Diving into `ggplot2`

Code

Instructor - Soumya Mukherjee

Content Credit- Dr. Matthew Beckman and Olivia Beck

July 19, 2023

Agenda

- brief introduction to `::`
- `ggplot()` flyover
- remarks about facets
- `color` vs. `fill`

PackageName::FunctionName

- We can use the `::` function to reference functions inside packages.
    - This helps us be extra sure that we are using the exact function we want to be
    - Helps avoid conflict (i.e. when 2 packages that have a function with the same name that do different things)

Hide

```
library(tidyverse)
head(diamonds)
```

carat <dbl>	cut <ord>	color <ord>	clarity <ord>	depth <dbl>	table <dbl>	price <int>	x <dbl>	y <dbl>	z <dbl>
0.23	Ideal	E	SI2	61.5	55	326	3.95	3.98	2.43
0.21	Premium	E	SI1	59.8	61	326	3.89	3.84	2.31
0.23	Good	E	VS1	56.9	65	327	4.05	4.07	2.31
0.29	Premium	I	VS2	62.4	58	334	4.20	4.23	2.63

carat <dbl>	cut <ord>	color <ord>	clarity <ord>	depth <dbl>	table <dbl>	price <int>	x <dbl>	y <dbl>	z <dbl>
0.31	Good	J	SI2	63.3	58	335	4.34	4.35	2.75
0.24	Very Good	J	VVS2	62.8	57	336	3.94	3.96	2.48

6 rows

Hide

```
#?filter

#filter from dplyr
diamonds %>%
  dplyr::filter(color == "E") %>%
  head()
```

carat <dbl>	cut <ord>	color <ord>	clarity <ord>	depth <dbl>	table <dbl>	price <int>	x <dbl>	y <dbl>	z <dbl>
0.23	Ideal	E	SI2	61.5	55	326	3.95	3.98	2.43
0.21	Premium	E	SI1	59.8	61	326	3.89	3.84	2.31
0.23	Good	E	VS1	56.9	65	327	4.05	4.07	2.31
0.22	Fair	E	VS2	65.1	61	337	3.87	3.78	2.49
0.20	Premium	E	SI2	60.2	62	345	3.79	3.75	2.27
0.32	Premium	E	I1	60.9	58	345	4.38	4.42	2.68

6 rows

Hide

```
#normal
diamonds %>%
  filter(color == "E") %>%
  head()
```

carat <dbl>	cut <ord>	color <ord>	clarity <ord>	depth <dbl>	table <dbl>	price <int>	x <dbl>	y <dbl>	z <dbl>
0.23	Ideal	E	SI2	61.5	55	326	3.95	3.98	2.43
0.21	Premium	E	SI1	59.8	61	326	3.89	3.84	2.31
0.23	Good	E	VS1	56.9	65	327	4.05	4.07	2.31
0.22	Fair	E	VS2	65.1	61	337	3.87	3.78	2.49
0.20	Premium	E	SI2	60.2	62	345	3.79	3.75	2.27
0.32	Premium	E	I1	60.9	58	345	4.38	4.42	2.68

6 rows

Hide

```
NA
```

Building Graphics

1. Draw by hand (or imagine) the specific plot that you intend to construct
2. Data Wrangling (if needed) to get the data in glyph-ready form, or verify that the current form is glyph-ready for your purposes.
3. Establish the frame using a `ggplot()` statement
4. Create the intended glyph using `geom_[style]()` such as
    - `geom_point()`
    - `geom_bar()`
    - `geom_boxplot()`
    - `geom_density()`
    - `geom_vline()`

    - `geom_segment()`
    - `geom_histogram()`
    - and *many* more
5. Map variables to the graphical attributes of the glyph using: `aes( )`

- Rule of thumb: anytime when you are plotting with ggplot, ALL variables need to be inside an `aes` (except facets, later in slides).

6. Add additional layers to the frame using the `+` symbol
    - Note: **not** `%>%` between layers of `ggplot2` graphics
    - Maybe think “add layer” in `ggplot2` portions, instead of “and then” with `%>%` syntax

Steps 4 and 5 can be switched.

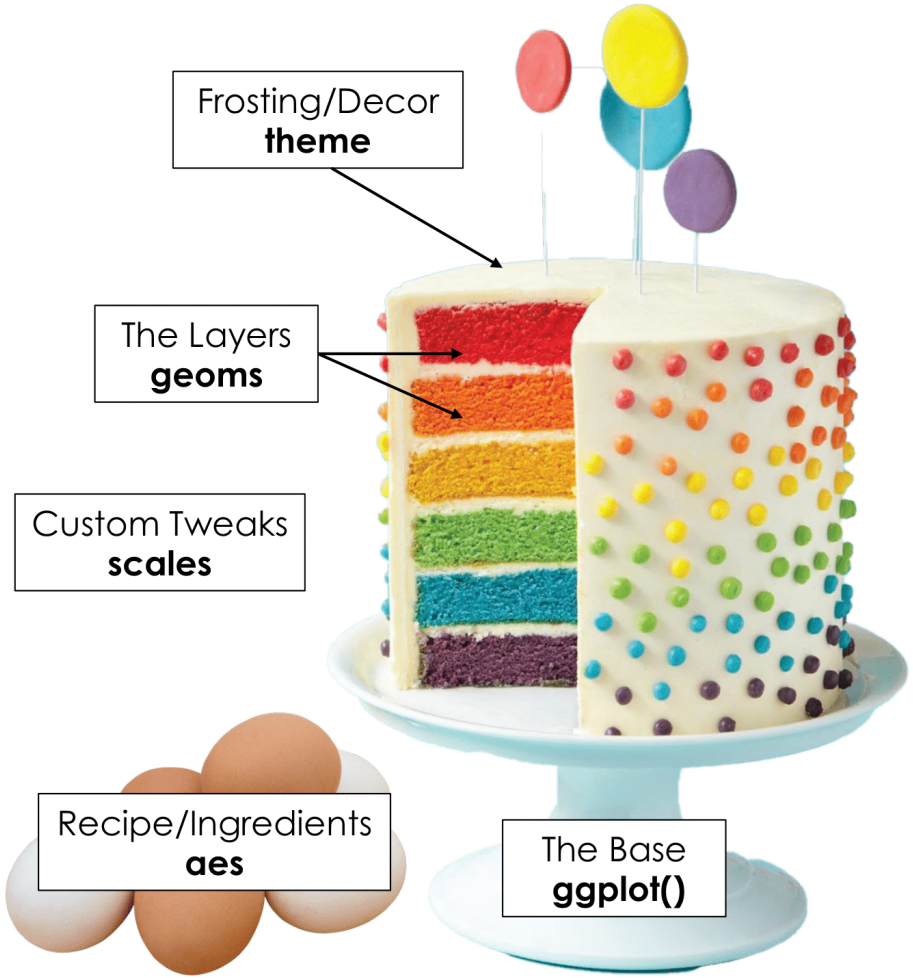
ggplot is a little bit like cake...

We *always* start by setting up the foundation with **ggplot()**

We specify our ingredients (data variables) with an **aes mapping**

We can create *layers* to our plot with **geoms**

We can style our ~~cake~~ ggplot with **themes.** We have out-of-the-box options, or we can go totally custom!



https://twitter.com/tanya_shapiro/status/1576935152575340544?t=vwaW8h6CC62h0pkwv9n5Yg&s=19
(https://twitter.com/tanya_shapiro/status/1576935152575340544?t=vwaW8h6CC62h0pkwv9n5Yg&s=19)

Example: Baby Names

Let's look at our `BabyNames` names data set agian.

Hide

```
# data intake
data("BabyNames", package = "dcData")

# inspect data intake
glimpse(BabyNames)
```

```
Rows: 1,792,091
Columns: 4
$ name  <chr> "Mary", "Anna", "Emma", "Elizabeth", "Minnie", "Mar…
$ sex   <chr> "F", "F", "F", "F", "F", "F", "F", "F", "F", "F", "…
$ count <int> 7065, 2604, 2003, 1939, 1746, 1578, 1472, 1414, 132…
$ year  <int> 1880, 1880, 1880, 1880, 1880, 1880, 1880, 1880, 188…
```

wrangle into glyph-ready form

Hide

```
names <- c("Olivia", "Zoe", "Quentin")

Names <-
  BabyNames %>%
  filter(name %in% names) %>%
  group_by(name, year) %>%
  summarise(total = sum(count, na.rm = TRUE))
```

```
`summarise()` has grouped output by 'name'. You can override using the `.groups` argument.
```

Hide

```
Names %>%
  head()
```

name <chr>	year <int>	total <int>
Olivia	1880	44
Olivia	1881	51
Olivia	1882	52
Olivia	1883	46
Olivia	1884	54
Olivia	1885	59

6 rows

Hide

```
NA
```

in the beginning you might use esquisser to get started–here's the default result

This isn't easy to read, and it's in bad form.

Hide

```
# esquisser(Names)
# ggplot(data = Names, aes(x = year, y = total)) + geom_line()  + aes(colour = name) + theme(legend.position = "right") + la
bs(title = "")
ggplot(Names) +
  aes(x = year, y = total, colour = name) +
  geom_line() +
  scale_color_hue(direction = 1) +
  theme_gray()
```

we can do better

1. establish the frame
2. plot the glyphs (i.e., select a geom)
3. map the aesthetics
4. add labels and title
5. other features (e.g., alpha, sizing, etc)

Our Plot

1. Establish the Frame

Nothing is here! That is exactly what is supposed to happen. Calling `ggplot()` only tells us R that we are ready to plot and I want to call some space to create my plot.

Hide

```
ggplot(data = Names)
```

Hide

```
NA
```

2. plot the glyphs (i.e., select a geom)

Still Nothing! We need to tell it what our axis are.

Note that ggplot uses `+`, NOT `%>%`. This is because we are **adding** layers to our plots.

Hide

```
ggplot(data = Names) +
  geom_line()
```

```
Error in `geom_line()`:
! Problem while setting up geom.
ℹ Error occurred in the 1st layer.
Caused by error in `compute_geom_1()`:
! `geom_line()` requires the following missing aesthetics:
  x and y
Backtrace:
  1. base (local) `<fn>`(x)
  2. ggplot2:::print.ggplot(x)
  4. ggplot2:::ggplot_build.ggplot(x)
  5. ggplot2:::by_layer(...)
 12. ggplot2 (local) f(l = layers[[i]], d = data[[i]])
 13. l$compute_geom_1(d)
 14. ggplot2 (local) compute_geom_1(..., self = self)
```

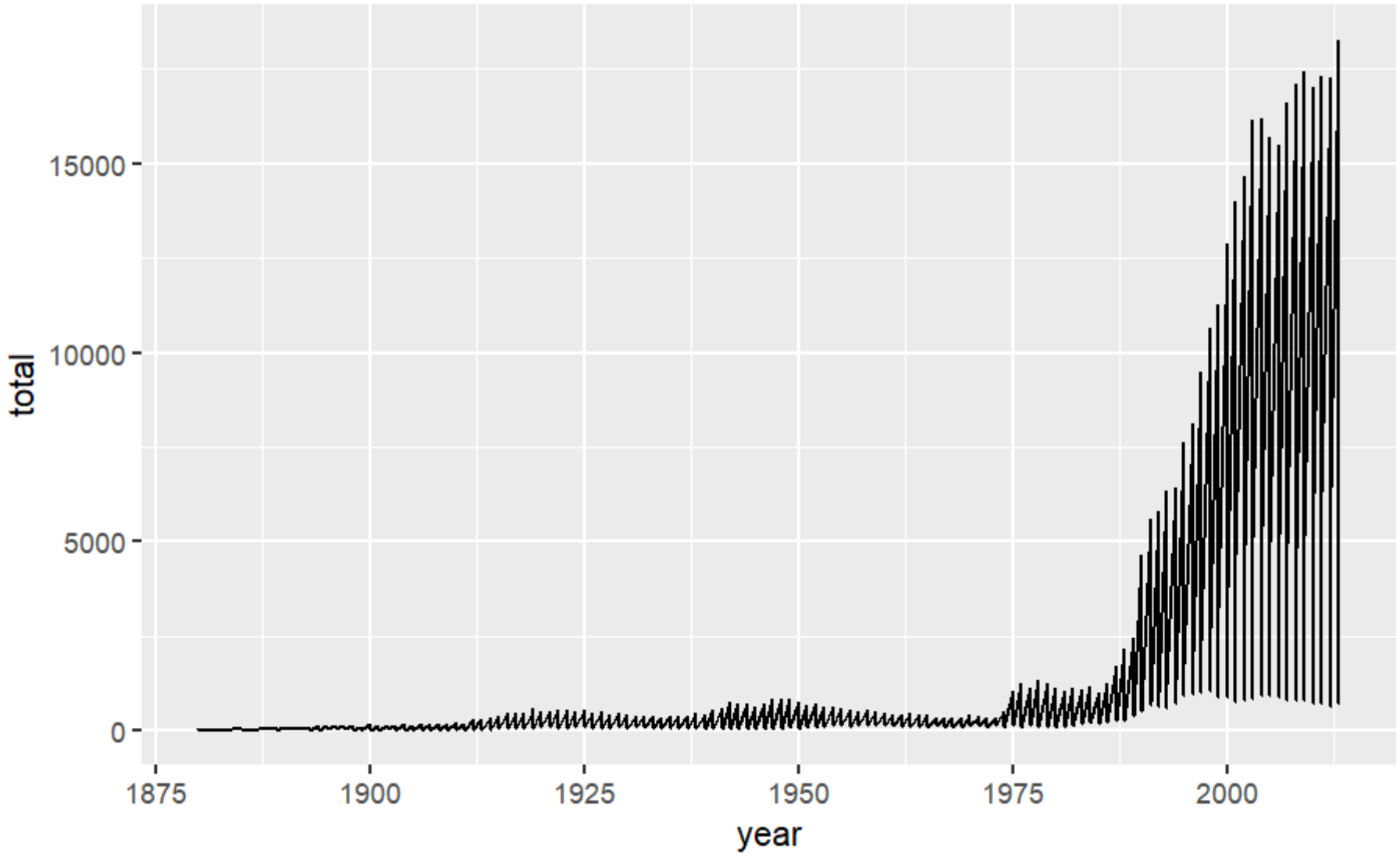
Note - this is why I like to map aesthetics first, so we can avoid errors.

