Graphics with ggplot2

David Gerard

2020-10-20

Learning Objectives

- ggplot2.
- Basics of data visualization.
- Chapter 3 RDS.
- Data Visualization Cheat Sheet.
- Ggplot2 Overview.

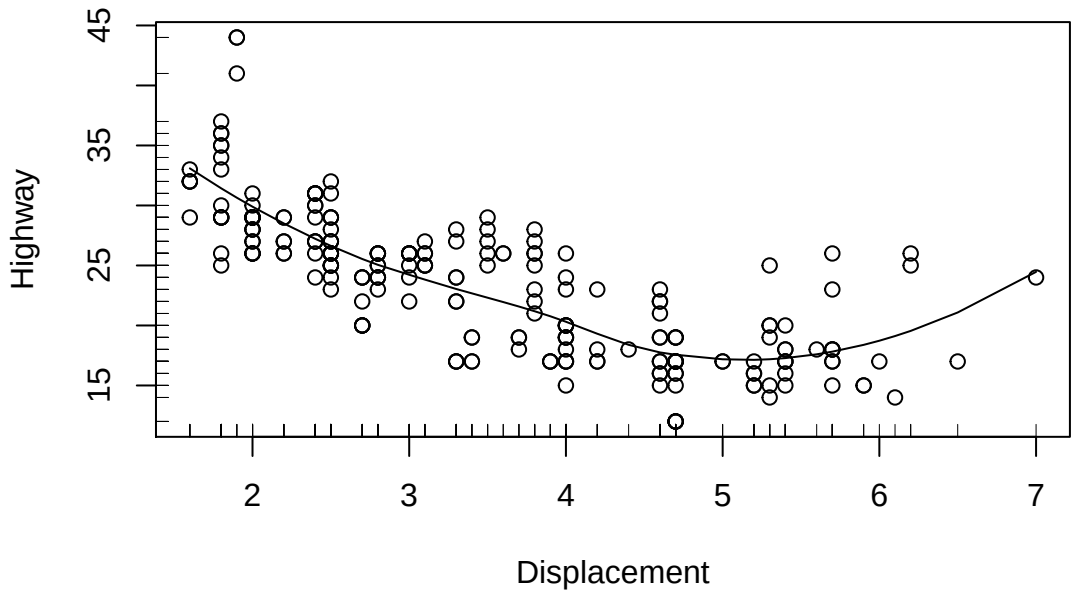
Background

- The three main plotting systems in R are base R, lattice, and ggplot2 (from the tidyverse) (with base R-like syntax implemented in the `qplot()` function).
- In base R: start with a blank plot and build on it by adding elements — the "artist's canvas".
- In lattice: One function, specify everything at once.
- ggplot2: A "grammar of graphics". You map from variables to aesthetic attributes (size, color, shape) of geometric objects (lines, bars, points).
- Let's look at the three ways to make a scatterplot with a loess smoother and a rug plot.
  - Load data and packages:

```
library(ggplot2)
library(lattice)
data("mpg")
```

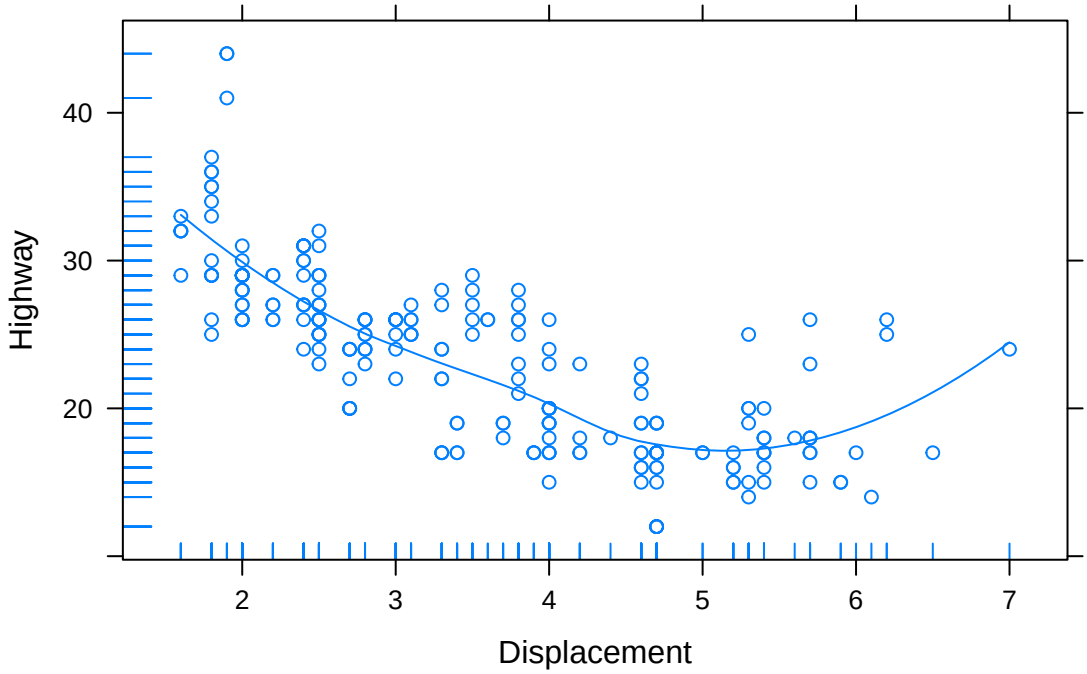
- base R: Everything is a separate non-connected function

```r
## Make a scatterplot
plot(x = mpg$displ, y = mpg$hwy, xlab = "Displacement", ylab = "Highway")

## Add loess curve
lout <- loess(hwy ~ displ, data = mpg)
ord_x <- order(lout$x)
lines(lout$x[ord_x], lout$fitted[ord_x])

## Add two rug plots
rug(mpg$displ, side = 1)
rug(mpg$hwy, side = 2)
```



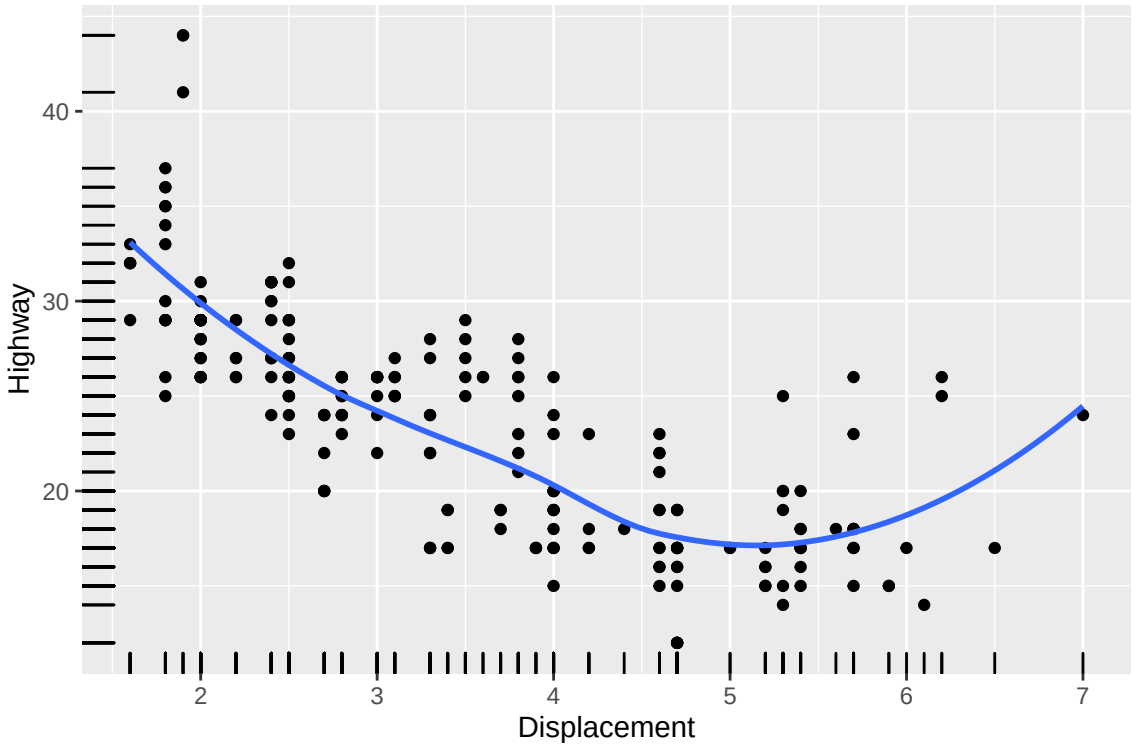
– lattice: Everything is done at once within some mega-function:

```
xyplot(hwy ~ displ, data = mpg,
       xlab = "Displacement", ylab = "Highway",
       panel = function(x, y) {
         panel.xyplot(x = x, y = y)                    ## Add scatterplot
         panel.loess(x = x, y = y, span = 0.75,
                     family = "gaussian", degree = 2) ## Add loess curve
         panel.rug(x = x, y = y)                       ## Add rug plot
       })
```



- ggplot2: Many connected functions

```
ggplot(mpg, mapping = aes(x = displ, y = hwy)) + ## set aesthetic maps
  geom_point() +                                 ## apply aesthetic maps to
                                                 ##     point objects
  geom_smooth(se = FALSE, method = loess) +      ## apply to smooth objects
  geom_rug() +                                   ## apply to rug objects
  xlab("Displacement") +
  ylab("Highway")
```



- lattice is the hardest to learn and the least used.
- base R is very flexible, but harder to use for most tasks.
- ggplot2 is very easy to use for most tasks, but very hard to use when you want to do something weird.

ggplot2 Foundation

- ggplot consists of the following elements:
  - **Data Frame** (`data`): Contains the variables that you want to plot.
  - Transformation (`stat`): Transforms data into a new data frame (e.g. by calculating the number of observational units in each bin for a histogram).
  - **Aesthetic Mapping** (`mapping`): Assign variables to be aesthetics (e.g. x-coordinate, y-coordinate, color, shape, transparency, line thickness, line type) of geometric objects.
  - **Geometric Objects** (`geom`): The type of plot you want (e.g. scatterplot, line plot, histogram, barplot)
  - Position Adjustment (`position`): Jitter objects, overlap histogram bins, etc..
  - **Facets**: Create multiple plots, one for each value of a categorical variable.
  - Coordinate systems: Cartesian/polar
- You can get most of your plotting done by understanding the basics of data frames, aesthetic mappings,

geoms, and facets (the other aspects of the grammar of graphics are less often useful to modify).

- `ggplot()` creates a blank `gg` objects. It has two main arguments:
  - `data`: Must be a data frame. Contains the variables that you want to map onto aesthetics.
  - `mapping`: A definition for which variables map onto which aesthetics. You almost always use the `aes()` function to perform this mapping.
  - What is the output of the following?

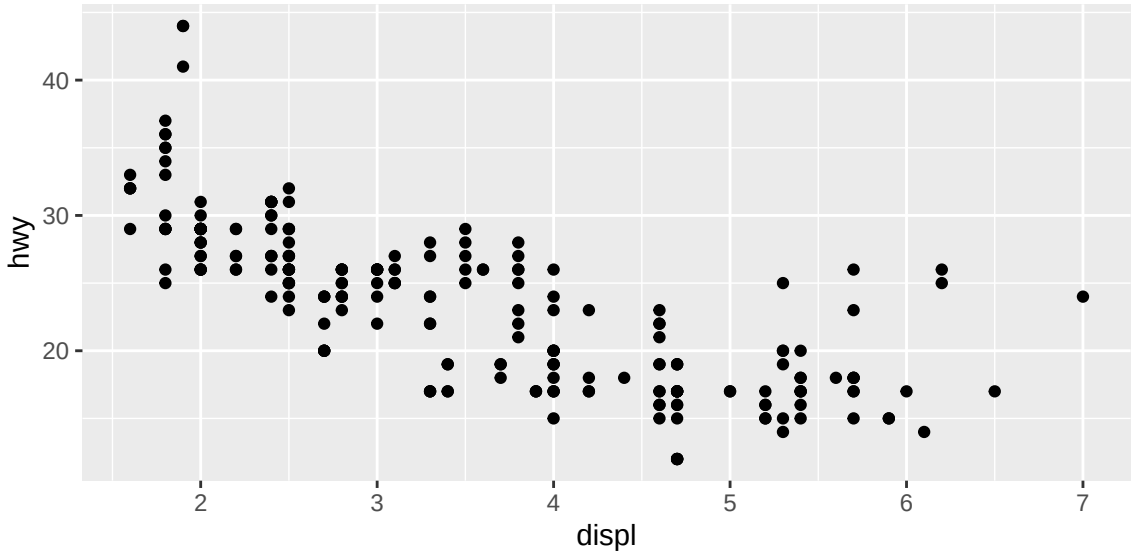
```
ggplot(data = mpg, mapping = aes(x = displ, y = hwy))
```

- `ggplot()` is almost always followed by a `+` and then a `geom` argument to specify which geometric objects receive the aesthetic mapping. We will now go through many different types of geoms and the situations where they are useful:
- **Always make sure the + is at the end of a line and not at the beginning of a line.**
- To get a list of the possible aesthetic mappings of a geom, look at the help page.

