

Data Journalism with R and the Tidyverse

Code, data and visuals for storytellers

Matt Waite & Sarah Cohen (original authors); updated by Sean Mussenden, Rob Weis

8/29/2022

Table of contents

1	Introduction	6
1.1	Installations	6
1.2	About this book	7
1.3	What we'll cover	7
2	Learn a new way to read	9
2.1	Read like a reporter	10
What were the questions?	10
Go beyond the numbers	10
2.2	Reading tips	11
2.3	Analyze data for story, not study	12
3	Newsroom math	15
3.1	Why numbers?	15
3.2	Overcoming your fear of math	16
3.3	Put math in its place	16
3.4	Going further	17
	Tipsheets	17
4	Census Data	18
5	Spreadsheets	19
	Introduction	20
	Tutorials	20
6	Spreadsheet Refresher	21
6.1	Re-learning Excel from the ground up	22
The spreadsheet grid	22
Mouse shapes	22
Selecting cells and ranges	23
Reading the screen	24
Entering data	25
Locking in headings	25
Formatting tricks	27

6.2	Getting started with a dataset	27
	First steps	27
	Interview your data	27
6.3	Video walkthrough	28
6.4	Keyboard shortcuts	28
7	Sorting and filtering to find stories	30
7.1	A sorting miracle	30
7.2	Sorting and filtering as a reporting tool	30
7.3	Example data	32
7.4	Get the data into Google Sheets	32
7.5	Understanding data types	33
	7.5.1 Sorting rows	34
	7.5.2 Filtering	35
8	Spreadsheet Formulas	37
8.1	Formulas in spreadsheets	37
8.2	Common spreadsheet arithmetic	38
	8.2.1 Check the government's math with SUM	38
	8.2.2 Change in spending	40
	8.2.3 Percent change	40
	8.2.4 Parts of a whole: percent of total	41
8.3	While we're at it: two kinds of averages	41
8.4	FAQs	42
9	Grouping with pivot tables	43
	Confusing grouping with sorting or arranging	43
	When to use filter vs. pivot tables	44
9.1	Tutorial	44
	Setting up the pivot table	45
	Counting , or "how many"?	45
	Percents of total	47
	More variables	48
	Even more variables	49
9.2	FAQ	51
	I have too many columns	51
	I want to sort by percents, not numbers	51
	Things aren't adding up	51
	Its a crazy number!	51
	This is so frustrating - I can't get what I want	51
10	Cleaning data with Google Sheets	52
10.0.1	Text to columns	52

10.0.2 Normalizing	52
10.0.3 Lowercase or Uppercase character conversion	53
10.0.4 White space	53
11 Getting started with R and RStudio	55
R or Python?	56
11.1 Install R and RStudio	56
11.2 Unlocking packages and the tidyverse	57
11.3 Set up RStudio for data reporting	57
11.4 The screen	58
The Console	58
Files tab	61
Environment	61
Typing into the console	61
11.5 Working directory	62
11.6 Interactive R tutorial	62
11.7 Relax!	63
11.8 Other resources	63
12 Loading and Analyzing Data	64
12.0.1 Install software to grab data	64
12.0.2 Data	65
12.0.3 Import Data	65
12.0.4 Explore Data	66
12.0.5 Navigation Tips	69
12.0.6 Dplyr	69
12.0.7 Find Your News Organization	71
12.0.8 Visualize	74
12.0.9 What You Have Learned So Far	75
12.0.10 Questions	75
12.0.11 Tutorials	76
13 Using GitHub	77
13.1 How It Works	77
13.2 Getting Started	77
13.3 Video overview	78
13.4 Advanced Use	78
14 Mutuating, Aggregates, Filters	79
14.0.1 R Libraries Background	79
14.1 Importing data	80
14.1.1 Group by and count	82

14.2 Interviewing Your Data: Min, Max, Mean, Medians	84
14.2.1 Filters: Extracting Needles from Haystacks	85
14.2.2 Other summarization methods: mean, median, min and max	87
14.2.3 Using sum	88
15 Visualizing your data for reporting	90
15.0.1 Why to use visualization	90
15.0.2 311 Call Data	91
15.0.3 Bar charts	92
15.1 Line charts	100
(APPENDIX) Appendix	105
Newsroom numbers cheat sheet	106
The PERS: Fractions, rates, percents and per capita	106
Fractions and percents	107
Rates and per capita	108
Measuring change	111
Simple differences	111
Percent change / Percent difference	112
Going further with percents and rates	114
Average and typical values	115
The average or mean	115
The median	116

1 Introduction

Welcome to data journalism. The main goal of this course is to expand your ability to report and tell stories using data. You will use these tools to discover trends in data, like what Rachell Sanchez-Smith found with the [sharp jump in COVID-19 cases in Arkansas children](#). You will learn how to create and publish graphics and maps. It's hard work but it is a lot of fun and very rewarding.

We have some basic goals for you to reach in this class. By the end of the semester, we want you to have basic proficiency and independence with data analysis. We want you to be able to write about data clearly, using the Associated Press style as a benchmark. We want you to be able to find and download a dataset, clean it up, visualize it.

The skills you will learn in the coming weeks are in high demand in journalism and beyond. Examine this BuzzFeed job description from 2017:

“We’re looking for someone with a passion for news and a commitment to using data to find amazing, important stories — both quick hits and deeper analyses that drive conversations,” the posting seeking a data journalist says. It goes on to list five things BuzzFeed is looking for: Excellent collaborator, clear writer, deep statistical understanding, knowledge of obtaining and restructuring data.

“You should have a strong command of at least one toolset that (a) allows for filtering, joining, pivoting, and aggregating tabular data, and (b) enables reproducible workflows.”

You will learn these skills in this book. You’ll get a taste of modern data journalism through Google Sheets and programming in R, a statistics language. You’ll be challenged to think programmatically while thinking about a story you can tell to readers in a way that they’ll want to read. Combining them together has the power to change policy, expose injustice and deeply inform.

1.1 Installations

This book begins with a basic review of Google Sheets and then shifts to the R statistical language. To follow along, you’ll do the following:

1. Install the R language on your computer. Go to the [this website](#), click download R based on your operating system. If that link somehow doesn't work, check [R Project website](#) and find a different location.
2. Install [R Studio Desktop](#). The free version is great.

Going forward, you'll see passages like this:

```
install.packages("tidyverse")
```

That is code that you'll need to run common software packages in your R Studio.

1.2 About this book

This book is the collection of class materials compiled by various data journalism professors around the country: Matt Waite at the University of Nebraska-Lincoln's College of Journalism and Mass Communications and Sarah Cohen of Arizona State University. This version was edited by Derek Willis, Sean Mussenden and Rob Wells at the University of Maryland Philip Merrill College of Journalism.

There's some things you should know about it:

- It is free for students.
- The topics will remain the same but the text is going to be constantly tinkered with.
- What is the work of the authors is copyright Matt Waite 2020, Sarah Cohen 2022 and Derek Willis, Sean Mussenden and Rob Wells 2022.
- The text is [Attribution-NonCommercial-ShareAlike 4.0 International](#) Creative Commons licensed. That means you can share it and change it, but only if you share your changes with the same license and it cannot be used for commercial purposes. I'm not making money on this so you can't either.
- As such, the whole book – authored in Quarto – in its original form is [open sourced on Github](#). Pull requests welcomed!

1.3 What we'll cover

- Spreadsheets
- R Basics
- Replication, Data Diary
- Data basics and structures
- Aggregates
- Mutating

- Working with dates
- Filters
- Data cleaning techniques, Janitor
- Pulling Data from PDFs
- Joins
- Basic data scraping
- Getting data from APIs: Census
- Visualizing for reporting: Basics
- Visualizing for reporting: Publishing
- Geographic data basics
- Geographic queries
- Geographic visualization
- Text analysis basics
- Writing with and about data
- Data journalism ethics

2 Learn a new way to read

Getting started in data journalism often feels as if you've left the newsroom and entered the land of statistics, computer programming and data science. This chapter will help you start seeing data reporting in a new way, by learning how to study great works of the craft as a writer rather than a reader.

← Thread



jelani cobb 
@jelani9

..

Here's a bit of writing advice I often share with students: engineers don't look at a bridge the same way pedestrians or drivers do. The former understand the bridge as a language of angles and load bearing structures. Writers should read books in that same way.

4:44 PM · Dec 20, 2021 · Twitter for iPhone

Figure 2.1: jelani cobb

Jelani Cobb tweeted, “an engineer doesn’t look at a bridge the same way pedestrians or drivers do.” They see it as a “language of angles and load bearing structures.” We just see a bridge. While he was referring to long-form writing, reporting with data can also be learned by example – if you spend enough time with the examples.

Almost all good writers and reporters try to learn from exemplary work. I know more than one reporter who studies prize-winning journalism to hone their craft. This site will have plenty of examples, but you should stay on the lookout for others.

2.1 Read like a reporter

Try to approach data or empirical reporting as a reporter first, and a consumer second. The goal is to triangulate how the story was discovered, reported and constructed. You'll want to think about why *this* story, told this way, at this time, was considered newsworthy enough to publish when another approach on the same topic might not have been.

What were the questions?

In data journalism, we often start with a tip, or a hypothesis. Sometimes it's a simple question. Walt Bogdanich of The New York Times is renowned for seeing stories around every corner. Bogdanich has said that the prize-winning story "[A Disability Epidemic Among a Railroad's Retirees](#)" came from a simple question he had when railway workers went on strike over pension benefits – how much were they worth? The story led to an FBI investigation and arrests, along with pension reform at the largest commuter rail in the country.

The hypothesis for some stories might be more directed. In 2021, the Howard Center for Investigative Journalism at ASU published "[Little victims everywhere](#)", a set of stories on the lack of justice for survivors of child sexual assault on Native American reservations. That story came after previous reporters for the center analyzed data from the Justice Department showing that the FBI dropped most of the cases it investigated, and the Justice Department then only prosecuted about half of the matters referred to it by investigators. The hypothesis was that they were rarely pursued because federal prosecutors – usually focused on immigration, white collar crime and drugs – weren't as prepared to pursue violent crime in Indian Country.

When studying a data-driven investigation, try to imagine what the reporters were trying to prove or disprove, and what they used to do it. In journalism, we rely on a mixture of quantitative and qualitative methods. It's not enough to prove the "numbers" or have the statistical evidence. That is just the beginning of the story. We are supposed to ground-truth them with the stories of actual people and places.

Go beyond the numbers

It's easy to focus on the numbers or statistics that make up the key findings, or the reason for the story. Some reporters make the mistake of thinking all of the numbers came from the same place – a rarity in most long-form investigations. Instead, the sources have been woven together and are a mix of original research and research done by others. Try to pay attention to any sourcing done in the piece. Sometimes, it will tell you that the analysis was original. Other times it's more subtle.

But don't just look at the statistics being reported in the story. In many (most?) investigations, some of the key people, places or time elements come directly from a database.

Sarah Cohen at Arizona State University analyzed the Paycheck Protection Program loan data for ProPublica and found a handful of sketchy-looking records from a single county in coastal New Jersey. It turned out to be a [pretty good story](#).

Often, the place that a reporter visits is determined by examples found in data. In [this story on rural development](#) funds, all of the examples came from an analysis of the database. Once the data gave us a good lead, the reporters examined press releases and other easy-to-get sources before calling and visiting the recipients or towns.

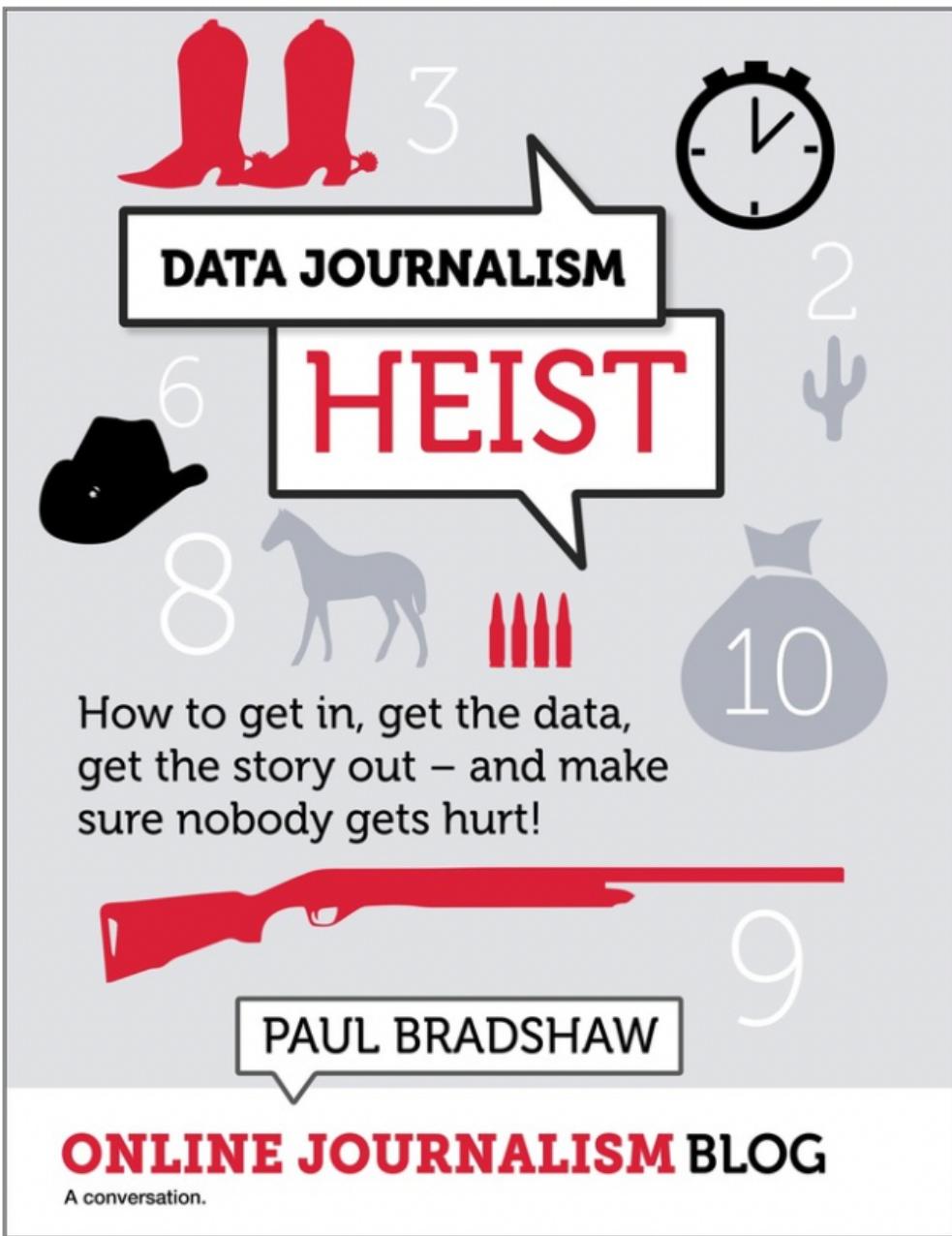
2.2 Reading tips

You'll get better at reading investigations and data-driven work over time, but for now, remember to go beyond the obvious:

- Where might the reporters have found their key examples, and what made them good characters or illustrations of the larger issue? Could they have come from the data?
- What do you think came first – a narrative single example that was broadened by data (naively, qualitative method), or a big idea that was illustrated with characters (quantitative method)?
- What records were used? Were they public records, leaks, or proprietary data?
- What methods did they use? Did they do their own testing, use statistical analysis, or geographic methods? You won't always know, but look for a methodology section or a description alongside each story.
- How might you localize or adapt these methods to find your own stories?
- Pick out the key findings (usually in the nut graf or in a series of bullets after the opening chapter): are they controversial? How might they have been derived? What might have been the investigative hypothesis? Have they given critics their due and tried to falsify their own work?
- How effective is the writing and presentation of the story? What makes it compelling journalism rather than a dry study? How might you have done it differently? Is a video story better told in text, or would a text story have made a good documentary? Are the visual elements well integrated? Does the writing draw you in and keep you reading? Think about structure, story length, entry points and graphics all working together.
- Are you convinced? Are there holes or questions that didn't get addressed?

2.3 Analyze data for story, not study

As journalists we'll often be using data, social science methods and even interviewing differently than true experts. We're seeking stories, not studies. Recognizing news in data is one of the hardest skills for less experienced reporters new to data journalism. This list of potential newsworthy data points is adapted from Paul Bradshaw's "[Data Journalism Heist](#)".



LAST UPDATED ON 2015-06-10

- Compare the claims of powerful people and institutions against facts – the classic investigative approach.
- Report on *unexpected* highs and lows (of change, or of some other characteristic)

- Look for outliers – individual values that buck a trend seen in the rest
- Verify or bust some myths
- Find signs of distress, happiness or dishonesty or any other emotion.
- Uncover *new* or *under-reported* long-term trends.
- Find data suggesting your area is *the same* or *different* than most others of its kind.

Bradshaw also did a recent study of data journalism pieces: “[Here are the angles journalists use most often to tell the stories in data](#)”, in Online Journalism Blog. I’m not sure I agree, only because he’s looking mainly at visualizations rather than stories, but they’re worth considering.

3 Newsroom math

Jo Craven McGinty, then of The New York Times, used simple rates and ratios to discover that a 6-story brick New Jersey hospital was the most expensive in the nation. In 2012, Bayonne Medical Center “charged the highest amounts in the country for nearly one-quarter of the most common hospital treatments,” the [Times story said](#).

To do this story, McGinty only needed to know the number of the procedures reported to the government and the total amount each hospital charged. Dividing those to find an average price, then ranking the most common procedures, led to this surprising result.

3.1 Why numbers?

Using averages, percentages and percent change is the bread and butter of data journalism, leading to stories ranging from home price comparisons to school reports and crime trends. It may have been charming at one time for reporters to announce that they didn’t “do” math, but no longer. Instead, it is now an announcement that the reporter can only do some of the job. You will never be able to tackle complicated, in-depth stories without reviewing basic math.

The good news is that most of the math and statistics you need in a newsroom isn’t nearly as difficult as high school algebra. You learned it somewhere around the 4th grade. You then had a decade to forget it before deciding you didn’t like math. But mastering this most basic arithmetic again is a requirement in the modern age.

In working with typical newsroom math, you will need to learn how to:

- Overcome your fear of numbers
- Integrate numbers into your reporting
- Routinely compute averages, differences and rates
- Simplify and select the right numbers for your story

While this chapter covers general tips, you can find specific instructions for typical newsroom math in this [Appendix A](#), an excerpt from Sarah Cohen’s outstanding book, [Numbers in the Newsroom](#). It’s worth getting your own copy if you don’t already have one.

3.2 Overcoming your fear of math

When we learned to read, we got used to the idea that 26 letters in American English could be assembled into units that we understand without thinking – words, sentences, paragraphs and books. We never got the same comfort level with 10 digits, and neither did our audience.

Think of your own reaction to seeing a page of words. Now imagine it as a page of numbers.

Instead, picture the number “five”. It’s easy. It might be fingers or it might be a team on a basketball court. But it’s simple to understand.

Now picture the number 275 million. It’s hard. Unfortunately, 275 billion isn’t much harder, even though it’s magnitudes larger. (A million seconds goes by in about 11 days but you may not have been alive for a billion seconds – about 36 years.)

The easiest way to get used to some numbers is to learn ways to cut them down to size by calculating rates, ratios or percentages. In your analysis, keep an eye out for the simplest *accurate* way to characterize the numbers you want to use. “Characterize” is the important word here – it’s not usually necessary to be overly precise so long as your story doesn’t hinge on a nuanced reading of small differences. (And is anything that depends on that news? It may not be.)

Here’s one example of putting huge numbers in perspective. Pay attention to what you really can picture - it’s probably the \$21 equivalent.

The Chicago hedge fund billionaire Kenneth C. Griffin, for example, earns about \$68.5 million a month after taxes, according to court filings made by his wife in their divorce. He has given a total of \$300,000 to groups backing Republican presidential candidates. That is a huge sum on its face, yet is the equivalent of only \$21.17 for a typical American household, according to Congressional Budget Office data on after-tax income. *“Buying Power”, Nicholas Confessore, Sarah Cohen and Karen Yourish, The New York Times, October 2015*

Originally the reporters had written it even more simply, but editors found the facts so unbelievable that they wanted give readers a chance to do the math themselves. That’s reasonable, but here’s an even simpler way to say it: “earned nearly \$1 billion after taxes...He has given \$300,000 to groups backing candidates, the equivalent of a dinner at Olive Garden for the typical American family , based on Congressional Budget Office income data.” (And yes, the reporter checked the price for an Olive Garden meal at the time for four people.)

3.3 Put math in its place

For journalists, numbers – or facts – make up the third leg of a stool supported by human stories or anecdotes , and insightful comment from experts. They serve us in three ways:

- **As summaries.** Almost by definition, a number counts something, averages something, or otherwise summarizes something. Sometimes, it does a good job, as in the average height of Americans. Sometimes it does a terrible job, as in the average income of Americans. Try to find summaries that accurately characterize the real world.
- **As opinions.** Sometimes it's an opinion derived after years of impartial study. Sometimes it's an opinion tinged with partisan or selective choices of facts. Use them accordingly.
- **As guesses.** Sometimes it's a good guess, sometimes it's an off-the-cuff guess. And sometimes it's a hopeful guess. Even when everything is presumably counted many times, it's still a (very nearly accurate) guess. Yes, the "audits" of presidential election results in several states in 2021 found a handful of errors – not a meaningful number, but a few just the same.

Once you find the humanity in your numbers, by cutting them down to size and relegating them to their proper role, you'll find yourself less fearful. You'll be able to characterize what you've learned rather than numb your readers with every number in your notebook. You may even find that finding facts on your own is fun.

3.4 Going further

Tipsheets

- Steve Doig's "[Math Crib Sheet](#)"
- [Appendix A](#): Common newsroom math, adapted from drafts of the book *Numbers in the Newsroom*, by Sarah Cohen.
- A viral Twitter thread:

4 Census Data

We will be using data from the U.S. Census for many of these exercises. Here's a quick rundown of the origins and inner-workings of this important dataset.

Each decade, the Census Bureau counts every person living in the United States and the five U.S. territories. This is known as the Decennial Census and it is used to apportion seats in the U.S. House of Representatives, among other things.

In addition, the Census Bureau conducts ongoing survey of communities, known as the American Community Survey or ACS that provides more timely information about social, economic, housing, and demographic data every year. You can find information on housing, small business ownership, population profiles and much more. The ACS uses an annual sample size of about 3.5 million addresses and is collecting information daily.

Congressional lawmakers look at the ACS data to determine distribution of federal spending, among other things. Read [more about census data here](#)

Here's something important to know about ACS results: Data are pooled across a calendar year. So the ACS numbers reflect data collected over a period of time. By contrast, the Decennial Census is a single point-in-time count of the population.

We will be using ACS data to examine trends in wealth in Baltimore neighborhoods because this survey provides detail not available yet in the Decennial Census.

You can access Census Data through multiple ways. The U.S. Census Bureau offers [data.census.gov](#). Using this site requires some training and patience. Here is a good place to start, a [presentation the Census Bureau staff made](#) to the Investigative Reporters and Editors conference in 2019.

Another useful resource is [censusreporter.org](#), a site not affiliated with the Census Bureau that's designed to make it easier to navigate and retrieve the ACS data.

When we use R, we will use a software library called [tidycensus](#) that makes it very easy to retrieve Census data from the Census API, or application programming interface, basically a raw data feed optimized for R, python and similar programs. Stay tuned on that later this semester.

