Hands-On Data Visualization

Interactive Storytelling from Spreadsheets to Code

Jack Dougherty Ilya Ilyankou

2020-06-23

Contents

Introduction

This open-access **book-in-progress**, by Jack Dougherty and Ilya Ilyankou, is under contract with O'Reilly Media, Inc., and was last updated on: 23 Jun 2020

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 and Tableau Public, then gradually working up to editing open-source Chart.js and Leaflet code templates 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.

Read for free online at https://HandsOnDataViz.org or purchase print/eBook editions, to come from the publisher.

Please send corrections or suggestions for this book-in-progress to handsondataviz@gmail.com, or open an issue or submit a pull request on its 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 Jack Dougherty and Ilya Ilyankou," so that we can maintain the option of publishing this book in other forms.

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

Trademarks

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

Disclaimer

The information is this book is provided without warranty. The lead author, contributors, 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.

Authors

Authors	About Us
	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.
	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.

Acknowledgements

An earlier version of this book was titled *Data Visualization For All* and designed to accompany a free online edX course by the same name at Trinity College. Two co-instructors for this edX course contributed valuable ideas and co-created videos: Stacy Lam, Trinity Class of 2019, and David Tatem, Instructional Technologist at Trinity College. Veronica X. Armendariz, Trinity Class of 2016, also made valuable contributions to an earlier version of the book while she was a Teaching Assistant for the DataViz internship seminar.

Videos for the edX course were produced with Trinity College Information Technology staff and friends: Angie Wolf, Sean Donnelly, Ron Perkins, Samuel Oyebefun, Phil Duffy, and Christopher Brown.

Funding for student contributors to an earlier version of this book was generously provided by the Community Learning Initiative and Information Technology Services at Trinity College in Hartford, Connecticut.

Thanks to many individuals and organizations for generously making time to help us learn many of the skills that we teach in this book: Alvin Chang and Andrew Ba Tran, who tutored and shared their Leaflet map templates while at

The Connecticut Mirror; Patrick Murray-John, formerly at the Roy Rosenzweig Center for History and New Media, who clued us into being *code-curious*, and many others at The Humanities and Technology Camp (THATCamp) conference series...

What is Data Visualization?

Data visualization is broadly defined as a method of encoding quantitative, relational, or spatial information into images. Classic examples include Charles Menard's figurative map of Napoleon's defeat and retreat during the Russian campaign of 1812, and John Snow's dot map of cholera cases during the London epidemic of 1854.

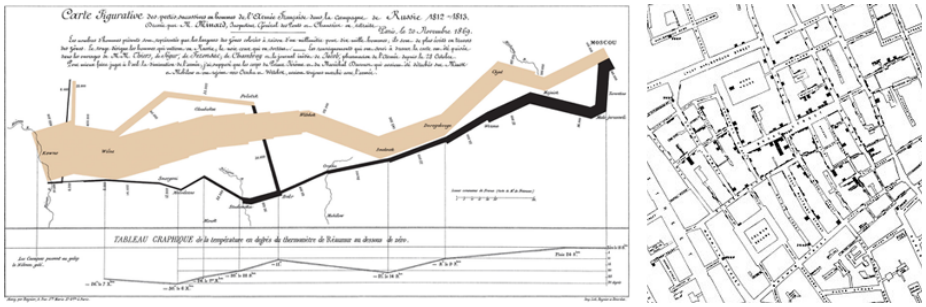


Figure 1: Images: Menard's figurative map (left) and Snow's dot map (right), from Wikimedia

This free online introductory book focuses on selected topics in data visualization:

Charts and maps Despite the growing variety of visualization types, this book features chapters on creating charts and maps, and a wide range of ways to communicate with these classic models.

Reusable tools and templates: Unlike infographics created for one-time use, all of the tools and templates in this book are recyclable, and allow you to upload a new dataset to display your story.

Free and easy-to-learn: We have selected data visualization tools that are free to use (or work on a freemium model, where advanced features or higher usage requires payment), and searched for those that we believe are easy-to-learn, based on our teaching experience with undergraduate students and non-profit community organizations.

Interactive on the open web: Many books assume that you will deliver your data visualizations to in-person audiences on printed paper or presentation

slides. But in this book, we show how to embed interactive charts and maps on your website, to share with the wider public.

Storytelling: Data visualization is more than pretty pictures. In this book, the best visualizations are those that tell your data story – and pull readers' attention to what really matters – by combining images and text, and offering exploration with explanation.

Learn more

- Michael Friendly and Daniel J. Denis, "Milestones in the History of Thematic Cartography, Statistical Graphics, and Data Visualization," 2001, http://www.datavis.ca/milestones/
- Isabel Meirelles, Design for Information: An Introduction to the Histories, Theories, and Best Practices Behind Effective Information Visualizations (Rockport Publishers, 2013), http://isabelmeirelles.com/book-design-for-information/
- Edward Tufte, The Visual Display of Quantitative Information (Graphics Press, 1983), and subsequent works at https://www.edwardtufte.com

Why this book?

Hands-On Data Visualization, an open-access online textbook, seeks to help you tell your story—and show your data—through the power of the public web.

This open-access book reflects what I've learned while teaching data visualization to undergraduate students at Trinity College, and now to a global online class on the Trinity edX platform. Over the past few years, Trinity students and I have built interactive charts and maps in partnership with non-profit organizations in Hartford, Connecticut, to help them share their stories with data on the public web. Also, my students and colleagues have used these tools to create On The Line: How Schooling, Housing, and Civil Rights Shaped Hartford and its Suburbs, an open-access book-in-progress that features interactive historical maps of urban-suburban change. Students and colleagues who wrote tutorials, designed learning exercises, or developed code templates for *Hands-On Data Visualization* are listed as authors and contributors.

Although my outstanding colleagues have professional training, do not confuse them with me, the proverbial "Jack of all trades, master of none." I do not consider myself an expert in data visualization, nor should anyone mistake me for a computer scientist or data scientist. Inspect my higher education transcripts and you'll see only one computer science class (something called FORTRAN77 back in 1982), and not a single course in statistics, sadly. Instead, my desire to learn data visualization was driven by my need as an historian to tell stories about urban-suburban places and change over time. If you've ever watched me

teach a class or deliver a presentation on these topics – always talking with my hands in the air – you'll understand my primal need to create charts and maps. Stories become more persuasive when supported with data, especially well-crafted images that convey data relationships more clearly than words. Furthermore, these data stories become more powerful when we share them online, where they reach broader audiences who can interact with and evaluate our evidence.

In the early 2000s, when I began to dabble in data visualization, our tools were expensive, not easy to learn, and not designed to share our stories on the public web. (One of my well-worn jokes is point to the bald spot on my head, and claim that it was caused while tearing out my hair in frustration while using ArcGIS.) But everything began to change around 2005 when Google Maps publicly released its application programming interface (API) that allowed people with some coding skills to show data points on an interactive web map. Gradually, between 2008-11, I began learning what was possible by working on map projects with talented programmers and geographers, such as Jean-Pierre Haeberly at Trinity, and Michael Howser at the University of Connecticut Libraries Map and Geographic Information Center (MAGIC, my favorite acronym), thanks to a grant from the National Endowment for the Humanities. Free and low-cost workshops sponsored by The Humanities and Technology Camp (THATCamp) at the Center for History and New Media at George Mason University, and Transparency Camp by the Sunlight Foundation, introduced me to many people (especially Mano Marks and Derek Eder) who demonstrated easier-to-use tools and templates, such as Google Fusion Tables and GitHub. Closer to home, Alvin Chang and other data journalists at the Connecticut Mirror showed me how to tell stories on the web with more flexible open-source tools, such as Leaflet and Highcharts.

All of these data visualization lessons I learned have been so valuable—to me, my students, our community partners, and thousands of readers on the web—that my co-authors and I have agreed to share our knowledge with everyone for free. This open-access book is guided by the principle of democratization of knowledge for the public good, hence the book's title: *Hands-On Data Visualization*. Not everyone can afford to make this choice, I realize. But the mission of Trinity College is to engage, connect, and transform, with both our local city of Hartford and the world at large. Since Trinity already pays my salary as a tenured professor, the right thing to do with the knowledge my students and I have gained is to pay it forward. That's why we created *Hands-On Data Visualization.*

If this free book is valuable for your education, then join us by sharing and supporting it for future readers:

- Tell your friends about the book and share the link via social media, text, or email

- Improve the book by adding comments or suggesting new chapters on our GitBook platform

Try out the tutorials, explore the online examples, share what you've learned with others, and dream about better ways to tell your data stories.

How to Read

This open-access book-in-progress is free to read online at http://HandsOnDataViz.org to fully experience the interactive charts, maps, and video clips. Any modern web browser will display the book, but readers may prefer larger screens (desktop, laptop, tablet) over smaller screens (such as smartphones). In your web browser, try these toolbar features near the top of the page:

- Menu
- Search
- Font to adjust text size and display
- View source code on GitHub
- Shortcuts (arrow keys to navigate; `s` to toggle sidebar; `f` to toggle search)
- Social Media
- Share

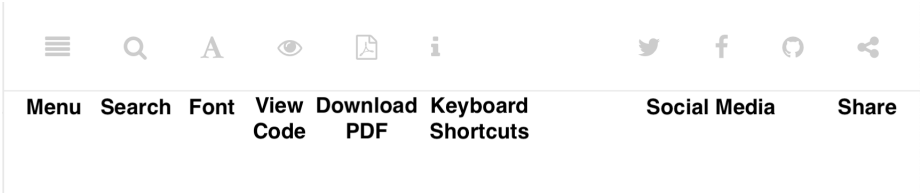


Figure 2: Screenshot: How to read

Open links in new tabs

Keep your place when reading online and moving between pages.

- Two-finger trackpad click
- or Control + click (Mac)
- or Alt + click (Chromebook)
- or right-click (Windows and others)

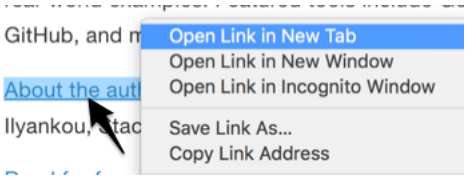


Figure 3: Screenshot: Open link in new tab (on Mac)

Use a second monitor

If you have a small screen, consider connecting a second monitor, or work next to a second computer or tablet. This allows you to view tutorials in one screen and build visualizations in the other screen.

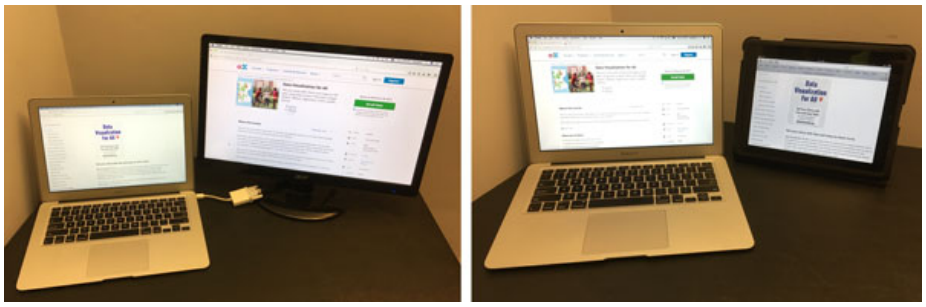


Figure 4: Image: Laptop with second monitor, and with tablet

Refresh browser

To view the most up-to-date content in your web browser, do a "hard refresh" to bypass any saved content in your browser cache.

- Ctrl + F5 (most Windows-Linux browsers)
- Command + Shift + R (Chrome or Firefox for Mac)
- Shift + Reload button toolbar (Safari for Mac)

Chapter 1

Choose Tools to Tell Your Data Story

Do you feel overwhelmed by the enormous range of data visualization tools? There's been so many different tools released in recent years that anyone would have a hard time deciding which ones to use. Even if you limit your choices to the dozen or so tools specifically mentioned in this book, how do you make wise decisions?

- Draw and Write Your Data Story reminds us to start with the most important item in your toolkit: ***your story***. Begin by drawing pictures and writing questions or sentences to capture your ideas on paper, and then choose the most appropriate tools to create your vision.
- Ask Questions When Choosing Tools lists several criteria to consider when making software decisions. Many of us look for free or affordable tools in the perfect sweet spot—easy-to-learn, yet powerful—and that's the focus of this book.
- Rate Three Simple Map Tools invites readers to create a basic interactive point map using three different online tools, and to evaluate each one using selected criteria from the chapter above.

Enroll in our free online course **TO DO add link**, which introduces these topics in the brief video below, and offers more exercises and opportunities to interact with instructors and other learners.

Watch the YouTube Video

Draw and Write Your Data Story

Before you dive deeply into software, think about the most important item in your toolkit: **your story**. The primary reason we're designing visualizations is to improve how we communicate our data story to other people, so let's begin there.

Push away the computer and pick up some old-school tools:

- colored markers or pencils
- lots of blank paper
- your imagination

First, at the top of the page, write down your data story.

- Is it in the form of a question? If so, figure out how to pose the question.
- Or maybe it's in the form of an answer to that question? If so, spell out your clearest statement.
- If you're lucky, perhaps you already can envision a full story, with a beginning, middle, and end.
- Whatever form it takes in your head, write out the words that come to mind.

Further down the page (or on a separate sheet), draw quick pictures of the visualizations that comes to your mind, even if you don't yet have any data. No artistic skills are required. Just use your imagination. - Do you envision some type of chart? Sketch a picture. - Or do you imagine some type of map? Show what it might look like. - Will your visualization be interactive? Insert arrows, buttons, whatever.

Finally, share your data story with someone else and talk through your preliminary ideas. Does your sketch and sentences help to convey the broader idea that you're trying to communicate? If so, this is one good sign that your data story is worth pursuing, with the visualization tools, templates, and techniques in other chapters of this book.

Ask Questions When Choosing Tools

When each of us decides which digital tools best fit our needs, we often face trade-offs. On one hand, many of us prefer easy-to-learn tools, especially those with a drag-and-drop interface, but they often force us to settle for limited

options. On the other hand, we also favor powerful tools that allow us to control and customize our work, yet most of these require higher-level coding skills. The goal of this book is to find the best of both worlds: that "sweet spot" where tools are both friendly and flexible.

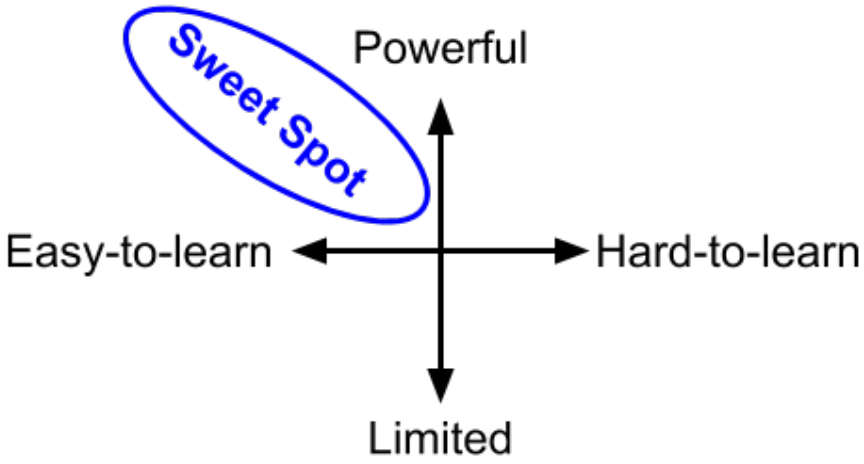


Figure 1.1: Diagram: the 'sweet spot' for easy-to-learn and powerful tools

Before testing out new tools, try listing the criteria that guide your decision-making process. What are the most important factors that influence whether or not you add another item to your digital toolkit? Here's the list that came to our minds:

1. Price: Is the tool free, or is there a "freemium" model to pay for more features or higher usage?
2. Easy-to-learn: Is the tool relatively simple for new users without coding skills?
3. Power: Does the tool support large amounts of data, and various types of visualizations?
4. Customization: Can I modify details about how my work appears?
5. Data Migration: Can I easily move my data in and out, in case I switch to a different tool? Hint for historians: Future-proof your digital history projects! Choose tools that allow you to easily export and migrate data to other platforms. Design projects to keep your data separate from its digital presentation.
6. Hosting: Can I decide exactly where my data and visualizations will be stored online?
7. Support: Is the tool actively maintained by its creators, and do they answer questions?

8. Open Source: Is the tool's software visible, can it be modified, and redistributed?
9. Security: Is the tool and my data protected from malicious hackers and malware?
10. Collaborative: Does the tool allow several people to work together on one shared product?
11. Privacy: Under the terms of service, is my data and work private or public?
12. Error-friendly: When something fails, does the tool point out possible problems and solutions?
13. Cross-platform: Does this tool work across different computer operating systems?
14. Mobile-friendly: Will it correctly display my work on various mobile devices and browsers?

That's a long list! It's even longer than the number of tools we'll mention in this book. But don't let it overwhelm you. The diagram at the top of the page illustrates the two most important criteria for the many free tools that are currently available: easy-to-learn and powerful features.

Learn more about choosing tools

Carl V. Lewis, Dataviz.tools: A curated guide to the best tools, resources and technologies for data visualization, http://dataviz.tools

Lincoln Mullen, "How to Make Prudent Choices About Your Tools," ProfHacker blog, Chronicle of Higher Education, August 14, 2013, http://www.chronicle.com/blogs/profhacker/how-to-make-prudent-choices-about-your-tools

Lisa 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/

Lisa Spiro and colleagues, DiRT: Digital Research Tools Directory (formerly Bamboo), http://dirtdirectory.org

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

Rate Three Simple Map Tools

Let's explore criteria from the previous chapter by comparing three different tools, and reflecting on which factors you feel are most important when making decisions about your toolkit. We'll test three drag-and-drop tools to transform sample address data into a simple interactive point map.

Each tool can **geocode** address data by looking up a location (such as 500 Main Street, Hartford CT) in a large database, deciding on the best match, and converting this data into latitude and longitude coordinates (such as 41.762, -72.674).

For our sample data, we'll use this table of 9 locations in North America, with 3 intentional mistakes to test for geocoding errors.

Group	Description	Address
Government	US Capitol	East Capitol St NE & First St SE, Washington, DC 20004
Government	Parliament Hill	Wellington St, Ottawa, ON K1A 0A4, Canada
Government	Palacio Nacional	Plaza de la Constitución, Centro, 06000 Ciudad de México, CDMX, Mexico
Higher Education	Trinity College	300 Summit Street, Hartford, CT 06106
Higher Education	University of British Columbia	2329 West Mall, Vancouver, BC V6T 1Z4, Canada
Higher Education	Instituto Tecnológico de Monterrey	Av. Eugenio Garza Sada 2501 Sur, 64849 Monterrey, N.L., Mexico
Error Check	Incorrect spelling of Albuquerque	1801 Mountain Rd, Albakerky NM
Error Check	Imaginary street address	1 Stacylam Street, Chicago IL
Error Check	Street address without city	100 Main Street

Figure 1.2: Image: Sample address data screenshot

First, click this link and Save to download the sample file to your computer: sample-address-data in CSV format. CSV means comma-separated-values, a generic spreadsheet format that many tools can easily open. If you need help with downloading, see this short video tutorial.

Next, build a point map with the sample data, by following the tutorials for the three tools below.

Tool	Step-by-step tutorial in this book
Google My Maps	My Maps tutorial
Carto Builder	Carto tutorial

Finally, rate your experience using each tool with these selected criteria:

- Easy-to-learn: Which tool was the simplest for creating a basic point map?
- Price: Which of these free tools provided the most services at no cost?
- Customization: Which tool enabled you to modify the most details about your map?
- Data Migration: Which tool most easily allowed you to import and export your data?
- Error-friendly: Which tool geocoded most accurately or signaled possible errors?

Recommended: Enroll in our free online course **LINK TO DO** to compare your ratings to other students.

Chapter 2

Improve Your Spreadsheet Skills

Spreadsheets are wonderful tools to organize data into tables of rows and columns. With a spreadsheet, you can sort, filter, calculate, aggregate, and reorganize information to help you find the stories buried inside.

Four common spreadsheet tools:

Tool	Features
Google Sheets	Free, online collaborative spreadsheet on Google Drive. Requires free account.
Microsoft Excel Online	Free online spreadsheet on Microsoft OneDrive. Requires free account. Cannot open CSV generic spreadsheet files.
Microsoft Excel desktop	Not free (US $100+) spreadsheet for Mac or Windows desktop, as part of Microsoft Office.
LibreOffice	Free, open-source alternative to Microsoft Office desktop. Donation requested during download.

Which spreadsheet tool should you use? That depends on how you wish to share and store data for your project.

- If you are the **only person** working on a data project, use any spreadsheet tool.

- If you need to **protect private data**, avoid online tools and use any desktop spreadsheet.
- If you need to **share live data** with others, use Google Sheets.

This introductory online book features Google Sheets because it's a free and easy-to-learn tool for collaborating and sharing data with others. The basic spreadsheet methods shown here are very similar across all spreadsheet tools. But advanced users may need more complex tools to manage very large datasets, or relational databases, or to perform deeper analysis.

If you're new to spreadsheets or want to refresh your skills, see the following chapters:

- Upload and Convert to Google Sheets
- Make a Copy with Google Sheets
- Share with Google Sheets
- Save as CSV or ODS Format
- Sort and Filter Data
- Calculate with Formulas
- Group Data with Pivot Tables
- Match Data with VLookup
- Collect and Share Survey Data with Google Forms

Enroll in our free online course] which offers more spreadsheet exercises and opportunities to interact with instructors and other learners.

Upload Files and Convert to Google Sheets

Google Drive can convert many file types into Google Sheets format:

- Microsoft Excel (.xls and .xlsx)
- OpenDocument Spreadsheet (.ods)
- Comma-separated values (.csv)
- Tab-separated values (.tab)
- Text files (.txt) into Google Sheets format

Tutorial

1) Sign in to your free Google Drive account (http://drive.google.com)

2) To convert files into Google Sheets format, open the Settings (upper-right gear symbol), and **check the box** to Convert uploaded files to Google Docs.

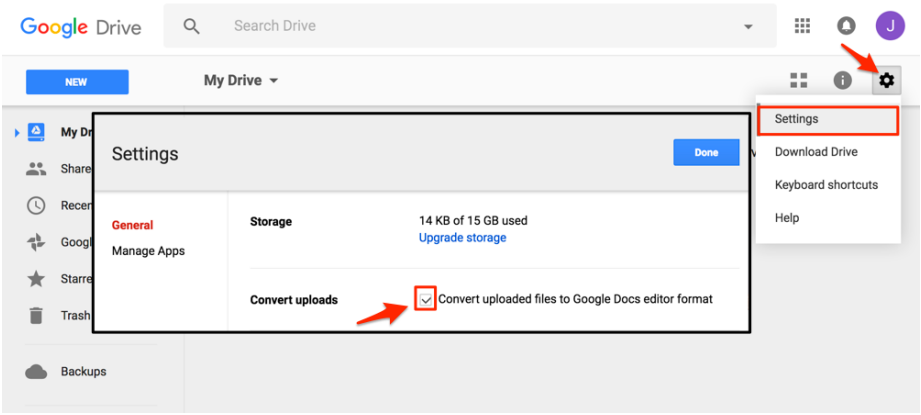


Figure 2.1: Screenshot: Check the box to Convert uploaded files to Google Docs format

3) To upload your file, use the New > File Upload menu OR drag-and-drop it into your Google Drive screen.

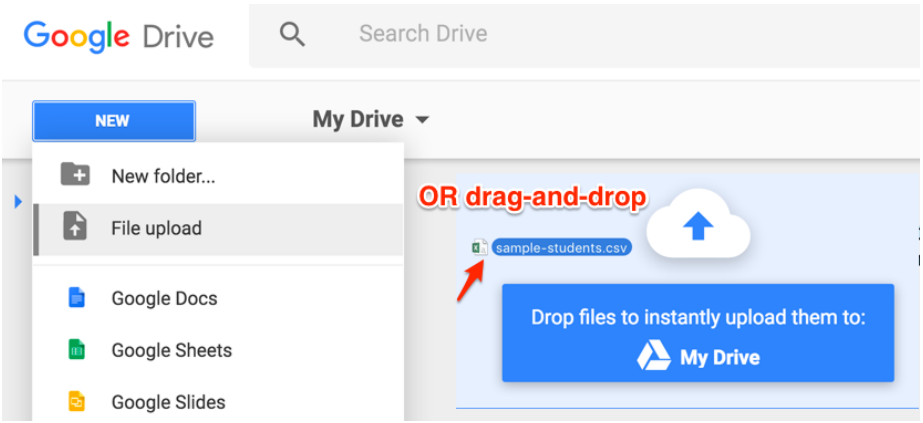


Figure 2.2: Screenshot: New > File upload OR Drag-and-drop the file

4) When your file is successfully converted, the Google Sheets icon will appear. Recommended: Right-click to rename the file and remove the old extension (.xlsx or .csv or other), since it is no longer in this old format.

5) Google Drive files that display different icons have **not** been converted into Google Sheets format.

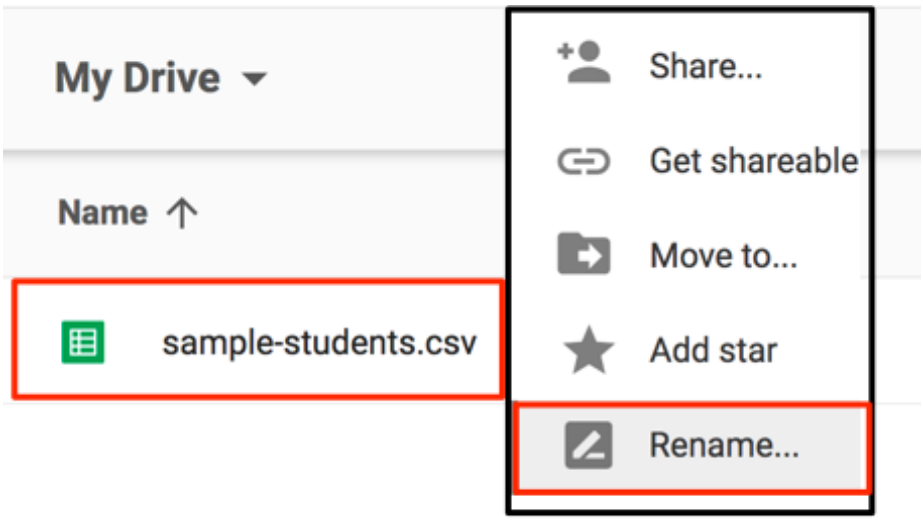


Figure 2.3: Screenshot: Right-click the Google Sheets icon to remove old file extension

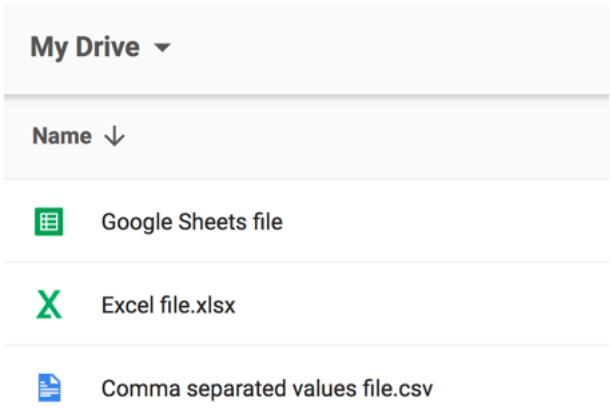


Figure 2.4: Screenshot: Spreadsheet format icons in Google Drive

Beware: A different way to convert spreadsheets into Google Sheets format is the File > Import menu, but this creates two files in your Google Drive (such as data and data.csv), which is confusing.

Make a Copy with Google Sheets

In this book, you will open links to Google Sheets that allow you to view – but not edit – the contents. How can you quickly make your own version that you can edit?

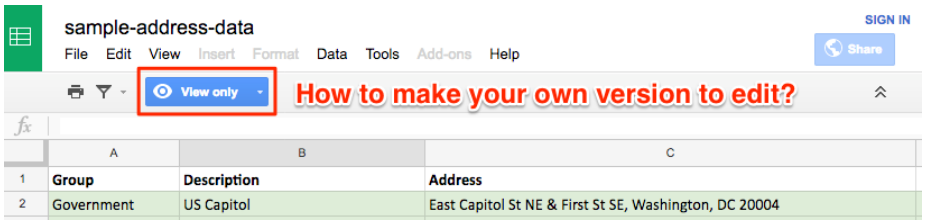


Figure 2.5: Screenshot: View-only in Google Sheets

Best solution

1) Sign in to your Google account in the upper-right corner. Requires a free account.

2) Go to File > Make a Copy to save a duplicate of the spreadsheet to your Google Drive. By default, your copy will be private to you. Go to the Share Data with Google Sheets chapter in this book to allow others to view, comment, or edit your spreadsheet.

Highly recommended: Create folders in your Google Drive to keep your files organized and easily findable.

Alternate solution

Another option is to File > Download As into a different format, such as:

- Microsoft Excel (.xlsx)
- OpenData System (.ods), a generic multi-tab spreadsheet
- Comma-separated values (.csv), a generic single sheet

No Google account is required.

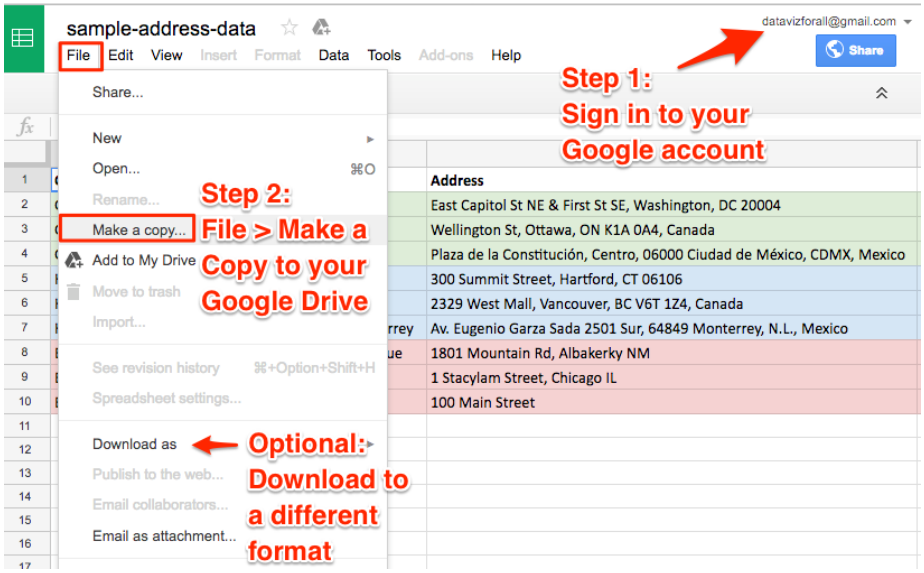


Figure 2.6: Screenshot: Sign in and File > Make a Copy in Google Sheets

Share Data with Google Sheets

To share live spreadsheet data with other people, use Google Sheets (https://www.google.com/sheets/about/). Requires a free Google Drive account.

Video with step-by-step tutorial

1) Sign in to your Google Drive (http://drive.google.com), and in the New menu, select Google Sheets.

2) New spreadsheets are private by default. Only the owner can view and edit.

3) To open your spreadsheet to others, click the blue Share button.

4) To share data with specific individuals, enter their Google usernames.

5) Or, to share data more widely, click the **Advanced** button on the next screen. (I wish Google made this button larger!)

6) Click the Change button and decide on Link Sharing settings:

- Public on the web (anyone can find your data)
- Anyone with the link (similar to an unlisted phone number)
- Off (only specific people you list by Google usernames)

Below those settings, select the Access level:

- Can view
- Can comment
- Can edit (for co-authored data)

7) Select Save, and scroll down on the next screen to select Done.

Tip: To avoid sending a long Google Sheets link to others, use a free link-shortening service such as Bit.ly (http://bit.ly). Requires a free account.

Learn more

- "Share Files from Google Drive," Google help page, https://support.google.com/docs/answer/2494822
- Jack Dougherty, "How to Co-Author and Peer Edit with Google Docs," Web Writing: How and Why for Liberal Arts Teaching and Learning, (2015), http://epress.trincoll.edu/webwriting/chapter/how-to-google-docs

Save Spreadsheets in CSV or ODS

To transfer spreadsheet data to another platform, or import it into a visualization tool, you may need to convert your file into a different format. Consider two options:

Comma-separated values (.csv)

- to transfer only one sheet of data, with no formulas or formatting, into a wide range of spreadsheet and visualization tools

OpenDocument Spreadsheet (.ods)

- to transfer multiple sheets, with basic formulas and formatting, into many spreadsheet tools (Excel, Google Sheets, LibreOffice)

Convert to CSV or ODS with Google Sheets

In the File > Download As menu, select either ODS (to convert a Google Sheets file with multiple tabs, formulas, and formatting) or CSV (to capture only the data in the current sheet).

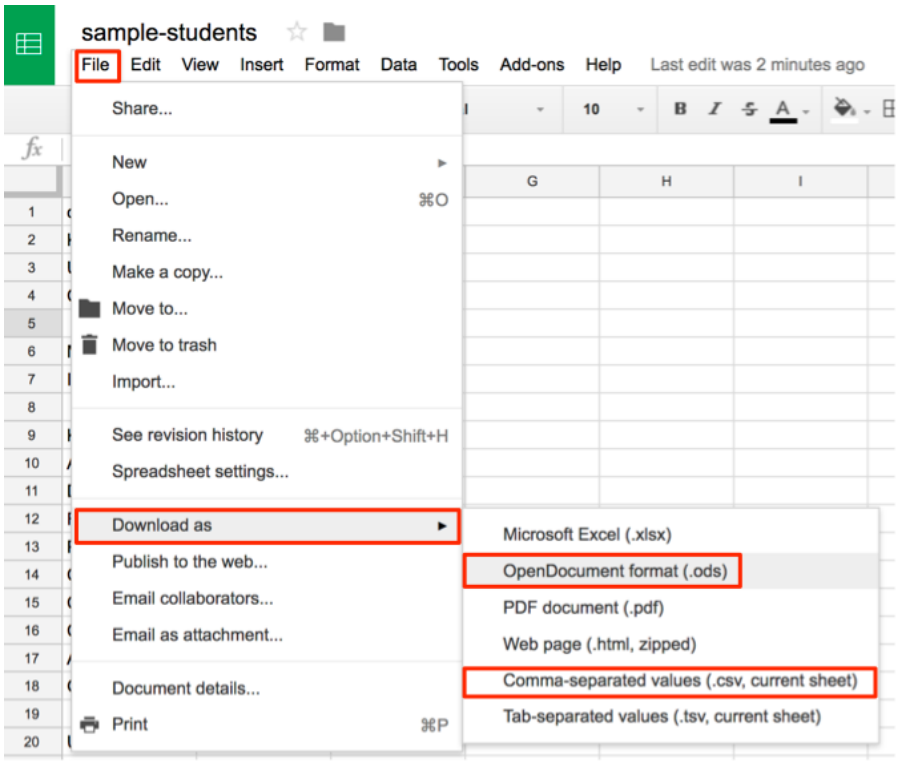


Figure 2.7: Screenshot: Download Google Sheets data in ODS or CSV format

Convert to ODS with Microsoft Excel

In the File > Save As menu, select ODS format.

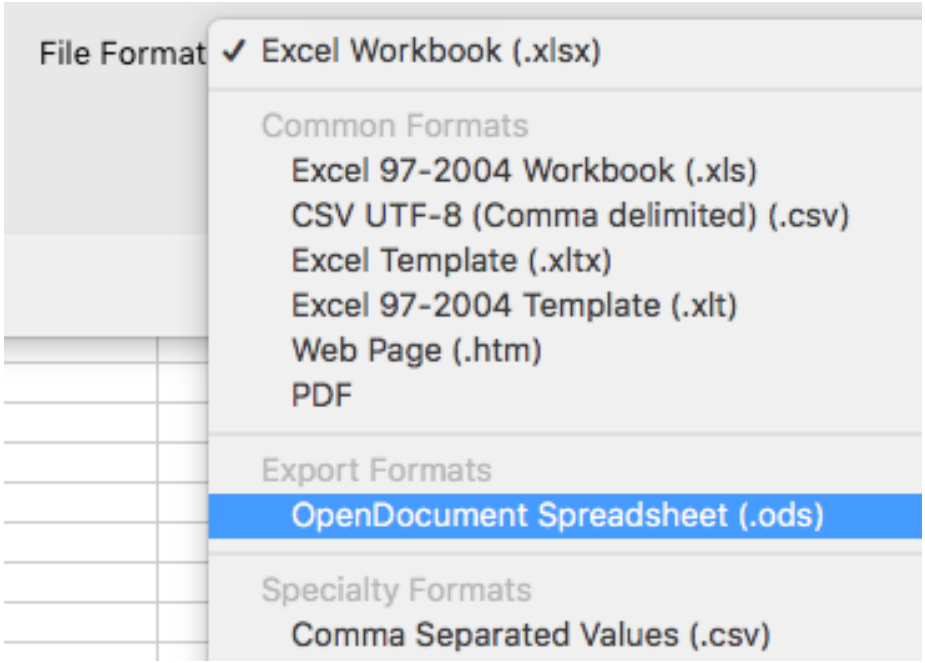


Figure 2.8: Screenshot: Save as ODS with Excel for Mac

Convert to CSV with Microsoft Excel

1) Note that CSV format will save only the first sheet of a multi-sheet Excel workbook. If you have source information or other data in other tabs, keep your original Excel file for backup purposes. You can give them parallel file names:

- data.csv
- data.xlsx

2) In the Excel file, select the File > Save As menu, and select CSV format.

3) Older versions of Excel may warn you that some features (such as formulas and formatting) will not be saved in a generic CSV data file. Be sure to keep a backup Excel version, then click Continue to save your data into CSV format.

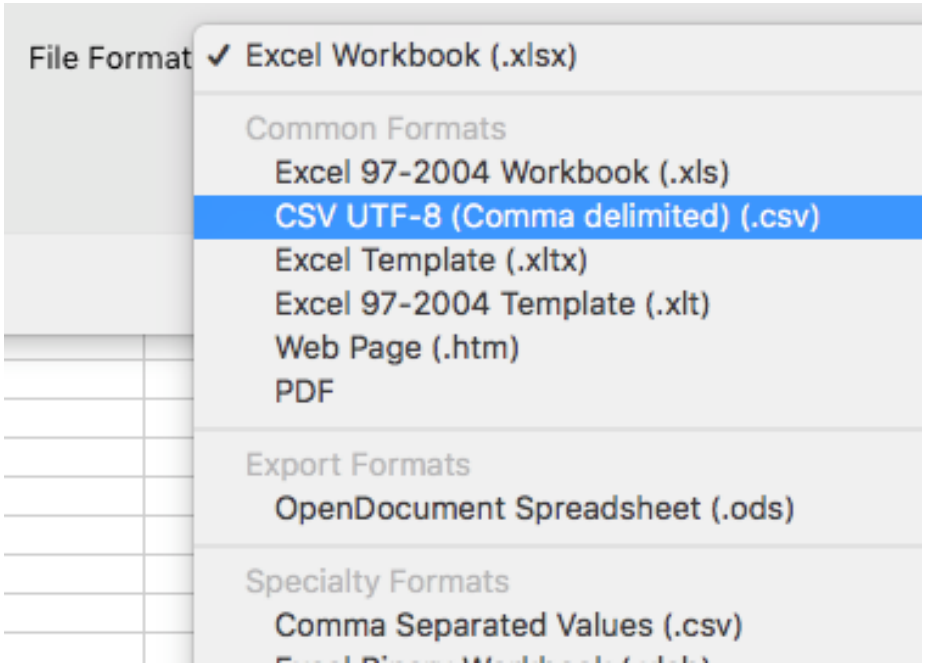


Figure 2.9: Screenshot: Save as CSV in Excel for Mac

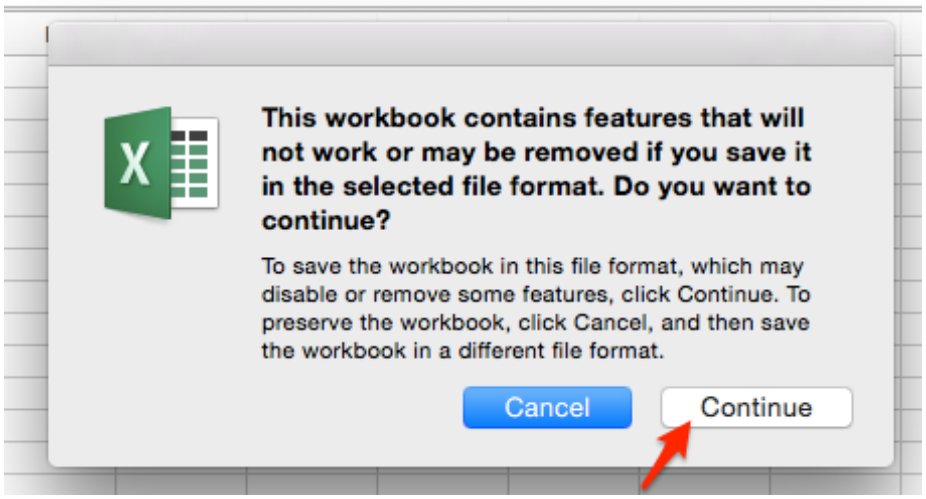


Figure 2.10: Screenshot: Older Excel warning before saving in CSV format

4) In older versions of Excel, when you quit the application, another screen will ask if you wish to save the CSV file a second time. **Don't let Excel confuse you.** If you have not made any changes to the Excel file since the step above, click Don't Save, because you already saved the file in CSV format.

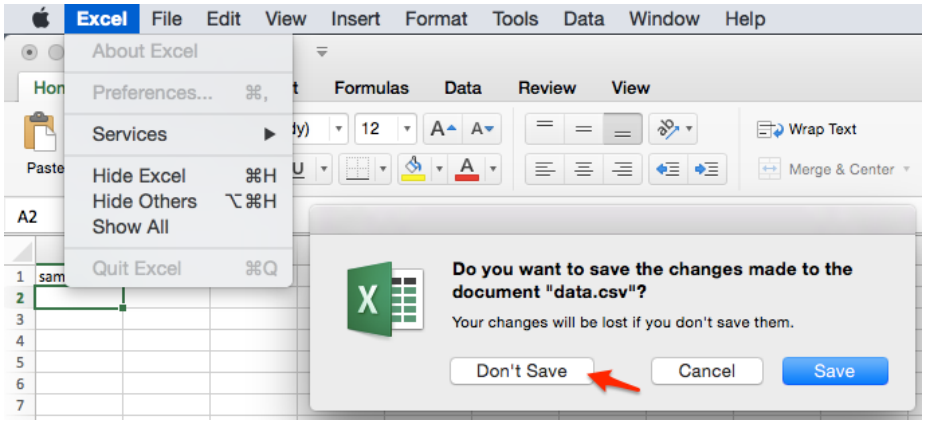


Figure 2.11: Screenshot: Older Excel version: click Don't Save

Sort and Filter Data

TODO:

- write intro on the title concepts
- when possible, start text by posing a common problem, and how this method can solve it
- redo visuals: Google Sheets with better example
- add Filter data

Sort data by columns

To sort data rows by a column, select the entire spreadsheet (top-left corner icon), then right-click or look for the sort menu. Be sure to select the entire sheet to avoid accidentally sorting one column without the adjacent ones.

Filter data by columns

TO DO

Calculate with Formulas and Functions

TODO:

- when possible, start text by posing a common problem, and how this method can solve it
- redo visuals: Google Sheets with better examples
- see other notes inserted below

Simple formulas can save you lots of time. The big advantage of spreadsheet tools is the ability to insert simple formulas to calculate numbers, or combine columns of text, for entire rows and columns.

Write a simple formula

In most spreadsheets, begin writing a simple formula with an equal sign, and refer to specific cells and functions, such as:

- = A2 + B2 + C2

Write formulas with built-in functions

TODO: rewrite to show how this is same as above

- = Sum(A2:C2)

TODO: rewrite to show common numerical and textual functions

- = Average(A2:C2)

Copy and paste, or drag formulas

If you've inserted a formula into one row, how can you quickly do the same calculation across all rows?

Spreadsheets can magically automate calculations across rows or columns. In most cases, you can copy and paste a formula into new cells. Sometimes you can click-and-drag the lower-right corner of a formula cell (which may appear as a cross-hair) to automate calculations.

Copy and Paste > Special > Values to replace formulas with data

After inserting calculations in a spreadsheet, sometimes dynamic formulas must be replaced with static data before the results can be visualized. One solution is to select and copy a column (or the entire sheet), then paste > special > values to replace the formula with numerical results.

Remember that if you need to check or run the calculations again at a later point, click (or right-click) the tab to save a copy to the spreadsheet as a backup.

Create a column of consecutive numbers

To quickly create a column of consecutive numbers, such as unique ID numbers, in most spreadsheet tools:

- Insert the number 1 into a cell and press Return
- Click the cell and float the cursor over the bottom-right corner, where it will change into a cross-hair symbol
- On a Mac, hold down the Option key and drag the cross-hair down to create consecutive numbers
- **TO DO** insert equivalent commands for Windows, Chromebook

Group Data with Pivot Tables

Here's a common problem: You open a large spreadsheet with many rows of data, such as a list of students. Your goal is to count students by categories, such as the number of students by each year of birth. What's the most efficient way to do this?

A solution: Create a pivot table to aggregate (or group together) and summarize data in another spreadsheet tab.

While pivot tables may look different across spreadsheet tools, the concept is the same.

Video with step-by-step tutorial for Google Sheets

1) Click this link and Save to download to your computer: sample-students in CSV format. CSV means comma-separated values, a generic spreadsheet format that most tools can easily open.

	A	B	C	D
1	studentID	year	gender	country
2	1	1980	m	KR
3	2	1988	f	US
4	3	1981	m	CO
5	4	1984	f	
6	5	1976	m	MX
7	6		m	IN
8	7			
9	8	1979	f	KE
10	9	1983	f	AR

Figure 2.12: Screenshot: Long spreadsheet of student data

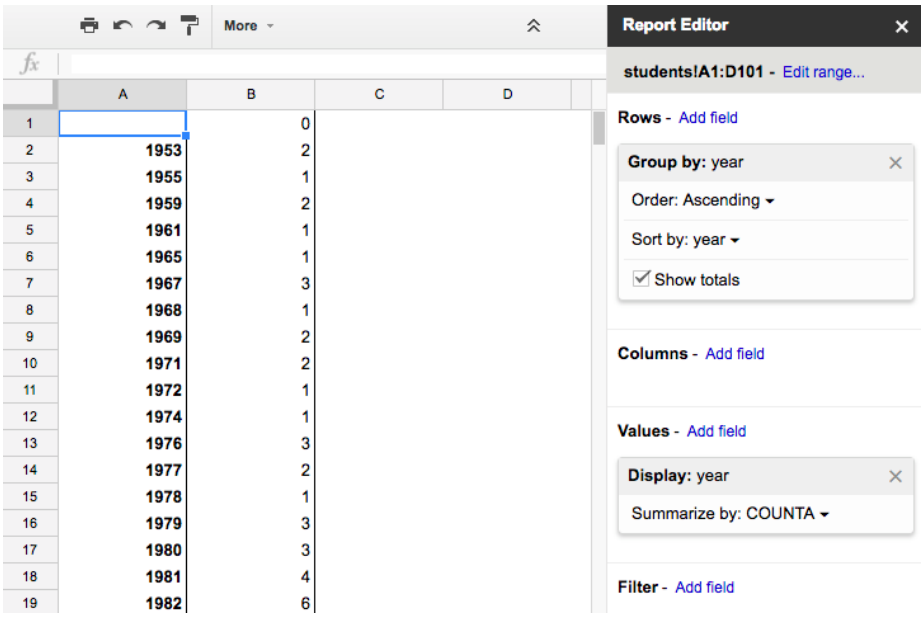


Figure 2.13: Screenshot: Pivot table of count by year of birth

2) Sign into Google Drive (requires free account) and drag-and-drop the sample CSV file to instantly upload. Before you do this, make sure your Settings (gear symbol) is set to Convert Uploads to Google Docs editor format (the default setting).

3) Shift-click to select all columns that you wish to pivot.

4) Select Data > Pivot Table..., which opens a new spreadsheet tab.

5) In Report Editor, select Rows > Add Field > Year to list all entries in order.

6) In Report Editor, select Values > Add Field > Year to summarize all values for each entry.

7) Change Summarize by SUM to Summarize by COUNTA (to count alphabetical or numerical entries), or COUNT (to count only numeric values).

More Advanced Pivot Table with Google Sheets

In addition to grouping by rows, you can create more advanced pivot tables by grouping by columns and filtering results. For example, the pivot table shown below shows rows by birth year, columns by gender (blank, female, male, other), and filters results to show only 18 students from one country: US.

Learn More

- Google, Create and Use Pivot Tables Help Page https://support.google.com/docs/answer/1272898
- LibreOffice, Creating Pivot Tables Help Page https://help.libreoffice.org/Calc/Creating_Pivot_Tables
- Andrew Ba Tran, "Tutorial: How to Make Pivot Tables in Google Sheets," TrendCT, September 4, 2015, http://trendct.org/2015/09/04/tutorial-how-to-make-pivot-tables-in-google-sheets

Match Columns with VLOOKUP

Here's a common problem: Sheet 1 contains a long roster of students enrolled in our *Data Visualization For All* course, with a two-letter code for their nation. Sheet 2 contains the list of codes for each nation. How can we quickly match up this information in one sheet, so that each row contains the nation for each student?

