Programming for Data Analytics

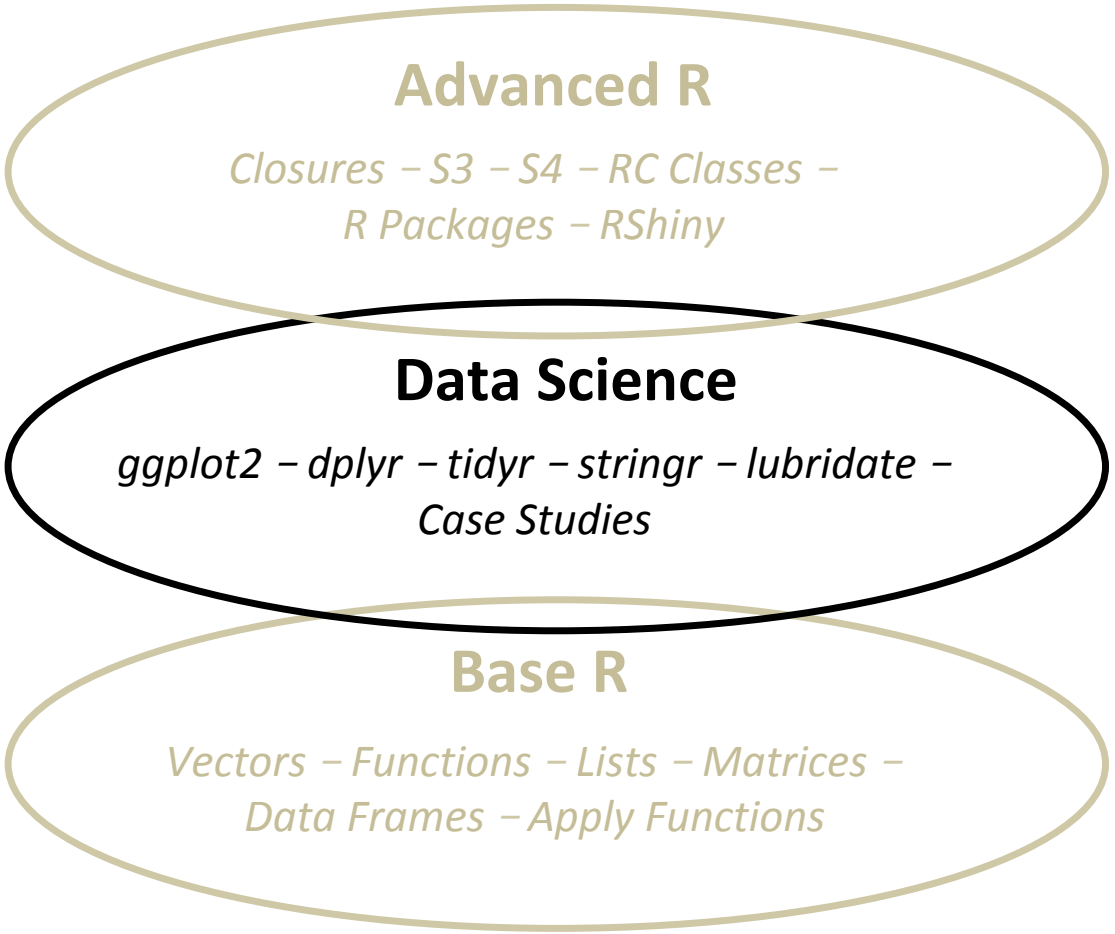
5. ggplot2

Dr. Jim Duggan,
School of Engineering & Informatics
National University of Ireland Galway.

https://github.com/JimDuggan/CT5102

Lecture Overview

- Tibble
- Data Exploration
- Aesthetic Mappings
- Common Problems
- Facets
- Geometric Objects
- Statistical Transformations
- Layered Grammar of Graphics

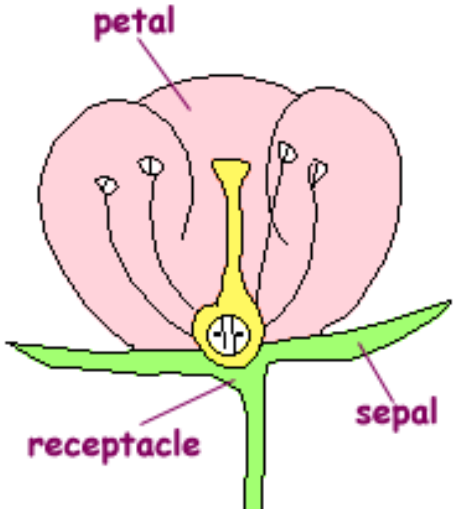


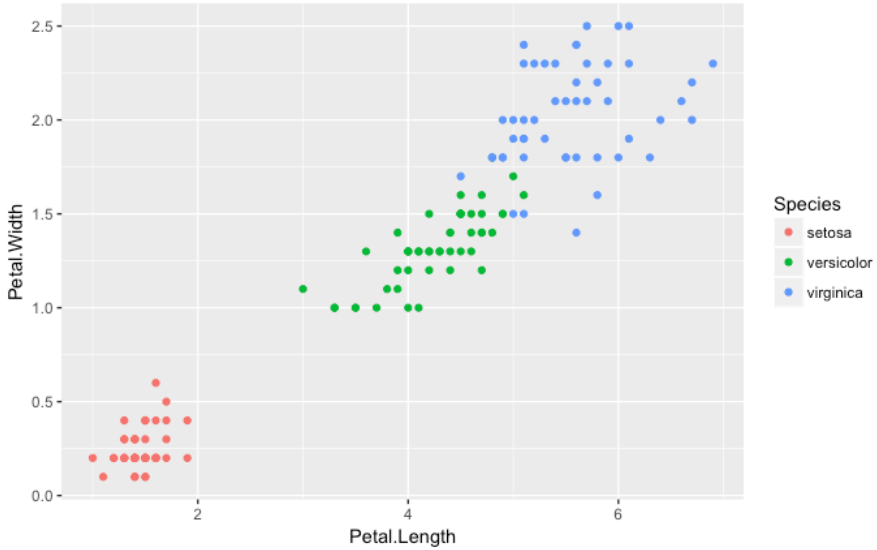
(1) The tibble

- "Tibbles are data frames, but they tweak some older behaviours to make life a little easier"
- One of the unifying features of the **tidyverse**
- To coerce a data frame to a tibble, use as_tibble()
- A tibble can be created from individual vectors using tibble()

```
> as_tibble(iris)
# A tibble: 150 × 5
   Sepal.Length Sepal.Width Petal.Length Petal.Width
          <dbl>       <dbl>        <dbl>       <dbl>
1           5.1         3.5          1.4         0.2
2           4.9         3.0          1.4         0.2
3           4.7         3.2          1.3         0.2
4           4.6         3.1          1.5         0.2
5           5.0         3.6          1.4         0.2
6           5.4         3.9          1.7         0.4
7           4.6         3.4          1.4         0.3
8           5.0         3.4          1.5         0.2
9           4.4         2.9          1.4         0.2
10          4.9         3.1          1.5         0.1
# ... with 140 more rows, and 1 more variables:
#   Species <fctr>
```

Data Set: Iris Data Set





Iris flower data set

From Wikipedia, the free encyclopedia

The ***Iris* flower data set** or **Fisher's *Iris* data set** is a multivariate data set introduced by the British statistician and biologist Ronald Fisher in his 1936 paper *The use of multiple measurements in taxonomic problems* as an example of linear discriminant analysis.[1] It is sometimes called **Anderson's *Iris* data set** because Edgar Anderson collected the data to quantify the morphologic variation of *Iris* flowers of three related species.[2] Two of the three species were collected in the Gaspé Peninsula "all from the same pasture, and picked on the same day and measured at the same time by the same person with the same apparatus".[3]

The data set consists of 50 samples from each of three species of *Iris* (*Iris setosa*, *Iris virginica* and *Iris versicolor*). Four features were measured from each sample: the length and the width of the sepals and petals, in centimetres. Based on the combination of these four features, Fisher developed a linear discriminant model to distinguish the species from each other.

tibble v data.frame - Printing

- Printing: tibbles have a refined print method that only shows the first 10 rows, and all columns that fit the screen
- Each column also reports its type

```
> as_tibble(iris)
# A tibble: 150 × 5
   Sepal.Length Sepal.Width
          <dbl>       <dbl>
1           5.1         3.5
2           4.9         3.0
3           4.7         3.2
4           4.6         3.1
5           5.0         3.6
6           5.4         3.9
7           4.6         3.4
8           5.0         3.4
9           4.4         2.9
10          4.9         3.1
# ... with 140 more rows, and 3 more
#   variables: Petal.Length <dbl>,
#   Petal.Width <dbl>, Species <fctr>
```

tibble v data.frame - Subsetting

- [], $ and [[work similar to a data frame
- Partial matching not supported

```
> t <- slice(as_tibble(iris),1:5)
>
> t
# A tibble: 5 × 5
  Sepal.Length Sepal.Width Petal.Length Petal.Width Species
         <dbl>       <dbl>        <dbl>       <dbl>  <fctr>
1          5.1         3.5          1.4         0.2  setosa
2          4.9         3.0          1.4         0.2  setosa
3          4.7         3.2          1.3         0.2  setosa
4          4.6         3.1          1.5         0.2  setosa
5          5.0         3.6          1.4         0.2  setosa
>
> t$Sepal.Length
[1] 5.1 4.9 4.7 4.6 5.0
>
> t[["Sepal.Length"]]
[1] 5.1 4.9 4.7 4.6 5.0
>
> t$Sepal.Lengt
NULL
Warning message:
Unknown column 'Sepal.Lengt'
```

tibble abbreviations

Abbreviation	Data Type
int	integers
dbl	doubles (real numbers)
chr	character vectors (strings
dttm	date-times
lgl	logical
fctr	factor (categorical variables with fixed possible values)
date	dates

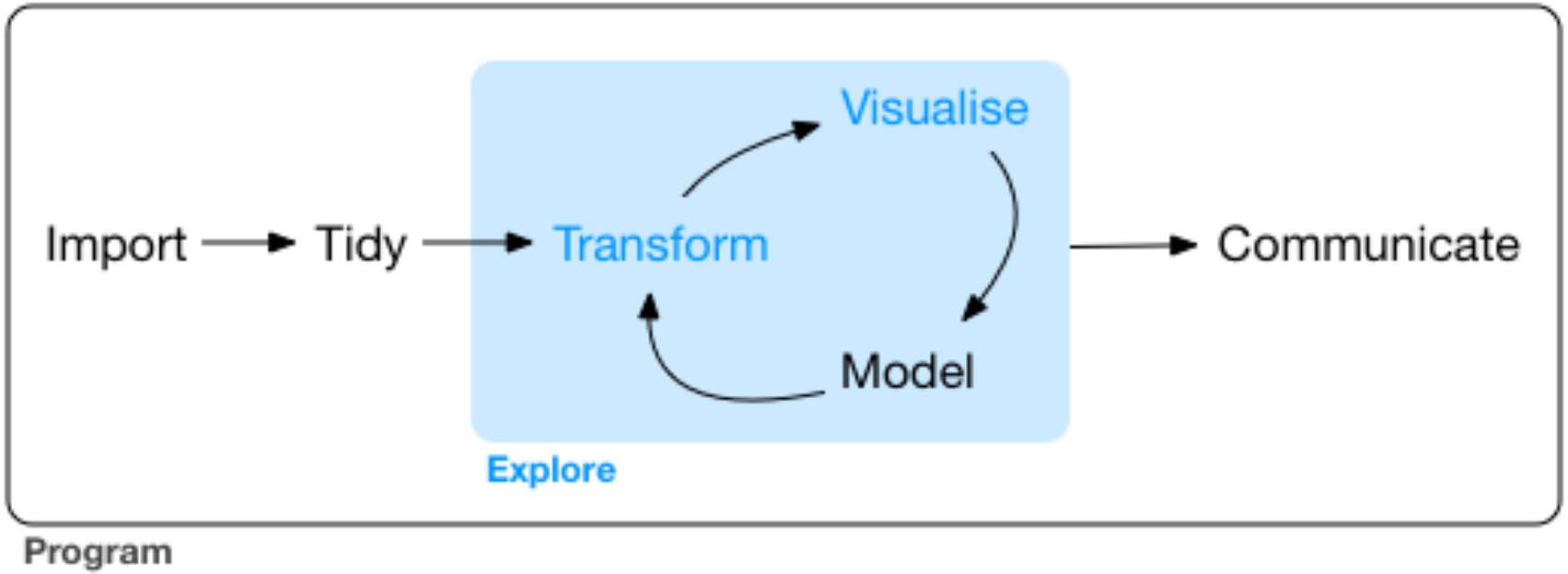
Challenge 5.1
(p124 Wickham & Grolemund)

- Compare and contrast the following operations on a data.frame and equivalent tibble. What is different? Why might the default data frame behaviour cause you frustration?

```
df <- data.frame(abc=1, xyz="a")
df$x
df[, "xyz"]
df[, c("abc", "xyz")]
```

(2) Data Exploration

"Data exploration is the art of looking at your data, rapidly generating hypotheses, quickly testing them, then repeating again and again and again." (Wickham and Grolemund 2017).



