Columbia University
in the City of New York

Lecture 4: Data Visualization I

Data Science for Business Analytics

Thibault Vatter <thibault.vatter@unil.ch>

Department of Statistics, Columbia University and HEC Lausanne, UNIL

11.03.2018

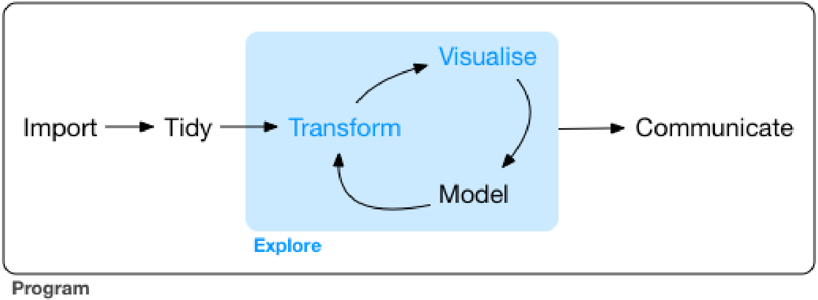
Outline

1 Overview

2 Bad graphs

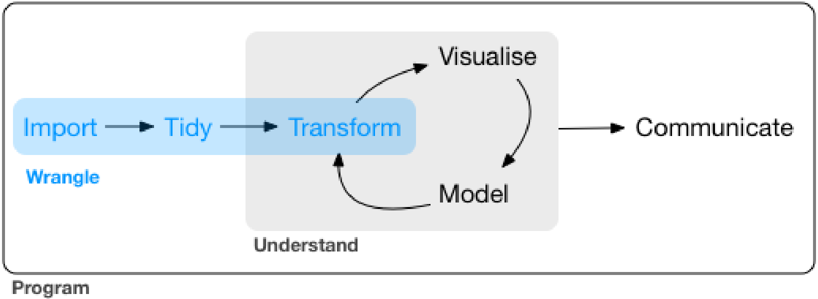
3 A grammar of graphics

Today

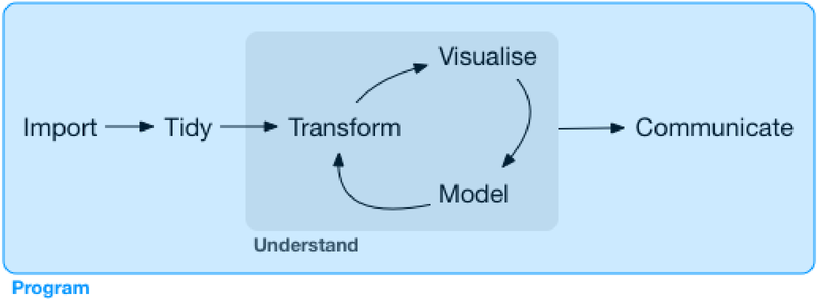


source: R for Data Science (like most figures in what follows)

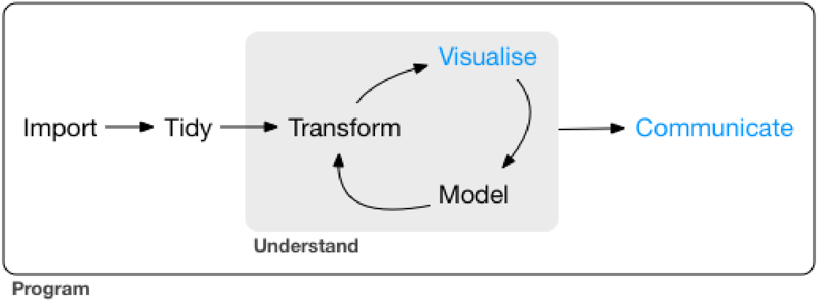
This morning



This morning



This afternoon



Outline

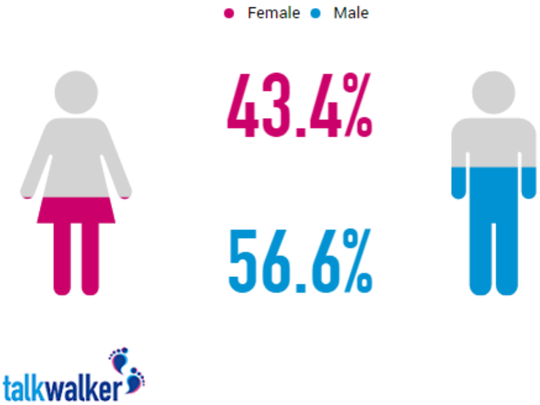
1 Overview

2 Bad graphs

3 A grammar of graphics

Data content

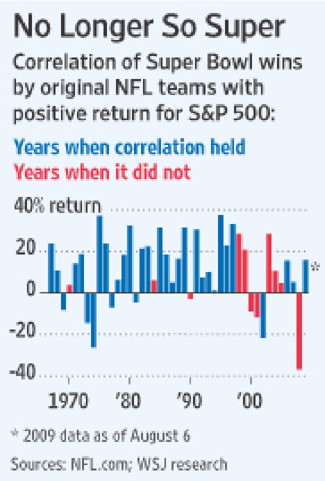
- Makes no sense to use graphs for very small amounts of data.
- The human brain is capable of grasping a few values.



source: talkwalker.com

Data relevance

- Graphs are only as good as the data they display.
- No creativity can produce a good graph from poor data.
- Leinweber (the author of *Nerds on Wall Street*) showed that the S&P500 could be “predicted” at 75% by the butter production in Bangladesh (or 99% when adding cheese in USA, and the population of sheep).



No Longer So Super

Correlation of Super Bowl wins by original NFL teams with positive return for S&P 500:

* 2009 data as of August 6

Sources: NFL.com; WSJ research

Complexity

- Graphs shouldn't be more complex than the data they portray.
- Unnecessary complexity can be introduced by irrelevant
  - decoration
  - colour
  - 3d effects
- These are collectively known as "chartjunk".

