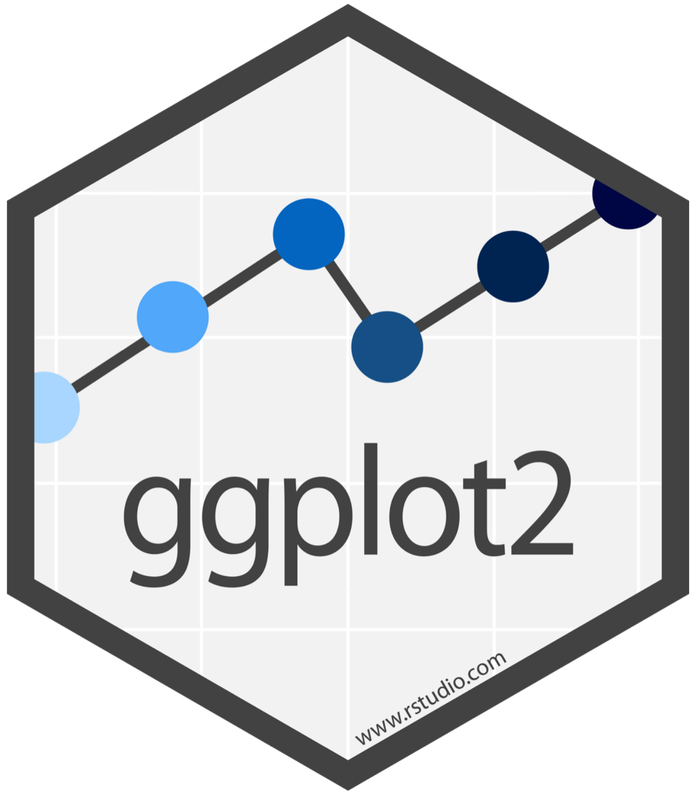
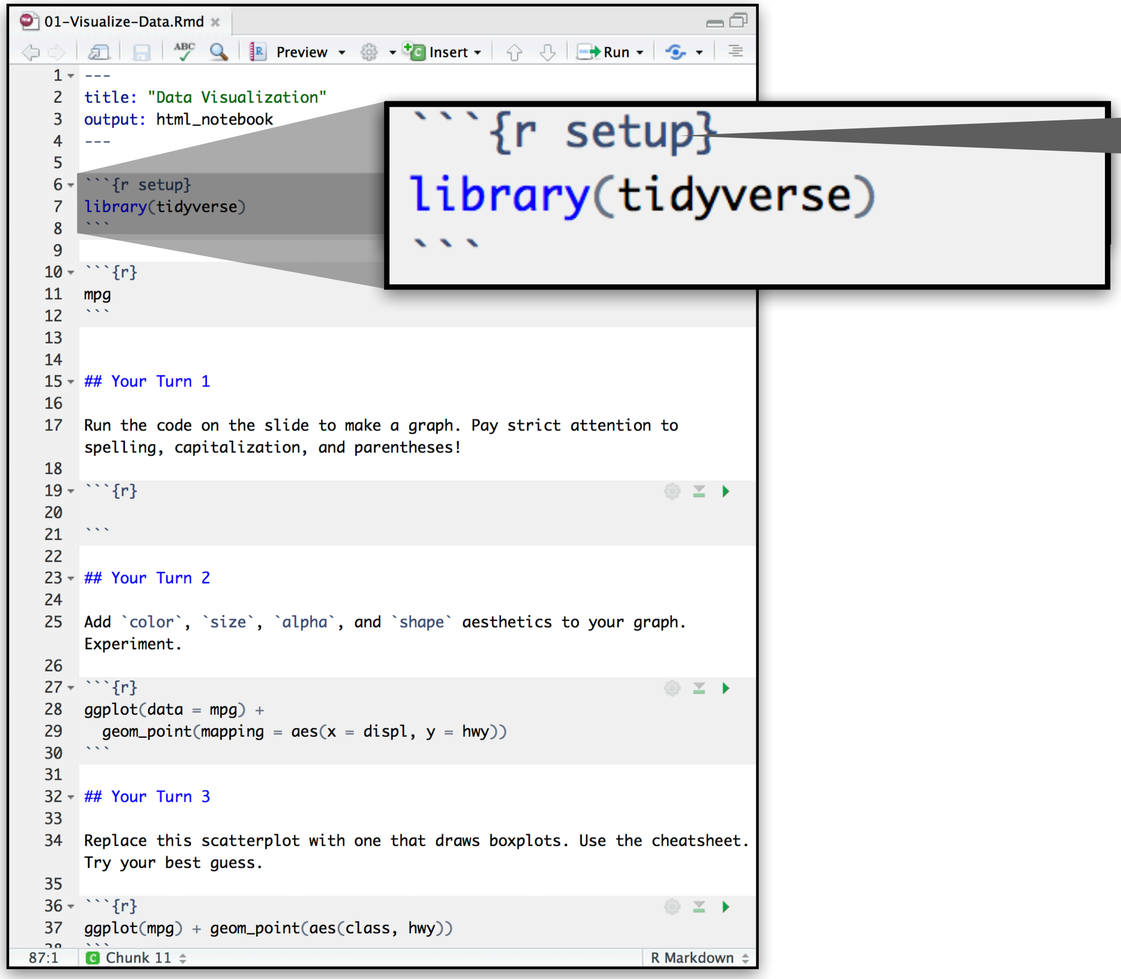
Visualize Data with



Setup

The setup chunk is always run once before anything else



chunk labels are optional, the setup label is special

income

Reported employment income of graduates (2010, 2011) two years and five years after graduation

```
income
```

http://github.com/paleolimbot/unienrollment

Quiz

Confer with your group.

What relationship do you expect to see between how much a graduate makes after 2 years (income_2yr) and 5 years (income_5yr)?

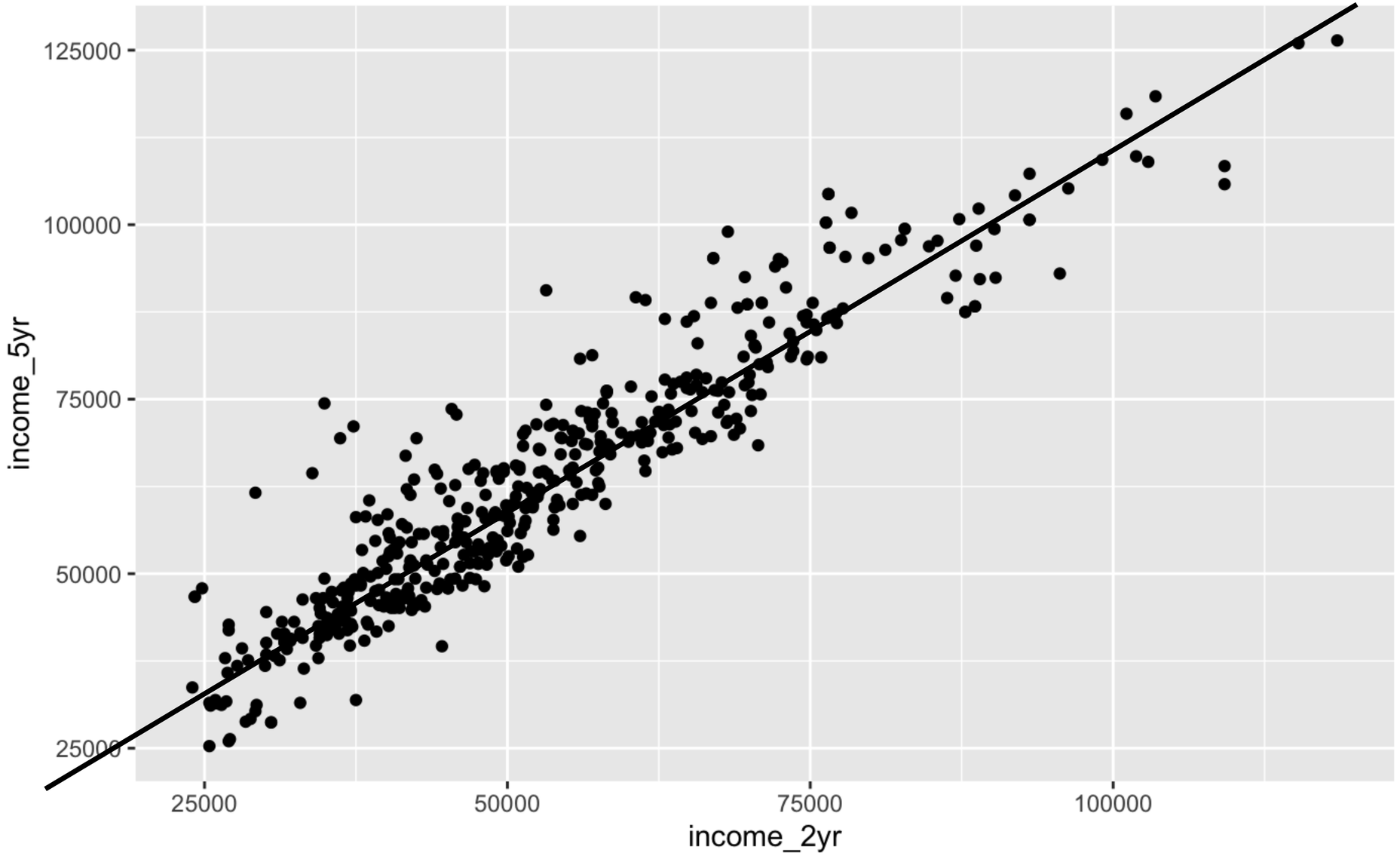
No peeking ahead!

Your Turn 1

Run this code in your notebook to make a graph.

Pay strict attention to spelling, capitalization, and parentheses!

```
ggplot(data = income) +
  geom_point(mapping = aes(x = income_5yr, y = income_2yr))
```



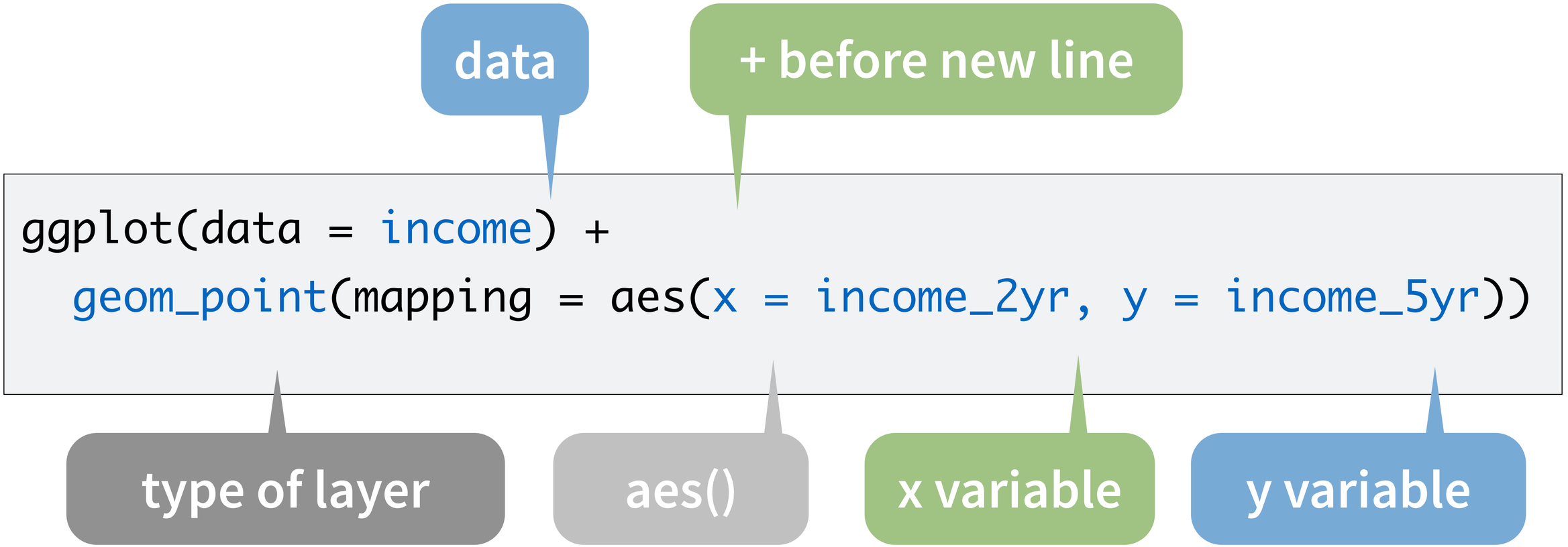

```
ggplot(data = income) +
  geom_point(mapping = aes(x = income_2yr, y = income_5yr))
```

1. "Initialize" a plot with ggplot()
2. Add layers with geom_ functions

```
ggplot(data = income) +
  geom_point(mapping = aes(x = income_2yr, y = income_5yr))
```

Pro tip: Always put the + at the end of a line, Never at the start

```
ggplot(data = income) +
  geom_point(mapping = aes(x = income_2yr, y = income_5yr))
```


data
+ before new line
ggplot(data = income) +
geom_point(mapping = aes(x = income_2yr, y = income_5yr))
type of layer
aes()
x variable
y variable

A template

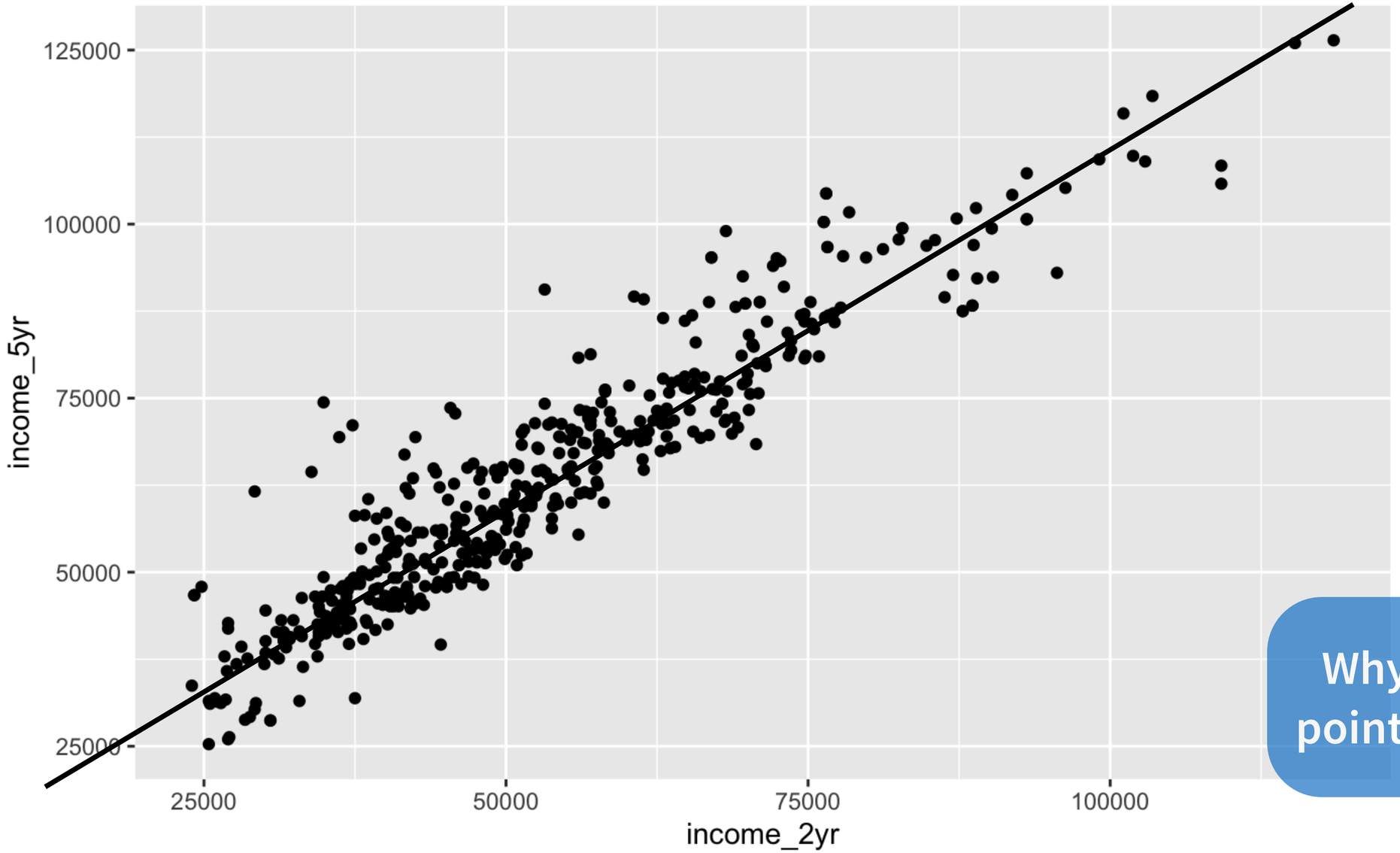
```
ggplot(data = <DATA>) +
  <GEOM_FUNCTION>(mapping = aes(<MAPPINGS>))
```

```
  geom_point(mapping = aes(x = income_2yr, y = income_5yr))
```

A template

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

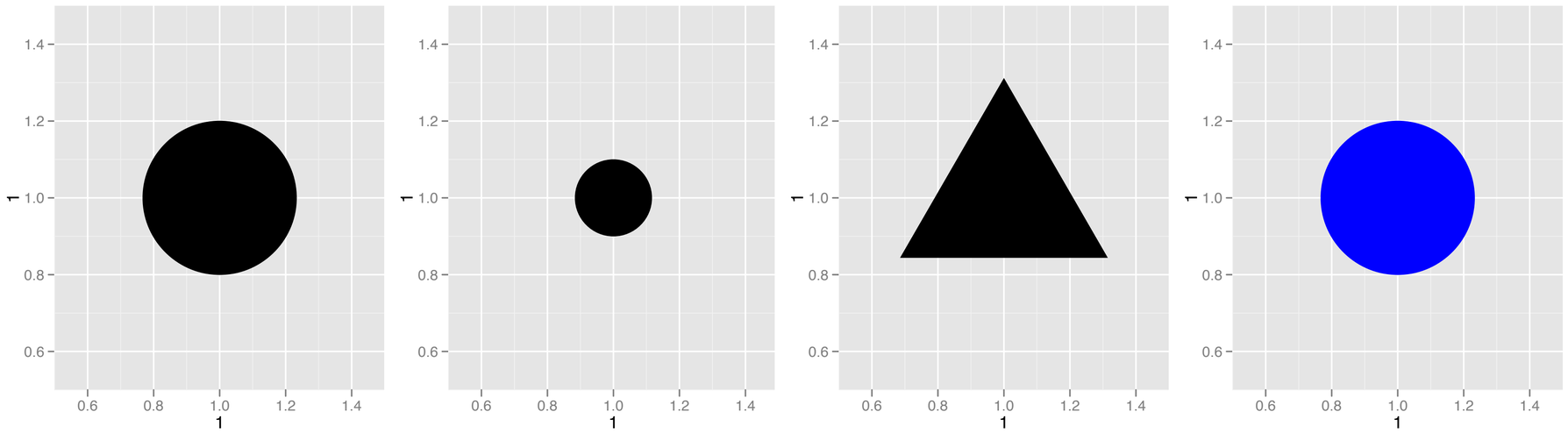
Mappings

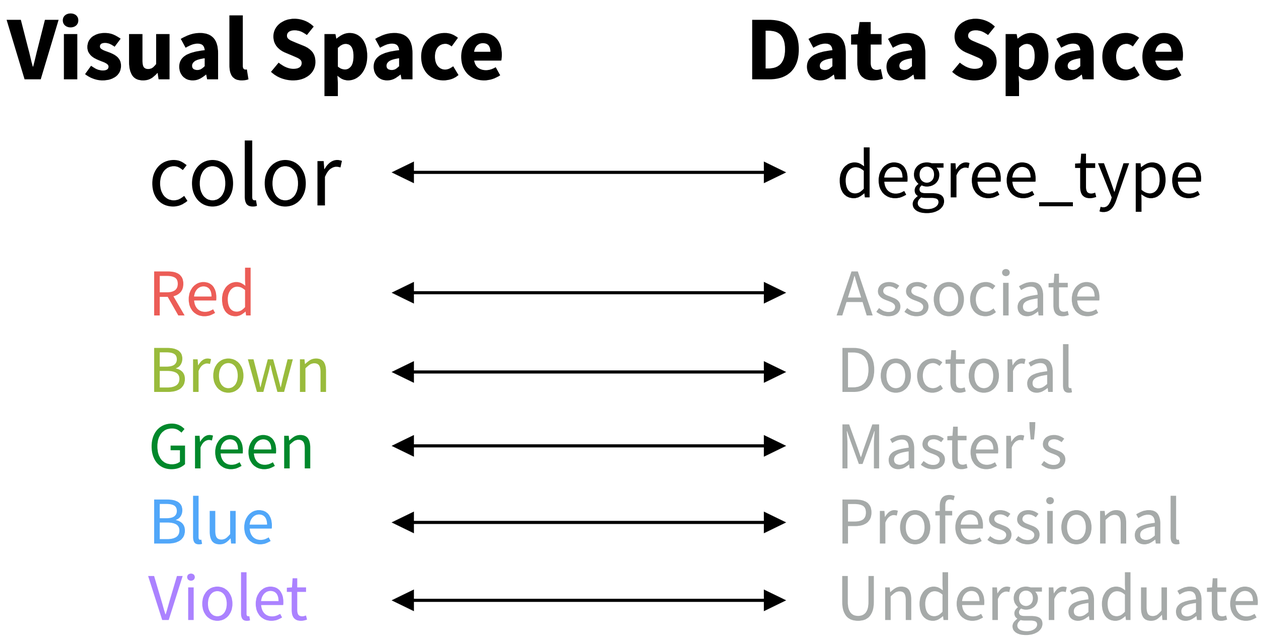


Why are some points different?

```
ggplot(data = income) +
  geom_point(mapping = aes(x = income_2yr, y = income_5yr))
```

