Data visualization

Harlley Lima & Manel Slokom

November 28, 2018

Overview

1. ggplot2
   - Aesthetic mappings
   - Facets
   - Geometric objects
   - Position adjustments

2. Exploratory data analysis
   - Variation
   - Covariation

R base graphics

- Data and analysis results are best communicated through *visualization*
- R standard graphics includes several functions that provide standard statistical plots, like:
  - `plot()`: generic x-y plotting
  - `barplot()`: bar plots
  - `boxplot()`: boxplot
  - `hist()`: hitograms
- This tutorial will focus on `ggplot2`, one of the most elegant and most versatile for making graphs

Data visualization

`ggplot2`:

- offers a powerful graphics language for creating elegant and complex plots
- implements the grammar of graphics: a coherent system for describing and building graphs

If you'd like to learn more about the theoretical underpinnings of `ggplot2`, I'd recommend reading The Layered Grammar of Graphics, `http://vita.had.co.nz/papers/layered-grammar.pdf`

ggplot2

Prerequisites:

- Load the `tidyverse` by running this code:

```
library(tidyverse)
```

- If you run this code and get the error message "there is no package called tidyverse", you'll need to first install it, then run `library()` once again:

```
install.packages(tidyverse)
library(tidyverse)
```

First steps

The `mpg` data frame

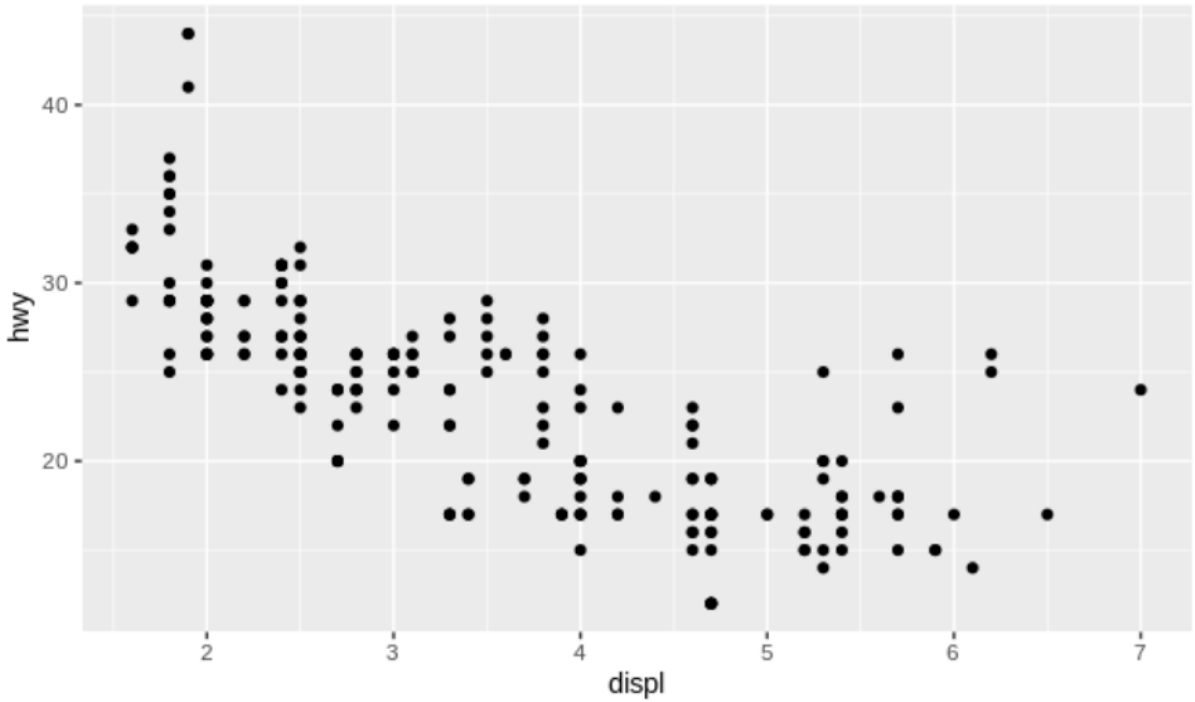
- `mpg` contains observations collected on 38 models of car
- Among the variables in `mpg` are:
  - `displ`: a car's engine size, in litres
  - `hwy`: a car's fuel efficiency on the highway, in miles per gallon (mpg)

Creating a ggplot

Do cars with big engines use more fuel than cars with small engines?

- To plot `mpg`, run this code to put displ on the x-axis and hwy on the y-axis:

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



Creating a ggplot

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

- `ggplot()` creates a coordinate system that you can add layers to
- The first argument of `ggplot()` is the dataset to use in the graph
- `ggplot(data = mpg)` creates an empty graph

Creating a ggplot

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

- The function `geom_point()` adds a layer of points to your plot
- `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

A graphing template

```
ggplot(data = <DATA>) +
     <GEOM_FUNCTION>(mapping = aes(<MAPPINGS>))
```

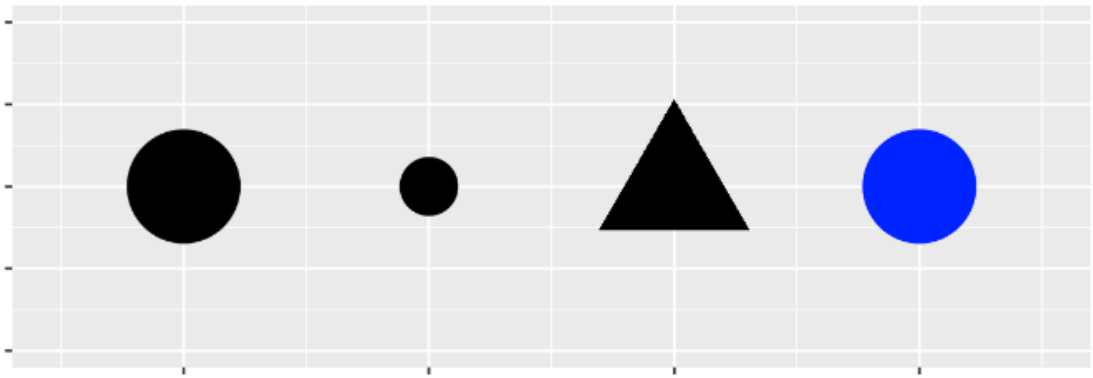
```
g <- ggplot(data = <DATA>)
g + <GEOM_FUNCTION>(mapping = aes(<MAPPINGS>))
```

The rest of this tutorial will show you how to complete and extend this template to make different types of graphs

Aesthetic mappings

Aesthetic mappings

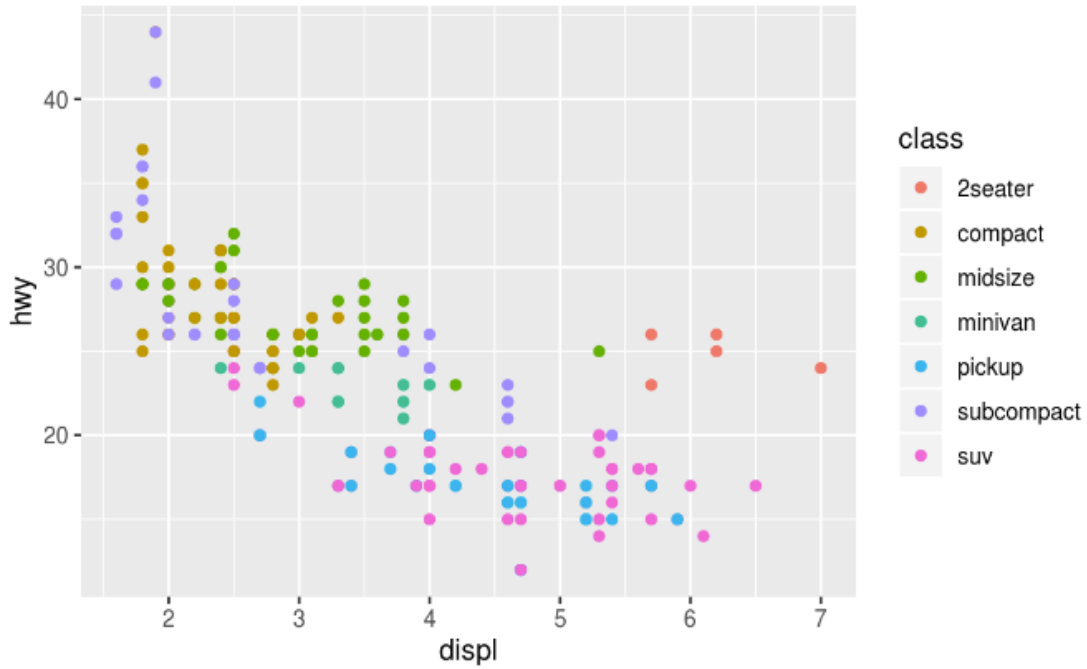
- An aesthetic is a visual property of the objects in your plot
- Aesthetics include things:
  - size
  - shape
  - color of the points
- Here we change the levels of a point's size, shape, and color to make the point small, triangular, or blue:



Aesthetic mappings

We can add a third variable, like `class`, to a two dimensional scatterplot by mapping it to an aesthetic:

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



Aesthetic mappings

We can add a third variable, like `class`, to a two dimensional scatterplot by mapping it to an aesthetic:

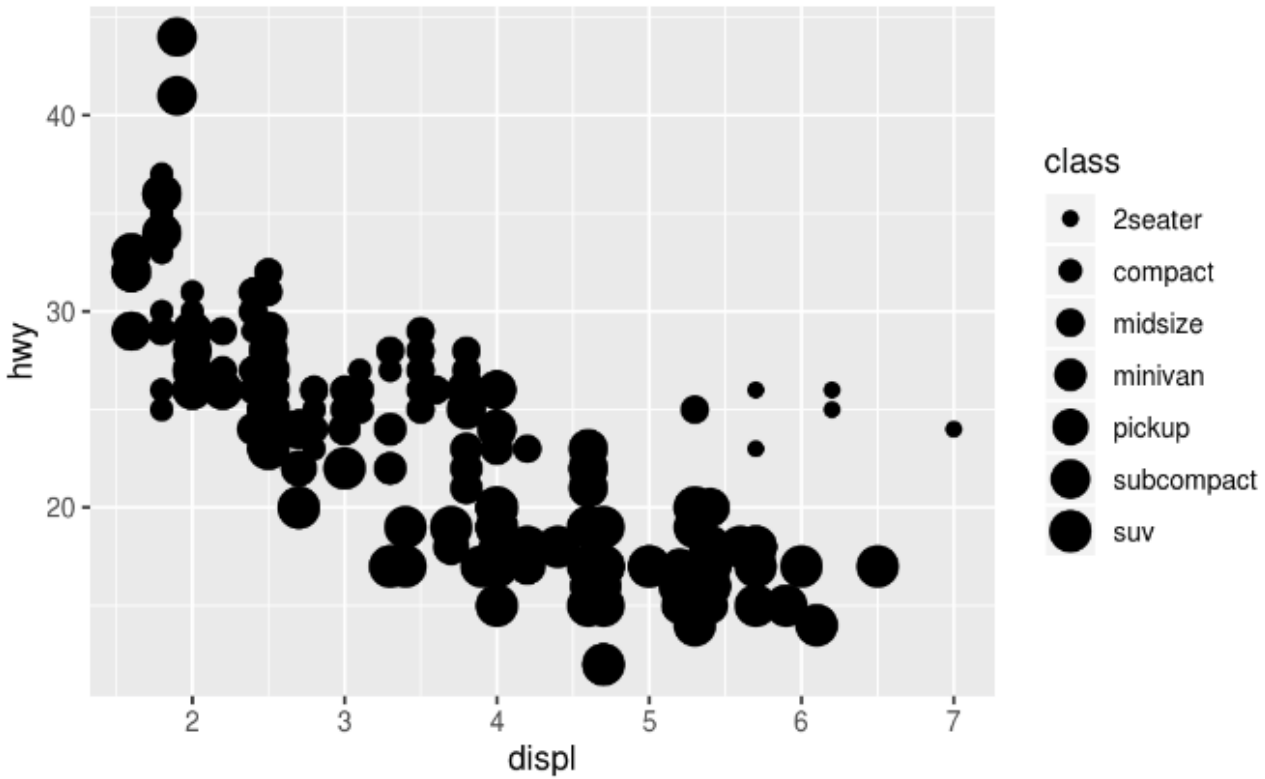
```
ggplot(data = mpg) +
     geom_point(mapping = aes(x = displ, y = hwy, color = class))
```

- The `class` variable of the `mpg` dataset classifies cars into groups such as compact, midsize, and pickup
- `ggplot2` will assign a unique level of the aesthetic to each unique value of the variable
- `ggplot2` will also add a legend

Aesthetic mappings

We can also map `class` to the size aesthetic in the same way:

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



Aesthetic mappings

We can also map `class` to the size aesthetic in the same way:

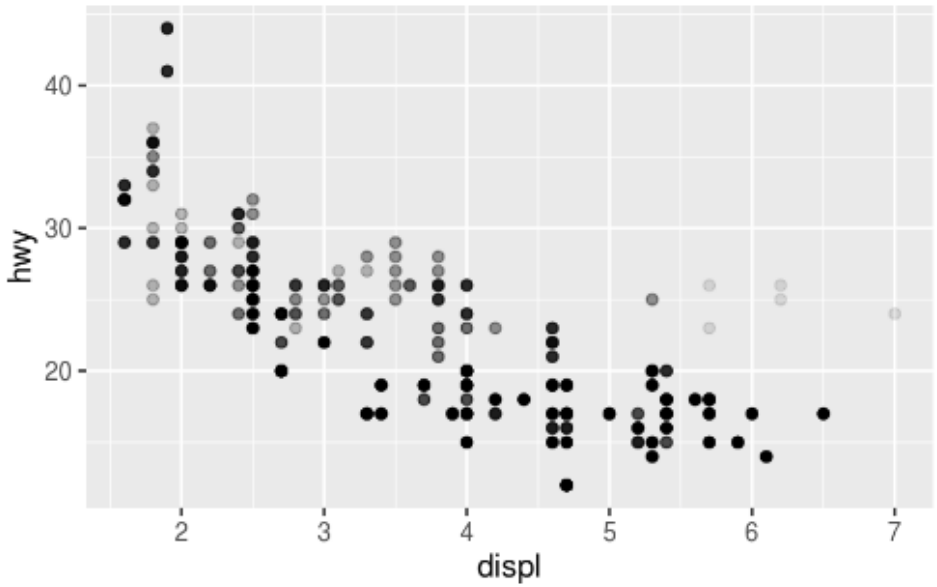
```
ggplot(data = mpg) +
     geom_point(mapping = aes(x = displ, y = hwy, size = class))
```

- We got a warning because mapping an unordered variable (`class`) to an ordered aesthetic (`size`) is not a good idea.

