Data Analytics
CS301
Plotting and Basic Data Transformations

Week 4: 22nd and 24th Sept

Fall 2020

Oliver BONHAM-CARTER

Ask the Mileage Data

Ask: *What classes of cars (i.e,. suv's, trucks, etc.) get the best city and highway mileage?*

I know! I will use some MPG data from the Tidyverse library and see what the data says!!

```
library(tidyverse)
  # check the data
View(mpg)
  # run simple plot
ggplot(data = mpg) +
 geom_point(mapping = aes(x = mpg$displ, y = mpg$hwy ))
```

From Last Time: Code for a Simple GGPlot

- **ggplot(data = mpg) + geom_point(mapping = aes(x = displ, y = hwy))**

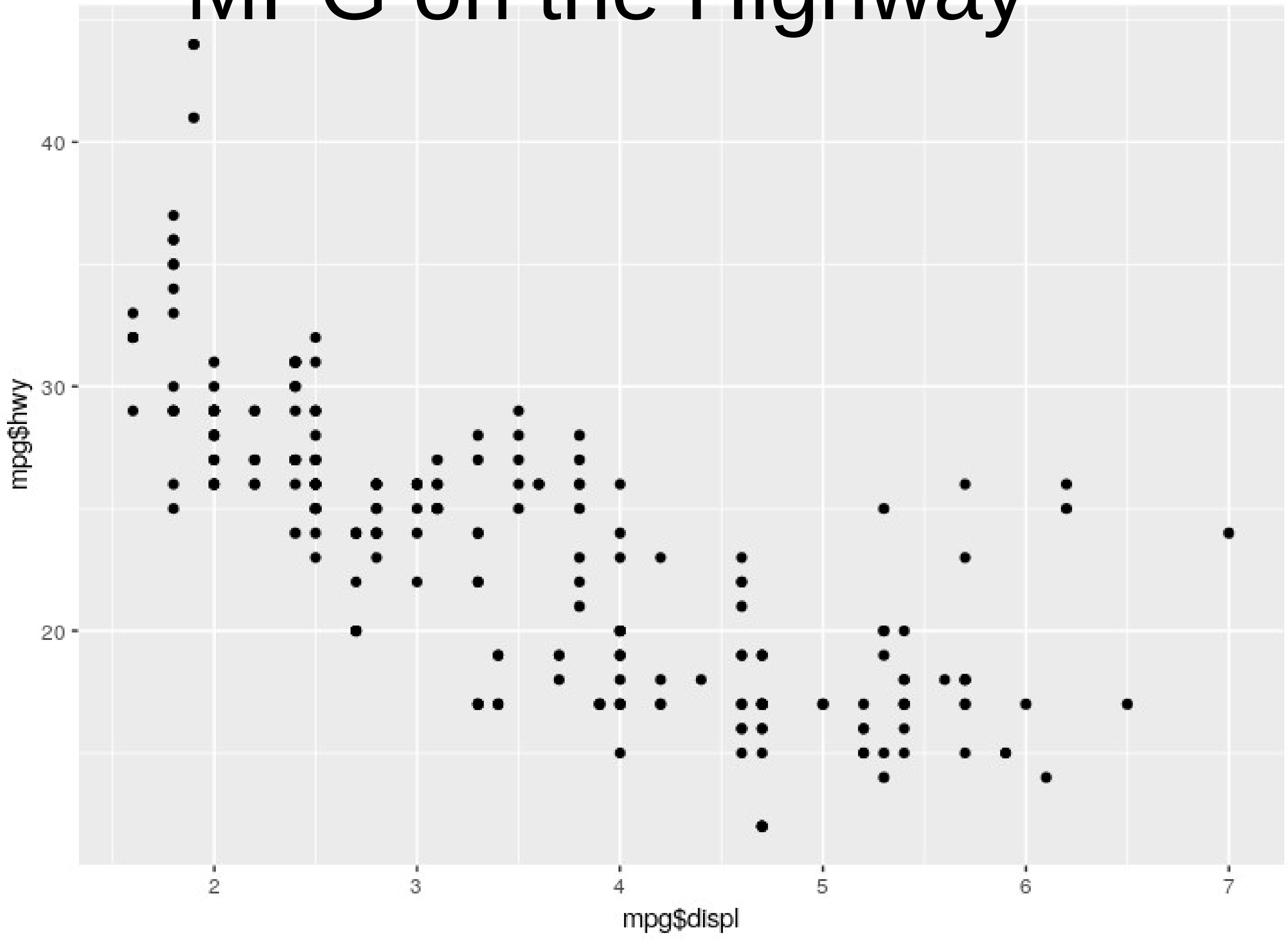
- Establish the *canvas* (where the plot is shown)
- **Ggplot()**

- Link to the data (set is called, ‘mpg’)
  - **ggplot(data = mpg)**

- Compute the geometery of point placement on canvas
  - **geom_point(mapping = …)**

- Compute the aesthetics of the plot (titles, color, point type, etc)
  - **aes(x = displ, y = hwy)**

Displacement (Car Weight) by MPG on the Highway

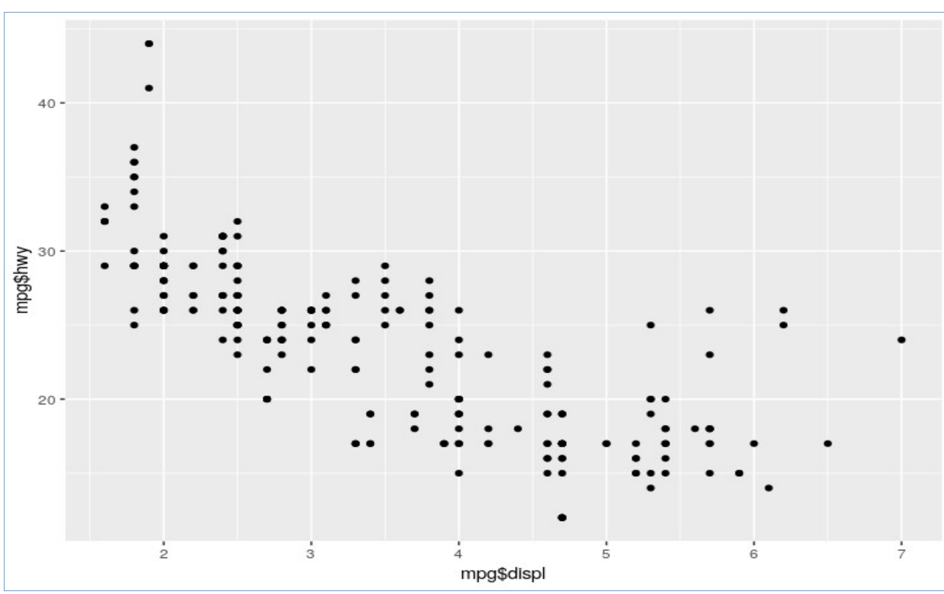



```
ggplot(data = mpg) +
  geom_point(mapping = aes(x = mpg$displ, y = mpg$hwy ))
```

Displacement (Car Weight) by MPG on the Highway

Is there more to learn from this data?

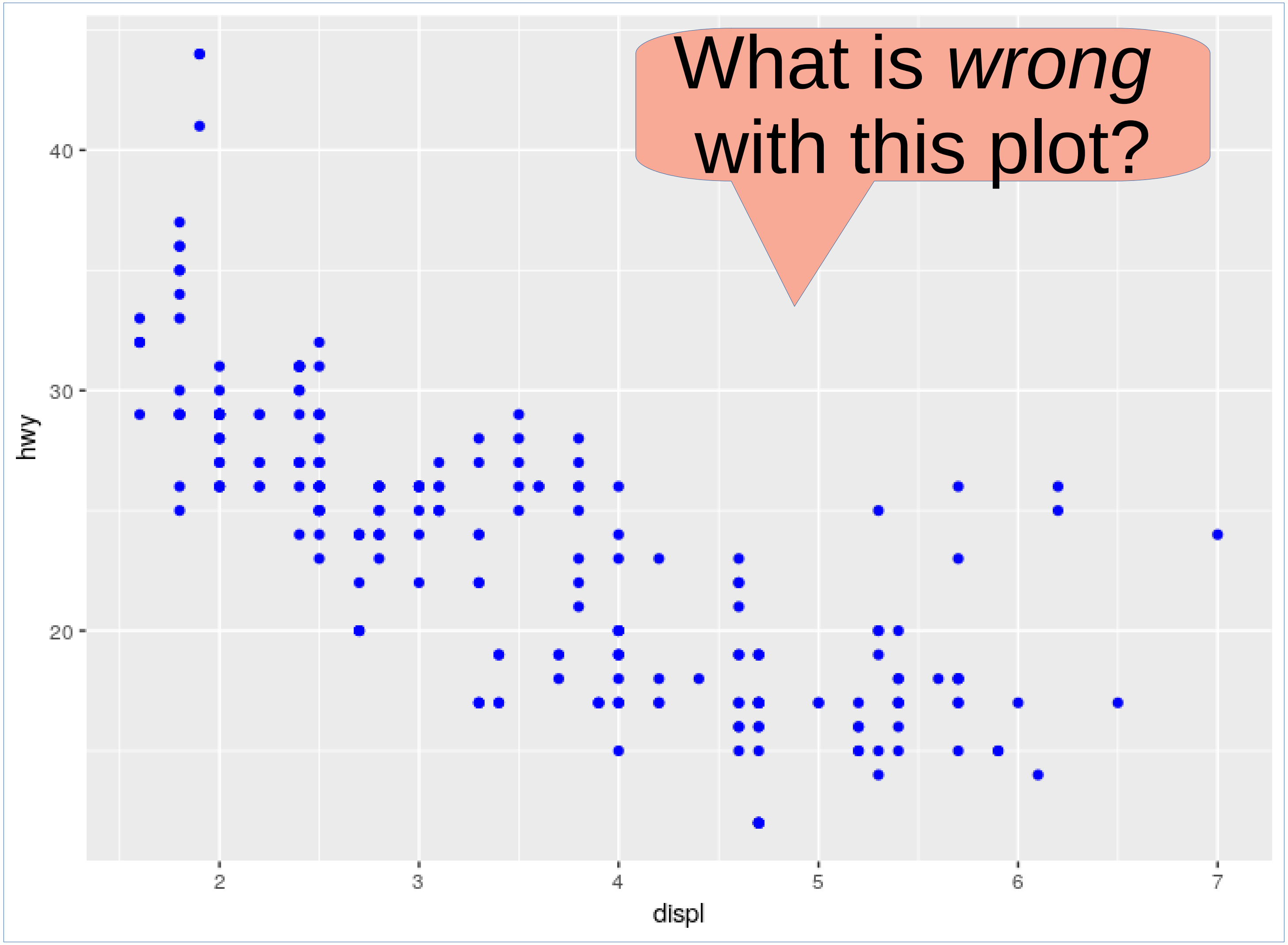
What is *wrong* with this the previous plot?



Yes!

