

Hands-On Data Visualization

Interactive Storytelling from Spreadsheets to Code

Jack Dougherty Ilya Ilyankou

2020-09-24

Contents

Preface	7
Authors & Acknowledgements	7
Audience and Outline	10
Follow Along on a Computer	10
1 Introduction	13
2 Choose Tools to Tell Your Story	21
Start Sketching Your Data Story	21
Tools and Tradeoffs	24
Recommended Tools	29
Use a Password Manager	30
3 Strengthen Your Spreadsheet Skills	31
Select your Spreadsheet Tools	32
Download to CSV or ODS Format	35
Make a Copy of a Google Sheet	36
Share Your Google Sheets	38
Upload and Convert to Google Sheets	40
Geocode Addresses with Google Sheets Add-On	43
Collect Data with Google Forms	46
Sort and Filter Data	49
Calculate with Formulas	51
Summarize Data with Pivot Tables	54

Match Columns with VLOOKUP	57
Connect Sheets with a Relational Database	61
4 Find and Question Your Data	65
Guiding Questions for Your Search	66
Public and Private Data	68
Mask or Aggregate Sensitive Data	72
Open Data Repositories	73
Source Your Data	75
Recognize Bad Data	77
Question Your Data	78
5 Clean Up Messy Data	81
Find and Replace with Blank	82
Transpose Rows and Columns	83
Split Data into Separate Columns	86
Combine Data into One Column	89
Extract Tables from PDFs with Tabula	92
Clean Data with OpenRefine	95
6 Chart Your Data	101
Chart Design Principles	105
Google Sheets Charts	112
- Bar and Column Charts	114
- Pie, Line, and Area Charts	126
Datawrapper Charts	131
- Annotated Charts	132
- Range Charts	136
- Scatter and Bubble Charts	140
Tableau Public Charts	146
- Scatter Chart	148
- Filtered Line Chart	153

CONTENTS	5
7 Map Your Data	159
Map Design Principles	161
Point Map with Google My Maps	166
Choropleth Map with Datawrapper	173
Filtered Point Map with Socrata	178
Polygon Map with Tableau Public	185
8 Table Your Data	191
Table Design Principles	192
Datawrapper Table with Sparklines	192
9 Embed On Your Web	201
Create a Simple Web Page with GitHub Pages	202
Copy an iframe code from a Google Sheets interactive chart	204
Convert a Weblink into an Iframe	207
Embed an Iframe in GitHub Pages	208
Embed an Iframe on WordPress.org	210
Embed Tableau Public on your Website	212
10 Edit and Host Code with GitHub	217
Copy, Edit, and Host a Simple Leaflet Map Template	219
Create a New Repo and Upload Files on GitHub	228
GitHub Desktop and Atom Editor to Code Efficiently	231
11 Chart.js and Highcharts Templates	243
Bar or Column Chart with Chart.js	246
Error Bars with Chart.js	248
Line Chart with Chart.js	250
Annotated Line Chart with Highcharts	252
Scatter Chart with Chart.js	254
Bubble Chart with Chart.js	255

12 Leaflet Map Templates	259
Leaflet Maps with Google Sheets	261
Leaflet Storymaps with Google Sheets	273
Get Your Google Sheets API Key	280
Leaflet Maps with CSV Data	287
Leaflet Maps with Open Data API	292
Leaflet Heatmap	299
Searchable Map	299
13 Transform Your Map Data	301
Bulk Geocode with US Census	302
Pivot Points into Polygon Data	303
Normalize Data for Meaningful Maps	304
Convert to GeoJSON format	308
Draw and Edit with GeoJson.io	311
Edit and Join with Mapshaper	316
Georectify with MapWarper	331
Convert Compressed KMZ to KML	331
14 Detect Bias in Data Stories	335
How to Lie with Charts	336
How to Lie with Maps	337
15 Tell Your Data Story	339
A Fix Common Mistakes	341

Preface

This **BOOK-IN-PROGRESS** was last updated on: **24 Sep 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

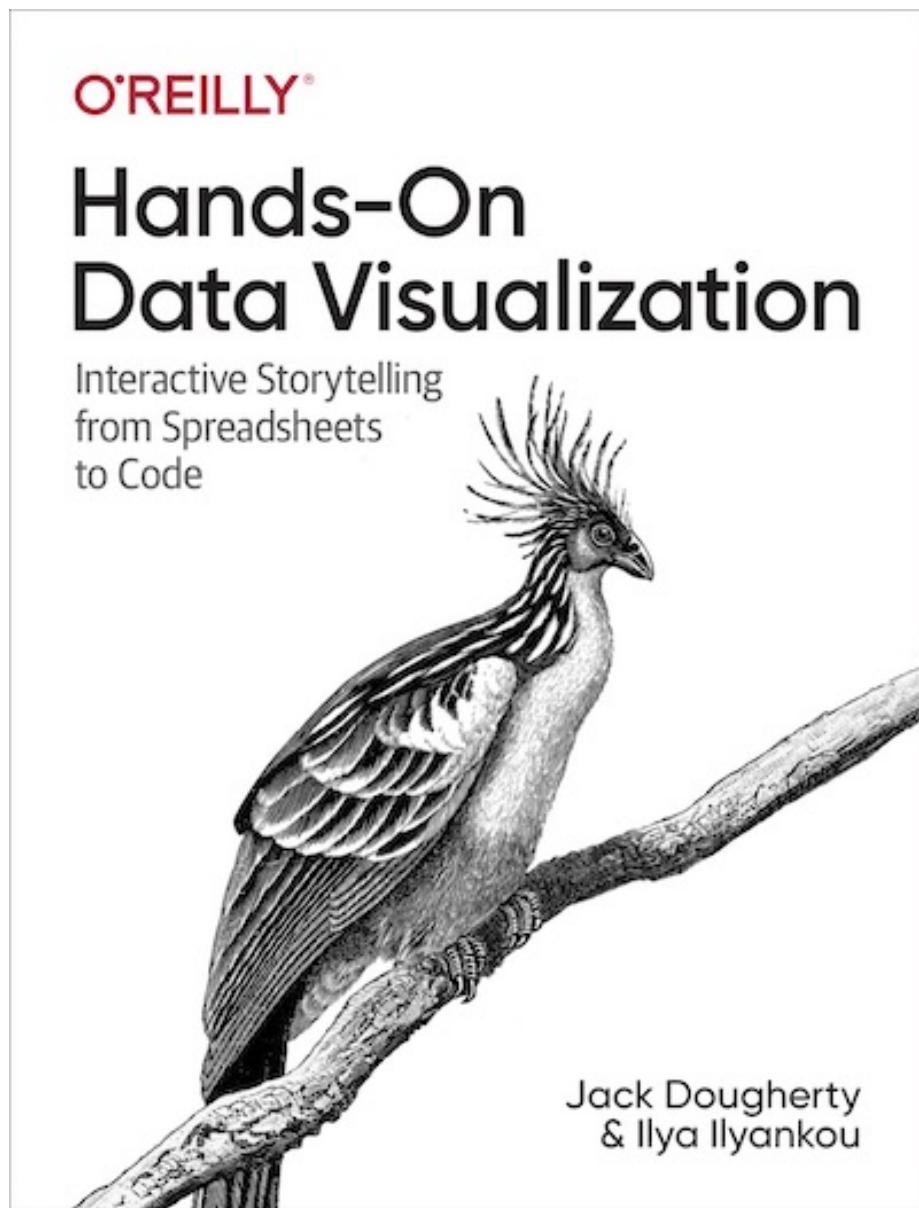


Figure 1: Book cover: Read about the hoatzin “reptile bird”

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

In this book, we will learn how to create data visualizations through a series of design principles and step-by-step tutorials, in order to make your information-based analysis and writing more insightful and compelling. Just as sentences become more persuasive with supporting evidence and source notes, your data-driven arguments can become more powerful when accompanied by appropriate tables, charts, or maps. While words tell us stories, visualizations show us *data stories* by transforming quantitative, relational, or spatial patterns into images. Data visualizations work best when they draw our attention to highly meaningful patterns that would be difficult to communicate through text alone, yet jump out at our eyes in pictures.

Our book features a growing number of free and easy-to-learn digital tools for creating data visualizations. Unlike infographics, which are typically created as one-time artwork, visualizations can be reused and modified multiple times by updating the underlying data. We also focus on creating interactive data visualizations that you can embed on your website. While static visualizations can be distributed on paper or PDF documents, interactive visualizations will engage wider audiences on the internet by encouraging them to interact with the data, explore patterns that interest them, and download files if desired.

But the remarkable growth of data visualizations and the internet in recent decades also poses a serious problem. Today we encounter more words, numbers, and images than at any prior point in our lives, but much of it is not trustworthy. The “information age” has devolved into an “age of disinformation.” When presented with conflicting stories, who do you believe? How do you make wise decisions about what information to trust? In particular, for this book about data visualization, what types of evidence persuade you, and why?

For example, how can you tell whether or not we—the authors of this book—are lying to you?

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.¹

Is this second claim more believable than the first one? It now makes a more descriptive 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 wording of the claim make you wonder about the 40 percent of the population in between the two extremes? Or, what about the authority of the source? Who is responsible for creating the World Inequality Database, and do you trust them?

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

Year	Top 10 Percent	Middle 40 Percent	Bottom 50 Percent
1970	\$136,308	\$44,353	\$16,515
2019	\$352,815	\$76,462	\$19,117

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,

¹World 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.

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.²

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.

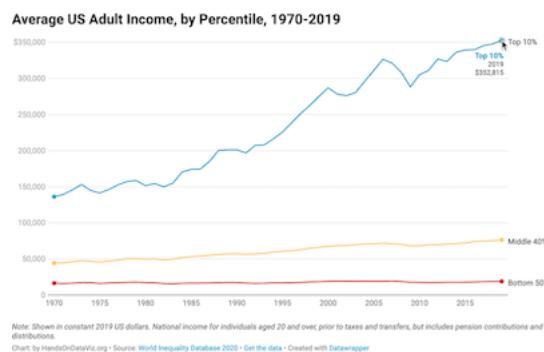


Figure 1.1: Explore the interactive line chart of US adult income inequality over time.

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

²The 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 submitted 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 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/>

difference. But the line chart also communicates a powerful, visualized story about income gaps that grabs our 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, far distant from the other lines. Meanwhile, as the middle-tier income line rose slightly over time, the lowest-tier slightly declined, meaning that the bottom half of the population has less purchasing power in 2019 than they did at their peak in the early 2000s. The rich got richer, and the poor got poorer, as the saying goes. But the chart reveals how rapidly those riches grew, and how poverty became even worse, over time.

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 liars.

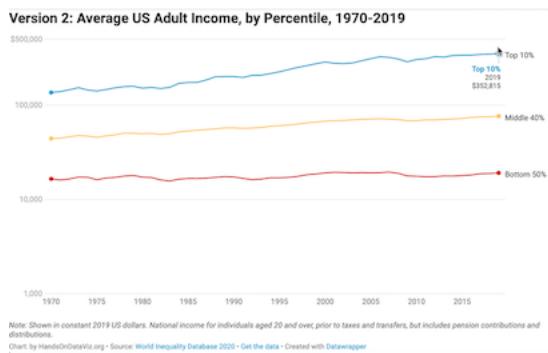


Figure 1.2: Explore the alternative version of the interactive line chart of US adult income inequality over time, using the same data as the first version.

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 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 early months of 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 disguise the crisis. We have the ability to illuminate the truth with charts, but we can also use charts to lie.

Let's expand our analysis of income inequality beyond national borders. 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 received by the top 1 percent. In other words, how large a slice of the pie goes to 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.³

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 data for all of the nations featured in our claim, and we also want to encourage readers to explore information around the globe. Since this is spatial data, let's transform it into a map, as shown in Figure 1.3.

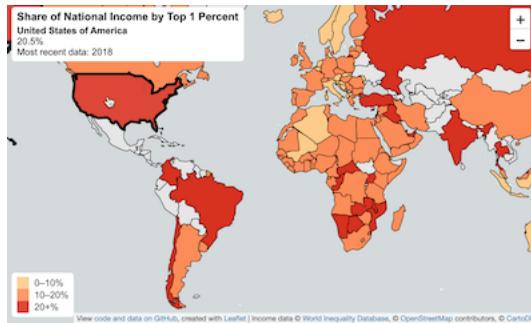


Figure 1.3: Explore the interactive map of world income inequality, measured by the share of national income received by the top 1 percent of the population, based on the most recent data available. Source: World Inequality Database 2020.

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

³World 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.

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. The US and some other nations stand out in dark red, at the highest level of the legend, where the top 1 percent of the population receives 20 percent or more of the national income. By contrast, as your eye floats across the Atlantic, nearly all of the European nations appear in lighter beige and orange colors, showing that their top-tier receive a smaller share of the national income, typically between 5 and 15 percent. Income inequality burns red in the US, while Europe appears calm.

Now let's introduce the 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?

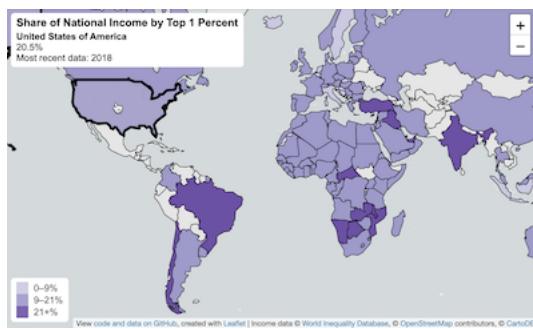


Figure 1.4: Explore the alternative version of the interactive map of world income inequality, using the same data as the map above.

Why does the second map in Figure 1.4 look different than the first map in Figure 1.3? The US is now shaded in a lighter purple color, similar to that of Europe. Did the inequality crisis simply fade away from the US? Which map tells the truth?

This time, neither map is misleading. Both make valid 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 legend in the bottom-left corner of both maps. The first map sorts nations in three ranges (0-10, 10-20, 20+), while the second map uses slightly different range cutoffs (0-9, 9-21, 21+). Since the US share is 20.5 percent, in the first map it falls into the top bucket with the darkest color, but in the second map it falls into the middle bucket with a lighter color. Yet both ranges are equally valid, as there is no clear rule about using one versus the other in this case. In the same way, the first map displays three shades of red, while the second map uses purple. Based on design principles discussed later in the book, both are equally valid color choices.

To summarize, people can lie with charts and maps, and some design choices

can intentionally mislead. But data visualization is also interpretive and open-ended. It's common to create more than one picture of the truth.

That poses a challenge for us as authors of this book. Data visualizations do not always represent the truth. Like words, they can be manipulated to mislead your audience. To help you create true and meaningful charts and maps, we'll point you toward principles of good design, encourage thoughtful habits of mind, and in some cases specifically tell you what *not* to do. But data visualization can be a slippery subject, sometimes more art than science. Often there is no *single* correct answer to a visualization problem, but rather *several* plausible ones, each with their own strengths and weaknesses. As a reader of this book who's learning about data visualization, your job is to search for *good answers* without necessarily expecting to find *the one right answer*, especially as visualization tools and methods continue to evolve. Also, keep your guard up and continue to watch out for data visualizations that lie.

Summary

TODO: do we need this here?

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 and powerful ones.

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. Have fun!

TODO: decide whether to add amateur-styled sketches of problem, question, data, and visualization. We could place one image immediately after each step, or conclude with a smaller 4-panel compilation (2x2 landscape oriented page scans) after all four steps.

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 2.1. 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.



Figure 2.1: 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.

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 15: Telling 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 about building a chart or map, but need examples of different types, look at the beginning of Chapter 6: Chart Your Data and Chapter 7: 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.

Tools and Tradeoffs

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. Of course, 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 removed a freemium tool from an earlier version of this book because 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? Consider 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. 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 when selecting tools for this book, we highlight those such as Datawrapper, which includes a built-in check for colorblindness, and Highcharts, which offers some chart types designed for low-vision people using screen readers, as shown in Figure 2.2.

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

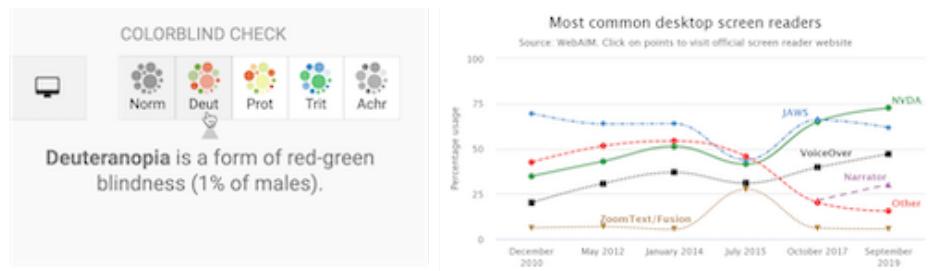


Figure 2.2: On the left, the Datawrapper built-in check for colorblindness. On the right, a Highcharts line chart designed for low-vision accessibility.

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.

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'll list all of the tools recommended in the pages of this book, with a brief description and a reference to the chapter that discusses each in more detail.

Although we tried to minimize the size of our digital toolkit, we were surprised to discover that the final roster includes ZZZ [TODO INSERT NUMBER] tools! We agree that this number may still seem overwhelming at first to beginners. Let's be clear: 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.

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 of 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>

TODO: Insert Table: Tools featured in this book, with links to relevant chapters, and how they rate on various criteria above

You may have other tool recommendations...and as we stated, tools change and evolve over time. So if you'd like us to consider another tool that's not already on the list, email or tweet us... explain what your recommendation would replace in the toolkit, and how it rates better on our criteria above.

Use a Password Manager

TODO: If you don't already have one, start using a password manager. One tool to rule them all! ...add password manager tutorial to keep track of your accounts for the online tools you'll use in this book. The free and open-source BitWarden.com tool nicely integrates with most browsers and even smartphones.

Summary

TODO Now you have a better sense of different types of data visualization tools, and how to make wise decisions when choosing among them... The next chapter is designed to strengthen your skills for using the most common tool in our data 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 the following data:

	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					

Below the table, there are tabs for 'data' and 'notes', and a section labeled 'Tabs for multiple sheets'.

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 Suite business account subscription, 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, privacy 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.

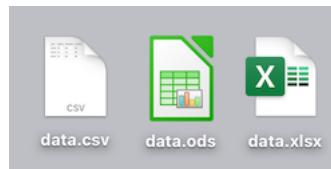


Figure 3.2: Three data formats commonly seen on your computer—`csv`, `ods`, and `xlsx`—when displayed properly in the Mac Finder.

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.

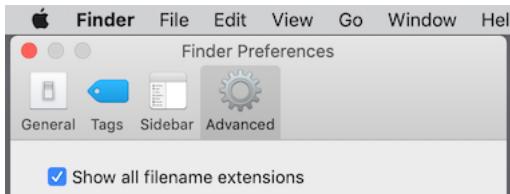


Figure 3.3: On a Mac, go to *Finder-Preferences-Advanced* and check the box to *Show all filename extensions*.

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 tools in later chapters of this book. Only the downloadable Excel tool (which

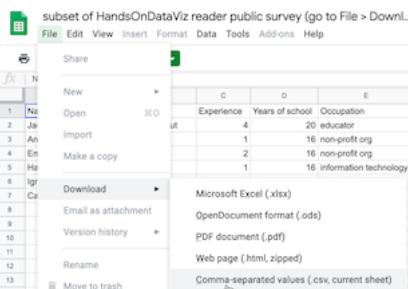


Figure 3.4: In Google Sheets, go to *File - Download As* to export data in several common formats.

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 ...”.

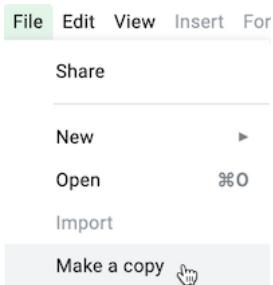


Figure 3.5: Go to *File - Make a Copy* to create your own version of this Google Sheet.

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.

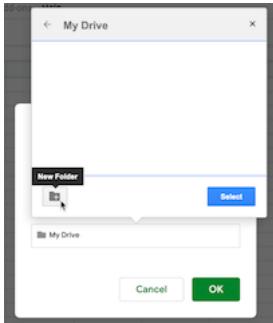


Figure 3.6: Click the *My Drive* and *New folder* buttons to save your work in a folder.

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.

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.

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

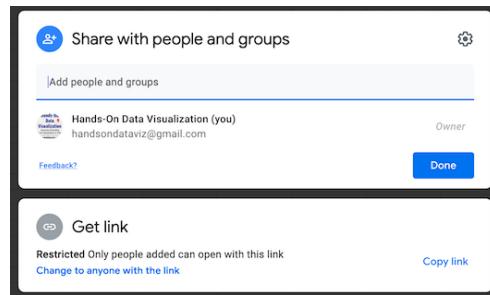


Figure 3.7: Click the *Share* button to grant access to individuals (top half) or anyone with the link (bottom half).

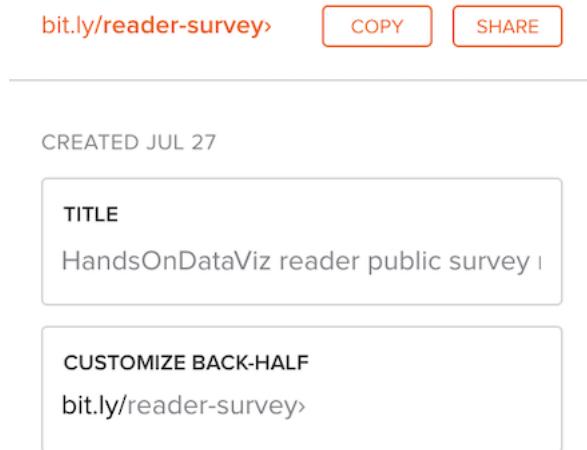


Figure 3.8: Use a free link-shortening service, such as Bitly.com, and customize its back-end.

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.
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

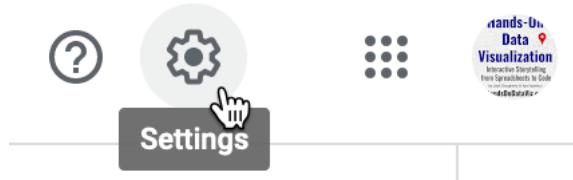


Figure 3.9: Click your Google Drive *Gear Symbol - Settings* in the upper-right corner.

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.

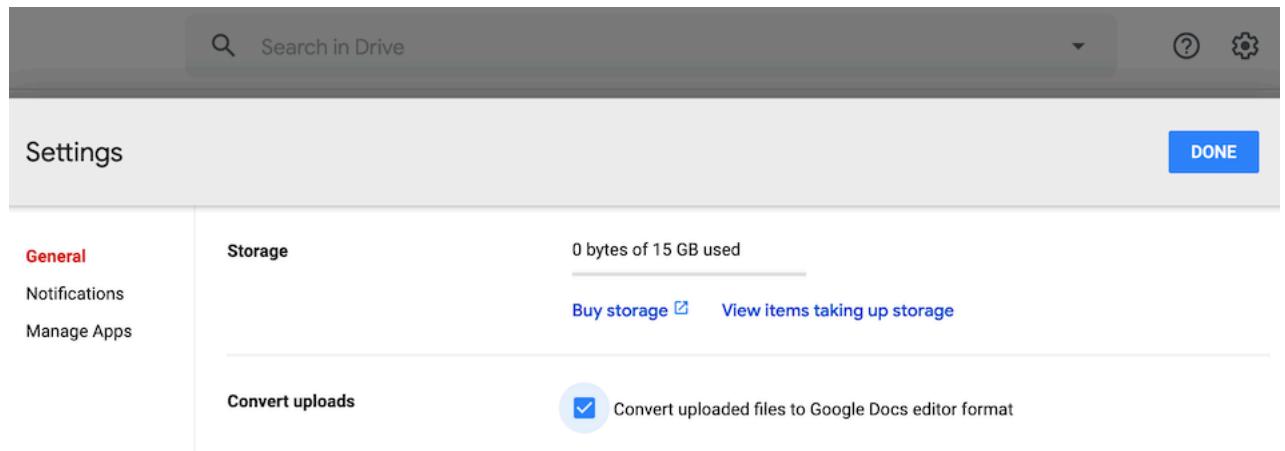


Figure 3.10: Inside your Google Drive Settings, check the box to automatically convert all uploads.

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.

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

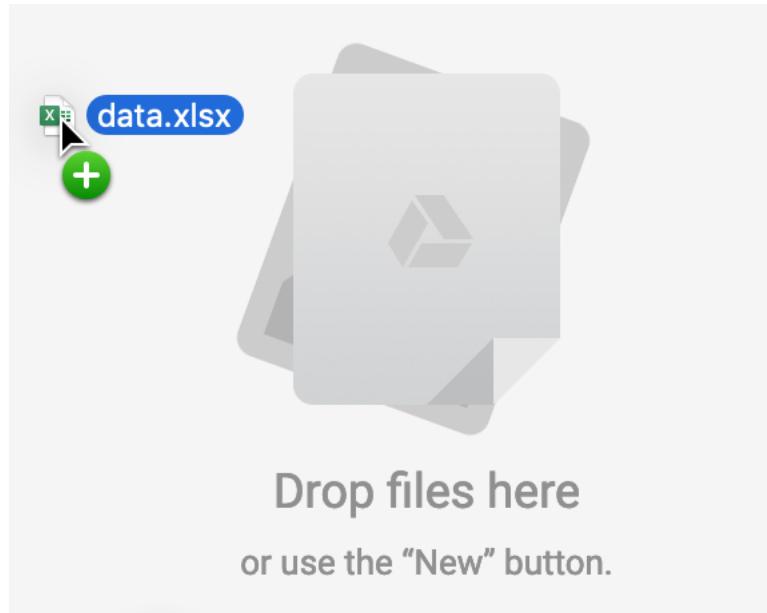


Figure 3.11: Drag-and-drop your sample Excel file into your Google Drive to upload it.

-
- + **Google Sheets file**

 - X **Microsoft Excel file.xlsx**

 - = **Comma Separated Values file.csv**

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

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 with Google Sheets Add-On

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 12.

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.

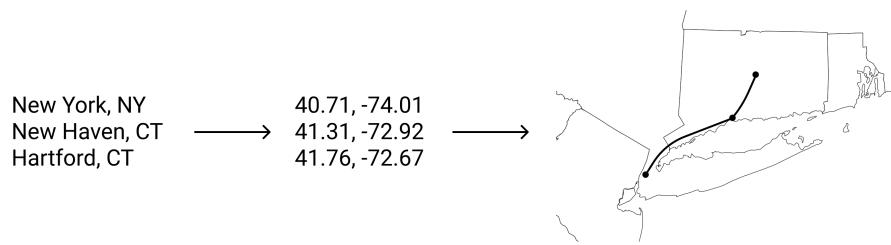


Figure 3.13: To map addresses, you first need to geocode them.

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.

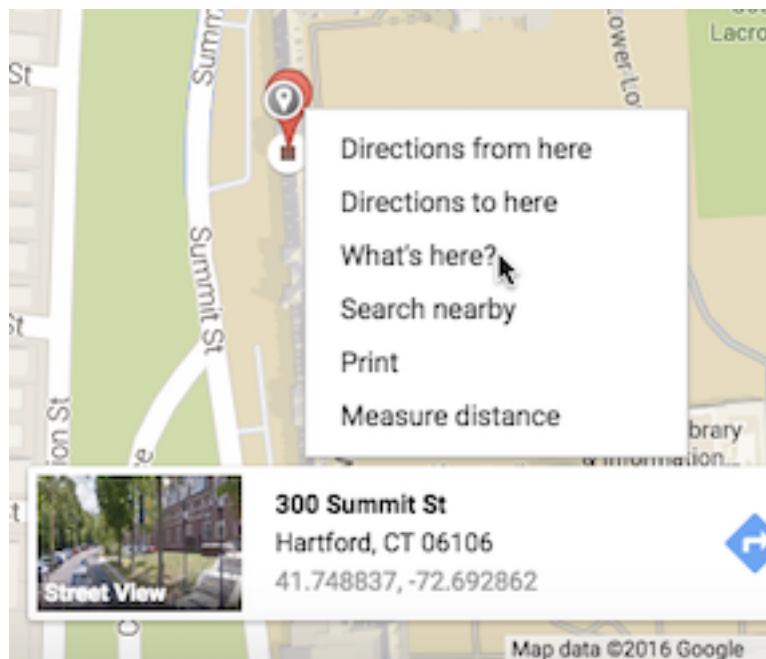


Figure 3.14: To geocode one address, search in Google Maps and right-click *What's here?* to show coordinates.

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 *Create Template*. The Add-on will create a new tab, named *Geocoding*, and automatically insert three sample addresses, as shown in Figure 3.15.

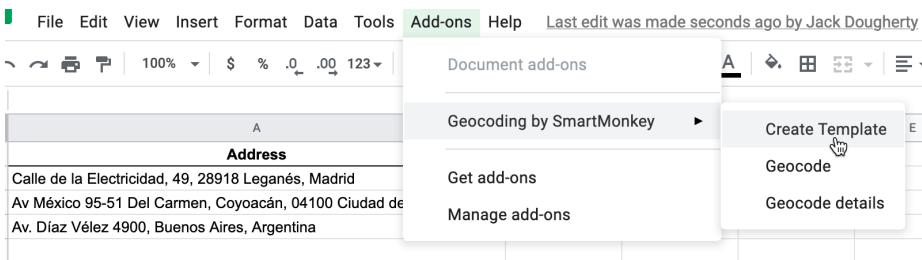


Figure 3.15: In the Google Sheets *Add-On* menu, select *Geocoding by SmartMonkey – Create Template*.

3. Select the *Geocoding by SmartMonkey > Geocode Details* menu. The Add-on will create another spreadsheet tab, called *Geocoding Details*, and display the results from Google services for three new columns—latitude, longitude, and address found—as shown in Figure 3.16. Always review the quality of geocoded results by comparing the *Address found* column to the original *Address* entered.

When you paste your own address data into the spreadsheet to geocode it, follow these guidelines:

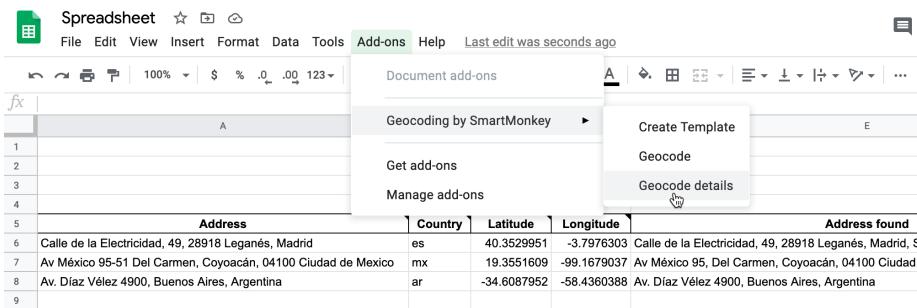


Figure 3.16: Select *Geocode Details* to view latitude, longitude, and address found for each entry.

- Each cell requires a full address. 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.
- The address should follow the format of the national postal service of the country where it is located. Separate terms with spaces.
- Do not skip any rows in the *Address* column.
- 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.
- 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 13: 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 3.17.

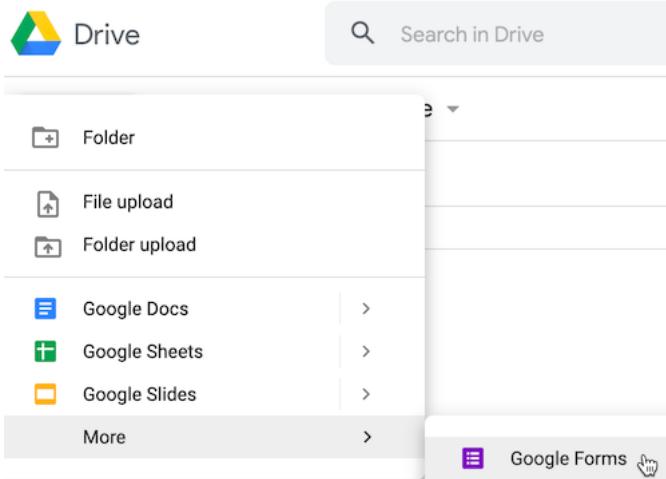


Figure 3.17: The Google Forms tool is partially hidden in the Google Drive *New - More* menu.

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 3.18. 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 3.19. 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 3.20. 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 9: Embed On Your Web.

The Google Forms *Responses* tab will show individual results you receive, and also includes a powerful button to open the data in a linked Google Sheet, as shown in Figure 3.21.

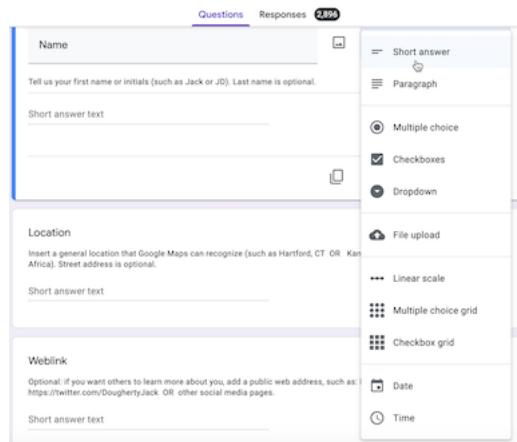


Figure 3.18: The Google Forms *Questions* tab allows you to designate different types of responses.

Figure 3.19: Click the three-dot kebab menu to *Show - Description* to add details for any question.

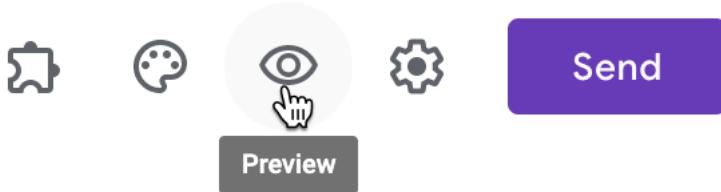


Figure 3.20: Click the *Eyeball symbol* to preview your form.

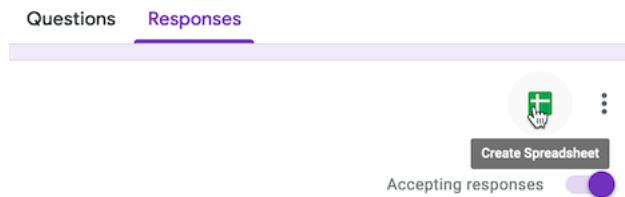


Figure 3.21: The Google Forms *Responses* tab includes a button to open results in a linked Google Sheet.

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 3.22. 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.

	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	Newham, IL
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 3.22: Click the upper-left corner to select all cells before sorting.

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!

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 3.23, to display the minimum at the top, the maximum at the bottom, and the range in between.

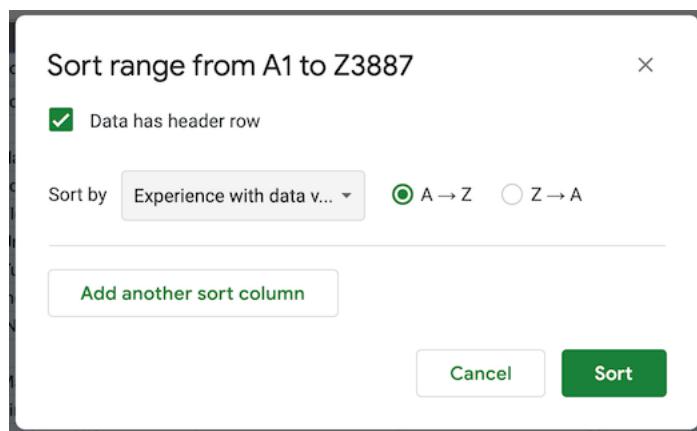


Figure 3.23: Go to *Data - Sort Range*, check the header row box, and sort by *Experience with dataviz* in ascending order.

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 3.24. 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*.

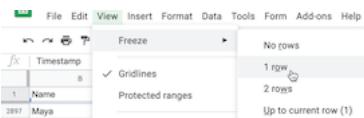


Figure 3.24: In Google Sheets, go to *View - Freeze* to select the number of rows to continuously display when scrolling downward.

- 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 3.25. Click “OK”.

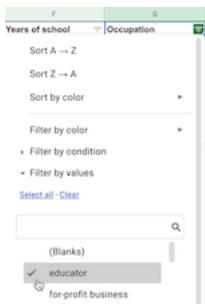


Figure 3.25: Go to *Data - Create a Filter*, click the downward arrow in the *Occupation* column, select only *educator*.

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 3.26.

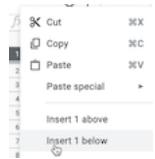


Figure 3.26: Right-click on row number 1 and select *Insert 1 below*.

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 3.27.

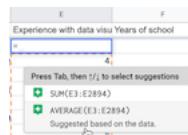


Figure 3.27: Type `=` to start a formula and select the suggestion for average, or type it directly in with the correct range.

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 3.28. 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.

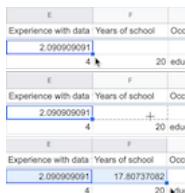


Figure 3.28: Click on the blue bottom-right dot in cell E2, then hold-and-drag your crosshair cursor in cell F2, and let go to automatically paste and update the formula.

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 3.29.

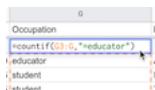


Figure 3.29: Select or enter a formula that counts responses if the entry is *educator*.

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 13: 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 3.30. The new sheet will include a Pivot Table tab at the bottom.

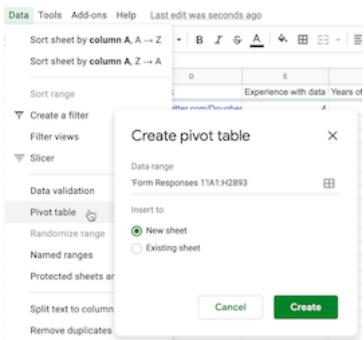


Figure 3.30: Go to *Data - Pivot Table*, and create in a new sheet.

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 3.31.

Figure 3.31: 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 3.32.

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 3.33.

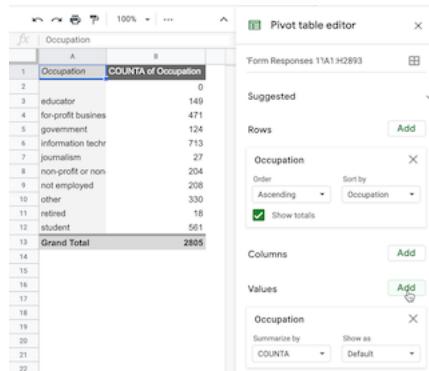


Figure 3.32: In the *Pivot table editor*, click the Values *Add* button and select *Occupation*.

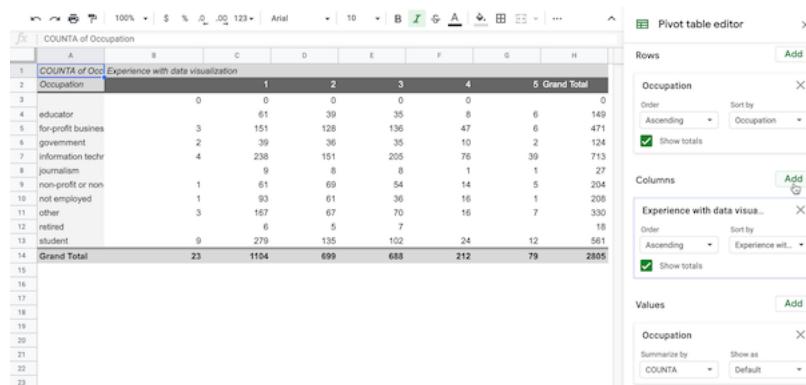


Figure 3.33: In the *Pivot table editor*, click the Columns *Add* button and select *Experience with data visualization*.

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 3.34. 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.

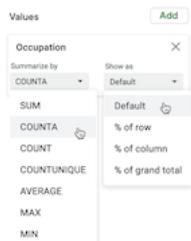


Figure 3.34: In the *Pivot table editor*, see multiple options to summarize *Values*.

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.

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 3.35 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.

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

Below the table are tabs for 'names' and 'addresses' with dropdown menus, and a 'source' button.

Figure 3.35: 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.

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.

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 3.36. This creates new column headers where our lookup results will be automatically pasted.

	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				

+ names addresses source

Figure 3.36: Paste the last four column headers from the *addresses* sheet into the *names* sheet.

- 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.
- 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 3.37. 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.

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.

The image shows two Google Sheets side-by-side. The top sheet is titled 'names' and has columns A through F. Column A contains names and organization names, column B contains organization names, and column C contains the formula =VLOOKUP(B2,addresses!A:E,2, false). The bottom sheet is titled 'addresses' and has columns A through E. It lists various food banks with their addresses, cities, states, and zip codes. The formula in the 'names' sheet is pointing to the 'addresses' sheet's data.

	A	B	C	D	E	F
1	name	organization	street	city	state	zip
2	Denise B.	Central Texas Food Bank	=VLOOKUP(B2,addresses!A:E,2, false)			
3	Demick 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 3.37: The VLOOKUP formula in cell C2 of the *names* sheet (top) searches for matches across columns A to E in the *addresses* sheet (bottom).

- 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 3.38.

The screenshot shows the 'names' sheet after the formula has been copied and pasted across columns D, E, and F. The formula =VLOOKUP(B2,addresses!A:E,2, false) is now in cells C2 through F2, and the values from the 'addresses' sheet are displayed in columns D, E, and F for the first row.

	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	Demick C.	Central Texas Food Bank				

Figure 3.38: 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 3.39.

The screenshot shows the 'names' sheet with the formula copied and pasted down to row 11. The formula =VLOOKUP(B2,addresses!A:E,2, false) is now in cells C2 through F11, and the values from the 'addresses' sheet are displayed in columns D, E, and F for all rows.

	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	Demick 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 City	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 City	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 City	UT	84119
12						

Figure 3.39: 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.

Connect Sheets with a 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 3.39 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 3.35 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 3.40. 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.

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

Figure 3.40: In this Airtable sample, we linked the *organization* column in the *people* sheet to the *food banks* sheet.

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 3.41. 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 9: Embed on your Web. See more about its features in the Airtable Support page.

The screenshot shows the Airtable interface with two sheets. The left sheet, titled 'people', has a grid view with 7 rows. The right sheet, titled 'Food Banks sample', shows the expanded details for the row where 'Denise B.' is listed in the 'name' column. The expanded view includes fields for 'NAME' (Denise B.) and 'ORGANIZATION' (Central Texas Food Bank), which then links to a detailed view of the organization's address: STREET (6500 Metropolis Dr), CITY (Austin), and STATE (TX).

	name	organization
1	Denise B.	Central Texas Food Bank
2	Derrick C.	Central Texas Food Bank
3	Eric S.	Arkansas Food Bank
4	Ginette B.	Utah Food Bank
5	Greg F.	Arkansas Food Bank
6	Kent L.	Utah Food Bank
7	Mark J.	Central Texas Food Bank

Figure 3.41: In this Airtable demo, click on a row in one sheet to expand and view its linked data in another sheet.

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 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 6 and interactive maps in Chapter 7, 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, we 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

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

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.

(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

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

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 Eric 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:

- 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,

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.

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/>

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 “FOIA,” 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 FOIA request, while other times they may delay or reject it, which may pressure the requester to attempt to resolve the 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, 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 13: 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 9: Embed on Your Web.
- **Application Program 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 12: 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 Public Data.
- Google Dataset Search.
- 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.

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.

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.

	A	B	C	D	E	F
1	DOHMH (Department of Health and Mental Hygiene) New York City Restaurant Inspection Results					
2	data only for January 2020					
3	Each row is a restaurant citation					
4	Source:					
5	https://nycopendata.socrata.com/Health/DOHMH-New-York-City-Restaurant-Inspection-Results/43nn-pn8j					
6	Downloaded by Jack Dougherty, July 2020					

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

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 06119. If someone converts a column of zip codes to numerical data, it will strip out the leading zero and appear as 6119. 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 represents the state of North Carolina.
- 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.
- 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.

Another way to review the quality of data entry in any column in a spreadsheet is to create a filter or a pivot table as described in Chapter 3. This allows you to quickly inspect the range of values that appear in that column, and whether they match what you expected to find.

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 to *question your data* by looking more deeply than what appears at its surface level. Read the source notes and 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 recorded?

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 6: 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 allow 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 6 and telling your data story in Chapter 15.

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.

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 **tOWN** 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.

	Geo_NAME	Pop2010
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

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 comes up when working with longitudinal or historical data. For example, you often find the data organized with years placed horizontally as column headers, as shown in Figure 5.3.

But if you wish to create a line chart, which you'll learn in Chapter 6: Chart Your Data, you need to transpose the data above, so that the years run down

	A	B	C	D	E	F	G
1	Year	1970	1980	1990	2000	2010	2017
2	Beef	79.6	72.1	63.9	64.5	56.7	54
3	Pork	48.1	52.1	46.4	47.8	44.3	47
4	Chicken	27.4	32.7	42.4	54.2	58	64
e							

Figure 5.3: We often find data with years placed in horizontal column headers.

the first vertical column, as shown in Figure 5.4.

	A	B	C	D
1	Year	Beef	Pork	Chicken
2	1970	79.6	46.1	27.4
3	1980	72.1	52.1	32.7
4	1990	63.9	46.4	42.4
5	2000	64.5	47.8	54.2
6	2010	56.7	44.3	58
7	2017	54	47	64

Figure 5.4: But we need to transpose the data to place years in the first vertical column.

Let's 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 5.5.

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.

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 5.6.
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).

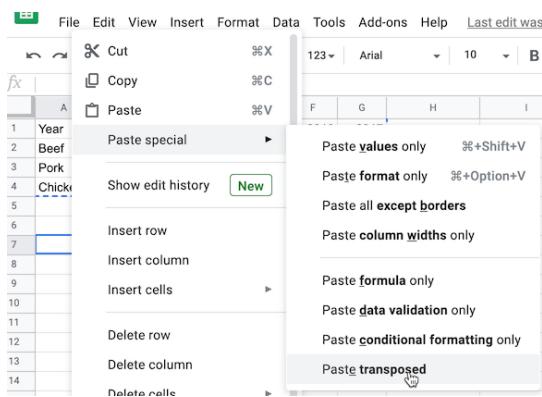


Figure 5.5: Go to *Edit - Paste Special - Paste Transposed* to swap rows and columns.

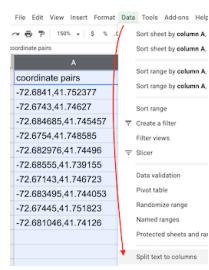


Figure 5.6: Select *Data - Split text to columns* to automatically separate data.

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 5.7, and press Enter. Now you have successfully split the full address into four columns.

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

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

(&). 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

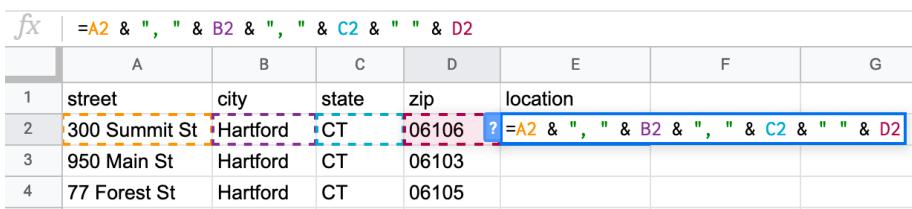
But imagine you need to geocode the addresses using a tool like the one we introduced in Chapter 3, which requires all of the data to be combined into one column like this:

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 5.8, 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.



	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 5.8: 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 7

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.¹

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 5.9. 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

¹Lisa 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>

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.

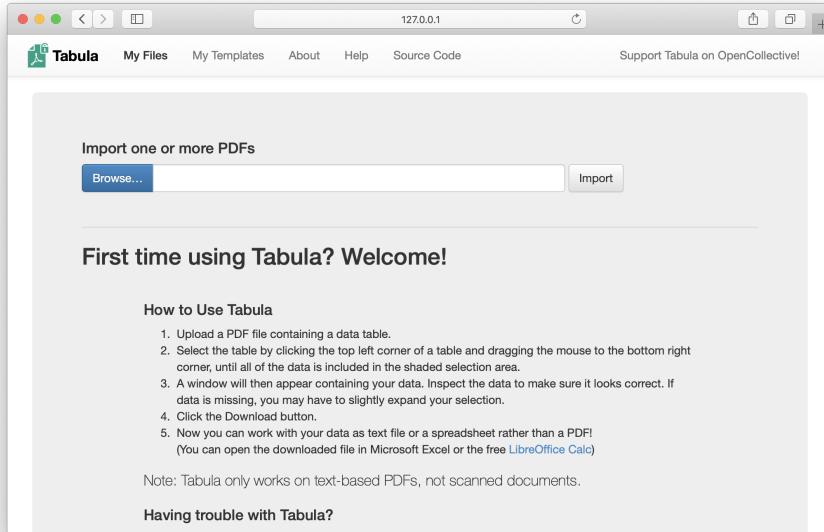


Figure 5.9: Tabula welcome page.

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 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 5.10.

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.

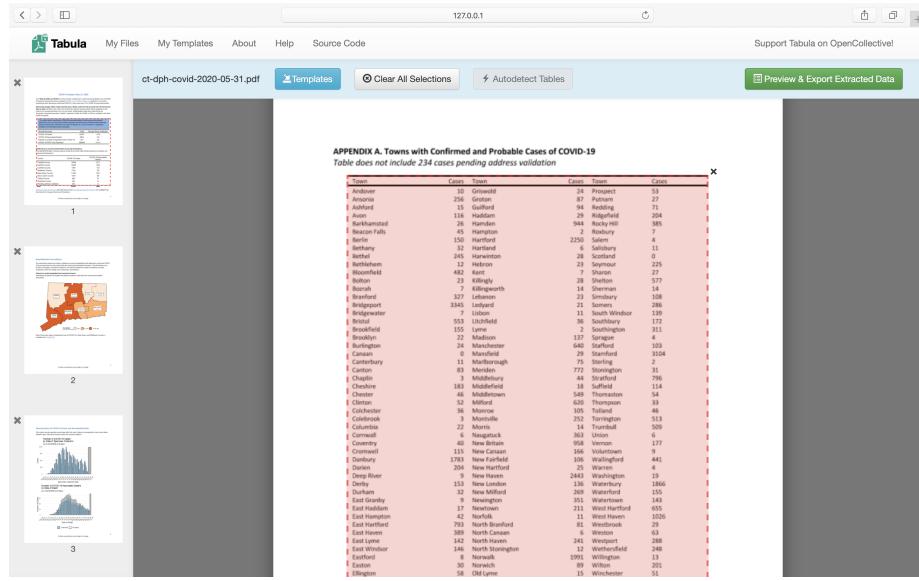


Figure 5.10: Click *Autodetect Tables*, which Tabula will highlight in red.