3. Map the aesthetics

Rule of thumb: anytime when you are plotting with ggplot, ALL variables need to be inside an `aes` (except facets, later in slides).

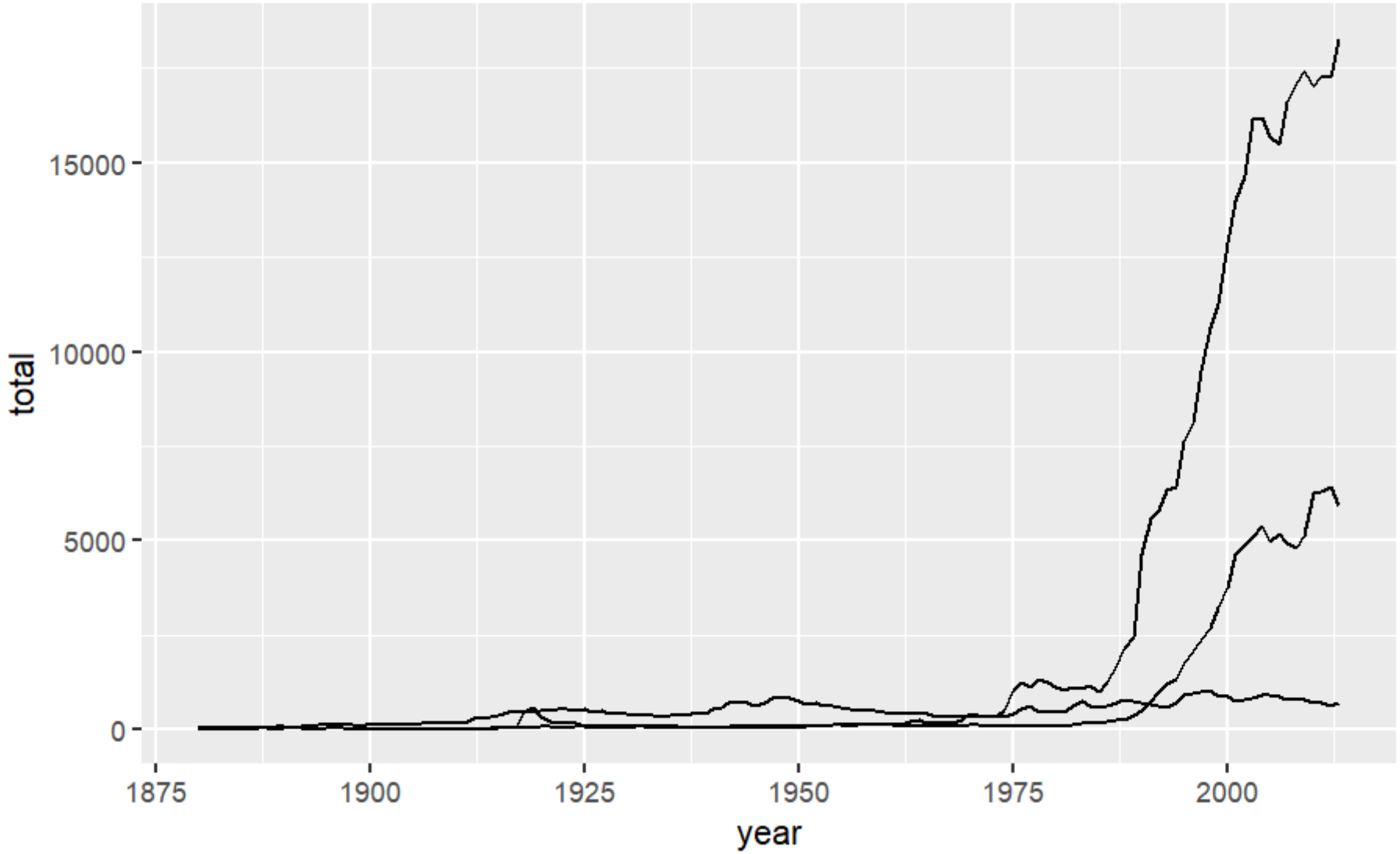
Hide

```
#not Quite
ggplot(data = Names) +
  geom_line( aes(x = year, y = total))
```



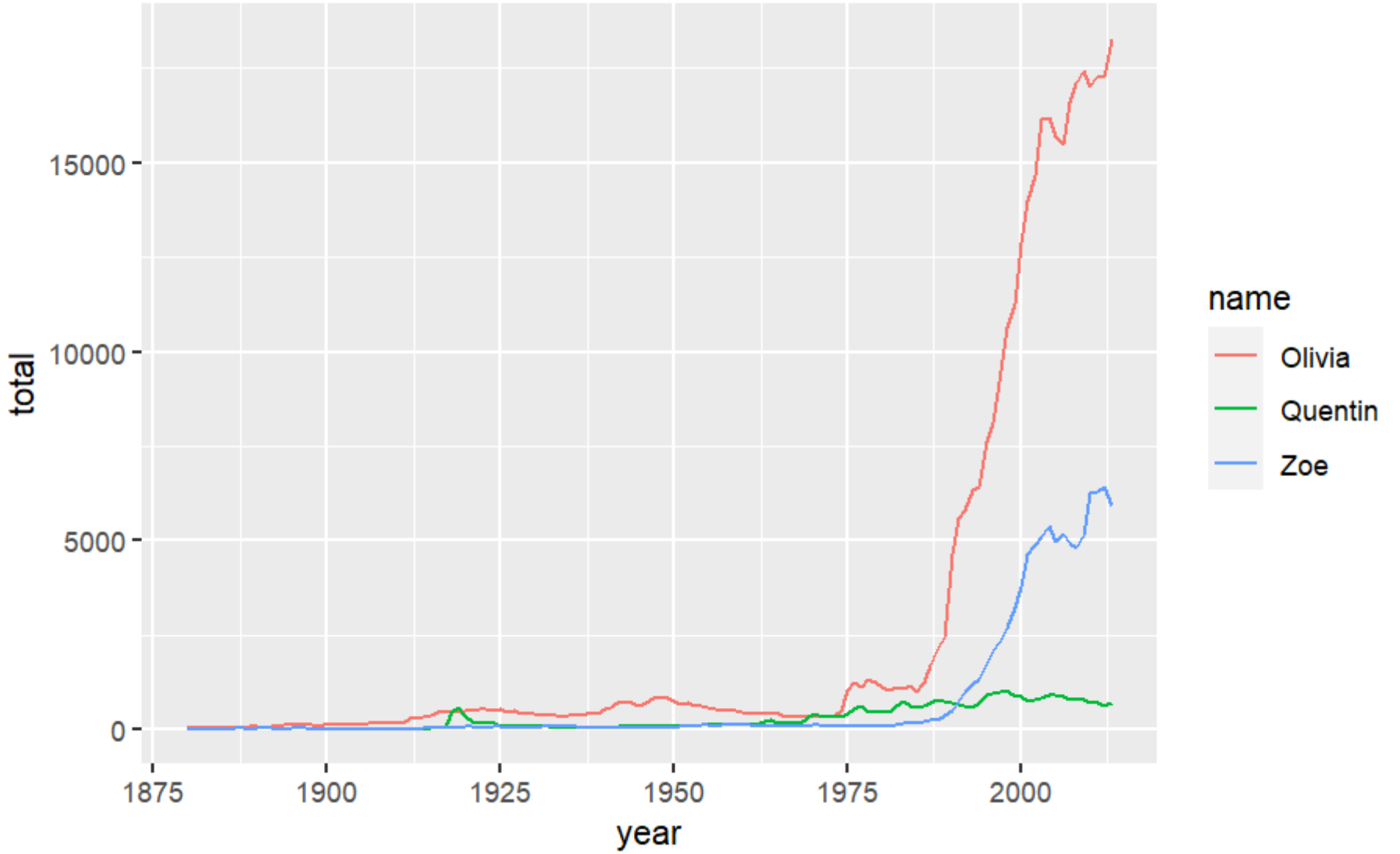
Hide

```
#add groups
ggplot(data = Names) +
  geom_line( aes(x = year, y = total, group = name))
```



Hide

```
#add color
ggplot(data = Names) +
  geom_line( aes(x = year, y = total, color = name))
```



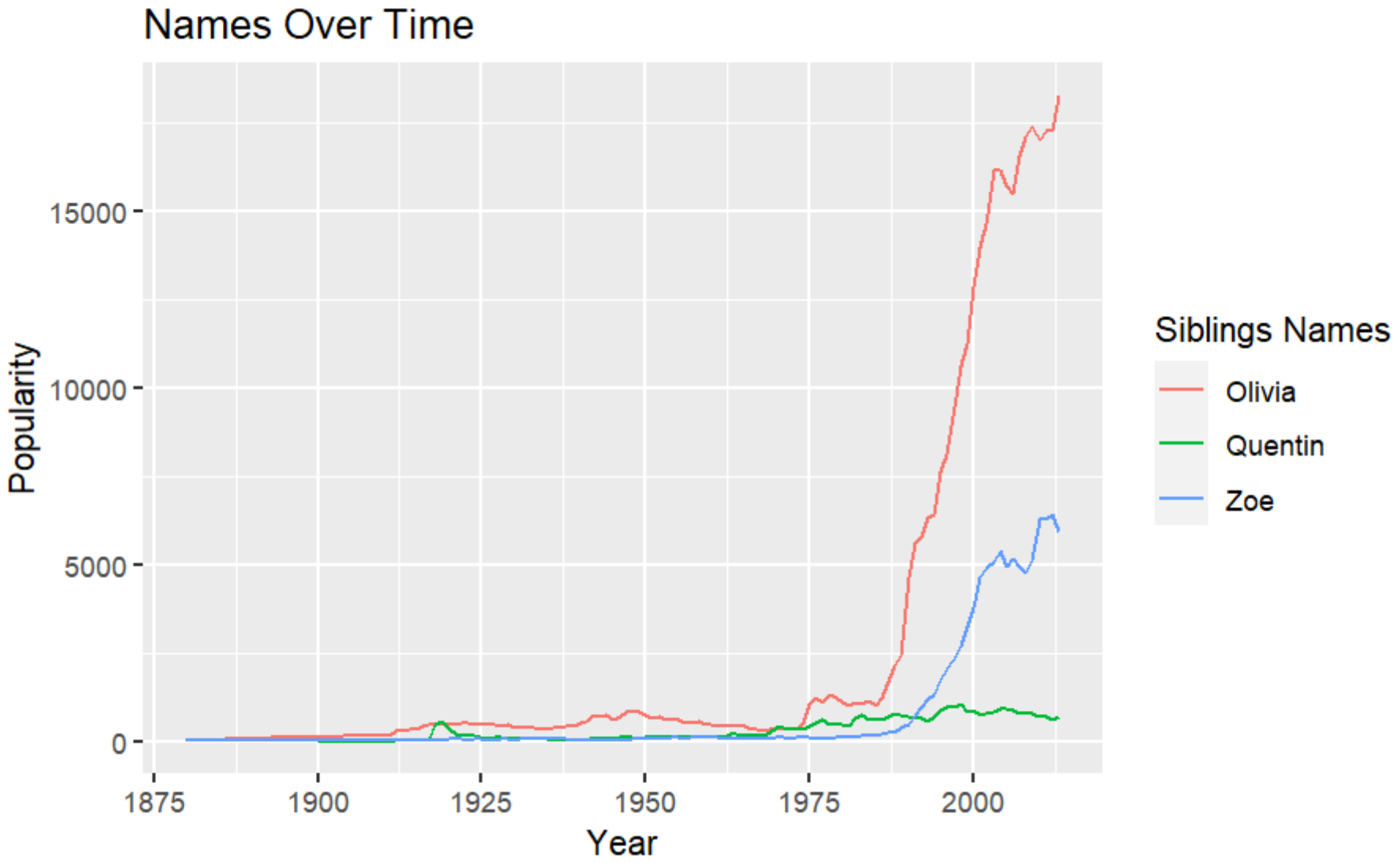
Hide

```
NA
NA
```

4. Add labels and title

Hide

```
ggplot(data = Names) +
  geom_line( aes(x = year, y = total, color = name)) +
  ggtitle("Names Over Time") +
  xlab("Year") +
  ylab("Popularity") +
  guides(color = guide_legend(title = "Siblings Names" ))
```