Comparing Two Quantitative Variables

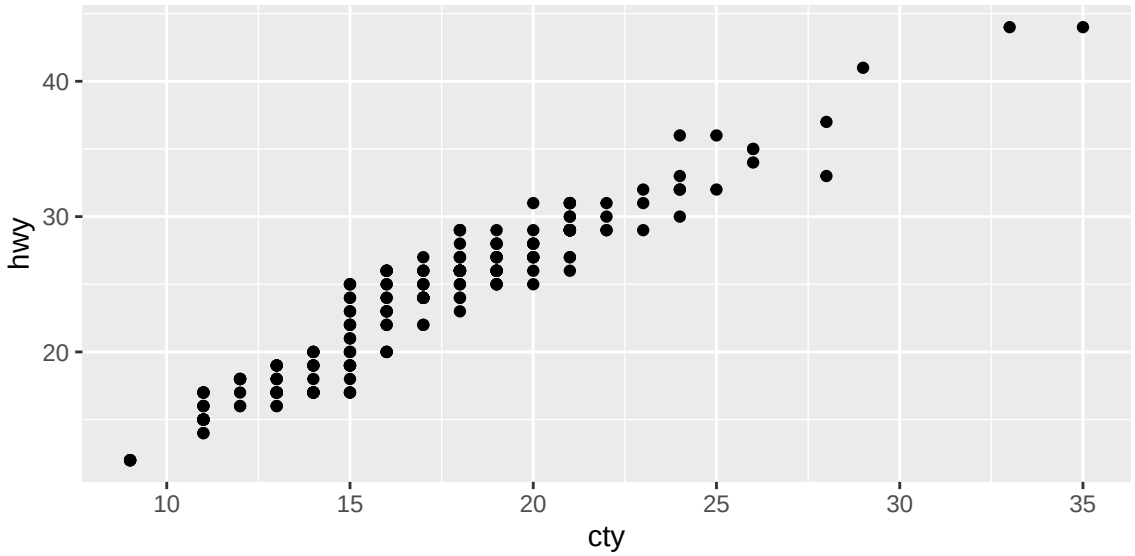
- A scatterplot is used to explore the relationship between two **quantitative** variables.
- "Quantitative" means that arithmetic operations ($+/-/\times/\div$) make sense on them.
- E.g. `hwy` and `displ` are two quantitative variables.
- Not all numeric variables are quantitative (e.g. phone numbers).
- Use `geom_point()` to create a scatterplot:

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



- `displ` is mapped onto the `x` aesthetic and `hwy` is mapped onto the `y` aesthetic.
- The above indicates a **negative** relationship between `hwy` and `displ`
- The following indicates a **positive** relationship between `hwy` and `cty`

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

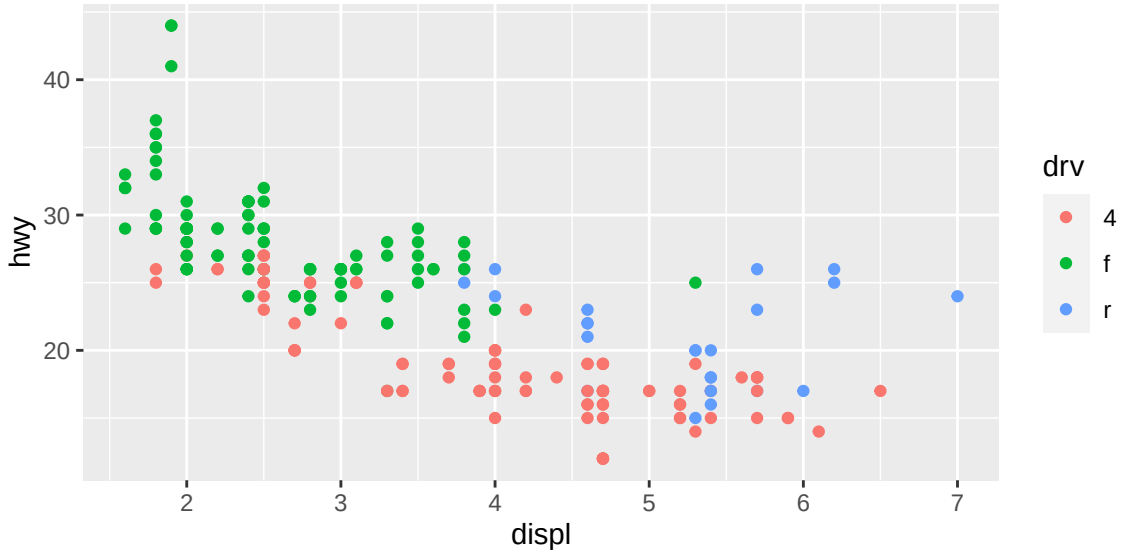


- **Exercise**: Create a scatterplot of engine displacement vs city miles per gallon. Think carefully about which variable should be on the x-axis.

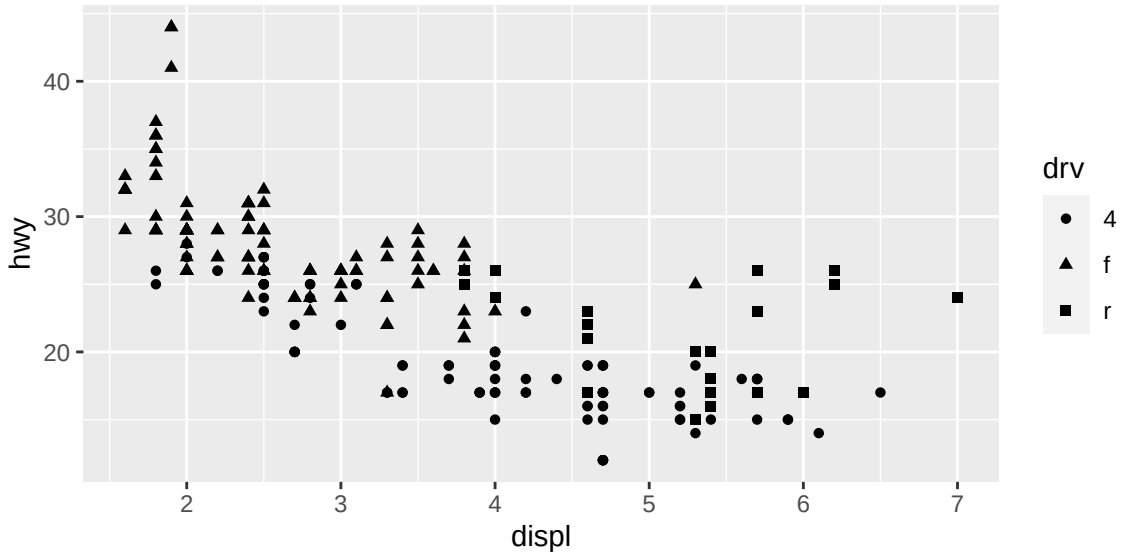
Annotating by a categorical variable

- We might be interested in the association of two quantitative variables at different values of a categorical variable.
- A **categorical** variable places observational units into different groups or categories based on the level of the categorical variable.
- We often map categorical variables to the `color` (better) or `shape` (worse) aesthetics:

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



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



- **Exercise**: What happens when you map a categorical variable to the `size` or `alpha` aesthetics?
- **Exercise**: Consider the `penguins` dataset that comes from the `palmerpenguins` package:

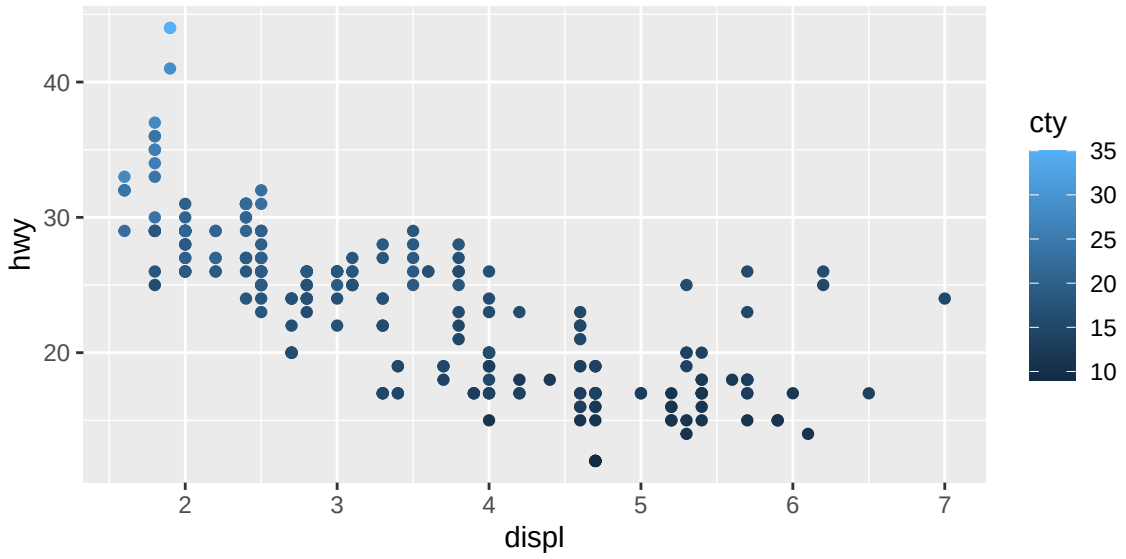
```
# install.packages("palmerpenguins")
library(palmerpenguins)
data("penguins")
```

Make a plot of bill length vs bill depth, color coding by the species of penguin. What do you notice?

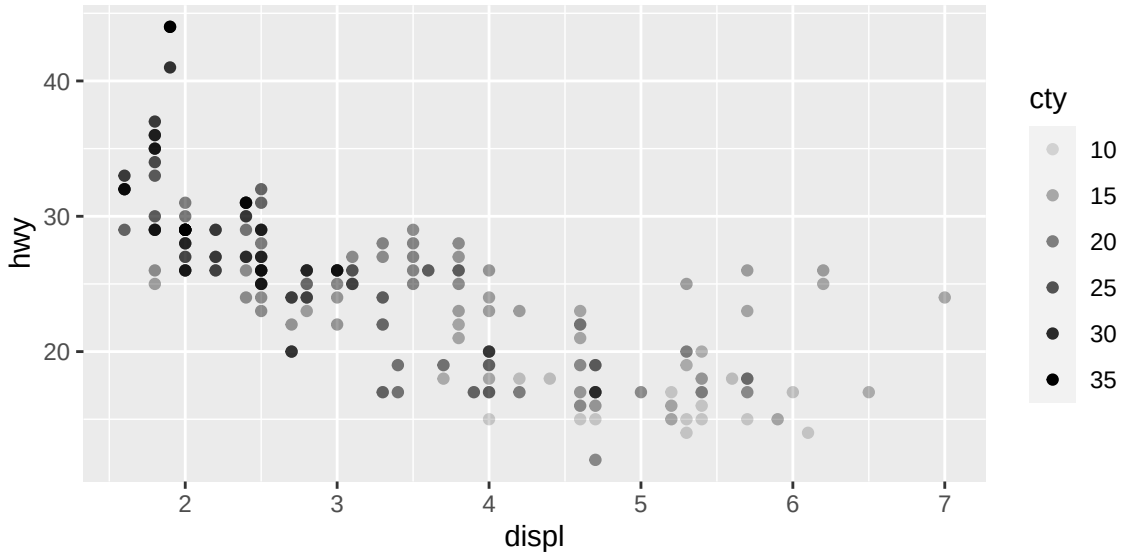
Annotating by a quantitative variable

- If we want to look at the association between two quantitative variables at different levels of a third quantitative variable, we can map that third quantitative variable to the `color` (good), `alpha` (good), or size (worst) aesthetic.

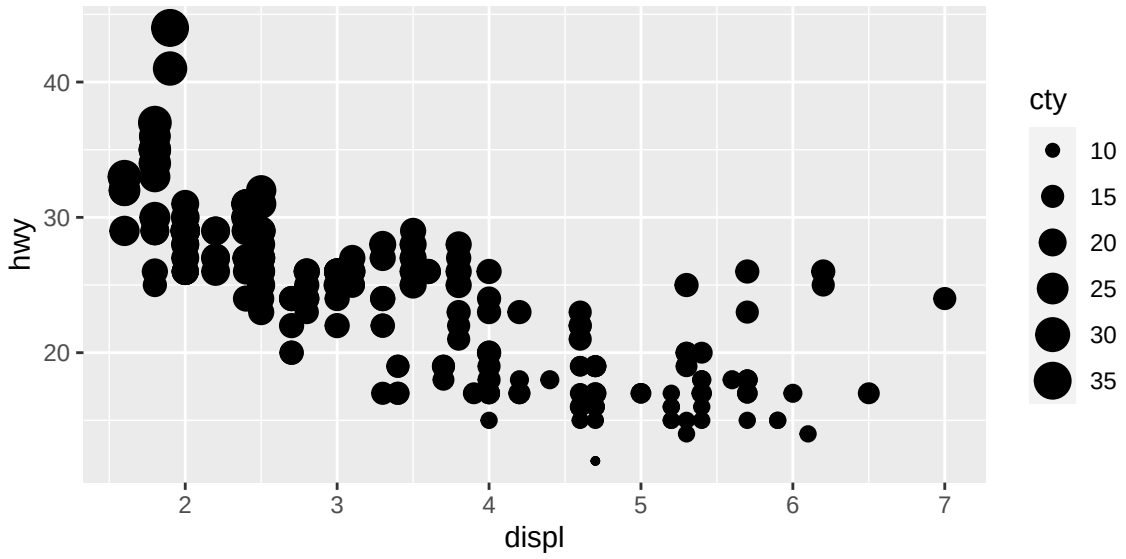
```
ggplot(data = mpg, mapping = aes(x = displ, y = hwy, color = cty)) +
          geom_point()
```




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

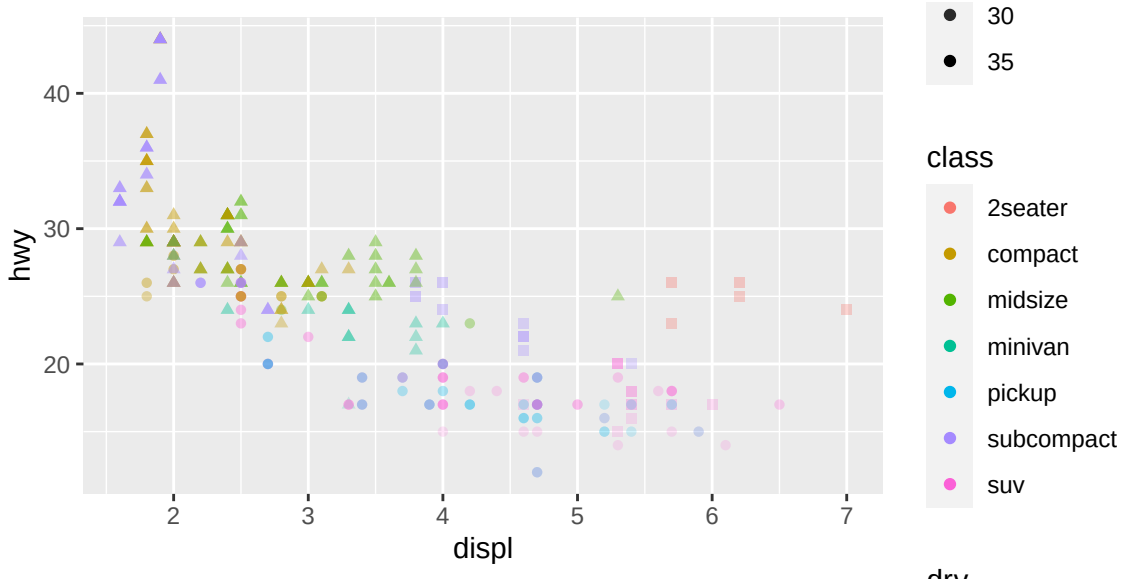



```
ggplot(data = mpg, mapping = aes(x = displ, y = hwy, size = cty)) +
              geom_point()
```



- **Exercise**: What happens when we map a quantitative variable to the `shape` aesthetic?
- We can map multiple variables to multiple aesthetics at the same time, but the resulting plot might get too complicated.

