Introduction to Data Visualization with ggplot2

OCRUG Hackathon – April 10, 2021

Goals for this module

- Learn about the grammar of graphics and how to become a data visualization "chef" (instead of a data viz customer)

- Learn about the ggplot2 R package

- Learn essential ggplot2 syntax and create basic data visualizations

A few notes

- ggplot2 is a deep package – it will take some time to learn and master all that's there
- This intro tutorial just scratches the surface, not complete (at all…)
- One of the hardest things about ggplot2 is the syntax
  - even a simple plots require more code vs. base R
  - plot code can have lots of pieces to keep track of
  - lots of parentheses
  - when to use quotes and when not to
- ggplot2 is not the easiest plotting library to learn, but its power and flexibility are worth the learning curve

Ggplot2 and the Grammar of Graphics

- ggplot2 is one of the most popular R packages
- Based on the ideas of L. Wilkinson and collaborators, forms a grammar for construction data visualizations
- ggplot2 allows for the creation of highly flexible and customized plots using a "layer-by-layer" approach
- See H. Wickham's paper and references for more info http://vita.had.co.nz/papers/layered-grammar.pdf

ggplot2 let's you become a data viz chef!

Many data vis tools/packages give you a menu of plots to choose from

You're a customer
You only get what you can order

ggplot2 gives you the ingredients to make any kind of plot you want

You're a chef
You can make anything you want

High-level view of making a ggplot2 plot

1. Start with a data set

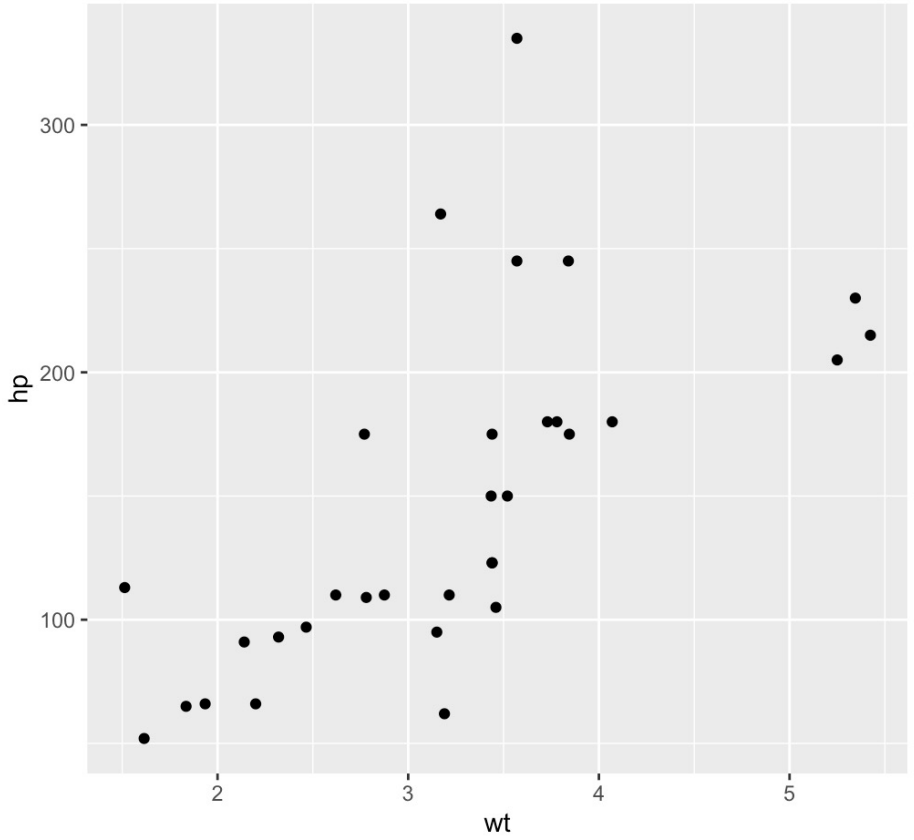
	mpg	cyl	disp	hp	drat	wt	qsec	vs	am	gear	carb
Mazda RX4	21.0	6	160.0	110	3.90	2.620	16.46	0	1	4	4
Mazda RX4 Wag	21.0	6	160.0	110	3.90	2.875	17.02	0	1	4	4
Datsun 710	22.8	4	108.0	93	3.85	2.320	18.61	1	1	4	1
Hornet 4 Drive	21.4	6	258.0	110	3.08	3.215	19.44	1	0	3	1
Hornet Sportabout	18.7	8	360.0	175	3.15	3.440	17.02	0	0	3	2

2. Write ggplot2 code

```
# loading tidyverse will also load ggplot2
library(tidyverse)

# Make a simple plot
ggplot(mtcars) +
  geom_point(mapping = aes(x = wt, y = hp))
```

3. Out comes a ggplot2 plot



Important ggplot2 concepts

- Goal: map data to visual representations/properties to make a plot
- **geom**s: geometric (think shapes) representations for your data
  - points
  - lines
  - bars
- aesthetics, **aes**: visual properties of the objects in your plot
  - position (x, y)
  - size
  - shape
  - Color
- **mapping**: a link between variables in your data and their aesthetic properties