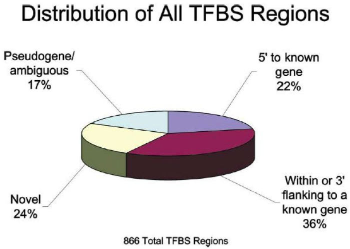


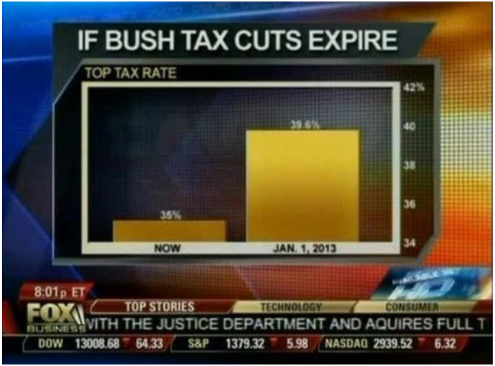
Figure 1. Classification of TFBS Regions

TFBS regions for Sp1, cMyc, and p53 were classified based upon proximity to annotations (RefSeq, Sanger hand-curated annotations, GenBank full-length mRNAs, and Ensembl predicted genes). The proximity was calculated from the center of each TFBS region. TFBS regions were classified as follows: within 5 kb of the 5′ most exon of a gene, within 5 kb of the 3′ terminal exon, or within a gene, novel or outside of any annotation, and pseudogene/ambiguous (TFBS overlapping or flanking pseudogene annotations, limited to chromosome 22, or TFBS regions falling into more than one of the above categories).

source: Cawley S, et al. (2004), Cell 116:499-509, Figure 1

Distorsion

- Graphs shouldn't be distorted pictures of the portrayed values.
- Distortion can be either deliberate or accidental.
- It is useful to know how to produce truth bending graphs.
- Sometimes, misleading is used as a synonym of distorted.



source: tatisticshowto.com/misleading-graphs/

More on distortion

Common sources of distortion:

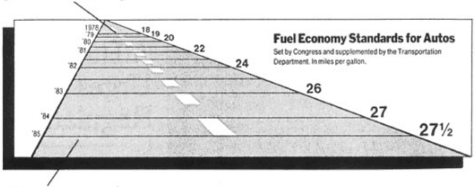
- 3 dimensional “effects”
- linear scaling when using area or volume to represent values

The “lie factor”:

- Measure of the amount of distortion in a graph (don’t take this too seriously) defined by Ed Tufte of Yale University
- lie factor $= \frac{\text{size of effect shown in graphic}}{\text{size of effect shown in data}}$
- If the lie factor of a graph is greater than 1, the graph is exaggerating the size of the effect.

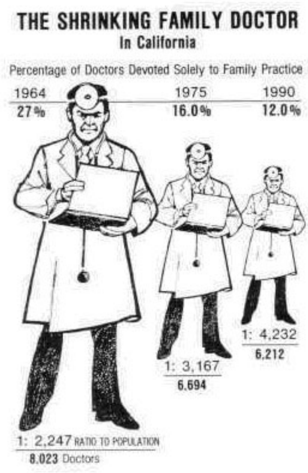
More on the lie factor

This line, representing 18 miles per gallon in 1978, is 0.6 inches long.



This line, representing 27.5 miles per gallon in 1985, is 5.3 inches long.

$$\text{lie factor} = \frac{\frac{5.3-0.6}{0.6}}{\frac{27.5-18}{18}} = 14.8$$



lie factor = 2.8

Drawing good graphs

The three main rules:

- If the “story” is simple, keep it simple.
- If the “story” is complex, make it look simple.
- Tell the truth – do not distort the data.

Specifically:

- There should be a high data to chart ratio.
- Use the appropriate graph for the appropriate purpose.
  - Most graphs presented in Excel are POOR CHOICES!
  - In particular, never use a pie chart!
- Make sure that the graph is complete (e.g., all axes must be labeled, there should be a title).

Outline

1 Overview

2 Bad graphs

3 A grammar of graphics

A grammar of graphics

> *"A grammar of graphics is a tool that enables us to concisely describe the components of a graphic. Such a grammar allows us to move beyond named graphics (e.g., the"scatterplot") and gain insight into the deep structure that underlies statistical graphics." — Hadley Wickham*

`ggplot2` is an R implementation of the concept:

- A coherent system for describing and building graphs, based on The Grammar of Graphics.
- Do more faster by learning one system and applying it in many places.

To learn more, read The Layered Grammar of Graphics.

The mpg data frame

Data from the US EPA on 38 models of car:

```
mpg
```

```
## # A tibble: 234 x 11
##    manufacturer model displ  year   cyl trans drv     cty   hwy fl
##    <chr>        <chr> <dbl> <int> <int> <chr> <chr> <int> <int> <chr>
##  1 audi         a4     1.80  1999     4 auto~ f        18    29 p
##  2 audi         a4     1.80  1999     4 manu~ f        21    29 p
##  3 audi         a4     2.00  2008     4 manu~ f        20    31 p
##  4 audi         a4     2.00  2008     4 auto~ f        21    30 p
##  5 audi         a4     2.80  1999     6 auto~ f        16    26 p
##  6 audi         a4     2.80  1999     6 manu~ f        18    26 p
##  7 audi         a4     3.10  2008     6 auto~ f        18    27 p
##  8 audi         a4 q~  1.80  1999     4 manu~ 4        18    26 p
##  9 audi         a4 q~  1.80  1999     4 auto~ 4        16    25 p
## 10 audi         a4 q~  2.00  2008     4 manu~ 4        20    28 p
## # ... with 224 more rows, and 1 more variable: class <chr>
```

Among the variables in mpg are:

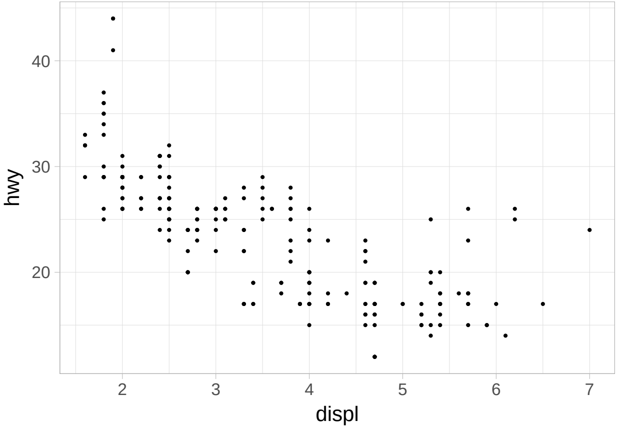
1. displ, a car's engine size, in litres.
2. hwy, a car's fuel efficiency on the highway (in miles per gallon).

A few questions

- Do cars with big engines use more fuel than cars with small engines?
- What does the relationship between engine size and fuel efficiency look like?
- Is it positive? Negative? Linear? Nonlinear?