Aesthetics



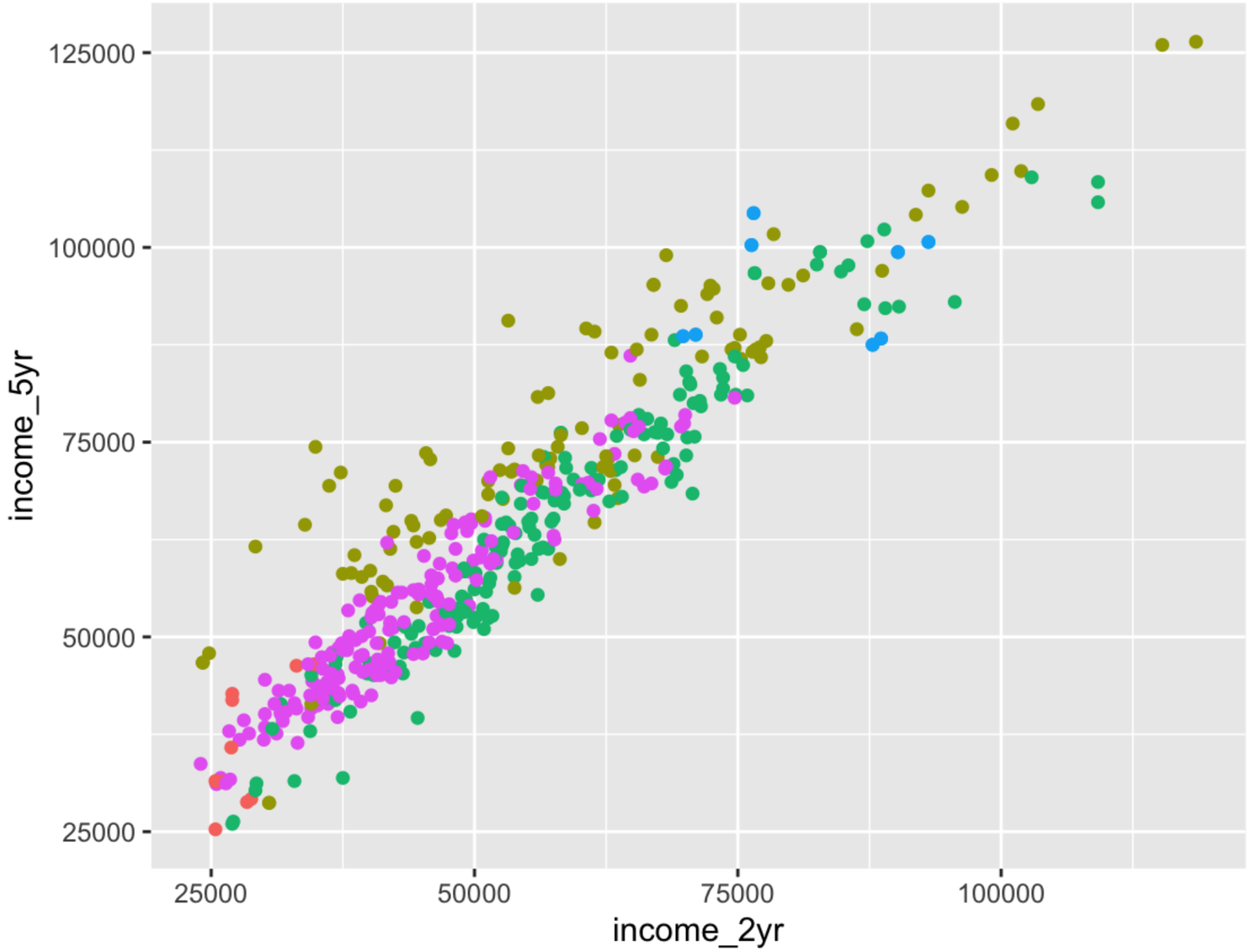

Visual Space
Data Space
color
degree_type
Red
Associate
Brown
Doctoral
Green
Master's
Blue
Professional
Violet
Undergraduate

Aesthetics

aesthetic property

Variable to map it to

```
ggplot(income) + geom_point(aes(x = income_2yr, y = income_5yr, color = degree_type))
ggplot(income) + geom_point(aes(x = income_2yr, y = income_5yr, size = degree_type))
ggplot(income) + geom_point(aes(x = income_2yr, y = income_5yr, shape = degree_type))
ggplot(income) + geom_point(aes(x = income_2yr, y = income_5yr, alpha = degree_type))
```



Legend added automatically

degree_type
- Associate
- Doctoral
- Master's
- Professional
- Undergraduate

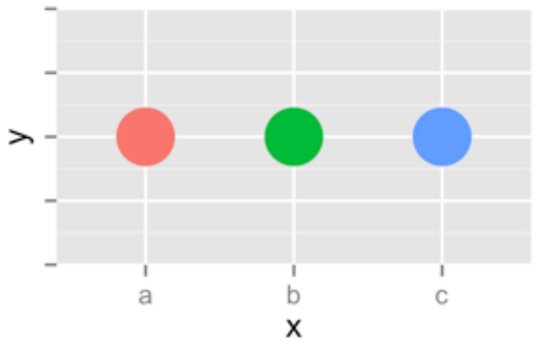
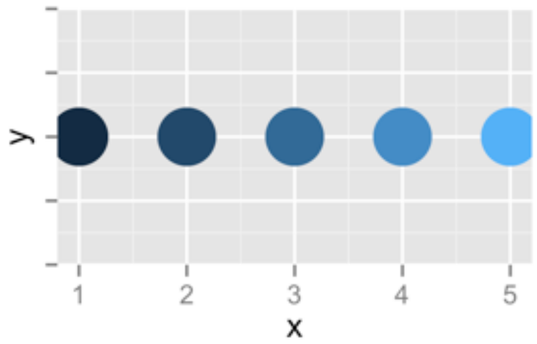
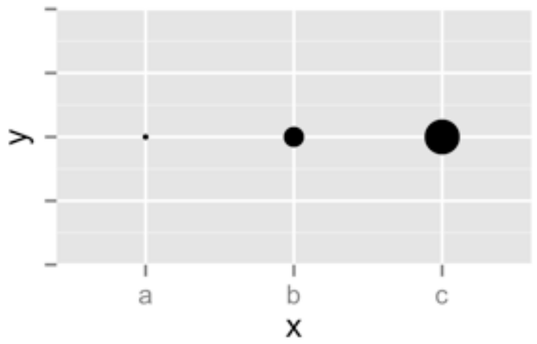
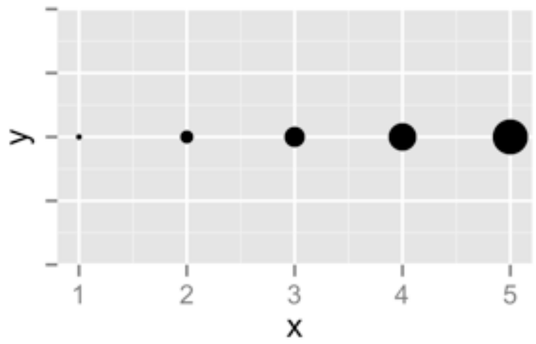
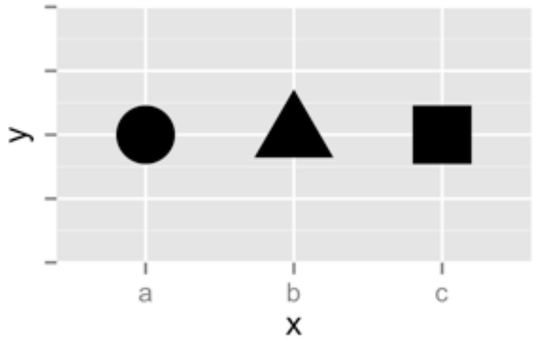
```
ggplot(data = income) +
  geom_point(mapping = aes(x = income_2yr, y = income_5yr, color = degree_type))
```

Your Turn 2

In the next chunk, add color, size, alpha, and shape aesthetics to your graph. Experiment.

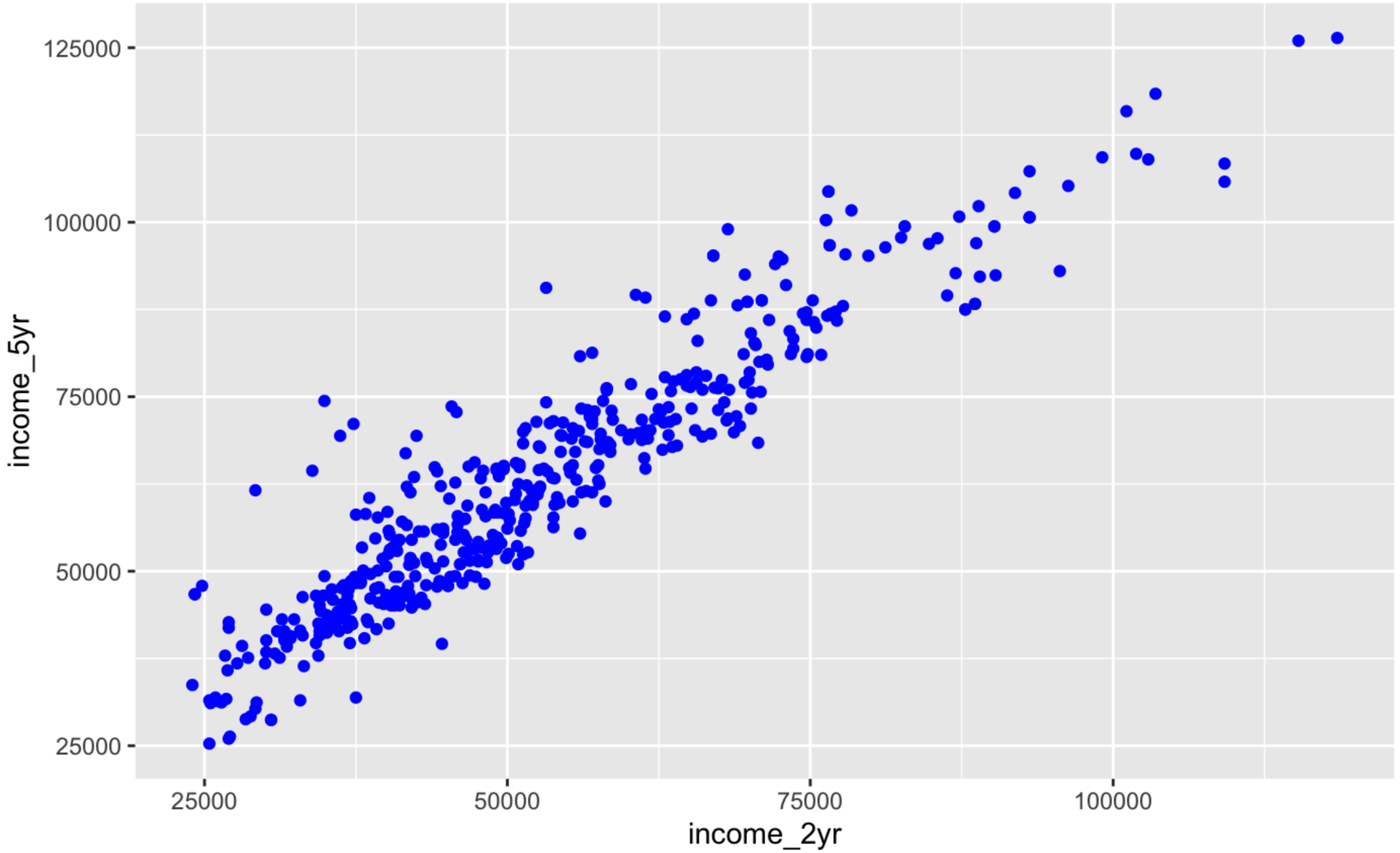
Do different things happen when you map aesthetics to discrete and continuous variables?

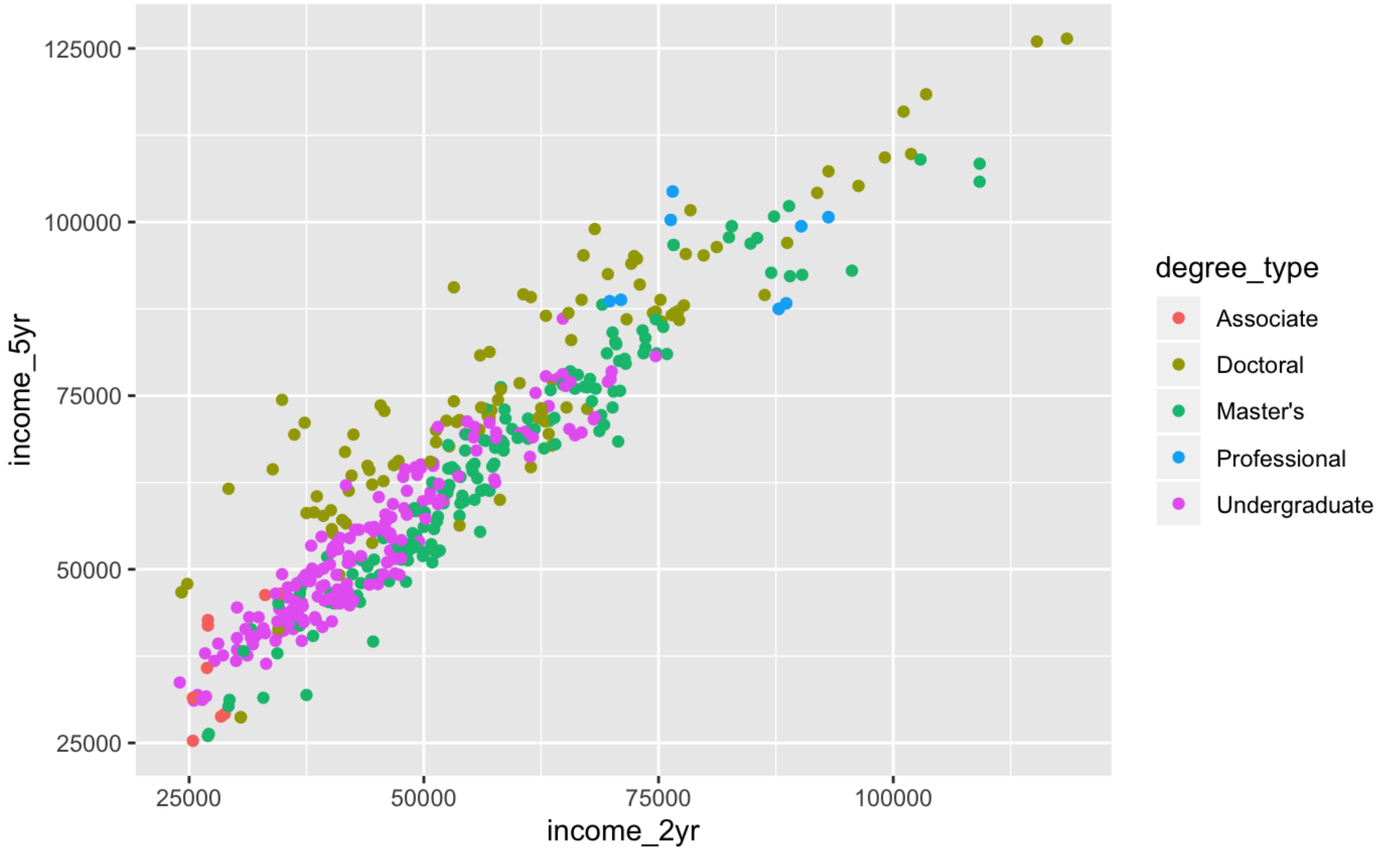
What happens when you use more than one aesthetic?

	Discrete	Continuous
Color		
Size		
Shape		

set vs. map

How would you make this plot?

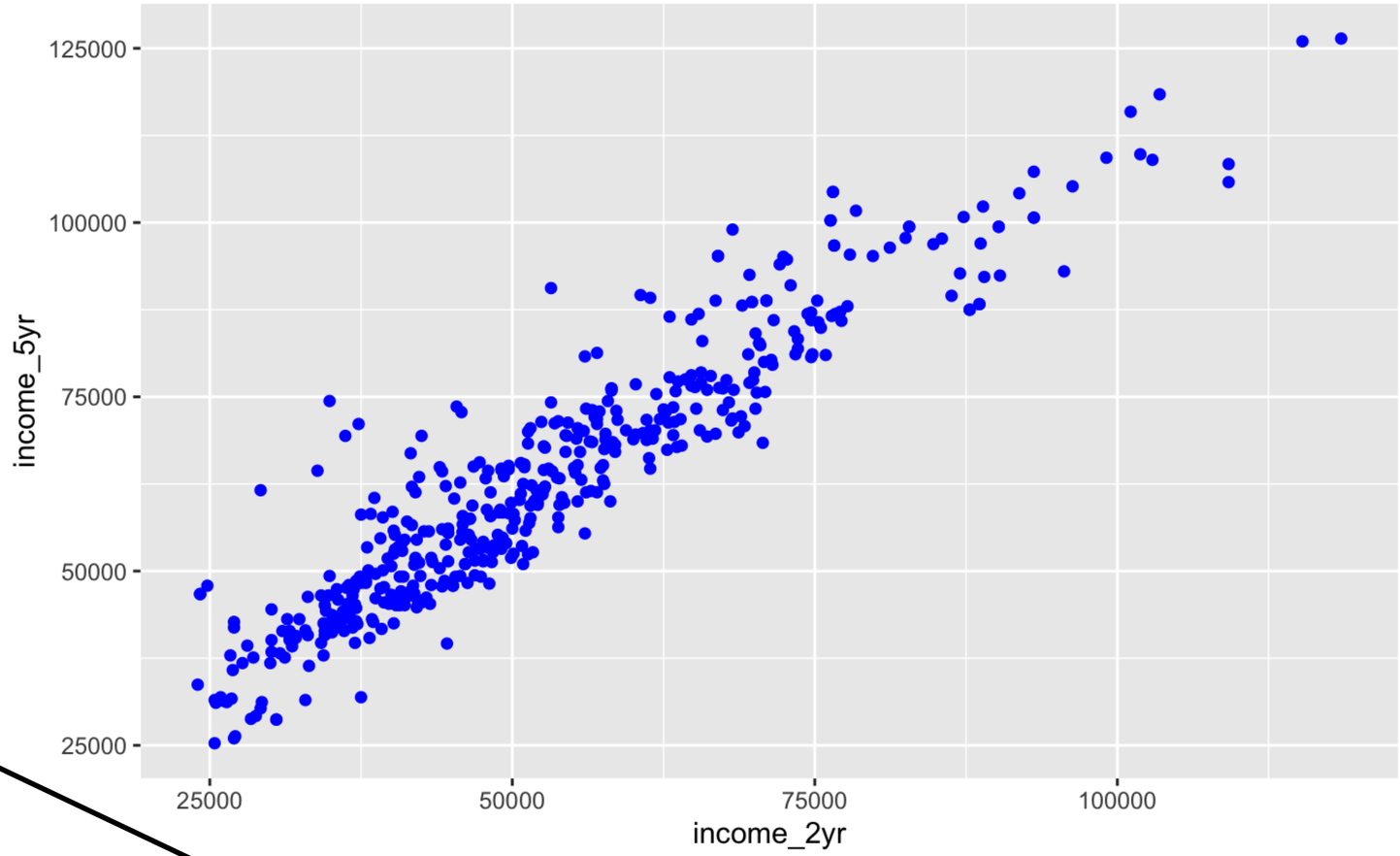




Inside of aes(): maps an aesthetic to a variable

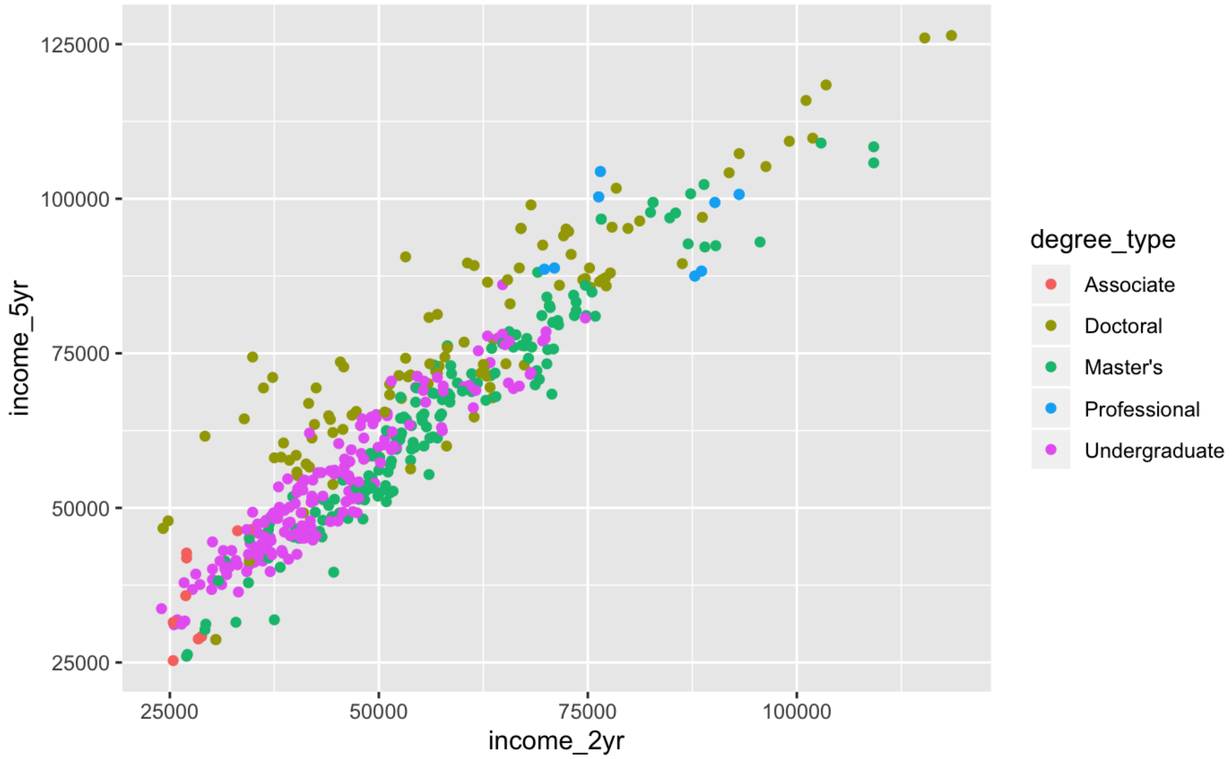
```
ggplot(income) + geom_point(aes(x = income_2yr, y = income_5yr, color = degree_type))
```

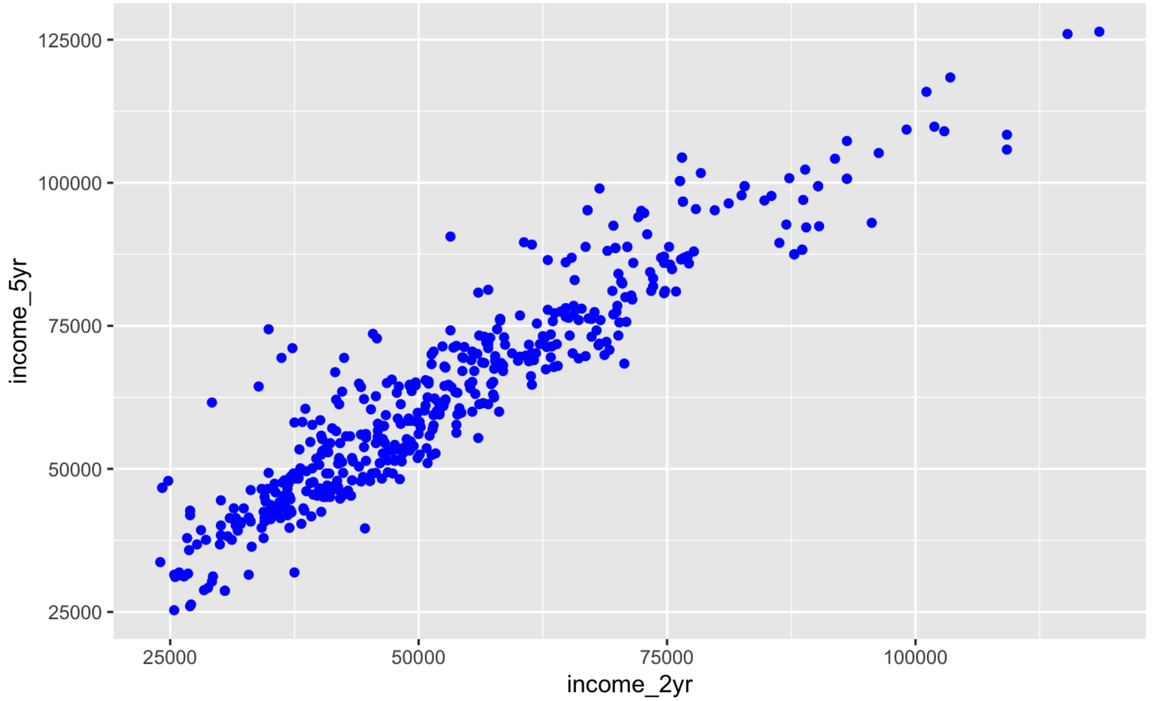
Outside of aes(): sets an aesthetic to a value




```
ggplot(income) + geom_point(aes(x = income_2yr, y = income_5yr, color = degree_type))
```

