Data Visualization

Ray Caraher

2023-10-16

Section 1

Intro to Data Visualization

So Far

So far, we have learned how to read data into R, “tidy” it, and generate some useful insights by looking at descriptive statistics.

But oftentimes the most meaningful insights don’t come from looking at numbers, but from looking at figures.

R (and the tidyverse) has an incredible set of tools for making great visualizations.

What is a good graph?

We have also seen plenty of bad visualizations at this point.

So let's review how plots work.

Recall Cairo (2021): Visualizations are made up of two parts:

- The Scaffolding: titles, legends, scales, bylines (who made the graphic?), sources (where did the information come from?), etc.
  - Communicates what the visualization is about, what is being measured, and how it is being measured
- The Content: The information itself visually *encoded*

Scaffolding and Content

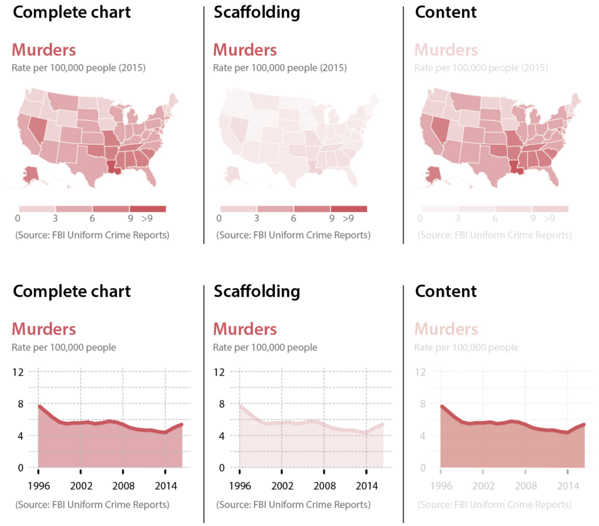


Figure 1: Scaffolding and Content

Visual Encodings

- The means in which we choose to communicate the content
- Examples: Length, size, color, lines, points, direction. etc.

Visual Encoding Example

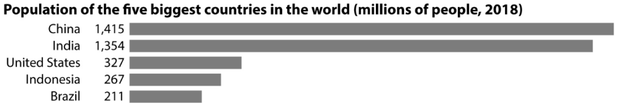


Figure 2: Scaffolding and Content

Here, population of a country is encoded as length.

It allows easy visual comparison between countries by looking at the different lengths. Since India's length is about 4x that of the US, then it tells us population is about 4x the size (if the graph is well designed).

ggplot2

ggplot2 is the tidyverse system of generating visualizations. It provides a robust set of tools encoding information and providing effective scaffolding.

Like the rest of the tidyverse, ggplot2 has a logic and grammar to it.

We can load ggplot2() with the tidyverse. But we also want to load the package with the data we want to use, as well as the package ggthemes, which has colorblind-safe colors. You might need to install these if you don't have them yet using install.packages().

```r
library(tidyverse)
library(palmerpenguins)
library(ggthemes)
```

The Palmer Station Penguins Data

Do penguins with longer flippers weight more or less than penguins with shorter flippers? We can use the palmerpenguins data and visualizations to find out! This is data set is on 344 penguins collected near Palmer Station, Antarctica.

```r
glimpse(penguins)
```

```
## Rows: 344
## Columns: 8
## $ species           <fct> Adelie, Adelie, Adelie, Adelie, Adelie, Adelie, Adel~
## $ island            <fct> Torgersen, Torgersen, Torgersen, Torgersen, Torgerse~
## $ bill_length_mm    <dbl> 39.1, 39.5, 40.3, NA, 36.7, 39.3, 38.9, 39.2, 34.1, ~
## $ bill_depth_mm     <dbl> 18.7, 17.4, 18.0, NA, 19.3, 20.6, 17.8, 19.6, 18.1, ~
## $ flipper_length_mm <int> 181, 186, 195, NA, 193, 190, 181, 195, 193, 190, 186~
## $ body_mass_g       <int> 3750, 3800, 3250, NA, 3450, 3650, 3625, 4675, 3475, ~
## $ sex               <fct> male, female, female, NA, female, male, female, male~
## $ year              <int> 2007, 2007, 2007, 2007, 2007, 2007, 2007, 2007, 2007~
```