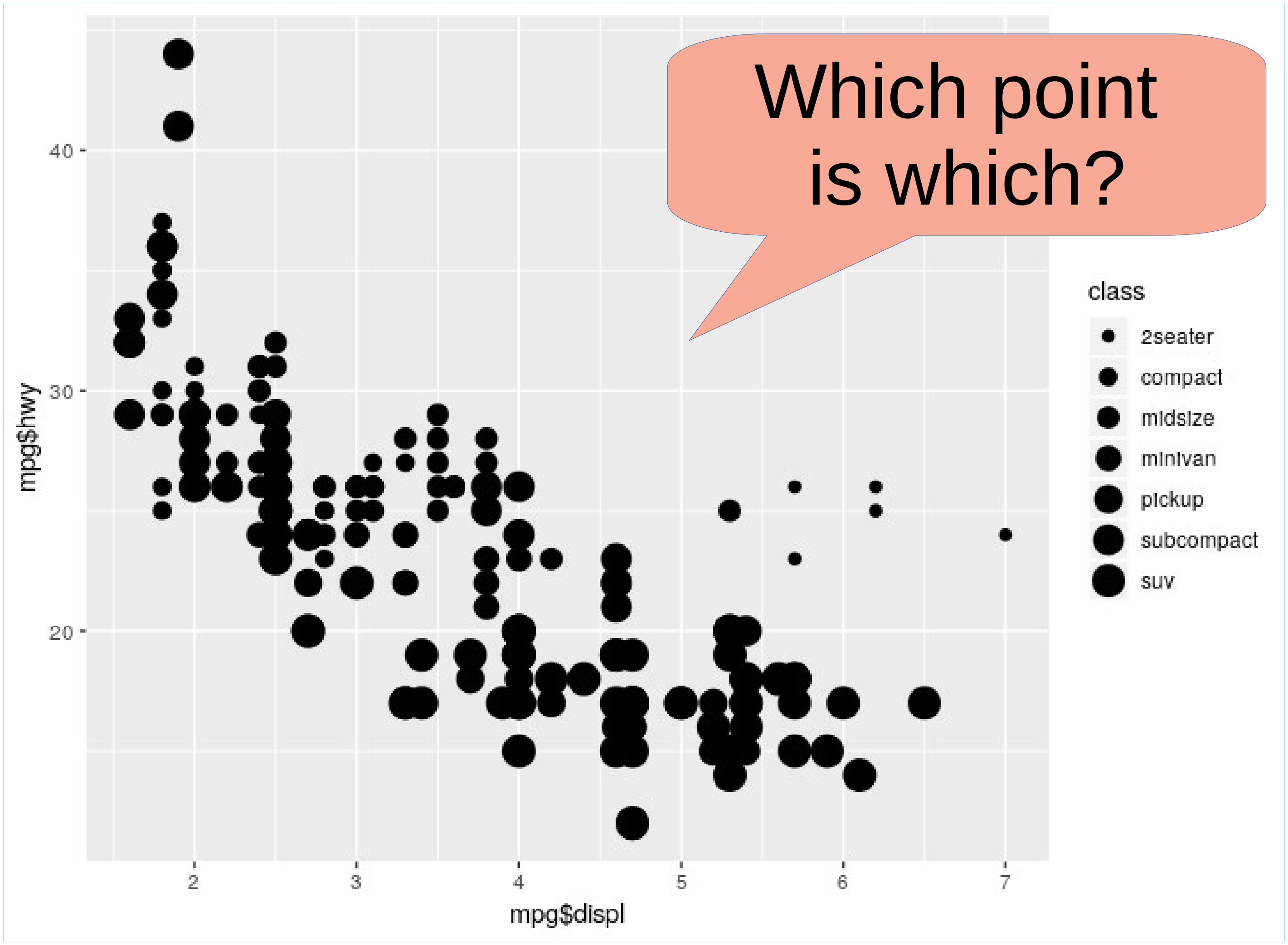
No??

New Blue Plot?




```
ggplot(data = mpg) + geom_point(mapping = aes(x = displ,
y = hwy), color = "blue")
```

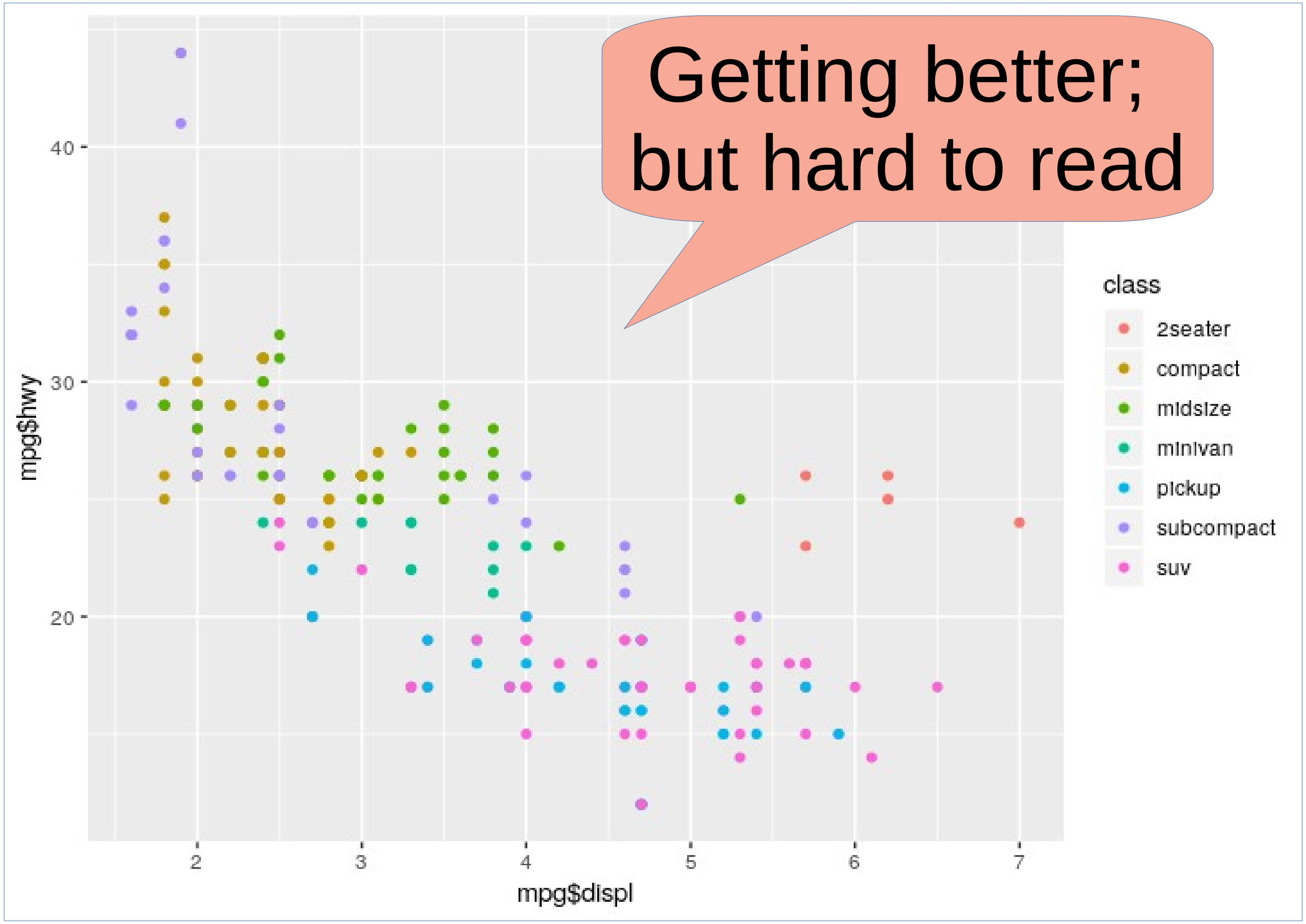
Try Sizing the points for Dimension



Which point is which?

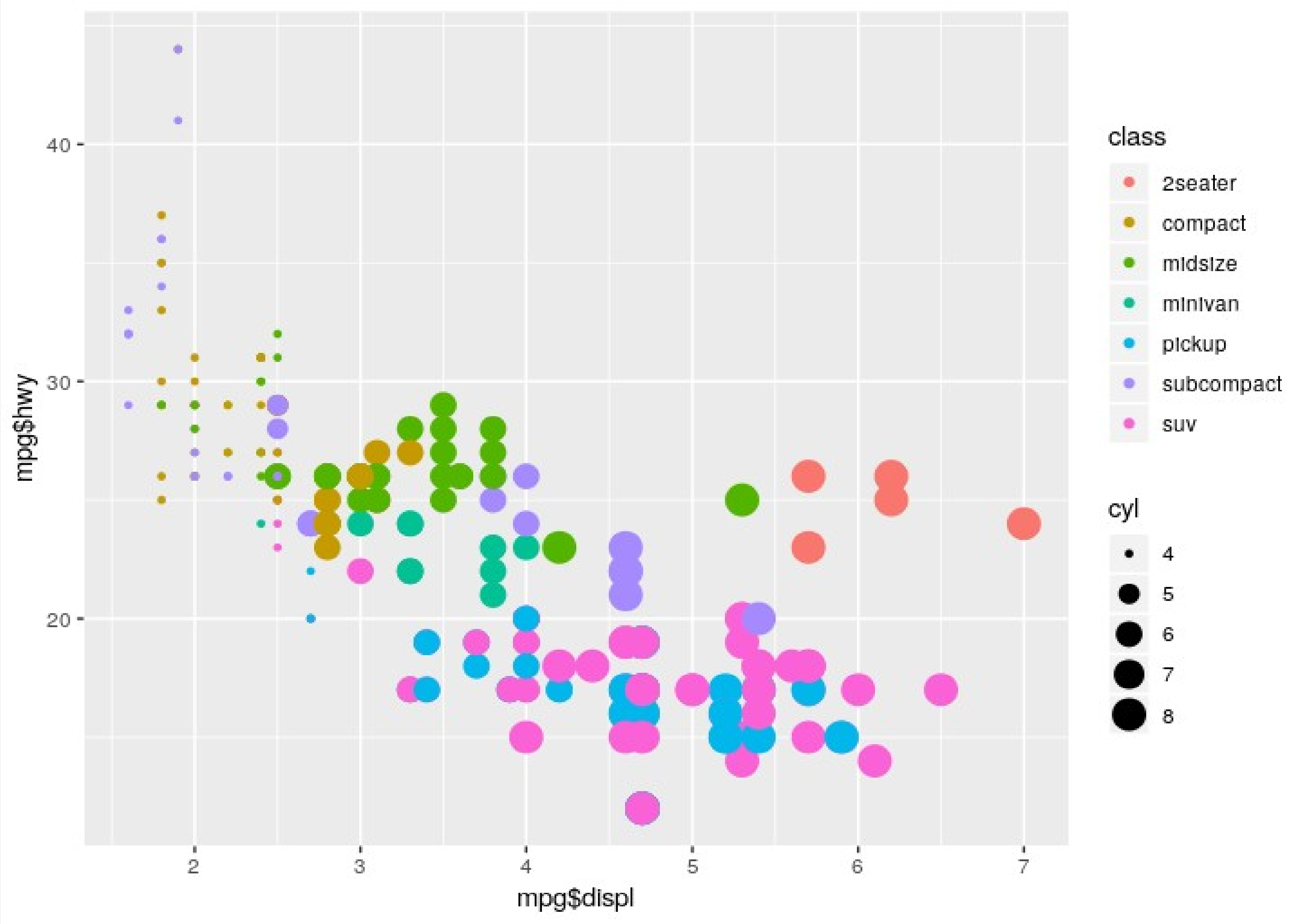
ggplot(data = mpg) + geom_point(mapping = aes(x = mpg$displ, y = mpg$hwy, size = class))