One solution: Spreadsheets contain a VLOOKUP function, which "looks up" data across two or more vertical columns, and automatically fills in matching

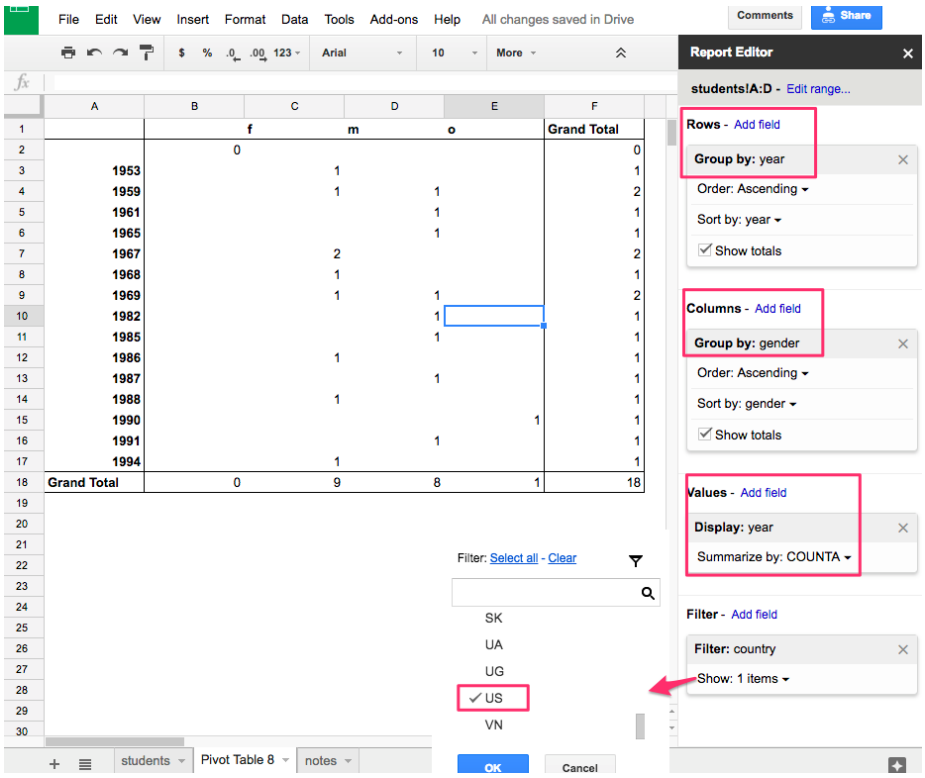


Figure 2.14: Screenshot: Advanced pivot table by year of birth and gender for US

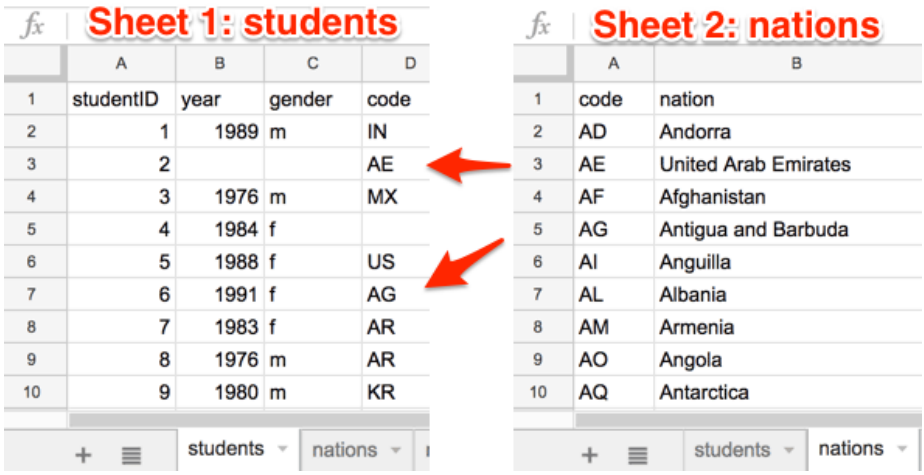
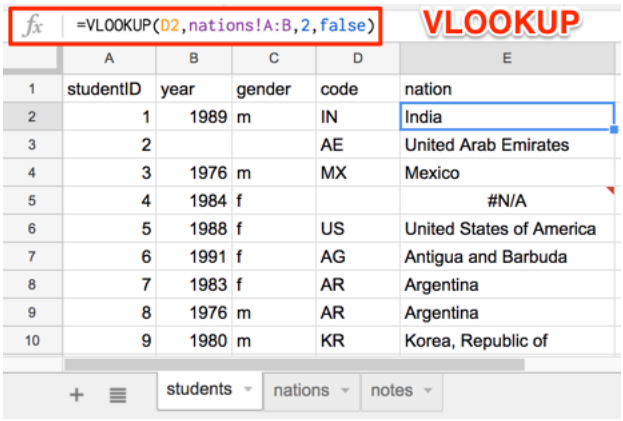


Figure 2.15: Screenshot: Problem - How to match columns in two sheets?

entries. This tutorial demonstrates how to set up this calculation in Google Sheets and Excel



	A	B	C	D	E
1	studentID	year	gender	code	nation
2	1	1989	m	IN	India
3	2			AE	United Arab Emirates
4	3	1976	m	MX	Mexico
5	4	1984	f		#N/A
6	5	1988	f	US	United States of America
7	6	1991	f	AG	Antigua and Barbuda
8	7	1983	f	AR	Argentina
9	8	1976	m	AR	Argentina
10	9	1980	m	KR	Korea, Republic of

Figure 2.16: Screenshot: Solution - Use the VLookup function

Video with step-by-step tutorial for Google Sheets

1) Click this link and Save to download to your computer: sample-students-nations in .ODS format. ODS means OpenDocument System, a generic multi-tab format that most spreadsheet tools can easily open.

2) To upload the downloaded file to Google Sheets, see the Upload Files and Convert tutorial in this book, and remember that Settings (gear symbol) must be set to Convert files to Google format. Or, open the file with Microsoft Excel or LibreOffice, and the directions below will be similar.

3) In the students sheet, type "nation" as a column header into cell E1.

4) Click in cell E2, start typing "=VLOOKUP" and the spreadsheet tool will suggest that you complete the formula in this format:

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

- search_key = the Sheet 1 cell we are trying to match
- range = the columns in Sheet 2 where matches may exist
- index = the column in the Sheet 2 range that contains the desired result, where 1 = first column, 2 = second column, etc.
- [is_sorted] = if the first column of the range is sorted, enter "true" to find the closest match; otherwise enter "false" to return exact matches only

4) You can type in the formula, or fill it out by clicking on cells, columns, and sheets as shown in the video above.

Collect and Share Data with Google Forms

TODO: write simple tutorial for Google Forms and explain how to share the spreadsheet; also mention other web form services

Chapter 3

Find and Know Your Data

Searching for open data: Increasing numbers of governmental agencies and non-profit organizations are publicly sharing *open data* on the web. When starting a new data visualization project, ask yourself these questions:

- Do I have the most relevant data for my project?
- Is it the most current data, in the most user-friendly format?
- Is data available at the individual level, or aggregated into larger groups?
- Which organizations might have collected data for my topic?
- Which open data repositories might have published this data?

What features do open repositories offer?

- View and export: At minimum, most open data repositories allow users to view their data and export it into common spreadsheet formats. Some also provide geographical boundaries for polygon maps.
- Built-in visualization tools: Some 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.
- Static and Live data: Most repositories offer static datasets for a specific time period, but some also provide "live" data that is continuously updated.
- Application Programming Interface (APIs): Some repositories provide endpoints with code instructions that allow users to pull data directly from the platform into an external sites or online visualization, which is ideal for continuously updated data.

Know Your Data

Before starting to create charts or maps, get to know your data.

- Where did it come from?
- Who compiled the data, and for what purpose?
- What do the data labels really mean?
- Ask yourself: Am I working with the *most* recent version, in the *best* available format?

TODO: add resource https://github.com/Quartz/bad-data-guide

open data inception 1600+ sites portal http://opendatainception.io/

- Know your data: go out into the field to directly observe how the original data is measured and collected

https://www.opendatanetwork.com/

Closely examine your data files to understand their meaning, sources of origin, and limitations. TODO: expand on this theme with examples of bad and misleading data

1) Always ask: Am I using the best available data?

- Compare the HFS list to the City of Hartford's current list of food establishments: https://data.hartford.gov/browse
- go to Public Health Category
- click on the "dataset" version (updated 10 Feb 2016), which is same data but different view than the "map" version
- click on light blue "export" button into any format you wish to compare with the HFS list (see screenshot)
- decide which list is best for your organization's goal

US and Census Bureau Open Data

The U.S. Census Bureau collects and shares population, housing, and economic data on its open repositories.

- The Decennial Census is a full count of the population every ten years, most recently in 2010 and the upcoming one in 2020. Because decennial data are counts and not estimates, they represent "true" values and hence come without margins of errors.

- The American Community Survey (ACS) (https://www.census.gov/programs-surveys/acs/) is annual sample count, which produces:
  - 1-year estimates for areas with populations of 65,000+
  - 5-year estimates for all census areas
  - ACS used to release 3-year estimates for geographies with population of 20,000+, but discontinued after the 2011-2013 release.

Because ACS produces estimates and not "true" counts, data comes with margins of errors. Generally, margins of errors are higher for smaller geographies (eg census blocks) and smaller values (eg the number of Asian females aged 60+ who live in Union, CT). Hence, one needs to be critical when using ACS or other survey data.

Census areas are geographic divisions in this *general format*:

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

Census areas in the Hartford region

The interactive map below illustrates hierarchical relations among geographical census entities for the Hartford region, from state to census block level.

Learn more: Explore the standard hierarchy of US Census geographic entities and definitions (https://www2.census.gov/geo/pdfs/reference/geodiagram.pdf)

See also in this book: Geocode addresses with the US Census Geocoder

Data.census.gov

Data.census.gov (https://data.census.gov) is the main platform to access US Census data. It provides an easy search across census and survey tables. There is an interface to view tables for various years and geographies, and a download button to save data as CSV or PDF. It replaced American FactFinder (https://factfinder.census.gov) in July 2019.

Social Explorer

Social Explorer (https://www.socialexplorer.com/) is a popular tool to view and download census and related demographic data, past and present. The platform

allows users to create data maps that may be exported as static images or presentation slides. Social Explorer requires subscription, but many academic institutions provide access.

TODO: create tutorial on how to cleanly download census data from Social Explorer and Census.gov to join with geography, especially census tract numbers

Data.gov

Data.gov (https://www.data.gov/) is the official open data repository for US federal government agencies, managed by the US General Services Administration, and powered by an open-source CKAN and WordPress platform.

National Center for Education Statistics

National Center for Education Statistics (NCES) (https://nces.ed.gov/) is the primary federal agency for collecting and reporting education data.

- Elementary/Secondary Information System (ELSi) (https://nces.ed.gov/ccd/elsi) - create custom tables and charts from the Common Core of Data (CCD) and Private School Survey.

Boundaries

- TODO
- link and source files and scale
- http://mapstarter.com/

Source Your Data Files

Source your data. Spell out exactly where it came from, so that someone other than you, several years in the future, could understand its origin.

Label the file name

Everyone has seen examples of bad file names:

- data.xls
- bldgdatalist.csv
- data77.xls

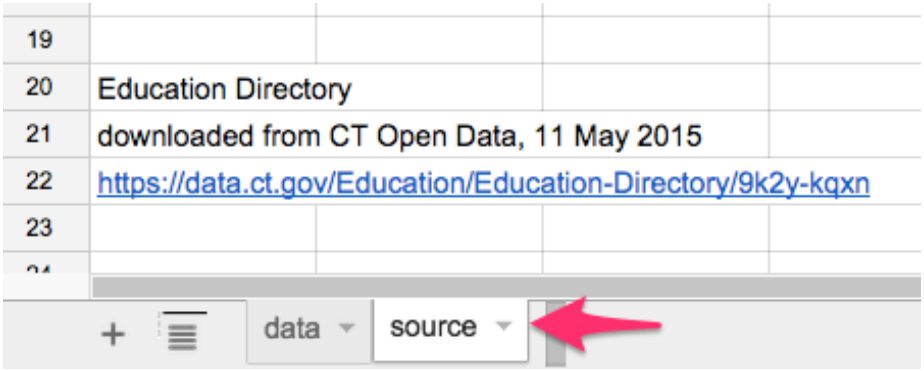
Write a short but meaningful file name. It is a good idea to include data source in file name (eg `acs2018`, `worldbank`, or `eurostat`). If different versions of the data are floating around, add the current date at the end, in YYYY-MM-DD format. Good file names look like this:

- town-demographics-2019-12-02.xls
- census2010_population_by_county.csv
- eurostat-1999-2019-CO2_emissions.xlsx

Save source data in separate sheet

Before modifying the original dataset, make sure to duplicate it to avoid any data losses. One way is to click (or right-click) on the spreadsheet tab to copy the sheet to another tab as a backup.

Add a *source* tab, after the data, with notes to remind you and others about its origins and when it was last updated.



Learn more

Lisa Charlotte Rost, *How to prepare your data for analysis and charting in Excel & Google Sheets*, https://blog.datawrapper.de/prepare-and-clean-up-data-for-

data-visualization/

TODO: Source your data

```
- explain that data cannot be copyrighted, but representations of data can be
- open-source and creative commons
- credit sources and collaborators on dataviz products and readme files
- Whose perspectives does your data privilege? Whose stories remain untold?
```

Public or Private Data?

Many of the free web-based tools in this book require that your publicly share your data. Check each tool and decide whether it is appropriate for your data, which may have some privacy restrictions.

In some cases, individual data privacy is protected by law, but a government agency may aggregate (sort into larger groups) or anonymize (remove personally identifiable details) data to make it public. For example:

- Individual-level census data is private for about 70 years, but the US Census Bureau publicly releases anonymous data for aggregated areas (such as census blocks, tracts, towns, etc.)
- Patient-level health records are private, but public health officials share town- and county-level health data.
- Student-level education data is private, but school districts and state agencies publicly share grade-level and school-level data.

In other cases, individual data is not private. For example:

- When individuals contribute to political campaigns, most US and state laws require that the donor name, address, and amount is public data.
- When an individual buys home in Connecticut, the owner's name, address, purchase amount, and other details about the home are public data.

Know Your Data: Is It Good or Bad?

Before starting to create charts or maps, get to know your data.

- Where did it come from?
- Who compiled the data, and for what purpose?
- What do the data labels really mean?
- Ask yourself: Am I working with the *most* recent version, in the *best* available format?

Closely examine your data files to understand their meaning, sources of origin, and limitations. TODO: expand on this theme with examples of bad and misleading data

TODO: cite and explain this resource https://github.com/Quartz/bad-data-guide

Learn more

Christopher Ingraham, *An alarming number of scientific papers contain Excel errors*, https://www.washingtonpost.com/news/wonk/wp/2016/08/26/an-alarming-number-of-scientific-papers-contain-excel-errors/

Connecticut Open Data

Since this book was created in Hartford, Connecticut, we include state and municipal open data repositories and boundary files.

Connecticut Open Data (http://data.ct.gov), the official portal for state government agencies, is hosted on the Socrata platform, which offers built-in data visualization tools and APIs. See also how to create a filtered point map with Socrata in this book.

See also separate repositories for individual state agencies:

- Office of the State Comptroller (http://www.osc.ct.gov/openCT.html)
- CT State Department of Education (http://www.sde.ct.gov/sde/cwp/view.asp?a=2758&q=334520)
- Office of Policy and Management (http://ct.gov/opm/cwp/view.asp?a=3006&Q=383258&opmNav_GID=1386)
- link to all CT state government agencies (http://portal.ct.gov/Department-and-Agencies/)

Connecticut State Data Center (http://ctsdc.uconn.edu/), part of the U.S. Census Data Center Network, is the lead agency for US Census data and other socioeconomic data for Connecticut, and is based at the University of Connecticut Libraries. The site also features data visualizations created on the Tableau platform and provides population projections for the state of Connecticut.

MAGIC: The Map and Geographic Information Center (http://magic.lib.uconn.edu), based at the University of Connecticut Libraries, specializes in providing geographic, aerial photography, and map images for the state, past and present. The site also features interactive maps.

DataHaven (http://ctdatahaven.org/), a non-profit organization, collects and interprets information about Connecticut neighborhoods, such as its Community Wellbeing Survey. Data resources feature neighborhood profiles for densely-populated areas (New Haven and Hartford-West Hartford), and town profiles for other areas across the state.

Connecticut Data Collaborative (http://ctdata.org) is a public-private partnership that advocates for open data access to drive planning, policy, budgeting and decision making in Connecticut at the state, regional and local levels. We democratize public data through custom data exploration tools and a dynamic town profile tool, hosted on the open-source CKAN platform. Users can find state and federal data on topics such as public health, education, crime, municipal data, and racial profiling data.

Hartford Data (http://data.hartford.gov), the official portal of the City of Hartford municipal government, is hosted on the Socrata platform, which features built-in visualizations and APIs. See also how to create a filtered point map with Socrata in this book. Also, the Hartford Data site links to the City's ArcGIS Online geographic data (http://gisdata.hartford.gov/) and the City's financial data (http://checkbook.hartford.gov/) and budget (http://budget.hartford.gov/).

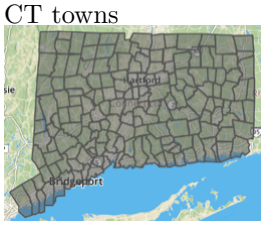
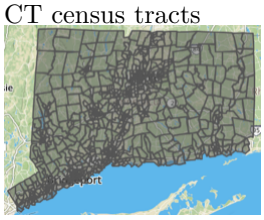
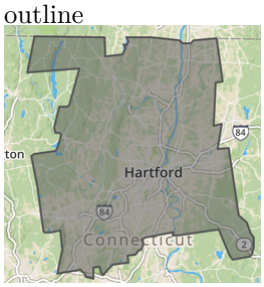
In addition to the official repositories above, Connecticut news organizations that create data visualizations often include links to download data files.

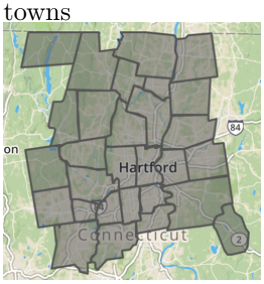
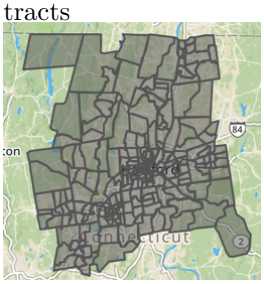
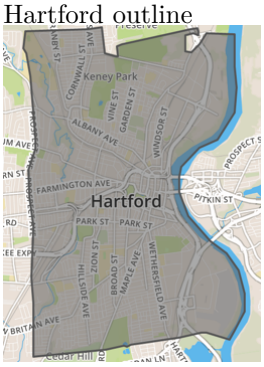
Connecticut Mirror / Trend CT (http://ctmirror.org/) and (http://trendct.org/) are publications of the Connecticut News Project, an independent, nonpartisan, nonprofit organization that focuses on state policy issues. Most of their data visualizations are built with open-source code, with publicly accessible data files. See also their GitHub repository (https://github.com/trendct).

Hartford Courant Data Desk (http://www.courant.com/data-desk) produces digital visualizations for the *Hartford Courant*, the largest daily newspaper in Connecticut, owned by Tribune Publishing. Many of these data visualizations are published on the Tableau platform, which allows readers to download the underlying data.

Boundaries

- Converted from shapefile WGS84 to GeoJSON format
- To download a GeoJSON file, right-click the link and Save to your computer
- If you accidentally open the GeoJSON code in your browser, select File > Save Web Page to download it
- To view or edit, drag files into http://geojson.io or http://mapshaper.org
- Learn more in the Transform Your Map Data chapter of this book

Geography	Year-Source-Size	Right-click + Save to download GeoJSON
CT outline 	2010 Census UConn MAGIC WGS84 1:100,000	ct-outline.geojson
CT counties 	2010 Census UConn MAGIC WGS84 1:100,000	ct-counties.geojson
CT towns 	2010 Census UConn MAGIC WGS84 simplified to 224k	ct-towns.geojson
CT census tracts 	2010 Census UConn MAGIC WGS84 1:100,000	ct-tracts-2010.geojson
Hartford County outline 	2010 Census UConn MAGIC WGS84 1:100,000	hartfordcounty-outline.geojson

Geography	Year-Source-Size	Right-click + Save to download GeoJSON
Hartford County towns 	2010 Census UConn MAGIC WGS84 1:100,000	hartfordcounty-towns.geojson
Hartford County tracts 	2010 Census UConn MAGIC WGS84 1:100,000	hartfordcounty-tracts-2010.geojson
Hartford outline 	2010 Census UConn MAGIC WGS84 1:100,000	hartford-outline.geojson

Geography	Year-Source-Size	Right-click + Save to download GeoJSON
Hartford census tracts 	2010 Census UConn MAGIC WGS84 1:100,000	hartford-tracts-2010.geojson
Hartford neighborhoods 	2015 Hartford Open Data 1:50,000	hartford-neighborhoods.geojson

TODO: - add Capitol Region Council of Governments (CRCOG) http://www.crcog.org/ - add school districts (and clarify elementary-secondary) - add Capitol Region Education Council (CREC) http://www.crec.org/ - add school attendance areas from federal site - describe Freedom of Information Act (FOIA) data requests in Connecticut

Chapter 4

Clean Up Messy Data

More often than not, datasets will be messy and hard to visualize right away. They will have missing values, various spelling of the same categories, dates in different formats, text in numeric-only columns, multiple things in the same columns, and other unexpected things (see Figure 4.1 for inspiration). Don't be surprised if you find yourself spending longer cleaning up data than actually analyzing and visualizing it—it is often the case for data analysts.

Year	City	Amount
1990	New York City	$1,123,456.00
1995-96		2.2 mil
2000s	NYC	No data
2020	New_York	50000001

Figure 4.1: More often than not, raw data looks like this.

It is important to learn several tools in order to know which one to use to clean your data efficiently. We will start by looking at fairly basic data cleanup using Google Sheets. Keep in mind that the same principles (and in most cases even the same formulas) can be use in Microsoft Excel, LibreOffice Calc, Mac's Numbers, or other spreadsheet packages.

We will then show you how to extract table data from PDF documents using a free tool called Tabula. Tabula is used by data journalists and researchers worldwide to analyze government spendings, procurement records and all sorts of other datasets that get trapped in PDFs.

At the end, we will introduce OpenRefine, an extremely powerful and versatile tool to clean up the messiest spreadsheets, such as those containing dozens of misspelled versions of universities or town names.

Clean Data with Spreadsheets

Let's take a look at some techniques to clean up data directly in your favorite spreadsheet tool. We will use Google Sheets as an example, but the same principles will apply to most other software packages, such as Excel, Calc, or Numbers.

Find and Replace with a blank

A common problem with census data is that geographic names contain unnecessary words. For example, your data can look something like that:

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

But you want a clean list of towns, either to display in a chart, or to merge with a different dataset:

```
Hartford
New Haven
Stamford
```

Here's one quick solution: In any spreadsheet tool, use the Find and Replace command to remove unwanted characters. You can download our sample file, which contains 169 Connecticut towns and their population, for the exercise.

1. Select the column you want to modify by clicking on the column header. If you don't, you will be searching and replacing in the entire spreadsheet.
2. From *Edit* menu, choose *Find and replace* item. You will see the window like is shown in Figure 4.2.
3. In the *Find* field, type `town`, without quotation marks and leaving a space before the word. If you don't leave the space, you will accidentally remove *town* from *Newtown*, and you will end up with trailing spaces which can cause troubles in the future.
4. Leave the *Replace with* field blank.
5. *Search* field should be set to the range you selected in step 1, or *All sheets* if you didn't select anything.
6. 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.
7. Press the *Replace all* button. Since this sample file contains 169 towns, the window will state that 169 instances of "town" have been replaced.

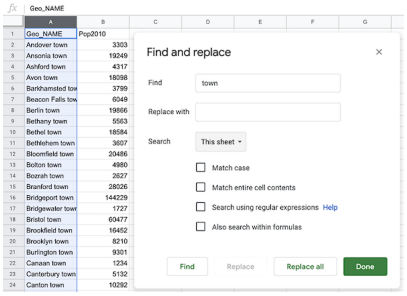


Figure 4.2: Find and Replace window in Google Sheets.

Split data into two or more columns

Sometimes multiple pieces of data appear in a single cell, such as names (`John Doe`), coordinate pairs (`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 Let's begin with a simple example of coordinate pairs. You can use our sample file, which is a collection of latitude-longitude pairs separated by a comma, with each pair on a new line.

1. 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.
2. 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.
3. Go to *Data* and select *Split text to columns*, as in Figure 4.3.
4. Google Sheets will try to guess your separator automatically. 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 — more about that in Example 2 of this section).
5. You can rename columns into *Longitude* (first number) and *Latitude* (second number).

Example 2 Now, let's look at a slightly more complicated example. Imagine your dataset is structured as follows:

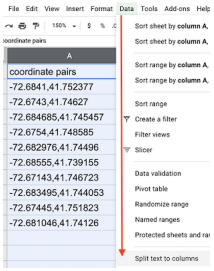


Figure 4.3: Select Data > Split text to columns to automatically separate data.

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

Each cell contains a full address, but you want to split it into four cells: street address (300 Summit St), city (Hartford), state (CT), and zipcode (06106). Notice that the separator between the street and the rest of the address is a comma, a separator between the city and state is a space, and there are two dash lines between state and zipcode.

1. Start splitting left to right. So your first separator will be a comma. Select your column (or one or more cells within that column), and go to *Data > Split text to columns*.
2. Google Sheets should 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).
3. Now, select only the second column and perform *Split text to columns*. You will see that the city is now separate from the state and zipcode, and Google Sheets chose space as a separator (if it didn't, choose *Space* from the dropdown menu).
4. Next, select only the third column and perform *Split text to columns* again. Google Sheets won't recognize `--` as a separator, so you will have to manually select *Custom*, type `--` in the text field, and hit Enter. You should now have four columns.

Tip: Google Sheets will treat zipcodes 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 concatenating 0s using the formula introduced in Combine separate columns into one.

Combine separate columns into one

Now, let's see how to perform the reverse action. Imagine you receive address data in separate columns, formatted like this:

```
| Street        | City        | State      | Zip        |
| 300 Summit St | Hartford    | CT         | 06106      |
```

The data comes is four columns: street address, city, state, and zipcode. Let's say your mapping tool requires you to combine all of this terms into one location column, like that:

```
| Location                          |
| 300 Summit St, Hartford, CT 06106 |
```

You can write a simple formula to combine (or concatenate) terms using ampersands (`&`) as cells values connectors, and quoted spaces (`" "`), or spaces with commas (`", "`), or a dash with spaces on both sides (`" - "`), or anything else as term separators.

For example, imagine that a spreadsheet contains an address that is separated into four columns—*Address, City, State,* and *Zip*—as shown in columns A-D in Figure 4.4. In column E, you can add new header named *Location* and insert a formula in this format, to combine the items using ampersands (`&`) and separating them with quoted spaces (`" "`), like this: `=A2&" "&B2&" "&C2&" "&D2`

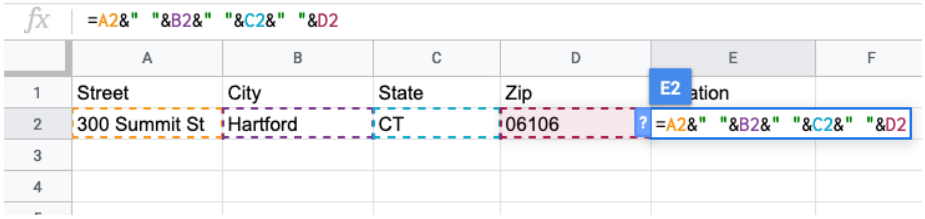


Figure 4.4: Use ampersands to combine items and separate them with spaces.

You are now able to split data to columns using custom separators, and concatenate values from different cells into one. But what if your table is trapped inside a PDF? In the next section, we will introduce Tabula and show you how to convert tables from PDF documents into tables that you can analyze in Google Sheets, Microsoft Excel, or similar packages.

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.

Tabula is a free tool that runs on Java, 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.

Note: Keep in mind that PDFs generally come in two flavors, image-based and text-based. You know your PDF is text-based if you can use cursor to select and copy-paste text. These are great for Tabula. Image-based PDFs are those that were created from scanning documents. Before they can be processed with Tabula, you will need to use an optical character recognition (OCR) software, such as Adobe Acrobat, to create a text-based PDF.

Set Up Tabula

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 prompted with a security message (such as "Tabula is an app downloaded from the internet. Are you sure you want to open it?"), follow the instruction to proceed (on a Mac, click *Open*; you might have to go to System Preferences > Security & Privacy, and resolve the issue there).

Your default system browser should open, like shown in Figure 4.5. The URL will be something like `http://127.0.0.1:8080/`, meaning Tabula is running on your local machine. 127.0.0.1, also known as `localhost`, is the hostname for your machine. `8080` is called port (it's okay if you see a different port—most likely because 8080 was taken by some other program running on your computer). If for any reason you decide to use a different browser, just copy-paste the URL.

Load a PDF and Autodetect Tables

Since the beginning of the Covid-19 pandemic, the Department of Public Health in Connecticut has been issuing daily PDFs with case and death count by town. For the demonstration, we will use one of those PDFs from May 31, 2020.

1. Select the PDF you want to extract data from by clicking the blue *Browse...* button.

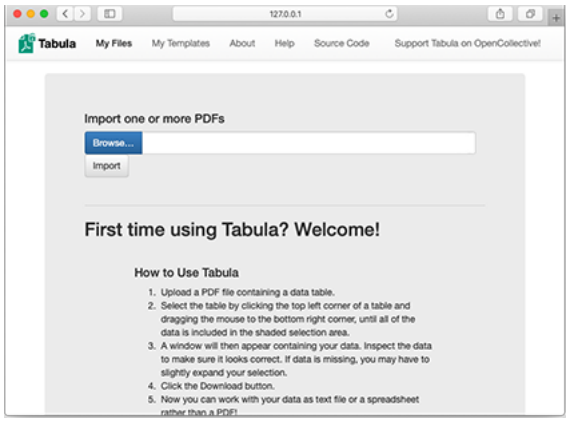


Figure 4.5: Tabula welcome page.

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. The easiest first step is to let Tabula autodetect tables. Click the relevant button in the header. You will see that each table is highlighted in red, like shown in Figure 4.6.

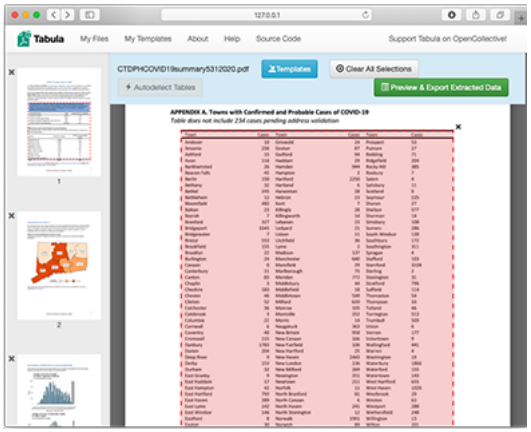


Figure 4.6: Selected tables are highlighted in red.

Manually Adjust Selections and Export

1. Click *Preview & Export Extracted Data* green button to see how Tabula thinks the data should be exported.
2. 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.
3. If the tables still don't look right, or you with to remove some tables that Tabula auto-detected, hit *Revise selection* button. That will bring you back to the PDF viewer.
4. 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).
5. 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.
6. 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 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 a Downloads folder on your computer (or wherever you chose to save it). It is ready to be opened in Google Sheets or Microsoft Excel, analyzed, and visualized! In the following section, we are going to look how to clean up messy datasets with OpenRefine.

Clean Data with OpenRefine

Consider a dataset that looks like the one in Figure 4.7. Can you spot any problems with it?

Notice how the funding amounts (last column) are not standardized. Some amounts have commas as thousands separators, some have spaces, and some start with a dollar character. Notice also how the Country column includes various spellings of North and South Korea. Datasets like this can be an absolute nightmare to analyze. Luckily, OpenRefine provides powerful tools to clean up and standardize such data.

Note: This data exerpt 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 (although we did not alter values).

	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 4.7: First 20 rows of the sample dataset. Can you spot any problems with it?

Download this sample dataset or use your own file with messy data. Inspect the file in any spreadsheet software. You can see that the dataset has four columns: year (between 2000 and 2018, inclusive), country (North or South Korea), a US funding agency, and funding amount (in 2018 US dollars). Let's now use OpenRefine to clean it up.

Set up OpenRefine

You can download a copy of OpenRefine for Linux, Mac, or Windows from the official download page. 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. Go to System Preferences -> Security and Privacy, and hit *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), like shown in Figure 4.8.

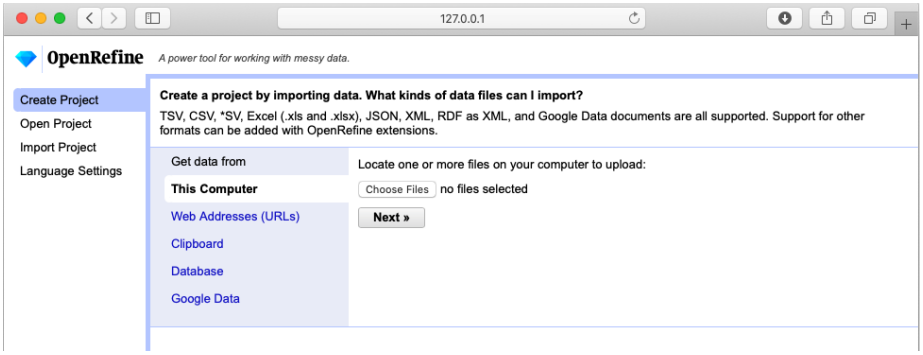


Figure 4.8: OpenRefine starting page.

Load Data and Start a New Project

To begin cleaning up your messy dataset, you should load it into a new project. OpenRefine lets you upload a dataset from your local machine, or a remote URL on the web (including a Google Spreadsheet), or copy/paste data into a text field. OpenRefine is able to extract data directly from SQL databases, but this is beyond the scope of this book. We assume that you downloaded the sample dataset we provided (or you are using your own file), so let's load it from your computer.

1. Under *Get data from: This computer*, click *Browse...* and select the file. Click Next.
2. 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 4.9.
3. Hit *Create Project* in the upper-right corner.

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

Convert Dollar Amounts from Text to Numbers

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

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

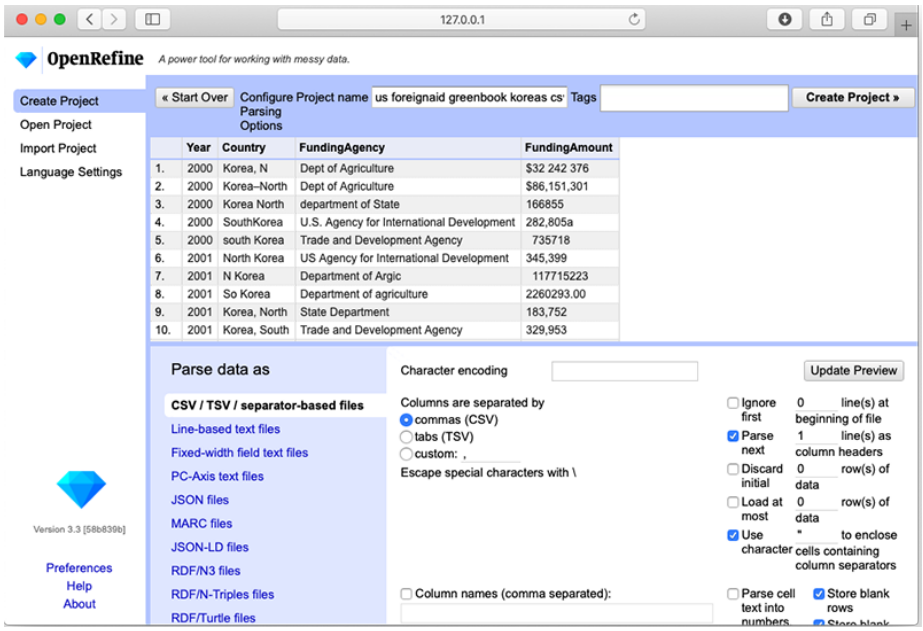


Figure 4.9: OpenRefine parsing options.

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. Perform Step 1 again. You will see that all but three cells turned green. That is because we have 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 4.10).

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.

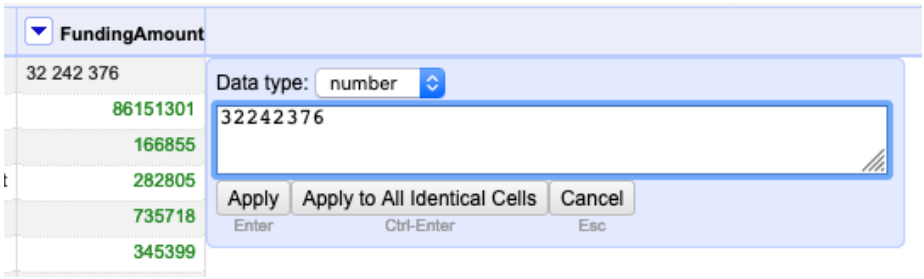


Figure 4.10: Manually remove spaces and extra characters, and change data type to number.

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 4.11.
2. You will have a choice of two clustering methods, *key collision* or *nearest neighbor*. Both methods can be powered by different functions, but let's leave the default *key collision* 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. 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.

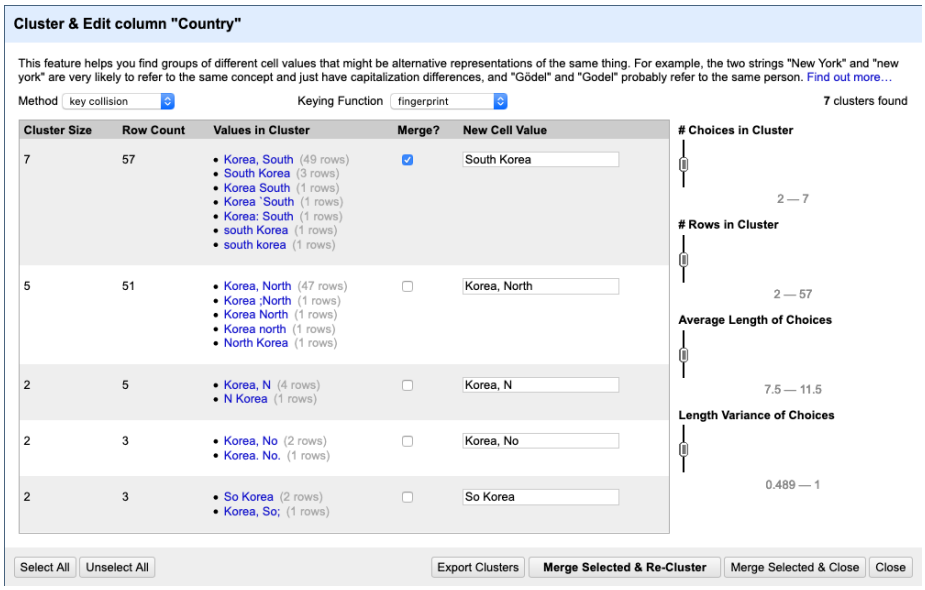


Figure 4.11: Cluster similar text values.

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.

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.

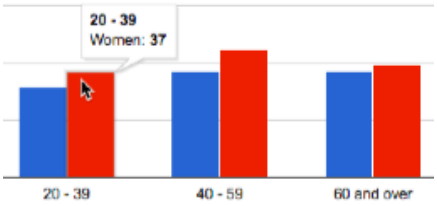
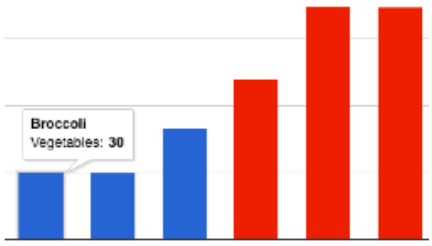
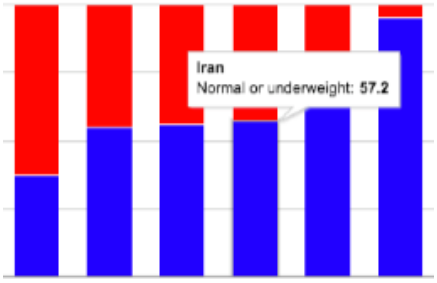
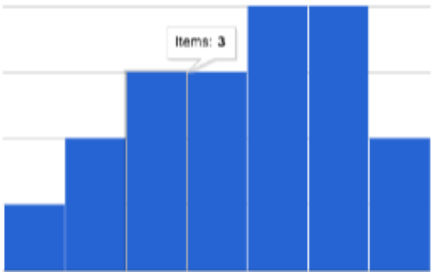
Chapter 5

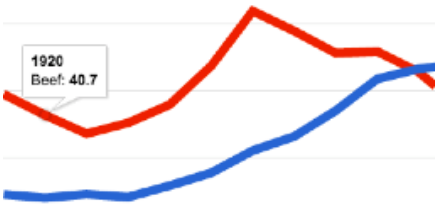
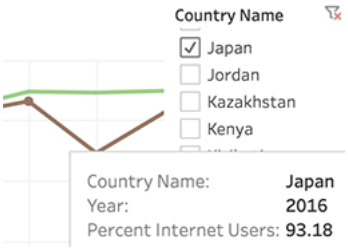
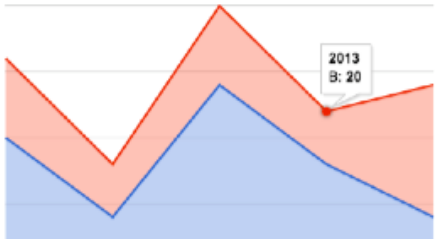
Chart Your Data

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. In addition to static chart images, this book focuses on interactive charts that display more data when you float your cursor over them in your web browser. Later you'll learn how to embed interactive charts on your web in chapter 7.

To begin, this grid of basic chart types will help you decide which type you wish to create. Your decision will be based on the format of your data, and 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 to create it. This chapter features easy-to-learn drag-and-drop tools, such as the Google Sheets chart tool, and for more advanced charts, Tableau Public, the free version of the powerful software used by many data analysts and visualization practitioners. The grid also refers to more powerful chart tools, such as Chart.js code templates in chapter 9, which give you ever more control over how your design and display your data, but also require learning how to edit and host code templates with GitHub in chapter 8.

Basic chart types	Best use and tutorial chapters
Grouped column or bar 	Best to compare categories side-by-side. Vertical columns, or horizontal bars for long labels. Easy tool: Google Sheets bar and column tutorialPower tool: Chart.js templates
Separated column or bar 	Best to compare categories in separate clusters. Vertical columns, or horizontal bars for long labels.Easy tool: Google Sheets bar and column tutorialPower tool: Chart.js templates
Stacked column or bar 	Best to compare sub-categories, or parts of a whole. Vertical columns, or horizontal bars for long labels.Easy tool: Google Sheets bar and column tutorialPower tool: Chart.js templates
Histogram 	Best to show distribution of raw data, with number of values in each bucket.Easy tool: Google Sheets bar and column tutorialPower tool: Chart.js templates

Basic chart types	Best use and tutorial chapters
Pie chart 	Best to show parts of a whole, but hard to estimate size of slices.Easy tool: Google Sheets pie chart tutorialPower tool: Chart.js templates
Line chart 	Best to show continuous data, such as change over time.Easy tool: Google Sheets line chart tutorialPower tool: Chart.js templates
Filtered line chart 	Best to show multiple lines of continuous data, with on-off toggle buttons. Easy tool: Tableau Public filtered line chart tutorial
Stacked area chart 	Best to show parts of a whole, with change over time. Easy tool: Google Sheets stacked area tutorialPower tool: Chart.js templates

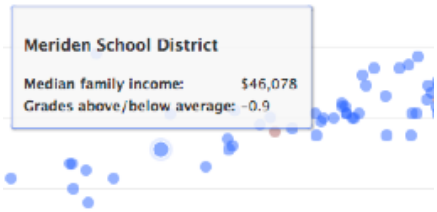
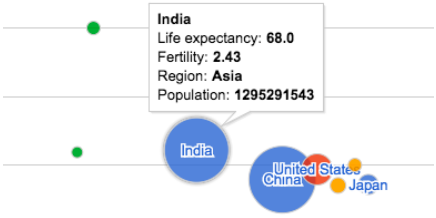
Basic chart types	Best use and tutorial chapters
XY Scatter chart 	Best to show relationship between two sets of data. Easy tool: Google Sheets scatter chart tutorial or Tableau Public scatter chart tutorialPower tool: Chart.js templates
Bubble chart 	Best to show relationship between three or four sets of data, using bubble size and color.Easy tool: Google Sheets bubble chart tutorialPower tool: Chart.js templates

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 5.1. It shows basic chart components that are shared among most chart types.

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, "Black and Asian Population More Likely to Die of Covid-19", or "Millions of Tons of Plastic Enter the Ocean Every Year" are both clear titles.

Sometimes a more "dry" and "technical" title is preferred. Our two titles can then be changed to "Covid-19 Deaths by Race in New York City, March 2020"

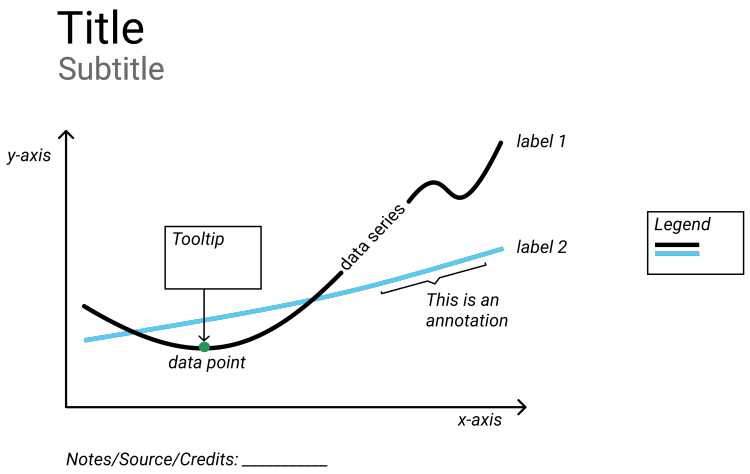


Figure 5.1: Common chart components.

and "Tons of Plastic Entering the Ocean, 1950–2020", respectively.

Often these two styles are combined into a title ("story") and a subtitle ("technical"), like that:

```
Black and Asian Population More Likely to Die of Covid-19
Covid-19 Deaths by Race in New York City, March 2020
```

Make sure your subtitle is less prominent than the title. You can achieve this by decreasing font size, or changing font 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 any *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.

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 de-clutter 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 chars must start at zero

Unlike line charts, bar or column charts need to have their value axis start at zero. This is to ensure that a bar twice the length of another bar represents twice its value. The Figure 5.2 shows a good and a bad example.

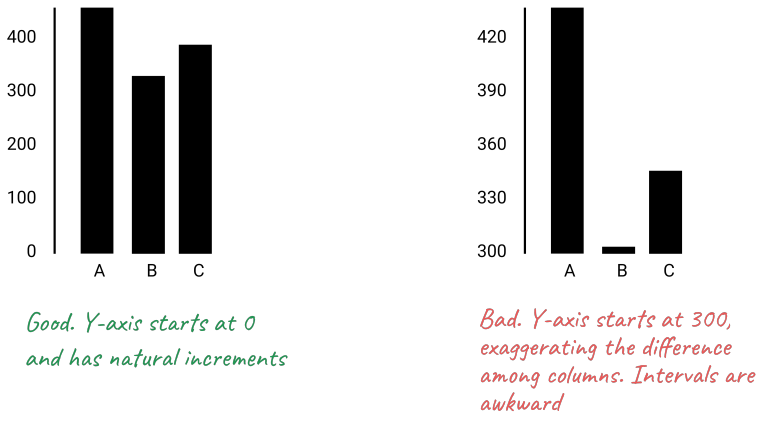


Figure 5.2: Start your bar chart at zero.

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 12.

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 5.3, 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.

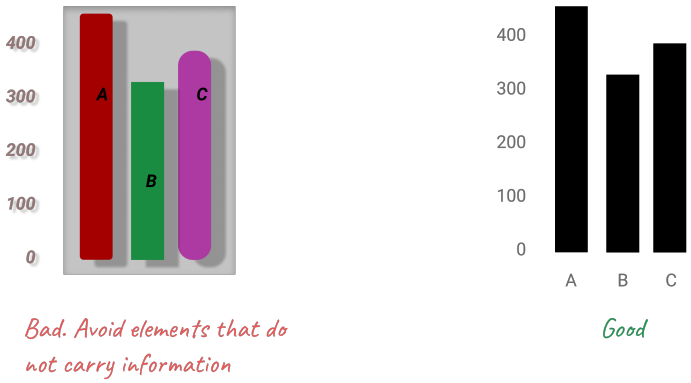


Figure 5.3: Avoid chart junk.

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. And don't let anyone tell you otherwise.

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 5.4 illustrates that.

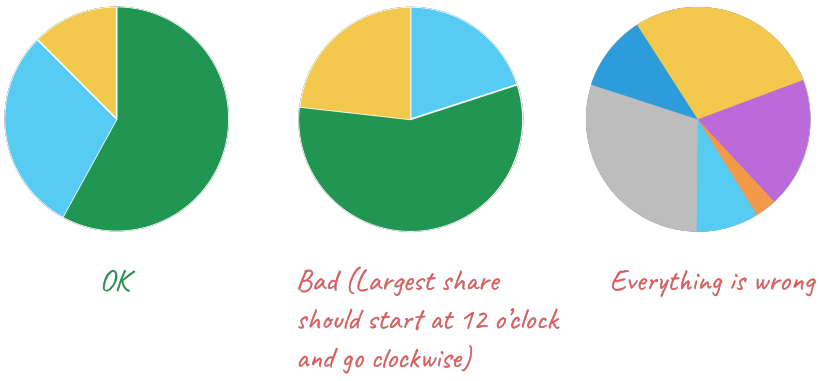


Figure 5.4: Sort slices in pie charts from largest to smallest, and start at 12 o'clock.

If your pie chart has more than five slices, consider showing your data in a bar chart, either stacked or separated, like Figure 5.5 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 5.6 to see how much easier it is to read horizontally-oriented labels.

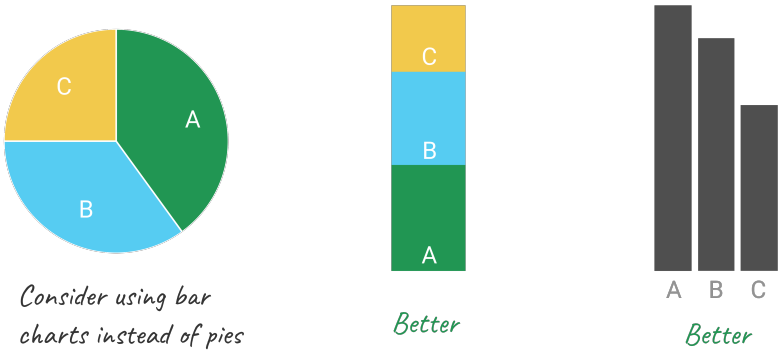


Figure 5.5: Consider using bar charts instead of pies.

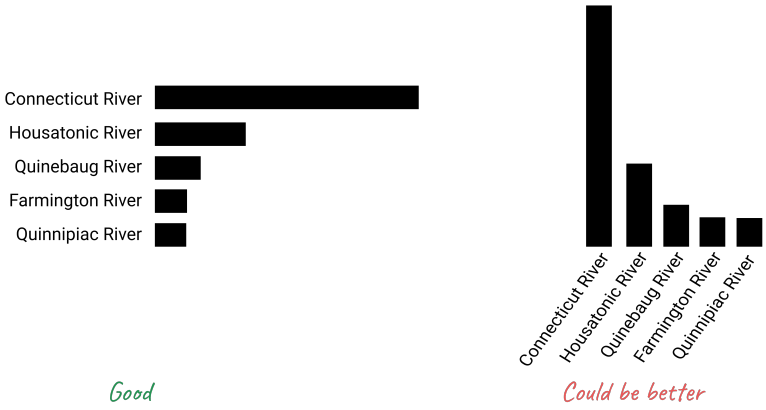


Figure 5.6: For long labels, use horizontal bar charts.