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

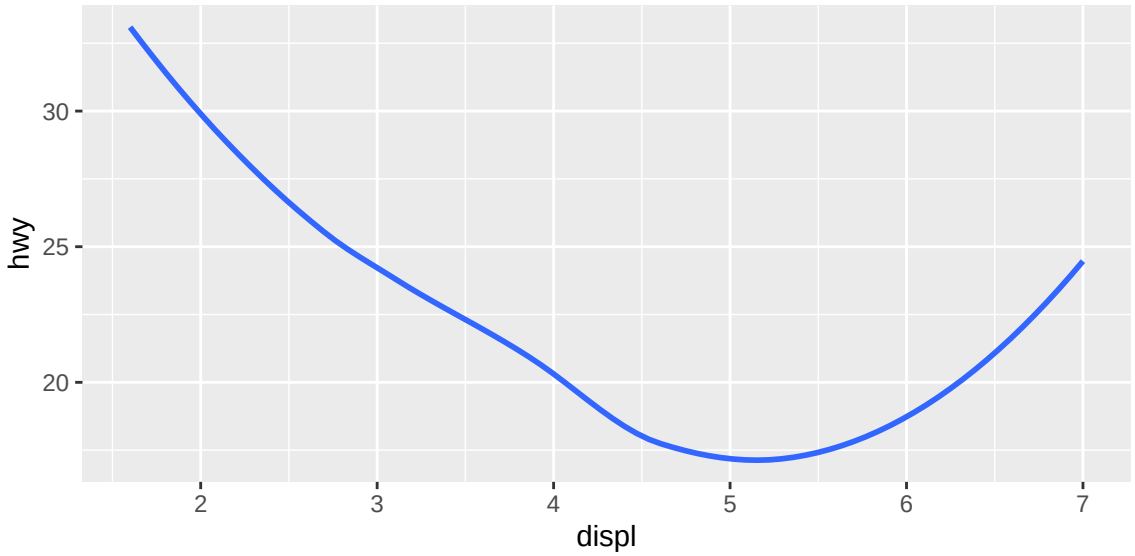


- **Exercise**: Using the `penguins` dataset, plot bill length vs bill depth, annotating by flipper length and body mass and species. Do you notice anything?

Smoothing

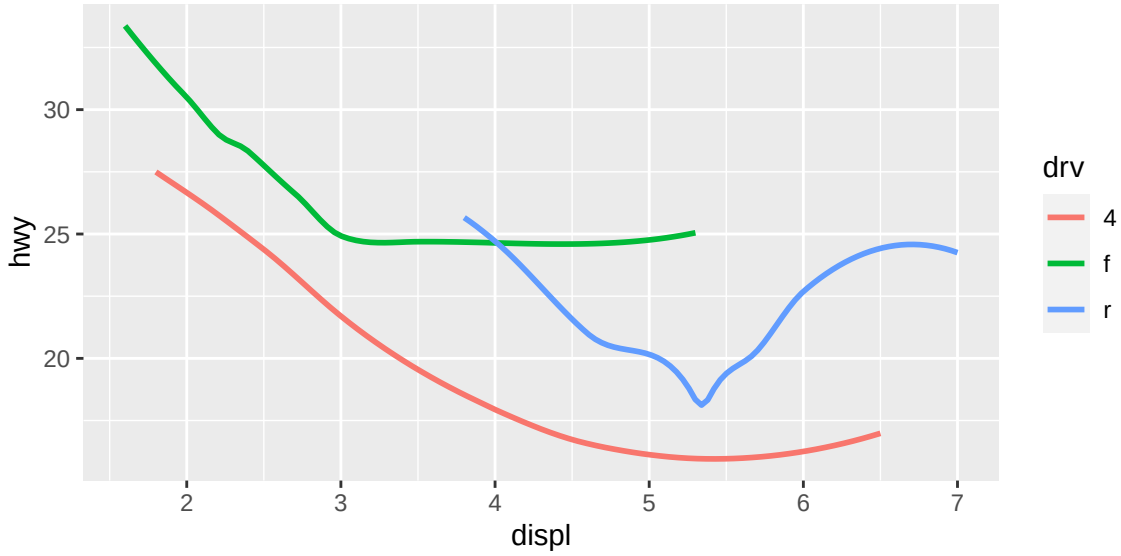
- `geom_smooth()` will create trend lines (either non-parametric or parametric) and plot these.

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

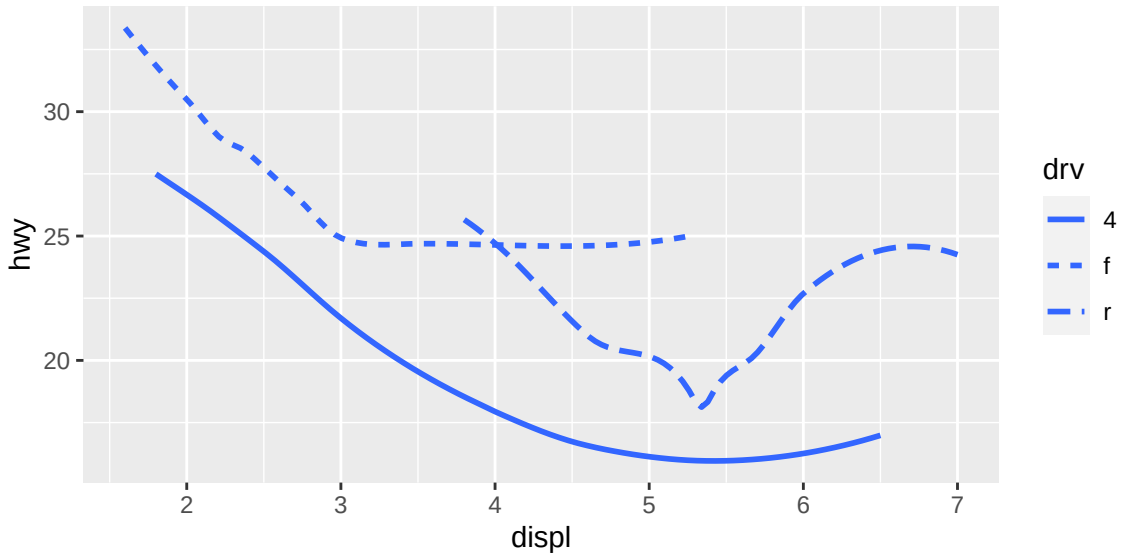


- You can pass categorical variables to the `color` (best) or `linetype` (good) aesthetics

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

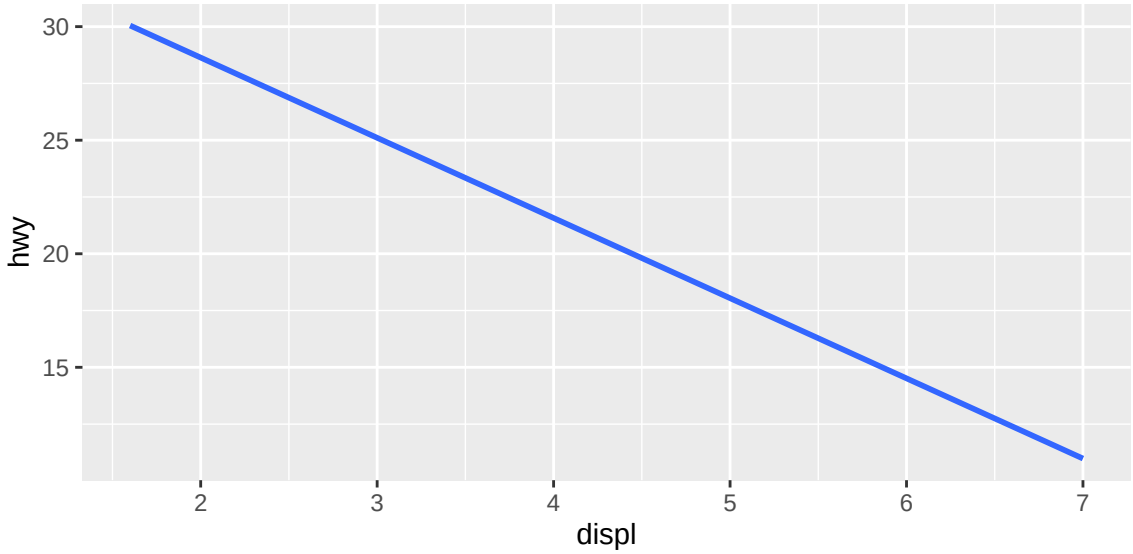



```
ggplot(data = mpg, mapping = aes(x = displ, y = hwy, linetype = drv)) +
          geom_smooth(se = FALSE)
```



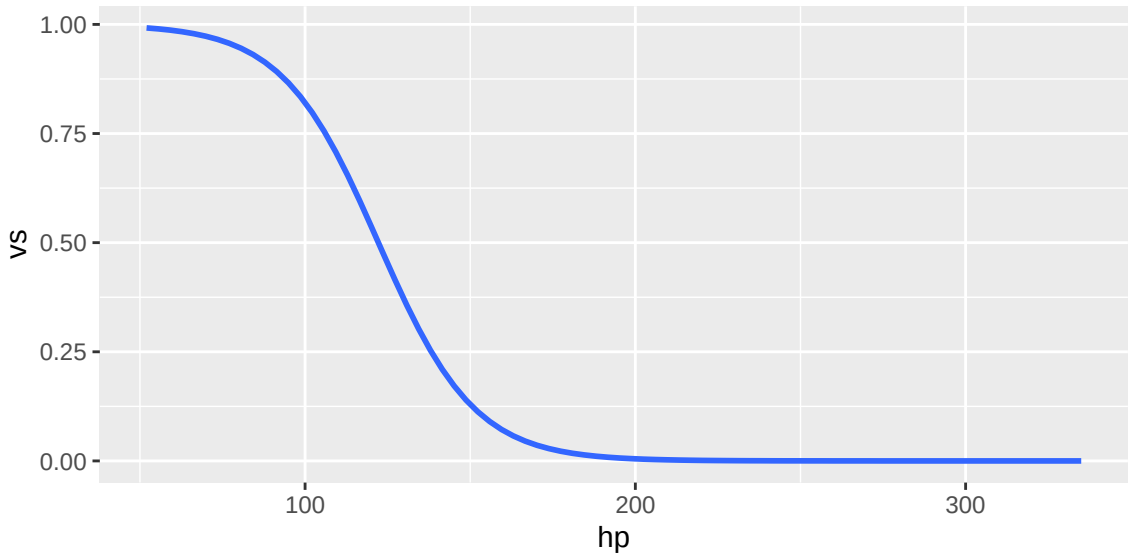
- The default for `geom_smooth()` is to calculate a loess smoother, but other options are available. For example, we can plot the OLS fit by changing the `method` argument.