The (basic) ggplot2 plotting template

The variable name of the data table you want to create a plot with

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

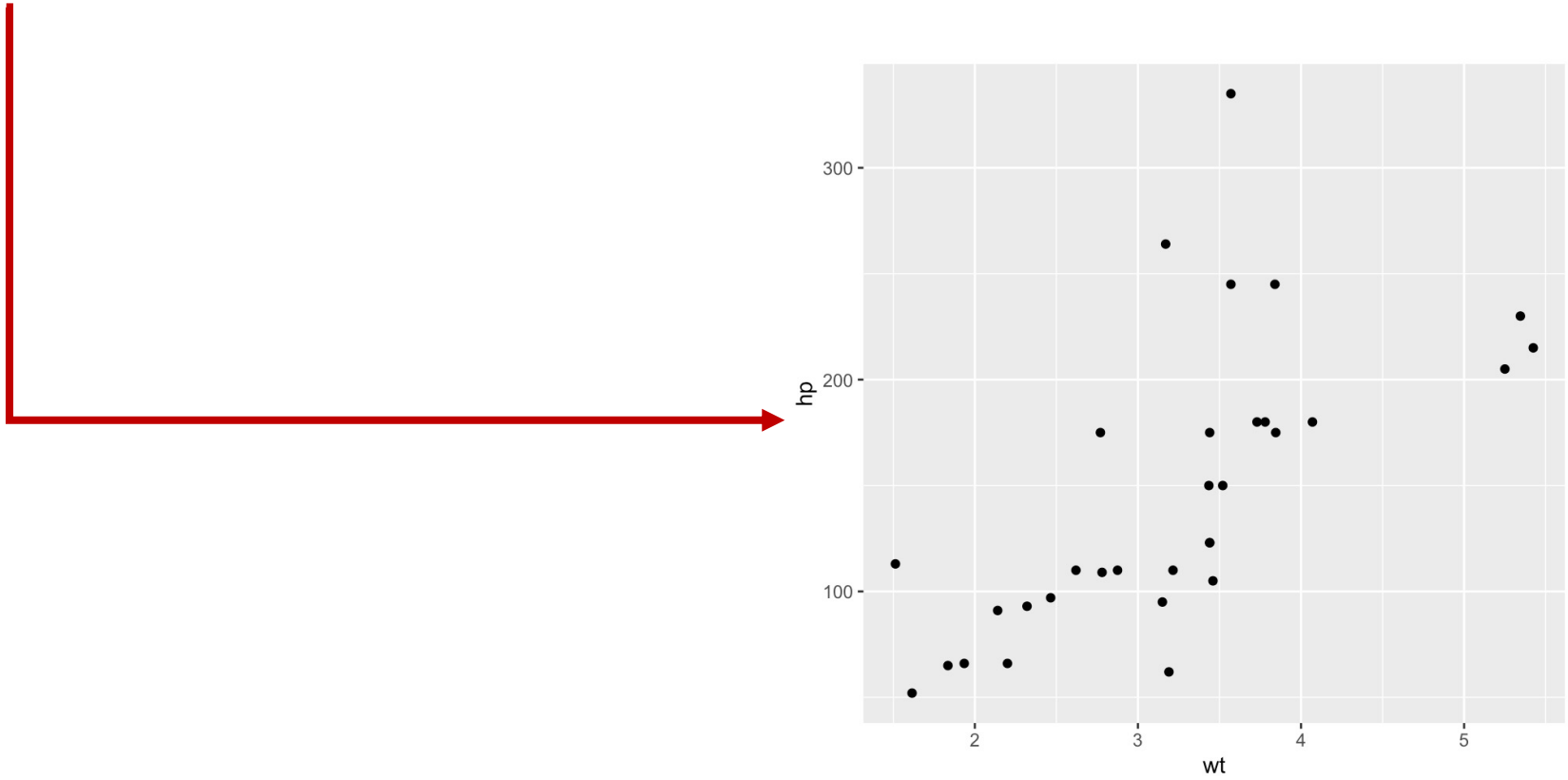
A ggplot2 function that defines the type of geometric representation you want to use, e.g.

```
geom_point
geom_line
geom_histogram
geom_boxplot
```

expression(s) that link columns in your <DATA> to aesthetic properties

Let's revisit the first plotting example

```
ggplot(data = mtcars) +
  geom_point(mapping = aes(x = wt, y = hp))
```



What's going on here?

1. Initialize the plot with the data set you want to plot

```
ggplot(data = mtcars) +
  geom_point(mapping = aes(x = wt, y = hp))
```

2. Use the + sign to add a layer to your plot

3. Specify a geom function for the type of representation you want to plot

4. Define the mapping of columns in your data to the geom's aesthetic properties

Notes

- `geom_point` says we want to represent our data as points
- (2-D) points need x and y positions, so we need to specify these in our mapping
- `wt` and `hp` are columns in the mtcars data table
- you don't put quotes (" ") around the column names inside `aes`

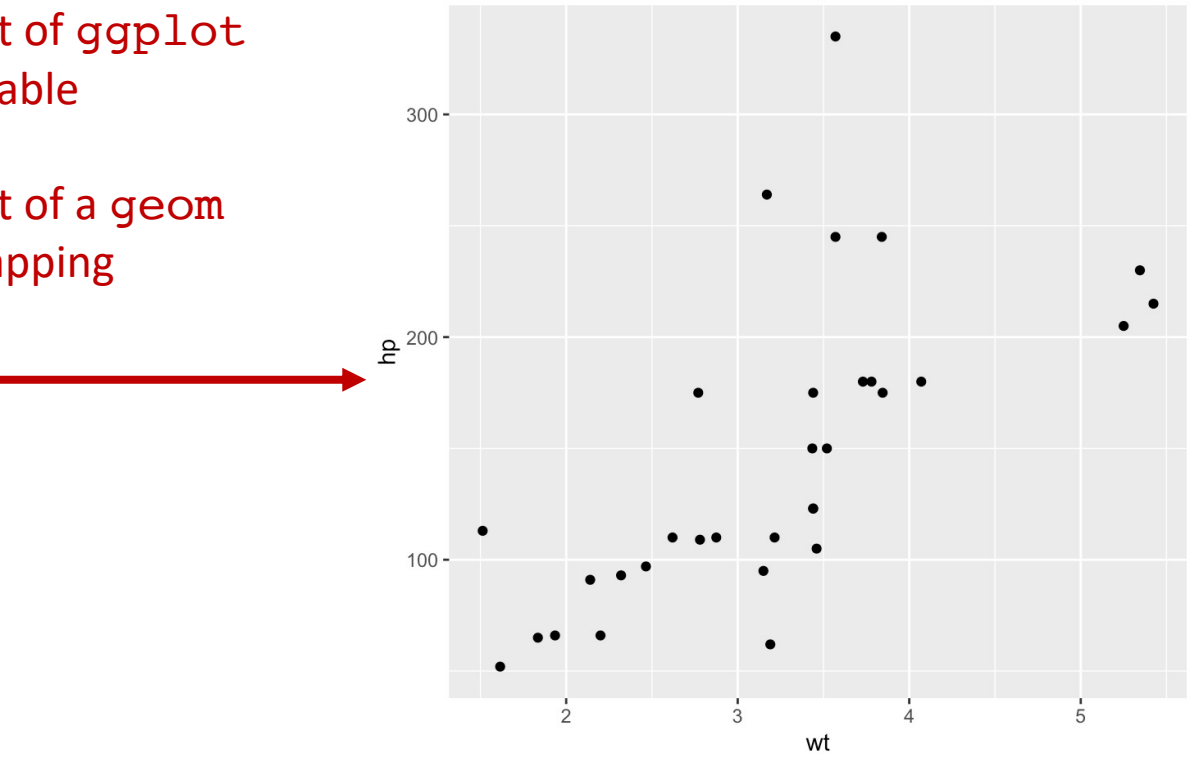
In practice, the code is often simplified further