Data journalist Paul Overberg, now with The Wall Street Journal, compiled [this useful guide about terminology](#) when dealing with Census data.

5 Spreadsheets

Introduction

Some people consider using spreadsheets the table stakes for getting into data journalism. It's relatively easy to see what you're doing and you can easily share your work with your colleagues. In fact, pieces of the [Pulitzer-Prize winning COVID-19 coverage](#) from The New York Times was compiled using an elaborate and highly tuned set of Google spreadsheets with dozens of contributors.

This guide uses Google Sheets, which allows students to do the exercises regardless of their computer operating system, Mac, Windows or Linux. The exercises can be easily adapted to Microsoft Excel, which can handle larger datasets and has more options for pivot tables and other more advanced functions. However, Google Sheets have their own advanced functions for scraping websites or importing non-tabular file formats like JSON.

Most of the screen shots and instructions are created with a MacOS Monterey. Some come from earlier Mac versions, but are largely the same now. Windows users should replace any instructions for using the CMD- key with the CTL- key. There is a table that compares keystrokes for Apple desktops, laptops and Windows machines for Excel at the bottom of [Spreadsheet Refresher](#)

Tutorials

Spreadsheets are used in almost every workplace in America. This section covers most of what you need in the newsroom, which is a different set of skills than in other businesses.

- [Spreadsheet Refresher](#) : Start over with good habits
- [Sorting and filtering to find stories](#) : The first step of interviewing data
- [Grouping with pivot tables](#): Aggregating, and the super power of spreadsheets
- [Spreadsheet Formulas](#): Percents, sums, and other basic computations used in newsrooms.

6 Spreadsheet Refresher

Spreadsheets are everywhere, so it's worth re-learning how to use them well. Reporters usually use spreadsheets in three ways:

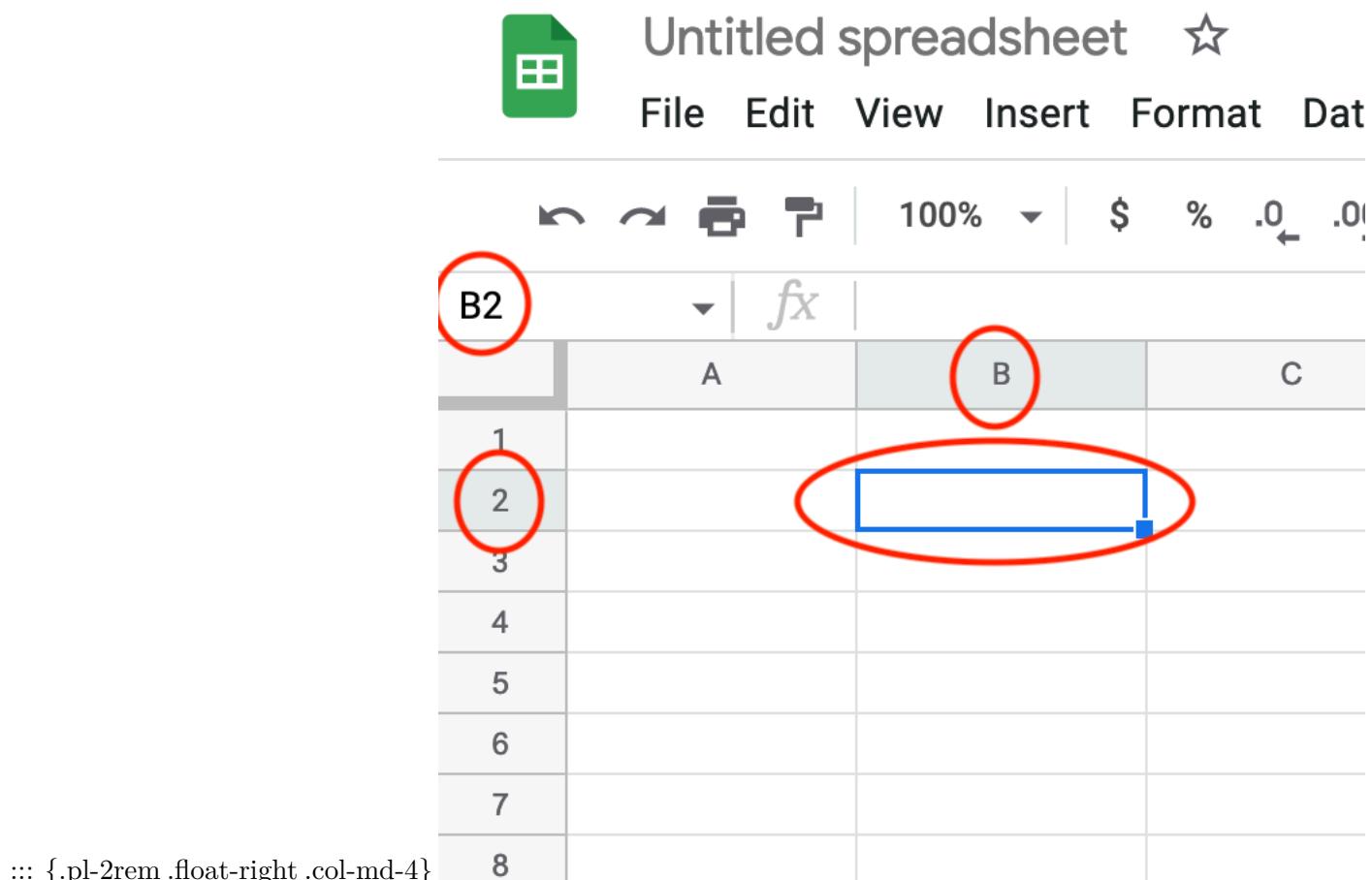
- To create original databases of events for sorting, filtering and counting. Examples include a long-running court case; the details of each opioid death in a city; a list of police shootings and their documents; or even a list of your own public records requests or contact log.
- To use data created by others for fast, simple analysis and data cleanup. Many government agencies provide their information in spreadsheet form, but they often require some rejiggering before you can use them.
- To perform simple, straightforward analysis on data and share with team members. This is becoming less common as more reporters learn programming languages, but it's still common in newsrooms to share data, especially through Google Sheets.

This guide will Google Sheets since the program is available to anyone regardless of operating system. Google Sheets are easy to share for reporting teams.

Some reporters flinch at typing in 30 or 100 entries into a spreadsheet. You shouldn't. If you learn to take notes in a structured way, you'll always be able to find and verify your work. If you try to calculate a sum of 30 numbers on a calculator, you'll have to type them all in at least twice anyway. Also, getting used to these easy tasks on a spreadsheet keeps your muscles trained for when you need to do more.

6.1 Re-learning Excel from the ground up

The spreadsheet grid



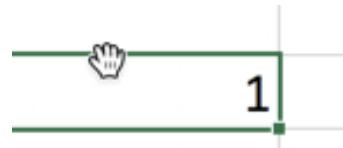
When you start up a spreadsheet, you'll see letters across the top and numbers down the side. If you ever played Battleship, you'll recognize the idea – every little square, or cell, is referenced by the intersection of its column letter and row number:

B2 is the cell that is currently active. You can tell because it's outlined in the sheet and it's shown on the upper left corner.

Mouse shapes



The Copy Tool, or the thin black cross. When you see this, you'll copy anything that's selected. This can be good or bad.



The Evil Hand. If you use this symbol, you will MOVE the selection to a new location. This is very rarely a good idea or something you intend.

Selecting cells and ranges

Spreadsheets act only on the cells or regions you have selected. If you begin typing, you'll start entering information into the currently selected cell.

To select: Hold the cursor over the cell and click ONCE – *not twice*. Check the formula bar to make sure you've selected what you think you've got. You can also look at the bottom right of your spreadsheet for more information.

You'll often work with *ranges* of cells in formulas. These are defined by the corners of the area you want to work on – often a column of information. In the example below, the range is A1:C6, with the “:” referring to the word “through”.

To select a group of cells and act on them all at once: Hover the cursor over one corner, click ONCE and drag to the diagonal corner. Make sure the Evil Hand is nowhere to be seen. The entire area will be shaded in except for the currently selected cell. Look at the upper right corner to see how many rows and columns you selected.

	A1			
1				
2				
3				
4				

To select a column or row : Hover the cursor over the letter at the top of the column. For a row, hover it over the row number in the margin

Reading the screen

The areas of the spreadsheet have different visual clues, and learning to read them will make your life much easier.

This image shows some key areas on the screen when you're just viewing the sheet:

	E4	B	C	D	E	F	G
1	neighborhood	population	white	blk_afam	white - black diff	asian	nathaw_p
2	Abell	88	606	213	393	3	5
3	Allendale	355	18	3497	-3479	6	6
4	Arcadia	523	523	537	86	12	12
5	Arlington	33	2496	-2463	9	9	9
6	Armistead Gar	3458	2698	108	2590	14	14
7	Ashburton	2520	33	2431	-2398	4	4
8	Baltimore Higl	2703	1023	700	323	75	75
9	Patterson Park	5820	2904	1682	1222	143	143
10	Barclay	2181	302	1764	-1462	23	23
11	Barre Circle	450	220	154	74	34	34
12	Beechfield	3708	3216	-2853	29		

Figure 6.1: ready

This is how it changes when you're editing

	A	B	C	D	E	F	G
1	neighborhood	population	white	blk_afam	white - black diff	asian	nathaw_p
2	Abell	889	606	213	=C2-D2)	33	
3	Allendale	3554	18	3497	-3479	6	
4	Arcadia	1235	623	537	86	12	
5	Arlington	2598	33	2565	-2463	9	
6	Armistead Gar	3458	2698	760	-590	14	
7	Ashburton	2520	31	2519	1	4	
8	Baltimore Higl	2703	1023	1680	23	75	
9	Patterson Park	5820	2904	2916	1222	143	
10	Barclay	2181	302	1764	-1462	23	
11	Barre Circle	450	228	154	74	34	
12	Beechfield	3708	363	3216	-2853	29	

Figure 6.2: editing

Entering data

Select the cell and start typing. The information you type won't be locked into the cell until you hit the Return / Enter key, or move your selection to another cell. Hit "Escape" to cancel the entry.

You can't do a lot of things while you're editing, so if you have a lot of greyed out menu items, look at your formula bar to see if you are still editing a cell.

If you're having trouble getting to a menu item or seeing the result of your work, try hitting "Escape" and try again. You may not have actually entered the information into the sheet.

Locking in headings

As your spreadsheet grows vertically with more rows, you'll want to be able to see the top all the time. When it grows horizontally with more columns, you'll probably want to see columns in the left, such as names. This is called "Freezing Panes" – you freeze part of the page so it stays in place when you move around.

Select View in the menu, then Freeze, then the number of rows to freeze. Select 1 row. As you now scroll down the sheet, your headings will remain but you can see the data as you move deeper into the sheet.

Baltimore Neighborhoods .XLSX

File Edit View Insert Format Data Tools Help Last edit was 22 minutes ago

C10 A

neighborhood

Abell
Allendale
Arcadia
Arlington
Armistead Gar
Ashburton
Baltimore Hig
Patterson Park
Barclay
Barre Circle
Beechfield
Belair-Parkside
Bellona-Gittin

	2703	1023	700	
9	5820	2904	1682	
10	2181	302	1764	
11	450	228	154	
12	3708	363	3216	-2853 29
13	444	108	299	-191 7
14	599	481	64	417 35

No rows
1 row
2 rows
Up to row 10
No columns
1 column
2 columns
Up to column C

Figure 6.3: freeze panes

Formatting tricks

- Use the buttons or the format dialog box to make numbers easier to read.
- If a column is filled with a lot of text, select Format, then Wrapping from the menu to wrap text. This means that when you double-click to widen a column, it will get taller, not wider. This is good when you need to save valuable real estate on the screen.

6.2 Getting started with a dataset

SLOW DOWN! Don't do anything until you understand what you have in front of you and can predict what your next mouse click will do to it.

Most data we encounter was created by someone else for some purpose other than ours. This means that you can't assume anything. It may not be complete. It may be inaccurate. It may mean something completely different than it appears at first blush.

First steps

- Document where you got the spreadsheet and how you can get back to the original. Create a new tab (click the + sign in the lower left), name it Data Dictionary, copy the URL of your source data and any other notes about it. Make this your regular practice. It will save time and stress on deadline.
- Read anything you can about what it contains. Look for documentation that comes with the data.
- Save the original into a safe place with its original name and metadata. Work on a copy.
- If the spreadsheet shows ##### instead of words or numbers, widen your columns. If it shows 7E-14 or something like that, format them as numbers, not "General".
- Check your corners – look at the top left and bottom right. Is the data all in one area? Are there footnotes or other non-data sections mixed in? We're going to want to fix that later.

Interview your data

Headings

The most fraught part of data reporting is understanding what each *column* actually means. These often have cryptic, bureaucratic names. You may need to go back to the source of the data to be sure you actually understand them.

If your data doesn't have any headings, that's going to be your first priority. In effect, you'll need to build what we call a *data dictionary* or *record layout* if one hasn't been provided. Many reporters create these as a page in a dataset.

Unit of analysis

A *unit of analysis* refers to the items that are listed in the rows of your dataset. Ideally, every row should be at the same unit of analysis – a person, an inspection, or a city, for example. Summaries should be separated by a blank row, or moved to a different sheet. Think of this as the noun you'd use to describe every row.

Row numbers

The data was probably given to you in some sort of natural sort order. Different computer systems sort differently – some are case-sensitive, others are not. It may depend on when and where the data was created! The order of the data may even depend on a column you don't have. If you don't do something now, you'll never be able to get back to the original order, which could have meaning for both the agency and for fact-checking.

6.3 Video walkthrough

These first steps, along with adding an ID row, are shown here. You can [follow along with the same dataset](#).

Getting started with Google Sheets

[Getting started with Google Sheets](<https://www.youtube.com/embed/1hGoYzmkhfc>)

6.4 Keyboard shortcuts

Google Sheets keyboard shortcuts can be found in the menu: Help, then Keyboard Shortcuts.

Keyboard shortcuts		<input type="text"/> Search keyboard shortcuts	X
Editing	Editing		
Menus	Absolute/relative references (when entering a formula)	F4	
Formatting	Accept Smart Fill suggestion	⌘+Shift+Y	
Data	Copy	⌘C	
Review	Cut	⌘X	
Selection	Define word	⌘+Shift+Y	
Screen reader support	Delete rows/columns or Open delete menu	⌘+Option+-	(i)
File commands	Edit description	⌘+Shift+E	
View	Fill down	⌘D	
Navigation	Fill range	⌘+Enter	
Enable compatible spreadsheet shortcuts		<input checked="" type="checkbox"/>	
VIEW COMPATIBLE SHORTCUTS			HELP

Figure 6.4: Keyboard shortcuts

7 Sorting and filtering to find stories

7.1 A sorting miracle

After [police in Ferguson, Mo., killed Michael Brown in 2014](#), advocates and journalists began examining the racial and ethnic gaps between police departments and the communities they served.

The New York Times found a 7-year-old survey conducted by the Justice Department that allowed it to [compare the data for major cities in a standalone graphic](#) that it published later that year.

When newer data reflecting departments' makeup in 2012 was released a year later, Matt Apuzzo and Sarah Cohen hoped it would show some differences. It didn't. So we were left trying to find news in the data that was clearly of public interest.

Cohen matched up the demographics of police departments with their cities and then started sorting, filtering and Googling. Could there be news in the outliers on the list? Which departments most closely represented their communities? Which ones had unusually large gaps?

Cohen quickly stumbled on telling anecdote to frame the story: Inkster, Mich. had one of the least representative departments in the country, and had recently hired a new police chief to help mend the department's fraught relationship with its largely African-American community. Where had he come from? Selma, Ala., one of the most representative police departments in the nation. Interviews with the chief, William T. Riley III, suggested one reason for some cities' disparities: there was no state or federal money to pay for training new police officers.

The story, "[Police Chiefs, Looking to Diversity Forces, Face Structural Hurdles](#)" helped explain the persistent gap between the makeup of police in some areas and the communities they served.

7.2 Sorting and filtering as a reporting tool

Sorting and filtering can:

- Narrow your focus to specific items that you want to examine in your story.



Figure 7.1: Chief William T. Riley III. Credit: Laura McDermott for The New York Times

- Show you rows containing the highest and lowest values of any column. That can be news or it can be errors or other problems with the data.
- Let you answer quick “how many?” questions, with a count of the rows that match your criteria. (In the next lesson, you’ll see that pivot tables, or group-by queries, are much more powerful for this in most cases.)

7.3 Example data

Data from the Washington Post police shootings database for use in this tutorial - [Documentation from the Post's github site](#) :::

- The data for this and several other chapters is the Washington Post’s public data collection of police shootings in the U.S. It includes the nation’s best guess about each fatal police shooting since 2015. There are a couple of caveats:
- It excludes deadly police interactions other than shooting a firearm at the suspect. Any strangulation, car crashes, Tasers without guns or other methods are excluded.
- It is based primarily on news reports and the results public records requests so it often contains the story as told by police. We know that many of those reports are sugar-coated at best, and lies at worst.
- The Post says this is a list of fatal shootings, but doesn’t say what happens if more than one person is killed. The [2019 shooting of D’Angelo Brown & Megan Rivera in West Memphis](#) is shown as two rows¹ in the data even though it was one event. So each row might be considered a shooting “victim”, a “suspect” or a shooting “fatality” rather than a “shooting”.

The screenshots in this tutorial may not match exactly to what you get on the Washington Post data. This tutorial used data current to Aug. 3, 2022.

It’s a good example set for us because it’s been used as the basis of many stories, it has at least one of each *data type* that we plan to deal with in Google Sheets, and it is [well documented on the Post’s github site](#).

7.4 Get the data into Google Sheets

- Download the [police shooting data from the Washington Post](<https://github.com/washingtonpost/data-police-shootings/releases/download/v0.1/fatal-police-shootings-data.csv>)
- Open Google Sheets. File | Import | Upload | Select the downloaded file “fatal-police-shootings-data.csv”. After it uploads, select the green “Import Data” button.

¹Finding these is something that’s pretty hard in a spreadsheet but will be really easy in R.

•

7.5 Understanding data types

When you open the spreadsheet, the first thing to notice is its *granularity*. Unlike Census or budget spreadsheets, this is a list capturing specific characteristics of each fatality. Each column has the same *type* of data from top to bottom. Those types are:

- **Text.** Text or “character” columns can come in long or short form. When they are standardized (the values can contain only one of a small list of values), they’re called “categorical”. If they’re more free-form, they might be called “free text”. The computer doesn’t know the difference, but you should. The Post data has examples of both. In spreadsheets, text is left-justified (they move toward the left of the cell and will line up vertically at the beginning)
- **Numbers.** These are pure numbers with no commas, dollar signs or other embellishments. In Google Sheets, as we’ll see in the computing section, these can be formatted to *look* like numbers we care about, but underneath they’re just numbers. Adding up a column of numbers that has a word in it or has missing values will just be ignored in Google Sheets. It will trip up most other languages. These are right-justified, so the last digit is always lined up vertically.
- **Logical:** This is a subset of text. It can take one of only two values – yes or no, true or false. There is no “maybe”.
- **Date and times:** These are actual dates on the calendar, which have magical properties. Underneath, they are a number. In Google Sheets, that number is the number of days since Jan. 1, 1900.² They can also have time attached to them, which in Google Sheets is a fraction of a day. What this means is that the number 44,536.5 is really Dec. 6, 2021 at noon. In Google Sheets, you use a format to tell the spreadsheet how you want to see the date or time, just the way you look at dollar values with commas and symbols. (If you get a spreadsheet with a lot of dates of 1/1/1900, it means there is a 0 in that column, which is sometimes a fill-in for “I don’t know.”)

Here’s a picture of a date that is shown in a variety of formats.

Unformatted		Formatted values					
As a number		"Short date"	"Long date"	Time	Date & mil. time	Month	Day of the week
44540.87431		12/10/21	Friday, December 10, 2021	8:59:00 PM	12/10/21 20:59	Dec. 2021	Friday

Figure 7.2: date formats

²Each language deals with dates and times a little differently. We’ll see how R does it later on. But just know that dates can be tricky because of these differences and [time is even more tricky](#)

All of these are the same, underlying value – the number at the left. Notice that all of these are right-justified.

This means that when you see “Friday, December 10”, the computer sees 44540.87431. When you put the dates in order, they won’t be alphabetized with all of the Fridays shown together. Instead, they’ll be arranged by the actual date and time.

It also means that you can compute 911 response times even when it crosses midnight, or compute the someone’s age today given a date of birth. Keeping actual calendar dates in your data will give it much more power than just having the words. (Google Sheets uses the 1st of the month as a stand-in for an actual date when all you know is the month and year.)

7.5.1 Sorting rows

Sorting means rearranging the rows of a data table into a different order. Some reporters take a conceptual shortcut and call this “sorting columns”. That thinking will only get you into trouble – it lets you forget that you want to keep the rows in tact while changing the order in which you see them. In fact, in other languages it’s called “order by” or “arrange” by one or more columns – a much clearer way to think of it.

In Google Sheets, look for the sort options under the Data tab at the top of your screen. In this case, sorting from oldest to newest gives you a list of the fatalities in chronological order, including the time of day.

To sort your data:

- Make a copy of your data. Left click on the “fatal-police-shootings-data” tab, select Duplicate
- Select your data by clicking the box above Row 1 and to the left of Column A
- Select Data | Sort Range | Advanced Range Sorting Options
- Click “Data has header row” and then select date from the Sort by dialog box. Select Z
→ A
- Select Sort

Adding fields to the sort

Adding more columns to the sort box tells Google Sheets what to do when the first one is the same or tied. For example, sorting first by state then by date gives you a list that shows all of the events by state in sequence:

The screenshot shows a Google Sheets interface with a sorting dialog box overlaid. The dialog box title is "Sort range from A1 to S7641". It has a checked checkbox for "Data has header row". Under "Sort by", "state" is selected and "A → Z" is chosen. Under "then by", "date" is selected and "Z → A" is chosen. There is a "Cancel" button and a green "Sort" button. The background table contains columns for ID, Name, Date, Manner of Death, Weapon, and Gender.

7.5.2 Filtering

Filtering means picking out only some of the rows you want to see based on a criteria you select in a column. Think of it as casting a fishing net – the more filters you add, the fewer fish will be caught.

To activate filters in Google Sheets, from the Menu:

- Data | Filter Views | Create a New Filter View
- You'll see little triangles next to the column headings.

Click the “armed” heading. You will see options for various weapons. All are selected by default with a check mark. To select just “ax”, click on clear and then select “ax.” The sheet now is filtered to just weapons using an ax. To remove the filter, repeat the steps and “select all” and the entire sheet is displayed again.

Each filter you select adds more conditions, narrowing your net.

To find fatalities that involved a firearm with a Taser, use the drop-down menu under `manner_of_death` select “shot and Tasered”.

This method works for small-ish and simple-ish columns. If your column has more than 10,000 different entries, such as names or addresses, only the first 10,000 will be considered. We only caught these for stories when someone did a fact-check using a different method of filtering. If your column has a lot of distinct entries, use option that says “Choose One”, and then use the “Contains” option. Better yet, don’t use filtering for counting things at all.

Add more filters to narrow down your list of cases even more. For example, the New York Times ran a series of stories in 2021 about unarmed people shot by police. One story was about those who were fleeing by car. Here's one way to get a preliminary list of those cases:

1. Remove any filter you already have on.
2. Turn on the filters again if you turned them off.
3. Choose “unarmed” under `armed` and “car” under `flee`.

(Of course, the Times didn't stop there in trying to find more cases and teasing out more of them from this and other data. But this is a start.)