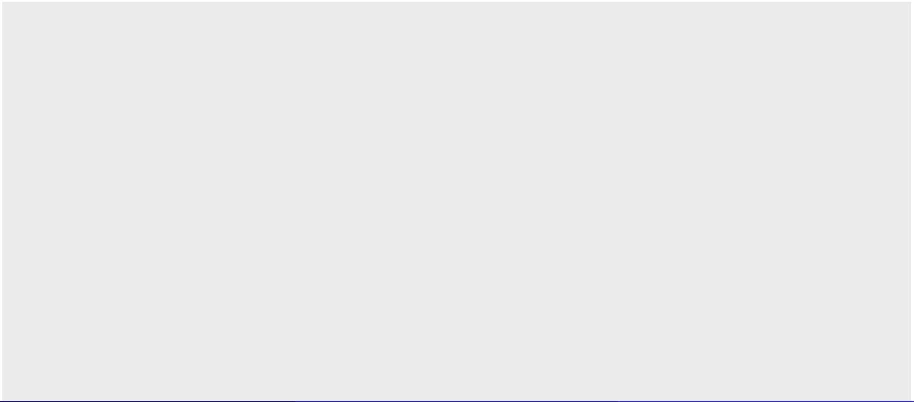
Among the variables are species, island where the penguin was measured, various measurements on the bodies, and the sex.

Intro to ggplot

With ggplot2 system, we first begin by defining the plot object. We can then add "layers" which either provide scaffolding or encode information.

First, let's generate a plot object. It will be empty since we have not defined any layers to this plot

```r
ggplot(data = penguins)
```



Geometries

We now want to tell ggplot how to visually encode our information. We need to provide two things to R:

- Provide the geometric feature that we will use (points, bars, lines, etc.)
- Provide the aesthetic mapping from these geometric features to variables

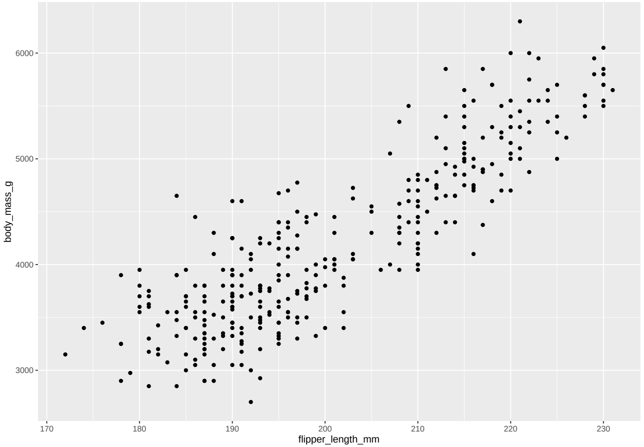
We do this by taking out basic ggplot(data = penguins) function and adding layers. First, we specify the geometric feature with the function geom_...() function (i.e., geom_point(), geom_line(), geom_bar(), and many others). Then within the geom_...() function we provide an aesthetic mapping between the geometry and the variables in our data.

For example, let's make a scatterplot using geom_point() and then map the flipper length on the x-axis and body-mass on the y-axis. Also, notice how R will automatically drop missing values from the figure. This is why it is important that you make sure R has correctly identified these values!

geom_point()

```r
ggplot(data = penguins) +
  geom_point(aes(x = flipper_length_mm, y = body_mass_g))
```

```
## Warning: Removed 2 rows containing missing values (`geom_point()`).
```



geom_point()

This shows the positive relation between flipper length and body mass pretty well. But:

- We should always be skeptical of any apparent relationship between two variables and ask if some other variable is really driving the relationship.
- What if this positive relationship is really just driven by the fact that some of the penguins are different species?
- In other words, does this positive relationship hold **within** species?

We can examine this by adding additional aesthetics and layers!