Try Coloring for Dimension



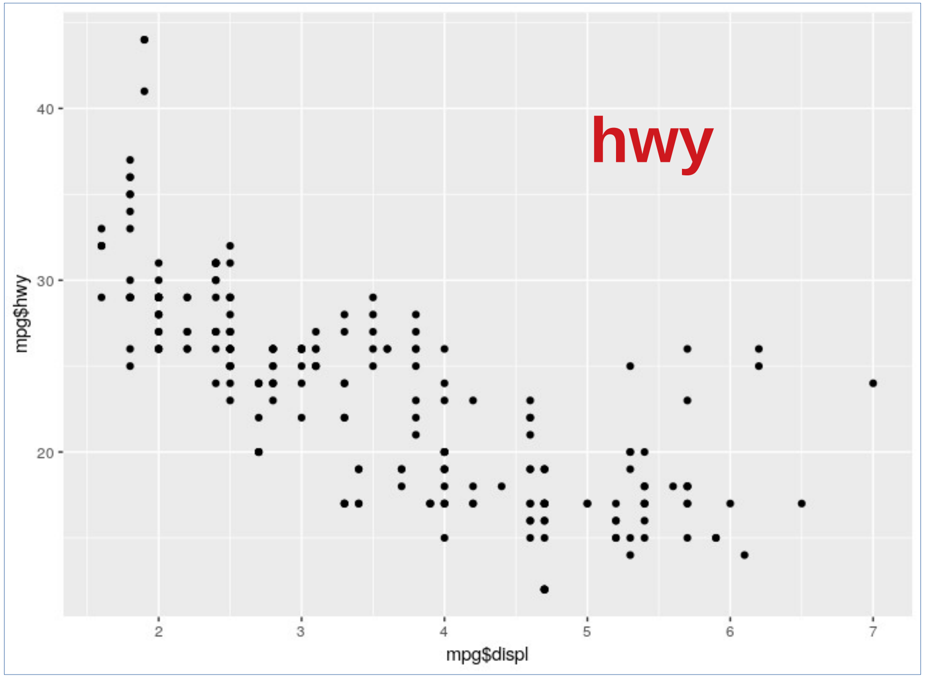
ggplot(data = mpg) + geom_point(mapping = aes(x = mpg$displ, y = mpg$hwy, color = class))

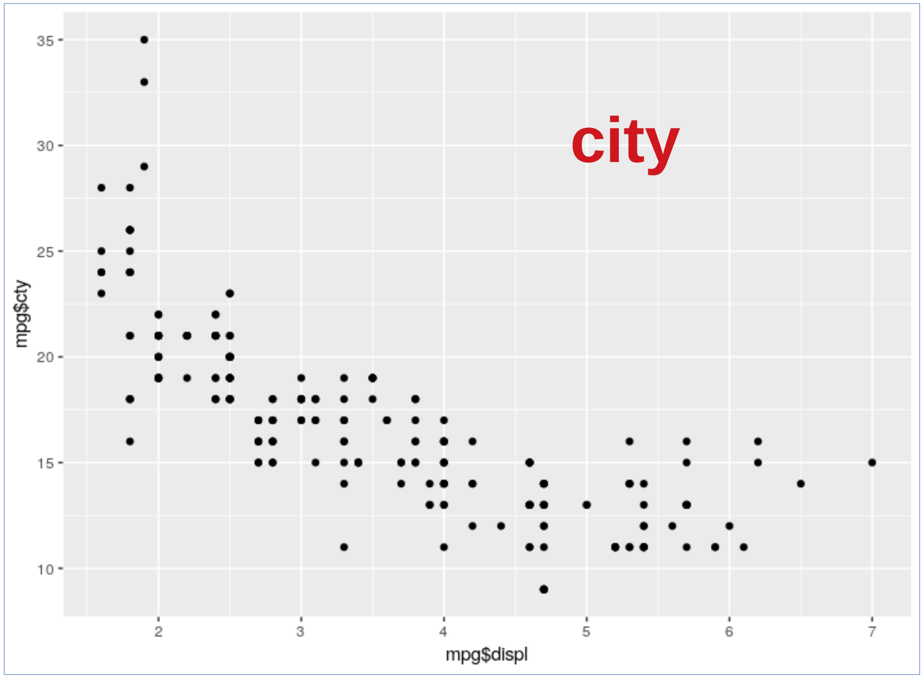
Combine Color, Sized Points and Cycle



ggplot(data = mpg) + geom_point(mapping = aes(x = mpg$displ, y = mpg$hwy, color = class, size = cyl))

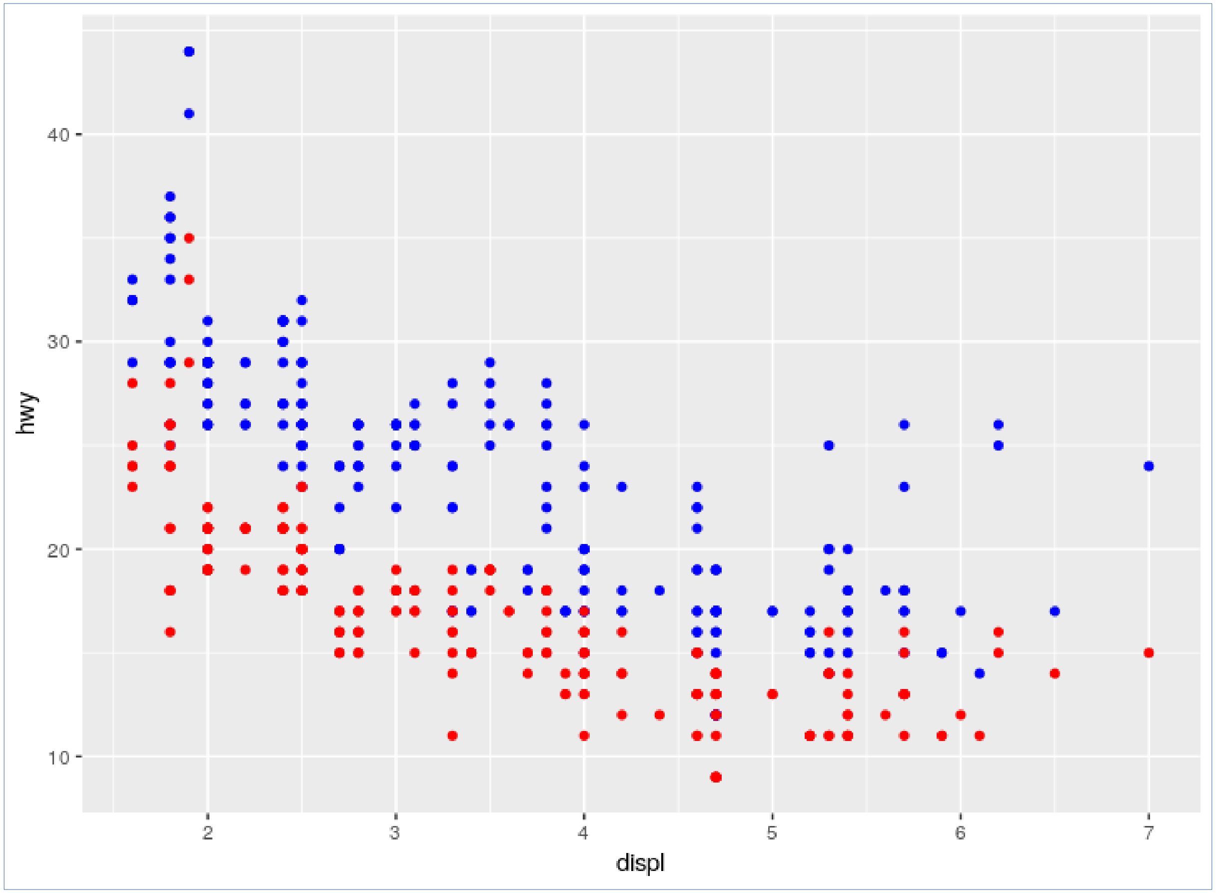
Comparing Hwy and City Mileage






```
# hwy mileage
ggplot(data = mpg) + geom_point(mapping = aes(x = mpg$displ, y
= mpg$hwy ))

# city mileage
ggplot(data = mpg) + geom_point(mapping = aes(x = mpg$displ, y
= mpg$cty ))
```

Comparing Hwy and City Mileage

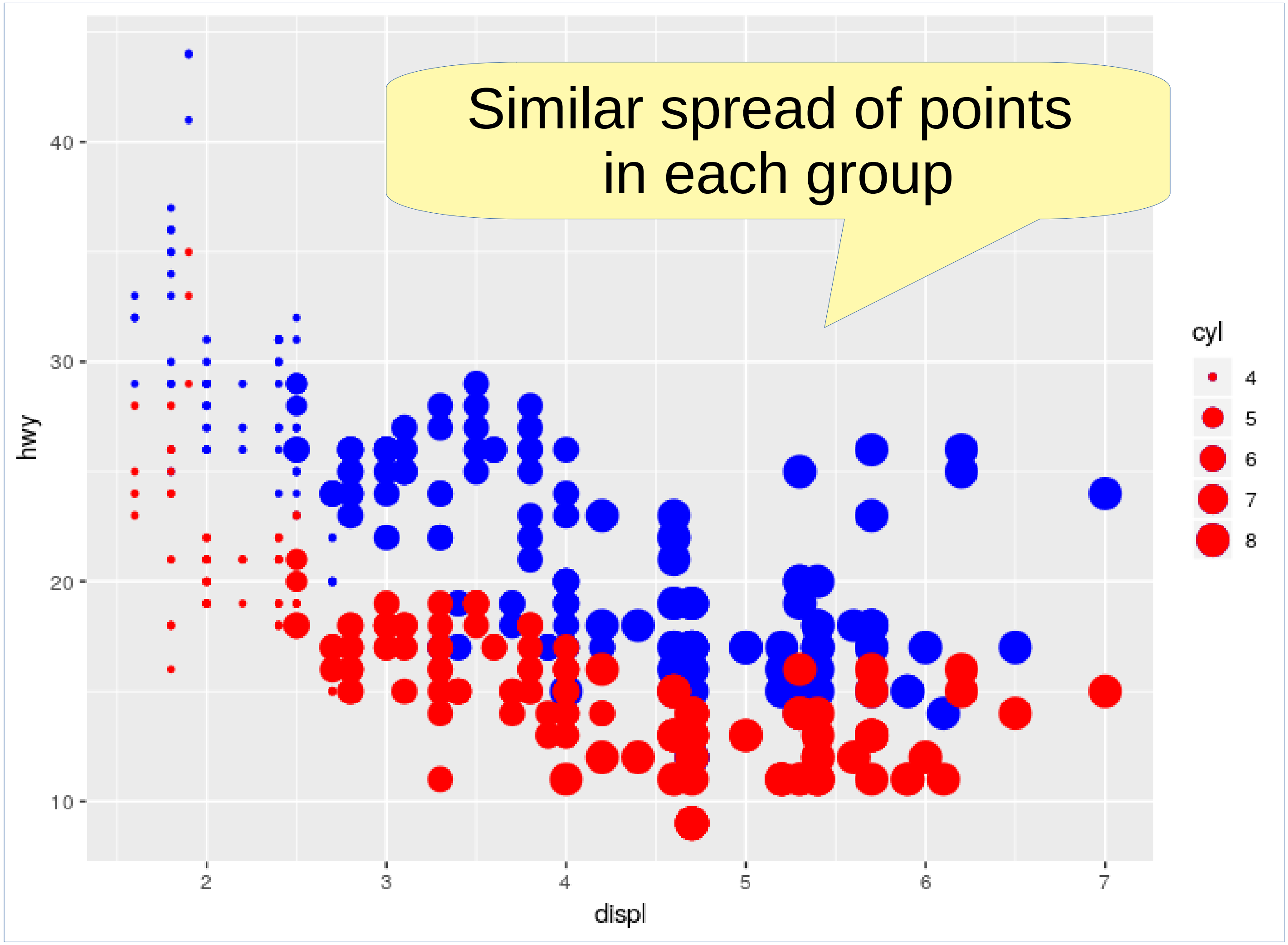



```
#incorporate hwy and cty mileage together in same plot
ggplot(data = mpg) + geom_point(mapping = aes(x = displ, y = hwy), color = "blue") + geom_point(mapping = aes(x = displ, y = cty), color="Red")
```

