

Data Visualization in R

customizing ggplot2

2024-07-18

Scales: *position scales*

`scale_x_continuous()`

`scale_y_date()` `scale_x_log10()`

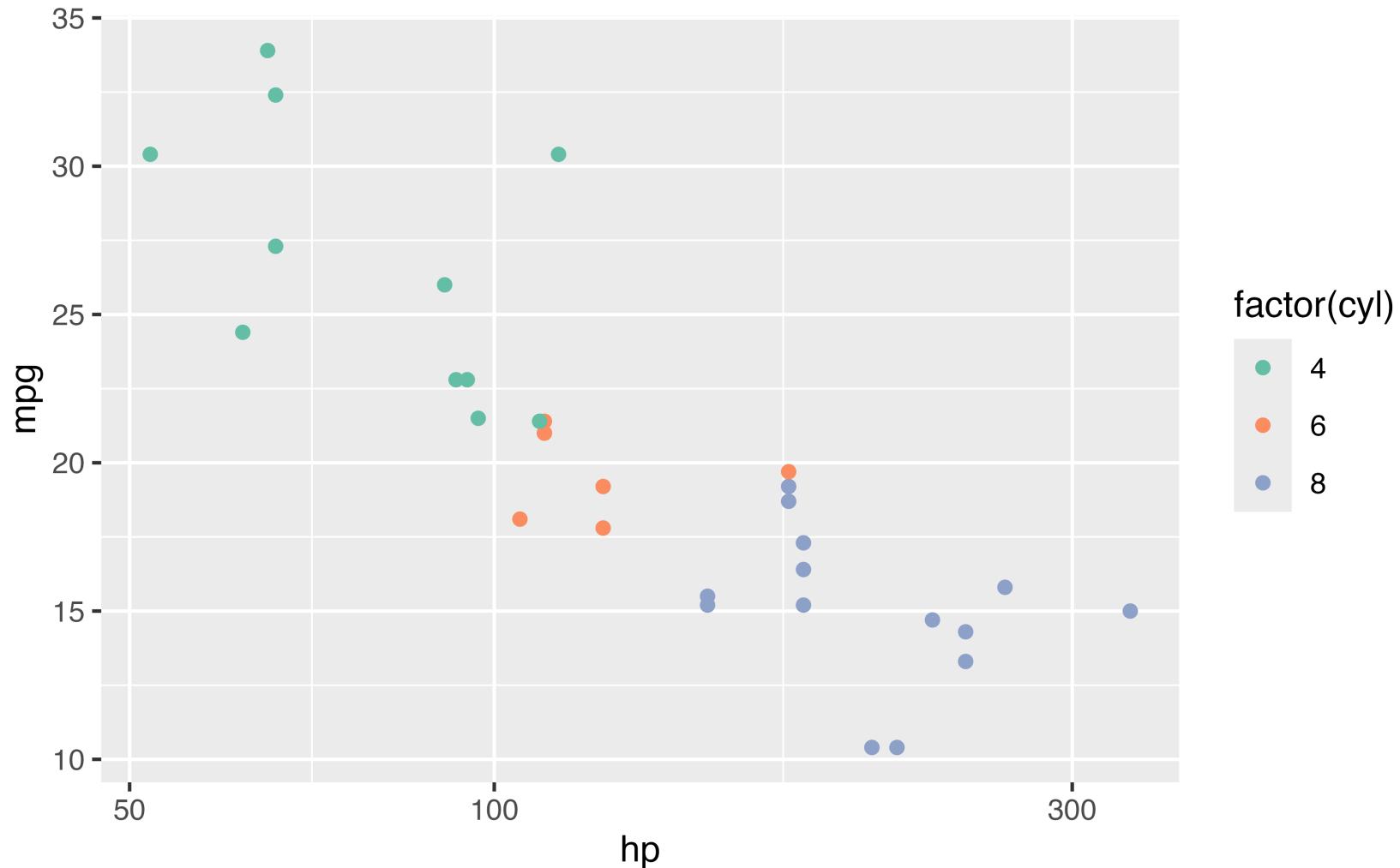
Scales: *aesthetic scales*

`scale_color_hue()`

`scale_fill_brewer()`

`scale_shape_manual()`

```
1 mtcars |>
2   ggplot(aes(hp, mpg, color = factor(cyl))) +
3   geom_point() +
4   scale_x_log10() +
5   scale_color_brewer(palette = "Set2")
```

