

# Lecture 5.5 – Grammar of Graphics

## Learning Objectives:

1. Learn the basics of `ggplot2`.

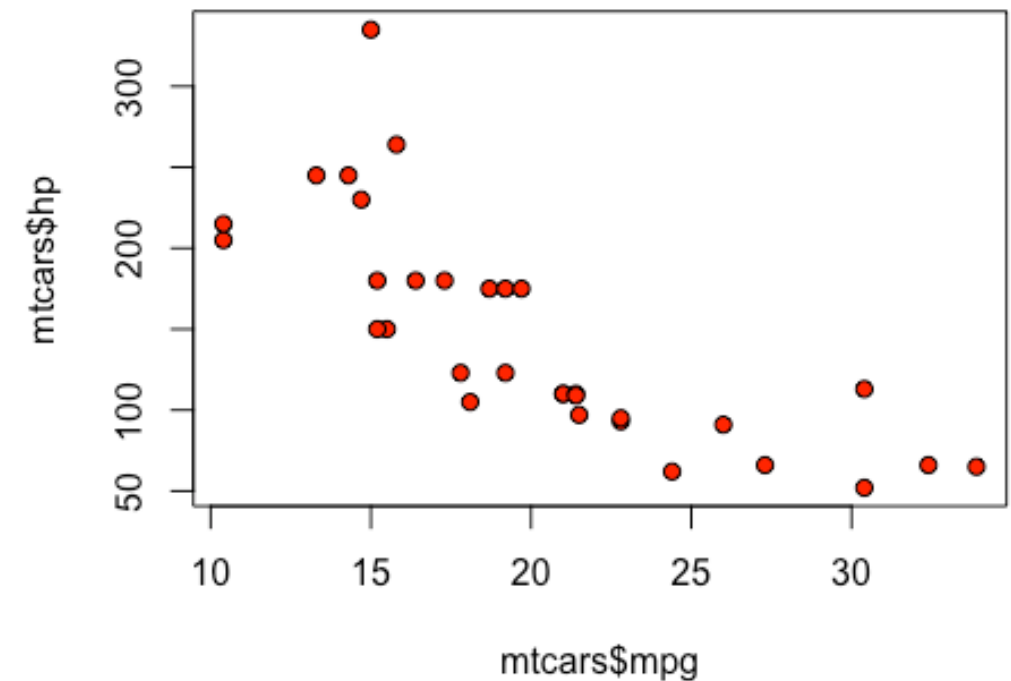
# The Plotting Systems of R

## - Base graphics and **grid**

```
plot(x = mtcars$mpg, y = mtcars$hp,  
     col = "black", bg = "red",  
     pch = 21)
```

### Benefits:

- simple and quick
- handles a variety of data types
- not many background calculations

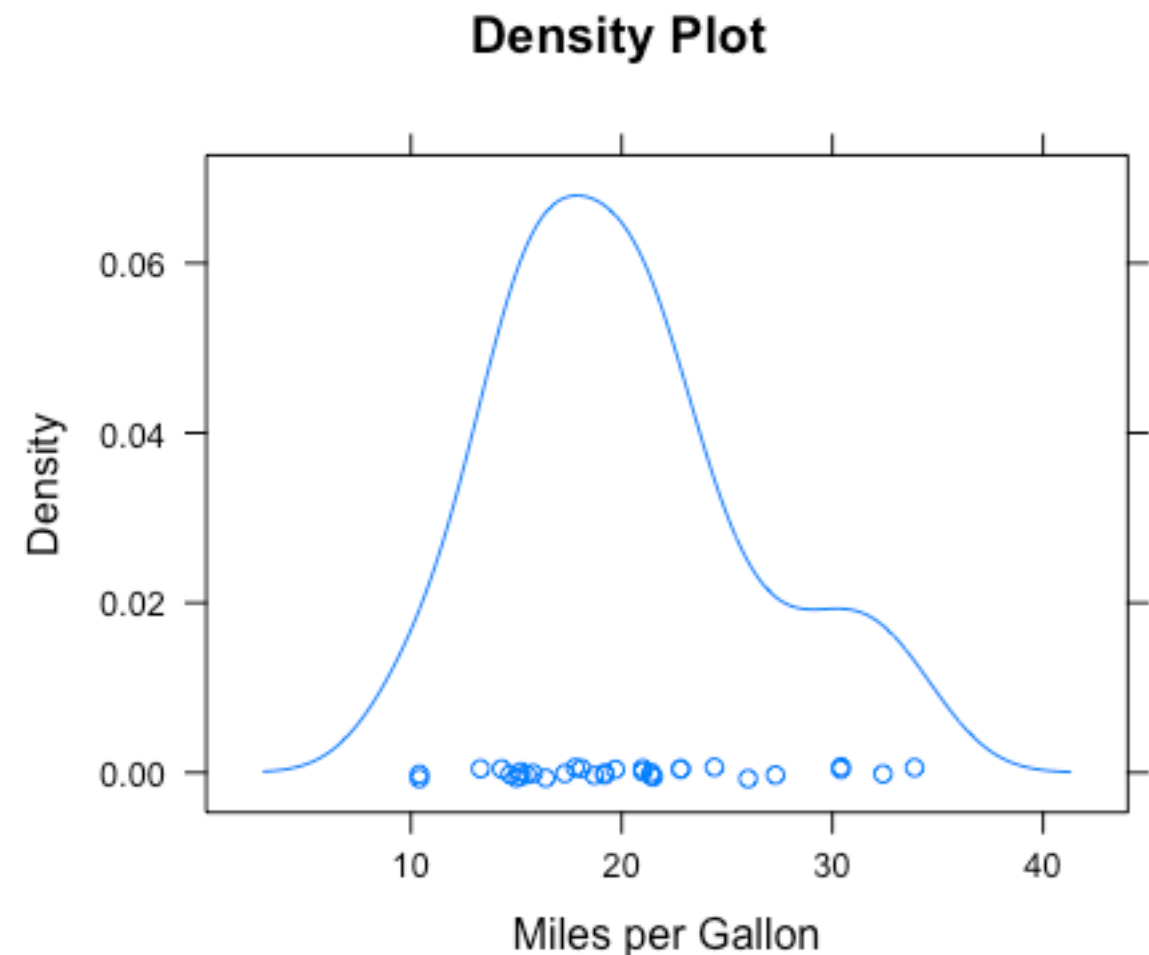


## - The Trellis system in **lattice**

```
densityplot(~mpg,  
            main="Density Plot",  
            xlab="Miles per Gallon")
```