Data Visualisation with **ggplot2**

"The simple graph has brought more information to the data analyst's mind that any other device." – John Tukey

```
> dt <- ggplot2::mpg
>
> dt
# A tibble: 234 × 11
   manufacturer      model displ  year   cyl      trans   drv   cty   hwy    fl   class
          <chr>      <chr> <dbl> <int> <int>      <chr> <chr> <int> <int> <chr>   <chr>
1          audi         a4   1.8  1999     4   auto(l5)     f    18    29     p compact
2          audi         a4   1.8  1999     4 manual(m5)     f    21    29     p compact
3          audi         a4   2.0  2008     4 manual(m6)     f    20    31     p compact
4          audi         a4   2.0  2008     4   auto(av)     f    21    30     p compact
5          audi         a4   2.8  1999     6   auto(l5)     f    16    26     p compact
6          audi         a4   2.8  1999     6 manual(m5)     f    18    26     p compact
7          audi         a4   3.1  2008     6   auto(av)     f    18    27     p compact
8          audi a4 quattro   1.8  1999     4 manual(m5)     4    18    26     p compact
9          audi a4 quattro   1.8  1999     4   auto(l5)     4    16    25     p compact
10         audi a4 quattro   2.0  2008     4 manual(m6)     4    20    28     p compact
# ... with 224 more rows
```

Fuel Economy Data Set (ggplot2::mpg)

This dataset contains a subset of the fuel economy data that the EPA makes available on http://fueleconomy.gov. It contains only models which had a new release every year between 1999 and 2008 - this was used as a proxy for the popularity of the car.

manufacturer	manufacturer	**drv**	f = front-wheel drive, r = rear wheel drive, 4 = 4wd
model	model name	**cty**	city miles per gallon
displ	engine displacement, in litres	**hwy**	highway miles per gallon
year	year of manufacture	**fl**	fuel type
cyl	number of cylinders	**class**	“type” of car
trans	type of transmission		

First Steps

- Generate a first graph to help answer the following question:
  - *Do cars with big engines use more fuel than cars with small engines*
- What might the relationship between engine size and fuel efficiency look like?
  - Positive or negative?
  - Linear or non-linear?

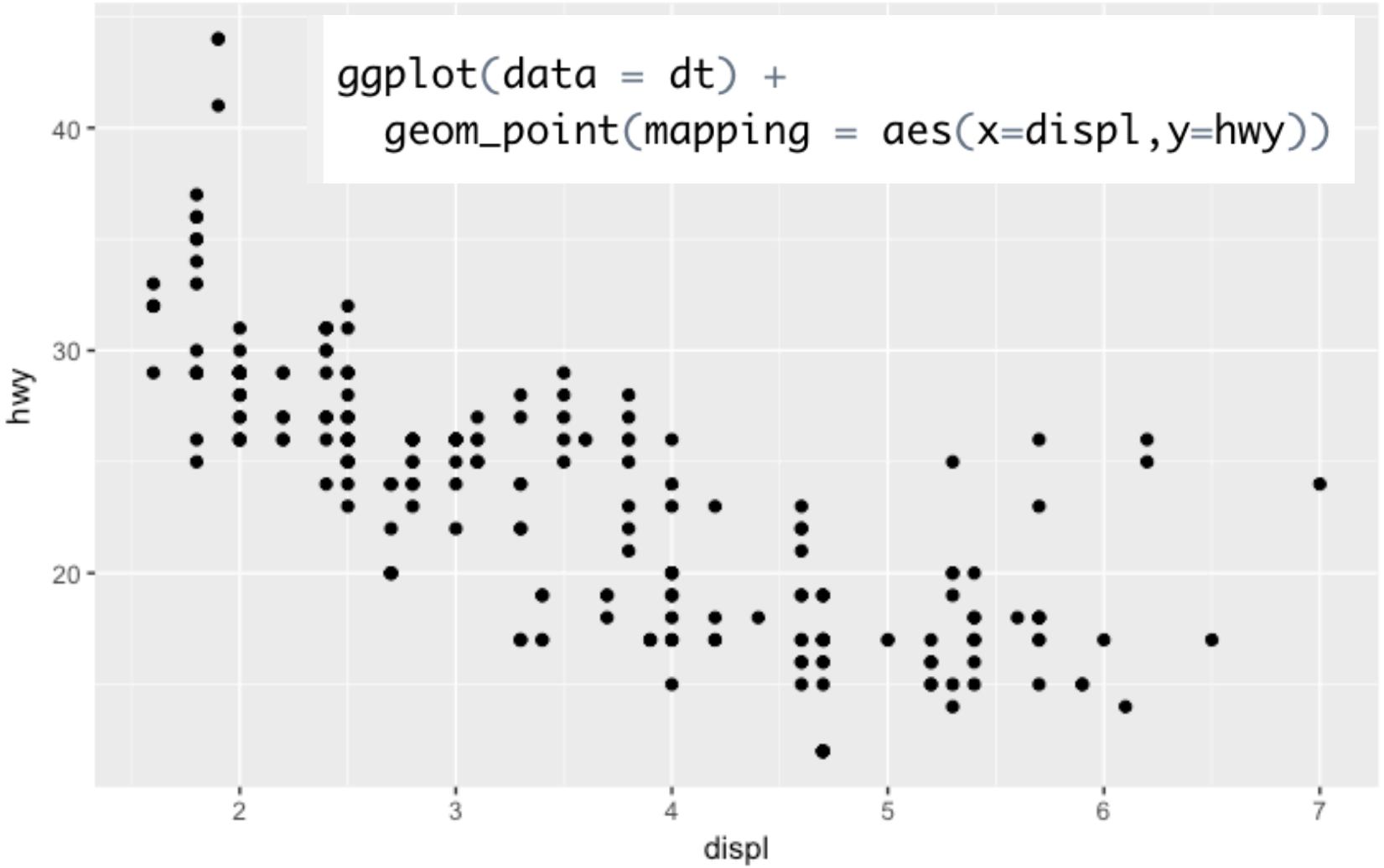
Selecting data

```
> dt
# A tibble: 234 × 11
   manufacturer      model displ  year   cyl      trans   drv   cty   hwy    fl   class
          <chr>      <chr> <dbl> <int> <int>      <chr> <chr> <int> <int> <chr>   <chr>
1          audi         a4   1.8  1999     4   auto(l5)     f    18    29     p compact
2          audi         a4   1.8  1999     4 manual(m5)     f    21    29     p compact
3          audi         a4   2.0  2008     4 manual(m6)     f    20    31     p compact
4          audi         a4   2.0  2008     4   auto(av)     f    21    30     p compact
5          audi         a4   2.8  1999     6   auto(l5)     f    16    26     p compact
```

- Among the variables are:
  - **displ**, a car's engine size in litres
  - **hwy**, a car's fuel efficiency on the highway in miles per gallon

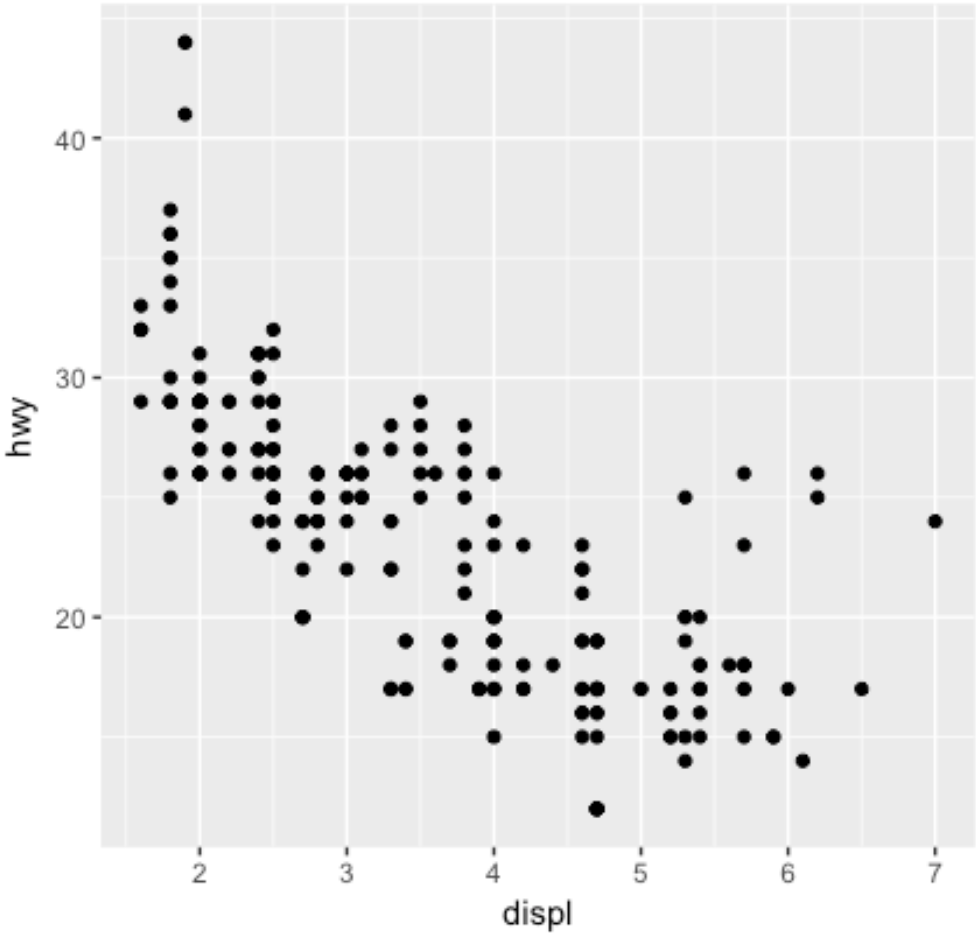
Creating a ggplot

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



Interpreting the plot

- The plot shows a negative relationship between engine size (displ) and fuel efficiency (hwy)
- Cars with big engines use more fuel
- Does this confirm or refute your hypothesis about fuel efficiency and engine size?



Challenge 5.2

- Explore the hypothesis that city driving is less fuel efficient that highway driving
- Use ggplot to present the points on the same graph, and colour each data set differently
- Does the data confirm or refute your initial hypothesis?

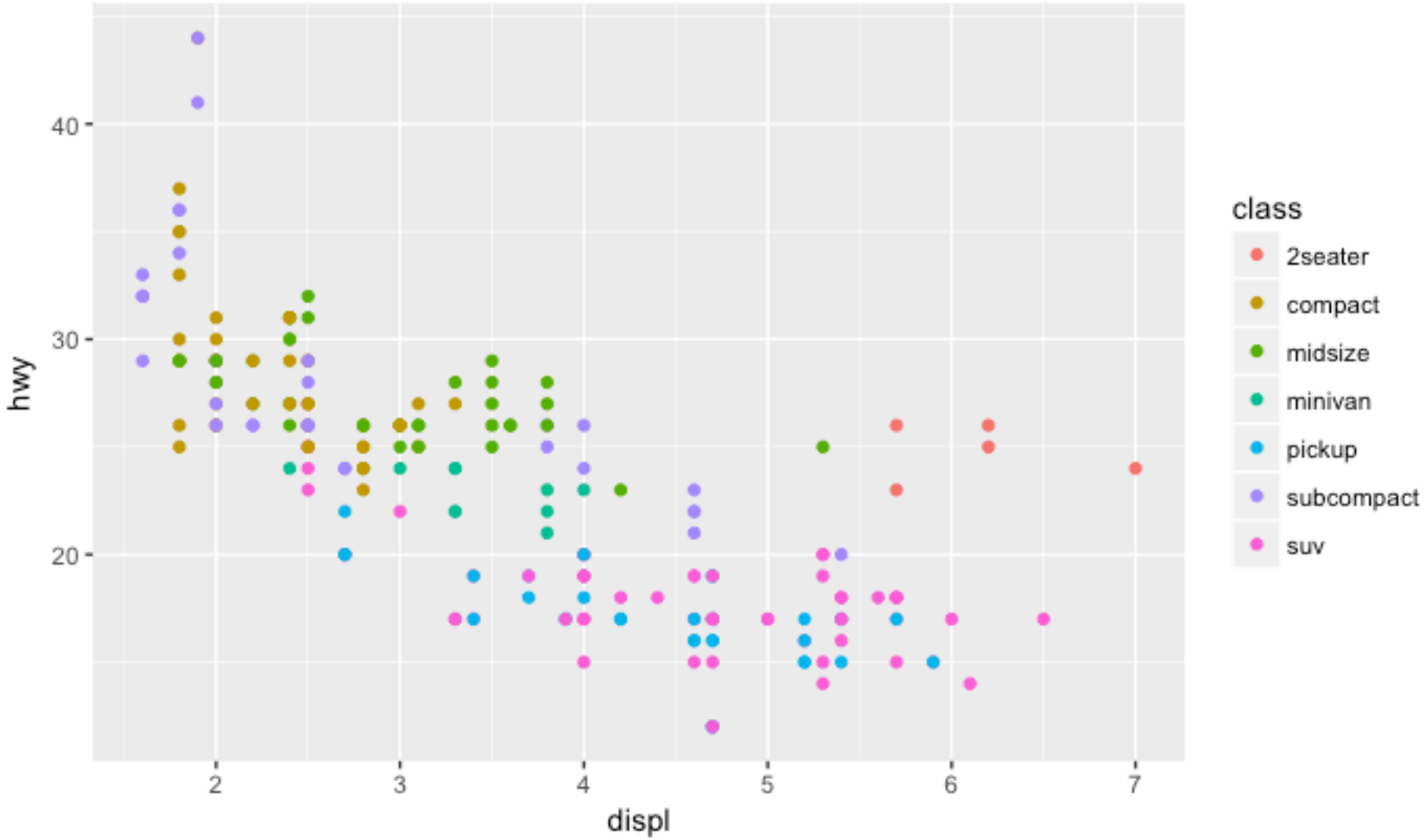
(3) Aesthetic Mappings

"The greatest value of a picture is when it forces us to notice what we never expected to see" – John Tukey

```
> unique(dt$class)
[1] "compact"    "midsize"    "suv"        "2seater"    "minivan"
[6] "pickup"     "subcompact"
```

- A third variable can be added to a 2-D plot by mapping it to an aesthetic.
- An aesthetic is a visual property of the plot's objects.
- An aesthetic's *level* could be colour, size or shape.

```
ggplot(data = dt) +
  geom_point(mapping = aes(x=displ,y=hwy,colour=class))
```