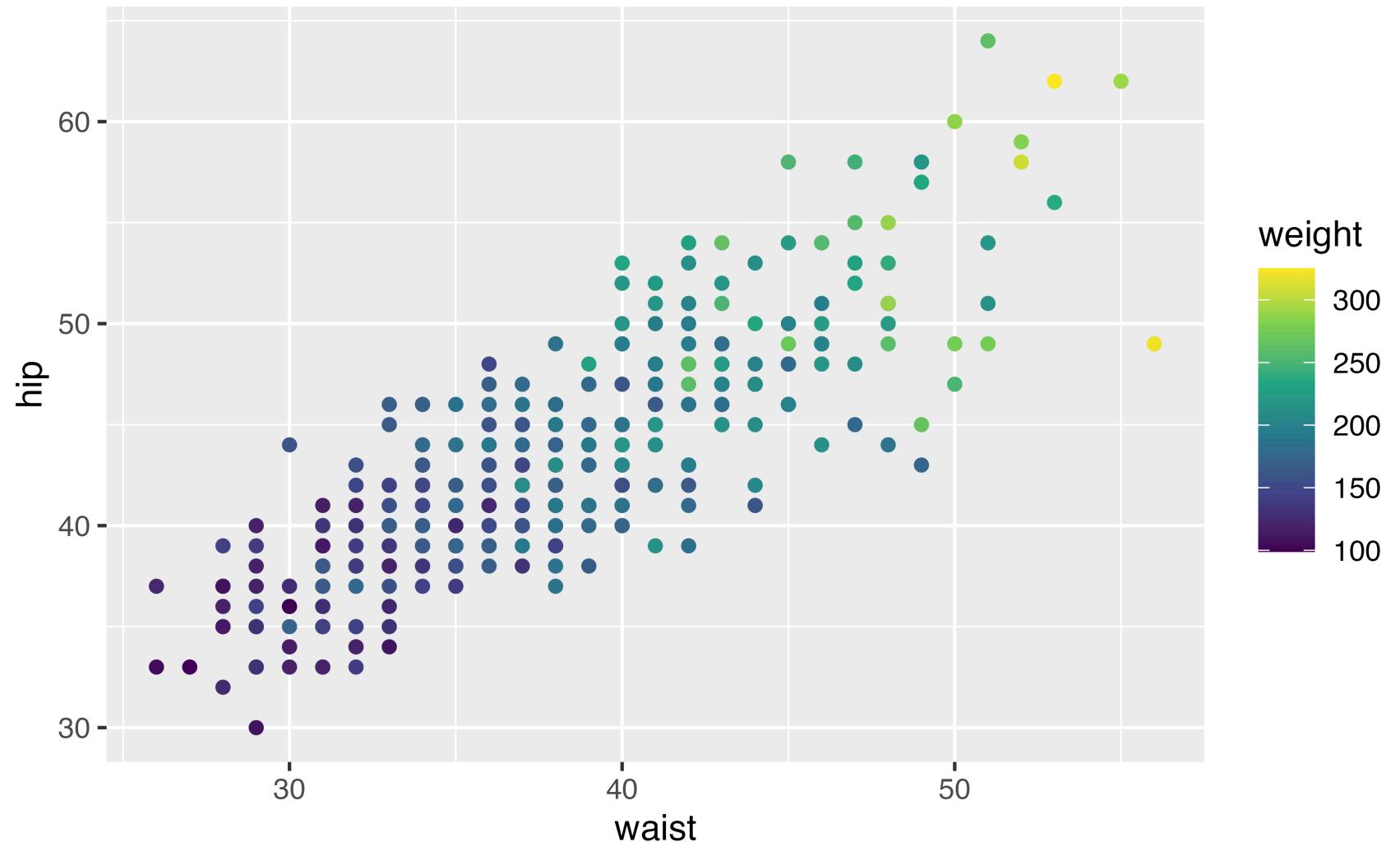


Your Turn 10

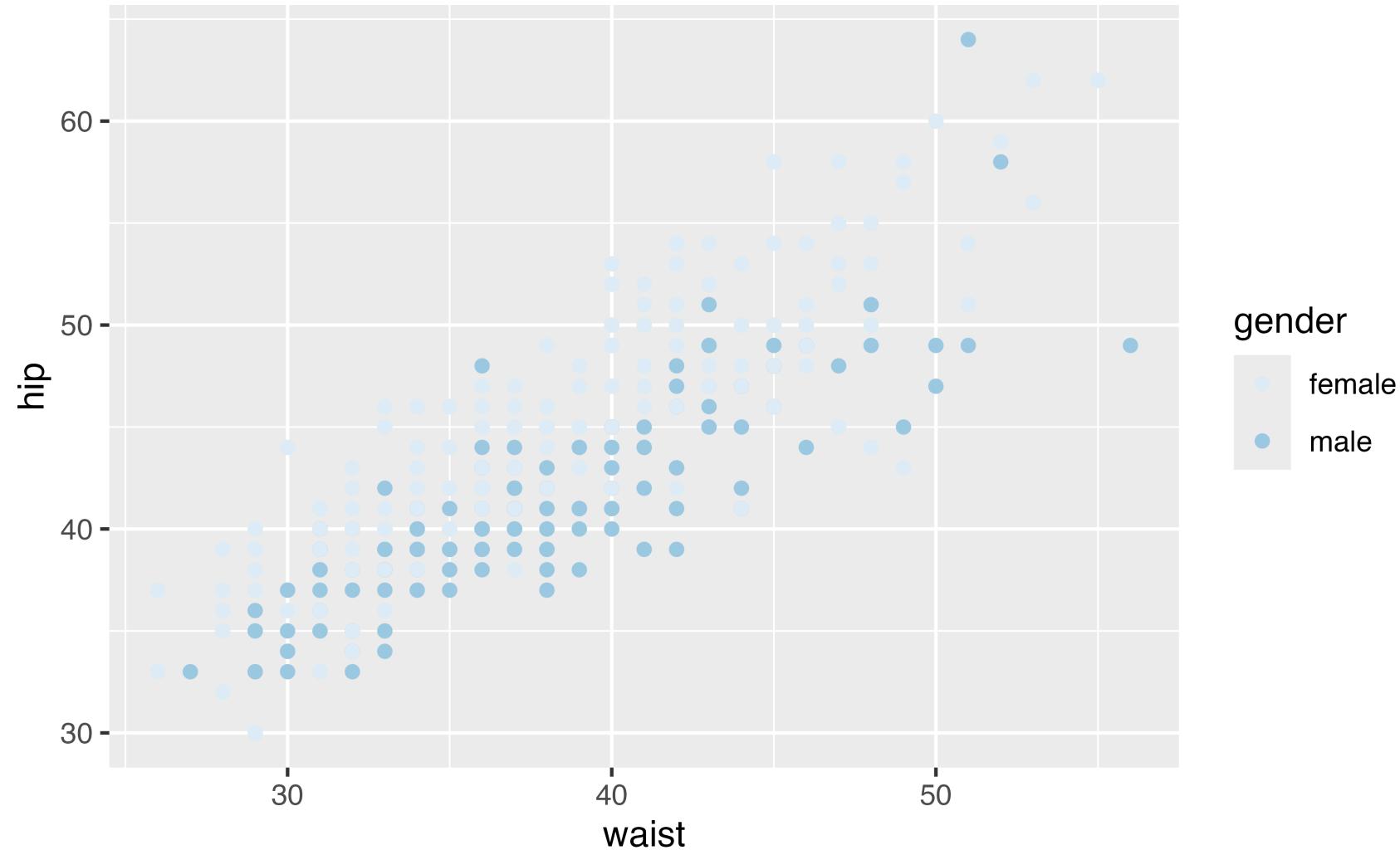
1. Change the color scale by adding a scale layer. Experiment with `scale_color_distiller()` and `scale_color_viridis_c()`. Check the help pages for different palette options.
2. Set the color aesthetic to `gender`. Try `scale_color_brewer()`.
3. Set the colors manually with `scale_color_manual()`. Use `values = c("#E69F00", "#56B4E9")` in the function call.
4. Change the legend title for the color legend. Use the `name` argument in whatever scale function you're using.

```
1 diabetes |>
2   ggplot(aes(waist, hip, color = weight)) +
3     geom_point()
```

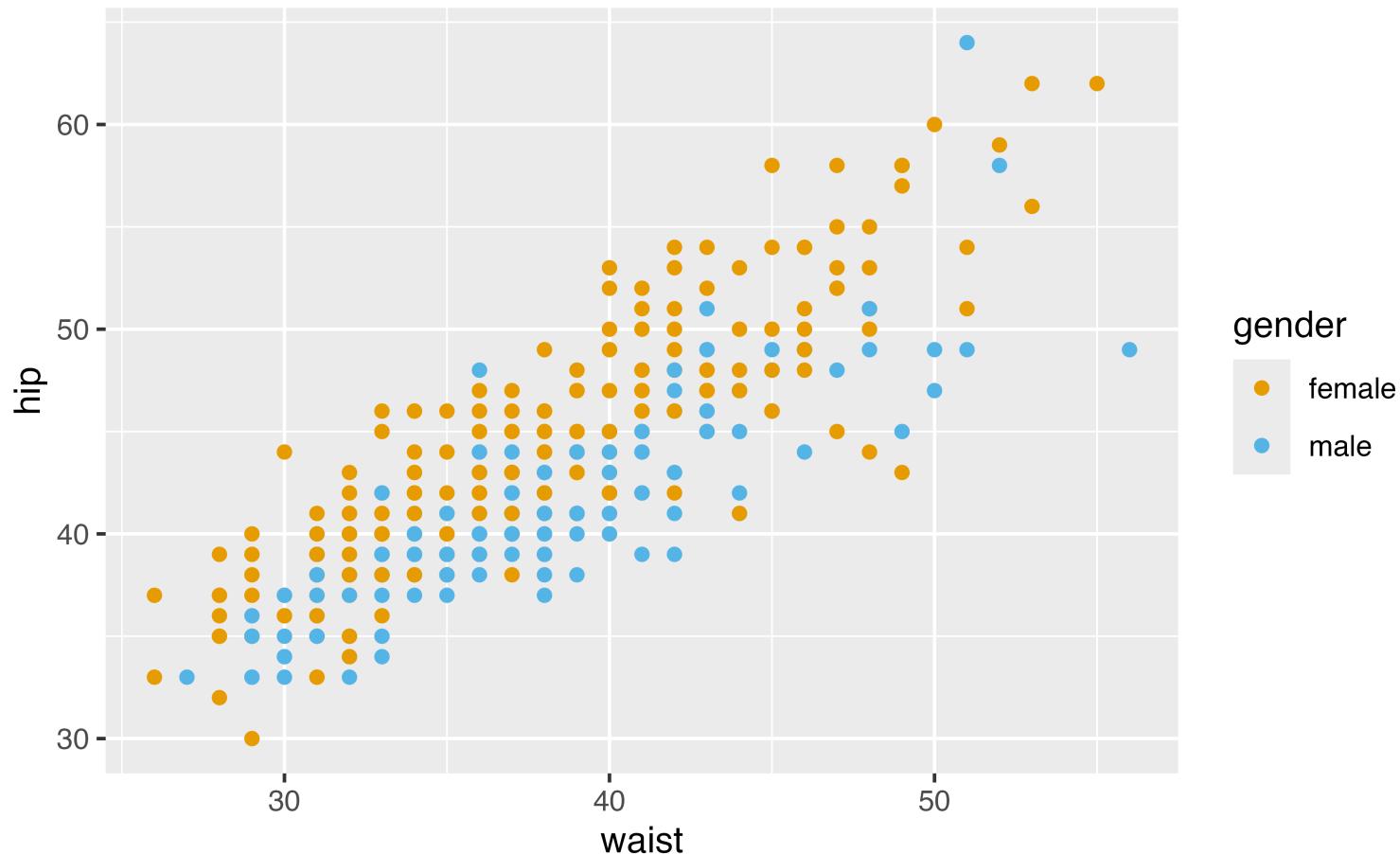
```
1 diabetes |>
2   ggplot(aes(waist, hip, color = weight)) +
3   geom_point() +
4   scale_color_viridis_c()
```



