

Hands-On Data Visualization

Interactive Storytelling from Spreadsheets to Code

Jack Dougherty Ilya Ilyankou

2020-08-18

Contents

Introduction	21
Authors	23
Acknowledgements	24
What is Data Visualization?	25
Why this book?	26
How to Read on the Web	28
1 Choose Tools to Tell Your Data Story	31
Draw and Write Your Data Story	32
Ask Questions When Choosing Tools	33
Rate Three Simple Map Tools	35
2 Strengthen Your Spreadsheet Skills	37
Select your Spreadsheet Tools	38
Download to CSV or ODS Format	41
Make a Copy of a Google Sheet	42
Share Your Google Sheets	44
Upload and Convert to Google Sheets	46
Collect Data with Google Forms	48
Sort and Filter Data	51
Calculate with Formulas	53
Summarize Data with Pivot Tables	56
Match Columns with VLOOKUP	59
Connect Sheets with a Relational Database	63

3 Find and Question Your Data	67
Guiding Questions for your Data Search	68
Public and Private Data	70
Open Data Repositories	73
Source Your Data	75
Recognize Bad Data	76
Question Your Data	77
4 Clean Up Messy Data	81
Clean Data with Spreadsheets	82
Extract Tables from PDFs with Tabula	86
Clean Data with OpenRefine	89
5 Table Your Data	97
6 Chart Your Data	99
Chart Design Principles	102
Google Sheets Charts	109
Column and Bar Charts with Google Sheets	109
Pie, Line, and Area Charts with Google Sheets	120
XY Scatter and Bubble Charts with Google Sheets	123
Create Charts with Tableau Public	129
Create XY Scatter Chart with Tableau Public	129
Create Filtered Line Chart with Tableau Public	134
7 Map Your Data	139
Map Design Principles	141
Point Map with Google My Maps	145
Choropleth Map with Datawrapper	149
Filtered Point Map with Socrata	155
Polygon Map with Tableau Public	164

CONTENTS	5
8 Embed On Your Web	171
Create a Simple Web Page with GitHub Pages	172
Copy an iframe code from a Google Sheets interactive chart	174
Convert a Weblink into an Iframe	177
Embed an Iframe in GitHub Pages	178
Embed an Iframe on WordPress.org	180
Embed Tableau Public on your Website	182
9 Edit and Host Code with GitHub	187
Copy, Edit, and Host a Simple Leaflet Map Template	189
Create a New Repo and Upload Files on GitHub	197
GitHub Desktop and Atom Editor to Code Efficiently	202
10 Chart.js and Highcharts Templates	213
Bar Chart.js with CSV Data	215
Line Chart.js with CSV Data	216
Scatter Chart.js with CSV Data	216
Bubble Chart.js with CSV Data	217
11 Leaflet Map Templates	219
Leaflet Maps with Google Sheets	221
Leaflet Storymaps with Google Sheets	233
Get Your Google Sheets API Key	240
Leaflet Maps with CSV Data	247
Leaflet Maps with Open Data API	252
12 Transform Your Map Data	259
Geocode Locations into Coordinates with US Census or Google	260
Pivot Address-Level Point Data into Polygon Data	265
Normalize Data to Create Meaningful Choropleth Maps	267
Convert to GeoJSON format	269
GeoJson.io to Convert, Edit, and Create Map Data	273
Mapshaper to Convert, Edit, and Join Data	277

Georectify a Digitized Map with MapWarper	292
Convert a Compressed KMZ file to KML format	292
13 Detect Bias in Data Stories	295
How to Lie with Charts	295
How to Lie with Maps	297
14 Tell Your Data Story	299
A Fix Common Mistakes	301

List of Tables

6.1	Basic Chart Types and Tutorials	100
7.1	Basic Map Types and Tutorials	139
10.1	Chart Templates and Tutorials	214
11.1	Map Templates and Tutorials	220

List of Figures

1	Images: Menard’s figurative map (left) and Snow’s dot map (right), from Wikimedia	25
2	Screenshot: How to read	29
3	Screenshot: Open link in new tab (on Mac)	29
4	Image: Laptop with second monitor, and with tablet	30
1.1	Diagram: the ‘sweet spot’ for easy-to-learn and powerful tools	33
1.2	Image: Sample address data screenshot	35
2.1	Screenshot of a typical spreadsheet, with headers, tabs, and the active cell displaying a formula.	38
2.2	Three data formats commonly seen on your computer—csv, ods, and xlsx—when displayed properly in the Mac Finder.	40
2.3	On a Mac, go to <i>Finder-Preferences-Advanced</i> and check the box to <i>Show all filename extensions</i>	41
2.4	In Google Sheets, go to <i>File - Download As</i> to export data in several common formats.	42
2.5	Go to <i>File - Make a Copy</i> to create your own version of this Google Sheet.	43
2.6	Click the <i>My Drive</i> and <i>New folder</i> buttons to save your work in a folder.	43
2.7	Click the <i>Share</i> button to grant access to individuals (top half) or anyone with the link (bottom half).	45
2.8	Use a free link-shortening service, such as Bitly.com, and customize its back-end.	45
2.9	Click your Google Drive <i>Gear Symbol - Settings</i> in the upper-right corner.	46

2.10 Inside your Google Drive Settings, check the box to automatically convert all uploads.	47
2.11 Drag-and-drop your sample Excel file into your Google Drive to upload it.	47
2.12 If you forget to convert uploads, Google Drive will keep files in their original format with these icons.	48
2.13 The Google Forms tool is partially hidden in the Google Drive <i>New - More</i> menu.	49
2.14 The Google Forms <i>Questions</i> tab allows you to designate different types of responses.	50
2.15 Click the three-dot kebab menu to <i>Show - Description</i> to add details for any question.	50
2.16 Click the <i>Eyeball symbol</i> to preview your form.	51
2.17 The Google Forms <i>Responses</i> tab includes a button to open results in a linked Google Sheet.	51
2.18 Click the upper-left corner to select all cells before sorting.	52
2.19 Go to <i>Data - Sort Range</i> , check the header row box, and sort by <i>Experience with dataviz</i> in ascending order.	52
2.20 In Google Sheets, go to <i>View - Freeze</i> to select the number of rows to continuously display when scrolling downward.	53
2.21 Go to <i>Data - Create a Filter</i> , click the downward arrow in the <i>Occupation</i> column, select only <i>educator</i>	53
2.22 Right-click on row number 1 and select <i>Insert 1 below</i>	54
2.23 Type $=$ to start a formula and select the suggestion for average, or type it directly in with the correct range.	54
2.24 Click on the blue bottom-right dot in cell E2, then hold-and-drag your crosshair cursor in cell F2, and let go to automatically paste and update the formula.	55
2.25 Select or enter a formula that counts responses if the entry is <i>educator</i>	56
2.26 Go to <i>Data - Pivot Table</i> , and create in a new sheet.	57
2.27 In the <i>Pivot table editor</i> , click the Rows <i>Add</i> button and select <i>Occupation</i>	57
2.28 In the <i>Pivot table editor</i> , click the Values <i>Add</i> button and select <i>Occupation</i>	58
2.29 In the <i>Pivot table editor</i> , click the Columns <i>Add</i> button and select <i>Experience with data visualization</i>	58

2.30 In the <i>Pivot table editor</i> , see multiple options to summarize <i>Values</i>	59
2.31 Your goal is to create one mailing list that matches individual names and organizations on the left sheet with their addresses on the right sheet.	60
2.32 Paste the last four column headers from the <i>addresses</i> sheet into the <i>names</i> sheet.	61
2.33 The VLOOKUP formula in cell C2 of the <i>names</i> sheet (top) searches for matches across columns A to E in the <i>addresses</i> sheet (bottom).	61
2.34 Click on cell C2, then hold-and-drag the bottom-right blue dot across columns D to F, which automatically pastes and updates the formula.	62
2.35 Click on cell F2, then hold-and-drag the bottom-right blue dot down to row 11, which automatically pastes and updates the formula.	62
2.36 In this Airtable sample, we linked the <i>organization</i> column in the <i>people</i> sheet to the <i>food banks</i> sheet.	64
2.37 In this Airtable demo, click on a row in one sheet to expand and view its linked data in another sheet.	65
3.1 Create separate spreadsheet tabs for data, notes, and backup.	76
4.1 More often than not, raw data looks like this.	81
4.2 Find and Replace window in Google Sheets.	83
4.3 Select <i>Data - Split text to columns</i> to automatically separate data. .	84
4.4 Use ampersands to combine items and separate them with spaces. .	86
4.5 Tabula welcome page.	88
4.6 Selected tables are highlighted in red.	89
4.7 First 20 rows of the sample dataset. Can you spot any problems with it?	90
4.8 OpenRefine starting page.	91
4.9 OpenRefine parsing options.	92
4.10 Manually remove spaces and extra characters, and change data type to number.	93
4.11 Cluster similar text values.	94
6.1 Common chart components.	103

6.2	Start your bar chart at zero.	105
6.3	Chart junk distracts the viewer, so stay away from shadows, 3D perspectives, unnecessary colors and other fancy elements.	106
6.4	Sort slices in pie charts from largest to smallest, and start at 12 o'clock.	107
6.5	Consider using bar charts instead of pies.	107
6.6	For long labels, use horizontal bar charts.	108
6.7	For long labels, use horizontal bar charts.	108
6.8	Don't use colors just for the sake of it.	110
6.9	Make sure important things catch the eye first.	111
6.10	Grouped column chart with data from StateOfObesity.org. Explore the full-screen interactive version.	112
6.11	Make your own copy of the Google Sheet template.	112
6.12	Float cursor in top-right corner of the chart to make the 3-dot (kebab) menu appear, and select Delete.	113
6.13	You should be able to distinguish kebab from hamburger menu icons.	113
6.14	Format data in columns to make colored grouped columns in your chart.	113
6.15	Select your data and then Insert the Chart.	114
6.16	Change the default to Column chart, with Stacking none.	114
6.17	Select Customize to edit title, labels, and more.	115
6.18	Click the Share button and then click <i>Change to anyone with the link</i> to make your data public.	115
6.19	Select Publish Chart to embed an interactive chart on another web page.	116
6.20	Separated bar chart with data from Starbucks and McDonalds. Explore the full-screen interactive version.	117
6.21	Create a separated column or bar chart by leaving some cells blank.	117
6.22	Stacked column chart with data from WHO and CDC. Explore the full-screen interactive version.	118
6.23	Create a stacked column or bar chart by structuring your data as shown.	119
6.24	Histogram chart with fictitious source data. Explore the full-screen interactive version.	120

6.25 Pie chart with fictitious source data. Explore the full-screen interactive version.	121
6.26 Line chart showing meat availability per capita in the US, according to the US Department of Agriculture. Explore the full-screen interactive version.	122
6.27 Data for the line chart shown in Figure 6.26.	123
6.28 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.	124
6.29 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.	125
6.30 Data for a scatterplot is usually represented in 3 columns: x-values, y-values, and labels.	125
6.31 In the chart's Setup window, choose <i>Add labels</i> to the Series.	126
6.32 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.	127
6.33 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.	128
6.34 Bubble chart data. Bubble size represents population, color – region.	129
6.35 This scatterplot is made in Tableau Public an shows the relationship between household income and test scores in Connecticut school districts.	130
6.36 Drag data sheet into <i>Drag tables here</i> area.	131
6.37 Drag data fields to the right places in Tableau.	132
6.38 This scatter chart is ready to be published.	133
6.39 This scatter chart is ready to be published.	134
6.40 Internet Access by Country, 1995–2018.	135
6.41 After you drag Country Name to the Filters card, make sure the Filter is displayed.	136
6.42 This workbook is ready to be published.	137
7.1 Map elements.	142

7.2 Examples of sequential, diverging, and qualitative color schemes from ColorBrewer.	142
7.3 Representing per capita income in US states using three different classifications.	144
7.4 ColorBrewer interface.	145
7.5 A map of airports in Nigeria built using Google My Maps.	146
7.6 A map of airports in Nigeria built using Google My Maps.	146
7.7 Add title and description to your map.	147
7.8 Check LATITUDE and LONGITUDE as your location columns.	148
7.9 Choose City as the title for your markers.	148
7.10 In My Maps, you can change marker colors and icons.	150
7.11 Make sure anyone with the link can view your map before you share it.	151
7.12 My Maps can generate an iframe code to include the map on your own website.	152
7.13 This choropleth map is created in Datawrapper	153
7.14 Sign in to Datawrapper, click <i>New Map</i> , and choose <i>Choropleth</i>	153
7.15 To map home prices by US state, choose appropriate boundaries.	154
7.16 Tell Datawrapper which column contains geography names (ISO-Codes of states).	155
7.17 Choose red/blues divergent color scheme, and make sure to match median with the neutral (middle) value.	156
7.18 To reference values from the spreadsheet, add column names in double curly brackets.	157
7.19 In this tutorial, we will build a point map of hospitals in Texas using Socrata.	158
7.20 Go to Visualize > Launch New Visualization.	159
7.21 Select Texas as the only value for State field.	160
7.22 To show individual points instead of clusters, set Stop Clustering at Zoom Level to 1.	161
7.23 Let's display different types of hospitals in different colors.	162
7.24 To edit tooltip information, use Flyout Details menu item.	163
7.25 Click <i>Share</i> button to bring up <i>Share and Embed</i> window.	164
7.26 Copy iframe code to embed this map in another website.	165

7.27 This choropleth map is made in Tableau Public	166
7.28 When finished inspecting the connected file, go to Sheet 1.	166
7.29 Drag and drop variables to the right places.	167
7.30 Filter values to remove countries with no data from display. . . .	168
7.31 Change the color scheme to Reds with 5 steps.	169
7.32 Change tooltip text to make it more user-friendly.	170
8.1 Screenshot: Drop-down menu to publish a Google Sheets chart .	175
8.2 Screenshot: Publish to the web for a Google Sheets chart	175
8.3 Screenshot: Copy the iframe code from a Google Sheets chart .	176
8.4 Screenshot: Edit and Share buttons in Tableau Public web page	183
8.5 Screenshot: Edit and Share buttons in Tableau Public web page	184
9.1 Create your own version of this simple interactive Leaflet map. .	189
9.2 Click <i>Use this template</i> to make your own copy.	190
9.3 Name your copied repo <code>leaflet-map-simple</code>	190
9.4 Click the index.html file to view the code.	191
9.5 Click the pencil button to edit the code.	192
9.6 Click the green <i>Commit Changes</i> button to save your edits. .	192
9.7 Click the <i>Settings</i> button to access GitHub Pages and publish your work on the web.	193
9.8 In <i>Settings</i> , go to <i>GitHub Pages</i> , and switch the source from <i>None</i> to <i>Master</i>	194
9.9 In <i>Settings</i> for <i>GitHub Pages</i> , right-click your published map link to open in a new tab.	194
9.10 On your first browser tab, click the repo title.	195
9.11 Open and edit the <code>README</code> file to paste the link to your live map.	196
9.12 Click <i>Code</i> and select <i>Download Zip</i> to create a compressed folder of a repo on your computer.	199
9.13 Click the plus (+) symbol in upper-right corner to create a new repo.	199
9.14 Name your new repo <i>practice</i> , check the box to <i>Initialize this repo</i> <i>with a README</i> , and <i>Add a license</i> (select <i>MIT</i>) to match any code you plan to upload.	200

9.15 Click the <i>Upload Files</i> button.	201
9.16 Drag-and-drop the <code>index.html</code> file to the upload screen.	201
9.17 After clicking the Delete Repository button, GitHub will ask you to type your username and repo name to confirm.	202
9.18 In your GitHub repo on the web, click <i>Add file</i> to <i>Open with GitHub Desktop</i> to download and install GitHub Desktop.	203
9.19 Click the blue <i>Sign in to GitHub.com</i> button to link GitHub Desktop to your GitHub account.	204
9.20 Click the <i>Continue</i> button to authorize GitHub Desktop to send commits to your GitHub account.	204
9.21 Select your <i>leaflet-map-simple</i> repo and click the <i>Clone</i> button to copy it to your local computer.	204
9.22 Select the Local Path where your repo will be stored on your computer, then click <i>Clone</i>	205
9.23 If asked how you plan to use this fork, select the default <i>To contribute to the parent project</i> and click <i>Continue</i>	206
9.24 Now you have two copies of your repo: in your GitHub online account (on the left) and on your local computer (on the right, as shown in the Mac Finder). Windows screens will look different.	206
9.25 In GitHub Desktop, confirm the Current Repo and click the <i>Open in Atom</i> button to edit the code.	207
9.26 Atom Editor opens your repo as a <i>project</i> , where you can click files to view code. Edit your map title.	207
9.27 To clean up your Atom Editor workspace, right-click to <i>Remove Project Folder</i>	208
9.28 Right-click the <code>index.html</code> file on your local computer and open with a browser to check your edits.	208
9.29 In this two-step process, click <i>Commit to Master</i> , then click <i>Push origin</i> to save and copy your edits from your local computer to your GitHub web account, as shown in this animated GIF.	209
9.30 Drag-and-drop the file to the upload screen.	210
11.1 Explore a live demonstration of Leaflet Maps with Google Sheets.	222
11.2 Explore the live Google Sheet template that feeds data into the Leaflet map above.	223
11.3 In <i>Settings</i> , go to <i>GitHub Pages</i> , and switch the source from <i>None</i> to <i>Master</i>	225

11.4 Copy the Google Sheet address at the top of the browser, NOT the <i>Publish to the web</i> address.	227
11.5 Paste in <i>your</i> Google Sheet URL to replace <i>our</i> URL.	228
11.6 To use our built-in Geocoder menu, shift-click to select all columns from Location through Source.	230
11.7 One option is to display a table of viewable markers at the bottom of your map.	230
11.8 One way to finalize your map is to download each Google Sheets tab as a CSV file.	234
11.9 Screenshot: File > Publish the link to your Google Sheet	237
11.10 Screenshot: Copy the Google Sheet URL, not the Publish URL . .	237
11.11 Select <i>Create a Project</i> or use the menu to select a new project. .	241
11.12 Give your project a meaningful short name.	242
11.13 Press the + <i>Enable APIs and Services</i> button.	243
11.14 Search for <i>Google Sheets</i> and select this result.	243
11.15 Select the <i>Enable</i> button for Google Sheets API.	244
11.16 Select <i>Credentials - Create Credentials - API key</i>	244
11.17 Copy your API key and press <i>Restrict key</i>	245
11.18 Choose <i>API restrictions - Restrict key - Google Sheets API</i> . .	246
11.19 Paste in <i>your</i> Google Sheets API key to replace <i>our</i> key.	247
11.20 Police Incidents dataset on Hartford Open Data portal	253
11.21 Formatted JSON example in Firefox	254
11.22 Unformatted JSON example in Firefox	254
11.23 Formatted GeoJSON example in Firefox	255
11.24 Screenshot: Sample API endpoint in Socrata open data repo . .	257
11.25 Screenshot: API endpoint with .geojson suffix in Chrome browser	258
12.1 To map addresses, you need to geocode them first.	259
12.2 To geocode one address, search in Google Maps and right-click <i>What's here?</i> to show coordinates.	261
12.3 Put addresses in the first column, and use Geocoder to fill in the remaining five.	262
12.4 Put addresses in the first column, and use Geocoder to fill in the remaining seven.	263

12.5 You can count addresses by state (or other area) to produce polygon, or choropleth, maps instead of point maps.	265
12.6 In Socrata, you can export the entire dataset as a CSV.	266
12.7 Use pivot tables in any spreadsheet software to count addresses per area (such as state, county, or zip code).	267
12.8 Choropleth maps work best with normalized values.	268
12.9 Geospatial data can be raster or vector.	270
12.10 GitHub can show previews of GeoJSON files stored in repositories.	272
12.11 GeoJson.io successfully imported Hartford parks KML file. . . .	274
12.12A spreadsheet with lat/lon columns can be transformed into a GeoJSON with point features.	275
12.13 Use drawing tools to create points, lines, and polygons in GeoJson.io.	276
12.14 You can use Mapshaper to quickly convert between geospatial file formats.	279
12.15 Use <i>edit attributes</i> tool (under Cursor tool) to edit attributes of polygons, lines, and points.	280
12.16 Consider simplifying geometries with Mapshaper to make your web maps faster.	281
12.17 Use Simplify & Repair tools in Mapshaper.	282
12.18 Mapshaper lets you dissolve boundaries to create an outline shape.	282
12.19 When clipping, make sure your active layer is the one being clipped (with many features), not the clipping feature itself. . . .	284
12.20 Mapshaper lets you join spatial and CSV files using common keys (for example, town names).	286
12.21 Mapshaper's -join can count points in polygons.	288
12.22 Create a two-column crosswalk of towns and which districts they should be merged to.	290
12.23 Merge polygons based on a predefined crosswalk.	291
12.24 In Google Earth Pro, right-click the KMZ layer and choose <i>Save Place As</i>	293
12.25 Save as KML, not KMZ.	293
13.1 Screenshot: Edit the Min and Max values of the Y-axis	296

A.1	On a Mac, go to <i>Finder</i> then <i>Preferences</i> then <i>Advanced</i> and check the box to <i>Show all filename extensions</i>	302
A.2	Screenshot: Replace http with https	303
A.3	Screenshot: Curly quotes versus straight quotes	304
A.4	Screenshot: Extra files in GitHub repo will block iframe in your README	307
A.5	Screenshot: User accidentally renamed column headers in the Points tab	308
A.6	Screenshot: Do not rename or delete	309

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: 18 Aug 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, Datawrapper, and Tableau Public. You'll also gradually learn how to edit open-source code templates like Chart.js, Highcharts, and Leaflet on GitHub. Follow along with the step-by-step tutorials, real-world examples, and online resources. This book is ideal for students, non-profit organizations, small business owners, local governments, journalists, academics, or anyone who wants to tell their story and show the data. No coding experience is required.

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>.

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

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

Acknowledgements

An earlier draft 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 students who worked on the earlier draft 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...

We appreciate everyone who provided feedback to improve this book, especially Amelia Blevins, our developmental editor at O'Reilly, and technical reviewers Carl Allchin,...

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.

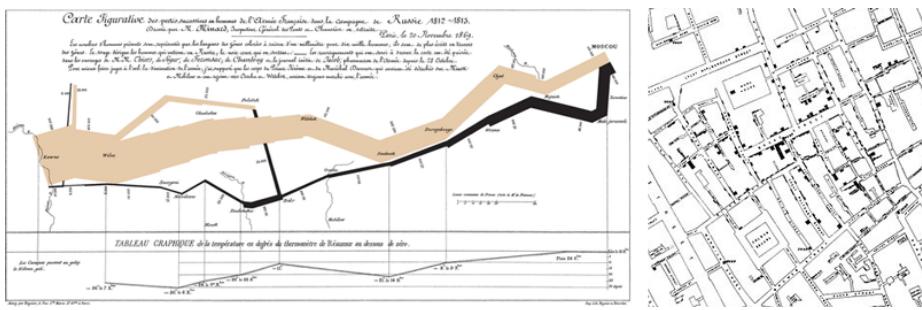


Figure 1: Images: Menard's figurative map (left) and Snow's dot map (right), from Wikimedia

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.

- 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 Haeblerly 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.

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

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

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

How to Read on the Web

TODO: use conditional formatting to make this section appear only in the HTML edition; may need to convert to a free-standing chapter

This open-access book-in-progress is free to read on the web 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 (laptops or desktops) over smaller screens (such as smartphones or tablets). 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

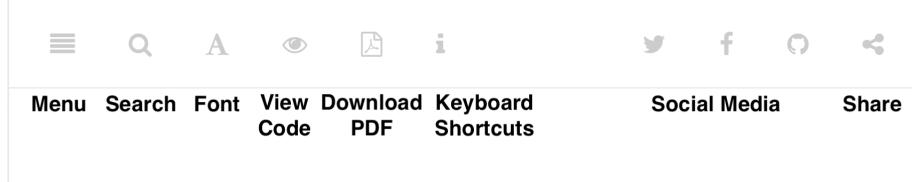


Figure 2: Screenshot: How to read

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)

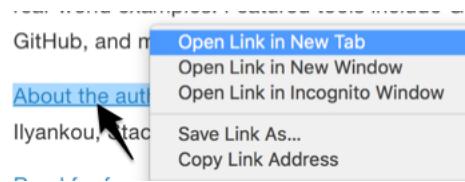


Figure 3: Screenshot: Open link in new tab (on Mac)

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.

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)

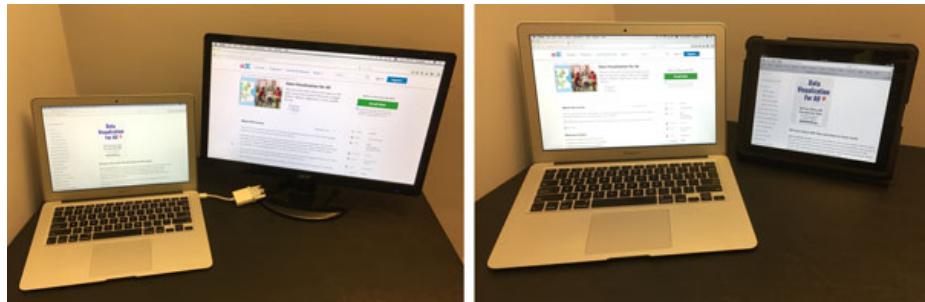


Figure 4: Image: Laptop with second monitor, and with tablet

Chapter 1

Choose Tools to Tell Your Data Story

TODO: Reorganize and rewrite chapter Revise title? – Choose Tools to Draw Your Data Story

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

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

Once you have a clearer mental (and physical) picture of what you seek to create, then choose digital tools...

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.

TODO: add password manager tutorial to keep track of your accounts for the online tools you'll use in this book. The free and open-source BitWarden.com tool nicely integrates with most browsers and even smartphones.

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. [Use some of this from ch 3? – Start the process by writing down your question—literally in the form of a question, punctuated with a question mark—to clarify your own thinking, and also so that you can clearly communicate it to others who may be assisting you. All too often, our brains automatically jump ahead to try to identify the *answer*, without reflecting on the best way frame the *question* in a way that does not limit the range of possible results.]
- 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.

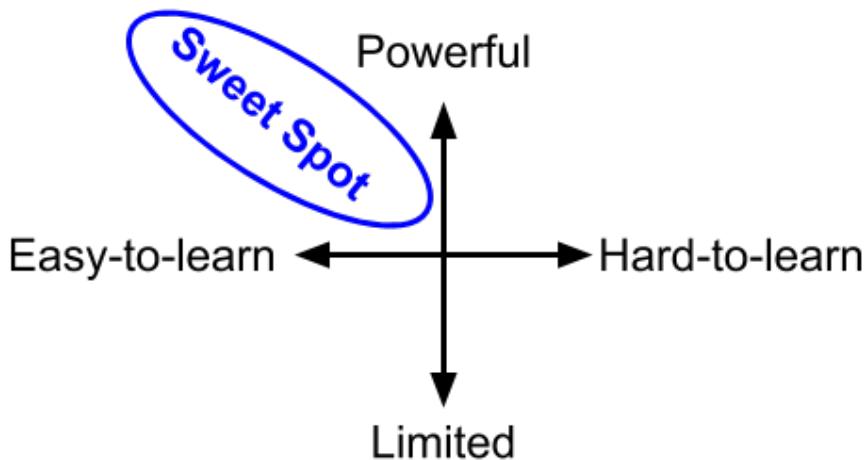


Figure 1.1: Diagram: the 'sweet spot' for easy-to-learn and powerful tools

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.

TODO: expand on privacy to review sample “terms of service” to use free tools such as Google Drive - <https://support.google.com/drive/answer/2450387?hl=en#:~:text=As%20our%20Terms%20of%20Service,store%20in%20your%20Drive%20account.>
- See alimSpyingStudentsSchool2017 - Many of the free web-based tools in this book require that you publicly share your data. Check each tool and decide whether it is appropriate for your data, which may have some privacy restrictions.

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.

Summary

TODO

Chapter 2

Strengthen Your Spreadsheet Skills

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

The interactive charts and maps that we'll construct later this book are built on data tables, which we typically open with spreadsheet tools, such as Google Sheets, LibreOffice, or Microsoft Excel. Spreadsheets typically contain columns and rows of numerical or textual data, as shown in Figure 2.1. The first row often contains headers, meaning labels describing the data in each column. Also, columns are automatically labeled with letters, and rows with numbers, so that every cell or box in the grid can be referenced, such as C2. When you click on a cell, it may display a formula that automatically runs a calculation with references other cells. Formulas always begin with an equal sign, and may simply add up other cells (such as `=C2+C3+C4`), or may contain a function that performs a specific operation (such as calculating the average of a range of cells: `=average(C2:C7)`). Some spreadsheet files contain multiple sheets (sometimes called workbooks), where each tab across the bottom opens a specific sheet.

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

The screenshot shows a spreadsheet interface with a table of data. The table has columns labeled A through E. Row 1 contains the header labels: Name, Location, Experience, Years of school, and Occupation. Rows 2 through 7 contain data for five individuals: Jack, Anthony, Emily, Hayat, Ignacio, and Carly. The formula bar at the top shows '=average(C2:C7)'. The cell C8 contains the value '2.17', which is highlighted with a blue border and labeled 'Active cell (see formula at top)' in red text. The bottom navigation bar includes buttons for '+', '≡', 'data', 'notes', and 'Tabs for multiple sheets'.

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

+ ≡ data ▾ notes ▾ Tabs for multiple sheets

Figure 2.1: Screenshot of a typical spreadsheet, with headers, tabs, and the active cell displaying a formula.

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

Select your Spreadsheet Tools

Which spreadsheet tools should you use? As we describe in more detail in Chapter 1: Choose Tools to Tell Your Data Story, the answer depends on how you respond to different questions about your work. First, is your data public or private? If private, consider using a downloadable spreadsheet tool that runs on your computer, to reduce the risk of an accidental data breach that might happen when using an online spreadsheet tool that automatically stores your data in the cloud. Second, will you be working solo or with other people? For collaborative projects, consider using an online spreadsheet tool that's designed to allow other team members to simultaneously view or edit data. Third, do you need to import or export data in any specific format (which we'll describe in the next section), such as Comma Separated Values (CSV)? If yes, then choose a spreadsheet tool that supports that format. Finally, do you prefer a free tool, or are you willing to pay for it, or donate funds to support open-source development?

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

- Google Sheets is a free online spreadsheet tool that works in any modern web browser, and automatically stores your data in the cloud. While data you upload is private by default, you can choose to share it with specific

individuals or anyone on the internet, and allow them to view or edit for real-time collaboration, similar to Google Documents. Google Sheets also imports and exports data in CSV, ODS, Excel, and other formats. You can sign up for a free personal Google Drive account with the same username as your Google Mail account, or create a separate account under a new username to reduce Google's invasion into your private life. Another option is to pay for a Google Suite business account subscription, which offers nearly identical tools, but with sharing settings designed for larger organizations or educational institutions.

- LibreOffice is a free downloadable suite of tools, including its Calc spreadsheet, available for Mac, Windows, and Linux computers, and is an increasingly popular alternative to Microsoft Office. When you download LibreOffice, its sponsor organization, The Document Foundation, requests a donation to continue its open-source software development. The Calc spreadsheet tool imports and exports data in its native ODS format, as well as CSV, Excel, and others. While an online collaborative platform is under development, it is not yet available for broad usage.
- Microsoft Excel is the spreadsheet tool in the Microsoft Office suite, which is available in different versions, though commonly confused as the company has changed its product names over time. A paid subscription to Microsoft 365 provides you with two versions: the full-featured downloadable version of Excel (which is what most people mean when they simply say "Excel") for Windows or Mac computers and other devices, and access to a simpler online Excel through your browser, including file sharing with collaborators through Microsoft's online hosting service. If you do not wish to pay for a subscription, anyone can sign up for a free version of online Excel at Microsoft's Office on the Web, but this does *not* include the full-featured downloadable version. The online Excel tool has limitations. For example, neither the paid nor the free version of online Excel allows you to save files in the single-sheet generic Comma Separated Values (.csv) format, an important feature required by some data visualization tools in later chapters of this book. You can only export to CSV format using the downloadable Excel tool, which is now available only with a paid Microsoft 365 subscription.

Deciding which spreadsheet tools to use is not a simple choice. Sometimes our decisions change from project to project, depending on costs, data formats, privacy concerns, and the personal preferences of any collaborators. Occasionally we've also had co-workers or clients specifically request that we send them non-sensitive spreadsheet data attached to an email, rather than sharing it through a spreadsheet tool platform that was designed for collaboration. So it's best to be familiar with all three commonly-used spreadsheet tools above, and to understand their respective strengths and weaknesses.

In this book, we primarily use Google Sheets for most of our examples. All of the data we distribute through this book is public. Also, we wanted a spreadsheet

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

Sidebar: Common data formats

Spreadsheet tools organize data in different formats. When you download spreadsheet data to your computer, you typically see its filename, followed by a period and a 3- or 4-character abbreviated extension, which represents the data format, as shown in Figure 2.2. The most common data formats we use in this book are:

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

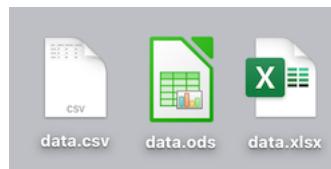


Figure 2.2: Three data formats commonly seen on your computer—`csv`, `ods`, and `xlsx`—when displayed properly in the Mac Finder.

Warning: Several tools in this book may not work properly on a Mac computer that does not display the filename extensions, meaning the abbreviated file format after the period, such as `data.csv` or `map.geojson`. The Mac operating system hides these by default, so you need to turn them on by going to `Finder > Preferences > Advanced`, and check the box to *Show all filename extensions*, as shown in Figure 2.3.

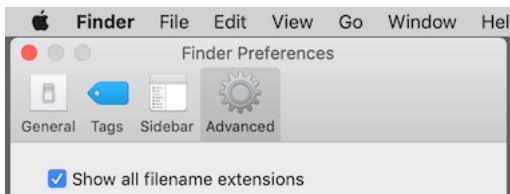


Figure 2.3: On a Mac, go to *Finder-Preferences-Advanced* and check the box to *Show all filename extensions*.

Download to CSV or ODS Format

In Chapter 1: Choose Tools to Tell Your Data Story, we learned the advantages of selecting software tools that support data migration, so that you can export your work to other platforms. Since digital technology is always changing, it's a good rule of thumb to never upload important data into a tool if you can't easily get it back out. Ideally, spreadsheet tools should allow you to export your work in generic or open-data file formats, such as Comma Separated Values (CSV) and OpenDocument Spreadsheet (ODS), to maximize your options to migrate to other platforms.

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

One reason we feature Google Sheets in this book is because it exports data in several common formats. To try it, open this Google Sheets sample data file in a new tab, and go to *File > Download As* to export in CSV format (for only the data in the active sheet) or ODS format (which keeps data and most formulas in multi-tab spreadsheets), or other formats such as Excel, as shown in Figure 2.4. Similarly, in the downloadable LibreOffice and its Calc spreadsheet tool, select *File > Save As* to save data in its native ODS format, or to export to CSV, Excel, or other formats.

But exporting data can be trickier in Microsoft Excel. Using the online Excel tool in your browser (either the free or paid version), you *cannot* save files in the generic single-sheet CSV format, a step required by some data visualization tools in later chapters of this book. Only the downloadable Excel tool (which now requires a paid subscription) will export in CSV format, a step required by some data visualization tools in later chapters of this book. And when using the downloadable Excel tool to save in CSV format, the steps sometimes confuse people. First, if you see multiple CSV options, choose *CSV UTF-8*, which

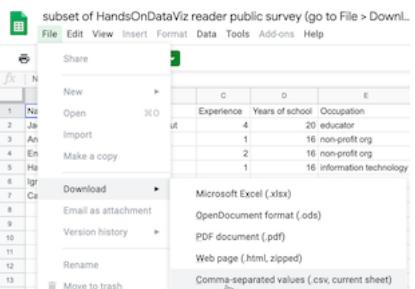


Figure 2.4: In Google Sheets, go to *File - Download As* to export data in several common formats.

should work best across different computer platforms. Second, if your Excel workbook contains multiple sheets or formulas, you may see a warning that it cannot be saved in CSV format, which only saves data (not formulas) contained in the active sheet (not all sheets). If you understand this, click *OK* to continue. Third, on the next screen, Excel may warn you about “Possible data loss” when saving an Excel file in CSV format, for reasons described above. Overall, when working with the downloadable Excel tool, first save the full-version of your Excel file in XLSX format before exporting a single sheet in CSV format.

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

Make a Copy of a Google Sheet

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

Let’s begin by making a copy of a real dataset that may interest you, because it includes people like you. So far about 3,000 readers of this book have responded to a quick public survey about their general location, prior level of experience and education, and goals for learning data visualization. If you haven’t already done so, fill out the quick survey form to contribute your own response, and also to give you a better sense of how the questions were posed. Later in this chapter you’ll learn how to create your own online form with a link to spreadsheet results.

1. Open this Google Sheet of Hands-On Data Visualization reader public

survey responses in a new tab in your browser. We set it to “View only” so that anyone on the internet can see the contents, but not edit the original file.

2. Sign in to your Google account by clicking the blue button in the upper-right corner.
3. Go to *File > Make a Copy* to create a duplicate of this Google Sheet in your Google Drive, as shown in Figure 2.5. You can rename the file to remove “Copy of...”.

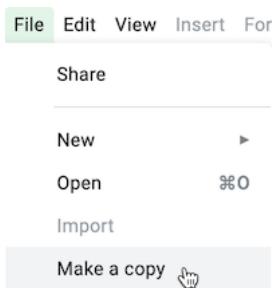


Figure 2.5: Go to *File - Make a Copy* to create your own version of this Google Sheet.

4. To keep your Google Drive files organized, save them in folders with relevant names to make them easier to find. For example, you can click the *My Drive* button and the *New folder* button to create a folder for your data, before clicking *OK*, as shown in Figure 2.6.

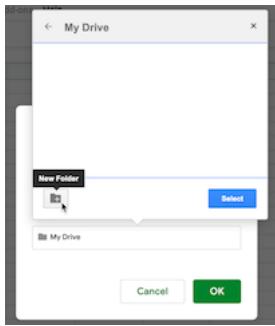


Figure 2.6: Click the *My Drive* and *New folder* buttons to save your work in a folder.

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

Share Your Google Sheets

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

1. Log into your Google Drive account, click the *New* button, select *Google Sheets*, and create a blank spreadsheet. You will need to name your file to proceed with next steps.
2. Click the *Share* button in the upper-right corner, and your options will appear on the *Share with people and groups* screen, as shown in Figure 2.7.
3. In the top half of the screen, you can share access with specific individuals by typing their Google usernames into the *Add people and groups* field. For each person or group you add, on the next screen select the drop-down menu to choose if they can *View*, *Comment* on, or *Edit* the file. Decide if you wish to notify them with a link to the file and optional message.
4. In the lower half of the screen, you can share access more widely by clicking on *Change to anyone with the link*. On the next screen, the default option is to allow anyone who has the link to *View* the file, but you can change this to allow anyone to *Comment* on or *Edit* it. Also, you can click *Copy link* to paste the web address to your data in an email or public website.

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

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

then use a free link-shortening service. For example, by using our free Bitly.com account and its handy Chrome browser extension or Firefox browser extension, we can paste in a long URL and customize the back-end to something shorter, such as bit.ly/reader-survey, as shown in Figure 2.8. If someone else has already claimed your preferred custom name, you'll need to think up a different one. Beware that `bit.ly` links are case-sensitive, so we prefer to customize the back-end in all lower-case to match the front-end.