(4) Common Problems

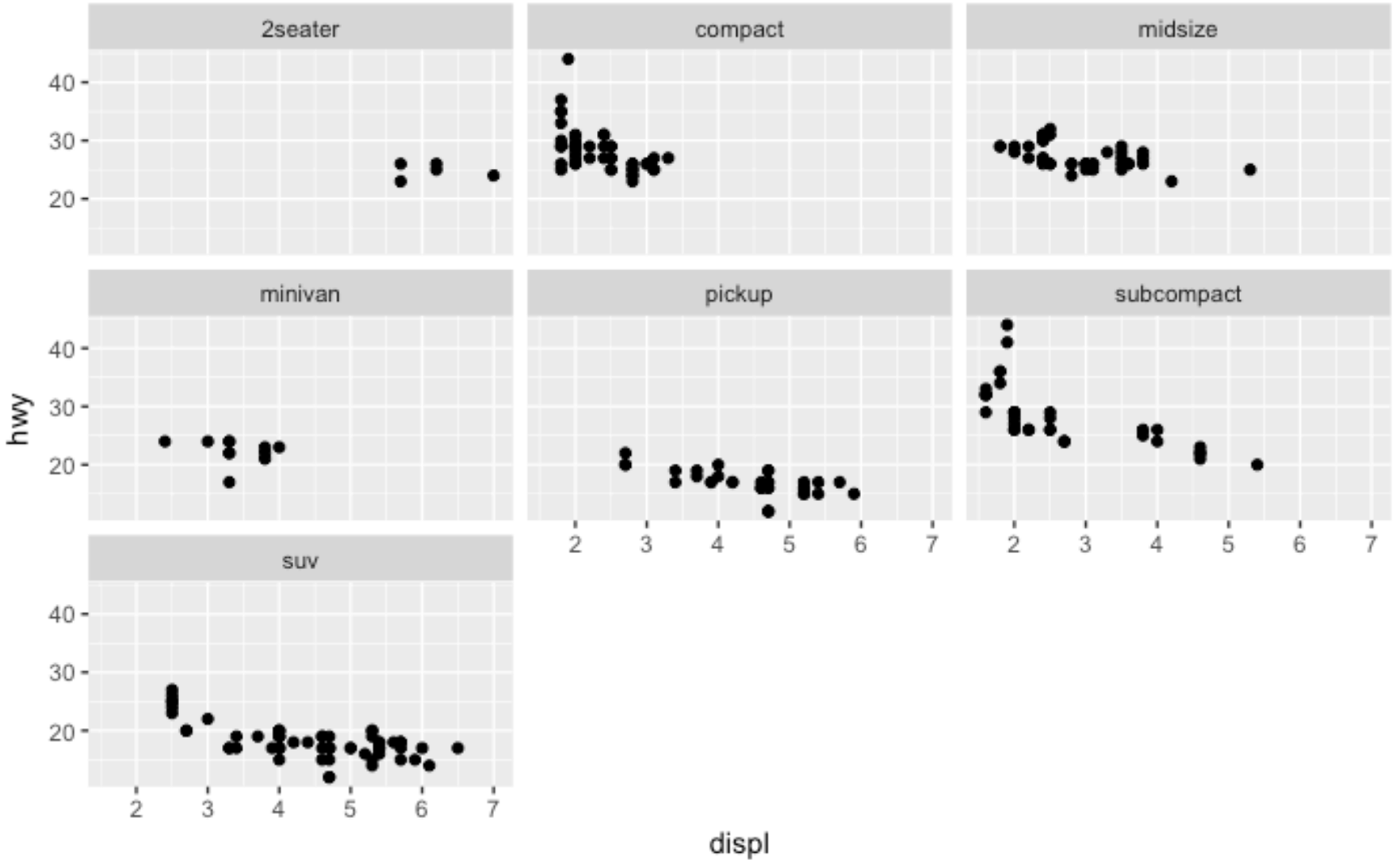
- R can be “extremely picky, and a misplaced character can make all the difference”
- Make sure every (is matched with a)
- For ggplot calls, the + must come at the end of the line, not at the start (see below)
- You can get help about any function by running ? function_name

```
> ggplot(data=d)
>   +geom_point(aes(x=displ,y=hwy),colour="blue")
Error in +geom_point(aes(x = displ, y = hwy), colour = "blue") :
  invalid argument to unary operator
```

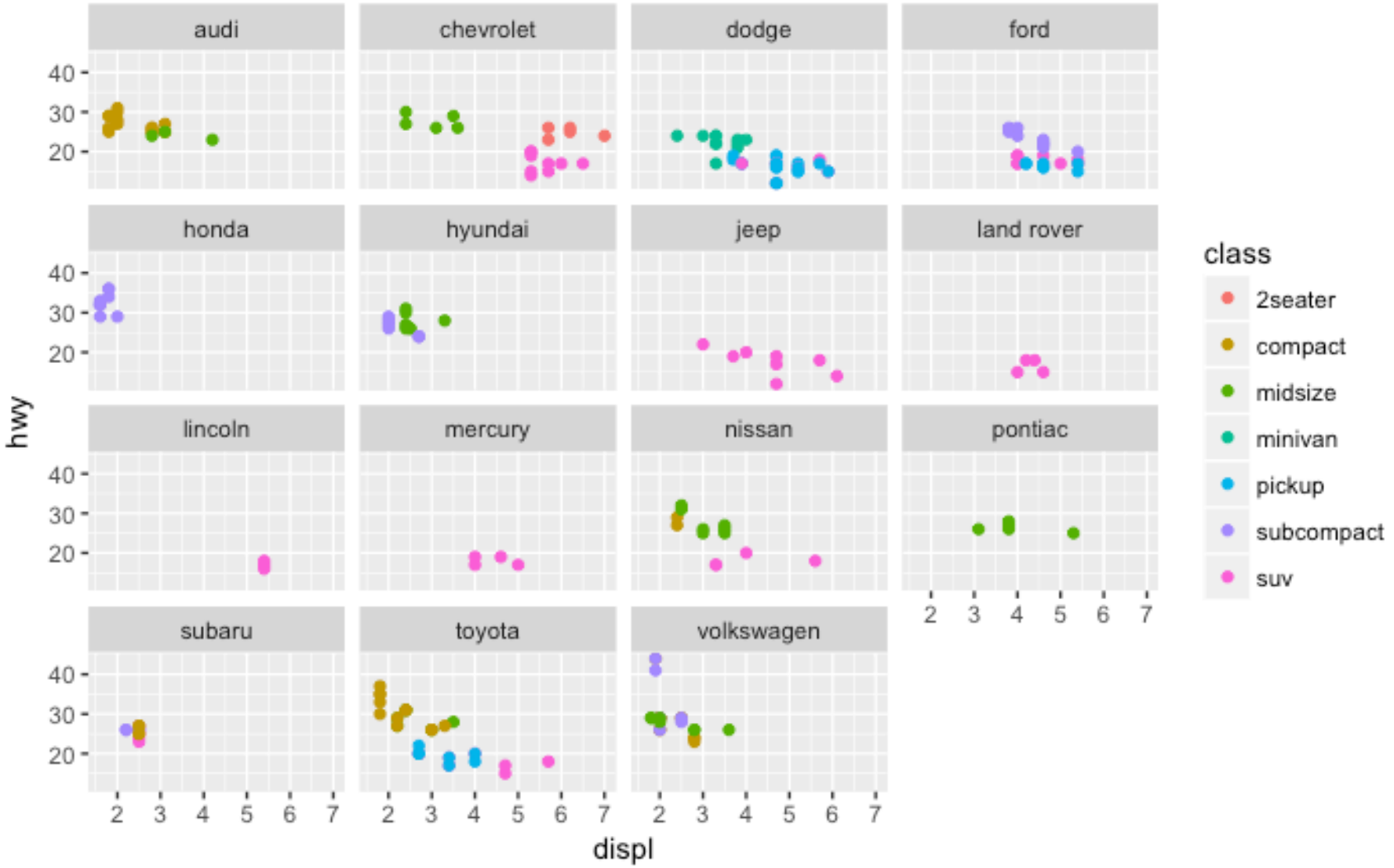
(5) Facets

- Another way to add categorical variables is to split a plot into facets, subplots that display one subset of the data.
- To facet your plot by a single variable, use facet_wrap(), with ~ followed by the variable name
- To facet on the combination of two variables, used facet_grid()

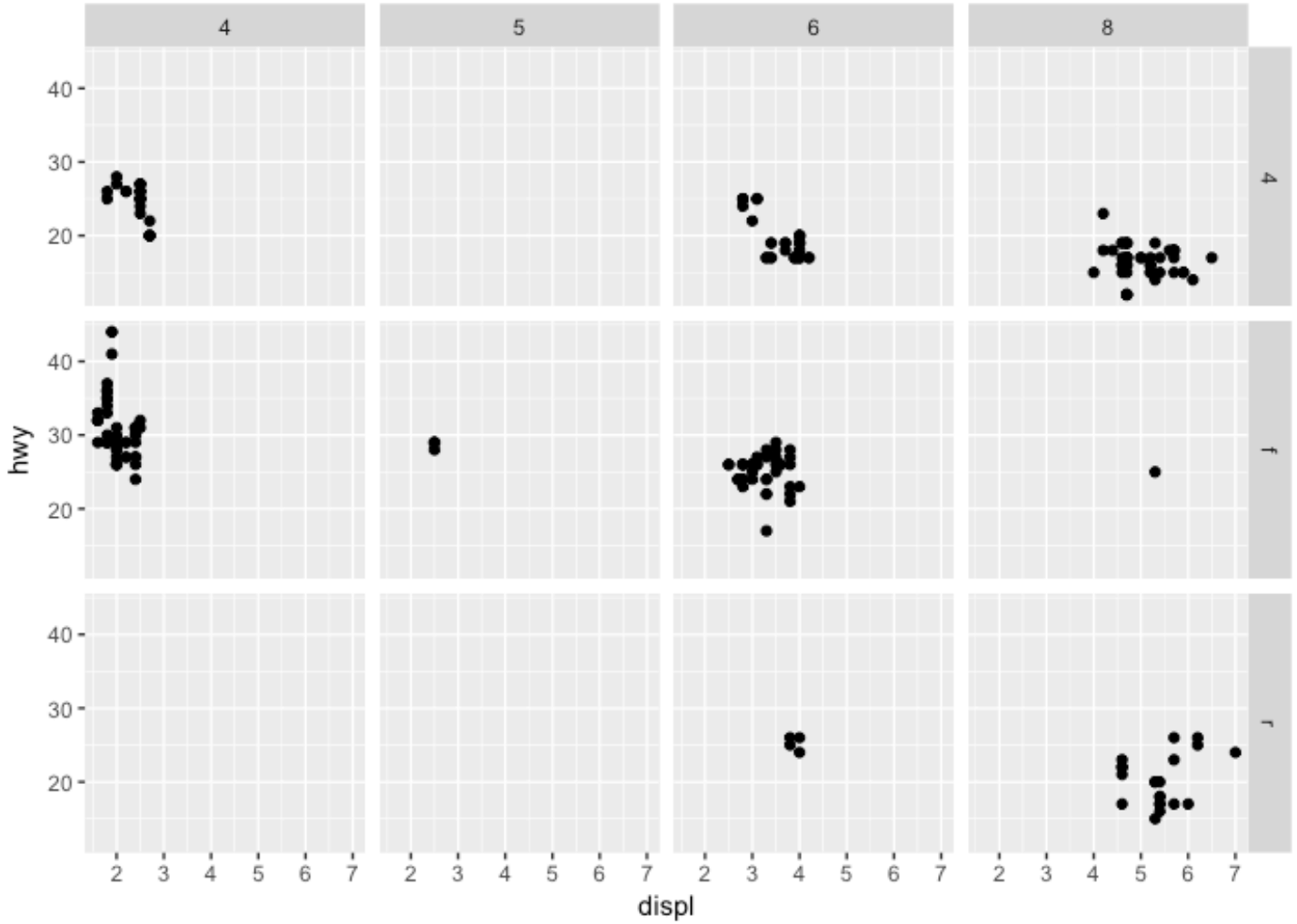
```
ggplot(data = dt) +
  geom_point(mapping = aes(x=displ,y=hwy)) + facet_wrap(~class)
```