Arrange elements logically

If your bar chart shows different categories, consider ordering them, like is shown in Figure 5.7. 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.

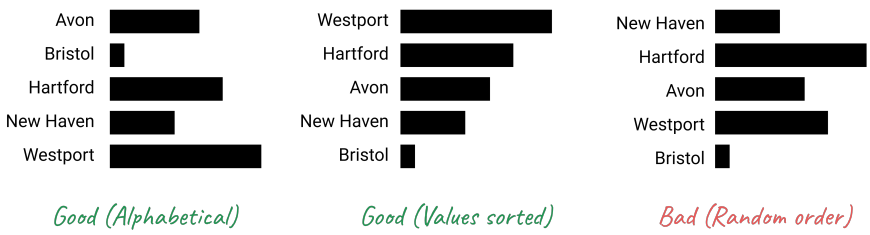


Figure 5.7: For long labels, use horizontal bar charts.

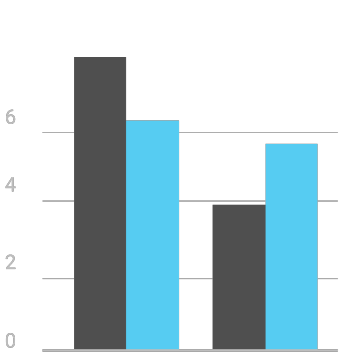
Do not overload your chart

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, use (but do not overuse) **bolding** to highlight major insights. Consider using commas as thousands separators for readability (`1,000,000` is much easier to read than `1000000`).

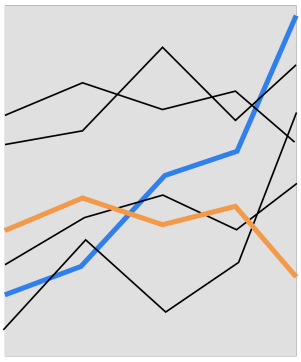
Be careful with the colors

The use of color is a complex topic, and there are plenty of books and research devoted to it. 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.

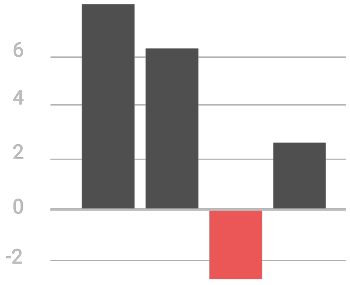
Whatever colors you end up choosing, they need to be distinguishable (otherwise what is the point?). Do not use colors that are too similar in hue (for example, various shades of green—-leave them for choropleth maps). Certain color combinations are hard to interpret for color-blind people, like green/red



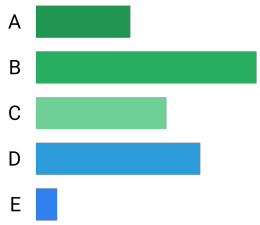
Good. Blue allows to distinguish between series



Good. Colored lines are distinguishable



Fine. Red adds emphasis, but is not absolutely necessary



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

Figure 5.8: Don't use colors just for the sake of it.

or yellow/blue, so be very careful with those. Figure 5.8 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 5.9. Notice how your eyes are drawn to the bars and their corresponding values, not bright colors or secondary components like the axes lines.

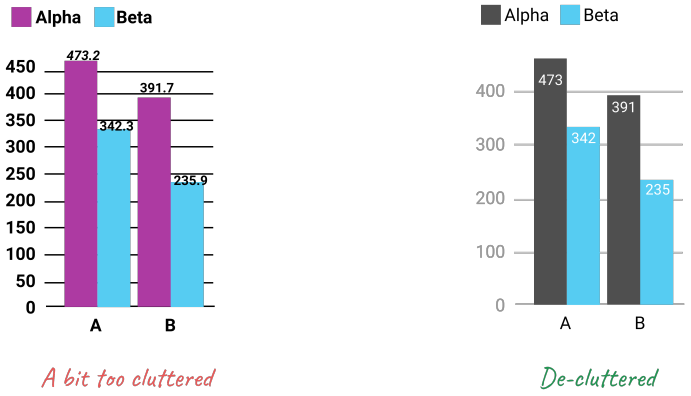


Figure 5.9: Make sure important things catch the eye first.

Google Sheets Charts

Google Sheets is a well-known spreadsheet program that allows you to create basic charts using its intuitive drag-and-drop interface. Most people who create charts with Google Sheets export them as static *png* images. But in fact these interactive charts can be easily embedded on your website, as you'll learn in chapter 7.

In this section, we will look at creating column and bar charts that are separated, grouped, and stacked. We will also look at making pie, line, area, and scatter charts, and learn to visualize three-dimensional data using bubble charts.

As most easy-to-use tools, Google Sheets has its shortcomings. You won't be able to control tooltips of scatterplot tooltips, or cite or link to source data inside charts. You won't be able to annotate or highlight items. But you *will* be able to *quickly* make good-looking interactive charts *quickly*.

Tip: Visit Types of charts & graphs in Google Sheets for an overview of the various chart types supported by this tool.

Column and Bar Charts with Google Sheets

Column and bar charts are some of the most common types of charts in data visualization (column charts are just vertical bar charts). They are used to compare values across categories.

In this tutorial, we will use three small datasets to build interactive separated, grouped, and stacked bar charts in Google Sheets:

- Obesity in the US (by US CDC and StateOfObesity.org project)
- High-Calorie Fast-Food Items
- Global Database on Body Mass Index by World Health Organization

Before you begin, you will need to create a free Google Drive account. If you already Google Mail, you can use the same username for your Google Drive account.

Grouped Column and Bar Charts

Figure 5.10 shows differences in obesity between men and women, grouped together in three age brackets to allow for easier gender comparisons across the same ages. In the interactive web version, hover over columns and see tooltips with data.

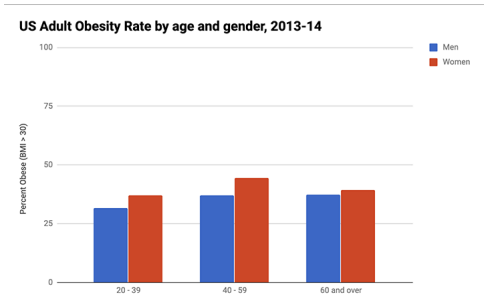


Figure 5.10: Grouped column chart with data from StateOfObesity.org. Explore the full-screen interactive version.

The following steps will help you recreate an interactive grouped column (or horizontal bar) chart.

1. Right-click to open link in new tab: Google Sheet Column chart with grouped data template
2. Sign in to continue to Google Sheets (which is part of Google Drive). If you don't already have a Google account, you can create one.

3. Select File > Make a Copy to save your own version to your Google Drive, as shown in Figure 5.11.

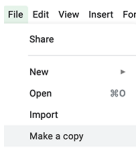


Figure 5.11: Make your own copy of the Google Sheet template.

4. 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 menu appear, and select Delete, as shown in Figure 5.12.

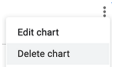


Figure 5.12: Float cursor in top-right corner of the chart to make the 3-dot menu appear, and select Delete.

5. Format your data to make each column a data series, as shown in Figure 5.13, which means it will display as a separate color in the chart.

	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 5.13: Format data in columns to make colored grouped columns in your chart.

6. 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 5.14.

7. In the Chart Editor, change the default selection to Column chart, with Stacking none, to display Grouped Columns, as shown in Figure 5.15. 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 5.16.

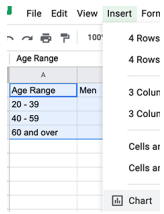


Figure 5.14: Select your data and then Insert > Chart.

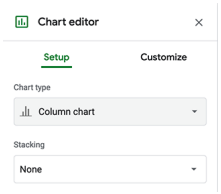


Figure 5.15: Change the default to Column chart, with Stacking none.

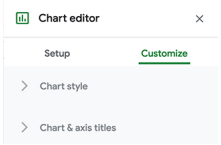


Figure 5.16: Select Customize to edit title, labels, and more.

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 5.17. This means your sheet is no longer Restricted to only you, but can be viewed by anyone with the link. See additional options.

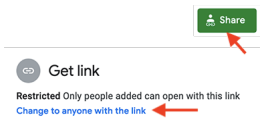


Figure 5.17: Click the Share button and then click "Change to anyone with the link." to make your data public.

10. To embed an interactive version of your chart in another web page, click the 3-dot menu in the upper-right corner of your chart, and select Publish Chart, as shown in Figure 5.18. In the next screen, select Embed and press the Publish button. See Chapter 7 Embed on the Web to learn what to do with the iframe code.

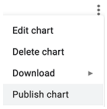


Figure 5.18: Select Publish Chart to embed an interactive chart on another web page, as described in Chapter 7.

Note: Currently, there is no easy way to cite or link to your source data inside a Google Sheets chart. Instead, cite and link to your source data in the text of the web page. Remember that citing your sources adds credibility to your work.

Separated Column and Bar Charts

When you visualize independent categories of data, and you don't want them to appear grouped together, then create a chart with separated columns (or horizontal bars, if you have long data labels). For example, Figure 5.19 is a separated bar chart of calorie counts of fast food items for two restaurant chains, Starbucks and McDonald's. Unlike the grouped column chart in Figure 5.10, here the bars are separated from each other, because we do not need to make comparisons between sub-groups.

The only difference between making a grouped versus a separated chart is how you structure your data. To make Google Sheets separate columns or bars, you

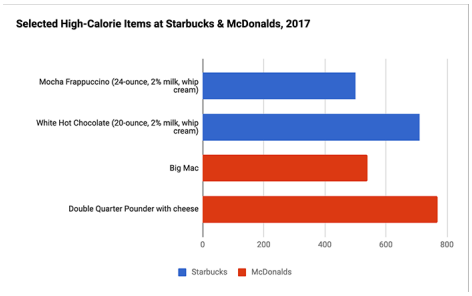


Figure 5.19: Separated bar chart with data from Starbucks and McDonalds. Explore the full-screen interactive version.

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

Figure 5.20: Create a separated column or bar chart by leaving some cells blank.

need to leave some cells blank, as shown in Figure 5.20. The rest of the steps remain the same as above.

To create your own separated column or bar chart using the fast-food example, right-click to open this link in a new tab: Google Sheet Separated Bar Chart template.

Stacked Column and Bar Charts

Stacked column and bar charts can be used to compare subcategories. They can also be used to represents parts of a whole instead of pie charts. For example, the stacked column chart in Figure 5.21 compares the percentage of overweight residents across nations, where colors allow for easy comparisons of weight-group subcategories across nations.

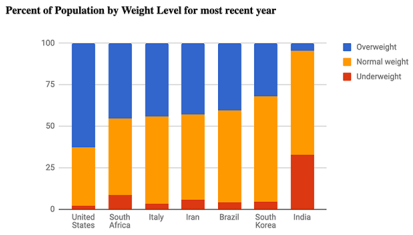


Figure 5.21: Stacked column chart with data from WHO and CDC. Explore the full-screen interactive version.

To create a stacked column or bar chart, structure your data so that each column will become a new series with its own color, as shown in Figure 5.22. Then in the Chart Editor window, choose Chart Type > Stacked column chart (or Stacked bar chart). The rest of the steps are similar to the ones above.

	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 5.22: Create a stacked column or bar chart by structuring your data as shown.

To create your own stacked column or bar chart using the international weight level example, right-click to open this link in a new tab: Google Sheets Stacked Column Chart template.

Histograms

Histogram is a type of bar chart that represents distribution of items, whether numerical or categorical. To bulid a histogram, you need to assign each data point to one of the non-overlapping *buckets* (or *bins*).

Let's say you want to know what time of day are you more likely to get an email. One approach would be to download metadata about all emails you received in 2020, and assign them to a bucket between 0 and 23 according to the email hour. Hours will become your bins, and email counts will be your frequency data. Then your final dataset would look something like this:

```
Hour  Emails
0       12
1       11
2       7
...
21    24
22    34
23    22
```

You can now make a histogram. The good news is, Google Sheets considers histograms to be regular column charts, so you should be able to use a previous tutorial to make one.

Select two columns with the data you want to visualize, and go to Insert > Chart. In the Chart editor window, in the Setup tab, select Chart type > Column chart. See the result in Figure 5.23

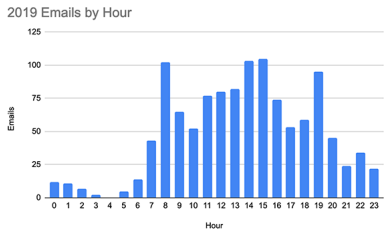


Figure 5.23: Histogram chart with fictitious source data. Explore the full-screen interactive version.

To create your own histogram chart using our fictional email dataset, right-click to open this link in a new tab: Histogram Chart template.

Bins in a histogram should span (in other words, "cover") the entire range of values of your dataset. This way you don't leave out any data. We recommend you use bins of the same size (like 24 1-hour bins, or four 6-hour bins) to ensure readers can compare across bars. For example, if you want to create a less detailed histogram, you can combine hours into larger bins, such as *Morning*, *Afternoon*, *Evening*, and *Night* to cover the hours of 6–11, 12–17, 18–23, and 0–5, respectively. Then your dataset will look like:

```
Time of Day,Emails
Morning,353
Afternoon,497
Evening,279
Night,37
```

Pie, Line, and Area Charts with Google Sheets

Pie Chart

As we mentioned in the Chart Aesthetics section, you need to be careful when using pie charts. First, remember to not have too many slices (ideally you should limit slices to 5). They should be arranged from largest to smallest and start at 12 o'clock. To separate slices, you can use different slice colors, or lines.

Make sure your data adds up to 100%. For example, if you want to show a pie chart with the number of fruit your store had sold in a day——21 apples, 5 oranges, and 32 bananas——the sum of all fruit, 58, is your 100%. Then a reader can figure out that of all fruits sold, approximately 55% were bananas. This example is illustrated in Figure 5.24. If you decided to include *some*, but *not all* other items that your store has sold (for example, you include pizzas but exclude ice cream), your pie chart would not make sense.

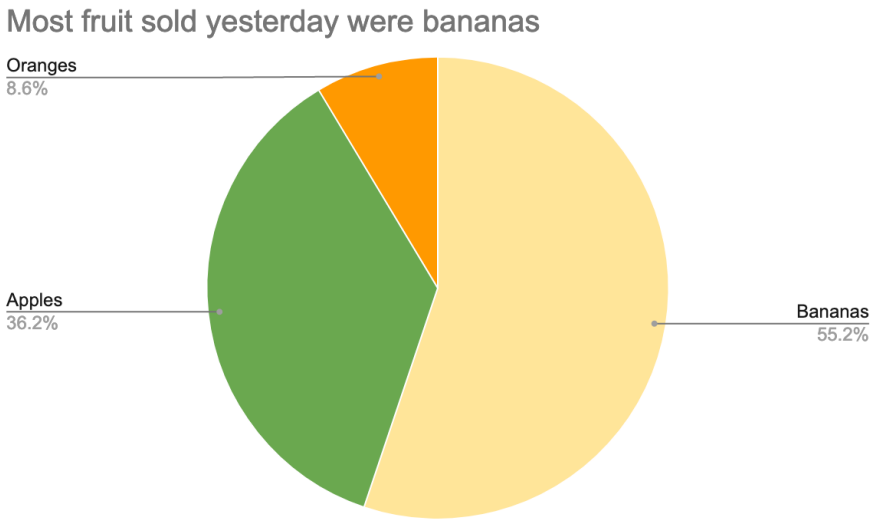


Figure 5.24: Pie chart with fictitious source data. Explore the full-screen interactive version.

To make a pie chart with Google Sheets, arrange your data in two columns, *Label* and *Value*. Values can be expressed as either percentages or counts. For example,

```
| Apple  |  21 |
| Orange |   5 |
| Banana |  32 |
```

Select all cells and go to *Insert* > *Chart*. Google Sheets is good at guessing chart types, so it is possible the chart you will see right away will be a pie. If not, in Chart editor in tab Setup, select *Pie chart* from the Chart type dropdown list.

Notice that slices are ordered the same way they appear in the spreadsheet. We highly recommend you sort values from largest to smallest: right-click the header of your values column, and click `Sort sheet Z-A`. You will see that the chart updates automatically.

Right-click on the chart, and choose *Chart & axis titles* > *Chart title* to add a meaningful title. In *Customize* tab of the Chart editor, you can also change colors and add borders to slices.

Line Chart

The most common use of line charts is to represent values at different points in time, in other words to show change over time. The line chart in Figure 5.25 shows per-capita meat availability in the US for the past 110 years. You can see that the level of chicken (shown in orange) rises steadily and surpasses beef (red) and pork (blue).

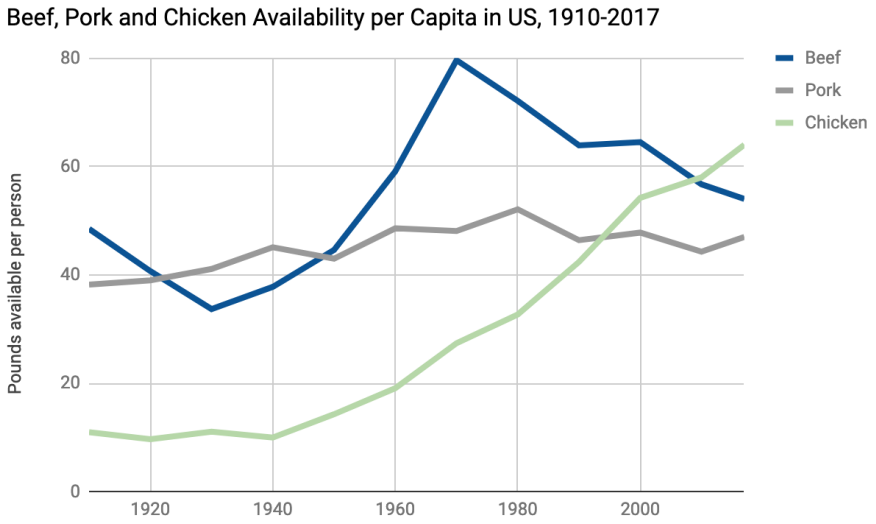


Figure 5.25: Line chart showing meat availability per capita in the US, according to the US Department of Agriculture. Explore the full-screen interactive version.

The simplest way to organize your data is to use the first column as x-axis labels, and each additional column as a new series (which will become its own line). For example, the meat data from the line chart is structured as shown in Figure 5.26.

	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 5.26: Data for the line chart shown in Figure 5.25.

The data is available in the Google Sheet Line chart template. If you wish to use it, just make a copy to your own Google Drive from the File menu.

Select the data, and choose *Insert* > *Chart*. It is possible Google Sheets will create a line chart right away. If not, in Chart editor in tab Setup, select *Line chart* from the Chart type dropdown list.

Stacked Area Chart

The line chart in the previous example made it possible to see how individual meat availability changed over time. It was hard, however, to estimate if the overall meat availability went up or down. (That is, of course, if we assume that beef, pork, and chicken are the only meats we eat).

We can see how availability of individual meat types, *and* the total meat availability over time using a stacked area chart, like shown in Figure 5.27. Here, we can still see that chicken has been on the rise since 1970s. We can also see that the total availability was on the rise between 1910 and 1970 with a small dip around 1930s, and it didn't change much between 1970 and 2017.

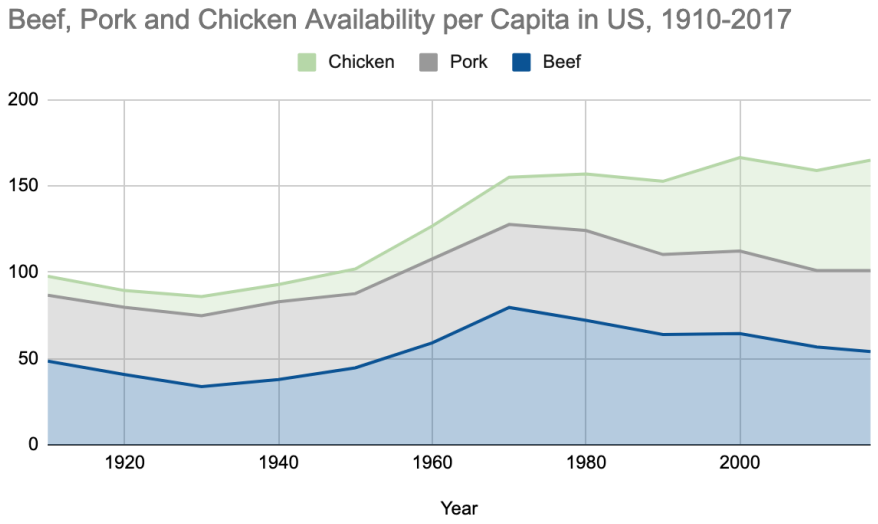


Figure 5.27: In addition to individual meat availability, stacked area charts show the overall availability. See data by US Department of Agriculture. Explore the full-screen interactive version.

The data for the stacked area chart is available from the Google Sheet Stacked area chart template, which you copy to your own Drive.

Set up the data exactly as you would with a line chart (first column is labels for the x-axis, second and following columns are series, or lines). Select it, and choose *Insert* > *Chart*. In the Chart editor, in tab Setup, select *Stacked area chart* from the Chart type dropdown list.

XY Scatter and Bubble Charts with Google Sheets

Consider using XY scatter charts, also known as scatterplots, to display data coordinates to show the relationship between two variables. The first example below compares the relationships between life expectancy (shown on the X axis) and fertility (shown on the Y axis), which each nation is represented as a dot (an X-Y coordinate). Bubble charts are basically scatter charts on steroids, meaning that they can display the relationship of up to five variables. Further below you'll build a bubble chart based on the same XY life expectancy-fertility dataset, with added variables for population (displayed as circle size) and region of the world (displayed as circle color). Fancier bubble charts animate the circles to represent one more variable: change over time.

XY Scatter chart

The scatter chart in Figure 5.28 uses World Bank data to reveal a downward slope: nations with lower fertility (births per woman) tend to have higher life expectancy. You can also phrase is the other way, nations with higher life expectancy at birth have lower fertility. Remember that correlation does not mean causation, so you cannot use this chart to argue that fewer births result in longer lives, or that longer-living females give birth to fewer children.

The data used in Figure 5.28 is available from our Google Sheets Scatter chart template. You can copy it to your own Google Drive so that you're able to edit it (go to *File > Make a copy*).

Figure 5.29 shows the first few rows of the dataset. Notice that the data is structured in three columns. The first column, *Life Expectancy*, is plotted on the x-axis (horizontal). The second column, *Fertility*, is plotted on the y-axis (vertical). The third column contains *Country* labels.

To build a scatter chart, select the **two** columns that contain your numeric data, and go to *Insert > Chart*. Google Sheets will likely to guess the chart type and you will see a scatterplot, but if not, you can always manually pick Scatter chart from the *Chart type* dropdown. Make sure your x-axis is set to Life Expectancy, and your Series shows Fertility. Note that both Life Expectancy and Fertility have `123` icon, meaning they are numeric.

You will see a lot of scatter charts out there that do not label data points, and that's okay. Some scatter plots are designed to show whether or not there is a correlation, and knowing which points are which is not important. But sometimes labels are important for your storytelling.

In Chart editor, click on the 3-dot menu for your Series dataset (Fertility), and then *Add labels* (see Figure 5.30). The labels added by default will be the x-values of points. To make Google Sheets read labels from the third column

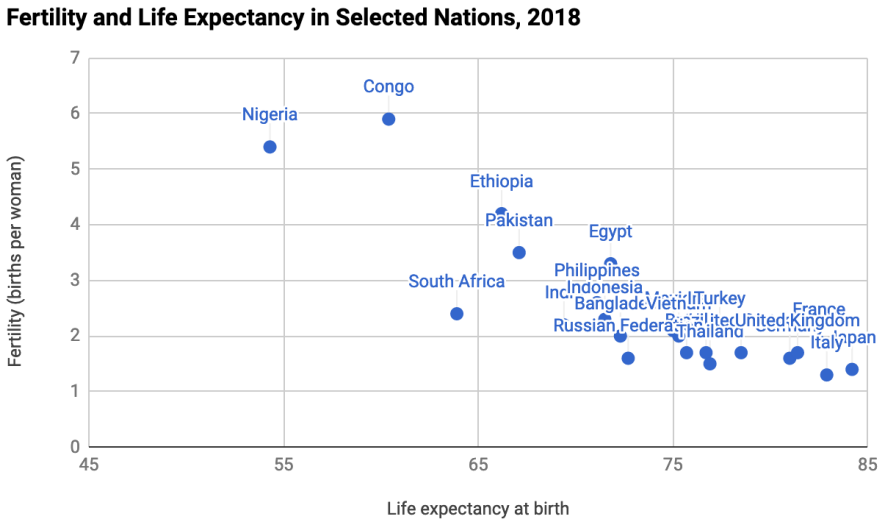


Figure 5.28: This scatter chart shows that nations with lower fertility tend to have higher life expectancy. See data by World Bank. Explore the full-screen interactive version.

	A	B	C
1	Life Expectancy	Fertility	Country
2	76.7	1.7	China
3	69.4	2.2	India
4	78.5	1.7	United States
5	71.5	2.3	Indonesia
6	75.7	1.7	Brazil
7	67.1	3.5	Pakistan
8	54.3	5.4	Nigeria
9	72.3	2	Bangladesh
10	72.7	1.6	Russian Federation

Figure 5.29: Data for a scatterplot is usually represented in 3 columns: x-values, y-values, and labels.

(*Country*), click the name of your label dataset (Life Expectancy), then *Select a data range* button in the upper-right corner of the dropdown, and choose cells in the relevant columns. Make sure to include the header (first row) if all other data ranges include it.

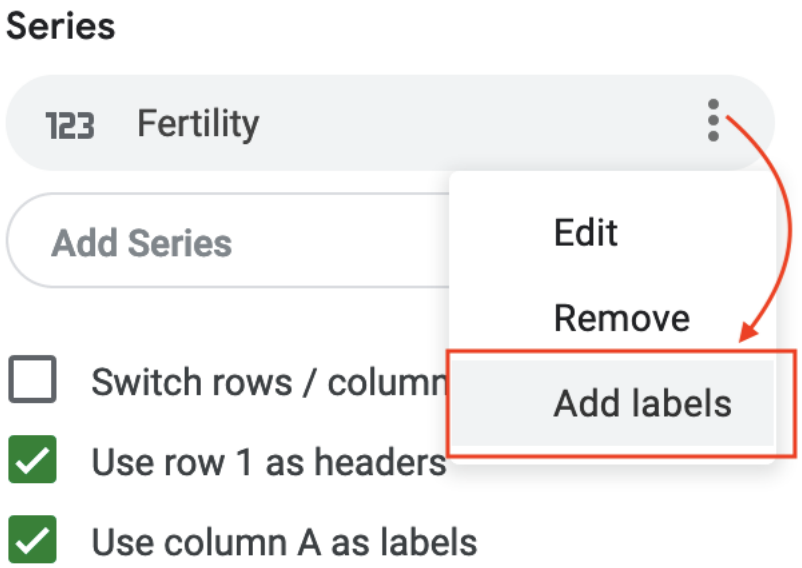


Figure 5.30: In the chart's Setup window, choose *Add labels* to the Series.

Tip: You may notice that some data points are too close to edges, and their labels are cut off. To fix this, go to Customize tab of the Chart editor. There, you can set minimum and maximum values for both horizontal and vertical axes. Unlike in bar charts, axes in scatter plots do not have to start at zero. You can set your minimum and maximum values to be a few units below and above the extreme points of your data range.

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 "burger menu" on many mobile devices. TODO: Add picture of each icon?

Bubble chart with 3 columns

In this tutorial, we will show you a little trick that you can use if you want a scatter chart with both data values displayed in a tooltip. We will use the same World Bank dataset as we did for the scatter plot.

The bubble chart (more about the *proper* use of bubble charts in the next section) in Figure 5.31 shows the same data as our scatterplot on life expectancy vs fertility.

In the interactive version of the chart, hover your cursor over each bubble (dot) to reveal a tooltip with the country name and the two data points.

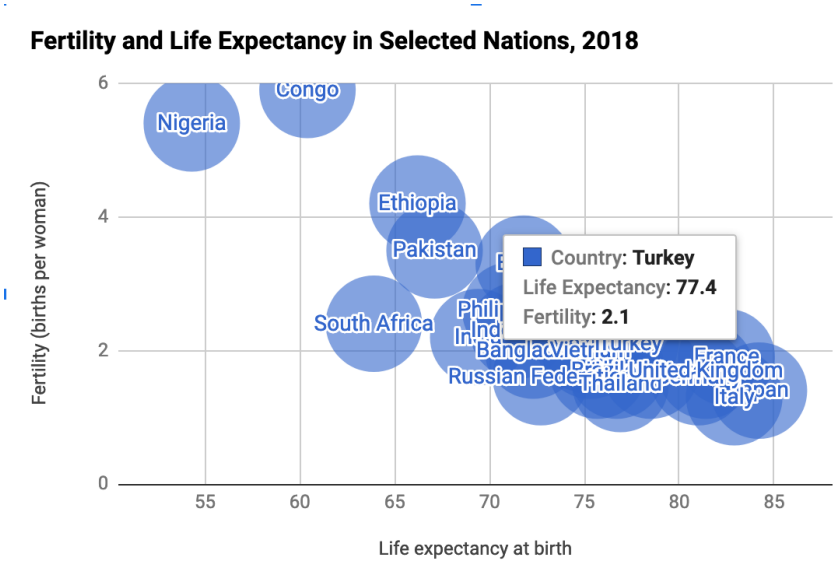


Figure 5.31: This bubble chart is essentially a scatter chart, because no other dimensions (colors, sizes) are used. See data by World Bank. Explore the full-screen interactive version.

The data for this example is available in Google Sheets Bubble chart with 3 columns template.

Notice that we moved the labels column (*Country*) to be the first one in the dataset, but the order shouldn't matter in this case. So our first column is the label for each bubble, the second column is the data to be plotted on horizontal x-axis, and the third column (fertility) will be place on the y-axis.

Select all three columns, and go to *Insert* > *Chart*. Google Sheets will likely create a stacked column chart by default, so choose *Bubble* from the Chart type dropdown window.

If you want to remove labels from the bubbles, remove the **ID** series (click on the kebab menu > Remove).

Unfortunately, there is no easy way to reduce all bubbles to a uniformly smaller size. In the following section, we will introduce you to the proper way of using bubble charts.

Bubble chart with 5 columns

Bubble charts are a good alternative to scatter charts if you need to include one or two extra series in addition to your x- and y-coordinates. One of those can be expressed through bubble size (bigger bubbles represent larger values). Another one can make use of color (best for categorical data).

The bubble chart in Figure 5.32 shows fertility and life expectancy for a subset of the nations, with population (shown by bubble size) and region (shown by bubble color). Float your cursor over bubbles to view data details in the interactive version of the chart.

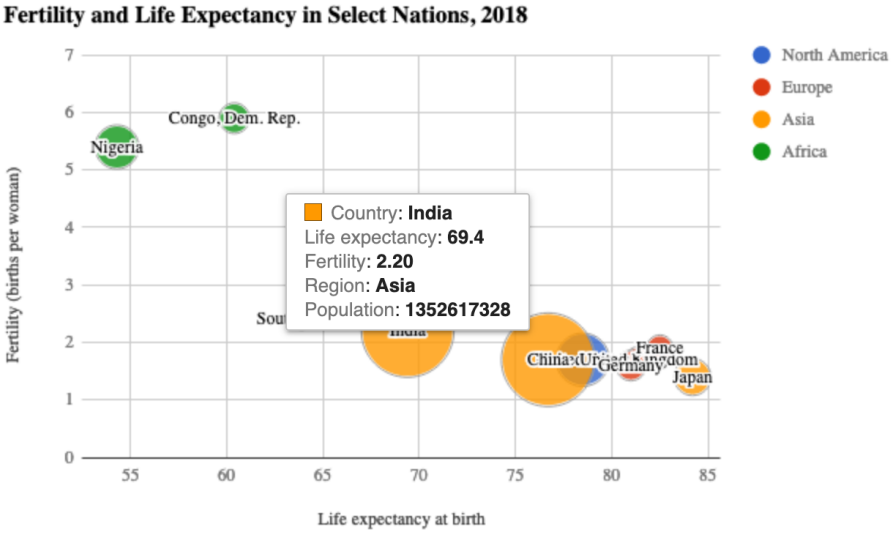


Figure 5.32: This bubble chart shows fertility and life expectancy for several countries, including their population (shown by bubble size) and region (shown by bubble color). See data by World Bank. Explore the full-screen interactive version.

The five-column dataset is available in this Google Sheets Bubble chart with 5 columns template. The columns are arranged in the following order: country label, x-axis value, y-axis value, color, and bubble size.

Select all data and go to *Insert* > *Chart*, and choose Bubble as the Chart type. Make sure your **ID**, **X-axis**, **Y-axis**, **Series**, and **Size** fields contains the series you want to display, and make sure to have *Use row 1 as headers* option checked.

To change labels color, go to Customize tab of the Chart editor, and set Text color under the Bubble menu. Make it gray or black, so that it won't interfere with the bubble colors themselves.

	A	B	C	D	E
1	Country	Life expectancy	Fertility	Region	Population
2	United States	78.5	1.70	North America	326687501
3	United Kingdom	81.4	1.70	Europe	66460344
4	China	76.7	1.70	Asia	1392730000
5	India	69.4	2.20	Asia	1352617328
6	Japan	84.2	1.40	Asia	126529100
7	Germany	81.0	1.60	Europe	82905782
8	France	82.5	1.90	Europe	66977107
9	Congo, Dem. R	60.4	5.90	Africa	84068091
10	Nigeria	54.3	5.40	Africa	195874740
11	South Africa	63.9	2.40	Africa	57779622

Figure 5.33: Bubble chart data. Bubble size represents population, color – region.

Tip: If some of your bubbles are too close to the borders, set Min and Max values for the axis manually under Horizontal axis and Vertical axis menus.

Create Charts with Tableau Public

Tableau is powerful data visualization software used by many professionals and organizations to analyze and present data. Tableau can combine multiple datasets to show in a single chart (or a map), and allows to create dashboards with multiple visualizations. Individual visualizations and dashboards can be published and embedded on your website through an iframe.

This book focuses on the free Tableau Public tool, available to download for Mac or Windows. This free version of Tableau Public is very similar to the pricier versions that the company sells, but one constraint is that the data visualizations you create will be public, as the name suggests, so do not use it for any sensitive or confidential data that should not be shared with others.

You might be overwhelmed by the amount of options and features Tableau provides through its interface. We will show you the very basics enough to get started, and if you want to dive further, there are many great books on Tableau available.

In this book, we will show you how to add datasets to Tableau Public, and how to create a scatterplot and a filtered line chart.

Create XY Scatter Chart with Tableau Public

Just to remind you, scatter charts plot two variables against each other, on x- and y-axis, revealing possible correlations. With Tableau Public, you can create an interactive scatter chart, letting users hover over points to view specific details.

Figure 5.34 illustrates a strong relationship between Connecticut school district income and test scores.

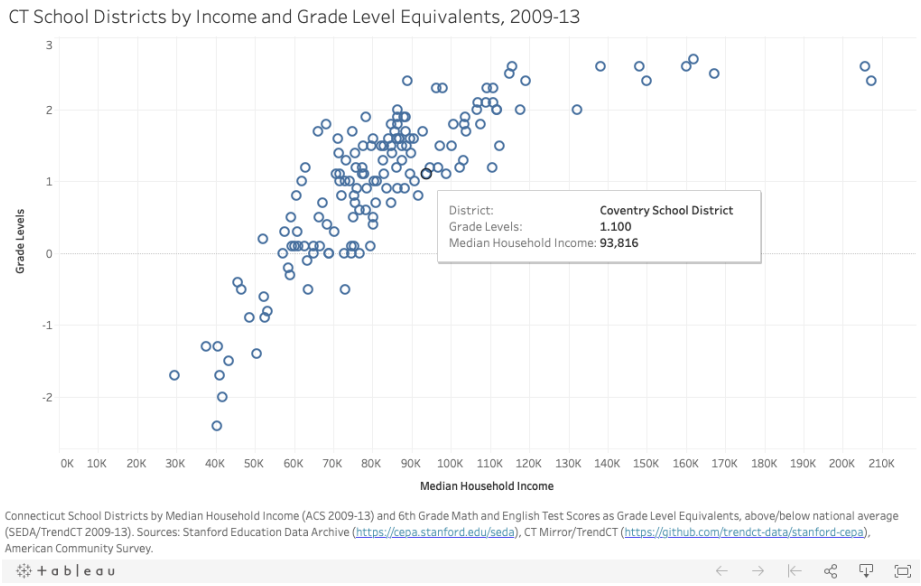


Figure 5.34: Household income vs test scores in Connecticut school districts. Made with Tableau Public.

Install Tableau and Get Data

You can download Tableau Public for Windows or Mac from Tableau's official website. You will need to provide your email address.

If you wish to use the dataset from the scatter plot in Figure 5.34, you can download the sample Excel file. This data file consists of three columns: district, median household income, and grade levels (above/below national average for 6th grade Math and English test scores). The Notes tab explains how this data is based on the work of Sean Reardon et al. at the Stanford Education Data Archive, Motoko Rich et al. at The New York Times, Andrew Ba Tran at TrendCT, and the American Community Survey 2009-13 via Social Explorer.

Connect Data and Create a Scatterplot

Tableau Public's welcome page includes three sections: Connect, Open, and Discover.

1. Under Connect, choose Microsoft Excel if you decided to use the sample dataset or your own Excel file. To load a CSV file, choose *Text file*. If your data is in Google Sheets, click *More...* and choose Google Sheets. Once 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`.

2. Drag `data` sheet into *Drag tables here* area, like is shown in Figure 5.35. 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.

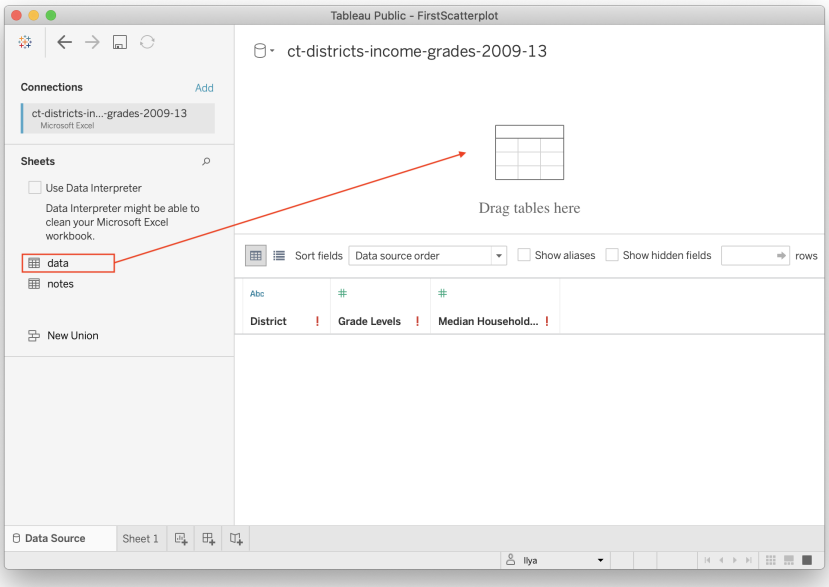


Figure 5.35: Drag `data` sheet into *Drag tables here* area.

3. Go to *Sheet 1* tab (in the lower-left corner of the window) to view your worksheet. Although it may feel overwhelming at first, the key is learning where to drag items from the Data pane (left) into the main worksheet. **Dimensions** are any information that is qualitative or categorical, and are

shown in blue with *Abc* tables. **Measures** are quantitative information about the dimensions, shown in green with # icons.

4. Drag the *Grade Levels* measure into the **Rows** field above the charting area, which for now is just empty space. You can consult Figure 5.36 for this and two following steps. Tableau will apply a summation function to it, and you will see the `SUM(Grade Levels)` appearing in the Rows row, and a blue bar in the charting area. It makes little sense so far, so let's plot another measure (variable).

5. Drag *Median Household Income* to the **Columns** field (just above the Rows field). Tableau will once again apply the summation function, so you will see `SUM(Median Household Income)` in the Columns. The bar chart will transform into a scatter chart with just one data point in the upper-right corner. That is because the data for both is aggregated (remember the `SUM` function).

6. We want to tell Tableau to disaggregate the household and grade levels variables. To do so, drag *District* dimention into the lower portion of the Marks area. You will now see a real scatter chart in the charting area. If you hover over points, you will see all three values associated with it.

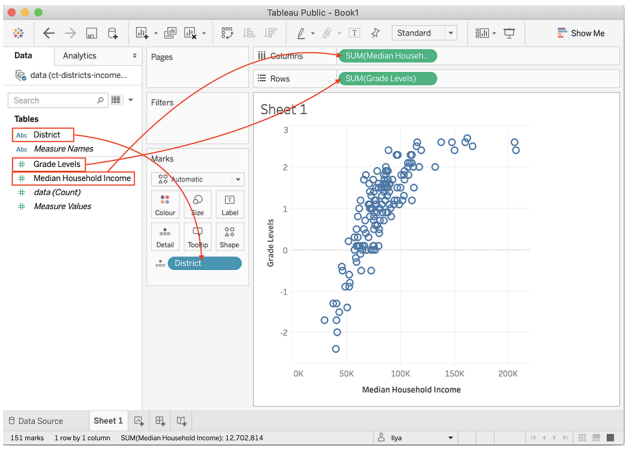


Figure 5.36: Drag measures and dimensions to the right places in Tableau.

Add Title and Caption, and Publish

Give your scatter chart a meaningful title by double-clicking on default *Sheet 1* title above the charting area.

You will normally need to provide additional information about the chart, such as source of the data, who built the visualization and when, and other important

things. You can do so inside a Caption, a text block that accompanies your Tableau visualization. In the menu, go to *Worksheet > Show Caption*. Double-click the Caption block that appeared, and edit the text.

As a result, your final worksheet will look like shown in Figure 5.37.

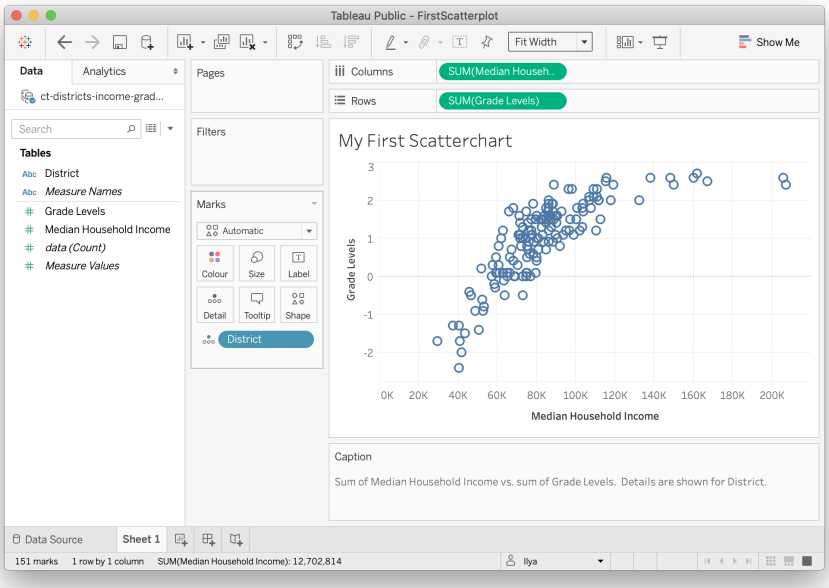


Figure 5.37: This scatter chart is ready to be published.

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

To publish the chart to the web,

1. 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.

2. Once signed in, a window to set the workbook title will appear. Change the deafult *Book1* title to something meaningful, as this name will appear in the URL for your published work. Click Save.

3. Once the dashboard is saved, Tableau Public will open up a window in your default browser with the visualization. In the green ribbon above the chart, click *Edit Details* to edit chart's title or description. Under *Toolbar Settings*, see checkbox to *Allow others to download or explore and copy this workbook and its data* (Figure 5.38), and enable/disable it as

you think is appropriate. As advocates for open and accessible data, we recommend leaving the box checked.

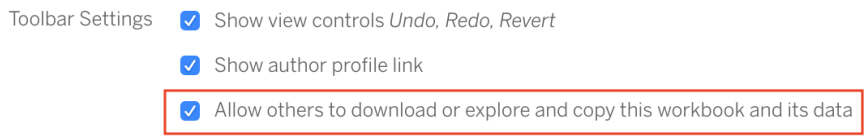


Figure 5.38: This scatter chart is ready to be published.

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

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

To learn more, see Tableau Public resources page.

Create Filtered Line Chart with Tableau Public

One of the advantages of interactive visualizations over static (including printed) is the ability to store a lot more data, and show it only when required. In other words, an interactive visualization can be made into a data-exploration tool that won't overwhelm the viewer at first sight, but will allow the viewer to "dig" and find specific data points and patterns.

In this tutorial, we will build an interactive filtered line chart with Tableau Public like is shown in Figure 5.39. The filter will be a collection of checkboxes that allow to add/remove lines from the chart. Viewers can hover over each line to identify the school name and data attached to it.

We will use % Population with Internet Access by the World Bank. You can download the dataset here.

We assume that you have Tableau installed. If not, see the previous tutorial, Create XY Scatter Chart with Tableau Public.

Connect Text File and Build a Line Chart

Open Tableau Public, and under Connect menu, choose *Text file*. Tableau may or may not have imported the table automatically. If you see the preview of the table with three columns: *Country Name*, *Year*, and *Percent Internet Users*, you can proceed to Sheet 1.

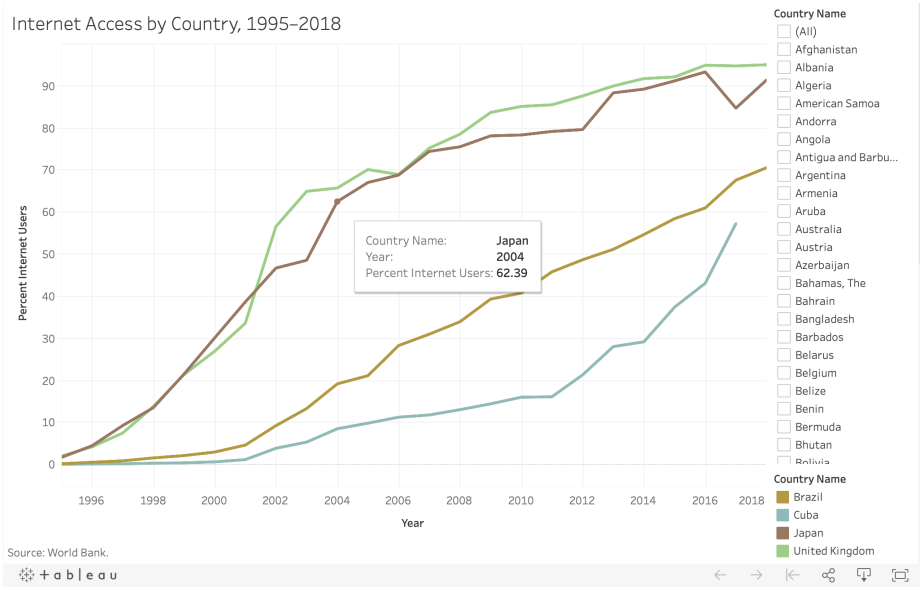


Figure 5.39: Internet Access by Country, 1995–2018.

If not, drag and drop the file (under Files section in the left) to the *Drag tables here* area. Once you see the preview, go to Sheet 1.

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* that Tableau guessed from the country names).

To build a line chart,

1. Drag Year variable to **Columns**.
2. Drag Percent Internet Users variable to **Rows**. The variable 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 box of the Marks card. Tableau will warn you that the recommended number of colors should not exceed 20. Since we will be adding filtering, we don't care about it much. So go ahead and press *Add all members* button.
4. Now you should see an absolute 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. Right-click on *Country Name* pill in Filters card, and check Show Filter (see Figure @ref(fig:tableau-filtered-show-filter.png))

6. You will see a list of options with all checkboxes on have appeared to the right of the visualization. Click *(All)* to add/remove all options, and add a few of your favorite countries to see how the interactive filtering works.

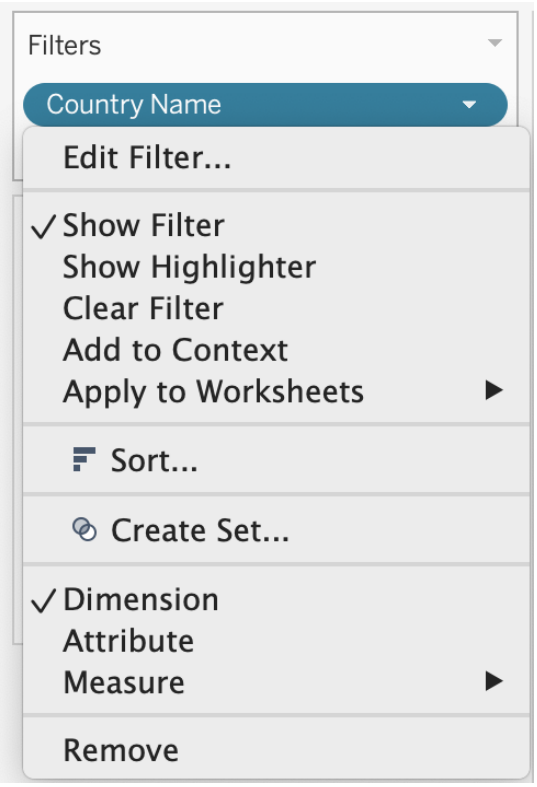


Figure 5.40: Once you dragged Country Name to the Filters card, make sure the Filter is displayed.

Add Title and Caption, and Publish

Replace *Sheet 1* title (above the chart) with "Internet Access by Country, 1995–2018" by double-clicking on it. 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.

Change *Standard* to *Fit Width* in the dropdown above the Columns field.

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.

Once your filtered line chart looks like the one shown in Figure 5.41, you are ready to publish.