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

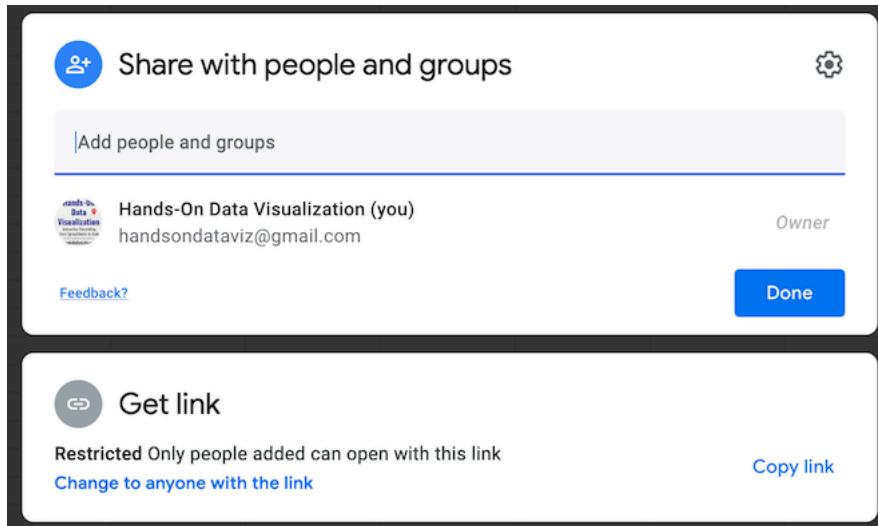


Figure 2.7: Click the *Share* button to grant access to individuals (top half) or anyone with the link (bottom half).

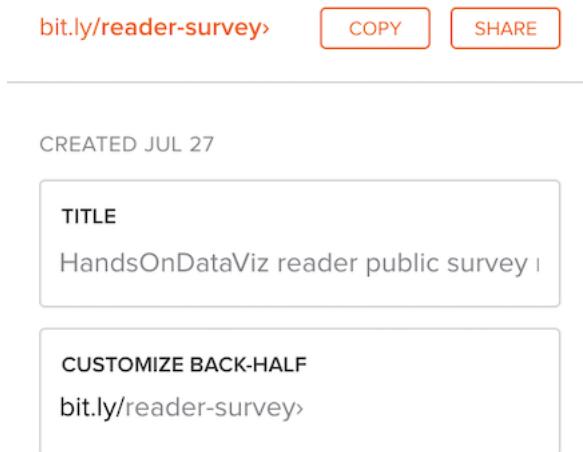


Figure 2.8: Use a free link-shortening service, such as Bitly.com, and customize its back-end.

Upload and Convert to Google Sheets

We feature Google Sheets in this book partly because it supports data migration, meaning the ability to import and export files in many common formats. But imports work best when you check the *Convert uploads* box, which is hidden inside the Google Drive Settings gear symbol as shown in Figure 2.9. Checking this box automatically transforms Microsoft Excel sheets into Google Sheets format (and also Microsoft Word and PowerPoint files into Google Documents and Slides formats), which allows easier editing. If you don't check this box, then Google will keep your files in their original format, which makes them harder to edit. Google turns off this conversion setting by default on new accounts, but we'll teach you how to turn it on, and the benefits of doing so.

1. Find a sample Excel file you can use on your computer. If you don't have one, open and save to download to your computer this Excel file of a subset of the Hands-On Data Visualization reader public survey responses
2. Log into your Google Drive account, and click the *Gear symbol* in the upper-right corner, as shown in Figure 2.9, to open the Settings screen. Note that this global *Gear symbol > Settings* appears at Google Drive level, *not* inside each Google Sheet.

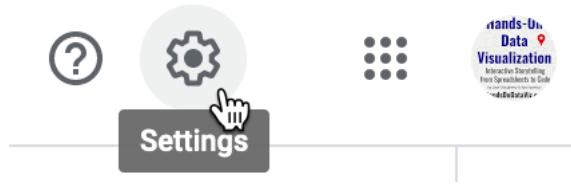


Figure 2.9: Click your Google Drive *Gear Symbol - Settings* in the upper-right corner.

3. On the Settings screen, check the box to *Convert uploaded files to Google Docs editor format*, as shown in Figure 2.10, and click *Done*. This turns on the conversion setting globally, meaning it will convert all possible files that you upload in the future—including Microsoft Excel, Word, PowerPoint, and more—unless you turn it off.
4. Upload a sample Excel file from your computer to your Google Drive. Either drag-and-drop it to the desired folder, as shown in Figure 2.11, or use the *New* button and select *File upload*.

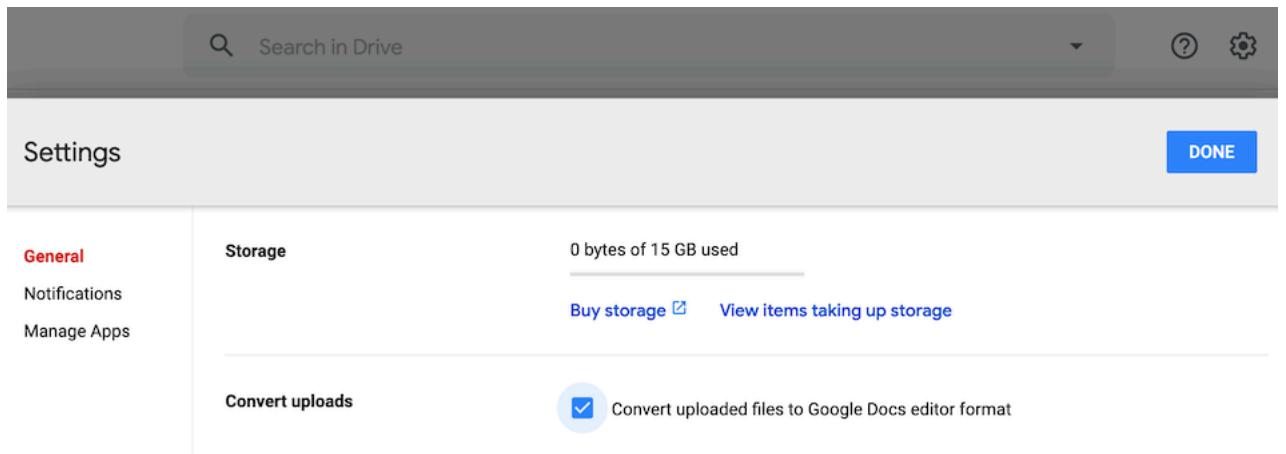


Figure 2.10: Inside your Google Drive Settings, check the box to automatically convert all uploads.

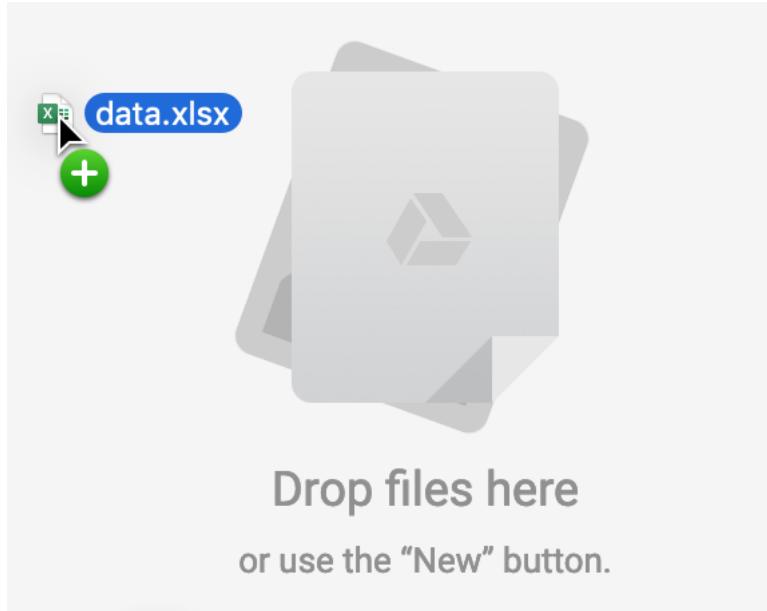


Figure 2.11: Drag-and-drop your sample Excel file into your Google Drive to upload it.

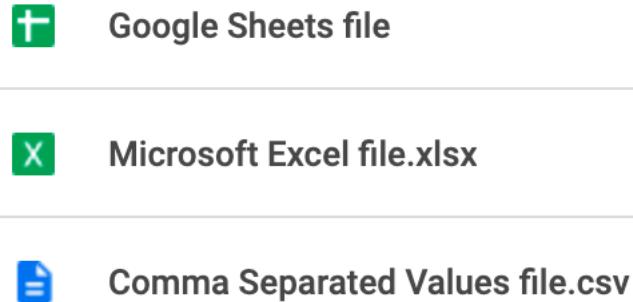


Figure 2.12: If you forget to convert uploads, Google Drive will keep files in their original format with these icons.

If you forget to check the *Convert uploads* box, Google Drive will keep uploaded files in their original format, and display their icons and file name extensions such as `.xlsx` or `.csv`, as shown in Figure 2.12.

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

Now that you know how to upload and convert an existing dataset, in the next section you will learn how to collect data using an online form, and access it as a spreadsheet.

Collect Data with Google Forms

As you saw in prior sections, we invite readers of this book to fill out a quick online form so that we can learn more about people like you, and to continue to make revisions to match your expectations. So far about 3,000 readers have responded, and you can view this public spreadsheet of survey responses about their generation location, prior level of experience and education, and goals for learning data visualization. In this section, you'll learn how to create an online form and link the results to a live Google Sheet.

Inside your Google Drive account, one tool that's often overlooked is Google

Forms, which is partially hidden under *New > More > Google Forms*, as shown in Figure 2.13.

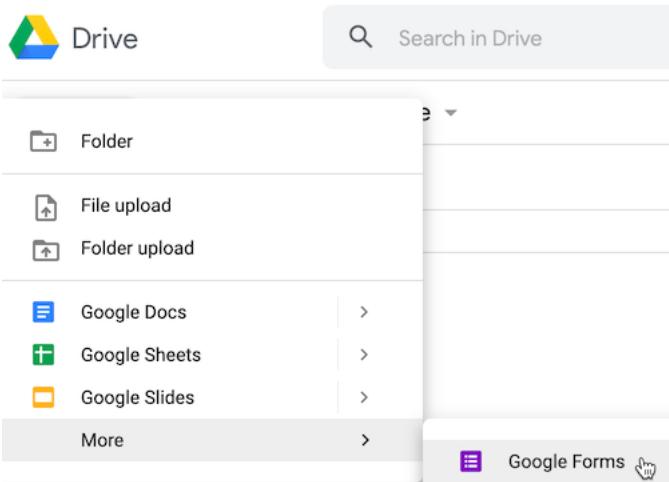


Figure 2.13: The Google Forms tool is partially hidden in the Google Drive *New - More* menu.

The Google Forms *Questions* tab allows you to design questions with different types of responses: short- and paragraph-length answers, multiple choice, checkboxes, file uploads, etc., as shown in Figure 2.14.

Give each question a very short title, since these will appear as column headers in the linked spreadsheet you'll create further below. If a question needs more explanation or examples, click the three-dot kebab menu in the bottom-right corner to *Show > Description*, which opens a text box where you can type in more details, as shown in Figure 2.15. Also, you can *Show > Response validation*, which requires users to follow a particular format, such as an email address or phone number.

To preview how your online will appear to recipients, click the *Eyeball symbol* near the top of the page, as shown in Figure 2.16. When your form is complete, click the *Send* button to distribute it via email, a link, or to embed the live form as an iframe on a web page. Learn more about the latter option in Chapter 8: Embed On Your Web.

The Google Forms *Responses* tab will show individual results you receive, and also includes a powerful button to open the data in a linked Google Sheet, as shown in Figure 2.17.

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

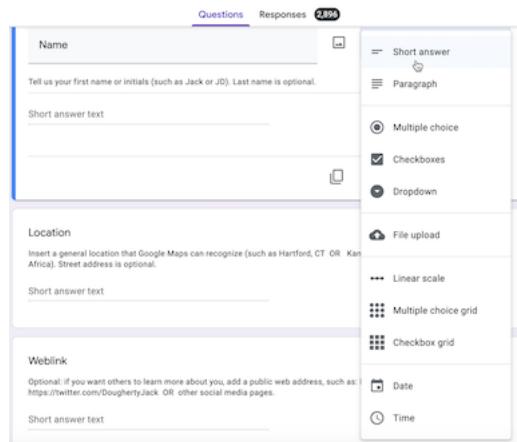


Figure 2.14: The Google Forms *Questions* tab allows you to designate different types of responses.

Name

Tell us your first name or initials (such as Jack or JD). Last name is optional.

Short answer text

Location

Insert a general location that Google Maps can recognize (such as Hartford, CT OR Kansas, USA OR Cape Town, South Africa). Street address is optional.

Show Description Response validation

Figure 2.15: Click the three-dot kebab menu to *Show - Description* to add details for any question.

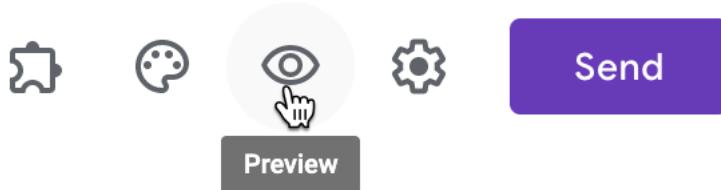


Figure 2.16: Click the *Eyeball symbol* to preview your form.

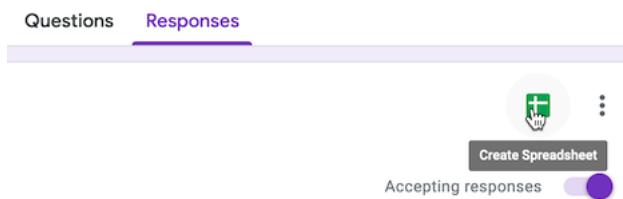


Figure 2.17: The Google Forms *Responses* tab includes a button to open results in a linked Google Sheet.

Sort and Filter Data

Spreadsheet tools help you delve into your data and lift its stories to the surface. A basic step in organizing your data is to *sort* a table by a particular column, to quickly view its minimum and maximum values, and the range that lies in between. A related method is to *filter* an entire table to display only rows that contain certain values, to help them stand out for further study among all of the other entries. Both of these methods become more powerful when your spreadsheets contain hundreds or thousands of rows of data.

To learn how to sort and filter, let's explore a large dataset of around 3,000 readers of this book who responded to a quick public survey about their general location, prior level of experience and education, and goals for learning data visualization. If you haven't already done so, fill out the quick survey form to contribute your own response, and also to give you a better sense of how the questions were posed.

1. Open this Google Sheet of Hands-On Data Visualization reader public survey responses in a new tab in your browser.
2. Login to your Google Sheets account, and go to *File > Make a Copy* to create your own version that you can edit.
3. Before sorting, click the upper-left corner of the sheet to select all cells, as

shown in Figure 2.18. The entire sheet should become light blue to show you've selected all cells.

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

Figure 2.18: Click the upper-left corner to select all cells before sorting.

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

4. Go to *Data > Sort Range* to review all of your sort options. In the next screen, check the *Data has header row* box to view the column headers in your data. Let's sort the *Experience with data visualization* column in ascending order (from A-Z), as shown in Figure 2.19, to display the minimum at the top, the maximum at the bottom, and the range in between.

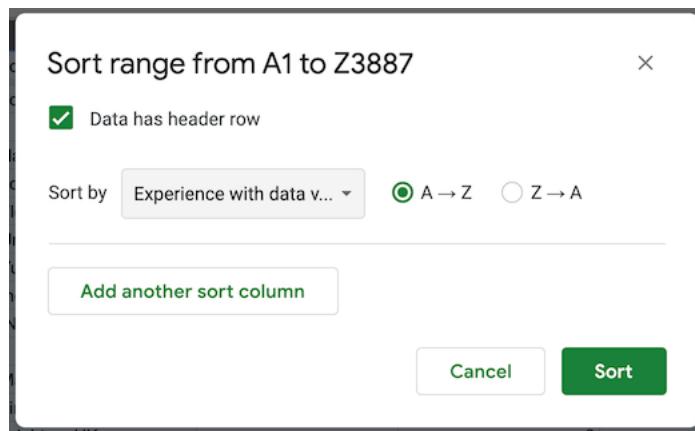


Figure 2.19: Go to *Data - Sort Range*, check the header row box, and sort by *Experience with dataviz* in ascending order.

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

Tip: When working with large spreadsheets, you can “freeze” the first row so that column headers will still appear as you scroll downward. In Google Sheets, go to *View > Freeze* and select 1 row, as shown in Figure 2.20. You can also freeze one or more columns to continuously display when scrolling sideways. LibreOffice has a same option to *View > Freeze Rows and Columns*, but Excel has a different option called *Window > Split*.

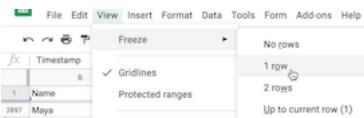


Figure 2.20: In Google Sheets, go to *View - Freeze* to select the number of rows to continuously display when scrolling downward.

- Now let's try filtering your sheet. Go to *Data > Create a Filter*, which inserts downward arrows in each column header. Click on the downward arrow in the *Occupation* column, and see options to display or hide rows of data. For example, click the “Clear” button to undo all options, then click only *educator* to display only rows with that response, as shown in Figure 2.21. Click “OK”.

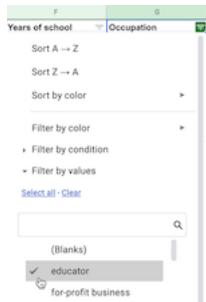


Figure 2.21: Go to *Data - Create a Filter*, click the downward arrow in the *Occupation* column, select only *educator*.

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

Calculate with Formulas

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

Let's explore a large dataset of around 3,000 readers of this book who responded to a quick public survey about their general location, prior level of experience and education, and goals for learning data visualization. If you haven't already done so, fill out the quick survey form to contribute your own response, and also to give you a better sense of how the questions were posed.

1. Open this Google Sheet of Hands-On Data Visualization reader public survey responses in a new tab in your browser.
2. Log into your Google Drive account, and go to *File > Make a Copy* to edit your own version.
3. Add a blank row immediately below the header to make space for our calculations. Right-click on row number 1 and select *Insert 1 below* to add a new row, as shown in Figure 2.22.

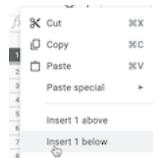


Figure 2.22: Right-click on row number 1 and select *Insert 1 below*.

4. Let's calculate the average level of reader experience with data visualization. Click on cell E2 in the new blank row you just created, and type an equal symbol (=) to start a formula. Google Sheets will automatically suggest possible formulas based on the context, and you can select one that displays the average for current values in the column, such as `=AVERAGE(E3:E2894)`, then press *Return* or *Enter* on your keyboard, as shown in Figure 2.23.

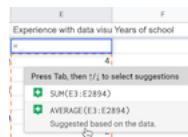


Figure 2.23: Type `=` to start a formula and select the suggestion for average, or type it directly in with the correct range.

Since our live spreadsheet has a growing number of survey responses, you will have a larger number in the last cell reference to include all of the entries in your version. Currently, the average level of reader experience with data visualization is around 2 on a scale from 1 (beginner) to 5 (professional), but this may change

as more readers fill out the survey. Note that if any readers leave this question blank, spreadsheet tools ignore empty cells when performing calculations.

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

5. Part of the magic of spreadsheets is that you can use the built-in hold-and-drag feature to copy and paste a formula across other columns or rows, and it will automatically update its cell references. Click in cell E2, and then press and hold down on the blue dot in the bottom-right corner of that cell, which transforms your cursor into a crosshair symbol. Drag your cursor to cell F2 and let go, and show in Figure 2.24. The formula will be automatically pasted and updated for the new column to `=AVERAGE(F3:F2894)` or `AVERAGE(F3:F)`, depending on which way you entered it above. Once again, since this is a live spreadsheet with a growing number of responses, your sheet will have a larger number in the last cell reference.

E	F	Occ
Experience with data	Years of school	Occ
2.09090091	4	20 edu
Experience with data	Years of school	Occ
2.09090091	4	20 edu
Experience with data	Years of school	Occ
2.09090091	17.80737062	stu

Figure 2.24: Click on the blue bottom-right dot in cell E2, then hold-and-drag your crosshair cursor in cell F2, and let go to automatically paste and update the formula.

6. Since the *Occupation* column contains a defined set of text responses, let's use a different function to count them using an *if statement*, such as the number of responses if a reader listed “educator”. Click in cell G2 and type the equal symbol (=) to start a new formula. Google Sheets will automatically suggest possible formulas based on the context, and you can select one that displays the count if the response is *educator* for current values in the entire column. You can directly type in the formula `=COUNTIF(G3:G2894, "=educator")`, where your last cell reference will be a larger number to reflect all of the rows in your version, or type in the Google Sheets syntax `=COUNTIF(G3:G, "=educator")` that runs the calculation on the entire column without naming a specific endpoint, as shown in Figure 2.25.

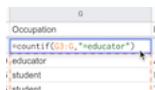


Figure 2.25: Select or enter a formula that counts responses if the entry is *educator*.

Spreadsheet tools contain many more functions to perform numerical calculations and also to modify text. Read more about functions in this support pages for Google Sheets, LibreOffice, or Microsoft Excel support page. See additional spreadsheet skills in later chapters of the book. Chapter 4: Clean Up Messy Data demonstrates how to find and replace, split data into columns, and combine columns of data (such as when you need the street address, city, and postal code all in one line). Chapter 12: Transform Your Map Data also features more advanced spreadsheet skills and tools, such as how to geocode addresses, pivot address points into polygons, and how to normalize data to create more meaningful polygon maps.

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

Summarize Data with Pivot Tables

Pivot tables are another powerful feature built into spreadsheet tools to help you reorganize your data and summarize it in a new way, hence the name “pivot.” Yet pivot tables are often overlooked by people who were never taught about them, or have not yet discovered how to use them. In this section, we’ll start with a large dataset of around 3,000 readers of this book who responded to a quick public survey. Each row represents an individual reader, including their occupation and prior level of experience with data visualization. You’ll learn how to “pivot” this individual-level data into a new table that displays the total number of reader responses by two categories: occupation and experience level.

1. Open this Google Sheet of Hands-On Data Visualization reader public survey responses in a new tab in your browser. Log into your Google Drive account, and go to *File > Make a Copy* to edit your own version.
2. Or, if you have already created your own copy for the prior section on Formulas and Functions, delete row 2 that contains our calculations, because we don’t want those getting mixed into our pivot table.
3. Go to *Data > Pivot Table*, and on the next screen, select *Create* in a new sheet, as shown in Figure 2.26. The new sheet will include a Pivot Table tab at the bottom.

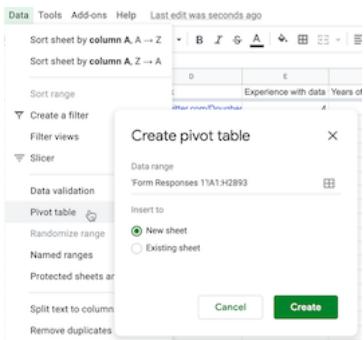


Figure 2.26: Go to *Data - Pivot Table*, and create in a new sheet.

4. In the *Pivot table editor* screen, you can regroup data from the first sheet by adding rows, columns, and values. First, click the Rows *Add* button and select *Occupation*, which displays the unique entries in that column, as shown in Figure 2.27.

Figure 2.27: In the *Pivot table editor*, click the Rows *Add* button and select *Occupation*.

5. Next, to count the number of responses for each entry, click the Values *Add* button and select *Occupation* again. Google Sheets will automatically summarize the values by *COUNTA*, meaning it displays the frequency of each textual response, as shown in Figure 2.28.

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

6. Furthermore, you can create a more advanced pivot cross-tabulation of occupation and experience among reader responses. Click on the *Columns* button to add *Experience with data visualization*, as shown in Figure 2.29.

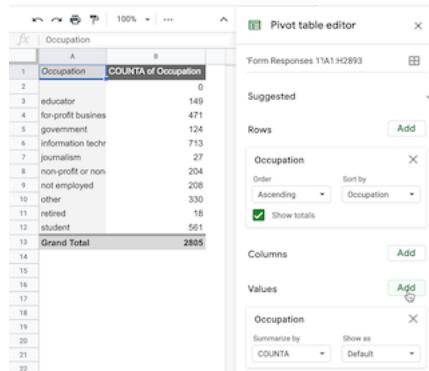


Figure 2.28: In the *Pivot table editor*, click the Values *Add* button and select *Occupation*.

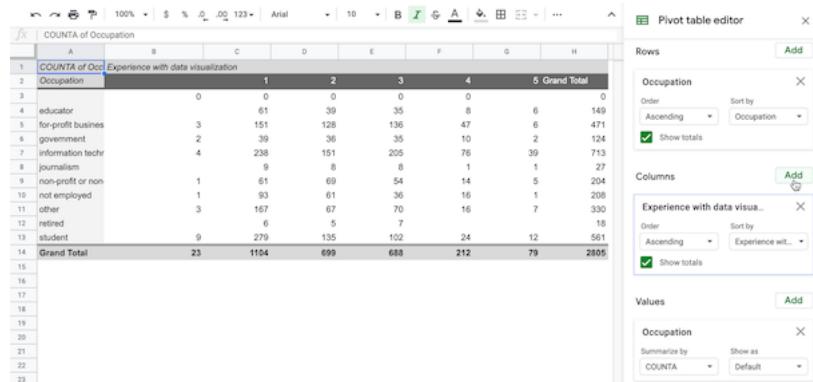


Figure 2.29: In the *Pivot table editor*, click the Columns *Add* button and select *Experience with data visualization*.

To go one step further, *Filter* the data to limit the pivot table results by another category. For example, you can click the Filters *Add* button and select *Years of school* to display only readers who listed 20 or more years.

Deciding how to add *Values* in the *Pivot table editor* can be challenging, because there are multiple options to summarize the data, as shown in Figure 2.30. Google Sheets will offer its automated guess based on the context, but you may need to manually select the best option to represent your data as desired. Three of the most common options to summarize values are:

- SUM: the total value of numeric responses (What is the total years of schooling for readers?)
- COUNT: frequency of numeric responses (How many readers listed 20 years of schooling?)

- COUNTA: frequency of text responses (How many readers listed occupation as “educator”)

Although Google Sheets pivot tables show raw numbers by default, you also can choose to display them as percentages of the row, of the column, or of the grand total.

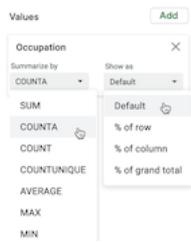


Figure 2.30: In the *Pivot table editor*, see multiple options to summarize *Values*.

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

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

Match Columns with VLOOKUP

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

Here’s a scenario that illustrates why and how to use the VLOOKUP function. Figure 2.31 shows two different sheets with sample data about food banks that help feed hungry people in different parts of the US, drawn from Feeding America: Find Your Local Food Bank. The first sheet lists individual people at each food bank, the second sheet lists the address for each food bank, and the two share a common column named *organization*. Your goal is to produce one sheet

that serves as a mailing list, where each row contains one individual's name, organization, and full mailing address. Since we're using a small data sample to simplify this tutorial, it may be tempting to manually copy and paste in the data. But imagine an actual case that includes over 200 US food banks and many more individuals, where using an automated method to match and paste data is essential.

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

Below the table are tabs for 'names' and 'addresses' with dropdown menus for sorting and filtering.

Figure 2.31: Your goal is to create one mailing list that matches individual names and organizations on the left sheet with their addresses on the right sheet.

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

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

2. In your editable copy of the Google Sheet, the *names* tab will be our destination for the mailing list we will create. Go to the *addresses* sheet, copy the column headers for *street - city - state - zip*, and paste them into cells C1 through F1 on the *names* sheet, as shown in Figure 2.32. This creates new column headers where our lookup results will be automatically pasted.
3. In the *names* sheet, click in cell C2 and type `=VLOOKUP`, and Google Sheets will suggest that you complete the full formula in this format:

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

+ names addresses source

Figure 2.32: Paste the last four column headers from the *addresses* sheet into the *names* sheet.

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

Here's what each part means:

- search_key = The cell in 1st sheet you wish to match.
 - range = At least two columns in the 2nd sheet to search for your match and desired result.
 - index = The column in the 2nd sheet range that contains your desired result, where 1 = first column, 2 = second column, etc.
 - [is_sorted] = Enter `false` to find exact matches only, which makes sense in this case. Otherwise, enter `true` if the first column of the 2nd sheet range is sorted and you will accept the closest match, even if not an exact one.
4. You can either type in the formula with comma separators =VLOOKUP(B2,'addresses'!A:E,2,false), or click on the relevant cells, columns, and sheets for the tool to automatically enter it for you, as shown in Figure 2.33. What's new here is that this formula in the *names* sheet refers to a range of columns A to E in the *addresses* sheet. Press *Return* or *Enter* on your keyboard.

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

+ names addresses source

	A	B	C	D	E
1	organization	street	city	state	zip
2	Arkansas Food Bank	4301 W 65th St	Little Rock	AR	72209
3	Central Texas Food Bank	6500 Metropolis Dr	Austin	TX	78744
4	Utah Food Bank	3150 South 900 West	Salt Lake City	UT	84119
5					

+ names addresses source

Figure 2.33: The VLOOKUP formula in cell C2 of the *names* sheet (top) searches for matches across columns A to E in the *addresses* sheet (bottom).

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

- B2 = The search_key: the cell in the *organization* column you wish to match in the *names* sheet

- '`addresses`'!A:E = The range where you are searching for your match and results across columns A to E in the *addresses* sheet.
 - 2 = The index, meaning your desired result appears in the 2nd column (*street*) of the range above.
 - `false` = Find exact matches only.
5. After you enter the full VLOOKUP formula, it will display the exact match for the first organization, the Central Texas Food Bank, whose address is 6500 Metropolis Dr. Click and hold down on the blue dot in the bottom-right corner of cell C2, and drag your crosshair cursor across columns D to F and let go, which will automatically paste and update the formula for the city, state, and zip columns, as shown in Figure 2.34.

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

Figure 2.34: Click on cell C2, then hold-and-drag the bottom-right blue dot across columns D to F, which automatically pastes and updates the formula.

6. Finally, use the same hold-and-drag method to paste and update the formula downward to fill in all rows, as shown in Figure 2.35.

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

Figure 2.35: Click on cell F2, then hold-and-drag the bottom-right blue dot down to row 11, which automatically pastes and updates the formula.

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

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

Connect Sheets with a Relational Database

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

A spreadsheet is sometimes called a “flat-file database” because all of the records are stored in rows and columns in a single table. For example, if you kept a single spreadsheet of US food bank staff, every row would list an individual person, organization, and addresses, just like the mailing list we created in Figure 2.35 in the prior section on VLOOKUP.

But keeping all of your data in a single spreadsheet can raise problems. For example, it contains lots of duplicated entries. For people who all work at the same food bank, each row contains a duplicate of that organization’s address. If an organization moves to a new location, you need to update all of the rows that contain those addresses. Or if two organizations merge together under a new name, you need to update all of the rows for individuals affected by that change. While keeping all of your information organized in a single spreadsheet initially sounds like a good idea, when your dataset grows in size and internal relationships (such as tracking people who are connected to organizations, etc.), continually updating every row becomes a lot of extra work.

Instead of a single spreadsheet, consider using a relational database, which organizes information into separate sheets (also known as tables), but continually maintains the relevant connections between them. Look back at the two-sheet problem we presented in Figure 2.31 at the beginning of the VLOOKUP section. The first sheet lists individual people at each food bank, the second sheet lists the address for each food bank, and the two sheets share a column named *organization* that shows how they are related. Relational databases can save you time. For example, if you update an organization’s address in one sheet, the linked sheet will automatically reflect this change in every row for staff who work at that organization.

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

To demonstrate, we imported both of the Google Sheets above into this live Airtable database called Food Banks sample, which anyone with the link can view, but not edit. At the top are tabs to view each sheet, named *people* and *food banks*. To transform this into a relational database, we used Airtable settings to

link the *organization* column in the *people* sheet to the *food banks* sheet, where the addresses are stored, as shown in Figure 2.36.

The screenshot shows the Airtable interface with two sheets: 'people' and 'food banks'. The 'people' sheet has a table with columns 'A' (row index) and 'name'. The 'organization' column is currently empty. A modal window is open over the table, titled 'Link to food banks'. It contains a dropdown menu with 'Arkansas Food Bank' and 'Utah Food Bank' as options. Below the dropdown, there are two checkboxes: 'Allow linking to multiple records' (which is checked) and 'Limit record selection to a view'. At the bottom right of the modal are 'Cancel' and 'Save' buttons. The 'food banks' sheet header is visible at the top of the interface.

Figure 2.36: In this Airtable sample, we linked the *organization* column in the *people* sheet to the *food banks* sheet.

In Airtable, click on a linked row to expand it and view related data. For example, if you click and expand on the first row in the *people* sheet, their organization's full address appears from the *food banks* sheet, as shown in Figure 2.37. In our editable version, if we update the address for one organization in the *food banks* sheet, it's automatically changed for all employees linked to that organization in the *people* sheet. In addition, Airtable allows you to sort, filter, and create different views of your data that you can share with others, a topic we'll cover in Chapter 8: Embed on your Web. See more about its features in the Airtable Support page.

It's important to understand the conceptual differences between a "flat-file" spreadsheet and a relational database to help you determine when to use one tool versus another. As you've learned in the sections above, spreadsheets are your best choice to begin organizing and analyzing your data, using methods such as sorting, filtering, pivoting, and lookup, to help reveal the underlying stories that you may wish to visualize. But relational databases are your best choice when maintaining large amounts of data with internal links, like one-to-many relationships, such as an organization with several employees.

The screenshot shows the Airtable interface with two sheets. The left sheet, titled 'people', has a grid view with columns for 'name' and 'organization'. The right sheet, titled 'Food Banks sample', displays detailed information for the selected record 'Denise B.'. The expanded record includes fields for 'NAME' (Denise B.) and 'ORGANIZATION' (Central Texas Food Bank), which further details the street address (6500 Metropolis Dr), city (Austin), and state (TX).

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

Denise B.

NAME
Denise B.

ORGANIZATION

Central Texas Food Bank

STREET	CITY	STATE
6500 Metropolis Dr	Austin	TX

Figure 2.37: In this Airtable demo, click on a row in one sheet to expand and view its linked data in another sheet.

Summary

If you’re one of the many people who “never really learned” about spreadsheets in school or on the job, or if you’ve taught yourself bits and pieces along the way, we hope that this chapter has successfully strengthened your skills. All of the subsequent chapters in this book, especially those on designing interactive charts in chapter 6 and interactive maps in chapter 7, require a basic level of familiarity with spreadsheets. In addition to serving as incredible time-savers when it comes to tedious data tasks, the spreadsheet tools and methods featured above are designed to help you share, sort, calculate, pivot, and lookup matching data, with the broader goal of visualizing your data stories.

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

Chapter 3

Find and Question Your Data

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

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

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

Guiding Questions for your Data Search

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

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

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

Also, it’s perfectly normal to revise your question as your research evolves. For example, we once began a data project by naively asking *What were Connecticut public school test scores in the 1960s?* Soon we discovered that standardized state-level school testing as we know it today did not appear in states like Connecticut until the mid-1980s school accountability movement. Even then, results were not widely visible to the public until newspapers began to publish them once a year in print in the 1990s. Later, real estate firms, school-ratings companies, and government agencies began to publish data continuously on the web as the Internet expanded in the late 1990s and early 2000s. Based on what we learned, we revised our research question to *When and how did Connecticut homebuyers start to become aware of school test scores, and how did these influence the prices they were willing to pay for access to selected public school attendance areas?*² Be prepared to refine your question when the

²Jack Dougherty et al., “School Choice in Suburbia: Test Scores, Race, and Hous-

evidence leads you in a better direction.

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

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

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

What if no one has collected the data you're looking for? Sometimes this happens due to more than a simple oversight. In *Data Feminism*, Catherine D'Ignazio and Lauren Klein underscore how issues of data collection "are directly connected to larger issues of power and privilege" by recounting a story

ing Markets," *American Journal of Education* 115, no. 4 (August 2009): 523–48, http://digitalrepository.trincoll.edu/cssp_papers/1.

about tennis star Serena Williams. When Williams experienced life-threatening complications while giving birth to her daughter in 2017, she called public attention to the way that she, a Black woman, needed to advocate for herself in the hospital. After her experience, she wrote on social media that “Black women are over 3 times more likely than white women to die from pregnancy- or childbirth-related causes,” citing the US Centers for Disease Control and Prevention (CDC). When journalists followed up to investigate further, they discovered the absence of detailed data on maternal mortality, and what a 2014 United Nations report described as a “particularly weak” aspect of data collection in the US healthcare system. Journalists reported that “there was still no national system for tracking complications sustained in pregnancy and childbirth,” despite comparable systems for other health issues such as heart attacks or hip replacements. Power structures are designed to count people whose lives either are highly valued, or under a high degree of surveillance. D’Ignazio and Klein call on us critically examine these power systems, collect data to counter their effects, and make everyone’s labor in this process more visible.³ If no one has collected the data you’re looking for, perhaps you can make valuable steps to publicly recognize the issue and contribute to positive change.

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

Public and Private Data

When searching for data, you also need to be informed about debates about public and private data. Not only do these debates influence what kinds of search you will conduct, but they also involve you in deeper ethical issues about data collection and distribution. At stake are two major questions. The first asks: *To what extent should organizations be allowed to collect data about private individuals?* The second question is: *When our government collects data, to what extent should it be publicly available?* This section offers our general observations about these debates, based primarily on our context in the United States. Since we are not lawyers (thank goodness!), please consult with legal experts for advice about your specific case.

Several critics of “big data” worry that governments are becoming more like a totalitarian “Big Brother” as they collect more data about individual citizens in the digital age. In the United States, concerns mounted in 2013 when whistleblower Eric Snowden disclosed how the National Security Agency conducted global surveillance using US citizen email and phone records provided

³D’Ignazio and Klein, *Data Feminism*, chap. 1.

by telecommunications companies. Shoshana Zuboff, a Harvard Business School professor and author of *The Age of Surveillance Capitalism*, warns of an equal threat posed by corporations that collect and commodify massive amounts of individually-identifiable data for profit.⁴ Due to the rise of digital commerce, powerful technology companies own data that you and others consider to be private:

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

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

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

⁴Shoshana Zuboff, *The Age of Surveillance Capitalism: The Fight for a Human Future at the New Frontier of Power* (PublicAffairs, 2019), https://www.google.com/books/edition/The_Age_of_Surveillance_Capitalism/IRqrDQAAQBAJ.

may delay or reject it, which may pressure the requester to attempt to resolve the issue through time-consuming litigation. Around the world, over 100 nations have their own version of freedom of information laws, with the oldest being Sweden's 1766 Freedom of the Press Act, but these laws vary widely.

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

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

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

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

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

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

If you work with data, get informed about key debates over what should be public or private, become active in policy discussions about whose interests are being served, and contribute to making positive change. In the next section, you'll learn how to explore datasets that governments and non-governmental organizations have intentionally shared with the public.