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.
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 5.11. 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 5.11: 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 work on a Mac, the downloaded file will be a .dmg file. You will likely encounter a security message that will prevent OpenRefine from launching be-

cause Apple cannot identify the developer. Go to *System Preferences > Security and Privacy > General tab*, and click the *Open Anyway* button in the lower half of the window. If prompted with another window, click *Open*.

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

Once launched, you should see OpenRefine in your browser with 127.0.0.1:3333 address (localhost, port 3333), as shown in Figure 5.12.

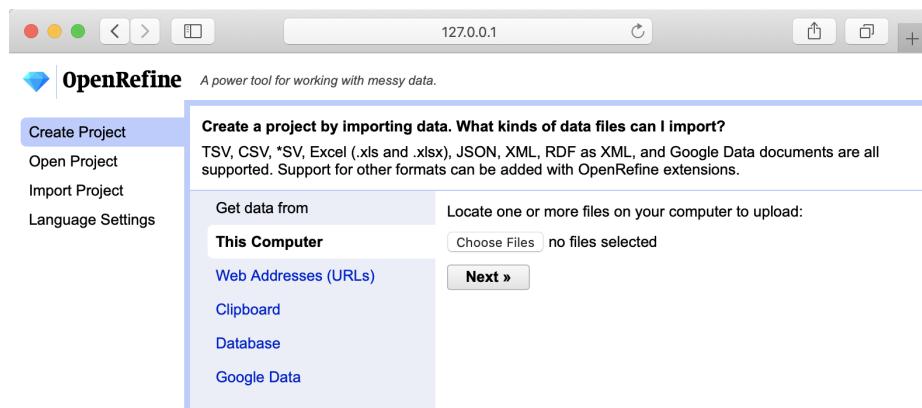


Figure 5.12: OpenRefine starting page.

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 5.13.
4. Hit *Create Project* in the upper-right corner.

Now when you've successfully read the data into a new project, let's start the

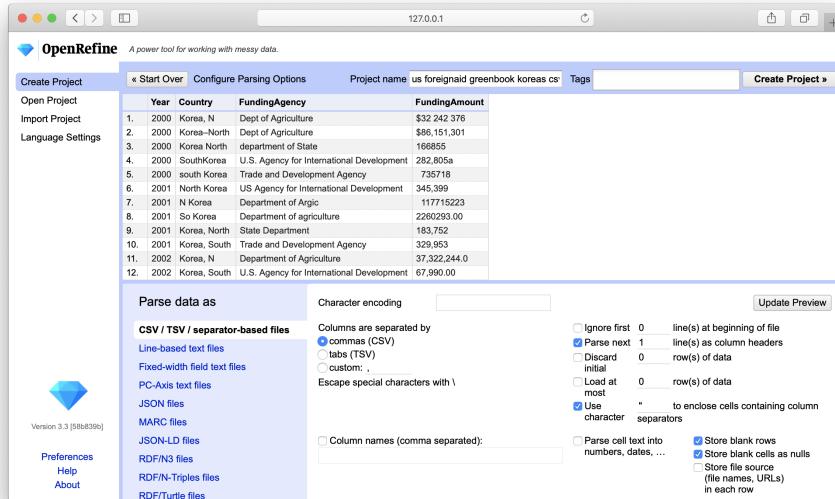


Figure 5.13: OpenRefine parsing options.

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 5.14).

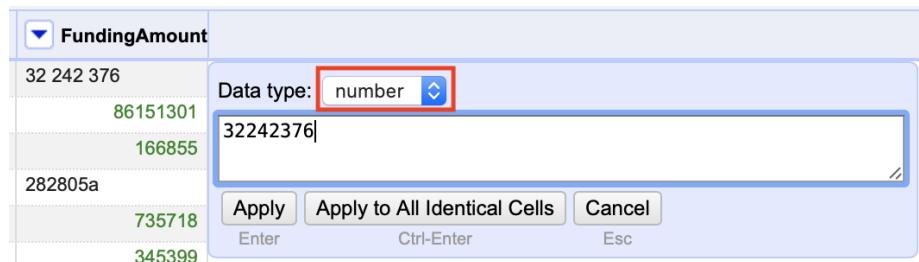


Figure 5.14: Manually remove spaces and extra characters, and change data type to number.

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 5.15.

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 5.15). 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.

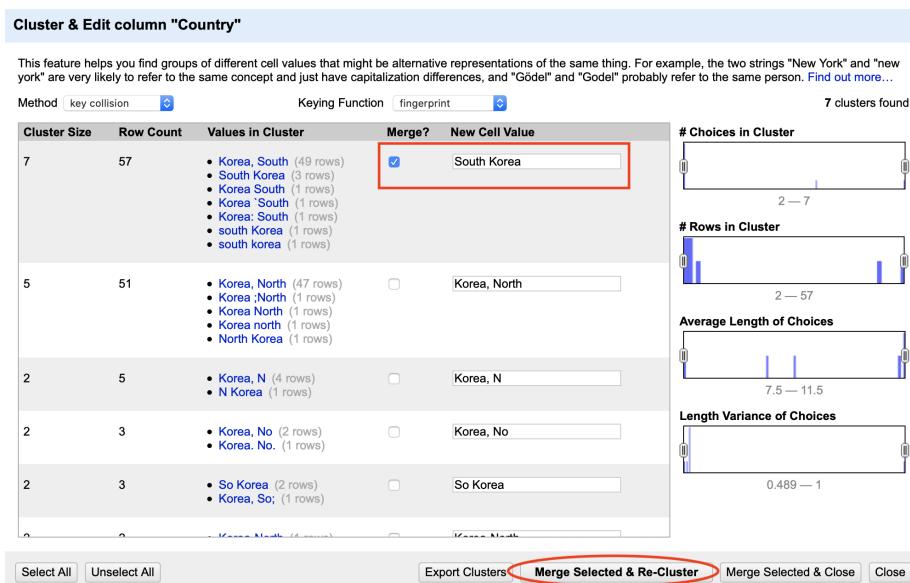


Figure 5.15: Cluster similar text values.

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 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

Chart Your Data

Charts pull readers deeper into your story. Even if your data contains geographical information, sometimes a chart tells your story better than a map. But designing meaningful, interactive charts requires careful thought about how to communicate your data story with your audience.

In this chapter, we will look at main principles of chart design, and learn to identify good charts from bad ones. You will learn important rules that apply to all charts, and also some aesthetic guidelines to follow when customizing your own designs. While all of the tools we describe allow you to create *static* charts that you can download as PNG or JPG files, this book focuses on creating *interactive* charts that reveal more details about your data when you float your cursor over them in your web browser. Later you'll learn how to embed interactive charts on your website in Chapter 9.

To begin, decide which type of chart you wish to create in Table 6.1. Your decision will be based on the format of your data, and the story you wish to tell, such as the type of data comparison you wish to draw to your reader's attention. Once you choose your chart type, follow our tool recommendations and step-by-step tutorials. This chapter features easy drag-and-drop tools such as Google Sheets, Datawrapper, and Tableau Public. But Table 6.1 also refers to more powerful and customizable chart code templates, such as Chart.js and Highcharts templates in Chapter 11, which give you ever more control over customizing your design and storing your data, but request you to learn how to edit and host code templates with GitHub in Chapter 10.

Note that we blend our tutorials for *bar* and *column charts* because they're essentially the same, except that bars are oriented horizontally and columns vertically. Look at your data labels when deciding which type to create. If you need to display long labels, such as "Mocha Frappuccino 24-ounce" and "Double Quarter Pounder with cheese," make a horizontal bar chart to create sufficient space to make them readable. Or if you have short labels, such as "Starbucks"

and “McDonald’s,” you can create either a bar or column chart.

Table 6.1: Basic Chart Types and Tutorials

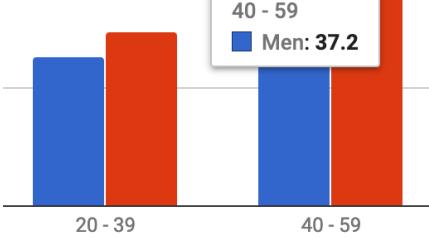
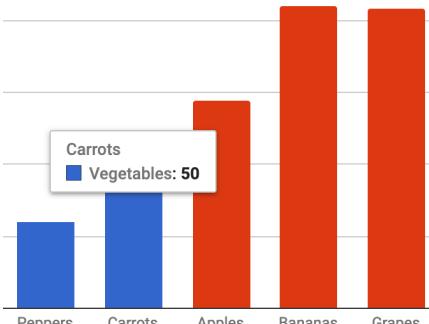
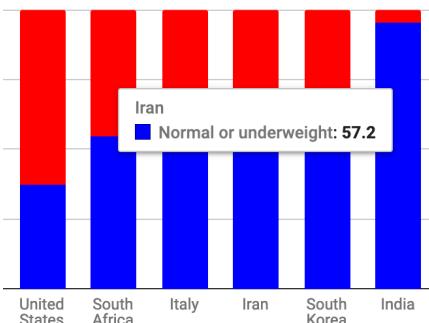
Chart	Best use and tutorials in this book																					
Grouped bar or column chart	<p>Best to compare categories side-by-side. If labels are long, use horizontal bars instead of vertical columns. Easy tools: Bar and Column Charts in Google Sheets tutorialPower tool: Chart.js and Highcharts templates</p>  <p>A grouped bar chart comparing men's and women's heights across two age groups: 20-39 and 40-59. The x-axis labels are '20 - 39' and '40 - 59'. The y-axis has numerical tick marks. For each age group, there are two bars: a blue one for men and a red one for women. A tooltip for the blue bar in the '40 - 59' group shows 'Men: 37.2'.</p> <table border="1"> <thead> <tr> <th>Age Group</th> <th>Gender</th> <th>Height (approx.)</th> </tr> </thead> <tbody> <tr> <td>20 - 39</td> <td>Men</td> <td>37.2</td> </tr> <tr> <td>20 - 39</td> <td>Women</td> <td>40.5</td> </tr> <tr> <td>40 - 59</td> <td>Men</td> <td>37.2</td> </tr> <tr> <td>40 - 59</td> <td>Women</td> <td>40.5</td> </tr> </tbody> </table>	Age Group	Gender	Height (approx.)	20 - 39	Men	37.2	20 - 39	Women	40.5	40 - 59	Men	37.2	40 - 59	Women	40.5						
Age Group	Gender	Height (approx.)																				
20 - 39	Men	37.2																				
20 - 39	Women	40.5																				
40 - 59	Men	37.2																				
40 - 59	Women	40.5																				
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: Chart.js and Highcharts templates</p>  <p>A split bar chart comparing vegetable and fruit consumption. The x-axis labels are 'Peppers', 'Carrots', 'Apples', 'Bananas', and 'Grapes'. The y-axis has numerical tick marks. For each category, there are two bars: a blue one for vegetables and a red one for fruits. A tooltip for the blue bar in the 'Carrots' category shows 'Vegetables: 50'.</p> <table border="1"> <thead> <tr> <th>Category</th> <th>Vegetable</th> <th>Fruit</th> </tr> </thead> <tbody> <tr> <td>Peppers</td> <td>50</td> <td>0</td> </tr> <tr> <td>Carrots</td> <td>50</td> <td>0</td> </tr> <tr> <td>Apples</td> <td>0</td> <td>50</td> </tr> <tr> <td>Bananas</td> <td>0</td> <td>50</td> </tr> <tr> <td>Grapes</td> <td>0</td> <td>50</td> </tr> </tbody> </table>	Category	Vegetable	Fruit	Peppers	50	0	Carrots	50	0	Apples	0	50	Bananas	0	50	Grapes	0	50			
Category	Vegetable	Fruit																				
Peppers	50	0																				
Carrots	50	0																				
Apples	0	50																				
Bananas	0	50																				
Grapes	0	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: Chart.js and Highcharts templates</p>  <p>A stacked bar chart showing overweight status by country. The x-axis labels are 'United States', 'South Africa', 'Italy', 'Iran', 'South Korea', and 'India'. The y-axis has numerical tick marks. Each bar is composed of two segments: a blue segment at the bottom and a red segment above it. A tooltip for the blue segment in the 'Iran' bar shows 'Normal or underweight: 57.2'.</p> <table border="1"> <thead> <tr> <th>Country</th> <th>Normal or underweight</th> <th>Overweight</th> </tr> </thead> <tbody> <tr> <td>United States</td> <td>57.2</td> <td>42.8</td> </tr> <tr> <td>South Africa</td> <td>57.2</td> <td>42.8</td> </tr> <tr> <td>Italy</td> <td>57.2</td> <td>42.8</td> </tr> <tr> <td>Iran</td> <td>57.2</td> <td>42.8</td> </tr> <tr> <td>South Korea</td> <td>57.2</td> <td>42.8</td> </tr> <tr> <td>India</td> <td>57.2</td> <td>42.8</td> </tr> </tbody> </table>	Country	Normal or underweight	Overweight	United States	57.2	42.8	South Africa	57.2	42.8	Italy	57.2	42.8	Iran	57.2	42.8	South Korea	57.2	42.8	India	57.2	42.8
Country	Normal or underweight	Overweight																				
United States	57.2	42.8																				
South Africa	57.2	42.8																				
Italy	57.2	42.8																				
Iran	57.2	42.8																				
South Korea	57.2	42.8																				
India	57.2	42.8																				

Chart	Best use and tutorials in this book
Histogram	<p>Best to show distribution of raw data, with number of values in each bucket. If labels are long, use horizontal bars instead of vertical columns..Easy tools: Bar and Column Charts in Google Sheets tutorialPower tool: Chart.js and Highcharts templates</p>
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.Easy tool: TODO DECIDE Power tool: Chart.js and Highcharts templates</p>
Pie chart	<p>Best to show parts of a whole, but hard to estimate size of slices.Easy tools: Pie Chart in Google Sheets tutorialPower tool: Chart.js and Highcharts templates</p>

Chart	Best use and tutorials in this book
Line chart	Best to show continuous data, such as change over time. Easy tools: Line Chart in Google Sheets tutorialPower tool: Chart.js and Highcharts templates
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: 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

Chart	Best use and tutorials in this book
Range chart	Best to show gaps between data points, such as inequalities. Easy tools: Range Chart in Datawrapper tutorial Power tool: TODO DECIDE
Scatter chart	Best to show the relationship between two datasets as XY coordinates to reveal possible correlations. Easy tool: Scatter and Bubble Chart in Datawrapper tutorial or Scatter Chart in Tableau Public tutorial. Power tool: 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: Chart.js and Highcharts templates
Sparklines	Best to compare data trends with tiny line or bar charts, aligned in a table column. Easy tool: Interactive Table with Sparklines in Datawrapper tutorial

Chart Design Principles

Although not a science, data visualization comes with a set of rules, principles, and best practices that create a basis for clear and eloquent charts. Some of those rules are less rigid than others, but prior to “breaking” them, it is important to establish why they are important.

Before you begin, ask yourself: Do I really need a chart to tell this data story? Or would a table or text alone do a better job? Making a good chart takes time

and effort, so make sure it enhances your story.

Deconstructing a Chart

Let's take a look at Figure 6.1. It shows basic chart components that are shared among most chart types.

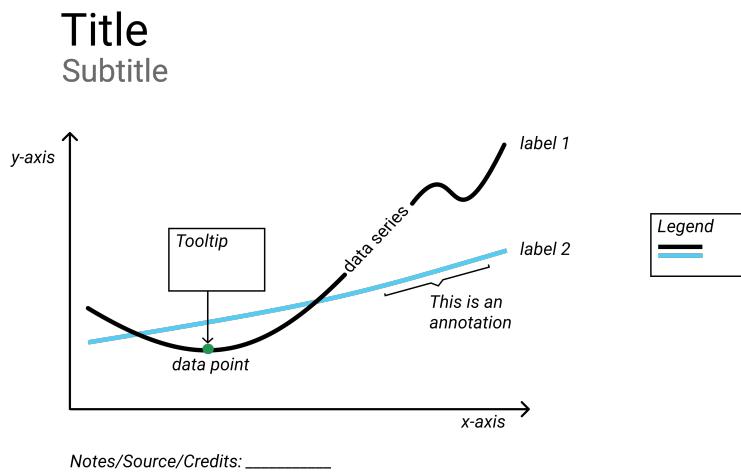


Figure 6.1: Common chart components.

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, there are some that are hard to bend.

Bar charts must start at zero

Bar charts use *length* to represent value, therefore their value axis *must start at zero*. That applies to column and area charts as well. This is to ensure that a bar twice the length of another bar represents twice its value. The Figure 6.2 shows a good and a bad example.

Starting y-axis at anything other than zero is a common trick used by some media and politicians to exaggerate differences in surveys and election results. Learn more about how to detect bias in data stories in Chapter 14.

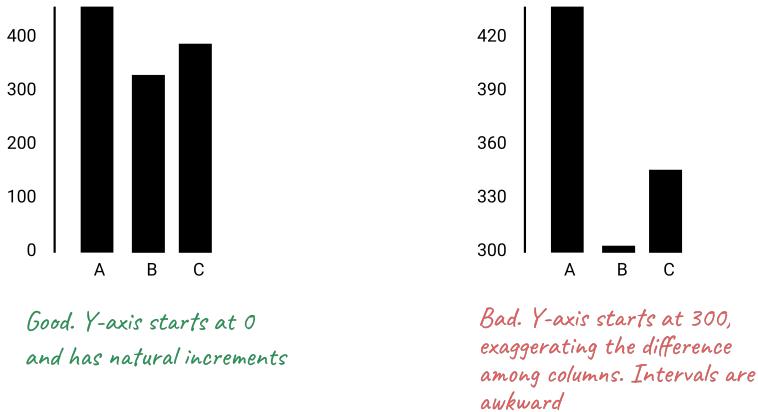


Figure 6.2: Start your bar chart at zero.

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 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 absolute numbers, 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 6.3, 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.

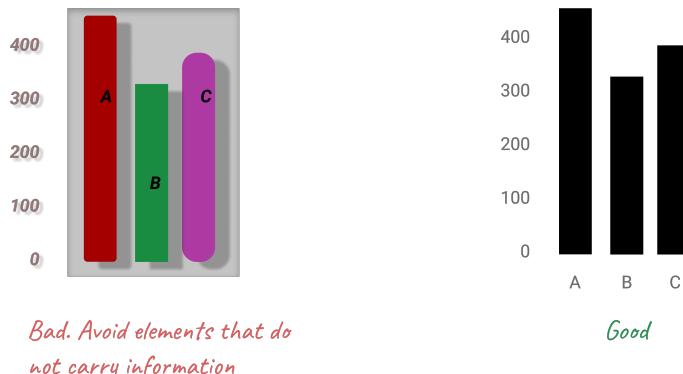


Figure 6.3: Chart junk distracts the viewer, so stay away from shadows, 3D perspectives, unnecessary colors and other fancy elements.

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 6.4 illustrates that.

If your pie chart has more than five slices, consider showing your data in a bar chart, either stacked or split, like Figure 6.5 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 6.6 to see how much easier it is to read horizontally-oriented labels.

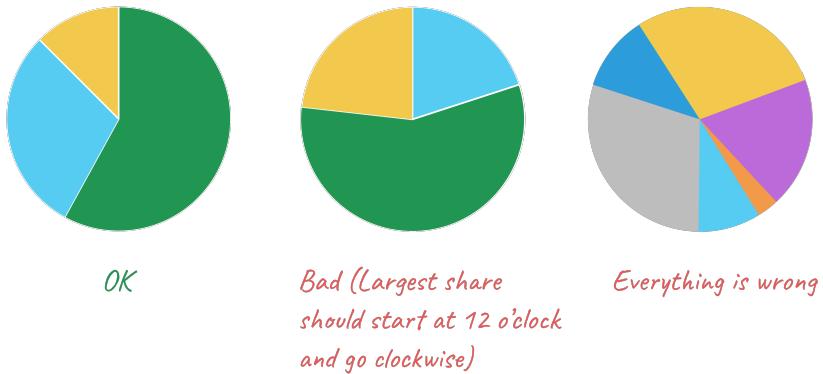


Figure 6.4: Sort slices in pie charts from largest to smallest, and start at 12 o'clock.

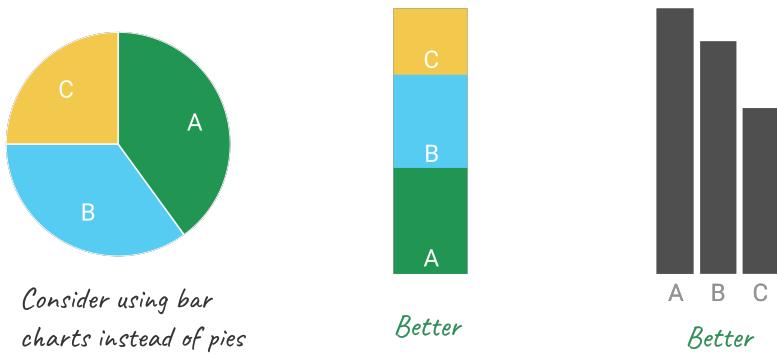


Figure 6.5: Consider using bar charts instead of pies.

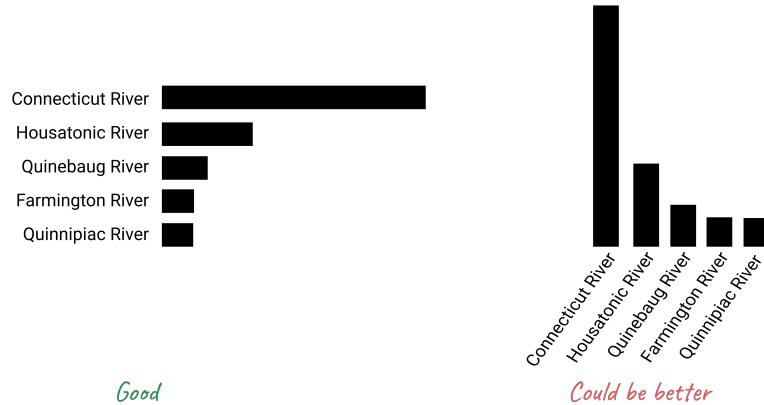


Figure 6.6: For long labels, use horizontal bar charts.

Arrange elements logically

If your bar chart shows different categories, consider ordering them, like is shown in Figure 6.7. 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.

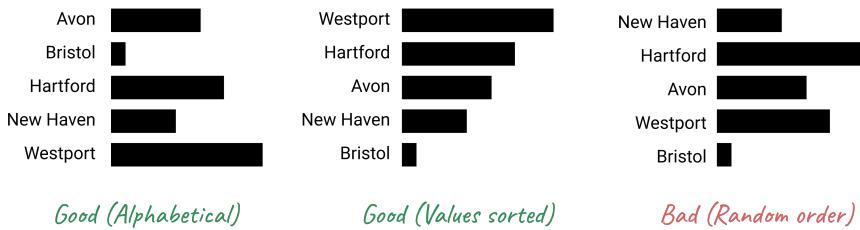


Figure 6.7: For long labels, use horizontal bar charts.

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 the colors

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" blog post for Datawrapper.¹ But some principles are fairly universal. First, do not use colors just for the sake of it, most charts are fine being monochromatic. Second, remember that colors come with some meaning attached, which can vary among cultures. In the world of business, red is conventionally used to represent loss, and it would be unwise to use this color to show profit. Make sure you avoid random colors.

Whatever colors you end up choosing, they need to be distinguishable (otherwise what is the point?). Do not use colors that are too similar in hue (for example, various shades of green—leave them for choropleth maps). Certain color combinations are hard to interpret for color-blind people, like green/red or yellow/blue, so be very careful with those. Figure 6.8 shows some good and bad examples of color use.

If you follow the advice, you should end up with a de-cluttered chart as shown in Figure 6.9. Notice how your eyes are drawn to the bars and their corresponding values, not bright colors or secondary components like the axes lines.

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* 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 9.

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

¹Lisa Charlotte Rost, "Your Friendly Guide to Colors in Data Visualisation," Chartable: A Blog by Datawrapper, July 31, 2018, <https://blog.datawrapper.de/colorguide/>



Figure 6.8: Don't use colors just for the sake of it.



Figure 6.9: Make sure important things catch the eye first.

Sheets does not meet your needs, refer back to Table 6.1 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 four different types: grouped, split, stacked, and histograms. 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 6.10. Now you can easily create a grouped column chart to displays these data series side-by-side, as shown in Figure 6.11.

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

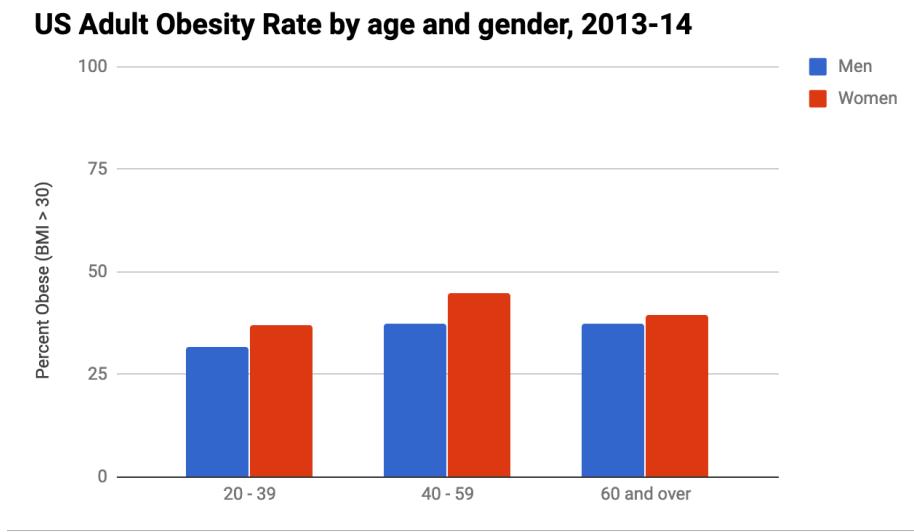


Figure 6.11: Grouped Column chart: Explore the interactive version. Data from StateOfObesity.org.

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 6.12.

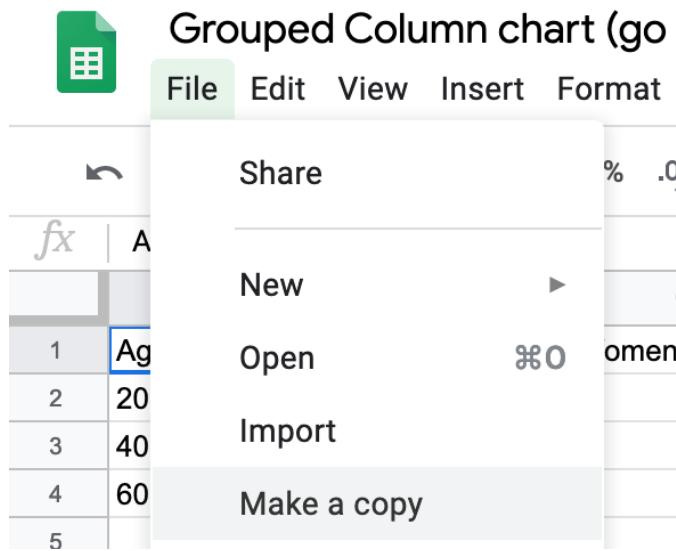


Figure 6.12: Make your own copy of the Google Sheet template.

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 6.13.

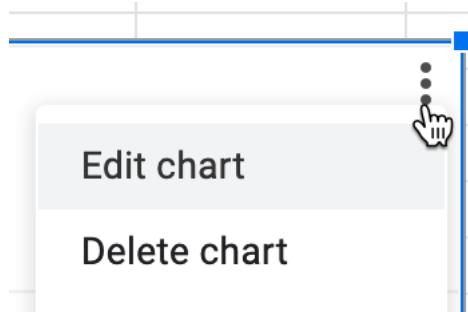


Figure 6.13: Float cursor in top-right corner of the chart to make the 3-dot kebab menu appear, and select Delete.

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 6.14. Software developers must be hungry.

3. Format your data to make each column a data series (such as male and



Figure 6.14: Distinguish between the hamburger verus kebab menu icons.

female), as shown in Figure 6.10, 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 6.15.
7. In the Chart Editor, change the default selection to *Column chart*, with *Stacking none*, to display Grouped Columns, as shown in Figure 6.16. 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 6.17. Also, you can select the chart and axis titles to edit them.
9. 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 6.18. This means your sheet is no longer Restricted to only you, but can be viewed by anyone with the link. See additional options.
10. 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 6.19. In the next screen, select Embed and press the Publish button. See Chapter 9: Embed on the Web to learn what to do with the iframe code.

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

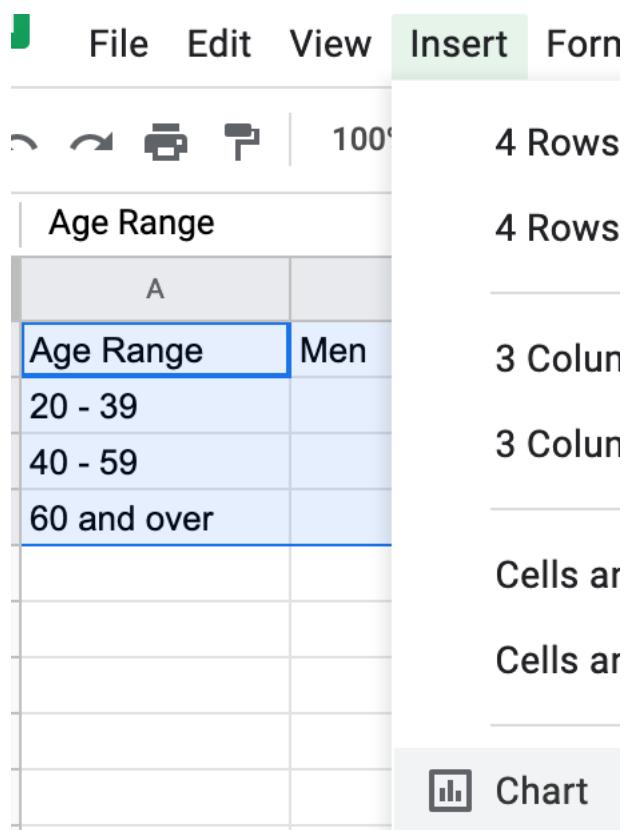


Figure 6.15: Select your data and Insert the Chart.

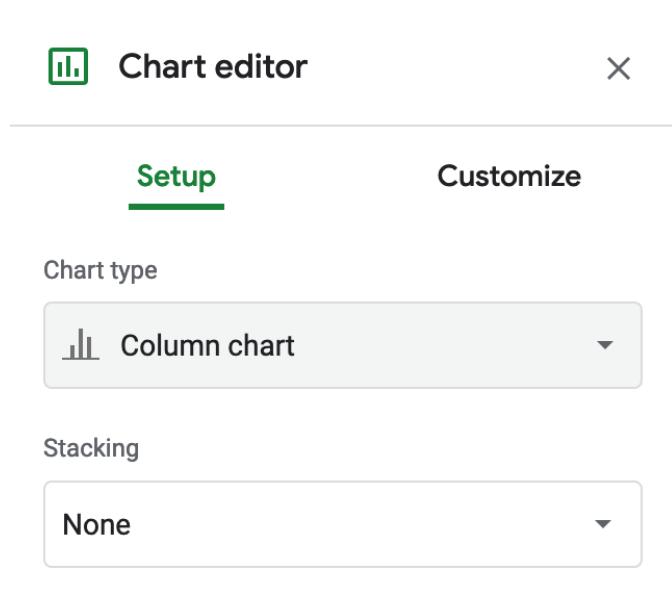


Figure 6.16: Change the default to Column chart, with Stacking none.

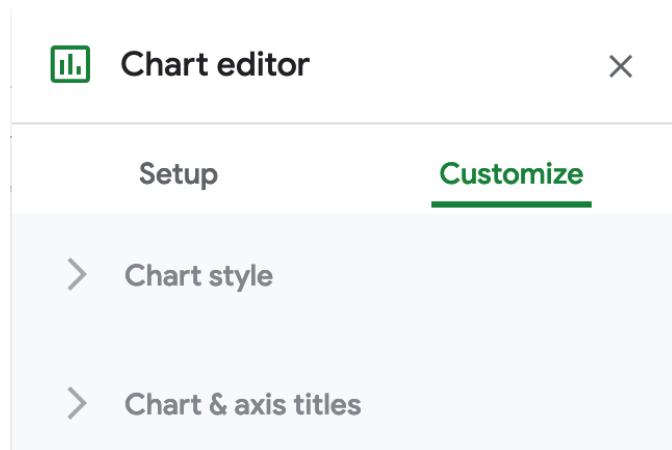


Figure 6.17: Select Customize to edit title, labels, and more.

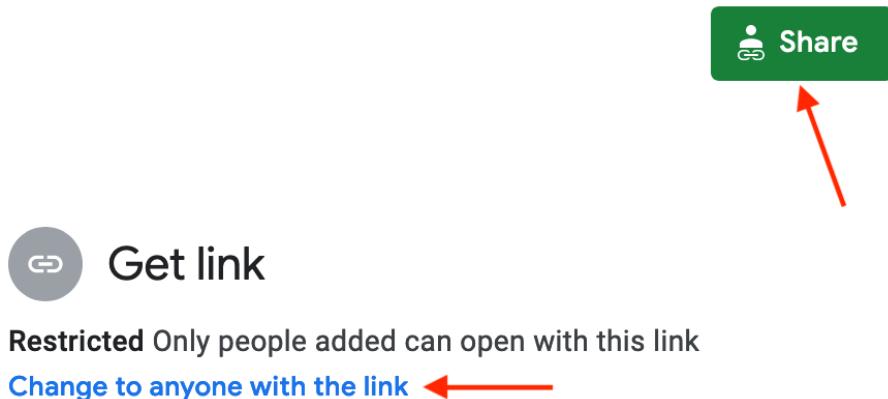


Figure 6.18: Click the Share button and then click *Change to anyone with the link* to make your data public.

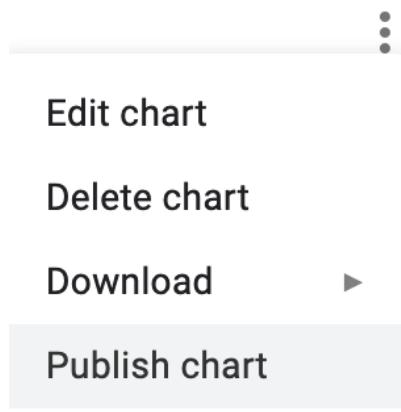


Figure 6.19: Select Publish Chart to embed an interactive chart on another web page.

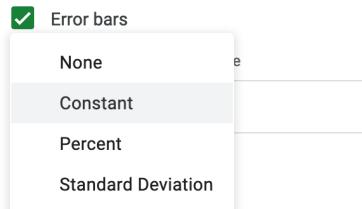


Figure 6.20: Google Sheets is very limited when it comes to setting error bars.

percent or a constant value, as shown in Figure @ref(fig:images/06-chart/chart-error-bar.png). This setting will then be applied to all data points from that series.

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 6.21. 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 6.22. Unlike the grouped column chart previously shown in Figure 6.11, 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	McDonald
2	Mocha Frappuccino (24-ounce, 2% milk, whip cream)	500	
3	White Hot Chocolate (20-ounce, 2% milk, whip cream)	710	
4	Big Mac		5
5	Double Quarter Pounder with cheese		7

Figure 6.21: To create a split bar (or column) chart, format each data series vertically, and leave cells blank where appropriate.

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.

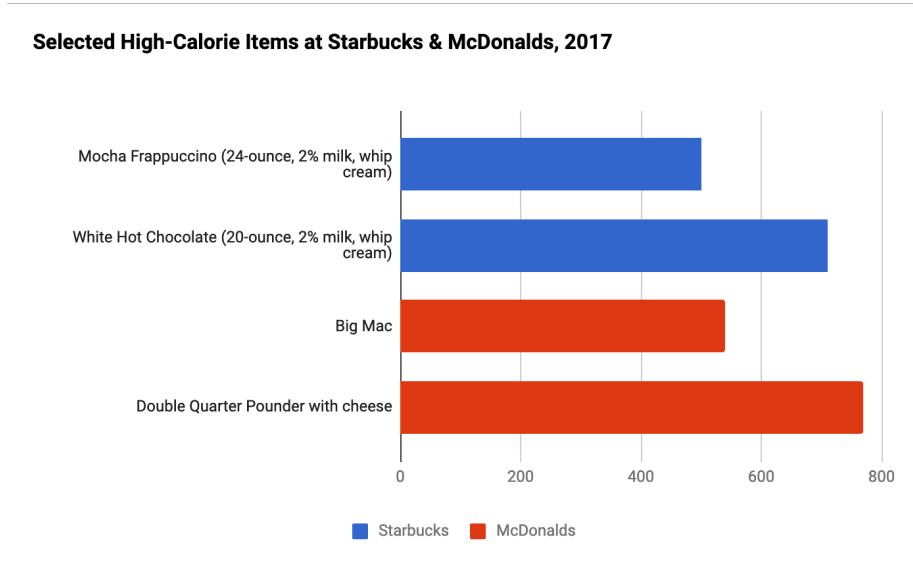


Figure 6.22: Split bar chart: Explore the full-screen interactive version. Data from Starbucks and McDonalds.

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 6.23. Now you can easily create a stacked column (or bar) chart that displays comparisons of weight-level subcategories across nations, as shown in Figure 6.24. 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 *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 6.25.

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

Percent of Population by Weight Level for most recent year

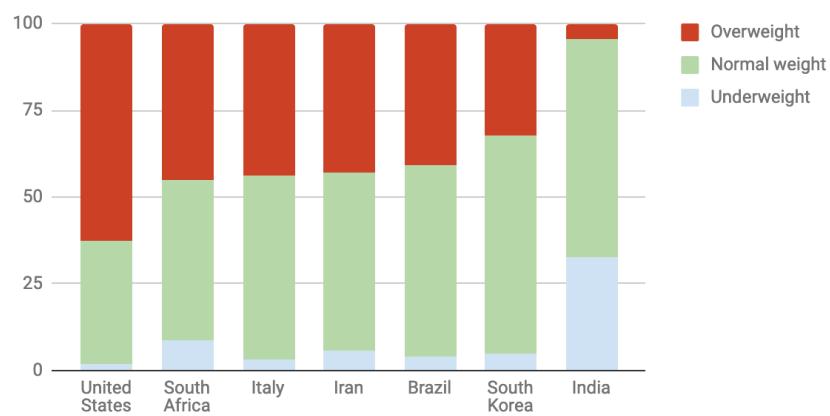


Figure 6.24: Stacked column chart: Explore the interactive version. Data from WHO and CDC.

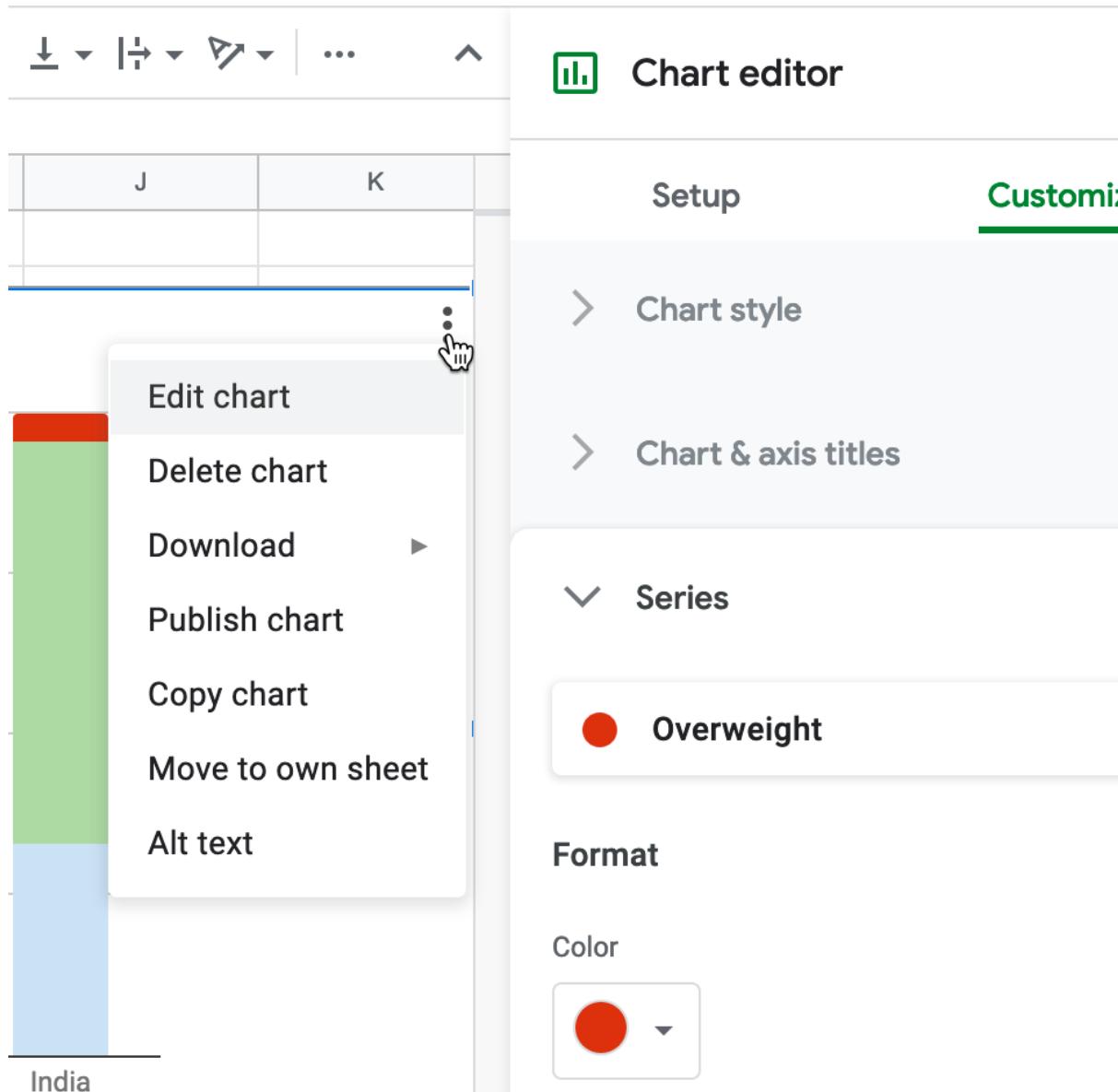


Figure 6.25: To edit a column color, select *Edit Chart - Customize - Series*.

Histograms

A histogram is a specific type of bar or column chart that is best for showing the distribution of raw data, with the number of values in each bucket (or bin). To build a histogram, you need to assign each data point, whether numerical or categorical, into one of the non-overlapping buckets. For example, imagine that you wish to track the number of customers each hour in a local coffee shop. Format the raw customer data series in a vertical column in your Google Sheet, as shown in Figure 6.26. Now you can easily create a histogram column chart that displays the number of customers per hour, as shown in Figure 6.27, which resembles the “popular times” format for businesses in Google Maps. This coffee shop experiences a morning rush and an afternoon rush, but the middle of the day and late evenings are relatively quiet.

	A	B
1	Hour	Customers
2	8 AM	17
3	9 AM	35
4	10 AM	27
5	11 AM	15
-	12 PM	-

Figure 6.26: To create a histogram, format the raw data series vertically in Google Sheets.

Create your own version using our Histogram Chart in Google Sheets template with coffee shop customer data. Google Sheets considers histograms to be regular column charts, so in the Chart Editor window, choose *Chart Type > Column chart* and follow the rest of the directions in the Bar and Column Chart tutorial above. Alternatively, you could choose to sort customers into larger bins, such as time of day (morning, afternoon, evening), as shown in the second tab of the template.

Tip: We set a custom number format to display *8 AM* and other times as desired in the *Hour* column. In Google Sheets, select a column, then go to *Format > Number > More Formats > Custom Number Formats* to define your preferred format.

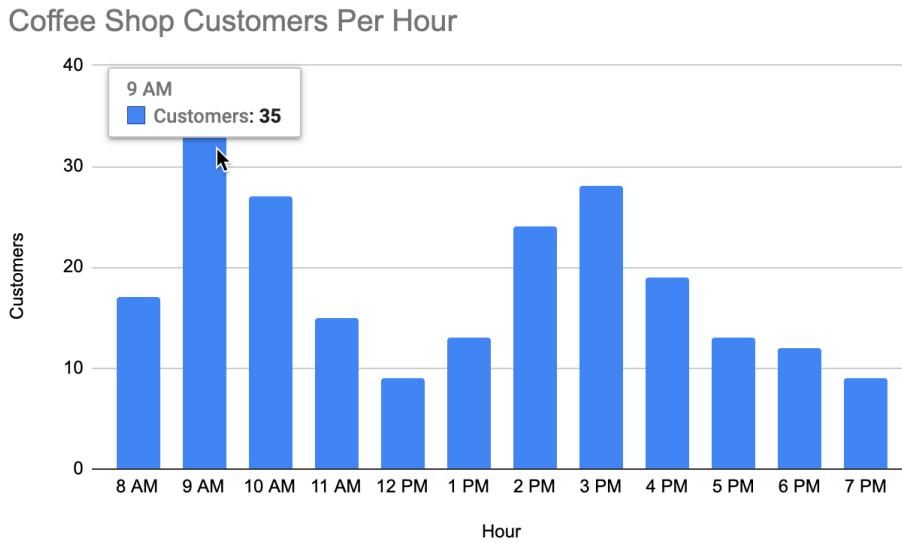


Figure 6.27: Histogram: Explore the interactive version. Fictitious data on coffee shop customers.

- 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 6.1 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 6.28. 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 6.29. 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 6.28: To create a pie chart, format the data values vertically in Google Sheets.

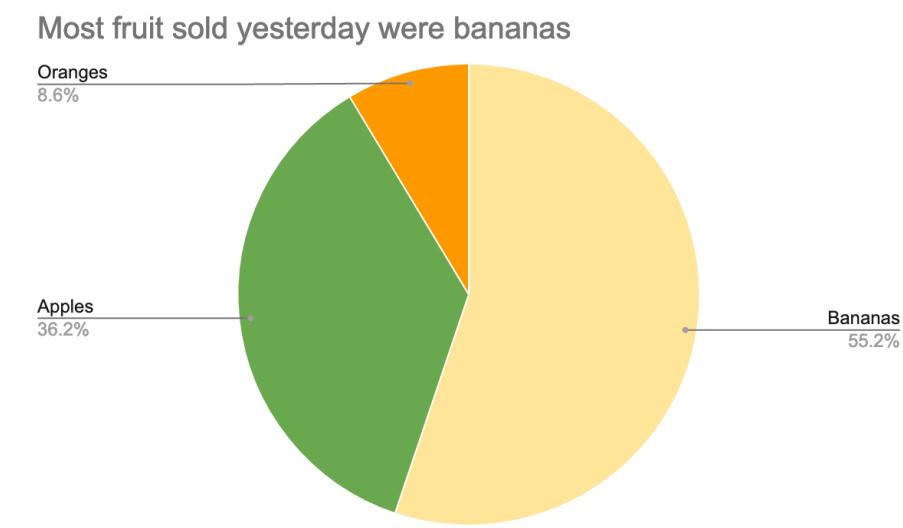


Figure 6.29: Pie chart: Explore the interactive version. Data is fictitious.

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 6.30. Now you can easily create a line chart that emphasizes each data series changed over time, as shown in Figure 6.31. 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.

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

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

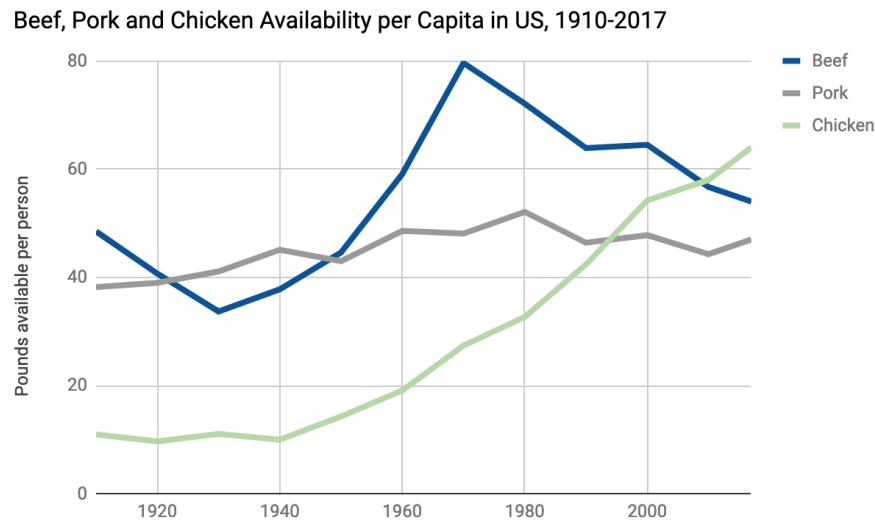


Figure 6.31: Line chart: Explore the interactive version. Data from US Department of Agriculture.

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 6.30. 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 6.32. 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.

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.

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

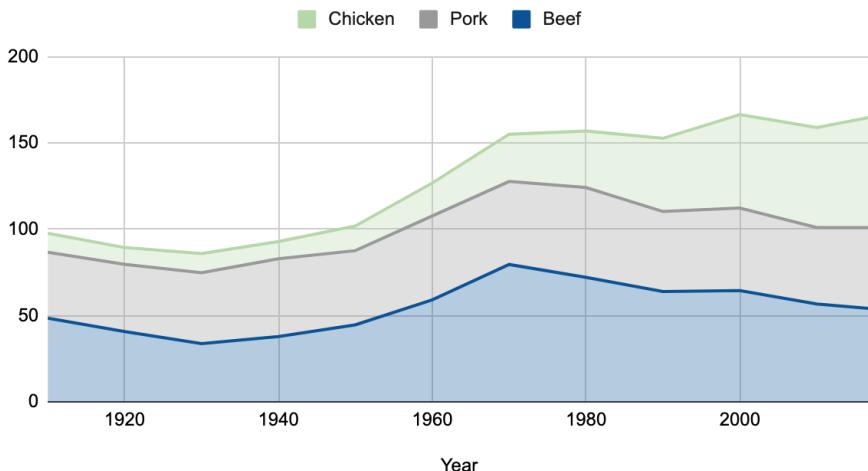


Figure 6.32: Stacked area chart: Explore the interactive version. Data from US Department of Agriculture.

