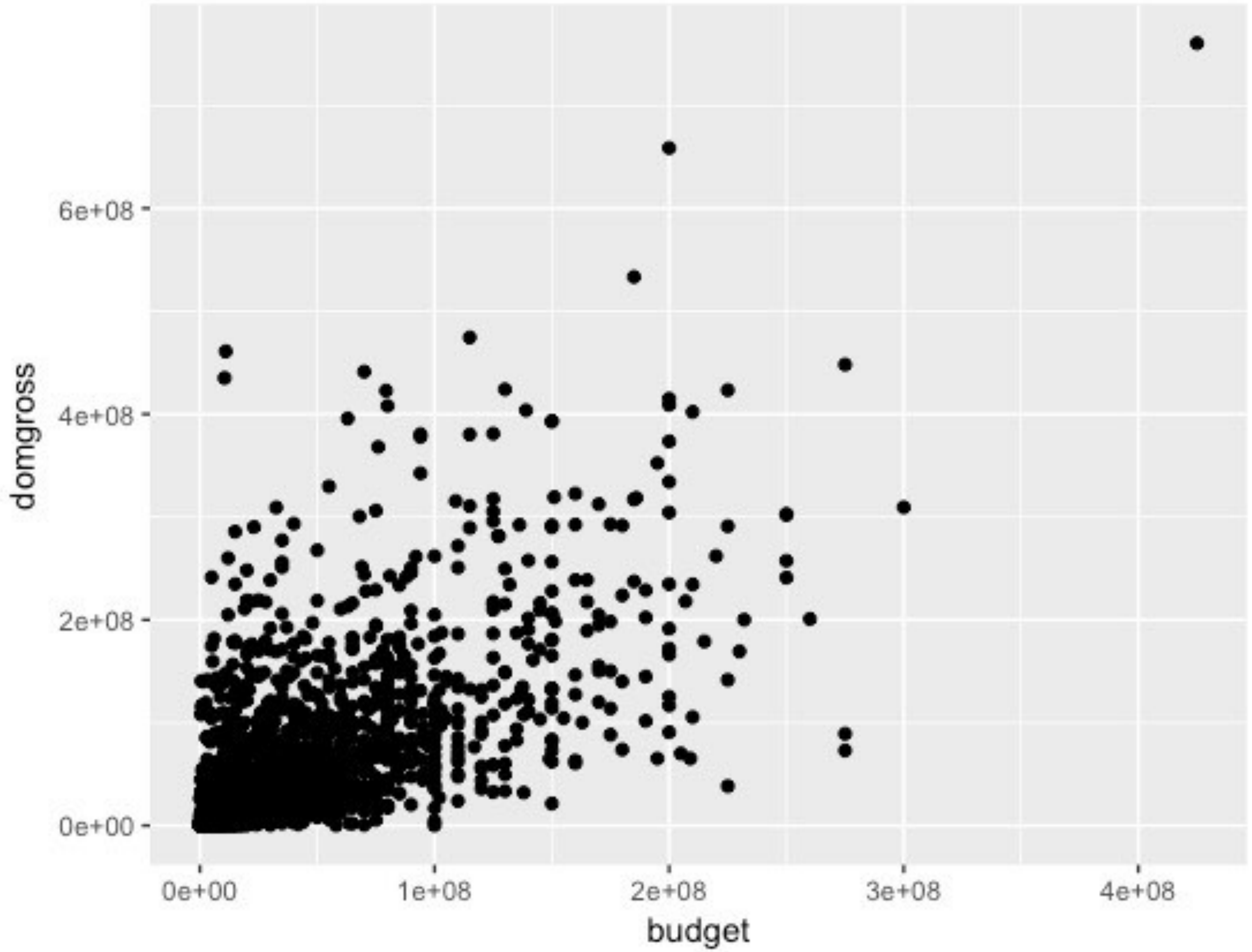
Not required, sensible defaults supplied



Review: Basic ggplot2 viz

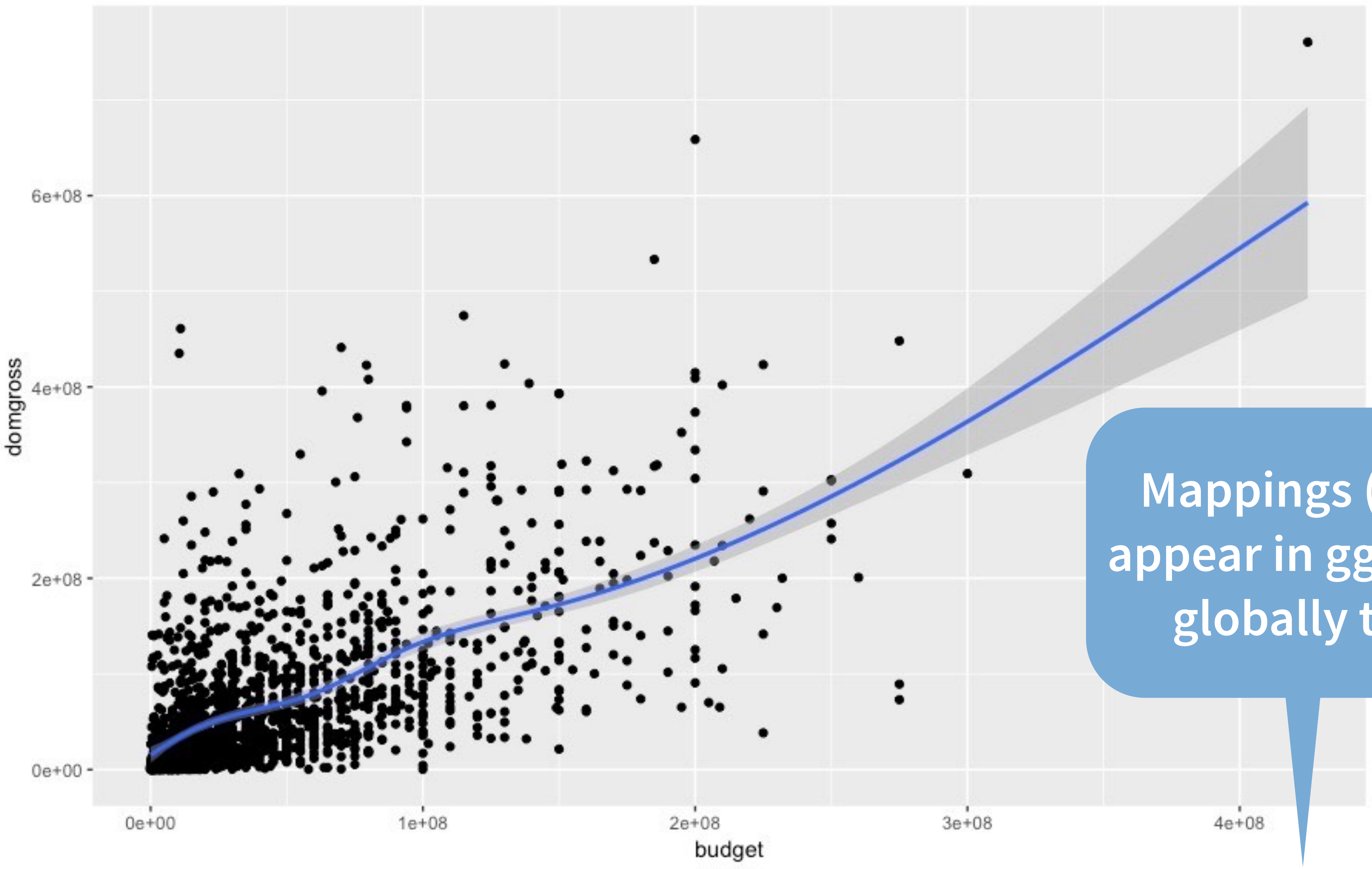
```
ggplot(data=bechdel) +
    geom_bar(aes(x=clean_test, fill=clean_test)) +
    scale_fill_brewer(palette = "Dark2")
```





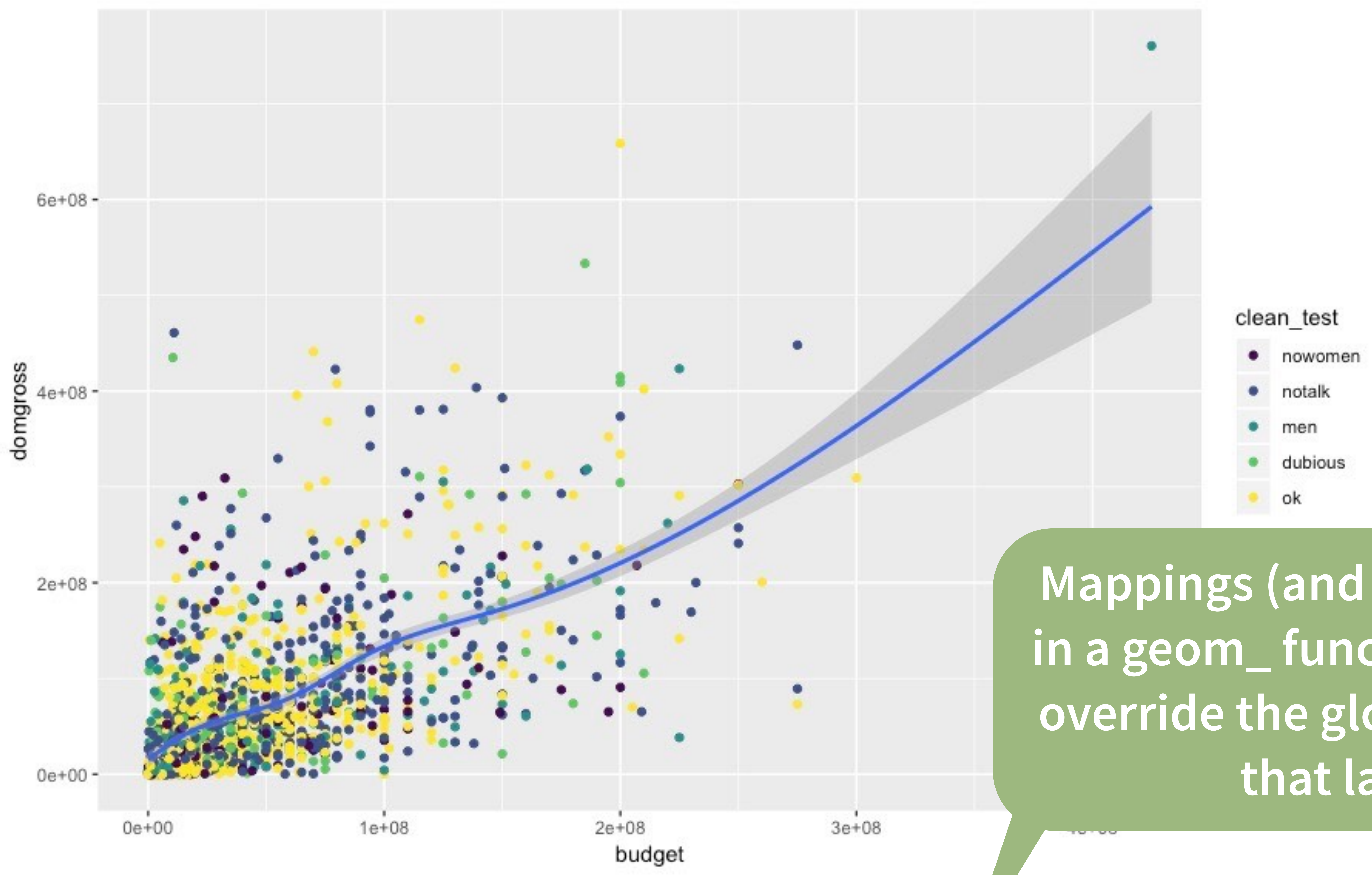
```
ggplot(data = bechdel) +
  geom_point(mapping = aes(x = budget, y = domgross))
```

