

Data Visualization in R

ggplot2 and the grammar of graphics

2/20/23

gg plot 2:

Build a data
MASTERpiece

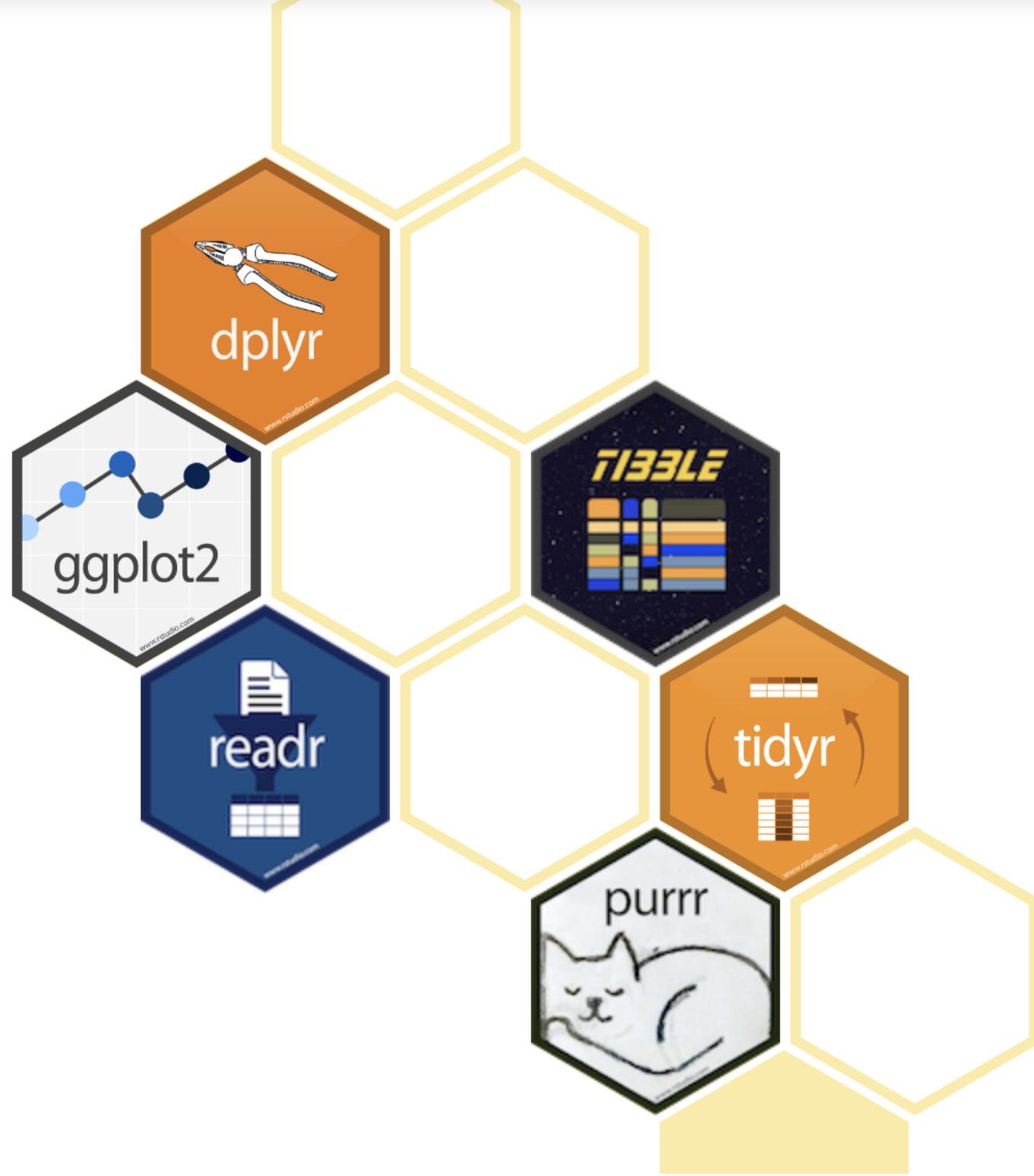


Art by Allison Horst

Data Visualization with R

`ggplot2` works well with the `tidyverse` and is friendly and powerful

Better plots are better communication



ggplot2: Elegant Data Visualizations in R



A Layered Grammar of Graphics

Data is mapped to aesthetics; Statistics and plot are linked

Sensible defaults; Infinitely extensible

Publication quality and beyond

<https://nyti.ms/2jUp36n>

<http://bit.ly/2KSGZLu>



```
1 # print prettily
2 as_tibble(mtcars)
```

```
# A tibble: 32 × 11
  mpg   cyl  disp    hp  drat    wt  qsec    vs    am
  <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
1 21       6   160   110   3.9   2.62  16.5     0     1
2 21       6   160   110   3.9   2.88  17.0     0     1
3 22.8     4   108    93   3.85  2.32  18.6     1     1
4 21.4     6   258   110   3.08  3.22  19.4     1     0
5 18.7     8   360   175   3.15  3.44  17.0     0     0
6 18.1     6   225   105   2.76  3.46  20.2     1     0
7 14.3     8   360   245   3.21  3.57  15.8     0     0
8 24.4     4   147.   62    3.69  3.19  20        1     0
9 22.8     4   141.   95    3.92  3.15  22.9     1     0
10 19.2    6   168.   123   3.92  3.44  18.3    1     0
# ... with 22 more rows
```



new data alert!



mtcars

	mpg	cyl	disp	hp	drat	wt	qsec	vs	am	gear	carb
Mazda RX4	21.0	6	160.0	110	3.90	2.620	16.46	0	1	4	4
Mazda RX4 Wag	21.0	6	160.0	110	3.90	2.875	17.02	0	1	4	4
Datsun 710	22.8	4	108.0	93	3.85	2.320	18.61	1	1	4	1
Hornet 4 Drive	21.4	6	258.0	110	3.08	3.215	19.44	1	0	3	1
Hornet Sportabout	18.7	8	360.0	175	3.15	3.440	17.02	0	0	3	2
Valiant	18.1	6	225.0	105	2.76	3.460	20.22	1	0	3	1
Duster 360	14.3	8	360.0	245	3.21	3.570	15.84	0	0	3	4
Merc 240D	24.4	4	146.7	62	3.69	3.190	20.00	1	0	4	2
Merc 230	22.8	4	140.8	95	3.92	3.150	22.90	1	0	4	2
Merc 280	19.2	6	167.6	123	3.92	3.440	18.30	1	0	4	4
Merc 280C	17.8	6	167.6	123	3.92	3.440	18.90	1	0	4	4
Merc 450SE	16.4	8	275.8	180	3.07	4.070	17.40	0	0	3	3
Merc 450SL	17.3	8	275.8	180	3.07	3.730	17.60	0	0	3	3
Merc 450SLC	15.2	8	275.8	180	3.07	3.780	18.00	0	0	3	3
Cadillac Fleetwood	10.4	8	472.0	205	2.93	5.250	17.98	0	0	3	4
Lincoln Continental	10.4	8	460.0	215	3.00	5.424	17.82	0	0	3	4
Chrysler Imperial	14.7	8	440.0	230	3.23	5.345	17.42	0	0	3	4
Fiat 128	32.4	4	78.7	66	4.08	2.200	10.47	1	1	4	1

Where does it come from?

base R

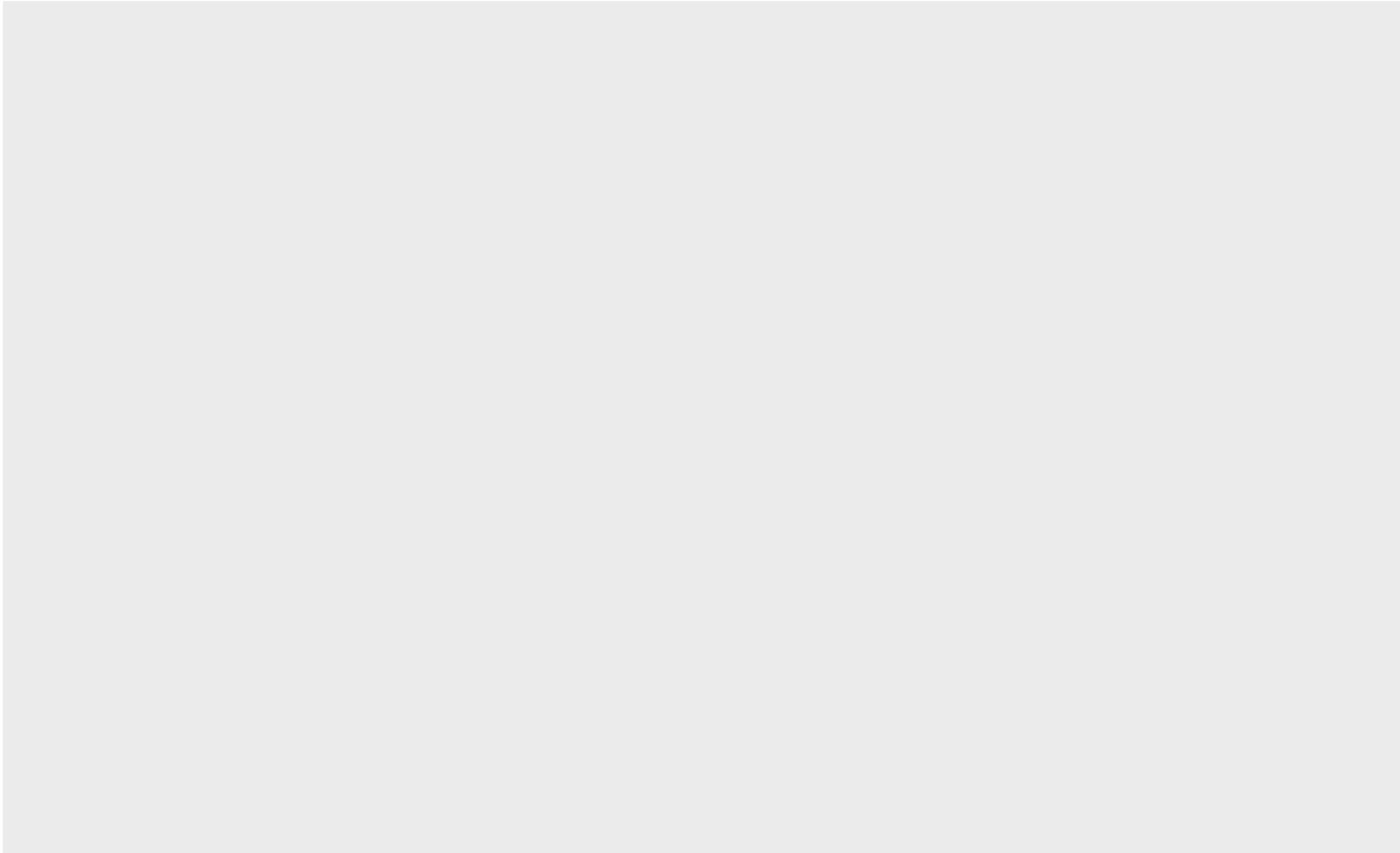
How can I use it?

`View(mtcars)`

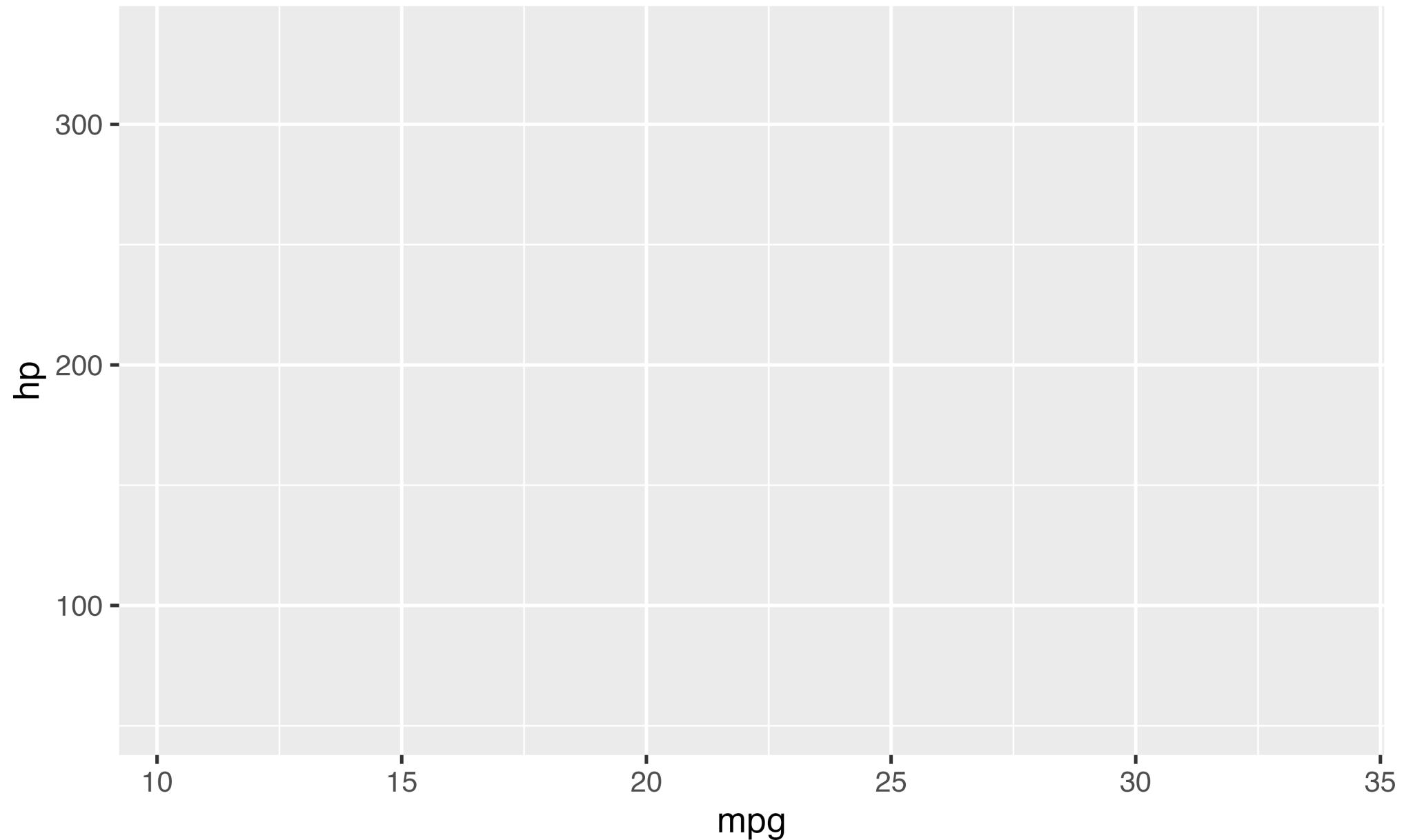


it's invisible!

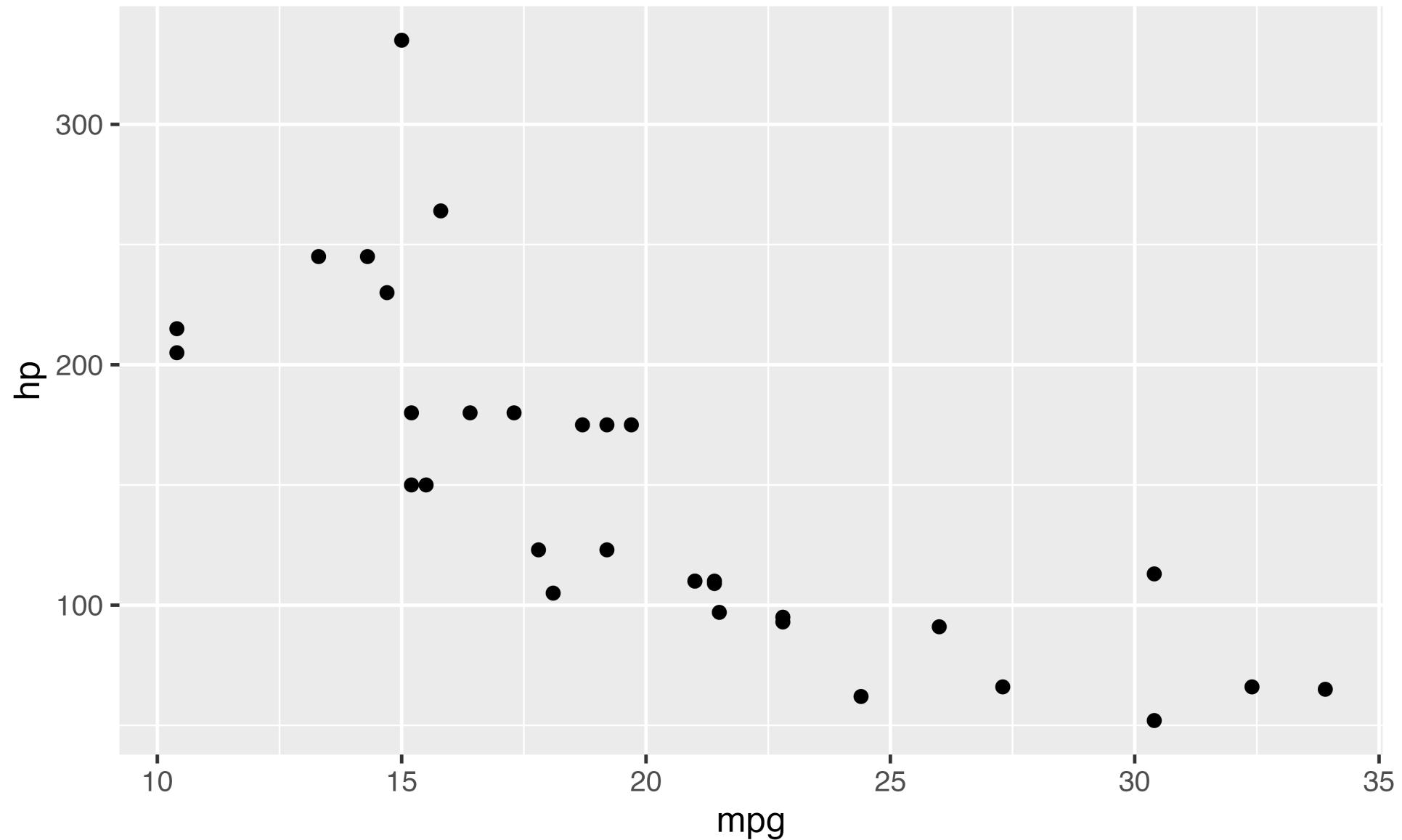
```
1 ggplot()
```



```
1 ggplot(mtcars, aes(x = mpg, y = hp))
```



```
1 ggplot(mtcars, aes(x = mpg, y = hp)) +  
2   geom_point()
```



`ggplot()`

```
ggplot(data = <data>, mapping =  
aes(<mapping>)) +  
<geom_function>()
```

Add layers with +

Put + at the end of a line

Connect *variables in your dataset* to *aesthetics in the plot* with
`aes()`

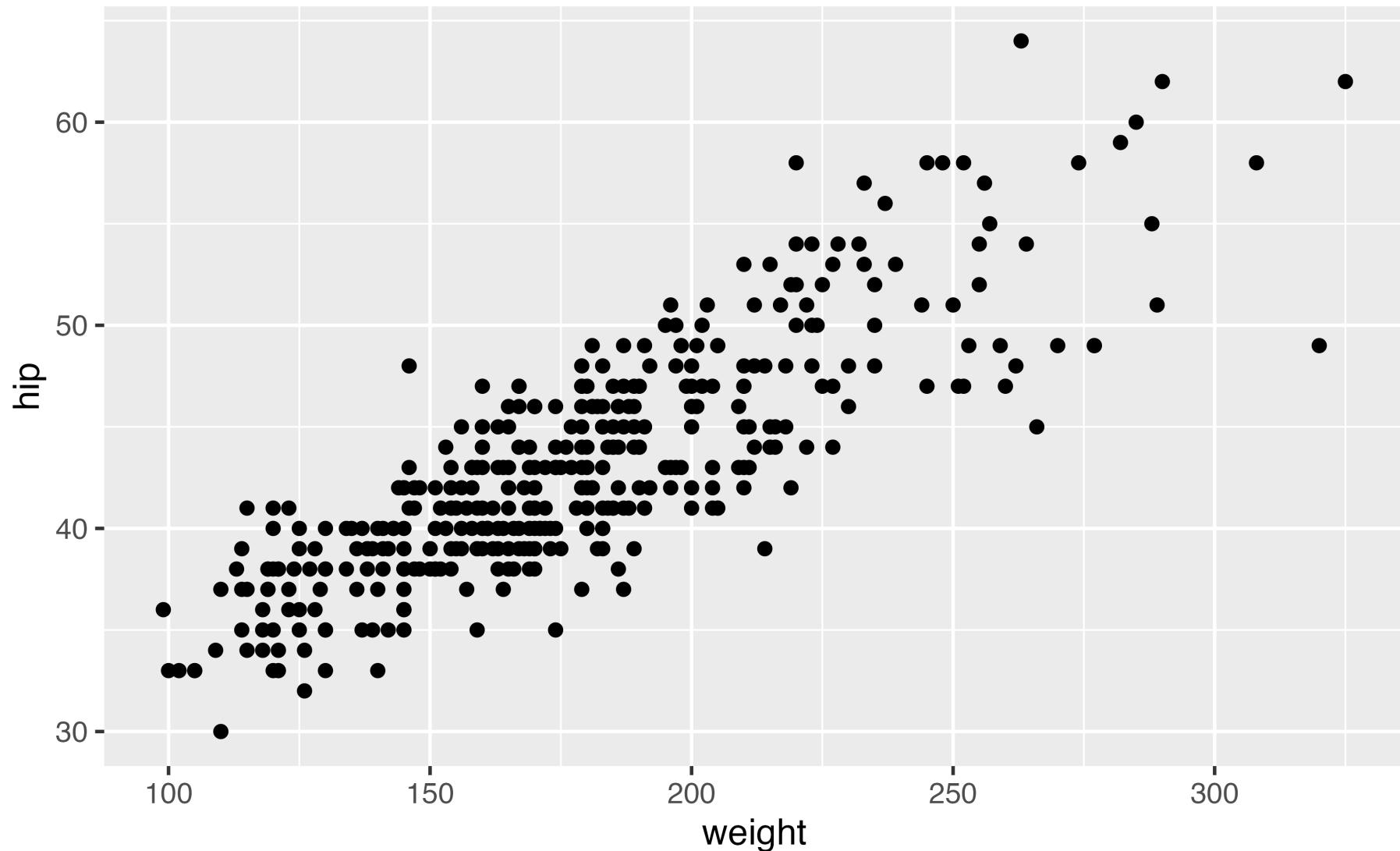
Your Turn 1

Read in the **diabetes** data.