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



- If you have binary response data, you can plot the fitted logistic curves by

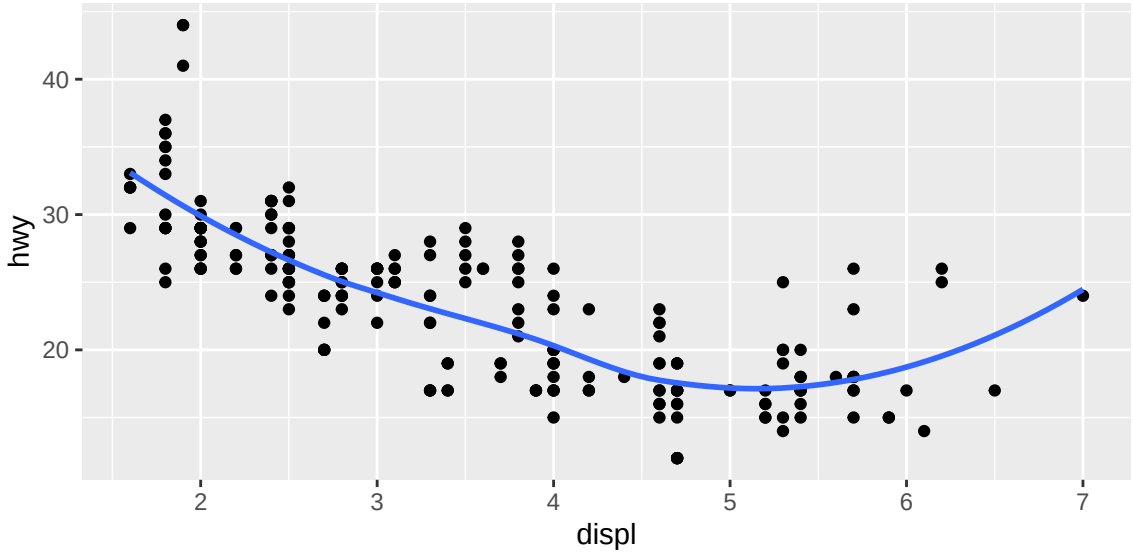
```
ggplot(data = mtcars, mapping = aes(x = hp, y = vs)) +
  geom_smooth(method = glm, method.args = list(family = "binomial"), se = FALSE)
```



Layering Geoms:

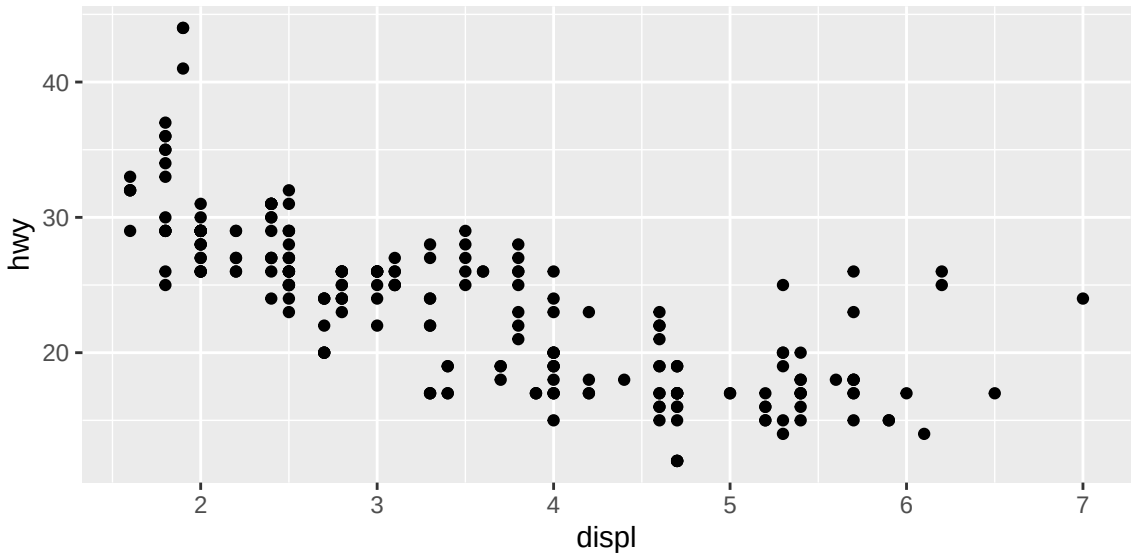
- You can layer multiple geoms that use the same data and aesthetic map as defined in the `ggplot()` call:

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



- You can equivalently define the aesthetics in the particular geom you are using. However, later geoms will not have access to these maps and you would have to redefine them.

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

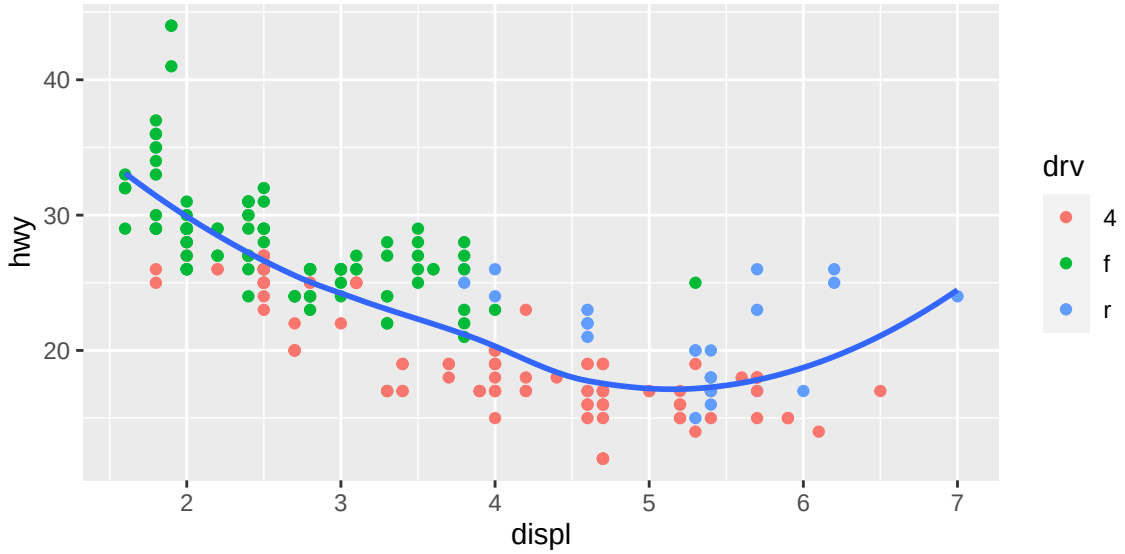



```
## Should produce an error
ggplot(data = mpg) +
  geom_point(mapping = aes(x = displ, y = hwy)) +
  geom_smooth()
```

- Place the aesthetic maps that are shared in the `ggplot()` call and place the geom-specific aesthetic maps within the `geom_` call

```
## Rug plot only on x-axis
ggplot(data = mpg, mapping = aes(x = displ, y = hwy)) +
  geom_point(mapping = aes(color = drv)) +
  geom_smooth(se = FALSE)
```

```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```

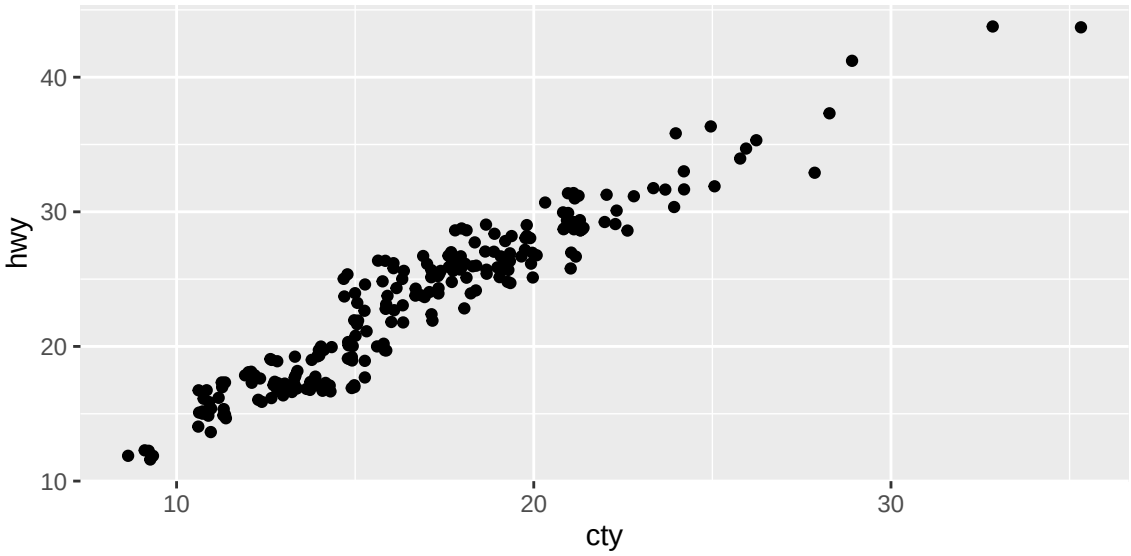


- **Exercise**: From the `penguins` dataset, try adding the OLS line to the scatterplot of bill length vs bill depth. Color code by species, but the OLS line should not depend on species.

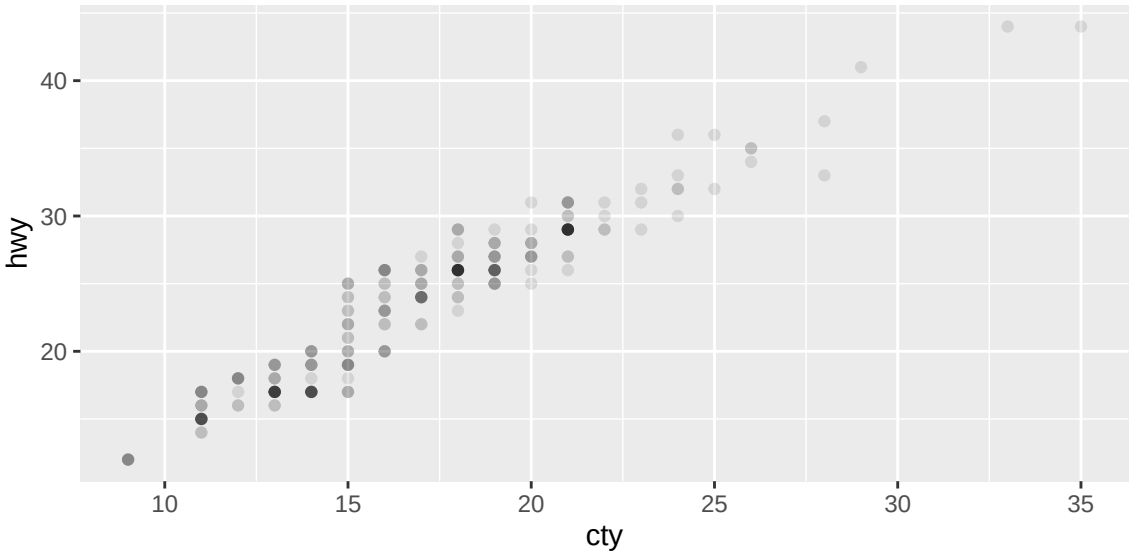
Overplotting

- The variables `cty` and `hwy` are rounded, so result in a lot of overlapping.
- Three ways to counteract this:
  1. If you have small data, use `geom_jitter()`
  2. If you have large data, set the transparency for all points to be low.
  3. If you have a really large dataset, randomly subsample the observational units (chapter 5).

```
ggplot(data = mpg, mapping = aes(x = cty, y = hwy)) +
  geom_jitter()
```




```
ggplot(data = mpg, mapping = aes(x = cty, y = hwy)) +
  geom_point(alpha = 0.1)
```



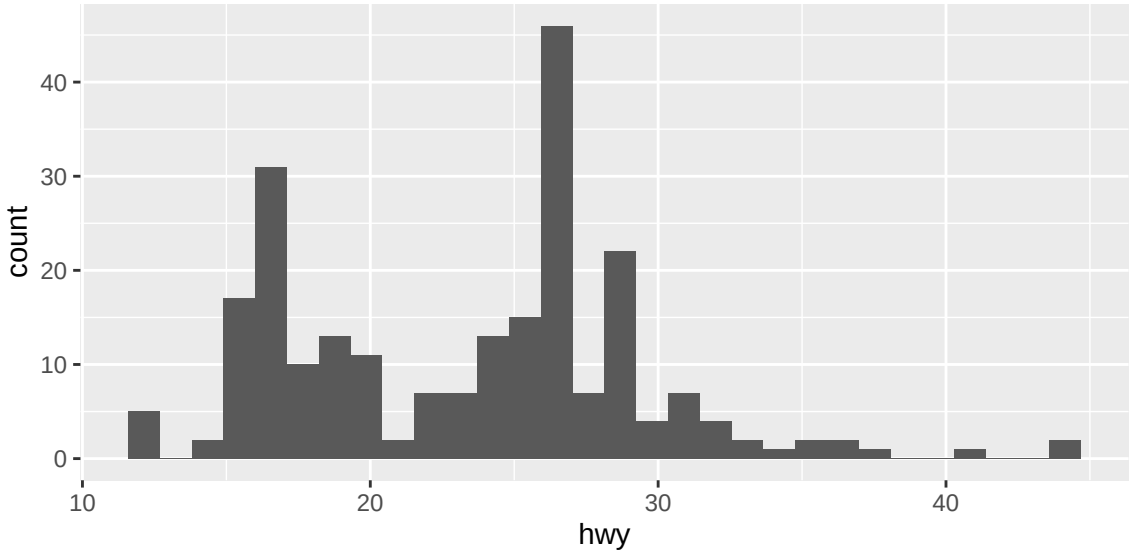
- **Exercise**: Consider the `diamonds` dataset from ggplot2. Load it and then plot `carat` vs `price`. Account for overplotting if possible.

One Quantitative Variable

- A histogram allows us to explore the distribution of a quantitative variable.
- "Distribution" = what values a variable takes and how often it takes those values.
- Bins observations into bins of equal *width*, plots counts on y-axis and variable values on x-axis.

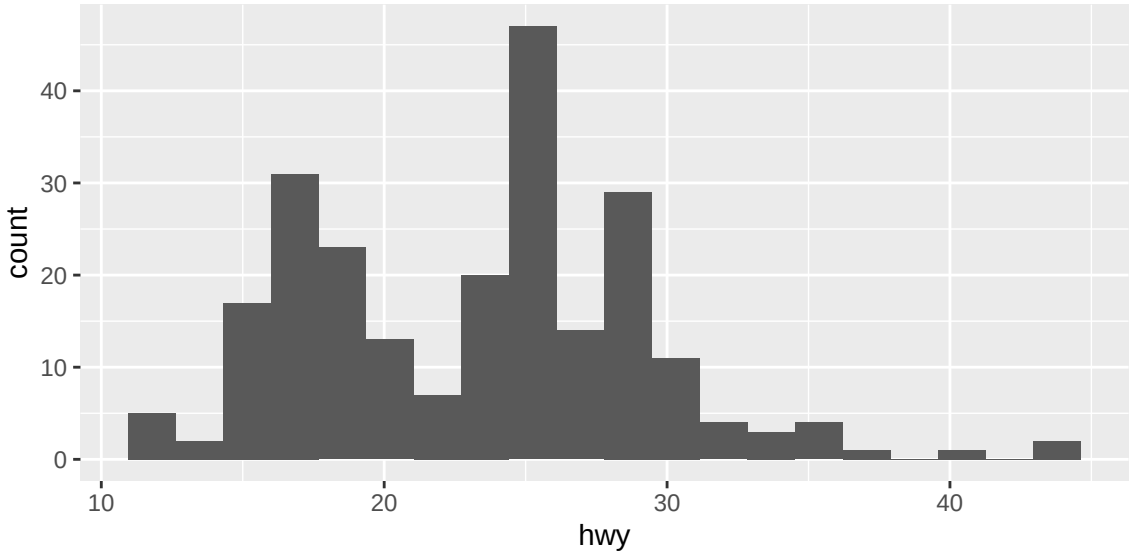
```
ggplot(data = mpg, mapping = aes(x = hwy)) +
  geom_histogram()
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