global vs. local



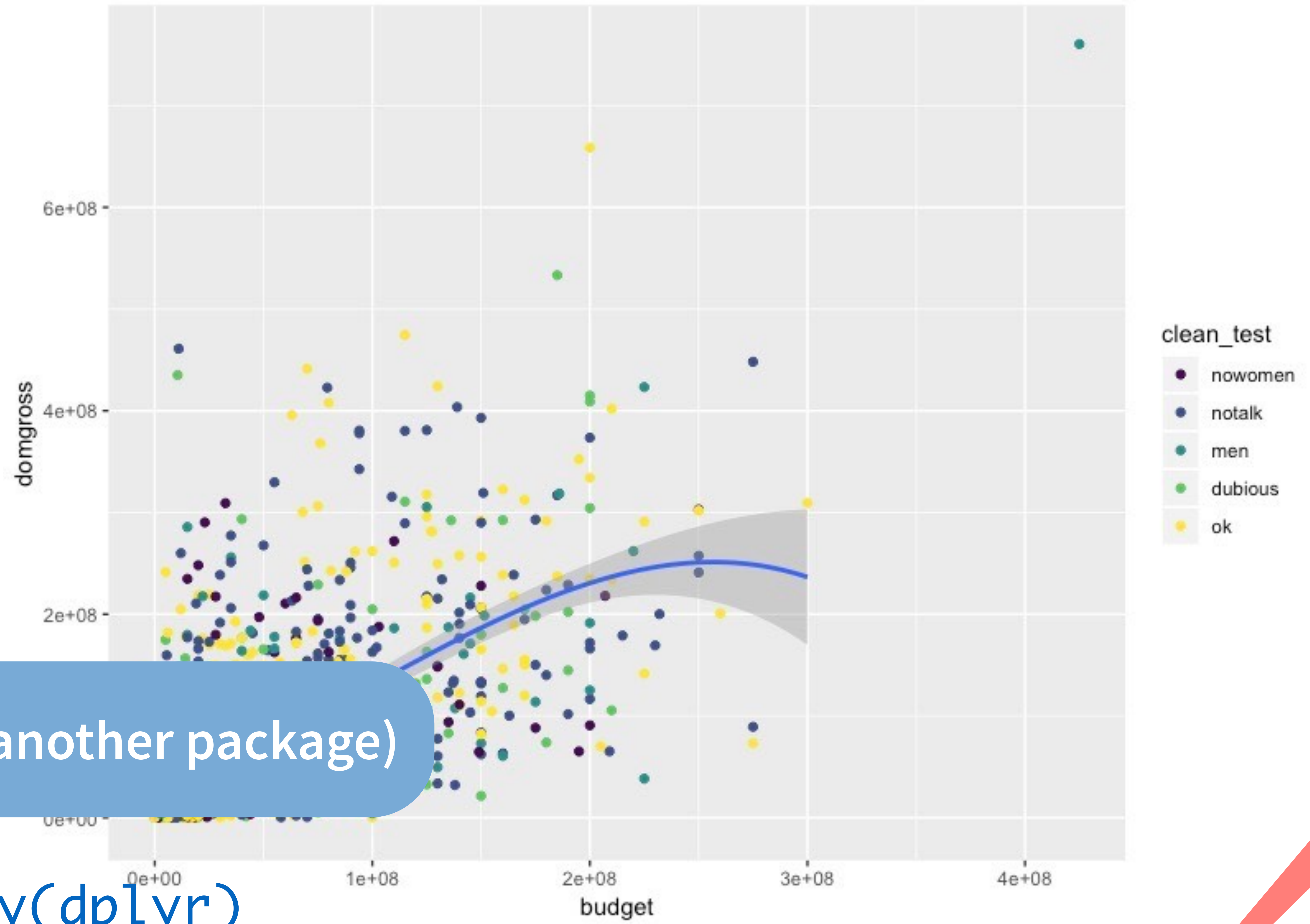
Mappings (and data) that appear in ggplot() will apply globally to every layer

```
ggplot(data = bechdel, mapping = aes(x = budget, y = domgross)) +
  geom_point() +
  geom_smooth()
```



Mappings (and data) that appear in a geom_ function will add to or override the global mappings for that layer only

```
ggplot(data = bechdel, mapping = aes(x = budget, y = domgross)) +
  geom_point(mapping = aes(color = clean_test)) +
  geom_smooth()
```



(requires another package)

data can also be set locally or globally

```
library(dplyr)
ggplot(data = bechdel, mapping = aes(x = budget, y = domgross)) +
  geom_point(mapping = aes(color = clean_test)) +
  geom_smooth(data = filter(bechdel, clean_test == "ok"))
```

Controlling the details

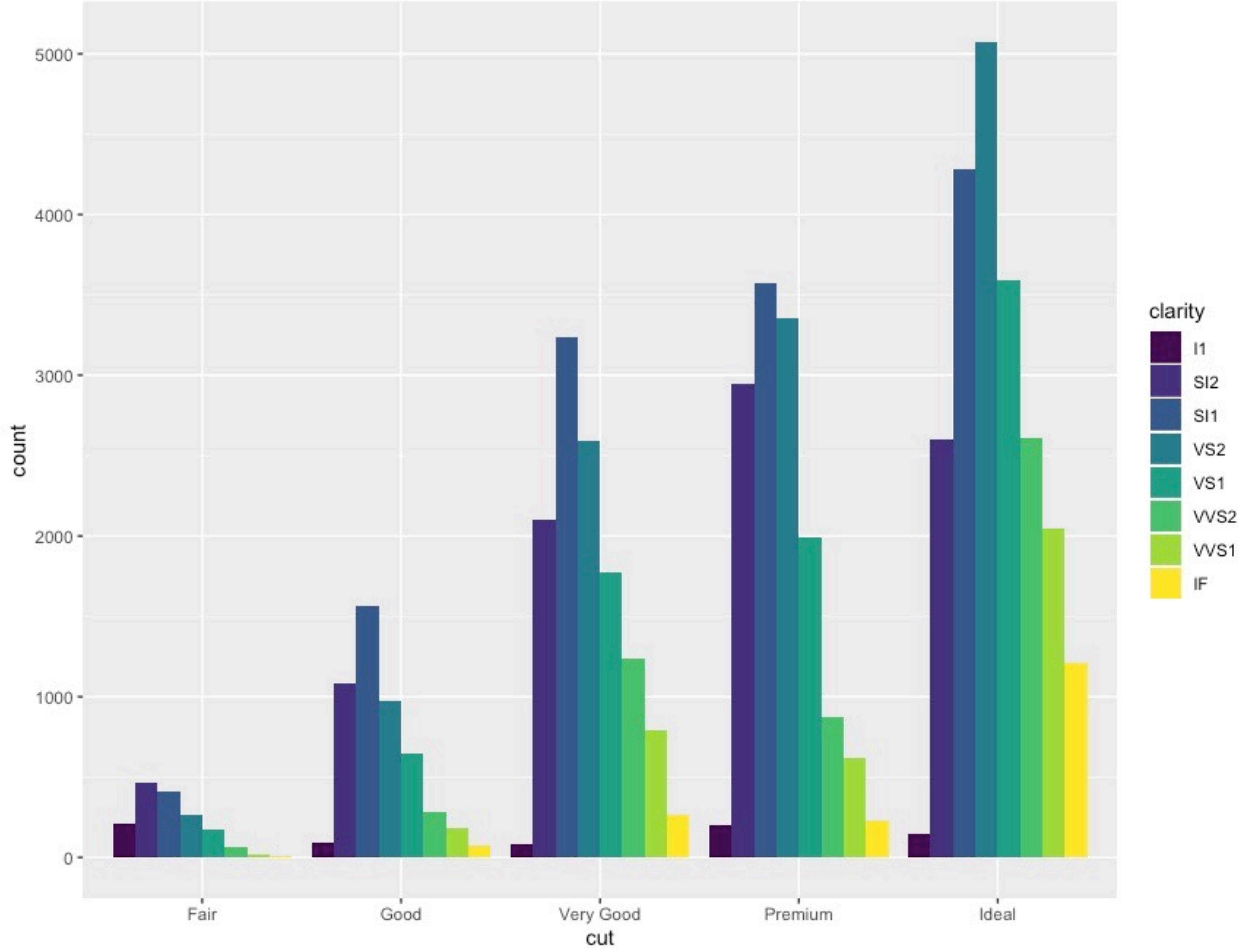
Defaults in ggplot2 were chosen based on research

- Grey background
- Light gridlines
- Colorblind-friendly color scales
- Colorscales that match continuous and discrete variables