Write and run the code from this slide to make a graph. Pay strict attention to spelling, capitalization, and parentheses!

```
1 ggplot(data = diabetes, mapping = aes(x = weight, y = hip)) +  
2   geom_point()
```

```
1 diabetes <- read_csv("diabetes.csv")
2 ggplot(data = diabetes, mapping = aes(x = weight, y = hip)) +
3   geom_point()
```



Aesthetics: aes()

```
ggplot(data = <data>, mapping =  
aes(<mapping>)) +  
<geom_function>()
```

Aesthetics *map* the data to the plot

Aesthetics: aes()

```
1 ggplot(mtcars, aes(x = mpg, y = hp, color = cyl)) + geom_point()  
2  
3 ggplot(mtcars, aes(x = mpg, y = hp, size = cyl)) + geom_point()  
4  
5 ggplot(mtcars, aes(x = mpg, y = hp, alpha = cyl)) + geom_point()  
6  
7 ggplot(mtcars, aes(x = mpg, y = hp, shape = cyl)) + geom_point()
```

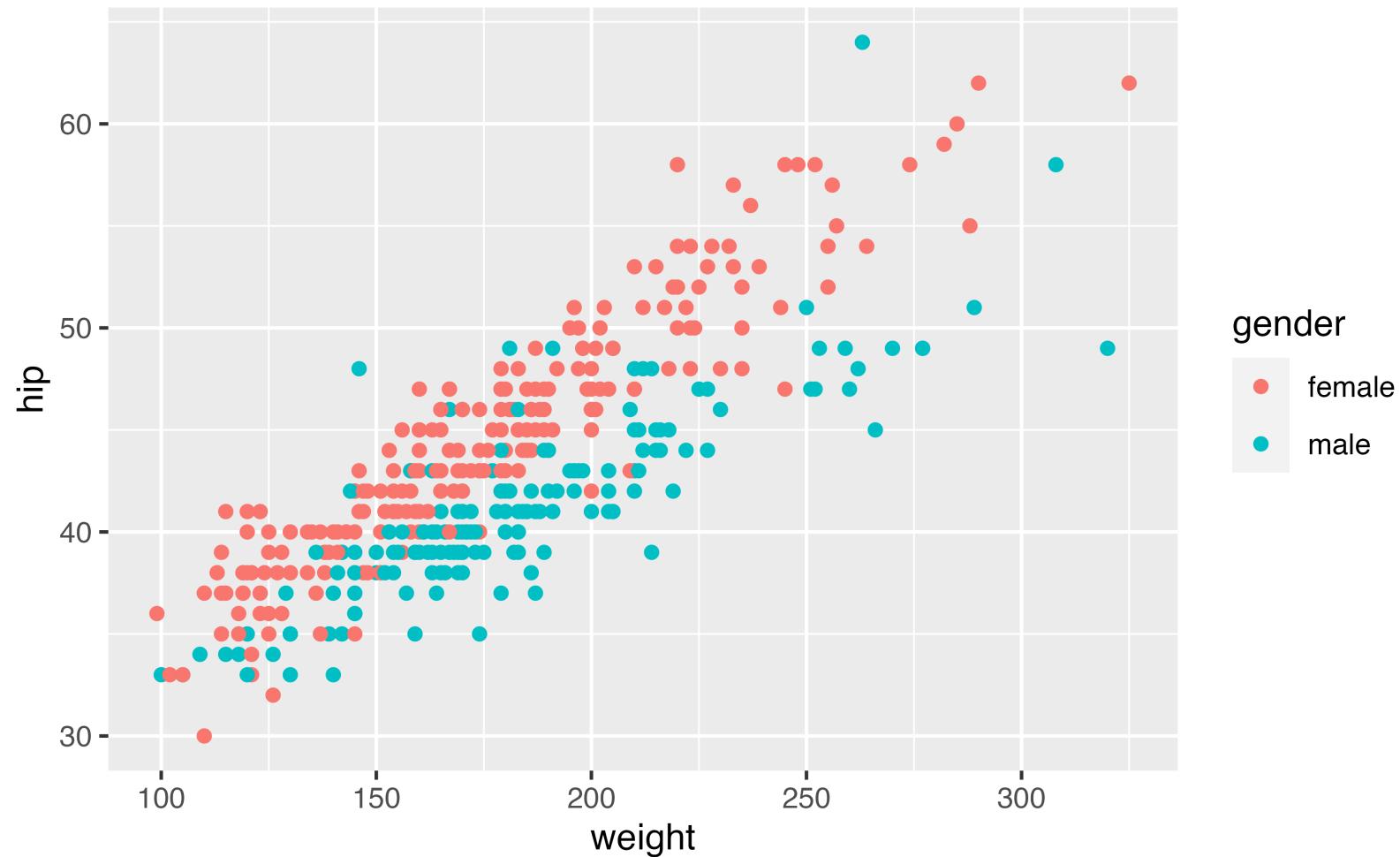
Your Turn 2

Add **color**, **size**, **alpha**, and **shape** aesthetics to your graph using the **gender** variable.
Experiment.

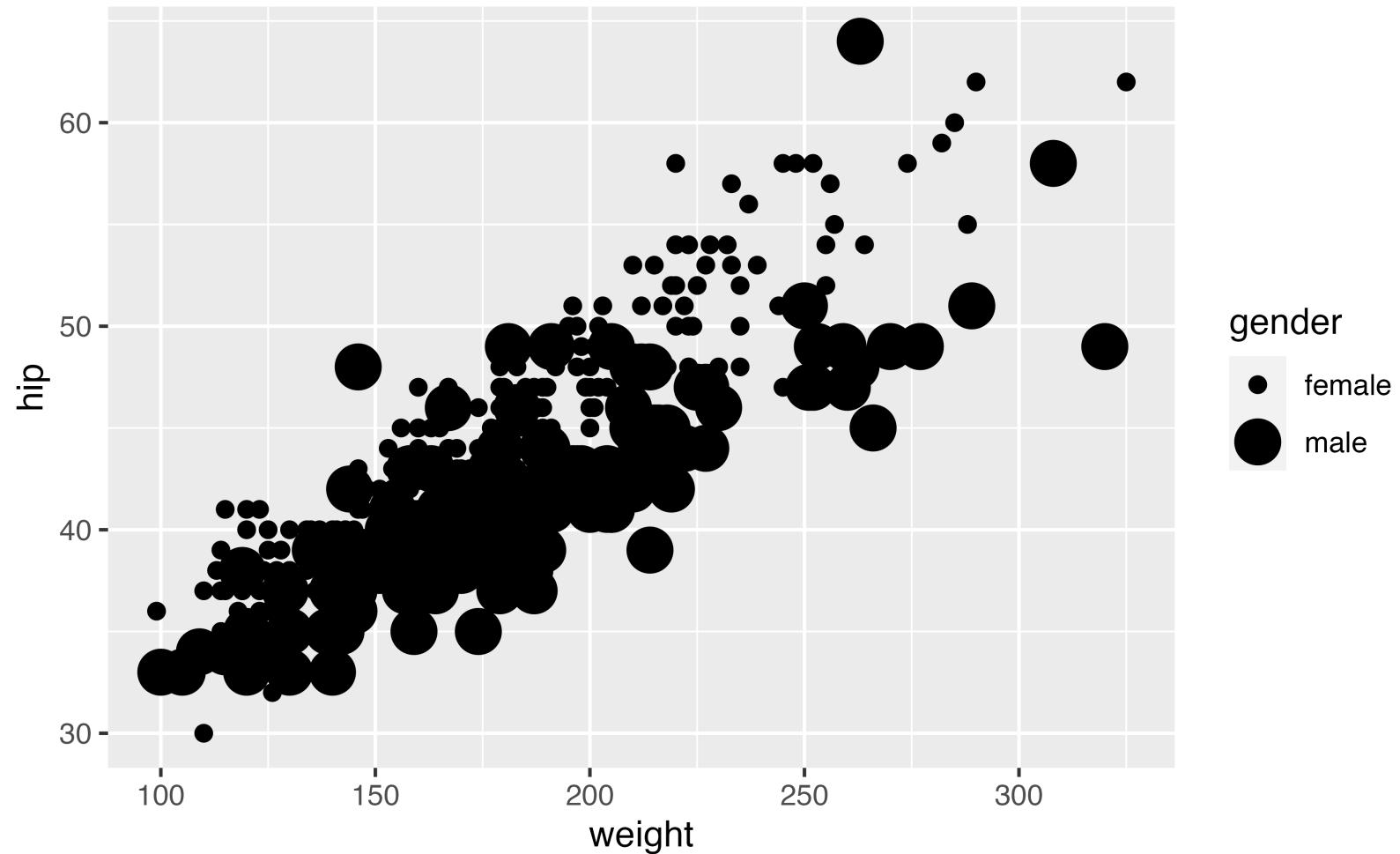
```
1 ggplot(  
2   data = diabetes,  
3   mapping = aes(x = weight, y = hip))  
4 ) +  
5   geom_point()
```

Try moving the **mapping** argument to **geom_point()**. Add in any aesthetics you found helpful.

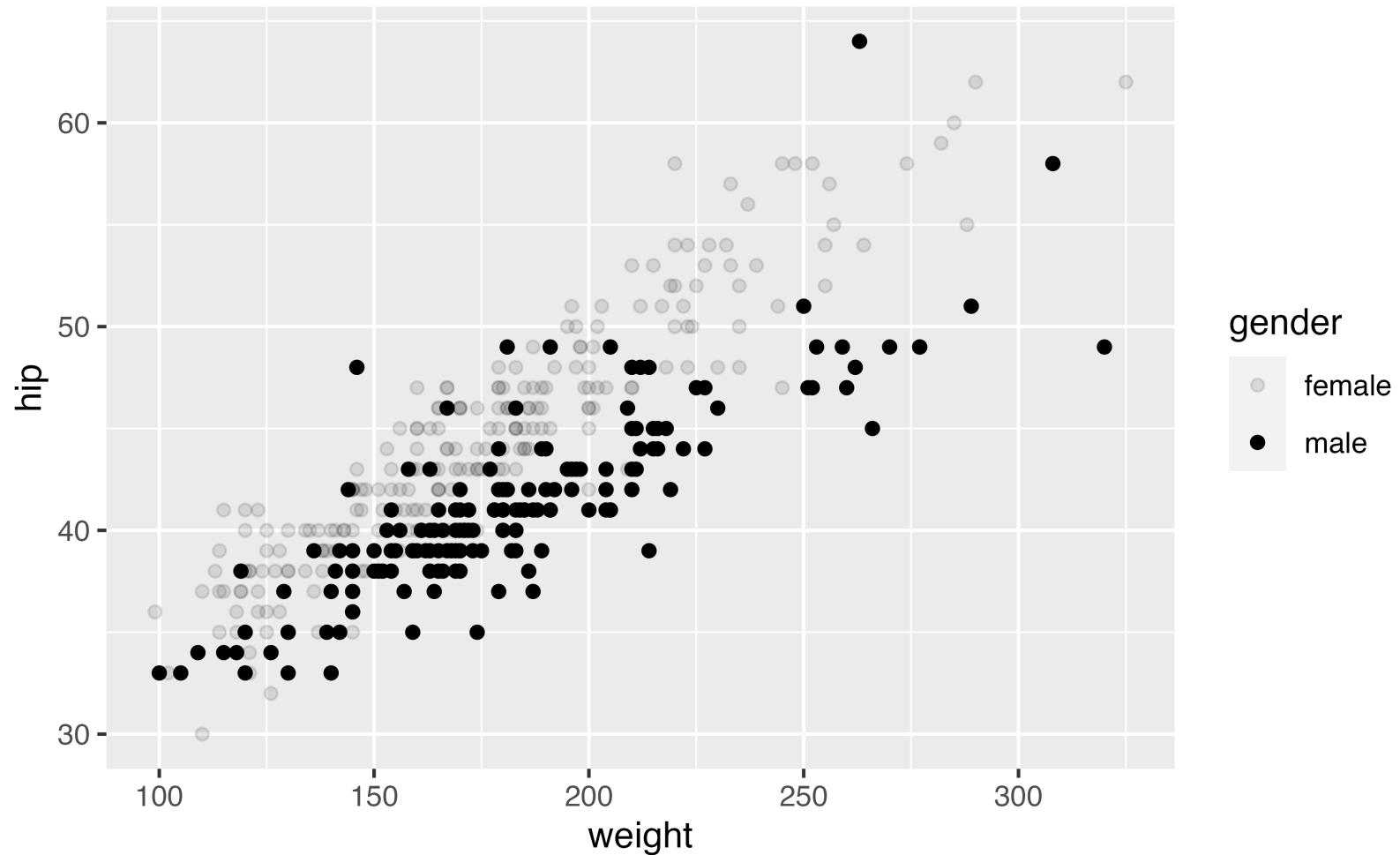
```
1 ggplot(  
2   data = diabetes,  
3   mapping = aes(x = weight, y = hip, color = gender)  
4 ) +  
5   geom_point()
```



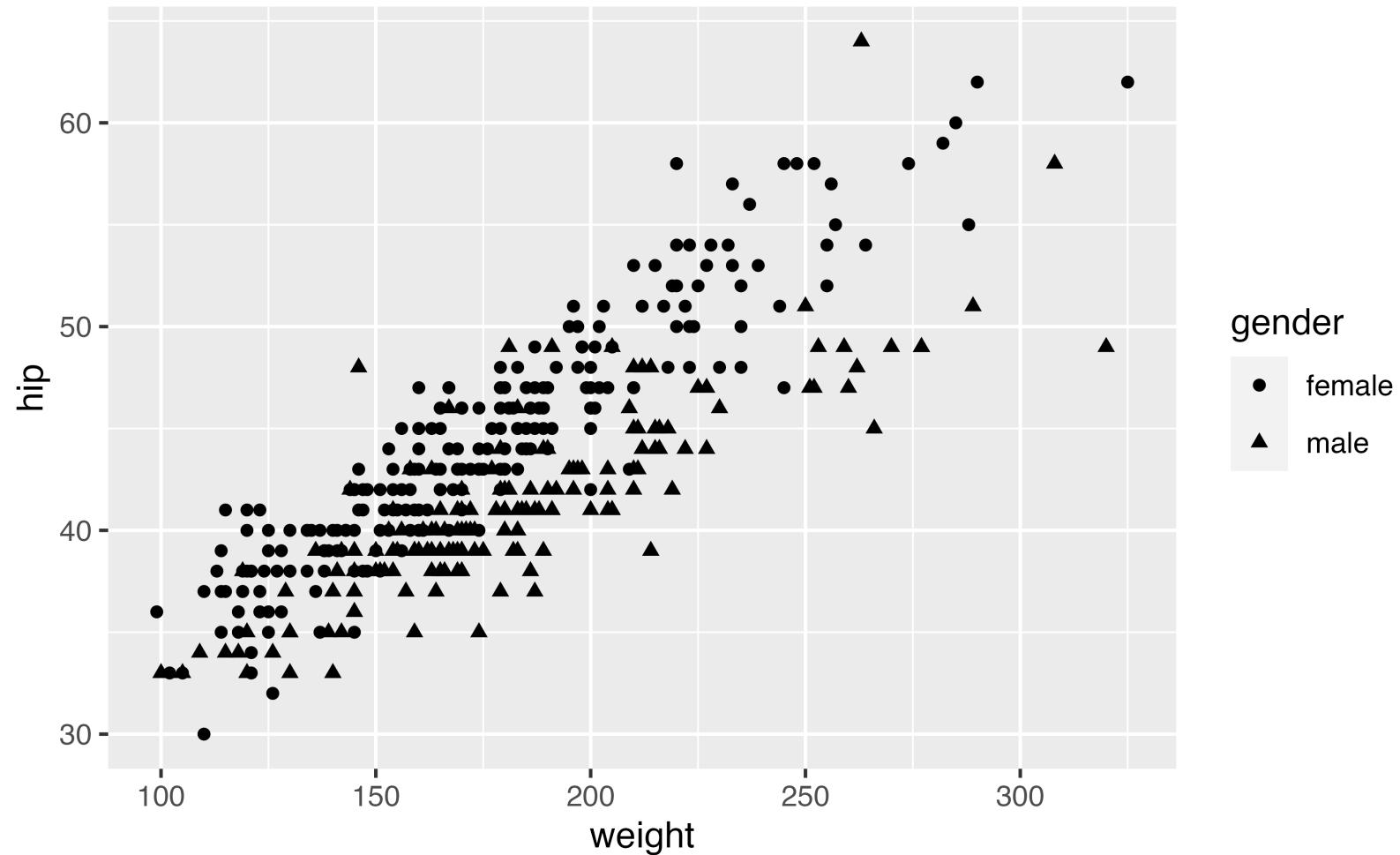
```
1 ggplot(  
2   data = diabetes,  
3   mapping = aes(x = weight, y = hip, size = gender)  
4 ) +  
5   geom_point()
```