Add Sized Points

Hwy

City

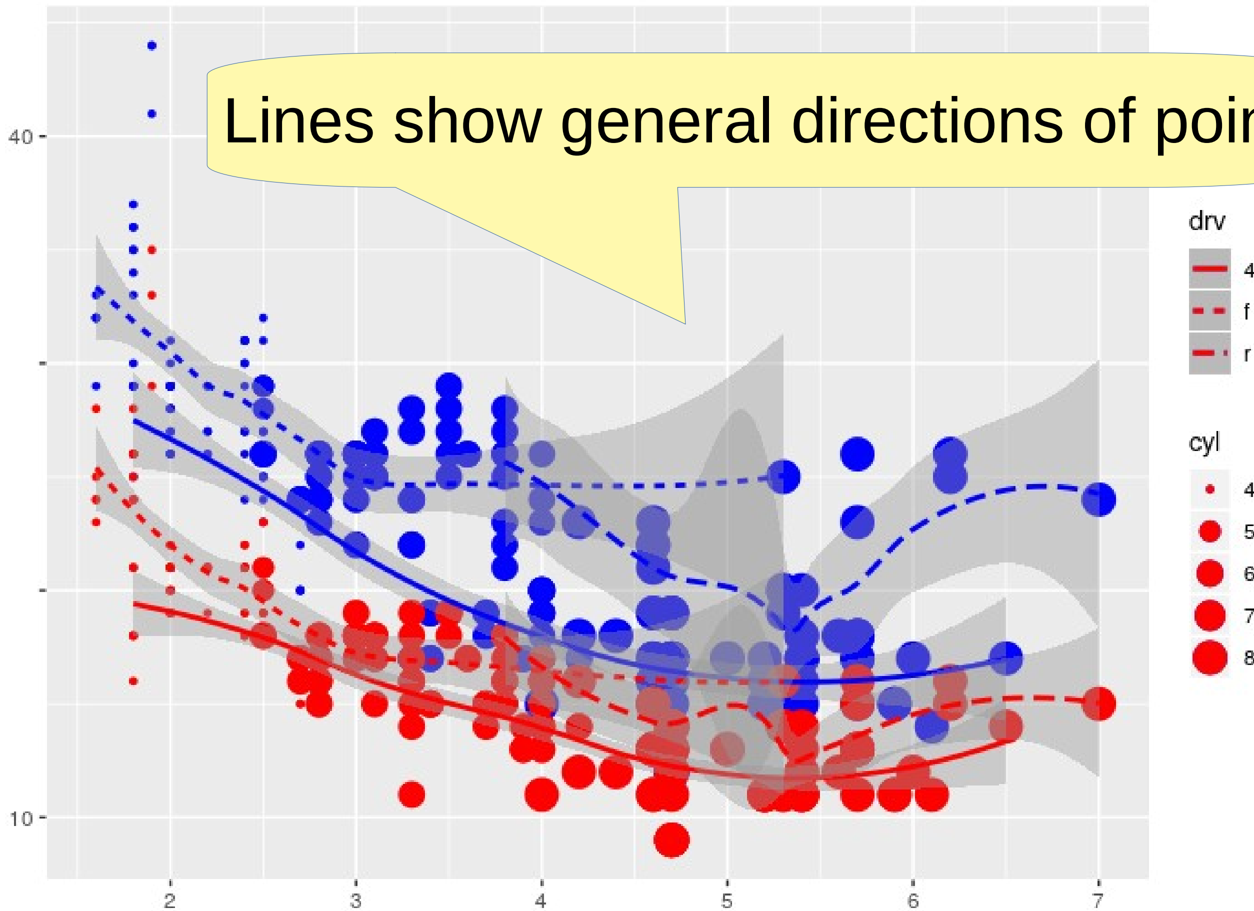


ggplot(data = mpg) + geom_point(mapping = aes(x = displ, y = hwy, size = cyl), color = "blue") + geom_point(mapping = aes(x = displ, y = cty, size = cyl), color="Red")

Add a *Smooth-Line*

Lines show general directions of points

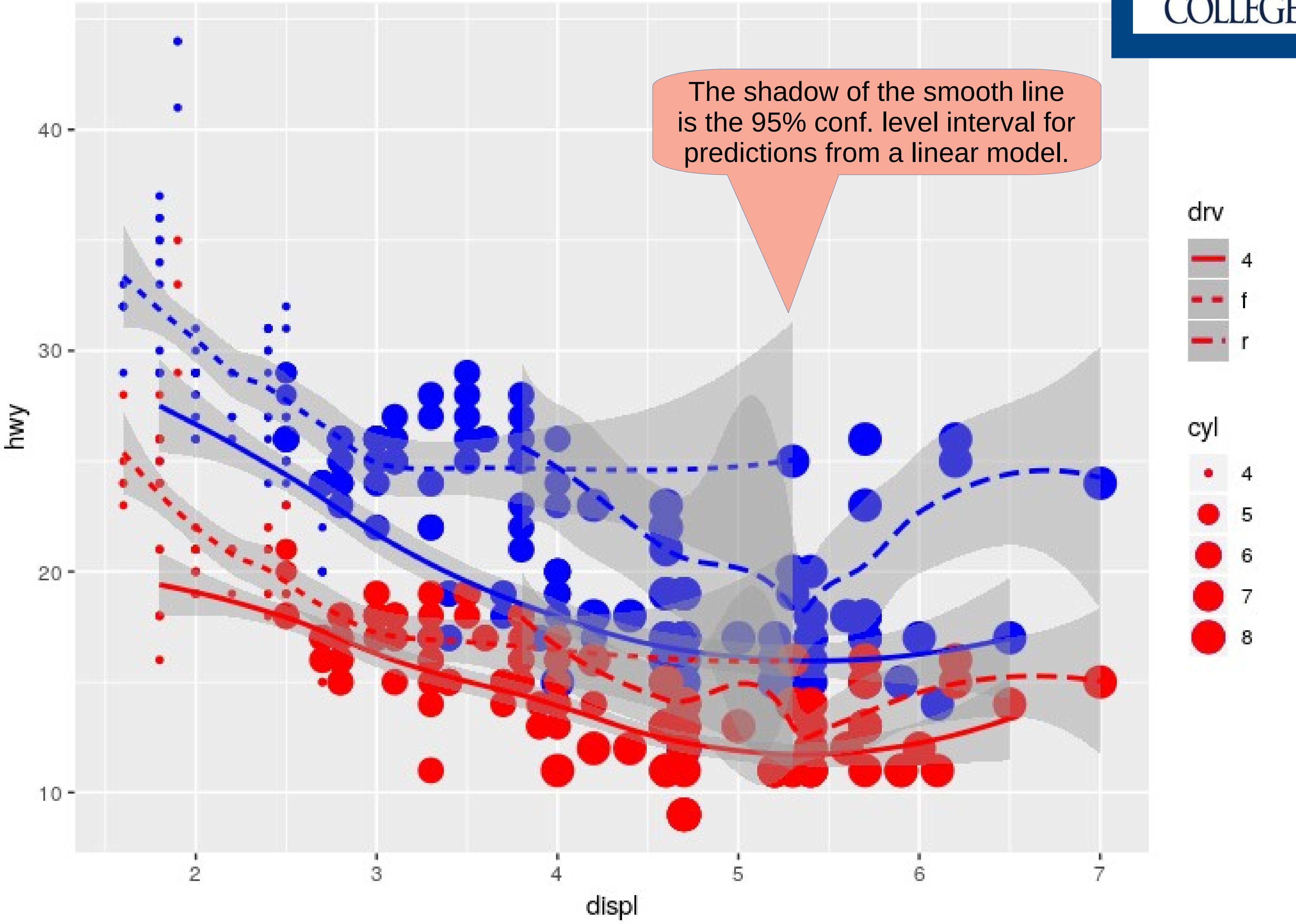
Hwy



City

```
ggplot(data = mpg) + geom_point(mapping = aes(x = displ, y = hwy,
size = cyl), color = "blue") + geom_point(mapping = aes(x = displ, y
= cty, size = cyl), color="Red") + geom_smooth(mapping = aes(x =
displ, y = hwy, linetype = drv), color = "blue") +
geom_smooth(mapping = aes(x = displ, y = cty, linetype = drv),
color = "red")
```