…but we don’t always want the default

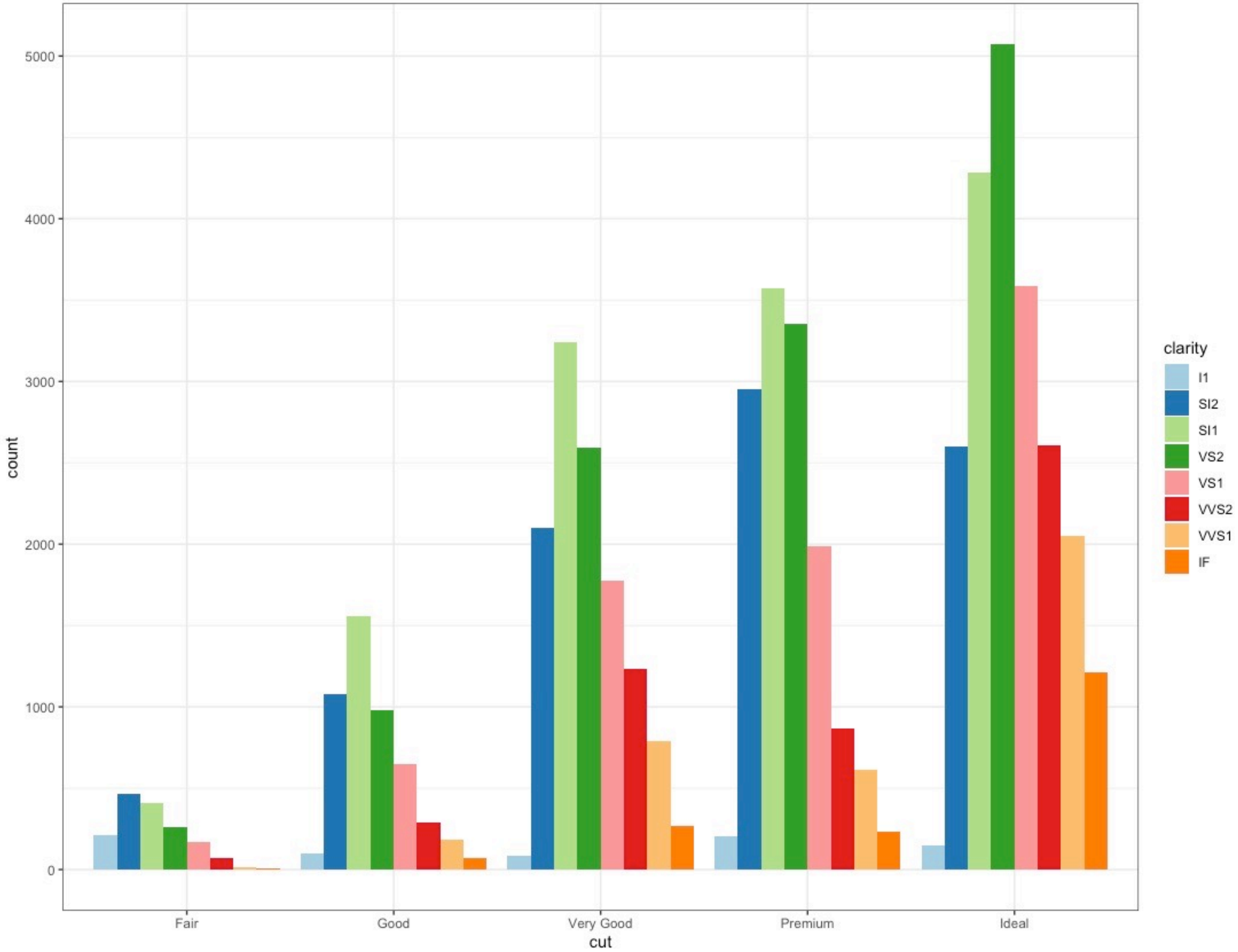
Position Adjustments

How overlapping objects are arranged



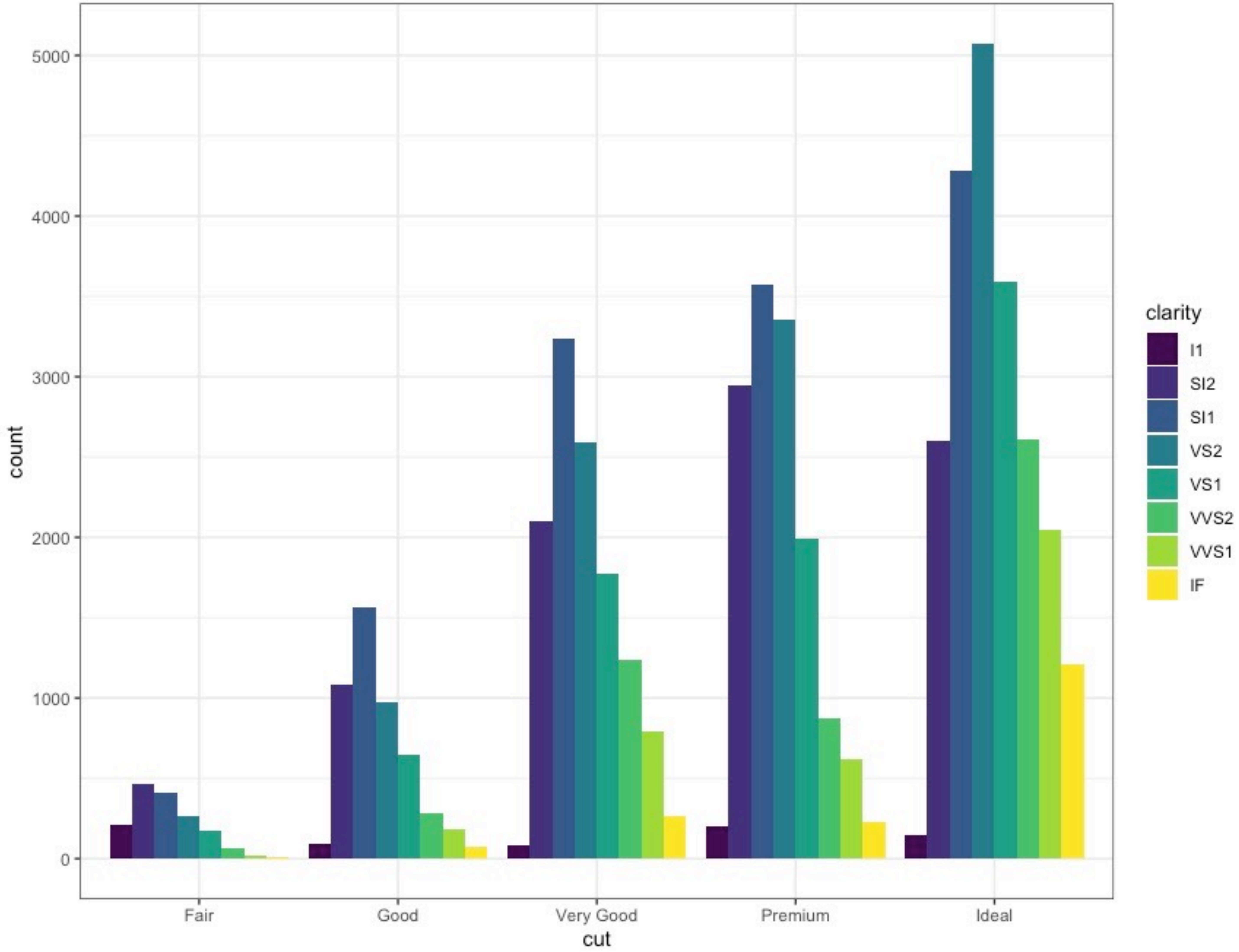
Scales

Customize color scales, other mappings

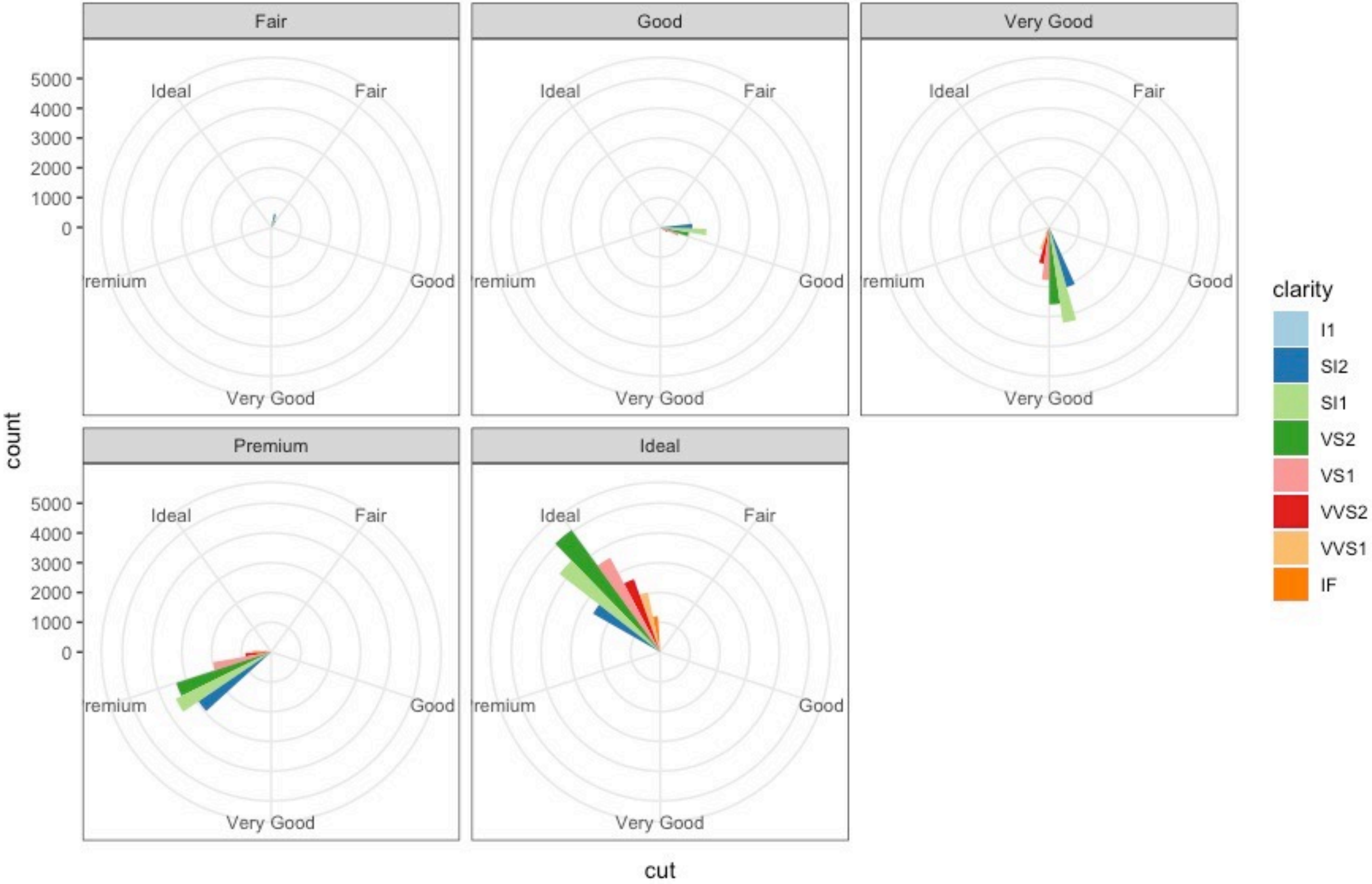


Themes

Visual appearance of non-data elements



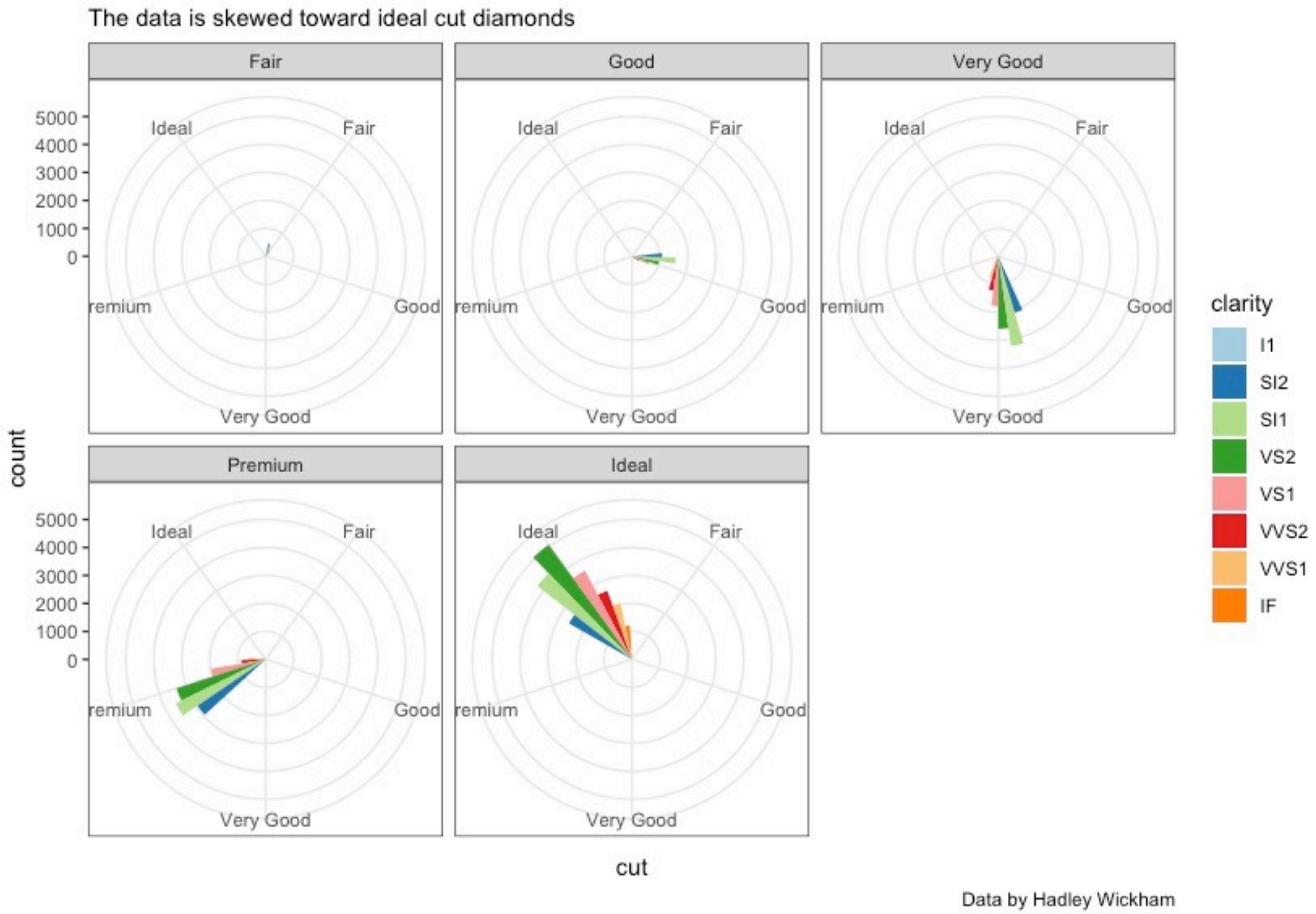
Coordinate systems



Titles and captions

Diamonds data

The data set is skewed towards ideal cut diamonds

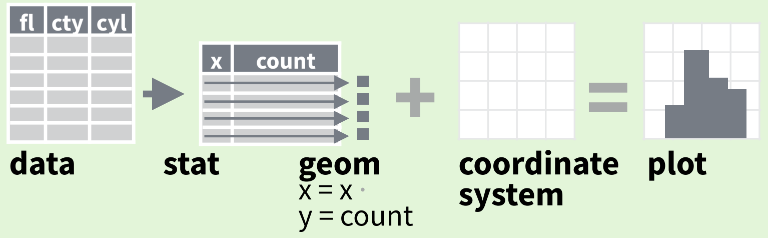


Data by Hadley Wickham

Stats

An alternative way to build a layer.

A stat builds new variables to plot (e.g., count, prop).



Visualize a stat by changing the default stat of a geom function, **geom_bar(stat="count")** or by using a stat function, **stat_count(geom="bar")**, which calls a default geom to make a layer (equivalent to a geom function).
Use **after_stat(name)** syntax to map the stat variable **name** to an aesthetic.

geom to use · stat function · geommappings

i + stat_density_2d(aes(fill = after_stat(level)), geom = "polygon"**)**

variable created by stat

c + stat_bin(binwidth = 1, boundary = 10**)**
x, y | count, ncount, density, ndensity

c + stat_count(width = 1**) x, y** | count, prop

c + stat_density(adjust = 1, kernel = "gaussian"**)**
x, y | count, density, scaled

e + stat_bin_2d(bins = 30, drop = T**)**
x, y, fill | count, density

e + stat_bin_hex(bins = 30**) x, y, fill** | count, density

e + stat_density_2d(contour = TRUE, n = 100**)**
x, y, color, size | level

e + stat_ellipse(level = 0.95, segments = 51, type = "t"**)**

l + stat_contour(aes(z = z)**) x, y, z, order** | level

l + stat_summary_hex(aes(z = z), bins = 30, fun = max**)**
x, y, z, fill | value

l + stat_summary_2d(aes(z = z), bins = 30, fun = mean**)**
x, y, z, fill | value

f + stat_boxplot(coef = 1.5**)**
x, y | lower, middle, upper, width , ymin, ymax

f + stat_ydensity(kernel = "gaussian", scale = "area"**) x, y** | density, scaled, count, n, violinwidth, width

e + stat_ecdf(n = 40**) x, y** | x, y

e + stat_quantile(quantiles = c(0.1, 0.9), formula = y ~ log(x), method = "rq"**) x, y** | quantile

e + stat_smooth(method = "lm", formula = y ~ x, se = T, level = 0.95**) x, y** | se, x, y, ymin, ymax

ggplot() + xlim(-5, 5**) + stat_function(**fun = dnorm, n = 20, geom = "point"**) x** | x, y

ggplot() + stat_qq(aes(sample = 1:100)**)**
x, y, sample | sample, theoretical

e + stat_sum() x, y, size | n, prop

e + stat_summary(fun.data = "mean_cl_boot"**)**

h + stat_summary_bin(fun = "mean", geom = "bar"**)**

e + stat_identity()

e + stat_unique()

Scales

Override defaults with **scales** package.