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 maps and tables, which we'll discuss later in chapters 6 and 7. 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 9. 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 6.1 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 6.33. 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.
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 6.34, 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*

US Unemployment Rate, Seasonally Adjusted, 2000-2020

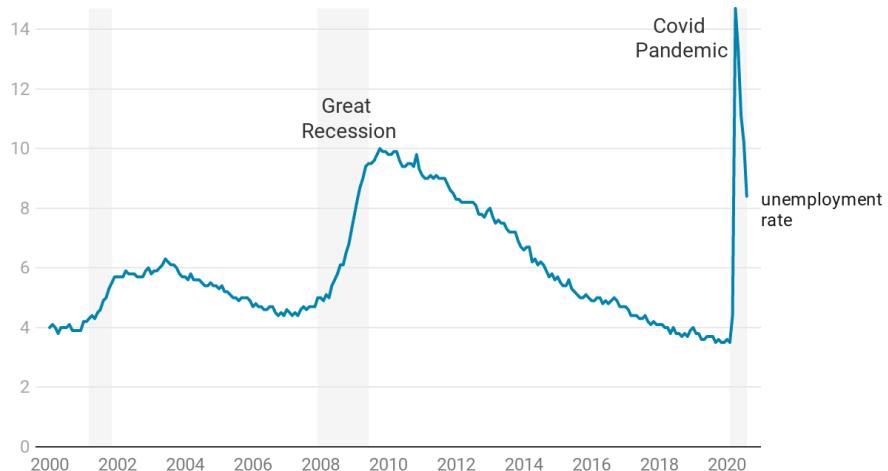


Chart: HandsOnDataViz.org • Source: US Federal Reserve Open Data • Created with Datawrapper

Figure 6.33: Line chart with annotation: Explore the interactive version. Data from US Federal Reserve Open Data.

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.

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.

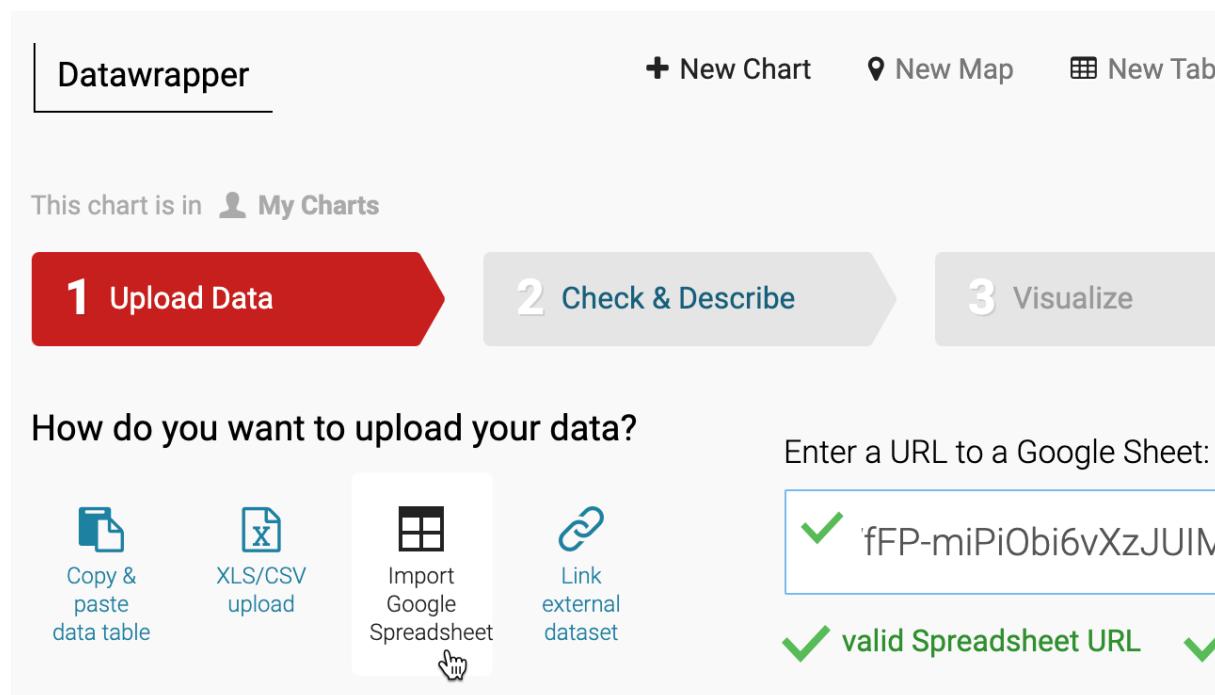


Figure 6.34: To upload data from a shared Google Sheet, click the button and paste the link.

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 6.35. 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 2020 for comparison. You can fine-tune the positioning of a label by typing its year-month code into the x-axis coordinate text field, and its unemployment rate into the y-axis text field. To change the appearance or delete an annotation, click its downward arrow symbol for options.

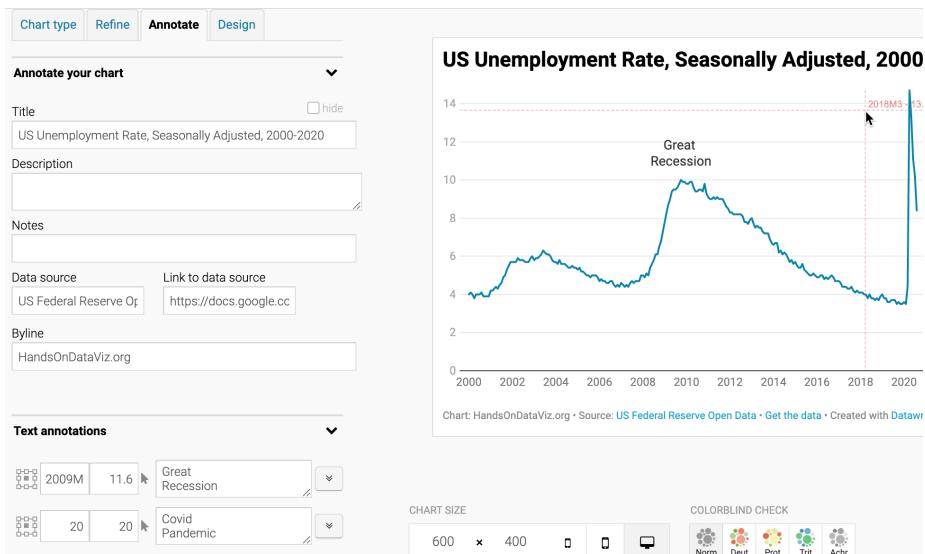


Figure 6.35: Add text annotations by typing a label and positioning it on the chart.

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 6.36. 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).

9. Click *Proceed* or advance to the *Publish & Embed* screen to share your chart with others. If you are logged into your free Datawrapper account, click the blue *Publish chart* button to generate the code to embed the interactive chart on your website, and read more in Chapter 9: Embed on Your Web. Also, click the *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. Finally, 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.

Congratulations on creating your first interactive Datawrapper chart. If you logged into your free Datawrapper account, all of your charts are stored in the *My Charts* menu in the top-right corner of the screen. 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 6.37. 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 6.38. 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.

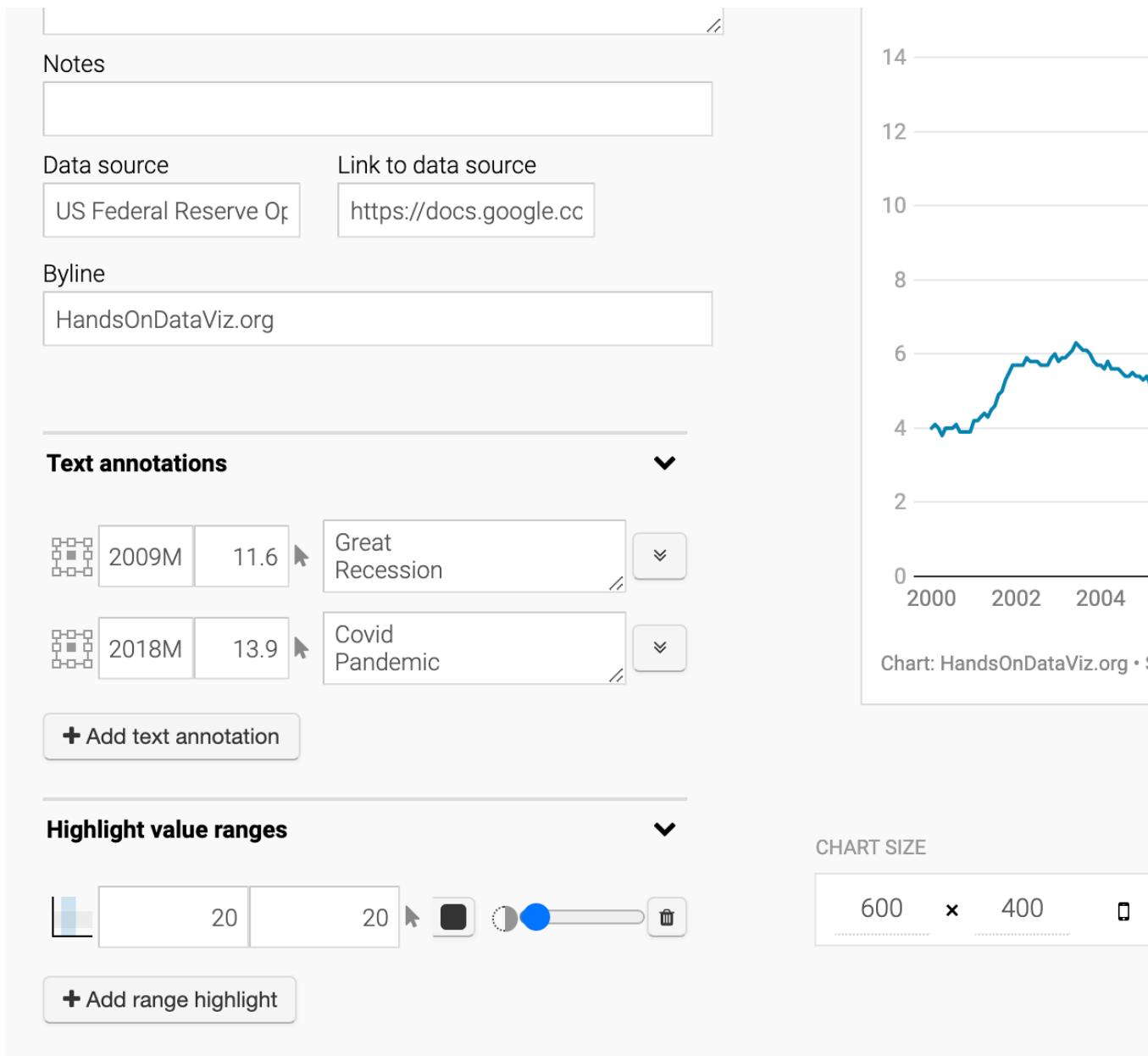


Figure 6.36: Add a range highlight by “drawing” a rectangular bar on the chart.

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

Figure 6.37: Organize your range chart data labels into column headers, with categories and values in each row.

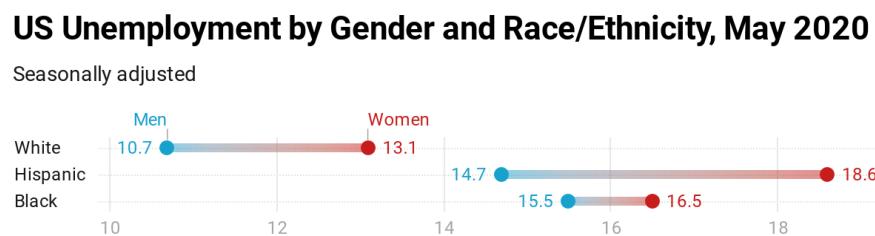


Chart: by HandsOnDataViz.org • Source: Federal Reserve Open Data • Created with Datawrapper

Figure 6.38: 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 6.39.

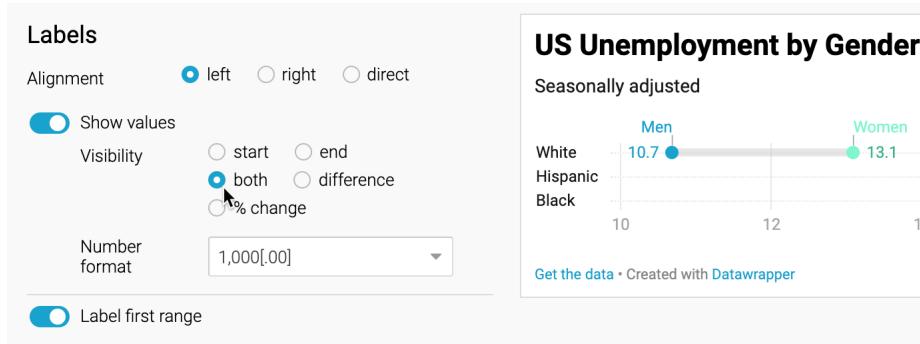


Figure 6.39: Modify the labels settings to show values at both ends of each range, and to place your data labels on your first range.

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 6.40.

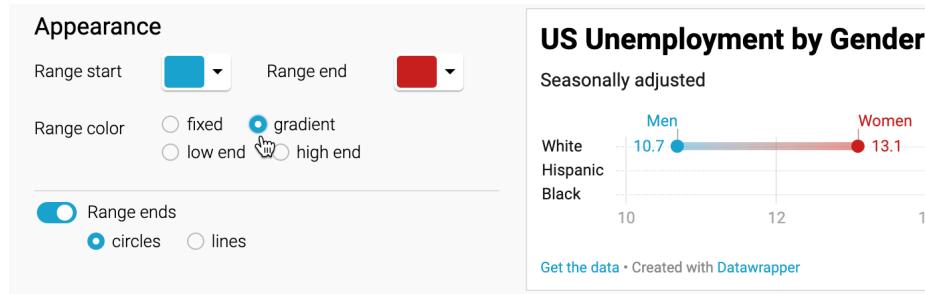


Figure 6.40: Modify the appearance settings to improve the color and add a gradient.

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 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 6.41. 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 6.42. 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 6.41: 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

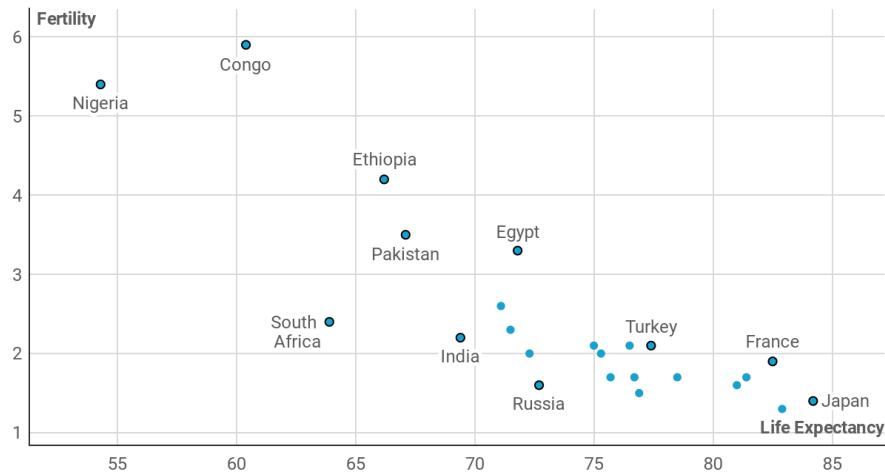


Chart: HandsOnDataViz.org • Source: World Bank • Created with Datawrapper

Figure 6.42: 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 6.43. Press *Save* to close the tooltip editor window.

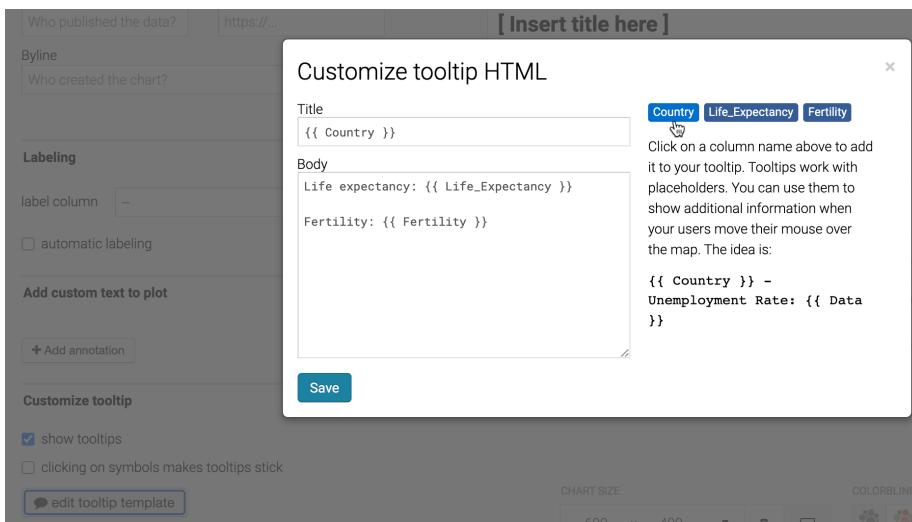


Figure 6.43: In the tooltip editor window, type and click column headers to customize the display.

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 6.44.
8. Finish the annotations to add your title and data source, then proceed to publish and embed your chart, as described in Chapter 9: Embed on Your 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

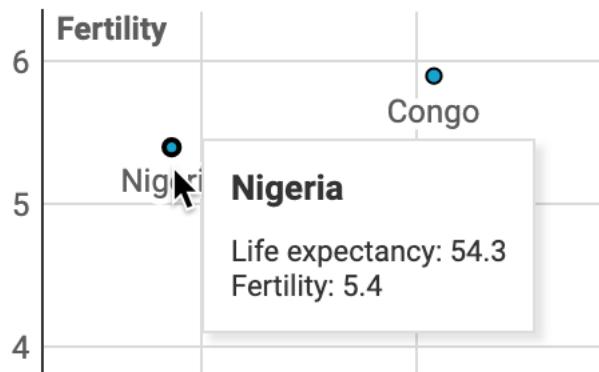


Figure 6.44: Hover over a data point to inspect the edited tooltip display.

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 6.45. 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 6.46.

	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 6.45: 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

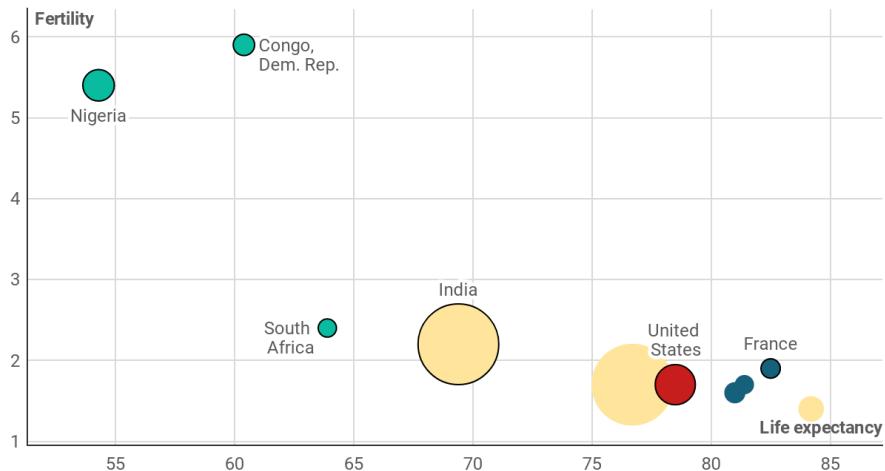


Chart: HandsOnDataViz.org • Source: World Bank • Created with Datawrapper

Figure 6.46: 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 6.47. 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 6.48. 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, as described in Chapter 9: Embed on Your Web.

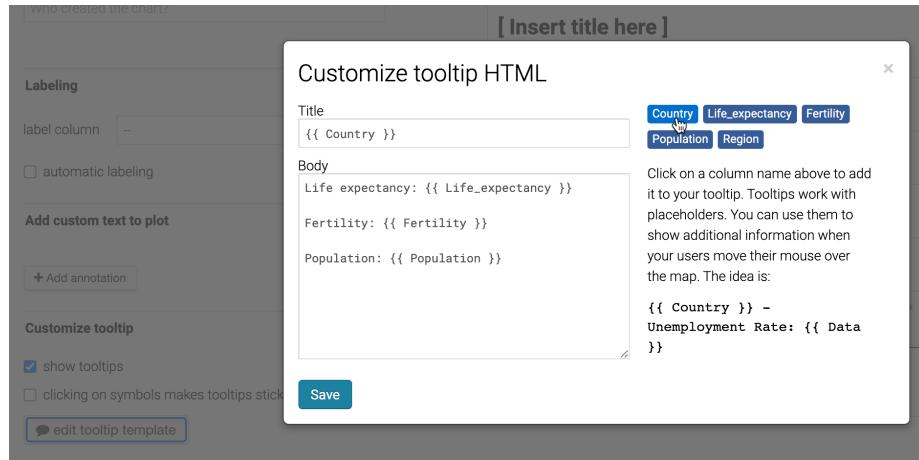


Figure 6.47: In the tooltip editor window, type and click column headers to customize the display.

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 13. 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

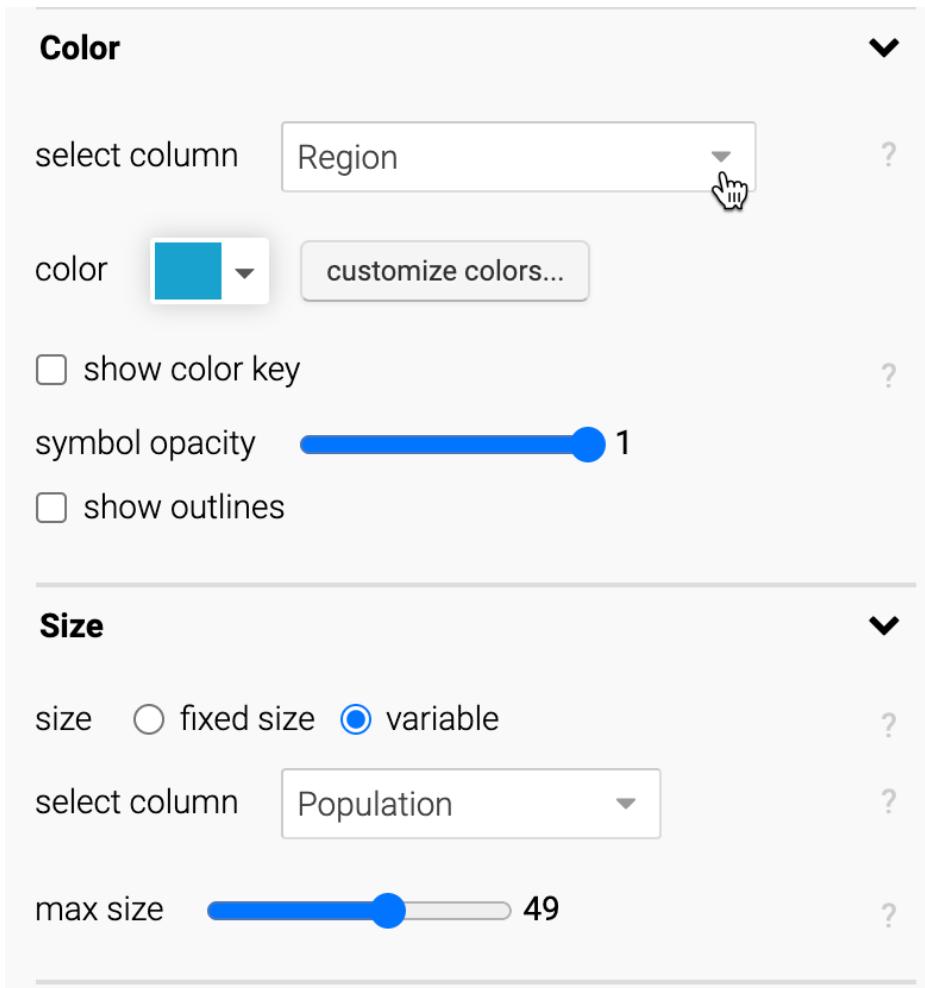


Figure 6.48: In the *Visualize* screen, modify the bubble colors and set size to variable.

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

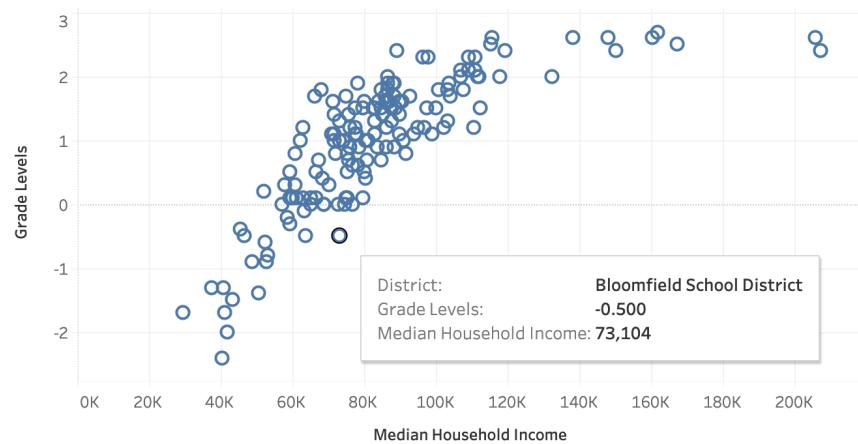
Scatter charts are best to show the relationship between two datasets, placed on the x- and y-axis, to reveal possible correlations. With Tableau Public, you can create an interactive scatter chart, where you can hover your cursor over points to view more details about the data. Organize your data in 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 6.49, 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.

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.

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.

[+ a b | e a u](#) ← → ↵ ⌂ ↴

Figure 6.49: Scatter chart in Tableau Public: Explore the interactive version.
Data by CT Mirror/TrendCT and Stanford CEPA.

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 6.50. 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.

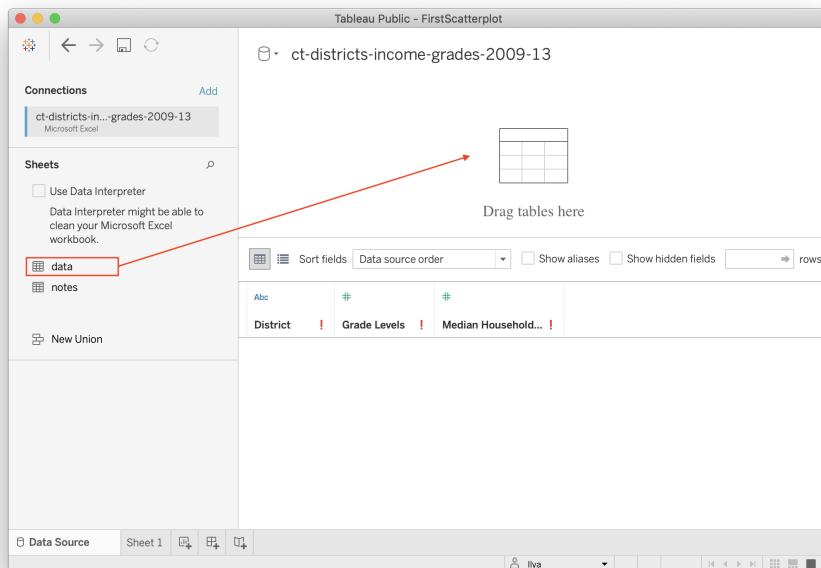


Figure 6.50: Drag data sheet into *Drag tables here* area.

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 6.51 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.

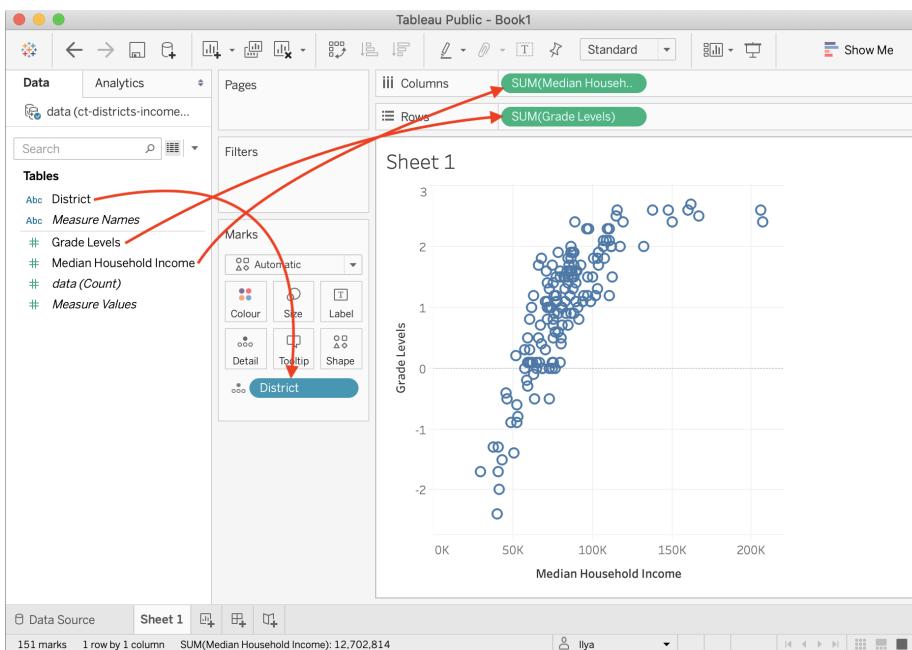


Figure 6.51: Drag data fields to the right locations in Tableau Public.

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 6.52.

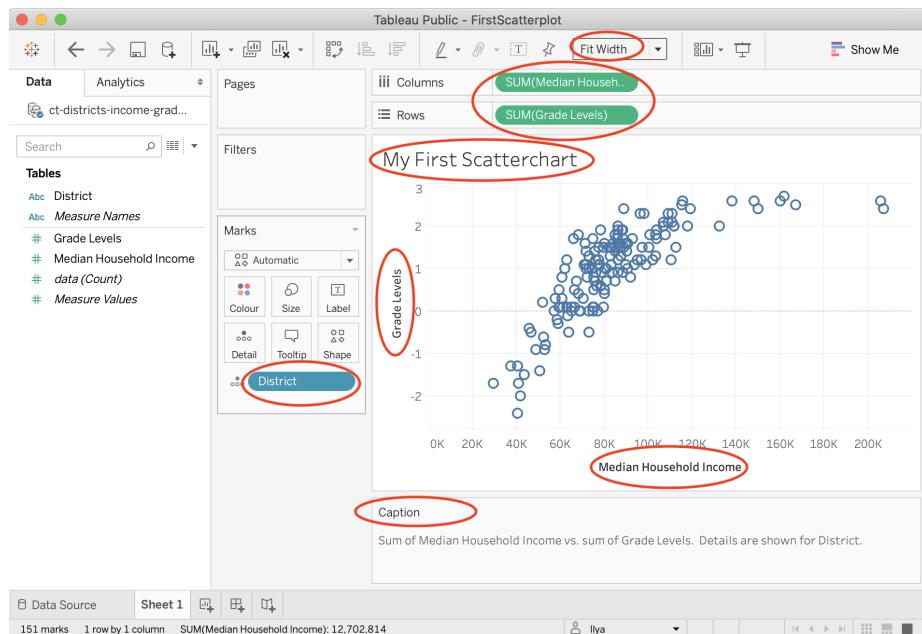


Figure 6.52: This scatter chart is ready to be published.

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

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 6.53: 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 6.54. 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 6.55.

fx

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

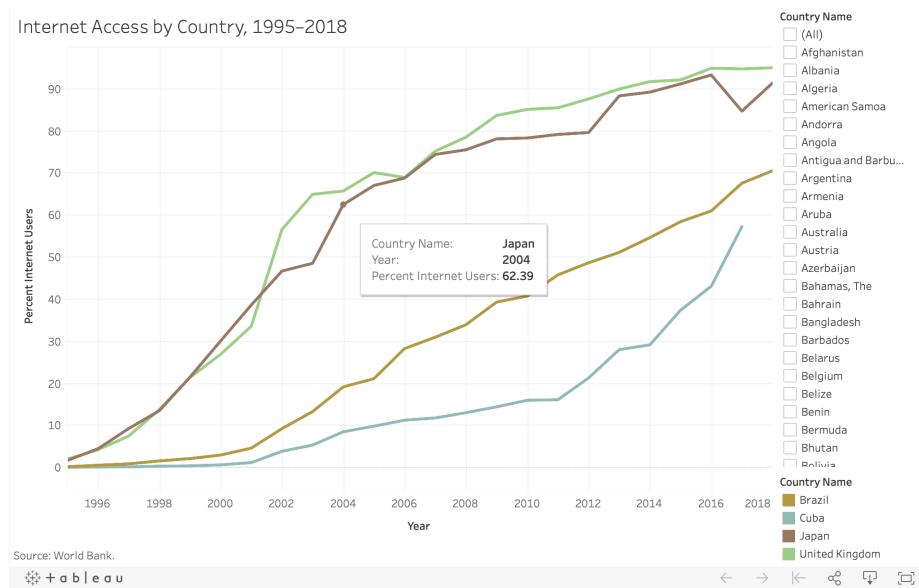


Figure 6.55: Filtered Line chart: Explore the interactive version. Data from World Bank.

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 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 6.56.
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.
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 6.57, 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.

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

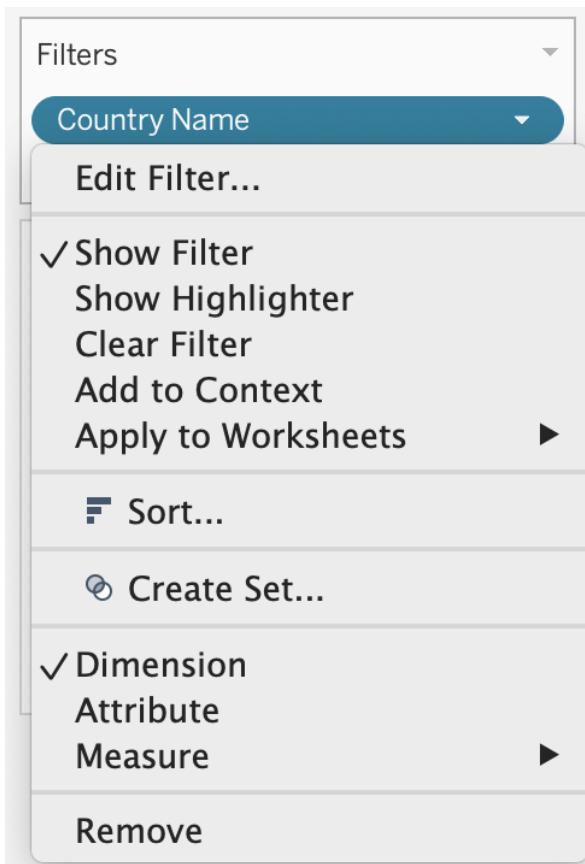


Figure 6.56: After you drag Country Name to the Filters card, make sure the Filter is displayed.

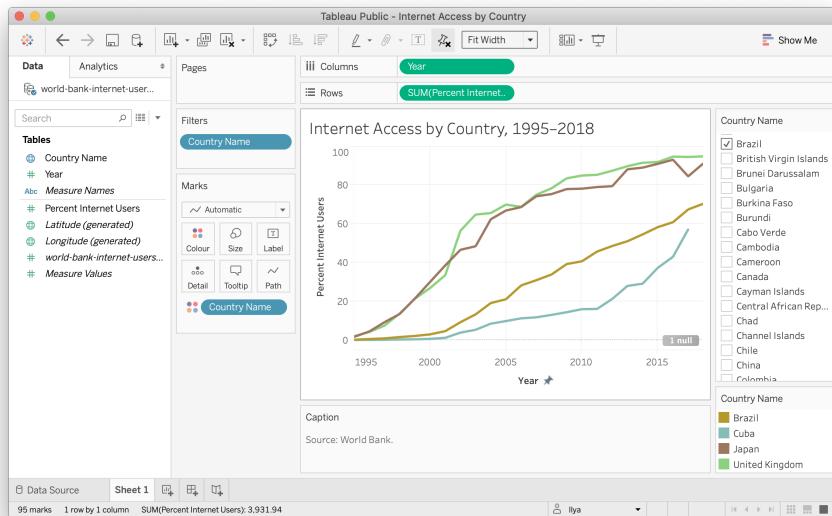


Figure 6.57: This workbook is ready to be published.

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 11: 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 10.

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 9.

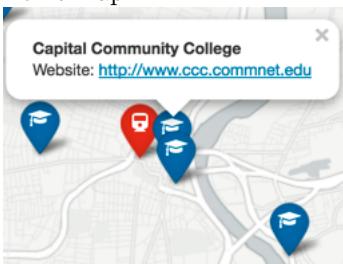
Chapter 7

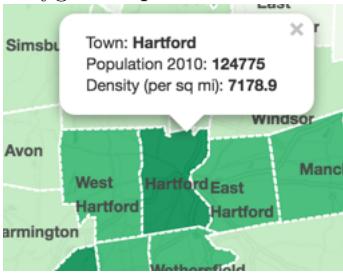
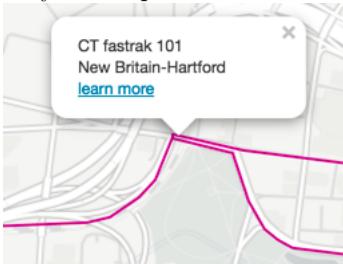
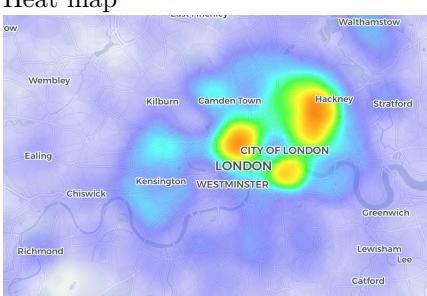
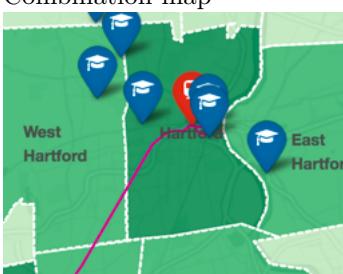
Map Your Data

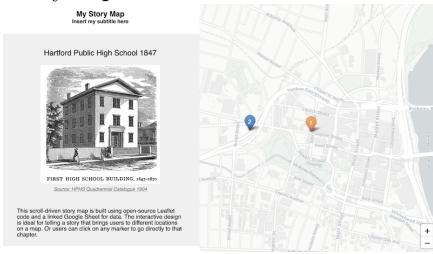
Maps entice readers to explore your data story and develop a stronger sense of place. But good maps require careful thought about how to clearly communicate spatial concepts with your audience.

We will begin this chapter by introducing key principles and definitions related to maps. We will then practice making interactive maps using free online tools, including Google My Maps, Socrata, Tableau, and Datawrapper. We will build two point maps and two choropleth maps. By the end of this chapter, you will be able to use four powerful tools to map your data.

Table 7.1: Basic Map Types and Tutorials

Map	Best use and tutorials in this book
Point map 	Best to show specific locations, such as addresses with geocoded coordinates, with colors for different categories. Easy tool: Google My Maps tutorialPower tool: Leaflet Maps with Google Sheets and other Leaflet templates

Map	Best use and tutorials in this book
Polygon map 	Best to show regions (such as nations or neighborhoods), with colors or shading to represent data values. Also known as choropleth map. Easy tool: n/a Power tools: Tableau Public or Leaflet Maps with Google Sheets and other Leaflet templates
Polyline map 	Best to show routes (such as trails or transit), with colors for different categories. Easy tool: n/a Power tool: Leaflet Maps with Google Sheets and other Leaflet templates
Heat map 	Used to show hotspots or clusters of events, such as high-crime areas. Easy tool: n/a Power tool: Leaflet Heatmap template
Combination map 	Best to show any combination of points, polygons, or polylines. Easy tool: n/a Power tool: Leaflet Maps with Google Sheets and other Leaflet templates

Map	Best use and tutorials in this book
Storymap 	Best for guided point-by-point journey through a historical narrative, with optional photos, audio, or video on an interactive map. Power tool: Leaflet Storymaps with Google Sheets

TODO ABOVE:

- UPDATE table to match chapter contents.
- Decide about removing Storymap “easy tools” because neither allow you to export data you enter: Knight Lab’s StoryMap, ESRI Story Maps
- Add tab-view map for historical change template?
- Add synchronized side-by-side map template?

Map Design Principles

Most of the data collected today comes with some sort of geospatial component. People’s home and places of business have addresses associated with them, such as **1012 Broad St, Hartford, CT**. When we track our runs or bike journeys on Strava, they come with latitude and longitude components. National statistical agencies across the world collect data about regions and territories, such as average income or population counts, making it possible to compare countries and other geographical entities.

However, just because data *can* be mapped does not mean it *should* be mapped. Before you decide to create a map, ask yourself: Does location really matter to your story? If precise values are more important to your story than spatial patterns, consider using a simple table to show values. Most people are familiar with the table, and can easily retrieve information from it as long as you arrange it logically (for example, alphabetically or sorted by value). If you want to show change over time for various geographies, consider using a line chart instead. Sometimes even a simple bar chart can be a much better alternative to a map.

An effective map should show interesting geospatial patterns and should be easy to read in both black-and-white (as is often the case with printed materials) and color.

Understand the Vocabulary

Take a look at Figure 7.1 to get familiar with main basic elements of an interactive map. Similar to a chart, a good maps should have a title and a description that gives a bit of context about what the map is showing.

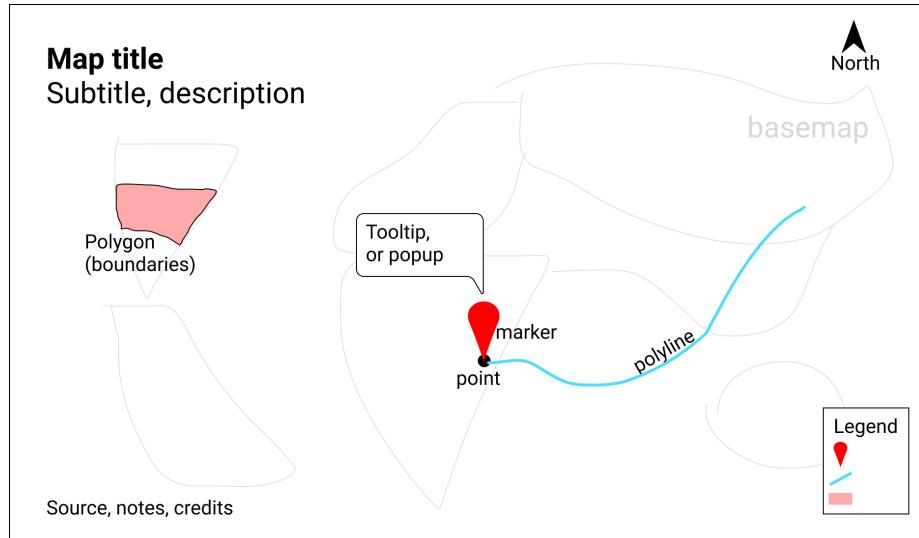


Figure 7.1: Map elements.

The data in the map is presented as layers. A base layer is often satellite imagery of the earth or vector representations of buildings and streets (also known as vector tiles). A base layer provides the foundation of the map. The data displayed on the maps can be generally described as points (such as marker locations of nearby restaurants), polylines (connected points, such as roads or trails), and polygons (polylines where the final point is connected with the initial one). Polygons represent areas, such as building footprints or country boundaries.

The legend provides the mapping between shapes and colors and the values they represent. For example, you may wish to use blue markers to represent restaurants, and orange markers to represent bars, and legend will be the right place to explain that difference.

Interactive maps often “hide” data inside popups or tooltips – boxes with information that appear when you click or hover over map elements.

Interactive web maps often have zoom controls (+ and - buttons) to allow users to inspect data from various “distances”.

Color Palettes: Sequential, Diverging, and Qualitative

If you build a choropleth map, the choice of colors is very important as color is the main way to represent values. So let's talk about color palettes.

Color palettes can be grouped into sequential, diverging, and qualitative. The examples are shown in Figure 7.2.

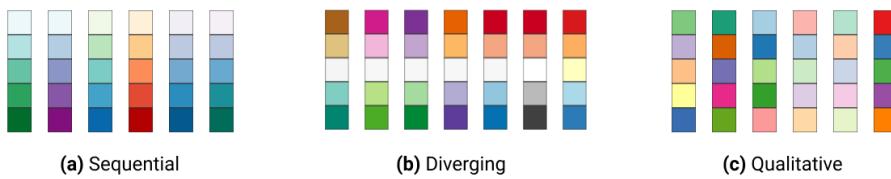


Figure 7.2: Examples of sequential, diverging, and qualitative color schemes from ColorBrewer.

Sequential palettes are used to represent continuous numeric values, in other words, anything that can be placed on a scale. For example, average income, population counts, percent unemployed, etc. Usually such palettes are single-hue (for example, different shades of blue), and typically darker colors represent higher values (but not always).

Note: See Normalize Data to Create Meaningful Choropleth Maps section to see an example of sequential palette use, and learn why representing normalized instead of absolute values may be a better idea.

Diverging palettes also represent continuous values. Unlike sequential palettes, however, diverging colors represent “direction” from some reference value, such as below or above zero, or below or above the average value. Diverging palettes typically have two distinct hues for “positive” and “negative” values, with a neutral color in the middle.

Qualitative palettes typically consist of distinct hues that represent distinct classes. For example, the US Department of State issues travel advisory to foreign destinations ranging from “exercise normal precautions” to “do not travel” relying on a series of qualitative and quantitative measures, such as the likelihood of terror attacks, political and criminal situation, etc. But classes can also be derived from numerical values, as is the case with the World Bank’s classification of countries by income. The organization groups countries and territories into “low”, “lower-middle”, “higher-middle”, and “high” income categories based on gross national income per capita.

It is important you choose an appropriate classification for your choropleth map. While the nature of certain datasets will make the choice of a color palette obvious, most of the time you will have to actively choose *how* to display your data.

In Figure 7.3a, we presented the same dataset using three different color palettes. The map in Figure 7.3a represents per capita income for the contiguous US states using a sequential color scheme consisting of five shades of blue. The darker the color, the higher the income. You can quickly see that states in the north-east, such as Connecticut, Massachusetts, New Jersey, and Maryland have the highest per capita income.

In Figure 7.3b, we used a divergent color scheme to show whether states have higher or lower per capita incomes than the United States as a whole. We subtracted the US per capita income value of \$33,831 from each state's value. This new relative measure is positive for states with higher per capita income, and negative for the states whose per capita income is lower than in the US. In the map, sub-zero values are painted in orange, and above-zero values are in purple. Such color palette could be appropriate for a story about the north-south divide.

You can also split the 48 states into three groups of 16 based on their per capita income, and group them in three thirds, as “low”, “middle”, and “top” third. This is what we did in the map shown in Figure 7.3c.

ColorBrewer

One of the most useful color picking tools for meaningful choropleth maps is ColorBrewer, created by Cynthia Brewer and Mark Harrower. You can see ColorBrewer's interface in Figure 7.4.

ColorBrewer can generate color palettes for a specified number of classes - between three and nine for sequential, three and eleven for diverging, and three and twelve for qualitative. You can also choose palettes that are colorblind safe and print friendly.

Remember that more colors (or “buckets”) does not equal better maps. People are quite bad at distinguishing hues, and an excessive number of buckets will make it harder for reader to compare map values with the legend. If you build a sequential color palette, we recommend you start with *five* buckets, then try four and six and decide what is appropriate for your data *and* your story.

Fewer colors create a *coarse* map with differences in colored ranges becoming more visible. More colors create a more *granular* map, but differences in colored ranges become less visible.

At this point, you should have some understanding of how maps work. So let's move to the first practical exercise of creating a point map with Google My Maps.

Per-Capita Income in the United States, \$

Contiguous US only. Data by 2018 American Community Survey

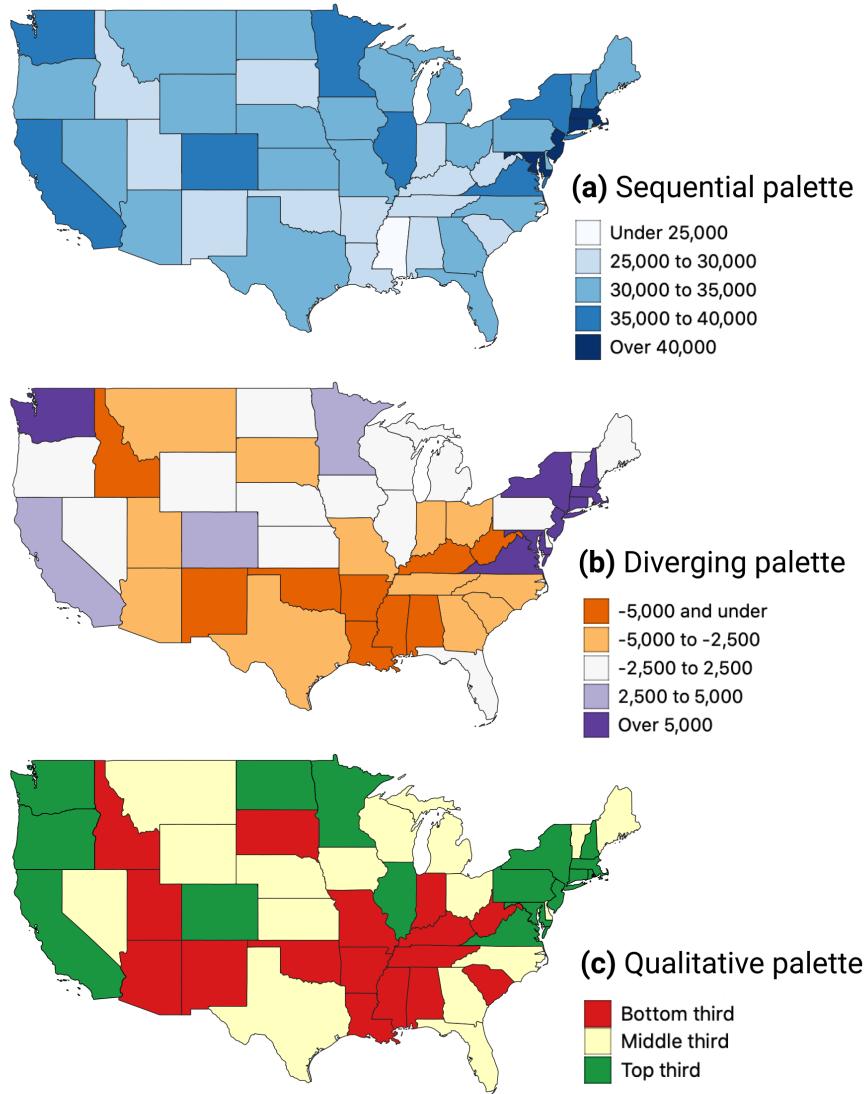


Figure 7.3: Representing per capita income in US states using three different classifications.