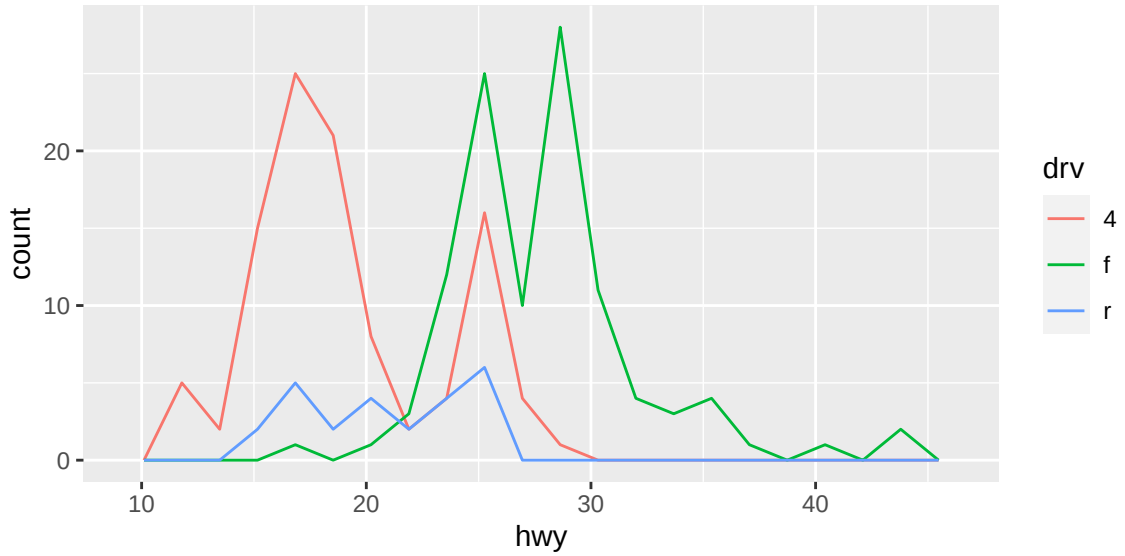
- Always play around with the `bins` argument until you get something nice-looking

```
ggplot(data = mpg, mapping = aes(x = hwy)) +
  geom_histogram(bins = 20)
```



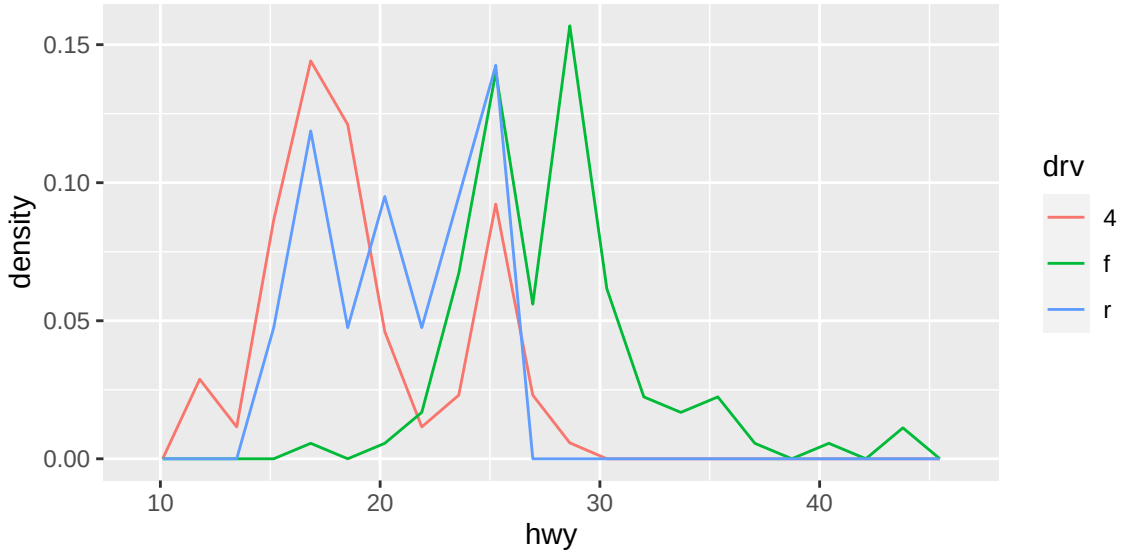
- Use `geom_freqpoly()` if you want to annotate by a categorical variable. This will create a "frequency polygon" which is just a histogram, except lines, rather than bars, represent the counts.

```
ggplot(data = mpg, mapping = aes(x = hwy, color = drv)) +
  geom_freqpoly(bins = 20)
```



- When you annotate by a categorical variable in a frequency polygon, it's best to plot the density rather than the counts. This puts all of the frequency polygons on the same y-scale.

```
ggplot(data = mpg, mapping = aes(x = hwy, y = ..density.., color = drv)) +
  geom_freqpoly(bins = 20)
```

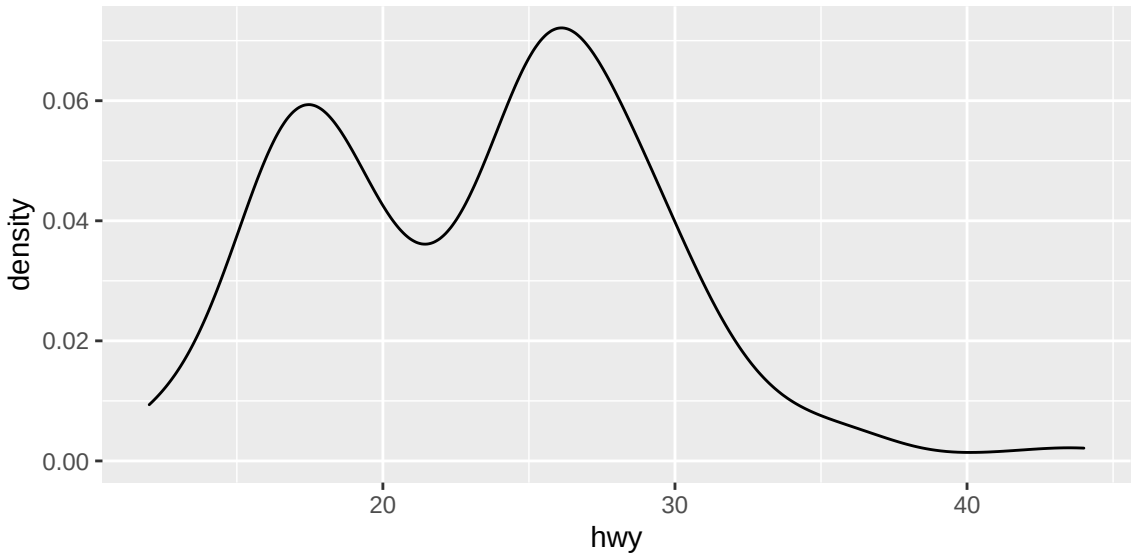


- **Exercise**: Create a frequency polygon of the price of a diamond, color coding by cut.

Density Plot

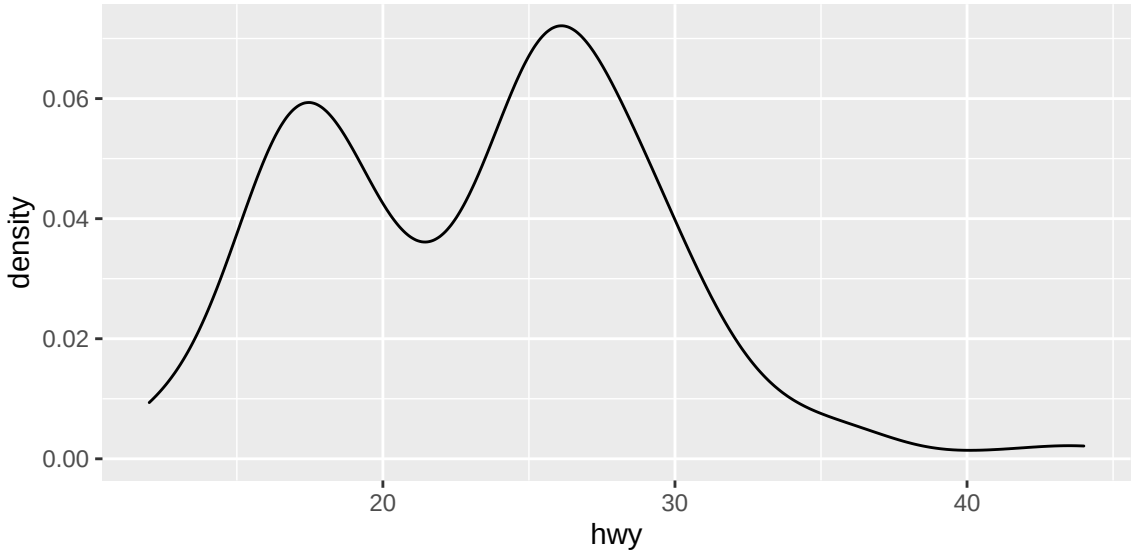
- A density is a "smoothed" histogram. It can be created with `geom_density()`
- Densities and histograms generally give us identical information.

```
ggplot(data = mpg, mapping = aes(x = hwy)) +
  geom_density()
```



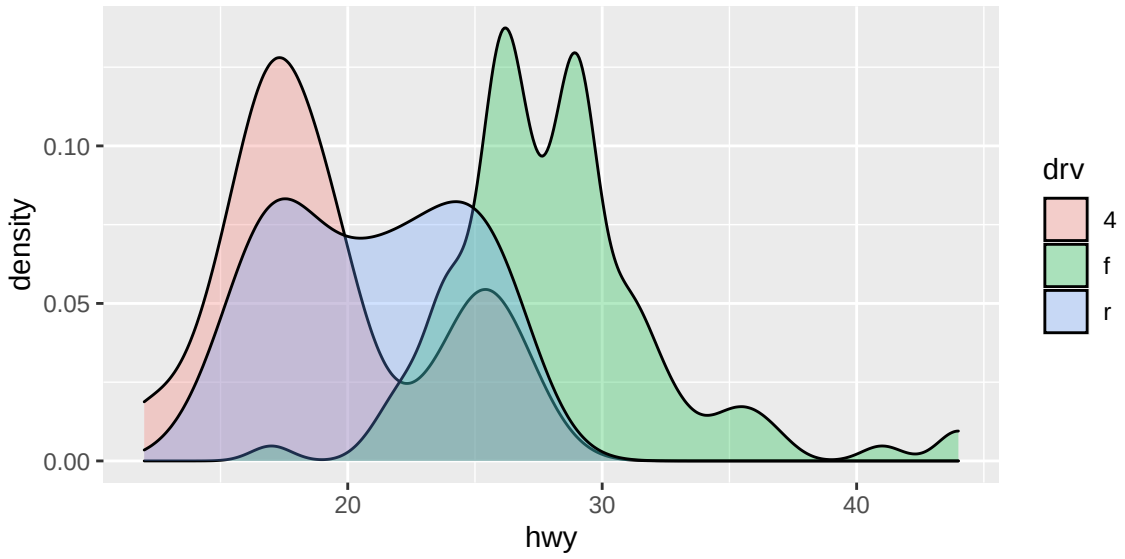
- You can plot the densities for the different levels of a categorical variable by mapping the `fill` aesthetic

```
ggplot(data = mpg, mapping = aes(x = hwy)) +
  geom_density(alpha = 0.3)
```



- You can change the transparency to see all densities.

