

Hands-On Data Visualization

Interactive Storytelling from Spreadsheets to Code

Jack Dougherty Ilya Ilyankou

2020-11-20

Contents

Preface	9
Authors & Acknowledgements	9
Audience and Outline	12
Follow Along on a Computer	12
1 Introduction: Why Data Visualization?	15
What Can You Believe?	16
Some Pictures Are More Persuasive	18
Different Shades of the Truth	19
Organization of the Book	22
2 Choose Tools to Tell Your Story	25
Start Sketching Your Data Story	25
Ten Factors When Considering Tools	28
Our Recommended Tools	33
Use a Password Manager	34
3 Strengthen Your Spreadsheet Skills	37
Select your Spreadsheet Tools	38
Download to CSV or ODS Format	41
Make a Copy of a Google Sheet	42
Share Your Google Sheets	44
Upload and Convert to Google Sheets	45
Geocode Addresses in Google Sheets	48

Collect Data with Google Forms	51
Sort and Filter Data	53
Calculate with Formulas	55
Summarize Data with Pivot Tables	58
Match Columns with VLOOKUP	61
Spreadsheet vs. Relational Database	64
4 Find and Question Your Data	69
Guiding Questions for Your Search	70
Public and Private Data	73
Mask or Aggregate Sensitive Data	76
Open Data Repositories	78
Source Your Data	79
Recognize Bad Data	81
Question Your Data	83
5 Clean Up Messy Data	87
Find and Replace with Blank	88
Transpose Rows and Columns	90
Split Data into Separate Columns	91
Combine Data into One Column	93
Extract Tables from PDFs with Tabula	95
Clean Data with OpenRefine	98
6 Make Meaningful Comparisons	105
Precisely Describe Comparisons	106
Normalize Your Data	108
Beware of Biased Comparisons	111

CONTENTS	5
7 Chart Your Data	115
Chart Design Principles	119
Google Sheets Charts	127
- Bar and Column Charts	129
- Histograms	138
- Pie, Line, and Area Charts	140
Datawrapper Charts	145
- Annotated Charts	146
- Range Charts	150
- Scatter and Bubble Charts	153
Tableau Public Charts	160
- Scatter Chart	161
- Filtered Line Chart	165
8 Map Your Data	171
Map Design Principles	174
Design Choropleth Colors & Intervals	179
Normalize Choropleth Map Data	185
Point Map with BatchGeo	187
Point Map with Google My Maps	190
Symbol Point Map with Datawrapper	195
Choropleth Map with Datawrapper	200
Choropleth Map with Tableau Public	206
Current Map with Socrata Open Data	213
9 Table Your Data	221
Table Design Principles	222
Datawrapper Table with Sparklines	224
10 Embed On the Web	233
Static Image vs Interactive iframe	233
Get the Embed Code or iframe Tag	235
Paste Code or iframe to Website	243

11 Edit and Host Code with GitHub	249
Copy, Edit, and Host a Simple Leaflet Map Template	251
Convert GitHub Pages Link to iframe	259
Create a New Repo and Upload Files on GitHub	260
GitHub Desktop and Atom Editor to Code Efficiently	265
12 Chart.js and Highcharts Templates	275
Bar or Column Chart with Chart.js	278
Error Bars with Chart.js	281
Line Chart with Chart.js	283
Annotated Line Chart with Highcharts	284
Scatter Chart with Chart.js	286
Bubble Chart with Chart.js	288
13 Leaflet Map Templates	291
Leaflet Maps with Google Sheets	294
Leaflet Storymaps with Google Sheets	306
Get Your Google Sheets API Key	320
Leaflet Maps with CSV Data	325
Leaflet Heatmap Points with CSV Data	326
Leaflet Searchable Point Map	328
Leaflet Maps with Open Data APIs	330
14 Transform Your Map Data	333
Bulk Geocode with US Census	334
Pivot Points into Polygon Data	335
GeoJSON and Geospatial Data	336
Find GeoJSON Boundary Files	342
Draw and Edit with GeoJson.io	342
Edit and Join with Mapshaper	347
Georeference with MapWarper	361
Convert Compressed KMZ to KML	361

CONTENTS	7
15 Detect Lies and Reduce Data Bias	365
How to Lie with Charts	366
How to Lie with Maps	376
Recognize and Reduce Data Bias	382
Map Area and Projection Bias	386
16 Tell and Show Your Data Story	391
Build a Narrative on a Storyboard	392
Draw Attention to Meaning	395
Acknowledge Sources & Uncertainty	397
Decide On Your Data Story Format	399
A Fix Common Mistakes	401

Preface

This **BOOK-IN-PROGRESS** was last updated on: 20 Nov 2020.

Read the open-access web edition at <https://HandsOnDataViz.org>. This book is under contract with O'Reilly Media, Inc., which will sell print and ebook versions in April 2021.

Tell your story and show it with data, using free and easy-to-learn tools on the web. This introductory book teaches you how to design interactive charts and customized maps for your website, beginning with easy drag-and-drop tools, such as Google Sheets, Datawrapper, and Tableau Public. You'll also gradually learn how to edit open-source code templates like Chart.js, Highcharts, and Leaflet on GitHub. Follow along with the step-by-step tutorials, real-world examples, and online resources. This book is ideal for students, non-profit organizations, small business owners, local governments, journalists, academics, or anyone who wants to tell their story and show the data. No coding experience is required.

Send corrections for this book-in-progress to handsondataviz@gmail.com, or open an issue or submit a pull request on our GitHub repository. If you submit a GitHub pull request, in your commit message, please add the sentence "I assign the copyright of this contribution to authors Jack Dougherty and Ilya Ilyankou" to give us the option to publish it, with credit to you.

View open-source code for the book text and code templates at <https://github.com/handsondataviz>.

Hands-On Data Visualization is copyrighted by Jack Dougherty and Ilya Ilyankou and distributed under a Creative Commons BY-NC-ND 4.0 International License. You may freely share this content for non-commercial purposes, with a source credit to <http://HandsOnDataViz.org>.

Authors & Acknowledgements

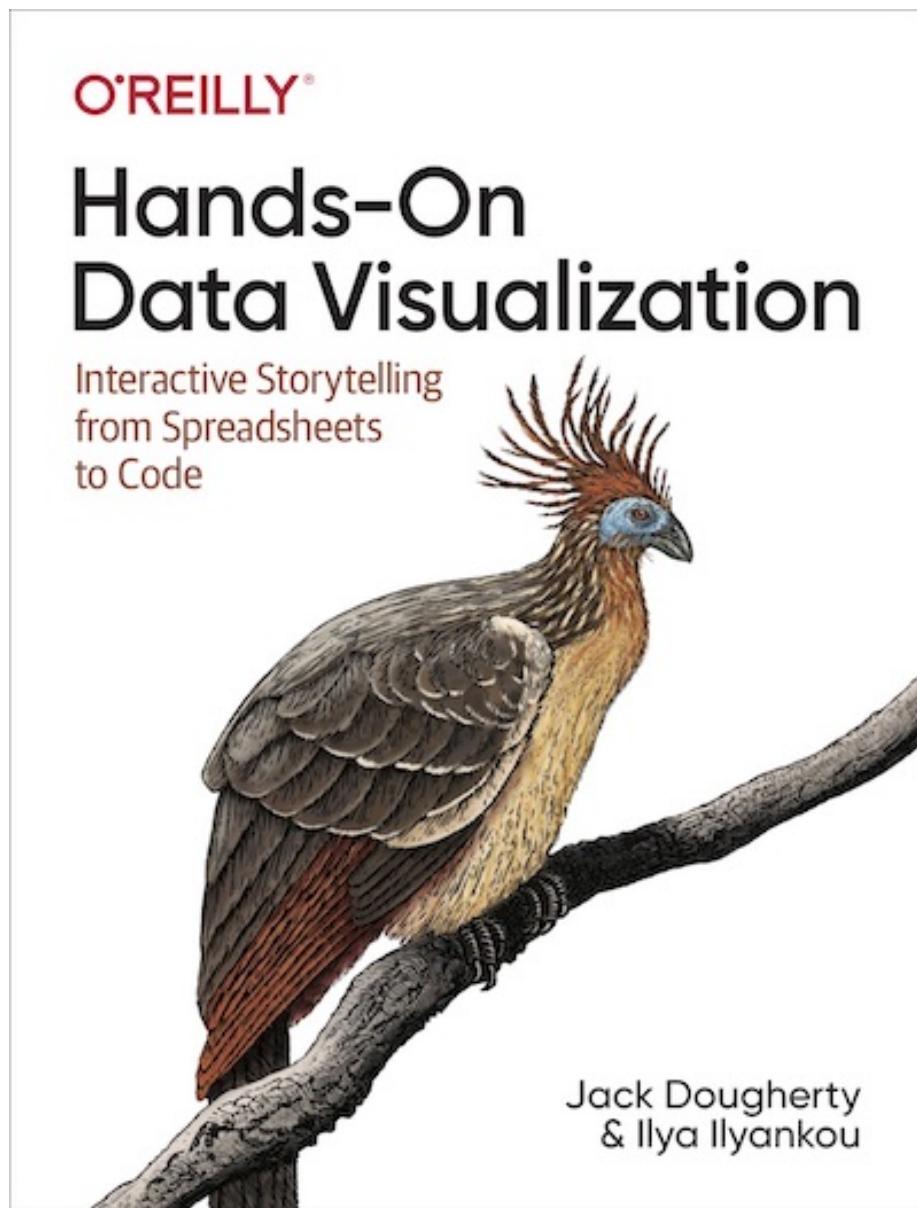


Figure 1: Book cover: Read about the hoatzin “reptile bird”

Authors	About Us
	<p>Jack Dougherty is Professor of Educational Studies at Trinity College in Hartford, Connecticut, where he and his students partner with community organizations to help tell their data stories on the web. Follow him on Twitter and on GitHub.</p>
	<p>Ilya Ilyankou is a Civic Technologist at Connecticut Data Collaborative. He has completed a double major in Computer Science and Studio Arts in the Class of 2018 at Trinity College. Visit his website or follow him on GitHub.</p>

Acknowledgements

We originally launched an earlier draft of this book under a different title, *Data Visualization For All*, to accompany a free online edX course by the same name, sponsored by Trinity College. Co-instructors Stacy Lam (Trinity Class of 2019) and David Tatem (Instructional Technologist) contributed valuable ideas and energy as we created content for that course in Spring 2017, which to date has enrolled over 23,000 students, though only a fraction actually complete the 6-week curriculum. Thanks also to the Trinity Information Technology staff and friends who produced the edX course videos: Angie Wolf, Sean Donnelly, Ron Perkins, Samuel Oyebefun, Phil Duffy, and Christopher Brown. Also, Veronica X. Armendariz (Trinity Class of 2016) made valuable contributions to the early version of the book while serving as a teaching assistant for the Data Visualization internship seminar that brought together Trinity undergraduates and Hartford community partners. Funding for students who worked on the earlier draft was generously provided by the Office of Community Learning and Information Technology Services at Trinity College.

Thanks to many individuals and organizations who helped us to learn many of the skills that we teach in this book, especially Alvin Chang and Andrew Ba Tran, who previously were data journalists at *The Connecticut Mirror*; and

Michael Howser, Steve Batt, and their colleagues at the University of Connecticut Libraries Map and Geographic Information Center (MAGIC). Also, many people inspired us to be *code-curious* at The Humanities and Technology Camp (THATCamp) events, sponsored by the Roy Rosenzweig Center for History and New Media at George Mason University, and engaged us with civic technology for the public good at Transparency Camp events sponsored by the Sunlight Foundation. We also appreciate Scott Gaul for inviting us to share our work-in-progress at Hartford data workshops and discussions, sponsored by the Hartford Foundation for Public Giving.

We thank everyone at O'Reilly Media who worked with us to bring you this book, especially our outstanding developmental editor, Amelia Blevins, and other members of the team: Nick Adams, Jonathan Hassel, Andy Kwan, Katie Tozer,... We also thank O'Reilly's support for technical reviewers whose valuable feedback helped us to improve the manuscript, including Carl Allchin, Derek Eder, Erica Hayes, etc...., and additional readers, including Gared Bard, Nick Klagge....

Audience and Outline

TODO: Intended audience

TODO: Chapter outline

TODO: what this book will not cover...

Follow Along on a Computer

TODO: Decide about renaming to “Follow Tutorials On A Computer”

To follow the steps in this book, we recommend either a desktop or laptop computer, running either the Mac or Windows or Linux operating system, with an internet connection and a modern web browser such as Chrome, Firefox, Safari, or Edge. Another good option is a Chromebook laptop, which enables you to complete *most* of the steps in this book, and we'll point out any limitations in specific chapters. While it's possible to use a tablet or smartphone device, we do not recommend it because you cannot follow all of the steps, and you'll also get frustrated with the small screen and perhaps throw your device (or this book) across the room, and possibly injure someone (and we will not be held responsible!)

If you're working on a laptop, consider buying or borrowing an external mouse that can plug into your machine. We've met several people who found it much easier to click, hover, and scroll with a mouse rather than a laptop's built-in trackpad.

If you're new to working with computers—or teaching new users with this book—consider starting with mouse exercises. All of the tools in this book assume that users already know how to click tiny buttons, hover over links, and scroll web pages, but rarely are these skills taught, and everyone needs to learn them at some point in our lives.

Trademarks

TODO: Discuss with editor if this is necessary, and if so, whether to place in the frontmatter?

Any use of a trademarked name without a trademark symbol is for readability purposes only. We have no intention of infringing on the trademark.

- GitHub and the GitHub logo are registered trademarks of GitHub, Inc.
- Google and the Google logo are registered trademarks of Google Inc.
- WordPress is a registered trademark of the WordPress Foundation
- TODO: Add others...

Disclaimer

The information in this book is provided without warranty. The authors and publisher have neither liability nor responsibility to any person or entity related to any loss or damages arising from the information contained in this book.

Chapter 1

Introduction: Why Data Visualization?

In this book, you'll learn how to create true and meaningful data visualizations through chapters that blend design principles and step-by-step tutorials, in order to make your information-based analysis and arguments more insightful and compelling. Just as sentences become more persuasive with supporting evidence and source notes, your data-driven writing becomes more powerful when paired with appropriate tables, charts, or maps. Words tell us stories, but visualizations show us *data stories* by transforming quantitative, relational, or spatial patterns into images. When visualizations are well-designed, they draw our attention to what is most important in the data in ways that would be difficult to communicate through text alone.

Our book features a growing number of free and easy-to-learn digital tools for creating *data visualizations*. We broadly define this term primarily as *charts*, which encode data as images, and *maps* which add a spatial dimension. While *tables* do not illustrate data in the same way, we include them in this book because of our pragmatic need to navigate new learners through a decision-making process that often results in building one of these three products. Furthermore, in this digital era we define data visualizations as images that can be easily re-used by modifying the underlying information, typically stored in a data file, in contrast to *infographics* that are generally designed as single-use artwork.¹

As educators, we designed *Hands-On Data Visualization* to introduce key concepts and provide step-by-step tutorials for new learners. You can teach yourself, or use the book to teach others. Also, unlike many technical books that focus

¹Note that other data visualization books may use these terms differently. For example, all visualizations are defined as “charts” in Alberto Cairo, *How Charts Lie: Getting Smarter About Visual Information* (W. W. Norton & Company, 2019), https://www.google.com/books/edition/How_Charts_Lie_Getting_Smarter_about_Vis/qP2KDwAAQBAJ, p. 23.

solely on one tool, our book guides you on how to choose among over twenty free and easy-to-use visualization tools that we recommend. Finally, while some other books only focus on *static* visualizations that can be distributed only on paper or PDF documents, we demonstrate how to design *interactive* tables, charts, and maps, and embed them on the web. Interactive visualizations engage wider audiences on the internet by inviting them to interact with the data, explore patterns that interest them, download files if desired, and easily share your work on social media.

Data visualizations have spread widely across on the internet over the last decade. Today in our web browsers we encounter more digital charts and maps than we previously saw in the print-only past. But rapid growth also raises serious problems. The “information age” now overlaps with the “age of disinformation.” Now that nearly anyone can post online, how do you make wise decisions about whom to trust? When presented with conflicting data stories about divisive policy issues such as social inequality or climate change, which one do you believe? In the next section, we’ll delve into this thorny topic by exploring what types of evidence persuades you, and why. And we’ll share this dirty little secret about data visualization: it illuminates our path in pursuit of the truth, but it also empowers us to deceive and lie.

What Can You Believe?

To begin, how do you know whether or not to believe us, the authors of this book? Could we be lying to you? How do you determine what information is truthful? Let’s start with a simple one-sentence statement:

Claim 1. Economic inequality has sharply risen in the United States since the 1970s.

Do you believe this claim—or not? Perhaps you’ve never thought about the topic in this particular way before now (and if so, it’s time to wake up). It’s possible your response depends on whether this statement blends in with your prior beliefs, or pushes against them. Or perhaps you’ve been taught to be skeptical of claims lacking supporting evidence (and if so, thank your teachers).

So let’s move on to a more complex two-sentence statement that also cites a source:

Claim 2. In 1970, the top 10 percent of US adults received an average income of about \$135,000 in today’s dollars, compared to the bottom 50 percent who earned around \$16,500. This inequality gap grew sharply over the next five decades, as the top tier income climbed to about \$350,000, while the bottom half barely moved to about \$19,000, according to the World Inequality Database.²

²World Inequality Database, “Income Inequality, USA, 1913-2019,” 2020, https://wid.world/share/#0/countrytimeseries/aptinc_p50p90_z;aptinc_p90p100_z;aptinc_p0p50_z/US/2015/kk/k/x/yearly/a/false/0/400000/curve/false.

Is this second claim more believable than the first one? It now makes a more precise claim by defining economic inequality in terms of average income for the upper 10 percent versus the bottom 50 percent over time. Also, this sentence pins its claims to a specific source, and invites us to read further by following the footnote. But how do these factors influence its persuasiveness? Does the sentence lead you to ask about the trustworthiness of the source and how it defines “income”? Does the wording make you wonder about the other 40 percent of the population between the two extremes?

To answer some of those questions, let’s supplement the second claim with a bit more information, as shown in Table 1.1.

Table 1.1: Average US Adult Income, 1970-2019

US Income Tier	1970	2019
Top 10 Percent	\$136,308	\$352,815
Middle 40 Percent	\$44,353	\$76,462
Bottom 50 Percent	\$16,515	\$19,177

Note: Shown in constant 2019 US dollars. National income for individuals aged 20 and over, prior to taxes and transfers, but includes pension contributions and distributions. Source: World Inequality Database 2020

Does Table 1.1 make Claim 2 more persuasive? Since the table contains essentially the same information as the two sentences about top and bottom income levels, it shouldn’t make any difference. But the table communicates the evidence more effectively, and makes a more compelling case. For many people, it’s easier to read and grasp the relationship between numbers when they’re organized in a grid, rather than complex sentences. As your eyes skim down the columns, you automatically notice the huge jump in income for the top 10 percent, which nearly tripled over time, while the bottom 50 percent barely budged. In addition, the table fills in more information that was missing from the text about the middle 40 percent, whose income grew over time, but not nearly as much as the top tier. Furthermore, the note at the bottom of the table adds a bit more context about how the data is “shown in constant 2019 US dollars,” which means that the 1970s numbers were adjusted to account for changes to the cost of living and purchasing power of dollars over a half-century. The note also briefly mentions other terms used by the World Inequality Database to calculate income (such as taxes, transfers, and pensions), though you would need to consult the source for clearer definitions. Social scientists use different methods to measure income inequality, but generally report findings similar to those shown here.³

³The World Inequality Database builds on the work of economists Thomas Piketty, Emmanuel Saez, and their colleagues, who have constructed US historical income data based not only on self-reported surveys, but also large samples of tax returns submit-

Some Pictures Are More Persuasive

Now let's substitute a data visualization—specifically the line chart in Figure 1.1—in place of the table, to compare which one is more persuasive.

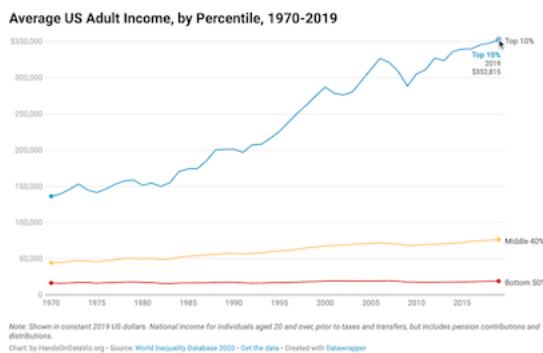


Figure 1.1: Explore the interactive line chart of US adult income inequality over time.

Is Figure 1.1 more persuasive than Table 1.1? Since the line chart contains the same historical start and stop points as the table, it should not make any difference. But the line chart also communicates a powerful, visualized data story about income gaps that grabs your attention more effectively than the table. As your eyes follow the colored lines horizontally across the page, the widening inequality between the top versus the middle and bottom tiers is striking. The chart also packs so much granular information into one image. Looking closely, you also notice how the top-tier income level was relatively stable during the 1970s, then spiked upward from the 1980s to the present, and grew more distant from other lines. Meanwhile, as the middle-tier income rose slightly over time, the fate of the lowest-tier remained relatively flat, reached its peak in 2007, and then dipped back downward for much of the past decade. The rich got richer, and the poor got poorer, as the saying goes. But the chart reveals how rapidly those riches grew, while poverty remained recalcitrant in recent years.

Now let's insert Figure 1.2, which contains the same data as Figure 1.1, but presented in a different format. Which chart should you believe? Remember, we warned you to watch out for people who use data visualizations to tell lies.

ted to the Internal Revenue Service. See WID methods at World Inequality Database, “Methodology,” WID - World Inequality Database, 2020, <https://wid.world/methodology/>. See overview of methodological approaches in Chad Stone et al., “A Guide to Statistics on Historical Trends in Income Inequality,” Center on Budget and Policy Priorities, January 13, 2020, <https://www.cbpp.org/research/poverty-and-inequality/a-guide-to-statistics-on-historical-trends-in-income-inequality>. See comparable findings on US income inequality by the Pew Charitable Trust in Julia Menasce Horowitz, Ruth Igielniak, and Rakesh Kochhar, “Trends in U.S. Income and Wealth Inequality,” Pew Research Center’s Social & Demographic Trends Project, January 9, 2020, <https://www.pewsocialtrends.org/2020/01/09/trends-in-income-and-wealth-inequality/>

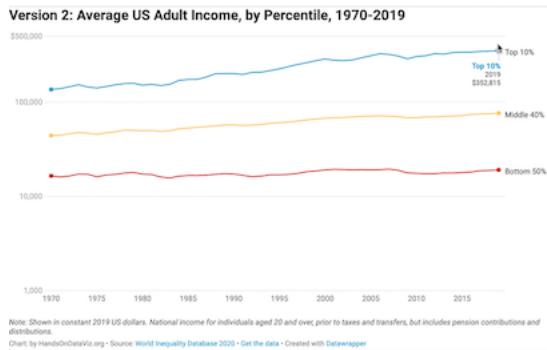


Figure 1.2: Explore the alternative version of the interactive line chart of US adult income inequality over time, using the same data as the first version.

What's going on? If Figure 1.2 contains the same data as Figure 1.1, why do they look so different? What happened to the striking growth in inequality gaps, which now seem to be smoothed away? Did the crisis suddenly disappear? Was it a hoax?

Although the chart in Figure 1.2 is technically accurate, it intentionally misleads readers. Look closely at the labels in the vertical axis. The distance between the first and second figures (\$1,000 to \$10,000) is the same as the distance between the second and the third (\$10,000 to \$100,000), but those jumps represent very different amounts of money (\$9,000 versus \$90,000). That's because this chart was constructed with a logarithmic scale, which is most appropriate for showing exponential growth. You may recall seeing logarithmic scales during the Covid pandemic, when they were appropriately used to illustrate very high growth rates, which are difficult to display with a traditional linear scale. This second chart is technically accurate, because the data points and scale labels match up, but it's misleading because there is no good reason to interpret this income data using a logarithmic scale, other than to deceive us about this crisis. People can use charts to illuminate the truth, but also can use them to disguise it.

Different Shades of the Truth

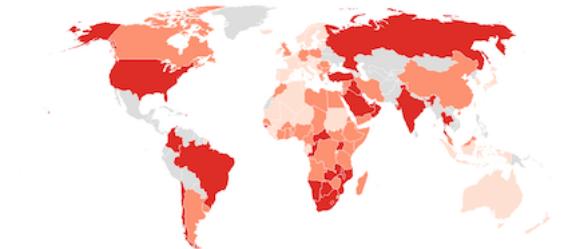
Let's expand our analysis of income inequality beyond the borders of one nation. Here's a new claim that introduces comparative evidence and its source. Unlike the prior US examples that showed historical data for three income tiers, this global example focuses on the most current year of data available for the top 1 percent in each nation. Also, instead of measuring income in US dollars, this international comparison measures the percentage share of the national income held by the top 1 percent. In other words, how large a slice of the pie is eaten by the richest 1 percent in each nation?

Claim 3. Income inequality is more severe in the United States, where the richest 1 percent of the population currently receives 20 percent of the national income. By contrast, in most European nations the richest 1 percent receives a smaller share, ranging between 6 to 15 percent of the national income.⁴

Following the same train of thought above, let's supplement this claim with a visualization to evaluate its persuasiveness. While we could create a table or a chart, those would not be the most effective ways to quickly display information for over 120 nations in our dataset. Since this is spatial data, let's transform it into an interactive map to help us identify any geographic patterns and to encourage readers to explore income levels around the globe, as shown in Figure 1.3.

Map: Share of National Income by Top 1 Percent

13% 19%



Map: By HandsOnDataViz.org • Source: World Inequality Database • Created with Datawrapper

Figure 1.3: Explore the interactive map of world income inequality, measured by the share of national income held by the top 1 percent of the population, based on the most recent data available. Source: World Inequality Database 2020.

Is Figure 1.3 more persuasive than Claim 3? While the map and the text present the same data about income inequality in the US versus Europe, there should be no difference. But the map pulls you into a powerful story that vividly illustrates gaps between the rich and poor, similar to the chart example above. Colors in the map signal a crisis. Income inequality in the US (along with Russia and Brazil) stands out in dark red at the highest level of the legend, where the top 1 percent holds 19% or more of the national income. By contrast, as your eye floats across the Atlantic, nearly all of the European nations appear

⁴World Inequality Database, “Top 1% National Income Share,” 2020, https://wid.world/world/#sptinc_p99p100_z/US;FR;DE;CN;ZA;GB;WO/last/eu/k/p/yearly/s/false/5.070499999999999/30/curve/false/country.

in lighter beige and orange colors, indicating no urgent crisis as their top-tier holds a smaller share of the national income.

Now let's introduce the alternative map in Figure 1.4, which contains the same data as shown in Figure 1.3, but is displayed in a different format. Which map should you believe?

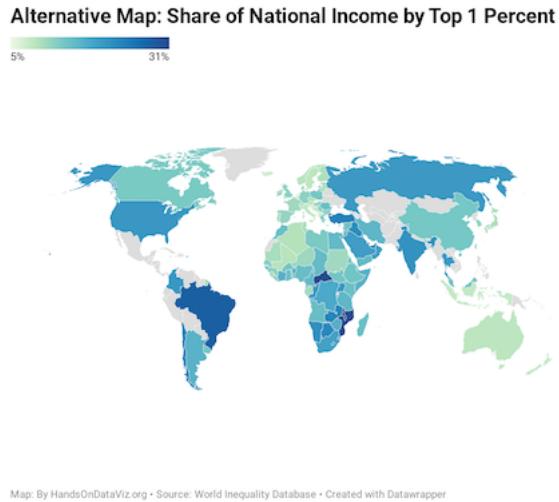


Figure 1.4: Explore an alternative version of the interactive map of world income inequality, using the same data as the map above.

Why does the second map in Figure 1.4 look different than the first map in Figure 1.3? Instead of dark red, the US is now colored medium-blue, closer on the spectrum to Canada and most European nations. Did the inequality crisis simply fade away from the US, and move to dark-blue Brazil? Which map tells the truth?

This time, neither map is misleading. Both make truthful interpretations of the data with reasonable design choices, even though they create very different impressions in our eyes. To understand why, look closely at the map legends. The first map sorts nations in three categories (less than 13%, 13-19%, 19% and above), while the second map displays the entire range in a green-blue color gradient. Since the US share is 20.5 percent, in the first map it falls into the top bucket with the darkest red color, but in the second map it falls somewhere closer to the middle as medium-blue color. Yet both maps are equally valid, because neither violates a definitive rule in map design nor intentionally disguises data. People can mislead with maps, but it's also possible to make more than one portrait of the truth.

The interpretive nature of data visualization poses a serious challenge. As the authors of this book, our goal is to guide you in creating truthful and meaningful

charts and maps. We'll point you toward principles of good design, encourage thoughtful habits of mind, and try to show by example. Occasionally we'll even tell you what *not* to do. But data visualization is a slippery subject to teach, sometimes more art than science. We know that charts and maps can be manipulated—just like words—to mislead your audience, and we'll demonstrate common deception techniques to help you spot them in other people's work, and consciously avoid them in your own. But newcomers may be frustrated by the somewhat fuzzy rules of data visualization. Often there is no *single* correct answer to a problem, but rather *several* plausible solutions, each with their own strengths and weaknesses. As a learner, your job is to continually search for *better answers* without necessarily expecting to find the *one right answer*, especially as visualization methods and tools continue to evolve, and people invent new ways to show the truth.

Organization of the Book

We've organized the chapters of this book to serve as an introductory hands-on guide to data visualization, from spreadsheets to code. Also, we assume no prior skills other than general familiarity with operating a computer, along with an innate curiosity about telling stories with data. Imagine the book in four parts.

In part one, you'll develop foundational skills about envisioning your data story, along with the tools and data you'll need to tell it. We'll gradually move from Chapter 2: Choose Tools to Tell Your Data Story to Chapter 3: Improve Your Spreadsheet Skills to Chapter 4: Find and Question Your Data to Chapter 5: Clean Up Messy Data to Chapter 6: Make Meaningful Comparisons. These chapters feature hands-on tutorials to enrich learning by doing.

In part two, you'll build lots of visualizations with easy-to-learn drag-and-drop tools, and find out which types work best with different data stories. We'll start with Chapter 7: Chart Your Data, Chapter 8: Map Your Data, and Chapter 9: Table Your Data and develop your understanding of the interpretive style that each one emphasizes. In Chapter 9: Embed on the Web, you'll learn how to insert all of these interactive visualizations on common web platforms, to invite readers to explore your data and share your work more widely.

In part three, you'll advance to working with more powerful tools, specifically code templates, that give you more control over customizing the appearance of your visualizations and where you host them online. We'll start with Chapter 11: Edit and Host Code with GitHub, and walk you through the easy web interface for this popular open-source coding platform. Then you'll build using Chapter 12: Chart.js and Highcharts Templates and Chapter 13: Leaflet Map Templates, and discover more advanced spatial tools in Chapter 14: Transform Your Map Data. At the end of the book we include an Appendix: How to Fix Common Problems to consult when you accidentally break your code, which is also a great way to learn how it works.

In part four, we'll wrap up all of the visualization skills you've developed by returning to the central theme of this introduction: telling true and meaningful stories with data. In Chapter 15: Detect Lies and Reduce Data Bias, you'll learn how to lie with charts and maps in order to do a better job of telling the truth. Finally, Chapter 16: Tell and Show Your Data Story emphasizes how the goal of data visualization is not simply to make pictures about numbers, but to craft a truthful narrative that convinces readers how and why your interpretation matters.

Summary

Now you have a clearer sense of our primary goal for this book. We aim for you to learn how to tell true and meaningful stories with interactive data visualizations, while being mindful of the ways that people can use them to mislead. In the next chapter, let's get started on clarifying the data story you wish to tell, and factors to consider when choosing tools to do the job.

Chapter 2

Choose Tools to Tell Your Story

If you feel overwhelmed by the avalanche of digital tools available today, you're not alone. When you're simply trying to do your regular work, keeping up with the latest software developments can feel like an additional part-time job you didn't sign up for. Digital tools are constantly changing and evolving. That's good news if you like to experiment and choose among different options, but not-so-good news if you lack the time to make complex decisions.

In this chapter, we'll help you navigate your way through the decision-making process. We'll begin with the most important step—sketching out your data story—to help identify the types of tools you need to tell it effectively. Next, we'll review ten factors to consider when choosing digital tools and the tradeoffs involved. Finally, we'll present our list of recommended data visualization tools, plus one extra to help you get organized: a password manager. All of these tools are free to use, and the book introduces them gradually, from easy-to-learn beginner tools to more advanced power tools that grant you more control over where your work is hosted and how it looks.

Start Sketching Your Data Story

Before we dive into digital tools, let's focus on what's most important: our *data story*. We build visualizations to help us tell a story about the information we've gathered, a narrative that draws the audience's attention to meaningful patterns and key insights amid all of the pieces of data. Help them to see the forest, rather than listing every single tree.

But in the early stage of a data visualization project, a common problem is that we don't yet have a clear sense of the key pieces of our data story, or how they

fit together. That's perfectly normal. One of the best ways to address that problem is a quick exercise that's designed to move partially-formed ideas from inside our heads out onto pieces of paper, to help you and any co-workers see them more clearly.

For this exercise, push away your computer and pick up some of our favorite old-school tools:

- several blank sheets of paper
- colored pencils, pens, or markers
- your imagination

Get ready to sketch out your data story in words and pictures. No artistic skills are required.

1. On the first sheet of paper, *write down the problem* that motivates your data project. If you prefer a prompt, try filling in these blanks: *We need to find out _____ in order to _____*. In many cases, people come to data visualization with an information-driven problem, which they hope will lead them to achieve a broader goal. For example, when working on the first draft of this book, our problem statement was: *We need to find out our readers' backgrounds and interests about data visualization, in order to write a better introductory guide that meets their needs*.
2. On the second sheet of paper, *rewrite your problem statement into a question*. Write a question for which you genuinely do not yet know the answer—and punctuate it with a question mark. If your brain is tempted to jump ahead and try to answer the question, fight that urge. Instead, focus on framing the question, by using more precise wording than what you wrote above, without limiting the range of possible results. For example, when working on the first draft, our question was: *How do readers of our book describe their prior experience with data visualization, education level, and learning goals?* While we had some preliminary guesses, we honestly didn't know the answer at that stage, which made it an authentic question.
3. On the third sheet of paper, *draw pictures and arrows to show how you'll find data* to answer your question above. Are you conducting door-to-door interviews with neighborhood residents, or sending an online survey to customers, or downloading family income and county maps from the US Census? Sketch a picture of your data collection process, to show how you plan to bring together different pieces of information. For example, when writing the first draft of our book, we asked readers to fill out a quick online survey form, and reminded them not to insert any private data, because we shared back their collected responses in a public spreadsheet.

4. On the fourth sheet of paper, *sketch at least one type of visualization you plan to create* after you obtain your data above. Do you envision some type of chart, like a bar, line, or scatter chart? Or do you imagine some type of map, maybe with points or polygons? If your visualizations will be interactive, try to show the concept using buttons and more than one sheet of paper. You can add *imaginary data* at this stage because it's just a preliminary sketch, as shown in Figure 2.1. Have fun!



Figure 2.1: Sketch out your story idea on four pages: problem, question, find data, visualize.

This exercise can help you in multiple ways, whether you do it by yourself, or even better, with a team of co-workers, as shown in Figure 2.2. First, by migrating ideas from your mind to paper, you'll make your thinking clearer not only for you, but also more visible for others. When ideas are sketched out, you can reflect on them, listen to feedback, cross-out not-so-good ones, and replace them with better ones on new sheets of paper. If your initial sketches are too complicated or confusing, break down those ideas into separate pages to make them more coherent.

Second, look at your sheets like a storyboard. Spread them out on a table, move them around to potentially reorder the sequence, start to define the three essential stages of your story: the beginning, middle, and end. Also, these pages can help you organize your thinking about how you'll communicate your data story to larger audiences, such as a presentation slide deck, or paragraphs and pictures for your next report or web page. Don't throw them away, because we'll return to this exercise at the end of the book in Chapter 16: Tell and Show Your Data Story.

Finally, this sketching exercise can help you identify which chapters you should focus on in the body of this book. If you're puzzled about where to search for data, check out Chapter 4: Find and Question Your Data. If you're thinking



Figure 2.2: The data story sketching exercise can be done solo, but works even better with a team of people. In our data visualization course, college students and community partners collaborate on framing the data story for their projects.

about building a chart or map, but need examples of different types, look at the beginning of Chapter 7: Chart Your Data and Chapter 8: Map Your Data.

Now that you have a clearer sense of the story you wish to tell, and some initial ideas about the visualizations you wish to create, in the next two sections we'll discuss tools to do the job, and factors you should consider when deciding among them.

Ten Factors When Considering Tools

Making decisions between the seemingly endless number of digital tools can feel overwhelming. To help you navigate your decision-making process, below we list ten key factors that we consider when evaluating new visualization tools or online services. When comparing options, many decisions involve some type of tradeoff, a balance between competing wants and needs, such as ease-of-use versus extensive features. By identifying key factors, we believe that each reader can make a more informed decision about which tools offer the best tradeoff for you, since all of us are different. Furthermore, we worded our categories broadly, because the concepts can be applied to other areas of your digital life, but followed up with more context about data visualization in particular.

1. Easy-to-learn

How much time will be required to learn a new tool? In our busy lives, this is often the most important factor, but also one that varies widely, as your personal investment of time and energy depends on your prior experience in using related tools and grasping key concepts. In this book, we use the label *Easy Tools* to identify those best suited for beginners (and even some advanced users prefer them, too). They usually feature a graphical user interface, meaning you operate them with pull-down menus or drag-and-drop steps, rather than memorizing commands to be typed into a blank screen. The better ones also offer user-friendly error messages that guide you in the right direction after a

wrong turn. Later in the book, we'll introduce *Power Tools* that provide more control and customization of your visualizations, such as code templates that you can copy and edit, which is easier than writing them from scratch. Overall, when deciding which tools to include in this book, we placed easy-to-learn at the top of our list. In fact, we removed a popular free drag-and-drop tool from an earlier draft of this book because even *we* had difficulty following our own instructions in how to use it. When faced with several good options, choose simplicity.

2. Free or Affordable

Is the tool free to use? Or is it based on a *freemium* model that offers basic functions for free, with premium features at a price? Or does it require paying a one-time purchase or monthly subscription fee? Of course, the answer to what is affordable will vary for each reader. We fully understand that the business model for many software developers requires steady revenue, and both of us willingly pay to use specific tools necessary for our work. If you regularly rely on a tool to do your job, with no clear alternative, it's in your best interest to financially support their continued existence. But when creating this book, we were impressed by the wide array of high-quality data visualization tools that are available at no cost to users. To increase access to data visualization for all readers, every tool we recommend is free, or its core features are freely available.

3. Powerful

Does the tool offer all of the features you anticipate needing? For example, does it support building sufficient types of data visualizations for your project? Although more is usually better, some types of charts are obscure and rarely used, such as radar charts and waterfall charts. Also, look out for limits on the amount of data you can upload, or restrictions on visualizations you create. For example, we previously removed a freemium tool from an earlier version of this book when the company began to require a paid license if your map was viewed more than 100 times on the web. Furthermore, to what extent does the tool allow you to customize the appearance of your visualizations? Since drag-and-drop and freemium tools commonly limit your display options, you may need to make tradeoffs between them versus more powerful and customizable tools. In this book, we begin with easy tools and gradually introduce more advanced ones in each chapter, to help you identify your ideal combination of simplicity and power.

4. Supported

Does the developer regularly maintain and update the tool, and respond to questions or issues? Is there an active user community that supports the tool and shares its knowledge about using it? If you've worked with digital tools as long as we have, you'll recognize our pain in losing several whose developers pulled the plug. For example, the Killed By Google lists nearly 200 applications and online services that this multi-billion dollar corporation closed down. One of these was a popular data visualization tool, Google Fusion Tables, which

once occupied a full chapter in an earlier version of this book, when we removed when Google shut down the tool after a ten-year run in 2019. Although none of us can predict which online tools will persist in future years, we looked for signs of active support before including them in this book, such as regular updates, stars earned on a GitHub developer’s site, and questions answered in the StackOverflow user forum. But never assume that the future will resemble the past. The continuous evolution of digital tools means that some become extinct.

5. Portable

How easily can you migrate your data *into* and *out of* a tool? For example, we stopped recommending an online story map tool created by a well-known software company when we discovered that while users could easily upload locations, text, and photos, but there was no way to export all of their work! As digital technology inevitably changes, all data will need to migrate to another platform, and it’s your job to be prepared for this eventual transition. Think about the issue as historical preservation, to increase the likelihood that your projects will continue to function on some unknown platform in the future. If your current tool developer announced that it was shutting down next month, could you easily extract all of the underlying data in a commonly-used file format to upload to a different tool? A key step to future-proof your visualizations is to ensure that your data files are easily separated from the presentation software that generates the charts or maps. When recommending tools for this book, we favored those that support portable data downloads for future migrations.

6. Secure and Private

This category combines related questions about security and privacy. First, does the online tool or service take reasonable precautions to protect your personal information from malicious hackers and malware? Review a list of major data breaches on Wikipedia to help you make informed decisions. If your tool developer recently experienced a malicious data hack, find out how they responded. Second, when you access tools through your browser, does they track your web activity across different sites? Also be aware of internet censorship by different governments around the globe, as compiled by Wikipedia, unless you happen to be reading this book in China, which has blocked access to all of Wikipedia since April 2019. Finally, does the tool clearly explain whether the data you enter or the products you create will stay private or become public? For example, some companies offer free access to their visualization tools, but in exchange require you to make your data, charts, and maps publicly accessible. That tradeoff may be acceptable if you’re working with open-access data and already plan to freely share your visualizations, as many journalists and scholars do. In any case, make sure the terms of service are clearly defined before you start using a tool.

7. Collaborative

Does the tool allow people to work together and co-create a data visualization?

If so, does the tool allow different levels of access or version control to help prevent team members from accidentally overwriting each other's contributions? Prior generations of digital tools were designed primarily for solo users, in part to address security and privacy issues raised above. But today, many data visualization projects require access and input from multiple team members. Collaboration is essential for success. As co-authors of this book, who jointly wrote the text and co-created many of the visualizations, we favor a newer generation of tools designed for team work environments.

8. Cross-Platform

This category refers to both creating and consuming digital content. First, does the tool work across different computer operating systems? In this book, we highlight several tools that run inside any modern web browser, which usually (but not always) means they will operate on all major desktop and laptop computer platforms, such as Windows, Mac, Chromebook, and Linux. When necessary, we specify when a tool will only run on specific computer operating systems, and this often reduces access for people using lower-cost computers. Second, does the tool create visualizations that are responsive to different screen sizes? In other words, does it produce charts and maps that display satisfactorily on smaller devices, such as smartphones and tablets? In this book, we favor cross-platform tools that also display content responsively on smaller devices, but we do not necessarily expect that tools can be operated on small devices to create visualizations. In other words, when we say that a tool runs inside any modern web browser, we don't necessarily mean phone and tablet browsers, but sometimes they work there, too.

9. Open-Source

Is the tool's software code publicly viewable? Can the code be modified and redistributed, so that other developers can suggest improvements, or build new features or extensions? We recognize that many developers rely on non-public proprietary code to sell their tools at a profit, and several of those appear in the book. But we also have been impressed with the number of high-quality data visualization tools offered under different types of open-source licensing arrangements, by sustainable communities of volunteer developers, non-profit organizations, and also for-profit companies who recognize some economic benefits of open-source code development. When recommending tools for this book, we highlight open-source options when available.

10. Accessible for Visually-Impaired Readers

Does the tool create visualizations that are accessible for visually-impaired readers? Although disability advocacy laws were passed decades ago, digital technology still lags behind and is slowly catching up, especially in the field of data visualization. But some tools include a built-in check for colorblindness and offer chart types designed for low-vision people using screen readers, as shown in Figure 2.3.

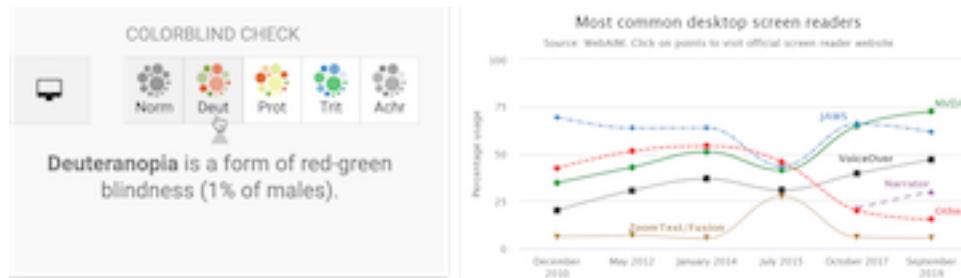


Figure 2.3: On the left, the Datawrapper built-in check for colorblindness. On the right, a Highcharts line chart designed for low-vision accessibility.

Those are ten factors we consider when deciding whether to add another item into our digital toolkit. Of course, your list may vary, and might include other values that are vitally important yet sometimes harder to judge, such as a software developer’s ethical business practices or contribution to the public good. Whatever criteria you value, make them explicit in your decision-making process, and inform others about what influences your choices.

Also consider alternative ways to make tool decisions. When visualization designer Lisa Charlotte Rost reflected on her fascinating experiment in recreating one chart with 24 different tools, she concluded that “there are no perfect tools, just good tools for people with certain goals.” On a related note, when digital historian Lincoln Mullen offered advice on making prudent choices about digital tools, his first recommendation was: “The best possible tool is the one you’re already using to get work done.” Don’t fall into the familiar trap of believing that your productivity will increase if only you began to use yet another new tool. Mullen’s second piece of advice was: “Prefer the tool that your local co-workers use.” Even if different tool is objectively better, it may be outweighed by the benefits of mutual support and collaboration with people using a less-awesome application in your local setting.¹

Now that you understand the factors driving our decisions, in the next section you’ll see an overview of our tool recommendations, with a quick description and link to the chapter where we introduce each of them.

¹Lisa Charlotte Rost, “What I Learned Recreating One Chart Using 24 Tools,” Source, December 8, 2016, <https://source.opennews.org/en-US/articles/what-i-learned-recreating-one-chart-using-24-tools/>; Lincoln Mullen, “How to Make Prudent Choices About Your Tools,” ProfHacker, August 14, 2013, <https://lincolnmullen.com/blog/how-to-make-prudent-choices-about-your-tools/>. See also criteria for educational tools by Audrey Watters, “The Audrey Test: Or, What Should Every Techie Know About Education?” Hack Education, March 17, 2012, <http://hackeducation.com/2012/03/17/what-every-techie-should-know-about-education>

Our Recommended Tools

When creating this book, we aimed to identify the most essential data visualization tasks that beginners are likely to face, and the digital toolkit needed to complete those tasks. In the prior section we listed ten factors that influenced our tool recommendations, such as being easy-to-learn, free or affordable, with powerful capacity. In this section, we have listed all of the tools featured in this book, with recommended uses and references to chapters where they appear, as shown in Table 2.1. Your data visualization projects may only require you to use only a small number of these, or perhaps even just one tool. But it's important to be aware of the different types of tools, because you may not realize how they can help you if don't know that they exist.

Table 2.1: Recommended Tools and Uses, with Chapter References

Tools	Collect	Clean	Chart	Geocode	Map	Table	Code	Transform
Google Sheets spreadsheet/charts	Ch3	Ch5	Ch7	Ch3		Ch9		
LibreOffice Calc spreadsheet/charts	Ch3							
Airtable relational database	Ch3							
Tabula PDF table extractor			Ch5					
OpenRefine data cleaner			Ch5					
Datawrapper charts/maps/tables				Ch7	Ch8	Ch8	Ch9	
Tableau Public charts/maps/tables				Ch7		Ch8	Ch8	
Chart.js code templates					Ch12			
Highcharts code templates					Ch12			
Google My Maps point map maker						Ch8	Ch8	
Leaflet map code templates							Ch13	
GitHub edit & host code								Ch11
GitHub Desktop & Atom code editor								Ch11
GeoJson.io edit & draw geodata								Ch14
Mapshaper edit & join geodata								Ch14
MapWarper georeference images								Ch14

In addition to the tools featured in Table 2.1, you'll find several other useful tools mentioned in the text, including ColorBrewer to select map colors, the Geocoding by SmartMonkey add-on for Google Sheets, US Census Geocoder for bulk addresses and census geography, and the Socrata open-repository chart and map tools. Also, consider enhancing your web security by installing the free Privacy Badger browser extension from the Electronic Frontier Foundation to view and exercise some control over who's tracking you, and also review the EFF's Surveillance Self-Defense Guide.

Of course, we recognize that digital tools are continually changing and evolving. As time goes by, we expect that some tools will no longer be available, and we also anticipate discovering newer ones that do a better job in telling our data

stories. If you'd like to recommend a tool that's not currently on our list, email us at handsondataviz@gmail.com or tweet us at [@handsondataviz](https://twitter.com/handsondataviz), and tell us how it rates on the ten factors that guide our selection process above.

Use a Password Manager

Finally, we highly recommend a password manager: think of it as one tool to rule them all! Password managers help you to keep track of all of the accounts you will create when using several of the online tools above. We recommend installing BitWarden, an open-source password manager that offers its core features for free for Windows, Mac, and Linux computers, all major web browsers, and iOS and Android mobile devices. When you install BitWarden, you create one universal password (be careful not to forget it) that grants you access to all of the account usernames and passwords you catalog. You also install the Bitwarden extension in your preferred web browsers. When you register for a new account in your browser, the password manager typically asks if you wish to store that information in your vault with end-to-end encryption. Also, when you visit that site in the future, the password manager usually recognizes it and enters your login credentials with one click, as shown in Figure 2.4.

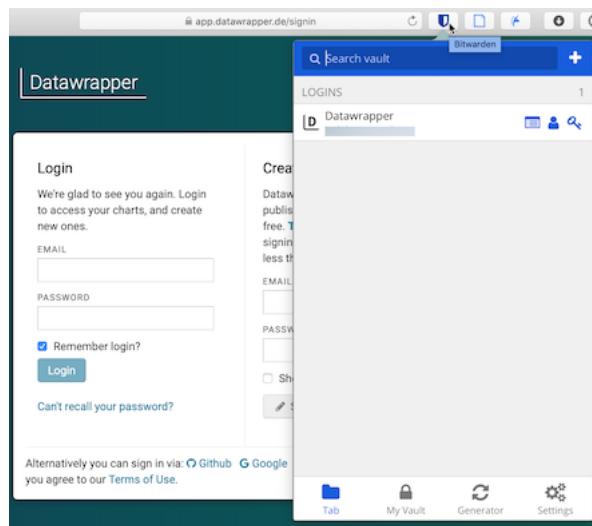


Figure 2.4: The Bitwarden browser extension recognizes sites you have previously stored, and enters your credentials with one click.

We recommend storing your passwords inside a tool like BitWarden, rather than in a specific web browser (such as Chrome or Firefox) for two reasons. First, you can set up BitWarden to sync and access your passwords across *different* browsers and *multiple* devices, including your laptop and mobile phone. Second,

if your primary browser or computer crashes, you still have online access to your secure BitWarden vault, which means you can continue to work on a different computer.

Summary

Now you have a better sense of the wide range of data visualization tools we recommend in this book, and how to make wise decisions when choosing among tools in general. Always keep the data story in the forefront of your mind, since the tools are simply means to help you achieve that end. The next chapter is designed to strengthen your skills regarding the most common tool in our data visualization toolkit: spreadsheets.

Chapter 3

Strengthen Your Spreadsheet Skills

Before we begin to design data visualizations, it's important to make sure our spreadsheet skills are up to speed. While teaching this topic, we've heard many people describe how they "never really learned" how to use spreadsheet tools as part of their official schooling or workplace training. But spreadsheet skills are vital to learn, not only as incredible time-savers for tedious tasks, but more importantly, to help us discover the stories buried inside our data.

The interactive charts and maps that we'll construct later this book are built on data tables, which we typically open with spreadsheet tools, such as Google Sheets, LibreOffice, or Microsoft Excel. Spreadsheets typically contain columns and rows of numerical or textual data, as shown in Figure 3.1. The first row often contains headers, meaning labels describing the data in each column. Also, columns are automatically labeled with letters, and rows with numbers, so that every cell or box in the grid can be referenced, such as C2. When you click on a cell, it may display a formula that automatically runs a calculation with references other cells. Formulas always begin with an equal sign, and may simply add up other cells (such as `=C2+C3+C4`), or may contain a function that performs a specific operation (such as calculating the average of a range of cells: `=average(C2:C7)`). Some spreadsheet files contain multiple sheets (sometimes called workbooks), where each tab across the bottom opens a specific sheet.

In this chapter, we'll start by reviewing basic steps, such as sharing, uploading, geocoding with add-on tools, and collecting data with online forms. Then we'll move on to ways of organizing and analyzing your data, such as sorting and filtering, calculating with formulas, and summarizing with pivot tables. Finally, we'll examine ways to connect different sheets, such as matching columns with lookup tables, and relational databases. We illustrate all of these methods with beginner-level users in mind, meaning they do not require any prior background.

The screenshot shows a spreadsheet interface with a table of data. The table has columns labeled A through E. Row 1 contains the header labels: Name, Location, Experience, Years of school, and Occupation. Rows 2 through 7 contain data points. Row 8 is the formula row, showing the result of the formula =average(C2:C7) in cell C8, which is highlighted with a blue border. The formula bar at the top also displays =average(C2:C7). A red annotation 'Active cell (see formula at top)' is overlaid on the cell C8. The bottom navigation bar includes icons for '+', '≡', 'data', 'notes', and 'Tabs for multiple sheets'.

	A	B	C	D	E
1	Name	Location	Experience	Years of school	Occupation
2	Jack	Hartford, Connecticut	4	20	educator
3	Anthony	Juba, South Sudan	1	16	non-profit org
4	Emily	Boston, MA	2	16	non-profit org
5	Hayat	Pakistan	1	16	information technology
6	Ignacio	Buenos Aires, Argentina	3	16	for-profit business
7	Carly	Montreal	2	20	student
8			2.17	Active cell (see formula at top)	
9					

Figure 3.1: Screenshot of a typical spreadsheet, with headers, tabs, and the active cell displaying a formula.

We'll practice several of these skills using sample data that may interest you, because it includes people like you. So far over 3,000 readers of this book have responded to a quick public survey about their general location, prior level of experience and education, and goals for learning data visualization. If you haven't already done so, fill out the quick survey form to contribute your own response, and also to give you a better sense of how the questions were posed, then see the results in the public sample dataset.

If you want to learn ways to make your computer do more of the tedious data preparation work for you, this chapter is definitely for you. Or if you already feel very familiar with spreadsheets, you should at least skim this chapter, and perhaps you'll learn a trick or two that will help you to create charts and maps more efficiently later in the book.

Select your Spreadsheet Tools

Which spreadsheet tools should you use? As we describe in more detail in Chapter 2: Choose Tools to Tell Your Story, the answer depends on how you respond to different questions about your work. First, is your data public or private? If private, consider using a downloadable spreadsheet tool that runs on your computer, to reduce the risk of an accidental data breach that might happen when using an online spreadsheet tool that automatically stores your data in the cloud. Second, will you be working solo or with other people? For collaborative projects, consider using an online spreadsheet tool that's designed to allow other team members to simultaneously view or edit data. Third, do you need to import or export data in any specific format (which we'll describe in the next section), such as Comma Separated Values (CSV)? If yes, then choose a spreadsheet tool that supports that format. Finally, do you prefer a free

tool, or are you willing to pay for it, or donate funds to support open-source development?

Here's how three common spreadsheet tools compare on these questions:

- Google Sheets is a free online spreadsheet tool that works in any modern web browser, and automatically stores your data in the cloud. While data you upload is private by default, you can choose to share it with specific individuals or anyone on the internet, and allow them to view or edit for real-time collaboration, similar to Google Documents. Google Sheets also imports and exports data in CSV, ODS, Excel, and other formats. You can sign up for a free personal Google Drive account with the same username as your Google Mail account, or create a separate account under a new username to reduce Google's invasion into your private life. Another option is to pay for a Google Workspace business account subscription (formerly known as G Suite), which offers nearly identical tools, but with sharing settings designed for larger organizations or educational institutions.
- LibreOffice is a free downloadable suite of tools, including its Calc spreadsheet, available for Mac, Windows, and Linux computers, and is an increasingly popular alternative to Microsoft Office. When you download LibreOffice, its sponsor organization, The Document Foundation, requests a donation to continue its open-source software development. The Calc spreadsheet tool imports and exports data in its native ODS format, as well as CSV, Excel, and others. While an online collaborative platform is under development, it is not yet available for broad usage.
- Microsoft Excel is the spreadsheet tool in the Microsoft Office suite, which is available in different versions, though commonly confused as the company has changed its product names over time. A paid subscription to Microsoft 365 provides you with two versions: the full-featured downloadable version of Excel (which is what most people mean when they simply say "Excel") for Windows or Mac computers and other devices, and access to a simpler online Excel through your browser, including file sharing with collaborators through Microsoft's online hosting service. If you do not wish to pay for a subscription, anyone can sign up for a free version of online Excel at Microsoft's Office on the Web, but this does *not* include the full-featured downloadable version. The online Excel tool has limitations. For example, neither the paid nor the free version of online Excel allows you to save files in the single-sheet generic Comma Separated Values (.csv) format, an important feature required by some data visualization tools in later chapters of this book. You can only export to CSV format using the downloadable Excel tool, which is now available only with a paid Microsoft 365 subscription.

Deciding which spreadsheet tools to use is not a simple choice. Sometimes our decisions change from project to project, depending on costs, data formats, pri-

vacy concerns, and the personal preferences of any collaborators. Occasionally we've also had co-workers or clients specifically request that we send them non-sensitive spreadsheet data attached to an email, rather than sharing it through a spreadsheet tool platform that was designed for collaboration. So it's best to be familiar with all three commonly-used spreadsheet tools above, and to understand their respective strengths and weaknesses.

In this book, we primarily use Google Sheets for most of our examples. All of the data we distribute through this book is public. Also, we wanted a spreadsheet tool designed for collaboration, so that we can share links to data files with readers like you, so that you can view our original version, and either make a copy to edit in your own Google Drive, or download in a different format to use in LibreOffice or Excel. Most of the spreadsheet methods we teach look the same across all spreadsheet tools, and we point out exceptions when relevant.

Sidebar: Common data formats

Spreadsheet tools organize data in different formats. When you download spreadsheet data to your computer, you typically see its filename, followed by a period and a 3- or 4-character abbreviated extension, which represents the data format, as shown in Figure 3.2. The most common data formats we use in this book are:

- `.csv` means Comma Separated Values, a generic format for a single sheet of simple data, which saves no formulas nor styling.
- `.ods` means OpenDocument Spreadsheet, a standardized open format that saves multi-tabbed sheets, formulas, styling, etc.
- `.xlsx` or the older `.xls` means Excel, a Microsoft format that supports multi-tabbed sheets, formulas, styling, etc.
- `.gsheet` means Google Sheets, which also supports multi-tabbed sheets, formulas, styling, etc., but you don't normally see these on your computer because they are primarily designed to exist online.



Figure 3.2: Three data formats commonly seen on your computer—`csv`, `ods`, and `xlsx`—when displayed properly in the Mac Finder.

Warning: Several tools in this book may not work properly on a Mac computer that does not display the filename extensions, meaning the abbreviated file

format after the period, such as `data.csv` or `map.geojson`. The Mac operating system hides these by default, so you need to turn them on by going to Finder > Preferences > Advanced, and check the box to *Show all filename extensions*, as shown in Figure 3.3.



Figure 3.3: On a Mac, go to *Finder-Preferences-Advanced* and check the box to *Show all filename extensions*.

Download to CSV or ODS Format

In Chapter 2: Choose Tools to Tell Your Story, you learned why we recommend software that supports portability, so you can migrate data to other platforms as technology evolves. Never upload important data into a tool that doesn't allow you to easily get it back out. Ideally, spreadsheet tools should allow you to export your work in generic or open-data file formats, such as Comma Separated Values (CSV) and OpenDocument Spreadsheet (ODS), to maximize your options to migrate to other platforms.

Warning: If you're working in any spreadsheet with multiple tabs and formulas, a CSV export will save only the *active sheet* (meaning the one you're currently viewing), and only the *data* in that sheet (meaning that if you inserted formulas to run calculations, only the results would appear, not the formulas). Later in this book you may need to create a CSV file to import into a data visualization tool, so if the source was a multi-tabbed spreadsheet with formulas, keep track of the original.

One reason we feature Google Sheets in this book is because it exports data in several common formats. To try it, open this Google Sheets sample data file in a new tab, and go to *File > Download* to export in CSV format (for only the data in the active sheet) or ODS format (which keeps data and most formulas in multi-tab spreadsheets), or other formats such as Excel, as shown in Figure 3.4. Similarly, in the downloadable LibreOffice and its Calc spreadsheet tool, select *File > Save As* to save data in its native ODS format, or to export to CSV, Excel, or other formats.

But exporting data can be trickier in Microsoft Excel. Using the online Excel tool in your browser (either the free or paid version), you *cannot* save files in the generic single-sheet CSV format, a step required by some data visualization

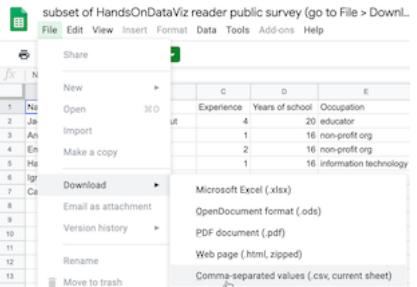


Figure 3.4: In Google Sheets, go to *File - Download As* to export data in several common formats.

tools in later chapters of this book. Only the downloadable Excel tool (which now requires a paid subscription) will export in CSV format, a step required by some data visualization tools in later chapters of this book. And when using the downloadable Excel tool to save in CSV format, the steps sometimes confuse people. First, if you see multiple CSV options, choose *CSV UTF-8*, which should work best across different computer platforms. Second, if your Excel workbook contains multiple sheets or formulas, you may see a warning that it cannot be saved in CSV format, which only saves data (not formulas) contained in the active sheet (not all sheets). If you understand this, click *OK* to continue. Third, on the next screen, Excel may warn you about “Possible data loss” when saving an Excel file in CSV format, for reasons described above. Overall, when working with the downloadable Excel tool, first save the full-version of your Excel file in XLSX format before exporting a single sheet in CSV format.

Once you’ve learned how to export your spreadsheet data into an open format, you’re ready to migrate it into other data visualization tools or platforms that we’ll introduce in later chapters of this book. Data portability is key for ensuring that your charts and maps will last well into the future.

Make a Copy of a Google Sheet

In this book we provide several data files using Google Sheets. Our links point to the online files, and we set the sharing settings to allow anyone to view—but not edit—the original version. This allows everyone to have access to the data, but no one can accidentally modify the contents. In order for you to complete several exercises in this chapter, you need to learn how to make your own copy of our Google Sheets—which you can edit—without changing our originals.

1. Open this Google Sheet of Hands-On Data Visualization reader public survey responses in a new tab in your browser. We set it to “View only”

so that anyone on the internet can see the contents, but not edit the original file. Learn more about the survey at the top of the chapter.

2. Sign in to your Google account by clicking the blue button in the upper-right corner.
3. Go to *File > Make a Copy* to create a duplicate of this Google Sheet in your Google Drive, as shown in Figure 3.5. You can rename the file to remove “Copy of...”.

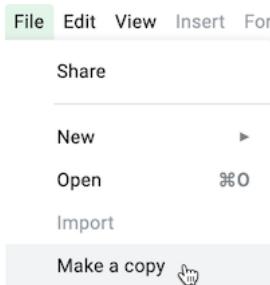


Figure 3.5: Go to *File - Make a Copy* to create your own version of this Google Sheet.

4. To keep your Google Drive files organized, save them in folders with relevant names to make them easier to find. For example, you can click the *My Drive* button and the *New folder* button to create a folder for your data, before clicking *OK*, as shown in Figure 3.6.



Figure 3.6: Click the *My Drive* and *New folder* buttons to save your work in a folder.

Your copy of the Google Sheet will be private to you only, by default. In the next section we'll learn about different options for sharing your Google Sheet data with others.

Share Your Google Sheets

If you're working on a collaborative project with other people, Google Sheets offers several ways to share your data online, even with people who do not own a Google account. When you create a new Sheet, its default setting is private, meaning only you can view or edit its contents. In this section, you'll learn how to expand those options using the *Share* button.

1. Log into your Google Drive account, click the *New* button, select *Google Sheets*, and create a blank spreadsheet. You will need to name your file to proceed with next steps.
2. Click the *Share* button in the upper-right corner, and your options will appear on the *Share with people and groups* screen, as shown in Figure 3.7.
3. In the top half of the screen, you can share access with specific individuals by typing their Google usernames into the *Add people and groups* field. For each person or group you add, on the next screen select the drop-down menu to assign them to be *Viewer*, *Commenter*, or *Editor* of the file. Decide if you wish to notify them with a link to the file and optional message.
4. In the lower half of the screen, you can share access more widely by clicking on *Change to anyone with the link*. On the next screen, the default option is to allow anyone who has the link to *View* the file, but you can change this to allow anyone to *Comment* on or *Edit* it. Also, you can click *Copy link* to paste the web address to your data in an email or public website.



Figure 3.7: Click the *Share* button to grant access to individuals (top half) or anyone with the link (bottom half).

Tip: If you don't want to send people a really long and ugly Google Sheet web address such as:

https://docs.google.com/spreadsheets/d/1egX_akJccnCSzdk1aaDdtrEGe5HcaTr1OW-Yf6mJ3Uo

then use a free link-shortening service. For example, by using our free Bitly.com account and its handy Chrome browser extension or Firefox browser extension, we can paste in a long URL and customize the back-end to something shorter, such as bit.ly/reader-survey, as shown in Figure 3.8. If someone else has already claimed your preferred custom name, you'll need to think up a different one. Beware that bit.ly links are case-sensitive, so we prefer to customize the back-end in all lower-case to match the front-end.

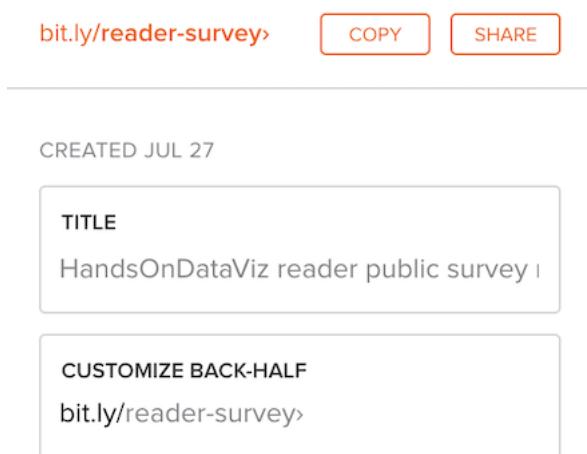


Figure 3.8: Use a free link-shortening service, such as Bitly.com, and customize its back-end.

Now that you have different options for sharing a Google Sheet, let's learn how to upload and convert data from different formats.

Upload and Convert to Google Sheets

We feature Google Sheets in this book partly because it supports data migration, meaning the ability to import and export files in many common formats. But imports work best when you check the *Convert uploads* box, which is hidden inside the Google Drive Settings gear symbol as shown in Figure 3.9. Checking this box automatically transforms Microsoft Excel sheets into Google Sheets format (and also Microsoft Word and PowerPoint files into Google Documents and Slides formats), which allows easier editing. If you don't check this box, then Google will keep your files in their original format, which makes them harder to edit. Google turns off this conversion setting by default on new accounts, but we'll teach you how to turn it on, and the benefits of doing so.

1. Find a sample Excel file you can use on your computer. If you don't have one, open and save to download to your computer this Excel file of a subset of the Hands-On Data Visualization reader public survey responses.
2. Log into your Google Drive account, and click the *Gear symbol* in the upper-right corner, as shown in Figure 3.9, to open the Settings screen. Note that this global *Gear symbol > Settings* appears at Google Drive level, *not* inside each Google Sheet.



Figure 3.9: Click your Google Drive *Gear Symbol - Settings* in the upper-right corner.

3. On the Settings screen, check the box to *Convert uploaded files to Google Docs editor format*, as shown in Figure 3.10, and click *Done*. This turns on the conversion setting globally, meaning it will convert all possible files that you upload in the future—including Microsoft Excel, Word, PowerPoint, and more—unless you turn it off.



Figure 3.10: Inside your Google Drive Settings, check the box to automatically convert all uploads.

4. Upload a sample Excel file from your computer to your Google Drive. Either drag-and-drop it to the desired folder, as shown in Figure 3.11, or use the *New* button and select *File upload*.

If you forget to check the *Convert uploads* box, Google Drive will keep uploaded files in their original format, and display their icons and file name extensions such as `.xlsx` or `.csv`, as shown in Figure 3.12.

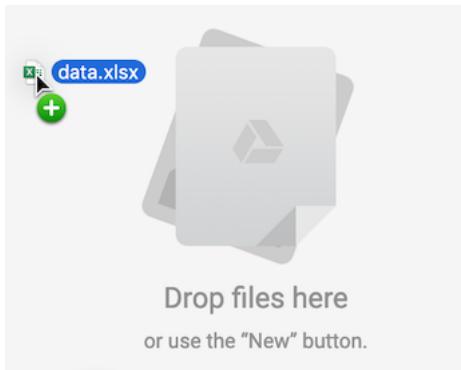


Figure 3.11: Drag-and-drop your sample Excel file into your Google Drive to upload it.

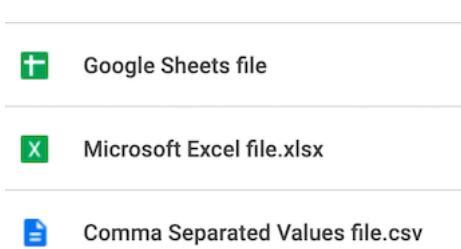


Figure 3.12: If you forget to convert uploads, Google Drive will keep files in their original format with these icons.

Tip: Google Drive now allows you to edit Microsoft Office file formats, but not all features are guaranteed to work across platforms. Also, Google Drive now allows you to convert a specific uploaded Excel file into its Google format by using the *File > Save as Google Sheets* menu. Finally, to convert individual files to your Google Drive, while keeping the global conversion setting off, from inside any Google Sheet you can select *File > Import > Upload*. But we recommend that most people turn on the global conversion setting as described above, except in cases where you intentionally use Google Drive to edit an Excel-formatted file, and understand that some features may not work.

Now that you know how to upload and convert an existing dataset, in the next section you'll learn how to install and use a Google Sheets add-on tool to geocode address data into latitude and longitude coordinates.

Geocode Addresses in Google Sheets

In this section, you'll learn how to geocode data by installing a free Google Sheets add-on tool. This allows you to geocode addresses directly inside your spreadsheet, which will be very useful when using Leaflet map code templates in Chapter 13.

Geocoding means converting addresses or location names into geographic coordinates (or x- and y-coordinates) that can be plotted on a map, as shown in Figure 3.13. For example, the Statue of Liberty in the New York City area is located at *40.69, -74.04*. The first number is the latitude and the second is the longitude. Since the equator is 0 degrees latitude, positive latitude is the northern hemisphere, and negative latitude is in the southern hemisphere. Similarly, the prime meridian is 0 degrees longitude, which passes through Greenwich, England. So positive longitude is east of the meridian, and negative longitude is west, until you reach the opposite side of the globe, roughly near the International Date Line in the Pacific Ocean.

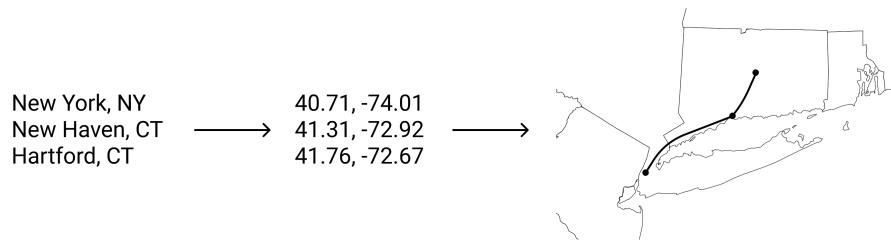


Figure 3.13: To map addresses, you first need to geocode them.

If you have just one or two addresses, you can quickly geocode them with Google Maps. Search for an address, right-click on that point, and select *What's here?*

to reveal a popup window with its latitude and longitude, as shown in Figure 3.14.

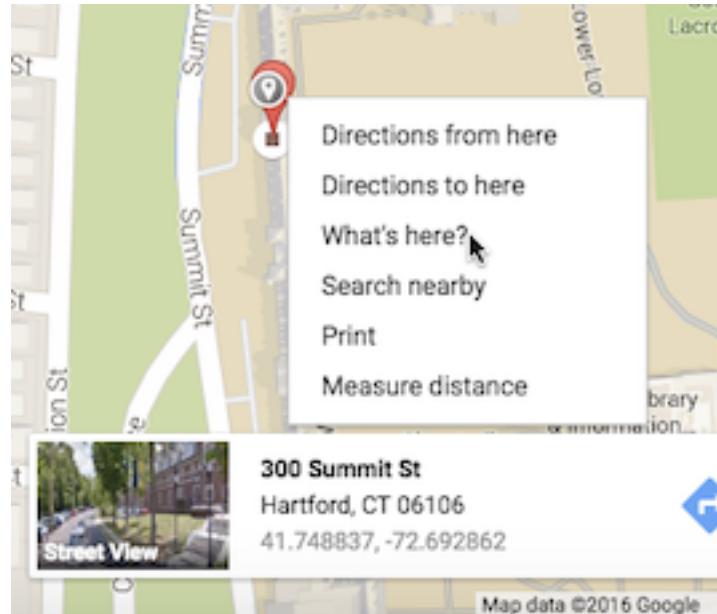


Figure 3.14: To geocode one address, search in Google Maps and right-click *What's here?* to show coordinates.

But what if you need to geocode a dozen or a hundred addresses? To geocode multiple addresses inside your spreadsheet, install a free Google Sheets Add-on called Geocoding by SmartMonkey, created by Xavier Ruiz, the CEO of SmartMonkey, a geographic route-planning company in Barcelona, Spain. Add-ons are created by third-party companies to expand features for Google Sheets, Google Documents, and related tools. Add-ons are verified to meet Google's requirements and distributed through its G Suite Marketplace.

1. Sign into your Google Drive account, go to the Geocoding by SmartMonkey Add-on page, and click the blue button to install it in your Google Sheets. The Add-on will ask for your permission before installing, and if you agree, press *Continue*. In the next window, choose your Google Drive account, and if you agree with the terms, click *Allow* to complete the installation. Google will email you to confirm that you have installed this third-party app with access to your account. You can always review permissions and revoke access in the future, if desired.
2. Go to your Google Drive and create a new Google Sheet. Select the *Add-ons* menu to see the new *Geocoding by SmartMonkey* options, and select *Geocode Details* menu. The Add-on will create a new sheet with sample

data and display results for three new columns: *Latitude*, *Longitude*, and *Address found*, as shown in Figure 3.15. Always review the quality of geocoded results by comparing the *Address found* column to the original *Address* entered.

The screenshot shows a Google Sheets spreadsheet with a menu bar at the top. The 'Add-ons' menu is open, displaying the 'Geocoding by SmartMonkey' option under 'Document add-ons'. A sub-menu for 'Geocoding by SmartMonkey' is also open, showing 'Create Template', 'Geocode', and 'Geocode details' (with a cursor pointing to it). Below the menu, there is a table with four rows of address data. The table has columns for 'Address', 'Country', 'Latitude', 'Longitude', and 'Address found'. The first row is a sample entry: 'Calle de la Electricidad, 49, 28918 Leganés, Madrid' (Country: es) with coordinates 40.3529951, -3.7976303 and 'Calle de la Electricidad, 49, 28918 Leganés, Madrid'. The second row is: 'Av México 95-51 Del Carmen, Coyoacán, 04100 Ciudad de Mexico' (Country: mx) with coordinates 19.3551609, -99.1679037 and 'Av México 95, Del Carmen, Coyoacán, 04100 Ciudad de Mexico'. The third row is: 'Av. Díaz Vélez 4900, Buenos Aires, Argentina' (Country: ar) with coordinates -34.6087952, -58.4360388 and 'Av. Díaz Vélez 4900, Buenos Aires, Argentina'. At the bottom of the sheet, there is a tab labeled 'Geocoding Details'.

Address	Country	Latitude	Longitude	Address found
Calle de la Electricidad, 49, 28918 Leganés, Madrid	es	40.3529951	-3.7976303	Calle de la Electricidad, 49, 28918 Leganés, Madrid
Av México 95-51 Del Carmen, Coyoacán, 04100 Ciudad de Mexico	mx	19.3551609	-99.1679037	Av México 95, Del Carmen, Coyoacán, 04100 Ciudad de Mexico
Av. Díaz Vélez 4900, Buenos Aires, Argentina	ar	-34.6087952	-58.4360388	Av. Díaz Vélez 4900, Buenos Aires, Argentina

Figure 3.15: Select *Add-ons–Geocoding by SmartMonkey–Geocode Details* to display sample data with results for three new columns: *Latitude*, *Longitude*, and *Address found*.

- Paste your own address data to replace the sample data in the sheet, and geocode it as you did in the step above. Follow these guidelines to improve the quality of your results:

- Do not skip any rows in the *Address* column.
- Insert the full address using the format of the national postal service of the country where it is located. Separate terms with spaces.
- You can leave the *Country* column blank, but its default value is the United States. To specify other nations, use their top-level Internet domain code, such as `es` for Spain.
- If your original data splits street, city, state, and zip code into different columns, see how to Combine Data into One Column in Chapter 5: Clean Up Messy Data.
- Give the tool time to work. For example, if you enter 50 addresses, expect to wait at least 15 seconds for your geocoded results.
- Always inspect the quality of your results, and never assume that geocoding services from any provider are accurate.

If you need a faster geocoding service for US addresses, which can handle up to 10,000 requests in one upload, see bulk geocoding with the US Census in Chapter 14: Transform Your Map Data.

Now that you know how to use a Google Sheets Add-on to geocode addresses, in the next section you will learn how to collect data using an online form, and access it as a spreadsheet.

Collect Data with Google Forms

At the top of this chapter, we invited you and other readers of this book to fill out a quick online survey, which publicly shares all of the responses in a sample dataset, so that we can learn more about people like you, and to continue to make revisions to match your expectations. In this section, you'll learn how to create your own online form and link the results to a live Google Sheet.

Inside your Google Drive account, one tool that's often overlooked is Google Forms, which is partially hidden under *New > More > Google Forms*, as shown in Figure 3.16.

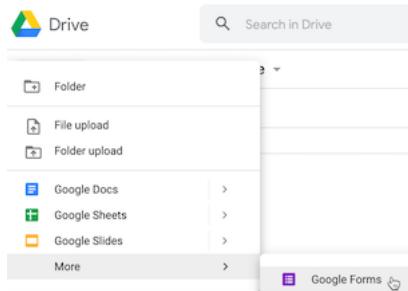


Figure 3.16: The Google Forms tool is partially hidden in the Google Drive *New - More* menu.

The Google Forms *Questions* tab allows you to design questions with different types of responses: short- and paragraph-length answers, multiple choice, checkboxes, file uploads, etc., as shown in Figure 3.17. Furthermore, Google Forms attempts to interpret questions you enter in order to predictively assign them to a type.

Give each question a very short title, since these will appear as column headers in the linked spreadsheet you'll create further below. If a question needs more explanation or examples, click the three-dot kebab menu in the bottom-right corner to *Show > Description*, which opens a text box where you can type in more details, as shown in Figure 3.18. Also, you can *Show > Response validation*, which requires users to follow a particular format, such as an email address or phone number. Furthermore, you can select the *Required* field to require users to respond to a question before proceeding. See additional options on the Google Forms support page.

To preview how your online will appear to recipients, click the *Eyeball symbol* near the top of the page, as shown in Figure 3.19. When your form is complete, click the *Send* button to distribute it via email, a link, or to embed the live form as an iframe on a web page. Learn more about the latter option in Chapter 10: Embed on the Web.

The Google Forms *Responses* tab will show individual results you receive, and

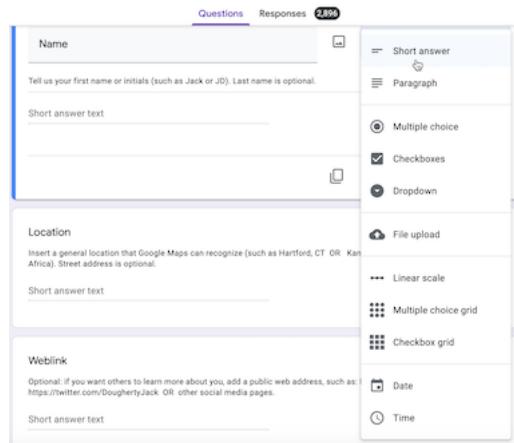


Figure 3.17: The Google Forms *Questions* tab allows you to designate different types of responses.

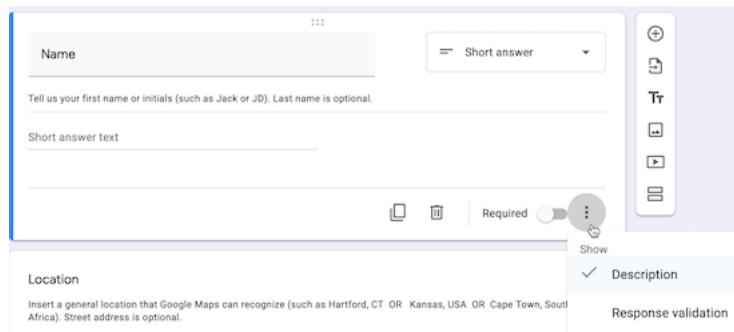


Figure 3.18: Click the three-dot kebab menu to *Show - Description* to add details for any question.



Figure 3.19: Click the *Eyeball symbol* to preview your form.

also includes a powerful button to open the data in a linked Google Sheet, as shown in Figure 3.20.

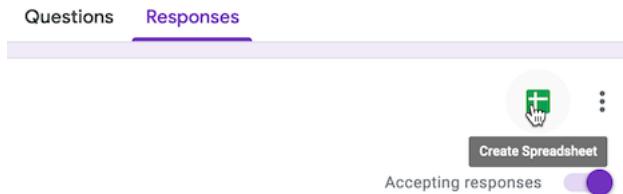


Figure 3.20: The Google Forms *Responses* tab includes a button to open results in a linked Google Sheet.

Now that you've learned how to collect data with an online form and linked spreadsheet, the next two sections will teach you how to sort, filter, and pivot tables to begin analyzing their contents and the stories they reveal.

Sort and Filter Data

Spreadsheet tools help you to dig deeper into your data and raise the stories you find to the surface. A basic step in organizing your data is to *sort* a table by a particular column, to quickly view its minimum and maximum values, and the range that lies in between. A related method is to *filter* an entire table to display only rows that contain certain values, to help them stand out for further study among all of the other entries. Both of these methods become more powerful when your spreadsheets contain hundreds or thousands of rows of data. To learn how to sort and filter, let's explore the reader survey sample dataset we described at the top of the chapter.

1. Open this Google Sheet of Hands-On Data Visualization reader public survey responses in a new tab in your browser.
2. Login to your Google Sheets account, and go to *File > Make a Copy* to create your own version that you can edit.
3. Before sorting, click the upper-left corner of the sheet to select all cells, as shown in Figure 3.21. When the entire sheet becomes light blue, and all of the alphabetical column and numerical row headers become dark grey, this confirms you've selected all cells.

Warning: If you forget to select all cells, you might accidentally sort one column independently of the others, which will scramble your dataset and make it meaningless. Always select all cells before sorting!

	A	B	C
1	Timestamp	Name	Location
2	1/14/2017 11:49:02	Jack	Hartford, CT
3	2/4/2017 9:02:39	Ania	Needham, MA
4	2/8/2017 14:35:56	Devan Suggs	Hartford, CT
5	2/8/2017 17:42:02	Alex	Chicago, IL
6	2/8/2017 21:49:00	Nhat Pham	Hanoi, Viet

Figure 3.21: Click the upper-left corner to select all cells before sorting.

4. In the top menu, go to *Data > Sort Range* to review all of your sort options. In the next screen, check the *Data has header row* box to view the column headers in your data. Let’s sort the *Experience with data visualization* column in ascending order (from A-Z), as shown in Figure 3.22, to display the minimum at the top, the maximum at the bottom, and the range in between.

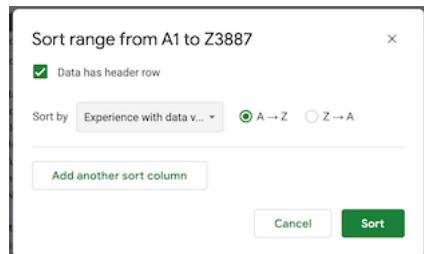


Figure 3.22: Go to *Data - Sort Range*, check the header row box, and sort by *Experience with dataviz* in ascending order.

Scroll through your sorted data and you’ll see that over 1,000 readers rated themselves as beginners (level 1) with data visualization.

Tip: When working with large spreadsheets, you can “freeze” the first row so that column headers will still appear as you scroll downward. In Google Sheets, go to *View > Freeze* and select 1 row, as shown in Figure 3.23. You can also freeze one or more columns to continuously display when scrolling sideways. LibreOffice has a same option to *View > Freeze Rows and Columns*, but Excel has a different option called *Window > Split*.

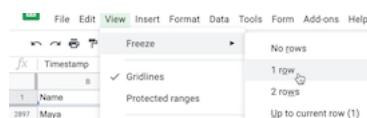


Figure 3.23: In Google Sheets, go to *View - Freeze* to select the number of rows to continuously display when scrolling downward.

5. Now let’s try filtering your sheet. Go to *Data > Create a Filter*, which inserts downward arrows in each column header. Click on the downward

arrow-shaped toggle in the *Occupation* column, and see options to display or hide rows of data. For example, look under *Filter by values*, then click the “Clear” button to undo all options, then click only *educator* to display only rows with that response, as shown in Figure 3.24. Click “OK”.

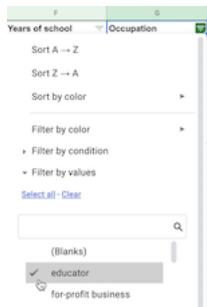


Figure 3.24: Go to *Data - Create a Filter*, click the downward arrow in the *Occupation* column, select only *educator*.

Now your view of reader responses is sorted by experience, and filtered to show only educators. Scroll through their one-sentence goals for learning about data visualization. How do they compare to your own goals? In the next section, we’ll learn how to start analyzing your data with simple formulas and functions.

Calculate with Formulas

Spreadsheet tools can save you lots of time when you insert simple formulas and functions to automatically perform calculations across entire rows and columns of data. Formulas always begin with an equal sign, and may simply add up other cells (such as `=C2+C3+C4`), or may contain a function that performs a specific operation (such as calculating the sum of a range of cells: `=SUM(C2:C100)`). In this section you’ll learn how to write two formulas with functions: one to calculate an average numeric value, and another to count the frequency of a specific text response. Once again, let’s learn this skill using the reader survey sample dataset we described at the top of the chapter.

1. Open this Google Sheet of Hands-On Data Visualization reader public survey responses in a new tab in your browser.
2. Log into your Google Drive account, and go to *File > Make a Copy* to edit your own version.
3. Add a blank row immediately below the header to make space for our calculations. Right-click on row number 1 and select *Insert 1 below* to add a new row, as shown in Figure 3.25.

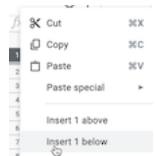


Figure 3.25: Right-click on row number 1 and select *Insert 1 below*.

4. Let's calculate the average level of reader experience with data visualization. Click on cell E2 in the new blank row you just created, and type an equal symbol (=) to start a formula. Google Sheets will automatically suggest possible formulas based on the context, and you can select one that displays the average for current values in the column, such as `=AVERAGE(E3:E2894)`, then press *Return* or *Enter* on your keyboard, as shown in Figure 3.26.

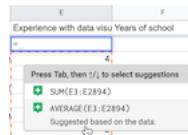


Figure 3.26: Type `=` to start a formula and select the suggestion for average, or type it directly in with the correct range.

Since our live spreadsheet has a growing number of survey responses, you will have a larger number in the last cell reference to include all of the entries in your version. Currently, the average level of reader experience with data visualization is around 2 on a scale from 1 (beginner) to 5 (professional), but this may change as more readers fill out the survey. Note that if any readers leave this question blank, spreadsheet tools ignore empty cells when performing calculations.

Tip: In Google Sheets, another way to write the formula above is `=AVERAGE(E3:E)`, which averages *all* values in column E, beginning with cell E3, without specifying the last cell reference. Using this syntax will keep your calculations up-to-date if more rows are added, but it does *not* work with LibreOffice or Excel.

5. Part of the magic of spreadsheets is that you can use the built-in hold-and-drag feature to copy and paste a formula across other columns or rows, and it will automatically update its cell references. Click in cell E2, and then press and hold down on the blue dot in the bottom-right corner of that cell, which transforms your cursor into a crosshair symbol. Drag your cursor to cell F2 and let go, and show in Figure 3.27. The formula will be automatically pasted and updated for the new column to `=AVERAGE(F3:F2894)` or `AVERAGE(F3:F)`, depending on which way you

entered it above. Once again, since this is a live spreadsheet with a growing number of responses, your sheet will have a larger number in the last cell reference.

E	F		
Experience with data		Years of school	Occ
	2.090909091	4	20
	2.090909091	4	20
	2.090909091	17.80737082	4

Figure 3.27: Click on the blue bottom-right dot in cell E2, then hold-and-drag your crosshair cursor in cell F2, and let go to automatically paste and update the formula.

6. Since the *Occupation* column contains a defined set of text responses, let's use a different function to count them using an *if statement*, such as the number of responses if a reader listed "educator". Click in cell G2 and type the equal symbol (=) to start a new formula. Google Sheets will automatically suggest possible formulas based on the context, and you can select one that displays the count if the response is *educator* for current values in the entire column. You can directly type in the formula `=COUNTIF(G3:G2894, "=educator")`, where your last cell reference will be a larger number to reflect all of the rows in your version, or type in the Google Sheets syntax `=COUNTIF(G3:G, "=educator")` that runs the calculation on the entire column without naming a specific endpoint, as shown in Figure 3.28.

G
Occupation
<code>=COUNTIF(G3:G, "educator")</code>
educator
student

Figure 3.28: Select or enter a formula that counts responses if the entry is *educator*.

Spreadsheet tools contain many more functions to perform numerical calculations and also to modify text. Read more about functions in this support pages for Google Sheets, LibreOffice, or Microsoft Excel support page.

See additional spreadsheet skills in later chapters of the book, such as how to find and replace with blank, split data into separate columns, and combine data into one column in Chapter 5: Clean Up Messy Data. See also how to pivot address points into polygons and how to normalize data in Chapter 14: Transform Your Map Data.

Now that you've learned how to count one type of survey response, the next section will teach you how to regroup data with pivot tables that summarize all responses by different categories.

Summarize Data with Pivot Tables

Pivot tables are another powerful feature built into spreadsheet tools to help you reorganize your data and summarize it in a new way, hence the name "pivot." Yet pivot tables are often overlooked by people who were never taught about them, or have not yet discovered how to use them. Let's learn this skill using the reader survey sample dataset we described at the top of the chapter. Each row represents an individual reader, including their occupation and prior level of experience with data visualization. You'll learn how to "pivot" this individual-level data into a new table that displays the total number of reader responses by two categories: occupation and experience level.

1. Open this Google Sheet of Hands-On Data Visualization reader public survey responses in a new tab in your browser. Log into your Google Drive account, and go to *File > Make a Copy* to edit your own version.
2. Or, if you have already created your own copy for the prior section on Formulas and Functions, delete row 2 that contains our calculations, because we don't want those getting mixed into our pivot table.
3. Go to *Data > Pivot Table*, and on the next screen, select *Create* in a new sheet, as shown in Figure 3.29. The new sheet will include a Pivot Table tab at the bottom.

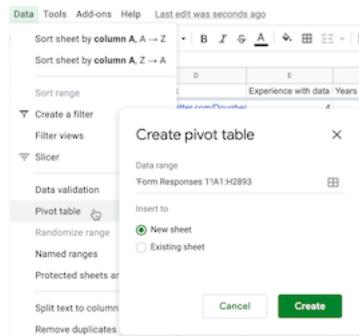


Figure 3.29: Go to *Data - Pivot Table*, and create in a new sheet.

4. In the *Pivot table editor* screen, you can regroup data from the first sheet by adding rows, columns, and values. First, click the Rows *Add* button

and select *Occupation*, which displays the unique entries in that column, as shown in Figure 3.30.

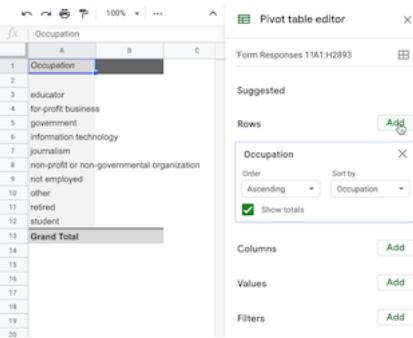


Figure 3.30: In the *Pivot table editor*, click the Rows *Add* button and select *Occupation*.

5. Next, to count the number of responses for each entry, click the Values *Add* button and select *Occupation* again. Google Sheets will automatically summarize the values by *COUNTA*, meaning it displays the frequency of each textual response, as shown in Figure 3.31.

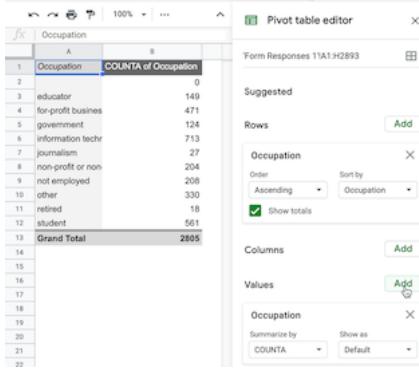


Figure 3.31: In the *Pivot table editor*, click the Values *Add* button and select *Occupation*.

Currently, the top three occupations listed by readers are information technology, for-profit business, and student. Since this is a live spreadsheet, these rankings may change as more readers respond to the survey.

6. Furthermore, you can create a more advanced pivot cross-tabulation of occupation and experience among reader responses. Click on the *Columns* button to add *Experience with data visualization*, as shown in Figure 3.32.

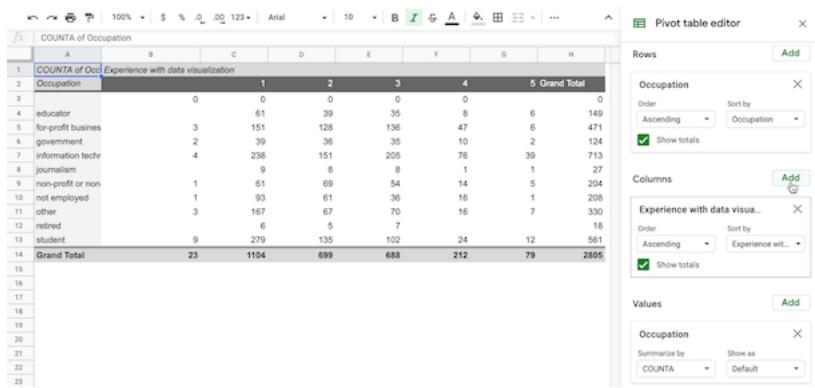


Figure 3.32: In the *Pivot table editor*, click the Columns *Add* button and select *Experience with data visualization*.

To go one step further, *Filter* the data to limit the pivot table results by another category. For example, in the drop-down menu, you can click the Filters *Add* button, select *Years of school*, then under *Filter by values* select *Clear*, then check *20* to display only readers who listed 20 or more years.

Deciding how to add *Values* in the *Pivot table editor* can be challenging, because there are multiple options to summarize the data, as shown in Figure 3.33. Google Sheets will offer its automated guess based on the context, but you may need to manually select the best option to represent your data as desired. Three of the most common options to summarize values are:

- SUM: the total value of numeric responses (What is the total years of schooling for readers?)
- COUNT: frequency of numeric responses (How many readers listed 20 years of schooling?)
- COUNTA: frequency of text responses (How many readers listed occupation as “educator”)

Although Google Sheets pivot tables display raw numbers by default, under the *Show as* drop-down menu you can choose to display them as percentages of the row, of the column, or of the grand total.

While designing pivot tables may look differently across other spreadsheet tools, the concept is the same. Learn more about how pivot tables work in the support pages for Google Sheets or LibreOffice or Microsoft Excel. Remember that you can download the Google Sheets data and export to ODS or Excel format to experiment with pivot tables in other tools.

Now that you've learned how to regroup and summarize data with pivot tables, in the next section you'll learn a related method to connect matching data columns across different spreadsheets using VLOOKUP.

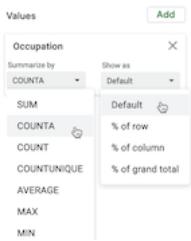


Figure 3.33: In the *Pivot table editor*, see multiple options to summarize *Values*.

Match Columns with VLOOKUP

Spreadsheet tools also allow you to “look up” data in one sheet and automatically find and paste matching data from another sheet. This section introduces the VLOOKUP function, where the “V” stands for “vertical,” meaning matches across columns, which is the most common way to look up data. You’ll learn how to write a function in one sheet that looks for matching cells in select columns in a second sheet, and pastes the relevant data into a new column in the first sheet. If you’ve ever faced the tedious task of manually looking up and matching data between two different spreadsheets, this automated method will save you lots of time.

Here’s a scenario that illustrates why and how to use the VLOOKUP function. Figure 3.34 shows two different sheets with sample data about food banks that help feed hungry people in different parts of the US, drawn from Feeding America: Find Your Local Food Bank. The first sheet lists individual people at each food bank, the second sheet lists the address for each food bank, and the two share a common column named *organization*. Your goal is to produce one sheet that serves as a mailing list, where each row contains one individual’s name, organization, and full mailing address. Since we’re using a small data sample to simplify this tutorial, it may be tempting to manually copy and paste in the data. But imagine an actual case that includes over 200 US food banks and many more individuals, where using an automated method to match and paste data is essential.