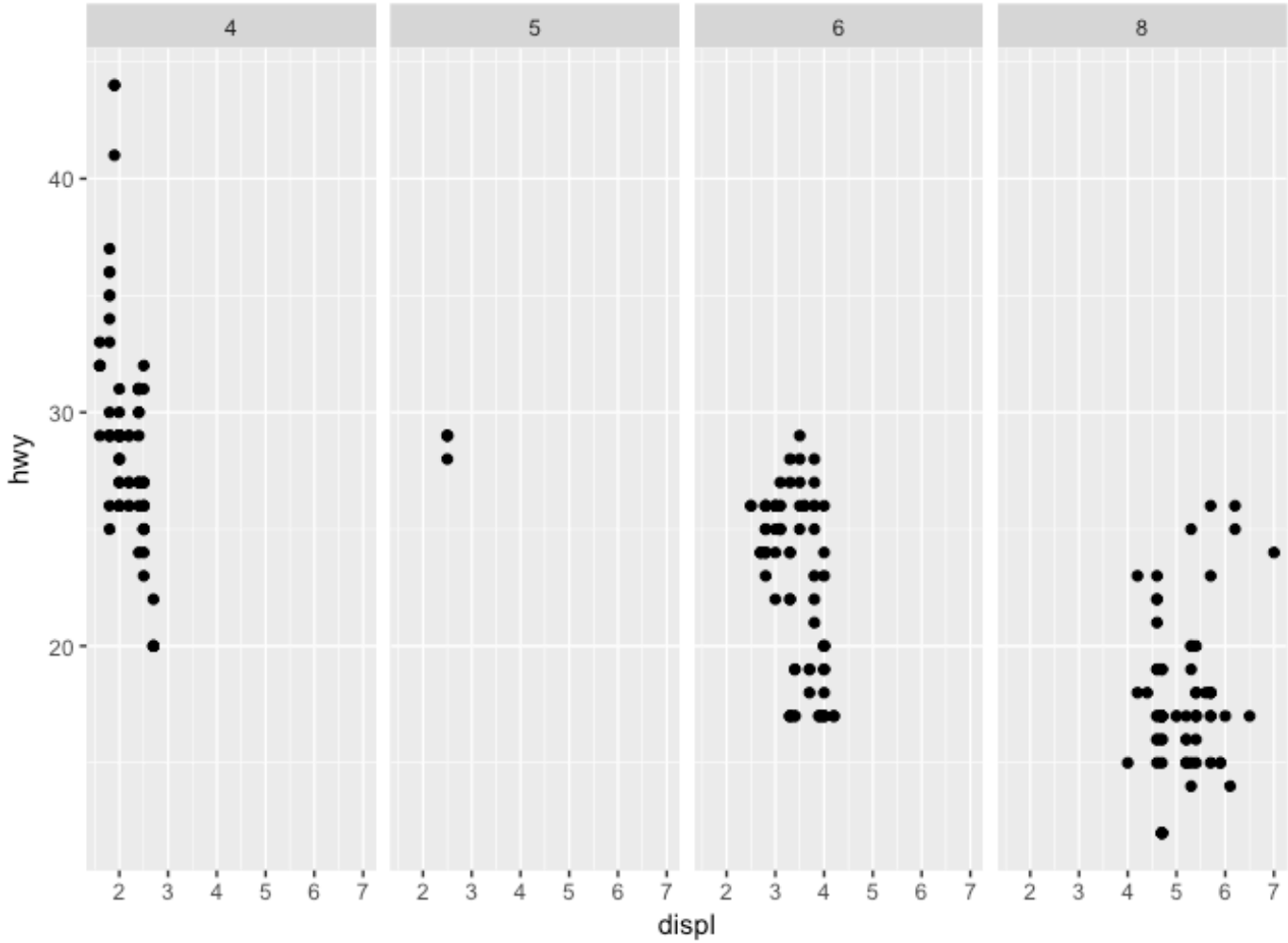
```
ggplot(data = dt) +
  geom_point(mapping = aes(x=displ,y=hwy,colour=class)) +
  facet_wrap(~manufacturer)
```



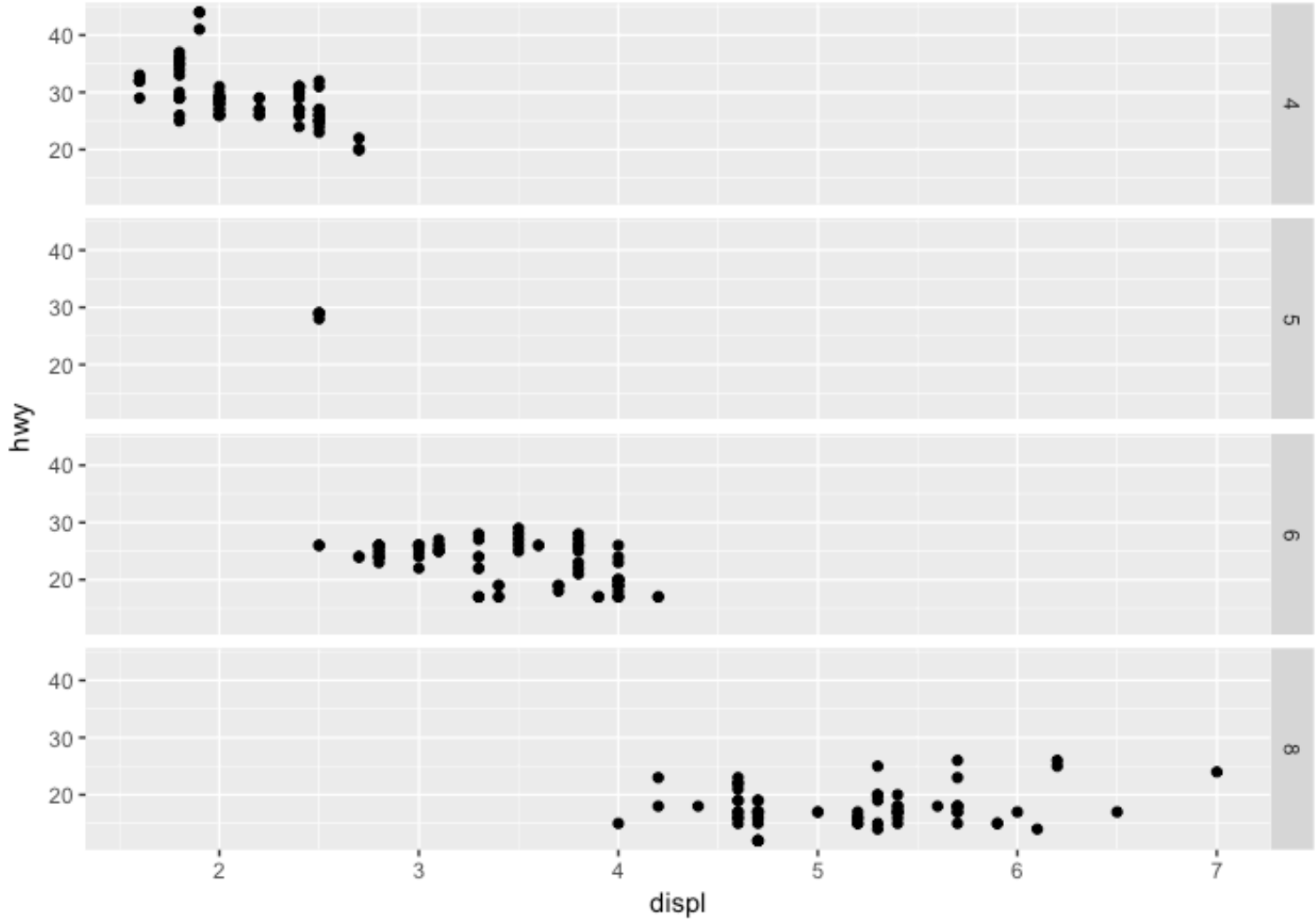
```
ggplot(data=mpg) +
  geom_point(mapping = aes(x=displ, y = hwy)) +
  facet_grid(drv ~ cyl)
```



```
ggplot(data=mpg) +
  geom_point(mapping = aes(x=displ, y = hwy)) +
  facet_grid(. ~ cyl)
```

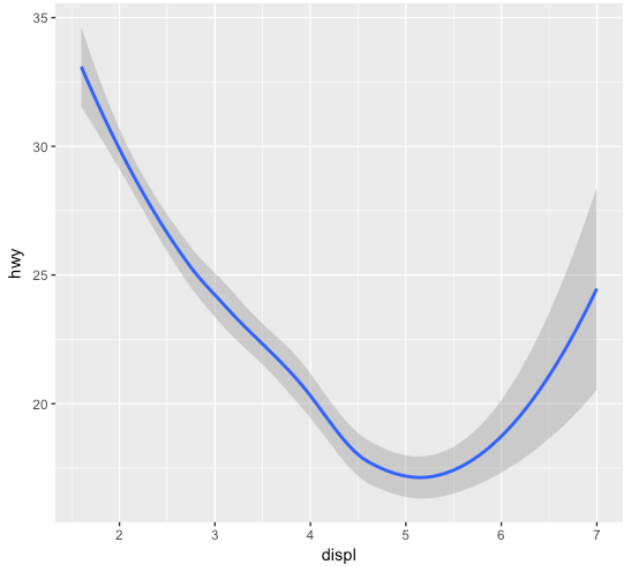


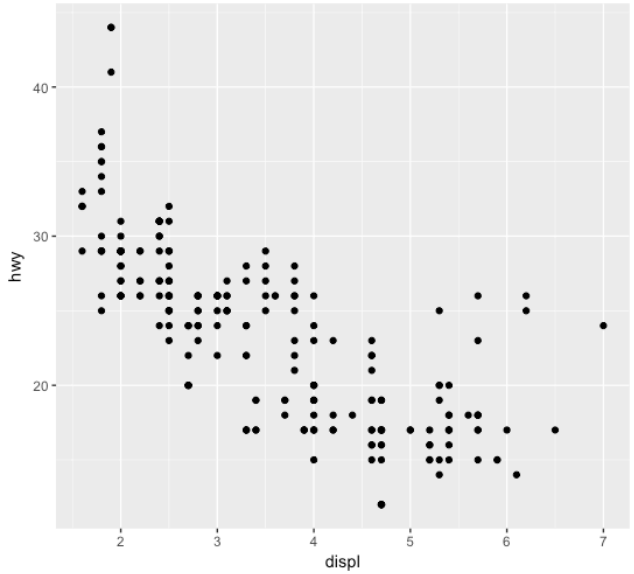
```
ggplot(data=mpg) +
  geom_point(mapping = aes(x=displ, y = hwy)) +
  facet_grid(cyl ~ .)
```



(6) Geometric Objects

- Both of these plots contain the same x and y variable, and describe the same data
- The plots are not identical, they use a different visual object to represent the data
- In ggplot2 syntax, we say the use different ***geoms***



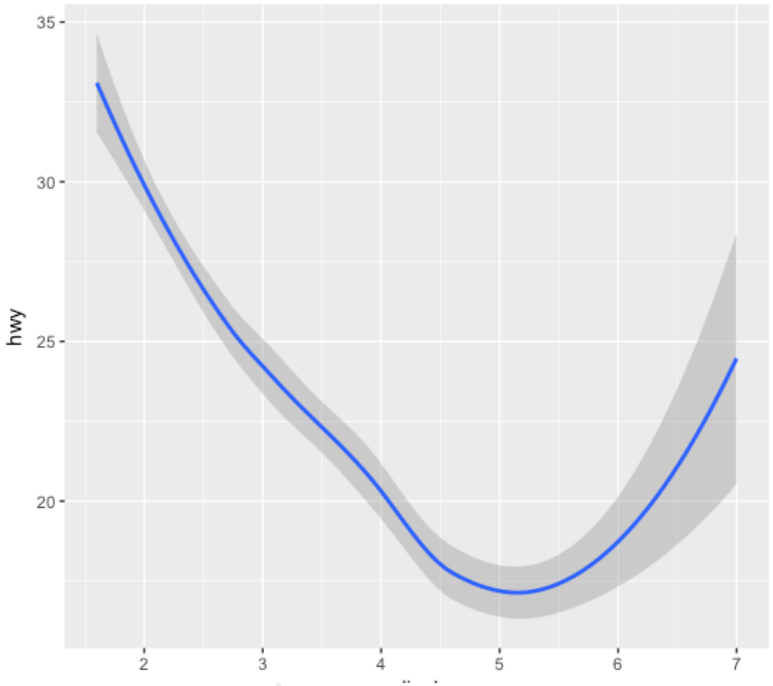


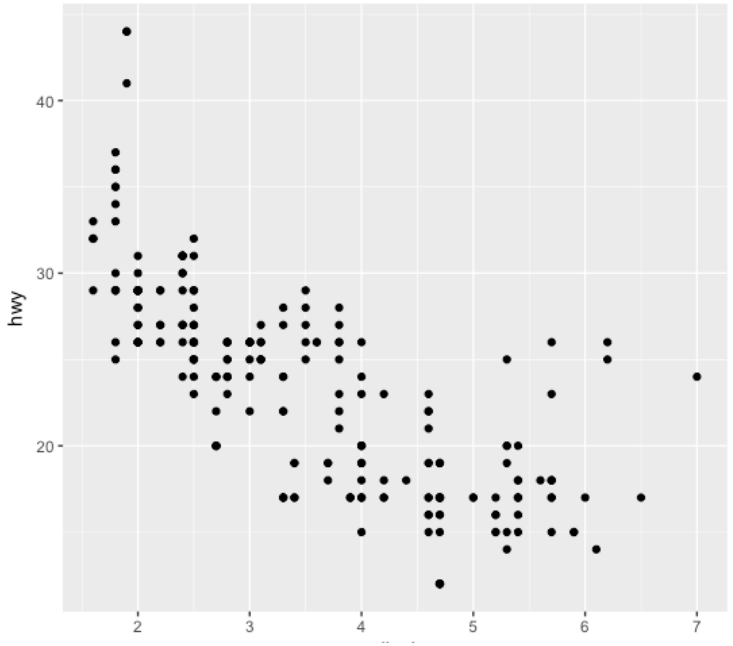
geom

- A geom is a geometrical object that a plot uses to represent data
- Bar charts use bar geoms, line charts use line geoms, and scatter plots use the point geom.
- To change the geom in your plot, simply change the geom function that is added to the ggplot call.