### Benefits:

- quick
- many specialized stats plots
- not too many background calculations



# The Plotting Systems of R

- Grammar of Graphics `ggplot2`

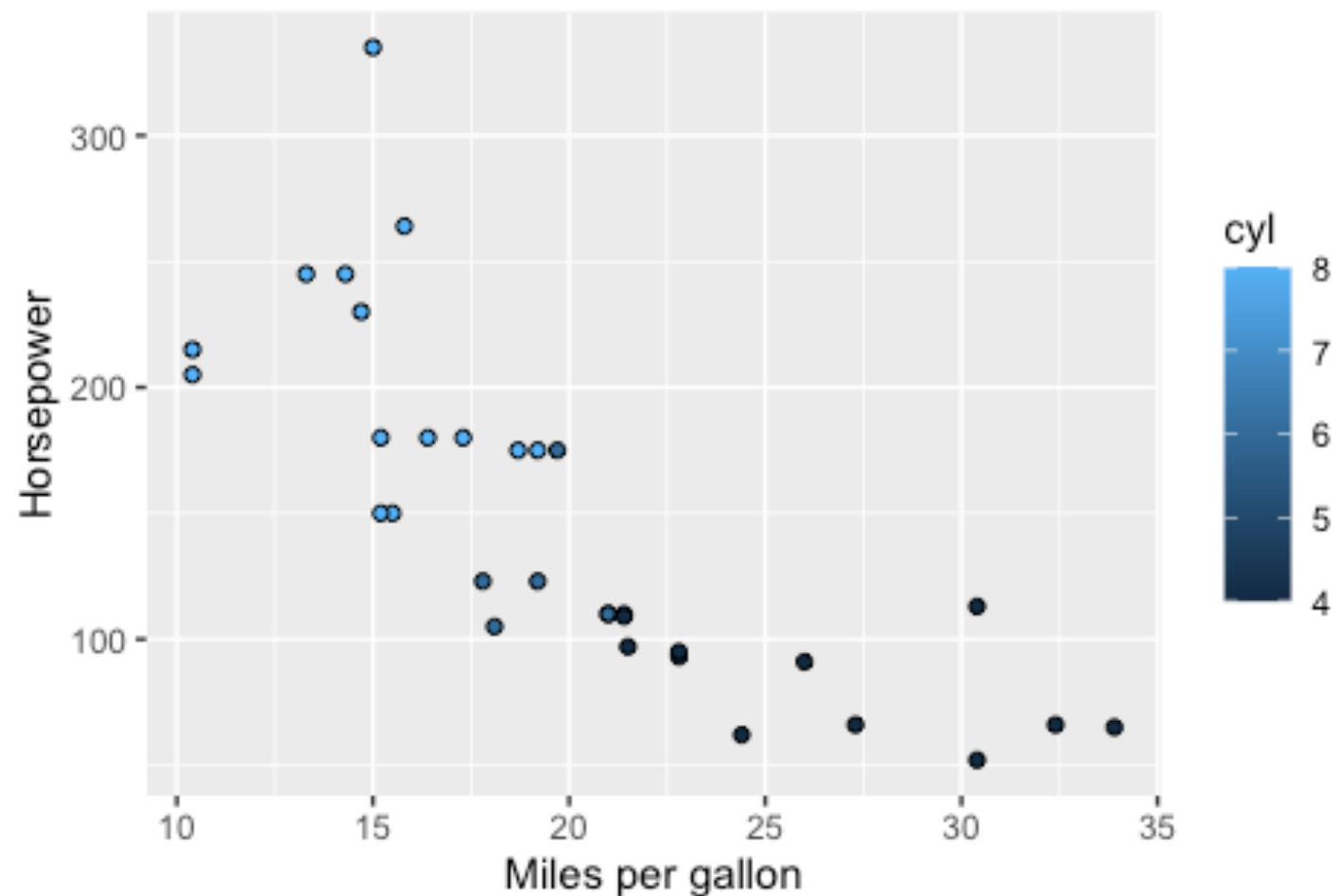
```
ggplot(mtcars, aes(x=mpg, y=hp, fill=cyl)) +  
  geom_point(pch=21, size=2) +  
  xlab("Miles per gallon") + ylab("Horsepower")
```

## Benefits:

- beautiful visualizations
- many specialized plots
- consistent syntax
- have full control over aesthetics

## Drawbacks:

- lots of background calculations
- slow
- only uses data frames in long format



# Grammar of Graphics: terminology

## Data

	mpg	cyl	displacement	hp
Mazda RX4	21.0	6	160.0	110
Mazda RX4 Wag	21.0	6	160.0	110
Datsun 710	22.8	4	108.0	93
Hornet 4 Drive	21.4	6	258.0	110
Hornet Sportabout	18.7	8	360.0	175
Valiant	18.1	6	225.0	105
Duster 360	14.3	8	360.0	245
Merc 240D	24.4	4	146.7	62
Merc 230	22.8	4	140.8	95