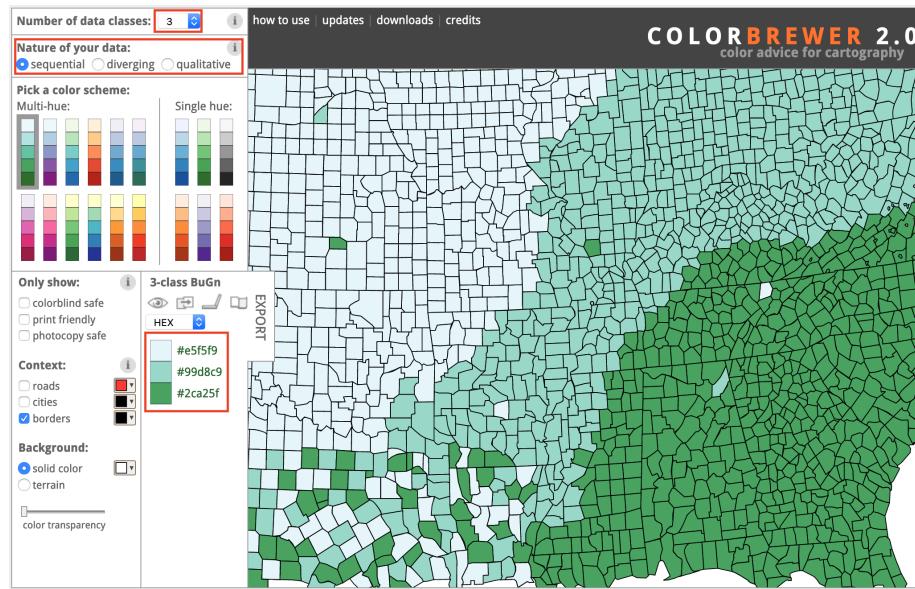


Figure 7.4: ColorBrewer interface.

Point Map with Google My Maps

My Maps is Google's service that allows users to create custom maps using Google Maps platform. It is perhaps the fastest ways of building point and basic polygon maps, although it limits your styling options.

My Maps is most powerful when it comes to collaboration. The platform functions within Google Drive, and so allows you to invite other users with Google accounts to work on the map.

In this section, we will look at building a point map of airports in Nigeria, as is shown in Figure 7.5. We will create a map, change a baselayer, import point data, style points, and share the map.

Create a New Map in My Maps

Navigate to Google My Maps. In the upper-right corner, click `+ Create a New Map` button, as shown in Figure 7.6.

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 7.7 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

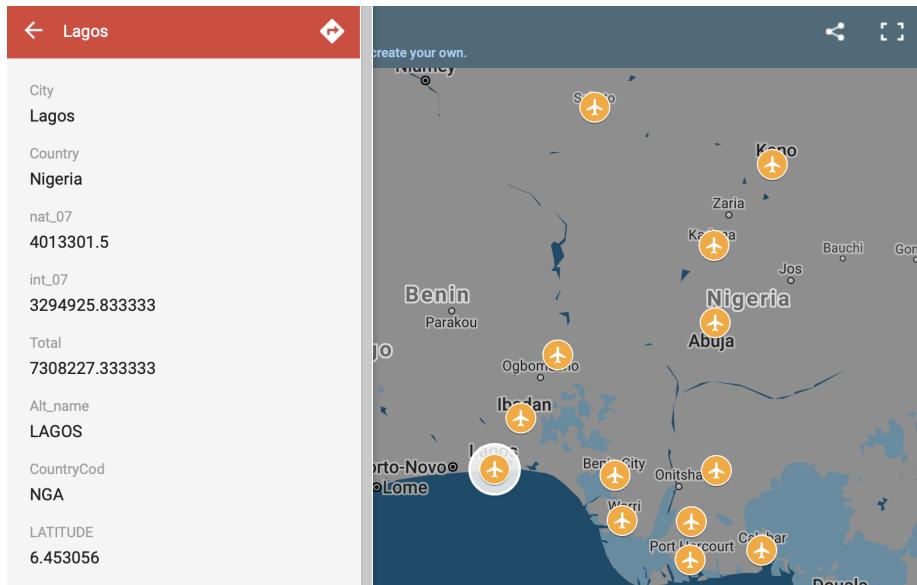


Figure 7.5: A map of airports in Nigeria built using Google My Maps.

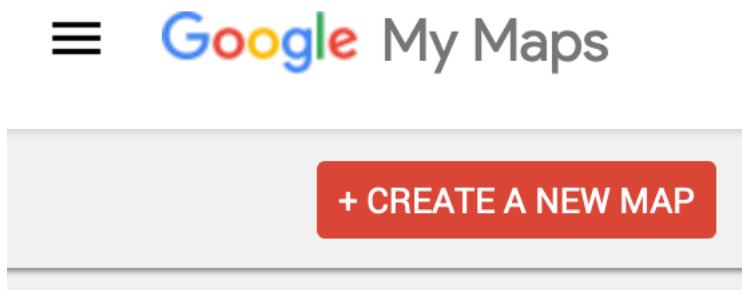


Figure 7.6: A map of airports in Nigeria built using Google My Maps.

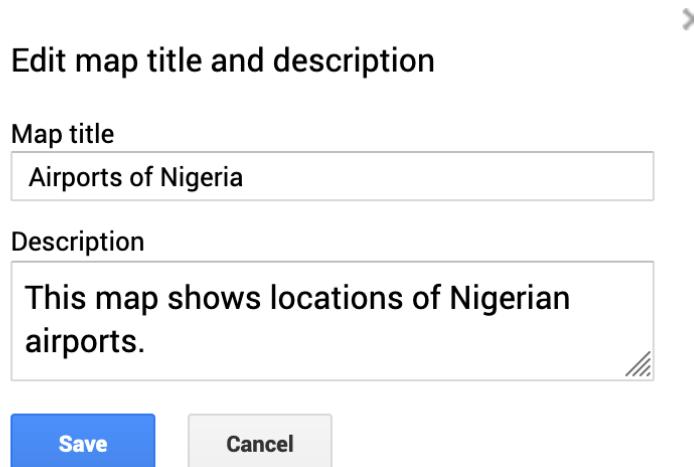


Figure 7.7: Add title and description to your map.

of nine available basemaps. For this tutorial, we chose *Dark Landmass*.

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 7.8.

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 7.9, 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 layers menu will change to an orange airplane, as shown in Figure 7.10.

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*

Choose columns to position your placemarks

Select the columns from your file that tell us where to put placemarks on the map, such as addresses or latitude-longitude pairs. All columns will be imported.

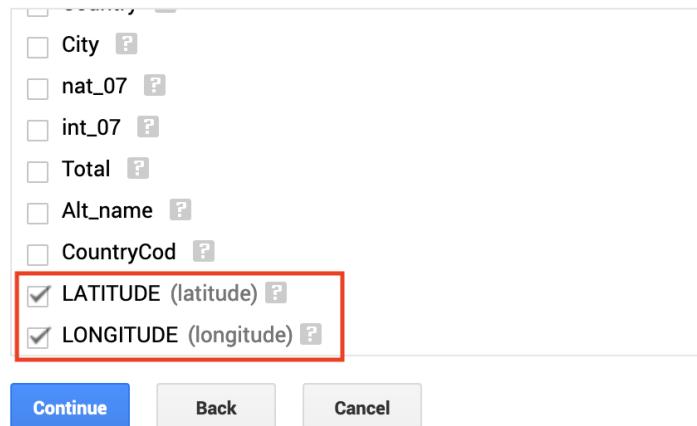


Figure 7.8: Check LATITUDE and LONGITUDE as your location columns.

Choose a column to title your markers

Pick a column to use as the title for the placemarks, such as the name of the location or person.

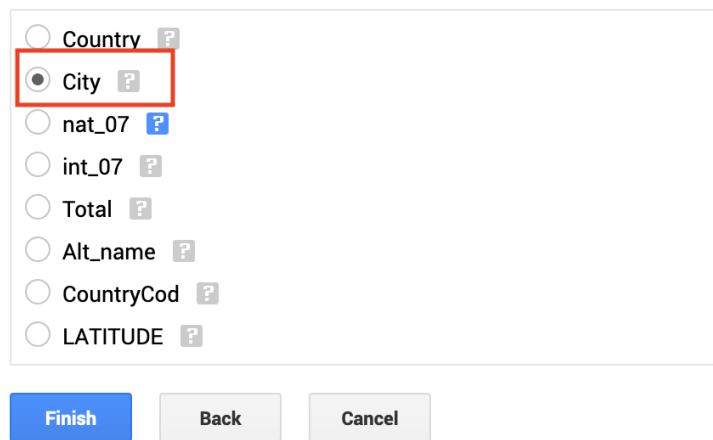


Figure 7.9: Choose City as the title for your markers.

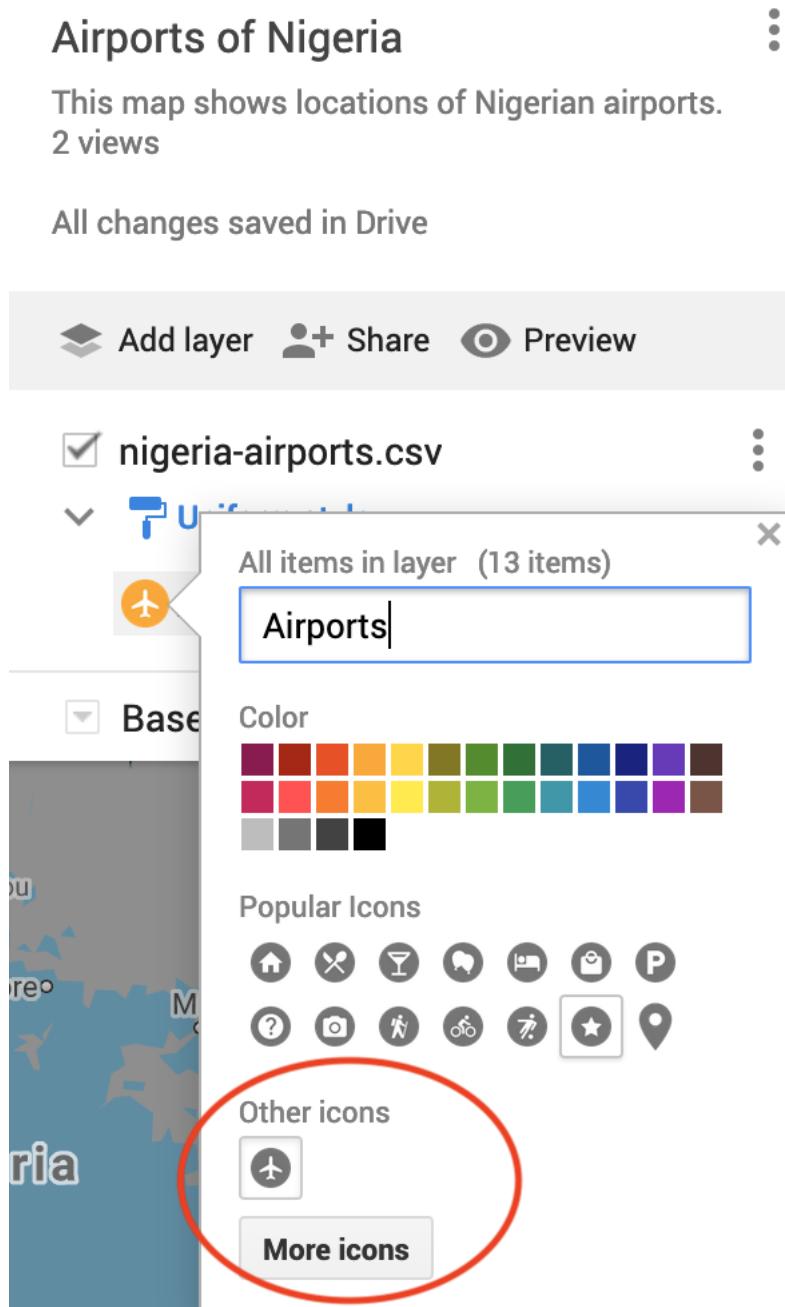


Figure 7.10: In My Maps, you can change marker colors and icons.

layer.

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 to the right of the markers.

Click *Preview* to see how the map looks like outside of the My Maps editing studio.

Share Your Google Map

If you are happy with the result, click *Share*, and click *Change to anyone with the link* (see Figure 7.11), just as you would with any other Google Drive document.

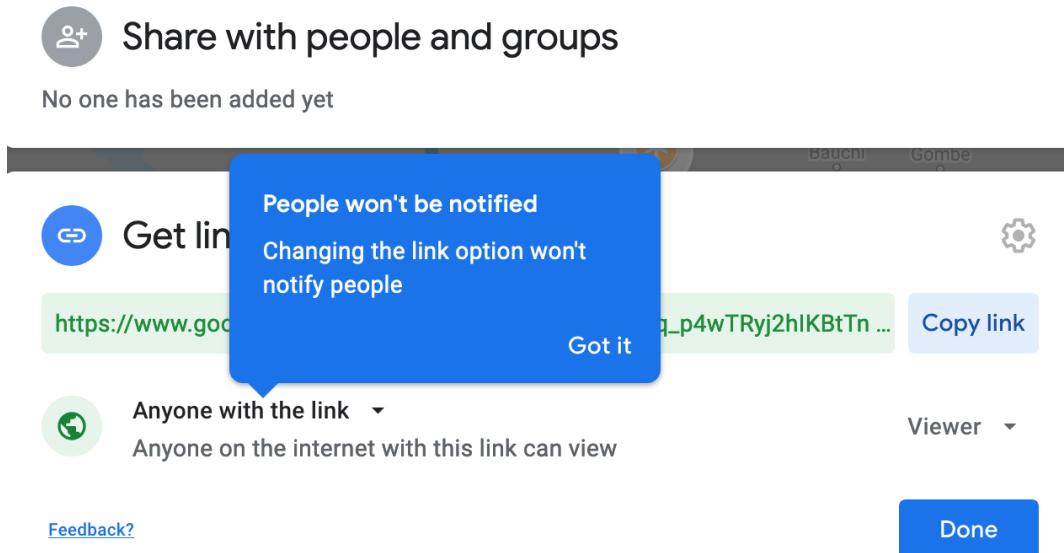


Figure 7.11: Make sure anyone with the link can view your map before you share it.

You can now generate a code snippet to embed the map as an iframe. From the main kebab menu to the right of the map title, choose *Embed on my site* (Figure 7.12). You can use now use this iframe code to embed your map to your Wordpress, Squarespace, or any other 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

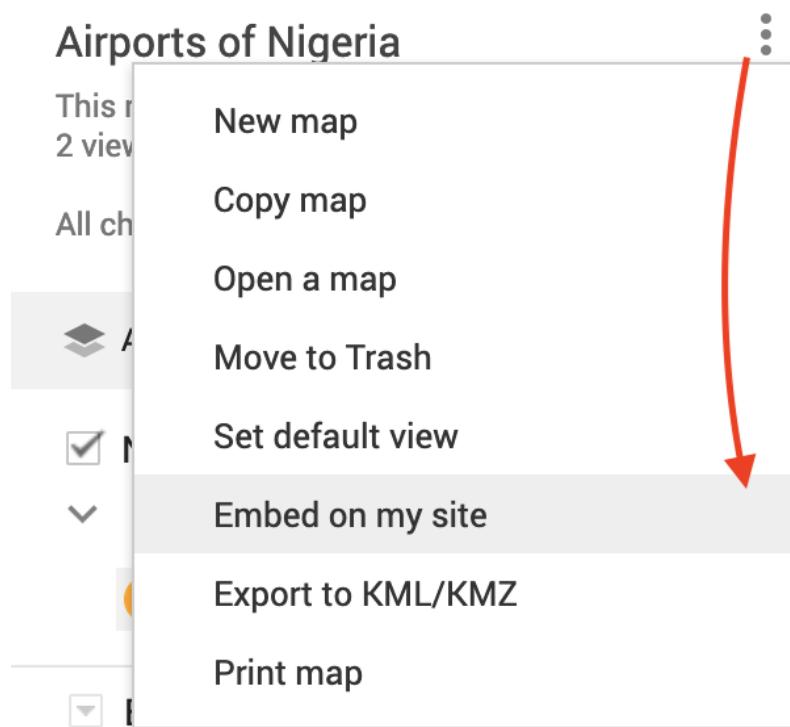


Figure 7.12: My Maps can generate an iframe code to include the map on your own website.

(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.

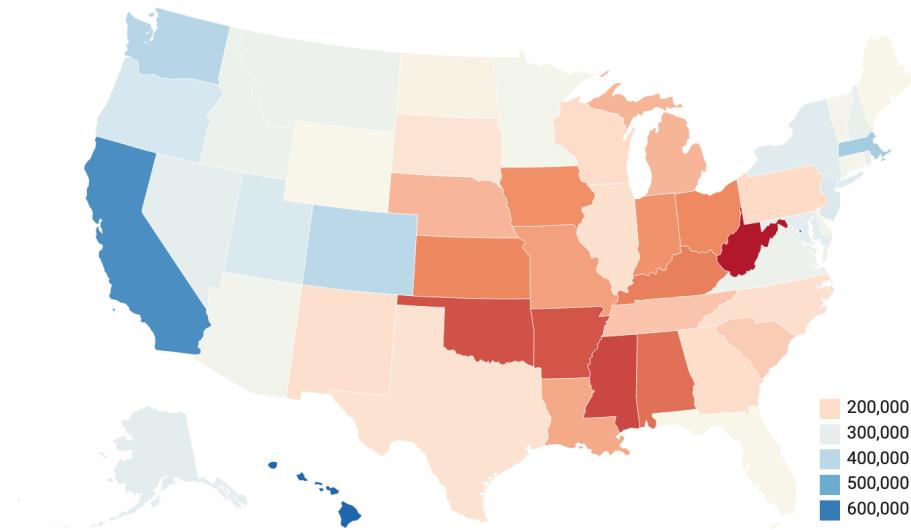
Choropleth Map with Datawrapper

In addition to creating wonderful interactive charts, Datawrapper lets you create stylish and interactive point and polygon maps. In this section, we will create a choropleth map of average home values in US states in 2019 according to Zillow data, as shown in Figure 7.13.

We calculated average 2019 prices for 50 US states and DC, and put them in a spreadsheet, which you can download for this tutorial.

2019 Home Prices in the United States

Homes in Hawaii are more expensive than in DC or California (second and third in nation)



Values are the average of 12 months in 2019

Map: HandsOnDataViz • Source: [Zillow Data](#) • Get the data • Created with [Datawrapper](#)

Figure 7.13: This choropleth map is created in Datawrapper

Create a New Choropleth Map

Sign in to your Datawrapper account and click *New Map* in the header. Datawrapper will offer a choice of a *Choropleth*, *Symbol*, and *Locator* maps, as shown in Figure 7.14. Go ahead and choose *choropleth*.

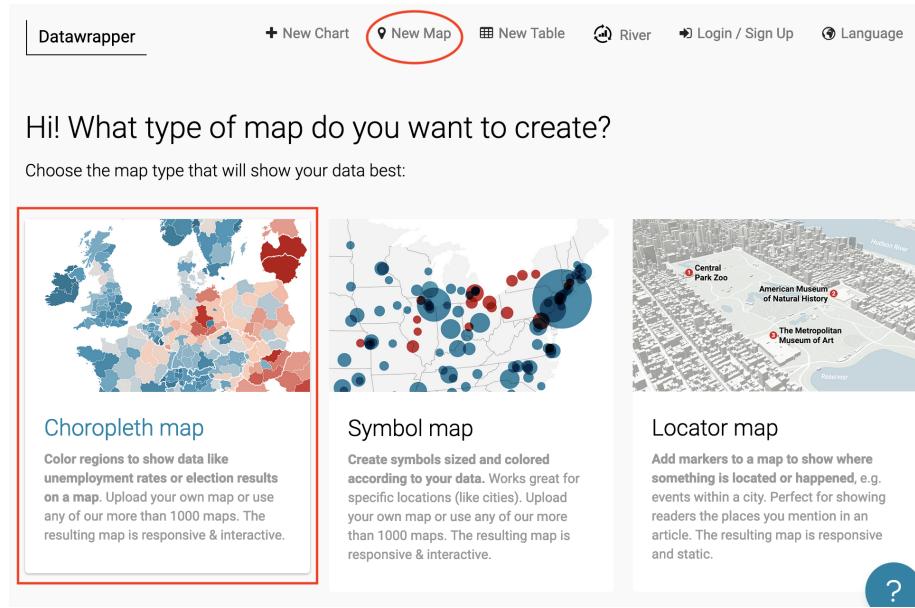


Figure 7.14: Sign in to Datawrapper, click *New Map*, and choose *Choropleth*.

Datawrapper splits the process of creating a map into four steps. In step one, you need to choose your map boundaries. Datawrapper has a wide collection of most common geographical boundaries, including states and counties and municipalities for most countries in the world, and zip code and census tract areas for the United States. But you are not limited to the boundaries that Datawrapper already has. You can also upload your own custom geographies as a TopoJSON or GeoJSON file, and you will learn more about that in the Convert to GeoJSON section of Chapter 13. Because we are mapping average home prices by US state, choose *United States > States*, as shown in Figure 7.15, and hit *Next* to go to step 2.

In the second step, you can attach data to the chosen boundaries. You can set values manually in the interactive table that Datawrapper offers. This is fine for several values, but is too time-consuming for all 50 values (District of Columbia is not a US state, so is not present in the boundaries). Instead, click *Import your dataset* button under the table.

Datawrapper will warn you that you need either *Names* (full state names, such as *Connecticut*), or *ISO-Codes* (the standardized two-letter codes for the state,

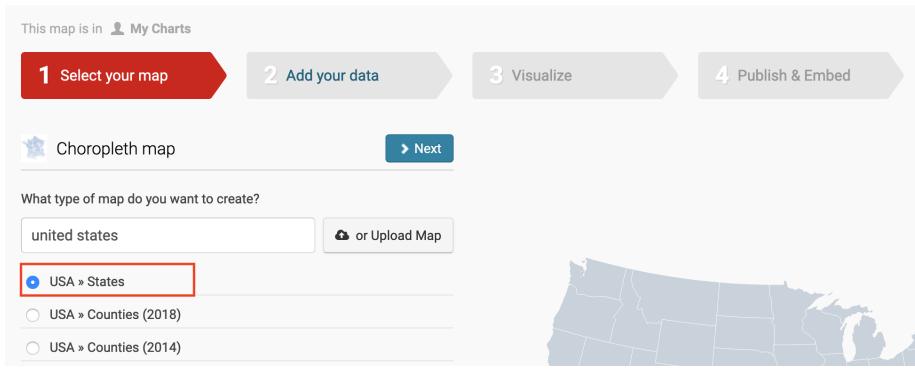


Figure 7.15: To map home prices by US state, choose appropriate boundaries.

such as CT for Connecticut) columns in order to perform merging. Since the dataset that we prepared contains both columns (titled *State* and *StateAbbr*), you can confidently press the *Start import* button.

In the following window, you can either copy/paste values from the spreadsheet, or upload a CSV file. We recommend uploading a file, so click the link and choose the file `us-states-home-values-zillow-2019.csv`. The table will then get populated, and Datawrapper will ask for help identifying the relevant column with geographical names. Make sure your *Matched as ISO-Codes* tooltip is blue above the *StateAbbr* column, scroll down and click *Next*, as shown in Figure 7.16.

Match your columns

Please select which column in your dataset contains "**ISO-Codes**".

MATCH AS ISO-CODES	MATCHED AS ISO-CODES	MATCH AS ISO-CODES
State	StateAbbr	HomeValue2019
California	CA	559861
Texas	TX	207201
New York	NY	325115
Florida	FL	244993

Figure 7.16: Tell Datawrapper which column contains geography names (ISO-Codes of states).

In a similar fashion, you will then be asked to identify a column that contains *values* to be mapped. Make sure it is *HomeValue2019* that is *Matched As Values*.

Note: See Normalize Data to Create Meaningful Choropleth Maps section to learn why you should avoid displaying absolute values or counts in choropleth maps.

Style and publish your map

When finished uploading data, go to step 3 to begin visualizing. Start with the *Refine* tab and choose a diverging color palette, from reds to blues. Make sure your “middle” color value corresponds to the state with the median home price by choosing *min/median/max* value from the Stops dropdown, as shown in Figure 7.17.

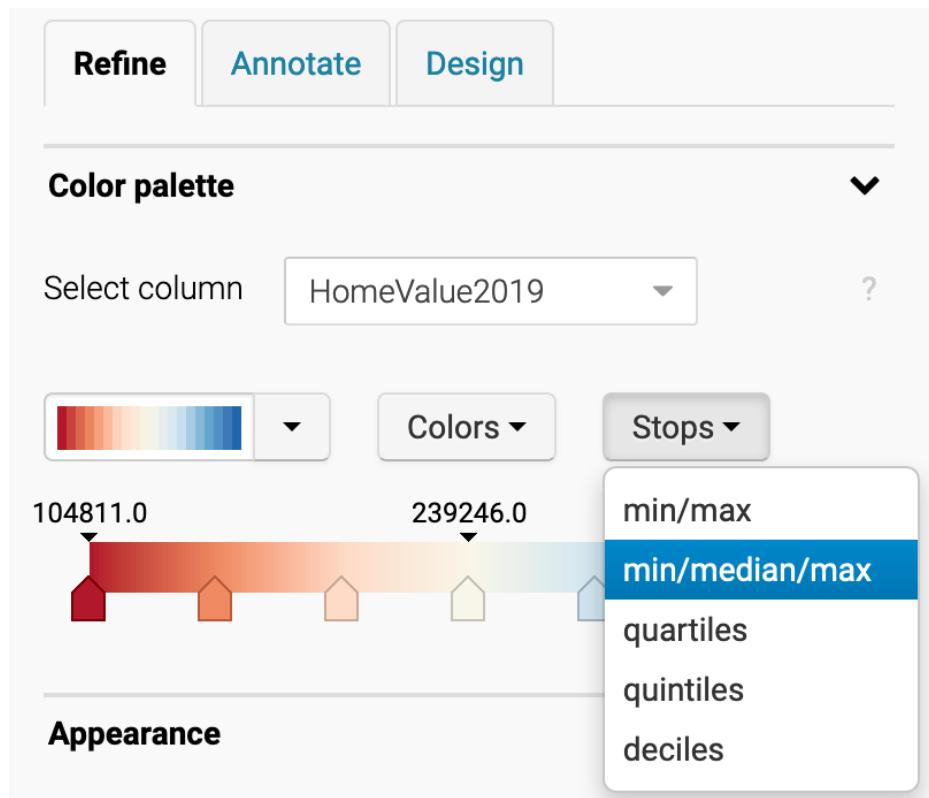


Figure 7.17: Choose red/blues divergent color scheme, and make sure to match median with the neutral (middle) value.

When finished with colors, scroll down to **Tooltips** section and click *Customize tooltips* button. In a popup window, set Title to the state name and Body to the

average home value, as shown in Figure 7.18. The easiest way to do this is to click on column names in the upper-right corner of the popup window, where all available names are listed, which is circled in Figure 7.18. Note that references to variables are put in double curly brackets (`{{ ColumnNameGoesHere }}`). Make sure your dollar symbol (\$) is outside of the curly brackets. Click *Save*, and hover over the map to make sure your tooltip displays the state name and the average home price.

Customize tooltip HTML

Title

`{{ State }}`

Body

`${{ HomeValue2019 }}`

Click on a column name above to add it to your tooltip. Tooltips work with placeholders. You can use them to show additional information when your users move their mouse over the map. The idea is:

```
 {{ Country }} -  
Unemployment Rate: {{ Data }}  
}}
```

Save

Figure 7.18: To reference values from the spreadsheet, add column names in double curly brackets.

You can now move to the *Annotate* tab and set values for the map's title and description, which is the line just below the title. Make sure you reference *Zillow Data* as data source. Being transparent about your sources and methods adds credibility to your work.

In the final, fourth step, you can publish the chart and copy the iframe code that Datawrapper generated for you.

In the following section, we will create a point map inside Socrata data platform.

Filtered Point Map with Socrata

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 7.19.

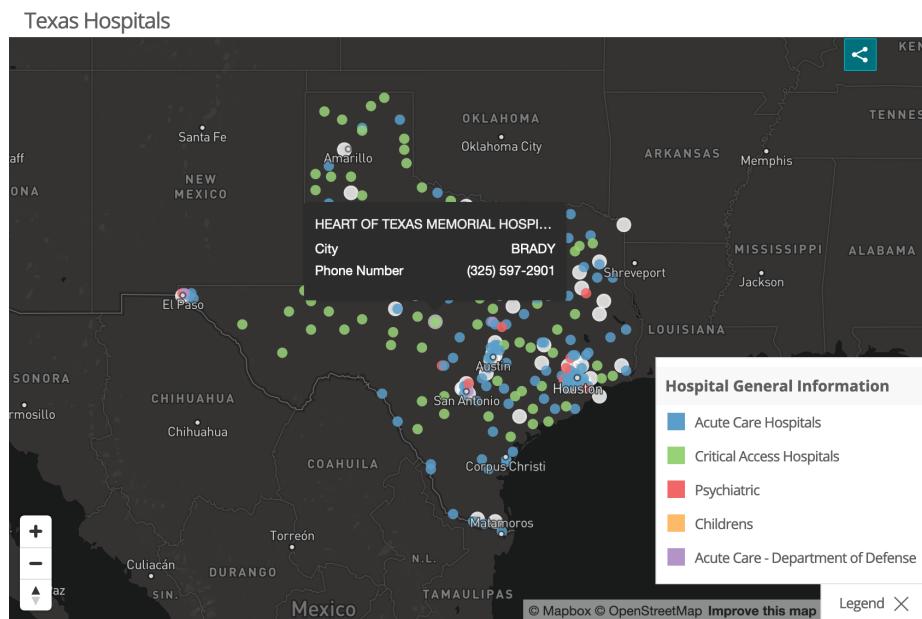


Figure 7.19: In this tutorial, we will build a point map of hospitals in Texas using Socrata.

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.

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, to create a free account.

Note: You can still practice creating a map in Socrata without being logged in. You won't be able to save or share it, however.

Once you have an account, log in using your credentials. 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 7.20. This will open up a *Configure Visualization* studio where you can create the map.

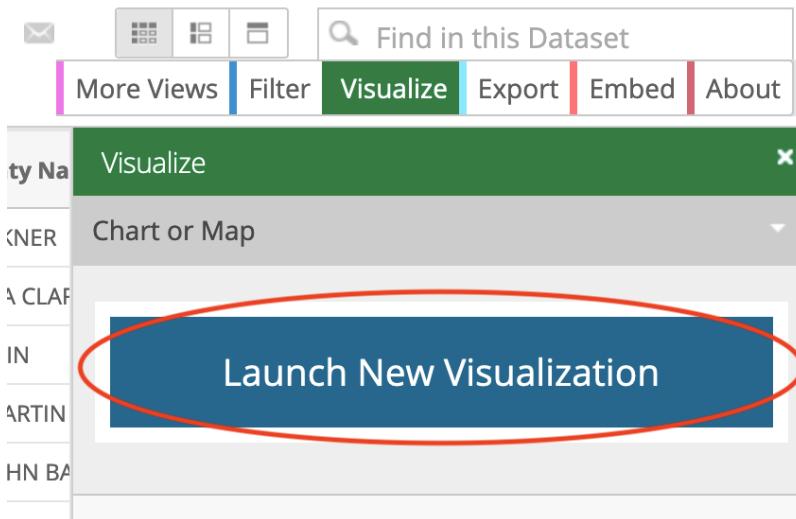


Figure 7.20: Go to Visualize > Launch New Visualization.

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.

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 7.21, and scroll down and click Apply. Socrata should zoom in on the map and center on Texas. Close *Filters* window to free screen space.

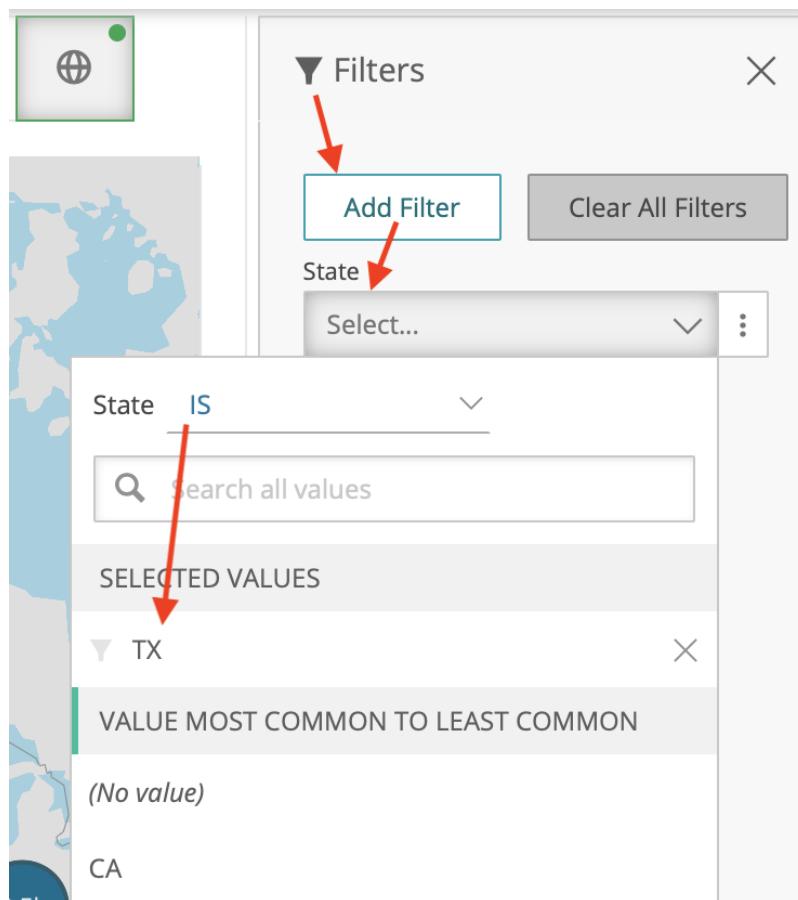


Figure 7.21: Select Texas as the only value for State field.

Let's now **disaggregate** the map so that we can see individual hospitals instead of clusters. Go to *Map Settings > Cluster*, and bring the *Stop Clustering at Zoom Level* slider to 1, as shown in Figure 7.22. You will see the map now

shows individual points.

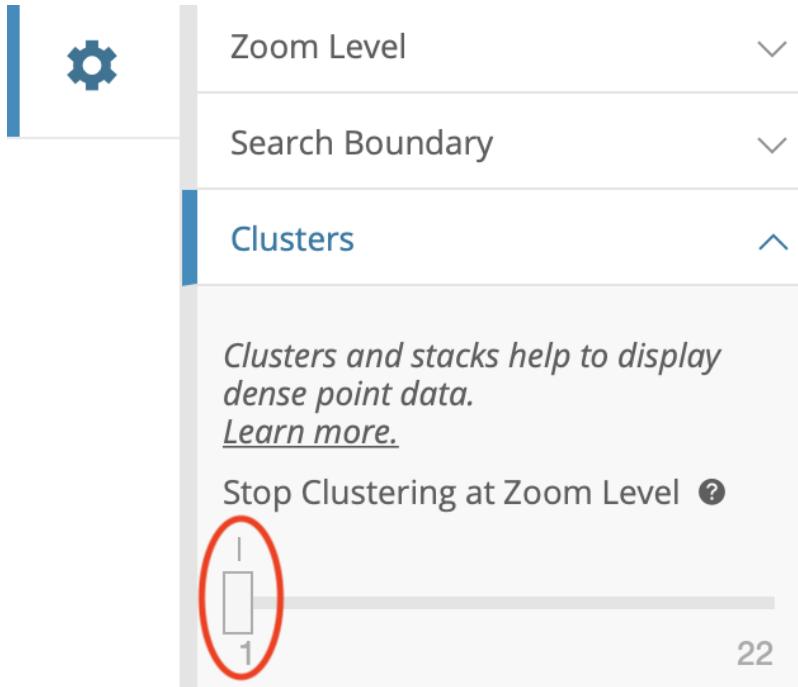


Figure 7.22: To show individual points instead of clusters, set Stop Clustering at Zoom Level to 1.

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 **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 7.23.

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*,

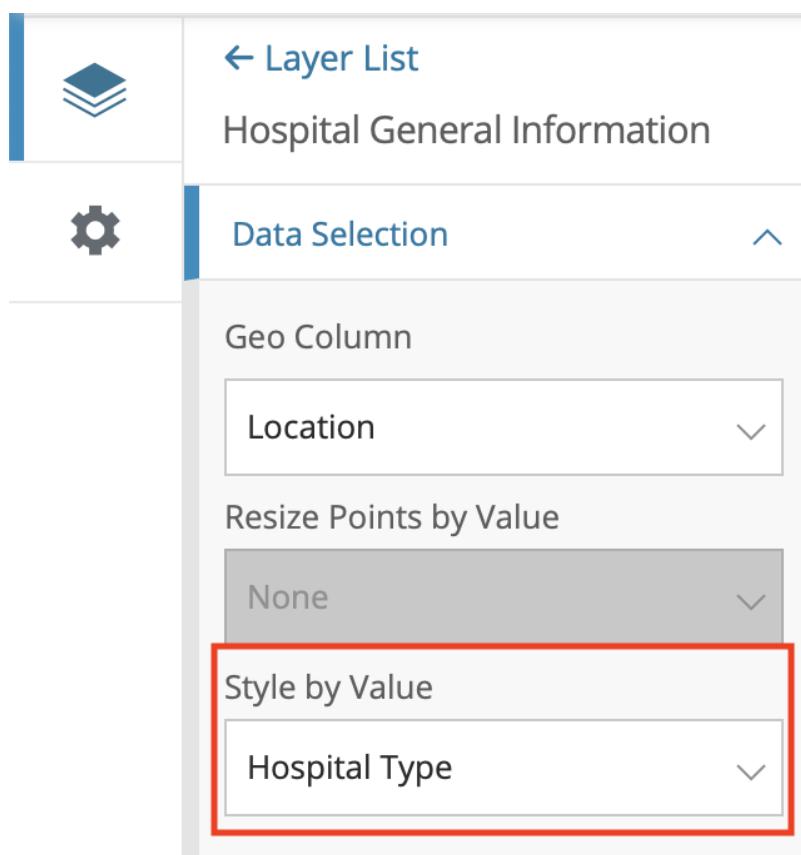


Figure 7.23: Let's display different types of hospitals in different colors.

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 7.24.

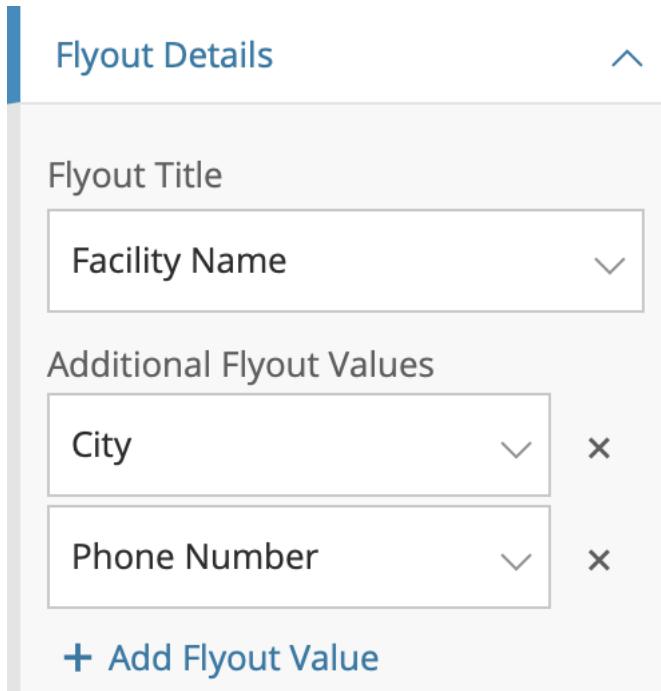


Figure 7.24: To edit tooltip information, use Flyout Details menu item.

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 7.25), and copy the generated code from *Embed Code* text area (Figure 7.26). Learn more in Chapter 9: Embed on Your Web.

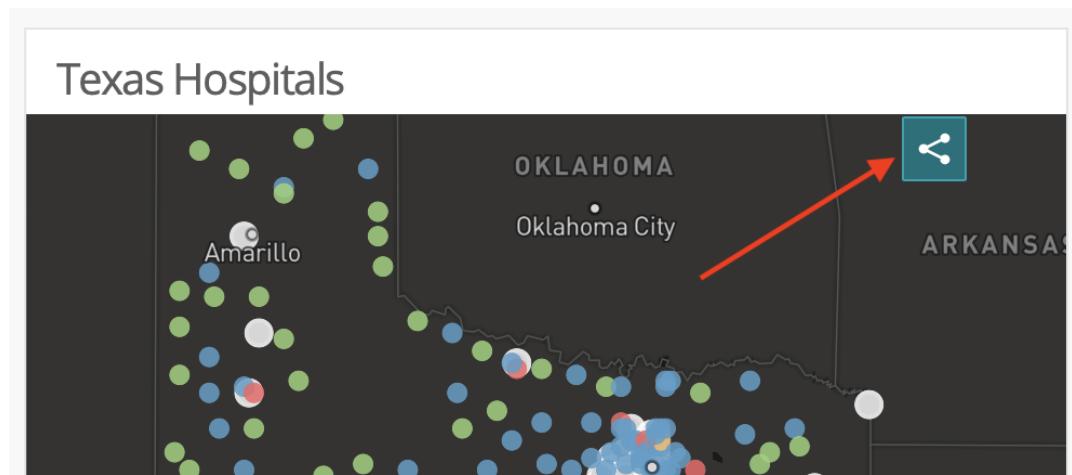


Figure 7.25: Click *Share* button to bring up *Share and Embed* window.



Figure 7.26: 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.

Polygon Map with Tableau Public

In the previous chapter, we looked at using Tableau Public to build scatterplots and filtered line charts. Tableau can also be used to create point and polygon maps.

In this tutorial, we will create a choropleth map of military spending per country as percentage of gross domestic product as shown in Figure 7.27. Remember that choropleth maps work best when they show relative, not absolute numbers. Displaying total spending per country is a bad idea, as bigger countries tend to have larger populations and as a result larger values for a lot of measures, including military spending.

To create maps in Tableau, you do not always need geospatial files. Tableau can recognize a lot of locations, including boundaries for countries and territories, counties within countries, states, US zip codes, airport codes, city locations, and some others. A simple spreadsheet that references these locations can suffice. In this tutorial, we will rely on Tableau to automatically recognize country names. If you require custom boundaries or points (such as town names in your local language), take a look at Create Tableau Maps from Spatial Files article from the official Tableau documentation.

Before we begin, download the 2018 data that we obtained from the World Bank.

Create a Choropleth Map in Tableau

Launch Tableau. In **Connect** section on start up, choose *Text file*, and select `military-spending-2018.csv`. From Data Source tab, inspect the dataset

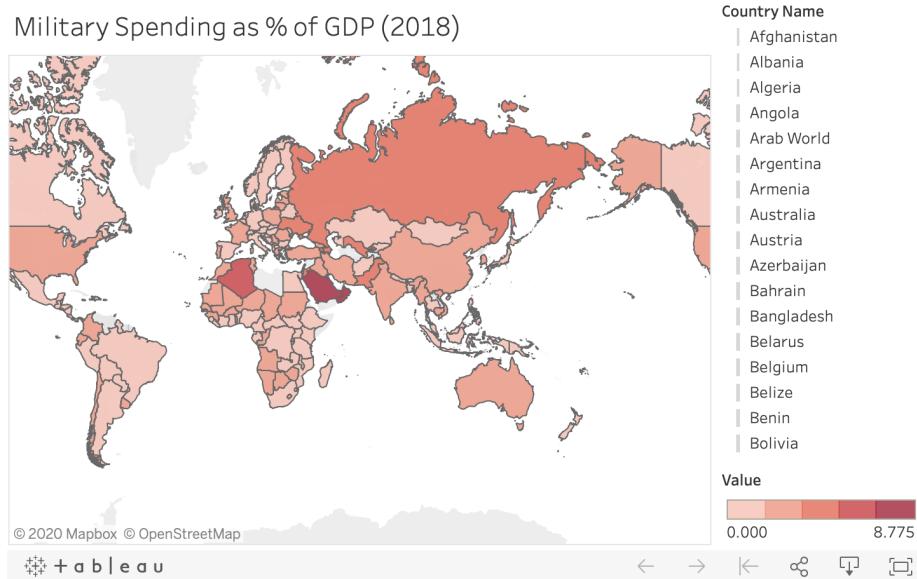


Figure 7.27: Filled Polygon (Choropleth) map: Explore the interactive version. Data from TODO.

contents. It contains four columns: Country Name, which includes countries and territories defined by the World Bank, the three-digit Country Code, Indicator Name (same for all rows), and percent value, as shown in Figure 7.28. Notice that some values are set to *null* (not available).

In the variables list on the left, notice how Tableau generated *Latitude* and *Longitude* fields based on country names and country codes (by the way, you only need one or the other, not both).

Drag and drop *Longitude* to **Columns**, and *Latitude* to **Rows**, as shown in Figure 7.29. You should see that the chart area shows an empty map of the world (if not, double-check that *Latitude* is indeed in Rows, not Columns). In Marks box, change *Automatic* to *Map*, and drag *Country Name* variable to the **Size** box of the Marks card. You will see that country outlines turned blue.

We want colors to represent military spending values, so drag *Value* variable to the **Color** box of the Marks card. You should now see a proper choropleth map.

Places like Greenland and Libya do not have available values, but they are still painted with the lightest color, which is misleading. To remove countries with *null* values from the map, drag *Values* to the *Filters* card. A popup window will ask you how you want to filter, just leave everything unchanged. This will leave the whole range of values, and exclude *null* values (see the checkbox in the lower-right corner of the Filter window in Figure 7.30).

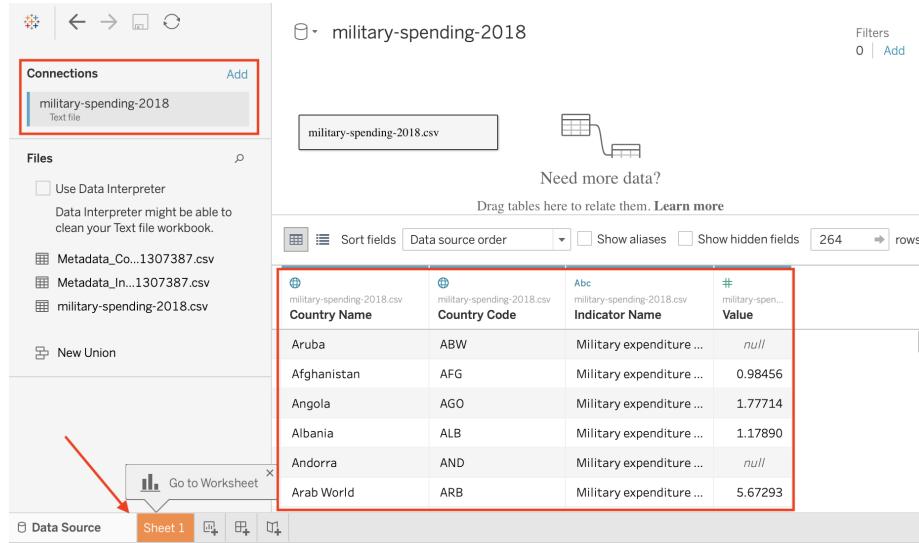


Figure 7.28: When finished inspecting the connected file, go to Sheet 1.

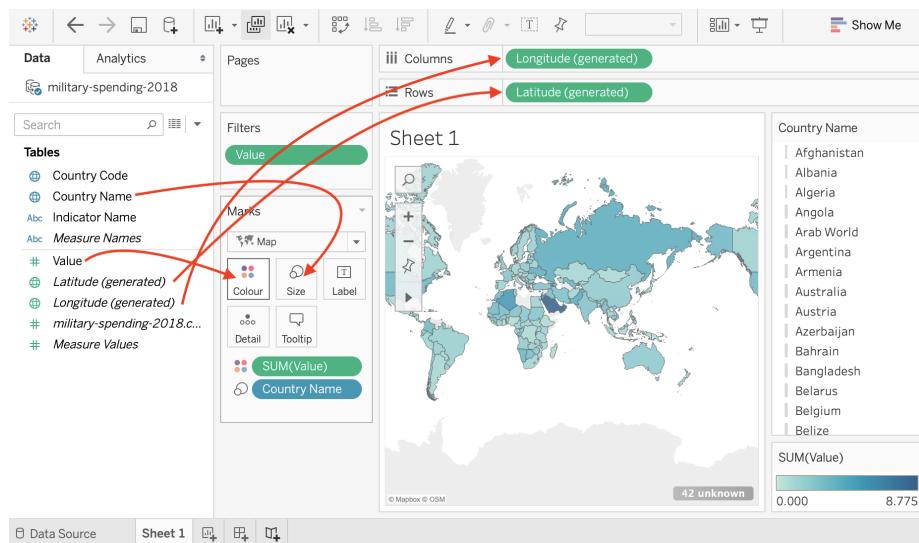


Figure 7.29: Drag and drop variables to the right places.