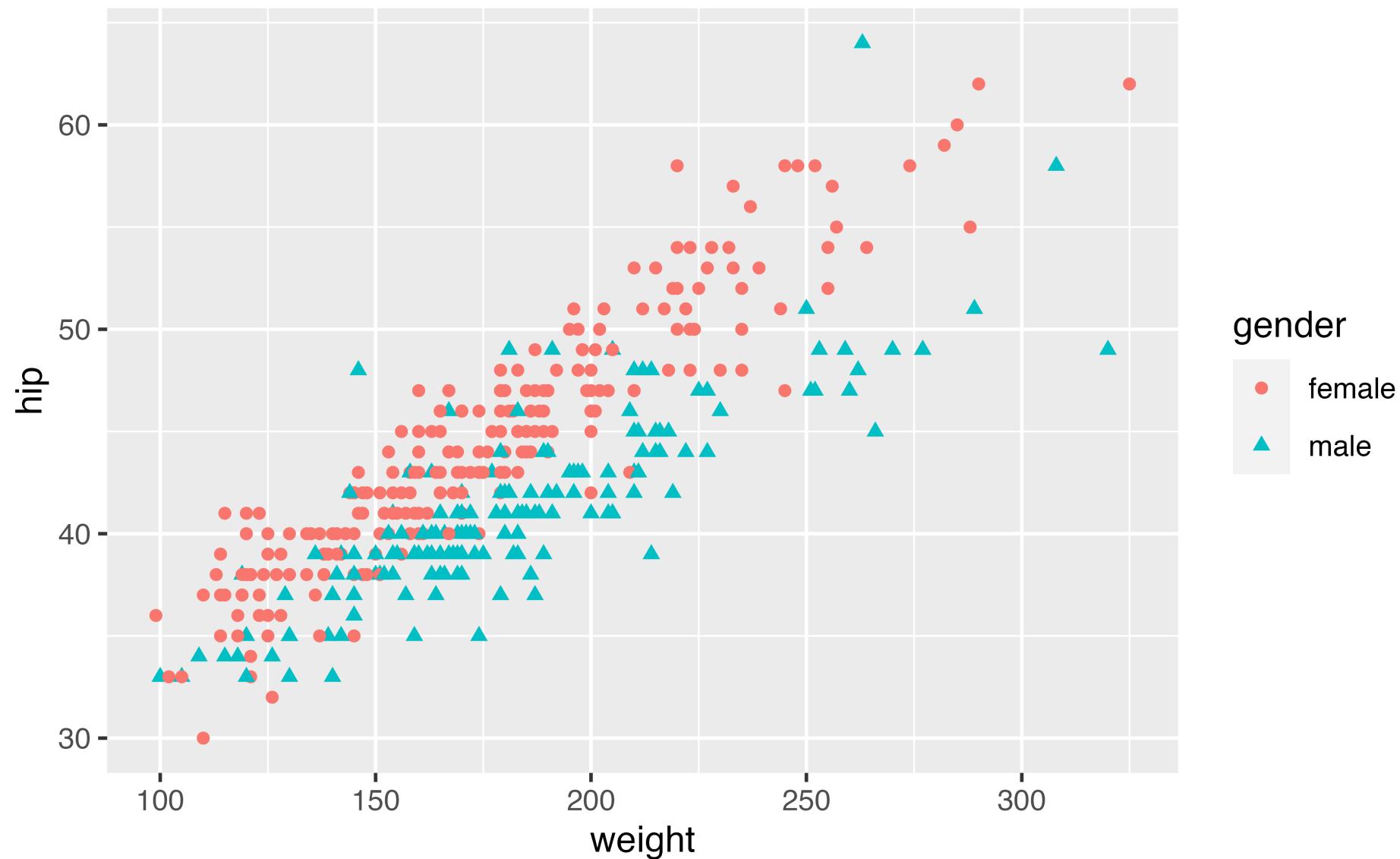
```
1 ggplot(  
2   data = diabetes,  
3   mapping = aes(x = weight, y = hip, alpha = gender)  
4 ) +  
5   geom_point()
```



```
1 ggplot(  
2   data = diabetes,  
3   mapping = aes(x = weight, y = hip, shape = gender)  
4 ) +  
5   geom_point()
```

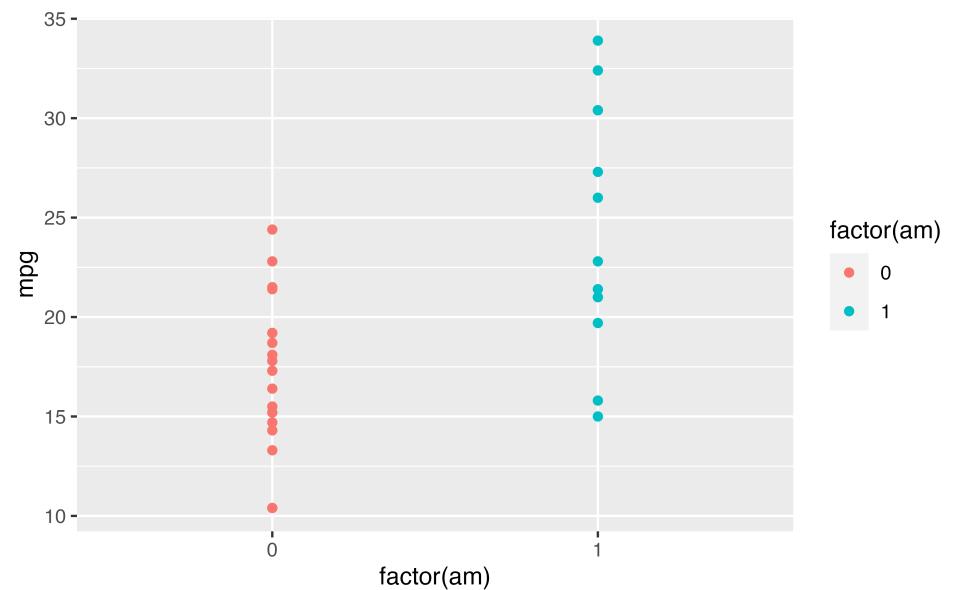


```
1 ggplot(data = diabetes) +  
2   geom_point(  
3     mapping = aes(  
4       x = weight,  
5       y = hip,  
6       color = gender,  
7       shape = gender  
8     )  
9   )
```



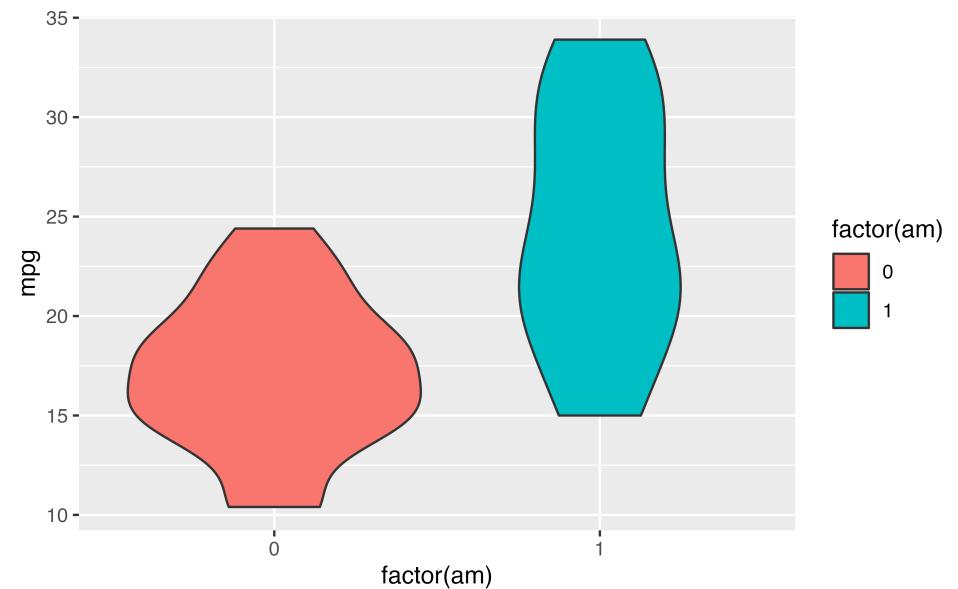
geoms: the shape of the data

```
1 ggplot(  
2   mtcars,  
3   aes(  
4     x = factor(am),  
5     y = mpg,  
6     color = factor(am)  
7   )  
8 ) +  
9 geom_point()
```



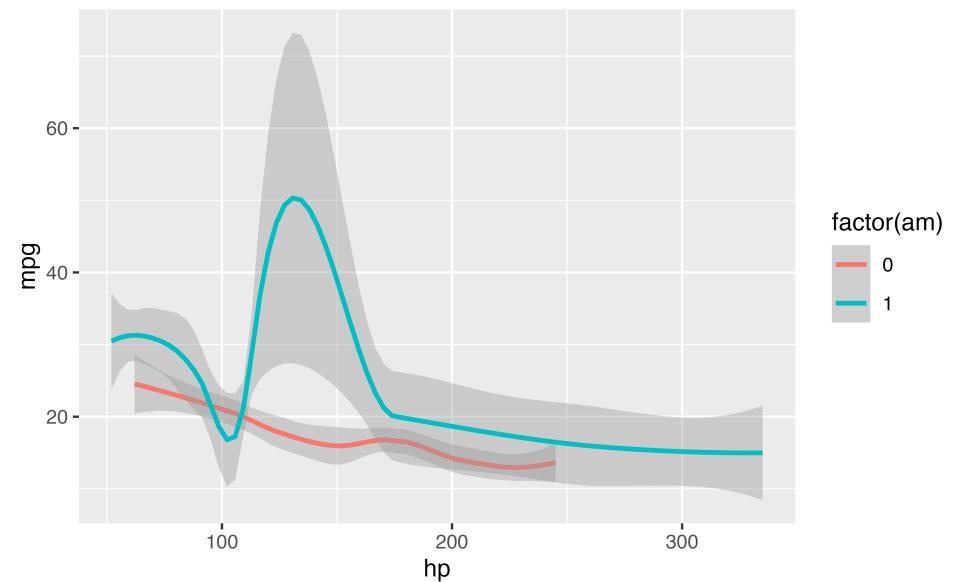
geoms: the shape of the data

```
1 ggplot(  
2   mtcars,  
3   aes(  
4     x = factor(am),  
5     y = mpg,  
6     fill = factor(am)  
7   )  
8 ) +  
9 geom_violin()
```



geoms: the shape of the data

```
1 ggplot(  
2   mtcars,  
3   aes(  
4     x = hp,  
5     y = mpg,  
6     color = factor(am)  
7   )  
8 ) +  
9 geom_smooth()
```



geoms: the shape of the data

Geoms

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

GRAPHICAL PRIMITIVES

```
a <- ggplot(economics, aes(date, unemploy))
b <- ggplot(seals, aes(x = long, y = lat))
```



LINE SEGMENTS

common aesthetics: x, y, alpha, color, linetype, size

```
b + geom_abline(aes(intercept = 0, slope = 1))
b + geom_hline(aes(yintercept = lat))
b + geom_vline(aes(xintercept = long))

b + geom_segment(aes(yend = lat + 1, xend = long + 1))
b + geom_spoke(aes(angle = 1:1155, radius = 1))
```

ONE VARIABLE continuous

```
c <- ggplot(mpg, aes(hwy)); c2 <- ggplot(mpg)

c + geom_area(stat = "bin")
x, y, alpha, color, fill, linetype, size

c + geom_density(kernel = "gaussian")
x, y, alpha, color, fill, group, linetype, size, weight

c + geom_dotplot()
x, y, alpha, color, fill

c + geom_freqpoly()
x, y, alpha, color, group, linetype, size

c + geom_histogram(binwidth = 5)
x, y, alpha, color, fill, linetype, size, weight

c2 + geom_qq(aes(sample = hwy))
x, y, alpha, color, fill, linetype, size, weight
```

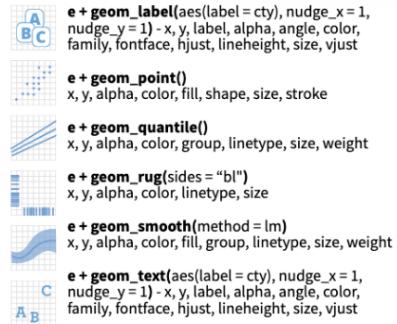
discrete

```
d <- ggplot(mpg, aes(f1))

d + geom_bar()
```

TWO VARIABLES both continuous

```
e <- ggplot(mpg, aes(cty, hwy))
```



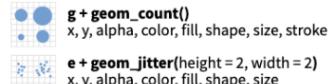
one discrete, one continuous

```
f <- ggplot(mpg, aes(class, hwy))
```



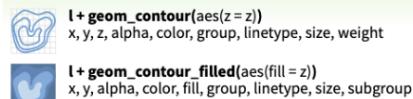
both discrete

```
g <- ggplot(diamonds, aes(cut, color))
```



THREE VARIABLES

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



ggplot2

continuous bivariate distribution

```
h <- ggplot(diamonds, aes(carat, price))
```



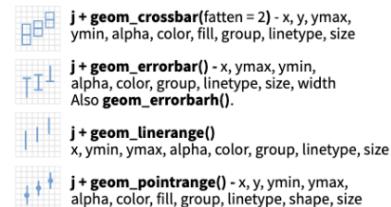
continuous function

```
i <- ggplot(economics, aes(date, unemploy))
```



visualizing error

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



maps

```
data <- data.frame(murder = USArests$Murder, state = tolower(rownames(USArests)))
```

```
map <- map_data("state")
```

```
k <- ggplot(data, aes(fill = murder))
```

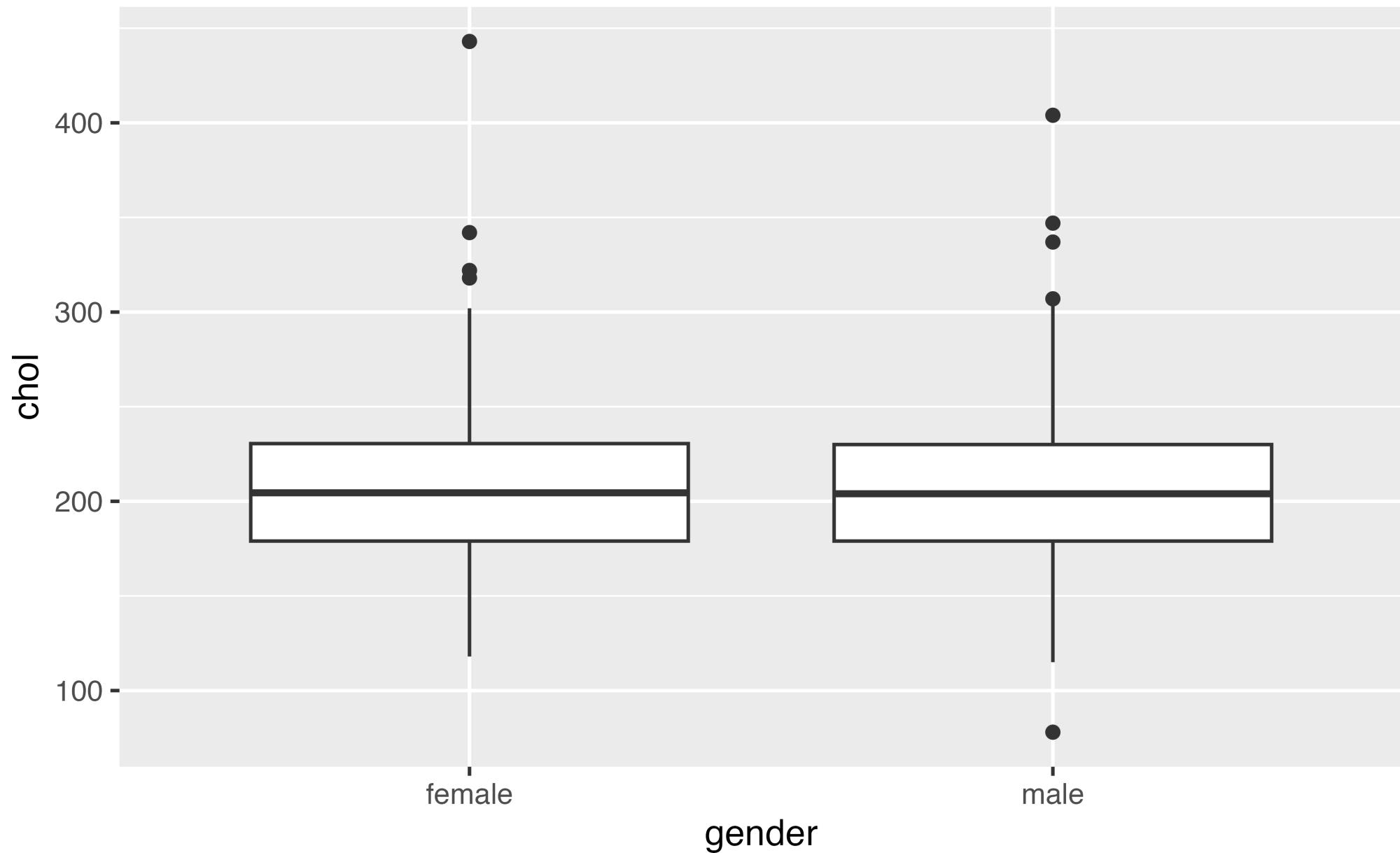
```
k + geom_map(aes(map_id = state), map = map)
+ expand_limits(x = map$long, y = map$lat)
map_id, alpha, color, fill, linetype, size
```

Your Turn 3

Replace this scatterplot with one that draws boxplots.

```
1 ggplot(diabetes, aes(gender, chol)) + geom_point()
```

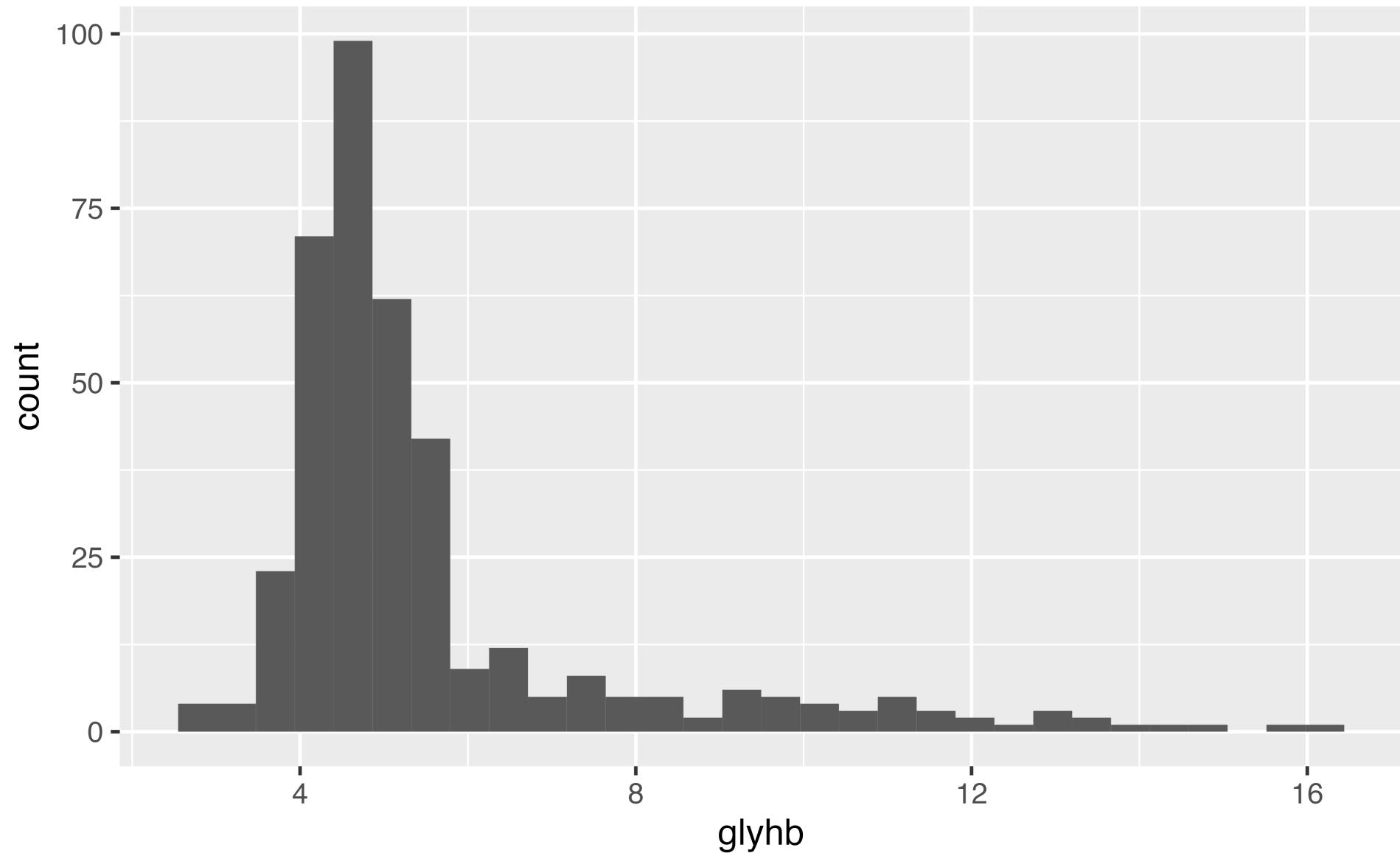
```
1 ggplot(diabetes, aes(gender, chol)) + geom_boxplot()
```