Different kinds of filters

There are several filter options. You can filter by various conditions. For numerical data, you can set a minimum or maximum value or a range of values. This is useful for dates to specify a certain time period. For text, you can filter if a word contains a few letters, useful to capture spelling variations.

	armed	age	gender	race	city	state	signs_of_mental_illness	threat_
1	armed					TX	FALSE	undeter
22	unarmed					TX	FALSE	other
882	unarmed					CA	FALSE	other
907	unarmed					CA	FALSE	undeter
1061	unarmed					AZ	FALSE	attack
1640	unarm					AK	FALSE	undeter
1728	un					FL	TRUE	attack
1764	un					TX	FALSE	other
1877	una					IL	FALSE	other
1911	unarm					TN	FALSE	other
1948	unarmed					MN	FALSE	undeter
1951	unarmed					CA	FALSE	attack
1963	unarmed					AZ	FALSE	other
1980	unarmed					GA	FALSE	undeter
2106	unarmed					TX	FALSE	undeter
2390	unarmed					NC	FALSE	other
2471	unarmed					WA	FALSE	other
2684	unarmed					TX	FALSE	other
3277	unarmed					CA	FALSE	other
3293	unarmed					MI	FALSE	other
3310	unarmed					CA	FALSE	other
3531	unarmed					AZ	FALSE	other
3640	unarmed					NV	TRUE	other
3647	unarmed					CO	TRUE	undeter
3653	unarmed					CO	FALSE	other
3666	unarmed					CA	FALSE	other
3671	unarmed					PA	FALSE	attack
3742	unarmed							

8 Spreadsheet Formulas

The quick review of math in Google Sheets uses the City of Baltimore's 2022 budget, compared with previous years.

Get the [Google Sheet](#) to follow along

You should get into the habit of creating unique identifiers, checking your corners and looking for documentation before you ever start working with a spreadsheet. These habits were covered in [Replication and the data diary](#) and on [an Excel refresher](#).

8.1 Formulas in spreadsheets

Every formula begins with the equals sign (=). Rather than the values you want to work with in the formula, you'll use *references* to other cells in the sheet.

The easiest formulas are simple arithmetic: adding, subtracting, multiplying and dividing two or more cells. You'll just use simple operators to do this:

operator	symbol	example
addition	+	=A2+B2
subtraction	-	=A2-B2
multiplication	*	=A2*B2
division	/	=A2/B2

Here's what a spreadsheet looks like while editing some simple arithmetic:

The other kind of formula is a *function*. A function is a command that has a name, and requires *arguments* – usually the cell addresses or the range of addresses that it will act on. Every programming language has functions built in and many have extensions, or packages or libraries, that add even more as users find things they want to do more efficiently. You begin using a function the same way you begin a formula – with an = sign. Here are three common functions that create summary statistics for the numbers contained in a *range* of addresses. A range is a set of cells defined by its corner cell address: the top left through the bottom right.

You'll usually use them on a single column at a time.

	A	B	C	D	E	F
1	TABLE 5E-1					
2	MORTALITY BY COUNTY OF RESIDENCE AND YEAR, ARIZONA, 2006-2016					
4		2015	2016			
5	ARIZONA	54,152	56,480	=C5-B5		
6	Apache	646	653			
7	Cochise	1,305	1,342			
8	Coconino	814	857			
9	Gila	814	832			

Figure 8.1: formula

Formula	What it does
=SUM(start:finish)	Adds up the numbers between start and finish
=AVERAGE(start:finish)	Computes the mean of the numbers
=MEDIAN(start:finish)	Derives the median of the numbers

...where “start” means the first cell you want to include, and finish means the last cell. Use the cell address of the first number you want to include , a colon, then the cell address of the last number you want to include. You can also select them while you’re editing the formula.

Here’s an example of adding up all of the rows in a list by county:

8.2 Common spreadsheet arithmetic

The budget document shows three years’ of data: The actual spending in the fiscal year that ended in 2016; the spending that was estimated for the end of fiscal year 2017; and the proposed spending for fiscal year 2018. The first page of the document shows these amounts for broad spending categories.

You may want to widen the columns and format the numbers before you start:

8.2.1 Check the government’s math with SUM

Our first job is to make sure the government has provided us data that adds up. To do that, we’ll SUM all of the departments’ spending.

To add up the numbers from FY 2020 Actual, enter the following formula in cell C16, just below the number provided by the government:

=SUM(C2:C13)
and hit the enter key

Copy that formula to the right. Notice how the formula changes the addresses that it is using as you move to the right – it’s adjusted them to refer to the current column.

	A	B	C
1		year_2015	year_2016
2	Apache	646	653
3	Cochise	1,305	1,342
4	Coconino	814	857
5	Gila	814	832
6	Graham	251	278
7	Greenlee	69	52
8	La Paz	254	270
9	Maricopa	28,945	30,311
10	Mohave	3,024	3,181
11	Navajo	907	1,010
12	Pima	9,241	9,492
13	Pinal	2,968	2,991
14	Santa Cruz	294	301
15	Yavapai	2,918	2,955
16	Yuma	1,427	1,506
17	Unknown	275	449
18			
19		=SUM(B2:B17)	
20			

Figure 8.2: formula

8.2.2 Change in spending

The increase or decrease in projected spending from 2017 to 2018 is just the difference between the two values, beginning in cell F3

new-old, or =E2-D2

When you copy it down, note how the references to each row also adjusted. In line 3, it's E3-D3, and so on. Excel and other spreadsheets assume that, most of the time, you want these kinds of adjustments to be made.

8.2.3 Percent change

We can't tell the *rate* of growth for each department until we calculate the percent change from one year to another. Now that we already have the change, the percent change is easy. The formula is:

(new - old) / old

... or just scream "NOO"

The new-old is already in column F, so all that's left is to divide again. In grade school, you also had to move the decimal place over two spots, since the concept of percent change is "out of 100". Excel formats will do that for you.

Remember, it's always (new-old)/old , **NOT** the big one minus the little one. Doing it correctly, the answer could be negative, meaning the value fell.

B	C	D	E	F	G
	Act 2016	Est 2017	Budget 2018	change from 2017	percent chan
rnment	\$103,679	\$111,601	\$129,653	\$18,052	=F2/D2
ce	\$874,560	\$930,155	\$1,032,609	\$102,454	
on	\$53,367	\$61,104	\$63,272	\$2,168	
development	\$541,951	\$605,421	\$650,439	\$45,018	
enrichment	\$176,318	\$207,237	\$239,428	\$32,191	
al services	\$243,029	\$259,893	\$268,072	\$8,179	
	\$378,827	\$429,960	\$450,858	\$20,898	

Figure 8.3: formula

When you're done, you can format the answer as a percentage to get it into whole numbers.

It's also worth comparing the picture you get by looking at raw numbers vs. percentages. It's instructive that federal grants are up 308%.

8.2.4 Parts of a whole: percent of total

We'd also like to know what portion of the total spending is eaten up by each department. To do that, we need the percent of total.

In our case, let's use the total that the government gave us. In practice, you'd have to decide what to do if your figures didn't match those provided by officials. You can't assume that the total is wrong – you could be missing a category, or there could be a mistake in one of the line items.

The formula for percent of total is:

`category / total`

Here's a good trick with spreadsheets when you need to divide against a fixed total. You don't have to type in each formula one by one, though. Instead, you'll use anchors, known in spreadsheets as "absolute references". Think of a dollar sign as an anchor or stickpin, holding down the location of part of your formula. If you put the stickpin before the letter in the formula, it holds the column in place. If you put it before the number, it holds the row in place. If you put it in both places, it holds the cell in place.

In this case, we want to see what percentage property taxes, income taxes, etc. are of the total revenues in FY22, which is \$4,331,049,486 (cell E14). Let's figure it out.

- Left click column F, insert column to the left. Name it Pct of Total
- Create formula to divide property taxes into total revenues: =(E2/E14)
- Modify formula so it will anchor to the E14 as you move down the spreadsheet
=(E2/\$E\$14)
- Copy formula down to F13

8.3 While we're at it: two kinds of averages

Although it doesn't make a lot of sense in this context, we'll go ahead and calculate the *average* or *mean* size of each department, and then calculate the *median* size.

Simple average, or mean

A simple average, also known as the mean, is skewed toward very high or very low values. Its formula is

`sum of pieces / # of pieces that were summed`

But in Google Sheets, all we need is the word AVERAGE:

`=AVERAGE(C2:C9)`

Median

In Google Sheets, you can get the median of a list of numbers by just using the formula, `MEDIAN()`

`= MEDIAN(C2:C9)`

Doing simple calculations like this on data that is provided to you by the government lets you ask better questions when you get an interview, and may even convince officials to talk with you. There's a big difference between asking them to tell you what the budget numbers are, and asking them to explain specific results!

8.4 FAQs

Should I use average or median?

It depends. Averages are easier to explain but can be misleading. Usually, if they're very different, median will be a better representation of the typical person, city or department. Averages in these cases are more like totals.

My percents are small numbers with decimal points

Use the format as a % button to move the decimal point over two places and insert the percentage symbol.

9 Grouping with pivot tables

In the wake of a police shooting in 2016, reporter Mitch Smith obtained a [list of traffic stops](#) from the St. Anthony Police Department in Minnesota. He was writing a story on Philando Castile's death and was running out of time. He wanted to answer a simple question: Were minority motorists more likely to be stopped in St. Anthony than whites?

Rob Gebeloff made a quick pivot table to answer the question. That night, [Smith wrote](#):

In each of the three small suburbs patrolled by the St. Anthony police, less than 10 percent of the population is black. But data released by the city on Tuesday showed that a far higher percentage of the people ticketed or arrested by St. Anthony officers were African-American.

Last year, around 19 percent of those cited by St. Anthony police were black, as were roughly 41 percent of people arrested by the department, a review of the city's data showed. Those percentages do not include the large number of defendants whose race was unknown.

Summarizing a list of items in a spreadsheet is done using pivot tables. In other languages, it's considered "aggregating" or "grouping and summarizing". Think of pivot tables and grouping as answering the questions, "How many?" and "How much?" They are particularly powerful when your question also has the words "the most" or the "the least" or "of each". Some examples:

- Which *Zip Code* had *the most* crimes?
- What *month* had *the least* total rainfall?
- *How much* did *each candidate* raise last quarter?
- In playing cards, *how many* of *each suit* do I have in my hand?
- On average, are *Cronkite students* *taller or shorter* than in other schools?

Confusing grouping with sorting or arranging

Many reporters confuse this summarization with "sorting". One reason is that this is how we express the concept in plain language: "I want to sort Skittles by color."

But in data analysis, sorting and grouping are very different things. *Sorting* involves shuffling a table's rows into some order based on the values in a column. In other languages,

this is called *arranging* or *ordering*, much clearer concepts. *Grouping*, which is what pivot tables do, is a path to aggregating and computing summary statistics such as a count (the number of items), sum (how much they add up to), or average for category. It means “make piles and compute statistics for each one.”

When to use filter vs. pivot tables

Something that trips up beginners is a desire to see details and totals at the same time, which is more difficult than it sounds.

A filter is used to *display* your selected items as a list. You’ll get to see all of the detail and every column. As a convenience, Google Sheets shows you how many items are in that filtered list. That’s great when you want to just look at them, or get more information about them. For instance, if you had a list of crimes by ZIP code, you might just want to see the list in your neighborhood – where, exactly, were they? When did they happen? Was it at night or the morning? What crimes happened on which blocks?

A pivot table is used when you *just want to see summaries* – does my ZIP code have more crime than others? Are robberies more common than car theft in my Zip code, and how does that compare to others?

In practice, you’ll go back and forth between summary and detail. They’re both important, just different.

9.1 Tutorial

::: {.alert .alert-info .opacity-2} We will continue to use [data from the Washington Post police shootings database](#) for this tutorial.

First, let’s modify this spreadsheet to include the descriptions of race and ethnicities: A for Asian, B for Black, etc.

- Select Column I, city, and insert a new column to the left. Name it race_ethnicity
- Create a filter. Select race, filter for A
- Type Asian in Column I. Copy Asian down the entire column so every A in column H corresponds with Asian in Column I
- Repeat: B = Black. H = Hispanic. W = White, non-Hispanic, N= Native American, O=Other, blanks=Unknown.

Follow [this video](#) to see the process.

Setting up the pivot table

From the main menu on Google Sheets, choose *Insert*, then *Pivot table*, then New sheet.

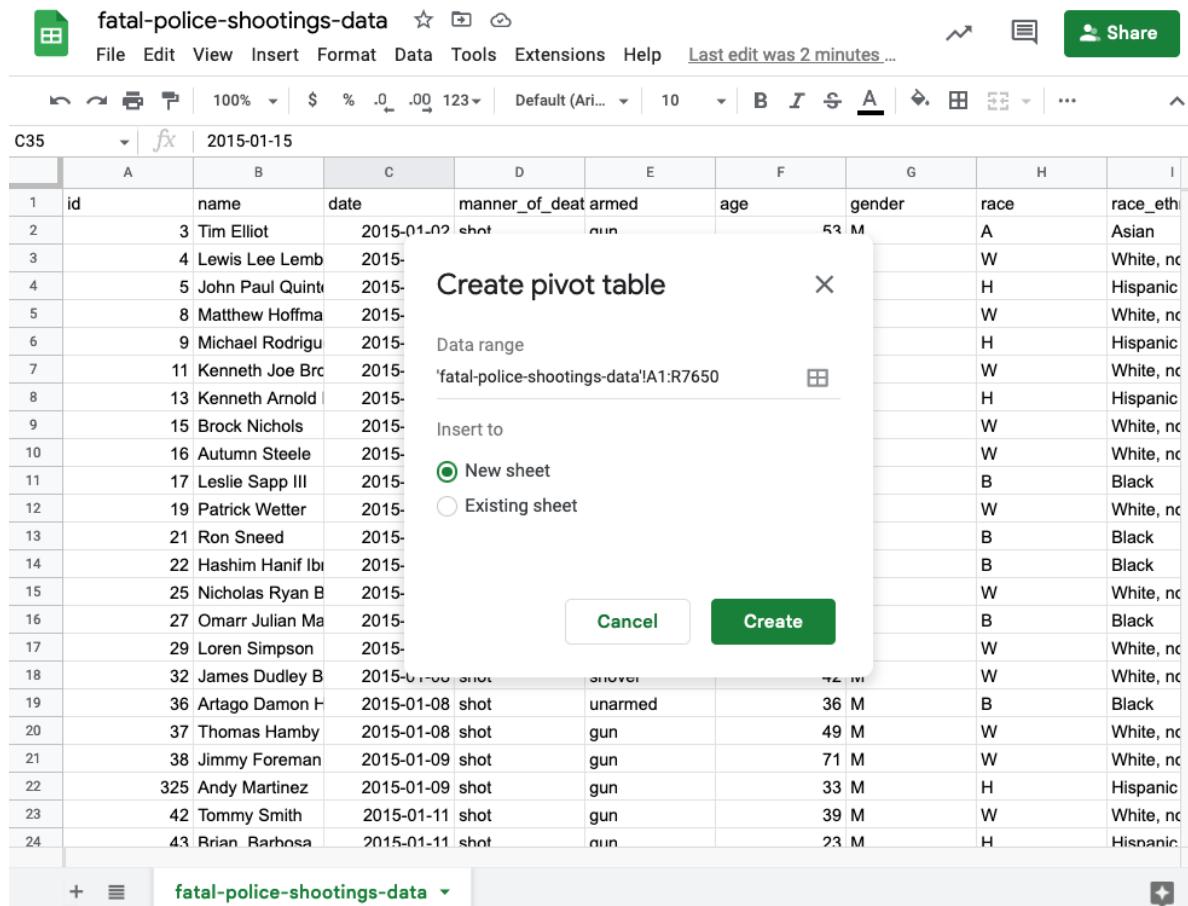


Figure 9.1: insert menu

Next, you will see the Pivot Table editor. Here's what it looks like:

Counting , or “how many”?

For Rows, select Add and then race_ethnicity. For values, select Add and then state. You will now see all of the race and ethnicity totaled.

We're totalling on state because it's good to have something that's always filled out into the Values area (`state` is a safe one in this data).

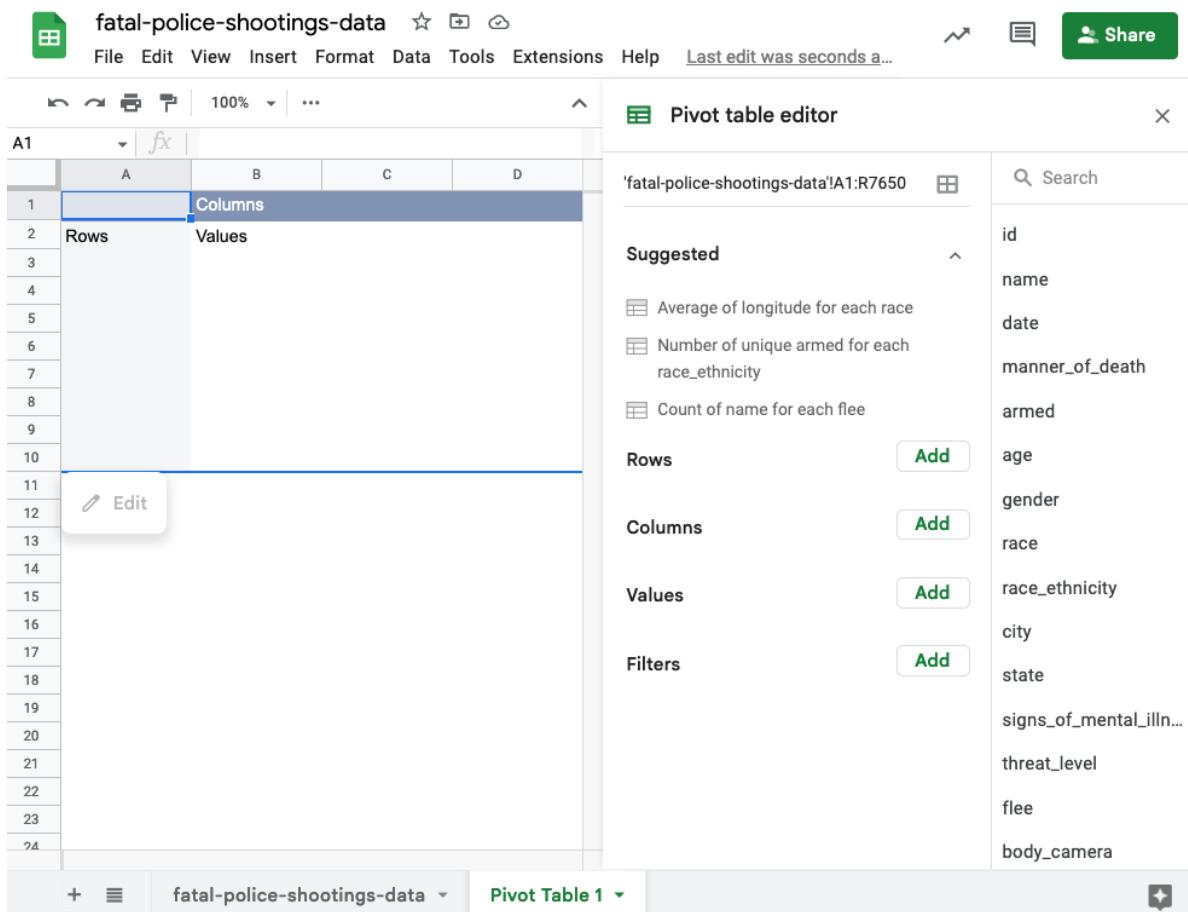
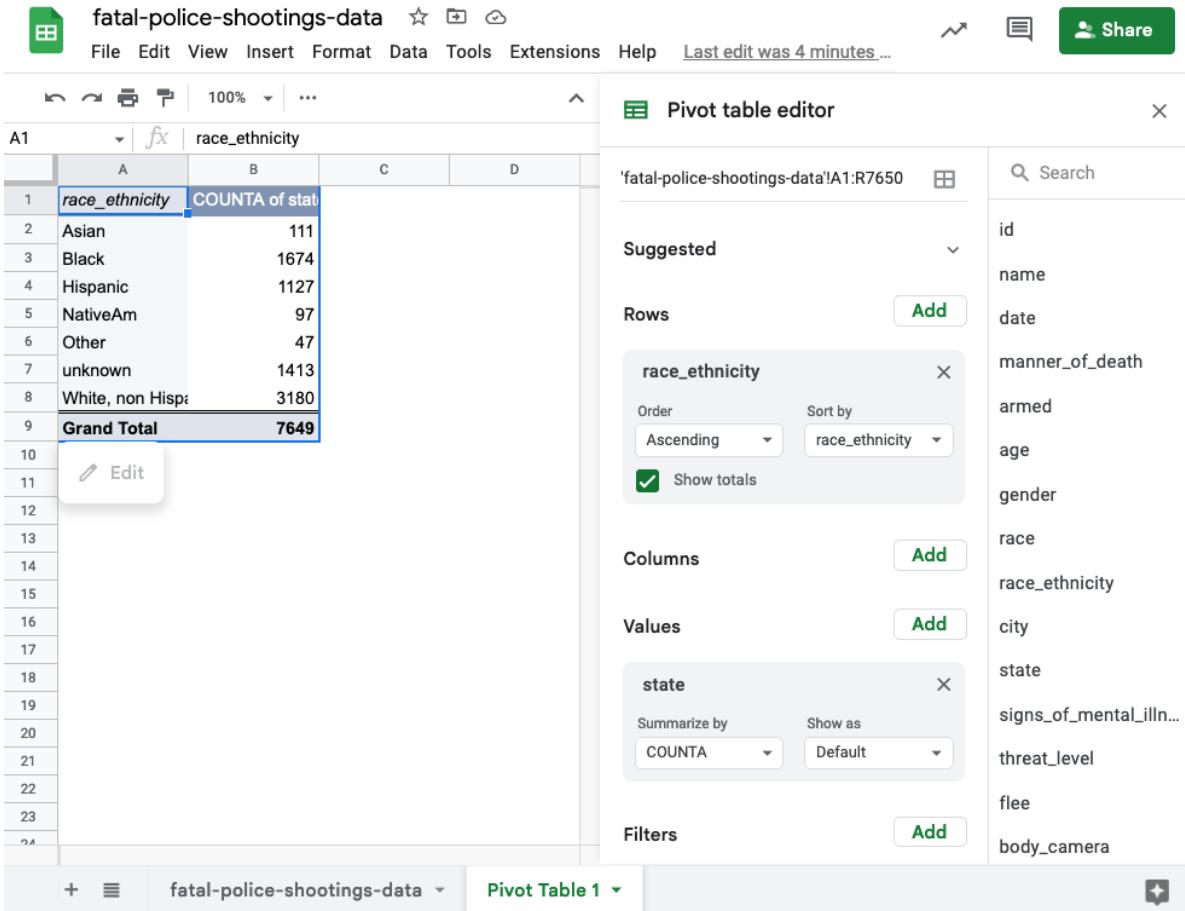