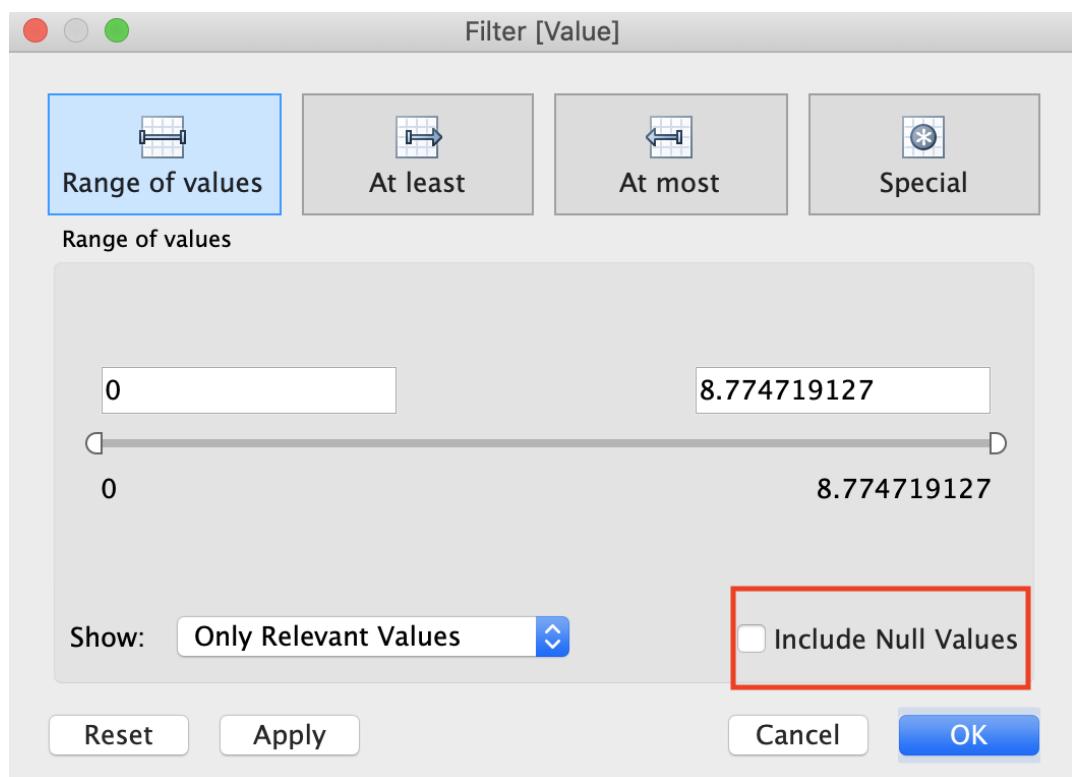


Figure 7.30: Filter values to remove countries with no data from display.

You can change the color scheme by clicking the Color box of the Marks card, and then *Edit colors*. Change the palette to *Reds*, and make it stepped rather than continuous, as shown in Figure 7.31.

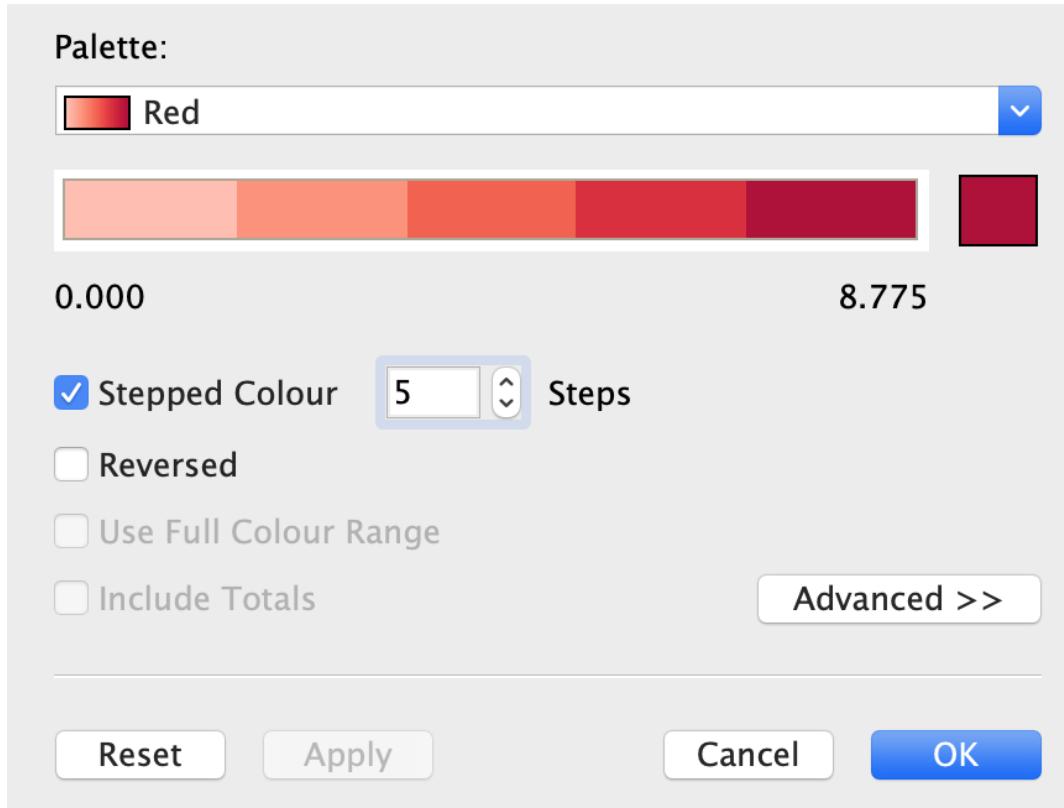


Figure 7.31: Change the color scheme to Reds with 5 steps.

You may notice the tooltip calls values *Value* when hovering over countries. Click the Tooltip box of the Marks card to change text to *Military spending*, and add a percentage sign after the value itself, as shown in Figure 7.32. Make sure not to change values between < and >, as these are references to variables.

And finally, let's add a proper title to the map. Double-click the default *Sheet 1* name to bring up the *Edit Title* window, and change the name of your chart to a more meaningful *Military Spending as % of GDP (2018)*.

Publish Your Tableau Map

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

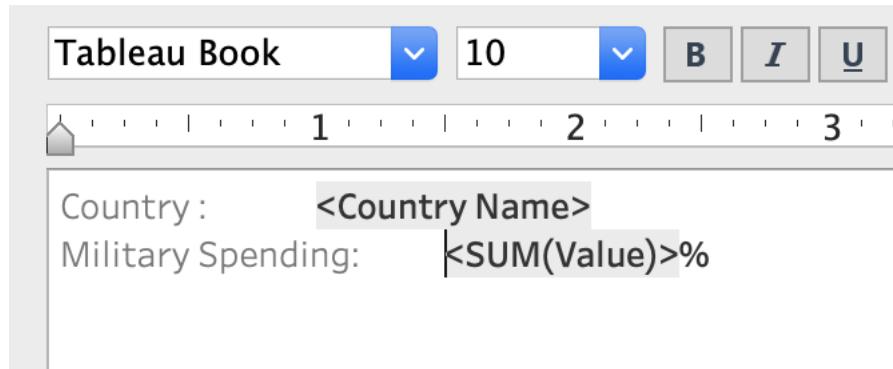


Figure 7.32: Change tooltip text to make it more user-friendly.

title, such as *Military Spending*, and click Save.

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 13, we will talk more about geospatial data, how it can be obtained, stored, modified, and shared.

Chapter 8

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.

In this chapter, you'll learn about table design principles and how to use Datawrapper, a tool we introduced in Chapter 6: Chart Your Data and Chapter 7: Map Your Data, but now use to create an interactive table. Interactive tables have many advantages over static tables when you publish your information online rather than print-only. 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 Datawrapper table. Sparklines blend the best qualities of tables and charts by making it easier for readers to visually scan for patterns while skimming down the rows and columns of your data table. Later in Chapter 9: Embed On Your Web, you'll learn how to integrate your interactive table into your website.

Table Design Principles

Let's begin with some principles of good table design, similar to how we learned about chart design in Chapter 6 and map design in Chapter 7. 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 8.1.

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 15: Tell Your Data Story.

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. Remove any clutter or unnecessary repetition that stands in the way of these goals.

Now that you understand key principles of table design, see how several of them 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 6 and maps in Chapter 7. 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

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
Fruit	Banana	Yellow	
	Apple	Red	
	Blueberries	Blue	
Vegetable	Kale	Green	
	Carrot	Orange	
	Eggplant	Purple	

Figure 8.1: A sample table that illustrates selected design principles.

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."² While Tufte envisioned sparklines on a static sheet of paper or PDF document, you'll create them inside an interactive table, as shown in Figure 8.2. 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

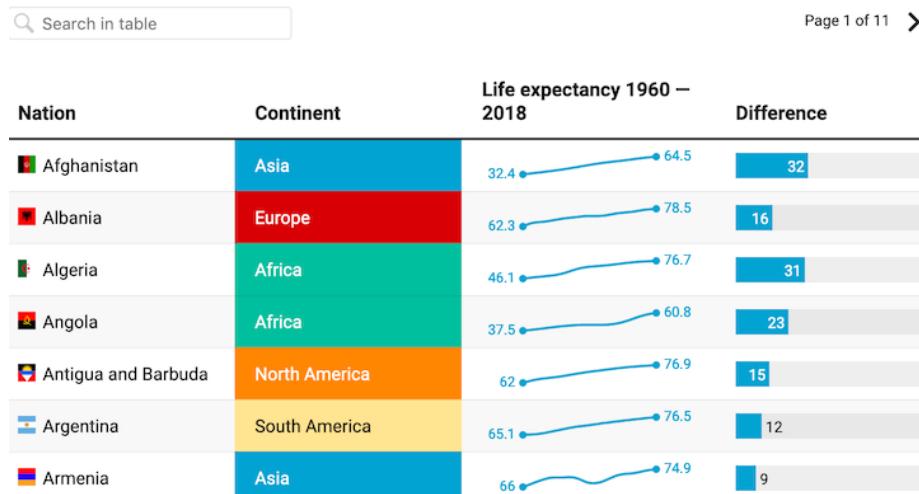


Figure 8.2: Table with sparklines. Explore the interactive version.

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

²Edward R. Tufte, *Beautiful Evidence* (Graphics Press, 2006), <http://books.google.com/books?isbn=0961392177>, pp. 46-63.

Chapter 6, 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 7, 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 and select *New Table*. 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 *Upload Data* screen, select that you wish to import your data from a Google Spreadsheet. Paste in the web address of our cleaned-up Google Sheet and click *Proceed*.
4. Inspect the data in the *Check and Describe* screen. 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 8.3.

The screenshot shows the 'Customize columns' interface in Datawrapper. On the left, a sidebar lists columns: Name, Code, Nation (selected), Continent, Life expectancy 1960, # 1960, and # 1961. Below this is a toolbar with 'Select all', 'none', and 'invert'. On the right, a preview table shows country names, their three-letter codes, and flags. The 'Nation' column is being customized. The 'Replace country codes with flags' checkbox is checked. Other settings visible include 'Show on desktop' and 'mobile', 'Style' (B, I, A, C), 'Border' (none), 'Alignment' (auto), and checkboxes for 'Color cells based on categories', 'Fixed column width', and 'Show as bar chart'. The preview table lists countries from Algeria to Belarus, each with its code and flag.

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 8.3: 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 drop-down 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 8.4.

Continent	Country	Continent
Asia	Algeria	Africa
Asia	Angola	Africa
Asia	Antigua and Barbuda	North America
Asia	Argentina	South America
Asia	Armenia	Asia
Asia	Aruba	South America
Asia	Australia	Oceania
Asia	Austria	Europe
Asia	Azerbaijan	Asia
Asia	Bahamas	North America
Asia	Bahrain	Asia
Asia	Bangladesh	Asia
North America	Barbados	North America
Europe	Belarus	Europe
Europe	Belgium	Europe

Figure 8.4: Customize the *Continent* column and push slider to color cells based on categories.

- 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 8.5. Then click *Proceed* to return to the *Visualize* window.

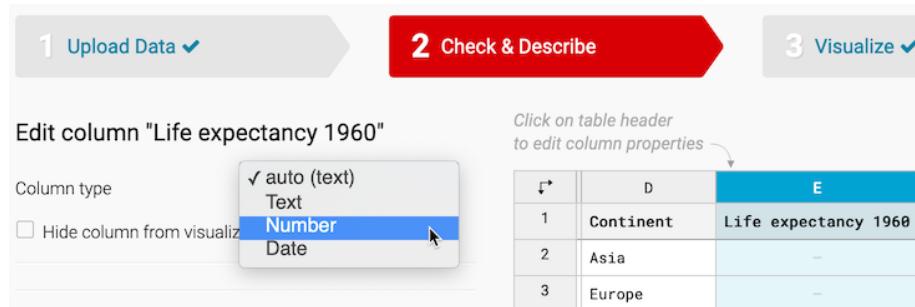


Figure 8.5: Go back to *Check & Describe* to change the properties of column E from textual to numerical data.

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 8.6. These steps will create the sparklines in the column, renamed *Life expectancy 1960–2018*, as shown in Figure 8.7.

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 8.7.
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 8.2. Click the blue *Publish chart* button to obtain the embed code to place your visualization on the web, which you'll learn about in Chapter 9: *Embed On Your Web*.

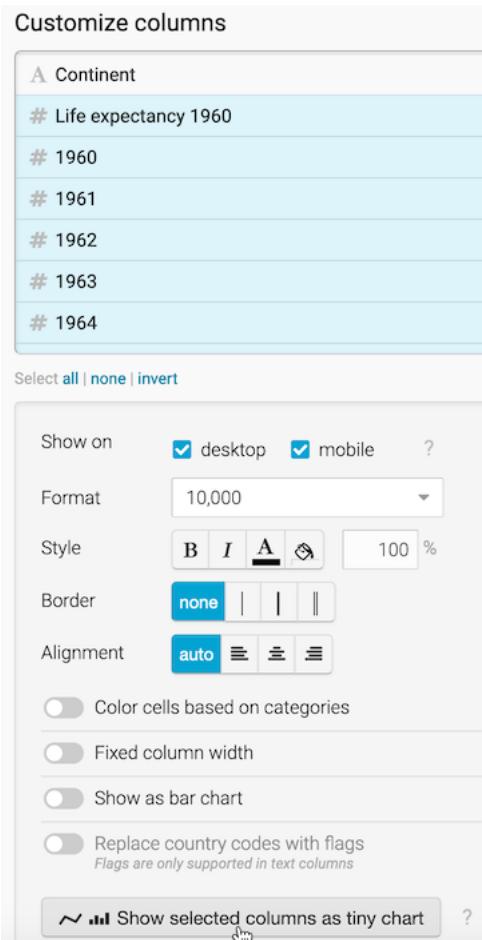


Figure 8.6: Shift-click to select all columns from *Life expectancy 1960–2018* down to *2018*, then click on *Show selected columns as tiny chart*.

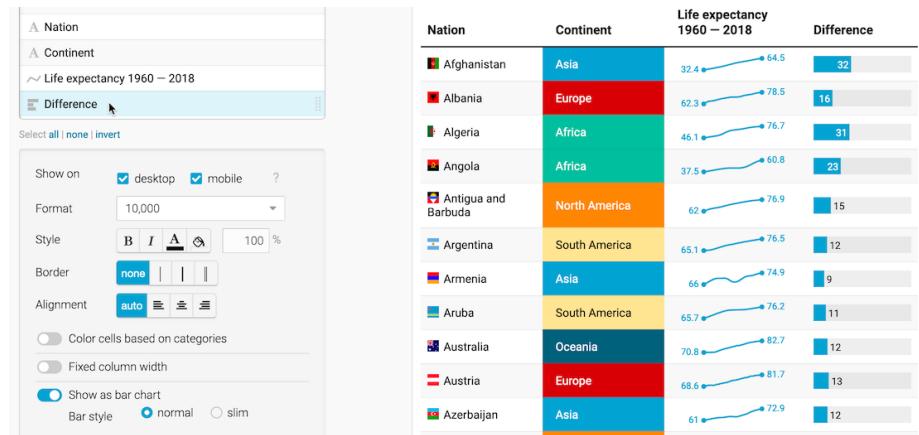


Figure 8.7: Select the *Difference* column and *Show as bar chart*.

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 9

Embed On Your Web

TODO: Reorganize and rewrite chapter

After you create a chart or map, how do display it inside your website as an *interactive* visualization? Our goal is not a static picture, but a live chart or map that users can explore. This is an important question for beginners, since data visualizations are not valuable unless you can control where and how your work appears. This chapter walks you through the key steps.

First, you need to own a website that supports iframe codes (which we'll explain below). If you do not have a website that supports this, then follow this quick tutorial to Create a simple web page with GitHub Pages. Even if you already have a website, still do this tutorial, because it introduces a tool used many times in this book.

Second, you need to copy or create an iframe code from your chart or map. An iframe is one line of HTML code with instructions on how to display a web page from a specific address (called a URL). A simple iframe looks like this:

```
<iframe src="https://handsondataviz.org/embed/index.html"></iframe>
```

No coding skills are necessary. See these easy-to-follow examples:

-Copy iframe from a Google Sheets chart -Convert a link into an iframe

Finally, you need to paste (or embed) the iframe code inside your website. Like a picture frame, an iframe allows you to display one web page (your data visualization) inside another web page (your personal website). But unlike a picture frame, where the image is static, an iframe makes content interactive, so visitors can explore the chart or map on your site, even though it may actually be hosted on an entirely different website. Go to this third tutorial, which combines the two steps above, called Embed Iframe in GitHub Pages.

See more tutorials in this chapter to copy iframes from other visualization tools (such as Tableau Public and embed them in other common websites (such as

WordPress, etc.) ** TO DO: add more tutorials and links **

Create a Simple Web Page with GitHub Pages

Question: After you create an interactive chart or map, how do you embed the live version in a website that you control?

The full answer requires three steps:

- 1) Create a web page that supports iframe codes
- 2) Copy or create an iframe code from your visualization
- 3) Embed (or paste) the iframe code into your web page

This tutorial focuses on the **first step**. If you don't already have your own website, or if you are not sure whether your site supports iframe codes, then follow the steps below. We will create a simple web page with a free and friendly tool called GitHub <http://github.com>, and host it on the public web with the built-in GitHub Pages feature. For **steps 2 and 3**, see the Copy iframe from Google Sheets tutorial and the Embed iframe in GitHub Pages tutorial in this chapter.

Tool review: GitHub <http://github.com> is a versatile tool that can be used to create simple web pages.

- Pros:
 - Free and easy-to-learn tool to edit and host simple pages on the public web.
 - All steps below can be completed in your web browser.
- Cons:
 - All work on GitHub is public by default. Private repositories (folders) require payment.
 - New users sometimes confuse the links for code repositories versus published web pages.

Video

- 1) Sign up for free GitHub account, then sign in, at <http://github.com>.
- 2) Create a new repository (also called a “project” or similar to a “folder”).
- 3) Name your repository (or “repo”), and select Initialize with a README file. Optional steps: add a description and select a license.

- 4) Scroll down and click the green button to Create your repo, which will appear in a new browser tab, with this URL format:

`https://github.com/YOUR-USERNAME/YOUR-REPO-NAME`

- 5) In your GitHub repo, click on Settings, scroll down to GitHub Pages, select Master branch as your source, then Save. This publishes the code from your repo to the public web.

Hint: Do NOT select Theme Chooser for this exercise. It will create additional files that will interfere with displaying an iframe in your README.md file.

- 6) When the Settings page refreshes, scroll back down to GitHub Pages to see the new link to your published website, which will appear in this format:

`https://YOUR-USERNAME.github.io/YOUR-REPO-NAME`

- 7) Right-click and Copy the link to your published web site.
- 8) At the top of the page, click on the repo name to return to the main level.
- 9) Click the README.md file to open it in your browser, and click the pencil symbol to edit it.
- 10) Inside your README.md file, paste the link to your published web site, and type any text you wish to appear. The .md extension refers to Markdown, an easy-to-read computer language that GitHub Pages can process.
- 11) Scroll down and click the green Commit button to save your edits.
- 12) When your GitHub repo page refreshes, click on the new link to go to your published web site. **BE PATIENT!** Your new site may not appear instantly. Refresh the browser every 10 seconds. You may need to wait up to 1 minute for a new site to appear the first time, but later changes will be much faster.

Remember that GitHub Pages is designed to create simple web pages and sites. See other web publishing tools mentioned in this chapter to create more sophisticated web sites.

Copy an iframe code from a Google Sheets interactive chart

Question: After you create an interactive chart or map, how do you embed the live version in a website that you control?

The full answer requires three steps:

1. Create a web page that supports iframe codes
2. Copy the iframe code from your visualization
3. Embed (or paste) the iframe code into your web page

This tutorial focuses on the **second step**, and shows how to publish a Google Sheets interactive chart, and copy its iframe code. Details may differ for other visualization tools, but the general iframe concept will be similar to most cases. For **steps 1 and 3**, see the Create a Simple Web Page with GitHub Pages tutorial and the Embed iframe in GitHub Pages tutorial in this chapter.

Tutorial

- 1) Create a Google Sheets chart, which requires a free Google Drive account.
Learn more in the Google Sheets Charts tutorial in this book.
- 2) Click the drop-down menu in the upper-right corner of the interactive chart and select Publish chart. Click OK on next screen.
- 3) Select the Embed tab, select the Interactive version, and click the blue Publish button. If you make changes to the chart, they will continue to be published to the web automatically, unless you click the Stop button or checkbox at the bottom.
- 4) Copy the iframe embed code.

No coding skills are necessary, but it helps to be code-curious. This iframe is a line of HTML code that contains these instructions:

- iframe tags to mark the beginning and end
- width and height: to display your chart in a second site, in pixels
- seamless frameborder: “0” means no border will appear around the chart in the second site
- scrolling: “no” means the chart will not include its own web scrolling feature

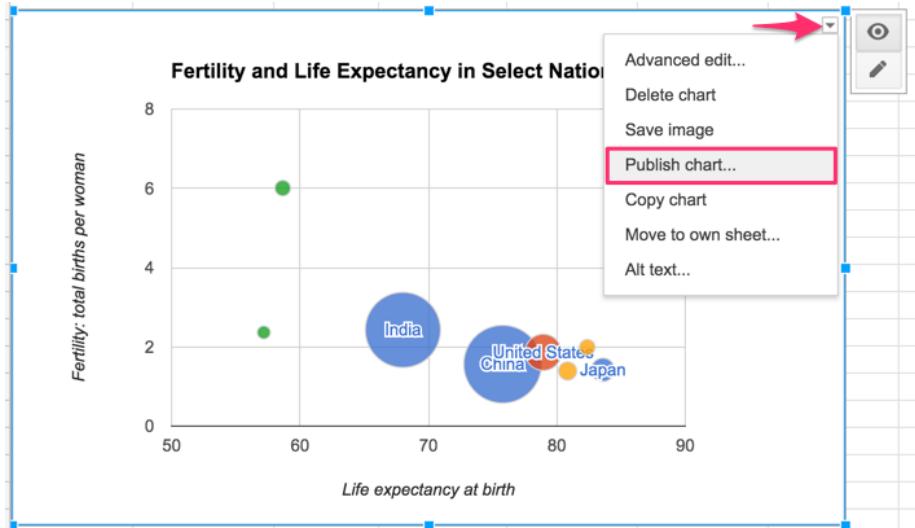


Figure 9.1: Screenshot: Drop-down menu to publish a Google Sheets chart

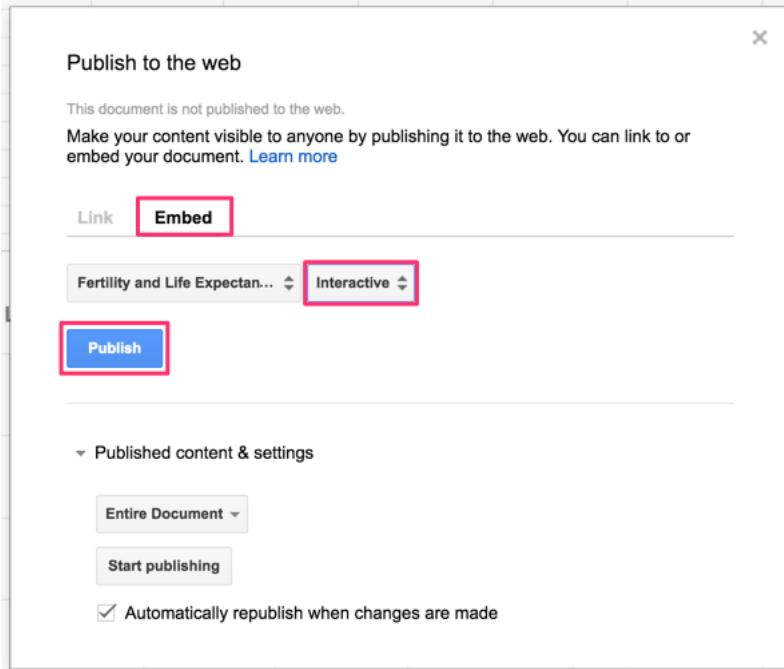


Figure 9.2: Screenshot: Publish to the web for a Google Sheets chart

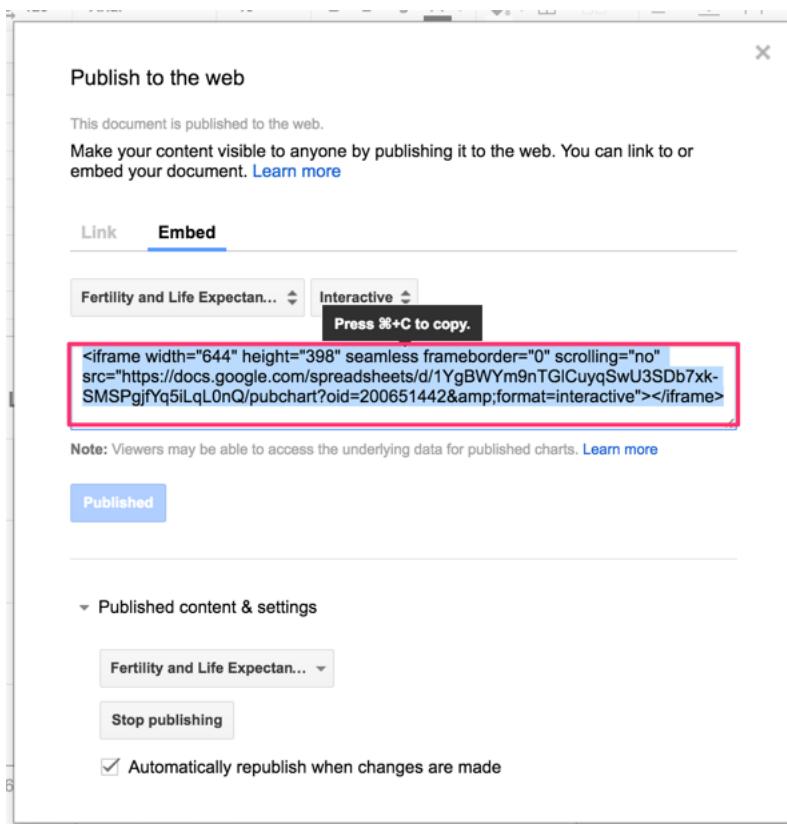


Figure 9.3: Screenshot: Copy the iframe code from a Google Sheets chart

- src: the web address (or URL) of the visualization to be displayed in the second site

See the next tutorial in this chapter, Embed iframe in GitHub Pages, to learn how to paste the iframe into a simple web page. Or see related tutorials in this chapter to embed an iframe in other common web sites.

Convert a Weblink into an Iframe

After you publish your data visualization to the web, how do you convert its weblink (or URL) into an iframe, to embed in your personal website?

The answer depends: did you publish your visualization as a code template on GitHub Pages? Or did you publish it using a drop-and-drag tool such as Google Sheets or Tableau Public?

Published with a code template on GitHub Pages

If you published your visualization from a code template (such as Leaflet or Chart.js) with GitHub Pages, follow these easy steps:

- 1) Copy the URL of your published visualization on GitHub, which will be in this format:

```
https://USERNAME.github.io/REPOSITORY
```

- 2) Add `iframe` tags to the beginning and end, insert `src=` and enclose the URL inside quotation marks, like this:

```
<iframe src="https://USERNAME.github.io/REPOSITORY"></iframe>
```

- 3) Optional: Insert preferred width and height (in pixels by default, or percentages), like this:

```
<iframe src="https://USERNAME.github.io/REPOSITORY" width="90%" height="400"></iframe>
```

- 4) Go to the appropriate tutorial to embed your iframe in your personal website:
 - Embed an iframe in GitHub Pages
 - Embed an iframe in WordPress.org

Published with Google Sheets or Tableau Public

Or, if you published your visualization using a drop-and-drag tool, see these tutorials:

- Copy an iframe code from a Google Sheets interactive chart
- Embed Tableau Public on your Website

Embed an Iframe in GitHub Pages

Question: After you create an interactive chart or map, how do you embed the live version in a website that you control?

Here's the full three-step answer that combines lessons from the Embed on the Web chapter introduction and the two previous tutorials:

- 1) First, create a web page that supports iframe embed codes. If you don't know what that means or don't yet have a personal website, go back to the previous tutorial, Create a Simple Web Page with GitHub Pages, or see the video and step-by-step instructions below.
- 2) Second, copy or create an iframe code from your data visualization. Go back to the previous tutorial, Copy an iframe code from a Google Sheets interactive chart, or see the video and step-by-step instructions below.
- 3) Third, embed (or paste) the iframe code into your website. The video and instructions below show how to paste an iframe from a Google Sheets interactive chart into a simple web page with GitHub Pages.

Try it:

The goal is to embed the iframe code from a Google Sheets interactive chart, which resides on a Google web server, into your GitHub Pages web site. The result will be similar to the one below:

TODO: Convert to code-chunk iframe: <https://docs.google.com/spreadsheets/d/1YgBWYm9nTGlCuyqSwU3SDb7xk-SMSPgjfYq5iLqL0nQ/pubchart?oid=200651442&format=interactive>

[Video](<https://youtube.be/enjhlnqaXOE>)

- 1) Sign up for free GitHub account, then sign in, at <https://github.com>.
- 2) Create a **new repository** (think of it as a folder that contains your project).

- 3) Name your repository (or “repo”), and select *Initialize this repository with a README*. Optional steps: add a description and select a license.
- 4) Scroll down and click the green button to Create your repo, which will appear in a new browser tab, with this URL format:

`https://github.com/YOUR-USERNAME/YOUR-REPO-NAME`

- 5) In your GitHub repo, click on Settings tab, scroll down to *GitHub Pages*, select **master branch** as your Source, then Save. This publishes the code from your repo to the public web.
- 6) When the Settings page refreshes, scroll back down to GitHub Pages to see the new link to your published website, which will appear in this format:

`https://YOUR-USERNAME.github.io/YOUR-REPO-NAME`

- 7) Right-click and Copy this link to your published web site.
- 8) At the top of the page, click on the repo name to return to the main level.
- 9) Click the README.md file to open it in your browser, and click the pencil symbol in the upper right corner to edit it.
- 10) Inside your README.md file, paste the link to your published web site, and type any text you wish to appear. The .md extension refers to Markdown, an easy-to-read markup language that GitHub Pages can process and display as HTML.
- 11) Go to a data visualization you have created, such as a Google Sheets chart, select Publish > Embed, and copy the iframe code. This line of HTML code displays the interactive visualization website inside your personal website.
- 12) Scroll down and click Commit to save your edits.
- 13) When your GitHub repo page refreshes, click on the new link to go to your published web site. **BE PATIENT!** Your new site may not appear instantly. Refresh the browser every 10 seconds. You may need to wait for a few minutes for a new site to appear the first time, but later changes will be much faster.

Important:

- A published README.md file will display an HTML iframe code, unless you add other HTML files (such as index.html) to your repository.

Remember that GitHub Pages is designed to create simple web pages and sites. See other web publishing tools mentioned in this chapter to create more sophisticated web sites.

Embed an Iframe on WordPress.org

TODO:

- rewrite this tutorial to merge the two versions (top and bottom)
- then update all links and check all `code` tags

To embed one web page (the data visualization) inside a second web page (the organization's website), we use a simple HTML code known as **iframe**. (Read more about the iframetag at W3Schools.)

The **general iframe concept** works across many data visualization tools and many websites: - Copy the embed code or URL from your dataviz website - Paste (and modify) the code as an iframe in your destination website

To embed your dataviz in a self-hosted Wordpress.org site, the [iframe plugin] (<http://wordpress.org/plugins/iframe/>) must be installed and activated. This plugin allows authors to embed iframe codes inside posts/pages, 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. **I have already installed and activated** the iframe plugin on my site, and the Dashboard view looks like this:



Note that most WordPress.com sites do NOT support an iframe embed code.

But details vary, so read and experiment with the examples that follow.

- 5) To embed the iframe in a WordPress.org site, the iframe plugin must be installed, as explained in the Embed with iframe on WordPress.org chapter. **TO DO** fix self-reference
- 6) Log into your Wordpress.org site and create a new post. In the editor window, switch from the Visual to the Text tab, which allows users to modify the code behind your post. Paste the iframe code from your interactive dataviz.

The screenshot shows the WordPress editor's Text tab selected. The toolbar above has buttons for Add Media, Visual, and Text. The Text tab is highlighted with a red box. Below the toolbar is a row of buttons: b, i, link, b-quote, del, ins, img, ul, ol, li, code, and more. A 'close tags' button is also present. The main area contains the following HTML code:

```
<iframe width="600" height="371" seamless frameborder="0" scrolling="no"
src="https://docs.google.com/spreadsheets/d/1fwnl5hvkkwz-
YDZrogGnx274BqmozG1IeXyjJ2TKmE/pubchart?
oid=462316012&format=interactive"></iframe>
```

- 7) Initially, the code you pasted includes HTML iframe tags at the front <iframe... and the end ...></iframe>, which looks like this:

```
<iframe width="600" height="371" seamless frameborder="0" scrolling="no"
src="https://docs.google.com/spreadsheets/d/1fwnl5hvkkwz-YDZrogGnx274BqmozG1IeXyjJ2TKmE/pubchart?
```

- 8) Modify the front end of the iframe code by replacing the less-than symbol (<) with a square opening bracket ([). Modify the back end by erasing the greater-than symbol (>) and the end tag (). Replace the back end with a square closing bracket (]).

The screenshot shows the WordPress editor's Text tab selected. The toolbar above has buttons for Add Media, Visual, and Text. The Text tab is highlighted with a red box. Below the toolbar is a row of buttons: b, i, link, b-quote, del, ins, img, ul, ol, li, code, and more. A 'close tags' button is also present. The main area contains the following HTML code with annotations:

replace with square bracket

[**replace the end tag with square bracket**

```
[iframe width="600" height="371" seamless frameborder="0" scrolling="no"
src="https://docs.google.com/spreadsheets/d/1fwnl5hvkkwz-
YDZrogGnx274BqmozG1IeXyjJ2TKmE/pubchart?
oid=462316012&format=interactive"]
```

Your modified code should look like this:

```
[iframe width="600" height="371" seamless frameborder="0" scrolling="no"
src="https://docs.google.com/spreadsheets/d/1fwnl5hvkkwz-YDZrogGnx274BqmozG1IeXyjJ2TKmE/pubchart?
```

- 9) Click Preview or Publish/View Post to see how it appears on the web.
- 10) If desired, continue to modify the iframe code to improve the display of your dataviz on your website. For example, the initial code was 600 pixels wide (width="600"). To display the dataviz across the full width of your website, change this part of the code to 100% (width="100%").

The goal is to embed an interactive chart inside your website, so that users can explore the data. This tutorial displays a *very basic chart* to simplify the process, and the end result will appear like the one below. Try it.

TODO: Convert to code-chunk iframe: <https://docs.google.com/spreadsheets/d/1fwnl5hvkkwz-YDZrogGnx274BqmozGlleXyjJ2TKmE/pubchart?oid=462316012&format=interactive>

Embed Tableau Public on your Website

Question: After learning how to create an interactive data visualization with Tableau Public in this book, how do I embed it on my website?

Answer: Tableau Public supports two embedding methods, and your choice depends on your type of website.

- A) Embed code: if you can paste directly into an HTML web page
- B) Convert Link to iframe: to paste into WordPress.org, Wix, SquareSpace, Weebly, and many other web platforms

Try it:

Both methods produce an embedded visualization like the one below. Float your cursor over points to view data details.

TODO: convert to code-chunk iframe: <https://public.tableau.com/views/CTSchoolDistrictsbyIncomeandGradeLevels2009-13/Sheet1?:showVizHome=no&:embed=true>

A) Embed code method for HTML web pages

- 1) Use this method if you can paste HTML and JavaScript code directly into a website with HTML pages.
- 2) Go to the public web page of any Tableau Public visualization, such as this sample: <https://public.tableau.com/profile/jackdougherty#!/vizhome/CTSchoolDistrictsbyIncomeandGradeLevels2009-13/Sheet1>
- 3) Before you begin the embed process, click the upper-right Edit Details button to make any final modifications to the title or toolbar settings.
- 4) Click the bottom-right Share button, click inside the **Embed Code** field, and copy its contents. A typical embed code is a long string of HTML and JavaScript instructions to display the visualization.



Figure 9.4: Screenshot: Edit and Share buttons in Tableau Public web page

- 5) Open an HTML page on your website and paste the embed code in the body section. Below is an example of a sample Tableau Public embed code pasted between the body tags of a simple HTML page.

TODO: find a way to replace this triple backtic code snippet, since it may throw errors into Markdown output.

```
<!DOCTYPE html>
<html>
<head>
    <title>sample web page</title>
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta charset="utf-8">
</head>
<body>
    <div class='tableauPlaceholder' id='viz1489158014225' style='position: relative'><noscript><a href="https://public.tableau.com/views/CTSchoolDistrictsbyIncomeandGradeLevel/CTSchoolDistrictsbyIncomeandGradeLevel?embed=true">View in Browser</a></noscript><div class='tableauPlaceholderContent'><div>This visualization requires JavaScript to display properly. Please enable JavaScript in your browser's settings, and then refresh this page.</div></div></div>
</body>
</html>
```

B) Convert Link to iframe method

- 1) Use this method if you need to paste an iframe into common web authoring platforms (such as WordPress.org, Squarespace, Wix, Weebly, etc.), since these platforms typically do not support HTML and JavaScript code pasted directly into content.
- 2) Go to the public web page of any Tableau Public visualization, such as this sample: <https://public.tableau.com/profile/jackdougherty#!/>
vizhome/CTSchoolDistrictsbyIncomeandGradeLevels2009-13/Sheet1
- 3) Before you begin the embed process, click the upper-right Edit Details button to make any final modifications to the title or toolbar settings.
- 4) Click the bottom-right Share button, click inside the **Link** field (NOT the Embed Code field), and copy its contents.



Figure 9.5: Screenshot: Edit and Share buttons in Tableau Public web page

- 5) A typical link will look similar to this example (scroll to right to see all):

<https://public.tableau.com/views/CTSchoolDistrictsbyIncomeandGradeLevels2009-13/Sheet1>

- 6) We need to edit the link to convert it into an iframe format. First, delete any code that appears after the question mark, to make it look like this (scroll to right to see all):

```
https://public.tableau.com/views/CTSchoolDistrictsbyIncomeandGradeLevels2009-13/Sheet1?
```

- 7) Add this snippet of code to the end, to replace what you deleted above:

```
:showVizHome=no&:embed=true
```

- 8) Now your edited link should look similar to this (scroll to right to see all):

```
https://public.tableau.com/views/CTSchoolDistrictsbyIncomeandGradeLevels2009-13/Sheet1?:showVizH
```

- 9) Enclose the link inside an iframe source tag `src=` with quotes, to make it look similar to this (scroll to right to see all):

```
src="https://public.tableau.com/views/CTSchoolDistrictsbyIncomeandGradeLevels2009-13/Sheet1?:showVizH"
```

- 10) Add iframe tags for `width` and `height` in percentages or pixels (default), to make it look similar to this (scroll to right to see all):

```
src="https://public.tableau.com/views/CTSchoolDistrictsbyIncomeandGradeLevels2009-13/Sheet1?:showVizH" width="90%" height="500"
```

Hint: Insert 90% width, rather than 100, to help readers easily scroll down your web page

- 11) Add iframe tags at the beginning and end, to make it look similar to this (scroll to right to see all):

```
<iframe src="https://public.tableau.com/views/CTSchoolDistrictsbyIncomeandGradeLevels2009-13/Sheet1?:showVizH" width="90%" height="500"></iframe>
```

Exceptions to the last step above. As described in the Embed iframe on WordPress chapter in this book, in a self-hosted WordPress.org site, with the iframe plugin, insert iframe brackets rather than HTML tags to make a shortcode like this (scroll to right to see all):

```
[iframe src="https://public.tableau.com/views/CTSchoolDistrictsbyIncomeandGradeLevels2009-13/Sheet1?:showVizH" width="90%" height="500"]
```

Learn more: Embedding Tableau Public Views in iframe, Tableau Support page <http://kb.tableau.com/articles/howto/embedding-tableau-public-views-in-iframes>

Summary

TODO

Chapter 10

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. 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 7: 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 10.1.

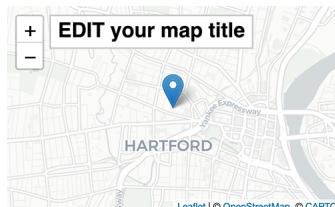


Figure 10.1: Create your own version of this simple interactive Leaflet map.

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 10.2.

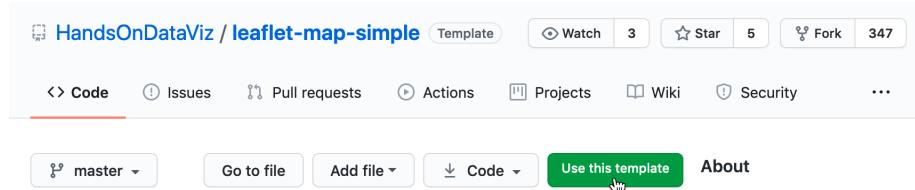


Figure 10.2: Click *Use this template* to make your own copy.

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 10.3. 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 10.3: Name your copied repo `leaflet-map-simple`.

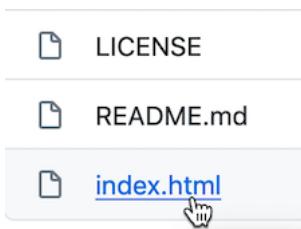


Figure 10.4: Click the `index.html` file to view the code.

5. Click on the `index.html` file to view the code, as shown in Figure 10.4.

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 10.5.

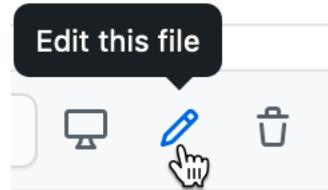


Figure 10.5: Click the pencil button to edit the code.

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 10.6.

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

Commit changes

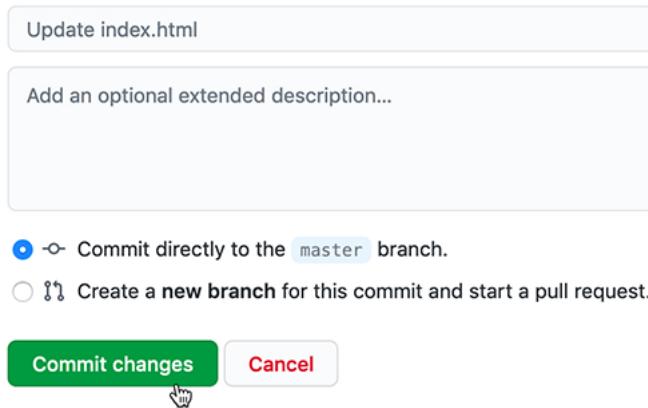


Figure 10.6: Click the green *Commit Changes* button to save your edits.

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.

9. To access GitHub Pages, scroll to the top of your repo page and click the *Settings* button as shown in Figure 10.7.

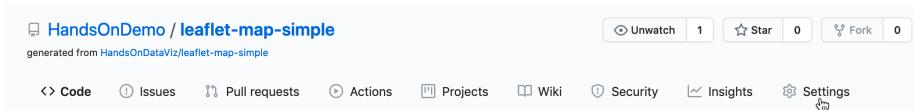


Figure 10.7: Click the *Settings* button to access GitHub Pages and publish your work on the web.

10. In the Settings screen, scroll way down to the GitHub Pages area. In the drop-down menu, change *Source* from *None* to *Master*, keep the default */(root)* option in the middle, and press *Save* as shown in Figure 10.8. 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.

Note: **TODO:** GitHub recently announced it plans to change the default branch from *Master* to *Main* to eliminate its master-slave metaphor. GitHub recom-

GitHub Pages

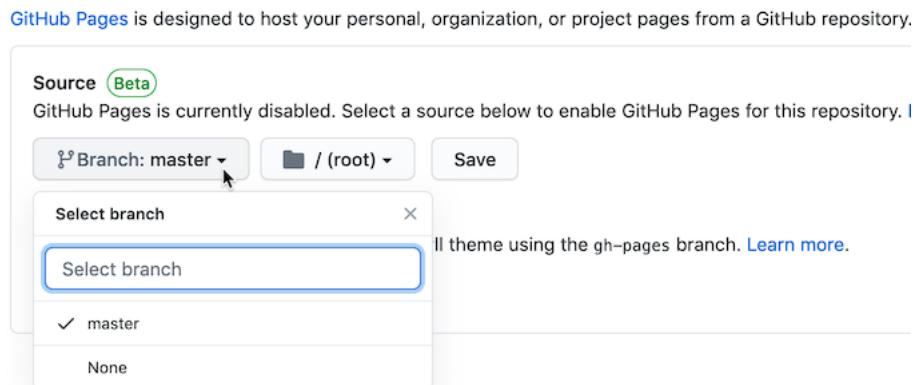


Figure 10.8: In *Settings*, go to *GitHub Pages*, and switch the source from *None* to *Master*.

mends waiting until later in 2020 for their system to support this change. When that happens, we need to update repos, text, and screenshots. See more at <https://github.com/github/renaming>

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 10.9.

GitHub Pages

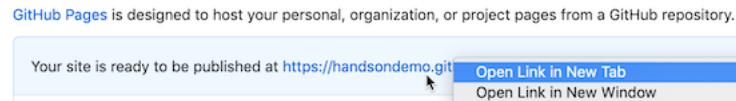


Figure 10.9: In *Settings* for *GitHub Pages*, right-click your published map link to open in a new tab.

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 10.10.