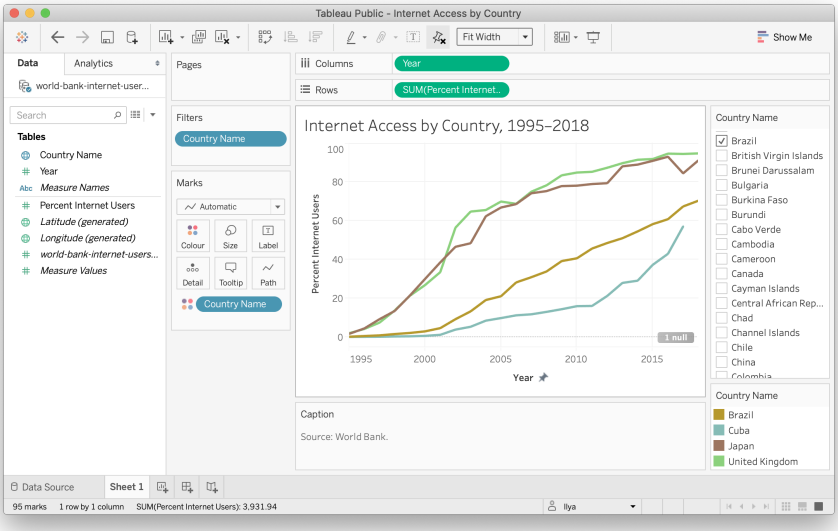


Figure 5.41: This workbook is ready to be published.

To publish the filtered line chart to the web, go to *File > Save to Tableau Public As...*. You may be prompted with the window to log in to your account (or create one if you don't have it yet). The next steps are fairly self-explanatory, and you can consult the previous tutorial for more information on publishing.

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.

To learn more, see Tableau Public resources page.

Summary

Congratulations on creating interactive charts that pull readers deeper into your story, and encourage them to explore the underlying data! As you continue to create more, always match the chart type to your data format and the story you wish to emphasize. Also, design your charts based on the principles and aesthetic guidelines in 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.

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 7.

To learn about more powerful charting tools, see Chart.js code templates in chapter 9, which give you ever more control over how your design and display your data, but also require learning how to edit and host code templates with GitHub in chapter 8.

Chapter 6

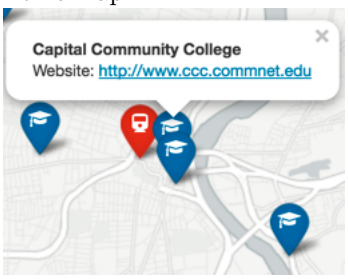
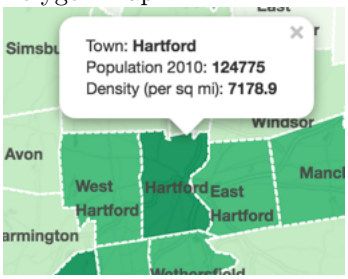
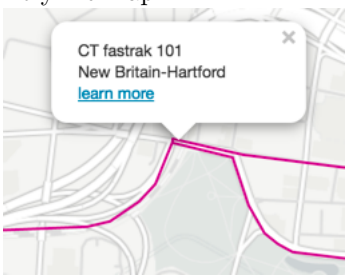
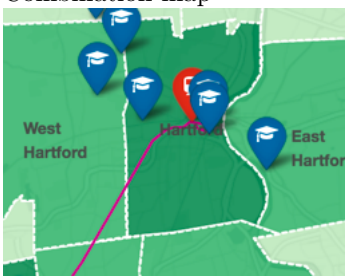
Map Your Data

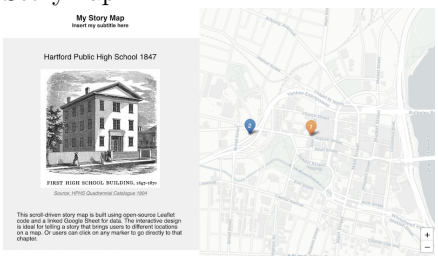
Maps entice readers to explore your data story and develop a stronger sense of place. But good maps require careful thought about how to clearly communicate spatial concepts with your audience. This book features free tools to create interactive maps that you can embed in your website. In this chapter, you will learn how to:

- Practice key principles of map design.
- Choose a map type that matches your data story and format, with tutorial links in the table below. Beginners may start with easy-to-learn tools such as Google My Maps, then move up to more powerful tools, such as Leaflet, which require you to Edit and Host Code with GitHub or other web servers.

See also related chapters in this book:

- Draw and write your data story to capture your ideas on paper
- Improve spreadsheet skills, Find and know your data, and Clean your data
- Transform your map data
- Embed your interactive chart on your website
- Detect bias in data stories, including How to lie with maps
- Tell your data story, including its most meaningful insights and limitations

Basic map types	Best use and tutorial chapters
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
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
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

Basic map types	Best use and tutorial chapters
Storymap 	Best for guided point-by-point journey through a historical narrative, with optional photos, audio, or video on an interactive map. Easy tool: Knight Lab's StoryMap, ESRI Story Maps Power tool: Leaflet Storymaps with Google Sheets

TODO:

- heat map
- tab-view map for historical change
- synchronized side-by-side map

Map Design Principles

Ask Before You Map: Before you leap into a mapping project, consider these questions:

Does your data contain geographic information? Common examples:

- Specific locations or addresses (examples: *Trinity College*, or *300 Summit St, Hartford, CT*)
- Latitude and longitude coordinates (example: *41.756, -72.675*)
- Regions that are legally recognized (such as nations, states, counties, census tracts) or that correspond to a boundary map in your possession (such as designated neighborhoods or health districts)

While there are many more types of geographic information, these examples above are the most common. If your data lacks geographic information, or if you do not possess the corresponding boundary information, it may not be possible to map it.

Does location really matter to your data story?

Sometimes a well-designed chart, rather than a map, may be the best way to visualize your data story. Consider these alternatives:

- to show change over time across different locations, consider a line chart

- to show the relationship between two or more datasets across different locations, consider an XY scatter chart or bubble chart

If a map is the best way to tell your data story, then choose an appropriate type. See table of basic map types in this book.

Map Design Principles

1. Understand basic map vocabulary: title, legend, baselayer, marker, popup, tooltip, zoom level, polygon, polyline, source.

2. Add source credits and bylines—with links to view data tables and details—to build credibility and accountability.

3. Choose colors wisely.

    - Use color to logically organize your data. Avoid random colors (Wong pp. 40, 44).
    - Avoid bad combinations from opposite sides of color wheel, such as red/green or yellow/blue (Wong pp. 40, 44).
    - Use contrast (such as color vs gray) to call attention to your data story (Knaflic pp. 87-88)

4. Choose basemaps wisely. Basemaps themselves may contain a lot of information, such as terrain, roads, parks, town names, buildings, etc. They may also use colors that can be distracting to the viewer. Think about the minimum number of elements required in the basemap to tell your story.

Figure 6.1: The view of San Francisco with different basemaps

Design polygon maps with ColorBrewer

One of the most useful tools for creating meaningful polygon (or choropleth) maps is ColorBrewer http://colorbrewer2.org created by Cynthia Brewer, Mark Harrower and the Pennsylvania State University.

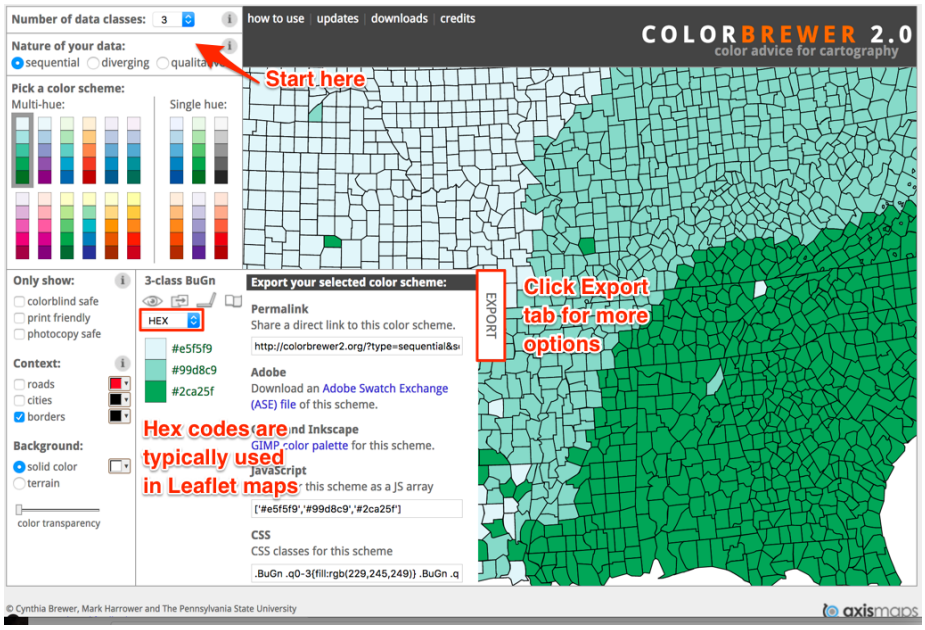


Figure 6.2: Screenshot: ColorBrewer web interface

1) Think about the **number of data classes** (or "dividers" or "buckets"). More does not necessarily mean better. Try different numbers and color schemes, and decide if you (and your audience) can easily distinguish between them.

   - A smaller number sorts your data into fewer buckets, and shows a more **coarse map**, but differences in colored ranges become **more visible**.
   - A larger number sorts your data into more buckets, and shows a more **granular map**, but differences in colored ranges become **less visible**.

2) Think about the **nature of data** you are going to display.

- Sequential: best to show steps from lower values (light color) to higher values (dark color)

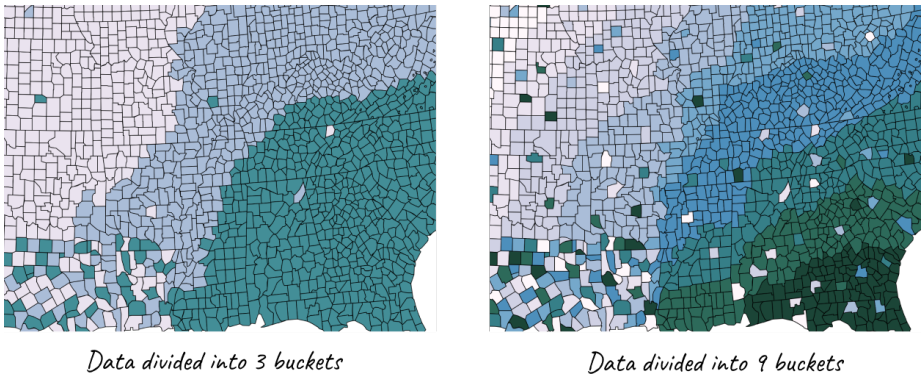


Figure 6.3: Screenshots: ColorBrewer web interface

  - Example: a scale that increases from 1 to 100
- Diverging: best to show extremes (dark colors) around a neutral middle (light color)
  - Example: a scale that highlights extremes from -100 to 0 to 100
- Qualitative: best to show different categories, represented by their own color
  - Example: a map legend of the dominant crop in each area: apples, oranges, bananas

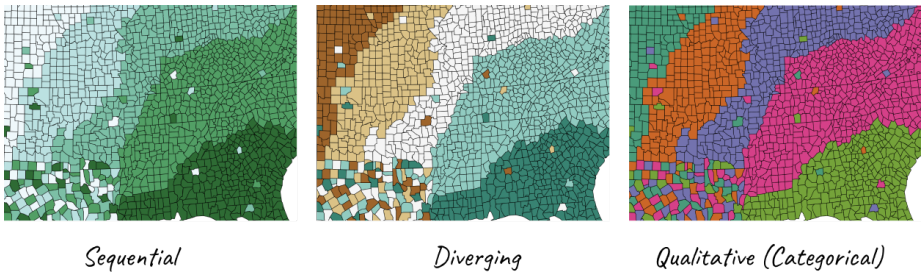


Figure 6.4: Screenshots: ColorBrewer web interface

3) Pick a **color scheme**, with options for colorblind-safe and print-friendly.
   - Think about the ideal format for your audiences. Are readers more likely to view your visualization on a computer screen, or in print, or both?
4) Click the Export tab to view all options. Some Leaflet map templates in this book use specific color names (such as “red” or “darkgreen”) and

some use hexadecimal codes, abbreviated as "hex codes" (such as #ff0000 or #336600). To learn more, use a Color Picker tool, such as https://www.w3schools.com/colors/colors_picker.asp

Beware that polygon map design choices about data classes and colors reflect the biases of the author and the software. Read the Detect Bias in Data Stories chapter in this book, especially How to Lie with Maps

Learn more

- Axis Maps, "The Basics of Data Classification," 2010, http://axismaps.github.io/thematic-cartography/articles/classification.html
- Lisa Charlotte Rost, "Your Friendly Guide to Colors in Data Visualisation," Lisa Charlotte Rost, April 22, 2016, https://lisacharlotterost.github.io/2016/04/22/Colors-for-DataVis/.
- Josh Stevens, "Bivariate Choropleth Maps: A How-To Guide," February 18, 2015, http://www.joshuastevens.net/cartography/make-a-bivariate-choropleth-map/.

Point Map with Google My Maps

TODO: add text, check current documentation and features at https://www.google.com/maps/about/mymaps/

Try It

Explore the interactive point map below, or view the full-screen version, created with Google My Maps https://www.google.com/maps/d/.

Tool Review

- Pros
  - Easy-to learn free mapping tool to import and style point, polyline, and polygon layers and basemap layers
  - Share and collaborate through the Google Drive platform
  - Geocoding error warning
- Cons
  - Limited options to customize map markers
  - Cannot easily create colored polygon maps from data values
  - Cannot extract geocoded data to migrate to another tool

Video with Step-by-Step Tutorial

Let's build a simple point map with sample data, using Google My Maps https://www.google.com/maps/d/. Requires signing up for a free Google Drive account.

1) Click this link and Save to download to your computer: sample-address-data in CSV format. CSV means comma-separated values, a generic spreadsheet format that most tools can easily open. For help with downloading, see this short video tutorial.

2) Open and sign in to Google My Maps https://www.google.com/maps/d/

3) Click the red + symbol to create a new map, which will be saved automatically to your Google Drive folder.

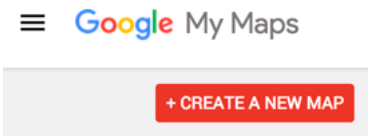


Figure 6.5: Image: Create a new map

4) In the map layers area, click the blue Import link. Drag-and-drop the CSV address data file into the web interface to import it.

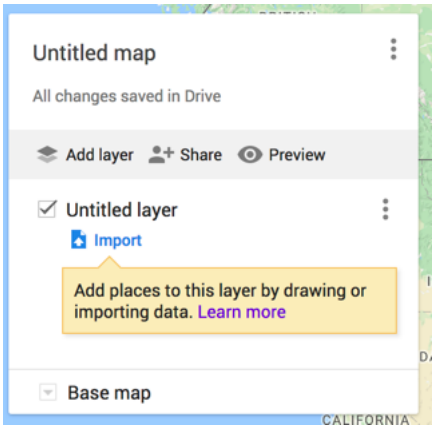


Figure 6.6: Image: Import a data layer

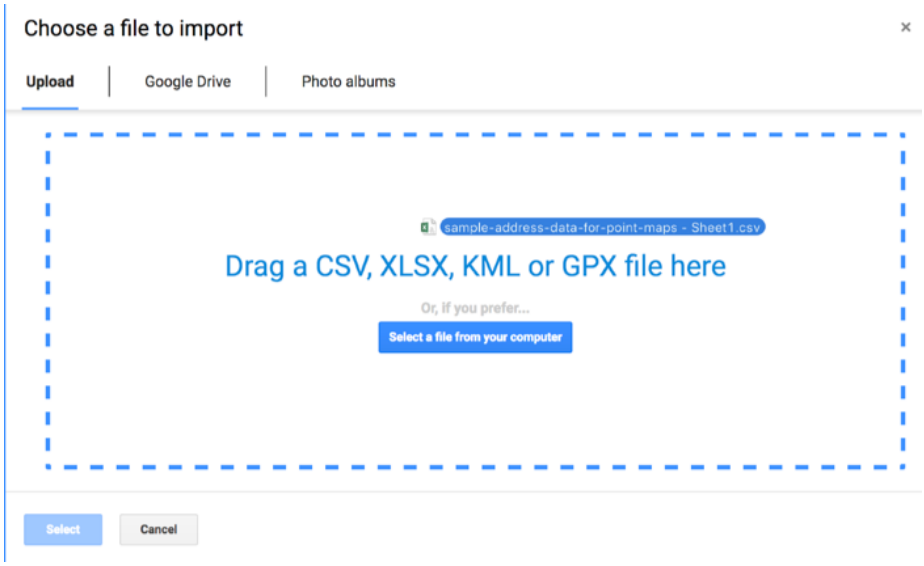


Figure 6.7: Image: Drag-and-drop data into My Maps

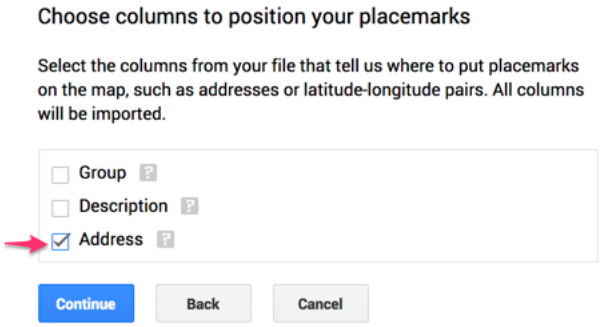


Figure 6.8: Image: Choose columns to position placemarks

5) Choose columns to position your placements. Select “Address” for this sample data, then Continue.

6) Choose a column to title your markers. Select “Description” for this sample data, then Finish.

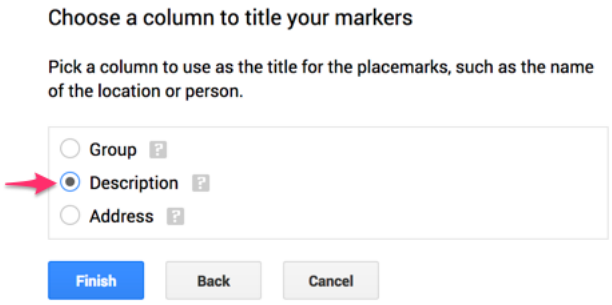


Figure 6.9: Image: Choose column to title markers

7) After My Maps uploads and geocodes your sample data, click Open Data Table to inspect the results.

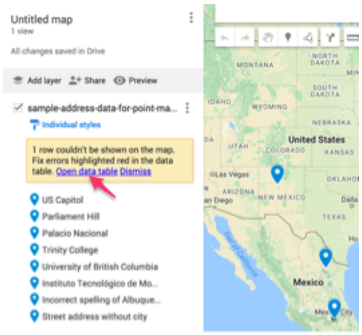


Figure 6.10: Image: Open Data Table to inspect geocoding errors

8) To style the map markers, click Individual Styles. In this sample data, you can select Group Places By > Style By > Group. This will color markers according to the three categories.

9) To publish your map on the web, click Share, add a map title, change from Private to Public on the Web, so that anyone can view your map. Click Save and Done.

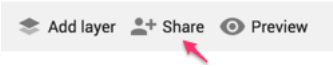


Figure 6.11: Image: Share link

10) To embed the map on your own website, click the three vertical dots next to the map title for more options, and select Embed On My Site. The tool will generate an iframe code for you to copy. For next steps, go to the Embed on Your Web chapters in this book.

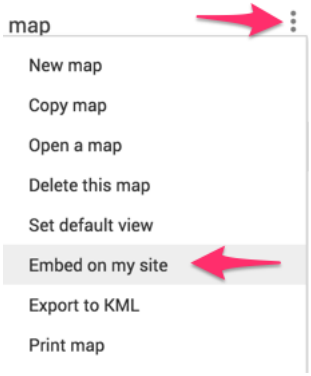


Figure 6.12: Image: Embed map on your site

Learn more

- Google My Maps Help Page https://support.google.com/mymaps/answer/3024396

Point Map with Carto Builder

TODO:

- Test this tool and decide if it still warrants inclusion in this book
- See note about old versus newer Cartobuilder – still relevant?
- if this tool stays in the book, check the iframe below to see if update is needed

Try It

Explore the interactive point map below, or view the full-screen version ,created with Carto Builder https://carto.com.

Tool Review

- Pros:
  - Free and powerful drag-and-drop map tool in the browser
  - Customize point markers and polygon colors by data values
  - Additional features include geographic analysis tools
- Cons:
  - Several steps required to create simple point or polygon map
  - New users may get lost when moving through multiple screens
  - Free account allows only 400 geocodes per month

Video with Step-By-Step Tutorial

Before you begin: This tutorial uses the newer Carto Builder, rather than older Carto Editor tool. Learn more at https://carto.com/learn/guides/intro/migrating-from-carto-editor-to-carto-builder. If you have an old Carto account that has not automatically updated to the new Builder tool, you may need to create a brand-new account to use this tutorial.

Let's build a simple point map with sample data, using Carto Builder https:/carto.com. Requires signing up for a free account.

1) Click this link and Save to download to your computer: sample-address-data in CSV format. CSV means comma-separated-values, a generic spreadsheet format that many tools can easily open.

2) Open Carto in your browser https://carto.com.

3) The Carto Dashboard displays two views: Maps and Datasets. Always begin with Datasets, then move to Maps. (Hint: If your dashboard looks very different than mine, then you might still be using the older Carto Editor, rather than the newer Carto Builder.)

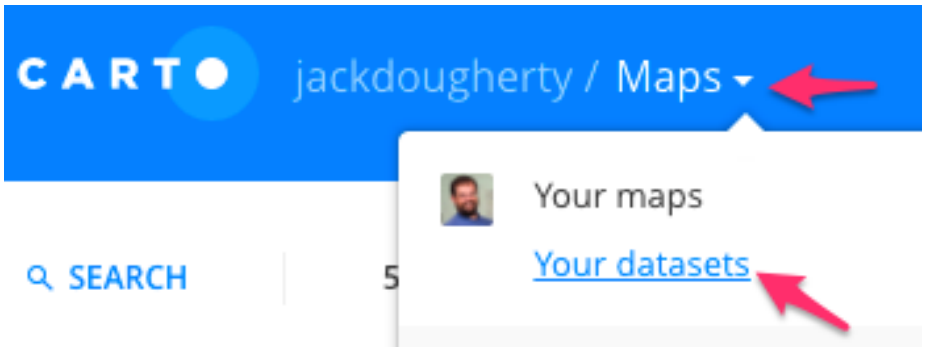


Figure 6.13: Image: Carto Builder dashboard: Begin with Datasets

4) First, connect your dataset, and soon we'll turn it into a map. Click blue button to add New Dataset.

5) Drag-and-drop the CSV sample address data to upload it, and select Connect Dataset. (Be patient. Sometimes this takes more than 30 seconds.)

6) Inspect your connected dataset.

7) Click the blue Create Map button.

8) Click the Edit Your Map button.

9) In your map data layer, click Add Analysis.

10) In the next screen of Analysis options, select Georeference, then click the Add Analysis button.

11) Back in your map data layer, under Georeference options, select Type > Street Addresses (scroll down to the bottom) for this sample data.

12) Under Parameters, for Column Street Address (abbreviated as Col. Street Ad.), select the "address" field for this sample data. Press the Apply button.

13) After Carto has attempted to geocode your address data, click Style This Analysis. Or, go to the map data layer and click the Style tab.

14) In Style options, for Aggregation select none (the default).

15) Under Style options:

- select Fill Number to change circle sizes
- enter a larger size, such as 13, to make our sample points more visible
- select Fill Color to change circle color
- switch from Solid (all points are same color) to By Value, and scroll down to Group (at the bottom) to automatically color by categories for this sample data. (Hint: If you don't see Group in the menu, click somewhere else and try it again.)

16) In the Pop-up tab, select a Window Style, then select boxes in Show Items to display.

17) In the Legend tab, click Select a Style to display information, and your color-coded groups from above should automatically appear on your map. (Hint: A legend may automatically appear after styling your markers by color.)

18) Before publishing your map: If you wish to rename it, do it now by selecting the three vertical dots next to the file name, and select Rename.

19) To publish your map on the web: Next to your map file name, click the blue "back" arrow (NOT your browser back button) to return to the data layer. Click the green Public button, and on the next screen, click the blue Publish button.

20) On the next screen, Get The Link generates a weblink to your map, and Embed It generates an iframe code to insert the live map in your website. For next steps, go to the Embed on Your Web chapters in this book.

21) If you make edits to your map, you must click the blue Update button to republish your map to the web.

Learn more

- Getting Started with Carto Builder https://carto.com/learn/guides/intro/getting-started-with-carto-builder

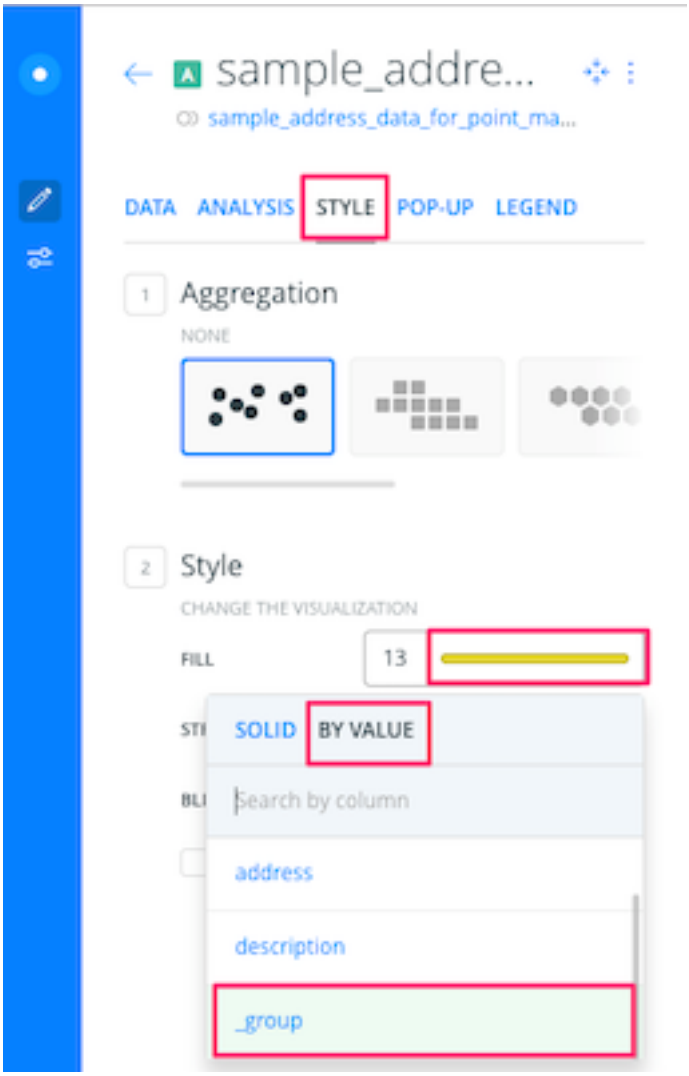


Figure 6.14: Image: Style points by value

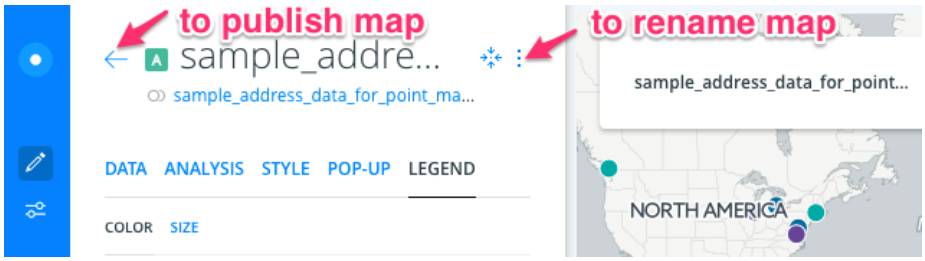


Figure 6.15: Image: Click to rename or publish your map

Filtered Point Map with Socrata Open Data

TODO: decide whether to keep or not; originally co-authored with Veronica.

Open data repositories recently launched by the State of Connecticut and the City of Hartford both use the Socrata platform, which offer user-friendly ways to view, filter, and export data. Also, the Socrata platform includes built-in support to create interactive charts and maps, and to embed them on your own websites. This tutorial demonstrates these features by creating an interactive point map of selected schools from the Connecticut Education Directory in the state data portal. The final product looks like this:

CT Schools Map 2015

Powered by Socrata

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 fire, crime, and property data repositories.

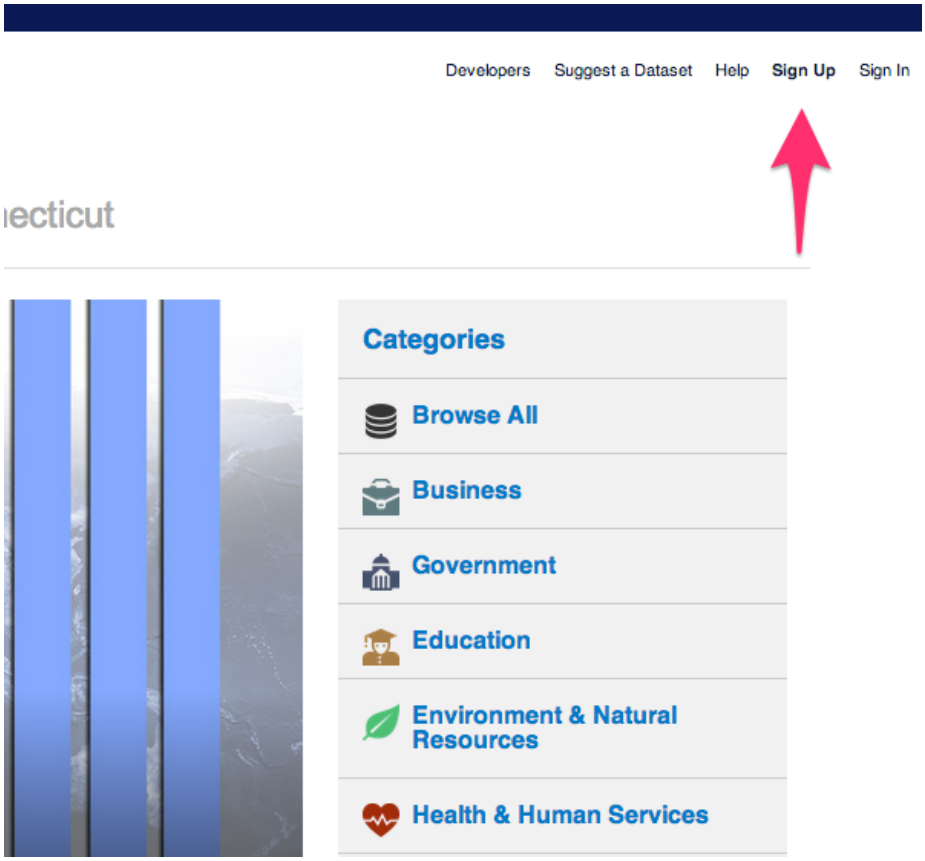
But there are limitations to creating your chart or map on an open data repository platform. First, if the agency stops using the platform, or changes the structure of the underlying data, your online chart or map may stop functioning. Second, you are usually limited to using data tables and geographic boundaries that already exist on that platform, since importing your own may not be an option.

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" data), and import it into your preferred dataviz tool, such as those described in other chapters of this book. A second, more advanced alternative, is to learn how to pull live data from the repository directly into your dataviz, using an Application Programming Interface (API), which requires coding skills that

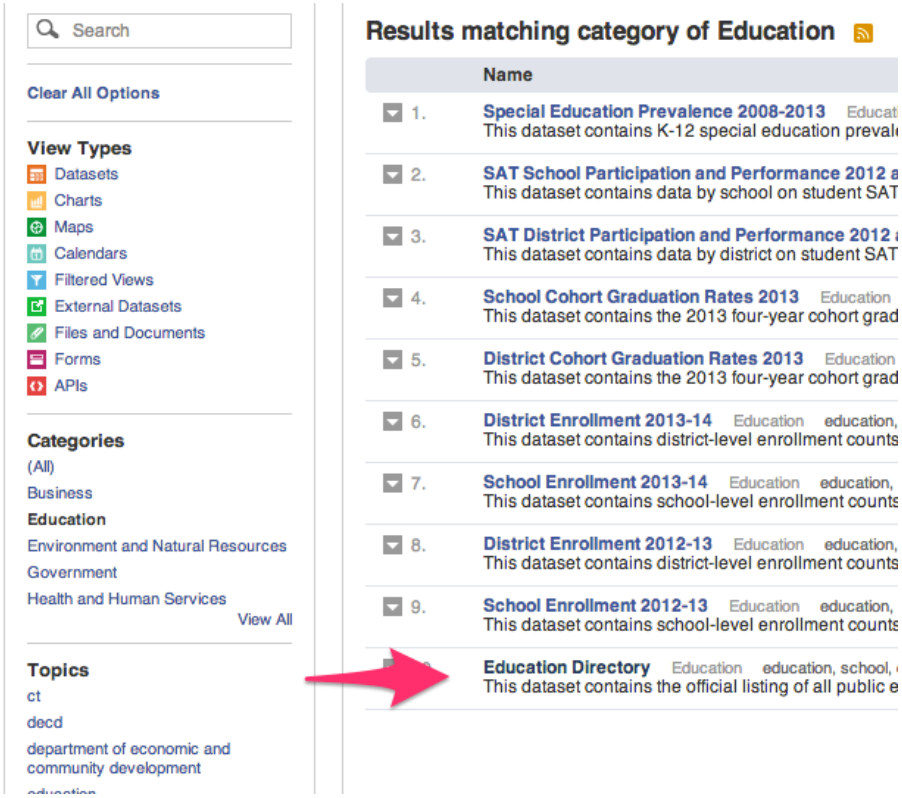
are beyond the scope of this tutorial. To learn more about the Socrata API: https://dev.socrata.com/.

Steps to create a filtered point map

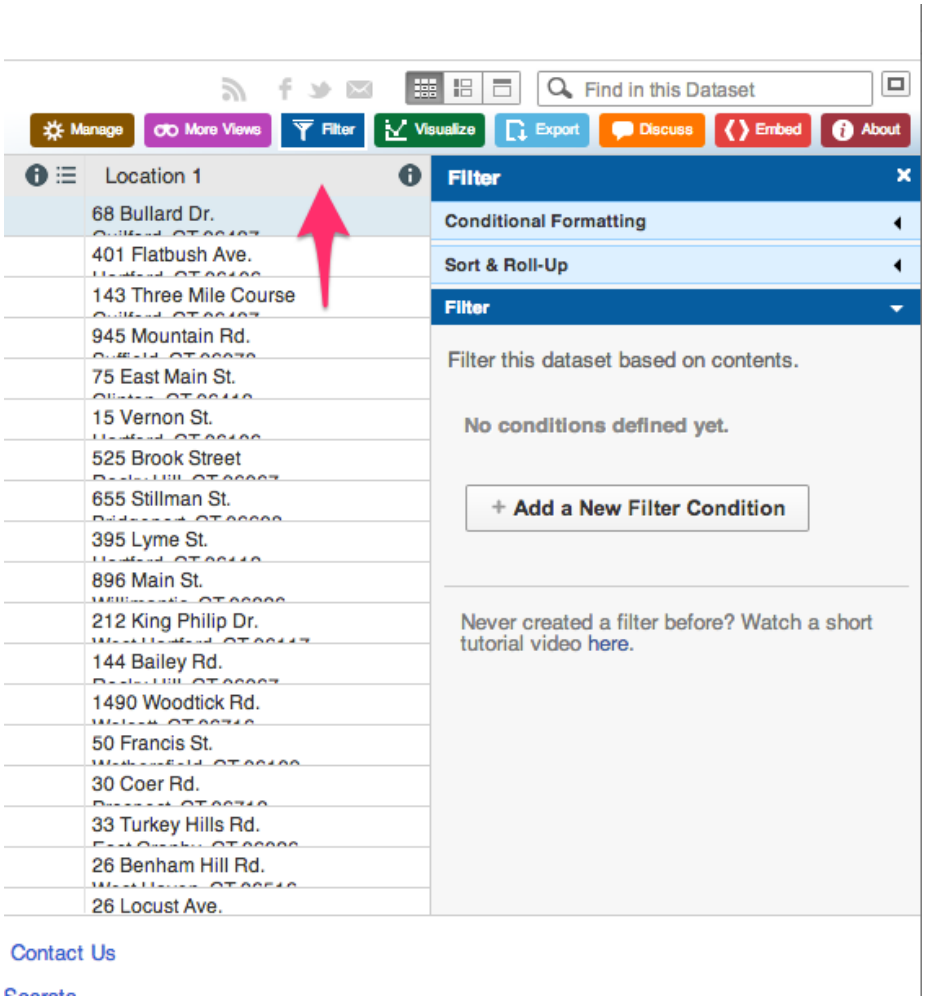
Sign up for a free account ID on any Socrata platform, such as https://data.ct.gov/signup. One account will work on all Socrata sites.



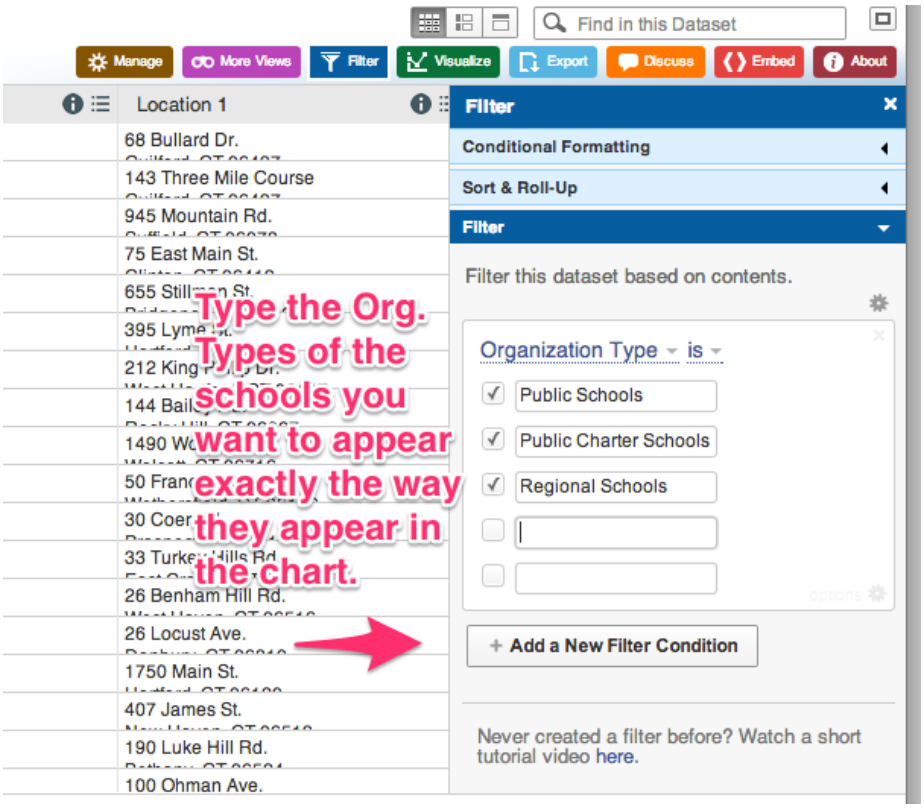
Select your desired dataset in Socrata. In this tutorial, we will use CT Open Data > Education > CT Education Directory. The data table must include a location column that includes geocoordinates. If there is address data but no geocoordinates, then post a suggestion to the Socrata site administrator to add a geocoded column.



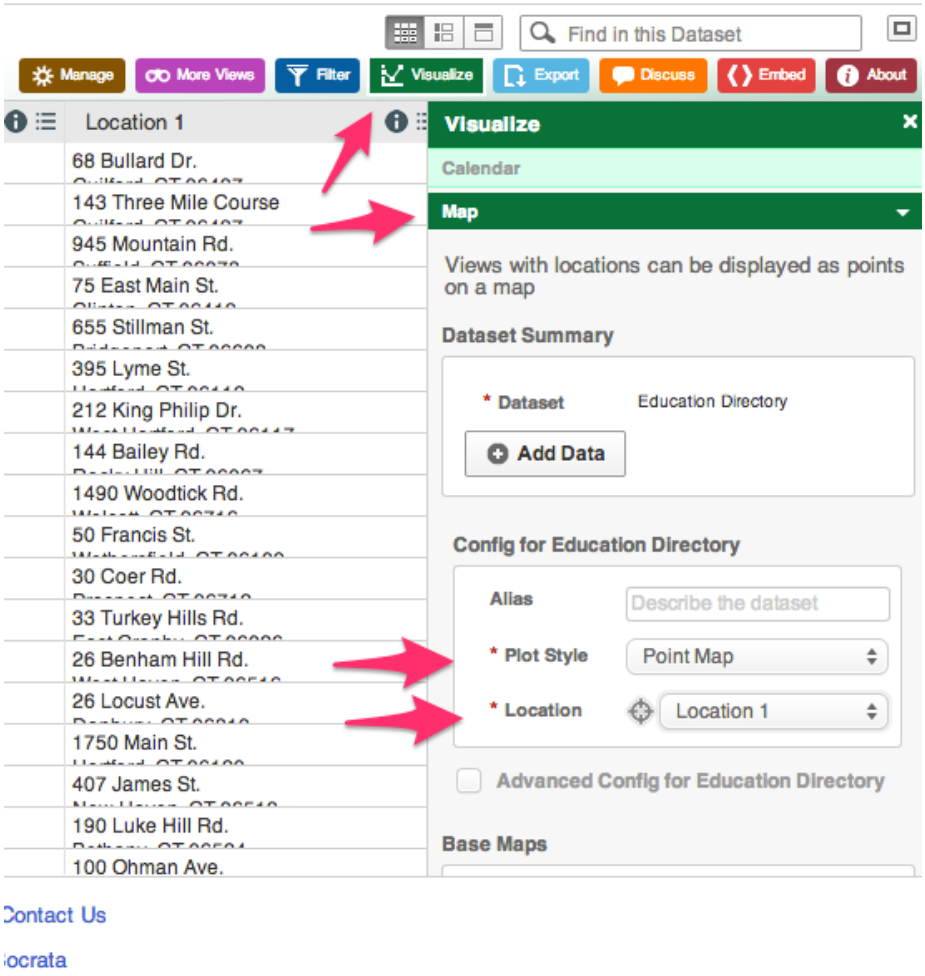
Filter the data to display only the desired rows. The CT Education Directory lists both district offices and school addresses, but for this map we only wish to display the latter. On the top-right corner of the table, click the Filter tab.



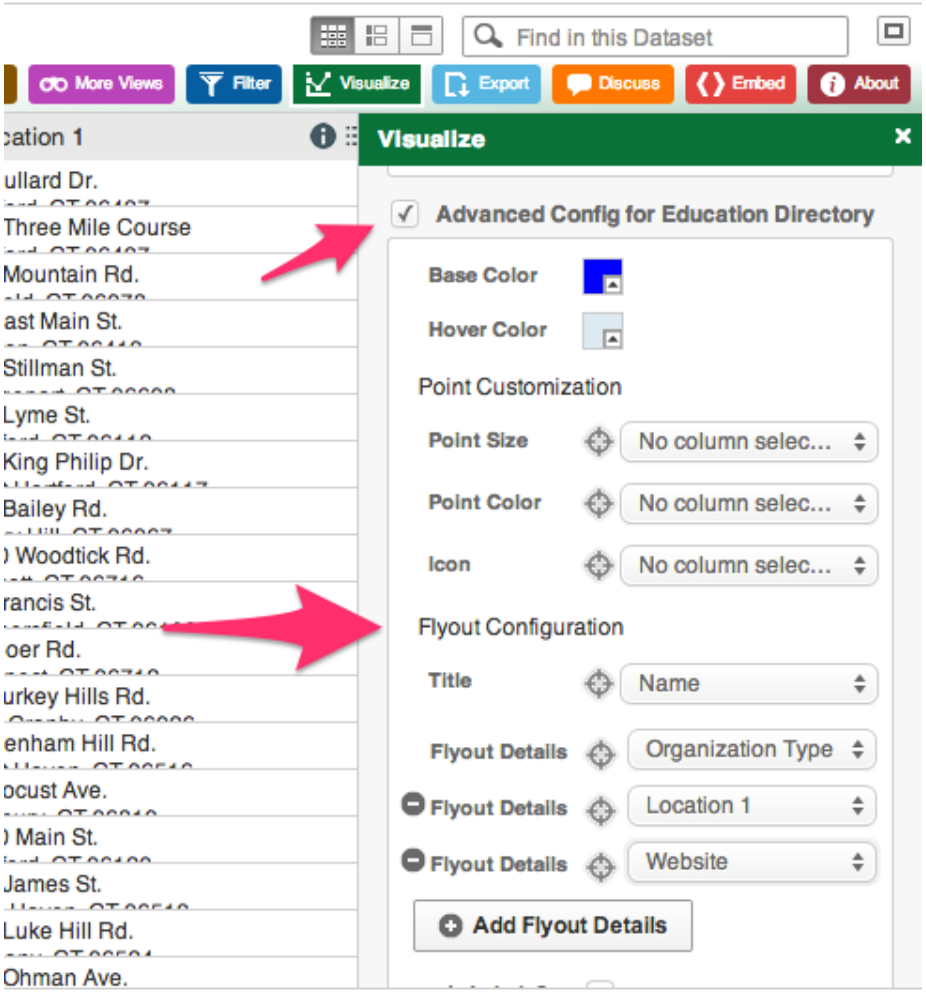
Add a New Filter Condition, which displays only the rows you select. In this tutorial, select "Organization Type" and "is", then type the exact name from the table, such as "Public Schools." Be sure to type it correctly or the filter may not work. If you wish to select multiple types, add a new filter condition for each. In this tutorial, we also will filter for other types: Public Charter Schools, CT Technical High Schools, Regional Schools, State Agency Facilities, Endowed and Incorporated Academies Schools, and Regional Education Service Center Schools.



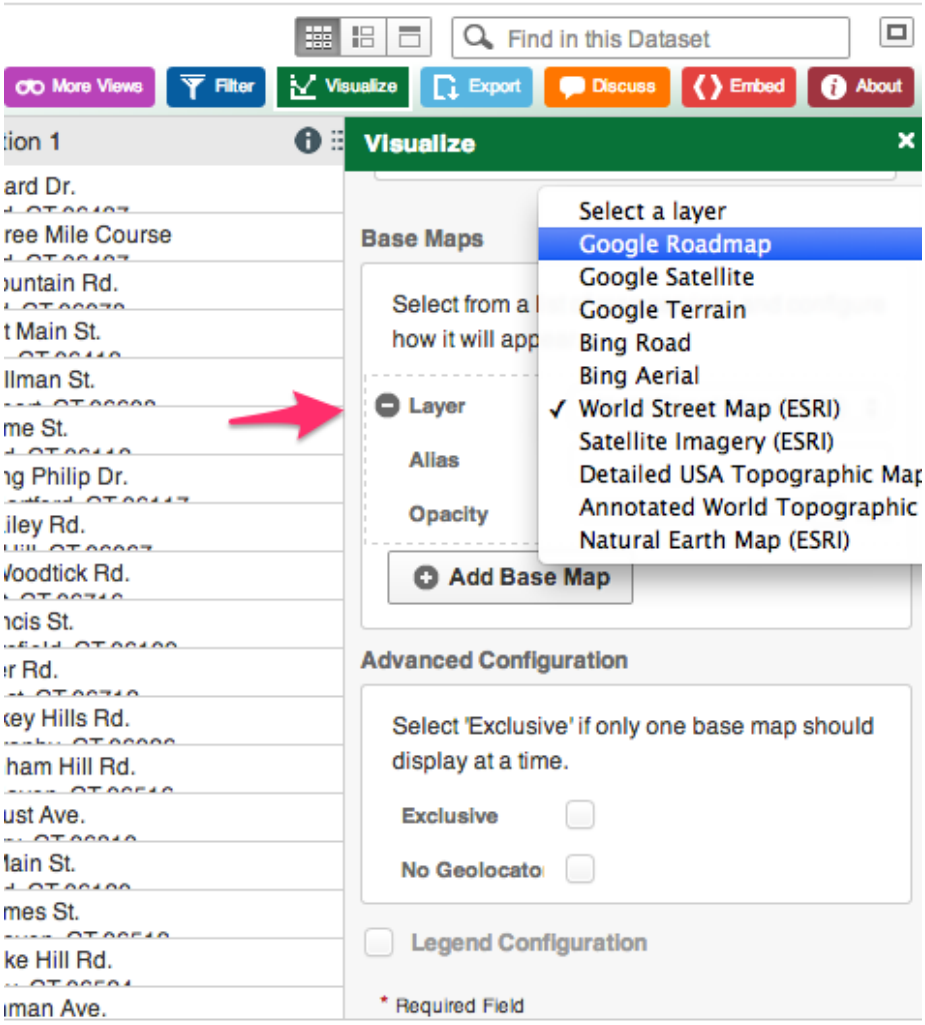
Select the Visualize tab and choose Map, which will display several options. First, under Config for Education Direction, select Point Map as the Plot Style, and choose the Location column to identify the geocoordinates.



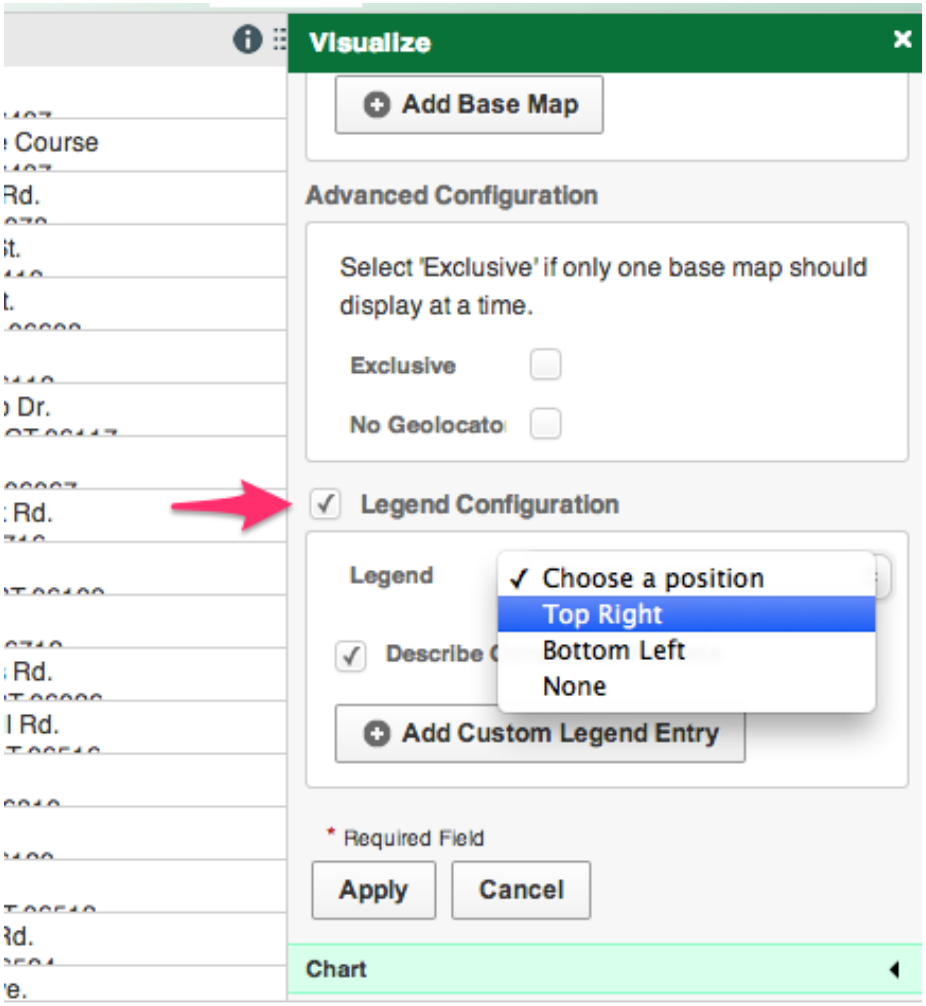
Further below in the Visualize > Map options, select the checkbox for Advanced Config for the Education Directory to edit the Flyout Details (similar to a pop-up information window) that displays details when users click on a map point. Select data items you wish to display, such as Title: Name, and additional Flyout Details: Organization Type, Location I, and Website. Further down, select the "w/o labels" checkbox to avoid displaying the column headers in your flyout details.