## Mapping

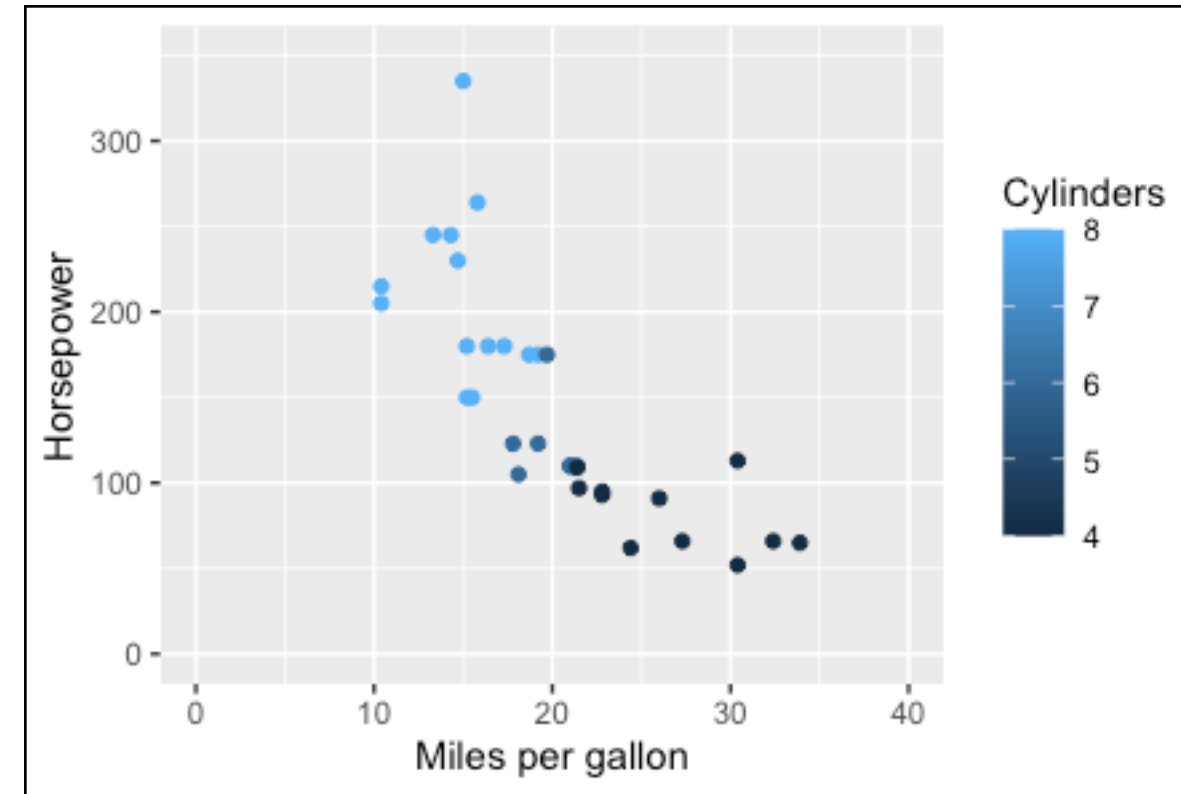
## Aesthetic attributes

x

y

color/fill

## Geometric object



`ggplot( mtcars,`

`aes( x = mpg,`  
 `y = hp,`  
 `fill = cyl)`

`) + geom_point()`

## Scales

`+ xlim(0, 40)`

## Guides

`+ labs(color="Cylinders")`

**Other stuff:**

# Changing Basic Plot Attributes

- Changing basic attributes:

**Base plot:** `p <- ggplot(mtcars, aes(x = mpg, y = hp)) + geom_point()`

- Plot labels and titles:

**X-axis label:** `p + xlab("Miles per Gallon")`

**Y-axis label:** `p + ylab("Horsepower")`

**Plot title:** `p + ggtitle("Horsepower vs MPG from MtCARS")`

- Axis scales:

**Change x limits:** `p + xlims(min, max)`

**Change y limits:** `p + ylims(min, max)`

**Change color limits:** `p + lims(color = c(newlimits))`

- Formatting of plot area and text:

**Most items:** `p + theme()`

**Minimal theme:** `p + theme_minimal()`

**There are lots of  
arguments here!**



# Changing Aesthetic Attributes

If independent of mapping,  
you will change them in the  
geom!

- Changing aesthetic attributes:

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

- Point size, shape, color, and fill:

**Point size:** ... + geom\_point(size = 2)

(2 is 2x size of  
default points)

**Point shape:** ... + geom\_point(shape = 19)

**or**

... + geom\_point(pch = 19)

**Point color:** ... + geom\_point(color = "red")

**Point fill:** ... + geom\_point(pch = 21,  
color = "black", fill = "red")

# Check Your Understanding

**Use the Orange data set to create a scatterplot with `ggplot2` of circumference versus tree age.**

**Add to the plot:**

- 1. Adding axis labels and a title.**
- 2. Changing the color, shape, and size of the points.**
- 3. Rotating the x-axis labels by 45 degrees.**

# Data: wide versus long format

- To reap many of the benefits of the `ggplot` package, you may need to reshape your data set.

**Wide:** Columns represent different measurements

Seed	Year 3	Year 5	Year 10	Year 15	Year 20	Year 25
301	4.51	10.89	28.72	41.75	52.70	60.92
303	4.55	10.92	29.07	42.83	53.88	63.39
305	4.79	11.37	30.21	44.40	52.82	64.10
307	4.81	11.20	28.66	41.66	53.31	63.05

**Long:** Each row is a unique observation

Seed	Age	Height
301	3	4.51
301	5	10.89
301	10	28.72
301	15	41.74
301	20	52.70
301	25	60.92
303	3	4.55
303	5	10.92
303	10	29.07
303	15	42.38
303	20	53.88
303	25	63.39



# Data: wide versus long format

**Dataset:** Loblolly

**Package:** tidyr

**Wide:** Columns represent different measurements

Seed	Year 3	Year 5	Year 10	Year 15	Year 20	Year 25
301	4.51	10.89	28.72	41.75	52.70	60.92
303	4.55	10.92	29.07	42.83	53.88	63.39
305	4.79	11.37	30.21	44.40	52.82	64.10
307	4.81	11.20	28.66	41.66	53.31	63.05

```
Loblolly.wide <- pivot_wider(Loblolly,  
                             names_from = age,  
                             values_from = height)
```

```
Loblolly.long <-  
pivot_longer(Loblolly.wide, id.vars =  
             "Seed")
```

**Long:** Each row is a unique observation

Seed	Age	Height
301	3	4.51
301	5	10.89
301	10	28.72
301	15	41.74
301	20	52.70
301	25	60.92
303	3	4.55
303	5	10.92
303	10	29.07
303	15	42.38
303	20	53.88
303	25	63.39