Aesthetic mappings

We can also map `class` to the alpha aesthetic (transparency) in the same way:

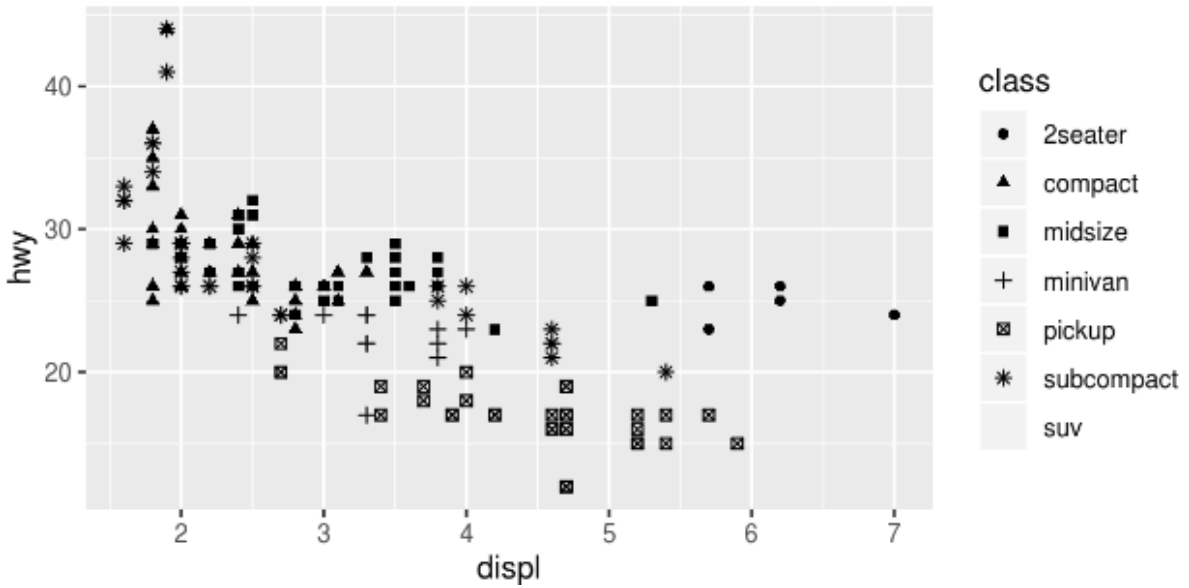
```
ggplot(data = mpg) +
     geom_point(mapping = aes(x = displ, y = hwy, alpha = class))
```



Aesthetic mappings

We can also map `class` to the shape aesthetic in the same way:

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



Facets

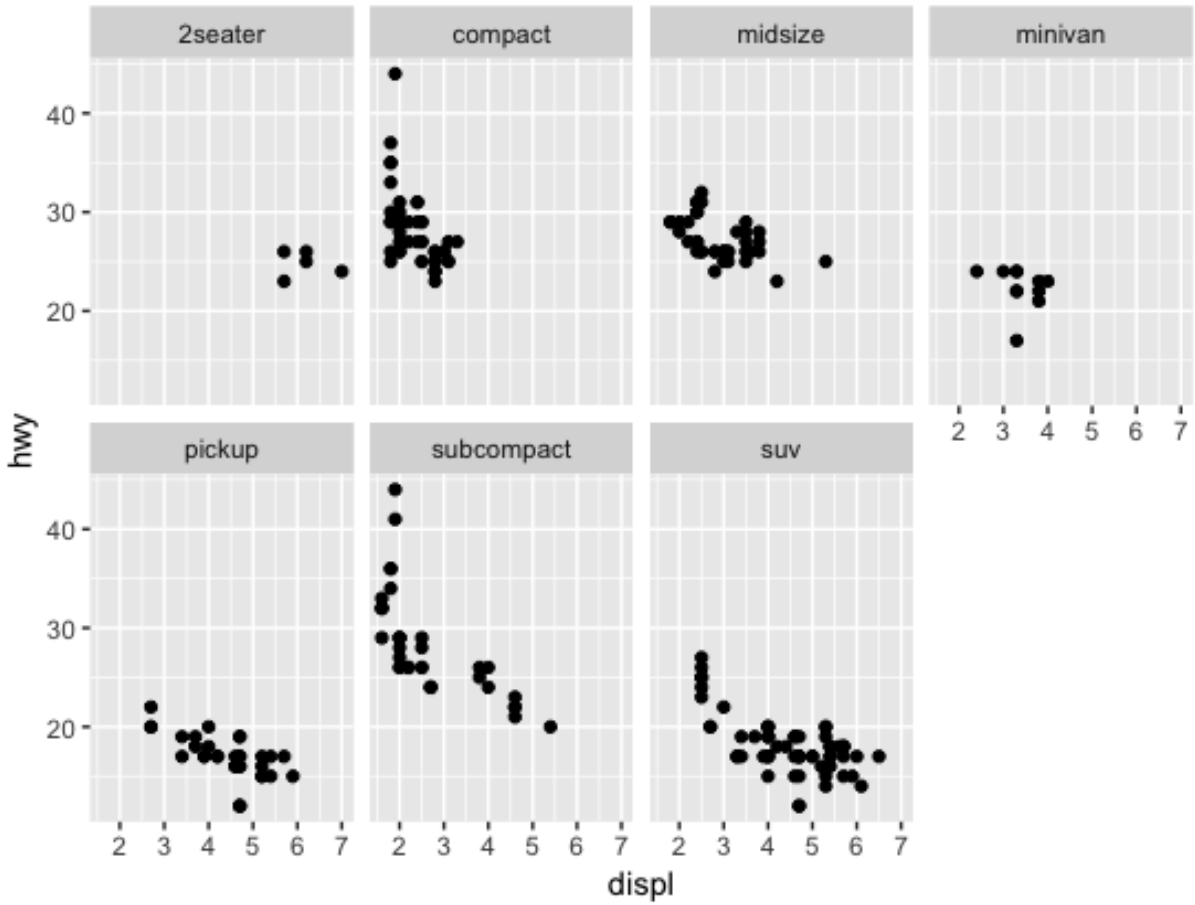
Facets

- One way to add additional variables is with aesthetics.
- Another way (for categorical variables) is to split your plot into facets, subplots that each display one subset of the data.

To facet your plot by a single variable, use `facet_wrap()`

Facets

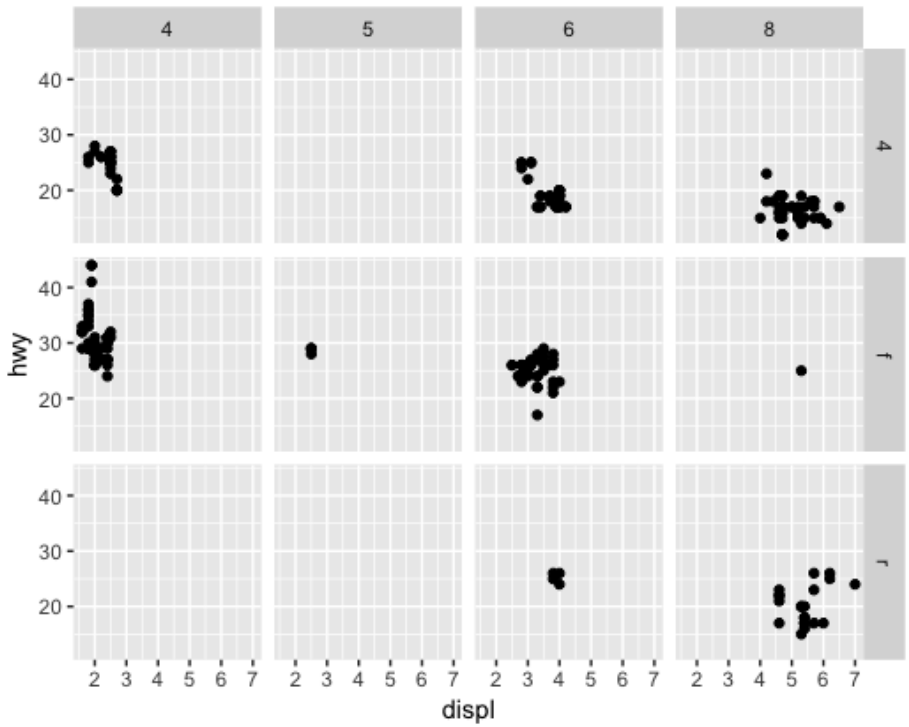
```
> ggplot(data = mpg) +
   geom_point(mapping = aes(x = displ, y = hwy)) +
   facet_wrap(~ class, nrow = 2)
```



Facets

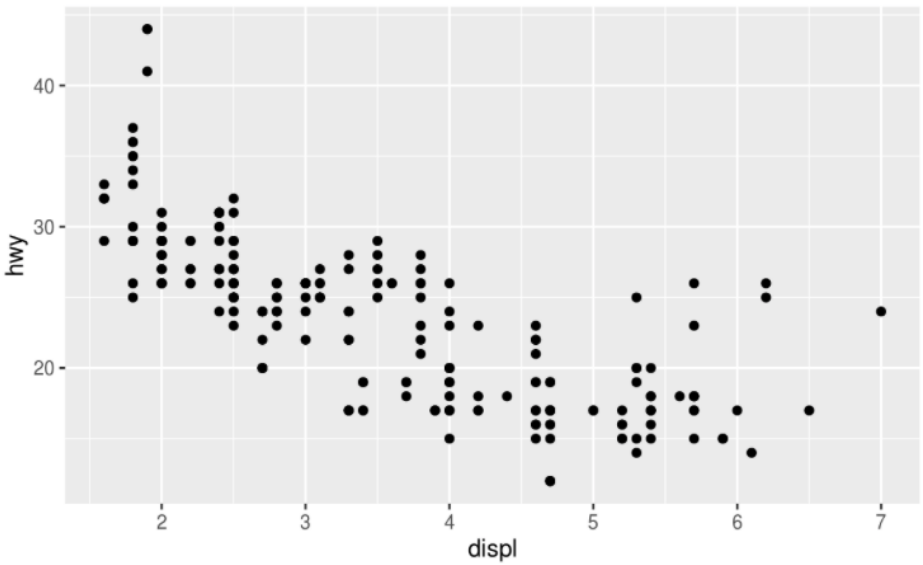
To facet your plot on the combination of two variables: `facet_grid()`
`facet_grid()` should contain two variable names separated by a $\sim$

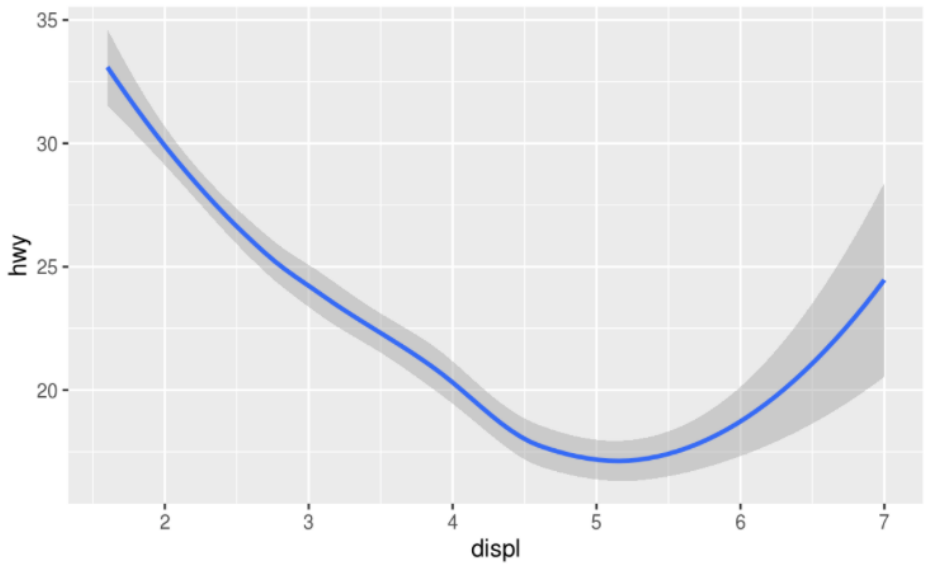
```
> ggplot(data = mpg) +
   geom_point(mapping = aes(x = displ, y = hwy)) +
   facet_grid(drv ~ cyl)
```



Geometric objects

Geometric objects





- both plots contain the same `x` variable
- the same `y` variable
- both describe the same data

Geometric objects

- The plots are not identical
- Each plot uses a **different visual object** to represent the data
- In ggplot2 syntax, it is called "geoms"