More aesthetics

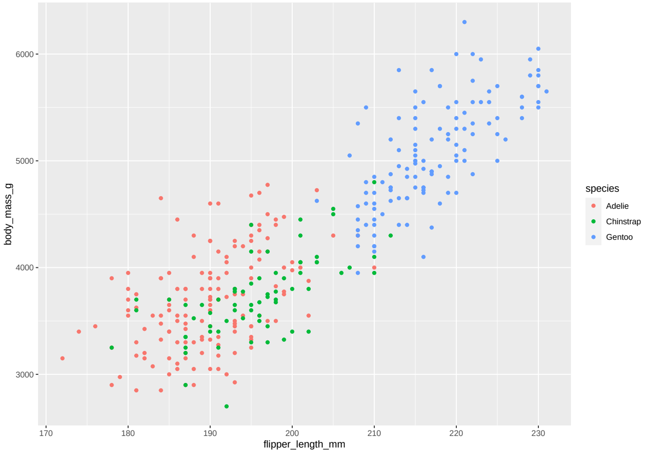
Let's encode information about the species to the visualization to see if this is the case.

- We could do this by either adding an additional geometric feature, or an aesthetic.
- In this case, it makes sense to use different colored points to represent species
- That is a new aesthetic mapping, rather than a new geometric feature

Color aesthetic

```r
ggplot(data = penguins) +
  geom_point(aes(x = flipper_length_mm, y = body_mass_g, color = species))
```

```
## Warning: Removed 2 rows containing missing values (`geom_point()`).
```



Color aesthetic

When we add a character vector to an aesthetic, R will automatically assign a unique color to each unique word (here, species)

geom_smooth()

We see now that even within species, there seems to be a positive relationship between these two variables.

But let's add a line of best fit to make sure and more easily generalize this relationship.

Since a line is a geometric feature, we will add a new layer (rather than add a new aesthetic to an existing one) using geom_smooth().

We can use the same aesthetic mapping as our point geometry, but without color since we only want one smooth line, not one for each species.

Note that we need an additional argument with geom_smooth(): method.

This tells R how we want to draw the line of best fit.

We will tell R to use method = "lm" so the fit is linear.

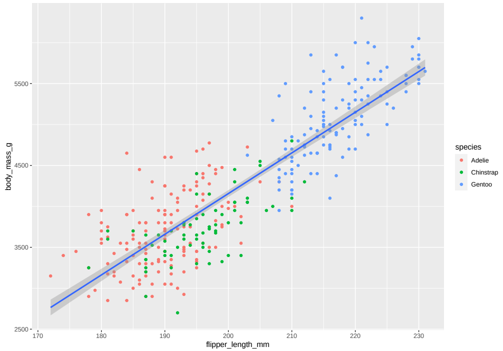
geom_smooth()

```r
ggplot(data = penguins) +
  geom_point(aes(x = flipper_length_mm, y = body_mass_g, color = species)) +
  geom_smooth(aes(x = flipper_length_mm, y = body_mass_g), method = "lm")
```

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

## Warning: Removed 2 rows containing non-finite values (`stat_smooth()`).

## Warning: Removed 2 rows containing missing values (`geom_point()`).
```



Shape aesthetic

In general, we do not want to only represent information in color. Many people are colorblind, and also read in black-and-white when printed.

So we can also map the species to the shape aesthetic. Again since species is a categorical variable, R will automatically chose a shape for each species.

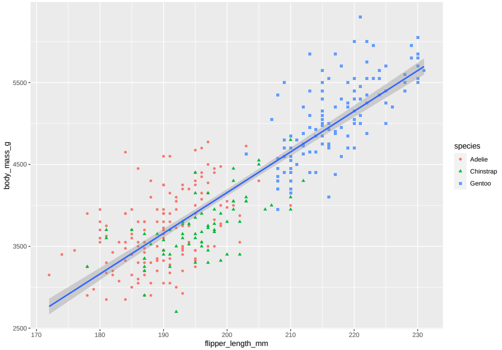
Shape aesthetic

```r
ggplot(data = penguins) +
  geom_point(aes(x = flipper_length_mm, y = body_mass_g, color = species, shape = sp
  geom_smooth(aes(x = flipper_length_mm, y = body_mass_g), method = "lm")
```

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

## Warning: Removed 2 rows containing non-finite values (`stat_smooth()`).

## Warning: Removed 2 rows containing missing values (`geom_point()`).
```



Labelling

Lastly, notice the scaffolding on this plot isn't great.