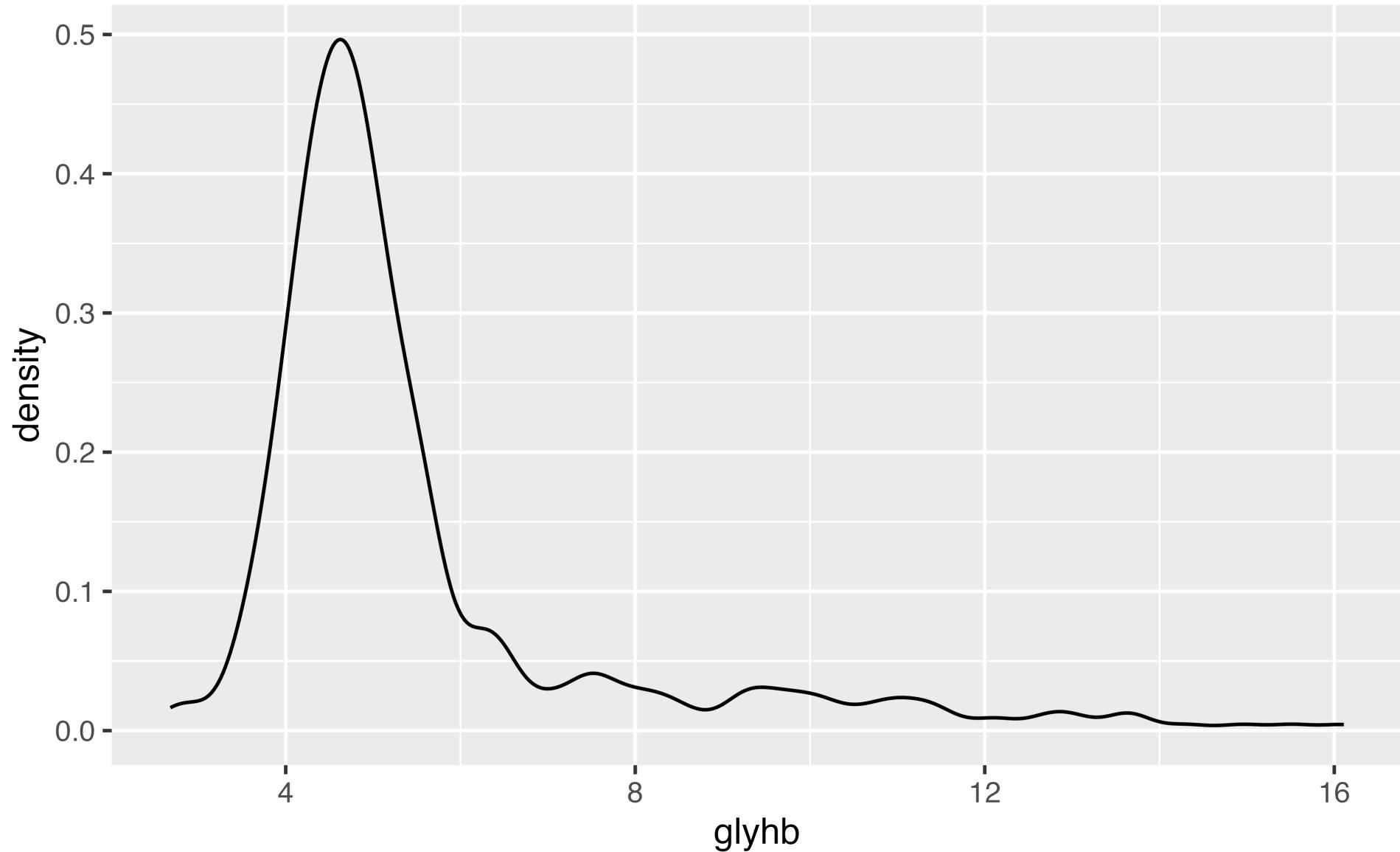
Your Turn 4

1. Make a histogram of the **glyhb** variable in **diabetes**.
2. Redo the **glyhb** plot as a density plot.

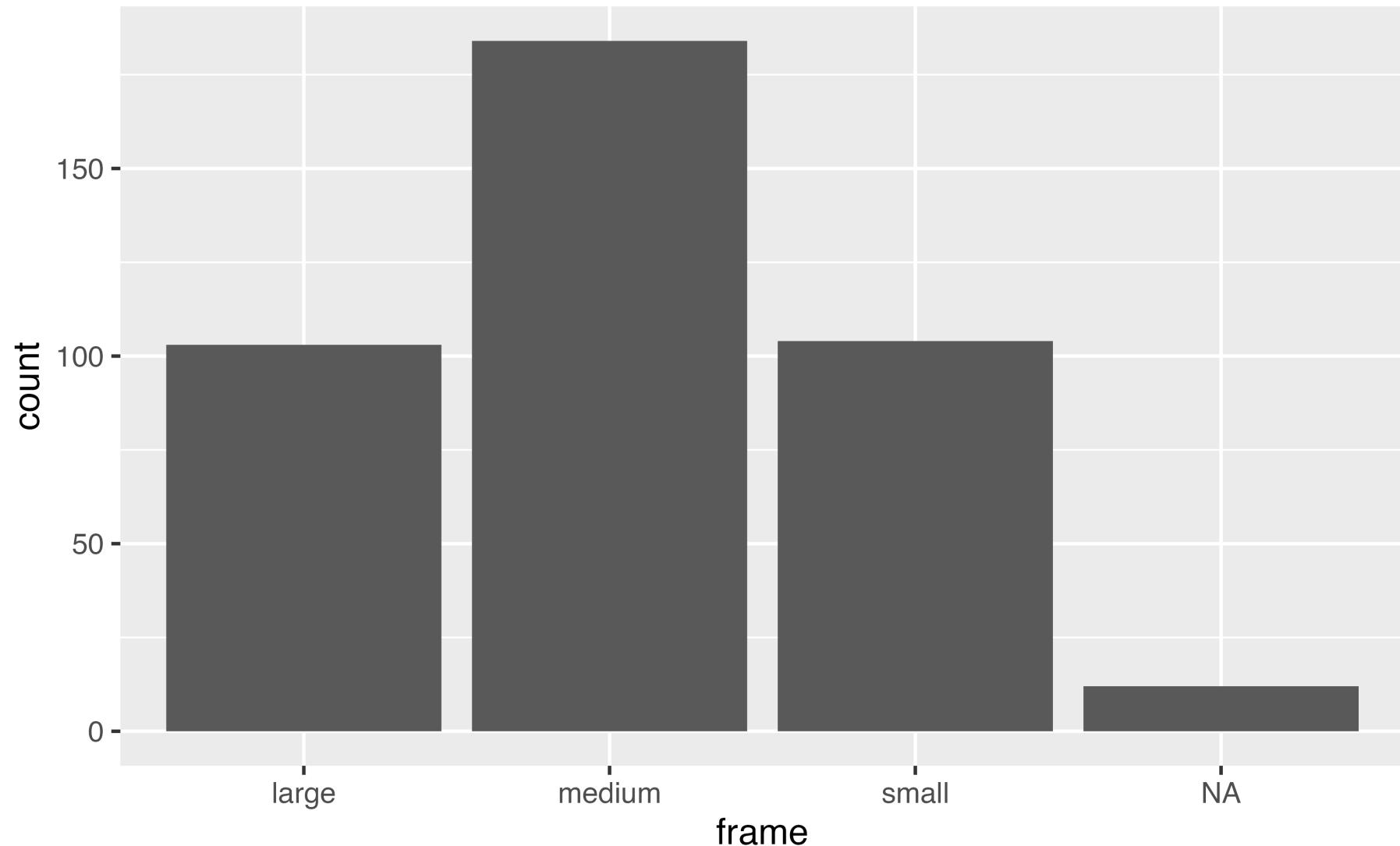
```
1 ggplot(diabetes, aes(x = glyhb)) +  
2   geom_histogram()
```



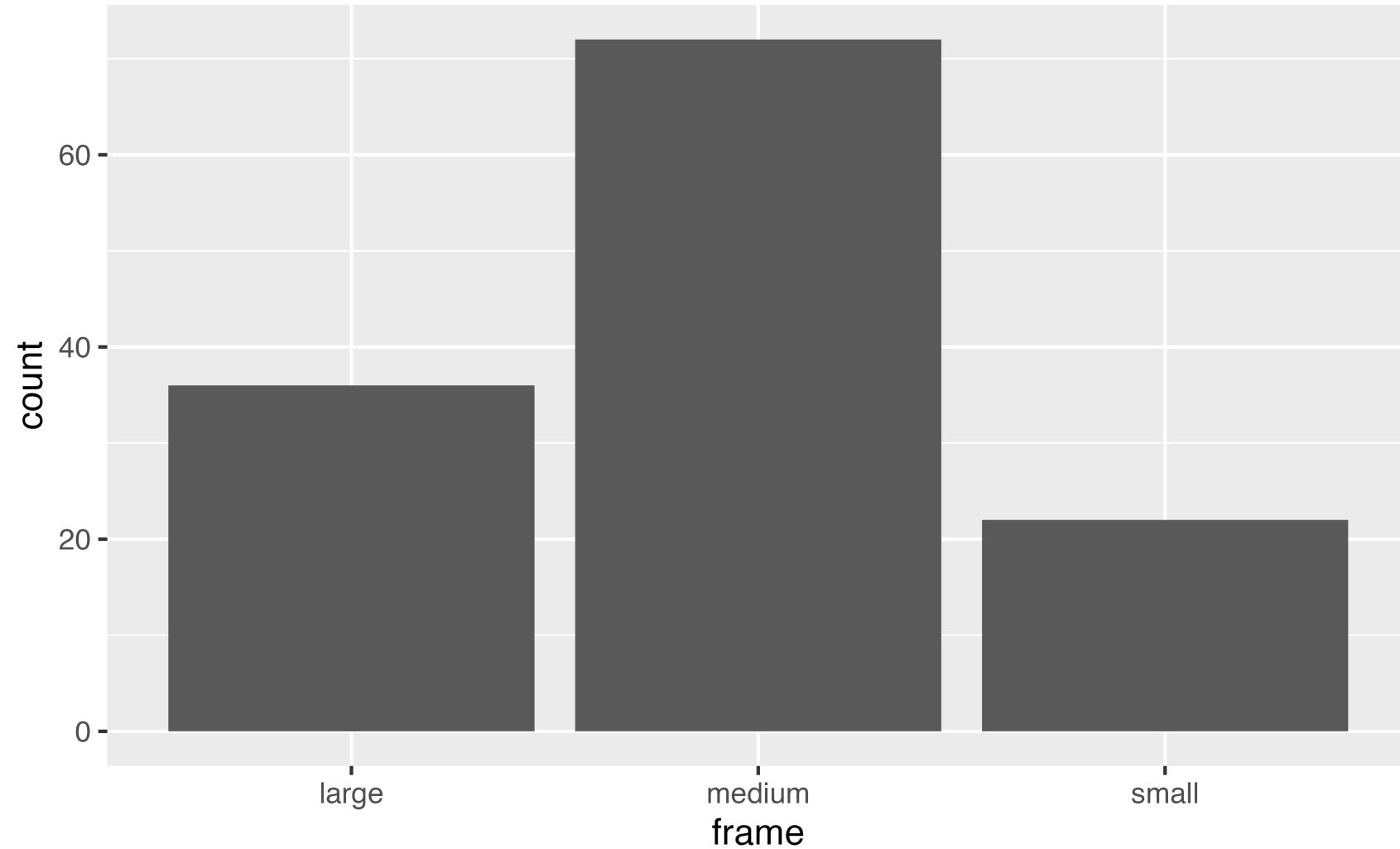
```
1 ggplot(diabetes, aes(x = glyhb)) +  
2   geom_density()
```



```
1 diabetes |>
2   ggplot(aes(x = frame)) +
3   geom_bar()
```



```
1 diabetes |>
2   drop_na() |>
3   ggplot(aes(x = frame)) +
4   geom_bar()
```

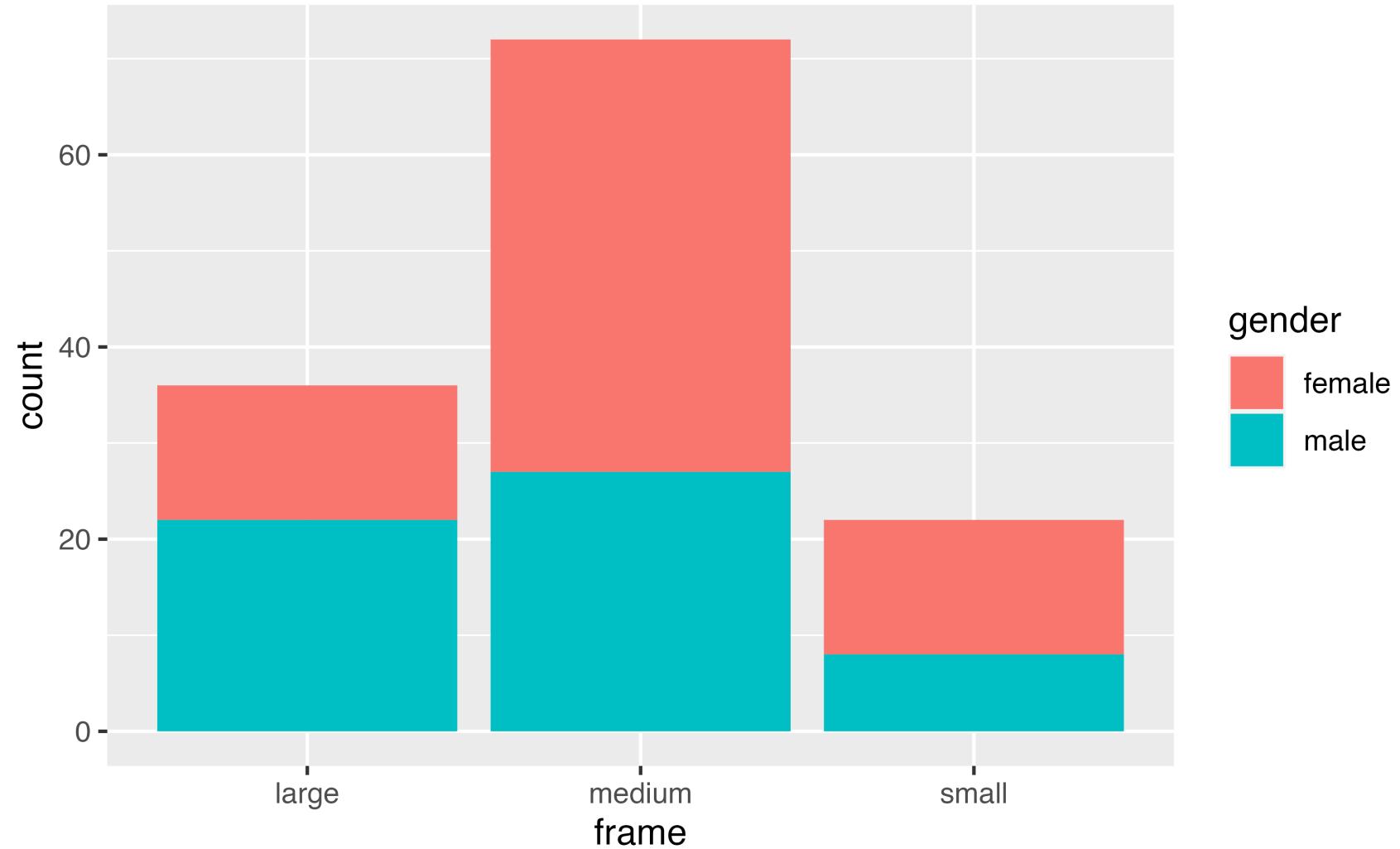


Your Turn 5

Make a bar chart of **frame** colored by **gender**.
Then, try it with the **fill** aesthetic instead of **color**.

```
1 diabetes |>
2   drop_na() |>
3   _____() +
4   _____()
```

```
1 diabetes |>
2   drop_na() |>
3   ggplot(aes(x = frame, fill = gender)) +
4   geom_bar()
```



Positions

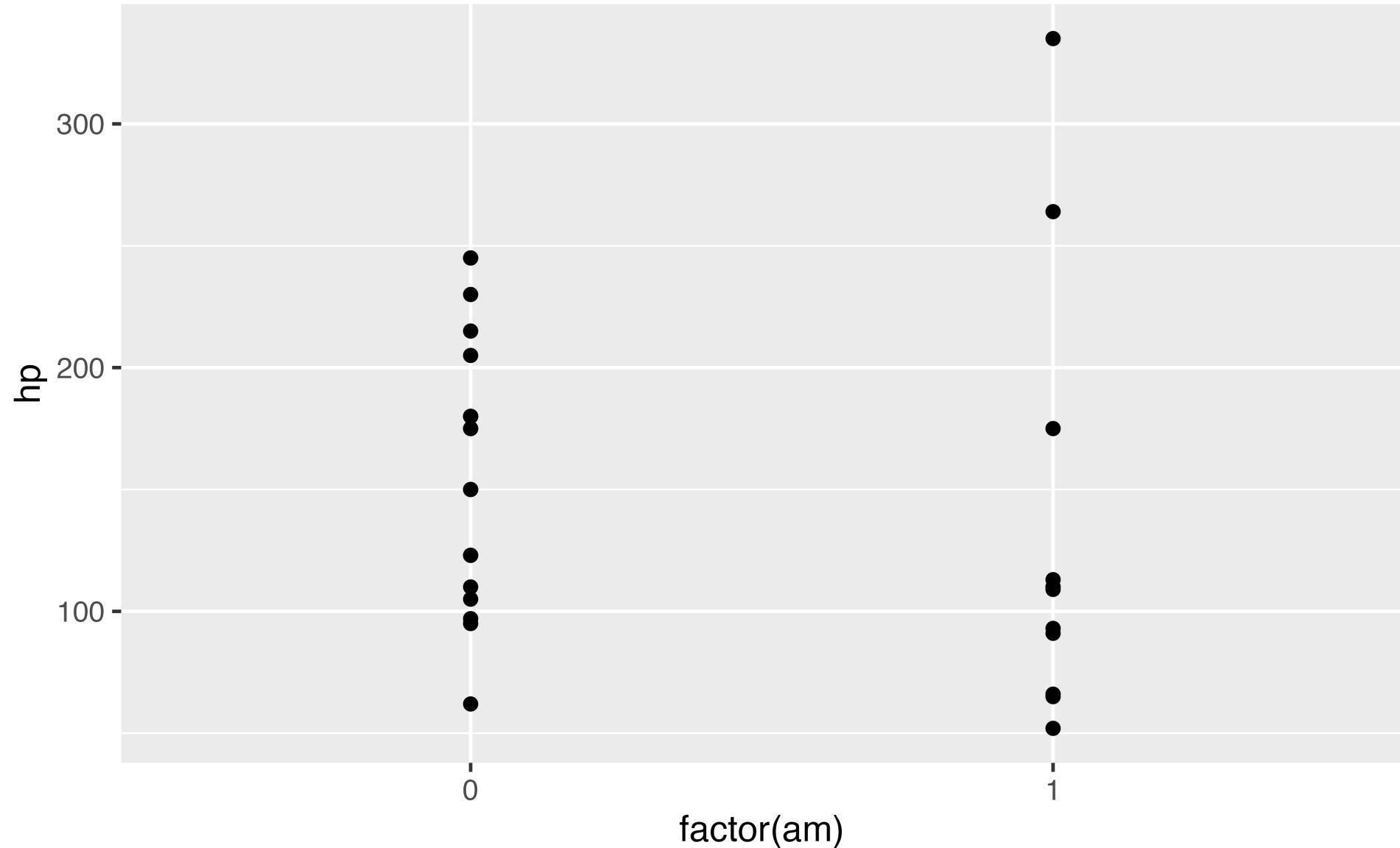
```
geom_bar(position = "  
<POSITION>")
```

When we have aesthetics mapped, how are they positioned?