Scales map data values to the visual values of an aesthetic. To change a mapping, add a new scale.

n <- d + geom_bar(aes(fill = fl))

scale_ · aesthetic to adjust · prepackaged scale to use · scale-specific arguments

```
n + scale_fill_manual(
    values = c("skyblue", "royalblue", "blue", "navy"),
    limits = c("d", "e", "p", "r"), breaks =c("d", "e", "p", “r"),
    name = "fuel", labels = c("D", "E", "P", "R"))
```

range of values to include in · title to use in legend/axis · labels to use in legend/axis · breaks to use in legend/axis

GENERAL PURPOSE SCALES

Use with most aesthetics

scale_*_continuous() - Map cont’ values to visual ones.
scale_*_discrete() - Map discrete values to visual ones.
scale_*_binned() - Map continuous values to discrete bins.
scale_*_identity() - Use data values as visual ones.
scale_*_manual(values = c()**)** - Map discrete values to manually chosen visual ones.
scale_*_date(date_labels = "%m/%d"), date_breaks = "2 weeks"**)** - Treat data values as dates.
scale_*_datetime() - Treat data values as date times. Same as scale_*_date(). See ?strptime for label formats.

X & Y LOCATION SCALES

Use with x or y aesthetics (x shown here)

scale_x_log10() - Plot x on log10 scale.
scale_x_reverse() - Reverse the direction of the x axis.
scale_x_sqrt() - Plot x on square root scale.

COLOR AND FILL SCALES (DISCRETE)

n + scale_fill_brewer(palette = "Blues"**)**
For palette choices:
RColorBrewer::display.brewer.all()

n + scale_fill_grey(start = 0.2, end = 0.8, na.value = "red"**)**

COLOR AND FILL SCALES (CONTINUOUS)

o <- c + geom_dotplot(aes(fill = x))

o + scale_fill_distiller(palette = “Blues”**)**

o + scale_fill_gradient(low="red", high=“yellow"**)**

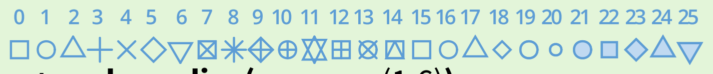
o + scale_fill_gradient2(low = "red", high = “blue”, mid = "white", midpoint = 25**)**

o + scale_fill_gradientn(colors = topo.colors(6)**)**
Also: rainbow(), heat.colors(), terrain.colors(), cm.colors(), RColorBrewer::brewer.pal()

SHAPE AND SIZE SCALES

p <- e + geom_point(aes(shape = fl, size = cyl))

p + scale_shape() + scale_size()
p + scale_shape_manual(values = c(3:7)**)**



p + scale_radius(range = c(1,6)**)**
p + scale_size_area(max_size = 6**)**

Coordinate Systems

r <- d + geom_bar()

r + coord_cartesian(xlim = c(0, 5)**)** - xlim, ylim
The default cartesian coordinate system.

r + coord_fixed(ratio = 1/2**)**
ratio, xlim, ylim - Cartesian coordinates with fixed aspect ratio between x and y units.

r + coord_flip()
Flip cartesian coordinates by switching x and y aesthetic mappings.

r + coord_polar(theta = "x", direction=1**)**
theta, start, direction - Polar coordinates.

r + coord_trans(y = “sqrt"**)** - x, y, xlim, ylim
Transformed cartesian coordinates. Set xtrans and ytrans to the name of a window function.

π + coord_quickmap()
π + coord_map(projection = "ortho", orientation = c(41, -74, 0)**)** - projection, xlim, ylim
Map projections from the mapproj package (mercator (default), azequalarea, lagrange, etc.).

Position Adjustments

Position adjustments determine how to arrange geoms that would otherwise occupy the same space.

s <- ggplot(mpg, aes(fl, fill = drv))

s + geom_bar(position = "dodge")
Arrange elements side by side.

s + geom_bar(position = "fill")
Stack elements on top of one another, normalize height.

e + geom_point(position = "jitter")
Add random noise to X and Y position of each element to avoid overplotting.

e + geom_label(position = "nudge")
Nudge labels away from points.

s + geom_bar(position = "stack")
Stack elements on top of one another.

Each position adjustment can be recast as a function with manual **width** and **height** arguments:
s + geom_bar(position = position_dodge(width = 1))

Themes

r + theme_bw()
White background with grid lines.

r + theme_gray()
Grey background (default theme).

r + theme_dark()
Dark for contrast.

r + theme_classic()

r + theme_light()

r + theme_linedraw()

r + theme_minimal()
Minimal theme.

r + theme_void()
Empty theme.

r + theme() Customize aspects of the theme such as axis, legend, panel, and facet properties.
r + ggtitle(“Title”) + theme(plot.title.postion = “plot”)
r + theme(panel.background = element_rect(fill = “blue”))

Faceting

Facets divide a plot into subplots based on the values of one or more discrete variables.

t <- ggplot(mpg, aes(cty, hwy)) + geom_point()

t + facet_grid(. ~ fl)
Facet into columns based on fl.

t + facet_grid(year ~ .)
Facet into rows based on year.

t + facet_grid(year ~ fl)
Facet into both rows and columns.

t + facet_wrap(~ fl)
Wrap facets into a rectangular layout.

Set **scales** to let axis limits vary across facets.

t + facet_grid(drv ~ fl, scales = "free")
x and y axis limits adjust to individual facets:
"free_x" - x axis limits adjust
"free_y" - y axis limits adjust

Set **labeller** to adjust facet label:

t + facet_grid(. ~ fl, labeller = label_both)

fl: c	fl: d	fl: e	fl: p	fl: r

t + facet_grid(fl ~ ., labeller = label_bquote(alpha ^ .(fl)**))**

α^c	α^d	α^e	α^p	α^r

Labels and Legends

Use **labs()** to label the elements of your plot.

t + labs(x = "New x axis label", **y** = "New y axis label",
title ="Add a title above the plot",
subtitle = "Add a subtitle below title",
caption = "Add a caption below plot",
alt = "Add alt text to the plot",
<AES> = "New <AES> legend title"**)**

t + annotate(geom = "text", x = 8, y = 9, label = “A"**)**
Places a geom with manually selected aesthetics.

p + guides(x = guide_axis(n.dodge = 2)**)** Avoid crowded or overlapping labels with guide_axis(n.dodge or angle).

n + guides(fill = “none"**)** Set legend type for each aesthetic: colorbar, legend, or none (no legend).

n + theme(legend.position = "bottom"**)**
Place legend at "bottom", "top", "left", or “right”.

n + scale_fill_discrete(name = "Title", labels = c("A", "B", "C", "D", "E")**)**
Set legend title and labels with a scale function.

Zooming

Without clipping (preferred):

t + coord_cartesian(xlim = c(0, 100), ylim = c(10, 20)**)**

With clipping (removes unseen data points):

t + xlim(0, 100**) + ylim(**10, 20**)**

t + scale_x_continuous(limits = c(0, 100)**) + scale_y_continuous(**limits = c(0, 100)**)**

• info@posit.co • posit.co • Learn more at **ggplot2.tidyverse.org** • HTML cheatsheets at **pos.it/cheatsheets** • ggplot2 3.4.2 • Updated: 2023-07

Copy the masters

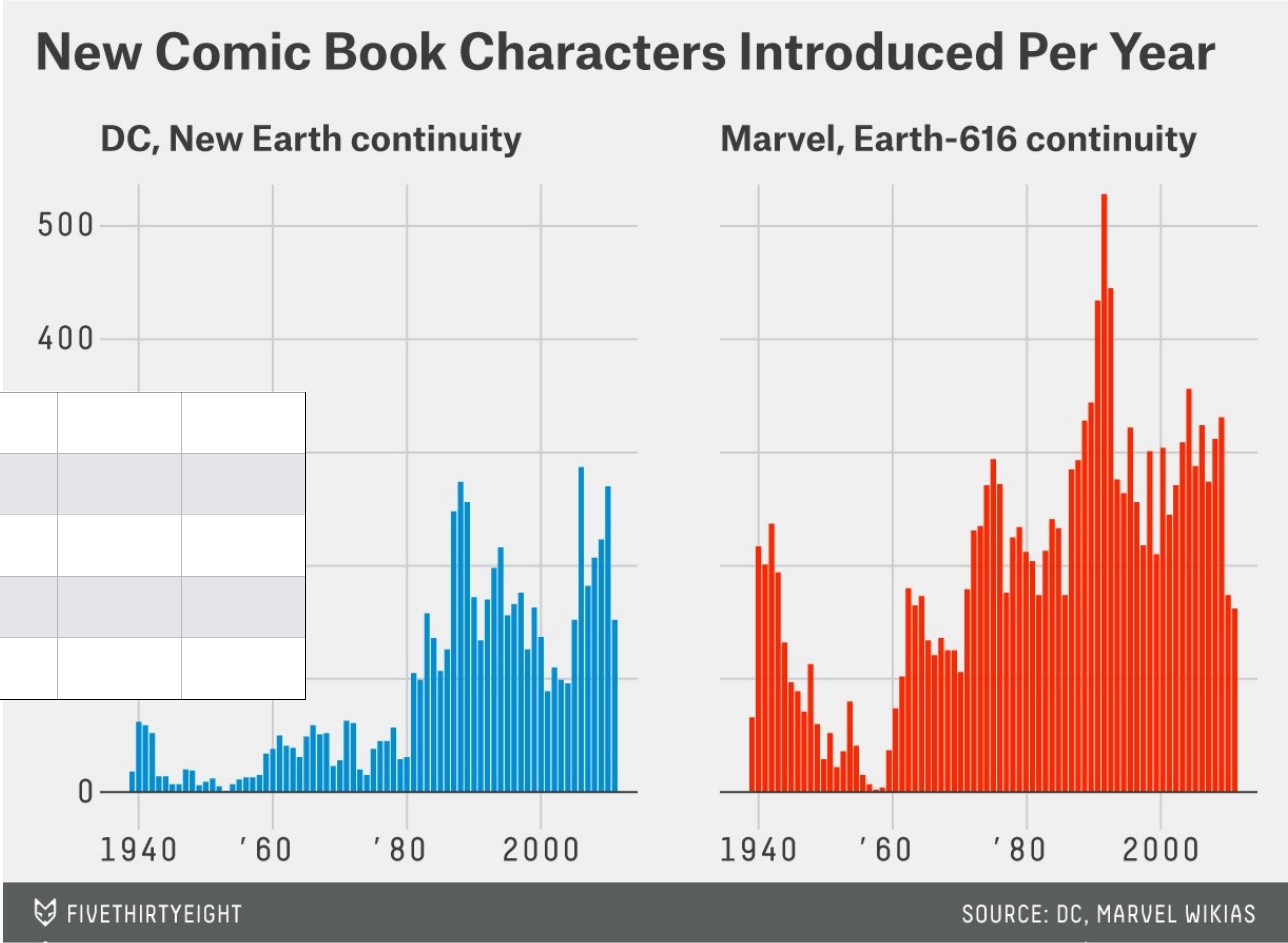
- You are (eventually) going to reproduce a plot made by the folks at FiveThirtyEight

- This is a hard assignment for two reasons

  1. There are lots of finicky settings for all sorts of plot details

  2. Data needs to be in the right format before you can easily plot it

Brainstorm: what does the dataset that produced this graph look like?