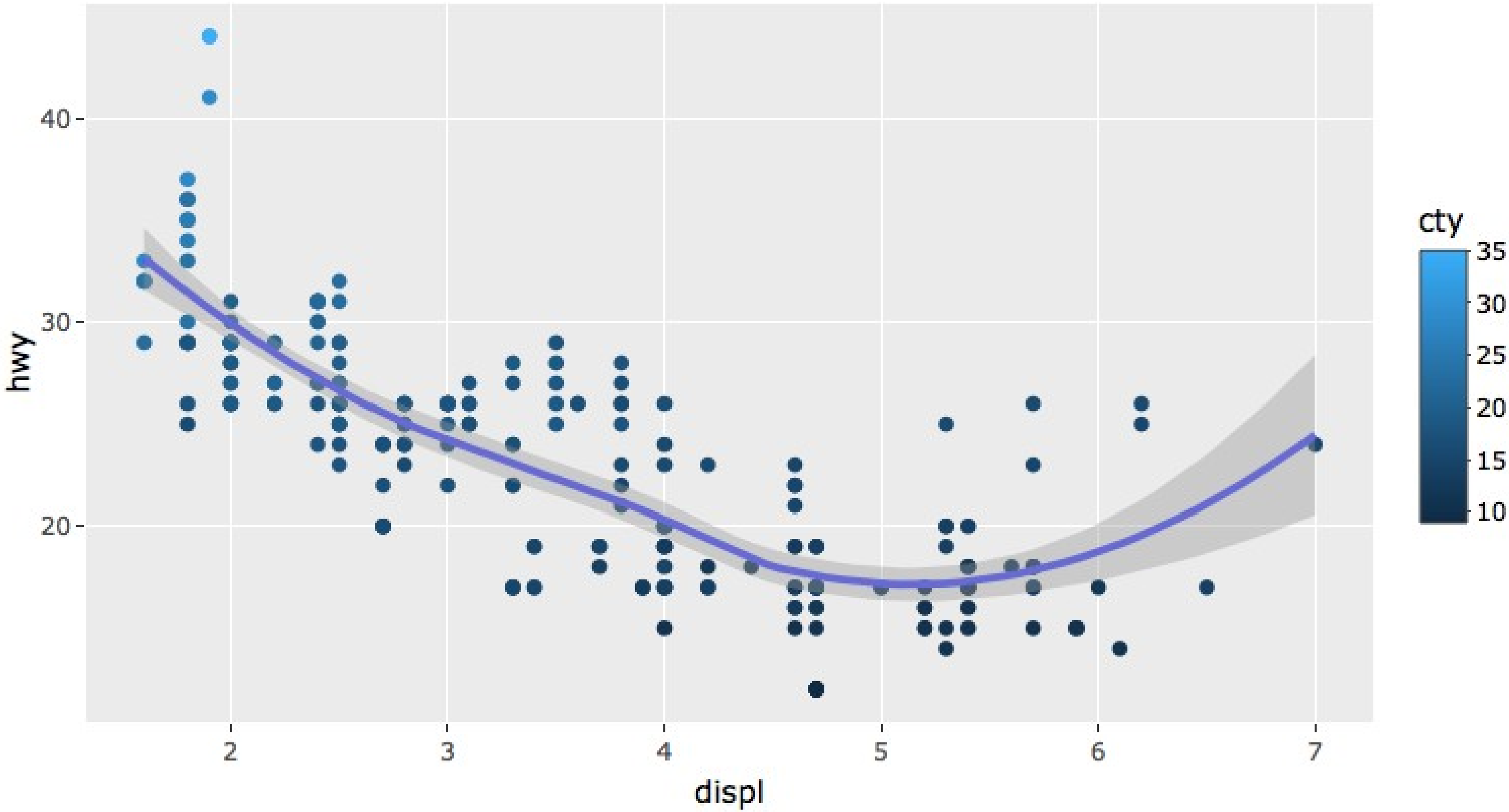
Bigger Image of the Previous Plot



Add Plotly For Interaction

```
# ref: https://plot.ly/ggplot2/stat_smooth/
#install.packages("plotly")

library(plotly)

p <- ggplot(mpg, aes(displ, hwy, color = cty))
p <- p + geom_point() + stat_smooth()

p <- ggplotly(p)

p
```

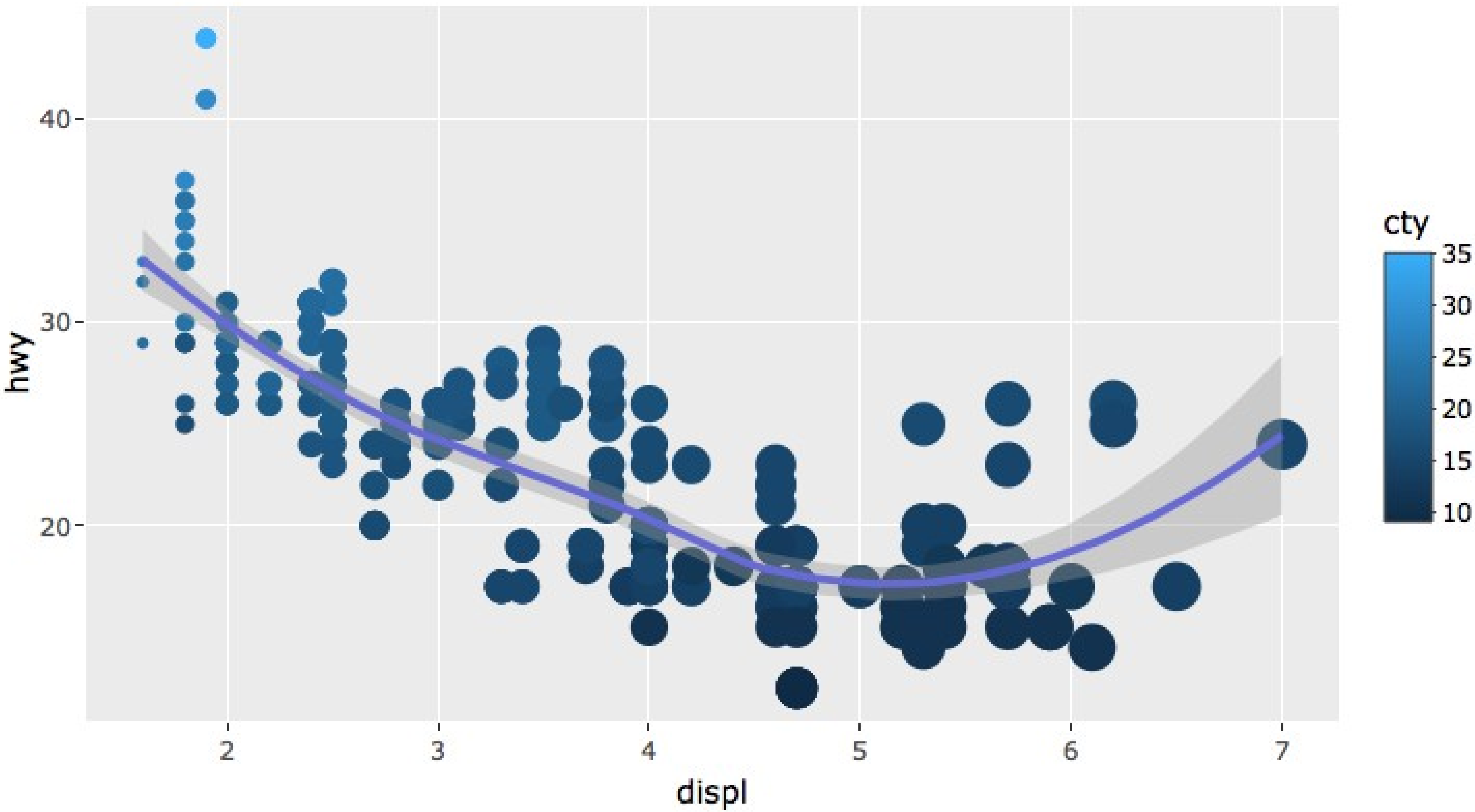
Add Plotly For Interaction



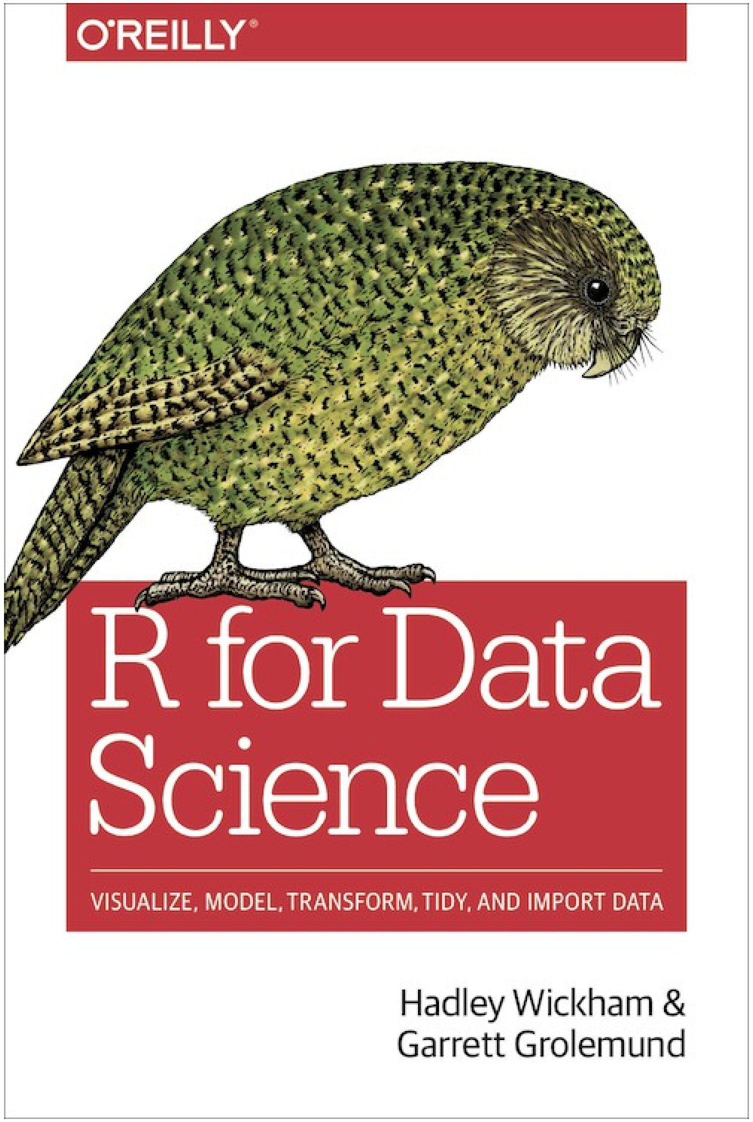
Add Plotly For Interaction

```
# ref: https://plot.ly/ggplot2/stat_smooth/
#install.packages("plotly")

library(plotly)

p <- ggplot(mpg, aes(displ, hwy, color = cty, size = displ))
p <- p + geom_point() + stat_smooth()

p <- ggplotly(p)

p
```

Add Plotly For Interaction



Where in the Web?
Where in the Book?



- Note the chapter differences!
- Book:
  - Chap 3: Data Transformation with dplyr
  - Pages 43 - 73
- Web:
  - Chap 5: Data Transformation with dplyr
  - http://r4ds.had.co.nz/transform.html

Transformation?

- What you want to show is in the data
- Unfortunately: To begin to show this is complicated.
  - Too much noise
  - Clutter
  - Unrelated pieces of data in the way

Filters

- Filters allow us to keep part of the whole while removing what we do not want

Filters to Transform Data?

Dictionary

transformation

trans·for·ma·tion

/ˌtran(t)sfərˈmāSH(ə)n/

noun

a thorough or dramatic change in form or appearance.
"its landscape has undergone a radical transformation"
synonyms: change, alteration, mutation, conversion, metamorphosis, transfiguration, transmutation, sea change; More

- a metamorphosis during the life cycle of an animal.
- PHYSICS
  the induced or spontaneous change of one element into another by a nuclear process.

Data Transformation

- Filter out the unwanted stuff to leave the "good" stuff
- Easier to work with and visualize
- **Data transformation:** the process of converting data or information from one format to another,
- Usually from the format of a source system into the required format of a new destination system.