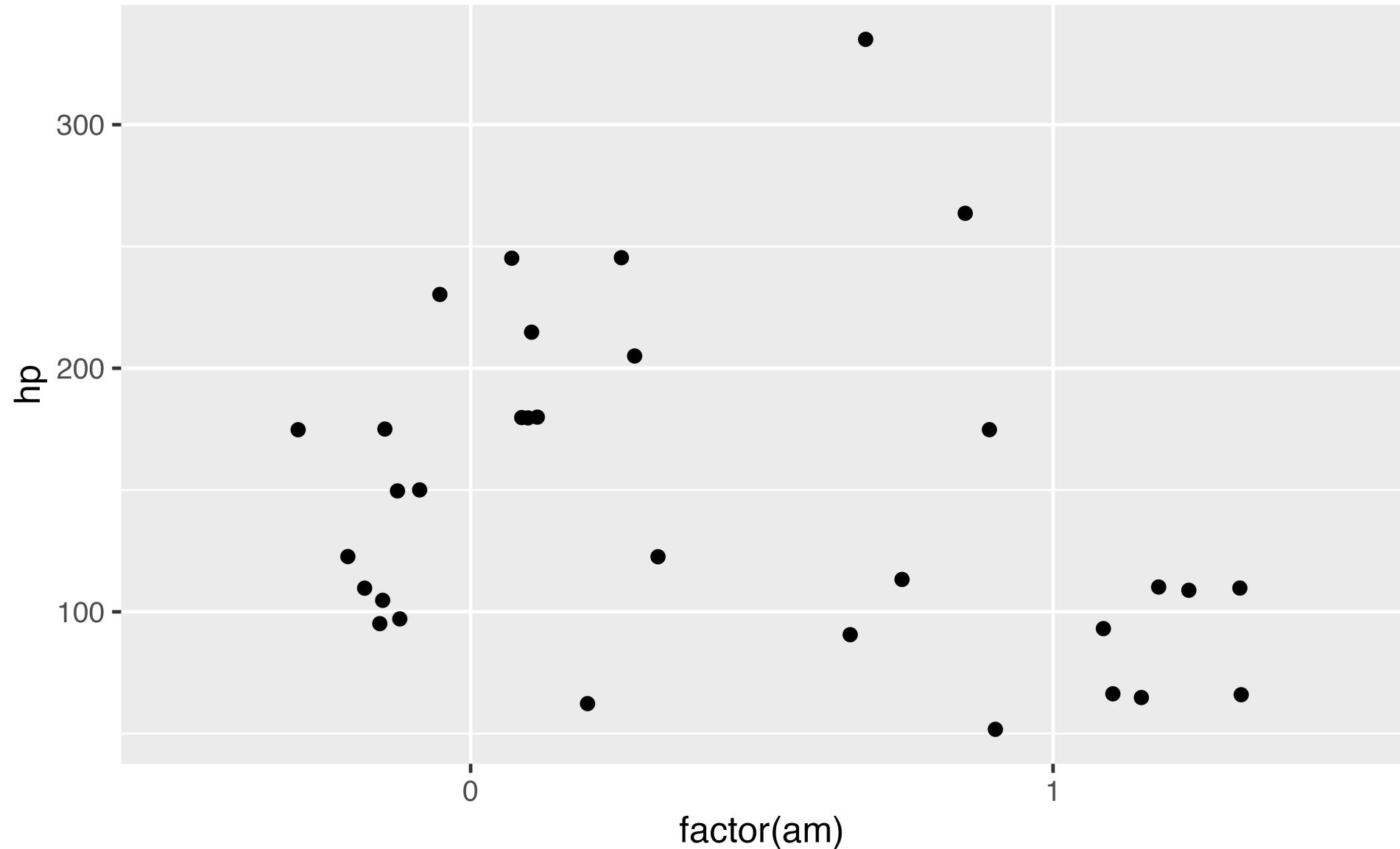
geom_bar: dodge, fill, stacked (default)

geom_point: jitter

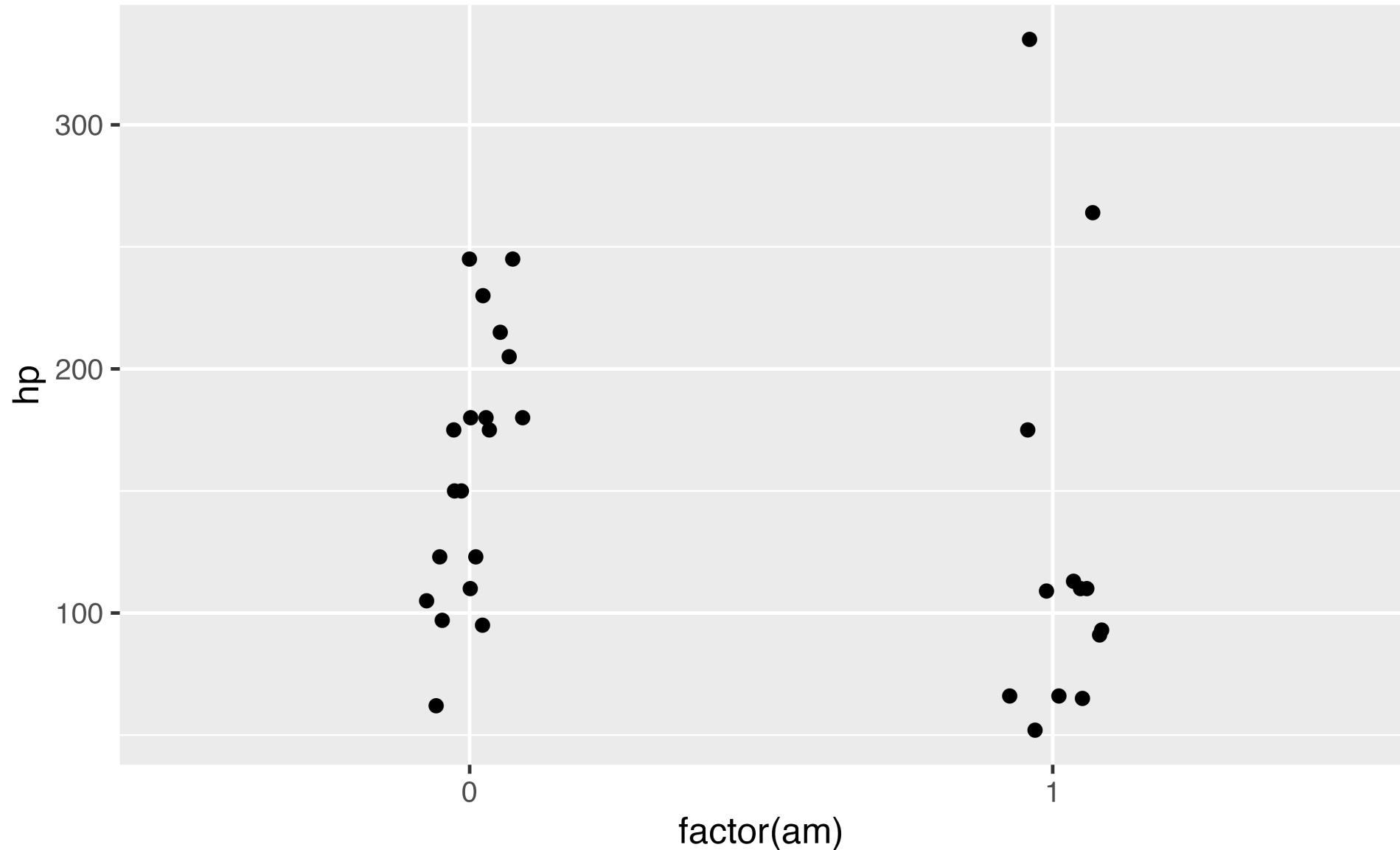
```
1 ggplot(mtcars, aes(x = factor(am), y = hp)) +  
2   geom_point()
```



```
1 ggplot(mtcars, aes(x = factor(am), y = hp)) +  
2   geom_point(position = "jitter")
```



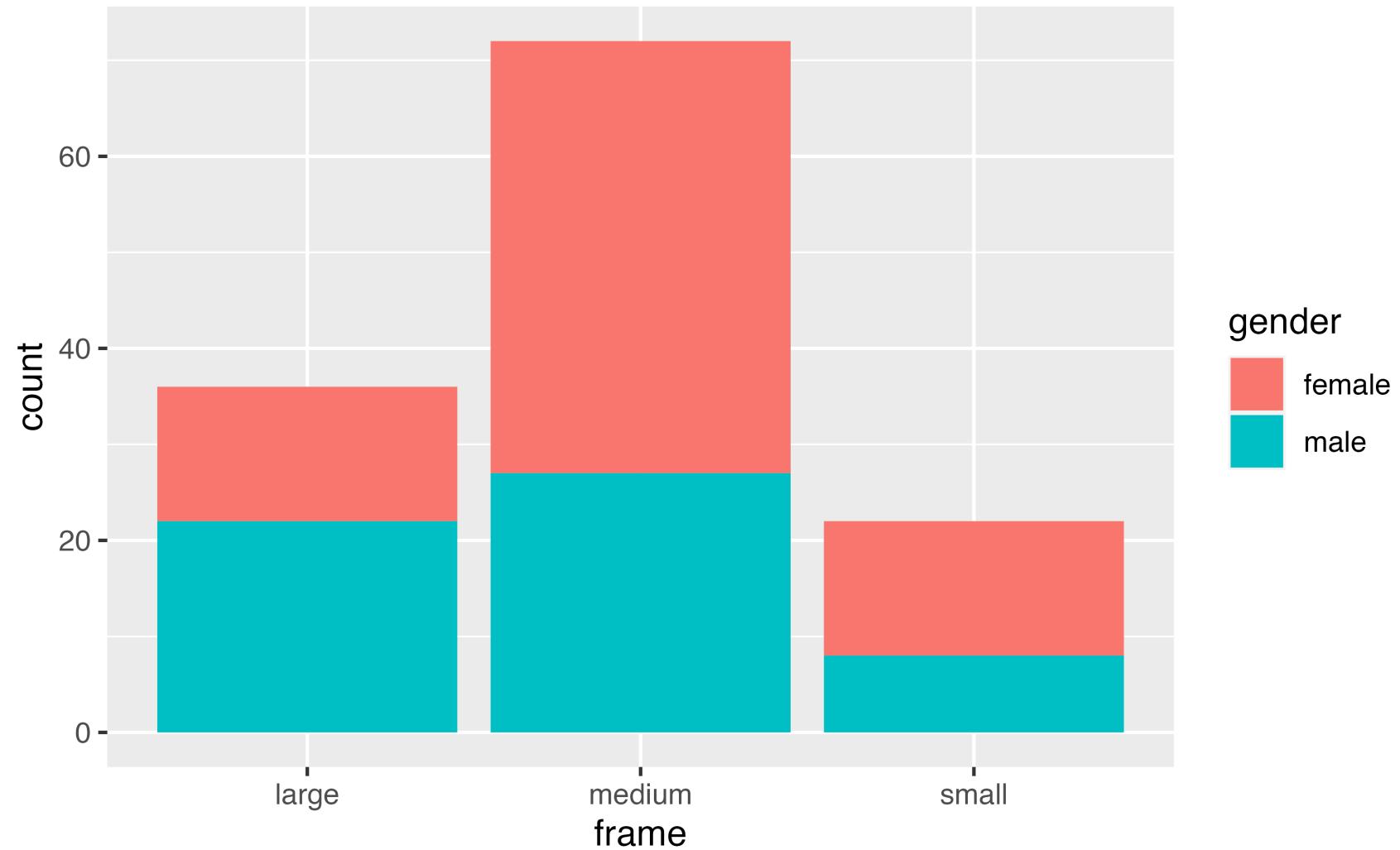
```
1 ggplot(mtcars, aes(x = factor(am), y = hp)) +  
2   geom_jitter(width = .1, height = 0)
```



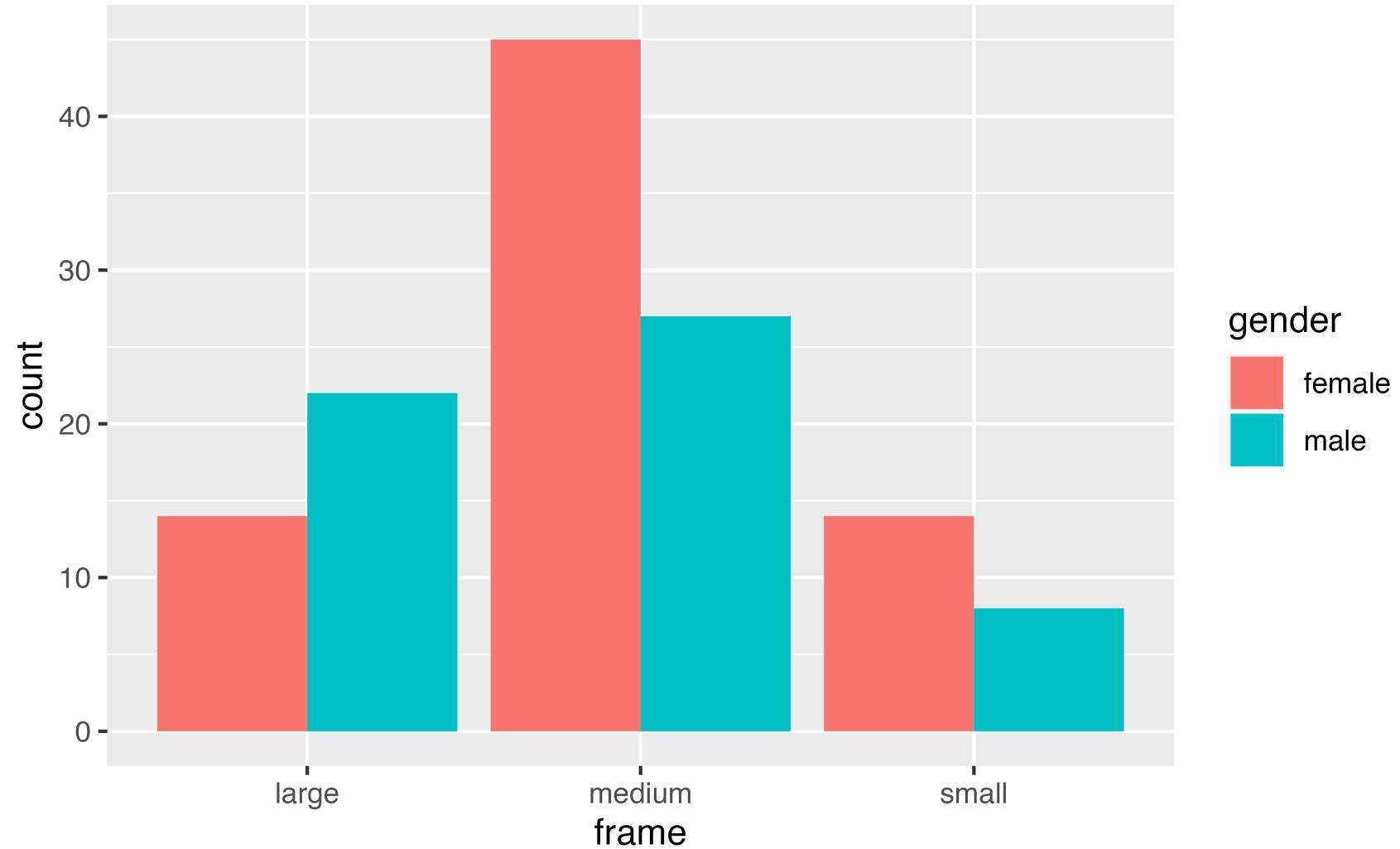
Your Turn 6

Take your code for the bar chart before (using the **fill** aesthetic). Experiment with different **position** values: "dodge", "fill", "stack"

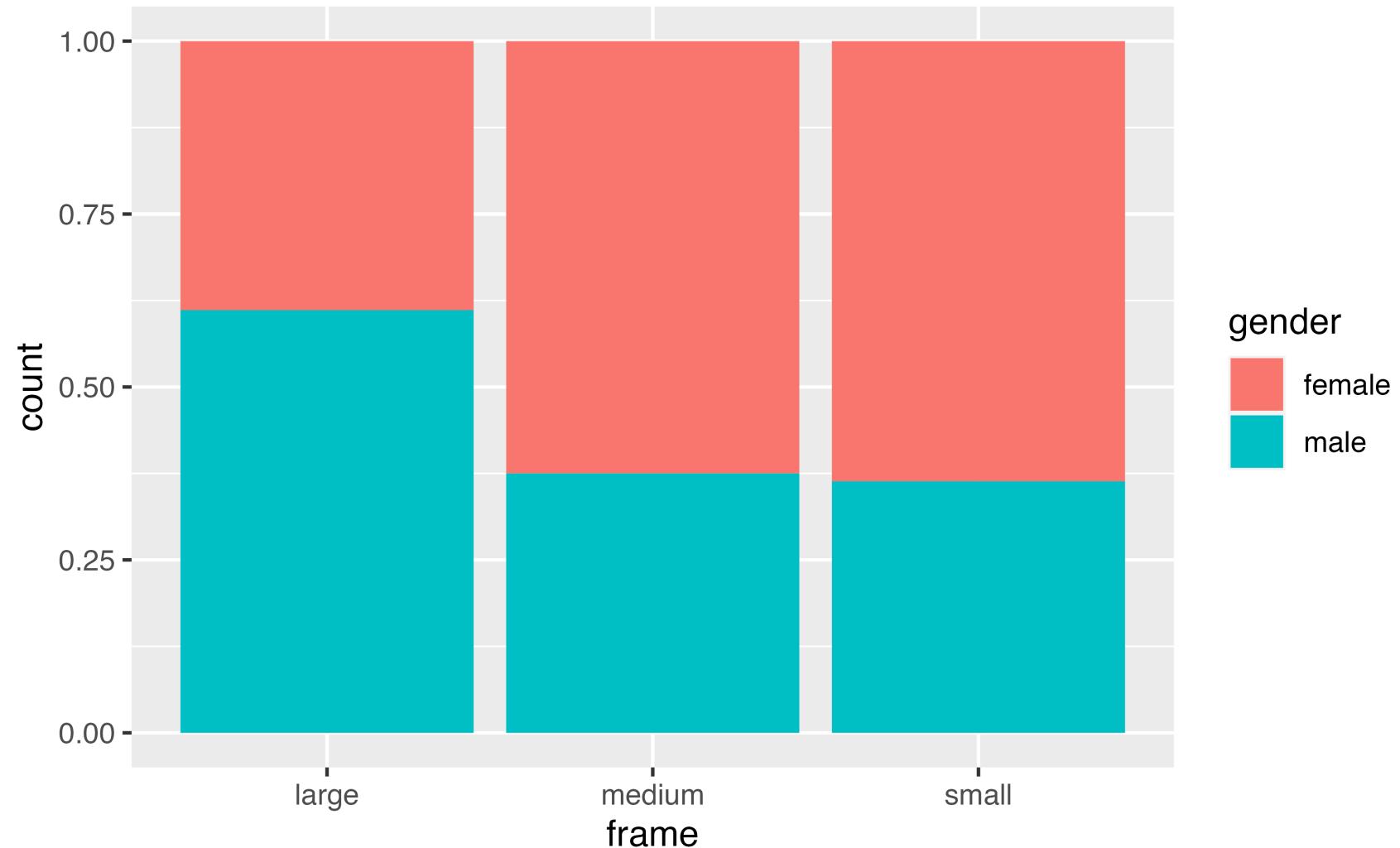
```
1 diabetes |>
2   drop_na() |>
3   ggplot(aes(x = frame, fill = gender)) +
4   geom_bar(position = "stack")
```



```
1 diabetes |>
2   drop_na() |>
3   ggplot(aes(x = frame, fill = gender)) +
4   geom_bar(position = "dodge")
```



```
1 diabetes |>
2   drop_na() |>
3   ggplot(aes(x = frame, fill = gender)) +
4   geom_bar(position = "fill")
```



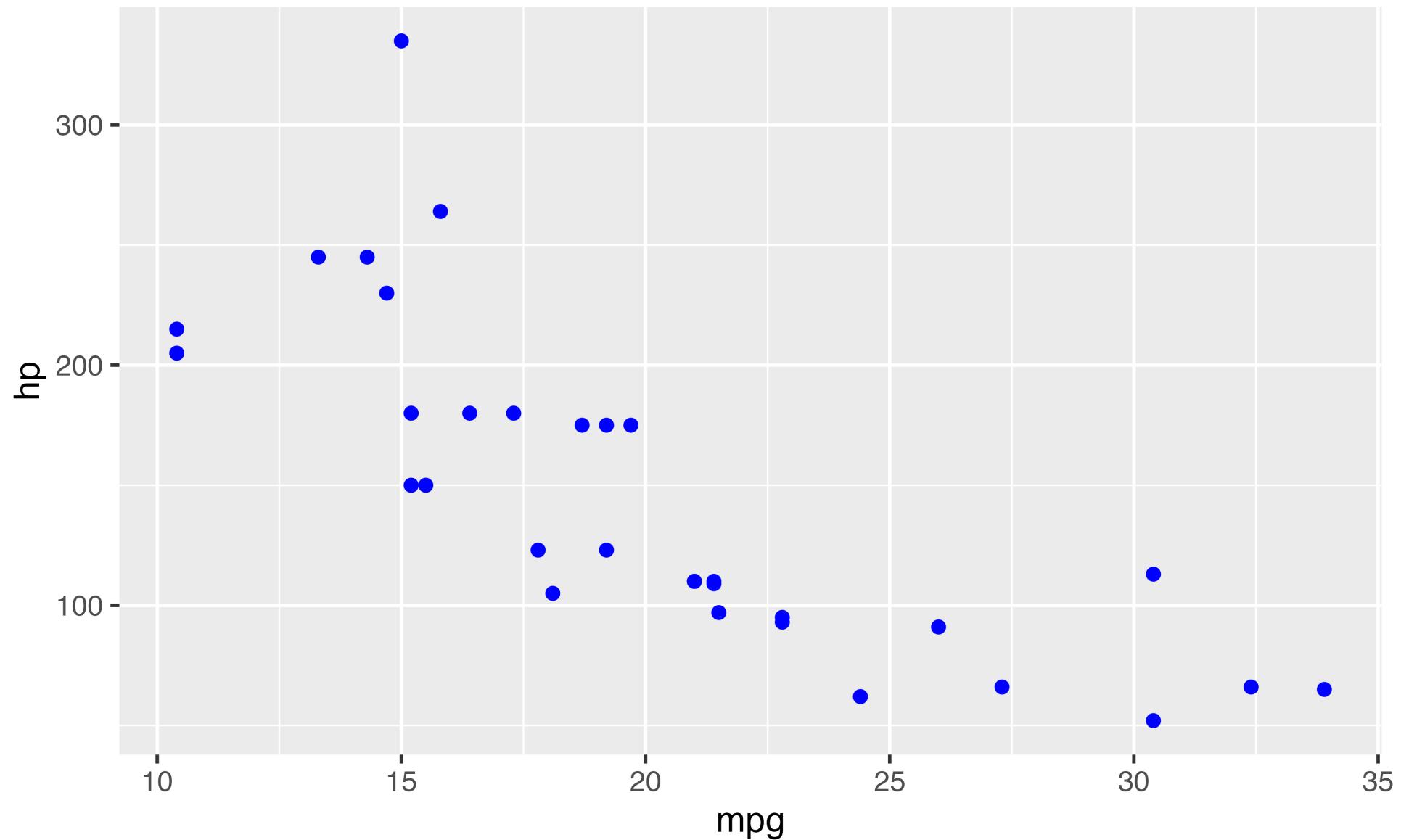
Mapping vs setting

Cool, but how do I just make everything *blue*?