There few labels, it is not clear what the variables are, we don't have a source, and we don't have a title to tell us what this even is!

We can add this information easily using the labs() function, where we can add titles and label each aesthetic.

We can also change the colors to be more colorblind-accessible with + scale_color_colorblind().

Final version code

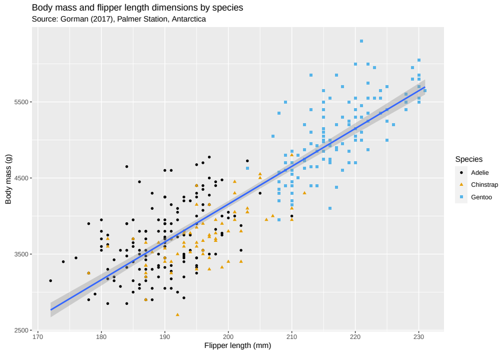
```r
ggplot(data = penguins) +
  geom_point(aes(x = flipper_length_mm, y = body_mass_g, color = species, shape = sp
  geom_smooth(aes(x = flipper_length_mm, y = body_mass_g), method = "lm") +
  labs(
    title = "Body mass and flipper length dimensions by species",
    subtitle = "Source: Gorman (2017), Palmer Station, Antarctica",
    x = "Flipper length (mm)", y = "Body mass (g)",
    color = "Species", shape = "Species"
  ) +
  scale_color_colorblind()
```

Final version figure

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

## Warning: Removed 2 rows containing non-finite values (`stat_smooth()`).

## Warning: Removed 2 rows containing missing values (`geom_point()`).
```



Other geometries

Scatterplots are good for showing the relationship between two variables.

But what about the distribution of one variable?

One option is to use bar graphs (for categorical variables) or histograms (for numeric variables).

geom_bar()

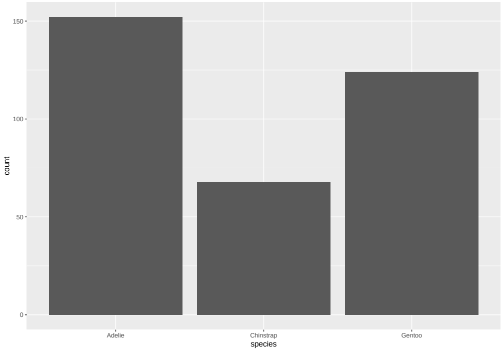
Let's just saw we want to see the number of each species of penguins in our data.

We can so that using geom_bar(), which creates the a bar geometric feature.

The height of the bar shows the number of observations of each species in the data.

geom_bar()

```r
ggplot(penguins) +
  geom_bar(aes(x = species))
```



geom_bar()

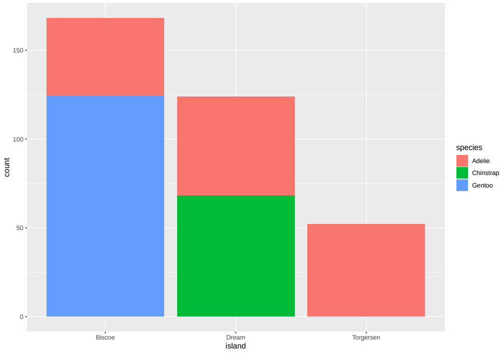
We can also use stacked bar plots to show the relative distribution of two categorical variables.

Let's say we want to see how many species of each type are on each island.

We can use the "fill" aesthetic for geom_bar(), which "fills in" the bars with the color of the species.

geom_bar()

```r
ggplot(penguins) +
  geom_bar(aes(x = island, fill = species))
```



geom_histogram()

If we do not have a categorical variable, but a numeric, we can similarly look at the distribution with bars.

A histogram divides the x-axis into equally spaced bins and then the height of the bar shows the number of observations in each bin.

The “binwidth” argument controls the number of bins.

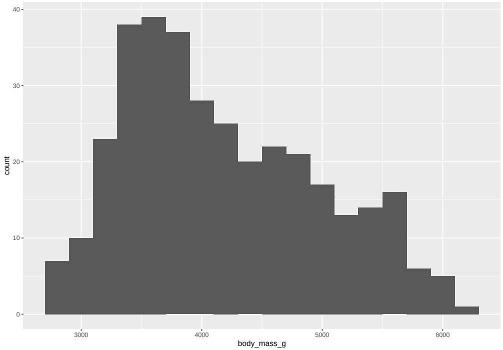
Higher binwidth bins means fewer bars.

Let’s look at body_mass_g using different binwidths.

binwidth = 200