New Comic Book Characters Introduced Per Year

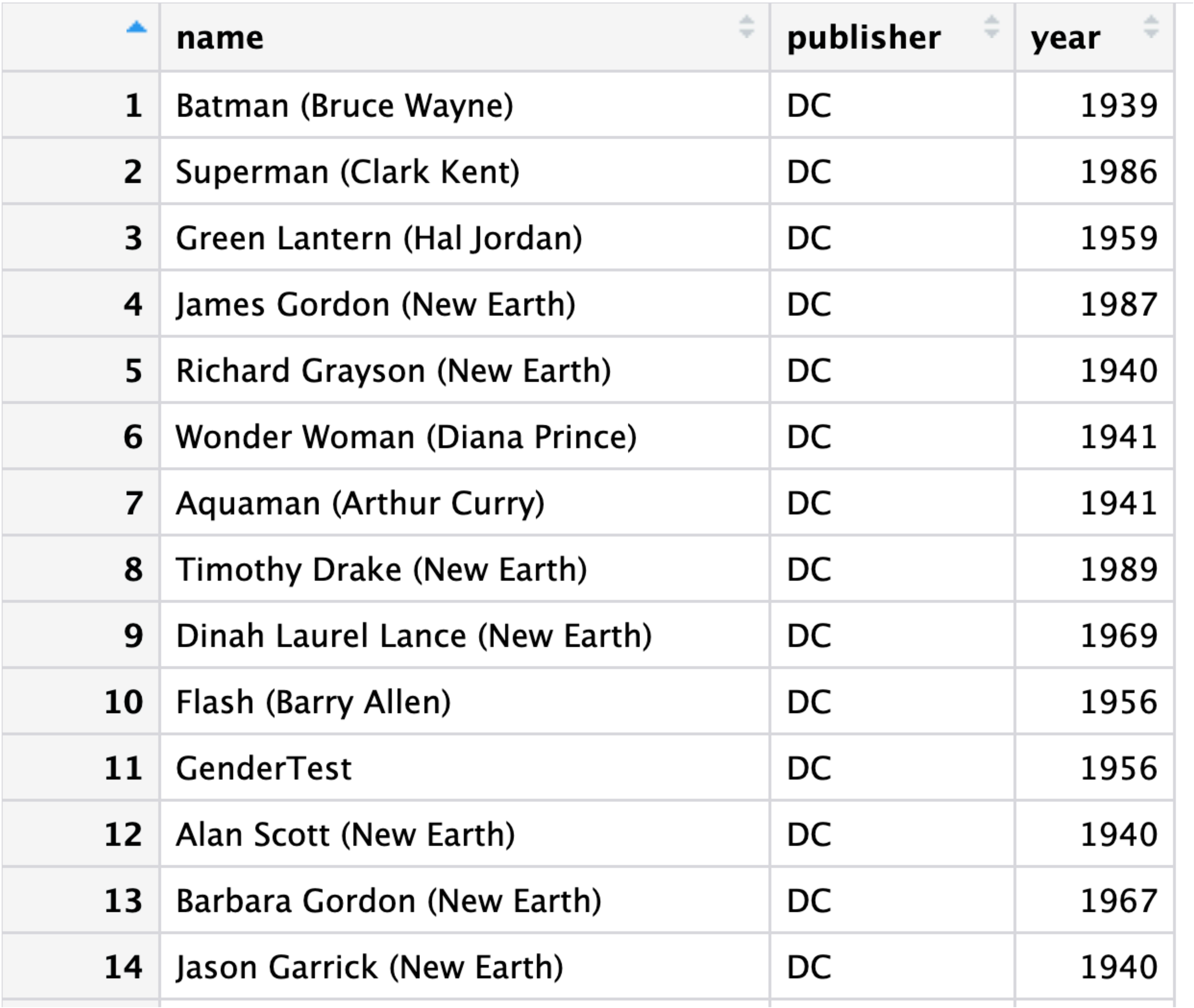
FIVETHIRTYEIGHT

SOURCE: DC, MARVEL WIKIAS

Your Turn 1

Open comic_characters.qmd

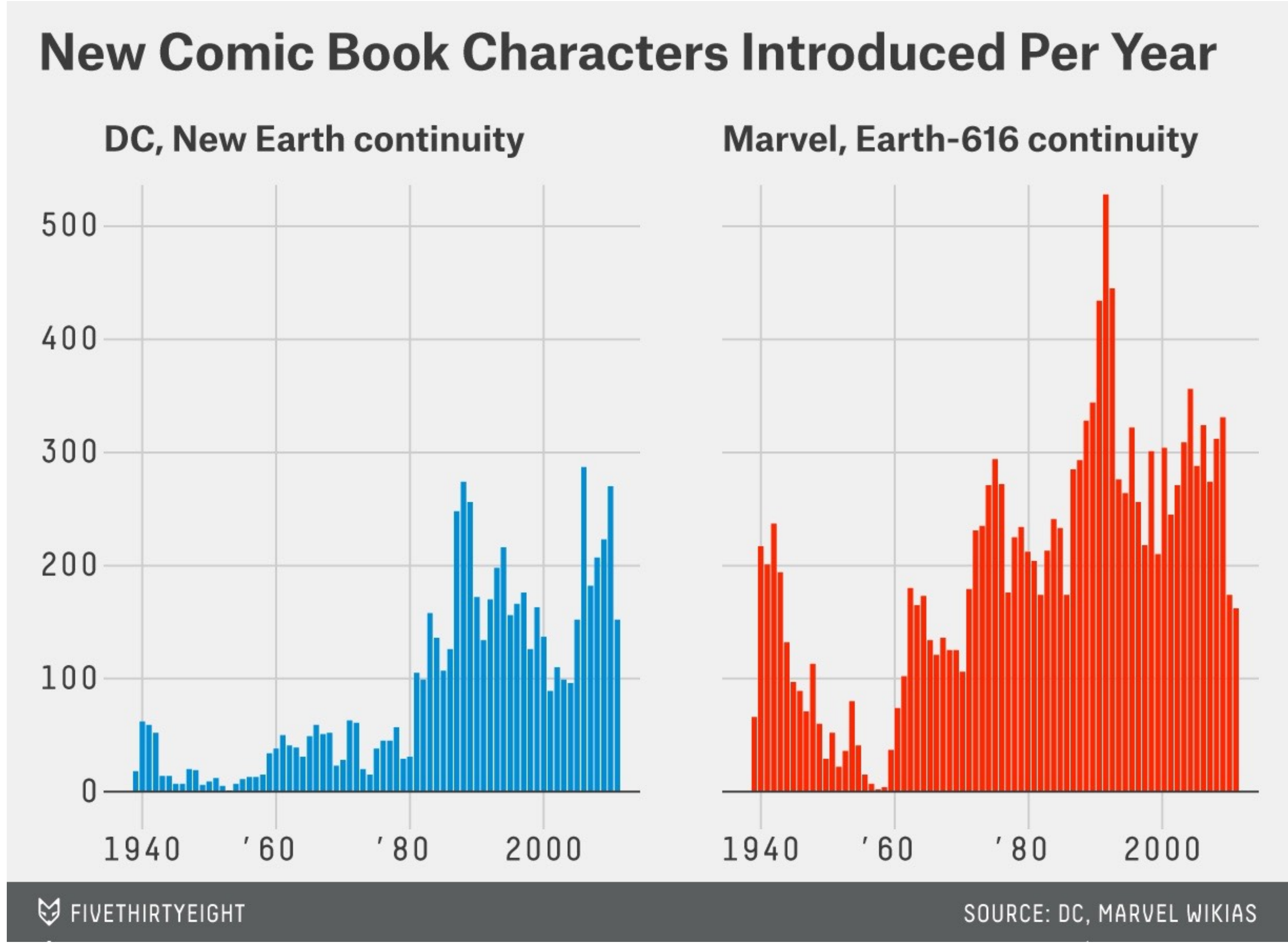
Read through the data-wrangling code, and make comments where you don’t understand what’s happening.



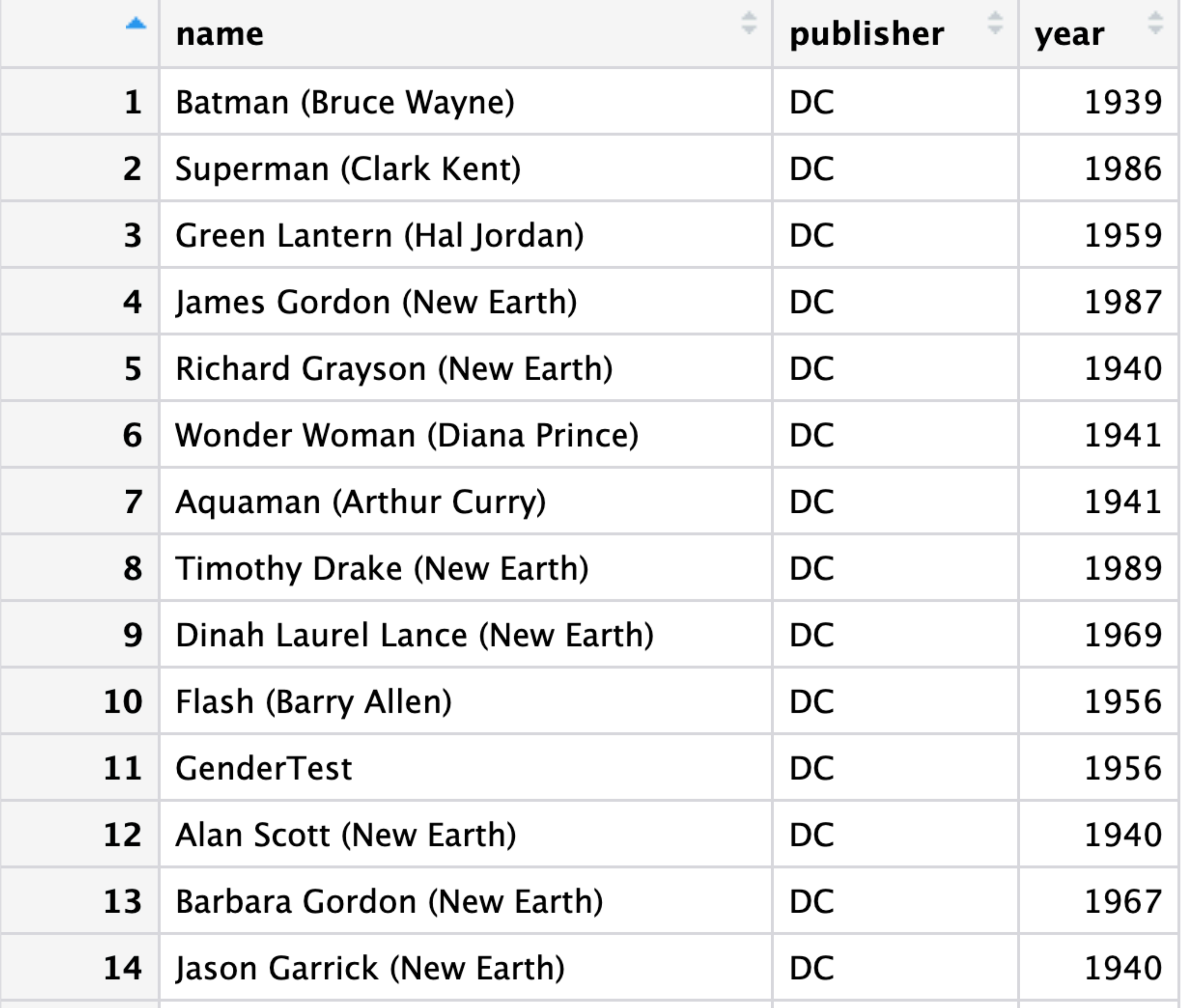
	name	publisher	year
1	Batman (Bruce Wayne)	DC	1939
2	Superman (Clark Kent)	DC	1986
3	Green Lantern (Hal Jordan)	DC	1959
4	James Gordon (New Earth)	DC	1987
5	Richard Grayson (New Earth)	DC	1940
6	Wonder Woman (Diana Prince)	DC	1941
7	Aquaman (Arthur Curry)	DC	1941
8	Timothy Drake (New Earth)	DC	1989
9	Dinah Laurel Lance (New Earth)	DC	1969
10	Flash (Barry Allen)	DC	1956
11	GenderTest	DC	1956
12	Alan Scott (New Earth)	DC	1940
13	Barbara Gordon (New Earth)	DC	1967
14	Jason Garrick (New Earth)	DC	1940

Brainstorm:
what would the pseudocode look like?

```
ggplot(data = <DATA>) +
  <GEOM_FUNCTION>(mapping = aes(<MAPPINGS>))
```