Creating a ggplot

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

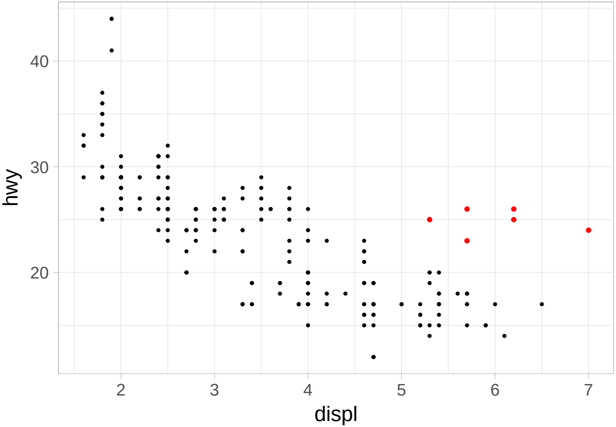


A graphing template

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

Aesthetic mappings

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

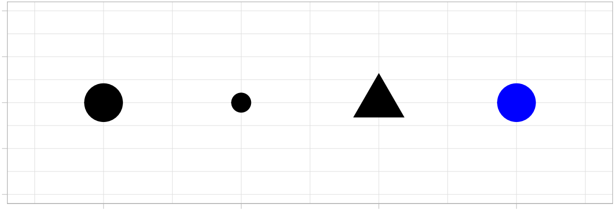


Aesthetic

One could add a third variable to a two dimensional scatterplot by mapping it to an **aesthetic**:

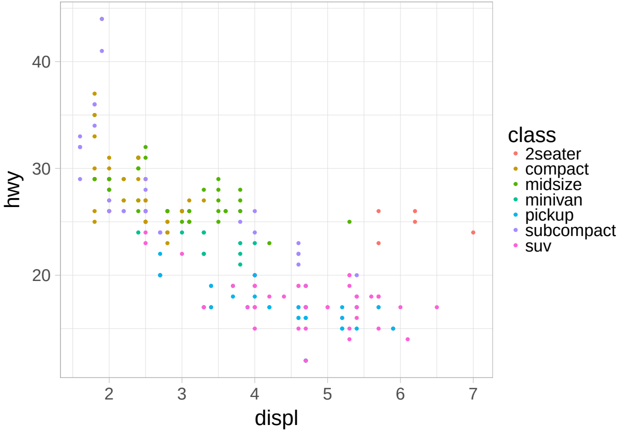
- a visual property of the objects in your plot
- include things like the size, the shape, or the color of your points

We use the word **"value"** to describe data and **"level"** to describe aesthetic properties.



Adding classes to your plot

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

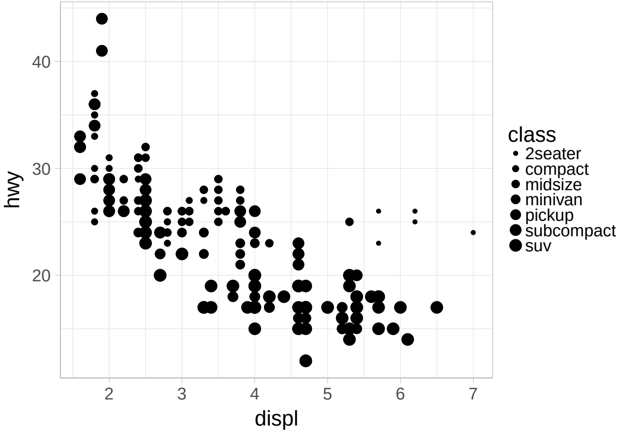


If you prefer British English, like Hadley, you can use `colour` instead of `color`.

The size aesthetic

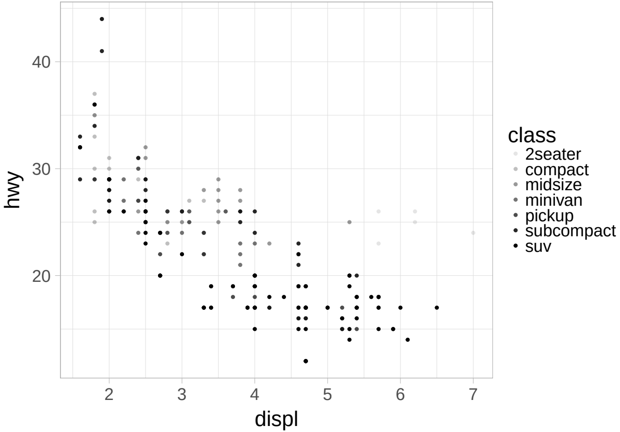
```r
ggplot(data = mpg) +
  geom_point(mapping = aes(x = displ, y = hwy, size = class))
```

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



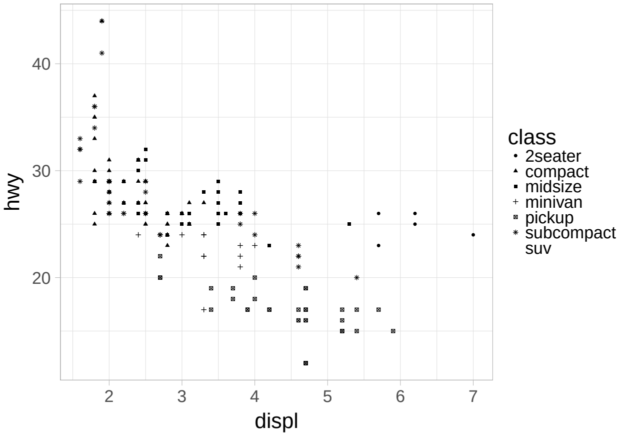
The alpha aesthetic

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



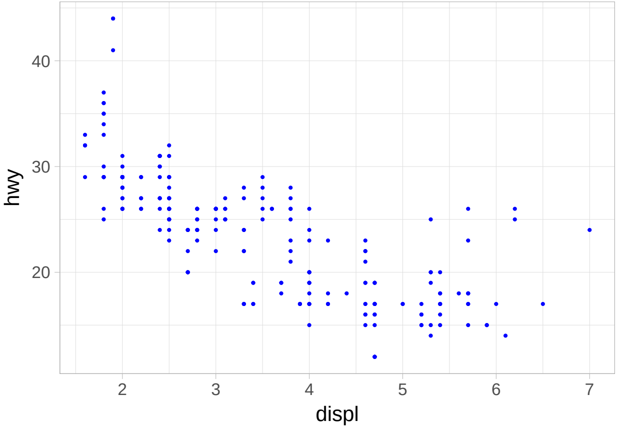
The shape aesthetic

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



Set the aesthetics manually

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



Set the aesthetics manually cont’d

You’ll need to pick a value that makes sense for that aesthetic:

- The name of a color as a character string.
- The size of a point in mm.
- The shape of a point as a number.

