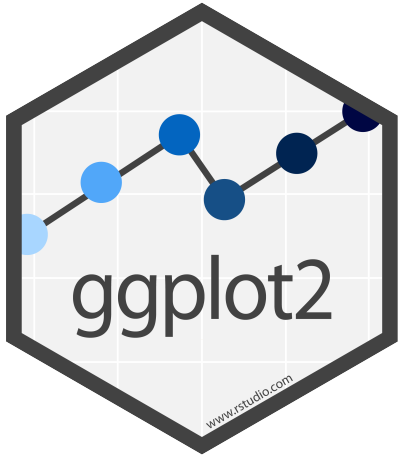
Data Visualization

JSC 370: Data Science II

Background



This lecture provides an introduction to ggplot2, an R package that provides vastly better graphics options than R's default plots, histrograms, etc.

This section is based on chapter 3 of "R for Data Science"

Background

`ggplot2` is part of the Tidyverse. The tidyverse is..."an opinionated collection of R packages designed for data science. All packages share an underlying design philosophy, grammar, and data structures." (https://www.tidyverse.org/)

```
library(tidyverse)
library(data.table)
```

ggplot2

`ggplot2` is designed on the principle of adding layers.

ggplot2

- With ggplot2 a plot is initiated with the function `ggplot()`
- The first argument of `ggplot()` is the dataset to use in the graph
- Layers are added to `ggplot()` with `+`
- Layers include `geom` functions such as point, lines, etc
- Each `geom` function takes a `mapping` argument, which is always paired with `aes()`
- The `aes()` mapping takes the x and y axes of the plot

```
ggplot(data = data) +
    geom_function(mapping = aes(mappings))
```

Data frames

ggplot2 is designed to work with data frames. A data frame is a list of vectors (of equal length). The vectors can be of any type. For example, the following variable df is a data frame containing three vectors 'index', 'animal', and 'owned'.

```
index = c(1, 2, 3)
animal = c("cat", "dog", "rabbit")
owned = c(TRUE, FALSE, FALSE)
df = data.frame(index, animal, owned)       # df is a data frame
print(df)
```

```
##   index animal owned
## 1     1    cat  TRUE
## 2     2    dog FALSE
## 3     3 rabbit FALSE
```

The top line of the table is called the header and contains the variable names.

Here is a data frame that is part of ggplot2:

```
mpg
```

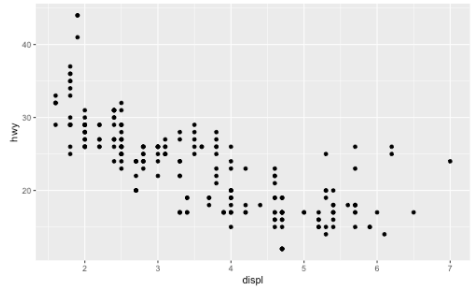
```
## # A tibble: 234 × 11
##    manufacturer model      displ  year   cyl trans drv     cty   hwy fl    class
##    <chr>        <chr>      <dbl> <int> <int> <chr> <chr> <int> <int> <chr> <chr>
##  1 audi         a4           1.8  1999     4 auto… f        18    29 p     comp…
##  2 audi         a4           1.8  1999     4 manu… f        21    29 p     comp…
##  3 audi         a4           2    2008     4 manu… f        20    31 p     comp…
##  4 audi         a4           2    2008     4 auto… f        21    30 p     comp…
##  5 audi         a4           2.8  1999     6 auto… f        16    26 p     comp…
##  6 audi         a4           2.8  1999     6 manu… f        18    26 p     comp…
##  7 audi         a4           3.1  2008     6 auto… f        18    27 p     comp…
##  8 audi         a4 quattro   1.8  1999     4 manu… 4        18    26 p     comp…
##  9 audi         a4 quattro   1.8  1999     4 auto… 4        16    25 p     comp…
## 10 audi         a4 quattro   2    2008     4 manu… 4        20    28 p     comp…
## # … with 224 more rows
```

```
head(mpg)
```

```
## # A tibble: 6 × 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     compa…
## 2 audi         a4      1.8  1999     4 manual(m5) f        21    29 p     compa…
## 3 audi         a4      2    2008     4 manual(m6) f        20    31 p     compa…
## 4 audi         a4      2    2008     4 auto(av)   f        21    30 p     compa…
## 5 audi         a4      2.8  1999     6 auto(l5)   f        16    26 p     compa…
## 6 audi         a4      2.8  1999     6 manual(m5) f        18    26 p     compa…
```

Basic plot 1

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



- As expected, we see that cars with big engines use more fuel.