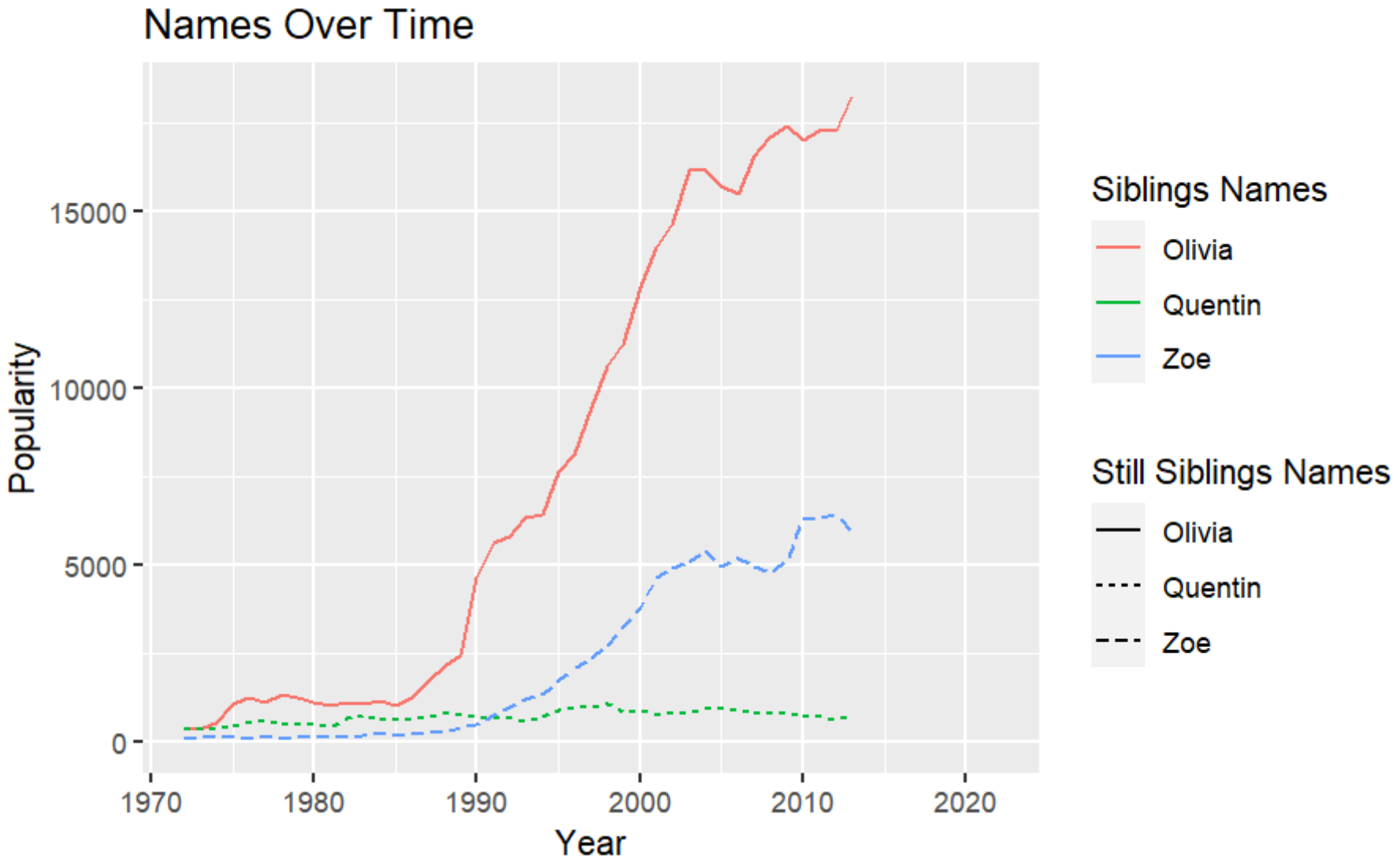
Hide

NA
NA

5. other features (e.g., alpha, sizing, etc)

Hide

```
ggplot(data = Names) +
  geom_line( aes(x = year, y = total, color = name, linetype = name)) +
  ggtitle("Names Over Time") +
  xlim(c(1972, 2022))+
  xlab("Year") +
  ylab("Popularity") +
  guides(color = guide_legend(title = "Siblings Names" ),
         linetype = guide_legend(title = "Still Siblings Names" ))
```


Names Over Time
Popularity
15000
10000
5000
0
1970
1980
1990
2000
2010
2020
Year
Siblings Names
Olivia
Quentin
Zoe
Still Siblings Names
Olivia
Quentin
Zoe

Hide

NA
NA

Remarks about faceting: `facet_wrap()`

The syntax for facets requires a formula syntax we haven't seen much yet. Also, there are two main ways to plot with facets. Here are a few pointers:

- `facet_wrap()` just makes a box for each level of the categorical variable
  - Syntax: `facet_wrap( ~ categoricalVariable)`
  - For example:

Hide

```
data("NCHS")

# 1is.na(smoker) gets cases that are non-missing for `smoker` (i.e. removes NA's)
Heights <-
  NCHS %>%
  filter(age > 20, !is.na(smoker)) %>%
  group_by(sex, smoker, age) %>%
  summarise(height = mean(height, na.rm = TRUE))
```

```
`summarise()` has grouped output by 'sex', 'smoker'. You can override using the `.groups` argument.
```

Hide

```
head(Heights)
```

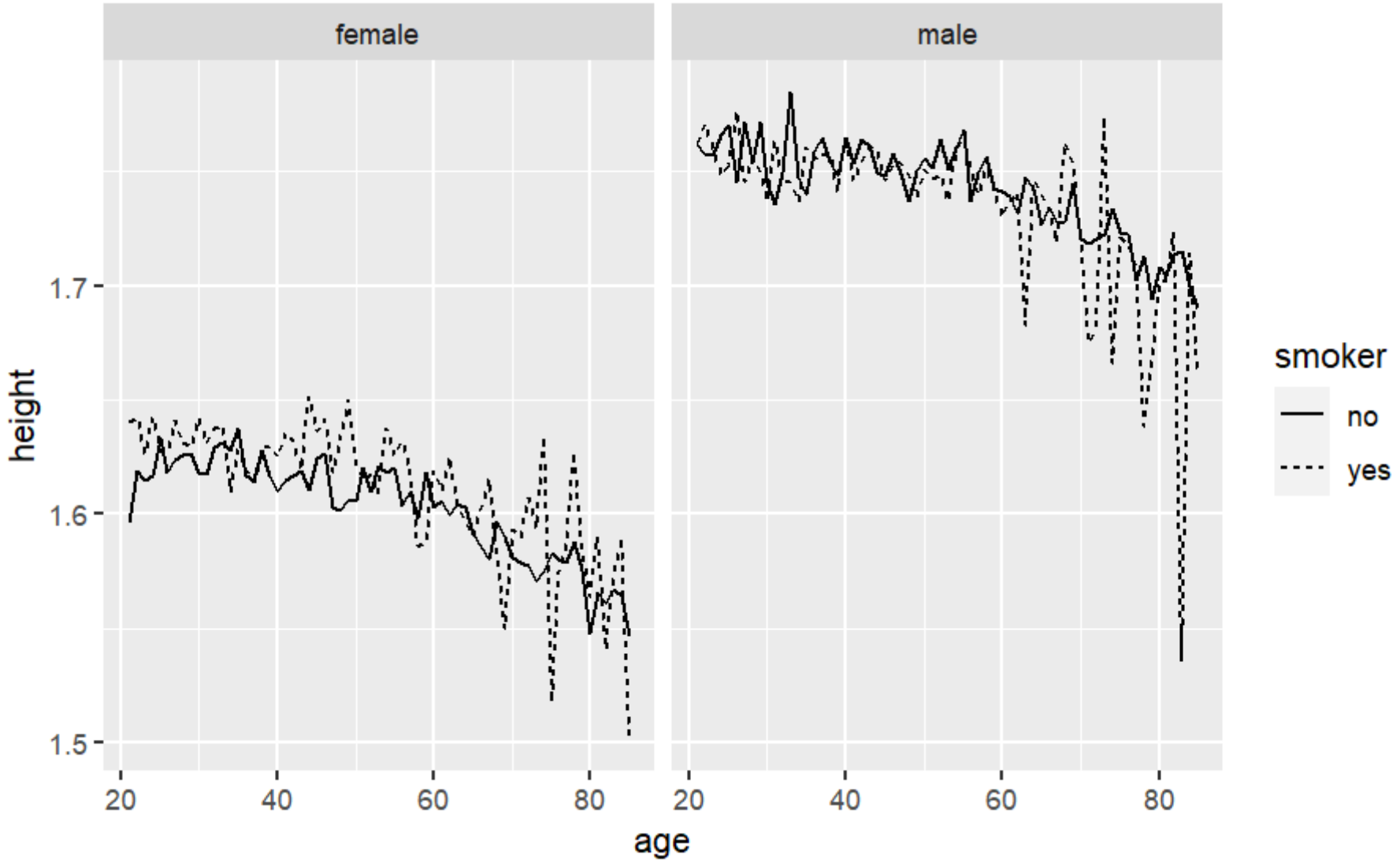
sex <fctr>	**smoker** <fctr>	**age** <dbl>	**height** <dbl>
female	no	21	1.595759
female	no	22	1.618918
female	no	23	1.614600
female	no	24	1.616612
female	no	25	1.633018

sex	smoker	age	height
<fctr>	<fctr>	<dbl>	<dbl>
female	no	26	1.618016

6 rows

Hide

```
Heights %>%
  ggplot(aes(x = age, y = height)) +
  geom_line(aes(linetype = smoker)) +
  facet_wrap( ~ sex)
```