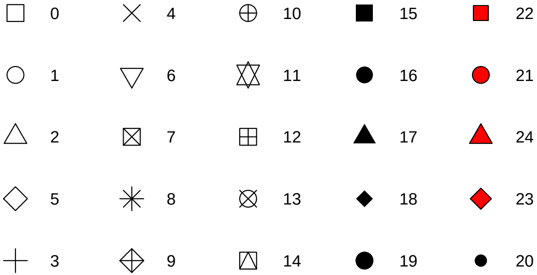


Figure: The hollow shapes (0–14) have a border determined by ‘color‘; the solid shapes (15–18) are filled with ‘color‘; the filled shapes (21–24) have a border of ‘color‘ and are filled with ‘fill‘.

Common problems

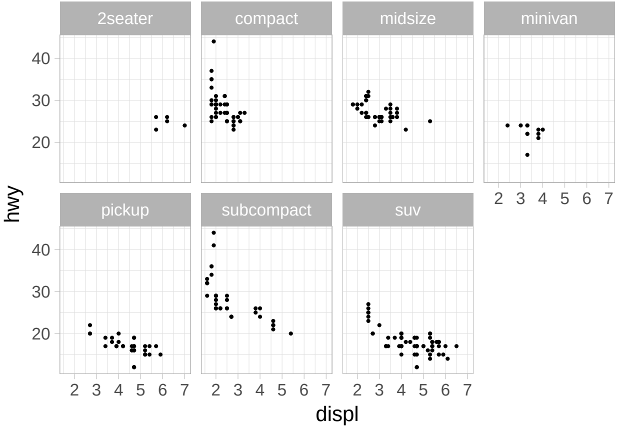
- Check that every (is matched with a) and every " is paired with another ".
- Check that the + is not in the wrong place

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

- You can get help about any R function by running ?function_name in the console, or selecting the function name and pressing F1 in RStudio (use the examples section).
- If that doesn't help, carefully read the error message, the answer will often be buried there!
- Use Google: try googling the error message, as it's likely someone else has had the same problem, and has gotten help online.

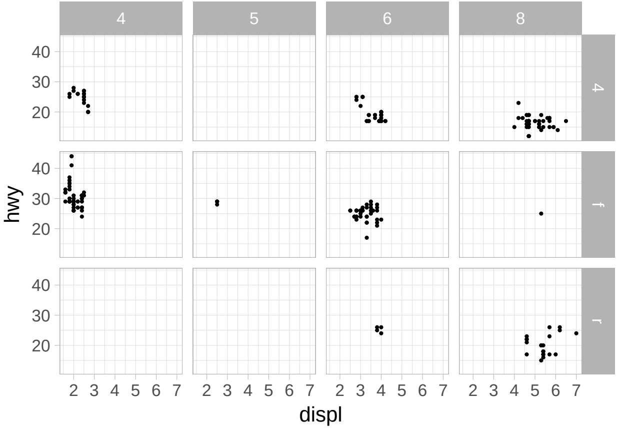
Facets wrap

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



Facets grid

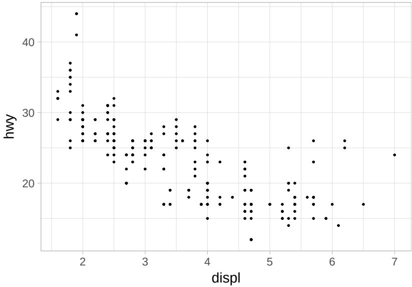
```
ggplot(data = mpg) +
  geom_point(mapping = aes(x = displ, y = hwy)) +
  facet_grid(drv ~ cyl)
```

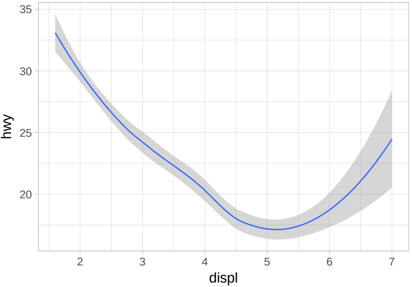


How are these two plots similar?

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

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



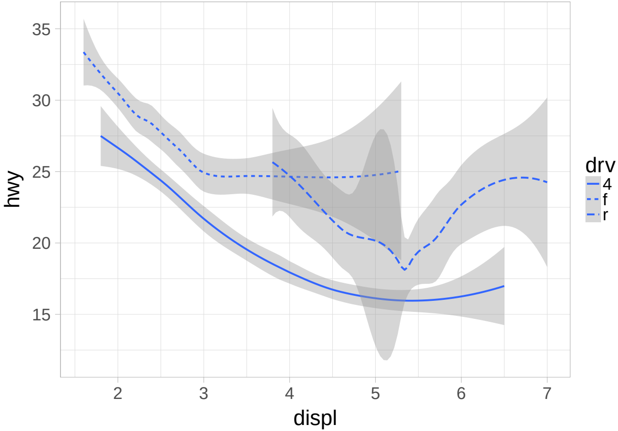


Geometric objects

- A **geom** is the geometrical object that a plot uses to represent data.
- People often describe plots by the type of geom that the plot uses.
- E.g., bar charts use bar geoms, line charts use line geoms, boxplots use boxplot geoms, and so on.
- Scatterplots use the point geom.
- Every **geom** function in ggplot2 takes a `mapping` argument.
- However, **not every aesthetic works with every geom** ().
- E.g., **shape** exists for `geom_point` but not for `geom_line`, and conversely for **linetype**.

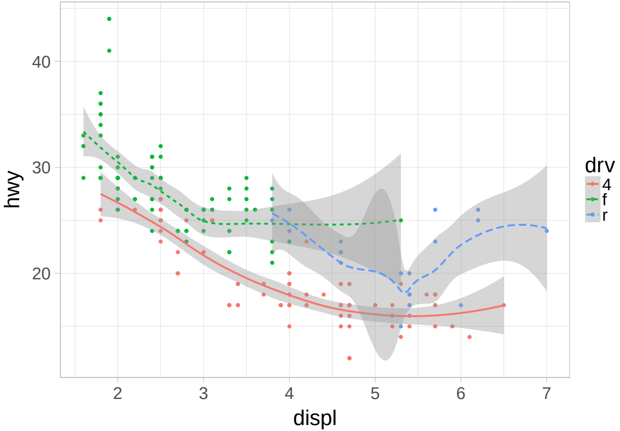
The linetype aesthetic

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



Combining two geoms

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

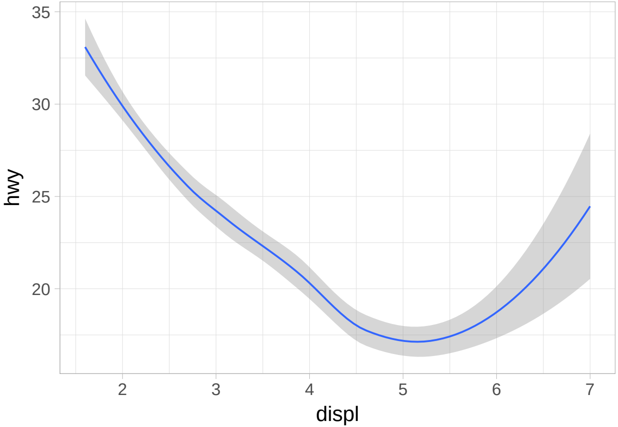


More on geoms

- `ggplot2` provides over 30 geoms.
- extension packages provide even more.
- Use RStudio's data visualization cheatsheet.
- To learn more about any single geom, use help: `?geom_smooth`.