- ggplot2 begins a plot with the function ggplot(), which creates a coordinate system that you can add layers to. The first argument of ggplot() is the dataset to use in the graph. You complete the graph by adding one or more layers to ggplot().

Basic plot 2

- geom_point() adds a layer of points to your plot, to create a scatterplot.

- ggplot2 comes with many geom functions that each add a different type of layer to a plot.

- Each geom function in ggplot2 takes a mapping argument.

- This defines how variables in your dataset are mapped to visual properties.

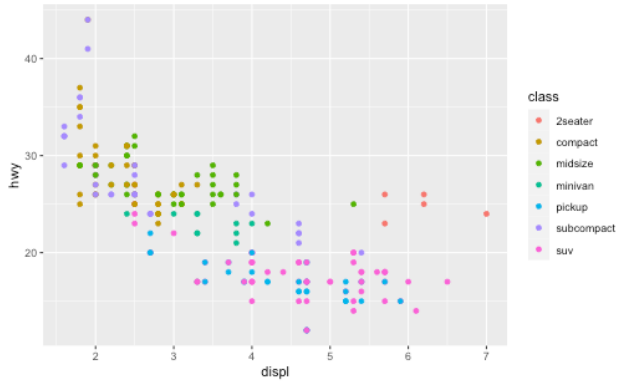
- The mapping argument is always paired with aes(), and the x and y arguments of aes() specify which variables to map to the x and y axes. ggplot2 looks for the mapped variables in the data argument, in this case, mpg.

- One common problem when creating ggplot2 graphics is to put the +

Coloring by a variable - using aesthetics

You can convey information about your data by mapping the aesthetics in your plot to the variables in your dataset. For example, you can map the colors of your points to the class variable to reveal the class of each car. ggpot chooses colors, and adds a legend, automatically.

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

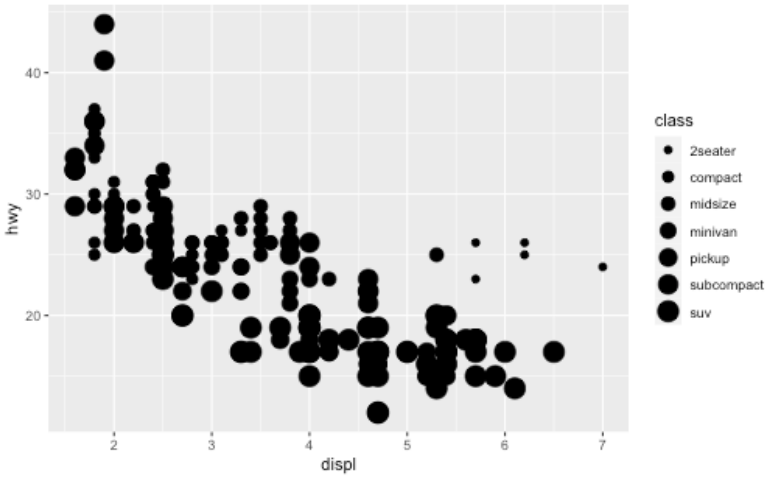


Determining point size using a variable

You can map point size to a variable as well:

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

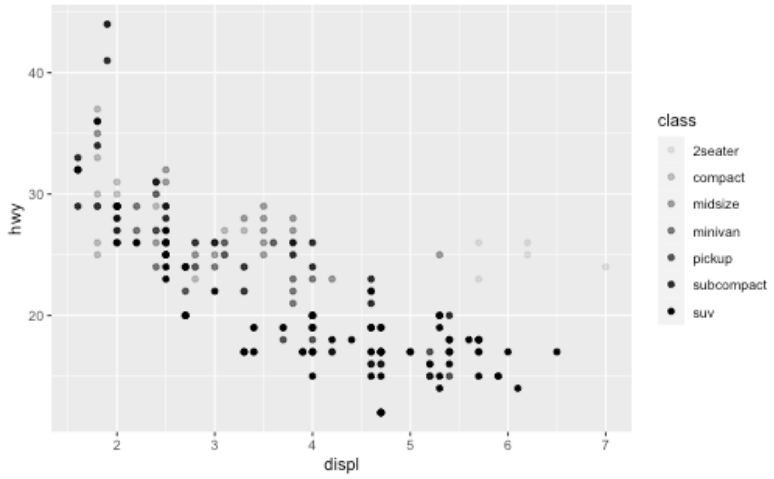
```
## Warning: Using size for a discrete variable is not advised.
```