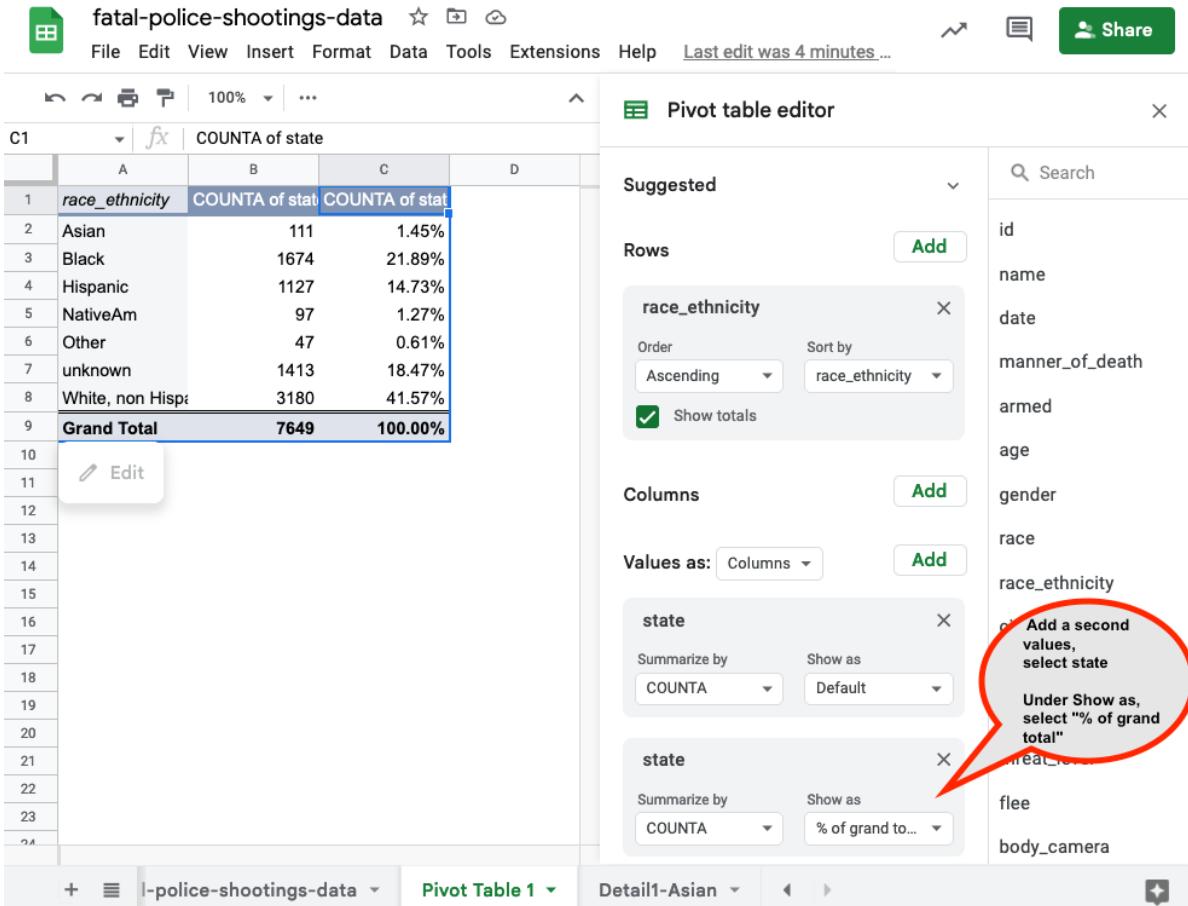
Figure 9.2: pivot menu



Percents of total

It's hard to compare raw numbers unless they're really small. Instead, we'd like to know what *percent* of fatalities by ethnicity. To get a "Percent of Column total", do the following:

- Add a second values, select state
- Under Show as, select "% of grand total"



More variables

Suppose you'd like to see the number of fatalities by year, with the years across the top and the ethnicity down the sides. Add a year variable to columns

- Remove the percent of total column
- Select Columns, then year

COUNTA of stat year

	A	B	C	D	E	F	G
1	COUNTA of stat year	2015	2016	2017	2018	2019	2020
2	race_ethnicity	15	15	16	21	20	15
3	Asian	15	15	16	21	20	15
4	Black	258	236	222	228	251	243
5	Hispanic	173	161	180	169	168	171
6	NativeAm	9	17	22	16	13	9
7	Other	14	11	6	4	9	3
8	unknown	23	53	77	94	114	120
9	White, non Hisp	502	465	458	461	424	459
10	Grand Total	994	958	981	993	999	1020

Even more variables

Say you wanted to see each city's total shootings by year. Which one had the most last year, and which one had the most overall?

This is actually really hard in a pivot table, because there are cities with the same names in different states. It means you'd need to have a pivot table with TWO columns down the side, and one across the top. Here's an attempt at solving the problems:

- First, Rows, add state
- Rows, add city
- Values, add state, CountA is the default
- Columns, add year

The problem is we can't sort by the combination of city and state. But it does help answer the question on some level.

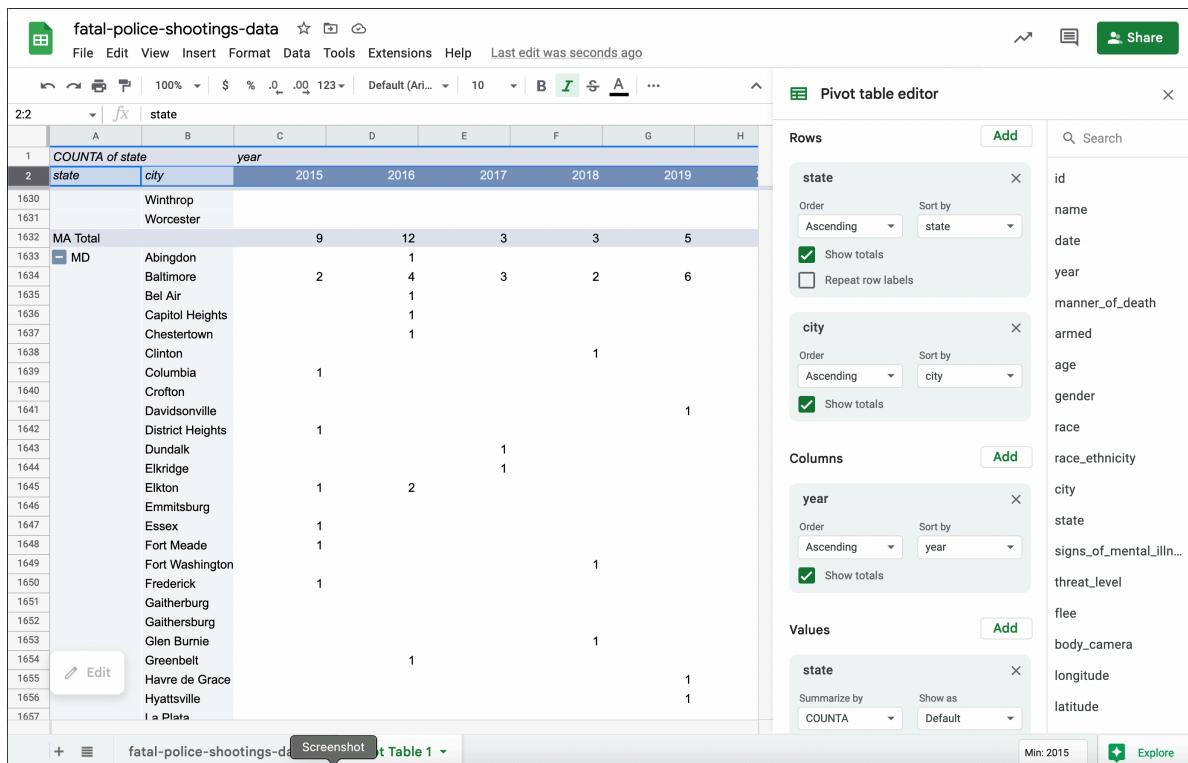


Figure 9.3: More Variables!

9.2 FAQ

I have too many columns

If you want two sets of statistics – say, number of fatalities and percent of fatalities – across the top, it can get very wide and confusing very quickly. One alternative is to change it into more of a vertical rectangle by dragging the “Values” element from the columns to the rows on the right. (This only shows up when you have two calculations being made.)

I want to sort by percents, not numbers

You can't.

Things aren't adding up

You have to be super careful about which column you use to Count things – it has to always be filled out (there can't be any blanks). Go through the filters and find one that doesn't have (Blanks) at the bottom to be sure.

Its a crazy number!

You might have dragged a numeric column into the “Values” area. Check to see if it says “Count” or “Sum”. Change it to “Count” if it has something else on it, unless you wanted to add up that column.

This is so frustrating - I can't get what I want

Right? It's time to go to a programming language!

10 Cleaning data with Google Sheets

This chapter will demonstrate some of the bedrock data cleaning skills using Google Sheets, techniques that can be used in Excel and other spreadsheets. We will normalize and clean data by deleting rows, stripping whitespace, making characters lowercase or uppercase. In addition, you will learn to split text to columns, a very handy tool for splitting up dates.

Make a copy of your data before cleaning)

We will use a version of the Washington Post [police shooting data](#) to conduct these exercises.

10.0.1 Text to columns

We want to split up the date field into day, month and year. Currently, the format is 2015-01-02. Luckily, the fields all share a common separator, a hyphen, and we can ask Google Sheets to split all according to the hyphen. Other common separators are commas and spaces.

First steps when modifying data: make a backup copy! - Left click on the tab “Police Shootings to Clean” - Select duplicate - Rename “Copy of Police Shootings to Clean” to “Original Police Shootings to Clean.” Do not touch this version.

Time to split text to columns. I am extra paranoid (for good reason) and so I always duplicate a date field before modifying it. Duplicate the date column (click on Column C, left click, copy, then Insert column to left, select new blank Column C and paste), save the copy as date-original.

- Select date column
- Select Data | Split text to columns
- See a dialog box: Separator. Select Custom and type in a dash - and enter. You now have the date field chopped up to year, month and day. Rename column E for month and column f for day.

10.0.2 Normalizing

Scroll down the race_ethnicity column and you will see a number of different categories for the same thing: white, White, non Hispanic and Black, African Am. To see all the variations of categorical variables, create a filter and [check the different variables](#)

This presents a big problem when you are trying to group and summarize based on these variable names. See [this chart](#)

We see white totals 44 and White, non Hispanic total 3,136. We want those to be together – the total is 3,180 – because they are the same thing. Also note that African Am totals 29 and Black totals 1,645, and we would want to combine those as well.

Let's fix it!

Before changing any data, let's work with a copy of the column. - Select race_ethnicity (Column k), left click, copy - Left click on Column K, insert column to right, paste - Rename as race_ethnicity2

Renaming variables. We will rename all “white” as “White, non Hispanic” - Filter race_ethnicity (Column K) to white - in race_ethnicity2, write “White, non Hispanic” in the first column and copy down the list

See [how this process works](#)

10.0.3 Lowercase or Uppercase character conversion

Create a filter and notice two variations on Native American: NativeAm and nativeam. You can resolve these differences easily by converting all to Upper or Lower case text using the =UPPER or =LOWER functions.

- To convert NativeAm to lower case, filter on race_ethnicity (Column K) for NativeAm.
- In race_ethnicity2 (Column L), insert a blank column, and type the function =LOWER(K67) and hit enter.

The result should be nativeam as the first entry in race_ethnicity2.

See [this example](#)

10.0.4 White space

One obnoxious feature of spreadsheet data is the invisible “white space” or hidden character or carriage returns that can impede your ability to group and summarize variables. Look at the age column. See how some numbers are flush left while most are flush right. The flush left data has hidden white space. You can fix this by clicking on individual cells and deleting the space around the number or you can do it with a function.

- Select age (Column H), left click on Column H, insert column to right, rename as age2
- In cell I2, type =TRIM(H2) and enter. Copy the formula down.

Note how all of the values have been normalized.

These are some of the basic go-to tools for data cleaning in Google Sheets, which can be adapted to Excel, R and other programming languages.

11 Getting started with R and RStudio

In this chapter

- Install R , RStudio
- The power of packages, especially the `tidyverse`
- Set up defaults for reporting with data
- Issue your first R commands
- Work in projects
- Relax!

This is probably your first introduction to coding. Don't be worried. With effort, much of what reporters do in coding can be learned in a few weeks.

Like most reporters, I learned the coding that I know (which isn't a lot) because I wanted to get a story done. In our class, we are not trying to become a programmer or social scientist. We're working on stories.

You saw during the pivot table lesson that spreadsheets have limits. We couldn't easily get the city with the most police shootings because we would have had to put both city and state into the pivot table. A median is missing from pivot tables entirely. It's easy to lose track of where you are and what you did. That's the reason to learn some coding – there is something we want to know that isn't very easy to get in other ways.

All programming languages have one thing in common: You write instructions, called algorithms, and the program executes your statements in order. It means you can do more complicated work in computer programming than in point-and-click parts of Excel. It can also scale – you can repeat your instructions millions of times, tweak it a little, and re-run the program without messing anything else up. The computer won't mind. Really.

Writing code can also be self-documenting. You can largely eliminate those painstaking Excel data diaries and replace them with documents that explain your work as you go. You'll still need to record information about your interviews and decisions, but you'll no longer have to write down every mouse click.

If you're nervous about getting started with programming, take look at the Appendix: [A gentle introduction to programming](#) and Jesse Lecy's “[Learning how to Learn](#).”

R or Python?

If you ask a data scientist or technologist which language you should learn first, you'll start a heated debate often between advocates of R, Python, Javascript , SQL, Julia and others. Ask the same question of a data journalist and the answer will be: "Choose one that is free and that your colleagues use so you can get help." For our purposes, it really doesn't matter – any of the standard languages will do.

The only rule I would make is to try to stick to your first language for a little while before deciding you want to learn a different one. It would be like trying to learn Portuguese and Spanish at the same time, when you know neither one to begin with. They're related, but very different.

Employers who hire data reporters usually don't care which programming language you know because it's relatively easy to learn another once you're comfortable with the concepts and good data journalism habits. In a few cases, such as the Associated Press, R is preferred.

11.1 Install R and RStudio

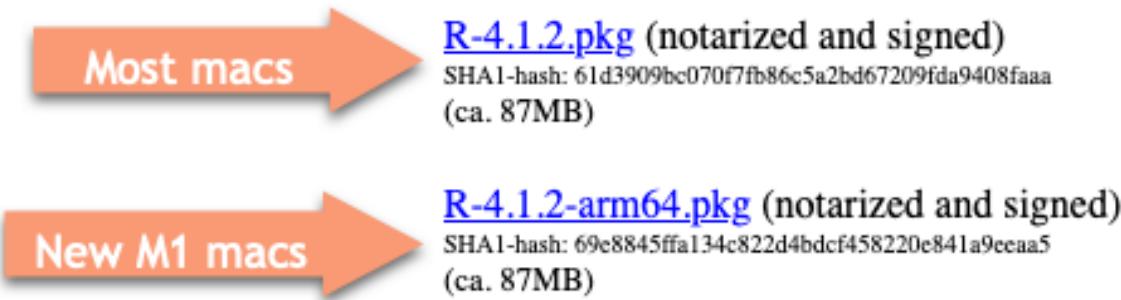
- R is the programming language itself, and has to be installed first
- RStudio is a software program that makes it easier to interact with the programming language. Install it second.
- Packages are sets of programs written by volunteers and data scientists that perform specialized jobs more easily than working with the “base” R language. A package must be installed once on your computer, then invoked to use them in a program.

Follow this interactive tutorial on installing R, RStudio and the tidyverse on your computer:

<https://learnr-examples.shinyapps.io/ex-setup-r/#section-welcome> .

There are two differences between the video and today:

- The tidyverse will take much longer to finish installation. It has a lot to do and often looks like it's stalled.
- There are two versions of R for Mac users: The traditional one and the one for the new M1 chip on the latest machines. Choose the one that matches your machine by checking the apple in the upper left and looking at “About this Mac”. It will say “Apple M1” as the processor if you have it.



11.2 Unlocking packages and the tidyverse

The real power of R comes with packages. Packages are bundles of programs that others have found useful to extend the base R language. R is almost useless without them. There are more than 10,000 packages available for R, each doing a special job.

If you followed along with the tutorial, the last thing you did was install a “package” called the **tidyverse**. Almost everything we do from now on depends on that step.

The tidyverse packages up a whole set of other packages that are designed to work together smoothly with similar grammar and syntax. It’s particularly useful for the kind of work reporters do – importing, cleaning and analyzing data that we get from others and can’t control how it’s structured.

We’ll be working almost exclusively within the tidyverse in this course. I strongly suggest that when you Google for help, put the word “tidyverse” somewhere in your query. Otherwise, you may get answers that look inscrutable and unfamiliar.

The tidyverse is the brainchild of Hadley Wickham, a statistician from New Zealand, who famously identified **tidy data** principles. He’s currently the chief data scientist for RStudio in Houston.

11.3 Set up RStudio for data reporting

Staying organized is one of the challenges of data reporting – you’re constantly re-downloading and re-jiggering your analysis and it’s easy to get your material separated. This setup helps

ensure that you always know where to find your work and can move it to another computer seamlessly.

Before you start, decide on a folder you'll use to store all of your R work. Within my Documents folder, I created a sub-folder called **data-class**. It will make this guide a little easier if you do the same thing, especially if you're not very familiar with using directories and folders.



Start up RStudio once you've made your folder. Make sure you start up RStudio (not the R language) by searching for it in Spotlight or in the Search bar in Windows. Here's what they look like:

Get to the Preferences (under the RStudio menu item on a Mac) and make sure it looks like this in the General tab:

(I've turned OFF all of the options to restore anything when you start up RStudio and set up a default working directory by browsing to the one I just created.)

Under the R Markdown options, make sure that the box called "Execute setup chunk automatically" is checked.

11.4 The screen

This is what your screen probably looks like:

The Console

The Console is where you can type commands and interact directly with the programming language. Think of it as a very powerful calculator at first. One reason to use it is to install packages.

If you followed the installation demo, you've already used the console to install one package. (Go back and do that part now if you skipped it.) Install a few more that will be useful in this course.

Copy these commands one at a time, and paste them into the Console, then hit Return/Enter to execute the command.

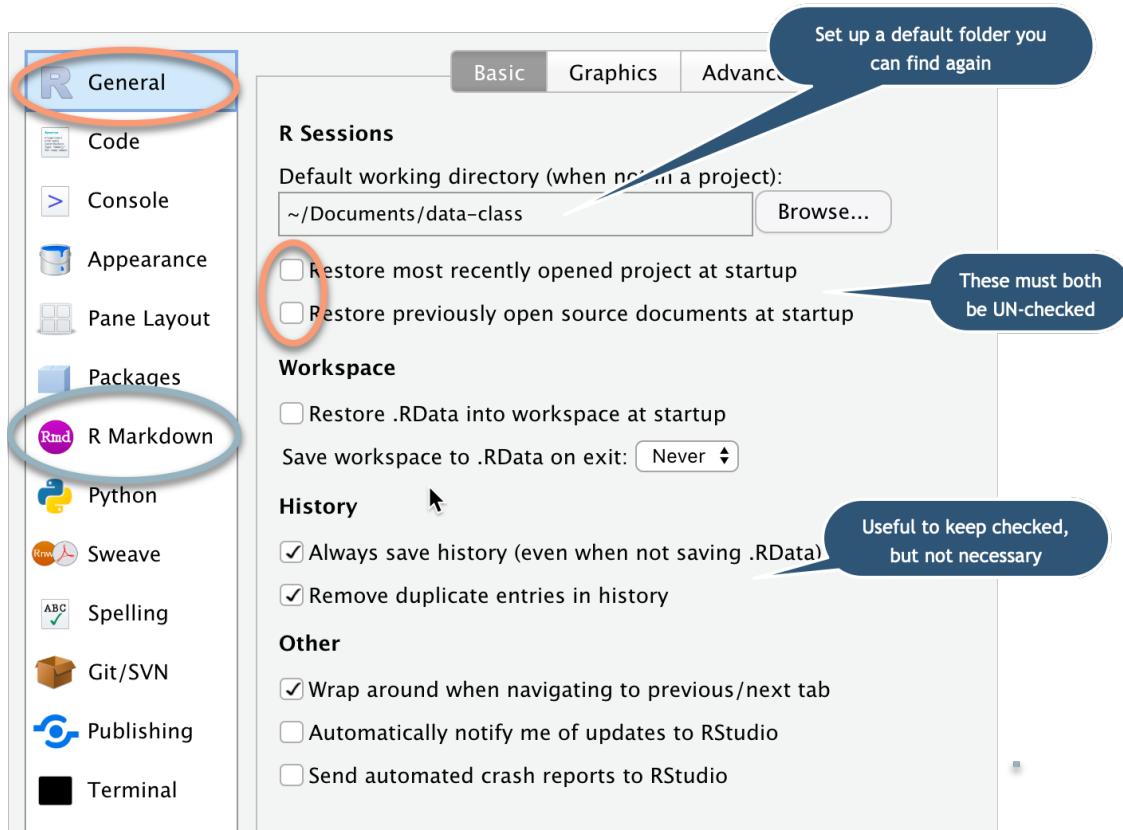


Figure 11.1: see below

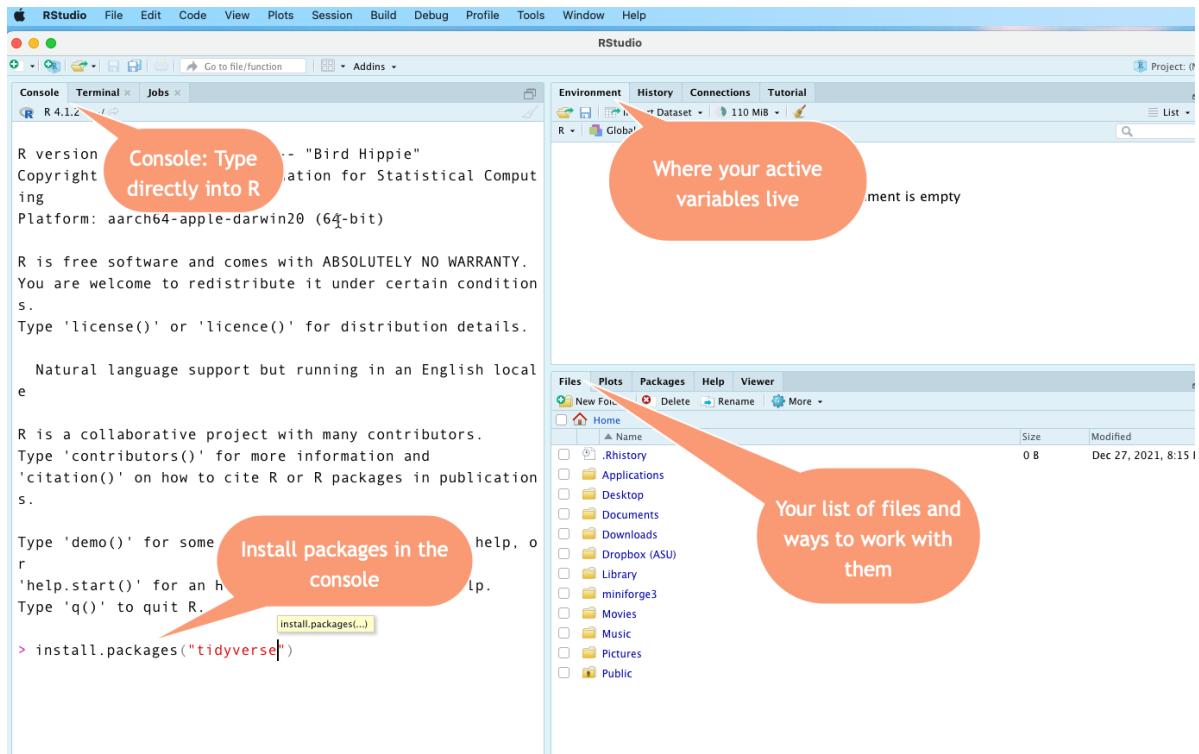


Figure 11.2: console

```
install.packages("janitor")
install.packages("rmarkdown")
install.packages("skimr")
install.packages("swirl")
```

These package names should all be in quotes. We'll be installing other packages later in this guide, but for now that is everything you need.