Figure 10.10: 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 10.11. 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 for editing. The content of the README file is as follows:

```
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 10.11: 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 11 and Leaflet map templates in Chapter 12.

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 11 and Leaflet map templates in Chapter 12. 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 10.12. 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 `master` or `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 10.13.

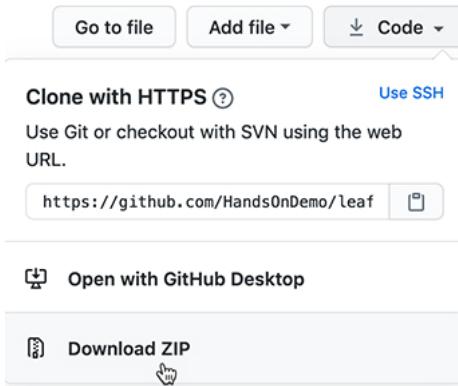


Figure 10.12: Click *Code* and select *Download Zip* to create a compressed folder of a repo on your computer.

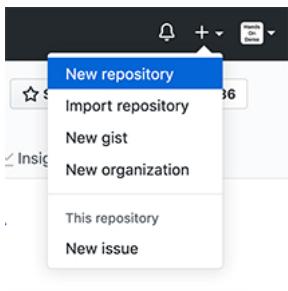


Figure 10.13: Click the plus (+) symbol in upper-right corner to create a new repo.

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 10.14.

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 10.15.

Create a new repository

A repository contains all project files, including the revision history. [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 10.14: 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.

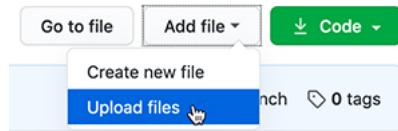


Figure 10.15: Click the *Upload Files* button.

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 10.16. 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.

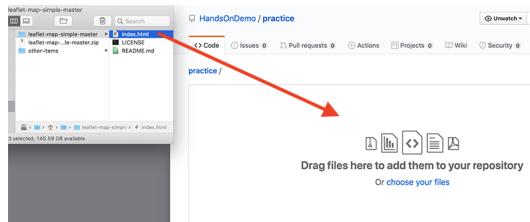


Figure 10.16: Drag-and-drop the `index.html` file to the upload screen.

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 10.17. 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.

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

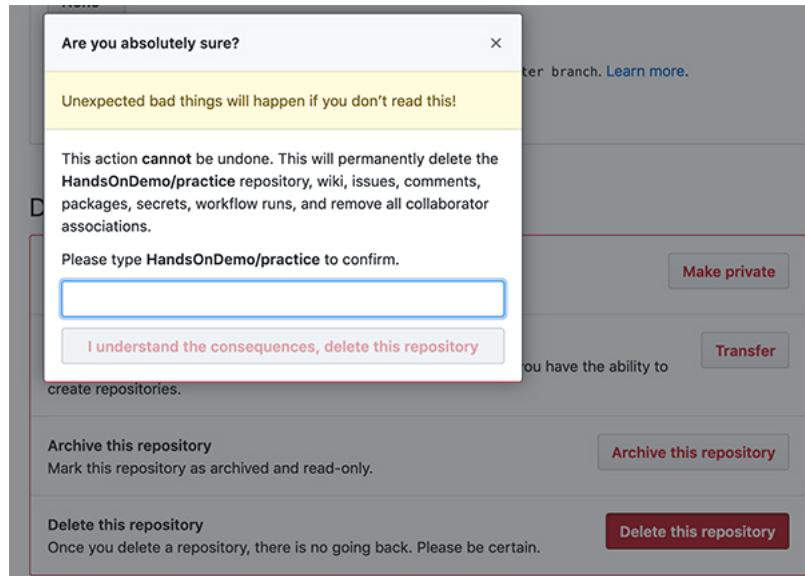


Figure 10.17: After clicking the Delete Repository button, GitHub will ask you to type your username and repo name to confirm.

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 10.18. The next screen will show a link to the GitHub Desktop web page, and you should download and install the application.

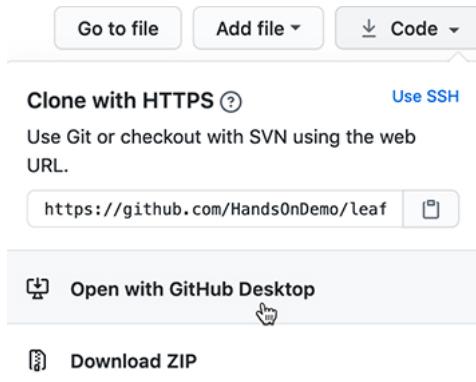


Figure 10.18: In your GitHub repo on the web, click *Code* to *Open with GitHub Desktop* to download and install GitHub Desktop.

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 10.19, 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.

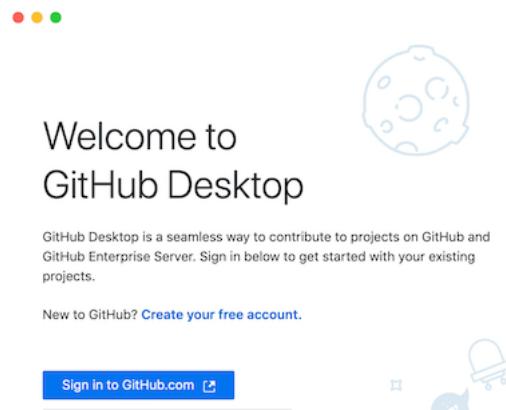


Figure 10.19: Click the blue *Sign in to GitHub.com* button to link GitHub Desktop to your GitHub account.

3. In the next setup screen, GitHub Desktop asks you to configure Git, the underlying software that runs GitHub. Confirm that it displays your user-name and click *Continue*, as shown in Figure 10.20.

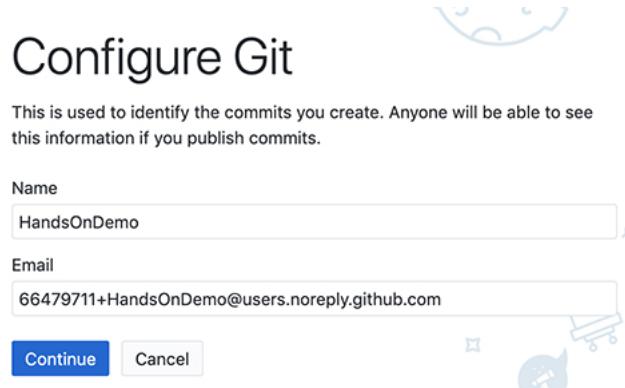


Figure 10.20: Click the *Continue* button to authorize GitHub Desktop to send commits to your GitHub account.

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 10.21.

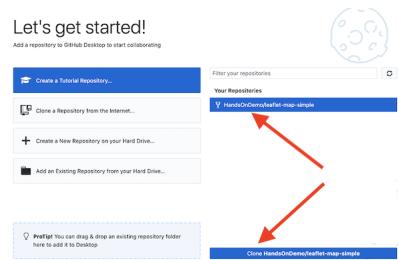


Figure 10.21: Select your `leaflet-map-simple` repo and click the *Clone* button to copy it to your local computer.

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 10.22. Before you click the *Clone* button, remember the path to this location, since you’ll need to find it later.

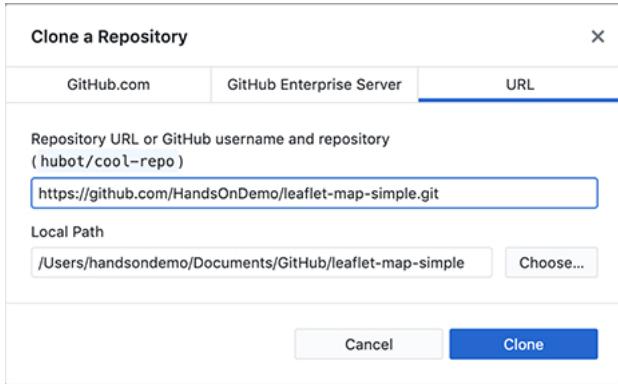


Figure 10.22: Select the Local Path where your repo will be stored on your computer, then click *Clone*.

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 10.23.
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 10.24. 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 10.25.
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 10.26. Open your `index.html` file and edit the title of your map, around line 22, then save your work.
10. 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 10.27. 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.

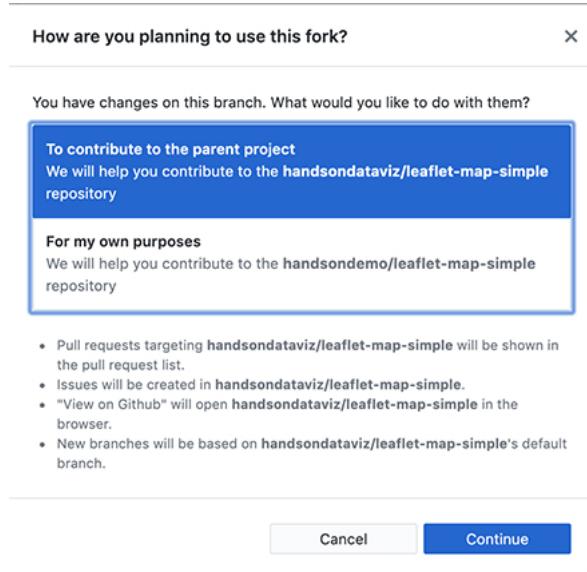


Figure 10.23: If asked how you plan to use this fork, select the default *To contribute to the parent project* and click *Continue*.

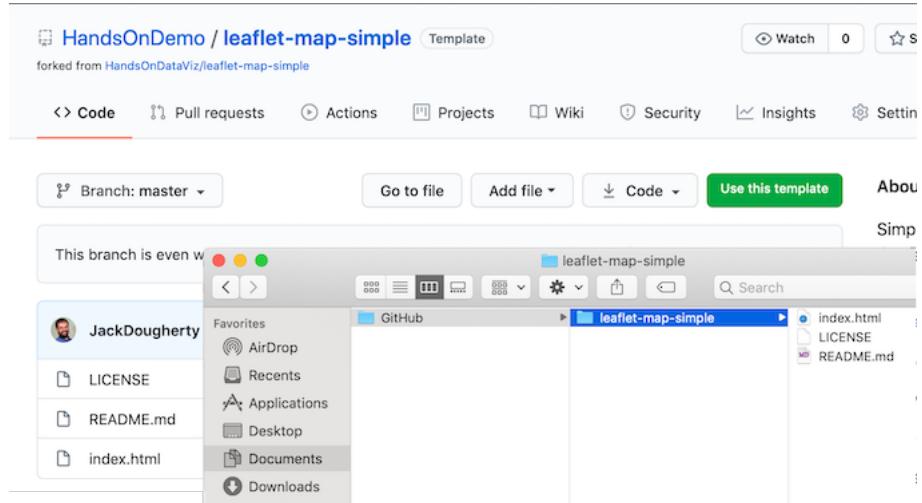


Figure 10.24: Now you have two copies of your repo: in your GitHub online account (on the left) and on your local computer (on the right, as shown in the Mac Finder). Windows screens will look different.

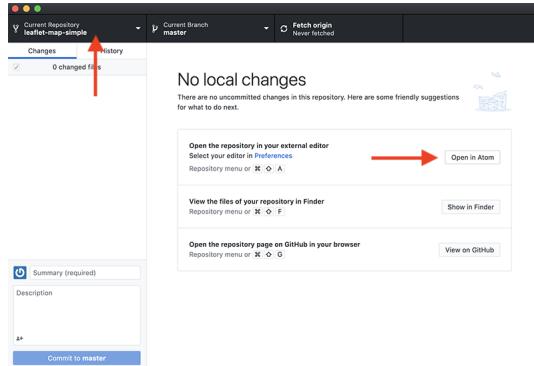


Figure 10.25: In GitHub Desktop, confirm the Current Repo and click the *Open in Atom* button to edit the code.

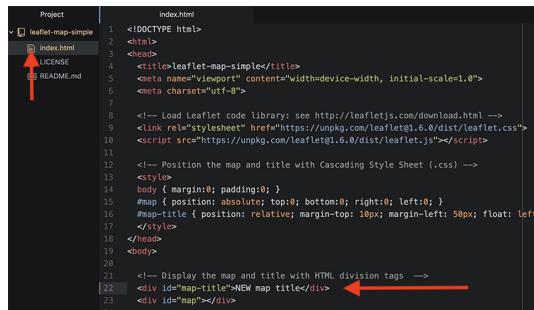


Figure 10.26: Atom Editor opens your repo as a *project*, where you can click files to view code. Edit your map title.

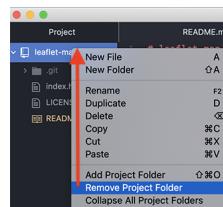


Figure 10.27: To clean up your Atom Editor workspace, right-click to *Remove Project Folder*.

11. 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 10.28.

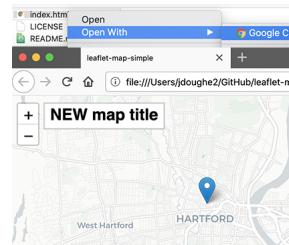


Figure 10.28: Right-click the `index.html` file on your local computer and open with a browser to check your edits.

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 to Master* 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 10.29.

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

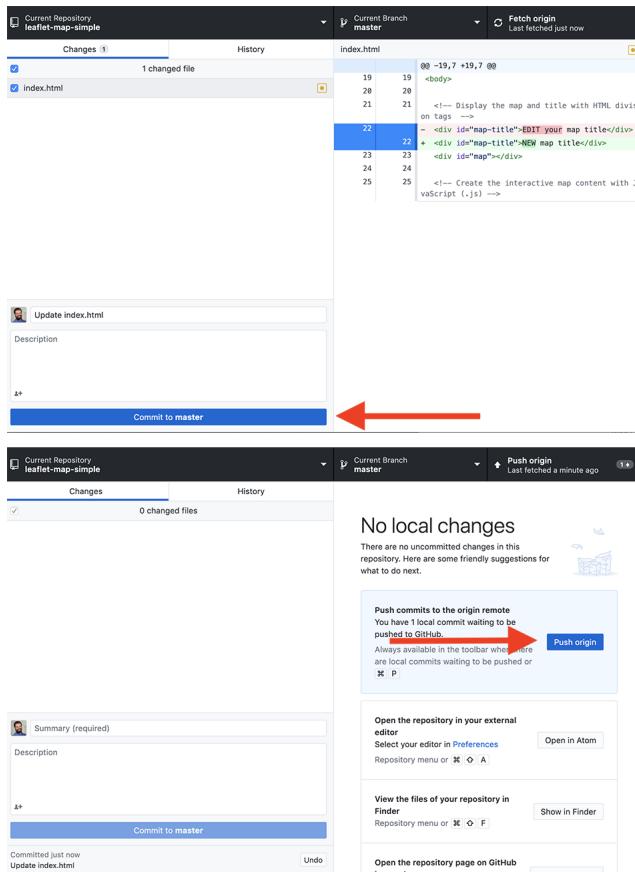


Figure 10.29: In this two-step process, click *Commit to Master*, then click *Push origin* to save and copy your edits from your local computer to your GitHub web account, as shown in this animated GIF.

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 10.30.

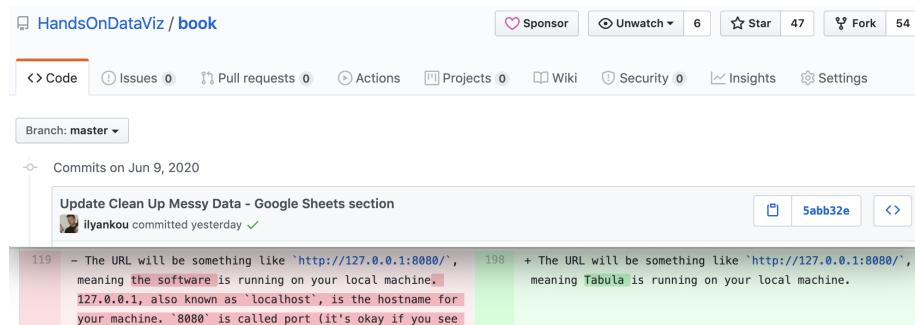


Figure 10.30: Drag-and-drop the file to the upload screen.

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 sug-

gested 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.

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 makes it much easier to work with Chart.js and Highcharts code templates in Chapter 11 and the Leaflet map code templates in Chapter 12. Let’s build on your brand-new coding skills to create more customized charts and maps in the next two chapters.

Chapter 11

Chart.js and Highcharts Templates

We recommend that you start creating interactive charts with easy drag-and-drop tools we described in Chapter 6: Chart Your Data, such as Google Sheets, Datawrapper and Tableau Public. But when you're ready to explore more powerful tools, this chapter features Chart.js and Highcharts code templates, which offer many benefits for more advanced users. By copying and modifying our templates, you can customize the appearance and interactivity of your visualizations, upload your data in CSV format, and freely publish it with your code on GitHub Pages or any server you choose, without relying on a drag-and-drop tool platform to continue its service into the future. But in order to work with our templates, you first need to learn how to edit and host code with GitHub in Chapter 10.

Most of the templates we feature are designed with Chart.js, an open-source code library that is freely distributed under an MIT license, which anyone can use and modify. This means that the code library is extensible by other users, who have created awesome Chart.js resources, such as additional chart types and plugins that provide more features. Also, we include some templates designed with Highcharts to expand the range of chart types and features not fully covered by Chart.js. Although Highcharts code is open-source, its licensing terms differ. Highcharts is free for non-commercial use, such as a personal, school, or non-profit organization website. But if you create charts for commercial or governmental use, you'll need to purchase a Highcharts license. In either case, we designed our Chart.js and Highcharts code templates to make them easier for beginners to understand. See the various types of charts and follow the links to the templates and tutorials in Table 11.1.

Table 11.1: Chart Code Templates and Tutorials

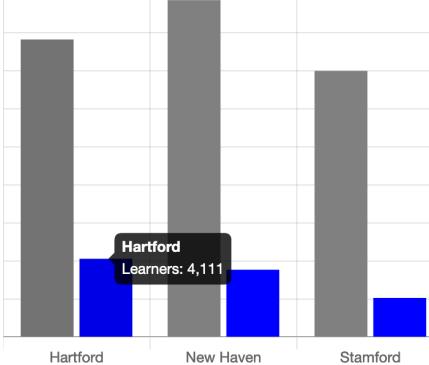
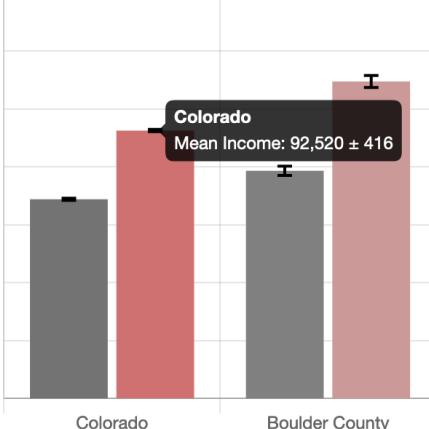
Chart	Best use and tutorials in this book									
Bar or Column Chart  <p>A bar chart comparing three locations: Hartford, New Haven, and Stamford. The y-axis represents the number of learners. Hartford has approximately 4,111 learners, New Haven has approximately 10,000 learners, and Stamford has approximately 3,000 learners. A callout box highlights Hartford with the text "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>~10,000</td> </tr> <tr> <td>Stamford</td> <td>~3,000</td> </tr> </tbody> </table>	Location	Learners	Hartford	4,111	New Haven	~10,000	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	~10,000									
Stamford	~3,000									
Error Bars in a Bar/Column Chart  <p>A bar chart comparing the mean income for 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 \$85,000 ± \$1,000. Error bars are shown above each bar.</p> <table border="1"> <thead> <tr> <th>Location</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>\$85,000</td> <td>± \$1,000</td> </tr> </tbody> </table>	Location	Mean Income	Error	Colorado	\$92,520	± \$416	Boulder County	\$85,000	± \$1,000	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
Location	Mean Income	Error								
Colorado	\$92,520	± \$416								
Boulder County	\$85,000	± \$1,000								

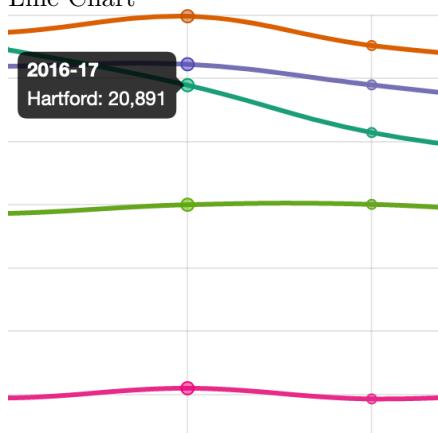
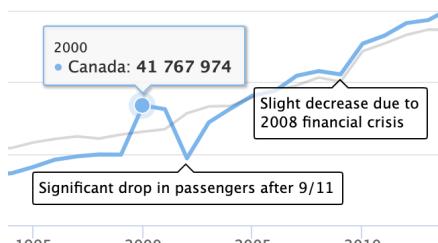
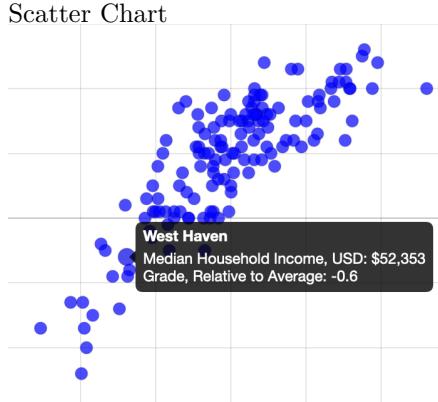
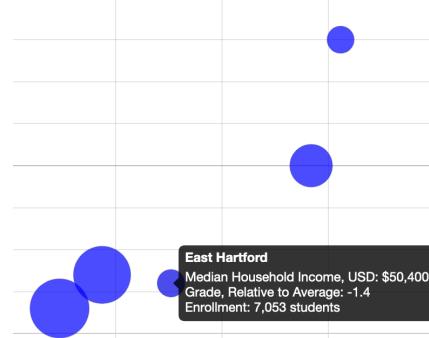
Chart	Best use and tutorials in this book
Line Chart 	Best to show continuous data, such as change over time. Power tool: Line Chart with CSV data in Chart.js code template and tutorial
Annotated Line Chart 	Best to add contextual notes inside chart of continuous data, such as change over time. Power 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

For additional chart types, see Chart.js samples and Highcharts Demos.

These code templates vary from simple to complex, but all of them rely on four basic pillars:

- HTML: language to structure content on the web (example: index.html)
- CSS, or Cascading Style Sheet: to shape how content appears on the web (example: style.css)
- JavaScript: code to create the chart and interactivity (example: script.js)
- CSV: data that powers the visualization that is expressed in comma-separated format (example: data.csv)

Also, these templates refer to other code elements:

- library: link to online instructions to complete routine tasks (example: Chart.js)
- data: content to appear in chart, typically in CSV format (example: data.csv)

Bar or Column Chart with Chart.js

Bar or column charts are best to compare categories side-by-side. If labels are long, use horizontal bars instead of vertical columns. Always start the x-axis (for a horizontal bar chart) or y-axis (for a vertical column chart) at zero. This interactive Chart.js code template pulls the data from a CSV file, as shown in Figure 11.1.

To create your own bar or column chart with CSV data using our Chart.js template:

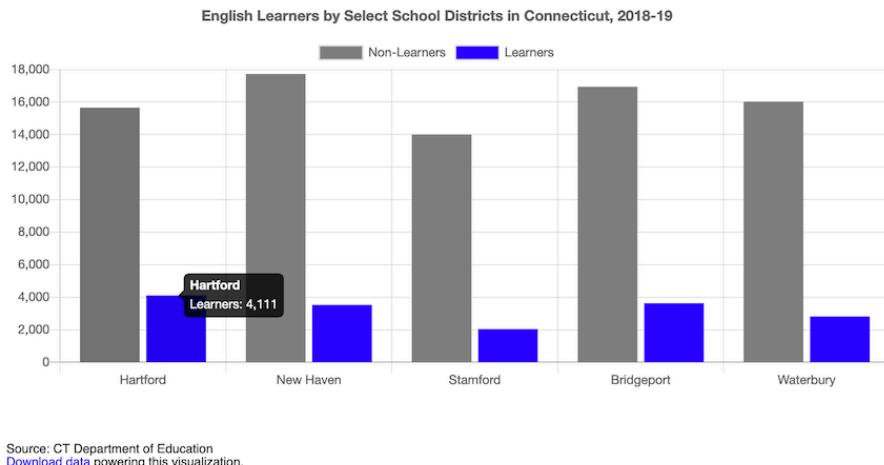


Figure 11.1: Bar chart with Chart.js: explore the interactive version.

1. Go to our GitHub repo for Chart.js template that contains the code for the chart in Figure 11.1, log into your GitHub account, and click *Use this template* to create a copy that you can edit, as described in Chapter 10: Edit and Host Code with GitHub.
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 shown in the code snippet below:

```
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;
```

For more customization, see Chart.js documentation.

Error Bars with Chart.js

If your data comes with uncertainty (margins of error), we recommend you show it in your visualizations. The bar chart template shown in Figure 11.2 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.

To create your own bar or column chart with error bars, with data loaded from a CSV file, using our Chart.js template:

1. Go to our GitHub repo for this Chart.js template that contains the code for the chart in Figure 11.2, log into your GitHub account, and click *Use*

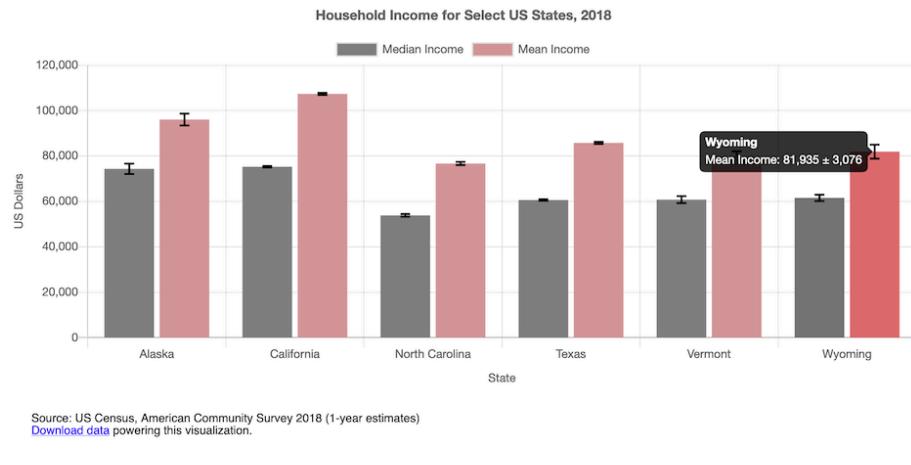


Figure 11.2: Interactive bar chart with error bars in Chart.js. Explore the interactive version.

this template to create a copy that you can edit, as described in Chapter 10: Edit and Host Code with GitHub.

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. The line chart in Figure 11.3 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.

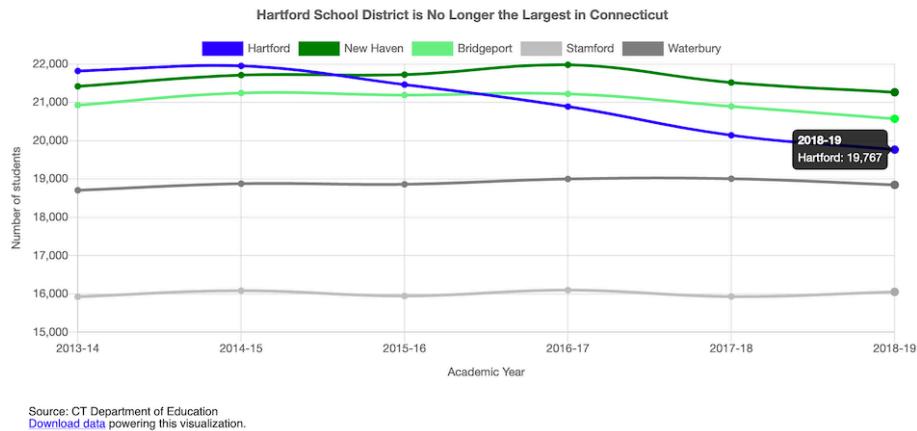


Figure 11.3: Interactive line chart with Chart.js. Explore the interactive version.

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 11.3, log into your GitHub account, and click *Use this template* to create a copy that you can edit, as described in Chapter 10: Edit and Host Code with GitHub.
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, but do not overload your chart with lines as humans can distinguish a limited number of colors.

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;

```

For more customization, see Chart.js documentation.

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 11.4 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.

To create your own annotated line chart with Highcharts, with data loaded from a CSV file, you can:

1. Go to our GitHub repo that contains code for the chart shown in Figure 11.4, log into your GitHub account, and click *Use this template* to create a copy that you can edit, as described in Chapter 10: Edit and Host Code with GitHub.
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



Figure 11.4: Interactive annotated chart with Highcharts. Explore the interactive version.

	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;
```

For additional customization, see Highcharts API reference.

Scatter Chart with Chart.js

Scatter charts (also *scatterplots*) are used to display data of 2 or more dimensions. The scatter chart in Figure 11.5 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.

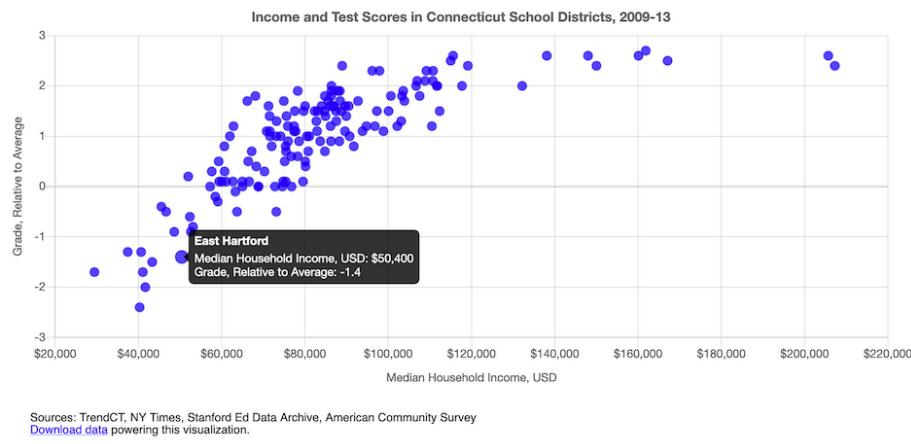


Figure 11.5: Interactive scatter chart with Chart.js. Explore the interactive version.

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 11.5, log into your GitHub account, and click *Use this template* to create a copy that you can edit, as described in Chapter 10: Edit and Host Code with GitHub.
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

```

For more customization, see Chart.js documentation.

Going beyond two dimensions

To show more than two dimensions in scatter charts, one can:

- **color** each data point differently to show third dimension, eg use shades of red and green to show 5-year trend in test performance;
- **resize** each data point to display fourth dimension, eg number of students in each school district;
- use different **icons** or **glyphs** to display fifth dimension, eg circles for male students and squares for female students.

Remember not to overwhelm the viewer and communicate only the data that are necessary to prove or illustrate your idea.

Bubble Chart with Chart.js

Bubble charts are similar to scatter plots, but it adds one more dimension: the size of each point (marker) also represents a value.

The bubble chart in Figure 11.6 shows how median household income (x-axis) and test performance (y-axis) in 6 school districts in Connecticut are related.

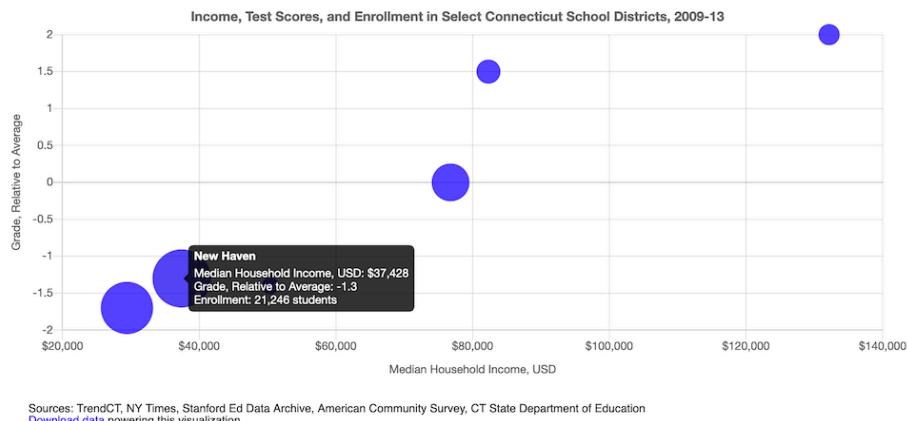


Figure 11.6: Interactive bubble chart with Chart.js. Explore the interactive version.

The size of data point corresponds to the number of students enrolled in the school district: bigger circles represent larger school districts.

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, as described in Chapter 10: Edit and Host Code with GitHub.
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.

Going beyond three dimensions

To show more than three dimensions in bubble charts, one can:

- **color** each data point differently to show fourth dimension, eg use shades of red and green to show 5-year trend in test performance;
- use different **icons** or **glyphs** to display fifth dimension, eg circles for male students and squares for female students.

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 for basic chart types. You can use them as a base to kickstart your interactive visualization. Both Chart.js and Highcharts have many more customizations than we suggested in code snippets. Look at the additional documentation

In the next chapter, we will introduce Leaflet.js map templates that you can as easily copy from GitHub to develop your own.

Chapter 12

Leaflet Map Templates

In Chapter 7: Map Your Data, we described several drag-and-drop tools designed for beginners, such as 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 with skills you learned in Chapter 10: 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 first two easy-to-use 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, displays your map as a scrolling narrative of chapters to guide readers through a storyline, with the option to display paragraphs of text, images, audio or video clips, and historical map backgrounds loaded into your linked Google Sheet and GitHub repo. If you wish to add some of these extra features, look back at Chapter 3 to geocode addresses with a Google Sheets Add-on, or jump ahead to Chapter 13: Transform Your Map Data to learn how to create and edit polygons and polylines with the GeoJson.io tool, edit or join data with polygons using the MapShaper tool, or georectify a scanned map to use as a background overlay with the MapWarper tool.

Our remaining 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* about how things work on your computer, start with the Leaflet Maps with CSV Data tutorial, which walks you through the steps of creating a point map that pulls data from a CSV file, a generic spreadsheet format we discussed in Chapter 3. Then move on to the Leaflet Maps with

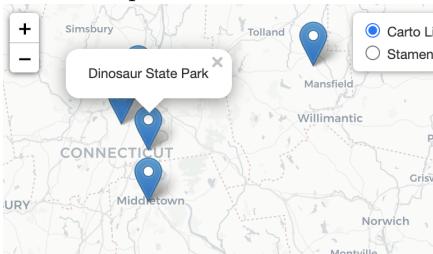
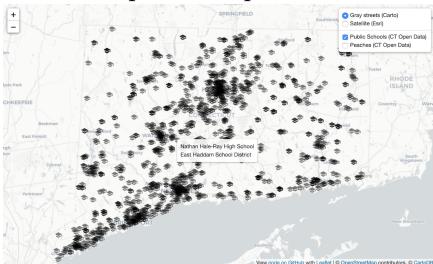
Open Data API tutorial, to learn how to code using an application program interface to pull information directly from open data repositories as we described in Chapter 4. In both of these templates, you’ll learn how Leaflet maps are written using three coding languages:

- HTML: to structure content on the web page, typically in a file named `index.html`.
- CSS or Cascading Style Sheet: to shape how content appears on the page, either inside `index.html` or in a separate file such as `style.css`.
- JavaScript: to create the interactive map using instructions from the Leaflet code library, either inside `index.html` or in a separate file, such as `script.js`.

Explore our Leaflet map templates and you’ll also see how they refer to different code components, such as basemap tiles from various open-access online providers, such as Carto, Esri, Stamen, and Open Street Map, that allow you to zoom into background maps. You’ll also see data files to place information about points, polygons, or polylines on top of the map, usually in CSV or GeoJSON format—see Chapter 13, with names similar to `data.csv` or `map.geojson`. 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 you solve problems that may arise, see Fix Common Mistakes in the appendix.

Table 12.1: Map Templates 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 display your map as a scrolling narrative of chapters to guide readers through a storyline, with the option to include paragraphs of text, images, audio or video clips, and historical map backgrounds loaded into your linked Google Sheet (or CSV) and GitHub repo. Template with tutorial: Leaflet Storymaps with Google Sheets

Map Templates	Best use and tutorials in this book
Leaflet Maps with CSV Data 	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
Leaflet Maps with Open Data API 	Learn how to code your own Leaflet map with an application program interface (API) that pulls content directly from open data repositories, such as Socrata and others. Template with tutorial: Leaflet Maps with Open Data API

TODO: decide whether to add any other Leaflet map templates we created for HandsOnDataViz or OnTheLine

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 add new layers of point, polygon, or polyline data. In these cases, consider making a copy of our Leaflet Maps with Google Sheets template on GitHub. It gives you more control over choosing colors, icons, and images, and also the option to display a data table of point markers. To customize your interactive map, you enter data into a Google Sheet template, which you link directly to your copy of the Leaflet code repository, as shown in Figure 12.1 and Figure 12.2.

TODO: Create and insert a new version of the demo, featuring ECGreenway route thru CT, points with photos, and pop density of towns to highlight how this bike route connects cities.

Tutorial Outline

Make sure you meet these requirements, and read this overview to prepare yourself for this multi-step tutorial.

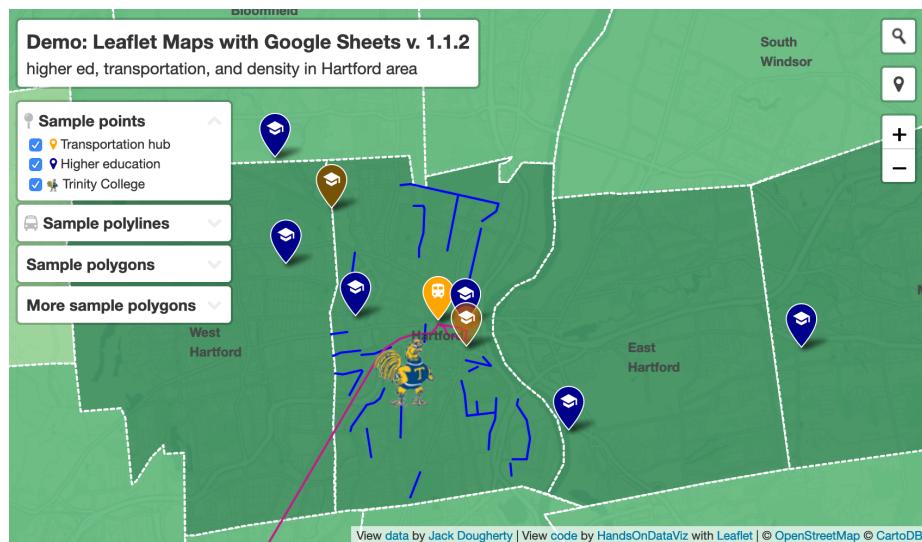


Figure 12.1: Explore a live demonstration of Leaflet Maps with Google Sheets.

Before you begin, you must:

- Have a Google Drive account.
- Know how to File > Make a Copy in Google Sheets, as described in Chapter 3.
- Have a GitHub account.
- Know how to Edit and Host Code with GitHub, as described in Chapter 10.

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 half, 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, based on information you enter into the linked Google Sheet and upload to your GitHub repo.

	A	B
1	Setting	Customize
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-jBPBt5y
20	Search Radius	5 miles
21	Search Results Zoom Level	12

+
≡
Options ▾
Points ▾
Polylines ▾
Polygons ▾
Polygons1 ▾

Figure 12.2: Explore the live Google Sheet template that feeds data into the Leaflet map above.

- E) Geocode locations and customize new markers in the Points tab
- F) Remove or display point, polygon, or polylines data and legends

Then you will finalize your map by following either step G *OR* step H:

- 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. Right-click to open this GitHub code template in a new tab: <https://github.com/handsondataviz/leaflet-maps-with-google-sheets>
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 *Master*, keep the default `/(root)` setting, and press *Save* as shown in Figure 12.3. 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.

Note: **TODO:** GitHub recently announced it plans to change the default branch from *Master* to *Main* to eliminate its master-slave metaphor. GitHub recommends waiting until later in 2020 for their system to support this change. When that happens, we need to update repos, text, and screenshots. See more at <https://github.com/github/renaming>

5. Scroll down to GitHub Pages section again, and copy the link to your published web site, which will appear in this format:

GitHub Pages

[GitHub Pages](#) is designed to host your personal, organization, or project pages from a GitHub repository.

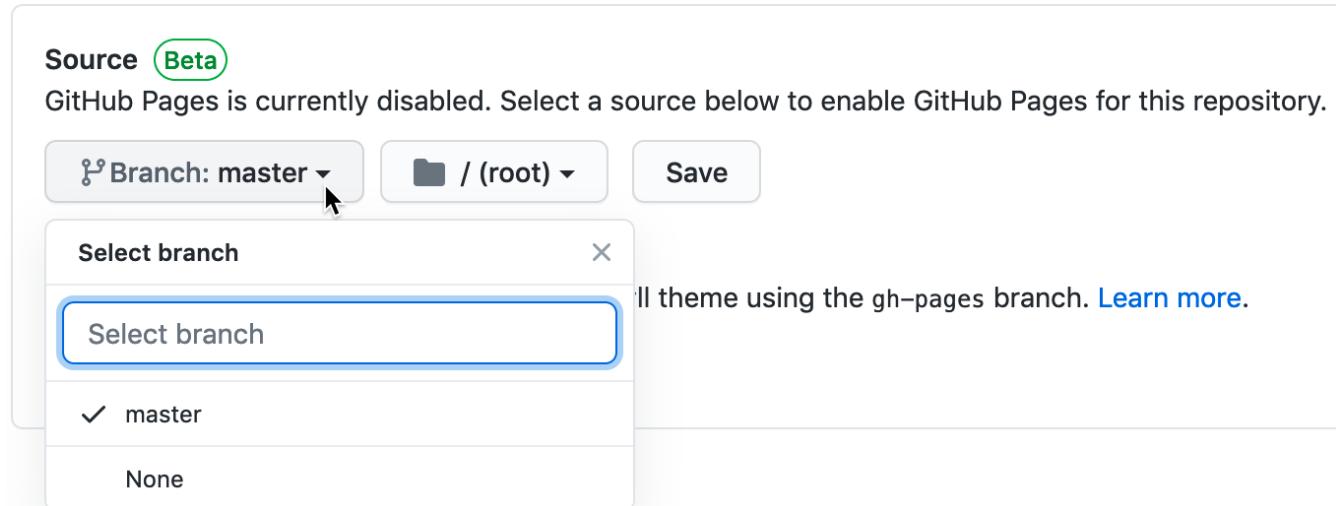


Figure 12.3: In *Settings*, go to *GitHub Pages*, and switch the source from *None* to *Master*.

<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, and paste in the link to *your* published site. Scroll down and *Commit* to save your edits.

TODO: Insert image here

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 this 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 12.4.

C) Paste your Google Sheet browser address in two places in your GitHub repo

Our next task is to link your Google Sheet to your Leaflet code in GitHub, so that content from your Sheet will appear in your 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.

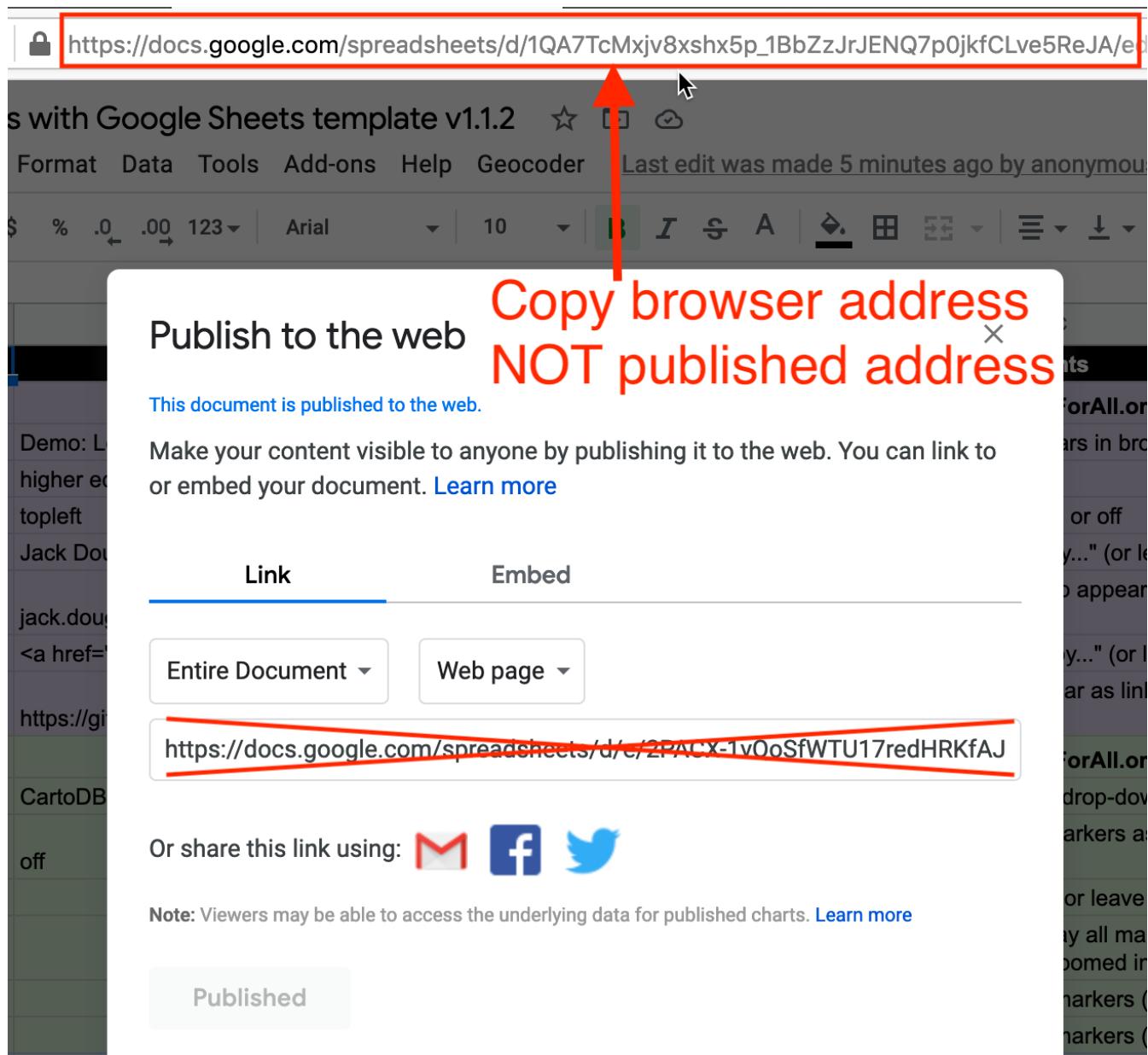


Figure 12.4: Copy the Google Sheet address at the top of the browser, NOT the *Publish to the web* address.

2. Paste *your* Google Sheet address or URL (which usually ends in `...XYZ/edit#gid=0`) to replace the existing URL, as shown in Figure 12.5. Be careful *NOT* to erase the single quotation marks or the semicolon at the end. See separate instructions about the Google API key further below.



```

<> Edit file ⌂ Preview changes Spaces ↴ 2
1 var googleDocURL = 'https://docs.google.com/spreadsheets/d/1ZxvU8eGyuN9M8GxTU9acKVJv70iC3px\_m3EVFs0HN9g/edit#gid=0';
2 var googleApiKey = 'AIzaSyBh9nKJJfK87WhY';

```

Figure 12.5: Paste in *your* Google Sheet URL to replace *our* URL.

3. Scroll to bottom of page and press *Commit* to save your changes. Now your published Google Sheet is linked to your published Leaflet map code.
4. Also, let's paste your Google Sheets browser address in a second place to keep track of it. In your GitHub repo, click to open the README.md file, click the pencil symbol to edit it, and paste *your* Google Sheet browser address to replace the existing address. Scroll down and commit to save your changes.

TODO: Insert image here

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 and 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

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: Search Font Awesome Icons and insert any standard icon name such as `school` or `bus`, or leave blank for no icon inside the marker. To create your own custom icon, see further below.
- Marker Color: Search W3Schools Color Names and insert any standard name such as `blue` or `darkblue`. Or insert a web color code such as `#775307` or `rgba(200,100,0,0.5)`.
- 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.
- Image: Insert link to an external image link, or upload a small image to the `media` folder in your GitHub repo and add its pathname, such as `media/trinity-college.jpg`. TODO: add this to GSheet after code update
- Description: Add text to appear in the marker pop-up window. You may include HTML web links in this format: `link text`. If a map link should open in a new browser tab, set the `target` attribute to `_blank`. Learn about HTML syntax at W3Schools.

The next three columns—Location, Latitude, and Longitude—help to place your marker on the map. Although the Leaflet code 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 that we introduced in Chapter 3 to convert addresses to x- and y-coordinates in a separate tab of your Google Sheet, and paste the results of those three columns into Location, Latitude, and Longitude.

Optional: You can display a table of viewable markers at the bottom of your map, as shown in Figure 12.6. 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.

Optional: You can create your own custom marker, such as the Trinity College Bantam mascot in the demo. Use an image editing tool to create a square image, 64 x 64 pixels or smaller, with a transparent background. Save it in PNG format with a filename using all lower-case letters with no spaces. Upload the image to the `markers` folder in your GitHub repo. In the Marker Icon column, set the

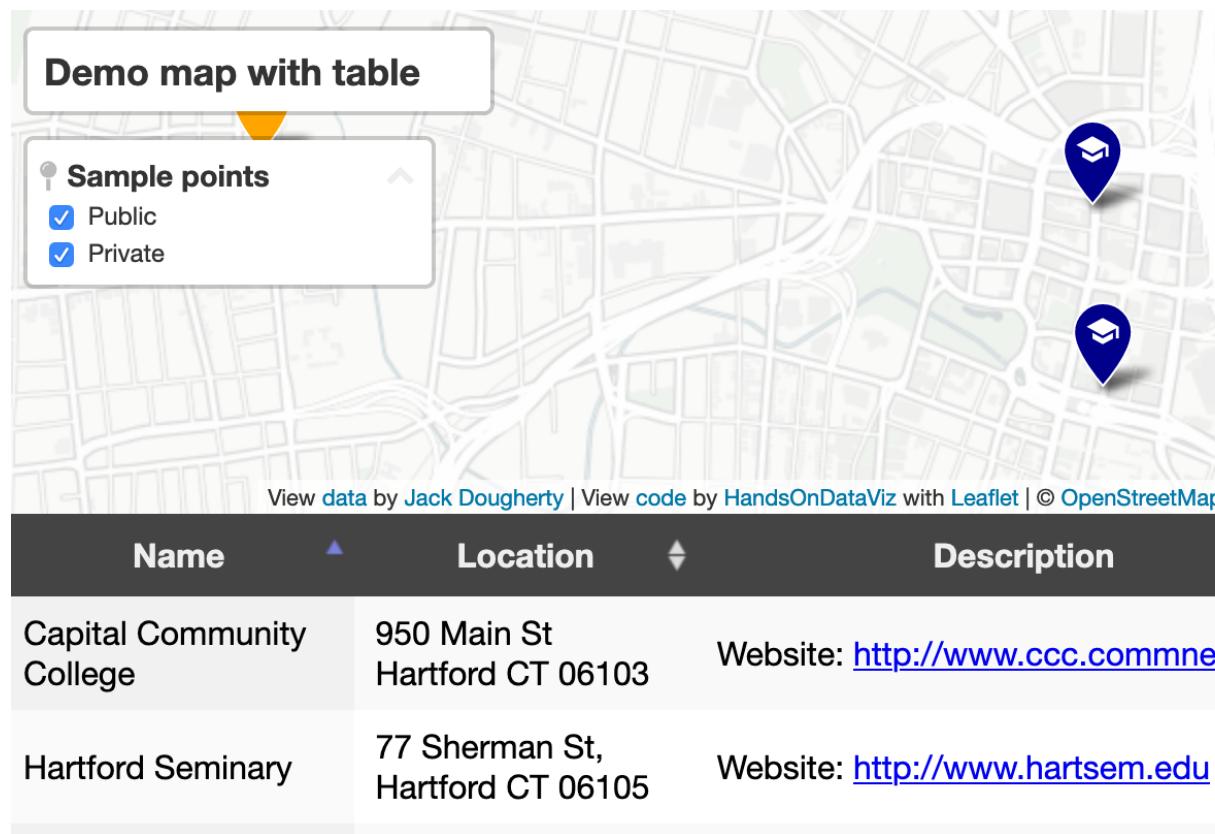


Figure 12.6: One option is to display a table of viewable markers at the bottom of your map.

pathname in this format: `markers/custom-trinity-64.png`. In the Custom Size column, set to **64x64** or similar.

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 in Chapter 13.

After you've prepared your GeoJSON data, name the files using all lower-case characters and no spaces, and upload them into the `geometry` 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 *Polyline*s tab, enter the GeoJSON URL pathname to the file you uploaded to your GitHub repo, such as `geometry/polylines-bike-lanes.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 `geometry/polylines-town-population.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 7, 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 dropdown menu to *Duplicate* the sheet, and name additional sheets in this format: *Polygons1*, *Polygons2*, etc.

Finalize Your Map

Now you're ready to finalize your map. Read the options below and choose either step G *OR* step H.

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, save each tab as a CSV file and upload them to GitHub. The Leaflet map code will pull data directly from your CSV files, rather than your Google Sheets. And you can *still* edit your CSV files in GitHub. Moving your data from Google Sheets to CSV format is the *best* long-term preservation strategy, because it keeps your map and data together in the same GitHub folder, and removes the risk that your map will break due to an interruption to Google services, as described in Step H.

To move your map data from Google Sheets to CSV format, go to each tab and select *File > Download As* into CSV format, as shown in Figure 12.7, using these file names: **TODO: should we warn to keep the first letter upper-case?**

- Options.csv

- Points.csv
- Polylines.csv
- Polygons.csv (if needed, also use: Polygons1.csv, Polygons2.csv, etc.)
- Notes.csv (or .txt)

Upload all of these CSV files into the main level (or root level) of your GitHub repository. The Leaflet code looks here first for the data, and when it locates them, will pull the data directly from the CSV files into your map, skipping over the Google Sheets. TODO: Confirm this code request is working

H) Get your own Google Sheets API Key to insert into the code

If you wish to keep using your Google Sheets to store your data, go to the chapter section named 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

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 J or K above, since it will stop working if we change our key.

If problems arise, see the Fix Common Mistakes section of the appendix.