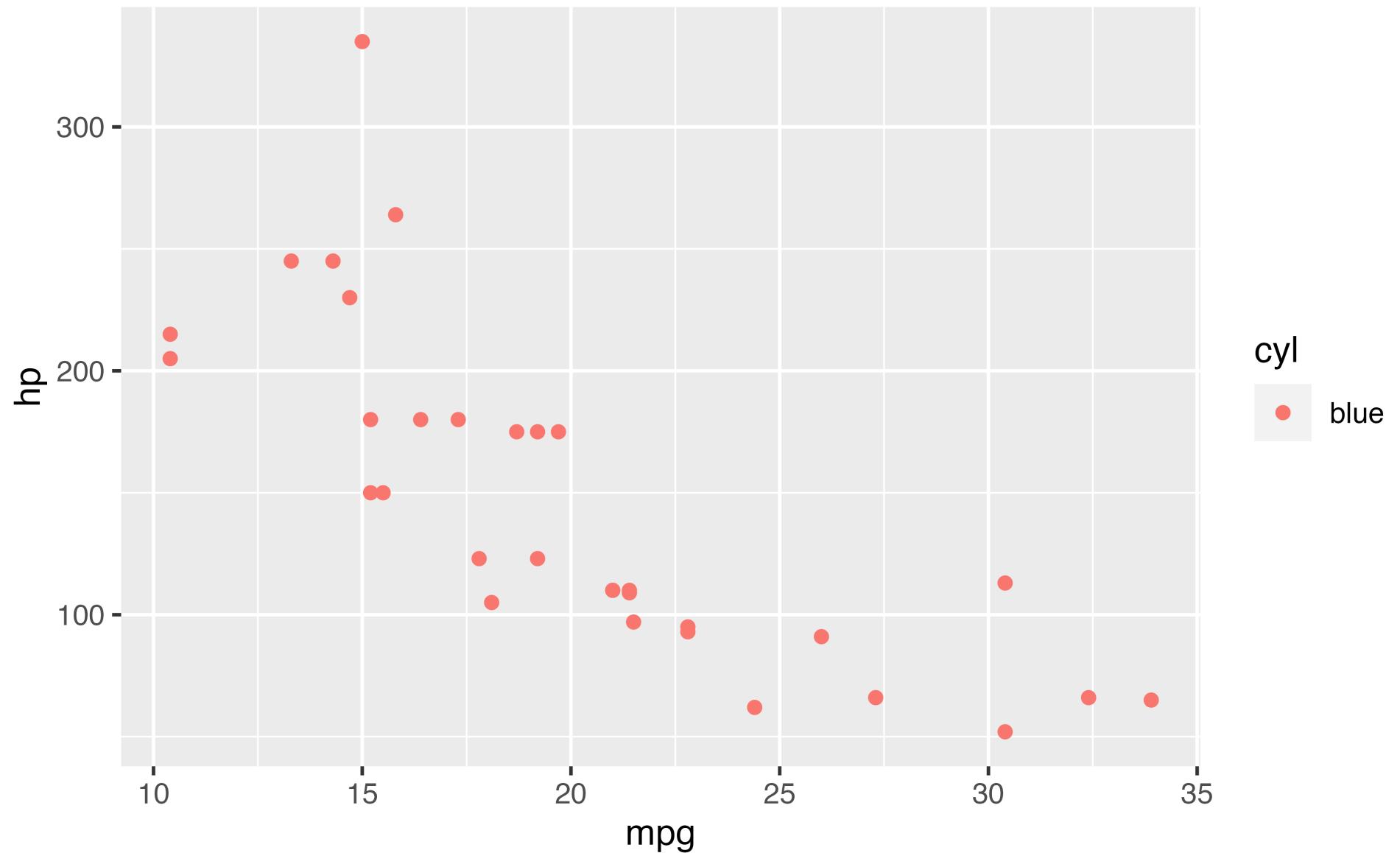
```
geom_point(aes(x, y), color = "blue")  
geom_bar(aes(x, y), fill = "blue")
```

To set a color, put it **outside** aes()

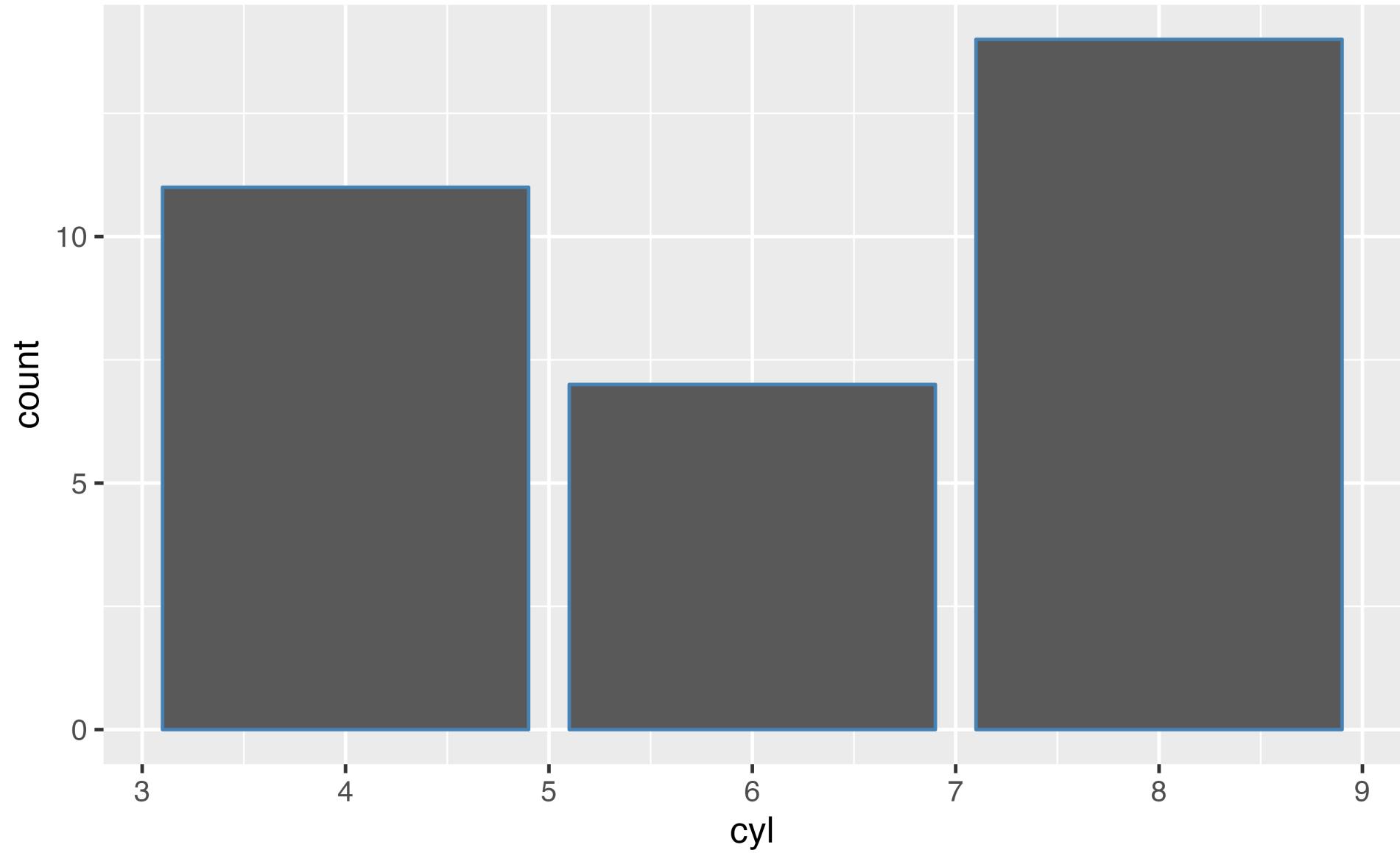
```
1 ggplot(mtcars, aes(x = mpg, y = hp, color = cyl)) +  
2   geom_point(color = "blue")
```



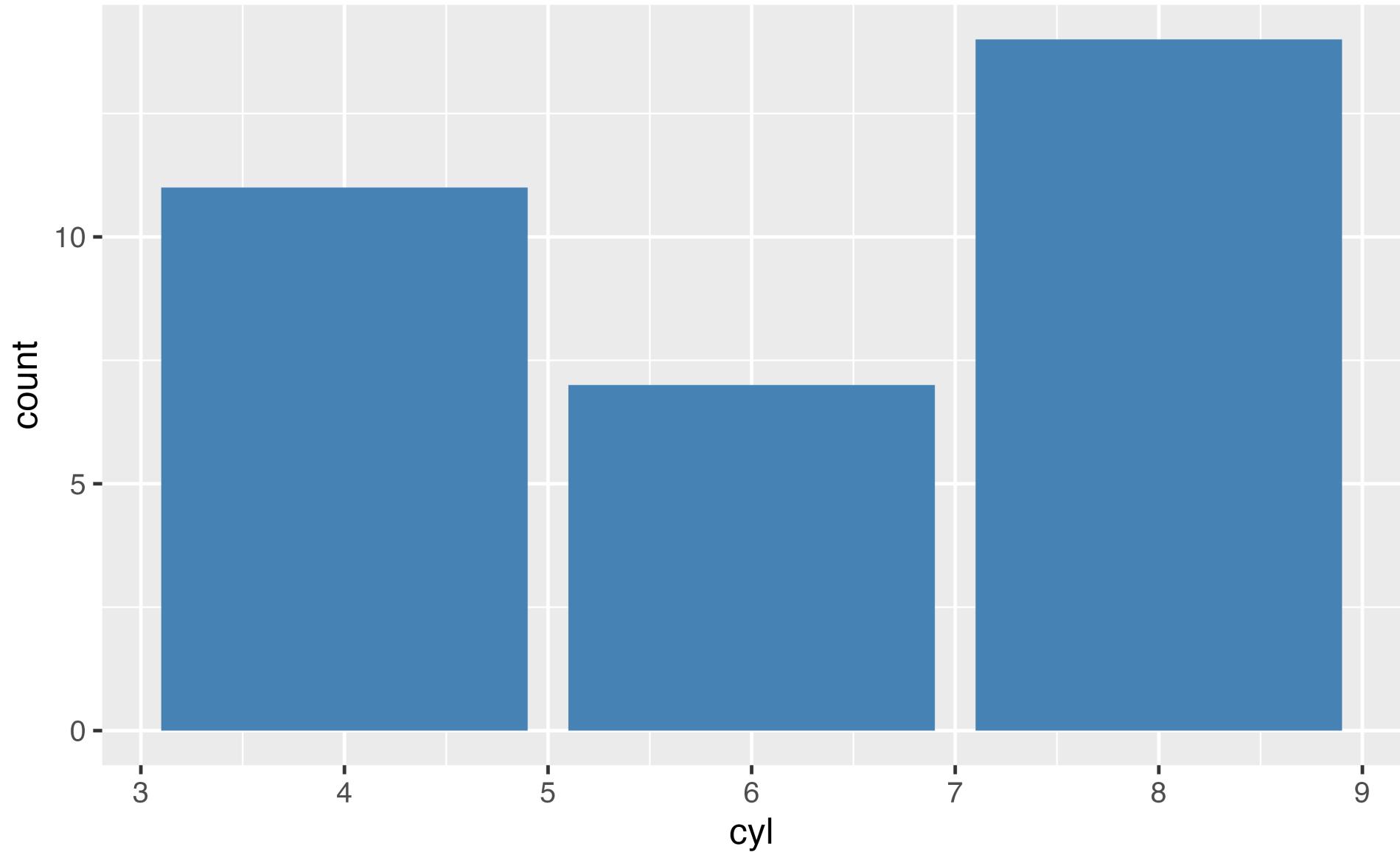
```
1 ggplot(mtcars, aes(x = mpg, y = hp, color = cyl)) +  
2   geom_point(aes(color = "blue"))
```



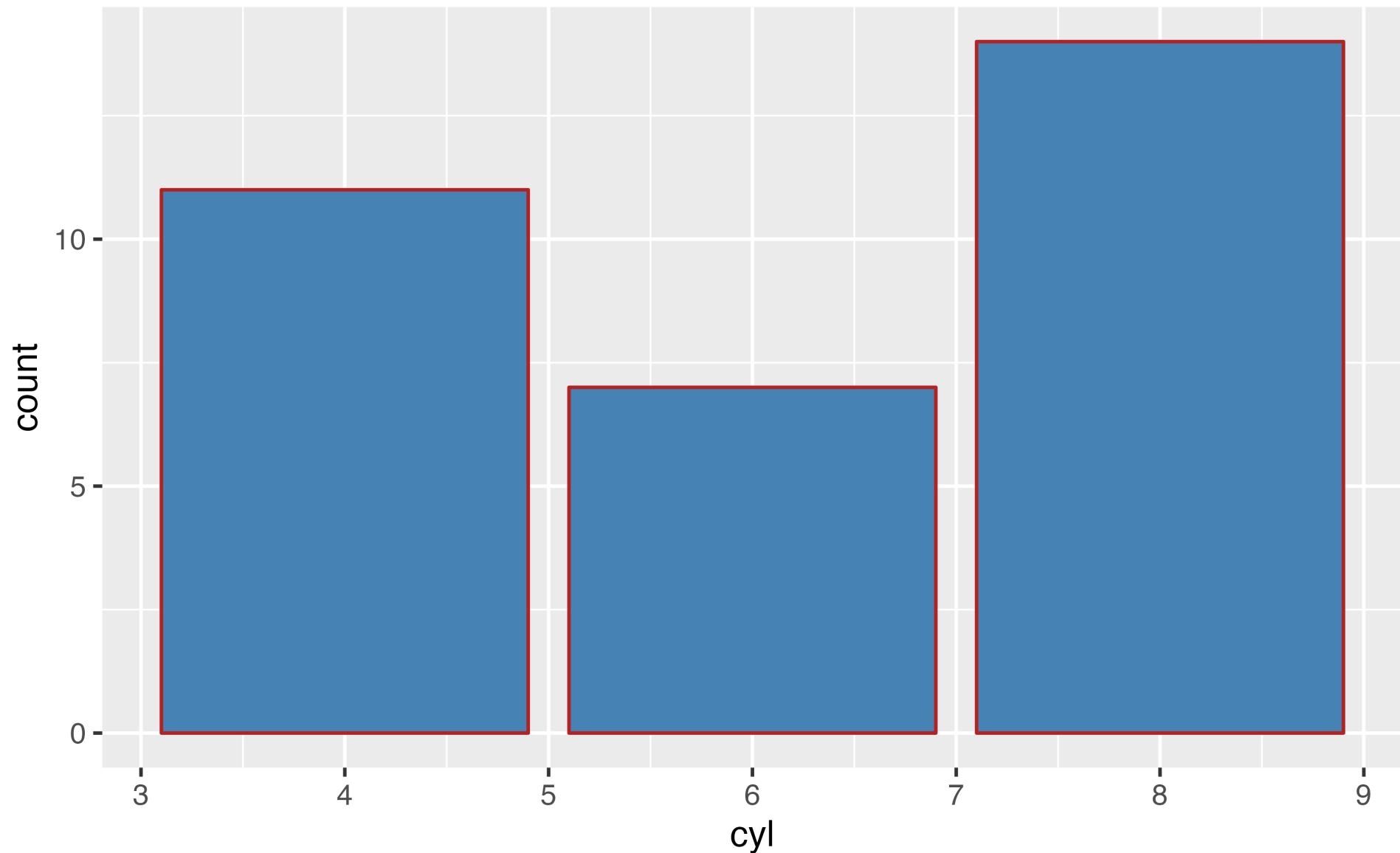
```
1 ggplot(mtcars, aes(x = cyl)) +  
2   geom_bar(color = "steelblue")
```



```
1 ggplot(mtcars, aes(x = cyl)) +  
2   geom_bar(fill = "steelblue")
```



```
1 ggplot(mtcars, aes(x = cyl)) +  
2   geom_bar(color = "firebrick", fill = "steelblue")
```



Adding layers

```
1 ggplot(data = <DATA>, mapping = aes(<MAPPINGS>)) +  
2   <GEOM_FUNCTION>() +  
3   <GEOM_FUNCTION>() +  
4   <SCALE_FUNCTION>() +  
5   <THEME_FUNCTION>()
```

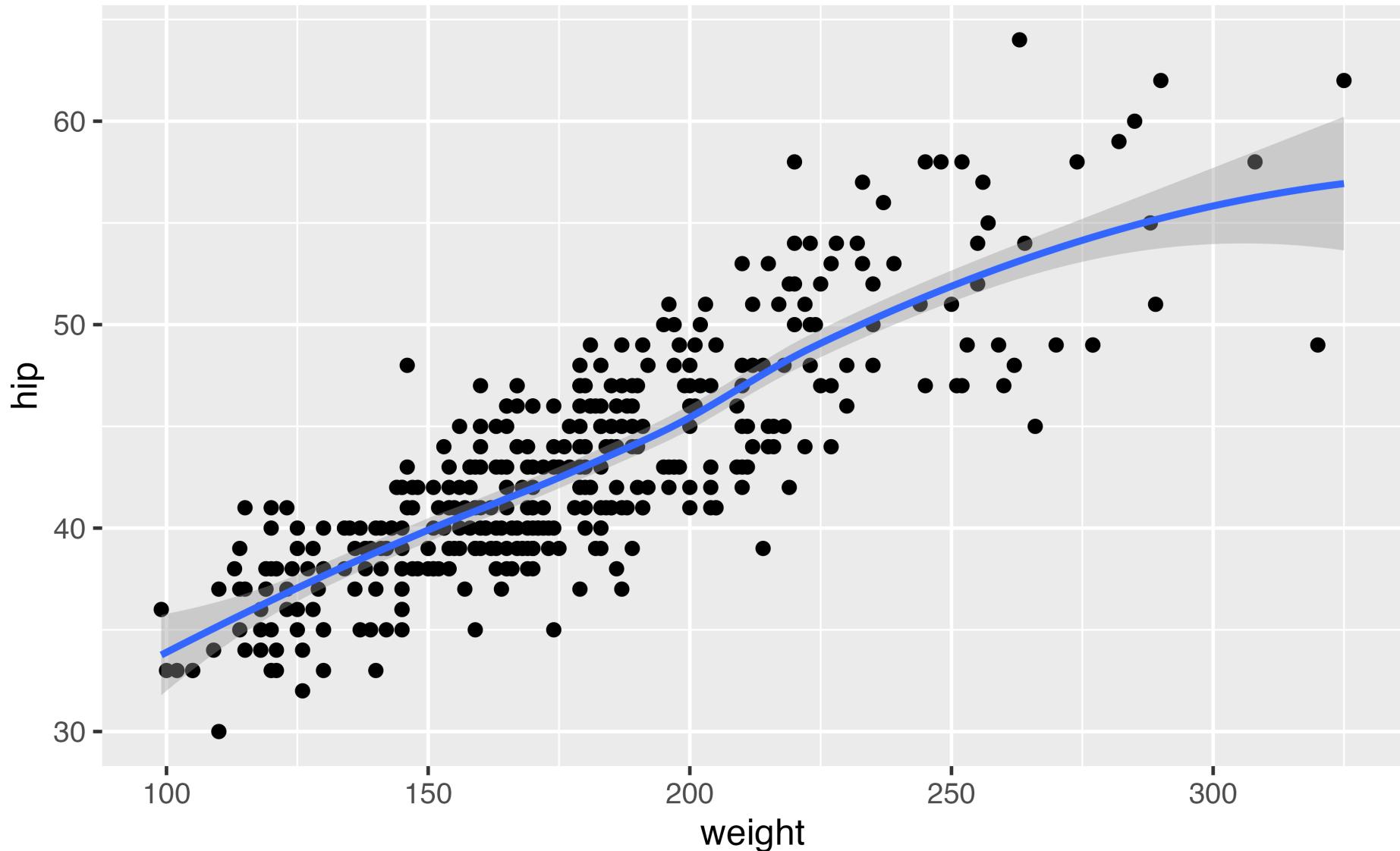
Your Turn 7

Run the code after every change you make.