We removed the function argument names because the argument were specified in the assumed order

```
ggplot(mtcars) +
  geom_point(aes(x = wt, y = hp))
```

The first argument of `ggplot` is the input data table

The first argument of a `geom` function is the mapping



A more complex plot

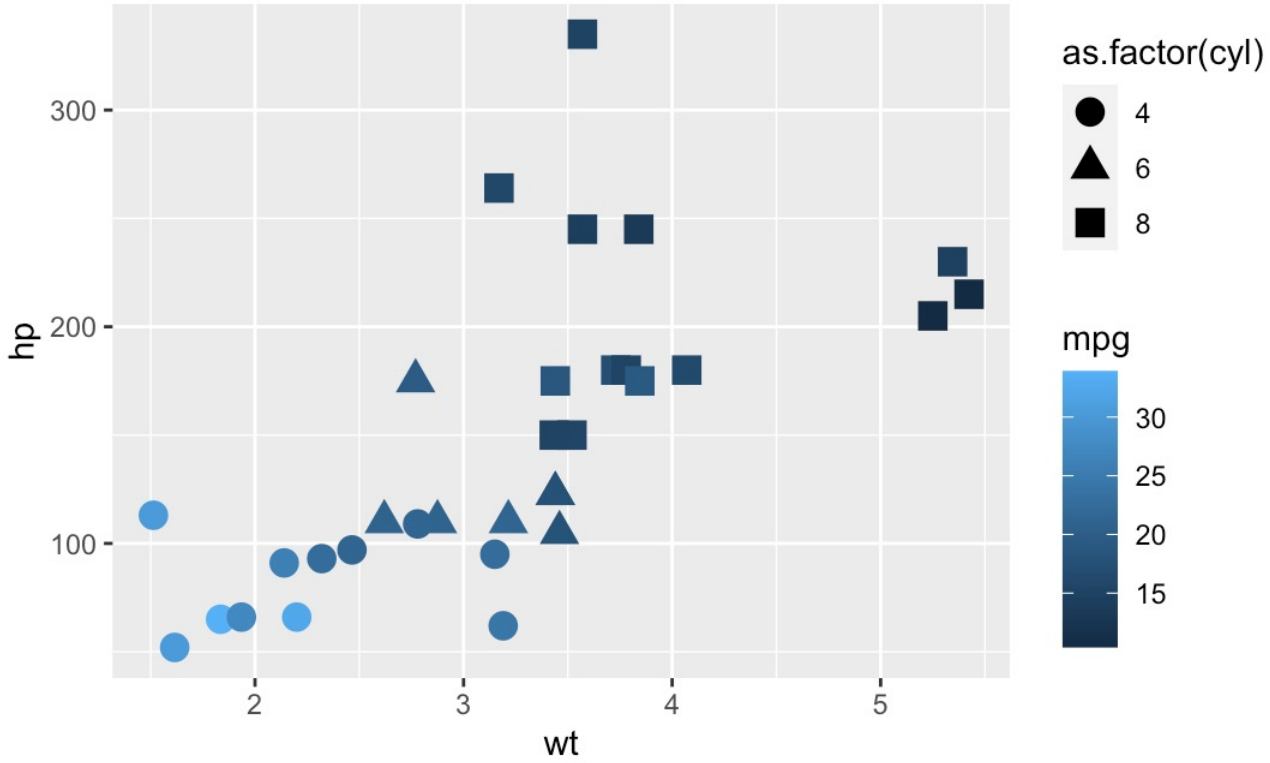
We are also mapping point shape and size to our data.
Note: a shape is a discrete object *not* numerical, so we need to make it a factor

```
ggplot(mtcars) +
  geom_point(aes(x = wt, y = hp, shape = as.factor(cyl), color = mpg),
             size = 4)
```

We also specified the size of the points, but it is a defined single value, NOT mapped to the input data table.

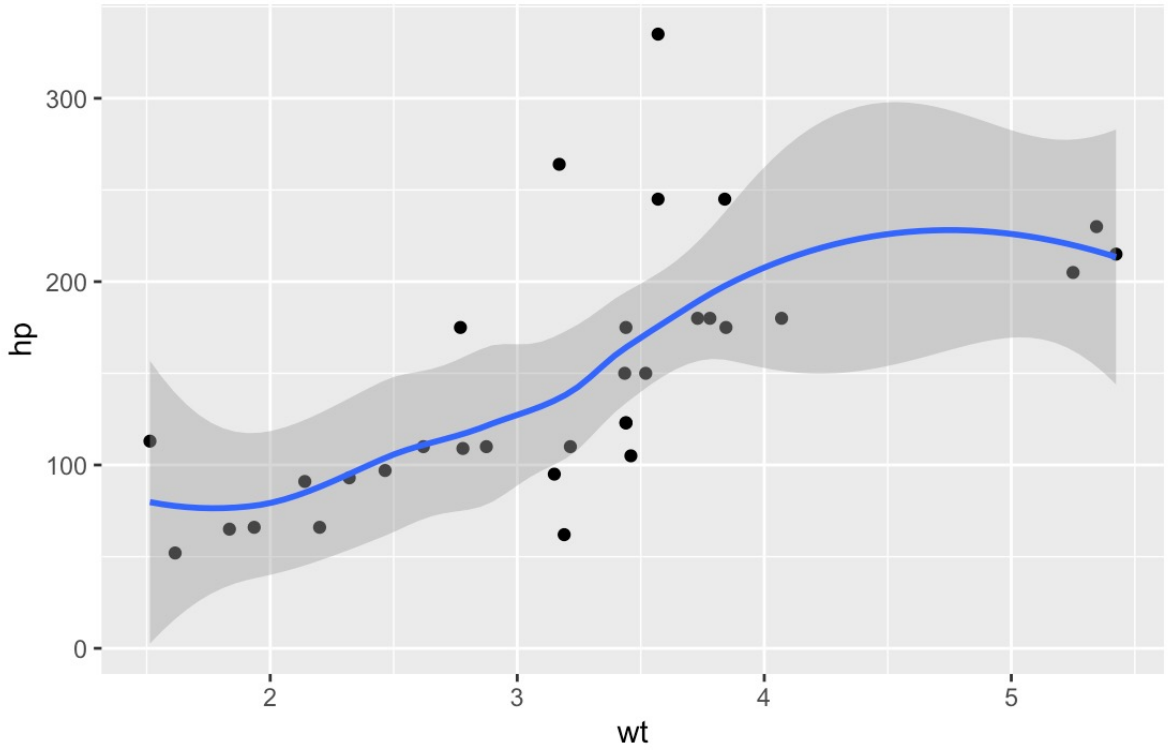
Defined single values go OUTSIDE `aes`

What if you wanted to make all the points red? Where would you put this in the code?