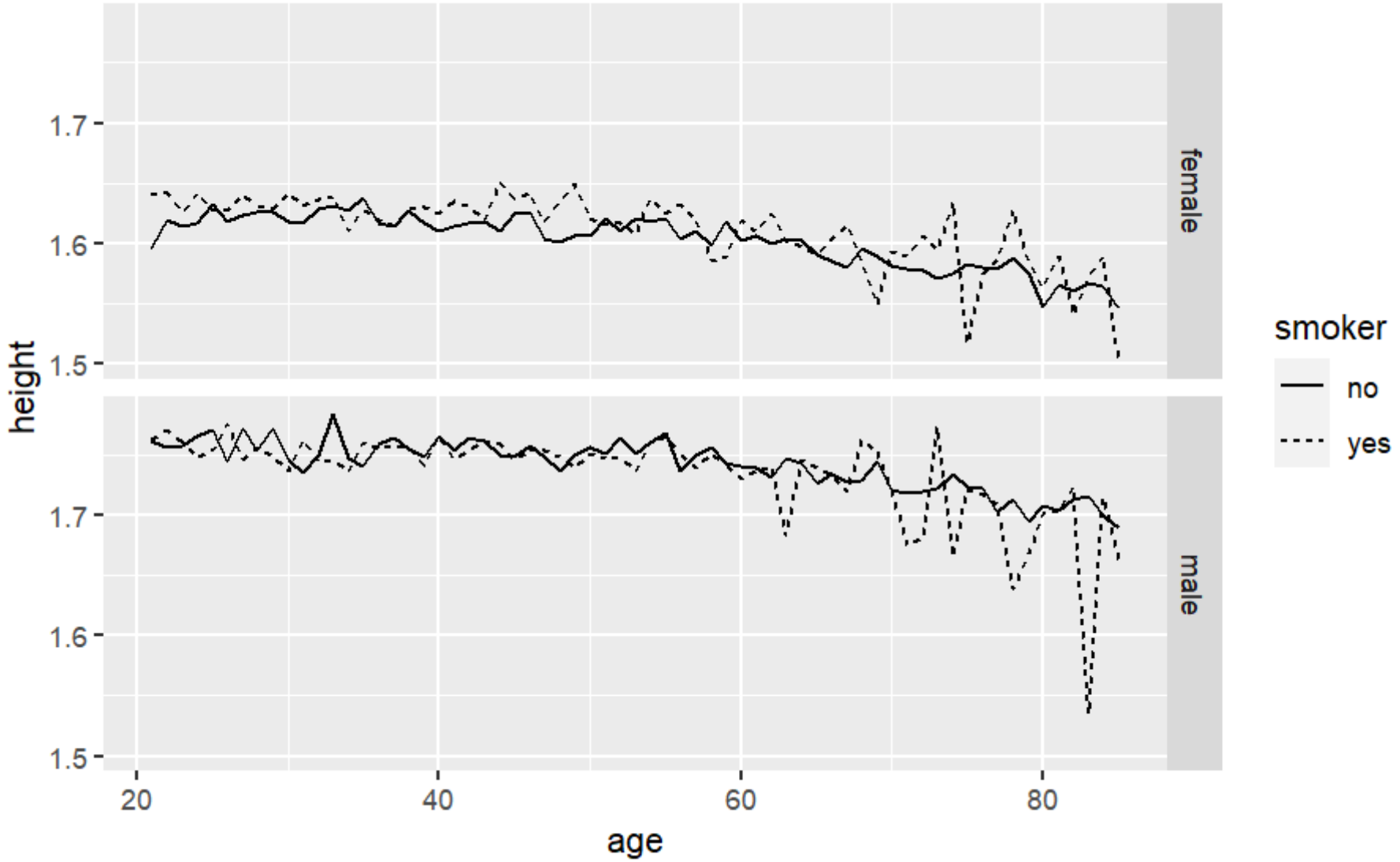
Remarks about faceting: `facet_grid()`

- `facet_grid()` allows control of row & column facets
- `facet_grid()` syntax:
  - row & column facets: `facet_grid(rows ~ cols)`
  - row facets only: `facet_grid( rows ~ . )` (note the required " . ")

- column facets only: `facet_grid( ~ cols)` (no " `.` " this time)

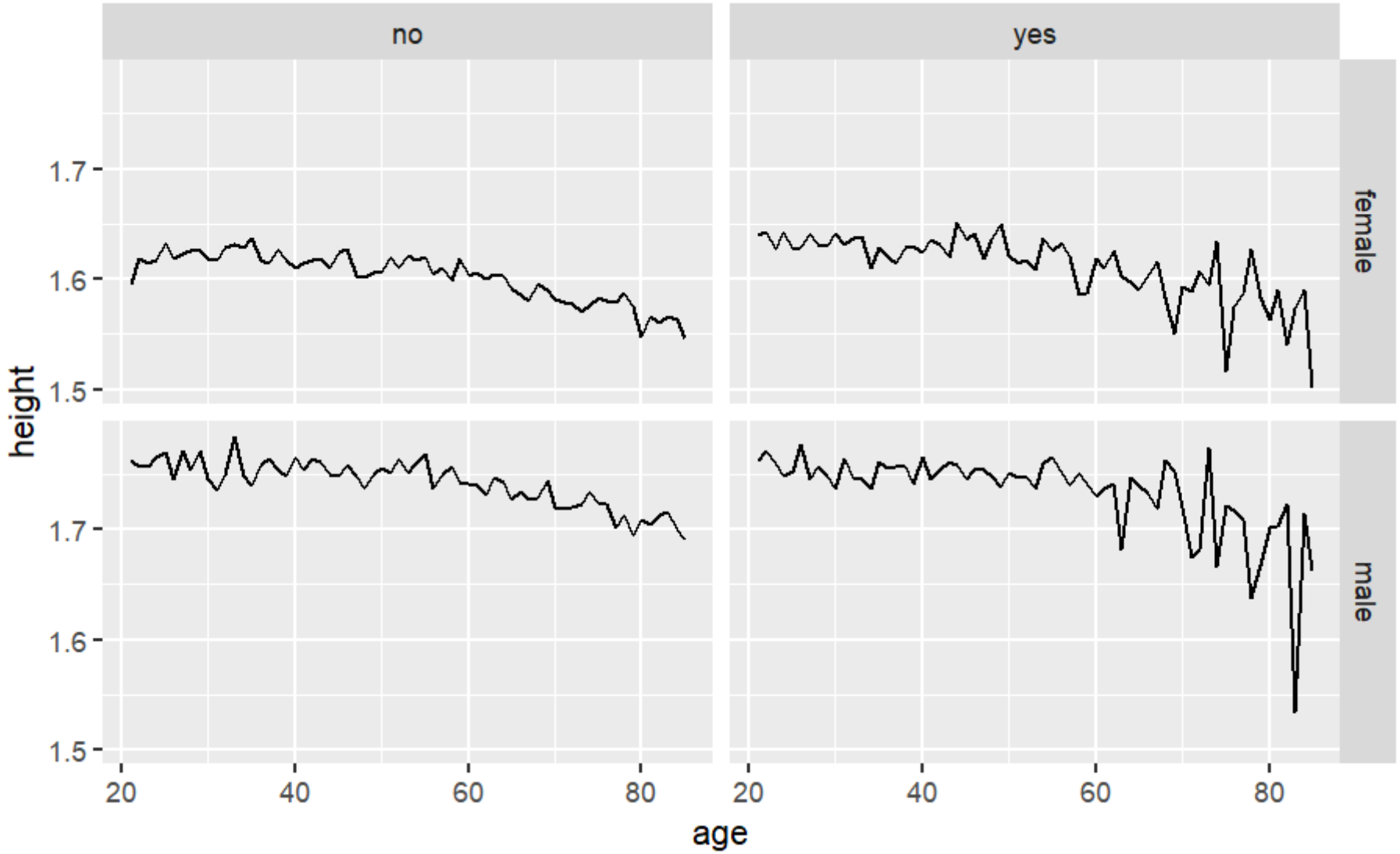
Hide

```
Heights %>%
  ggplot(aes(x = age, y = height)) +
  geom_line(aes(linetype = smoker)) +
  facet_grid(sex ~ .)
```



Hide

```
Heights %>%
  ggplot(aes(x = age, y = height)) +
  geom_line() +
  facet_grid(sex ~ smoker)
```



Difference between `color` and `fill`

Hide

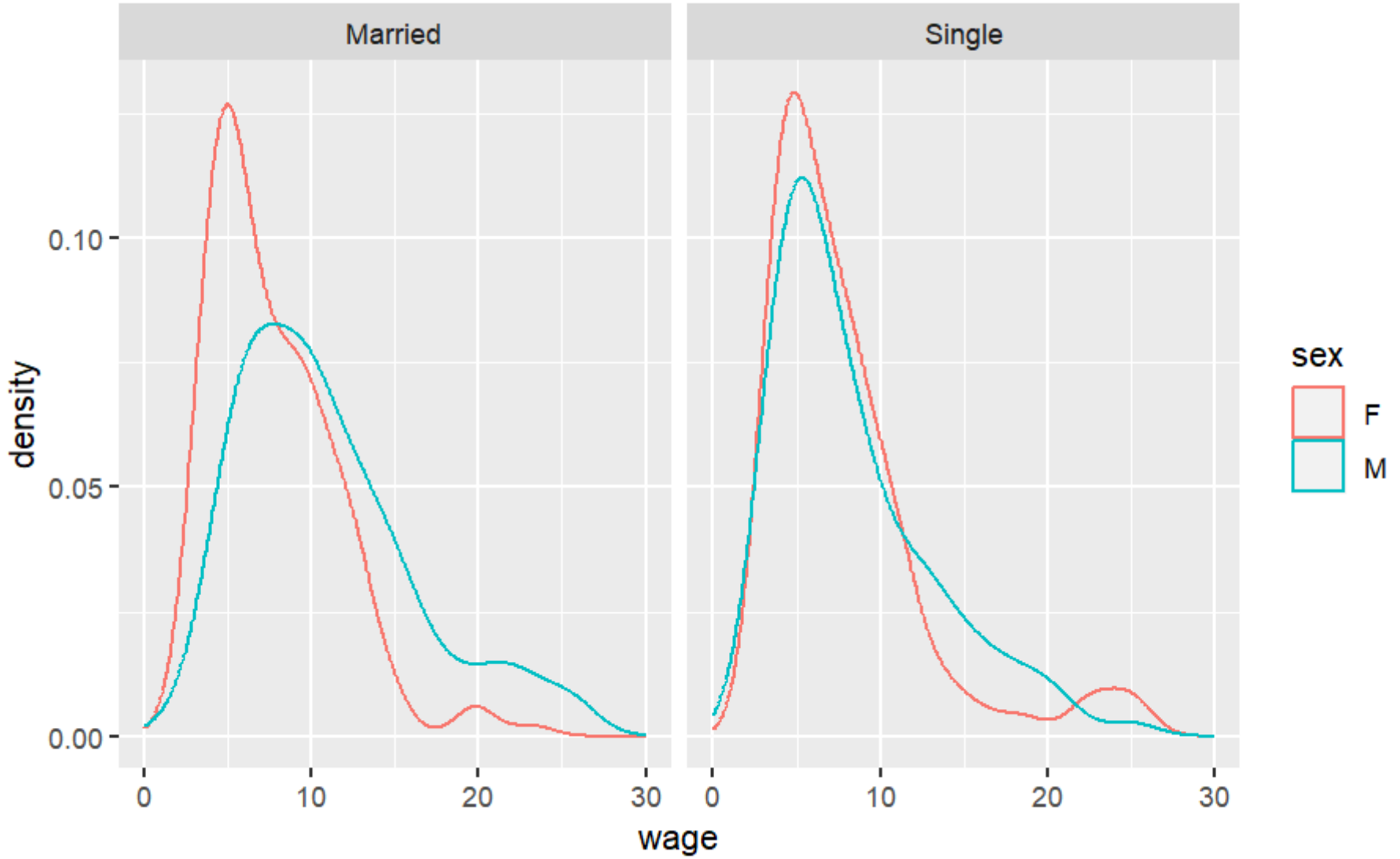
```
library(mosaicData)

head(CPS85)
```

	wage <dbl>	educ <int>	race <fctr>	sex <fctr>	hispanic <fctr>	south <fctr>	married <fctr>	exper <int>	union <fctr>
1	9.0	10	W	M	NH	NS	Married	27	Not
2	5.5	12	W	M	NH	NS	Married	20	Not
3	3.8	12	W	F	NH	NS	Single	4	Not
4	10.5	12	W	F	NH	NS	Married	29	Not
5	15.0	12	W	M	NH	NS	Married	40	Union
6	9.0	16	W	F	NH	NS	Married	27	Not