TODO: Start again here

Leaflet Storymaps with Google Sheets

TODO: Add intro text

Try it: Explore the map or right-click to view full-screen map in a new tab

The map pulls the point data and settings from a linked Google Sheet, which you can explore below or right-click to view full-screen Sheet in a new tab

Features

- Show map points, text, images, and links with scrolling narrative
- Free and open-source code template, built on Leaflet and linked to Google Sheets
- Fork the code and host your live map on the web for free with GitHub Pages

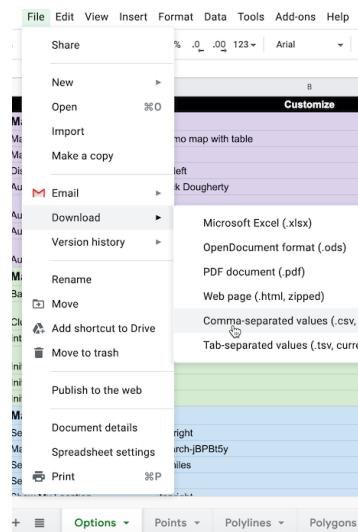


Figure 12.7: One way to finalize your map is to download each Google Sheets tab as a CSV file.

- Geocode location data with US Census or Google, using script inside the Google Sheet
- Easy-to-modify data and map options in Google Sheet tabs or uploaded CSV files
- Responsive design resizes your maps to display inside most mobile devices

Create Your Own

- A) Fork (copy) the code template and publish your version with GitHub Pages
- B) File > Make a Copy of Google Sheet template, Share, and File > Publish
- C) Paste your Google Sheet URL in two places in your GitHub repo
- C2) NEW: Create a free Google Sheets API key to paste into the code
- D) Modify your map settings in the Options tab and test your live map
- E) Geocode locations in the Points tab

To solve problems, see the Fix Common Mistakes section of the appendix.

A) Fork (copy) the code template and publish your version with GitHub Pages

Before you begin, this tutorial assumes that you:

- have a free Google Drive account, and learned the File > Make a Copy in Google Sheets tutorial in this book
- have a free GitHub account, and understand concepts from the Edit and Host Code with GitHub chapter in this book

- 1) Right-click to open this GitHub code template in a new tab: <https://github.com/handsondataviz/leaflet-storymaps-with-google-sheets>
- 2) In the upper-right corner of the code template, sign in to your free GitHub account
- 3) In the upper-right corner, click Fork to copy the template (also called a code repository, or repo) into your own account. The web address (URL) of the new copy in your account will follow this format:

<https://github.com/USERNAME/leaflet-storymaps-with-google-sheets>

Reminder: You can only fork a GitHub repo **one time**. If needed, see how to make a second copy in the Create a New Repo in GitHub chapter in this book.

- 4) In your new copy of the code repo, click on Settings, scroll down to the GitHub Pages area, select Master, and Save. This publishes your code to a live map on a public website that you control.
- 5) Scroll down to GitHub Pages section again, and copy the link to your published web site, which will follow this format:

`https://USERNAME.github.io/leaflet-storymaps-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, written in easy-to-read Markdown code.
- 8) Delete the link to the current live site, and paste in the link to YOUR site. Scroll down and Commit to save your edits.
- 9) On your repo main page, right-click the link to your live map to open in a new tab. **Be patient** during busy periods on GitHub, when your website may take up to 1 minute to appear the first time.

B) File > Make a Copy of Google Sheet template, Share, and File > Publish

- 1) Right-click to open this Google Sheets template in a new tab: `https://docs.google.com/spreadsheets/d/1AO6XHL_0JafWZF4KEejkdDNqfuZWUk3SINlQ6MjlRFM/`
- 2) Sign into your Google account
- 3) File > Make a Copy of the Google Sheet template to your Google Drive
- 4) Click the blue Share button, click Advanced, click to change Private to Anyone with the link > Can View the Sheet. This will make your public data easier to view in your map.
- 5) File > Publish the Link to your Google Sheet to the public web, so the Leaflet map code can read it.
- 6) At the top of your browser, copy your Google Sheet web address or URL (which usually ends in ...XYZ/edit#gid=0). Do NOT copy the published URL (which usually ends in ...XYZ/pubhtml).

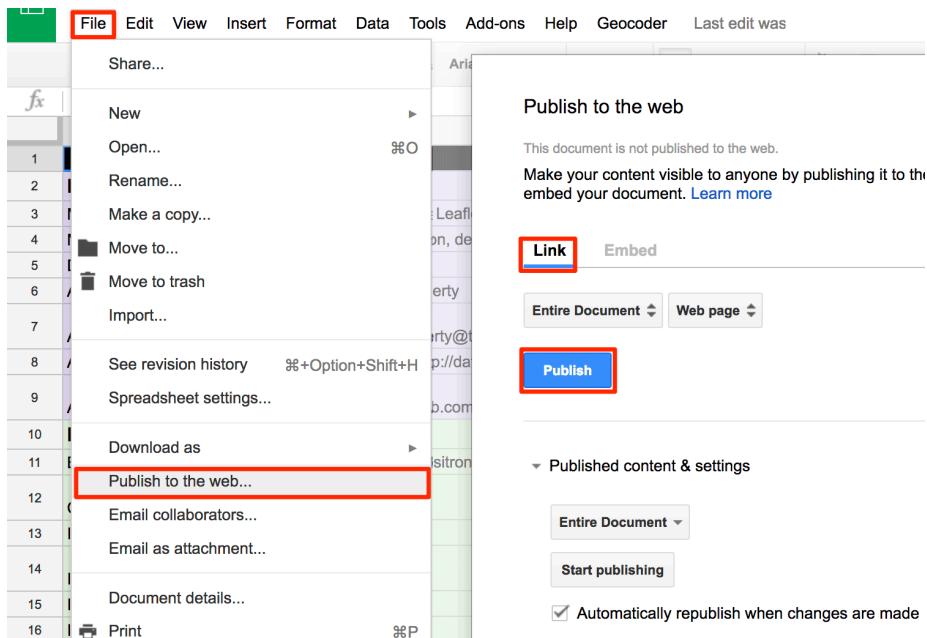


Figure 12.8: Screenshot: File > Publish the link to your Google Sheet

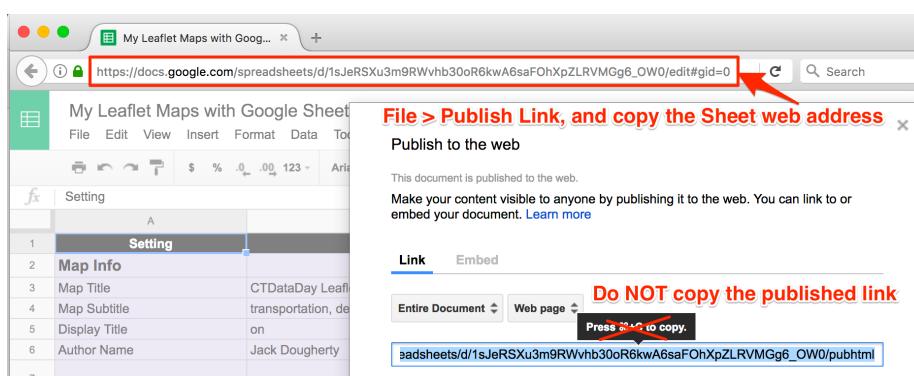


Figure 12.9: Screenshot: Copy the Google Sheet URL, not the Published URL

C) Paste your Google Sheet URL in two places in your GitHub repo

- 1) First, connect your Google Sheet directly to your Leaflet Map code. In your Github code repo, click to open this file: `google-doc-url.js`
- 2) Click the pencil symbol to edit the file.
- 3) Paste your Google Sheet URL into the code to replace the current URL. Do not delete the single-quotation marks or semicolon.
- 4) Scroll to bottom of page and press Commit to save your changes. Now the Leaflet Map code can locate your published Google Sheet.
- 5) Next, let's paste your Google Sheet URL in a second place to keep track of it. Go to the `README.md` file in your GitHub repo, click to open and edit, and paste your Google Sheet web address to replace the existing link near the top. Commit to save your changes.

D) Modify your map settings in the Options tab and test your live map

In the top-level of your GitHub repo, test the new links to your map and your Google Sheet to make sure they work and point to your versions.

** TO DO - redo GIF **

In your linked Google Sheet, go to the Options Tab and modify these items:

- 1) Map Title – insert your own title
- 2) Map Subtitle – insert your own version
- 3) Author Name – insert your own name, or first name, or initials (will be public)
- 4) Author Email or Website – insert your own (will be public), or delete the current name to make it blank

Open the link to your live map in a new browser tab and refresh to see your changes.

E) Geocode locations and customize new markers in the Points tab

In your new map, our next goal is to add and modify the appearance of a new set of point markers, based on new addresses that you will enter and geocode.

In the Points tab of your Google Sheet:

- 1) Do NOT delete or rename any column headers. However, you have the option to add new column headers to display in your map table.
- 2) Geocode your new data inside your Google Sheet by dragging your cursor to select 6 columns of data: Location - Latitude - Longitude - Found - Quality - Source
- 3) In the Geocoder menu that appears in this Google Sheet template, select one of the geocoding services. If one service cannot locate your data, try the other. Always inspect the accuracy of the Found column.

Open the link to your live map in a new browser tab and refresh to see your changes. If your new markers appear correctly, then delete the existing rows that came with this template.

TODO

Add documentation for new features added in 2020

Add links to your text in the Google Sheet

Add line breaks to your text in the Google Sheet

TODO to code: Add Scroll Down text and symbol after the subtitle

Markers

I added a new column to the Chapter tab called “Marker”. It has a drop-down with currently three options: Numerated (defaults to that, even if empty value), Plain (with no number), and No marker. The latter is what you want. It can be potentially extended to colours, types of markers, etc. <https://github.com/handsondataviz/leaflet-storymaps-with-google-sheets/blob/master/scripts/storymap.js#L121-L131>

Overlay GeoJSONs

I added two columns, GeoJSON Overlay with the URL to the GeoJSON, and GeoJSON Feature Properties, which is CSS that defines style of features. List the styles separated by semicolon, and no quotation marks required. Eg `fillColor: orange; weight:2; opacity: 0.5, color: red, fillOpacity: 0.1` In the code, you will see two vertical lines: they mean “or”. If the value of the left-most expression is not undefined, it uses it. If not, it keeps moving to the right until there is a value that is not an empty string. For example, <https://github.com/handsondataviz/leaflet-storymaps-with-google-sheets/blob/master/scripts/storymap.js#L310> `color: feature.properties.COLOR || props.color || 'silver'`,

Will first attempt to extract the color from the COLOR property of each geoJson feature (useful for choropleth). If not found, it tries the

GeoJSON Feature Properties “color”. If that is not set, it uses silver. <https://github.com/handsondataviz/leaflet-storymaps-with-google-sheets/blob/master/scripts/storymap.js#L288-L316>

Data in local CSV files

If googleDocURL variable does not exist (eg you delete the file) or is an empty string, it reads two spreadsheets: Options.csv and Chapters.csv from the /csv folder. Otherwise, it reads from the google sheet. <https://github.com/handsondataviz/leaflet-storymaps-with-google-sheets/blob/master/scripts/storymap.js#L13-L35> When data is read from a .CSV, it links that in the attribution (<https://github.com/handsondataviz/leaflet-storymaps-with-google-sheets/blob/master/scripts/storymap.js#L393-L396>)

Modify your Style Sheet

To adjust title size: In GitHub, go to css/styles.css file, scroll all the way to the bottom, and adjust font-size values (or just use the links below). See your title around line 170, and change font-size up or down....

To add a horizontal line, you need to be a bit creative (see screenshot attached)! Break down text in your Description with the following code for the horizontal line:

```
<span style="display:block;width:100%;height:1px;background-color:silver; margin: 20px 0;"></span>
```

When you copy-paste this snippet, the straight quotation marks do not turn into curly marks, otherwise it won't work.

Learn more: To solve problems, see 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.

Google requires a API (application program interface) key to allow your computer code to read data from your Google Sheets, beginning with version 4 in September 2020. (If you created your own Leaflet Maps or Storymaps with Google Sheets using our template prior to September 2020, and you want to continue to pull data from your Google Sheet, you'll need to update your code to make sure it keeps working **TODO: explain pull request.**) Google requires this API key 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 API key for free 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 12.10. Or alternatively, go to the upper-left drop-down menu to *Select a project > New project*.

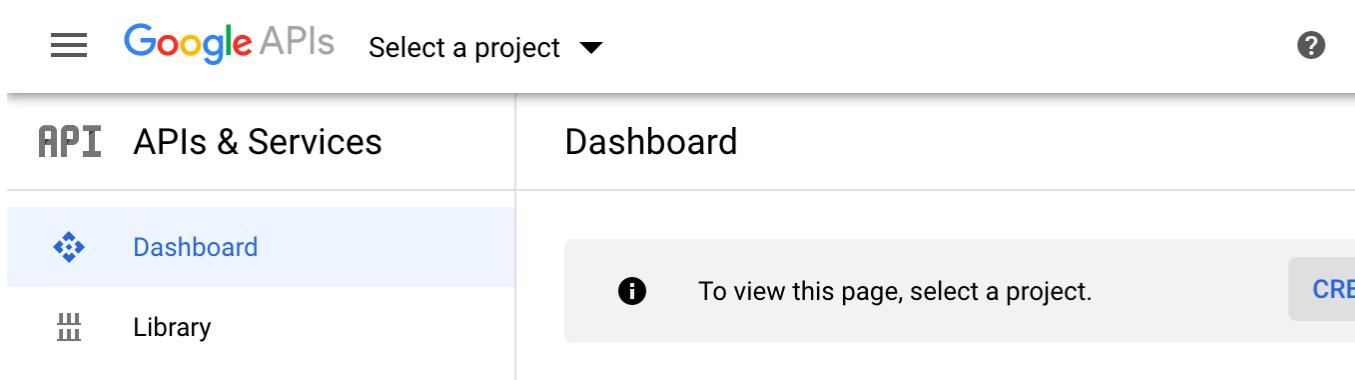


Figure 12.10: Select *Create a Project* or use the menu to select a new project.

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 12.11.

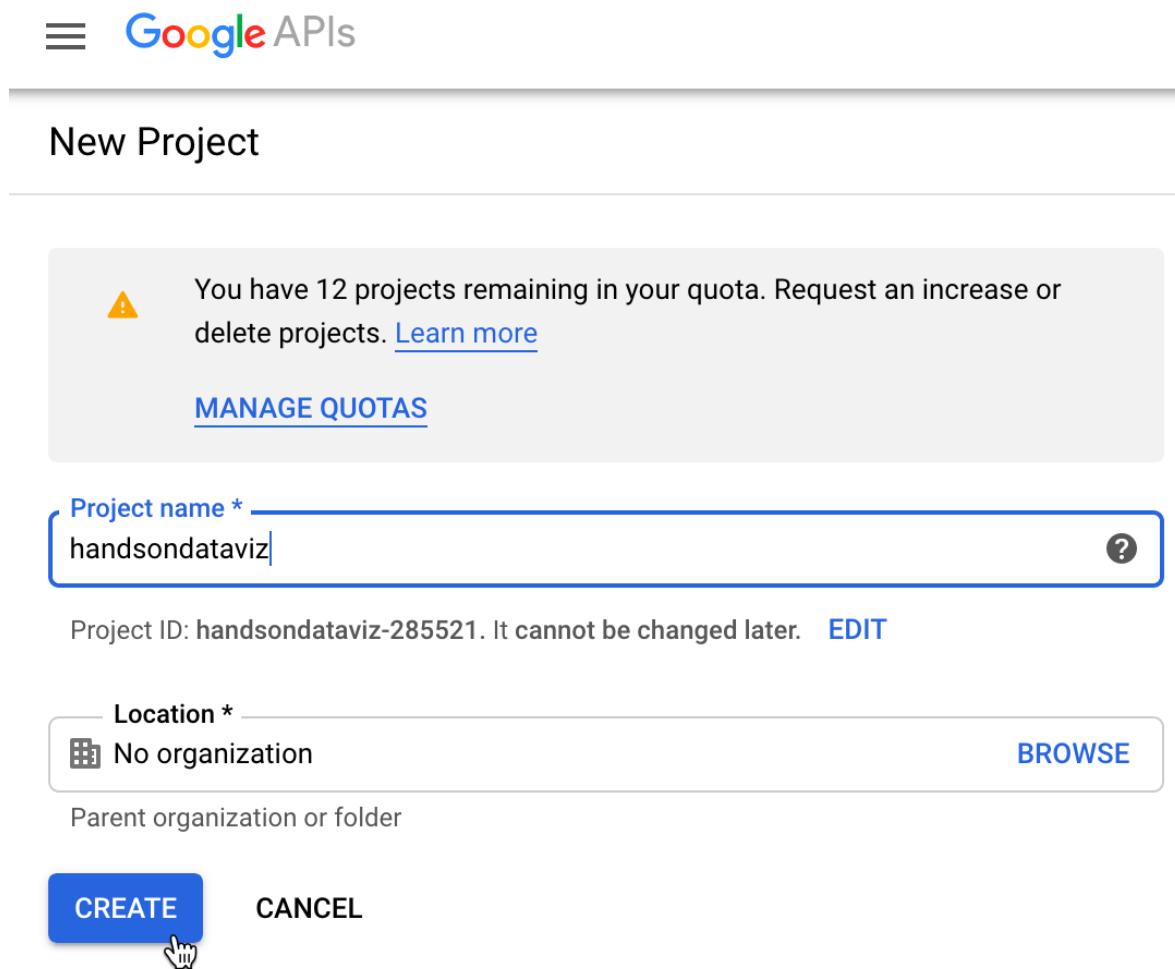


Figure 12.11: Give your project a meaningful short name.

4. In the next screen, press the *+ Enable APIs and Services* at the top of the menu, as shown in Figure 12.12. Make sure that your new project name appears near the top.

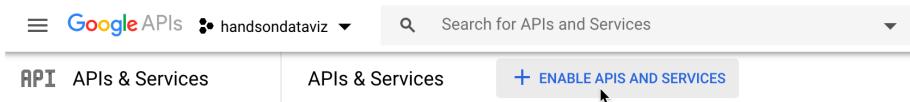


Figure 12.12: Press the *+ Enable APIs and Services* button.

5. In the next screen, enter *Google Sheets* into the search bar, and select this result, as shown in Figure 12.13.

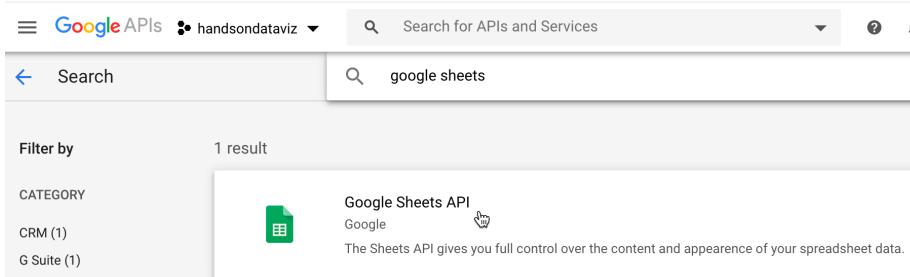


Figure 12.13: Search for *Google Sheets* and select this result.

6. In the next screen, select the *Enable* button to turn on the Google Sheets API for your project, as shown in Figure 12.14.
7. In the left sidebar menu, click *Credentials*, then click *+ Create Credentials* and select *API key*, as shown in Figure 12.15.
8. In the next screen, the console will generate your API key. Copy it, then press *Restrict key*, as shown in Figure 12.16.
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 12.17.

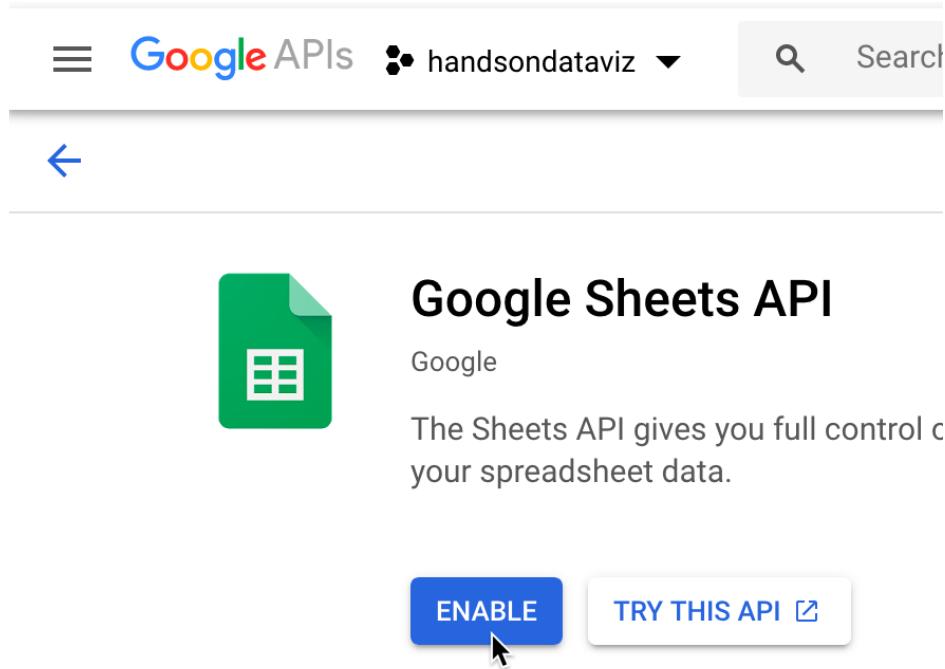


Figure 12.14: Select the *Enable* button for Google Sheets API.

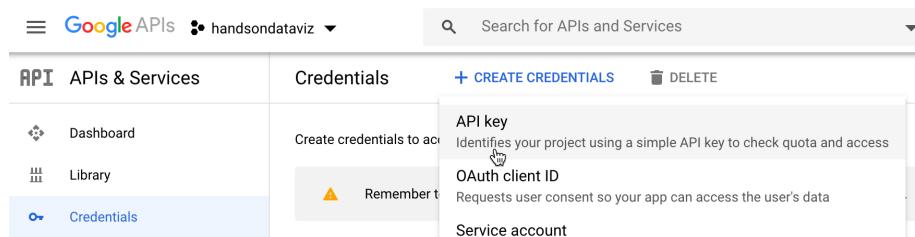


Figure 12.15: Select *Credentials - Create Credentials - API key*.

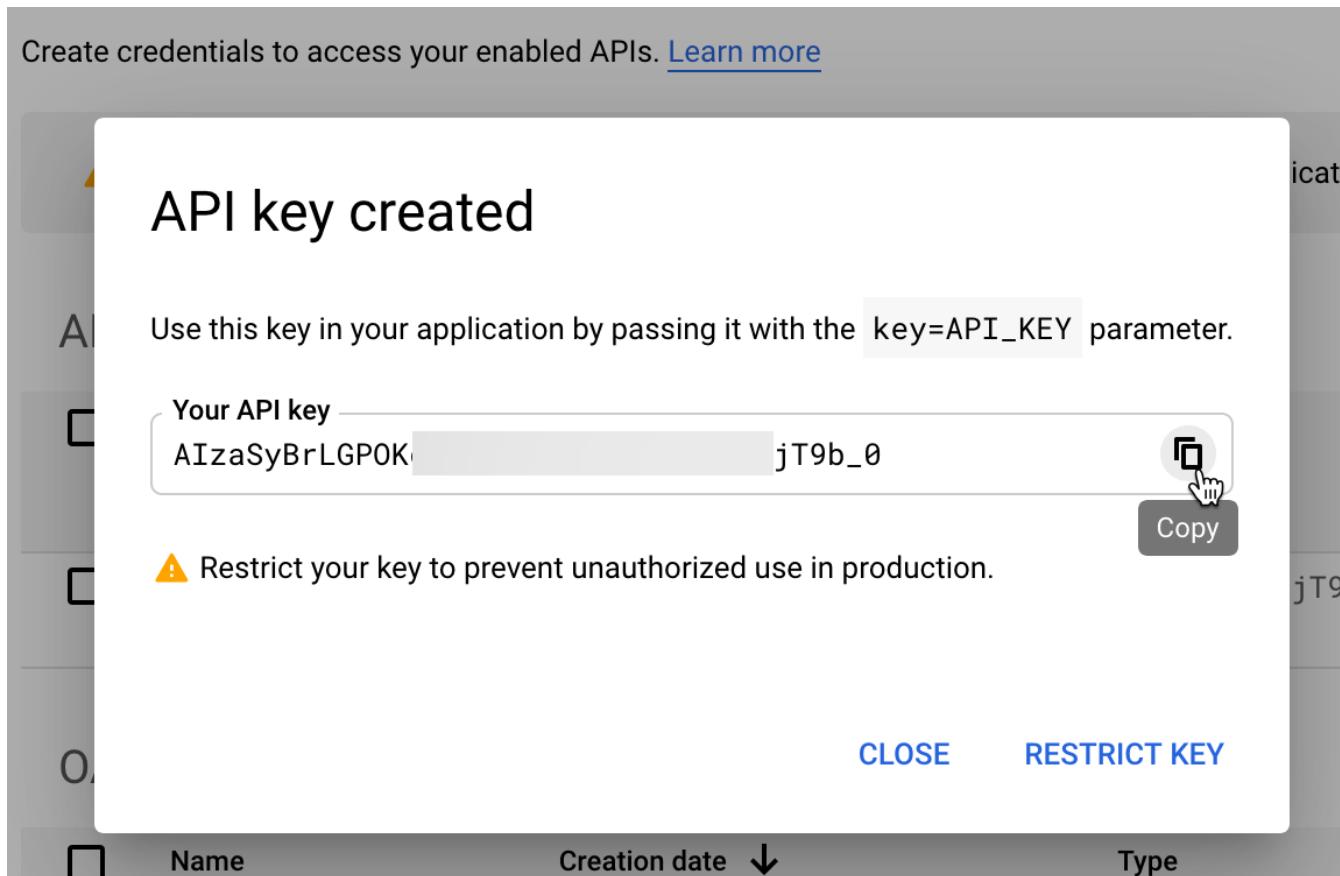


Figure 12.16: Copy your API key and press *Restrict key*.

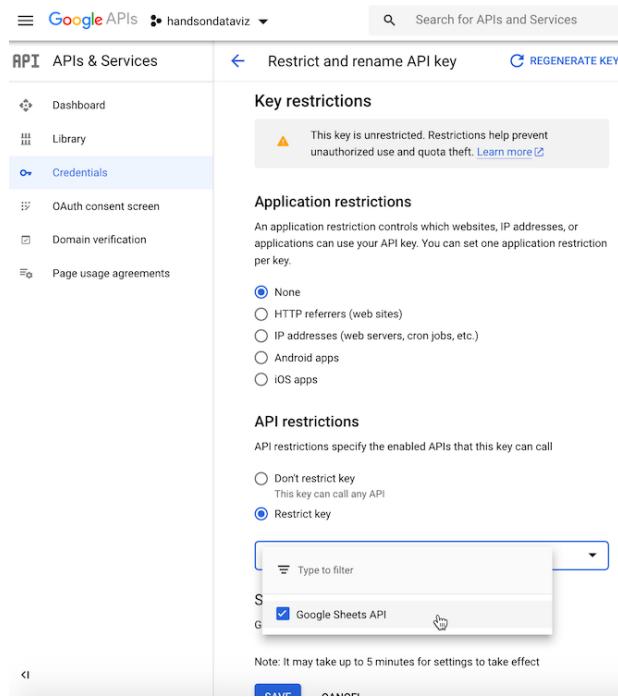


Figure 12.17: Choose *API restrictions* - *Restrict key* - *Google Sheets API*

```
Executable File | 2 lines (2 sloc) | 179 Bytes
1 var googleDocURL = 'https://docs.google.com/spreadsheets/d/1fH16jlqfwUhUzN-oBOK4ZRVC EqHQLSf6XU5qquiWcg/edit#gid=0';
2 var googleApiKey = 'AIzaSyBhIAgJJfk87WhY';
```

Raw Blame

Figure 12.18: Paste in *your* Google Sheets API key to replace *our* key.

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 12.18. 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.

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.

Leaflet Maps with CSV Data

TODO: REWRITE this to serve as a more advanced version using repo <https://github.com/HandsOnDataViz/leaflet-map-csv> rather than leaflet-map-simple (used in ch8)

This tutorial introduces more sophisticated Leaflet map code templates (<http://leafletjs.com>) that you can modify and host online with GitHub in your browser (<http://github.com>). You will learn how to:

- A) Fork (copy) Leaflet template to your GitHub account
- B) Publish your live map to public web with GitHub Pages
- C) Modify your map title and add layer controls
- D) Geocode addresses in a Google Sheet and upload points from data.csv

Code templates help us to move beyond the limits of drag-and-drop web mapping services (such as Google MyMaps) and to create more customized visualizations on a web server that you control. Before you begin, learn the broad concepts in the chapter introduction Edit and Host Code with GitHub. If you

have problems with this tutorial, go to the Fix Common Mistakes section of the appendix.

TODO: add demo, remove unnecessary basic steps from below (covered in prior chapter)

Video

A) Fork (copy) Leaflet template to your GitHub account

Before you begin, sign up for a free GitHub account: <http://github.com>

- 1) Right-click to open this GitHub code template in a new tab: <https://github.com/handsondataviz/leaflet-map-csv>
- 2) In the upper-right corner of the code template, sign in to your free GitHub account
- 3) In the upper-right corner, click Fork to copy the template (also called a code repository, or repo) into your GitHub account. The web address (URL) of the new copy in your account will follow this format:

`https://github.com/USERNAME/REPOSITORY`

Reminder: You can only fork a GitHub repo **one time**. If needed, see how to make a second copy in the Create a New Repo in GitHub chapter in this book.

B) Publish your live map to public web with GitHub Pages

- 4) In your new copy of the code repo, click on Settings, scroll down to the GitHub Pages area, select Master, and Save. This publishes your code template to a live map on a public website that you control.
- 5) Scroll down to GitHub Pages section again, to select and copy the link to your published web site, which will follow this format:

`https://USERNAME.github.io/REPOSITORY`

- 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, written in easy-to-read Markdown code.
- 8) Delete the link to the current live site, and paste in the link to your site. Scroll down and Commit to save your edits.
- 9) On your repo main page, right-click on the link to your published site to open in a new tab. **Be patient** during busy periods, because your website may take up to 1 minute to appear the first time.

C) Modify your map title and add layer controls

- 10) Go back to your browser tab for your code repo. Click on the index.html file (which contains the map code), and click the pencil icon to edit it.
- 11) Explore the map code, which contains HTML, CSS, and JavaScript. Look for sections that begin with “EDIT” for items that you can easily change. Scroll down to Commit your changes.
- 12) Go to your live website browser tab and refresh the page to view your edits. **Be patient** during busy periods, when some edits may take up to 1 minute to appear.
- 13) To change your map title in the index.html file, click the pencil symbol (to edit) and go to lines 23-25. Replace “EDIT your map title” with your new title:

TODO: decide if these triple backtic snippets will stay, or if they will throw errors into Markdown output

```
<!-- Display the map and title with HTML division tags -->
<div id="map-title">EDIT your map title</div>
<div id="map"></div>
```

- 14) To change your initial map zoom level, edit the index.html file and go to line 33. The zoom range for this map is from 1 (max zoom out) to 18 (max zoom in).

```
// Set up initial map center and zoom level
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
});
```

- 15) To change the default basemap, edit lines 46 and 52 to delete “.addTo(map)” from the Carto light layer, then add it to the Stamen colored terrain layer. DO NOT erase the semicolons!

Your original code looks like this (scroll to right to see all):

```
/* Carto light-gray basemap tiles with labels */
var light = L.tileLayer('https://cartodb-basemaps-{s}.global.ssl.fastly.net/light_all/{z}/{x}/{y}.png',
  attribution: '&copy; <a href="http://www.openstreetmap.org/copyright">OpenStreetMap</a>, &copy; Carto'
).addTo(map); // EDIT - insert or remove ".addTo(map)" before last semicolon to display by default
```

```
// controlLayers.addBaseLayer(light, 'Carto Light basemap');
/* Stamen colored terrain basemap tiles with labels */
var terrain = L.tileLayer('https://stamen-tiles.a.ssl.fastly.net/terrain/{z}/{x}/{y}')
    attribution: 'Map tiles by <a href="http://stamen.com">Stamen Design</a>, under <a href="http://creativecommons.org/licenses/by-sa/2.0/">CC-BY-SA</a>';
}); // EDIT - insert or remove ".addTo(map)" before last semicolon to display by default
// controlLayers.addBaseLayer(terrain, 'Stamen Terrain basemap');
```

After you edit the code, it should look like this (scroll to right to see all):

```
/* Carto light-gray basemap tiles with labels */
var light = L.tileLayer('https://cartodb-basemaps-{s}.global.ssl.fastly.net/light_all/{z}/{x}/{y}.png')
    attribution: '&copy; <a href="http://www.openstreetmap.org/copyright">OpenStreetMap</a>';
}); // EDIT - insert or remove ".addTo(map)" before last semicolon to display by default
// controlLayers.addBaseLayer(light, 'Carto Light basemap');
/* Stamen colored terrain basemap tiles with labels */
var terrain = L.tileLayer('https://stamen-tiles.a.ssl.fastly.net/terrain/{z}/{x}/{y}.png')
    attribution: 'Map tiles by <a href="http://stamen.com">Stamen Design</a>, under <a href="http://creativecommons.org/licenses/by-sa/2.0/">CC-BY-SA</a>';
}).addTo(map); // EDIT - insert or remove ".addTo(map)" before last semicolon to display by default
// controlLayers.addBaseLayer(terrain, 'Stamen Terrain basemap');
```

- 16) To add a control panel that turns on/off map layers, delete the code comment symbols (//) that appear in front of lines 38-41, 47, and 53 to activate these sections. When you remove code comments in GitHub, the color changes from gray text (inactive code) to colored text (active code). After you remove the code comments, your file should look like this (scroll to right to see all):

```
/* Control panel to display map layers */
var controlLayers = L.control.layers( null, null, {
    position: "topright",
    collapsed: false
}).addTo(map);

/* Carto light-gray basemap tiles with labels */
var light = L.tileLayer('https://cartodb-basemaps-{s}.global.ssl.fastly.net/light_all/{z}/{x}/{y}.png')
    attribution: '&copy; <a href="http://www.openstreetmap.org/copyright">OpenStreetMap</a>';
}); // EDIT - insert or remove ".addTo(map)" before last semicolon to display by default
controlLayers.addBaseLayer(light, 'Carto Light basemap');
/* Stamen colored terrain basemap tiles with labels */
var terrain = L.tileLayer('https://stamen-tiles.a.ssl.fastly.net/terrain/{z}/{x}/{y}.png')
    attribution: 'Map tiles by <a href="http://stamen.com">Stamen Design</a>, under <a href="http://creativecommons.org/licenses/by-sa/2.0/">CC-BY-SA</a>';
}).addTo(map); // EDIT - insert or remove ".addTo(map)" before last semicolon to display by default
controlLayers.addBaseLayer(terrain, 'Stamen Terrain basemap');
```

- 17) To change one point on the map, you could edit the latitude and longitude coordinates of the single marker in lines 55-57. To find coordinates for any location and to learn more, go to <http://www.latlong.net>

```
/* Display a blue point marker with pop-up text */
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
```

But a better way to display several points is to remove the code comment symbols (//) in front of lines 60-69 to activate this section of code, which pulls map points from the data.csv file in your GitHub repository. After your edits, this section should look like this (scroll right to see all):

```
/* Upload Latitude/Longitude markers from data.csv file, show Title in pop-up, and override init...
var customLayer = L.geoJson(null, {
  onEachFeature: function(feature, layer) {
    layer.bindPopup(feature.properties.Title);
  }
});
var runLayer = omnivore.csv('data.csv', null, customLayer)
.on('ready', function() {
  map.fitBounds(runLayer.getBounds());
}) .addTo(map);
controlLayers.addOverlay(customLayer, 'Markers from data.csv');
```

D) Geocode addresses in Google Sheet and upload points from data.csv

- 18) A better way to display multiple points on your map is to prepare and upload a new data.csv file to your GitHub repository. First, right-click to open this Google Sheets template in a new tab: Leaflet Maps Simple data points with Geocoder
- 19) Since this sheet is view-only, you cannot edit it. Instead, sign in to your Google account in the upper-right corner.
- 20) Go to File > Make a Copy, which will save a duplicate version to your Google Drive, which you can edit.
- 21) In your copy of the Google Sheet, select any cells and press Delete on your keyboard to erase contents. Type new titles and addresses into columns A and B.
- 22) To geocode your new addresses (which means converting them into latitude and longitude coordinates), select all of the contents across 6 columns, from Address (B) to Source (G).

- 23) Go to the Geocoder menu that appears in this special Google Sheet template, and select any service, such as US Census (for US addresses) or Google Maps. The first time you run the geocoder, the script will ask for permission.
- 24) After you have geocoded your addresses, go to File > Download As > Comma-separated values (.CSV format) to save the file to your computer.
- 25) In your computer, right-click the downloaded file to rename it to: data.csv
- 26) In your GitHub repository, click Upload Files, then drag-and-drop your new data.csv file, and Commit to upload it. Go to your live map browser tab and refresh to view changes. **Be patient* during busy periods, when some edits may take up to 1 minute to appear.**

Leaflet Maps with Open Data API

TODO:

- Note this new title and URL, which is more general than the older title and “leaflet-maps-with-socrata” URL
- Blend in other section below this one
- Update the example to pull map data from a continuously updated map, since the current example has not been updated since 2018
- Decide if there’s anything useful to borrow from the other example repo, such as a non-Socrata endpoint?: <https://github.com/HandsOnDataViz/leaflet-data-apis>
- write intro to connect more directly to the Open Data section in ch 3, and describe open data APIs in general, with Socrata API serving as a convenient example.

Source: Current Class 1 - Class 4 Food Establishments, City of Hartford

Why pair Leaflet maps with Socrata data?

Leaflet, a friendly and flexible open-source code library for creating interactive web maps, plays nicely with Socrata, an open data platform used by several government agencies and organizations. Benefits of pairing Leaflet and Socrata:

- Although the Socrata data platform includes built-in visualization tools for anyone to create charts and maps, Leaflet gives you more control over your map design. Furthermore, Leaflet allows you to create maps that bring together data from both Socrata and non-Socrata sources.

- Socrata datasets include an API (application program interface) endpoint, in the form of a web address. This endpoint enables other computers to easily access the most recent data online, instead of a static version that was manually downloaded.
- Newer Socrata datasets that include locations (such as latitude and longitude coordinates) also provide endpoints in GeoJSON format. Since Leaflet maps easily process GeoJSON data, only a few lines of code are required.
- However, Socrata GeoJSON endpoints do not currently support “real-time” data, such as up-to-the-minute locations of public transportation, etc. In these cases, you may need to access data through a provider other than Socrata, most likely in a different format, which may require more coding skills.

About Socrata API endpoints

Go to any Socrata open data platform, find a dataset, and click the API tab. As an example, you can use City of Hartford’s Police Incidents dataset.



Figure 12.19: Police Incidents dataset on Hartford Open Data portal

Copy the API endpoint. The default version is JSON.

If you’re new to APIs, test the endpoint by pasting it into your browser address line. Ideally you would see a formatted JSON view (use Chrome or Firefox for better results).

If your browser does not support JSON view, you will see the raw JSON stream only, like the one shown below.

```

JSON Raw Data Headers
Save Copy Collapse All Filter JSON
▶ 8: {…}
▶ 9: {…}
▶ 10: {…}
▶ 11:
  case_number: "5000007"
  date: "2005-01-01T00:00:00.000"
  time_24hr: "0030"
  address: "CHURCH ST & TRUMBULL ST"
  ucr_1_category: "32* - PROPERTY DAMAGE ACCIDENT"
  ucr_1_description: "PROP DAM ACC"
  ucr_1_code: "3224"
  ucr_2_category: "23* - DRIVING LAWS"
  ucr_2_description: "FOLL TOO CLOSE"
  ucr_2_code: "2334"
  neighborhood: "DOWNTOWN"
  ▶ geom: {…}
  :@computed_region_ugzy_yqsh: "19"
  :@computed_region_35zh_8fiz: "10"
  :@computed_region_2vdc_22if: "15050"
  :@computed_region_haf6_6xye: "1041"
▶ 12: {…}
▶ 13: {…}
▶ 14: {…}
▶ 15: {…}
▶ 16: {…}

```

Figure 12.20: Formatted JSON example in Firefox

```

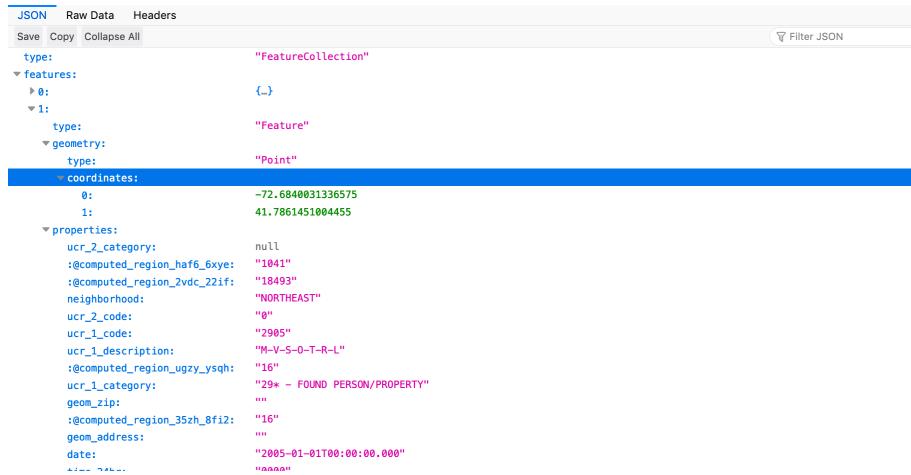
[{"case_number": "9810396", "date": "2005-01-01T00:00:00.000", "time_24hr": "0000", "address": "56 VINE ST", "ucr_1_category": "55* - REPORT-RELATED", "ucr_1_description": "CASE DRAWN IN ERROR", "ucr_1_code": "5520", "ucr_2_code": "0", "neighborhood": "UPPER ALBANY", "geom": {"latitude": "41.7809708152311", "longitude": "-72.6881141026066"}, "human_address": {"address": "", "city": "", "state": "", "zip": ""}, "@computed_region_ugzy_yqsh": "15", "@computed_region_2vdc_22if": "18493", "@computed_region_haf6_6xye": "1041", {"case_number": "5000007", "date": "2005-01-01T00:00:00.000", "time_24hr": "0000", "address": "14 GILMAN ST", "ucr_1_category": "29* - FOUND PERSON", "ucr_1_description": "UCR 1 DESCRIPTION", "ucr_1_code": "1909", "ucr_2_code": "0", "neighborhood": "DOWNTOWN", "geom": {"latitude": "41.7861451004455", "longitude": "-72.6840931336575"}, "human_address": {"address": "", "city": "", "state": "", "zip": ""}, "@computed_region_ugzy_yqsh": "16", "@computed_region_35zh_8fiz": "16", "@computed_region_2vdc_22if": "18493", "@computed_region_haf6_6xye": "1041", {"case_number": "5000024", "date": "2005-01-01T00:00:00.000", "time_24hr": "0000", "address": "115 ASYLUM ST", "ucr_1_category": "55* - REPORT-RELATED", "ucr_1_description": "CASE DRAWN IN ERROR TO", "ucr_1_code": "5510", "ucr_2_code": "0", "neighborhood": "DOWNTOWN", "geom": {"latitude": "41.7669464534884", "longitude": "-72.6753377122773"}, "human_address": {"address": "", "city": "", "state": "", "zip": ""}, "@computed_region_ugzy_yqsh": "17", "@computed_region_2vdc_22if": "18493", "@computed_region_haf6_6xye": "1041", {"case_number": "5000009", "date": "2005-01-01T00:00:00.000", "time_24hr": "0000", "address": "127 IRVING ST", "ucr_1_category": "34* - OTHER ACCIDENT", "ucr_1_description": "OCC-INJ-POLICE", "ucr_1_code": "3448", "ucr_2_code": "0", "neighborhood": "UPPER ALBANY", "geom": {"latitude": "41.7801122575092", "longitude": "-72.6861183820887"}, "human_address": {"address": "", "city": "", "state": "", "zip": ""}, "@computed_region_ugzy_yqsh": "15", "@computed_region_2vdc_22if": "18493", "@computed_region_haf6_6xye": "1041", {"case_number": "5029127", "date": "2005-01-01T00:00:00.000", "time_24hr": "0000", "address": "14 GILMAN ST", "ucr_1_category": "19* - CRIMES AGAINST THE PUBLIC", "ucr_1_description": "SIMPLE TRESPASS", "ucr_1_code": "1909", "ucr_2_category": "24* - MOTOR VEHICLE LAWS", "ucr_2_description": "MISUSE OF PLATE", "ucr_2_code": "3504", "neighborhood": "DOWNTOWN", "geom": {"latitude": "41.7376529965414", "longitude": "-72.678166787566"}, "human_address": {"address": "", "city": "", "state": "", "zip": ""}, "@computed_region_ugzy_yqsh": "21", "@computed_region_35zh_8fiz": "21", "@computed_region_2vdc_22if": "18494", "@computed_region_haf6_6xye": "1041", {"case_number": "5000022", "date": "2005-01-01T00:00:00.000", "time_24hr": "0005", "address": "29 ANNAWAN ST", "ucr_1_category": "19* - CRIMES AGAINST THE PUBLIC", "ucr_1_description": "BREACH-PEACE", "ucr_1_code": "1901", "ucr_2_category": "19* - CRIMES AGAINST THE PUBLIC", "ucr_2_description": "DOMESTIC", "ucr_2_code": "1904", "neighborhood": "BARRY SQUARE", "geom": {"latitude": "41.7492666840371", "longitude": "-72.6754861409539"}, "human_address": {"address": "", "city": "", "state": "", "zip": ""}, "@computed_region_ugzy_yqsh": "17", "@computed_region_2vdc_22if": "18493", "@computed_region_haf6_6xye": "1041", {"case_number": "5000046", "date": "2005-01-01T00:00:00.000", "time_24hr": "0005", "address": "FOOT GUARD PL & HOADLEY PL", "ucr_1_category": "66* LARCENY", "ucr_1_description": "LARC3-FROM M/V", "ucr_1_code": "1909", "ucr_2_category": "35* - MIS- CRIMES AGAINST PROPERTY", "ucr_2_description": "CR MISCHIEF 3*", "ucr_2_code": "3503", "neighborhood": "DOWNTOWN", "geom": {"latitude": "41.70668928111", "longitude": "-72.680617911612"}, "human_address": {"address": "", "city": "", "state": "", "zip": ""}, "@computed_region_ugzy_yqsh": "19", "@computed_region_35zh_8fiz": "19", "@computed_region_2vdc_22if": "15050", "@computed_region_haf6_6xye": "1041", {"case_number": "5000009", "date": "2005-01-01T00:00:00.000", "time_24hr": "0011", "address": "949 ALBANY AV", "ucr_1_category": "5211 - SHOTS FIRED UNCONFIRMED", "ucr_1_description": "SHOTS FIRED UNCONFIRMED", "ucr_1_code": "5211", "ucr_2_code": "0", "neighborhood": "UPPER ALBANY", "geom": {"latitude": "41.7803325207027", "longitude": "-72.6524460975509"}, "human_address": {"address": "", "city": "", "state": "", "zip": ""}, "@computed_region_ugzy_yqsh": "20", "@computed_region_35zh_8fiz": "19", "@computed_region_2vdc_22if": "18485", "@computed_region_haf6_6xye": "1041"

```

Figure 12.21: Unformatted JSON example in Firefox

Test if this Socrata endpoint supports GeoJSON format by changing the extension in the API dropdown menu from **JSON** to **GeoJSON**. GeoJSON format works best with Leaflet because the coding is simpler.

If your endpoint supports GeoJSON format, your browser will display a data stream similar to the one below.



The screenshot shows a JSON object representing a FeatureCollection. It includes features (0 and 1), each with a geometry (Point) and coordinates (-72.6840031336575, 41.7861451004455). The properties section contains various Socrata-specific fields like ucr_2_category, ucr_2_code, and date.

```

{
  "type": "FeatureCollection",
  "features": [
    {
      "type": "Feature",
      "geometry": {
        "type": "Point",
        "coordinates": [
          -72.6840031336575,
          41.7861451004455
        ]
      },
      "properties": {
        "ucr_2_category": null,
        "ucr_2_code": "1841",
        "neighborhood": "NORTHEAST",
        "ucr_1_code": "2905",
        "ucr_1_description": "M-V-S-O-T-R-L",
        "date": "2005-01-01T00:00:00.000"
      }
    }
  ]
}

```

Figure 12.22: Formatted GeoJSON example in Firefox

If your Socrata endpoint only supports JSON format, but includes data columns with latitude and longitude, see other Leaflet examples further below.

Register for Socrata App Token

- Socrata requires developers to register for a free app token at <https://opendata.socrata.com/signup>

Demonstration Maps

GeoJSON endpoint with circle markers and tooltips

- map <https://handsondataviz.github.io/leaflet-socrata/index.html>
- code <https://github.com/handsondataviz/leaflet-socrata/index.html>
- data <https://data.hartford.gov/Public-Health/Current-Class-1-Class-4-Food-Establishments/xkvv-76v8>
- note: location data appears as latitude and longitude coordinates in the `geom` column

- steps to create your own (MORE TODO HERE)
 - select API button, copy endpoint, and change suffix from .json to .geojson
 - copy this Leaflet map template, which includes this key section of code:
 - paste and explain the code

GeoJSON endpoint with simple data filter, default marker styling and pop-up info

- map <https://handsondataviz.github.io/leaflet-socrata/index-geojson-filter>
- code <https://github.com/handsondataviz/leaflet-socrata/>
- data <https://data.ct.gov/Environment-and-Natural-Resources/Agricultural-Commodities-Grown-By-Farmer/y6p2-px98>

Multiple Socrata datasets with Leaflet control layers legend

- map <https://handsondataviz.github.io/leaflet-socrata/index-control-layers.html>
- code <https://github.com/handsondataviz/leaflet-socrata/index-control-layers.html>

Older JSON-only endpoint, with separate columns for latitude, longitude

- map <https://handsondataviz.github.io/leaflet-socrata/index-json.html>
- code <https://github.com/handsondataviz/leaflet-socrata/index-json.html>
- data <https://opendata.demo.socrata.com/Government/Kentucky-Farmers-Market-Map/3bfj-rqn7>

Learn more: - <https://dev.socrata.com/> - <https://github.com/chriswhong/simpleSodaLeaflet>

Thanks to

- Chris Metcalf <https://github.com/chrismetcalf>
- Tyler Klyeklamp <https://data.ct.gov/>

TODO: blend this section into the one above

Pull Open Data into Leaflet Map with APIs { - #leaflet-maps-open-apis } TODO: Decide whether to keep or not. Up to this point in the book, we've built charts and maps using static data that you have downloaded from other sites. But some open data repositories have APIs, or application program interfaces, which means the software that allows computers to communicate with one another. Below is a Leaflet Map template that uses APIs to pull in the most current data from three different open repository platforms: Socrata, Esri ArcGIS Online, and USGS.