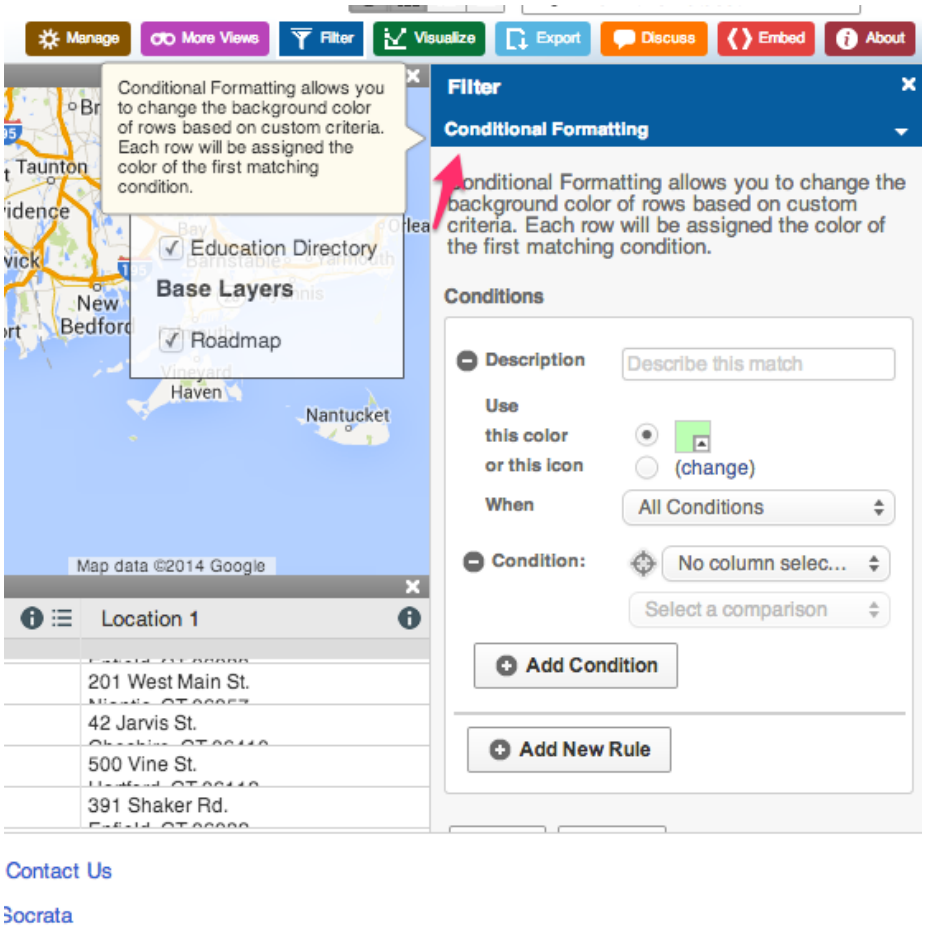
In Visualize > Map > Base Maps, select your desired background map, such as Google Roadmap.



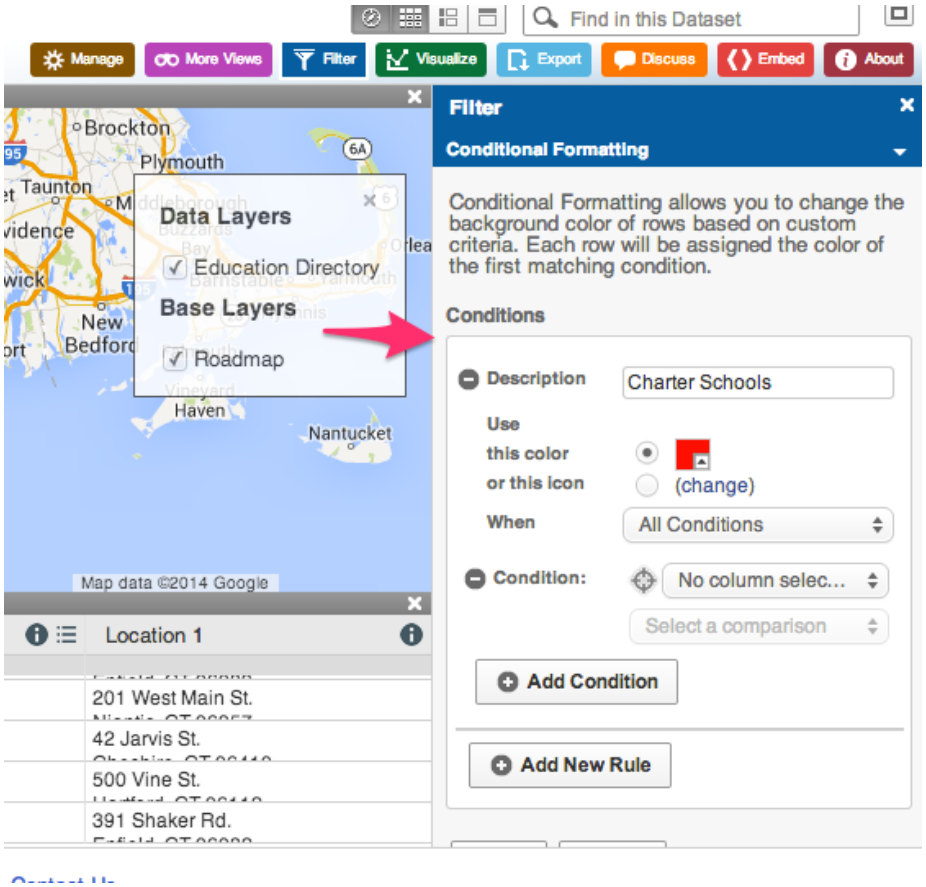
Add a legend to display once you build the map. In the Advanced Configuration area, select the Legend Configuration checkbox and mark its position. After selecting all of these map options, click Apply. Socrata will generate your map with default point colors. Double-check to make sure your data appear, and that your Visualize settings are correct, before moving to the next step.



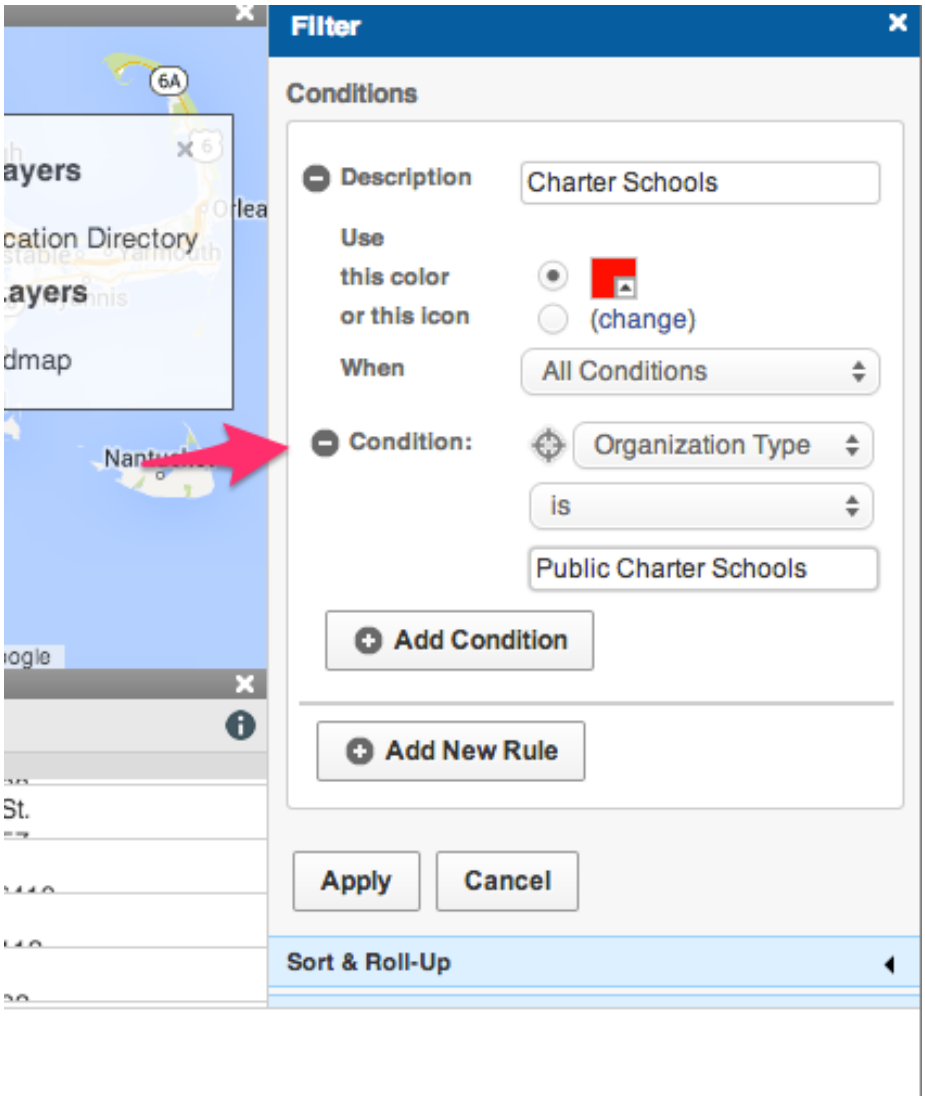
Assign point colors and legend labels by returning to the Filter tab, and select Conditional Formatting. Understand the difference between these two features. Previously, we used Filter to display only selected types of data (in this case, school buildings, rather than district administrative offices). Now, we will use Conditional Formatting to assign color codes and labels to our filtered data.



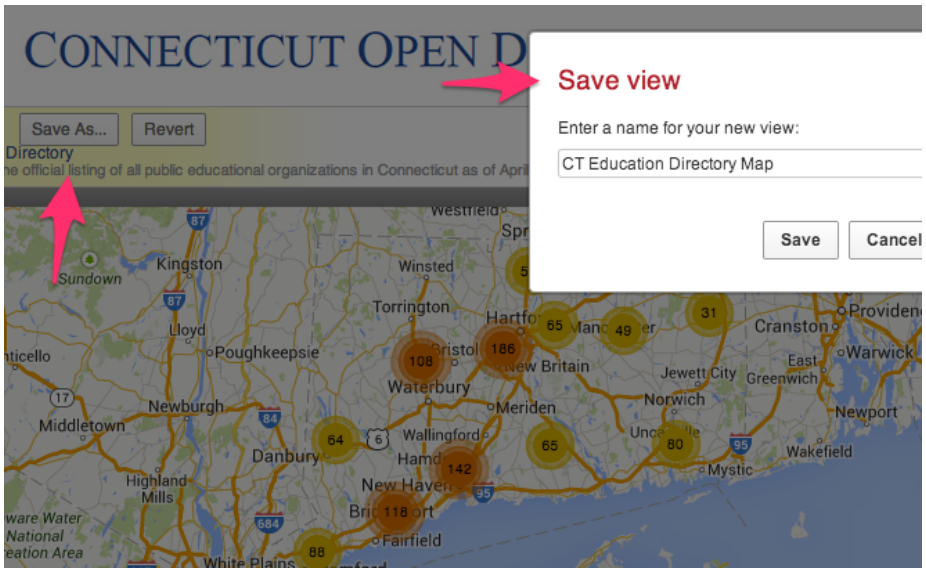
In the Conditional Formatting section, type a name into the Description that you wish to display in the legend. You may type a shorter name than the longer name that appears in the data table, such as “Charter Schools” instead of the longer “Public Charter Schools.” Also, select a color for each Description.



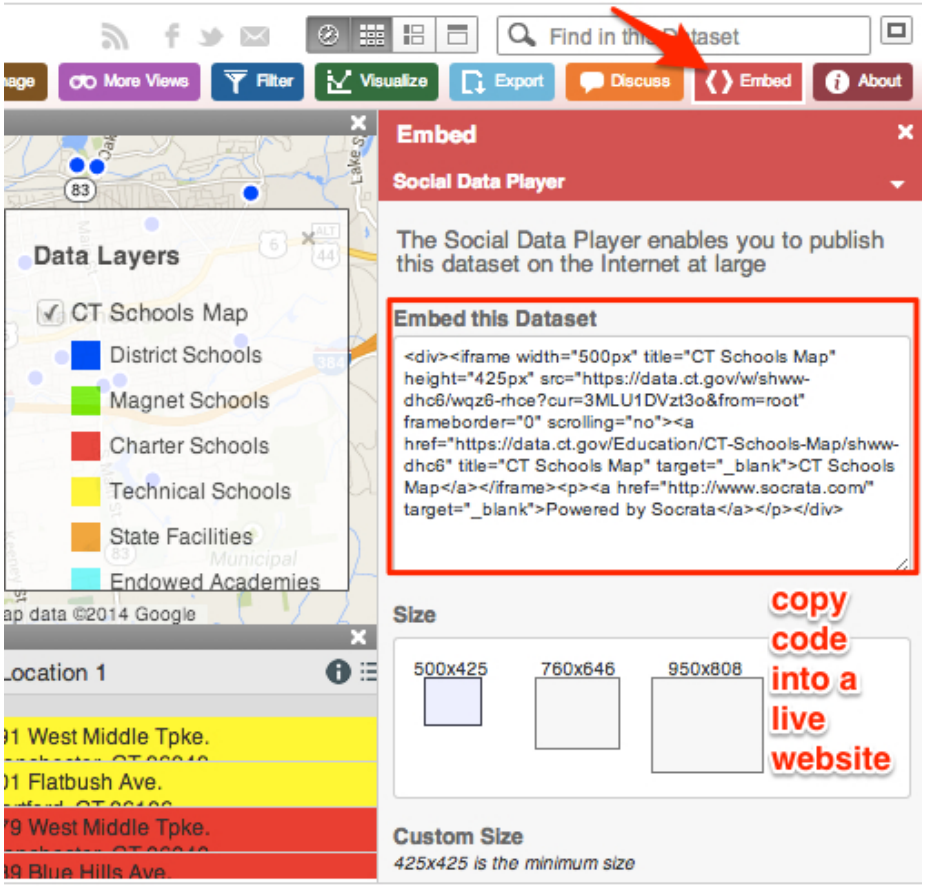
Continue to add Conditional Formatting by defining the data columns. In this example, select "All Conditions Apply," choose "Organization Type" and "Is", then type the category exactly as it appears in the data table (such as Public Charter Schools). For this map of schools in the CT Education Directory data table, we added several more types (Regional Schools, CT Technical High Schools, etc.) and also added a second rule to identify Magnet Schools (where Organization Type is Public Schools, and Interdistrict Schools is 1).



After setting all of your Conditional Formatting, press Apply at the bottom of the tab. Double-check that your visualization appears exactly as you wish, then Save As under an appropriate name. Note that your visualization will become **publicly visible** to other users on the Socrata open data platform, though you have the option to remove it via your individual profile view.



Visualizations created in the Socrata platform produce HTML iframe codes, which allows you to embed the dataviz in your own website. Select the Embed tab to view and copy the code. Then go to the Embed on the Web chapters in this book.



Polygon Maps and Storyboards with Social Explorer

TODO: decide whether to keep or not, since free license terms changed

The Social Explorer free edition http://socialexplorer.com offers one solution to creating colored polygon maps with US Census demographic data. Explore the embedded sample map below.

Advantages

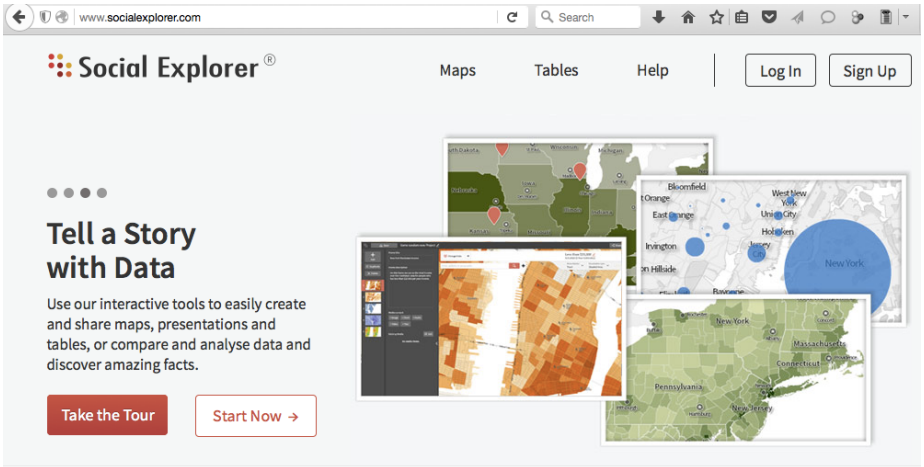
- Quick and easy-to-learn
- Free edition includes basic census data
- Export your static maps into presentation slides
- Share link or embed iframe to your interactive map

Limitations

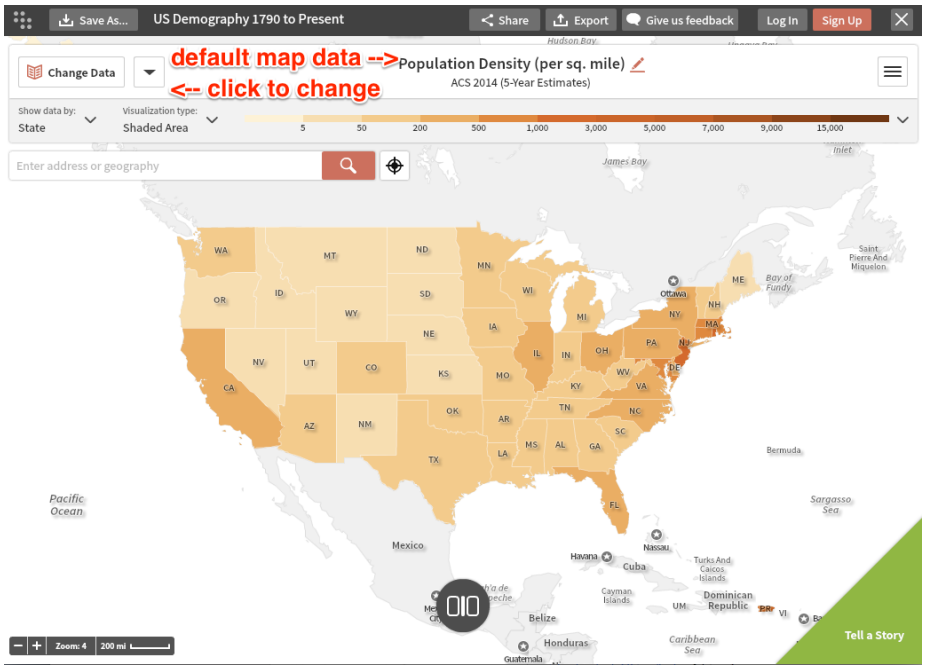
- Maps are limited to the demographic data inside the tool.
- Polygon map boundaries are limited to state, county, census tract. The tool does not display municipal data for cities, towns, etc.
- Full census and historical data requires professional subscription.
- Pro subscription available through several academic libraries, but few public libraries.

Quick overview of features

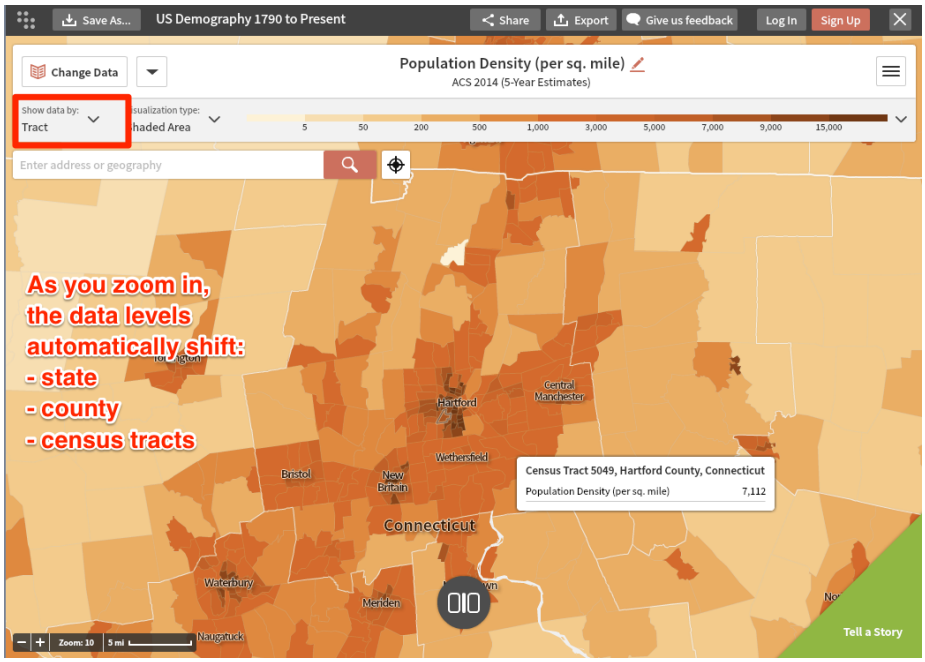
Start at the Social Explorer website http://socialexplorer.com and click on Maps. This tutorial demonstrates features available on the free edition.



The default map view shows US population density, based on the American Community Survey (ACS) 5-year estimates. Click the Change Data button to explore other options.

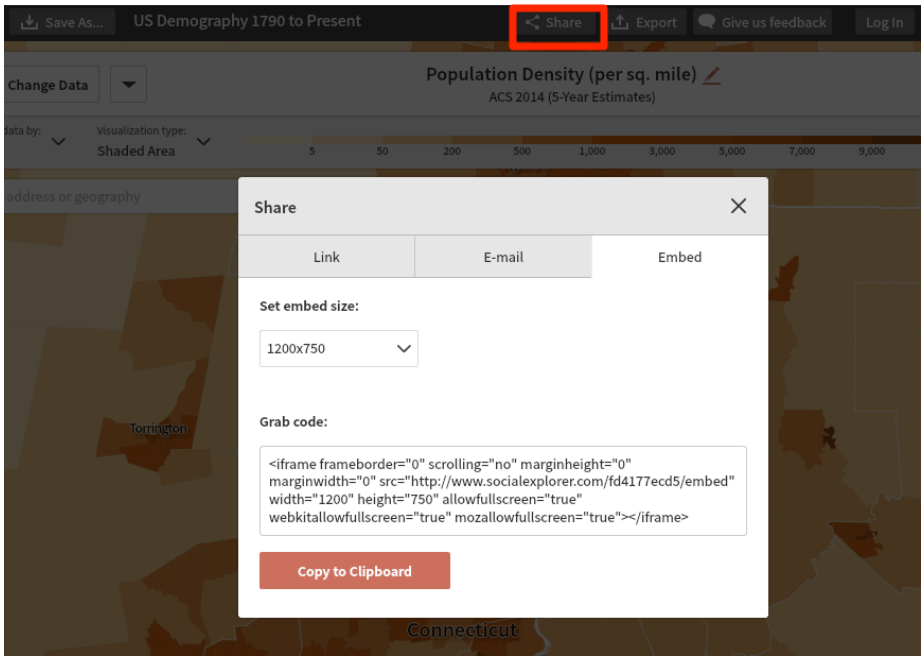


Geographic boundaries automatically change with the zoom level. As you zoom in, the data levels automatically shift from state, to county, to census tract.

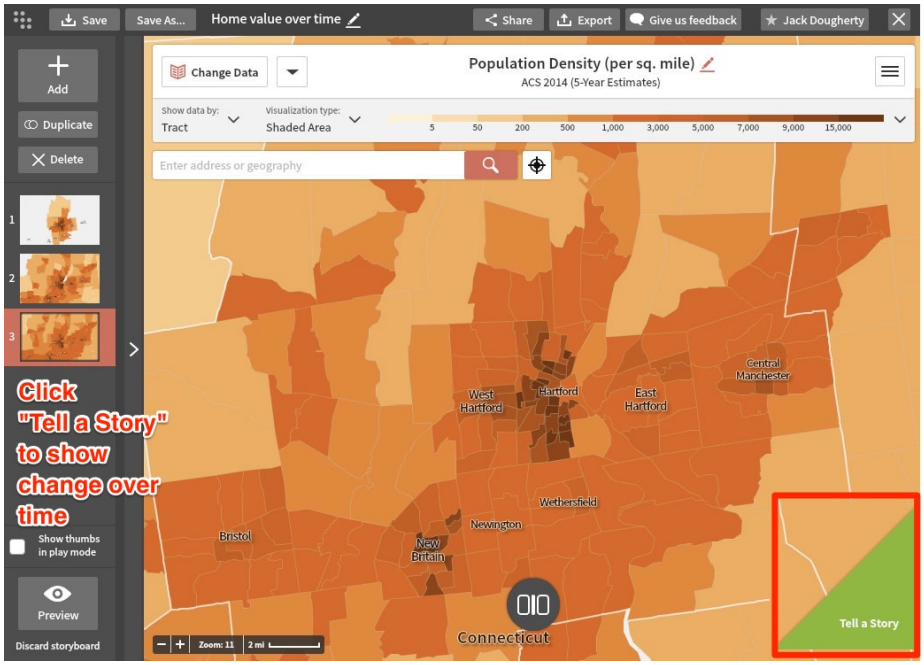


Click the Share button to copy the link to your map, or the iframe code to

embed it inside your own website.



Create a free account to save your online map views. Click the Tell a Story button, add a series of interactive map views, and show change over time.



TODO: Is this still true? All of the steps above can be done with the free version, but data is limited. Check if an academic library near you has a professional subscription.

Polygon Map with Tableau Public

Tableau Public is freely-available software that contains powerful tools to quickly create interactive polygon maps for common boundaries (such as US States, or World Nations.) If you need to create customized maps for less-common boundaries, see our chapter on Leaflet Maps with Google Sheets in this volume.

Important: Tableau Public is designed to publicly display your data, which makes this free tool very appropriate for educators, journalists, non-profit organizations, or other users who wish to openly share their map. If you desire a private tool to restrict your data, Tableau offers other tools that require payment.

See also the Tableau Public support page https://www.tableau.com/support/public for additional resources, including video tutorials.

1) Download and install the free Tableau Public tool, available for Mac or Windows, at https://public.tableau.com/en-us/s/download. Do not confuse with other Tableau products that require payment. Installation may require up to 5-10 minutes.

2) Click this link and Save to download to your computer: us-sample-data in CSV format. CSV means comma-separated values, a generic spreadsheet format that most tools can easily open. For help with downloading, see this short video tutorial.

3) Open the us-sample-data.csv file with any spreadsheet tool to view its contents.

4) Launch Tableau Public. In the Connect column of the first screen, click "Text file" to connect to the CSV file you downloaded above. (If you had an Excel file in .xlsx format, you would click that instead.) Navigate to select the us-sample.data.csv file on your computer.

5) At first, Tableau Public does **NOT** recognize the names of US areas, which initially appear simply as "text" values (with the "Abc" symbol). Click and hold down the mouse directly on the "Abc" symbol, and use the drop-down menu to convert to Geographic role > State/Province. A tiny globe symbol will appear to show that Tableau Public now recognizes this column as geographic data, which is essential in order to make a map.

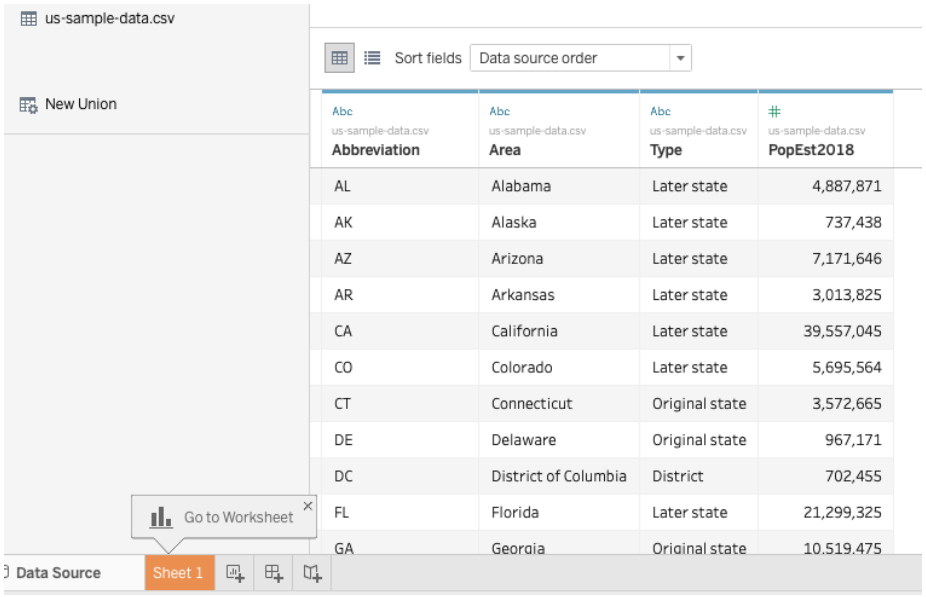


Figure 6.16: Column displayed as text data

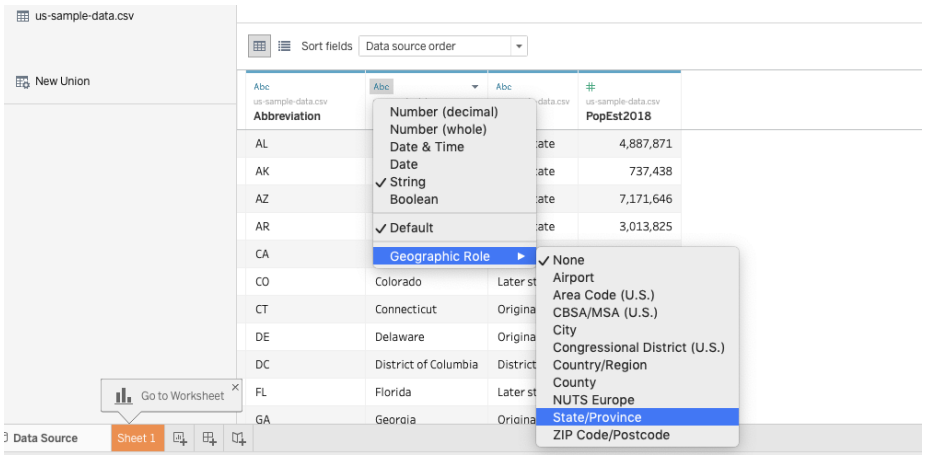


Figure 6.17: Column converted to geographic data

6) Go to the Worksheet view, by clicking on "Sheet 1" in the bottom-left corner. The goal is to build a polygon map, based on the dimensions and variables provided by Tableau Public.

Step A - Drag the "Area" dimension to the middle of the worksheet to create the geographic areas

Step B - In the "Marks" panel, change the drop-down menu from "Automatic" to "Map"

Step C - Drag the "Type" dimension into the "Color" box of the Marks panel to show polygon colors according to type

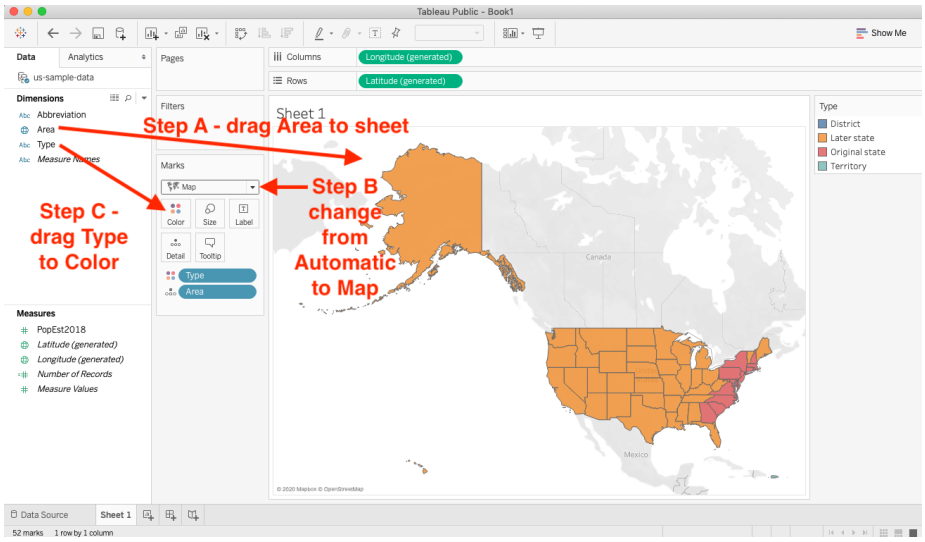


Figure 6.18: Steps A-B-C above

Optional: Add more items, such as "Abbreviation" dimension to "Label" box to display state abbreviations, or "Area" dimension to "tooltips" to display on mouseover.

7) To display your map online, click "Dashboard" tab in the bottom-left corner.

8) Drag "Sheet 1" (the default name of your map) into your dashboard. Also, drag the map legend from the corner into the lower body of the map (or choose other legend options).

9) To publish your map online, go to File > Save to Tableau Public As... This will require you to create a free Tableau Public Account.

10) Modify your final online product as desired, and see options to embed elsewhere on the web.

Chapter 7

Embed On Your Web

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 **

Enroll in our free online course **link to do**, which introduces these topics in the brief video below, and offers more exercises and opportunities to interact with instructors and other learners.

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 with step-by-step tutorial

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

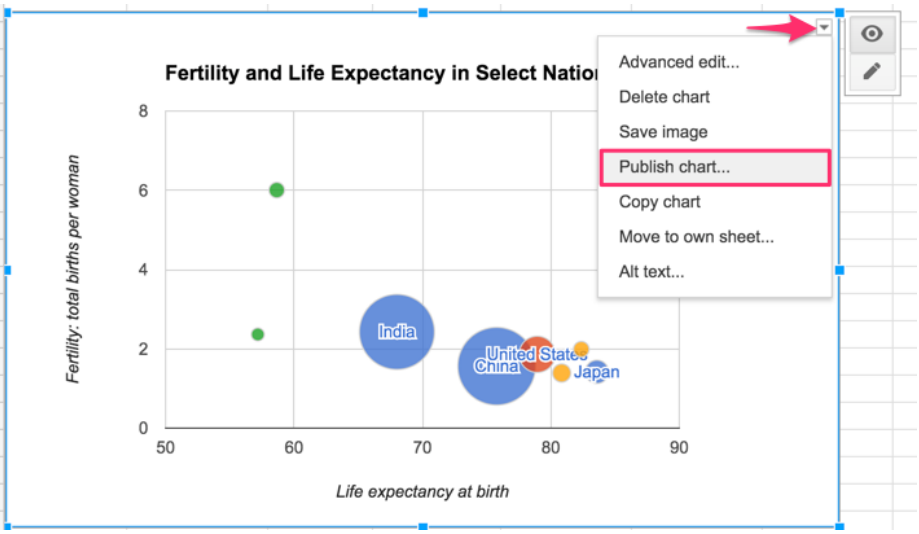


Figure 7.1: Screenshot: Drop-down menu to publish a Google Sheets chart

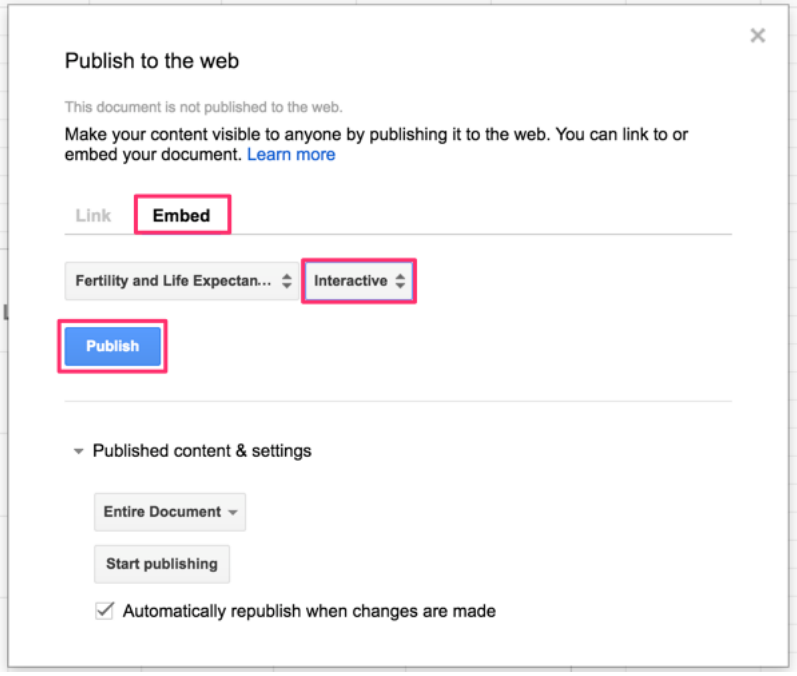


Figure 7.2: Screenshot: Publish to the web for a Google Sheets chart

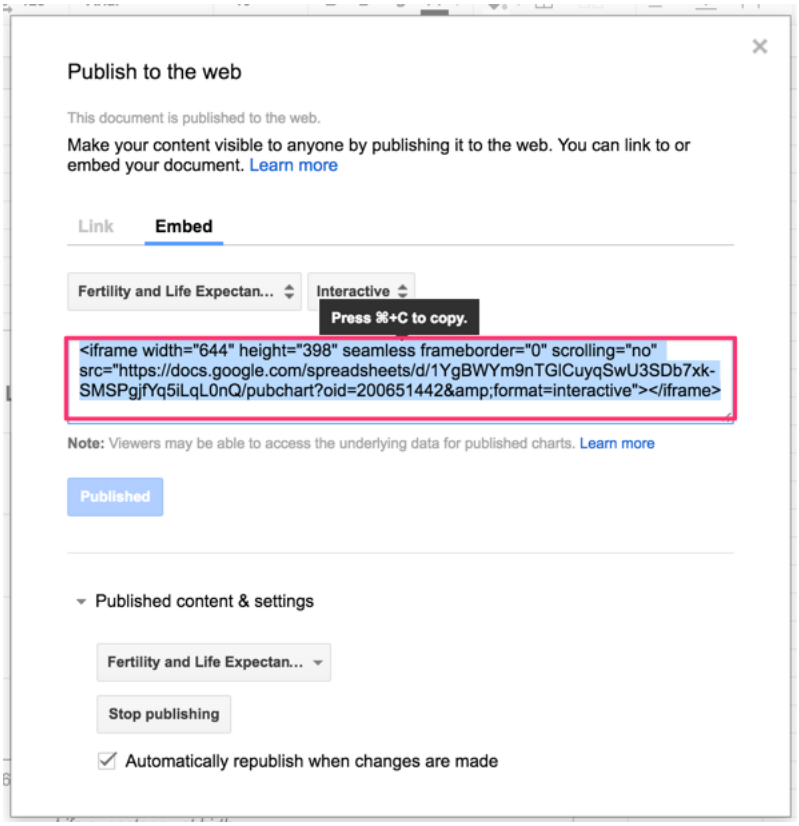


Figure 7.3: Screenshot: Copy the iframe code from a Google Sheets chart

- 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/RESPOSITORY"></iframe>
```

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

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

Video tutorial and step-by-step instructions

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.

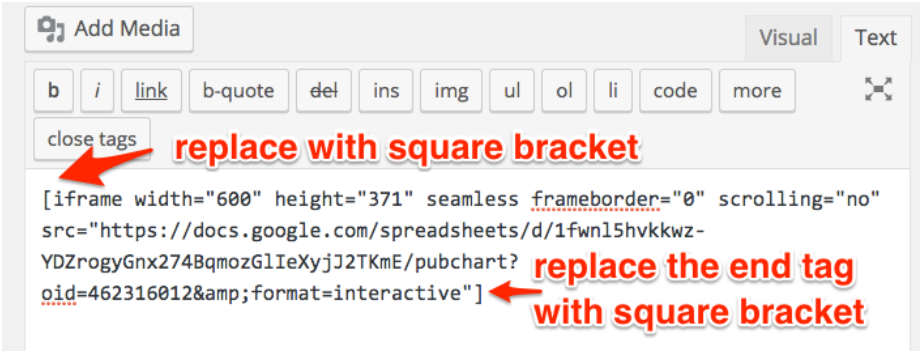
5) To embed the iframe in a WordPress.org site, the iframe plugin must be installed, as explained in the Embed with iframe on WordPress.org chapter. **TO DO** fix self-reference

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.

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
```

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 ( ]).



Your modified code should look like this:

```
[iframe width="600" height="371" seamless frameborder="0" scrolling="no" src="https://docs.google
```

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.

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.

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.

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.

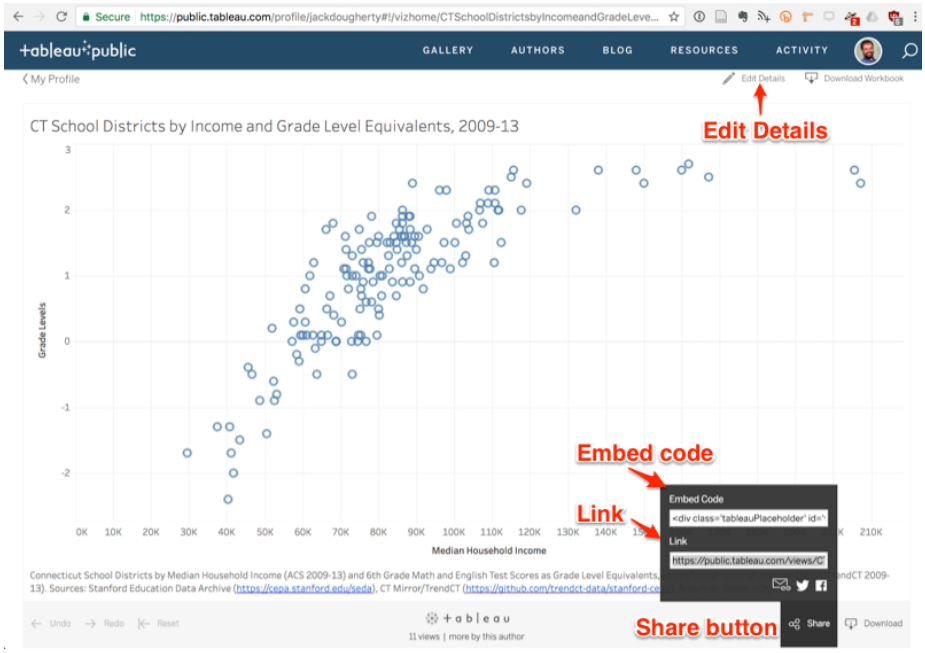


Figure 7.4: Screenshot: Edit and Share buttons in Tableau Public web page

```
<!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 h
</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.

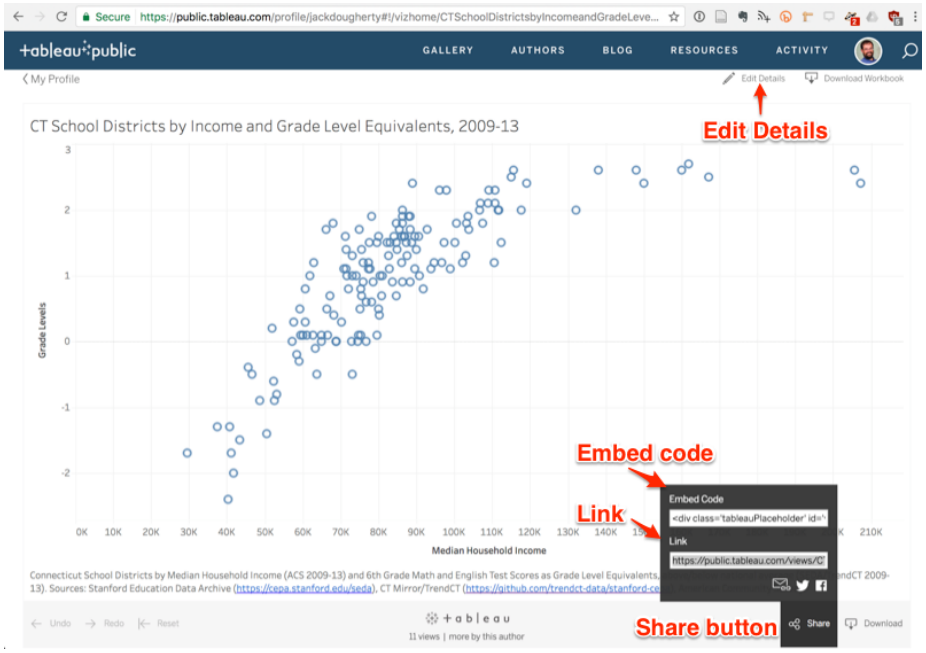


Figure 7.5: Screenshot: Edit and Share buttons in Tableau Public web page

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?:showVizHo
```

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?:show
```

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?:show
```

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/Shee
```

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/Shee
```

Learn more

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

Chapter 8

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 fork, 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 more advanced tools, such as GitHub Desktop and Atom Editor to work more efficiently on your personal computer.

If problems arise, turn to the Fix Common Code Errors 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!

Fork, 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 that creates an interactive map using Leaflet, an open-source code library that's very popular with coders, journalists, businesses, and government agencies. Many people chose 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 covered in previous chapters, such as Google My Maps or Tableau Public, Leaflet requires you to write (or copy and paste) several lines of code, which need to be hosted on a web server so that other people can view your map in their web browser. Fortunately, we can do all of these steps in our web browser on GitHub. This means you can do this on any type of computer: Mac, Windows, Chromebook, etc.

Here's an overview of the key steps we'll cover in this section:

- Get a free GitHub account and fork your copy of a simple Leaflet map code template
- Edit the Leaflet map title, starting 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 8.1.

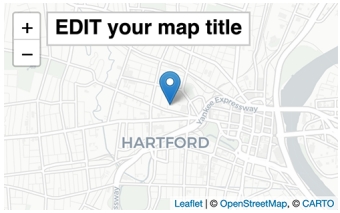


Figure 8.1: Create your own version of this simple interactive Leaflet map.

Before you begin, 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.

1. 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

2. To create your own copy of our template, click the Fork button as shown in Figure 8.2.

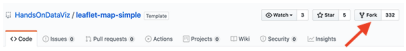


Figure 8.2: Click the Fork button to make your own copy of the code template.

When you fork someone else's repo, in the upper-right corner of your browser you should see something like `USERNAME/leaflet-map-simple` forked from `HandsOnDataViz/leaflet-map-simple`, where `USERNAME` refers to your GitHub account username. This proves that you copied our template into your GitHub account. This very simple repo includes 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 that contains the map code.

Tip: By design, GitHub allows you to fork a repo *one time*, so that you don't accidentally create two versions with the same name. If you wish to create a second version, go to the Create a New Repo and Upload Files on GitHub section of this chapter.

3. Click on the `index.html` file to view the code, as shown in Figure 8.3.

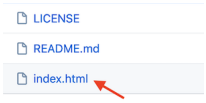


Figure 8.3: Click the Index file to view the code.

In case this is the first time you're looking at computer code, we've inserted several "code comments" to explain what's happening. The first block you see is written in HyperText Markup Language (HTML) that 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, written in a language called Cascading Style Sheet (CSS). 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.

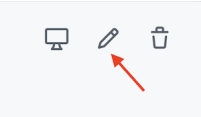


Figure 8.4: Click the pencil button to edit the code.

4. To edit the code, click on the the pencil symbol in the upper-right corner, as shown in Figure 8.4.

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.

5. 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 which appear on both ends inside the `<` `>` symbols.

6. To save your edit, scroll to the bottom of the page and click the green *Commit Changes* button, as shown in Figure 8.5.

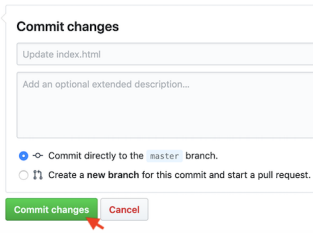


Figure 8.5: Click the green *Commit Changes* button to save your edits.

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 by default, GitHub commits your changes directly to the `master` branch of your code, which we'll explain later.

TODO: At this writing, GitHub is considering changing the default branch from "master" to "main". Stay tuned for updates!

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 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 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.

7. To access GitHub Pages, scroll to the top of your repo page and click the *Settings* button as shown in Figure 8.6.

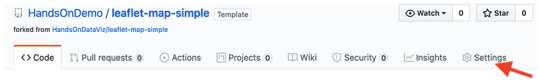


Figure 8.6: Click the *Settings* button to access GitHub Pages and publish your work on the web.

8. In the Settings screen, scroll down to the GitHub Pages area, and use the drop-down menu to change *Source* from *None* to *Master Branch*, as shown in Figure 8.7. There is no *commit* or *save* button here, and the change will happen automatically. This step tells GitHub to publish a live version of your map on the public web, where anyone can access it in their browser, if they have the web address.

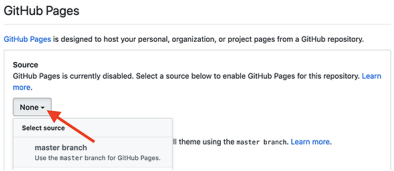


Figure 8.7: Under GitHub Pages, switch the source from *None* to *Master* as shown in this animated GIF.

9. Scroll back down to the GitHub Pages area 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 8.8.

10. Click on the new tab to view your live map, with your new title at the top. GitHub Pages automatically generates a public web address for your repo in this format, `https://USERNAME.github.io/leaflet-map-simple`, where `USERNAME` is your GitHub account username. Remember why we told you not to create your account with a username like `DrunkBrownieChef6789`? Here's why.

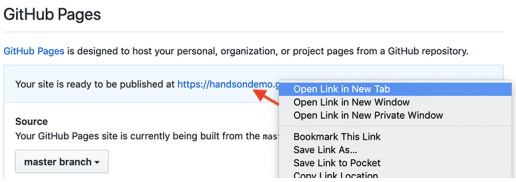


Figure 8.8: Under GitHub Pages, double-click your published map link as shown in this animated GIF.