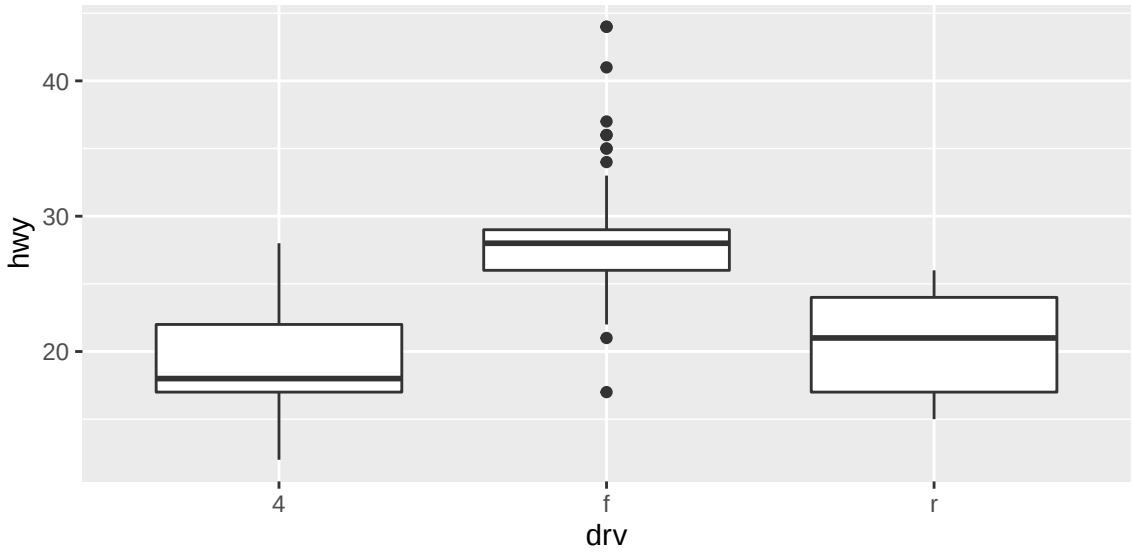
```
ggplot(data = mpg, mapping = aes(x = hwy, fill = drv)) +
  geom_density(alpha = 0.3)
```



One categorical variable, one quantitative variable.

- The best plot for visualizing the association between a categorical and a quantitative variable is the boxplot. Use `geom_boxplot()` for this.

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



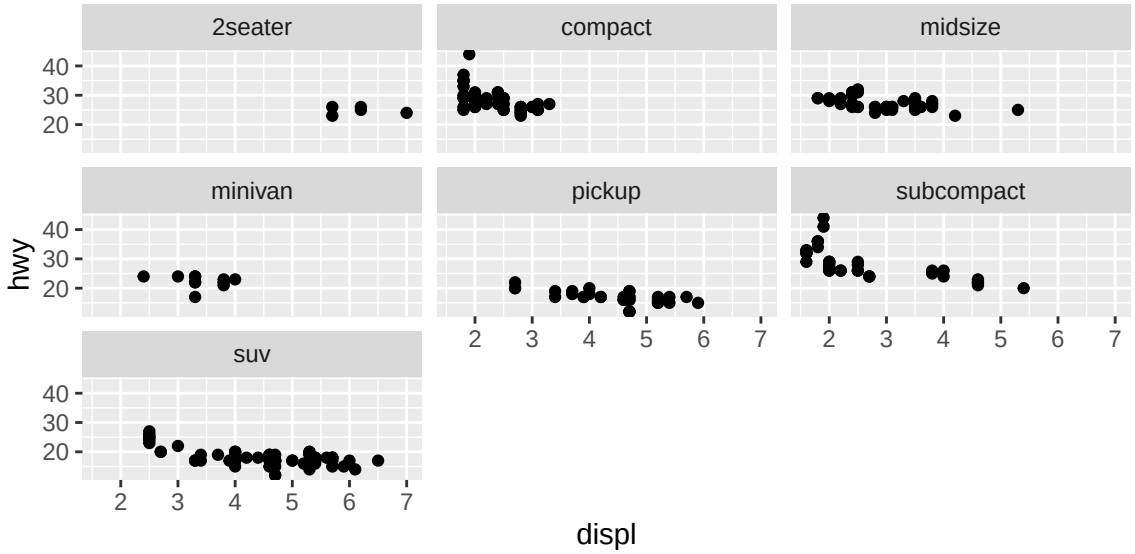
- **Exercise**: Create a boxplot for cut vs log-price. Does anything surprise you? Can you think of another plot that could explain your surprise?

Facets

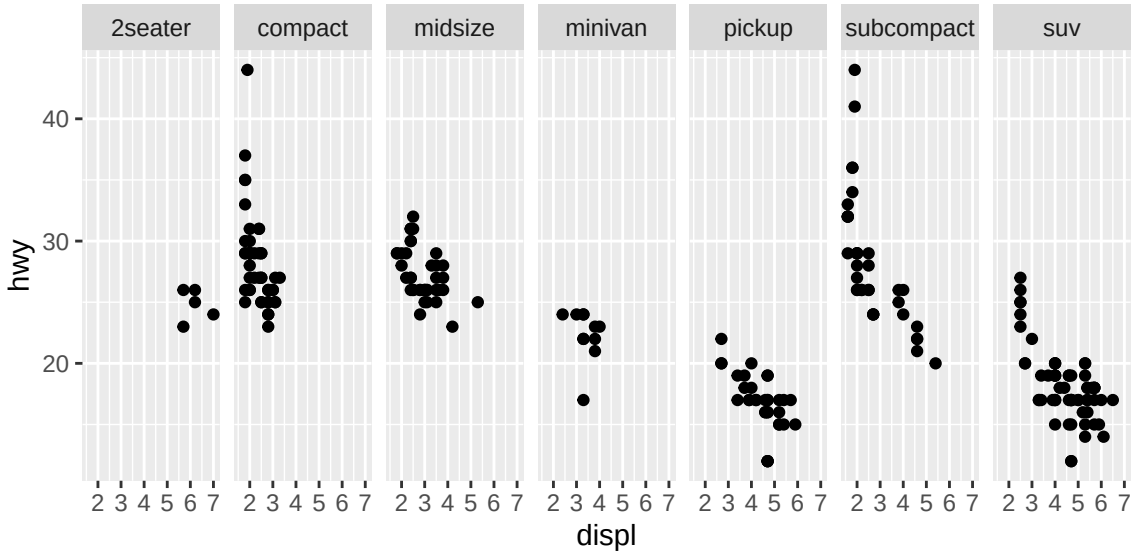
- If you want to create multiple plots from the same data, you can use the `facet_wrap()` or `facet_grid()` arguments.

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

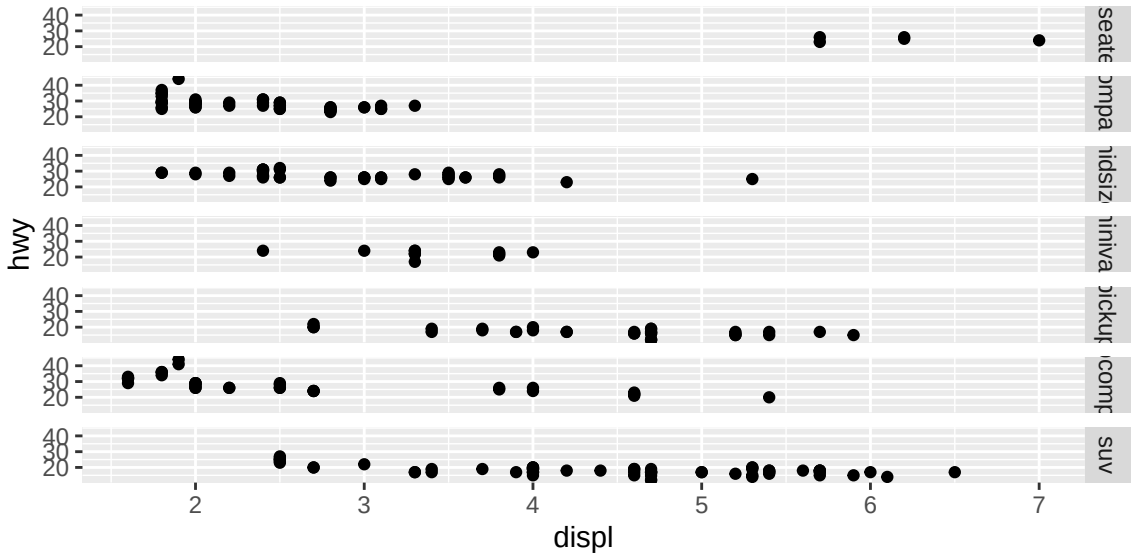
```
  facet_wrap( ~ class)
```



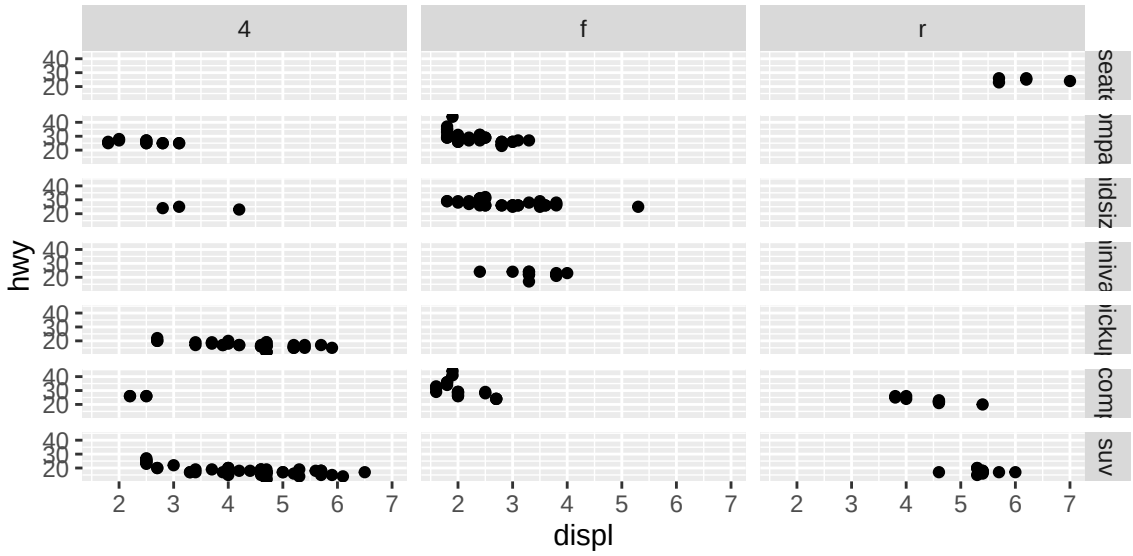

```
ggplot(data = mpg, mapping = aes(x = displ, y = hwy)) +
  geom_point() +
  facet_grid(. ~ class)
```




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




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

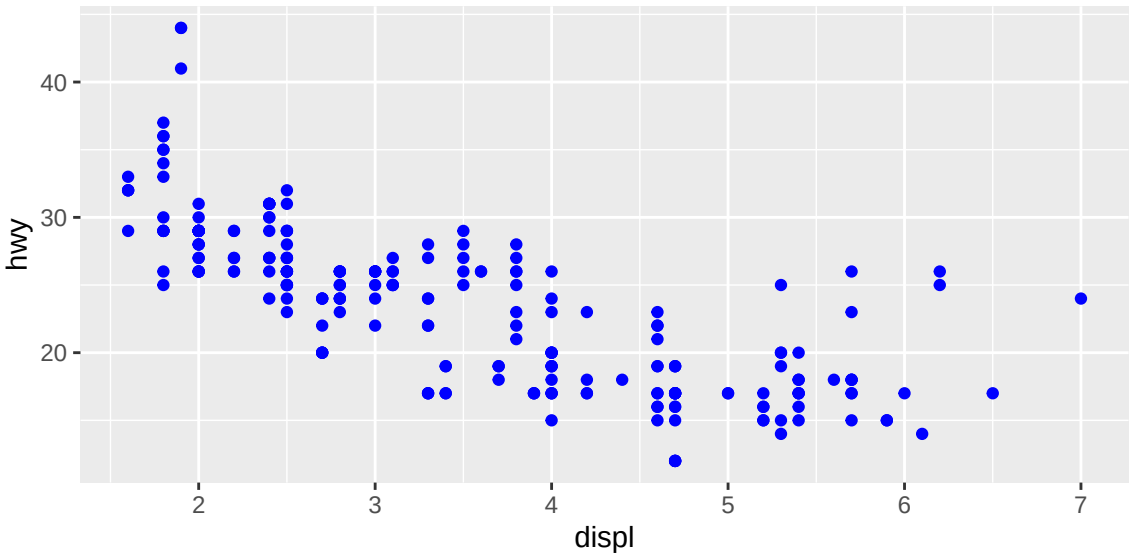


- **Exercise**: Create a boxplot of cut vs price for each level of color.

Aesthetics for all objects.

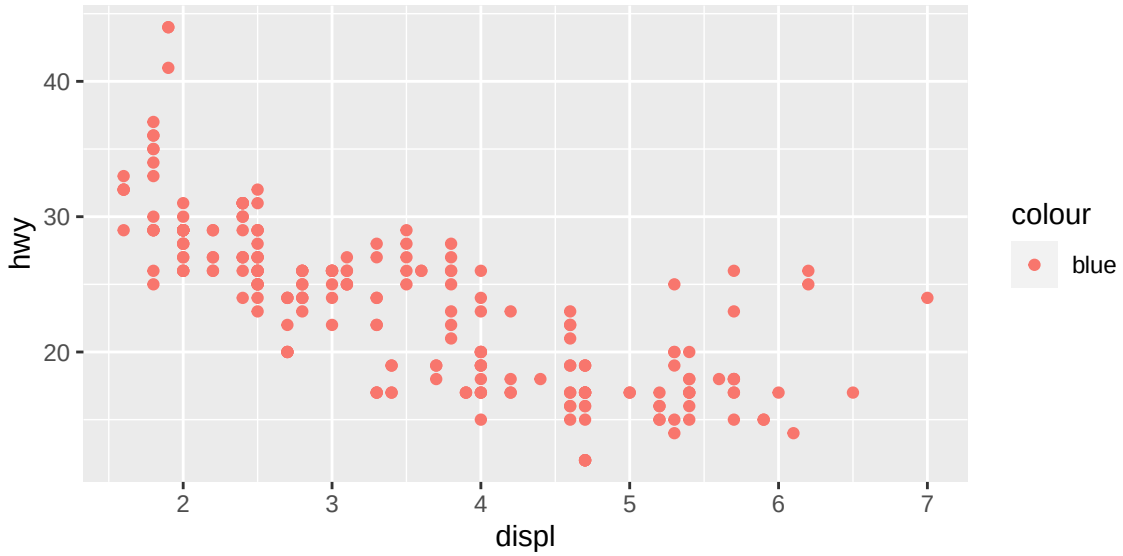
- Suppose we like scatterplots with blue points. The correct code to use is:

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



- The `"blue"` must be outside of the `aes()` call. This tells ggplot2 to set the aesthetic manually, instead of via a map from a variable.
- If you put `"blue"` inside of the `aes()` call, then ggplot2 will assume that you are creating a categorical variable where every observational unit has the value `"blue"`.