1. Open this Google Sheet of Food Bank sample names and addresses in a new browser tab. Log into your Google Drive, and go to *File > Make a Copy* to create your own version that you can edit.

We simplified this two-sheet problem by placing both tables in the same Google Sheet. Click on the first tab, called *names*, and the second tab, called *addresses*. In the future, if you need to move two separate Google Sheets into the same file, go to the tab of one sheet, right-click the tab to *Copy to > Existing spreadsheet*, and select the name of the other sheet.

The screenshot shows two Google Sheets tabs: 'names' and 'addresses'. The 'names' sheet contains the following data:

	A	B
1	name	organization
2	Denise B.	Central Texas Food Bank
3	Derrick C.	Central Texas Food Bank
4	Eric S.	Arkansas Food Bank
5	Ginette B.	Utah Food Bank
6	Greg F.	Arkansas Food Bank
7	Kent L.	Utah Food Bank
8	Mark J.	Central Texas Food Bank
9	Rhonda S.	Arkansas Food Bank
10	Sarah R.	Arkansas Food Bank
11	Scott W.	Utah Food Bank
...		

The 'addresses' sheet contains the following data:

	A	B	C	D	E
1	organization	street	city	state	zip
2	Arkansas Food Bank	4301 W 65th St	Little Rock	AR	77209
3	Central Texas Food Bank	6500 Metropolis Dr	Austin	TX	78744
4	Utah Food Bank	3150 South 900 West	Salt Lake City	UT	84119
5					
6					
7					
8					
9					
10					
11					
...					

Below the sheets are tabs labeled 'names', 'addresses', '+', 'names', 'addresses', and 'source'.

Figure 3.34: Your goal is to create one mailing list that matches individual names and organizations on the left sheet with their addresses on the right sheet.

2. In your editable copy of the Google Sheet, the *names* tab will be our destination for the mailing list we will create. Go to the *addresses* sheet, copy the column headers for *street - city - state - zip*, and paste them into cells C1 through F1 on the *names* sheet, as shown in Figure 3.35. This creates new column headers where our lookup results will be automatically pasted.

The screenshot shows the 'names' sheet with the following data:

	A	B	C	D	E	F
1	name	organization	street	city	state	zip
2	Denise B.	Central Texas Food Bank				
3	Derrick C.	Central Texas Food Bank				
4	Eric S.	Arkansas Food Bank				
5	Ginette B.	Utah Food Bank				

Below the sheets are tabs labeled '+', 'names', 'addresses', and 'source'.

Figure 3.35: Paste the last four column headers from the *addresses* sheet into the *names* sheet.

3. In the *names* sheet, click in cell C2 and type =VLOOKUP, and Google Sheets will suggest that you complete the full formula in this format:

```
VLOOKUP(search_key, range, index, [is_sorted])
```

Here's what each part means:

- `search_key` = The cell in 1st sheet you wish to match.
- `range` = At least two columns in the 2nd sheet to search for your match and desired result.
- `index` = The column in the 2nd sheet range that contains your desired result, where 1 = first column, 2 = second column, etc.
- `[is_sorted]` = Enter `false` to find exact matches only, which makes sense in this case. Otherwise, enter `true` if the first column of the 2nd sheet range is sorted and you will accept the closest match, even if not an exact one.

4. One option is to directly type this formula into cell C2, using comma separators: `=VLOOKUP(B2, 'addresses'!A:E, 2, false)`. Another option is to click on the *VLOOKUP Vertical lookup* grey box that Google Sheets suggests, and click on the relevant cells, columns, and sheets for the formula to be automatically entered for you, as shown in Figure 3.36. What's new here is that this formula in the *names* sheet refers to a range of columns A to E in the *addresses* sheet. Press *Return* or *Enter* on your keyboard.

A	B	C	D	E	F
1 name	organization	street	city	state	zip
2 Denise B.	Central Texas Food Bank	<code>=VLOOKUP(B2, 'addresses'!A:E, 2, false)</code>			
3 Derrick C.	Central Texas Food Bank				
4 Eric S.	Arkansas Food Bank				
5 Ginette B.	Utah Food Bank				

A	B	C	D	E
1 organization	street	city	state	zip
2 Arkansas Food Bank	4301 W 65th St	Little Rock	AR	77209
3 Central Texas Food Bank	6500 Metropolis Dr	Austin	TX	78744
4 Utah Food Bank	3150 South 900 West	Salt Lake City	UT	84119
5				

Figure 3.36: The VLOOKUP formula in cell C2 of the *names* sheet (top) searches for matches across columns A to E in the *addresses* sheet (bottom).

Let's break down each part of the formula you entered in cell C2 of the *names* sheet:

- `B2` = The search_key: the cell in the *organization* column you wish to match in the *names* sheet
- `'addresses'!A:E` = The range where you are searching for your match and results across columns A to E in the *addresses* sheet.
- `2` = The index, meaning your desired result appears in the 2nd column (*street*) of the range above.
- `false` = Find exact matches only.

5. After you enter the full VLOOKUP formula, it will display the exact match for the first organization, the Central Texas Food Bank, whose address is 6500 Metropolis Dr. Click and hold down on the blue dot in the bottom-right corner of cell C2, and drag your crosshair cursor across columns D to F and let go, which will automatically paste and update the formula for the city, state, and zip columns, as shown in Figure 3.37.

A	B	C	D	E	F
1 name	organization	street	city	state	zip
2 Denise B.	Central Texas Food Bank	6500 Metropolis Dr	Austin	TX	78744
3 Derrick C.	Central Texas Food Bank				

Figure 3.37: Click on cell C2, then hold-and-drag the bottom-right blue dot across columns D to F, which automatically pastes and updates the formula.

- Finally, use the same hold-and-drag method to paste and update the formula downward to fill in all rows, as shown in Figure 3.38.

	A	B	C	D	E	F
1	name	organization	street	city	state	zip
2	Denise B.	Central Texas Food Bank	6500 Metropolis Dr	Austin	TX	78744
3	Derick C.	Central Texas Food Bank	6500 Metropolis Dr	Austin	TX	78744
4	Eric S.	Arkansas Food Bank	4301 W 65th St	Little Rock	AR	77209
5	Ginette B.	Utah Food Bank	3150 South 900 West	Salt Lake Ct	UT	84119
6	Greg F.	Arkansas Food Bank	4301 W 65th St	Little Rock	AR	77209
7	Kent L.	Utah Food Bank	3150 South 900 West	Salt Lake Ct	UT	84119
8	Mark J.	Central Texas Food Bank	6500 Metropolis Dr	Austin	TX	78744
9	Rhonda S.	Arkansas Food Bank	4301 W 65th St	Little Rock	AR	77209
10	Sarah R.	Arkansas Food Bank	4301 W 65th St	Little Rock	AR	77209
11	Scott W.	Utah Food Bank	3150 South 900 West	Salt Lake Ct	UT	84119
12						

Below the table, there are three tabs labeled "names", "addresses", and "source".

Figure 3.38: Click on cell F2, then hold-and-drag the bottom-right blue dot down to row 11, which automatically pastes and updates the formula.

Warning: If you save this spreadsheet in CSV format, your calculated results will appear in the CSV sheet, but any formulas you created to produce those results will disappear. Always keep track of your original spreadsheet to remind yourself how you constructed formulas.

You've successfully created a mailing list—including each person's name, organization, and full mailing address—using the VLOOKUP function to match and paste data from two sheets. Now that you understand how to use formulas to connect different spreadsheets, the next section will teach you how to manage multiple relationships between spreadsheets with the help of a relational database.

Spreadsheet vs. Relational Database

In the previous section, you learned how the VLOOKUP function can search for matching data in columns across spreadsheets and automatically paste results. Building on that concept, let's distinguish between a spreadsheet and a relational database, and under what circumstances it might be wiser to use the latter.

A spreadsheet is sometimes called a “flat-file database” because all of the records are stored in rows and columns in a single table. For example, if you kept a single spreadsheet of US food bank staff, every row would list an individual person, organization, and addresses, just like the mailing list we created in Figure 3.38 in the prior section on VLOOKUP.

But keeping all of your data in a single spreadsheet can raise problems. For example, it contains lots of duplicated entries. For people who all work at the same food bank, each row contains a duplicate of that organization's address. If an organization moves to a new location, you need to update all of the rows that contain those addresses. Or if two organizations merge together under a

new name, you need to update all of the rows for individuals affected by that change. While keeping all of your information organized in a single spreadsheet initially sounds like a good idea, when your dataset grows in size and internal relationships (such as tracking people who are connected to organizations, etc.), continually updating every row becomes a lot of extra work.

Instead of a single spreadsheet, consider using a relational database, which organizes information into separate sheets (also known as tables), but continually maintains the relevant connections between them. Look back at the two-sheet problem we presented in Figure 3.34 at the beginning of the VLOOKUP section. The first sheet lists individual people at each food bank, the second sheet lists the address for each food bank, and the two sheets share a column named *organization* that shows how they are related. Relational databases can save you time. For example, if you update an organization's address in one sheet, the linked sheet will automatically reflect this change in every row for staff who work at that organization.

Although Google Sheets is a great spreadsheet, it's not a relational database. Instead, consider a better tool such as Airtable, which allows you to create relational databases in your web browser with up to 1,200 free records (or more with the paid version), using existing templates or your own designs. Airtable enables data migration by importing or exporting all records in CSV format, and it also supports real-time editor collaboration with co-workers.

To demonstrate, we imported both of the Google Sheets above into this live Airtable database called Food Banks sample, which anyone with the link can view, but only we can edit. At the top are tabs to view each sheet, named *people* and *food banks*. To transform this into a relational database, we used Airtable settings to link the *organization* column in the *people* sheet to the *food banks* sheet, where the addresses are stored, as shown in Figure 3.39. In our editable version, we double-clicked on the column name, then selected *Link to another record* in the drop-down menu, to connect it to another tab.

In our Airtable sample, click on a linked row to expand it and view related data. For example, if you click and expand on the first row the *people* sheet, their organization's full address appears from the *food banks* sheet, as shown in Figure 3.40. In our editable version, if we update the address for one organization in the *food banks* sheet, it's automatically changed for all employees linked to that organization in the *people* sheet. In addition, Airtable allows you to sort, filter, and create different views of your data that you can share with others, a topic we'll cover in Chapter 10: Embed on the Web. See more about its features in the Airtable Support page.

It's important to understand the conceptual differences between a "flat-file" spreadsheet and a relational database to help you determine when to use one tool versus another. As you've learned in the sections above, spreadsheets are your best choice to begin organizing and analyzing your data, using methods such as sorting, filtering, pivoting, and lookup, to help reveal the underlying

A screenshot of the Airtable interface. The top navigation bar shows two sheets: "people" and "food banks". The "people" sheet is currently selected and displayed as a grid view. A specific row for "Denise B." is selected. A modal dialog box is open over the grid, titled "organization". Inside the dialog, there is a dropdown menu labeled "Link to food banks" which is expanded, showing a list of organization names: "Arkansas Food Bank", "Arkansas Food Bank", and "Utah Food Bank". Below the dropdown, there are two options: "Allow linking to multiple records" (which is checked) and "Limit record selection to a view". At the bottom right of the dialog are "Cancel" and "Save" buttons.

Figure 3.39: In this Airtable sample, we linked the *organization* column in the *people* sheet to the *food banks* sheet.

A screenshot of the Airtable interface. The top navigation bar shows two sheets: "people" and "Food Banks sample". The "Food Banks sample" sheet is currently selected and displayed as a grid view. A specific row for "Denise B." is selected. A detailed view of her record is shown in a sidebar on the right. The sidebar contains fields for "NAME" (Denise B.) and "ORGANIZATION" (Central Texas Food Bank). Below these, a table provides address details: STREET (6500 Metropolis Dr), CITY (Austin), and STATE (TX).

Figure 3.40: In this Airtable demo, click on a row in one sheet to expand and view its linked data in another sheet.

stories that you may wish to visualize. But relational databases are your best choice when maintaining large amounts of data with internal links, like one-to-many relationships, such as an organization with several employees.

Summary

If you're one of the many people who "never really learned" about spreadsheets in school or on the job, or if you've taught yourself bits and pieces along the way, we hope that this chapter has successfully strengthened your skills. All of the subsequent chapters in this book, especially those on designing interactive charts in Chapter 7 and interactive maps in Chapter 8, require a basic level of familiarity with spreadsheets. In addition to serving as incredible time-savers when it comes to tedious data tasks, the spreadsheet tools and methods featured above are designed to help you share, sort, calculate, pivot, and lookup matching data, with the broader goal of visualizing your data stories.

The next chapter describes strategies for finding and questioning your data, particularly on open data sites operated by governmental and non-profit organizations, where you'll also need spreadsheet skills to download and organize public information.

Chapter 4

Find and Question Your Data

In the early stages of a visualization project, we often start with two interrelated issues: *Where can I find reliable data?*, and after you find something, *What does this data truly represent?* If you leap too quickly into constructing charts and maps without thinking deeply about these dual issues, you run the risk of creating meaningless, or perhaps worse, misleading visualizations. This chapter breaks down both of these broad issues by providing concrete strategies to guide your search, understand debates about public and private data, mask or aggregate sensitive data, navigate a growing number of open data repositories, source your data origins, and recognize bad data. Finally, once you've found some files, we propose some ways to question and acknowledge the limitations of your data.

Information does not magically appear out of thin air. Instead, people collect and publish data, with explicit or implicit purposes, within the social contexts and power structures of their times. As data visualization advocates, we strongly favor evidence-based reasoning over less-informed alternatives. But we caution against embracing so-called data objectivity, since numbers and other forms of data are *not* neutral. Therefore, when working with data, pause to inquire more deeply about *Whose stories are told?* and *Whose perspectives remain unspoken?* Only by asking these types of questions, according to *Data Feminism* authors Catherine D'Ignazio and Lauren Klein, will we “start to see how privilege is baked into our data practices and our data products.”¹

¹Catherine D'Ignazio and Lauren F. Klein, *Data Feminism* (MIT Press, 2020), <https://data-feminism.mitpress.mit.edu/>.

Guiding Questions for Your Search

For many people, a data search is simply googling some keywords on the web. Sometimes that works, sometimes not. When that approach flounders, we reflect on the many lessons we've learned about data-hunting while working alongside talented librarians, journalists, and researchers. Collectively, they taught us a set of guiding questions that outline a more thoughtful process about *how to search* for data:

What exactly is the question you're seeking to answer with data?

Literally write it down—in the form of a question, punctuated with a question mark at the end—to clarify your own thinking, and also so that you can clearly communicate it to others who can assist you. All too often, our brains automatically leap ahead to try to identify the *answer*, without reflecting on the best way frame the *question* in a way that does not limit the range of possible outcomes.

Look back at data visualization projects that made a lasting impression on you to identify the underlying question that motivated them. In their coverage of the US opioid epidemic, the *Washington Post* and the West Virginia *Charleston Gazette-Mail* successfully fought a legal battle to obtain a US Drug Enforcement Agency database that the federal government and the drug industry sought to keep secret. In 2019, a team of data journalists published the database with interactive maps to answer one of their central questions: *How many prescription opioid pills were sent to each US county, per capita, and which companies and distributors were responsible?* Their maps revealed high clusters in several rural Appalachian counties that received over 150 opioid pills per resident, on average, each year from 2006 to 2014. Moreover, only six companies distributed over three-quarters of the 100 billion oxycodone and hydrocodone pills across the US during this period: McKesson Corp., Walgreens, Cardinal Health, Amerisource-Bergen, CVS and Walmart.² Even if you're not working with data as large or as controversial as this one, the broader lesson is to clearly identify the question you're seeking to answer.

Also, it's perfectly normal to revise your question as your research evolves. For example, Jack and his students once began a data project by naively asking *What were Connecticut public school test scores in the 1960s?* Soon we discovered that standardized state-level school testing as we know it today did not appear in states like Connecticut until the mid-1980s school accountability movement. Even then, results were not widely visible to the public until newspapers began to publish them once a year in print in the 1990s. Later, real estate firms, school-ratings companies, and government agencies began to publish data continuously on the web as the Internet expanded in the late 1990s and early 2000s. Based on what we learned, we revised our research question to *When and how did*

²“Drilling into the DEA’s Pain Pill Database,” Washington Post, July 16, 2019, <https://www.washingtonpost.com/graphics/2019/investigations/dea-pain-pill-database/>

*Connecticut homebuyers start to become aware of school test scores, and how did these influence the prices they were willing to pay for access to selected public school attendance areas?*³ Be prepared to refine your question when the evidence leads you in a better direction.

What types of organizations may have collected or published the data you seek?

If a governmental organization may have been involved, then at what level: local, regional, state/provincial, national, or international? Which branch of government: executive, legislative, judicial? Or which particular governmental agency might have been responsible for compiling or distributing this information? Since all of these different structures can be overwhelming, reach out to librarians who are trained to work with government documents and databases, often at state government libraries, or at local institutions participating in the US Federal Depository Library Program. Or might the data you seek have been compiled by a non-governmental organization, such as academic institutions, journalists, non-profit groups, or for-profit corporations? Figuring out *which organizations* might have collected and published the data can help point you to the digital or print materials they typically publish, and most appropriate tools to focus your search in that particular area.

What level(s) of data are available?

Is information disaggregated by individual cases or aggregated into larger groups? Librarians can help us to decipher how and why different organizations publish data in different formats. For example, US Census seeks to collect data every ten years about each person residing in the nation, but under the law, this individual-level data is confidential and not released to the public for 72 years. You can look up individual census data for 1940 and earlier decades at the US National Archives and other websites. But the US Census publishes current data for larger areas, such as neighborhood-level block groups, census tracts, cities, and states, by aggregating individual records into data tables, and suppressing small-numbered cells to protect people's privacy. Librarians can help us understand organization's guidelines on when and how they make data available at different levels.

TODO: DECIDE whether to add an example about why, in general, smaller units of data are better than larger ones. Example from Cairo: voting data about entire US states will vary from voting data for smaller geographic units within those states, such as counties, cities, or neighborhood-level voting precincts, for at least two reasons. First, voting patterns vary across smaller geographic units... Second, not all residents vote.... In most cases, use the smallest unit of data you can obtain to allow you to produce more granular interpretations. While you cannot obtain data about how each individual voted, you can obtain data at the precinct, city, or county level inside each

³Jack Dougherty et al., "School Choice in Suburbia: Test Scores, Race, and Housing Markets," *American Journal of Education* 115, no. 4 (August 2009): 523–48, http://digitalrepository.trincoll.edu/cssp_papers/1.

state. However, there are some cases when using a larger unit makes more sense than a smaller unit. For example, US presidential elections are determined by state-level electoral votes (except for Maine and Nebraska?). So if you wish to emphasize patterns in electoral votes, compare data for larger geographic units...

TODO: DECIDE whether to add this interactive map (currently in CT web-only chapter) to illustrate hierarchical relations among geographical census entities for the Hartford region, from state to census block level. <https://github.com/HandsOnDataViz/census-divisions-hartford>

ALSO DECIDE if this list is necessary or whether we could incorporate into the map legend

- State
- County
- County subdivisions (equivalent to Connecticut towns and cities)
- Census tracts (designated areas, roughly 2,500 to 8,000 people)
- Block groups (sub-unit of tract, roughly 600 to 3,000 people)
- Census blocks (sub-unit of block group, but not always a city block)

Have prior publications drawn on similar data, and if so, how can we trace their sources?

Some of our best ideas began when reading an article or book that described its source of evidence, and we imagined new ways to visualize that data. Several times we have stumbled across a data table in a print publication, or perhaps an old web page, which sparked our interest in tracking down a newer version to explore. Even *outdated* data helps by demonstrating how someone or some organization collected it at one point in time. Follow the footnotes to track down its origins. Use Google Scholar and more specialized research databases (ask librarians for assistance if needed) to track down the source of previously-published data. One bonus is that if you can locate more current data, you may be able to design a visualization that compares change over time.

What if no one has collected the data you're looking for?

Sometimes this happens due to more than a simple oversight. In *Data Feminism*, Catherine D'Ignazio and Lauren Klein underscore how issues of data collection “are directly connected to larger issues of power and privilege” by recounting a story about tennis star Serena Williams. When Williams experienced life-threatening complications while giving birth to her daughter in 2017, she called public attention to the way that she, a Black woman, needed to advocate for herself in the hospital. After her experience, she wrote on social media that “Black women are over 3 times more likely than white women to die from pregnancy- or childbirth-related causes,” citing the US Centers for Disease Control and Prevention (CDC). When journalists followed up to investigate further, they discovered the absence of detailed data on maternal mortality, and what

a 2014 United Nations report described as a “particularly weak” aspect of data collection in the US healthcare system. Journalists reported that “there was still no national system for tracking complications sustained in pregnancy and childbirth,” despite comparable systems for other health issues such as heart attacks or hip replacements. Power structures are designed to count people whose lives either are highly valued, or under a high degree of surveillance. D’Ignazio and Klein call on us critically examine these power systems, collect data to counter their effects, and make everyone’s labor in this process more visible.⁴ If no one has collected the data you’re looking for, perhaps you can make valuable steps to publicly recognize the issue and contribute to positive change.

Hunting for data involves much more than googling keywords. Deepen your search by reflecting on the types of questions that librarians, journalists, and other researchers have taught us to ask: What types of organizations might—or might not—have collected the data? At what levels? At any prior point in time? And under what social and political contexts? In the next section, you’ll learn more about related issues to consider over public and private data.

Public and Private Data

When searching for data, you also need to be informed about debates regarding public and private data. Not only do these debates influence the kinds of data you might be able to legally use in your visualizations, but they also raise deeper ethical issues about the extent to which anyone should be able to collect or circulate private information about individuals. This section offers our general observations on these debates, based primarily on our context in the United States. Since we are not lawyers (thank goodness!), please consult with legal experts for advice about your specific case if needed.

The first debate asks: *To what extent should anyone be allowed to collect data about private individuals?* Several critics of “big data” worry that governments are becoming more like a totalitarian “Big Brother” as they collect more data about individual citizens in the digital age. In the United States, concerns mounted in 2013 when whistleblower Edward Snowden disclosed how the National Security Agency conducted global surveillance using US citizen email and phone records provided by telecommunications companies. Shoshana Zuboff, a Harvard Business School professor and author of *The Age of Surveillance Capitalism*, warns of an equal threat posed by corporations that collect and commodify massive amounts of individually-identifiable data for profit.⁵ Due to the rise of digital commerce, powerful technology companies own data that you and others consider to be private:

⁴D’Ignazio and Klein, *Data Feminism*, chap. 1.

⁵Shoshana Zuboff, *The Age of Surveillance Capitalism: The Fight for a Human Future at the New Frontier of Power* (PublicAffairs, 2019), https://www.google.com/books/edition/The_Age_of_Surveillance_Capitalism/IRqrDQAAQBAJ.

- Google knows what words you typed into their search engine, as shown in aggregated form in Google Trends. Also, Google's Chrome browser tracks your web activity through cookies, as described by *Washington Post* technology reporter Geoffrey Fowler.⁶
- Amazon eavesdrops and records your conversations around its Alexa home assistants, as Fowler also documents.⁷
- Facebook follows which friends and political causes you favor, and Fowler also reports how it tracks your off-Facebook activity, such as purchases made at other businesses, to improve its targeted advertising.⁸

Some point out that “big data” collected by large corporations can offer public benefits. For example, Apple shared its aggregated mobility data collected from iPhone users to help public health officials compare which populations stayed home rather than travel during the Covid pandemic. But others point out that corporations are largely setting their own terms for how they collect data and what they can do with it. Although California has begun to implement its Consumer Privacy Act in 2020, which promises to allow individuals the right to review and delete the data that companies collect about them, US state and federal government has not fully entered this policy arena. If you work with data that was collected from individuals by public or private organizations, learn about these controversies to help you make wise and ethical choices on what to include in your visualizations.

The second question is: *When our government collects data, to what extent should it be publicly available?* In the United States, the 1966 Freedom of Information Act and its subsequent amendments have sought to open access to information in the federal government, with the view that increased transparency would promote public scrutiny and pressure on officials to make positive changes. In addition, state governments operate under their own freedom of information laws, sometimes called “open records” or “sunshine laws.” When people say they’ve submitted a “FOI,” it means they’ve sent a written request to a government agency for information that they believe should be public under the law. But federal and state FOIA laws are complex, and courts have interpreted cases in different ways over time, as summarized in the Open Government Guide by the Reporters Committee for Freedom of the Press, and also by the National Freedom of Information Coalition. Sometimes government agencies quickly agree and comply with a FOI request, while other times they may delay or reject it, which may pressure the requester to attempt to resolve the

⁶Geoffrey A. Fowler, “Goodbye, Chrome: Google’s Web Browser Has Become Spy Software,” *Washington Post*, June 21, 2019, <https://www.washingtonpost.com/technology/2019/06/21/google-chrome-has-become-surveillance-software-its-time-switch/>

⁷Geoffrey A. Fowler, “Alexa Has Been Eavesdropping on You This Whole Time,” *Washington Post*, May 6, 2019, <https://www.washingtonpost.com/technology/2019/05/06/alexas-been-eavesdropping-you-this-whole-time/>

⁸Geoffrey A. Fowler, “Facebook Will Now Show You Exactly How It Stalks You — Even When You’re Not Using Facebook,” *Washington Post*, January 28, 2020, <https://www.washingtonpost.com/technology/2020/01/28/off-facebook-activity-page/>

issue through time-consuming litigation. Around the world, over 100 nations have their own version of freedom of information laws, with the oldest being Sweden's 1766 Freedom of the Press Act, but these laws vary widely.

In most cases, individual-level data collected by US federal and state governments is considered private, except in cases where our governmental process has determined that a broader interest is served by making it public. To illustrate this distinction, let's begin with two cases where US federal law protects the privacy of individual-level data:

- Patient-level health data is generally protected under the Privacy Rule of the Health Insurance Portability and Accountability Act, commonly known as HIPAA. In order for public health officials to track broad trends about illness in the population, individual patient data must be aggregated into larger anonymized datasets in ways that protect specific people's confidentiality.
- Similarly, student-level education data is generally protected under the Family Educational Rights and Privacy Act, commonly known as FERPA. Public education officials regularly aggregate individual student records into larger anonymized public datasets to track the broad progress of schools, districts, and states, without revealing individually-identifiable data.

On the other hand, here are three cases where government has ruled that the public interest is served by making individual-level data widely available:

- Individual contributions to political candidates are public information in the US Federal Election Commission database, and related databases by non-profit organizations, such as Follow The Money by the National Institute on Money in Politics and Open Secrets by the Center for Responsive Politics. The latter two sites describe more details about donations submitted through political action committees and controversial exceptions to campaign finance laws. Across the US, state-level political contribution laws vary widely, and these public records are stored in separate databases. For example, anyone can search the Connecticut Campaign Reporting Information System to find donations made by the first author to state-level political campaigns.
- Individual property ownership records are public, and increasingly hosted online by many local governments. A privately-funded company compiled this US public records directory with links to county and municipal property records, where available. For example, anyone can search the property assessment database for the Town of West Hartford, Connecticut to find property owned by the first author, its square footage, and purchase price.

- Individual salaries for officers of tax-exempt organizations are public, which they are required to file on Internal Revenue Service (IRS) 990 forms each year. For example, anyone can search 990 forms on ProPublica’s Nonprofit Explorer, and view the salary and other compensation of the top officers of the first author’s employer and the second author’s alma mater, Trinity College in Hartford, Connecticut.

Social and political pressures are continually changing the boundary over what types of individual-level data collected by government should be made publicly available. For example, the Black Lives Matter movement has gradually made more individual-level data about violence by police officers more widely available. For example, in 2001 the State of New Jersey required local police departments to document any “use of force” by officers, whether minor or major, such as firing their gun. But no one could easily search these paper forms until a team of journalists from NJ Advance Media submitted over 500 public records requests and compiled The Force Report digital database, where anyone can look up individual officers and investigate patterns of violent behavior. Similarly, a team of ProPublica journalists created The NYPD Files database, which now allows anyone to search closed cases of civilian complaints against New York City police officers, by name or precinct, for patterns of substantiated allegations.

Everyone who works with data needs to get informed about key debates over what should be public or private, become active in policy discussions about whose interests are being served, and contribute to making positive change. In the next section, you’ll learn about ethical choices you’ll need to make when working with sensitive individual-level data.

Mask or Aggregate Sensitive Data

Even if individual-level data is legally and publicly accessible, each of us is responsible for making ethical decisions about if and how to use it when creating data visualizations. When working with sensitive data, some ethical questions to ask are: *What are the risks that publicly sharing individual-level data might cause more harm than good?* and *Is there a way to tell the same data story without publicly sharing details that may intrude on individual privacy?* There are no simple answers to these ethical questions, since every situation is different and requires weighing the risks of individual harm versus the benefits of broader knowledge about vital public issues. But this section clarifies some of the alternatives to blindly redistributing sensitive information, such as masking and aggregating individual-level data.

Imagine that you’re exploring crime data and wish to create an interactive map about the frequency of different types of 911 police calls across several neighborhoods. If you search for public data about police calls, as described in the

Open Data section in this chapter, you'll see different policies and practices for sharing individual-level data published by police call centers. In many US states, information about victims of sexual crimes or child abuse (such as the address where police were sent) is considered confidential and exempt from public release, so it's not included in the open data. But some police departments publish open data about calls with the full address for other types of crimes, in a format like this:

Date	Full Address	Category
Jan 1	1234 Main St	Aggravated Assault

While this information is publicly available, it's possible that you could cause some type of physical or emotional harm to the victims by redistributing detailed information about a violent crime with their full address in your data visualization.

One alternative is to *mask* details in sensitive data. For example, some police departments hide the last few digits of street addresses in their open data reports to protect individual privacy, while still showing the general location, in a format like this:

Date	Masked Address	Category
Jan 1	1XXX Main St	Aggravated Assault

You can also mask individual-level data when appropriate, using methods similar to the Find and Replace method with your spreadsheet tool as in Chapter 5: Clean Up Messy Data.

Another strategy is to *aggregate* individual-level data into larger groups, which can protect privacy while showing broader patterns. In the example above, if you're exploring crime data across different neighborhoods, grouping individual 911 calls into larger geographic areas, such as census tracts or area names, in a format like this:

Neighborhood	Crime Category	Frequency
East Side	Aggravated Assault	13
West Side	Aggravated Assault	21

Aggregating individual-level details into larger, yet meaningful categories, is also a better way to tell data stories about the bigger picture. To aggregate simple spreadsheet data, see the summarizing with pivot tables section in Chapter 3. To geocode US addresses into census areas, or to pivot address points into a polygon map, or to normalize data to create more meaningful maps, see Chapter 14: Transform Your Map Data.

In the next section, you'll learn how to explore datasets that governments and non-governmental organizations have intentionally shared with the public.

Open Data Repositories

Over the past decade, an increasing number of governmental and non-governmental organizations around the globe have begun to pro-actively share public data through open data repositories. While some of these datasets were previously available as individual files on isolated websites, these growing networks have made open data easier to find, enabled more frequent agency updates, and sometimes support live interaction with other computers. Open data repositories often include these features:

- **View and Export:** At minimum, open data repositories allow users to view and export data in common spreadsheet formats, such as CSV, ODS, and XLSX. Some repositories also provide geographical boundary files for creating maps.
- **Built-in Visualization Tools:** Several repositories offer built-in tools for users to create interactive charts or maps on the platform site. Some also provide code snippets for users to embed these built-in visualizations into their own websites, which you'll learn more about in Chapter 10: Embed on the Web.
- **Application Programming Interface (APIs):** Some repositories provide endpoints with code instructions that allow other computers to pull data directly from the platform into an external site or online visualization. When repositories continuously update data and publish an API endpoint, it can be an ideal way to display live or "almost live" data in your visualization, which you'll learn more about in Chapter 13: Leaflet Map Templates.

Due to the recent growth of open data repositories, especially in governmental policy and scientific research, there is no single website that lists all of them. Instead, we list just a few sites from the US and around the globe to spark readers' curiosity and encourage you to dig deeper:

- Data.gov, the official repository for US federal government agencies.
- Data.census.gov, the main platform to access US Census Bureau data. The Decennial Census is a full count of the population every ten years, while the American Community Survey (ACS) is an annual sample count that produces one-year, three-year, or five-year estimates for different census geographies, with margins of error.
- Eurostat, the statistical office of the European Union.
- Federal Reserve Economic Research, for US and international data.
- Global Open Data Index, by the Open Knowledge Foundation.
- Google Dataset Search.
- Harvard Dataverse, open to all researchers from any discipline.
- Humanitarian Data Exchange, by the United Nations Office for the Coordination of Humanitarian Affairs.

- IPUMS, Integrated Public Use Microdata Series, the world’s largest individual-level population database, with microdata samples from US and international census records and surveys, hosted by the University of Minnesota.
- openAfrica, by Code for Africa.
- Open Data Inception, a map-oriented global directory.
- Open Data Network, a directory by Socrata, primarily of US state and municipal open data platforms.
- United Nations data.
- World Bank Open Data, a global collection of economic development data.
- World Inequality Database, global data on income and wealth inequality.

For more options, see *Open Data* listings that have been organized and maintained by staff at several libraries, including the University of Rochester, SUNY Geneseo, Brown University, and many others.

In addition, better-resourced higher-education libraries and other organizations may pay subscription fees that allow their students and staff to access “closed” data repositories. For example, Social Explorer offers decades of demographic, economic, health, education, religion, and crime data for local and national geographies, primarily for the US, Canada, and Europe. Previously, Social Explorer made many files available to the public, but it now requires a paid subscription or 14-day free trial. Also, Policy Map provides demographic, economic, housing, and quality of life data for US areas, and makes some publicly visible in its Open Map view, but you need a subscription to download them.

See also how to find geographic boundary files in GeoJSON format, an open data standard used for creating maps in this book, in Chapter 14: Transform Your Map Data.

Now that you’ve learned more about navigating open data repositories, the next section will teach you ways to properly source the data that you discover.

Source Your Data

When you find data, write the source information inside the downloaded file or a new file you create. Add key details about its origins, so that you—or someone else in the future—can replicate your steps. We recommend doing this in two places: the spreadsheet file name and a source notes tab. As a third step, make a backup sheet of your data.

The first step is to label every data file that you download or create. All of us have experienced “bad file names” like these, which you should avoid:

- data.csv
- file.ods

- download.xlsx

Write a short but meaningful file name. While there's no perfect system, a good strategy is to abbreviate the source (such as `census` or `worldbank` or `eurostat`), add topic keywords, and a date or range. If you or co-workers will be working on different versions of a downloaded file, include the current date in YYYY-MM-DD (year-month-date) format. If you plan to upload files to the web, type names in all lower-case and replace blank spaces with dashes (-) or underscores (_). Better file names look like this:

- town-demographics-2019-12-02.csv
- census2010_population_by_county.ods
- eurostat-1999-2019-co2-emissions.xlsx

The second step is to save more detailed source notes about the data on a separate tab inside the spreadsheet, which works for multi-tab spreadsheet tools such as Google Sheets, LibreOffice, and Excel. Add a new tab named *notes* that describes the origins of the data, a longer description for any abbreviated labels, and when it was last updated, as shown in Figure 4.1. Add your own name and give credit to collaborators who worked with you. If you need to create a CSV file from this data, give it a parallel name to your multi-tabbed spreadsheet file so that you can easily find your original source notes again in the future.

Figure 4.1: Create separate spreadsheet tabs for data, notes, and backup.

A third step is to make a backup of the original data before cleaning or editing it. For a simple one-sheet file in a multi-tab spreadsheet tool, right-click on the tab containing the data to make a duplicate copy in another tab, also shown in Figure 4.1. Clearly label the new tab as a backup and leave it alone! For CSV files or more complex spreadsheets, create a separate backup file. To be clear, these simple backup strategy only helps you from making non-fixable edits to your original data. Make sure you have a broader strategy to backup your files from your computer or cloud account in case either of those are deleted or those systems crash.

Make a habit of using these three sourcing strategies—filenames, notes, and backups—to increase the credibility and replicability of your data visualizations.

In the next section, we'll explore more ways to reduce your chances of making "bad data" errors.

Recognize Bad Data

When your data search produces some results, another key step is to open the file, quickly scroll through the content, and look for any warning signs that it might contain "bad data." If you fail to catch a problem in your data at an early stage, it could lead to false conclusions and diminish the credibility of all of your work. Fortunately, members of the data visualization community have shared multiple examples of problems we've previously encountered, to help save newer members from making the same embarrassing mistakes. One popular crowd-sourced compilation by data journalists was The Quartz Guide to Bad Data, last updated in 2018. Watch out for spreadsheets containing these "bad data" warning signs:

- Missing values: If you see blank or "null" entries, does that mean data was not collected? Or maybe a respondent did not answer? If you're unsure, find out from the data creator. Also beware when humans enter a 0 or -1 to represent a missing value, without thinking about its consequences on running spreadsheet calculations, such as SUM or AVERAGE.
- Missing leading zeros: One of the zip codes for Hartford, Connecticut is 06106. If someone converts a column of zip codes to numerical data, it will strip out the leading zero and appear as 6106. Similarly, the US Census Bureau lists every place using a FIPS code, and some of these also begin with a meaningful zero character. For example, the FIPS code for Los Angeles County, California is 037, but if someone accidentally converts a column of text to numbers, it will strip out the leading zero and convert that FIPS code to 37, which may break some functions that rely on this code being a 3-digit number, or may make some people interpret it as a 2-digit state code for North Carolina instead.
- 65536 rows or 255 columns: These are the maximum number of rows supported by older-style Excel spreadsheets, or columns supported by Apple Numbers spreadsheet, respectively. If your spreadsheet stops exactly at either of these limits, you probably have only partial data. As we wrote this, the BBC reported that Public Health England lost thousands of Covid test results due to this row limit in older Excel spreadsheets.
- Inconsistent date formats: For example, November 3rd, 2020 is commonly entered in spreadsheets in the US as 11/3/2020 (month-date-year), while people in other locations around the globe commonly type it as 3/11/2020 (date-month-year). Check your source.
- Dates such as January 1st 1900, 1904, or 1970: These are default timestamps in Excel spreadsheets and Unix operating systems, which may indicate the actual date was blank or overwritten.

- Dates similar to 43891: When you type March 1 during the year 2020 into Microsoft Excel, it automatically displays as 1-Mar, but is saved using Excel's internal date system as 43891. If someone converts this column from date to text format, you'll see Excel's 5-digit number, not the dates you're expecting.

Other ways to review the quality of data entry in any spreadsheet column are to sort or pivot the data as described in Chapter 3, or to create a histogram as you will learn in Chapter 7. These methods enable you to quickly inspect the range of values that appear in a column and to help you identify bad data.

Also beware of bad data due to poor geocoding, when locations have been translated into latitude and longitude coordinates that cannot be trusted. For example, visualization expert Alberto Cairo describes how data *appeared* to show that Kansas residents viewed more online pornography than other US states. But on closer examination, the internet protocol (IP) addresses of many viewers could not be accurately geocoded, perhaps because they sought to maintain their privacy by using virtual private networks (VPN) to disguise their location. As a result, the geocoding tool automatically placed large numbers of users in the geographic center of the contiguous US, which happens to be in Kansas.⁹ Similarly, when global data is poorly geocoded, the population booms on imaginary “Null Island,” which is actually a weather buoy located in the Atlantic Ocean at the intersection of the prime meridian and the equator, where the latitude and longitude coordinates are 0,0. For these reasons, carefully inspect geocoded data for errors caused by tools that mistakenly place results in the exact center of your geography, as shown in Figure 4.2.

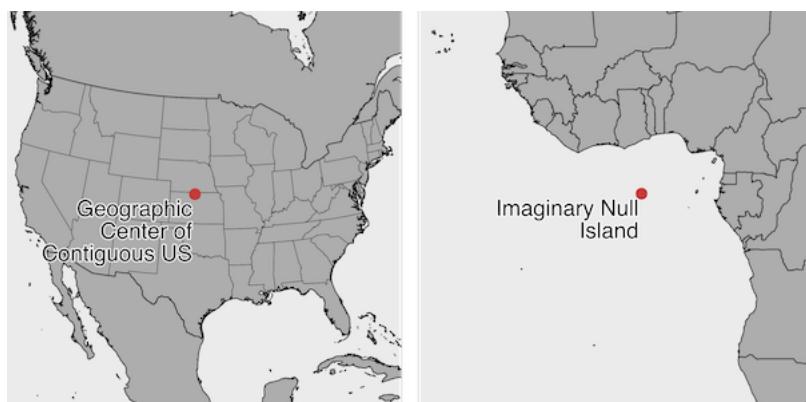


Figure 4.2: Beware of bad geocoding that automatically places data in the geographic center of the contiguous United States (in northern Kansas), or on imaginary Null Island in the Atlantic Ocean (the location of coordinates 0,0).

⁹Cairo, *How Charts Lie*, 2019, pp. 99-102

What should you do when you discover bad data in your project? Sometimes small issues are relatively straightforward and do not call into question the integrity of the entire dataset. Sometimes you can fix these using methods we describe in Chapter 5: Clean Up Messy Data. But larger issues can be more problematic. Follow the source of your data stream to try to identify where the issue began. If you cannot find and fix the issue on your own, contact the data provider to ask for their input, since they should have a strong interest in improving the quality of the data. If they cannot resolve an important data issue, then you need to pause and think carefully. In this case, is it wiser to continue working with problematic data and add a cautionary note to readers, or should you stop using the dataset entirely and call attention to its underlying problem? These are not easy decisions, and you should ask for opinions from colleagues. In any case, never ignore the warning signs of bad data.

Finally, you can help to prevent bad data from occurring by following key steps we've outlined above. Give meaningful names to your data files, and add source notes in a separate tab about when and where you obtained it, along with any definitions or details about what it claims to measure and how it was recorded. Explain what any blanks or null values mean, and avoid replacing those with zeroes or other symbols. Always watch out for formatting issues when entering data or running calculations in spreadsheets.

In the next section, you'll learn more questions to help you understand your data at a deeper level.

Question Your Data

Now that you've found, sourced, and inspected some files, the next step is to *question your data* by looking more deeply than what appears at its surface level. Read the *metadata*, which are the notes that describe the data and its sources. Examine the contents to reflect on what is explicitly stated—or unstated—to better understand its origin, context, and limitations. You cannot program a computer to do this step for you, as it requires critical-thinking skills to see beyond the characters and numbers appearing on your screen.

One place to start is to ask: *What do the data labels really mean?* and to consider these potential issues:

What are full definitions for abbreviated column headers?

Spreadsheets often contain abbreviated column headers, such as *Elevation* or *Income*. Sometimes the original software limited the number of characters that could be entered, or the people who created the header names preferred to keep them short. But was *Elevation* entered in meters or feet? An abbreviated data label does not answer that key question, so you'll need to check the source notes, or if that's not available, compare elevation data for a specific point in the dataset to a known source that includes the measurement unit. Similarly, if

you're working with US Census data, does the *Income* abbreviation refer to per person, per family, or per household? Also, does the value reflect the *median* (the mid-point in a range of numbers) or the *mean* (the average, calculated by adding up the sum and dividing by the number of values). Check definitions in the source notes.

How exactly was the data collected?

For example, was *Elevation* for a specific location measured by a GPS unit on the ground? Or was the location geocoded on a digital map that contains elevation data? In most cases the two methods will yield different results, and whether that matters depends on the degree of precision required in your work. Similarly, when the US Census reports data from its annual American Community Survey (ACS) estimates for *Income* and other variables, these are drawn from small samples of respondents for lower levels of geography, such as a census tract with roughly 4,000 residents, which can generate very high margins of error. For example, it's not uncommon to see ACS estimates for a census tract with a mean family income of \$50,000—but also with a \$25,000 margin of error—which tells you that the actual value is somewhere between \$25,000 and \$75,000. As a result, some ACS estimates for small geographic units are effectively meaningless. Check how data was recorded, and note any reported margins of error, in the source notes. See also how to create error bars in Chapter 7: Chart Your Data.

To what extent is the data socially constructed?

What do the data labels reveal or hide about how people defined categories in different social and political contexts, which differ across place and time? For example, we designed an interactive historical map of racial change for Hartford County, Connecticut using over 100 years of US Census data. But Census categories for race and ethnicity changed dramatically during those decades because people in power redefined these contested terms and moved who belonged in which group.

Into the 1930s, the US Census separated “Native White” and “Foreign-born White” in its official reports, then combined and generally reported these as “White” in later decades. Also, the Census classified “Mexican” as “Other races” in 1930, then moved this group back to “White” in 1940, then began to report “Puerto Rican or Spanish surname” data in 1960, followed by “Hispanic or Latino” in later decades, as an ethnic category that was distinct from race. The Census finally replaced “Negro” with “Black” in 1980, and in 2000 allowed people to select more than one racial category, such as both “White” and “Black,” unlike prior decades when these terms were mutually exclusive and people could choose only one. As a result, historical changes in the social construction of race and ethnicity influenced how we designed our map to display “White” or “White alone” over time, with additional census categories relevant to each decade shown in the pop-up window, with our explanation of our decisions in the caption and source notes. There is no single definitive way to visualize

socially-constructed data when definitions change across decades. But when you make choices about data, describe your thought process in the notes.

Here's a paradox about working with data: some of these deep questions may not be fully answerable if the data was collected by someone other than yourself, especially if that person came from a distant place, or time period, or a different position in a social hierarchy. But even if you cannot fully answer these questions, don't let that stop you from asking good questions about the origins, context, and underlying meaning of your data. Only by clarifying what we know—and what we don't know—can we begin to recognize the limitations of the data. When you create visualizations, your job is also to *acknowledge the limitations of the data* by making thoughtful decisions about its design, and how you describe what it does—and does not—tell us. We'll return to these topics when discussing chart design in Chapter 7 as well as telling and showing your data story in Chapter 16.

Summary

This chapter reviewed two broad questions that everyone should ask during the early stages of their visualization project: *Where can I find data?* and *What do I really know about it?* We broke down both questions into more specific parts to develop your knowledge and skills in guiding questions for your search, engaging with debates over public and private data, masking and aggregating sensitive data, navigating open data repositories, sourcing data origins, recognizing bad data, and questioning your data more deeply than its surface level. Remember these lessons as you leap into the next few chapters on cleaning data and creating interactive charts and maps.

Chapter 5

Clean Up Messy Data

More often than not, datasets will be messy and hard to visualize right away. They will have missing values, dates in different formats, text in numeric-only columns, multiple items in the same columns, various spellings of the same name, and other unexpected things. See Figure 5.1 for inspiration. Don't be surprised if you find yourself spending more time cleaning up data than you do analyzing and visualizing it.

Year	City	Amount
1990	New York City	\$1,123,456.00
1995-96		2.2 mil
2000s	NYC	No data
2020	New_York	5000000+

Figure 5.1: More often than not, raw data looks messy.

In this chapter you'll learn about different tools, in order to help you make decisions about which one to use to clean up your data efficiently. First, we'll start with basic cleanup methods using spreadsheets, such as find and replace with a blank, transpose rows and columns of data, split data into separate columns, and combine columns into one. While we feature Google Sheets in our examples, the same principles (and in most cases the same formulas) can be used in Microsoft Excel, LibreOffice Calc, Mac's Numbers, or other spreadsheet packages. Next, you will learn how to extract table data from text-based PDF documents with Tabula, a free tool used by data journalists and researchers worldwide to analyze spending data, health reports, and all sorts of other datasets that get trapped in PDFs. Finally, we will introduce OpenRefine, a powerful and versatile tool to clean up the messiest spreadsheets, such as those containing dozens of different spellings of the same name.

TODO below: Add Google Sheets Smart Cleanup feature when available in late October <https://gsuiteupdates.blogspot.com/2020/09/smart-cleanup-suggestions-google-sheets.html>

Find and Replace with Blank

One of the simplest and most powerful cleanup tools inside your spreadsheet is the *Find and Replace* command. You can also use it to bulk-change different spellings of the same name, such as shortening a country's name (from *Republic of India* to *India*), or expanding a name (from *US* to *United States*), or translating names (from *Italy* to *Italia*). Also, you can use find and replace with a blank entry to remove units of measurement that sometimes reside in the same cells as the numbers (such as changing *321 kg* to *321*).

Let's look at *Find and Replace* in practice. A common problem with US Census data is that geographic names contain unnecessary words. For example, when you download data on the population of Connecticut towns, the location column will contain the word "town" after every name:

```
Hartford town  
New Haven town  
Stamford town
```

But usually you want a clean list of towns, either to display in a chart or to merge with another dataset, like this:

```
Hartford  
New Haven  
Stamford
```

Let's use *Find and Replace* on a sample US Census file we downloaded with 169 Connecticut town names and their populations, to remove the unwanted "town" label after each place name.

1. Open the CT Town Geonames file in Google Sheets, sign in with your account, and go to *File > Make a Copy* to create a version you can edit in your Google Drive.
2. Select the column you want to modify by clicking its header. If you don't select a column, you will be searching and replacing in the entire spreadsheet.
3. In the *Edit* menu, choose *Find and replace*. You will see the window like is shown in Figure 5.2.

4. In the *Find* field, type `town`, and be sure to *insert a blank space* before the word. If you do not insert a space, you will accidentally remove `town` from places such as `Newtown`. Also, you'll accidentally create trailing spaces, or whitespace at the end of a line without any other characters following it, which can cause troubles in the future.
5. Leave the *Replace with* field blank. Do not insert a space. Just leave it empty.
6. The *Search* field should be set to the range you selected in step 2, or *All sheets* if you didn't select anything.
7. You have the option to *match case*. If checked, `town` and `Town` and `t0wN` will be treated differently. For our purpose, you can leave *match case* unchecked.
8. Press the *Replace all* button. Since this sample file contains 169 towns, the window will state that 169 instances of "town" have been replaced.
9. Inspect the resulting sheet. Make sure that places that include `town` in their name, such as `Newtown`, remained untouched.

	A	B	C	D	E	F	G	
1	Geo_NAME	Pop2010						
2	Andover town	3303						
3	Ansonia town	19249						
4	Ashford town	4317						
5	Avon town	18098						
6	Barkhamsted tow	3799						
7	Beacon Falls tow	6049						
8	Berlin town	19866						
9	Bethany town	5563						
10	Bethel town	18584						
11	Bethlehem town	3607						
12	Bloomfield town	20486						
13	Bolton town	4980						
14	Bozrah town	2627						
15	Branford town	28026						
16	Bridgeport town	144229						
17	Bridgewater towr	1727						
18	Bristol town	60477						
19	Brookfield town	16452						
20	Brooklyn town	8210						
21	Burlington town	9301						
22	Canaan town	1234						
23	Canterbury town	5132						
24	Canton town	10292						

Find and replace

Find Make sure it begins with a space character

Replace with

Search

Match case
 Match entire cell contents
 Search using regular expressions [Help](#)
 Also search within formulas

Figure 5.2: Find and Replace window in Google Sheets.

Transpose Rows and Columns

Sometimes you download good data, but your visualization tool requires you to transpose, or swap the rows and the columns, in order to create the chart or map you desire. This problem often arises when working with time-series or historical data, because tables and charts approach them in opposite directions. When designing a table, the proper method is to place dates horizontally as column headers, so that we read them from left-to-right, like this:¹

Year	2000	2010	2020
Series1
Series2

But when designing a line chart in Google Sheets and similar tools, which you'll learn in Chapter 7: Chart Your Data, we need to transpose the data so that dates run vertically down the first column, in order for the software to read them as labels for a data series, like this:

Year	Series1	Series2
2000
2010
2020

Learn how to transpose rows and columns in our sample data:

1. Open the Transpose sample data file in Google Sheets, sign in with your account, and go to *File > Make a Copy* to create a version you can edit in your Google Drive.
2. Select all of the rows and columns you wish to transpose, and go to *Edit > Copy*.
3. Scroll further down the spreadsheet and click on a cell, or open a new spreadsheet tab, and go to *Edit > Paste Special > Paste Transposed*, as shown in Figure 5.3.

Now that you know how to clean up data by transposing rows and columns, in the next section you'll learn how to split data into separate columns.

¹Stephen Few, *Show Me the Numbers: Designing Tables and Graphs to Enlighten*, Second edition (Burlingame, CA: Analytics Press, 2012), p. 166

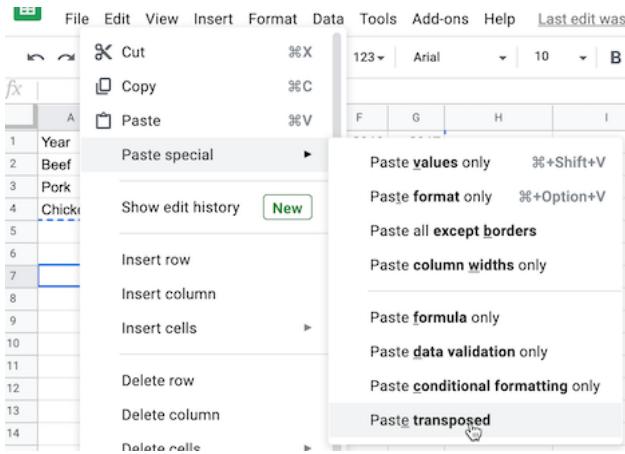


Figure 5.3: Go to *Edit - Paste Special - Paste Transposed* to swap rows and columns.

Split Data into Separate Columns

Sometimes multiple pieces of data appear in a single cell, such as first and last names (John Doe), geographic coordinates (40.12,-72.12), or addresses (300 Summit St, Hartford, CT, 06106). For your analysis, you might want to split them into separate entities, so that your *FullName* column (with John Doe in it) becomes *FirstName* (John) and *LastName* (Doe) columns, coordinates become *Latitude* and *Longitude* columns, and your *FullAddress* column becomes 4 columns, *Street*, *City*, *State*, and *Zip* (postcode).

Example 1: Simple Splitting

Let's begin with a simple example of splitting pairs of geographic coordinates, separated by commas, into separate columns.

1. Open the Split Coordinate Pairs sample data in Google Sheets, sign in with your account, and go to *File > Make a Copy* to create a version you can edit in your Google Drive.
2. Select the data you wish to split, either the full column or just several rows. Note that you can only split data from one column at a time.
3. Make sure there is no data in the column to the right of the one you're splitting, because all data there will be written over.
4. Go to *Data* and select *Split text to columns*, as in Figure 5.4.
5. Google Sheets will automatically try to guess your separator. You will see that your coordinates are now split with the comma, and the Separator is set to *Detect automatically* in the dropdown. You can manually change

it to a comma (,), a semicolon (;), a period (.), a space character, or any other custom character (or even a sequence of characters, which we'll discuss in Example 2 of this section).

6. You can rename the new columns into *Longitude* (first number) and *Latitude* (second number).

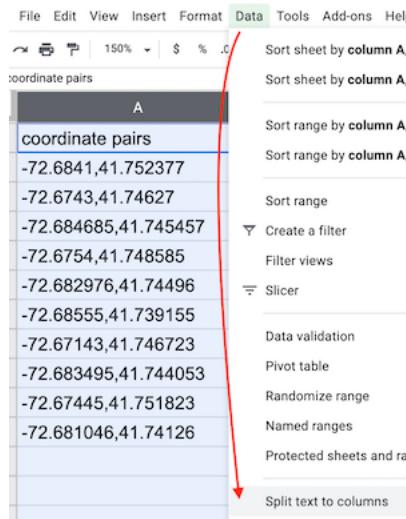


Figure 5.4: Select *Data - Split text to columns* to automatically separate data.

Example 2: Complex Splitting

Now, let's look at a slightly more complicated example. Each cell contains a full address, which you want to split into four columns: street, city, state, and zipcode (postcode). But notice how the separators differ: a comma between street and city, a space between city and state, and two dashes between state and the zipcode. In this case, you'll need to manually add some instructions to properly split the text into four columns.

Location	

300 Summit St, Hartford CT--06106	
1012 Broad St, Hartford CT--06106	
37 Alden St, Hartford CT--06114	

1. Open the Split Complex Address sample file in Google Sheets, sign in to your account, and go to *File > Make a Copy* to save a version in your Google Drive that you can edit.

2. Select the column and go to *Data > Split text to columns* to start splitting from left to right.
3. Google Sheets will automatically split your cell into two parts, **300 Summit St** and **Hartford CT--06106**, using comma as a separator. (If it didn't, just select *Comma* from the dropdown menu that appeared).
4. Now select only the second column and perform *Split text to columns* again. Google Sheets will automatically separate the city from the state and zip code, because it automatically chose a space as the separator. (If it did not, choose *Space* from the dropdown menu).
5. Finally, select only the third column and perform *Split text to columns* again. Google Sheets won't recognize the two dashes as a separator, so you need to manually select *Custom*, type those two dashes (--) in the *Custom separator* field, as shown in Figure 5.5, and press Enter. Now you have successfully split the full address into four columns.

A	B	C	D	E	F
Complex Address					
300 Summit St	Hartford	CT--06106			
1012 Broad St	Hartford	CT--06106			
37 Alden St	Hartford	CT--06114			
			Separator:	Custom	Custom separator

Figure 5.5: To split the last column, select a *Custom* separator and manually type in two dashes.

Tip: Google Sheets will treat zip codes as numbers and will delete leading zeros (so 06106 will become 6106). To fix that, select the column, and go to *Format > Number > Plain text*. Now you can manually re-add zeros. If your dataset is large, consider adding zeros using the formula introduced in the following section.

Combine Data into One Column

Let's perform the reverse action by combining data into one column with a spreadsheet formula, also called concatenation, using the ampersand symbol (&). Imagine you receive address data in four separate columns: street address, city, state, and zip code.

Street	City	State	Zip	
-----	-----	-----	-----	
300 Summit St	Hartford	CT	06106	

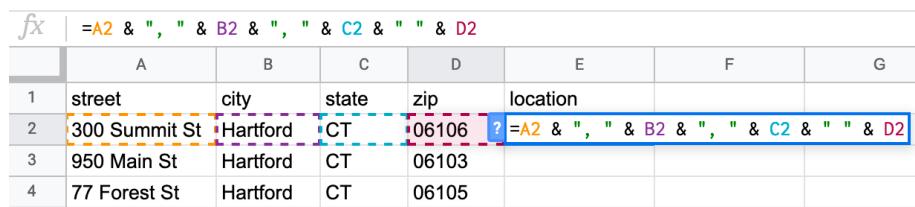
But imagine you need to geocode the addresses using a tool like the one we introduced in Chapter 3, which requires all of the data to be combined into one column like this:

Location	

300 Summit St, Hartford, CT 06106	

Using any spreadsheet, you can write a simple formula to combine (or concatenate) terms using the ampersand (&) symbol. Also, you can add separators into your formula, such as quoted space (" "), or spaces with commas (", "), or any combination of characters. Let's try it with some sample data.

1. Open the Combine Separate Columns sample data in Google Sheets, sign in with your account, and go to *File > Make a Copy* to create a version you can edit in your Google Drive. The sheet contains addresses that are separated into four columns: street, city, state, and zip.
2. In column E, type a new header named *location*.
3. In cell E2, type in the following formula, which combines the four items using ampersands, and separates them with quoted commas and spaces, as shown in Figure 5.6, and press *Enter*. =A2 & ", " & B2 & ", " & C2 & " " & D2
4. Click cell E2 and drag the bottom-right corner cross-hair downward to fill in the rest of the column.



The screenshot shows a Google Sheets spreadsheet. The first row (header) has columns A through E labeled: street, city, state, zip, and location. The second row (data) contains the values: 300 Summit St, Hartford, CT, 06106. The formula for the location column is =A2 & ", " & B2 & ", " & C2 & " " & D2. The third and fourth rows show the results of applying this formula: 950 Main St, Hartford, CT, 06103 and 77 Forest St, Hartford, CT, 06105 respectively. The formula bar at the top shows the full formula =A2 & ", " & B2 & ", " & C2 & " " & D2.

	A	B	C	D	E	F	G
1	street	city	state	zip	location		
2	300 Summit St	Hartford	CT	06106	? =A2 & ", " & B2 & ", " & C2 & " " & D2		
3	950 Main St	Hartford	CT	06103			
4	77 Forest St	Hartford	CT	06105			

Figure 5.6: Use ampersands to combine items, and insert quoted spaces with commas as separators.

Now that you have successfully combined the terms into one location column, you can use the Geocoding by SmartMonkey Google Sheets Add-on we described in Chapter 3 to find the latitude and longitude coordinates, in order to map your data as we'll discuss in Chapter 8

Note: Lisa Charlotte Rost from Datawrapper has written a brilliant blog post about cleaning and preparing your spreadsheet data for analysis and visualization, which we recommend for further reading.²

²Lisa Charlotte Rost, “How to Prepare Your Data for Analysis and Charting in Excel & Google Sheets,” Chartable: A Blog by Datawrapper, accessed August 28, 2020, <https://blog.datawrapper.de/prepare-and-clean-up-data-for-data-visualization/index.html>

Spreadsheets are great tools to find and replace data, split data into separate columns, or combine data into one column. But what if your data table is trapped inside a PDF? In the next section, we will introduce Tabula and show you how to convert tables from text-based PDF documents into tables that you can analyze in spreadsheets.

Extract Tables from PDFs with Tabula

It sometimes happens that the dataset you are interested in is only available as a PDF document. Don't despair, you can *likely* use Tabula to extract tables and save them as CSV files. Keep in mind that PDFs generally come in two flavors: text-based and image-based. If you can use cursor to select and copy-paste text in your PDF, then it's text-based, which is great because you can process it with Tabula. But if you cannot select and copy-paste items inside a PDF, then it's image-based, meaning it was probably created as a scanned version of the original document. You need to use optical character recognition (OCR) software, such as Adobe Acrobat Pro or another OCR tool, to convert an image-based PDF into a text-based PDF. Furthermore, Tabula can only extract data from tables, not charts or other types of visualizations.

Tabula is a free tool that runs on Java in your browser, and is available for Mac, Windows, and Linux computers. It runs on your local machine and does not send your data to the cloud, so you can also use it for sensitive documents.

To get started, download the newest version of Tabula. You can use download buttons on the left-hand side, or scroll down to the *Download & Install Tabula* section to download a copy for your platform. Unlike most other programs, Tabula does not require installation. Just unzip the downloaded archive, and double-click the icon. If you work on a Mac, you may see a warning that states, "Tabula is an app downloaded from the internet. Are you sure you want to open it?" If so, click *Open*. Or you may have to go to *System Preferences > Security & Privacy > General tab*, and click the *Open Anyway* button in the lower half of the window to open the app the first time.

When you start up Tabula, the default system browser will open, as shown in Figure 5.7. Tabula runs on your local computer, not the internet. The URL in the browser will be something like <http://127.0.0.1:8080/>. The first portion is the localhost or hostname for your computer, and 8080 refers to the port number. If you see a different port number, that's fine, and just means that the initial number is already in use by some other program on your computer. If for any reason you decide to use a different browser, just copy-and-paste the URL.

Now let's upload a sample text-based PDF and detect any tables we wish to extract. In the beginning of the Covid-19 pandemic, the Department of Public Health in Connecticut issued data on cases and deaths only in PDF document

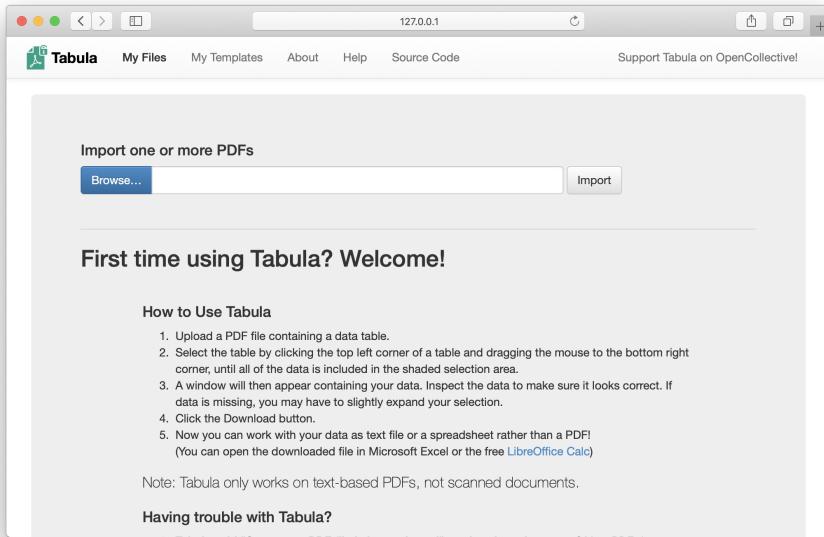


Figure 5.7: Tabula welcome page.

format. For this demonstration, you can use our sample text-based PDF from May 31, 2020, or provide your own.

1. Select the PDF you want to extract data from by clicking the blue *Browse...* button.
2. Click *Import*. Tabula will begin analyzing the file.
3. As soon as Tabula finishes loading the PDF, you will see a PDF viewer with individual pages. The interface is fairly clean, with only four buttons in the header.
4. Click the *Autodetect Tables* button to let Tabula look for relevant data. The tool highlights each table it detects in red, as shown in Figure 5.8.

Now let's manually adjust our selected tables and export the data.

5. Click *Preview & Export Extracted Data* green button to see how Tabula thinks the data should be exported.
6. If the preview tables don't contain the data you want, try switching between *Stream* and *Lattice* extraction methods in the left-hand-side bar.
7. If the tables still don't look right, or you wish to remove some tables that Tabula auto-detected, hit *Revise selection* button. That will bring you back to the PDF viewer.

Town	Cases	Deaths	
Ashley Falls	30	Griswold	24
Amistad	256	Groton	87
Ansonia	15	Hartford	94
Aston	116	Hadlyme	29
Aston	26	Haddam	944
Bantam	45	Hartford	985
Berlin	150	Hartford	2250
Bethany	32	Hartford	5
Bethel	245	Hartford	28
Bethel	123	Hartford	23
Bloomfield	482	Kent	2
Bloomfield	73	Killington	29
Bolton	7	Killington	14
Brimfield	227	Lebanon	23
Bridgewater	334	Litchfield	23
Brownfield	7	Lisbon	11
Brownfield	153	Litchfield	36
Brownfield	155	Lyme	2
Brownfield	22	Madison	117
Brownfield	24	Middlefield	540
Canaan	0	Mansfield	29
Canton	11	Mansfield	3104
Canterbury	83	Minden	73
Chester	3	Middletown	772
Cheshire	183	Middlefield	44
Citrus	46	Middlefield	18
Citrus	52	Middlefield	548
Cochituate	36	Middlefield	620
Cochituate	3	Milford	105
Columbus	22	Milford	105
Comstock	6	Milford	233
Cromwell	40	New Britain	14
Cromwell	115	New Canaan	509
Danbury	1373	New Haven	120
Darien	204	New Hartford	25
Darien	9	New Haven	2443
Deep River	153	New London	136
Derby	32	New Milford	269
Durham	9	New Milford	323
East Granby	17	Newtown	211
East Haddam	42	Newtown	13
East Hartland	109	North Branford	81
East Hartford	793	North Branford	29
East Haven	209	North Haven	6
East Litchfield	142	North Haven	243
East Lynde	146	North Stonington	12
East Litchfield	8	North Stonington	1993
Easton	30	Norwich	89
Easton	56	Oliver Ames	12

Figure 5.8: Click *Autodetect Tables*, which Tabula will highlight in red.

8. Now you can *Clear All Selections* and manually select tables of interest. Use drag-and-drop movements to select tables of interest (or parts of tables).
9. If you want to “copy” selection to some or all pages, you can use *Repeat this Selection* dropdown, which appears in the lower-right corner of your selections, to propagate changes. This is extremely useful if your PDF consists of many similarly-formatted pages.
10. Once you are happy with the result, you can export it. If you have only one table, we recommend using CSV as export format. If you have more than one table, consider switching export format in the drop-down menu to *zip of CSVs*. This way each table will be saved as an individual file, rather than all tables inside one CSV file.

Once you exported your data, you can find it in the Downloads folder on your computer (or wherever you chose to save it), where it is ready to open with a spreadsheet tool to analyze and visualize.

In the following section, we are going to look how to clean up messy datasets with OpenRefine.

Clean Data with OpenRefine

Look at the sample US Foreign Aid dataset shown in Figure 5.9. Can you spot any problems with it?

	A	B	C	D
1	Year	Country	FundingAgency	FundingAmount
2	2000	Korea, N	Dept of Agriculture	\$32 242 376
3	2000	Korea–North	Dept of Agriculture	\$86,151,301
4	2000	Korea North	department of State	166855
5	2000	SouthKorea	U.S. Agency for International Development	282,805a
6	2000	south Korea	Trade and Development Agency	735718
7	2001	North Korea	US Agency for International Development	345,399
8	2001	N Korea	Department of Argic	117715223
9	2001	So Korea	Department of agriculture	2260293
10	2001	Korea, North	State Department	183,752
11	2001	Korea, South	Trade and Development Agency	329,953
12	2002	Korea, N	Department of Agriculture	37,322,244.00
13	2002	Korea, South	U.S. Agency for International Development	67,990.00
14	2002	Korea, South	Trade and Development Agency	\$294,340
15	2003	Korea, North	U.S. Agency for International Development	\$333 823
16	2003	Korea, North	Department - Agriculture	\$26,766,828
17	2003	Korea, North	Department - Agriculture	\$19,337,695
18	2003	Korea, No	Department of State	220,323
19	2003	Korea, South	U.S. Agency for International Development	66,765
20	2003	Korea, South	Trade and Development Agency	19,899

Figure 5.9: Can you spot any problems with this sample data?

Notice how the *Country* column various spellings of North and South Korea. Also note how the *FundingAmount* column is not standardized. Some amounts use commas to separate thousands, while some uses spaces. Some amounts start with a dollar sign, and some do not. Datasets like this can be an absolute nightmare to analyze. Luckily, OpenRefine provides powerful tools to clean up and standardize data.

Note: This data excerpt is from US Overseas Loans and Grants (Greenbook) dataset, which shows US economic and military assistance to various countries. We chose to only include assistance to South Korea and North Korea for the years between 2000 and 2018. We added deliberate misspellings and formatting issues for demonstration purposes, but we did not alter values.

Set up OpenRefine

Let's use OpenRefine to clean up this messy data. Download OpenRefine for Windows, Mac, or Linux. Just like Tabula, it runs in your browser and no data leaves your local machine, which is great for confidentiality.

If you use Windows, unzip the downloaded file. Double-click the .exe file, and OpenRefine should open in your default browser.

If you work on a Mac, the downloaded file will be a .dmg file to be installed. But you will likely see a security warning that prevents OpenRefine from launching because Apple cannot identify the developer for this open-source project. To resolve the problem, go to *System Preferences > Security and Privacy > General tab*, and click the *Open Anyway* button in the lower half of the window, as shown in Figure 5.10. If prompted with another window, click *Open*.

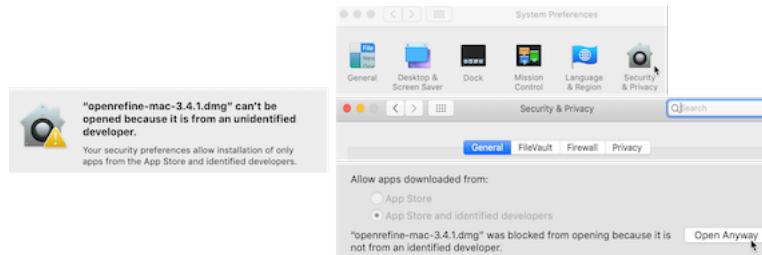


Figure 5.10: If your Mac displays a warning about launching Open Refine (on left), go to *System Preferences - Security and Privacy - General tab* and click *Open Anyways* (on right).

Once launched, you should see OpenRefine in your browser with 127.0.0.1:3333 address (localhost, port 3333), as shown in Figure 5.11.

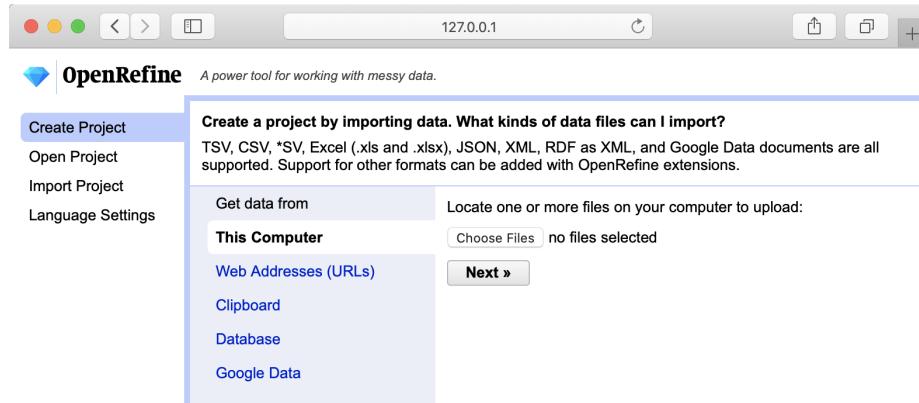


Figure 5.11: OpenRefine starting page.

Load Data and Start a New Project

To start cleaning up messy dataset, we need to load it into a new project. OpenRefine lets you upload a dataset from your local machine, or a remote web address (such as a Google Sheet). OpenRefine also can extract data directly from SQL databases, but this is beyond the scope of this book.

1. Download the sample messy data on US Foreign Aid in CSV format to your computer.
2. Under *Get data from: This computer*, click *Browse...* and select the CSV file you downloaded above. Click *Next*.
3. Before you can start cleaning up data, OpenRefine allows you to make sure data is *parsed* properly. In our case, parsing means the way the data is split into columns. Make sure OpenRefine assigned values to the right columns, or change setting in *Parse data as* block at the bottom of the page until it starts looking meaningful, like shown in Figure 5.12.
4. Hit *Create Project* in the upper-right corner.

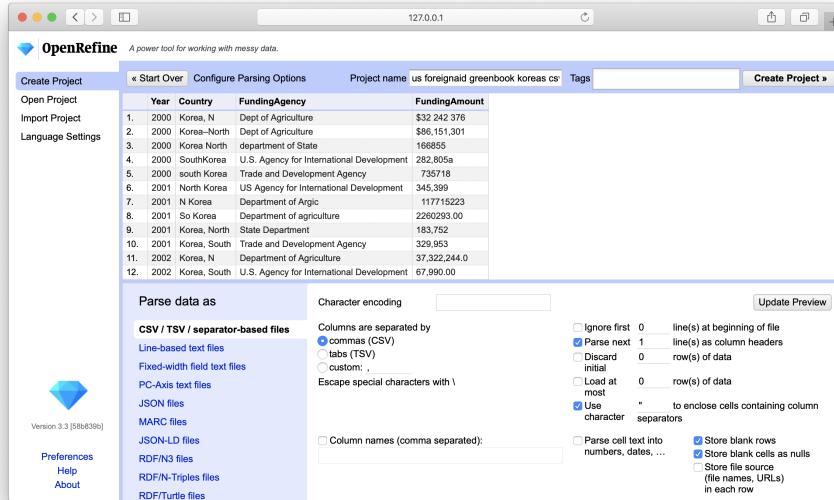


Figure 5.12: OpenRefine parsing options.

Now when you've successfully read the data into a new project, let's start the fun part: converting text into numbers, removing unnecessary characters, and fixing the spellings for North and South Koreas.

Convert Dollar Amounts from Text to Numbers

Once your project is created, you will see the first 10 rows of the dataset. You can change it to 5, 10, 25, or 50 by clicking the appropriate number in the header

Each column header has its own menu (callable by clicking the arrow-down button). Left-aligned numbers in a column are likely represented as text (as

is the case with FundingAmount column in our example), and they need to be transformed into numeric format.

1. To transform text into numbers, open the column menu, and go to *Edit cells > Common transforms > To number*.
2. You will see that some numbers became green and right-aligned (success!), but most did not change. That is because dollar sign (\$) and commas (,) confuse OpenRefine and prevent values to be converted into numbers.
3. Let's remove \$ and , from the FundingAmount column. In the column menu, choose *Edit cells > Transform*. In the Expression window, type `value.replace(',', '')` and notice how commas disappear in the preview window. When you confirm your formula works, click *OK*.
4. Now, repeat the previous step, but instead of a comma, remove the \$ character. (Your expression will become `value.replace('$', '')`).
5. In steps 3 and 4, we replaced text (string) values with other string values, making OpenRefine think this column is no longer numeric. As a result, all values are once again left-aligned and in black. Perform step 1 again to see that all but three cells turning green (successfully converting to numeric). Now we need to remove spaces and an a character at the end of one number. Fix those manually by hovering over cells, and clicking the **edit** button (in the new popup window, make sure to change *Data type* to *number*, and hit *Apply*, like in Figure 5.13).

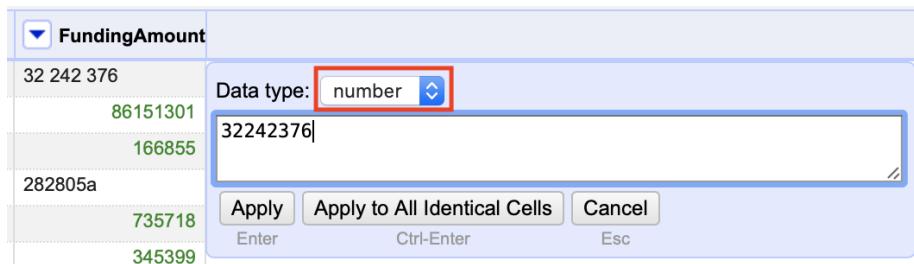


Figure 5.13: Manually remove spaces and extra characters, and change data type to number.

At this point, all funding amounts should be clean numbers, right-aligned and colored in green. We're ready to move on to the Country column and fix different spellings of Koreas.

Cluster Similar Spellings

When you combine different data sources, or process survey data where respondents wrote down their answers as opposed to selecting them from a dropdown menu, you might end up with multiple spellings of the same word (town name,

education level – you name it!). One of the most powerful features of OpenRefine is the ability to cluster similar responses.

If you use our original sample file, take a look at the *Country* column and all variations of North and South Korea spellings. From *Country* column’s dropdown menu, go to *Facet > Text facet*. This will open up a window in the left-hand side with all spellings (and counts) of column values. 26 choices for a column that should have just two distinct values, North Korea and South Korea!

1. To begin standardizing spellings, click on the arrow-down button of *Country* column header, and choose *Edit cells > Cluster and edit*. You will see a window like the one shown in Figure 5.14.
2. You will have a choice of two clustering methods, *key collision* or *nearest neighbor*. Key collision clustering is a much faster technique that is appropriate for larger datasets, but it is less flexible. Nearest neighbor is a more computationally expensive approach and will be slow on larger datasets, but it allows for greater fine-tuning and precision. Both methods can be powered by different functions, which you can read about on the project’s Wiki page. For the purpose of this exercise, let’s leave the default *key collision* method with *fingerprint* function.
3. OpenRefine will calculate a list of clusters. *Values in Cluster* column contains grouped spellings that OpenRefine considers the same. If you agree with a grouping, check the *Merge?* box, and assign the “true” value to the *New Cell Value* input box (see first cluster in Figure 5.14). In our example, this would be either **North Korea** or **South Korea**.
4. You can go through all groupings, or stop after one or two and click *Merge Selected & Re-Cluster* button. The clusters you chose to merge will be merged, and grouping will be re-calculated (don’t worry, the window won’t go anywhere). Keep regrouping until you are happy with the result.

Spend some time playing with *Keying function* parameters, and notice how they produce clusters of different sizes and accuracy.

Export

Once you are done cleaning up and clustering data, save the clean dataset by clicking *Export* button in the upper-right corner of OpenRefine window. You can choose your format (we recommend CSV, or comma-separated value). Now you have a clean dataset that is ready to be processed and visualized.

Summary

In this chapter, we looked at cleaning up tables in Google Sheets, liberating tabular data trapped in PDFs using Tabula, and using OpenRefine to clean up

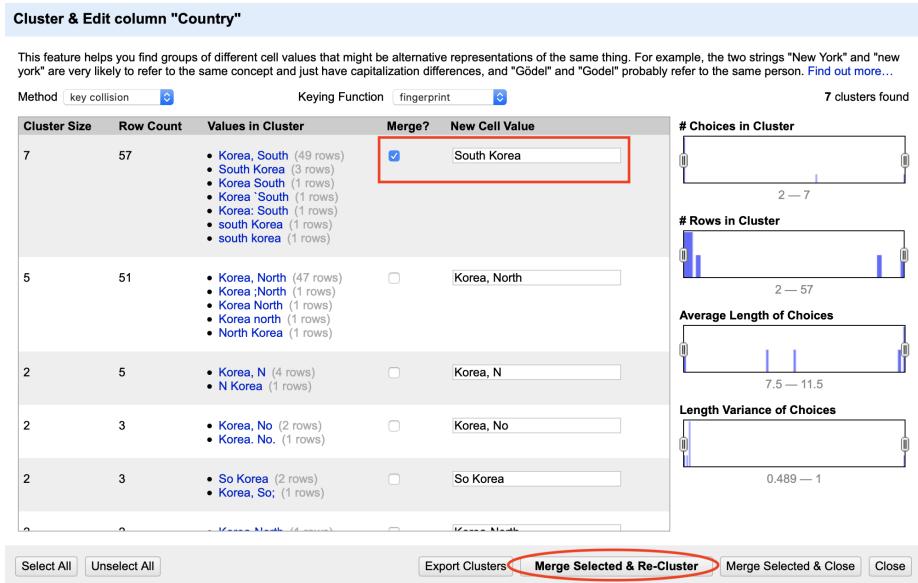


Figure 5.14: Cluster similar text values.

very messy datasets. You will often find yourself using several of these tools on the same dataset before it becomes good enough for your analysis. We encourage you to learn more formulas in Google Sheets, and explore extra functionality of OpenRefine in your spare time. The more clean-up tools and techniques you know, the more able and adaptable you become to tackle more complex cases.

You now know how to clean up your data, so let's proceed to visualizing it. In the following chapter, we will introduce you to a range of free data visualization tools that you can use to build interactive charts and embed them in your website.

Chapter 6

Make Meaningful Comparisons

Now that you've refined your data story, improved your spreadsheet skills, found and questioned your data, and cleaned up any messy parts, this chapter focuses on the key question to ask while analyzing your evidence: "Compared to what?" That's how statistician Edward Tufte defined the "heart of quantitative reasoning."¹. We search for insightful findings in our data by judging their significance against each other, to identify those that truly stand out. Sometimes we need to adjust our scales to ensure that we're weighing data fairly, or as the saying goes, comparing apples to apples, not apples to oranges. Before you communicate your findings in any format—text, tables, charts, or maps—be sure that you're making meaningful comparisons, because without this, your work may become meaningless.

This book does not intend to cover statistical data analysis, since many excellent resources already address this expansive field of study.² Instead, this chapter offers several common-sense strategies to make meaningful comparisons while you analyze your data, in order to help you design true and insightful visualizations that tell your story. You will learn to precisely choose words when describing comparisons, why and how to normalize your data, and advice on watching out for biased comparisons.

¹Edward R Tufte, *Envisioning Information* (Cheshire, CT: Graphics Press, 1990), https://www.google.com/books/edition/Envisioning_Information/_EZiAAAAMAAJ, p. 67

²For a reader-friendly introduction to statistical logic and its limits, see Charles Wheelan, *Naked Statistics: Stripping the Dread from the Data* (W. W. Norton & Company, 2013), https://www.google.com/books/edition/Naked_Statistics_Stripping_the_Dread_fro/j5qYPqsBJb0C; David Spiegelhalter, *The Art of Statistics: How to Learn from Data* (Basic Books, 2019), https://www.google.com/books/edition/The_Art_of_Statistics/04-FDwAAQBAJ

Precisely Describe Comparisons

Sometimes we make poor comparisons because we fail to clarify our meaning of commonly-used words that can have different definitions. Three troublesome words are *average*, *percent*, and *causes*. We use them loosely in everyday conversation, but they require more precision when working with data.

Imagine a series of numbers: 1, 2, 3, 4, 5. When calculating the *average*, by hand or with a built-in spreadsheet formula as described in chapter 3, we add up the sum and divide by the count of numbers. A more precise term is the *mean*, which in this case equals 3. A different term is the *median*, which refers to the number in the middle of the ordered series, also known as the *50th percentile*, which in this case is also 3.

When working with data, the terms *median* and *percentile* are more useful terms when making comparisons because they resist the influence of *outliers* at the extreme ends of the series. For example, imagine the same numbers as above, but replace the 5 with 100 to serve as an outlier. Suddenly the mean jumps up to 22, but the median remains the same at 3, as shown in Figure 6.1. There's an old joke that when a billionaire walks into a room, everyone becomes a millionaire—on average—but the median remains the same. Since we ordinary people don't actually become richer by the presence of the billionaire outlier among us, the median is a better term to make meaningful comparisons about the overall distribution of the data.

	$=MEDIAN(A1:E1)$						
	A	B	C	D	E	F	G
1	1	2	3	4	100	Mean	22
2					Median	3	

Figure 6.1: The *median* is a more useful comparative term than *average* or *mean* because it resists the influence of outliers.

Percentage is another common term, which nearly everyone intuitively grasps as a *ratio* of parts per hundred. For example, an old 1970s commercial for Trident gum claimed that “4 out of 5 dentists surveyed recommend sugarless gum for their patients who chew gum.”³ Even if you’re too young to remember that slogan, or wonder how that survey was actually conducted, or are puzzling over how the fifth dentist resisted such intense peer pressure, we all understand that 4 out of 5 dentists is equivalent to $4/5 = 0.8 = 80\%$.

But confusion arises sometimes when people hastily compare percentages, so we need to carefully choose our words. One term is *percent change* (also called relative change), which works best when comparing *old versus new values*. Percent change is calculated by the difference between new and old values, divided

³Andrew Adam Newman, “Selling Gum with Health Claims,” *The New York Times: Business*, July 27, 2009, <https://www.nytimes.com/2009/07/28/business/media/28adco.html>

by the absolute value of the old value, or $(\text{New value} - \text{Old value}) / |\text{Old value}|$. For example, if 4 dentists recommended sugarless gum in 1960, but peer pressure finally prevailed and 5 dentists recommend it in 2020, we calculate the percent change as $(5-4)/4 = 1/4 = 0.25 = 25\%$.

Another term is *percentage point difference*, which works best when comparing *old versus new percentages* and is calculated by subtracting one from the other. For example, if 80 percent of dentists recommended sugarless gum in 1960, but 100 percent recommended it in 2020, we could compare the two figures by calculating the difference as $\text{New percentage} - \text{Old percentage} = 100\% - 80\% = 20 \text{ percentage point difference}$.

When we precisely use each term, there are two correct ways to compare these figures. One way is to state that “The number of dentists who recommended sugarless gum increased 25 percent over time.” Another way is to state that “The percentage of dentists who recommended sugarless gum increased 20 percentage points over time.” Both statements are accurate. Even if someone confuses the two terms, there’s not a big gap between a “25 percent change” and a “20 percent point increase” in this particular example.

But consider a different example where someone intentionally misleads you with imprecise wording about percentages. Imagine a politician who proposes to raise the sales tax on products and services you purchase from 5 to 6 percent. If that politician says, “it’s only a 1 percent increase,” they’re wrong. Instead, there are two truthful ways describe this change. One way is to state that the tax “will increase 20 percent” because $(6-5)/5 = 0.20$. Another way is to state that the tax “will increase by 1 percentage point” because $6\% - 5\% = 1 \text{ percentage point difference}$. See why the politician preferred to say it in their misleading way, rather than either of the two correct ways? Don’t let anyone fool you by describing how percentages change with very loose wording, and be precise about its meaning in your own work to avoid confusing other people.

A final recommendation about using more precise language is to be cautious with words that suggest a *cause-and-effect relationship* in your data. In everyday conversation, there are many ways that we loosely imply that a causal relationship, where an action directly results in a reaction. For example, when we say one thing “leads to” another, or “promotes” growth, or “sparks” change, those words suggest causality. While that’s fine in daily conversation, we need to choose our words more carefully when discussing data, using three concepts. The first step is to describe any *correlation* between two variables, which means to show how they are associated or related interdependently. But statisticians always warn us that correlation does not imply causation. The fact that two things are related does not necessarily mean that one causes the other to happen. In order to show causation, we must take the second step of proving both correlation and demonstrating a *persuasive theory* for how one factor (sometimes called the independent variable) creates a change in another factor (called the dependent variable). Third, we need to identify and isolate any *confounding*

variables that we have not considered that may also influence the cause-and-effect relationship. While the details are beyond the scope of this book, be mindful of the concepts—and choose your words carefully—when working with data.

See also table design recommendations for showing data correlations and possible causal relationships in Chapter 9: Table Your Data.

Now that you have a clearer understanding of how to use key words to describe data relationships more precisely, in the next section you'll build on this knowledge and adjust data to create more meaningful comparisons.

Normalize Your Data

When we work with data expressed in *counts*, such as 3,133 motor vehicle crash deaths in Florida in 2018, it usually makes no sense to compare these numbers until we *normalize* them. This means to adjust data that has been collected using different scales into a common reference scale, or in other words to convert *raw data* into *rates* to make more meaningful comparisons. Even if you've never heard the term, perhaps you're already normalizing data without realizing it.

Here's an example about motor vehicle safety that was inspired by visualization expert Alberto Cairo, with updated 2018 data from the Insurance Institute for Highway Safety (IIHS) and the US Department of Transportation.⁴ Over 36,000 people died in motor vehicle crashes in 2018, including car and truck drivers and occupants, motorcyclists, pedestrians, and bicyclists. Although only a small fraction of this data appears in the tables below, you can view all of the data in Google Sheets format, and save an editable copy to your Google Drive, to follow along in this exercise.

Let's start with what appears to be a simple question, and see where our search for more meaningful comparisons takes us.

1. *Which US states had the lowest number of motor vehicle crash deaths?*

When we sort the data by the numbers of deaths, the District of Columbia *appears* to be the safest state with only 31 deaths, as shown in Table 6.1, even though Washington DC is not legally recognized as a state.

⁴Alberto Cairo, *The Truthful Art: Data, Charts, and Maps for Communication* (Pearson Education, 2016), https://www.google.com/books/edition/The_Truthful_Art/8dKKCwAAQBAJ, pp. 71-74.

Table 6.1: US States with lowest number of motor vehicle crash deaths, 2018

State	Deaths
District of Columbia	31
Rhode Island	59
Vermont	68
Alaska	80
North Dakota	105

But wait—this isn’t a fair comparison. Take another look at the five states above and you’ll notice that all of them have smaller populations than larger states, such as California and Texas, which appear at the very bottom of the full dataset. To paint a more accurate picture, let’s rephrase the question to adjust for population differences.

2. *Which US states had the lowest number of motor vehicle crash deaths when adjusted for population?* Now let’s normalize the death data by taking into account the total population of each state. In our spreadsheet, we calculate it as `Deaths / Population * 100,000`. While it’s also accurate to divide deaths by population to find a *per capita* rate, those very small decimals would be difficult for most people to compare, so we multiply by 100,000 to present the results more clearly. When we sort the data, Washington DC *appears* to be the safest once again, with only 4.4 motor vehicle crash deaths per 100,000 residents, as shown in Table 6.2

Table 6.2: US States with lowest number of motor vehicle crash deaths per population, 2018

State	Deaths	Population	Deaths per 100,000 population
District of Columbia	31	702,455	4.4
New York	943	19,542,209	4.8
Massachusetts	360	6,902,149	5.2
Rhode Island	59	1,057,315	5.6
New Jersey	564	8,908,520	6.3

But wait—this still isn’t a fair comparison. Look at the five states on the list and you’ll notice that all of them are located along the Northeastern US corridor, which has a high concentration of public transit, such as trains and subways. If people in urban areas like New York and Boston are less likely to drive motor vehicles, or take shorter trips than people in rural states where homes are more distantly spread out, that might affect our data. Let’s strive

for a better comparison and rephrase the question again, this time to adjust for differences in mileage, not population.

3. *Which US states had the lowest number of motor vehicle crash deaths when adjusted for vehicle mileage?* Once again, we normalize the death data by adjusting it to account for a different factor: vehicle miles traveled (VMT), the estimated total number of miles (in millions) traveled by cars, vans, trucks, and motorcycles, on all roads and highways in the state, in 2018. In our spreadsheet, we calculate it as **Deaths / Vehicle Miles * 100**, with the multiplier to present the results more clearly. This time Massachusetts *appears* to be the safest state, with only 0.54 motor vehicle crash deaths per 100 million miles traveled, as shown in as shown in Table 6.3. Also, note that the District of Columbia has fallen further down the list and been replaced by Minnesota.

Table 6.3: US States with lowest number of motor vehicle crash deaths per miles traveled, 2018

State	Deaths	Vehicle miles traveled (millions)	Deaths per 100 million vehicle miles traveled
Massachusetts	360	66,772	0.54
Minnesota	381	60,438	0.63
New Jersey	564	77,539	0.73
Rhode Island	59	8,009	0.74
New York	943	123,510	0.76

Have we finally found the *safest* state as judged by motor vehicle crash deaths? Not necessarily. While we normalized the raw data relative to the population and amount of driving, the IIHS reminds us that several other factors may influence these numbers, such as vehicle types, average speed, traffic laws, weather, and so forth. But as Alberto Cairo reminds us, every time we refine our calculations to make a more meaningful comparison, our interpretation becomes a closer representation of the truth. “It’s unrealistic to pretend that we can create a *perfect* model,” Cairo reminds us. “But we can certainly come up with a *good enough* one.”⁵

As we demonstrated above, the most common way to normalize data is to adjust raw counts into relative rates, such as percentages or per capita. But there are many other ways to normalize data, so make sure you’re familiar with different methods when you find and question your data, as described in chapter 4. When working with historical data (also called time-series or longitudinal data), you

⁵Cairo, p. 95

may need to *adjust for change over time*. For example, it's not fair to directly compare median household income in 1970 versus 2020, because \$10,000 US dollars had far more purchasing power a half-century ago than it does today, due to inflation and related factors. Similarly, economists distinguish between *nominal data* (unadjusted) versus *real data* (adjusted over time), typically by converting figures into "constant dollars" for a particular year that allow better comparisons by accounting for purchasing power.⁶ Also, economic data is often *seasonally adjusted* to improve comparisons for data that regularly varies across the year, such as employment or revenue during the summer tourism season versus the winter holiday shopping season. Another normalization method is to create an *index* to measure how values have risen or fallen in relation to a given reference point over time. Furthermore, statisticians often normalize data collected using different scales by calculating its *standard score*, also known as its *z-score*, to make better comparisons. While these methods are beyond the scope of this book, it's important to be familiar the broader concept: everyone agrees that it's better to compare apples to apples, rather than apples to oranges.

Finally, you do *not* always need to normalize your data, because sometimes its format already does this for you. Unlike raw numbers or simple counts, most *measured variables* do not need normalization because they already appear on a common scale. One example of a measured variable is *median age*, the age of the "middle" person in a population, when sorted from youngest to oldest. Since we know that humans live anywhere between 0 and 120 years or so, we can directly compare the median age among different populations. Similarly, another measured variable is *median income*, if measured in the same currency and in the same time period, because this offers a common scale that allows direct comparisons across different populations.

Now that you have a better sense of why, when, and how to normalize data, the next section will warn you to watch out for biased comparisons in data sampling methods.

Beware of Biased Comparisons

Everyone knows not to *cherry-pick* your data, which means to select only the evidence that supports a pre-determined conclusion, while ignoring the remainder. When we make a commitment to tell true and meaningful data stories, we agree to keep an open mind, examine all of the relevant evidence, and weigh the merits of competing interpretations. If you agree to these principles, then also watch out for biased data comparisons, especially sampling procedures that may appear legitimate on the surface, but actually contain less-visible factors that skew the evidence. While we may believe we're operating with open minds, we

⁶"What's Real About Wages?" Federal Reserve Bank of St. Louis, The FRED Blog, February 8, 2018, <https://fredblog.stlouisfed.org/2018/02/are-wages-increasing-or-decreasing/>

can overlook partially-hidden processes that effectively cherry-pick our evidence without our knowledge.

Selection bias happens when we believe we have chosen our data sample fairly, but some behind-the-scenes process influences its composition and skews our analysis. For example, if you conduct surveys by email with US adults, your sample will not be representative of senior citizens aged 65 or older, who are less likely to own a computer or smartphone, according to the Pew Research Center. Also beware of *participation bias*, sometimes called non-response bias. If your survey has a low response rate, it is likely that those who choose to respond possess certain qualities that make them less representative of the general population.

In particular, *self-selection bias* often arises when attempting to evaluate the effectiveness of a particular program or treatment where people applied or volunteered to participate, as shown in Figure 6.2. Imagine that your job is to determine if a weight-loss program actually works, which requires a deeper understanding of how data samples were chosen. Avoid the mistake of comparing the progress of non-participants (group A) versus participants who signed up for this program (group B), because those two groups were *not* randomly chosen. Participants differ because they took the initiative to join a weight-loss program, and most likely have higher motivation to improve their diet and exercise more often than non-participants. Self-selection bias secretly shapes the composition of both groups, which results in a meaningless comparison. We often fool ourselves and overlook how this voluntary participation skews our understanding of program effectiveness, whether the topic is weight-loss clinics, counseling programs, or public and private school choice.

How can we reduce self-selection bias in program evaluation data? As you learned in Chapter 4, it's important to question your data by looking below the surface level to fully comprehend how terms have been defined, and how data was collected and recorded. By contrast, a well-designed program evaluation will reduce self-selection bias by *randomly dividing* all volunteer participants (group B) into two sub-groups: half will be assigned to participate in one weight-loss program (group C) and the other half will be assigned to a different weight-loss program (group D), as shown in Figure 6.2. Since sub-groups C and D were selected by chance from the same larger group of volunteers, we can be more confident when comparing their progress because there is no reason to suspect any difference in motivation or other hard-to-see factors. Of course, there are many more research design details that are beyond the scope of this book, such as ensuring that sample sizes are sufficiently large, and comparing participants before, during, and after the weight-loss activity, and so forth. But the logic of avoiding selection bias is simple: randomly divide people who apply or volunteer to participate into sub-groups, to better compare program effectiveness among people with similar motivations and other hard-to-see characteristics.