```
ggplot(penguins) +
  geom_histogram(aes(x = body_mass_g), binwidth = 200)
```

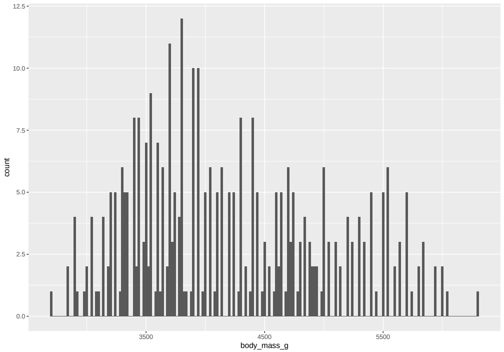
```
## Warning: Removed 2 rows containing non-finite values (`stat_bin()`).
```



binwidth = 20

```
ggplot(penguins) +
  geom_histogram(aes(x = body_mass_g), binwidth = 20)
```

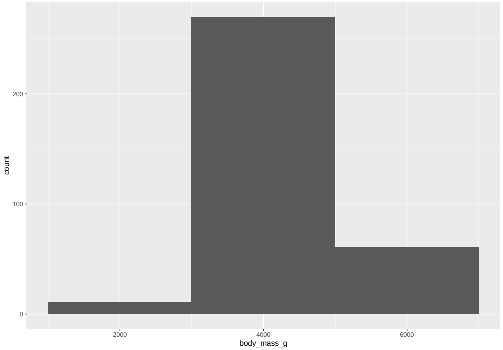
```
## Warning: Removed 2 rows containing non-finite values (`stat
```



binwidth = 2000

```r
ggplot(penguins) +
  geom_histogram(aes(x = body_mass_g), binwidth = 2000)
```

```
## Warning: Removed 2 rows containing non-finite values (`stat
```



Other aestheic mappings

We have talked about color, shape and fill, but there are other aesthetic mappings that are geom-specific.

One example is “linetype”, which works with geometries which draw lines.

Let’s fit a different line-of-best-fit for each penguin species, and draw it as a separate color.

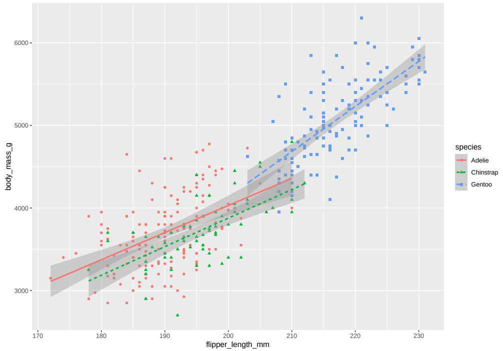
linetype

```
ggplot(data = penguins) +
  geom_point(aes(x = flipper_length_mm, y = body_mass_g, color = species, shape = sp
  geom_smooth(aes(x = flipper_length_mm, y = body_mass_g, linetype = species, color
```

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

## Warning: Removed 2 rows containing non-finite values (`stat_smooth()`).

## Warning: Removed 2 rows containing missing values (`geom_point()`).
```



Facets

One option instead of mapping a variable such as species to color is to instead create a separate subplot for each species.

This is another way to visually compare differences in relationships across penguin species (or any categorical variable).

We can do this easily using the facet_wrap() function.

We use the "$\sim$" symbol to tell facet_wrap() which column to use.