Tip: If your map does *not* appear right away, wait up to 30 seconds for GitHub Pages to finish processing your edits. Then do a "hard refresh" to bypass any saved content in your browser 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)

Note: GitHub creates two different types of web addresses, where you should replace `USERNAME` and `REPOSITORY` with your own:

- The GitHub repo address, where you can edit your code:
  - `https://github.com/USERNAME/REPOSITORY`
- The GitHub Pages address, which publishes a live version of your code:
  - `https://USERNAME.github.io/REPOSITORY`

The live version of your code points to the `index.html` page by default, so you do not need to include it in the web address.

Finally, web addresses are *not* case sensitive, meaning that you can type all of it in lower-case to save time!

Tip: When working with the GitHub web interface, keep two browser tabs open. The first tab contains your GitHub repo, where you can edit your code. The second tab contains your GitHub Pages live version, where you can view the results of your edits. By opening your live map in a new tab, you can easily go back to edit your code repo in the first tab, and flip back to view the live results in the second tab.

Let's go back to your GitHub repo and edit the GitHub Pages links so that they point to *your* live map, in place of *our* live map.

Figure 8.9: On your first browser tab, click the repo title.

11. 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 8.9.

If you can't find your first browser tab, you can retype your repo home page address in this format, and insert your GitHub username: `https://github.com/USERNAME/leaflet-map-simple`.

12. Copy the web address of your live map (in your second browser tab) and paste it into two places on your repo home page (in your first tab). First, click the *Edit* button near the top-right corner of your repo, paste your link there, and save. Second, open the `README.md` file or scroll down to the bottom of the repo home page, click the pencil symbol to edit it, paste your link under the label *(replace with link to your site)*, and scroll down to commit the change. See both steps in Figure 8.10.

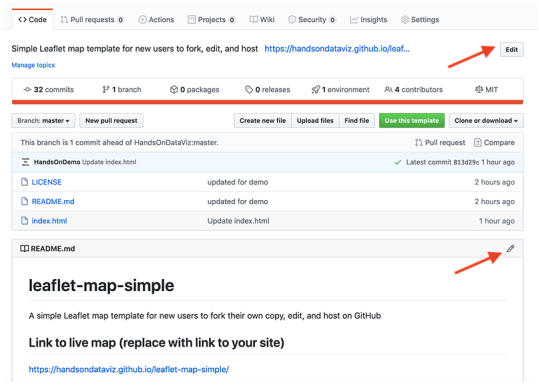


Figure 8.10: Paste the link to your live map at the top of your repo page, and also in your README page.

Pasting both of these links helps point people who discover your GitHub repo to *your* live map, rather than *our* version for this book.

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.

13. 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, and find your coordinates with online tools such as LatLong.net or Google Maps. **TODO:** Show how to find coords in GMaps here, or link if it appears earlier in the book.

```
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
```

14. 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 Code Errors 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 code templates in chapter 9 and Leaflet map code templates in chapter 10.

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 forked an existing repo on GitHub, the next step is to learn how to create a brand-new repo and upload different types of files. These skills will be helpful for several scenarios later in this book. First, since GitHub allows you to create only *one fork* of an existing repository, if you wish to make a *second* copy, you'll need to download the code and upload it into a new repo. Second, Chapter 9 on Chart.js code templates and chapter 10 on Leaflet map code templates allow you to upload your own files to create data visualizations. Once again, we'll demonstrate how to do all of these steps in GitHub's beginner-level browser interface.

Let's start with GitHub's *one fork* rule. Imagine that you wish to create a second copy of the leaflet-map-simple template described in the prior section. If you attempt to create a second fork, GitHub will "gray out" the Fork button and display an error message stating that you "Cannot fork because you own this repository. . . " as shown in Figure 8.11. There's a good reason for GitHub's one-fork rule: it's designed to prevent you from accidentally creating a second copy, with the same name as your first fork, which would overwrite and erase your previous work.

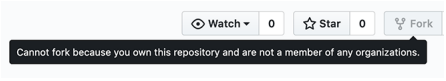


Figure 8.11: GitHub's one-fork rule prevents you from creating a second fork of a repo.

So how do you create a second copy of a GitHub repo? We'll show you two solutions. The first solution, if it exists in your case, is easy. Look for a green "Use this template" button in the upper-right screen, as shown in Figure 8.12, and if you see it, click it. GitHub will ask you to create a brand-new repository name for the second copy of this template, to avoid confusing it with the first copy you made. The "Use this template" button will appear only if the people

who created the GitHub repo set it up as a template, and that's exactly what we, the authors of this book, did to make it super-simple for all of you to create multiple copies of our GitHub repos.

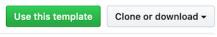


Figure 8.12: If a green *Use this template* button appears, click it to work around GitHub's one-fork rule.

But what if you're trying to make a second copy of a GitHub repo where the *Fork* button is grayed-out and there's no green *Use this template* button? Here's a recommended workaround that follows three general 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

Imagine that you've already created one fork of the leaflet-map-simple repo, as we did in the prior section. You wish to create a second copy, but the *Use this template* green button does not appear, either because the repo was created before that feature existed, or the people who created the repo didn't set it up that way.

1. Click on the gray *Clone or download* drop-down menu button on the right-side of the screen, as shown in Figure 8.12, and select *Download ZIP*. 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 `leaflet-map-simple-master` with three files: `index.html` and `LICENSE` and `README.md`. The word `master` refers to the master branch of your repo.

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 8.13.

4. On the next screen, GitHub will ask you to create 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.

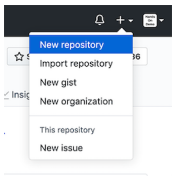


Figure 8.13: Click the plus (+) symbol in upper-right corner to create a new repo.

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 8.14.

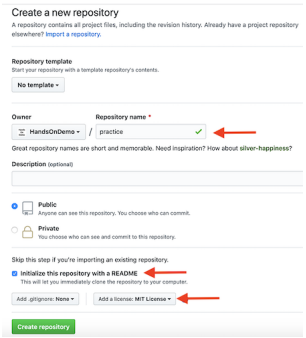


Figure 8.14: After naming your new repo, check the box to *Initialize this repo with a README* and *Add a license* to match the code (select "MIT").

Your new repo will have a web address similar to `https://github.com/USERNAME/practice`.

5. On your new repo home page, click the *Upload Files* button, near the middle of the screen, as shown in Figure 8.15.

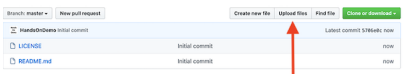


Figure 8.15: Click the *Upload Files* button.

6. Upload the `index.html` file that you previously downloaded to your local computer by dragging-and-dropping it into the upload area of your GitHub repo in your browser, as shown in Figure 8.16. 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.

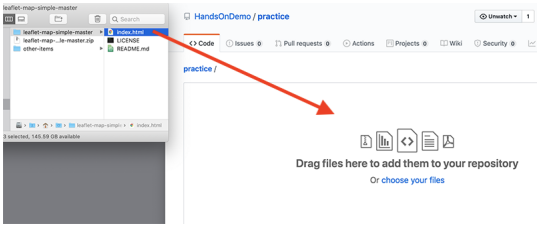


Figure 8.16: Drag-and-drop the file to the upload screen.

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 Fork, 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 8.17. 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.

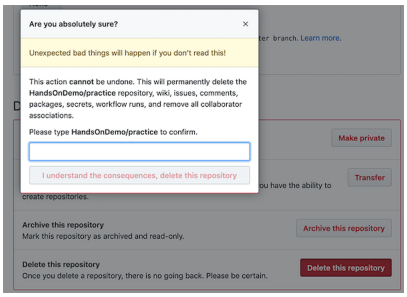


Figure 8.17: After clicking the Delete Repository button, GitHub will ask you to type your username and repo name to confirm.

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, but it can be very slow, especially if you need to modify 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 Fork, Edit, and Host a Simple Leaflet Map Template section of this chapter. Click the *Clone or download* button on the right side, and select *Open in Desktop*, as shown in Figure 8.18. The next screen will show a link to the GitHub Desktop web page, and you should download and install the application.

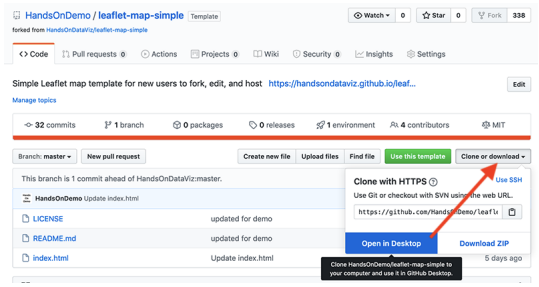


Figure 8.18: In your GitHub web repo, click *Clone or download* and *Open in Desktop* to download and install GitHub Desktop.

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 8.19, 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.

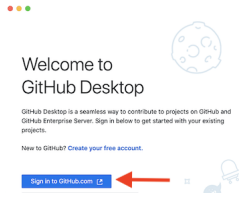


Figure 8.19: Click the blue *Sign in to GitHub.com* button to link GitHub Desktop to your GitHub account.

3. In the next setup screen, GitHub Desktop asks you to configure Git, the underlying software that runs GitHub. Confirm that it displays your username and click *Continue*, as shown in Figure 8.20.

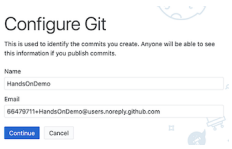


Figure 8.20: Click the *Continue* button to authorize GitHub Desktop to send commits to your GitHub account.

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 8.21.

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

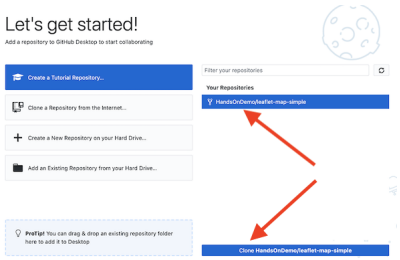


Figure 8.21: Select your "leaflet-map-simple" repo and click the *Clone* button to copy it to your local computer.

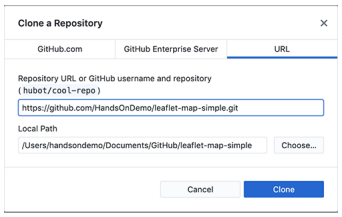


Figure 8.22: Select the Local Path where your repo will be stored on your computer, then click *Clone*.

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 8.23.

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 8.24. 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 8.25.

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 8.26. Open your `index.html` file and edit the title of your map, around line 22, then save your work.

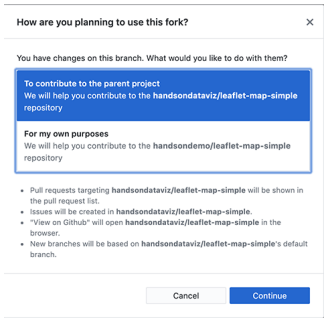


Figure 8.23: If asked how you plan to use this fork, select the default "To contribute to the parent project" and click *Continue*.

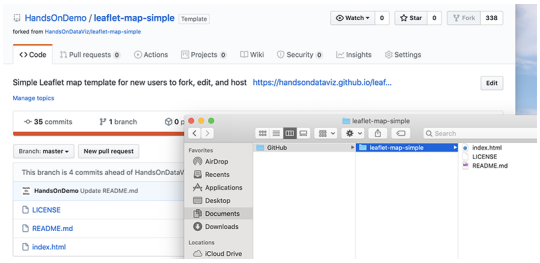


Figure 8.24: Now you have two copies of your repo: in your GitHub web account (on the left) and on your local computer (on the right, as shown in the Mac Finder). Windows screens will look different.

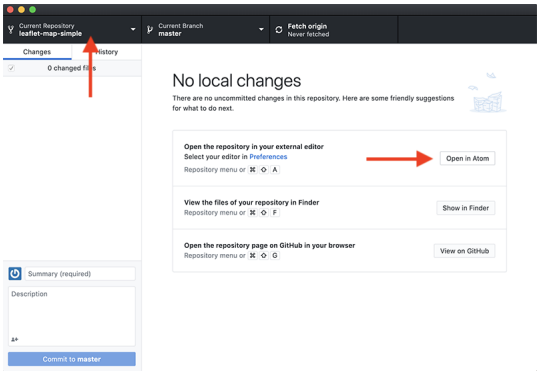


Figure 8.25: In GitHub Desktop, confirm the Current Repo and click the *Open in Atom* button to edit the code.

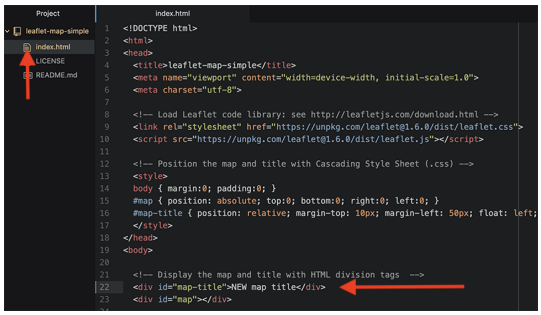


Figure 8.26: Atom Editor opens your repo as a "project," where you can click files to view code. Edit your map title.

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 8.27. 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.

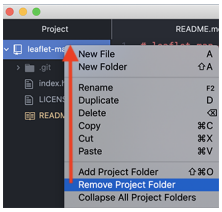


Figure 8.27: To clean up your Atom Editor workspace, right-click to *Remove Project Folder*.

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 8.28.

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.

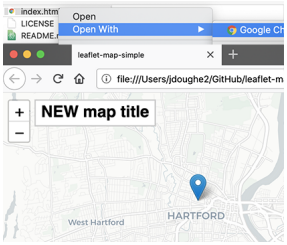


Figure 8.28: Right-click the index.html file on your local computer and open with a browser to check your edits.

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 8.29.

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 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.

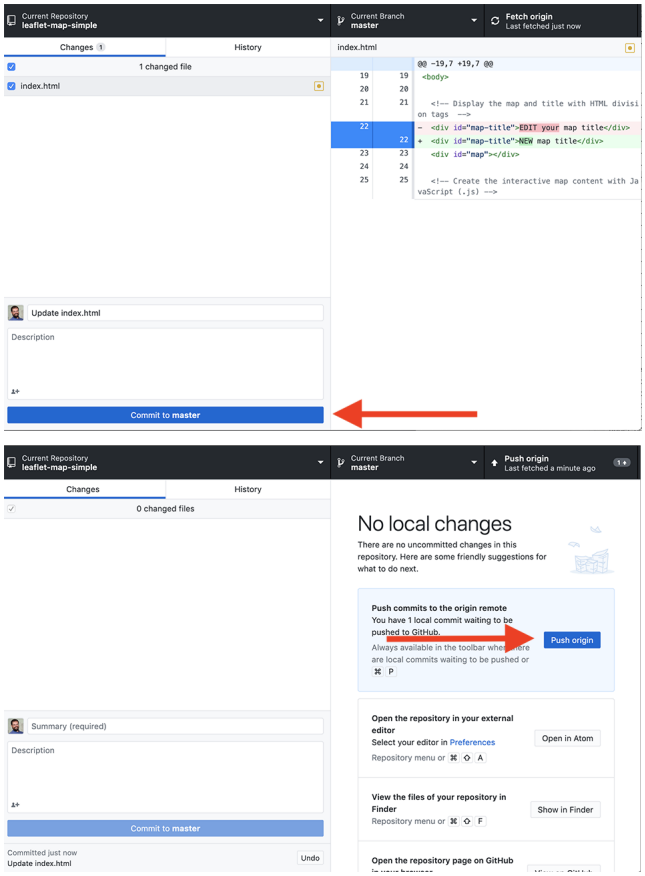


Figure 8.29: In this two-step process, click *Commit to Master*, then click *Push origin* to save and copy your edits from your local computer to your GitHub web account, as shown in this animated GIF.

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 8.30.

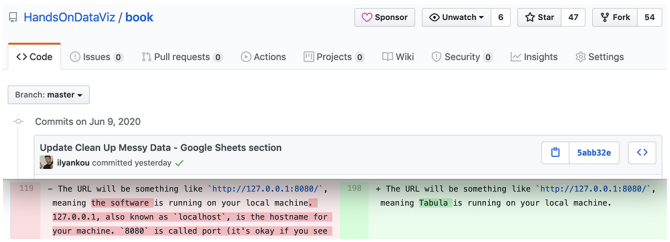


Figure 8.30: Drag-and-drop the file to the upload screen.

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 suggested code edits, which the owner can accept or reject. Third, GitHub allows collaborators to create different "branches" of a repo (the default is called "master") 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." To resolve this conflict and preserve everyone's work, you may need to use the Command Line Interface (CLI) version of GitHub, which means typing commands directly into the Terminal application on Mac or Windows. Many professional coders regularly work on the Command Line with GitHub, but this requires memorizing a list of commands and 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 code templates in Chapter 9 and the Leaflet map code templates in Chapter 10. Let's build on your brand-new coding skills to create more customized charts and maps in the next two chapters.

Chapter 9

Chart.js Code Templates

While beginners appreciate the drag-and-drop chart tools and tutorials described earlier in this book, such as Google Sheets and Tableau Public, more advanced users may wish to customize their visualizations, add more complex data, and control exactly how and where their work appears on the web. A more powerful and relatively easy-to-learn solution is to use code templates built with Chart.js https://www.chartjs.org/, an open-source library, which you can modify and host on GitHub, as described in this book.

Working with Chart.js

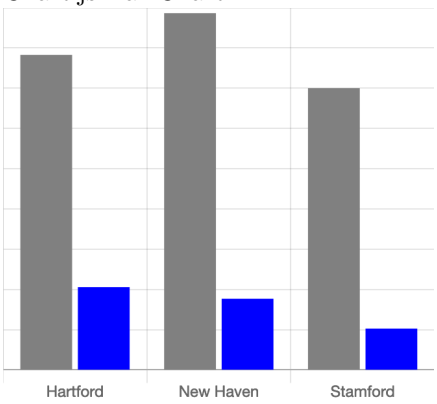
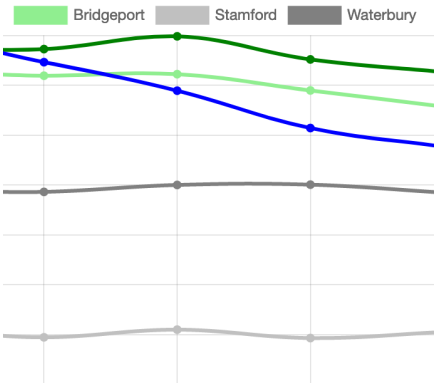
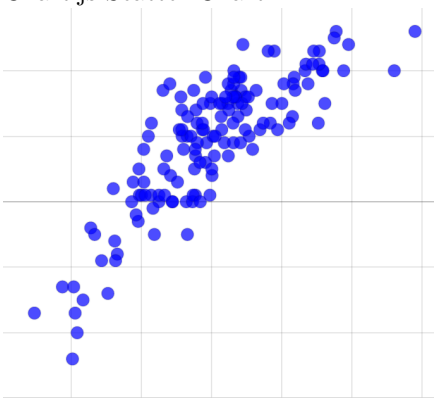
Pros

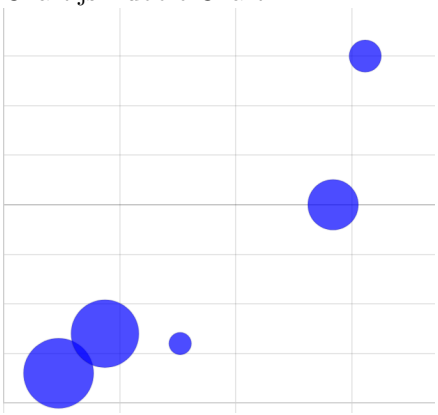
- Open-source code that is distributed under MIT license and is free for all and
- Easier for beginners to understand than more complex visualization code libraries such as D3.js

9.0.0.0.1 Cons

- Must host your own code repositories to publish to the web (with a free service such as GitHub Pages)

9.0.0.1 List of Chart.js templates

Templates	Best use and tutorial chapters
Chart.js Bar Chart 	Bar charts (vertical bar charts are often called column charts) can be used to compare categorical data. Template with tutorial: Bar Chart.js with CSV Data
Chart.js Line Chart 	Line charts are normally used to show trends (temporal data). Template with tutorial: Line Chart with CSV Data
Chart.js Scatter Chart 	Scatter charts (also scatterplots) are used to display data of 2 or more dimensions. Template with tutorial: Scatter Chart with CSV Data

Templates	Best use and tutorial chapters
Chart.js Bubble Chart 	Bubble charts are used to display data of 3 or more dimensions. Template with tutorial: Bubbble Chart with CSV Data

Inside the templates

The templates featured above 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) or pulled from Google Sheets

Learn more

- Chart.js Samples, https://www.chartjs.org/samples/latest/

Bar Chart.js with CSV Data

Bar charts (vertical bar charts are often called *column charts*) can be used to compare categorical data. The y-axis (or x-axis for horizontal bar chart) should

always start at 0.

Demo: https://handsondataviz.github.io/chartjs-templates/bar-chart/index.html

Source and instructions: https://github.com/handsondataviz/chartjs-templates/tree/master/bar-chart

Line Chart.js with CSV Data

Line charts are often used to show temporal data (trends). The x-axis often represents time intervals. Unlike column or bar charts, y-axes of line charts do not necessarily start at 0.

Demo: https://handsondataviz.github.io/chartjs-templates/line-chart/index.html

Source and instructions: https://github.com/handsondataviz/chartjs-templates/tree/master/line-chart

Scatter Chart.js with CSV Data

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

Demo: https://handsondataviz.github.io/chartjs-templates/scatter-chart/index.html

Source and instructions: https://github.com/handsondataviz/chartjs-templates/tree/master/scatter-chart

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.js with CSV Data

Bubble charts are similar to scatter plots. The size of each dot (marker) is used to represent an additional dimension.

In the demo below, the bubble chart shows the relationship between median household income (x-axis) and test performance (y-axis) in 6 school districts in Connecticut. The size of data point (marker) corresponds to the number of students enrolled in the school district: bigger circles represent larger school districts.

Demo: https://handsondataviz.github.io/chartjs-templates/bubble-chart/index.html

Source and instructions: https://github.com/handsondataviz/chartjs-templates/tree/master/bubble-chart

Tip: Use semi-transparent circles

Data points may obstruct each other. To avoid this, play with color transparency. For example, `rgba(160, 0, 0, 0.5)` is a semi-transparent red in RGBA color model. The `a` stands for `alpha`, and is a number between 0 and 1, where 1 is solid, and 0 is completely transparent. Using transparency, you will be able to see data points that are hidden behind bigger neighbors.

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.

Chapter 10

Leaflet Map Templates

While beginners appreciate the drag-and-drop map tools and tutorials described earlier in this book, Google My Maps and Carto, more advanced users may wish to customize their visualizations, add more complex data and interactivity, and control exactly how and where their work appears on the web. A more powerful and relatively easy-to-learn solution is to use code templates built with Leaflet https://leafletjs.com, an open-source library, which you can modify and host on GitHub, as described in this book.

Working with Leaflet

Pros:

- Open-source code, which anyone can freely use online, download, modify, or expand with plugins
- Easier for beginners to understand than some other map code libraries
- Compact code library (less than 40 KB) that runs on JavaScript in all modern web browsers

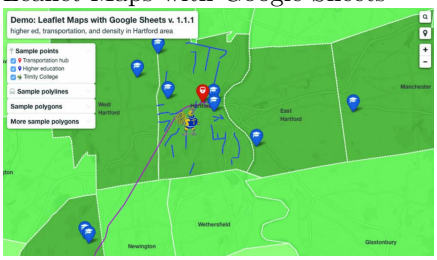
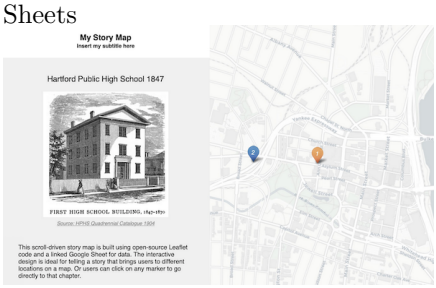
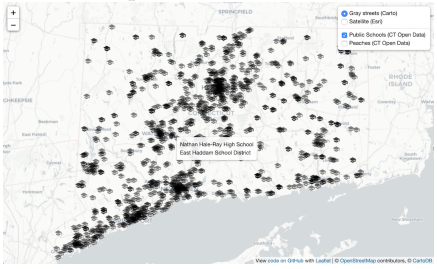
Cons:

- Must host your own code repositories to publish to the web (with a free service such as GitHub Pages)
- Must rely on open-source community of developers to maintain the core code or specific plugins

Leaflet Map Templates

TODO: add and clean up Leaflet Map CSV https://github.com/HandsOnDataViz/leaflet-map-csv to serve as a fuller tutorial for Leaflet Maps, and explain how

this will teach more principles of modifying Leaflet code. . . . then geocode and upload more data points:

Template	Best use and tutorial chapter
Leaflet Maps with Google Sheets 	Best to show points, polygons, polylines, or point table data. Upload your GeoJSON data and modify settings in linked Google Sheet (or CSV) and GitHub repo. Template with tutorial: Leaflet Maps with Google Sheets
Leaflet Storymaps with Google Sheets 	Create a scrolling narrative with points, text, images, and links. Template with tutorial: Leaflet Storymaps with Google Sheets
Leaflet Maps with Socrata API 	Create a Leaflet map powered by data from any Socrata data portal. Template with tutorial: Leaflet Maps with Socrata

Inside Leaflet code templates

The templates featured below vary from simple to complex, but all of them include three basic types of code:

- HTML: 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: to create the map and interactivity (example: script.js)

Also, these templates refer to other types of code:

- library: link to online instructions to complete routine tasks (examples: Leaflet, jQuery)
- basemap tiles: link to online background map (example: Carto Positron, a light-gray street map)
- data: content to appear on map, typically in CSV or GeoJSON format (examples: data.csv, data.geojson)

Fork and Edit Leaflet Map with CSV Data

TODO: REWRITE this to serve as a more advanced version (with repo leaflet-map-csv) than the prior chapter (with leaflet-map-simple)

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 Code Errors section of the appendix.

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

Video with step-by-step tutorial

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-simple
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:

```
<!-- 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}/{
    attribution: '&copy; <a href="http://www.openstreetmap.org/copyright">OpenStreetMap</a>, &cop
  }).addTo(map); // EDIT - insert or remove ".addTo(map)" before last semicolon to display by de
  // 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
  }); // 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}
  attribution: '&copy; <a href="http://www.openstreetmap.org/copyright">OpenStreetMap</a>, &copy;
}); // 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}.pn
  attribution: 'Map tiles by <a href="http://stamen.com">Stamen Design</a>, under <a hr
}).addTo(map); // EDIT - insert or remove ".addTo(map)" before last semicolon to displ
// 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/
  attribution: '&copy; <a href="http://www.openstreetmap.org/copyright">OpenStreetMap<,
}); // EDIT - insert or remove ".addTo(map)" before last semicolon to display by defau
 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}.pn
  attribution: 'Map tiles by <a href="http://stamen.com">Stamen Design</a>, under <a hr
}).addTo(map); // EDIT - insert or remove ".addTo(map)" before last semicolon to displ
 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 initi
 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 Google Sheets template

Question: If you have moved beyond simple drag-and-drop point map tool, such as Google My Maps tutorials in this book, and want to create point and/or polygon and/or polyline maps, where should you go?

Answer: Copy and customize our open-source template for Leaflet Maps with Google Sheets. Control the map options display data that you upload to your Google Sheet and GitHub repository. No coding skills required, other than pasting one line of code to link your map with your sheet. Requires two free accounts: Google and GitHub.

Video and list of features

- Best to show points, polygons, and/or polylines, with table of points in map view
- 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
- Geocode location data with US Census or Google, using script inside the Google Sheet
- Easy-to-modify data labels and map options in Google Sheet tabs or uploaded CSV files
- Upload your polygon and polyline GeoJSON files, and custom markers, to your GitHub repo
- Show multiple polygon layers, each with their own color legend and ranges (numerical or text)
- Responsive design resizes your maps to display inside most mobile devices

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

Part 1: Create and customize your map

In the first part of this tutorial, you will learn how to create your own copy of the Leaflet Maps with Google Sheets template, publish your Google Sheet, and paste its web address into your GitHub repo.

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

Part 2: Upload and display your map data

In the second part of this tutorial, you will learn how to geocode and customize your own point markers, and either hide or upload your own polygon and/or polyline GeoJSON data.

- E) Geocode locations and customize new markers in the Points tab
- F) Hide the polygon and polyline legends and default GeoJSON data
- G) Upload and display your own polygon GeoJSON data
- H) Upload and display your own polyline GeoJSON data
- I) Upload and display customized marker icons
- J) Optional: Save Google Sheets as CSV and upload to GitHub
- ** TO DO: second half video**

To solve problems, see Fix Common Code Errors 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-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 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-maps-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-maps-with-google-sheets
```

Figure 10.1: Screencast: Fork

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/1ZxvU8eGyuN9M8GxTU9acKVJv70iC3px_m3EVFsOHN9g

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.

Figure 10.2: Screencast: Share Google Sheet

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`).

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.

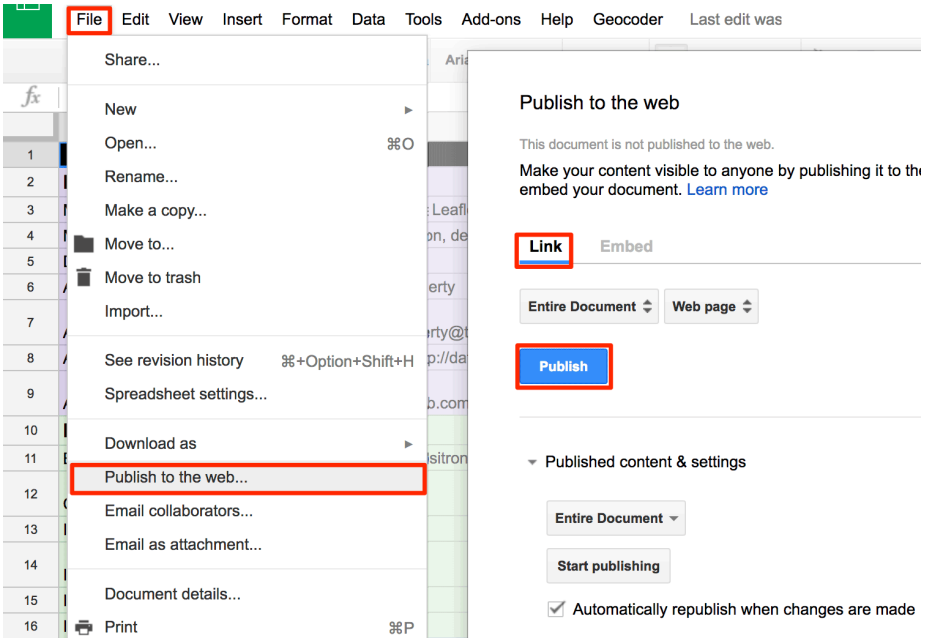


Figure 10.3: Screenshot: File > Publish the link to your Google Sheet

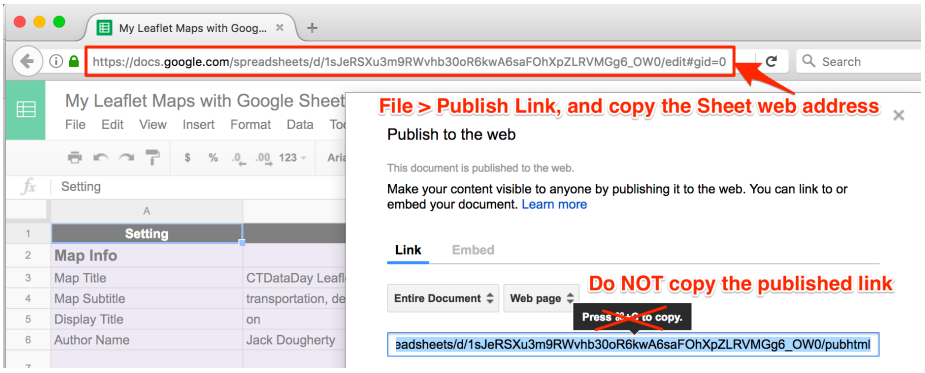


Figure 10.4: Screenshot: Copy the Google Sheet URL, not the Publish URL

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.

Figure 10.5: Screencast: Copy Google Sheet URL and paste into GitHub code

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) Think of a simple data story that involves at least four geocodeable locations, divided into at least two groups. If you need an example, use this sample table of "Famous Places in New York City":

Group	Name	Location
Landmark	Empire State Building	350 5th Ave, New York, NY 10118
Landmark	Metropolitan Museum of Art	1000 5th Ave, New York, NY 10028
Transit	Grand Central Terminal	89 E 42nd St, New York, NY 10017
Transit	Penn Station	159 West 33rd Street, New York, NY 10120

2) Enter your Group, Name, and Location data into new rows below the current data.

3) Go to the Font Awesome Icons site, http://fontawesome.io/icons, scroll down to Search Icons, find your desired icon code name, and insert this into the Marker Icon (column B) of your Points sheet. For example, search for and insert the icon code "train" or "building" to display markers with either of these symbols in your map. (To upload your own customized marker, see section H further below.)

4) In Marker Color (column C), use the drop-down menu to select a marker color.

5) In Icon Color (column D), insert a color word (example: white) or hex code (example: #fff) to color the icon symbol inside your marker. Recommended: use white icon colors with dark marker colors.

6) Leave Custom Size (column E) blank.

7) Optional:

- In Image (column G), insert the URL (preferably https://, not http://) of a small-to-medium sized image on the web
- In Description (column G), insert text and/or a web link enclosed with an HTML a href tag with target set to blank

8) Do NOT delete or rename any column headers. However, you have the option to add new column headers to display in your map table.

9) Geocode your new data inside your Google Sheet by dragging your cursor to select 6 columns of data: Location - Latitude - Longitude - Found - Quality - Source

10) 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.

F) Hide the polygon and polyline legends and default GeoJSON data

To show a simple point map, learn how to turn off and hide the polygon and polyline legend and default data that came with this template. (See how to add your own GeoJSON data in section G below.)

In your linked Google Sheet:

1) In the Options tab, Polyline Legend Position (cell B 35) – select Off to hide the legend

2) In the Polygons tab, Polygon Legend Position (cell B 4) – select Off to hide the legend

3) In the Polygons tab, Polygon GeoJSON URL (cell B 6) – delete contents to remove polygons

4) Go to the next tab, named Polygons1, in its drop-down menu, select Delete to remove sheet

5) In the Polylines tab, delete the entire row (rows 2 and 3) to remove the existing lines

Go to the browser tab with your new map, and refresh the page to see your changes.

Optional:

- in the Options tab, Display Table (cell B 29), turn off to hide the table in your map
- or modify the list of item in Table Columns (cell B 30) to change the display in your table

G) Upload and display your own polygon GeoJSON data

1) Prepare your polygon file in GeoJSON format. View or modify the GeoJSON file properties (such as name, data fields, etc.) with one of these tools:

- GeoJSON.io, http://geojson.io – Drag-and-drop your file, and select the Table tab to view or rename properties. See GeoJSON.io tutorial in this book, OR
- MapShaper, http://mapshaper.org – Drag-and-drop your file. To edit, see MapShaper tutorial in this book

2) In your GitHub repo, click to open the Geometry subfolder, then click Upload Files, drag-and-drop your geojson file, and Commit changes

** TO DO ** - turn on settings that you turned off in step F above

3) In your linked Google sheet, go to Polygons tab to adjust these settings:

- change Polygon GeoJSON URL (cell B 6) to match the pathname of the file you uploaded above
- change Polygon GeoJSON Name (cell B 5) to the title to be displayed for this polygon layer
- change Polygon Legend Title (cell B 3) for the title in the polygon legend box

4) To adjust the polygon legend colors and range, see the Polygon Data and Color Settings sections of the Polygon tab in Google Sheets.

5) The code supports multiple polygon layers, which you can add (or delete) by duplicating the Polygons tab. Name them Polygons1, Polygons2, etc.

- TO DO *
- Explain: To use both the automatic ColorBrewer Palette and manual colors, insert blanks (goes to automatic palette above), separated by semicolons.

H) Upload and display your own polyline GeoJSON data

Follow similar steps as described in the Polygon section above, but adjust settings in the Polylines tab of your linked Google Sheet.

I) Upload and display customized marker icons

** TO DO **

J) Optional: Save Google Sheets as CSV and upload to GitHub

If desired, you can save your table data with your code, rather than in a separate Google Sheet. Go to each Sheet tab and File > Save As in CSV format, with these file names:

- options.csv
- points.csv
- polygons.csv (also: polygons1.csv, polygons2.csv, etc.)
- polylines.csv
- notes.csv (or .txt)

Upload these files into the main level of your GitHub code repository, where the template will process them automatically.

Learn more

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

Leaflet Storymaps with Google Sheets and Scrolling Narrative

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
- 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
- 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 Fix Common Code Errors 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_0JafWZF4KEejkdDNqfuZWUk3SlNlQ6MjlRFM/

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`).

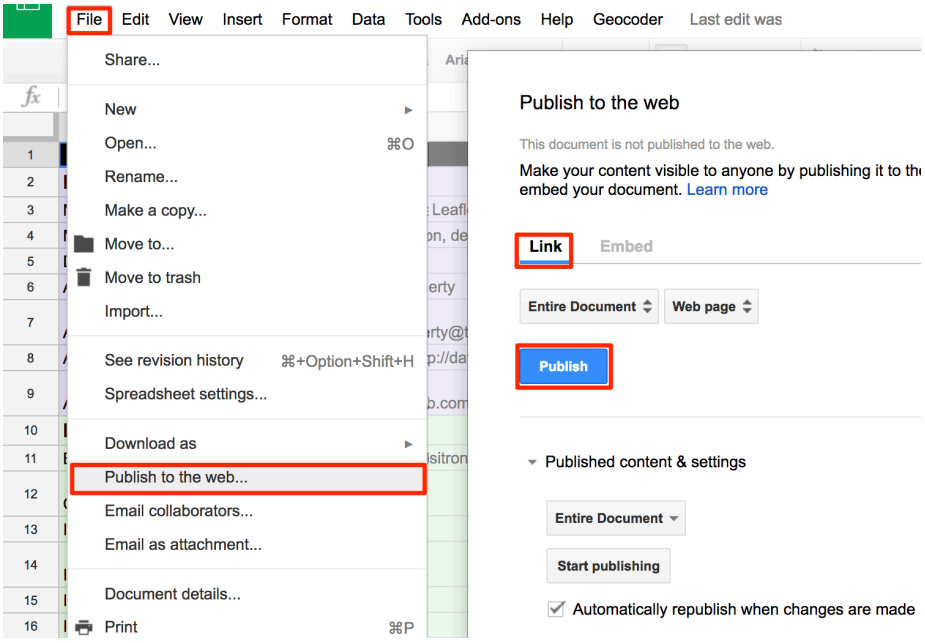


Figure 10.6: Screenshot: File > Publish the link to your Google Sheet

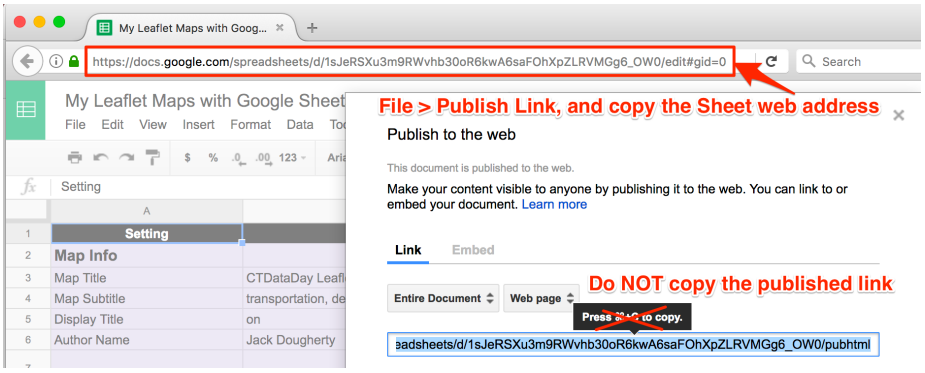


Figure 10.7: Screenshot: Copy the Google Sheet URL, not the Publish URL

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

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)

Learn more

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

Leaflet Maps with Socrata API Open Data

TODO: Decide whether to keep or not, and if so, add intro

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.

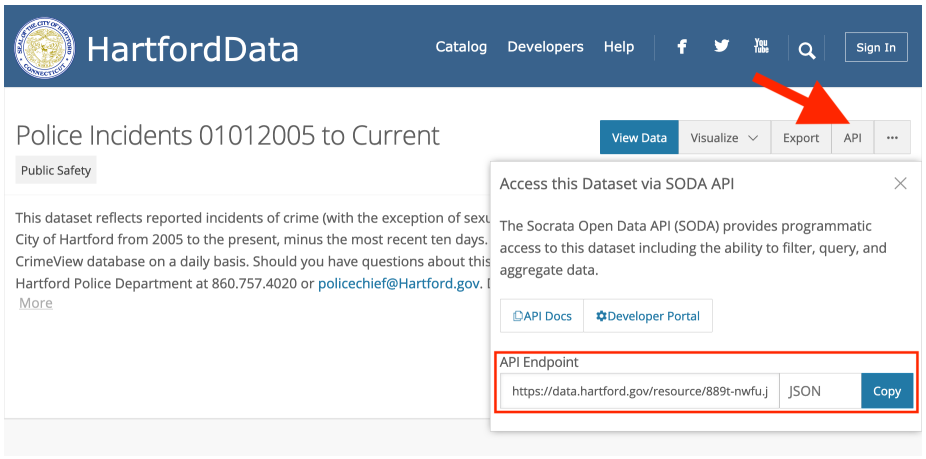


Figure 10.8: Police Incidents dataset on Hartford Open Data portal

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).

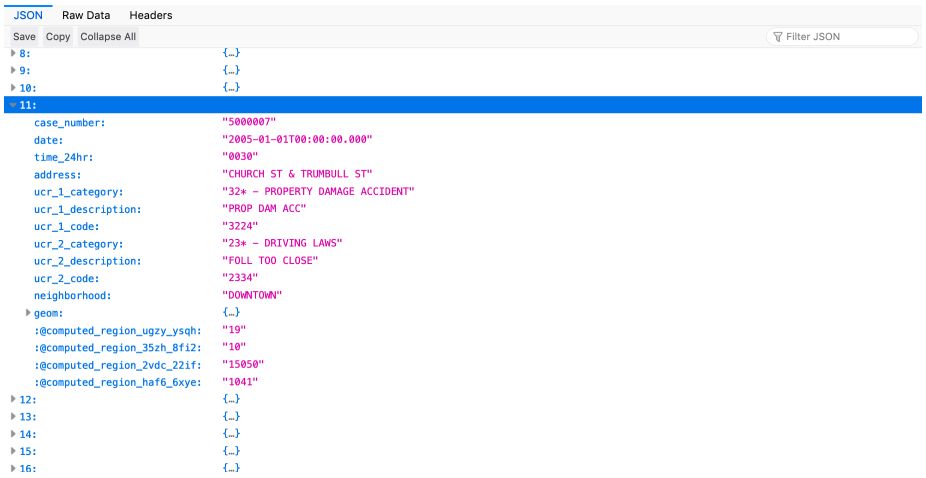


Figure 10.9: Formatted JSON example in Firefox

If your browser does not support JSON view, you will see the raw JSON stream

only, like the one shown below.

Figure 10.10: Unformatted JSON example in Firefox

Test if this Socrata endpoint supports GeoJSON format by changing the extention 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.

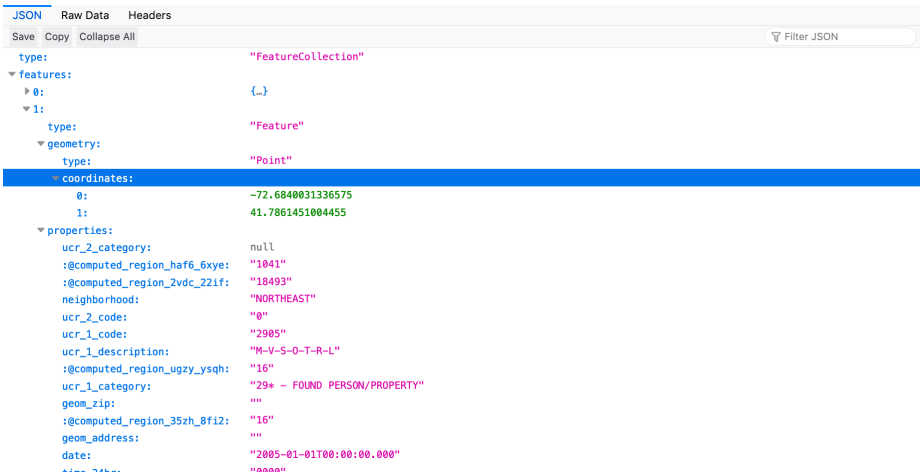


Figure 10.11: Formatted GeoJSON example in Firefox

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-Commoditites-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/

Pull Open Data into Leaflet Map with 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

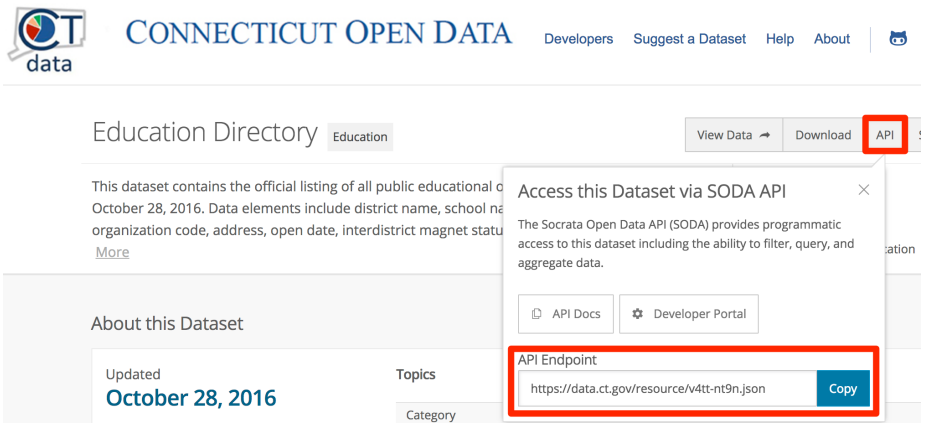


Figure 10.12: Screenshot: Sample API endpoint in Socrata open data repo

3) Inside the open data repo, look for an API button and copy the endpoint.

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:

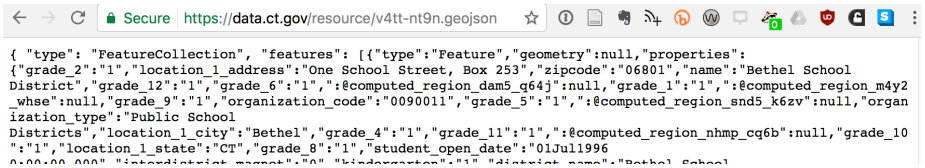


Figure 10.13: Screenshot: API endpoint with .geojson suffix in Chrome browser

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=QVVY3I72SVPbxBYlTM8fA7eet"
  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); // n
      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 Thematic Polygon Map with Clickable Info Window template

TODO: Decide whether to keep or not

Try it

View demo in new page

- https://handsondataviz.github.io/leaflet-map-polygon-click/

To Do

- Insert internal references to prior steps in this book. See the Edit and Host Code Templates section in this book.
- Requires a free GitHub account to host your own version on the web.

Create Your Own: Fork a copy of the code template on GitHub

- https://github.com/handsondataviz/leaflet-map-polygon-click
- Remember, if you have already forked one copy, go to your GitHub repository Settings to rename it, or create a new GitHub repo and use GitHub Desktop to upload template Files

Obtain a polygon boundary map in GeoJSON format

- Find open data repositories to download maps in geojson and other formats
- If map is in shapefile or KML or other format, convert with http://geojson.io or http://mapshaper.org
- Import polygon map into http://mapshaper.org. In this example, map filename is: ct-towns-simple.geojson
  - See tutorial on Mapshaper.org to delete unwanted data columns or simplify file size
  - Export as CSV to create generic spreadsheet of polygon names. In this example, column header is "town"

Prepare your spreadsheet data and join with the polygon map

- Open CSV with any spreadsheet tool to view data column of polygon names.
- Download or prepare your new spreadsheet data in rows to match polygon names.
- Insert columns of data into the CSV sheet. Use VLOOKUP function if needed.
- Save CSV with new name. In this example: ct-towns.csv
- Import ct-towns.csv as second layer into MapShaper.org.
- Use the drop-down to select the polygon map (ct-towns-simple.geojson) as the active, displayed layer.
- Click the Console and enter command to join the CSV table to the GeoJSON polygon, where the matching data columns are both named "town"

```
-join ct-towns.csv keys=town,town
```

- Export the newly joined map with a new filename in GeoJSON format
- Change the file suffix from .json to .geojson to avoid confusion. The new joined map data file is now named: ct-towns-density.geojson

Upload your map data and edit template in your GitHub repo

- The GitHub repo you created in the first step contains these files:
  - ct-towns-density-2010.csv (the spreadsheet joined into the polygon map)
  - ct-towns-density.geojson (the joined map data file)
  - index.html (the primary web page)
  - script.js (code to operate the map, to be modified below)
  - style.css (code that styles the map)
  - README.md (edit to insert a link to your own version)
  - LICENSE (terms of use for this free and open-source code)
- Upload your own map data geojson file
- Recommended: upload your own CSV spreadsheet file to
- In the script.js file, look for code comments labeled "Edit" to change references to geojson map data and its column headers, and also colors and ranges for the polygons and legend
- In GitHub, go to Branches and delete the existing "gh-pages" branch
- In GitHub, go to drop-down menu for Master branch, and type "gh-pages" to create new branch
- Content in the gh-pages branch will be hosted on the live web
- Edit the README.md link to point to your own gh-pages branch, in this format: `https://USERNAME.github.io/REPO-NAME/`

Leaflet Thematic Polygon Map with Hover Info Window template

TODO: Decide whether to keep or not

Try it

View demo in new page

- https://handsondataviz.github.io/leaflet-map-polygon-hover/

To Do

- Insert internal references to prior steps in this book. See the Edit and Host Code Templates section in this book.
- Requires a free GitHub account to host your own version on the web.