Controlling point transparency using the "alpha" aesthetic

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

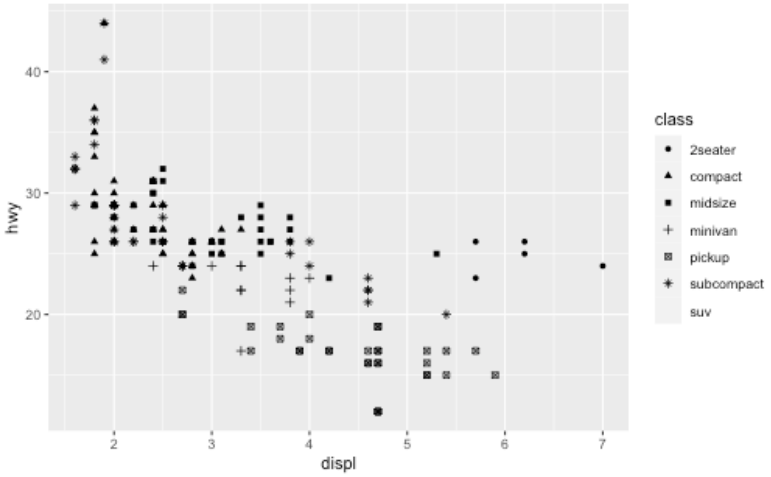
```
## Warning: Using alpha for a discrete variable is not advised.
```



Controlling point shape

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

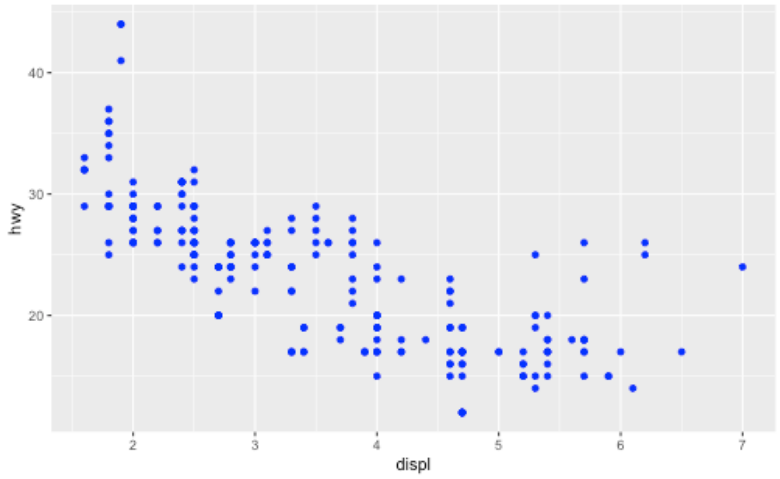
```
## Warning: The shape palette can deal with a maximum of 6 discrete values because
## more than 6 becomes difficult to discriminate; you have 7. Consider
## specifying shapes manually if you must have them.

## Warning: Removed 62 rows containing missing values (geom_point).
```



Manual control of aesthetics

To control aesthetics manually, set the aesthetic by name as an argument of your geom function; i.e. it goes outside of aes().

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



Summary of aesthetics

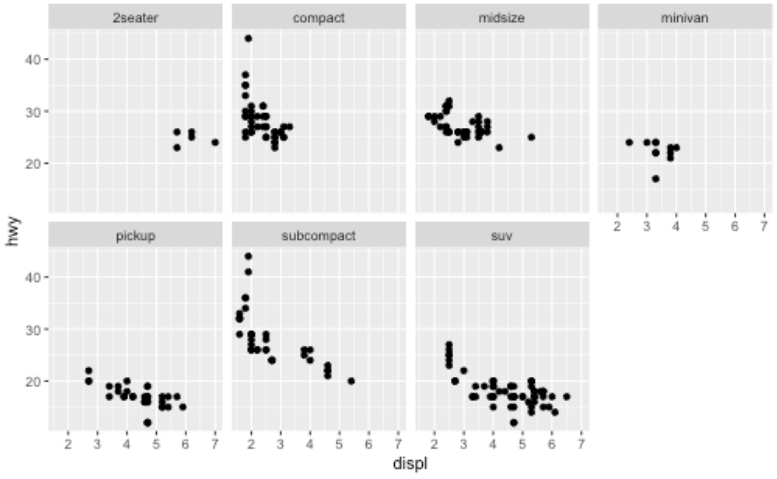
The various aesthetics...

code	description
x	position on x-axis
y	position on y-axis
shape	shape
color	color of element borders
fill	color inside of elements
size	size
alpha	transparency
linetype	type of line

Facets 1

Facets are particularly useful for categorical variables...