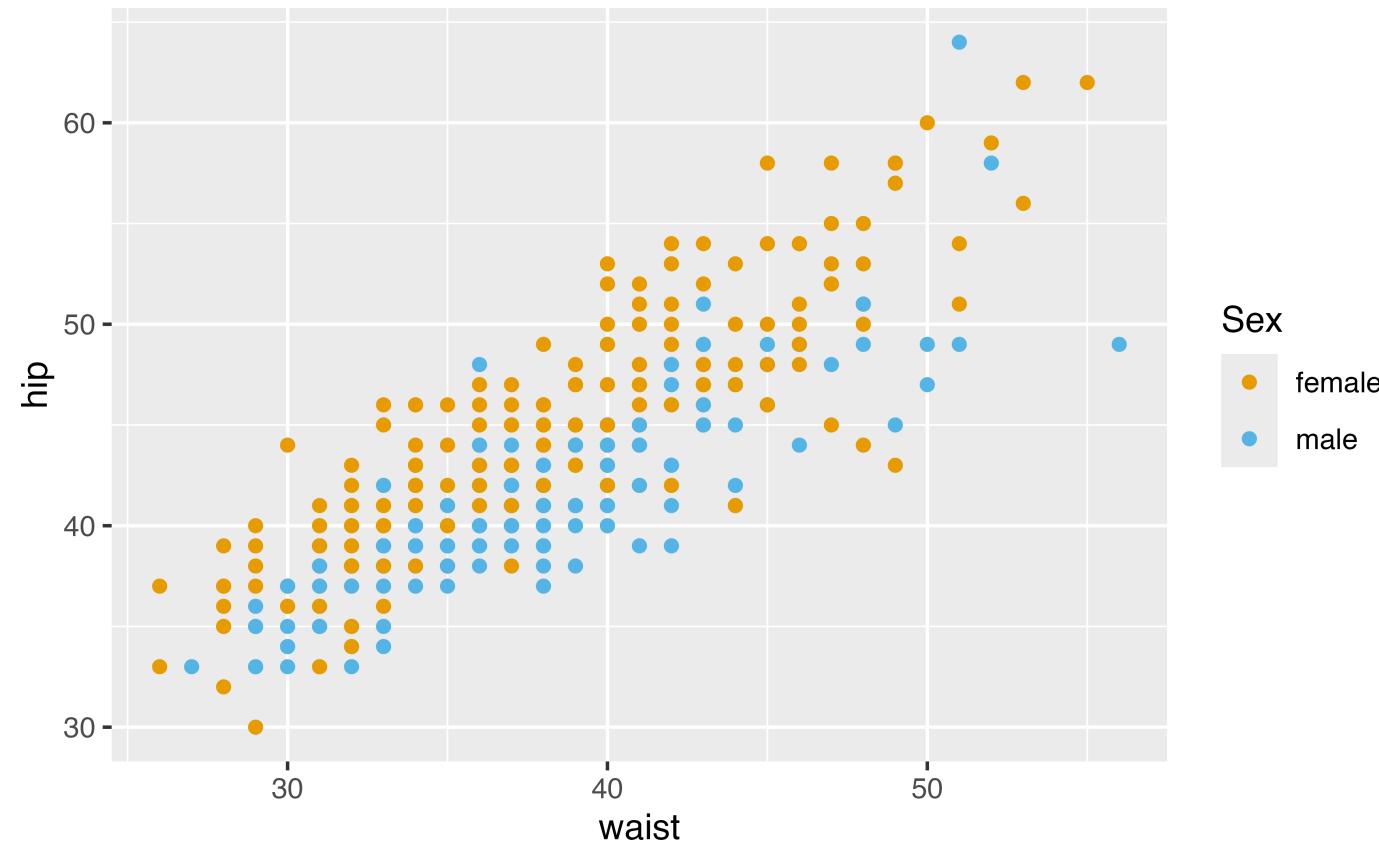
```
1 diabetes |>
2   ggplot(aes(waist, hip, color = gender)) +
3   geom_point() +
4   scale_color_brewer()
```



```
1 diabetes |>
2   ggplot(aes(waist, hip, color = gender)) +
3   geom_point() +
4   scale_color_manual(
5     values = c("#E69F00", "#56B4E9"))
6 )
```



```
1 diabetes |>
2   ggplot(aes(waist, hip, color = gender)) +
3   geom_point() +
4   scale_color_manual(
5     name = "Sex",
6     values = c("#E69F00", "#56B4E9"))
7 )
```



Color Palettes

<https://github.com/EmilHvitfeldt/r-color-palettes>

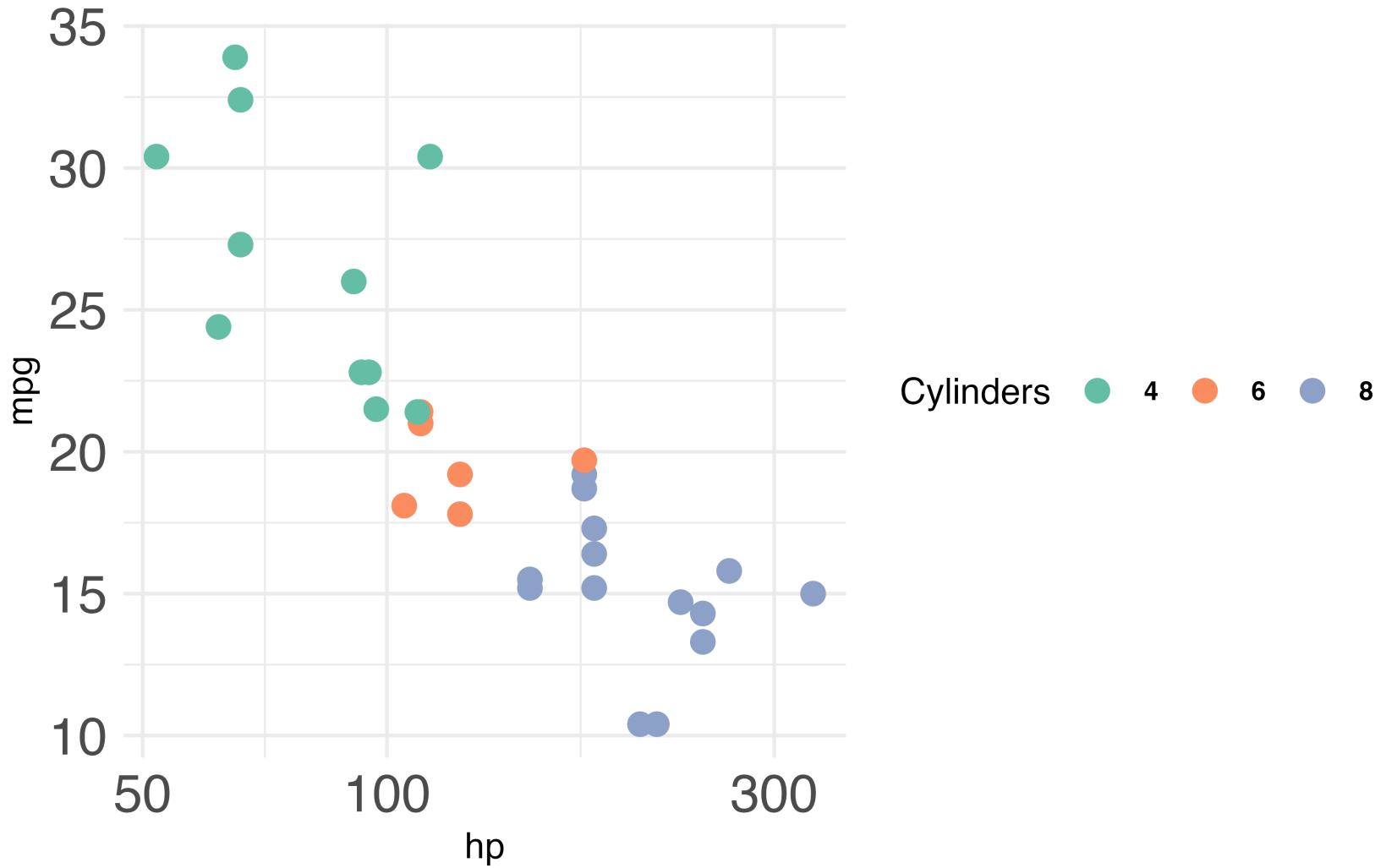
Themes

Non-data ink (text, background, etc)

Pre-specified themes: `theme_gray()` (default),
`theme_minimal()`, `theme_light()`, etc.