Files tab

We won't be using many of the tabs in the lower right, but the Files tab can help you if you're having trouble navigating your work. Under the More button, you can choose "Go to working directory", since that's where R thinks you've parked all of your work. This can be confusing in R, which is why we'll be working in "projects" that bundle up all of your work in one place.

Environment

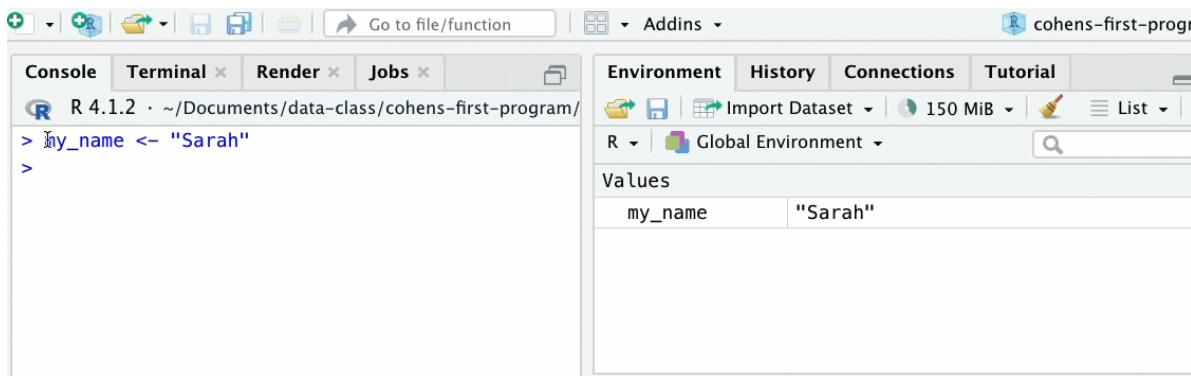
The upper right screen is the Environment, which is where your active *variables* live. A variable is a named thing. It might be a word, a list of words or numbers, or a data frame (spreadsheet). Anything that you want to use has to be listed in that environment before you can reference it. This will make more sense later.

Typing into the console

When you type this: 5+5 after the > prompt, you'll get this back after you press Return/Enter:
[1] 10

When you type this: "Merrill" (with quotes) after the > prompt, you'll get this back: [1]
"Merrill"

To create a new variable, you'll use the *assignment* operator <- (two characters : A less than sign and a hyphen). Here is how I would create the variable called `my_name` (lower case, no spaces). Notice how it appears in the Environment after being created. Then I can print it by typing the name of the variable instead of the letters of my name in quotes:



The console remembers your commands, but you have to type them one at a time and it will forget them when you leave for the day. That's why we're going to work in programs called R Markdown documents most of the time.

11.5 Working directory

Here's how to set your working directory so you can keep your files organized. First, you should have created a folder within the Documents folder called `data_class`. Run the following command to see if R Studio is pointing to that folder

```
getwd()
```

```
[1] "/Users/smussend/Desktop/git_repos/data_journalism_interactive_textbook/03_tutorials/qmd/
```

If it is not, you can navigate R Studio to that folder using the Files tab in the lower right corner window. Once you find your `data_class` folder, then select the Set Working Directory option under the More menu.

You can also set the path programmatically using `setwd()` which means set working directory. Just find the path directory using Finder – [directions are here](#) – and copy that link to this command.

```
setwd("/Users/YOURNAME/Documents/data_class")
```

11.6 Interactive R tutorial

One of the packages you installed earlier was called `swirl`. Invoke it now by typing `library(swirl)` into the Console. You can follow the instructions from there. Don't bother going beyond the first chapter – it's more geared at other kinds of jobs than ours.

Relax by Silwia Bartyzel via Unsplash

11.7 Relax!

You're all set up and we're ready to start programming. Congratulate yourself - everything is new, nothing is intuitive and the screen is intimidating. You've come a long way.

11.8 Other resources

Sharon Machlis' [Practical R for Mass Communications and Journalism](#) has an intro to R and RStudio in chapters 2.3 through 2.6

Ben Stenhaug created a fast-paced video introducing the RStudio interface. Don't worry too much about what some of it means just yet – just see if you can get used to the different parts of the screen.

12 Loading and Analyzing Data

This tutorial, based on a 2022 presentation for the National Institute for Computer Assisted Reporting (NICAR), will show you the essential workflow for any data analysis project in R. You will learn to import data, explore it, use basic commands to sort and filter and build summary tables. You will build a basic data visualization from your work – all in R code.

You will be executing commands in this document that are contained in “chunks,” which are separate from the text and contain live R code. **Click the green arrow at the right on line 6** and run the `help.start()` command

```
help.start()
```

In the bottom right window of R Studio, you will see a Help window that displays basic help commands for the program.

12.0.1 Install software to grab data

Tidyverse: Eight separate software packages to perform data import, tidying, manipulation, visualisation, and programming

Rio: Easy importing features

Janitor: Data cleaning

You should have installed tidyverse already. If not, then delete the hashtag in front of `install.packages("tidyverse")` and run the code chunk at line 22.

```
#install.packages("tidyverse")
install.packages("rio")
install.packages("janitor")
```

Remember, package installation usually is a one-time thing on your hard drive. But when you need to load the software libraries each time you start a script. Libraries are bits of software you will have to load each time into R to make things run.

```
library(tidyverse)
library(rio)
library(janitor)
```

Check to see what's installed by clicking on "Packages" tab in File Manager, lower right pane

12.0.2 Data

We will work with a dataset of MediaSalaries that I cleaned and modified slightly in March 2022 for this tutorial. Some of the detail has been removed so we can make calculations. This MediaSalaries sheet was a crowdsourced project involving reporters worldwide to share information about salaries and benefits. Open this [file in Google Sheets](#)

1) Select Salaries tab

IRE Old School: Four Corners Test!

13 Columns

1658 Rows

Numeric data in Salary, Years Experience

Mixed string data in Gender Identity / Ethnicity, Job duties

12.0.3 Import Data

We'll now load this data into R. You can load spreadsheets from the Internet as follows:

```
MediaBucks <- rio::import("https://docs.google.com/spreadsheets/d/1jkbQFwIdaWv8K00Ad6Wq7Zx
```

What happened? Look at the table

```
View(MediaBucks)
```

What happened?

R grabbed the spreadsheet from the folder

We told R to grab the first sheet, RealMediaSalaries2

R created a dataframe called MediaBucks

basics of R: <- is known as an "assignment operator."

It means: "Make the object named to the left equal to the output of the code to the right."

12.0.4 Explore Data

Click the green arrow code chunk to get the answers below.

How many rows?

```
nrow(MediaBucks)
```

```
[1] 1658
```

How many columns?

```
ncol(MediaBucks)
```

```
[1] 13
```

Dimensions: Gives number rows, then columns

```
dim(MediaBucks)
```

```
[1] 1658 13
```

Names of your columns

```
colnames(MediaBucks)
```

```
[1] "TITLE"                  "COMPANY"  
[3] "Salary"                 "Salary_Details"  
[5] "Salary_Details2"        "Race"  
[7] "Gender"                 "YEARS_EXPERIENCE"  
[9] "LOCATION"                "JOB_DUTIES"  
[11] "PREV_SALARIES_TITLES"   "SALARY_original"  
[13] "Gender_Ethnicity_Original"
```

OR

```
names(MediaBucks)
```

```
[1] "TITLE"                      "COMPANY"
[3] "Salary"                     "Salary_Details"
[5] "Salary_Details2"            "Race"
[7] "Gender"                     "YEARS_EXPERIENCE"
[9] "LOCATION"                   "JOB_DUTIES"
[11] "PREV_SALARIES_TITLES"      "SALARY_original"
[13] "Gender_Ethnicity_Original"
```

Check data types

```
str(MediaBucks)
```

```
'data.frame': 1658 obs. of 13 variables:
 $ TITLE                  : chr "Photojournalist" "Staff Writer" "Staff reporter" "Reporte...
 $ COMPANY                : chr "College Student Newspaper" "Paxton Media" "Schurz Commun...
 $ Salary                  : num 0 12 12 13.2 14.9 ...
 $ Salary_Details          : chr "" "hour" "hour" "hour" ...
 $ Salary_Details2         : chr "" "anhour" "-2012" "~32,000/year" ...
 $ Race                   : chr "asian" "white" "white" "white" ...
 $ Gender                 : chr "male" "female" "female" "female" ...
 $ YEARS_EXPERIENCE        : chr "3" "8" "2" "3" ...
 $ LOCATION                : chr "" "Georgia" "Central Kentucky" "Houston" ...
 $ JOB_DUTIES              : chr "Photo" "Write and cover news stories, photograph news, w...
 $ PREV_SALARIES_TITLES    : chr "" " " " " ...
 $ SALARY_original          : chr "0" "$12 an hour" "$12/hour (2012)" "13.25/hour ~ 32,000/y...
 $ Gender_Ethnicity_Original: chr "Cis male asian" "White female" "cis white female" "Cis wl...
```

Let's look at the first six rows

```
head(MediaBucks)
```

	TITLE	COMPANY	Salary	Salary_Details
1	Photojournalist	College Student Newspaper	0.00	
2	Staff Writer	Paxton Media	12.00	hour
3	Staff reporter	Schurz Communications Inc.	12.00	hour
4	Reporter	Hearst Newspapers	13.25	hour
5	Staff Reporter	Ogden Newspapers	14.91	hour
6	Desk Assistant	KTLA	15.00	hour
	Salary_Details2	Race	YEARS_EXPERIENCE	LOCATION
1		asian	male	3
2	anhour	white	female	8
				Georgia

```

3      -2012    white female          2 Central Kentucky
4 ~32,000/year   white female          3 Houston
5     perhour    white male 3+ years experience      Utah
6     noanswer   female             2 Los Angeles

1
2
3
4
5
6 Organize daily field drives and retrieves file videos\nUse FTP system to send/receive video
PREV_SALARIES_TITLES           SALARY_original Gender_Ethnicity_Original
1                               0          Cis male asian
2                               $12 an hour      White female
3                               $12/hour (2012)  cis white female
4                               13.25/hour ~ 32,000/year  Cis white female
5                               $14.91 per hour    cis white male
6                               $15/hour            cis female

```

Here is a quick way to view the range of your data

```
summary(MediaBucks$Salary)
```

Min.	1st Qu.	Median	Mean	3rd Qu.	Max.	NA's
0	42000	60000	64194	78000	770000	4

Size and scope

```
sum(MediaBucks$Salary, na.rm=TRUE)
```

```
[1] 106177432
```

\$106 million! for 1,658 journalists

Context: NYT earnings in 2020 = \$100 m Facebook profit for one day: \$114 million
(Q42021=\$10.3B)

average

```
mean(MediaBucks$Salary, na.rm=TRUE)
```

```
[1] 64194.34
```

Distribution

```
quantile(MediaBucks$Salary, c(0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9), na.rm=TRUE)
```

10%	20%	30%	40%	50%	60%	70%	80%	90%
28720	39000	45000	52000	60000	65000	75000	84000	101700

```
quantile(MediaBucks$Salary, c(0.25, 0.50, 0.75, 0.9, 0.99), na.rm=TRUE)
```

25%	50%	75%	90%	99%
42000	60000	78000	101700	200000

12.0.5 Navigation Tips

Shortcut Commands

Tab - Autocomplete

In Console Window (lower left)

--Control (or Command) + UP arrow - last lines run

Control (or Command) + Enter - Runs current or selected lines of code in the top left box of

Shift + Control (or Command) +P - Reruns previous region code

12.0.6 Dplyr

dplyr has many tools for data analysis

select Choose which columns to include

filter Filter the data

arrange Sort the data, by size for continuous variables, by date, or alphabetically

group_by Group the data by a categorical variable

Build a simple summary table by Gender

```
MediaBucks %>%
  select(Gender, Salary) %>%
  group_by(Gender) %>%
  summarize(Total = sum(Salary, na.rm=TRUE))
```

```
# A tibble: 4 x 2
  Gender      Total
  <chr>     <dbl>
1 female    63198034.
2 male     35201242.
3 na       3061718.
4 noanswer 4716438.
```

What is the sample size?

```
MediaBucks %>%
  count(Gender) %>%
  arrange(desc(n))
```

```
Gender      n
1 female  1025
2 male   503
3 noanswer 71
4 na     59
```

Better idea: Check Averages!

Build a simple summary table by Gender

```
MediaBucks %>%
  select(Gender, Salary) %>%
  group_by(Gender) %>%
  summarize(Avg_Salary = mean(Salary, na.rm=TRUE))
```

```
# A tibble: 4 x 2
  Gender      Avg_Salary
  <chr>        <dbl>
1 female     61717.
2 male      69983.
3 na        52788.
4 noanswer  68354.
```

Quick filter out hourly workers

```
MediaSalary <- MediaBucks %>%
  filter(Salary >= 1000)
```

Just give me a list of the top 10 salaries and companies: use slice_max. slice_max and slice_min are features in the Dplyr library (part of Tidyverse) that produce quick summary tables. See what else you can do with [the slice commands](#).

```
MediaBucks %>%
  select(COMPANY, Salary) %>%
  slice_max(Salary, n = 10)
```

	COMPANY	Salary
1	Tribune Publishing	770000
2	The Onion	550008
3	G/O Media (The Onion)	500000
4	G/O Media (The Onion)	500000
5	VoxMedia	400000
6	ProPublica	395000
7	Digital_First_Media	375000
8	The Intercept	368249
9	NewYorkTimes	350000
10	LBI Media	291200

Questions:

- 1: View the range of your data
- 2: Number of rows
- 3: Number of rows cut with filter

12.0.7 Find Your News Organization

Filter

```
WSJ <- subset(MediaBucks, COMPANY=="WallStreetJournal")  
  
summary(WSJ$Salary)
```

Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
38	41000	51100	64275	75750	236000

Using Wildcards

```
Journal <- subset(MediaBucks, grepl("?Journal", COMPANY))
```

```
Bloom <- subset(MediaBucks, grepl("?Bloomberg", COMPANY))
```

More Tables

Build a table with several companies of your choice

```
BigBoys <- filter(MediaSalary, COMPANY %in% c("NewYorkTimes", "WallStreetJournal", "Bloomb
```

Table with just reporter salaries

```
Reporters <- subset(MediaBucks, grepl("?reporter", TITLE))
summary(Reporters$Salary)
```

Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
12	35000	50625	57077	67250	230504

Questions:

- 1: Who is making \$230,504 as a reporter???
- 2: Make a table for editors, figure out medians.
- 3: Find highest paid editor. Resent them.
- 4: Make a table for any position involving data

Table with Black reporters at Wall Street Journal

```
WSJ_Black <- MediaBucks %>% filter(Race == "black", COMPANY == "WallStreetJournal")
```

Build a simple summary table by Race

```
Race <- MediaBucks %>%
  select(Race, Salary) %>%
  group_by(Race) %>%
  summarize(Avg_Salary = mean(Salary, na.rm=TRUE)) %>%
  arrange(desc(Avg_Salary))
Race
```

```
# A tibble: 13 x 2
  Race      Avg_Salary
  <chr>     <dbl>
1 chinese    76167.
2 australian  75000
3 african     73485.
4 hispanic    70481.
5 black       69371.
6 poc         68061.
7 asian       66427.
8 noanswer   64925.
9 latina      64806.
10 white      63025.
11 mixed      55756.
12 mideastern 51833.
13 native     50000
```

Wait! What are the totals by race?

```
MediaBucks %>%
  count(Race) %>%
  arrange(desc(n))
```

	Race	n
1	white	1094
2	noanswer	192
3	poc	115
4	asian	77
5	black	57
6	hispanic	53
7	latina	23
8	mixed	22
9	african	13
10	mideastern	7
11	chinese	3
12	australian	1
13	native	1

Advanced: Build a summary table and count by race

```

MediaBucks %>%
  select(Race, Salary) %>%
  group_by(Race) %>%
  summarize(Total=n(),
            Avg = mean(Salary, na.rm=TRUE)) %>%
  arrange(desc(Total))

# A tibble: 13 x 3
  Race      Total     Avg
  <chr>    <int>   <dbl>
1 white      1094 63025.
2 noanswer    192 64925.
3 poc         115 68061.
4 asian        77 66427.
5 black        57 69371.
6 hispanic     53 70481.
7 latina       23 64806.
8 mixed        22 55756.
9 african      13 73485.
10 mideastern   7 51833.
11 chinese      3 76167.
12 australian    1 75000
13 native        1 50000

```

#details: <https://stackoverflow.com/questions/36183601/average-and-count-with-aggregation->

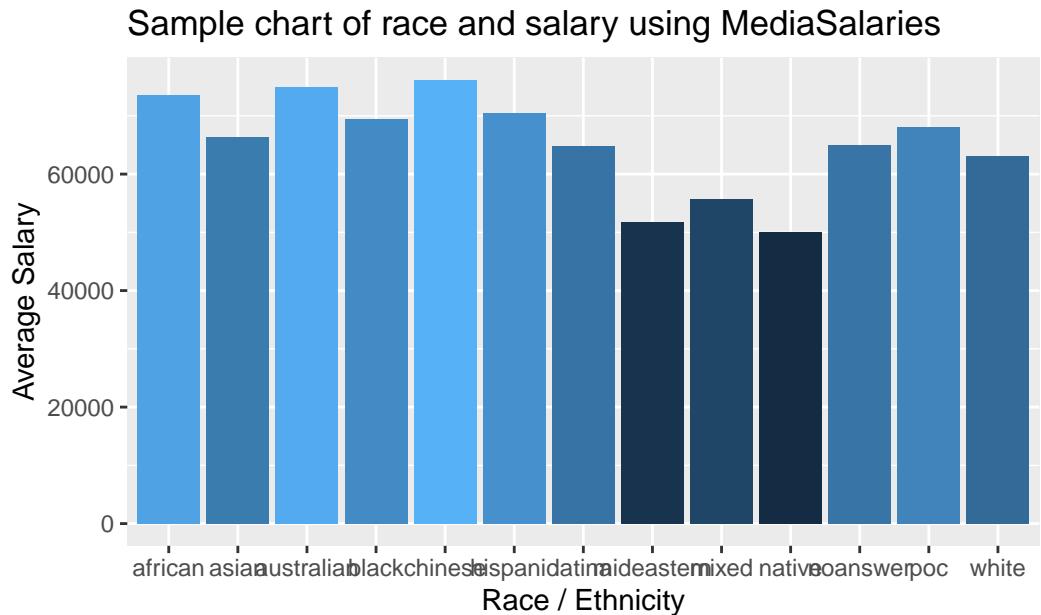
12.0.8 Visualize

Let's make a simple chart of our salaries by race.

```

Race %>%
  ggplot(aes(x = Race, y = Avg_Salary, fill = Avg_Salary)) +
  geom_col(position = "dodge") +
  theme(legend.position = "none") +
  labs(title = "Sample chart of race and salary using MediaSalaries",
       caption = "1658 records, sample data. Graphic by Rob Wells, 8/11/2022",
       y="Average Salary",
       x="Race / Ethnicity")

```



This is a basic chart using ggplot. To break down the code:

- These lines: `Race %>% ggplot(aes(x = Race, y = Avg_Salary, fill = Avg_Salary))`
- + - Uses the Race table, calls the ggplot program, assigns the x axis to Race, y axis to Avg_Salary and fills the color according to Avg_Salary
- These lines: `geom_col(position = "dodge") + theme(legend.position = "none")`
- + - creates a chart of columns, and removes a legend box.
- These lines: `labs(title = Assign the headline and captions.`

12.0.9 What You Have Learned So Far

- How to navigate in R studio
- How to install libraries and packages
- How to import a .xlsx file into R
- How to obtain summary statistics (`summary`)
- How to build basic tables from a dataset
- How to conduct filter queries from a dataset

12.0.10 Questions

1: Build a table for NewYorkTimes employees, and determine median salary of NewYorkTimes employees.

```
#your answer here
```

2: Identify title, gender and race of the highest paid position at NYT

```
#your answer here
```

3: Search for Bloomberg, check median salary, compare to NYT results above.

```
#your answer here
```

12.0.11 Tutorials

Excellent book by Sharon Machlis <https://www.routledge.com/Practical-R-for-Mass-Communication-and-Journalism/Machlis/p/book/9781138726918>

First five chapters are free on her website. My recommendation: buy the book.
<https://www.machlis.com/R4Journalists/>

All Cheat Sheets <https://www.rstudio.com/resources/cheatsheets/>

MaryJo Webster tutorials http://mjwebster.github.io/DataJ/R_tutorials/opiate_deaths.nb.html
https://github.com/mjwebster/R_tutorials/blob/master/Using_R.Rmd

Aldhous' R tutorial <http://paldhous.github.io/NICAR/2018/r-analysis.html>

Ron Campbell Lecture <https://github.com/roncampbell/NICAR2018/blob/master/Intro%20to%20R.md>

Excellent Tutorial Spelling out Excel and Comparable Commands in R <https://trendct.org/2015/06/12/r-for-beginners-how-to-transition-from-excel-to-r/> https://docs.google.com/presentation/d/1O0eFLypJLP-PAC63Ghq2QURAnhFo6Dxc7nGt4y_l90s/edit#slide=id.g1bc441664e_0_59

Andrew Ba Tran first Data Analysis Steps Using R https://docs.google.com/presentation/d/1O0eFLypJLP-PAC63Ghq2QURAnhFo6Dxc7nGt4y_l90s/edit#slide=id.p

Charts https://www.rdocumentation.org/packages/ggplot2/versions/1.0.1/topics/geom_bar
[http://www.cookbook-r.com/Graphs/Bar_and_line_graphs_\(ggplot2\)/](http://www.cookbook-r.com/Graphs/Bar_and_line_graphs_(ggplot2)/)

Base R Cheat Sheet <https://www.povertyactionlab.org/sites/default/files/r-cheat-sheet.pdf>

13 Using GitHub

GitHub is a platform for managing and storing files, data and code built atop Git, a popular open source version control software. GitHub accounts are free and it's [easy to get started](#). The one prerequisite is that you have [Git installed on your local computer](#). There are installers for Mac, Windows and Linux.

13.1 How It Works

Version control is based on the ideas that you want to keep track of changes you make to a collection of files and that multiple people can work together without getting in each other's way or having to do things in a set order. For individual users, it's great for making sure that you always have your work.

GitHub users work in what are known as repositories on their local computers and also *push* changes to a remote repository located on GitHub. That remote repository is key: if you lose your computer, you can fetch a version of your files from GitHub. If you want to work with someone else on the same files, you can each have a local copy, push changes to GitHub and then pull each others' changes back to your local computers.

So, like Microsoft Word's track changes but with a remote backup and multiple editors.

13.2 Getting Started

After installing Git and signing up for a GitHub account, [download and install GitHub Desktop](#). It will have you sign into your GitHub account and then you'll have access to any existing repositories. If you don't have any, that's fine! You can [make one locally](#).

GitHub has [good documentation for working in the Desktop app](#), and while the emphasis in this book will be on using GitHub for version control, it also supports recording issues (read: problems or questions) with your files, contributing to projects that aren't yours and more.

13.3 Video overview

This [YouTube video from Coder Coder](#) provides a nice overview of Git, GitHub and GitHub Desktop.

13.4 Advanced Use

Although our focus is on the GitHub Desktop app, you can use Git and GitHub from your computer's command line interface, and GitHub has a purpose-built [command line client](#), too. GitHub can also serve as a publishing platform for many types of files, and entire websites are hosted on [GitHub Pages](#).