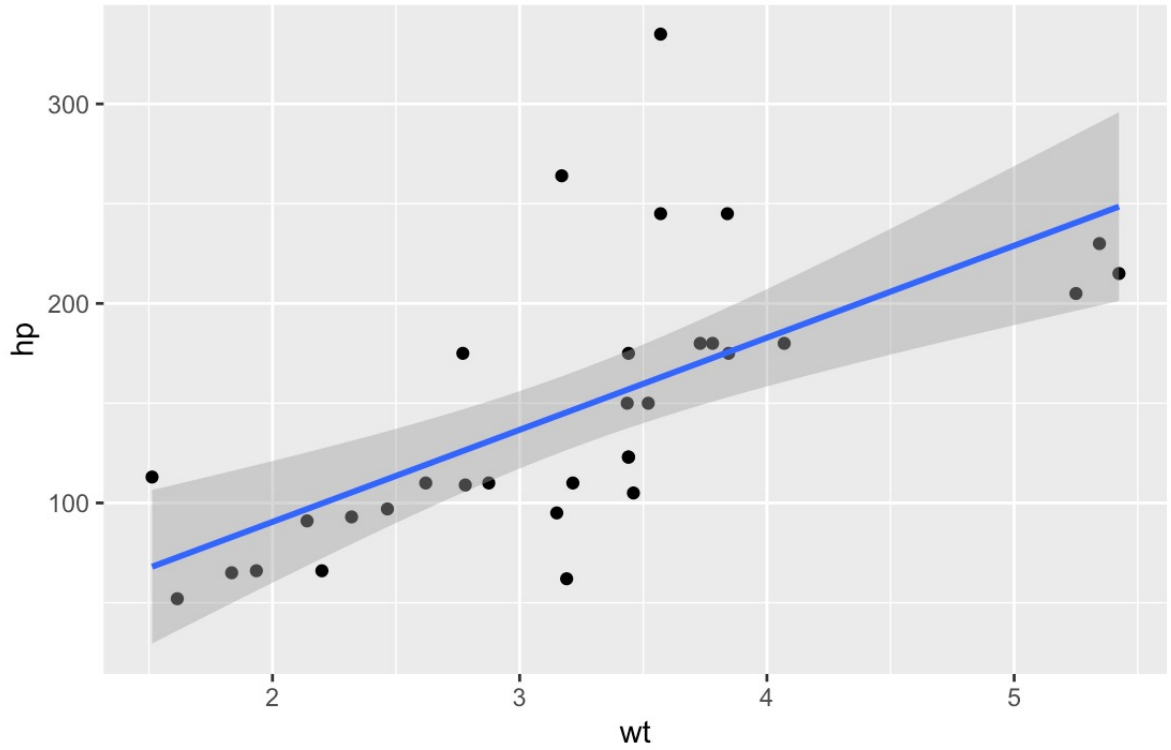


You can build plots up layer by layer using +

```
ggplot(mtcars) +
  geom_point(aes(x = wt, y = hp)) +
  geom_smooth(aes(x = wt, y = hp))
```




```
ggplot(mtcars) +
  geom_point(aes(x = wt, y = hp)) +
  geom_smooth(aes(x = wt, y = hp),
              method = "lm")
```



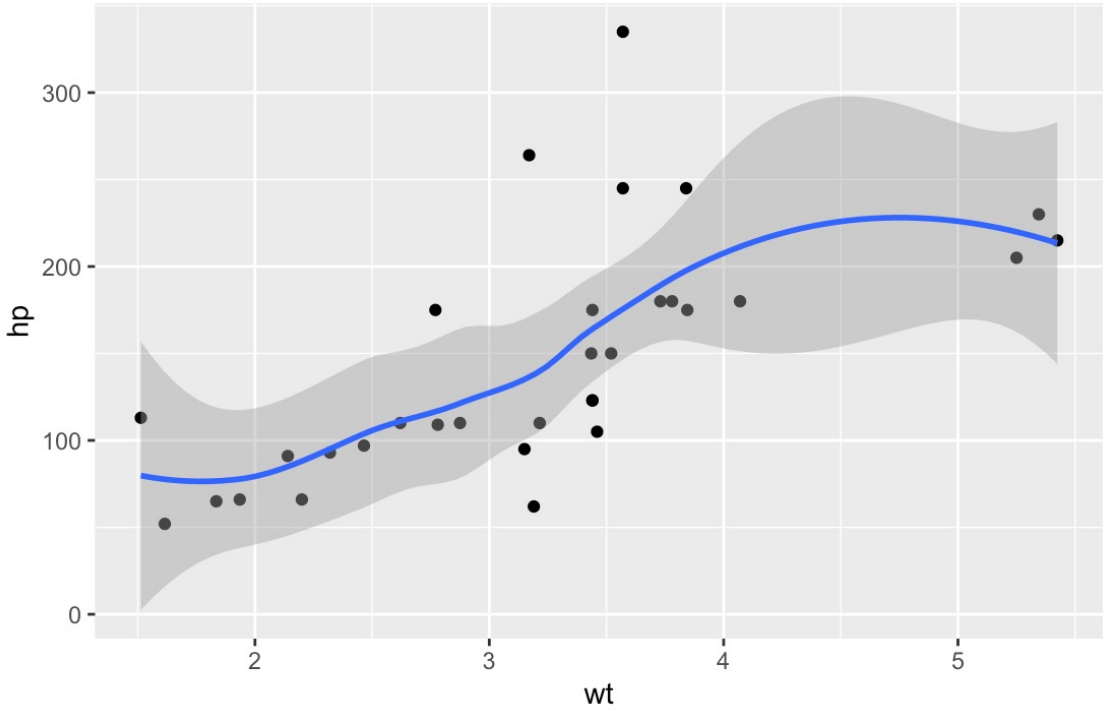
You can also put mapping inside the ggplot part