`theme()`

```
1 mtcars |>
2   ggplot(aes(hp, mpg, color = factor(cyl))) +
3   geom_point(size = 3) +
4   scale_x_log10() +
5   scale_colour_brewer(name = "Cylinders", palette = "Se-
6   theme_minimal() +
7   theme(
8     axis.text = element_text(size = 16),
9     legend.text = element_text(size = 8, face = "bold")
10    legend.direction = "horizontal"
11  )
```



theme elements

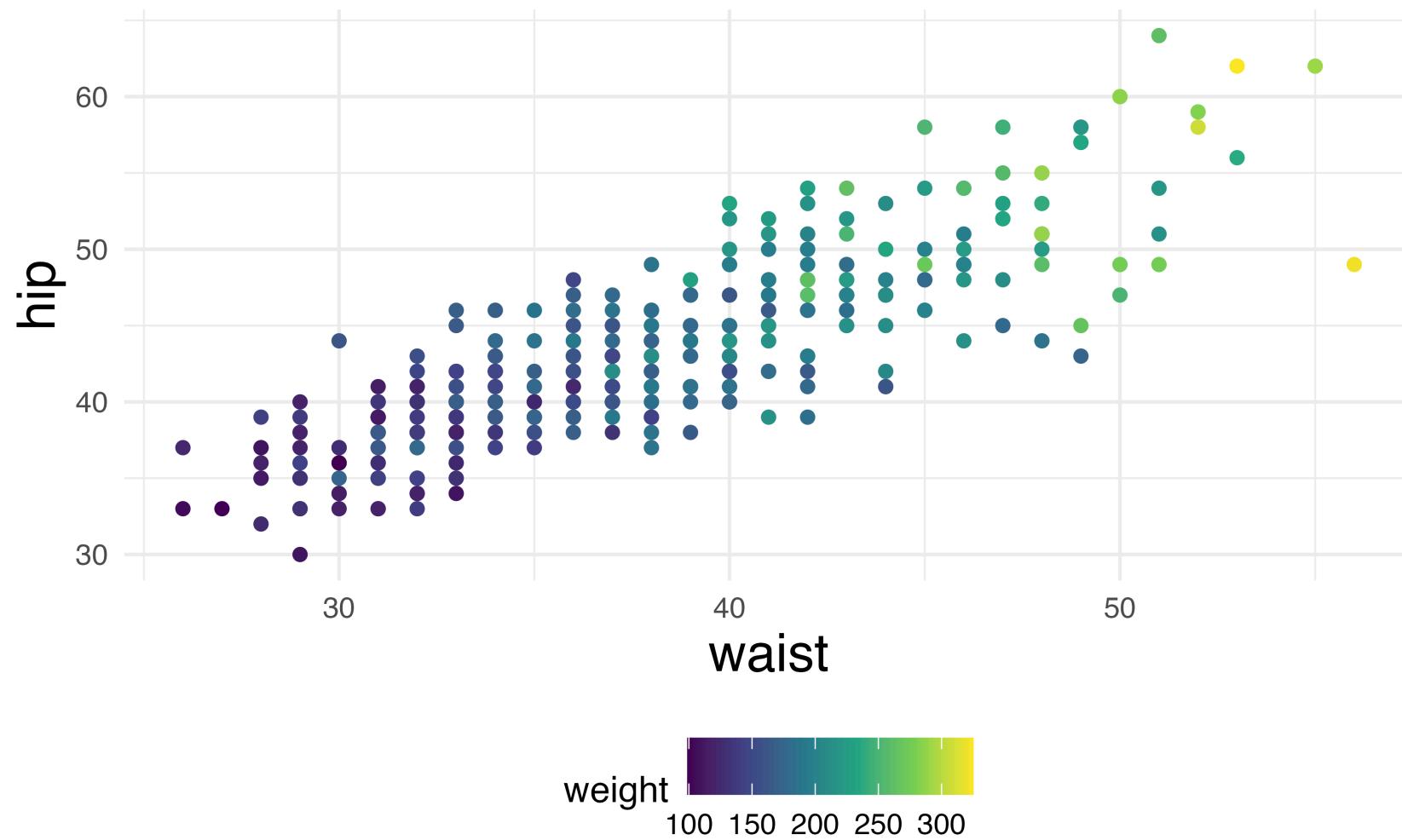
element	draws
<code>element_blank()</code>	nothing (remove element)
<code>element_line()</code>	lines
<code>element_rect()</code>	borders and backgrounds
<code>element_text()</code>	text

Your Turn 11

1. Change the theme using one of the built-in theme functions.
2. Use `theme()` to change the legend to the bottom with `legend.position = "bottom"`.
3. Remove the axis ticks by setting the `axis.ticks` argument to `element_blank()`
4. Change the font size for the axis titles. Use `element_text()`. Check the help page if you don't know what option to change.

```
1 diabetes |>
2   ggplot(aes(waist, hip, color = weight)) +
3   geom_point() +
4   scale_color_viridis_c()
```

```
1 diabetes |>
2   ggplot(aes(waist, hip, color = weight)) +
3   geom_point() +
4   scale_color_viridis_c() +
5   theme_minimal() +
6   theme(
7     legend.position = "bottom",
8     axis.ticks = element_blank(),
9     axis.title = element_text(size = 16)
10    )
```