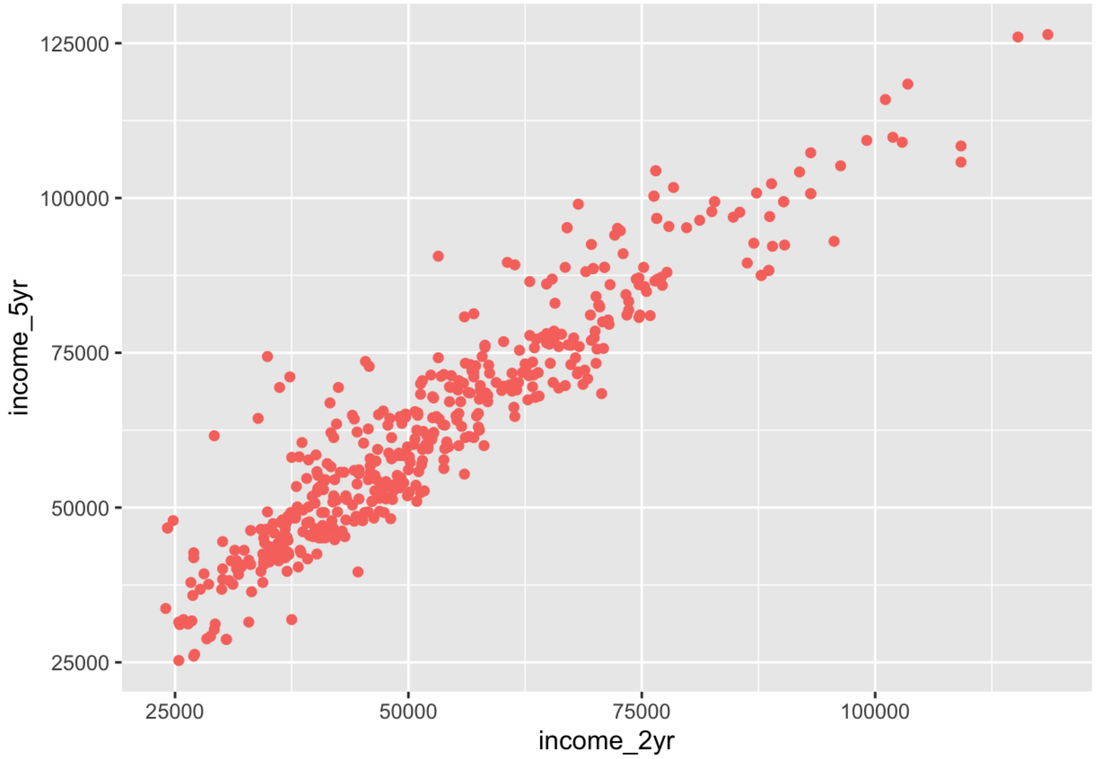
```
ggplot(income) + geom_point(aes(x = income_2yr, y = income_5yr), color = "blue")
```

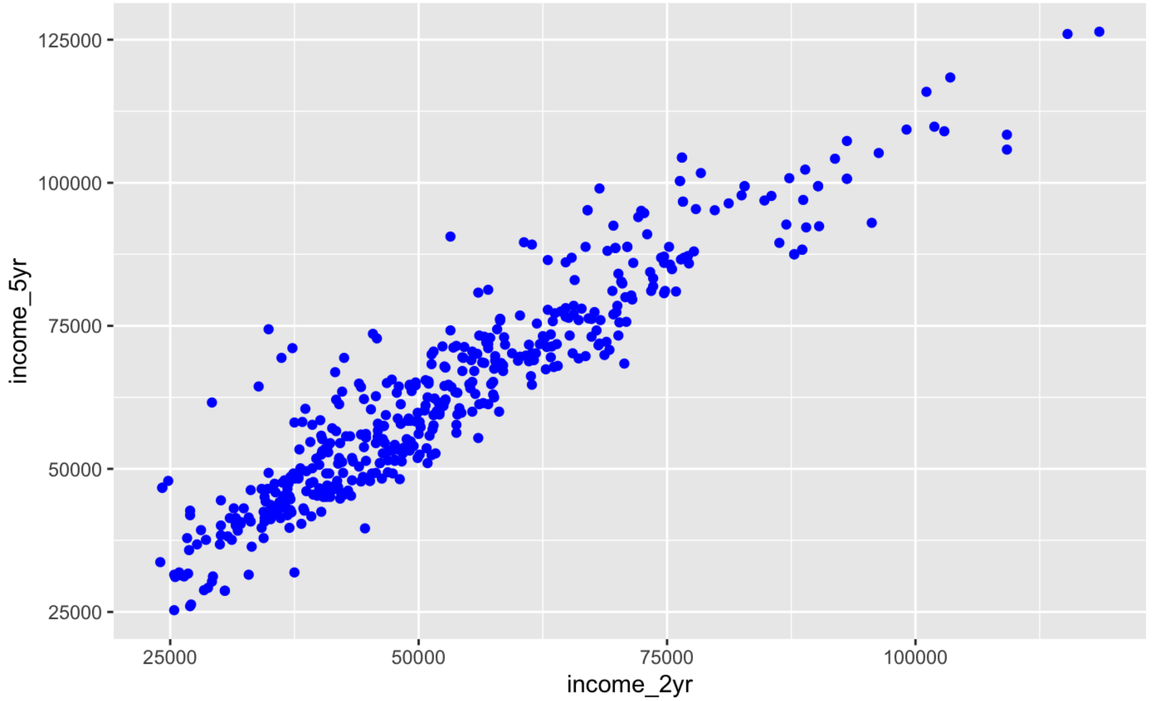





```
ggplot(income) + geom_point(aes(x = income_2yr, y = income_5yr, color = degree_type))
```

```
ggplot(income) + geom_point(aes(x = income_2yr, y = income_5yr), color = "blue")
```





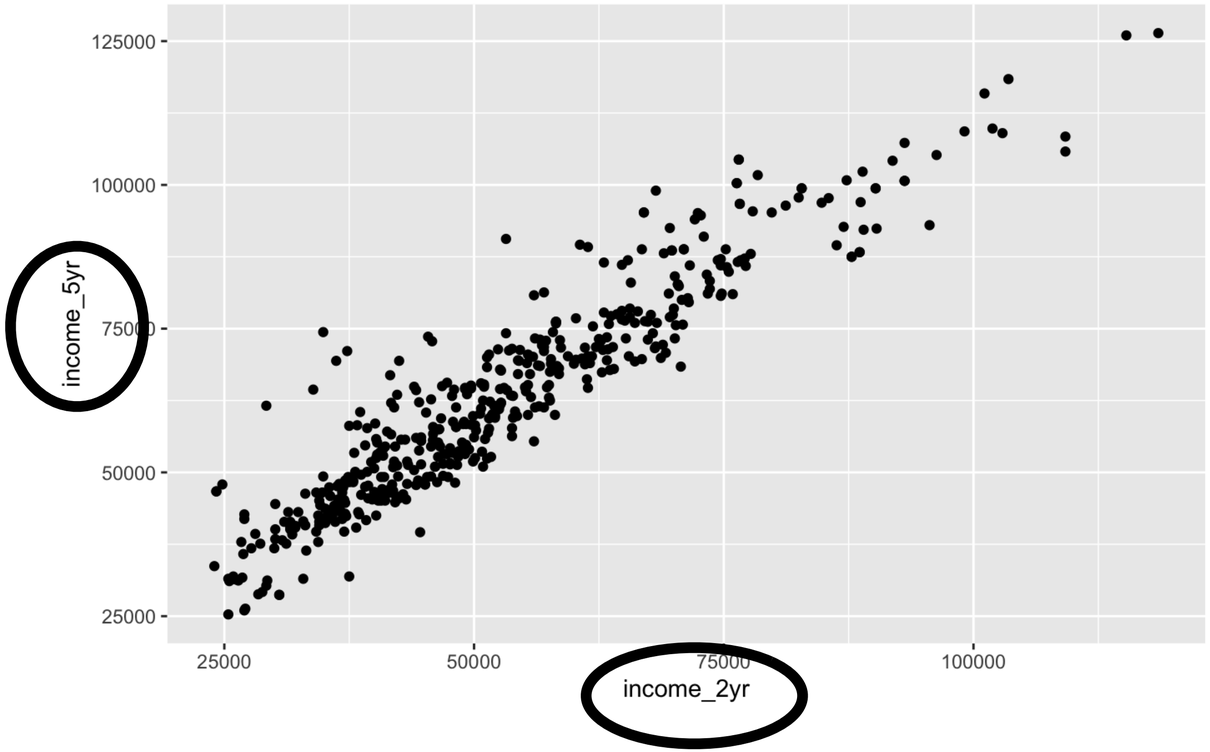

```
ggplot(income) + geom_point(aes(x = income_2yr, y = income_5yr, color = "blue"))
```

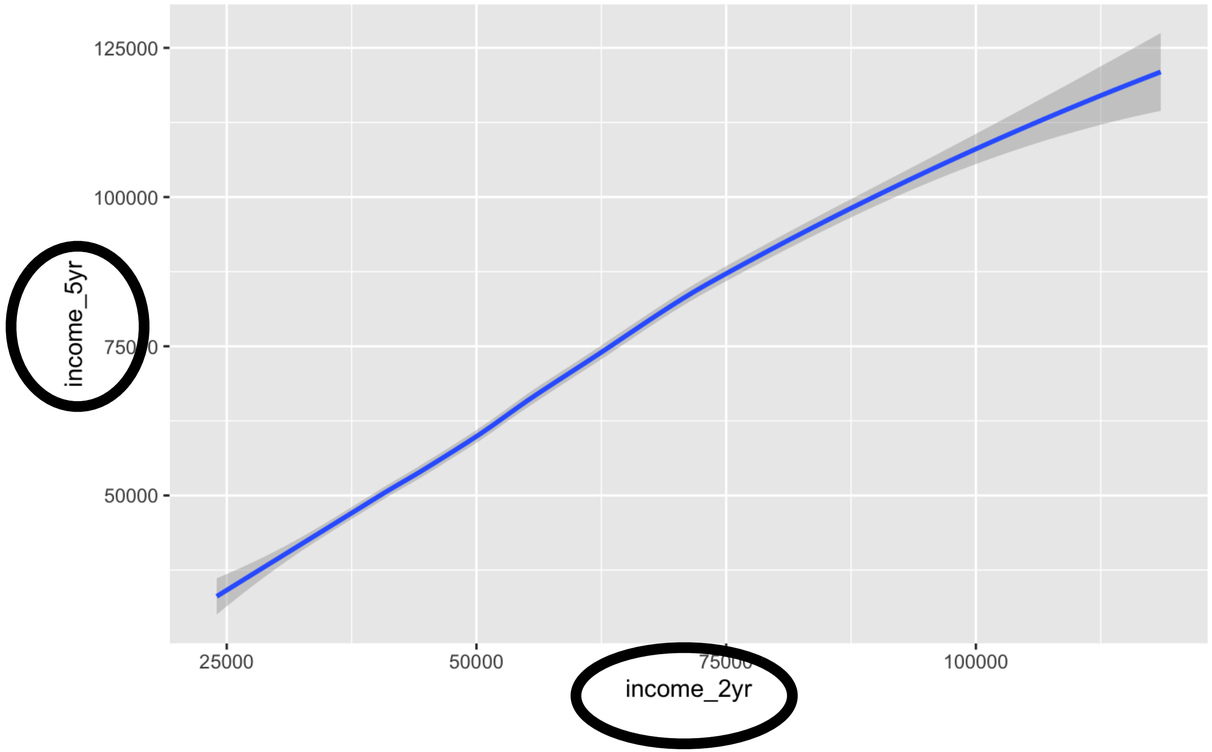
```
ggplot(income) + geom_point(aes(x = income_2yr, y = income_5yr), color = "blue")
```

Geoms

How are these plots similar?

Same: x var , y var , data

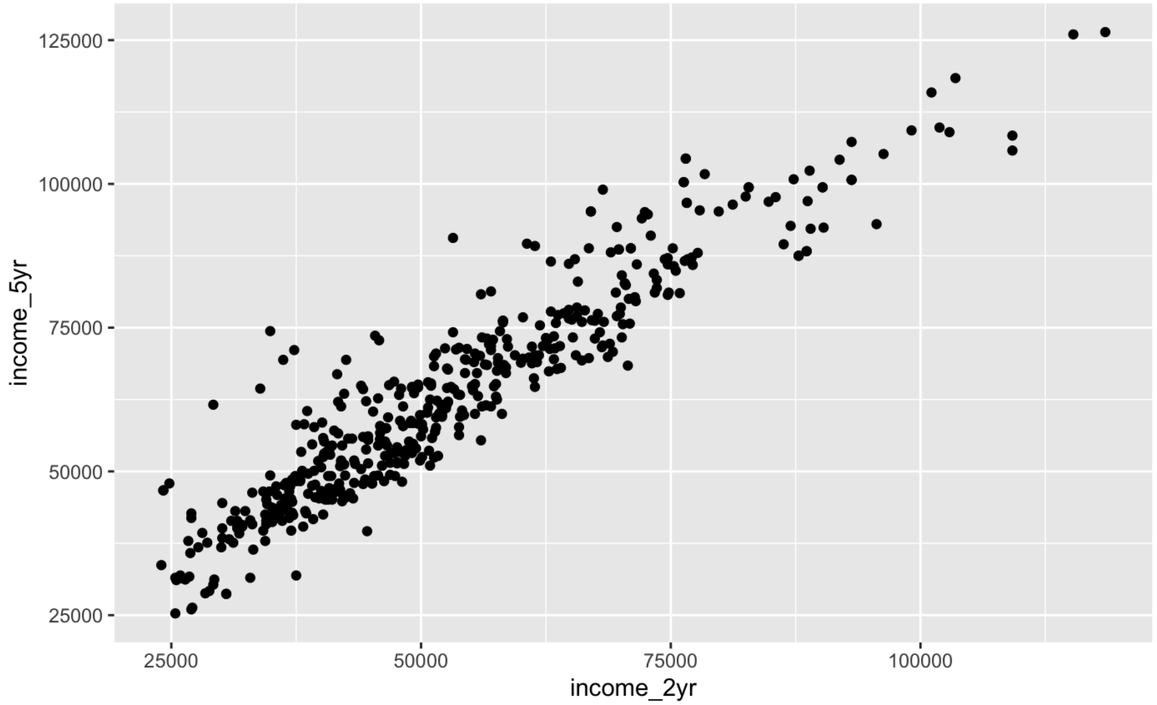


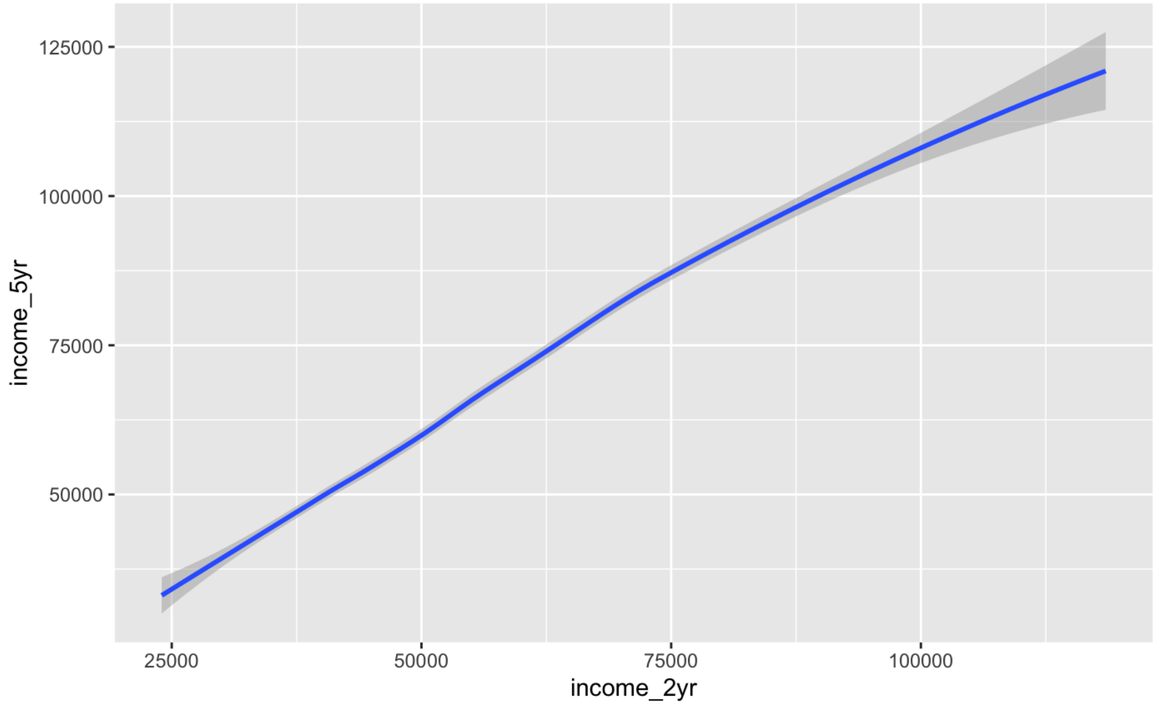


How are these plots different?

Different: geometric object (geom),

e.g. the visual object used to represent the data





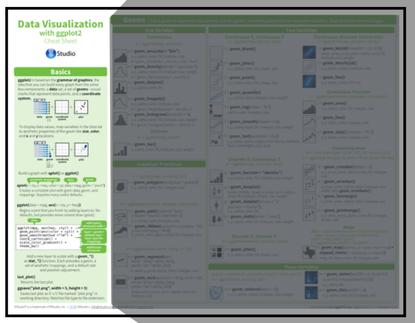
ggplot2

geoms

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

geom_ functions

Each requires a mapping argument.



Geoms - Use a geom to represent data points, use the geom's aesthetic properties to represent variables. Each function returns a layer.

One Variable

Continuous

a <- ggplot(mpg, aes(hwy))

- a + **geom_area(stat = "bin")** — x, y, alpha, color, fill, linetype, size; b + geom_area(aes(y = ..density..), stat = "bin")
- a + **geom_density(**kernel = "gaussian"**)** — x, y, alpha, color, fill, linetype, size, weight; b + geom_density(aes(y = ..county..))
- a + **geom_dotplot()** — x, y, alpha, color, fill
- a + **geom_freqpoly()** — x, y, alpha, color, linetype, size; b + geom_freqpoly(aes(y = ..density..))
- a + **geom_histogram(**binwidth = 5**)** — x, y, alpha, color, fill, linetype, size, weight; b + geom_histogram(aes(y = ..density..))

Discrete

b <- ggplot(mpg, aes(fl))

- b + **geom_bar()** — x, alpha, color, fill, linetype, size, weight

Graphical Primitives

map <- map_data("state")
c <- ggplot(map, aes(long, lat))

- c + **geom_polygon(**aes(group = group)**)** — x, y, alpha, color, fill, linetype, size

d <- ggplot(economics, aes(date, unemploy))

- d + **geom_path(**lineend="butt", linejoin="round', linemitre=1**)** — x, y, alpha, color, linetype, size
- d + **geom_ribbon(**aes(ymin=unemploy - 900, ymax=unemploy + 900**)**) — x, ymax, ymin, alpha, color, fill, linetype, size

e <- ggplot(seals, aes(x = long, y = lat))

- e + **geom_segment(**aes(xend = long + delta_long, yend = lat + delta_lat**)**) — x, xend, y, yend, alpha, color, linetype, size
- e + **geom_rect(**aes(xmin = long, ymin = lat, xmax= long + delta_long, ymax = lat + delta_lat**)**) — xmax, xmin, ymax, ymin, alpha, color, fill, linetype, size

Two Variables

Continuous X, Continuous Y

f <- ggplot(mpg, aes(cty, hwy))

- f + **geom_blank()** (Useful for expanding limits)
- f + **geom_jitter()** — x, y, alpha, color, fill, shape, size
- f + **geom_point()** — x, y, alpha, color, fill, shape, size
- f + **geom_quantile()** — x, y, alpha, color, linetype, size, weight
- f + **geom_rug(**sides = "bl"**)** — alpha, color, linetype, size
- f + **geom_smooth(**method = lm**)** — x, y, alpha, color, fill, linetype, size, weight
- f + **geom_text(**aes(label = cty)**)** — x, y, label, alpha, angle, color, family, fontface, hjust, lineheight, size, vjust

Discrete X, Continuous Y

g <- ggplot(mpg, aes(class, hwy))

- g + **geom_bar(stat = "identity")** — x, y, alpha, color, fill, linetype, size, weight
- g + **geom_boxplot()** — lower, middle, upper, x, ymax, ymin, alpha, color, fill, linetype, shape, size, weight
- g + **geom_dotplot(**binaxis = "y", stackdir = "center"**)** — x, y, alpha, color, fill
- g + **geom_violin(**scale = "area"**)** — x, y, alpha, color, fill, linetype, size, weight

Discrete X, Discrete Y

h <- ggplot(diamonds, aes(cut, color))

- h + **geom_jitter()** — x, y, alpha, color, fill, shape, size

Continuous Bivariate Distribution

i <- ggplot(movies, aes(year, rating))

- i + **geom_bin2d(**binwidth = c(5, 0.5)**)** — xmax, xmin, ymax, ymin, alpha, color, fill, linetype, size, weight
- i + **geom_density2d()** — x, y, alpha, colour, linetype, size
- i + **geom_hex()** — x, y, alpha, colour, fill size

Continuous Function

j <- ggplot(economics, aes(date, unemploy))

- j + **geom_area()** — x, y, alpha, color, fill, linetype, size
- j + **geom_line()** — x, y, alpha, color, linetype, size
- j + **geom_step(**direction = "hv"**)** — x, y, alpha, color, linetype, size

Visualizing error

df <- data.frame(grp = c("A", "B"), fit = 4:5, se = 1:2)
k <- ggplot(df, aes(grp, fit, ymin = fit-se, ymax = fit+se))

- k + **geom_crossbar(**fatten = 2**)** — x, y, ymax, ymin, alpha, color, fill, linetype, size
- k + **geom_errorbar()** — x, ymax, ymin, alpha, color, linetype, size, width (also **geom_errorbarh()**)
- k + **geom_linerange()** — x, ymin, ymax, alpha, color, linetype, size
- k + **geom_pointrange()** — x, y, ymin, ymax, alpha, color, fill, linetype, shape, size

Maps

data <- data.frame(murder = USArrests$Murder, state = tolower(rownames(USArrests)))
map <- map_data("state")
l <- ggplot(data, aes(fill = murder))

- l + **geom_map(**aes(map_id = state), map = map**)** + **expand_limits(**x = map$long, y = map$lat**)** — map_id, alpha, color, fill, linetype, size