6 rows | 1-10 of 11 columns

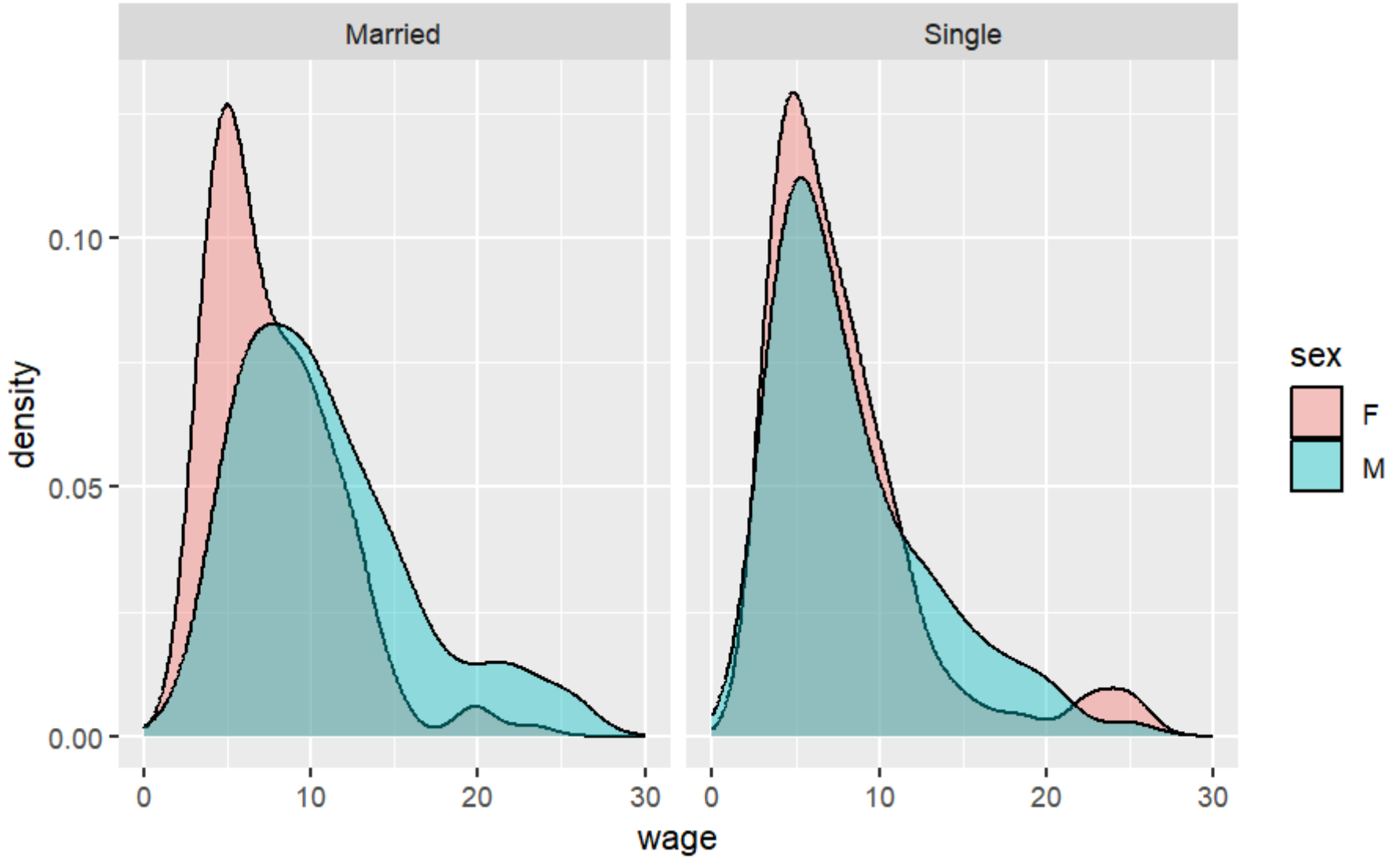
Hide

```
CPS85 %>%
  ggplot() +
  geom_density(aes(x = wage, color = sex), alpha = 0.4)+
  facet_grid( ~ married) +
  xlim(0,30)
```



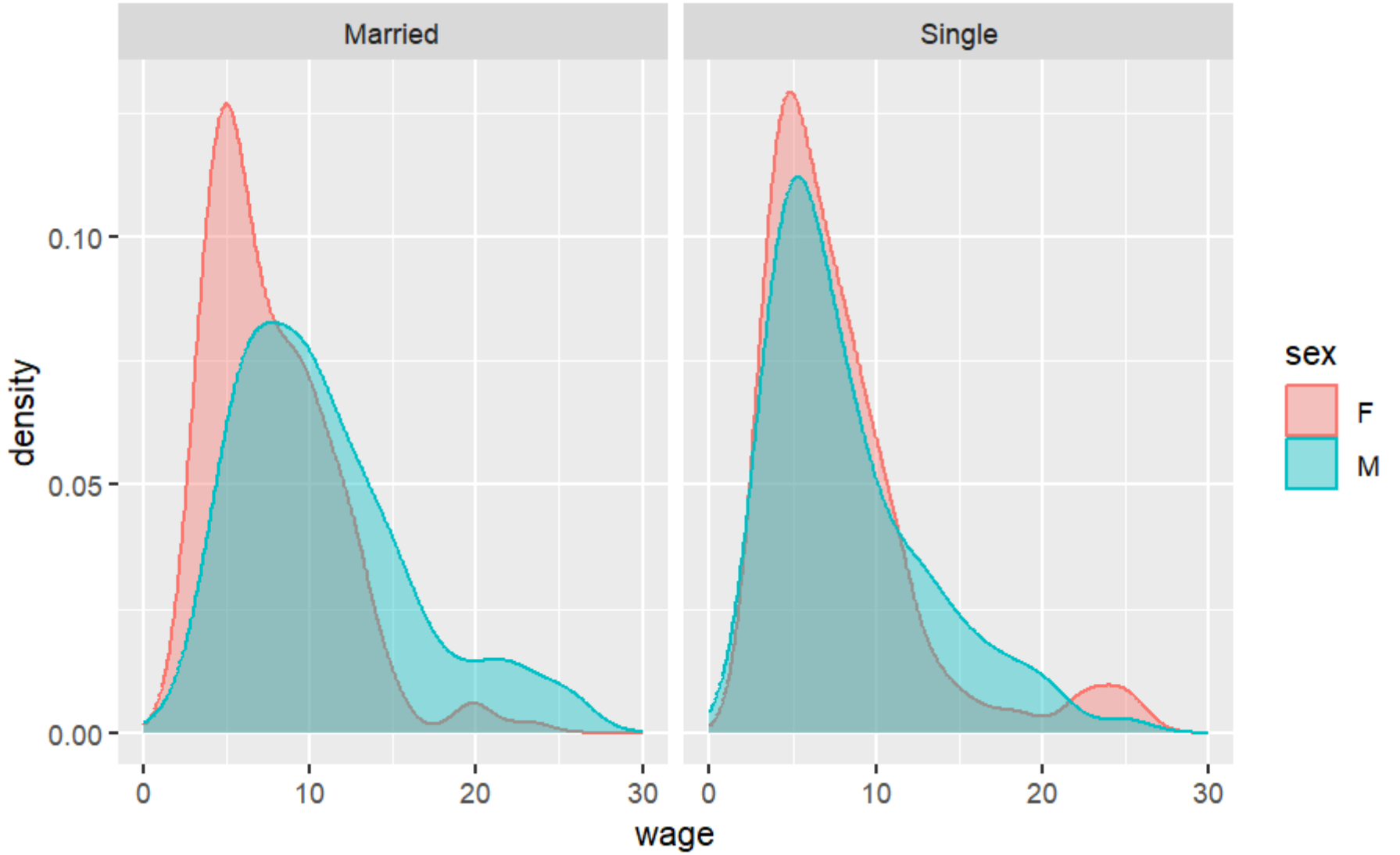
Hide

```
CPS85 %>%
  ggplot() +
  geom_density(aes(x = wage, fill = sex), alpha = 0.4)+
  facet_grid( ~ married) +
  xlim(0,30)
```



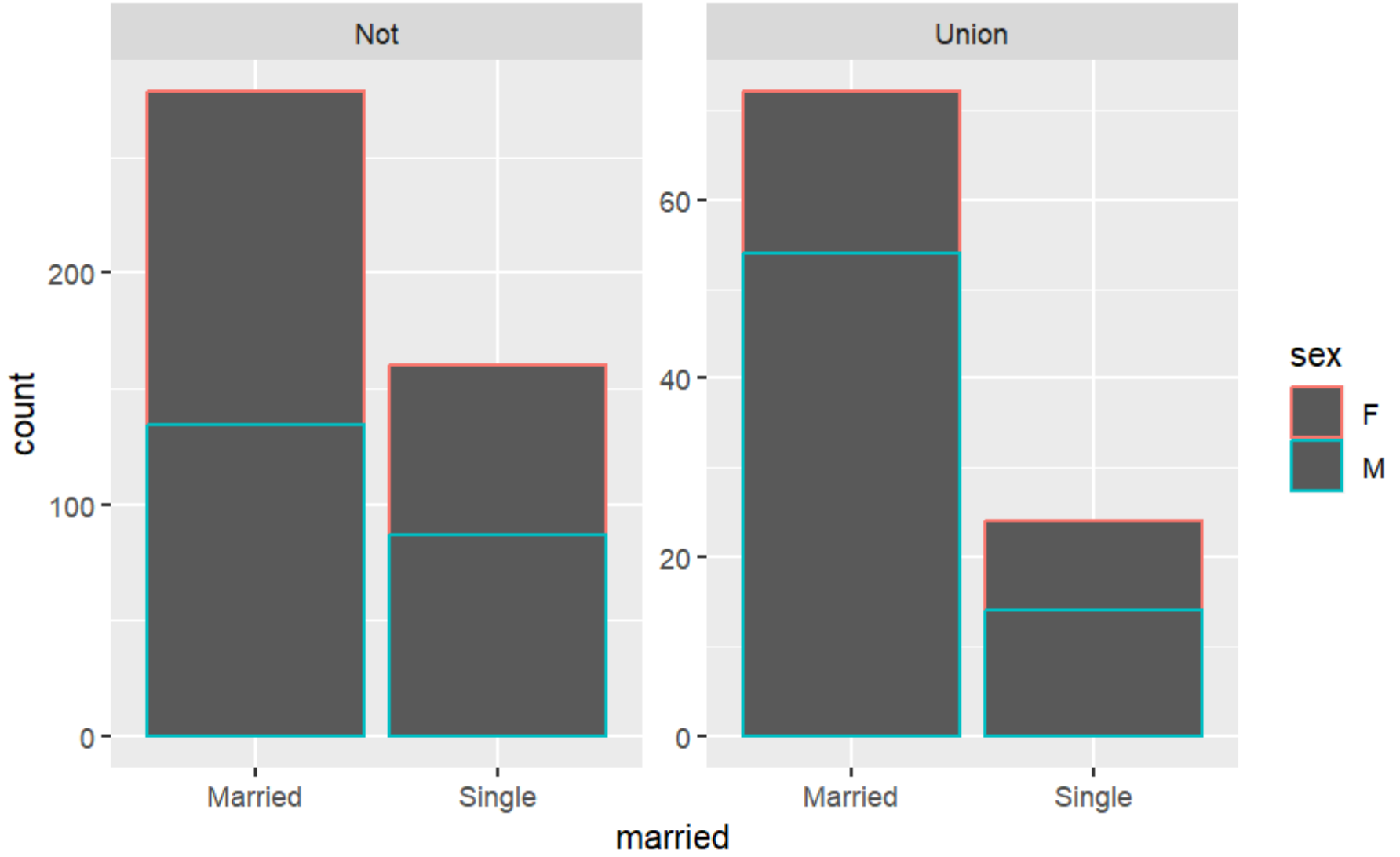
Hide

```
CPS85 %>%
  ggplot() +
  geom_density(aes(x = wage, fill = sex, color = sex), alpha = 0.4)+
  facet_grid( ~ married) +
  xlim(0,30)
```



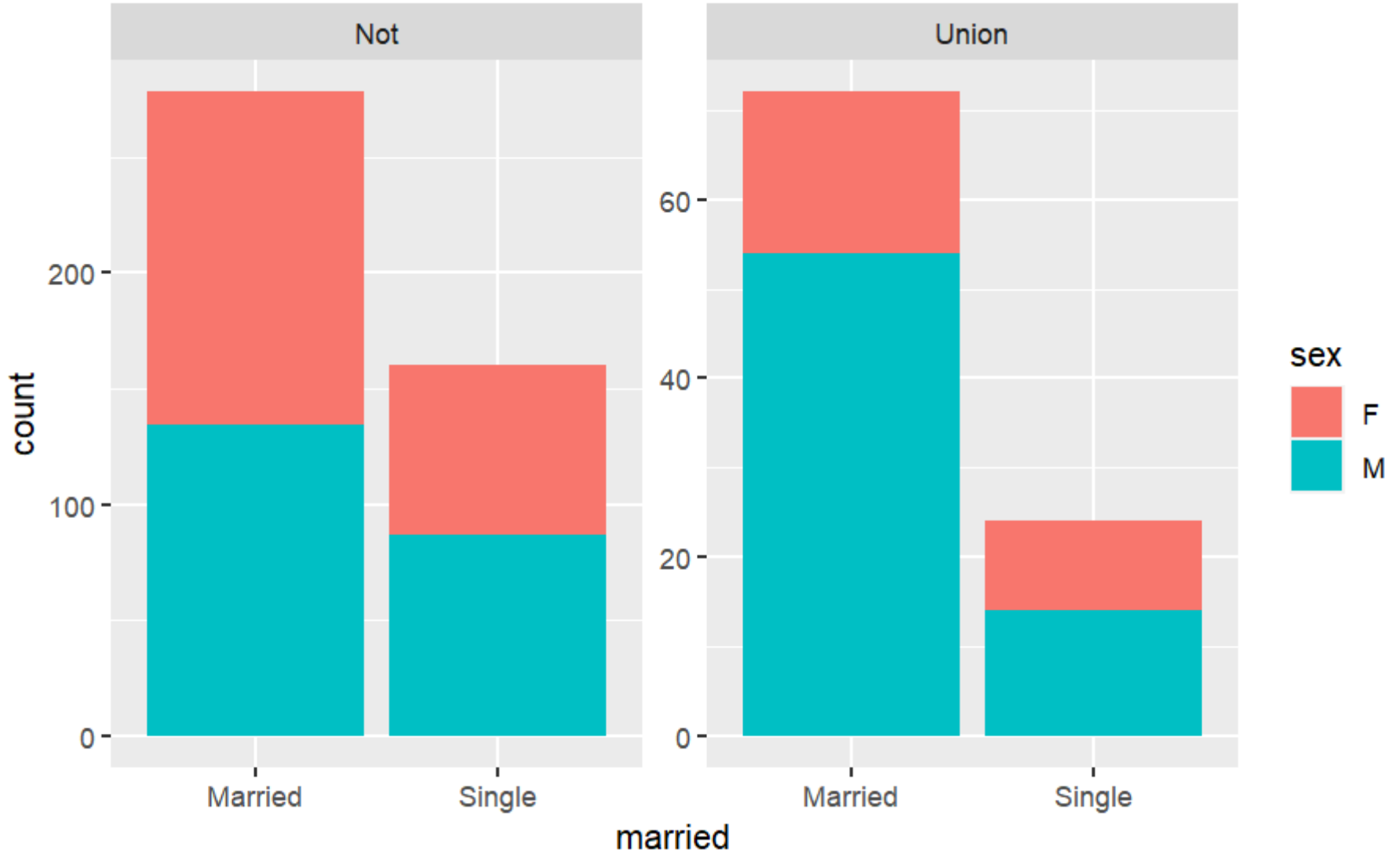
Hide

```
CPS85%>%
  ggplot(aes(x = married, color = sex)) +
  geom_bar() +
  facet_wrap( ~ union, scales = "free")  #Note the scales here
```