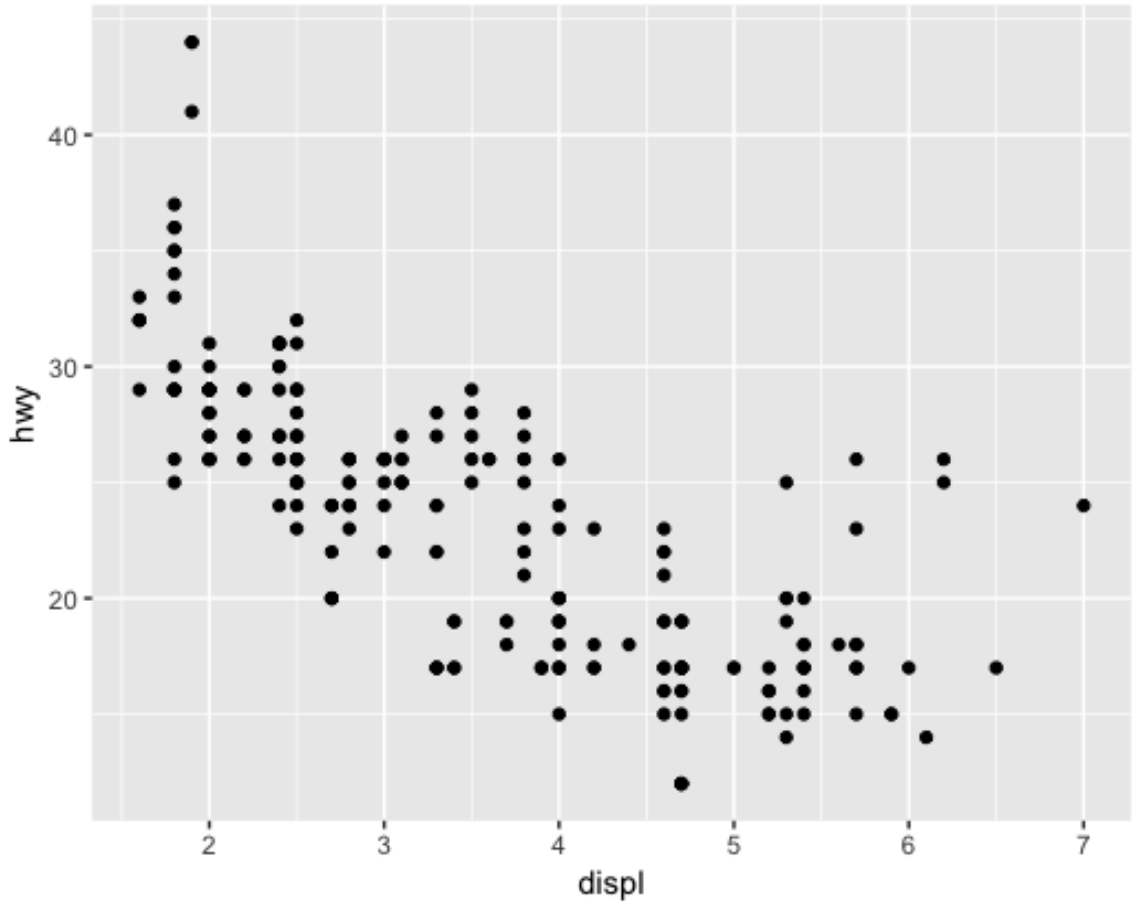
Geom

A geom is the geometrical object that a plot uses to represent data

- Bar charts use bar geoms
- Line charts use line geoms
- Boxplots use boxplot geoms
- Scatterplots break the trend by using the point geom

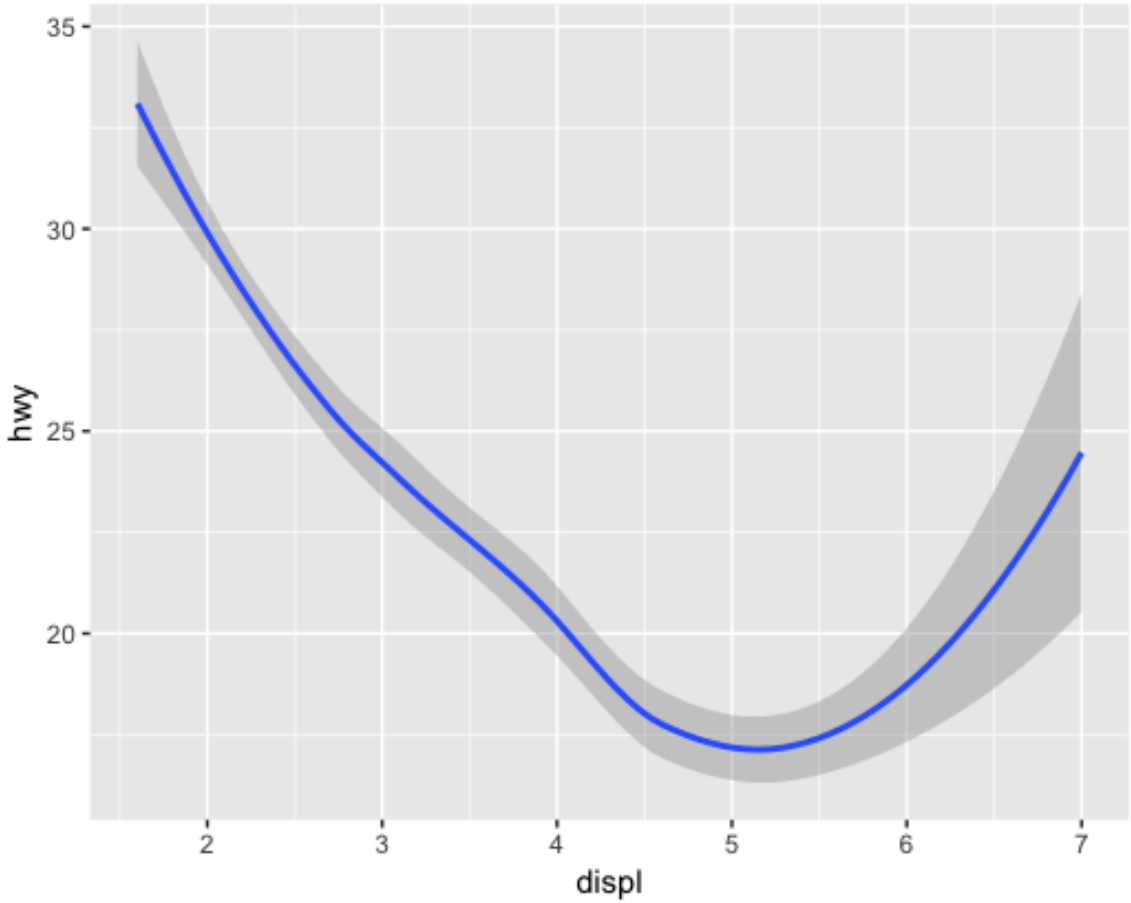
Example

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



Example

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



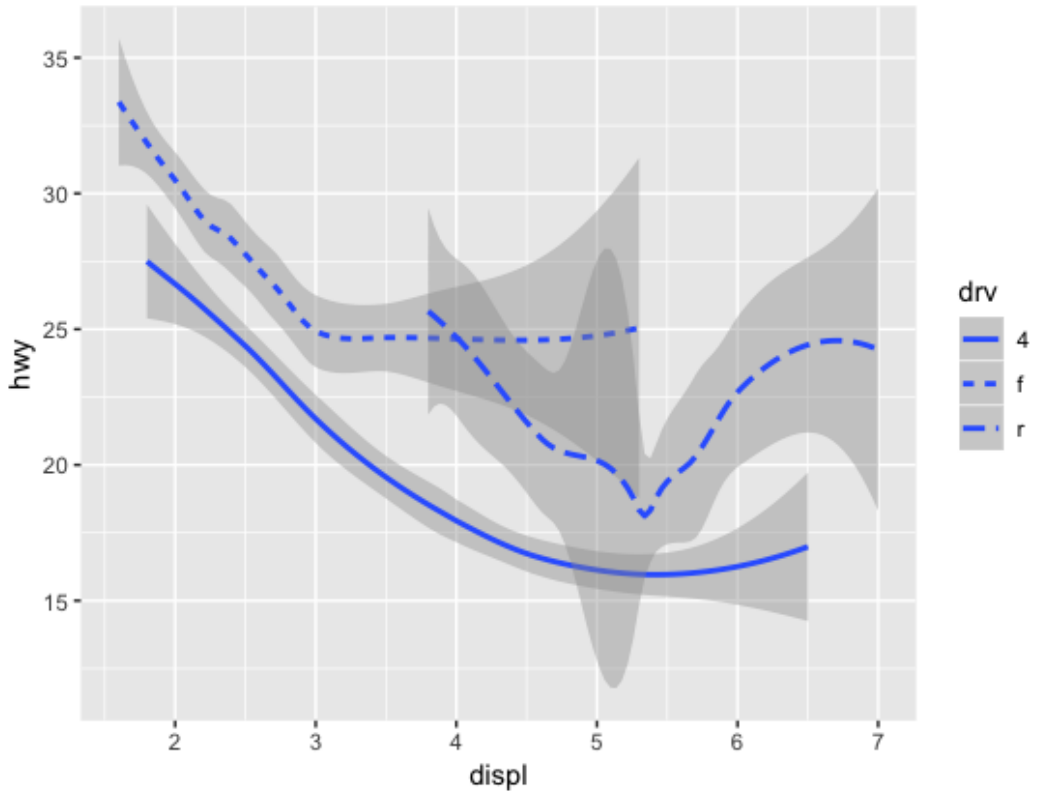
Geometric objects

- Every geom function in `ggplot2` takes a mapping argument
- Not every aesthetic works with every geom:
  - We can set the shape of a point
  - We can't set the "shape" of a line
  - you could set the linetype of a line

Geometric objects

`geom_smooth()` will draw a different line, with a different linetype, for each unique value of the variable:

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



Exercise

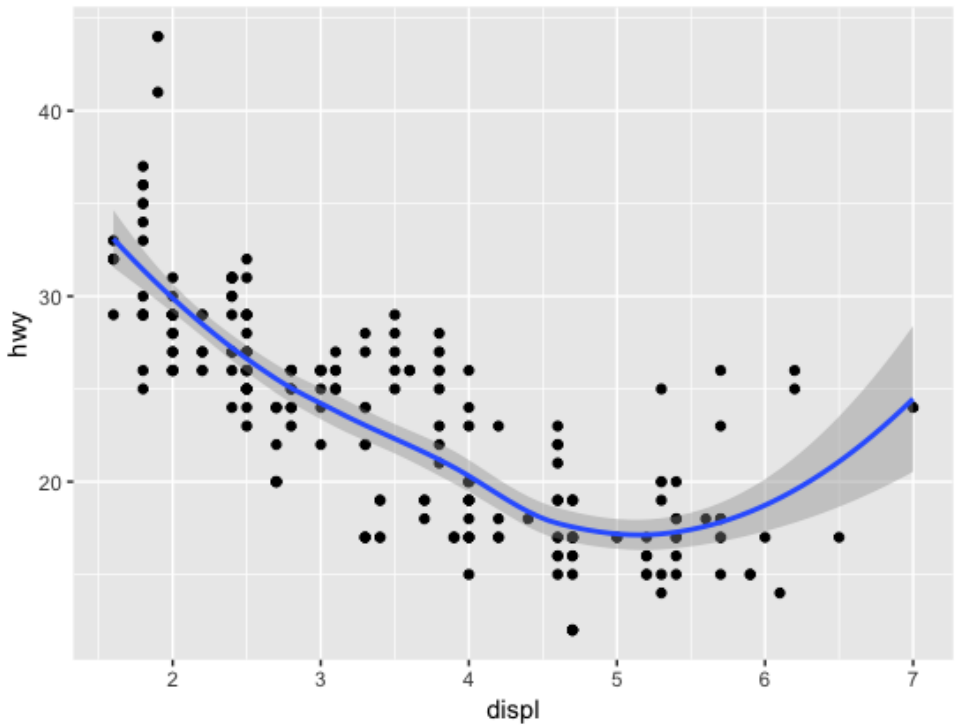
Find the difference between the three plots:

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

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

>ggplot(data = mpg) +
   geom_smooth(
  mapping = aes(x = displ, y = hwy, color = drv),
  show.legend = FALSE
  )
```

Geometric objects

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))
```



Geometric objects

- Duplication in the code.
- Avoid this type of repetition by:
  - Treating these mappings as global mappings
  - Passing the set of mappings to `ggplot()`.

Similarly:

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

Geometric objects

If we place `mappings` in a geom function, ggplot2 will treat them as local mappings for the layer.

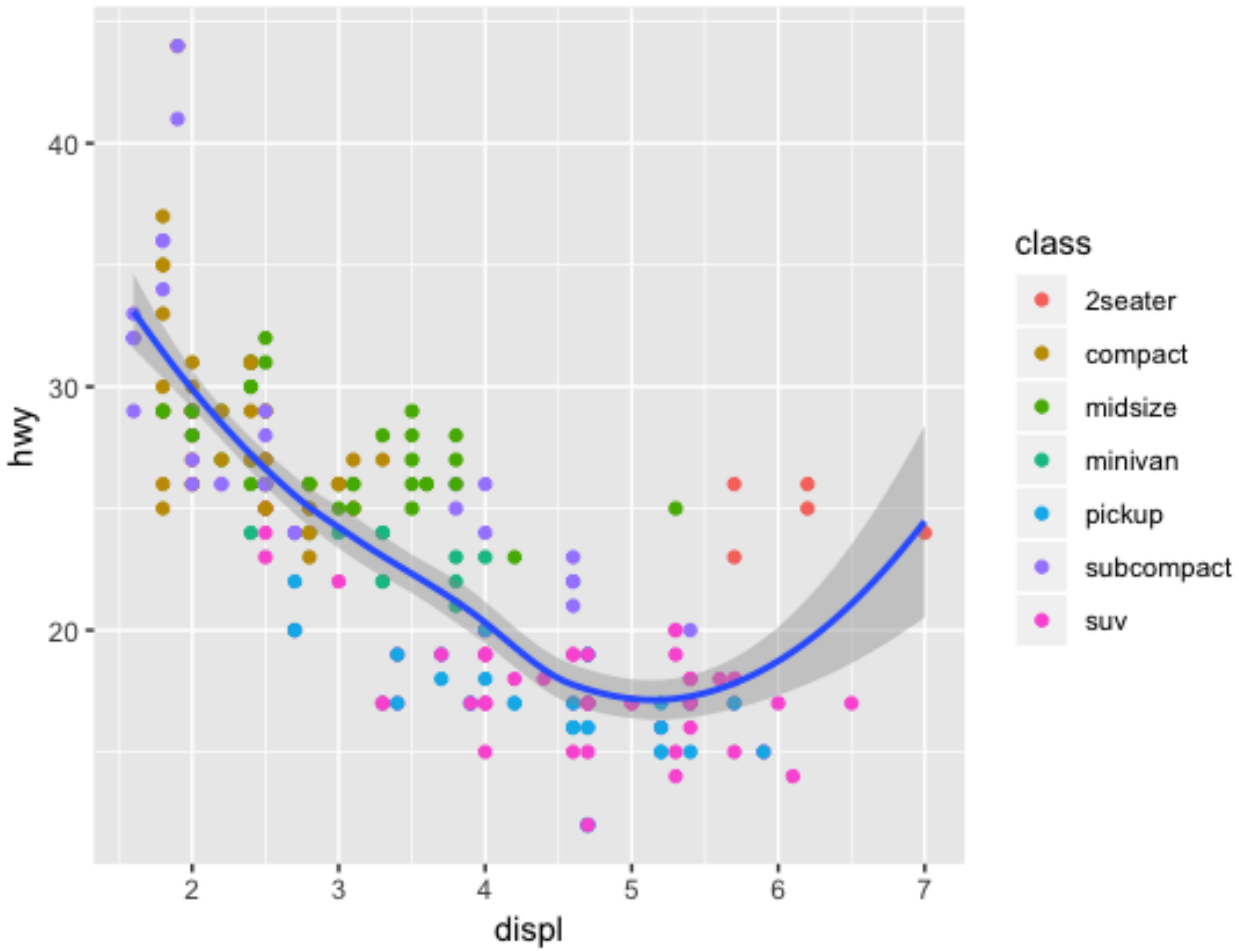
- To extend or overwrite the global mappings for that layer only.