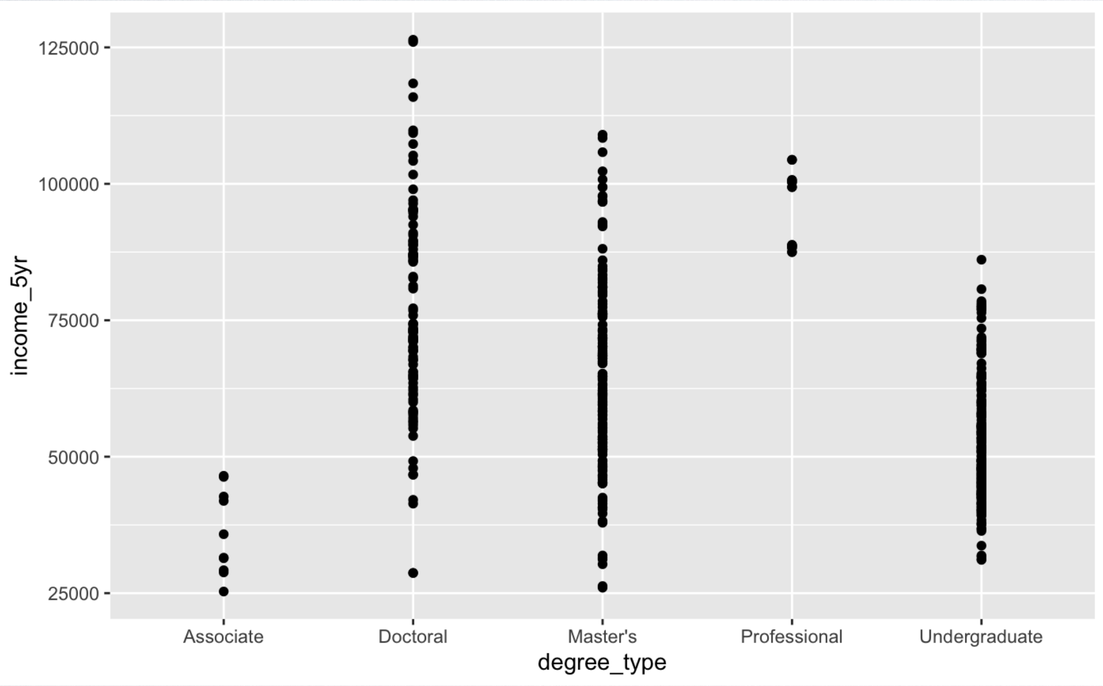
Three Variables

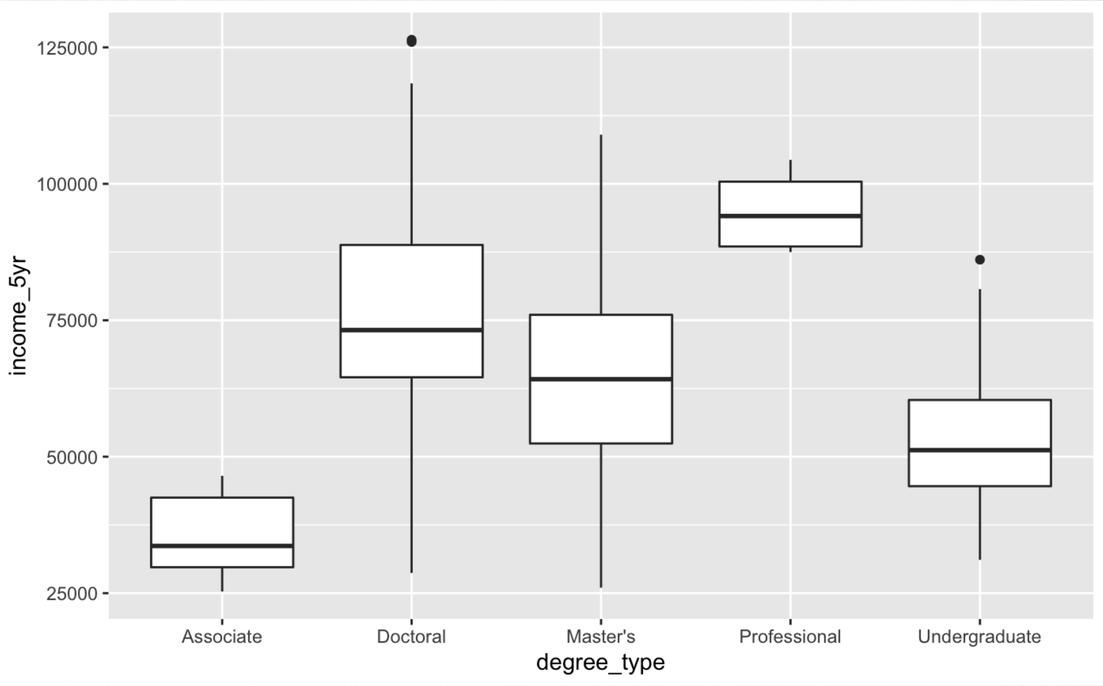
seals$z <- with(seals, sqrt(delta_long^2 + delta_lat^2))
m <- ggplot(seals, aes(long, lat))

- m + **geom_contour(**aes(z = z)**)** — x, y, z, alpha, colour, linetype, size, weight
- m + **geom_raster(**aes(fill = z), hjust=0.5, vjust=0.5, interpolate=FALSE**)** — x, y, alpha, fill (fast)
- m + **geom_tile(**aes(fill = z)**)** — x, y, alpha, color, fill, linetype, size (slow)

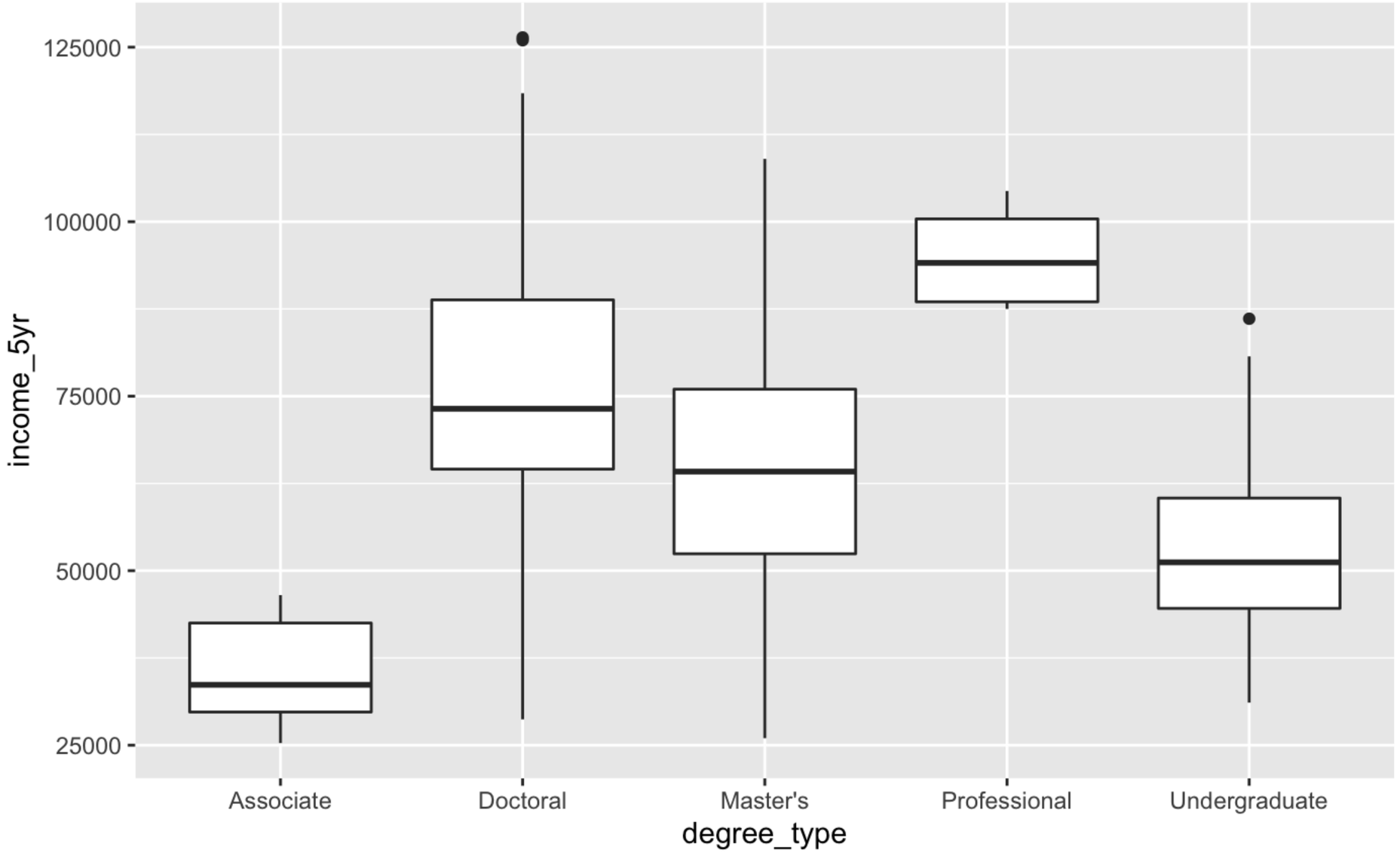
Your Turn 3

With your partner, decide how to replace this scatterplot with one that draws boxplots. Use the cheatsheet. Try your best guess.





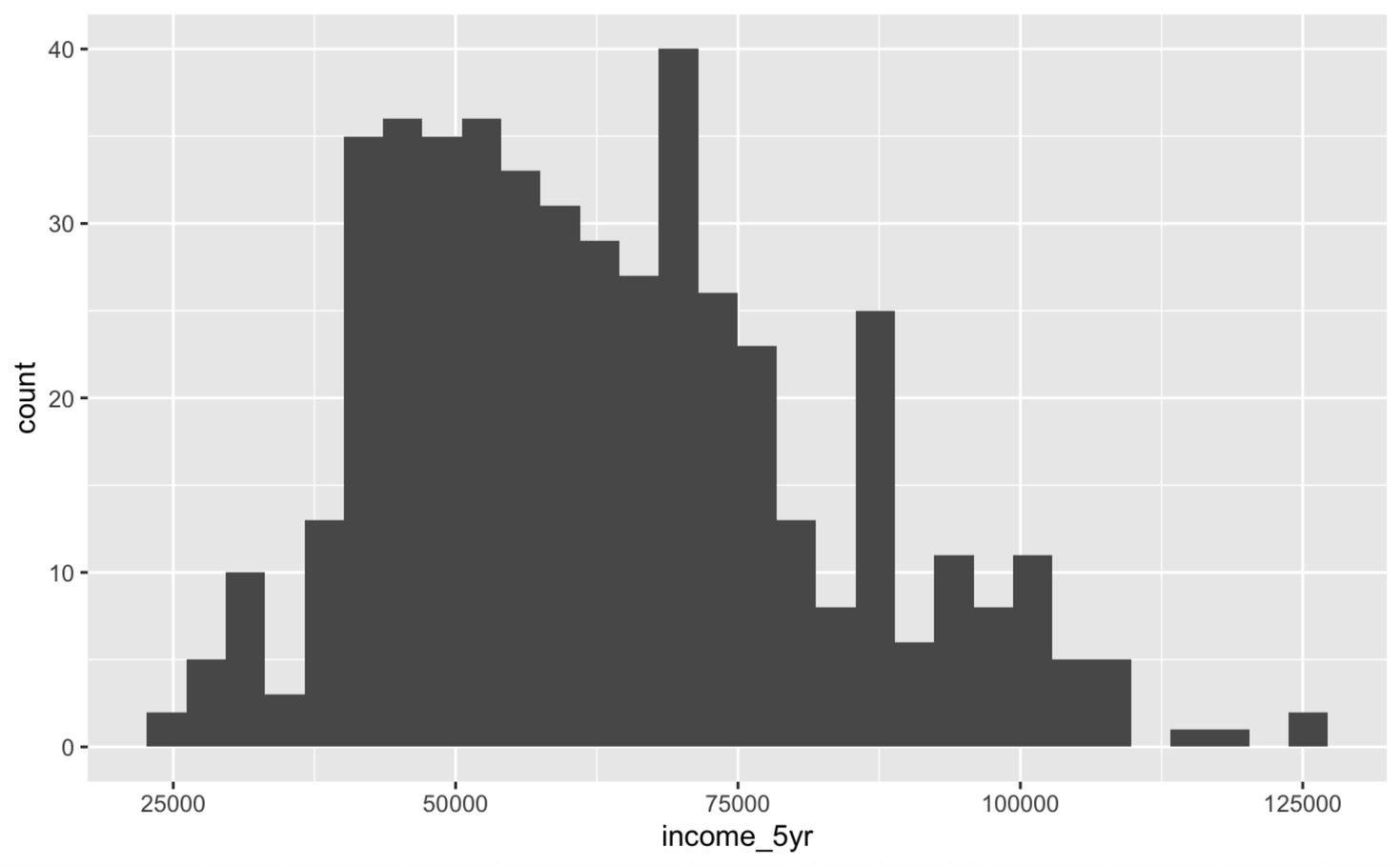

```
ggplot(income) +
  geom_point(aes(rock_name, income_5yr))
```

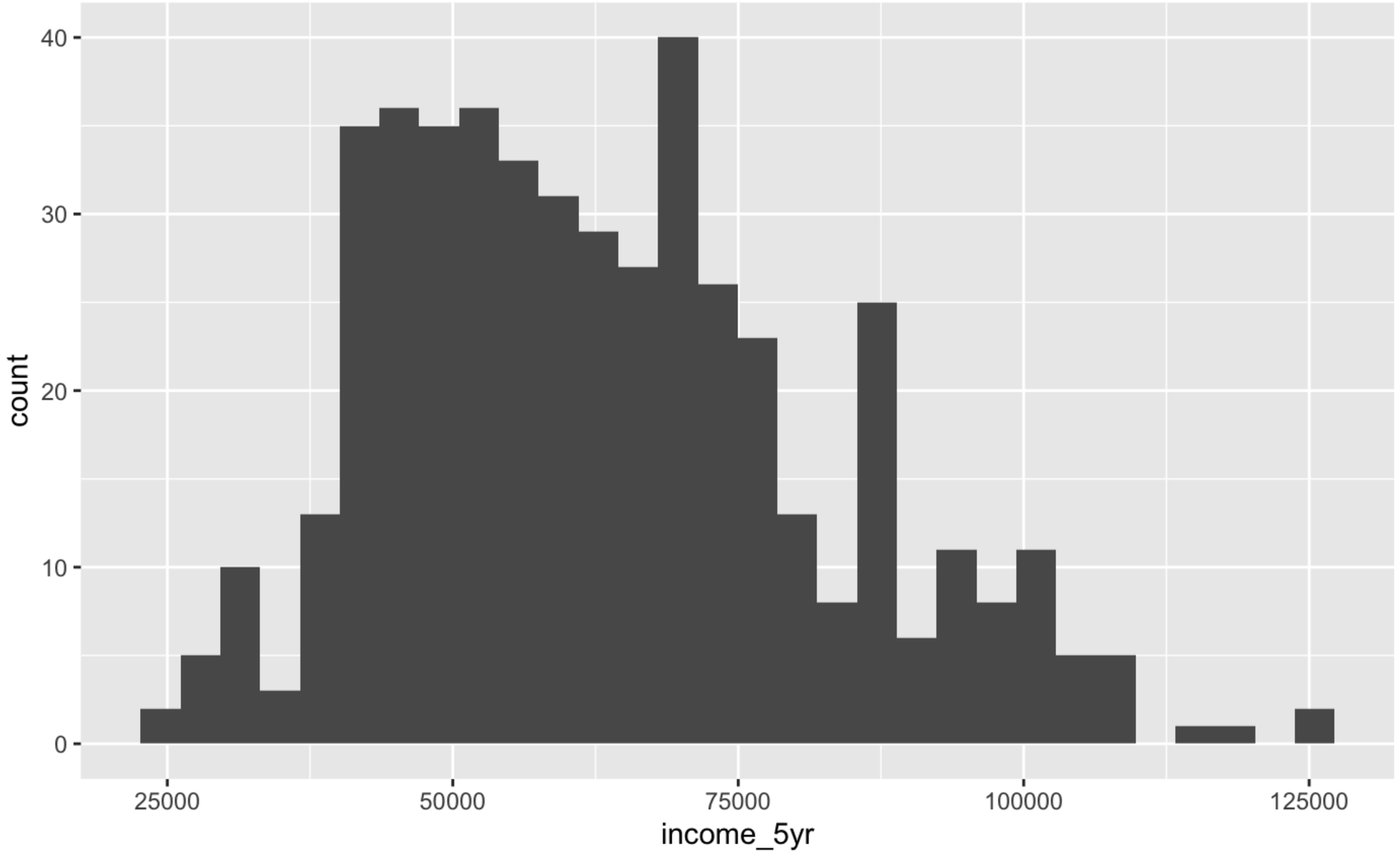



```
ggplot(income) +
  geom_boxplot(aes(rock_name, income_5yr))
```

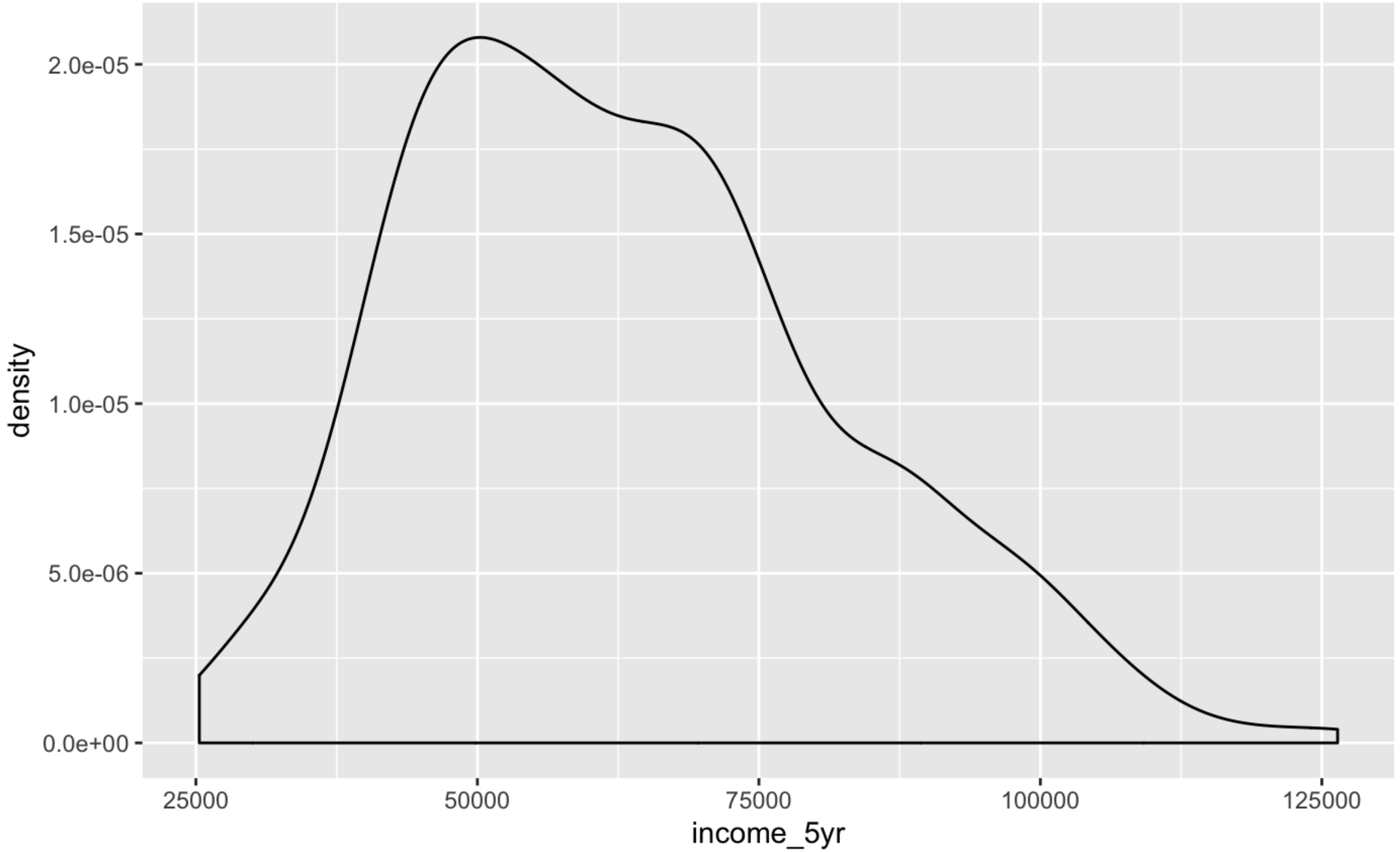
Your Turn 4

With your partner, make the histogram of **income_5yr** below.
Use the cheatsheet. Hint: do not supply a **y** variable.