Bias warnings appear in several chapters of this book, because we continually need to be aware of different types that negatively influence our work at various

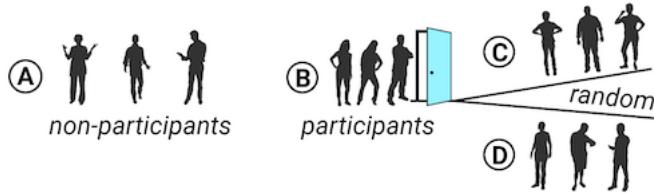


Figure 6.2: Do not compare program non-participants (A) versus those who apply or volunteer to participate (B). Instead, randomly split all participants into two sub-groups (C and D) to compare program effectiveness.

stages of the data visualization process. Later in Chapter 15 you'll learn how to recognize and reduce other types of biases when working with data, such as cognitive biases, algorithmic biases, intergroup biases, and map area biases.

Summary

Although we do not claim to teach you statistical data analysis in this book, in this chapter we discussed several common-sense strategies to make meaningful comparisons while analyzing your data. You learned how to use words more precisely for comparing data, why and how to normalize data, and advice on watching out for biased comparisons. In prior chapters you built up your skills on refining your data story, working with spreadsheets, finding and questioning data, and cleaning up messy data. Now you can combine all of this knowledge and begin to create interactive charts and maps in the next few chapters.

Chapter 7

Chart Your Data

Charts pull readers deeper into your story. Images such as the slope of a line chart, or clusterings of dots on a scatter chart, can communicate your evidence to readers' eyes more effectively than text or tables. But creating meaningful charts that draw our attention to key insights in your data requires clear thinking about design choices.

In this chapter, we will examine principles of chart design, and learn to identify good charts from bad ones. You will review important rules that apply to all charts, and also some aesthetic guidelines to follow when customizing your own designs. While many tools allow you to download charts as *static* images, our book also demonstrates how to construct *interactive* charts that invite readers to explore the data in their web browsers. Later you'll learn how to embed interactive charts on your website in Chapter 10.

Learn about different types of charts you can create in this book in Table 7.1. Decisions about chart types are based on two main factors: the format of your data, and the kind of story you wish to tell. For example, line charts work best to show a series of continuous data points (such as change over time), while range charts are better suited to emphasize the distance between data categories (such as inequality gaps). After selecting your chart type, follow our tool recommendations and step-by-step tutorials. This chapter features *Easy Tools* with drag-and-drop menus, such as Google Sheets, Datawrapper, and Tableau Public. But the table also points you to *Power Tools* that give you more control to customize and host your visualizations, such as Chart.js and Highcharts code templates in Chapter 12. These advanced tools require prior knowledge on how to edit and host code templates with GitHub in Chapter 11.

A note about terminology: we jointly refer to *bar* and *column charts* because they're essentially the same, except that bars are oriented horizontally and columns vertically. The main difference is the length of your data labels. Use bar charts to display longer labels (such as "Mocha Frappuccino 24-ounce" and

“Double Quarter Pounder with cheese”) since they require more horizontal reading space. But you can use either bar or column charts for shorter labels that do not require as much room (such as “Starbucks” and “McDonald’s”). You’ll also notice that all of the examples in this chapter focus on food (because we were really hungry when writing it) and healthy eating (because we also need to lose weight).

Table 7.1: Basic Chart Types, Best Uses, and Tutorials

Chart	Best use and tutorials in this book																					
Grouped bar or column chart	<p>Best to compare categories side-by-side. If labels are long, use horizontal bars instead of vertical columns. Easy tools: Bar and Column Charts in Google Sheets tutorialPower tool: Ch 12: Chart.js and Highcharts templates</p> <table border="1"> <thead> <tr> <th>Age Group</th> <th>Gender</th> <th>BMI Range</th> </tr> </thead> <tbody> <tr> <td rowspan="2">20 - 39</td> <td>Men</td> <td>40 - 59</td> </tr> <tr> <td>Women</td> <td>40 - 59</td> </tr> <tr> <td rowspan="2">40 - 59</td> <td>Men</td> <td>40 - 59</td> </tr> <tr> <td>Women</td> <td>40 - 59</td> </tr> </tbody> </table>	Age Group	Gender	BMI Range	20 - 39	Men	40 - 59	Women	40 - 59	40 - 59	Men	40 - 59	Women	40 - 59								
Age Group	Gender	BMI Range																				
20 - 39	Men	40 - 59																				
	Women	40 - 59																				
40 - 59	Men	40 - 59																				
	Women	40 - 59																				
Split bar or column chart	<p>Best to compare categories in separate clusters. If labels are long, use horizontal bars instead of vertical columns. Easy tools: Bar and Column Charts in Google Sheets tutorialPower tool: Ch 12: Chart.js and Highcharts templates</p> <table border="1"> <thead> <tr> <th>Category</th> <th>Vegetable Type</th> <th>Value</th> </tr> </thead> <tbody> <tr> <td>Peppers</td> <td>Vegetables</td> <td>50</td> </tr> <tr> <td>Carrots</td> <td>Vegetables</td> <td>50</td> </tr> <tr> <td>Apples</td> <td>Vegetables</td> <td>50</td> </tr> <tr> <td>Bananas</td> <td>Vegetables</td> <td>50</td> </tr> <tr> <td>Grapes</td> <td>Vegetables</td> <td>50</td> </tr> </tbody> </table>	Category	Vegetable Type	Value	Peppers	Vegetables	50	Carrots	Vegetables	50	Apples	Vegetables	50	Bananas	Vegetables	50	Grapes	Vegetables	50			
Category	Vegetable Type	Value																				
Peppers	Vegetables	50																				
Carrots	Vegetables	50																				
Apples	Vegetables	50																				
Bananas	Vegetables	50																				
Grapes	Vegetables	50																				
Stacked bar or column chart	<p>Best to compare sub-categories, or parts of a whole. If labels are long, use horizontal bars instead of vertical columns. Easy tools: Bar and Column Charts in Google Sheets tutorialPower tool: Ch 12: Chart.js and Highcharts templates</p> <table border="1"> <thead> <tr> <th>Country</th> <th>Category</th> <th>Value</th> </tr> </thead> <tbody> <tr> <td>United States</td> <td>Normal or underweight</td> <td>57.2</td> </tr> <tr> <td>South Africa</td> <td>Normal or underweight</td> <td>57.2</td> </tr> <tr> <td>Italy</td> <td>Normal or underweight</td> <td>57.2</td> </tr> <tr> <td>Iran</td> <td>Normal or underweight</td> <td>57.2</td> </tr> <tr> <td>South Korea</td> <td>Normal or underweight</td> <td>57.2</td> </tr> <tr> <td>India</td> <td>Normal or underweight</td> <td>57.2</td> </tr> </tbody> </table>	Country	Category	Value	United States	Normal or underweight	57.2	South Africa	Normal or underweight	57.2	Italy	Normal or underweight	57.2	Iran	Normal or underweight	57.2	South Korea	Normal or underweight	57.2	India	Normal or underweight	57.2
Country	Category	Value																				
United States	Normal or underweight	57.2																				
South Africa	Normal or underweight	57.2																				
Italy	Normal or underweight	57.2																				
Iran	Normal or underweight	57.2																				
South Korea	Normal or underweight	57.2																				
India	Normal or underweight	57.2																				

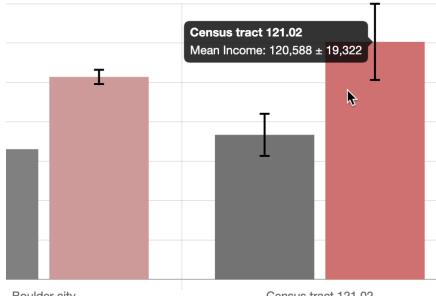
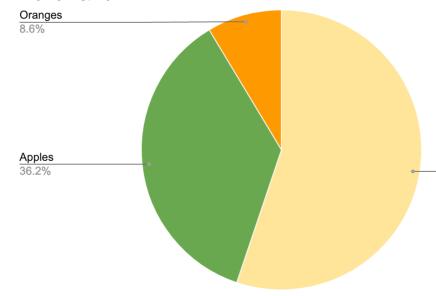
Chart	Best use and tutorials in this book
Error bars in bar or column chart	<p>Best to show margin of error bars when comparing categories side-by-side. If labels are long, use horizontal bars instead of vertical columns.</p> <p>Easy tool: TODO DECIDE Power tool: Ch 12: Chart.js and Highcharts templates</p> 
Histogram	<p>Best to show distribution of raw data, with number of values in each bucket.</p> <p>Easy tools: Histogram Chart in Google Sheets tutorial Power tool: Ch 12: Chart.js and Highcharts templates</p> 
Pie chart	<p>Best to show parts of a whole, but hard to estimate size of slices.</p> <p>Easy tools: Pie Chart in Google Sheets tutorial Power tool: Ch 12: Chart.js and Highcharts templates</p> 
Line chart	<p>Best to show continuous data, such as change over time.</p> <p>Easy tools: Line Chart in Google Sheets tutorial Power tool: Ch 12: Chart.js and Highcharts templates</p> 

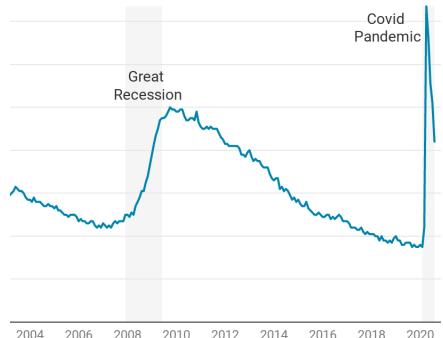
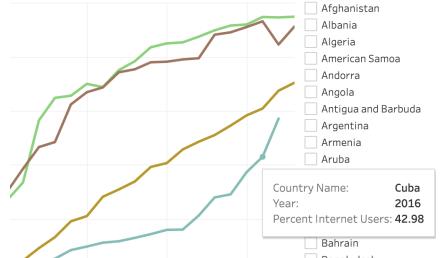
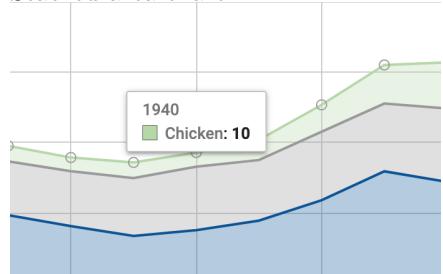
Chart	Best use and tutorials in this book
Annotated line chart 	Best to add notes or highlight data inside a chart, such as historical context in a line chart. Easy tools: Annotated Chart in Datawrapper tutorialPower tool: Ch 12: Chart.js and Highcharts templates
Filtered line chart 	Best to show multiple lines of continuous data, which users can toggle on and off. Easy tool: Filtered Line Chart in Tableau Public tutorial
Stacked area chart 	Best to show parts of a whole, with continuous data such as change over time. Easy tools: Stacked Area Chart in Google Sheets tutorialPower tool: TODO DECIDE
Range chart 	Best to show gaps between data points, such as inequalities. Easy tools: Range Chart in Datawrapper tutorialPower tool: TODO DECIDE

Chart	Best use and tutorials in this book
Scatter chart	Best to show the relationship between two variables, with each dot representing its X and Y coordinates. Easy tool: Scatter and Bubble Chart in Datawrapper tutorial or Scatter Chart in Tableau Public tutorial. Power tool: Ch 12: Chart.js and Highcharts templates
Bubble chart	Best to show the relationship between three or four sets of data, with XY coordinates, bubble size, and color. Easy tool: Scatter and Bubble Chart in Datawrapper tutorial Power tool: Ch 12: Chart.js and Highcharts templates
Sparklines	Best to compare data trends with tiny line or bar charts, aligned in a table column. Easy tool: Ch 9: Interactive Table with Sparklines in Datawrapper tutorial

See also the DataViz Catalogue site by Severino Ribecca for a more extensive collection of chart types and use cases.

TODO above: improve and standardize chart images, especially range chart

Chart Design Principles

There are *so* many different types of charts. However, just because data *can* be made into a chart does not necessarily mean that it *should* be turned into one. Before creating a chart, stop and ask: *Does a visualized data pattern really matter to your story?* Sometimes a simple table, or even text alone, can communicate the idea more effectively to your audience. Since creating a

well-designed chart requires time and effort, make sure it enhances your data story.

Although not a science, data visualization comes with a set of principles and best practices that serve as a foundation for creating truthful and eloquent charts. In this section, we'll identify some important rules about chart design. But you may be surprised to learn that some rules are less rigid than others, and can be "broken" when necessary to emphasize a point, as long as you are honestly interpreting the data. To begin to understand the difference, let's start by establishing a common vocabulary about charts.

Deconstruct a Chart

Let's take a look at Figure 7.1. It shows basic chart components that are shared among most chart types.

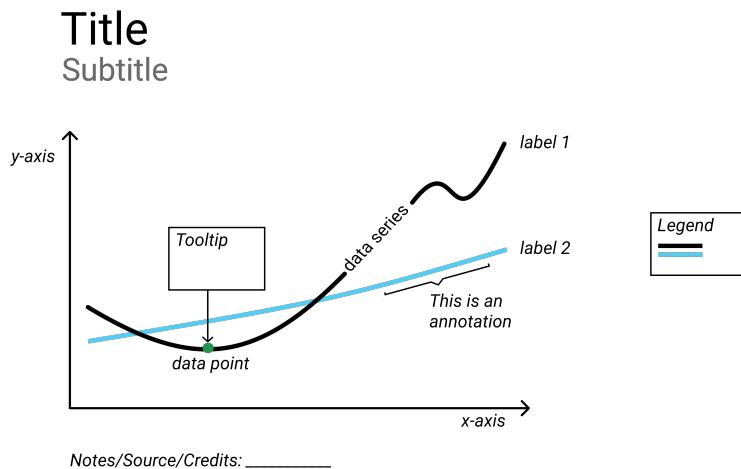


Figure 7.1: Common chart components.

A *title* is perhaps the most important element of any chart. A good title is short, clear, and tells a story on its own. For example, “Pandemic Hits Black and Latino Population Hardest”, or “Millions of Tons of Plastic Enter the Ocean Every Year” are both clear titles that quickly convey a larger story.

Sometimes your editor or audience will prefer a more technical title for your chart. If so, the two titles above could be changed, respectively, to “Covid-19 Deaths by Race in New York City, Spring 2020” and “Tons of Plastic Entering the Ocean, 1950–2020.”

A hybrid strategy is to combine a story-oriented title with a more technical subtitle, such as: “Pandemic Hits Black and Latino Population Hardest: Covid-19 Deaths by Race in New York City, Spring 2020.” If you follow this model,

make your subtitle less prominent than your title by decreasing its font size, or changing its font style or color, or both.

Horizontal (x) and vertical (y) *axes* define the scale and units of measure.

A *data series* is a collection of observations, which is usually a row or a column of numbers, or *data points*, in your dataset.

Labels and *annotations* are often used across the chart to give more context. For example, a line chart showing US unemployment levels between 1900 and 2020 can have a “Great Depression” annotation around 1930s, and “Covid-19 Impact” annotation for 2020, both representing spikes in unemployment. You might also choose to label items directly instead of relying on axes, which is common with bar charts. In that case, a relevant axis can be hidden and the chart will look less cluttered.

A *legend* shows symbology, such as colors and shapes used in the chart, and their meaning (usually values that they represent).

You should add *Notes*, *Data Sources*, and *Credits* underneath the chart to give more context about where the data came from, how it was processed and analyzed, and who created the visualization. Remember that being open about these things helps build credibility and accountability.

If your data comes with uncertainty (or margins of error), use *error bars* to show it, if possible. If not, accompany your chart with a statement like “the data comes with uncertainty of up to 20% of the value”, or “for geographies X and Y, margins of error exceed 10%”. This will help readers assess the reliability of the data source.

In interactive charts, a *tooltip* is often used to provide more data or context once a user clicks or hovers over a data point or a data series. Tooltips are great for complex visualizations with multiple layers of data, because they declutter the chart. But because tooltips are harder to interact with on smaller screens, such as phones and tablets, and are invisible when the chart is printed, only rely on them to convey additional, nice-to-have information. Make sure all essential information is visible without any user interaction.

Some Rules are More Important than Others

Although the vast majority of rules in data visualization are open to interpretation, as long as you honestly interpret the data, here are two rules that cannot be bent: zero-baselines for bar and column charts, and 100-percent baselines for pie charts.

Bar and Column Charts Must Begin at Zero

Bar and column charts use *length* and *height* to represent value, therefore their value axis *must start at the zero baseline*. This ensures that a bar twice the length of another bar represents twice its value. Figure 7.2 contrasts a good and a bad example. The same rule applies to area charts, which display filled-in area underneath the line to represent value. Starting the baseline at a number other than zero is a trick commonly used to exaggerate differences in opinion polls and election results, as we describe later in Chapter 15: Detect Lies and Reduce Data Bias

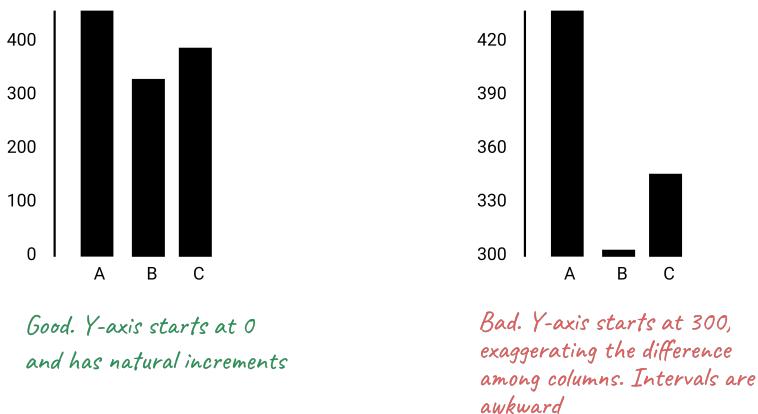


Figure 7.2: Start your bar chart at zero.

But the zero-baseline rule does *not* apply to line charts. Visualization expert Alberto Cairo points out that line charts represent values through the *position* and *angle* of the line, rather than its height or length. Starting a line chart at a number other than zero does *not* necessarily distort its encoded information because our eyes rely on its shape to determine its meaning, rather than its proximity to the baseline.¹ In fact, if you mistakenly force a line chart to begin at the zero baseline, you may inadvertently produce a misleading chart, as you will learn in the How to Lie with Charts section of Chapter 15.

TODO: Discuss with Ilya about adding a simpler chart comparison to demonstrate Cairo's primary argument, not the secondary argument about misleading charts

Pie Charts Represent 100%

Pie charts is one of the most contentious issues in data visualization. Most dataviz practitioners will recommend avoiding them entirely, saying that people

¹Cairo, *How Charts Lie*, 2019, p. 61.

are bad at accurately estimating sizes of different slices. We take a less dramatic stance, as long as you adhere to the recommendations we give in the next section.

But the one and only thing in data visualization that every single professional will agree on is that *pie charts represent 100% of the quantity*. If slices sum up to anything other than 100%, it is a crime. If you design a survey titled *Are you a cat or a dog person?* and include *I am both* as the third option, forget about putting the results into a pie chart.

Chart Aesthetics

Remember that you create a chart to help the reader understand the story, not to confuse them. Decide if you want to show raw counts, percentages, or percent changes, and do the math for your readers.

Avoid chart junk

Start with a white background and add elements as you see appropriate. You should be able to justify each element you add. To do so, ask yourself: Does this element improve the chart, or can I drop it without decreasing readability? This way you won't end up with so-called "chart junk" as shown in Figure 7.3, which includes 3D perspectives, shadows, and unnecessary elements. They might have looked cool in early versions of Microsoft Office, but let's stay away from them today. Chart junk distracts the viewer and reduces chart readability and comprehension. It also looks unprofessional and doesn't add credibility to you as a storyteller.



Figure 7.3: Chart junk distracts the viewer, so stay away from shadows, 3D perspectives, unnecessary colors and other fancy elements.

Do not use shadows or thick outlines with bar charts, because the reader might

think that decorative elements are part of the chart, and thus misread the values that bars represent.

The only justification for using three dimensions is to plot three-dimensional data, which has x, y, and z values. For example, you can build a three-dimensional map of population density, where x and y values represent latitude and longitude. In most cases, however, three dimensions are best represented in a bubble chart, or a scatterplot with varying shapes and/or colors.

Beware of pie charts

Remember that pie charts only show part-to-whole relationship, so all slices need to add up to 100%. Generally, the fewer slices—the better. Arrange slices from largest to smallest, clockwise, and put the largest slice at 12 o'clock. Figure 7.4 illustrates that.

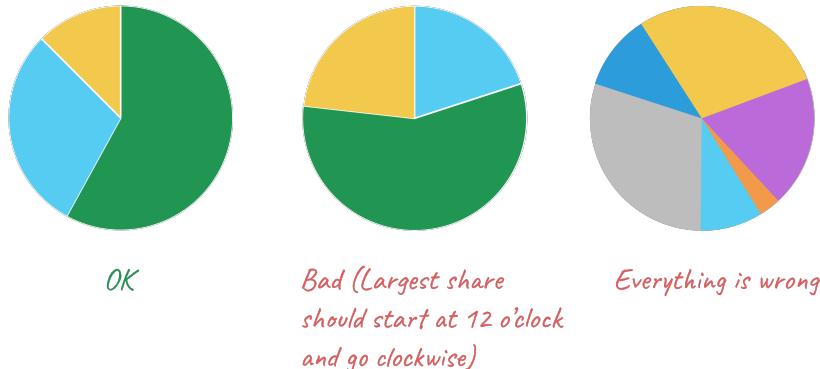


Figure 7.4: Sort slices in pie charts from largest to smallest, and start at 12 o'clock.

If your pie chart has more than five slices, consider showing your data in a bar chart, either stacked or split, like Figure 7.5 shows.

Don't make people turn their heads to read labels

When your column chart has long x-axis labels that have to be rotated (often 90 degrees) to fit, consider turning the chart 90 degrees so that it becomes a horizontal bar chart. Take a look at Figure 7.6 to see how much easier it is to read horizontally-oriented labels.

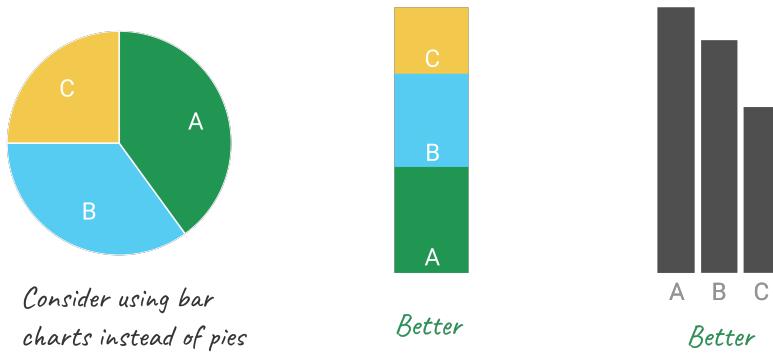


Figure 7.5: Consider using bar charts instead of pies.

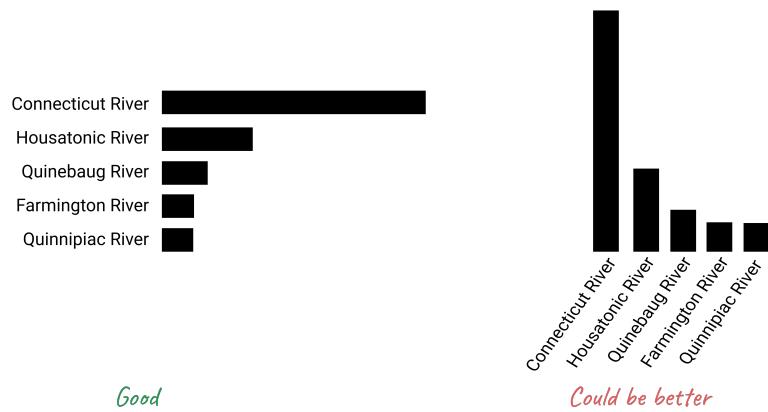


Figure 7.6: For long labels, use horizontal bar charts.

Arrange elements logically

If your bar chart shows different categories, consider ordering them, like is shown in Figure 7.7. You might want to sort them alphabetically, which can be useful if you want the reader to be able to quickly look up an item, such as their town. Ordering categories by value is another common technique that makes comparisons possible. If your columns represent a value of something at a particular time, they have to be ordered sequentially, of course.

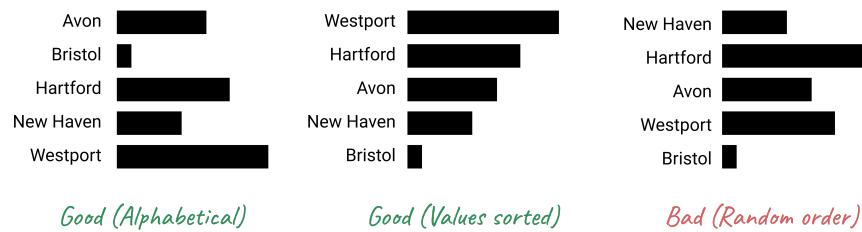


Figure 7.7: For long labels, use horizontal bar charts.

Do not overload your chart

When labelling axes, choose natural increments that space equally, such as [0, 20, 40, 60, 80, 100], or [1, 10, 100, 1000] for a logarithmic scale. Do not overload your scales. Keep your typography simple, and use (but do not overuse) **bold type** to highlight major insights. Consider using commas as thousands separators for readability (1,000,000 is much easier to read than 1000000).

Be careful with colors

In this section, we would like to briefly introduce three important rules about colors. First, remember that in most cases monochromatic (single-hue) charts suffice, and there may be no need to introduce the extra dimension of color at all.

Second, refer to the color wheel and standard harmony rules when choosing your palette. Consider the rule of complementary colors—opposites in the color wheel—to find color pairs, such as blue and orange or yellow and purple. Analogous colors, or neighbors in the color wheel, make good palettes, such as orange, red, and pink.

Third, stay away from pure saturated colors and instead choose their “earthier” versions, such as olive green instead of bright green, or navy instead of neon blue.

Once you have chosen the color palette for your visualization, ask yourself:

1. Is there a conflict of meaning between colors and the phenomenon they represent? Am I using red to represent profit or green to represent death rate? This question is complex as colors carry different associations for different social groups and cultures, but try to exercise your best sensitivity.
2. Can people with color blindness interpret your chart? Palettes that contain reds and greens, or yellows and blues can be challenging. Consider using Color Oracle or another simulator to make sure your visualization is accessible.
3. Will the colors be distinguishable in black-and-white? Even if you don't expect viewers printing your chart, they may. You can use Color Oracle or another simulator to check that your colors have different brightness levels and look distinguishable in grayscale. Figure 7.8 shows some good and bad examples of color use.

The use of color is a complex topic, and there are plenty of books and research devoted to it. For an excellent overview, see Lisa Charlotte Rost's "A Friendly Guide to Colors in Data Visualization" and "How to Pick More Beautiful Colors for Your Data Visualizations," both on the Datawrapper blog.²

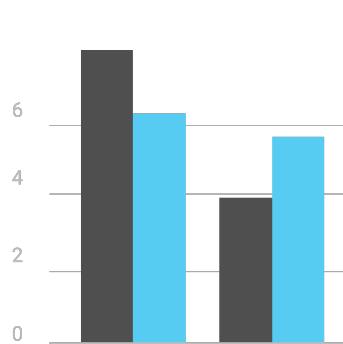
If you follow our advice, you should end up with a de-cluttered chart as shown in Figure 7.9. Notice how your eyes are drawn to the bars and their corresponding values, not bright colors or secondary components like the axes lines.

TODO: Compare our rules and recommendations to different dataviz style guides, such as Urban Institute, and cite them <http://urbaninstitute.github.io/graphics-styleguide/>

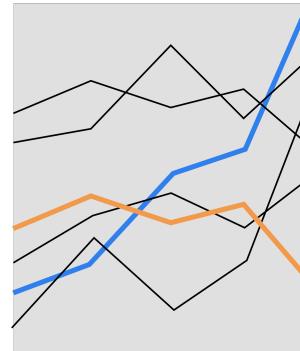
Google Sheets Charts

In this section, you'll learn about the pros and cons of creating interactive charts in Google Sheets, the powerful spreadsheet tool we introduced in Chapter 3. Google Sheets has many advantages for newcomers to data visualization. First, Google Sheets allows you to clean, analyze, share, and publish charts, all in the same platform. One tool does it all, which makes it easier to organize your work by keeping it all together in one place. Second, Google Sheets is familiar and easy to learn to many users, so it will help you to *quickly* create good-looking interactive charts. See all of the types of charts you can create with Google Sheets. Although some people export charts as static images in *JPG* or *PNG*

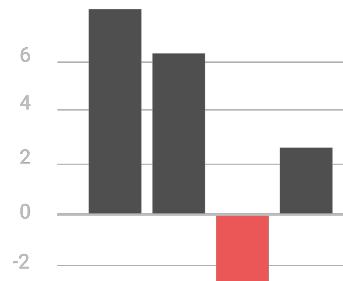
²Lisa Charlotte Rost, "Your Friendly Guide to Colors in Data Visualisation," Chartable: A Blog by Datawrapper, July 31, 2018, <https://blog.datawrapper.de/colorguide/>; Lisa Charlotte Rost, "How to Pick More Beautiful Colors for Your Data Visualizations," Chartable, accessed October 21, 2020, <https://blog.datawrapper.de/beautifulcolors/index.html>



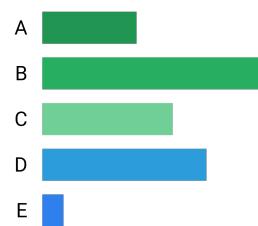
Good. Blue allows to distinguish between series



Good. Colored lines are distinguishable



Fine. Red adds emphasis, but is not absolutely necessary



Bad. Colors are too similar, and are not needed for a separated bar chart

Figure 7.8: Don't use colors just for the sake of it.



Figure 7.9: Make sure important things catch the eye first.

format, this chapter focuses on creating interactive charts that display more info about your data when you hover over them in your browser. Later, you'll learn how to embed an interactive chart on your website in Chapter 10.

But Google Sheets also has limitations. First, there is no easy way to cite or link to your source data inside a Google Sheets chart, so you will need to add this key information to the text of a web page that contains your embedded interactive chart. Second, you cannot add text annotations or highlight specific items inside your charts. Finally, you are limited in customizing your chart design, especially tooltips when hovering over data visualizations. If Google Sheets does not meet your needs, refer back to Table 7.1 for other tools and tutorials, such as Datawrapper, Tableau Public, and Chart.js and Highcharts code templates.

In the next two sections, we'll review the most appropriate cases to use bar and column charts, followed by pie, line, and area charts. Each section features hands-on examples and step-by-step instructions with sample datasets to help you learn.

- Bar and Column Charts

Before you begin, be sure to review the pros and cons of designing charts with Google Sheets in the prior section. In this section, you'll learn how to create bar and column charts, the most common visualization methods to compare values across categories. We'll focus on why and how to create three different types: grouped, split, and stacked. For all of these, we blend the instructions for bar and column charts because they're essentially the same, though oriented

in different directions. If your data contains long labels, create a horizontal bar chart, instead of a vertical column chart, to give them more space for readability.

Grouped Bar and Column Charts

A grouped bar or column chart is best to compare categories side-by-side. For example, if you wish to emphasize gender differences in obesity across age brackets, then format the male and female data series together in vertical columns in your Google Sheet, as shown in Figure 7.10. Now you can easily create a grouped column chart to displays these data series side-by-side, as shown in Figure 7.11.

	A	B	C
1	Age Range	Men	Women
2	20 - 39	31.6	37
3	40 - 59	37.2	44.6
4	60 and over	37.5	39.4

Figure 7.10: To create a grouped bar or column chart, format each data series vertically in Google Sheets.

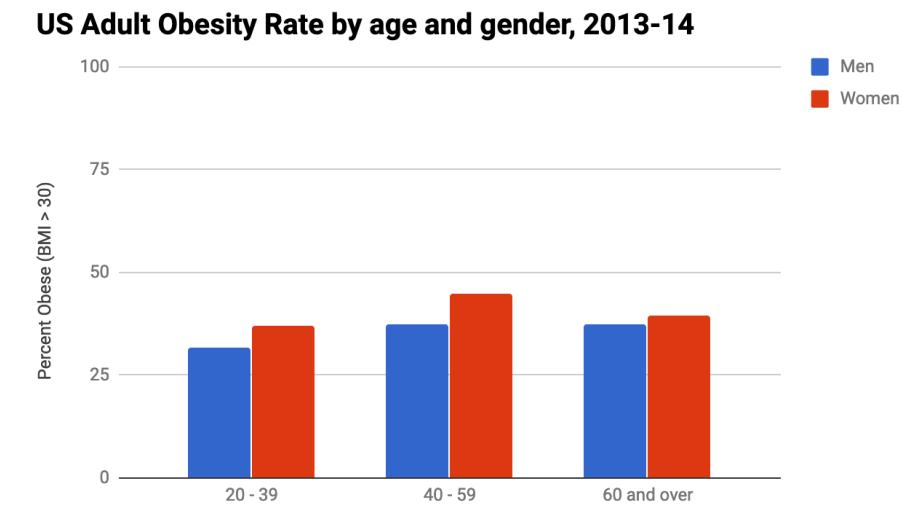


Figure 7.11: Grouped Column chart: Explore the interactive version. Data from StateOfObesity.org.

To create your own interactive grouped column (or bar) chart, use our template and follow these steps.

1. Open our Grouped Column chart template in Google Sheets with US obesity data by gender and age. Sign in to your account, and go to *File* > *Make a Copy* to save a version you can edit to your own Google Drive, as shown in Figure 7.12.

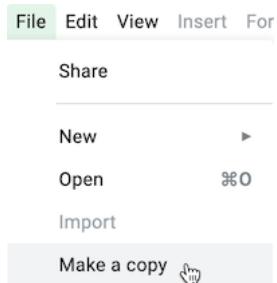


Figure 7.12: Make your own copy of the Google Sheet template.

2. To remove the current chart from your copy of the spreadsheet, float your cursor to the top-right corner of the chart to make the 3-dot kebab menu appear, and select *Delete*, as shown in Figure 7.13.

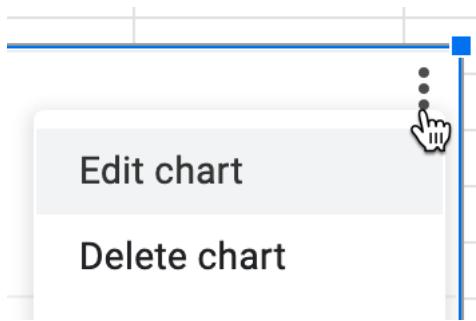


Figure 7.13: Float cursor in top-right corner of the chart to make the 3-dot kebab menu appear, and select Delete.

Note: Another name for the 3-dot menu symbol is the “kebab menu” because it resembles Middle Eastern food cooked on a skewer, in contrast to the three-line “hamburger menu” on many mobile devices, as shown in Figure 7.14. Software developers must be hungry.



Hamburger menu



Kebab menu

Figure 7.14: Distinguish between the hamburger versus kebab menu icons.

3. Format your data to make each column a data series (such as male and female), as shown in Figure 7.10, which means it will display as a separate color in the chart. Feel free to add more than two columns.
4. Use your cursor to select only the data you wish to chart, then go to the *Insert* menu and select *Chart*, as shown in Figure 7.15.

A screenshot of the Microsoft Word ribbon. The 'Insert' tab is highlighted in green. Below the ribbon, a table is selected. The first row of the table has the header 'Age Range' and 'Men'. The second row contains the data '20 - 39'. The third row contains '40 - 59'. The fourth row contains '60 and over'. To the right of the table, there is a dropdown menu with several options: '4 Rows', '4 Rows', '3 Column', '3 Column', 'Cells are', 'Cells are', and 'Chart'. The 'Chart' option is highlighted.

Figure 7.15: Select your data and Insert the Chart.

7. In the Chart Editor, change the default selection to *Column chart*, with *Stacking none*, to display Grouped Columns, as shown in Figure 7.16. Or select *Horizontal bar chart* if you have longer labels.
8. To customize title, labels, and more, in the Chart Editor select *Customize*, as shown in Figure 7.17. Also, you can select the chart and axis titles to edit them.

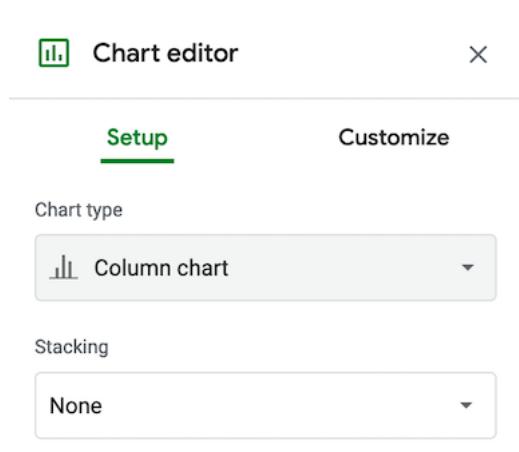


Figure 7.16: Change the default to Column chart, with Stacking none.

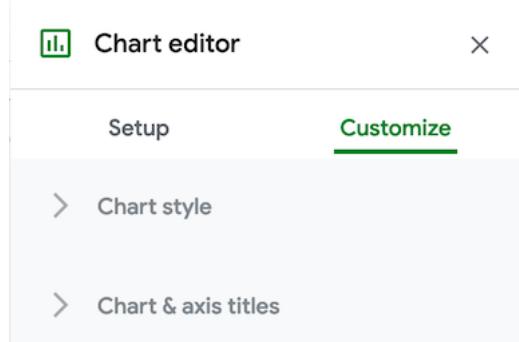


Figure 7.17: Select Customize to edit title, labels, and more.

- To make your data public, go to the upper-right corner of your sheet to click the Share button, and in the next screen, click the words “Change to anyone with the link,” as shown in Figure 7.18. This means your sheet is no longer Restricted to only you, but can be viewed by anyone with the link. See additional options.

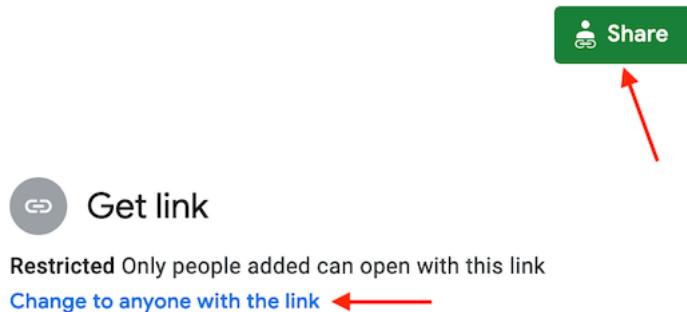


Figure 7.18: Click the Share button and then click *Change to anyone with the link* to make your data public.

- To embed an interactive version of your chart in another web page, click the kebab menu in the upper-right corner of your chart, and select Publish Chart, as shown in Figure 7.19. In the next screen, select Embed and press the Publish button. See Chapter 10: Embed on the Web to learn what to do with the iframe code.

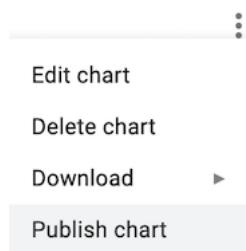


Figure 7.19: Select Publish Chart to embed an interactive chart on another web page.

Unfortunately Google Sheets functionality is very limited when it comes to displaying error bars, or uncertainty. You can only assign either constant numbers or percent values as error bar values to individual series (not data points). To do so, in *Chart editor*, select *Customize* tab, scroll down to *Series* and select a series from the dropdown. Check *Error bars*, and customize its value as either

percent or a constant value, as shown in Figure @ref(fig:images/07-chart/chart-error-bar.png). This setting will then be applied to all data points from that series.

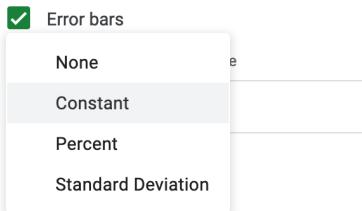


Figure 7.20: Google Sheets is very limited when it comes to setting error bars.

Since there is no easy way to cite or link to your source data inside a Google Sheets chart, you will need to add this information to the text of the web page that contains your interactive chart. Remember that citing your data sources adds credibility to your work.

Split Bar and Column Charts

A split column (or bar) chart is best to compare categories in separate clusters. For example, imagine you wish to emphasize calorie counts for selected foods offered at two different restaurants, Starbucks and McDonalds. Format the restaurant data in vertical columns in your Google Sheet, as shown in Figure 7.21. Since food items are unique to each restaurant, only enter calorie data in the appropriate column, and leave other cells blank. Now you can easily create a split bar (or column) chart that displays the restaurant data in different clusters, as shown in Figure 7.22. Unlike the grouped column chart previously shown in Figure 7.11, here the bars are separated from each other, because we do not wish to draw comparisons between food items that are unique to each restaurant. Also, our chart displays horizontal bars (not columns) because our some data labels are long.

	A	B	C
1	Fast Food items	Starbucks	McDonalds
2	Mocha Frappuccino (24-ounce, 2% milk, whip cream)	500	
3	White Hot Chocolate (20-ounce, 2% milk, whip cream)	710	
4	Big Mac		540
5	Double Quarter Pounder with cheese		770

Figure 7.21: To create a split bar (or column) chart, format each data series vertically, and leave cells blank where appropriate.

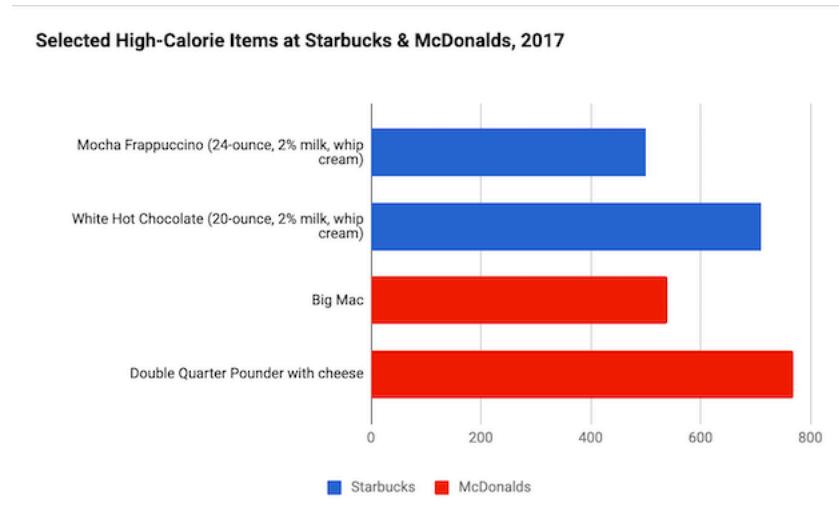


Figure 7.22: Split bar chart: Explore the full-screen interactive version. Data from Starbucks and McDonalds.

Create your own version using our Split Bar Chart in Google Sheets template with Starbucks and McDonalds data. Organize each data series vertically so that it becomes its own color in the chart. Leave cells blank when no direct comparisons are appropriate. The remainder of the steps are similar to the grouped column chart tutorial above.

Stacked Bar and Column Charts

Stacked column (or bar) charts are best to compare subcategories, or parts of a whole. For example, if you wish to compare the percentage of overweight residents across nations, format each weight-level data series in vertical columns in your Google Sheet, as shown in Figure 7.23. Now you can easily create a stacked column (or bar) chart that displays comparisons of weight-level subcategories across nations, as shown in Figure 7.24. Often it's better to use a stacked chart instead of multiple pie charts, because people can see differences more precisely in rectangular stacks than in circular pie slices.

Create your own version using our Stacked Column Chart in Google Sheets template with international weight-level data. Organize each data series vertically so that it becomes its own color in the chart. In the Chart Editor window, choose *Chart Type > Stacked column chart* (or choose *Stacked bar chart* if you have long data labels). The rest of the steps are similar to the ones above.

To change the color of a data series (for example, to show Overweight category in red), click the kebab menu in the top-right corner of the chart, then go to

	A	B	C	D
1	Nation	Underweight	Normal weight	Overweight
2	United States	2	35.2	62.8
3	South Africa	8.6	46.2	45.1
4	Italy	3.4	52.6	44
5	Iran	5.7	51.5	42.8
6	Brazil	4	55.4	40.6
7	South Korea	4.7	63.2	32.1
8	India	32.9	62.5	4.5

Figure 7.23: To create a stacked column (or bar) chart, format each data series vertically in Google Sheets.

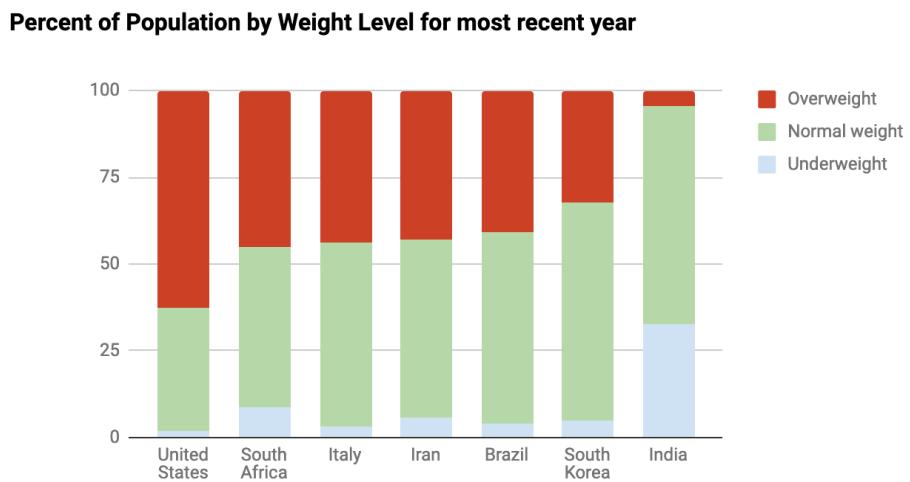


Figure 7.24: Stacked column chart: Explore the interactive version. Data from WHO and CDC.

Edit Chart > Customize > Series. Then choose the appropriate series from the dropdown menu, and set its color in the dropdown menu, as shown in Figure 7.25.

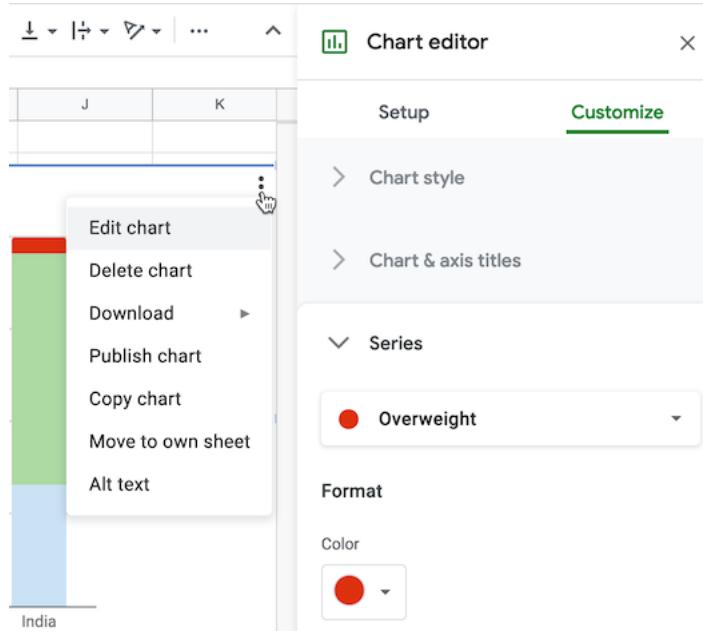


Figure 7.25: To edit a column color, select *Edit Chart - Customize - Series*.

- Histograms

A histogram chart is best for showing the distribution of raw data, with the number of values displayed in each bucket. Although a histogram may look similar to a column chart, the two are different. Since histograms show *continuous* data, you can adjust the bucket ranges to explore frequency patterns. For example, you can shift histogram buckets from 0-1, 1-2, 2-3, etc. to 0-2, 2-4, etc. But column (and bar) charts show *categorical* data (such as the number of apples, bananas, etc.) so their ranges cannot be adjusted. Figure 7.26 shows a histogram of average daily calorie consumption in 174 countries in 2006–2008 based on data from the United Nations Food and Agriculture Organization.

Histograms are designed to represent distributions, not individual values. Figure 7.26 shows two peaks, one of 2,100–2,300 calories and the second one at 2,700–3,100 calories (two consecutive buckets). We can also see that daily calorie consumption in five countries does not exceed 1,900, and exceeds 3,700 calories in three countries. Histograms by themselves, without annotations, don't tell us what those countries are.

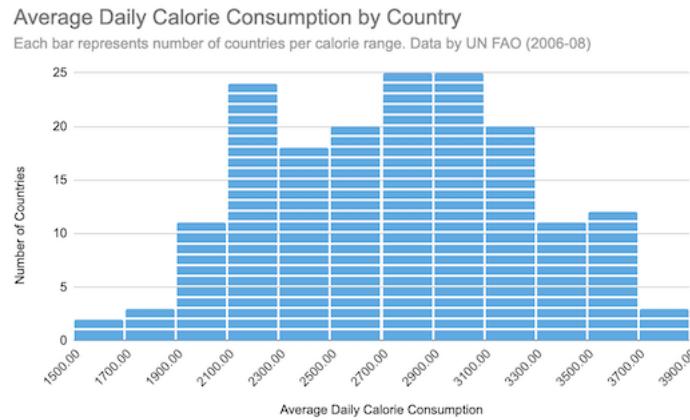


Figure 7.26: Histogram: Explore the full-screen interactive version.

Note: You can view and copy the Google Sheet with the calorie consumption data. You can use *Sort sheet A->Z* function to see that five countries with lowest per-capita calorie consumption are Eritrea (1,590), Burundi (1,680), Comoros (1,840), Haiti (1,850), and Zambia (1,880). The three highest are Greece (3,710), USA (3,750), and Austria (3,800). Note that the dataset is over a decade old and things may be different today.

Histograms use *buckets* or *bins*, which are predefined ranges of values, and count how many data points (usually rows in your dataset) fall within each interval. Each interval's count is represented by a bar. Intervals should not overlap, and we recommend you make them all equal size. Consider making your ranges “pretty”, that is, using whole numbers such as multiples of 5 (5, 10, 15, 20) or 100 (1500, 1600, 1700, 1800) for breakpoints, but only if that makes sense for your data distribution.

Now, let's look at how to create the histogram in Figure 7.26 in Google Sheets. Our dataset contains two columns (*Country* and *Average Daily Calorie Consumption*) and 174 records, as shown in Figure 7.27. But in reality you only need one column that lists all values to build a histogram.

Select a column with values, and go to *Insert > Chart*. Google Sheets will likely automatically choose *Histogram chart* as the *Chart type* in Chart editor, but if not, use the dropdown and set it manually (you will find *Histogram* under the *Other* category). While some readers know what histograms represent, it may be useful to add a y-axis label (eg *Number of countries*) and a subtitle (eg *Each bar represents number of countries per calorie range*). You can add both from the *Customize* tab in the Chart editor.

To further assist the reader in interpreting the histogram, you can break down the column into individual items (in our case, countries), which will appear as

	A	B
1	Country	Average Daily Calorie Consumption
2	Albania	2890
3	Algeria	3090
4	Angola	1960
5	Antigua and Barbuda	2330
6	Argentina	3030
7	Armenia	2260
8	Australia	3220
9	Austria	3800
10	Azerbaijan	3020

Figure 7.27: To create a histogram, you only need one column with numeric values.

blocks with white boundaries. You can do that, and manually set the range of each bucket in the Chart editor (*Customize > Histogram > Show item dividers* and *Customize > Histogram > Bucket size*). Larger intervals will contain more datapoints and will also appear wider in the chart as fewer larger intervals are needed to cover the entire range. Smaller intervals will contain fewer datapoints each, and will appear narrower.

Unfortunately, currently there is no way to get rid of decimal points in the x-axis labels in a Google Sheets histogram, even though all breakpoints may be integers.

TODO: DISCUSS with AMELIA and ILYA the organization of the Histogram section and its sample data. See Meeting Notes

- Pie, Line, and Area Charts

Before starting this section, be sure to review the pros and cons of designing charts with Google Sheets, as well as beginner-level step-by-step instructions for creating bar and column charts, in the previous sections of this chapter. In this section, you'll learn why and how to use Google Sheets to build three more types of interactive visualizations: pie charts (to show parts of a whole), line charts (to show change over time), and stacked area charts (to combine showing parts of a whole, with change over time). If Google Sheets or these chart types do not meet your needs, refer back to Table 7.1 for other tools and tutorials.

Pie Charts

Some people use pie charts to show parts of a whole, but we urge caution with this type of chart for reasons explained further below. For example, if you wish to show the number of different fruits sold by a store in one day, as a proportion of total fruit sold, then format the labels and values in vertical columns in your Google Sheet, as shown in Figure 7.28. Values can be expressed as either raw numbers or percentages. Now you can easily create a pie chart that displays these values as colored slices of a circle, as shown in Figure 7.29. Viewers can see that bananas made up slightly over half of the fruit sold, followed by apples and oranges.

	A	B
1	Bananas	32
2	Apples	21
3	Oranges	5

Figure 7.28: To create a pie chart, format the data values vertically in Google Sheets.

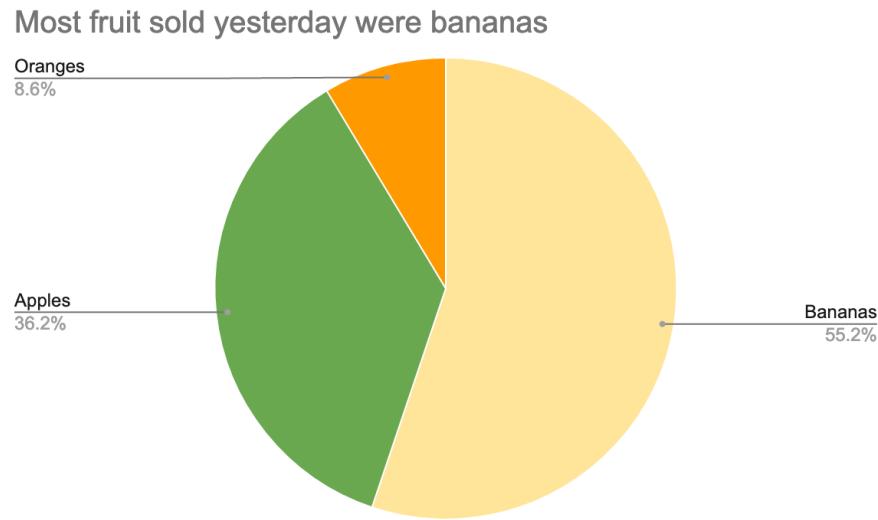


Figure 7.29: Pie chart: Explore the interactive version. Data is fictitious.

But you need to be careful when using pie charts, as we described in the Chart

Design section of this chapter. First, make sure your data adds up to 100 percent. If you created a pie chart that displayed *some* but *not all* of the fruits sold, it would not make sense. Second, avoid creating too many slices, since people cannot easily distinguish smaller ones. Ideally, use no more than 5 slices in a pie chart. Finally, start the pie at the top of the circle (12 o'clock) and arrange the slices clockwise, from largest to smallest.

Create your own version using our Pie Chart in Google Sheets template. The steps are similar to those in prior Google Sheets chart tutorials in this chapter. Go to *File > Make a Copy* to create a version you can edit in your Google Drive. Select all of the cells and go to *Insert > Chart*. If Google Sheets does not correctly guess that you wish to create a pie chart, then in the Chart editor window, in the Setup tab, select *Pie chart* from the *Chart type* dropdown list.

Note that slices are ordered the same way they appear in the spreadsheet. Select the entire sheet and *Sort* the values column from largest to smallest, or from Z to A. In *Customize* tab of the Chart editor, you can change colors and add borders to slices. Then add a meaningful title and labels as desired.

Line Charts

A line chart is the best way to represent continuous data, such as change over time. For example, imagine you wish to compare the availability of different meats per capita in the US over the past century. In your Google Sheet, organize the time units (such as years) into the first column, since these will appear on the horizontal X-axis. Also, place each data series (such as beef, pork, chicken) alongside the vertical time-unit column, and each series will become its own line, as shown in Figure 7.30. Now you can easily create a line chart that emphasizes each data series changed over time, as shown in Figure 7.31. In the US, the amount of chicken per capita steadily rose and surpassed pork and beef around 2000.

Create your own version using our Line Chart in Google Sheets template. The steps are similar to those in prior Google Sheets chart tutorials in this chapter. Go to *File > Make a Copy* to create a version you can edit in your Google Drive. Select the data, and choose *Insert > Chart*. If Google Sheets does not correctly guess that you wish to create a line chart, in the Chart editor, Setup tab, select *Line chart* from the *Chart type* dropdown list.

Sidebar: Tables and charts approach time-series data in opposite directions. When designing a table, the proper method is to place dates horizontally as column headers, so that we read them from left-to-right, like this:³

³Few, *Show Me the Numbers*, p. 166

	A	B	C	D
1	Year	Beef	Pork	Chicken
2	1910	48.5	38.2	11
3	1920	40.7	39	9.7
4	1930	33.7	41.1	11.1
5	1940	37.8	45.1	10
6	1950	44.6	43	14.3
7	1960	59.1	48.6	19.1
8	1970	79.6	48.1	27.4
9	1980	72.1	52.1	32.7
10	1990	63.9	46.4	42.4
11	2000	64.5	47.8	54.2
12	2010	56.7	44.3	58
13	2017	54	47	64

Figure 7.30: To create a line chart, format the time units and each data series in vertical columns.

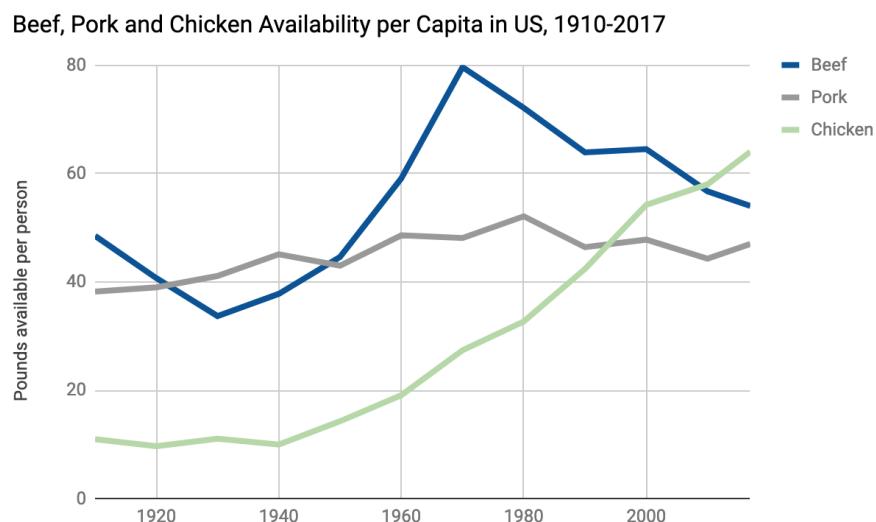


Figure 7.31: Line chart: Explore the interactive version. Data from US Department of Agriculture.

Year	2000	2010	2020
Series1
Series2

But when designing a line chart in Google Sheets and several other tools, we organize the spreadsheet by placing the dates vertically down the first column, so that the tool reads them as labels for a data series, like this:

Year	Series1	Series2
2000
2010
2020

To convert data from tables to charts, learn how to transpose rows and columns in Chapter 5: Clean Up Messy Data.

Stacked Area Charts

Area charts resemble line charts with filled space underneath. The most useful type is a stacked area chart, which is best for combining two concepts from above: showing parts of the whole (like a pie chart) and continuous data over time (like a line chart). For example, the line chart above shows how the availability of three different meats changed over time. However, if you also wish to show how the total availability of these combined meats went up or down over time, it's hard to see this in a line chart. Instead, use a stacked line chart to visualize the availability of each meat *and* the total combined availability per capita over time. Stacked line charts show both aspects of your data simultaneously.

To create a stacked area chart, organize the data in the same way as you did for the line chart in Figure 7.30. Now you can easily create a stacked line chart that displays the availability of each meat—and their combined total—over time, as shown in Figure 7.32. Overall, we can see that total available meat per capita increased after the 1930s Depression, and chicken steadily became a larger portion of the total after 1970.

Create your own version using our Stacked Area Chart in Google Sheets template. The steps are similar to those in prior Google Sheets chart tutorials in this chapter. Go to *File > Make a Copy* to create a version you can edit in your Google Drive. Set up the data exactly as you would for a line chart, with the first column for time units in the X-axis, and place each data series in its own column. Select the data, and choose *Insert > Chart*. In the Chart editor, in tab Setup, select *Stacked area chart* from the Chart type dropdown list.

Beef, Pork and Chicken Availability per Capita in US, 1910-2017

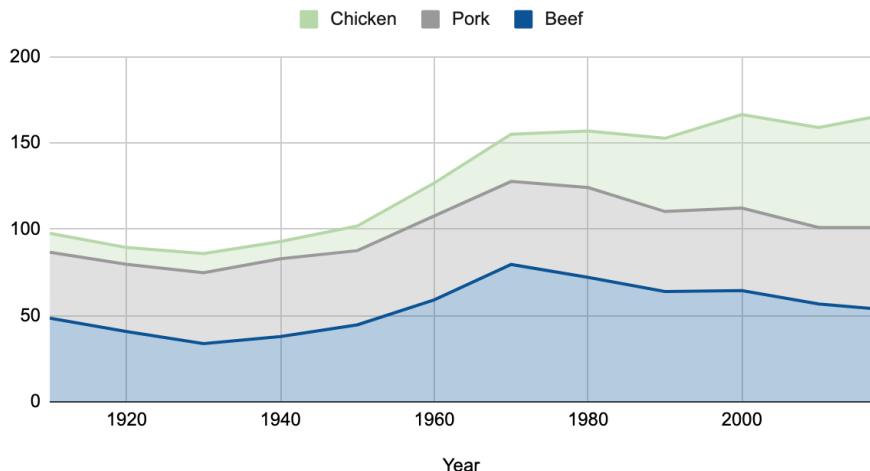


Figure 7.32: Stacked area chart: Explore the interactive version. Data from US Department of Agriculture.

Now that you've built several basic charts in Google Sheets, in the next section we'll build some slightly more advanced charts in a different tool, Datawrapper.

Datawrapper Charts

Another free and collaborative tool for creating interactive charts is Datawrapper, which has several advantages over Google Sheets. First, you can start creating in Datawrapper right away in your browser, even without creating an account, and its four-step process is intuitive for many new users. Second, you can add credit bylines, data sources, and even allow visitors to download your data from links inside your Datawrapper visualizations that you publish online, which makes your work more credible and accessible. Third, Datawrapper supports a wider array of interactive chart types than Google Sheets, as well as other visualizations we'll discuss later, such as maps in chapter 8 and tables in chapter 9. You can build all of the basic charts we've constructed so far in this chapter, as well as three new types where Datawrapper stands out: annotated charts, range charts, and scatter and bubble charts. Later, you'll learn how to embed interactive Datawrapper charts on your website in Chapter 10. Finally, we highly recommend the Datawrapper Academy support pages, the extensive gallery of examples, and well-designed training materials to help you and others learn beyond the basics covered here.

While Datawrapper is fabulous, it cannot fulfill all of your data visualization needs. You'll still need a spreadsheet tool, such as Google Sheets, to organize and analyze your data as described in Chapter 3, record your detailed source notes and save raw data files as described in Chapter 4, and clean up your data as described in Chapter 5. While Datawrapper can transpose data (swap the rows and columns), it cannot create pivot tables or lookup and merge data from different columns, which Google Sheets and other spreadsheet tools can do. The main reason we start this chapter with Google Sheets is because it's simpler for newcomers to use one tool for both spreadsheets and basic charts.

Now you're ready to use Datawrapper to create new types of charts that step beyond the basics. But if Datawrapper or the chart types in this section do not meet your needs, refer back to Table 7.1 for other tools and tutorials, or prior chapters on spreadsheets, sourcing, and cleaning up data.

- Annotated Charts

An annotated chart is best to highlight specific data or add contextual notes inside the visualization. Well-designed annotations can help answer the “so what?” question by briefly noting the significance of data in the chart, with greater detail in the sentences or paragraphs that follow. Be cautious with annotations, because it's important to avoid adding unnecessary “chart junk,” as described in Chart Design Principles section of this chapter.

You can add annotations to any chart created with Datawrapper, and we'll illustrate how with a line chart about US unemployment data from 2000-2020, since adding a bit of historical context often helps readers to better understand data stories about change over time. To create a line chart in Datawrapper, organize your data the same way you did in the Google Sheets line chart tutorial above. Place units of time (such as months-years) in the first column, and numerical data values (such as the unemployment rate) in the second column. Now you're ready to create an interactive line chart with annotations, as shown in Figure 7.33. Since 2000, the unemployment rate has peaked three times, but the tallest peak occurred during the 2020 economic crisis during the Covid pandemic.

Create your own annotated line chart in Datawrapper by following this tutorial:

1. Open the US Unemployment Seasonally Adjusted 2000-2020 sample data in Google Sheets and go to *File > Make a Copy* to create your own version in your Google Drive. Or download the sample data in Excel format to your computer.
2. Open Datawrapper in your browser and click *Start Creating*. We recommend that you create a free account to better manage your visualizations, but it's not required.

US Unemployment Rate, Seasonally Adjusted, 2000-2020

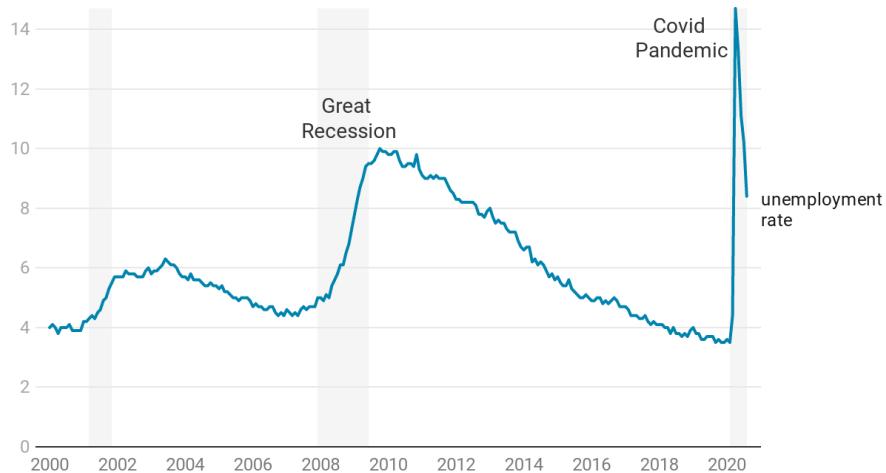


Chart: HandsOnDataViz.org · Source: US Federal Reserve Open Data · Created with Datawrapper

Figure 7.33: Line chart with annotation: Explore the interactive version. Data from US Federal Reserve Open Data.

3. In the *Upload Data* screen, click *Import Google Spreadsheet* and paste the link to the data in the shared Google Sheet above, as shown in Figure 7.34, then click *Proceed*. To upload a Google Sheet, the Share setting must be changed from *Private*, the default setting, to *Anyone with the link can view* at minimum. Also, if you update cells in your Google Sheet, they will be updated automatically in a linked Datawrapper chart, but not after your chart is published online. Alternatively, you can upload data by copying and pasting it into the data table window, or uploading an Excel or CSV file.
 4. In the *Check and Describe* screen, inspect your data to make sure that numbers appear in blue, dates in green, and text in black type, and click *Proceed*.

Tip: If needed, at the bottom of the *Check and Describe* screen there is a button that will transpose your data (swap rows and columns), which is useful in cases where the data you receive is organized in the opposite direction from what Datawrapper expects. But our sample data does not need to be transposed, since it's organized correctly.

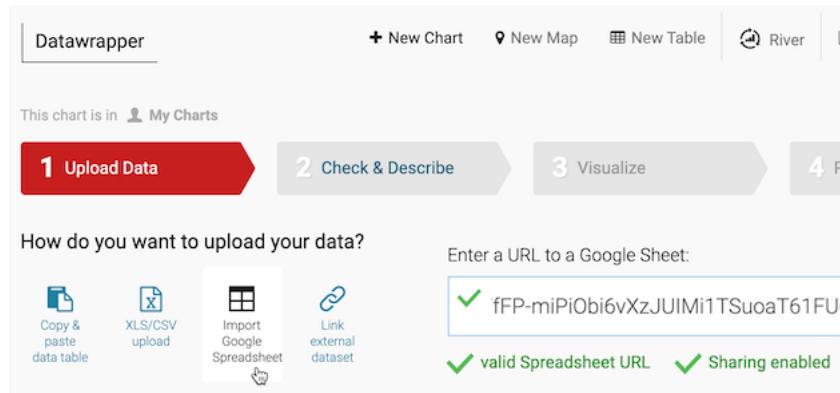


Figure 7.34: To upload data from a shared Google Sheet, click the button and paste the link.

5. In the *Visualize* screen, Datawrapper will attempt to guess the chart type you desire, based on the data format. If you entered our sample data correctly, it will correctly display a line chart. But if not, you can select a different chart type.
6. Click the *Annotate* tab near the top-left of the *Visualize* screen. Type in a meaningful title, such as “US Unemployment Rate, Seasonally Adjusted, 2000-2020.” Also, add a data source, such as “US Federal Reserve Open Data”, and a link to the source, such as the shared Google Sheet or the Federal Reserve Open Data web page. Finally, in the byline line, add your name or organization to credit the people who created this chart. You’ll see all of these details and links appear automatically at the bottom of your chart, to add credibility to your work.
7. Scroll down further in the *Annotate* tab to the *Text annotations* section, and click the button to add one to the chart. Type “Great Recession” into the text field of your first annotation, and move your cursor to place it around the unemployment peak in 2009, as shown in Figure 7.35. This helps readers to place the Great Recession in historical context. Click to add another text annotation, type “Covid Pandemic” into the text field, and place it around the second unemployment peak in early 2000 for comparison. You can fine-tune the positioning of a label by typing its year-month code into the x-axis coordinate text field, or its unemployment rate into the y-axis text field. To change the appearance or delete an annotation, click its downward arrow symbol for options.
8. Scroll down further in the *Annotate* tab to the *Highlight value ranges* section, and click the button to add one to the chart. Click inside the chart to “draw” a bar from December 2007 to June 2009, as shown in Figure 7.36.

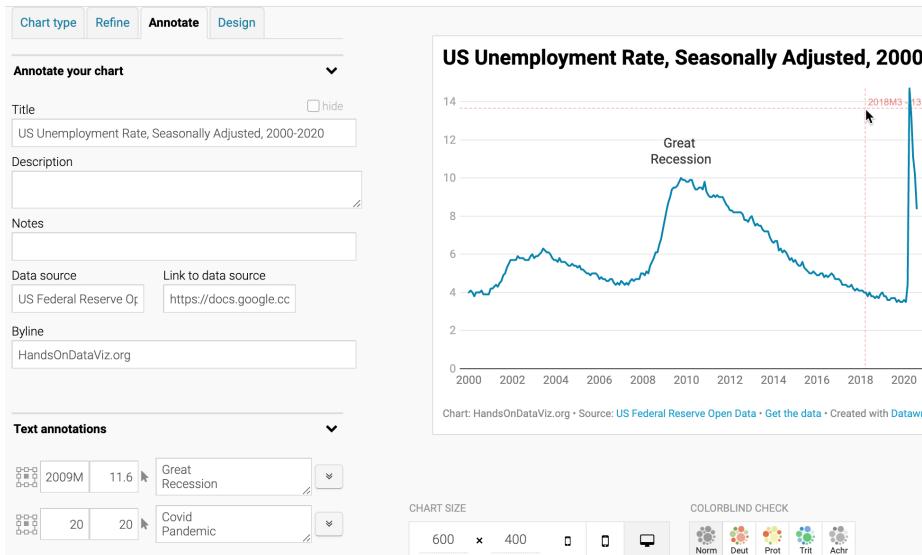


Figure 7.35: Add text annotations by typing a label and positioning it on the chart.

This period represents the official beginning and ending of the US Great Recession in the eyes of economists, although unemployment continued to grow for the population at large. When you finish “drawing” on the chart, the red bar will become light gray. Once again, you can fine-tune the positioning of a highlighted range by typing its year-month codes into the date fields, and change its appearance or delete it by clicking the downward arrow symbol for more options. To highlight other official recession periods, draw two more ranges: March–November 2001 and February–August 2020 (the most current data as we write this).

- Click *Proceed* or advance to the *Publish & Embed* screen to share your work with others. If you logged into your free Datawrapper account, your work is automatically saved online in the *My Charts* menu in the top-right corner of the screen. Also, you can click the blue *Publish* button to generate the code to embed your interactive chart on your website, as you’ll learn about in Chapter 10: Embed on the Web. In addition, you can *add your chart to River* if you wish to share your work more widely by allowing other Datawrapper users to adapt and reuse your chart. Furthermore, scroll all the way down and click the *Download PNG* button to export a static image of your chart. Additional exporting and publishing options require a paid Datawrapper account. Or, if you prefer not to create an account, you can enter your email to receive the embed code.

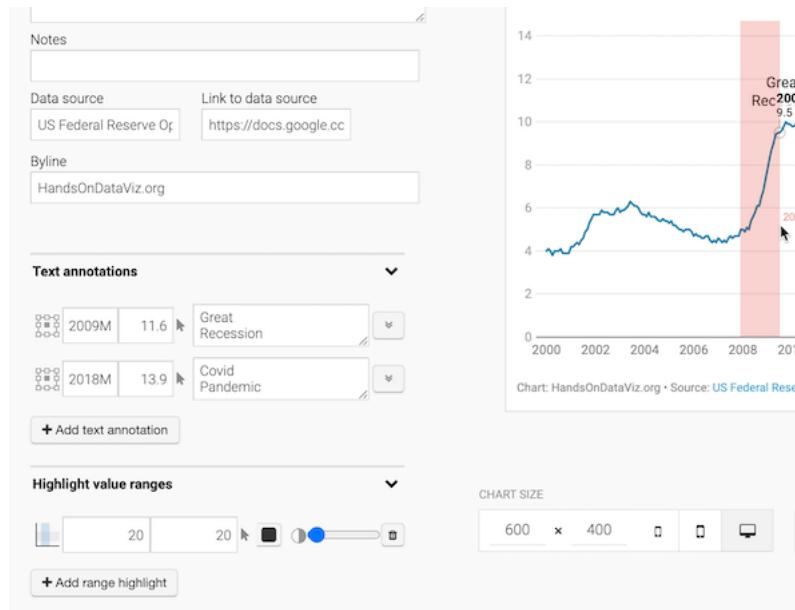


Figure 7.36: Add a range highlight by “drawing” a rectangular bar on the chart.

Congratulations on creating your first interactive Datawrapper chart. Now let’s use Datawrapper to create a new type of chart, called a range chart.

- Range Charts

A range chart (also known as a specific type of dot chart) is best to show gaps between data points, such as inequalities. The line chart above illustrated how the overall US unemployment rate changed over time. But if we wish to look at differences in gender and race/ethnicity for a specific point in time, a range chart will highlight any gaps. Organize the data for a range chart into rows and columns, as shown in Figure 7.37. The column headers contain data labels you wish to highlight, such *Men* and *Women*. The first column contains your categories, such as racial/ethnic groups, followed by the numerical values under each header. Now you can easily create an interactive range chart as shown in Figure 7.38. The employment gender gap is visible in all three racial/ethnic groups, but the gap was widest between Hispanic men and women in May 2020.

To create your own range chart with our sample data, follow this tutorial. Since we assume that you’re already familiar with Datawrapper from the previous tutorial on annotated line charts, these steps are abbreviated. So if you get lost, see more details and images above.

A	B	C
Unemployment May 2020	Men	Women
White	10.7	13.1
Hispanic	14.7	18.6
Black	15.5	16.5

Figure 7.37: Organize your range chart data labels into column headers, with categories and values in each row.

US Unemployment by Gender and Race/Ethnicity, May 2020

Seasonally adjusted

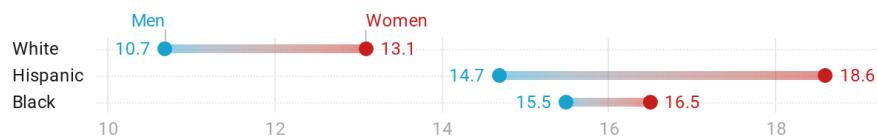


Chart: by HandsOnDataViz.org • Source: Federal Reserve Open Data • Created with Datawrapper

Figure 7.38: Range chart: Explore the interactive version. Data from US Federal Reserve Open Data.

1. Open the US Unemployment by Gender and Race/Ethnicity May 2020 sample data in Google Sheets and go to *File > Make a Copy* to create your own version in your Google Drive. Or download the sample data in Excel format to your computer.
2. Open Datawrapper in your browser and click *Start Creating*. We recommend that you create a free account to better manage your visualizations, but it's not required.
3. In the *Upload Data* screen, click *Import Google Spreadsheet* and paste the link to the data in the shared Google Sheet above, then click *Proceed*. Alternatively, you can upload data by copying and pasting it into the data table window, or uploading an Excel or CSV file.
4. In the *Check and Describe* screen, inspect your data, then click *Proceed*.
5. In the *Visualize* screen, Datawrapper will attempt to guess the chart type you desire, based on the data format, but you will need to select *Range Plot*.
6. Click the *Annotate* tab near the top-left of the *Visualize* screen to add a meaningful title, data source, and byline credits.
7. Click the *Refine* tab of the *Visualize* screen to modify the range chart appearance. You have several options, but here's the most important ones in this case. First, in the *Labels* section, change the visibility of the values from *start* to *both*, which places numbers at both ends of each range. Second, push the slider to *Label first range*, which places the word *Men* and *Women* above the first range, as shown in Figure 7.39.

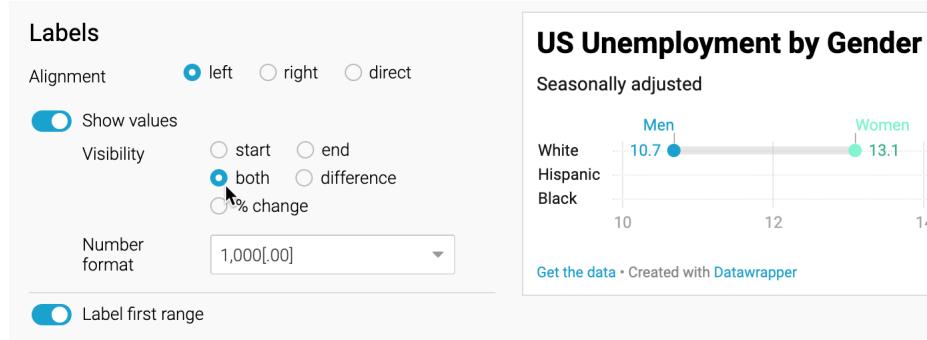


Figure 7.39: Modify the labels settings to show values at both ends of each range, and to place your data labels on your first range.

8. Still in the *Refine* tab, scroll down to the *Appearance* section to improve the colors. Select the *Range end* drop-down menu to select a better color,

such as red. Also, change the *Range color* setting to *gradient* to emphasize the range, as shown in Figure 7.40.

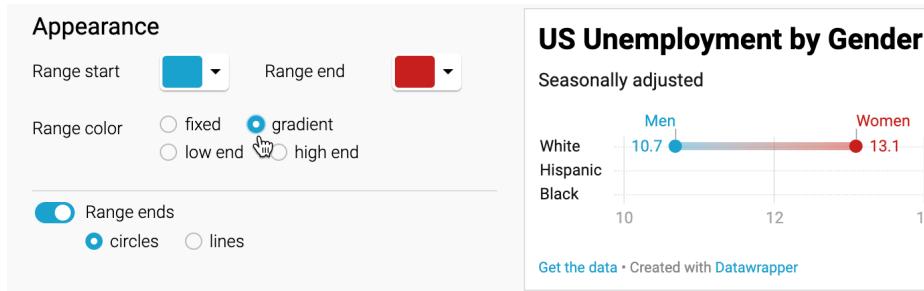


Figure 7.40: Modify the appearance settings to improve the color and add a gradient.

Tip: The *Refine* tab also includes options to resort or group data rows, change the chart size for different devices, and check visibility for colorblind readers.

- After modifying your visualization, proceed to the *Publish and Embed* screen, and follow the prompts to share your work, or refer to the more detailed tutorial above.

Now that you've completed a range chart, let's create another new chart type, scatter and bubble charts, which are relatively easy to design in Datawrapper.

- Scatter and Bubble Charts

Scatter charts (also known as scatter plots) are best to show the relationship between two datasets by displaying their XY coordinates as dots to reveal possible correlations. In the scatter chart example below, each dot represents a nation, with its life expectancy on the horizontal X axis and its fertility rate (births per woman) on the vertical Y axis. The overall dot pattern illustrates a correlation between these two datasets: life expectancy tends to increase as fertility decreases.

Bubble charts go further than scatter charts by adding two more visual elements—dot size and color—to represent a third or fourth dataset. The bubble chart example further below begins with the same life expectancy and fertility data for each nation that we previously saw in the scatter chart, but the size of each circular dot represents a third dataset (population) and its color indicates a fourth dataset (region of the world). As a result, bubble charts are scatter charts on steroids, because they pack even more information into the visualization.

Fancier bubble charts introduce one more visual element—animation—to represent a fifth dataset, such as change over time. Although creating an animated bubble chart is beyond the scope of this book, watch a famous TED talk by Hans Rosling, a renowned Swedish professor of global health, to see animated bubble charts in action, and learn more about his work at the Gapminder Foundation.

In this section, you'll learn why and how to create a scatter chart and a bubble chart in Datawrapper. Be sure to read about the pros and cons of designing charts with Datawrapper in the prior section.

Scatter Charts

A scatter chart is best to show the relationship between two sets of data as XY coordinates on a grid. Imagine you wish to compare life expectancy and fertility data for different nations. Organize your data in three columns, as shown in Figure 7.41. The first column contains the *Country* labels, and the second column, *Life Expectancy*, will appear on the horizontal x-axis, while the third column, *Fertility*, will appear on the vertical y-axis. Now you can easily create a scatter chart that displays a relationship between these datasets, as shown in Figure 7.42. One way to summarize the chart is that nations with lower fertility rates (or fewer births per woman) tend to have high life expectancy rates. But another way to phrase it is that nations with higher life expectancy at birth have lower fertility. Remember that correlation is not causation, so you cannot use this chart to argue that fewer births produce longer lives, or that longer-living females create fewer children.

	A	B	C
1	Country	Life Expectancy	Fertility
2	China	76.7	1.7
3	India	69.4	2.2
4	United States	78.5	1.7
5	Indonesia	71.5	2.3
6	Brazil	75.7	1.7
7	Pakistan	67.1	3.5

Figure 7.41: To create a scatter chart in Datawrapper, format data in three columns: labels, x-values, and y-values.

Fertility and Life Expectancy in selected nations, 2018

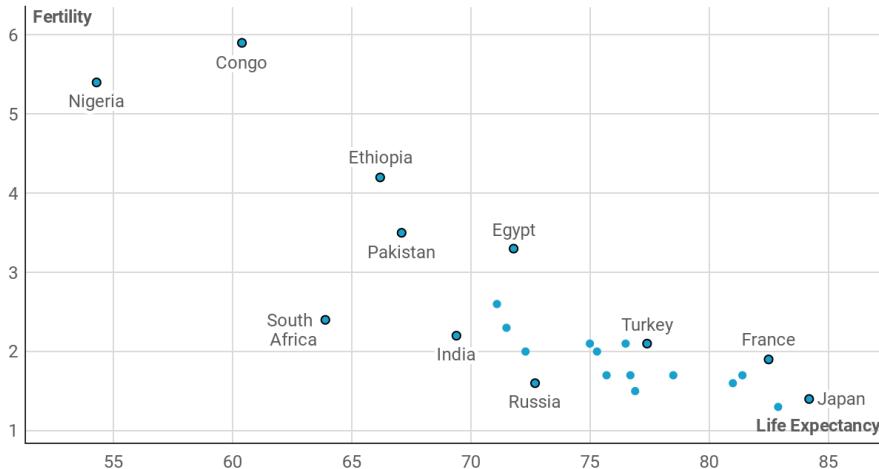


Chart: HandsOnDataViz.org • Source: World Bank • Created with Datawrapper

Figure 7.42: Scatter chart: Explore the interactive version. Data from the World Bank.

Create your own interactive scatter chart in Datawrapper, and edit the tooltips to properly display your data:

1. Open our Scatter Chart sample data in Google Sheets, or use your own data in a similar format.
2. Open Datawrapper and click to start a new chart.
3. In the Datawrapper *Upload Data* screen, either copy and paste the link to the data tab of the Google Sheet above, or copy and directly paste in the data. Click *Proceed*.
4. In the *Check and Describe* screen, inspect your data and make sure that the *Life Expectancy* and *Fertility* columns are blue, which indicates numeric data. Click *Proceed*.
5. In the *Visualize* screen, under the *Chart type* tab, select *Scatter Plot*. Float your cursor over the scatter chart that appears in the right-hand window, and you'll notice that we still need to edit the tooltips to properly display data for each point.
6. In the *Visualize* screen, under the *Annotate* tab, scroll down to *Customize tooltip*, and click *edit tooltip template*. In the *Customize tooltip HTML*

window, click inside the *Title* field and click on the blue column name *Country* to add it there. The *Title* field now appears as `{{ Country }}`, which means that the proper country name will appear in the tooltip when you hover over each point. In addition, click inside the *Body* field, type *Life expectancy:*, then click the blue column with the same name to add it, so that `{{ Life_Expectancy }}` appears after it. Press *return* twice on your keyboard, then type *Fertility:* and click on the blue column with the same name to add it, so that `{{ Fertility }}` appears right after it, as shown in Figure 7.43. Press *Save* to close the tooltip editor window.

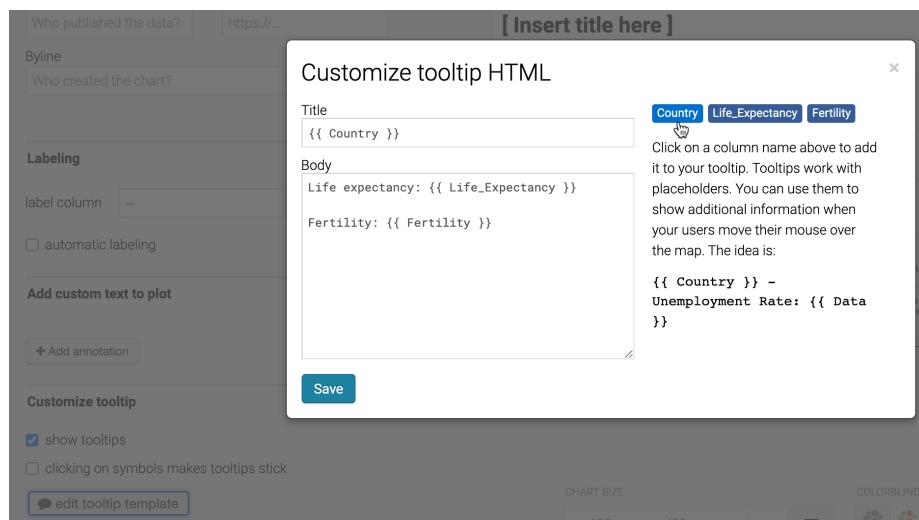


Figure 7.43: In the tooltip editor window, type and click column headers to customize the display.

7. Back in the *Visualize* screen, when you hover your cursor over a point, the tooltip will properly display its data according to your editor settings above, as shown in Figure Figure 7.44.
8. Finish the annotations to add your title and data source, then proceed to publish and embed your chart by following the prompts or reading the more detailed Datawrapper tutorial above. Learn about your next steps in Chapter 10: Embed on the Web.

Bubble Charts

In your scatter chart above, you learned how to visualize the relationship between two datasets: life expectancy (the X-axis coordinate) and fertility (the

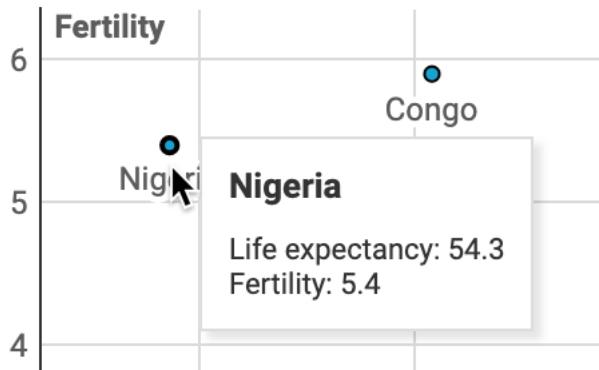


Figure 7.44: Hover over a data point to inspect the edited tooltip display.

Y-axis coordinate). Now let's expand on this concept by creating a bubble chart that adds two more datasets: population (shown by the size of each point, or bubble) and region of the world (shown by the color of each bubble). We'll use similar World Bank data as before, with two additional columns, as shown in Figure 7.45. Note that we're using numeric data (population) for bubble size, but categorical data (regions) for color. Now you can easily create a bubble chart that displays a relationship between these four datasets, as shown in Figure 7.46.

	A	B	C	D	E
1	Country	Life expectancy	Fertility	Population	Region
2	United States	78.5	1.70	326687501	North America
3	United Kingdom	81.4	1.70	66460344	Europe
4	China	76.7	1.70	1392730000	Asia
5	India	69.4	2.20	1352617328	Asia
6	Japan	84.2	1.40	126529100	Asia

Figure 7.45: To create a bubble chart in Datawrapper, organize the data into five columns: labels, x-axis, y-axis, bubble size, bubble color.

Create your own interactive bubble chart in Datawrapper, and edit the tooltips, bubble sizes, and colors to display your data:

1. Open our Scatter Chart sample data in Google Sheets, or use your own data in a similar format.

Fertility, Life Expectancy, and Population in nations, 2018

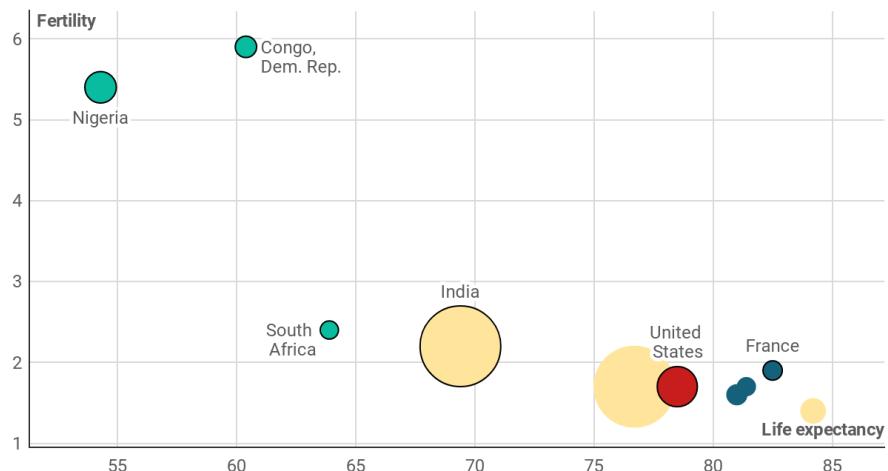


Chart: HandsOnDataViz.org • Source: World Bank • Created with Datawrapper

Figure 7.46: Bubble chart: Explore the interactive version. Data from the World Bank.

2. Open Datawrapper and click to start a new chart.
3. Follow steps 3–5 above to upload, check, and visualize the data as a *Scatter Plot* chart type.
4. In the *Visualize* screen, under the *Annotate* tab, scroll down to *Customize tooltip*, and click *edit tooltip template*. In the *Customize tooltip HTML* window, type in the fields and click on the blue column names to customize your tooltips to display country, life expectancy, fertility, and population, as shown in Figure 7.47. Press *Save* to close the tooltip editor window.
5. Back in the *Visualize* screen, under the *Refine* tab, scroll down to *Color*, select column for *Region*, and click the *customize colors* button to assign a unique color to each. Then scroll down to *Size*, check the box to change size to *variable*, select column for *Population*, and increase the max size slider, as shown in Figure 7.48. Click *Proceed*.
6. Test your visualization tooltips. Then finish the annotations to add your title and data source, and proceed to publish and embed your chart, by following the prompts or reading the more detailed Datawrapper tutorial above. See your next steps in Chapter 10: Embed on the Web.

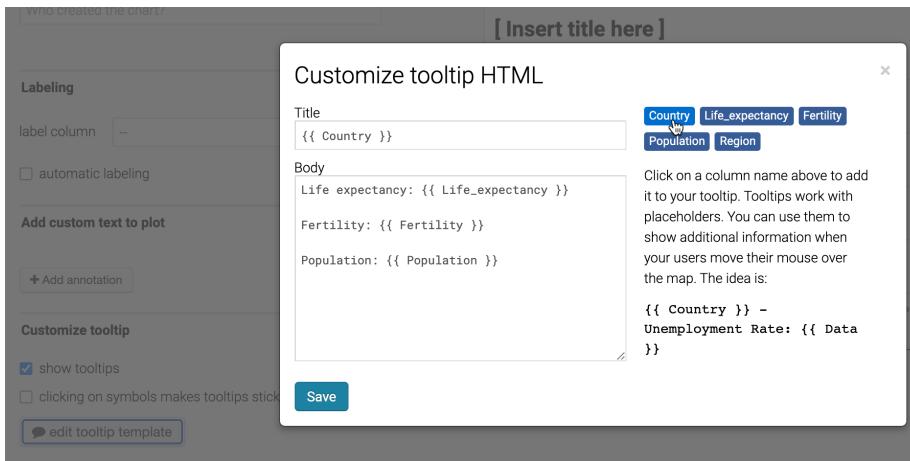


Figure 7.47: In the tooltip editor window, type and click column headers to customize the display.

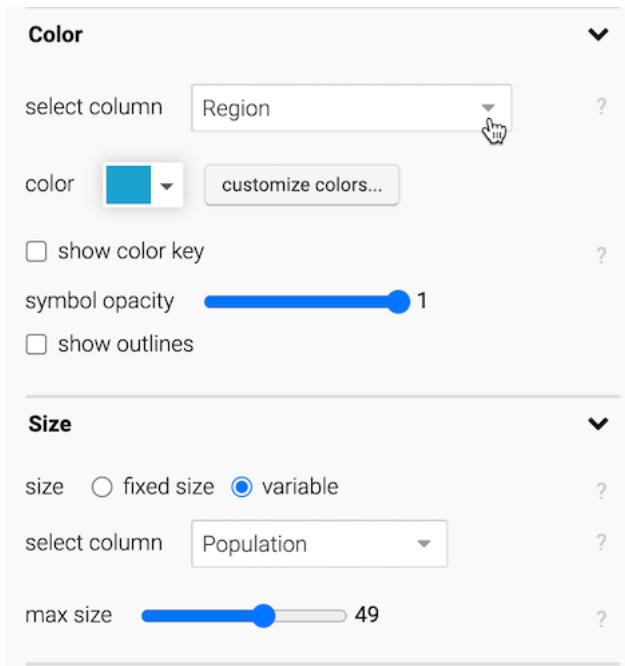


Figure 7.48: In the *Visualize* screen, modify the bubble colors and set size to variable.

For more information about creating scatter and bubble charts, see the Datawrapper Academy support site.

Now that you've learned how to create a scatter chart in Datawrapper, in the next section you'll learn how to create the same chart type with a different tool, Tableau Public, to build up your skills so that you can make more complex charts with this powerful tool.

Tableau Public Charts

Tableau is a powerful data visualization tool used by many professionals and organizations to analyze and present data. Our book focuses on the free version, Tableau Public, a desktop application for Mac or Windows computers, which you can download at no cost by providing an email address. The free Tableau Public tool is very similar to the pricier Tableau versions sold by the company, with one important constraint. Everyon data visualization you publish becomes public, as the product name suggests, so do not use Tableau Public for any sensitive or confidential data that you do not wish to share with others.

Tableau Public has several features that make it stand out from other drag-and-drop tools in this book. First, you can prepare, pivot, and join data inside Tableau Public, similar to some of the spreadsheet skills in Chapter 3, data cleaning methods in Chapter 5, and tools to transform map data coming up in Chapter 14. Second, Tableau Public offers a wider array of chart types than other free tools. Finally, with Tableau Public you can combine multiple visualizations (including tables, charts, and maps) into interactive dashboards or stories, which you can publish and embed on your website. Learn more about all of these features in the Tableau Public resources page.

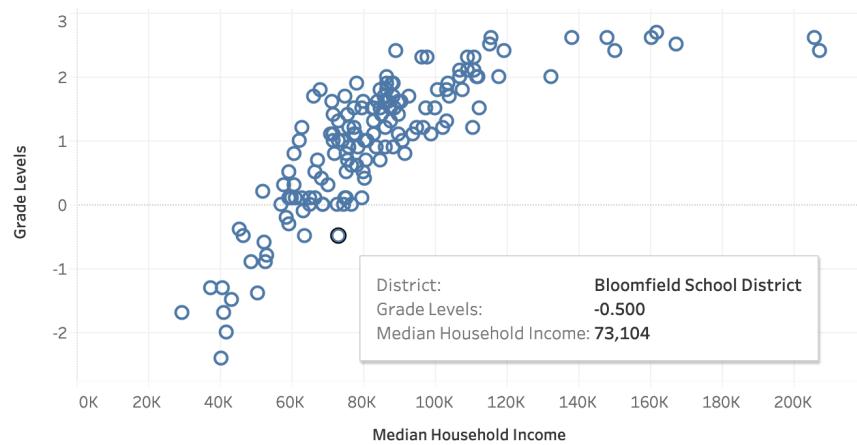
But Tableau Public also has some drawbacks. First, it may take several minutes to install and start up the application the first time. Second, if you feel overwhelmed by its design interface, you're not alone. Its drag-and-drop layout to build charts and maps initially can be confusing at first glance, and its internal vocabulary of data terms may seem unfamiliar. While Tableau Public is a powerful tool, perhaps it offers too many options.