14 Mutuating, Aggregates, Filters

In this chapter, you will learn additional core data skills that allow you to filter, summarize and append new calculations to datasets. The skills in this chapter, once mastered, will transform your ability to analyze data in ways that are difficult in spreadsheets.

14.0.1 R Libraries Background

R is a statistical programming language that is purpose built for data analysis.

There is a basic R program, which geeks call Base R, and it does a lot. We will be bringing in additional software programs known as libraries that do things to make R really operate better and easier.

The two libraries we are going to need for this assignment are `readr` and `dplyr`. The library `readr` reads different types of data in. For this assignment, we're going to read in csv data or Comma Separated Values data. That's data that has a comma between each column of data. Then we're going to use `dplyr` to analyze it.

To use a library, you need to import it. Put all your library steps at the top of your notebooks. Both of these libraries are contained in the [Tidyverse suite of eight essential software packages](#). So just load tidyverse and dplyr and readr are ready to go.

```
library(tidyverse)

-- Attaching packages ----- tidyverse 1.3.2 --
v ggplot2 3.3.6     v purrr   0.3.4
v tibble   3.1.8     v dplyr   1.0.9
v tidyr    1.2.0     v stringr 1.4.0
v readr    2.1.2     vforcats 0.5.1
-- Conflicts ----- tidyverse_conflicts() --
x dplyr::filter() masks stats::filter()
x dplyr::lag()    masks stats::lag()
```

14.1 Importing data

The first thing we need to do is get some data to work with. We do that by reading it in. In our case, we're going to read a datatable from an "CSV" file, a stripped down version of a spreadsheet you might open in a program like Google Sheets, in which each column is separated by a comma.

So step 1 is to import the data. The code to import the data looks like this:

```
baltcity_income<- read_csv("assets/data/baltcity_income_clean.csv") %>%  
as.data.frame()
```

Let's unpack that.

The first part – **baltcity_income** – is the name of a variable.

A **variable** is just a name that we'll use to refer to some more complex thing. In this case, the more complex thing is the data we're importing into R that will be stored as a **dataframe**, which is one way R stores data.

We can call this variable whatever we want. The variable name doesn't matter, technically. We could use any word. Generally we want variable names to be descriptive, hence **baltcity_income**. It's good to keep variable names lower case and one word but two or more words need to be connected by an underscore. You can't start a variable with a number.

The `<-` is the **variable assignment operator**. It's how we know we're assigning something to a word. Think of the arrow as saying "Take everything on the right of this arrow and stuff it into the thing on the left."

read_csv() is a function, one that only works when we've loaded the tidyverse. A **function** is a little bit of computer code that takes in information and follows a series of pre-determined steps and spits it back out. A recipe to make pizza is a kind of function. We might call it **make_pizza()**.

The function does one thing. It takes a preset collection of ingredients – flour, water, oil, cheese, tomato, salt – and passes them through each step outlined in a recipe, in order. Things like: mix flour and water and oil, knead, let it sit, roll it out, put tomato sauce and cheese on it, bake it in an oven, then take it out.

The output of our **make pizza()** function is a finished pie.

We'll make use of a lot of pre-written functions from the tidyverse and other packages, and even write some of our own. Back to this line of code:

```
baltcity_income<- read_csv("assets/data/baltcity_income_clean.csv") %>%  
as.data.frame()
```

Inside of the **read_csv()** function, we've put the name of the file we want to load. Things we put inside of function, to customize what the function does, are called **arguments**. And

lastly, we added a **pipe operator** `%>%` (shift + cntl + M) that adds another command to turn the imported data into a data frame with `as.data.frame()`. The pipe operator - `%>%` - basically tells R to “and then do this.”

Here is the entire command in a code chunk. Run it by clicking the green arrow to the right below.

```
baltcity_income <- read_csv("assets/data/baltcity_income_clean.csv")
```

In this data set, each row represents a Census district, and each column represents a feature of that district: its location, the median household income in 2010, 2016, 2020, the neighborhood identifier and geographic coordinators.

After loading the data, it’s a good idea to get a sense of its shape. What does it look like? There are several ways we can examine it.

By looking in the R Studio environment window, we can see the number of rows (called “obs.”, which is short for observations), and the number of columns (called variables). We can double click on the dataframe name in the environment window, and explore it like a spreadsheet.

There are several useful functions for getting a sense of the dataset right in our markdown document.

If we run `glimpse(baltcity_income)`, it will give us a list of the columns, the data type for each column and and the first few values for each column.

```
glimpse(baltcity_income)
```

```
Rows: 200
Columns: 6
$ Neighborhood <chr> "Canton", "Patterson Park North & East", "Canton", "Canto~
$ x2010      <dbl> 75938, 58409, 77841, 70313, 86157, 60439, 47131, 59236, 2~
$ x2016      <dbl> 100985, 96171, 116875, 97153, 85986, 79063, 64643, 100778~
$ x2020      <dbl> 128839, 130357, 151389, 114946, 98194, 95536, 86125, 1018~
$ Census     <dbl> 101.00, 102.00, 103.00, 104.00, 105.00, 201.00, 202.00, 2~
$ GEOID      <dbl> 24510010100, 24510010200, 24510010300, 24510010400, 24510~
```

If we type `head(baltcity_income)`, it will print out the columns and the first six rows of data.

```
head(baltcity_income)
```

```
# A tibble: 6 x 6
  Neighborhood      x2010   x2016   x2020 Census    GEOID
  <chr>          <dbl>   <dbl>   <dbl>  <dbl>    <dbl>
1 Canton           75938  100985  128839    101 24510010100
2 Patterson Park North & East 58409  96171  130357    102 24510010200
3 Canton           77841  116875  151389    103 24510010300
4 Canton           70313  97153   114946    104 24510010400
5 Fells Point      86157  85986   98194     105 24510010500
6 Fells Point      60439  79063   95536     201 24510020100
```

We can also click on the data name in the R Studio environment window to explore it interactively.

14.1.1 Group by and count

There is some overlap among the Census tracts, which typically are groups of between 1,200 to 8,000 people, and neighborhoods, which can be much larger. Let's figure out how many neighborhoods are represented in this dataset.

dplyr has a group by function that compiles things together and then produces simple summaries by counting things, or averaging them together. It's a good place to start.

The first step of every analysis starts with the data being used. Then we apply functions to the data.

In our case, the pattern that you'll use many, many times is: `data %>% group_by(COLUMN NAME) %>% summarise(VARIABLE NAME = AGGREGATE FUNCTION(COLUMN NAME))`

In our dataset, the column with neighborhood identifier is called “Neighborhood.” Neighborhoods overlap with Census tracts.

Here's the code to count the number of census tracts in each neighborhood:

```
baltcity_income %>%
  group_by(Neighborhood) %>%
  summarise(
    count_tracts = n()
  )
```

```
# A tibble: 56 x 2
  Neighborhood      count_tracts
  <chr>                <int>
1 Allendale/Irvington/S. Hilton        6
```

```

2 Beechfield/Ten Hills/West Hills           3
3 Belair-Edison                            4
4 Brooklyn/Curtis Bay/Hawkins Point        4
5 Canton                                    3
6 Cedonia/Frankford                        5
7 Cherry Hill                             3
8 Chinquapin Park/Belvedere                2
9 Claremont/Armistead                      4
10 Clifton-Berea                           5
# ... with 46 more rows
# i Use `print(n = ...)` to see more rows

```

So let's walk through that.

We start with our dataset – `baltcity_incomes` – and then we tell it to group the data by a given field in the data. In this case, we wanted to group together all the counties, signified by the field name `Neighborhood`, which you could get from using the `glimpse()` function. After we group the data, we need to count them up.

In `dplyr`, we use the `summarize()` function, [which can do alot more than just count things](#).

Inside the parentheses in `summarize`, we set up the summaries we want. In this case, we just want a count of the number of loans for each county grouping. The line of code `count_tracts = n()`, says create a new field, called `count_tracts` and set it equal to `n()`. `n()` is a function that counts the number of rows or records in each group. Why the letter `n`? The letter `n` is a common symbol used to denote a count of something.

When we run that, we get a list of counties with a count next to them. But it's not in any order.

So we'll add another “and then do this” symbol – `%>%` – and use a new function called `arrange()`. `Arrange` does what you think it does – it arranges data in order. By default, it's in ascending order – smallest to largest. But if we want to know the county with the most loans, we need to sort it in descending order. That looks like this:

```

baltcity_income %>%
  group_by(Neighborhood) %>%
  summarise(
    count_tracts = n()
  ) %>%
  arrange(desc(count_tracts))

# A tibble: 56 x 2
  Neighborhood      count_tracts
  <fct>                  <dbl>
1 Beechfield/Ten Hills/West Hills     3
2 Belair-Edison                     4
3 Brooklyn/Curtis Bay/Hawkins Point 4
4 Canton                            3
5 Cedonia/Frankford                 5
6 Cherry Hill                       3
7 Chinquapin Park/Belvedere        2
8 Claremont/Armistead               4
9 Clifton-Berea                     5
10 ...
# ... with 46 more rows
# i Use `print(n = ...)` to see more rows

```

```

<chr>
1 Southwest Baltimore
2 Allendale/Irvington/S. Hilton
3 Medfield/Hampden/Woodberry/Remington
4 Sandtown-Winchester/Harlem Park
5 Cedonia/Frankford
6 Clifton-Berea
7 Greater Charles Village/Barclay
8 Greater Rosemont
9 Harford/Echodale
10 Inner Harbor/Federal Hill
# ... with 46 more rows
# i Use `print(n = ...)` to see more rows

```

The Census data contains a column detailing the neighborhood. It has associated the Census tracts to neighborhood names. This dataset may have several neighborhood values since Census tracts are a smaller unit of measurement.

Southwest Baltimore neighborhood is spread out over 8 census tracts, more than any other neighborhood.

Here's the code to determine the count the number of census tracts in each neighborhood:

```

baltcity_income %>%
  summarise(
    count_tracts = n()
  )

# A tibble: 1 x 1
  count_tracts
  <int>
1        200

```

14.2 Interviewing Your Data: Min, Max, Mean, Medians

What is the typical median income? What about the highest and lowest median incomes in the city? For that, we can use the `min()` and `max()` functions.

```

baltcity_income %>%
  select(Neighborhood, x2010, x2016, x2020, Census) %>%
  summarise(

```

```

    count_tracts = n(),
    x2020_median = median(x2020, na.rm=TRUE),
    min_2020 = min(x2020, na.rm=TRUE),
    max_2020 = max(x2020, na.rm=TRUE)
  )
}

# A tibble: 1 x 4
count_tracts x2020_median min_2020 max_2020
<int>          <dbl>      <dbl>     <dbl>
1            200        49875     13559    199531

```

Here we see the typical median household income is \$49,875 by census tract for Baltimore City in 2020 (see result for x2020 median). The lowest median income was \$13,559 and the highest was \$199,531. From another Census analysis, we know hat citywide, the median household income was \$52,164 for 2016-2020. The `na.rm=TRUE` argument lets R knock out any empty rows from the calculation.

Here's another quick way to determine the distribution of a particular column of data

```
summary(baltcity_income$x2020)
```

Min.	1st Qu.	Median	Mean	3rd Qu.	Max.	NA's
13559	35702	49875	56311	64372	199531	5

This tells us the minimum, maximum, median, average (mean), and the first and third quartile, as well as rows with no values.

14.2.1 Filters: Extracting Needles from Haystacks

Where are these rich and poor places? Let's filter for the lowest value, \$13,559, and find out where it is

```
baltcity_income %>%
  #temp code - remove later
  as.data.frame() %>%
  select(Neighborhood, x2020) %>%
  filter(x2020 ==13559)
```

```

Neighborhood x2020
1 Upton/Druid Heights 13559

```

It is part of the Upton/Druid Heights neighborhood in West Baltimore.

We can stack filters using the Or connector: | It's above the enter key on a Mac keyboard

```
baltcity_income %>%
  select(Neighborhood, x2020) %>%
  filter((x2020 ==13559) | (x2020==199531))
```

```
# A tibble: 2 x 2
  Neighborhood           x2020
  <chr>                  <dbl>
1 Upton/Druid Heights    13559
2 North Baltimore/Guilford/Homeland 199531
```

Now we know in one report the wealthiest neighborhood, North Baltimore/Guilford/Homeland, and the poorest, Upton/Druid Heights.

Read this for [more details about logical operators](#).

Let's filter for the wealthy neighborhoods, all above \$100,000

```
baltcity_income %>%
  select(Neighborhood, x2020) %>%
  filter(x2020 > 100000) %>%
  arrange(desc(x2020))

# A tibble: 17 x 2
  Neighborhood           x2020
  <chr>                  <dbl>
1 North Baltimore/Guilford/Homeland 199531
2 North Baltimore/Guilford/Homeland 195353
3 Greater Roland Park/Poplar Hill  155605
4 South Baltimore            151659
5 Canton                   151389
6 Greater Roland Park/Poplar Hill 151146
7 Inner Harbor/Federal Hill   133333
8 Highlandtown              130769
9 Patterson Park North & East  130357
10 Canton                   128839
11 South Baltimore            121685
12 Inner Harbor/Federal Hill 120729
13 Canton                   114946
```

14 Mount Washington/ColdSpring	109688
15 Highlandtown	107438
16 Inner Harbor/Federal Hill	107344
17 Fells Point	101815

Using the summarise function, we can figure out an average value on a column. In this case, we're going to average all of the median income values by census tract.

```
baltcity_income %>%
  select(Neighborhood, x2020, Census) %>%
  summarise(
    count_tracts = n(),
    x2020_avg = mean(x2020, na.rm=TRUE))

# A tibble: 1 x 2
  count_tracts x2020_avg
  <int>      <dbl>
1          200     56311.
```

In the example above, we created a new summary value called x2020_avg that holds the result of the math, the average of the entire x2020 column of median incomes.

14.2.2 Other summarization methods: mean, median, min and max

Here's another trick, pulling out the minimum and maximum values

```
baltcity_income %>%
  select(Neighborhood, x2020, Census) %>%
  summarise(
    count_tracts = n(),
    min_2020 = min(x2020, na.rm=TRUE),
    max_2020 = max(x2020, na.rm=TRUE))

# A tibble: 1 x 3
  count_tracts min_2020 max_2020
  <int>      <dbl>     <dbl>
1          200     13559     199531
```

To kick it up a notch, here's the same idea but with averages and medians for the three years in our data: 2010, 2016, 2020.

```

baltcity_income %>%
  select(Neighborhood, x2010, x2016, x2020, Census) %>%
  summarise(
    count_tracts = n(),
    x2020_median = median(x2020, na.rm=TRUE),
    x2020_avg = mean(x2020, na.rm=TRUE),
    x2016_median = median(x2016, na.rm=TRUE),
    x2016_avg = mean(x2016, na.rm=TRUE),
    x2010_median = median(x2010, na.rm=TRUE),
    x2010_avg = mean(x2010, na.rm=TRUE))

# A tibble: 1 x 7
# ... with abbreviated variable names 1: x2010_median, 2: x2010_avg
  count_tracts x2020_median x2020_avg x2016_median x2016_avg x2010_med~1 x2010~2
  <int>          <dbl>      <dbl>          <dbl>      <dbl>          <dbl>      <dbl>
1        200       49875     56311.       39583     46744.       37080.   41923.
# ... with abbreviated variable names 1: x2010_median, 2: x2010_avg

```

Use the right diamond at `x2016_median` to see columns 5-7.

14.2.3 Using sum

There's much more we can do to summarize each group. Let's pull in another dataset and summarize by group.

```
#loading 2020 and 2010 Baltimore City population by race
baltcity_race <- read_csv("assets/data/baltcity_race_8_13.csv") %>%
  as.data.frame()
```

Let's say we wanted to know the total population by white people in Baltimore? For that, we could use the `sum()` function to add up all of the population in the column "x2020_white". We put the column we want to total – "amount" – inside the `sum()` function `sum(amount)`. Note that we can simply add a new summarize function here, keeping our `count_loans` field in our output table.

This abbreviated slice of Census data contains columns detailing the population by race in Census tracts. There is the `x2020_total` which provides the full population, then `x2020_white`, `x2020_black`, `x2020_hispanic`. We omitted Asians and Pacific islanders and people identifying with more than one race for simplicity in this example.

Here we can select a race variable and summarize it.

```

baltcity_race %>%
  select(x2020_white, x2020_black) %>%
  summarize(
    white_total = sum(x2020_white, na.rm = TRUE),
    black_total = sum(x2020_black, na.rm = TRUE)
  )

white_total black_total
1      178996      375002

```

Pre-Lab Question: We know the median income for Baltimore City (I just told you a few paragraphs ago). Construct a filter for all census tracts below the citywide median household income for 2020. Count them. What percentage of the city's census tracts are below the median? Put that in code too. Draft a tweet with your findings.

Answer this question in English: Write in Elms

```
`<!-- quarto-file-metadata: eyJyZXNvdXJjZURpcii6Ii4ifQ== -->`{=html}
```

```
```{=html}
```

```
<!-- quarto-file-metadata: eyJyZXNvdXJjZURpcii6Ii4iLCJib29rSXRlbVR5cGUI0iJjaGFwdGVyIiwiYm9va
```

# 15 Visualizing your data for reporting

## 15.0.1 Why to use visualization

Before you launch into trying to chart or map your data, take a minute to think about the many roles that static and interactive graphic elements play in your journalism.

Sometimes, you can help yourself and your story by just creating a quick chart, which helps you see patterns in the data that wouldn't otherwise surface. In the reporting phase, visualizations can:

- Help you identify themes and questions for the rest of your reporting
- Identify outliers – good stories, or perhaps errors, in your data
- Help you find typical examples
- Show holes in your reporting

Visualizations also play multiple roles in publishing:

- Illustrate a point made in a story in a more compelling way
- Remove unnecessarily technical information from prose
- Allow exploration, provide transparency about your reporting to readers

Visualizing data is becoming a much greater part of journalism. Large news organizations are creating graphics desks that create complex visuals with data to inform the public about important events. To do it well is a course on its own. And not every story needs a feat of programming and art.

Good news: one of the best libraries for visualizing data is in the tidyverse and it's pretty simple to make simple charts quickly with just a little bit of code. It's called [ggplot2](#).

Let's revisit some data we've used in the past and turn it into charts. First, let's load libraries. When we load the tidyverse, we get ggplot2.

```
library(tidyverse)
```

```
-- Attaching packages ----- tidyverse 1.3.2 --
v ggplot2 3.3.6 v purrr 0.3.4
v tibble 3.1.8 v dplyr 1.0.9
```

```

v tidyverse 1.2.0 v stringr 1.4.0
v readr 2.1.2 v forcats 0.5.1
-- Conflicts ----- tidyverse_conflicts() --
x dplyr::filter() masks stats::filter()
x dplyr::lag() masks stats::lag()

```

## 15.0.2 311 Call Data

The dataset we'll use involves the 311 calls for service to the San Francisco Police Department. The 311 calls are for general non-emergency issues ranging from blocked sidewalks to graffiti to homeless concerns. Details [are here](#) If you're curious, [here is the original dataset: 3,048,797 records](#)

This tutorial uses a subset of this data <sup>1</sup>

```
SF <- read_csv("assets/data/SF_311.csv")
```

Clean column names, Process dates

The imported data recognized the date as a character text, which means you cannot run an analysis to group by years, months or days. The code below did four things - 1: `janitor::clean_names` normalized the column names - 2: `as.Date(SF$call_date)` converted text into date fields - 3: `lubridate::mdy(SF$call_date)` converted the date into a format to extract the dates in detail using lubridate, an important software library for date analysis - 4: `lubridate::year(SF$call_date2)` extracted the year into a new column.

```

#This cleans column names
SF <- janitor::clean_names(SF)
#This processes dates for analysis
SF$call_date2 <- as.Date(SF$call_date)
SF$call_date2 <- lubridate::mdy(SF$call_date)
#This creates a new column for year
SF$year <- lubridate::year(SF$call_date2)

```

Check our data. Note that `call_date2` is a `<date>` field.

```
glimpse(SF)
```

Rows: 157,237

---

<sup>1</sup>The Calls for Service were filtered as follows: CONTAINS homeless, 915, 919, 920: Downloaded 157,237 records 3/31/16 to 11/30/2019. This is 5.1% of all calls in the broader database. File renamed to: SF\_311.xlsx

```
Columns: 16
$ crime_id <dbl> 190040497, 190200644, 190213959, 190271011, 1-
$ original_crime_type_name <chr> "911", "Homeless Complaint", "Homeless Compla-
$ report_date <chr> "1/4/19", "1/20/19", "1/21/19", "1/27/19", "1-
$ call_date <chr> "1/4/19", "1/20/19", "1/21/19", "1/27/19", "1-
$ offense_date <chr> "1/4/19", "1/20/19", "1/21/19", "1/27/19", "1-
$ call_time <time> 06:58:00, 06:19:00, 22:42:00, 09:28:00, 13:2-
$ call_date_time <chr> "1/4/19 6:58", "1/20/19 6:19", "1/21/19 22:42-
$ disposition <chr> "HAN", "HAN", "ADV", "HAN", "HAN", "HA-
$ address <chr> "400 Block Of Jones St", "8th And Market", "M-
$ city <chr> "San Francisco", NA, "San Francisco", "San Fr-
$ state <chr> "CA", "CA", "CA", "CA", "CA", "CA", "CA-
$ agency_id <dbl> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, ~
$ address_type <chr> "Premise Address", "Geo-Override", "Intersect-
$ common_location <chr> NA, NA, NA, NA, NA, NA, "Jose Coronado Plgd, ~
$ call_date2 <date> 2019-01-04, 2019-01-20, 2019-01-21, 2019-01-~~
$ year <dbl> 2019, 2019, 2019, 2019, 2019, 2019, 201-
```

### 15.0.3 Bar charts

The first kind of chart we'll create is a simple bar chart.

It's a chart designed to show differences between things – the magnitude of one thing, compared to the next thing, and the next, and the next.

So if we have thing, like a county, or a state, or a group name, and then a count of that group, we can make a bar chart.

So what does the chart the years of the 311 call data look like?

First, we'll do a basic count of the years. There are just four years in the dataset, from 2016-2019. We built a simple table called `years`.