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

Geometric objects

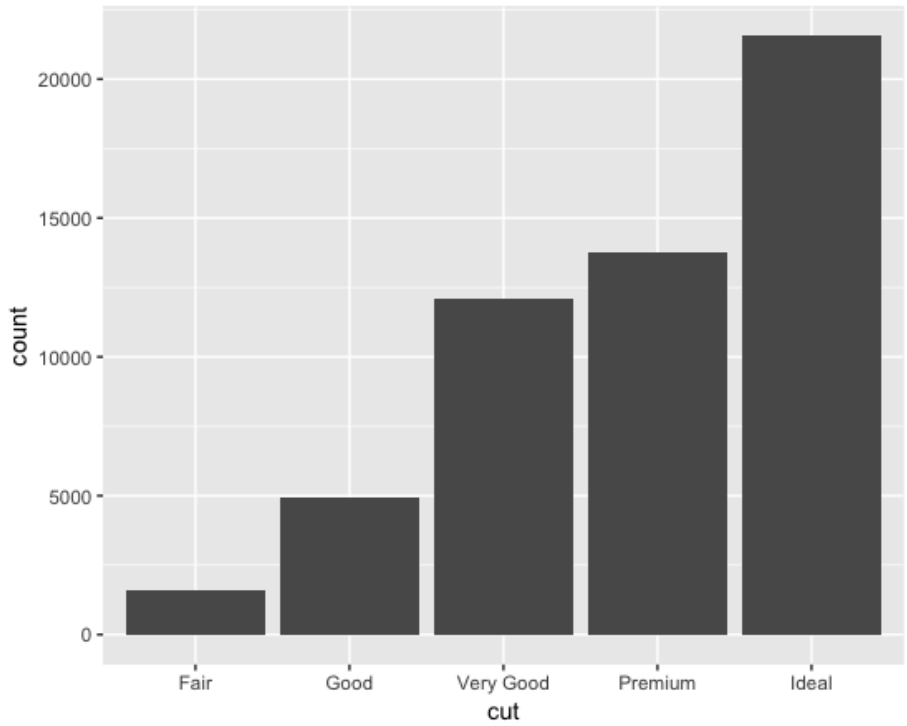
If we place `mappings` in a geom function, ggplot2 will treat them as local mappings for the layer.

- To extend or overwrite the global mappings for that layer only.



Bar chart

```
> diamonds # load dataset
>ggplot(data = diamonds) +
   geom_bar(mapping = aes(x = cut))
```



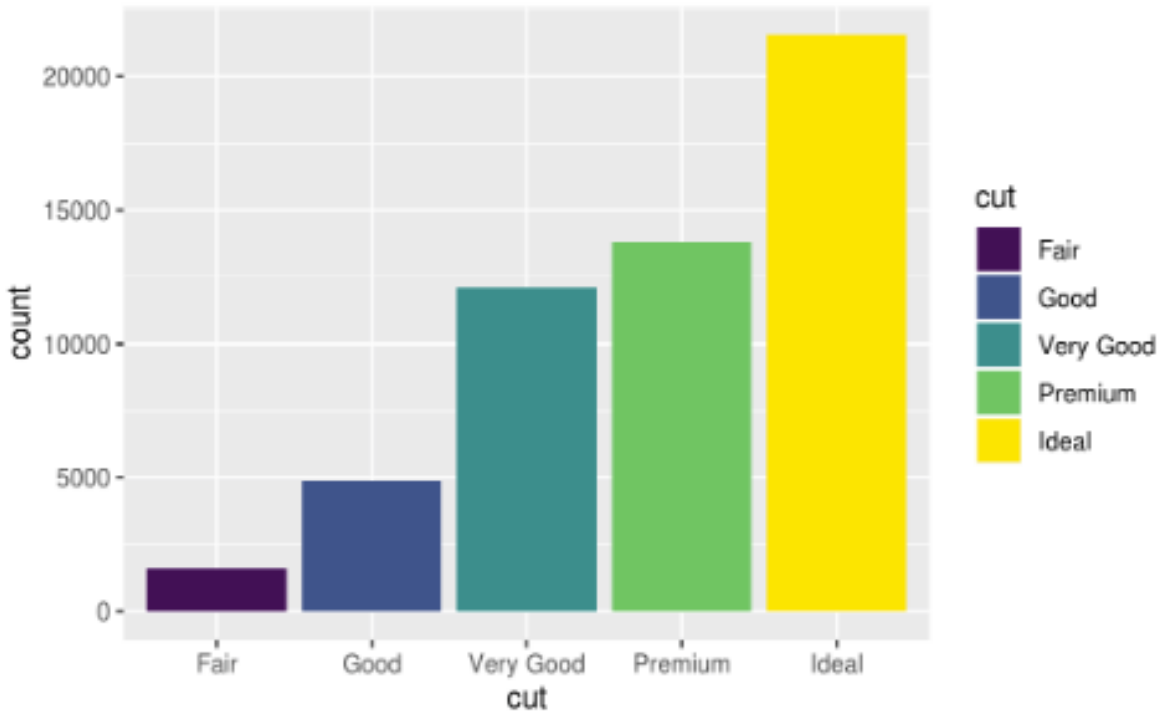
More diamonds are available with high quality cuts.

Position adjustments

Position adjustments

You can color a bar chart using `fill`:

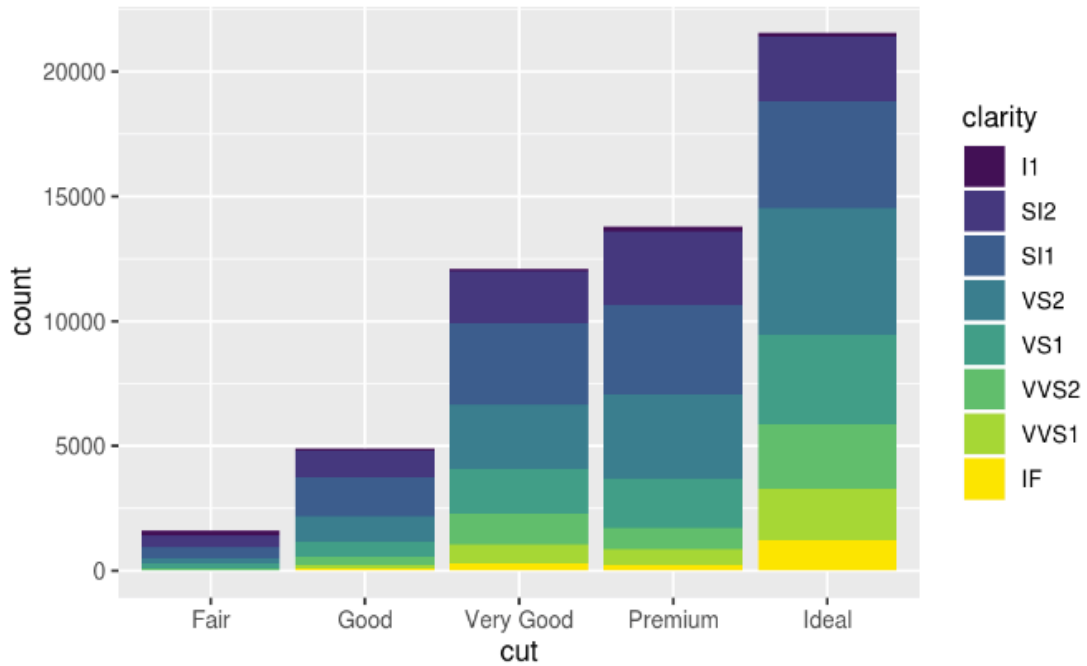
```
ggplot(data = diamonds) +
    geom_bar(mapping = aes(x = cut, fill = cut))
```



Position adjustments

Note what happens if you map the fill aesthetic to another variable, like `clarity`

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



The bars are automatically stacked. Each colored rectangle represents a combination of cut and clarity

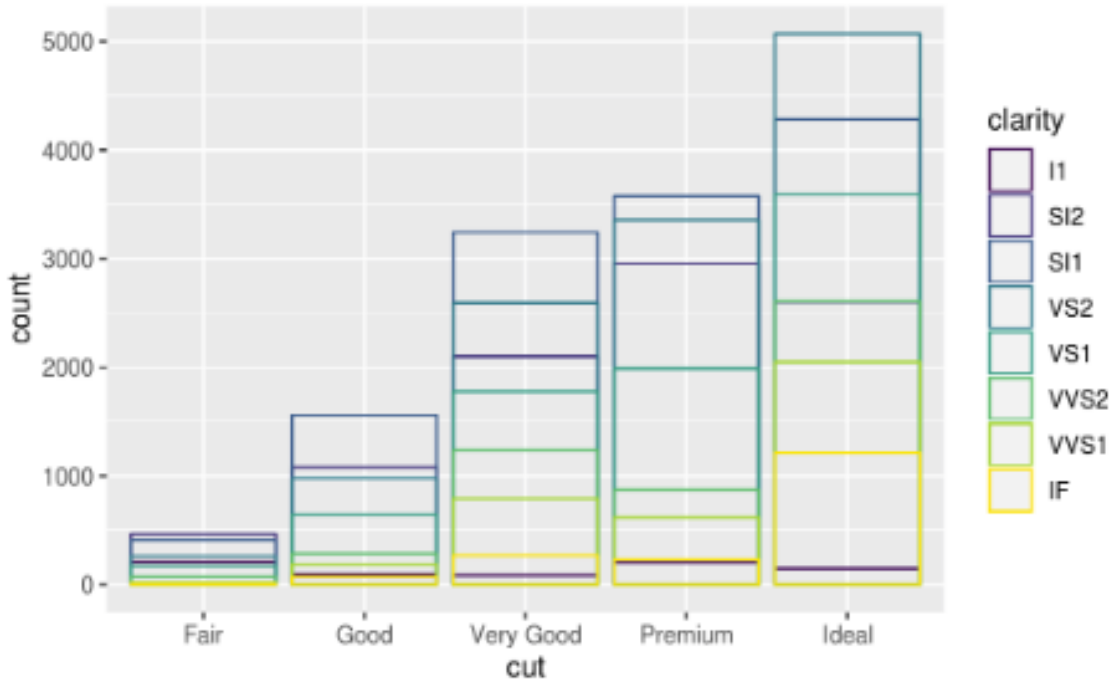
Position adjustments

We can use one of three other options:

- `position = "identity"`: will place each object exactly where it falls in the context of the graph

Position adjustments

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



To see the overlapping we need to make the bars completely transparent by setting `fill = NA`

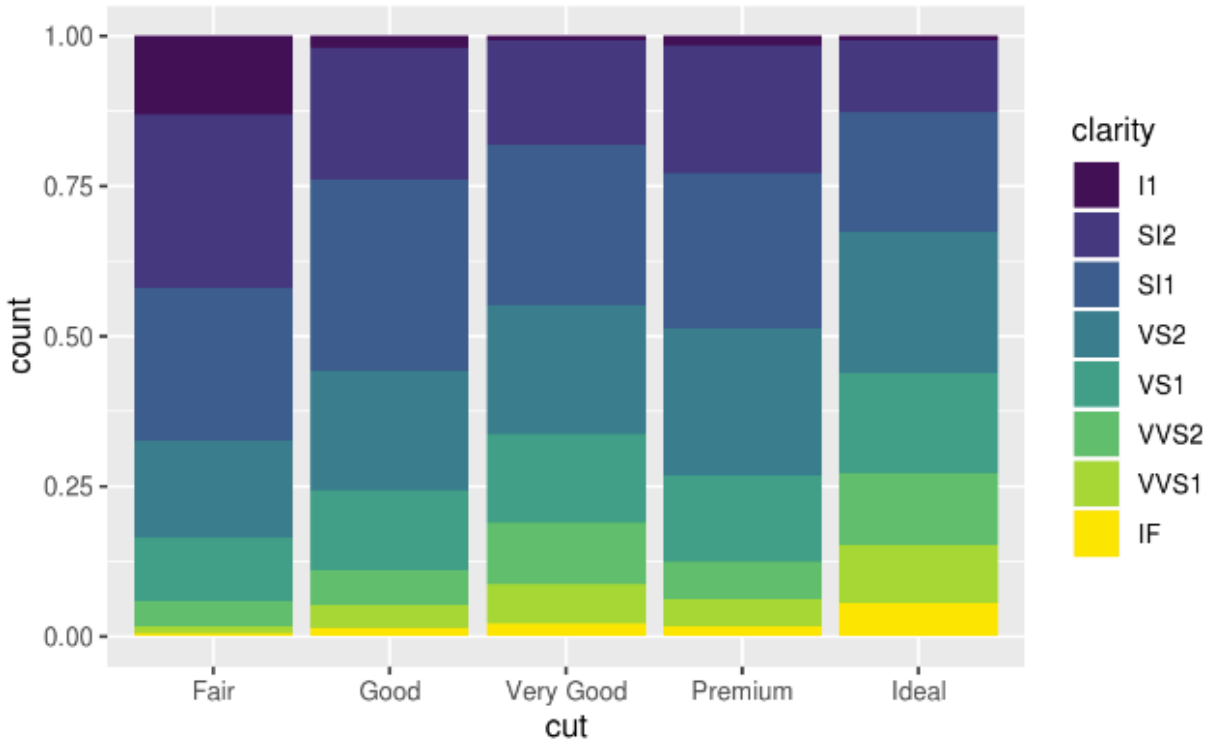
Position adjustments

If you don't want a stacked bar chart, we can use one of three other options:

- `position = "fill"`: works like stacking, but makes each set of stacked bars the same height

Position adjustments

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



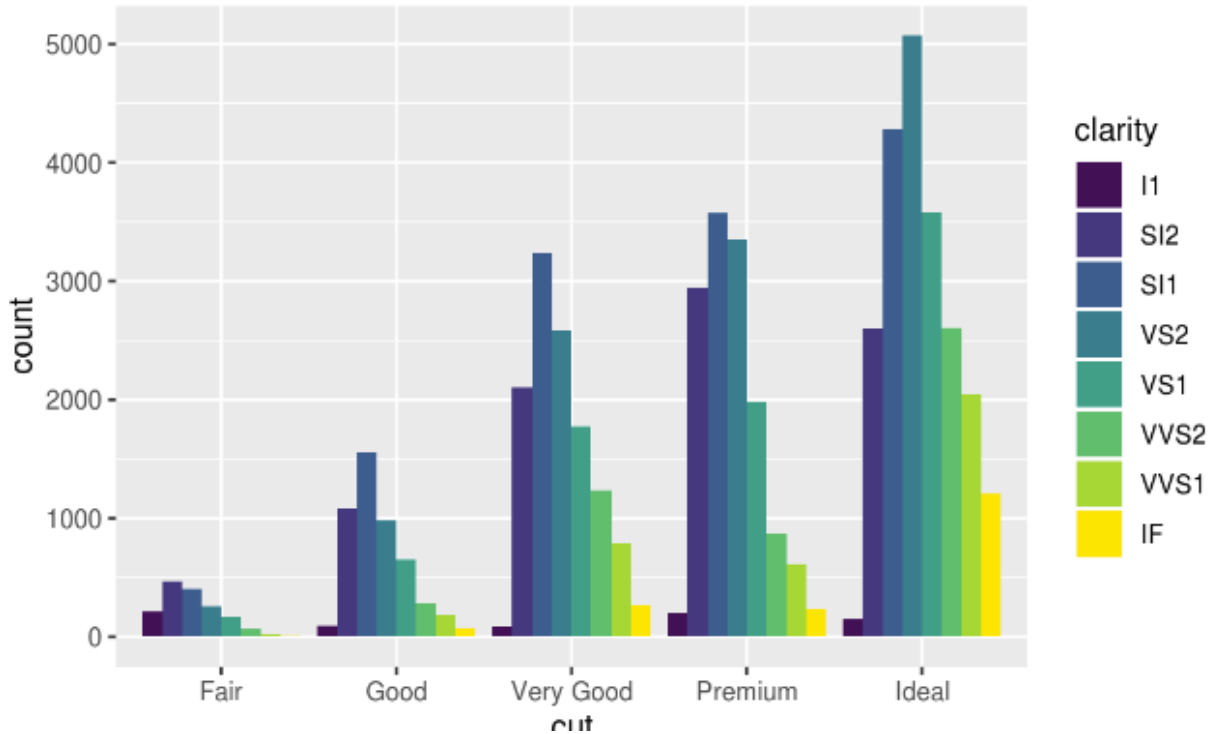
Position adjustments

If we don't want a stacked bar chart, we can use one of three other options:

- `position = "dodge"`: places overlapping objects directly beside one another

Position adjustments

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



Position adjustments

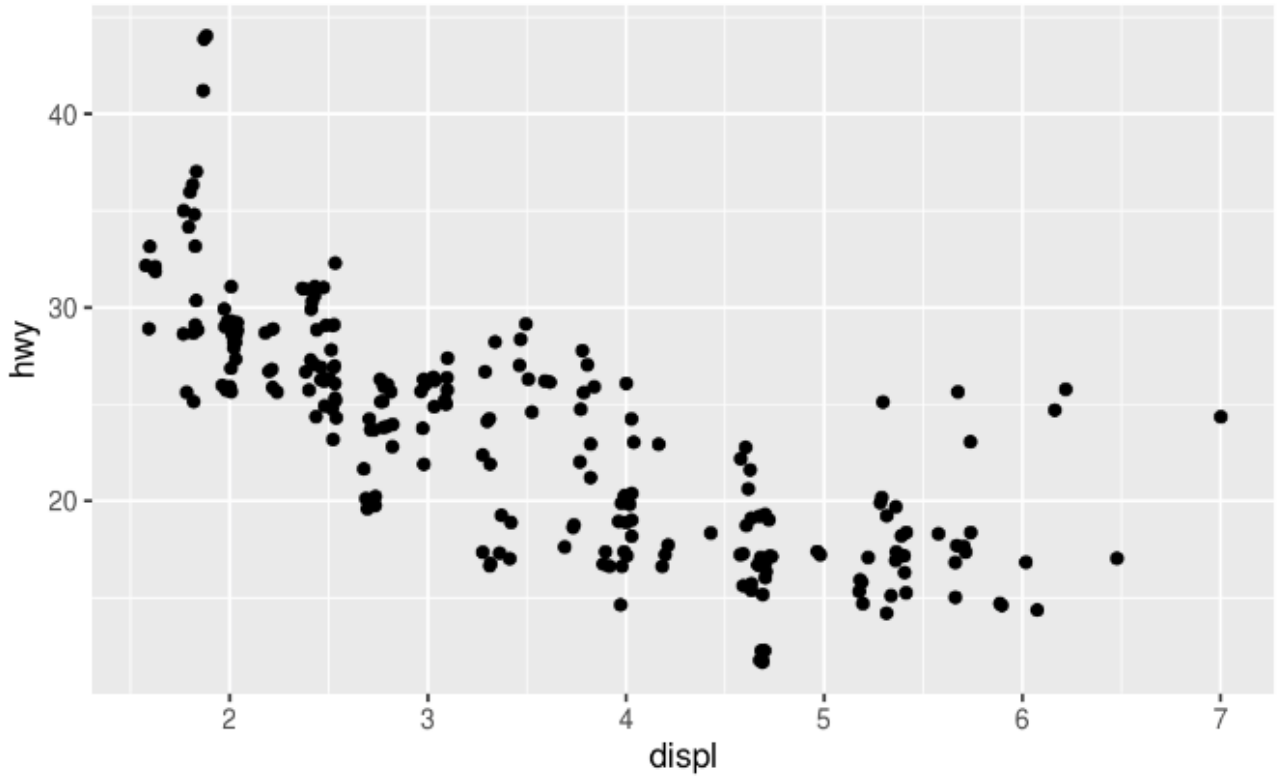
- There's one other type of adjustment that's useful for scatterplots
- Recall our first scatterplot:

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

- The plot displays only 126 points, even though there are 234 observations in the dataset
- The values of `hwy` and `displ` are rounded so the points appear on a grid and many points overlap each other
- We can avoid this gridding by setting the position adjustment to jitter. `position = "jitter"`

Position adjustments

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



Exploratory data analysis

Exploratory data analysis

- To explore the data
- Exploratory Data Analysis: EDA
- EDA is an iterative cycle:

  1. Generate questions about the data
  2. Search for answers by tidying, visualizing and modelling the data

- EDA is a state of mind.

Prerequisites

- library(tidyverse)
- library(ggplot2)

Variation

Variation

- **Variation** is the tendency of the values of a variable to change from measurement to measurement.
- If we measure any *continuous variable* twice, we will get two different results.
- *Categorical variables* can also vary if we measure across different:
  - Subjects e.g. the eye colors of different people
  - different times e.g. the energy levels of an electron at different moments

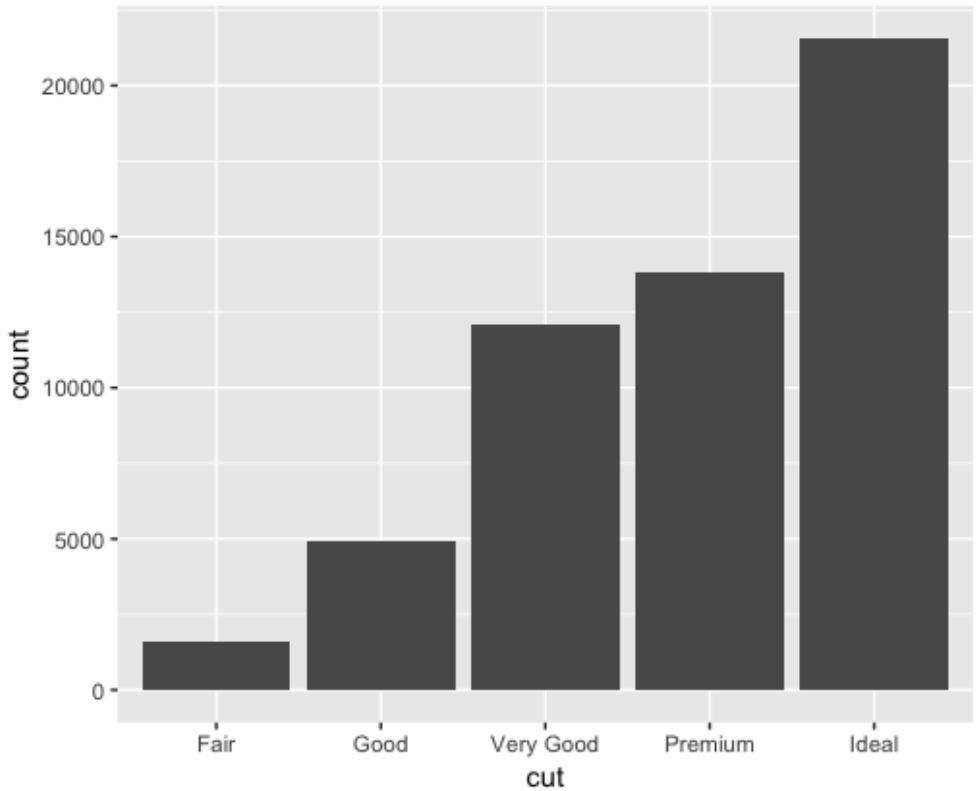
The best way to understand that pattern is to **visualize** the distribution of the variable's values.

Visualizing distributions

Bar chart

To examine the distribution of a **categorical** variable, use a **bar chart**:

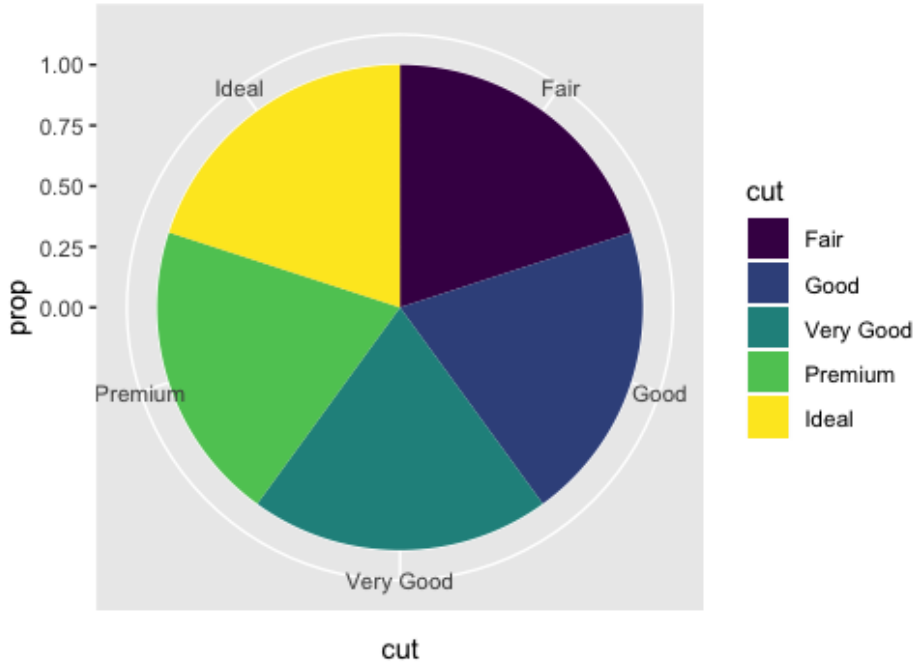
```
>ggplot(data = diamonds) +
    geom_bar(mapping = aes(x = cut))
```