# Data: wide versus long format

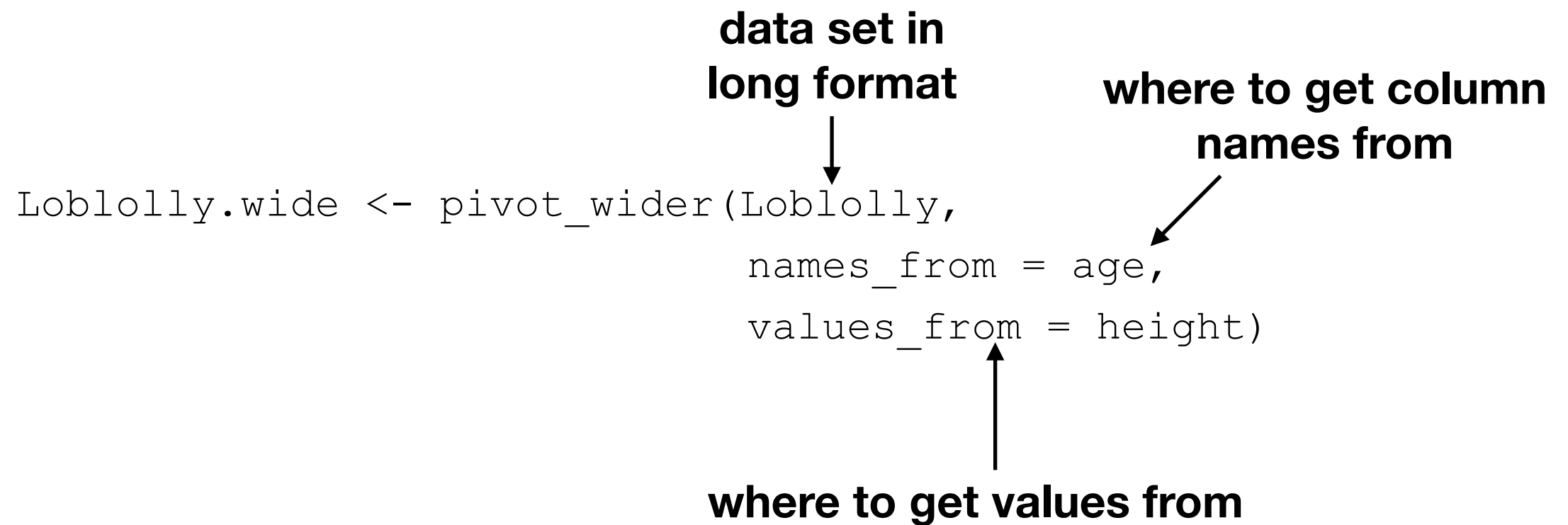
- The `pivot_wider()` function: long to wide

**data set in  
long format**

**where to get column  
names from**

```
Loblolly.wide <- pivot_wider(Loblolly,  
                             names_from = age,  
                             values_from = height)
```

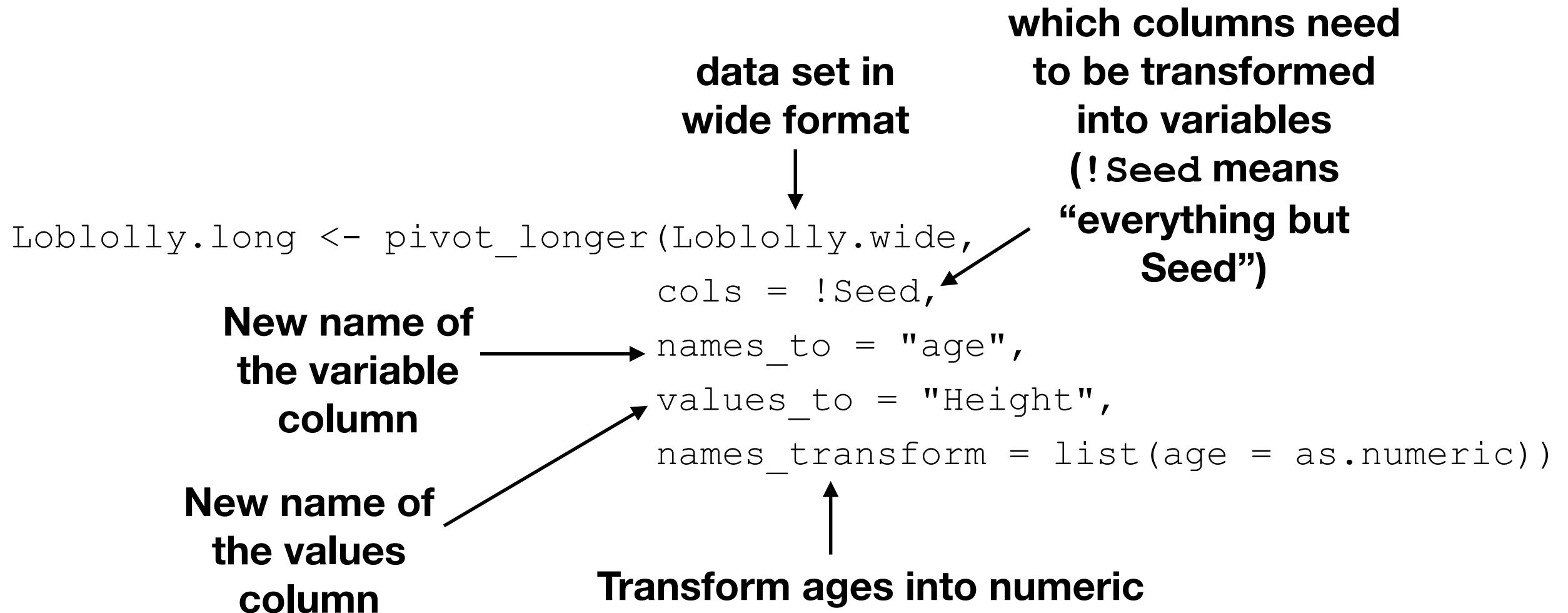
**where to get values from**



- The new columns are named based on **names\_from**
- The cell values are filled from **values\_from**

# Data: wide versus long format

- The `pivot_longer()` function: wide to long



- Pick the columns that should be data values (`cols`). The column names will become variables in long format. The variable column will be named by `names_to`.
- Pick the name of that new column where the values will be stored (`values_to`).