Let the Transformation Begin!!

- # Install the library containing the data (if necessary)
  install.packages("nycflights13")
  library(nycflights13)
  library(tidyverse)

- # check that the data is found in the library
  nycflights13::flights
- View(flights)

What is the Data?

- # assign this data to an object.

  flights <- nycflights13::flights
- # View the table's columns

  names(nycflights13::flights)
- #Or, run,

  names(flights)

What does your data say about the types of questions that you can ask?!

Flight Data

flights

Filter

	year	month	day	dep_time	sched_dep_time	dep_delay	arr_time	sched_arr_time	arr_delay	carrier	flight	tailnum	origin	dest	air_time	distance	hour	minute	time_hour
1	2013	1	1	517	515	2	830	819	11	UA	1545	N14228	EWR	IAH	227	1400	5	15	2013-01-01 05:00
2	2013	1	1	533	529	4	850	830	20	UA	1714	N24211	LGA	IAH	227	1416	5	29	2013-01-01 05:00
3	2013	1	1	542	540	2	923	850	33	AA	1141	N619AA	JFK	MIA	160	1089	5	40	2013-01-01 05:00
4	2013	1	1	544	545	-1	1004	1022	-18	B6	725	N804JB	JFK	BQN	183	1576	5	45	2013-01-01 05:00
5	2013	1	1	554	600	-6	812	837	-25	DL	461	N668DN	LGA	ATL	116	762	6	0	2013-01-01 06:00
6	2013	1	1	554	558	-4	740	728	12	UA	1696	N39463	EWR	ORD	150	719	5	58	2013-01-01 05:00
7	2013	1	1	555	600	-5	913	854	19	B6	507	N516JB	EWR	FLL	158	1065	6	0	2013-01-01 06:00
8	2013	1	1	557	600	-3	709	723	-14	EV	5708	N829AS	LGA	IAD	53	229	6	0	2013-01-01 06:00
9	2013	1	1	557	600	-3	838	846	-8	B6	79	N593JB	JFK	MCO	140	944	6	0	2013-01-01 06:00
10	2013	1	1	558	600	-2	753	745	8	AA	301	N3ALAA	LGA	ORD	138	733	6	0	2013-01-01 06:00
11	2013	1	1	558	600	-2	849	851	-2	B6	49	N793JB	JFK	PBI	149	1028	6	0	2013-01-01 06:00
12	2013	1	1	558	600	-2	853	856	-3	B6	71	N657JB	JFK	TPA	158	1005	6	0	2013-01-01 06:00
13	2013	1	1	558	600	-2	924	917	7	UA	194	N29129	JFK	LAX	345	2475	6	0	2013-01-01 06:00

Showing 1 to 13 of 336,776 entries

```
> View(flights)
> names(nycflights13::flights)
 [1] "year"           "month"          "day"            "dep_time"       "sched_dep_time" "dep_delay"
 [7] "arr_time"       "sched_arr_time" "arr_delay"      "carrier"        "flight"         "tailnum"
[13] "origin"         "dest"           "air_time"       "distance"       "hour"           "minute"
[19] "time_hour"
```

Upon A Closer Inspection...

- This data frame contains all 336,776 flights that departed from New York City in 2013. The data comes from the US Bureau of Transportation Statistics, and is documented in ?flights.
- Flight numbers,
- Date, takeoff time and duration of flight
- Scheduled departure and arrival times
- Actual departure and arrival times (delays)
- Carrier
- Airports (origin and destination for a flight)
- Distance flown
- And more...

What are the Elements?

- #show whole dataset

  View(flights)

- # show first and second row of data table

  flights[1:2,] #flights[rows, cols]
- # show first and second cols
- flights[,1:2]
- # show cols 1 and 5 (using a vector)
- flights[,c(1,5)]

Data Types?

- #show the data types

  flights[1,]

```
> flights[1,]
# A tibble: 1 x 19
   year month   day dep_time sched_dep_time dep_delay arr_time sched_arr_time arr_delay
  <int> <int> <int>    <int>          <int>     <dbl>    <int>          <int>     <dbl>
1  2013     1     1      517            515         2      830            819        11
# ... with 10 more variables: carrier <chr>, flight <int>, tailnum <chr>, origin <chr>,
#   dest <chr>, air_time <dbl>, distance <dbl>, hour <dbl>, minute <dbl>, time_hour <dttm>
```

Why should we care about the data type?

Just My Type!

- **int** stands for integers.
- **dbl** stands for doubles, or real numbers.
- **chr** stands for character vectors, or strings.
- **dttm** stands for date-times (a date + a time).

- #others
- **lgl** stands for logical, vectors that contain only TRUE or FALSE.
- **fctr** stands for factors, which R uses to represent categorical variables with fixed possible values.
- **date** stands for dates.

dplyr Basics

- Five key dplyr functions
  - Pick observations by their values (**filter()**).
  - Reorder the rows (**arrange()**).
  - Pick variables by their names (**select()**).
  - Create new variables with functions of existing variables (**mutate()**).
  - Collapse many values down to a single summary (**summarise()**).
- Find help for each: ?keyword

Filter()

- #filter(object, column_header to consider)

  filter(flights, month == 1, day == 1)

  filter(flights, month == 1, dep_time == 554)

- #Assign a variable to this particular object

  dep_timeFlights554 <- filter(flights, month == 1, dep_time == 554)

- View(dep_timeFlights554)

Comparisons with Filter()

- R provides the standard suite: >, >=, <, <=, != (not equal), and == (equal).

- # select * from flights where month == 1;

  filter(flights, month == 1)

- #What happens here?

  filter(flights, month >=11)

  filter(flights, month <=11)

De Morgan's Law with Filter()

- #De Morgan's law: !(x & y) is the same as !x | !y, !(x | y) is the same as !x & !y.
- #For example, Use **OR** if you wanted to find flights that were not delayed (on arrival or departure) by more than two hours, you could use either of the following two filters:

  filter(flights, !(arr_delay > 120 | dep_delay > 120))

  filter(flights, arr_delay <= 120, dep_delay <= 120)

Arrange()

arrange() works similarly to filter() except that instead of selecting rows, it changes their order.

- #Show rows and cols as ordered by a particular column.
- #arrange(object, column_header)
- #What happens here?

arrange(flights, minute)

filter(flights, day == 30, dep_time == 554)

Arrange()

- #If you provide more than one column name, each additional column will be used to break ties in the values of preceding columns.

  arrange(flights, year, month, day)

- #Use desc() to re-order by a column in **descending** order.

  arrange(flights, desc(arr_delay))

  arrange(flights, arr_delay)

Select()

- #select() allows you to rapidly zoom in on a useful subset using names of the variables as parameters

  select(flights, year, month, day)

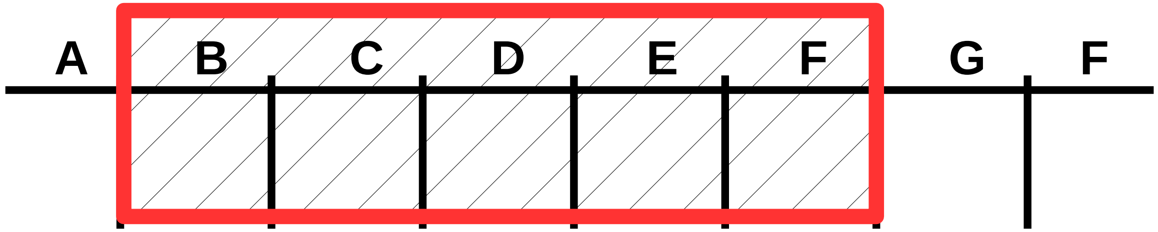
- # Select all columns going across the headers found between year and day (inclusive)

  select(flights, year:day)

- # Select all columns except those from year to day (inclusive)

  select(flights, -(year:day))

Selecting(data, A:F)



Mutate()

- #add new columns that are functions of existing columns
- #create a new object from flights having new cols.
- # xx and yy could be equations using existing data.
- xy <- mutate(flights,xx = day, yy = month)
- View(xy)

Summarise()

- Collapse your data into a single subset
- Use with group_by() to organize data into groups to help you see results from that time.

```
# A tibble: 365 x 4
# Groups:   year, month [?]
   year month   day  mean
  <int> <int> <int> <dbl>
1  2013     1     1 11.5
2  2013     1     2 13.9
3  2013     1     3 11.0
4  2013     1     4  8.95
5  2013     1     5  5.73
6  2013     1     6  7.15
7  2013     1     7  5.42
8  2013     1     8  2.55
```

```
by_day <- group_by(flights, year, month, day)
summarise(by_day, delay = mean(dep_delay, na.rm = TRUE))

# or, another way to enter the command using pipes...

flights %>%
  group_by(year, month, day) %>%
  summarise(mean = mean(dep_delay, na.rm = TRUE))
```

Lets Work With Data

- Load some libraries
  - `library(ggplot2)` or use `install.packages(ggplot2)`
- What data can we play with?
  - `data()`
- A good habit to define a variable for a data set

  `myData = diamonds`
  - `View(myData) # what do you see now?`

Diamonds in Data

	carat	cut	color	clarity	depth	table	price	x	y	z
1	0.23	Ideal	E	SI2	61.5	55.0	326	3.95	3.98	2.43
2	0.21	Premium	E	SI1	59.8	61.0	326	3.89	3.84	2.31
3	0.23	Good	E	VS1	56.9	65.0	327	4.05	4.07	2.31
4	0.29	Premium	I	VS2	62.4	58.0	334	4.20	4.23	2.63
5	0.31	Good	J	SI2	63.3	58.0	335	4.34	4.35	2.75
6	0.24	Very Good	J	VVS2	62.8	57.0	336	3.94	3.96	2.48
7	0.24	Very Good	I	VVS1	62.3	57.0	336	3.95	3.98	2.47
8	0.26	Very Good	H	SI1	61.9	55.0	337	4.07	4.11	2.53
9	0.22	Fair	E	VS2	65.1	61.0	337	3.87	3.78	2.49
10	0.23	Very Good	H	VS1	59.4	61.0	338	4.00	4.05	2.39