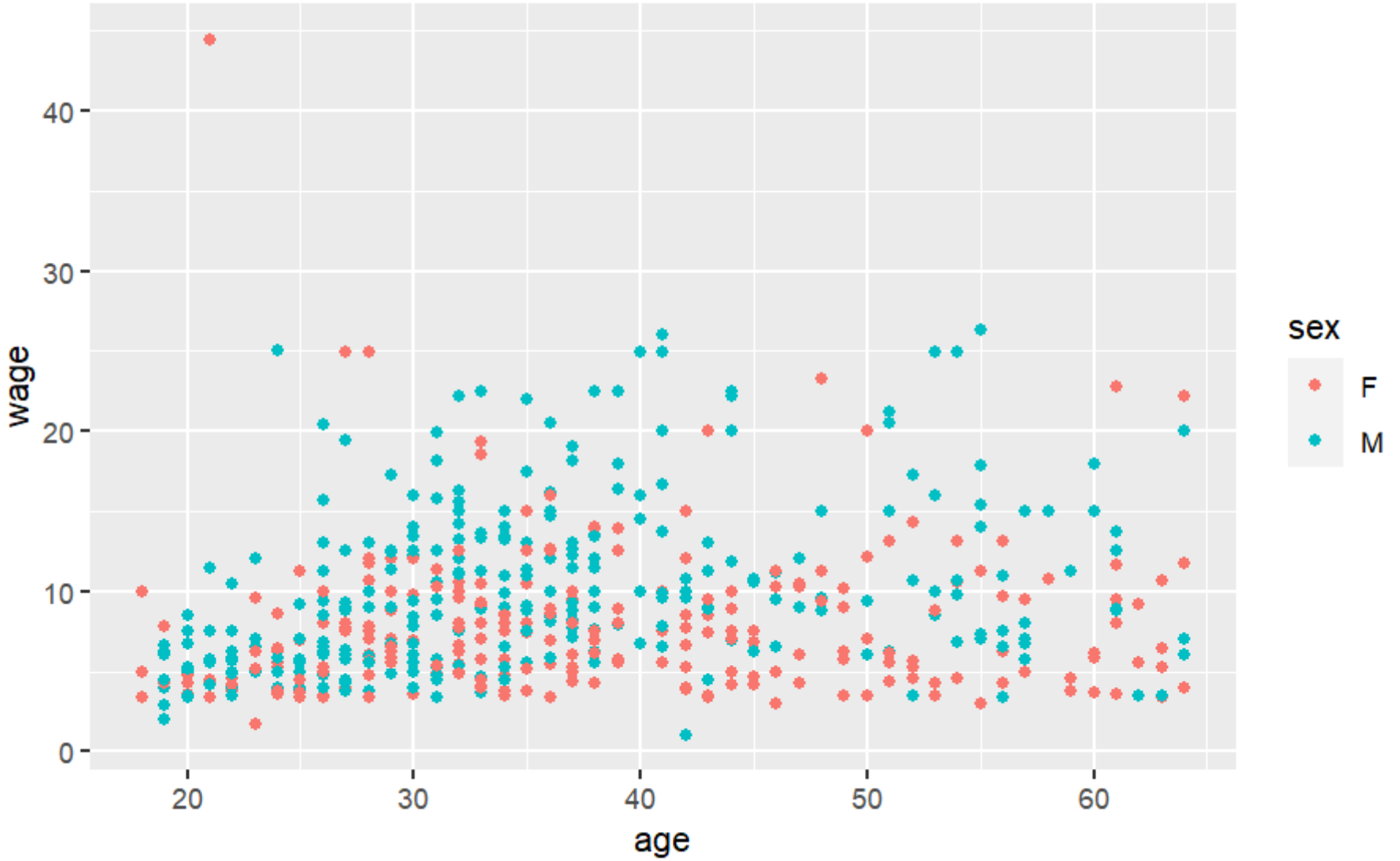
Hide

```
CPS85%>%
  ggplot(aes(x = married, fill = sex)) +
  geom_bar()+
  facet_wrap( ~ union, scales = "free")  #Note the scales here
```



Hide

```
CPS85%>%
  ggplot(aes(x = age, y = wage, color = sex)) +
  geom_point()
```



Hide

```
CPS85%>%
  ggplot(aes(x = age, y = wage, fill = sex)) +  #fill does not work for points!
  geom_point()
```

Hide

NA
NA

Another Example using Diamonds Data

1. establish the frame
2. plot the glyphs (i.e., select a geom)
3. map the aesthetics
4. add labels and title
5. other features (e.g., alpha, sizing, etc)
6. Establish the Frame

Hide

```
ggplot(data = diamonds)
```

2. plot the glyphs (i.e., select a geom)

Hide

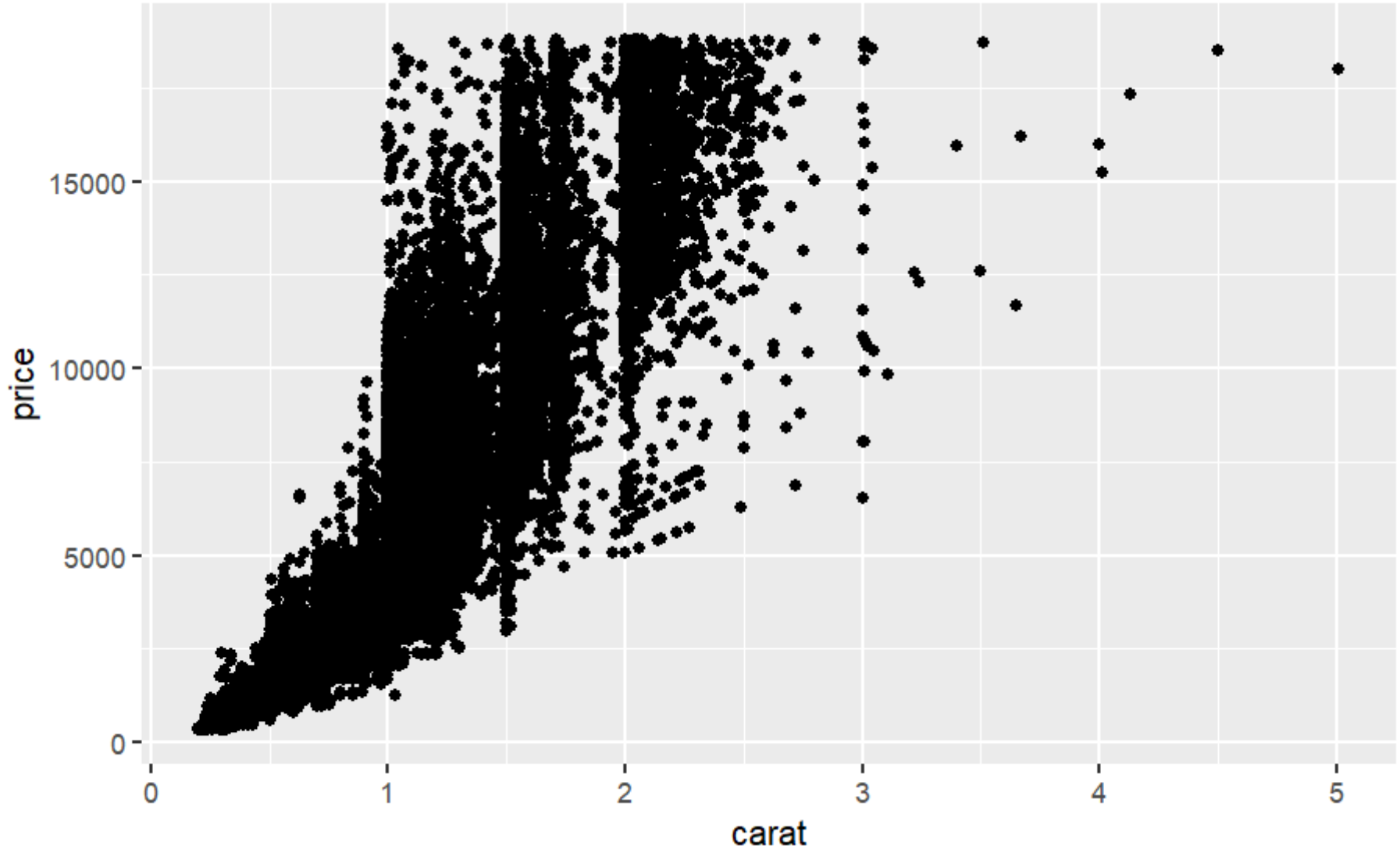
```
ggplot(data = diamonds) +
  geom_point()
```

```
Error in `geom_point()`:
! Problem while setting up geom.
ℹ Error occurred in the 1st layer.
Caused by error in `compute_geom_1()`:
! `geom_point()` requires the following missing aesthetics:
  x and y
Backtrace:
  1. base (local) `<fn>`(x)
  2. ggplot2:::print.ggplot(x)
  4. ggplot2:::ggplot_build.ggplot(x)
  5. ggplot2:::by_layer(...)
 12. ggplot2 (local) f(l = layers[[i]], d = data[[i]])
 13. l$compute_geom_1(d)
 14. ggplot2 (local) compute_geom_1(..., self = self)
```