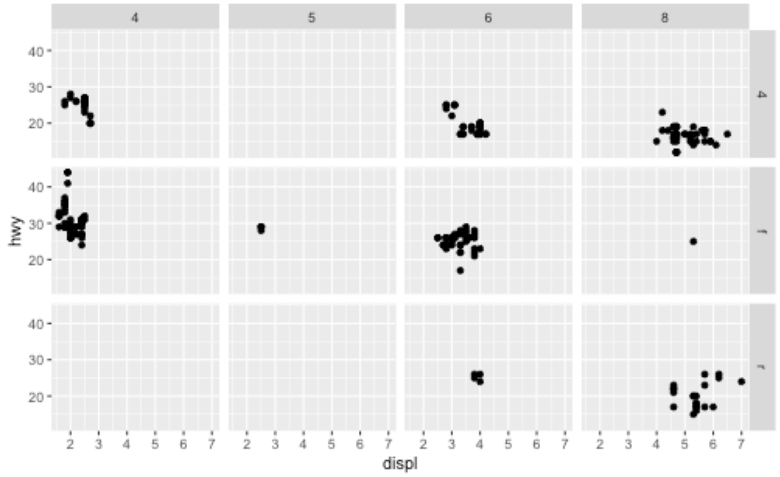
```
ggplot(data = mpg) +
  geom_point(mapping = aes(x = displ, y = hwy)) +
  facet_wrap(~ class, nrow = 2)
```



Facets 2

Or you can facet on two variables...

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



Geometric objects 1

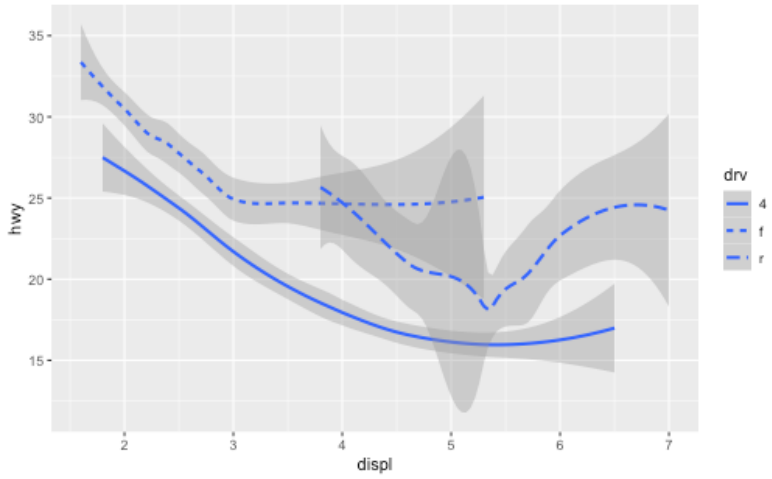
Geometric objects are used to control the type of plot you draw. So far we have used scatterplots (via 'geom_point'). But now let's try plotting a smoothed line fitted to the data (and note how we do side-by-side plots)

```
library(cowplot)
scatterplot <- ggplot(data = mpg) + geom_point(mapping = aes(x = displ, y = hwy))
lineplot <- ggplot(data = mpg) + geom_smooth(mapping = aes(x = displ, y = hwy))
plot_grid(scatterplot, lineplot, labels = "AUTO")
```

Geometric objects 2

Note that not every aesthetic works with every geom function. But now there are some new ones we can use.