Examples of using different geoms

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

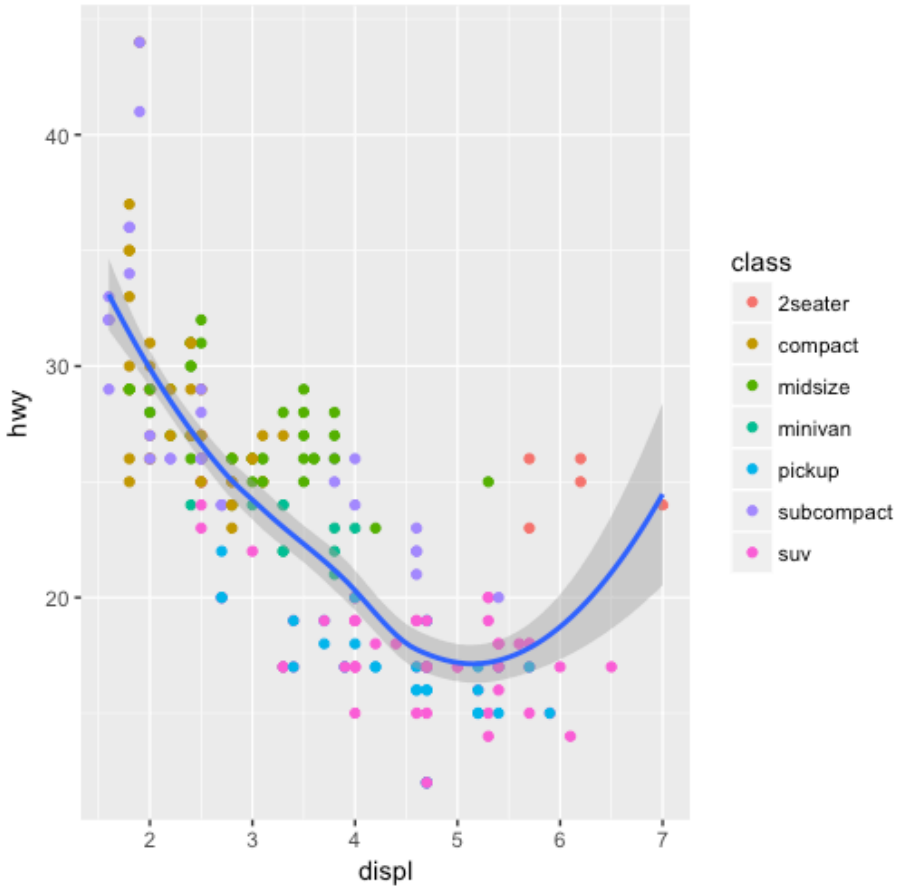





```
ggplot(data=mpg)+
  geom_point(mapping=aes(x=displ,y=hwy))
```

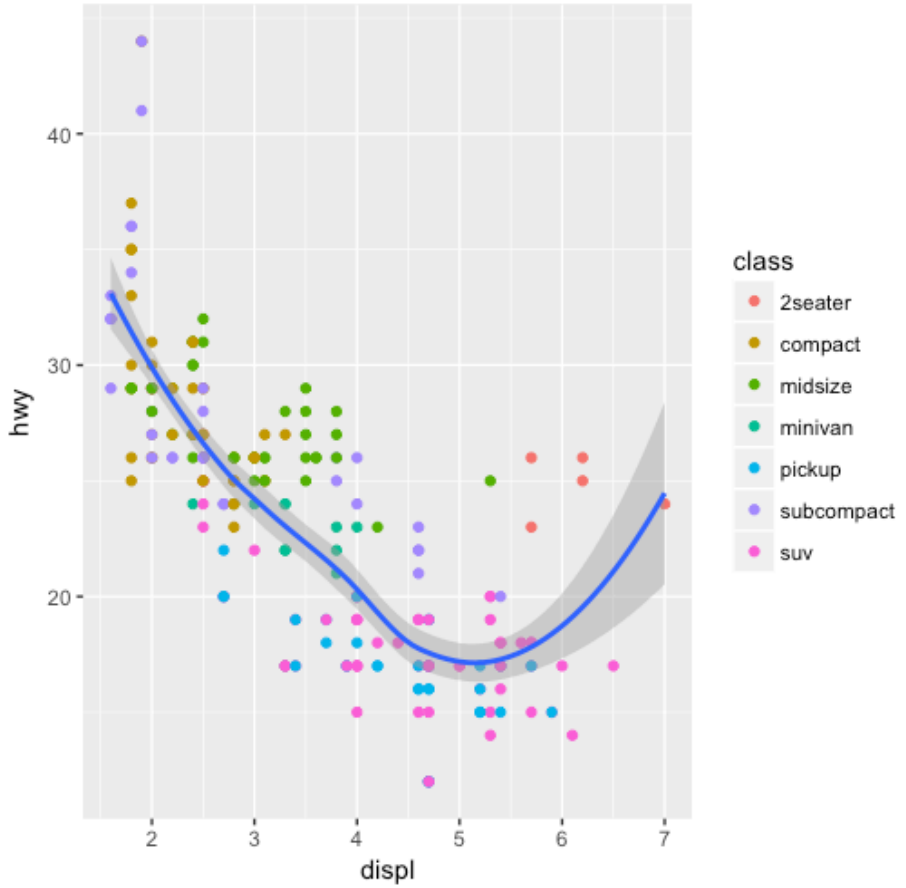
Displaying Multiple geoms

- Multiple geoms can be displayed on the same plot
- Data can be specified in first ggplot() call, and shared by all geoms
- Also, different geoms can have their own data



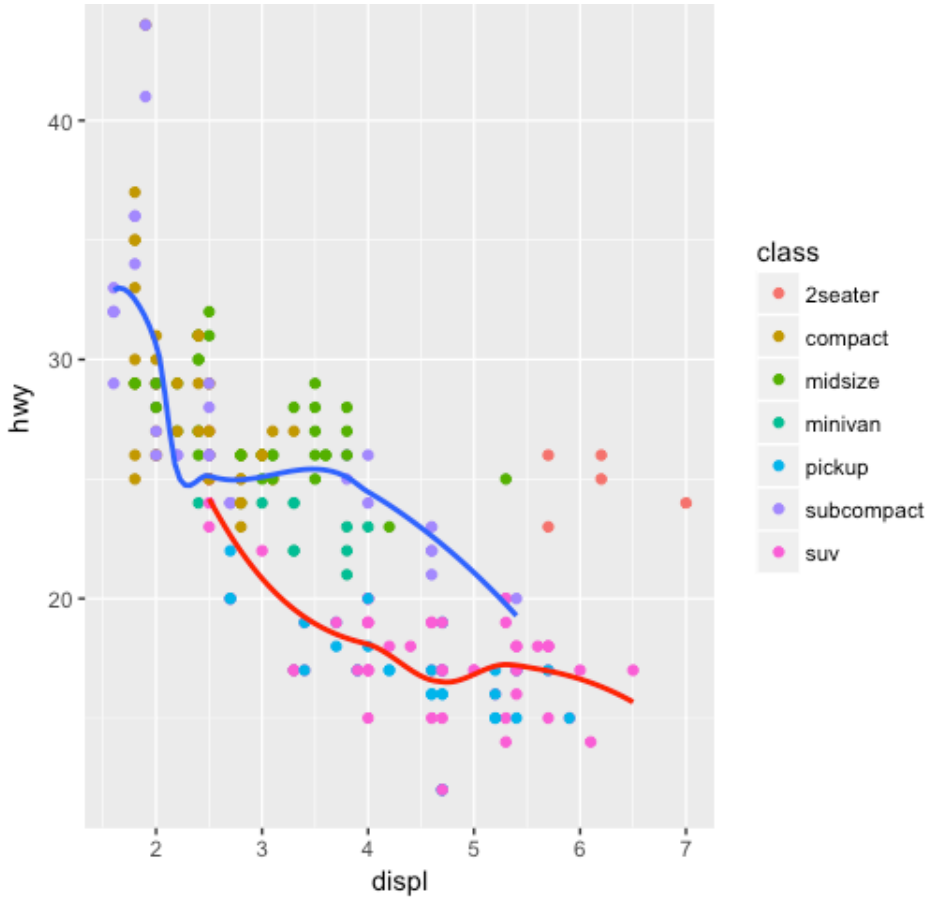
```
ggplot(data=mpg, mapping = aes(x=displ, y= hwy)) +
  geom_point(aes(colour=class)) + geom_smooth()
```

- Data and x,y can be defined in the first call, and then used by the different geoms
- Additional attributes can then be added for geoms (i.e. for specific layers)
- *This makes it possible to display different aesthetics in different layers*



```
ggplot(data=mpg,mapping=aes(x=displ,y=hwy))+
  geom_point(mapping=aes(colour=class))+
  geom_smooth(data=filter(mpg,class=="subcompact"),
              se=F)+
  geom_smooth(data=filter(mpg,class=="suv"),
              se=F,colour="red")
```

- Different data can be specified for each layer
- A local data argument can override a global data argument for a specific layer
- filter() will be explained in a subsequent lecture, it is part of dplyr()



Sample plot geoms

Geom	Purpose
geom_smooth()	Fits a smoother to data and displays the smooth and its standard error
geom_boxplot()	Produces a box-and-whisker plot to summarise the distribution of a set of points
geom_histogram() geom_freqpoly()	Shows the distribution of continuous variables
geom_bar()	Shows the distribution of categorical variables
geom_path() geom_line()	Draws lines between data points
geom_area()	Draws an area plot, which is a line plot filled to the y-axis. Multiple groups will be stacked upon each other
geom_rect() geom_tile() geom_raster()	Draw rectangles
geom_polygon()	Draws polygons, which are filled paths.

Challenge 5.3

- Will these two graphs look different. Why/ why not?

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

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

diamonds data set (ggplot2)

A dataset containing the prices and other attributes of almost 54,000 diamonds.

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

Explanation of variables

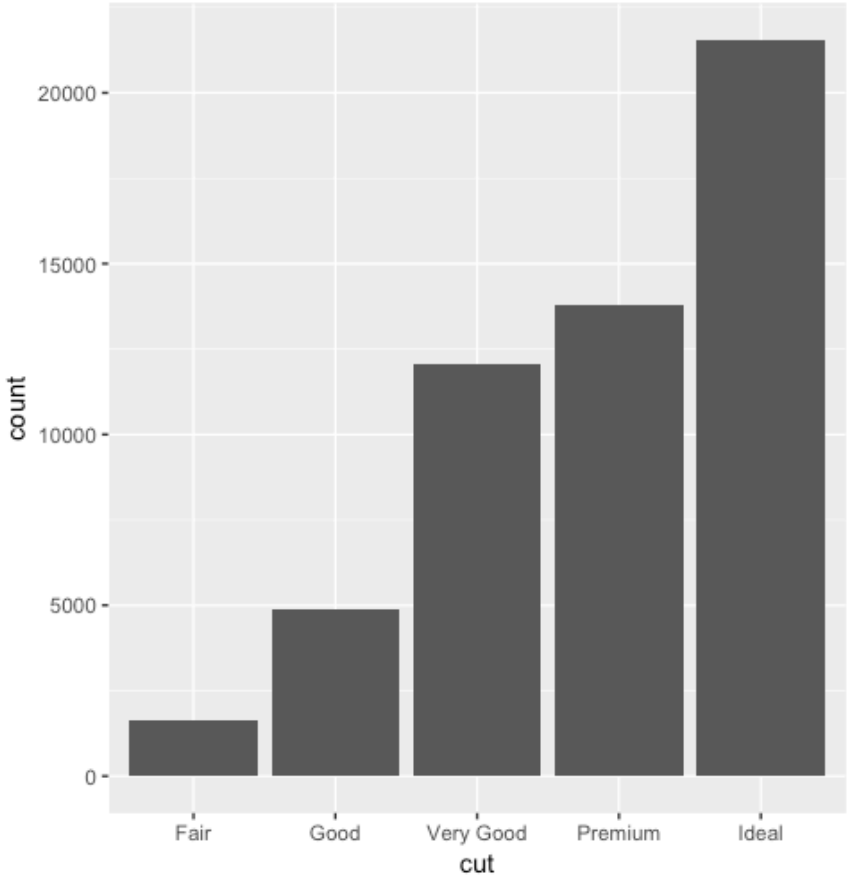
Feature	Explanation
price	price in US dollars $326–$18,823
carat	weight of the diamond (0.2–5.01)
cut	quality of the cut (Fair, Good, Very Good, Premium, Ideal)
color	diamond colour, from J (worst) to D (best)
clarity	a measurement of how clear the diamond is (I1 (worst), SI1, SI2, VS1, VS2, VVS1, VVS2, IF (best))
x	length in mm (0–10.74)
y	width in mm (0–58.9)
z	depth in mm (0–31.8)
depth	total depth percentage = z / mean(x, y) = 2 * z / (x + y) (43–79)
table	width of top of diamond relative to widest point (43–95)

Summary of dataset

```
> summary(diamonds)
     carat               cut        color        clarity          depth      
 Min.   :0.2000   Fair      : 1610   D: 6775   SI1     :13065   Min.   :43.00  
 1st Qu.:0.4000   Good      : 4906   E: 9797   VS2     :12258   1st Qu.:61.00  
 Median :0.7000   Very Good:12082   F: 9542   SI2     : 9194   Median :61.80  
 Mean   :0.7979   Premium   :13791   G:11292   VS1     : 8171   Mean   :61.75  
 3rd Qu.:1.0400   Ideal     :21551   H: 8304   VVS2    : 5066   3rd Qu.:62.50  
 Max.   :5.0100                     I: 5422   VVS1    : 3655   Max.   :79.00  
                                    J: 2808   (Other): 2531                  
     table           price             x                y                z          
 Min.   :43.00   Min.   :  326   Min.   : 0.000   Min.   : 0.000   Min.   : 0.000  
 1st Qu.:56.00   1st Qu.:  950   1st Qu.: 4.710   1st Qu.: 4.720   1st Qu.: 2.910  
 Median :57.00   Median : 2401   Median : 5.700   Median : 5.710   Median : 3.530  
 Mean   :57.46   Mean   : 3933   Mean   : 5.731   Mean   : 5.735   Mean   : 3.539  
 3rd Qu.:59.00   3rd Qu.: 5324   3rd Qu.: 6.540   3rd Qu.: 6.540   3rd Qu.: 4.040  
 Max.   :95.00   Max.   :18823   Max.   :10.740   Max.   :58.900   Max.   :31.800  
```

(7) Statistical Transformations

- Lets explore the *bar chart*: appears simple, yet reveals a subtle feature of plots
- The bar chart geom_bar() shows the total number of diamonds, grouped by cut
- **But where does the count come from?**