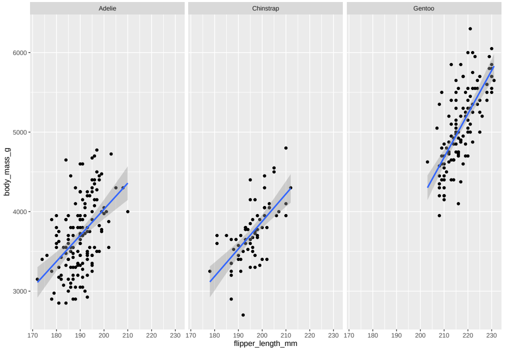
facet_wrap()

```r
ggplot(data = penguins) +
  geom_point(aes(x = flipper_length_mm, y = body_mass_g)) +
  geom_smooth(aes(x = flipper_length_mm, y = body_mass_g), method = "lm") +
  facet_wrap(~ species)
```

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

## Warning: Removed 2 rows containing non-finite values (`stat_smooth()`).

## Warning: Removed 2 rows containing missing values (`geom_point()`).
```



Set aesthetics

Sometimes you may want to use aesthetics in your figures without mapping them to a variable.

For example, rather than having a different color point on a scatterplot for different penguin species, you may just want all points to be blue.

We can do this by assigning the color outside of the aes() function.

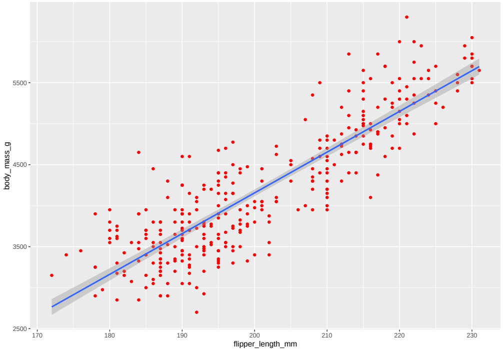
Setting constant aesthetics

```r
ggplot(data = penguins) +
  geom_point(aes(x = flipper_length_mm, y = body_mass_g), color = "red") +
  geom_smooth(aes(x = flipper_length_mm, y = body_mass_g), method = "lm")
```

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

## Warning: Removed 2 rows containing non-finite values (`stat_smooth()`).

## Warning: Removed 2 rows containing missing values (`geom_point()`).
```



Section 2

Exploratory Data Analysis using Visualizations

Exploratory Data Analysis

With tools of data visualization and data transformation, we can do some rudimentary but real data analysis.

Exploratory Data Analysis:

An iterative type of data analysis that uses simple tools to understand the relationships within and between variables.

Generally revolves around exploring the answers to two sets of questions:

- What type of *variation* occurs within my variables?
- What type of *covariation* occurs between my variables?

diamonds dataset

We will explore these questions using the diamonds data set.

Let's first look at the variable names and types.

It contains information on the price, weight(carat), cut, color, and size of 54,000 diamonds.

diamonds dataset

```r
glimpse(diamonds)
```

```
## Rows: 53,940
## Columns: 10
## $ carat   <dbl> 0.23, 0.21, 0.23, 0.29, 0.31, 0.24, 0.24, 0.26, 0.22, 0.23, 0.~
## $ cut     <ord> Ideal, Premium, Good, Premium, Good, Very Good, Very Good, Ver~
## $ color   <ord> E, E, E, I, J, J, I, H, E, H, J, J, F, J, E, E, I, J, J, J, I,~
## $ clarity <ord> SI2, SI1, VS1, VS2, SI2, VVS2, VVS1, SI1, VS2, VS1, SI1, VS1, ~
## $ depth   <dbl> 61.5, 59.8, 56.9, 62.4, 63.3, 62.8, 62.3, 61.9, 65.1, 59.4, 64~
## $ table   <dbl> 55, 61, 65, 58, 58, 57, 57, 55, 61, 61, 55, 56, 61, 54, 62, 58~
## $ price   <int> 326, 326, 327, 334, 335, 336, 336, 337, 337, 338, 339, 340, 34~
## $ x       <dbl> 3.95, 3.89, 4.05, 4.20, 4.34, 3.94, 3.95, 4.07, 3.87, 4.00, 4.~
## $ y       <dbl> 3.98, 3.84, 4.07, 4.23, 4.35, 3.96, 3.98, 4.11, 3.78, 4.05, 4.~
## $ z       <dbl> 2.43, 2.31, 2.31, 2.63, 2.75, 2.48, 2.47, 2.53, 2.49, 2.39, 2.~
```