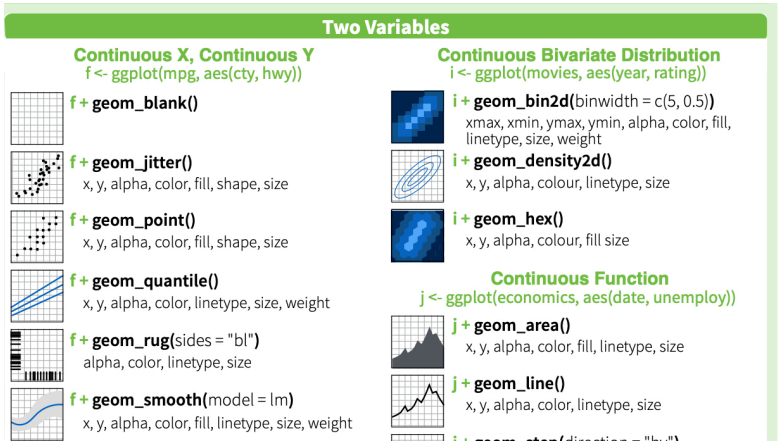
```
ggplot(data = mpg) +
  geom_smooth(mapping = aes(x = displ, y = hwy, linetype = drv))
```



Geoms - reference

ggplot2 provides over 40 geoms, and extension packages provide even more (see https://ggplot2.tidyverse.org/reference/ for a sampling).

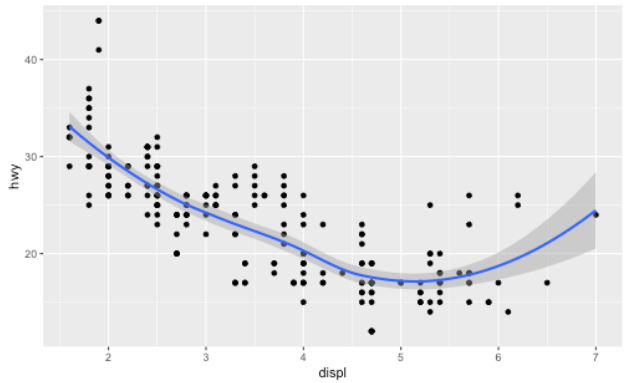
The best way to get a comprehensive overview is the ggplot2 cheatsheet, which you can find at https://rstudio.com/wp-content/uploads/2015/03/ggplot2-cheatsheet.pdf



Multiple geoms 1

To display multiple geoms in the same plot, add multiple geom functions to ggplot():

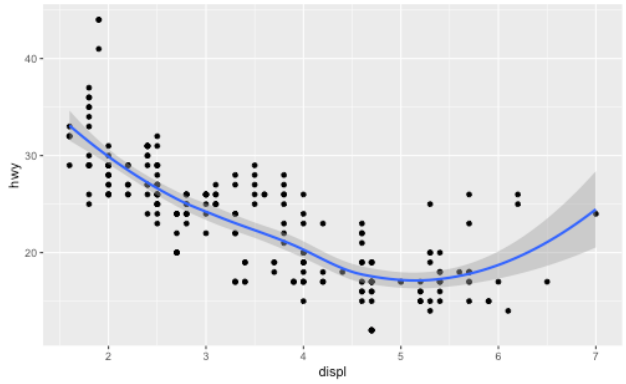
```
ggplot(data = mpg) +
  geom_point(mapping = aes(x = displ, y = hwy)) +
  geom_smooth(mapping = aes(x = displ, y = hwy))
```



Multiple geoms 2

We can avoid repetition of aethetics by passing a set of mappings to ggplot(). ggplot2 will treat these mappings as global mappings that apply to each geom in the graph.

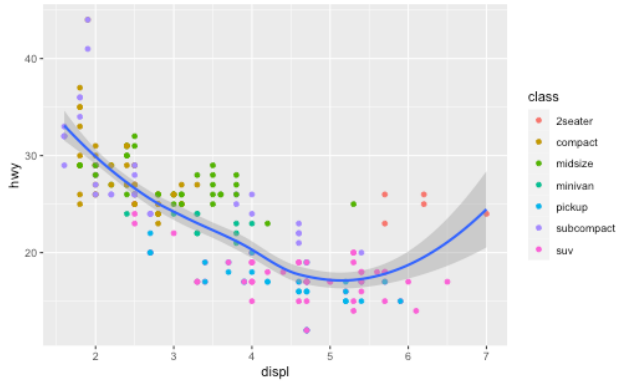
```
ggplot(data = mpg, mapping = aes(x = displ, y = hwy)) +
  geom_point() +
  geom_smooth()
```



Multiple geoms 3

If you place mappings in a geom function, ggplot2 will use these mappings to extend or overwrite the global mappings for that layer only. This makes it possible to display different aesthetics in different layers.