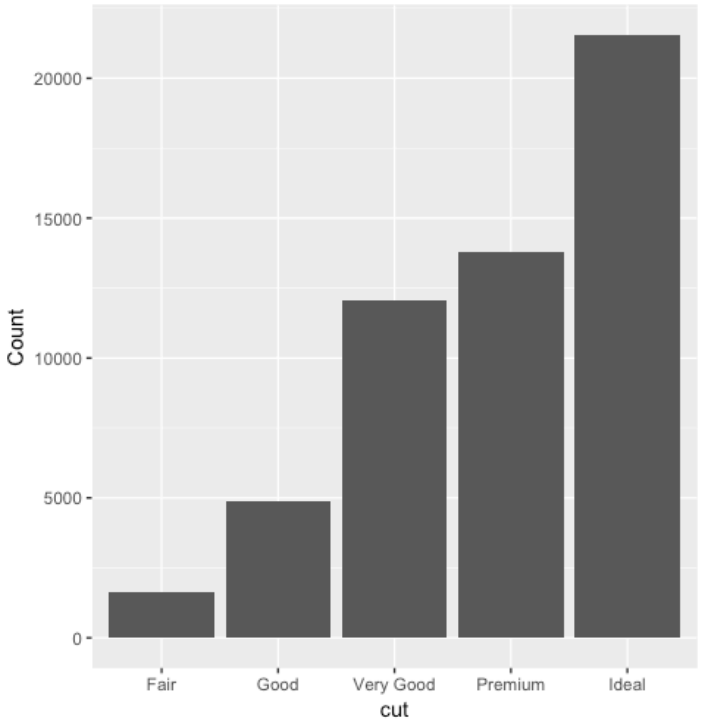

```
ggplot(data=diamonds) +
  geom_bar(mapping = aes(x = cut))
```

Explanation

- Many graphs, like scatterplots, plot the raw values of the dataset
- However, other graphs (e.g. bar charts) calculate new values to plot
  - **Bar charts, histograms and frequency polygons** bin your data and plot bin counts, the number of points that fall in each bin
  - **Smoothers** fit a model to your data and the plot predictions from the model
  - **Boxplots** compute a robust summary of the distribution and display a specially formatted box

Overriding the default stat

- Every geom has a default stat, and every stat has a default geom.
- What is the aggregated data was already contained in 5 rows?
- **Use stat="identity"**

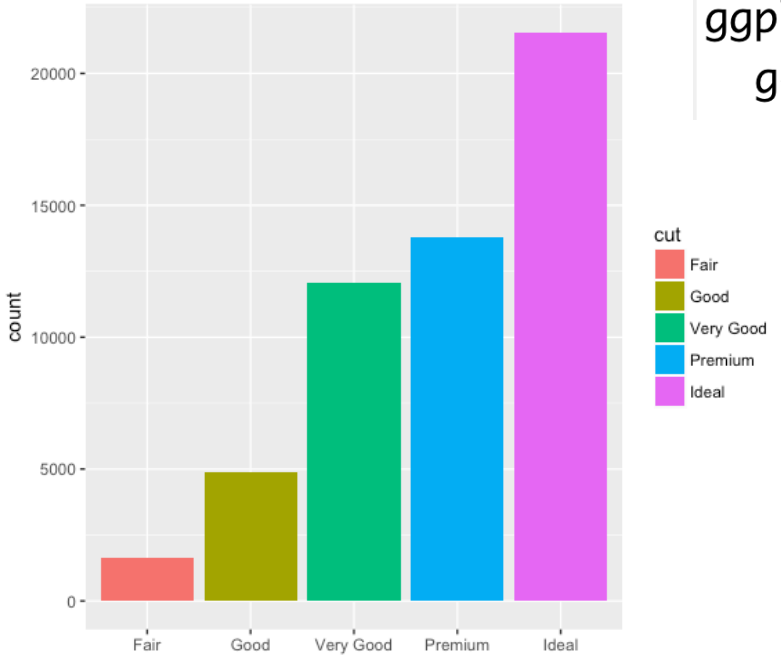



```
        cut Count
      <ord> <int>
1      Fair  1610
2      Good  4906
3 Very Good 12082
4   Premium 13791
5     Ideal 21551
```

```
ggplot(data=agr) +
  geom_bar(mapping = aes(x = cut, y=Count),
           stat="identity")
```

fill aesthetic for bar charts

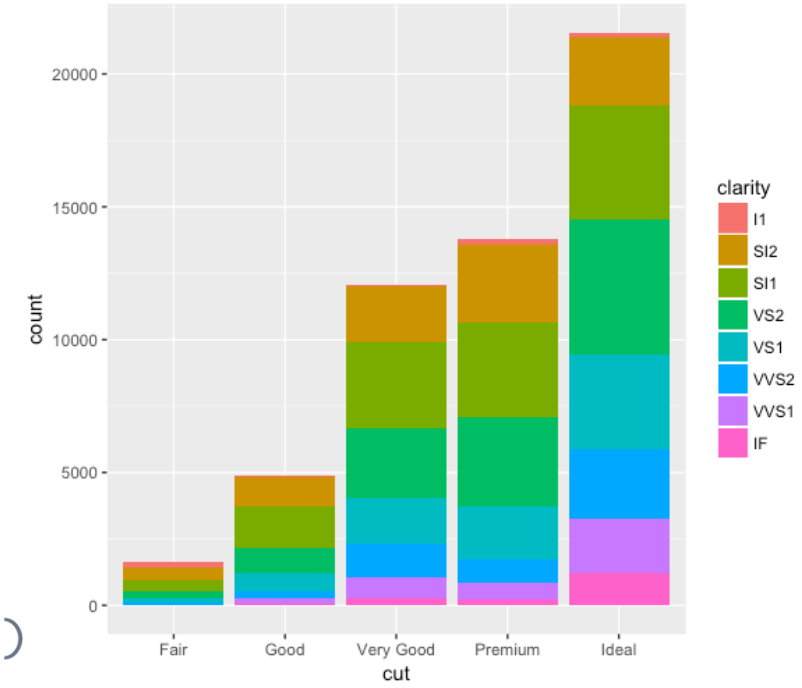
- Bar charts can be coloured using the fill aesthetic




```
ggplot(data=diamonds) +
  geom_bar(mapping=aes(x=cut,fill=cut))
```

- When a different variable is used, the graph has further detail

```
ggplot(data=diamonds) +
  geom_bar(mapping=aes(x=cut,fill=clarity))
```

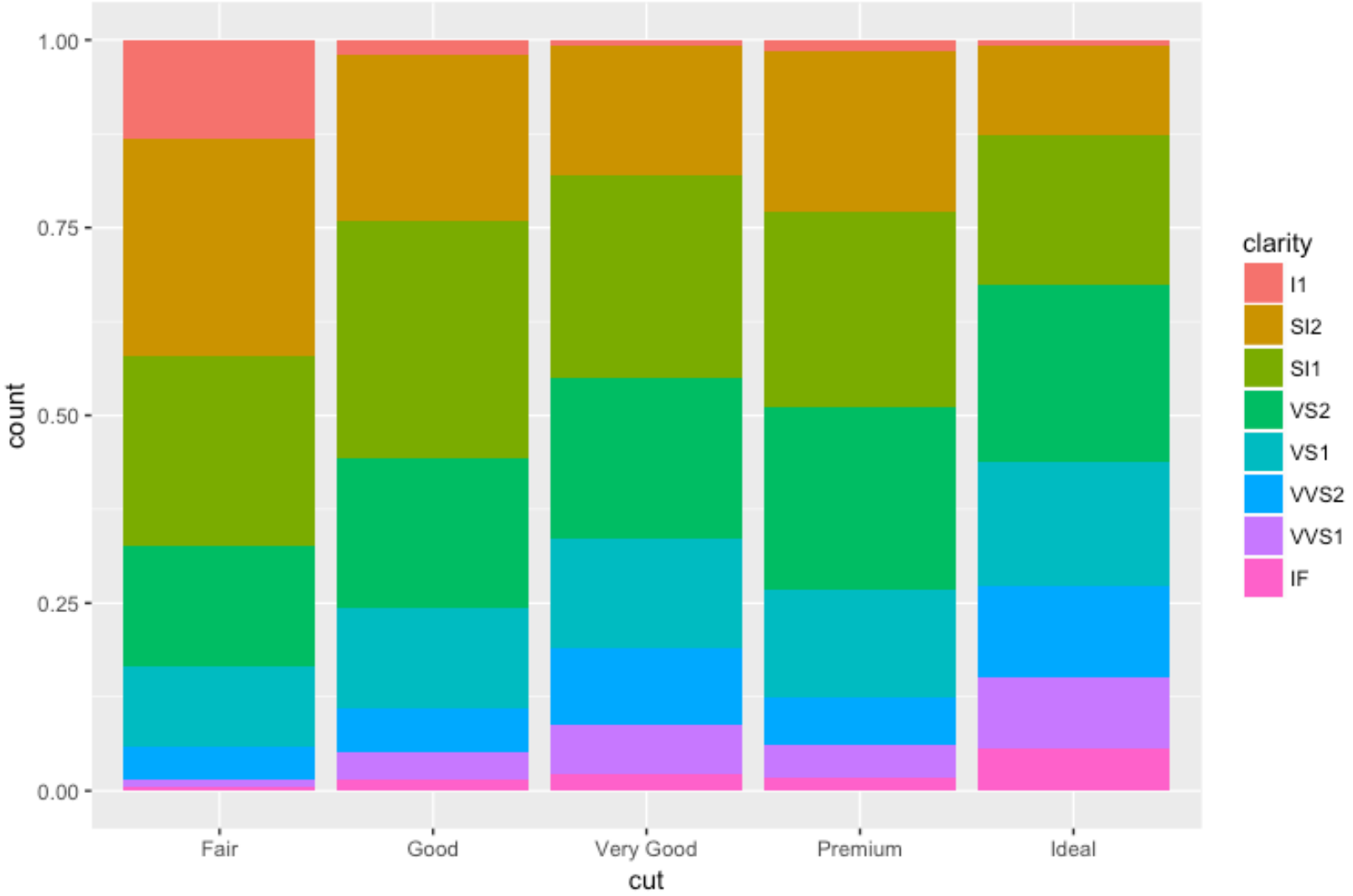


Stacking options

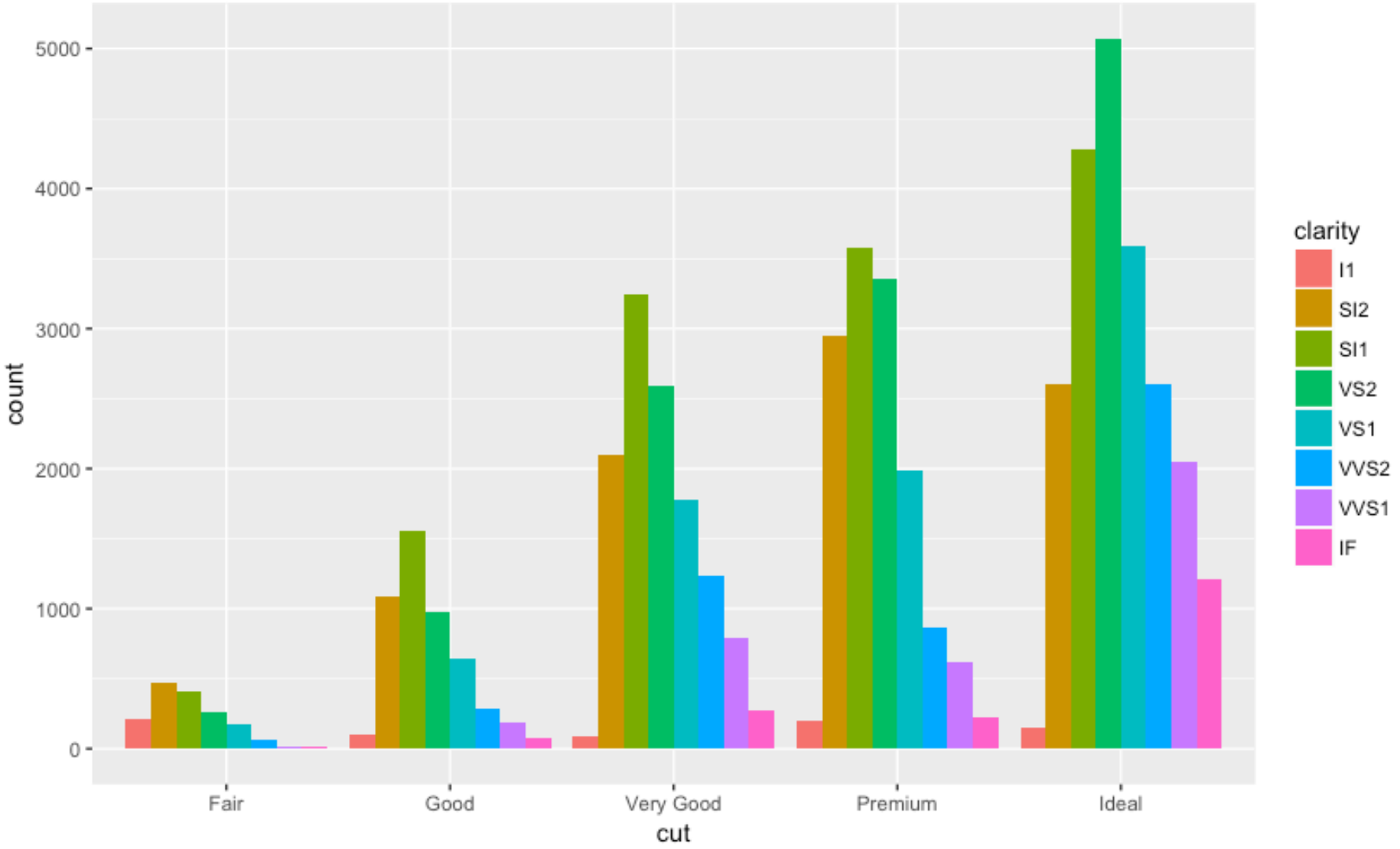
- Stacking is performed automatically by the position adjustment specified by the **position** argument
- Examples include "identity", "fill" and "dodge"

- "fill"
  - Works like stacking, but each stacked bar is the same height
  - Makes it easier to compare proportions
- "dodge"
  - Places objects directly beside one another
  - Makes it easier to compare individual values

```
ggplot(data=diamonds) +
  geom_bar(mapping=aes(x=cut,fill=clarity),
           position="fill")
```