```
years <- SF %>%
 count(year) %>%
 group_by(year)

years

A tibble: 4 x 2
Groups: year [4]
 year n
 <dbl> <int>
```

```
1 2016 37802
2 2017 49523
3 2018 36057
4 2019 33855
```

Now let's create a bar chart using ggplot.

We need to tell ggplot what kind of chart to make.

In ggplot, we work with two key concepts called geometries (abbreviated frequently as **geom**) and aesthetics (abbreviated as **aes**).

Geometries are the shape of the data: line charts, bar charts, scatterplots, histograms, pie charts, etc.

Aesthetics help ggplot know what component of our data to visualize – why we'll visualize values from one column instead of another.

In a bar chart, we first pass in the data to the geometry, then set the aesthetic.

In the codeblock below, we've added a new function, `geom_bar()`.

Using `geom_bar()` – as opposed to `geom_line()` – says we're making a bar chart.

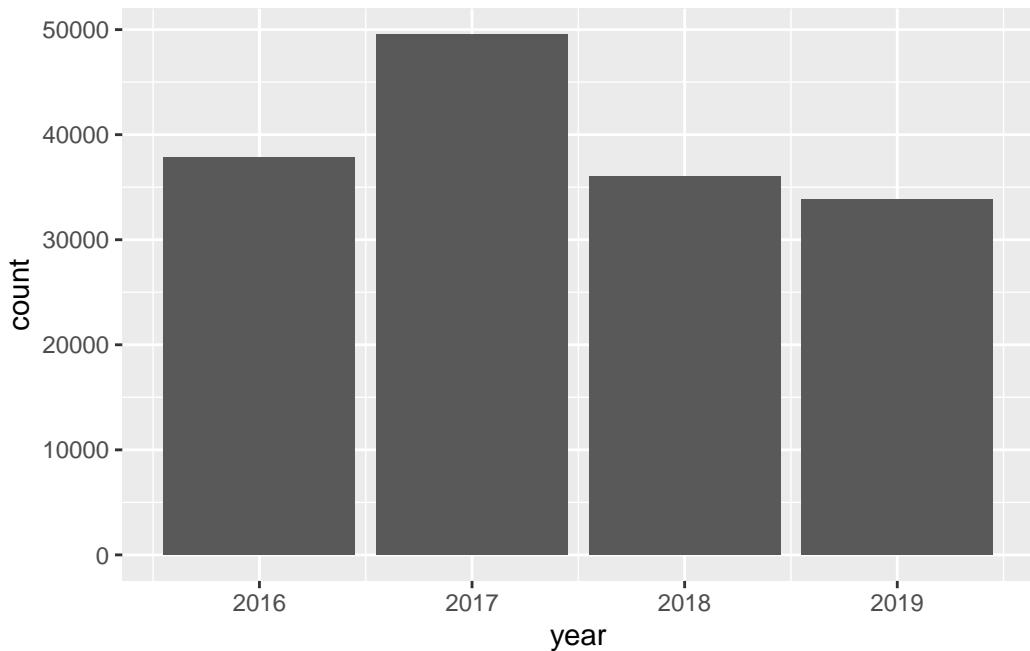
Inside of that function, the aesthetic, `aes`, says which columns to use in drawing the chart.

We're setting the values on the x axis (horizontal) to be the **year**. We set weight to **n** or the total calls per year, and it uses that value to "weight" or set the height of each bar.

One quirk here with ggplot.

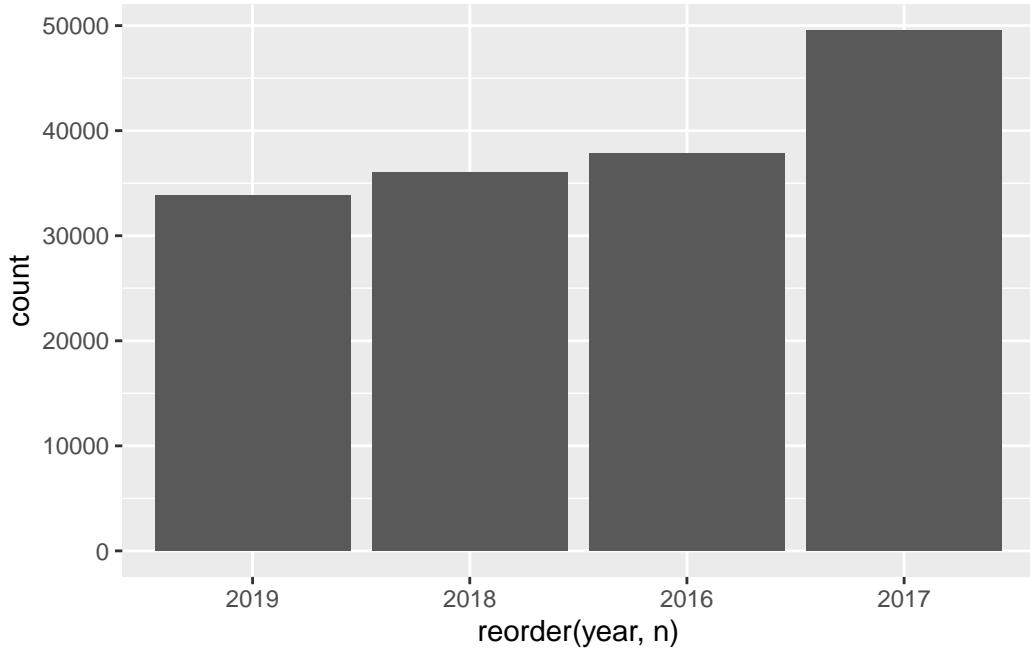
After we've invoked the `ggplot()` function, you'll notice we're using a `+` symbol. It means the same thing as `%>%` – "and then do this". It's just a quirk of `ggplot()` that after you invoke the `ggplot()` function, you use `+` instead of `%>%`. It makes no sense to me either, just something to live with.

```
years %>%
 ggplot() +
 geom_bar(aes(x=year, weight=n))
```



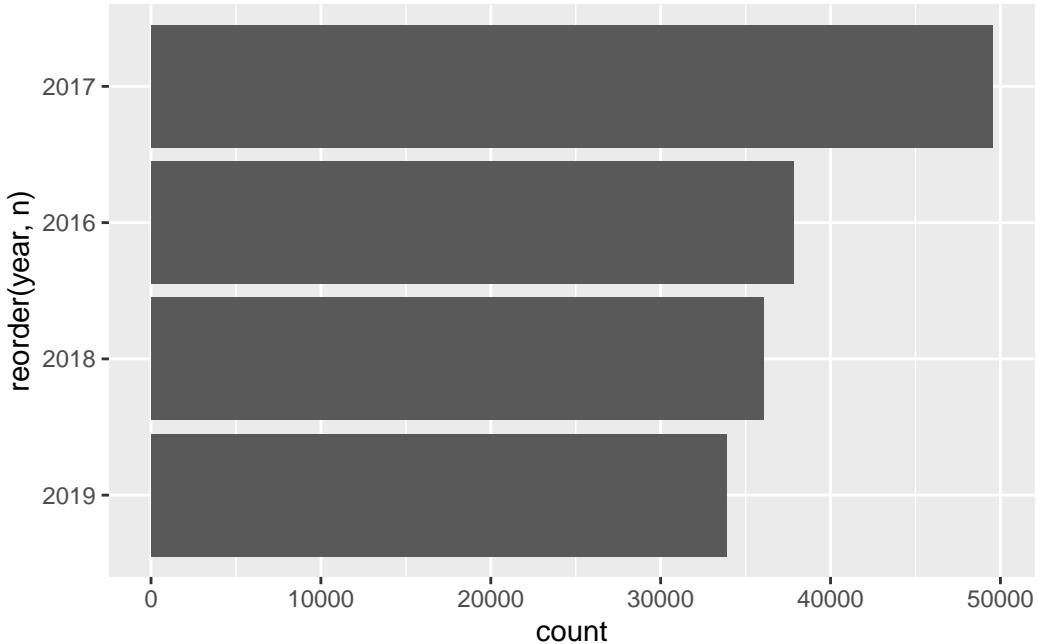
This is a very basic chart. But it's hard to derive much meaning from this chart. We can order the bars from highest to lowest by count of calls per year. We can fix that by using the `reorder()` function to do just that:

```
years %>%
 ggplot() +
 geom_bar(aes(x=reorder(year, n), weight=n))
```



This gives us another view of the data. But the bottom labels can be flipped if we wanted to avoid overlapping names. We can fix that by flipping it from a vertical bar chart (also called a column chart) to a horizontal one. `coord_flip()` does that for you.

```
years %>%
 ggplot() +
 geom_bar(aes(x=reorder(year, n), weight=n)) +
 coord_flip()
```



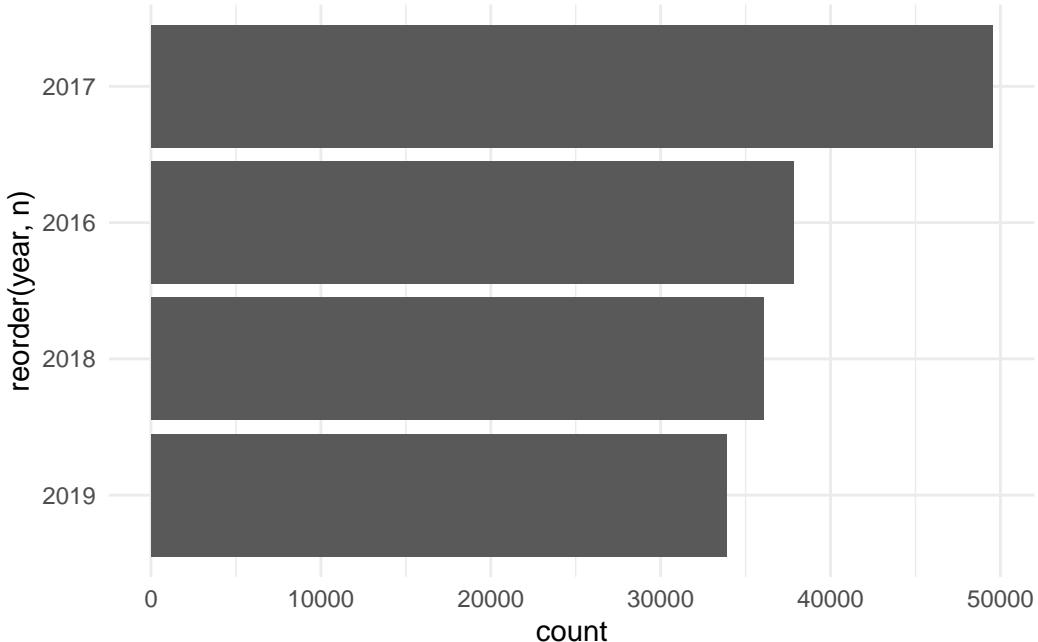
Notice how this clearly shows a trend about the decline in 311 calls per year.

We're mainly going to use these charts to help us in reporting, so style isn't that important.

We can pretty up these charts for publication with some more code. To style the chart, we can change or even modify the "theme", a kind of skin that makes the chart look better.

Here I'm changing the theme slightly to remove the gray background with one of ggplot's built in themes, `theme_minimal()`

```
years %>%
 ggplot() +
 geom_bar(aes(x=reorder(year, n), weight=n)) +
 coord_flip() +
 theme_minimal()
```



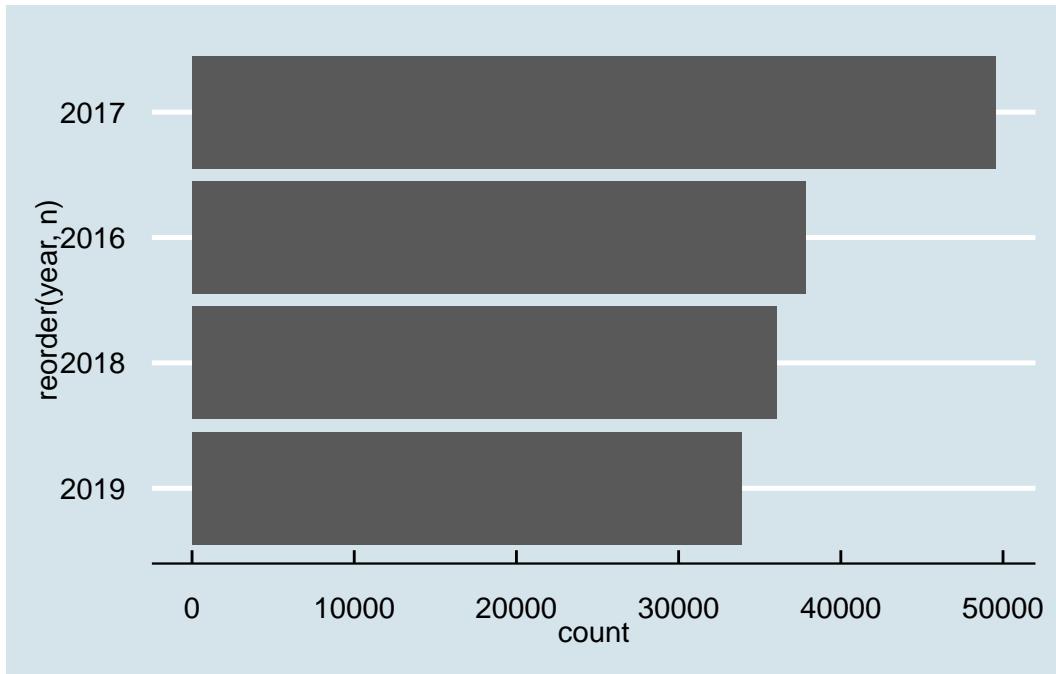
The ggplot universe is pretty big, and lots of people have made and released cool themes for you to use. Want to make your graphics look kind of like [The Economist's](#) graphics? There's a theme for that.

First, you have to install and load a package that contains lots of extra themes, called [ggthemes](#).

```
#install.packages('ggthemes')
library(ggthemes)
```

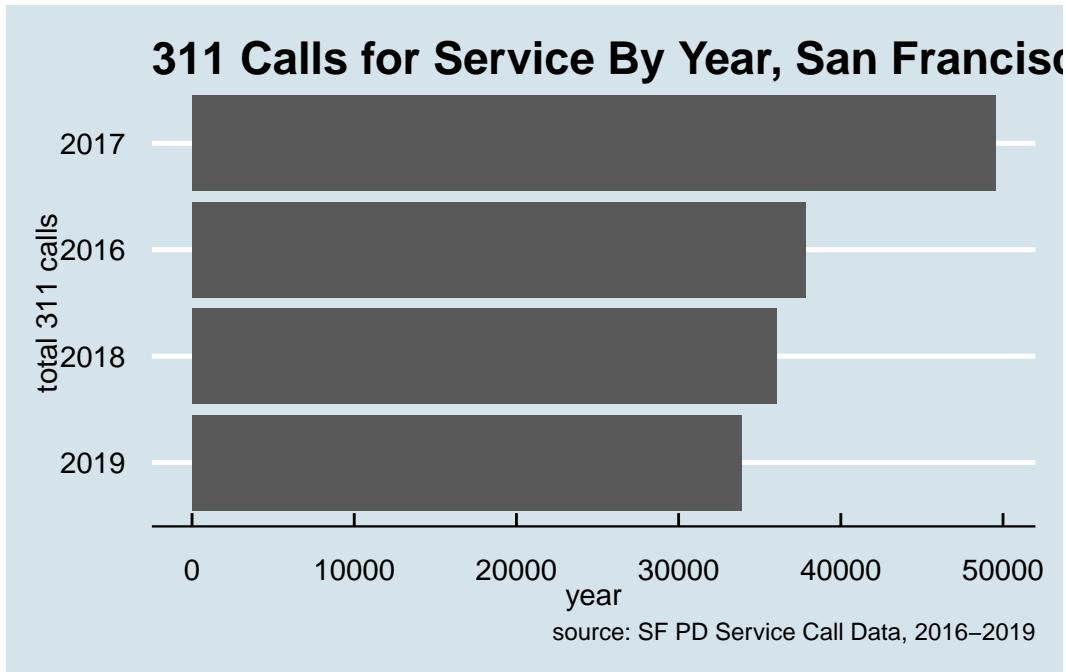
And now we'll apply the economist theme from that package with `theme_economist()`

```
years %>%
 ggplot() +
 geom_bar(aes(x=reorder(year, n), weight=n)) +
 coord_flip() +
 theme_minimal() +
 theme_economist()
```



Those axis titles are kind of a mess. Let's change "count" on the x axis to "total 311 calls" and change "reorder(year, n)" to "year". And while we're at it, let's add a basic title, "311 Calls for Service By Year, San Francisco" and a source as a caption. We'll use a new function, `labs()`, which is short for labels.

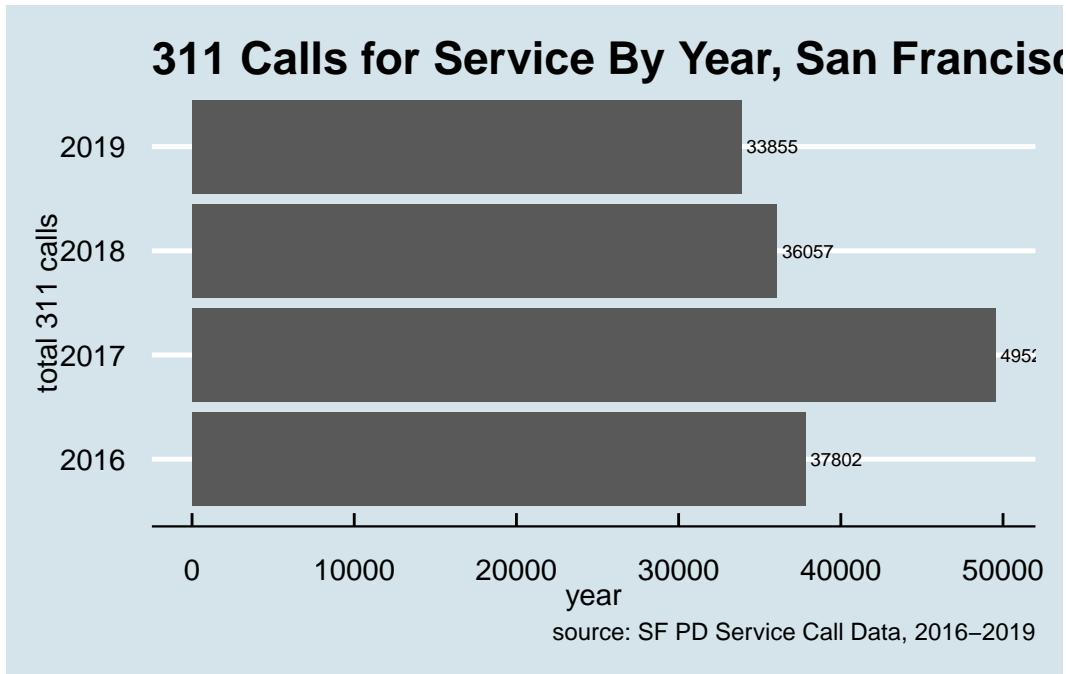
```
years %>%
 ggplot() +
 geom_bar(aes(x=reorder(year, n), weight=n)) +
 coord_flip() +
 theme_minimal() +
 theme_economist() +
 labs(
 title="311 Calls for Service By Year, San Francisco",
 x = "total 311 calls",
 y = "year",
 caption = "source: SF PD Service Call Data, 2016-2019")
```



Not super pretty, but good enough to show an editor to help them understand the conclusions you reached with your analysis.

With `geom_text`, we can add numbers to the bars

```
years %>%
 ggplot(aes(x=year, y=n, weight=n)) +
 coord_flip() +
 theme_economist() +
 geom_col()+
 geom_text(aes(label=n), hjust = -.1, size = 2.5) +
 labs(
 title="311 Calls for Service By Year, San Francisco",
 x = "total 311 calls",
 y = "year",
 caption = "source: SF PD Service Call Data, 2016–2019")
```

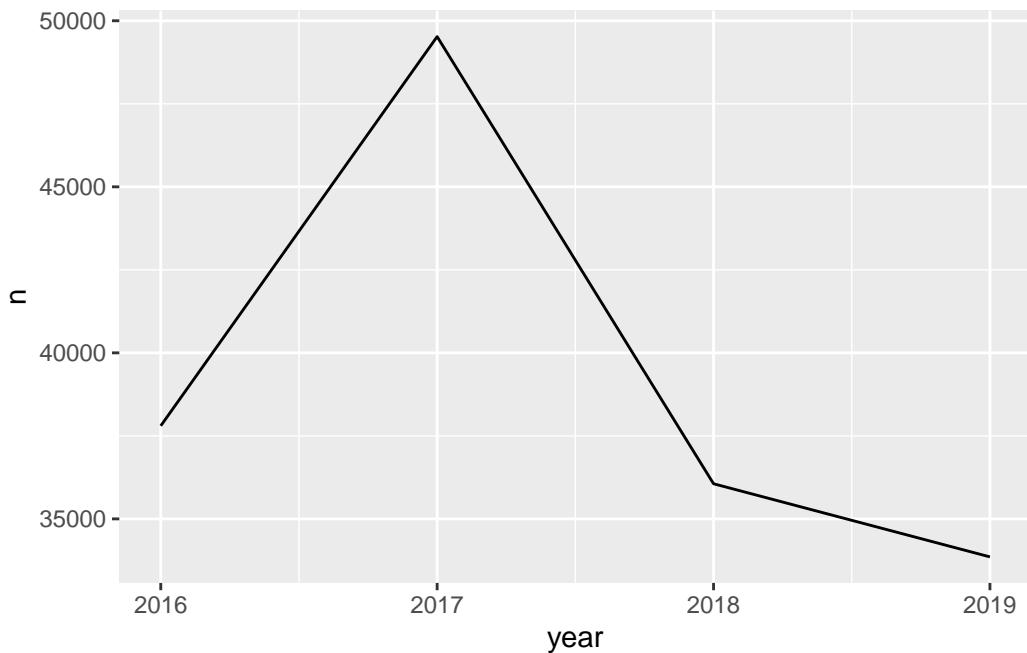


## 15.1 Line charts

Let's look at how to make another common chart type that will help you understand patterns in your data.

Line charts can show change over time. It works much the same as a bar chart, code wise, but instead of a weight, it uses a y. We'll put the year on the x axis and count of loans, or n, on the y axis.

```
years %>%
 ggplot() +
 geom_line(aes(x=year, y=n))
```



It's not super pretty, but there's an obvious pattern, a bog peak in 2017.

Let's build another table with some detail by day.

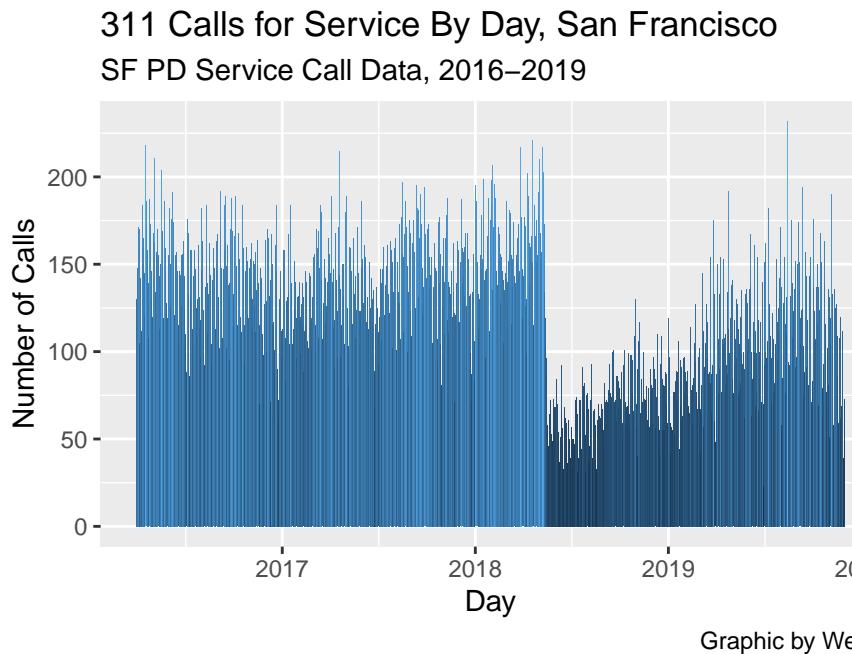
```
days <- SF %>%
 count(call_date2)
days
```

```
A tibble: 1,340 x 2
 call_date2 n
 <date> <int>
1 2016-03-31 10
2 2016-04-01 130
3 2016-04-02 119
4 2016-04-03 148
5 2016-04-04 171
6 2016-04-05 170
7 2016-04-06 170
8 2016-04-07 105
9 2016-04-08 142
10 2016-04-09 95
... with 1,330 more rows
```

```
i Use `print(n = ...)` to see more rows
```

Here is the chart by day

```
days %>%
 ggplot(aes(x = call_date2, y = n, fill = n)) +
 geom_bar(stat = "identity") +
 labs(title = "311 Calls for Service By Day, San Francisco",
 subtitle = "SF PD Service Call Data, 2016–2019",
 caption = "Graphic by Wells",
 y="Number of Calls",
 x="Day")
```



By charting this, we can quickly see a pattern that can help guide our reporting: why was there such a huge drop in mid-2018?

Let's build a line chart for just a few variables based on year and disposition: Citations of people, Cancelled Calls and Admonished people on the scene.