New Comic Book Characters Introduced Per Year

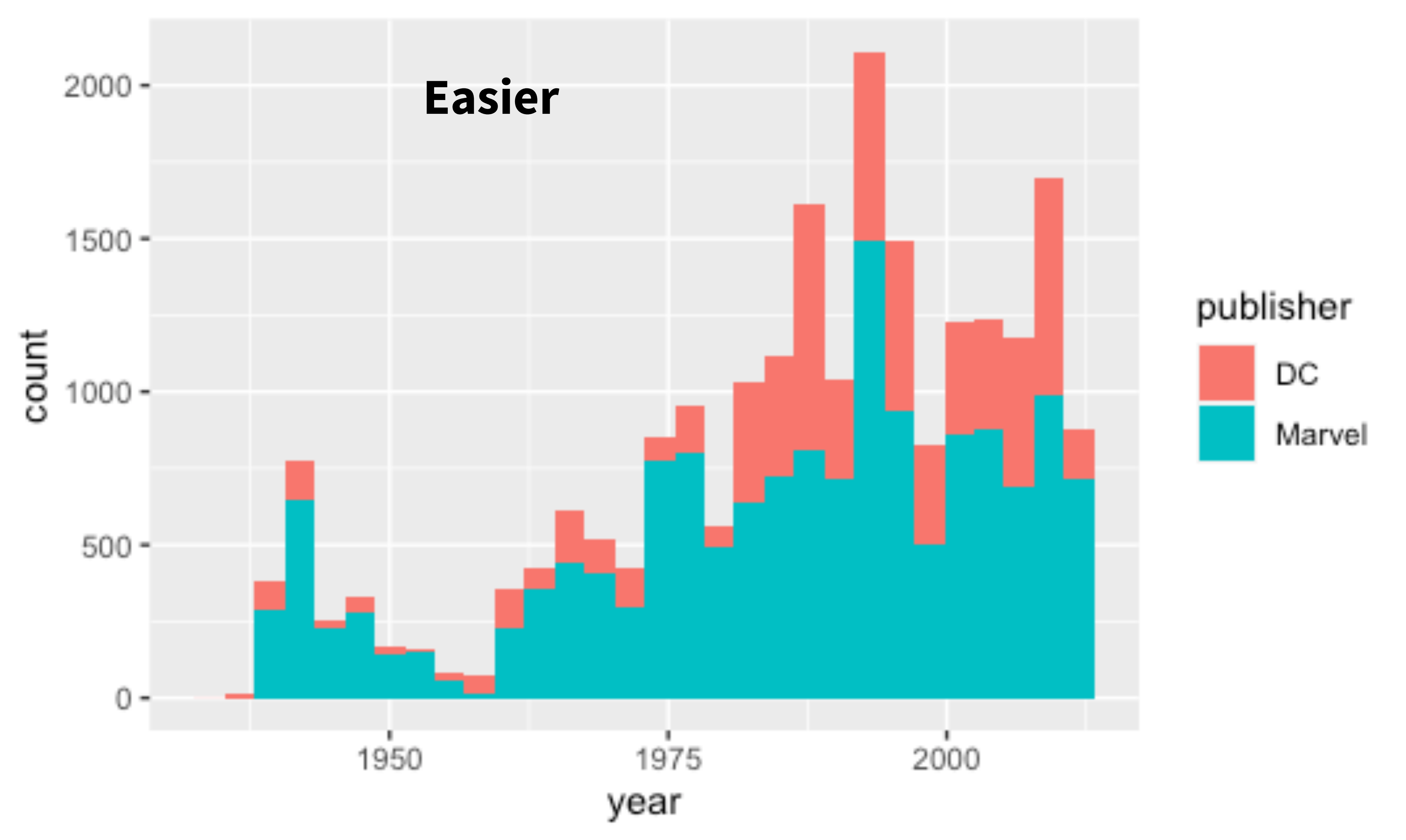
SOURCE: DC, MARVEL WIKIAS

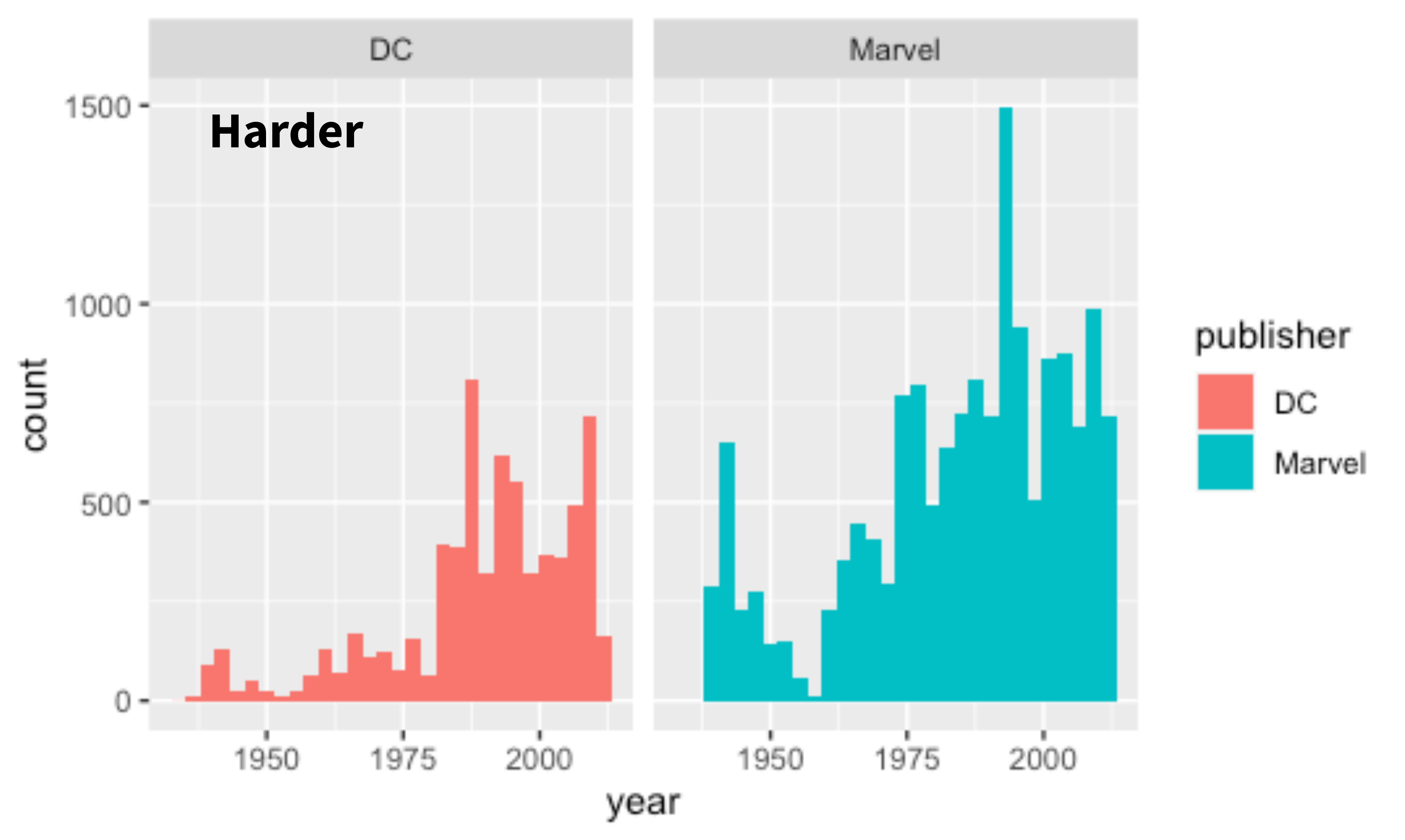


	name	publisher	year
1	Batman (Bruce Wayne)	DC	1939
2	Superman (Clark Kent)	DC	1986
3	Green Lantern (Hal Jordan)	DC	1959
4	James Gordon (New Earth)	DC	1987
5	Richard Grayson (New Earth)	DC	1940
6	Wonder Woman (Diana Prince)	DC	1941
7	Aquaman (Arthur Curry)	DC	1941
8	Timothy Drake (New Earth)	DC	1989
9	Dinah Laurel Lance (New Earth)	DC	1969
10	Flash (Barry Allen)	DC	1956
11	GenderTest	DC	1956
12	Alan Scott (New Earth)	DC	1940
13	Barbara Gordon (New Earth)	DC	1967
14	Jason Garrick (New Earth)	DC	1940

Your Turn 2

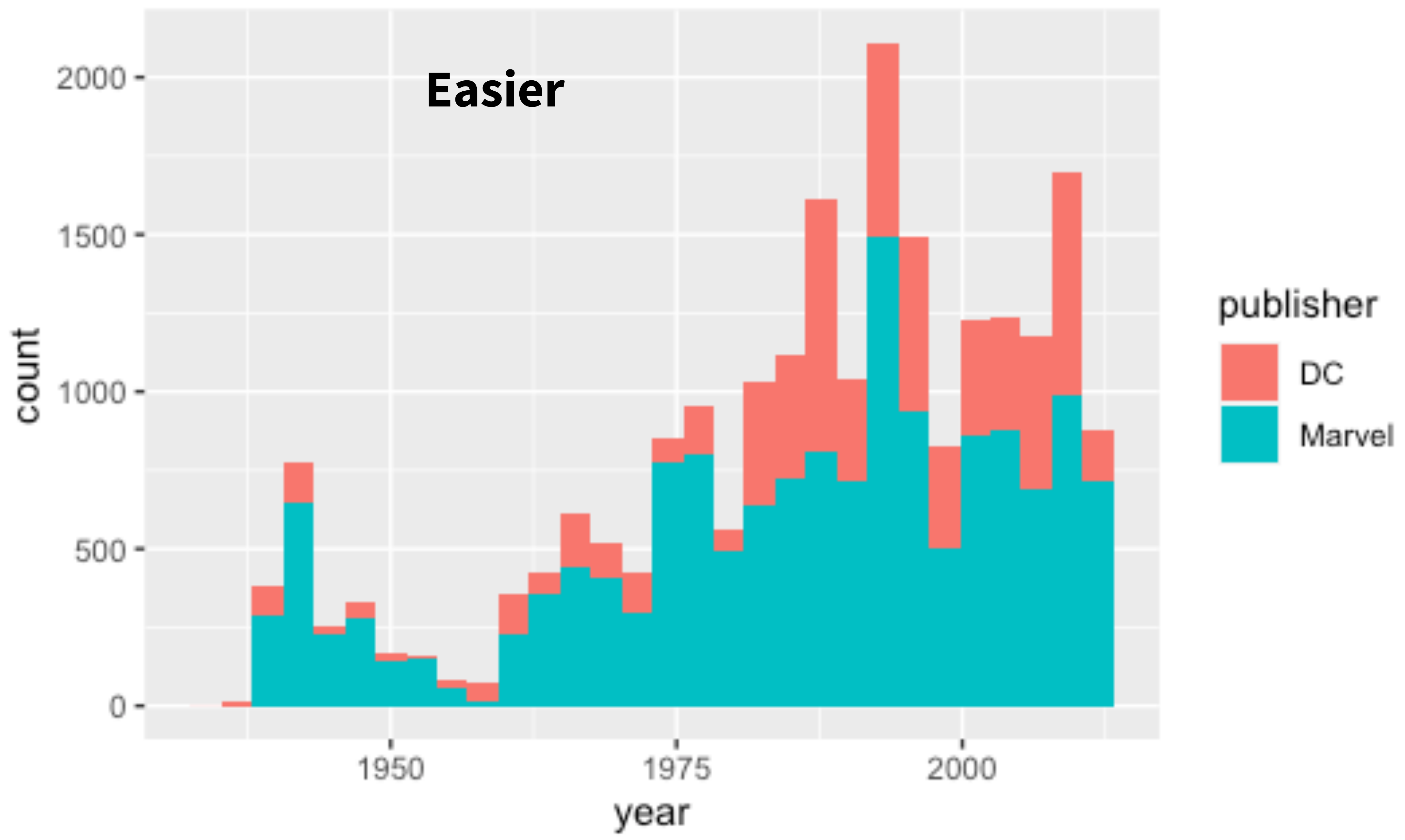
Try to make a basic visualization like the FiveThirtyEight graphic.