- What is this data?
- ? diamonds for help

Ask: Diamonds in Data

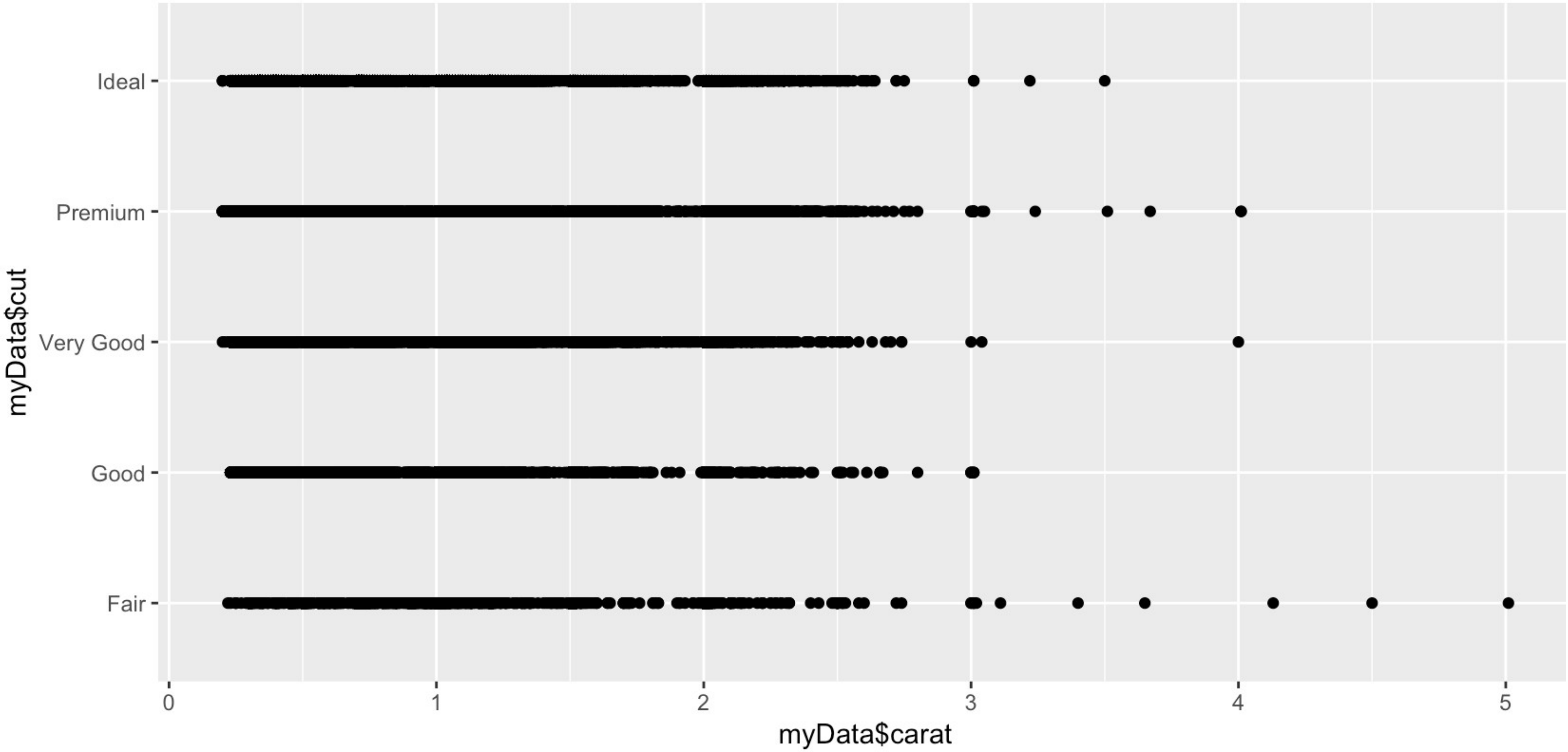
- Let’s ask about the carats and cuts relationship:
  - **carat**: weight of the diamond (0.2–5.01)
  - **cut**: quality of the cut (Fair, Good, Very Good, Premium, Ideal)

You can get some help in remembering the column names using the ‘$’ and TAB

```
ggplot(data = myData) +
geom_point(aes(x=myData$carat, y=myData$cut))
```

Ask: Diamonds in Data

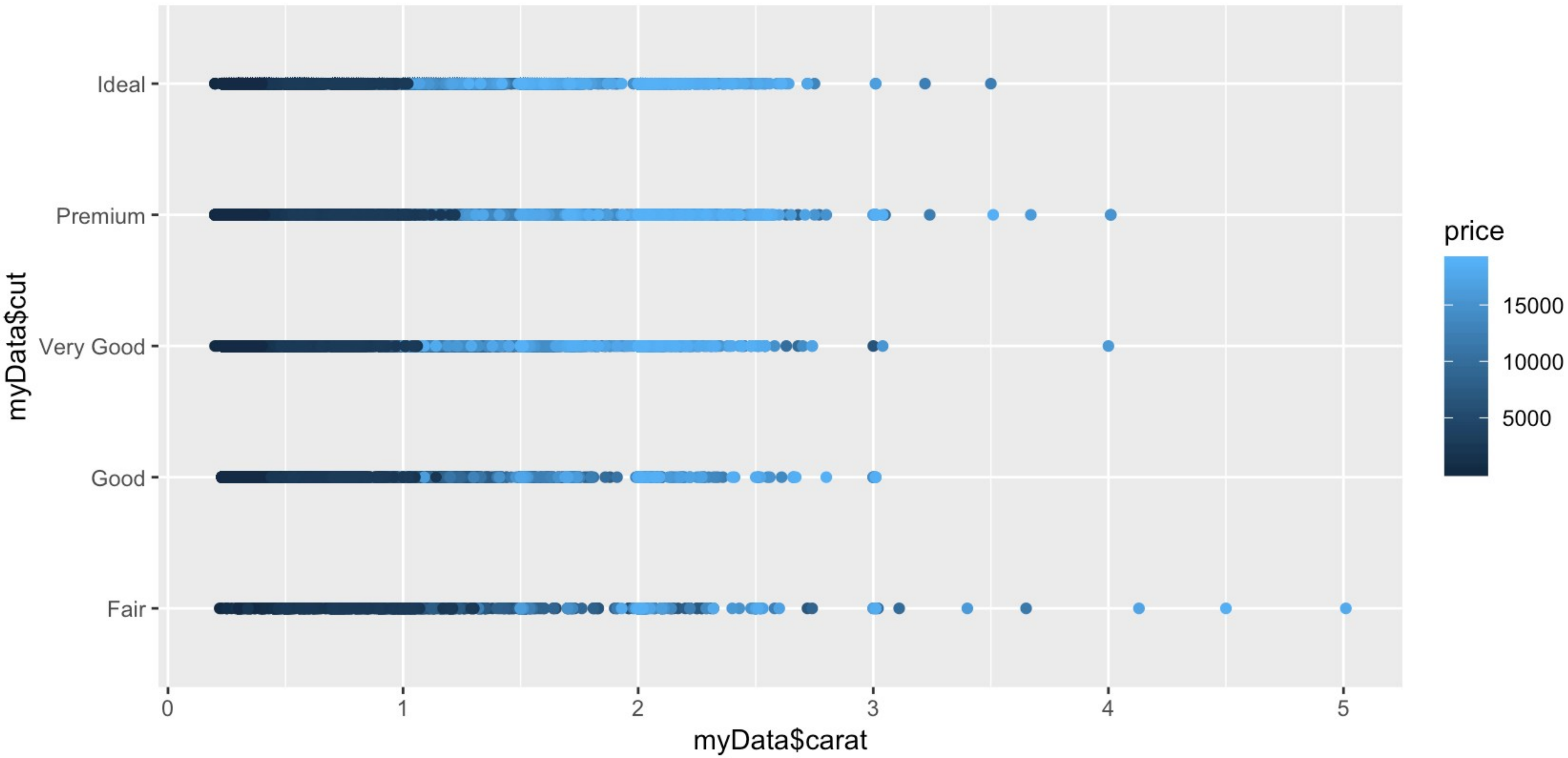
- ggplot(data = myData) + geom_point(aes(x=myData$carat, y=myData$cut))



Do I understand what this plot is telling me?

How Do The Prices Compare?

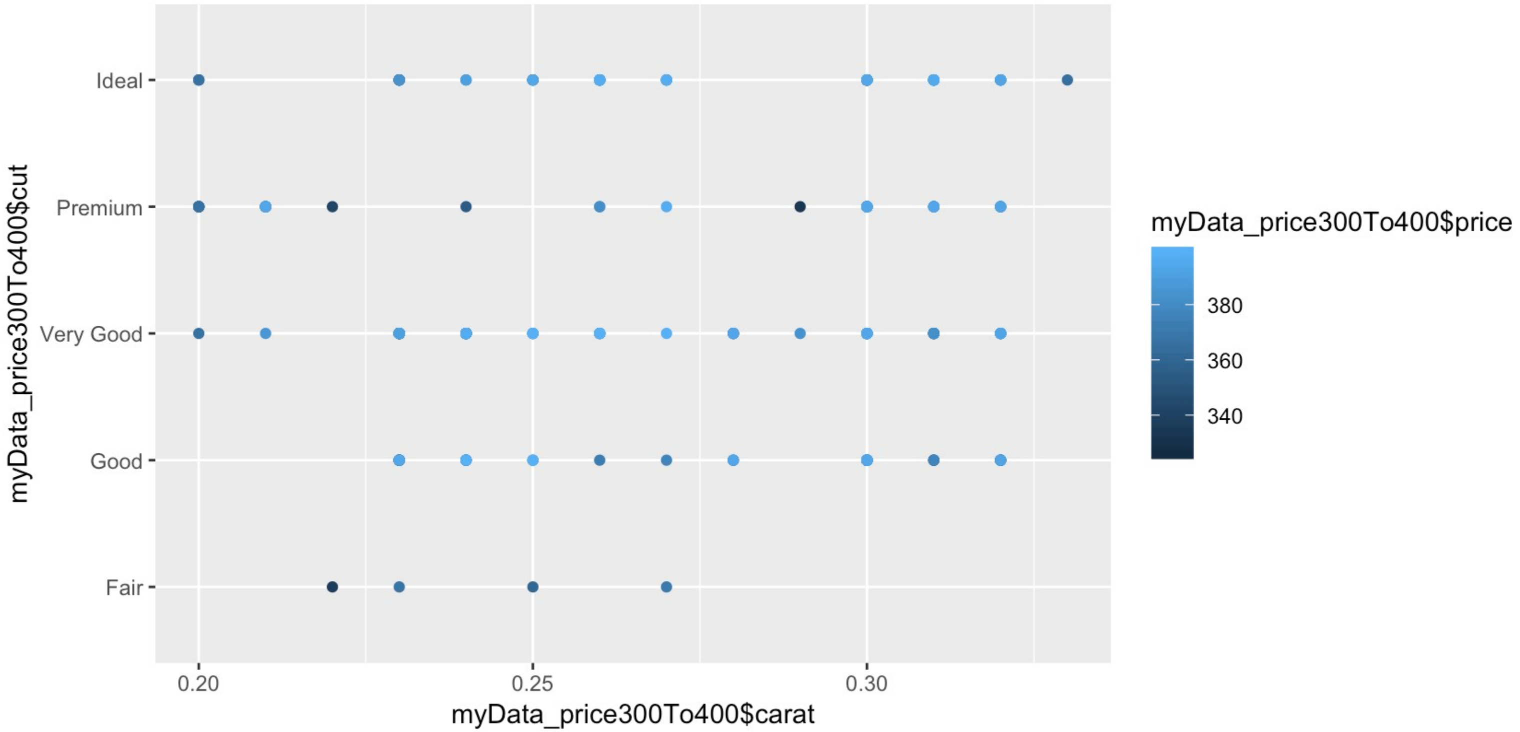
- ggplot(data = myData) + geom_point(aes(x=myData$carat, y=myData$cut, color = price))



Isolate Diamonds in Price Range: $300 to $400

```
myData_price300To400 <- filter(myData, price >= 300 & price < 400)
```

```
ggplot(data = myData_price300To400) +
geom_point(aes(x=myData_price300To400$carat,
y=myData_price300To400$cut, color =
myData_price300To400$price))
```



Convert Price
to Another Currency?