In the next section we'll keep things simple by starting with the basics of Tableau Public, with step-by-step tutorials to create two different types of charts. First, you'll build on skills you already learned in the section above by building a scatter chart in Tableau Public. Second, you'll learn how to create a filtered line chart, which demonstrates more of the tool's strengths in interactive visualization design.

- Scatter Chart

Scatter charts are best to show the relationship between two datasets, placed on the x- and y-axis, to reveal possible correlations. With Tableau Public, you can create an interactive scatter chart, where you can hover your cursor over points to view more details about the data. Organize your data in three columns, the same way as the Datawrapper scatter chart tutorial: the first column for data labels, the second column for the x-axis, and the third column for the y-axis. Then you can create an interactive scatter chart as shown in Figure 7.49, which illustrates a strong relationship between household income and test scores (above or below the national average for 6th grade math and English) in Connecticut public school districts. To learn more about the data and related visualizations, see Sean Reardon et al. at the Stanford Education Data Archive, Motoko Rich et al. at The New York Times, Andrew Ba Tran at CT Mirror/TrendCT, and this TrendCT GitHub repo.

CT School Districts by Income and Grade Level Equivalents, 2009-13



Connecticut School Districts by Median Household Income (ACS 2009-13) and 6th Grade Math and English Test Scores as Grade Level Equivalents, above/below national average (SEDA/TrendCT 2009-13). Sources: Stanford Education Data Archive (<https://cepa.stanford.edu/seda>), CT Mirror/TrendCT (<https://github.com/trendct-data/stanford-cepa>), American Community Survey.

← → ↺ ⌂ ⌄

Figure 7.49: Scatter chart in Tableau Public: Explore the interactive version. Data by CT Mirror/TrendCT and Stanford CEPA.

To create your own scatter chart using this sample data in Tableau Public, follow this tutorial.

Install Tableau Public and Connect Data

1. Download the CT Districts-Income-Grades sample data in Excel format, or view and download the Google Sheets version. The data file consists of three columns: district, median household income, and test score levels.
2. Install and start up the free Tableau Public desktop application for Mac or Windows. It may require several minutes to complete this process. Tableau Public's welcome page includes three sections: Connect, Open, and Discover.
3. Under *Connect*, you can choose to upload a Microsoft Excel file, or choose *Text file* to upload a CSV file, or other options. Or to connect to a server, such as Google Sheets, click *More...* to connect to your account. After you successfully connect to your data source, you will see it under *Connections* in the *Data Source* tab. Under *Sheets*, you will see two tables, **data** and **notes**.
4. Drag the **data** sheet into *Drag tables here* area, as shown in Figure 7.50. You will see the preview of the table under the drag-and-drop area. You have successfully connected one data source to Tableau Public, and you are ready to build your first chart.

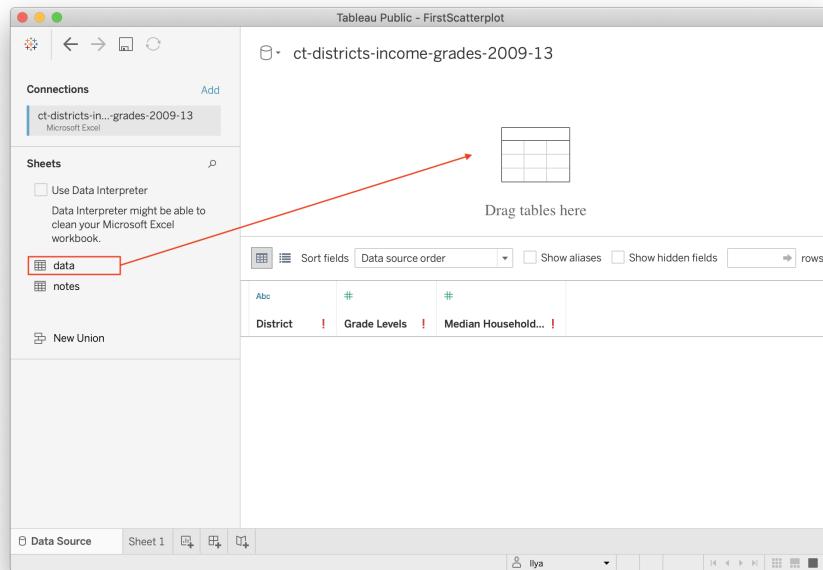


Figure 7.50: Drag **data** sheet into *Drag tables here* area.

Create Scatter Chart in the Worksheet

1. In the *Data source* screen, click on the orange *Sheet 1* tab (in the lower-left corner) to go to your worksheet, where you will build the chart.

Although it may feel overwhelming at first, the key is learning where to drag items from the Data pane (left) into the main worksheet. Tableau marks all data fields in blue (for discrete values, mostly text fields or numeric labels) or green (for continuous values, mostly numbers).

2. In your worksheet, drag the *Grade Levels* field into the *Rows* field above the charting area, which for now is just empty space. See Figure 7.51 for this dragging step and the following two steps. Tableau will apply a summation function to it, and you will see `SUM(Grade Levels)` appear in the *Rows* row, and a blue bar in the charting area. It makes little sense so far, so let's plot another data field.
3. Drag *Median Household Income* to the *Columns* field, just above the *Rows* field. In general, choosing between Rows and Columns shelves can be challenging, but it is convenient to think of *Columns* shelf as representing your x-axis, and *Rows* as y-axis. Once again, Tableau will apply the summation function, so you will see `SUM(Median Household Income)` in the *Columns* shelf. The bar chart will automatically transform into a scatter chart with just one data point in the upper-right corner, because the data for both is aggregated (remember the SUM function).
4. We want to tell Tableau to disaggregate the household and grade levels variables. In other words, we want to introduce an extra level of granularity, or *detail* to our visualization. To do so, drag the *District* dimension into the *Detail* shelf of the *Marks* card. Now a real scatter chart will appear in the charting area. If you hover over points, you will see all three values associated with these points.

Add Title and Caption, and Publish

Give your scatter chart a meaningful title by double-clicking on the default *Sheet 1* title above the charting area. Add more information about the chart, such as source of the data, who built the visualization and when, and other details to add credibility to your work. You can do so inside a *Caption*, a text block that accompanies your Tableau chart. In the menu, go to *Worksheet > Show Caption*. Double-click the Caption block that appears, and edit the text. As a result, your final worksheet will look like shown in Figure 7.52.

Tip: In the dropdown above the *Columns* shelf, change *Standard* to *Fit Width* to ensure your chart occupies 100 percent of available horizontal space.

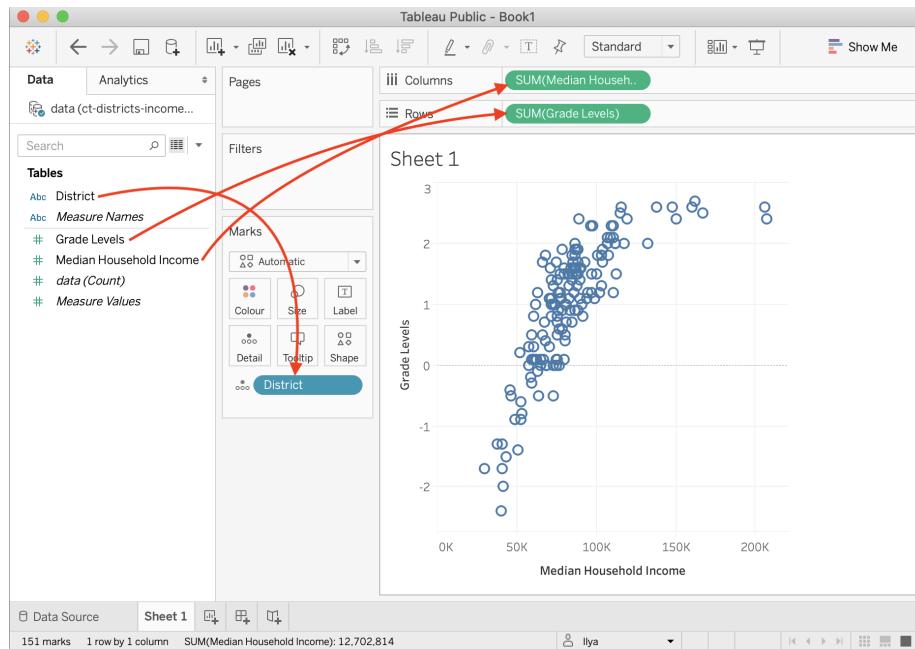


Figure 7.51: Drag data fields to the right locations in Tableau Public.

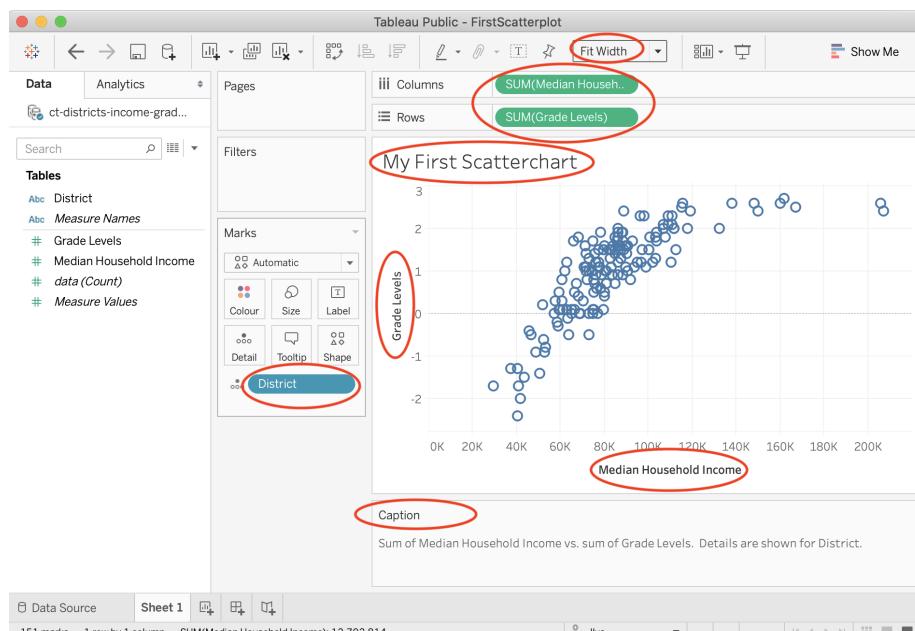


Figure 7.52: This scatter chart is ready to be published.

1. To publish your interactive chart on the public web, go to *File > Save to Tableau Public As...*. A window to sign in to your account will pop up. If you don't have an account, click *Create one now for free* at the bottom, and save the login details in your password manager.
2. After signing in, a window to set the workbook title will appear. Change the default *Book1* title to something meaningful, as this name will appear in the public web address for your published work. Click *Save*.
3. After saving your workbook on the public web, Tableau Public will open up a window in your default browser with the visualization. In the green banner above the chart, click *Edit Details* to edit the title or description. Under *Toolbar Settings*, see the checkbox to *Allow others to download or explore and copy this workbook and its data*, and select the setting you wish, as shown in Figure @ref(fig:tableau-toolbar-settings). If you are publishing your visualization on the web, we also recommend that you keep this box checked so that others can download your data and see how you constructed it, to improve data accessibility for all.



Figure 7.53: This scatter chart is ready to be published.

Tip: Your entire portfolio of Tableau Public visualizations is online at <https://public.tableau.com/profile/USERNAME>, where **USERNAME** is your unique username.

See the [Embed Tableau Public on Your Website](#) section of this book to insert the interactive version of your chart on a web page that you control.

- Filtered Line Chart

Now that you've learned how to create a scatter chart in Tableau Public, let's move on to a new type of chart that highlights the tool's strengths. Instead of *static* charts, such as those found in print or PDFs, this book features *interactive* charts for their ability to display more data. But you can also design interactive charts to show only the amount of data you desire. In other words, your interactive visualization can become a data-exploration tool that allows users to "dig" and find specific data points and patterns, without overwhelming them with too much information at once.

In this tutorial, we will build an interactive filtered line chart with Tableau Public, to visualize how internet access has changed in different nations over time. Organize the data in three columns, as shown in Figure 7.54. The first column, *Country Name*, are the data labels that become the colored lines. The second column, *Year*, will appear on the horizontal x-axis. The third column, *Percent Internet Users*, are numeric values that appear on the vertical y-axis. Now you can create a filtered line chart with checkboxes, to show only selected lines on startup to avoid overwhelming users, while allowing them to toggle on other lines, and hover over each one for more details, as shown in Figure 7.55.

	A	B	C
1	CountryName	Year	PercentInternetUsers
839	Cameroon	2016	23.20297197
840	Cameroon	2017	23.20297197
841	Cameroon	2018	
842	Canada	1995	4.163525253
843	Canada	1996	6.76023965
844	Canada	1997	15.07235736
845	Canada	1998	24.8974003

Figure 7.54: In a filtered line chart, organize the data in three columns, data labels, year, and numeric values.

To create your own filtered line chart using this sample data in Tableau Public, follow this tutorial. We assume that you have already installed the free Tableau Public desktop application for Mac or Windows, and have already become familiar with the tool by completing the previous Scatter Chart with Tableau Public tutorial, since the steps below are abbreviated.

Connect Data to Tableau Public

1. Download the World Bank Internet Users 1995-2018 sample data in Excel format, or view and download the Google Sheets version. The file consists of three columns: data labels, year, and numeric values.
2. Open Tableau Public, and under the *Connect* menu, you can upload your data as a Microsoft Excel file, or choose *Text file* to upload a CSV file, or click *More...* to connect to a server and upload a Google Sheet from your

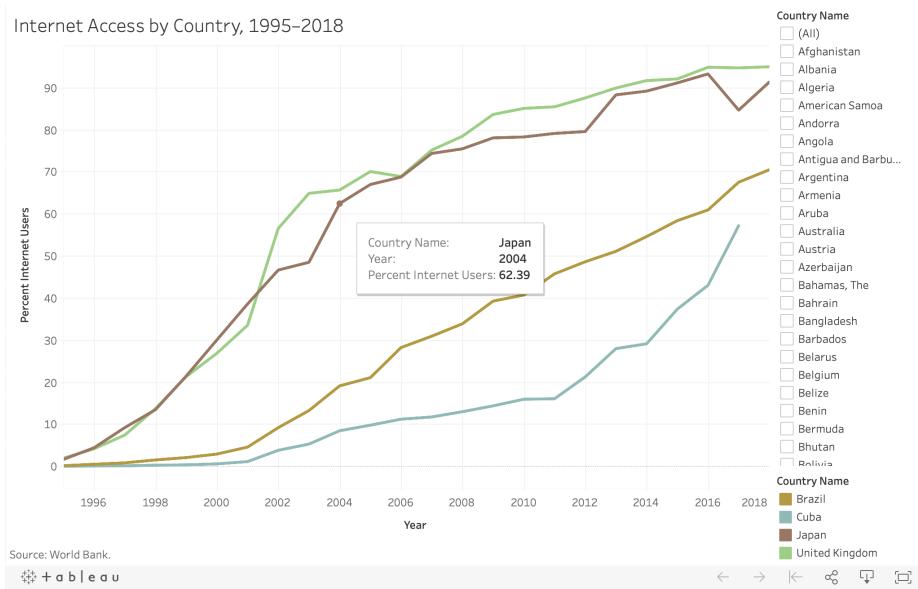


Figure 7.55: Filtered Line chart: Explore the interactive version. Data from World Bank.

account. After you successfully connect to your data source, you will see it under *Connections* in the *Data Source* tab. Under *Sheets*, you will see two tables, **data** and **notes**. Drag the **data** sheet into *Drag tables here* area to preview it.

3. In the *Data source* screen, click on the orange *Sheet 1* tab (in the lower-left corner) to go to your worksheet, where you will build the chart.

In your worksheet, your variables will be listed under *Tables* in the left-hand side. The original variables are displayed in normal font, the *generated* variables will be shown in *italics* (such as *Latitude* and *Longitude*, which Tableau guessed from the country names). Now you are ready to begin building your interactive chart.

Build and Publish a Filtered Line Chart

1. Drag the *Year* variable to *Columns* shelf. This will place years along the x-axis.
2. Drag the *Percent Internet Users* variable to *Rows* shelf to place them on the y-axis. The value in the shelf will change to **SUM(Percent Internet Users)**. You should see a single line chart that sums up percentages for each year. That is completely incorrect, so let's fix it.

3. In order to “break” aggregation, drag-and-drop *Country Name* to the *Color* shelf of the *Marks* card. Tableau will warn you that the recommended number of colors should not exceed 20. Since we will be adding checkbox filtering, ignore this warning, and go ahead and press the *Add all members* button.
4. At first, everything will look like a spaghetti plate of lines and colors! To add filtering, drag *Country Name* to the *Filters* card. In the Filter window, make sure all countries are checked, and click *OK*.
5. In the *Filters* card, click the dropdown arrow of the **Country Name** symbol, then scroll down and select *Show Filter*, as shown in Figure 7.56.
6. You will see a list of options with all checkboxes to appear on the right side of the chart. Click *(All)* to add/remove all options, and select a few countries to see how the interactive filtering works. The checkboxes you select at this stage will appear “on” in the published map. You may notice that some countries from your “on” selection got assigned the same value. The good news is, Tableau lets you change colors of individual datapoints (in our case, countries). From the *Marks* card, click *Color* shelf, and then *Edit Colors*.... Double-click a country from the *Select Data Item*: list to bring up a color picker window, pick your favorite color, and click *OK*. Although you can ensure that your pre-selected countries are painted in unique colors, there will be repetitions among other countries as your palette is limited to 20 colors. Unfortunately, there is little you can do to go around this.

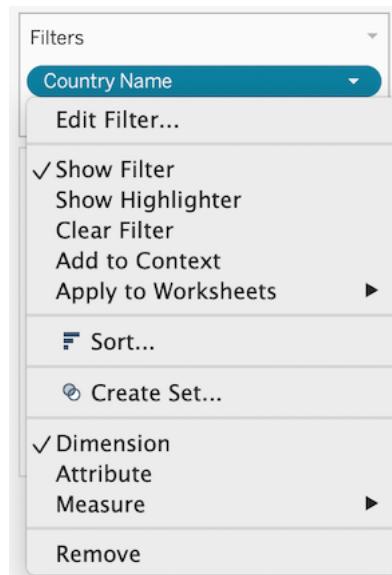


Figure 7.56: After you drag Country Name to the Filters card, make sure the Filter is displayed.

7. Double-click on the *Sheet 1* title (above the chart) and replace it with a more meaningful title, such as “Internet Access by Country, 1995–2018.” In the menu, go to *Worksheet > Show Caption* to add a Caption block under the chart. Use this space to add source of your data (World Bank), and perhaps credit yourself as the author of this visualization.
8. Change *Standard* to *Fit Width* in the drop-down menu above the *Columns* shelf.
9. You may notice that the x-axis (Year) starts with 1994 and ends with 2020, although our data is for 1995–2018. Double-click on the x-axis, and change *Range* from *Automatic* to *Fixed*, with the Fixed start of 1995, and the Fixed end of 2018. Close the window and see that the empty space on the edges has disappeared.
10. Once your filtered line chart looks like the one shown in Figure 7.57, you are ready to publish. Go to *File > Save to Tableau Public As...*, and log into your account, or create one if you haven’t yet done so. Follow the prompts to publish your chart on the public web, or see the previous Scatter Chart in Tableau Public tutorial for more details.

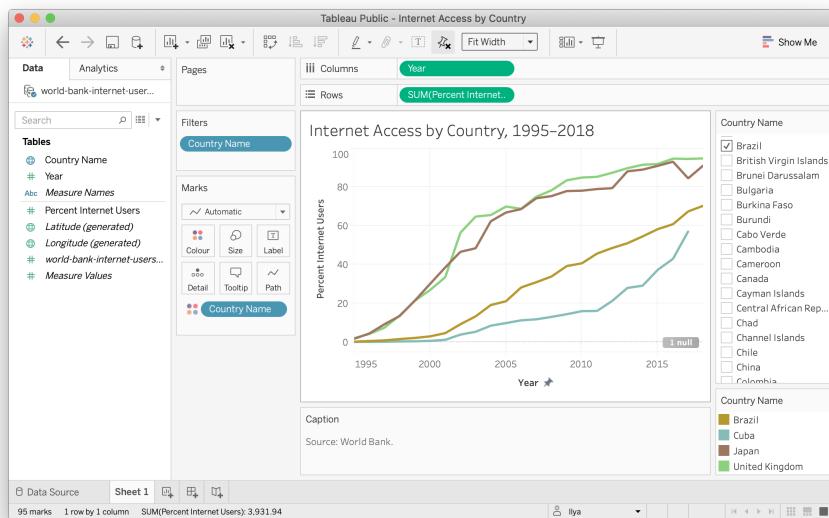


Figure 7.57: This workbook is ready to be published.

See the [Embed Tableau Public on Your Website](#) section of this book to insert the interactive version of your chart on a web page that you control.

Summary

Congratulations on creating interactive charts that pull readers deeper into your story, and encourage them to explore the underlying data! As you continue to create more, always match the chart type to your data format and the story you wish to emphasize. Also, design your charts based on the principles and aesthetic guidelines outlined near the top of this chapter. While anyone can click a few buttons to quickly create a chart nowadays, your audiences will greatly appreciate well-designed charts that thoughtfully call their attention to meaningful patterns in the data.

In this chapter you learned how to create different types of interactive charts with Google Sheets, Datawrapper, and Tableau Public. For more advanced chart design with open-source code, see Chapter 12: Chart.js and Highcharts templates, which give you ever more control over how your design and display your data, but also requires learning how to edit and host code templates with GitHub in Chapter 11.

The next chapter on Map Your Data follows a similar format to introduce different map types, design principles, and hands-on tutorials to create interactive visualizations with spatial data. Later you'll learn how to embed interactive charts on your web in Chapter 10.

Chapter 8

Map Your Data

Maps draw your readers into data that includes a spatial dimension, while also developing a stronger sense of place. Seeing the relative distance between points on a map, or identifying geographic patterns in a *choropleth* map (where colored polygons represent data values), relays information to readers' eyes more effectively than text, tables, or charts. But creating meaningful maps that draw our attention to key insights in your data requires clear thinking about design choices.

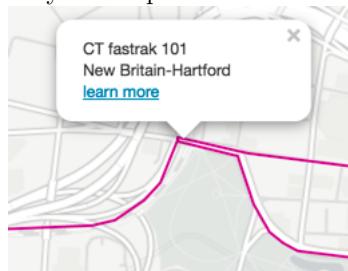
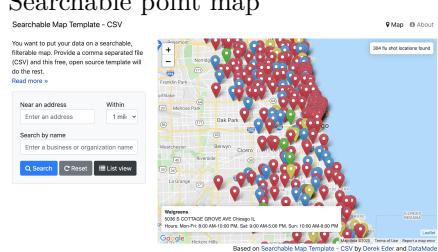
In this chapter, we will examine principles of map design and distinguish between good and bad maps. You will learn about rules that apply to all maps, and specific guidelines for creating choropleth maps. While many tools allow you to download maps as *static* images, our book also demonstrates how to construct *interactive* charts that invite readers to zoom in and explore the data in their web browsers. Later you'll learn how to embed interactive charts on your website in Chapter 10.

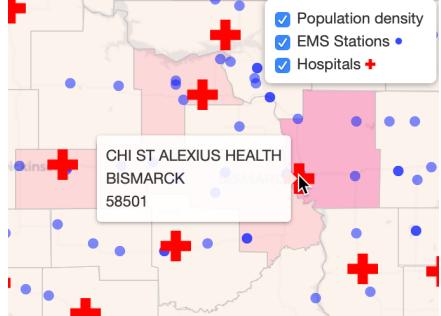
Learn about different types of maps you can create in this book in Table 8.1. Decisions about map types are based on two main factors: the format of your data, and the kind of story you wish to tell. For example, point maps work best to show specific locations with colored markers to represent categories (such as hospitals), while choropleth maps are best suited to display relative values for regions (such as birth rates across US states). After selecting your map type, follow our tool recommendations and step-by-step tutorials. This chapter features *Easy Tools* with drag-and-drop menus, such as Google My Maps, Datawrapper, and Tableau Public. But the table also points you to *Power Tools* that give you more control to customize and host your visualizations, such as Leaflet code templates in Chapter 13. These advanced tools require prior knowledge on how to edit and host code templates with GitHub in Chapter 11.

TODO: Improve and standardize map images

Table 8.1: Basic Map Types, Best Uses, and Tutorials

Map	Best use and tutorials in this book
Point map with custom icons	Best to show specific locations (such as addresses or geocoordinates) with colored markers for categories, or custom icons, plus text and images in popup windows. Easy tool: BatchGeo tutorial or Google My Maps tutorial. Power tool: Ch 13: Leaflet Maps with Google Sheets tutorial
Symbol point map	Best to show specific locations (such as cities), with variable-sized shapes or colors to represent data values (such as population growth). Easy tool: Symbol Point Map with Datawrapper tutorial
Choropleth (colored polygon) map	Best to show patterns across geographic areas (such as neighborhoods or nations) by coloring polygons to represent data values. Easy tool: Choropleth map with Datawrapper tutorial or Choropleth map with Tableau Public tutorial Power tools: Ch 13: Leaflet Maps with Google Sheets tutorial
Heat point map	Best to show clusters of points as colored hotspots to emphasize high frequency or density of cases. Power tool: Ch 13: Leaflet Heatmap code template

Map	Best use and tutorials in this book
Story map 	Best to show a point-by-point guided tour, with a scrolling narrative to display text, images, audio, video, and scanned map backgrounds. Power tool: Ch 13: Leaflet Storymaps with Google Sheets tutorial
Polyline map 	Best to show routes (such as trails or transit), with colors for different categories. Easy Tool: Google My Maps tutorial Power tool: Ch 13: Leaflet Maps with Google Sheets tutorial
Customized point-polyline-polygon 	Best to show any combination of points, polylines, or polygons, with customized icons for categories, and colored regions to represent data values. Power tool: Ch 13: Leaflet Maps with Google Sheets tutorial
Searchable point map 	Best to show specific locations for users to search by name or proximity, or filter by category, with optional list view. Power Tool: Ch 13: Leaflet Searchable Point Map code template

Map	Best use and tutorials in this book
Current map from open-data repository 	Best to show the most current information pulled directly from an open-data repository such as Socrata and others. Easy tool: Current map with Socrata open data tutorialPower tool: Ch 13: Leaflet Maps with Open Data API code template

Map Design Principles

Much of the data collected today includes a spatial component that can be mapped. Whether you look up a city address or take a photo of a tree in the forest, both can be geocoded as points on a map. We also can draw lines and shapes to illustrate geographical boundaries of neighborhoods or nations, and color them to represent different values, such as population and income.

However, just because data *can* be mapped does not always mean it *should* be mapped. Before creating a map, stop and ask yourself: *Does location really matter to your story?* Even when your data includes geographic information, sometimes a chart tells your story better than a map. For example, you can clearly show differences between geographic areas in a bar chart, or trace how they rise and fall on different rates over time with a line chart, or compare two variables for each area in a scatter chart. Sometimes a simple table, or even text alone, communicates your point more effectively to your audience. Since creating a well-designed map requires time and energy, make sure it actually enhances your data story.

As you learned in the previous chapter about charts, data visualization is not a science, but comes with a set of principles and best practices that serve as a foundation for creating true and meaningful maps. In this section, we'll identify a few rules about map design, but you may be surprised to learn that some rules are less rigid than others, and can be "broken" when necessary to emphasize a point, as long as you are honestly interpreting the data. To begin to understand the difference, let's start by establishing a common vocabulary about maps.

Deconstruct a Map

Our book features how to create *interactive* maps, also called *tiled web maps* or *slippy maps* because you can zoom into and pan around to explore map data layers on top of a seamless set of basemap tiles. They usually include *zoom controls* (+ and - buttons) to view data at various levels that change the display of the basemap tiles. Basemaps that display pictorial images of streets and buildings are known as *vector* tiles, while those that display satellite imagery are known as *raster* tiles, and we'll explain the difference below. Take a look at Figure 8.1 to learn about basic elements in the maps you'll create in this chapter.

The top layer of interactive maps generally consists of a combination of *points*, *polylines*, and *polygons*. Points show specific places, such as the street address of a home or business, sometimes with a location marker. Each point represents a pair of latitude and longitude coordinates. For example, 40.69, -74.04 marks the location of the Statue of Liberty in New York City. Polylines are connected strings of points, such as roads or transportation networks, and we place the “poly-” prefix before “lines” to remind us that they may contain multiple branches. Polygons represent closed areas defined by lines, such as building footprints, census tracts, or state or national boundaries. Since all points, polylines, and polygons consist of latitude and longitude coordinates, they are called vector data. You can zoom up very close to vector data or vector basemaps without diminishing their visual quality. By contrast, our ability to zoom into raster data or rasterbasemaps is limited by the resolution of the original image, which gets fuzzier as we get closer. You'll learn more about these terms in the GeoJSON and Geospatial Data section of Chapter 14.

On interactive maps, any of these top-layer data elements may display a hidden *tooltip* (when you hover the cursor over them) or a *popup* (when you click on them) that reveals additional information about its properties. Like a traditional static map, the *legend* identifies the meaning of symbols, shapes, and colors. Maps also may include a *north arrow* or *scale* to orient readers to direction and relative distance. Similar to a chart, good maps should include a title and brief description to provide context about what it shows, along with its data sources, clarifying notes, and credit to the individuals or organizations that helped to create them.

TODO: ILYA please review the text above, which I revised again to address reviewers' comments. Should we create a pair of vector and raster map images, side-by-side, with “regular” vs “zoomed in” views to show differences, similar to https://en.wikipedia.org/wiki/Vector_graphics?

Clarify Point versus Polygon Data

Before you start to create a map, make sure you understand your data format and what it represents. Avoid mistakes commonly made by novices by pausing

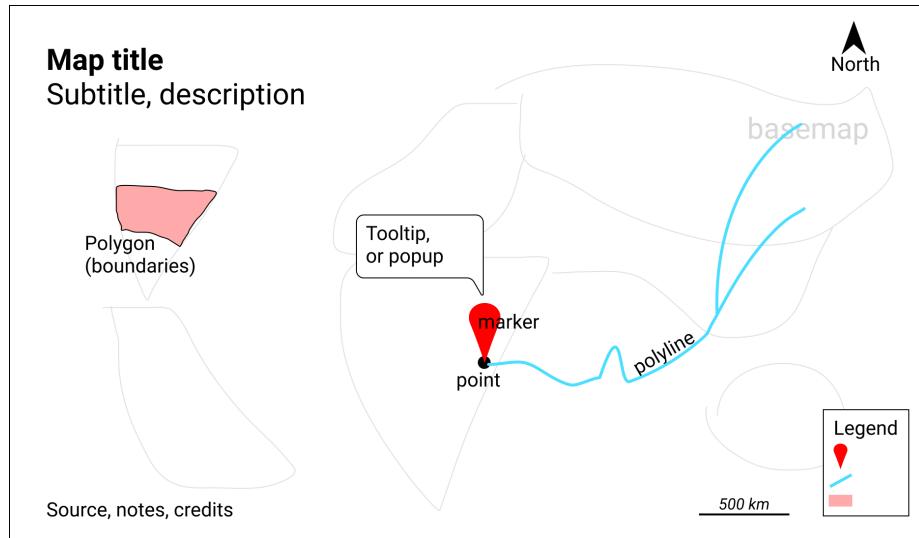


Figure 8.1: Key elements of an interactive map.

to ask these questions. First, *Can your data be mapped?* Sometimes the information we collect has no geographic component, or no consistent one, which makes it difficult or impossible to place on a map. If the answer is yes, then proceed to the second question: *Can the data be mapped as points or polygons?* These are the two most likely cases (which are sometimes confused), in contrast to the less-common third option, *polylines*, which represent paths and routes.

What type of data do you see listed below: points or polygons?

1. 36.48, -118.56 (latitude and longitude for Joshua Tree National Park, CA)
2. 2800 E Observatory Rd, Los Angeles, CA
3. Haight and Ashbury Street, San Francisco, CA
4. Balboa Park, San Diego, CA
5. Census tract 4087, Alameda County, CA
6. City of Los Angeles, CA
7. San Diego County, CA
8. State of California

In most cases, numbers 1-4 represent *point* data because they usually refer to a *specific locations* that can be displayed as point markers on a map. By contrast, numbers 5-8 generally represent *polygon* data because they usually refer to *geographic boundaries* that can be displayed as closed shapes on a map.

This point-versus-polygon distinction applies *most* of the time, but not always, with exceptions depending on your data story. First, it is possible, but not common, to represent all items 1-8 as *point* data on a map. For example, to tell

a data story about population growth for California cities, it would make sense to create a symbol point map with different-sized circles to represent data for each city. To do this, your map tool would need to find the center-point of the City of Los Angeles polygon boundary in order to place its population circle on a specific point on the map. A second way the point-versus-polygon distinction gets blurry is because some places we normally consider to be specific points *also* have polygon-shaped borders. For example, if you enter “Balboa Park, San Diego CA” into Google Maps, it will display the result as a map marker, which suggests it is point data. But Balboa Park also has a geographic boundary that covers 1.8 square miles (4.8 square kilometers). If you told a data story about how much land in San Diego was devoted to public space, it would make sense to create a choropleth map that displays Balboa Park as a polygon rather than a point. Third, it’s also possible to transform points into polygon data with pivot tables, a topic we introduced in Chapter 3. For example, to tell a data story about the number of hospital beds in each California county, you could obtain point-level data about beds in each hospital, then pivot them to sum up the total number of beds in each county, and display these polygon-level results in a choropleth map. See a more detailed example in the Pivot Points into Polygon Data section of Chapter 14: Transform Your Map Data

In summary, clarify if your spatial data should represent points or polygons, since those two categories are sometimes confused. If you envision them as points, then create a point-style map; or if polygons, then create a choropleth map. Those are the most common methods used by mapmakers, but there are plenty of exceptions, depending on your data story.

Map One Variable, Not Two

Newcomers to data visualization sometimes are so proud of placing one variable on a map that they figure two variables must be twice as good. But this usually is not true. In general, we recommend against placing two variables on the same map, because it’s very difficult for most readers to recognize patterns that help them to grasp your data story. For example, look at what happens when you try to place one variable (using a symbol point map) on top of another variable (using a choropleth map), as shown with imaginary data in Figure 8.2. It’s difficult to recognize patterns when a symbol point map is placed on a choropleth map.

Consider alternative ways to visualize the relationship between two variables. Rather than a map, you could create a scatter chart to show any relationship between variables X and Y. This type of chart works best if the relationship between variables is more important to your data story than any geographic patterns. Or, if geographic patterns matter for one of the variables, you could pair a choropleth map of that variable next to a scatterchart of both variables. Finally, you could create two choropleth maps, one for each variable, and place them side-by-side with text to explain their similarities or differences.

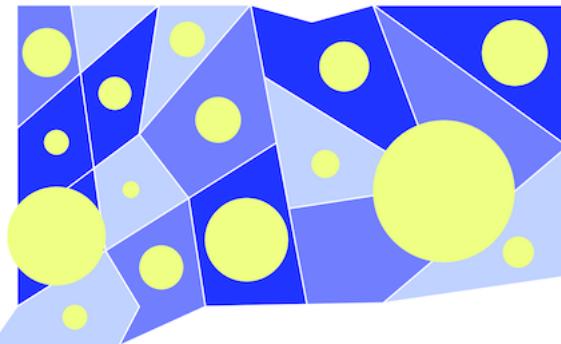


Figure 8.2: Avoid placing a symbol point map on top of a choropleth map, because it's hard to see the relationship between two variables.

TODO: ILYA let's discuss if the figure works above; also decide if supplemental images are needed, perhaps as "bad viz" and "better viz" examples

Choose Smaller Geographies for Choropleth Maps

Choropleth maps are best for showing geographic patterns across regions by coloring polygons to represent data values. Therefore, we generally recommend selecting *smaller* geographies to display more *granular* patterns, since larger geographies display aggregated data that may hide what's happening at lower levels. Geographers refer to this concept as the modifiable aerial unit problem, which means that the way you slice up your data affects how we analyze its appearance on the map. Stacking together lots of small slices reveals more detail than one big slice.

For example, compare the two choropleth maps of typical home values in the Northeastern United States in according to Zillow research data for September 2020. Zillow defines typical values as a smoothed, seasonally adjusted measure of all single-family residences, condos, and coops in the 35th to 65th percentile range, similar to the median value at the 50th percentile, with some additional lower- and higher-value homes. Both choropleth maps use the same scale. The key difference is the size of the geographic units: the first map shows home values at the larger state level, while the second map shows home values at the smaller county level, as shown in Figure 8.3.

Which map is best? Since both are truthful depictions of the data, the answer depends on the story you wish to tell. If you want to emphasize state-to-state differences, choose the first map because it clearly highlights how typical Massachusetts home prices are higher than those in surrounding Northeastern states. Or if you want to emphasize variation inside states, choose the second map, which demonstrates higher price levels in the New York City and Boston

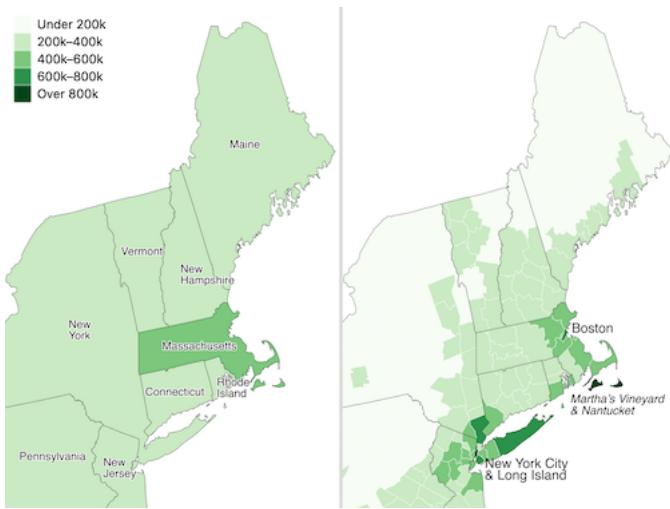


Figure 8.3: Zillow typical home values in September 2020 shown at the larger state level (left) versus the smaller county level (right).

metropolitan regions, in comparison to more rural counties in those two states. If you’re unsure, it’s usually better to map smaller geographies, because it’s possible to see both state-level and within-state variations at the same time, if the design includes appropriate labels and geographic outlines. But don’t turn *smaller is better* into a rigid rule, since it doesn’t work as you move further down the scale. For example, if we created a third map to display every individual home sale in the Northeastern US, it would be *too detailed* to see meaningful patterns. Look for just the right level of geography to clearly tell your data story.

Design Choropleth Colors & Intervals

When you build a choropleth map, your choices about how to represent data with colors will determine its overall appearance, so it’s important to learn key principles. Good choropleth maps make true and insightful geographic patterns clearly visible to readers, whether they are printed in black-and-white on paper or displayed in color on a computer screen. Furthermore, the best choropleth maps are designed to be interpreted correctly by people with colorblindness. For an excellent overview of visualization colors in general, see Lisa Charlotte Rost’s “A Friendly Guide to Colors in Data Visualization” and “How to Pick More Beautiful Colors for Your Data Visualizations,” both on the Datawrapper blog.¹

¹Rost, “Your Friendly Guide to Colors in Data Visualization.”; Rost, “How to Pick More Beautiful Colors for Your Data Visualizations.”

To illustrate key concepts about colors in choropleth map design, let's explore a wonderful tool called ColorBrewer, created by Cynthia Brewer and Mark Harrower.² See the interface in Figure 8.4. Since ColorBrewer is a design assistant, do not expect to upload your data into it to create a map. Instead, ColorBrewer will recommend color palettes that work best with our map data and the type of story we wish to tell, and allow us to export those color codes into our preferred mapping tool.

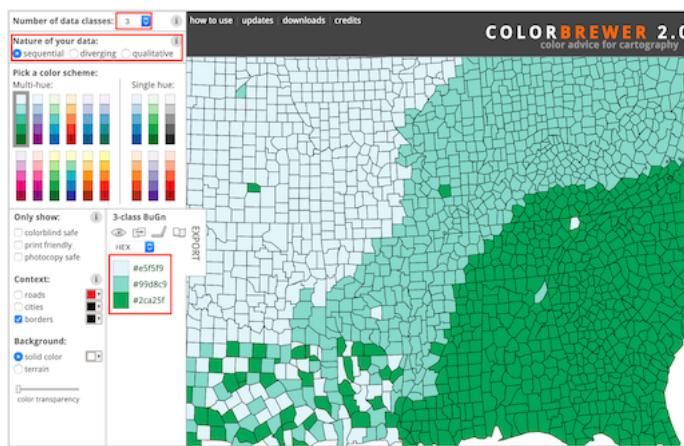


Figure 8.4: The ColorBrewer interface: data classes, type of color scheme, and recommended color codes.

In this section, we'll focus on two important decisions you'll need to make when designing choropleth maps: choosing the type of color palette (sequential, divergent, or qualitative) and the intervals to group together similar-colored data points.

When you open ColorBrewer, the top row asks you to select the number of data classes in your choropleth map, which means the number of intervals or steps in your color range. This design tool can recommend distinct colors for up to twelve data classes, depending on the type of scheme you select. But for now, use the default setting of 3, and we'll return to this topic later when we discuss intervals in more detail further below.

Choose Choropleth Palettes to Match Your Data

One of the most important decisions you'll make when designing a choropleth map is to select the type of palette. You're not simply choosing a color, but the *arrangement of colors* to help readers correctly interpret your information. The

²See also Cynthia A. Brewer, *Designing Better Maps: A Guide for GIS Users* (Esri Press, 2016), https://www.google.com/books/edition/Designing_Better_Maps/gFErrgEACAAJ

rule is straightforward: choose an appropriate color palette that matches your data format, and the story you wish to tell.

ColorBrewer groups palettes into three types—sequential, diverging, and qualitative—as shown in Figure 8.5.

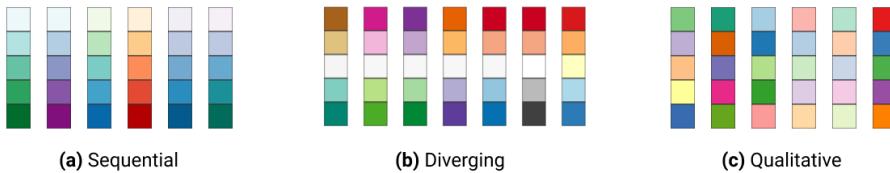


Figure 8.5: Sequential, diverging, and qualitative color palettes from ColorBrewer.

Sequential palettes work best to show low-to-high numeric values. Examples include anything that can be placed in sequence on a scale, such as median income, amount of rainfall, or percent of the population who voted in the prior election. Sequential palettes can be single-hue (such as different shades of blue) or multi-hue (such as yellow-orange-red). Darker colors *usually* represent higher values, but not always.

Diverging palettes work best to show numeric values above and below a standard level (such as zero, the average, or the median). They typically have two distinct hues to represent positive and negative directions, with darker colors at the extremes, and a neutral color in the middle. Examples include income above or below the median level, rainfall above or below seasonal average, or percentage of voters above or below the norm.

Qualitative palettes work best to show categorical data, rather than numeric scales. They typically feature unique colors that stand apart from one another to emphasize differences. Examples include different types of land use (residential, commercial, open space, water) or categories such as a stoplight-colored warning system (green, yellow, and red).

To illustrate the difference between *sequential* and *diverging* numeric values, compare the two maps that display the same data on income per capita in the contiguous US states in 2018 in Figure 8.6. The sequential color palette shows five shades of blue to represent the low-to-high range of income levels, and it works best for a data story that emphasizes the highest income levels, shown by the darker blue colors along the Northeastern coast from Maryland to Massachusetts. By contrast, the diverging color palette shows dark orange for below-average states, dark purple for above-average states, and a neutral color in the middle, and it works best for a data story that emphasizes an economic division between lower-income Southern states versus higher-income East Coast and West Coast states.

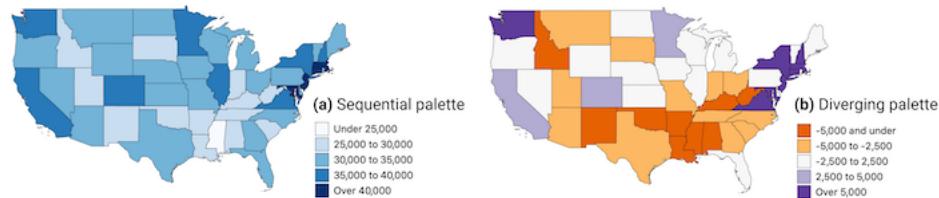


Figure 8.6: Sequential versus diverging color palettes to illustrate per capita income in US dollars in the contiguous states, from American Community Survey, 2018.

After you select data classes and a color palette, ColorBrewer displays alphanumeric codes that web browsers translate into colors. You can select hexadecimal codes (#**fffffff** is white), RGB codes (255, 255, 255 is white), or CMYK codes (0,0,0,0 is white), and export them in different formats, as shown in Figure 8.7, if your preferred map tool allows you to import color palettes.

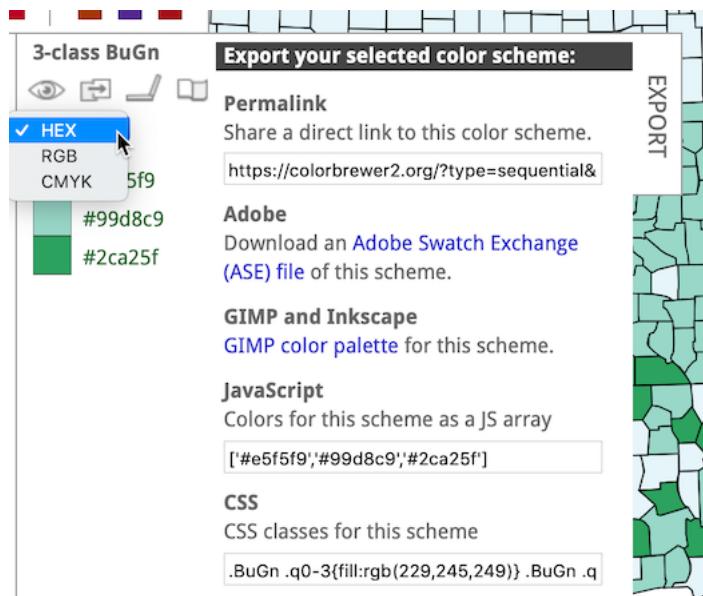


Figure 8.7: Click open the *Export* tab to display your color palette codes in various formats.

Choose Color Intervals to Group Choropleth Map Data

Another important design choice for choropleth maps is the color intervals, which determine how you group and display data by using similar colors. Since

your ability to set intervals varies across different mapping tools, this section will explain broad concepts, and specific map tutorials will demonstrate how to apply them.

Some mapping tools allow you to choose two different *types of color intervals* to show movement up or down a data scale, as shown in Figure 8.8. *Steps* are clearly-marked color dividers, like a staircase, while *continuous* is a gradual change in hue, like a ramp.



Figure 8.8: Two types of color intervals: steps and continuous.

If both options exist, which one is best? There is no clear map design rule about this, but we can gain some wisdom from some excellent Datawrapper Academy posts about choropleth map design. On one hand, some recommend using continuous intervals to show greater geographical diversity, except when it's important to your data story to display a threshold, where steps make sense to show areas above or below a certain line. On the other hand, some point out that people are quite bad at distinguishing different hues on a continuous scale, so recommend using clearly-defined steps to help readers match colors to data values in your legend. Therefore, our general advice is to make design choices that are both honest and insightful: tell the truth about the data and also draw our attention to what matters about this interpretation.

Some mapping tools also allow you to choose how to *interpolate* your data, meaning the method for grouping numbers to represent similar colors on your map. This may involve a two-part decision about step dividers and numerical methods.

First, if you choose *steps*, how many dividers should you use to slice up your data? Once again, there is no clear rule. Fewer steps creates a *coarse* map that highlights broad differences, while more steps creates a *granular* map that emphasizes geographic diversity between areas. But adding more steps also makes differences less visible. Remember that simply adding more colors does not necessarily make a better map. We recommend experimenting with ColorBrewer to raise or lower the *Number of data classes* (also known as steps or dividers) for different types of color palettes, to visualize the consequences of your possible design choices, as shown in Figure 8.9. Make decisions with the best interests of your readers in mind, to represent your data in honest and insightful ways.

Second, whether you choose steps or continuous, which *interpolation* method is the best way to group your data into similar colors on your scale? Map tools may display the options in different ways, as seen in Figure 8.10, and they also may vary depending on whether you selected steps or continuous colors.

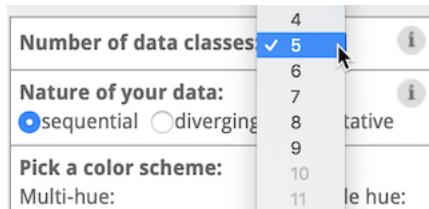


Figure 8.9: Experiment with ColorBrewer data classes (or steps) and color palettes.

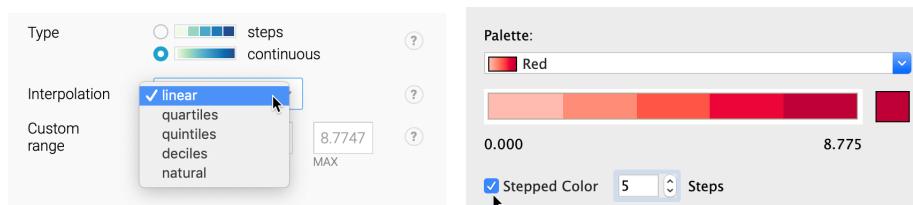


Figure 8.10: Interpolation options in two map tools: Datawrapper (left) and Tableau Public (right).

- *Linear* means that the values are placed in a straight line, from lowest to highest. This method works best when the data are evenly distributed, otherwise the colors draw too much attention to the outliers at the low and high ends of the scale. If your goal is to emphasize the outliers, however, then linear color scale may do the trick.
- *Quantiles* means that the values are divided into groups of an equal number. More specifically, *quartiles*, *quintiles*, and *deciles* mean dividing the values into four, five, or ten groups of equal quantity. This method works best when the data are not evenly distributed, because it uses colors to draw more attention to the diversity of groups inside the scale, not just the low and high ends.
- *Rounded values* are similar to quantiles, but decimals in the scale are replaced with rounded numbers that look nicer to readers' eyes, such as multiples of 5s or 10s.
- *Natural breaks (Jenks)* groups close values together while maximizing differences between other groups. It offers a compromise between linear (which emphasizes the extreme ends) and quantiles (which emphasize internal diversity).
- *Custom* allows you to manually place dividers wherever you wish along the color scale. We generally recommend to *not* use custom settings because they are more likely to create a misleading map, as you'll learn in Chapter 15: Detect Lies and Reduce Data Bias.

TODO: ILYA let's discuss ways to visualize different interpolation methods

by combining mini-histograms of sample data with mini-maps to visually contrast different data and approaches, building on these resources: <https://academy.datawrapper.de/article/117-color-palette-for-your-map>; and also <https://theconversation.com/next-slide-please-data-visualisation-expert-on-whats-wrong-with-the-uk-governments-coronavirus-charts-149329>

Which interpolation method is best? While there are no rigid rules, some methods above work better for different types of data stories. If you wish to emphasize the lows and highs in your data, choose linear because it uses color to draw attention to the extreme ends of the scale. Or if you wish to emphasize geographic diversity in your data, consider quantiles (or any of its cousins) because they use color to differentiate the middle portions of the scale. Or if you're not sure what your data looks like, create a histogram as you learned in Chapter 7 to visualize it and help make wise map design choices. [TODO: demonstrate a histogram with map data here? <https://academy.datawrapper.de/article/294-how-to-customize-stepped-color-scales>]

In any case, be very aware of how color palettes and interpolation dramatically shape the appearance of choropleth maps and how the data appears in readers' eyes. Always create maps show us the story and tell the truth. [TODO: ADD THIS? In general, Datawrapper recommends choosing ranges to make sure readers "see all the differences in the data," rather than hiding them out of sight.... cite <https://academy.datawrapper.de/article/134-what-to-consider-when-creating-choropleth-maps>] TODO ALSO: Review all of the recommendations in the Datawrapper Academy post above and decide ones to include as rules versus recommendations vs neither. For example: "use the same intervals... 0, 25, 50 instead of 0, 15, 50..." runs counter to other advice we give about interpolation.

Normalize Choropleth Map Data

We introduced the concept of normalizing data in Chapter 6: Make Meaningful Comparisons. Normalization means adjusting data that was collected using different scales into a common scale, in order to make more appropriate comparisons. For example, it makes little sense to compare the total number of Covid cases between nations with very different populations, such as 9.61 million cases in the United States (estimated population 328.2 million) and 0.49 million cases in Belgium (estimated population 11.5 million) as of November 6, 2020. A better strategy is to normalize the data by comparing cases per capita (such as 2,928 cases per 100,000 in the United States versus 4,260 per 100,000 in Belgium) to adjust for prior differences in population.

In the same way, choropleth maps work best when they display relative values (such as percentages or per capita rates) rather than raw counts. If you ignore normalization when creating a choropleth map and display raw numbers, you'll essentially recreate a population map, which doesn't tell us anything new. For

example, compare two maps shown in Figure 8.11. They both are about Covid-19 cases in the continental US as of June 26, 2020. Figure 8.11a shows total number of recorded cases per state, and Figure 8.11b shows Covid-19 cases adjusted by the state's population. Darker colors represent higher values. Do you notice any differences in spatial patterns?

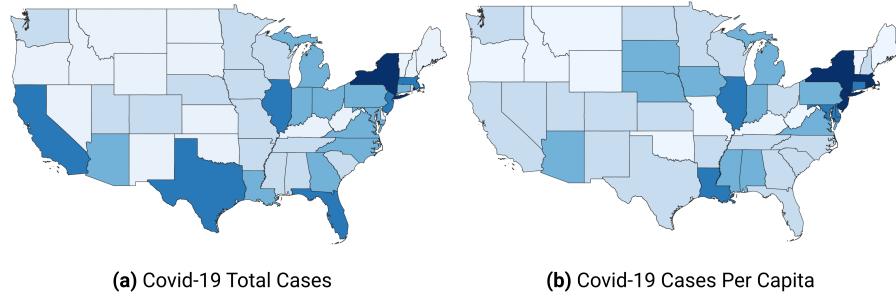


Figure 8.11: Choropleth maps work best with normalized values.

Both maps show Covid-19 data collected by the New York Times and published on GitHub. In the map in Figure 8.11b, we normalized values by dividing the total number of cases by the population in each state, according to the 2018 US Census American Community Survey, the most recent data available on the day of writing. We did not add legends and other important cartographic elements so that you can better focus on interpreting spatial patterns. In both cases, we used Jenks natural breaks for classification.

What are the worst-hit states according to the map showing total Covid-19 counts (shown in Figure 8.11a)? If you are familiar with the US geography, you can quickly tell that these are New York, New Jersey, Massachusetts, Florida, Illinois, Texas, and California. But five of these happen to be some of the most populous states in the US, so it makes sense that they will also have higher Covid-19 cases.

Now, how about the map in Figure 8.11b? You can see that New York and its neighbors, including New Jersey and Massachusetts, have by far the highest rates per capita (per person), which we saw in the first map. But you can also see that in fact California, Texas, and Florida were impacted to a lesser extent than the map on the left had suggested. So the map with per-capita values is a much better illustration to the story about New York being the *first* epicenter of the Covid-19 crisis in the United States.

TODO: Include a very simple normalization calculation to demonstrate raw counts versus normalized data ($\text{Cases} / \text{Population} = \text{Cases Per Capita}$) for two states, such as Texas and New York. Decide whether to show data at a specific moment (May 1 2020 near peak for NYC) or most up-to-date figure at this writing (which will show higher case rates elsewhere). Either way, clearly label the date in the caption.

At this point, you should have a better idea of key principles in map design, and what makes them work (or not) when communicating data images to our eyes. In the next section, we'll begin our first hands-on tutorial with creating a point map. Later in the chapter we'll return to exercises in creating choropleth maps.

Point Map with BatchGeo

Although BatchGeo is not the most powerful mapping tool available, its easy-to-learn interface stands out as one of the fastest ways to create an interactive point map. With the free version, organize up to 250 rows of data in a spreadsheet template, copy and paste them into the platform. BatchGeo will geocode your data and display map markers with colored categories on a Google Map base layer, which can be clicked to show linked titles, text, and small photos (which must be hosted online elsewhere) categories, descriptions, and links (including images hosted elsewhere online). Since the free version does *not* include an account login, enter your email to receive a link to the finished map, like the sample shown in Figure 8.12.

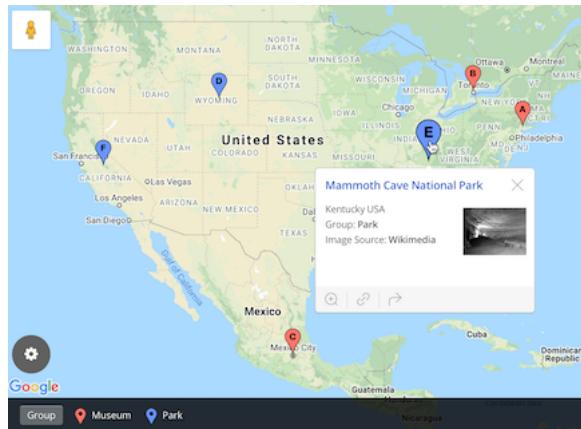


Figure 8.12: Point map of museums and parks with BatchGeo: Explore the interactive version.

To create your own point map with BatchGeo, follow this quick tutorial.

1. Open this BatchGeo data template in Google Sheets, log into your Google account, and go to *File > Make a Copy* to create a version you can edit in your Google Drive.
2. You can delete and enter your own data into the template, as shown in Figure 8.13. Not every address field is required, but adding more will

improve the accuracy of the geocoder. Several columns are optional and can be deleted (such as URL, Image URL, Image Source). If you enter a URL, its active link will appear in the Name field. Also, you can enter an Image URL if a photo is hosted elsewhere online, with an ideal size of 200x200 pixels.

	A	B	C	D	E	F	G	H	I	J
1	Group	Name	Address	City	State	Postalcode	Country	URL	Image URL	Image Source
2	Museum	Metropolitan Museum of Art	1000 Fifth Avenue	New York	New York	10028	USA	https://www.metmuseum.org	https://upload.wikimedia.org	
3	Museum	Royal Ontario Museum	1000 Queen's Park	Toronto	Ontario	M5S 2C6	Canada	https://www.rom.on.ca/en	https://upload.wikimedia.org	
4	Museum	National Museum of Anthropology	Av. Paseo de la Reforma	Mexico City	CDMX		Mexico	https://mnha.inah.gob.mx	https://upload.wikimedia.org	
5	Park	Yellowstone National Park			Wyoming		USA	https://www.nps.gov/yell/	https://upload.wikimedia.org	
6	Park	Mammoth Cave National Park		Kentucky			USA	https://www.nps.gov/maca/	https://upload.wikimedia.org	
7	Park	Yosemite National Park		Mariposa	California		USA	https://www.nps.gov/yose/	https://upload.wikimedia.org	
8										

Figure 8.13: Enter your data into the template, then select all and copy it.

3. Select and copy your data from the template, open the BatchGeo site, click the large field, and paste in your data, as shown in Figure 8.14.

Paste your location data below to map it:

Group	Name	Address	City	State	Postalcode	Country	URL	Image URL
Museum	Metropolitan Museum o...	1000 Fifth Avenue	New York	New York	10028	USA	https://www.metmuseu...	https://upload.wikimed...
Museum	Royal Ontario Museum	1000 Queen's Park	Toronto	Ontario	M5S 2C6	Canada	https://www.rom.on.ca/en	https://upload.wikimed...
Museum	National Museum of Ant...	Av. Paseo de la Reforma	Mexico City	CDMX		Mexico	https://mnha.inah.gob.mx	https://upload.wikimed...
Park	Yellowstone National Park			Wyoming		USA	https://www.nps.gov/yell/	https://upload.wikimed...

(Don't forget to include some header columns - You can also try our [Spreadsheet Template \(Excel\)](#), or hit "Map Now" and try it out with our example data.)

Figure 8.14: Click the large field and paste in your data from the template.

4. Click the *Validate and Set Options* button to inspect or adjust how BatchGeo will upload your data fields and preview a pop-up window, as shown in Figure 8.15. Here you can change the default *Region* from *United States* to other areas of the world. Also, click the *Advanced* button to see more options to change the basemap layer or point symbols. Then click the *Make Map* button.
5. In the *Map* window, click the *Save and Continue* button to enter a title and set other options, such making your map *public* or *unlisted*, as well as customizing the layout, as shown in Figure 8.16. Insert your email address to receive a link to your online map. Click *Save Map*.

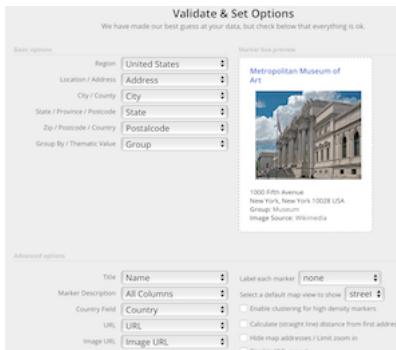


Figure 8.15: Click the *Validate* button to inspect and preview your data and advanced options.

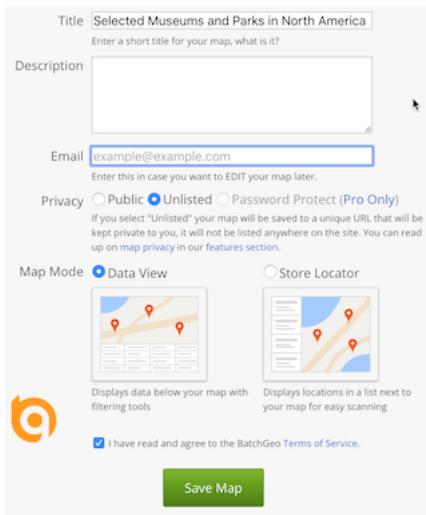


Figure 8.16: Be sure to enter your email address to receive a link to your online map.

6. Check your email for a message from BatchGeo with a link to your live map, and another link to make edits, as shown in Figure 8.17. The email also will include a code to embed your live map on the web, which you'll learn about in Chapter 10.

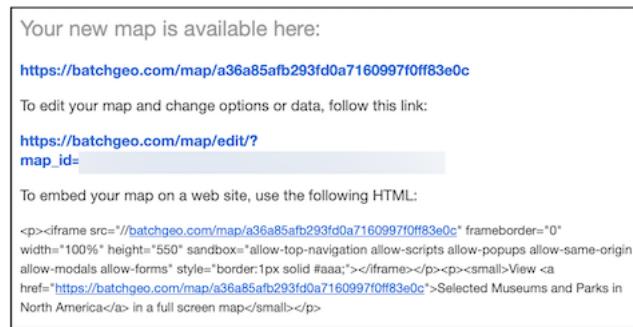


Figure 8.17: Save the BatchGeo email with links and an embed code for your map.

Overall, BatchGeo is a free and easy-to-learn point map tool, but one with several limitations. You can only create maps with 250 points or less, and only a few options to customize their appearance, unless you upgrade to the paid version. Also, while the tool geocodes your location data, it does not show you the latitude and longitude coordinates, but see the Geocode section of Chapter 3 for a Google Sheets add-on that will show you the geocoordinates. See more info on the BatchGeo support page and also their post about power-user tips.

Now that you've learned about one tool to create point maps, let's compare it with a second tool, Google My Maps, which includes some additional features.

Point Map with Google My Maps

TODo BELOW: Explain how Google My Maps is different from Google Maps; mention at the top there are more options than points; DECIDE if this sample data about Nigerian airports is the best fit, since it's great to have international examples but unsure if Shapefile-to-CSV transformation makes sense here, as new users probably need the tool to geocode their data.

My Maps is Google's service that allows users to create custom maps on top of Google Maps. The maps that you create will stay in your Google account and will not in any way interfere with the Google Maps product that you and others use to look up directions or find nearby restaurants. You may choose to share your map either through embedding it on your website or providing a direct link to your colleagues or friends.

Google My Maps is one of the simplest ways to build basic point, line, and polygon maps, although it limits your styling options. My Maps is most powerful when it comes to collaboration, and you can invite anyone with a Google account to jointly work on the map.

In this section, we will look at building a point map of airports in Nigeria, as is shown in Figure 8.18. We will create a map, change a baselayer, import point data, style points, and share the map.

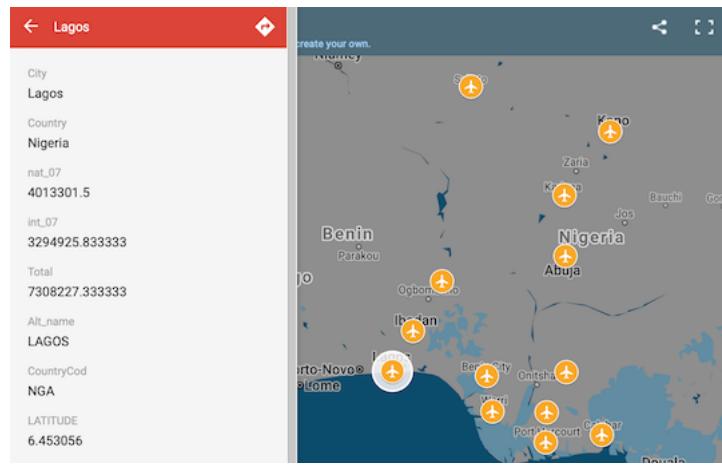


Figure 8.18: A map of airports in Nigeria built using Google My Maps.

Create a New Map in My Maps

Navigate to Google My Maps. In the upper-left corner, click `+ Create a New Map` button, as shown in Figure 8.19.



Figure 8.19: A map of airports in Nigeria built using Google My Maps.

You will see a typical Google Maps with no data. Click on the current title (`Untitled map`), and add appropriate title and description in the modal window that appeared (see Figure 8.20 for inspiration).

Before we add any points, let's change the basemap to something less boring. At the bottom of the control window, open `Base map` dropdown, and pick one of nine available basemaps. For this tutorial, we chose `Dark Landmass`.

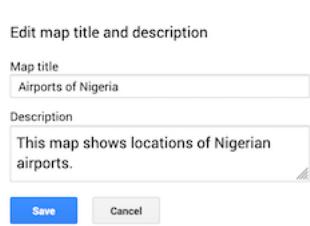


Figure 8.20: Add title and description to your map.

Let's now proceed to the most important step—adding data. You can download a dataset of Nigerian airports that we got from the World Bank as a Shapefile and then converted to a CSV.

Under *Untitled layer* item, click *Import* button, and drag-and-drop the CSV file. Once the data file is uploaded, My Maps will ask which columns contain location data. In our case, these are *LATITUDE* and *LONGITUDE* columns, as shown in Figure 8.21.

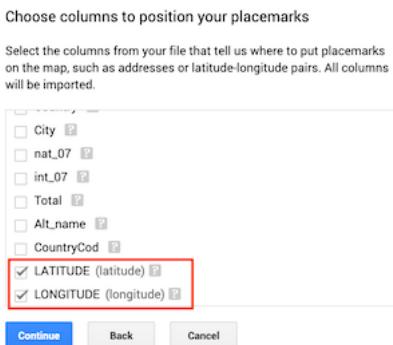


Figure 8.21: Check LATITUDE and LONGITUDE as your location columns.

Once the two boxes are checked, click *Continue*. Another window will pop up, asking which column to use to annotate points. Choose *City*, as shown in Figure 8.22, and then *Finish*.

It will take a few moments for My Maps to create a new layer, which will be added to the layer menu as *nigeria-airports.csv*. Once the layer is created, My Maps will center the map to fit the points.

Let's replace the original blue markers to orange airport symbols. In layers menu, hover over *All items* and click the paint bucket symbol on the right. Change “All items” text to “Airports”, choose orange color, and click on *More icons* to find an airport symbol (we recommend using Filter to search for “airport”, or simply scroll down to Transportation section). The marker in the



Figure 8.22: Choose City as the title for your markers.

layers menu will change to an orange airplane, as shown in Figure 8.23.

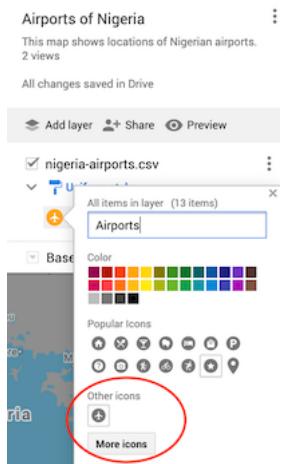


Figure 8.23: In My Maps, you can change marker colors and icons.

Click on the layer name, which by default is set to the name of imported file (`nigeria-airports.csv`), and change it to *Nigerian Airports*. Alternatively open a kebab menu to the right of the layer name, and choose *Rename this layer*, and then click *Save*.

You can accompany each marker with a label. Click *Uniform style* under the layer, and choose *Alt_name* in the *Set labels* dropdown menu. You will see alternative airport names, such as *BENIN*, displayed beneath the markers.

Click *Preview* to see how the map looks like outside of the My Maps editing studio. Additionally, you can add photos, directions, lines, etc., and display or hide columns by checking or unchecking them when clicking on a map point.

Share Your Google My Map

If you are happy with the result, click *Share* button underneath the map's title and description. In the modal window, make sure *Enable link sharing* is activated, as shown in Figure 8.24, and copy the generated link. You can share with link with anyone, with or without a Google account, and they will be able to see your map.

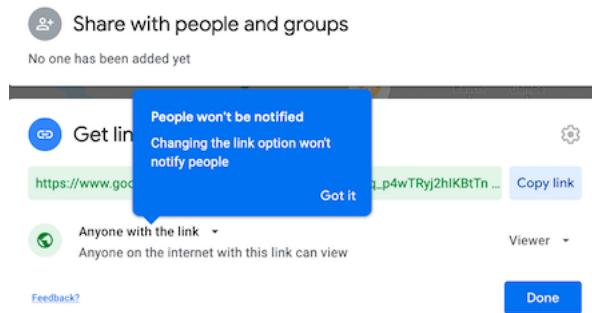


Figure 8.24: Make sure anyone with the link can view your map before you share it.

If you want to embed the map as an iframe, from the main kebab menu to the right of the map title, choose *Embed on my site*, as shown in Figure 8.25. This will generate an HTML embed code that you can copy to your website, as described in Chapter 10: Embed on the Web.

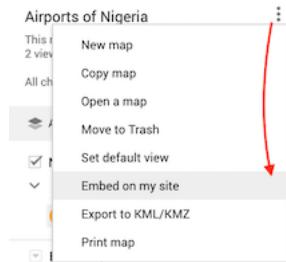


Figure 8.25: My Maps can generate an iframe code to include the map on your own website.

Going Beyond Points

Google My Maps has more powerful features for map making. Instead of uploading datasets with latitude/longitude pairs, you can use simple addresses (and My Maps will take care of geocoding), or add markers by clicking on the

map using *Add marker* feature. You can classify points based on a property, and use different colors to represent them.

You are not limited to just point maps. You can also draw your own shapes, including lines and polygons. You can add data to multiple layers. Unfortunately, Google My Maps has no comprehensive documentation, so you have to explore the studio yourself if you want to create more complex projects.

Symbol Point Map with Datawrapper

We first introduced you to the free and easy-to-learn Datawrapper tool in Chapter 7: Chart Your Data. It's also offers powerful features to create different types of maps, with professional-looking design elements. With Datawrapper you can start to work right away in your browser, with no account required unless you wish to save and share your work online.

In this section, you'll learn how to create a symbol point map, which displays specific locations using variable-sized shapes or colors to represent their data values. Our sample symbol map displays about 300 major US cities as point locations with two variables: the 2019 estimated population (using circle size) and the percent change in population since 2010 (using circle color), as shown in Figure 8.26. Remember that we use *point* data to create symbol maps, but *polygon* data to create choropleth maps, and you'll learn how to design those with Datawrapper in the subsequent section. Later in the book, we'll explain how to embed your interactive Datawrapper maps on the web in Chapter 10.

Datawrapper splits the process of creating a map into four steps: select map, add data, visualize, then publish and embed. To create your own symbol point map, follow this tutorial.

1. Open the US Cities Population Change 2010-2019 data in Google Sheets. Read the notes to understand its origin and some data issues. We downloaded city population data for 2010-2019 from the US Census. But during this time period, some cities were newly incorporated or merged with outlying areas, which skews their population data over time. Also, we collected data for 5 major cities in Puerto Rico, a US territory, but they do not appear in the Datawrapper map for reasons explained in step 4.

Good maps often require cleaning up messy data as described in Chapter 5. In our spreadsheet we narrowed the original list down to about 300 cities with more than 100,000 residents in either 2010 or 2019. Also, since we're relying on Datawrapper to correctly identify *place names*, we combined *city* and *state* into one column to improve geocoding accuracy. Learn more about place name geocoding at the Datawrapper Academy. Also, we created a new column named *Percent Change*, which we calculated this way: $(2019 - 2010) / 2010 * 100$.

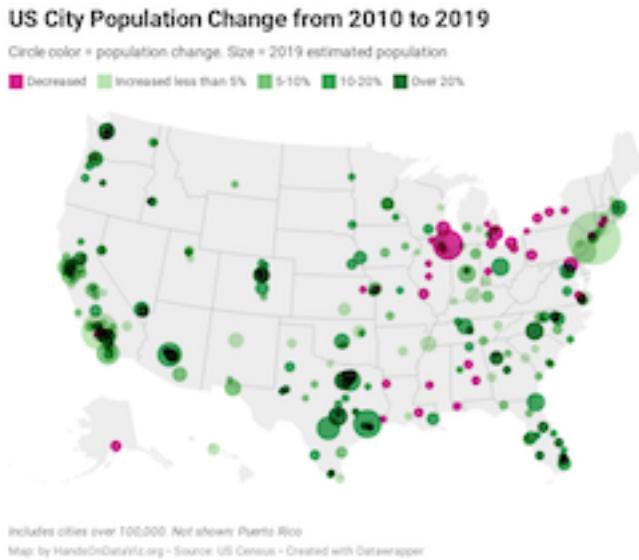


Figure 8.26: Symbol point map of US city population growth with Datawrapper. Explore the interactive version.

2. In the Google Sheet, go to *File > Download* and select the Comma-Separated Values (CSV) format to save the data to your local computer.
3. Open Datawrapper, click on *Start Creating*, then the *New Map* button, and select *Symbol map* as shown in Figure 8.27.
4. In the *Select your map* screen, enter *USA* and search for options. Note that we selected *USA > States** as our best current option, because *USA > States and Territories* does not yet correctly display geocoded locations for Puerto Rico, a US territory. Proceed to the next screen.
5. In the *Add your data* screen, click the *Import your dataset* button. In the next window, click the *Addresses and Place Names* button because our data is organized this way. In the *Import* window, click to *Upload a CSV file*, and select the file you downloaded above.
6. In the *Match your columns* screen, select the *City-State* column to be *Matched as Address*, then scroll down to click the *Next* button, as shown in Figure 8.28. In the next screen click *Go*, then see your geocoded data displayed on a map in the following screen.
7. Click the *Visualize* button to *Refine* your map. Our goal is to display two variables: 2019 population as the circle size, and percent change as the

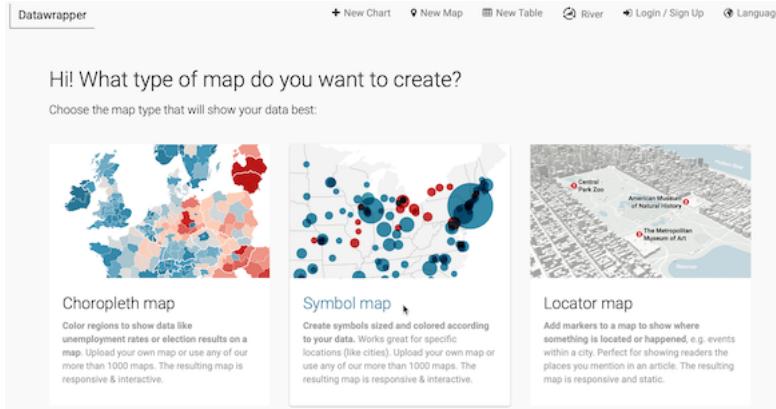


Figure 8.27: Start to create a symbol map in Datawrapper.

Match your columns

Please select which column in your dataset contains "Address".

MATCHED AS ADDRESS	MATCH AS ADDRESS	MATCH AS ADDRESS	MATCH AS ADDRESS
City-State	Population2010	PopEstimate2019	PctChange2010-15
New York, New York	8175133	8336817	1.98
Los Angeles, California	3792621	3979576	4.93
Chicago, Illinois	2695598	2693976	-0.06
Houston, Texas	2099451	2320268	10.52
Phoenix, Arizona	1445632	1680992	16.28
Philadelphia, Pennsylvania	1526006	1584064	3.80
San Antonio, Texas	1327407	1547253	16.56
San Diego, California	1307402	1423851	8.91

First row as caption NEXT

Figure 8.28: Select the *City-State* column to be matched as the *Address*.

circle color. Under *Symbol shape and size*, select the *circle* symbol, to be sized by *Pop Estimate 2019*, with a maximum symbol size of 25 pixels. Under *Symbol colors*, select the *Percent Change 2010-2019* column, as shown in Figure 8.29.

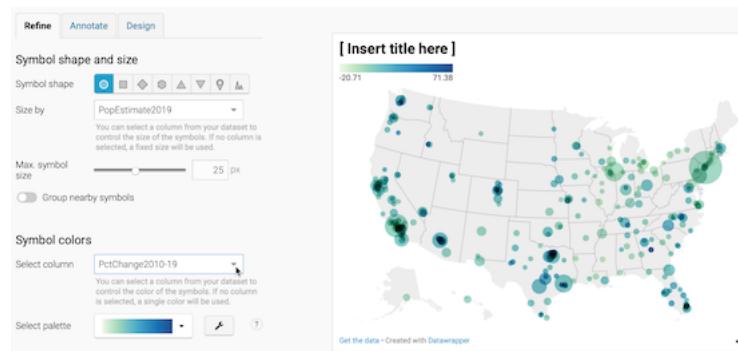


Figure 8.29: Refine your map by selecting data to display symbol shapes, sizes, and colors.

8. If you wish to customize the color palette and intervals to match our example, click the *wrench symbol* next to the palette. Click the *Import colors* button and paste in five hexadecimal codes from ColorBrewer, as described in the Choropleth Design section. The first code is dark pink, followed by a 4-class sequential green: #d01c8b, #bae4b3, #74c476, #31a354, #006d2c. See Figure 8.30.



Figure 8.30: Create a new color palette by importing five hexadecimal color codes from ColorBrewer.

9. To continue customizing intervals to match our example, set the steps to 5 and *Custom*. Manually type in custom intervals for below 0% (bright pink), 0 to 5% (light green), and so forth up the scale. Click the *More options* button, and under *Legend*, change *Labels* to *custom*, and click each label to edit the text that appears on the map menu, as shown in Figure 8.31. Learn more about these options in the Datawrapper Academy post on customizing your symbol map.

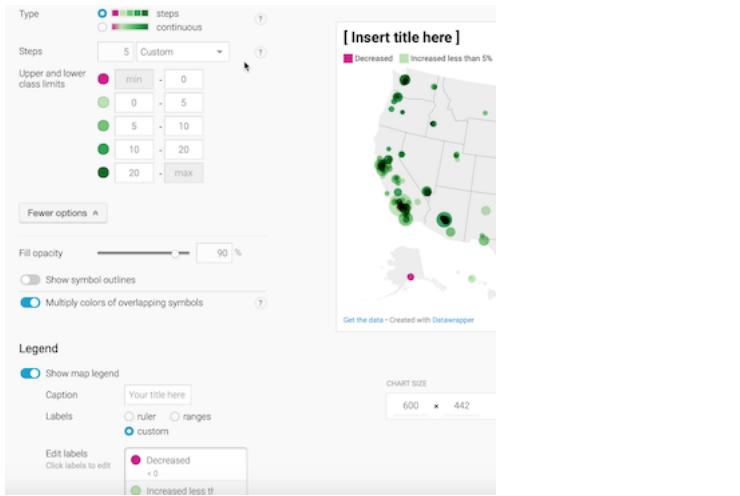


Figure 8.31: Customize the interval ranges and edit the legend.

10. Under the *Visualize* screen, click the *Annotate* tab to insert a title, source notes, credits, and customize the tooltips as described by Datawrapper Academy.
11. Click *Proceed* or advance to the *Publish & Embed* screen to share your work with others. If you logged into your free Datawrapper account, your work is automatically saved online in the *My Charts* menu in the top-right corner of the screen. Also, you can click the blue *Publish* button to generate the code to embed your interactive map on your website, as you'll learn about in Chapter 10: Embed on the Web. In addition, you can *add your chart to River* if you wish to share your work more widely by allowing other Datawrapper users to adapt and reuse it. Furthermore, scroll all the way down and click the *Download PNG* button to export a static image of your map. Additional exporting and publishing options require a paid Datawrapper account. Or, if you prefer not to create an account, you can enter your email to receive the embed code.

For assistance and additional options, see the Datawrapper Academy support pages on symbol maps.

Now that you've created a symbol point map with Datawrapper, in the next section we'll build our skills with this tool to create a choropleth map.

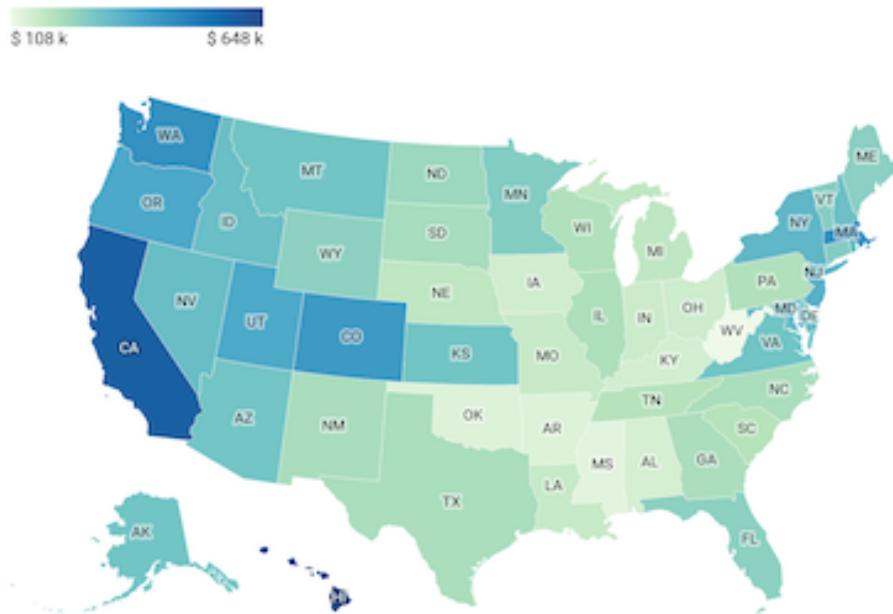
Choropleth Map with Datawrapper

Now that you've learned how to use Datawrapper to create charts and symbol maps, now let's design a choropleth map. These maps are best to show patterns across geographic areas by coloring polygons to represent data values. Datawrapper offers a wide collection of common geographical boundaries, including world regions, states and provinces, and also hexagons (cartograms), counties, congressional districts, and census tracts for the United States.

In this section, you'll create a choropleth map of typical home values for US states in August 2020 according to the Zillow Home Value Index, as shown in Figure 8.32. The index reflects typical home values (meaning those in the 35th to 65th percentile range, around the median) for single-family residences, condos, and co-ops, and it is smoothed and seasonally adjusted.

Typical Home Values in US States, August 2020

Statewide values are highest in Hawaii, California, and Massachusetts, according to Zillow



Map: by HandsOnDataViz.org • Source: Zillow • Created with Datawrapper

Figure 8.32: Choropleth map of 2020 home values in US states with Datawrapper. Explore the interactive version.

Datawrapper splits the process of creating a map into four steps: select map, add data, visualize, then publish and embed. To create your own choropleth

map, follow this tutorial.

1. Open the Home Value Index data in Google Sheets, which we downloaded from the Zillow research site. Read the notes to understand its origin and some data issues. For example, Washington DC was included in the data, but it does not yet appear in the Datawrapper map of US states, because the District of Columbia is not a state.

Good maps often require cleaning up messy data as described in Chapter 5. In our spreadsheet we removed all of the columns except two, August 2019 and August 2020, and we also inserted a *Percent Change* column, which we calculated this way: $(2020 - 2019) / 2019 * 100$. Also, we're fortunate that Datawrapper easily recognizes US state names and abbreviations. If you're mapping other types of geographic areas, learn more about international ISO codes and place name geocoding at the Datawrapper Academy.

TODO: DECIDE whether to keep basemap or change to one of the others that now reflect DC. DECIDE whether to mention other types of codes recognized by Datawrapper, which vary with map type, such as FIPS and ANSI.

2. In the Google Sheet, go to *File > Download* and select the Comma-Separated Values (CSV) format to save the data to your local computer.
3. Open Datawrapper, click on *Start Creating*, then click the *New Map* button, and select *Choropleth map* as shown in Figure 8.33. No login is required to create a map, but you should sign up for a free account in order to save your work and publish your map online.
4. In the *Select your map* screen, choose your geographic boundaries, and in this case choose *USA > States*, as shown in Figure 8.34, then click *Proceed*. Since Washington, DC is not a state, it does not appear in this map of US States.

Hint: If Datawrapper does not list your preferred map outline, you can upload your own custom geography data in GeoJSON or TopoJSON format, which you will learn more about in the GeoJSON and geospatial data formats section of Chapter 14.

5. In the *Add your data* screen, you can manually enter data for each area, which would be fine for just a few, but not for 50 states. Instead, scroll down and click the *Import your dataset* button, as shown in Figure 8.35. Datawrapper will explain that your data must include either *Names* (such as *California*) or *ISO-Codes* (standardized two-letter codes for states, such as *CA*, or three-letter codes for nations, such as *USA*). Since we prepared our data this way, click *Start Import*.

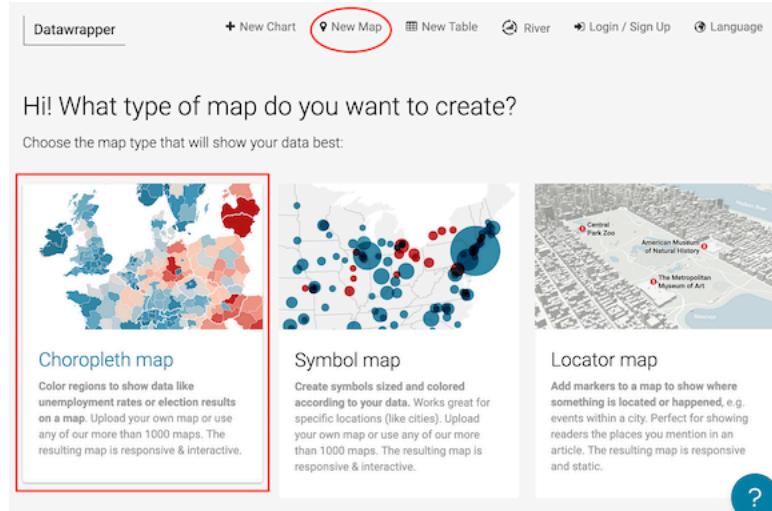


Figure 8.33: In Datawrapper, click *New Map*, and choose *Choropleth*.

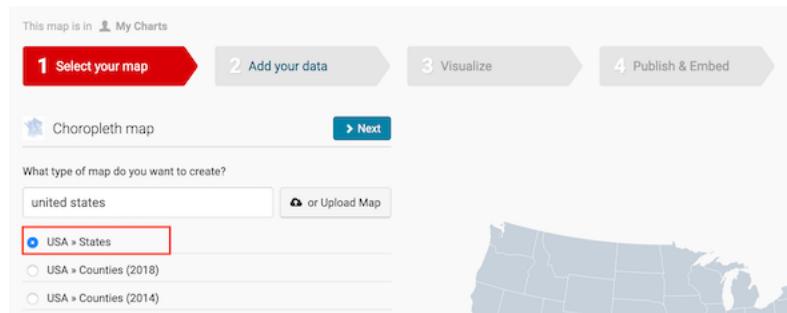


Figure 8.34: Choose *USA - States* for your map outline.

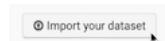


Figure 8.35: Scroll down below the *add data* table to import your dataset.

TODO: Add screenshot of names/codes drop-down menu, which is helpful because it shows how Datawrapper labels each geography, and one could copy and paste this into a spreadsheet and use VLookup. Also include image of this import button, since it's hidden far below the table that most users see.

6. On the *Import your dataset* screen, instead of pasting your data, we recommend that you click to *upload a CSV file* and select the file you downloaded in step 2.
7. In the *Match your columns* screen, click the column that matches up with ISO codes, as shown in Figure 8.36. You may need to scroll down a bit to click the *Next* button, then *Continue*.

Please select which column in your dataset contains "ISO-CODES".

Code	Name	Aug2019 Homi	Aug2020 Homi	Pct Change 20
CA	California	556814	587412	5.50
TX	Texas	208048	215658	3.66
NY	New York	325168	335502	3.18
FL	Florida	246057	258257	4.96
IL	Illinois	205652	209335	1.79
PA	Pennsylvania	195056	204876	5.03
OH	Ohio	150429	160093	6.42
MI	Michigan	173521	182327	5.07
GA	Georgia	202179	213026	5.37
NC	North Carolina	202455	214726	6.06
NJ	New Jersey	340740	353518	3.75

First row as caption

NEXT

Figure 8.36: Select the data column that contains matching ISO codes.

8. In a similar way on the next screen, click the column of data values that you initially want to map, and click *Matched as values*. For this tutorial, select *Aug2020 Home Values*, then scroll down to click *Next*, then *Go*, then *Proceed*. You'll be able to map other data values in a later step.

TODO Above: Add screenshot of Aug2020 Home Values column, as reviewer requested.

9. In the *Visualize* screen, under the *Refine* tab, click the *wrench symbol* next to the color palette to review the *default* map settings, as shown in Figure 8.37. Never automatically accept the default map, but it's a good place to start and explore how factors shape its appearance.

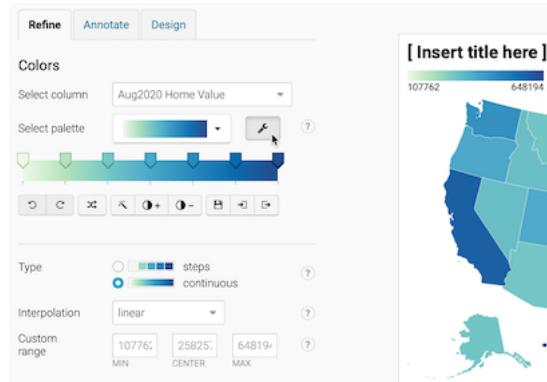


Figure 8.37: Under the *Refine* tab, click the *wrench symbol* to see the default map settings.

Let's review key concepts we first introduced in the Design Choropleth Colors & Intervals section of this chapter. The default map shows a *continuous* green-to-blue color palette, with *linear* interpolation, which means the home values are distributed in a straight line up the scale. These colors and intervals work better for a data story that emphasizes the low and high extremes.

10. In the *Refine* tab, experiment with different types of interpolation to change how values are assigned to colors. For example, change from *linear* to *quartiles*, which groups the data into four groups of equal size, as shown in Figure 8.38. This map works better for a data story that emphasizes geographic diversity, since we see more contrast between states in the middle range.

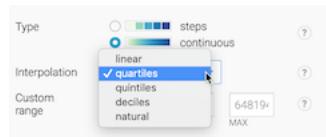


Figure 8.38: Under the *Refine* tab, change the interpolation from *linear* to *quartiles* and see how the map changes.

11. Experiment with other colors, intervals, and data columns. Change the palette from *sequential* to *diverging* colors, which display a neutral color in the middle range and two dark colors at the extremes. Change from a *continuous* gradient to *steps*, and choose different numbers of dividers. Change the data column to *Pct Change 2019-20* to normalize the choropleth map data as discussed earlier in this chapter, since home values are

so different across the country. For example, see the map of percent change in home value from 2019 to 2020, with a diverging red-to-blue palette, 5 steps, and rounded values in Figure 8.39.

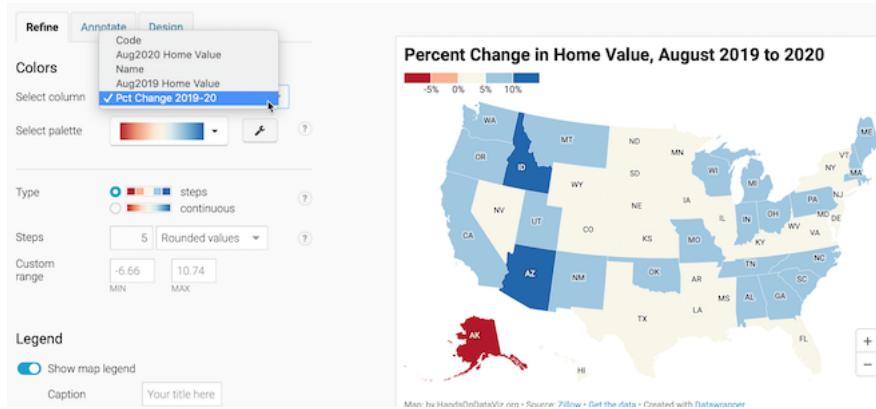


Figure 8.39: Experiment with other colors, intervals, and data columns to find true and meaningful stories.

Which data columns, colors, and intervals make the best map? There's no easy answer, since there's more than one way to make a true and meaningful map. But keep two principles in mind. First, make sure that you honestly show the data, rather than hide or disguise it. Second, reflect on what kind of data story you believe is important to tell, since design choices emphasize different interpretations of the data. Review our guidance in the Design Choropleth Colors & Intervals section.

Let's move on to finalize the labels and styling of the map before we publish and share it with others.

- Under the *Refine* tab, customize the legend format. For example, to convert long numbers (such as 107762) into abbreviated dollars (\$ 108 k), we selected *custom format* and inserted the code (\$ 0 a).

TODO above: Add screenshot of where to enter codes in legend format, as reviewer requested.

- Under the *Annotate* tab, add a title, description, and source credits, to add credibility to your work. You can also add map labels and customize tooltips that will display when readers hover their cursor over different states. The easiest way to edit tooltips is to click on blue column names, or format them using their drop-down menus, to make the proper codes appear in double curly brackets, as shown in Figure 8.40. Learn more about customizing tooltips from Datawrapper Academy.

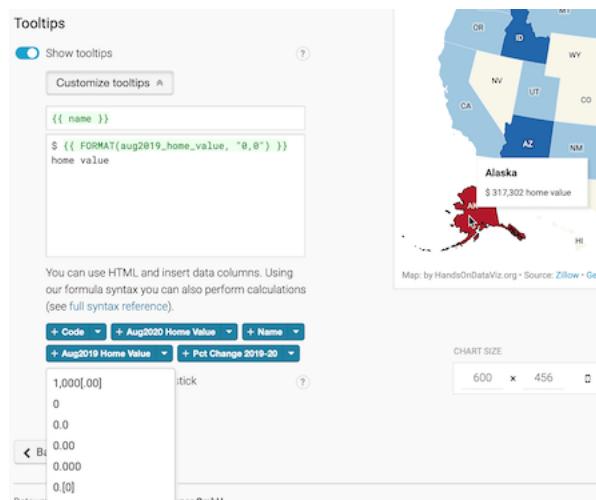


Figure 8.40: To edit tooltips, click the blue column names or use drop-down menus to format the codes.

TODO above: ADD example of how to customize tooltips and add annotations, as reviewer requested here.

- Finally, click *Proceed* or advance to the *Publish & Embed* screen to share your work with others. Follow the prompts, or the more detailed Datawrapper tutorial above, to obtain an embed code to your interactive map, and learn more about your next steps in Chapter 10: Embed on the Web.

Learn more about choropleth map design in this excellent series of posts by the Datawrapper Academy.

Now that you've learned how to create a choropleth map using one tool, Datawrapper, let's compare the process using a different tool, Tableau Public.

Choropleth Map with Tableau Public

We first introduced you to the free Tableau Public desktop application (for Mac or Windows) when building scattercharts and filtered line charts in Chapter 7. Now let's use the same tool to create an interactive choropleth map, and compare the process with the Datawrapper tool we learned in the prior section. Tableau Public can create many different types of map for geographical place names or ISO codes it already recognizes, such as nations, states, counties, and airports. But Tableau Public cannot geocode street addresses by itself, so you'll

need to obtain their latitude and longitude with another tool, such as those described in the geocode section of Chapter 3. Furthermore, if you want to upload customized map boundaries, learn how to Create Tableau Maps from Spatial Files on the support page.

In this section, we will create a choropleth map of healthcare spending per country as a percentage of their gross domestic product (GDP), as shown in Figure 8.41. Remember that choropleth maps work best when we normalize the data to show relative, rather than absolute, numbers. Creating a map of total health spending per country would not be very meaningful, as larger nations tend to have larger economies, so we'll base our map on the percentage of their economy that is spent on healthcare.

Before we start, you should obtain and install the free Tableau Public desktop application if you don't have it yet. It is available for Mac or Windows. You will need to enter an email address to download the application.

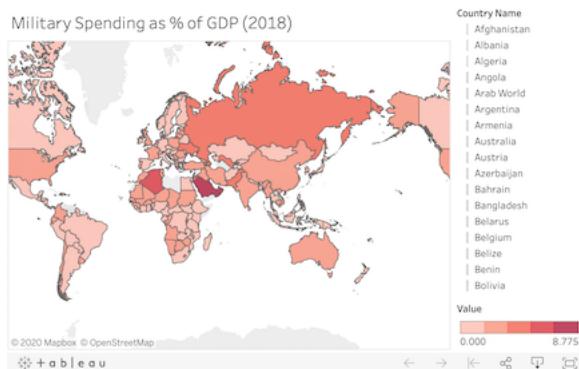


Figure 8.41: Choropleth map of healthcare spending with Tableau Public. Explore the interactive version. Data from the World Bank.

Let's look at the steps involved to create a choropleth from Figure 8.41 in detail.