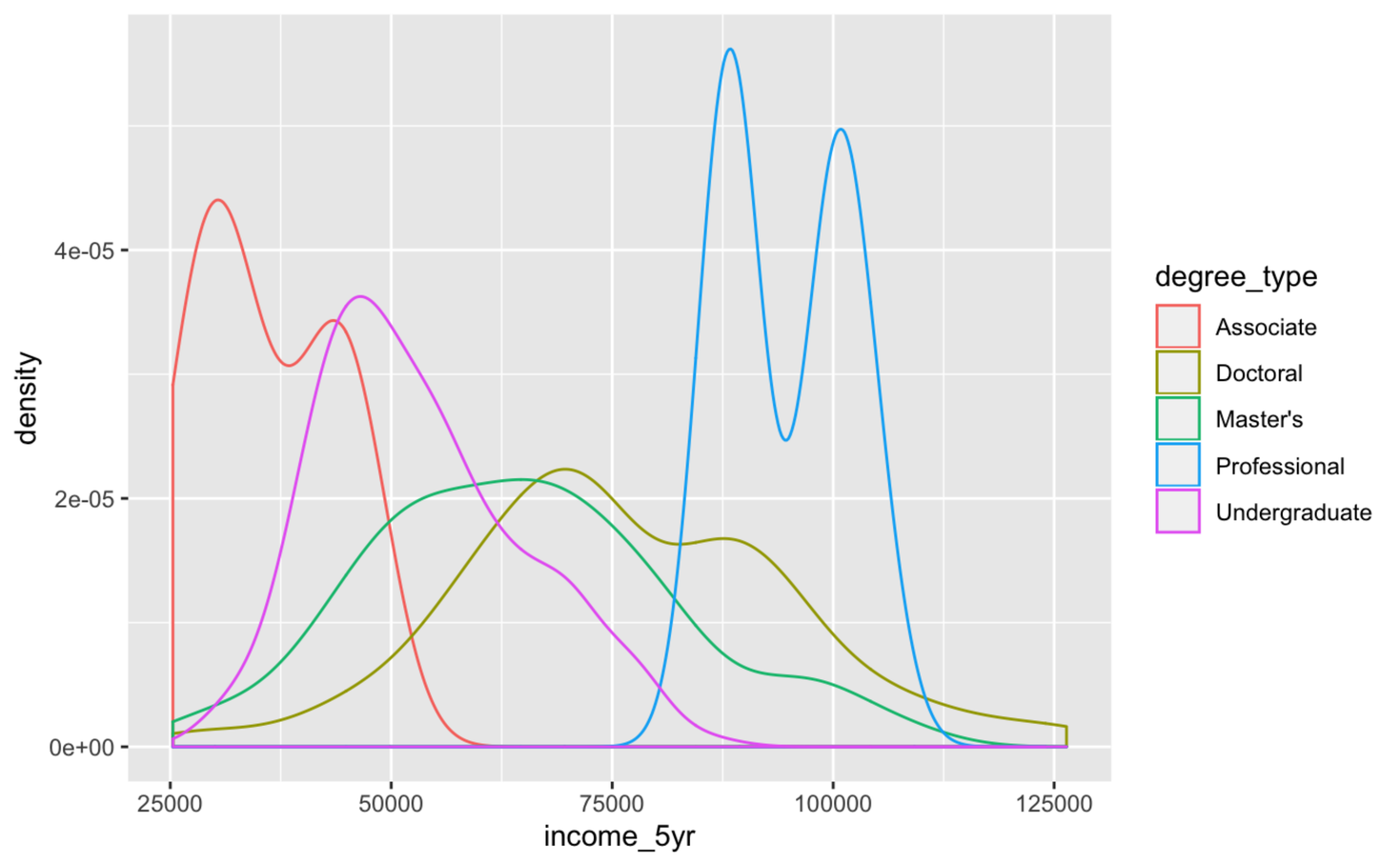

```
ggplot(data = income) +
  geom_histogram(mapping = aes(x = income_5yr))
```



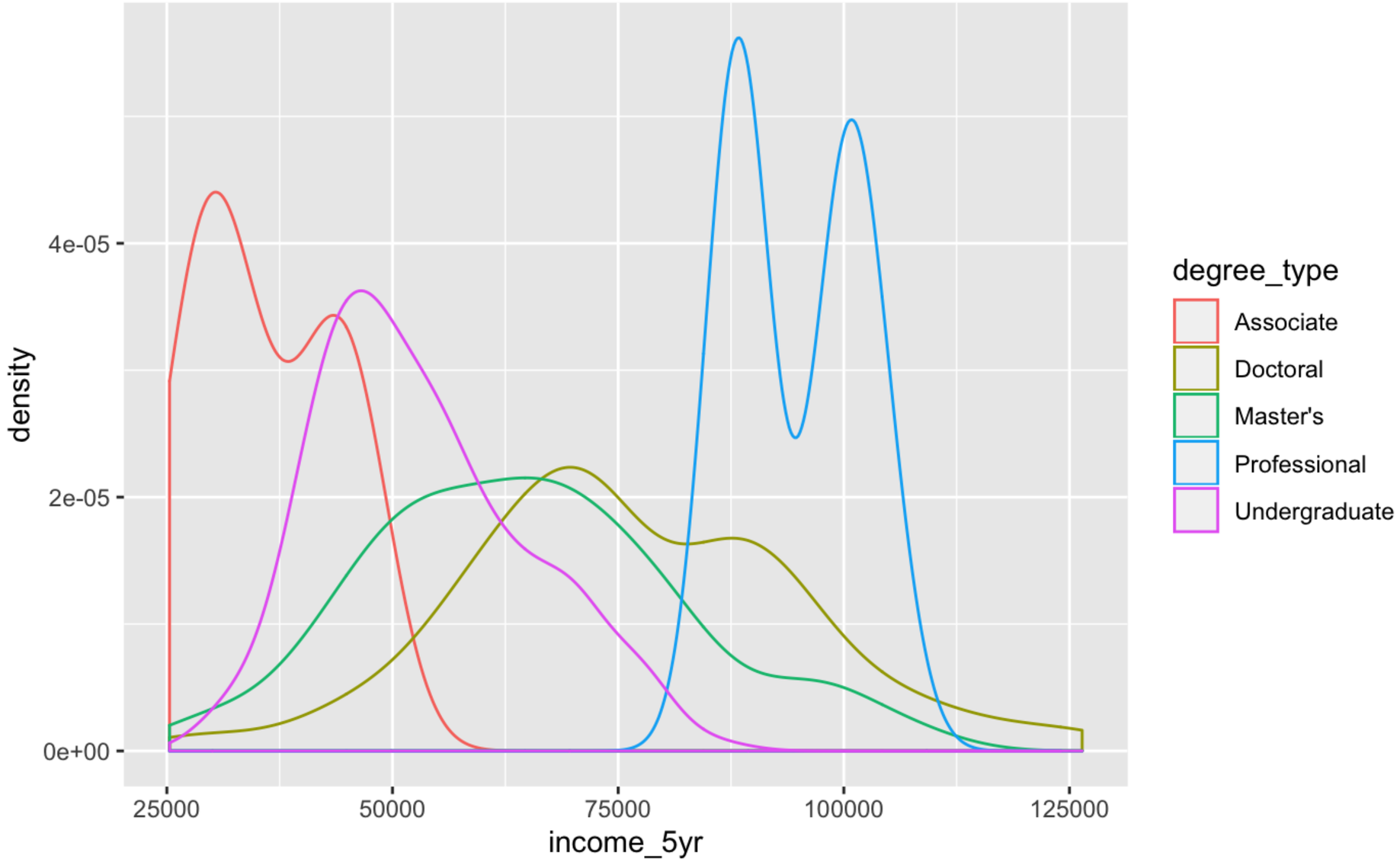

```
ggplot(data = income) +
  geom_density(mapping = aes(x = income_5yr))
```

Your Turn 5

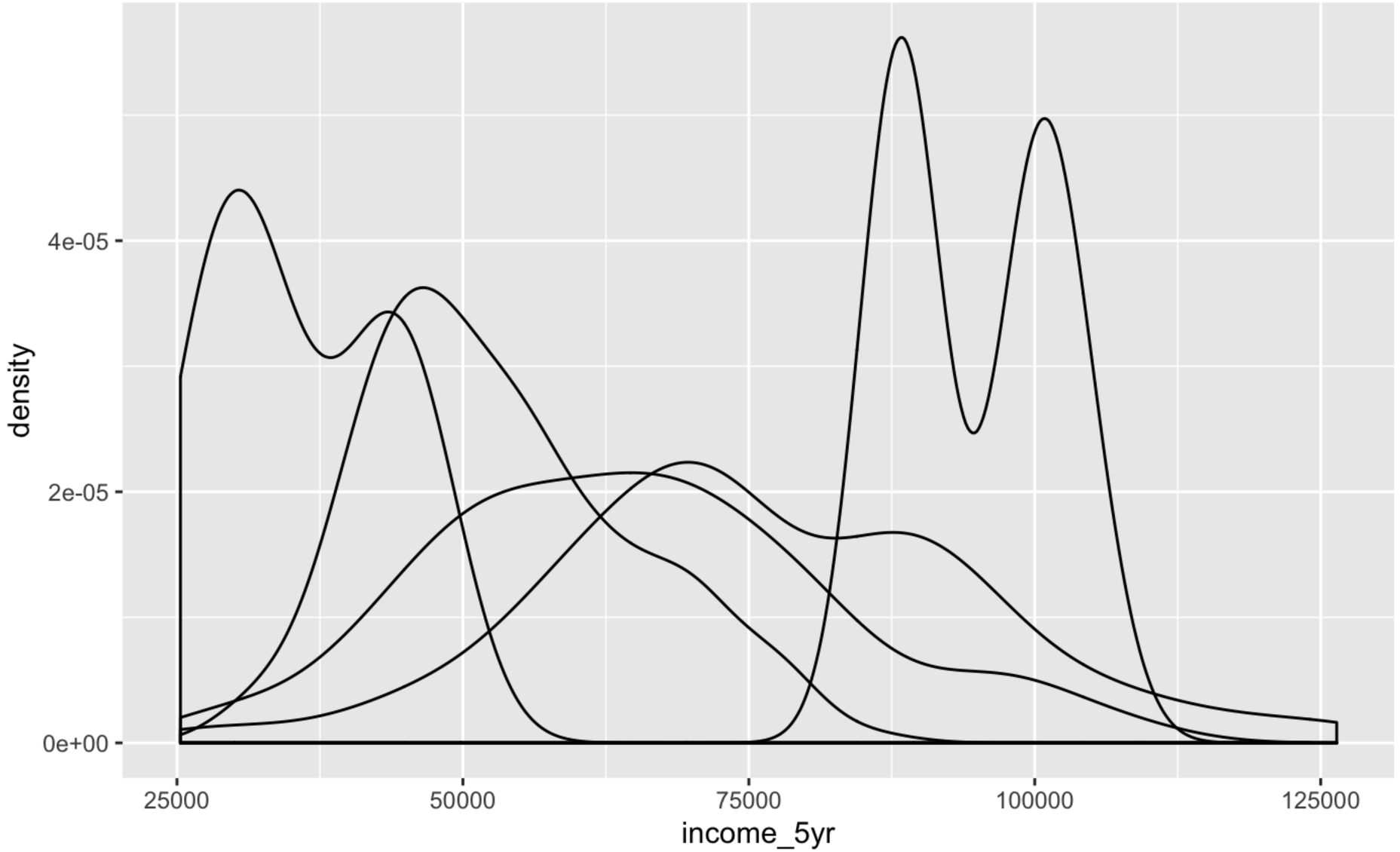
With your partner, make the density plot of **income_5yr** colored by **degree_type** below. Use the cheatsheet. Try your best guess.



02:00



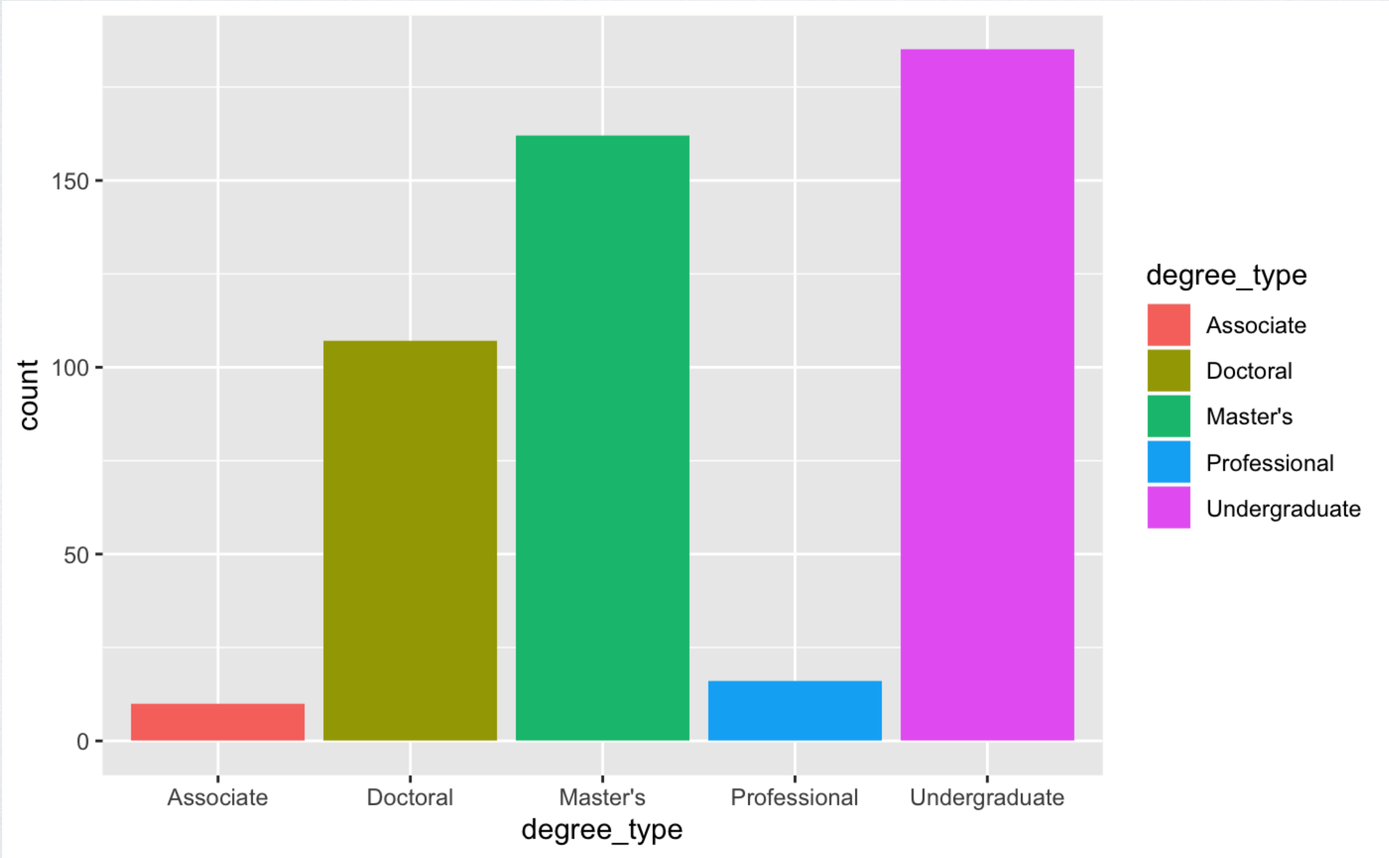

```
ggplot(data = income) +
  geom_density(mapping = aes(x = income_5yr, color = degree_type))
```

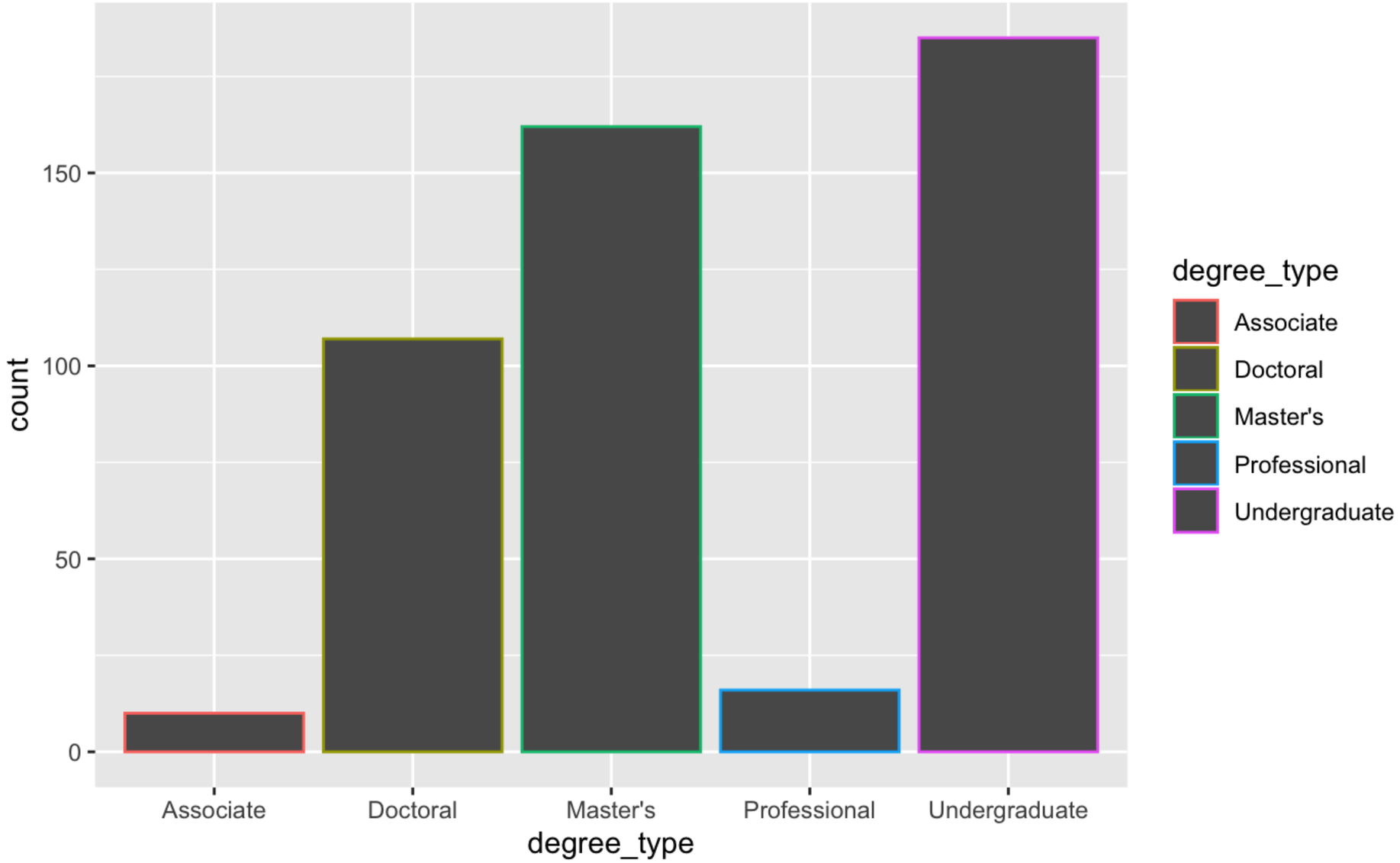



```
ggplot(data = income) +
  geom_density(mapping = aes(x = income_5yr, group = degree_type))
```

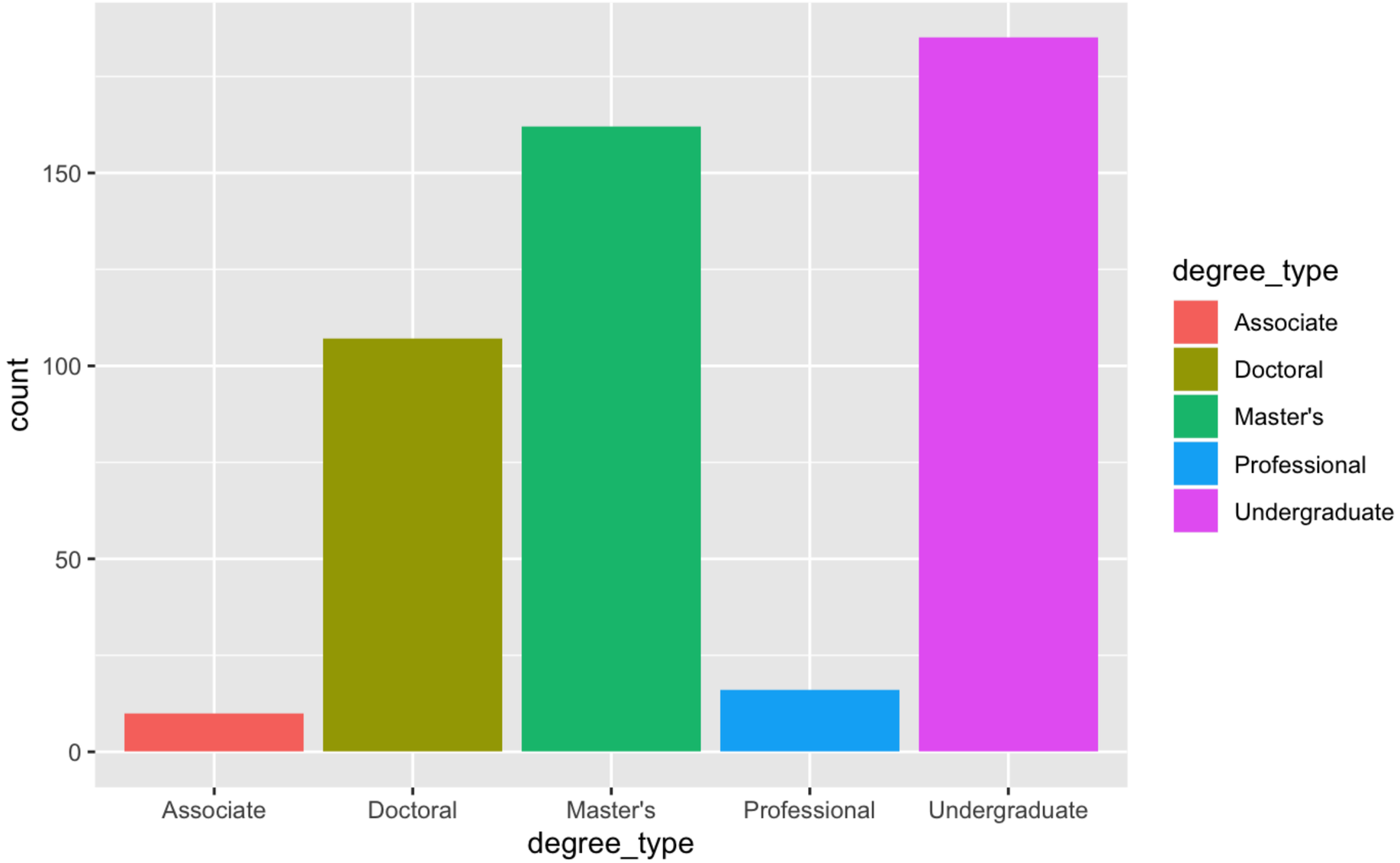
Your Turn

With your partner, make the bar chart of **degree_type** colored by **degree_type** below. Use the cheatsheet. Try your best guess.






```
ggplot(data = income) +
  geom_bar(mapping = aes(x = degree_type, color = degree_type))
```



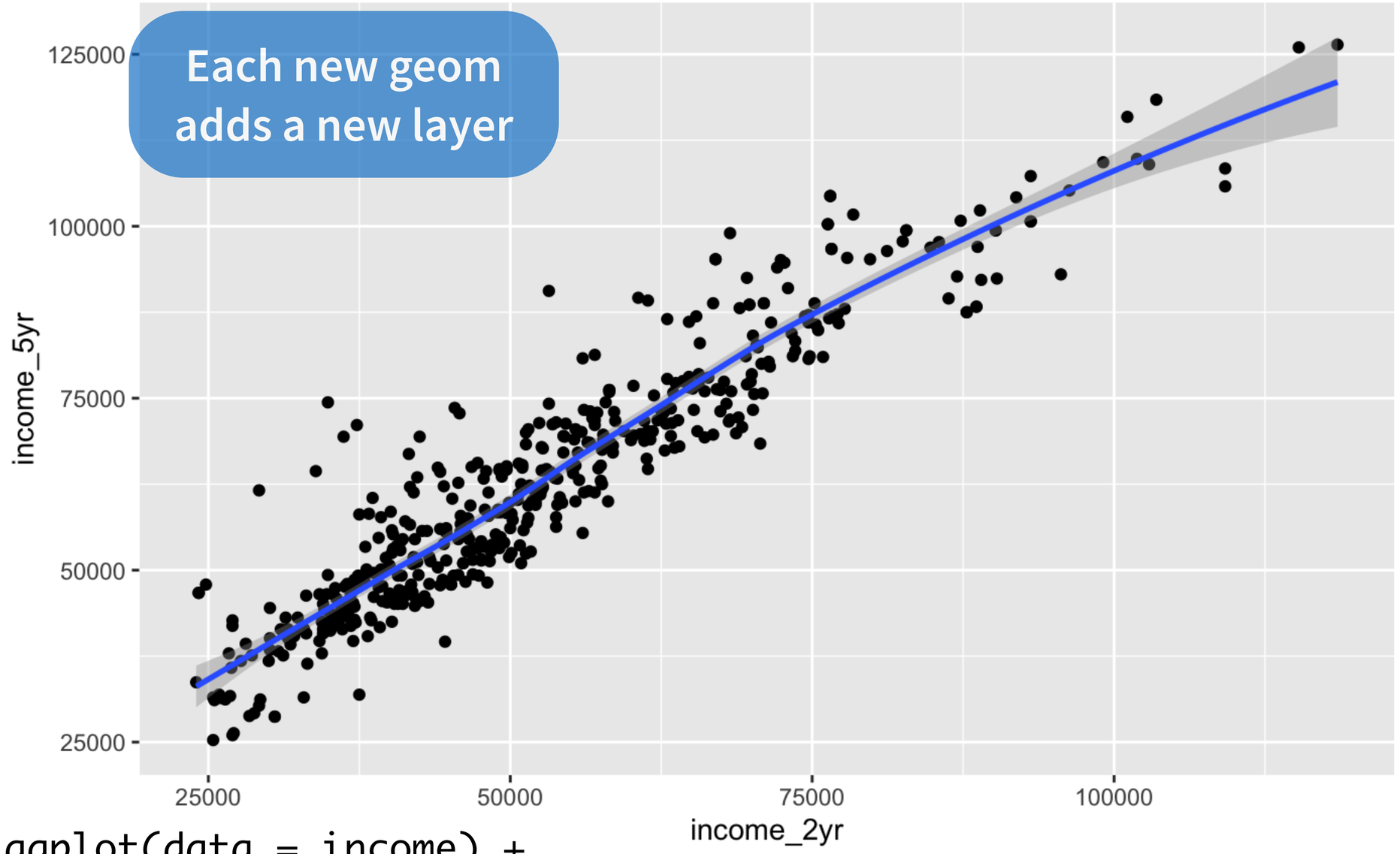

```
ggplot(data = income) +
  geom_bar(mapping = aes(x = degree_type, fill = degree_type))
```