Variation

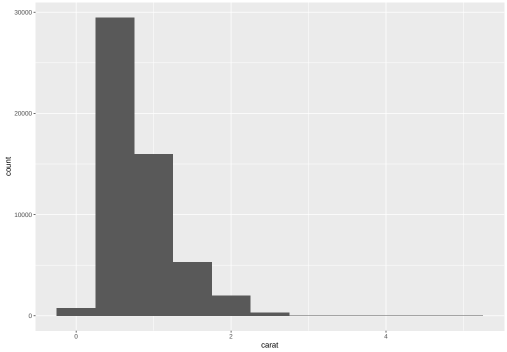
Variation is the tendency of the values of a variable to change from measurement to measurement or observation to observation.

- For example, the commute time to campus is one variable, and it can *vary* between people (live at different addresses, etc.) and within a person (different traffic on different days, etc.).

In the diamonds data, we can look at how the carat(weight) varies within the dataset to reveal important insights.

Let's use a histogram for this.

carat variation

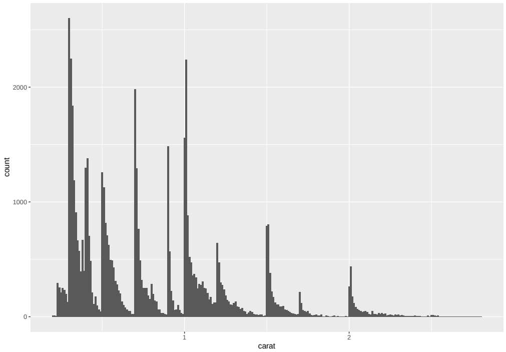


carat variation

We can see that most diamonds weight less then 2 carats.

Let's zoom in on this data by only looking at carat values less than 3, and a smaller binwidth to see some more variation.

carat variation



carat variation

Clearly there are several clusters of carat observations which may provide some insight into the diamond industry.

We see how some interesting variation that can lead us to more questions:

- Why are carats more likely to be whole numbers?
- Why are there more clusters between 0 and 1 than between 1 and 2?
- Why does each cluster appear to have the same tapering shape?

Answering these questions would require you go get some industry-specific insights, look at more graphs, or run some models.

But we see how looking at data visualizations and iteratively creating them can both teach you about the data, provide insights, and lead you to more questions which can then lead to further insights.

Analyzing variation can also help us identify those observations which have abnormally large or small values, which we can then omit or alter if need be.

Covariation

If variation describes the behavior within a variable, covariation describes the behavior between variables.

Covariation is the tendency for the values of two or more variables to vary together in a related way.

We can examine covariation by visualizing two or more variables together.

How we do this depends on the type of variable we are examining (i.e., a numerical and a categorical, two numerical, and two categorical variables).

Numerical vs. categorical

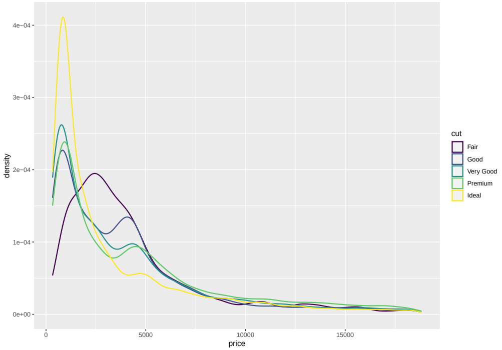
Let's focus on the relationship between price and cut (the quality) - so a numerical vs. a categorical variable.

Here we can look at a density plot, which is like a histogram in that it plots where observations are clustered.

Density plot

```r
ggplot(diamonds, aes(x = price)) +
  geom_density(aes(color = cut), binwidth = 500, linewidth = 0.75)
```

```
## Warning in geom_density(aes(color = cut), binwidth = 500, linewidth = 0.75):
## Ignoring unknown parameters: `binwidth`
```



Density plot

We see something interesting - that the average price for higher quality diamonds (ideal) is actually lower than fair cuts.

This non-intuitive answer would lead you to more questions - namely why - and then encourage a further round of data analyses.