Create Your Own: Fork a copy of the code template on GitHub

- https://github.com/handsondataviz/leaflet-map-polygon-hover/
- Remember, if you have already forked one copy, go to your GitHub repository Settings to rename it, or create a new GitHub repo and use GitHub Desktop to upload template Files

TO DO - describe all steps, which are similar to click version

Leaflet Thematic Polygon Map with Multi-Year Tabs template

TODO: decide whether to keep or not

Try it

View demo in new page

- https://handsondataviz.github.io/leaflet-map-polygon-tabs/

** To Do **

- Insert internal references to prior steps in this book. See the Edit and Host Code Templates section in this book.
- Requires a free GitHub account to host your own version on the web.
- describe all steps, which are similar to the prior chapter

Create Your Own: Fork a copy of the code template on GitHub

- https://github.com/handsondataviz/leaflet-map-polygon-tabs/
- Remember, if you have already forked one copy, go to your GitHub repository Settings to rename it, or create a new GitHub repo and use GitHub Desktop to upload template Files

Chapter 11

Transform Your Map Data

Interactive web maps are made up of different layers, such as background basemaps, colored or shaded polygons, and/or colored point markers. This chapter describes how to transform your data into layers that you can upload into online map tools and templates. Specifically, you will learn how to:

- Geocode locations into coordinates with US Census or Google
- Pivot address-level point data into polygon data
- Normalize data to create more meaningful polygon maps
- Convert map data with GeoJSON.io or Mapshaper.org
- Join spreadsheets and polygon boundaries with MapShaper.org

Enroll in our free online course **TO DO add link**, which introduces these topics in the brief video below, and offers more exercises and opportunities to interact with instructors and other learners.

Geocode Locations into Coordinates with US Census or Google

Many free map tools geocode locations by placing them on a map, such as Google My Maps tutorials in this book. But those tools typically do not allow you to easily extract the latitude-longitude coordinates for each point.

We created two free Google Sheets Geocoder scripts that have several advantages:

- convert locations (Hartford CT) or addresses (300 Summit St, Hartford CT) into latitude-longitude coordinates (41.748, -72.692) inside your Google Sheet

- show the location found in the geocoding database, and match quality, to review your results
- convert US addresses into US Census geography, such as census tracts, block groups, and blocks

As with any geocoding service, accuracy is not guaranteed. Inspect your results in the Found and Quality columns.

Google Sheets Geocoder: US Census or Google

- Geocode locations into latitude, longitude, with source and match quality, inside a Google Sheet
- Go to Google Sheet template, sign in to your account, and File > Make a Copy to your Google Drive https://docs.google.com/spreadsheets/d/1XvtkzuVyQ_7Ud47ypDJ4KOmz_5lOpC9sqeEDBbJ5Pbg/edit#gid=0
- Insert locations, select 6 columns, and select Geocoder menu: US Census or Google (limit 1000 daily per user)
- Google Sheets script will ask for permission to run the first time
- Note: The Leaflet Maps with Google Sheets template in this book includes this Geocoder script.

Figure 11.1: Screencast: Google Sheets Geocoder: US Census or Google

Google Sheets Geocoder: US Census Geographies

- Geocode US addresses into latitude, longitude, GeoID, census tract, inside a Google Sheet
- Go to Google Sheet template, sign in to your account, and File > Make a Copy to your Google Drive https://docs.google.com/spreadsheets/d/1x_E9KwZ88c_kZvhZ13IF7BNwYKTJFxbfDu77sU1vn5w/edit#gid=0

- Insert locations, select 8 columns, and select Geocoder menu: US Census 2010 Geographies
- Google Sheets script will ask for permission to run the first time

Figure 11.2: Screencast: Google Sheets Geocoder: US Census Geographies

About US Census 15-character GeoID

- Make sure that column G is formatted as text (to preserve leading zeros), not number
- Break down a sample GeoID: 090035245022001
  - state = 09
  - county = 003
  - tract = 524502 = 5245.02
  - block group = 2
  - block = 001

How it works The Google Sheet Geocoder runs from a script insert in the Google Sheet, which calls one of two free geocoding services:

- US Census Geocoder https://geocoding.geo.census.gov/geocoder. See more detailed documentation at http://www.census.gov/geo/maps-data/data/geocoder.html
- Geocode with Google Apps: The Maps Service of Google Apps allows users to geocode street addresses without using the Google Maps API, with a limit of 1,000 searches daily per user, https://developers.google.com/apps-script/reference/maps/geocoder

How to insert the Geocoder Script into any Google Sheet If you do not wish to File > Make a Copy of the Google Sheet templates above, you can insert the open-source Geocoder Scripts into your own Google Sheet:

- Go to Google Sheets Geocoder repo on GitHub
- Sign in to your Google Sheets, then select Tools > Script Editor
- File > Create New Script File
- Open and copy a script (such as geocoder-census-google.gs) and paste into your Script Editor
- Save and rename to geocoder-census-google.gs
- Refresh your Google Sheet and look for new Geocoder menu

TODO

- Also describe and link back how to split columns to form multi-columns addresses
- also describe and link back to how to unify columns to form a one-column address
- add this https://developers.google.com/maps/faq#geocoder_queryformat

See also: Batch upload to US Census

- Available at US Census Geocoder https://geocoding.geo.census.gov/geocoder/
- Upload CSV table with up to 1000 rows for faster processing, in this format, WITHOUT column headers:

AnyID | Street | City | State | Zip |
:— | :— | :— | :—- | : — |
1 | 300 Summit St | Hartford | CT | 06106 |

- Find Locations using > Address Batch (returns latitude, longitude coordinates)
- Find Geographies using > Address Batch (returns lat, lng, census geographies)
- Limitations:
  - Inputs and outputs have no column headers, which may confuse novices
  - Large batches may be delayed a few minutes during peak time periods
  - Unmatched addresses need to be manually corrected and re-submitted

Try it: Batch Upload to US Census

1) Right-click and Save this CSV file to your computer: sample-addresses-50. CSV means comma-separated values, a generic spreadsheet format that most data tools can easily open.

2) Use any spreadsheet tool to organize your address data into five columns: any ID number, street, city, state, zip code. **Remove all column headers**.

	A	B	C	D	E
1	1	300 Summit St	Hartford	CT	06106
2	2	960 Main St	Hartford	CT	
3	3	133 Allen Pl, Apt 2	Hartford	CT	06106

Hints:

- If your data lacks ID numbers, quickly create a column of consecutive numbers, as shown in this book.
- If your address data includes apartment numbers, leave them in.
- Only the ID and address fields are required. City, state, and zip code may be blank if you lack any of this information, but fewer matches will be exact.
- If your address data is combined into one cell, such as: 300 Summit St, Hartford, CT 06106
  - then try to clean your data with the split column method in this book.
- If you need to temporarily move other non-address data columns into a second spreadsheet, remember to paste the column of ID numbers into the second sheet. After geocoding, sort both sheets by the ID column, then paste to rematch the data.

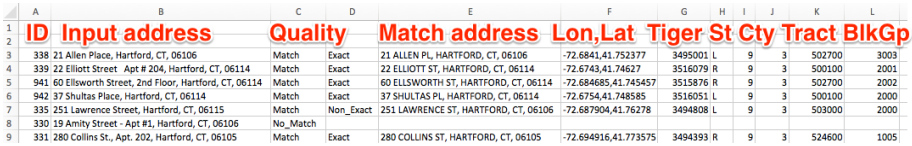
3) Save the file in CSV generic spreadsheet format, in batches of no more than 1,0000 rows per file. Learn more about saving in CSV format in this book.

4) Go to US Census Geocoder (https://www.census.gov/geo/maps-data/data/geocoder.html)

5) Select the Find Geographies Using... Address Batch button for maximum results, including lat-long coordinates and census geography (tracts and block groups). *If census geography is not needed, select Find Locations Using... Address Batch.*

6) Click the Choose button to upload your CSV file. Use the default benchmark and vintage settings for the most current data. Click the Get Results button, and be patient if using the service during busy weekday hours.

7) Census Geocoder will download the results through your web browser in a file named: GeocodeResults.csv. Since these results do not contain column headers, use the screenshot below for guidance, or read the Census Geocoder documentation for more details.



ID	Input address	Quality		Match address	Lon,Lat	Tiger	St	Cty	Tract	BlkGp
338	21 Allen Place, Hartford, CT, 06106	Match	Exact	21 ALLEN PL, HARTFORD, CT, 06106	-72.6841,41.752377	3495001	L	9	502700	3003
339	22 Elliott Street Apt # 204, Hartford, CT, 06114	Match	Exact	22 ELLIOTT ST, HARTFORD, CT, 06114	-72.6743,41.74627	3516079	R	9	500100	2001
941	60 Ellsworth Street, 2nd Floor, Hartford, CT, 06114	Match	Exact	60 ELLSWORTH ST, HARTFORD, CT, 06114	-72.684685,41.745457	3515876	R	9	502700	2002
942	37 Shultas Place, Hartford, CT, 06114	Match	Exact	37 SHULTAS PL, HARTFORD, CT, 06114	-72.6754,41.748585	3516051	L	9	500100	2000
335	251 Lawrence Street, Hartford, CT, 06115	Match	Non_Exact	251 LAWRENCE ST, HARTFORD, CT, 06106	-72.687904,41.76278	3494808	L	9	503000	2000
330	19 Amity Street - Apt #1, Hartford, CT, 06106	No_Match								
331	280 Collins St., Apt. 202, Hartford, CT, 06105	Match	Exact	280 COLLINS ST, HARTFORD, CT, 06105	-72.694916,41.773575	3494393	R	9	3	524600	1005

8) Use a spreadsheet tool to open the CSV file. Sort results by the match quality (columns C and D), with these entries: match exact, match non-exact, tie, no-match.

9) For results without an exact match, check the address for typos, and try to re-geocode in a separate CSV file. The US Census Geocoder tool is very good, but not perfect. For a few rows of hard-to-match data, use a different geocoding tool, such as the Google Maps > What's Here feature described at the top of this page, to look up individual addresses and coordinates.

Learn more

- Aggregate individual rows of data into groups by census area with pivot tables.
- Download census data by tract or block group, and use the VLOOKUP formula to join or merge this rows of data that you have geocoded by census tract or block group.

Pivot Address-Level Point Data into Polygon Data

Problem: If I begin with address-level point data, how can I transform this into polygon map data?

One solution: In any spreadsheet, split your address data into separate columns (such as Street, City, State), then create a pivot table to aggregate rows into groups (such as the number of addresses in each City or State).

If your location data is combined into one column (example: 300 Summit St, Hartford CT), then see the Spreadsheets: Split Data Columns tutorial in this book.

Here's an example using a long list of US hospitals from the Medicare open data repository, which is already split into separate columns: https://data.medicare.gov/Hospital-Compare/Hospital-General-Information/xubh-q36u

1) Open the link above, see columns of data (Address, City, State, etc.), and click the blue Export button to download in the CSV generic spreadsheet format.

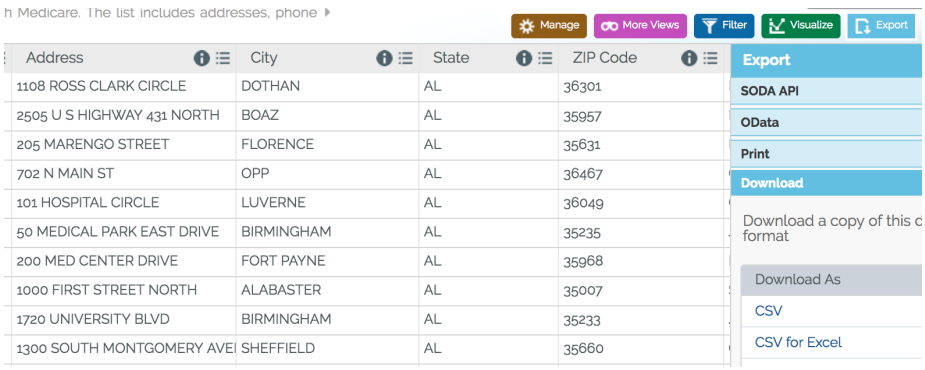


Figure 11.3: Screenshot: Export US Hospital data into CSV format

2) Open the file with any spreadsheet tool, and create a pivot table to count up the number of hospitals in each state. For help, see the Pivot Table tutorial in this book.

3) Now you can copy and paste the pivot table raw data of hospitals by US states. See the Normalize Data tutorial and also the Edit and Join Spreadsheet with Polygon Map using Mapshaper tutorial in this book.

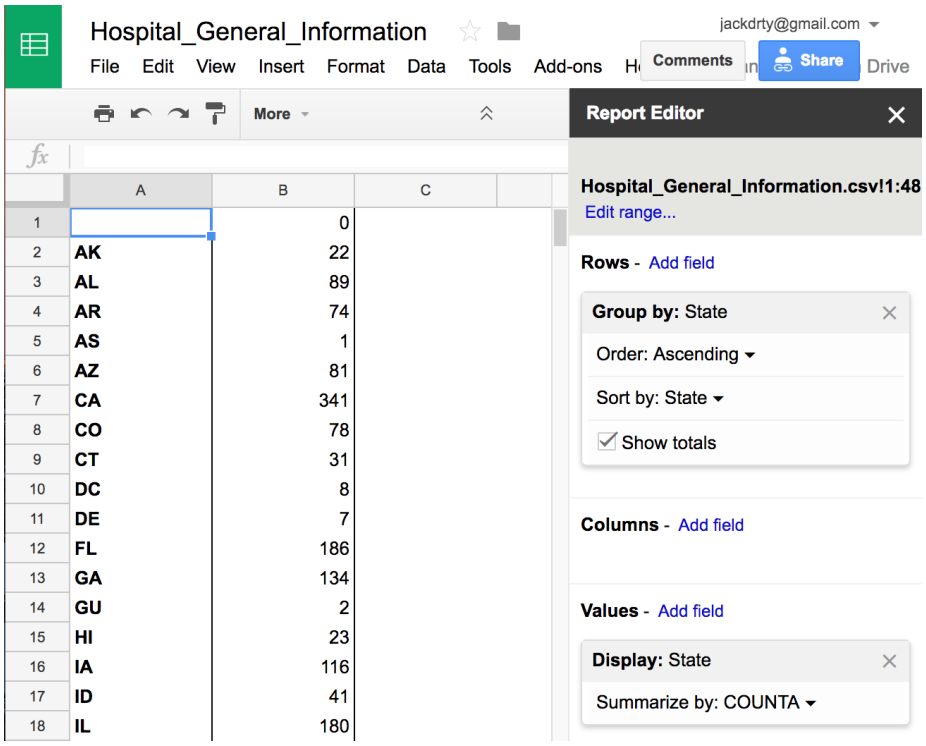


Figure 11.4: Screenshot: Pivot Table of US Hospitals by State

Other Solutions

- use the Google Sheets Geocoder: US Census Geographies tutorial in this book to convert addresses into census tracts, etc., and then pivot
- do a polygons-to-points spatial join with Mapshaper.org ** TO DO **

Normalize Data to Create Meaningful Polygon Maps

When preparing polygon maps, normalize your data to create more meaningful comparisons. Learn the difference between:

- **Raw data:** absolute values, such as the population of each US state (example: Connecticut population in 2015 = 3,590,886 people)
- **Normalized data:** represented on a standard scale (also known as standardized data), such as the population density of each US state (example: Connecticut 2015 population density = 3,590,886 people / 4,482 square miles = 742 people per square mile, equivalent to 1,922 people per square kilometer)

The difference between raw versus normalized data matters, especially in polygon maps. For example, the US states of Connecticut and Iowa have similar populations of about 3 million people each. But the rural midwestern state of Iowa has a much larger land area of over 55,000 square miles, while the more urbanized eastern state of Connecticut has a smaller land area of around 4,000 square miles. We can display all of this data in a table (as show below), but when making a polygon map, it makes most sense to show a normalized value, such as population density.

US State	Population 2015	Land Area (in square miles)	Density (pop per square mile)
Iowa	3,123,899	55,857	56
Connecticut	3,590,886	4,842	741

But raw data still matters, too. Although normalized data allows for easier comparisons across regions of different size, it can hide very low raw data values. For example, imagine two city neighborhoods with equally high unemployment rates of 20%, a normalized value. But if one neighborhood has a labor market population of 5,000 people while the other has only 500, the actual number of unemployed people in the second neighborhood is much smaller, as shown in the table below.

Neighborhood	Labor Market Population	Unemployment Rate	Actual Unemployed People
First	5,000	20%	1,000
Second	500	20%	100

Different ways to normalize data

After you understand the basic concept, also think about different ways to normalize the same data. Your method depends on the type of data story you wish to emphasize. Look at the table excerpt below on US population and land area by state in 2015:

	A	B	C
1	name	pop2015	land-area-sq-mi
2	Alabama	4858979	50645
3	Alaska	738432	570641
4	Arizona	6828065	113594
[redacted]			
52	Wyoming	586107	97093
53	Puerto Rico	3474182	3424
54	TOTAL	324893003	3535329

Figure 11.5: Screenshot: US population and land area

There are at least two acceptable ways to normalize this raw data:

- Normalized by area: Population per square mile in each state (calculate = pop / square miles)
- Normalized by total: Percent of total US population in each state (calculate = state pop / total US pop)

For example:

US State	Population 2015	Land Area (sq. mi)	Density (per square mile)
Connecticut	3,590,886	4,842	741

Convert to GeoJSON format

When you find map data, it may be stored in one of these common data formats below:

GeoJSON

GeoJSON is newer, popular open format for map data, and works across many tools, so is our top recommendation in this book. GeoJSON files can be used with Leaflet map code, Google Maps JS API code, Carto map tools, and more. Also, your GitHub repository will automatically display any GeoJSON files in a map view.

GeoJSON data must follow a structured format, but the file name may end with either `.geojson` or `.json`. The GeoJSON structured format orders coordinates in *longitude-latitude* format, the same as X-Y coordinates in mathematics. This is the opposite of Google Maps and several other web map tools, which order points in *latitude-longitude* format. For example, Hartford Connecticut is located at (-72.67, 41.76) in GeoJSON, but (41.76, -72.67) in Google Maps.

Shapefiles

The shapefile format was created in the 1990s by ESRI, the company that developed ArcGIS software. Shapefiles typically appear as a folder of subfiles with suffixes such as `.shp`, `.shx`, `.dbf`, and others. 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, this book recommends converting shapefiles into one of the more friendlier formats below.

Keyhole Markup Language (or KML)

The KML format rose in popularity during the late 2000s. Google Earth, a free and user-friendly tool, allowed many people to view and edit geographic data. KML files are commonly used in the Google Fusion Tables maps described in this book. Sometimes `.kml` files are distributed in a compressed `.kmz` format. See the chapter on converting from KMZ to KML format in this book.

GeoJson.io to Convert, Edit, and Create Map Data

TODO:

- rewrite into tool review and tutorial format
- place polygon conversion at top and specify import-export formats

Go to http://geojson.io to explore this open-source web tool to convert, edit, and create GeoJSON map data. The tool was originally developed by Tom MacWright, and is supported by Mapbox.com.

Convert a CSV spreadsheet of point data into GeoJSON

Use any spreadsheet tool and prepare a list of coordinate points (known as features). You must include column headers **lat** and **lon**, or a fuller spelling, such as *latitude* and *longitude*. The order of the columns does not matter. Also, you can add more headers to identify each point (example: name) and include more details (known as the properties of the features).

name	lat	lon	optional	optional2
Hartford	41.76	-72.67	city	https://en.wikipedia.org/wiki/Hartford,_Connecticut
Bloomfield	41.85	-72.73	suburb	https://en.wikipedia.org/wiki/Bloomfield,_Connecticut
West Hartford	41.76	-72.75	suburb	https://en.wikipedia.org/wiki/West_Hartford,_Connecticut

Save your spreadsheet in generic CSV format. *Hint:* see Save Spreadsheet as CSV chapter in this book.

Try it! Click this link and Save to download this sample file to your computer: name-lat-lon-info in CSV format. CSV means comma-separated values, a generic spreadsheet format that most data tools can easily open.

Drag the CSV file into the GeoJSON.io map window. Flip between the JSON and Table tabs to view or edit the data.

Figure 11.6: Screencast: GeoJson.io

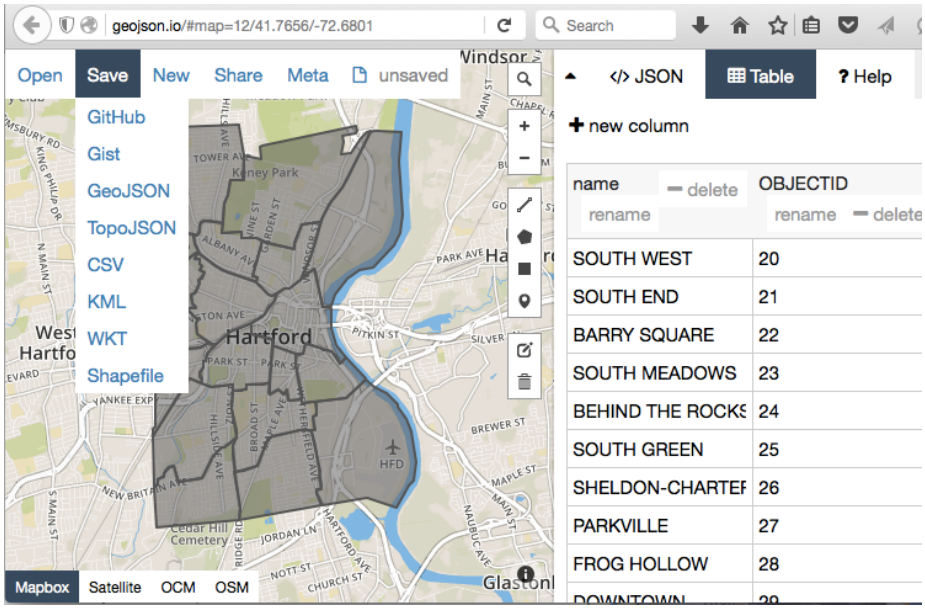
Select the Save menu and export into GeoJSON format.

Optional: Login to GeoJSON.io with your GitHub account and save directly to your repository.

Convert Shapefile or KML polygons into GeoJSON

Polygon boundary data is often shared as ArcGIS Shapefiles (.shp) or Keyhole Markup Language (.kml) files. Drag any of these (and other) files into the http://GeoJSON.io map window. Flip between the JSON and Table tabs to view or edit the data.

Select the Save menu and export into GeoJSON format.



Create GeoJSON data with drawing tools

Use the http://GeoJSON.io drawing tools to create points, polygons, and polylines. Flip between the JSON and Table tabs to view or edit the data.

Learn more about GeoJSON.io

Read about more advanced features and view the code at https://github.com/mapbox/geojson.io

MapShaper.org to Convert, Edit, and Join Data

TODO:

- rewrite into tool review/tutorial format

- put conversion at top and clarify import-export formats
- recommended browsers: Firefox or Chrome
- Mac users: go to Finder > Preferences > Advanced > turn on Show file extensions

MapShaper (http://MapShaper.org) is another versatile open-source mapping tool, developed and maintained by Matthew Bloch on GitHub. Using the web interface, users can:

- Import and export map layers in multiple formats: Shapefile, GeoJSON, CSV, and more
- Simplify (or smooth out) geographic details to reduce map file size
- Edit geography with powerful commands (dissolve, clip, join files, etc.)

This free and easy-to-learn MapShaper.org web tool has replaced *many* of my 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.org to be a quick alternative for some common time-consuming tasks.

The examples below focus on polygon boundary data to illustrate common map editing tasks. But MapShaper.org also works with other data layers, such as tables, points, and lines.

Import and convert map boundary files

Try it! Right-click the link and Save to download this sample file to your computer: ct-towns in GeoJSON format. If you accidentally open a page of GeoJSON code in your browser, select File > Save Page As to download to your computer.

1. Drag-and-drop any map layer into the http://MapShaper.org browser window.

- Import GeoJSON (.geojson or .json), TopoJSON, CSV, or Shapefile formats
- For Shapefiles, import the .shp (features), .dbf (attribute data), and .prj (projection) files. Reminder: the WGS84 projection is most portable across multiple platforms.
- KML/KMZ files are not compatible. To convert these into a format that Mapshaper can import, see the Convert KMZ to KML and Geojson.io chapters in this book.

2. Click the Export button and select your preferred format:

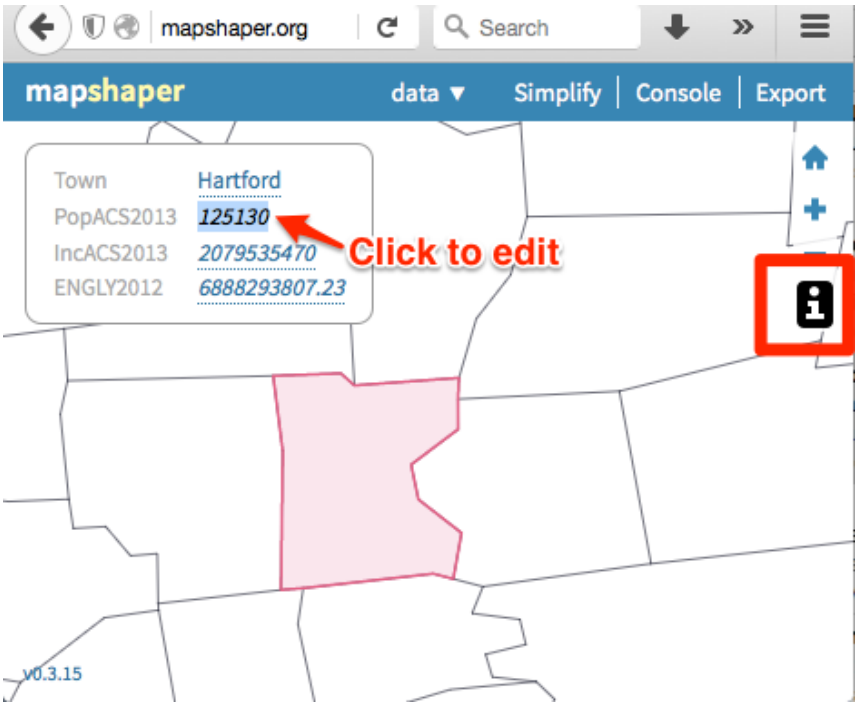
- Shapefile (best for ArcGIS/QGIS software)
- GeoJSON (best for Leaflet and GitHub tools in this book)
- TopoJSON (similar to GeoJSON, with topographical data)
- SVG (Scalable Vector Graphics, for print or online)
- CSV (Comma Separated Values, generic spreadsheet format)

Figure 11.7: Screencast: Mapshaper convert

Edit data for specific polygons

To edit data for any polygon in MapShaper.org:

- Click the "i" information button
- Select the polygon
- Click inside its pop-up info window to directly edit the data



Simplify map boundaries to reduce file size

If your data visualization project displays a zoomed-out state or national or world map, small geographic details that are invisible at these zoom levels may not be necessary. Consider using the Simplify command to reduce the file size, which may help your interactive web map to load faster for web visitors. The example below began with a detailed map of Connecticut town boundaries (1:100,000 scale) at 2MB, which MapShaper simplified – without visibly sacrificing details at the statewide zoom level – to a reduced size of about 200KB.

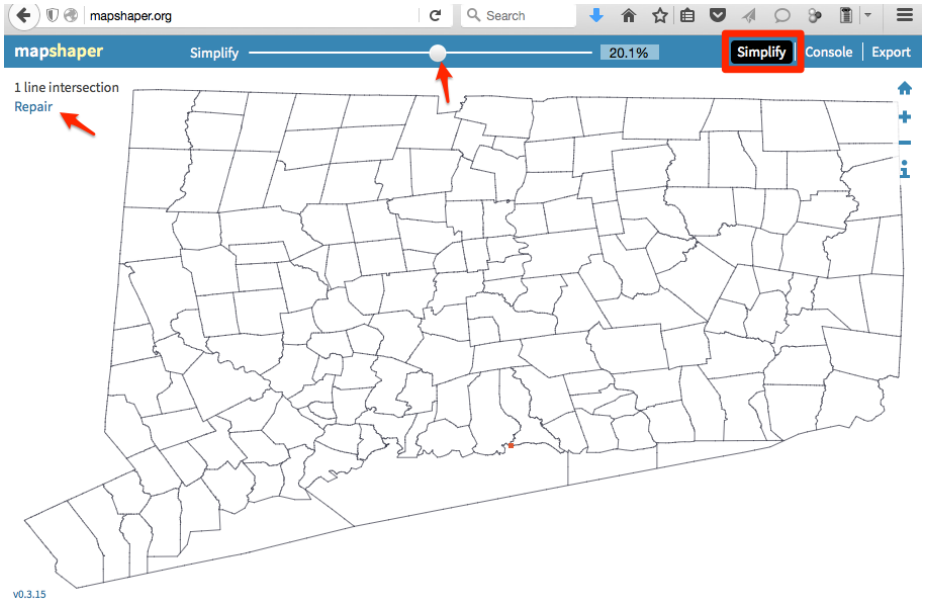
1. Try it! Download and upload the sample GeoJSON file as described in the Import section above.

2. Click the Simplify button to review options, and in most cases, accept the default settings. Click Next.

3. Slide the Simplify button from 100 percent down to an appropriate number for your map zoom level. If important geographic details disappear, you may have gone too far.

4. Look in the upper-left corner and click on recommended Repairs to your map file.

5. Complete the process by clicking Simplify once again. Export your file in the preferred format for your project.



Dissolve internal polygons to create an outline map

MapShaper.org also includes a Console button to type in commands for common map editing tasks. Imagine that you begin with a boundary map that includes internal polygons, but your goal is to remove all of them to create an outline map.

Click the Console button, which opens a window to type in commands. Enter the command below, then press return. Close the Console window and Export your outline map.

```
-dissolve
```

Clip a map to match an outline layer

Imagine that you start with a polygon map of all towns in Connecticut, and an outline map of Hartford County, a larger region that includes some (but not all) of those smaller towns. Your goal is to create a polygon map of all towns inside Hartford County. In other words, we will "clip" the statewide town map using the county outline map.

Figure 11.8: Screencast: Mapshaper dissolve

Try it! Right-click the link and Save to download both sample files to your computer:

- ct-towns in GeoJSON format
- hartfordcounty-outline in GeoJSON format
- If you accidentally open a page of GeoJSON code in your browser, select File > Save Page As to download to your computer.

Refresh the browser to start a new session in http://MapShaper.org.

1. Drag-and-drop the ct-towns.geojson file to import to MapShaper.
2. Drag-and-drop the hartfordcounty-outline.geojson map to MapShaper, and click Import to add this second layer.
3. In the drop-down menu, select the first map (ct-towns) to display it as the active layer.
4. Click the Console button, type or paste in the command below, and press enter.

```
-clip hartfordcounty-outline.geojson
```

5. The command above instructs MapShaper to clip the active map layer (ct-towns) using the second layer (hartfordcounty-outline).
6. Sometimes the boundaries of the clip layer do not precisely match up with your active layer, due to differences between their sources. If necessary, add the `cleanup` command to remove any null features or small "slivers" that remain after the clip.

```
- clip hartfordcounty-outline.geojson cleanup
Removed 3 null features and 5 slivers
```

TO DO fix animation to match new file names

Figure 11.9: Screencast: Mapshaper clip

Remove unwanted data columns

Sometimes your polygon map contains several columns of unwanted data. To quickly remove them, enter the "-filter-fields" Console command to keep only the columns you list. The example below deletes all columns *except* "town":

```
-filter-fields town
```

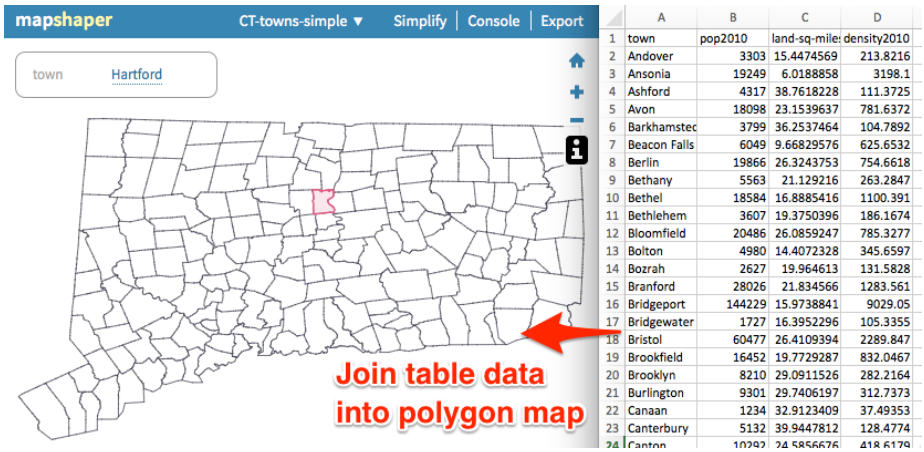
Join spreadsheet data with polygon map

** TO DO ** - fix images and animations to map the new file names and column headers

A common mapping task is to join (or merge) new data columns into a polygon boundary map, and MapShaper.org makes this very easy. Imagine that you have two files:

- Connecticut town boundary map
- a spreadsheet of town population data

Your goal is to unite these files, so that you can later display them in a thematic polygon map. Since these two files share a common column of data – the town names – you can join them together into one merged file.



	A	B	C	D
1	town	pop2010	land-sq-mile	density2010
2	Andover	3303	15.4474569	213.8216
3	Ansonia	19249	6.0188858	3198.1
4	Ashford	4317	38.7618228	111.3725
5	Avon	18098	23.1539637	781.6372
6	Barkhamstec	3799	36.2537464	104.7892
7	Beacon Falls	6049	9.66829576	625.6532
8	Berlin	19866	26.3243753	754.6618
9	Bethany	5563	21.129216	263.2847
10	Bethel	18584	16.8885416	1100.391
11	Bethlehem	3607	19.3750396	186.1674
12	Bloomfield	20486	26.0859247	785.3277
13	Bolton	4980	14.4072328	345.6597
14	Bozrah	2627	19.964613	131.5828
15	Branford	28026	21.834566	1283.561
16	Bridgeport	144229	15.9738841	9029.05
17	Bridgewater	1727	16.3952296	105.3355
18	Bristol	60477	26.4109394	2289.847
19	Brookfield	16452	19.7729287	832.0467
20	Brooklyn	8210	29.0911526	282.2164
21	Burlington	9301	29.7406197	312.7373
22	Canaan	1234	32.9123409	37.49353
23	Canterbury	5132	39.9447812	128.4774
24	Canton	10292	24.5856676	418.6179

Try it! Right-click each link and Save to download two sample files to your computer:

- ct-towns in GeoJSON format
- ct-towns-popdensity in CSV format

If you accidentally open a page of GeoJSON code in your browser, select File > Save Page As to download to your computer.

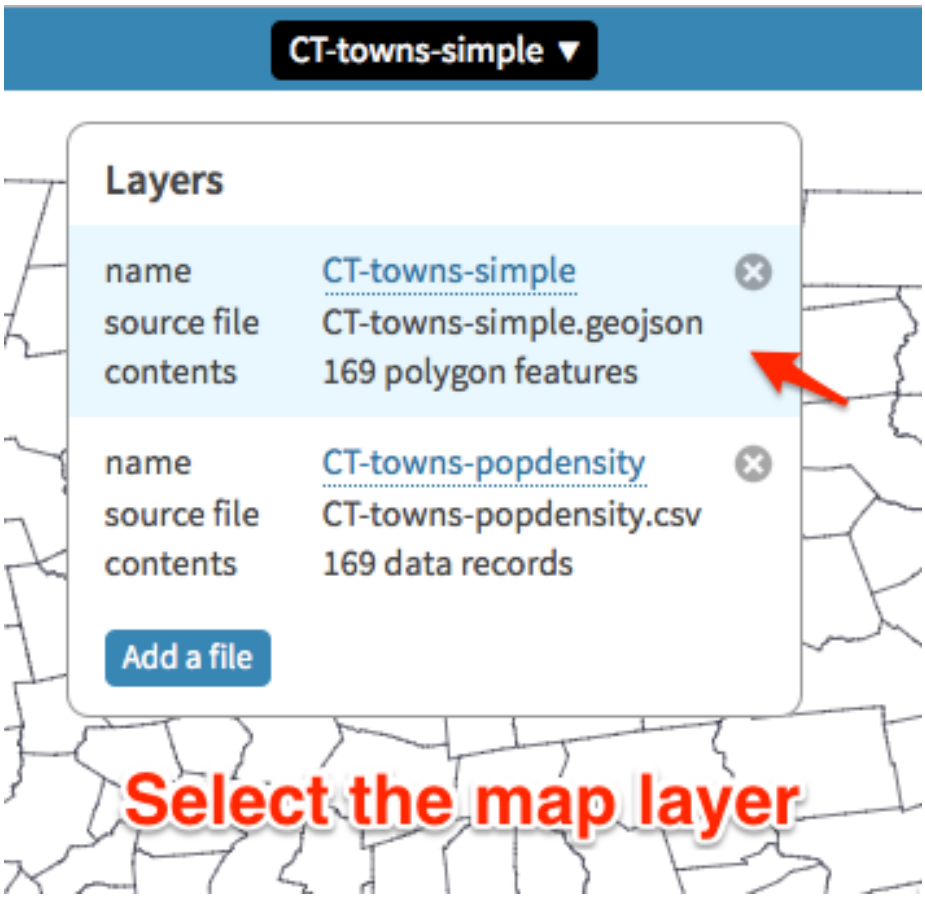
Refresh the browser to start a new session in http://MapShaper.org.

1. Drag-and-drop the ct-towns.geojson boundary file into MapShaper. Select the “i” info button and click on any polygon to confirm that the column header is “name”.

2. Open the ct-towns-popdensity.csv file with any spreadsheet tool and confirm that first column header also is “name”. Close this file.

3. Drag-and-drop the ct-towns-popdensity.csv file into MapShaper.org, and select the Import button to add it as a second layer. This table layer will appear as rectangular cells, because it does not contain geographic information.

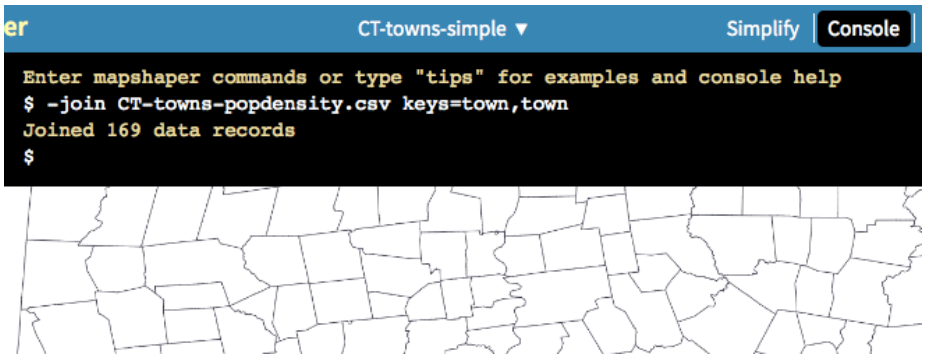
4. Click the drop-down menu and select the map to display it as the active layer.



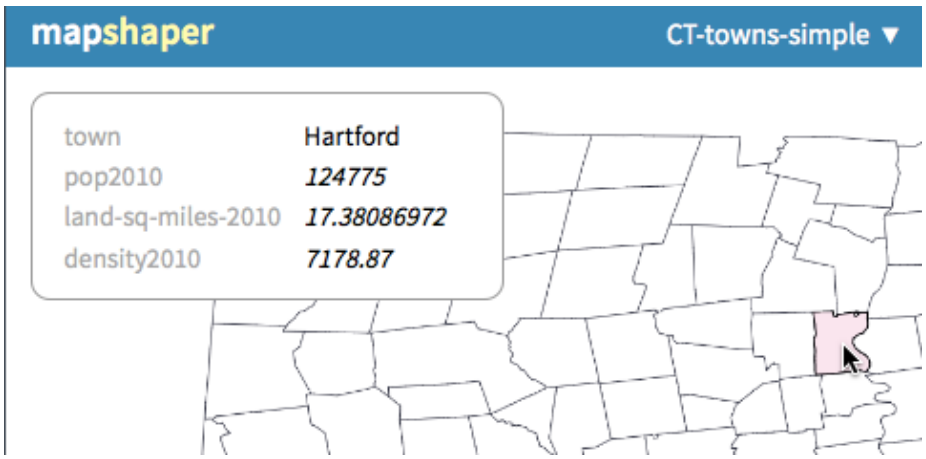
5. Click the Console button, type this command, and press return:

```
-join ct-towns-popdensity.csv keys=name,name
```

Type this precisely, with **no spaces** between the words in your keys. This command instructs MapShaper to join the active map layer to the CSV table layer, based on their shared column of data, labeled as "name" in both files. In this example, 169 rows are merged together.



6. Click the Console button to close the command window. Select the "i" info button and click any polygon to confirm that it now contains the new table data. Export the file in your preferred format.



More about joins

1. 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.org, and export in CSV format, and open with any spreadsheet tool. To match data columns in the CSV spreadsheet, use the VLOOKUP method in this book.

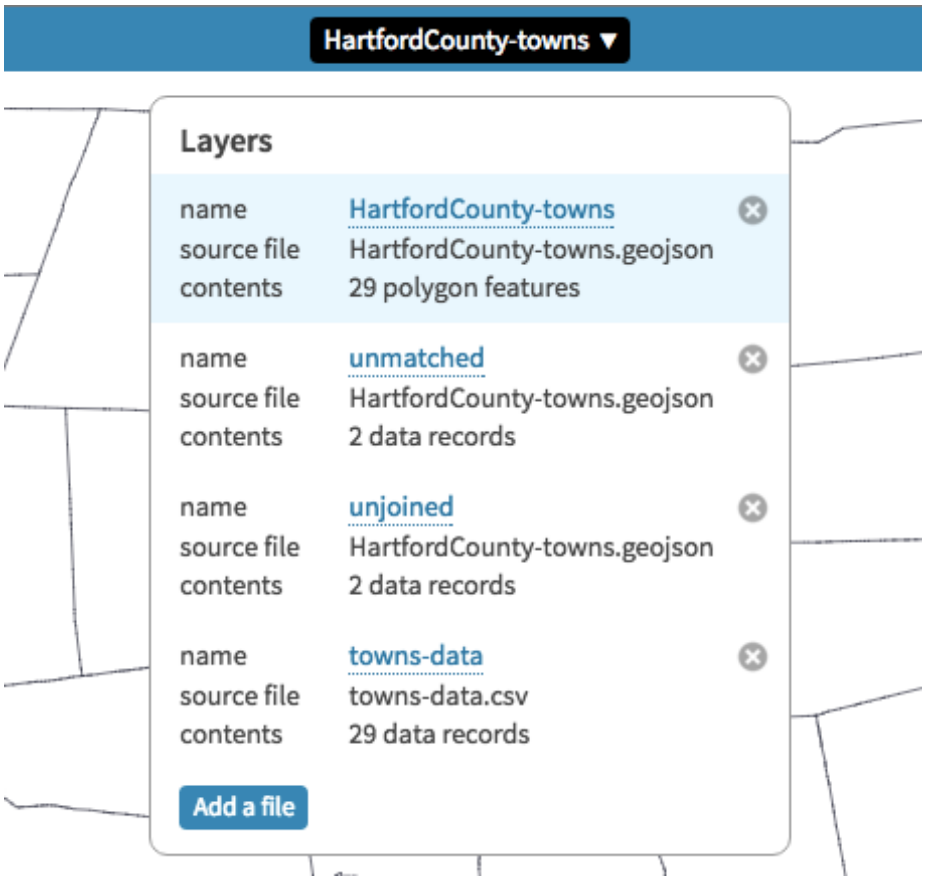
2. The simple join example above uses identical keys (name,name) because the two columns headers are the same. But if you need to join data where the headers are not the same, enter the first key (the polygon map) and the second key (the CSV table).

3. Mapshaper also helps you to keep track of data that are not properly joined or matched. For example, if the polygon map contains 169 rows (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:

```
Joined 168 data records
1/169 target records received no data
```

To capture data records that are not properly joined, add these terms at the end of your join command: `unjoined unmatched -info`. The first term saves a copy of each unmatched record from the target table to a new layer named "unmatched," and the second term saves a copy of each unjoined record from the source table into another layer named "unjoined." In the example below, see the console command and results, and a screenshot of the two new layers.

```
$ -join towns-data.csv keys=name,name unmatched unjoined -info
Joined 27 data records
2/29 target records received no data
2/29 source records could not be joined
Layer 1 ...
```



Merge selected polygons with join and dissolve commands

** TO DO ** fix screenshots to match new data files and column headers

Another common task is to merge selected polygons in a boundary map, which you can do in MapShaper with the join and dissolve commands you learned above. Imagine that you wish to create regional "cluster" boundaries from smaller polygon areas. For example, the Connecticut Department of Public Health has grouped individual towns, such as Bloomfield and West Hartford, into regional health districts. Your task is to begin with a statewide polygon map of all town boundaries, and to create a new polygon map that displays these regional clusters.

Try it! Right-click the link and Save to download the sample files to your computer: ct-towns in GeoJSON format. If you accidentally open a page of GeoJSON code in your browser, select File > Save Page As to download to your computer.

Refresh the browser to start a new session in http://MapShaper.org.

1. Import the ct-towns.geojson map file into http://MapShaper.org.

2. Export in CSV format. This steps creates a spreadsheet that lists all of the polygon town names, without geographic details.

	A	B
1	town	
2	Bethel	
3	Bridgeport	
4	Brookfield	
5	Danbury	
6	Darien	
7	Easton	
8	Fairfield	

3. Open the CSV file with any spreadsheet tool. Copy the contents of the "name" column, paste it into a second column, and change the header of this second column to "merged".

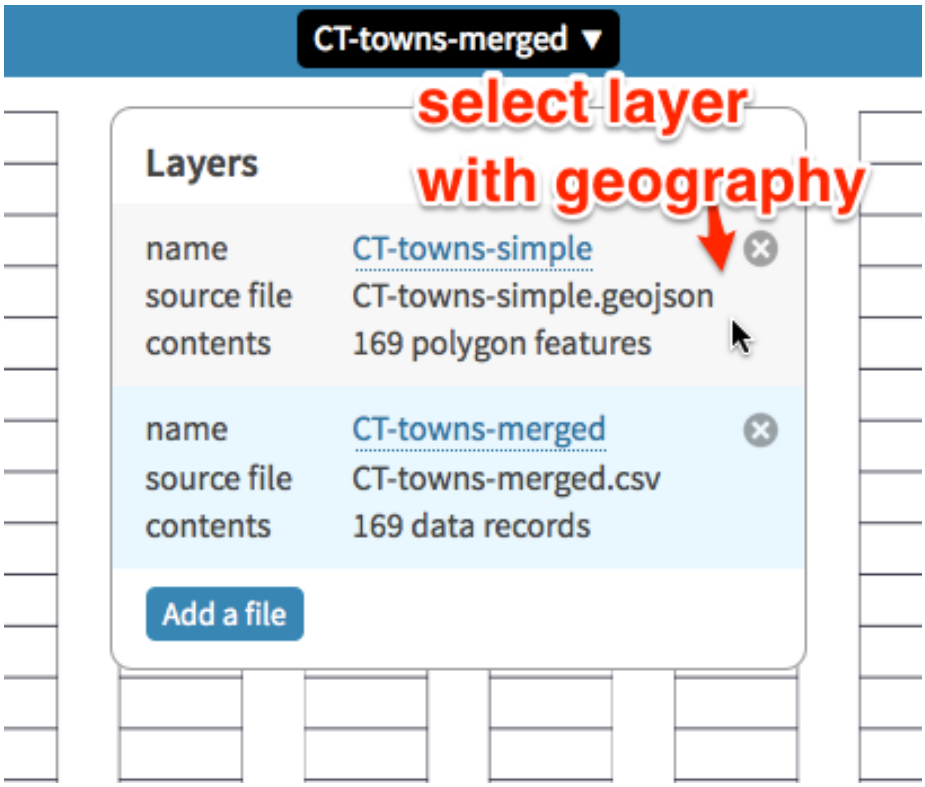
4. In the new "merged" column, create new listings for towns you wish to merge together. In this example, Bloomfield and West Hartford are merged into Bloomfield-West Hartford. Leave other towns unchanged.

	A	B
1	town	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

5. Save this spreadsheet in CSV format with a new file name, such as: ct-towns-merged.csv.

6. Drag this new ct-towns-merged.csv file into MapShaper, and click Import.

7. Use the drop-down menu to manage multiple layers in MapShaper. Since the CSV file has no geography, it appears as a series of rectangular cells. Instead, select the ct-towns.geojson map to display it as the active layer.



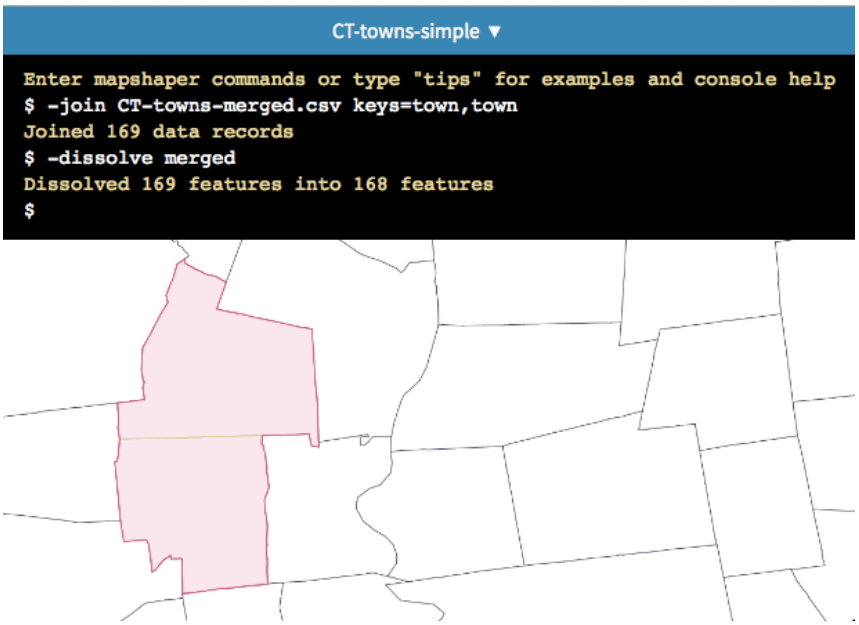
8. Click on the Console button, type in both of the commands below, and press Return at the end of each line:

```
-join CT-towns-merged.csv keys=name,name
-dissolve merged
```

How to understand the commands above:

- The first line "joins" the active layer (the polygon map) to the CSV spreadsheet, with "keys" to match their shared data columns, which are both labeled as "name".

- The second line dissolves the polygons of towns listed in the "merged" column of the CSV file. In this simple example, only Bloomfield and West Hartford are dissolved into a combined "Bloomfield-West Hartford" regional health district, and all of the other polygons remain the same.



Click the Console button to close its window. Select the "i" information button to inspect your merged polygons. Export the map in your preferred format.