Open Data Repositories

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

- **View and Export:** At minimum, open data repositories allow users to view and export data in common spreadsheet formats, such as CSV, ODS, and XLSX. Some repositories also provide geographical boundary files for creating maps.
- **Built-in Visualization Tools:** Several repositories offer built-in tools for users to create interactive charts or maps on the platform site. Some also

provide code snippets for users to embed these built-in visualizations into their own websites, which you'll learn more about in Chapter 8: Embed on Your Web.

- Application Program Interface (APIs): Some repositories provide endpoints with code instructions that allow other computers to pull data directly from the platform into an external site or online visualization. When repositories continuously update data and publish an API endpoint, it can be an ideal way to display live or “almost live” data in your visualization, which you'll learn more about in Chapter 11: Leaflet Map Templates.

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

- Data.gov, the official repository for US federal government agencies.
- Data.census.gov, the main platform to access US Census Bureau data. The Decennial Census is a full count of the population every ten years, while the American Community Survey (ACS) is an annual sample count that produces one-year, three-year, or five-year estimates for different census geographies, with margins of error.
- Eurostat, the statistical office of the European Union.
- Global Open Data Index, by the Open Knowledge Foundation.
- Google Public Data.
- Google Dataset Search.
- IPUMS, Integrated Public Use Microdata Series, the world's largest individual-level population database, with microdata samples from US and international census records and surveys, hosted by the University of Minnesota.
- openAfrica, by Code for Africa.
- Open Data Inception, a map-oriented global directory.
- Open Data Network, a directory by Socrata, primarily of US state and municipal open data platforms.
- World Bank Open Data, a global collection of economic development data.

In addition, students, staff, and faculty at better-funded institutions of higher education also may have access to a paid library subscription to “closed” data repositories. For example, Social Explorer includes decades of demographic, economic, health, education, religion, and crime data for local and national geographies, primarily for the US, Canada, and Europe. Previously, Social Explorer made many files available to the public, but it now requires a paid subscription or 14-day free trial.

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

Source Your Data

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

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

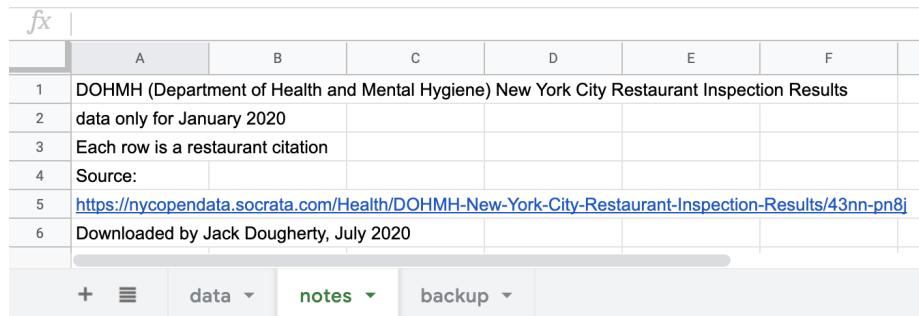
- data.csv
- file.ods
- download.xlsx

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

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

The second step is to save more detailed source notes about the data on a separate tab inside the spreadsheet, which works for multi-tab spreadsheet tools such as Google Sheets, LibreOffice, and Excel. Add a new tab named *notes* that describes the origins of the data, a longer description for any abbreviated labels, and when it was last updated, as shown in Figure 3.1. Add your own name and give credit to collaborators who worked with you. If you need to create a CSV file from this data, give it a parallel name to your multi-tabbed spreadsheet file so that you can easily relocate the original source notes.

A third step is to make a backup of the original data before cleaning or editing it. For a simple one-sheet file in a multi-tab spreadsheet tool, right-click on the tab containing the data to make a duplicate copy in another tab, also shown in Figure 3.1. Clearly label the new tab as a backup and leave it alone! For CSV files or more complex spreadsheets, create a separate backup file. To be clear, these simple backup strategy only helps you from making non-fixable edits to your original data. Make sure you have a broader strategy to backup your files from your computer or cloud account in case either of those are deleted or those systems crash.



The screenshot shows a spreadsheet interface with a tabs bar at the bottom. The active tab is 'notes'. Other tabs visible are 'data' and 'backup'. The main content area displays the following text:

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

Figure 3.1: Create separate spreadsheet tabs for data, notes, and backup.

Make a habit of using these three sourcing strategies—filenames, notes, and backups—to increase the credibility and replicability of your data visualizations. In the next section, we’ll explore more ways to reduce your chances of making “bad data” errors.

Recognize Bad Data

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

- Missing values: If you see blank or “null” entries, does that mean data was not collected? Or maybe a respondent did not answer? If you’re unsure, find out from the data creator. Also beware when humans enter a 0 or -1 to represent a missing value, without thinking about its consequences on running spreadsheet calculations, such as SUM or AVERAGE.
- Missing leading zeros: One of the zip codes for Hartford, Connecticut is 06119. If someone converts a column of zip codes to numerical data, it will strip out the leading zero and appear as 6119. Similarly, the US Census Bureau lists every place using a FIPS code, and some of these also begin with a meaningful zero character. For example, the FIPS code for Los Angeles County, California is 037, but if someone accidentally converts a column of text to numbers, it will strip out the leading zero and convert that FIPS code to 37, which represents the state of North Carolina.

- 65536 rows or 255 columns: These are the maximum number of rows supported by older-style Excel spreadsheets, or columns supported by Apple Numbers spreadsheet, respectively. If your spreadsheet stops exactly at either of these limits, you probably have only partial data.
- Inconsistent date formats: For example, November 3rd, 2020 is commonly entered in spreadsheets in the US as 11/3/2020 (month-date-year), while people in other locations around the globe commonly type it as 3/11/2020 (date-month-year). Check your source.
- Dates such as January 1st 1900, 1904, or 1970: These are default timestamps in Excel spreadsheets and Unix operating systems, which may indicate the actual date was blank or overwritten.
- Dates similar to 43891: When you type March 1 during the year 2020 into Microsoft Excel, it automatically displays as 1-Mar, but is saved using Excel's internal date system as 43891. If someone converts this column from date to text format, you'll see Excel's 5-digit number, not the dates you're expecting.

Another way to review the quality of data entry in any column in a spreadsheet is to create a filter or a pivot table as described in chapter 2. This allows you to quickly inspect the range of values that appear in that column, and whether they match what you expected to find.

What should you do when you discover bad data in your project? Sometimes small issues are relatively straightforward, do not call into question the integrity of the entire dataset, and you can fix them using methods to clean up messy data that we describe in chapter 4. But larger issues are more problematic. Follow the source of your data stream to try to identify where the issue began. If you cannot find and fix the issue on your own, contact the data provider to ask for their input, since they should have a strong interest in improving the quality of the data. If they cannot resolve an important data issue, then you need to pause and think carefully. In this case, is it wiser to continue working with problematic data and add a cautionary note to readers, or should you stop using the dataset entirely and call attention to its underlying problem? These are not easy decisions, and you should ask for opinions from colleagues. In any case, never ignore the warning signs of bad data.

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

Question Your Data

Now that you've found, sourced, and inspected some files, the next step to *question your data* by looking more deeply than what appears at its surface level. Read the source notes and examine the contents to reflect on what is explicitly stated—or unstated—to better understand its origin, context, and limitations.

You cannot program a computer to do this step for you, as it requires critical-thinking skills to see beyond the characters and numbers appearing on your screen.

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

- Abbreviated column headers, such as *Elevation* or *Income*, often appear in spreadsheets. Sometimes the original software limited the number of characters that could be entered, or the people who created the header names preferred to keep them short. But was *Elevation* entered in meters or feet? An abbreviated data label does not answer that key question, so you'll need to check the source notes, or if that's not available, compare elevation data for a specific point in the dataset to a known source that includes the measurement unit. Similarly, if you're working with US Census data, does the *Income* abbreviation refer to per person, per family, or per household? Also, does the value reflect the *median* (the mid-point in a range of numbers) or the *mean* (the average, calculated by adding up the sum and dividing by the number of values). Check definitions in the source notes.
- How exactly was the data recorded? For example, was *Elevation* for a specific location measured by a GPS unit on the ground? Or was the location geocoded on a digital map that contains elevation data? In most cases the two methods will yield different results, and whether that matters depends on the degree of precision required in your work. Similarly, when the US Census reports data from its annual American Community Survey (ACS) estimates for *Income* and other variables, these are drawn from small samples of respondents for lower levels of geography, such as a census tract with roughly 4,000 residents, which can generate very high margins of error. For example, it's not uncommon to see ACS estimates for a census tract with a mean family income of \$50,000—but also with a \$25,000 margin of error—which tells you that the actual value is somewhere between \$25,000 and \$75,000. As a result, some ACS estimates for small geographic units are effectively meaningless. Check how data was recorded, and any reported margins of error, in the source notes.
- To what extent is the data socially constructed? In other words, what do the data labels reveal or hide about how people defined categories in different social and political contexts, which differ across place and time? For example, we designed an interactive historical map of racial change for Hartford County, Connecticut using over 100 years of US Census data. But Census categories for race and ethnicity changed dramatically during those decades because people in power redefined these contested terms and moved who belonged in which group. For example, through the 1930s, the US Census separated “Native White” and “Foreign-born White” in its official reports, then combined and generally reported these as “White”

in later decades. Also, the Census classified “Mexican” as “Other races” in 1930, then moved this group back to “White” in 1940, then began to report “Puerto Rican or Spanish surname” data in 1960, followed by “Hispanic or Latino” in later decades, as an ethnic category that was distinct from race. The Census finally replaced “Negro” with “Black” in 1980, and in 2000 allow people to select more than one racial category, such as both “White” and “Black,” unlike prior decades when these terms were mutually exclusive and people could choose only one. As a result, historical changes in the social construction of race and ethnicity influenced how we designed our map to display “White” or “White alone” over time, with additional census categories relevant to each decade shown in the pop-up window, with our explanation of our decisions in the caption and source notes. There is no single definitive way to visualize socially-constructed data when definitions change across decades. But when you make choices about data, describe your thought process in the notes.

Here’s a paradox about working with data: some of these deep questions may not be fully answerable if the data was collected by someone other than yourself, especially if that person came from a distant place, or time period, or a different position in a social hierarchy. But even if you cannot fully answer these questions, don’t let that stop you from asking good questions about the origins, context, and underlying meaning of your data. Only by clarifying what we know—and what we don’t know—can we begin to recognize the limitations of the data. When you create visualizations, your job is also to *acknowledge the limitations of the data* by making thoughtful decisions about its design, and how you describe what it does—and does not—tell us. We’ll return to these topics when discussing chart design in Chapter 6 and telling your data story in chapter 14.

Summary

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

Chapter 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	5000000+

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 used 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

Find and Replace tool is one of the most powerful data clean-up tools in spreadsheets. You can use it to remove thousands separators from numbers (to change 1,234,567 to 1234567) or to remove units of measure that sometimes reside in the same cells as numbers (321 kg -> 321). You can also use it to bulk-change spellings, for example to shorten, expand, or translate country names (Republic of India -> India, US -> United States, Italy -> Italia).