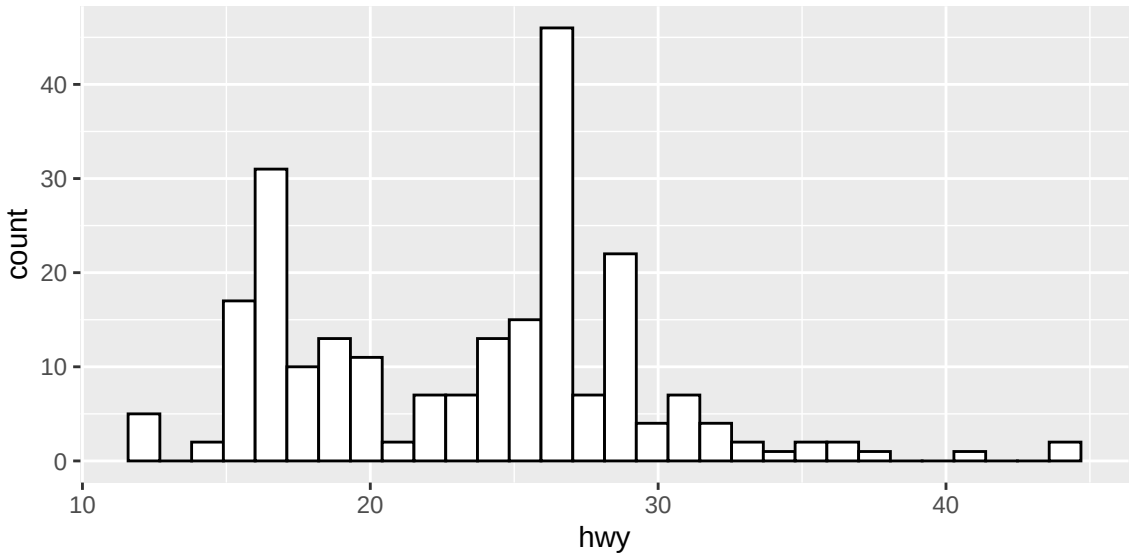
```
ggplot(data = mpg, mapping = aes(x = displ, y = hwy, color = "blue")) +
  geom_point()
```



- I often change the aesthetics for histograms.

```
ggplot(data = mpg, mapping = aes(x = hwy)) +
  geom_histogram(fill = "white", color = "black")
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

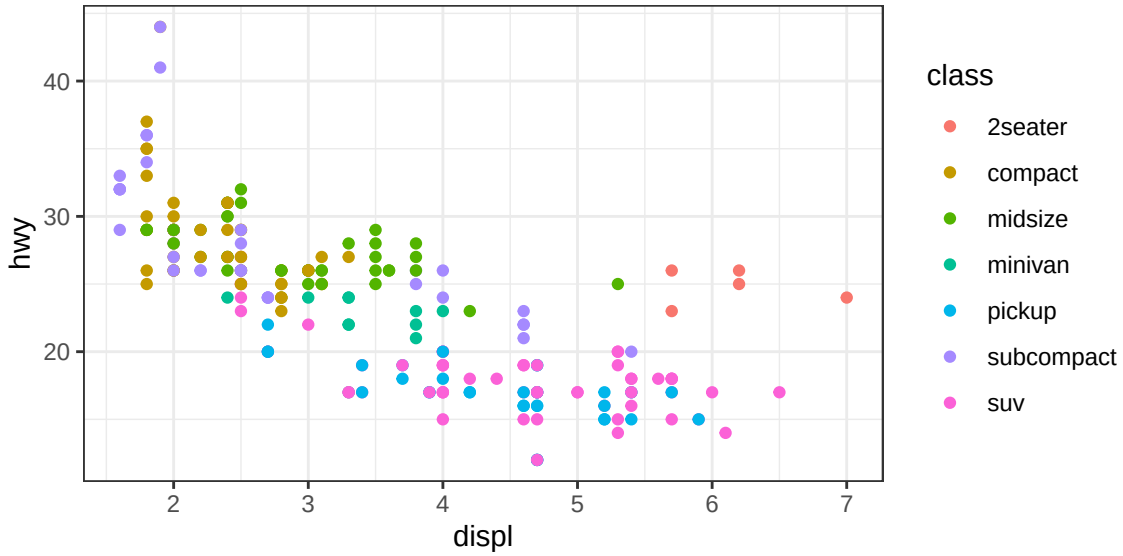


- I also often use the `alpha` argument frequently (see overplotting above).

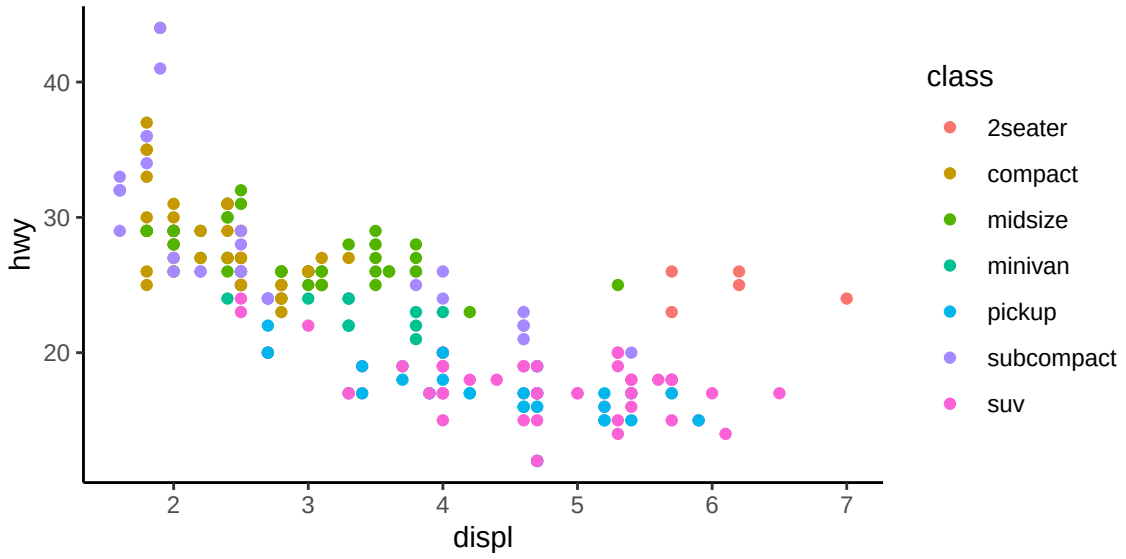
Themes

- The gray background is nice for presentations, but bad for publications. ggplot2 has a lot of themes that you can use. I think `theme_bw()` is the best.

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

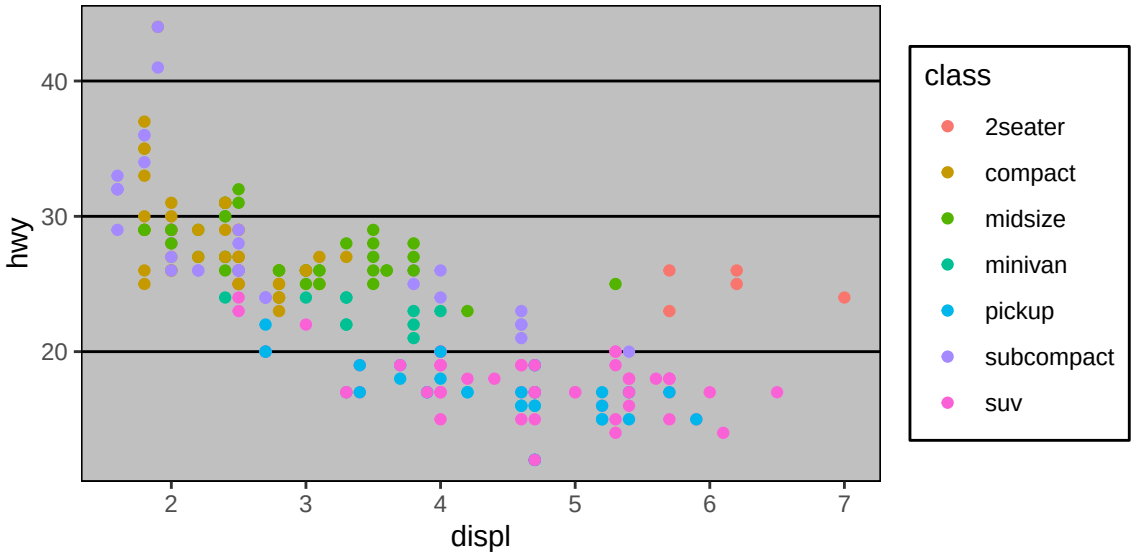



```
ggplot(data = mpg, mapping = aes(x = displ, y = hwy, color = class)) +
  geom_point() +
  theme_classic()
```

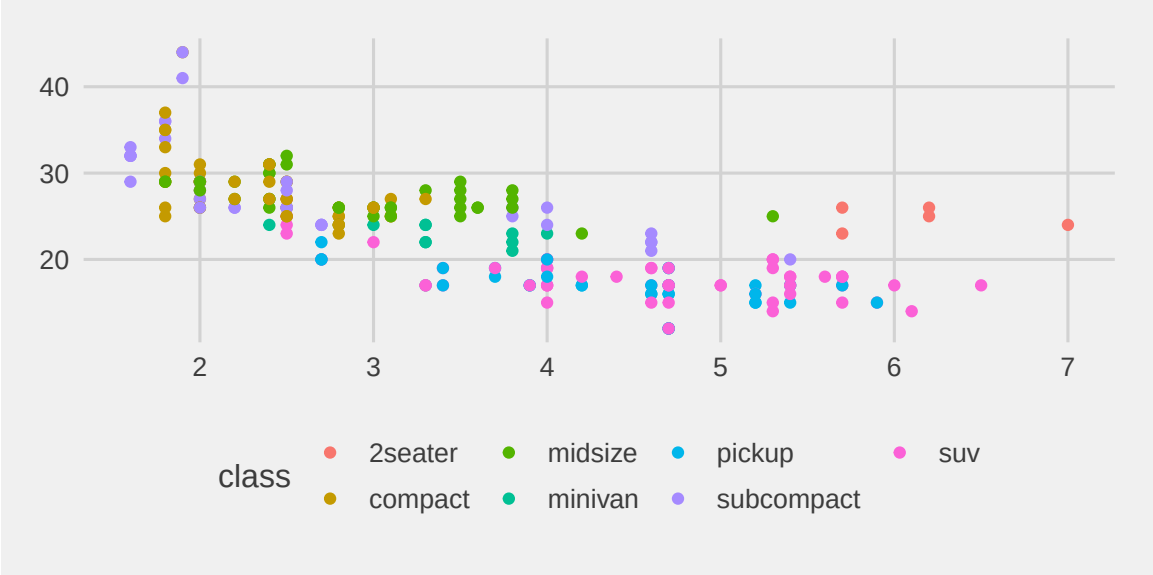


- Even more themes are available in the ggthemes package.

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




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

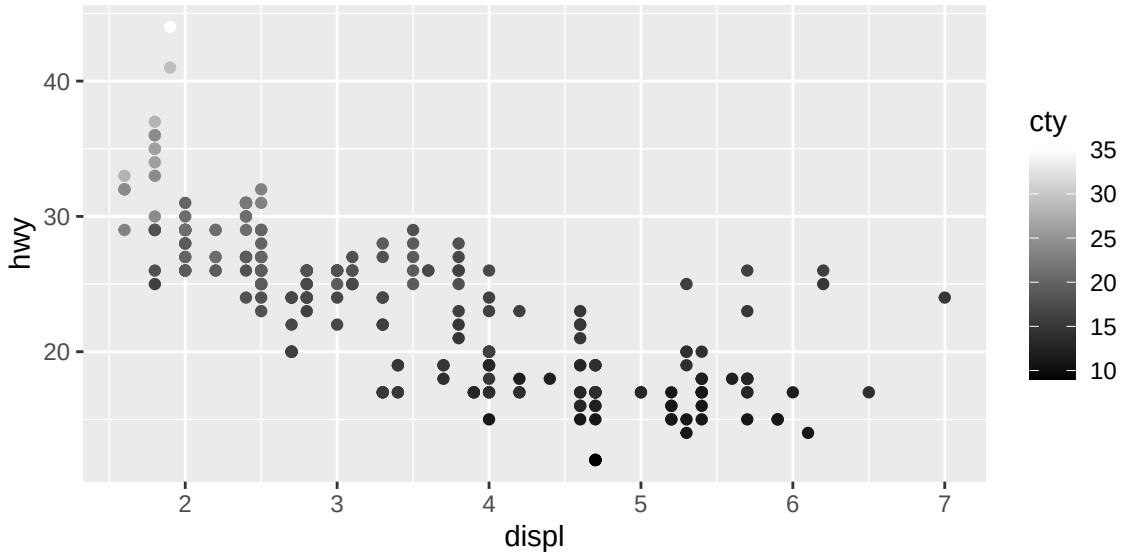


- **Exercise**: Create a boxplot of cut vs price for each level of color. Make the boxplots all orange and use the black and white theme.

Scales

- "Scales control how a plot maps data values to the visual values of an aesthetic. To change the mapping, add a custom scale."
- E.g.: The the scale of colors determines what color is the "largest" and which is the "smallest."

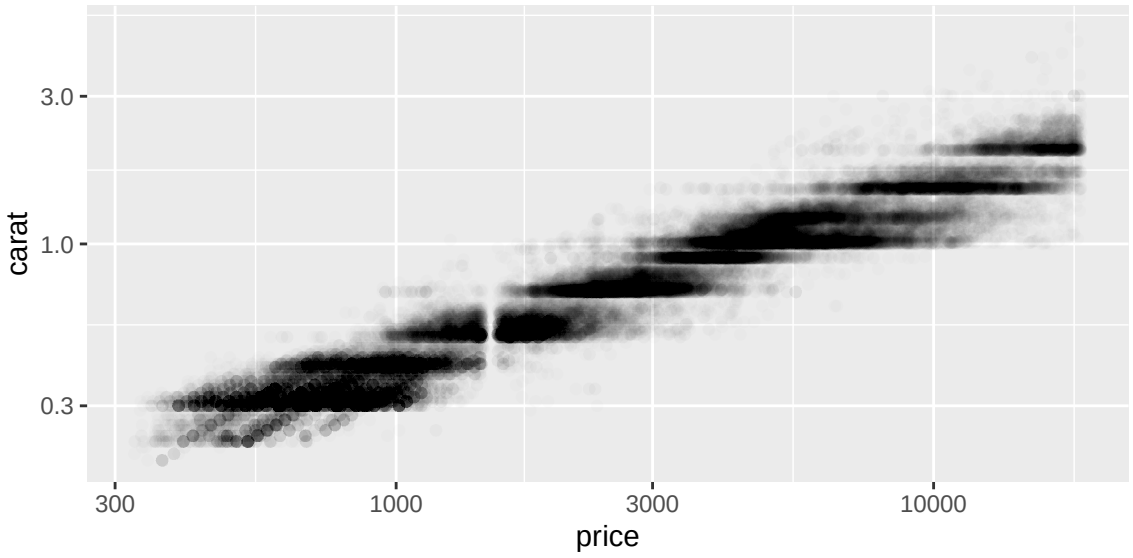
```
ggplot(data = mpg, mapping = aes(x = displ, y = hwy, color = cty)) +
  geom_point() +
  scale_color_continuous(low = "black", high = "white")
```



- E.g. The scale of an axis (log vs normal)