This code is duplicated (not optimal)

```
ggplot(mtcars) +
  geom_point(aes(x = wt, y = hp)) +
  geom_smooth(aes(x = wt, y = hp))
```

```
ggplot(mtcars, aes(x = wt, y = hp)) +
  geom_point() +
  geom_smooth()
```

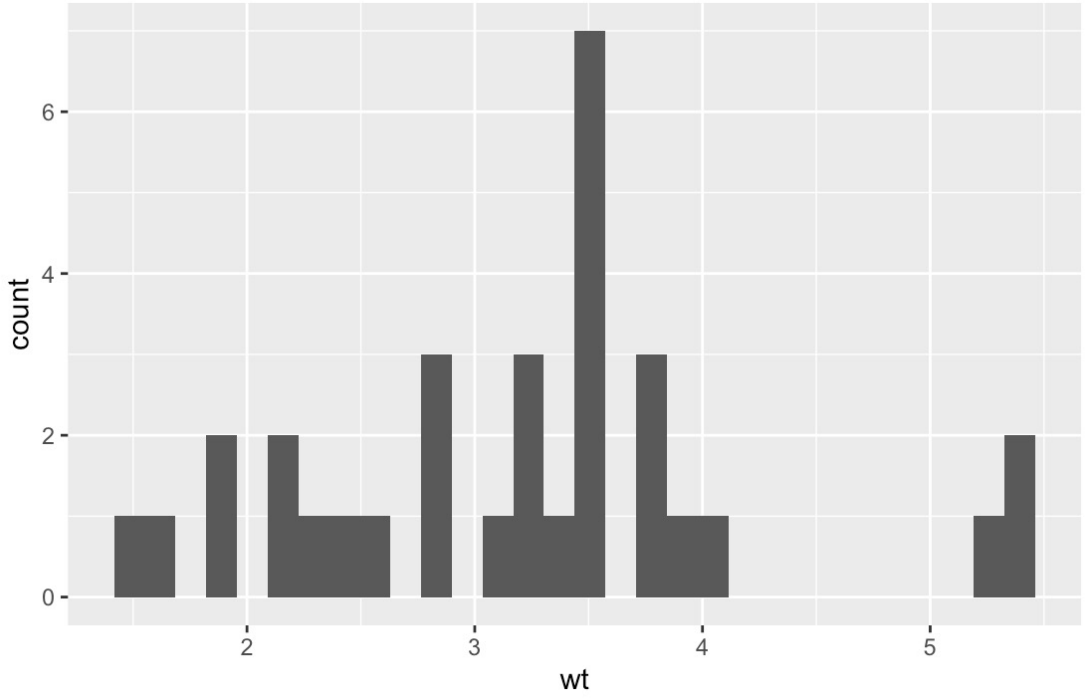
Move it inside the `ggplot` part and all `geom`'s will have the same mapping



Ways to represent data: 1 continuous variable

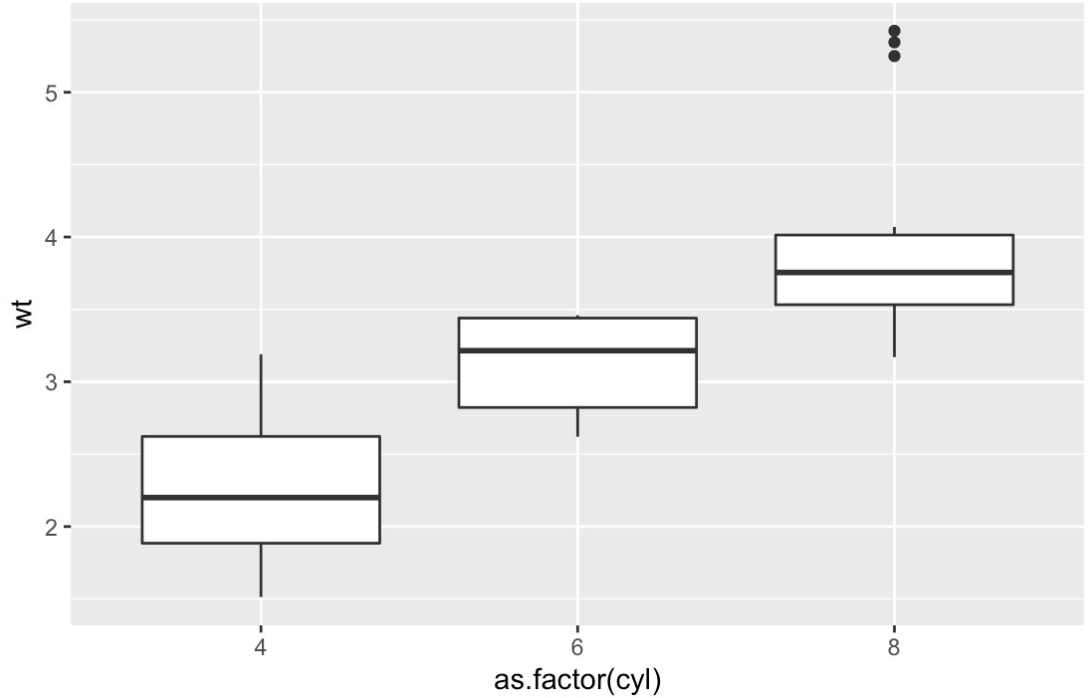
Histograms

```
ggplot(mtcars) +
  geom_hist(aes(x = wt))
```