Let's look at *Find and Replace* in practice. A common problem with US census data is that geographic names contain unnecessary words. For example, 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
```

We can use *Find and Replace* tool to remove the unwanted “town” part. 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 leave 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 t0wN 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.
8. Inspect the resulting sheet. Make sure town names such as Newtown remained untouched.

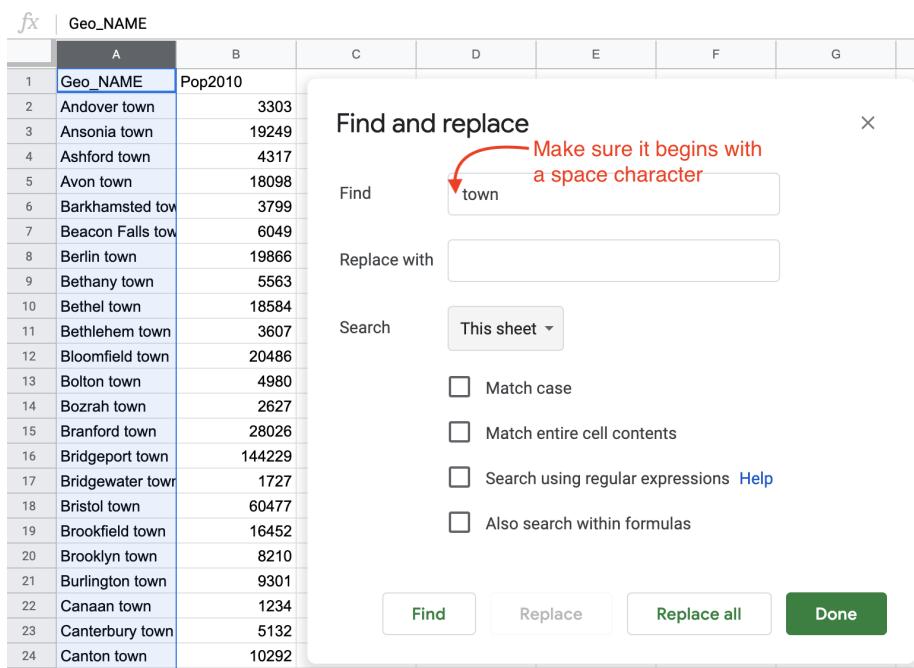


Figure 4.2: Find and Replace window in Google Sheets.

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).

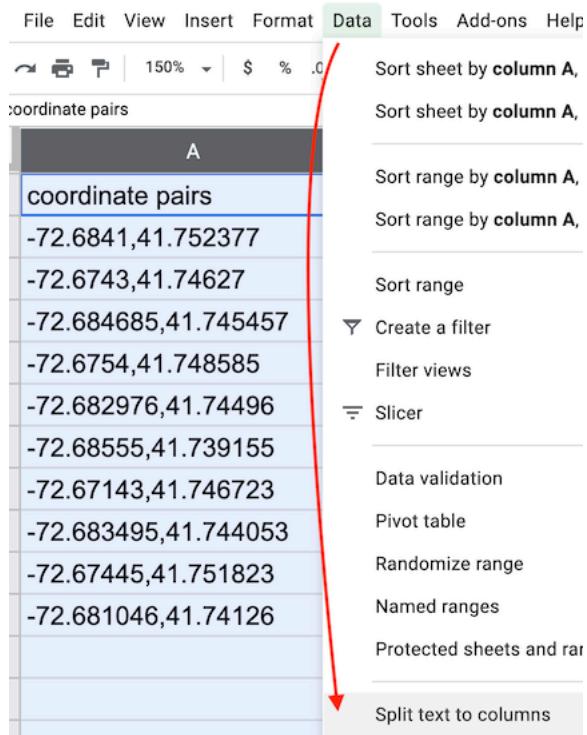


Figure 4.3: Select *Data* - *Split text to columns* to automatically separate data.

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

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 the following section.

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

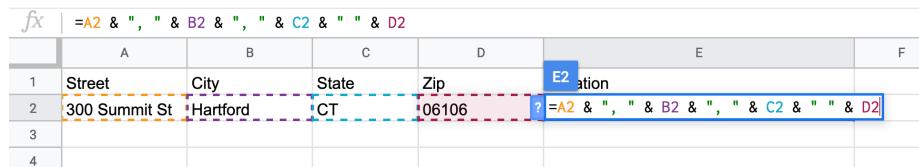
The data comes in four columns: street address, city, state, and zipcode. Let's say your mapping tool requires you to combine all of these terms into one location column, like that:

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 commas (" , ") or quoted spaces (" "), like this: `=A2 & ", " & B2 & ", " & C2 & " " & D2`.



A screenshot of a Google Sheets spreadsheet. The first row (header) has columns labeled A, B, C, D, and E. The second row (data) has cells containing 'Street', 'City', 'State', 'Zip', and a formula in cell E2. The formula is `=A2 & ", " & B2 & ", " & C2 & " " & D2`. The formula bar above shows the full formula. The cell E2 is highlighted with a blue border, and the formula is also highlighted with a dashed red box.

	A	B	C	D	E	F
1	Street	City	State	Zip	Location	
2	300 Summit St	Hartford	CT	06106	<code>=A2 & ", " & B2 & ", " & C2 & " " & D2</code>	
3						
4						

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

Note: Lisa Charlotte Rost from Datawrapper has written a brilliant blog post talking about data preparation for charting and analysis in Google Sheets, which we recommend for further reading.

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.
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.

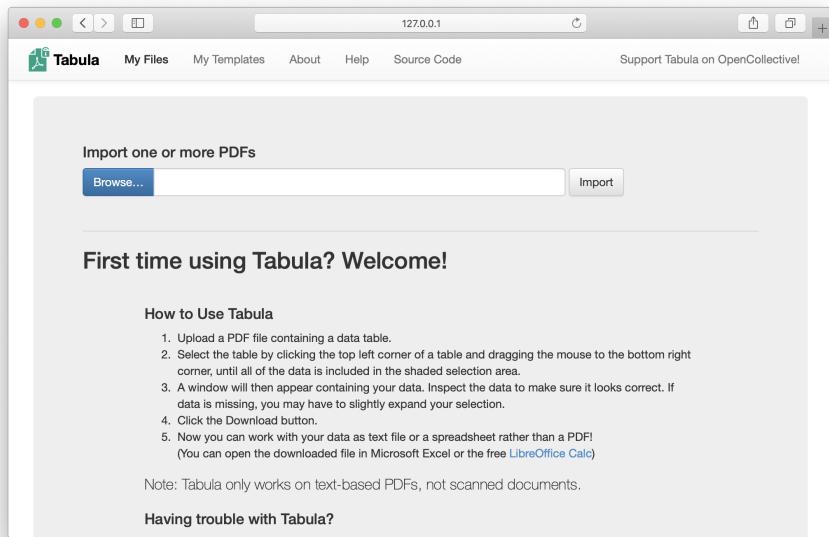


Figure 4.5: Tabula welcome page.

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.

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 wish 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.

The screenshot shows the Tabula software interface. At the top, there are tabs for 'My Files', 'My Templates', 'About', 'Help', and 'Source Code'. Below the tabs, there are three PDF documents labeled 1, 2, and 3. Document 1 contains a map of Connecticut with COVID-19 case data. Document 2 is titled 'APPENDIX A. Towns with Confirmed and Probable Cases of COVID-19' and includes a table of cases. Document 3 contains a bar chart. In the center, there is a large table titled 'Table does not include 234 cases pending address validation'. The table has columns for 'Towns', 'Cases', 'Deaths', 'Cases', 'Deaths', and 'Cases'. Many rows in the table are highlighted with red borders, indicating selected data. The table is titled 'APPENDIX A. Towns with Confirmed and Probable Cases of COVID-19'.

Towns	Cases	Deaths	Cases	Deaths	Cases
Ashley	30	Griswold	24	Preston	53
Amistad	256	Groton	87	Putnam	27
Ansonia	15	Hartford	94	Ridgefield	71
Avon	116	Haddam	29	Ridgefield	204
Barkhamsted	26	Hanover	944	Roxbury	885
Beacon Falls	45	Hartford	2	Roxbury	2
Berlin	150	Hartford	2250	Salem	4
Bethany	32	Hartford	1	Salem	1
Bethel	245	Havertown	28	Scituate	0
Bethlehem	12	Hartford	23	Sharon	220
Bloomfield	482	Kent	2	Sharon	27
Bolton	73	Killington	29	Simsbury	277
Borism	7	Killington	14	Sherman	14
Branford	227	Lebanon	23	Simsbury	108
Bridgewater	3345	Litchfield	23	Simsbury	286
Bristol	7	Litchfield	11	South Windsor	139
Brookfield	153	Litchfield	38	South Windsor	372
Brooklyn	155	Lyme	2	Southington	311
Brownsville	22	Madison	137	Sprague	4
Burlington	24	Middlefield	640	Sprague	103
Canaan	0	Mansfield	29	Stamford	3104
Canterbury	11	Mansfield	73	Stonington	1
Canton	83	Minden	722	Stonington	31
Chaplin	3	Middletown	44	Stonington	790
Cheshire	183	Middlefield	18	Suffield	114
Chester	46	Middletown	549	Thomaston	54
Chittenden	52	Middlefield	620	Thomaston	131
Cochituate	36	Monroe	105	Tolland	46
Colchester	3	Mosquiton	233	Tolland	153
Columbia	22	Mosquiton	14	Trumbull	509
Comstock	6	New Britain	162	Vernon	1
Cromwell	40	New Britain	958	Vernon	177
Cromwell	115	New Canaan	164	Vilistinen	9
Danbury	1373	New Haven	120	Vilistinen	441
Darien	204	New Hartford	25	Warren	4
Deep River	9	New Haven	2443	Watertown	19
Derby	153	New London	136	Waterverry	1866
Durham	5	New Milford	269	Watford	155
East Granby	9	New Haven	233	West Hartford	143
East Haddam	17	Newtown	211	West Hartford	655
East Hartland	42	Newtown	13	West Hartford	2500
East Haddam	793	North Bradford	81	Westbrook	29
East Haddam	489	North Haven	4	Westbrook	23
East Lymne	142	North Haven	243	Wingdale	288
East Windsor	146	North Stratford	12	Wethersfield	248
Eastford	8	North Stonington	1993	Wethersfield	33
Easton	30	Norwich	89	Wilton	201
Egyptian	56	Oliver	13	Whittemere	54

Figure 4.6: Selected tables are highlighted in red.

- 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.

	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?

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

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.

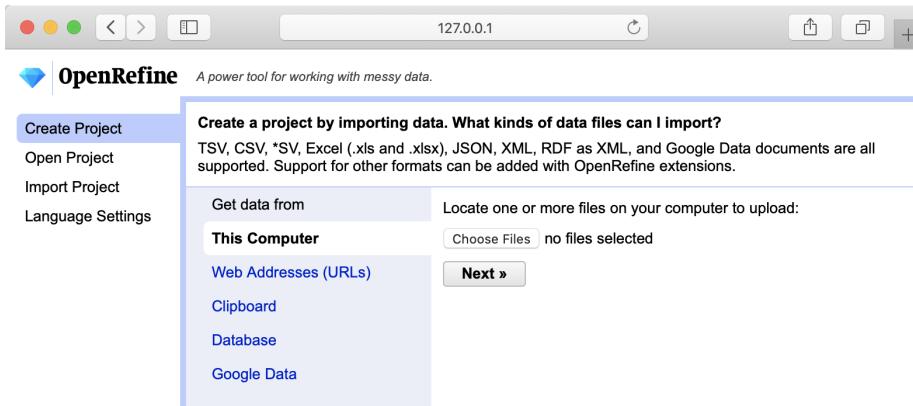


Figure 4.8: OpenRefine starting page.

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.

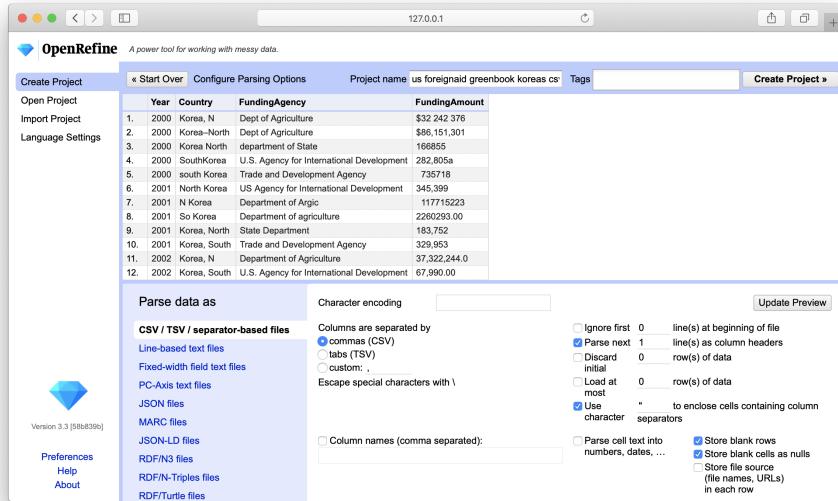


Figure 4.9: OpenRefine parsing options.

Convert Dollar Amounts from Text to Numbers

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

Each column header has its own menu (callable by clicking the arrow-down button). Left-aligned numbers in a column are likely represented as text (as is the case with FundingAmount column in our example), and they need to be transformed into numeric format.

1. To transform text into numbers, open the column menu, and go to *Edit cells > Common transforms > To number*.
2. You will see that some numbers became green and right-aligned (success!), but most did not change. That is because dollar sign (\$) and commas (,) confuse OpenRefine and prevent values to be converted into numbers.
3. Let's remove \$ and , from the FundingAmount column. In the column menu, choose *Edit cells > Transform*. In the Expression window, type `value.replace(',', '')` and notice how commas disappear in the preview window. When you confirm your formula works, click *OK*.
4. Now, repeat the previous step, but instead of a comma, remove the \$ character. (Your expression will become `value.replace('$', '')`).
5. In steps 3 and 4, we replaced text (string) values with other string values, making OpenRefine think this column is no longer numeric. As a result,

all values are once again left-aligned and in black. Perform step 1 again to see that all but three cells turning green (successfully converting to numeric). Now we need to remove spaces and an `a` character at the end of one number. Fix those manually by hovering over cells, and clicking the `edit` button (in the new popup window, make sure to change *Data type* to *number*, and hit *Apply*, like in Figure 4.10).

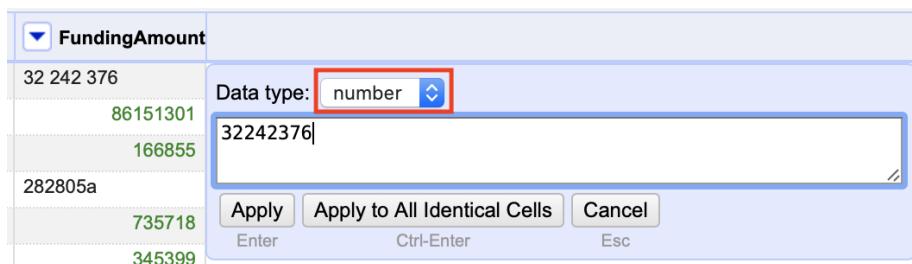


Figure 4.10: Manually remove spaces and extra characters, and change data type to number.

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

Cluster Similar Spellings

When you combine different data sources, or process survey data where respondents wrote down their answers as opposed to selecting them from a dropdown menu, you might end up with multiple spellings of the same word (town name, education level – you name it!). One of the most powerful features of OpenRefine is the ability to cluster similar responses.

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

1. To begin standardizing spellings, click on the arrow-down button of *Country* column header, and choose *Edit cells > Cluster and edit*. You will see a window like the one shown in Figure 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 (see first cluster in Figure 4.11). 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.

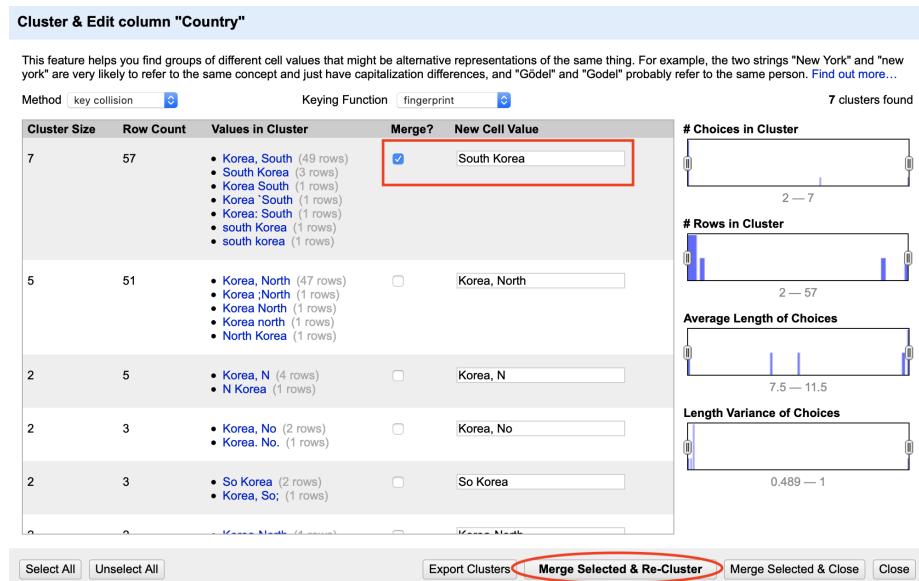


Figure 4.11: Cluster similar text values.

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. The more clean-up tools and techniques you know, the more able and adaptable you become to tackle more complex cases.

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

Chapter 5

Table Your Data

Before you leap into designing colorful charts or maps, consider that in some cases, the best way to present data may be through a table. In this chapter, you'll learn how to create an interactive table, which allows users to sort columns from low-to-high values, and search results to see specific rows. Interactive tables are most appropriate when you expect that your readers want to dig a bit into the data to see details for their local community, such as TODO: mention Wash Post story on opioid epidemic by county, or any listing of school data.

TODO: write chapter, with three examples:

- published Google Sheet
- Datawrapper
- Airtable, if you have a relational database

See the Embed chapter for how to place these in a web page

Chapter 6

Chart Your Data

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

In this chapter, we will look at main principles of chart design, and learn to identify good charts from bad ones. You will learn important rules that apply to all charts, and also some aesthetic guidelines to follow when customizing your own designs. 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 website in chapter 8.

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 the story you wish to tell, such as the type of data comparison you wish to draw to your reader's attention. Once you choose your chart type, follow our tool recommendations 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 and Highcharts templates in chapter 10, 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 9.

Table 6.1: Basic Chart Types and Tutorials

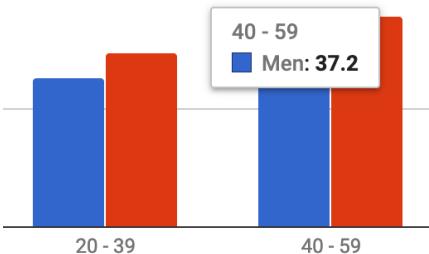
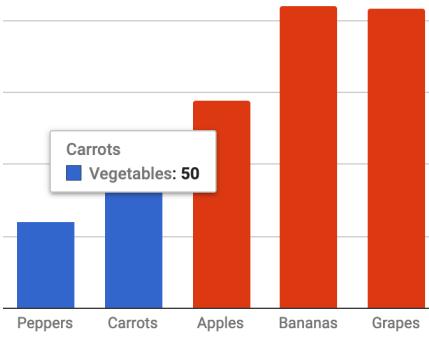
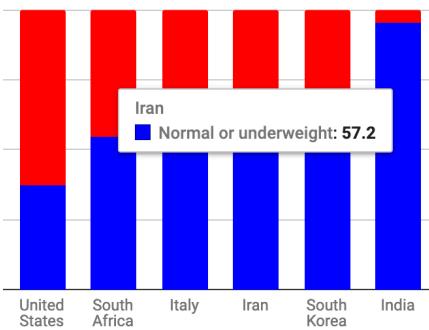
Chart	Best use and tutorials in this book														
Grouped column or bar	<p>Best to compare categories side-by-side. Vertical columns, or horizontal bars for long labels. Easy tool: Google Sheets bar and column tutorialPower tool: Chart.js and Highcharts templates</p>  <p>A grouped bar chart comparing men's health status across two age groups: 20-39 and 40-59. The y-axis represents a scale from 0 to 100. Blue bars represent the 20-39 group, and red bars represent the 40-59 group. A callout box highlights the 40-59 group with the text "40 - 59" and "Men: 37.2".</p> <table border="1"> <thead> <tr> <th>Age Group</th> <th>Men</th> </tr> </thead> <tbody> <tr> <td>20 - 39</td> <td>~65</td> </tr> <tr> <td>40 - 59</td> <td>37.2</td> </tr> </tbody> </table>	Age Group	Men	20 - 39	~65	40 - 59	37.2								
Age Group	Men														
20 - 39	~65														
40 - 59	37.2														
Separated column or bar	<p>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 and Highcharts templates</p>  <p>A separated bar chart comparing vegetable consumption across five fruit types: Peppers, Carrots, Apples, Bananas, and Grapes. The y-axis represents a scale from 0 to 100. Blue bars represent Peppers, Carrots, and Apples, while red bars represent Bananas and Grapes. A callout box highlights the Carrots category with the text "Carrots" and "Vegetables: 50".</p> <table border="1"> <thead> <tr> <th>Fruit Type</th> <th>Vegetables</th> </tr> </thead> <tbody> <tr> <td>Peppers</td> <td>~25</td> </tr> <tr> <td>Carrots</td> <td>50</td> </tr> <tr> <td>Apples</td> <td>~45</td> </tr> <tr> <td>Bananas</td> <td>~85</td> </tr> <tr> <td>Grapes</td> <td>~85</td> </tr> </tbody> </table>	Fruit Type	Vegetables	Peppers	~25	Carrots	50	Apples	~45	Bananas	~85	Grapes	~85		
Fruit Type	Vegetables														
Peppers	~25														
Carrots	50														
Apples	~45														
Bananas	~85														
Grapes	~85														
Stacked column or bar	<p>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 and Highcharts templates</p>  <p>A stacked bar chart comparing overweight status across six countries: United States, South Africa, Italy, Iran, South Korea, and India. The y-axis represents a scale from 0 to 100. Each bar is composed of blue (bottom) and red (top) segments. A callout box highlights the India bar with the text "Iran" and "Normal or underweight: 57.2".</p> <table border="1"> <thead> <tr> <th>Country</th> <th>Normal or underweight</th> </tr> </thead> <tbody> <tr> <td>United States</td> <td>~35</td> </tr> <tr> <td>South Africa</td> <td>~25</td> </tr> <tr> <td>Italy</td> <td>~35</td> </tr> <tr> <td>Iran</td> <td>~35</td> </tr> <tr> <td>South Korea</td> <td>~35</td> </tr> <tr> <td>India</td> <td>57.2</td> </tr> </tbody> </table>	Country	Normal or underweight	United States	~35	South Africa	~25	Italy	~35	Iran	~35	South Korea	~35	India	57.2
Country	Normal or underweight														
United States	~35														
South Africa	~25														
Italy	~35														
Iran	~35														
South Korea	~35														
India	57.2														

Chart	Best use and tutorials in this book
Histogram	Best to show distribution of raw data, with number of values in each bucket. Easy tool: Google Sheets bar and column tutorial Power tool: Chart.js and Highcharts templates
Pie chart	Best to show parts of a whole, but hard to estimate size of slices. Easy tool: Google Sheets pie chart tutorial Power tool: Chart.js and Highcharts templates
Line chart	Best to show continuous data, such as change over time. Easy tool: Google Sheets line chart tutorial Power tool: Chart.js and Highcharts 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

Chart	Best use and tutorials in this book
Stacked area chart	Best to show parts of a whole, with change over time. Easy tool: Google Sheets stacked area tutorialPower tool: Chart.js and Highcharts templates
XY Scatter chart	Best to show the relationship between two sets of data. Easy tool: Google Sheets scatter chart tutorial or Tableau Public scatter chart tutorialPower tool: Chart.js and Highcharts templates
Bubble chart	Best to show the relationship between three or four sets of data, using bubble size and color. Easy tool: Google Sheets bubble chart tutorialPower tool: Chart.js and Highcharts 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 6.1. It shows basic chart components that are shared among most chart types.

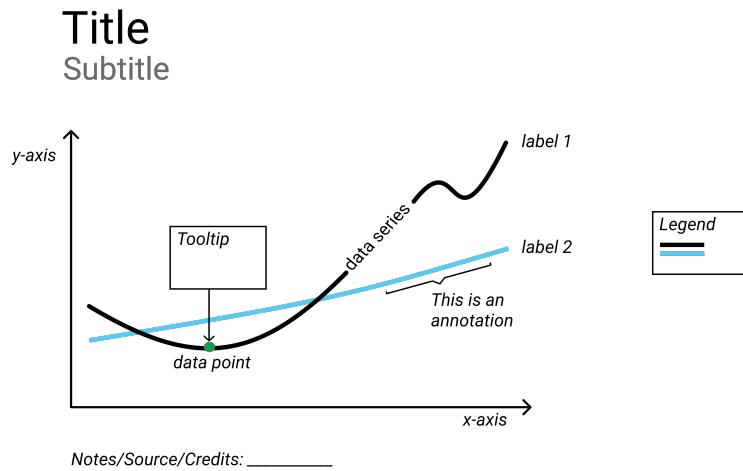


Figure 6.1: Common chart components.

A *title* is perhaps the most important element of any chart. A good title is short, clear, and tells a story on its own. For example, “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” 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 declutter the chart. But because tooltips are harder to interact with on smaller screens, such as phones and tablets, and are invisible when the chart is printed, only rely on them to convey additional, nice-to-have information. Make sure all essential information is visible without any user interaction.

Some Rules are More Important than Others

Although the vast majority of rules in data visualization are open to interpretation, there are some that are hard to bend.

Bar charts must start at zero

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

Starting y-axis at anything other than zero is a common trick used by some media and politicians to exaggerate differences in surveys and election results. Learn more about how to detect bias in data stories in chapter 13.

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.

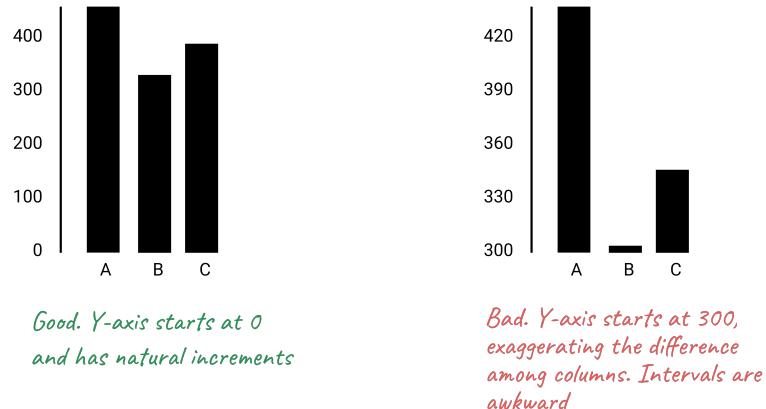


Figure 6.2: Start your bar chart at zero.

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

Chart Aesthetics

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

Avoid chart junk

Start with a white background and add elements as you see appropriate. You should be able to justify each element you add. To do so, ask yourself: Does this element improve the chart, or can I drop it without decreasing readability? This way you won't end up with so-called "chart junk" as shown in Figure 6.3, which includes 3D perspectives, shadows, and unnecessary elements. They might have looked cool in early versions of Microsoft Office, but let's stay away from them today. Chart junk distracts the viewer and reduces chart readability and comprehension. It also looks unprofessional and doesn't add credibility to you as a storyteller.

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.



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

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

Beware of pie charts

Remember that pie charts only show part-to-whole relationship, so all slices need to add up to 100%. Generally, the fewer slices—the better. Arrange slices from largest to smallest, clockwise, and put the largest slice at 12 o'clock. Figure 6.4 illustrates that.

If your pie chart has more than five slices, consider showing your data in a bar chart, either stacked or separated, like Figure 6.5 shows.

Don't make people turn their heads to read labels

When your column chart has long x-axis labels that have to be rotated (often 90 degrees) to fit, consider turning the chart 90 degrees so that it becomes a horizontal bar chart. Take a look at Figure 6.6 to see how much easier it is to read horizontally-oriented labels.

Arrange elements logically

If your bar chart shows different categories, consider ordering them, like is shown in Figure 6.7. You might want to sort them alphabetically, which can be useful

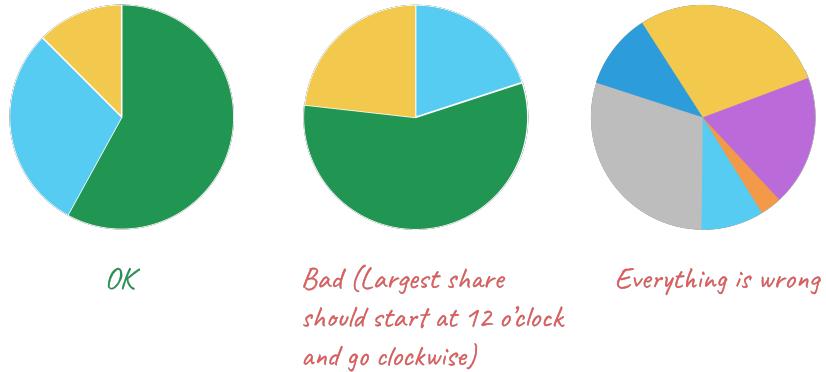


Figure 6.4: Sort slices in pie charts from largest to smallest, and start at 12 o'clock.

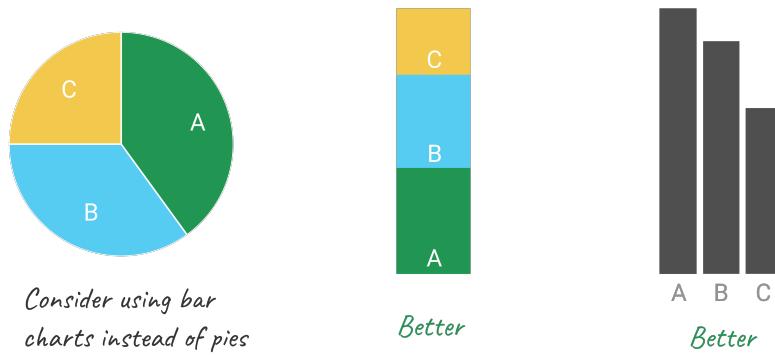


Figure 6.5: Consider using bar charts instead of pies.

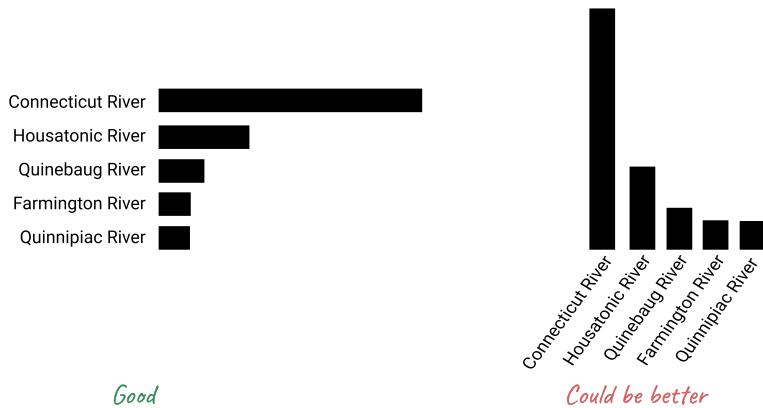


Figure 6.6: For long labels, use horizontal bar charts.

if you want the reader to be able to quickly look up an item, such as their town. Ordering categories by value is another common technique that makes comparisons possible. If your columns represent a value of something at a particular time, they have to be ordered sequentially, of course.



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

Do not overload your chart

When labelling axes, choose natural increments that space equally, such as [0, 20, 40, 60, 80, 100], or [1, 10, 100, 1000] for a logarithmic scale. Do not overload your scales. Keep your typography simple, and use (but do not overuse) **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 or yellow/blue, so be very careful with those. Figure 6.8 shows some good and bad examples of color use.

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

Google Sheets Charts

In addition to powerful data wrangling capabilities, Google Sheets has robust support for charting. 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 8.

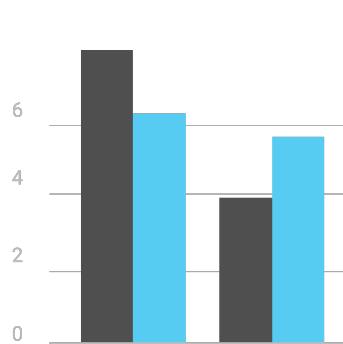
In this section, we will look at creating column and bar charts that are separated, grouped, and stacked. We will also look at making pie, line, area, and scatter charts, and learn to visualize three-dimensional data using bubble charts.

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*.

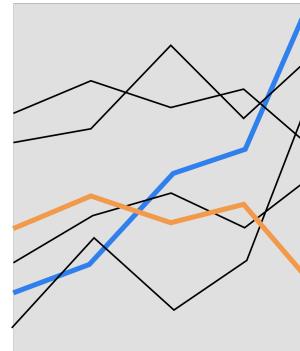
Tip: Visit Types of charts & graphs in Google Sheets for an overview of the various chart types supported by this tool.

Column and Bar Charts with Google Sheets

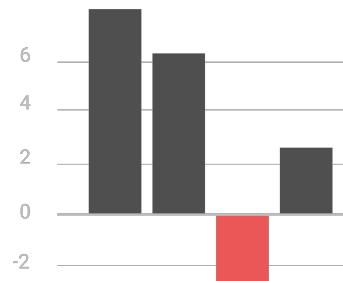
Column and bar charts are some of the most common types of charts in data visualization (column charts are just vertical bar charts). They are used to compare values across categories.



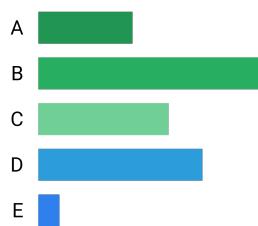
Good. Blue allows to distinguish between series



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



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

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

Grouped Column and Bar Charts

Figure 6.10 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.

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

1. Open Google Sheet Column chart with grouped data template in your browser.
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 6.11.

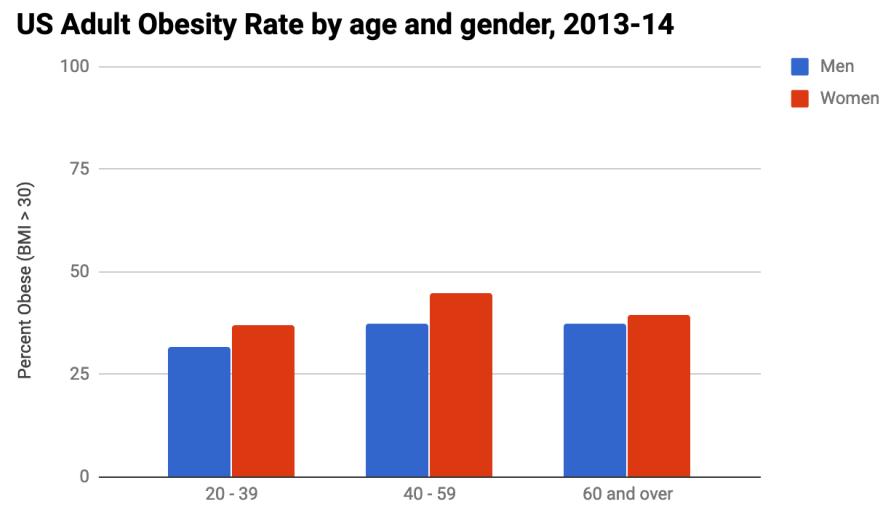


Figure 6.10: Grouped column chart with data from StateOfObesity.org. Explore the full-screen interactive version.

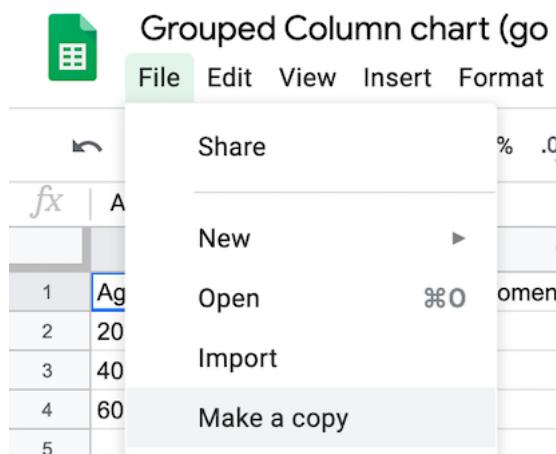


Figure 6.11: Make your own copy of the Google Sheet template.

- To remove the current chart from your copy of the spreadsheet, float your cursor to the top-right corner of the chart to make the 3-dot (kebab) menu appear, and select Delete, as shown in Figure 6.12.

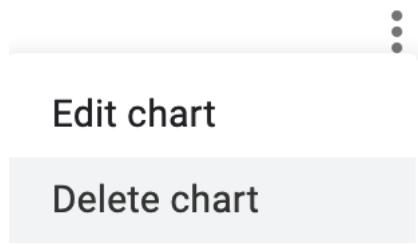


Figure 6.12: Float cursor in top-right corner of the chart to make the 3-dot (kebab) menu appear, and select Delete.

Note: Another name for the 3-dot menu symbol is the “kebab menu” because it resembles Middle Eastern food cooked on a skewer, in contrast to the three-line “hamburger menu” on many mobile devices, as shown in Figure 6.13.

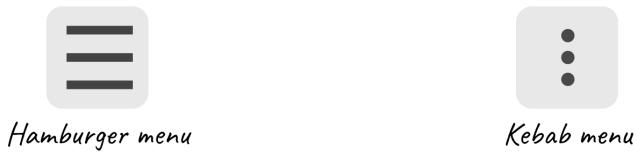


Figure 6.13: You should be able to distinguish kebab from hamburger menu icons.

- Format your data to make each column a data series, as shown in Figure 6.14, 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 6.14: 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 6.15.

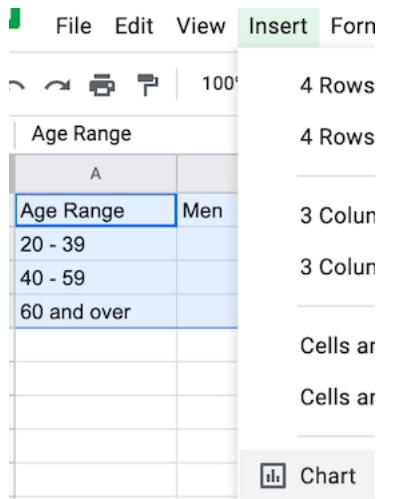


Figure 6.15: Select your data and then Insert the Chart.

7. In the Chart Editor, change the default selection to Column chart, with Stacking none, to display Grouped Columns, as shown in Figure 6.16. Or select *Horizontal bar chart* if you have longer labels.

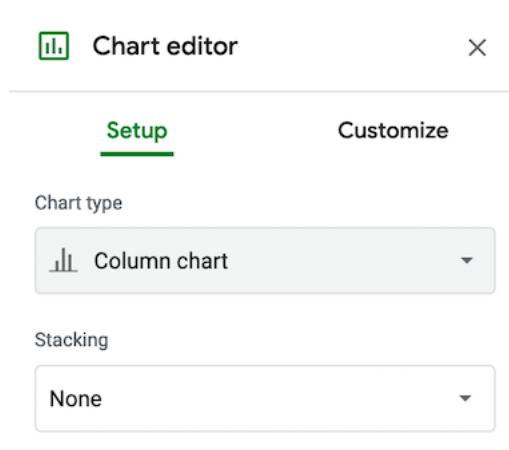


Figure 6.16: Change the default to Column chart, with Stacking none.

8. To customize title, labels, and more, in the Chart Editor select Customize, as shown in Figure 6.17.

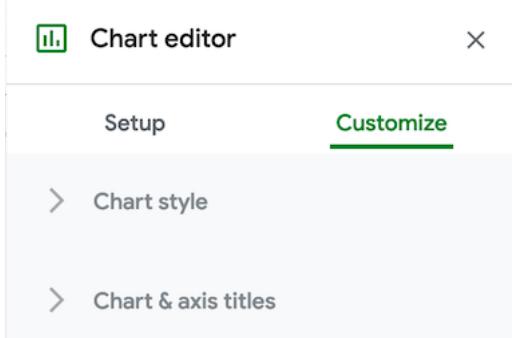


Figure 6.17: Select Customize to edit title, labels, and more.

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

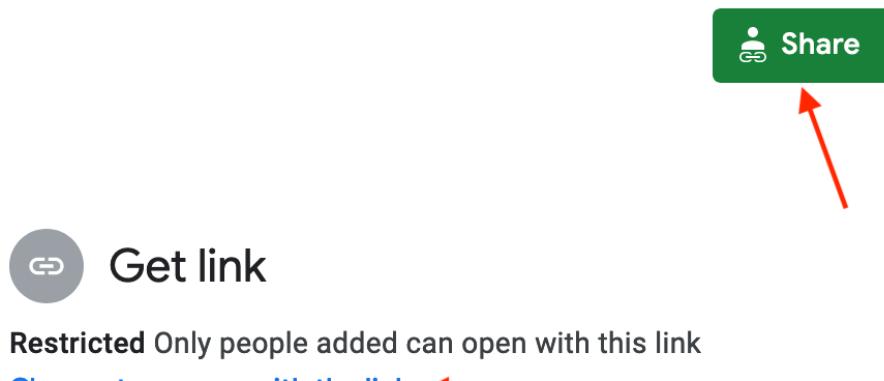


Figure 6.18: Click the Share button and then click *Change to anyone with the link* to make your data public.

10. To embed an interactive version of your chart in another web page, click the kebab menu in the upper-right corner of your chart, and select Publish Chart, as shown in Figure 6.19. In the next screen, select Embed and press the Publish button. See Chapter 8: Embed on the Web to learn what to do with the iframe code.

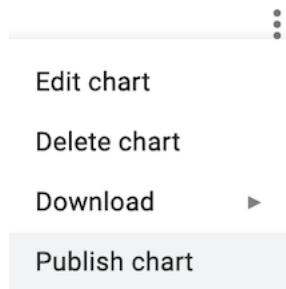


Figure 6.19: Select Publish Chart to embed an interactive chart on another web page.

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 6.20 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 6.10, 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 need to leave some cells blank, as shown in Figure 6.21. The rest of the steps remain the same as above.

To create your own separated column or bar chart using the fast-food example, make a copy of 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 represent parts of a whole instead of pie charts. For example, the stacked column chart in Figure 6.22 compares the percentage of overweight residents across nations, where colors allow for easy comparisons of weight-group subcategories across nations.

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 6.23. Then in the

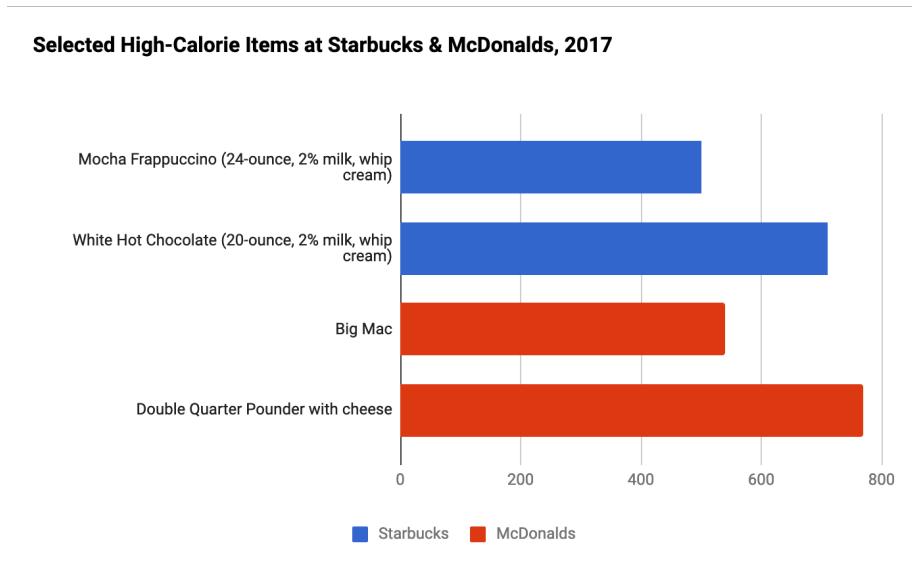


Figure 6.20: Separated bar chart with data from Starbucks and McDonalds. Explore the full-screen interactive version.

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

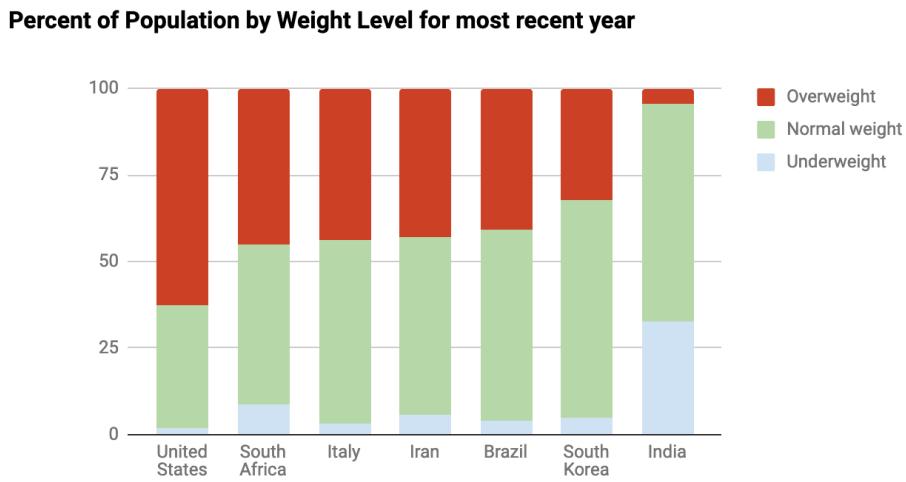


Figure 6.22: Stacked column chart with data from WHO and CDC. Explore the full-screen interactive version.

Chart Editor window, choose Chart Type > Stacked column chart (or Stacked bar chart). The rest of the steps are similar to the ones above.

To create your own stacked column or bar chart using the international weight level example, visit the Google Sheets Stacked Column Chart template and make a copy of the spreadsheet.

To change colors of series (for example, to show Overweight category in red), click the kebab menu in the top-right corner of the chart, then go to *Edit Chart* > *Customize* > *Series*. There, choose the appropriate series from the dropdown menu, and set its color from the Color dropdown menu that appears.

Histograms

Histogram is a type of bar chart that represents distribution of items, whether numerical or categorical. To build 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 you are 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
-----	-----

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

Figure 6.23: Create a stacked column or bar chart by structuring your data as shown.

0	12	
1	11	
2	7	
.....		
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 6.24

If you wish to use our fictional email dataset to create your own histogram, you can make a copy of the 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:

TimeOfDay	Emails	
-----	-----	

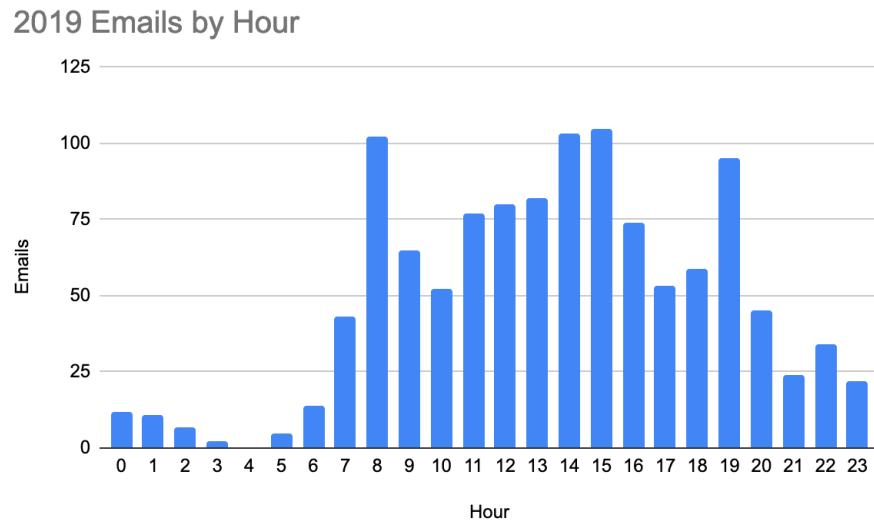


Figure 6.24: Histogram chart with fictitious source data. Explore the full-screen interactive version.

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 6.25. 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.

To make a pie chart with Google Sheets, arrange your data in two columns,

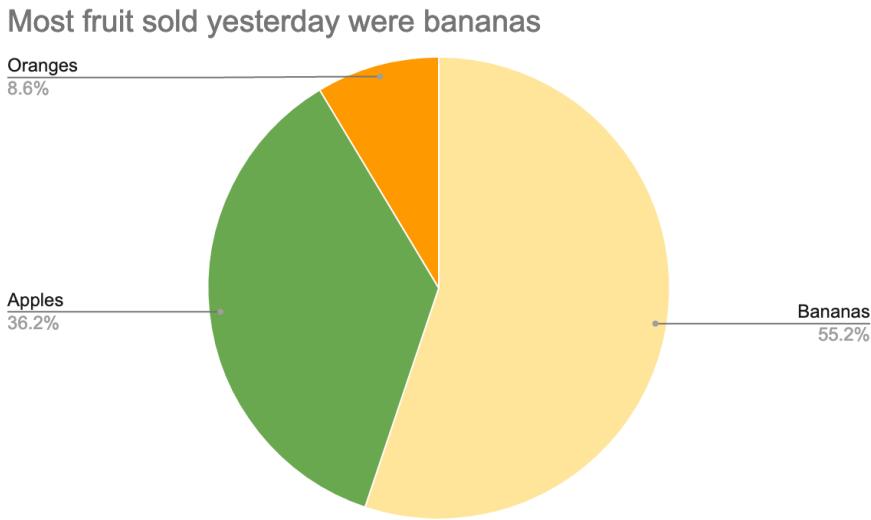


Figure 6.25: Pie chart with fictitious source data. Explore the full-screen interactive version.

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 choose **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 6.26

shows per-capita meat availability in the US for the past 110 years. You can see that the level of chicken (shown in light-green) rises steadily and surpasses beef (blue) and pork (gray).

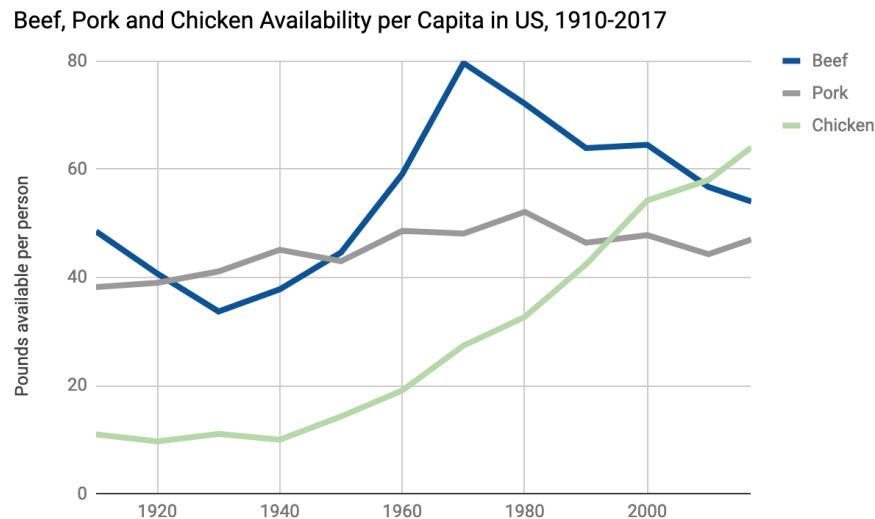


Figure 6.26: Line chart showing meat availability per capita in the US, according to the US Department of Agriculture. Explore the full-screen interactive version.

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 6.27.

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 6.28. Here, we

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

Figure 6.27: Data for the line chart shown in Figure 6.26.

can still see that chicken has been on the rise since the 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.

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. Such animated bubble charts were popularized by Hans Rosling, a renowned Swedish professor of global health.

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

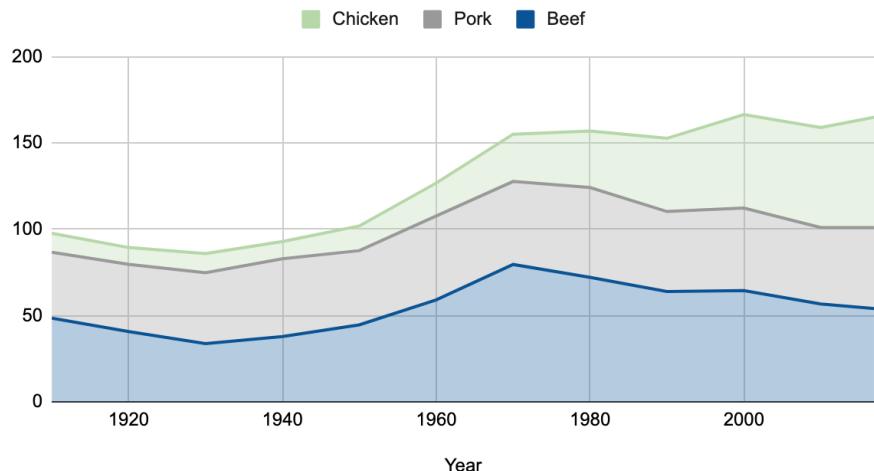


Figure 6.28: In addition to individual meat availability, stacked area charts show the overall availability. See data by US Department of Agriculture. Explore the full-screen interactive version.

Note: We recommend you watch one of Hans Rosling's famous TED talks to see animated bar charts in action. You can also visit Gapminder Foundation website to see more data visualizations and learn more about Hans's work and legacy.

XY Scatter chart

The scatter chart in Figure 6.29 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 it 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 6.29 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 6.30 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.

Fertility and Life Expectancy in Selected Nations, 2018

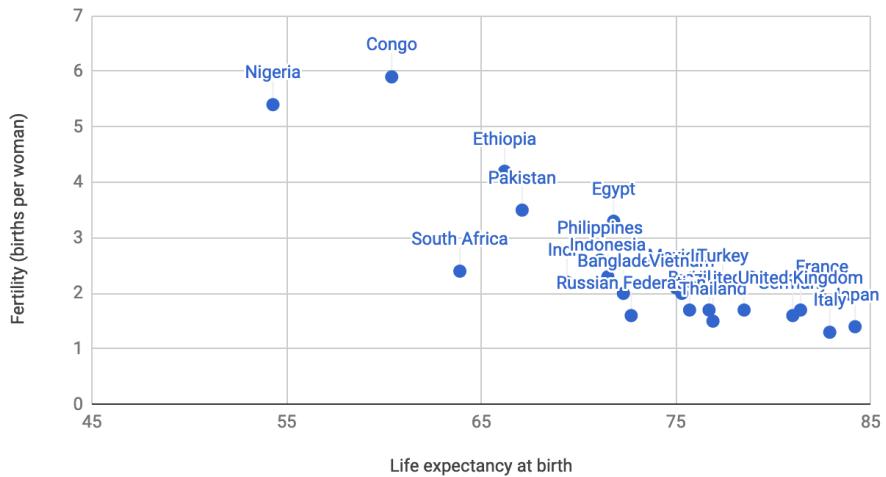


Figure 6.29: This scatter chart shows that nations with lower fertility tend to have higher life expectancy. See data by World Bank. Explore the full-screen interactive version.

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

To build a scatter chart, select the *two* columns that contain your numeric data, and go to *Insert > Chart*. Google Sheets will likely 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, open the kebab menu (3 dots) of your Series dataset (Fertility), and then *Add labels* (see Figure 6.31). The labels added by default will be the x-values of points. To make Google Sheets read labels from the third column (*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.

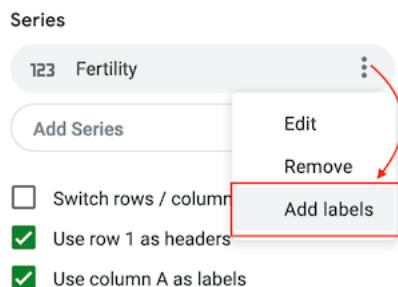


Figure 6.31: In the chart's Setup window, choose *Add labels* to the Series.

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.

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 6.32 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.

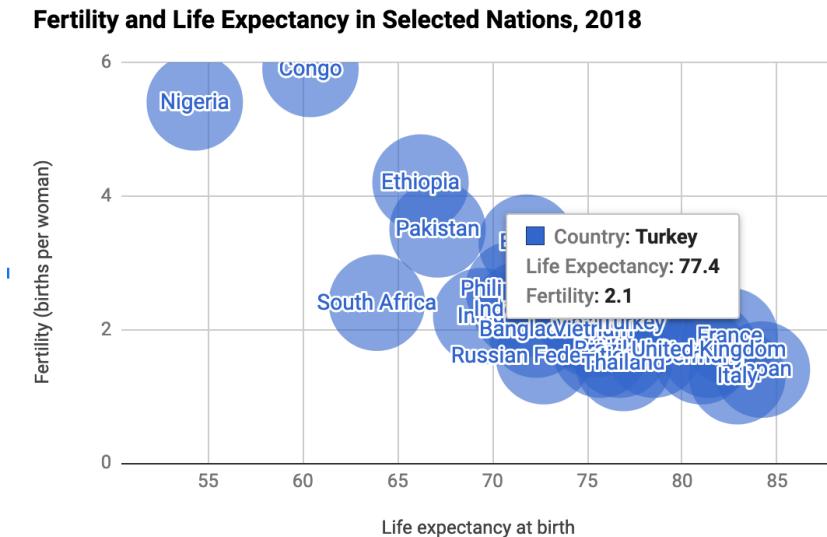


Figure 6.32: This bubble chart is essentially a scatter chart, because no other dimensions (colors, sizes) are used. See data by World Bank. Explore the full-screen interactive version.

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 placed 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 6.33 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.

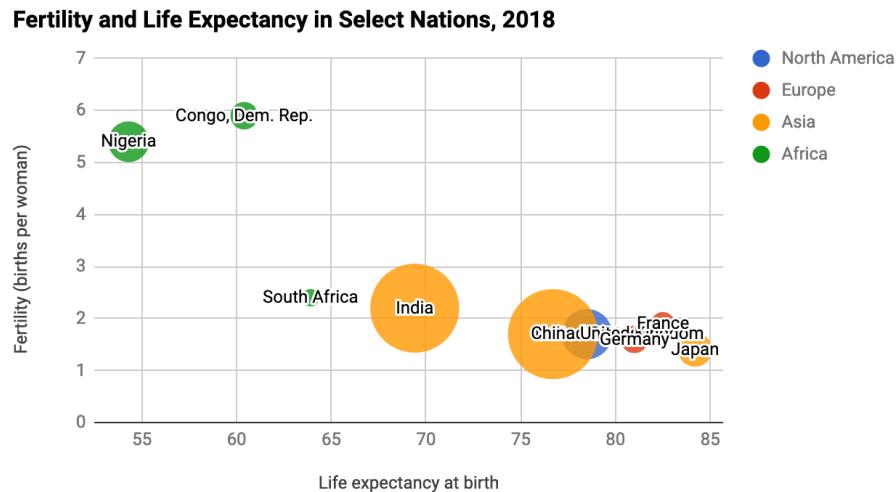


Figure 6.33: This bubble chart shows fertility and life expectancy for several countries, including their population (shown by bubble size) and region (shown by bubble color). See data by World Bank. Explore the full-screen interactive version.

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 contain 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.

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.

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

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

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

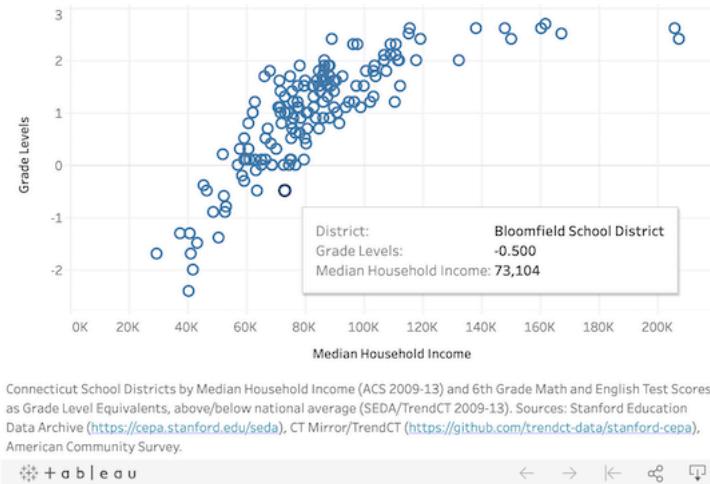


Figure 6.35: This scatterplot is made in Tableau Public and shows the relationship between household income and test scores in Connecticut school districts.

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 6.35, 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.

- 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 6.36. You will see the preview of the table under the drag-and-drop area. You have successfully connected one data source to Tableau Public, and you are ready to build your first chart.

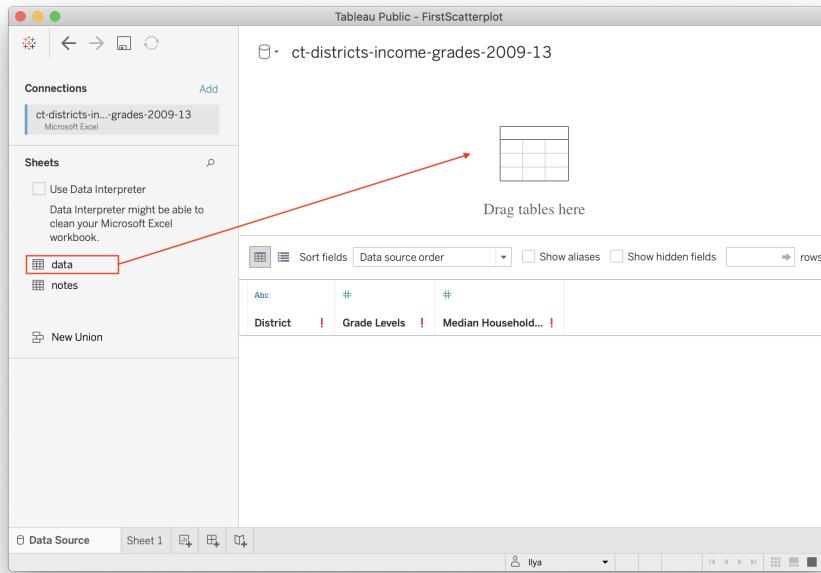


Figure 6.36: Drag **data** sheet into *Drag tables here* area.

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. Tableau marks all data fields as blue (discrete values, mostly text fields or numeric labels) or green (continuous values, mostly numbers).
4. Drag the *Grade Levels* field into the *Rows* field above the charting area, which for now is just empty space. You can consult Figure 6.37 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 data field.
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* dimension into the *Detail* box of the Marks card. 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.

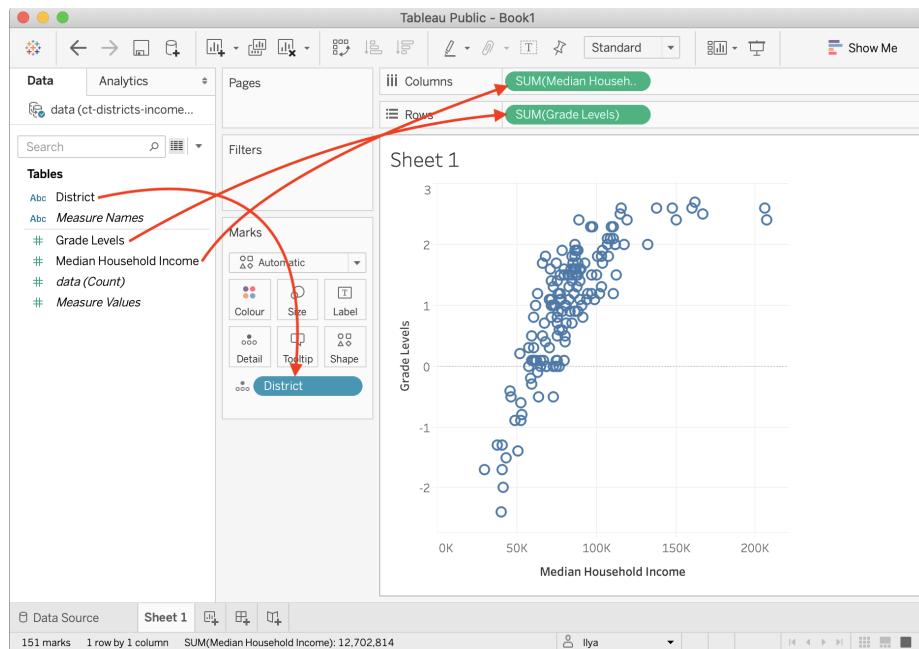


Figure 6.37: Drag data fields to the right places in Tableau.

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 6.38.

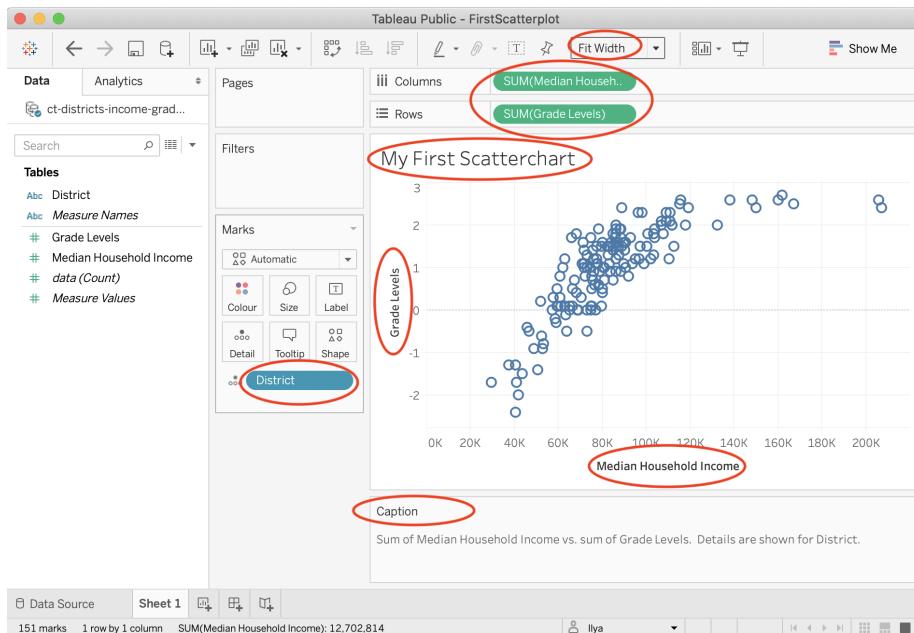


Figure 6.38: This scatter chart is ready to be published.