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



Multiple geoms 4

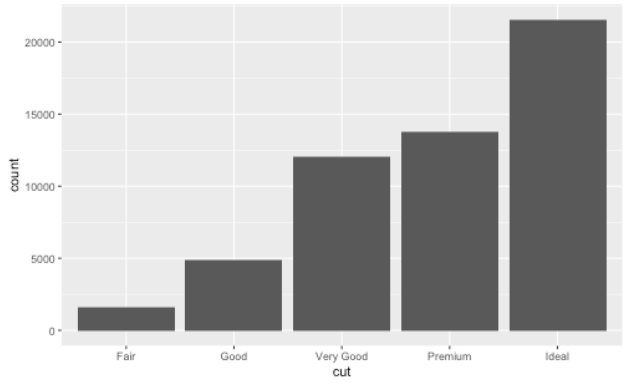
You can use the same idea to specify different data for each layer. Here, our smooth line displays just a subset of the mpg dataset, the subcompact cars. The local data argument in geom_smooth() overrides the global data argument in ggplot() for that layer only.

```
ggplot(data = mpg, mapping = aes(x = displ, y = hwy)) +
  geom_point(mapping = aes(color = class)) +
  geom_smooth(data = filter(mpg, class == "subcompact"), se = FALSE)
```

Statistical transformationas - e.g. Bar charts

This example uses the built-in dataset "diamonds".

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

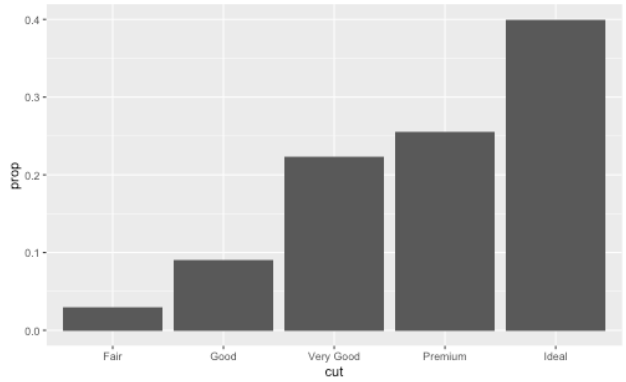


The algorithm uses a built-in statistical transformation, called a "stat", to calculate the counts.

Bar charts 2

You can over-ride the stat a geom uses to construct its plot. e.g., if we want to plpot proportions, rather than counts:

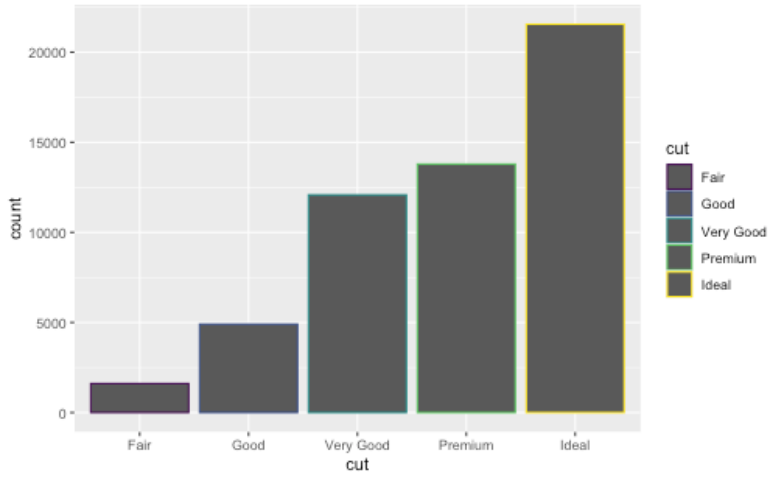
```
ggplot(data = diamonds) +
  geom_bar(mapping = aes(x = cut, y = stat(prop), group = 1))
```



Coloring barcharts

You can color a bar chart using either the color aesthetic, or, more usefully, fill:

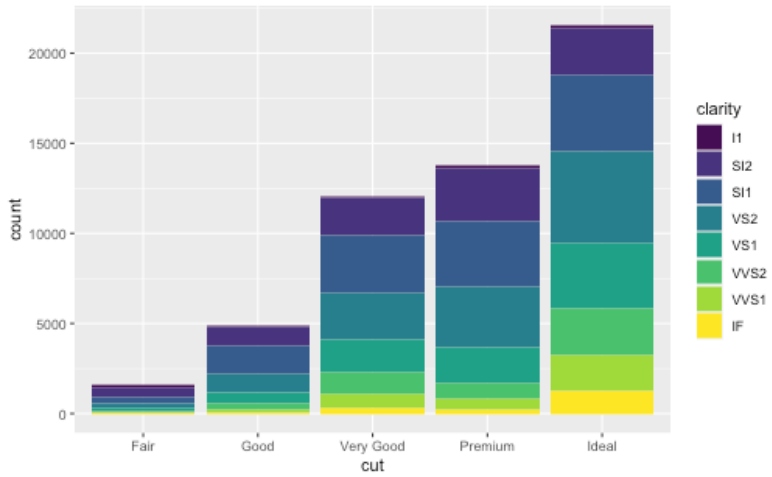
```
ggplot(data = diamonds) +
  geom_bar(mapping = aes(x = cut, colour = cut))
```



Coloring barcharts

More interestingly, you can fill by another variable (here, 'clarity')

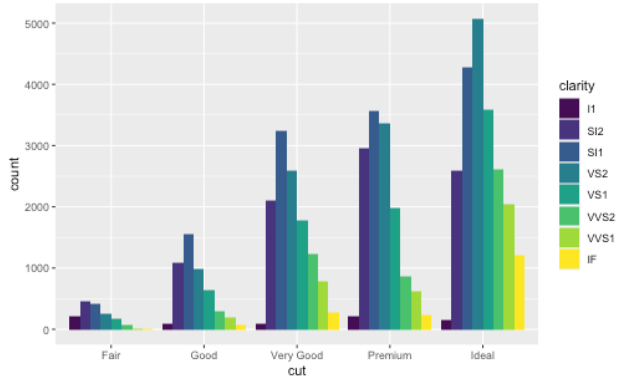
```
ggplot(data = diamonds) +
  geom_bar(mapping = aes(x = cut, fill = clarity))
```



Coloring barcharts

position = "dodge" places overlapping objects directly beside one another. This makes it easier to compare individual values.

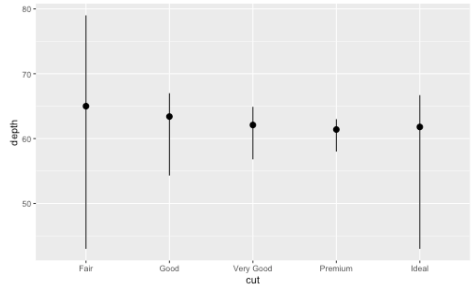
```
ggplot(data = diamonds) +
  geom_bar(mapping = aes(x = cut, fill = clarity), position = "dodge")
```



Statistical transformations - another example

You might want to draw greater attention to the statistical transformation in your code. For example, you might use stat_summary(), which summarizes the y values for each unique x value, to draw attention to the summary that you're computing:

```
ggplot(data = diamonds) +
  stat_summary(mapping = aes(x = cut, y = depth), fun.min = min,fun.max = max,fun = median)
```



Position adjustments

An option that can be very useful is position = "jitter". This adds a small amount of random noise to each point. This spreads out points that might otherwise be overlapping. e.g.,

```
nojitter <- ggplot(data = mpg) +
  geom_point(mapping = aes(x = displ, y = hwy))
jitter <- ggplot(data = mpg) +
  geom_point(mapping = aes(x = displ, y = hwy), position = "jitter")

plot_grid(nojitter, jitter, labels = "AUTO")
```

Coordinate systems

Coordinate systems are one of the more complicated corners of ggplot. To start with something simple, here's how to flip axes:

```
unflipped <- ggplot(data = mpg, mapping = aes(x = class, y = hwy)) +
  geom_boxplot()
flipped <- ggplot(data = mpg, mapping = aes(x = class, y = hwy)) +
  geom_boxplot() +
  coord_flip()
plot_grid(unflipped, flipped, labels = "AUTO")
```

A Great reference

A great (comprehensive) reference for everything you can do with ggplot2 is the R Graphics Cookbook:

https://r-graphics.org/

Reminder - the ggplot2 cheatsheet

A briefer summary can be found here:

https://rstudio.com/wp-content/uploads/2015/03/ggplot2-cheatsheet.pdf

Rstudio has a variety of other great Cheatsheets.

Finally, file under "useless but cool"

ggpattern - is a library for adding pattern fills to histograms. e.g.,