Boxplots

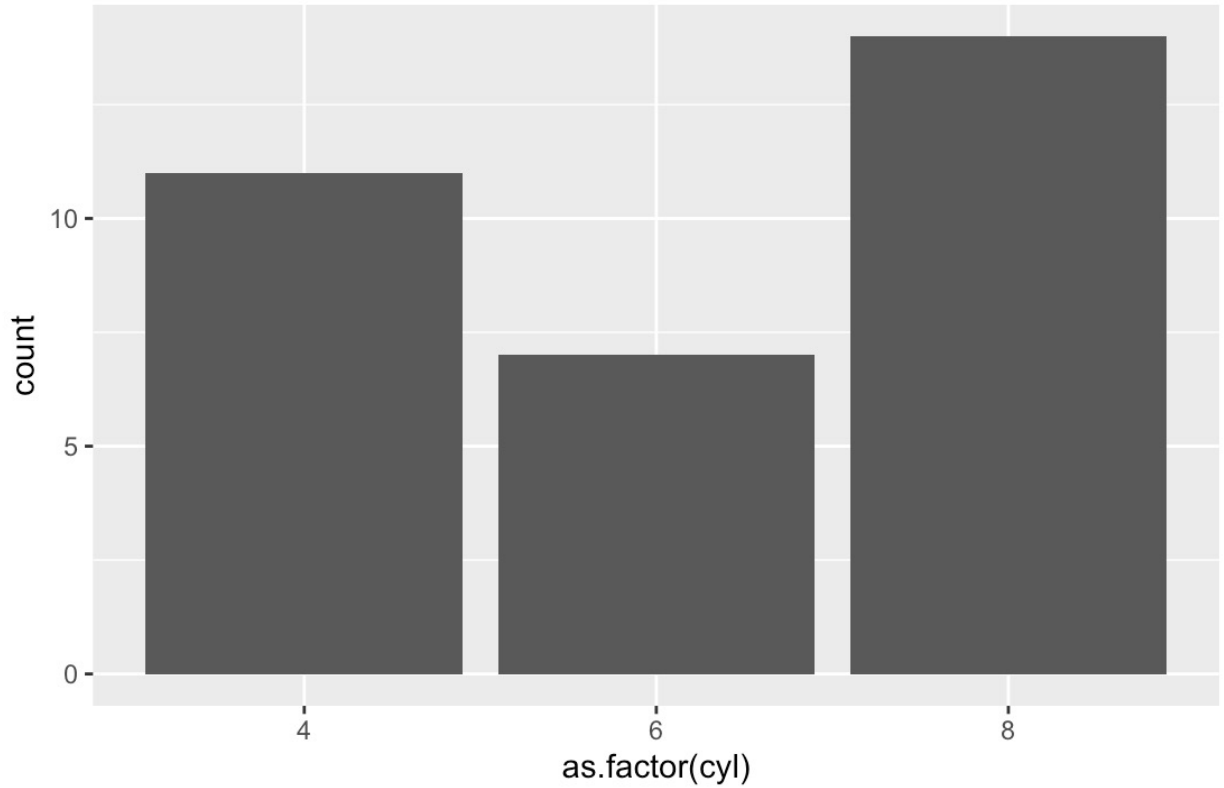
```
ggplot(mtcars) +
  geom_boxplot(aes(x = as.factor(cyl),
                   y = wt))
```



Ways to represent data: 1 categorical variable

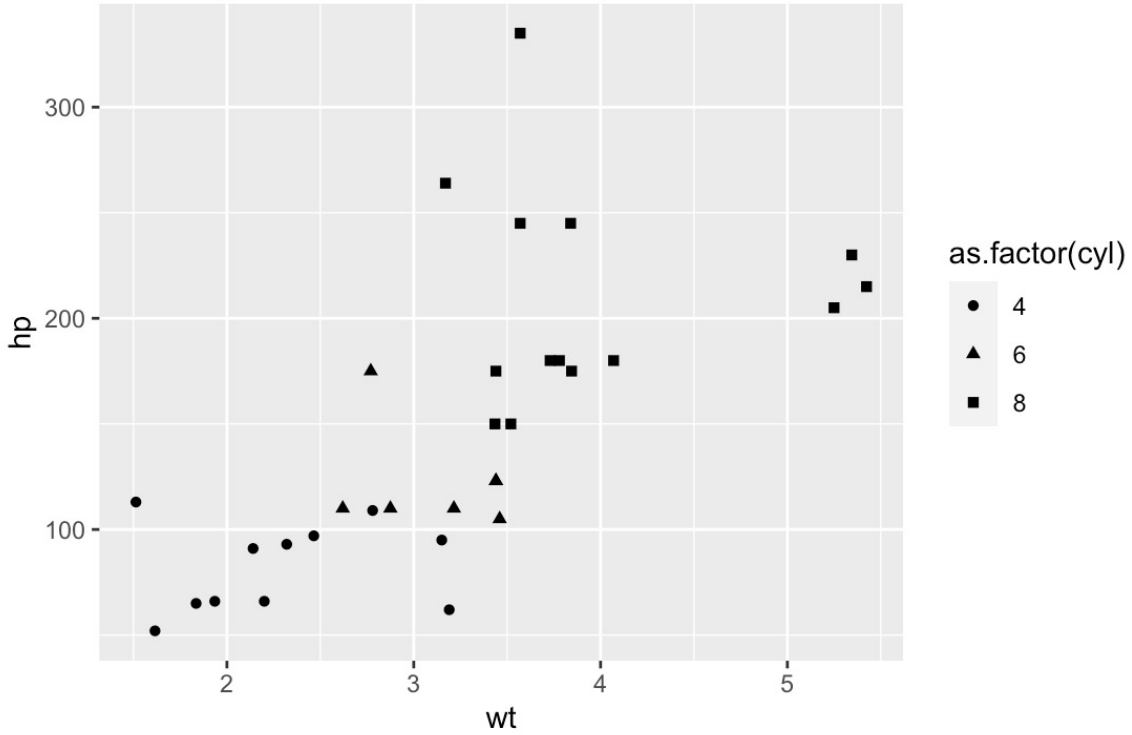
Bar Plots

```
ggplot(mtcars) +
  geom_bar(aes(x = as.factor(cyl))
```