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 default *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 the title or description. Under *Toolbar Settings*, see checkbox to *Allow others to download or explore and copy this workbook and its data* (Figure 6.39), and enable/disable it as you think is appropriate. As advocates for open and accessible data, we recommend leaving the box checked.

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.

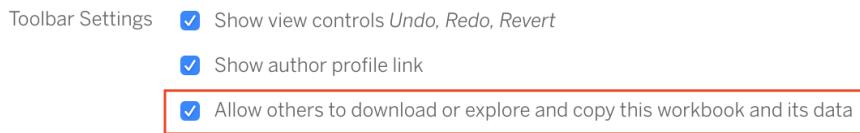


Figure 6.39: This scatter chart is ready to be published.

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

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 6.40. 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.

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

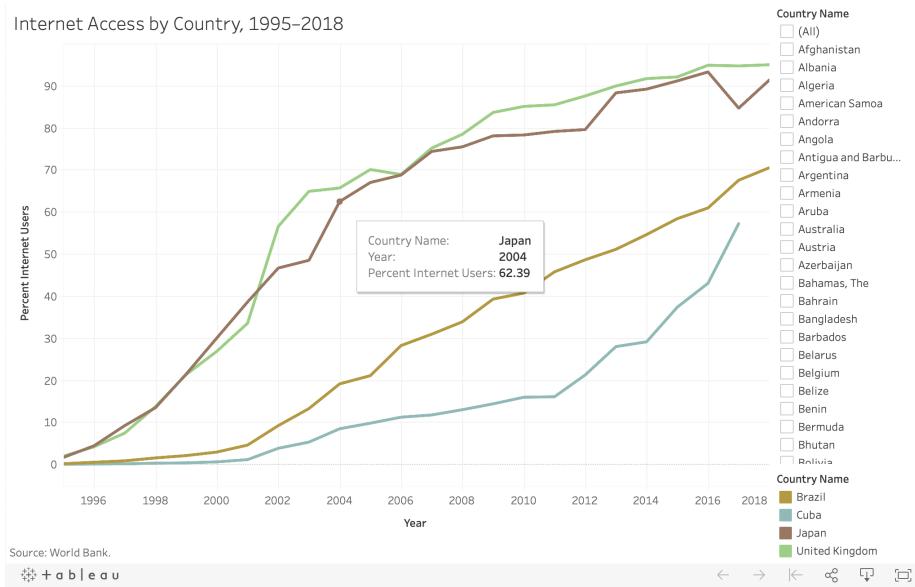


Figure 6.40: Internet Access by Country, 1995–2018.

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 6.41)
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.

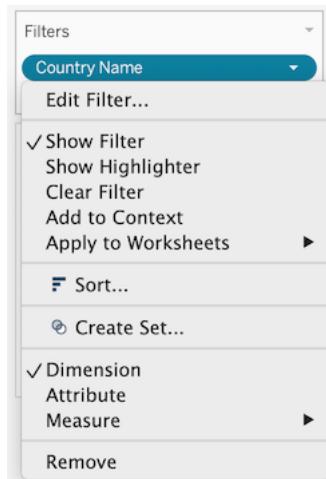


Figure 6.41: After you drag Country Name to the Filters card, make sure the Filter is displayed.

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 6.42, you are ready to publish.

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.

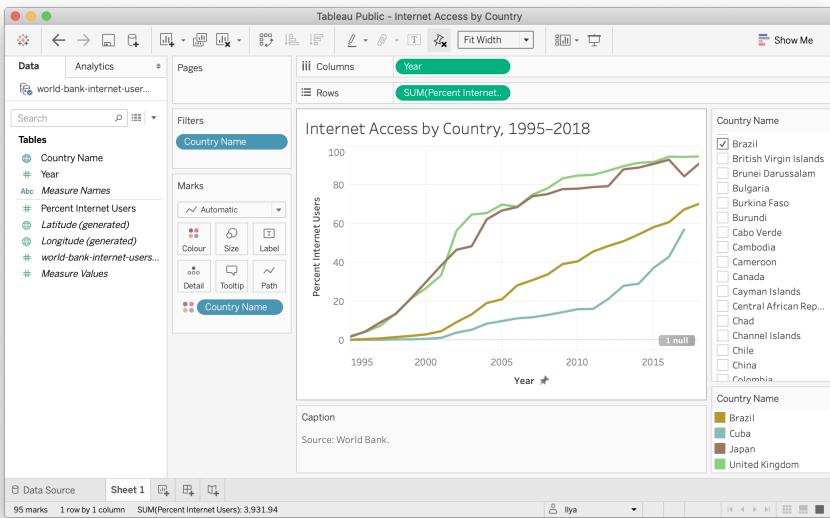


Figure 6.42: This workbook is ready to be published.

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 8.

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

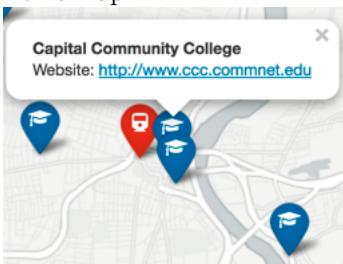
Chapter 7

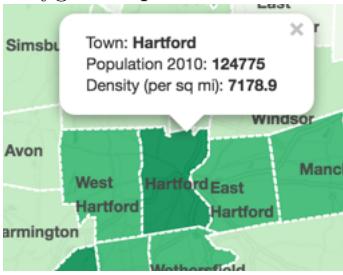
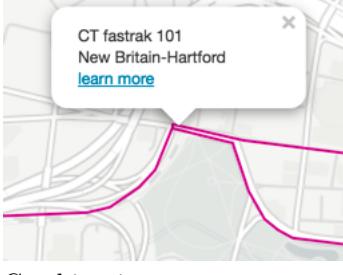
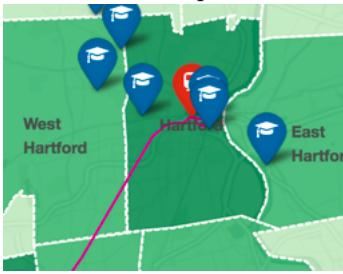
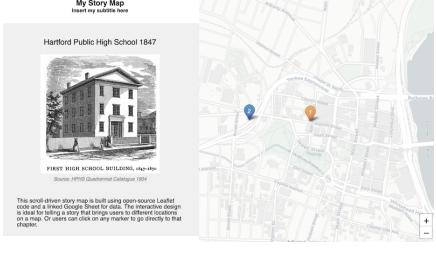
Map Your Data

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

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

Table 7.1: Basic Map Types and Tutorials

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

Map	Best use and tutorials in this book
Polygon map 	Best to show regions (such as nations or neighborhoods), with colors or shading to represent data values. Also known as choropleth map. Easy tool: n/a Power tools: Tableau Public or Leaflet Maps with Google Sheets and other Leaflet templates
Polyline map 	Best to show routes (such as trails or transit), with colors for different categories. Easy tool: n/a Power tool: Leaflet Maps with Google Sheets and other Leaflet templates
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
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 ABOVE:

- UPDATE table to match chapter contents.
- Add tab-view map for historical change template?
- Add synchronized side-by-side map template?

Map Design Principles

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

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

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

Understand the Vocabulary

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

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

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

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

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

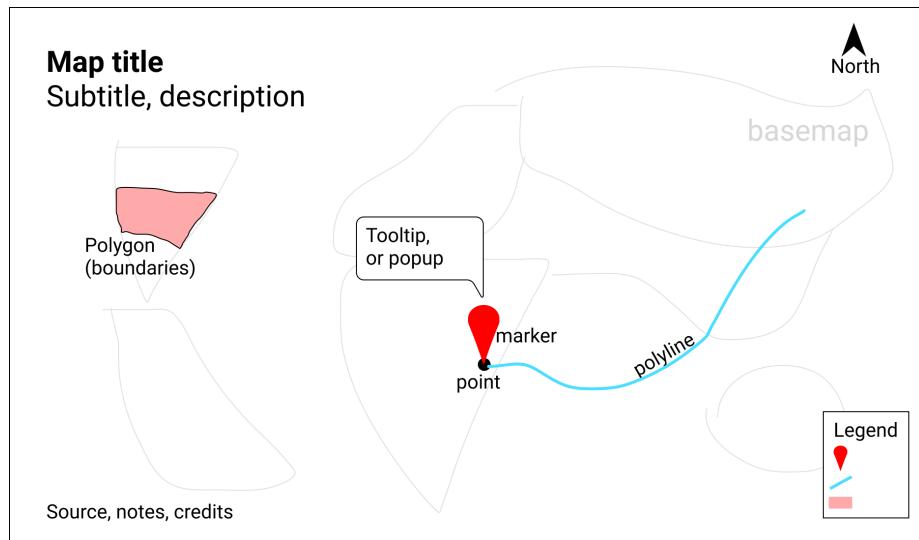


Figure 7.1: Map elements.

Color Palettes: Sequential, Diverging, and Qualitative

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

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

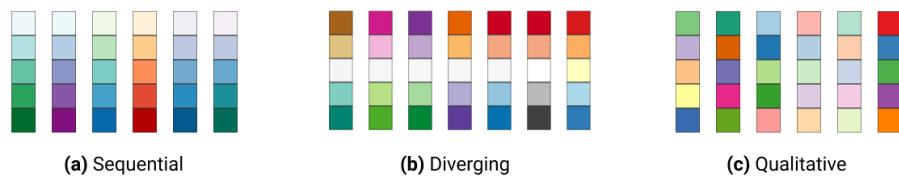


Figure 7.2: Examples of sequential, diverging, and qualitative color schemes from ColorBrewer.

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

Diverging palettes also represent continuous values. Unlike sequential palettes, however, diverging colors represent “direction” from some reference value, such

as below or above zero, or below or above the average value. Diverging palettes typically have two distinct hues for “positive” and “negative” values, with a neutral color in the middle.

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

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

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

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

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

ColorBrewer

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

ColorBrewer can generate color palettes for a specified number of classes - between three and nine for sequential, three and eleven for diverging, and three

Per-Capita Income in the United States, \$

Contiguous US only. Data by 2018 American Community Survey

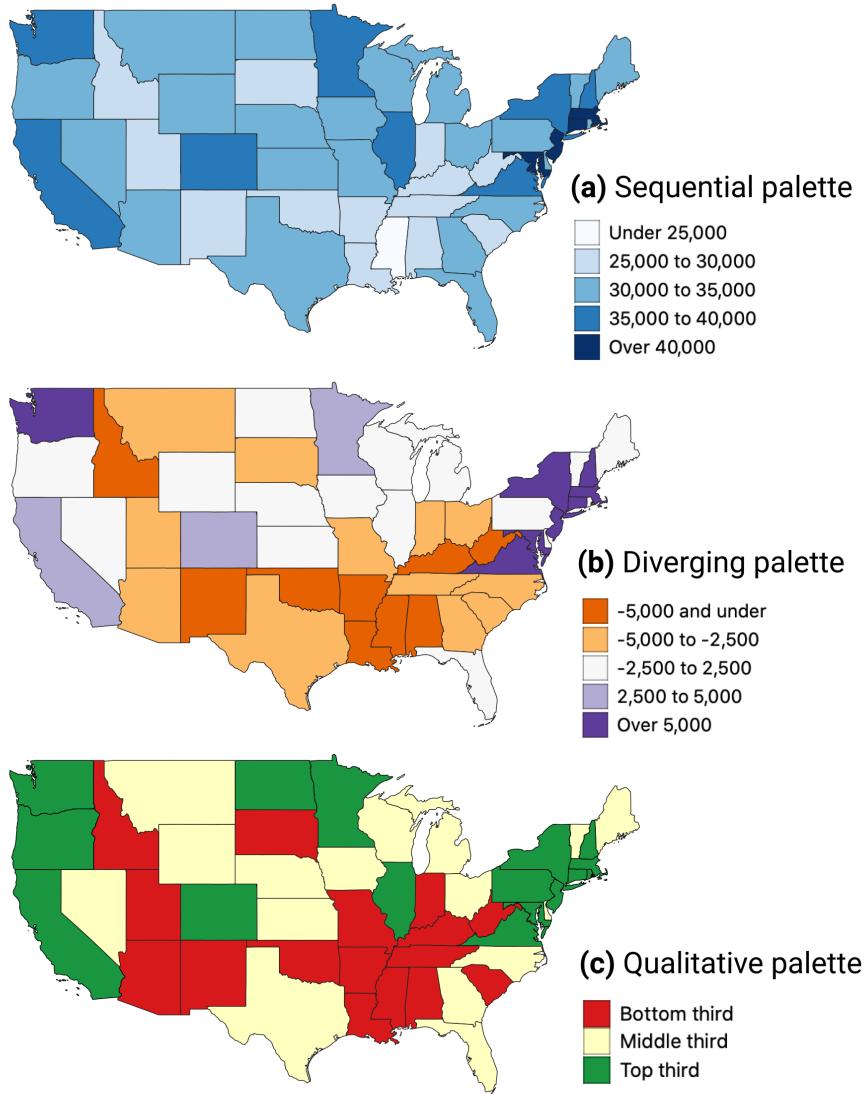


Figure 7.3: Representing per capita income in US states using three different classifications.

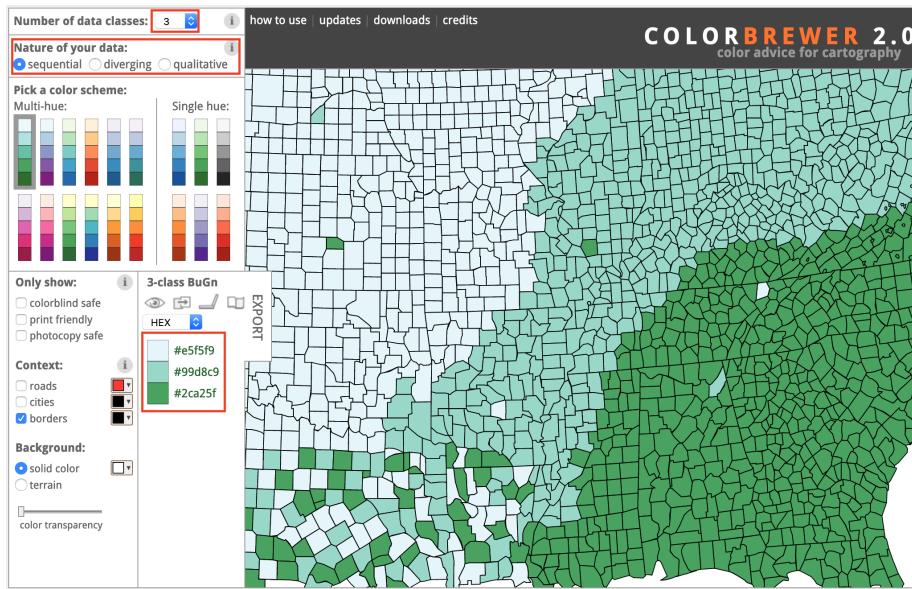


Figure 7.4: ColorBrewer interface.

and twelve for qualitative. You can also choose palettes that are colorblind safe and print friendly.

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

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

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

Point Map with Google My Maps

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

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

accounts to work on the map.

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

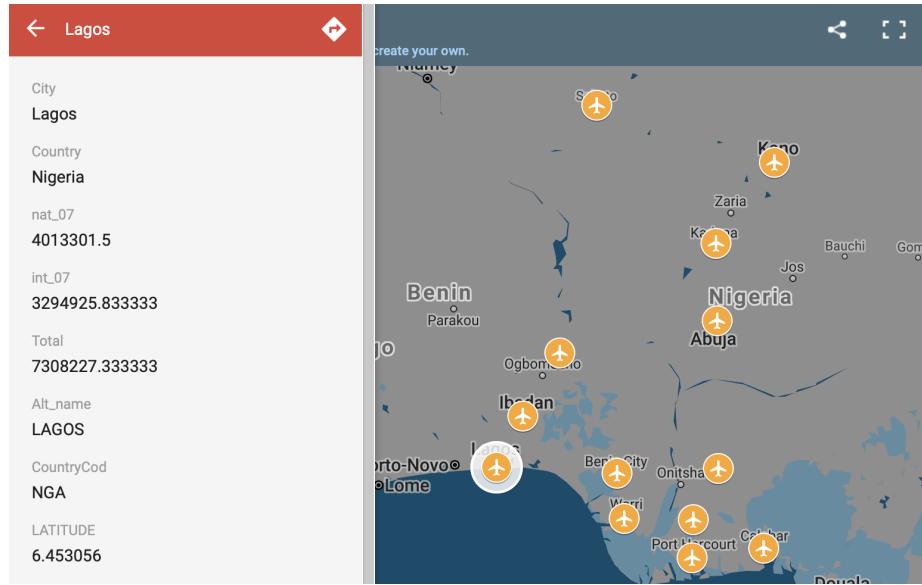


Figure 7.5: A map of airports in Nigeria built using Google My Maps.

Create a New Map in My Maps

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

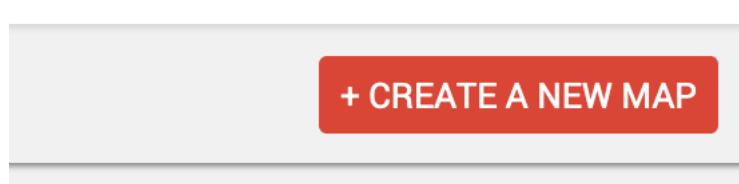


Figure 7.6: A map of airports in Nigeria built using Google My Maps.

You will see a typical Google Maps with no data. Click on the current title (*Untitled map*), and add appropriate title and description in the modal window that appeared (see Figure 7.7 for inspiration).

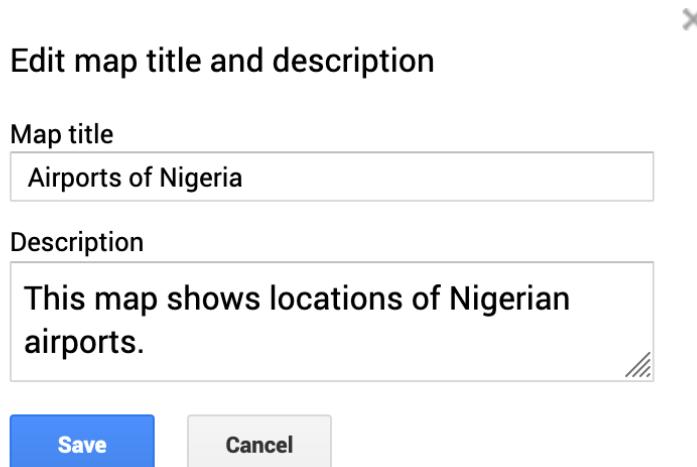


Figure 7.7: Add title and description to your map.

Before we add any points, let's change the basemap to something less boring. At the bottom of the control window, open *Base map* dropdown, and pick one of nine available basemaps. For this tutorial, we chose *Dark Landmass*.

Let's now proceed to the most important step—adding data. You can download a dataset of Nigerian airports that we got from the World Bank as a Shapefile and then converted to a CSV.

Under *Untitled layer* item, click *Import* button, and drag-and-drop the CSV file. Once the data file is uploaded, My Maps will ask which columns contain location data. In our case, these are *LATITUDE* and *LONGITUDE* columns, as shown in Figure 7.8.

Once the two boxes are checked, click *Continue*. Another window will pop up, asking which column to use to annotate points. Choose *City*, as shown in Figure 7.9, and then *Finish*.

It will take a few moments for My Maps to create a new layer, which will be added to the layer menu as *nigeria-airports.csv*. Once the layer is created, My Maps will center the map to fit the points.

Let's replace the original blue markers to orange airport symbols. In layers menu, hover over *All items* and click the paint bucket symbol on the right. Change "All items" text to "Airports", choose orange color, and click on *More icons* to find an airport symbol (we recommend using Filter to search for "air-

Choose columns to position your placemarks

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

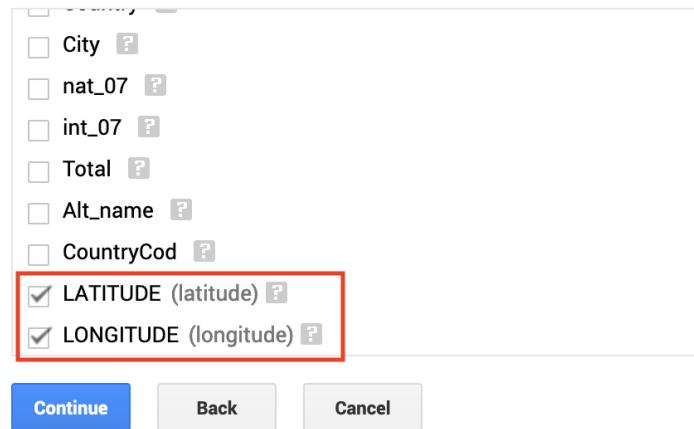


Figure 7.8: Check LATITUDE and LONGITUDE as your location columns.

Choose a column to title your markers

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

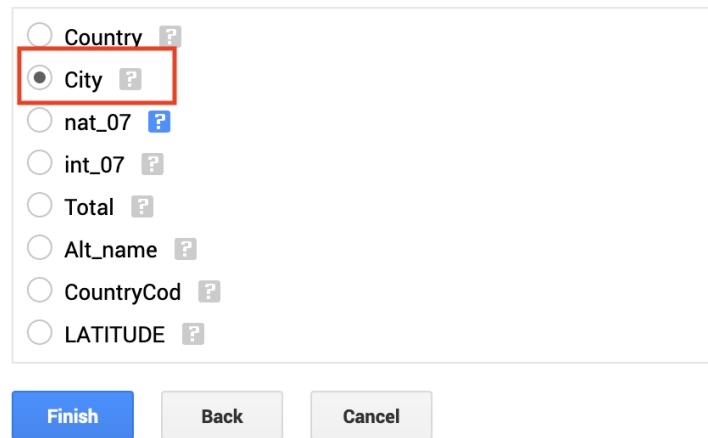


Figure 7.9: Choose City as the title for your markers.

port”, or simply scroll down to Transportation section). The marker in the layers menu will change to an orange airplane, as shown in Figure 7.10.

Click on the layer name, which by default is set to the name of imported file (`nigeria-airports.csv`), and change it to *Nigerian Airports*. Alternatively open a kebab menu to the right of the layer name, and choose *Rename this layer*.

You can accompany each marker with a label. Click *Uniform style* under the layer, and choose *Alt_name* in the *Set labels* dropdown menu. You will see alternative airport names, such as *BENIN*, displayed to the right of the markers.

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

Share Your Google Map

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

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

Going Beyond Points

Google My Maps has more powerful features for map making. Instead of uploading datasets with latitude/longitude pairs, you can use simple addresses (and My Maps will take care of geocoding), or add markers by clicking on the map using *Add marker* feature. You can classify points based on a property, and use different colors to represent them.

You are not limited to just point maps. You can also draw your own shapes, including lines and polygons. You can add data to multiple layers. Unfortunately, Google My Maps has no comprehensive documentation, so you have to explore the studio yourself if you want to create more complex projects.

Choropleth Map with Datawrapper

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

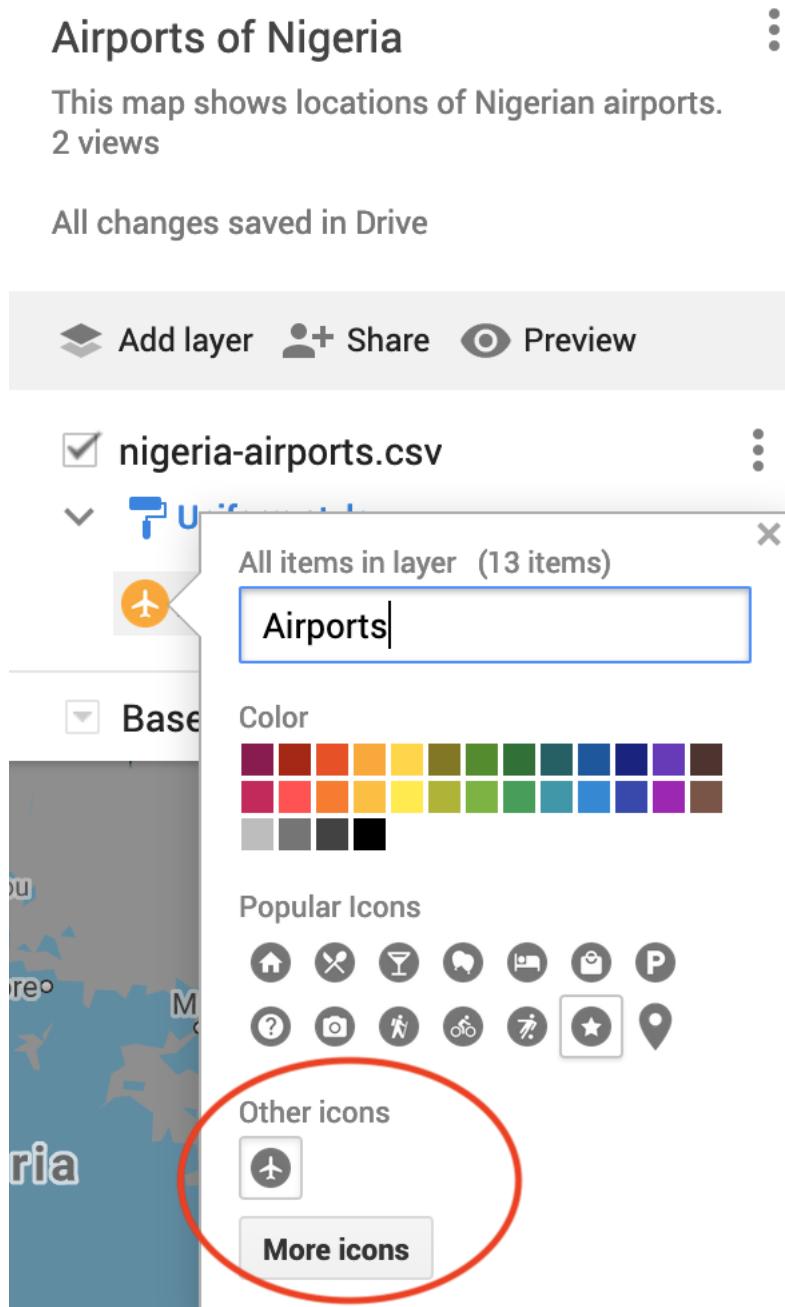


Figure 7.10: In My Maps, you can change marker colors and icons.

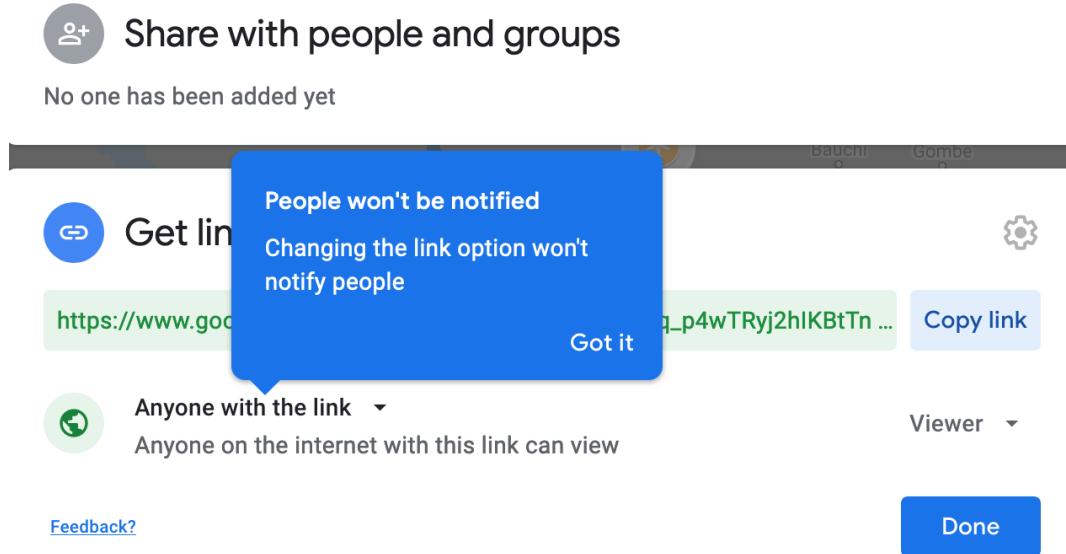


Figure 7.11: Make sure anyone with the link can view your map before you share it.

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

Create a New Choropleth Map

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

Datawrapper splits the process of creating a map into four steps. In step one, you need to choose your map boundaries. Datawrapper has a wide collection of most common geographical boundaries, including states and counties and municipalities for most countries in the world, and zip code and census tract areas for the United States. But you are not limited to the boundaries that Datawrapper already has. You can also upload your own custom geographies as a TopoJSON or GeoJSON file, and you will learn more about that in the Convert to GeoJSON section of chapter 12. Because we are mapping average home prices by US state, choose *United States > States*, as shown in Figure 7.15, and hit *Next* to go to step 2.

In the second step, you can attach data to the chosen boundaries. You can set values manually in the interactive table that Datawrapper offers. This is fine for several values, but is too time-consuming for all 50 values (District of Columbia

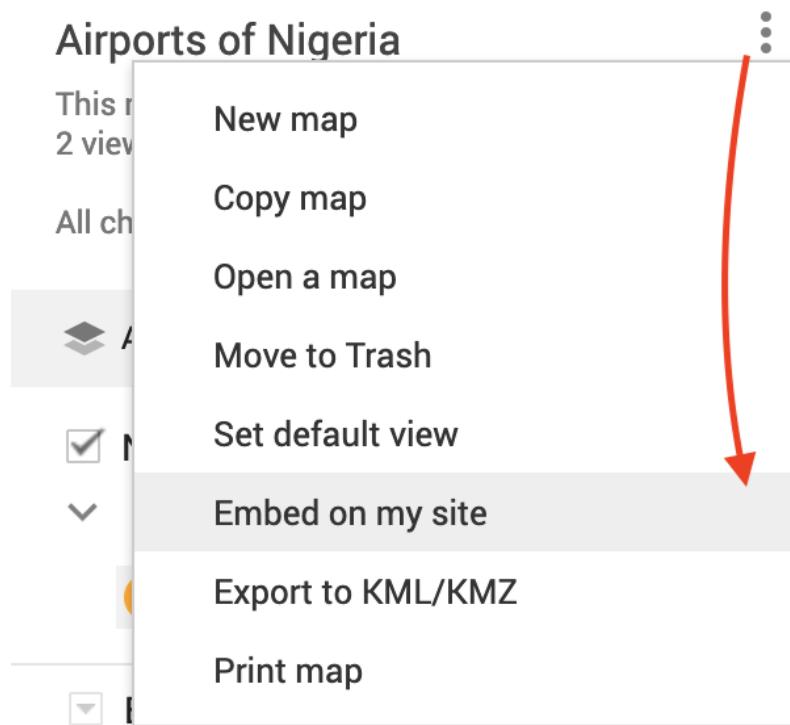
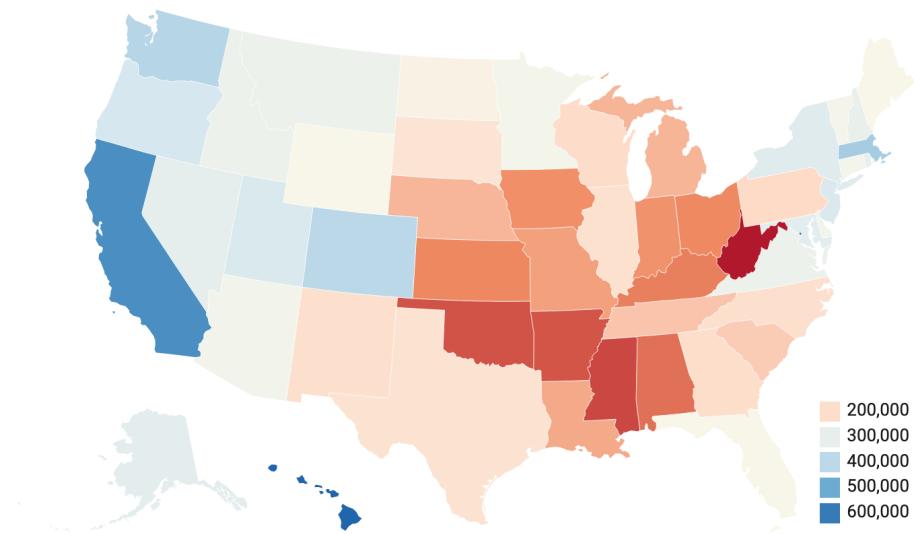


Figure 7.12: My Maps can generate an iframe code to include the map on your own website.

2019 Home Prices in the United States

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



Values are the average of 12 months in 2019

Map: HandsOnDataViz • Source: Zillow Data • Get the data • Created with Datawrapper

Figure 7.13: This choropleth map is created in Datawrapper

Datawrapper + New Chart **New Map** New Table River Login / Sign Up Language

Hi! What type of map do you want to create?

Choose the map type that will show your data best:

Choropleth map

Color regions to show data like unemployment rates or election results on a map. Upload your own map or use any of our more than 1000 maps. The resulting map is responsive & interactive.

Symbol map

Create symbols sized and colored according to your data. Works great for specific locations (like cities). Upload your own map or use any of our more than 1000 maps. The resulting map is responsive & interactive.

Locator map

Add markers to a map to show where something is located or happened, e.g. events within a city. Perfect for showing readers the places you mention in an article. The resulting map is responsive and static.

?

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

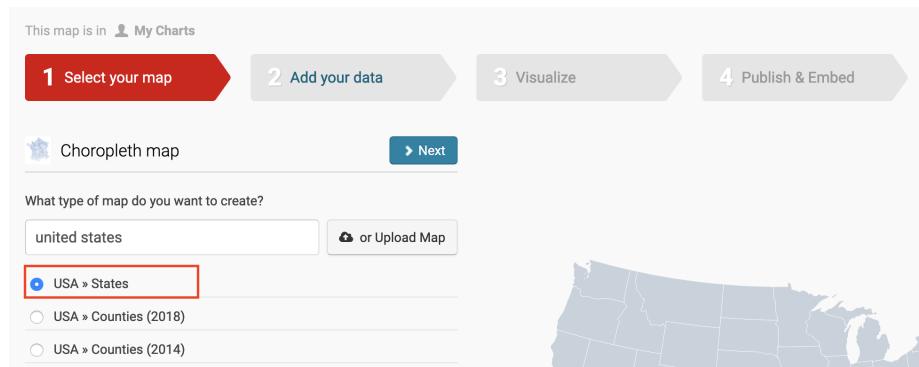


Figure 7.15: To map home prices by US state, choose appropriate boundaries.

is not a US state, so is not present in the boundaries). Instead, click *Import your dataset* button under the table.

Datawrapper will warn you that you need either *Names* (full state names, such as *Connecticut*), or *ISO-Codes* (the standardized two-letter codes for the state, such as *CT* for Connecticut) columns in order to perform merging. Since the dataset that we prepared contains both columns (titled *State* and *StateAbbr*), you can confidently press the *Start import* button.

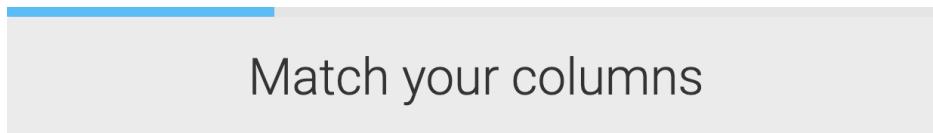
In the following window, you can either copy/paste values from the spreadsheet, or upload a CSV file. We recommend uploading a file, so click the link and choose the file *us-states-home-values-zillow-2019.csv*. The table will then get populated, and Datawrapper will ask for help identifying the relevant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

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

Style and publish your map

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

When finished with colors, scroll down to **Tooltips** section and click *Customize tooltips* button. In a popup window, set Title to the state name and Body to the average home value, as shown in Figure 7.18. Note that references to variables are put in double curly brackets (`{} ColumnNameGoesHere {}`). Make sure your



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

State	StateAbbr	HomeValue2019
California	CA	559861
Texas	TX	207201
New York	NY	325115
Florida	FL	244993

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

dollar symbol (\$) in outside of the curly brackets. Click *Save*, and hover over the map to make sure your tooltip displays the state name and the average home price.

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

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

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

Filtered Point Map with Socrata

Socrata is a database service that is used by government agencies, cities and countries to make open data available to the public. It offers user-friendly ways to view, filter, and export data. In addition, the Socrata platform includes built-in support to create interactive charts and maps, which can be embedded in other websites (including your own).

One advantage of creating data visualizations directly on an open data platform is that the chart or map is linked to the data repository. For example, if the Socrata platform administrator updates the data table, then a Socrata dataviz based on that data will be automatically updated, too. This may be especially

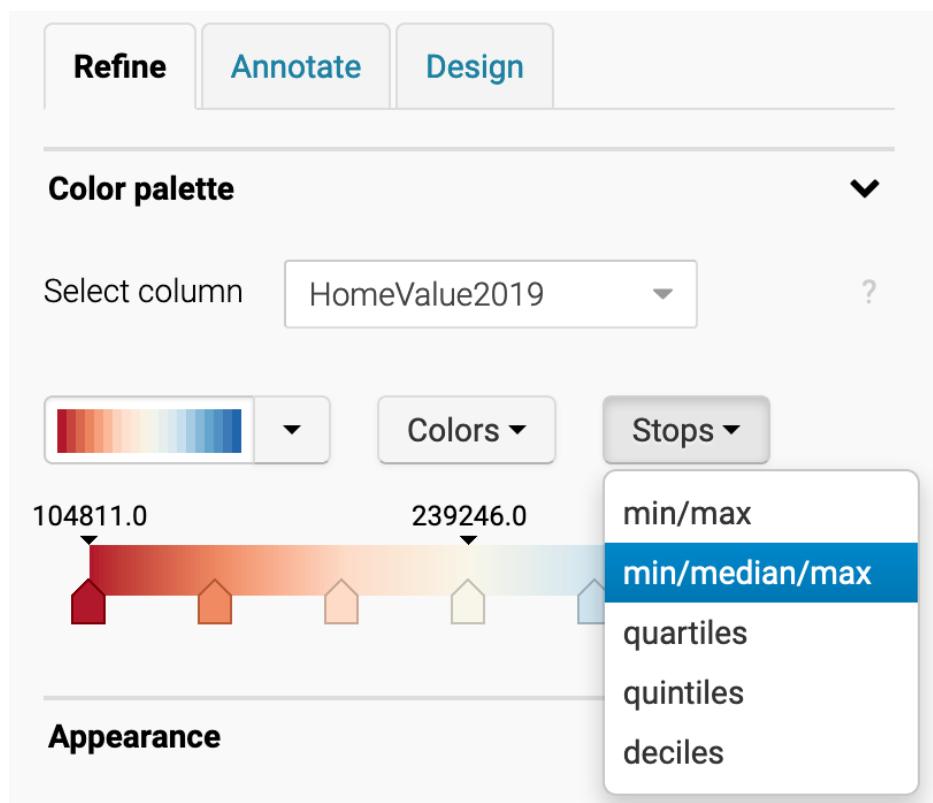


Figure 7.17: Choose red/blues divergent color scheme, and make sure to match median with the neutral (middle) value.

Customize tooltip HTML

Title
{{ State }}

Body
\${{ HomeValue2019 }}

StateAbbr HomeValue2019 State

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

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

Save

The screenshot shows a user interface for customizing tooltip HTML. On the left, there are two input fields: 'Title' containing '{{ State }}' and 'Body' containing '\${{ HomeValue2019 }}'. Both of these fields have red boxes around them, indicating they are the focus of the instructions. To the right, a list of column names is shown in blue buttons: 'StateAbbr', 'HomeValue2019', and 'State'. A large red oval surrounds these three buttons. Below this list is a explanatory text: 'Click on a column name above to add it to your tooltip. Tooltips work with placeholders. You can use them to show additional information when your users move their mouse over the map. The idea is:' followed by a code snippet: `{{ Country }} - Unemployment Rate: {{ Data }}`.

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

useful for “live” data that is continuously updated by agency administrators such as fires, crimes, and property data.

In this section, we will build an interactive point map of hospitals in Texas using General Hospital Information dataset by Medicare, which you can see in Figure 7.19.

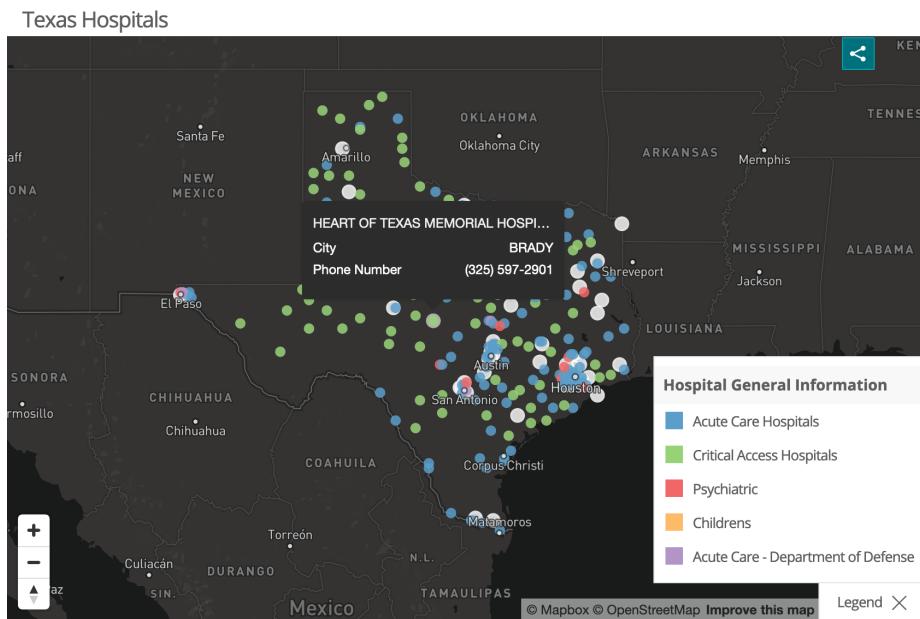


Figure 7.19: In this tutorial, we will build a point map of hospitals in Texas using Socrata.

Generally, in order to create a map in Socrata, you need to be a registered user, and the dataset you wish to visualize has to contain a column with location data. This is not just an address column (such as *3500 Gaston Avenue, Dallas, TX*), but a geocoded column that contains latitude and longitude values.

Sign Up for Socrata Account

Navigate to Data.Medicare.gov click *Sign In* button in the upper-right corner. Scroll down to *Sign Up* link. Follow the instructions, including setting up two-step authentication, to create a free account.

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

Once you have an account, log in using your credentials. Navigate to your profile by clicking your username in the upper-right corner and make sure you **Accept**

Terms and Conditions, otherwise you won't be able to save your draft map.

This username and password are only valid for Data.Medicare.gov, not other websites that use Socrata.

Create Your First Socrata Point Map

Navigate to the Hospital General Dataset, and in the menu on the right-hand side choose *Visualize > Launch New Visualization*, as shown in Figure 7.20. This will open up a *Configure Visualization* studio where you can create the map.

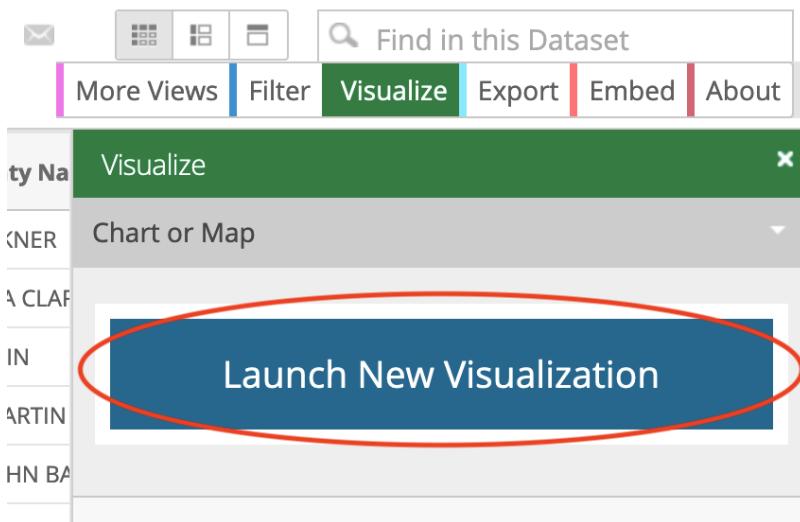


Figure 7.20: Go to Visualize > Launch New Visualization.

In the top menu, click *Map* (globe icon between a scatter chart icon and a calendar). You will see an updated layout of the studio, with the map in the middle, and *Map Layers* and *Map Settings* items in the side menu on the left.

Socrata was able to determine which column contains geospatial value, and automatically set Geo Column value to *Location* in the *Layer List* menu. By default, points are clustered (grouped together), so instead of seeing individual hospital locations, you see bubbles with numbers (such as 12 in Alaska).

Let's first select only hospitals that are located in the southern state of Texas. To do so, go to *Filters > Add filter*. The dropdown menu lists all columns (or fields) of the dataset, where we should choose *State*. In the newly appeared State dropdown, choose TX (for Texas) as shown in Figure 7.21, and scroll down and click *Apply*. Socrata should zoom in on the map and center on Texas. Close *Filters* window to free screen space.

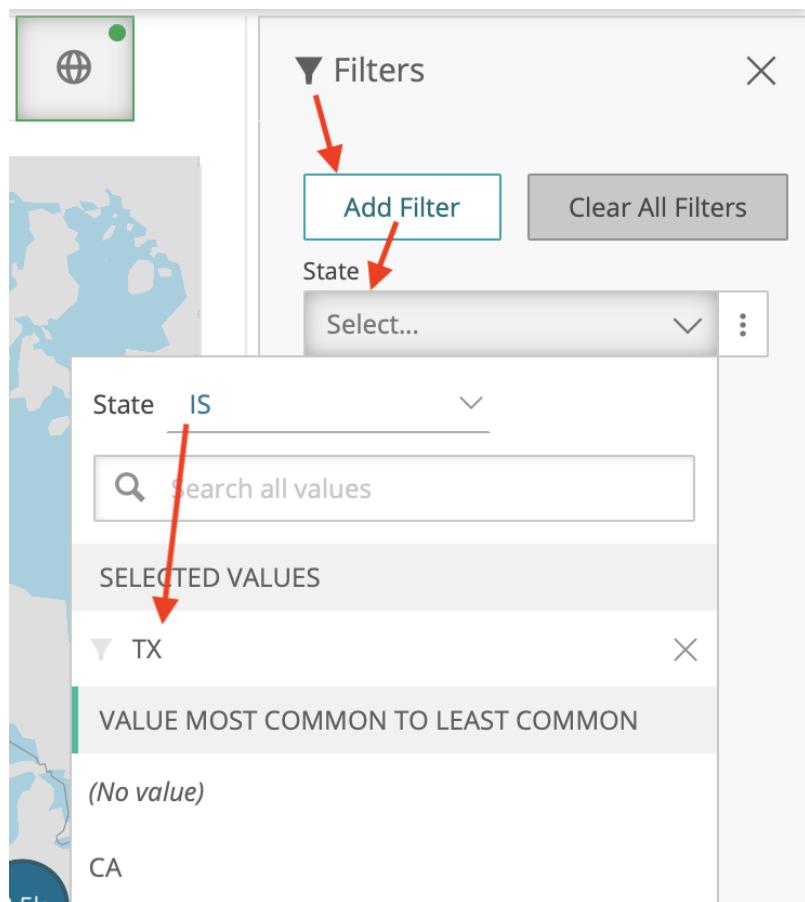


Figure 7.21: Select Texas as the only value for State field.

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

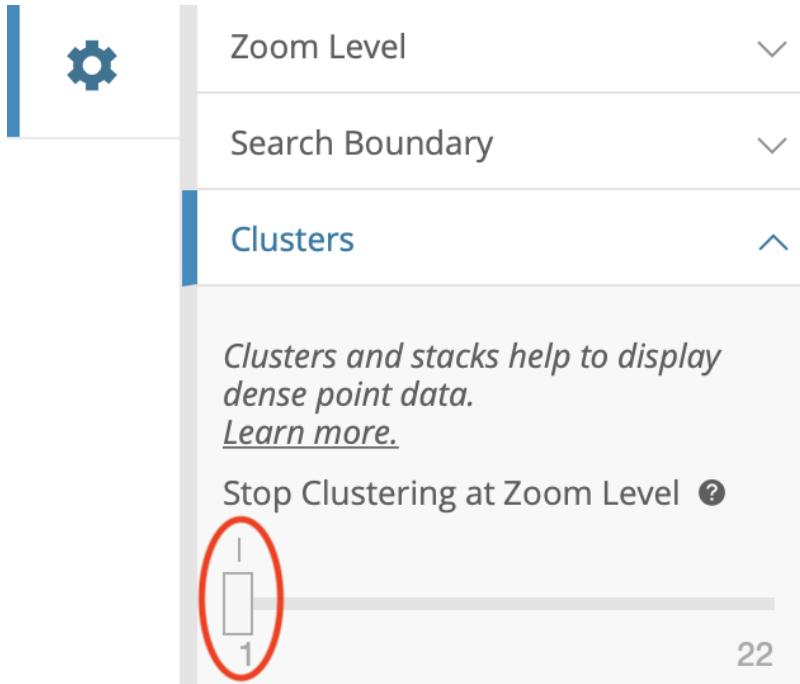


Figure 7.22: To show individual points instead of clusters, set Stop Clustering at Zoom Level to 1.

In the same accordion menu, change *Basemap > Type* to *Dark* to bring the map a fashionable 2020 look. In *General*, set Title to *Texas Hospitals*, and hide data table below the map by unchecking the *Show data table below visualization* box. Under *Map Controls*, uncheck *Show Search Bar* and *Show Locate Button* to get rid of unnecessary elements. Feel free to experiment with other settings as well.

Now, let's return back to **Map Layers** menu and choose our *Hospitals General Information* point layer. You can notice that in *Data Selection* accordion menu, *Resize Points by Value* is grayed out. That is because the dataset doesn't contain columns with continuous variables that can be transformed to point sizes. Instead, we can use *Style by Value* option to classify categorical points. The dataset contains multiple variables that can be effectively visualized, such as *Hospital Type*, *Emergency Services* (a yes/no category), *Mortality national comparison* and others. Let's stick with *Hospital Type*, as is illustrated in Figure 7.23.

If you look at the bottom-right corner of the map, you should notice a minimized

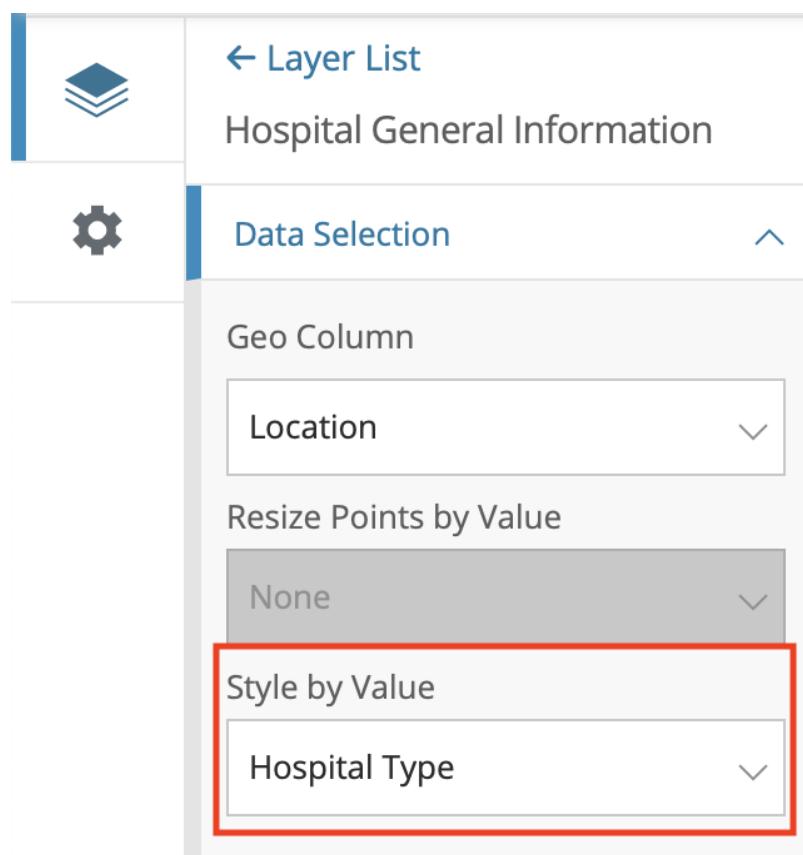


Figure 7.23: Let's display different types of hospitals in different colors.

Legend control. Click on it to see what each color represents.

Change the color palette (in *Color* menu) from *Categorical 1* to *Categorical 2*, which includes a wider range of unique colors. You can also use *Custom...* item to set individual colors, as well as change the order of categories in the legend.

To change what is shown in tooltips when you hover or click on points, go to **Flyout Details**, and set Flyout Title to *Facility Name*, adding city and phone number as additional flyout values, as is shown in Figure 7.24.

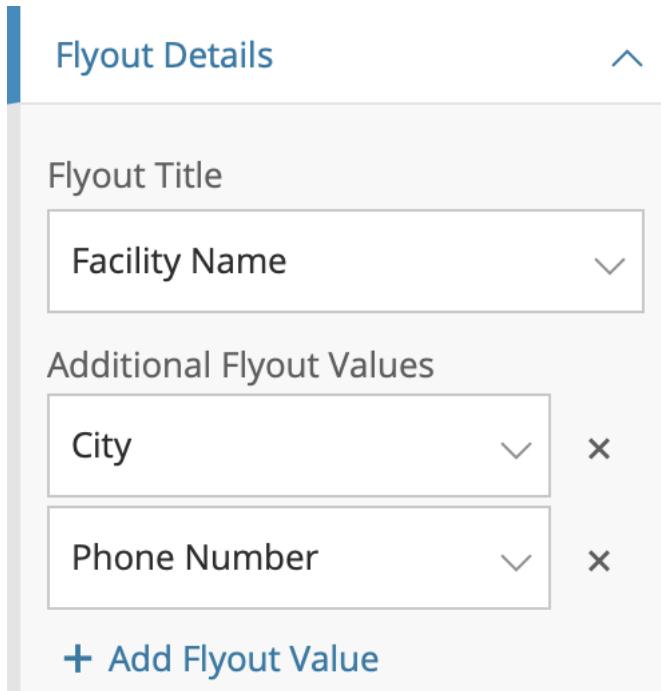


Figure 7.24: To edit tooltip information, use Flyout Details menu item.

At this point you should have a fully-functional interactive map showing all hospitals in Texas, colored according to their type. Before you can share it, you need to save it as a draft, and publish.

Save Draft and Publish

In the lower-right corner, click *Save Draft* button. Give your map a name, and hit *Save*. The gray ribbon at the top will tell you it is still a draft, and you can go ahead and *Publish...* it.

Now you can embed the map on your website as an iframe. To do so, click the *Share* button in the upper-right side of your map (see Figure 7.25), and copy

the generated code from *Embed Code* text area (Figure 7.26). See Chapter 7 of this book for further instructions about embedding.

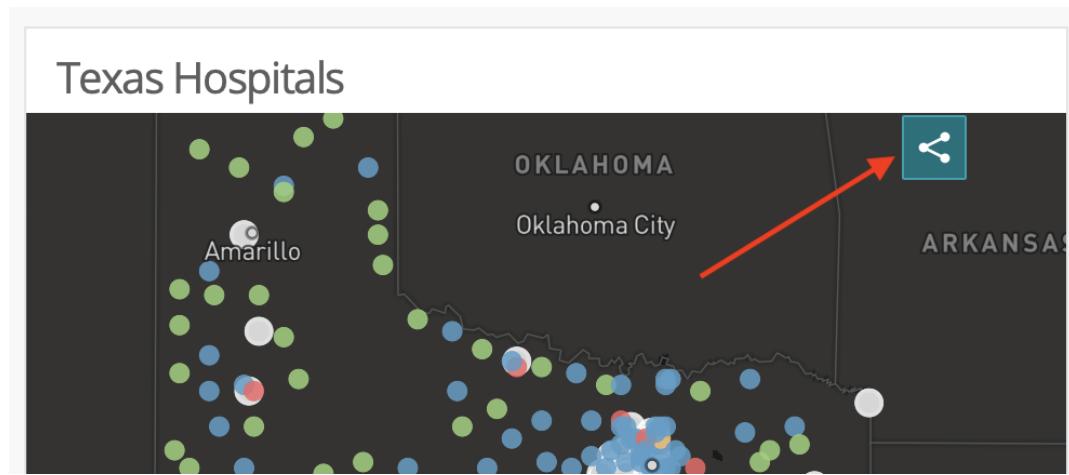


Figure 7.25: Click *Share* button to bring up *Share and Embed* window.

Limitations of Socrata

But there are limitations to creating your chart or map on an open data repository platform. First, if the agency stops using the platform, or changes the structure of the underlying data, your online map (or chart) may stop functioning. Second, you are generally limited to using datasets and geographic boundaries that already exist on that platform.

If these limitations concern you, a simple alternative is to export data from the open repository (which means that any “live” data would become “static”), and import it into your preferred dataviz tool, such as Tableau.

A second, more advanced alternative, is to learn to pull live data from Socrata using an API (Application Programming Interface). That requires coding skills that are beyond the scope of this book. Visit the official documentation to learn more about Socrata API.

Polygon Map with Tableau Public

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



Figure 7.26: Copy iframe code to embed this map in another website.

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

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

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

Create a Choropleth Map in Tableau

Launch Tableau. In **Connect** section on start up, choose *Text file*, and select **military-spending-2018.csv**. From Data Source tab, inspect the dataset contents. It contains four columns: Country Name, which includes countries and territories defined by the World Bank, the three-digit Country Code, Indicator Name (same for all rows), and percent value, as shown in Figure 7.28. Notice that some values are set to *null* (not available).

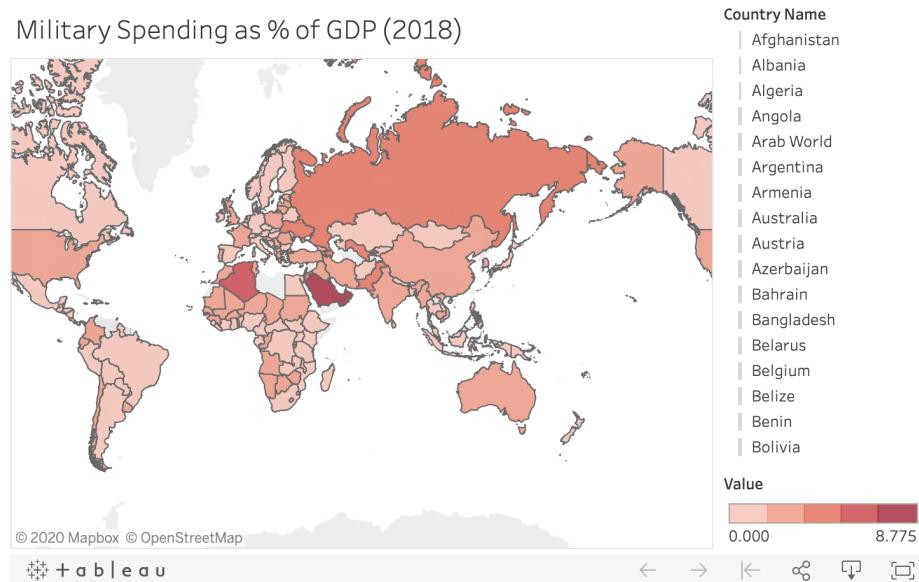


Figure 7.27: This choropleth map is made in Tableau Public

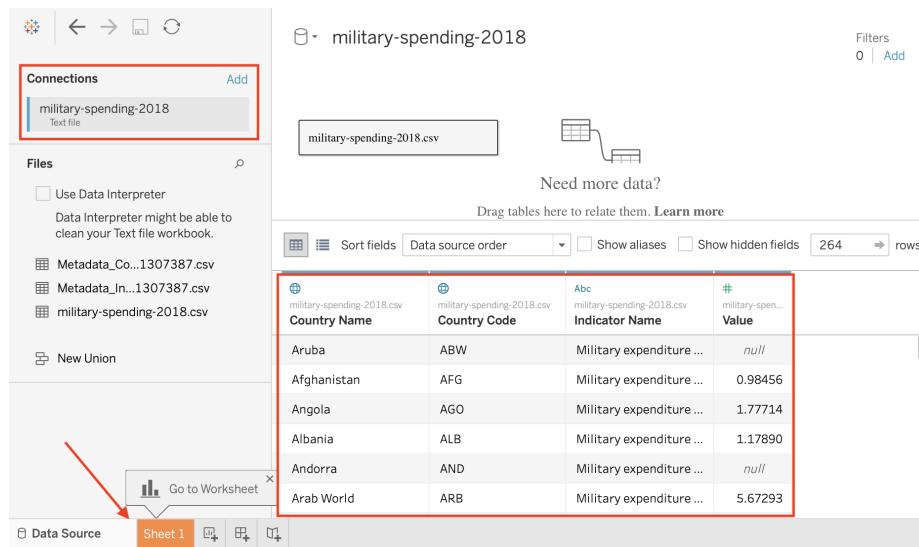


Figure 7.28: When finished inspecting the connected file, go to Sheet 1.

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

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

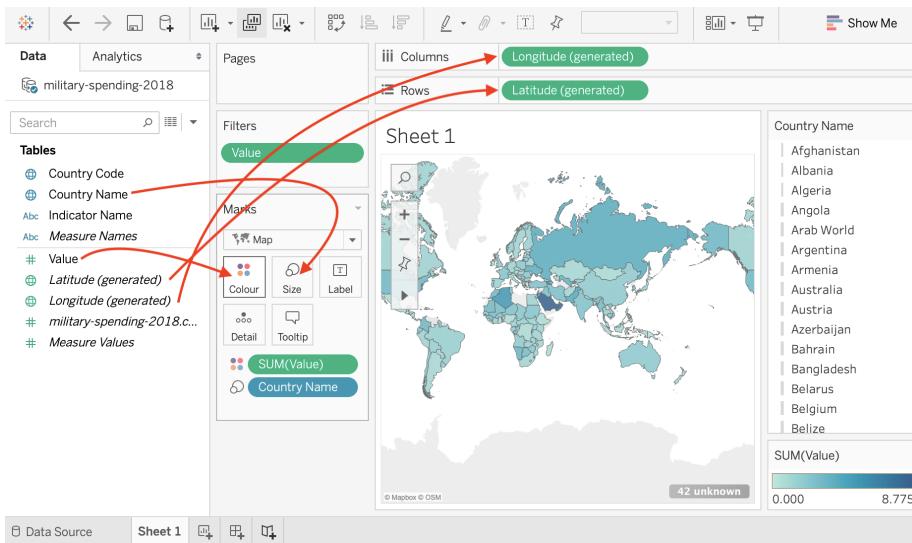


Figure 7.29: Drag and drop variables to the right places.

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

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

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

You may notice the tooltip calls values *Value* when hovering over countries. Click the Tooltip box of the Marks card to change text to *Military spending*, and add a percentage sign after the value itself, as shown in Figure 7.32. Make

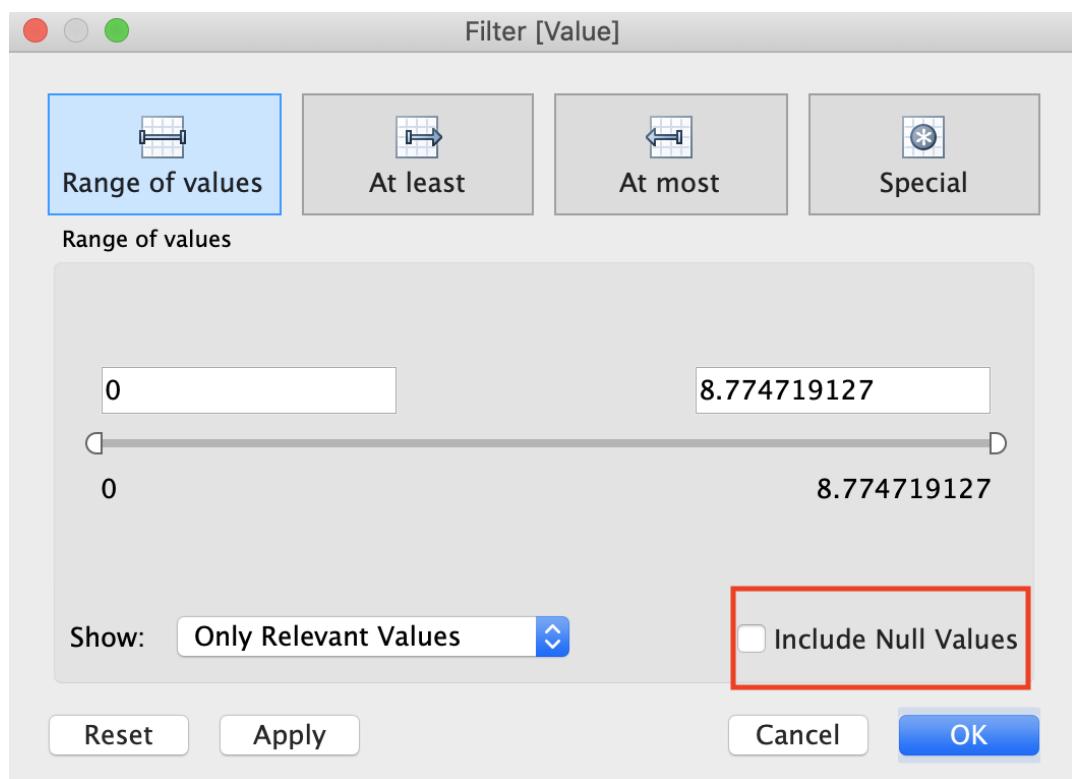


Figure 7.30: Filter values to remove countries with no data from display.

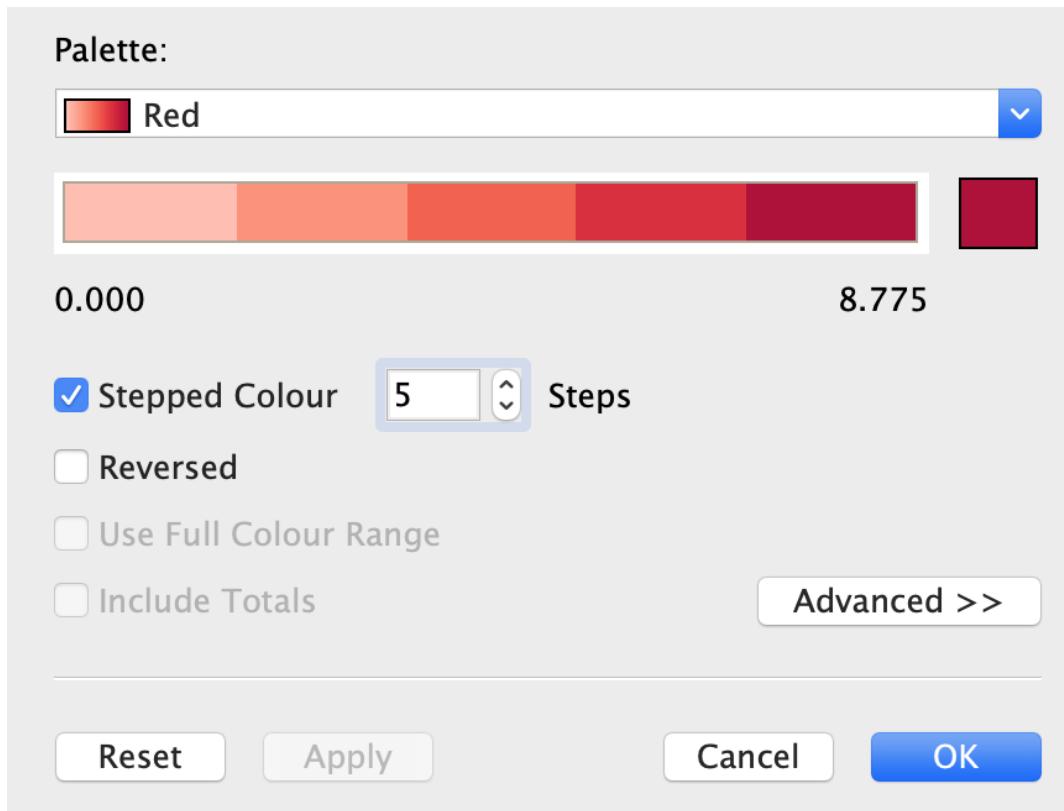


Figure 7.31: Change the color scheme to Reds with 5 steps.

sure not to change values between < and >, as these are references to variables.

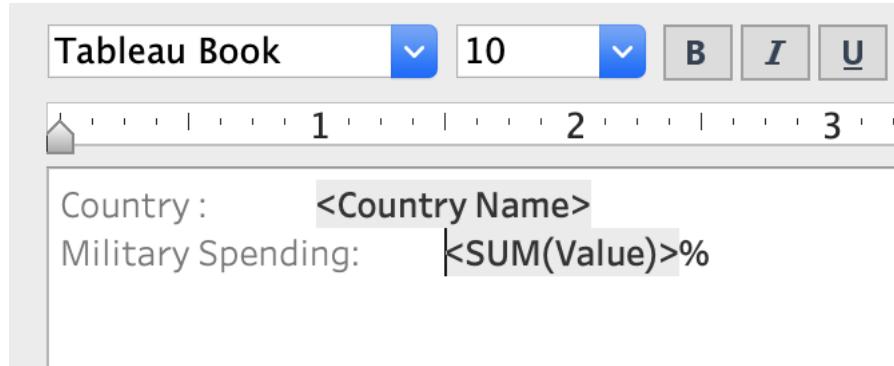


Figure 7.32: Change tooltip text to make it more user-friendly.

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

Publish Your Tableau Map

Once you are ready to publish and share the map, go to *File > Save to Tableau Public*. In the pop-up window, log in to your account if requested. Give it a title, such as *Military Spending*, and click *Save*.

Summary

In this chapter, we looked at free mapping platforms to create simple point and polygon maps. Google My Maps is a good choice for point maps that can be created in collaboration with others. If the data you are interested in lives on Socrata platform, you might be able to create a point map within the platform itself, and embed it as an iframe in your own website. Tableau is another very powerful tool to build and share polygon and point maps.

In reviewing all these tools, we only scratched the surface and showed simple examples to get you started quickly. All platforms allow layering data to create powerful exploration mapping visualizations.

None of the platforms required special geospatial data, as all were smart enough to perform geocoding and know the boundaries and coordinates of objects given to them. In Chapter 12, we will talk more about geospatial data, how it can be obtained, stored, modified, and shared.

Chapter 8

Embed On Your Web

TODO: Reorganize and rewrite chapter

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

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

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

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

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

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

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

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

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

Create a Simple Web Page with GitHub Pages

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

The full answer requires three steps:

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

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

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

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

Video

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

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

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

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

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

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

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

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

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

Copy an iframe code from a Google Sheets interactive chart

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

The full answer requires three steps:

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

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

Tutorial

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

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

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

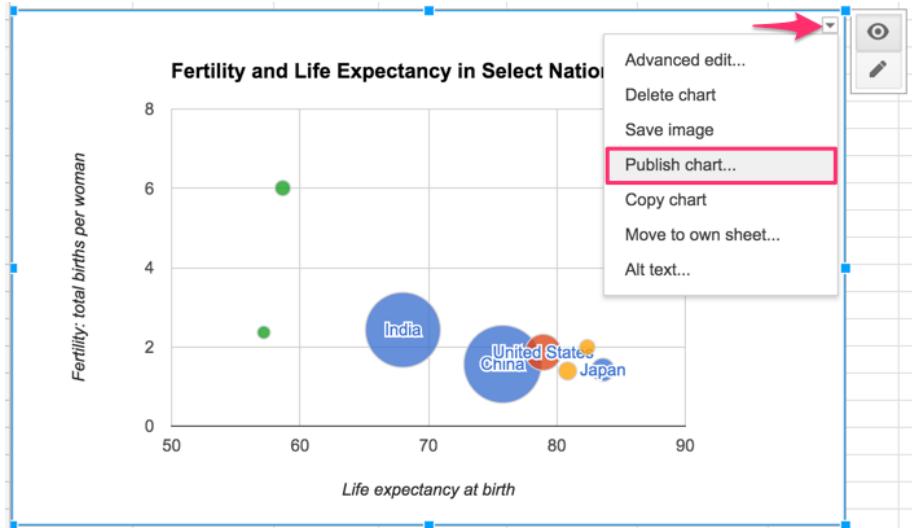


Figure 8.1: Screenshot: Drop-down menu to publish a Google Sheets chart

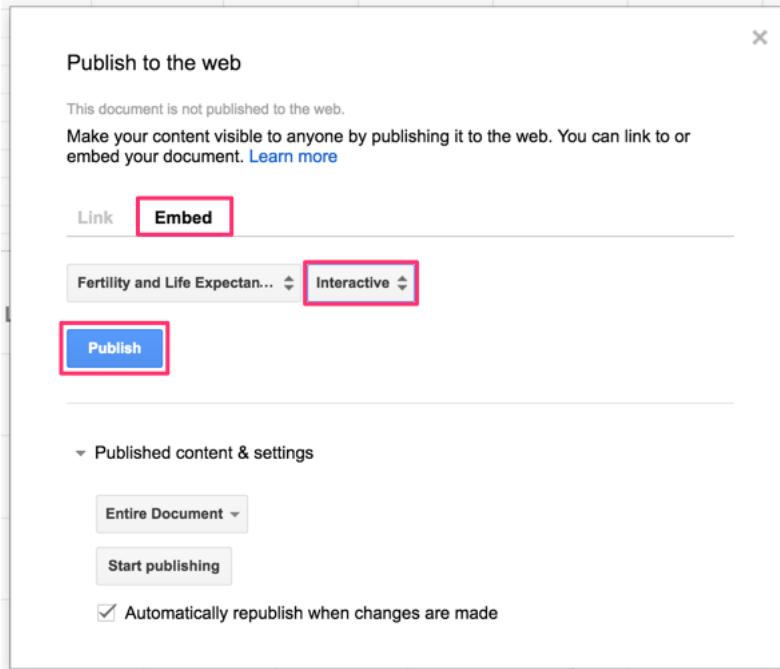


Figure 8.2: Screenshot: Publish to the web for a Google Sheets chart

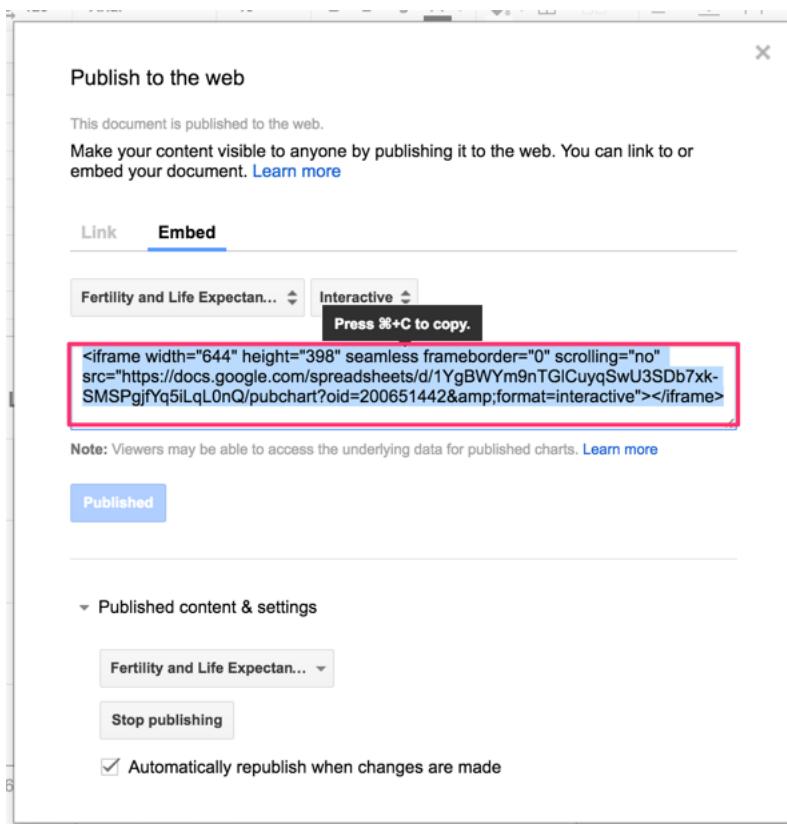


Figure 8.3: Screenshot: Copy the iframe code from a Google Sheets chart

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

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

Convert a Weblink into an Iframe

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

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

Published with a code template on GitHub Pages

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

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

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

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

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

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

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

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

Published with Google Sheets or Tableau Public

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

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

Embed an Iframe in GitHub Pages

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

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

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

Try it:

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

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

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

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

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

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

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

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

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

Important:

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

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

Embed an Iframe on WordPress.org

TODO:

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

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

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

To embed your dataviz in a self-hosted Wordpress.org site, the [iframe plugin] (<http://wordpress.org/plugins/iframe/>) must be installed and activated. This plugin allows authors to embed iframe codes inside posts/pages, in a modified "shortcode" format surrounded by square brackets. Without the plugin, self-hosted WordPress.org sites will usually "strip out" iframe codes for all users except the site administrator. **I have already installed and activated** the iframe plugin on my site, and the Dashboard view looks like this:



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

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

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

Select Text tab to embed iframe

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

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

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

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

replace with square bracket

replace the end tag with square bracket

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

Your modified code should look like this:

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

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

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

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

Embed Tableau Public on your Website

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

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

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

Try it:

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

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

A) Embed code method for HTML web pages

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

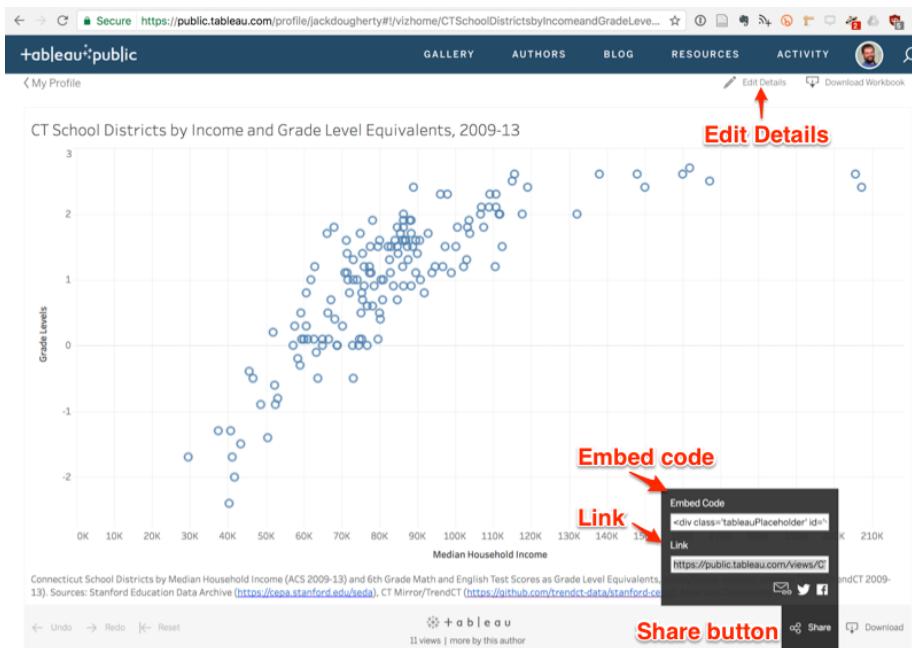


Figure 8.4: Screenshot: Edit and Share buttons in Tableau Public web page

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

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