```
ggplot(data = diamonds, mapping = aes(x = price, y = carat)) +
  scale_y_log10() +
```

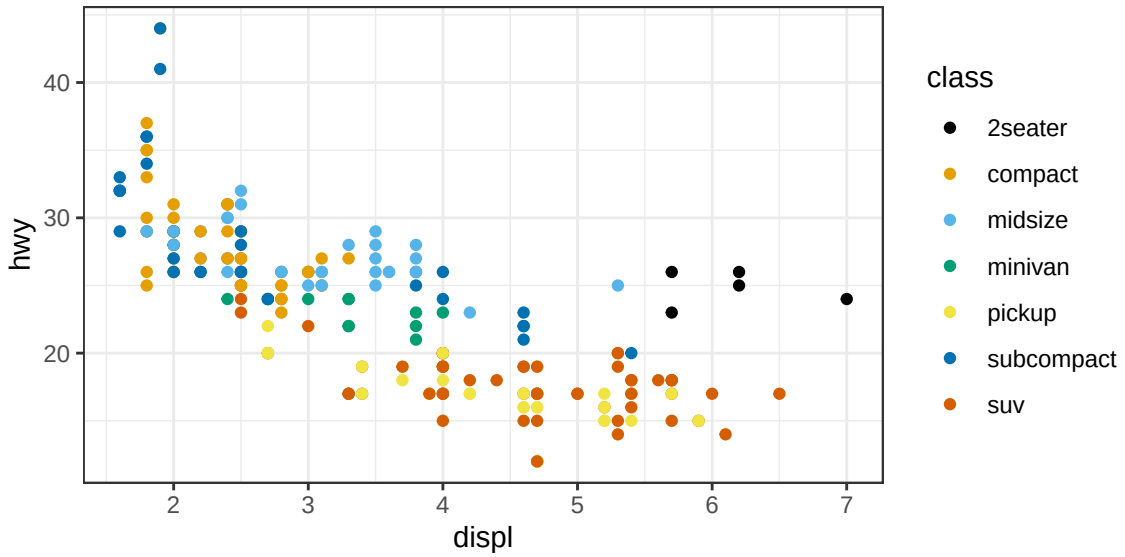
```
  scale_x_log10() +
  geom_point(alpha = 0.01)
```



Colorblind safe palettes

- When you publish, it's polite to use colorblind safe palettes. You can do this easily with the ggthemes package.

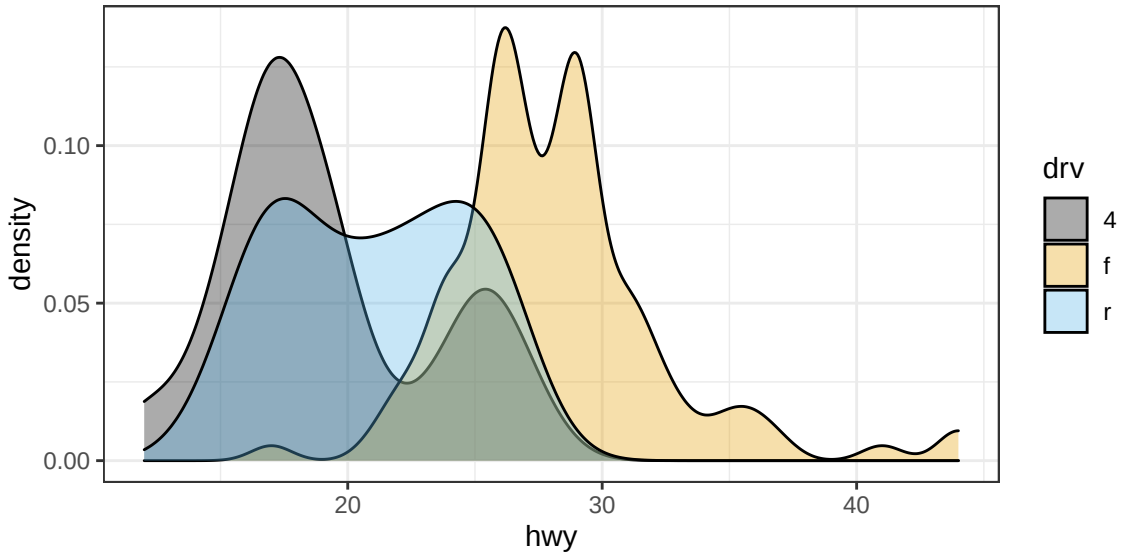
```
ggplot(data = mpg, mapping = aes(x = displ, y = hwy, color = class)) +
  geom_point() +
  theme_bw() +
  scale_color_colorblind()
```



- Use `scale_color_colorblind()` to change the scale of the `color` aesthetic. Use `scale_fill_colorblind()` to change the scale of the `fill` aesthetic, etc...

```
ggplot(data = mpg, mapping = aes(x = hwy, fill = drv)) +
  geom_density(alpha = 1/3) +
```

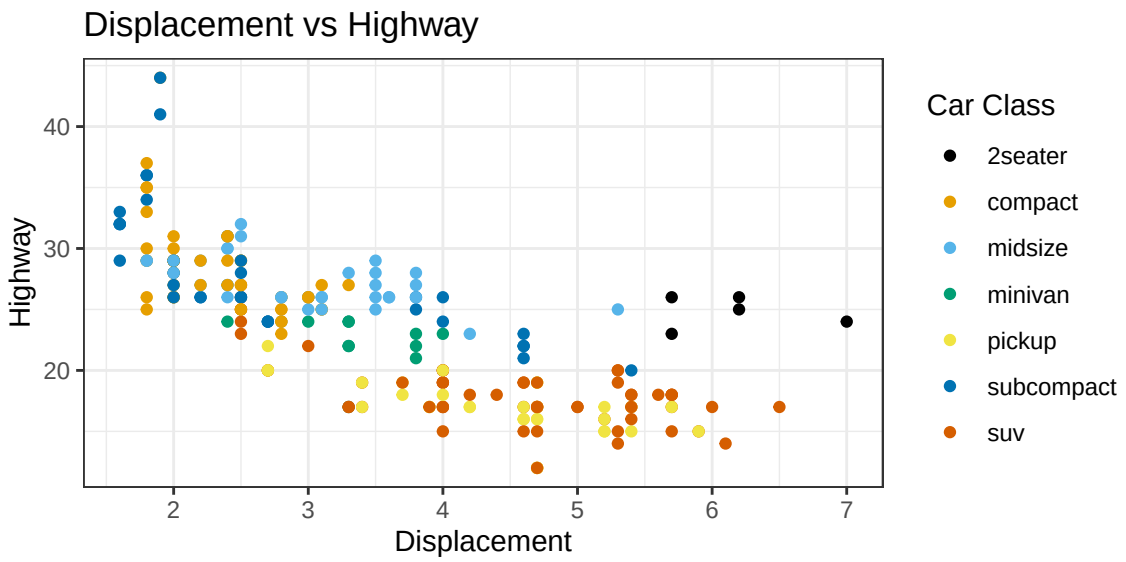
```
  theme_bw() +
  scale_fill_colorblind()
```



Labels

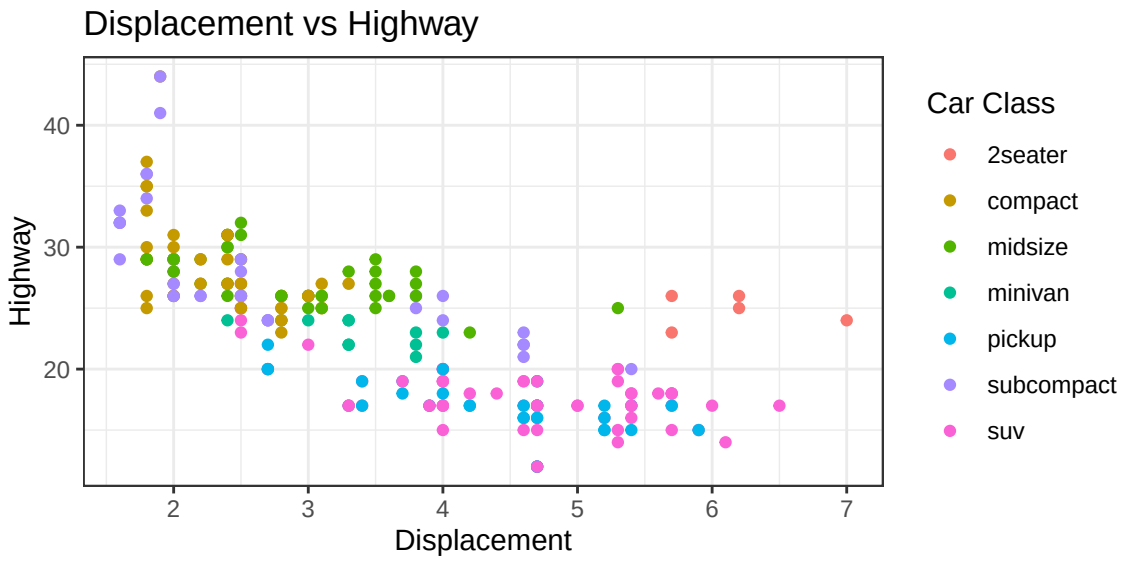
- You can set the axis labels manually with `xlab()` and `ylab()`.
- Titles may be added with `ggtitle()`.
- Legend titles need to be modified with a scale argument.

```
ggplot(data = mpg, mapping = aes(x = displ, y = hwy, color = class)) +
  geom_point() +
  theme_bw() +
  scale_color_colorblind(name = "Car Class") +
  xlab("Displacement") +
  ylab("Highway") +
  ggtitle("Displacement vs Highway")
```



- If you don't want colorblind scale, you can use `scale_color_discrete()` or `scale_fill_discrete()` or `scale_color_continuous()` or `scale_fill_continuous()` etc...

```
ggplot(data = mpg, mapping = aes(x = displ, y = hwy, color = class)) +
  geom_point() +
  theme_bw() +
  scale_color_discrete(name = "Car Class") +
  xlab("Displacement") +
  ylab("Highway") +
  ggtitle("Displacement vs Highway")
```



Saving Plots

- To save a plot, you can either print it and then use the R Studio functionality.

```
ggplot(data = mpg, mapping = aes(x = displ, y = hwy, color = class)) +
  geom_point() +
  theme_bw() +
  scale_color_colorblind() ->
  pl
```

- It's better to automate this process using the `ggsave()` function

```
ggsave(filename = "my_first_plot.pdf",
       plot = pl,
       width = 6,
       height = 4,
       family = "Times")
```

- You can also do this manually using `pdf()`:

```
pdf(file = "my_second_plot.pdf", width = 6, height = 4, family = "Times")
print(pl)
dev.off()
```

```
## pdf
##   2
```

- `pdf()` opens up a connection to a new pdf file. `print(pl)` prints our saved `gg` object to that pdf file (instead of the R graphics device). `dev.off()` closes the connection.
- **Exercise**: Create a boxplot of cut vs price for each level of color. Make the boxplots all orange and use the black and white theme. Save this plot to the output folder using `ggsave()`.