Your Turn 7

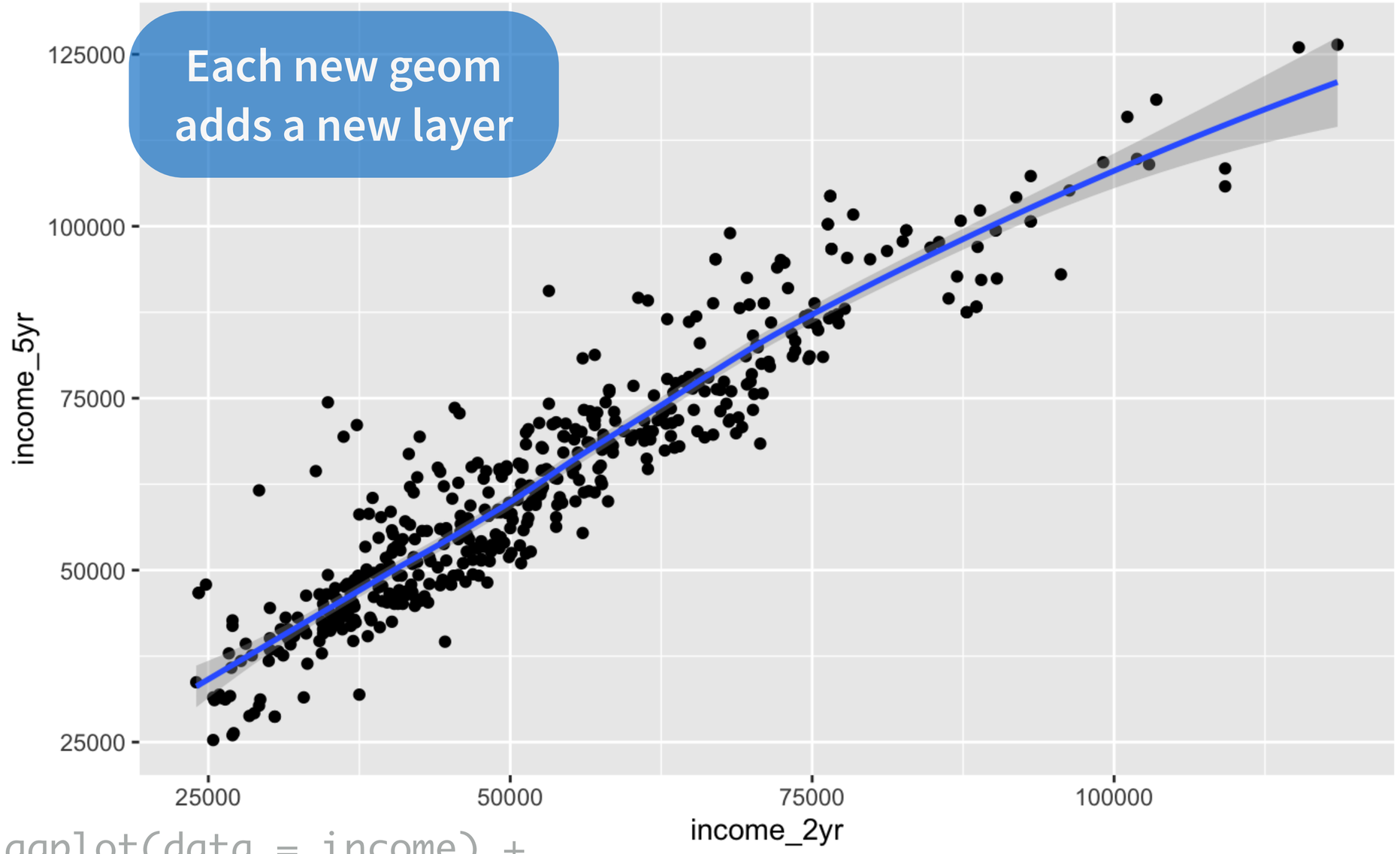
With your partner, predict what this code will do.

Then run it.

```
ggplot(income) +
  geom_point(aes(income_2yr, income_5yr)) +
  geom_smooth(aes(income_2yr, income_5yr))
```



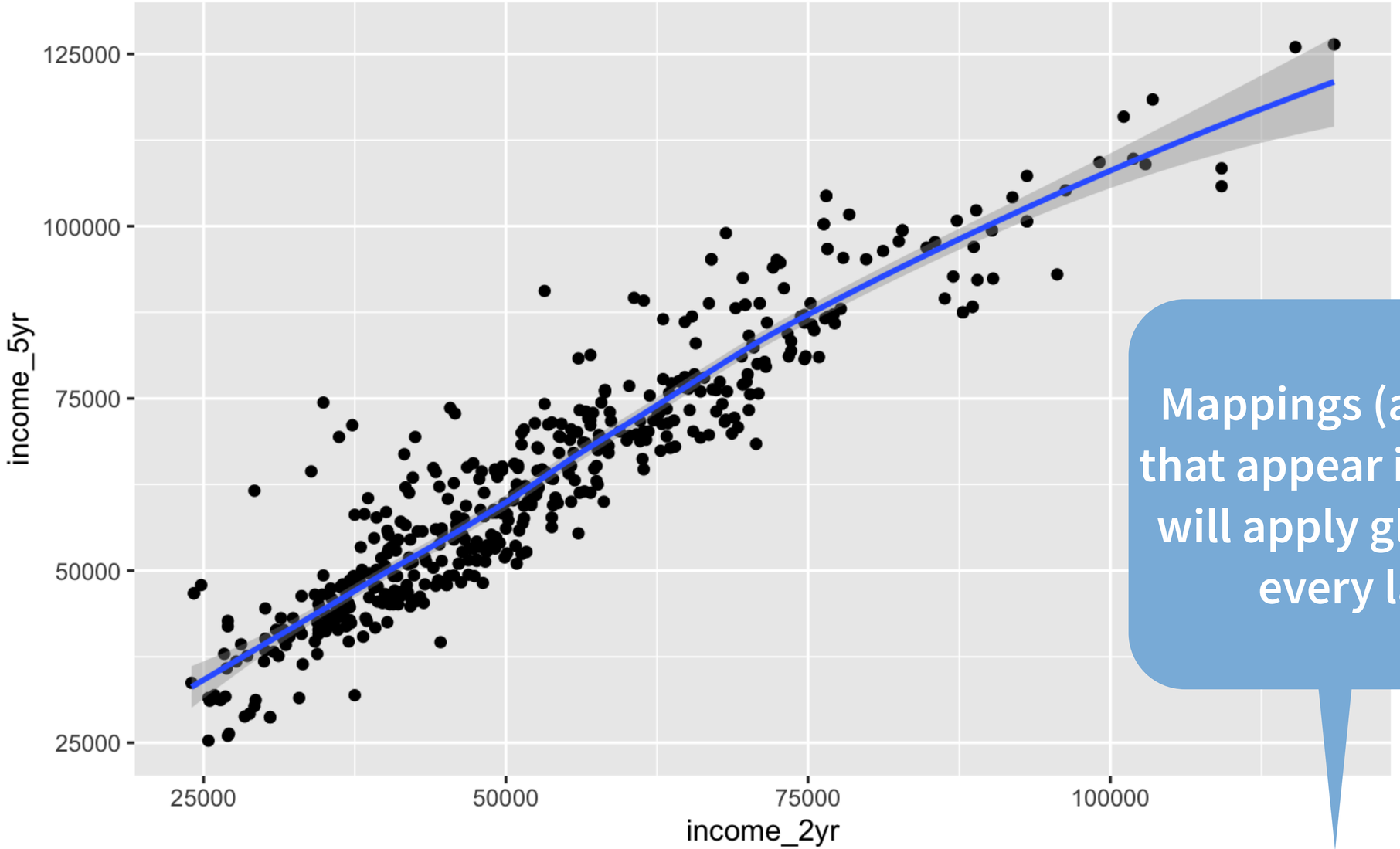

```
ggplot(data = income) +
  geom_point(mapping = aes(x = income_2yr, y = income_5yr)) +
  geom_smooth(mapping = aes(x = income_2yr, y = income_5yr))
```




```
ggplot(data = income) +
  geom_point(mapping = aes(x = income_2yr, y = income_5yr)) +
  geom_smooth(mapping = aes(x = income_2yr, y = income_5yr))
```

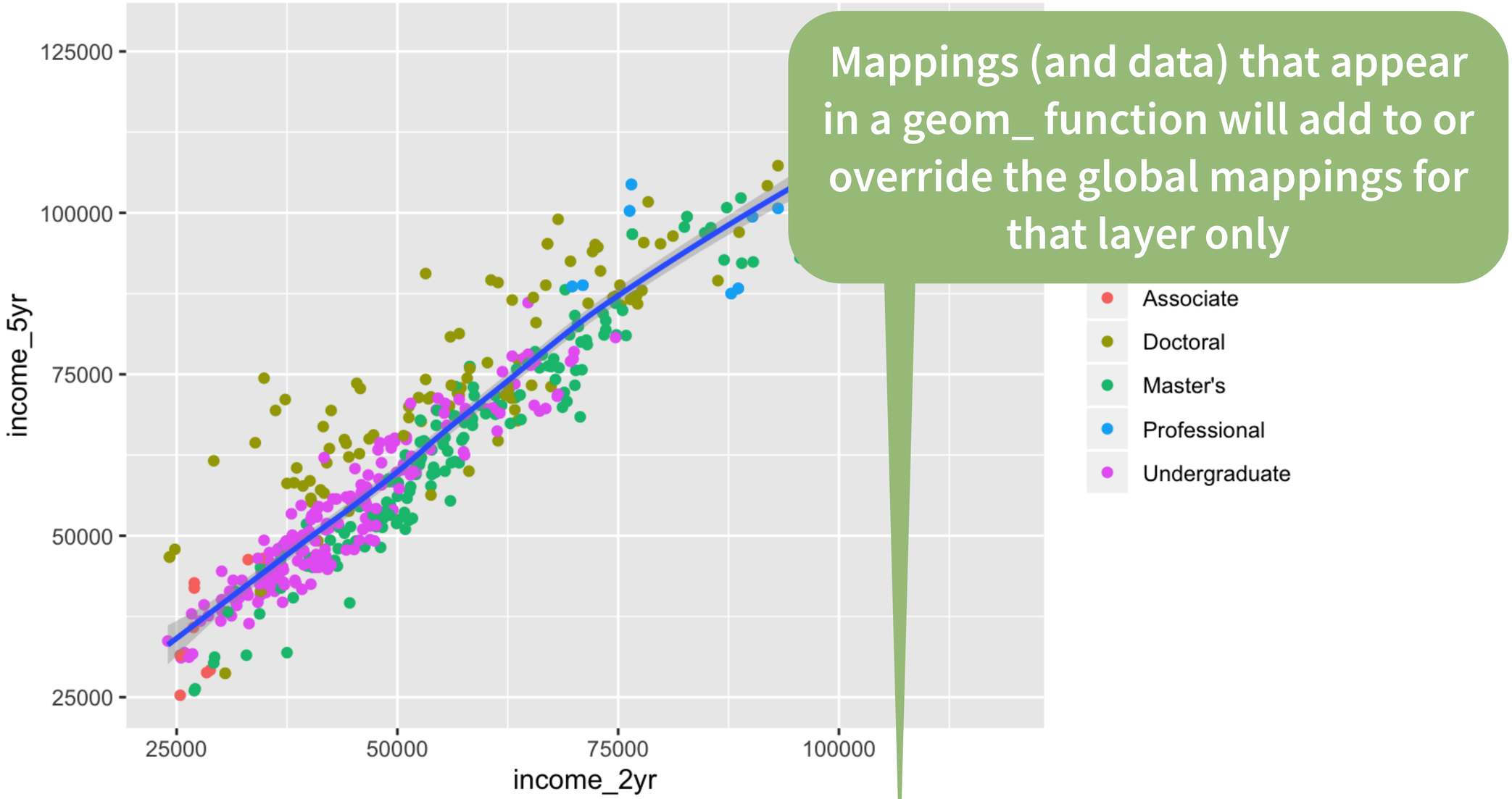
ggplot2

global vs. local



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

```
ggplot(data = income, mapping = aes(x = income_2yr, y = income_5yr)) +
  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 = income, mapping = aes(x = income_2yr, y = income_5yr)) +
  geom_point(mapping = aes(color = degree_type)) +
  geom_smooth()
```

Saving graphs

Your Turn 7

What does this command return?

```
getwd()
```

Working directory

R associates itself with a folder (i.e. directory) on your computer.

- This folder is known as your "working directory"
- When you save files, R will save them here
- When you load files, R will look for them here
- When using an R Notebook, the working directory is the same as the directory that the R Notebook is saved in

Saving plots

ggsave() saves the last plot.
Uses size on screen:

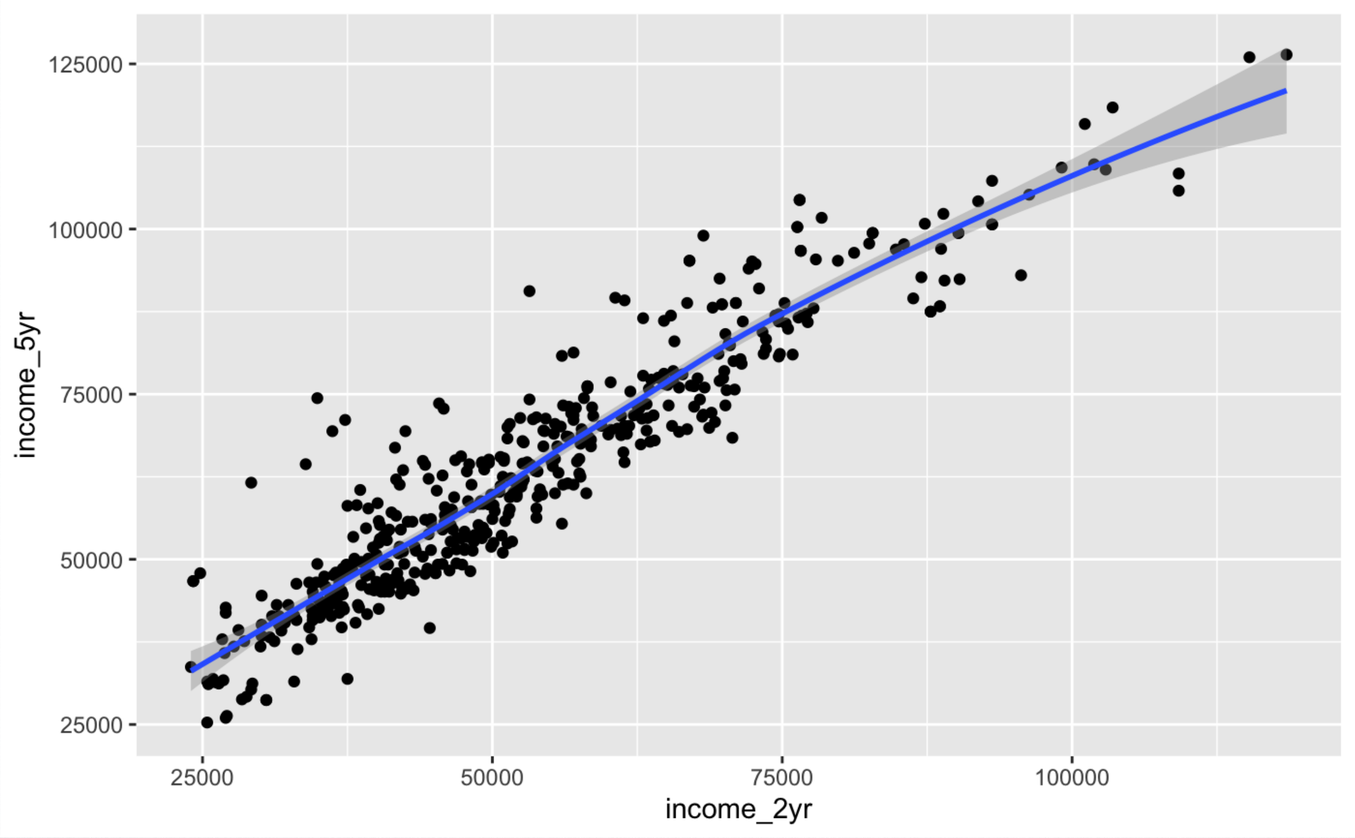
```
ggsave("my-plot.pdf")
ggsave("my-plot.png")
```

Specify size in inches

```
ggsave("my-plot.pdf", width = 6, height = 6)
```

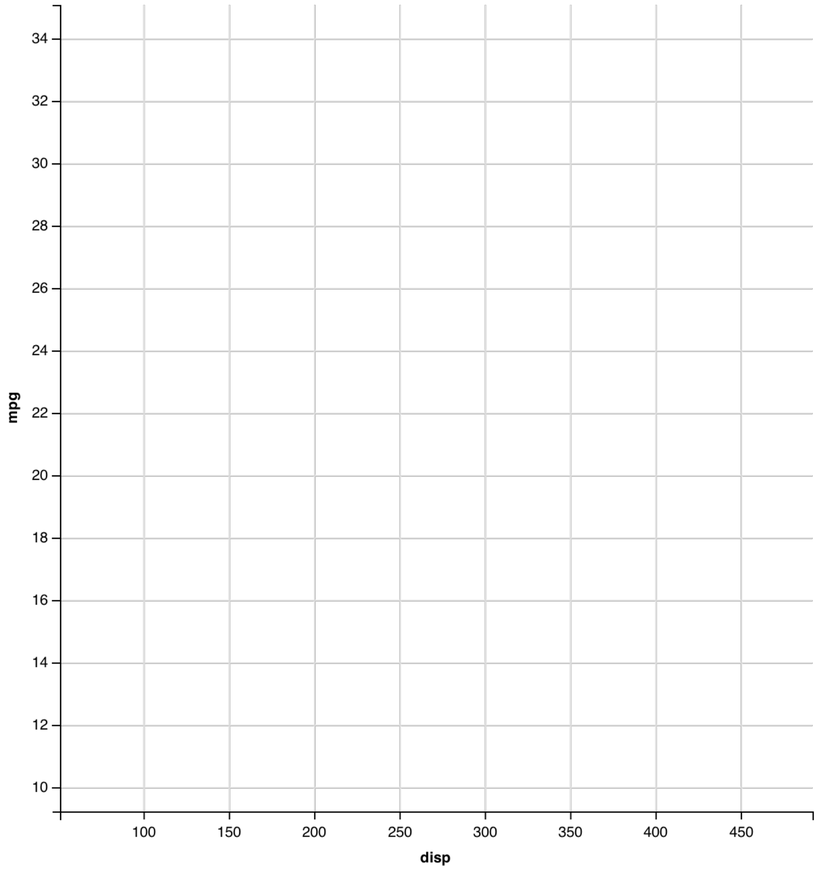
Your Turn 8

Save your last plot and then locate it in your files pane and download it. (You may have to refresh the files list).