```
dispo <- SF %>%
 filter(disposition == c("CAN", "CIT", "ADM"))
```

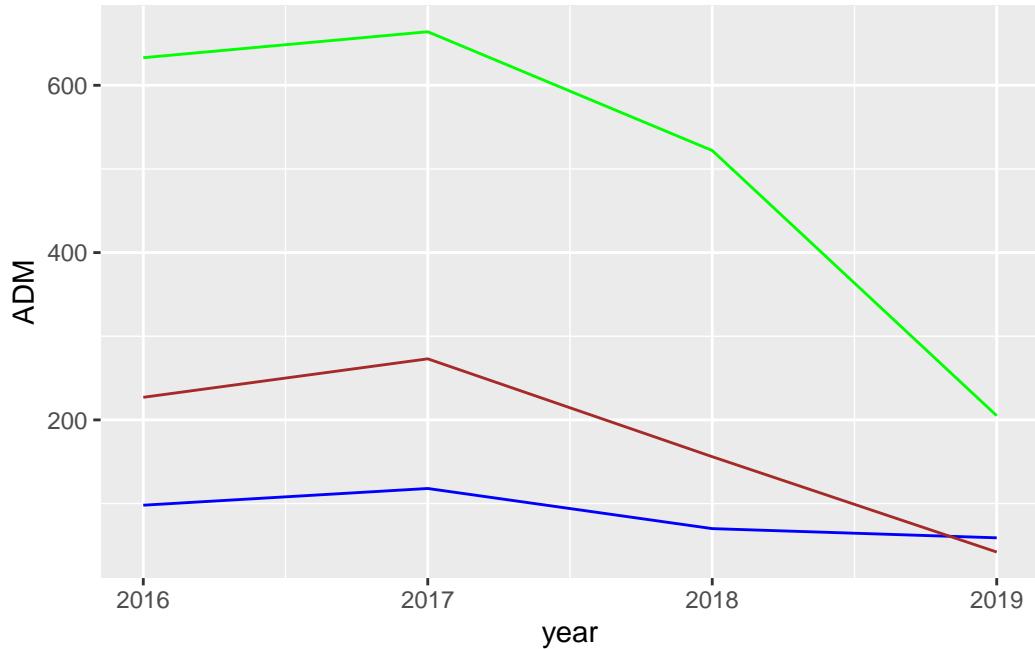
```
Warning in disposition == c("CAN", "CIT", "ADM"): longer object length is not a
multiple of shorter object length
```

```
dispo <- dispo %>%
 select(year, disposition) %>%
 group_by(year) %>%
 count(disposition)

dispo_pivot <- dispo %>%
 pivot_wider(names_from = "disposition", values_from = "n") %>%
 as.data.frame()
```

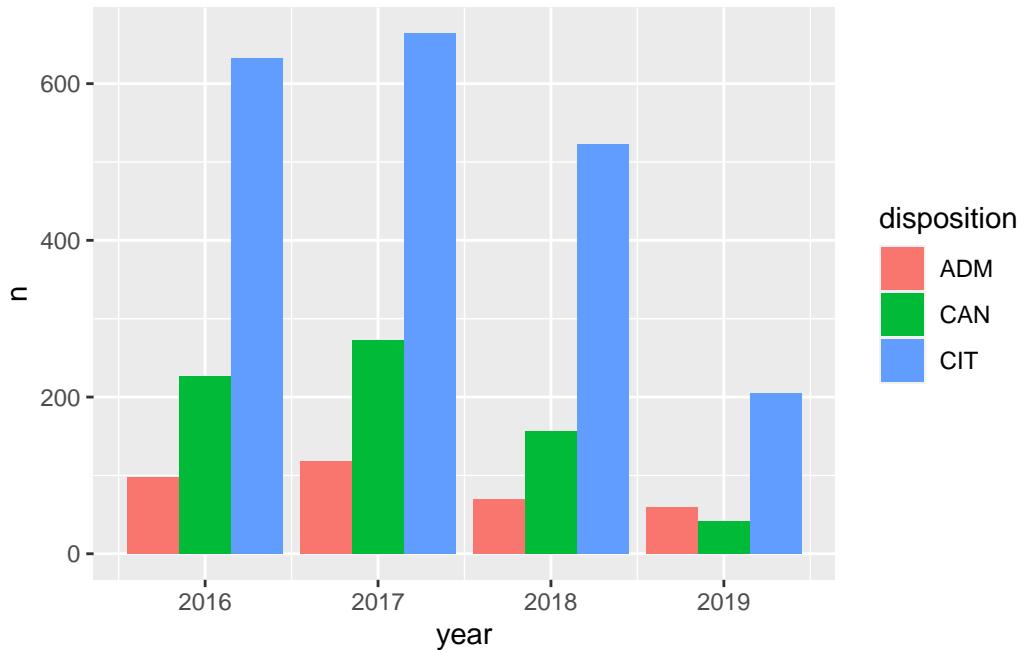
With this dataframe, we will now use separate lines per column to graph the results

```
ggplot(dispo_pivot, aes(x=year)) +
 geom_line(aes(y=ADM), color = "blue") +
 geom_line(aes(y=CAN), color = "brown") +
 geom_line(aes(y=CIT), color = "green")
```



Or you can plot them with side-by-side columns. Use dispo and plot the columns by type of disposition. Each individual variable in disposition is assigned its own column and color. The years are plotted on the x axis.

```
ggplot(dispo, aes(year, n, fill = disposition)) +
 geom_col(position = "dodge")
```



We're just scratching the surface of what ggplot can do, and chart types. There's so much more you can do, so many other chart types you can make. But the basics we've shown here will get you started.

## **(APPENDIX) Appendix**

# Newsroom numbers cheat sheet

(Selected and edited from drafts of “Numbers in the Newsroom”, 2nd edition, 2014, by Sarah Cohen. Full book available from [IRE](#))

Virtually every number we use must be compared with something – another time, another place or a total. This newsroom math guide helps you put numbers into perspective using three devices: “Per” something, change, and averages.

These are arithmetic problems – formulas many of us learned sometime around the fourth grade. Unfortunately, we had another decade to forget them before we got into reporting – then got caught in the trap of math phobia – before these tools became second nature.

The good news is that most news stories don’t depend on fancy math. Master this arithmetic again, and you’ll be able to tackle most reporting jobs.

NOTE TO READERS ON MOBILE DEVICES: This is going to look incomprehensible. Don’t even try until you get to a bigger screen.

## The PERS: Fractions, rates, percents and per capita

You can usually simplify your story if you can re-jigger your numbers into a rate, a ratio or a percentage. “One out of four” is a fraction, or a rate. “Forty percent” is another ratio or rate. And 235 deaths per 100,000 people is another.

Percents and fractions are used to scale off very large or very small numbers while putting them into perspective.

Rates are also used to level the playing field – they compare two items that have a different base.

When you see a lot of numbers in copy, examine them to see if a simple rate – “one of four” or 25 percent – would simplify your story.

## Fractions and percents

Repeat this: “Percents are fractions. Fractions are percents.” Remembering this all the time will keep you focused on the key element of percentages: They’re ratios, or rates, expressed as a fraction of 100.

### Figuring a percent:

Step 1: Know your base. Think of the words “out of.” It’s the total of all the groups.

Step 2: Divide the category you care about by the base.

Remember that a fraction sign (/) means “divided by” (÷).

Step 3: Move the decimal point two places to the right (or multiply by 100) to get the rate per hundred, or percent.

Step 4: Round the answer to no more than one decimal place. Better yet, look for an easier fraction your readers will understand.

---

### Formula

Step 1: Total = The base

Step 2: (Category / Total) = Proportion

Step 3: Proportion x 100 = Percent

Step 4: Round and simplify.

---

### Example

If 58 people say they will vote in an upcoming election and 92 say they won’t, this is how to compute the percent of people who claim they will vote:

Step 1: Base = number of people asked =  $92 + 58 = 150$

Step 2: Rate = 58 out of 150 =  $58/150 = .386666..$

Step 3: Percent =  $.38666... \times 100 = 38.666666....$

Step 4: Round and simplify: = nearly 40 percent

From fractions to percents and back

You probably know that 1 out of 4 is one-quarter, and that it's also 25 percent. But you may not know how to get from one to another.

*From fractions to percents:*

$1/4 = 1 \div 4 = 0.25$ . Move the decimal place over two places, or multiply by 100, to get 25%

*From percents to fractions:*

1. Write your percent as a fraction:  $25/100$
2. Try to find a “least common denominator:” 25 in this case goes into both the top and the bottom. You might want to round off either number to come out to a simple denominator.
3. Simplify:  $(25 / 25) / (100 / 25) = 1 / 4$

*To get “One out of “ numbers:*

1. Express your percentage as a proportion by dividing by 100, so 25% is 0.25.
2. Now divide one by that number:  $1 / .25 = 4$ , so your answer is one-fourth.

Tip for spreadsheet users: Excel allows you to format a number as a fraction or a percent. Play around with formats to see how the number is most easily described.

## Rates and per capita

As with percentages, per person or per capita rates are used to level the playing field.

They're often used when you need to compare two dissimilar places or events: Crimes in cities with different populations, deaths from various diseases or Gross Domestic Product across countries.

Rates also are often used with very big or very small numbers to change them into something we can understand.

Sometimes, though, a rate makes things more complicated, especially when events are rare and there is a consensus that they shouldn't ever happen. Some examples include the 32 crashes attributed to GM's faulty ignition switch, or the 64 deaths that the Centers for Disease Control associated with pharmacy compounding errors in 2012.

One rule of thumb is to use raw numbers when they are under 100, and revert to some kind of fraction or rate when they grow bigger.

## Rates for large numbers

A raw per-person figure is an average and should usually be used with very big numbers.

A Gross Domestic Product of \$17 trillion is hard to digest. So we reduce it to a number we can understand. If we divide it by 317 million, we get about \$54,000 for every man, woman and child in the country. It doesn't mean that each person earned \$54,000 – in fact, almost half of all families earned less than that altogether at this writing. Instead, it includes all of the income that is generated by companies as well as people.

But the device turns an incomprehensible number into something we can picture. It also helps if we want to compare countries – it levels the playing field by adjusting for the size of the country.

## Rates for small numbers - crime, death, and other rare events

Rates such as 23 per 1,000 people or something like it – are the same as percentages, but you multiply by something bigger than 100 or move the decimal place further to the right. Use these for very small numbers.

If 2.5 million people die in this country every year, then the percentage of people who die is a really small number: 0.789 per 100, or percent.

A number that little is hard to digest. So experts up the ante and express the figure as 789 deaths per 100,000 people.

Be careful about rates based on very small numbers. One example is the number of police shootings per 100,000 people. Most police departments in the country are very small and are more likely to serve only about 5,000 people. This means that just one shooting in the department can lift them from one of the lowest rates in the nation to one of the highest. Expect rates based on very small numbers to be unstable and potentially misleading.<sup>2</sup>

## Figuring a rate

Step 1: Choose your base. This is often difficult. In reporting on fatalities by make of car, should you use the number of cars on the road, the number sold, or the total miles driven each year? You'll have to decide.

Step 2: Divide the number you care about by the base. Choosing the numerator can also be tricky. Going back to the automobile fatality example, would you use the total number of deaths or the number of driver deaths? Take a hint using other reports you see on the topic. Experts have often come to an informal agreement about what the most telling number is.

Step 3: Multiply by a nice round number, such as 1,000, 100,000 or 1 million.

---

<sup>2</sup>Note the disclaimer [in this story on police shootings by The Washington Post](#), in which changes in the rates of police shootings may just be random.

Step 4: Round the answer and simplify.

---

### Formula

Step 1: Choose the base, or “total”

Step 2: (Category / Total) = Proportion or Rate

Step 3: Proportion x 1,000 = Rate per thousand

Step 4: Round to zero decimal places

---

### Example

According to the FBI Crime in the United States for 2012, there were 13,000 violent and property crimes in Pittsburgh out of a population of 312,000. There were 8,870 crimes in Tucson out of a population of 531,000. Figuring a rate per thousand residents lets you compare the two cities:

Pittsburgh	Tucson
Step 1: Base= 312,000 people	Step 1: Base = 531,000 people
Step 2: 13,000 crimes / 312,000 people = 0.041	Step 2: 8,870 / 531,000 people = 0.017
Step 3: 0.041 x 1,000 = 41 crimes per thousand	Step 3: .017 * 1,000 = 17 crimes per thousand

So the crime rate for Pittsburgh was nearly 2 1/2 times that of Tucson that year , or  $47/17 = 2.4$

Selecting your multiplier

Some people feel that changing their multiplier from 100 to something bigger is cheating.

After all, a 0.2 percent rate becomes a big number – 200 – when you change the base from 100 to 100,000!

In practice, though, there's nothing magical about using a base of 100 (or percent). Instead, use the number that makes sense for the comparison you're making.

- Choose a round number – 1,000, 1 million or 100,000.

- Choose the same number that the experts use: Crimes per 1,000 people, deaths per 100,000, or crashes per million miles driven, for example.
- Choose a base that will give you an easy way to express it to your readers. This is one that results in a number generally between 1 and 1,000 or so.
- Try to avoid using an outrageously large base. For instance, avoid expressing a local number in terms of 1 million people. Only a handful of cities have more than a million people.
- Keep the same base throughout your story. Don't shift from 100,000 to 1,000 in crime statistics, for instance, when you move from murders to total crime rates.

You will often have to balance these rules of thumb against each other to come up with a compromise that allows you to write gracefully while keeping the sense of scale appropriate for the comparisons you're making.

## **Measuring change**

We often write about change or difference, usually as a difference between place or time.

### **Simple differences**

A simple difference is just the result of subtracting one number from another. If you are measuring differences in time, it's the newer number minus the older number.

One time to use a simple difference is when the number is understandable without any calculations. Prices of common household goods, salaries and home prices are examples of numbers that needn't always be put into perspective using percentage changes.

In the end, we work in news. That means that sometimes you'll use a raw number when it's more newsworthy. This doesn't necessarily mean the number is more alarming – just more meaningful.

#### **Figuring a difference:**

Subtract the older number from the newer number.

This is not the same as subtracting the little number from the big number.

If a number has fallen you get a negative number. If a number has risen you get a positive number.

#### **Formula**

New – Old.

**Example** An executive made \$2.4 million last year. She made \$2.9 million this year. Her raise was:  $\$2.9 - \$2.4 = 0.5$  million, or \$500,000, or half a million dollars.

## Percent change / Percent difference

The most butchered form of newsroom math is the percent difference, or the percent change.

Part of the problem is that some folks have found five or six different ways to compute them. Unfortunately, only two of them work every time. I'll show you both because sometimes – especially when you want to compare rates – one is easier than the other.

Note that these methods work whether or not the number is going up or going down. If the number has fallen, you'll get a negative answer. If the number has risen, you'll get a positive one. And it still comes out right if the increase is bigger than 100 percent.<sup>3</sup>

In practice, I use Method 1 when I'm working in spreadsheets because I can look at the simple difference in one column and then use it in the formula for the percentage difference. I use Method 2 when I want to compare to percent changes to one another or when working with annual rates.

### Method 1: Subtract then divide

### Method 2: Divide then subtract

### Figuring a percent change

Step 1: Get the simple difference between the numbers by subtracting the older number from the newer number. It doesn't matter which one is bigger!

Step 2: Divide the answer by the older number.

Step 3: Multiply by 100, or move the decimal point two places to the right.

Step 4: Round off and simplify.

Step 1: Get the proportion of the new number compared to the old number. This is the same as the percent of total above, except the old number is the base.

Step 2: Subtract 1 from that ratio

Step 3: Multiply by 100, or move the decimal point two places to the right.

Step 4: Round off and simplify.

### Formula

---

<sup>3</sup>It's impossible for a number to fall more than 100 percent. That would mean it went below zero and then no formula works. There's no good way to show a percent change when a figure like annual company earnings goes from profit to loss.

Step 1: New – Old = Difference

Step 2: Difference / Old = Decimal answer

Step 3: Decimal x 100 = percentage difference

Step 4: Round off.

Step 1: New / Old = Ratio

Step 2: Ratio - 1 = Decimal answer

Step 3: Decimal x 100 = percentage difference

Step 4: Round off.

### Example

*An executive made \$2.4 million last year. They made \$2.9 million in this year.*

Step 1: Difference =  $2.9 - 2.4 = 0.5$

Step 2: Difference / Original number =  $0.5 / 2.4 = .208$

Step 3: Move the decimal point = 20.8%

Step 4: Round off and simplify:  $21\% = 21 / 100 =$  about  $20 / 100 =$  or about one-fifth.

Step 1: Ratio =  $2.9/2.4 = 1.208$

Step 2: Decimal answer =  $1.208 - 1 = .208$

Step 3: Move the decimal point = 20.8%

Step 4: Round off and simplify:  $21\% = 21 / 100 =$  about  $20 / 100 =$  or about one-fifth.

*So the executive got a raise equivalent to one-fifth of their original salary.*

Reversing or predicting a percent change

Remember that a number can grow many times, but it can only fall 100 percent to zero. This is a rough concept until you think it through. If you double a price of \$20, increasing it by 100%, it's \$40. If you triple it, it's \$60. But if you reduce the \$40 back to \$20, it's a 50 percent drop, to one-half the level, not a 100 percent decrease. In other words, percent changes can't be reversed.

This means that the ads claiming you'll use three times less detergent or a food contains three times less salt are wrong and impossible. What they probably mean is that it would be three times as much if you used the other brand or ate the other food, or the brand is one third as much. Here are two ways this makes a difference:

You need two of three numbers to reverse or predict a percent change:

- Where the number started

- Where it ended
- What the percent change would (or will) be

Any two of those will give you what you need. It's easiest if we use Method 2 above to get there. The example above assumes you know where it started and where it ended. Here's how to do it if you have either of the other two:

### **Where it starts and the percent change**

Example: You started with \$100 and it grew by a total of 12%. Or, you started with \$100 and it fell by a total of 12% (the percent change was -12%)

Step 1: Convert the percent change to a ratio by moving the decimal place back : .12 (up) or -.12 (down)

Step 2: Add 1, resulting in 1.12 (up) or .88 (down)

Step 3: Multiply the beginning number by that amount :  $\$100 \times 1.12 = \$112$  (up), or  $\$100 \times .88 = \$88$  (down)

### **Where it ends and the total percent change**

For example, say your house is worth \$330,000, and it had appreciated by a total of 15% over the past few years. Here's how to figure out where it started:

Step 1: Convert the percent change to a decimal, as above: 0.15

Step 2: Divide the current value by that amount =  $\$330 / .15 = \$287$

This isn't intuitive, but it differs because you're starting from a bigger base. 15 percent of 330 isn't the same as 15 percent of 287.

### **Going further with percents and rates**

There are three common problems in changes and rates you will probably encounter that aren't part of this guide. You should get help or look it up when these situations come up: <sup>4</sup>

1. Relative risk: That's the technical term for dividing two percentages. If the mortgage denial rate for Black homeowners was 10 percent, and the denial rate of white homeowners was 5 percent, it means that Black homeowners are twice as likely to be denied a loan. This can be used with both rates and with changes.
2. Annual rates: When you know that something has grown, say, 2 percent a year for 10 years, it's not the same thing as 20 percent. You have to annualize it.

---

<sup>4</sup>They are part of the "Numbers in the Newsroom" book from which this guide is derived.

3. Adjusting for inflation: Comparing values across two points in time – especially today – means putting them on the same footing. Generally, you want to convert old values to their buying power today. For example, it's hard to compare salaries for teachers today with those 50 years ago, because our money isn't worth as much today.

## Average and typical values

Averages<sup>5</sup> are just summaries. If a quote sums up an event, or an anecdote sums up a person using their actions instead of words, an average sums up a human condition of some kind – money, congestion, death or disease – in a single number.

Choosing your average carefully or deciding there may be another number or method to sum up a situation can mean the difference between accurately and inaccurately describing your story.

Understanding different kinds of “measures of central tendency” – what they tell us and what they don’t – is the first thing you learn in basic statistics classes. If an it doesn’t describe your data well, it’s not very productive to move forward into many other kinds of analysis.

Trying to compare populations over time is particularly tricky using averages because of giant demographic shifts. Between the Baby Boom and the Millennials came what some people call the Baby Bust. Getting average spending on education, for example, across these generations is really misleading – it will boom, then bust, then boom again and no one number will describe that pattern.<sup>6</sup>

Two types of averages are reviewed here. Consult an introductory statistics book if your work depends on an average.

### The average or mean

A “mean” is what people mean when they say the word “average”.

It's most descriptive when it summarizes numbers that don't vary too much at either the top or bottom ends. These averages will often be misleading when they refer to items measured in dollar amount like incomes, housing costs and the like.

#### Figuring a simple average or mean

Step 1: Add up a list of numbers.

Step 2: Divide the answer by the number of numbers you've added up.

---

<sup>5</sup>I'm using the term “average” freely here. Technically, a simple average and median are measures of central tendency, but I'll treat them as different types of averages for simplicity sake.

<sup>6</sup>This is sort of an example of “Simpson’s paradox” in that an average hides meaningful trends among sub-populations.

## **Formula**

Step 1: Sum of numbers

Step 2: Sum / Count of numbers

For spreadsheet users: =AVERAGE(list of numbers)

## **Example**

Here are six home prices on a block:

\$275,000	\$1,200,000
\$275,000	\$500,000
\$200,000	\$395,000

Step 1:  $275 + 275 + 200 + 1,200 + 500 + 395 = 2,845$  or \$2,845,000

Step 2:  $\$2,845,000 / 5 = \$569,000$ .

So the average home price is more than all but one on the list.

## **The median**

Medians are often used to summarize the value of things measured in dollars, especially home prices and incomes. They are not sensitive to one or two unusually high or low values the way the average in the previous example is.

But it's harder to get a median because you need a list of all values. For example, if you know the total income of a metropolitan area and the number of people in that area, you can compute the average – or per capita income – but not the median.

One way to express the median is to call it the “typical” value. Another way is to say that it's the “middle” value.

### **Figuring a median:**

Step 1: List all of your numbers in order, beginning with the lowest and ending with the highest.

Step 2: Count how many numbers you have and divide by two.

Step 3: Add 0.5. If that comes out to a whole number (like 13), count up the list that many values.

If it's not (like 12.5), take the average of the two numbers surrounding the number. <sup>7</sup>

---

<sup>7</sup>In statistical programs like R, there are various ways to specify how to deal with medians when there are ties like this. This is the most common way, but it may not be the way your program handles it.

In other words, this is the closest you can get to the middle of the list. This is a sorting and counting job, not a calculator job.

In a spreadsheet, use the =MEDIAN() function.

**Example:**

Step 1:

The same list, but listed from lowest to highest, with an extra expensive home

1. \$200,000
2. \$275,000
3. \$275,000
4. \$395,000
5. \$500,000
6. \$1,200,000

Step 2:  $6/2 = 3$

Step 3:  $3 + .5 = 3.5$

Step 4: Average the 3rd and 4th items on the list:  $(275 + 395) / 2 = \$335,000$

As a rule of thumb, the median will be more telling than the average when they're very different as in this example. But the word "median" sounds very technical to some readers and the average encompasses all of the values in a list, so we use it when they're not too different.