Visualizing distributions

Pie chart

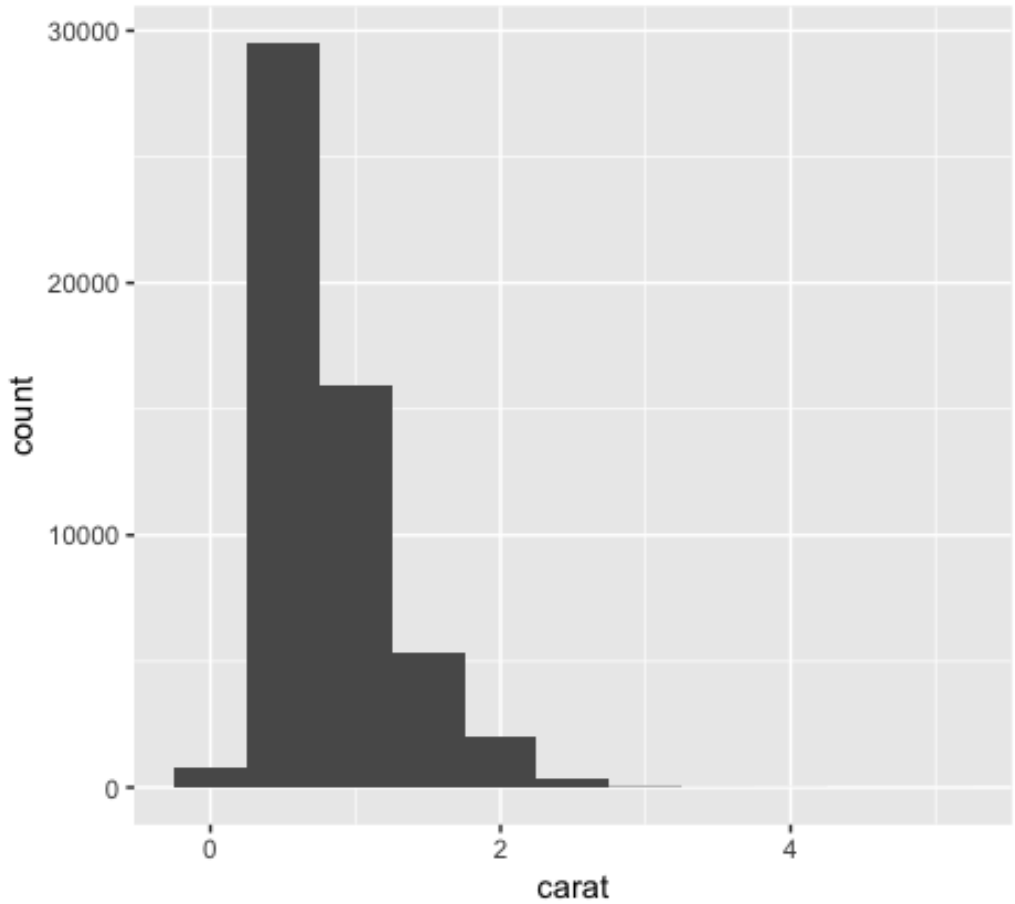
```
>ggplot(data = diamonds, mapping = aes(x = cut, y=..prop.., fill = cut)) +
    geom_bar(width = 1) +)
    coord_polar()
```



Visualizing distributions

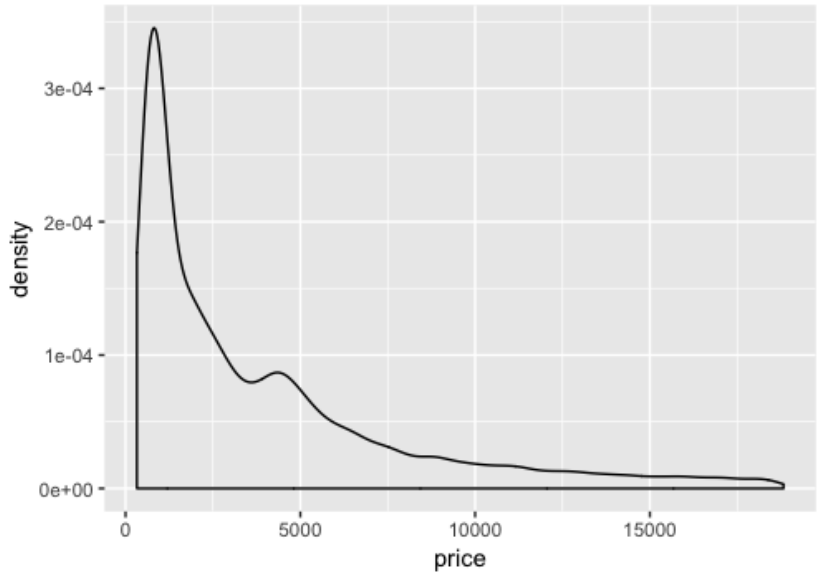
To examine the distribution of a **continuous** variable, use a **histogram**:

```
>ggplot(data = diamonds) +
    geom_histogram(mapping = aes(x = carat), binwidth = 0.5)
```



Visualizing distributions

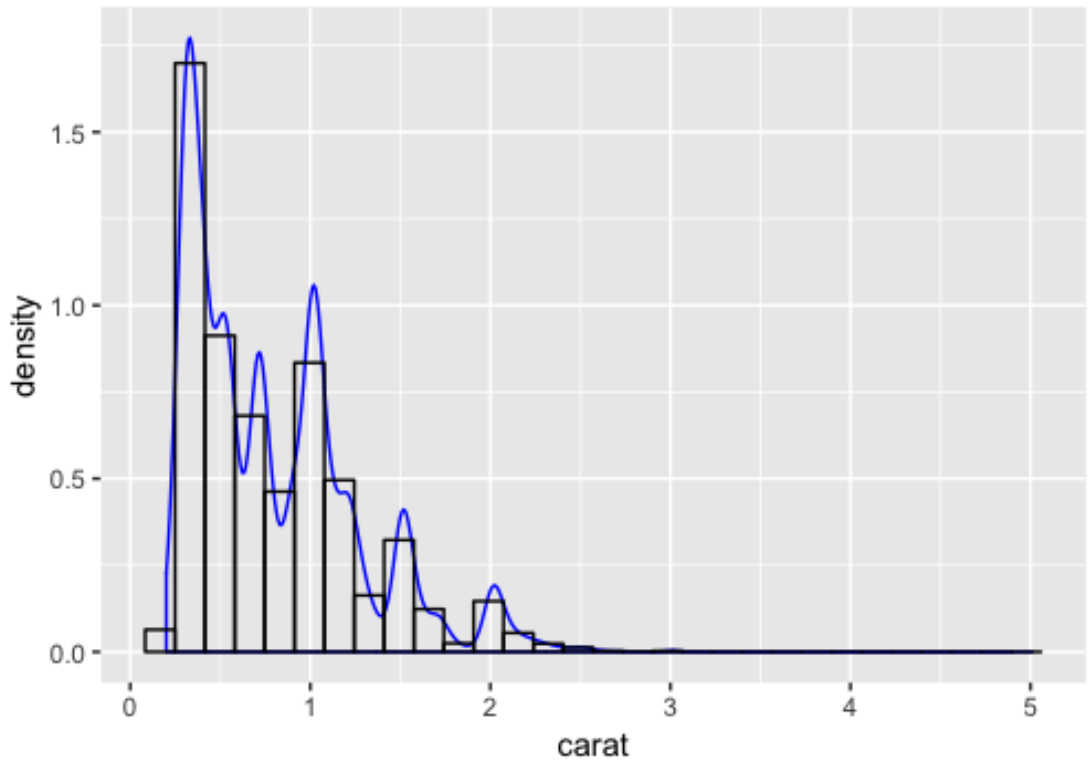
```
>ggplot(diamonds, aes(x =
price)) +
    geom_density()
```



Visualizing distributions

To show both a histogram and density plots together:

```
>ggplot(diamonds, aes(x = carat)) +
    geom_density(col="blue") +
    geom_histogram(aes(y=..density..), colour="black",
fill=NA)
```

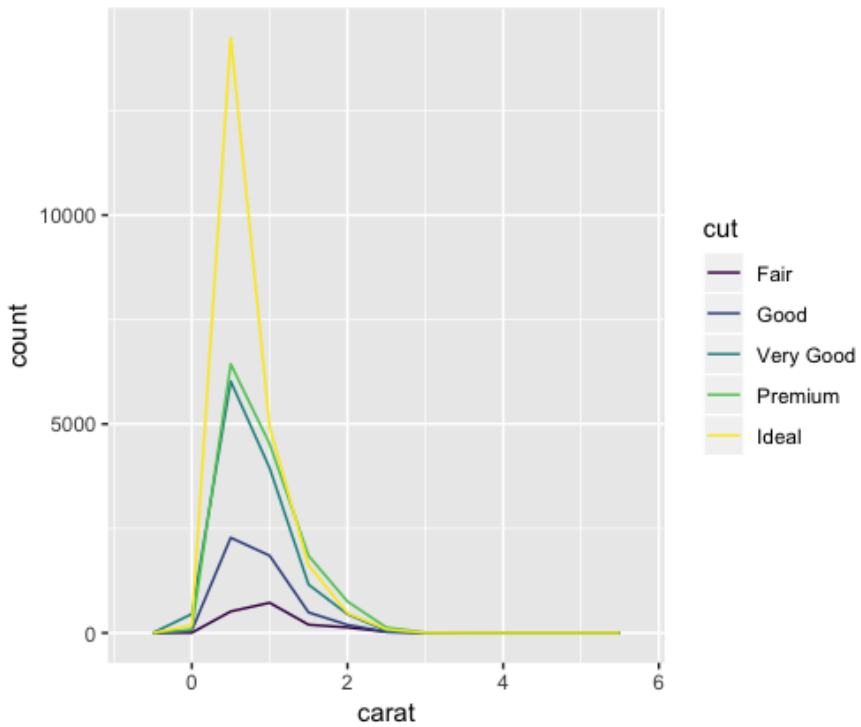


Visualizing distributions

To overlay multiple histograms in the same plot

geom_freqpoly

```
>ggplot(data = diamonds, mapping = aes(x = carat, colour =
cut)) +
    geom_freqpoly(binwidth = 0.1)
```



Covariation

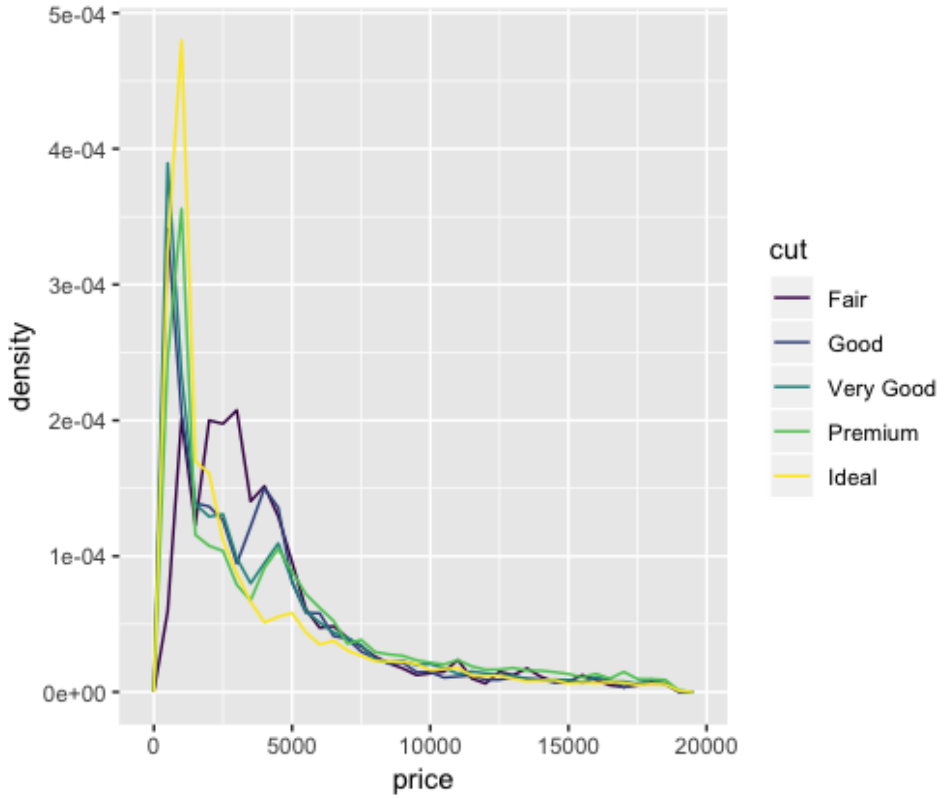
Covariation

- **Covariation** describes the behavior between variables
- It is the tendency of two or more variables to vary together in a related way

A categorical and continuous variable

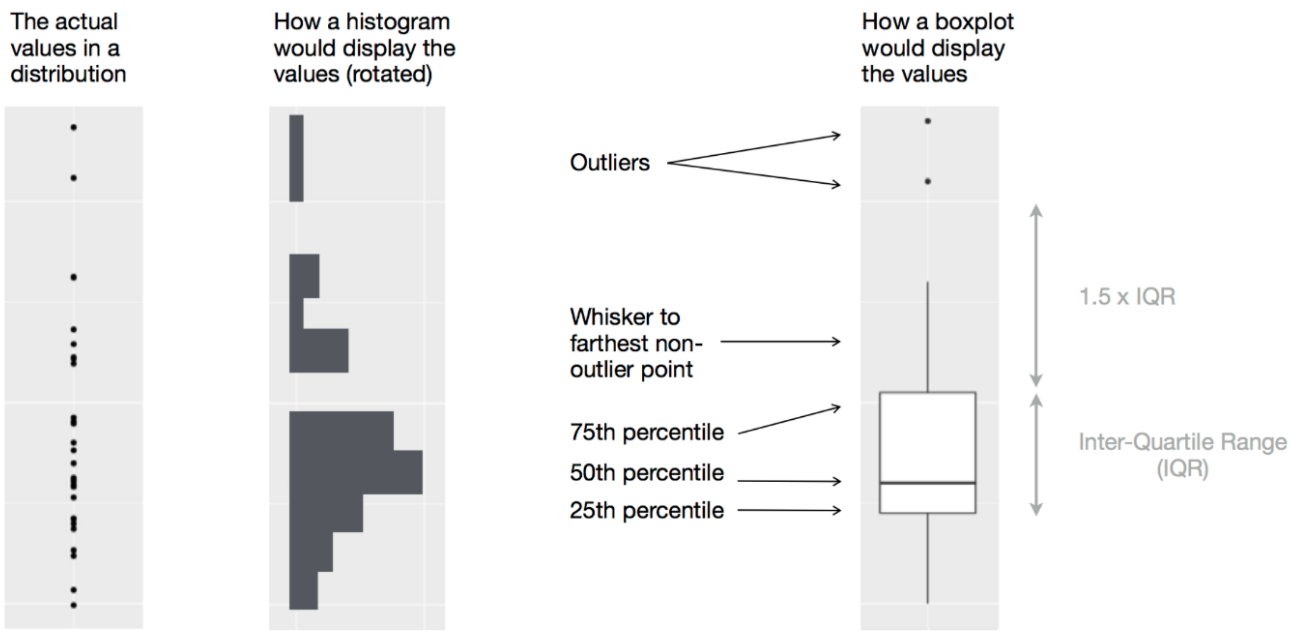
- Instead of displaying count, we'll display **density**.
- Density is the count standardized.

```
>ggplot(data = diamonds,
mapping = aes(x = price, y
= ..density..))  +
   geom_freqpoly(mapping =
aes(colour = cut),
binwidth = 500)
```