Learn more advanced MapShaper methods

- See the MapShaper GitHub project wiki (https://github.com/mbloch/mapshaper/wiki/) for more command references and tips about map simplification

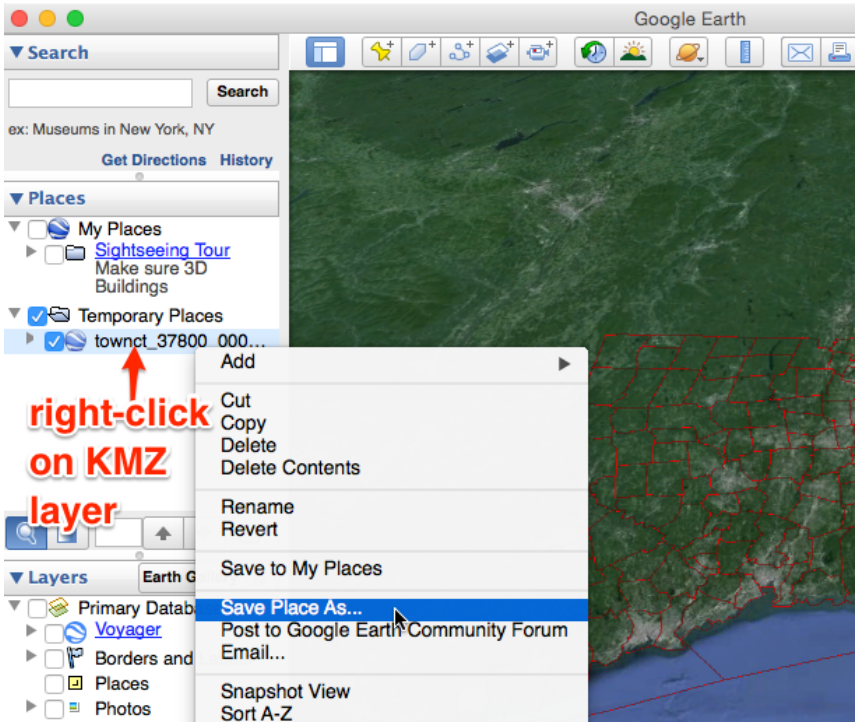
TO DO: illustrate concept of a point-to-polygon spatial join. When using the join command, "If the keys= option is missing, Mapshaper will perform a point-to-polygon or polygon-to-point spatial join."

Convert a Compressed KMZ file to KML format

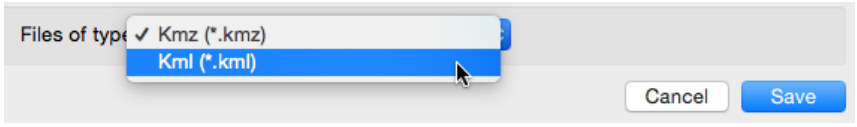
Sometimes KML files are distributed in compressed KMZ format. One easy conversion method:

- Install the free Google Earth tool
- Double-click on any .kmz file to open it in Google Earth

- Right-click (or control-click) on the .kmz layer and select *Save As*



- Use the *Save As* drop down menu to select .kml format



Or use any zip-utility and simply unzip the kmz file. Kmz is simply a 'zipped' version of a kml file!

Chapter 12

Detect Bias in Data Stories

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)

Enroll in our free online course *TO DO add link*, which introduces these topics in the brief video below, and offers more exercises and opportunities to interact with instructors and other learners.

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.

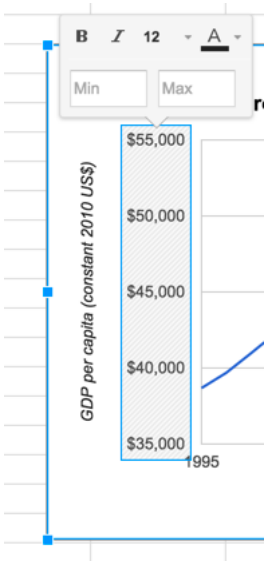


Figure 12.1: Screenshot: Edit the Min and Max values of the Y-axis

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:

** 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...

Chapter 13

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.

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

Appendix A

Fix Common 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: Start here; Reorganize the logic of subsections below, perhaps in this way

- Problems with iframes (since this chapter appears before code templates)
- Problems with GitHub forking and hosting (the core of this chapter)
- Problems with code templates (the latter portion of this chapter and the next two chapters)
- Problems with Mac computers

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
```

  - FIXED iframe (with correct quotation marks)

```
<iframe src="https://handsondataviz.github.io/leaflet-map-simple" width="90%" heig
```

- 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.

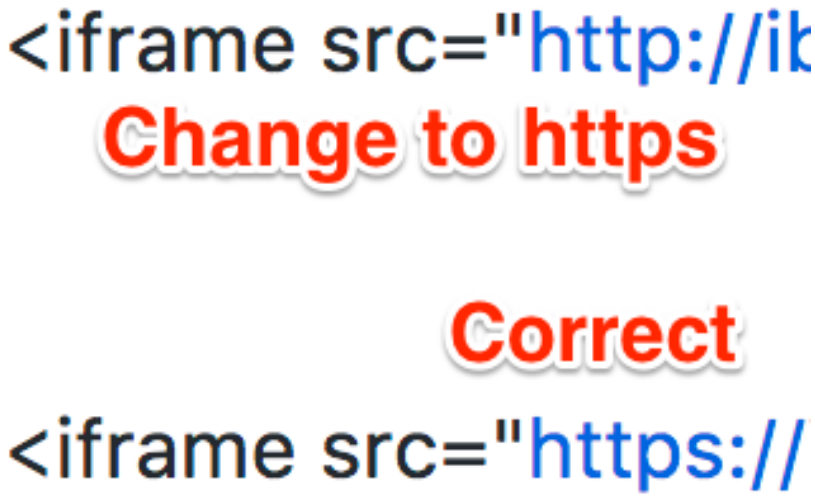


Figure A.1: Screenshot: Replace http with https

- 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?

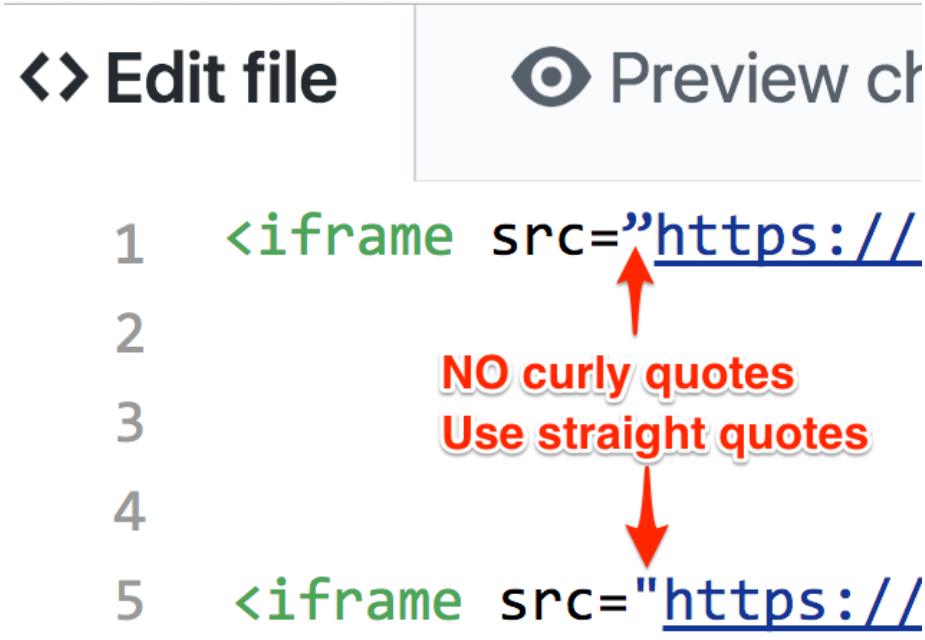


Figure A.2: Screenshot: Curly quotes versus straight quotes

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/REPOSITO`
- 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 do a "hard refresh" to bypass any saved content in your browser 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`)

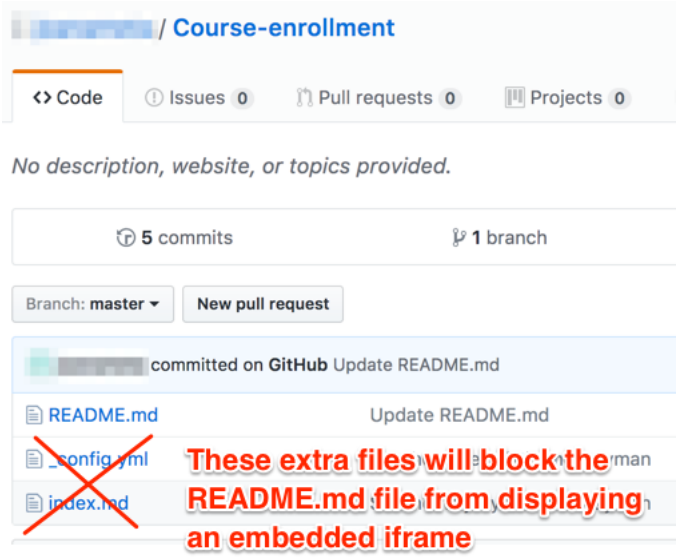


Figure A.3: Screenshot: Extra files in GitHub repo will block iframe in your README

- 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/REPOSIT`
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.

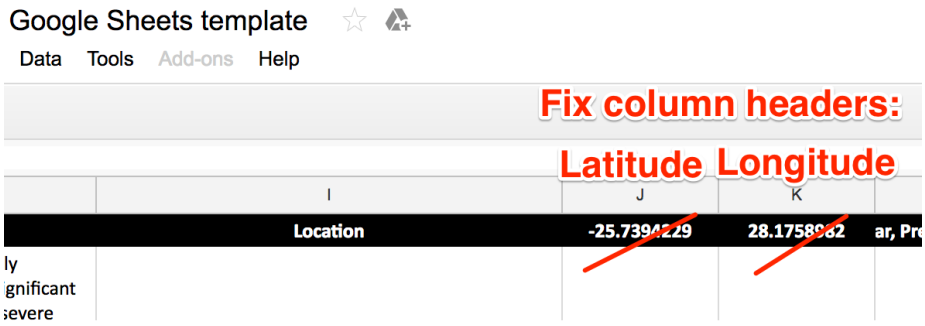


Figure A.4: Screenshot: User accidentally renamed column headers in the Points tab

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

Problems with Mac Computers

No file extensions Several tools in this book will not work properly if your Mac Finder does not display file extensions. In other words, every file should include a period followed by several letters (such as data.csv and map.geojson). If you do not see these extensions at the end of each file name, then go to Finder > Preferences > Advanced and check the box to turn them on, as show below:

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

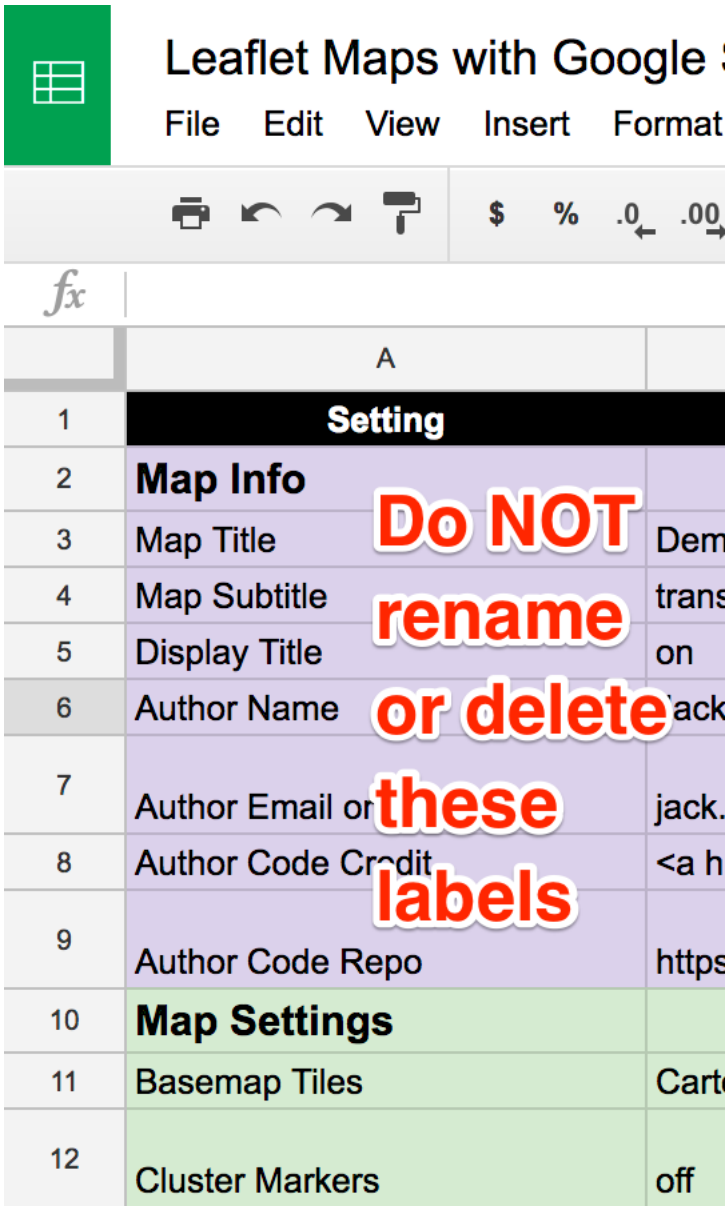


Figure A.5: Screenshot: Do not rename or delete

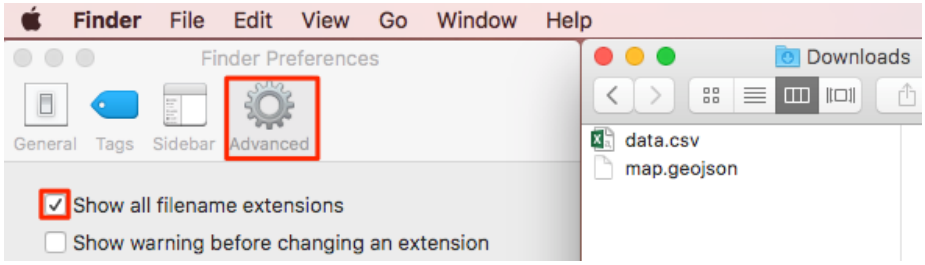
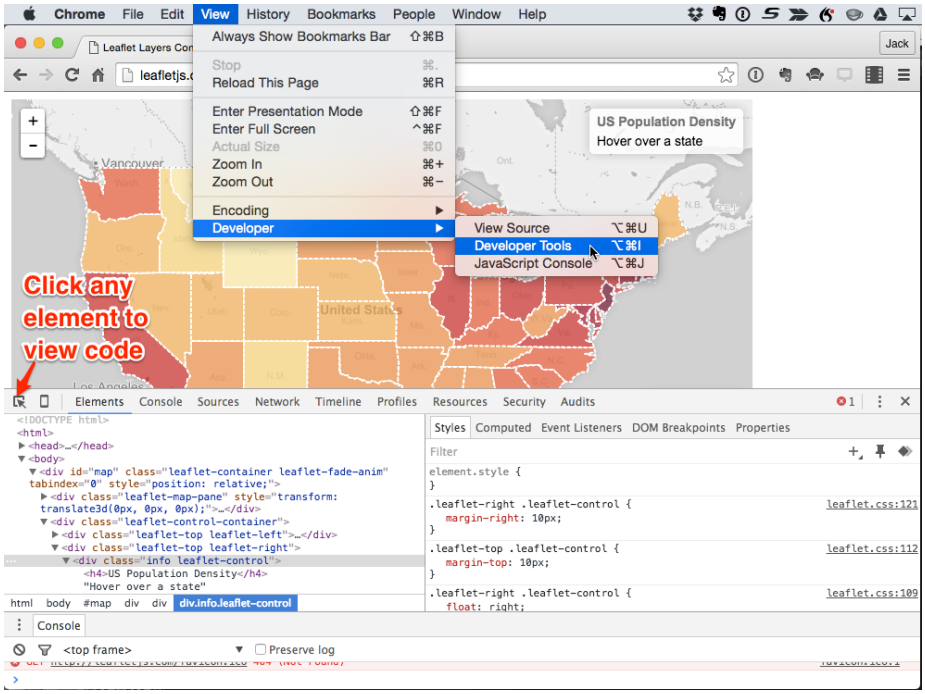
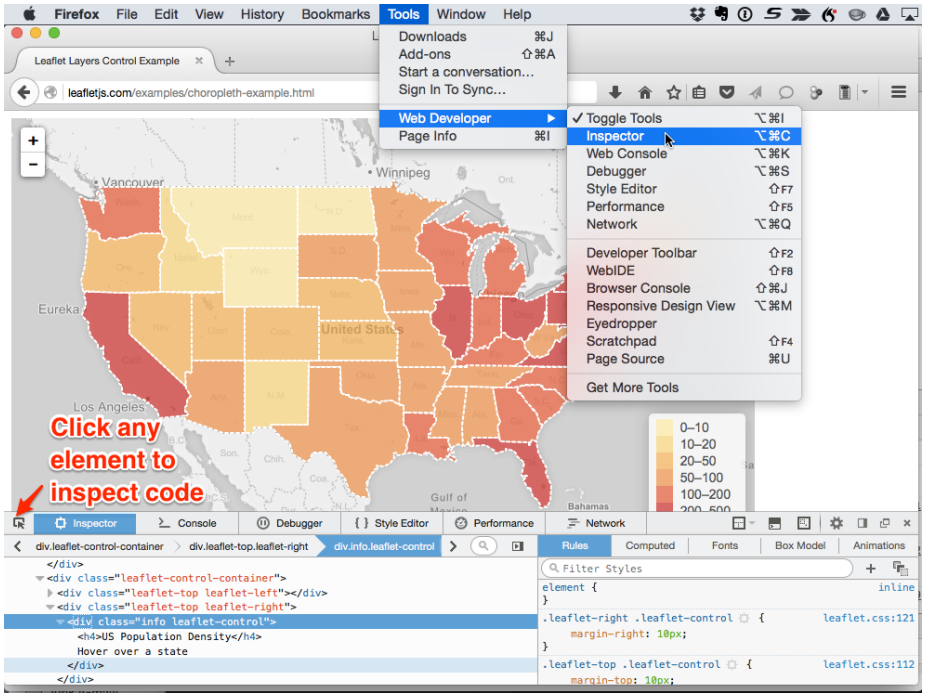


Figure A.6: Screenshot: Checkbox to show filename extensions



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


Firefox File Edit View History Bookmarks Tools Window Help
Downloads ⌘J
Add-ons ⇧⌘A
Start a conversation…
Sign In To Sync…
Web Developer
Page Info ⌘I
Toggle Tools ⌥⌘I
Inspector ⌥⌘C
Web Console ⌥⌘K
Debugger ⌥⌘S
Style Editor ⇧F7
Performance ⇧F5
Network ⌥⌘Q
Developer Toolbar ⇧F2
WebIDE ⇧F8
Browser Console ⇧⌘J
Responsive Design View ⌥⌘M
Eyedropper
Scratchpad ⇧F4
Page Source ⌘U
Get More Tools
Leaflet Layers Control Example
leafletjs.com/examples/choropleth-example.html
Vancouver
Winnipeg
Eureka
Los Angeles
United States
Gulf of Mexico
Bahamas
0–10
10–20
20–50
50–100
100–200
200–500
Click any element to inspect code
Inspector
Console
Debugger
Style Editor
Performance
Network
div.leaflet-control-container
div.leaflet-top.leaflet-right
div.info.leaflet-control
Rules
Computed
Fonts
Box Model
Animations
Filter Styles

<h4>US Population Density</h4>
Hover over a state

element { inline
}
.leaflet-right .leaflet-control { leaflet.css:121
margin-right: 10px;
}
.leaflet-top .leaflet-control { leaflet.css:112
margin-top: 10px;

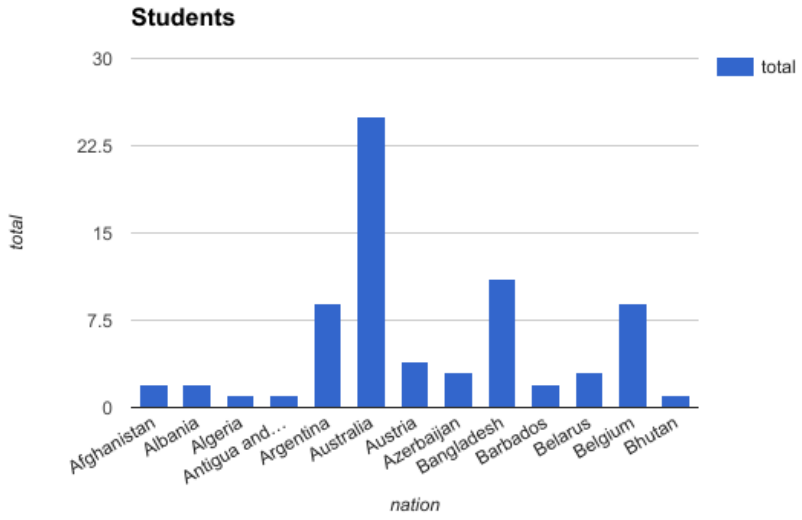
Appendix B

Peer Review Samples

ONLY FOR WEB EDITION: The next pages include partial-credit and full-credit samples for peer review in the Data Visualization for All edX course.

Section 2 Chart 1 Peer Review Sample

Students in the Data Visualization for All course come from several different countries, including Australia, Bangladesh, and Belgium.

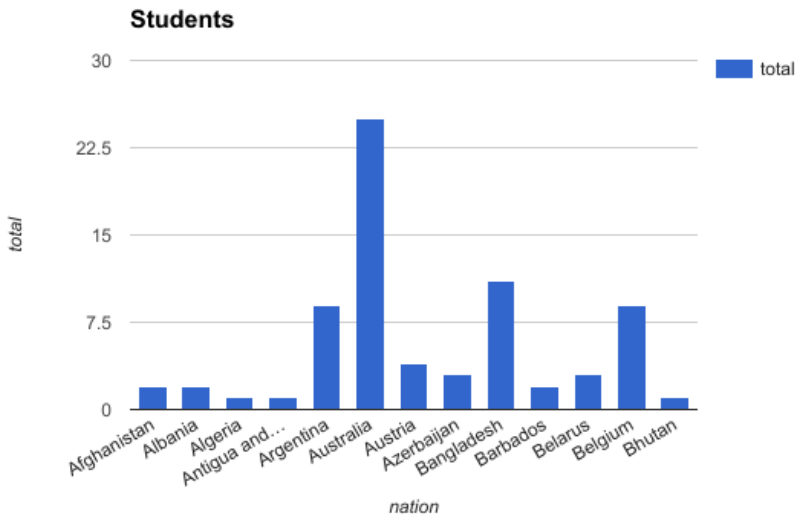


Evaluate

1. Story: Did the author clearly tell a meaningful story about the data, with text and visuals?
2. Chart Type: Did the author choose a chart type that best matches their data story?
3. Embed: Did the author embed an interactive chart into the web page?
4. Good Design: Did the author follow principles of good chart design?

Section 2 Chart 1 Peer Review Sample with Notes

Students in the Data Visualization for All course come from several different countries, including Australia, Bangladesh, and Belgium.



Evaluate

1. Story: Did the author clearly tell a meaningful story about the data, with text and visuals?

- No, this simple statement that students come from "several different countries" is not a very meaningful story.

2. Chart Type: Did the author choose a chart type that best matches their data story?

- No. Although a vertical column chart is a good start, a horizontal bar chart would be a better match for these long labels.

3. Embed: Did the author embed an interactive chart into the web page?

- No, when you try to float your cursor over the chart, it is a static image, not an interactive visualization.

4. Good Design: Did the author follow principles of good chart design?

- No, the chart ignores several design principles, such as: - Failure to sort data into a meaningful order - Failure to declutter the chart by removing the unnecessary legend

Section 2 Chart 2 Peer Review Sample

Nations with the highest percentage of female students enrolled in Data Visualization for All are the Ukraine (51 percent) and Turkey (47 percent), based on preliminary data for those with high enrollment levels (25 or more students).

View the preliminary data for 21 Feb 2017 from http://handsondataviz.org

Evaluate

1. Story: Did the author clearly tell a meaningful story about the data, with text and visuals?
2. Chart Type: Did the author choose a chart type that best matches their data story?
3. Embed: Did the author embed an interactive chart into the web page?
4. Good Design: Did the author follow principles of good chart design?

Section 2 Chart 2 Peer Review Sample with Notes

Nations with the highest percentage of female students enrolled in Data Visualization for All are the Ukraine (51 percent) and Turkey (47 percent), based on preliminary data for those with high enrollment levels (25 or more students).

View the preliminary data for 21 Feb 2017 from http://handsondataviz.org

Evaluate

1. Story: Did the author clearly tell a meaningful story about the data, with text and visuals?

- Yes, this insight on gender differences in student enrollments across nations is a meaningful story.

2. Chart Type: Did the author choose a chart type that best matches their data story?

- Yes, this stacked horizontal bar chart is a good match for showing part-to-whole relationships (gender by percentage) between different nations.

3. Embed: Did the author embed an interactive chart into the web page?

- Yes, when you float your cursor over it, the interactive chart tooltip shows data labels and values.

4. Good Design: Did the author follow principles of good chart design?

- Yes, the chart demonstrates good design principles, such as:
  - Sorting data into a meaningful order
  - Using color contrast (blue vs grays) to highlight percentages of female students

Section 3 Peer Review Sample 1

My Leaflet map

My Highcharts scatter chart

Evaluate

1. Leaflet map and title: Did the author embed an interactive Leaflet map with a new title?
2. Leaflet map layers: Did the author add controls that toggle on/off different map layers?
3. Leaflet point markers: Did the author upload a new set of markers, with pop-ups that show titles for each point?
4. Highcharts scatter chart: Did the author embed an interactive Highcharts scatter chart with a new title and axis labels?

5. Highcharts data tooltips: Did the author upload a new set of data, with tooltips that show labels and details for each point?
6. Additional comments for the author. What works well? What could be improved?

Section 3 Peer Review Sample 1 with Notes

My Leaflet map

My Highcharts scatter chart

Evaluate

1. Leaflet map and title: Did the author embed an interactive Leaflet map with a new title?

- No, the map title is the same as the original template, and is not new.

2. Leaflet map layers: Did the author add controls that toggle on/off different map layers?

- No, the map does not contain layer controls.

3. Leaflet point markers: Did the author upload a new set of markers, with pop-ups that show titles for each point?

- No, the map only contains one point, and the author did not upload a new set of points.

4. Highcharts scatter chart: Did the author embed an interactive Highcharts scatter chart with a new title and axis labels?

- No, the chart title and axis labels are the same as the original template, and are not new.

5. Highcharts data tooltips: Did the author upload a new set of data, with tooltips that show labels and details for each point?

- No, the author did not upload a new set of data points.

6. Additional comments for the author. What works well? What could be improved?

Section 3 Peer Review Sample 2

My Leaflet map

My Highcharts scatter chart

Evaluate

1. Leaflet map and title: Did the author embed an interactive Leaflet map with a new title?
2. Leaflet map layers: Did the author add controls that toggle on/off different map layers?
3. Leaflet point markers: Did the author upload a new set of markers, with pop-ups that show titles for each point?
4. Highcharts scatter chart: Did the author embed an interactive Highcharts scatter chart with a new title and axis labels?
5. Highcharts data tooltips: Did the author upload a new set of data, with tooltips that show labels and details for each point?
6. Additional comments for the author. What works well? What could be improved?

Section 3 Peer Review Sample 2 with Notes

My Leaflet map

My Highcharts scatter chart

Evaluate

1. Leaflet map and title: Did the author embed an interactive Leaflet map with a new title?

- Yes, the title in this map has changed from the original template

2. Leaflet map layers: Did the author add controls that toggle on/off different map layers?

- Yes, this map contains map layer controls.

3. Leaflet point markers: Did the author upload a new set of markers, with pop-ups that show titles for each point?

- Yes, this map contains new points that were added to the original template.

4. Highcharts scatter chart: Did the author embed an interactive Highcharts scatter chart with a new title and axis labels?

- Yes, this chart contains a new title and axis labels.

5. Highcharts data tooltips: Did the author upload a new set of data, with tooltips that show labels and details for each point?

- Yes, this chart contains a new set of data points that were uploaded to the original.

6. Additional comments for the author. What works well? What could be improved?

Appendix C

Publishing with Bookdown

This open-access book is built with free-to-use, open-source tools—primarily Bookdown, GitHub, and Zotero—and this chapter explains how, so that readers may do it themselves and share their knowledge to improve the process. In addition to our notes below, see also Yihui Xie's more comprehensive Bookdown guide.[1]

Our broad goal is an efficient workflow to compose one document in the easy-to-write Markdown format that Bookdown generates into multiple book products: an HTML web edition to read online, a PDF print edition for traditional book publishing, a Microsoft Word edition for editors who request it for copyediting, and option for other formats as desired.

Since Bookdown is an R code package, we composed the book manuscript in R-flavored Markdown, with one file (.Rmd) for each chapter. We use Bookdown to build these files in its GitBook style as a set of static HTML pages, which we upload to our GitHub repository. Readers can view the open-access web edition of the book at our custom domain: https://HandsOnDataViz. Also, we use Bookdown to build additional book outputs (PDF, MS Word, Markdown) and upload these to the `docs` folder of our GitHub repository, so that our O'Reilly Media editor may download and comment on the manuscript as we revise. Finally, we also have the option to use Pandoc alone to convert the full-book Markdown file (.md) into an AsciiDoc file (.asciidoc) for easier importing into the O'Reilly Atlas platform. See some caveats and workarounds below.

File Organization and Headers

We organized the GitHub repository for this book as a set of .Rmd files, one for each chapter. As co-authors, we are careful to work on different chapters of the

[1] Yihui Xie, *Bookdown: Authoring Books and Technical Documents with R Markdown*, 2018, https://bookdown.org/yihui/bookdown/

book, and to regularly push our commits to the repo. Only one of us regularly builds the book with Bookdown to avoid code merge conflicts.

Bookdown assigns a default ID to each header, which can be used for cross-references. The default ID for `# Topic` is `{#topic}`, and the default ID for `## Section Name` is `{#section-name}`, where spaces are replaced by dashes. But we do *not* rely on default IDs because they might change due to editing or contain duplicates across the book.

Instead, we *manually assign a unique ID* to each first- and second-level header in the following way. Note that the `{-}` symbol, used alone or in combination *with a space* and a unique ID, prevents auto-numbering in the second- thru fourth-level headers:

```
# Top-level chapter title {#unique-name}
## Second-level section title {- #unique-name}
### Third-level subhead {-}
#### Fourth-level subhead {-}
```

Also, we match the unique ID keyword to the file name for top-level chapters this way: `01-keyword.Rmd` to keep our work organized. Unique names should contain only *alphanumeric* characters (a-z, A-Z, 0-9) or dashes (-).

A special header in this book is the unnumbered header beginning with `(APPENDIX)`, which indicates that all chapters appearing afterwards are appendices. According to Bookdown, the numbering style will appear correctly in HTML and LaTeX/PDF output, but not in Word or ebooks.

```
# Chapter One

# Chapter Two

# (APPENDIX) Appendix {-}

# Appendix A

# Appendix B
```

In the Bookdown `index.Rmd` for the HTML book output and the PDF output, the `toc_depth: 2` setting displays chapter and section headers down to the second level in the Table of Contents.

The `split_by: section` setting divides the HTML pages at the second-level header, which creates shorter web pages with reduced scrolling for readers. For each web page, the unique ID becomes the file name, and is stored in the `docs` subfolder.

The `number_sections` setting is true for the HTML and PDF editions, and given the `toc_depth: 2`, this means that they will display two-level chapter-section numbering (1.1, 1.2, etc.) in the Table of Contents. Note that `number_sections` must be true to display Figure and Table numbers in x.x format, which is desired for this book. See relevant settings in this excerpt from `index.Rmd`:

```
output:
  bookdown::gitbook:
    ...
    toc_depth: 2
    split_by: section
    number_sections: true
    split_bib: true
    ...
bookdown::pdf_book:
  toc_depth: 2
  number_sections: true
```

Note that chapter and section numbering do *not* appear automatically in the MS Word output unless you supply a reference.docx file, as described below:

- https://bookdown.org/yihui/rmarkdown/word-document.html
- https://stackoverflow.com/questions/52924766/numbering-and-referring-sections-in-bookdown
- https://stackoverflow.com/questions/50609212/caption-styles-for-word-document2-in-bookdown

In the `_bookdown.yml` settings, all book outputs are built into the `docs` subfolder of our GitHub repo, as shown in this excerpt:

```
output_dir: "docs"
book_filename: "HandsOnDataViz"
language:
  label:
    fig: "Figure "
chapter_name: "Chapter "
```

In our GitHub repo, we set GitHub Pages to publish to the web using `master/docs`, which means that visitors can browse the source files at the root level, and view the HTML web pages hosted in the `docs` subfolder. We use the GitHub Pages custom domain setting so that the HTML edition is available at https://HandsOnDataViz.org.

The `docs` subfolder also may contain the following items, which are *not* generated by Bookdown and need to be manually created:

- `CNAME` file for the custom domain, generated by GitHub Pages.
- `.nojekyll` invisible empty file to ensure speedy processing of HTML files by GitHub Pages.
- `404.html` custom file to redirects any mistaken web addresses under the domain back to the `index.html` page.

One more option is to copy the Google Analytics code for the web book, paste it into an HTML file in the book repo, and include this reference in the `index.Rmd` code:

```
output:
  bookdown::gitbook:
  ...
  includes:
    in_header: google-analytics.html
```

Style Guide for *Hands-On Data Visualization*

View the underlying source code to understand how this page was composed at: https://github.com/HandsOnDataViz/book/blob/master/16-bookdown.Rmd

This book is composed in R-flavored Markdown (.Rmd), and each paragraph begins on a separate line. O'Reilly style guide prefers *italics* rather than bold. Use single back tics to display a monospaced `code` word.

O'Reilly guidelines recommend making your writing as conversational as possible. Imagine you're speaking to someone one on one, not giving a formal lecture to a large group. Refer to the reader as "you" and to yourself as "I" for a single-author book, and refer to yourselves as "we" for a co-authored book. Use active voice, not passive voice.

More from O'Reilly about chapter structure: Each chapter should begin with a paragraph or two that summarizes what the chapter is about and why that information is important to the overall topic of your book. Subsequent sections should walk readers through the information you're presenting. Keep readers oriented by including signposts like "As you learned in Chapter 3" and "I'll discuss this topic in more detail later in this chapter."

More from O'Reilly about transitions: End section X by saying something like, "Now that you understand X, you're ready to dig into topic Y," and start section Y by explaining how it relates to topic X. Daisy-chaining helps readers understand how concepts are connected and why you're covering them in this order. Finally, at the end of each chapter, summarize what you discussed in that chapter, and mention what the following chapter is going to cover.

O'Reilly encourages the use of tips, notes, and warnings, and assigns each of them an animal icon in their books (lemur, crow, and scorpion, respectively).

In this book manuscript, simply start each with a paragraph beginning with the keyword, followed by a colon, to simplify find-and-replace at a later date:

- Tip: A couple of sentences that convey a helpful bit of information, a quick way to do things better.
- Note: A couple of sentences of supplemental information. It describes something you want readers to keep in mind as they work, so you use a note to set it apart and make sure they see it.
- Warning: Similar to a note or tip, but specifically focused on a way to help readers avoid making a mistake or getting into trouble.

Insert an embedded link to O'Reilly. This appears as a colored clickable link in HTML and Word editions, and a non-colored but clickable link in the PDF edition. According to O'Reilly Atlas documentation, the AsciiDoc version should automatically unfurl for the printed edition.

Also, embed the link directly in the sentence, such as download this sample PDF, and avoid linking words such as "here" or "this web page."

When instructions refer to software menu items, use italics. Example: Select *File > Make a Copy* to save your own version to your Google Drive.

For lists, always insert a blank line *before* the items, unless they appear directly after hashtag header.

- unordered
- list

1. ordered
2. list

Dashes:

- Use a hyphen (1 dash) for hyphenated words, such as two-thirds or dog-friendly hotel.
- Use an en-dash (2 dashes) for ranges, such as the May–September magazine issue.
- Use an em-dash (3 dashes) to insert an additional thought—like this—in a sentence.

Insert `TODO` to note items to finish or review with co-author or editor.

Insert three back tics to insert a code block. Check character line length limits in O'Reilly style guide:

```
<link rel="stylesheet" href="https://unpkg.com/leaflet@1.6.0/dist/leaflet.css" />
<script src="https://unpkg.com/leaflet@1.6.0/dist/leaflet.js"></script>
```

Conditional Formatting

Conditional formatting offers the option to display text or images in some editions, but not other editions. Options:

1. Insert a HTML code comment `<!-- Comment -->` in the .Rmd file to hide a few lines of text. This appears as commented-out text in the HTML and .md formats, is not displayed in the HTML browser, and does not appear in any way in the PDF, MS Word or AsciiDoc formats.

Demo:

2. R package function `is_[html/latex]_output` allows conditional output for different book products, such as text that should appear in the HTML edition but not the PDF edition, or vice versa.

Demos:

This line appears in the PDF and Word versions, and is commented-out in the HTML and Markdown and AsciiDoc versions.

TODO: Create conditional formatting that displays *only* in the HTML edition, and allows the inclusion of R code-chunks to conditionally display images. See links for more complex conditional formatting:

- https://stackoverflow.com/questions/56808355/how-to-conditionally-process-sections-in-rmarkdown
- https://bookdown.org/yihui/rmarkdown-cookbook/latex-html.html
- https://blog.earo.me/2019/10/26/reduce-frictions-rmd/
- https://stackoverflow.com/questions/53861244/html-specific-section-in-bookdown
- https://stackoverflow.com/questions/41084020/add-a-html-block-above-each-chapter-header
- https://stackoverflow.com/questions/45360998/code-folding-in-bookdown

3. Option to customize the `style.css` code for the HTML book.
4. Option to add headers, footers, preambles to the HTML or LaTeX versions.
5. Build different versions of the HTML and LaTeX (PDF) books using different chapters by listing them in order in the `_bookdown.yml` file:

```
rmd_files:
  html: ["index.Rmd", "abstract.Rmd", "intro.Rmd"]
  latex: ["abstract.Rmd", "intro.Rmd"]
```

Cross-references

In order to cross-reference in Bookdown, assign a unique name or R code-chunk label to each chapter, section, figure, and table. Unique names and labels should contain only *alphanumeric* characters (a-z, A-Z, 0-9) or dashes (-).

To cross-reference any *chapter or section*, and allow readers to jump there, use a HTML link with the unique name, such as `index.html` or `style-guide.html`. Demos:

- See Introduction
- See "Style Guide" in Chapter x.

Contrary to the Bookdown manual, *avoid* using Bookdown unique ID links to cross-reference chapters or sections, because these create extraneous and imprecise URLs, such as this bad example.

To cross-reference figures and tables, and display their auto-number and allow readers to jump there, write a call-out with a Bookdown reference to a code-chunk label, such as `See Figure \@ref(fig:sample-map)` or `See Table \@ref(tab:left-table)`. Demos:

- See Figure C.1.
- See Table C.2.

Cross-reference interactivity varies by output:

- In HTML, all cross-refs are clickable.
- In PDF, all cross-refs are clickable (except chapter-level HTML links).
- In Word, no cross-refs are clickable (unless this varies with reference.docx).
- TBA with Markdown (.md) and AsciiDoc.

When writing cross-references in the text, the O'Reilly Style Guide prefers live cross references (e.g., "see Figure 2-1"), but if not feasible, use "preceding" or "following" because physical placement of elements may vary across print and digital formats. *Avoid* using "above" or "below."

Images

View the underlying source code to understand how this page was composed at: https://github.com/HandsOnDataViz/book/blob/master/16-bookdown.Rmd

Create high-resolution color screenshots and other static images in .png or .jpg format, with tight cropping on a high-resolution Retina monitor, typically at

144 ppi. Save items into the `images` subfolder by chapter. Make sure that color images include high contrast and/or shading, because they will be converted to grayscale by the publisher for the print book. Write file names in lowercase with dashes (not spaces) and begin with keyword of relevant section to keep related images grouped together. Despite being in separate folders, avoid duplicate image file names across the book. Avoid numbering images since they may not match the final sequence.

If a screenshot requires additional artwork or text for the HTML edition, make a copy of the original and add `-ann` to note that this version is annotated, save into the same folder with the same root file name, and use in the code-chunk image pathnames. The publisher will use the original image and add their own artwork for their editions.

If an image is larger than approximately 300px on either side, one more option is to reduce the image size in the PDF version. Use Photoshop (*not* Preview) to reduce the image size, and save a copy with the same file name with the .pdf extension into the folder. In some cases, the folder will contain only one version of each image, but in other cases it may contain up to three versions of an image:

```
images/05-chart/design-no-junk.png
images/05-chart/design-no-junk-ann.png
images/05-chart/design-no-junk-ann.pdf
```

If creating images to appear as the same size in sequence, add a code-comment with the image width, height, and resolution as a reminder to make others match up, like this:

```
<!-- Images below are 200x200 at 300 resolution -->
```

In this book, use *Markdown formatting only for images that appear inside tables* or *do not require captions or figure numbering*, and leave the caption field blank, like this example:

Co-Authors	About Us
	About Jack Dougherty
	About Ilya Ilyankou

Although Markdown formatting offers a simple syntax that easily converts into other formats with Bookdown/Pandoc, there is no auto-numbering in the HTML edition, while auto-numbering appears in the PDF edition, and numbered figures are required by the publisher. Furthermore, Markdown formatting does not allow conditional output.

Images using R code-chunks

For these reasons, this book *primarily uses R code-chunk formatting for images.* The syntax is more complex but supports auto-numbering in HTML and PDF, and conditional output for interactive and static images. Note that R code-chunk images do *not* easily convert with Pandoc from Markdown to AsciiDoc, but "Figure x Caption" appears as a placeholder.

Auto-numbering appears in `Figure x.x` format in HTML and PDF, but `Figure x` format in MS Word. Note that Word formatting can be changed with reference.docx.

Note that images in PDF output will "float" by design and may appear before or after the desired page, so always add a cross-reference call-out.

Write R code-chunk labels that follow the image file name, and avoid duplicate labels across the book:

```
ref:design-no-junk

images/05-chart/design-no-junk.png
```

Do not insert spaces inside the `ref:chunk-label` for the caption, but add a blank line to separate it from the code-chunk. After the code-chunk, add *another* blank line to avoid "undefined reference" Bookdown errors.

For each figure, manually add a cross-reference call-out and insert `fig.pos='h'` to place the image "here" on the page in the PDF output, to *attempt* to avoid PDF floating. Ignore the Bookdown LaTeX warning message "h float specifier changed to ht." See more at https://bookdown.org/yihui/bookdown/figures.html and https://bookdown.org/yihui/rmarkdown-cookbook/figure-placement.html

This Bookdown `index.Rmd` file includes two global R code-chunk settings immediately after the first header. One setting displays each code-chunk image without a code echo. The other setting automatically inserts the PDF version of an PNG/JPG image, whenever it exists, in the PDF output, which allows us to manually reduce the image sizes for the PDF book. Read more about these options in this Bookdown chapter: https://bookdown.org/yihui/bookdown/figures.html.

```
{r setup, include=FALSE}
knitr::opts_chunk$set(echo = FALSE)
options(knitr.graphics.auto_pdf = TRUE)
```

Demo: R code-chunk for small static image for HTML and PDF editions

Small is defined as each side less than 300px, as shown in Figure C.1.

Figure C.1: Caption here. Markdown embedded links are acceptable.

R code-chunk for larger static image using out.width and PDF img

For larger images, where one side is greater than 300px, set the out.width to a pixel number for ideal display in the HTML edition. Also, if needed, copy the image, use Photoshop to create a smaller image size, and save with same file name and a .pdf extension for auto-substitution in the PDF output . . . as shown in Figure C.2.

R code-chunks allow more complex conditional formatting, where an interactive map or animated GIF or YouTube video clip appears in the web version, and a static image with an embedded link appears in the PDF and MS Word outputs. To change the height of the default 400px iframe, add the new height to `include_url` as shown in the examples. However, to change the width of the default 675px iframe to less than 100 percent, add a line in a `custom-scripts.html` file.

Demo: R code-chunk for iframe in HTML and static image in PDF

. . . as shown in Figure C.3.

	A
1	Location
2	300 Summit St, Hartford, CT
3	84 Scarborough St Hartford CT
4	4 Frederick Rd, West Hartford
5	4 Fredrick Rd, West Hartford
6	4 Fredric Rd, West Hartford
7	
8	
9	
10	
11	
12	
13	
14	
15	
16	
17	
18	
19	
20	
21	
22	
23	
24	

Figure C.2: Using out.width=200 and smaller PDF image size.

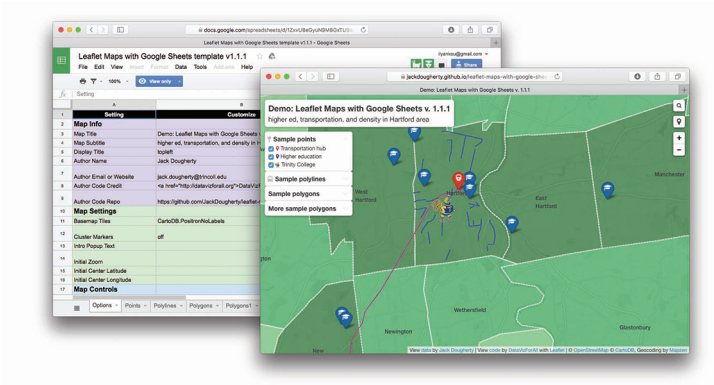


Figure C.3: Caption here, and add embedded link to explore the full-screen interactive map.

Demo: R code-chunk for animated GIF in HTML and static image in PDF

When appropriate, create animated GIF files using Camtasia, and add fade-to-black to mark the end-point in the looped version. TODO: ADD this here . . . as shown in Figure C.4.

Demo: R code-chunk for Youtube video in HTML and static image in PDF

Be sure to use the *embed* link from the YouTube *share* button.

. . . as shown in the video C.5.

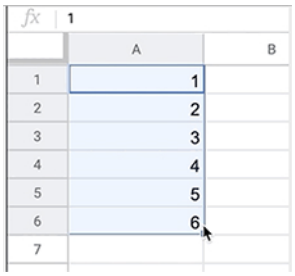


Figure C.4: Caption here, with embedded link to the animated GIF.

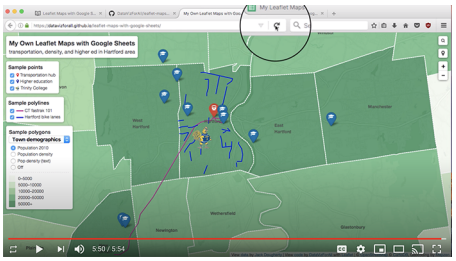


Figure C.5: Caption here, with embedded link to the YouTube video.

Demo: R code-chunk for YouTube video in HTML, with NO static image in PDF

This option may be relevant when you wish to display a video only in the HMTL edition, with no screenshot of it in the PDF edition. Note that this will alter figure-numbering between the HTML and PDF editions.

Tables

View the underlying source code to understand how this page was composed at: https://github.com/HandsOnDataViz/book/blob/master/16-bookdown.Rmd

Create tables in Markdown format, since it produces good output for HTML, PDF, Word, and Markdown. Use a tool such as Tables Generator to import significant table data in CSV format, format the column alignment as desired, and press Generate button to create table in Markdown format. For significant table data, save the CSV version in a GitHub repo for potential later use.

TODO: Check if any un-numbered Markdown tables in the chapter affect table auto-numbering.

Add the Markdown table code shown below to auto-number (Table x) in HTML, PDF, Word.

. . . as shown in Table C.2.

Table C.2: Left-justify content, remember blank Line

Much Much Longer Header	Short Header	Short Header
Left-justify text content with left-colons	Less	Here
Use more hyphens to grant more space to some columns	Less	Here

Table C.3: Right-justify content, remember blank line

Header1	Header2	Header3
123	456	789
Right-justify	numerical content	with right-colons
Use equal hyphens	to make equal space	for all columns

Workaround for Markdown-to-AsciiDoc: Currently, our attempt to use Pandoc to directly convert a Bookdown-generated Markdown file to AsciiDoc fails because Bookdown creates the .md file with tables in .html format, not Markdown. Our workaround is to paste the individual Markdown-formatted tables directly from the .Rmd into the large .md file prior to converting with Pandoc to AsciiDoc.

Notes and Bibliography

This book displays endnotes for each chapter in the HTML book, and footnotes at the bottom of pages for the PDF and MS Word books, followed by an alphabetized bibliography of all references cited on the last page. The notes and bibliography also appear in the full-book Markdown file.

To create notes, insert citation keys in the text, such as `@huffHowLieStatistics1954`, which are generated by Zotero bibliographic database with the Better BibTex extension, and export these in the *Better BibLaTeX* format into the `dataviz.bib` in the book repo. The repo also contains `.csl` file to generate the notes and bibliography in a specific Chicago-style format, downloaded from the Zotero Styles Repository. These instructions are referenced in the `index.Rmd` file for both the HTML and PDF formats, as shown in these excerpts:

```
bibliography: dataviz.bib
citation-style: chicago-fullnote-bibliography.csl
...
output:
```

```
bookdown::gitbook:
  ...
  pandoc_args: [ "--csl", "chicago-fullnote-bibliography.csl" ]

bookdown::pdf_book:
  ...
  citation_package: none
  pandoc_args: [ "--csl", "chicago-fullnote-bibliography.csl" ]
```

Here's a text-only note, with no Zotero citation.[2]

To create a note with citations only, separate Zotero/BibTeX citation keys with semi-colons:[3]

Since notes also may include text and punctuation in Markdown syntax, always insert a caret symbol prior to the brackets to demarcate a note:[4]

Note that the `chicago-fullnote-bibliography.csl` format automatically shortens the note after it its first reference.

Manual Pandoc Conversion to AsciiDoc

The O'Reilly Atlas platform can import AsciiDoc documents. While Bookdown does not directly generated this format, one workaround is use the Bookdown-generated large Markdown file, and manually convert it with Pandoc to AsciiDoc format.

- Download Pandoc
- Set Bookdown to build the book as one large Markdown file (docs folder, suffix .md)
- Use command line to navigate to subfolder with `pwd` and `cd`.
- Convert with: `pandoc handsondataviz.md --from markdown --to asciidoc --standalone --output handsondataviz.asciidoc`

See workaround notes in the Tables section and other sections above.

[2]This is a note, with no bibliographic reference.

[3]Darrell Huff, *How to Lie with Statistics* (W. W. Norton & Company, 1954–2010), 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

[4]Compare how "lying" is justified by Huff, *How to Lie with Statistics*, pp. 10-11 and Monmonier, *How to Lie with Maps*, pp. 11-12.

Huff, Darrell. *How to Lie with Statistics*. W. W. Norton & Company, 1954–2010. http://books.google.com/books?isbn=0393070875.

Monmonier, Mark S. *How to Lie with Maps*. 2nd ed. University of Chicago Press, 1996. http://books.google.com/books?isbn=0226534219.

Xie, Yihui. *Bookdown: Authoring Books and Technical Documents with R Markdown*, 2018. https://bookdown.org/yihui/bookdown/.