1. Open the Healthcare Spending by Nation as Percent of GDP data in Google Sheets, which we downloaded from the World Bank. Examine the data and the notes.

Good maps often require cleaning up messy data as described in Chapter 5. In our spreadsheet we removed rows for nations that did not report any data. Tableau Public recognizes many different types of geographic names (such as cities and nations), so we will rely on the tool to deal with any spelling issues and properly place all of them on the map.

2. In the Google Sheet, go to *File > Download* and select Comma-Separated Values (CSV) format to save the data to your local computer.

3. Launch Tableau Public. When you first open it, you will see the *Connect* menu on the left-hand side that displays file formats you can upload. Choose the *Text file* format and upload the healthcare spending CSV data file you've just downloaded in the previous step.

Note: Tableau lets you access data directly from Google Sheets that live in your Drive using *Connect > To a Server* option. So instead of downloading a CSV file in step 2, you could have made a copy of the sheet, and connected to it directly.

4. In the *Data Source* screen, inspect the dataset, which contains three columns: Country Name, Country Code, and Health Spending As % of GDP. Notice that a small globe appears at the top of the Country Name and Country Code columns, which shows that Tableau Public successfully recognized these as geographic data, rather than string or text data. Sometimes Tableau does not recognize location data automatically, so you need to manually change the data type. To do so, click the data type icon (e.g. globe or a green # for numeric values), and then choose *Geographic Role > Country/Region* as shown in Figure 8.42.

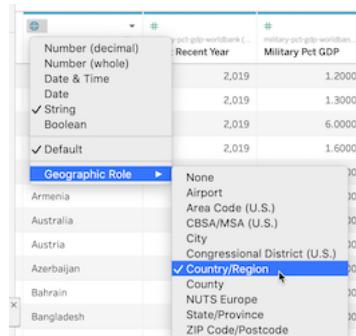


Figure 8.42: Make sure Tableau Public knows that the Country Name column contains geographic data.

5. In the bottom-left corner, click the orange *Sheet 1* button to create a worksheet with your first visualization, as shown in Figure 8.43.
6. In *Sheet 1*, create your choropleth map using a two-step process, as shown in Figure 8.44. First, drag-and-drop the *Country Name* field into the middle of the worksheet (alternatively to the Detail box of the Marks card) to create the map. The default view is the symbol map, which we need to replace with a polygon map. To add colored polygons, drag-and-drop the *Health Spending As % of GDP* field into the *Color* box of the *Marks* card to transform it into a choropleth map.

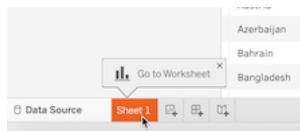


Figure 8.43: Click the orange button to go sheet 1 where you can create your map.

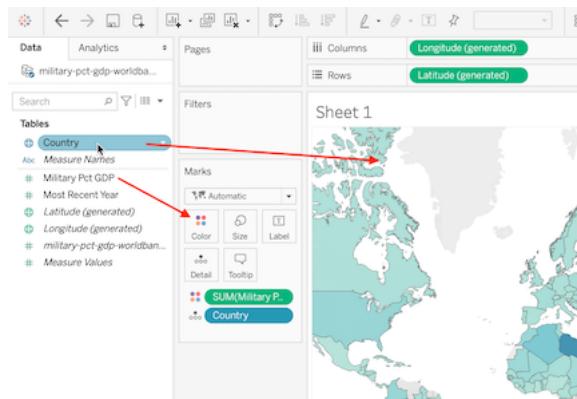


Figure 8.44: Drag and drop *Country Name* to the center of the sheet, then *Health Spending As % of GDP* to the *Color* box in the *Marks* card.

7. Tableau Public may hide the map legend behind the *Show Me* menu in the upper-right corner, so click the menu to shrink it and display your legend.
8. You can change the color palette by clicking the Color box of the Marks card, and then *Edit colors*. Change the palette to *Green*, and change it from continuous to steps, as shown in Figure 8.45.

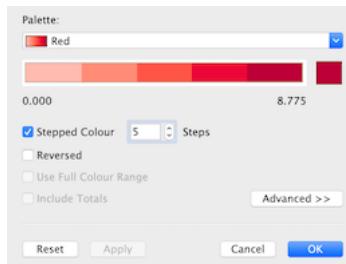


Figure 8.45: Change the color scheme to Green with 5 steps.

9. When you hover over countries, you will notice a tooltip that tells you

the name of the country and gives you the percent value. It is generally well-formatted as our initial data table had proper column headers. But we can make the tooltip even better. Click the Tooltip box of the Marks card, change the first instance of `Country Name` to just `Country` (do not change the grayed-out text inside < and > as these are variable names), and add a % sign at the end of the second row, as shown in Figure 8.46.



Figure 8.46: Change tooltip text to make it more user-friendly.

10. Let's make our map title more meaningful. Double-click the default *Sheet 1* name just above the map to bring up the *Edit Title* window, and change the name of your chart to *2017 Healthcare Spending by Country as % of GDP*.
11. At this point the data is loaded and should be displayed correctly, so we are going to create the final layout that include map's title and credits, the legend, and is appropriate for sharing. At the bottom-left of the program, create a *New Dashboard*, as shown in Figure 8.47. Dashboards in Tableau are layouts that can contain visualizations from multiple sheets, as well as text boxes, images, and other elements, creating rich exploratory interfaces. In this tutorial, we will stick to just a single sheet that contains our choropleth map.

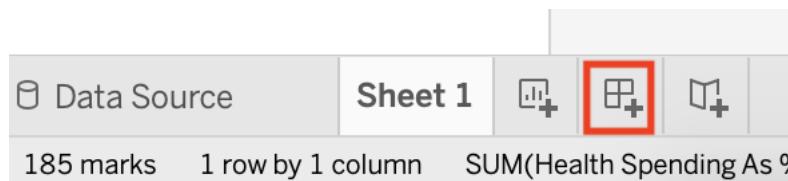


Figure 8.47: Before you publish the map, create a new dashboard to finalize your layout.

12. In your Dashboard 1 tab, change the size of the dashboard to Automatic so that the map is responsive and occupies 100% of the width on all devices. Drag and drop Sheet 1 to the *Drop sheets here* area, as shown in Figure 8.48. This will copy the map, the title, and the legend from Sheet 1.

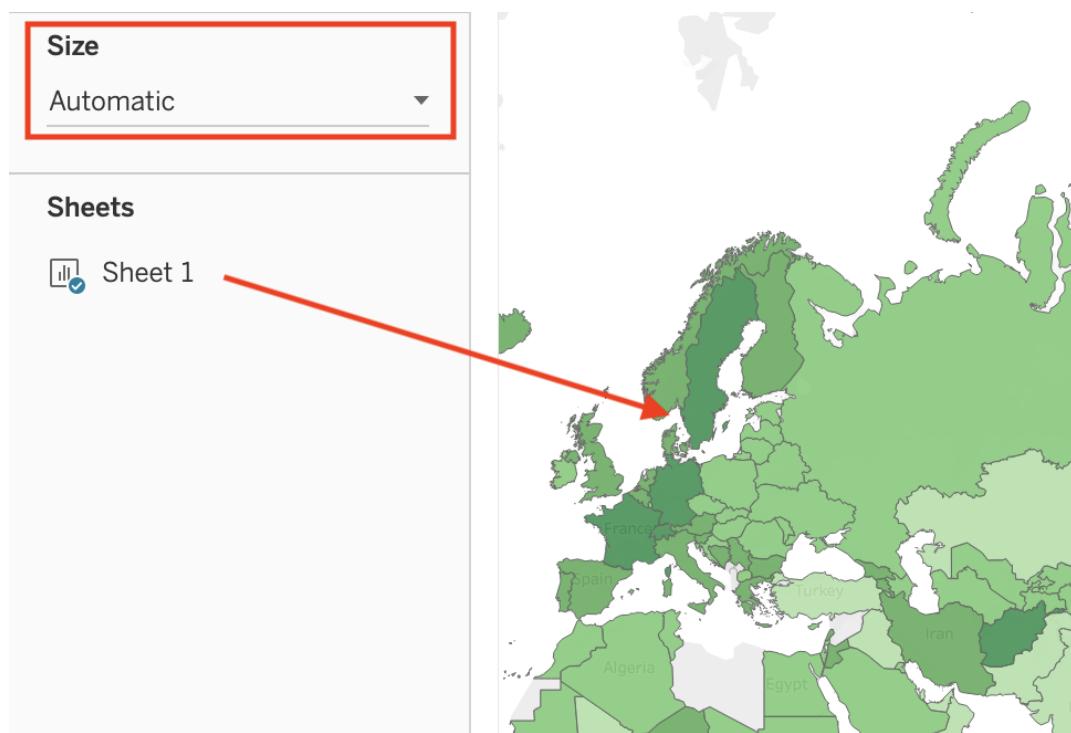


Figure 8.48: To create a responsive dashboard, change the Size to Automatic.

13. Right-click the upper part of the map legend, and select *Floating*, as shown in Figure 8.49. Now you are able to place your legend directly on top of the map to save space. Drag and drop it to one of the map's corners.

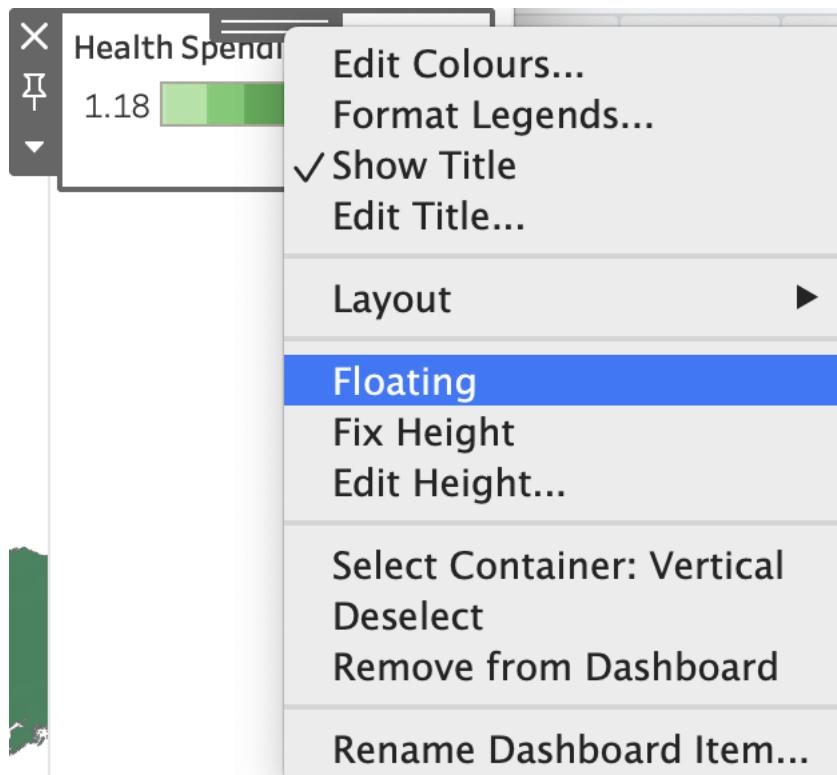


Figure 8.49: To place the legend on top of the map, make sure it is floating.

14. Finally, let's add a text block with data source underneath the map. From the Objects menu in the left-hand side, drag and drop *Text* to the lower half of the map. In the *Edit Text* window that appears, type *Data by the World Bank, 2017*, and click OK. Initially the text area will occupy half the height of the screen, so resize it like you would resize any window on your computer.

And we're done! Make sure you position your map's center and zoom level as you want it to be visible by others. In this case, the best would be to have a world view as we are showing data for most countries, although you may want to zoom in to a specific continent. Once you are ready to publish and share the map, go to *File > Save to Tableau Public*. In the pop-up window, log in

to your account if requested. Give it a title, such as Healthcare Spending, and click Save. See how to embed the map as an iframe in Chapter 10.

Warning: Tableau may not be the best tool to create choropleth maps where you want to have full control of color breaks. By default, Tableau uses a linear color scheme that, as we've learned earlier in the chapter, is prone to highlighting outliers, and there is no straightforward way to change the intervals to non-linear methods such as quantiles. If you are not happy with the way the linear scale represents your data, you can either filter your data to remove outliers from the map, or see Andy Kriebel's VizWiz tutorial to use table calculations to group items into quantiles.

Now that you know that Tableau can be used not just for charts, but for maps as well, let's see how to use Socrata's visualization capabilities to build a map that always uses the most current data available.

Current Map with Socrata Open Data

TODO: revise to emphasize how the map continuously pulls the most current information directly from the open-data repository; in this case, happens to be filtered point map of hospitals in Texas.... TODO: replace with a better example of frequently-updated "live" data?

Socrata is a database service that is used by government agencies, cities and countries to make open data available to the public. It offers user-friendly ways to view, filter, and export data. In addition, the Socrata platform includes built-in support to create interactive charts and maps, which can be embedded in other websites (including your own).

One advantage of creating data visualizations directly on an open data platform is that the chart or map is linked to the data repository. For example, if the Socrata platform administrator updates the data table, then a Socrata dataviz based on that data will be automatically updated, too. This may be especially useful for "live" data that is continuously updated by agency administrators such as fires, crimes, and property data.

In this section, we will build an interactive point map of hospitals in Texas using General Hospital Information dataset by Medicare, which you can see in Figure 8.50.

Generally, in order to create a map in Socrata, you need to be a registered user, and the dataset you wish to visualize has to contain a column with location data. This is not just an address column (such as **3500 Gaston Avenue, Dallas, TX**), but a geocoded column that contains latitude and longitude values.

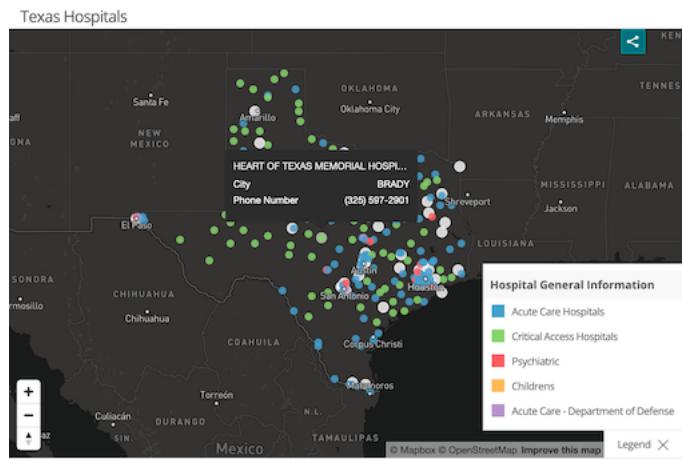


Figure 8.50: In this tutorial, we will build a point map of hospitals in Texas using Socrata.

Sign Up for Socrata Account

Navigate to Data.Medicare.gov click *Sign In* button in the upper-right corner. Scroll down to *Sign Up* link. Follow the instructions, including setting up two-step authentication (2FA), to create a free account.

Note: You can still practice creating maps in Socrata without being logged in. You won't be able to save or share your work, however.

Once you have an account, log in using your credentials (you may find this tutorial about logging-in with 2FA useful). Navigate to your profile by clicking your username in the upper-right corner and make sure you *Accept Terms and Conditions*, otherwise you won't be able to save your draft map.

This username and password are only valid for Data.Medicare.gov, not other websites that use Socrata.

Create Your First Socrata Point Map

Navigate to the Hospital General Dataset, and in the menu on the right-hand side choose *Visualize > Launch New Visualization*, as shown in Figure 8.51. This will open up a *Configure Visualization* studio where you can create the map.

In the top menu, click *Map* (globe icon between a scatter chart icon and a calendar). You will see an updated layout of the studio, with the map in the middle, and *Map Layers* and *Map Settings* items in the side menu on the left, as shown in Figure 8.52.

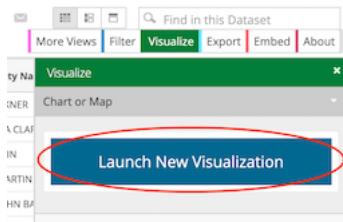


Figure 8.51: Go to Visualize > Launch New Visualization.

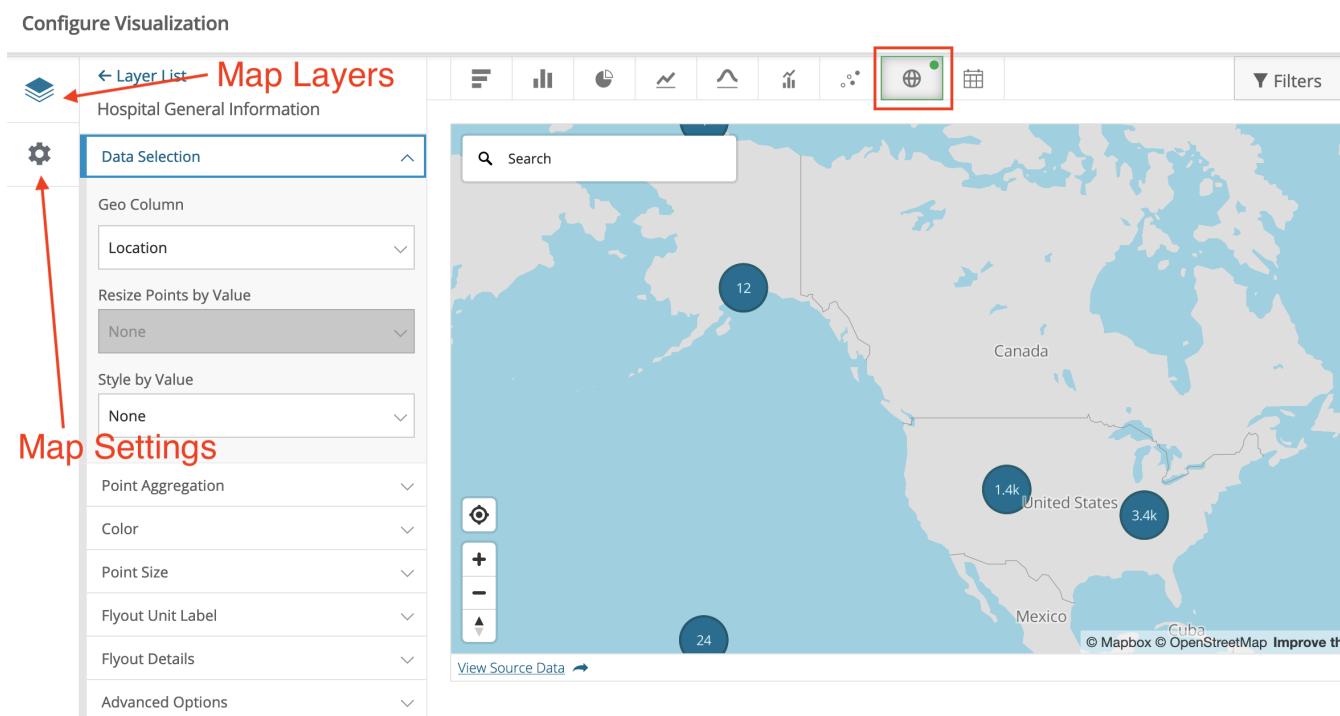


Figure 8.52: Your studio should look similar to this once you choose Map as the visualization type.

Socrata was able to determine which column contains geospatial value, and automatically set Geo Column value to *Location* in the *Layer List* menu. By default, points are clustered (grouped together), so instead of seeing individual hospital locations, you see bubbles with numbers (such as 12 in Alaska).

Let's first select only hospitals that are located in the southern state of Texas. To do so, go to *Filters > Add filter*. The dropdown menu lists all columns (or fields) of the dataset, where we should choose *State*. In the newly appeared State dropdown, choose TX (for Texas) as shown in Figure 8.53, and scroll down and click Apply. Socrata should zoom in on the map and center on Texas. Close *Filters* window to free screen space.

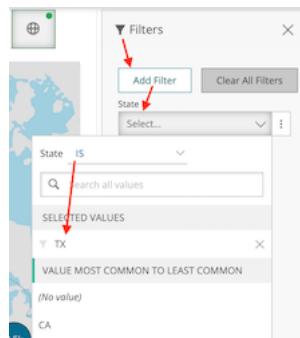


Figure 8.53: Select Texas as the only value for State field.

Let's now *disaggregate* the map so that we can see individual hospitals instead of clusters. Go to *Map Settings > Clusters*, and bring the *Stop Clustering at Zoom Level* slider to 1, as shown in Figure 8.54. You will see the map now shows individual points.

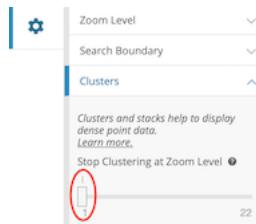


Figure 8.54: To show individual points instead of clusters, set Stop Clustering at Zoom Level to 1.

In the same accordion menu, change *Basemap > Type* to *Dark* to bring the map a fashionable 2020 look. In *General*, set Title to *Texas Hospitals*, and hide data table below the map by unchecking the *Show data table below visualization* box. Under *Map Controls*, uncheck *Show Search Bar* and *Show Locate Button* to get rid of unnecessary elements. Feel free to experiment with other settings as well.

Now, let's return back to the *Map Layers* menu and choose our *Hospitals General Information* point layer. You can notice that in *Data Selection* accordion menu, *Resize Points by Value* is grayed out. That is because the dataset doesn't contain columns with continuous variables that can be transformed to point sizes. Instead, we can use *Style by Value* option to classify categorical points. The dataset contains multiple variables that can be effectively visualized, such as *Hospital Type*, *Emergency Services* (a yes/no category), *Mortality national comparison* and others. Let's stick with *Hospital Type*, as is illustrated in Figure 8.55.

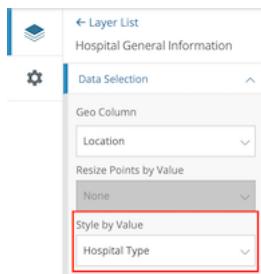


Figure 8.55: Let's display different types of hospitals in different colors.

If you look at the bottom-right corner of the map, you should notice a minimized *Legend* control. Click on it to see what each color represents.

Change the color palette (in *Color* menu) from *Categorical 1* to *Categorical 2*, which includes a wider range of unique colors. You can also use *Custom...* item to set individual colors, as well as change the order of categories in the legend.

To change what is shown in tooltips when you hover or click on points, go to **Flyout Details**, and set Flyout Title to *Facility Name*, adding city and phone number as additional flyout values, as is shown in Figure 8.56.

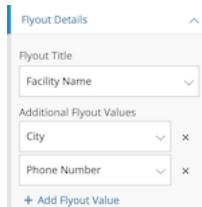


Figure 8.56: To edit tooltip information, use Flyout Details menu item.

At this point you should have a fully-functional interactive map showing all hospitals in Texas, colored according to their type. Before you can share it, you need to save it as a draft, and publish.

Save Draft and Publish

In the lower-right corner, click *Save Draft* button. Give your map a name, and hit *Save*. The gray ribbon at the top will tell you it is still a draft, and you can go ahead and *Publish...* it.

Now you can embed the map on your website as an iframe. To do so, click the *Share* button in the upper-right side of your map (see Figure 8.57), and copy the generated code from *Embed Code* text area (Figure 8.58). Learn more in Chapter 10: Embed on the Web.

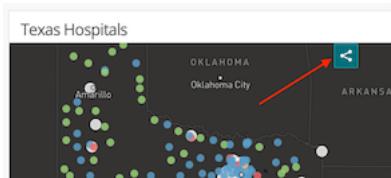


Figure 8.57: Click *Share* button to bring up *Share and Embed* window.

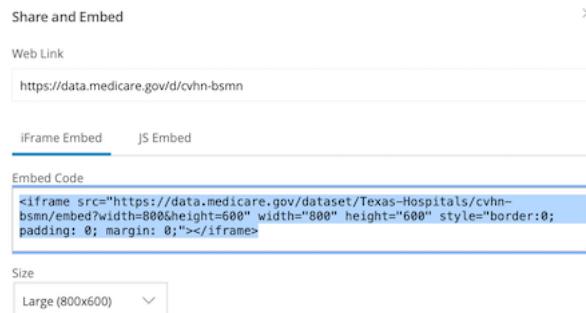


Figure 8.58: Copy iframe code to embed this map in another website.

Limitations of Socrata

But there are limitations to creating your chart or map on an open data repository platform. First, if the agency stops using the platform, or changes the structure of the underlying data, your online map (or chart) may stop functioning. Second, you are generally limited to using datasets and geographic boundaries that already exist on that platform.

If these limitations concern you, a simple alternative is to export data from the open repository (which means that any “live” data would become “static”), and import it into your preferred dataviz tool, such as Tableau.

A second, more advanced alternative, is to learn to pull live data from Socrata using an API (Application Programming Interface). That requires coding skills that are beyond the scope of this book. Visit the official documentation to learn more about Socrata API.

Summary

In this chapter, we looked at free mapping platforms to create simple point and polygon maps. Google My Maps is a good choice for point maps that can be created in collaboration with others. If the data you are interested in lives on Socrata platform, you might be able to create a point map within the platform itself, and embed it as an iframe in your own website. Tableau is another very powerful tool to build and share polygon and point maps.

In reviewing all these tools, we only scratched the surface and showed simple examples to get you started quickly. All platforms allow layering data to create powerful exploration mapping visualizations.

None of the platforms required special geospatial data, as all were smart enough to perform geocoding and know the boundaries and coordinates of objects given to them. In Chapter 14, we will talk more about geospatial data, how it can be obtained, stored, modified, and shared.

Chapter 9

Table Your Data

You might be surprised that a data visualization book which emphasizes charts and maps also includes a chapter on creating tables. We don't normally think about data tables as a type of visualization. But depending on your data and the story you wish to tell about it, sometimes a table is the most appropriate way to present information, especially when it's an interactive table on the web. Tables make sense when readers want to look up a specific row of data that's highly relevant to them, such as their local community or an organization they belong to, which can be too hard to identify inside a large chart or map. Also, tables work best when readers wish to precisely compare individual values to one another, but not necessarily to the rest of the dataset. Finally, tables work better than charts when there is no broad visual pattern to emphasize, and work better than maps when there is no particular spatial pattern. Before you start designing a chart or map, consider whether it makes more sense to create a table instead. Sometimes the best visualization is simply a good table.

TODO: Break this out into two separate sections and place in the body of the chapter, before Datawrapper?

Of course, you can create tables with spreadsheet tools. For example, with Google Sheets you can format rows and columns of data, and download the table as a static PDF image that you can place in a document, or convert into a PNG or JPG image to easily insert on a web page. Sometimes this simple approach makes sense. Also, you can use the spreadsheet pivot table feature in Chapter 3 to create a more sophisticated cross-tabulation, and export it as an image to insert in a document or website. With Google Sheets, you can also publish any of your tables online, and embed them on a web page as we'll discuss in Chapter 10, so that whenever you update your Google Sheet, the current data will automatically appear on the web page. Furthermore, you can also use Tableau Public, a tool we previously introduced in Chapter 7 and Chapter 8 to create a highlight table, which automatically colors the backgrounds of cells to draw your eye to higher versus lower values.

In this chapter, we'll focus on creating interactive tables, which have many advantages over static tables when publishing your information online, rather than only in print. First, interactive tables allow readers to search by keyword for specific details that interest them, which is vital when you present long tables with lots of rows. Second, readers can sort interactive tables in ascending or descending order for any column, which enables them to quickly scan those near the top or bottom of a long list. Finally, you'll also learn how to insert *sparklines*, or tiny charts that visually summarize data trends in each row, and automatically place them inside your interactive table. Sparklines blend the best qualities of tables and charts by making it easier for readers to visually scan for trends while skimming down the rows and columns of your data table. Later in Chapter 10: Embed on the Web, you'll learn how to integrate your interactive table into your website.

In this chapter, you'll learn about table design principles and how to use Datawrapper, a tool we introduced in Chapter 7: Chart Your Data and Chapter 8: Map Your Data to create an interactive table with sparklines.

Table Design Principles

Let's begin with some principles of good table design, similar to how we learned about chart design in Chapter 7 and map design in Chapter 8. Jonathan Schwabish, an economist who specializes in creating policy-relevant data visualizations, offers his advice in recent publications about creating tables that communicate well with multiple audiences.¹ Here's a summary of several of his key points, which also appear in Figure 9.1.

1. Make column headers stand out above the data.
2. Use light shading to separate rows or columns.
3. Left-align text and right-align numbers for easier reading.
4. Avoid repetition by placing labels only in the first row.
5. Group and sort data to highlight meaningful patterns.

In addition, Schwabish and others recommend using color to highlight key items or outliers in your data, a topic we'll discuss later in Chapter 16: Tell and Show Your Data Story.

¹Jon Schwabish, "Thread Summarizing 'Ten Guidelines for Better Tables,'" Twitter, August 3, 2020, <https://twitter.com/jschwabish/status/1290323581881266177>; Jonathan A. Schwabish, "Ten Guidelines for Better Tables," *Journal of Benefit-Cost Analysis* 11, no. 2: 151–78, accessed August 25, 2020, <https://doi.org/10.1017/bca.2020.11>; Jonathan Schwabish, *Better Data Visualizations: A Guide for Scholars, Researchers, and Wonks* (Columbia University Press, 2021), <https://cup.columbia.edu/book/better-data-visualizations/9780231193115>

Category	Food	Color	Calories per serving
Fruit	Banana	Yellow	105
	Apple	Red	95
	Blueberries	Blue	42
Vegetable	Kale	Green	34
	Carrot	Orange	26
	Eggplant	Purple	10

Figure 9.1: A sample table that illustrates selected design principles.

When creating cross-tabulations to illustrate data correlations and possible causal relationships, statistician Joel Best offers two more design recommendations.²

6. Place the independent variable (the suspected cause) at the top in the column headers, and the dependent variable (the possible effect) on the side for each row.
7. Calculate percentages from the raw numbers in a downward direction, so that each value of the independent variable (the suspected cause) totals 100 percent.

Let's apply these design principles by constructing two tables that calculate percentages: the wrong way versus the right way. The first table calculates percentages in the wrong direction—horizontally—which confuses us about whether X is correlated with Y. But the second table calculates percentages in the correct direction—vertically—which shows how X is correlated with Y, and may signal a causal relationship, as shown in Figure 9.2

TODO above: Find better data to replace his imaginary data below and rewrite text to match.

Overall, the core principles of table design reflect similar concepts we previously discussed in chart and map design. Organize your presentation of the data with the readers' eyes in mind, to focus their attention on the most important elements of the story, so that they “take away” the most meaningful interpretation of the information. Do the visualization work for them, so that you don't have to rely on them to draw the same mental connections in their own minds.

²Joel Best, *More Damned Lies and Statistics: How Numbers Confuse Public Issues* (Berkeley, CA: University of California Press, 2004), https://www.google.com/books/edition/More_Damned_Lies_and_Statistics/SWBr7D6VavoC, pp. 31-35.

Table 2. Calculating Percentages Across (still confusing)

	Right-Handed	Left-Handed	Total
Law-abiding	91% (810)	9% (80)	100% (890)
Delinquent	82% (90)	18% (20)	100% (110)

But watch what happens when we calculate the percentages down, as shown in Table 3, so that each column totals 100 percent. Suddenly, the pattern becomes clear: in our study, the percentage of lefties who are delinquent (20 percent) is twice that of righties (10 percent). These percentages help us understand the pattern in the numbers.

Table 3. Calculating Percentages Down (much clearer)

	Right-Handed	Left-Handed
Law-abiding	90% (810)	80% (80)
Delinquent	10% (90)	20% (20)
Total	100% (900)	100% (100)

Table 2 illustrates what we mean when we say that someone has calculated the percentages "in the wrong direction" or "in the wrong way." In general, percentages should be calculated so

Figure 9.2: TODO: Create side-by-side 'bad' and 'better' tables, with real data and superimposed arrows to show the direction of calculation, to improve on this placeholder from Best 2014 p. 32.

Remove any clutter or unnecessary repetition that stands in the way of these goals.

Now that you understand several key principles of table design, see how some (but not all) are built directly into the Datawrapper tool featured in the next section.

Datawrapper Table with Sparklines

In this section, you'll learn how to create an interactive table with Datawrapper, the free online drag-and-drop visualization tool we previously introduced to create charts in Chapter 7 and maps in Chapter 8. You can start creating in Datawrapper right away in your browser, even without an account, but signing up for a free one will help you to keep your visualizations organized. Remember that you'll probably still need a spreadsheet tool, such as Google Sheets, to compile and clean up data for large tables, but Datawrapper is the best tool to create and publish the interactive table online.

You'll also learn how to create sparklines, or tiny line charts that quickly summarize data trends. This chart type was refined by Edward Tufte, a Yale professor and data visualization pioneer, who described sparklines as "datawords... intense, simple, word-sized graphics."³ While Tufte envisioned sparklines on a

³Edward R. Tufte, *Beautiful Evidence* (Graphics Press, 2006), <http://books.google.com/books?isbn=0961392177>, pp. 46-63.

static sheet of paper or PDF document, you'll create them inside an interactive table, as shown in Figure 9.3. Readers can search by keyword, sort columns in ascending or descending order, and scroll through pages of sparklines to quickly identify data trends that would be difficult to spot in a traditional numbers-only table.

Life expectancy at birth by nation, 1960-2018

The table displays life expectancy data for seven nations, categorized by continent. Each row shows the nation's name, continent, initial life expectancy in 1960, final life expectancy in 2018, and the difference between the two. Sparklines in the 'Difference' column illustrate the trend over time.

Nation	Continent	Life expectancy 1960 – 2018	Difference
Afghanistan	Asia	32.4 → 64.5 32	
Albania	Europe	62.3 → 78.5 16	
Algeria	Africa	46.1 → 76.7 31	
Angola	Africa	37.5 → 60.8 23	
Antigua and Barbuda	North America	62 → 76.9 15	
Argentina	South America	65.1 → 76.5 12	
Armenia	Asia	66 → 74.9 9	

Figure 9.3: Table with sparklines. Explore the interactive version.

In this tutorial, you'll create an interactive table with sparklines to visualize differences in life expectancy at birth from 1960 to 2018 for over 195 nations around the world. Overall, life expectancy gradually rises in most nations, but a few display “dips” that stand out in the tiny line charts. For example, Cambodia and Vietnam both experienced a significant decrease in life expectancy, which corresponds with the deadly wars and refugee crises in both nations from the late 1960s to the mid-1970s. Sparklines help us to visually detect patterns like these, which anyone can investigate further by downloading the raw data through the link at the bottom of the interactive table.

While it's possible to present the same data in a filtered line chart as shown in Chapter 7, it would be difficult for readers to spot differences when shown over 180 lines at the same time. Likewise, it's also possible to present this data in a choropleth map as shown in Chapter 8, though it would be hard for readers to identify data for nations with smaller geographies compared to larger ones. In this particular case, when we want readers to be able to search, sort, or scroll through sparklines for all nations, the best visualization is a good table.

To create your own interactive table with sparklines, follow this tutorial, which

we adapted from Datawrapper training materials and their gallery of examples:

1. Open our cleaned-up World Bank data on life expectancy at birth, 1960 to 2018 in Google Sheets.

To simplify this tutorial, we downloaded life expectancy at birth from 1960 to 2018 by nation, in CSV format, from the World Bank, one of the open data repositories we listed in Chapter 4: Find and Question Your Data. In our spreadsheet, we cleaned up the data, such as removing nations with 5 or fewer years of data reported over a half-century, as described in the Notes tab in the Google Sheet. Using the VLookup spreadsheet method from Chapter 3, we merged in columns of two-letter nation codes and continents from Datawrapper. We also created two new columns: one named *Life Expectancy 1960* (intentionally blank for the sparkline to come) and *Difference* (which calculates the difference between the earliest and the most recent year of data available, in most cases from 1960 to 2018). See the Notes tab in the Google Sheet for more details.

2. Go to Datawrapper, click on *Start Creating*, and select *New Table* in the top navigation. You are not required to sign in, but if you wish to save your work, we recommend that you create a free account.
3. In the first *Upload Data* tab, select *Import Google Spreadsheet*, paste in the web address of our cleaned-up Google Sheet, and click *Proceed*. Your Google Sheet must be *shared* so that others can view it.
4. Inspect the data in the *Check and Describe* tab. Make sure that the *First row as label* box is checked, then click *Proceed*.
5. In the *Visualize* screen, under *Customize Table*, check two additional boxes: *Make Searchable* (so that users can search for nations by keyword) and *Stripe Table* (to make lines more readable).
6. Let's use a special Datawrapper code to display tiny flags before each country's name. In the *Nation* column, each entry begins with a two-letter country code, surrounded by colons, followed by the country name, such as :af: Afghanistan. We created the *Nation* column according to the Combine Data into One Column section of Chapter 5: Clean Up Messy Data.

Note: To learn more about flag icons, read the Datawrapper post on this topic and their list of country codes and flags on GitHub.

7. In the *Visualize* screen, under *Customize columns*, select the third line named *Nation*. Then scroll down and push the slider to *Replace country codes with flags*, as shown in Figure 9.4.

Customize columns

Name
Code
Nation 
Continent
Life expectancy 1960
1960
1961

Select all | none | invert

Show on desktop mobile ?

Style     100 %

Border    

Alignment    

Color cells based on categories

Fixed column width

Show as bar chart
You can only turn numeric columns into bar charts.

Replace country codes with flags

Format 1x1 4x3 circle

Algeria	:dz:	 Algeria
Angola	:ao:	 Angola
Antigua and Barbuda	:ag:	 Antigua and Barbuda
Argentina	:ar:	 Argentina
Armenia	:am:	 Armenia
Aruba	:aw:	 Aruba
Australia	:au:	 Australia
Austria	:at:	 Austria
Azerbaijan	:az:	 Azerbaijan
Bahamas	:bs:	 Bahamas
Bahrain	:bh:	 Bahrain
Bangladesh	:bd:	 Bangladesh
Barbados	:bb:	 Barbados
Belarus	:by:	 Belarus

Figure 9.4: Customize the *Nation* column and push slider to replace codes with flags.

8. Let's hide the first two columns, since they're no longer necessary to display. In the *Visualize* screen under *Customize columns*, select the *Name* column, then scroll down and un-check the boxes to *Show on desktop and mobile*. Repeat this step for the *Code* column. A “not visible” symbol (an eye with a slash through it) appears next to each customized column to remind us that we've hidden it.
9. Now let's color-code the *Continent* column to make it easier for readers to sort by category it in the interactive table. In the *Visualize* screen under *Customize columns*, select the *Continent* column, then scroll down and push the slider to select *Color cells based on categories*. In the dropdown menu, select the column *Continent*, and click on the *Background: customize colors* button. Select each continent and assign them different colors, as shown in Figure 9.5.

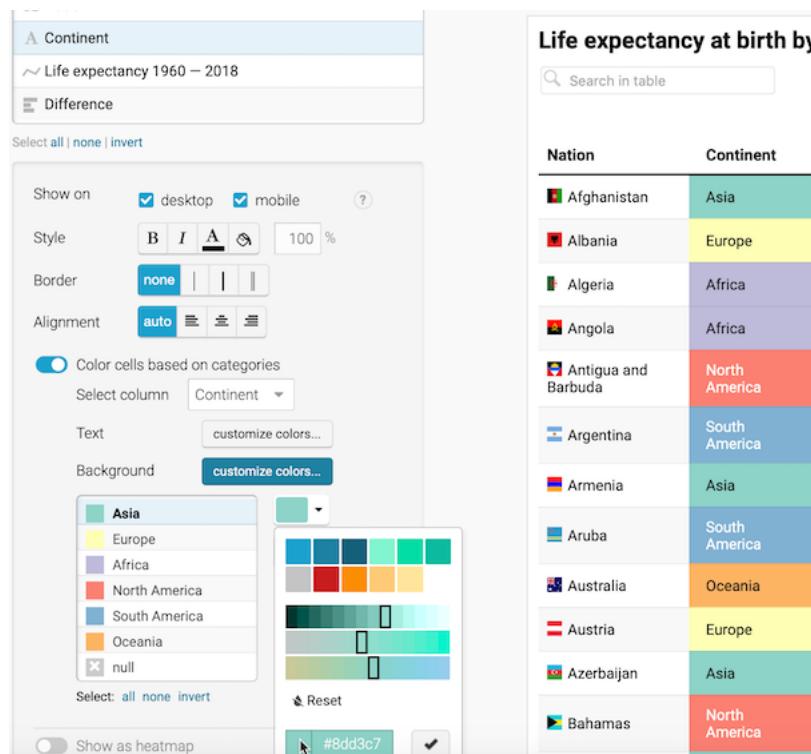


Figure 9.5: Customize the *Continent* column and push slider to color cells based on categories.

Tip: To choose colors for the six continents, we used the ColorBrewer design tool as described in Chapter 8, and selected a 6-class qualitative scheme. Although

this tool is designed primarily for choropleth maps, you can also use it to choose table and chart colors.

10. Now let's prepare our data to add sparklines, or tiny line charts, to visually represent change in the *Life expectancy 1960* column, which we intentionally left blank for this step. Before you begin, you must change this column from textual data (represented by the A symbol in the *Customize columns* window) to numerical data (represented by the # symbol). At the top of the screen, click on the 2. *Check and Describe* arrow to go back a step. (Datawrapper will save your work.) Now click on the table header to edit the properties for *column E: Life Expectancy 1960*. On the left side, use the drop-down menu to change its properties from *auto (text)* to *Number*, as shown in Figure 9.6. Then click *Proceed* to return to the *Visualize* window.

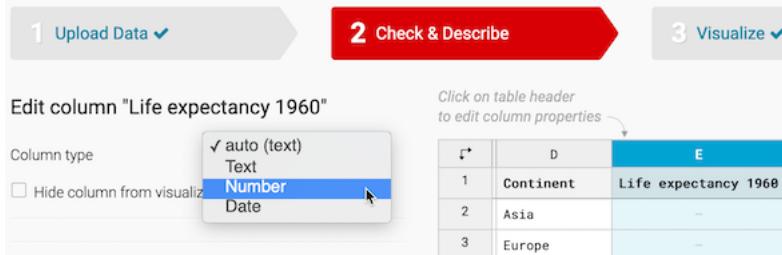


Figure 9.6: Go back to *Check & Describe* to change the properties of column E from textual to numerical data.

11. To create the sparklines, in the *Visualize* screen under *Customize columns*, select *all* of the columns from *Life expectancy 1960* down to *2018*. To select all at once, click on one column, then scroll down and shift-click on the next-to-last column. Then scroll down the page and click the *Show selected columns as tiny chart* button, as shown in Figure 9.7. These steps will create the sparklines in the column and automatically rename it to *Life expectancy 1960–2018*, as shown in Figure 9.8.

Tip: By design, we initially named this column *Life expectancy 1960*, because when we selected several columns to create sparklines, the tool added *-2018* to the end of the new column name.

12. Let's add one more visual element: a bar chart to visually represent the *Difference* column in the table. In the *Visualize* screen under *Customize columns*, select *Difference*. Then scroll down and push the slider to select *Show as bar chart*, as shown in Figure 9.8. Also, select a different bar color, such as black, to distinguish it from the continent colors.

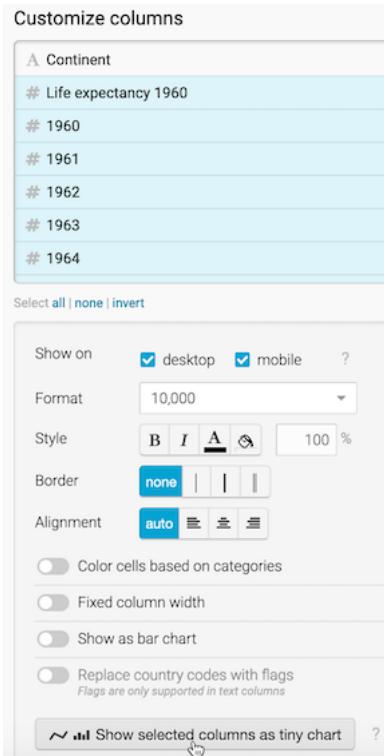


Figure 9.7: Shift-click to select all columns from *Life expectancy 1960–2018* down to *2018*, then click on *Show selected columns as tiny chart*.

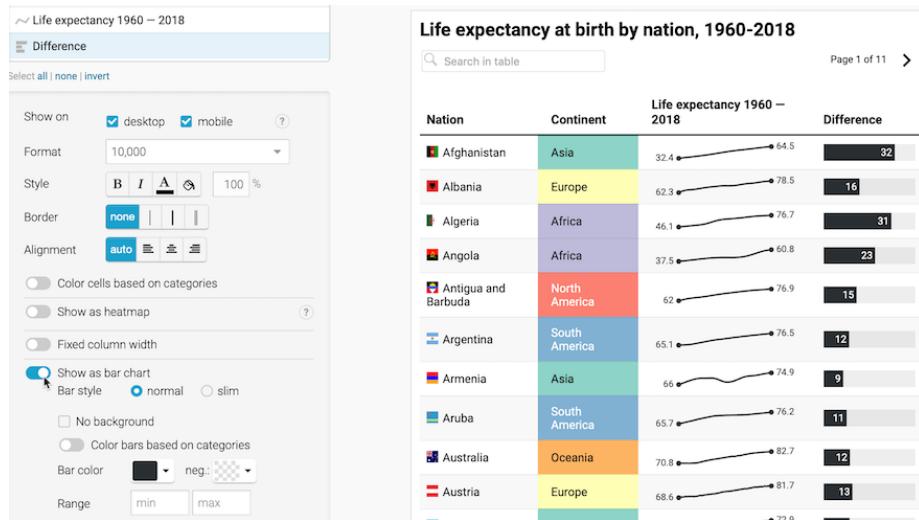


Figure 9.8: Select the *Difference* column and *Show as bar chart*.

13. In the *Visualize* screen, click the *Annotate* tab to add a title, data source, and byline.
14. Click on *Publish & Embed* to share the link to your interactive table, as previously shown in Figure 9.3. If you logged into your free Datawrapper account, your work is automatically saved online in the *My Charts* menu in the top-right corner of the screen. Also, you can click the blue *Publish* button to generate the code to embed your interactive chart on your website, as you'll learn about in Chapter 10: Embed on the Web. In addition, you can *add your chart to River* if you wish to share your work more widely by allowing other Datawrapper users to adapt and reuse your chart. Furthermore, scroll all the way down and click the *Download PNG* button to export a static image of your chart. Additional exporting and publishing options require a paid Datawrapper account. Or, if you prefer not to create an account, you can enter your email to receive the embed code.

To learn more, we highly recommend the Datawrapper Academy support pages, the extensive gallery of examples, and well-designed training materials.

Summary

In this chapter, we reviewed principles about table design, and how to create an interactive table with sparklines using Datawrapper. In the next chapter, you'll learn how to embed interactive charts, maps, and tables on your website so that readers can explore your data and engage with your stories.

Chapter 10

Embed On the Web

So far you've learned how to create charts in chapter 7 and maps in chapter 8. Our book emphasizes the benefits of designing *interactive* visualizations that engage broad audiences on the internet by inviting them to interact with your data, investigate new patterns, download files if desired, and easily share your work on social media. In this chapter, you'll learn about a computer code tag called an *iframe*. Like a picture frame, an iframe displays one web page (such as your interactive data visualization) inside a second web page that you control (such as your personal or organizational web site), and makes the content appear seamlessly so that audiences can still interact with it. Several of the visualization tools you've learned so far, such as Google Sheets, Datawrapper, and Tableau Public, generate an *embed code* that contains an iframe to the online chart or map you've created on their platform. We will demonstrate how to get the embed code or link from your visualization tool site, and paste the code into a second website to seamlessly display your interactive content. No coding skills are required in this introductory book, but it certainly helps to be *code-curious*.

TODO: ILYA Discuss whether we should add an abstract diagram of an iframe inside a web page, like a picture frame on a wall, to visually reinforce this concept for new users here

Static Image vs Interactive iframe

First, let's clarify the difference between *static* versus *interactive* visualizations. A static picture of a chart or map is a frozen image. Many visualization tools allow you to download static images of your charts or maps in .JPG or .PNG or .PDF format. Static images are useful when that's all that you want to insert in a document, a presentation slide, or even a web page. Another option is to paste a static image, and add a link or custom shortlink with the web address to an interactive chart or map, and invite audiences to explore it online.

Also, you capture a static image of any web page on your computer by taking a screenshot with these built-in commands:

- Chromebook: Press key combination *Ctrl + Shift + F5* for rectangular select tool.
- Macintosh: Press key combination *Shift + Command + 4* then click-and-drag the cross-hair capture tool.
- Windows: Press key combination *Windows key + Shift + S* to call up the *Snip & Sketch* tool.

An animated GIF file is a series of static images that captures motion on the screen. You can insert an animated GIF on a web page to illustrate a short sequence of steps while using an interactive visualization, but audiences cannot interact with it, other than to play the animated loop over again. Paid software tools such as Snagit allow you to create screenshots including drop-down menus and cursors, animated GIFs, and more.

By contrast, *interactive* visualizations allow audiences to directly engage with your data story through their web browsers. Visitors usually can float their cursor over a chart to view tooltips or underlying information, or zoom into a map and pan around, or search terms or sort columns in an interactive table. Interactive visualizations are usually hosted online, such as a chart or map tool platform, and are primarily designed to be viewed online, though in some cases it's possible for you to download and interact with them on your local computer.

Now let's turn to the central question: how can we make an interactive visualization, which resides on its online host (the primary site), appear seamlessly on a different website that we control (the secondary site)? While it's possible to insert a link on our secondary site to the charts or maps on the primary site, that's inconvenient for our audiences because it requires them to click away from the web page they were reading. A better solution is to insert an embed code that usually contains an `iframe` tag, written in Hypertext Markup Language (HTML), the code that displays content inside our web browsers. While you don't need any coding experience, you'll benefit in the long run by learning how to recognize the core features of an embed code and how it works.

In its simplest form, an `iframe` instructs the secondary site to display a web page address from the primary site, known as the source, as if it were a seamless picture frame on the wall of a room. The sample `iframe` code below begins with a start tag `<iframe ...>`, which contains the source `src='https://...'` with either single- or double-quotes around the primary site URL, then concludes with an end tag `</iframe>`. This sample `iframe` refers to an interactive US income inequality chart on the Datawrapper platform, which first appeared in the Introduction to this book, as shown in Figure 10.1.

```
<iframe src='https://https://datawrapper.dwcdn.net/LtRbj/'></iframe>
```

Sometimes embed codes or their `iframe` tags are *much longer* than the simple example above. For example, an `iframe` tag might include other attributes, such

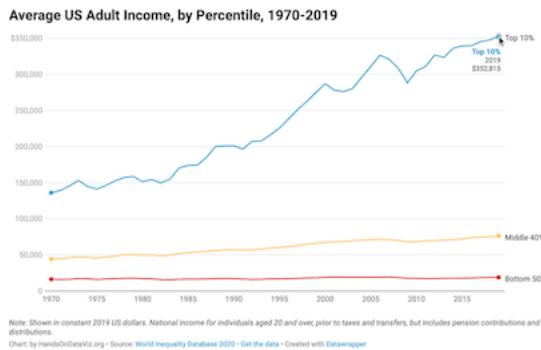


Figure 10.1: Depending on the format of your book, if a static chart appears above, but you can also view the interactive version.

as `width` or `height`, measured in pixels (`px`) or a percentage of its dimensions on the secondary site. Also, you may see other iframe tag elements, such as `seamless` or `frameborder="0"` or `scrolling="no"`, which create a seamless appearance between the iframe content and its surroundings. Finally, you may see *really long* embed codes that contain a dozen or more lines of code that even we don't fully understand. That's okay, because all of these are optional addons to improve the appearance of the iframe in the secondary site. The most essential ingredient of an embed code is the iframe and its three core parts: the iframe start tag, source web address, and end tag. When in doubt, look for those key ingredients.

Now that you have a clearer definition of an interactive visualization, embed codes, and iframe tags, in the next section we'll learn how to copy the embed code from different visualization platforms.

TODO above: confirm the screenshot commands for non-Mac platforms and if they are the simplest versions, and if older Windows commands are still needed

Get the Embed Code or iframe Tag

In this section, you'll learn how to copy the embed code or iframe tag that is automatically generated when you publish a chart or map on different visualization platforms featured in this book. Remember that embed codes contain the essential iframe tag, along with other bits of code to display the chart or map from the primary site and make it appear seamlessly on the secondary site.

We'll break this down into three steps for each visualization platform. First, we will demonstrate how to copy your embed code or iframe tag from Google Sheets, Datawrapper, Tableau Public, and other platforms listed below. [TODO: add others like BatchGeo or rewrite that sentence?] Second, we'll show you how to

practice paste the embed code or iframe tag into the W3Schools TryIt iframe page, as shown in Figure 10.2, to help you understand what's happening behind-the-scenes and to modify the code if needed. Third, we'll point you to the next section to learn how to properly paste the embed code in your preferred website, such as WordPress [and others TODO].



Figure 10.2: For each embed code below, you will *practice paste* it in place of the selected text of the W3Schools TryIt iframe page to see how it works.

from Google Sheets

1. After you create a Google Sheets chart in Chapter 7, click the 3-dot kebab menu in the upper-right corner of the chart to publish it, as shown in Figure 10.3.

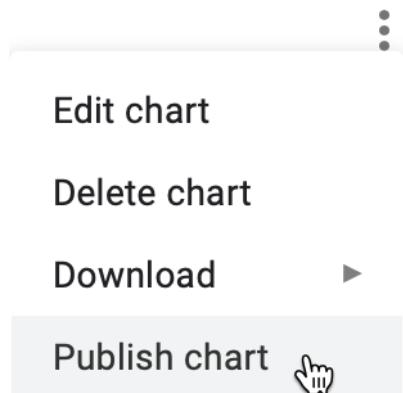


Figure 10.3: In your chart, click the three-dot kebab menu to publish it.

2. In the next screen, select the *Embed* tab and *Interactive* chart, and click the *Publish* button to share it online. Select and copy the embed code, as shown in Figure 10.4.

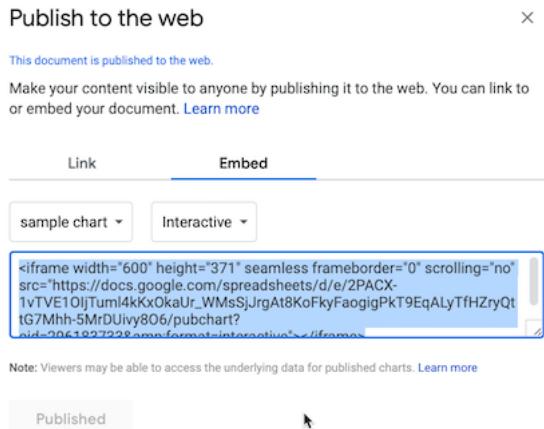


Figure 10.4: Click *Embed* and *Interactive* and *Publish*, then select and copy the embed code.

3. To better understand how the embed code works, open the W3Schools TryIt iframe page. Select the current iframe tag, paste in your embed code to replace it, and press the green *Run* button. The result should be similar to Figure 10.5, but instead will display your embed code and interactive visualization.



Figure 10.5: Paste your Google Sheets embed code to place of the current iframe tag in the TryIt page and click *Run*.

At first glance, the Google Sheets embed code may appear long, but it's actually a straightforward iframe tag with a long source link. Look closely and you'll see iframe settings such as `width` and `height` (measured here in pixels), and `seamless` and `frameborder='0'` and `scrolling='no'` to improve its appearance.

4. Now jump to the paste code to website section of this chapter to learn how to properly insert your embed code into your preferred platform.

from Datawrapper

1. After you create a Datawrapper chart in Chapter 7 or map in Chapter 8 or interactive table in Chapter 9, proceed to the final screen and click the *Publish* button, as shown in Figure 10.6. This publishes the interactive version of your chart or map online. Further down on the same screen you can also export a static image, if desired.

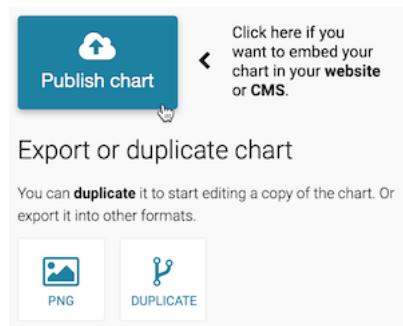


Figure 10.6: Proceed to the final screen and click the *Publish* button.

2. On the next screen, click *copy* to get the Datawrapper embed code, as shown in Figure 10.7. The default *responsive iframe* version of the embed code contains additional instructions to improve its appearance on both small and large device screens.
3. To better understand how the embed code works, open the W3Schools TryIt iframe page. Select the current iframe tag, paste in your embed code to replace it, and press the green *Run* button. The result should be similar to Figure 10.8, but instead will display your embed code and interactive visualization.

The Datawrapper embed code is *long*, but if you look closely, the first half contains a relatively straightforward iframe tag that includes familiar-looking attributes such `src`, `scrolling`, and `frameborder`, and `width` and `height` inside a style tag. The second half of the embed code contains JavaScript instructions to make the iframe appear responsive depending on the size of the device screen.

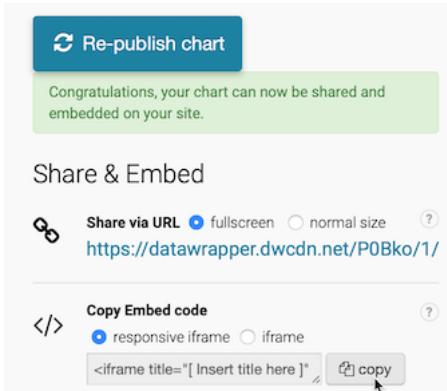


Figure 10.7: Copy the responsive iframe version of the Datawrapper embed code.

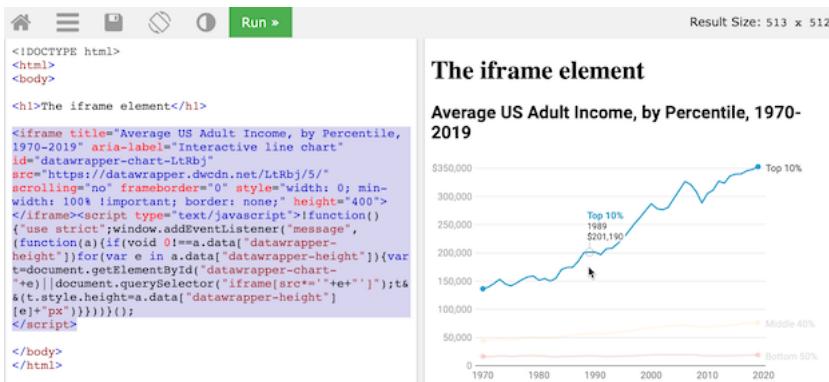


Figure 10.8: Paste your Datawrapper embed code in place of the current iframe tag in the TryIt page and click *Run*.

4. Always try to paste the *full embed code* in your desired web platform. Jump to the paste code to website section of this chapter to learn how to properly insert your embed code into common websites.
5. But if it doesn't work, go back to step 3 and experiment. Try to edit the embed code down to a *simple iframe*, and run it again to see how it looks, as shown in Figure 10.9. Sometimes a simple iframe works better on your website than a complex embed code.



Figure 10.9: If a complex embed code does not work in your website, go back and try to edit it down into a simple iframe.

Hint: The Datawrapper iframe tag source follows this general format: <https://datawrapper.dwcdn.net/abcdef/1/>, where the 1 refers to the first version of the chart or map you published. If you make edits and re-publish your visualization, Datawrapper will increase the last digit (to 2 and so forth), and *automatically redirect* older links to the *current version*, which keeps your work up-to-date for your audience.

from Tableau Public

1. After you create a Tableau Public chart in Chapter 7 or map in Chapter 8, publish your worksheet, dashboard, or story online by selecting *File > Save to Tableau Public* in the desktop application menu, as shown in Figure 10.10.
2. In your online Tableau Public account profile page, click to *View* the details of any of your published visualizations, as shown in Figure 10.11.

Tip: All of your published visualizations appear under your username account profile on the Tableau Public server. If you don't recall your username, search the Tableau Public server for your first and last name that you entered when creating your online account.

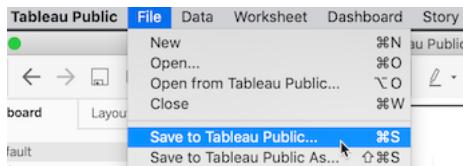


Figure 10.10: In the Tableau Public desktop application, select *File*–*Save to Tableau Public* to publish to the online server.

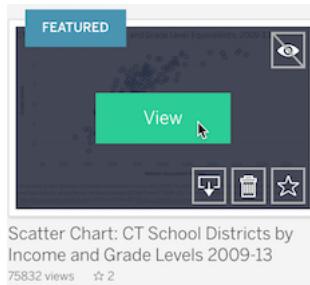


Figure 10.11: In your Tableau Public online profile page, click to *View* the details of a published visualization.

3. When viewing details for a published visualization in your Tableau Public online account, scroll down and click on the *Share* symbol in the lower-right corner. Select and copy its embed code, as shown in Figure 10.12.

4. To better understand how the embed code works, open the W3Schools TryIt iframe page. Select the current iframe tag, paste in your embed code to replace it, and press the green *Run* button. The result should be similar to Figure 10.13, but instead will display your embed code and interactive visualization.

The Tableau Public embed code is *very long* and does not fit in one image. Even we don't fully understand what's happening in this complex batch of code.

5. Always try to paste the *full embed code* in your desired web platform. Jump to the paste code to website section of this chapter to learn how to properly insert on different common websites.

6. But if it doesn't work, go back to step 3 and 4 and experiment. For example, you can copy the Tableau Public link to your visualization, *instead* of the embed code, try to convert it into a simple iframe tag, and run it again to see how it looks.



Figure 10.12: Scroll down in the online published visualization details, click the *Share* button, and copy the embed code.

The screenshot shows a 'TryIt' page with an 'iframe' element highlighted. The code within the iframe is a Tableau visualization's embed code. To the right of the iframe, the visualization itself is displayed, titled 'The iframe element'. It is a scatter plot titled 'CT School Districts by Income and Grade Level Equivalents, 2009-13'. The x-axis is labeled 'Median Household Income' and the y-axis is labeled 'Grade Levels'. A data point for 'Chaplin School District' is highlighted, showing values of 0.100 for Grade Levels and 75,368 for Median Household Income. The page also includes a 'Run' button at the top and a 'Result Size: 513 x 512' message.

Figure 10.13: Paste your Tableau public embed code in place of the current *iframe* tag in the TryIt page and click *Run*.

Here's some hints from the Tableau Public support page when trying to create iframes from links like this example:

<https://public.tableau.com/views/CTSchoolDistrictsbyIncomeandGradeLevels2009-13/Sheet1?:language=1>

6. Paste the link into the W3Schools TryIt page, and delete all of the code that appears *after* the question mark (?), so that it looks like this:

<https://public.tableau.com/views/CTSchoolDistrictsbyIncomeandGradeLevels2009-13/Sheet1?>

7. At the end, attach this code snippet to replace what you deleted above:

```
:showVizHome=no&:embed=true
```

8. Now your edited link should look like this:

<https://public.tableau.com/views/CTSchoolDistrictsbyIncomeandGradeLevels2009-13/Sheet1?:showVizHome=no&:embed=true>

9. Enclose your edit link inside an iframe source tag **src=** with quotes, to make it look similar to this:

`src="https://public.tableau.com/views/CTSchoolDistrictsbyIncomeandGradeLevels2009-13/Sheet1?:showVizHome=no&:embed=true"`

10. Add iframe start and end tags, and also attributes for **width**, **height**, **seamless**, **frameborder="0"**, and **scrolling="no"**, to make it look similar to this:

```
<iframe src="https://public.tableau.com/views/CTSchoolDistrictsbyIncomeandGradeLevels2009-13/Sheet1?:showVizHome=no&:embed=true" width="90%" height="500" seamless frameborder="0" scrolling="no"></iframe>
```

Hint: Insert **width="90%"**, rather than **100%**, to help readers to scroll more easily down your web page with a margin.

11. Press *Run* to see how it looks, as shown in Figure 10.14. Sometimes a simple iframe works better on your website than a complex embed code.

Now that you have a better sense of how to copy embed codes, and edit them down to simple iframes if needed, in the next section you'll learn how to paste them into common websites such as WordPress.

Paste Code or iframe to Website

In the prior section, you copied the embed code (or created an iframe) for your interactive visualization that is hosted online by the primary site. In this section, we'll demonstrate ways to properly paste the embed code or iframe to seamlessly display your interactive chart or map in the secondary site.

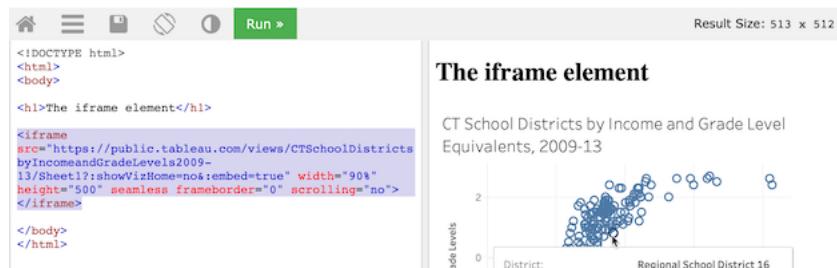


Figure 10.14: If a complex embed code does not work in your website, go back and copy the link to the visualization, and try to convert it into a simple iframe.

to WordPress.com sites

If you own *free* or personal or premium WordPress.com site, with a web address in the format `anyone.wordpress.com`, you *cannot* insert an embed code that contains an iframe or Javascript due to security concerns, as described on their support page. This means that if wish to show data visualizations created from this book on a WordPress.com site, you have two options. First, with your free or personal or premium plan, you can still insert a *static image* of a chart or map and a link to its interactive site, but that's clearly not ideal. Second, WordPress.com suggests that you can update to their paid Business or eCommerce plan, which supports embed codes that contain iframes or Javascript, following instructions similar to the self-hosted WordPress sites below.

to Self-hosted WordPress sites

Make sure you understand the difference between a WordPress.com site above versus a self-hosted WordPress site. The latter is sometimes called WordPress.org site because anyone can freely download the software from that address and *host it on their own webserver*, or more commonly, have access to a self-hosted WordPress server through their school or work, or by renting space on a vendor's webserver. But the web address of a self-hosted WordPress site does *not* necessarily need to end in `.org`. It also could be `.com` or `.edu` or any other ending, so don't let that confuse you.

There are several ways to insert an embed code or iframe in a self-hosted WordPress site, but your success may depend on your WP version, your access level, and the complexity of the code.

For the first method, assume that you're using WordPress 5.0 or above with the newer block editor, and you have editor or administrator access to your site. TODO: Check whether less-than-admin privileges strips out iframe codes, as it used to before WP 5.0

In your block editor, select a *custom HTML* block, and directly insert the embed code or the iframe, as shown in Figure 10.15.



Figure 10.15: Paste an embed code or iframe into a custom HTML block.

For the second method, assume that you're using WordPress with either the classic or block editor, and that you have administrator access to the site.

1. Install and activate the iframe plugin, as shown in Figure 10.16. This plugin allows authors to embed iframe codes in a modified “shortcode” format surrounded by square brackets. Without the plugin, self-hosted WordPress.org sites will usually “strip out” iframe codes for all users except the site administrator. TODO: update this image



Figure 10.16: Install and activate the *iframe plugin* on a self-hosted WP site.

2. In the WordPress editor, click to the *text* tab to view the code, and paste the embed code or iframe, as shown in Figure 10.17. Initially, the code you pasted includes iframe tags at the start (`<iframe...`) and the end (`...></iframe>`).
3. Modify the start tag by replacing the less-than symbol (`<`) with a square opening bracket (`[`). Modify the back end by erasing the greater-than symbol (`>`) and the entire end tag (`</iframe>`), and replace both of them with one square closing bracket (`]`), as shown in Figure 10.18.

TODO ABOVE: check method with more complex embed codes, check with non-admin privileges, and update images

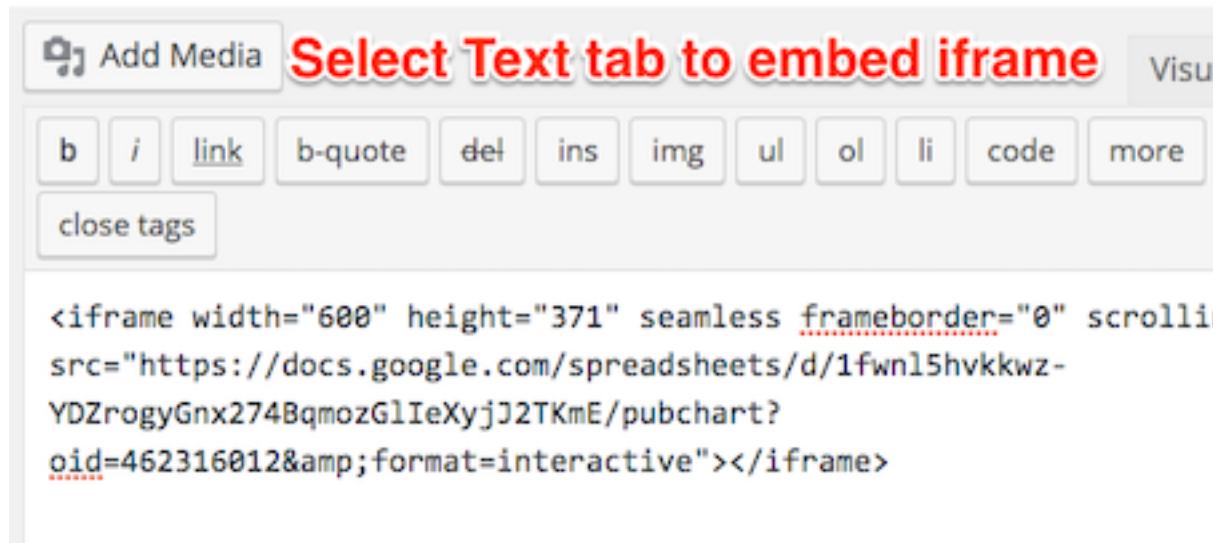


Figure 10.17: Install and activate the *iframe* plugin on a self-hosted WP site.

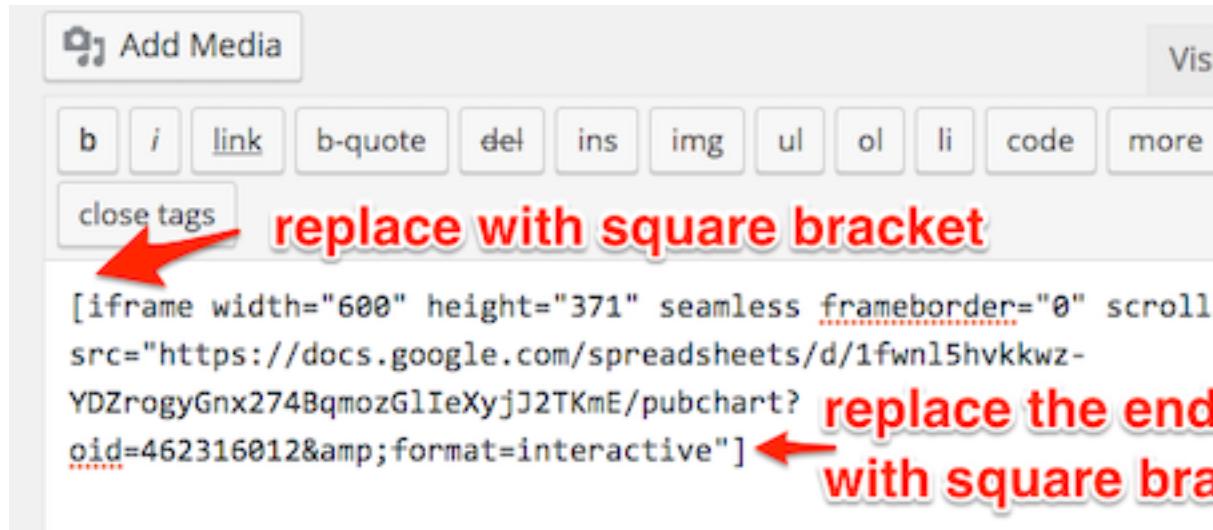


Figure 10.18: Modify the front and back end with square brackets.

to SquareSpace, Wix, or Weebly Sites

TODO

Summary

TODO

Chapter 11

Edit and Host Code with GitHub

In the first half of this book, you created interactive charts and maps on free drag-and-drop tool platforms created by companies such as Google and Tableau. These platforms are great for beginners, but their pre-set tools limit your options for designing and customizing your visualizations, and they also require you to depend upon their web servers and terms of service to host your data and work products. If these companies change their tools or terms, you have little choice in the matter, other than deleting your account and switching services, which means that your online charts and maps would appear to audiences as dead links.

In the second half of this book, get ready to make a big leap—and we'll help you through every step—by learning how to copy, edit, and host code templates. These templates are pre-written software instructions that allow you to upload your data, customize its appearance, and display your interactive charts and maps on a web site that you control. No prior coding experience is required, but it helps if you're *code-curious* and willing to experiment with your computer.

Code templates are similar to cookbook recipes. Imagine you're in your kitchen, looking at our favorite recipe we've publicly shared to make brownies (yum!), which begins with these three steps: `Melt butter`, `Add sugar`, `Mix in cocoa`. Recipes are templates, meaning that you can follow them precisely, or modify them to suit your tastes. Imagine that you copy our recipe (or “fork” it, as coders say) and insert a new step: `Add walnuts`. If you also publicly share your recipe, now there will be two versions of instructions, to suit both those who strongly prefer or dislike nuts in their brownies. (We do not take sides in this deeply polarizing dispute.)

Currently, the most popular cookbook among coders is GitHub, with more than 40 million users and over 100 million recipes (or “code repositories” or “repos”).

You can sign up for a free account and choose to make your repos private (like Grandma’s secret recipes) or public (like the ones we share below). Since GitHub was designed to be public, think twice before uploading any confidential or sensitive information that should not be shared with others. GitHub encourages sharing *open-source code*, meaning the creator grants permission for others to freely distribute and modify it, based on the conditions of the type of license they have selected.

When you create a brand-new repo, GitHub invites you to Choose a License. Two of the most popular open-source software licenses are the MIT License, which is very permissive, and the GNU General Public License version 3, which mandates that any modifications be shared under the same license. The latter version is often described as a *copyleft* license that requires any derivatives of the original code to remain publicly accessible, in contrast to traditional *copyright* that favors private ownership. When you fork a copy of someone’s open-source code on GitHub, look at the type of license they’ve chosen (if any), keep it in your version, and respect its terms.

To be clear, the GitHub platform is also owned by a large company (Microsoft purchased it in 2018), and when using it to share or host code, you’re also dependent on its tools and terms. But the magic of code templates is that you can migrate and host your work anywhere on the web. You could move to a competing repository-hosting service such as GitLab, or purchase your own domain name and server space through one of many web hosting services. Or you can choose a hybrid option, such as hosting your code on GitHub and choosing its custom domain option, to display it under a domain name that you’ve purchased, just like the web version of this book is hosted on GitHub under our domain name, <https://HandsOnDataViz.org>. If we choose to move the code away from GitHub, we have the option to repoint our domain to a different web host.

In the next section of this chapter, we will introduce basic steps to copy, edit, and host a simple Leaflet map code template on GitHub. When you publish any chart or map code template by hosting it on GitHub Pages, you can easily transform its online link into an iframe that you can embed on a secondary website, which we discussed in Chapter 10. Later you’ll learn how to create a new GitHub repo and upload code files.

This chapter introduces GitHub using its web browser interface, which works best for beginners. Later you’ll learn about intermediate-level tools, such as GitHub Desktop and Atom Editor, to work more efficiently with code repos on your personal computer.

If problems arise, turn to the Fix Common Mistakes section in the appendix. All of us make mistakes and accidentally “break our code” from time to time, and it’s a great way to learn how things work—and what to do when it doesn’t work!

Copy, Edit, and Host a Simple Leaflet Map Template

Now that you understand how GitHub code repositories are like a public cookbook of recipes, which anyone can copy and modify, let's get into the kitchen and start baking! In this section, we'll introduce you to a very simple code template based on Leaflet, an open-source code library for creating interactive maps that are very popular in journalism, business, government, and higher education. Many people choose Leaflet because the code is freely available to everyone, relatively easy to use, and has an active community of supporters who regularly update it. But unlike drag-and-drop tools that we previously covered in Chapter 8: Map Your Data, working with our Leaflet templates requires you to copy and edit a few lines of code before hosting it on the web. While no prior coding experience is necessary, it's helpful to know that these code templates are based on the three core languages that communicate with browsers: HyperText Markup Language (HTML), Cascading Style Sheets (CSS), and JavaScript. Furthermore, we can edit these code templates using the GitHub web interface, which means you can do this on any type of computer (Mac, Windows, Chromebook, etc.) with any modern web browser.

Here's an overview of the key steps you'll learn about GitHub in this section:

- Make a copy of our simple Leaflet map code template
- Edit the map title, start position, background layer, and marker
- Host a live online version of your modified map code on the public web

Your goal is to create your own version of this simple interactive map, with your edits, as shown in Figure 11.1.

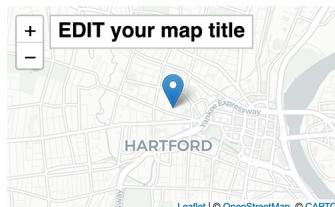


Figure 11.1: Create your own version of this simple interactive Leaflet map.

1. Create your own free account on GitHub. It may ask you to do a simple quiz to prove you're a human! If you don't see a confirmation message in your email, check your spam folder.

Tip: Choose a GitHub username that's relatively short, and one that you'll be happy seeing in the web address of charts and maps you'll publish online.

In other words, DrunkBrownieChef6789 may not be the wisest choice for a username, if BrownieChef is also available.

2. After you log into your GitHub account in your browser, go to our simple Leaflet map template at <https://github.com/HandsOnDataViz/leaflet-map-simple>
3. Click the green *Use this template* button to make your own copy of our repo, as shown in Figure 11.2.

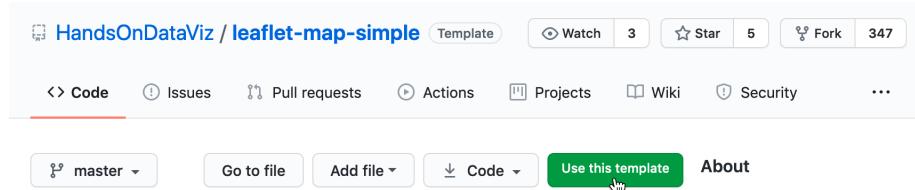


Figure 11.2: Click *Use this template* to make your own copy.

4. On the next screen, your account will appear as the owner. Name your copy of the repo `leaflet-map-simple`, the same as ours, as shown in Figure 11.3. Click the green *Create repository from template* button.

Note: We set up our repo using GitHub's template feature to make it easier for users to create their own copies. If you're trying to copy someone else's GitHub repo and don't see a *Template* button, then click the *Fork* button, which makes a copy a different way. Here's the difference: *Template* allows you to make *multiple* copies of the same repo by giving them different names, while *Fork* allows you to create *only one copy* of a repo because it uses the same name as the original, and GitHub prevents you from creating two repos with the same name. If you need to create a second fork of a GitHub repo, go to the Create a New Repo and Upload Files on GitHub section of this chapter.

The upper-left corner of the next screen will say `USERNAME/leaflet-map-simple` generated from `HandsOnDataViz/leaflet-map-simple`, where `USERNAME` refers to your GitHub account username. This confirms that you copied our template into your GitHub account, and it contains only three files:

- `LICENSE` shows that we've selected the MIT License, which allows anyone to copy and modify the code as they wish.
- `README.md` provides a simple description and link to the live demo, which we'll come back to later.
- `index.html` is the key file in this particular, because it contains the map code.

Create a new repository from leaflet-map-simple

The new repository will start with the same files and folders as [HandsOnDataViz/leaflet-map-simple](#).

Owner * Repository name *

/ 

Great repository names are short and memorable. Need inspiration? How about [ubiquitous-disco](#)?

Description (optional)

 **Public**
Anyone on the internet can see this repository. You choose who can commit.

 **Private**
You choose who can see and commit to this repository.

Include all branches
Copy all branches from HandsOnDataViz/leaflet-map-simple and not just master.

Create repository from template

Figure 11.3: Name your copied repo `leaflet-map-simple`.

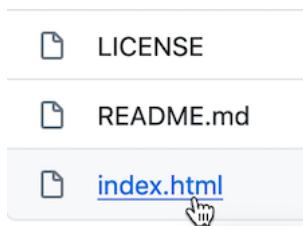


Figure 11.4: Click the `index.html` file to view the code.

5. Click on the `index.html` file to view the code, as shown in Figure 11.4.

If this is the first time you’re looking at computer code, it may feel overwhelming, but relax! We’ve inserted several “code comments” to explain what’s happening. The first block tells web browsers the formatting to read the rest of the page of code. The second block instructs the browser to load the Leaflet code library, the open-source software that constructs the interactive map. The third block describes where the map and title should be positioned on the screen. The good news is that you don’t need to touch any of those blocks of code, so leave them as-is. But you do want to modify a few lines further below.

6. To edit the code, click on the the pencil symbol in the upper-right corner, as shown in Figure 11.5.

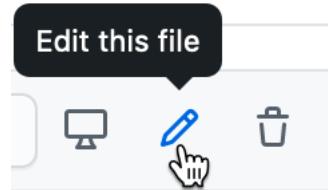


Figure 11.5: Click the pencil button to edit the code.

Let’s start by making one simple change to prove to everyone that you’re now editing *your* map, by modifying the map title, which appears in the HTML division tag block around lines 21-23.

7. In this line `<div id="map-title">EDIT your map title</div>`, type your new map title in place of the words `EDIT your map title`. Be careful not to erase the HTML tags that appear on both ends inside the `< >` symbols.
8. To save your edit, scroll to the bottom of the page and click the green *Commit Changes* button, as shown in Figure 11.6.

In the language of coders, we “commit” our changes in the same way that most people “save” a document, and later you’ll see how GitHub tracks each code commit so that you can roll them back if needed. By default, GitHub inserts a short description of your commit as “Update index.html”, and you have the option to customize that description when you start making lots of commits to keep track of your work. Also, GitHub commits your changes directly to the default branch of your code, which we’ll explain later.

Now let’s publish your edited map to the public web to see how it looks in a web browser. GitHub not only stores open-source code, but its built-in GitHub Pages

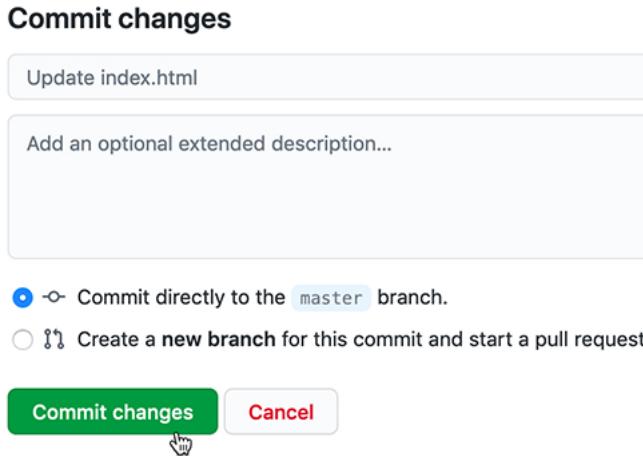


Figure 11.6: Click the green *Commit Changes* button to save your edits.

feature allows you to host a live online version of your HTML-based code, which anyone with the web address can view in their browser. While GitHub Pages is free to use, there are some restrictions on usage, file size, and content and it is not intended for running an online business or commercial transactions. But one advantage of code templates is that you can host them on any web server you control. Since we're already using GitHub to store and edit our code template, it's easy to turn on GitHub Pages to host it online.

Tip: If you wish to store your code on GitHub but need to scale up to a larger commercial-level web host, see freemium services such as Netlify, which automatically detects any changes you push to your GitHub repository, then deploys them to your online site.

9. To access GitHub Pages, scroll to the top of your repo page and click the *Settings* button as shown in Figure 11.7.

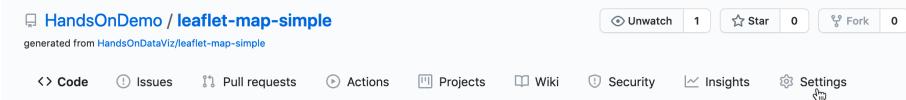


Figure 11.7: Click the *Settings* button to access GitHub Pages and publish your work on the web.

10. In the Settings screen, scroll way down to the GitHub Pages area. In the drop-down menu, change *Source* from *None* to *Main*, keep the default

/(root) option in the middle, and press *Save* as shown in Figure 11.8. This step tells GitHub to publish a live version of your map on the public web, where anyone can access it in their browser, if they have the web address.

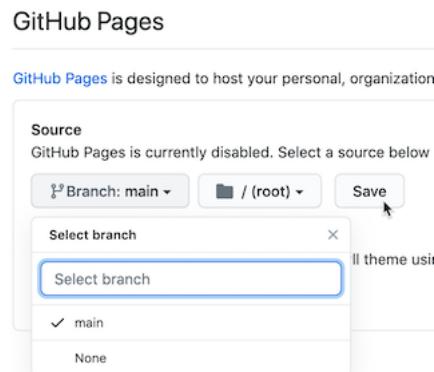


Figure 11.8: In *Settings*, go to *GitHub Pages*, and switch the source from *None* to *Main*.

Tip: In response to the Black Lives Matter movement in 2020, GitHub renamed its default branch from *master* to *main* to eliminate its master-slave metaphor.

11. Scroll back down to *Settings > GitHub Pages* to see the web address where your live map has been published online, and right-click it to open in a new browser tab, as shown in Figure 11.9.

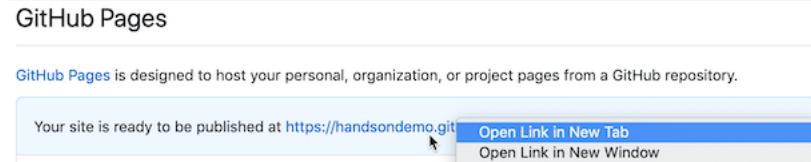


Figure 11.9: In *Settings* for *GitHub Pages*, right-click your published map link to open in a new tab.

Now you should have at least two tabs open in your browser. The first tab contains your GitHub repo, where you edit your code, with a web address in this format, and replace **USERNAME** and **REPOSITORY** with your own:

```
https://github.com/USERNAME/REPOSITORY
```

The second tab contains your GitHub Pages live website, where your edited code appears online. GitHub Pages automatically generates a public web address in this format:

<https://USERNAME.github.io/REPOSITORY>

Remember how we told you not to create your account with a username like DrunkBrownieChef6789? GitHub automatically places your username automatically in the public web address.

Keep both tabs open so you can easily go back and forth between editing your code and viewing the live results online.

Note: The live version of your code points to the `index.html` page by default, so it's not necessary to include it in the web address. Also, web addresses are *not* case sensitive, meaning you can save time by typing all of it in lower-case.

Tip: If your live map does *not* appear right away, wait up to 30 seconds for GitHub Pages to finish processing your edits. Then give your browser a “hard refresh” to bypass any saved content in your cache and re-download the entire web page from the server, using one of these key combinations:

- Ctrl + F5 (most browsers for Windows or Linux)
- Command + Shift + R (Chrome or Firefox for Mac)
- Shift + Reload button toolbar (Safari for Mac)
- Ctrl + Shift + Backspace (on Chromebook)

Now let's edit your GitHub repo so that the link points to *your* live map, instead of *our* live map.

12. Copy the web address of your live map from your second browser tab.
13. Go back to your first browser tab with your GitHub repo, and click on the repo title to return to its home page, as shown in Figure 11.10.



Figure 11.10: On your first browser tab, click the repo title.

14. On your repo page, click to open the `README.md` file, and click the pencil again to edit it, as shown in Figure 11.11. Paste your live web link under the label (*replace with link to your site*) and scroll down to commit the change.

The screenshot shows a GitHub repository page for 'HandsOnDemo'. At the top, it displays a commit history with one commit from 'HandsOnDemo' made 1 hour ago. Below the commit history, there are three files listed: 'LICENSE', 'README.md', and 'index.html'. The 'README.md' file is currently selected and open in an editor. The editor shows the following content:

```
leaflet-map-simple
A simple Leaflet map template for new users to fork their own copy, edit, and host on GitHub Pages
Link to live map (replace with link to your site)
https://handsondataviz.github.io/leaflet-map-simple/
```

Figure 11.11: Open and edit the README file to paste the link to your live map.

Now that you've successfully made simple edits and published your live map, let's make more edits to jazz it up and help you learn more about how Leaflet code works.

15. On your repo home page, click to open the `index.html` file, and click the pencil symbol to edit more code.

Wherever you see the `EDIT` code comment, this points out a line that you can easily modify. For example, look for the code block shown below that sets up the initial center point of the map and its zoom level. Insert a new latitude and longitude coordinate to set a new center point. To find coordinates, right-click on any point in Google Maps and select *What's here?*, as described in the geocoding section in Chapter 3.

```
var map = L.map('map', {
  center: [41.77, -72.69], // EDIT latitude, longitude to re-center map
  zoom: 12, // EDIT from 1 to 18 -- decrease to zoom out, increase to zoom in
  scrollWheelZoom: false
});
```

The next code block displays the basemap tile layer that serve as the map background. Our template uses a light map with all labels, publicly provided by CARTO, with credit to OpenStreetMap. One simple edit is to change `light_all` to `dark_all`, which will substitute a different CARTO basemap with inverted coloring. Or see many other Leaflet basemap code options that you can paste in at <https://leaflet-extras.github.io/leaflet-providers/preview/>.

Make sure to attribute the source, and also keep `) .addTo(map);` at the end of this code block, which displays the basemap.

```
L.tileLayer('https://{s}.basemaps.cartocdn.com/light_all/{z}/{x}/{y}{r}.png', {
  attribution: '&copy; <a href="http://www.openstreetmap.org/copyright">OpenStreetMap</a>,
  &copy; <a href="https://carto.com/attribution">CARTO</a>'
}) .addTo(map);
```

The last code block displays a single point marker on the map, colored blue by default in Leaflet, with a pop-up message when users click it. You can edit the marker coordinates, insert the pop-up text, or copy and paste the code block to create a second marker.

```
L.marker([41.77, -72.69]).addTo(map) // EDIT latitude, longitude to re-position marker
.bindPopup("Insert pop-up text here"); // EDIT pop-up text message
```

Warning: Be careful when editing your code. Accidentally removing or adding extra punctuation (such as quotation marks, commas, or semicolons) can stop your map from working. But breaking your code—and fixing it—can also be a great way to learn.

15. After making edits, remember to scroll down and press the *Commit* button to save changes. Then go to your browser tab with the live map, and do a hard-refresh to view changes. If your map edits do not appear right away, remember that GitHub Pages sometimes requires up to 30 seconds to process code edits. If you have problems, see the Fix Common Mistakes section in the appendix.

Congratulations! If this is the first time that you've edited computer code and hosted it online, you can now call yourself a “coder”. The process is similar to following and modifying a cookbook recipe, just like you also can call yourself a “chef” after baking your first batch of brownies! Although no one is likely to hire you as a full-time paid coder (or chef) at this early stage, you now understand several of the basic skills needed to copy, edit, and host code online, and you're ready to dive into the more advanced versions, such as Chart.js and Highcharts templates in Chapter 12 and Leaflet map templates in Chapter 13.

Convert GitHub Pages Link to iframe

In Chapter 10: Embed on the Web, we discussed the benefits of displaying interactive content from a primary site and making it appear seamlessly in a secondary site. You also learned how to convert very long Datawrapper and

Tableau Public embed codes into shorter iframe tags when needed, so that you can embed them more easily on a secondary website.

The same concept applies to GitHub Pages. When you publish a code template for a chart or map (or any content) on GitHub Pages, it generates an online link that you can convert into an iframe tag, using the same principles as above, to embed it on a secondary website. Follow these steps:

1. For any GitHub repository you have published online, go to its *Settings* page and scroll down to copy its GitHub Pages web address, which will appear in this general format:

```
https://USERNAME.github.io/REPOSITORY
```

2. Convert it into an iframe by enclosing the link inside quotation marks as the source, and adding both start and end tags, in this general format:

```
<iframe src="https://USERNAME.github.io/REPOSITORY"></iframe>
```

3. If desired, improve the iframe appearance on the secondary site by adding any of these optional attributes, such as `width` or `height` (measured in pixels by default, or percentages), or `seamless` or `frameborder="0"` or `scrolling="no"`, in this general format:

```
<iframe src="https://USERNAME.github.io/REPOSITORY" width="100%" height="400" seamless frameborder="0" scrolling="no"></iframe>
```

Hint: Either single-quote ('') marks (also called an apostrophe) or double-quote ("") marks are acceptable in your iframe code, but be consistent and avoid accidentally pasting in curly-quotes.

Now you are ready to paste your iframe into your preferred website, using methods described in Chapter 10, to display your interactive chart or map template from published repository using GitHub Pages.

Now you should have a better sense of how to edit and host code repositories on GitHub. The next section describes how to enhance your GitHub skills by creating new repos and uploading your files. These are essential steps to create a second copy of a code template or to work with more advanced templates in the next two chapters.

Create a New Repo and Upload Files on GitHub

Now that you've made a copy of our GitHub template, the next step is to learn how to create a brand-new repo and upload files. These skills will be

helpful for several scenarios. First, if you have to fork a repo, which GitHub allows you to do only one time, this method will allow you to create additional copies. Second, you'll need to upload some of your own files when creating data visualizations using Chart.js and Highcharts templates in Chapter 12 and Leaflet map templates in Chapter 13. Once again, we'll demonstrate how to do all of these steps in GitHub's beginner-level browser interface, but see the next section on GitHub Desktop for an intermediate-level interface that's more efficient for working with code templates.

In the previous section, you created a copy of our GitHub repo with the *Use this template* button, and we intentionally set up our repos with this newer feature because it allows the user to make *multiple* copies and assign each one a different name. Many other GitHub repos do not include a *Template* button, so to copy those you'll need to click the *Fork* button, which automatically generates a copy with the same repo name as the original. But what if you wish to fork someone's repo a second time? GitHub prevents you from creating a second fork to avoid violating one of its important rules: every repo in your account must have a unique name, to avoid overwriting and erasing your work.

So how do you make a second fork of a GitHub repo, if there's no *Use this template* button? Follow our recommended workaround that's summarized in these three steps:

- Download the existing GitHub repo to your local computer
 - Create a brand-new GitHub repo with a new name
 - Upload the existing code repo files to your brand-new repo
1. Click on the *Code > Download Zip* drop-down menu button on any repo, as shown in Figure 11.12. Your browser will download a zipped compressed folder with the contents of the repo to your local computer, and it may ask you where you wish to save it. Decide on a location and click OK.
 2. Navigate to the location on your computer where you saved the folder. Its file name should end with `.zip`, which means you need to double-click to “unzip” or de-compress the folder. After you unzip it, a new folder will appear named in this format, REPOSITORY-BRANCH, which refers to the repository name (such as `leaflet-map-simple`) and the branch name (such as `main`), and it will contain the repo files. One of those files is named `index.html`, which you'll use in a few steps below.
 3. Go back to your GitHub account in your web browser, click on the plus (+) symbol in the upper-right corner of your account, and select *New repository*, as shown in Figure 11.13.

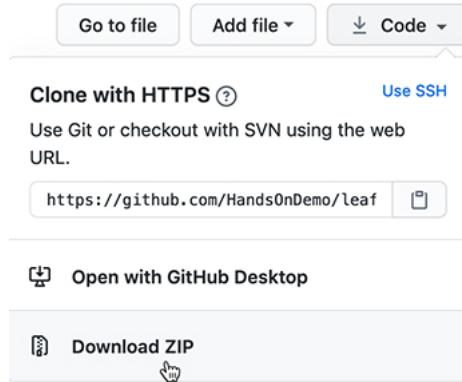


Figure 11.12: Click *Code* and select *Download Zip* to create a compressed folder of a repo on your computer.

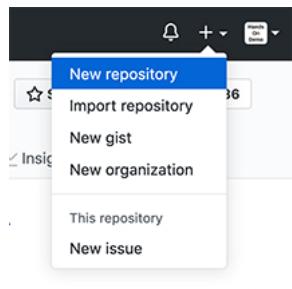


Figure 11.13: Click the plus (+) symbol in upper-right corner to create a new repo.

4. On the next screen, GitHub will ask you to enter a new repo name. Choose a short one, preferably all lower-case, and separate words with hyphens if needed. Let's name it **practice** because we'll delete it at the end of this tutorial.

Check the box to *Initialize this repository with a README* to simplify the next steps.

Also, select *Add a license* that matches the code you plan to upload, which in this case is *MIT License*. Other fields are optional. Click the green *Create Repository* button at the bottom when done, as shown in Figure 11.14.

Create a new repository

A repository contains all project files, including the revision history. / elsewhere? [Import a repository](#).

Repository template
Start your repository with a template repository's contents.

No template ▾

Owner * **Repository name ***

HandsOnDemo / practice ✓

Great repository names are short and memorable. Need inspiration?

Description (optional)

Public
Anyone on the Internet can see this repository. You choose who can

Private
You choose who can see and commit to this repository.

Skip this step if you're importing an existing repository.

Initialize this repository with a README
This will let you immediately clone the repository to your computer.

Add .gitignore: None ▾ Add a license: MIT License ⓘ

Create repository

Figure 11.14: Name your new repo *practice*, check the box to *Initialize this repo with a README*, and *Add a license* (select *MIT*) to match any code you plan to upload.

Your new repo will have a web address similar to <https://github.com/USERNAME/practice>.

5. On your new repo home page, click the *Add File > Upload Files* drop-down menu button, near the middle of the screen, as shown in Figure 11.15.
6. Inside the repo folder that you previously downloaded and unzipped on your local computer, drag-and-drop the `index.html` file to the upload screen of your GitHub repo in your browser, as shown in Figure 11.16. Do not upload `LICENSE` or `README.md` because your new repo already contains those two files. Scroll down to click the green *Commit Changes* button.