Try it: Explore the map below or view full-screen version in a new tab

How it works

- 1) Go to the GitHub repo for the map above: <https://github.com/handsondataviz/leaflet-data-apis>
- 2) Explore the code to see how different APIs work. For example, see the first map overlay, which pulls Connecticut School Directory data from the CT Open Data repository on a Socrata open data platform: <https://data.ct.gov/resource/v4tt-nt9n>
- 3) Inside the open data repo, look for an API button and copy the endpoint.



Figure 12.23: Screenshot: Sample API endpoint in Socrata open data repo

- 4) Paste the endpoint link into your browser, change the suffix from `.json` to `.geojson` and press return. In order to show the endpoint data as points on a map in this simple Leaflet template, the points must already be geocoded inside the open data repo, and the platform must support a

GeoJSON endpoint. In your browser, one sign of success is a long stream of GeoJSON data like this:



Figure 12.24: Screenshot: API endpoint with .geojson suffix in Chrome browser

- 5) In this section of the Leaflet map template, the code includes a jQuery function `$.getJSON` to call the open data endpoint in GeoJSON format: <https://data.ct.gov/resource/v4tt-nt9n.geojson>. It also requires a Socrata app token, and you can get your own token for free at: <https://dev.socrata.com/register>. Each geocoded school in the Socrata data repository is displayed as a blue circle, with data properties (such as: name) in a clickable pop-up.

```
// load open data from Socrata endpoint in GeoJSON format
// with simple marker styling: blue circles
// register your own Socrata app token at https://dev.socrata.com/register
// Connecticut School Directory, CT Open Data, https://data.ct.gov/resource/v4tt-nt9n
$.getJSON("https://data.ct.gov/resource/v4tt-nt9n.geojson?&$$app_token=QVY3I72SVPbxBYI")
  var geoJsonLayer = L.geoJson(data, {
    pointToLayer: function( feature, latlng ) {
      var circle = L.circleMarker(latlng, {
        radius: 6,
        fillColor: "blue",
        color: "blue",
        weight: 2,
        opacity: 1,
        fillOpacity: 0.7
      });
      circle.bindPopup(feature.properties.name + '<br>' + feature.properties.district_name);
      return circle;
    }
  }).addTo(map); // display by default
  controlLayers.addOverlay(geoJsonLayer, 'Public Schools (CT Open Data-Socrata)');
});
```

- 5) Fork a copy of this repo, play with the code, and try to insert GeoJSON endpoints from other open data repositories.

Leaflet Heatmap

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 high-crime areas.

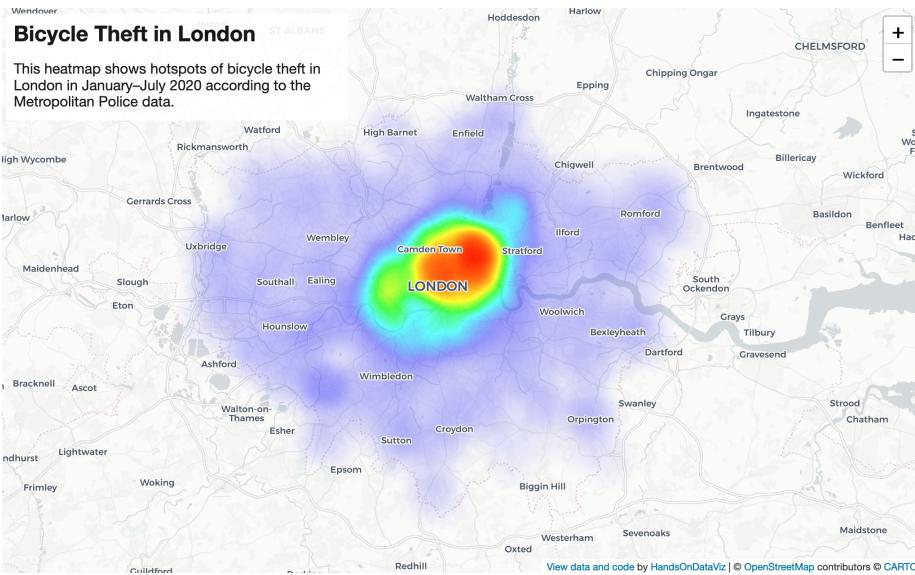


Figure 12.25: Explore a live demonstration of Leaflet Heatmap.

Searchable Map

TODO

Summary

TODO

Chapter 13

Transform Your Map 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. We will then talk about polygons and why you should normalize your data before creating choropleth maps. These map transformations happen inside spreadsheets, so you won't directly deal with map data until you are halfway through the chapter.

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 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 georectify a digitized map to display as a background overlay 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 13.1a), you want darker shades of blue to represent states with more hospitals (as in Figure 13.1b).

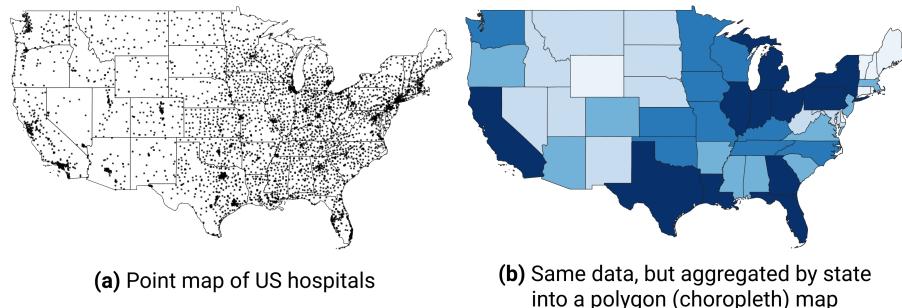


Figure 13.1: You can count addresses by state (or other area) to produce polygon, or choropleth, maps instead of point maps.

First, save the database to your local machine by going to *Export > Download > CSV* of Socrata interface. Figure 13.2 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 data normalization and why showing counts of hospitals per state doesn't really tell a good story.

Normalize Data for Meaningful Maps

Choropleth maps are best when they represent relative, not absolute values. Consider two maps shown in Figure 13.4. They both are about Covid-19 cases in the US states (excluding Alaska and Hawaii) as of June 26, 2020. Figure 13.4a shows total number of recorded cases per state, and Figure 13.4b shows Covid-19 cases adjusted by the state's population. Darker colors represent higher values. Do you notice any differences in spatial patterns?

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. At the top right, there are various navigation links like 'Home', 'Get started', 'Info', 'Developers', and 'Sign In'. Below these are buttons for 'More Views', 'Filter', 'Visualize', 'Export' (which is highlighted in blue), 'Embed', and 'About'. A modal window titled 'Export' is open on the right, listing options: SODA API, OData, Download (selected), Download Geospatial Data (with CSV selected), and other formats like JSON, RDF, RSS, TSV for Excel, XML, KML, and KMZ. The table on the left shows columns for Facility ID, Facility Name, Address, City, State, and a link to 'Export'. The bottom of the page includes a footer with links to feedback, Open Payments Data CMS.gov, and Data.Medicaid.gov.

Figure 13.2: In Socrata, you can export the entire dataset as a CSV.

The screenshot shows a spreadsheet application with a pivot table editor open. On the left, a data table displays 'COUNTA of State' for each US state. The columns are labeled A through D, and the first row contains the header 'State'. The data rows show counts for each 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), and LA (152). To the right of the data is the 'Pivot table editor' interface. It includes sections for 'Suggested', 'Rows', 'Columns', 'Values', and 'Filters'. The 'Rows' section has a red box around it, showing a 'State' field with 'Order: Ascending' and 'Sort by: State'. The 'Values' section also has a red box around it, showing a 'State' field with 'Summarize by: COUNTA' and 'Show as: Default'. Both sections have an 'Add' button at the top right.

Figure 13.3: Use pivot tables in any spreadsheet software to count addresses per area (such as state, county, or zip code).

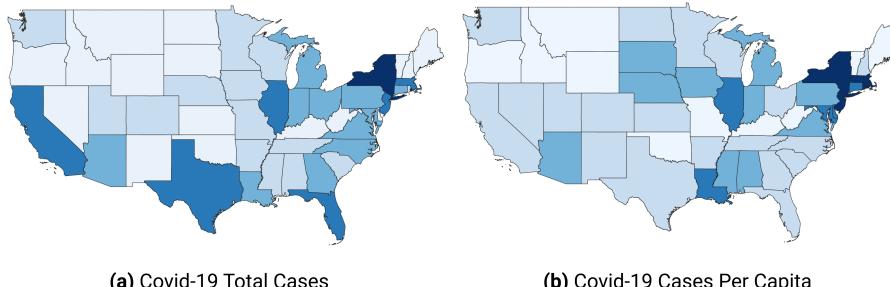


Figure 13.4: Choropleth maps work best with normalized values.

Both maps show Covid-19 data collected by the New York Times and published on GitHub. In the map in Figure 13.4b, we normalized (divided) values by population in each state, according to the 2018 US Census American Community Survey, the most recent data available on the day of writing. We didn't 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 13.4a)? 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 13.4b? 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.

Different ways to normalize data

You can normalize data in many ways, and there is not necessarily one acceptable way of doing it.

One of the most common ways of normalization is deriving “per capita”, or “per person” values. If values are small, such as rare disease cases or lottery winners, they can be presented as “per 1,000” or “per 100,000” people. Divide your quantity by population in that area to derive per capita values.

Choropleth maps work well with percentages. The good news is, humans like percentages too. It is quite natural for us to understand that a 9% unemployment rate means that of 100 people who were willing to work, nine were unable to find a job. To derive a percentage for unemployment, divide the number of unemployed people by labor force size (adult population who are willing and able to work), and multiply by 100.

Unlike counts, most *measured* variables do not need normalization because they belong to a scale. For example, median age (the age of the “middle” person in a population, when sorted from youngest to oldest) can be directly compared among populations. We know that humans live anywhere between 0 and 120 years or so, and we wouldn’t expect median ages to be vastly different from one country to another (maybe twice, but not tenfold). Median incomes, if measured in the same currency, also belong to the same scale and can be compared directly.

How not to normalize values

Absolute values are very important for context. Saying that “20% of blond men living in town X won the lottery” may sound like a catchy headline, but in reality the town has 450 residents, of those 200 are men, and of those only 5 have light hair color. One of those five (and here comes the 20%) was lucky to win the lottery, so technically the headline didn’t lie.

This is, of course, an extreme and comic example, but exaggerations in this spirit are not uncommon. If you want readers to trust you, make sure you are open about total counts when reporting normalized values (such as percentages or per capita values).

Absolute values are important for another reason: behind numbers there are often people, and smaller, normalized values may hide the scale of the problem. Saying that “the unemployment rate is only 5%” is valid, but the 5% of, say, Indian labor force (around 522 million) is about 26 million, which is pretty much the total population of Australia.

Exercise your best judgement when you normalize values. Make sure you don’t blow numbers out of proportion by normalizing values in smaller populations. But also don’t hide large counts behind smaller percentages for larger populations.

At this point, you should have enough geocoding and spreadsheet skills to aid you with map making. In the following section, we will talk about geographical data in general and will introduce different geospatial file formats to ensure you are ready to create, use, and share map data.

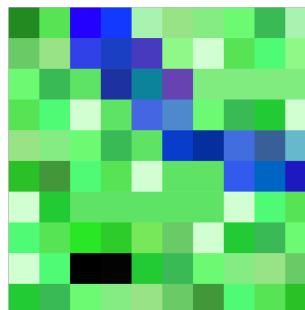
Convert to GeoJSON format

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 tools you can use to work with them. But before we do that, let's talk about the basics of geospatial (map) data.

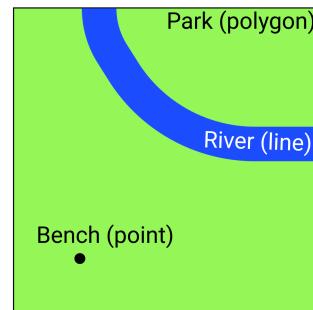
About geospatial data

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 13.5. Raster data, as shown in Figure 13.5a, 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.



(a) Raster is made of cells with values (in this case, color codes)



(b) Vector is made of points, which if connected become lines, which in turn create polygons

Figure 13.5: 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 13.5b. 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 13.6.

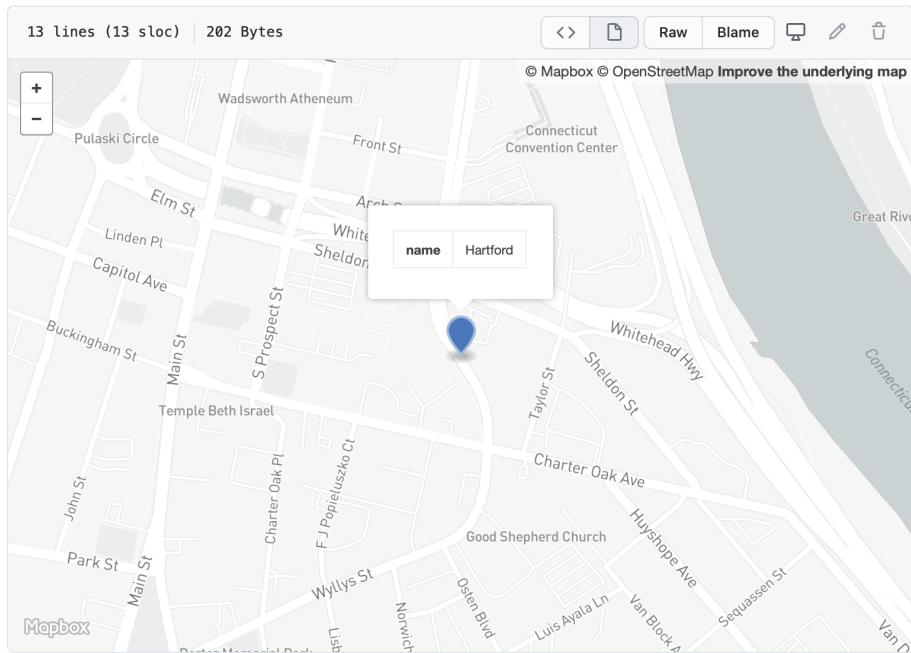


Figure 13.6: GitHub can show previews of GeoJSON files stored in repositories.

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 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.

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.

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 13.7 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.

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 13.8. You can download it for the exercise.

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.

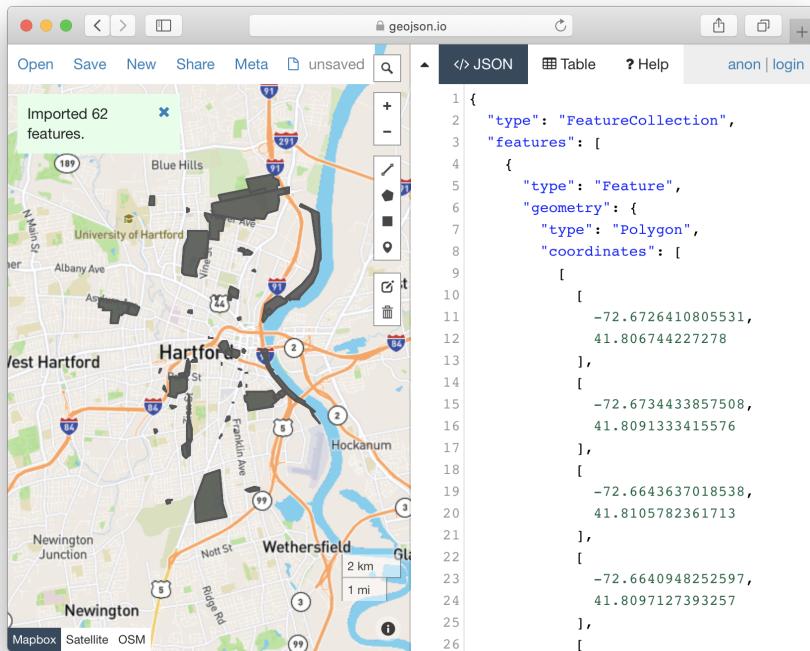


Figure 13.7: GeoJson.io successfully imported Hartford parks KML file.

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 13.8: A spreadsheet with lat/lon columns can be transformed into a GeoJSON with point features.

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 13.9.
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.

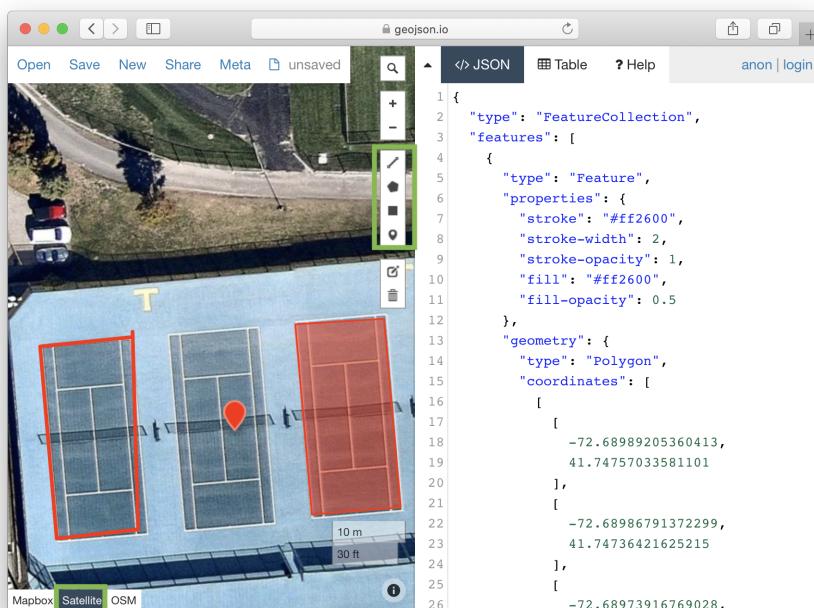


Figure 13.9: Use drawing tools to create points, lines, and polygons in GeoJSON.

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 13.10. 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 13.11 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.
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.

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.

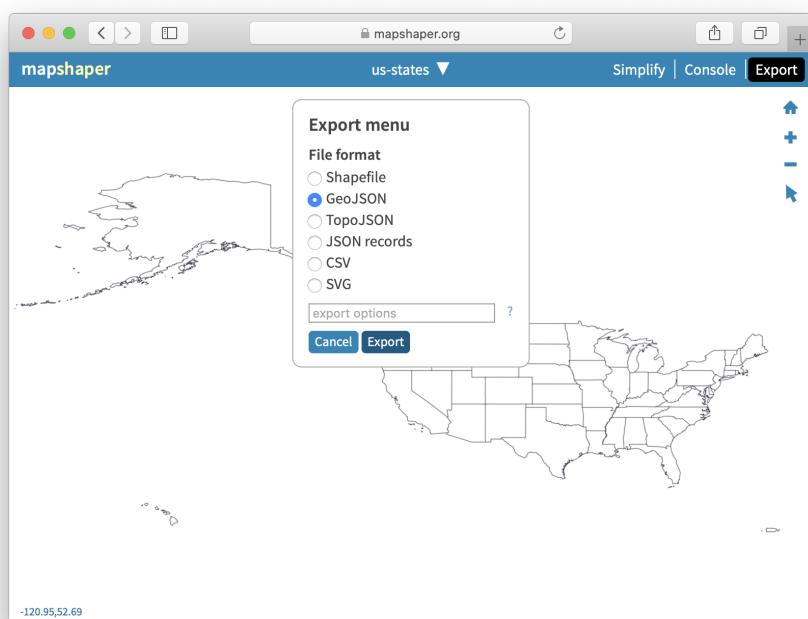


Figure 13.10: You can use Mapshaper to quickly convert between geospatial file formats.

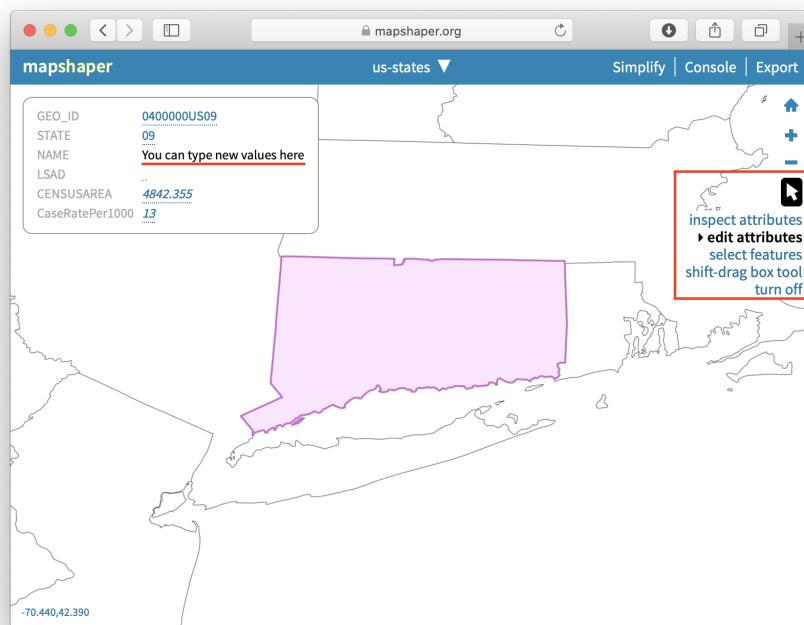


Figure 13.11: Use *edit attributes* tool (under Cursor tool) to edit attributes of polygons, lines, and points.

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 13.12. The map in Figure 13.12a is more detailed and is about 230 kilobytes, but the map in Figure 13.12b is only 37 kilobytes, 6 times smaller!



Figure 13.12: Consider simplifying geometries with Mapshaper to make your web maps faster.

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.
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 13.13), 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.

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,

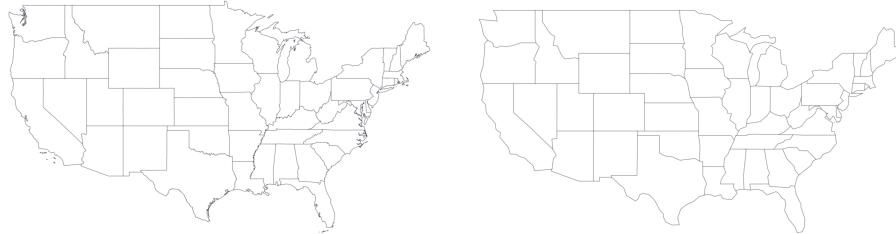


Figure 13.13: Use Simplify & Repair tools in Mapshaper.

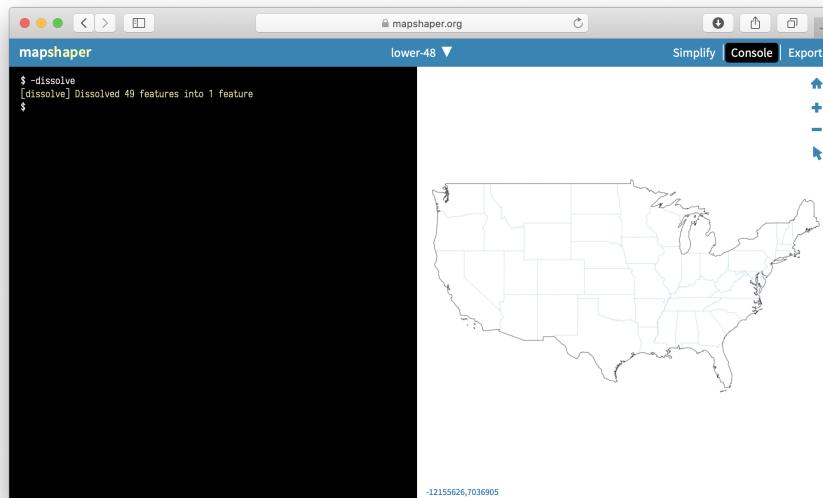


Figure 13.14: Mapshaper lets you dissolve boundaries to create an outline shape.

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 13.14.

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 13.15. You can now save the file on your computer using the *Export* button.

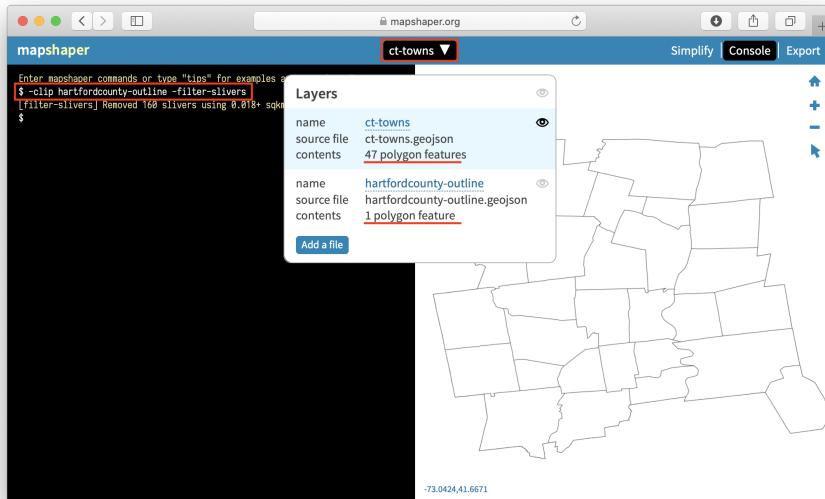


Figure 13.15: When clipping, make sure your active layer is the one being clipped (with many features), not the clipping feature itself.

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 13.16.
7. You can now save the file to your computer by clicking the *Export* button.

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.

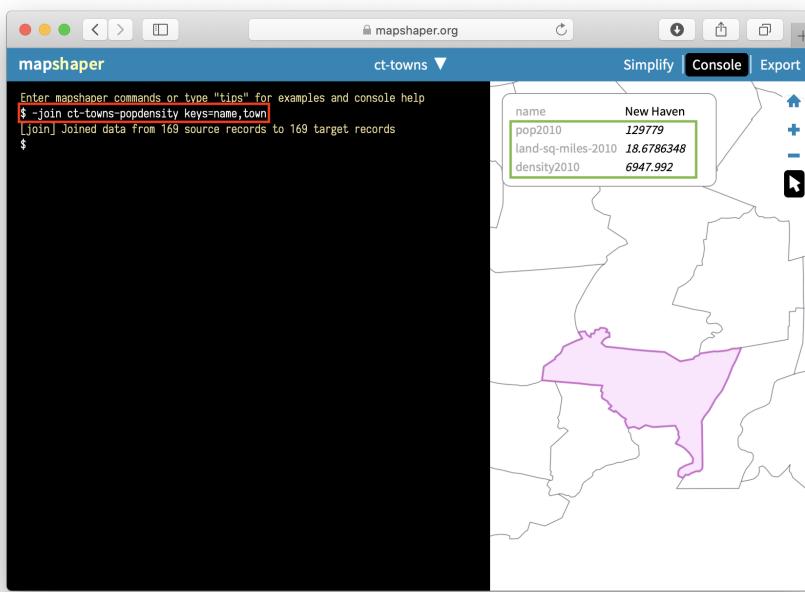


Figure 13.16: Mapshaper lets you join spatial and CSV files using common keys (for example, town names).

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 13.17.
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:

```
[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
```

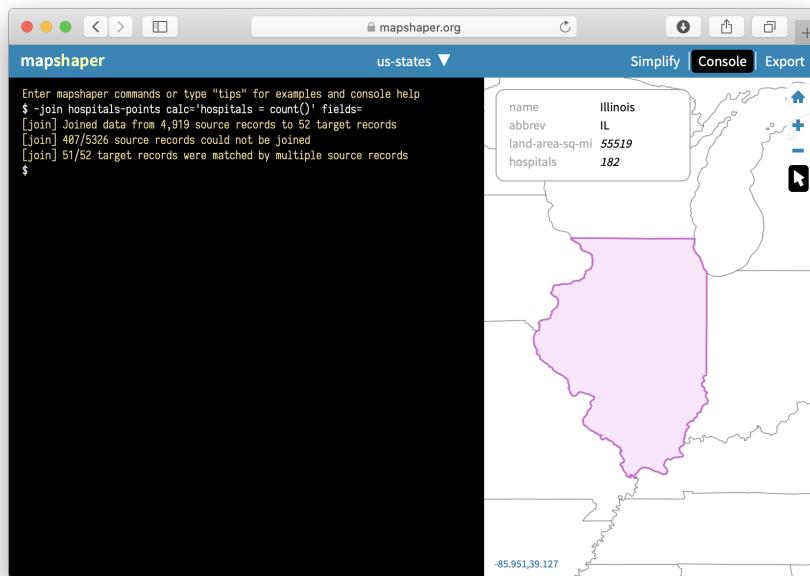


Figure 13.17: Mapshaper's -join can count points in polygons.

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 13.18. You may stop right here and move to the next step, or keep assigning district names to a few other neighboring towns.
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*.

	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 13.18: Create a two-column crosswalk of towns and which districts they should be merged to.

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 13.19 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

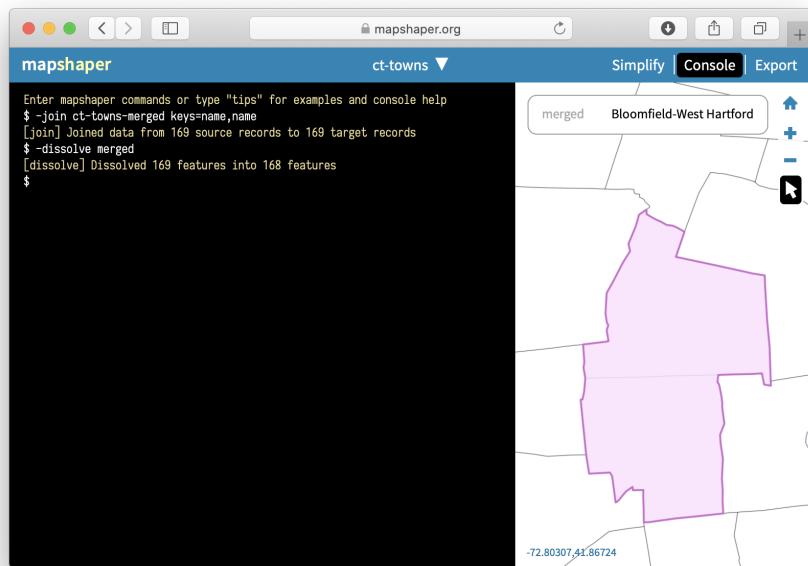


Figure 13.19: Merge polygons based on a predefined crosswalk.

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.

Georectify with MapWarper

TODO: write this section about using MapWarper, a tool created and hosted by Tim Waters, to upload and georectify 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."

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 13.20.
4. In the dropdown menu of *Save file...* window, choose KML format, like is shown in Figure 13.21.

Alternatively, you can use any zip-utility to extract a KML file from KMZ. KMZ is simply a 'zipped' version of a KML file!

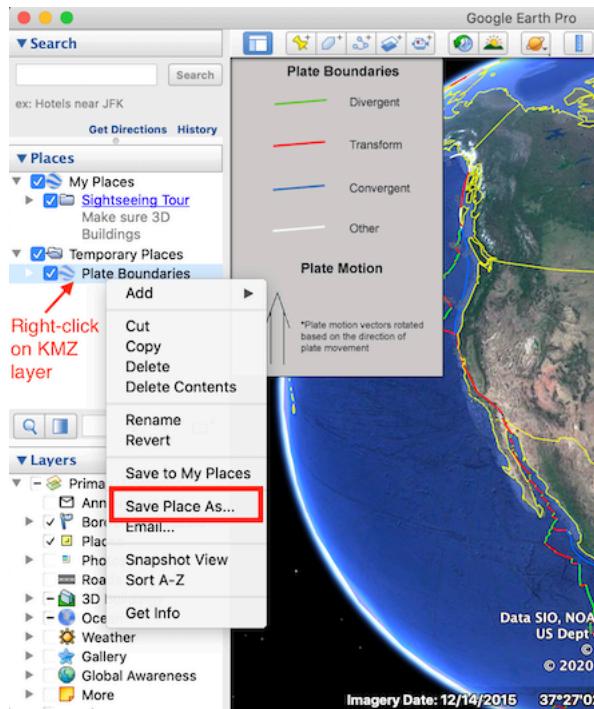


Figure 13.20: In Google Earth Pro, right-click the KMZ layer and choose *Save Place As...*

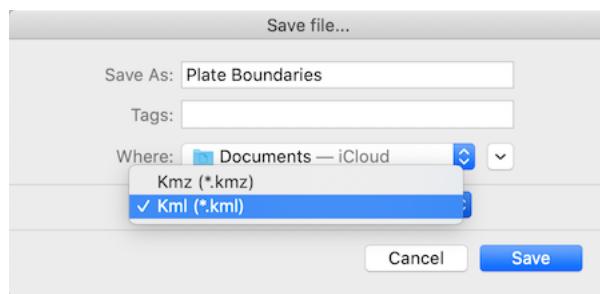


Figure 13.21: Save as KML, not KMZ.

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 14

Detect Bias in Data Stories

TODO: Rewrite chapter

While we like to believe data visualizations simply “tell the truth,” when you dig further into this topic, you realize that there are multiple ways to represent reality. In this chapter, you will learn how visualizations display the biases of the people and the software that create them. Although we cannot stop bias, we can teach people to look for and detect it, and be aware of our own.

Sections in this chapter:

- How to Lie with Charts, inspired by Darrell Huff (1954)
- How to Lie with Maps, inspired by Mark Monmonier (1996)

TODO: Build on themes from the new intro: Both maps make valid interpretations of the data, but generate very different impressions. You may be surprised to learn that mapmakers have few universal guidelines when selecting ranges or colors, other than a general dictum to make “honest and useful” decisions. Different mapmakers could choose to divide the legend into three, or four, or five ranges, and all would be correct. Furthermore, they could choose to divide ranges into equal intervals (such as quartiles or quintiles) or intervals that vary according to patterns in the data (such as standard deviations from the mean). Just like charts, we can mislead and lie with maps, but we can also create very different pictures of the truth.

TODO: Add new sections on other types of bias.

Example: Map with “empty” land ignoring indigenous people <https://twitter.com/indigenia/status/1308406887231246337?s=12>

Example: Watch for negative versus positive framing and how it affects readers. For example, British statistician David Spiegelhalter notes that US hospitals

report *mortality rates*, while UK hospitals report *survival* rates. When considering the risks of a surgical procedure for member of your family, a 5 percent mortality rate sounds worse than a 95 percent survival rate, even they're mathematically identical. Furthermore, augmenting the rate with raw numbers (such as 20 deaths out of 400 patients) raises the impression of risk even higher, because we begin to imagine real people's lives, not abstract percentages.¹

Learn more: - Darrell Huff, *How to Lie with Statistics* (W. W. Norton & Company, 1954), <http://books.google.com/books?isbn=0393070875> - Mark S. Monmonier, *How to Lie with Maps*, 2nd ed. (University of Chicago Press, 1996), <http://books.google.com/books?isbn=0226534219> - Nathan Yau, "How to Spot Visualization Lies," FlowingData, February 9, 2017, <http://flowingdata.com/2017/02/09/how-to-spot-visualization-lies/>

How to Lie with Charts

One of the best ways to learn how to detect bias in data visualization is to intentionally manipulate a chart, and tell two (or more) opposing stories with the same data. You'll learn what to watch out for when viewing other people's charts, and think more carefully about the ethical issues when you design your own.

This exercise was inspired by a classic book published more than fifty years ago: Darrell Huff, *How to Lie with Statistics* (W. W. Norton & Company, 1954), <http://books.google.com/books?isbn=0393070875>

Right-click this link and Save to download this sample data in CSV format to your computer: [us-gross-domestic-product-per-capita](#). This historical data on economic productivity comes from the World Bank, World Development Indicators, <http://data.worldbank.org/data-catalog/world-development-indicators>

Upload the CSV file to your Google Drive (with Settings to Convert to Google format) to create a Google Sheet.

Select the data cells and Insert > Chart > Line chart, similar to the default version shown below:

In your Google Sheet chart, double-click the vertical y-axis to edit the Minimum and Maximum values.

Make the line look "flatter" (slower economic growth) by lowering the minimum to \$36,000, and increasing the maximum to \$100,000, as shown below:

Make the line look like a "sharper increase" (faster economic growth) by increasing the minimum to \$38,000, and lowering maximum to \$52,000, as shown below:

¹David 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



Figure 14.1: Screenshot: Edit the Min and Max values of the Y-axis

** TO DO – add conclusion **

How to Lie with Maps

One of the best ways to learn how to detect bias in data visualization 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.

This exercise was inspired by Mark S. Monmonier, *How to Lie with Maps*, 2nd ed. (University of Chicago Press, 1996), <http://books.google.com/books?isbn=0226534219>

First, scroll through this data on Median Household Income for Hartford-area towns, 2011-15, from American Community Survey 5-year estimates. Or right-click to open this Google Sheet in a new tab.

Next, explore two different polygon maps of the same data. Use the drop-down menu to compare “Extreme Differences” versus “Uniform Equality”

Why are these two maps portray the same data so differently? To see the answer, look at the data ranges. . .

** TO DO **

Create your own version...

TODO: Add section on how interactive maps such as Google Maps change borders and data depending on the internet address of the user

Summary

TODO

Chapter 15

Tell Your Data Story

TODO: Write this chapter: Tell the story about your data, including its most meaningful insights and limitations. Write compelling titles, labels, and sentences to accompany your visualization. Call attention to the most meaningful insights in your chart, and explain any data limitations.

TODO: Return to the “Sketch Your Data Story” exercise from the introduction....These three simple steps will help you start sketching out your data story. To be sure, there’s much more work to be done, such as finding, cleaning, and visualizing data, which we’ll cover in the chapters to come. But transferring ideas from your brain onto paper is the big first step, because making your thoughts visible enables you to act on them more easily. Scribbling words and sketching pictures not only helps *you* to see your thoughts more clearly, but it also allows you to share what’s been inside your head more concretely with friends and colleagues. After reflecting on your sheets of paper and listening to feedback, you can cross out not-so-good ideas, replace them with better ones, and add new sheets of paper with more ideas. Finally, you can spread out the sheets on a table, move them around to reorder the sequence, and begin to define the three essential stages of your data story: the beginning, middle, and end. Start to imagine these sheets of paper as preliminary slides for your presentation desk, or paragraphs and pictures for your written report or website. We’ll come back to this exercise near the end of the book in Chapter 15: Telling Your Data Story

Write down what your eyes see in your visualization..... and tell us what it matters..... Despite what you may have been told, tables and charts and maps alone do not tell a story. You need to interpret what the viewers see....

Who's your audience?

What's your storytelling format?

- Many visualization books and workshops assume that you will hand a

sheet of paper to people sitting around a board room, or a static PDF document sent via email. . . . Those formats are fine, but do not reflect the wider range of ways to communicate with people

- a “data walk” to engage community stakeholders <https://www.urban.org/sites/default/files/publication/72906/2000510-data-walks-an-innovative-way-to-share-data-with-communities.pdf>
- presentation slides with live web connection for interactivity (or video screencast)
- website with textual narrative and interactive visualizations

This chapter draws inspiration from Cole Nussbaumer Knaflic, *Storytelling with Data: A Data Visualization Guide for Business Professionals* (Wiley, 2015), <http://www.storytellingwithdata.com/book/>

- Beginning, Middle, and End
- Draw Attention to Meaning
- Integrate Story with Your Data
- Write Clearly about What You Visualize
- Acknowledge Limitations of the Data
- Credit Data Sources and Collaborators

Credit sources and collaborators on dataviz products and readme files

Under US law, you cannot copyright data, such as the raw information in the rows and columns of a spreadsheet. But you can copy, but representations of data can be protected by copyright. . . . explain. . . . In the spirit of openness, we encourage you to share your data visualizations under a Creative Commons license. . . . explain. . . . in fact, this book is copyrighted, and the source text is publicly available under a Creative Commons TODO: TYPE license. . . .

using color to highlight key items or outliers in your data

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.

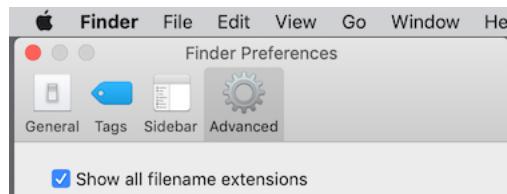


Figure A.1: On a Mac, go to *Finder* then *Preferences* then *Advanced* and check the box to *Show all filename extensions*.

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.

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?

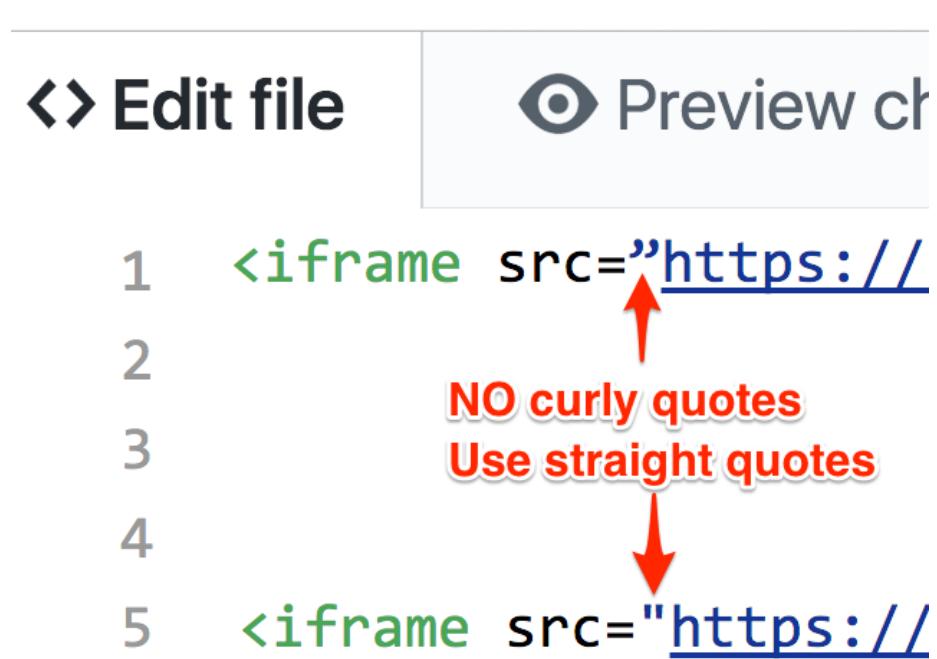


Figure A.3: Screenshot: Curly quotes versus straight quotes

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:
 - 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)

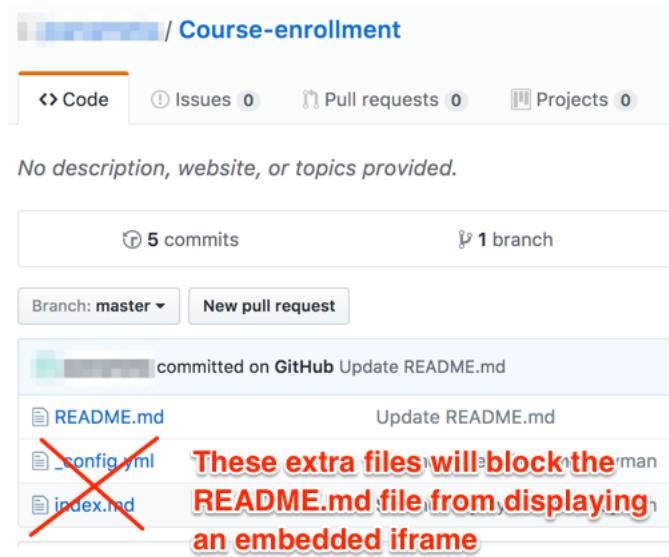


Figure A.4: Screenshot: Extra files in GitHub repo will block iframe in your README

- 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
 - 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.

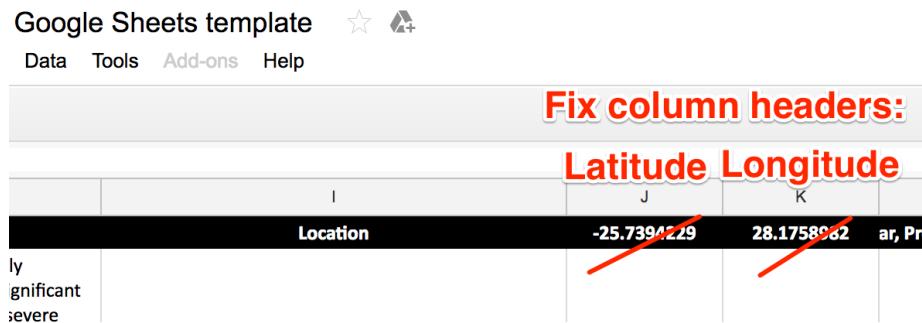


Figure A.5: Screenshot: User accidentally renamed column headers in the Points tab

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.

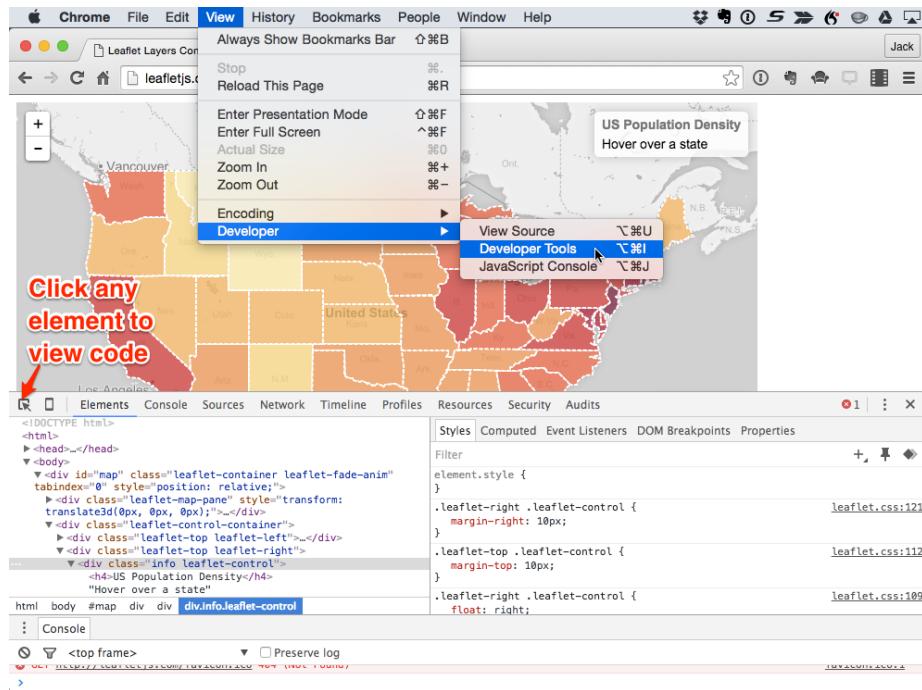
In Chrome for Mac, go to View > Developer > Developer Tools

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 Help icon. Below the menu is a toolbar with icons for print, refresh, and other functions, along with currency and decimal input fields.

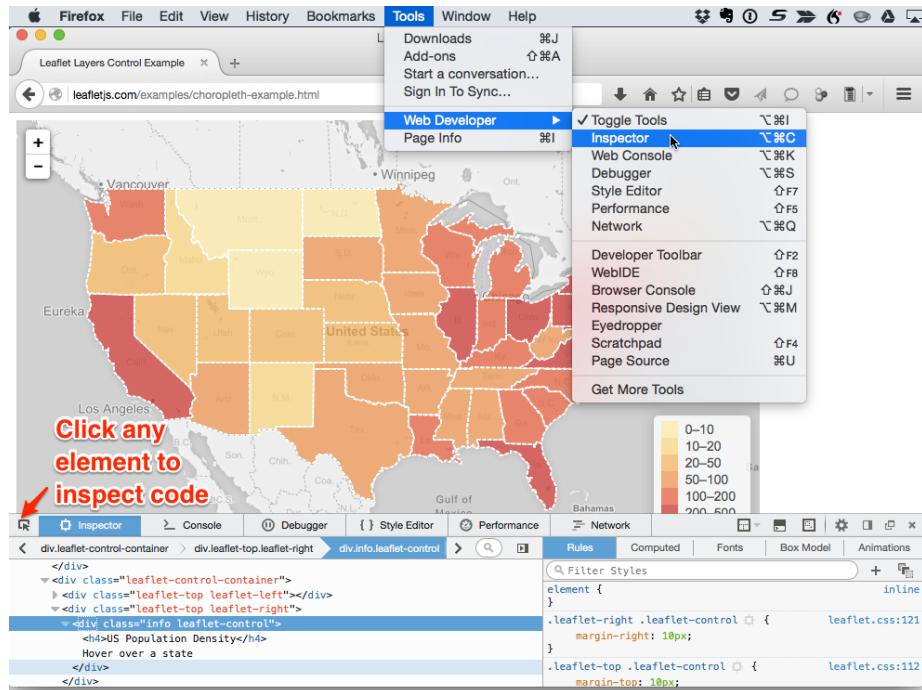
The spreadsheet contains a table with 12 rows and 3 columns. The first column is labeled "1", the second column is labeled "Setting", and the third column is labeled "A". Rows 2 through 9 show configuration settings for a map, while rows 10 through 12 show map settings. The text in the "Setting" column for rows 2 through 9 is bolded. A large red watermark with the text "Do NOT rename or delete these labels" is overlaid across the middle of the table.

	Setting	A
1	Map Info	Do NOT rename or delete these labels
2	Map Title	Demo
3	Map Subtitle	trans
4	Display Title	on
5	Author Name	Jack
6	Author Email or	jack.
7	Author Code Credit	<a href="
8	Author Code Repo	https://
9	Map Settings	
10	Basemap Tiles	Carto
11	Cluster Markers	off

Figure A.6: Screenshot: Do not rename or delete



In Firefox for Mac, go to Tools > Web Developer > Inspector



- 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/>.
- 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/alexa-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/>.
- 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/>.
- 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/>.
- Rost, Lisa Charlotte. "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>.
- Spiegelhalter, David. *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. *Beautiful Evidence*. Graphics Press, 2006. <http://books.google.com/books?isbn=0961392177>.
- 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>.
- 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.
- 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.