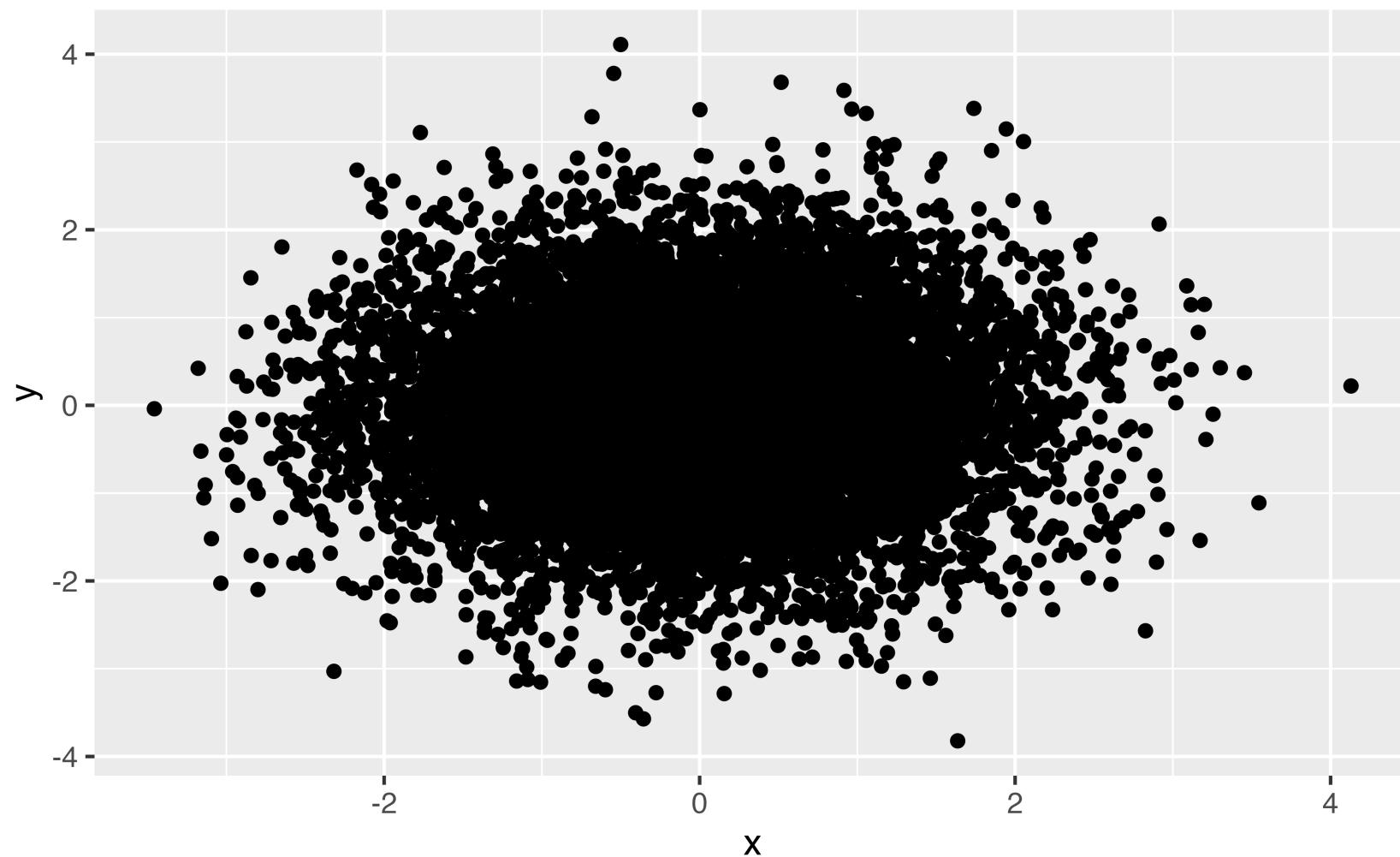
High-density plots

?rnorm, ?Distributions

```
1 bigger_data <- tibble(x = rnorm(10000), y = rnorm(10000))
```

High-density plots

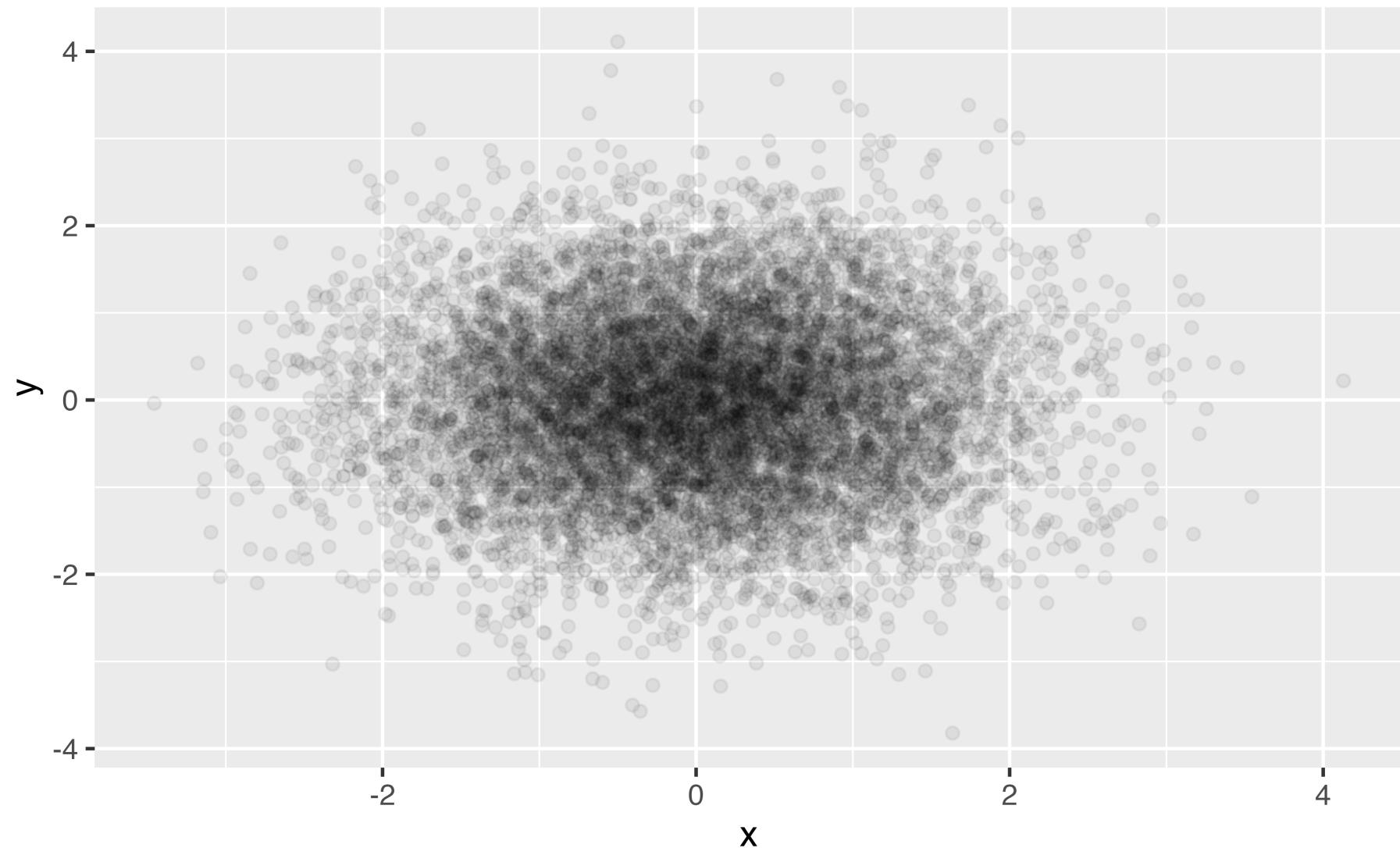
```
1 bigger_data |>  
2 ggplot(aes(x, y)) +  
3 geom_point()
```