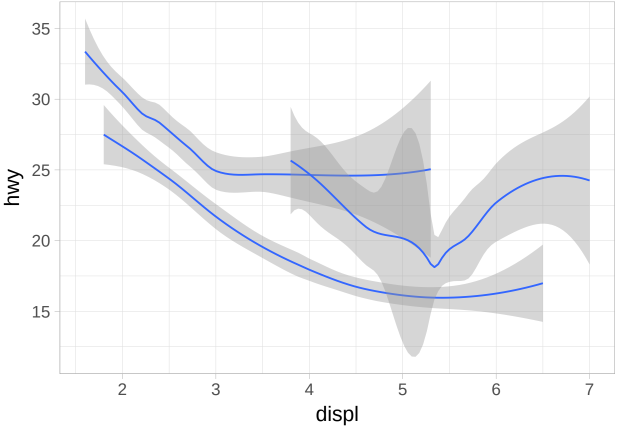
Geoms and legends

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



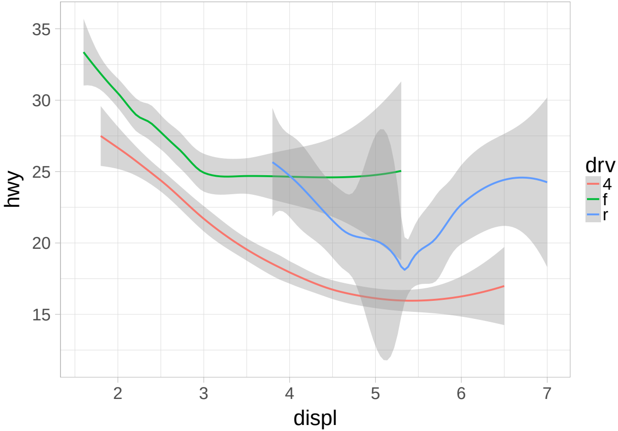
Geoms and legends

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



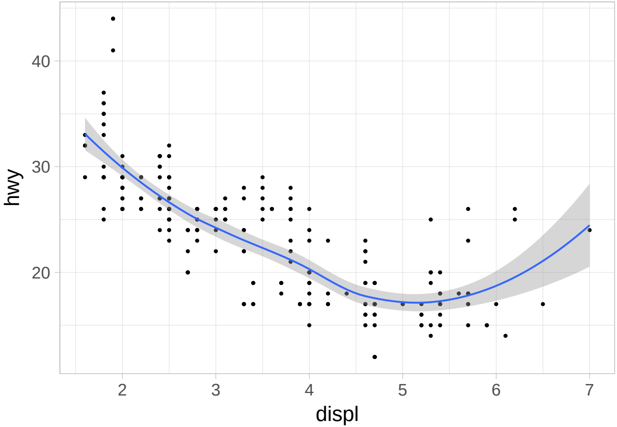
Geoms and legends

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



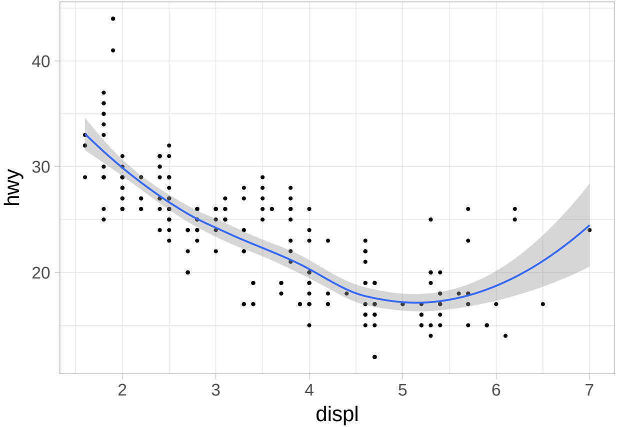
Multiple geoms in the same plot

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



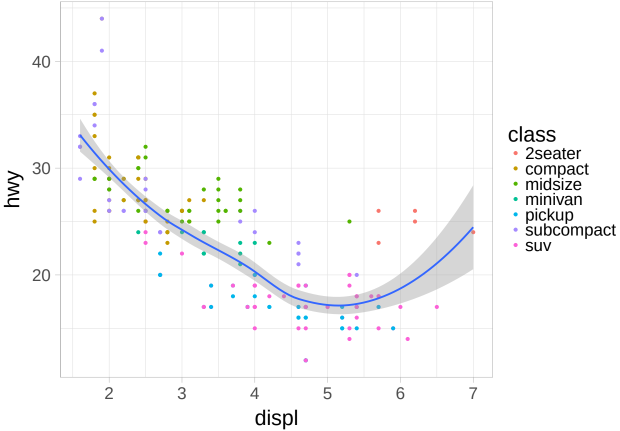
A better way

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



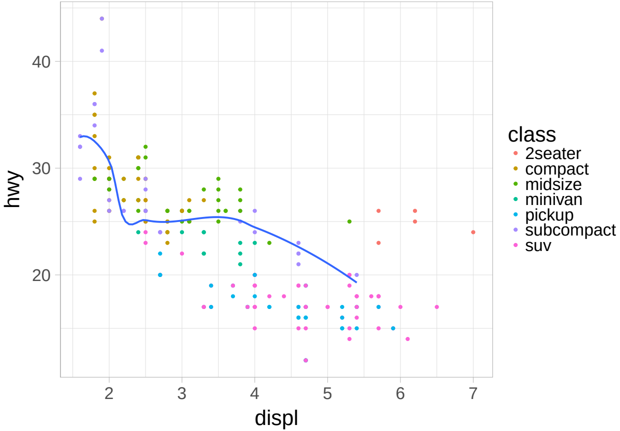
Local vs global mappings

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



Layer dependent data

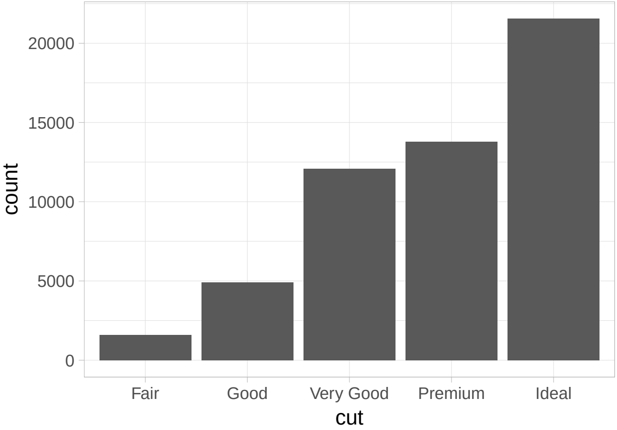
```
ggplot(data = mpg, mapping = aes(x = displ, y = hwy)) +
  geom_point(mapping = aes(color = class)) +
  geom_smooth(data = filter(mpg, class == "subcompact"), se = FALSE)
```