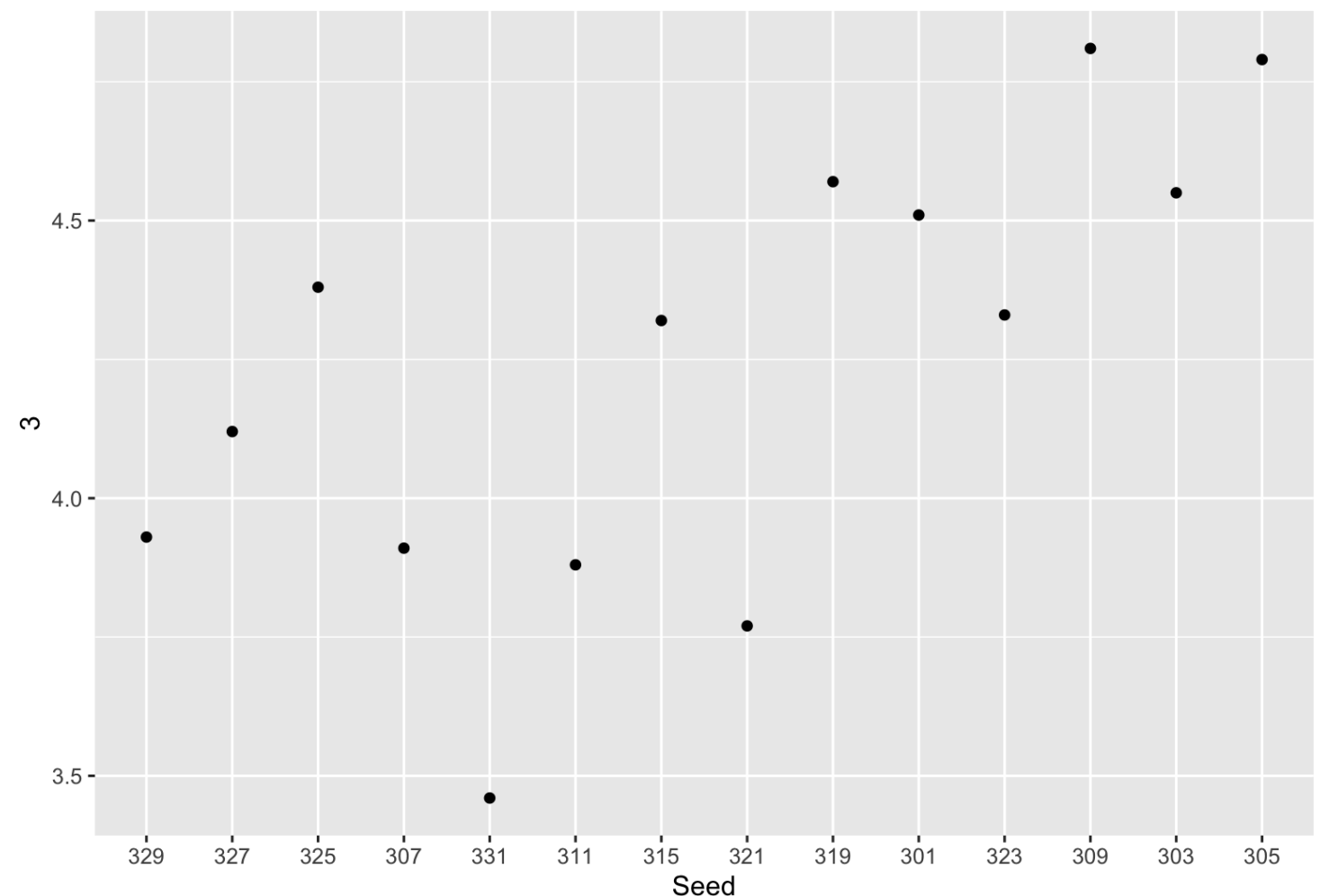
# Plotting: wide format

- Plotting in the wide format limits your options. You are forced to choose which columns to map, which limits you to the ways in which you can plot!

**column options for `Loblolly.wide`:**

	Seed	3	5	10	15	20	25
1	329	3.93	9.34	26.08	37.79	48.31	56.43
2	327	4.12	9.92	26.54	37.82	48.43	56.81

```
ggplot(Loblolly.wide,  
       aes(x = Seed,  
           y = `3`)) +  
  geom_point()
```