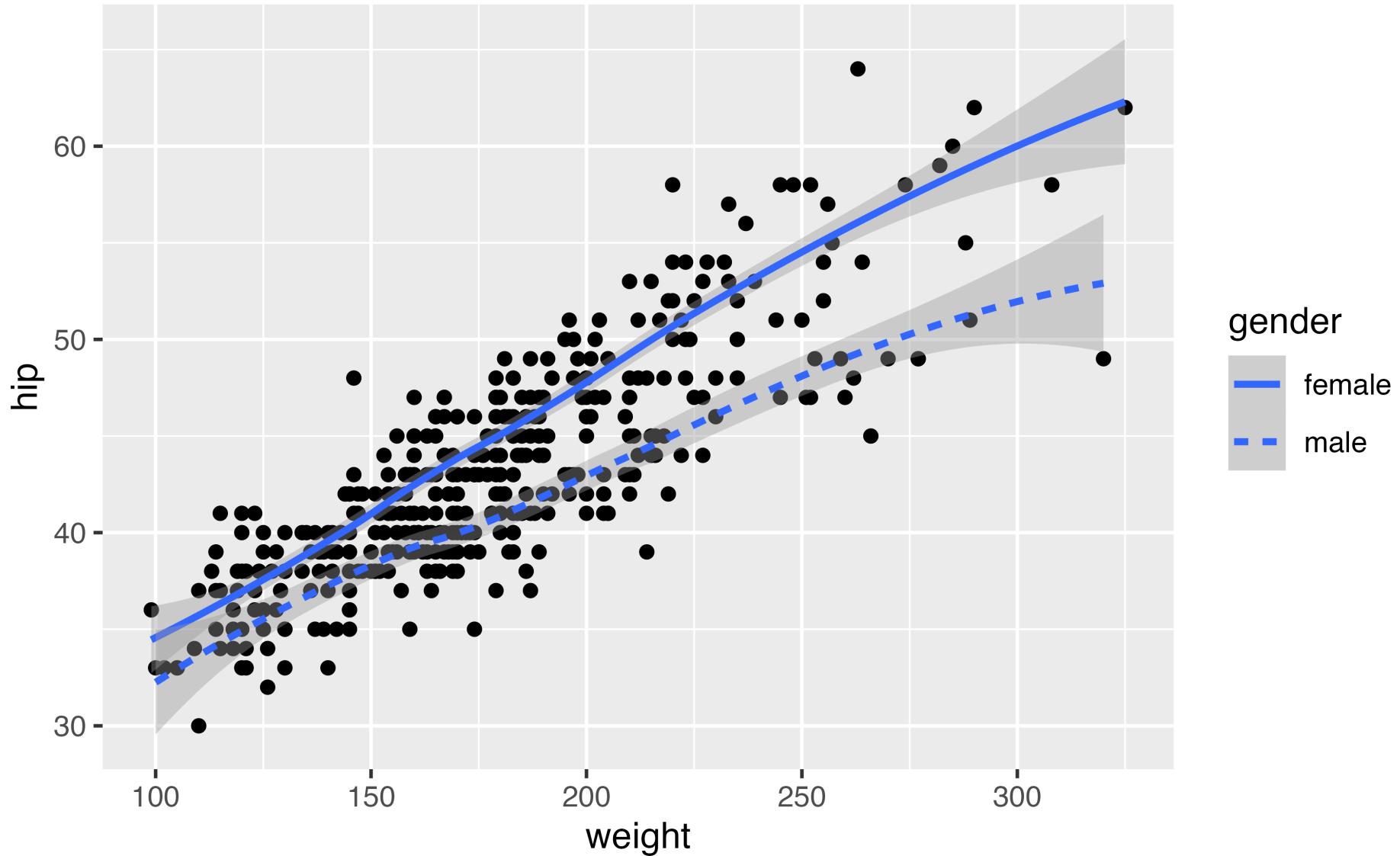
1. Predict what this code will do. Then run it.
2. Add a `linetype` aesthetic for `gender`. Run it again.
3. Set the color of `geom_smooth()` to “black”
4. Add `se = FALSE` to the `geom_smooth()`
5. It’s hard to see the lines well now. How about setting `alpha = .2` in `geom_point()`?
6. Jitter the points. You can either change the geom or change the `position` argument.
7. Add another layer, `theme_bw()`. Remember to use `+`.

```
1 ggplot(diabetes, aes(weight, hip)) +  
2   geom_point() +  
3   geom_smooth()
```

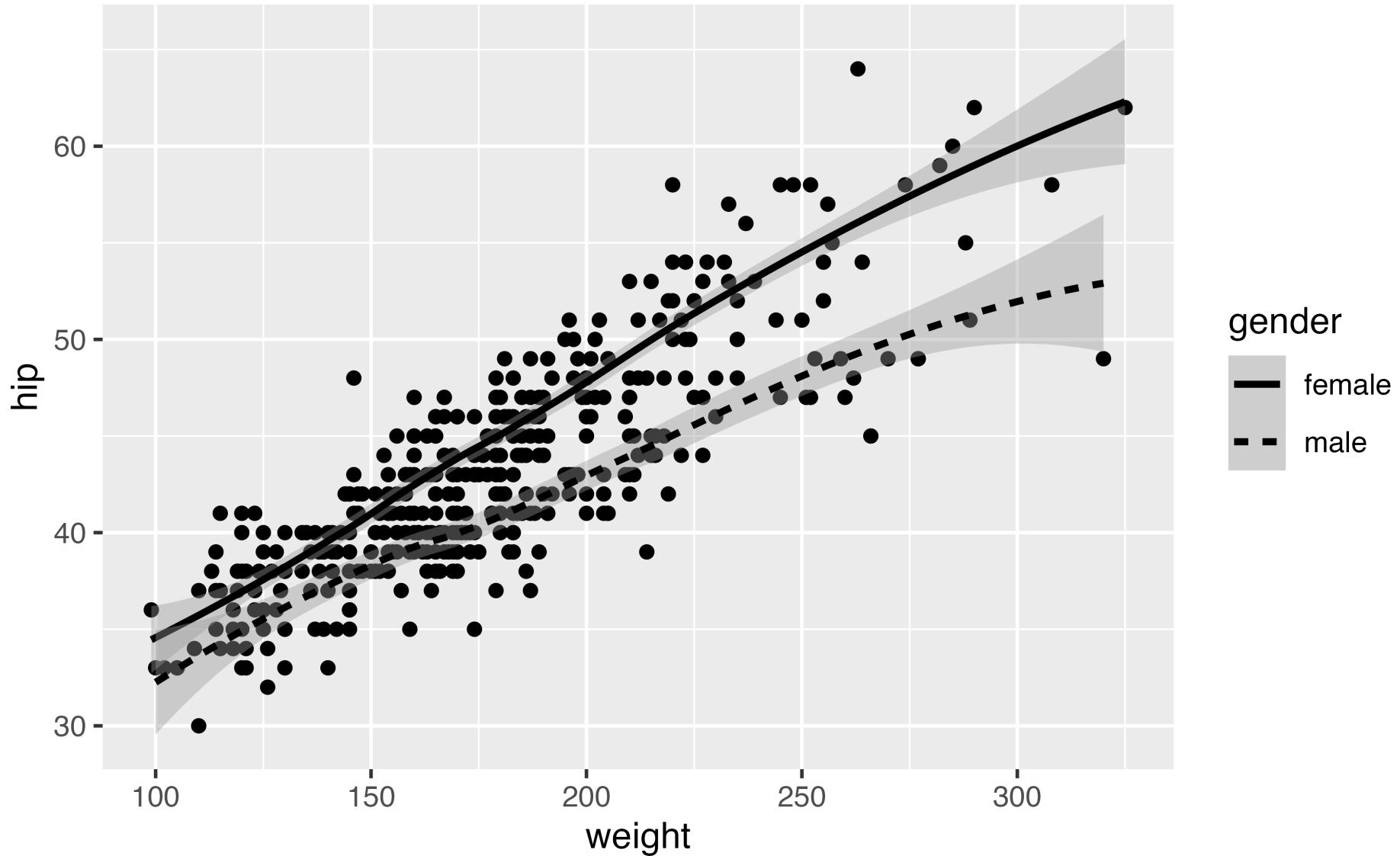
```
1 ggplot(diabetes, aes(weight, hip)) +  
2   geom_point() +  
3   geom_smooth()
```



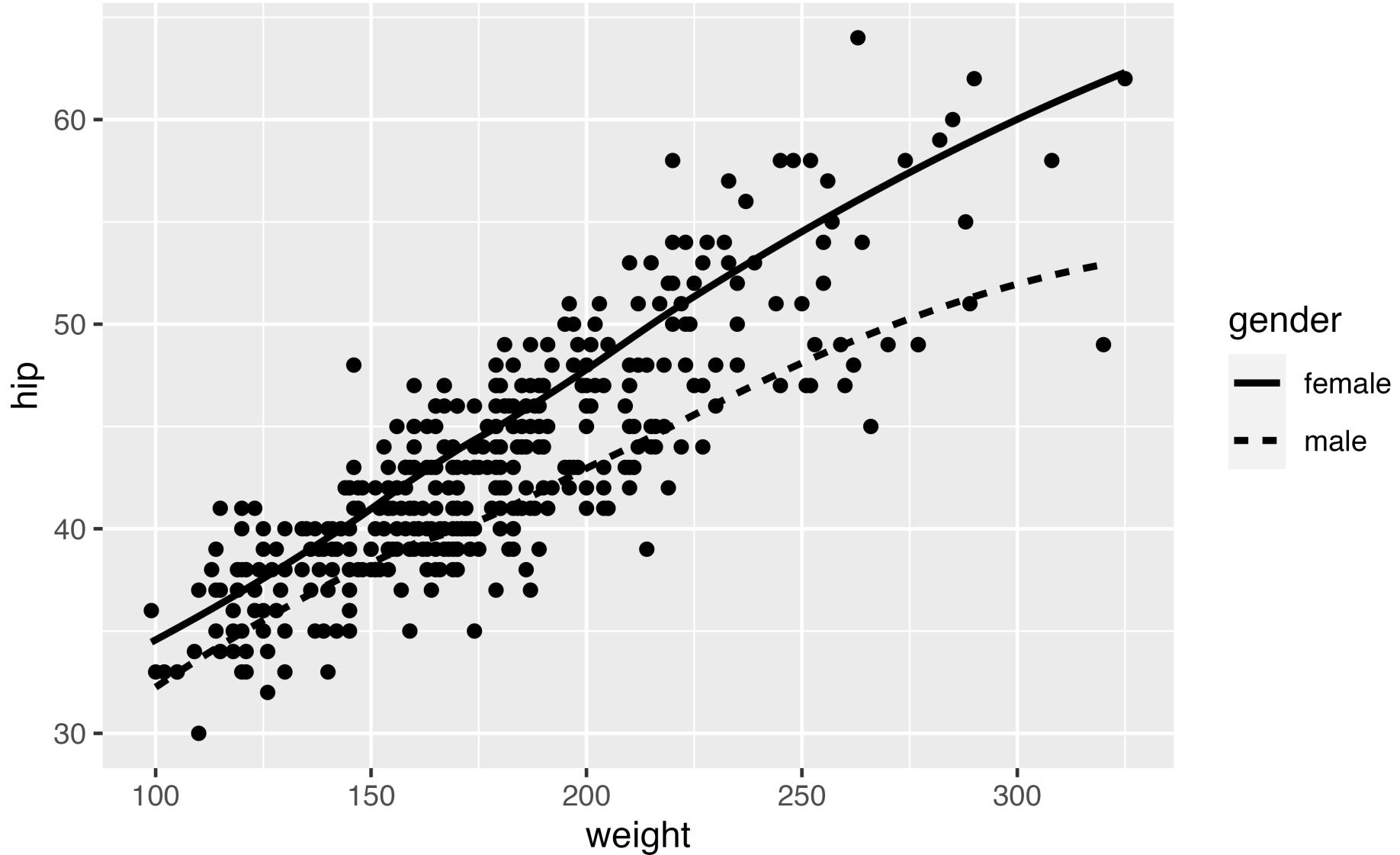
```
1 ggplot(diabetes, aes(weight, hip)) +  
2   geom_point() +  
3   geom_smooth(aes(linetype = gender))
```



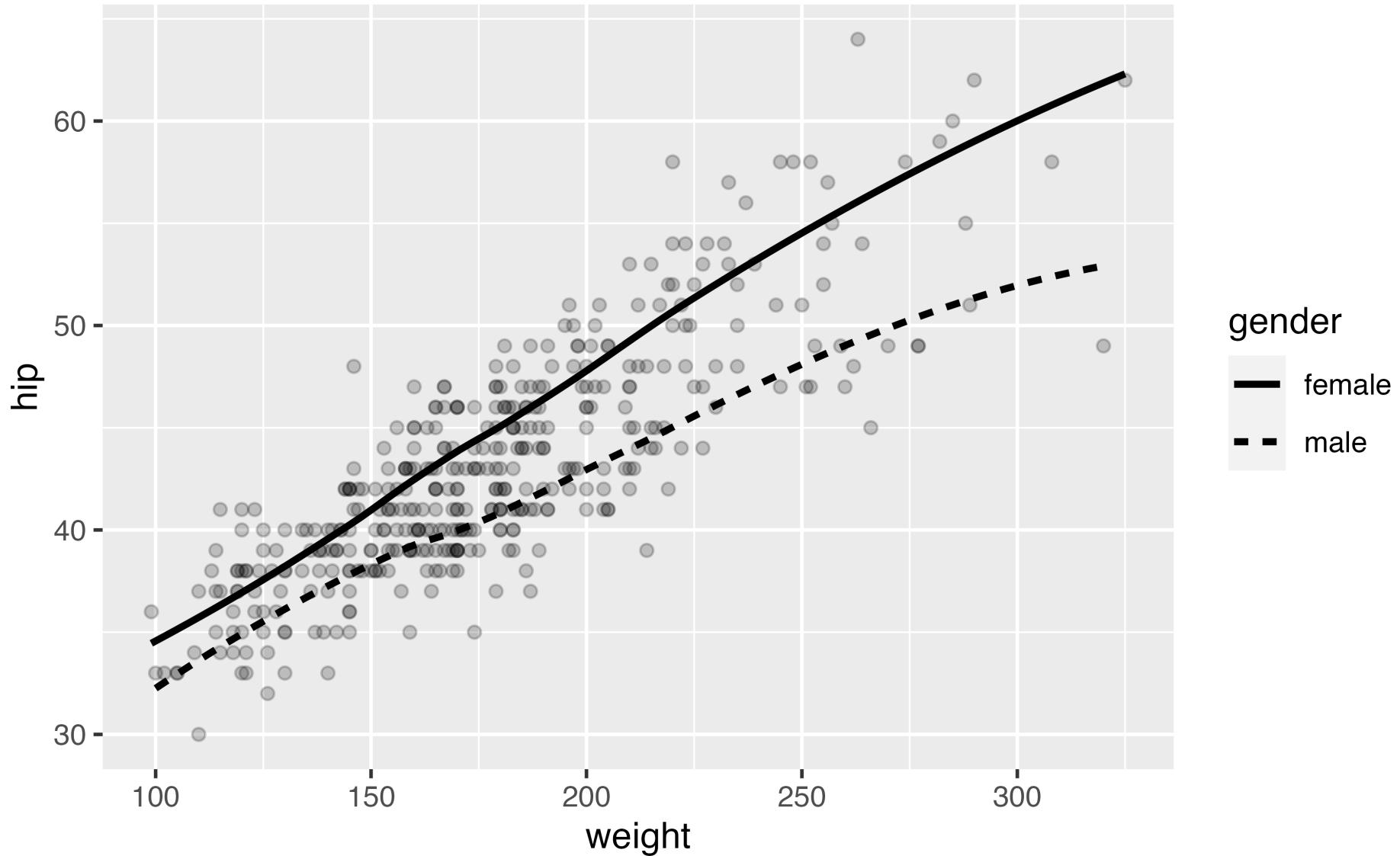
```
1 ggplot(diabetes, aes(weight, hip)) +  
2   geom_point() +  
3   geom_smooth(aes(linetype = gender), color = "black")
```



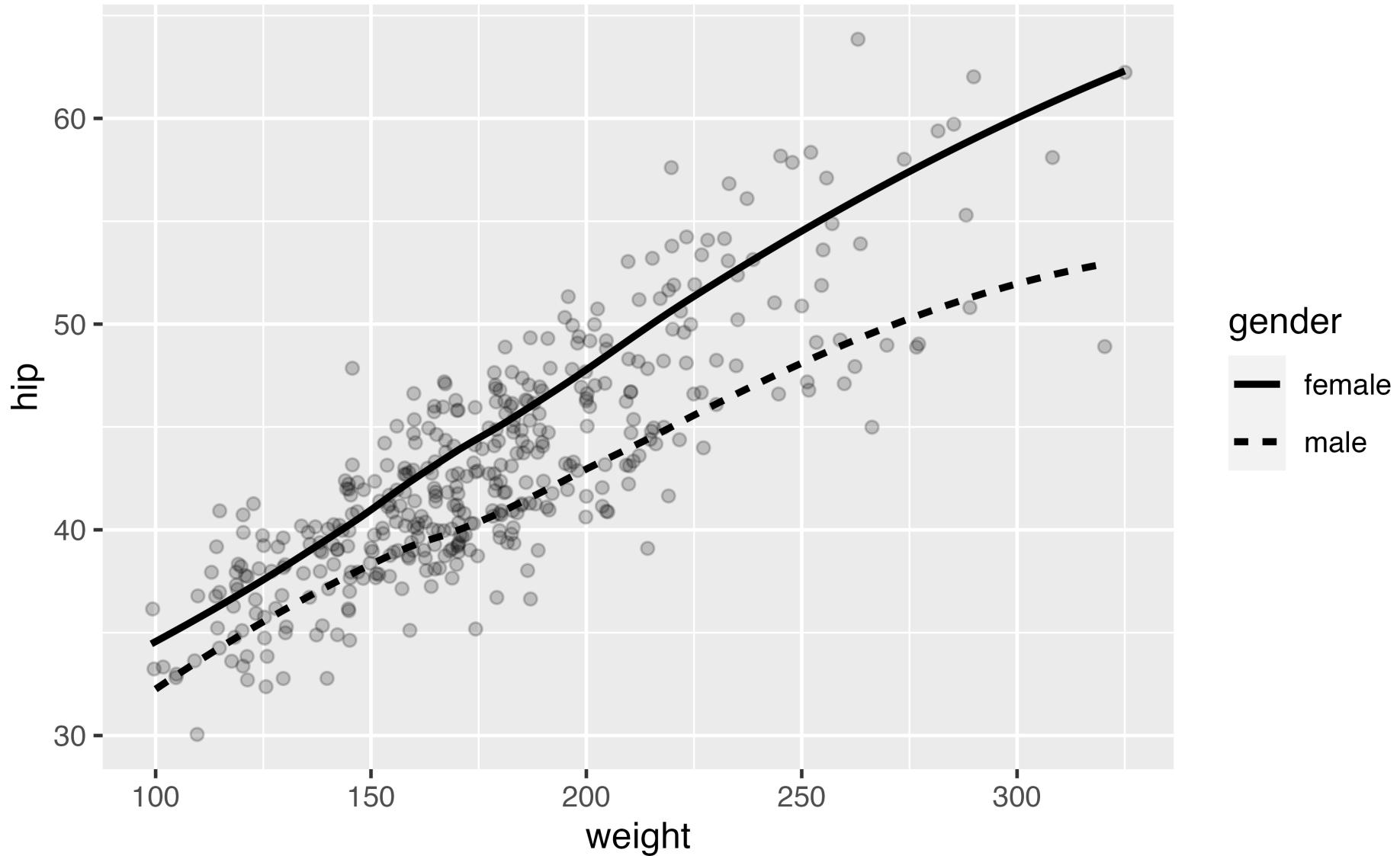
```
1 ggplot(diabetes, aes(weight, hip)) +  
2   geom_point() +  
3   geom_smooth(aes(linetype = gender), color = "black", se = FALSE)
```