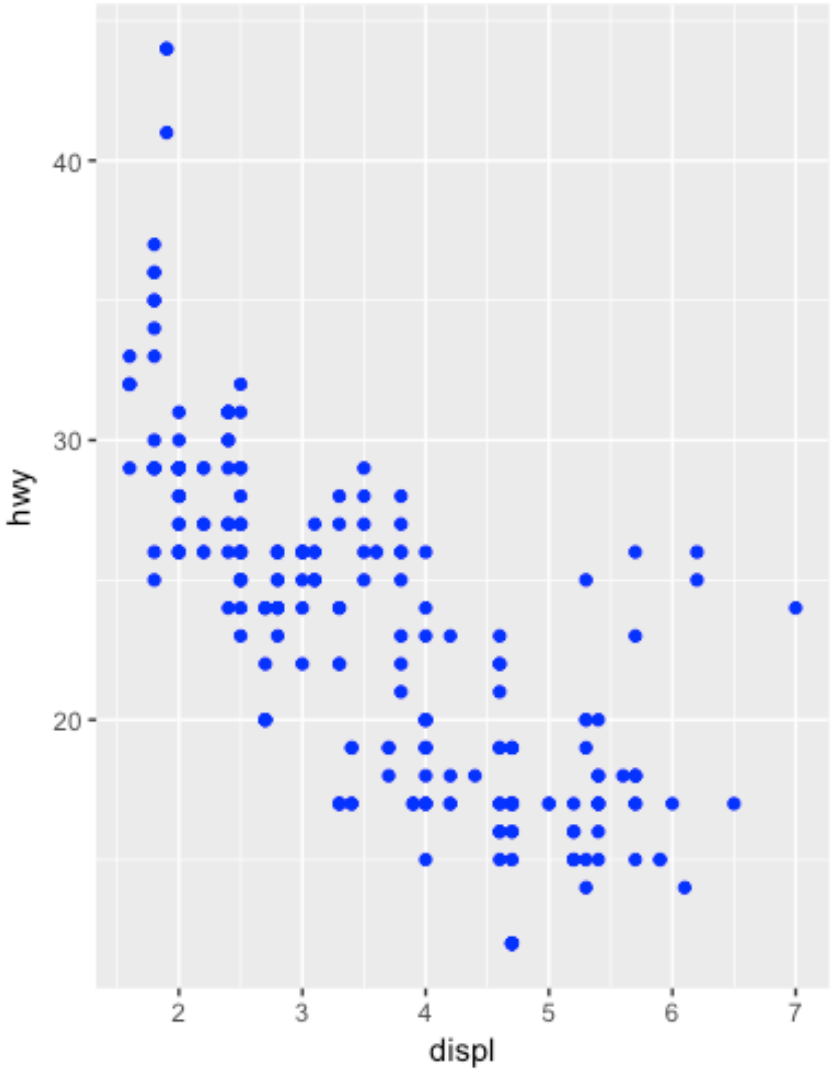


```
ggplot(data=diamonds) +
  geom_bar(mapping=aes(x=cut,fill=clarity),
           position="dodge")
```

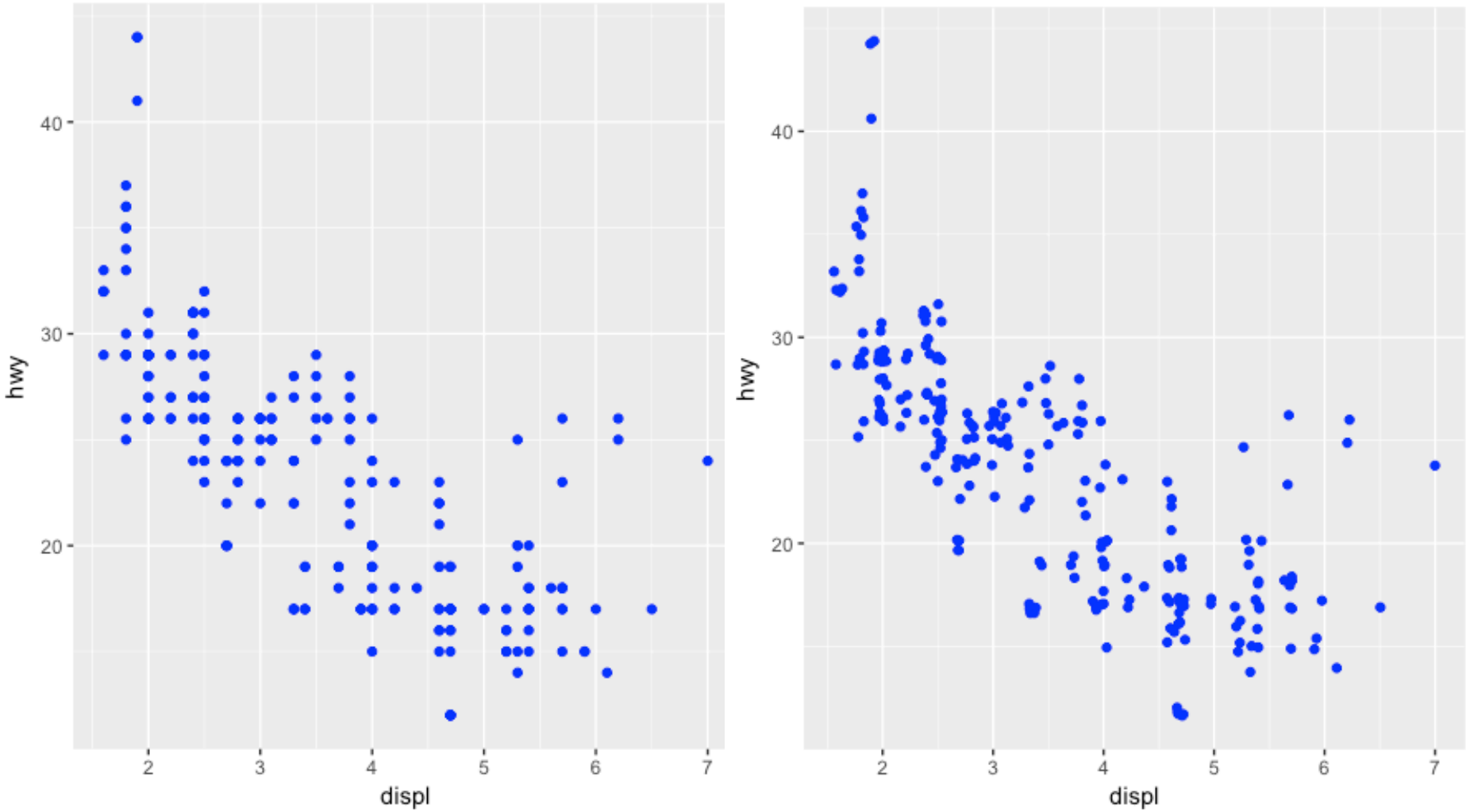


Additional adjustment

- Recall our first scatterplot
- 126 points displayed, yet there are 234 observations
- Many points can overlap, so it makes it hard to see where the mass of data is
- Are all points spread equally, or is there one special combination that contains 129 values?
- “jitter” adds random noise to each point.

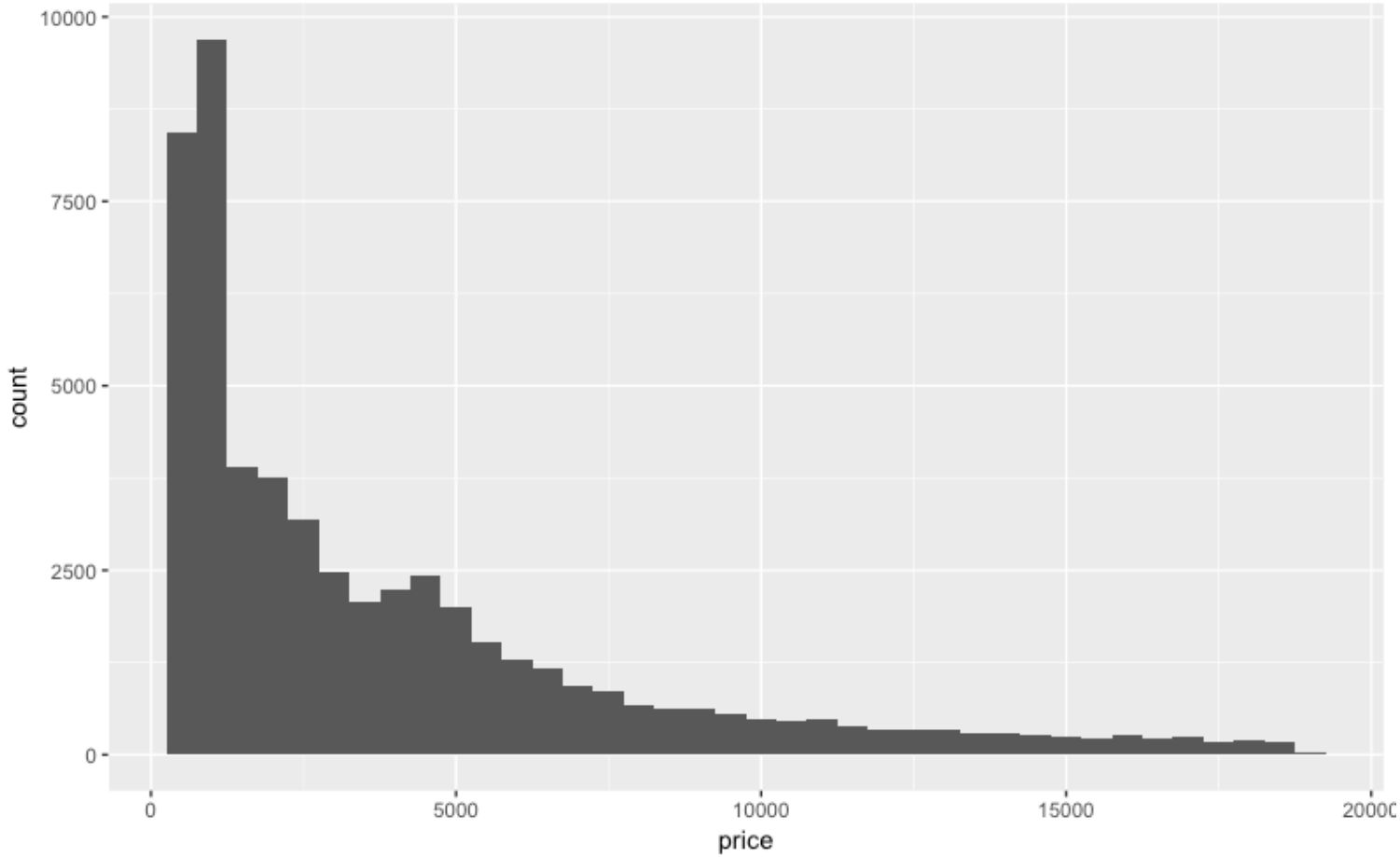


```
ggplot(data=mpg)+
  geom_point(mapping=aes(x=displ,y=hwy),
                 position="jitter",colour="blue")
```



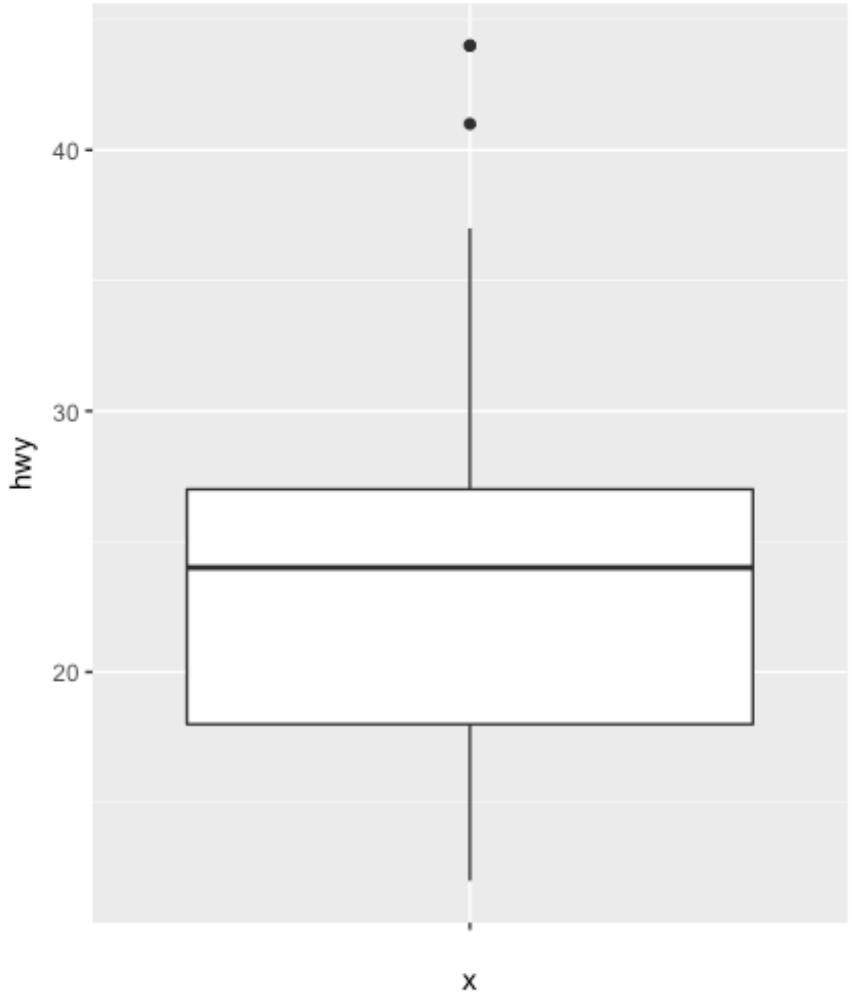
Histogram

```
ggplot(data=diamonds,mapping=aes(x=price)) +
  geom_histogram(binwidth = 500)
```