3. Map the aesthetics

Hide

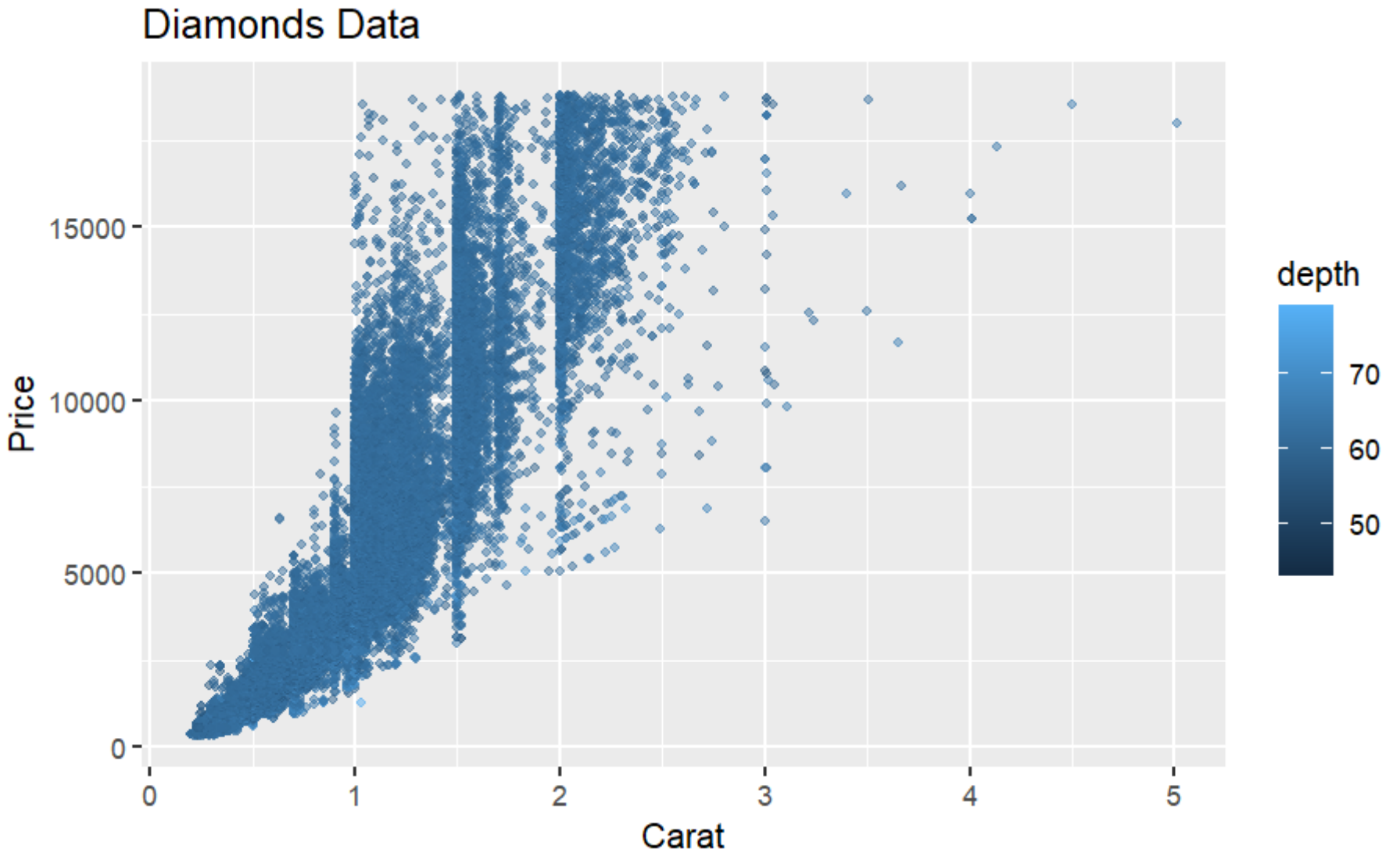
```
ggplot(data = diamonds, aes(x = carat, y = price)) +
  geom_point()
```



4. Add Titles and Labels

Hide

```
ggplot(data = diamonds, aes(x = carat, y = price)) +
  geom_point(aes(color = depth), alpha = 0.5, size = 1) +
  ggtitle("Diamonds Data") +
  xlab("Carat") +
  ylab("Price")
```



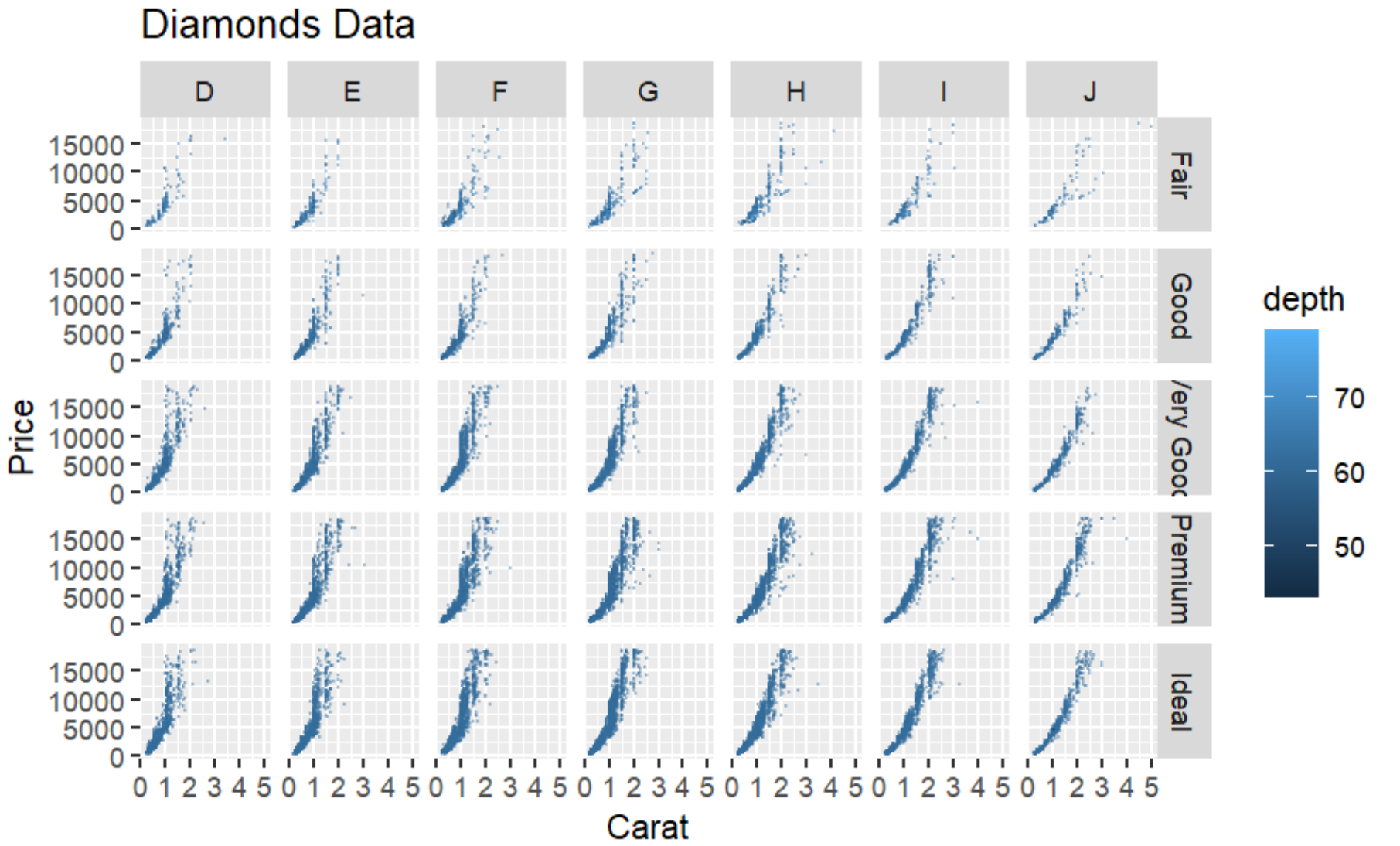
5. Add additional features

Notice that I can have `aes` inside multiple statements. Notice that when I use constants (like `alpha = 0.3, size = 0.1`) they ARE NOT inside `aes` .

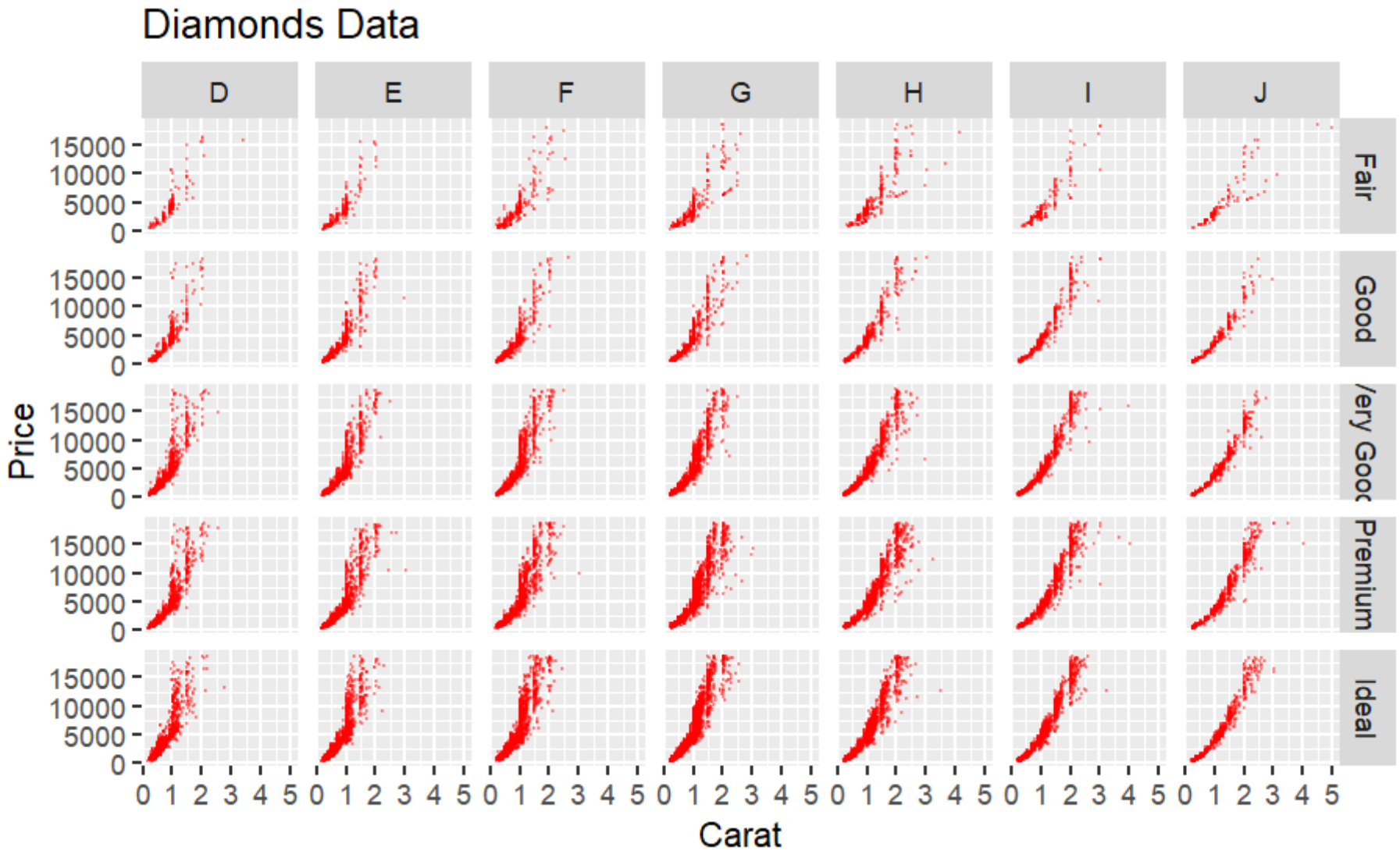
In general, variables go inside `aes` and constants go outside of it. (unless we are using facets then see previous materials.)

Hide

```
ggplot(data = diamonds, aes(x = carat, y = price)) +
  geom_point(aes(colour = depth), alpha = 0.3, size = 0.1) +
  ggtitle("Diamonds Data") +
  xlab("Carat") +
  ylab("Price") +
  facet_grid( cut ~ color)
```


Diamonds Data
D
E
F
G
H
I
J
Fair
Good
/ery Gooc
Premium
Ideal
Price
15000
10000
5000
0
0 1 2 3 4 5
Carat
depth
70
60
50
Hide

```
ggplot(data = diamonds, aes(x = carat, y = price)) +
  geom_point(colour = "red", alpha = 0.3, size = 0.1) +
  ggtitle("Diamonds Data") +
  xlab("Carat") +
  ylab("Price") +
  facet_grid( cut ~ color)
```