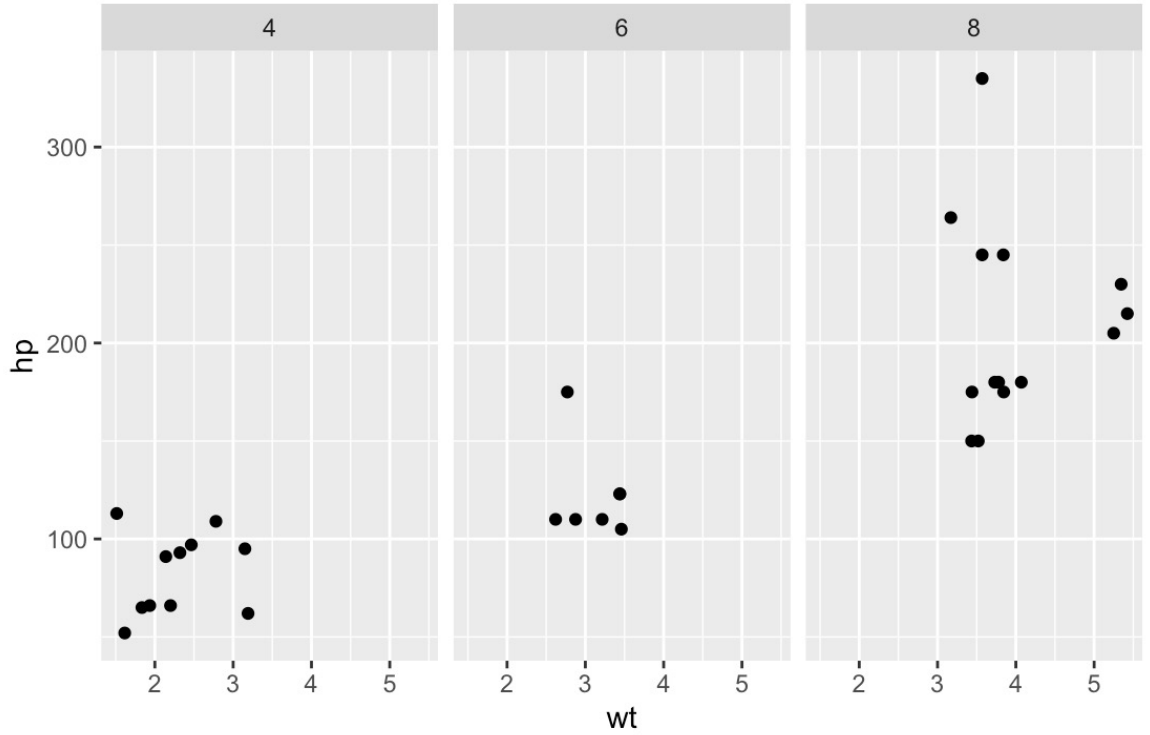


Facetting is useful for multivariate data

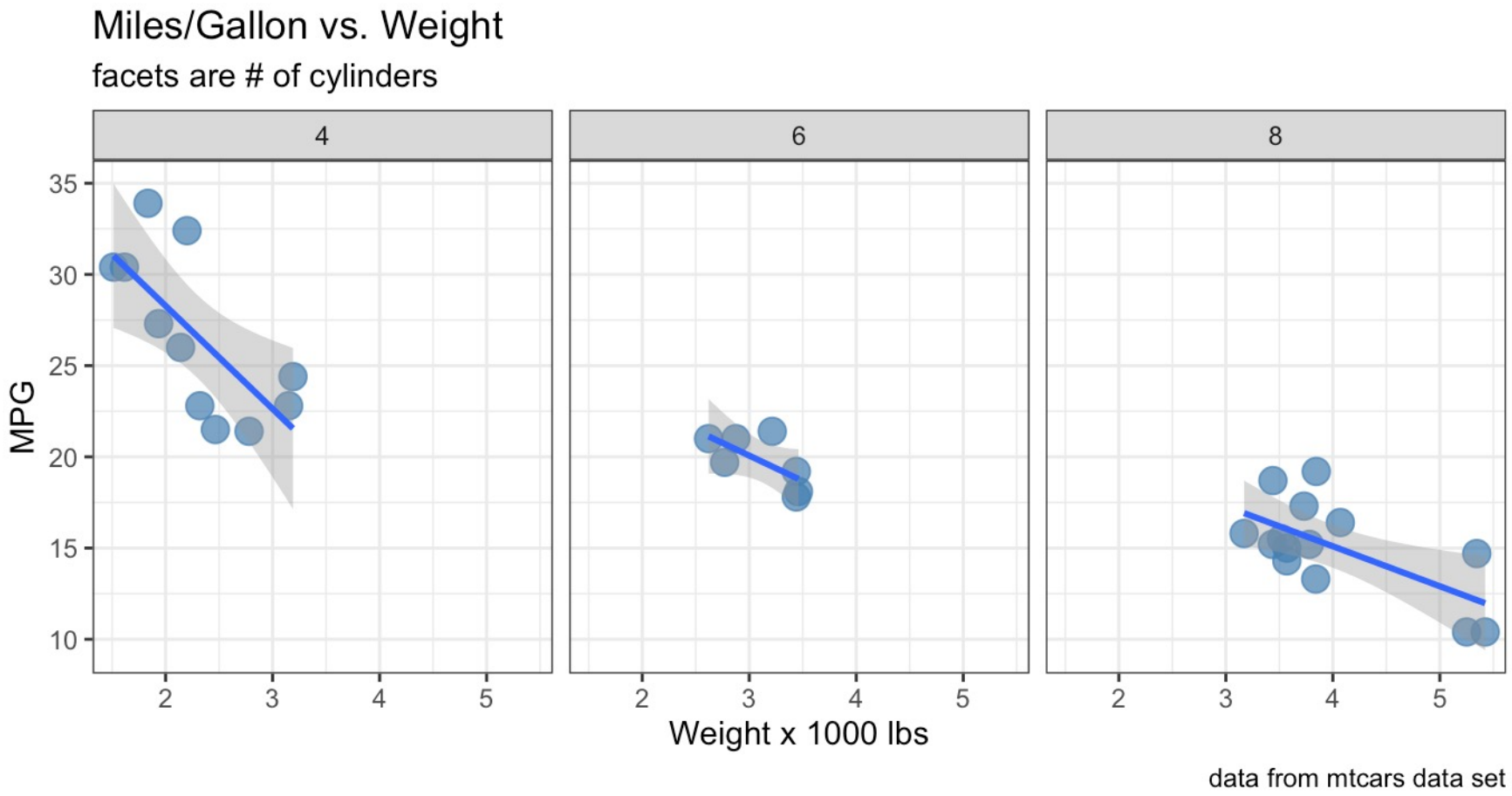
```
ggplot(mtcars) +
  geom_point(aes(x = wt, y = hp,
                 shape = as.factor(cyl)))
```