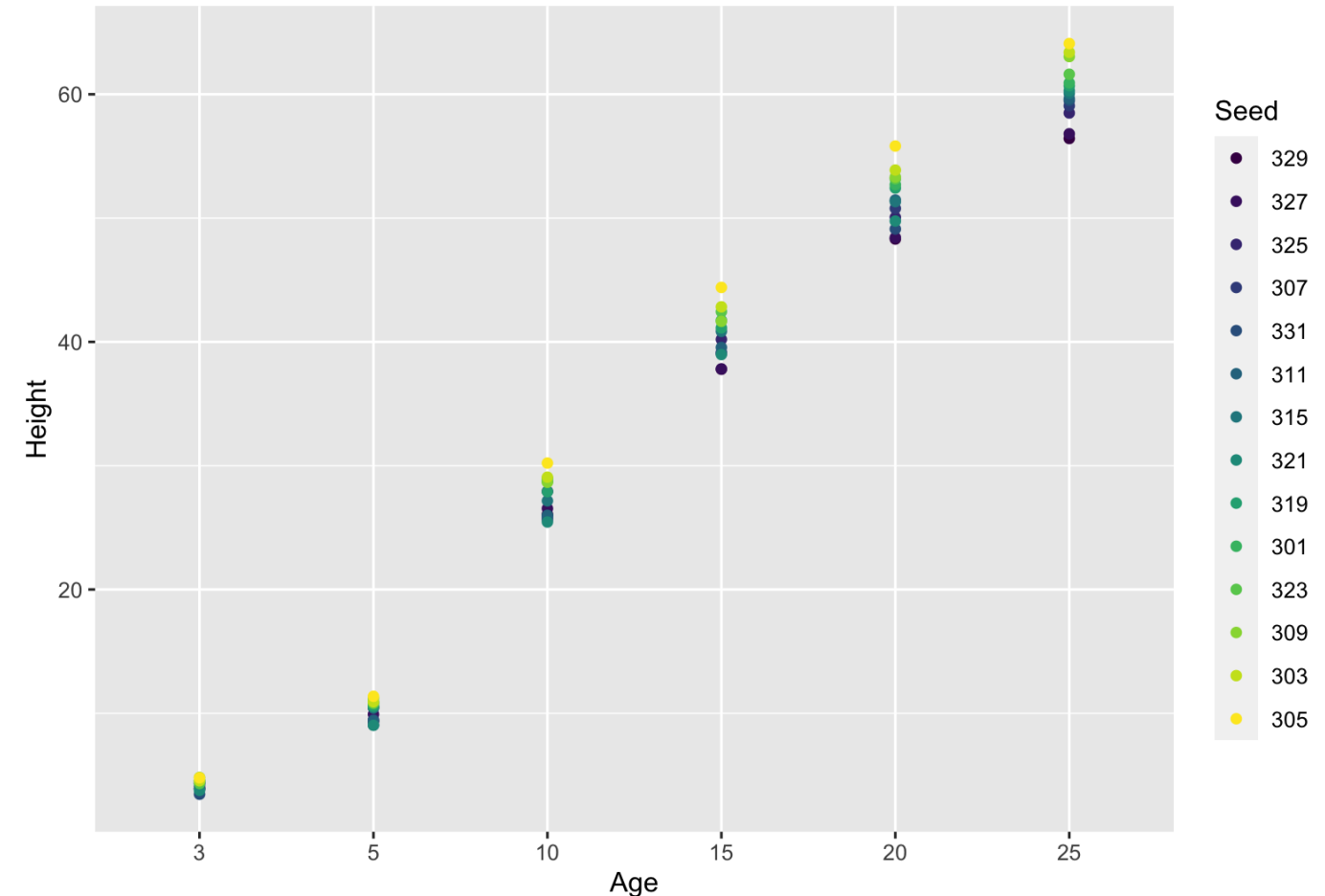
# Plotting: long format

- The long format doesn't suffer from this same issue. You have appropriate options for plotting and mapping aesthetics.

**column options for `Loblolly.long`:**

	Seed	Age	Height
1	329	3	3.93
2	327	3	4.12
3	325	3	4.38

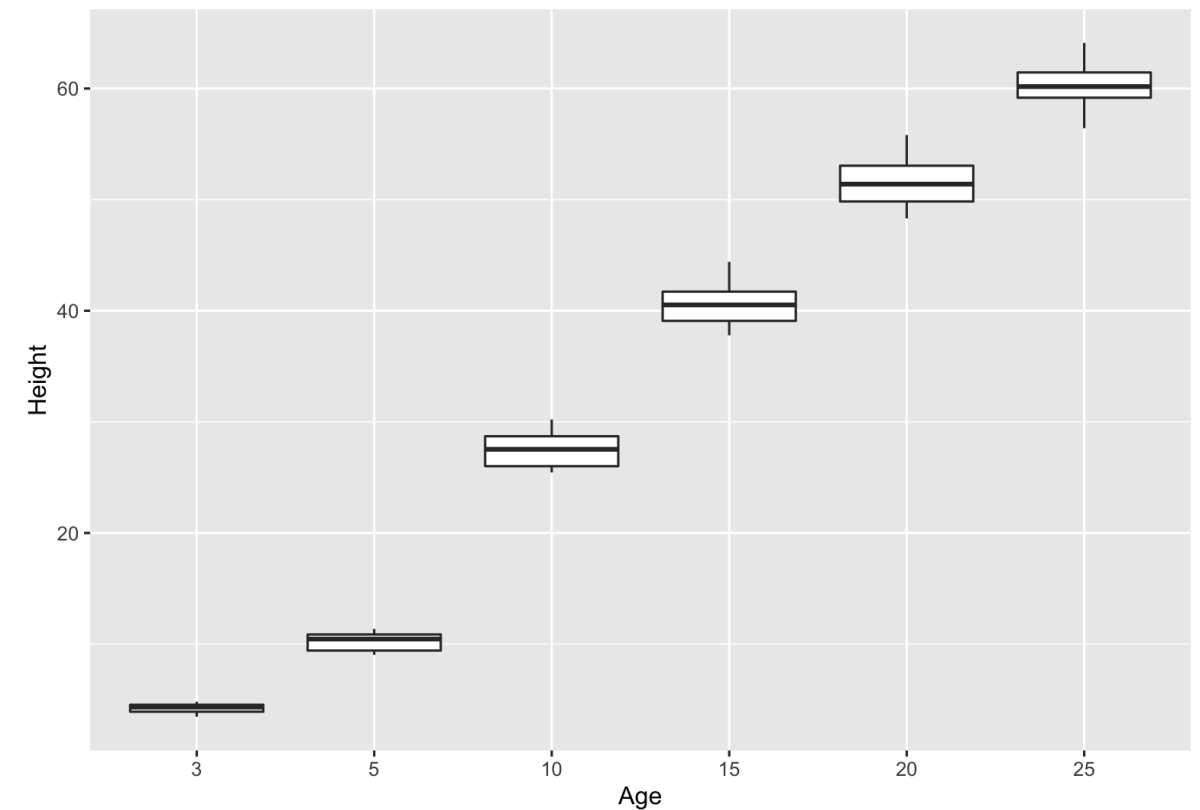
```
ggplot(Loblolly.long,  
       aes(x = age,  
           y = Height,  
           color = Seed)) +  
  geom_point(pch = 19)
```