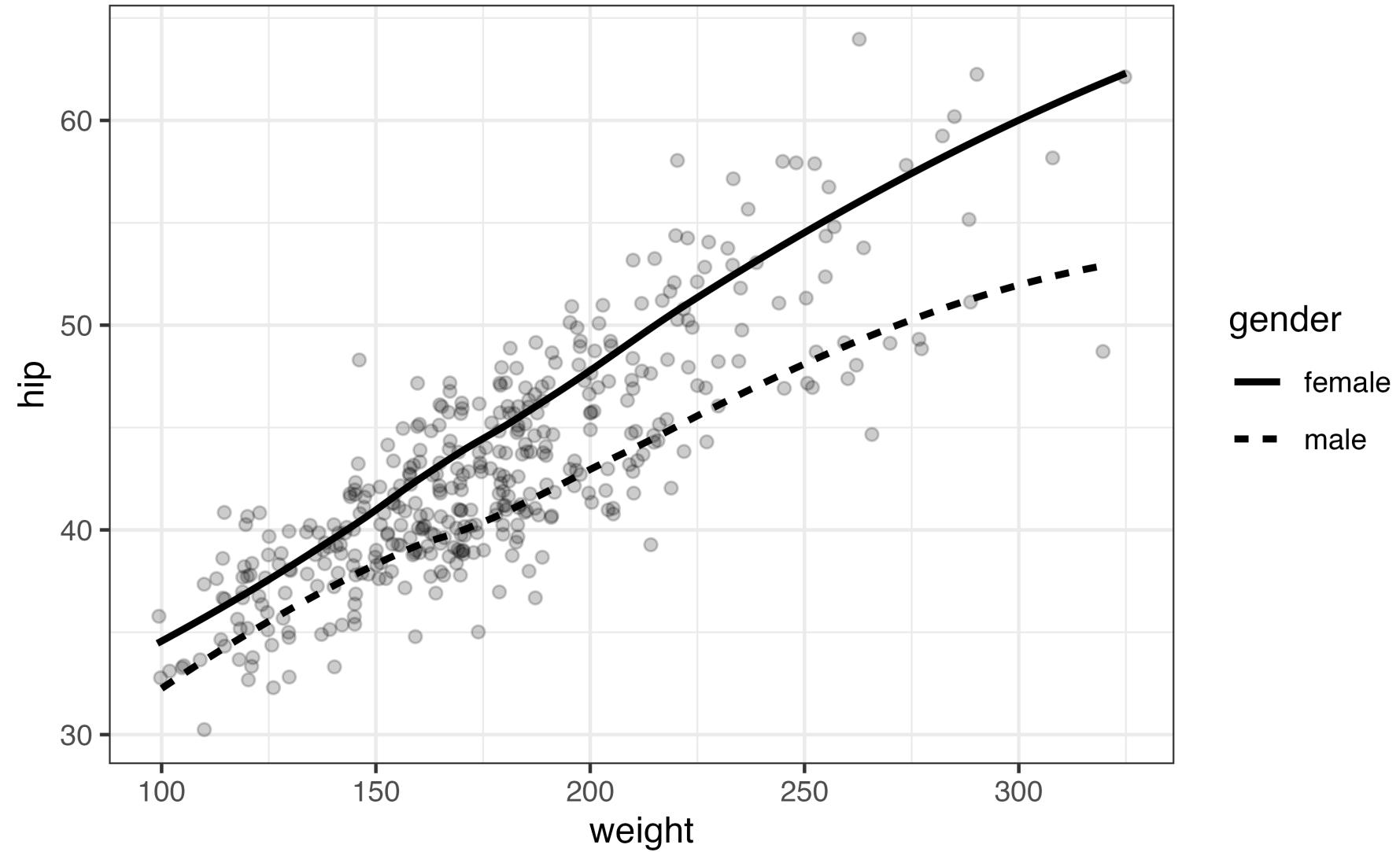
```
1 ggplot(diabetes, aes(weight, hip)) +  
2   geom_point(alpha = .2) +  
3   geom_smooth(aes(linetype = gender), color = "black", se = FALSE)
```



```
1 ggplot(diabetes, aes(weight, hip)) +  
2   geom_jitter(alpha = .2) +  
3   geom_smooth(aes(linetype = gender), color = "black", se = FALSE)
```



```
1 ggplot(diabetes, aes(weight, hip)) +  
2   geom_jitter(alpha = .2) +  
3   geom_smooth(aes(linetype = gender), color = "black", se = FALSE) +  
4   theme_bw()
```



Facets

Easily split your plot into panels

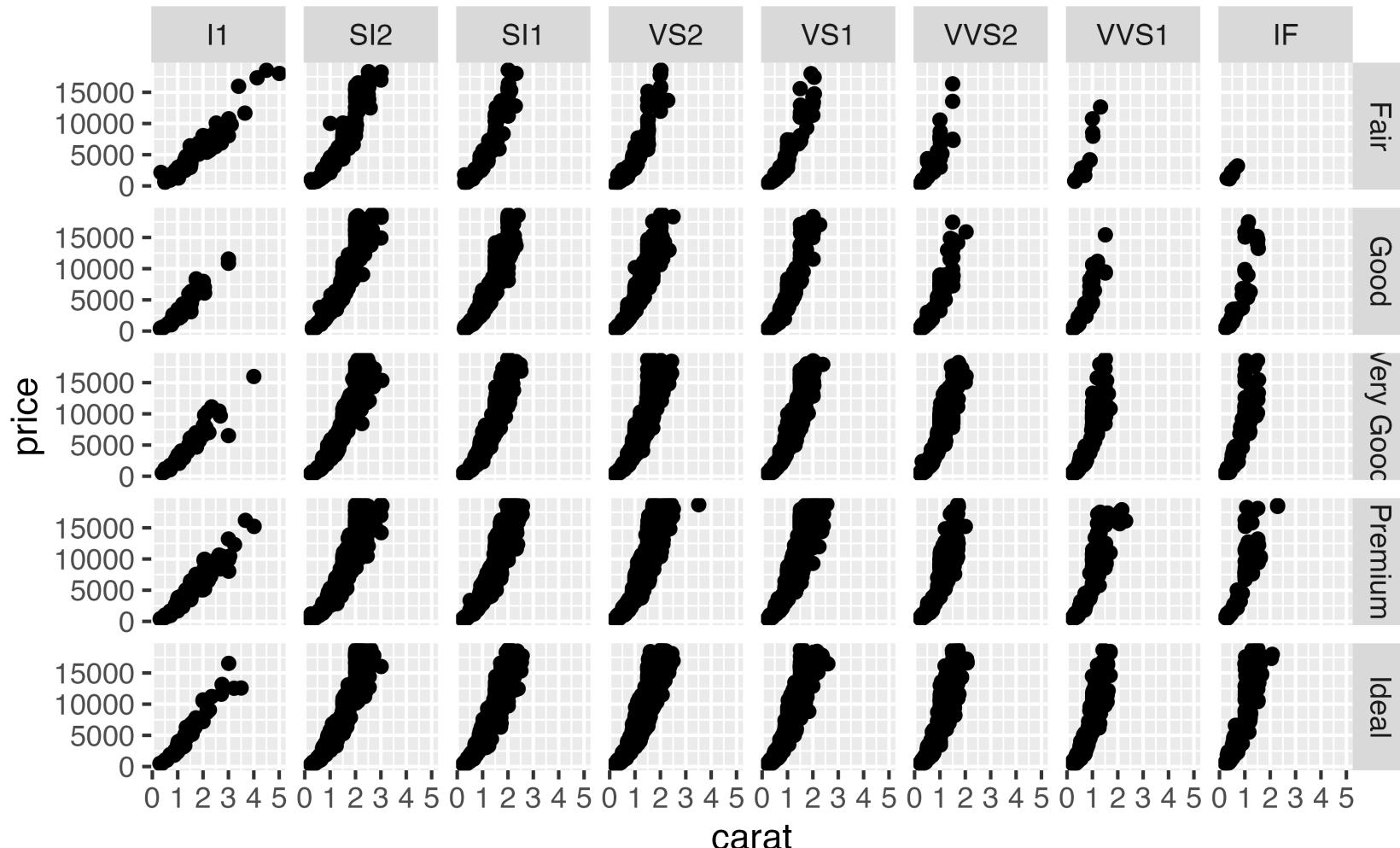
`facet_grid()`

`facet_wrap()`

`facet_grid(rows = vars(x), cols = vars(y))`

`facet_wrap(vars(x))`

```
1 diamonds |>
2   ggplot(aes(x = carat, y = price)) +
3   geom_point() +
4   facet_grid(rows = vars(cut), cols = vars(clarity))
```

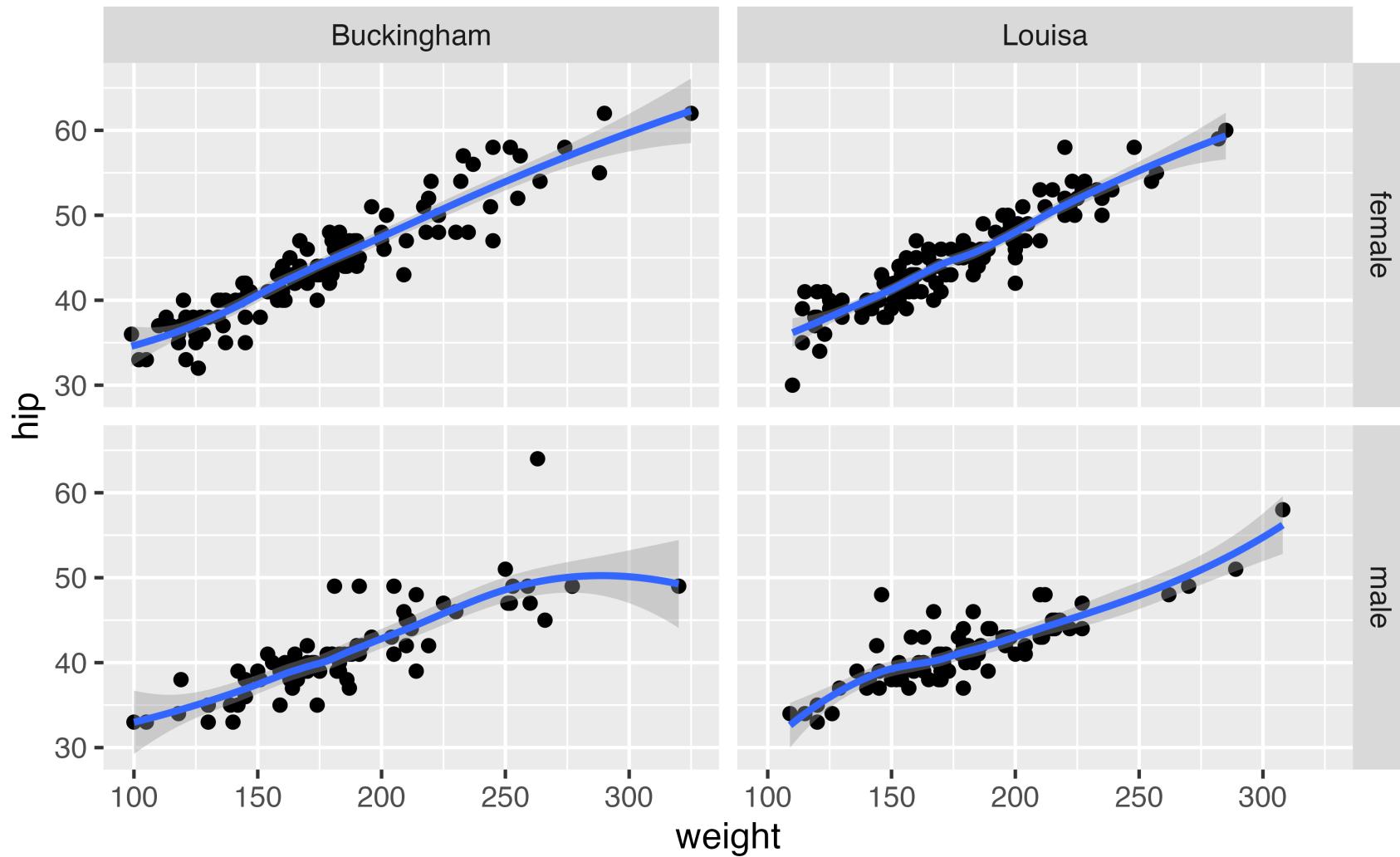


Your Turn 8

Use a facet grid by **gender** and **location**

```
1 ggplot(diabetes, aes(weight, hip)) +  
2   geom_point() +  
3   geom_smooth()
```

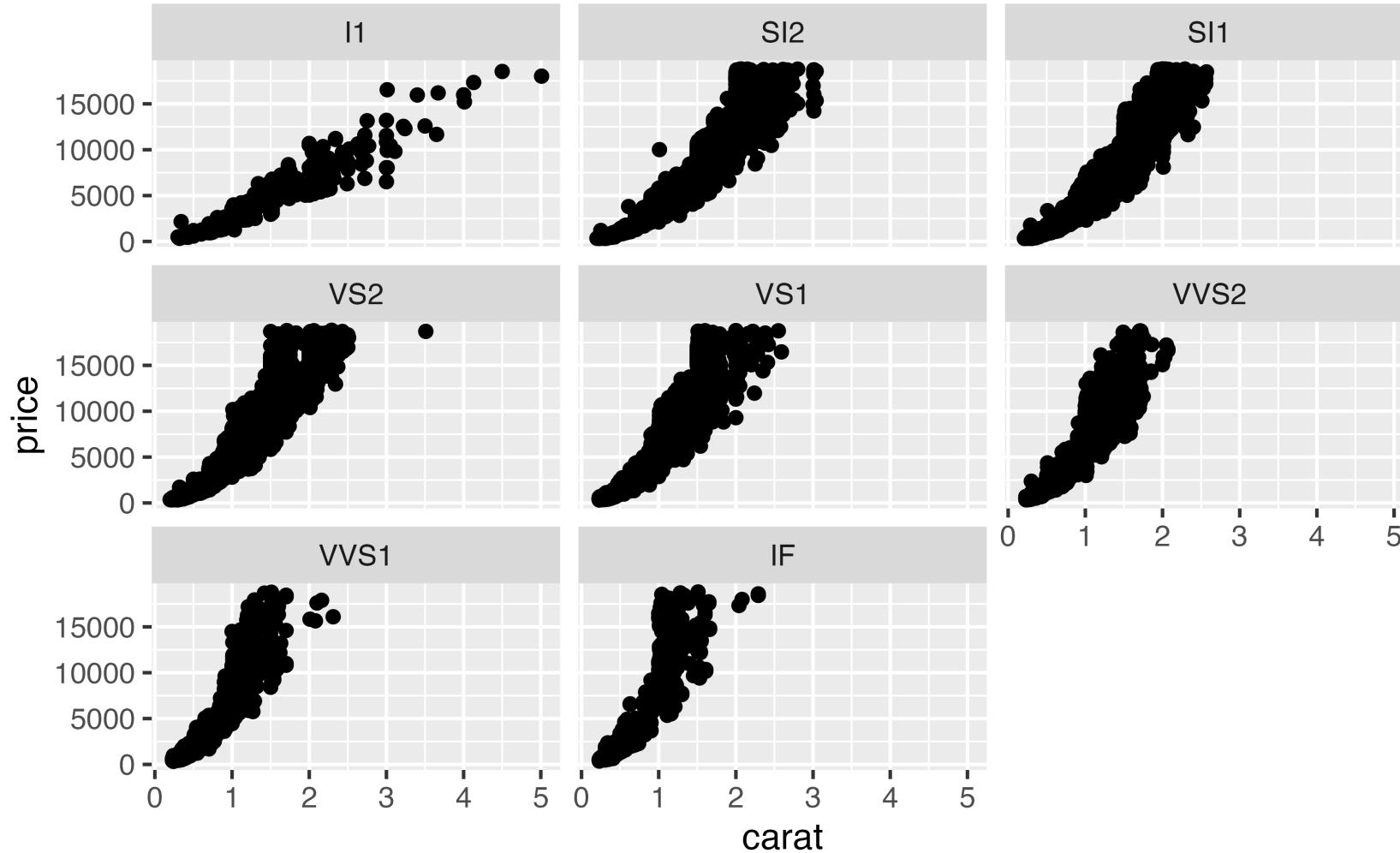
```
1 ggplot(diabetes, aes(weight, hip)) +  
2   geom_point() +  
3   geom_smooth() +  
4   facet_grid(rows = vars(gender), cols = vars(location))
```



facet_wrap()

```
1 diamonds |>
2   ggplot(aes(x = carat, y = price)) +
3   geom_point() +
4   facet_wrap(vars(clarity))
```

facet_wrap()



datasauRus

```
1 library(datasauRus)
2 datasaurus_dozen
```

```
# A tibble: 1,846 × 3
  dataset      x      y
  <chr>    <dbl>  <dbl>
1 dino      55.4   97.2
2 dino      51.5   96.0
3 dino      46.2   94.5
4 dino      42.8   91.4
5 dino      40.8   88.3
6 dino      38.7   84.9
7 dino      35.6   79.9
8 dino      33.1   77.6
9 dino      29.0   74.5
10 dino     26.2   71.4
# ... with 1,836 more rows
```



new data alert!



datasaurus_dozen

#	dataset	x	y
1	dino	55.38460	97.179500
2	dino	51.53850	96.025600
3	dino	46.15380	94.487200
4	dino	42.82050	91.410300
5	dino	40.76920	88.333300
6	dino	38.71790	84.871800
7	dino	35.64100	79.871800
8	dino	33.07690	77.564100
9	dino	28.97440	74.487200
10	dino	26.15380	71.410300
11	dino	23.07690	66.410300
12	dino	22.30770	61.794900
13	dino	22.30770	57.179500
14	dino	23.33330	52.948700
15	dino	25.89740	51.025600
16	dino	29.48720	51.025600
17	dino	32.82050	51.025600
18	dino	35.38460	51.410300

Where does it come from?
The datasauRus R package

How can I use it?

```
library(datasauRus)  
View(datasaurus_dozen)
```



it's invisible!

Your Turn 9: Challenge!

1. Load the `datasauRus` package. This package includes a data set called `datasaurus_dozen`.
2. Use `dplyr` to summarize the correlation between `x` and `y`. First, group it by `dataset`, and then summarize with the `cor()` function. Call the new variable `corr`. What's it look like?
3. Mutate `corr`. Round it to 2 digits. Then, mutate it again (or wrap it around your first change) using: `paste("corr:", corr)`
4. Save the summary data frame as `corrs`.
5. Pass `datasaurus_dozen` to `ggplot()` and add a point geom
6. Use a facet (wrap) for `dataset`.
7. Add a text geom. For this geom, set `data = corrs`. You also need to use `aes()` in this call to set `label = corr`, `x = 50`, and `y = 110`.

```
1 corrs <- datasaurus_dozen |>
2   group_by(dataset) |>
3   summarize(corr = cor(x, y)) |>
4   mutate(
5     corr = round(corr, 2),
6     corr = paste("corr:", corr)
7   )
```

1 corrs

```
# A tibble: 13 × 2
  dataset      corr
  <chr>       <chr>
1 away        corr: -0.06
2 bullseye    corr: -0.07
3 circle      corr: -0.07
4 dino         corr: -0.06
5 dots         corr: -0.06
6 h_lines      corr: -0.06
7 high_lines   corr: -0.07
8 slant_down   corr: -0.07
9 slant_up     corr: -0.07
10 star        corr: -0.06
11 ... 1: ... 2: ... ^ ^7
```

```
1 datasaurus_dozen |>
2   ggplot(aes(x, y)) +
3   geom_point() +
4   geom_text(data = corrs, aes(label = corr, x = 50, y = 110)) +
5   facet_wrap(vars(dataset))
```