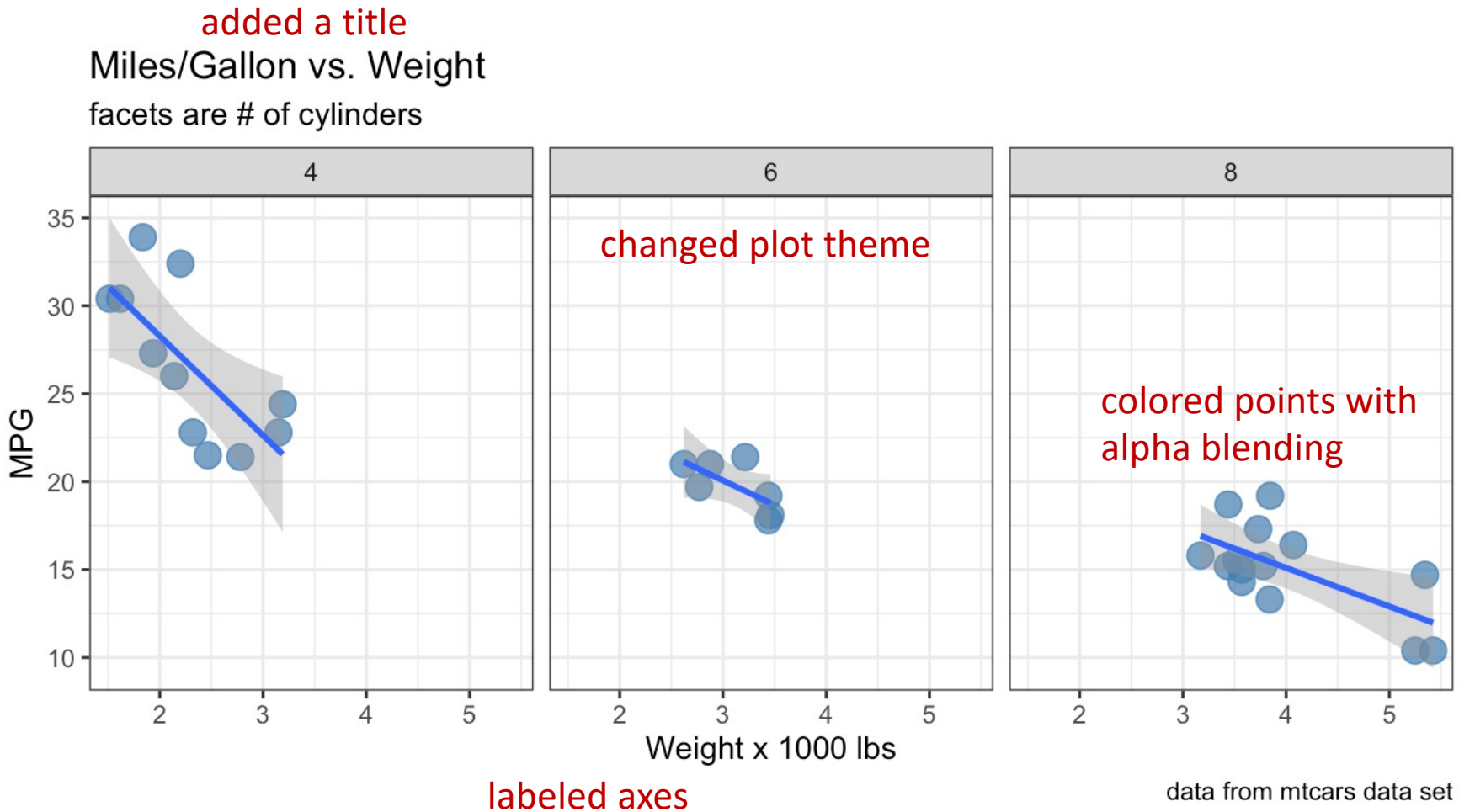

```
ggplot(mtcars) +
  geom_point(aes(x = wt, y = hp)) +
  facet_wrap(~ as.factor(cyl))
```



Let's make a more polished plot

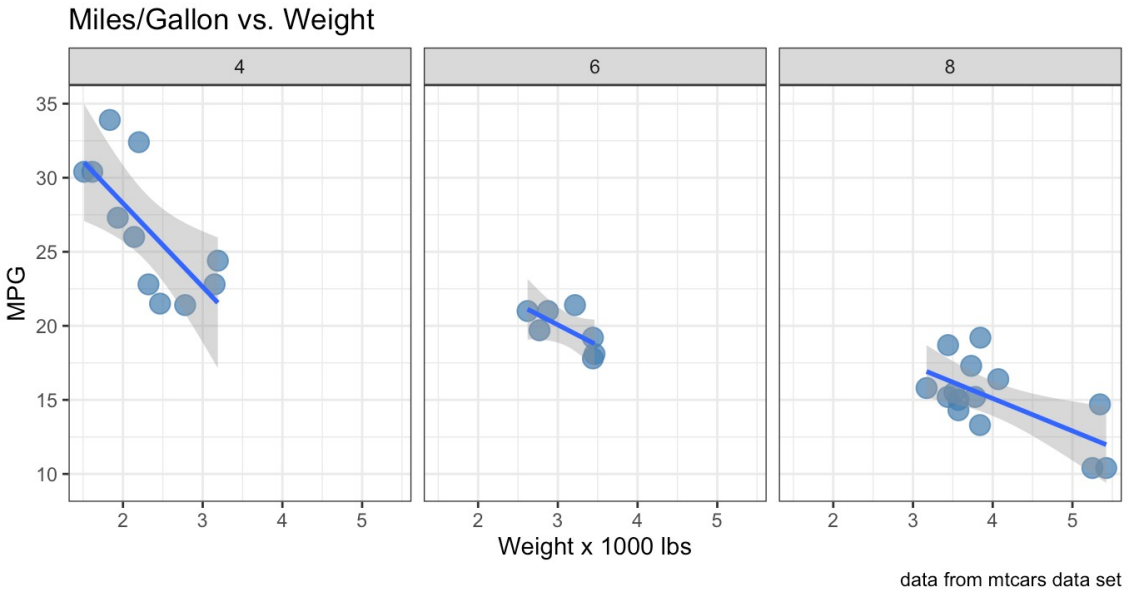


Let's make a more polished plot



Let's make a more polished plot

```
ggplot(mtcars, aes(x = wt, y = mpg)) +
  geom_point(color = "steelblue", size = 4, alpha = 0.75) +
  geom_smooth(method = "lm") +
  facet_wrap(~ as.factor(cyl)) +
  theme_bw() +
  labs(title = "Miles/Gallon vs. Weight",
       subtitle = "facets are # of cylinders",
       caption = "data from mtcars data set",
       x = "Weight x 1000 lbs",
       y = "MPG")
```



The more detailed ggplot2 plotting template

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

Check out the resources to learn more

Resources

- R For Data Science (Chapter 3 in particular)
- H. Wickham's Paper: A Layered Grammar of Graphics
- ggplot2 Reference (browse all the cool things you can do)
- ggplot2 Book