Categorical v. Categorical

One way to examine the covariation between two categorical variables is to look at the counts of observations which fall into each group.

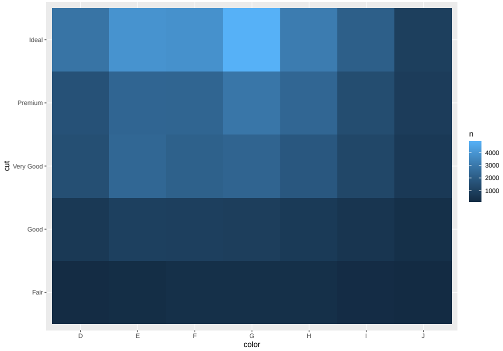
More specifically, we need to count the number of observations for each combination of levels of these categorical variables.

If we want to look at the relationship between the color (best to worst) and the cut, we can use count() then use a data visualization.

Here we can use geom_tile() which will show the number of obersevations in each combination by color in a grid pattern.

Categorical v. Categorical

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



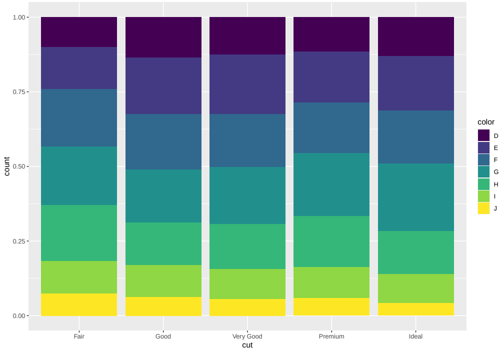
Categorical v. Categorical

We can also do a stacked bar plot to look at the proportions of each color-type within each cut-type.

To have geom_bar() compute proportions rather than counts, we need to add the position = “fill” argument after the aes() function.

Categorical v. Categorical

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



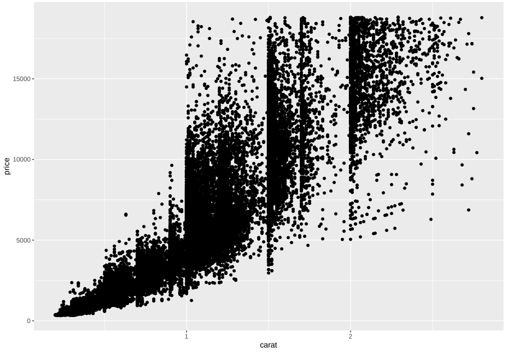
Numerical v. numerical

We have already used scatterplots which is an excellent way to visualize the relationship between two numeric varaibles.

Let's look at the relationship between price and carat again.

Numerical v. numerical

```r
ggplot(smaller, aes(x = carat, y = price)) +
  geom_point()
```



Numerical v. numerical

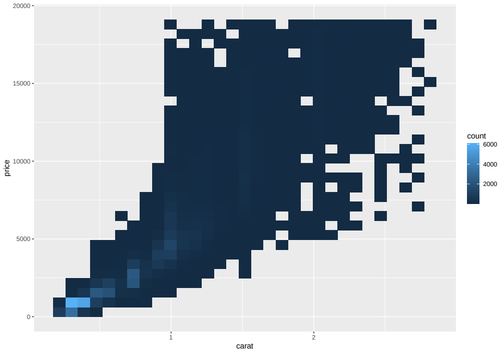
However, we can see some issues here.

Since there are so many observations, the scatterplot becomes a dark cloud making it hard to see anything but general patterns.

To make this easier to read, we can create “bins” where we instead plot small squares which contain a whole bunch of points, then use color to show the number of points in each bin.

Numerical v. numerical

```r
ggplot(smaller, aes(x = carat, y = price)) +
  geom_bin2d()
```



Patterns and Data Visualization

Data visualization provides us robust tools for identifying patterns in the data.

But now that we see some patterns, we need to ask another set of questions:

- Is this just a coincidence?
- How strong is the relationship implied by the pattern?
- What other variables might affect the relationship?
- Does the relationship change if you look at individual subgroups of the data?

Often, we will need to use modelling tools to answer some of these questions.

But you can see how far basic visualization tools got us.