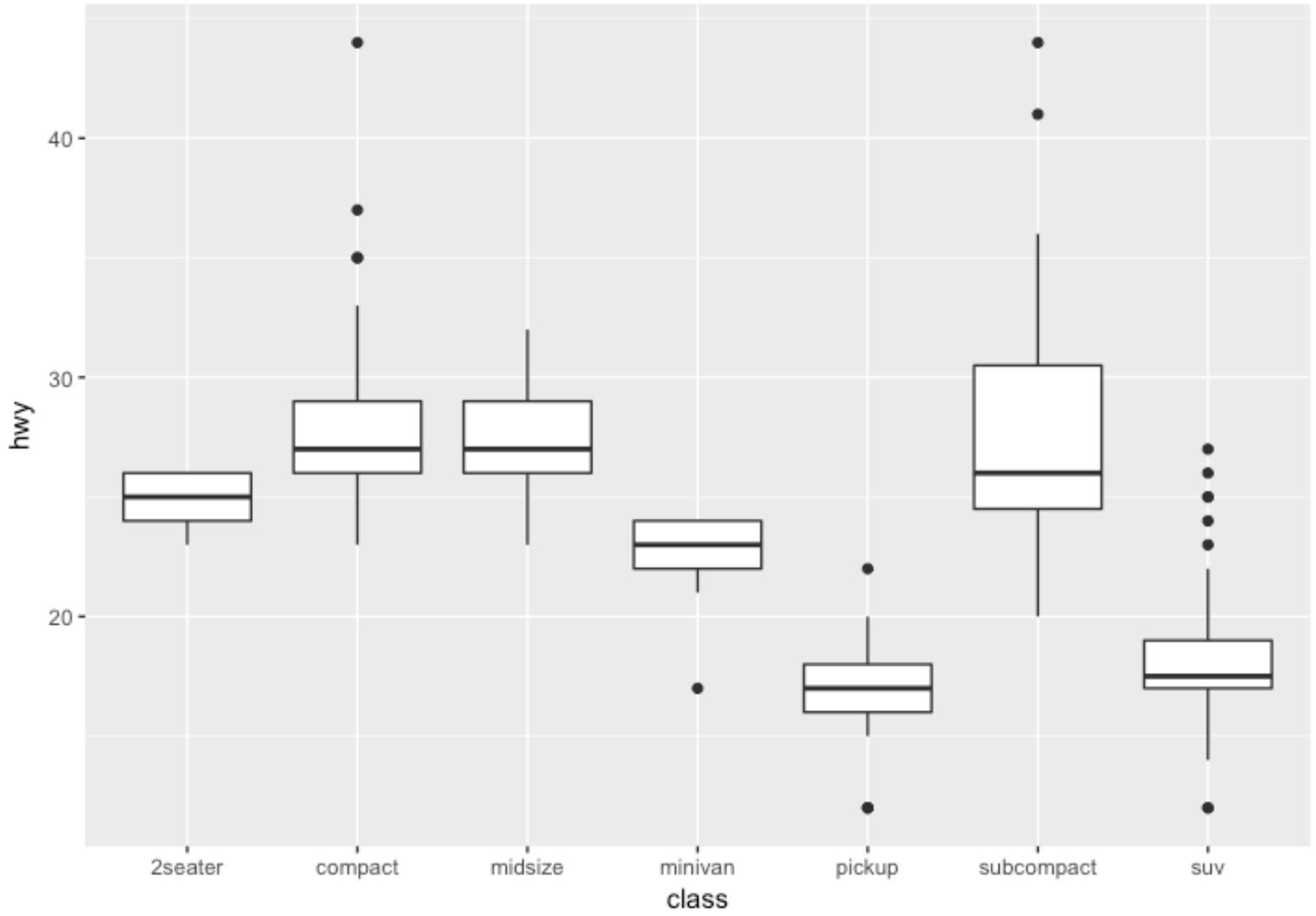


Boxplot

- Display the distribution of a continuous variable broken down by a categorical variable
- Box that stretches from the 25th to 75th percentile a distance known as the interquartile range (IRQ)
- Median in the middle of box
- Points outside more that 1.5 times the IQR from either edge of the box are displayed (outliers)
- Whisker extends to the farthest non-outlier point in the distribution



```
ggplot(data=mpg,mapping=aes(x=class,y=hwy)) +
  geom_boxplot()
```



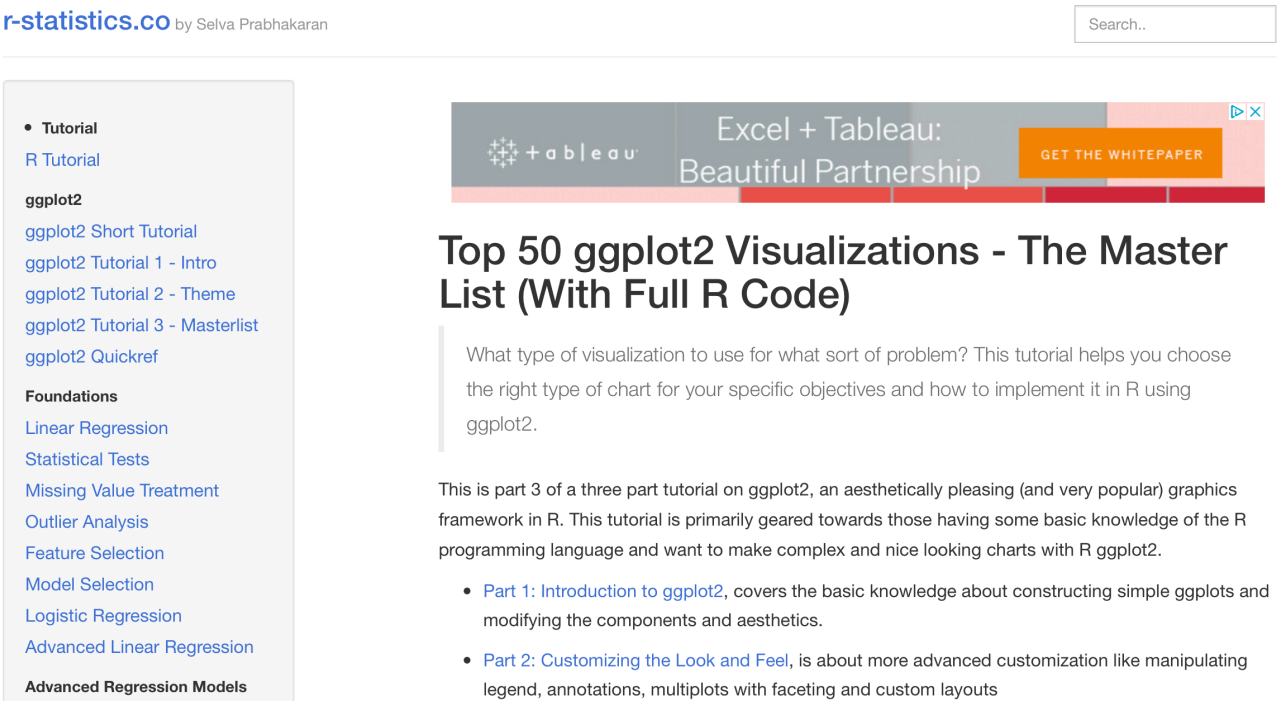
(8) The Layered Grammar of Graphics

- The ggplot2 approach can be summarised by a template
- It can take seven parameters, but usually not all need to be applied (defaults used)
- These seven parameters compose the grammar of graphics

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

Challenge 5.4

- Explore the range of ggplot2 plots at the following link: http://r-statistics.co/Top50-Ggplot2-Visualizations-MasterList-R-Code.html

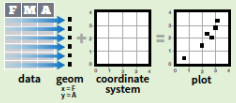


Data Visualization with ggplot2 Cheat Sheet

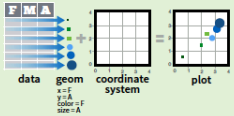
RStudio

Basics

ggplot2 is based on the **grammar of graphics**, the idea that you can build every graph from the same few components: a **data** set, a set of **geoms**—visual marks that represent data points, and a **coordinate system.**



To display data values, map variables in the data set to aesthetic properties of the geom like **size**, **color**, and **x** and **y** locations.



Build a graph with **qplot()** or **ggplot()**

aesthetic mappings · data · geom

qplot(x = cty, y = hwy, color = cyl, data = mpg, geom = "point")
Creates a complete plot with given data, geom, and mappings. Supplies many useful defaults.

ggplot(data = mpg, **aes**(x = cty, y = hwy))
Begins a plot that you finish by adding layers to. No defaults, but provides more control than qplot().

data

```
ggplot(mpg, aes(hwy, cty)) +
 geom_point(aes(color = cyl)) +
 geom_smooth(method ="lm") +
 coord_cartesian() +
 scale_color_gradient() +
 theme_bw()
```

add layers, elements with +
layer = geom + default stat + layer specific mappings
additional elements

Add a new layer to a plot with a **geom_*()** or **stat_*()** function. Each provides a geom, a set of aesthetic mappings, and a default stat and position adjustment.

last_plot()
Returns the last plot

ggsave("plot.png", width = 5, height = 5)
Saves last plot as 5' x 5' file named "plot.png" in working directory. Matches file type to file extension.

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

One Variable

Continuous

a <- ggplot(mpg, aes(hwy))

a + **geom_area(stat = "bin")**
x, y, alpha, color, fill, linetype, size
b + geom_area(aes(y = ..density..), stat = "bin")

a + **geom_density**(kernel = "gaussian")
x, y, alpha, color, fill, linetype, size, weight
b + geom_density(aes(y = ..county..))

a + **geom_dotplot()**
x, y, alpha, color, fill

a + **geom_freqpoly()**
x, y, alpha, color, linetype, size
b + geom_freqpoly(aes(y = ..density..))

a + **geom_histogram**(binwidth = 5)
x, y, alpha, color, fill, linetype, size, weight
b + geom_histogram(aes(y = ..density..))

Discrete

b <- ggplot(mpg, aes(fl))

b + **geom_bar()**
x, alpha, color, fill, linetype, size, weight

Graphical Primitives

c <- ggplot(map, aes(long, lat))

c + **geom_polygon**(aes(group = group))
x, y, alpha, color, fill, linetype, size

d <- ggplot(economics, aes(date, unemploy))

d + **geom_path**(lineend="butt", linejoin="round', linemitre=1)
x, y, alpha, color, linetype, size

d + **geom_ribbon**(aes(ymin=unemploy - 900, ymax=unemploy + 900))
x, ymax, ymin, alpha, color, fill, linetype, size

e <- ggplot(seals, aes(x = long, y = lat))

e + **geom_segment**(aes(
xend = long + delta_long,
yend = lat + delta_lat))
x, xend, y, yend, alpha, color, linetype, size

e + **geom_rect**(aes(xmin = long, ymin = lat, xmax= long + delta_long, ymax = lat + delta_lat))
xmax, xmin, ymax, ymin, alpha, color, fill, linetype, size

Two Variables

Continuous X, Continuous Y

f <- ggplot(mpg, aes(cty, hwy))

f + **geom_blank()**

f + **geom_jitter()**
x, y, alpha, color, fill, shape, size

f + **geom_point()**
x, y, alpha, color, fill, shape, size

f + **geom_quantile()**
x, y, alpha, color, linetype, size, weight

f + **geom_rug**(sides = "bl")
alpha, color, linetype, size

f + **geom_smooth**(model = lm)
x, y, alpha, color, fill, linetype, size, weight

f + **geom_text**(aes(label = cty))
x, y, label, alpha, angle, color, family, fontface, hjust, lineheight, size, vjust

Discrete X, Continuous Y

g <- ggplot(mpg, aes(class, hwy))

g + **geom_bar**(stat = "identity")
x, y, alpha, color, fill, linetype, size, weight

g + **geom_boxplot()**
lower, middle, upper, x, ymax, ymin, alpha, color, fill, linetype, shape, size, weight

g + **geom_dotplot**(binaxis = "y", stackdir = "center")
x, y, alpha, color, fill

g + **geom_violin**(scale = "area")
x, y, alpha, color, fill, linetype, size, weight

Discrete X, Discrete Y

h <- ggplot(diamonds, aes(cut, color))

h + **geom_jitter()**
x, y, alpha, color, fill, shape, size

Continuous Bivariate Distribution

i <- ggplot(movies, aes(year, rating))

i + **geom_bin2d**(binwidth = c(5, 0.5))
xmax, xmin, ymax, ymin, alpha, color, fill, linetype, size, weight

i + **geom_density2d()**
x, y, alpha, colour, linetype, size

i + **geom_hex()**
x, y, alpha, colour, fill size

Continuous Function

j <- ggplot(economics, aes(date, unemploy))

j + **geom_area()**
x, y, alpha, color, fill, linetype, size

j + **geom_line()**
x, y, alpha, color, linetype, size

j + **geom_step**(direction = "hv")
x, y, alpha, color, linetype, size

Visualizing error

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

k + **geom_crossbar**(fatten = 2)
x, y, ymax, ymin, alpha, color, fill, linetype, size

k + **geom_errorbar()**
x, ymax, ymin, alpha, color, linetype, size, width (also **geom_errorbarh()**)

k + **geom_linerange()**
x, ymin, ymax, alpha, color, linetype, size

k + **geom_pointrange()**
x, y, ymin, ymax, alpha, color, fill, linetype, shape, size

Maps

data <- data.frame(murder = USArrests$Murder, state = tolower(rownames(USArrests)))
map <- map_data("state")
l <- ggplot(data, aes(fill = murder))

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

Three Variables

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

m + **geom_contour**(aes(z = z))
x, y, z, alpha, colour, linetype, size, weight

m + **geom_raster**(aes(fill = z), hjust=0.5, vjust=0.5, interpolate=FALSE)
x, y, alpha, fill

m + **geom_tile**(aes(fill = z))
x, y, alpha, color, fill, linetype, size

Learn more at **docs.ggplot2.org** • ggplot2 0.9.3.1 • Updated: 3/15