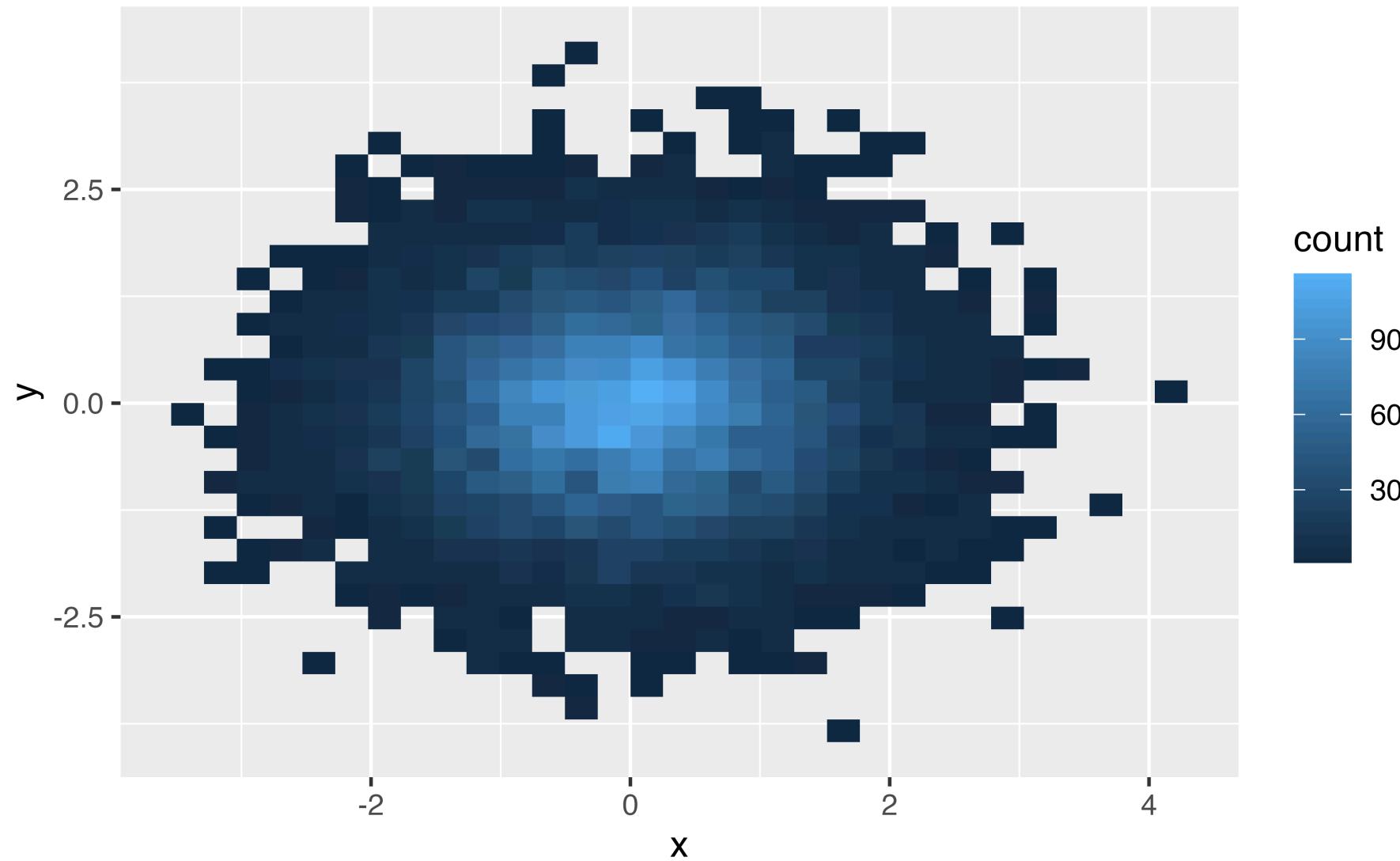
High-density plots

- 1 Transparency
- 2 Binning
- 3 Visualize other summary data

```
1 bigger_data |>
2   ggplot(aes(x, y)) +
3   geom_point(alpha = .05)
```



```
1 bigger_data |>
2   ggplot(aes(x, y)) +
3   geom_bin2d()
```



```
1 bigger_data |>
2   ggplot(aes(x, y)) +
3     geom_hex()
```

y

x

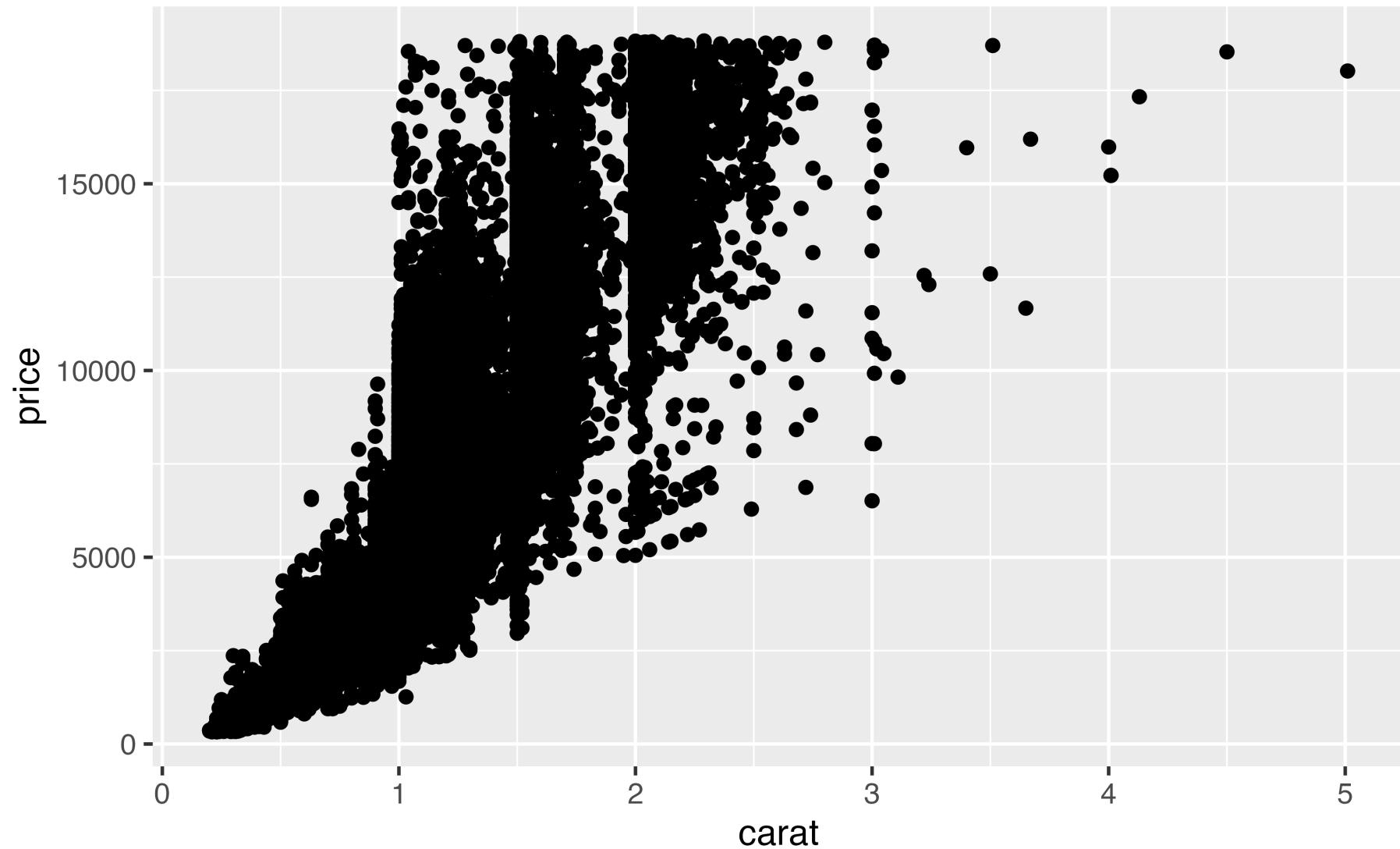
Your Turn 12

Take a look at the **diamonds** data set from **ggplot2**. How many rows does it have?