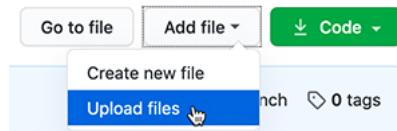


Figure 11.15: Click the *Upload Files* button.

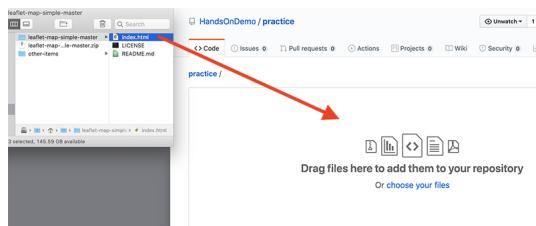


Figure 11.16: Drag-and-drop the `index.html` file to the upload screen.

When the upload is complete, your repo should contain three files, now including a copy of the `index.html` code that you previously downloaded from the `leaflet-map-simple` template. This achieved our goal of working around GitHub's one-fork rule, by creating a new repo and manually uploading a second copy of the code.

Optionally, you could use GitHub Pages to publish a live version of the code online, and paste the links to the live version at the top of your repo and your `README.md` file, as described in the Copy, Edit, and Host a Simple Leaflet Map Template section of this chapter.

7. Since this was only a `practice` repo, let's delete it from GitHub. In the repo screen of your browser, click the top-right *Settings* button, scroll all the way down to the *Danger Zone*, and click *Delete this repository*, as shown in Figure 11.17. GitHub will ask you to type in your username and repo name to ensure that you really want to delete the repo, to prove you are not a drunken brownie chef.

So far, you've learned how to copy, edit, and host code using the GitHub web interface, which is a great introduction for beginners. Now you're ready to move up to tools that will allow you to work more efficiently with GitHub, such as GitHub Desktop and Atom Editor, to quickly move entire repos to your local computer, edit the code, and move them back online.

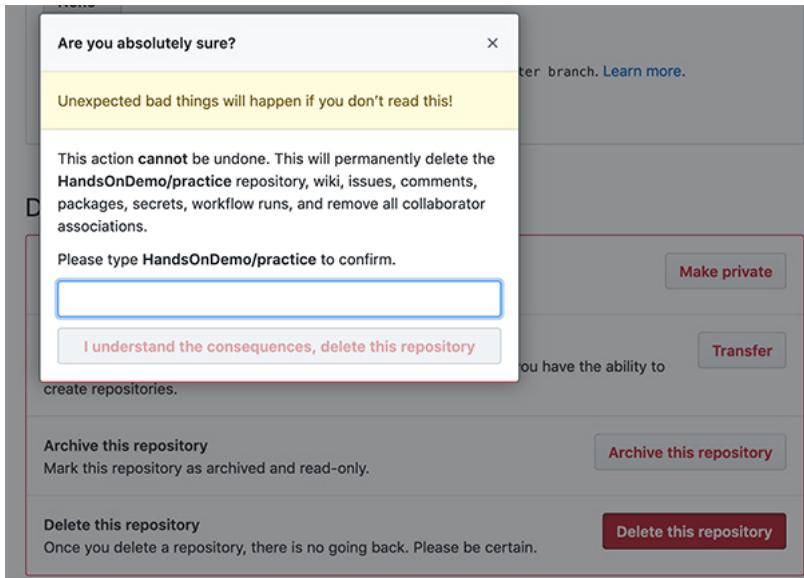


Figure 11.17: After clicking the Delete Repository button, GitHub will ask you to type your username and repo name to confirm.

GitHub Desktop and Atom Editor to Code Efficiently

Editing your code through the GitHub web interface is a good way to start, especially if you only need to make a few edits or upload a couple of files to your repo. But the web interface will feel very slow if you edit or upload multiple files in your repo. To speed up your work, we recommend that you download two free tools—GitHub Desktop and Atom Text Editor—which run on Mac or Windows computers. When you connect your GitHub web account to GitHub Desktop, it allows you to “pull” the most recent version of the code to your local computer’s hard drive, make and test your edits, and “push” your commits back to your GitHub web account. Atom Text Editor, which is also created by the makers of GitHub, allows you to view and edit code repos on your local computer more easily than the GitHub web interface. While there are many text editors for coders, Atom is designed to work well with GitHub Desktop.

Tip: Currently, neither GitHub Desktop nor Atom Editor are supported for Chromebooks, but Google’s Web Store offers several text editors, such as Text and Caret, which offer some of the functionality described below.

Let’s use GitHub Desktop to pull a copy of your `leaflet-map-simple` template to your local computer, make some edits in Atom Editor, and push your commits back up to GitHub.

1. Go to the GitHub web repo you wish to copy to your local computer. In your browser, navigate to <https://github.com/USERNAME/leaflet-map-simple>, using your GitHub username, to access the repo you created in the Copy, Edit, and Host a Simple Leaflet Map Template section of this chapter. Click the *Code > Open with GitHub Desktop* drop-down menu button near the middle of your screen, as shown in Figure 11.18. The next screen will show a link to the GitHub Desktop web page, and you should download and install the application.

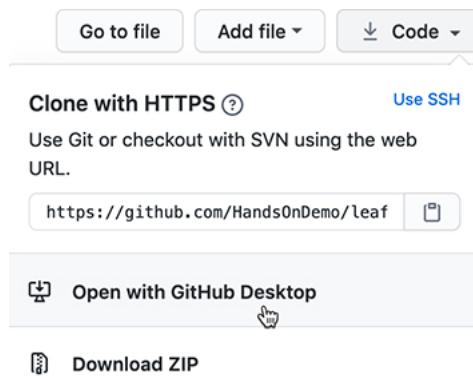


Figure 11.18: In your GitHub repo on the web, click *Code* to *Open with GitHub Desktop* to download and install GitHub Desktop.

2. When you open GitHub Desktop for the first time, you'll need to connect it to the GitHub web account you previously created in this chapter. On the welcome screen, click the blue *Sign in to GitHub.com* button, as shown in Figure 11.19, and login with your GitHub username and password. On the next screen, GitHub will ask you to click the green *Authorize desktop* button to confirm that you wish to connect to your account.
3. In the next setup screen, GitHub Desktop asks you to configure Git, the underlying software that runs GitHub. Confirm that it displays your username and click *Continue*, as shown in Figure 11.20.
4. On the “Let’s Get Started” with GitHub Desktop screen, click on *Your Repositories* on the right side to select your `leaflet-map-sample`, and further below click the blue button to *Clone* it to your local computer, as shown in Figure 11.21.

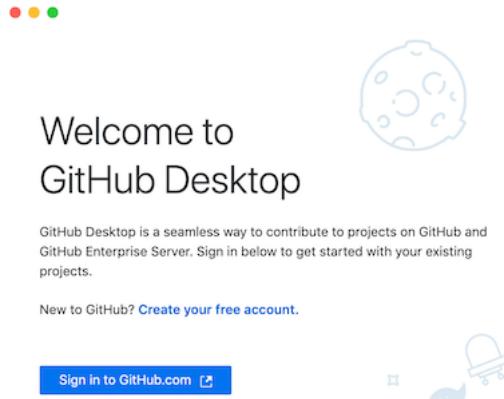


Figure 11.19: Click the blue *Sign in to GitHub.com* button to link GitHub Desktop to your GitHub account.

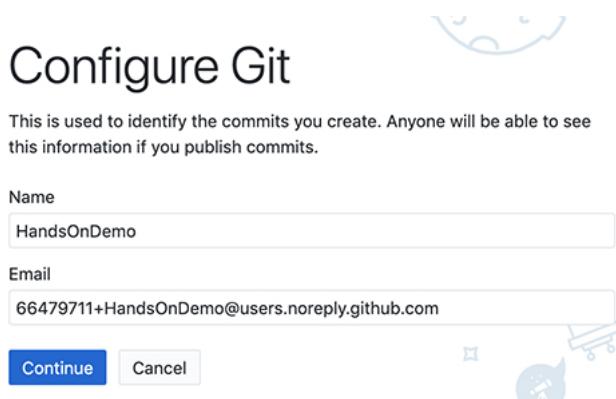


Figure 11.20: Click the *Continue* button to authorize GitHub Desktop to send commits to your GitHub account.

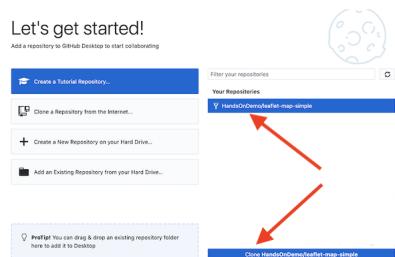


Figure 11.21: Select your *leaflet-map-simple* repo and click the *Clone* button to copy it to your local computer.

5. When you clone a repo, GitHub Desktop asks you to select the Local Path, meaning the location where you wish to store a copy of your GitHub repo on your local computer, as shown in Figure 11.22. Before you click the *Clone* button, remember the path to this location, since you'll need to find it later.

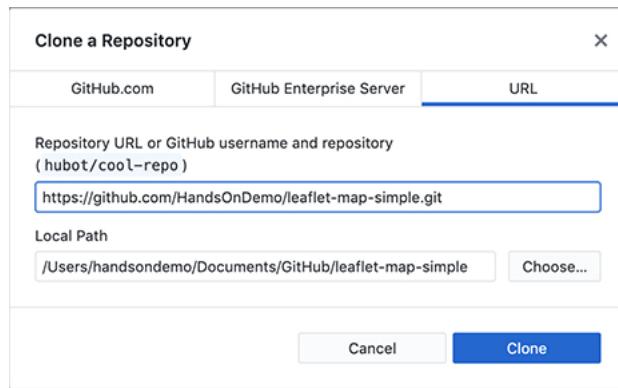


Figure 11.22: Select the Local Path where your repo will be stored on your computer, then click *Clone*.

6. On the next screen, GitHub Desktop may ask, “How are you planning to use this fork?” Select the default entry “To contribute to the parent project,” which means you plan to send your edits back to your GitHub web account, and click *Continue*, as shown in Figure 11.23.
7. Now you have copies of your GitHub repo in two places—in your GitHub web account and on your local computer—as shown in Figure 11.24. Your screen may look different, depending on whether you use Windows or Mac, and the Local Path you selected to store your files.
8. Before we can edit the code in your local computer, download and install the Atom Editor application. Then go to your GitHub Desktop screen, confirm that the Current Repository is `leaflet-map-simple`, and click the *Open in Atom* button as shown in Figure 11.25.
9. Since Atom Editor is integrated with GitHub Desktop, it opens up your entire repo as a “project,” where you can click files in the left window to open as new tabs to view and edit code, as shown in Figure 11.26. Open your `index.html` file and edit the title of your map, around line 22, then save your work.

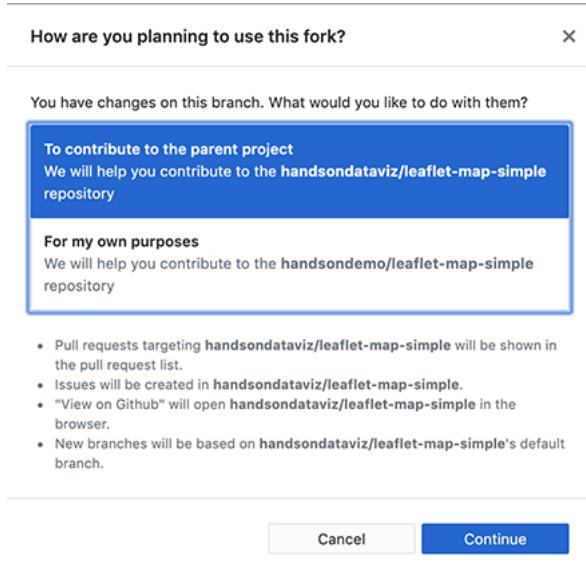


Figure 11.23: If asked how you plan to use this fork, select the default *To contribute to the parent project* and click *Continue*.

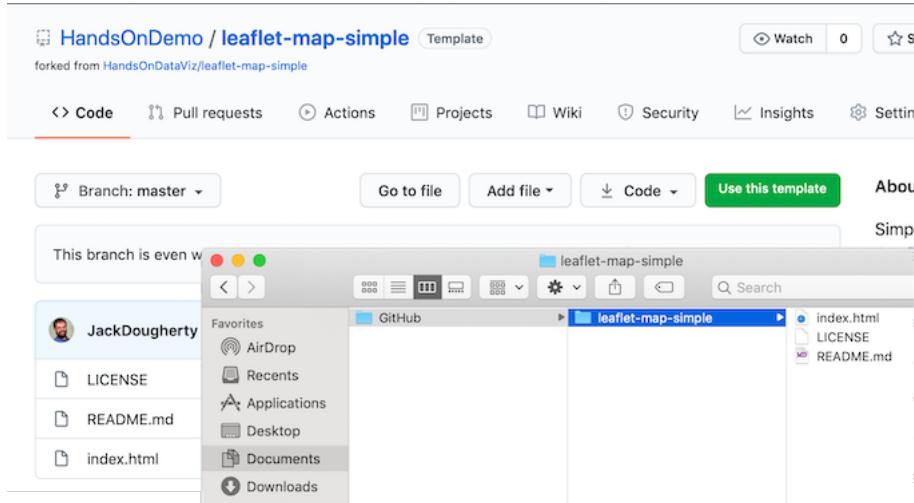


Figure 11.24: Now you have two copies of your repo: in your GitHub online account (on the left) and on your local computer (on the right, as shown in the Mac Finder). Windows screens will look different.

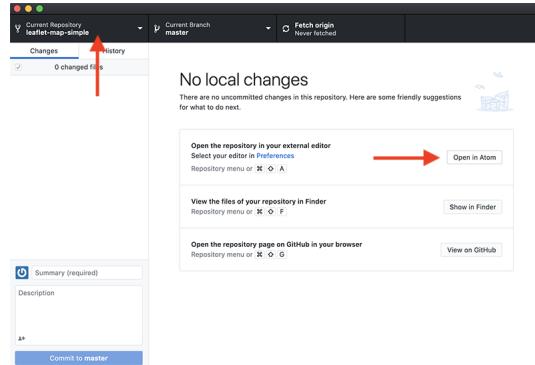


Figure 11.25: In GitHub Desktop, confirm the Current Repo and click the *Open in Atom* button to edit the code.

The screenshot shows the Atom code editor interface. On the left, there's a 'Project' sidebar with a tree view showing the repository structure: 'leaflet-map-simple' containing 'index.html', 'LICENSE', and 'README.md'. A red arrow points to the 'index.html' file icon. The main right pane shows the content of 'index.html'. The code is as follows:

```

Project index.html
1 <!DOCTYPE html>
2 <html>
3   <head>
4     <title>leaflet-map-simple</title>
5     <meta name="viewport" content="width=device-width, initial-scale=1.0">
6     <meta charset="utf-8">
7
8     <!-- Load Leaflet code library: see https://leafletjs.com/download.html -->
9     <link rel="stylesheet" href="https://unpkg.com/leaflet@1.6.0/dist/leaflet.css">
10    <script src="https://unpkg.com/leaflet@1.6.0/dist/leaflet.js"></script>
11
12    <!-- Position the map and title with Cascading Style Sheet (.css) -->
13    <style>
14      body { margin:0; padding:0; }
15      #map { position: absolute; top:0; bottom:0; right:0; left:0; }
16      #map-title { position: relative; margin-top: 10px; margin-left: 50px; float: left; }
17    </style>
18  </head>
19  <body>
20
21    <!-- Display the map and title with HTML division tags -->
22    <div id="map-title">NEW map title</div> ←
23    <div id="map"></div>

```

Figure 11.26: Atom Editor opens your repo as a *project*, where you can click files to view code. Edit your map title.

- After saving your code edit, it's a good habit to clean up your Atom Editor workspace. Right-click on the current Project and select *Remove Project Folder* in the menu, as shown in Figure 11.27. Next time you open up Atom Editor, you can right-click to *Add Project Folder*, and choose any GitHub repo that you have copied to your local computer.

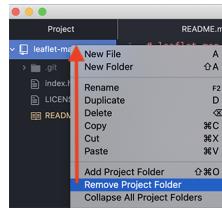


Figure 11.27: To clean up your Atom Editor workspace, right-click to *Remove Project Folder*.

- Now that you've edited the code for your map on your local computer, let's test how it looks before uploading it to GitHub. Go to the location where you saved the repo on your local computer, and right-click the `index.html` file, select Open With, and choose your preferred web browser, as shown in Figure 11.28.

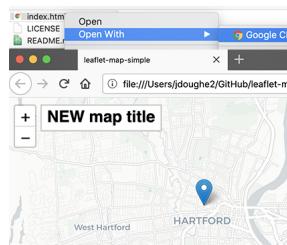


Figure 11.28: Right-click the `index.html` file on your local computer and open with a browser to check your edits.

Note: Since your browser is displaying only the *local computer* version of your code, the web address will begin with `file:///...` rather than `https://...`, as appears in your GitHub Pages online map. Also, if your code depends on online elements, those features may not function when viewing it locally. But for this simple Leaflet map template, your updated map title should appear, allowing you to check its appearance before pushing your edits to the web.

Now let's transfer your edits from your local computer to your GitHub web account, which you previously connected when you set up GitHub Desktop.

11. Go to GitHub Desktop, confirm that your Current Repo is `leaflet-map-simple`, and you will see your code edits summarized on the screen. In this two-step process, first click the blue *Commit* button at the bottom of the page to save your edits to your local copy of your repo. (If you edit multiple files, GitHub Desktop will ask you write a summary of your edit, to help you keep track of your work.) Second, click the blue *Push origin* button to transfer those edits to the parent copy of your repo on your GitHub web account. Both steps are shown in Figure 11.29.

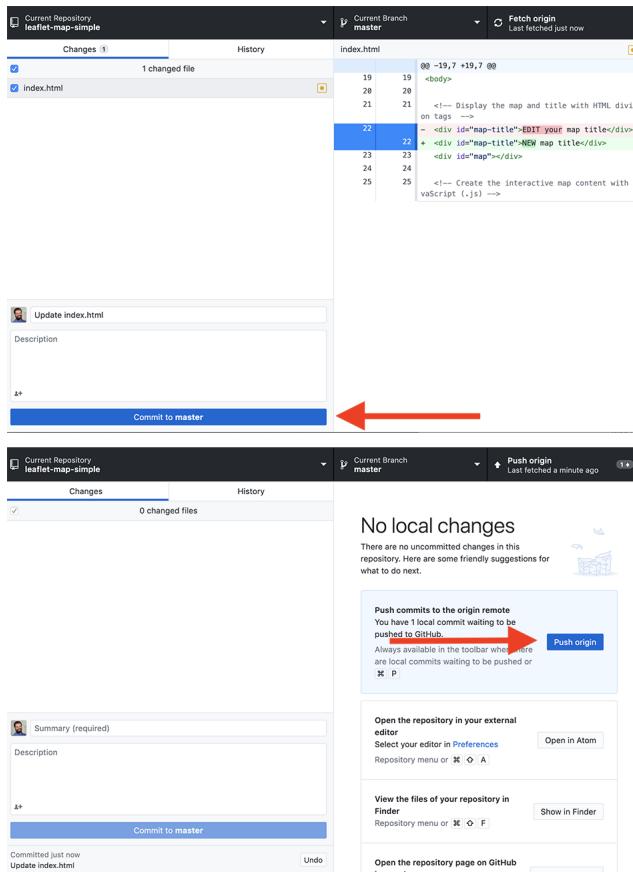


Figure 11.29: In this two-step process, click *Commit*, then click *Push origin* to save and copy your edits from your local computer to your GitHub web account, as shown in this animated GIF.

TODO above: update the annotated image and GIF to show commit to main, not master.

Congratulations! You've successfully navigated a round-trip journey of code, from your GitHub account to your local computer, and back again

to GitHub. Since you previously used the GitHub Pages settings to create an online version of your code, go see if your edited map title now appears on the public web. The web address you set up earlier follows this format <https://USERNAME.github.io/REPOSITORY>, substituting your GitHub username and repo name.

While you could have made the tiny code edit above in the GitHub web interface, hopefully you've begun to see many advantages of using GitHub Desktop and Atom Editor to edit code and push commits from your local computer. First, you can make more complex code modifications with Atom Editor, which includes search, find-and-replace, and other features to work more efficiently. Second, when you copy the repo to your local computer, you can quickly drag-and-drop multiple files and subfolders for complex visualizations, such as data, geography, and images. Third, depending on the type of code, you may be able to test how it works locally with your browser, before uploading your commits to the public web.

Tip: Atom Editor has many built-in features that recognize and help you edit code, plus the option to install more packages in the Preferences menu. One helpful built-in tool is *Edit > Toggle Comments*, which automatically detects the coding language and converts the selected text from executable code to non-executed code comments. Another built-in tool is *Edit > Lines > Auto Indent*, which automatically cleans up selected text or an entire page of code for easier reading.

GitHub also offers a powerful platform for collaborative projects, such as *Hands-On Data Visualization*. As co-authors, we composed the text of these book chapters and all of the sample code templates on GitHub. Jack started each day by “pulling” the most recent version of the book from our shared GitHub account to his local computer using GitHub Desktop, where he worked on sections and “pushed” his commits (aka edits) back to GitHub. At the same time, Ilya “pulled” the latest version and “pushed” his commits back to GitHub as well. Both of us see the commits that each other made, line-by-line in green and red (showing additions and deletions), by selecting the GitHub repo *Code* tab and clicking on one of our commits, as shown in Figure 11.30.

Although GitHub does not operate like Google Documents, which displays live edits, the platform has several advantages when working collaboratively with code. First, since GitHub tracks every commit we make, it allows us to go back and restore a very specific past version of the code if needed. Second, when GitHub repos are public, anyone can view your code and submit an “issue” to notify the owner about an idea or problem, or send a “pull request” of suggested code edits, which the owner can accept or reject. Third, GitHub allows collaborators to create different “branches” of a repo in order to make edits, and then “merge” the branches back together if desired. Occasionally, if two or more coders attempt to push incompatible commits to the same repo, GitHub will warn about a “Merge Conflict,” and ask you to resolve these conflicts in order to preserve everyone’s work.

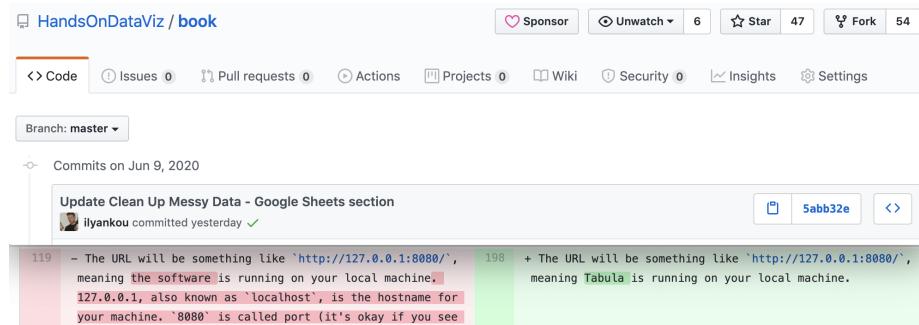


Figure 11.30: Drag-and-drop the file to the upload screen.

Many coders prefer to work on GitHub using its Command Line Interface (CLI), which means memorizing and typing specific commands directly into the Terminal application on Mac or Windows, but this is beyond the scope of this introductory book.

Summary

If this is the first time you've forked, edited, and hosted live code on the public web, welcome to the coding family! We hope you agree that GitHub is a powerful platform for engaging in this work and sharing with others. While beginners will appreciate the web interface, you'll find that the GitHub Desktop and Atom Editor tools make it much easier to work with Chart.js and Highcharts code templates in Chapter 12 and the Leaflet map code templates in Chapter 13. Let's build on your brand-new coding skills to create more customized charts and maps in the next two chapters.

Chapter 12

Chart.js and Highcharts Templates

In Chapter 7: Chart Your Data, we looked at powerful drag-and-drop tools, such as Google Sheets, Datawrapper and Tableau Public to build interactive charts.

In this chapter, we will look into creating interactive charts using two popular JavaScript libraries, Chart.js and Highcharts. Since we don't expect our readers to be proficient in JavaScript or any other programming language, we designed templates that you can copy to your own GitHub account, substitute data files, and publish them to the web without writing a single line of code. But for those of you who are code-curious, we will show how the JavaScript code in these templates can be customized.

Now, why would anyone prefer JavaScript to easy-to-use Datawrapper or Tableau, you may wonder? Well, a few reasons. Although JavaScript code may seem overwhelming and intimidating at first, it allows for greater customization in terms of colors, padding, interactivity, and data handling than most third-party tools can offer. In addition, you can never be sure that third-party apps will remain free, or at least have a free tier, forever, whereas open-source tools are here to stay free of charge.

Note: Although both libraries are open-source, Highcharts comes with a stricter license which allows it to be used for free for non-commercial projects only, such as personal, school, or non-profit organization website. Keeping that in mind, we primarily focus on Chart.js, which is distributed under MIT license that lets you use the library for commercial projects as well.

Table 12.1 lists all types of charts that we will look at in this chapter. Both libraries include many more default chart types that you can explore in Chart.js Samples and Highcharts Demos. However, we strongly advise against using

some chart types, such as three-dimensional ones, for reasons we discussed in the Chart Design Principles section of Chapter 7.

Table 12.1: Chart Code Templates, Best Uses, and Tutorials

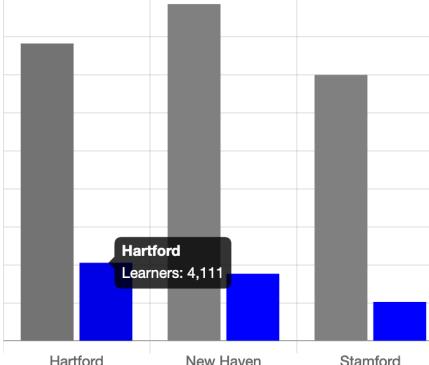
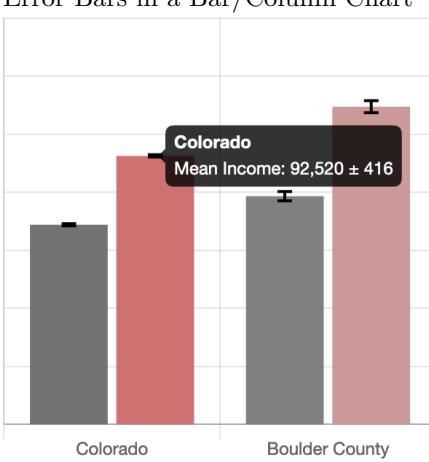
Chart	Best use and tutorials in this book									
Bar or Column Chart  <p>A bar chart comparing three locations: Hartford, New Haven, and Stamford. The y-axis represents the number of learners. Hartford has approximately 4,111 learners, New Haven has approximately 5,000 learners, and Stamford has approximately 3,000 learners. A tooltip for Hartford shows "Hartford Learners: 4,111".</p> <table border="1"> <thead> <tr> <th>Location</th> <th>Learners</th> </tr> </thead> <tbody> <tr> <td>Hartford</td> <td>4,111</td> </tr> <tr> <td>New Haven</td> <td>5,000</td> </tr> <tr> <td>Stamford</td> <td>3,000</td> </tr> </tbody> </table>	Location	Learners	Hartford	4,111	New Haven	5,000	Stamford	3,000	Best to compare categories side-by-side. If labels are long, use horizontal bars instead of vertical columns. Power tool: Bar or Column Chart with CSV data in Chart.js code template and tutorial	
Location	Learners									
Hartford	4,111									
New Haven	5,000									
Stamford	3,000									
Error Bars in a Bar/Column Chart  <p>A bar chart comparing two regions: Colorado and Boulder County. The y-axis represents mean income. Colorado has a mean income of \$92,520 ± \$416, and Boulder County has a mean income of \$95,000 ± \$400. Error bars are shown above each bar. A tooltip for Colorado shows "Colorado Mean Income: 92,520 ± 416".</p> <table border="1"> <thead> <tr> <th>Region</th> <th>Mean Income</th> <th>Error</th> </tr> </thead> <tbody> <tr> <td>Colorado</td> <td>\$92,520</td> <td>± \$416</td> </tr> <tr> <td>Boulder County</td> <td>\$95,000</td> <td>± \$400</td> </tr> </tbody> </table>	Region	Mean Income	Error	Colorado	\$92,520	± \$416	Boulder County	\$95,000	± \$400	Best to show margin of error bars when comparing categories side-by-side. If labels are long, use horizontal bars instead of vertical columns. Power tool: Error Bars in Bar/Column Chart with CSV data in Chart.js code template and tutorial
Region	Mean Income	Error								
Colorado	\$92,520	± \$416								
Boulder County	\$95,000	± \$400								

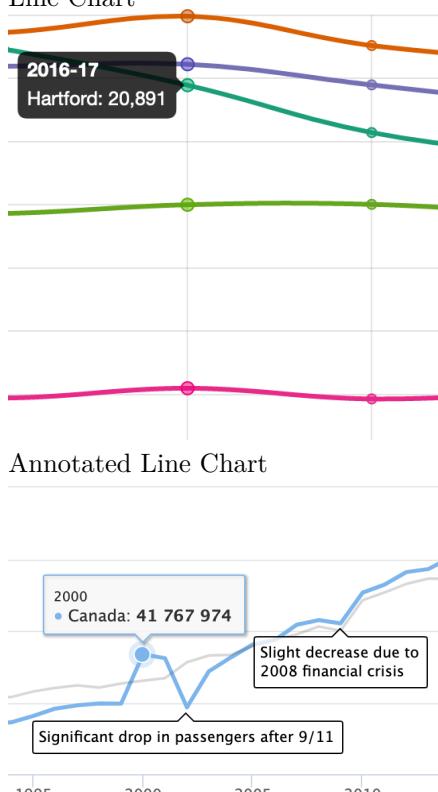
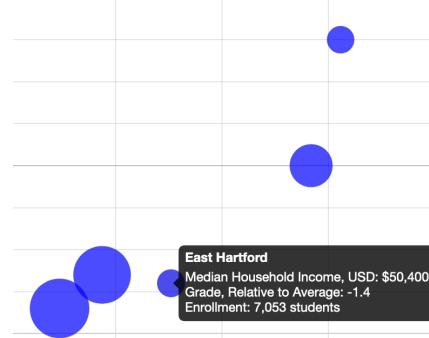
Chart	Best use and tutorials in this book
Line Chart	Best to show continuous data, such as change over time. Power tool: Line Chart with CSV data in Chart.js code template and tutorial
	Best to add contextual notes inside chart of continuous data, such as change over time. Power tool: Annotated Line Chart with CSV data in Highcharts code template and tutorial
Scatter Chart	Best to show the relationship between two datasets as XY coordinates to reveal possible correlations. Power tool: Scatter Chart with CSV data in Chart.js code template and tutorial

Chart	Best use and tutorials in this book
Bubble Chart 	Best to show the relationship between three or four sets of data, with XY coordinates, bubble size, and color. Power tool: Bubble Chart with CSV data in Chart.js code template and tutorial

Bar or Column Chart with Chart.js

In this section, we will show you how to create bar or column charts using a Chart.js. To do so, we will be using a Chart.js come template that pulls data from a CSV file, as shown in Figure 12.1. This column chart shows how many students in five school districts in Connecticut were English-language learners in 2018-2019 academic year.

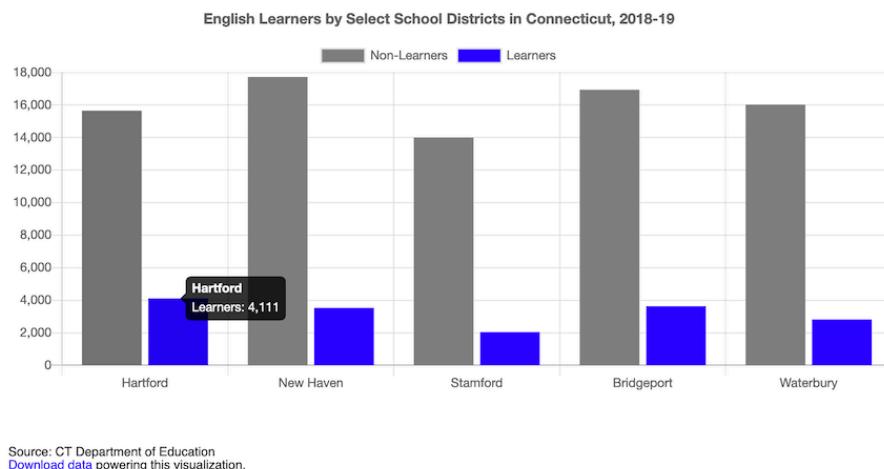


Figure 12.1: Bar chart with Chart.js: explore the interactive version.

To create your own bar or column chart with CSV data using our Chart.js template:

1. Go to our GitHub repo that contains the code for the chart in Figure 12.1, log into your GitHub account, and click *Use this template* to create a copy that you can edit.

Note: If you don't remember how to use GitHub, we recommend you revisit Chapter 11: Edit and Host Code with GitHub.

The repo contains three files that are directly related to the chart:

- `index.html` contains HTML (markdown) and CSS (stylesheets) that tell the browser how to style the document that contains the chart, and what libraries to load,
- `script.js` contains the JavaScript code that reads data from the CSV file and constructs the interactive chart, and
- `data.csv` is the comma-separated file that keeps all the data in the chart, and can be edited in a text editor, or Google Sheets/Excel etc.

The two remaining files are a `README.md` that describes the contents of the repo, and `bar.png` that is just an image that you can see in the README. All other GitHub templates in this chapter will be similarly structured.

2. Prepare your data in CSV format and upload into a `data.csv` file. Place labels that will appear along the axis in the first column, and each data series in its own column. Your CSV must contain at least two columns (labels and one data series). You can add as many data series columns as you wish.

district	nonlearner	learner
Hartford	15656	4111
New Haven	17730	3534

3. In `script.js`, customize the values of variables. Since you may not be familiar with JavaScript, let's take a look at the code snippet that describes a single variable in the file:

```
// `false` for vertical column chart, `true` for horizontal bar chart
var HORIZONTAL = false;
```

The first line starts with // and is a comment to help you understand what the variable in the next line is responsible for. It does not affect the code. As you can see, if the variable `HORIZONTAL` is set to `false`, the chart would have vertical bars (also known as columns). If set to `true`, the chart will contain horizontal bars. The second line contains the variable declaration itself. The equal sign (=) assigns the value that you see on the right (`false`) to the variable (`var`) called `HORIZONTAL` to the left. This line ends with the semicolon (;).

Below are some of the variables available for you to customize in `script.js`:

```

var TITLE = 'English Learners by Select School Districts in CT, 2018-19';

// `false` for vertical column chart, `true` for horizontal bar chart
var HORIZONTAL = false;

// `false` for individual bars, `true` for stacked bars
var STACKED = false;

// Which column defines 'bucket' names?
var LABELS = 'district';

// For each column representing a data series, define its name and color
var SERIES = [
    {
        column: 'nonlearner',
        name: 'Non-Learners',
        color: 'grey'
    },
    {
        column: 'learner',
        name: 'Learners',
        color: 'blue'
    }
];

// x-axis label and label in tooltip
var X_AXIS = 'School Districts';

// y-axis label, label in tooltip
var Y_AXIS = 'Number of Enrolled Students';

// `true` to show the grid, `false` to hide
var SHOW_GRID = true;

// `true` to show the legend, `false` to hide
var SHOW_LEGEND = true;

```

These basic variables should be enough to get you started. It is natural that you will want to move the legend, edit the appearance of the tooltip, or change the colors of the grid lines. We recommend you look at the official Chart.js documentation to get help with that.

Error Bars with Chart.js

If your data comes with uncertainty (margins of error), we recommend you show it in your visualizations with the use of error bars. The bar chart template shown in Figure 12.2 shows median and mean (average) income for different-sized geographies: the US state of Colorado, Boulder County, Boulder city, and a census tract in the city.

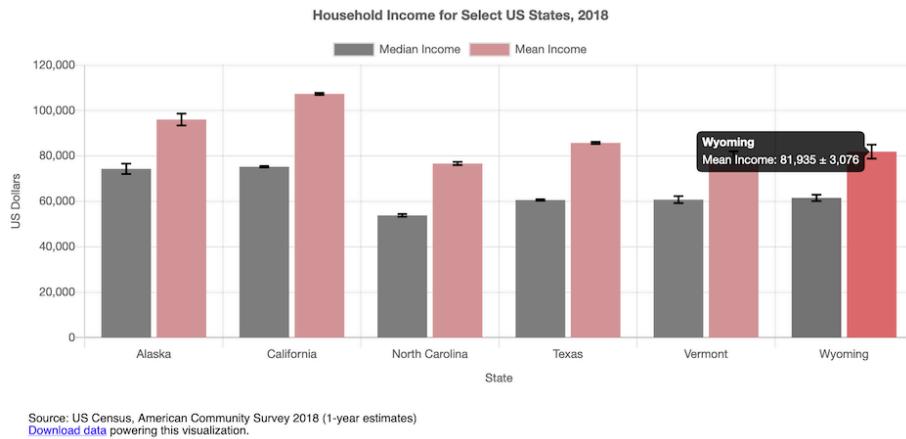


Figure 12.2: Interactive bar chart with error bars in Chart.js. Explore the interactive version.

To create your own bar or column chart with error bars, with data loaded from a CSV file, using our Chart.js template follow the steps below:

1. Go to our GitHub repo for this Chart.js template that contains the code for the chart in Figure 12.2, log into your GitHub account, and click *Use this template* to create a copy that you can edit.
2. Prepare your data in CSV format and upload into a `data.csv` file. Place labels that will appear along the axis in the first column, and each data series in its own column (accompanied by a column with uncertainty values). Your CSV must contain at least three columns (labels and one data series with associated uncertainty values). You can add as many data series columns as you wish.

geo	median	median_moe	mean	mean_moe
Colorado	68811	364	92520	416
Boulder County	78642	1583	109466	2061
Boulder city	66117	2590	102803	3614
Tract 121.02	73396	10696	120588	19322

3. In `script.js`, customize the values of variables shown in the code snippet below:

```

var TITLE = 'Household Income for Select US Geographies, 2018';

// `false` for vertical (column) chart, `true` for horizontal bar
var HORIZONTAL = false;

// `false` for individual bars, `true` for stacked bars
var STACKED = false;

// Which column defines "bucket" names?
var LABELS = 'geo';

// For each column representing a series, define its name and color
var SERIES = [
  {
    column: 'median',
    name: 'Median Income',
    color: 'grey',
    errorColumn: 'median_moe'
  },
  {
    column: 'mean',
    name: 'Mean Income',
    color: '#cc9999',
    errorColumn: 'mean_moe'
  }
];

// x-axis label and label in tooltip
var X_AXIS = 'Geography';

// y-axis label and label in tooltip
var Y_AXIS = 'US Dollars';

// `true` to show the grid, `false` to hide
var SHOW_GRID = true;

// `true` to show the legend, `false` to hide
var SHOW_LEGEND = true;

```

For more customization, see Chart.js documentation.

Line Chart with Chart.js

Line charts are often used to show temporal data, or change of values over time. The x-axis represents time intervals, and the y-axis represents observed values. Note that unlike column or bar charts, y-axes of line charts do not have to start at zero because we rely on the position and slope of the line to interpret its meaning. The line chart in Figure 12.3 shows the number of students in select school districts in Connecticut from 2012-2013 to 2018-19 academic years. Each line has a distinct color, and the legend helps establish the color-district relations.

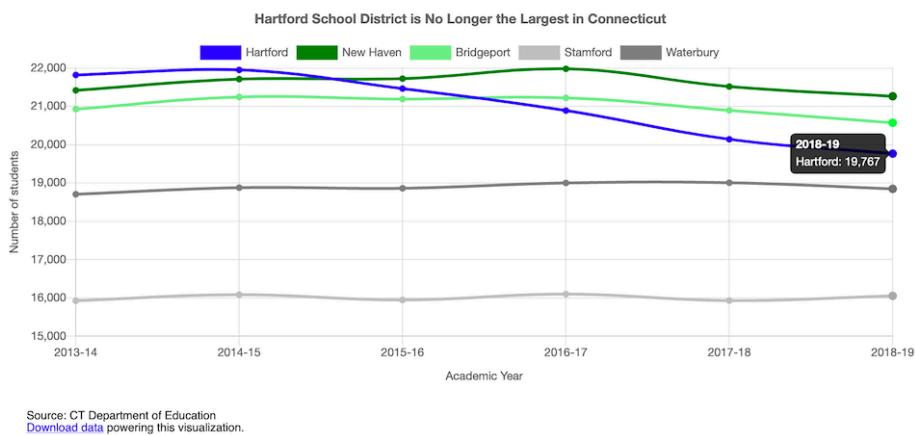


Figure 12.3: Interactive line chart with Chart.js. Explore the interactive version.

To create your own line chart with Chart.js, with data loaded from a CSV file, you can:

1. Go to our GitHub repo for the Chart.js template that contains the code of the line chart shown in Figure 12.3, log into your GitHub account, and click *Use this template* to create a copy that you can edit.
2. Prepare your data in CSV format and upload into a `data.csv` file. Place labels that will appear along the axis in the first column, and each data series in its own column. Your CSV must contain at least two columns (labels and one data series).

Tip: You can add as many data series columns as you wish, but choose a reasonable number of lines, since humans can distinguish only a limited number of colors. If you need to display multiple lines, consider using only one color to highlight the most significant line in your data story, and color others in gray, as you will learn in the Draw Attention to Meaning section of Chapter 16.

year	Hartford	New Haven	Bridgeport	Stamford	Waterbury
2013-14	21820	21420	20929	15927	18706
2014-15	21953	21711	21244	16085	18878
2015-16	21463	21725	21191	15946	18862
2016-17	20891	21981	21222	16100	19001
2017-18	20142	21518	20896	15931	19007
2018-19	19767	21264	20572	16053	18847

3. In `script.js`, customize the values of variables shown in the code snippet below:

```
var TITLE = 'Hartford School District is No Longer Largest in Connecticut';

// x-axis label and label in tooltip
var X_AXIS = 'Academic Year';

// y-axis label and label in tooltip
var Y_AXIS = 'Number of Students';

// Should y-axis start from 0? `true` or `false`
var BEGIN_AT_ZERO = false;

// `true` to show the grid, `false` to hide
var SHOW_GRID = true;

// `true` to show the legend, `false` to hide
var SHOW_LEGEND = true;
```

Remember to look at the official Chart.js documentation if you want to add more features. If something isn't working as desired, visit StackOverflow to see if anyone had already solved your problem.

Annotated Line Chart with Highcharts

Although annotations are common elements of various type charts, they are especially important in line charts. Annotations help give historic context to the lines, explain sudden dips or raises in values. Figure 12.4 shows change in air passenger traffic for Australia and Canada between 1970 and 2018 (according to the World Bank). You can notice that both countries experienced a dip in 2009, the year after the 2008 financial crisis as suggested by the annotation.

Unfortunately, Chart.js is not great at showing annotations. This is why we are switching to Highcharts for this particular example. But don't worry – you will see that the process is hardly different from the previous Chart.js examples.

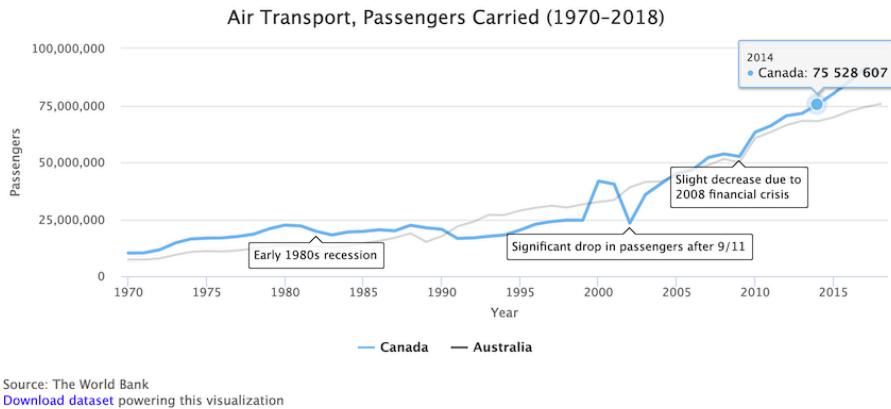


Figure 12.4: Interactive annotated chart with Highcharts. Explore the interactive version.

To create your own annotated line chart with Highcharts, with data loaded from a CSV file, do the following:

1. Go to our GitHub repo that contains code for the chart shown in Figure 12.4, log into your GitHub account, and click *Use this template* to create a copy that you can edit.
2. Prepare your data in CSV format and upload into a `data.csv` file. Place labels that will appear along the axis in the first column, and each data series in its own column. Your CSV must contain at least three columns (labels, one data series, and notes). You can add as many data series columns as you wish, but you can only have one annotation (final column) per row.

Year	Canada	Australia	Note
1980	22453000	13648800	
1981	22097100	13219500	
1982	19653800	13187900	Early 1980s recession

3. In `script.js`, customize the values of variables shown in the code snippet below:

```
var TITLE = 'Air Transport, Passengers Carried (1970-2018)';

// Caption underneath the chart
var CAPTION = 'Source: The World Bank';
```

```
// x-axis label and label in tooltip
var X_AXIS = 'Year';

// y-axis label and label in tooltip
var Y_AXIS = 'Passengers';

// Should y-axis start from 0? `true` or `false`
var BEGIN_AT_ZERO = true;

// `true` to show the legend, `false` to hide
var SHOW_LEGEND = true;
```

If you wish to further customize your chart, use the Highcharts API reference that lists all available features.

Scatter Chart with Chart.js

Now when you've seen Highcharts in action, let's get back to Chart.js and see how to build an interactive scatter chart. Remember that scatter charts (also *scatterplots*) are used to display data of 2 or more dimensions. Figure 12.5 shows the relationship between household income and test performance for school districts in Connecticut. Using x- and y-axes to show two dimensions, it is easy to see that test performance improves as household income goes up.

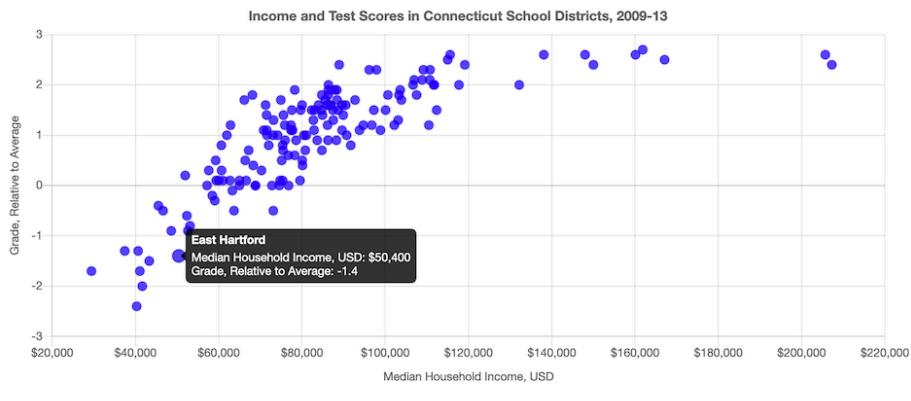


Figure 12.5: Interactive scatter chart with Chart.js. Explore the interactive version.

To create your own scatter plot with Chart.js, with data loaded from a CSV file, you can:

1. Go to our GitHub repo that contains the code for the chart shown in Figure 12.5, log into your GitHub account, and click *Use this template* to create a copy that you can edit.
2. Prepare your data in CSV format and upload into a `data.csv` file. The first two columns should contain x- and y-values respectively, and the third column should contain the point name that will appear on mouse hover.

```
| income | grades | district |
| 88438 | 1.7    | Andover   |
| 45505 | -0.4   | Ansonia   |
| 75127 | 0.5    | Ashford   |
| 115571| 2.6    | Avon      |
```

3. In `script.js`, customize the values of variables shown in the code snippet below:

```
var TITLE = 'Income and Test Scores in Connecticut School Districts, 2009-13';

var POINT_X = 'income'; // column name for x values in data.csv
var POINT_X_PREFIX = '$'; // prefix for x values, eg '$'
var POINT_X_POSTFIX = ''; // postfix for x values, eg '%'

var POINT_Y = 'grades'; // column name for y values in data.csv
var POINT_Y_PREFIX = ''; // prefix for x values, eg 'USD '
var POINT_Y_POSTFIX = ''; // postfix for x values, eg ' kg'

var POINT_NAME = 'district'; // point names that appear in tooltip
var POINT_COLOR = 'rgba(0,0,255,0.7)'; // eg `black` or `rgba(10,100,44,0.8)`
var POINT_RADIUS = 5; // radius of each data point

var X_AXIS = 'Median Household Income, USD'; // x-axis label, label in tooltip
var Y_AXIS = 'Grade, Relative to Average'; // y-axis label, label in tooltip

var SHOW_GRID = true; // `true` to show the grid, `false` to hide
```

A similarly good-looking interactive chart can be constructed in Highcharts, although it is up to you to undertake that challenge. In the meanwhile, remember to refer to the official Chart.js documentation if you want to further tweak your chart.

You may want to show an additional third variable, such as enrollment in each school district, in the same scatter chart. You can do so by resizing each dot so that larger school districts are marked with a larger circle, and smaller districts are shown using a smaller dot. Such use of size will result in a *bubble chart*, which we will look at next.

Bubble Chart with Chart.js

Bubble charts are similar to scatter plots, but it adds one more variable (also known as dimension): the size of each point (marker) also represents a value.

The bubble chart in Figure 12.6 shows how median household income (x-axis) and test performance (y-axis) in 6 school districts in Connecticut are related. The size of data point corresponds to the number of students enrolled in the school district: bigger circles represent larger school districts.

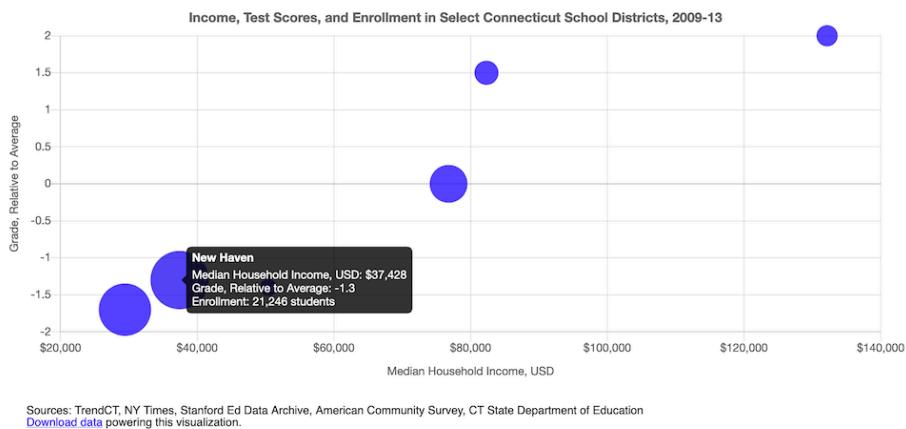


Figure 12.6: Interactive bubble chart with Chart.js. Explore the interactive version.

To create your own bubble chart with Chart.js, with data loaded from a CSV file, you can:

1. Go to our GitHub repo for this template, log into your GitHub account, and click *Use this template* to create a copy that you can edit.
2. Prepare your data in CSV format and upload into a `data.csv` file. The first two columns should contain x- and y-values respectively. The third column should contain bubble names that will appear on mouse hover. The final, fourth column, represents the size of your bubble.

income	grades	district	enrollment
29430	-1.7	Hartford	21965
82322	1.5	West Hartford	10078
50400	-1.4	East Hartford	7053

3. In `script.js`, customize the values of variables shown in the code snippet below:

```

var TITLE = 'Income, Test Scores, and Enrollment in Select \
Connecticut School Districts, 2009-13';

var POINT_X = 'income'; // column name for x values in data.csv
var POINT_X_PREFIX = '$'; // prefix for x values, eg '$'
var POINT_X_POSTFIX = ''; // postfix for x values, eg '%'

var POINT_Y = 'grades'; // column name for y values in data.csv
var POINT_Y_PREFIX = ''; // prefix for x values, eg 'USD '
var POINT_Y_POSTFIX = ''; // postfix for x values, eg ' kg'

var POINT_R = 'enrollment'; // column name for radius in data.csv
var POINT_R_DESCRIPTION = 'Enrollment'; // description of radius value
var POINT_R_PREFIX = ''; // prefix for radius values, eg 'USD '
var POINT_R_POSTFIX = ' students'; // postfix for radius values, eg ' kg'
var R_DENOMINATOR = 800; // use this to scale the dot sizes, or set to 1
                        // if your dataset contains precise radius values

var POINT_NAME = 'district'; // point names that appear in tooltip
var POINT_COLOR = 'rgba(0,0,255,0.7)'; // eg `black` or `rgba(10,100,44,0.8)`

var X_AXIS = 'Median Household Income, USD'; // x-axis label, label in tooltip
var Y_AXIS = 'Grade, Relative to Average'; // y-axis label, label in tooltip

var SHOW_GRID = true; // `true` to show the grid, `false` to hide

```

Tip: To display smaller data points that may be hidden behind larger neighbors, use semi-transparent circles with RGBA color codes. The first three characters represent red, green, and blue, while the a stands for alpha and represents the level of transparency on a scale from 0.0 (full transparent) to 1.0 (fully opaque). For example, `rgba(160, 0, 0, 0.5)` creates a red color that is semi-transparent. Learn more by playing with RGBA color values at W3Schools.

If you have more than three variables that you would like to show in your bubble chart, you can use *color* and *glyphs* (instead of simple dots) to represent two extra dimensions. For example, you may want to use the blue color to only show school districts in Fairfield County (generally a richer part of CT) and gray color to represent all other districts. You may want to use circles, squares, and triangles to represent results for males, females, and non-binary students. We won't be showing you how to achieve this, but we can assure you that it can be done in 5-10 extra lines of code.

Chart.js is pretty limitless when it comes to customization, but remember not to overwhelm the viewer and communicate only the data that are necessary to prove or illustrate your idea.

Summary

In this chapter, we introduced Chart.js and Highcharts templates that can be used to construct rich and interactive charts that you can host in your own GitHub account, and embed them anywhere on the web. You can use these templates as a base to kickstart your interactive visualizations. You can refer to Chart.js Samples and Chart.js documentation for more information on Chart.js customization and troubleshooting. Highcharts Demos gallery shows plenty of charts along with the code that you can copy, and Highcharts API Reference lists all features available to refine your visualizations. Just remember that you need to obtain a license to use Highcharts in commercial projects.

In the next chapter, we will introduce Leaflet.js map templates that were designed in a similar fashion to the chart templates we have just looked at. Leaflet is a leading open-source JavaScript library for web mapping, and will let you create stunning interactive maps that live in your GitHub account and can be shared across the web.

Chapter 13

Leaflet Map Templates

In Chapter 8: Map Your Data, we described several easy-to-learn drag-and-drop tools, such as BatchGeo, Google My Maps and Datawrapper. In this chapter, we offer more advanced map tutorials using our open-source code templates, which you can copy and modify, using skills you learned in Chapter 11: Edit and Host Code with GitHub. We built all of the templates in this chapter with Leaflet, a powerful open-source code library for creating interactive maps on desktop or mobile devices.

No coding skills are required to use our two introductory templates because they pull your map data from a linked Google Sheet. The first template, Leaflet Maps with Google Sheets, is a general-purpose tool that can display points, polygons, or polylines, using your choice of colors, icons, and images, based on data uploaded into your linked Google Sheet and GitHub repository. It also includes the option to display a table of point markers next to your map. The second template, Leaflet Storymaps with Google Sheets, guides readers through a point-by-point tour, with a scrolling narrative to display text, images, audio, video, and scanned map backgrounds, all loaded into your linked Google Sheet and GitHub repo. With either template, look back at Chapter 3 to geocode addresses with a Google Sheets Add-on.

Our more advanced Leaflet templates are designed to help users develop their map coding skills. Even if you have no prior coding experience, but can follow instructions and are *code-curious*, start with the Leaflet Point Map with CSV Data template, which introduces you to the basics of Leaflet map coding by pulling point data from a generic comma-separated values file. Then move on to more advanced coding examples, such as the Leaflet Heatmap template to show point clusters as hotspots, the Leaflet Searchable Point Map template to allow users to search and filter multiple locations, and the Leaflet Maps with Open Data APIs template to continuously pull the most current information directly from open repositories, a topic we introduced in Chapter 4. These Leaflet templates are written in the three most common coding languages on

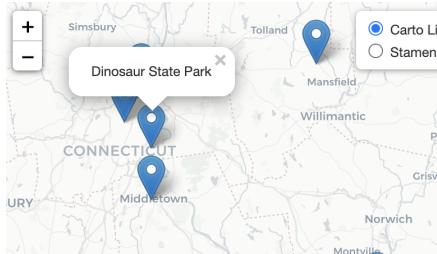
the web: Hypertext Markup Language (HTML) to structure content on a web page (typically in a file named `index.html`), Cascading Style Sheets (CSS) to shape how content appears on the page (either inside `index.html` or a separate file such as `style.css`), and JavaScript to create the interactive map using the open-source Leaflet code library (either inside `index.html` or a separate file such as `script.js`). These Leaflet templates also include links to other online components, such as zoomable basemap tiles from various open-access online providers, such as Carto, Esri, Stamen, and Open Street Map. Also, these code templates can place points on a map (typically from a `data.csv` file) or polygons or polyline geography (typically from a `map.geojson` file), which you'll learn to create in Chapter 14: Transform Your Map Data. If you're new to coding, creating Leaflet maps can be a great place to start and quickly see the results of what you've learned. To help solve problems that may arise, see how to Fix Common Mistakes in the appendix.

Table 13.1: Map Code Templates, Best Uses, and Tutorials

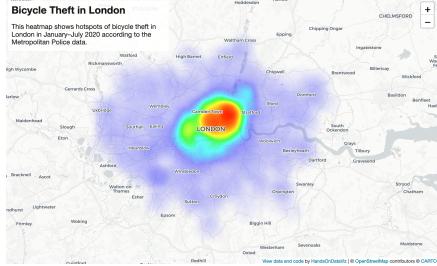
Map Templates	Best use and tutorials in this book
Leaflet Maps with Google Sheets 	Best to show interactive points, polygons, or polylines, using your choice of colors, styles, and icons, based on data loaded into your linked Google Sheet (or CSV file) and GitHub repository. Includes option to display a table of point map markers next to your map. Template with tutorial: Leaflet Maps with Google Sheets
Leaflet Storymaps with Google Sheets 	Best to show a point-by-point guided tour, with a scrolling narrative to display text, images, audio, video, and scanned map backgrounds loaded into your linked Google Sheet (or CSV file) and GitHub repository. Template with tutorial: Leaflet Storymaps with Google Sheets

Map Templates

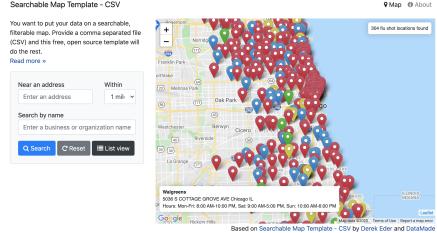
Leaflet Point Map with CSV Data



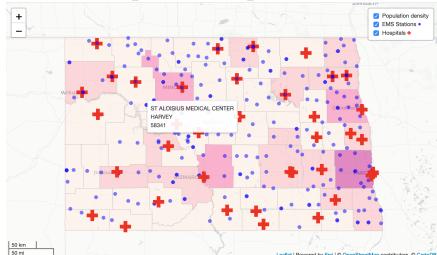
Leaflet Heatmap Points with CSV Data



Leaflet Searchable Point Map with CSV Data



Leaflet Maps with Open Data APIs



Best use and tutorials in this book

Learn how to code your own Leaflet point map that pulls data from a CSV file in your GitHub repo. Template with tutorial: Leaflet Maps with CSV Data

Best to show clusters of points as colored hotspots to emphasize high frequency or density of cases. Template with tutorial: Leaflet Heatmap

Best to show multiple locations for users to search by name or proximity, or filter by category, with optional list view. Developed by Derek Eder from DataMade. Template with tutorial: Leaflet Searchable Map with CSV

Learn how to code your own Leaflet map with an application programming interface (API) that continuously pulls the most current information directly from an open-data repository, such as Socrata and others. Template with tutorial: Leaflet Maps with Open Data APIs template

TODO above: Update screenshots after updating Leaflet demos, and make uniform size.

Leaflet Maps with Google Sheets

Sometimes you need to create a map that cannot be made easily with drag-and-drop tools, because you need to customize its appearance or show some combination of point, polygon, or polyline data. One solution is to build your map based on our Leaflet Maps with Google Sheets code template, which allows you to display custom point icons, pick any choropleth color palettes, and stack different combinations of map data layers, as shown in Figure 13.1. If you've explored prior chapters in this book, this template is a good template for newer users, because you enter your map data and settings in a linked Google Sheet, as shown in Figure 13.2, and upload images or geographic files into a folder in your GitHub repository. All of the data you enter can easily be exported and migrated to other platforms as visualization technology continues to evolve in the future, as we discussed in the how to choose tools section in Chapter 2. Furthermore, the map design is responsive, meaning it automatically resizes to look good on small or large screens. Finally, the Leaflet Maps template is built on flexible open-source software that's written primarily in JavaScript, a very common coding language for the web, so you can customize it further if you have skills or support from a developer.

TODO: Create and insert a new version of the demo, featuring ECGreenway route thru CT or East Coast, points with photos, and pop density of towns to highlight how this bike route connects cities.

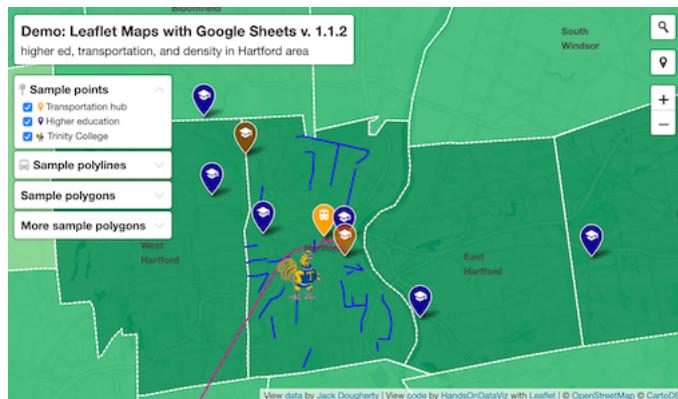


Figure 13.1: Explore the interactive Leaflet Maps with Google Sheets.

Tutorial Outline

Before beginning this tutorial, you must have a Google Drive account, and know how to File > Make a Copy in Google Sheets, as described in Chapter 3. Also, you must have a GitHub account, and know how to Edit and Host Code with

	A	B
	Setting	Customize
1		
2	Map Info	
3	Map Title	Demo: Leaflet Maps with Google Sheets v. 1.1.2
4	Map Subtitle	higher ed, transportation, and density in Hartford area
5	Display Title	topleft
6	Author Name	Jack Dougherty
7	Author Email or Website	jack.dougherty@trincoll.edu
8	Author Code Credit	HandsOnDataViz
9	Author Code Repo	https://github.com/JackDougherty/leaflet-maps-with-google-sheets
10	Map Settings	
11	Basemap Tiles	CartoDB PositronNoLabels
12	Cluster Markers	off
13	Intro Popup Text	
14	Initial Zoom	
15	Initial Center Latitude	
16	Initial Center Longitude	
17	Map Controls	
18	Search Button	topright
19	Mapzen Search Key	search-JPBt5y
20	Search Radius	5 miles
21	Search Results Zoom Level	12
	+ ≡ Options ▾	Points ▾ Polylines ▾ Polygons ▾ Polygons1 ▾

Figure 13.2: View the online Google Sheet template that feeds data into the Leaflet Maps demo above.

GitHub, as described in Chapter 11. We have omitted many screenshots below that illustrate steps we previously covered, so if you get lost, go back to that chapter.

In the first part of the tutorial, you will create and publish your copies of our GitHub and Google Sheets templates:

- A) Copy the GitHub template and publish your version with GitHub Pages.
- B) File > Make a Copy of Google Sheet template, Share, and Publish.
- C) Paste your Google Sheet browser address in two places in your GitHub repo.
- D) Update your Google Sheet *Options* tab info and refresh your live map.

In the second part, you will learn how to upload and display different types of map data, such as points, polygons, and polylines, and to edit colors, icons, and images, by entering data into the linked Google Sheet and uploading files to your GitHub repo.

- E) Geocode locations and customize new markers in the Points tab.

- F) Remove or display point, polygon, or polylines data and legends.

In the third part, you have two options to finalize your map before widely sharing it online:

- G) Save each Google Sheets tab as a CSV file and upload to GitHub.
- OR
- H) Get your own Google Sheets API Key to insert into the code.

If any problems arise, see the Fix Common Mistakes section of the appendix.

A) Copy the GitHub template and publish your version with GitHub Pages

1. Open the GitHub code template in a new tab.
2. In the upper-right corner of the code template, sign in to your free GitHub account.
3. In the upper-right corner, click the green *Use this template* button to make a copy of the repository in your GitHub account. On the next screen, name your repo `leaflet-maps-with-google-sheets` or choose a different meaningful name in all lower-case. Click the *Create repository from template* button. Your copy of the repo will follow this format:

<https://github.com/USERNAME/leaflet-maps-with-google-sheets>

4. In your new copy of the code repo, click the upper-right *Settings* button and scroll way down to the GitHub Pages area. In the drop-down menu, change *Source* from *None* to *Main*, keep the default */(root)* setting, and press *Save* as shown in Figure 13.3. This step tells GitHub to publish a live version of your map on the public web, where anyone can access it in their browser, if they have the web address.
5. Scroll down to GitHub Pages section again, and copy the link to your published web site, which will appear in this format:

<https://USERNAME.github.io/leaflet-maps-with-google-sheets>

6. Scroll up to the top, and click on your repo name to go back to its main page.

GitHub Pages

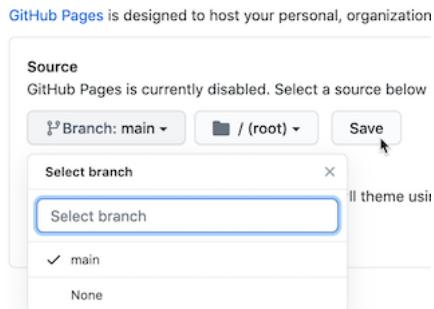


Figure 13.3: In *Settings*, go to *GitHub Pages*, switch the source from *None* to *Main*, and *Save*.

7. At the top level of your repo main page, click on README.md, and click the pencil icon to edit this file.
8. Delete the link to the *our* live site, as shown in Figure 13.4, and paste in the link to *your* published site. Scroll down to *Commit* your changes.

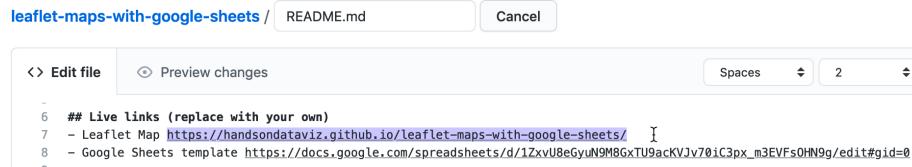


Figure 13.4: Edit your README file to replace the link to *our* site with the link to *your* site.

9. On your repo main page, right-click the link to open your live map in a new tab. *Be patient* during busy periods on GitHub, when your website may take up to 1 minute to appear for the first time.

B) File > Make a Copy of Google Sheet template, Share, and Publish

1. Open the Google Sheets template in a new tab.
2. Sign into your Google account, and select *File > Make a Copy* to save your own version of this Google Sheet on your Google Drive.
3. Click the blue *Share* button, and click *Change to anyone with the link*, then click *Done*. This publicly shares your map data, which is required to make this template work.

4. Go to *File > Publish to the Web*, and click the green *Publish* button to publish the entire document, so that the Leaflet code can read it. Then click the upper-right *X* symbol to close this window.
5. At the top of your browser, copy your Google Sheet address or URL (which usually ends in `...XYZ/edit#gid=0`). Do *NOT* copy the *Published to the web* address (which usually ends in `...XYZ/pubhtml`), as shown in Figure 13.5.

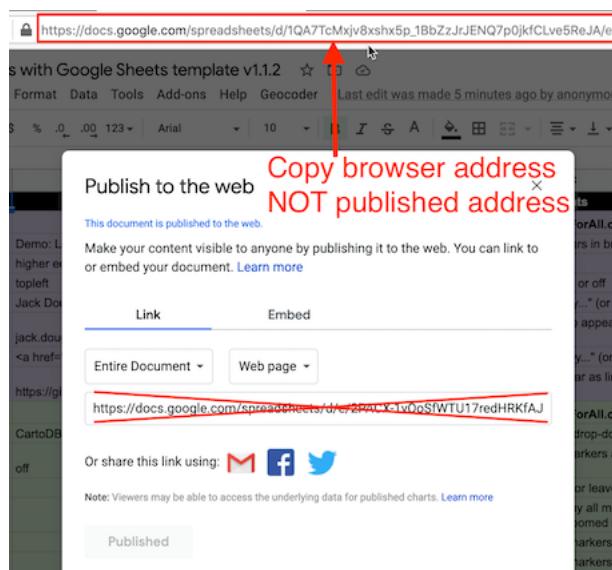


Figure 13.5: Copy the Google Sheet address at the top of the browser, NOT the *Published to the web* address.

C) Paste your Google Sheet browser address in two places in your GitHub repo

Our next task is to link your published Google Sheet to your Leaflet code in GitHub, so that it can pull your data from the Sheet to display on the map.

1. At the top of your GitHub repo, click to open the file named `google-doc-url.js`, and click the pencil symbol to edit it.
2. Paste *your* Google Sheet address or URL (which usually ends in `...XYZ/edit#gid=0`) to replace *our* existing URL, as shown in Figure 13.6. Be careful *NOT* to erase the single quotation marks or the semicolon at the end. Scroll down to *Commit* your changes. See separate instructions about the Google API key further below.

```

1 // paste in your published Google Sheets URL from the browser address bar
2 var googleDocURL = 'https://docs.google.com/spreadsheets/d/1ZxvU8eGyuN9M8GxTU9acKVJv70iC3px_m3EVFs0HN9g/edit#gid=0';
3
4 // insert your own Google Sheets API key from https://console.developers.google.com
5 var googleApiKey = 'AIzaSyBh9nKnVZm2RPeZa0ywCOxPAgJJfK87WhY';

```

Figure 13.6: Paste in *your* Google Sheet URL to replace *our* URL.

3. Also, let's paste your Google Sheet URL in second place to help you keep track of it. In your GitHub repo, click the README.md file to open it, click the pencil symbol to edit it, and paste *your* Google Sheet URL to replace *our* existing URL, as shown in Figure 13.7. Scroll down to *Commit* your changes.

```

6 ## Live links (replace with your own)
7 - Leaflet Map https://handsondataviz.github.io/leaflet-maps-with-google-sheets/
8 - Google Sheets template https://docs.google.com/spreadsheets/d/1ZxvU8eGyuN9M8GxTU9acKVJv70iC3px_m3EVFs0HN9g/edit#gid=0
9

```

Figure 13.7: Edit your README file to replace the link to *our* site with the link to *your* site.

Feel free to remove any other content on the README page that you do not wish to keep.

D) Update your Google Sheet *Options* tab info and refresh your live map

Now that your published Google Sheet is linked to your live map, go to the *Options* tab to update any of these items:

- Map Title
- Map Subtitle
- Author Name
- Author Email or Website
- Author Code Repo

Open the browser tab that displays your live map and refresh the page to see your changes. If your changes do not appear within a few seconds, see the Fix Common Problems section of the appendix.

E) Geocode locations and customize new markers in the Points tab

Now we can start to add new content to your map. In the *Points* tab of your Google Sheet, you'll see column headers to organize and display interactive markers on your map. Replace the demonstration data with your own, but do *not* delete or rename the column headers, since the Leaflet code looks for these specific names.

- Group: Create any labels to categorize groups of markers in your legend.
- Marker Icon: Insert any standard icon name such as `school` or `bus` that appears Font Awesome Icons, or leave blank for no icon inside the marker. To create your own custom icon, see further below.
- Marker Color: Insert any standard web color name such as `blue` or `darkblue`, or insert a web color code such as `#775307` or `rgba(200,100,0,0.5)`. See options at W3Schools Color Names.
- Icon Color: Set the color of the icon inside the marker. The default is `white`, which looks good inside darker-colored markers.
- Custom Size: Leave blank, unless you are creating your own custom icon further below.

The next set of columns include items that appear when users click on point markers:

- Name: Add a title to display in the marker pop-up window.
- Description: Add text to appear in the marker pop-up window. You may insert HTML tags to add line breaks (such as `
`), or to open external links in a new tab, such as `Visit W3Schools`. Learn about HTML syntax at W3Schools.
- Image: You have two options to display images. You can insert an external link to an image hosted by an online service (such as Flickr), as long as it begins with `https` (secure) and ends with either `.jpg` or `.png`. Or you can upload an image into the `media` subfolder in your GitHub repo, as shown in Figure 13.8, and enter the pathname in the Google Sheet in this format: `media/image.jpg` or `...png`.

Warning: Media file pathnames are case-sensitive, and we recommend using *all lowercase characters*, including the suffix ending. Also, since the code template automatically resizes images to fit, we recommend that you *reduce the size* of any images to 600x400 pixels or less prior to uploading, to make sure your map operates smoothly.

- Location, Latitude, Longitude: These place your markers at points on the map. Although the code template only requires Latitude and Longitude,

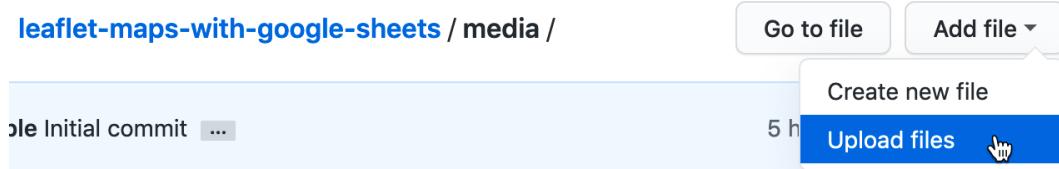


Figure 13.8: In GitHub, click to open the `media` folder and *Add file - Upload files*.

it's wise to paste an address or place name into the Location column as a reminder to correspond with the numerical coordinates. Use the Geocoding by SmartMonkey Add-on from Chapter 3 and select *Add-ons > Geocoding by SmartMonkey > Geocode Details* to create a new sheet with sample data and display results for three new columns: *Latitude*, *Longitude*, and *Address found*, as shown in Figure 13.9. Paste in your own address data and repeat the step above to geocode it, then copy and paste the results into your *Points* sheet.

A screenshot of a Google Sheets document. The menu bar shows 'File', 'Edit', 'View', 'Insert', 'Format', 'Data', 'Tools', 'Add-ons', 'Help', and 'Last edit was seconds ago'. The 'Add-ons' menu is open, showing 'Document add-ons' and 'Geocoding by SmartMonkey'. Under 'Geocoding by SmartMonkey', 'Geocode details' is selected and highlighted with a cursor icon. The main spreadsheet area contains a table with four columns: 'Address', 'Country', 'Latitude', and 'Longitude'. The 'Address found' column is also present. The table data is as follows:

Address	Country	Latitude	Longitude	Address found
Calle de la Electricidad, 49, 28918 Leganés, Madrid	es	40.3529951	-3.7976303	Calle de la Electricidad, 49, 28918 Leganés, Madrid
Av México 95-51 Del Carmen, Coyoacán, 04100 Ciudad de Mexico	mx	19.3551609	-99.1679037	Av México 95, Del Carmen, Coyoacán, 04100 Ciudad de Mexico
Av. Diaz Vélez 4900, Buenos Aires, Argentina	ar	-34.6087952	-58.4360388	Av. Diaz Vélez 4900, Buenos Aires, Argentina

The bottom of the screen shows the tabs 'Sheet1' and 'Geocoding Details'.

Figure 13.9: Select *Add-ons–Geocoding by SmartMonkey–Geocode Details* to display sample data with results for three new columns: *Latitude*, *Longitude*, and *Address found*.

Optional table of viewable markers: To display an interactive table at the bottom of your map, as shown in Figure 13.10. In the *Options* tab, set *Display Table* (cell B30) to *On*. You can also adjust the *Table Height*, and modify the display of *Table Columns* by entering the column headers, separated with commas.

TODO above: redo the screenshot after updating the map demo

Optional custom markers: To create your own custom marker, such as a thumbnail photo icon, use any image editing tool to reduce a photo to a square of 64 x 64 pixels. Save it in PNG format and choose a filename using all lower-case

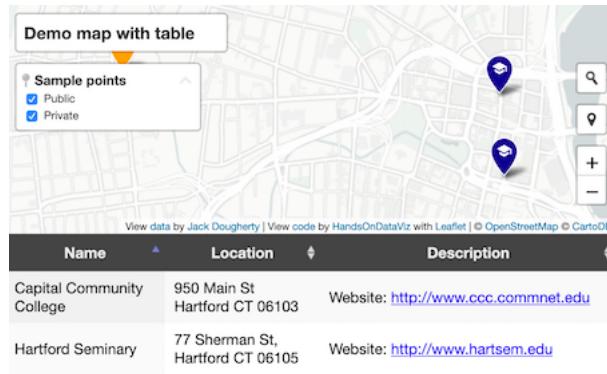


Figure 13.10: Optional: display interactive table of viewable markers at the bottom of your map.

characters with no spaces. Upload the image to the *media* folder in your GitHub repo as described above. In the Marker Icon column, enter the file pathname in this format: `media/image-thumbnail.png`. In the Custom Size column, set the dimensions to `64x64` or similar, such as `40x40` if desired.

TODO: Add screenshot of the sample custom marker here, once we finalize the new template

Open the browser tab that displays your live map and refresh the page to see your changes. If your changes do not appear within a few seconds, see the Fix Common Problems section of the appendix.

F) Remove or display point, polygon, or polylines data and legends

By default, the demo map displays three types of data—points, polygons, and polylines—and their legends. You can remove any of these from your map by modifying your linked Google Sheet:

To remove points:

- In the *Options* tab, set *Point Legend Position* (cell B27) to *Off* to hide it.
- In the *Points* tab, delete all rows of point data.

To remove polylines:

- In the *Options* tab, set *Polyline Legend Position* (cell B36) to *Off* to hide it.
- In the *Polylines* tab, delete all rows of polyline data.

To remove polygons:

- In the *Polygons* tab, set *Polygon Legend Position* (cell B4) to *Off* to hide it.
- Also in the *Polygons* tab, set *Polygon GeoJSON URL* (cell B6) to remove that data from your map.
- In the next tab *Polygons1*, use the tab drop-down menu to select *Delete* to remove the entire sheet.

You've already learned how to add more markers in the *Points* tab as described above. But if you wish to add new polygon or polyline data, you'll need to prepare those files in GeoJSON format using either the GeoJson.io tool tutorial or the MapShaper tool tutorial as described in Chapter 14.

After you've prepared your GeoJSON data, name the files using all lower-case characters and no spaces, and upload them into the `geojson` subfolder of your GitHub repo. Then update these settings in your linked Google Sheet:

To display polylines:

- In the *Options* tab, make sure *Polyline Legend Position* (cell B36) is visible by selecting *topleft* or a similar position.
- In the *Polylines* tab, enter the GeoJSON URL pathname to the file you uploaded to your GitHub repo, such as `geodata/lines.geojson`. Then insert a Display Name, Description, and Color.

To display polygons:

- In the *Polygons* tab, make sure *Polygon Legend Position* (cell B4) is visible by selecting *topleft* or a similar position.
- Also, in *Polygon GeoJSON URL* (cell B6) enter the pathname to the file you uploaded to your GitHub repo, such as `geodata/map.geojson`.
- Also, you can change the *Polygon Legend Title* (cell B3) and add an optional *Polygon Legend Icon* (cell B5).
- Also, edit the *Polygon Data* and *Color Settings* sections to modify the labels and ranges to align with the properties of your GeoJSON file. In the *Property Range Color Palette*, you can automatically select a color scheme from the ColorBrewer tool we described in the Map Design section of Chapter 8, or manually insert colors of your choice in the cell below.
- Read the *Hints* column in the *Polygons* sheet for tips on how to enter data.
- If you wish to display multiple polygon layers, use the *Polygons* tab drop-down menu to *Duplicate* the sheet, and name additional sheets in this format: *Polygons1*, *Polygons2*, etc.

TODO: after updating the map code, modify the text to match the demo file names

Finalize Your Map

Now you're ready to finalize your map. If you wish to share your map link with the public, read the options below and choose either step G *OR* step H.

Warning: We reserve the right to change *our* Google Sheets API key at any time, especially if other people overuse or abuse it. This means that you *must* finalize your map using either step G or H below before sharing it publicly, because it will *stop working* if we change our key.

G) Save each Google Sheets tab as a CSV file and upload to GitHub

If you have finished entering most of your data into your Google Sheets, downloading them into separate CSV files and uploading those into your GitHub repo is the *best* long-term preservation strategy. This approach keeps your map and data together in the same GitHub repo, and removes the risk that your map will break due to an interruption to Google services. Plus, you can still edit your map data. If this approach makes sense, follow these steps:

1. In your Google Sheets, go to each tab and select *File > Download* into CSV format, as shown in Figure 13.11, to create a separate file for each tab.
2. Shorten each file name as shown. The names must be exact, but they are *not* case-sensitive. Only the first file below is required, and others are optional, depending on your data.
 - Options.csv
 - Points.csv
 - Polylines.csv
 - Polygons.csv (If additional files, name them: Polygons1.csv, Polygons2.csv, etc.)
 - Notes.csv (or .txt) Recommended to keep any notes with your data, but not required.
3. In your GitHub repo, click the `csv` subfolder to open it, select *Add file > Upload files*, and upload all of the CSV files above into this subfolder, as shown in Figure 13.12. The Leaflet template code checks here first for data, and if it finds CSV files with the names above, it will pull the map data directly from them, instead of your Google Sheets. *Remember* that from this point forward, any edits in your Google Sheet will *no longer appear automatically* in your map.

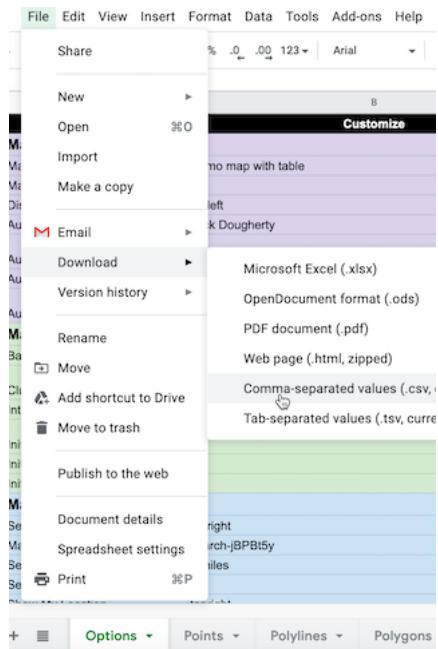


Figure 13.11: Download each Google Sheets tab as a separate CSV file.



Figure 13.12: Upload your map data files into the *csv* subfolder in GitHub.

4. If you wish to edit your map after uploading your CSV files, you have two options. You can make small edits directly to your CSV files by opening them in the GitHub web interface. Or you can make larger edits in the Google Sheet, and repeating the steps above to download them in CSV format and upload them to replace your existing files on GitHub.

H) Get your own Google Sheets API Key to insert into the code

As an alternative to step G, if you wish to continue to store your map data in your Google Sheets that is published online, go to the section of this chapter titled Get Your Own Google Sheets API Key, and insert it into the Leaflet map code as described, to avoid overusing our key. Google Sheets requires an API key to maintain reasonable usage limits on its service. You can get a free Google Sheets API key if you have a personal Google account, but *not* a Google Suite account provided by your school or business. TODO: confirm this detail

If problems arise, see the Fix Common Mistakes section of the appendix.

Leaflet Storymaps with Google Sheets

The Leaflet Storymaps code template is designed to show a point-by-point guided tour, with a scrolling narrative to display text, images, audio, video, and scanned map backgrounds, as shown in Figure 13.13. You enter all of your map data into a linked Google Sheet (or CSV file) or upload it into a GitHub repository, as shown in Figure 13.14. Although some other story map tools are easier to start using right away, such as the Knight Lab StoryMap and ESRI Story Maps, we do not recommend them because both lack *data portability*, meaning that you can't easily export any information you enter. By contrast, all of the information you add to the Leaflet Storymaps linked Google Sheet and GitHub repo can easily be exported and migrated to other platforms as visualization technology continues to evolve in the future, as we described when discussing how to choose tools wisely in Chapter 2. In addition, the Leaflet Storymaps template allows you to customize the appearance of your data, and to add more layers, such as historical maps and geographic boundaries, which you'll learn how to prepare in Chapter 14: Transform Your Map Data. Furthermore, the storymap design is responsive, so that it appears top-and-bottom on small screens and side-by-side on larger ones. Finally, the Leaflet template is built on flexible open-source software that's written primarily in JavaScript, a very common coding language for the web, so you can customize it further if you have skills or support from a developer.

TODO above: DECIDE how to address other storymap tools, with more details in editing notes

TODO BELOW: Create and insert a new version of the template and demonstrate all features (all media, tricks such as leaving blank chapters and locations to display multiple images of one location, etc.). TODO DISCUSS with Ilya - option to control responsive appearance?: <https://github.com/HandsOnDataViz/leaflet-storymaps-with-google-sheets/issues/75>

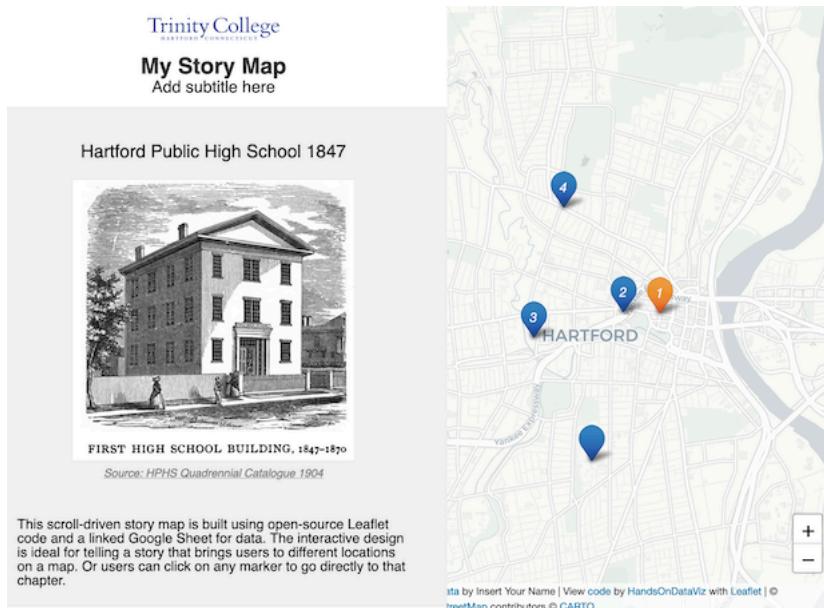


Figure 13.13: Explore the interactive Leaflet Storymaps with Google Sheets.

TODO: DECIDE if we should create gallery of examples with thumbnail images or links:

- <https://jhsgh.github.io/storymap/>
- <https://www.mappingtheuppermissouri.com/>
- <https://kensingtonremembers.org>
- <https://www.weneedtotalkabouttheborder.eu>
- https://www.laurentgontier.com/Storymap_Normandie/

Tutorial Outline

Before beginning this tutorial, you must have a Google Drive account, and know how to File > Make a Copy in Google Sheets, as described in Chapter 3. Also, you must have a GitHub account, and know how to Edit and Host Code with GitHub, as described in Chapter 11. We have omitted many screenshots below that illustrate steps we previously covered, so if you get lost, go back to that chapter.

Chapter	Description	Zoom	M
Hartford Public High School 1847	This scroll-driven story map is built using open-source Leaflet code and a linked Google Sheet for data. The interactive design is ideal for telling a story that brings users to different locations on a map. Or users can click on any marker to go directly to that chapter.		Numt
Hartford Public High School 1905	In the linked Google Sheet, Insert your point locations, zoom levels, narrative text, and map options (such as title, basemap style, and other settings). Click on the "View data" link at the bottom of this map to view the linked Google Sheet.	17	
Hartford Public High School 2011	Images can be uploaded a local subfolder, or pulled from an external URL (such as this photo on a Flickr server). The code automatically resizes images to fit the story map template, and you can modify the width and other settings in the Google Sheet Options tab.	17	
Jackie McLean & The Artists Collective	Along with images, Storymaps supports audio files and YouTube videos. Audio files must be mp3 (recommended), ogg, or wav: the map will generate an <audio> tag for them instead of .	17	
Bradley Airport <i><i class="fas fa-plane"></i></i> , Windsor Locks	The Leaflet 1.x code includes a 'fly to' feature that animates traveling from one map point to the next, which is ideal for transporting users across long distances.		14
	Create your own story map using the step-by-step tutorial in		

Figure 13.14: View the online Google Sheet template that feeds data into the Leaflet Storymaps demo above.

In the first part of the tutorial, you will create and publish your copies of our GitHub and Google Sheets templates:

- A) Copy the GitHub template and publish your version with GitHub Pages.
- B) File > Make a Copy of Google Sheet template, Share, and Publish.
- C) Paste your Google Sheet browser address in two places in your GitHub repo.
- D) Update your Google Sheet *Options* tab info and refresh your live map.

In the second part, you will learn how to geocode and customize point data in the linked Google Sheet, upload images and other map data to your GitHub repo, and add scanned background map layers if desired.

- E) Add text, media, markers, and geocode locations in the Google Sheet *Chapters* tab.
- F) Optional: Add georeferenced historical map image or GeoJSON overlays

In the third part, you have two options to finalize your story map before widely sharing it online:

- G) Save each Google Sheets tab as a CSV file and upload to GitHub.

- OR
- H) Get your own Google Sheets API Key to insert into the code.

If any problems arise, see the Fix Common Mistakes section of the appendix.

A) Copy the GitHub template and publish your version with GitHub Pages

1. Open the GitHub code template in a new tab.
2. In the upper-right corner of the code template, sign in to your free GitHub account.
3. In the upper-right corner, click the green *Use this template* button to make a copy of the repository in your GitHub account. On the next screen, name your repo `leaflet-storymaps-with-google-sheets` or choose a different meaningful name in all lower-case. Click the *Create repository from template* button. Your copy of the repo will follow this format:

<https://github.com/USERNAME/leaflet-storymaps-with-google-sheets>

4. In your new copy of the code repo, click the upper-right *Settings* button and scroll way down to the GitHub Pages area. In the drop-down menu, change *Source* from *None* to *Main*, keep the default */(root)* setting, and press *Save* as shown in Figure 13.15. This step tells GitHub to publish a live version of your map on the public web, where anyone can access it in their browser, if they have the web address.

GitHub Pages

[GitHub Pages](#) is designed to host your personal, organization

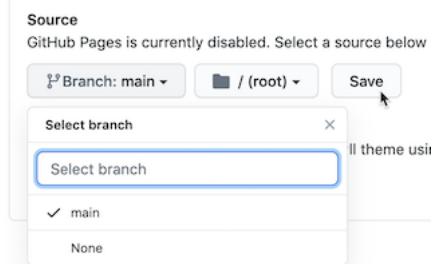


Figure 13.15: In *Settings*, go to *GitHub Pages*, switch the source from *None* to *Main*, and *Save*.

5. Scroll down to GitHub Pages section again, and copy the link to your published web site, which will appear in this format:

`https://USERNAME.github.io/leaflet-maps-with-google-sheets`

6. Scroll up to the top, and click on your repo name to go back to its main page.
7. At the top level of your repo main page, click on README.md, and click the pencil icon to edit this file.
8. Delete the link to the *our* live site, as shown in Figure 13.16, and paste in the link to *your* published site. Scroll down to *Commit* your changes.



The screenshot shows a GitHub repository's README.md file being edited. The code contains a comment indicating where to replace live links:

```

9  ## Live links (replace with your own)
10 - Leaflet Map https://handsondataviz.github.io/leaflet-storymaps-with-google-sheets/ I
11 - Google Sheets template https://docs.google.com/spreadsheets/d/1A06XHL\_0JafWZF4KEejkdDNqfuZWUK3SLNl06Mj1RFM/edit#gid=0
12

```

Figure 13.16: Edit your README file to replace the link to *our* site with the link to *your* site.

9. On your repo main page, right-click the link to open your live map in a new tab. *Be patient* during busy periods on GitHub, when your website may take up to 1 minute to appear for the first time.

B) File > Make a Copy of Google Sheet template, Share, and Publish

1. Open the Google Sheets template in a new tab.
2. Sign into your Google account, and select *File > Make a Copy* to save your own version of this Google Sheet on your Google Drive.
3. Click the blue *Share* button, and click *Change to anyone with the link*, then click *Done*. This publicly shares your map data, which is required to make this template work.
4. Go to *File > Publish to the Web*, and click the green *Publish* button to publish the entire document, so that the Leaflet code can read it. Then click the upper-right *X* symbol to close this window.
5. At the top of your browser, copy your Google Sheet address or URL (which usually ends in ...XYZ/edit#gid=0). Do *NOT* copy the *Published to the web* address (which usually ends in ...XYZ/pubhtml), as shown in Figure 13.17.

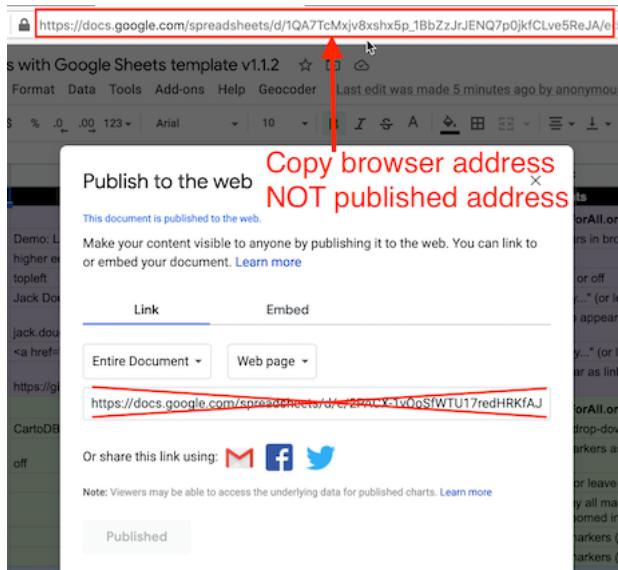


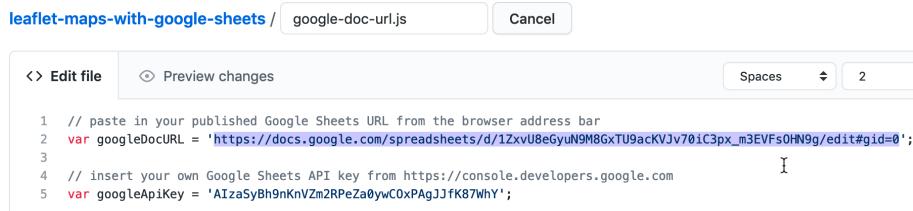
Figure 13.17: Copy the Google Sheet address at the top of the browser, NOT the *Publish to the web* address.

C) Paste your Google Sheet browser address in two places in your GitHub repo

Our next task is to link your published Google Sheet to your Leaflet code in GitHub, so that it can pull your data from the Sheet to display on the map.

1. At the top of your GitHub repo, click to open the file named `google-doc-url.js`, and click the pencil symbol to edit it.
2. Paste *your* Google Sheet address or URL (which usually ends in `...XYZ/edit#gid=0`) to replace *our* existing URL, as shown in Figure 13.18. Be careful *NOT* to erase the single quotation marks or the semicolon at the end. Scroll down to *Commit* your changes. See separate instructions about the Google API key further below.
3. Also, let's paste your Google Sheet URL in second place to help you keep track of it. In your GitHub repo, click the `README.md` file to open it, click the pencil symbol to edit it, and paste *your* Google Sheet URL to replace *our* existing URL, as shown in Figure 13.19. Scroll down to *Commit* your changes.

Feel free to remove any other content on the `README` page that you do not wish to keep.



```
// paste in your published Google Sheets URL from the browser address bar
var googleDocURL = 'https://docs.google.com/spreadsheets/d/1ZxvU8eGyuN9M8GxTU9acKVJv70iC3px_m3EVFs0HN9g/edit#gid=0';
// insert your own Google Sheets API key from https://console.developers.google.com
var googleApiKey = 'AIzaSyBh9nKnVZm2RPeZa0ywC0xPAgJJfK87WhY';
```

Figure 13.18: Paste in *your* Google Sheet URL to replace *our* URL.



```
## Live links (replace with your own)
- Leaflet Map https://handsondataviz.github.io/leaflet-storymaps-with-google-sheets/
- Google Sheets template https://docs.google.com/spreadsheets/d/1A06XHL_0JafwZF4KEejkdDnfuzhUK35lnl06Mj1RFM/edit#gid=0
```

Figure 13.19: Edit your README file to replace the link to *our* site with the link to *your* site.

D) Update your Google Sheet *Options* tab info and refresh your live map

Now that your published Google Sheet is linked to your live map, go to the *Options* tab to update any of these items:

- Storymap Title
- Storymap Subtitle – with code for downward arrow:
<small>Scroll down <i class='fa fa-chevron-down'></i></small>
- Author Name
- Author Email or Website
- Author GitHub Repo Link

Open the browser tab that displays your live map and refresh the page to see your changes. If your changes do not appear within a few seconds, see the Fix Common Problems section of the appendix.

E) Add text, media, markers, and geocode locations in the Chapters tab.

Now we can start to add new content to your map. In the *Chapters* tab of your Google Sheet, you'll see column headers to organize and display interactive markers on your map. Replace the demonstration data with your own, but do *not* delete or rename the column headers, since the Leaflet code looks for these specific names.

- Chapter: The title appearing at the top of each section in the scrolling narrative.
- Media Link: You have several options to display either an image, audio, or video in each chapter. For images, you can insert an external link to an online service (such as Flickr), as long as it begins with `https` (secure) and ends with either `.jpg` or `.png`. You can also insert a YouTube video link. Or you can upload an image file into the `media` subfolder in your GitHub repo, as shown in Figure 13.20, and enter the pathname in the Google Sheet in this format: `media/your-file-name.jpg` or `...png`. Similarly, you can upload an audio file in `.mp3` (recommended) or `.ogg` or `.wav` format.