Boxplot

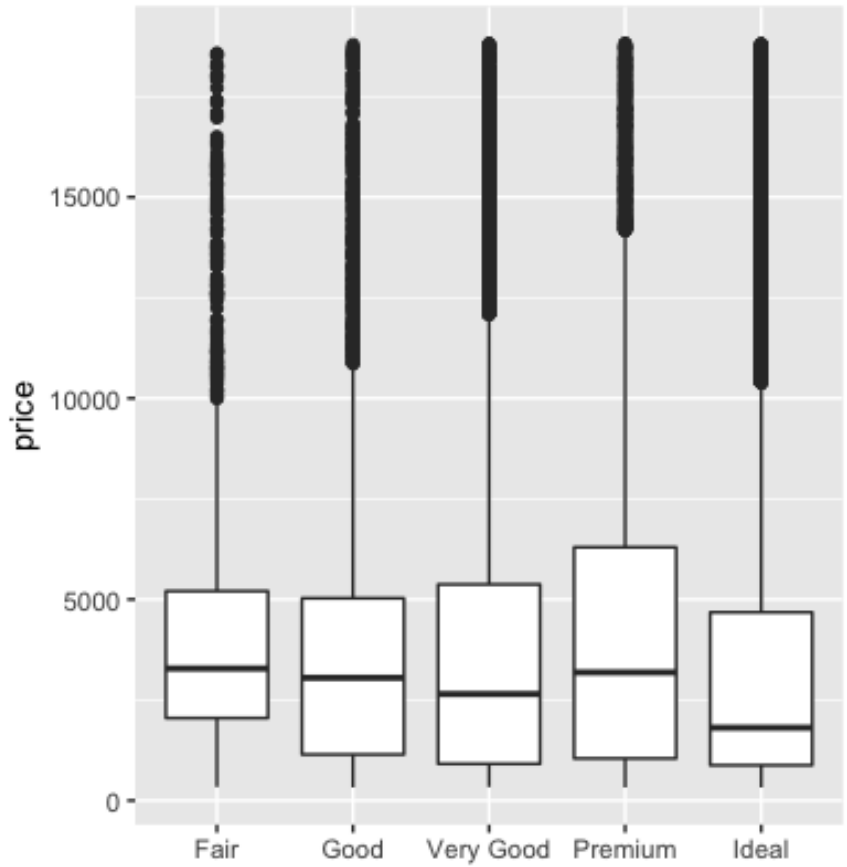
- To display the distribution of a continuous variable broken down by a categorical variable is the **boxplot**.
- A boxplot is a type of visual shorthand for a distribution of values.



Boxplot

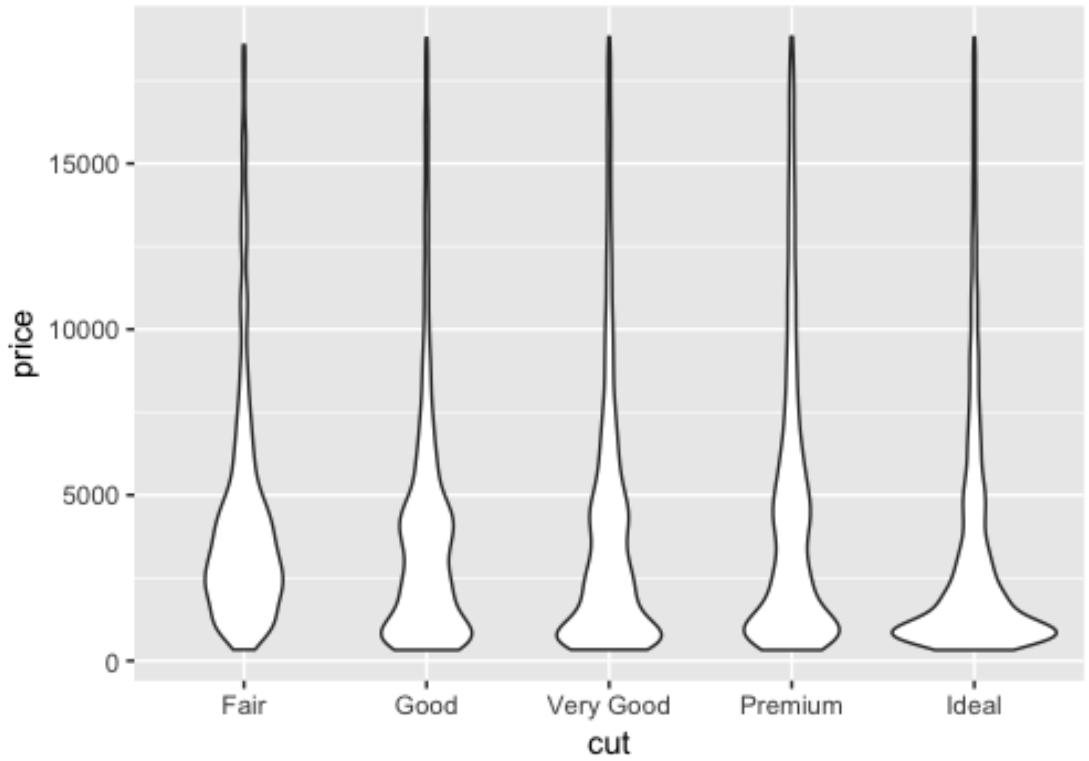
The distribution of price by cut using geom_boxplot()

```
>ggplot(data = diamonds, mapping = aes(x = cut, y =
price))+
    geom_boxplot()
```



Violin plot

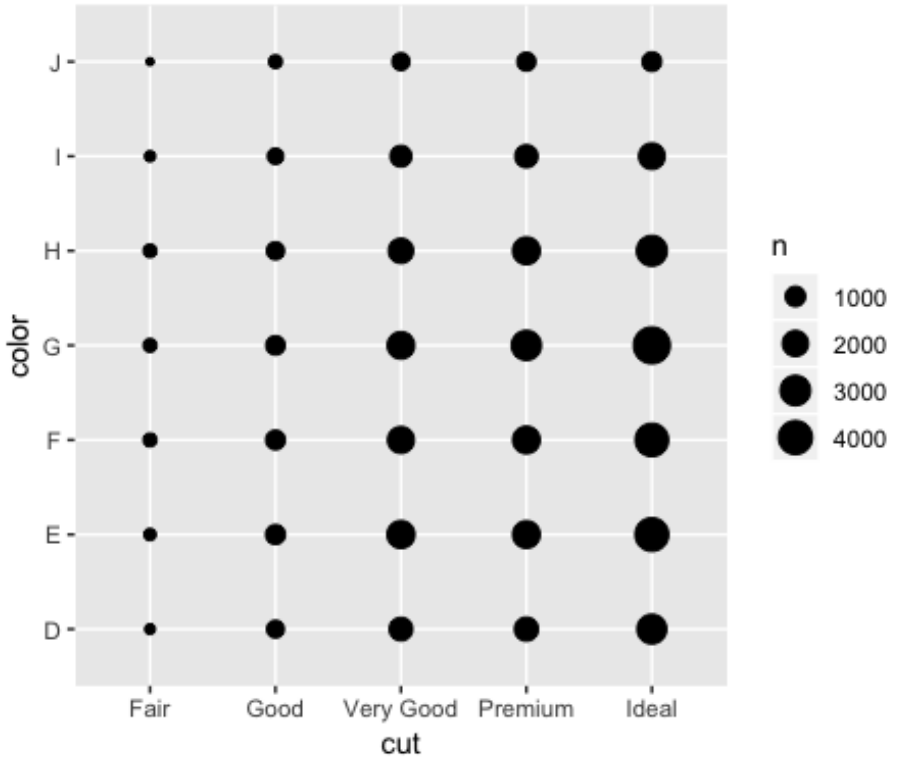
```
>ggplot(data = diamonds, mapping = aes(x = cut, y = price))
+
   geom_violin()
```



Two categorical variables

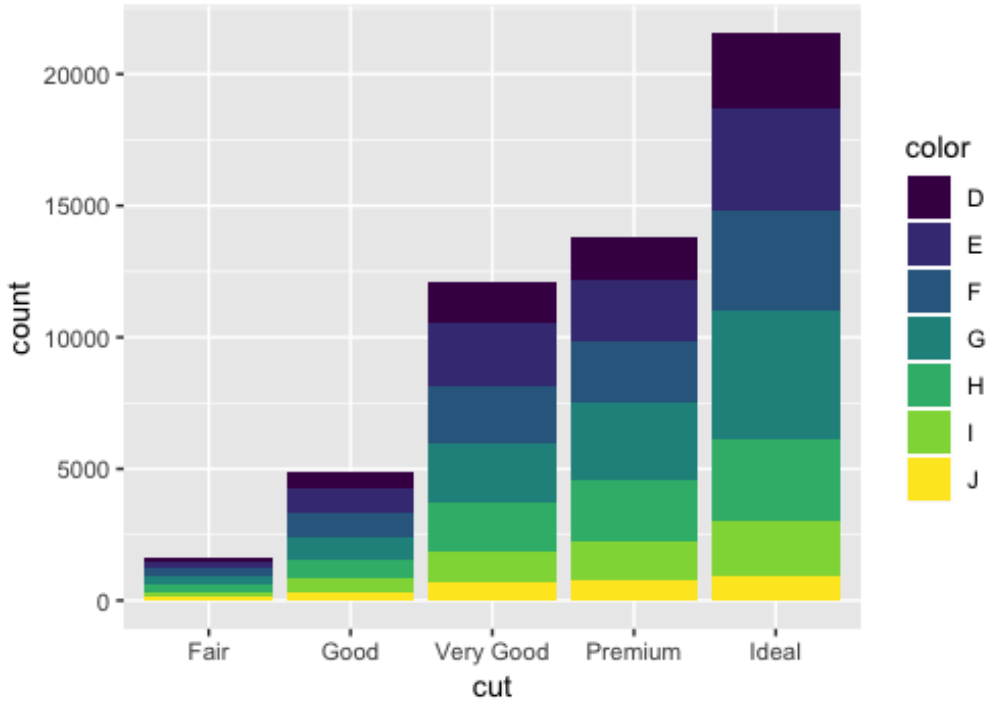
- To visualize the covariation between categorical variables.
- To count the number of observations for each combination.

```
>ggplot(data = diamonds) +
   geom_count(mapping = aes(x = cut, y = color))
```



Two categorical variables

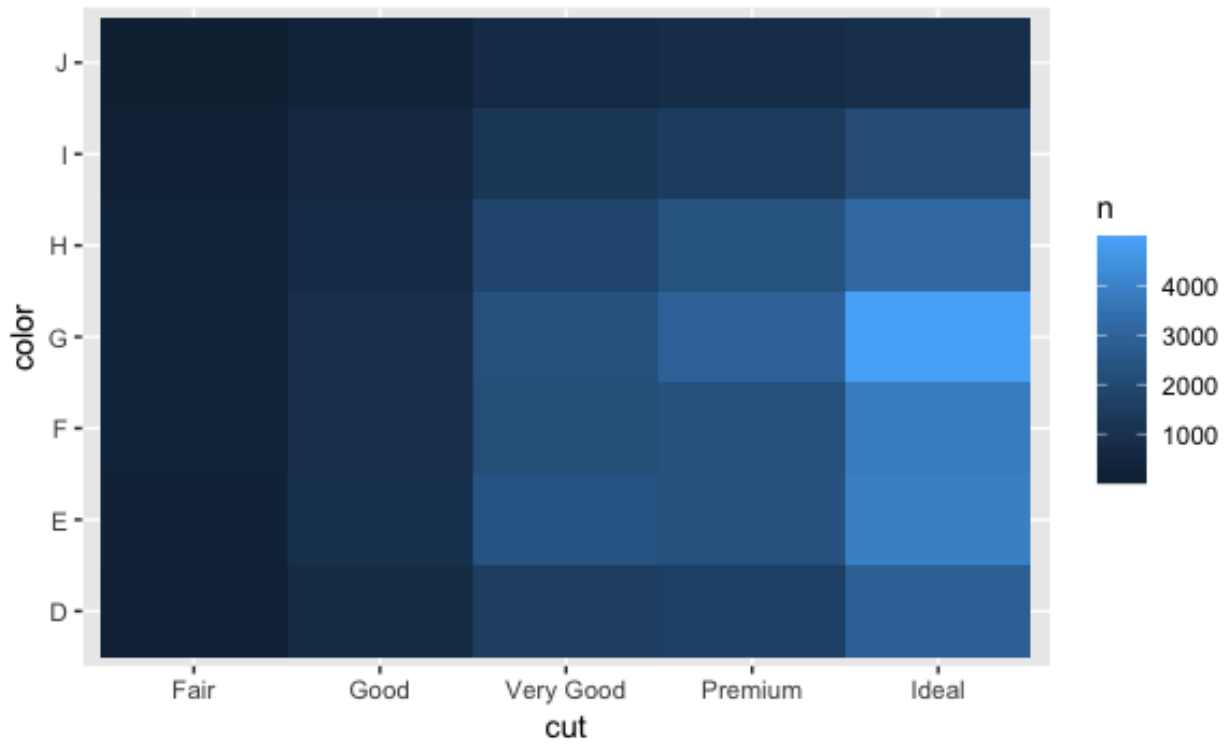
```
>ggplot(data = diamonds, aes(x = cut, fill = color)) +
   geom_bar()
```