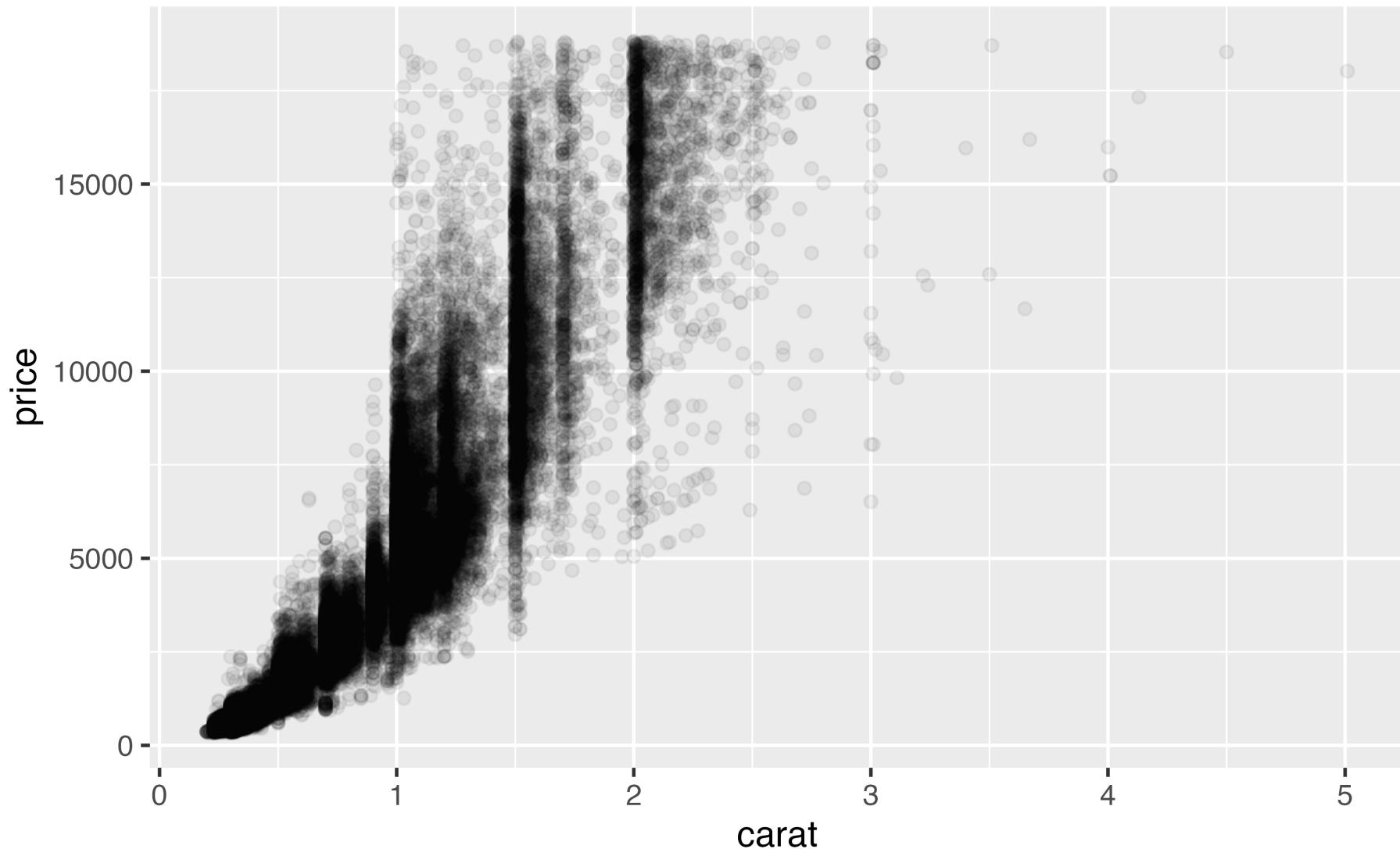
```
1 diamonds
```

1. Make a scatterplot of **carat** vs. **price**. How's it look?
2. Try adjusting the transparency.
3. Replace **geom_point()** with 2d bins.
4. Try hex bins.

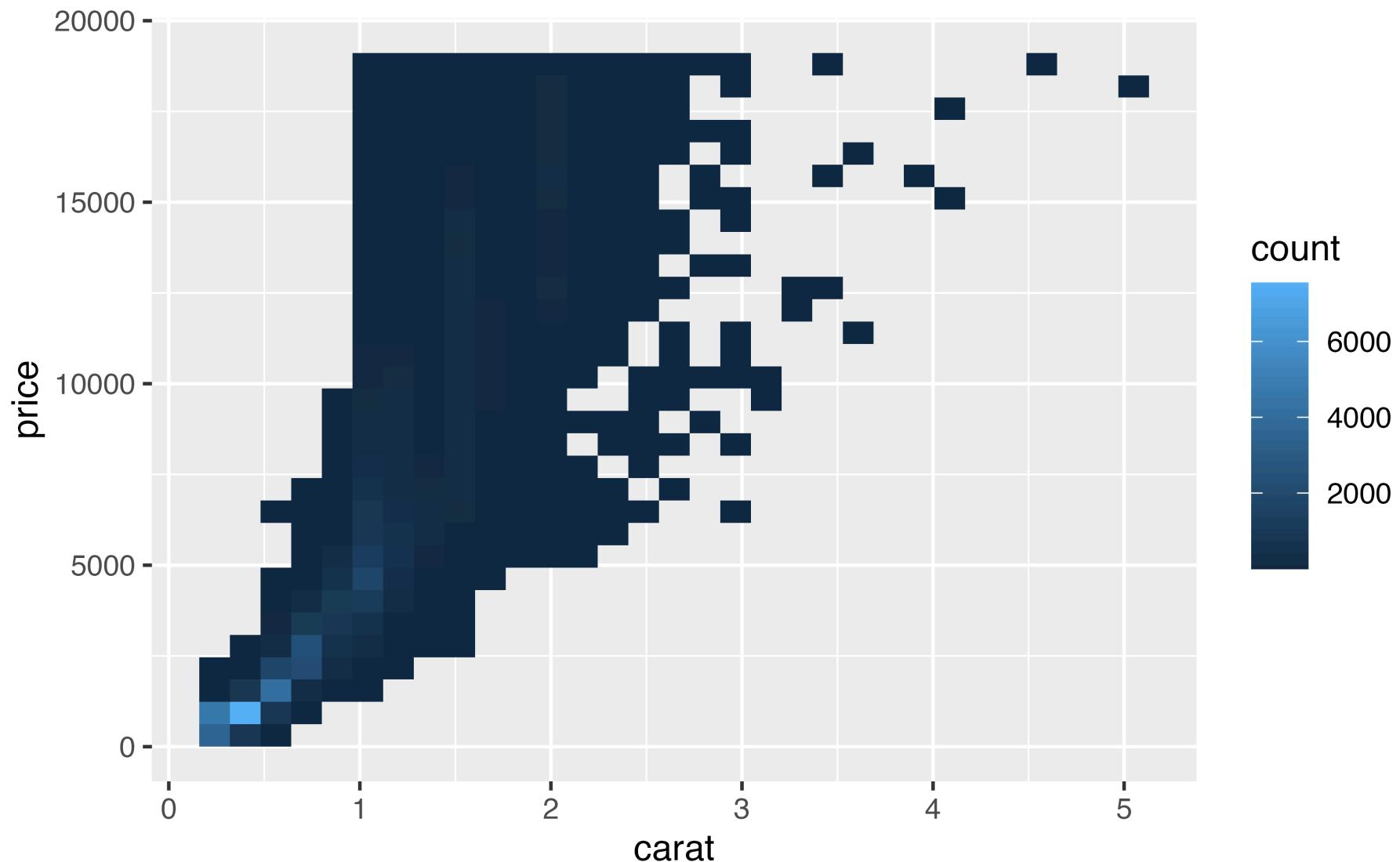
```
1 diamonds |>
2   ggplot(aes(x = carat, price)) +
3     geom_point()
```



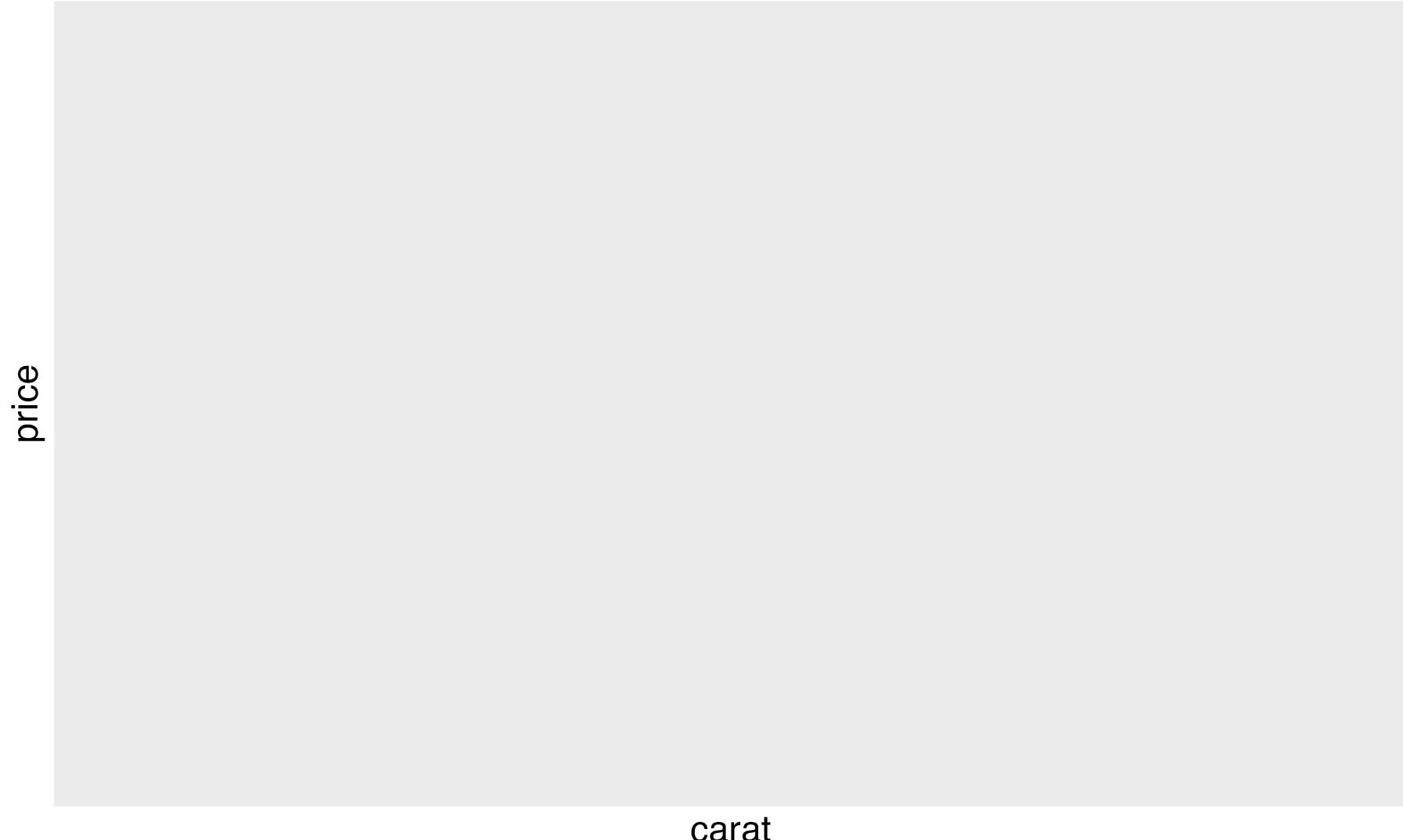
```
1 diamonds |>
2   ggplot(aes(x = carat, price)) +
3     geom_point(alpha = .05)
```



```
1 diamonds |>
2   ggplot(aes(x = carat, price)) +
3     geom_bin2d()
```



```
1 diamonds |>  
2   ggplot(aes(x = carat, price)) +  
3     geom_hex()
```