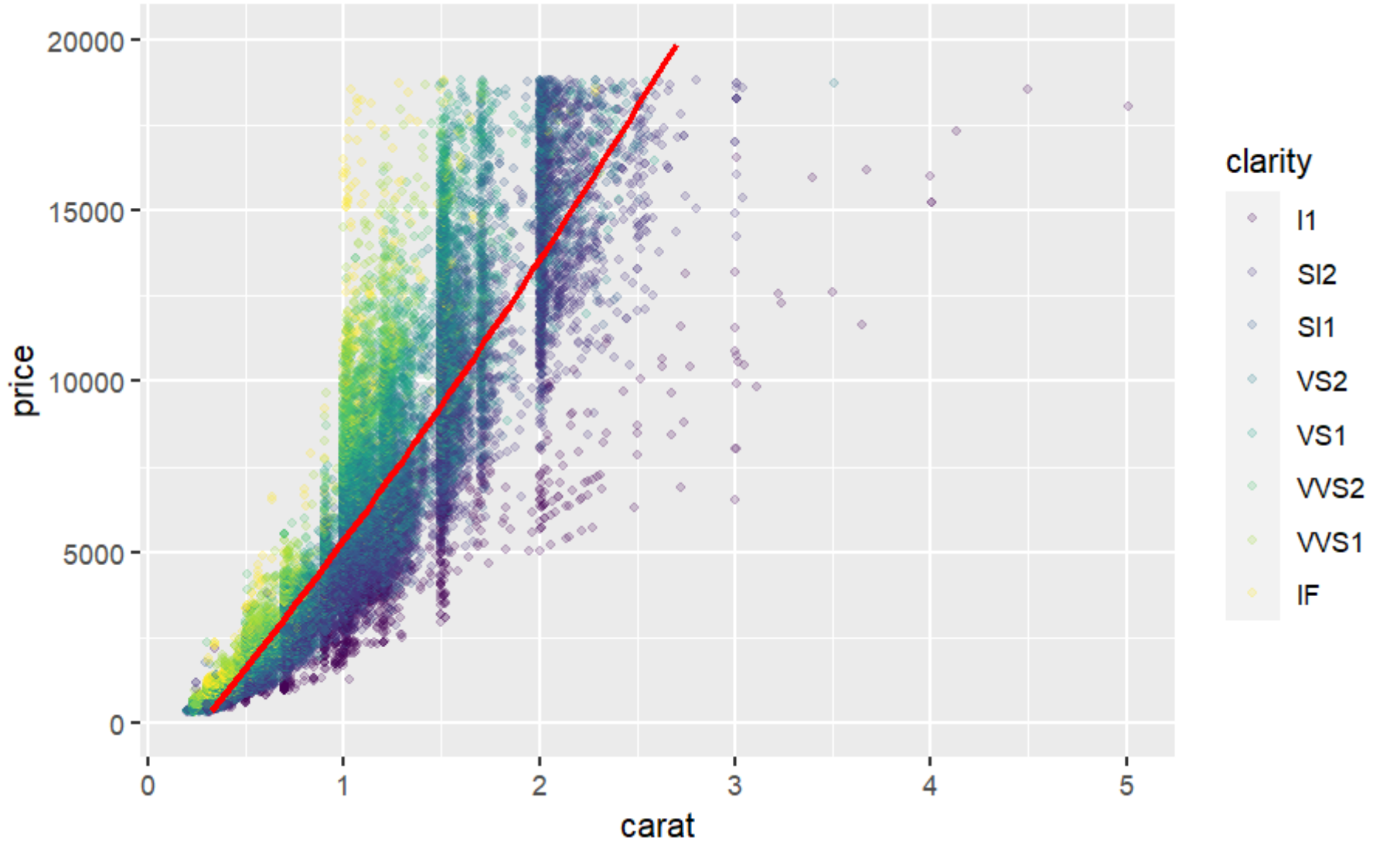


Side Note about placement of `aes`

`aes` can either go inside the `ggplot()` function, or inside the `geom_[chart]()` function itself, or both. The 3 following options create the same plots, but the code is slightly different.

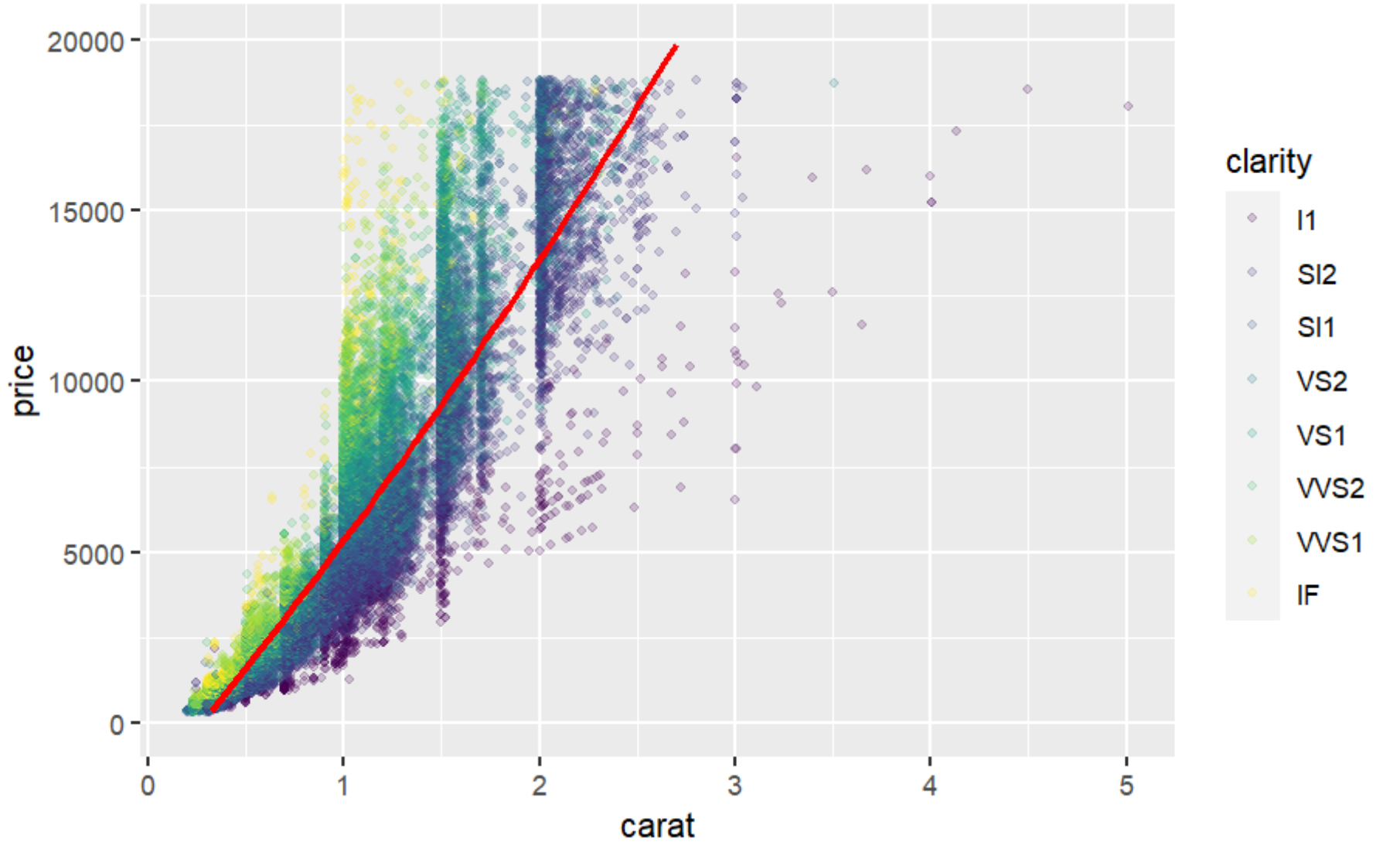
Hide

```
#option 1
ggplot(data = diamonds, ) +
  geom_point(aes(x = carat, y = price, color = clarity),
             alpha = 0.2,
             size = 1) +
  geom_smooth(method = "glm" ,
              formula = y ~ poly(x, 2),                    # y = b_0 + b_1 x + b_2 x^2 + e
              aes(x = carat, y = price),
              color = "red") +
  ylim(c(0, 20000))
```


20000
15000
10000
5000
0
price
0
1
2
3
4
5
carat
clarity
I1
SI2
SI1
VS2
VS1
VVS2
VVS1
IF

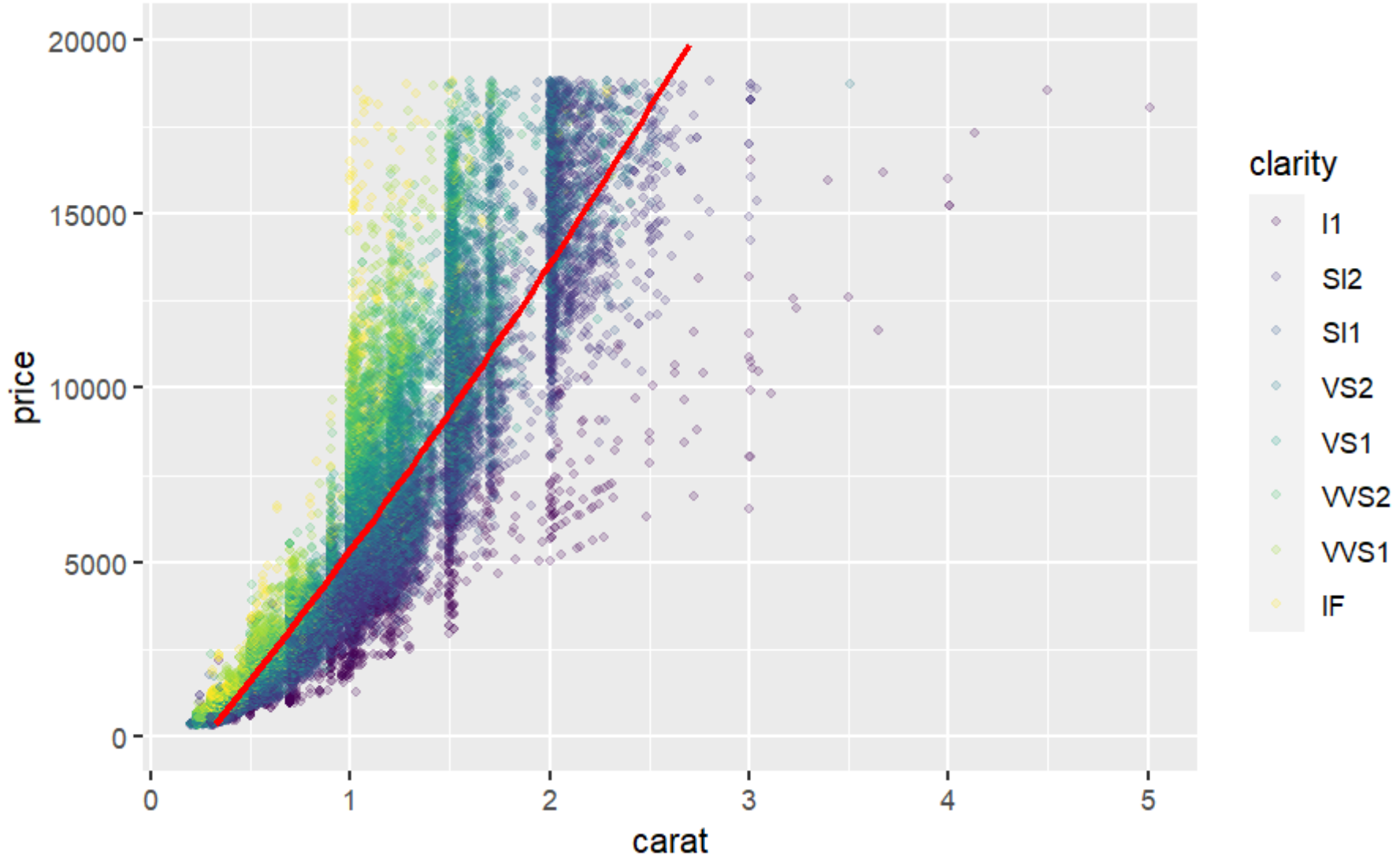
Hide

```
#option 2
ggplot(data = diamonds, aes(x = carat, y = price, color = clarity)) +
  geom_point(alpha = 0.2,
             size = 1) +
  geom_smooth(method = "glm" ,
              formula = y ~ poly(x, 2),                    # y = b_0 + b_1 x + b_2 x^2 + e
              aes(x = carat, y = price),
              color = "red") +
  ylim(c(0, 20000))
```


20000
15000
10000
5000
0
price
0
1
2
3
4
5
carat
clarity
I1
SI2
SI1
VS2
VS1
VVS2
VVS1
IF

Hide

```
#Option 3
ggplot(data = diamonds,  aes(x = carat, y = price) )+
  geom_point( aes(color = clarity),
              alpha = 0.2,
              size = 1) +
  geom_smooth(method = "glm" ,
              formula = y ~ poly(x, 2),                    # y = b_0 + b_1 x + b_2 x^2 + e
              color = "red") +
  ylim(c(0, 20000))
```



- I personally prefer to put “global” aesthetics in the `ggplot()` and “local” aesthetics in the `geom` .
    - Option 1 : all aesthetics are local to the geom
        - Note we have to repeat `x` and `y`
    - Option 2 : all aesthetics are global to the ggplot
        - Note that `color = clarity` is not needed for `geom_smooth`
    - Option 3 : global aesthetics are in the ggplot and local aesthetics and in the geom

    - Both `geom_point` and `geom_smooth` use `x` and `y` so I put them in the `ggplot()`
    - Only `geom_point` uses `color = clarity` so I put that ONLY in the `geom_point` function
- In my opinion, Option 3 is the “cleanest” code. This is partly based on stylistic preference and partly based on some internal mechanic of ggplot’s (that is beyond the scope of this course). How you write your code is up to you. Just keep it readable!
- But again, all 3 codes generate the the exact same plot (so does it really matter that much which option we use??)

Additional resources for learning more about ggplot2

- Check out http://www.sthda.com/english/wiki/ggplot2-essentials (http://www.sthda.com/english/wiki/ggplot2-essentials)

Reminders about assignments (all due Friday July 21, 9:59 am)

- Activity: PopularNames
- Reading quiz Chapters 10 and 11