Two categorical variables

Another approach:

```
>diamonds %>%
  count(color, cut) %>%
  ggplot(mapping = aes(x = cut, y = color)) +
  geom_tile(mapping = aes(fill = n))
```



Two continuous variables

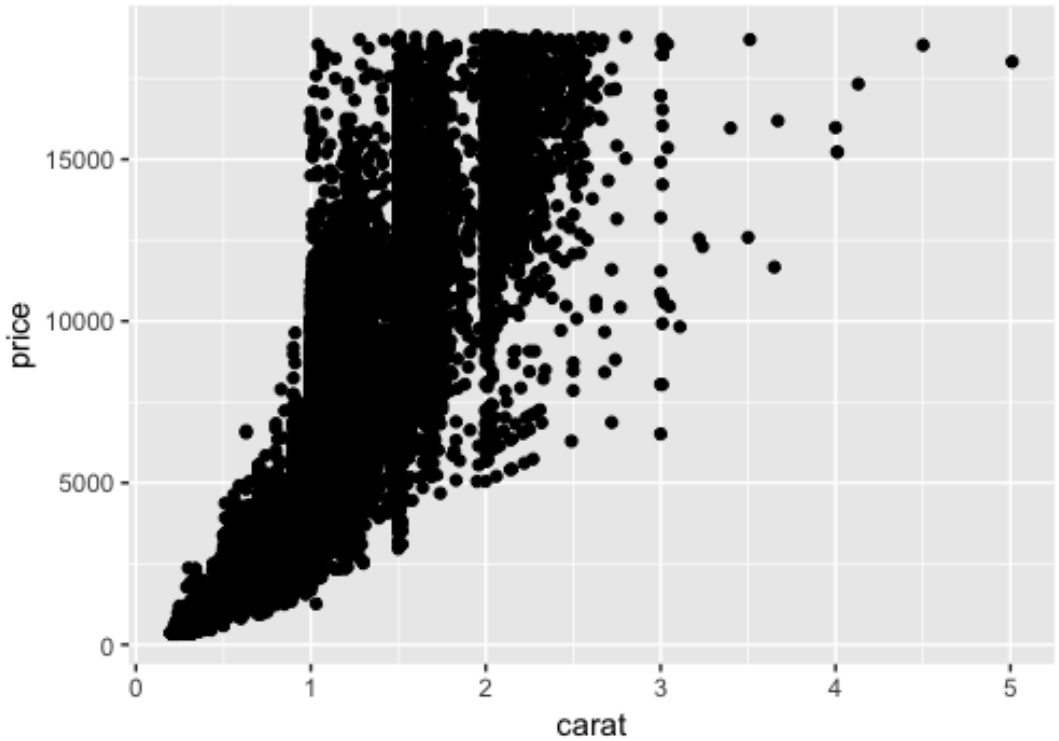
- Previously we used `geom_histogram()` and `geom_freqpoly()` to bin in **one** dimension.
- `geom_bin2d()` and `geom_hex()` to bin in **two** dimensions:

- `geom_bin2d()` and `geom_hex()` divide the coordinate plane into 2d bins.
- use a fill color to display how many points fall into each bin.

- `geom_bin2d()` creates rectangular bins.
- `geom_hex()` creates hexagonal bins.

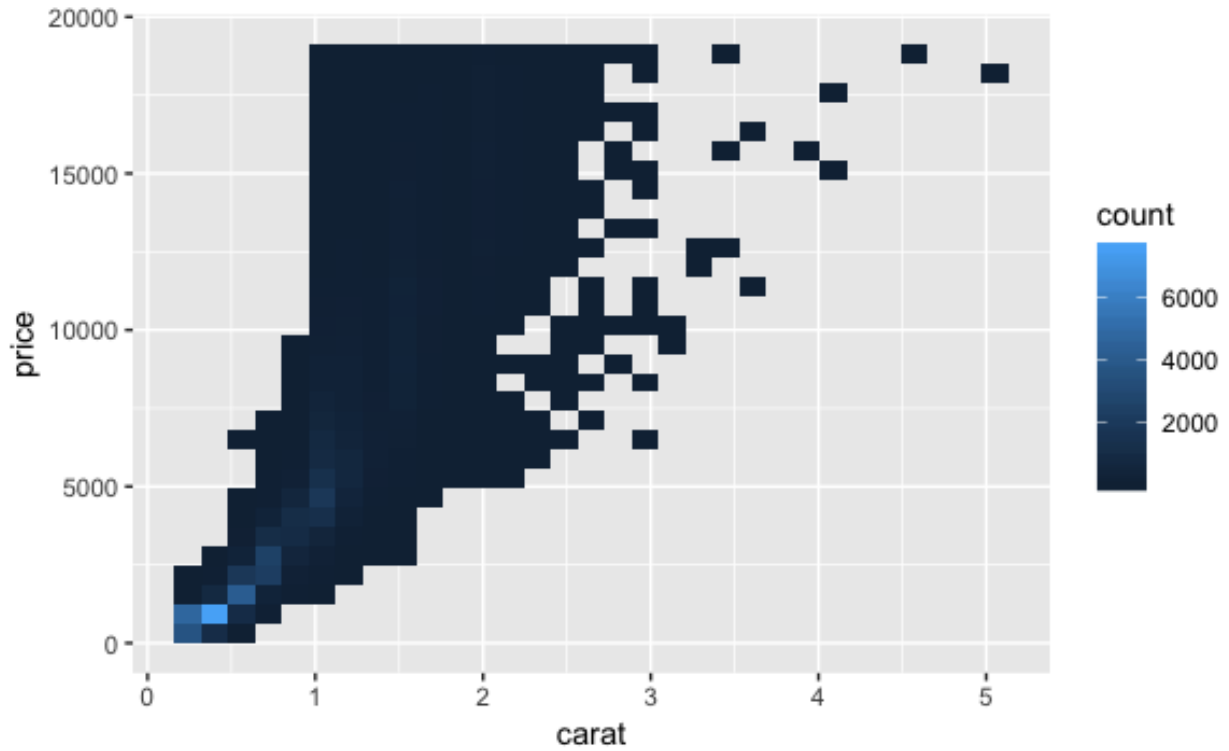
Example

```
# Scatterplot
>ggplot(diamonds, aes(x = carat, y = price)) +
    geom_point()
```



Example

```
>ggplot(data = diamonds) +
    geom_bin2d(mapping = aes(x = carat, y = price))
```



Example

```
# install.packages("hexbin") >ggplot(data = diamonds) +
    geom_hex(mapping = aes(x = carat, y = price))
```

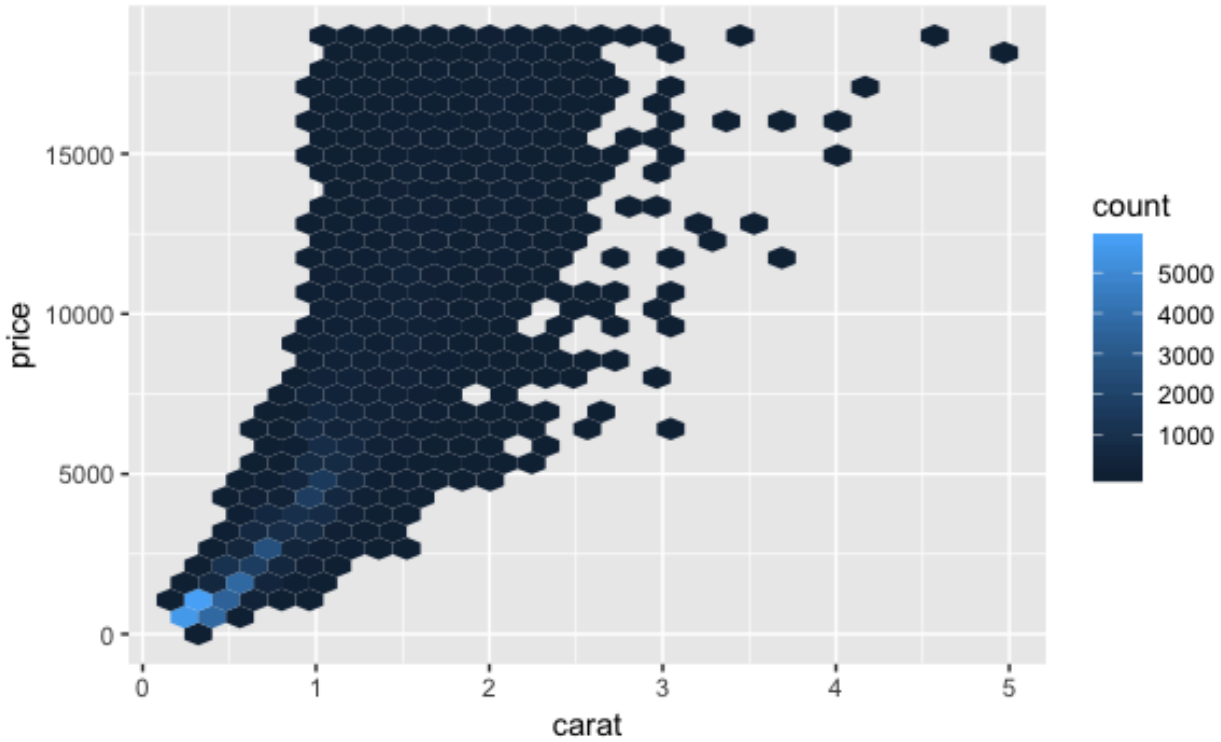
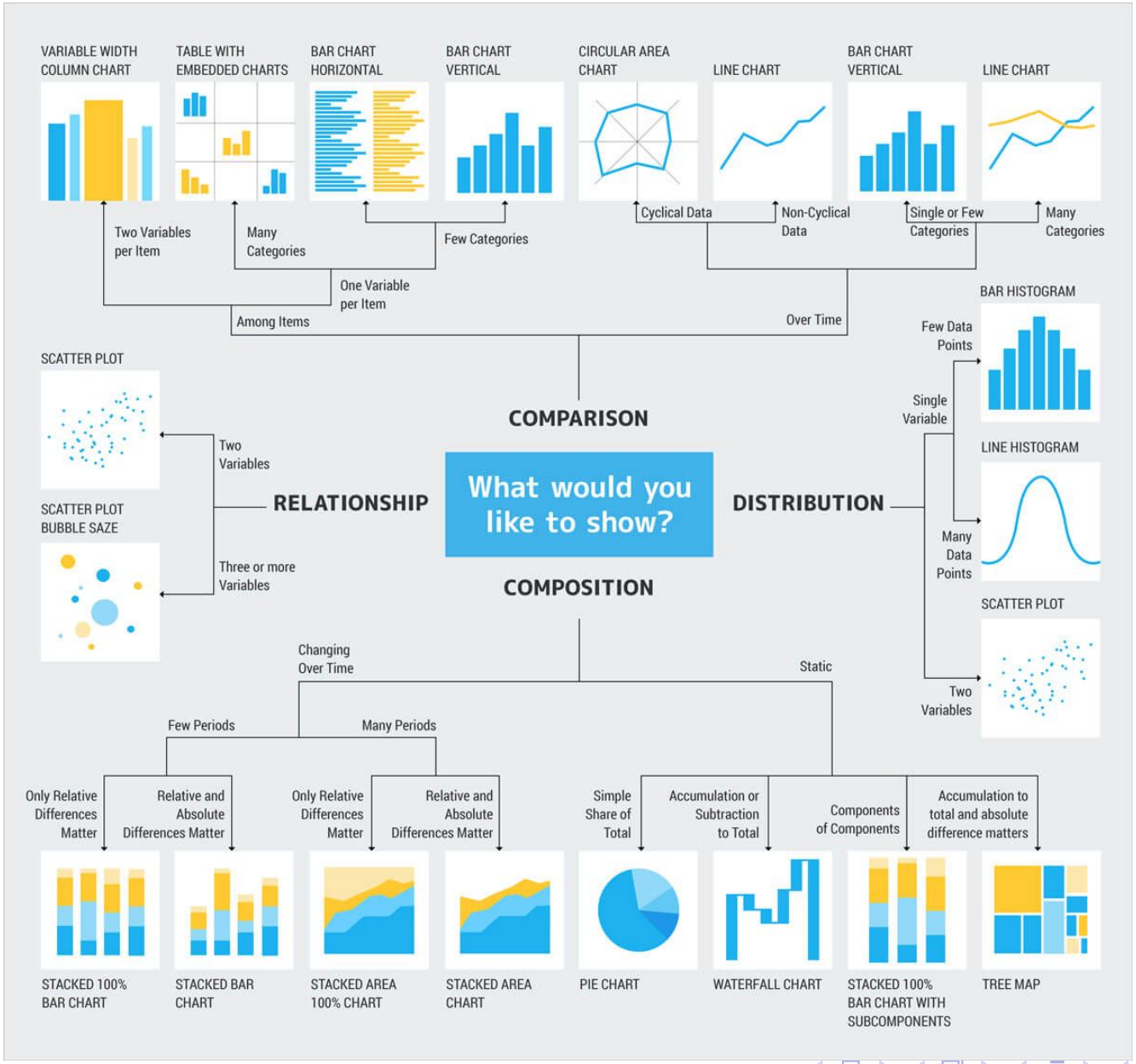


Chart type hierarchy



References

Hadley Wickham & Garrett Grolemund (2017)

R for data science: import, tidy, transform, visualize, and model data

O'Reilly.

Bret Larget (2014)

Chapter 2 R ggplot2 Examples

http://www.stat.wisc.edu/~larget/stat302/chap2.pdf

Visualization

Data Visualization in R

https://rstudio-pubs-static.s3.amazonaws.com/265574_fd78dcb867084d77918fe02525f0225e.html#labeling_plots.

Robin Donatello (2018)

Full Data Visualization tutorial

https://norcalbiostat.github.io/MATH130/full_data_viz_tutorial.html

The End