```
<!DOCTYPE html>
<html>
<head>
    <title>sample web page</title>
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta charset="utf-8">
</head>
<body>
    <div class='tableauPlaceholder' id='viz1489158014225' style='position: relative'><noscript><a href="https://public.tableau.com/views/CTSchoolDistrictsbyIncomeandGradeLevel/CTSchoolDistrictsbyIncomeandGradeLevel?embed=true">View in Browser</a></noscript><div class='tableauPlaceholderContent'><div><img alt="Scatter plot of CT School Districts by Income and Grade Level Equivalents, 2009-13" data-bbox="180 180 720 450" /><div><div>Median Household Income</div><div>Grade Levels</div></div></div></div></div>
```

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 8.5: Screenshot: Edit and Share buttons in Tableau Public web page

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Chapter 9

Edit and Host Code with GitHub

In the first half of this book, you created interactive charts and maps on free drag-and-drop tool platforms created by companies such as Google and Tableau. These platforms are great for beginners, but their pre-set tools limit your options for designing and customizing your visualizations, and they also require you to depend upon their web servers and terms of service to host your data and work products. If these companies change their tools or terms, you have little choice in the matter, other than deleting your account and switching services, which means that your online charts and maps would appear to audiences as dead links.

In the second half of this book, get ready to make a big leap—and we'll help you through every step—by learning how to copy, edit, and host code templates. These templates are pre-written software instructions that allow you to upload your data, customize its appearance, and display your interactive charts and maps on a web site that you control. No prior coding experience is required, but it helps if you're *code-curious* and willing to experiment with your computer.

Code templates are similar to cookbook recipes. Imagine you're in your kitchen, looking at our favorite recipe we've publicly shared to make brownies (yum!), which begins with these three steps: `Melt butter`, `Add sugar`, `Mix in cocoa`. Recipes are templates, meaning that you can follow them precisely, or modify them to suit your tastes. Imagine that you copy our recipe (or “fork” it, as coders say) and insert a new step: `Add walnuts`. If you also publicly share your recipe, now there will be two versions of instructions, to suit both those who strongly prefer or dislike nuts in their brownies. (We do not take sides in this deeply polarizing dispute.)

Currently, the most popular cookbook among coders is GitHub, with more than 40 million users and over 100 million recipes (or “code repositories” or “repos”).

You can sign up for a free account and choose to make your repos private (like Grandma’s secret recipes) or public (like the ones we share below). Since GitHub was designed to be public, think twice before uploading any confidential or sensitive information that should not be shared with others. GitHub encourages sharing *open-source code*, meaning the creator grants permission for others to freely distribute and modify it, based on the conditions of the type of license they have selected.

When you create a brand-new repo, GitHub invites you to Choose a License. Two of the most popular open-source software licenses are the MIT License, which is very permissive, and the GNU General Public License version 3, which mandates that any modifications be shared under the same license. The latter version is often described as a *copyleft* license that requires any derivatives of the original code to remain publicly accessible, in contrast to traditional *copyright* that favors private ownership. When you fork a copy of someone’s open-source code on GitHub, look at the type of license they’ve chosen (if any), keep it in your version, and respect its terms.

To be clear, the GitHub platform is also owned by a large company (Microsoft purchased it in 2018), and when using it to share or host code, you’re also dependent on its tools and terms. But the magic of code templates is that you can migrate and host your work anywhere on the web. You could move to a competing repository-hosting service such as GitLab, or purchase your own domain name and server space through one of many web hosting services. Or you can choose a hybrid option, such as hosting your code on GitHub and choosing its custom domain option, to display it under a domain name that you’ve purchased, just like the web version of this book is hosted on GitHub under our domain name, <https://HandsOnDataViz.org>. If we choose to move the code away from GitHub, we have the option to repoint our domain to a different web host.

In the next section of this chapter, we will introduce basic steps to copy, edit, and host a simple Leaflet map code template on GitHub. Later you’ll learn how to create a new GitHub repo and upload code files.

This chapter introduces GitHub using its web browser interface, which works best for beginners. Later you’ll learn about intermediate-level tools, such as GitHub Desktop and Atom Editor, to work more efficiently with code repos on your personal computer.

If problems arise, turn to the Fix Common Mistakes section in the appendix. All of us make mistakes and accidentally “break our code” from time to time, and it’s a great way to learn how things work—and what to do when it doesn’t work!

Copy, Edit, and Host a Simple Leaflet Map Template

Now that you understand how GitHub code repositories are like a public cookbook of recipes, which anyone can copy and modify, let's get into the kitchen and start baking! In this section, we'll introduce you to a very simple code template 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 you'll learn about GitHub in this section:

- Make a copy of our simple Leaflet map code template
- Edit the map title, start position, background layer, and marker
- Host a live online version of your modified map code on the public web

Your goal is to create your own version of this simple interactive map, with your edits, as shown in Figure 9.1.



Figure 9.1: Create your own version of this simple interactive Leaflet map.

1. Create your own free account on GitHub. It may ask you to do a simple quiz to prove you're a human! If you don't see a confirmation message in your email, check your spam folder.

Tip: Choose a GitHub username that's relatively short, and one that you'll be happy seeing in the web address of charts and maps you'll publish online. In other words, `DrunkBrownieChef6789` may not be the wisest choice for a username, if `BrownieChef` is also available.

2. After you log into your GitHub account in your browser, go to our simple Leaflet map template at <https://github.com/HandsOnDataViz/leaflet-map-simple>
3. Click the green *Use this template* button to make your own copy of our repo, as shown in Figure 9.2.

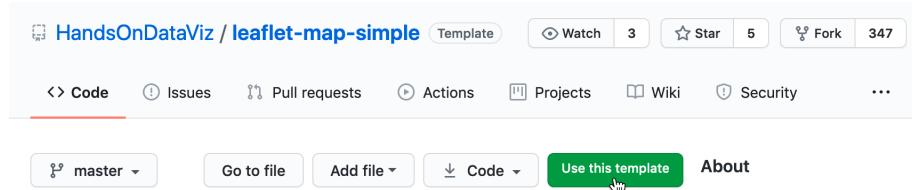


Figure 9.2: Click *Use this template* to make your own copy.

4. On the next screen, your account will appear as the owner. Name your copy of the repo **leaflet-map-simple**, the same as ours, as shown in Figure 9.3. Click the green *Create repository from template* button.

Create a new repository from leaflet-map-simple

The new repository will start with the same files and folders as [HandsOnDataViz/leaflet-map-simple](#).

Owner * Repository name *

/ leaflet-map-simple ✓

Great repository names are short and memorable. Need inspiration? How about [ubiquitous-disco](#)?

Description (optional)

Public
Anyone on the internet can see this repository. You choose who can commit.

Private
You choose who can see and commit to this repository.

Include all branches
Copy all branches from HandsOnDataViz/leaflet-map-simple and not just master.

Create repository from template

Figure 9.3: Name your copied repo **leaflet-map-simple**.

Note: We set up our repo using GitHub's template feature to make it easier for users to create their own copies. If you're trying to copy someone else's GitHub

repo and don't see a *Template* button, then click the *Fork* button, which makes a copy a different way. Here's the difference: *Template* allows you to make *multiple* copies of the same repo by giving them different names, while *Fork* allows you to create *only one copy* of a repo because it uses the same name as the original, and GitHub prevents you from creating two repos with the same name. If you need to create a second fork of a GitHub repo, go to the Create a New Repo and Upload Files on GitHub section of this chapter.

The upper-left corner of the next screen will say `USERNAME/leaflet-map-simple` generated from `HandsOnDataViz/leaflet-map-simple`, where `USERNAME` refers to your GitHub account username. This confirms that you copied our template into your GitHub account, and it contains only three files:

- `LICENSE` shows that we've selected the MIT License, which allows anyone to copy and modify the code as they wish.
- `README.md` provides a simple description and link to the live demo, which we'll come back to later.
- `index.html` is the key file in this particular, because it contains the map code.

5. Click on the `index.html` file to view the code, as shown in Figure 9.4.

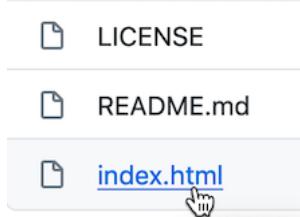


Figure 9.4: Click the `index.html` file to view the code.

If this is the first time you're looking at computer code, it may feel overwhelming, but relax! We've inserted several "code comments" to explain what's happening. The first block 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.

6. To edit the code, click on the the pencil symbol in the upper-right corner, as shown in Figure 9.5.

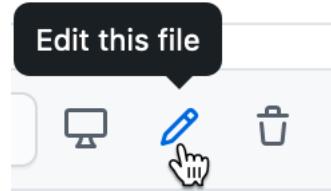


Figure 9.5: Click the pencil button to edit the code.

Let's start by making one simple change to prove to everyone that you're now editing *your* map, by modifying the map title, which appears in the HTML division tag block around lines 21-23.

7. In this line `<div id="map-title">EDIT your map title</div>`, type your new map title in place of the words `EDIT your map title`. Be careful not to erase the HTML tags that appear on both ends inside the `< >` symbols.
8. To save your edit, scroll to the bottom of the page and click the green *Commit Changes* button, as shown in Figure 9.6.

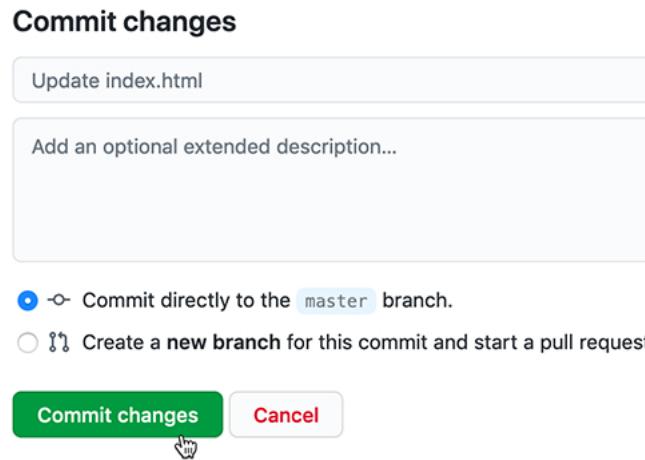


Figure 9.6: Click the green *Commit Changes* button to save your edits.

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

Now let's publish your edited map to the public web to see how it looks in a web browser. GitHub not only stores open-source code, but its built-in GitHub Pages 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.

9. To access GitHub Pages, scroll to the top of your repo page and click the *Settings* button as shown in Figure 9.7.

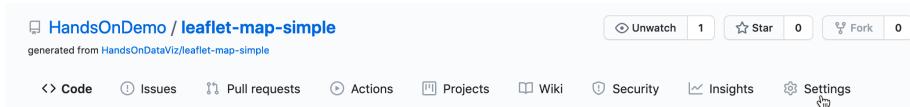


Figure 9.7: Click the *Settings* button to access GitHub Pages and publish your work on the web.

10. In the Settings screen, scroll way down to the GitHub Pages area. In the drop-down menu, change *Source* from *None* to *Master*, keep the default */(root)* option in the middle, and press *Save* as shown in Figure 9.8. This step tells GitHub to publish a live version of your map on the public web, where anyone can access it in their browser, if they have the web address.

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

11. Scroll back down to *Settings > GitHub Pages* to see the web address where your live map has been published online, and right-click it to open in a new browser tab, as shown in Figure 9.9.

Now you should have at least two tabs open in your browser. The first tab contains your GitHub repo, where you edit your code, with a web address in this format, and replace **USERNAME** and **REPOSITORY** with your own:

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

GitHub Pages

GitHub Pages is designed to host your personal, organization, or project pages from a GitHub repository.

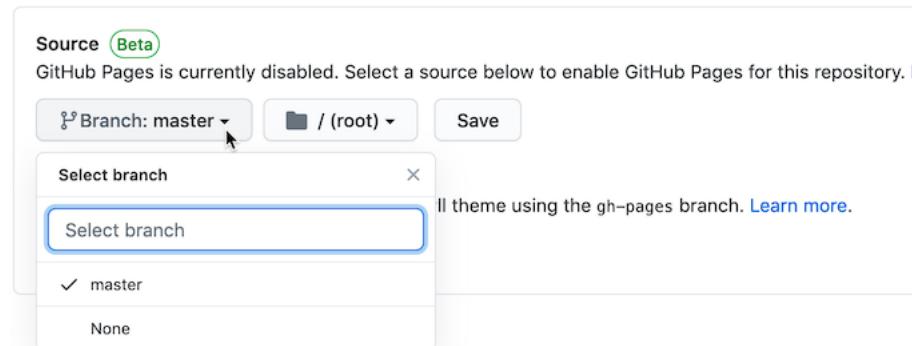


Figure 9.8: In *Settings*, go to *GitHub Pages*, and switch the source from *None* to *Master*.

GitHub Pages

GitHub Pages is designed to host your personal, organization, or project pages from a GitHub repository.

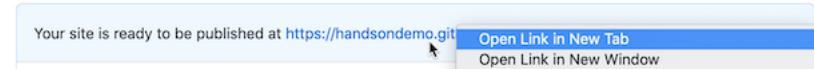


Figure 9.9: In *Settings* for *GitHub Pages*, right-click your published map link to open in a new tab.

The second tab contains your GitHub Pages live website, where your edited code appears online. GitHub Pages automatically generates a public web address in this format:

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

Remember how we told you not to create your account with a username like `DrunkBrownieChef6789`? GitHub automatically places your username automatically in the public web address.

Keep both tabs open so you can easily go back and forth between editing your code and viewing the live results online.

Note: The live version of your code points to the `index.html` page by default, so it's not necessary to include it in the web address. Also, web addresses are *not* case sensitive, meaning you can save time by typing all of it in lower-case.

Tip: If your live map does *not* appear right away, wait up to 30 seconds for GitHub Pages to finish processing your edits. Then give your browser a “hard refresh” to bypass any saved content in your cache and re-download the entire web page from the server, using one of these key combinations:

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

Now let's edit your the GitHub repo so that the link points to *your* live map, instead of *our* live map.

12. Copy the web address of your live map from your second browser tab.
13. Go back to your first browser tab with your GitHub repo, and click on the repo title to return to its home page, as shown in Figure 9.10.



Figure 9.10: On your first browser tab, click the repo title.

14. On your repo page, click to open the `README.md` file, and click the pencil again to edit it, as shown in Figure 9.11. Paste your live web link under

the label (*replace with link to your site*) and scroll down to commit the change.

The screenshot shows a GitHub repository page for 'HandsOnDemo'. At the top, it displays a commit from 'HandsOnDemo' made 1 hour ago with the commit hash 'bafe0cec'. Below this, there are three files listed: 'LICENSE', 'README.md', and 'index.html', all with 'Initial commit' status and '1 hour ago' timestamp. The 'README.md' file is currently selected, showing its content. The content of the README is:

```
leaflet-map-simple
```

A simple Leaflet map template for new users to fork their own copy, edit, and host on GitHub Pages

Link to live map (replace with link to your site)

<https://handsondataviz.github.io/leaflet-map-simple/>

Figure 9.11: Open and edit the README file to paste the link to your live map.

Now that you've successfully made simple edits and published your live map, let's make more edits to jazz it up and help you learn more about how Leaflet code works.

15. On your repo home page, click to open the `index.html` file, and click the pencil symbol to edit more code.

Wherever you see the EDIT code comment, this points out a line that you can easily modify. For example, look for the code block shown below that sets up the initial center point of the map and its zoom level. Insert a new latitude and longitude coordinate to set a new center point. To find coordinates, right-click on any point in Google Maps and select *What's here?*, as described in the Geocode Locations into Coordinates section of Chapter 12.

```
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}.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
```

15. After making edits, remember to scroll down and press the *Commit* button to save changes. Then go to your browser tab with the live map, and do a hard-refresh to view changes. If your map edits do not appear right away, remember that GitHub Pages sometimes requires up to 30 seconds to process code edits. If you have problems, see the Fix Common Mistakes section in the appendix.

Congratulations! If this is the first time that you've edited computer code and hosted it online, you can now call yourself a "coder". The process is similar to following and modifying a cookbook recipe, just like you also can call yourself a "chef" after baking your first batch of brownies! Although no one is likely to hire you as a full-time paid coder (or chef) at this early stage, you now understand several of the basic skills needed to copy, edit, and host code online, and you're ready to dive into the more advanced versions, such as Chart.js and Highcharts templates in chapter 10 and Leaflet map templates in chapter 11.

The next section describes how to enhance your GitHub skills by creating new repos and uploading your files. These are essential steps to create a second copy of a code template or to work with more advanced templates in the next two chapters.

Create a New Repo and Upload Files on GitHub

Now that you've made a copy of our GitHub template, the next step is to learn how to create a brand-new repo and upload files. These skills will be

helpful for several scenarios. First, if you have to fork a repo, which GitHub allows you to do only one time, this method will allow you to create additional copies. Second, you'll need to upload some of your own files when creating data visualizations using Chart.js and Highcharts templates in Chapter 10 and Leaflet map templates in Chapter 11. Once again, we'll demonstrate how to do all of these steps in GitHub's beginner-level browser interface, but see the next section on GitHub Desktop for an intermediate-level interface that's more efficient for working with code templates.

In the previous section, you created a copy of our GitHub repo with the *Use this template* button, and we intentionally set up our repos with this newer feature because it allows the user to make *multiple* copies and assign each one a different name. Many other GitHub repos do not include a *Template* button, so to copy those you'll need to click the *Fork* button, which automatically generates a copy with the same repo name as the original. But what if you wish to fork someone's repo a second time? GitHub prevents you from creating a second fork to avoid violating one of its important rules: every repo in your account must have a unique name, to avoid overwriting and erasing your work.

So how do you make a second fork of a GitHub repo, if there's no *Use this template* button? Follow our recommended workaround that's summarized in these three steps:

- Download the existing GitHub repo to your local computer
 - Create a brand-new GitHub repo with a new name
 - Upload the existing code repo files to your brand-new repo
1. Click on the *Code > Download Zip* drop-down menu button on any repo, as shown in Figure 9.12. Your browser will download a zipped compressed folder with the contents of the repo to your local computer, and it may ask you where you wish to save it. Decide on a location and click OK.
 2. Navigate to the location on your computer where you saved the folder. Its file name should end with `.zip`, which means you need to double-click to “unzip” or de-compress the folder. After you unzip it, a new folder will appear named in this format, `REPOSITORY-BRANCH`, which refers to the repository name (such as `leaflet-map-simple`) and the branch name (such as `master` or `main`), and it will contain the repo files.
 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 9.13.
 4. On the next screen, GitHub will ask you to enter a new repo name. Choose a short one, preferably all lower-case, and separate words with hyphens if

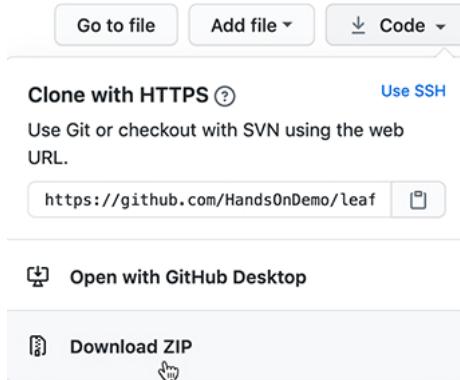


Figure 9.12: Click *Code* and select *Download Zip* to create a compressed folder of a repo on your computer.

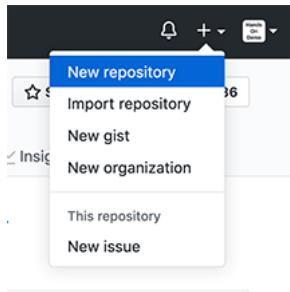


Figure 9.13: Click the plus (+) symbol in upper-right corner to create a new repo.

needed. Let's name it `practice` because we'll delete it at the end of this tutorial.

Check the box to *Initialize this repository with a README* to simplify the next steps.

Also, select *Add a license* that matches the code you plan to upload, which in this case is *MIT License*. Other fields are optional. Click the green *Create Repository* button at the bottom when done, as shown in Figure 9.14.

Create a new repository

A repository contains all project files, including the revision history. / elsewhere? [Import a repository](#).

Repository template
Start your repository with a template repository's contents.

No template ▾

Owner * **Repository name ***

HandsOnDemo / practice ✓

Great repository names are short and memorable. Need inspiration?

Description (optional)

Public
Anyone on the internet can see this repository. You choose who can

Private
You choose who can see and commit to this repository.

Skip this step if you're importing an existing repository.

Initialize this repository with a README
This will let you immediately clone the repository to your computer.

Add .gitignore: None ▾ Add a license: MIT License ⓘ

Create repository

Figure 9.14: Name your new repo `practice`, check the box to *Initialize this repo with a README*, and *Add a license* (select *MIT*) to match any code you plan to upload.

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

5. On your new repo home page, click the *Add File > Upload Files* drop-down menu button, near the middle of the screen, as shown in Figure 9.15.
6. Drag-and-drop the `index.html` file that you previously downloaded to your local computer into the upload screen of your GitHub repo in your browser, as shown in Figure 9.16. 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.

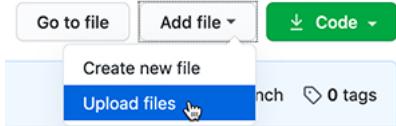


Figure 9.15: Click the *Upload Files* button.

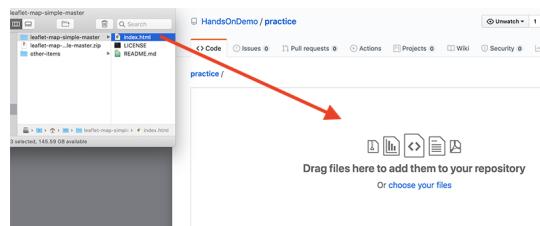


Figure 9.16: Drag-and-drop the `index.html` file to the upload screen.

When the upload is complete, your repo should contain three files, now including a copy of the `index.html` code that you previously downloaded from the `leaflet-map-simple` template. This achieved our goal of working around GitHub's one-fork rule, by creating a new repo and manually uploading a second copy of the code.

Optionally, you could use GitHub Pages to publish a live version of the code online, and paste the links to the live version at the top of your repo and your `README.md` file, as described in the Copy, Edit, and Host a Simple Leaflet Map Template section of this chapter.

7. Since this was only a `practice` repo, let's delete it from GitHub. In the repo screen of your browser, click the top-right *Settings* button, scroll all the way down to the *Danger Zone*, and click *Delete this repository*, as shown in Figure 9.17. GitHub will ask you to type in your username and repo name to ensure that you really want to delete the repo, to prove you are not a drunken brownie chef.

So far, you've learned how to copy, edit, and host code using the GitHub web interface, which is a great introduction for beginners. Now you're ready to move up to tools that will allow you to work more efficiently with GitHub, such as GitHub Desktop and Atom Editor, to quickly move entire repos to your local computer, edit the code, and move them back online.

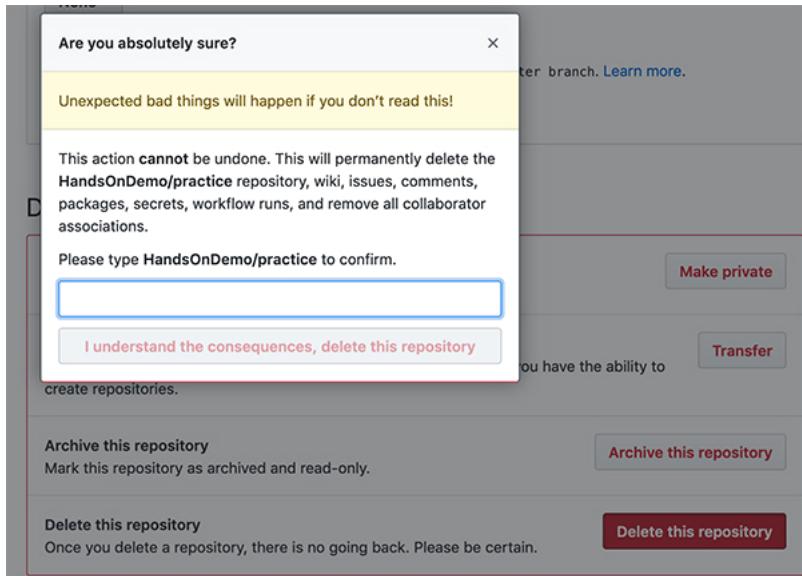


Figure 9.17: After clicking the Delete Repository button, GitHub will ask you to type your username and repo name to confirm.

GitHub Desktop and Atom Editor to Code Efficiently

Editing your code through the GitHub web interface is a good way to start, 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 Copy, Edit, and Host a Simple Leaflet Map Template section of this chapter. Click the *Add file > Open with GitHub Desktop* drop-down menu button near the middle of your screen, as shown in Figure 9.18. The next screen will show a link to the GitHub Desktop web page, and you should download and install the application.

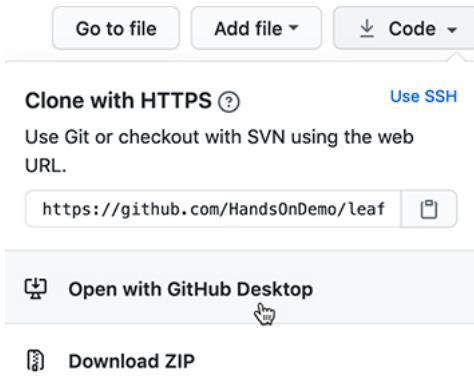


Figure 9.18: In your GitHub repo on the web, click *Add file* to *Open with GitHub Desktop* to download and install GitHub Desktop.

2. When you open GitHub Desktop for the first time, you'll need to connect it to the GitHub web account you previously created in this chapter. On the welcome screen, click the blue *Sign in to GitHub.com* button, as shown in Figure 9.19, and login with your GitHub username and password. On the next screen, GitHub will ask you to click the green *Authorize desktop* button to confirm that you wish to connect to your account.
3. In the next setup screen, GitHub Desktop asks you to configure Git, the underlying software that runs GitHub. Confirm that it displays your username and click *Continue*, as shown in Figure 9.20.
4. On the “Let’s Get Started” with GitHub Desktop screen, click on *Your Repositories* on the right side to select your `leaflet-map-sample`, and further below click the blue button to *Clone* it to your local computer, as shown in Figure 9.21.

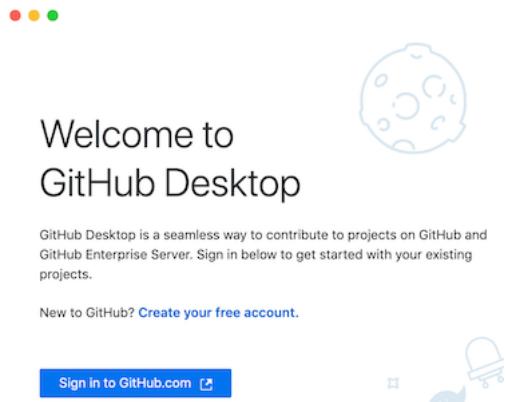


Figure 9.19: Click the blue *Sign in to GitHub.com* button to link GitHub Desktop to your GitHub account.

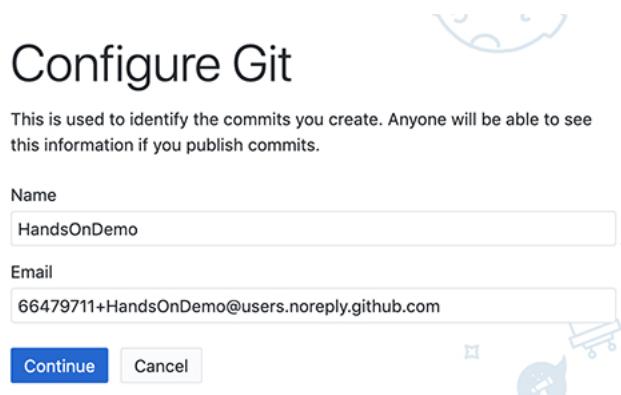


Figure 9.20: Click the *Continue* button to authorize GitHub Desktop to send commits to your GitHub account.

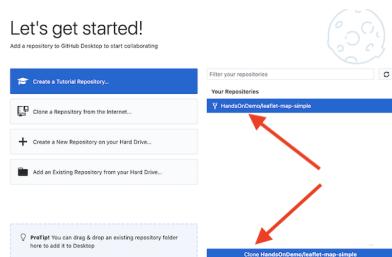


Figure 9.21: Select your *leaflet-map-simple* repo and click the *Clone* button to copy it to your local computer.

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

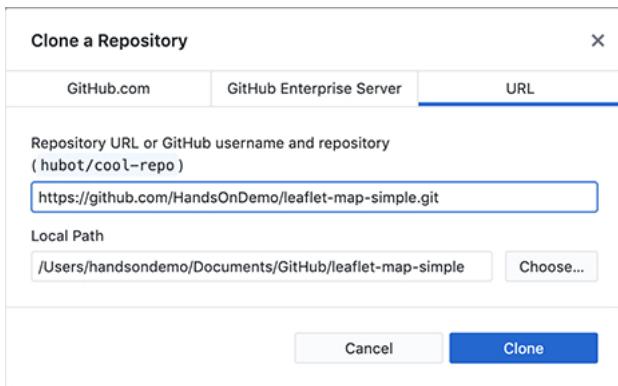


Figure 9.22: Select the Local Path where your repo will be stored on your computer, then click *Clone*.

- 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 9.23.
- 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 9.24. Your screen may look different, depending on whether you use Windows or Mac, and the Local Path you selected to store your files.
- 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 9.25.
- 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 9.26. Open your `index.html` file and edit the title of your map, around line 22, then save your work.

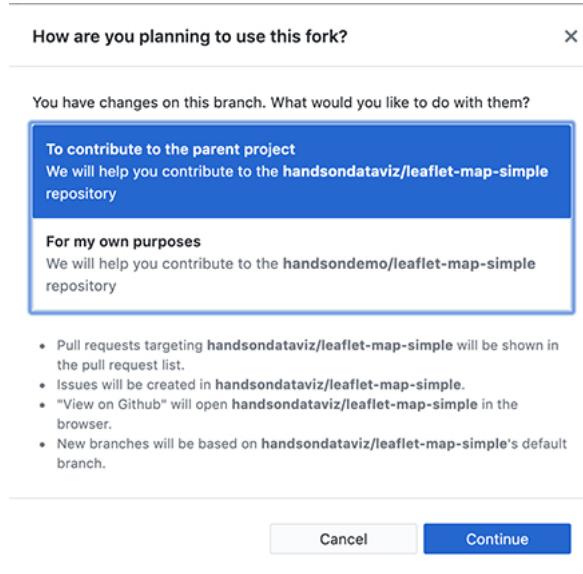


Figure 9.23: If asked how you plan to use this fork, select the default *To contribute to the parent project* and click *Continue*.

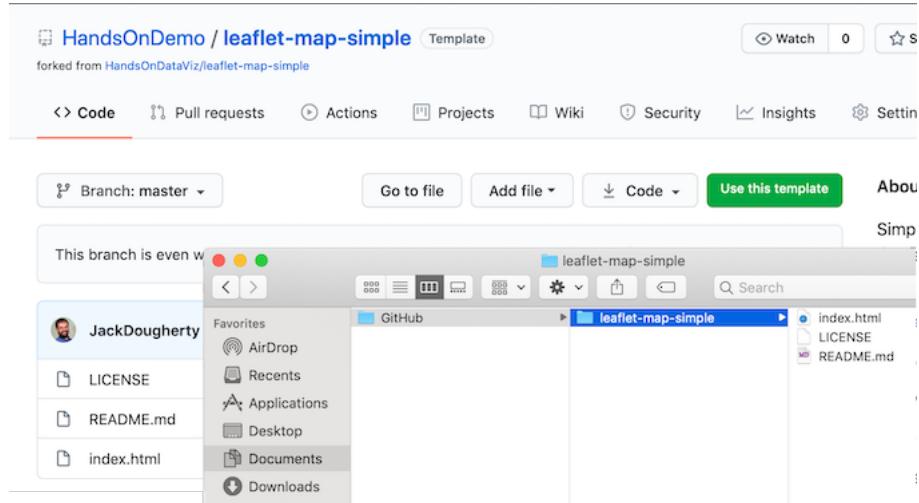


Figure 9.24: Now you have two copies of your repo: in your GitHub online account (on the left) and on your local computer (on the right, as shown in the Mac Finder). Windows screens will look different.

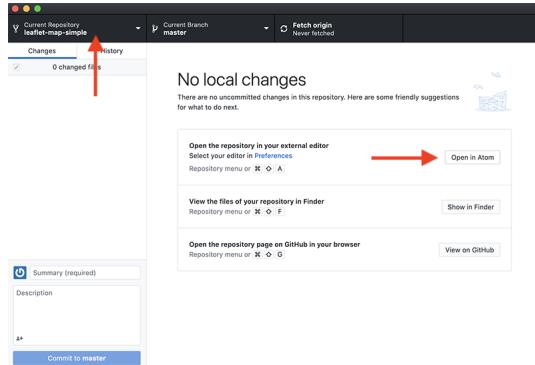


Figure 9.25: In GitHub Desktop, confirm the Current Repo and click the *Open in Atom* button to edit the code.

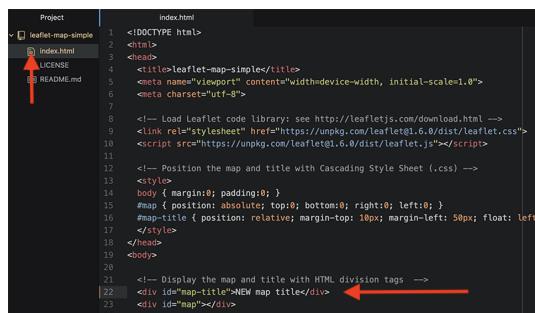


Figure 9.26: Atom Editor opens your repo as a *project*, where you can click files to view code. Edit your map title.

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 9.27. 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.

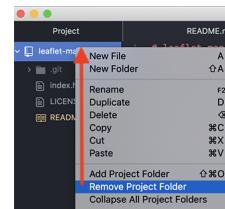


Figure 9.27: To clean up your Atom Editor workspace, right-click to *Remove Project Folder*.

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 9.28.

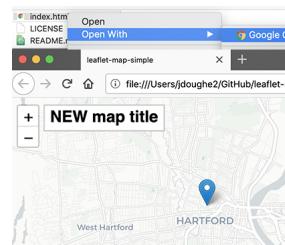


Figure 9.28: Right-click the `index.html` file on your local computer and open with a browser to check your edits.

Note: Since your browser is displaying only the *local computer* version of your code, the web address will begin with `file:///...` rather than `https://...`, as appears in your GitHub Pages online map. Also, if your code depends on online elements, those features may not function when viewing it locally. But for this simple Leaflet map template, your updated map title should appear, allowing you to check its appearance before pushing your edits to the web.

Now let's transfer your edits from your local computer to your GitHub web account, which you previously connected when you set up GitHub Desktop.

11. Go to GitHub Desktop, confirm that your Current Repo is `leaflet-map-simple`, and you will see your code edits summarized on the screen. In this two-step process, first click the blue *Commit to Master* button at the bottom of the page to save your edits to your local copy of your repo. (If you edit multiple files, GitHub Desktop will ask you write a summary of your edit, to help you keep track of your work.) Second, click the blue *Push origin* button to transfer those edits to the parent copy of your repo on your GitHub web account. Both steps are shown in Figure 9.29.

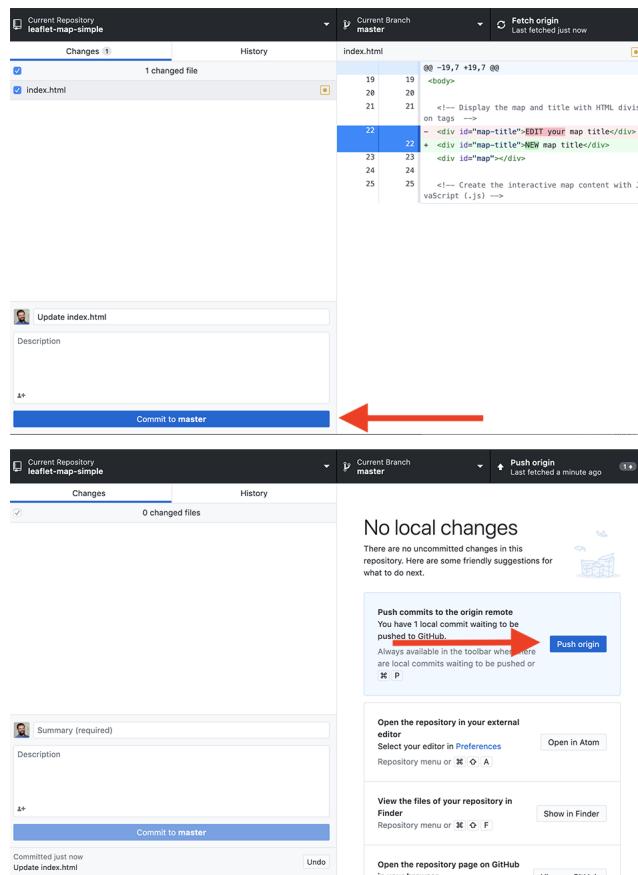


Figure 9.29: In this two-step process, click *Commit to Master*, then click *Push origin* to save and copy your edits from your local computer to your GitHub web account, as shown in this animated GIF.

Congratulations! You've successfully navigated a round-trip journey of code, from your GitHub account to your local computer, and back again to GitHub. Since you previously used the GitHub Pages settings to create an online version of your code, go see if your edited map title now appears

on the public web. The web address you set up earlier follows this format `https://USERNAME.github.io/REPOSITORY`, substituting your GitHub username and repo name.

While you could have made the tiny code edit above in the GitHub web interface, hopefully you've begun to see many advantages of using GitHub Desktop and Atom Editor to edit code and push commits from your local computer. First, you can make more complex code modifications with Atom Editor, which includes search, find-and-replace, and other features to work more efficiently. Second, when you copy the repo to your local computer, you can quickly drag-and-drop multiple files and subfolders for complex visualizations, such as data, geography, and images. Third, depending on the type of code, you may be able to test how it works locally with your browser, before uploading your commits to the public web.

Tip: Atom Editor has many built-in features that recognize and help you edit code, plus the option to install more packages in the Preferences menu. One helpful built-in tool is *Edit > Toggle Comments*, which automatically detects the coding language and converts the selected text from executable code to non-executed code comments. Another built-in tool is *Edit > Lines > Auto Indent*, which automatically cleans up selected text or an entire page of code for easier reading.

GitHub also offers a powerful platform for collaborative projects, such as *Hands-On Data Visualization*. As co-authors, we composed the text of these book chapters and all of the sample code templates on GitHub. Jack started each day by “pulling” the most recent version of the book from our shared GitHub account to his local computer using GitHub Desktop, where he worked on sections and “pushed” his commits (aka edits) back to GitHub. At the same time, Ilya “pulled” the latest version and “pushed” his commits back to GitHub as well. Both of us see the commits that each other made, line-by-line in green and red (showing additions and deletions), by selecting the GitHub repo *Code* tab and clicking on one of our commits, as shown in Figure 9.30.

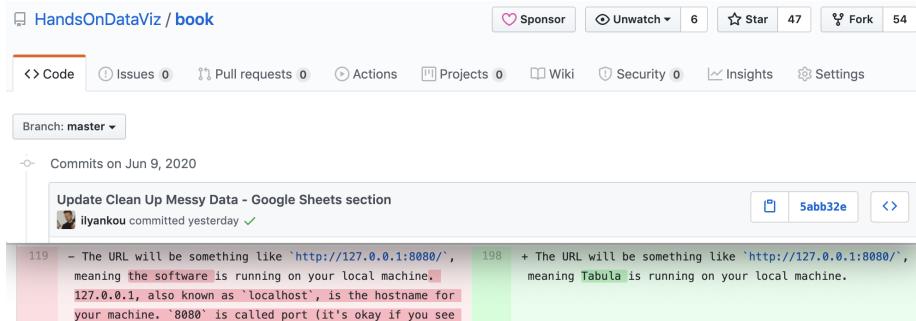


Figure 9.30: Drag-and-drop the file to the upload screen.

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 in order to make edits, and then “merge” the branches back together if desired. Occasionally, if two or more coders attempt to push incompatible commits to the same repo, GitHub will warn about a “Merge Conflict,” and ask you to resolve these conflicts in order to preserve everyone’s work.

Many coders prefer to work on GitHub using its Command Line Interface (CLI), which means memorizing and typing specific commands directly into the Terminal application on Mac or Windows, but this is beyond the scope of this introductory book.

Summary

If this is the first time you’ve forked, edited, and hosted live code on the public web, welcome to the coding family! We hope you agree that GitHub is a powerful platform for engaging in this work and sharing with others. While beginners will appreciate the web interface, you’ll find that the GitHub Desktop and Atom Editor tools makes it much easier to work with Chart.js and Highcharts code templates in Chapter 10 and the Leaflet map code templates in Chapter 11. Let’s build on your brand-new coding skills to create more customized charts and maps in the next two chapters.

Chapter 10

Chart.js and Highcharts Templates

TODO: Rewrite chapter to include Highcharts

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

10.0.0.0.1 Cons

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

Table 10.1: Chart Templates and Tutorials

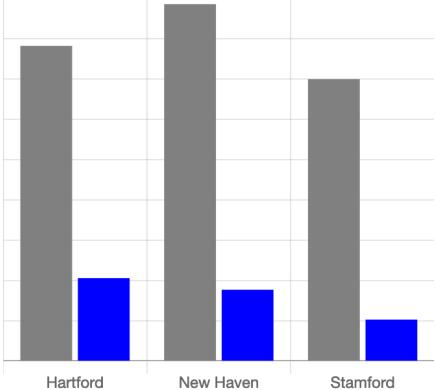
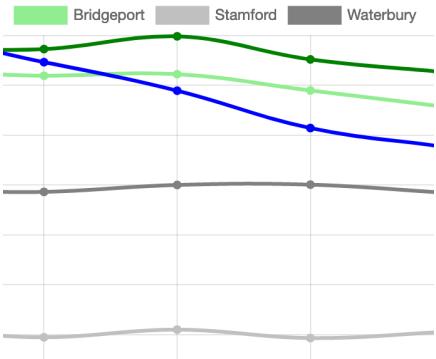
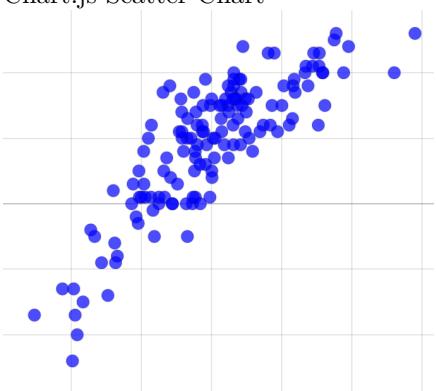
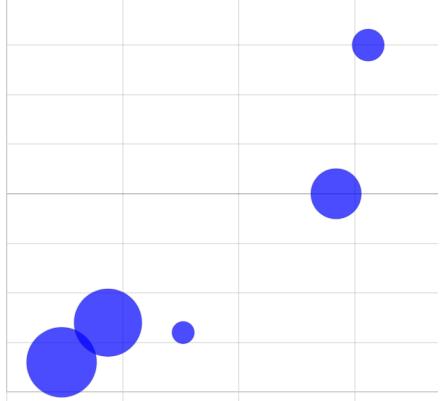
Chart Templates	Best use and tutorials in this book
Chart.js Bar Chart 	Bar charts (vertical bar charts are often called column charts) can be used to compare categorical data. Template with tutorial: Bar Chart.js with CSV Data
Chart.js Line Chart 	Line charts are normally used to show trends (temporal 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

Chart Templates	Best use and tutorials in this book
Chart.js Bubble Chart 	Bubble charts are used to display data of 3 or more dimensions. Template with tutorial: Bubble 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.

TODO: add R code-chunk iframe of src below

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.

TODO: add R code-chunk iframe of src below

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.

TODO: add R code-chunk iframe of src below

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.

TODO: add R code-chunk iframe of src below

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 11

Leaflet Map Templates

In Chapter 7: Map Your Data, we described several drag-and-drop tools designed for beginners, such as Google My Maps and Datawrapper. In this chapter, we offer more advanced map tutorials using our open-source code templates, which you can copy and modify with skills you learned in Chapter 9: Edit and Host Code with GitHub. We built all of the templates in this chapter with Leaflet, a powerful open-source code library for creating interactive maps on desktop or mobile devices.

No coding skills are required to use our first two easy-to-use templates because they pull your map data from a linked Google Sheet. The first template, Leaflet Maps with Google Sheets, is a general-purpose tool that can display points, polygons, or polylines, using your choice of colors, icons, and images, based on data uploaded into your linked Google Sheet and GitHub repository. It also includes the option to display a table of point markers next to your map. The second template, Leaflet Storymaps with Google Sheets, displays your map as a scrolling narrative of chapters to guide readers through a storyline, with the option to display paragraphs of text, images, audio or video clips, and historical map backgrounds loaded into your linked Google Sheet and GitHub repo. If you wish to add some of these extra features, look ahead to Chapter 12: Transform Your Map Data to learn how to locate addresses with our built-in Google Sheets Geocoder, create and edit polygons and polylines with the GeoJson.io tool, edit or join data with polygons using the MapShaper tool, or georectify a scanned map to use as a background overlay with the MapWarper tool.

Our remaining Leaflet templates are designed to help users develop their map coding skills. Even if you have no prior coding experience, but can follow instructions and are *code-curious* about how things work on your computer, start with the Leaflet Maps with CSV Data tutorial, which walks you through the steps of creating a point map that pulls data from a CSV file, a generic spreadsheet format we discussed in Chapter 2. Then move on to the Leaflet Maps with

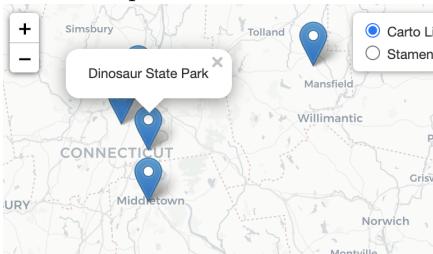
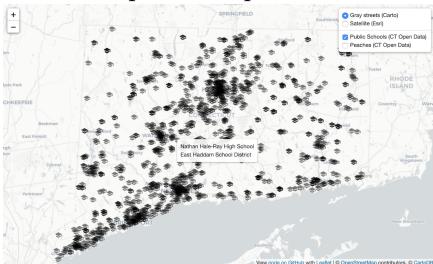
Open Data API tutorial, to learn how to code using an application program interface to pull information directly from open data repositories as we described in Chapter 3. In both of these templates, you’ll learn how Leaflet maps are written using three coding languages:

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

Explore our Leaflet map templates and you’ll also see how they refer to different code components, such as basemap tiles from various open-access online providers, such as Carto, Esri, Stamen, and Open Street Map, that allow you to zoom into background maps. You’ll also see data files to place information about points, polygons, or polylines on top of the map, usually in CSV or GeoJSON format—see chapter 12, with names similar to `data.csv` or `map.geojson`. If you’re new to coding, creating Leaflet maps can be a great place to start and quickly see the results of what you’ve learned. To help you solve problems that may arise, see Fix Common Mistakes in the appendix.

Table 11.1: Map Templates and Tutorials

Map Templates	Best use and tutorials in this book
Leaflet Maps with Google Sheets 	Best to show interactive points, polygons, or polylines, using your choice of colors, styles, and icons, based on data loaded into your linked Google Sheet (or CSV file) and GitHub repository. Includes option to display a table of point map markers next to your map. Template with tutorial: Leaflet Maps with Google Sheets
Leaflet Storymaps with Google Sheets 	Best to display your map as a scrolling narrative of chapters to guide readers through a storyline, with the option to include paragraphs of text, images, audio or video clips, and historical map backgrounds loaded into your linked Google Sheet (or CSV) and GitHub repo. Template with tutorial: Leaflet Storymaps with Google Sheets

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

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

Leaflet Maps with Google Sheets

Sometimes you need to create a map that cannot be made easily with drag-and-drop tools, because you need to customize its appearance or add new layers of point, polygon, or polyline data. In these cases, consider making a copy of our Leaflet Maps with Google Sheets template on GitHub. It gives you more control over choosing colors, icons, and images, and also the option to display a data table of point markers. To customize your interactive map, you enter data into a Google Sheet template, which you link directly to your copy of the Leaflet code repository, as shown in Figure 11.1 and Figure 11.2.

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

Tutorial Outline

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

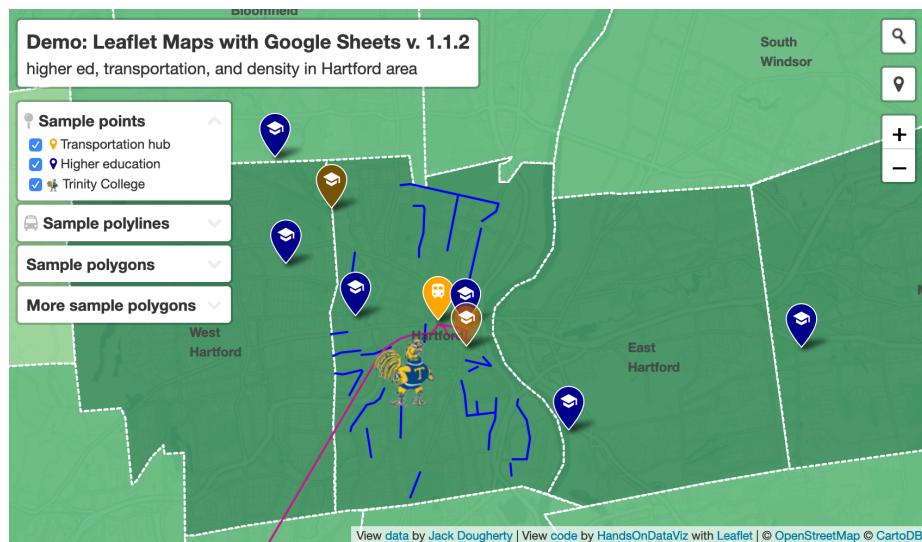


Figure 11.1: Explore a live demonstration of Leaflet Maps with Google Sheets.

Before you begin, you must:

- Have a Google Drive account.
- Know how to File > Make a Copy in Google Sheets, as described in Chapter 2.
- Have a GitHub account.
- Know how to Edit and Host Code with GitHub, as described in Chapter 9.

In the first part of the tutorial, you will create and publish your copies of our GitHub and Google Sheets templates:

- A) Copy the GitHub template and publish your version with GitHub Pages
- B) File > Make a Copy of Google Sheet template, Share, and Publish
- C) Paste your Google Sheet browser address in two places in your GitHub repo
- D) Update your Google Sheet *Options* tab info and refresh your live map

In the second half, you will learn how to upload and display different types of map data, such as points, polygons, and polylines, and to edit colors, icons, and images, based on information you enter into the linked Google Sheet and upload to your GitHub repo.

	A	B
1	Setting	Customize
2	Map Info	
3	Map Title	Demo: Leaflet Maps with Google Sheets v. 1.1.2
4	Map Subtitle	higher ed, transportation, and density in Hartford area
5	Display Title	topleft
6	Author Name	Jack Dougherty
7	Author Email or Website	jack.dougherty@trincoll.edu
8	Author Code Credit	HandsOnDataViz
9	Author Code Repo	https://github.com/JackDougherty/leaflet-maps-with-google-sheets
10	Map Settings	
11	Basemap Tiles	CartoDB.PositronNoLabels
12	Cluster Markers	off
13	Intro Popup Text	
14	Initial Zoom	
15	Initial Center Latitude	
16	Initial Center Longitude	
17	Map Controls	
18	Search Button	topright
19	Mapzen Search Key	search-jBPBt5y
20	Search Radius	5 miles
21	Search Results Zoom Level	12

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

Figure 11.2: Explore the live Google Sheet template that feeds data into the Leaflet map above.

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

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

- G) Save each Google Sheets tab as a CSV file and upload to GitHub
- OR
- H) Get your own Google Sheets API Key to insert into the code

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

A) Copy the GitHub template and publish your version with GitHub Pages

1. Right-click to open this GitHub code template in a new tab: <https://github.com/handsondataviz/leaflet-maps-with-google-sheets>
2. In the upper-right corner of the code template, sign in to your free GitHub account.
3. In the upper-right corner, click the green *Use this template* button to make a copy of the repository in your GitHub account. On the next screen, name your repo `leaflet-maps-with-google-sheets` or choose a different meaningful name in all lower-case. Click the *Create repository from template* button. Your copy of the repo will follow this format:

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

4. In your new copy of the code repo, click the upper-right *Settings* button and scroll way down to the GitHub Pages area. In the drop-down menu, change *Source* from *None* to *Master*, keep the default `/(root)` setting, and press *Save* as shown in Figure 11.3. This step tells GitHub to publish a live version of your map on the public web, where anyone can access it in their browser, if they have the web address.

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

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

GitHub Pages

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

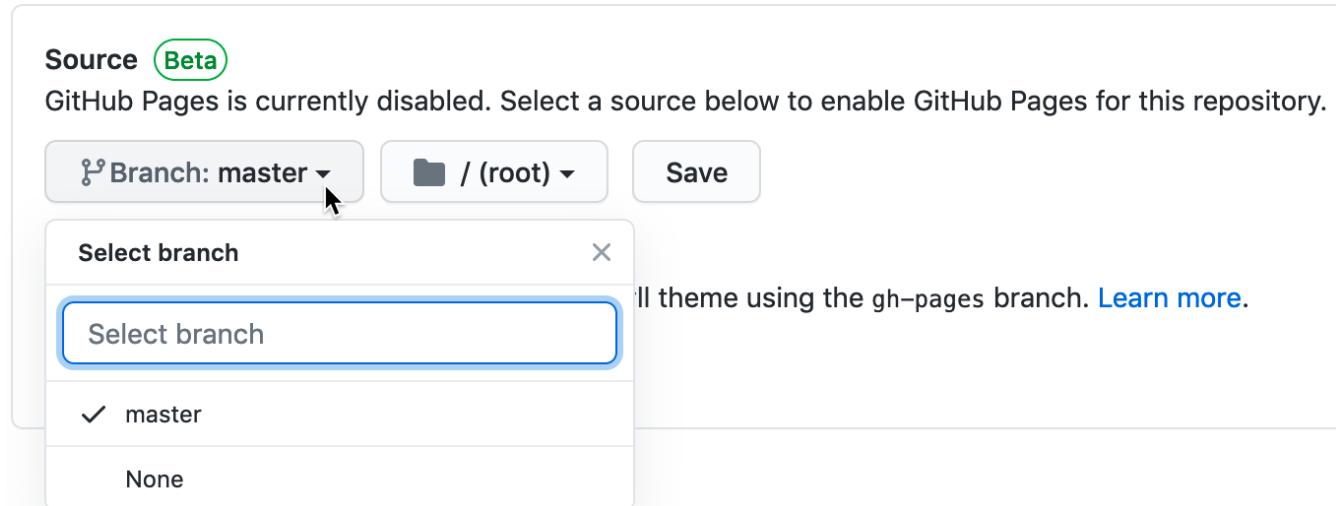


Figure 11.3: In *Settings*, go to *GitHub Pages*, and switch the source from *None* to *Master*.

<https://USERNAME.github.io/leaflet-maps-with-google-sheets>

6. Scroll up to the top, and click on your repo name to go back to its main page.
7. At the top level of your repo main page, click on README.md, and click the pencil icon to edit this file.
8. Delete the link to the *our* live site, and paste in the link to *your* published site. Scroll down and *Commit* to save your edits.

TODO: Insert image here

9. On your repo main page, right-click the link to open your live map in a new tab. *Be patient* during busy periods on GitHub, when your website may take up to 1 minute to appear for the first time.

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

1. Open this Google Sheets template in a new tab
2. Sign into your Google account, and select *File > Make a Copy* to save your own version of this Google Sheet on your Google Drive
3. Click the blue Share button, and click *Change to anyone with the link*, then click *Done*. This publicly shares your map data, which is required to make this template work.
4. Go to *File > Publish to the Web*, and click the green *Publish* button to publish the entire document, so that the Leaflet code can read it. Then click the upper-right *X* symbol to close this window.
5. At the top of your browser, copy your Google Sheet address or URL (which usually ends in ...XYZ/edit#gid=0). Do *NOT* copy the *Published to the web* address (which usually ends in ...XYZ/pubhtml), as shown in Figure 11.4.

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

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

1. At the top of your GitHub repo, click to open the file named google-doc-url.js, and click the pencil symbol to edit it.

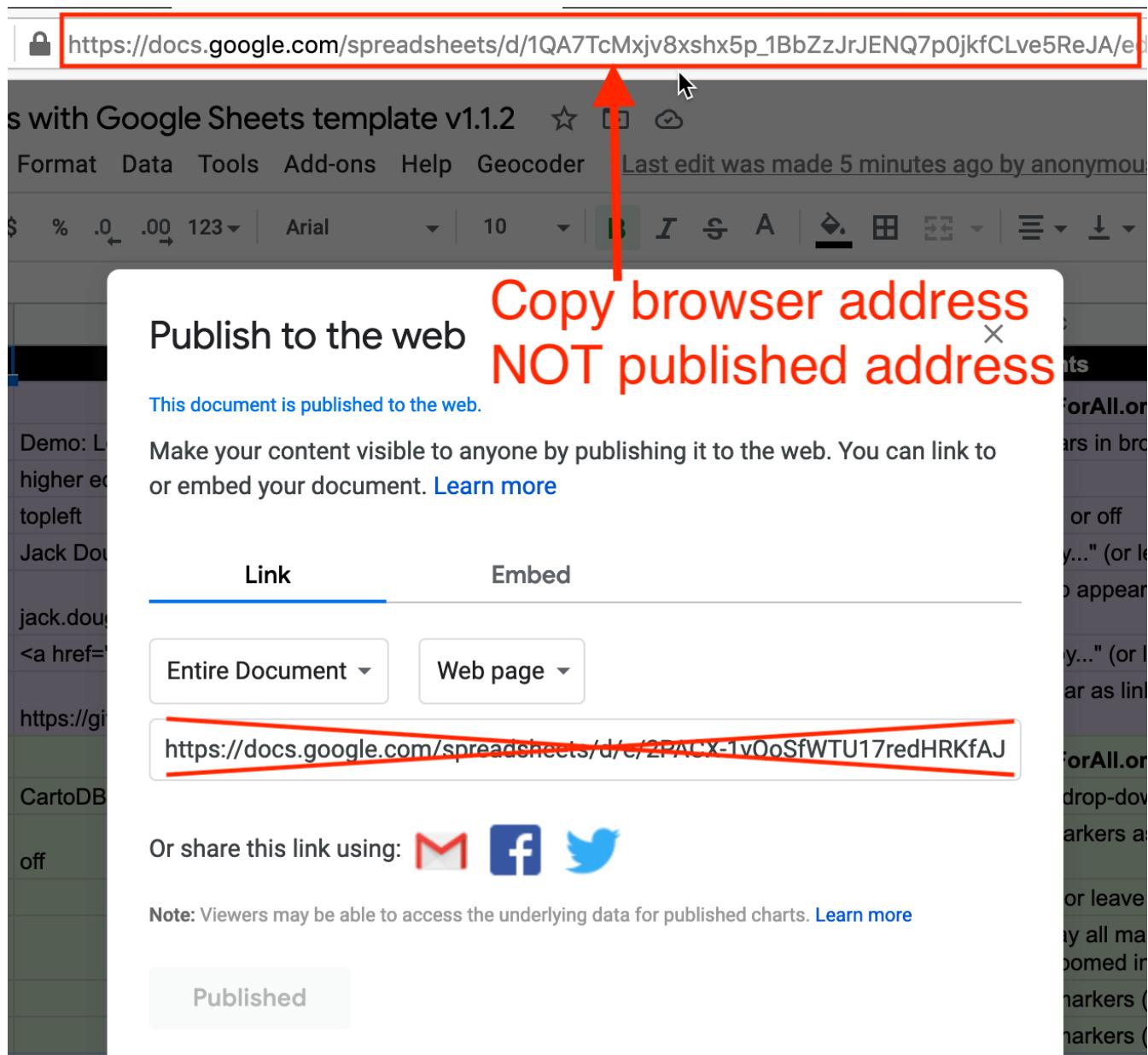


Figure 11.4: Copy the Google Sheet address at the top of the browser, NOT the *Publish to the web* address.

2. Paste *your* Google Sheet address or URL (which usually ends in `...XYZ/edit#gid=0`) to replace the existing URL, as shown in Figure 11.5. Be careful *NOT* to erase the single quotation marks or the semicolon at the end. See separate instructions about the Google API key further below.