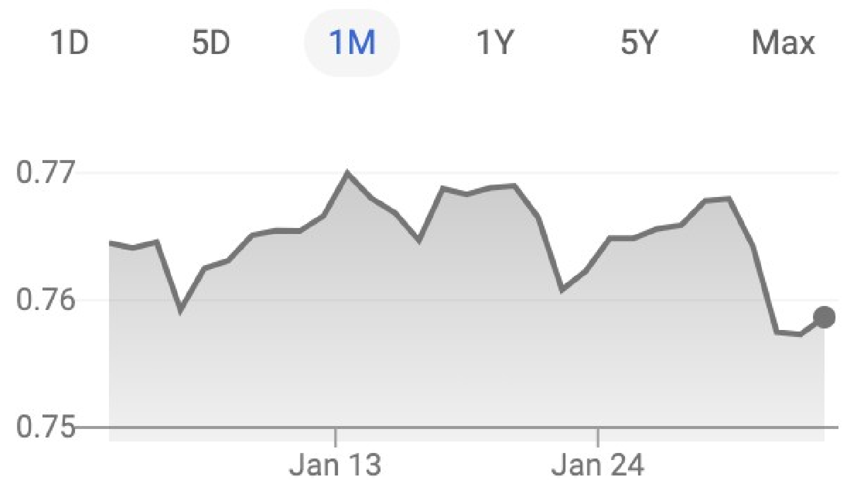
How about Pound Sterling?

1 United States Dollar equals

0.76 Pound sterling

Feb 3, 2:56 AM UTC · Disclaimer

1	United States Dolla
0.76	Pound sterling



Data provided by Morningstar for Currency and Coinbase for Cryptocurrency

Can we plot in pounds?!

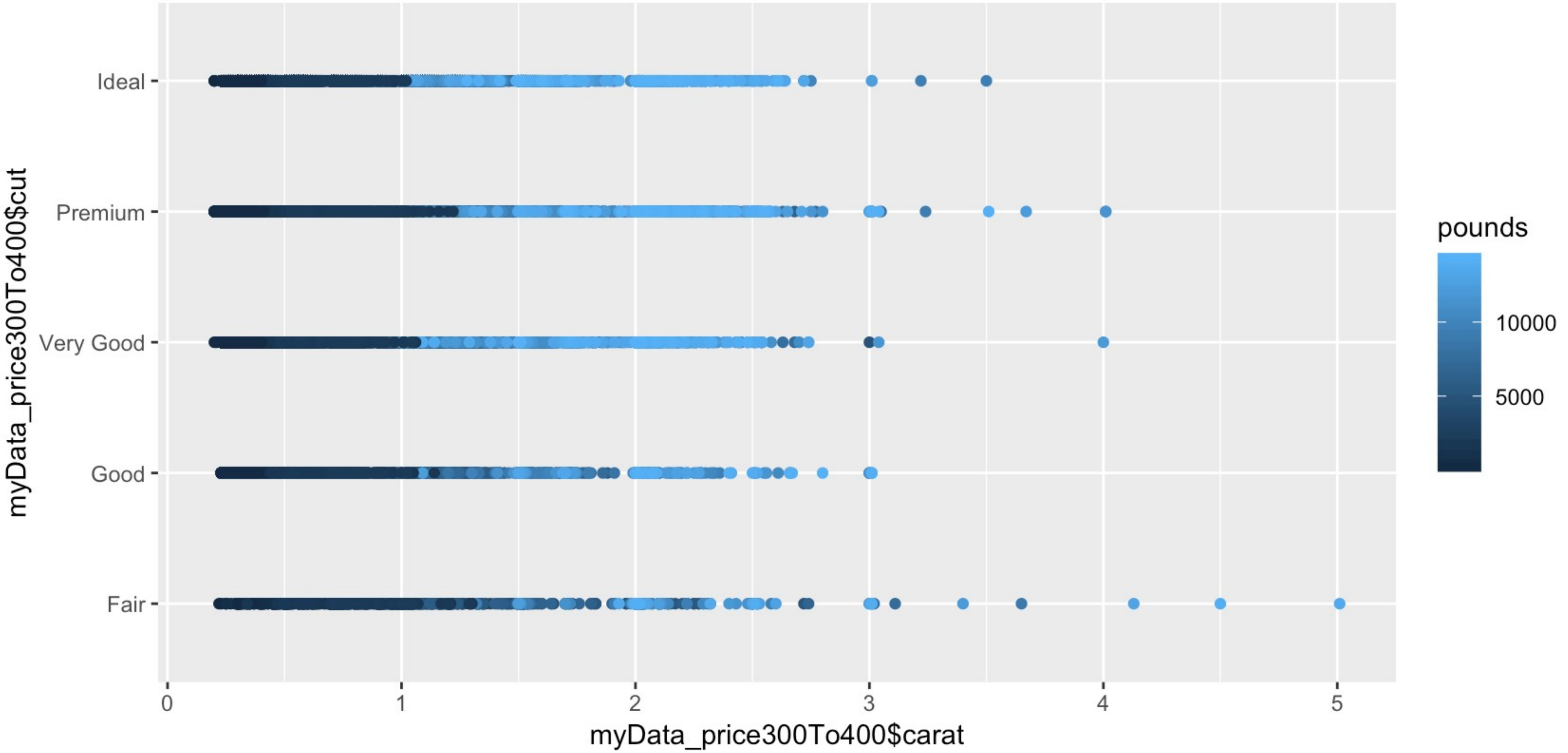
A New Header is Afoot

- Add a new column “pounds” to myData_price300To400

```
# create new col, overwrite old dataset
myData_price300To400 <- mutate(myData, pounds = price *
0.76)

# check data for new col
View(myData_price300To400)

# colourise by pounds sterling
ggplot(data = myData_price300To400) +
geom_point(aes(x=myData_price300To400$carat,
y=myData_price300To400$cut, color = pounds))
```

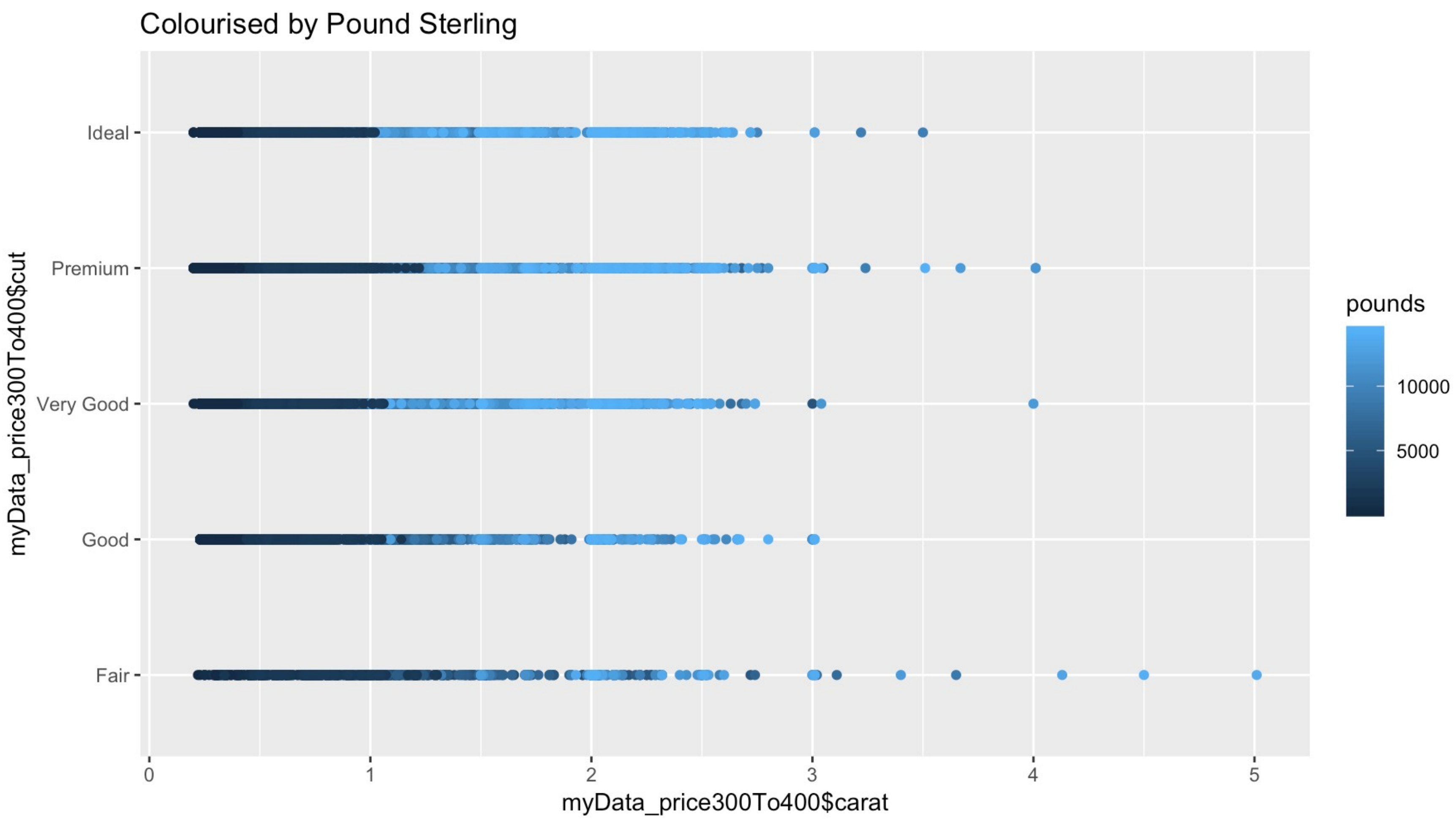
Add Pounds Currency

```
ggplot(data = myData_price300To400) +
geom_point(aes(x=myData_price300To400$carat,
y=myData_price300To400$cut, color = pounds))
```



Add a Title to the Plot

```
ggplot(data = myData_price300To400) +
geom_point(aes(x=myData_price300To400$carat,
y=myData_price300To400$cut, color = pounds)) + ggtitle("Colourised by Pound
Sterling")
```



Try Out On Another Dataset?

- **iris** data set gives the measurements in centimeters of the variables sepal length, sepal width, petal length and petal width, respectively, for 50 flowers from each of 3 species of iris. The species are Iris setosa, versicolor, and virginica.
- **ToothGrowth** data set contains the result from an experiment studying the effect of vitamin C on tooth growth in 60 Guinea pigs. Each animal received one of three dose levels of vitamin C (0.5, 1, and 2 mg/day) by one of two delivery methods, (orange juice or ascorbic acid (a form of vitamin C and coded as VC).
- **PlantGrowth**: Results obtained from an experiment to compare yields (as measured by dried weight of plants) obtained under a control and two different treatment condition.
- **USArrests**: This data set contains statistics about violent crime rates by us state.
- *Data() # to see more sets in R*