Grammar of Graphics

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



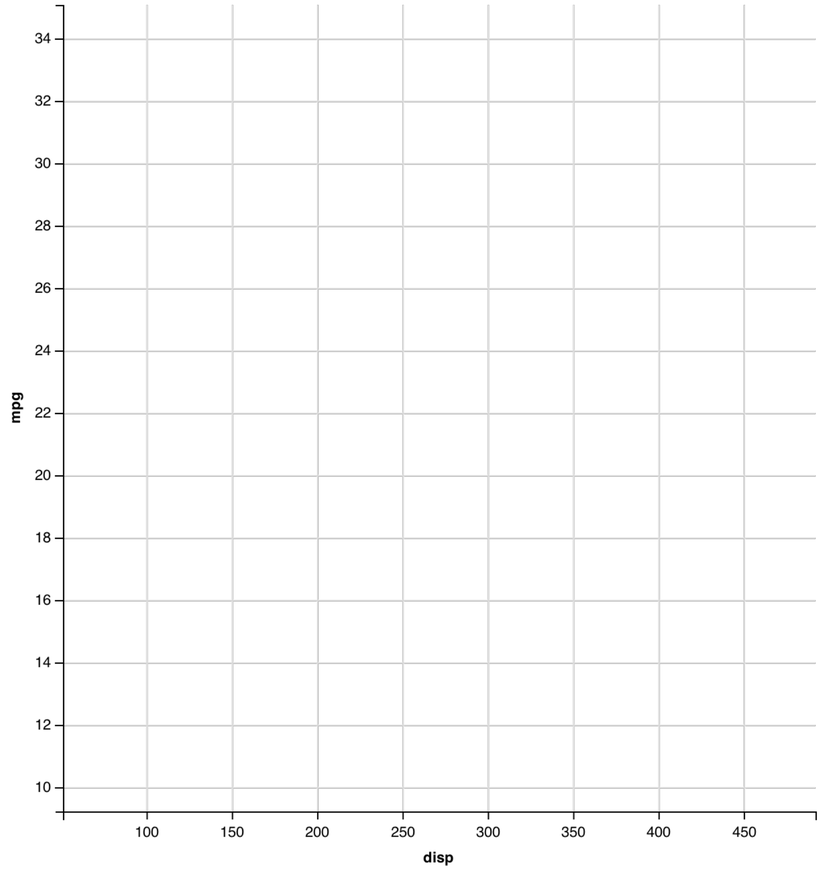
data geom

ggplot2

mappings

fill

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



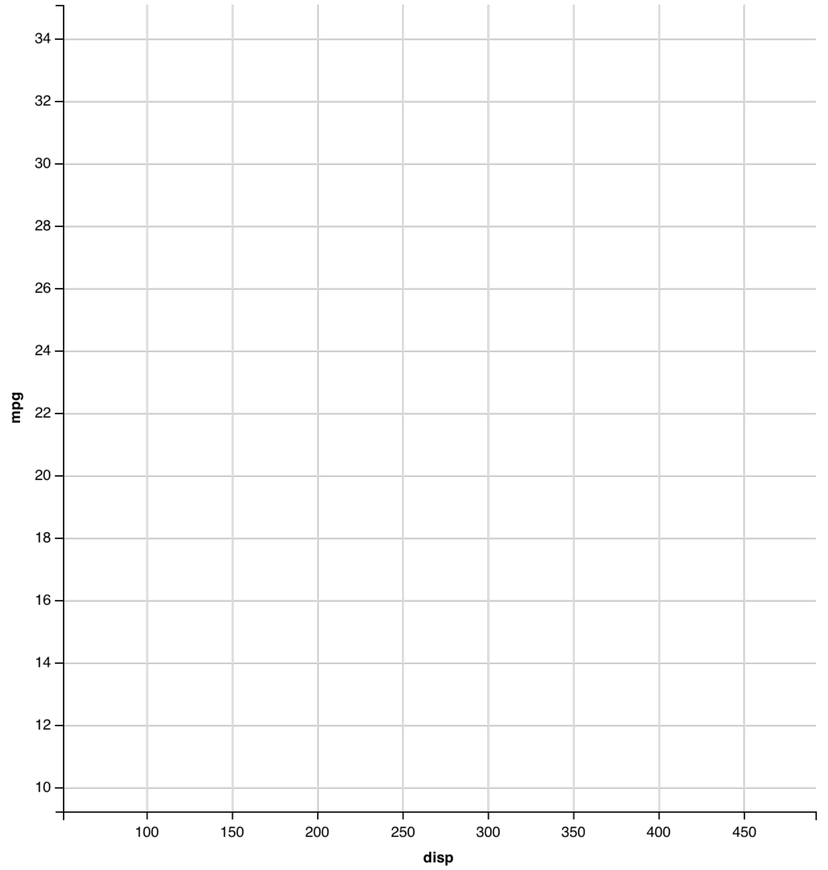
data geom

ggplot2

mappings

shape fill

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

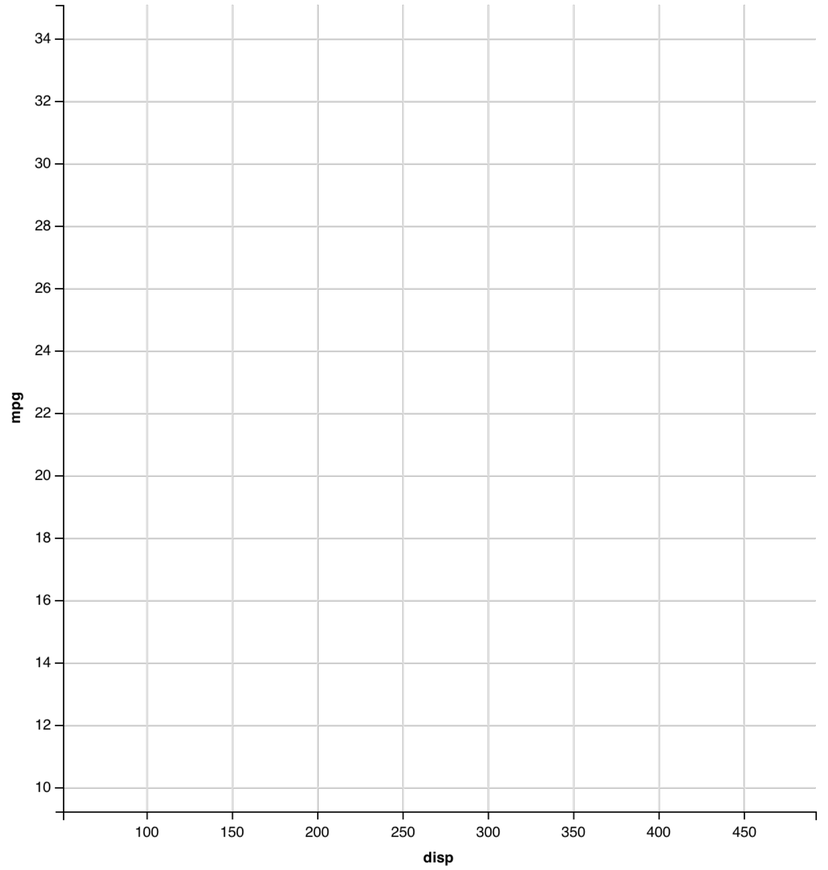


data geom

mappings

shape x fill

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



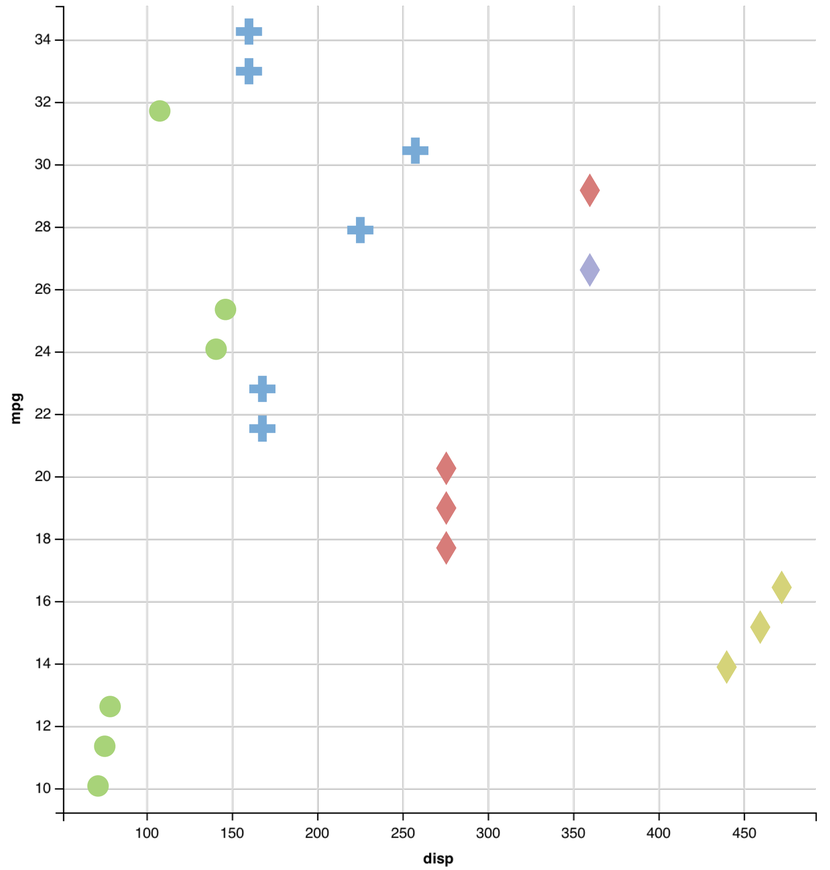
data geom

ggplot2

mappings

y shape x fill

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



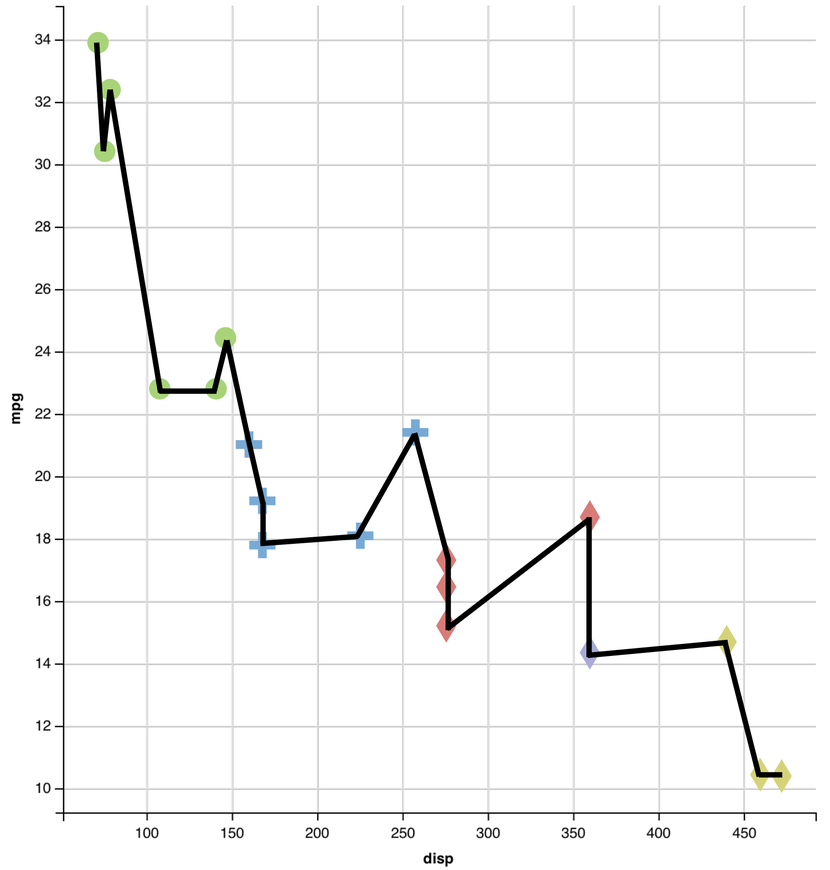
data geom

ggplot2

mappings

y shape x fill

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



data

geom

points

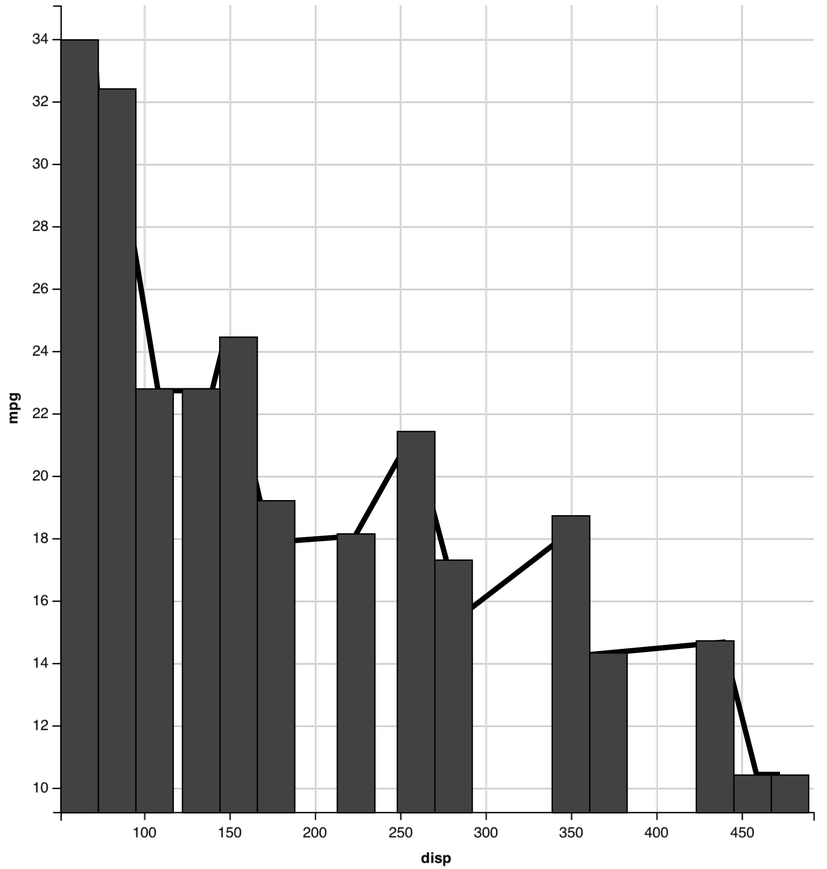
lines

ggplot2

mappings

y x

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



data **geom**

points

lines

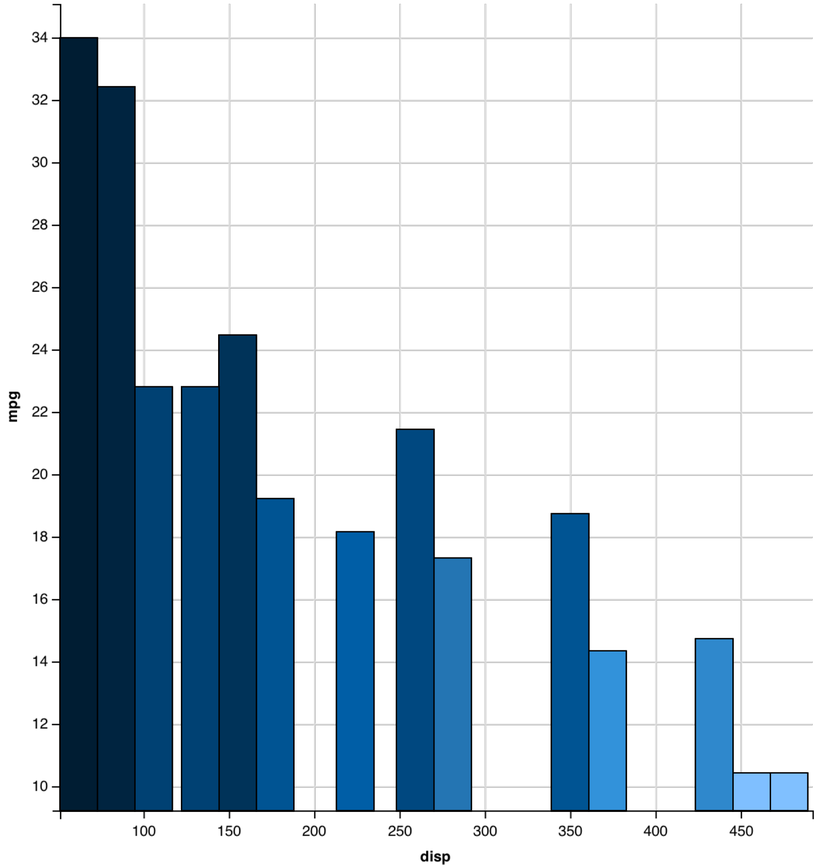
bars

ggplot2

mappings

y fill

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



data

geom

points

lines

bars

ggplot2

To make a graph

[template]

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

To make a graph

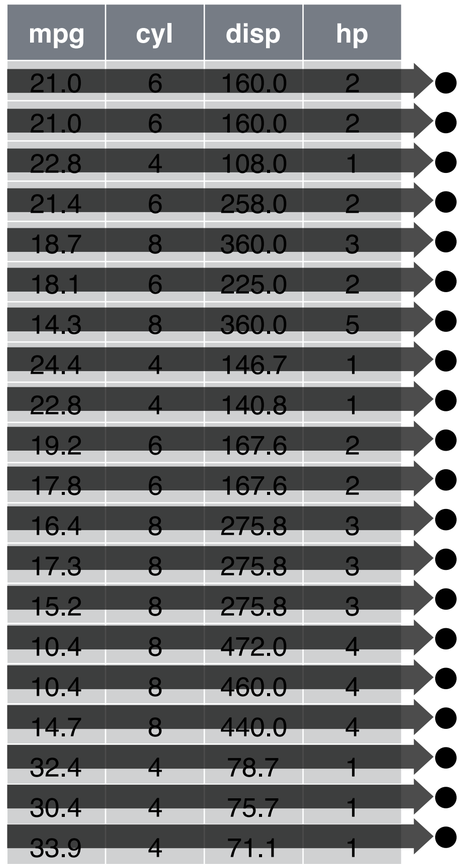
mpg	cyl	disp	hp
21.0	6	160.0	2
21.0	6	160.0	2
22.8	4	108.0	1
21.4	6	258.0	2
18.7	8	360.0	3
18.1	6	225.0	2
14.3	8	360.0	5
24.4	4	146.7	1
22.8	4	140.8	1
19.2	6	167.6	2
17.8	6	167.6	2
16.4	8	275.8	3
17.3	8	275.8	3
15.2	8	275.8	3
10.4	8	472.0	4
10.4	8	460.0	4
14.7	8	440.0	4
32.4	4	78.7	1
30.4	4	75.7	1
33.9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

data

1. Pick a **data** set

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

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

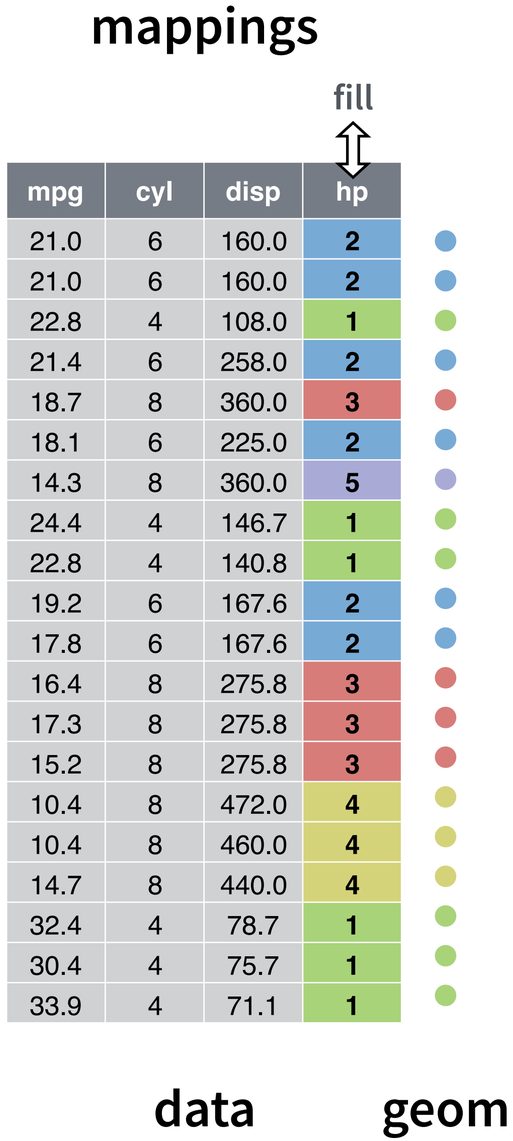
data **geom**

1. Pick a **data** set

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

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

To make a graph



1. Pick a **data** set

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

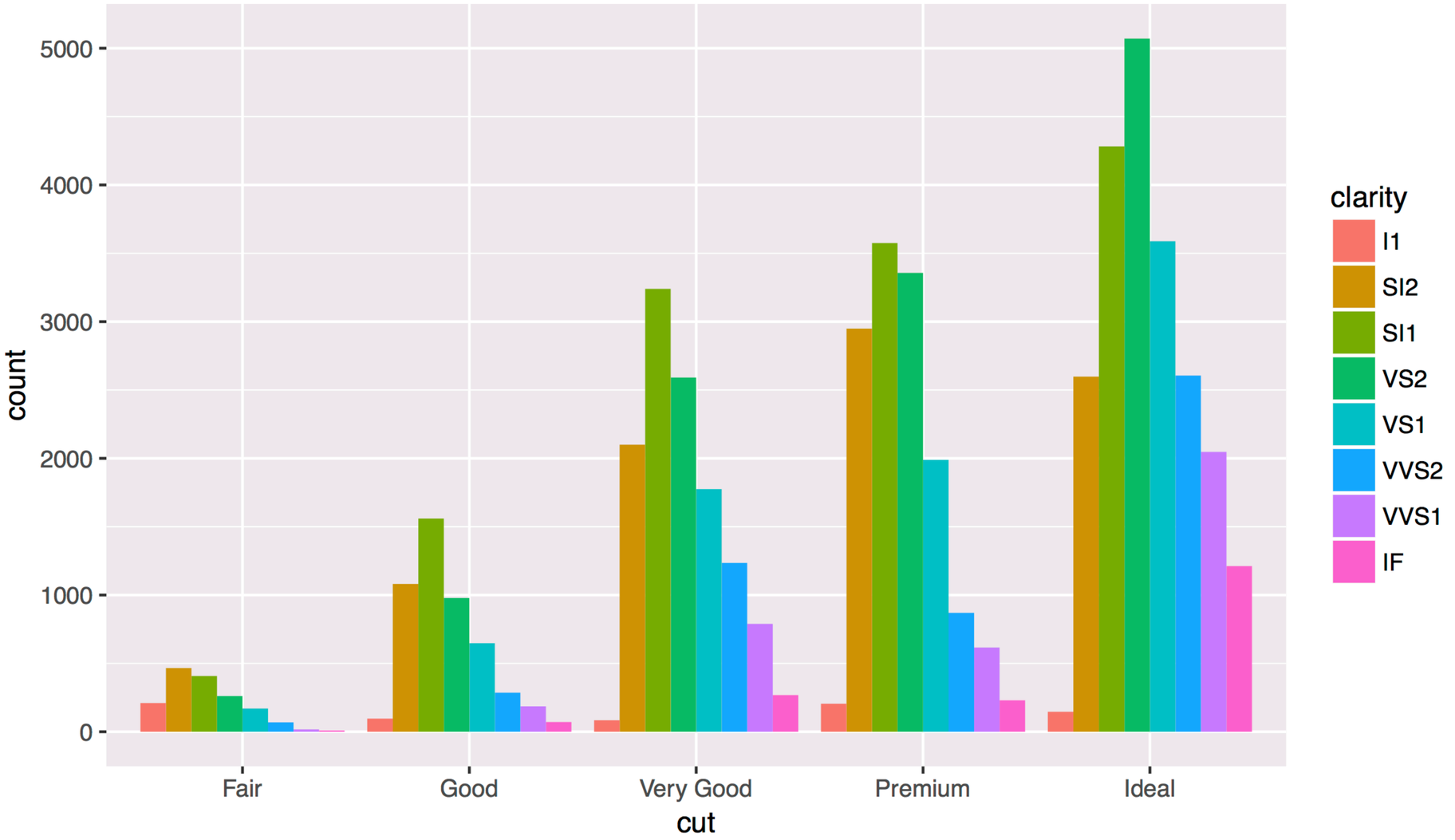
2. Choose a **geom** to display cases

3. **Map** aesthetic properties to variables

What else?