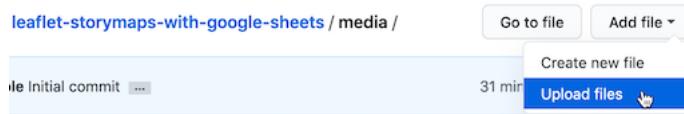


Figure 13.20: In GitHub, click to open the `media` folder and *Add file - Upload files*.

Warning: Media file pathnames are case-sensitive, and we recommend using *all lowercase characters*, including the suffix ending. Also, since the code template automatically resizes images to fit, we recommend that you *reduce the size* of any images to 600x400 pixels or less prior to uploading, to make sure your storymap scrolls quickly.

Tip: You can display multiple images for one location by creating a series of rows, but only list the *Chapter* and *Location* information in the first row of the series, and leave it blank for the others.

- Media Credit: To display text about the origin of the media, such as “Source:...”.
- Media Credit Link: Add a direct link to the source info in the Media Credit text above.
- Description: Designed to display about a paragraph or less of text for the Chapter. You may insert HTML tags to add line breaks (such as `
`), or to open external links in a new tab, such as `Visit W3Schools`. Learn about HTML syntax at W3Schools.
- Zoom: Set a low number to see continents, or a high number for city streets, typically on a scale from 1 to 18.

- Marker: Select either **Numbered** (the default) or **Plain** or **Hidden**. The latter works best when assigning several chapters to one location (to avoid stacking markers on top of each other) or when zooming out for a broader view (without highlighting one specific location).
- Marker Color: Insert any standard web color name such as **blue** or **darkblue**, or insert a web color code such as **#775307** or **rgba(200,100,0,0.5)**. See options at W3Schools Color Names.
- Location, Latitude, Longitude: These place your markers at points on the map. Although the code template only requires Latitude and Longitude, it's wise to paste an address or place name into the Location column as a reminder to correspond with the numerical coordinates. Use the Geocoding by SmartMonkey Add-on from Chapter 3 and select *Add-ons > Geocoding by SmartMonkey > Geocode Details* to create a new sheet with sample data and display results for three new columns: *Latitude*, *Longitude*, and *Address found*, as shown in Figure 13.21. Paste in your own address data and repeat the step above to geocode it, then copy and paste the results into your *Points* sheet.

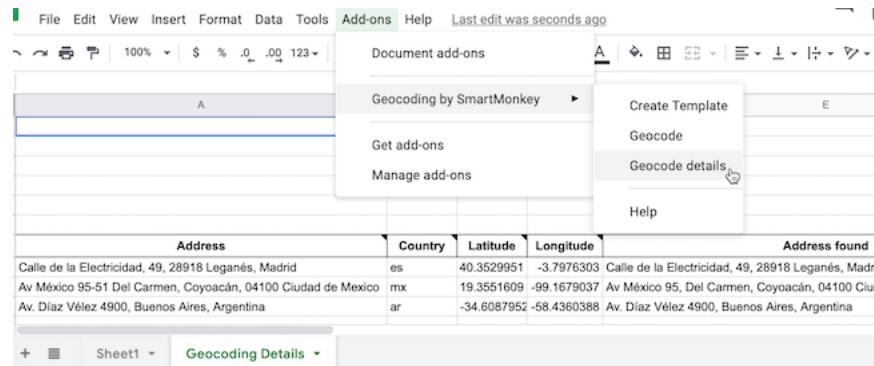


Figure 13.21: Select *Add-ons–Geocoding by SmartMonkey–Geocode Details* to display sample data with results for three new columns: *Latitude*, *Longitude*, and *Address found*.

Open the browser tab that displays your live map and refresh the page to see your changes. If your changes do not appear within a few seconds, see the Fix Common Problems section of the appendix.

F) Optional: Add historical map image or GeoJSON overlays

The code template allows you to enrich your story by placing two different types of layers on top of the background map: georeferenced map images (such as a historical map) and GeoJSON geodata (such as a pathway, boundary lines,

or a color-coded choropleth map). You can add both types of layers to specific chapters or the entire story. Also, you can adjust the transparency level to reveal or hide the present-day background map. To prepare both types of layers, you will need to jump ahead to Chapter 14: Transform Your Map Data, but here we'll explain the steps to insert them in your storymap template.

To add a historical map overlay to one or more story map chapters, it must be *georeferenced* (also called georectified), which means to digitally align the static map image with a more precise present-day interactive map. If you have a high-quality static image of a historical map, use the Mapwarper tool as described in Chapter 14 to align several known points with those on a present-day interactive map. Mapwarper transforms the static map image into interactive map tiles, and publicly hosts them online with a link in Google/OpenStreetMap format, similar to <https://mapwarper.net/maps/tile/14781/{z}/{x}/{y}.png>. Or you can search for historical maps that have already been georeferenced and transformed into tiles (and volunteer for crowdsourcing efforts to align maps) on platforms such as Mapwarper and the New York Public Library Mapwarper. Although map tile links are *not* viewable in a normal browser, they can be displayed by the Leaflet Storymaps code. Enter the tile link and your desired transparency level into the Overlay columns in the *Chapters* tab of your Google Sheet template, as shown in Figure 13.22.

- Overlay: Enter a map tile link in Google/OpenStreetMap format, similar to the sample above.
- Overlay Transparency: Enter a number from 0 (transparent) to 1 (opaque). The default is 0.7.



Figure 13.22: Enter map tile link and transparency level into the Google Sheet template (on left) to display it in one or more storymap chapters (on right).

TODO above: Update the image after updating the demo storymap. Confirm

if similar format works with georeferenced items from David Rumsey Map Collection, and check other platforms. Also, decide whether to add this line: Use secure https (ending with s) when available.

To add a visible path, geographic boundaries, or a filled choropleth map to your story, consider adding a GeoJSON data layer to one or more chapters. Read about GeoJSON and geospatial data formats in Chapter 14, where you can also learn how to find existing GeoJSON boundary files, or draw or edit your own geodata with the GeoJson.io tool or Mapshaper tool. We recommend that you name your GeoJSON files in lower-case characters with no spaces. Upload the file to your GitHub repository by opening the `geojson` folder and selecting *Add file - Upload files*. In your Google Sheet template, enter the pathname in the *GeoJSON Overlay* column in this format: `geojson/your-file-name.geojson`, as shown in Figure 13.23.

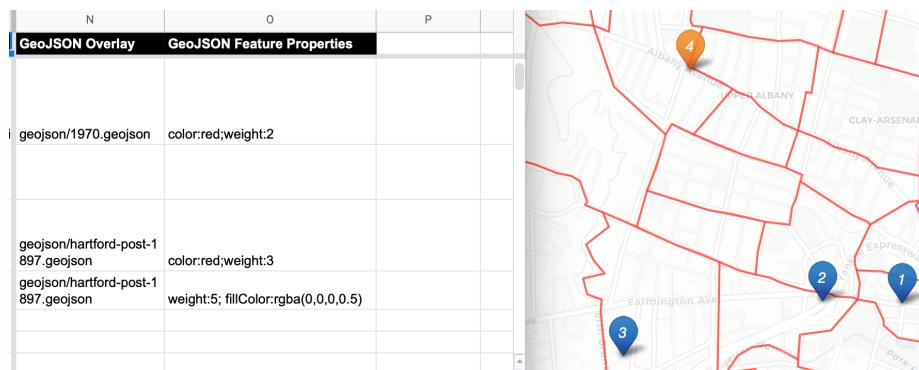


Figure 13.23: Enter the pathname in the *GeoJSON Overlay* column (on left) to display it in one or more storymap chapters (on right).

TODO above: Update demo storymap and update images to match. TODO below: DISCUSS with Ilya if we are using a standardized properties list re: <https://github.com/HandsOnDataViz/leaflet-storymaps-with-google-sheets/issues/74>. Convert list to a small table, to make it easier to see definitions and storymap default values?

When you create or edit GeoJSON data with a tool like GeoJson.io, you can directly edit its feature properties. If you wish to display the same properties you assigned to your GeoJSON file in your storymap, we recommend naming them as follows:

- weight (width of line; storymap template default is 1px)
- color (of line; default is white???)
- opacity (of line; default is ???)
- fillColor (of polygon; default is silver)
- fillOpacity (of polygon; default is 0.7)

Or you can enter properties and CSS codes in the *GeoJSON Feature Properties* template column, in this format, separated by semicolons, with no quotation marks required: `weight:3;color:red;opacity:1;fillColor:orange;fillOpacity:0.9.` You can assign colors with standard names, hex codes, or RGBA values as described in the W3Schools Colors Picker.

TODO DISCUSS with Ilya whether to mention additional customizations that are possible here (and are all of them also available in Leaflet Maps with Google Sheets?)

- insert logo – see Options tab of Google Sheets
- insert Google Analytics tracking ID – see how to get one.... and insert in Options tab of Google Sheets
- to insert a horizontal divider line in Description, copy and paste this text to your Google Sheets template: `` (is this necessary here? ...and make sure that the single-quote marks are not changed into curly apostrophes).
- to adjust title size and font: in GitHub, go to `css/styles.css` file, scroll down to (specify lines?), and adjust font-size values...
- About the responsive design: On larger screens, the scrolling narrative and map appear side-by-side. On smaller screens where the width is less than 768 pixels, it automatically switches to a top-and-bottom display.

Finalize Your Story Map

Now you're ready to finalize your map. If you wish to share your map link with the public, read the options below and choose either step G *OR* step H.

Warning: We reserve the right to change *our* Google Sheets API key at any time, especially if other people overuse or abuse it. This means that you *must* finalize your map using either step G or H below before sharing it publicly, because it will *stop working* if we change our key.

G) Save each Google Sheets tab as a CSV file and upload to GitHub

If you have finished entering most of your data into your Google Sheets, downloading them into separate CSV files and uploading those into your GitHub repo is the *best* long-term preservation strategy. This approach keeps your map and data together in the same GitHub repo, and removes the risk that your map will break due to an interruption to Google services. Plus, you can still edit your map data. If this approach makes sense, follow these steps:

1. In your Google Sheets, go to each tab and select *File > Download* into CSV format, as shown in Figure 13.24, to create a separate file for each tab.

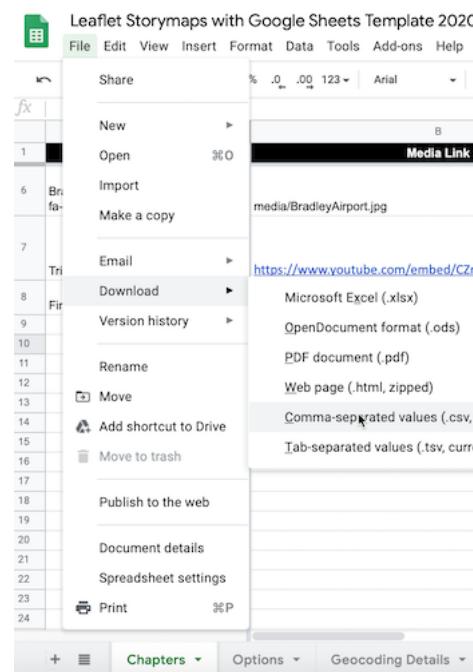


Figure 13.24: Download each Google Sheets tab as a separate CSV file.

2. Shorten each file name as shown. The names must be exact, but they are *not* case-sensitive. The first two files below are required, and others are optional.
 - Chapters.csv
 - Options.csv
 - Notes.csv (or .txt) Recommended to keep any notes with your data, but not required.

3. In your GitHub repo, click the `csv` subfolder to open it, select *Add file > Upload files*, and upload all of the CSV files above into this subfolder, as shown in Figure 13.25. The Leaflet template code checks here first for data, and if it finds CSV files with the names above, it will pull the map data directly from them, instead of your Google Sheets. *Remember* that from this point forward, any edits in your Google Sheet will *no longer appear automatically* in your map.



Figure 13.25: Upload your map data files into the `csv` subfolder in GitHub.

4. If you wish to edit your map after uploading your CSV files, you have two options. You can make small edits directly to your CSV files by opening them in the GitHub web interface. Or you can make larger edits in the Google Sheet, and repeating the steps above to download them in CSV format and upload them to replace your existing files on GitHub.

H) Get your own Google Sheets API Key to insert into the code

As an alternative to step G, if you wish to continue to store your map data in your Google Sheets that is published online, go to the section of this chapter titled Get Your Own Google Sheets API Key, and insert it into the Leaflet map code as described, to avoid overusing our key. Google Sheets requires an API key to maintain reasonable usage limits on its service. You can get a free Google Sheets API key if you have a personal Google account, but *not* a Google Suite account provided by your school or business. TODO: confirm this detail

If problems arise, see the Fix Common Mistakes section of the appendix.

Get Your Google Sheets API Key

After you've created your own version of Leaflet Maps with Google Sheets or Leaflet Storymaps with Google Sheets, there are two ways to finalize your map, as described above: either save your Google Sheet tabs in CSV format, or get your own Google Sheets API key and paste it into your Leaflet code on GitHub. You'll learn about the latter method in this section.

Beginning in January 2021, Google Sheets version 4 requires a API (application programming interface) key to allow code to read your data, in order to maintain reasonable limits on use of its services. For Google Sheets, the limit is 500 requests per 100 seconds per project, and 100 requests per 100 seconds per user. There is no daily usage limit.

You can get your own free Google Sheets API key by following the steps below. Overall, you will create and name your Google Cloud project, enable the Google Sheets API to allow a computer to read data from your Google Sheet, copy your new API key, and paste it into the Leaflet code in place of our key.

Before you begin:

- You need a personal Google account, *not* a Google Suite account issued by your school or business.
 - This tutorial presumes that you have already have completed the Leaflet Maps with Google Sheets or Leaflet Storymaps with Google Sheets template above, and wish to finalize your map.
 - If you already created a Google Sheets API key for one template above, you can also use that key for another template.
1. Go to the Google Developers Console at <https://console.developers.google.com/> and log in to your Google account. Google may ask you to identify your country and agree to its terms of service.
 2. Click on *Create a Project* on the opening screen, as shown in Figure 13.26. Or alternatively, go to the upper-left drop-down menu to *Select a project > New project*.

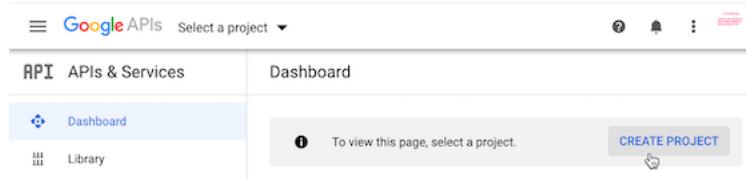


Figure 13.26: Select *Create a Project* or use the menu to select a new project.

3. In the next screen, give your new project a meaningful short name to remind you of its purpose, such as `handsondataviz`. You do not need to create an organization or parent folder. Then click *Create*, as shown in Figure 13.27.

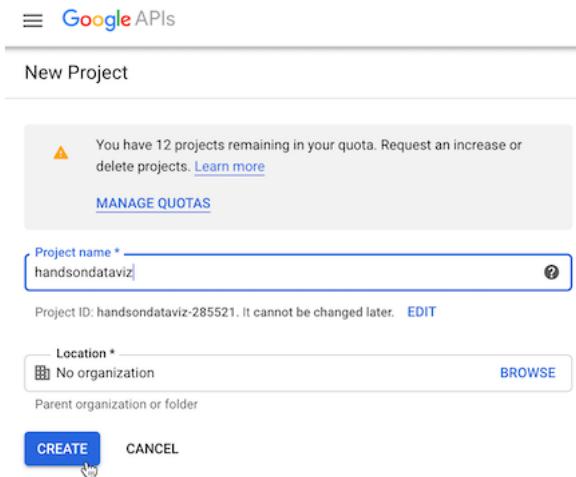


Figure 13.27: Give your project a meaningful short name.

4. In the next screen, press the *+ Enable APIs and Services* at the top of the menu, as shown in Figure 13.28. Make sure that your new project name appears near the top.

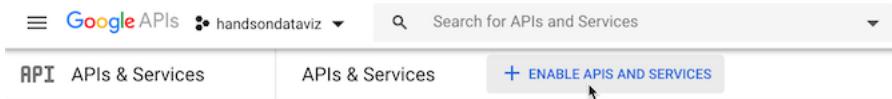


Figure 13.28: Press the *+ Enable APIs and Services* button.

5. In the next screen, enter *Google Sheets* into the search bar, and select this result, as shown in Figure 13.29.
6. In the next screen, select the *Enable* button to turn on the Google Sheets API for your project, as shown in Figure 13.30.
7. In the left sidebar menu, click *Credentials*, then click *+ Create Credentials* and select *API key*, as shown in Figure 13.31.

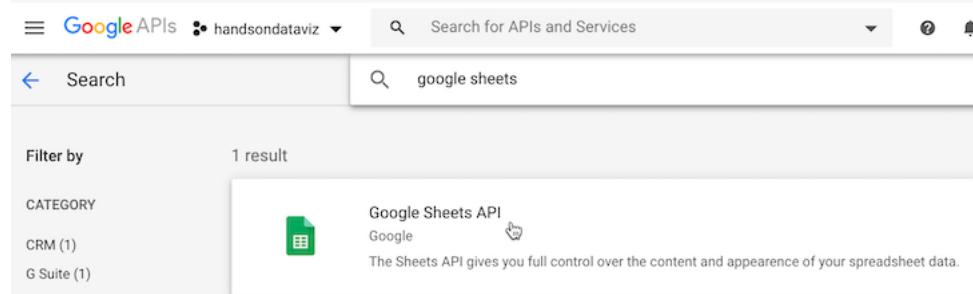


Figure 13.29: Search for *Google Sheets* and select this result.

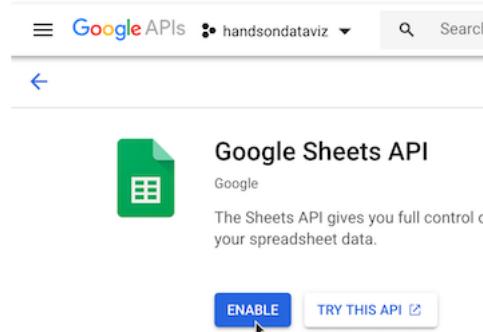


Figure 13.30: Select the *Enable* button for Google Sheets API.

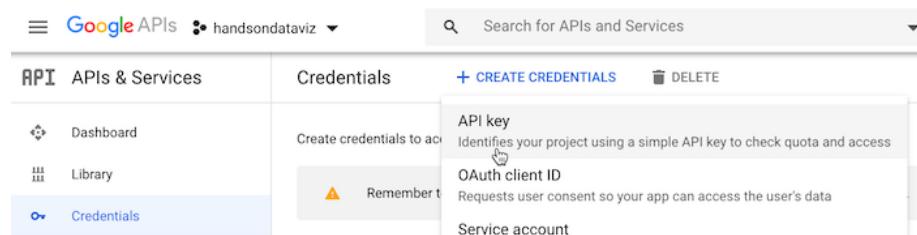


Figure 13.31: Select *Credentials - Create Credentials - API key*.

8. In the next screen, the console will generate your API key. Copy it, then press *Restrict key*, as shown in Figure 13.32.

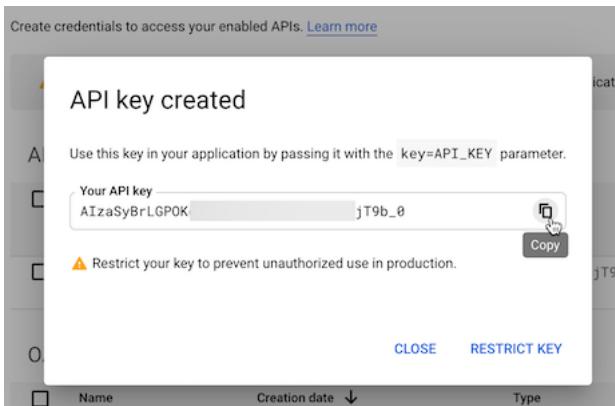


Figure 13.32: Copy your API key and press *Restrict key*.

9. In the new window, under *API restrictions*, choose the *Restrict key* radio button. In the dropdown that appears, choose *Google Sheets API*, then click *Save*, as shown in Figure 13.33.
10. In your Leaflet map code on your GitHub repo, open the `google-doc-url.js` file, click the pencil symbol to edit it, and paste in *your* Google Sheets API key to replace *our* key, as shown in Figure 13.34. Be careful not to erase the single-quote marks or the semicolon. Scroll down to *Commit* your changes.

You might receive a notification from GitHub stating that you have an exposed API key, but don't worry. This key can only be used with Google Sheets, you received it for free, and you did not attach any billing information to it, so Google cannot charge you for its use.

TODO above: Test our instructions on a new Google account to fix what reviewer Erica H. described here: I did notice when you get to Step 7, they ask you what kind of credentials you'll need, where you'll be calling the API from, and the kind of data you will be accessing. See screenshots attached. You might want to add a note about this in your book chapter in case folks don't know what to select if they are taken to this step automatically. I ended up clicking refresh accidentally and then I saw the + Create Credentials>API key dropdown menu as shown in your chapter instructions and I was able to generate my Google API key without entering those credential specifications

Now that you've learned how to create a Google Sheets API key to use with Leaflet Maps with Google Sheets or Leaflet Storymaps with Google Sheets, in the next sections you'll learn more about other types of Leaflet map templates.

Google APIs handsondataviz ▾

Search for APIs and Services

API APIs & Services

Dashboard

Library

Credentials

OAuth consent screen

Domain verification

Page usage agreements

← Restrict and rename API key

REGENERATE KEY

Key restrictions

This key is unrestricted. Restrictions help prevent unauthorized use and quota theft. [Learn more](#)

Application restrictions

An application restriction controls which websites, IP addresses, or applications can use your API key. You can set one application restriction per key.

None

HTTP referrers (web sites)

IP addresses (web servers, cron jobs, etc.)

Android apps

iOS apps

API restrictions

API restrictions specify the enabled APIs that this key can call

Don't restrict key
This key can call any API

Restrict key

Type to filter

S G

Google Sheets API

Note: It may take up to 5 minutes for settings to take effect

Figure 13.33: Choose API restrictions - Restrict key - Google Sheets API

Executable File | 2 lines (2 sloc) | 179 Bytes

Raw Blame   

```
1 var googleDocURL = 'https://docs.google.com/spreadsheets/d/1fH16jlqFWUhUzN-oBOK4ZRVCeqHQLSf6XUsqquiWcg/edit#gid=0';
2 var googleApiKey = 'AIzaSyBh...AqJJFK87WhY';
```

Figure 13.34: Paste in *your* Google Sheets API key to replace *our* key.

Leaflet Maps with CSV Data

Figure 13.35 shows a simple point map of some colleges and universities in Connecticut. But instead of individually creating markers in JavaScript using Leaflet's `L.marker()` function, the point data is stored in a local CSV file (`data.csv`) that is easy to modify in Excel or any text editor. Each time the map is loaded by the browser, point data from the CSV file is read and markers are generated “on the fly”.

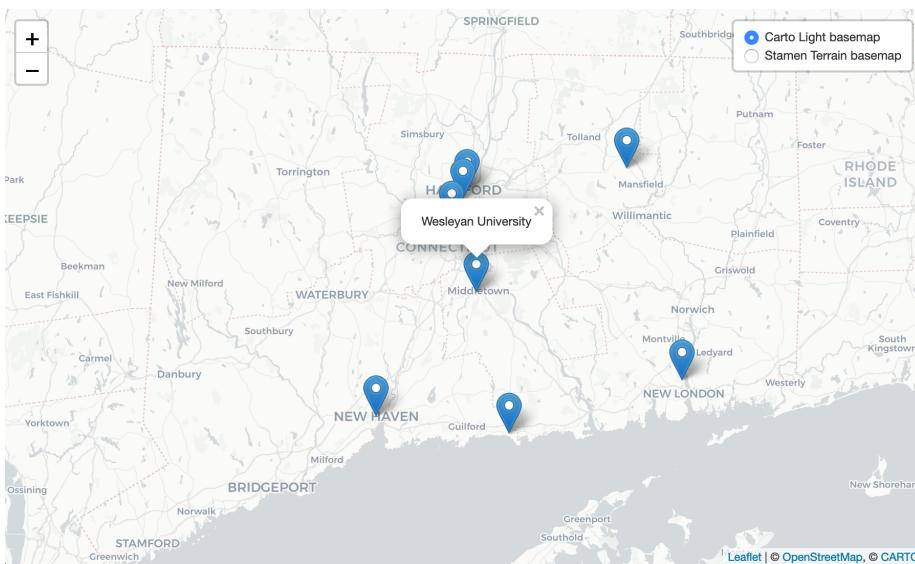


Figure 13.35: Explore the interactive Leaflet point map with CSV data.

You can adapt this template to create your own point map by following these instructions:

1. Visit the GitHub repo that stores the code for this template. Make sure you are logged in, and press *Use this template button* to create a copy of this repository in your own GitHub account.
2. Put your point data inside `data.csv`. The only relevant columns that will be read by the template are *Latitude*, *Longitude*, and *Title*. The first two determine the location of the marker, and the last one is displayed in a popup. The order of columns does not matter. There can be other columns in the dataset, but they will be ignored.

Your data can look like the following:

```
Title,Latitude,Longitude
```

```
Trinity College,41.745167,-72.69263
Wesleyan University,41.55709,-72.65691
```

3. Depending on the geography of your points, you will want to change the default position of the map on start. In `index.html`, find the `<script>` tag, and edit the following chunk of code:

```
var map = L.map('map', {
  center: [41.57, -72.69], // Default latitude and longitude on start
  zoom: 9, // Between 1 and 18; decrease to zoom out, increase to zoom in
  scrollWheelZoom: false
});
```

We used default Leaflet markers for code simplicity, but you may want to use custom icons instead. The code snippet below can give you an idea how to

```
var marker = L.marker([row.Latitude, row.Longitude], {
  opacity: 1,
  // Customize your icon
  icon: L.icon({
    iconUrl: 'path/to/your/icon.png',
    iconSize: [40, 60]
  })
}).bindPopup(row.Title);
```

For more information on icon customization, see this helpful Leaflet example.

Leaflet Heatmap Points with CSV Data

Heatmaps turn individual points into hotspots or clusters, allowing viewers to explore spatial distributions of events, such as areas of high and low population density or incidents of crime. Figure 13.36 shows an interactive heatmap of bike theft locations in London between January and July 2020. The underlying data are coordinate locations for each reported bike theft, which `Leaflet.heat` plugin transforms into areas of various densities. Red shows areas of highest density, or areas where bike theft appeared most often. When you zoom in, areas are re-calculated into more distinct clusters.

You can adapt the code we used for this London heatmap to create your own.

1. Visit the GitHub repository with our code, make sure you are logged in, and click *Use this template* button to make a personal copy of this repo.
2. Modify map's title and description inside `index.html`.



Figure 13.36: Explore the interactive Leaflet Heatmap.

3. Place your point coordinates data inside `data.csv`. Do *not* insert any column headers. Instead of the traditional order, you must write them in *latitude,longitude* (or *y,x*) order, one pair per line, like this:

```
51.506585,-0.139387
51.505467,-0.14655
51.507758,-0.141284
```

4. Depending on your data density, you might want to tweak `radius` and `blur` parameters inside the `<script>` tag of `index.html`:

```
var heat = L.heatLayer(data, {
  radius: 25,
  blur: 15,
})
```

Edit the following chunk of code to set your map's default position and zoom level:

```
var map = L.map('map', {
  center: [51.5, -0.1], // Initial map center
  zoom: 10, // Initial zoom level
})
```

If for some reason you cannot see clusters, make sure your point data is represented in `latitude,longitude` order, not the other way around. If you have few points, try increasing the value of `radius` property of `L.heatLayer`.

Leaflet Searchable Point Map

Figure 13.37 shows a powerful Leaflet template of a searchable and filterable point map, which draws from a CSV data file, developed by Derek Eder from DataMade. This map allows you to show points of interest, filter them by using *Search by name* functionality, and show them as a list instead of points on a map. In addition, the *About* page gives you plenty of space to talk about your map.

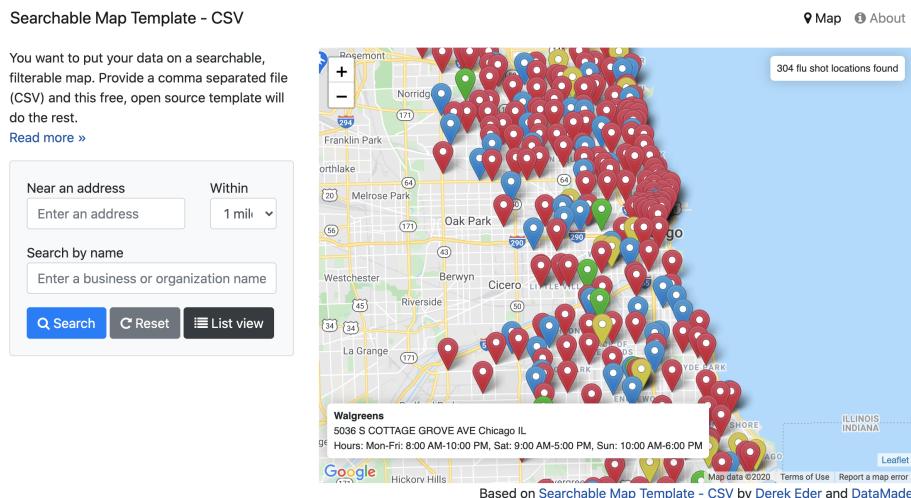


Figure 13.37: Explore the interactive Searchable Map template.

This template uses Leaflet.js in combination with Google Maps API to perform address search.

To begin using the template for your own project, visit the template's GitHub page, and fork it so that you get your own copy (see Edit and Host Code with GitHub chapter to remind yourself about forks).

Step 1: Prepare your data

This template will work with data in csv and geojson formats. If you have an `.xls` or `.xlsx` Excel file, save it in `csv` comma-separated values format with any spreadsheet tool. The `csv` file must have a `latitude` column and `longitude`

column and all rows must be geocoded. If you only have street-address or location data, learn how to geocode it in chapter 3.

Step 2: Download and edit this template

1. Download or clone this project and fire up your text editor of choice. Open up `/js/map.js` and set your map options in the `SearchableMapLib.initialize` function:
 - `map_centroid` - the lat/long you want your map to center on (find yours here)
 - `filePath` - Path to your map data file. This file needs to be in csv or geojson format and placed in the `data` folder. This file's first line must be the header, and it must have a latitude column and longitude column.
 - `fileType` - Set if you are loading in a `csv` or `geojson` file
2. Edit the templates in the `templates` folder for how you want your data displayed. These templates use EJS, which allows the display of your variables with HTML, as well as conditional logic. Documentation is here.
 - `/templates/hover.ejs` - template for when you hover over a dot on the map
 - `/templates/popup.ejs` - template for when a dot on the map is clicked
 - `/templates/table-row.ejs` - template for each row in the list view
3. Remove the custom filters and add your own.
 - `index.html` - custom HTML for filters starts around line 112
 - `/js/searchable_map_lib.js` - logic for custom filters starts around line 265

Step 3: Publish your map

1. Before you publish, you'll need to get a free Google Maps API key, which is similar but different from the Get Your Google Sheets API Key section in this chapter. Replace the Google Maps API key on this line of `index.html` with yours: `<script type="text/javascript" src="https://maps.google.com/maps/api/js?libraries=places&key=[YOUR KEY HERE]"></script>`
2. Upload this map and all the supporting files and folders to your site. This map requires no back-end code, so any host will work, such as GitHub Pages as described in Chapter 11, or Netlify, or your own web server.

Leaflet Maps with Open Data APIs

Leaflet maps can pull and display data from various open data repositories using APIs. Figure 13.38 shows an interactive map of North Dakota counties, colored by population density, with hospitals and EMS stations locations. Hospital information is pulled directly from Medicare.org Socrata database. County boundaries and population density are pulled from North Dakota GIS ArcGIS server. EMS stations are fetched from Homeland Infrastructure Foundation-Level Data ArcGIS server.

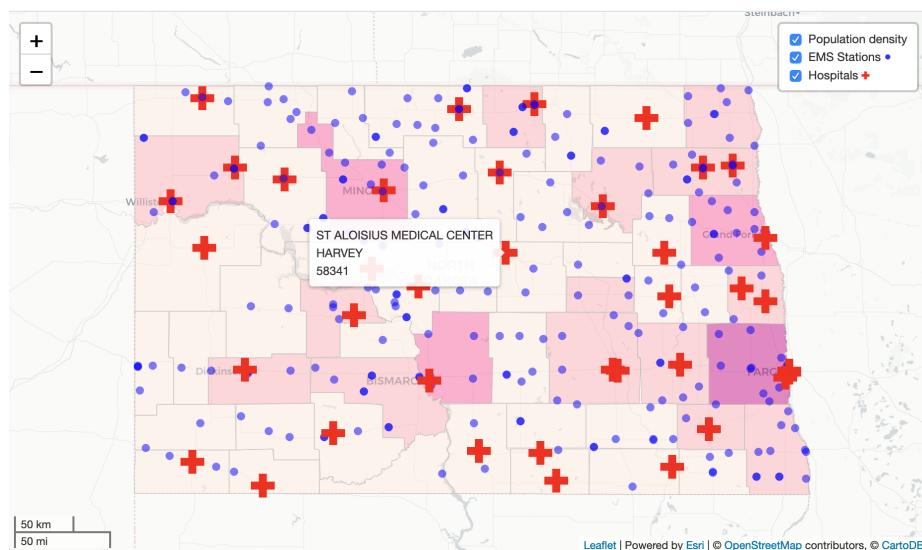


Figure 13.38: Explore the interactive Leaflet Map with Open Data.

You can enable Leaflet to pull data from ArcGIS servers using a free [esri-leaflet](#) plugin. Data from Socrata can be pulled using jQuery's `$.getJSON()` function, and then passed to Leaflet directly using `L.GeoJson()` function.

To adapt this template for your own project:

1. Visit the GitHub repository that contains the code for the map in Figure 13.38, and press the *Use this template* button to copy the repo to your own GitHub account.
2. All data is pulled from the code inside the `<script>` tag of `index.html`. To pull data from Socrata or another JSON/GeoJSON endpoint, modify the following code snippet with the appropriate URL and icon:

```

/*
  From Medicare's Socrata database, add general hospitals in North Dakota
  using simple filtering on the `state` column, and a GeoJSON endpoint.
  Each point is a custom .png icon with a tooltip containing hospital's name,
  city, and zipcode.
*/
$.getJSON("https://data.medicare.gov/resource/xubh-q36u.geojson?state=ND",
  function(data) {

    var hospitals = L.geoJson(data, {
      pointToLayer: function(feature, latlng) {
        return L.marker(latlng, {
          icon: L.icon({
            imageUrl: 'images/hospital.png',
            iconSize: [24, 24],
            iconAnchor: [12, 12],
            opacity: 0.5
          })
        }).bindTooltip(
          feature.properties.hospital_name
          + '<br>' + feature.properties.city
          + '<br>' + feature.properties.zip_code
        )
      }
    }).addTo(map)
  }
)

```

The following code snippet uses `esri-leaflet` plugin to pull polygon data from an ArcGIS server, and creates a choropleth layer based on population density (stored in `POP10_SQMI` variable of each feature, or polygon).

```

var counties = L.esri.featureLayer({
  url:'https://ndgishub.nd.gov/arcgis/rest/services/All_GovtBoundaries/MapServer/20',
  style: function(feature) {
    return {
      fillOpacity: 0.5,
      weight: 0.5,
      color: 'silver',
      fillColor: getDensityColor(feature.properties.POP10_SQMI)
    }
  }
}

```

```
}) .addTo(map)
```

Here, the `getDensityColor()` function returns a color for a given value based on pre-defined thresholds. In case of the North Dakota example, population density of over 100 people per square mile is assigned the darkest shade of red, while the density of 5 and under is shown with the lightest.

```
var getDensityColor = function(d) {
  return d > 100 ? '#7a0177' :
    d > 50 ? '#c51b8a' :
    d > 20 ? '#f768a1' :
    d > 5 ? '#fbb4b9' :
      '#feebe2'
}
```

While it is convenient to pull data directly from the source databases, remember that those resources are out of your control (unless you administer them, of course). Data changes often come unannounced. For example, if the dataset owner decides to rename the population density field from `POP10_SQMI` to `Pop10_sqmi`, your map will stop showing values correctly. Datasets may get moved to a different domain name or get deleted entirely, so it is wise to have a back-up file saved locally.

If you are more concerned about the long-term functioning of your map as opposed to displaying the most up-to-date version of the dataset, you may consider serving your data from local GeoJSON files instead (but ensure first that it is permitted by the data license).

Summary

In this chapter, we introduced Leaflet map templates for common map problems, such as telling stories about places using scrollable interface, showing point data from databases like Socrata, and creating heatmaps to visualize areas of high event density.

You can use these templates as a base to kickstart your own mapping projects. Leaflet.js is well-documented, and we recommend looking at their tutorials for more inspiration.

In the next chapter, we will talk about geospatial data and introduce several tools that can convert, create, and edit geospatial files.

Chapter 14

Transform Your Map Data

TODO: clean up intro to raise major concepts about different types of data you can place on an interactive map. Then refer to the first two sections to transform points only (bulk geocode and pivot points to polygons), before broadening to GeoJSON and other geospatial data formats, and editor tools, and finally how MapWarper transforms raster data.

All maps, including interactive web maps, are made up of different layers. These are background basemaps, colored or shaded polygons (also known as *choropleth* layers), lines, and point data that are often represented as markers.

In this chapter, we will look at multiple ways to convert and edit geospatial data to create layers (files) that you can use in your favorite mapping tools.

We will begin by looking at strategies to geocode large datasets, such as 10,000 addresses, with US Census tools. Before you can dive into creating shapes and dealing with boundaries in the map, we will introduce various file formats (most notably GeoJSON) and talk about geospatial data in general. You will learn that map data can be raster and vector, that geospatial data consists of location and attribute components, and how GeoJSON is different from Shapefiles and other geographical data formats.

With our tutorials, you will learn how to find geographic boundaries in GeoJSON format, convert or draw your own layer of map polygons or polylines on top of satellite imagery using the GeoJson.io tool, and also how to edit geospatial data and join it with spreadsheet data using the Mapshaper tool. Both are powerful, web-based open-source geodata tools that for common tasks can substitute for more complex geographic information system tools, such as ArcGIS, a Windows-only de-facto industry standard in geospatial software, or QGIS, a free and open-source alternative available for Mac, Linux, and Windows. Finally, you'll also learn how to georeference a high-quality static image of a map, and transform it into interactive map tiles that you can display on top of a zoomable Leaflet map, using the MapWarper tool.

By the end of this chapter, you should feel much more confident navigating the overwhelming world of geospatial data.

Bulk Geocode with US Census

In Chapter 3: Strengthen Your Spreadsheet Skills, you learned how to geocode addresses with a Google Sheets Add-On called Geocoding by SmartMonkey. Geocoding converts street addresses to latitude-longitude coordinates (such as *300 Summit St, Hartford CT, USA* to *41.75, -72.69*) that can be placed on maps. While the Geocoding by SmartMonkey Add-On for Google Sheets works well for medium-sized batches of addresses, sometimes you need a faster geocoding service for larger jobs.

One of the fastest ways to geocode up to 10,000 US addresses at a time is to use the US Census Geocoder. First, create a CSV file with 5 columns. Your file must *not* contain a header row, and needs to be formatted the following way:

```
| 1 | 300 Summit St | Hartford | CT | 06106 |
| 2 | 1012 Broad St | Hartford | CT | 06106 |
```

- Column 1: Unique IDs for each address, such as 1, 2, 3, etc. While it does not necessarily have to start at 1 or be in consecutive order, this is the easiest. To quickly create a column of consecutive numbers in most spreadsheets, enter 1, select the bottom-right corner of the cell, hold down the Option or Control key and drag your mouse downward.
- Column 2: Street address.
- Column 3: City.
- Column 4: State.
- Column 5: Zip Code.

Although some of your data, such as zipcodes or states, may be missing and the geocoder may still be able to recognize and geocode the location, unique IDs are absolutely necessary to include for each row (address).

Tip: If your original data combines address, city, state, and zip into one cell, then see how to Split Data into Separate Columns in Chapter 5: Clean Up Messy Data. But if your street addresses contain apartment numbers, you can leave them in.

Second, upload your CSV file to the US Census Geocoder address batch form. Select *Find Locations Using... > Address Batch*, then choose your file to upload. Select *Public_AR_Current* as the benchmark, and click *Get Results*.

Note: In left-side menu, you can switch from *Find Locations* to *Find Geographies* if you wish to obtain additional information, such as the GeoID for each address.

The US Census assigns a unique 15-digit GeoID to every place, and a sample (such as 090035245022001) consists of the state (09), followed by the county (003), the census tract (524502, or more conventional 5245.02), the census block group (2), and finally the census block (001).

In a few moments the tool will return a file named *GeocodeResults.csv* with geocoded results. It usually takes longer for larger files. Save it, and inspect it in your favorite spreadsheet tool. The resulting file is an eight-column CSV file with the original ID and address, match type (exact, non-exact, tie, or no match), and latitude-longitude coordinates. A *tie* means there are multiple possible results for your address. To see all possible matches of an address that received a *tie*, use *One Line* or *Address* tools in the left-side menu and search for that address.

Tip: If you see some unmatched addresses, use a filtering functionality of your spreadsheet to filter for unmatched addresses, then manually correct them, save as a separate CSV file, and re-upload. You can use the US Census Geocoder as many times as you want, as long as a single file doesn't exceed 10,000 records.

To learn more about this service, read the Overview and Documentation section of the US Census Geocoder.

If for some reason you cannot geocode address-level data, but you need to produce some mapping output, you can use pivot tables to get counts of points for specific areas, such as towns or states. In the next section, we will look at hospital addresses in the US and how we can count them by state using pivot tables.

Pivot Points into Polygon Data

If you deal with geographical data, you may find yourself in a situation where you have a list of addresses which need to be counted (*aggregated*) by area and displayed as a polygon map. In this case, a simple pivot table in a spreadsheet software can solve the problem.

Note: A special case of a polygon map is a *choropleth* map, which represents polygons that are colored in a particular way to represent underlying values. A lot of polygon maps end up being *choropleth* maps, so we will be using this term a lot in this book.

Let's take a look at a list of all hospitals that are registered with the Medicare program in the United States. The dataset is stored and displayed by Socrata, a web database popular among government agencies and city administrations. This particular dataset has information on each hospital's name, location (nicely divided into Address, City, State, and ZIP Code columns), a phone number and some other indicators, such as mortality and patient experience.

Now, imagine you are given a task to create a choropleth map of total hospitals by US state. Instead of showing individual hospitals as points (as in Figure 14.1a), you want darker shades of blue to represent states with more hospitals (as in Figure 14.1b).

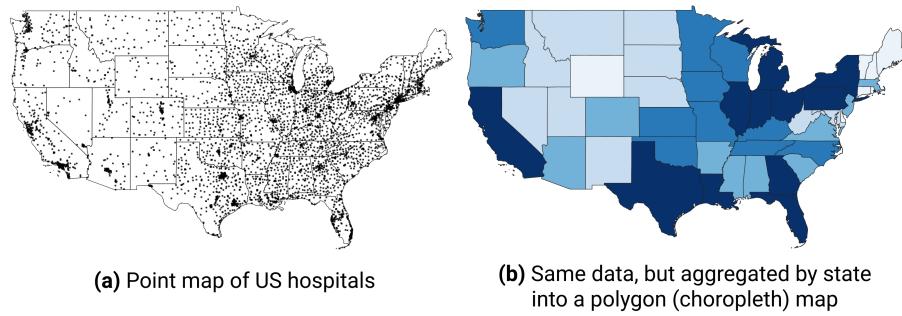


Figure 14.1: You can count addresses by state (or other area) to produce polygon, or choropleth, maps instead of point maps.

First, save the database to your local machine by going to *Export > Download > CSV* of Socrata interface. Figure 14.2 shows where you can find the Export button.

Next, open the file in your favorite spreadsheet tool. If you use Google Sheets, use *File > Import > Upload* to import CSV data. Make sure your address columns are present, and move on to creating a pivot table (in Google Sheets, go to *Data > Pivot table*, make sure the entire data range is selected, and click *Create*). In the pivot table, set *Rows* to *State*, because we want to get counts by state. Next, set pivot table's *Values* to *State*—or really any other column that has no missing values—and choose *Summarize by: COUNTA*. Voila!

Your aggregated dataset is ready, so save it as a CSV. If you use Google Sheets, go to *File > Download > Comma-separated values (.csv, current sheet)*. You can now merge this dataset with your polygons manually using editing capabilities of GeoJson.io, or merge it all in one go using powerful Mapshaper.

We will introduce both tools in the next few sections. But before we do that, let's talk about geographical data in general, introduce different geospatial file formats to ensure you are ready to create, use, and share map data.

GeoJSON and Geospatial Data

Geospatial data comes in an overwhelming number of file formats. We will tell you about a few most common ones so that you have a general idea of what

The screenshot shows a web browser displaying the Data.Medicare.gov website. The main page lists 'Hospital General Information' with a table of hospital details. On the right, there is a 'Export' dialog box open, showing options for 'SODA API', 'OData', and 'Download'. Under 'Download', 'CSV' is selected, and other options like 'JSON' and 'RSS' are listed below. At the bottom of the main page, there is a note about the dataset being a list of registered hospitals.

Figure 14.2: In Socrata, you can export the entire dataset as a CSV.

The screenshot shows a spreadsheet application with a pivot table editor open. The data table on the left contains two columns: 'State' and 'COUNTA of State'. The pivot table editor on the right has three sections: 'Rows', 'Columns', and 'Values'. The 'Rows' section is set to 'State' with 'Order' set to 'Ascending' and 'Sort by' set to 'State'. The 'Values' section is also set to 'State' with 'Summarize by' set to 'COUNTA' and 'Show as' set to 'Default'. Both sections have an 'Add' button. The 'Filters' section is currently empty.

State	COUNTA of State
AK	25
AL	97
AR	86
AS	1
AZ	93
CA	380
CO	94
CT	36
DC	9
DE	12
FL	210
GA	147
GU	2
HI	25
IA	118
ID	45
IL	193
IN	146
KS	141
KY	102
LA	152

Figure 14.3: Use pivot tables in any spreadsheet software to count addresses per area (such as state, county, or zip code).

tools you can use to work with them. But before we do that, let's talk about the basics of geospatial (map) data.

The first thing to know about geospatial data is that it consists of two components, *location* and *attribute*. When you use Google Maps to search for a restaurant, you get a red marker on the screen that points to the latitude and longitude of the physical location of the restaurant in the real world. These latitude and longitude (two numbers) are your location component. The name of the restaurant, its human-friendly address, and guest reviews are the attributes, which bring value to your location data.

Second, geospatial data can be *raster* or *vector*, as illustrated in Figure 14.4. Raster data, as shown in Figure 14.4a, is a grid of cells (“pixels”) of a certain size (for example, 1 meter by 1 meter). For example, satellite images of the Earth that you see on Google Maps are raster geospatial data. Each pixel contains the color of Earth that satellite cameras were able to capture. People and algorithms can then use raster data (images) to create outlines of buildings, lakes, roads, and other objects. These outlines become vector data. For example, most of OpenStreetMap was built by volunteers tracing outlines of objects from satellite images.

TODO: Use this opportunity to briefly show OSM editor and make a pitch for people to volunteer for this crowdsourced map of the world...

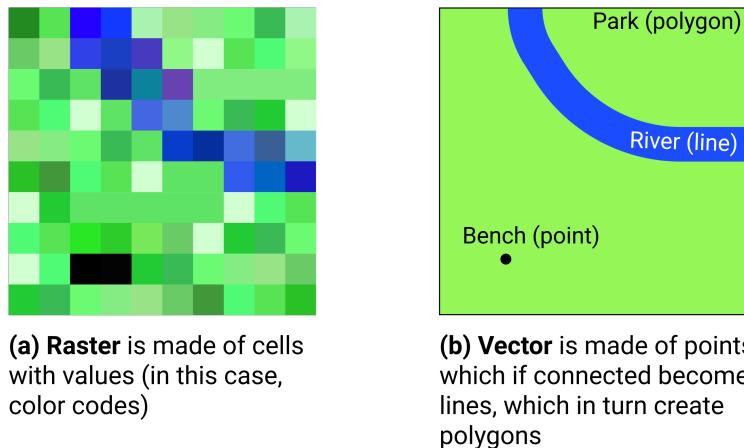


Figure 14.4: Geospatial data can be raster or vector.

In this book, we will focus on vector data, which is based on features, which can be points, lines, and polygons, as shown in Figure 14.4b. Vector data can be much more precise than raster data, because point coordinates can be expressed with precise decimals. In addition, vector data can contain as much extra *attribute* information about each object as desired, whereas raster data is

generally limited to 1 value per cell, whether it is the surface color (as is the case with satellite imagery), temperature, or altitude. Moreover, vector map files tend to be smaller in size than raster ones, which is important if you share files on the web.

Let's take a look at some of the most common vector file formats.

GeoJSON

GeoJSON is a newer, popular open format for map data that comes in `.geojson` or `.json` files. It was first developed in 2008, and then standardized in 2016 by the Internet Engineering Task Force (IETF). The code snippet below represents a single point with latitude of 41.76 and longitude of -72.67 in GeoJSON format. That point has a `name` attribute (property) whose value is *Hartford*.

```
{
  "type": "Feature",
  "geometry": {
    "type": "Point",
    "coordinates": [-72.67, 41.76]
  },
  "properties": {
    "name": "Hartford"
  }
}
```

Other feature types in GeoJSON can be polygons (`Polygon`) and line strings (`LineString`, also known as polylines), which are represented as arrays of points.

The simplicity and readability of GeoJSON allows you to edit it even in the most simple text editor. We strongly recommend you use and share your map data in GeoJSON. Web-based maps, such as those built with Leaflet, Mapbox, Google Maps JS API, and Carto, as well as ArcGIS and QGIS all support GeoJSON. By having your geospatial data stored and shared in GeoJSON, you ensure you can use it on the web with nearly any mapping tool. You can also be confident that other people will be able to use and extract data from the file without bulky and often expensive GIS software installed.

Also, your GitHub repository will automatically display any GeoJSON files in a map view, like is shown in Figure 14.5.

Warning: In GeoJSON, coordinates are ordered in *longitude-latitude* format, the same as X-Y coordinates in mathematics. This is the opposite of Google Maps and some other web map tools, which order values as *latitude-longitude*. For example, *Hartford, Conn.* is located at (-72.67, 41.76) according to GeoJSON, but at (41.76, -72.67) in Google Maps. Neither notation is right or wrong, just make sure you know which one you are dealing with. Tom MacWright created

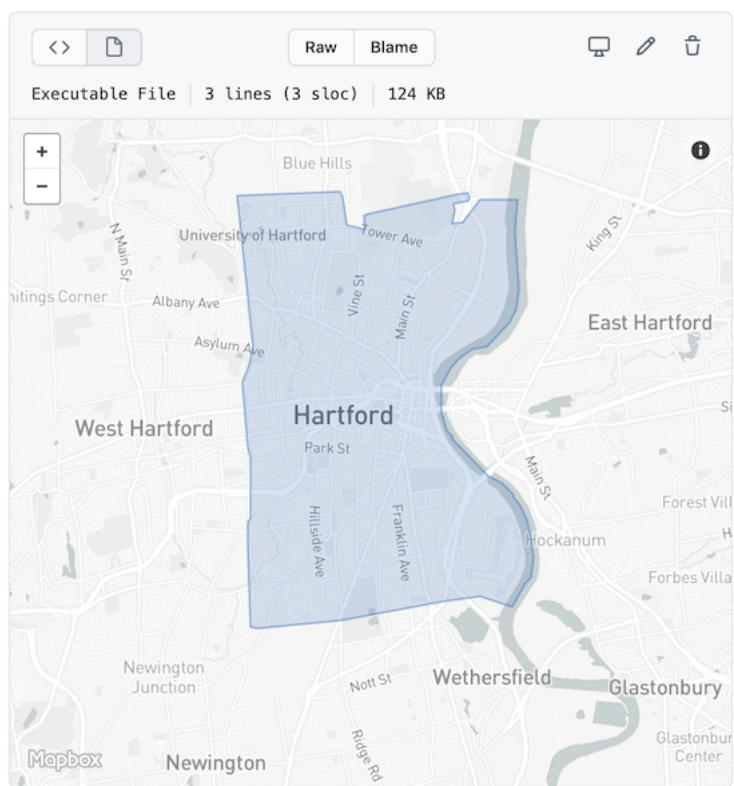


Figure 14.5: GitHub can show previews of GeoJSON files stored in repositories.

a great summary table showing lat/lon order of different geospatial formats and technologies.

Shapefiles

The shapefile format was created in the 1990s by Esri, the company that develops ArcGIS software. Shapefiles typically appear as a folder of subfiles with suffixes such as `.shp`, `.shx`, and `.dbf`. The folder with shapefiles is often compressed in a `.zip` file.

Although government agencies commonly distribute map data in shapefile format, the standard tools for editing these files—ArcGIS and its free and open-source cousin, QGIS—are not as easy to learn as other tools in this book. For this reason, we recommend converting shapefiles into GeoJSON files if possible. Mapshaper, discussed a bit later in the chapter, can perform such conversion.

GPS Exchange Format (GPX)

If you ever exported your Strava run or a bike ride from a GPS device, chances are you ended up with a `.gpx` file. GPX is an open standard and is based on XML markup language. Like GeoJSON, you can inspect a GPX file in any simple text editor to see its contents. Most likely, you will see a collection timestamps and latitude/longitude coordinates of the recording GPS device at that particular time. You should be able to convert GPX to GeoJSON with GeoJson.io utility discussed later in this chapter.

Keyhole Markup Language (or KML)

The KML format rose in popularity during the late 2000s. It was developed for Google Earth, a free and user-friendly tool that allowed many people to view and edit two- and three-dimensional geographic data. KML files were often used with maps powered by Google Fusion Tables, but that became history in late 2019. GeoJson.io should be able to convert your KML file into a GeoJSON.

Sometimes `.kml` files are distributed in a compressed `.kmz` format. See Converting from KMZ to KML format section of this book to learn to convert.

MapInfo TAB

Similar to Esri's shapefiles, MapInfo's TAB format comes as a folder with `.tab`, `.dat`, `.ind`, and some other files. It is a proprietary format created and supported by MapInfo, Esri's competitor, and is designed to work well with MapInfo Pro GIS software. Unfortunately, you will most likely need MapInfo Pro, QGIS, or ArcGIS to re-save these as GeoJSON or a Shapefile.

We've mentioned only a handful of the most common geospatial file formats. There is a myriad of other, less known formats for both raster and vector data. Remember that GeoJSON is one of the best, most universal formats for your *vector* data, and we strongly recommend to store and share your map data in GeoJSON. In the next section, we will look at free online tools to create, convert, join, crop, and in other ways manipulate GeoJSON files.

Find GeoJSON Boundary Files

You may be searching for geographic boundary files in GeoJSON format to create a customized map, such as a choropleth map in Datawrapper as described in Chapter 8, or when using a Leaflet map code template as described in Chapter 13, which both allow you to upload your own GeoJSON files. Since GeoJSON is an open data standard, you may find them in the open data repositories listed in Chapter 4.

Another way to find and download GeoJSON files is Gimme Geodata, a clever tool developed by Hans Hack, which provides quick access to multiple layers of OpenStreetMap boundary files. When you open the tool, search for a location and click a specific point on the map. The tool displays the names and outlines of different geographic boundaries around that point that have been uploaded into OpenStreetMap, which you can select and download in GeoJSON format. For example, when you search and click on Toronto Centre, the tool displays several neighborhood-level boundaries, the Old Toronto city boundary, the present-day Toronto city boundary, and regional and provincial boundaries, as shown in Figure 14.6. Read more details about each layer to evaluate its accuracy, then select any to download in GeoJSON format. The tool also includes an editor (the scissors symbol) to remove water areas, such as Lake Ontario, from the boundary file. When using any type of data that you downloaded from OpenStreetMap, be sure to credit the source in your final product like this: © OpenStreetMap contributors.

Draw and Edit with GeoJson.io

GeoJson.io is a popular open-source web tool to convert, edit, and create GeoJSON files. The tool was originally developed by Tom MacWright in 2013 and quickly became a go-to tool for geospatial practitioners.

In this tutorial, we will show you how to convert existing KML, GPX, TopoJSON, and even CSV files with lat/lon data into GeoJSON files. We will also look at editing attribute data and adding new features to GeoJSON files, and creating them from scratch by tracing satellite imagery.

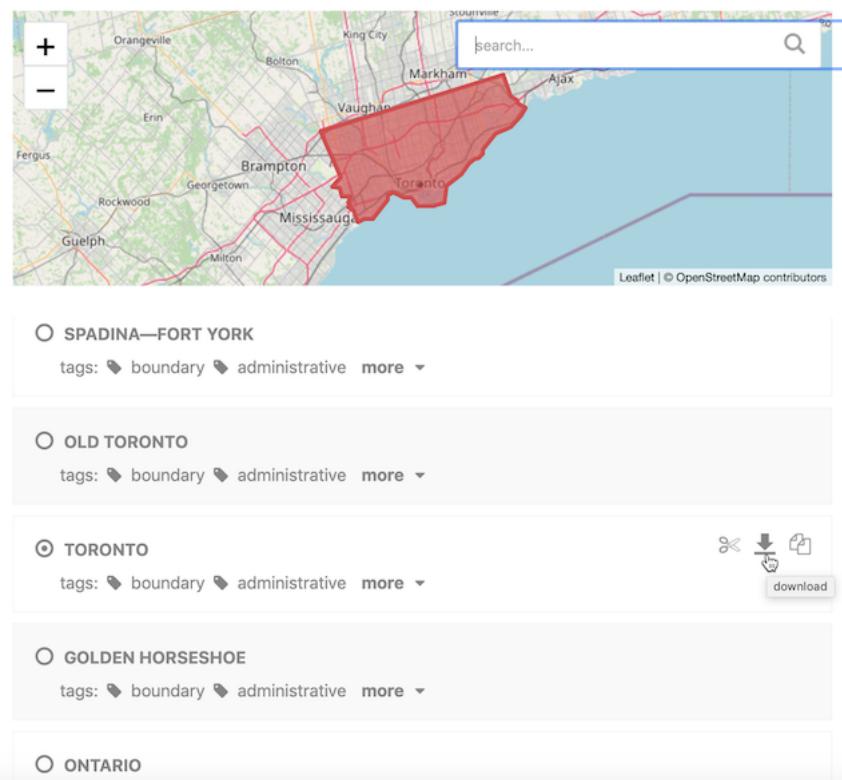


Figure 14.6: Use the Gimme Geodata tool to select a point and download surrounding geographic boundaries from Open Street Map.

Convert KML, GPX, and other formats into GeoJSON

Navigate to [GeoJson.io](#). You will see a map on the left, and a Table/JSON attribute view area on the right. At the start, it represents an empty feature collection. Remember that features are your points, polylines, and polygons.

Drag and drop your geospatial data file into the map area on the left. Alternatively, you can also import a file from *Open > File* menu. If you don't have a geospatial file, download Hartford parks in KML format. If GeoJson.io was able to recognize and import the file, you will see a green popup message in the upper-left corner saying how many features (in case of Hartford parks, only polygons) were imported. Figure 14.7 shows us that 62 features were imported from the sample Hartford parks file. You can see that the polygons appeared on top of the Mapbox world layer.

Note: If GeoJson.io couldn't import your file, you will see a red popup saying it "Could not detect file type". You will need to use a different tool, such as Mapshaper or QGIS, to convert your file to GeoJSON.

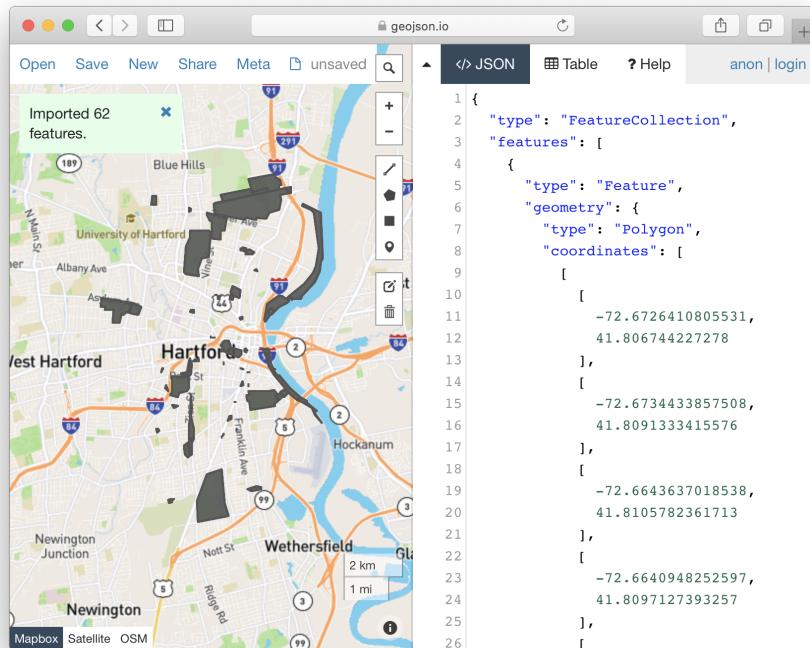


Figure 14.7: GeoJson.io successfully imported Hartford parks KML file.

You can now save your file to GeoJSON. Go to *Save > GeoJSON* to download.

a converted GeoJSON file to your computer.

Create GeoJSON from a CSV file

GeoJson.io can transform a spreadsheet with *latitude* (or *lat*) and *longitude* (or *lon*) columns into a GeoJSON file of point features. Each row in the spreadsheet becomes its own point, and all columns other than *lat* and *lon* become *attributes* (or *properties*) of point features. An example of such spreadsheet is shown in Figure 14.8. You can download it for the exercise.

A	B	C	D	E
town	lat	lon	community_type	wiki_link
Hartford	41.76	-72.67	urban core	https://en.wikipedia.org/wiki/Hartford,_Connecticut
Bloomfield	41.85	-72.73	urban periphery	https://en.wikipedia.org/wiki/Bloomfield,_Connecticut
West Hartford	41.76	-72.75	urban periphery	https://en.wikipedia.org/wiki/West_Hartford,_Connecticut
Wethersfield	41.71	-72.65	urban periphery	https://en.wikipedia.org/wiki/Wethersfield,_Connecticut
Avon	41.79	-72.86	suburban	https://en.wikipedia.org/wiki/Avon,_Connecticut
Glastonbury	41.69	-72.54	suburban	https://en.wikipedia.org/wiki/Glastonbury,_Connecticut

Figure 14.8: A spreadsheet with lat/lon columns can be transformed into a GeoJSON with point features.

1. Save your spreadsheet as a CSV file, and drag-and-drop it to the map area of GeoJson.io. You should see a green popup in the upper-left corner notifying you how many features were imported.

Note: If you had some data on the map already, GeoJson.io wouldn't erase anything but instead would add point features to the existing map.

2. Click on a marker to see a popup with point properties. If you used the sample file with towns around Hartford, you will see *town*, *community_type*, and *wiki_link* features in addition to the tool's default *marker-color*, *marker-size*, and *marker-symbol* fields.

Tip: The popup is interactive, and you can click and edit each property (including property names). You can also add a new property by clicking the *Add row* button. You can delete the marker by clicking *Delete feature* button.

3. Click *Save* to record all marker changes to the GeoJSON. This will close the popup window, and you will see updated markers in the JSON tab to the right of the map.
4. It may be quicker to view all data as a table instead of dealing with individual marker popups. In the *Table* tab to the right of the map, you can add, rename, and remove columns from *all* features (markers) at once. Table cells are also modifiable, so you can edit your data there.
5. Once you are happy with your map data, go to *Save > GeoJSON* to download the result to your computer. You can also log into GeoJson.io with your GitHub account and save directly to your repository.

Create a GeoJSON from scratch using drawing tools

GeoJson.io lets you create geospatial files from scratch, using simple drawing tools to put markers (points), lines, and polygons to appropriate locations. These are useful when you have no original file to work with. The following steps will show you how to create a new GeoJSON file and add markers, lines, and polygons to it.

1. Open GeoJson.io and in the lower-left corner switch from Mapbox (vector tiles) to Satellite.
 2. In the upper-right corner of the map, use the Search tool to find the area you're interested in mapping. For this exercise, we will use tennis courts at Trinity College, Hartford, as shown in Figure 14.9.

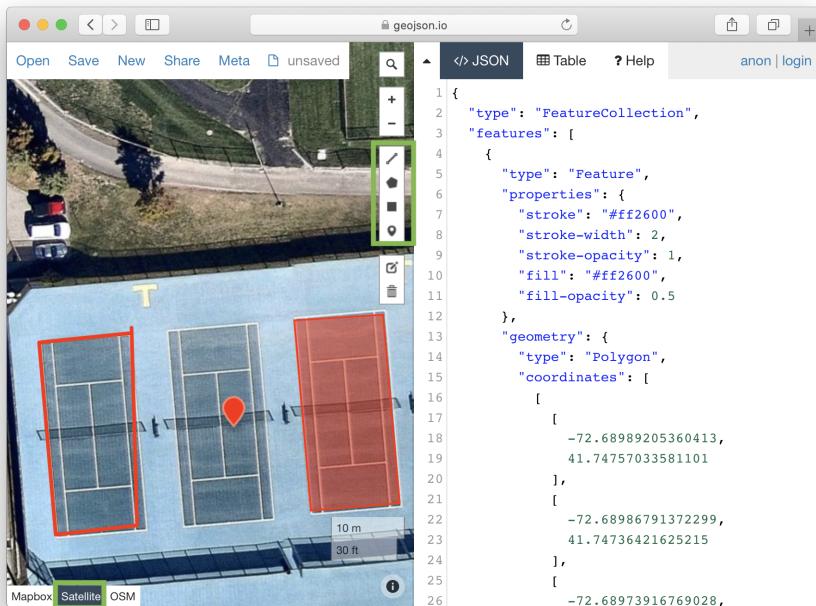


Figure 14.9: Use drawing tools to create points, lines, and polygons in GeoJSON.io.

3. In the toolbar, you have a choice of four drawing tools: a polyline (which is a series of points connected by lines, but not closed like a polygon), a polygon, a rectangle (which is just an instance of a polygon), and a marker (point). Let's start by creating a marker.

4. Click on the *Draw a marker* button, and click anywhere on the map to place it. You will see a gray marker that is now part of your map. You can modify its properties, or delete it in the interactive pop-up.
5. Next, choose *Draw a polyline* and click on multiple locations in the map to see connected lines appearing. To finish and create a feature, click again on the final point. Polylines are generally used for roads and paths.
6. Drawing a polygon is similar to drawing a polyline, except that you need to complete the feature by making your final point at the same location as your initial point. Polygons are used to define object boundaries, from continents to buildings, cars, and anything that has significant dimensions.
7. Use *Edit layers* tool (the one above *Delete*) to move a marker to a better position, or adjust the shapes of your features.

Once you are done creating features and their physical boundaries, it is time to add meaningful attribution data. Use the interactive popups or the Table view to give objects names and other qualities. When finished, save the GeoJSON to your computer.

Drawing tools can be used to correct your existing GeoJSON files. For example, if you created a GeoJSON from a CSV file, you might decide to move some markers with *Edit layers* tool instead of modifying their latitude and longitude values. Or you might decide that your polygons (eg those representing Hartford parks) are too “simplified”, and make them more precise with the satellite imagery.

In the next section, we will introduce Mapshaper, another free online tool to convert and modify geospatial files.

Edit and Join with Mapshaper

Like GeoJson.io, Mapshaper is a free, open-source editor that can convert geospatial files, edit attribute data, filter and dissolve features, simplify boundaries to make files smaller, and many more. Unlike GeoJson.io, Mapshaper doesn’t have drawing tools, so you won’t be able to create geospatial files from scratch.

Mapshaper is developed and maintained by Matthew Bloch on GitHub. It is written in JavaScript, so we recommend you use a recent version of Firefox or Chrome.

This free and easy-to-learn Mapshaper web tool has replaced *many* of our map preparation tasks that previously required expensive and hard-to-learn ArcGIS software, or its free but still-challenging-to-learn cousin, QGIS. Even advanced GIS users may discover Mapshaper to be a quick alternative for some common but time-consuming tasks.

Import, convert, and export map boundary files

You can use Mapshaper to convert between geospatial file formats. Unlike GeoJson.io, Mapshaper also supports Esri Shapefiles (which is a folder of individual files with the same name, but different file extensions), so you can easily convert a Shapefile into a web-friendly GeoJSON. In the following steps, we will convert a geospatial file by import it to Mapshaper, and then export it as a different file type.

1. Navigate to Mapshaper.org. The start page is two large drag-and-drop zones which you can use to import your file. The smaller area at the bottom, *Quick import*, uses default import settings and is a good way to begin.
2. Drag and drop your geospatial file to the *Quick import* area, or use our sample Shapefile of US state boundaries. This is a `.zip` archive which contains a folder with all necessary files.

Note: If you want to import a folder, you need to either select all files inside that folder and drop them all together to the import area, or create a `.zip` archive.

3. Each imported file becomes a layer, and is accessible from the dropdown menu in the top-middle of the browser window. There, you can see how many features each layer has, toggle their visibility, or delete them.
4. To export, go to *Export* in the upper-right corner, and select a desired file format. The choice of export formats is shown in Figure 14.10. As of July 2020, these are Shapefile, GeoJSON, TopoJSON (similar to GeoJSON, but with topographical data), JSON records, CSV, or SVG (Scalable Vector Graphics, for web and print). If you export more than one layer at a time, Mapshaper will archive them first, and you will download an `output.zip` that contains all exported layers.

Tip: Mapshaper doesn't work with KML or KMZ files, but you can use GeoJson.io to convert these.

Edit data for specific polygons

You can edit attribute data of individual polygons (and also points and lines) in Mapshaper. Figure 14.11 shows you how.

1. Import the file whose polygon attributes you want to edit.
2. Under the cursor tool, select *edit attributes*.
3. Click on the polygon you want to edit. A pop-up will appear in the upper-left corner listing all attributes and values of the polygon.

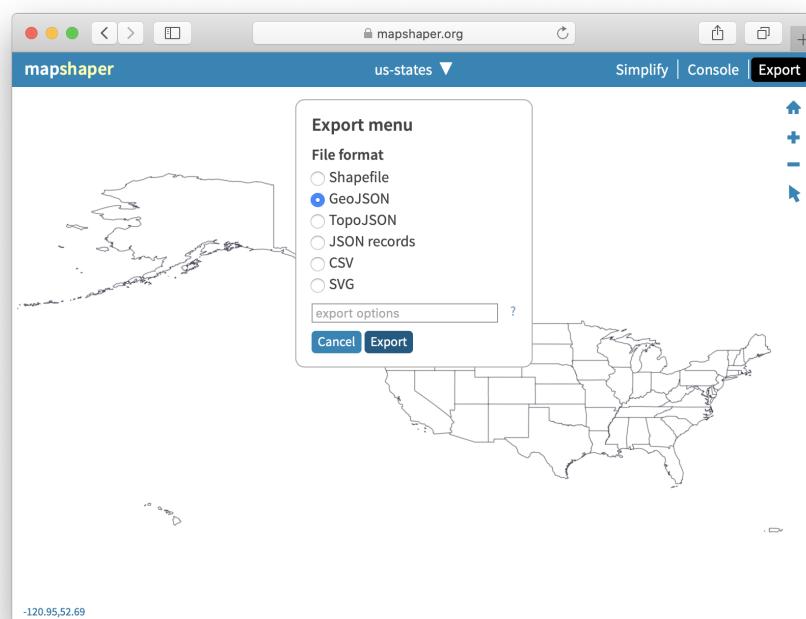


Figure 14.10: You can use Mapshaper to quickly convert between geospatial file formats.

4. Click on any value (underlined, in blue) and edit it.
5. When you are done, export your geospatial file by clicking *Export* and choosing the desired file format.

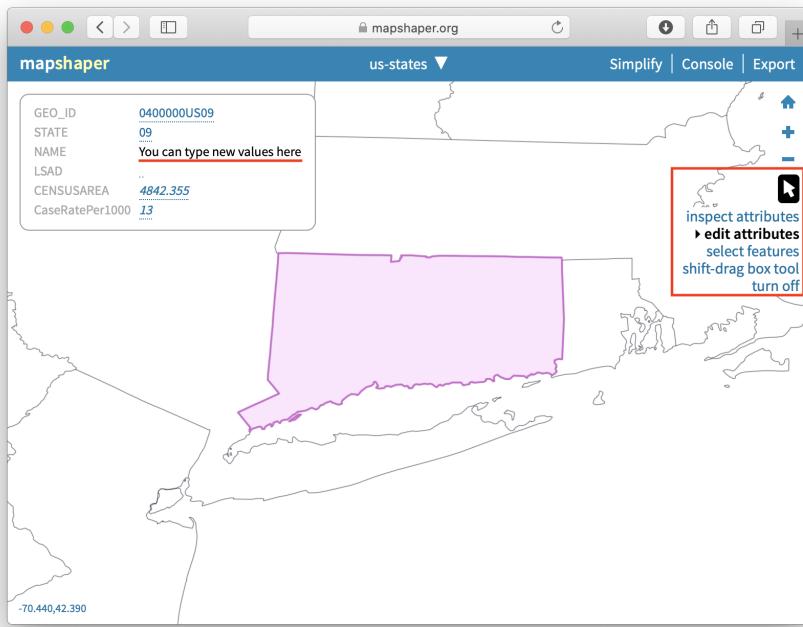


Figure 14.11: Use *edit attributes* tool (under Cursor tool) to edit attributes of polygons, lines, and points.

Simplify map boundaries to reduce file size

You may not need precise and detailed map boundaries for data visualization projects where zoomed-out geographies are shown. Detailed boundaries are heavy, and may slow down your web maps.

Consider two maps of the contiguous US states (also known as *the lower 48*, the term Ilya learned in 2018 while travelling in Alaska), shown in Figure 14.12. The map in Figure 14.12a is more detailed and is about 230 kilobytes, but the map in Figure 14.12b is only 37 kilobytes, 6 times smaller!

To simplify map boundaries in Mapshaper, follow the steps below.

1. Import your geo file to Mapshaper. You can use the sample contiguous US states GeoJSON.



Figure 14.12: Consider simplifying geometries with Mapshaper to make your web maps faster.

2. Click the *Simplify* button in the upper-right corner. The Simplification menu will appear, where you can choose one of three methods. We recommend checking *prevent shape removal*, and leaving the default *Visvalingam / weighted area*. Click *Apply*.
3. You will see a slider with 100% appear on top (Figure 14.13), replacing the layer selection dropdown. Move the slider to the right and see the map simplify its shape as you go. Stop when you think the map looks appropriate (when the shapes are still recognizable).
4. Mapshaper may suggest to repair line intersections in the upper-left corner. Click *Repair*.
5. You can now export your file using the *Export* feature.

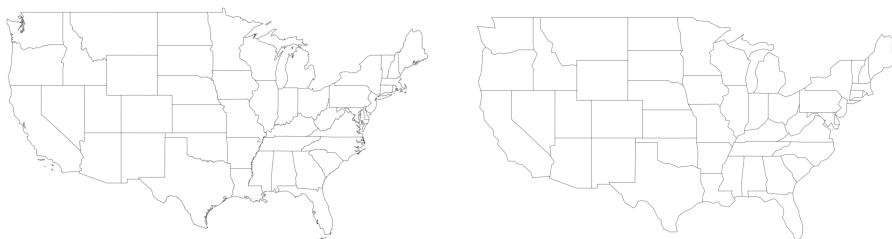


Figure 14.13: Use Simplify & Repair tools in Mapshaper.

Tip: You may find the US shape a bit unusual and vertically “shrunk”. In **Console**, type `-proj EPSG:3857` to change projection to Web Mercator, which is more common.

Dissolve internal polygons to create an outline map

Mapshaper's most powerful tools are available through the *Console*, which allows you to type commands for common map editing tasks. One of such tasks is to create an outline map by removing the internal boundaries. For example, you can dissolve state boundaries of the US map in the previous exercise to get the outline of the country, like is shown in Figure 14.14.

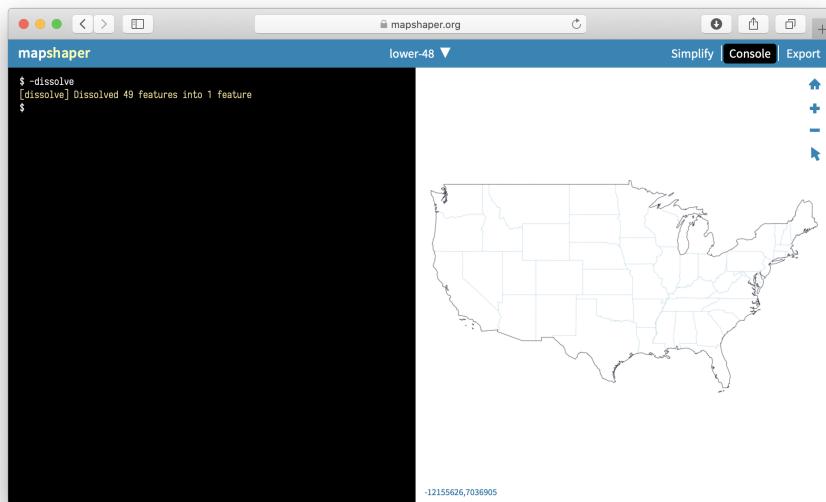


Figure 14.14: Mapshaper lets you dissolve boundaries to create an outline shape.

Click the *Console* button, which opens a window to type in commands. Enter the command below, then press return (Enter).

```
-dissolve
```

You will see that internal boundaries became lighter color, and that's Mapshaper's way of saying they no longer exist. You can now export your outline shape.

Clip a map to match an outline layer

The state of Connecticut consists of 8 counties, which in turn are divided into towns. There are a total of 169 towns in Connecticut. Imagine you are given a boundary file of all 169 towns, and the outline of Hartford county. You need to “cut” the original towns map to only include those towns that fall within Hartford county.

Mapshaper allows you to do just that using one simple `-clip` command.

1. Import two boundary files into Mapshaper. One is the larger one that is being clipped (if you use sample files, *ct-towns*), and one is the desired final shape (*hartfordcounty-outline*). The latter is what ArcGIS calls the “clip feature”.
2. Make sure your active layer is set to the map you are clipping (*ct-towns*).
3. In the *Console*, type `-clip` followed by the name of your clip layer, like that:

```
-clip hartfordcounty-outline
```

4. You should see your active layer got clipped. Sometimes you end up with tiny “slivers” of clipped areas that remain alongside the borders. If that is the case, use the `-filter-slivers` command to remove them, like that:

```
-clip hartfordcounty-outline -filter-slivers
```

5. Your Mapshaper state should look like pictured in Figure 14.15. You can now save the file on your computer using the *Export* button.

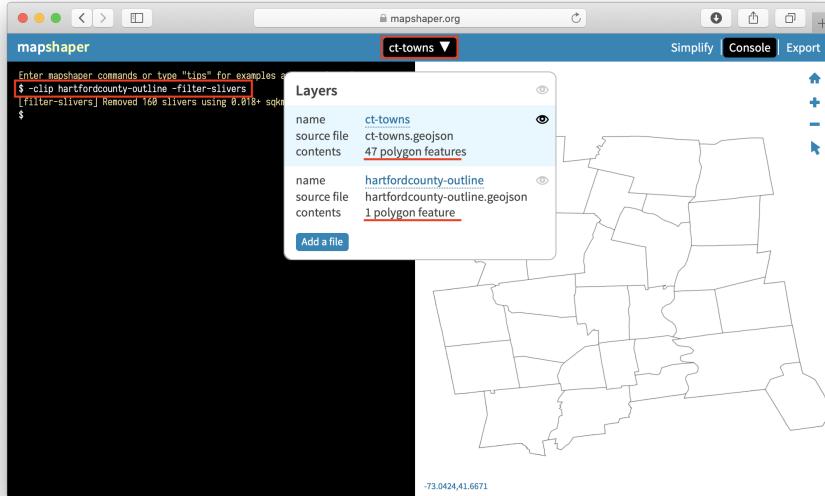


Figure 14.15: When clipping, make sure your active layer is the one being clipped (with many features), not the clipping feature itself.

Remove unwanted data fields

Sometimes map features, such as polygons, lines, and points, contain unwanted *attributes* (or fields, or columns) that you may want to remove. In the *Console*, type the `-filter-fields` editing command to remove unnecessary fields.

For example, remove all fields except *town*:

```
-filter-fields town
```

If you want to leave more than one field, separate them by a comma, but without spaces, like that:

```
-filter-fields town,state
```

Warning: If you leave a space after comma, you will get a *Command expects a single value* error.

Join spreadsheet data with polygon map

Combining spreadsheet data with geographical boundaries is a common task for geospatial practitioners. Imagine you have a file with Connecticut town boundaries, and you want to add population data to each of them in order to build a choropleth map.

Mapshaper provides a powerful `-join` command to join such files. Remember that you need some common keys in both datasets (such as *town name*, or *state*, or *country*) in order to join files. Otherwise Mapshaper has no way of knowing which numbers belong to which polygons.

1. Import both geospatial file and a CSV dataset into Mapshaper using Quick import box.
2. Make sure both files appear in the drop-down list of layers. Your CSV data will be shown as something that resembles a table. Use the *Cursor > inspect attributes* tool to make sure the data is imported correctly. If you use the sample CT files, note that the *ct-towns* layer has *name* attribute with the name of the town, and *ct-towns-popdensity* has town names in the *town* column.
3. Make your geospatial layer (*ct-towns*) is the one active.
4. Open the *Console*, and use the `-join` command, like this:

```
-join ct-towns-popdensity keys=name,town
```

where `ct-towns-popdensity` is the CSV layer you are merging with, and `keys` are the attributes that contain values to join by. In case with our sample files, these would be town names which are stored in `name` attribute of the map file, and `town` column of the CSV file.

5. You will see a message in the console notifying you if join was performed successfully, or if Mapshaper encountered any errors.
6. Use the *Cursor > inspect attributes* tool to make sure you see CSV columns as fields of your polygons, like is shown in Figure 14.16.
7. You can now save the file to your computer by clicking the *Export* button.

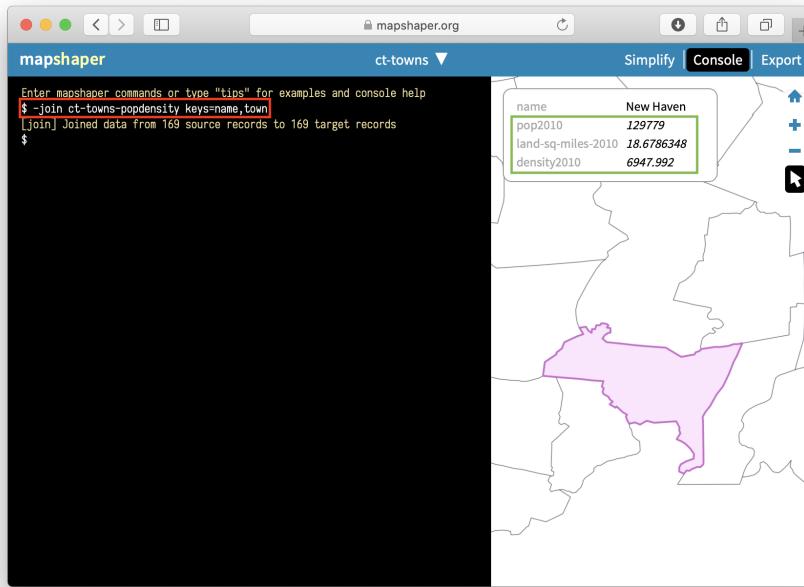


Figure 14.16: Mapshaper lets you join spatial and CSV files using common keys (for example, town names).

Tip: To avoid confusion, it may be useful to re-name your CSV column that contains key values to match the key attribute name of your map. In our example, you would rename `town` column to `name` column in the CSV, and your command would end with `keys=name,name`.

Do you remember aggregating address-level point records of hospitals into hospital counts per state discussed earlier in this chapter? Now is a good time to find that .CSV file and practice your merging skills.

Count points in polygons with Mapshaper

Mapshaper lets you count points in polygons, and record that number in polygon attributes using `-join` command.

1. Import two geospatial files, one containing polygon boundaries (for example, US state boundaries), and another containing points that you want to aggregate (for example, hospitals in the US).
2. Make sure your polygons (not points) layer is active by selecting it from the dropdown menu.
3. In the *Console*, perform `-join` using a `count()` function, like this:

```
-join hospitals-points calc='hospitals = count()' fields=
```

This command tells Mapshaper to count points inside *hospitals-points* layer and record them as *hospitals* attribute of the polygons. The `fields=` part tells Mapshaper to not copy any fields from the points, because we are performing many-to-one matching (many hospitals per state, in our case).

4. Use the *Cursor > inspect attributes* tool to make sure polygons obtained a new field with the recorded count of points, like is shown in Figure 14.17.
5. Save the new file using *Export* button and choosing the desired output format. In the section below, we will talk about what happens to objects that don't join.

More about joins

From the “Count points in polygons with Mapshaper” section of this chapter, you should recall that you do not need to specify *keys* if you want to perform join based on geographical locations between two geospatial layers (one being points, the other is polygons). If one of your files is a CSV, you need *keys*.

If you don't have a CSV table that matches the columns in your boundary map data, you can easily create one. Upload the boundary map to Mapshaper, and export in CSV format. Open the downloaded file in any spreadsheet tool. To match data columns in the CSV spreadsheet, use the VLOOKUP function.

In real life, you will rarely have perfect files with one-to-one matches, so you might want to have more information about which features didn't get matched so that you can fix your data. Mapshaper helps you keep track of data that is not properly joined or matched. For example, if the polygon map contains 169 features (one for each town in Connecticut), but the CSV table contains only 168 rows of data, Mapshaper will join all of those with matching keys, and then display this message:

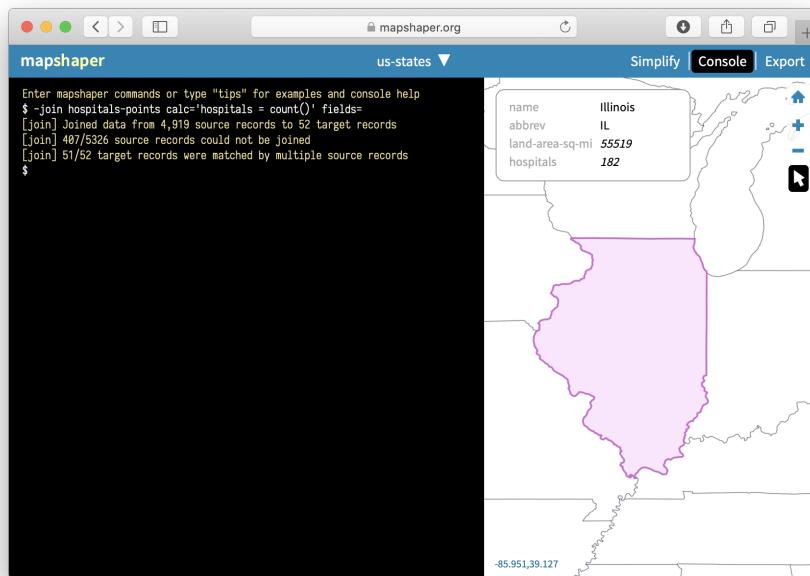


Figure 14.17: Mapshaper's -join can count points in polygons.

```
[join] Joined data from 168 source records to 168 target records
[join] 1/169 target records received no data
[join] 1/169 source records could not be joined
```

To get more details on which values were not joined, add `unjoined unmatched -info` flags to your join command, like this:

```
-join ct-towns-popdensity keys=name,town unjoined unmatched -info
```

The `unjoined` flag saves a copy of each unjoined record from the source table into another layer named *unjoined*. The `unmatched` flag saves a copy of each unmatched record from the target table to a new layer named *unmatched*. And the `-info` flag outputs some additional information about the joining procedure to the console.

Merge selected polygons with join and dissolve commands

In Mapshaper, you can merge selected polygons into larger “clusters” using `-join` and `-dissolve` commands.

Imagine that you are employed by the CT Department of Public Health, and your task is to divide 169 towns into 20 so-called Health Districts and produce a new geospatial file. By the way, health districts are a real thing in Connecticut.

You should begin by creating a *crosswalk* of towns and their health districts. Computer scientists and those working with data often use the term *crosswalk* to describe some kind of matching between two sets of data, such as zipcodes and towns where they are located. In our case, the crosswalk can be as simple as a two-column CSV list of a town and its district, each on a new line. Because your boss didn’t give you a list of towns in a spreadsheet format, but instead a GeoJSON file with town boundaries, let’s extract a list of towns from it.

1. Import `ct-towns.geojson` to Mapshaper using Quick import box.
2. You can use the *Cursor > inspect attributes* tool to see that each polygon has a `name` attribute with the name of the town.
3. Save attribute data as a CSV file using *Export* button. Open the file in any spreadsheet tool. You will see that your data is a one-column file with a `*name&` column that lists 169 towns.
4. Create a second column titled *merged* and copy-paste values from the first, `name` column. At this point your spreadsheet contains two columns with the same values.
5. Pick a few towns, for example *West Hartford* and *Bloomfield*, and assign “*Bloomfield-West Hartford*” to their *merged* column, like is shown in Figure 14.18. You may stop right here and move to the next step, or keep assigning district names to a few other neighboring towns.

	A	B	C
1	name	merged	
2	Bloomfield	Bloomfield-West Hartford	
3	West Hartford	Bloomfield-West Hartford	
4	Bethel	Bethel	
5	Bridgeport	Bridgeport	
6	Brookfield	Brookfield	
7	Danbury	Danbury	
8	Darien	Darien	
9	Easton	Easton	
10	Fairfield	Fairfield	
11	Greenwich	Greenwich	

Figure 14.18: Create a two-column crosswalk of towns and which districts they should be merged to.

6. Save this new file as *ct-towns-merged.csv*, and drag-and-drop it to Mapshaper on top of your *ct-towns* layer. Click *Import*.
7. This new CSV layer will be added as *ct-towns-merged* and will appear as a series of table cells. From the dropdown menu, select *ct-towns* to get back to your map.
8. Now you are ready to merge certain towns into districts according to your uploaded CSV file. Open the *Console*, and type:

```
-join ct-towns-merged keys=name,name
```

to join the CSV layer with the boundaries layer that you see on the screen, followed by

```
-dissolve merged
```

to dissolve polygons of towns according to the *merged* column of the CSV file.

In our example, only Bloomfield and West Hartford are dissolved into a combined “Bloomfield-West Hartford” regional health district (with the shared boundary between towns becoming grayed out), and all of the other polygons remain the same. Figure 14.19 shows the final result.

You can inspect attribute data of polygons using *Cursor > inspect attributes* tool, and save the resulting file using the *Export* button.

Learn more advanced MapShaper methods

There are many more commands within Mapshaper that are worth exploring if you are serious about GIS, such as changing projections, filtering features using

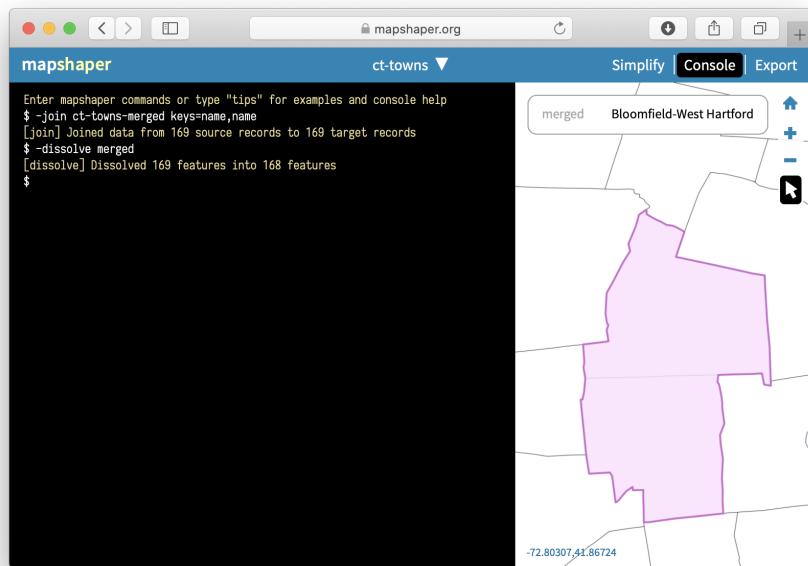


Figure 14.19: Merge polygons based on a predefined crosswalk.

JavaScript expressions, assigning colors to polygons based on values, and many more. Explore the Wiki of Mapshaper project on GitHub for more commands and examples.

Georeference with MapWarper

TODO: write this section about using MapWarper, a tool created and hosted by Tim Waters, to upload and georeference a scanned map. This means to properly position a scanned map based on standard coordinates, so that you can place it as an overlay on an interactive map, such as Leaflet Storymaps with Google Sheets.

Anyone can upload and georectify a map on the developer's public site at <http://mapwarper.net>, and also see how it's used by organizations such as the New York Public Library at <http://maps.nypl.org>.

Warning: MapWarper is a wonderful open-source tool and platform, but service may be interrupted. As of July 2020, the site warns: "Ran out of disk space. Maps older than 2 years will need re-warping to work. Downtime will happen again."

MODIFY TEXT FROM CH 13: To add a historical map overlay, it first must be *georeferenced* (also called georectified), which means to digitally align the static map image with a more precise present-day interactive map. If you have a high-quality static image of a historical map, use the Mapwarper tool as described in Chapter 14 to align several known points with those on a present-day interactive map. Mapwarper transforms the static map image into interactive map tiles, and publicly hosts them online with a link in Google/OpenStreetMap format, similar to <https://mapwarper.net/maps/tile/14781/{z}/{x}/{y}.png>. Or you can search for historical maps that have already been georeferenced and transformed into tiles, or contribute to crowdsourcing efforts to align maps, on platforms such as Mapwarper, the New York Public Library Mapwarper, and the David Rumsey Map Collection. [TODO: confirm how Rumsey links work] Although map tile links are *not* viewable in a normal browser, they can be displayed by the Leaflet Storymaps code by entering them into these columns in the *Chapters* tab of your Google Sheet template:

Convert Compressed KMZ to KML

In the previous sections, we looked at using Geojson.io and Mapshaper to convert geospatial files. However, not all file types can be converted with these tools. This chapter shows a particular example of a commonly requested .kmz <-> .kml pair conversion with Google Earth Pro.

KMZ is a compressed version of a KML file, and the easiest way to convert between the two is to use free Google Earth Pro (if you remember from *Convert to GeoJSON format* section, KML is a native format of Google Earth).

1. Download and install Google Earth Pro for desktop.
2. Double-click on any .kmz file to open it in Google Earth Pro. Alternatively, open Google Earth Pro first, and go to *File > Open* and choose your KMZ file.
3. Right-click (or control-click) on the KMZ layer under Places menu, and select *Save Place As...*, like is shown in Figure 14.20.

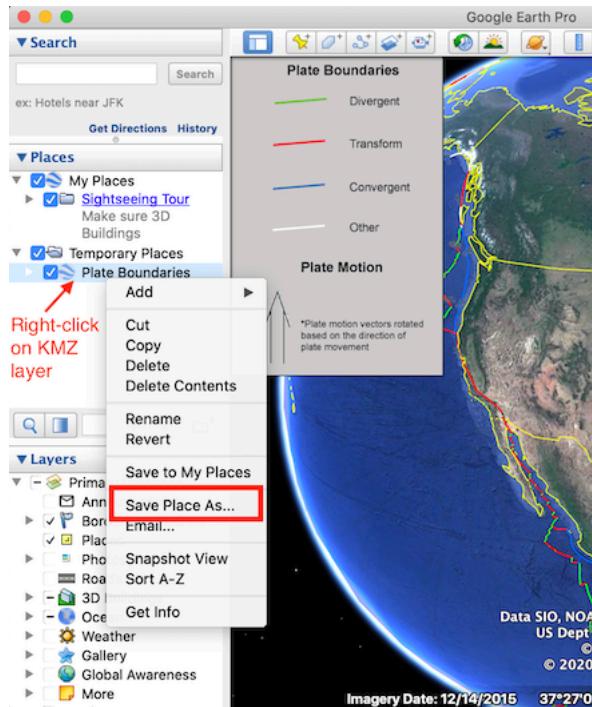


Figure 14.20: In Google Earth Pro, right-click the KMZ layer and choose *Save Place As...*.

4. In the dropdown menu of *Save file...* window, choose KML format, like is shown in Figure 14.21.

Alternatively, you can use any zip-utility to extract a KML file from KMZ. KMZ is simply a ‘zipped’ version of a KML file!

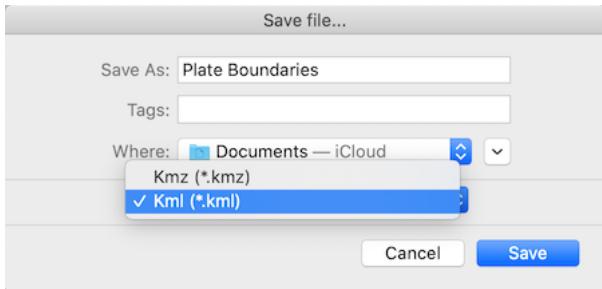


Figure 14.21: Save as KML, not KMZ.

Summary

In this chapter, you learned to use pivot tables to count addresses (points) by geographical area, such as states or cities (polygons). You learned that geospatial data can be vector or raster. The best file format to store, share, and use vector data is GeoJSON. You can use GeoJson.io to create, edit, or convert geospatial files inside your browser. Mapshaper is another online tool to convert, simplify, join or crop geospatial data.

In the following chapter, we will talk about detecting bias in charts and maps. so that you become a better storyteller and a more critical reader.