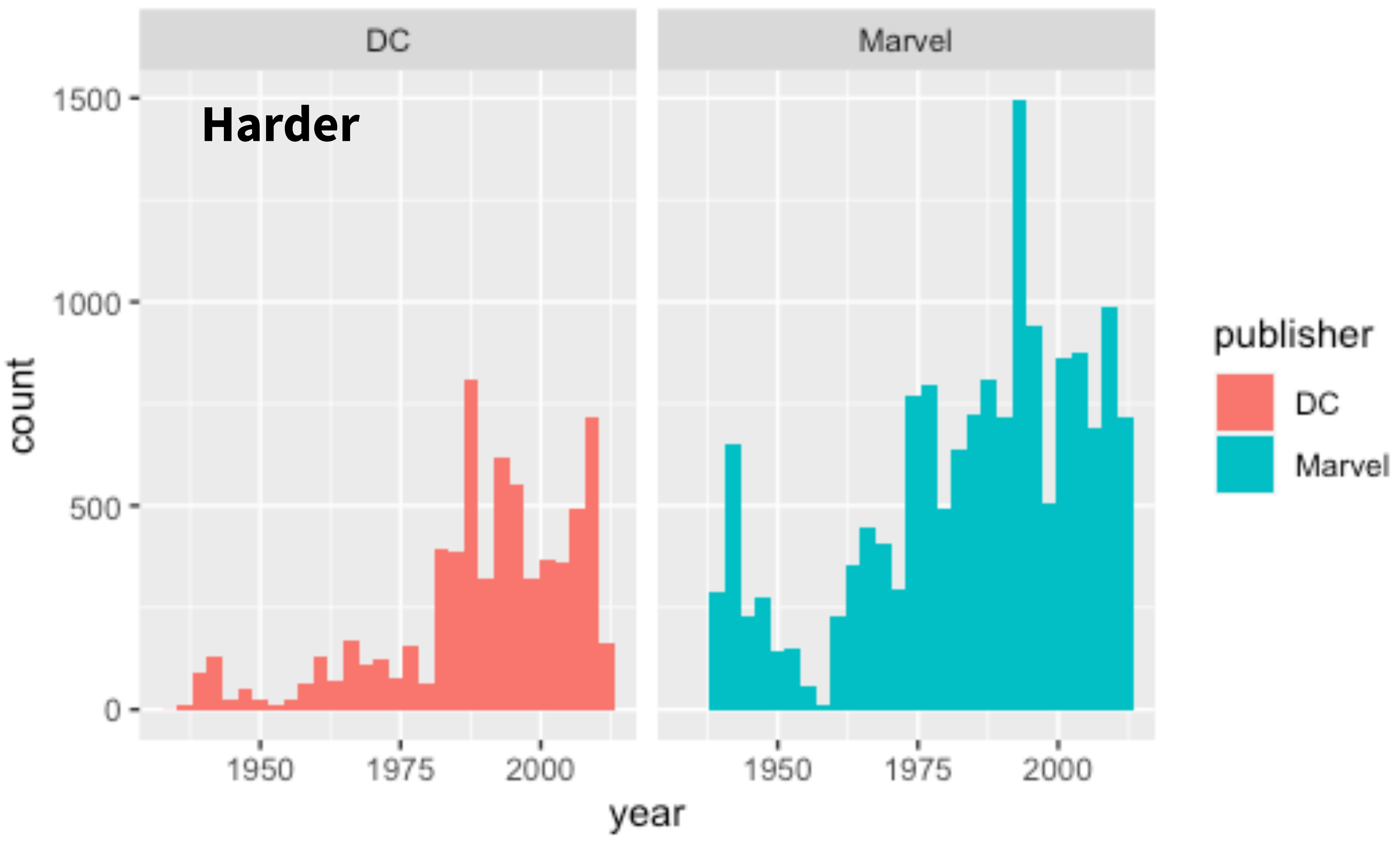


```
ggplot(comic_characters) +
     geom_histogram(aes(x = year, fill = publisher))
```

```
ggplot(comic_characters) +
     geom_histogram(aes(x = year, fill = publisher)) +
     facet_wrap(~publisher)
```