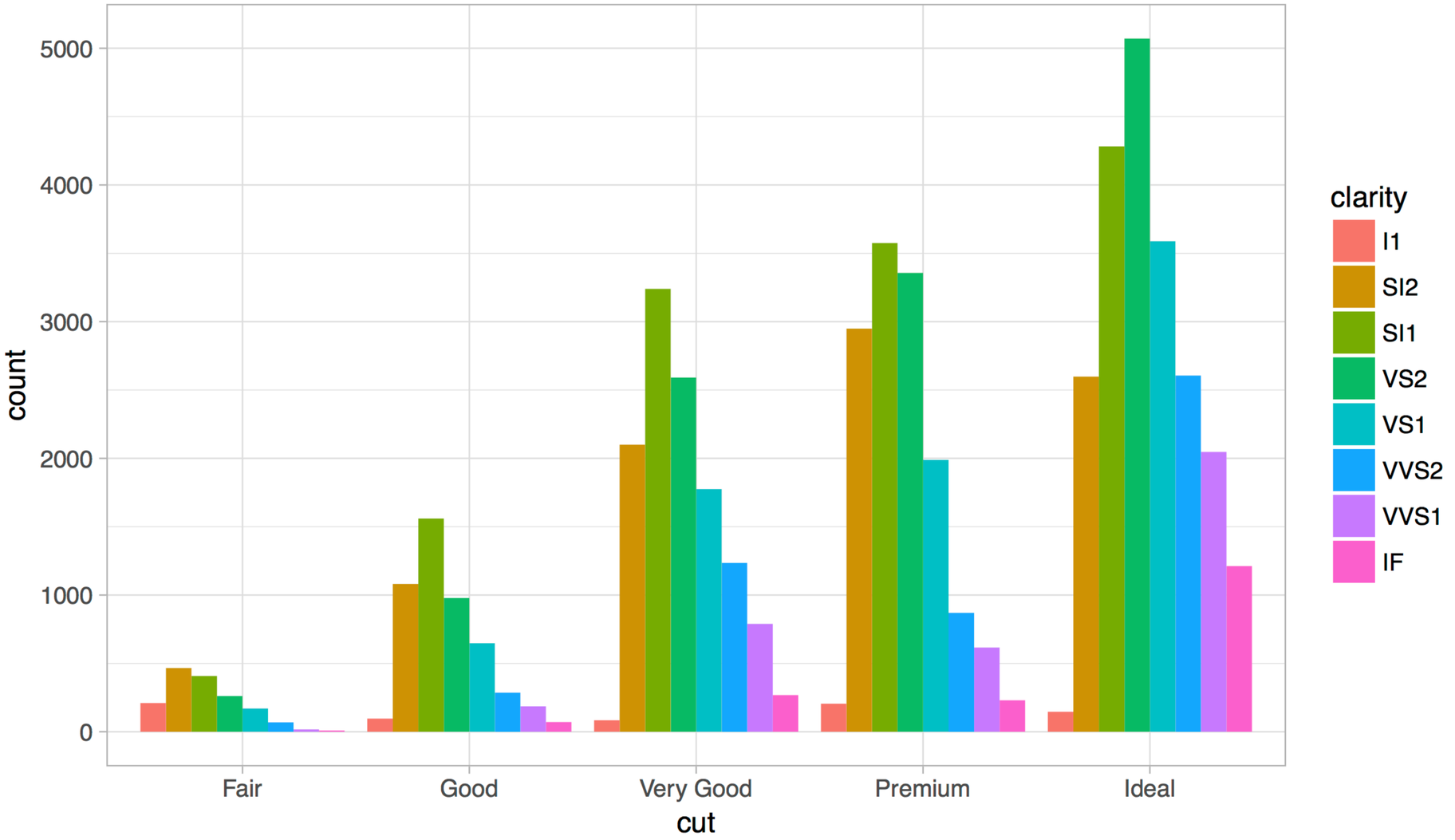
Position Adjustments

How overlapping objects are arranged



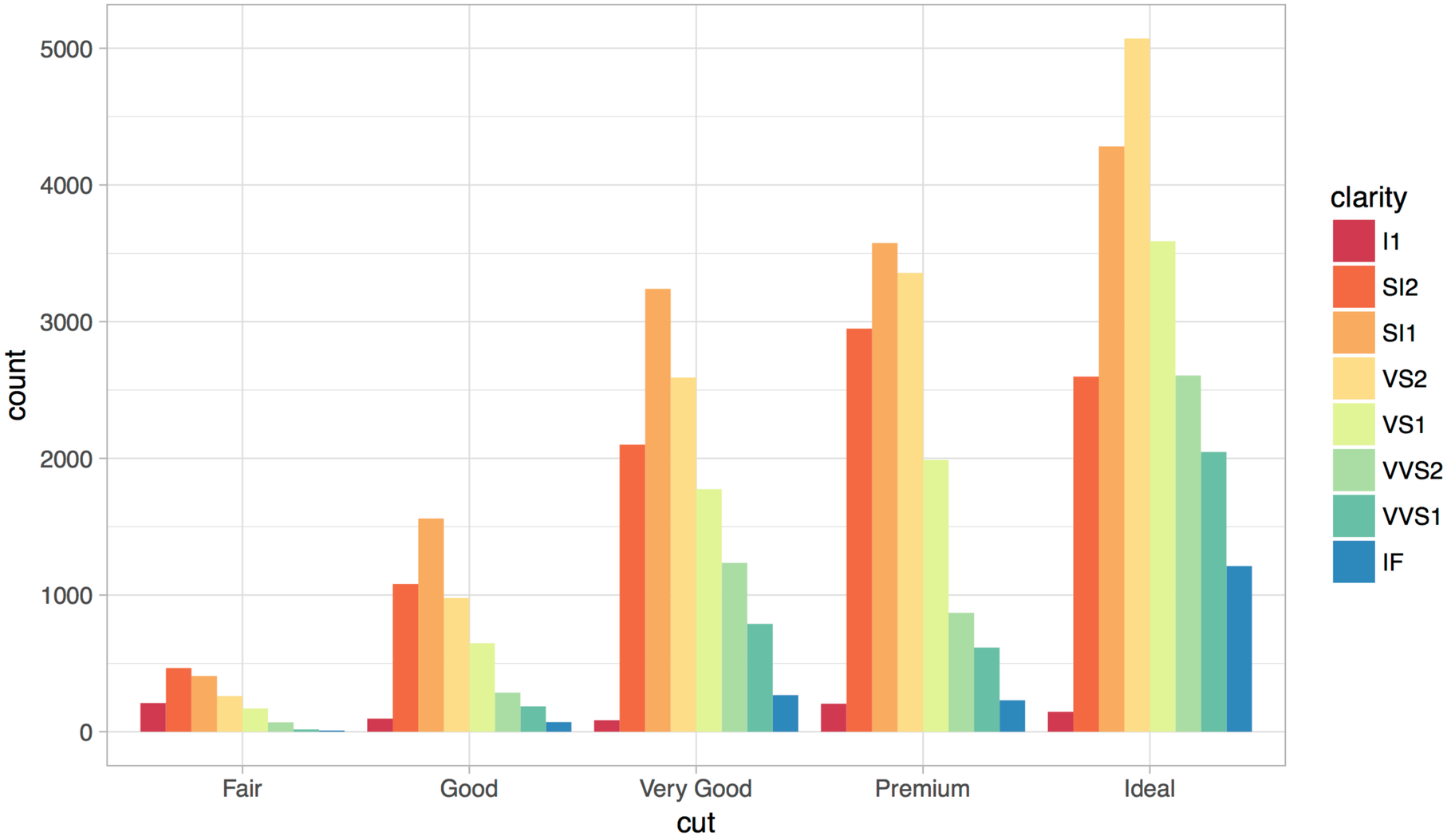
Themes

Visual appearance of non-data elements



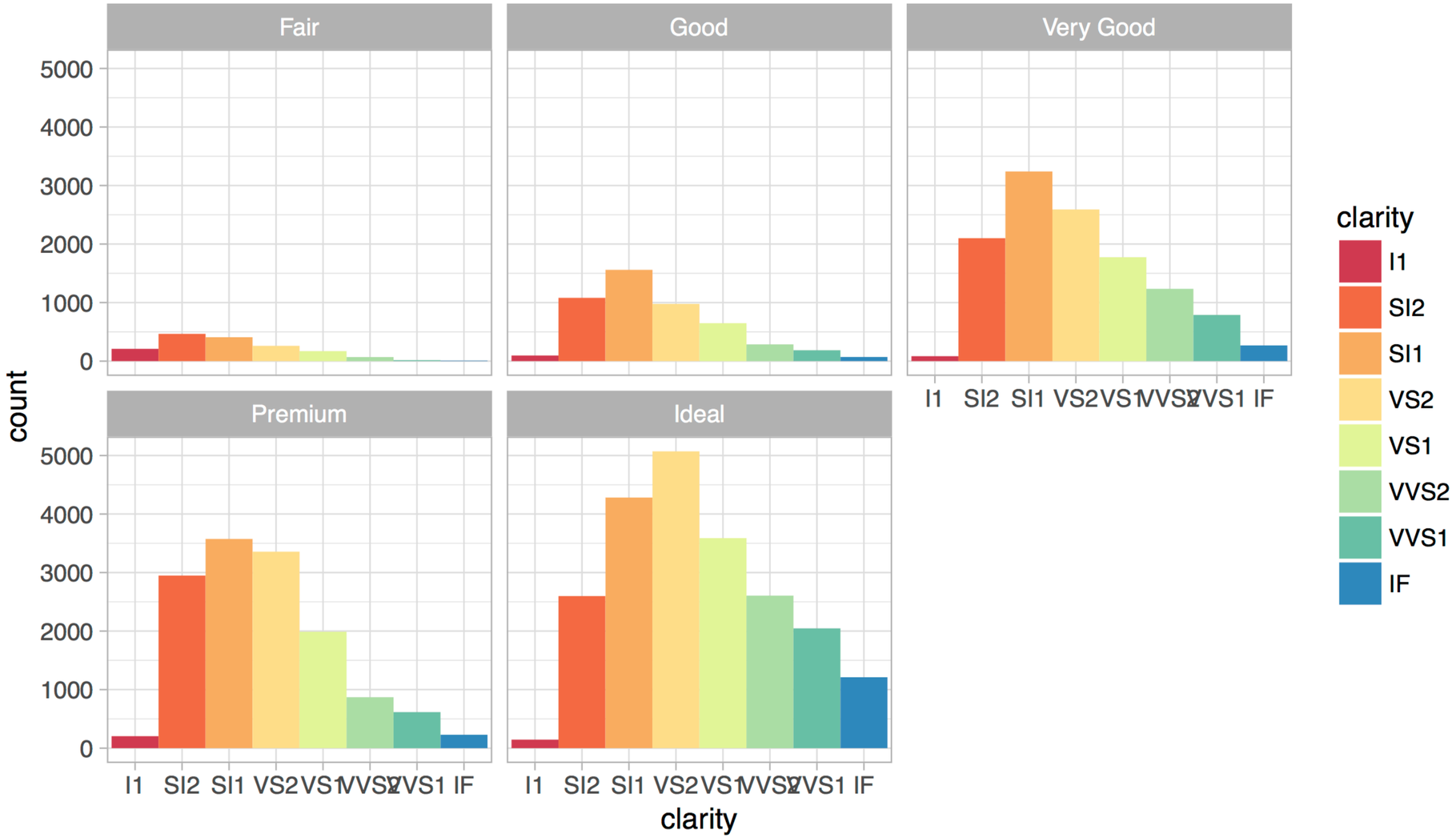
Scales

Customize color scales, other mappings

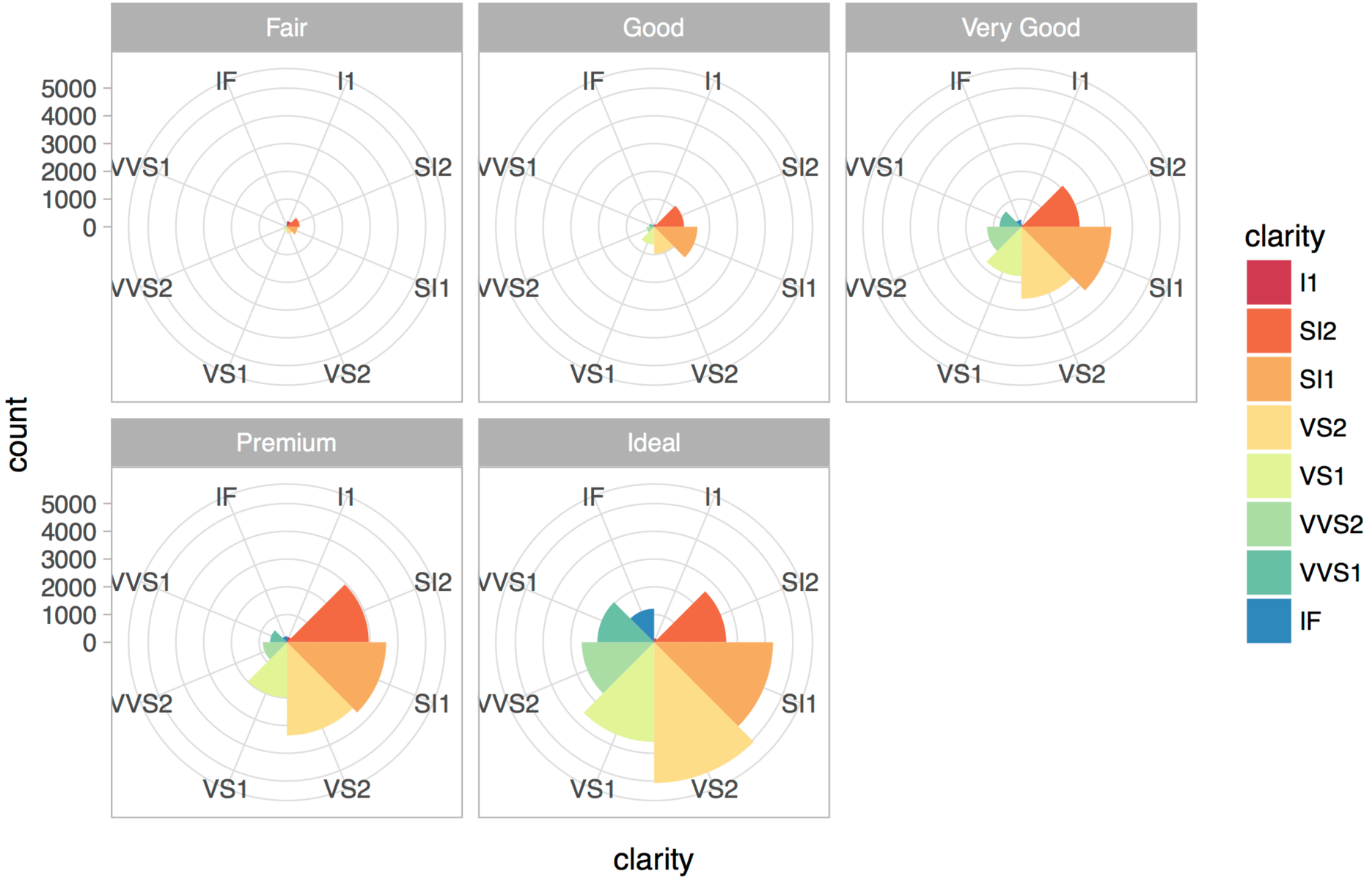


Facets

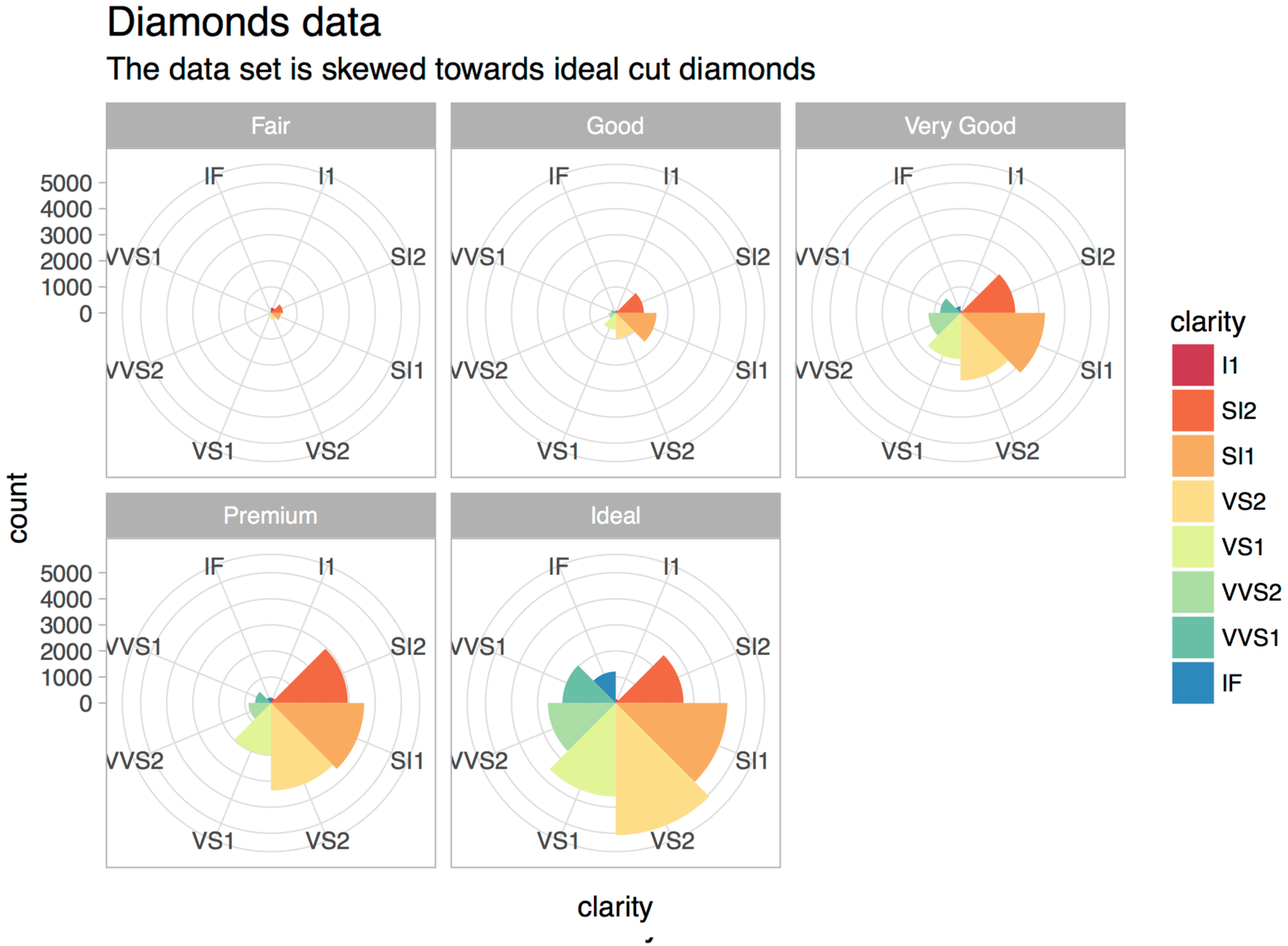
Subplots that display subsets of the data.



Coordinate systems



Titles and captions



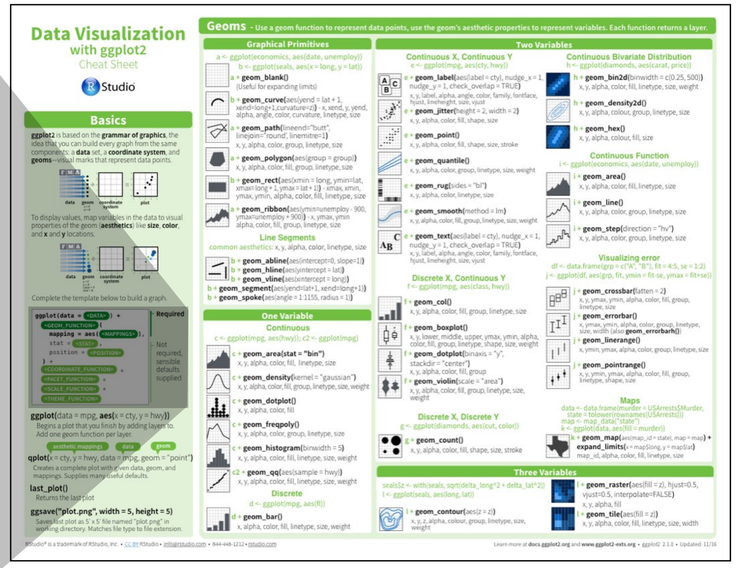
A ggplot2 template

Make any plot by filling in the parameters of this template

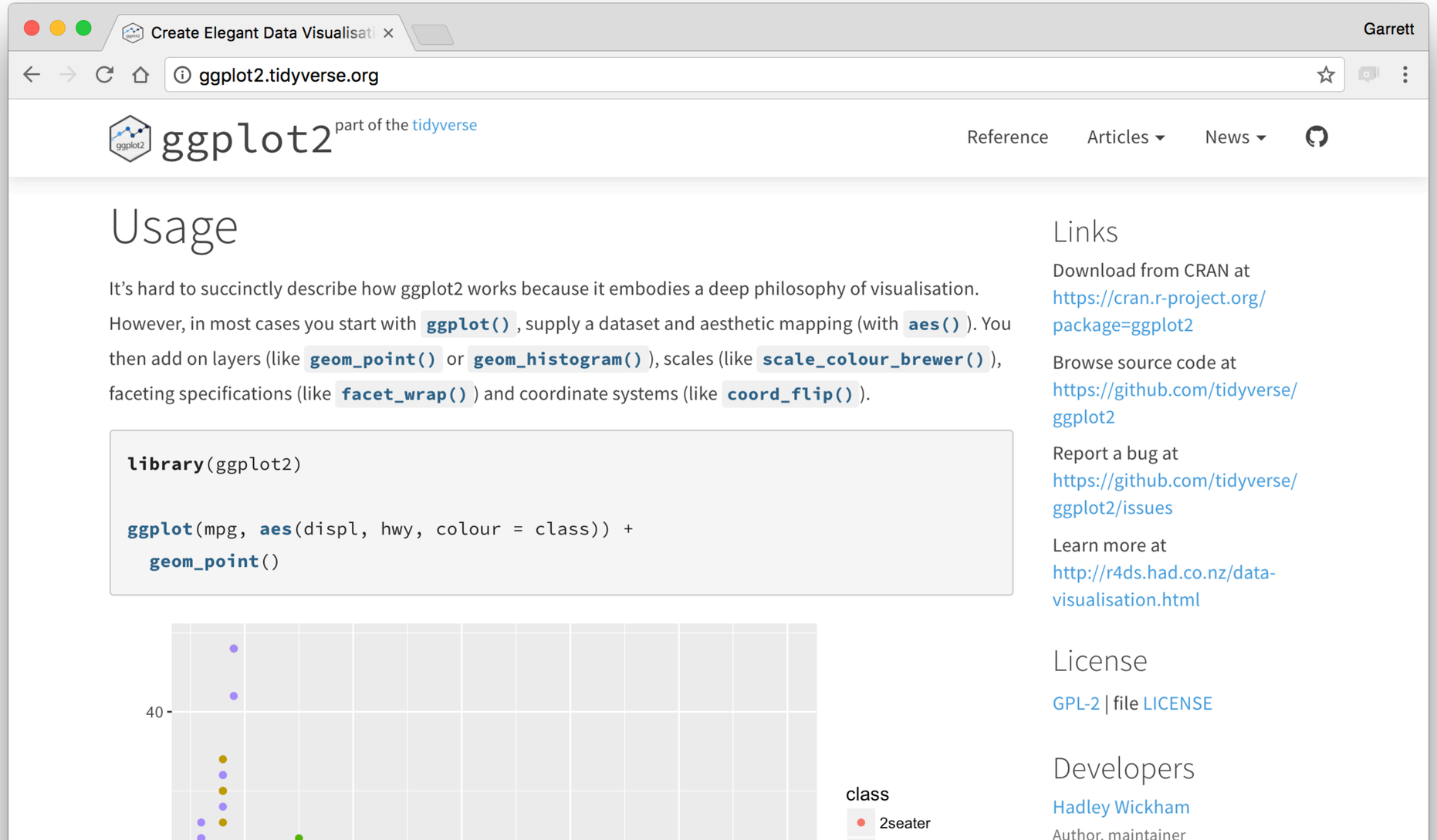
```
ggplot(data = <DATA>) +
  <GEOM_FUNCTION>(
    mapping = aes(<MAPPINGS>),
    stat = <STAT>,
    position = <POSITION>
  ) +
  <COORDINATE_FUNCTION> +
  <FACET_FUNCTION> +
  <SCALE_FUNCTION> +
  <THEME_FUNCTION>
```

Required

Not required, sensible defaults supplied



ggplot2.tidyverse.org



Visualize Data with