Chapter 15

Detect Lies and Reduce Data Bias

The goal of data visualization is to encode information into images that capture true and insightful stories. But we've warned you to watch out for people who lie with visualizations. Looking back at income inequality examples in the Introduction to this book, we intentionally manipulated charts in Figure 1.1 and Figure 1.2, and maps in Figure 1.3 and Figure 1.4, to demonstrate how the same data can be rearranged to paint very different pictures of reality. Does that mean all data visualizations are right? Definitely not. On closer examination, we declared that the second of the two charts about US income inequality was *misleading* because it intentionally used an inappropriate scale to hide the truth. But we also confided that the two world maps were *equally truthful*, even though the US appeared in a darker color (signaling a higher level of inequality) than the other.

How can two different visualizations be equally right? Our response may conflict with those who prefer to call their work *data science*, a label that suggests an objective world with only one right answer. Instead, we argue that data visualization is best understood as *interpretative skill* that still depends on evidence, but more than one portrayal of reality may be valid. As you recall, our field has only a few definitive rules about how *not* to visualize data, which we introduced in Chapter 7 on chart design and Chapter 8 on map design. Rather than a binary world, we argue that visualizations fall into three categories.

First, visualizations are *wrong* if they misstate the evidence or violate one of these rigid design rules. For examples of the latter, if a bar or column chart begins at a number other than zero, it's wrong because those types of charts represent values through *length* or *height*, which readers cannot determine if the baseline has been truncated. Similarly, if the slices of a pie chart adds up to more than 100 percent, it's wrong because readers cannot accurately interpret

the chart, which also incorrectly presents data.

Second, visualizations are *misleading* if they technically follow the design rules, but unreasonably hide or twist the appearance of relevant data. We acknowledge that the word “unreasonably” can be subject to debate here, but we’ll review several examples in this chapter, such as using inappropriate scales or warping the aspect ratio. Inserting this category between *wrong* and *truthful* underscores how charts and maps can accurately display data and adhere to design rules, yet misdirect us from the truth, just as a magician knows how to misdirect their audience while performing sleight of hand tricks.

Third, visualizations are *truthful* if they show accurate data and follow the design rules. Still, there’s a wide spectrum of quality within this category. When looking at two visualizations that are equally valid, sometimes we say that one is *better* than the other because it illuminates a meaningful data pattern that we did not yet recognize. Or we may say that one is better because it portrays these patterns more beautifully, or with less ink on the page and greater simplicity, than the other. In any case, let’s agree that we’re aiming for truthful visualizations, with a preference for the better side of the quality spectrum.

In this chapter, you’ll learn to sort out differences between the three categories: wrong, misleading, and truthful. The best way to improve your lie detector skills is through hands-on tutorials in the art of data deception, to better understand how to lie with charts and how to lie with maps. As the saying goes, it takes a thief to catch a thief. Learning *how to lie* not only make it harder for people to mislead you, but also educates you more deeply about the ethical decisions we make when designing visualizations that *tell the truth*, while recognizing there’s more than one path to that destination. Finally, we’ll discuss how to recognize and reduce other types of data bias, such as framing bias, intergroup bias, and map area bias, including how we define the United States. While we may not be able to stop bias entirely, in this chapter you’ll learn how to identify it in the works by other people, and strategies to reduce its presence in our own visualizations.¹

How to Lie with Charts

In this section, you’ll learn how to avoid being fooled by misleading charts, and also how to make your own charts more honest, by intentionally manipulating

¹The “how to lie” tutorials were inspired by several excellent works in data visualization: Cairo, *The Truthful Art*, 2016; Cairo, *How Charts Lie*, 2019; Darrell Huff, *How to Lie with Statistics* (W. W. Norton & Company, 1954), <http://books.google.com/books?isbn=0393070875>; Mark Monmonier, *How to Lie with Maps, Third Edition* (University of Chicago Press, 2018), https://www.google.com/books/edition/How_to_Lie_with_Maps_Third_Edition/MwdRDwAAQBAJ; Nathan Yau, “How to Spot Visualization Lies,” Flowing-Data, February 9, 2017, <http://flowingdata.com/2017/02/09/how-to-spot-visualization-lies/>; NASA JPL, “Educator Guide: Graphing Global Temperature Trends,” 2017, <https://www.jpl.nasa.gov/edu/teach/activity/graphing-global-temperature-trends/>

the same data to tell opposing stories. First you will *exaggerate* small differences in a column chart to make them seem larger. Second you will *diminish* the rate of growth in a line chart to make it appear more gradual. Together, these tutorials will teach you to watch out for key details when reading other people's charts, such as the vertical axis and aspect ratio. Paradoxically, by demonstrating *how to lie*, our goal is to teach you to *tell the truth* and to think more carefully about the ethics of designing your data stories.

Exaggerate Change in Charts

First we'll examine data about the economy, a topic that's often twisted by politicians to portray it more favorably for their perspective. The Gross Domestic Product (GDP) measures the market value of the final goods and services produced in a nation, which many economists consider to be the primary indicator of economic health. (Interestingly, not everyone agrees because GDP does not count unpaid household labor such as caring for one's children, nor does it consider the distribution of wealth across a nation's population.) We downloaded US GDP data from the US Federal Reserve open-data repository, which is measured in billions of dollars and published quarterly, with seasonal adjustments to allow for better comparisons across industries that vary during the year, such as summer-time farming and tourism versus winter-time holiday shopping. Your task is create a deceptive column chart that *exaggerates* small differences to make them appear larger in the reader's eye.

1. Open the US GDP mid-2019 data in Google Sheets, and go to *File > Make a Copy* to create a copy that you can edit in your own Google Drive. We'll create charts in Google Sheets, but you can also download the data to use in a different chart tool if you prefer.
2. Examine the data and read the notes. To simplify this example, we show only two figures: the US GDP for the 2nd quarter (April-June) and the 3rd quarter (July-September) in 2019. The 2nd quarter was about \$21.5 trillion, and the third quarter was slightly higher at \$21.7 trillion. In other words, the quarterly GDP rose by just under one percent, which we calculated this way: $(21747 - 21540)/21540 = 0.0096 = 0.96\%$.
3. Create a Google Sheets column chart in the same sheet using the *default* settings, although we never automatically accept them as the best representation of the truth. In the *data* sheet, select the two columns, and go to *Insert > Chart*, as you learned when we introduced charts with Google Sheets in Chapter 7. The tool should recognize your data and automatically produce a column chart, as shown in the left side of Figure 15.1. In this default view, with the zero baseline for the vertical axis, the difference between \$21.5 versus \$21.7 trillion looks relatively small to the reader.

4. *Truncate the vertical axis to exaggerate differences.* Instead of a zero baseline, let's manipulate the scale to make the 1 percent change in GDP look larger. Click on the three-dot kebab menu to open the *Chart editor* and select the *Customize* tab. Scroll down to the vertical axis settings, and reduce the scale by changing the minimum from 0 (the zero baseline) to 21500, and also change the maximum to 21800, as shown in the right side of Figure 15.1. Although the data remains the same, the small difference between the two columns in the chart now appears much larger in our eyes. Only people who read charts closely will notice this trick. The political candidate who's campaigning on rising economic growth will thank you!

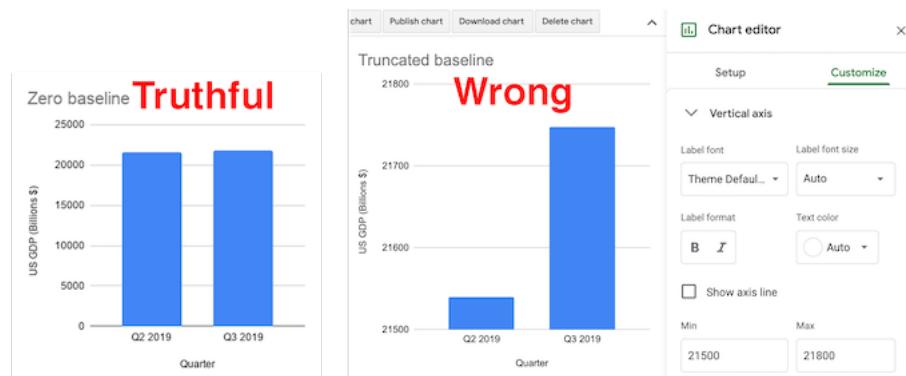


Figure 15.1: The Zero baseline GDP line chart (left), and the Truncated baseline line chart, with the Chart editor (right).

As you can see, the truncated baseline chart is *wrong* because you've violated one of the cardinal rules about chart design in Chapter 7. Column (and bar) charts *must* start at the zero baseline, because they represent value using *height* (and *length*). Readers cannot determine if a column is twice as high as another column unless both begin at the zero baseline. By contrast, the default chart with the zero baseline is *truthful*. But let's move on to a different example where the rules are not as clear.

Diminish Change in Charts

Next we'll examine data about climate change, one of the most pressing issues we face on our planet, yet deniers continue to resist the new reality, and some of them twist the facts. In this tutorial, we'll examine global temperature data from 1880 to the present, downloaded from the NASA, the US National Aeronautics and Space Administration. It shows that the mean global temperature has risen about 1 degree Celsius (or about 2 degrees Fahrenheit) during the past fifty years, and this warming has already begun to cause glacial melt and

rising sea levels. Your task is to create *misleading* line charts that *diminish* the appearance of rising global temperature change in the reader's eye.²

1. Open the global temperature change 1880-2019 data in Google Sheets, and go to *File > Make a Copy* to create a version you can edit in your own Google Drive.
2. Examine the data and read the notes. Temperature change refers to the mean global land-ocean surface temperature in degrees Celsius, estimated from many samples around the earth, relative to the temperature in 1951-1980, about 14°C (or 57°F). In other words, the 0.98 value for 2019 means that global temperatures were about 1°C above normal that year. Scientists define the 1951-80 period as "normal" based on standards from NASA and the US National Weather Service, and also because it's a familiar reference for many of today's adults who grew up during those decades. While there's other ways to measure temperature change, this data from NASA's Goddard Institute for Space Studies (NASA/GISS) is generally consistent with data compiled by other scientists at the Climatic Research Unit and the National Oceanic and Atmospheric Administration (NOAA).
3. Create a Google Sheets line chart by selecting the two columns in the *data* sheet, then *Insert > Chart*. The tool should recognize your time-series data and produce a *default* line chart, though we never automatically accept it as the best representation of the truth. Click on the three-dot kebab menu to open the *Chart editor* and select the *Customize* tab. Add a better title and vertical axis label, using the notes to clarify the source and how temperature change is measured, as shown in Figure 15.2.

Now let's create three more charts using the same data but different methods, and discuss why they are *not wrong* from a technical perspective, but nevertheless *very misleading*.

Lengthen the vertical axis to flatten the line

We'll use the same method as shown in the Exaggerate Change in Charts section above, but in the opposite direction. In the Google Sheets chart editor, customize the vertical axis by changing the minimum value to negative 5 and the maximum to positive 5, as shown in Figure 15.3. By increasing the length of the vertical scale, you flattened our perception of the rising line, and cancelled our climate emergency... but not really.

²The tutorial on misleading climate change data was inspired by a high school classroom activity created by the NASA Jet Propulsion Laboratory (JPL), as well as Alberto Cairo's analysis of charts by climate change deniers. NASA JPL; Cairo, *How Charts Lie*, 2019, pp. 65-67, 135-141.

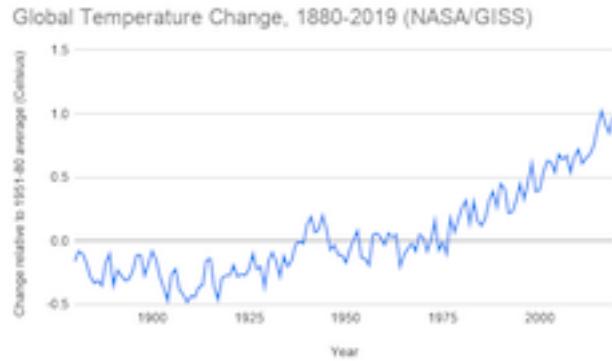


Figure 15.2: Default line chart of global temperature change. Explore the interactive version.

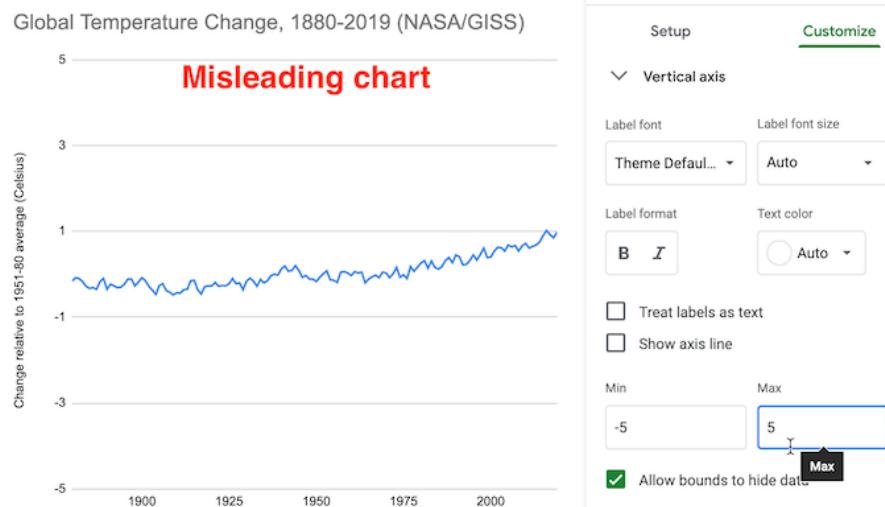


Figure 15.3: Misleading chart with a lengthened vertical axis.

What makes this flattened line chart *misleading* rather than *wrong*? In the first half of the tutorial, when you reduced the vertical axis of the US GDP chart, you violated the zero-baseline rule, because column and bar charts *must* begin at zero since they require readers to judge *height* and *length*, as described in the chart design section of Chapter 7. But you may be surprised to learn that the zero-baseline rule does *not* apply to line charts. Visualization expert Albert Cairo reminds us that line charts represent values in the *position* and *angle* of the line. Readers interpret the meaning of line charts by their shape, rather than their height, so the baseline is irrelevant. Therefore, flattening the line chart for temperature change may mislead readers, but it's technically not wrong, as long as it is labelled correctly.³

Widen the chart to warp its aspect ratio

In your Google Sheet, click the chart and drag the sides to make it very short and wide, as shown in Figure 15.4. Image measurements as listed in width by height, and we calculate the aspect ratio as width divided by height. Since the default chart is 600 x 370 pixels, its aspect ratio is about 1.6 to 1. But the stretched-out chart is 1090 x 191 pixels, and its ratio is about 5.7 to 1. By increasing the aspect ratio, you have flattened our perception of the rising line, and cancelled our climate crisis once again... but not really.

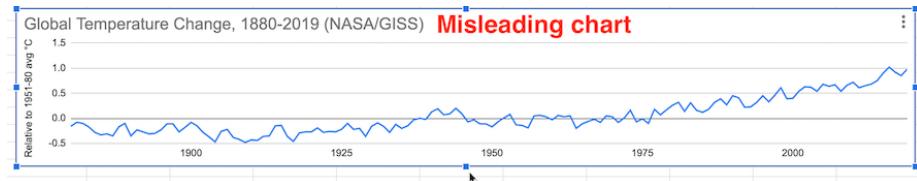


Figure 15.4: Misleading chart with a stretched aspect ratio.

What makes this warped line chart *misleading* rather than *wrong*? Once again, since changing the aspect ratio of a line chart does not violate a clearly-defined rule of data visualization, it's not technically wrong, as long as it's accurately labeled. But it's definitely misleading. Cairo states that we should design charts with an aspect ratio that "neither exaggerates nor minimizes change." What specifically does he suggest? Cairo recommends, yet clearly states this "isn't a universal rule of chart design," that the percent change expressed in a chart should roughly match its aspect ratio. For example, if a chart represents a 33 percent increase, which is the same as $33/100$ or $1/3$, he recommends an aspect ratio of 3:1 (because the fraction is flipped by placing width before height), or in other words, a line chart that is three times wider than its height.⁴

³Cairo, p. 61.

⁴Cairo, p. 69.

But Cairo does *not* propose his aspect ratio recommendation as a universal rule because he recognizes how it fails with very small or very large values. For example, if we apply Cairo's recommendation to our global temperature change chart, the difference between the lowest and highest values (-0.5° to 1°C) represents a 300% increase. In this case, we calculate the percent change using the lowest value of -0.5°C , rather than the initial value of 0°C , because dividing by zero is not defined, so $(1^{\circ}\text{C} - -0.5^{\circ}\text{C}) / |-0.5^{\circ}\text{C}| = 3 = 300\%$. Following Cairo's general recommendation, a 300% increase suggests a 1:3 aspect ratio, or a line chart three times taller than its width, as shown in Figure 15.5. While this very tall chart is technically correct, it's *misleading* because it *exaggerates change*, which is contrary to Cairo's main message. The aspect ratio recommendation becomes ridiculous when we divide by numbers that are very close to zero.

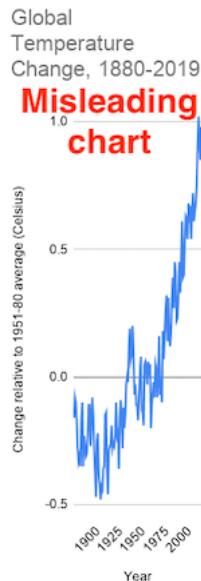


Figure 15.5: Following Cairo's “recommendation” for a 300% increase results in a 1:3 aspect ratio that exaggerates change.

Cairo acknowledges that his aspect ratio recommendation also can result in *misleading* charts in the opposite way that *diminish change*. For example, instead of *global temperature change*, which increased from 0° to 1°C , imagine a chart that displays *global temperature*, which increased from about 13° to 14°C (or about 55° to 57°F) over time. Even though a 1°C difference in average global temperature may not *feel* very significant to our bodies, it has dramatic consequences for the Earth. We can calculate the percent change as: $(14^{\circ}\text{C} - 13^{\circ}\text{C}) / 13^{\circ}\text{C} = 0.08 = 8\%$ percent increase, or about $1/12$. This translates into a 12:1 aspect ratio, or a line chart that is twelve times wider than it is tall,

as shown in Figure 15.6. Cairo warns that this significant global temperature increase looks “deceptively small,” so he cautions against using his aspect ratio recommendation in all cases.⁵



Figure 15.6: Following Cairo’s “recommendation” for an 8% increase results in a 12:1 aspect ratio that diminishes change.

Where does all of this leave us? If you feel confused, that’s because data visualization has *no universal rule about aspect ratios*. What should you do? First, never automatically accept the default chart. Second, explore how different aspect ratios affect its appearance. Finally, even Cairo argues that you should use own judgment and ignore his recommendation in several cases, because there is no single rule about aspect ratio that fits all circumstances.

Add more data and a dual vertical axis

Another common way to mislead is to add more data, such as a second data series that corresponds to a second vertical axis on the right side of a line chart. While it’s technically possible to construct a dual-axis chart, we strongly advise against them because they can easily be manipulated to mislead readers. Let’s illustrate how with an example that combines two prior datasets—global temperature change and US Gross Domestic Product—in one dual-axis chart. In the Google Sheet, go to the *temp+GDP* sheet, where you will see temperature change plus a new column: US Gross Domestic Product (GDP) in billions of dollars from 1929 to 2019, downloaded from the US Federal Reserve. To simplify this example, we deleted pre-1929 temperature data to match it up more neatly with available GDP data.

1. Select all three columns and *Insert > Chart* to produce a default line chart with two data series: temperature (in blue) and US GDP (in red).
2. In the *Chart editor*, select *Customize* and scroll down to *Series*. Change the drop-down menu from *Apply to all series* to *US GDP*. Just below that in the *Format* area, change the *Axis* menu from *Left axis* to *Right Axis*, which creates another vertical axis on the right side of the chart, connected only to the US GDP data, as shown in Figure 15.7.

⁵Cairo, p. 70.

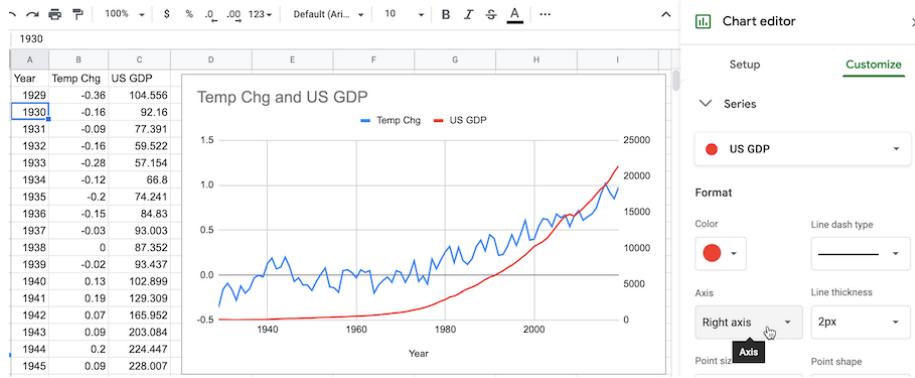


Figure 15.7: Add another vertical axis to the right side of the chart.

3. In the *Chart editor* > *Customize* tab, scroll down and you will now see separate controls for *Vertical Axis* (the left side, for temperature change only), and a brand-new menu for the *Right Axis* (for US GDP only), as shown in Figure 15.8.

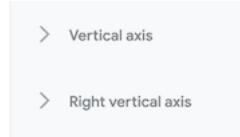


Figure 15.8: Brand-new menu for the right axis.

4. Finish your chart by adjusting *Vertical Axis* for temperature change, but with even more exaggeration than you did in step 5 above. This time, change the minimum value to 0 (to match the right-axis baseline for US GDP) and the maximum to 10, to flatten the temperature line even further. Add a title, source, and labels to make it look more authoritative, as shown in Figure 15.9.

What makes this dual axis chart *misleading* rather than *wrong*? Once again, since it does not violate a clearly-defined visualization design rule, the chart is not wrong. But many visualization experts strongly advise against dual-axis charts because they confuse most readers, do not clearly show relationships between two variables, and sometimes lead to mischief. Although both axes begin at zero in Figure 15.9, the left-side temperature scale has a top level of 10°C, which is unreasonable since the temperature line rises only 1°C. Therefore, by lowering our perception of the temperature line in comparison to the steadily rising GDP line, you've misled us into ignoring the consequences of climate

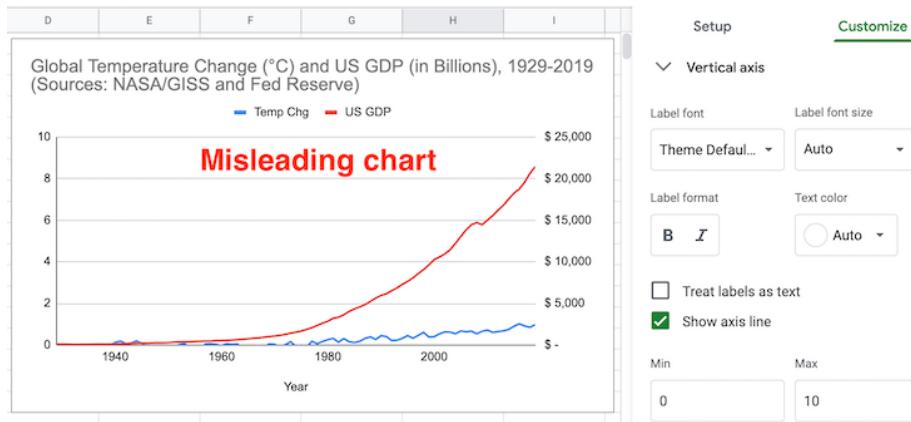


Figure 15.9: Misleading dual-axis chart of US GDP and global temperature change.

change while we enjoy a long-term economic boom! Two additional issues also make this chart problematic. Since the GDP data is *not* adjusted for inflation, its misleads us by comparing 1929 dollars to 2019 dollars, a topic we warned about in Chapter 6: Make Meaningful Comparisons. Furthermore, by accepting default colors assigned by Google Sheets, the climate data is displayed in a “cool” blue, which sends our brain the opposite message of rising temperatures and glacial melt. To sum it up, this chart misleads in three ways: an unreasonable vertical axis, non-comparable data, and color choice.

What’s a better alternative to a dual-axis line chart? If your goal is to visualize the relationship between two variables—global temperature and US GDP—then display them in a scatter chart, as we introduced in chapter 7. We can make a more meaningful comparison by plotting real US GDP, which has been adjusted into constant 2012 dollars, and entered alongside global temperature change in this Google Sheet. While a scatter chart does not show time in the same way as a line chart, float your cursor over points to see years in the interactive version of the Datawrapper scatter chart as shown in Figure 15.10. Overall, the growth of the US economy is strongly associated with rising global temperature change from 1929 to the present. Furthermore, it’s harder to mislead readers with a scatter chart because the axes are designed to display the full range of data, and our reading of the strength of the relationship is not tied to the aspect ratio.

To sum up, in this tutorial we created several charts about global temperature change. None of them were technically wrong, only some were truthful, but most were unreasonably manipulated to fool readers by hiding or disguising important patterns in the data. We demonstrated several ways that charts can be designed to deceive readers, but did not exhaust all of the options. For example, see additional readings on ways to create three-dimensional charts and to tilt the reader’s perspective below the baseline, which causes readers to

Relationship between US Real GDP and Global Temperature Change, 1929-2019

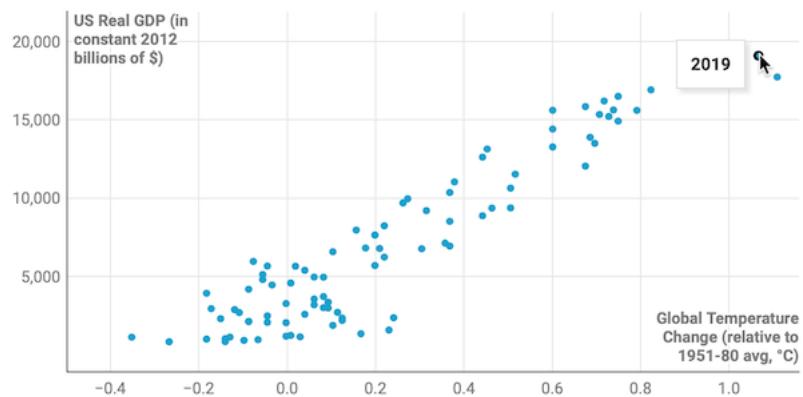


Chart: by HandsOnDataViz • Source: [Federal Reserve and NASA/GISS](#) • [Get the data](#) • Created with Datawrapper

Figure 15.10: Scatter chart of relationship between real US GDP and global temperature change from 1929 to 2019. Explore the interactive version.

misjudge the relative height of column or line charts.⁶

You may feel frustrated that data visualization lacks clearly-defined design rules for many cases, like we are accustomed to reading in our math, science, or grammar textbooks. Instead, remember that the important visualization rule is a *three-step process*: never automatically accept the default, explore how different designs affect the appearance of your interpretation, and use your best judgement to tell true and meaningful data stories.

Now that you've learned about how to lie with charts, in the next section you'll build on these skills to lie with maps.

How to Lie with Maps

One of the best ways to learn how to detect lies is to intentionally manipulate a map, and tell two (or more) opposing stories with the same data. You'll learn what to watch out for when viewing other people's maps, and think more carefully about the ethical issues when you design your own. We'll focus our attention on choropleth maps that use shading or color to represent values in geographic areas, because they are a topic of considerable mischief. This exercise was inspired by geographer Mark Monmonier's classic book by the same name,

⁶Cairo, p. 58.

How to Lie with Maps, originally published in 1991, now in its third edition.⁷

Before we get started, review the map design principles in Chapter 8 to avoid common mistakes when designing choropleth maps. For example, in most cases you should avoid mapping raw counts (such as the total number of people with a disease) and instead show relative rates (such as the percentage of people with a disease), because a raw count map would generally show that most people live in urban rather than rural areas. Also, this section assumes that you're already familiar with the steps for creating a Choropleth map with Datawrapper in Chapter 8.

Let's return to the two maps in the Introduction of this book, where we presented two different interpretations of world income inequality. In particular, Figure 1.3 colored the US in medium blue which suggested its level of inequality was similar to other nations, while Figure 1.4 made the US stand out in dark blue at the highest tier of inequality. We argued that both were *truthful* interpretations. You'll understand the concepts more clearly by following this hands-on tutorial to recreate both maps, plus one more. First, let's examine the data and upload it to Datawrapper to start making our choropleth maps.

1. Open the world income top 1 percent data in Google Sheets, and go to *File > Make a Copy* to create a version that you can edit in your own Google Drive.
2. Examine the data and read the notes. Overall, this data offers one way to make international comparisons about income distribution by showing “how big a slice of the pie” is held by the richest 1 percent in each nation. Each row lists a nation and its three-letter code, along with the percent share of pre-tax national income held by the top 1 percent of the population, and the most recent year when this data was collected by the World Inequality Database. For example, in Brazil, the top 1 percent of the population held 28.3 percent of the nation’s income in 2015, while in the United States, the top 1 percent held 20.5 percent in 2018.

Note: To be clear, social scientists have developed many other ways to compare the distribution of income or wealth across nations, and this topic is beyond the scope of this book. In this tutorial we capture this complex concept using one easy-to-understand variable: percent share of pre-tax national income held by the top 1 percent of the population in each nation.

3. Since we cannot directly import this Google Sheet into our Datawrapper mapping tool, go to *File > Download* to export the first tab in *CSV format* to your computer.

⁷Monmonier, *How to Lie with Maps, Third Edition*.

4. Open the Datawrapper visualization tool in your browser and upload your CSV map data. Select *New Map*, select *Choropleth map*, and select *World*, then *Proceed*. In the *Add your data* screen, scroll down below the table and select the *Import your dataset* button, then the *Start Import* button, then *click here to upload a CSV file*, and upload the CSV file you created in the step above. Click to confirm that the first column is *Matched as ISO code*, click *Continue*, then click to confirm that the *Percent Share* column is *Matched as Values*, then click *Go* and *Proceed* to visualize your map.
5. In the *Visualize* screen, in the *Colors* section of the *Refine* tab *Select palette*, click the *wrench symbol* to open up the color settings, as shown in Figure 15.11. Let's skip past the light-green-to-blue color palette, which you can modify later, and let's focus on settings for color ranges.

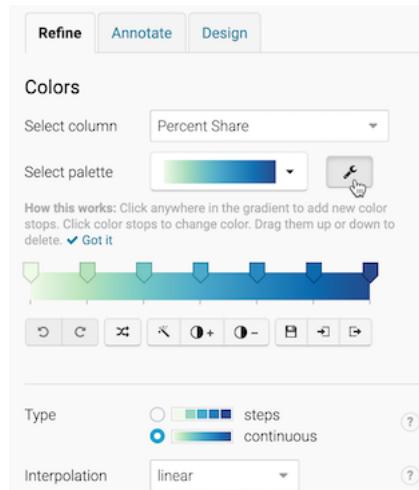


Figure 15.11: Click the *wrench symbol* to open the color settings.

Modify the map color ranges

While we never automatically accept the default visualization, it's a good place to begin. The default map displays a *continuous* type of range, with a *linear* interpolation of data values. This means that the map places all of the values in a straight line, from the minimum of 5% to the maximum of 31%, and assigns each value to a color along the gradient, as shown in Figure 15.12. Notice that the US (20.5%) blends in with a medium blue color, just above the midpoint in this range.

Create a second map with the same data but different settings. Change the *Type* setting to *steps*, and adjust to 3 steps, using *Natural breaks (Jenks)* in-

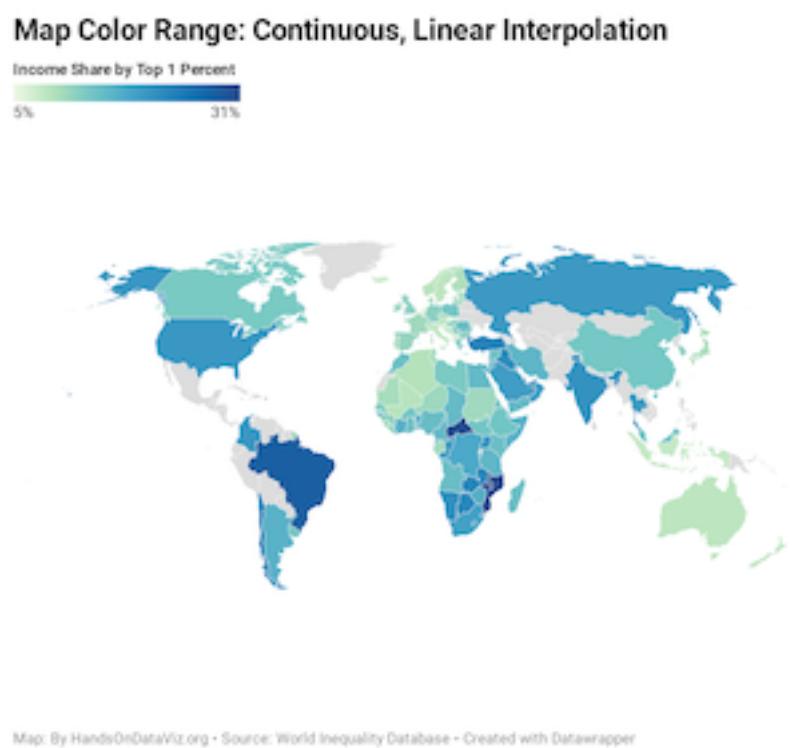
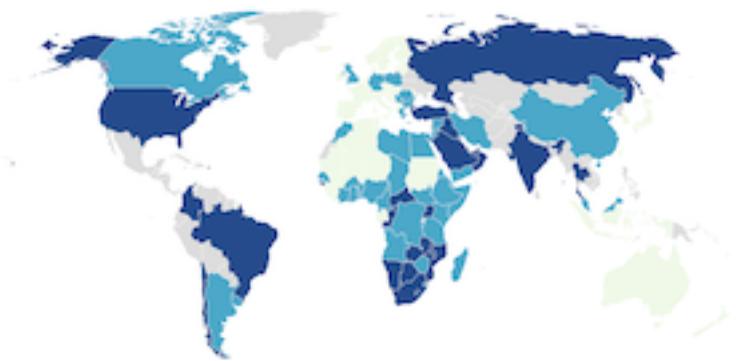


Figure 15.12: Income inequality map with continuous range and linear interpolation. Explore the interactive version.

terpolation, as shown in Figure 15.13. This means that the map now places all of the values in three ascending groups. Natural breaks offers a compromise between using colors to highlight the outliers versus diversity inside the range. Notice that the US (still 20.5%) now stands out in a dark blue color at the top third of this range (19% or above).

Map Color Range: 3 Steps, Natural Breaks



Map: By HandsOnDataViz.org • Source: World Inequality Database • Created with Datawrapper

Figure 15.13: Income inequality map with 3 steps and natural breaks interpolation. Explore the interactive version.

The first map portrays US income inequality to be similar to most nations, while the second map places the US at the higher end of the color scale. Which map is misleading? Which one is truthful? If you prefer clear and definitive rules in map design, this answer may frustrate you. Although the two maps generate very different impressions in our eyes, both maps present accurate data that is clearly labeled, based on reasonable and truthful interpretations of the data.

To understand what's happening behind the scenes with your choropleth map, visualization expert Alberto Cairo recommends creating a histogram to better understand the data distribution. Go back to the data in the Google Sheet and create a histogram, as we described in chapter 7 to view the frequency of nations when sorted by percent share into "buckets", as shown in Figure 15.14. While most nations are clumped around the median, this is not a normal distribution

curve, because a handful are outliers near the 30 percent mark. In the first map, which used continuous type and linear interpolation, the US appeared closer to the median and blended in with a medium blue. By contrast, the second map used 3 steps and natural breaks, which meant that the US appeared in the top range and stood out in dark blue.

Histogram: Frequency of Nations, Sorted by Percent Share

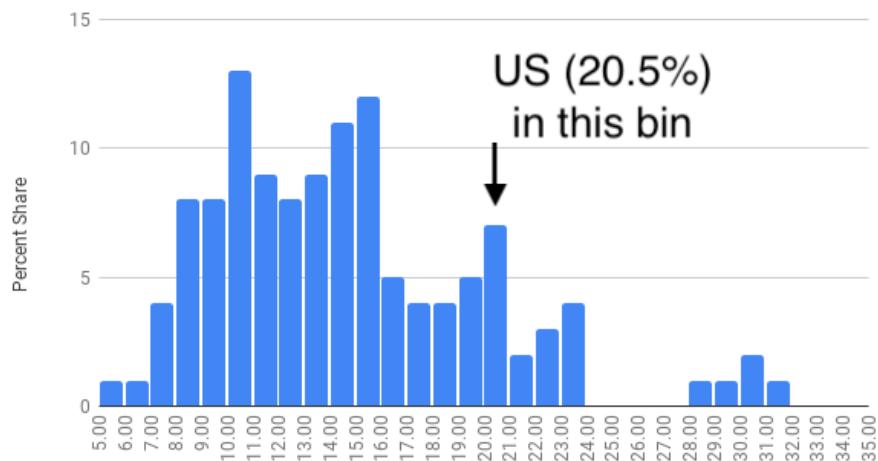


Figure 15.14: Histogram of income inequality map data.

So how *should* we make decisions when designing choropleth maps? Similar to the chart section, there are few universal rules, but several wise recommendations. First and foremost, always look for *better* ways to use map color ranges to show true and meaningful differences in the data, rather than hiding them out of sight. Datawrapper Academy recommends finding “a compromise between honesty and usefulness” when creating choropleth maps. In other words, tell the truth when displaying evidence *and* use design choices to emphasize an interpretation that calls our attention to what’s most important in the data story. For example, a *linear* interpolation works best to emphasize extreme lows and highs, while *quantiles* or other non-linear groupings reveal more geographic diversity in the middle ranges. Datawrapper Academy also recommends using a *continuous* color palette to show nuances in the data, unless your data story has a compelling reason to display discrete *steps* to emphasize regions above or below certain thresholds. If you choose steps, increasing the *number of steps* will display more contrast in your map, but *too many steps* can give the mistaken impression that light- and dark-colored regions are very different, when in fact their numbers may vary only slightly. Whatever you decide, avoid the temptation to manually adjust a map’s settings in ways that manipulate its appearance to fit a preconceived point of view. In sum, show us a story and

tell the truth. You may need to create several maps with different settings to decide which one is the best compromise.

Now that you have a clearer idea of how to lie with charts and maps, let's examine a related topic: recognizing and reducing data bias.

Recognize and Reduce Data Bias

We define bias as unfairly favoring one view over another. When working with data and designing visualizations, it's important to be aware of different types of bias, so that you can recognize them as potential factors that may influence your perception, and reduce their presence in your own work. The first step toward reducing bias is to correctly identify various types, which at first glance may appear hidden, so that we can call them out. In this section we'll discuss four categories of bias that anyone who works with data needs to recognize: sampling biases, cognitive biases, algorithmic biases, and intergroup biases. In a later section we'll address other types of biases that are highly relevant to anyone working with map data.

We previously warned you about *sampling biases* in Chapter 6: Make Meaningful Comparisons. This category covers data collection procedures that seem legitimate on the surface, but include partially-hidden processes that skew the evidence and yield inaccurate results. One example is *selection bias*, which means that the sample selected for your study differs systematically from the larger population. “What you see depends on where you look,” cautions professors Carol Bergstrom and Jevin West, authors of a book with an unforgettable title, *Calling Bullshit*.⁸ One obvious example is to attempting to find the average person’s height by measuring people who happen to be leaving a gymnasium after basketball practice. More subtle forms involve *participation bias*, such as survey methods that do not yield responses that are representative of the broader population. We also cautioned about *self-selection bias* in program evaluation data, where participants who apply or volunteer must be analyzed carefully to avoid comparisons with non-participants. Always question your data, as described in chapter 4, before you attempt to make meaningful comparisons. If you suspect that sampling issue may have snuck into the data collection process, either do not use the data, or clearly describe your concerns in your visualization notes and companion text to call out potential biases.

Cognitive biases refer to a category of human behaviors that skew how we interpret data. *Confirmation bias* is one example, which refers to the tendency to accept only claims that fit our preconceived notions of how the world works. *Pattern bias* is another example, where people tend to see meaningful relationships in data, even when numbers were randomly selected. Counter these

⁸Carl T. Bergstrom and Jevin D. West, *Calling Bullshit: The Art of Skepticism in a Data-Driven World* (Random House, 2020), https://www.google.com/books/edition/Calling_Bullshit/Plu9DwAAQBAJ, pp. 79, 104-133

biases by actively searching for alternative interpretations and considering contradictory findings with open eyes. Remind readers (and yourself) that data is noisy, and our brains are wired to see patterns even when none exist. See additional resources on statistical analysis mentioned in chapter 6 to learn about appropriate tests to determine whether patterns exist in your data at odds greater than chance.

TODO: FINISH REWRITE with transition....

Just as people can lie with charts and maps, let's not forget our long history of misleading audiences (and ourselves) with the word choices we make when describing data. *Framing bias* refers to negative or positive labels or conceptual categories that affect how we interpret information. For example, British statistician David Spiegelhalter notes that US hospitals tend to report *mortality rates*, while UK hospitals report *survival rates*. When weighing the risks of a surgical procedure for member of your family, a 5 percent mortality rate seems worse than a 95 percent survival rate, even though they're identical. Furthermore, Spiegelhalter observes that when we supplement rates with raw counts, it further increases our impression of risks. For example, if we told you a surgical procedure had a 5 percent mortality rate *and* that 20 out of 400 patients died, it seems worse because we begin to imagine real people's lives, not abstract percentages.⁹ The best way to counter framing bias is to be aware of its potential effect on our minds and to call it out, as we've attempted to do here.

TODO: FINISH REWRITE.... Also beware of *algorithmic bias* that is built into our software, ranging from simple examples (such as web visitor IP addresses being converted into the nation's geographic center) and more dangerous ones (TODO EXAMPLE from Ch6).... TODO INSERT AND EXPAND HERE? Also beware of *algorithmic bias* that people have built into our computer systems, which repeatedly favor some groups or outcomes over others, and often reinforce privileges held by dominant White, wealthy, masculine culture.... As we write this, several examples of algorithmic bias and machine-learning bias have appeared in the news. facial recognition across racial groups, or discrimination in home lending. examples: <https://www.nytimes.com/2019/08/20/upshot/housing-discrimination-algorithms-hud.html>; <https://www.brookings.edu/blog/techtank/2020/04/16/why-a-proposed-hud-rule-could-worsen-algorithm-driven-housing-discrimination/> Reduce bias by calling it out. Do not equate "digital" with "authoritative." Cite more comprehensive books on this topic.

Intergroup bias refers to multiple ways that people privilege or discriminate by social categories, such as race, gender, class, sexuality, etc. In the wake of the Black Lives Matter movement, greater attention has been called to ways that intergroup bias pervades data visualization, and ways to counter its impact. Jonathan Schwabish and Alice Feng describe how they applied a racial equity

⁹David Spiegelhalter, *The Art of Statistics: Learning from Data* (Penguin UK, 2019), https://www.google.com/books/edition/The_Art_of_Statistics/CiZeDwAAQBAJ, pp. 22-5

lens to revise the Urban Institute’s Data Visualization Style Guide with a racial equity lens.¹⁰ Some recommendations are straightforward and relatively simple to implement. For example, they recommend ordering group labels to focus on the data story, rather than listing “White” and “Men” at the top by default. Also, we should proactively acknowledge missing groups in our data by calling attention to those often omitted, such as non-binary and transgender people in US federal datasets. Furthermore, when choosing color palettes to represent people in charts and maps, avoid stereotypical colors (such as blue for men and pink for women), and on a more subtle level, avoid color-grouping Black, Latino, and Asian people as the polar opposites of White people.

Other proposals by Schwabish and Feng are likely to generate more discussion and debate. For example, they recommend to stop placing disaggregated racial and ethnic data on the same chart, because it encourages a “deficit-based perspective” that judges lower-performing groups by the standards of higher-performing ones. Instead, they suggest plotting data about racial and ethnic groups on separate but adjacent charts, each with its own reference to state or national averages for comparison, as shown in Figure 15.15. The idea is interesting, but the example about Covid-19 pandemic data raises more questions about whose interests are served by revising how data is visualized. On one hand, if predominantly White audiences perceive racial disparities in Covid data to be caused by *group behavior*, then it makes sense to stop feeding racist stereotypes and no longer compare different groups in the same chart. On the other hand, if these racial disparities are caused in part by *structural obstacles* to quality jobs, housing, and health care, then do separate charts make it harder to identify and challenge the roots of systemic racism? Schwabish and Feng raise important issues for deeper reflection. Yet once again, data visualization is not always driven by clearly-defined design rules. Instead, our mission is to find *better* ways to tell true and meaningful data stories, while working to identify and reduce bias all around us.

TODO above: DECIDE if description and critique of Schwabish and Feng is clear, interesting, and feasible with or without image (which probably would need to be redone and simplified).

Now that we’ve introduced various types of bias to consider when working with data visualization in general, in the next section we’ll focus on two additional types of bias that are specific to mapping.

¹⁰ Jonathan Schwabish and Alice Feng, “Applying Racial Equity Awareness in Data Visualization,” preprint (Open Science Framework, August 27, 2020), <https://doi.org/10.31219/osf.io/x8tbw>. See also this web post summary of the paper, Jonathan Schwabish and Alice Feng, “Applying Racial Equity Awareness in Data Visualization,” Medium, accessed October 16, 2020, https://medium.com/@urban_institute/applying-racial-equity-awareness-in-data-visualization-bd359bf7a7ff. Urban Institute, “Urban Institute Data Visualization Style Guide,” 2020, <http://urbaninstitute.github.io/graphics-styleguide/>



Figure 15.15: Schwabish and Feng recommend to stop placing racial and ethnic data on the same chart (left), and replace it with separate but adjacent charts with state or national averages as a comparison point (right).

Map Area and Projection Bias

Two additional types of bias that are specific to spatial visualizations are *map area bias* and *projection bias*, and beware of both types when creating choropleth maps, as described earlier in this chapter. Map area bias refers to the tendency for our eyes to focus primarily on larger regions on a map, and less on smaller ones. This bias diverts our attention to *geographic area* rather than *population size*, which is usually the more relevant common denominator in choropleth maps. A classic example arises every four years during US presidential elections. Conventional maps of US electoral votes exaggerate the influence of rural states with larger geographic areas (such as spacious Wyoming with less than 600,000 people), and diminish the influence of urban states with small areas (such as tiny Rhode Island with over 1,000,000 people). Although Wyoming covers 80 times more area than Rhode Island, it currently has only 3 electoral votes, while Rhode Island has 4. Yet many people cannot make this distinction while looking at a conventional electoral map, because our eyes tend to focus on states with larger geographic areas.

A related problem is *projection bias*. In order to portray a three-dimensional globe on a flat surface, geographers have developed different projection systems, and some of these, such as Mercator maps, inflate the size of nations located further away from the equator, which mistakenly gives the appearance that many North American countries (such as the United States and Russia) are more important than those in Central Africa. [TODO: check wording and describe how the ubiquitous Google Maps WGS84 standard compares, which I believe is still a pseudo-Mercator system: https://en.wikipedia.org/wiki/Web_Mercator_projection]. For an interactive visual depiction of this issue, see <http://googlemapsmania.blogspot.com/2020/09/how-map-projections-lie.html>

Note: Also beware of *contested territory bias* in several popular digital map tile services. For example, Google Maps displays different borders and map data depending on the internet address of the user. If you look at location X from a computer in China, it will show AAA, but if you look at the same location from a computer in Taiwan, it will display BBB. [TODO: Find this cite and complete the example]

One solution to both the map area and projection bias problem is to replace conventional map outlines with *cartograms* (sometimes called *population square* or *hexagon* maps). Cartograms display geographic regions by relative population size, rather than total area, and also do not rely on a projection system. One drawback is that cartograms require readers to recognize abstract shapes in place of familiar boundaries, since these population-based visualizations do not align perfectly with conventional geography-based maps, as shown in Figure 15.16.

TODO: Update maps above using 2020 election data in November? Use cartogram/hexagon from Datawrapper on right. TODO above: determine if car-

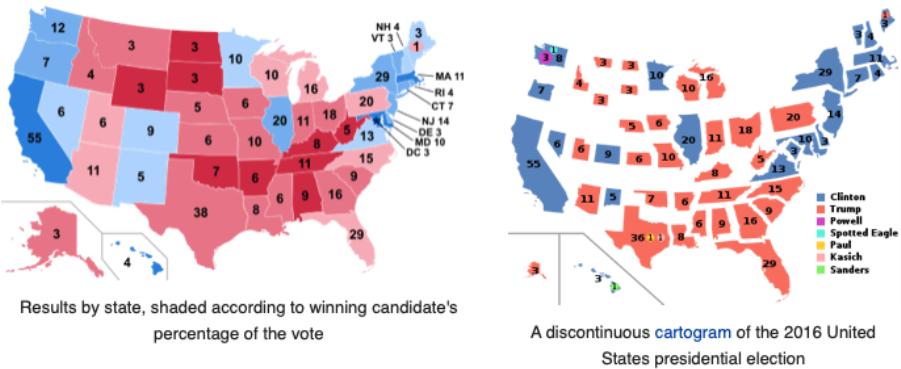


Figure 15.16: PLACEHOLDER: Conventional US map (left) versus cartogram (right) of US 2016 electoral vote.

tograms and pop squares are interchangeable terms, or if they have different definitions.

In the How to Lie with Maps section of this chapter, we created choropleth maps of world inequality data in Datawrapper. To convert one from a conventional world map to a population square map, follow this tutorial:

1. To modify an existing world inequality map that you may have saved in your Datawrapper account, go to *My Charts*, select and right-click on the map to make a duplicate, and edit it. Or follow the steps in the previous section to create a new map.
2. Go to the *Select your map* screen, and type “squares” to see all of those available types (including World population squares). Similarly, type “hexagons” to see all of the cartograms available (including US States). Select your preferred map, and proceed to visualize the data in the same way as other Datawrapper choropleth maps, as shown in Figure 15.17.

The US States bias

When working with data about the United States, consider the additional *framing bias* and *intergroup bias* that frequently causes visualizations to omit over 4 million US citizens. Does your data include the District of Columbia, which is not counted as a state, and whose 700,000 residents (more than Wyoming), a majority of whom are African-American, have no voting representation in the US Congress? Similarly, how does your data represent Puerto Rico, a US territory with over 3 million residents who are US citizens, mostly Spanish-speaking, but have no voting representation in Congress and no electoral votes? How about

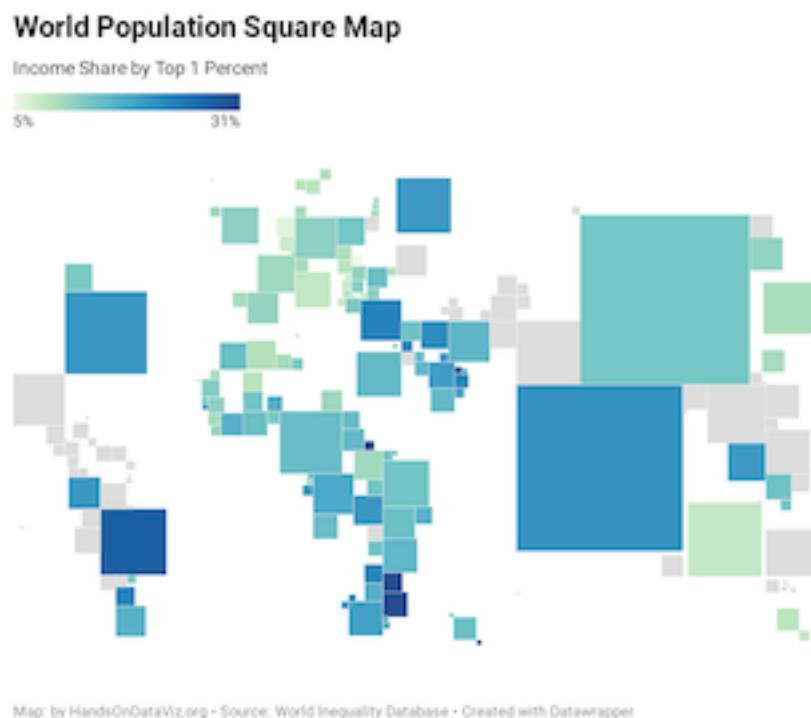


Figure 15.17: World population square map with income inequality data. Explore the interactive version.

other US territories such as the US Virgin Islands, Guam, the Northern Mariana Islands, and American Samoa?

Furthermore, what happens when you create a map of the United States? If your data does include residents of District of Columbia, Puerto Rico, or other US territories, what happens when you try to map it? Do these people become visible—or vanish? Most likely the answer depends on the default settings of your mapping tool, and the geographic outlines it uploads when you select “United States.” If the default setting includes only the 50 US states—even when you have data on DC or US territories—those 4 million US citizens will disappear from the map. And if you cannot easily find a way to map their data, call out the US States bias by describing the omission in the map notes and companion text. Whenever possible, include people of “the United States” rather than ignoring their existence. Tell true and meaningful stories.

TODO: Datawrapper kindly responded to our request for USA » States and Territories map codes and USA » States and Territories (hexagons), so update text and add example to show this.

Summary

TODO

Chapter 16

Tell and Show Your Data Story

In this concluding chapter, we'll draw on knowledge and skills you've developed while reading this book and offer some final recommendations for creating true and meaningful data stories. Here we emphasize *storytelling*. The goal of data visualization is not simply to make pictures about numbers, but also to craft a truthful narrative that convinces readers how and why your interpretation matters.

Writers have an old saying—"show, don't tell"—which means to let readers experience a story through the actions and feelings of its characters, rather than narration by the author. But we take a different stance, as shown in our chapter title: "tell and show" your data story. Make a regular habit of these three steps: tell your audience what you found that's interesting in the data, show them the visual evidence to support your argument, and remind us why it matters. In three words: tell—show—why. Whatever you do, avoid the bad habit of showing lots of pictures and leaving it up to the audience to guess what it all means. Because we rely on you, the storyteller, to guide us on a journey through the data and what aspects deserve our attention. Describe the forest, not every tree, but point out a few special trees as examples to help us understand how different parts of the forest stand out.

In this chapter, you'll learn how to build visualizations into the narrative of the storyboard that we started at the beginning of the book. Also, you will try out ways to draw attention to what's most meaningful in your data through text and color, as well as how to acknowledge sources and uncertainty. Finally, we'll discuss decisions you will need to make about the format of your data story, with our continual emphasis on sharing interactive visualizations rather than static images. Our inspiration for this chapter is drawn from excellent books by

visualization experts Cole Nussbaumer Knaflic and Alberto Cairo.¹

Build a Narrative on a Storyboard

TODO: Or alternative section title “Build Data into Your Storyboard”

Let’s return to the Sketch Your Data Story exercise from Chapter 2. We encouraged you to scribble words and sketch pictures on sheets of paper to lay out at least four initial elements of your story:

- Identify the *problem* that motivates your project.
- Reframe the problem into a researchable *question*.
- Describe your plan to *find* data to answer the question.
- Dream up one or more *visualizations* you might create using imaginary data.

Spread out these sheets like a *storyboard* to define the sequence of your narrative, as shown in Figure 16.1. Imagine them as preliminary slides for your presentation desk, or paragraphs and pictures for your written report or web page, for how you plan to explain the process to your audience. Or if you prefer digital over paper, another option is to convert blocks of text and images from your sheets into a Google Slides presentation or a draft Google Document, or your preferred tools for telling the data story. Of course, it’s perfectly normal to update the sheets you created at the beginning of your project to reflect changes in your thinking. For example, you may have refined your research question, found new sources during your search, and of course, turned your imagined visualizations into actual tables, charts, or maps with real data.

Let’s enrich your storyboard by adding content about what you discovered while searching, cleaning, analyzing, and visualizing your data. Select only your most meaningful tables, charts, or maps. Print them out on separate sheets of paper, or download static images or capture screenshots to place them in your draft slides or document. Leave room for you to write text at the top and bottom of each table, chart, or map in order to tell your data story. Here are the next two steps:

- At the top of each key visualization, *summarize* the most important message the data reveals.

¹Cole Nussbaumer Knaflic, *Storytelling with Data: A Data Visualization Guide for Business Professionals*, 1 edition (Hoboken, New Jersey: Wiley, 2015); Cole Nussbaumer Knaflic, *Storytelling with Data: Let's Practice!* (John Wiley & Sons, 2019), https://www.google.com/books/edition/Storytelling_with_Data/aGatDwAAQBAJ; Cairo, *The Truthful Art*, 2016; Cairo, *How Charts Lie*, 2019



Figure 16.1: Sketch out your story idea on four pages: problem, question, find data, visualize.

Verbalize what your eyes see as the most insightful finding for your most important visualizations.

Write a one-sentence summary of the most insightful finding for each visualization. Verbalize how it appears in your eyes, like our guide to the data forest. Two sentences are acceptable, but one succinct sentence is better. If your prose becomes too wordy, try writing the first sentence in “headline” style and the second as a more descriptive follow-up. Despite the old saying that a picture is worth a thousand words, data visualizations do *not* speak for themselves. Your job is to interpret their meaning for your audience. One of the best ways to translate charts or maps into words is to describe exactly what captures your eye as the designer, and communicate this to your reader, who is seeing it for the first time and relying on your guidance. In every case, you need to decide on the ideal mix of words and images.

- At the bottom of each visualization, tell us *why it matters*, and build up to how audiences should rethink or react.

A good way to discuss the significance of your data story is to focus on how this new information *changes us*. When you discovered interesting patterns in your data visualization, how did it make you feel about the problem you (or your organization) were trying to solve? How did your answers to the research question make you think about the issue in a new or different way? Overall, does your data story inspire you or others to take action in some way? Once again, think about these questions from the perspective of your audience, and find words that capture how the data story should change our mindset, alter our habits, or influence our next steps.

For example, we started our own data storyboard in chapter 2 to define our problem statement: *To learn more about readers' backgrounds in order to write a better data visualization book that meets their needs.* [TODO: Revise earlier chapter text to match?] We collected data from over 3,000 readers of an earlier draft of this book who responded to our online survey and agreed that we would publicly share the survey results, as we discussed in chapter 3. We cleaned up the data as described in chapter 5 because some responses were partially empty or contained locations that could not be accurately geocoded. Then we looked for meaningful comparisons as described in chapter 6 and visualized our most interesting results in two ways. We created a scatter chart as described in chapter 7 and also a point map as described in chapter 8. For this chapter, we followed our own advice above by writing short summaries at the top of each visualization, and explaining why it matters at the bottom.

What did we discover in our survey? First, over 70 percent of readers who responded live outside of North America. Most notably, 35 percent reside in Asia, 20 percent in Europe, 6 percent each in Africa and South America, and 3 percent in Oceania, as shown in the left side of Figure 16.2. Our first draft of the book mostly included examples from Hartford, Connecticut, where we both worked. While we knew that our book had a global audience, we were surprised to see how many readers—among those who responded to the survey—live outside of the United States. In order to be more inclusive and expand our international audience, we have revised the book to add more sample charts and maps from other regions where readers live. [TODO: we need to work more on this!] Second, we learned that readers who responded to our survey have relatively high levels of education, but limited data visualization experience. In particular, 89 percent reported completing the equivalent of a college degree (16 or more years of schooling), and 64% of these rated themselves as data visualization beginners (either 1 or 2 on the 5-point experiential scale), as shown in the right side of Figure 16.2. In our earlier draft of the book, our primary audience were college undergraduates, and we were uncertain about the reading and background levels of other readers. Based on the survey responses, we revised the manuscript to add deeper concepts about data visualization, because we believe most of our readers can grasp them, yet we continue to write at an introductory level that assumes no prior knowledge beyond a secondary school or early college education. Now we can add these new sheets to our storyboard.

Insert your new data visualization sheets (or slides, or blocks of text and images) into your storyboard. As you complete your work, your layout might look something like this:

- problem statement
- research question
- how you found data
- tell 1st data insight—show evidence—why it matters
- tell 2nd data insight—show evidence—why it matters
- ... and so forth toward your summary conclusion

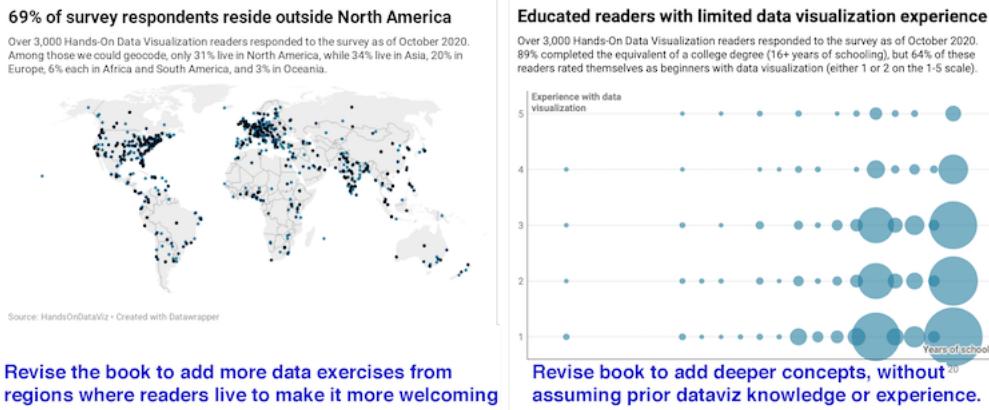


Figure 16.2: Verbalize meaningful insights at the top of each visualization, and tell why it matters at the bottom, then insert them into your storyboard.

But it's your job to organize your data story in the way that makes sense *to your audience*, who most likely will be viewing all of this content for the first time. While there is no one way to tell a story, consider this advice to avoid making rookie mistakes. First, tell us the problem and question *before* you offer an answer, because our brains expect to hear them in that order. Second, summarize each insight *before* you show us the supporting evidence, because once again, reversing the normal sequence makes it harder for us to follow your argument. Finally, make sure that your research question and key insights are *aligned* with one another, since your audience will be confused if you ask one question, but answer a different one. It's perfectly normal to tweak or fully revise the wording of your research question after you've dug deep into the data, because sometimes you don't really know what you're looking for until you've discovered it.

Now you should have a clearer sense of how a storyboard helps you to bring together narrative and data. In the next section, you'll learn how to refine your visualizations by using text and color to draw attention to what is most important.

Draw Attention to Meaning

When finalizing your visualizations, add finishing touches to draw attention to the most meaningful aspects of the data. In addition to writing text to accompany your charts and maps, you can also add annotations and use colors *inside* some types of visualizations to point out what's most significant in your data story. Let's demonstrate how to use these features to transform your visualization in Datawrapper, a tool we first introduced in Chapter 7: Chart

Your Data. This example was inspired by the Datawrapper Academy site.

One of the environmental challenges we face today is the ever-growing production of plastics. While these inexpensive and lightweight materials offer many quality-of-life benefits, we often deposit them in mismanaged waste streams that cause them to enter our rivers and oceans. To understand the growth of plastics, we consulted Our World In Data, and you can view the annual global production data from 1950-2015 in Google Sheets format.

First, let's upload the data in a single-column format to Datawrapper. By default, the tool transforms this time-series data into a line chart, as shown in Figure 16.3, which shows how global plastic production has increased over time.

year	plastics
1950	2
1951	2
...	

Annual Global Production of Plastics, 1950-2015

in millions of metric tons

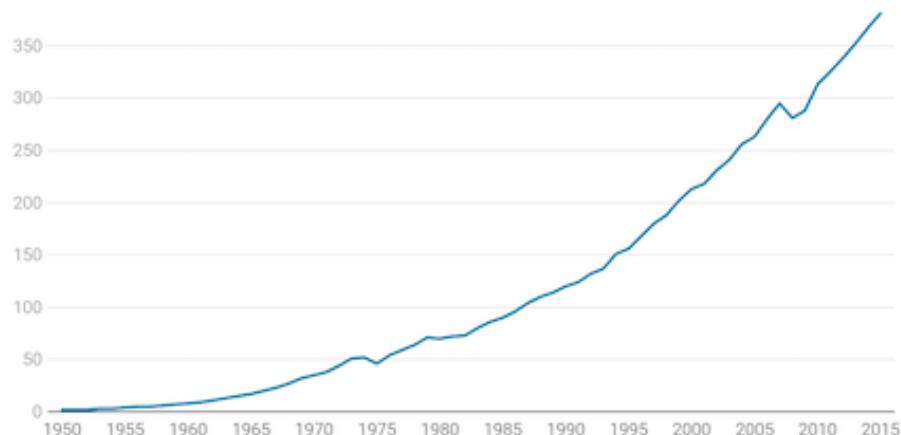


Chart: by HandsOnDataViz.org • Source: Our World In Data; Geyer et al. 2017 • Created with Datawrapper

Figure 16.3: The default GDP line chart on the left, and the reduced-axis chart on the right.

But Figure 16.3 does not yet focus on the bigger story. Our 60 percent of all of the plastics ever manufactured in the world have been made since 2000, or the last 15 years of this chart, according to our analysis of the data. Let's highlight this broader point by editing the chart in four ways, building on skills

you learned in prior chapters. First, divide the data into two columns, *before 2000* and *since 2000*, which allows you to apply different colors to each data series. Insert the same data for year 2000 in both columns to make the new chart look continuous. Second, change the chart type from the default *line chart* to an *area chart* to fill the space under the curve to draw attention to the total amount of plastics manufactured in global history. Third, in the *Refine* tab, since you do *not* want a stacked area chart, uncheck the *stack areas* box. Assign a dark blue color to draw more attention to the post-2000 data series, and a gray color to diminish the appearance of the pre-2000 data series, as shown in Figure 16.3. Finally, hide the old title and replace it by adding annotations inside the area chart, using colored text, to emphasize the new interpretation. Overall, redesigning your chart helps you to communicate a more meaningful data story that global plastic production is increasing *and* that our world has manufactured more than half of our historical total in just the past 15 years.

year	before 2000	since 2000
1999	202	
2000	213	213
2001		218
...		

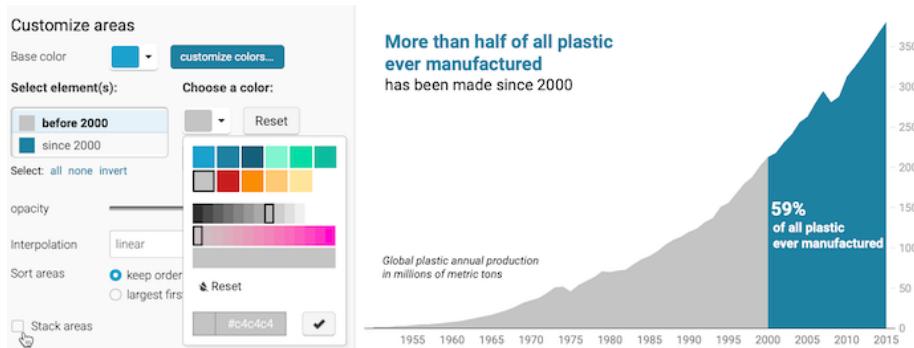


Figure 16.4: Explore the interactive version of the new area chart, which uses color and annotations to draw attention to post-2000 global plastic production.

Now that you have a clearer idea about why and how to draw your audience's attention to the most meaningful aspects of your data story, we'll build on those skills in the next section on acknowledging sources and ambiguous data.

Acknowledge Sources & Uncertainty

Since our goal is to tell data stories that are meaningful and true, build credibility into your work, which you can do in several ways:

First, always represent data truthfully. Do not hide or obscure relevant evidence, and avoid visualization methods that might mislead your audience, as we discussed in Chapter 15 on detecting lies and reducing bias. We place our trust in you to fairly interpret the meaning of the data, and to warn us against misinterpreting to reading too much into it.

Second, credit and source your data origins, as we described in Chapter 4: Find and Question Your Data. Some of the visualization tools and templates featured in this book make it easy to display links to online sources, so use that feature whenever feasible. When it's not, then write these important details into the text that accompanies your tables, charts, and maps. Also, let audiences know who created the visualization, and credit collaborators and other people who assisted in your work.

Third, save and show your data work at different stages of the process. Save notes and copies of the data as you download, clean, or transform it, and document the important decisions you make along the way. One simple method is to save different versions of your data in separate spreadsheet tabs, as shown in Chapter 3. For more complex projects, consider sharing your data and documenting your methods in a public GitHub repository, as shown in chapter 11. If someone questions your work—or if you need to replicate it with updated dataset—you'll be grateful to have notes that allow you to trace it backwards.

Finally, acknowledge the limitations of your data and disclose any uncertainty. Your work becomes more credible when you admit what you do *not* know or consider alternative interpretations. Some of our recommended chart tools in chapter 7 and chart code templates in chapter 12 allow you to insert error bars to show the confidence level in the data, so use those when appropriate. Furthermore, the two-column method shown in the prior section also works to visually distinguish between observed versus project data with solid versus dashed lines, as shown in the Google Sheets chart editor in Figure 16.5.

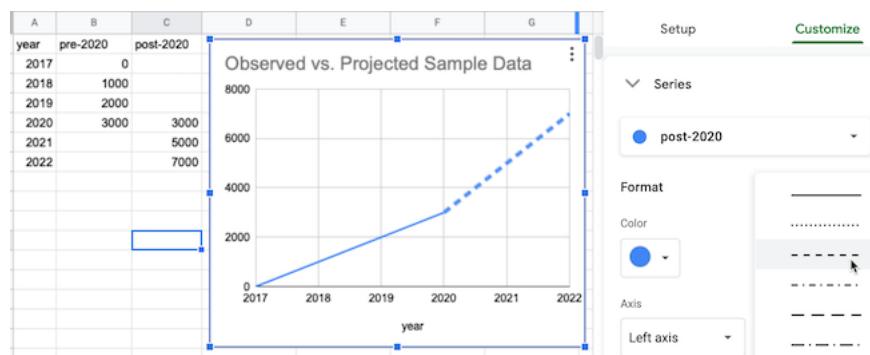


Figure 16.5: Split one data column into two columns to contrast observed data (solid line) versus projected data (dashed line).

Now that we've reviewed ways to build credibility in your work, let's move on to

decisions you'll need to make about telling your data story in different formats.

Decide On Your Data Story Format

Most data visualization books and workshops presume that you will deliver your final product on a sheet of paper to people sitting around a board room, or perhaps in a PDF document sent via email or posted online. Those *static* formats are fine, but do not fully reflect the wide range of ways to share your story with broader audiences in the digital age. Moreover, as we write these words during the Covid-19 pandemic, when sitting around an indoor table is not an option, we need to find more creative formats to communicate our data stories.

Given that our book has emphasized the benefits of creating interactive visualizations, which invites audiences to engage with your data by floating their cursor over the charts and maps, we also encourage you to consider more interactive formats for your stories, such as:

- website that combines textual narrative and interactive visualizations
- online presentation slides that link to live visualizations
- video that combines live or voiceover narration with interactive visualization screencast
- a data walk where community stakeholders move around and discuss connections to the story

Of course, different storytelling methods require you to tailor content to fit the format. Furthermore, not every format requires interactive visualizations, nor are they always the most appropriate choice. While the details are beyond the scope of this book, we encourage you not to fall into traditional mindsets and to think differently about ways to tell true and meaningful data stories.

TODO: DISCUSS whether this chapter ending is appropriate, or if it makes sense to offer more recommendations about ways to design data stories for each of these four non-traditional formats. If so, I saved my notes in the archive folder...

Summary

TODO

Appendix A

Fix Common Mistakes

TODO: Rewrite appendix to focus more broadly on “common mistakes” not just “code errors”

Creating your data visualizations through code templates hosted on GitHub has multiple advantages over drag-and-drop tools. Coding gives you more power to customize their appearance and interactive features, and to control where your data and products reside online. But there’s also a trade-off. Code can “break” and leave you staring at a blank screen. Sometimes problems happens through no fault of your own, such as when a “code dependency” to an online background map or code library is unexpectedly interrupted. But more often it seems that problems arise because we make simple mistakes that break our own code. Whatever the cause, one big drawback of working with code is that you’re also responsible for fixing it.

We designed this section as a guide to help new coders diagnose and solve common errors when working with code templates on GitHub. We understand the feeling you experience when a simple typo—such as a misplaced semicolon (;)—makes your data visualization disappear from the screen. Finding the source of the problem can be very frustrating. But breaking your code—and figuring out how to fix it—also can be a great way to learn, because trial-and-error on a computer often provides immediate feedback that supports the learning process and develops our thinking.

TODO: Reorganize contents, perhaps using this outline?

- Problems with Mac computers
- Problems with data tables
- Problems with iframes (since this chapter appears before code templates)
- Problems with GitHub forking and hosting
- Problems with code templates

Problems with Mac computers: cannot see filename extension

Several tools in this book will not work properly if your computer does not display the filename extensions, meaning the abbreviated file format that appears after the period, such as `data.csv` or `map.geojson`. The Mac computer operating system hides these by default, so you need to turn them on by going to `Finder > Preferences > Advanced`, and check the box to *Show all filename extensions*, as shown in Figure A.1.

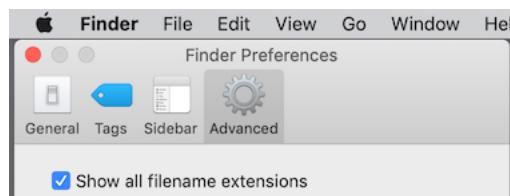


Figure A.1: On a Mac, go to *Finder* then *Preferences* then *Advanced* and check the box to *Show all filename extensions*.

Problems with data tables

Avoid typing blank spaces after column headers—or any spreadsheet entries—since some data visualization tools will not match them with headers lacking a blank character.

	A	B	C	D
1	name	all2015	healthcare2015	manufactur
2	Andover	189	36	67
3	Avon	2536	1866	391
4	Berlin	4907	1159	2822
5	Bloomfield			3688
6	Bolton	488	60	269

Problems with iframes

My iframe does not appear in my web page

- Go back to the Embed tutorials in this book to double-check the directions
- Items listed in your iframe (such as the URL, width, or height) should be enclosed inside straight quotation marks (single or double)
 - BROKEN iframe (missing quotation marks for width and height)

```
<iframe src="https://handsondataviz.github.io/leaflet-map-simple"
width=90% height=350></iframe>
```

– FIXED iframe (with correct quotation marks)

```
<iframe src="https://handsondataviz.github.io/leaflet-map-simple"
width="90%" height="350"></iframe>
```

- Use only `https` (the extra ‘s’ means ‘secure’), not `http`. Some web browsers will block content if it mixes http and https resources, and some code templates in this book require https.

<iframe src="`http://ik`
Change to https

Correct

<iframe src="`https://`

Figure A.2: Screenshot: Replace http with https

- Use only straight quotes, not curly quotes. Avoid pasting text from a word-processor into GitHub, which can accidentally carry over curly quotes. Typing directly into the GitHub editor will create straight quotes.

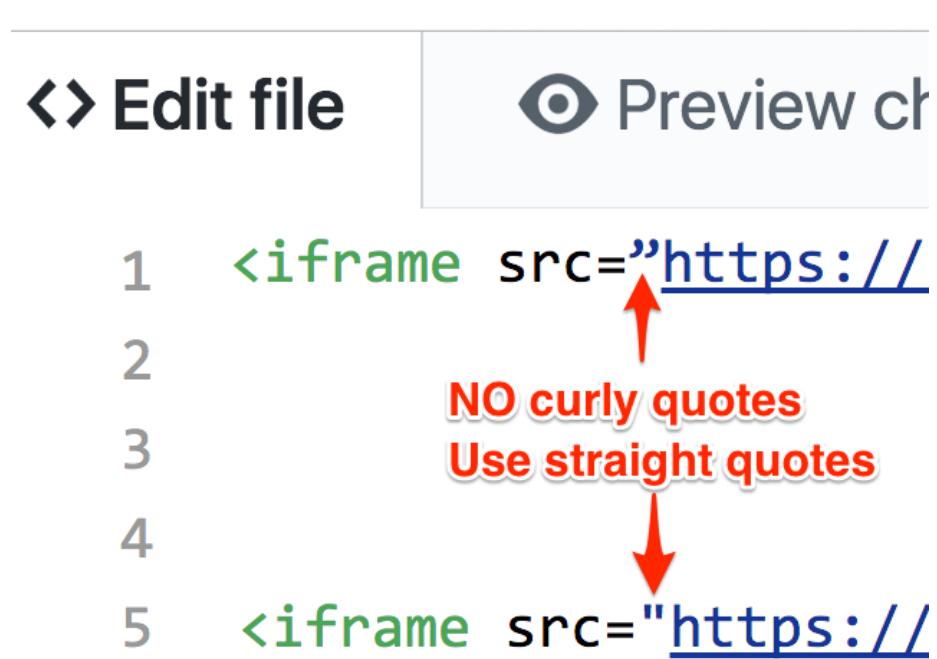


Figure A.3: Screenshot: Curly quotes versus straight quotes

Hint: Either single-quote (') marks (also called an apostrophe) or double-quote ("") marks are acceptable in your iframe code, but be consistent and avoid accidentally pasting in curly-quotes.

TODO: Test one way to fix GitHub errors by going into the commits and going back to a previous version of the code. Is this possible in the web version?

Safely Delete your GitHub Repo and Start Over

If you need to delete your GitHub repo and start over, here's a simple way to safely save your work:

- Go to the top-level of your GitHub repository, similar to <https://github.com/USERNAME/REPOSITORY>
- Click the green “Clone or Download” button, and select Download Zip to receive a compressed folder of your repo contents on your computer.
- In your GitHub repo, click on Settings (upper-right area) and scroll down to Delete This Repository.
- To prevent accidental deletions, GitHub requires you to type in the REPOSITORY name.
- Now you can start over in one of these ways:
 - If you wish to Create a Simple Web Page with GitHub Pages, follow that tutorial again.
 - OR
 - Fork another copy of the original GitHub repository to your account. After you create your copy, if you wish to add selected files that you previously downloaded to your computer, follow directions to Upload Code with GitHub in the second half of this tutorial in this book

Problems with Creating a Simple Web Page with GitHub Pages

If you followed the Create a Simple Web Page with GitHub Pages tutorial, it should have created two web links (or URLs):

- your code repository, in this format: <https://github.com/USERNAME/REPOSITORY>
- your published web page, in this format: <https://USERNAME.github.io/REPOSITORY>

Be sure to insert your GitHub username, and your GitHub repository name, in the general formats above.

These URLs are NOT case-sensitive, which means that <https://github.com/USERNAME> and <https://gitub.com/username> point to the same location.

My simple GitHub web page does not appear

- Make sure that you are pointing to the correct URL for your published web page, in the format shown above.
- Be patient. During busy periods on GitHub, it may take up to 1 minute for new content to appear in your browser.
- **MOVE UP** If your map does *not* appear right away, wait up to 30 seconds for GitHub Pages to finish processing your edits. Then give your browser a “hard refresh” to bypass any saved content in your cache and re-download the entire web page from the server, using one of these key combinations:
 - Ctrl + F5 (most browsers for Windows or Linux)
 - Command + Shift + R (Chrome or Firefox for Mac)
 - Shift + Reload button toolbar (Safari for Mac)
 - Ctrl + Shift + Backspace (on Chromebook)
- Test the link to your published web page in a different browser. If you normally use Chrome, try Firefox.
- On rare occasions, the GitHub service or GitHub Pages feature may be down. Check <https://status.github.com>.

My simple GitHub web page does not display my iframe

- If you followed the Create a Simple Web Page with GitHub Pages tutorial and inserted an iframe in the README.md file, it will appear in your published web page, under these conditions:
 - Ideally, your README.md should be the ONLY file in this GitHub repository
 - Any other files in your repo (such as index.html, default.html, or index.md) will block the iframe HTML code in your README.md from being published on the web. If you accidentally selected a GitHub Pages Theme, you need to delete any extra files it created: click each file, select trash can to delete it, and commit changes.

Problems with Leaflet Maps with Google Sheets template

My map does not appear

- 1) Confirm that you have completed all of the key steps in the Leaflet Maps with Google Sheets tutorial in this book, especially these:

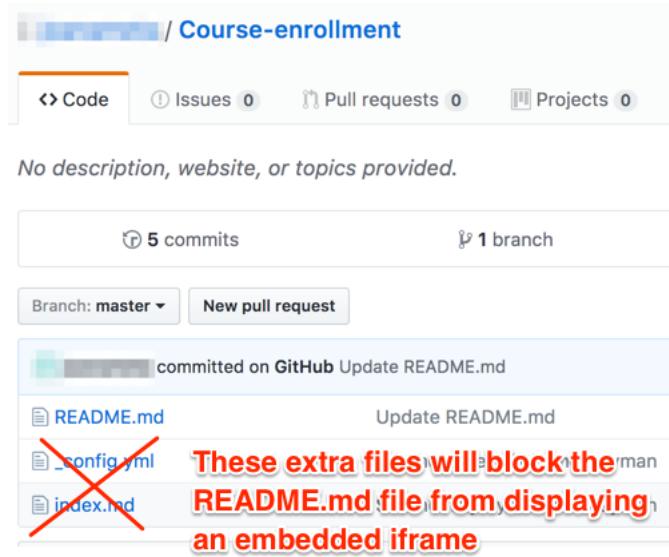


Figure A.4: Screenshot: Extra files in GitHub repo will block iframe in your README

- Sign in to Google and File > Make a Copy of the Google Sheet to your Google Drive.
- File > Publish your Google Sheet (Jack often forgets this key step!)
- Copy your Google Sheet web address from top of your browser (usually ends with ...XYZ/edit#gid=0) and paste into your `google-doc-url.js` file in your GitHub repo. Do NOT copy the *Published* web address (which usually ends with ...XYZ/pubhtml)
- When you paste your Google Sheet web address into `google-doc-url.js`, be careful not to erase single-quote marks or semicolon
- Go to your live map link, which should follow this format: <https://USERNAME.github.io/REPOSITORY>, refresh the browser, and wait at least 30 seconds.

2) Check your Google Sheet for errors:

- Do NOT rename column headers (in row 1) of any sheet, because the Leaflet Map code looks for these exact words.
- Do NOT rename row labels (in column A) of any sheet, due to the same reason above.
- In your Points tab, DO NOT leave any blank rows

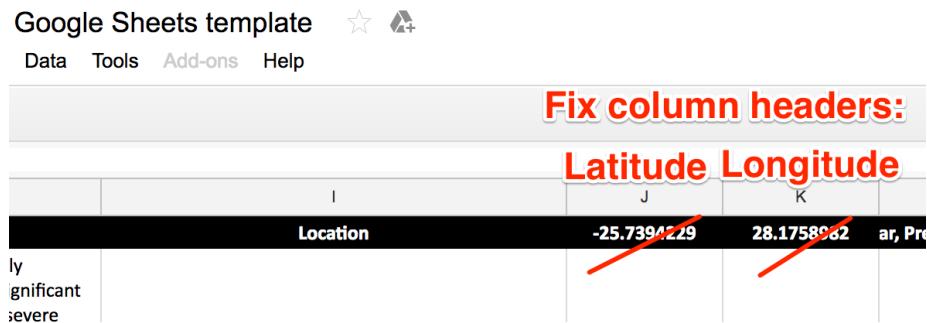


Figure A.5: Screenshot: User accidentally renamed column headers in the Points tab

- 3) Confirm on GitHub Status (<https://status.github.com/>) that all systems are operational.
- 4) If you cannot find the problem, go to the top of this page to Safely Delete Your GitHub Repo and Start Over. Also, make a new copy of the Google Sheet template, give it a new name, and copy data from your old sheet using File > Paste Special > Values Only.

Problems with Chart.js code templates

Chart displays old data If you upload new data to your Chart.js code template on GitHub Pages, and it does not appear in your browser after refreshing and waiting up to one minute, then GitHub Pages is probably not the cause of the problem. Instead, some browsers continue to show “old” Chart.js in the web cache. The simplest solution is to File > Quit your browser and re-open the link to your Chart.js

TODO: Our Chart.js templates appear blank (just text, no chart) when viewed in the local browser. But Leaflet maps appear mostly or partially complete. Why is this, and how should we inform readers about this? Discuss with Ilya

Solve Problems with Browser Developer Tools

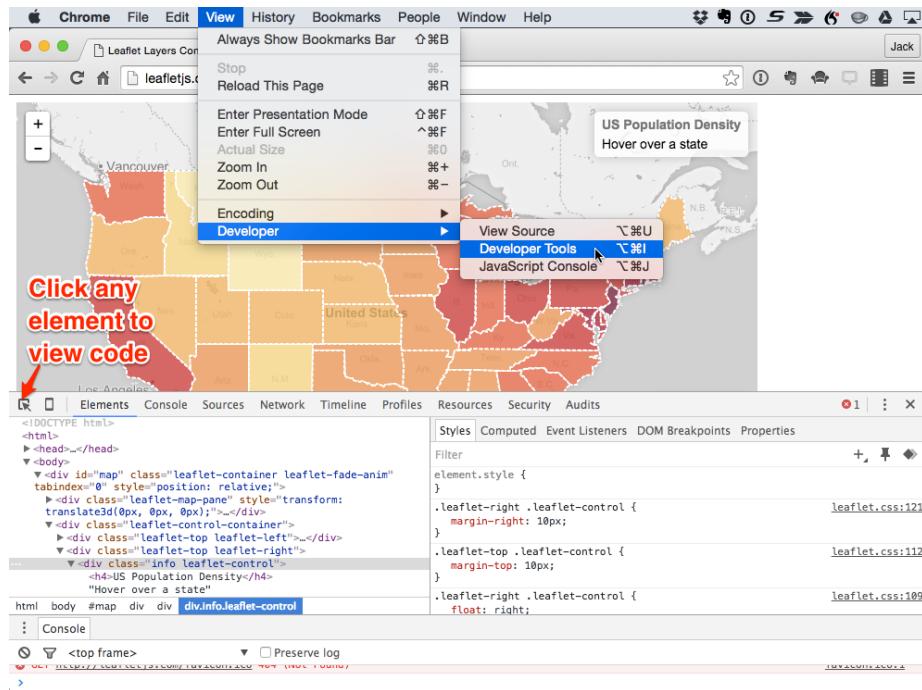
Peek inside any website and view the web code under the hood with the browser developer tools.

In Chrome for Mac, go to View > Developer > Developer Tools

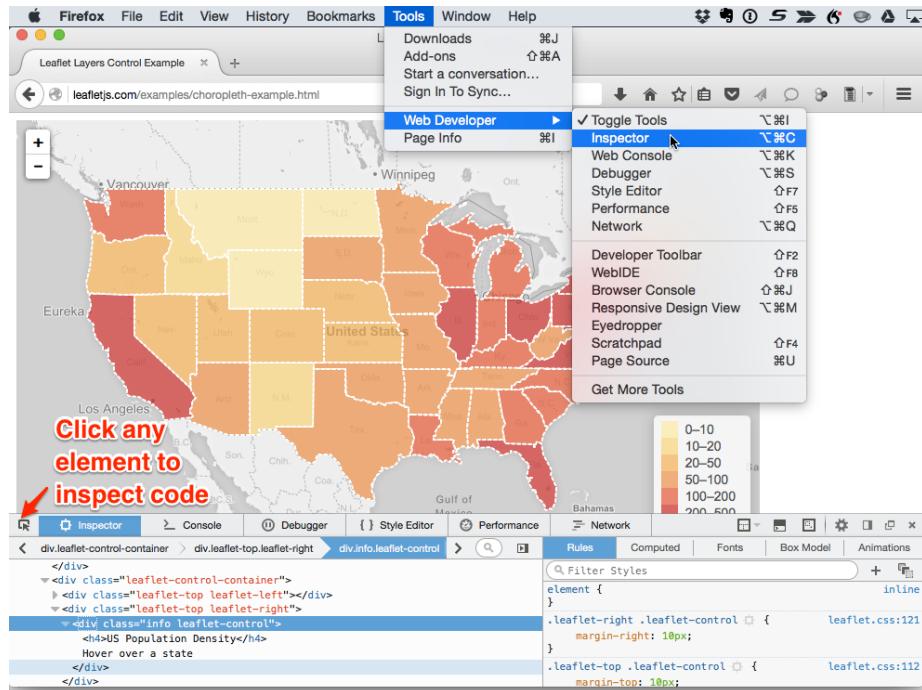
The screenshot shows a Microsoft Excel spreadsheet with the title "Leaflet Maps with Google Sheets" in the top bar. The menu bar includes File, Edit, View, Insert, Format, and a ribbon with icons for print, refresh, and search. Below the ribbon is a toolbar with icons for file, print, and other functions. The formula bar shows "fx". The main content is a table with 12 rows and 3 columns. The first column contains row numbers from 1 to 12. The second column contains labels for settings, and the third column contains their values. A large red watermark in the center of the table reads "Do NOT rename or delete these labels".

	A	
1	Setting	
2	Map Info	Do NOT rename or delete these labels
3	Map Title	Demo
4	Map Subtitle	trans
5	Display Title	on
6	Author Name	Jack
7	Author Email or	jack.
8	Author Code Credit	<a href="
9	Author Code Repo	https://
10	Map Settings	
11	Basemap Tiles	Carto
12	Cluster Markers	off

Figure A.6: Screenshot: Do not rename or delete



In Firefox for Mac, go to Tools > Web Developer > Inspector



- Bergstrom, Carl T., and Jevin D. West. *Calling Bullshit: The Art of Skepticism in a Data-Driven World*. Random House, 2020. https://www.google.com/books/edition/Calling_Bullshit/Plu9DwAAQBAJ.
- Best, Joel. *More Damned Lies and Statistics: How Numbers Confuse Public Issues*. Berkeley, CA: University of California Press, 2004. https://www.google.com/books/edition/More_Damned_Lies_and_Statistics/SWBr7D6VavoC.
- Brewer, Cynthia A. *Designing Better Maps: A Guide for GIS Users*. Esri Press, 2016. https://www.google.com/books/edition/Designing_Better_Maps/gFErrgEACAAJ.
- Cairo, Alberto. *How Charts Lie: Getting Smarter About Visual Information*. W. W. Norton & Company, 2019. https://www.google.com/books/edition/How_Charts_Lie_Getting_Smarter_about_Vis/qP2KDwAAQBAJ.
- . *The Truthful Art: Data, Charts, and Maps for Communication*. Pearson Education, 2016. https://www.google.com/books/edition/The_Truthful_Art/8dKKCwAAQBAJ.
- D'Ignazio, Catherine, and Lauren F. Klein. *Data Feminism*. MIT Press, 2020. <https://data-feminism.mitpress.mit.edu/>.
- Dougherty, Jack, Jeffrey Harrelson, Laura Maloney, Drew Murphy, Russell Smith, Michael Snow, and Diane Zannoni. "School Choice in Suburbia: Test Scores, Race, and Housing Markets." *American Journal of Education* 115, no. 4 (August 2009): 523–48. http://digitalrepository.trincoll.edu/cssp_papers/1.
- "Drilling into the DEA's Pain Pill Database." *Washington Post*, July 16, 2019. <https://www.washingtonpost.com/graphics/2019/investigations/dea-pain-pill-database/>.
- Few, Stephen. *Show Me the Numbers: Designing Tables and Graphs to Enlighten*. Second edition. Burlingame, CA: Analytics Press, 2012.
- Fowler, Geoffrey A. "Alexa Has Been Eavesdropping on You This Whole Time." *Washington Post*, May 6, 2019. <https://www.washingtonpost.com/technology/2019/05/06/alexas-has-been-eavesdropping-you-this-whole-time/>.
- . "Facebook Will Now Show You Exactly How It Stalks You — Even When You're Not Using Facebook." *Washington Post*, January 28, 2020. <https://www.washingtonpost.com/technology/2020/01/28/off-facebook-activity-page/>.
- . "Goodbye, Chrome: Google's Web Browser Has Become Spy Software." *Washington Post*, June 21, 2019. <https://www.washingtonpost.com/technology/2019/06/21/google-chrome-has-become-surveillance-software-its-time-switch/>.

- Huff, Darrell. *How to Lie with Statistics*. W. W. Norton & Company, 1954. <http://books.google.com/books?isbn=0393070875>.
- Knaflic, Cole Nussbaumer. *Storytelling with Data: A Data Visualization Guide for Business Professionals*. 1 edition. Hoboken, New Jersey: Wiley, 2015.
- . *Storytelling with Data: Let's Practice!* John Wiley & Sons, 2019. https://www.google.com/books/edition/Storytelling_with_Data/aGatDwAAQBAJ.
- Menasce Horowitz, Julia, Ruth Igielnik, and Rakesh Kochhar. “Trends in U.S. Income and Wealth Inequality.” Pew Research Center’s Social & Demographic Trends Project, January 9, 2020. <https://www.pewsocialtrends.org/2020/01/09/trends-in-income-and-wealth-inequality/>.
- Monmonier, Mark. *How to Lie with Maps, Third Edition*. University of Chicago Press, 2018. https://www.google.com/books/edition/How_to_Lie_with_Maps_Third_Edition/MwdRDwAAQBAJ.
- Mullen, Lincoln. “How to Make Prudent Choices About Your Tools.” ProfHacker, August 14, 2013. <https://lincolnmullen.com/blog/how-to-make-prudent-choices-about-your-tools/>.
- NASA JPL. “Educator Guide: Graphing Global Temperature Trends,” 2017. <https://www.jpl.nasa.gov/edu/teach/activity/graphing-global-temperature-trends/>.
- Newman, Andrew Adam. “Selling Gum with Health Claims.” *The New York Times: Business*, July 27, 2009. <https://www.nytimes.com/2009/07/28/business/media/28adco.html>.
- Rost, Lisa Charlotte. “How to Pick More Beautiful Colors for Your Data Visualizations.” Chartable. Accessed October 21, 2020. <https://blog.datawrapper.de/beautifulcolors/index.html>.
- . “How to Prepare Your Data for Analysis and Charting in Excel & Google Sheets.” Chartable: A Blog by Datawrapper. Accessed August 28, 2020. <https://blog.datawrapper.de/prepare-and-clean-up-data-for-data-visualization/index.html>.
- . “What I Learned Recreating One Chart Using 24 Tools.” Source, December 8, 2016. <https://source.opennews.org/en-US/articles/what-i-learned-recreating-one-chart-using-24-tools/>.
- . “Your Friendly Guide to Colors in Data Visualisation.” Chartable: A Blog by Datawrapper, July 31, 2018. <https://blog.datawrapper.de/colorguide/>.
- Schwabish, Jon. “Thread Summarizing ‘Ten Guidelines for Better Tables.’” Twitter, August 3, 2020. <https://twitter.com/jschwabish/status/1290323581881266177>.

- Schwabish, Jonathan. *Better Data Visualizations: A Guide for Scholars, Researchers, and Wonks.* Columbia University Press, 2021. <https://cup.columbia.edu/book/better-data-visualizations/9780231193115>.
- Schwabish, Jonathan A. "Ten Guidelines for Better Tables." *Journal of Benefit-Cost Analysis* 11, no. 2: 151–78. Accessed August 25, 2020. <https://doi.org/10.1017/bca.2020.11>.
- Schwabish, Jonathan, and Alice Feng. "Applying Racial Equity Awareness in Data Visualization." Preprint. Open Science Framework, August 27, 2020. <https://doi.org/10.31219/osf.io/x8tbw>.
- . "Applying Racial Equity Awareness in Data Visualization." Medium. Accessed October 16, 2020. https://medium.com/@urban_institute/applying-racial-equity-awareness-in-data-visualization-bd359bf7a7ff.
- Spiegelhalter, David. *The Art of Statistics: How to Learn from Data.* Basic Books, 2019. https://www.google.com/books/edition/The_Art_of_Statistics/04-FDwAAQBAJ.
- . *The Art of Statistics: Learning from Data.* Penguin UK, 2019. https://www.google.com/books/edition/The_Art_of_Statistics/CiZeDwAAQBAJ.
- Stone, Chad, Danilo Trisi, Arloc Sherman, and Jennifer Beltrán. "A Guide to Statistics on Historical Trends in Income Inequality." Center on Budget and Policy Priorities, January 13, 2020. <https://www.cbpp.org/research/poverty-and-inequality/a-guide-to-statistics-on-historical-trends-in-income-inequality>.
- Tufte, Edward R. *Envisioning Information.* Cheshire, CT: Graphics Press, 1990. https://www.google.com/books/edition/Envisioning_Information/_EZiAAAAMAAJ.
- Tufte, Edward R. *Beautiful Evidence.* Graphics Press, 2006. <http://books.google.com/books?isbn=0961392177>.
- Urban Institute. "Urban Institute Data Visualization Style Guide," 2020. <http://urbaninstitute.github.io/graphics-styleguide/>.
- Watters, Audrey. "'The Audrey Test': Or, What Should Every Techie Know About Education?" Hack Education, March 17, 2012. <http://hackeducation.com/2012/03/17/what-every-techie-should-know-about-education>.
- "What's Real About Wages?" Federal Reserve Bank of St. Louis. The FRED Blog, February 8, 2018. <https://fredblog.stlouisfed.org/2018/02/are-wages-increasing-or-decreasing/>.
- Wheelan, Charles. *Naked Statistics: Stripping the Dread from the Data.* W. W. Norton & Company, 2013. https://www.google.com/books/edition/Naked_Statistics_Stripping_the_Dread_fro/j5qYPqsBJb0C.

- World Inequality Database. “Income Inequality, USA, 1913-2019,” 2020. https://wid.world/share/#0/countrytimeseries/aptinc_p50p90_z;aptinc_p90p100_z;aptinc_p0p50_z/US/2015/kk/k/x/yearly/a/false/0/400000/curve/false.
- . “Methodology.” WID - World Inequality Database, 2020. <https://wid.world/methodology/>.
- . “Top 1% National Income Share,” 2020. https://wid.world/world/#sptinc_p99p100_z/US;FR;DE;CN;ZA;GB;WO/last/eu/k/p/yearly/s/false/5.070499999999999/30/curve/false/country.
- Yau, Nathan. “How to Spot Visualization Lies.” FlowingData, February 9, 2017. <http://flowingdata.com/2017/02/09/how-to-spot-visualization-lies/>.
- Zuboff, Shoshana. *The Age of Surveillance Capitalism: The Fight for a Human Future at the New Frontier of Power*. PublicAffairs, 2019. https://www.google.com/books/edition/The_Age_of_Surveillance_Capitalism/lRqrDQAAQBAJ.