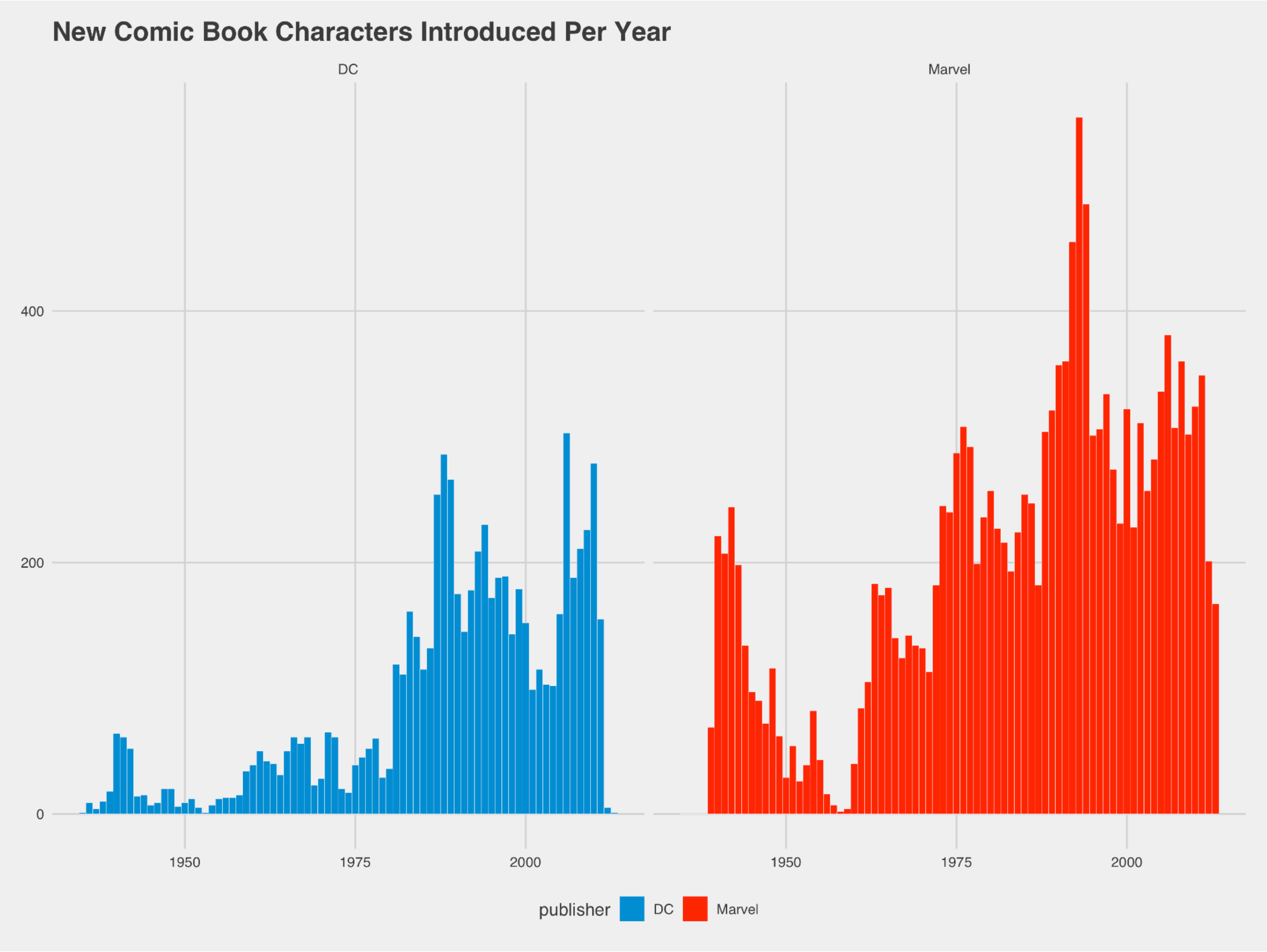
Easier



Harder

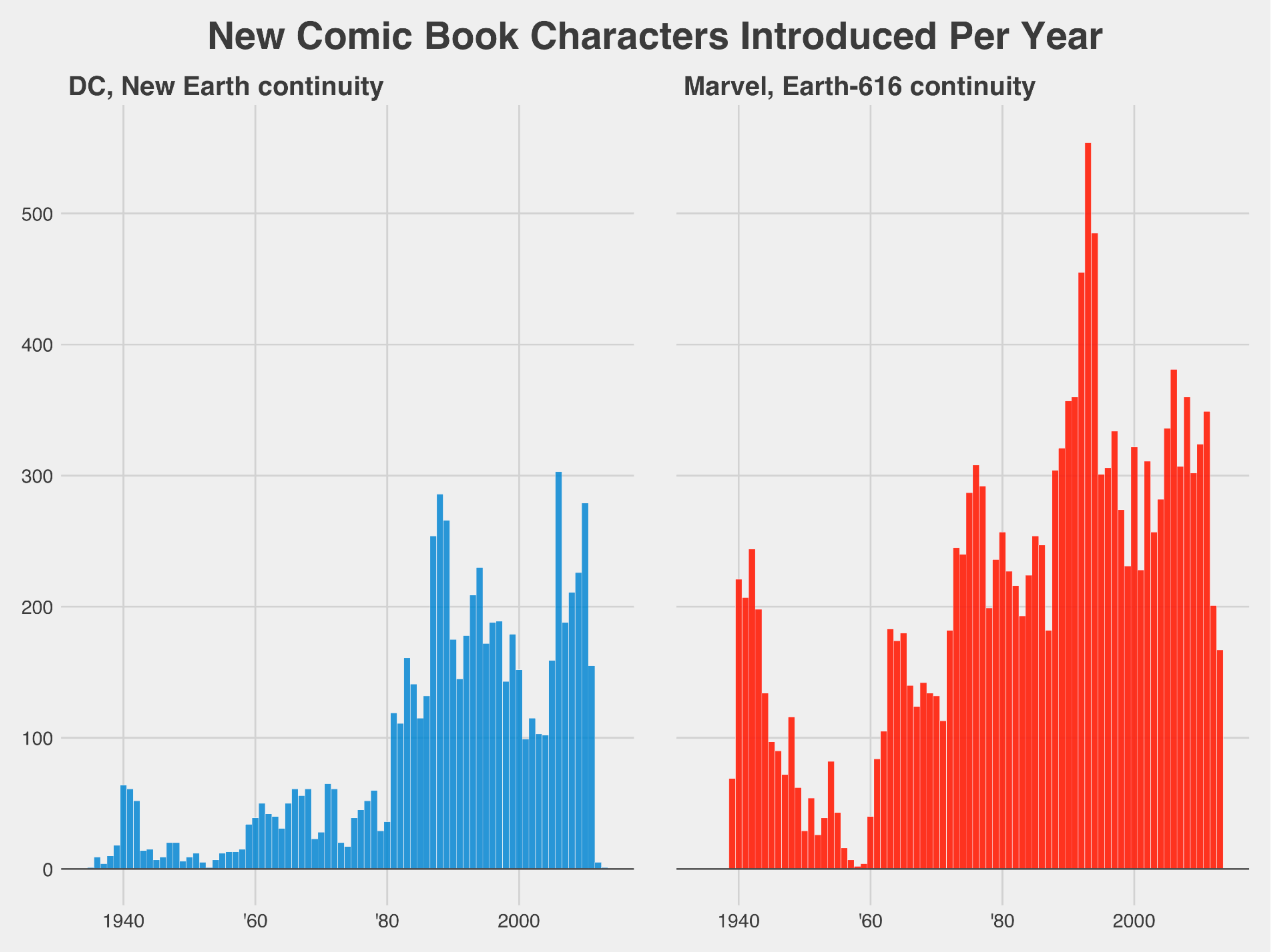
Controlling the details

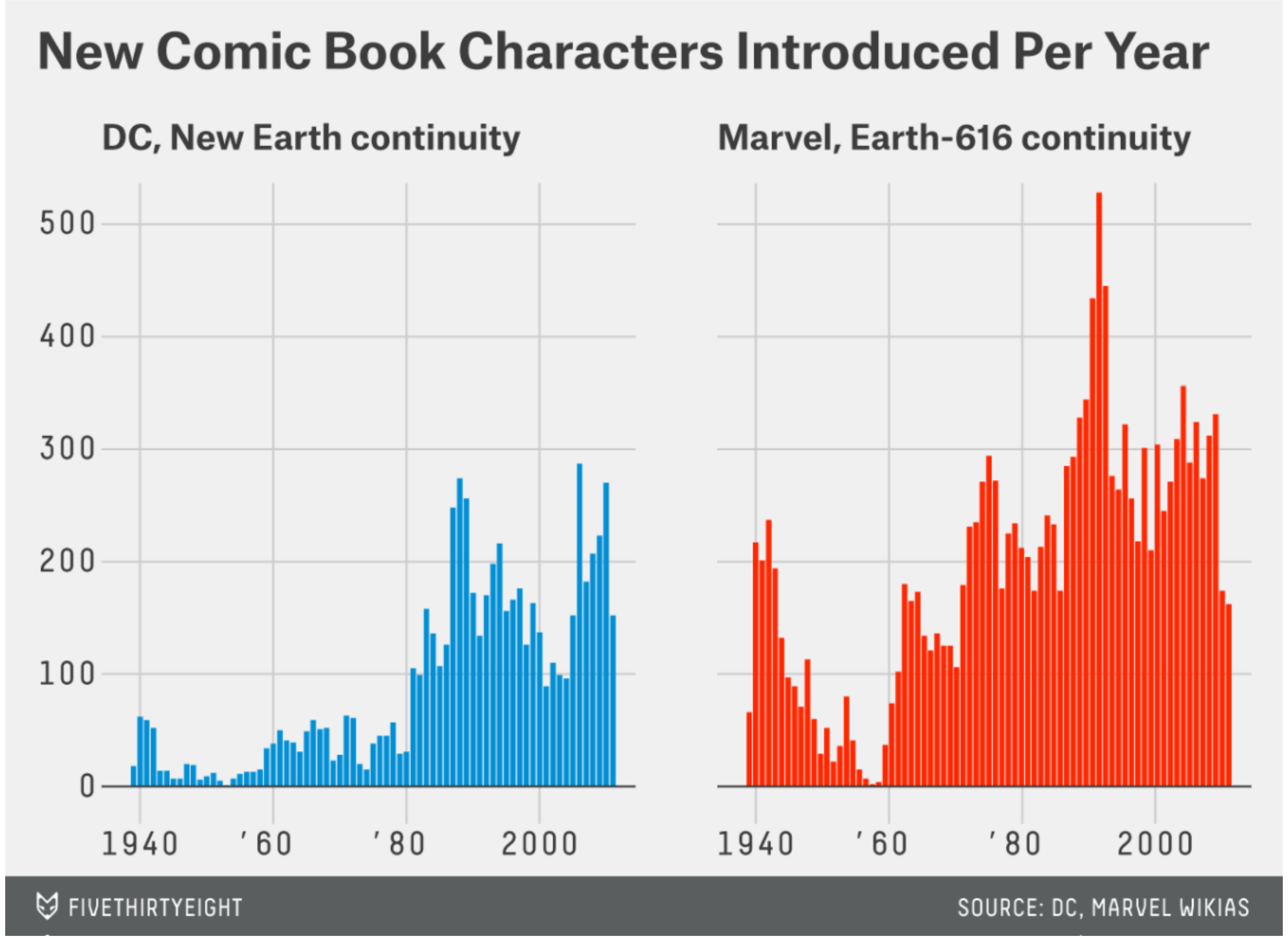
```
p1 <- ggplot(comic_characters) +
  geom_histogram(aes(x = year, fill = publisher),
                 binwidth = 1, color = "white", lwd = 0.1) +
  facet_wrap(~publisher) +
  theme_fivethirtyeight() +
  scale_fill_manual(values = c("#008fd5", "#ff2700")) +
  labs(title = "New Comic Book Characters Introduced Per Year")
p1
```



```
publisher_labels <- c(DC = "DC, New Earth continuity",
                      Marvel = "Marvel, Earth-616 continuity")
p1 <- ggplot(comic_characters) +
  geom_histogram(aes(x = year, fill = publisher), binwidth = 1,
                 color = "white", lwd = 0.1,
                 show.legend = FALSE, alpha = 0.9) +
  facet_wrap(~publisher, labeller = labeller(publisher = publisher_labels)) +
  scale_x_continuous(breaks = seq(1940, 2000, 20),
                     labels = c("1940", "'60", "'80", "2000")) +
  scale_y_continuous(limits = c(0, 555), breaks = seq(0, 500, 100)) +
  scale_fill_manual(values = c("#008fd5", "#ff2700"))+
  geom_hline(yintercept = 0, color = "grey31", size = 0.5) +
  theme_fivethirtyeight() +
  theme(axis.text.y = element_text(size = 13),
        axis.text.x = element_text(size = 13),
        plot.title = element_text(size = 26, hjust = 0.5),
        strip.text.x = element_text(size = 18, hjust = 0, face ="bold"),
        panel.spacing = unit(2, "lines"))  +
  labs(title = "New Comic Book Characters Introduced Per Year")
p1
```

Better, but still not quite!





Ugly plot

Today’s lab

I’d like you to experiment with themes, scales, position adjustments and other controllable elements.

Take a plot you’ve made (either today’s comic characters or maybe a basic viz) and adjust at least 5 things on it.

I’m borrowing this idea from Allison Horst.

One example: let them do their (data viz) worst

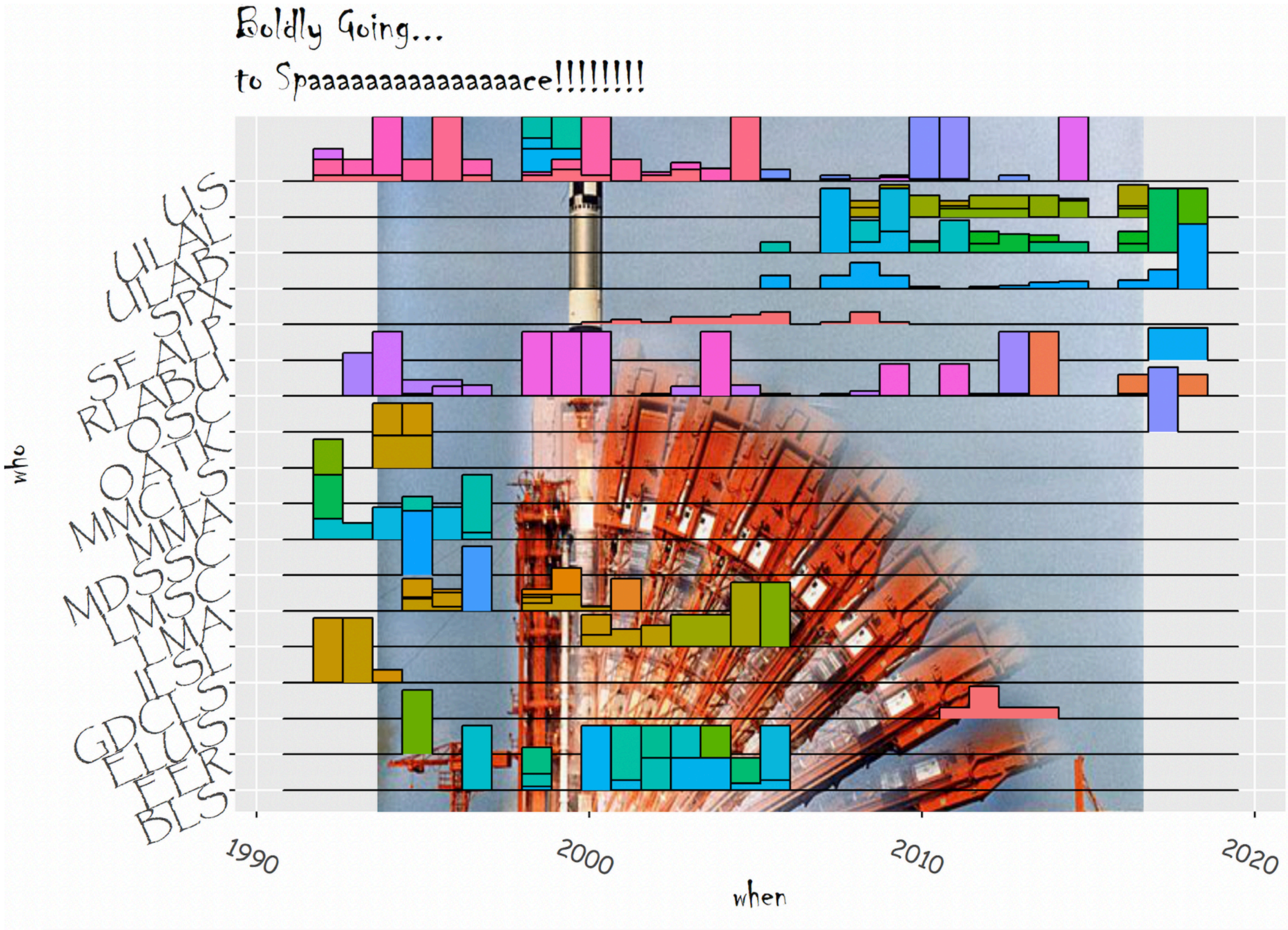
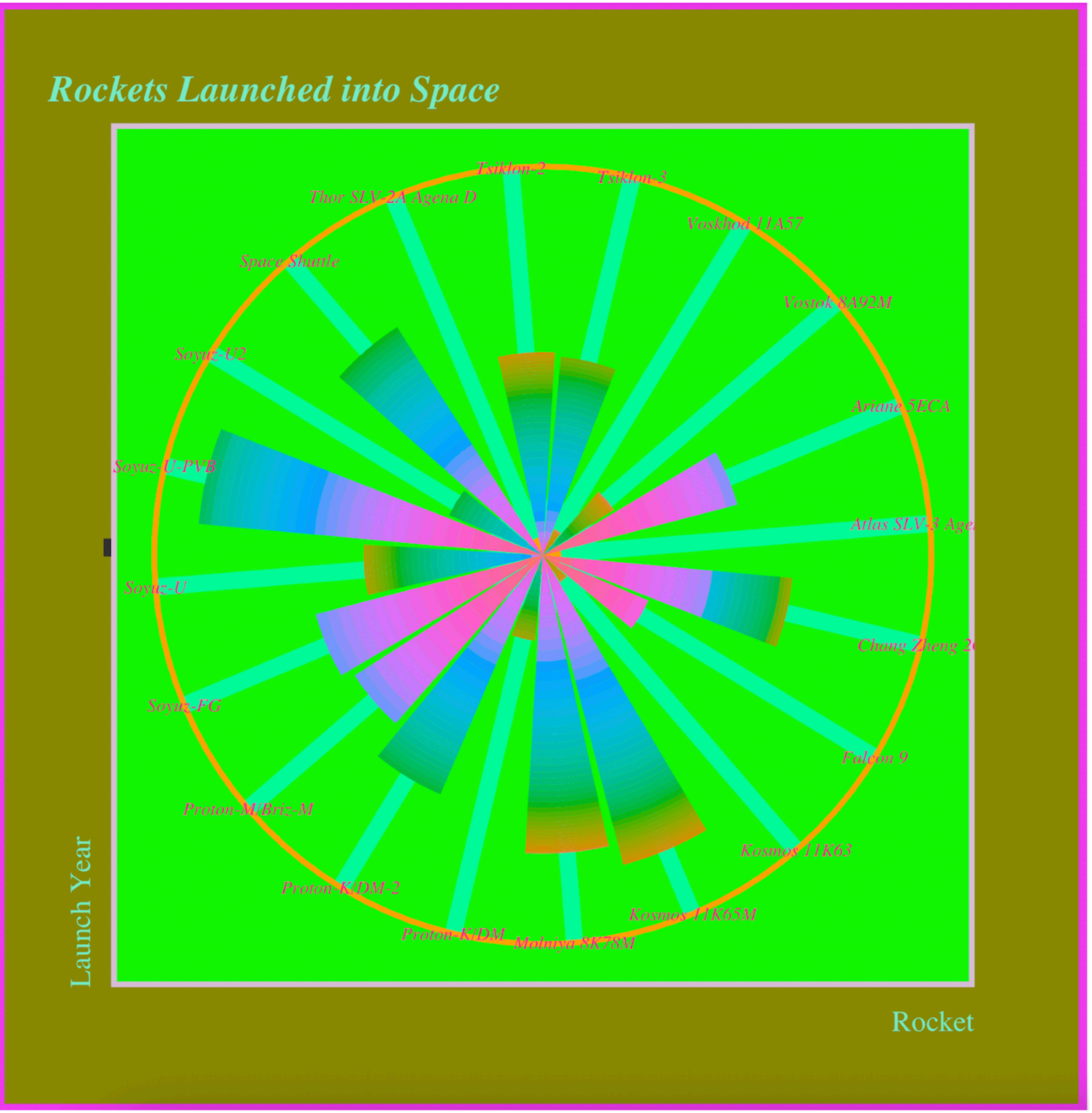


Photo: Gemini 10 launch, NASA



```
ggplot(launch, aes(x = type, y = launch_year)) +
  geom_bar(aes(fill = launch_year,
               size = total),
               stat="identity",
               show.legend = FALSE) +
  labs(title = "Rockets Launched into Space",
       x = "Rocket",
       y = "Launch Year") +
  theme(plot.background = element_rect(fill = "yellow4",
                                        colour = "magenta2",
                                        size = 3),
       panel.background = element_rect(fill = "green",
                                        color = "thistle",
                                        size = 2),
       panel.grid.major = element_line(color = "mediumspringgreen",
                                        size = 3),
       panel.grid.major.y = element_line(color = "orange",
                                        size = 1),
       text = element_text(family = "serif",
                           color = "aquamarine2"),
       axis.text = element_text(color = "deeppink",
                                hjust = -1,
                                face = "italic",
                                size = 7),
       axis.text.y = element_blank(),
       plot.margin = margin(t = 1, r = 1.5, b = 1, l = 1, unit = "cm"),
       plot.title = element_text(size=14,
                                face="bold.italic",
                                hjust = -.15),
       axis.title.x = element_text(hjust= 1),
       axis.title.y = element_text(hjust = 0)
       ) +
  coord_polar(start = 45)
```