Bar charts

The `diamonds` dataset contains about 54,000 diamonds, including the `price`, `carat`, `color`, `clarity`, and `cut` of each diamond.

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



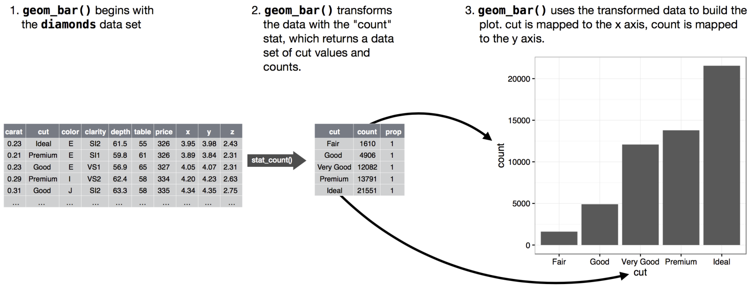
Beyond scatterplots

Other graphs, like bar charts, calculate new values to plot:

- bar charts, histograms, and frequency polygons bin your data and then plot bin counts, the number of points that fall in each bin.
- smoothers fit a model to your data and then plot predictions from the model.
- boxplots compute a robust summary of the distribution and then display a specially formatted box.

Statistical transformations

The algorithm used to calculate new values for a graph is called a **stat**, short for statistical transformation.



1. **`geom_bar()`** begins with the **`diamonds`** data set

carat	cut	color	clarity	depth	table	price	x	y	z
0.23	Ideal	E	SI2	61.5	55	326	3.95	3.98	2.43
0.21	Premium	E	SI1	59.8	61	326	3.89	3.84	2.31
0.23	Good	E	VS1	56.9	65	327	4.05	4.07	2.31
0.29	Premium	I	VS2	62.4	58	334	4.20	4.23	2.63
0.31	Good	J	SI2	63.3	58	335	4.34	4.35	2.75
...	...	...	...	...	...	...	...	...	...

2. **`geom_bar()`** transforms the data with the "count" stat, which returns a data set of cut values and counts.

cut	count	prop
Fair	1610	1
Good	4906	1
Very Good	12082	1
Premium	13791	1
Ideal	21551	1

3. **`geom_bar()`** uses the transformed data to build the plot. cut is mapped to the x axis, count is mapped to the y axis.

Learn which stat a geom uses by inspecting the default value for the `stat` argument.

Bar geom and counts

- ?geom_bar shows that the default value for stat is “count”.
- It means that geom_bar() uses stat_count().
- ?stat_count has a section called “Computed variables” with two new variables: count and prop.

You can generally use geoms and stats interchangeably, e.g.,

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

Geom and stat

- Every geom has a default stat and conversely.
- Typically use geoms without worrying about the underlying statistical transformation.

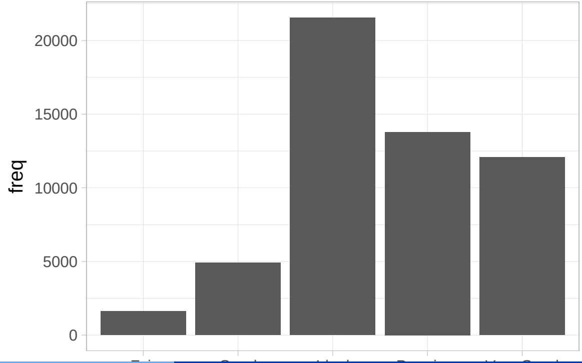
There are three reasons you might need to use a stat explicitly:

1. To override the default stat.
2. To override the default mapping from transformed variables to aesthetics.
3. To draw greater attention to the statistical transformation in your code.

- `ggplot2` provides over 20 stats for you to use.
- Each stat is a function, so you can get help in the usual way, e.g. `?stat_bin`.
- To see a complete list of stats, use RStudio's data visualization cheatsheet.

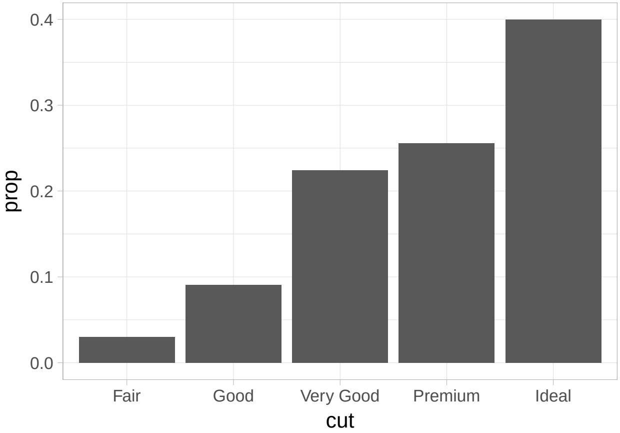
Use a stat explicitely I

```
demo <- tribble(~cut,         ~freq,
                "Fair",       1610,
                "Good",       4906,
                "Very Good",  12082,
                "Premium",    13791,
                "Ideal",      21551)
 ggplot(data = demo) +
   geom_bar(mapping = aes(x = cut, y = freq), stat = "identity")
```