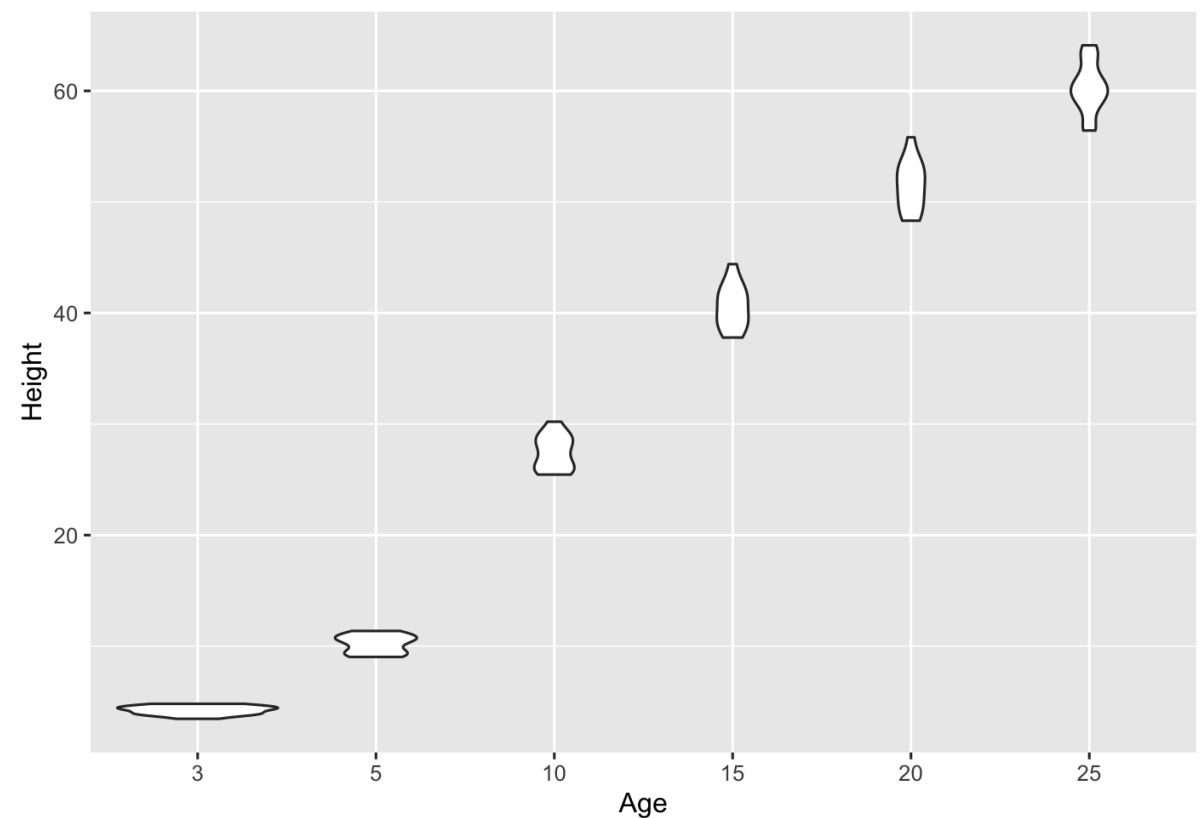
# Plotting: long format

- You can push many calculations for plot types into **ggplot** so as not to worry about doing them yourself!

```
ggplot(Loblolly.long,  
       aes(x = factor(age),  
           y = Height)) +  
  geom_boxplot()
```



```
ggplot(Loblolly.long,  
       aes(x = factor(age),  
           y = Height)) +  
  geom_violin()
```



# Check Your Understanding

1. Reshape the `fish_encounters` data set (in the `tidyr` package) into a wide format.
2. Reshape the `relig_income` data set (in the `tidyr` package) into a long format.

# Grouping and Aesthetic Values

- It's easy to change group characteristics using `aes()`, include the specific characteristic you want (`color`, `fill`, `shape`) or use `group` and define those later.

**Assign colors based on `cyl`:**

```
ggplot(mtcars, aes(x = mpg, y = hp,  
                  color = factor(cyl))) +  
  geom_point()
```

**Assign fill based on `cyl`:**

```
ggplot(mtcars, aes(x = mpg, y = hp,  
                  fill = factor(cyl))) +  
  geom_point(pch = 21)
```