Labels, titles, and legends

Labels, titles, and legends

Add a title:

`ggttitle()`

`labs(title = "My Awesome Plot")`

Labels, titles, and legends

Change a label:

`xlab()`, `ylab()`

`labs(x = "X Label", y = "Y Label")`

Labels, titles, and legends

Change a legend:

scale_*() functions

```
labs(color = "Wow, labs does  
everything", fill = "Yup")
```

Labels, titles, and legends

Remove the legend:

```
theme(legend.position = "none")
```

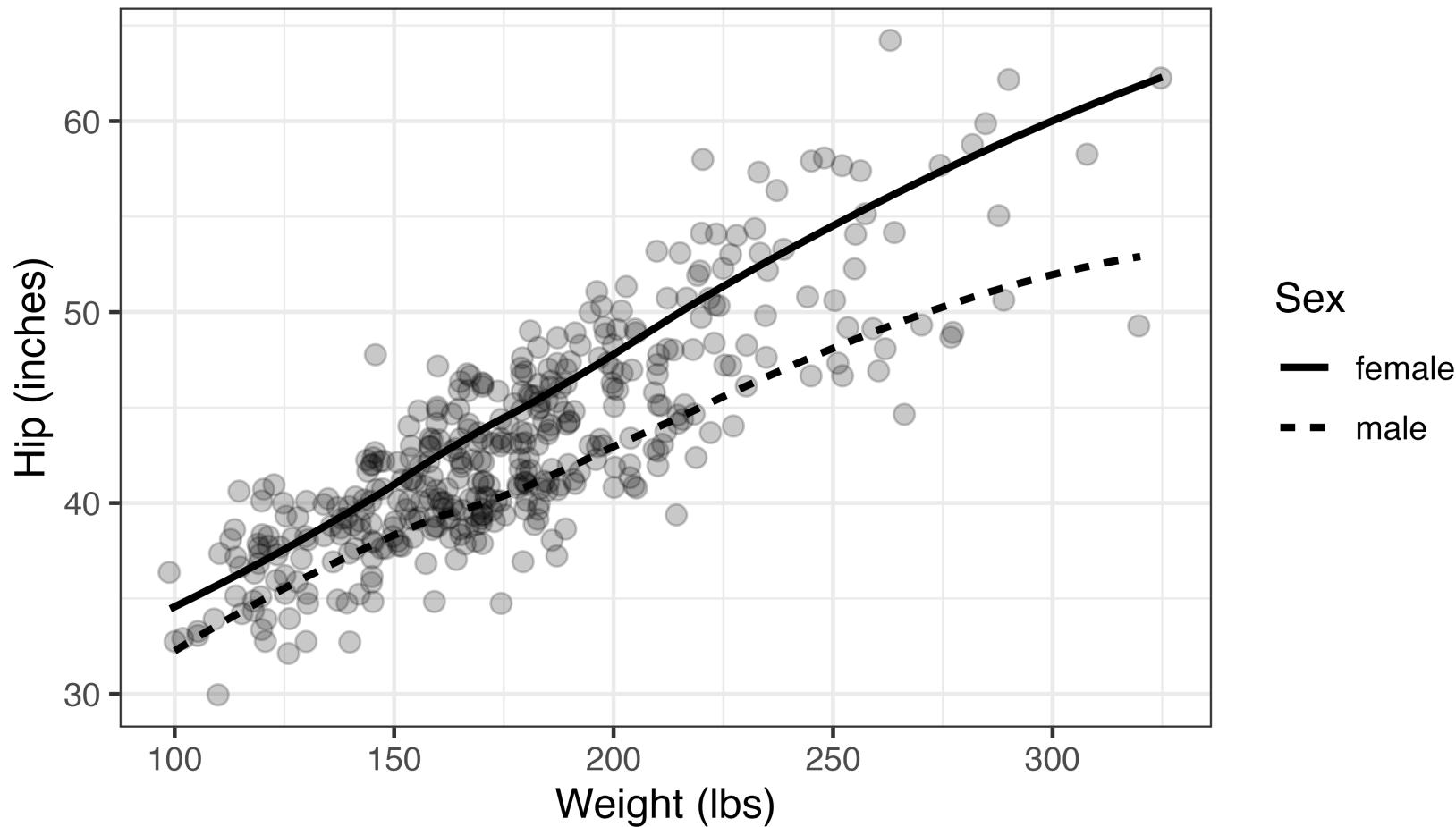
Your Turn 13

1. Add a title.
2. Change the x and y axis labels to include the units (inches for **hip** and pounds for **weight**). You can use either **labs()** or **xlab()** and **ylab()**
3. Add **scale_linetype()** and set the **name** argument to “Sex”.

```
1 ggplot(diabetes, aes(weight, hip, linetype = gender)) +  
2   geom_jitter(alpha = .2, size = 2.5) +  
3   geom_smooth(color = "black", se = FALSE) +  
4   theme_bw(base_size = 12)
```

```
1 ggplot(diabetes, aes(weight, hip, linetype = gender)) +
  2   geom_jitter(alpha = .2, size = 2.5) +
  3   geom_smooth(color = "black", se = FALSE) +
  4   theme_bw(base_size = 12) +
  5   labs(x = "Weight (lbs)", y = "Hip (inches)") +
  6   ggtitle("Hip and Weight by Sex") +
  7   scale_linetype(name = "Sex")
```

Hip and Weight by Sex



```
1 ggplot(diabetes, aes(weight, hip, linetype = gender)) +
2   geom_jitter(alpha = .2, size = 2.5) +
3   geom_smooth(color = "black", se = FALSE) +
4   theme_bw(base_size = 12) +
5   labs(
6     title = "Hip and Weight by Sex",
7     x = "Weight (lbs)",
8     y = "Hip (inches)",
9     linetype = "Sex"
10    )
```

Saving plots

`ggsave()`

```
1 ggsave(  
2   filename = "figure_name.png",  
3   dpi = 320  
4 )
```

Saving plots

`ggsave()`

```
1 ggsave(  
2   filename = "figure_name.png",  
3   plot = plot_you_just_made,  
4   dpi = 320  
5 )
```

Your Turn 14

Save the last plot and then locate it in the files pane.

Your Turn 14

Save the last plot and then locate it in the files pane.

```
1 ggsave("diabetes_weight_hip.png", dpi = 320)
```

Take aways:

You can use this code template to make thousands of graphs with ggplot2.

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



HELEN
GREEN

Resources

R for Data Science: A comprehensive but friendly introduction to the tidyverse. Free online.

Posit Recipes: Common code patterns in R (with some comparisons to SAS)

Data Visualization: A Practical Introduction: Mostly free online; great ggplot2 intro