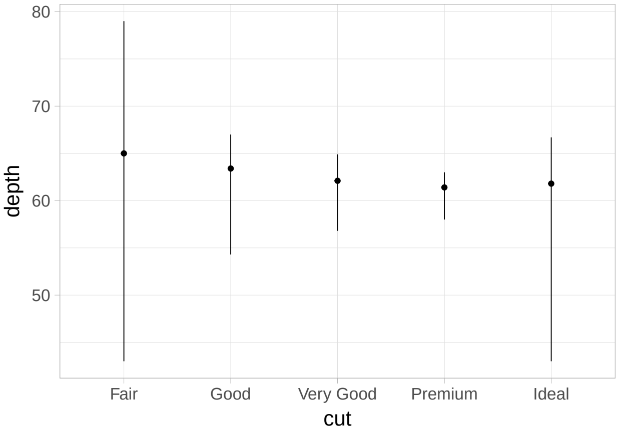
Use a stat explicitely II

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



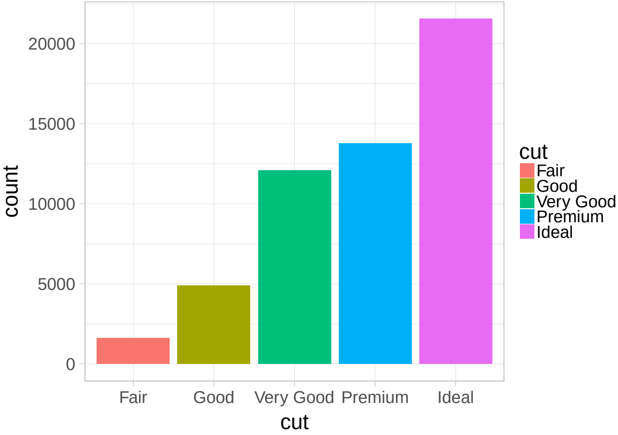
Use a stat explicitely III

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



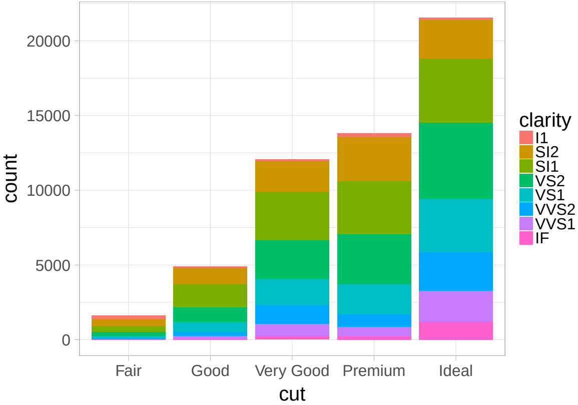
The fill aesthetic

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



Fill and position ajustements

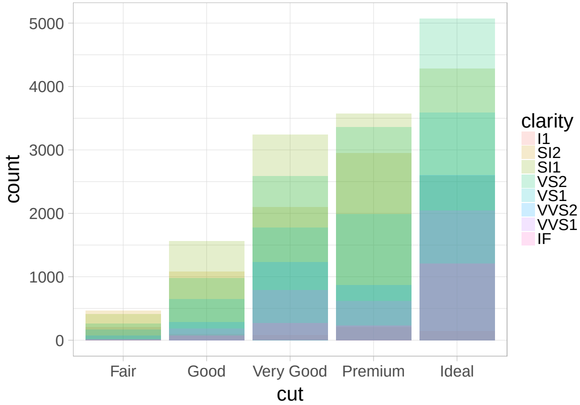
```
ggplot(data = diamonds) +
  geom_bar(mapping = aes(x = cut, fill = clarity))
```



- Automatically stacked by the **position adjustement**.
- `?position_stack` to learn more.

Fill with `position = "identity"`

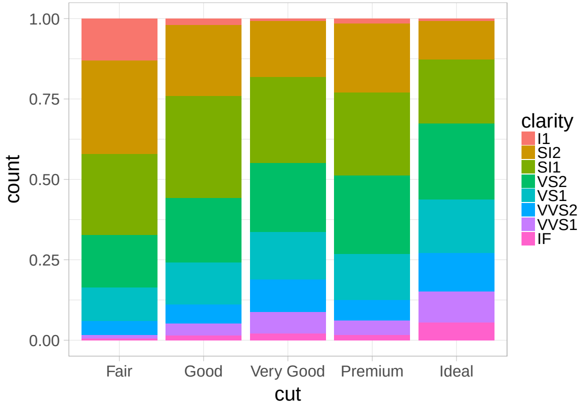
```
ggplot(data = diamonds, mapping = aes(x = cut, fill = clarity)) +
  geom_bar(alpha = 1/5, position = "identity")
```



- Not very useful for bars because of overlap.
- `?position_identity` to learn more.

Fill with `position = "fill"`

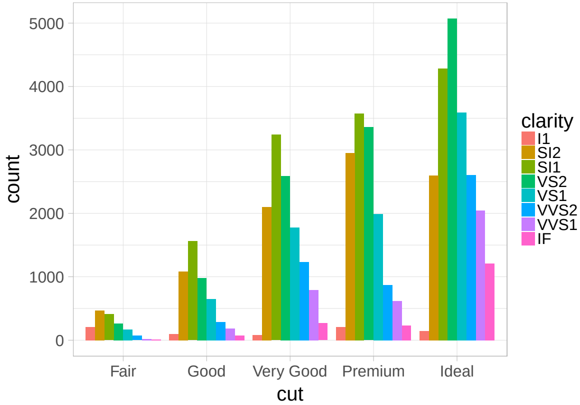
```
ggplot(data = diamonds) +
     geom_bar(mapping = aes(x = cut, fill = clarity), position = "fill")
```



- Makes it easier to compare proportions across groups.
- `?position_fill` to learn more.

Fill with `position = "dodge"`

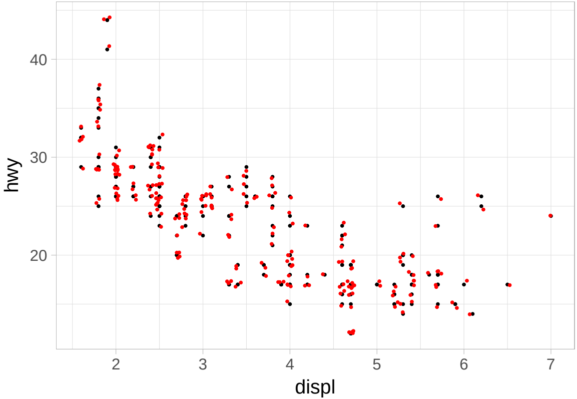
```
ggplot(data = diamonds) +
  geom_bar(mapping = aes(x = cut, fill = clarity), position = "dodge")
```



- Makes it easier to compare individual values.
- `?position_dodge` to learn more.

position = "jitter"

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



- Graph less/**more** accurate/**revealing** at small/**large** scales.
- ?position_jitter to learn more.