**Assign shape based on `cyl`:**

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

**Note: using `factor(cyl)` will help out these plots!**



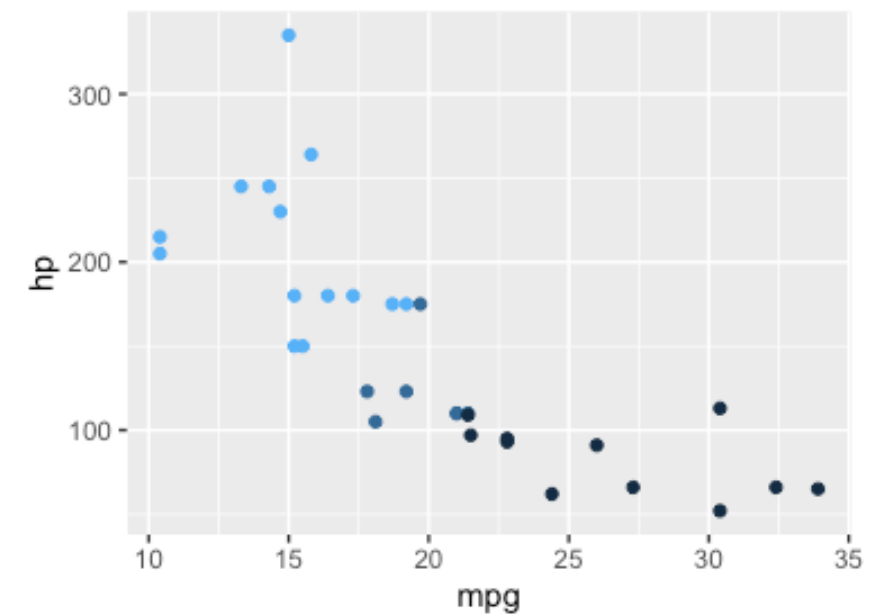
# Mapping aesthetics based on data

- Mapping aesthetics that change with data must be done with `aes()`

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



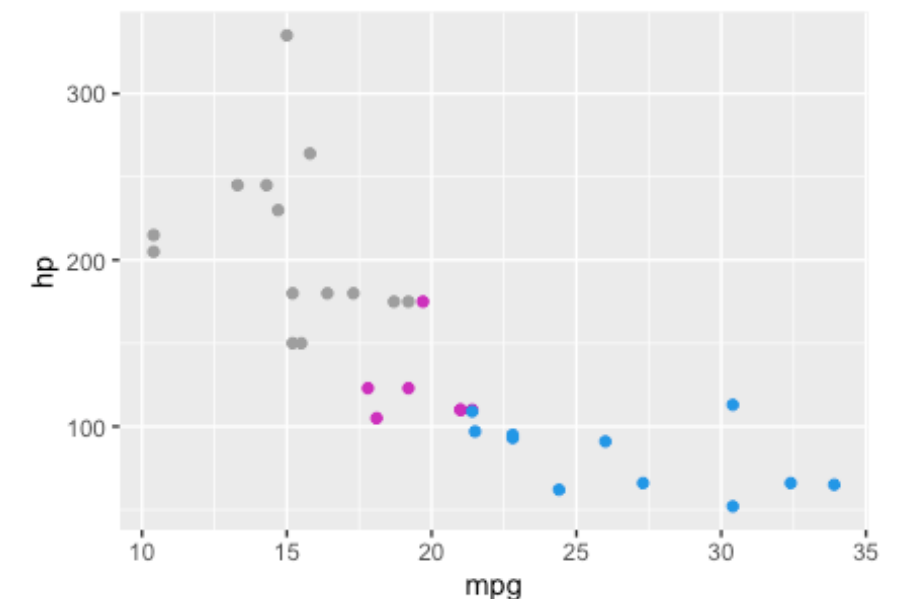
**Maps correctly**



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



**Maps incorrectly**



- Changing non-mapped aesthetics can be done anywhere

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

**Maps correctly**

# Adding data from other sets to existing plots

- Adding data manually from other sets is very clunky and should be avoided whenever possible.

```
p2 <- ggplot(Loblolly.wide, aes(x = Seed, y = `3`)) +  
  geom_point()
```

**Add another column from `Loblolly.wide` to the graph as red dots:**

```
p2 + geom_point(data = Loblolly.wide,  
  mapping = aes(x = Seed, y = `5`),  
  color = "red")
```

- the data set must be specified in the geom using **data=**
- mapping must be specified in the geom using **mapping=** and **aes()**
- other attributes have to be set independently in the geom
- it is ugly and won't give you a correct legend

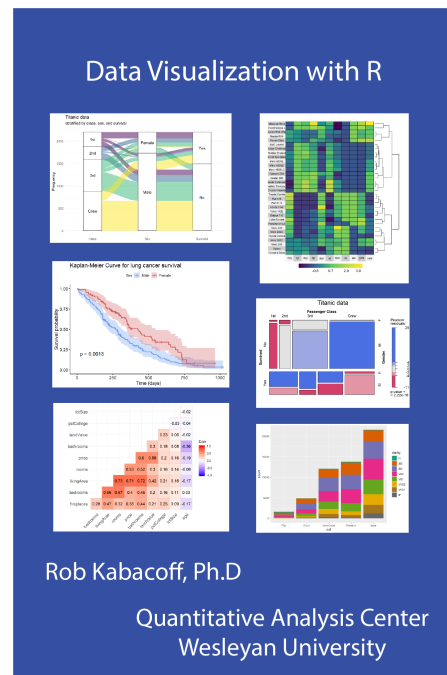
# Check Your Understanding

**Add to the plot in your last check your understanding by:**

- 1. Coloring the plot points by Tree**
- 2. Adding lines in the link data from the individual trees together**

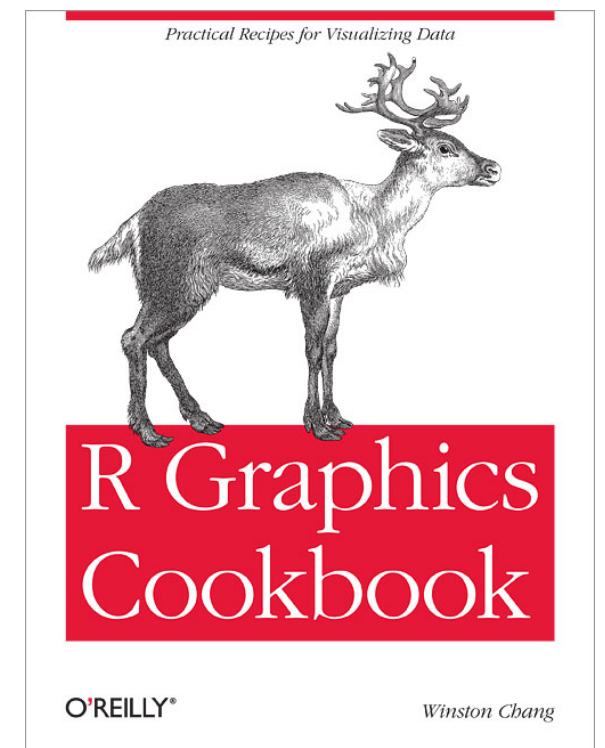
# Excellent Resources for ggplot2

**Tidyverse Reference Guide – <https://ggplot2.tidyverse.org/reference/>**



**Data Visualization with R – <https://rkabacoff.github.io/datavis/>**

**The R Graphics Cookbook – <https://r-graphics.org/>**



# Additional Resources

**<https://www.stat.auckland.ac.nz/~ihaka/787/lectures-trellis.pdf> –**

**The Trellis system in lattice (PDF lecture slides)**

**[http://www.cookbook-r.com/Manipulating\\_data/](http://www.cookbook-r.com/Manipulating_data/)**

**Converting data between wide and long format/ –**

**Converting between long and wide format with reshape2, tidyr, and base R**