Coordinate systems

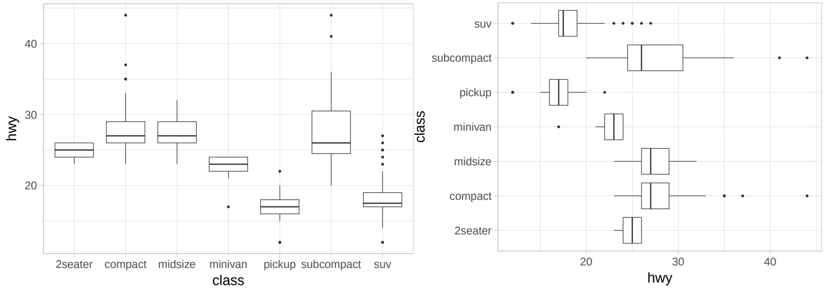
- The most complicated part of `ggplot2`.
- Default: the Cartesian coordinate system.

Other systems occasionally helpful:

- `coord_flip()` switches the x and y axes.
- `coord_quickmap()` sets the aspect ratio correctly for maps.
- `coord_polar()` uses polar coordinates.

coord_flip()

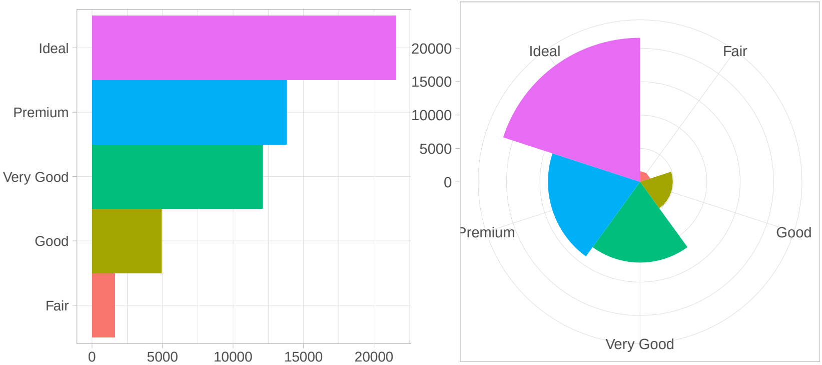
```
ggplot(data = mpg, mapping = aes(x = class, y = hwy)) +
  geom_boxplot()
ggplot(data = mpg, mapping = aes(x = class, y = hwy)) +
  geom_boxplot() +
  coord_flip()
```



- If you want horizontal boxplots.
- If you want long labels.

coord_polar()

```
bar <- ggplot(data = diamonds) +
      geom_bar(mapping = aes(x = cut, fill = cut),
                 show.legend = FALSE, width = 1) +
  theme(aspect.ratio = 1) + labs(x = NULL, y = NULL)
bar + coord_flip()
bar + coord_polar()
```



The layered grammar of graphics

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

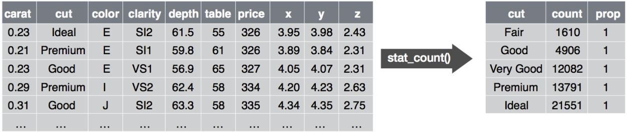
The grammar of graphics

- is a formal system for building plots,
- which uniquely describes *any* plot as a combination of
  - a dataset,
  - a geom,
  - a set of mappings,
  - a stat,
  - a position adjustment,
  - a coordinate system,
  - and a faceting scheme.

Example

1. Begin with the **`diamonds`** data set

2. Compute counts for each cut value with **`stat_count()`**.

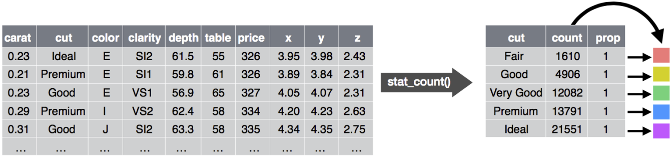


carat	cut	color	clarity	depth	table	price	x	y	z
0.23	Ideal	E	SI2	61.5	55	326	3.95	3.98	2.43
0.21	Premium	E	SI1	59.8	61	326	3.89	3.84	2.31
0.23	Good	E	VS1	56.9	65	327	4.05	4.07	2.31
0.29	Premium	I	VS2	62.4	58	334	4.20	4.23	2.63
0.31	Good	J	SI2	63.3	58	335	4.34	4.35	2.75
...	...	...	...	...	...	...	...	...	...

cut	count	prop
Fair	1610	1
Good	4906	1
Very Good	12082	1
Premium	13791	1
Ideal	21551	1

Example

3. Represent each observation with a bar.
4. Map the **`fill`** of each bar to the **`..count..`** variable.



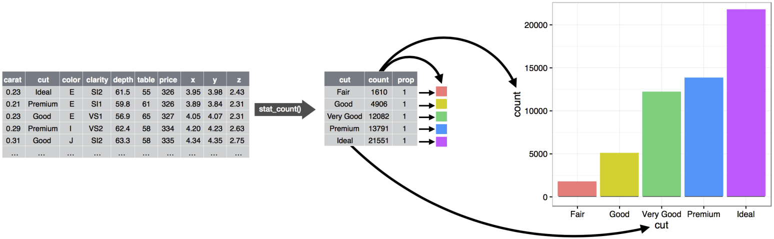
carat	cut	color	clarity	depth	table	price	x	y	z
0.23	Ideal	E	SI2	61.5	55	326	3.95	3.98	2.43
0.21	Premium	E	SI1	59.8	61	326	3.89	3.84	2.31
0.23	Good	E	VS1	56.9	65	327	4.05	4.07	2.31
0.29	Premium	I	VS2	62.4	58	334	4.20	4.23	2.63
0.31	Good	J	SI2	63.3	58	335	4.34	4.35	2.75
...	...	...	...	...	...	...	...	...	...

cut	count	prop
Fair	1610	1
Good	4906	1
Very Good	12082	1
Premium	13791	1
Ideal	21551	1

Example

5. Place geoms in a cartesian coordinate system.

6. Map the y values to `..count..` and the x values to `cut`.



carat	cut	color	clarity	depth	table	price	x	y	z
0.23	Ideal	E	SI2	61.5	55	326	3.95	3.98	2.43
0.21	Premium	E	SI1	59.8	61	326	3.89	3.84	2.31
0.23	Good	E	VS1	56.9	65	327	4.05	4.07	2.31
0.29	Premium	I	VS2	62.4	58	334	4.20	4.23	2.63
0.31	Good	J	SI2	63.3	58	335	4.34	4.35	2.75
...	...	...	...	...	...	...	...	...	...

stat_count()

cut	count	prop	
Fair	1610	1	→ 🟥
Good	4906	1	→ 🟨
Very Good	12082	1	→ 🟩
Premium	13791	1	→ 🟦
Ideal	21551	1	→ 🟪

Summary

- Think about how to best represent data.
- Be honest when using visualization.
- Use the full potential of modern visualization tools.