```

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

```

Figure 11.5: Paste in *your* Google Sheet URL to replace *our* URL.

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

TODO: Insert image here

D) Update your Google Sheet *Options* tab info and refresh your live map

Now that your published Google Sheet is linked to your live map, go to the *Options* tab and update any of these items:

- Map Title
- Map Subtitle
- Author Name
- Author Email or Website
- Author Code Repo

Open the browser tab that displays your live map and refresh the page to see your changes. If your changes do not appear within a few seconds, see the Fix Common Problems section of the appendix.

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

In the *Points* tab of your Google Sheet, you'll see column headers to organize and display interactive markers on your map. Replace the demonstration data with your own, but do *not* delete or rename the column headers, since the Leaflet code looks for these specific names.

- Group: Create any labels to categorize groups of markers in your legend.
- Marker Icon: Search Font Awesome Icons and insert any standard icon name such as `school` or `bus`, or leave blank for no icon inside the marker. To create your own custom icon, see further below.
- Marker Color: Search W3Schools Color Names and insert any standard name such as `blue` or `darkblue`. Or insert a web color code such as `#775307` or `rgba(200,100,0,0.5)`.
- Icon Color: Set the color of the icon inside the marker. The default is `white`, which looks good inside darker-colored markers.
- Custom Size: Leave blank, unless you are creating your own custom icon further below.

The next set of columns include items that appear when users click on point markers:

- Name: Add a title to display in the marker pop-up window.
- Image: Insert link to an external image link, or upload a small image to the *media* folder in your GitHub repo and add its pathname, such as `media/trinity-college.jpg`. TODO: add this to GSheet after code update
- Description: Add text to appear in the marker pop-up window. You may include HTML web links in this format: `link text`. If a map link should open in a new browser tab, set the *target* attribute to `_blank`. Learn about HTML syntax at W3Schools.

The next six columns help to place the Location of your marker on the map. The Leaflet code requires you to fill-in the Latitude and Longitude columns, which you can do using the built-in Geocoder menu in our Google Sheet template, which we introduced in chapter TODO: INSERT GEOCODER CHAPTER. Click the cell(s) you wish to geocode, and use shift-click to select all six columns from Location to Source (columns I to N). In the Geocoder menu, select either US Census or Google, as shown in Figure 11.6. The first time you run the Geocoder script, it will ask for your permission. TODO: since we still need Google to authorize our script, you will need to click *Advanced* and then *Go to Geocoder (unsafe)* near the bottom of the next screen. Trust us – it's safe to use! Sorry for the inconvenience. Always inspect the accuracy of your geocoded results.

Optional: You can display a table of viewable markers at the bottom of your map, as shown in Figure 11.7. In the *Options* tab, set *Display Table* (cell B30) to *On*. You can also adjust the *Table Height*, and modify the display of *Table Columns* by entering the column headers, separated with commas.

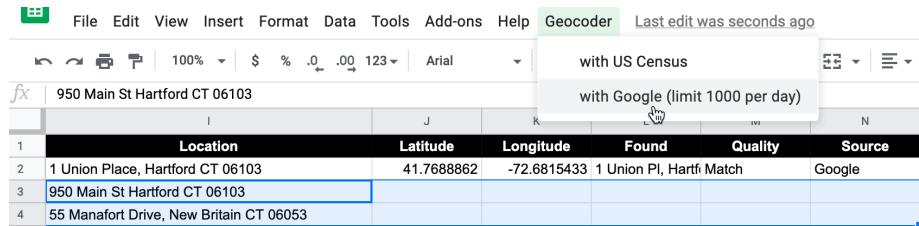


Figure 11.6: To use our built-in Geocoder menu, shift-click to select all columns from Location through Source.

Name	Location	Description
Capital Community College	950 Main St Hartford CT 06103	Website: http://www.ccc.commne...
Hartford Seminary	77 Sherman St, Hartford CT 06105	Website: http://www.hartsem.edu...

Figure 11.7: One option is to display a table of viewable markers at the bottom of your map.

Optional: You can create your own custom marker, such as the Trinity College Bantam mascot in the demo. Use an image editing tool to create a square image, 64 x 64 pixels or smaller, with a transparent background. Save it in PNG format with a filename using all lower-case letters with no spaces. Upload the image to the *markers* folder in your GitHub repo. In the Marker Icon column, set the pathname in this format: `markers/custom-trinity-64.png`. In the Custom Size column, set to `64x64` or similar.

Open the browser tab that displays your live map and refresh the page to see your changes. If your changes do not appear within a few seconds, see the Fix Common Problems section of the appendix.

F) Remove or display point, polygon, or polylines data and legends

By default, the demo map displays three types of data—points, polygons, and polylines—and their legends. You can remove any of these from your map by modifying your linked Google Sheet:

To remove points:

- In the *Options* tab, set *Point Legend Position* (cell B27) to *Off* to hide it.
- In the *Points* tab, delete all rows of point data.

To remove polylines:

- In the *Options* tab, set *Polyline Legend Position* (cell B36) to *Off* to hide it.
- In the *Polylines* tab, delete all rows of polyline data.

To remove polygons:

- In the *Polygons* tab, set *Polygon Legend Position* (cell B4) to *Off* to hide it.
- Also in the *Polygons* tab, set *Polygon GeoJSON URL* (cell B6) to remove that data from your map.
- In the next tab *Polygons1*, use the tab drop-down menu to select *Delete* to remove the entire sheet.

You've already learned how to add more markers in the *Points* tab as described above. But if you wish to add new polygon or polyline data, you'll need to prepare those files in GeoJSON format using either the GeoJson.io tool tutorial or the MapShaper tool tutorial in Chapter 12.

After you've prepared your GeoJSON data, name the files using all lower-case characters and no spaces, and upload them into the *geometry* subfolder of your GitHub repo. Then update these settings in your linked Google Sheet:

To display polylines:

- In the *Options* tab, make sure *Polyline Legend Position* (cell B36) is visible by selecting *topleft* or a similar position.
- In the *Polylines* tab, enter the GeoJSON URL pathname to the file you uploaded to your GitHub repo, such as `geometry/polylines-bike-lanes.geojson`. Then insert a Display Name, Description, and Color.

To display polygons:

- In the *Polygons* tab, make sure *Polygon Legend Position* (cell B4) is visible by selecting *topleft* or a similar position.
- Also, in *Polygon GeoJSON URL* (cell B6) enter the pathname to the file you uploaded to your GitHub repo, such as `geometry/polylines-town-population.geojson`.
- Also, you can change the *Polygon Legend Title* (cell B3) and add an optional *Polygon Legend Icon* (cell B5).
- Also, edit the *Polygon Data* and *Color Settings* sections to modify the labels and ranges to align with the properties of your GeoJSON file. In the *Property Range Color Palette*, you can automatically select a color scheme from the ColorBrewer tool we described in the Map Design section of Chapter 7, or manually insert colors of your choice in the cell below.
- Read the *Hints* column in the *Polygons* sheet for tips on how to enter data.
- If you wish to display multiple polygon layers, use the *Polygons* tab drop-down menu to *Duplicate* the sheet, and name additional sheets in this format: *Polygons1*, *Polygons2*, etc.

Finalize Your Map

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

G) Save each Google Sheets tab as a CSV file and upload to GitHub

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

To move your map data from Google Sheets to CSV format, go to each tab and select *File > Download As* into CSV format, as shown in Figure 11.8, using these file names: **TODO: should we warn to keep the first letter upper-case?**

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

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

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

If you wish to keep using your Google Sheets to store your data, go to the chapter section named Get Your Own Google Sheets API Key, and insert it into the Leaflet map code as described, to avoid overusing our key. Google Sheets requires an API key to maintain reasonable usage limits on its service. You can get a free Google Sheets API key if you have a personal Google account, but *not* a Google Suite account provided by your school or business. TODO: confirm this detail

Warning: We reserve the right to change *our* Google Sheets API key at any time, especially if other people overuse or abuse it. This means that you *must* finalize your map using either step J or K above, since it will stop working if we change our key.

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

TODO: Start again here

Leaflet Storymaps with Google Sheets

TODO: Add intro text

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

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

Features

- Show map points, text, images, and links with scrolling narrative
- Free and open-source code template, built on Leaflet and linked to Google Sheets

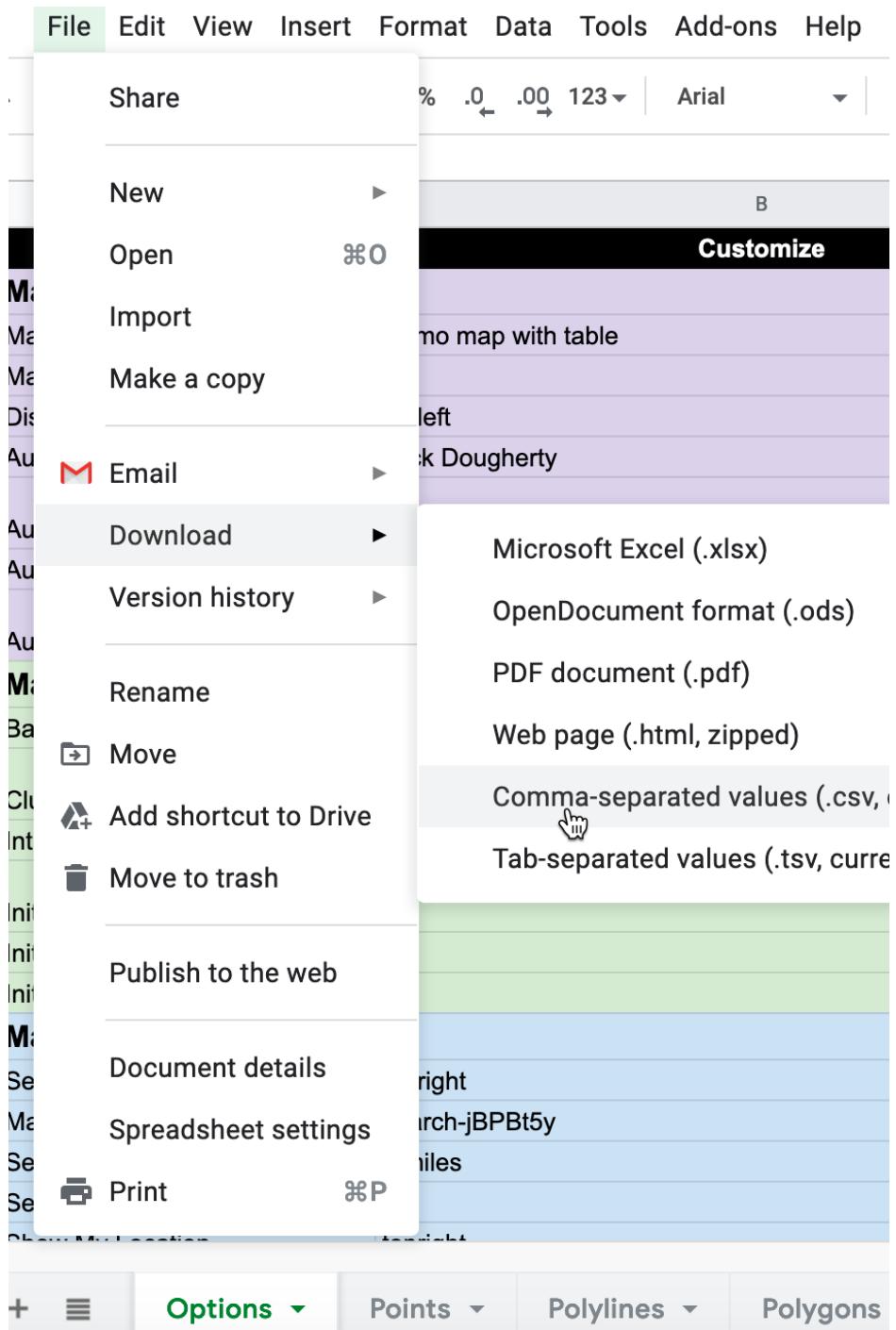


Figure 11.8: One way to finalize your map is to download each Google Sheets tab as a CSV file.

- 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
- C2) NEW: Create a free Google Sheets API key to paste into the code
- D) Modify your map settings in the Options tab and test your live map
- E) Geocode locations in the Points tab

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

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

Before you begin, this tutorial assumes that you:

- have a free Google Drive account, and learned the File > Make a Copy in Google Sheets tutorial in this book
 - have a free GitHub account, and understand concepts from the Edit and Host Code with GitHub chapter in this book
- 1) Right-click to open this GitHub code template in a new tab: <https://github.com/handsondataviz/leaflet-storymaps-with-google-sheets>
 - 2) In the upper-right corner of the code template, sign in to your free GitHub account
 - 3) In the upper-right corner, click Fork to copy the template (also called a code repository, or repo) into your own account. The web address (URL) of the new copy in your account will follow this format:

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

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

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

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

- 6) Scroll up to the top, and click on your repo name to go back to its main page.
- 7) At the top level of your repo main page, click on README.md, and click the pencil icon to edit this file, written in easy-to-read Markdown code.
- 8) Delete the link to the current live site, and paste in the link to YOUR site. Scroll down and Commit to save your edits.
- 9) On your repo main page, right-click the link to your live map to open in a new tab. **Be patient** during busy periods on GitHub, when your website may take up to 1 minute to appear the first time.

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

- 1) Right-click to open this Google Sheets template in a new tab: https://docs.google.com/spreadsheets/d/1AO6XHL_0JafWZF4KEejkdDNqfuZWUk3SINIQ6MjlRFM/
- 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).

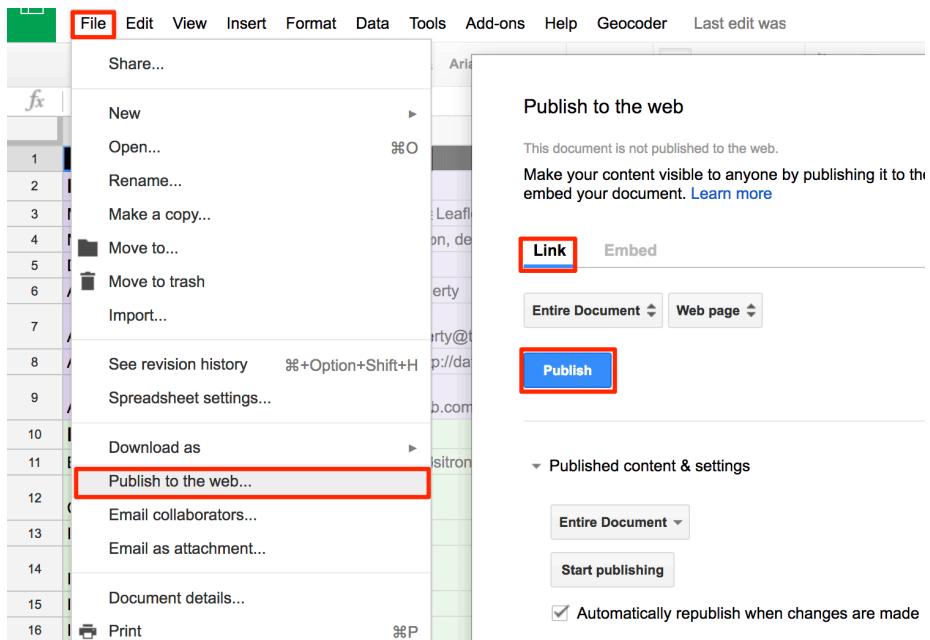


Figure 11.9: Screenshot: File > Publish the link to your Google Sheet

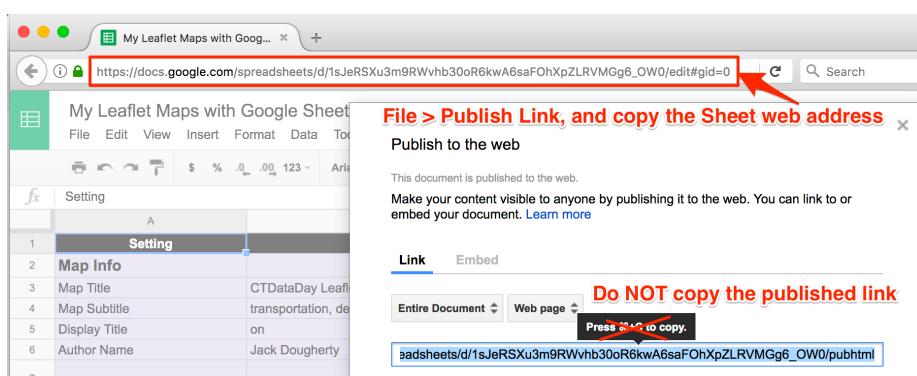


Figure 11.10: Screenshot: Copy the Google Sheet URL, not the Publish URL

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

Add links to your text in the Google Sheet

Add line breaks to your text in the Google Sheet

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

Markers

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

Overlay GeoJSONs

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

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

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

Data in local CSV files

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

Modify your Style Sheet

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

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

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

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

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

Get Your Google Sheets API Key

After you've created your own version of Leaflet Maps with Google Sheets or Leaflet Storymaps with Google Sheets, there are two ways to finalize your map, as described above: either save your Google Sheet tabs in CSV format, or get your own Google Sheets API key and paste it into your Leaflet code on GitHub. You'll learn about the latter method in this section.

Google requires a API (application program interface) key to allow your computer code to read data from your Google Sheets, beginning with version 4 in September 2020. (If you created your own Leaflet Maps or Storymaps with Google Sheets using our template prior to September 2020, and you want to continue to pull data from your Google Sheet, you'll need to update your code to make sure it keeps working **TODO: explain pull request.**) Google requires this API key to maintain reasonable limits on use of its services. For Google

Sheets, the limit is 500 requests per 100 seconds per project, and 100 requests per 100 seconds per user. There is no daily usage limit.

You can get your own API key for free by following the steps below. Overall, you will create and name your Google Cloud project, enable the Google Sheets API to allow a computer to read data from your Google Sheet, copy your new API key, and paste it into the Leaflet code in place of our key.

Before you begin:

- You need a personal Google account, *not* a Google Suite account issued by your school or business.
 - This tutorial presumes that you have already have completed the Leaflet Maps with Google Sheets or Leaflet Storymaps with Google Sheets template above, and wish to finalize your map.
 - If you already created a Google Sheets API key for one template above, you can also use that key for another template.
1. Go to the Google Developers Console at <https://console.developers.google.com/> and log in to your Google account. Google may ask you to identify your country and agree to its terms of service.
 2. Click on *Create a Project* on the opening screen, as shown in Figure 11.11. Or alternatively, go to the upper-left drop-down menu to *Select a project > New project*.

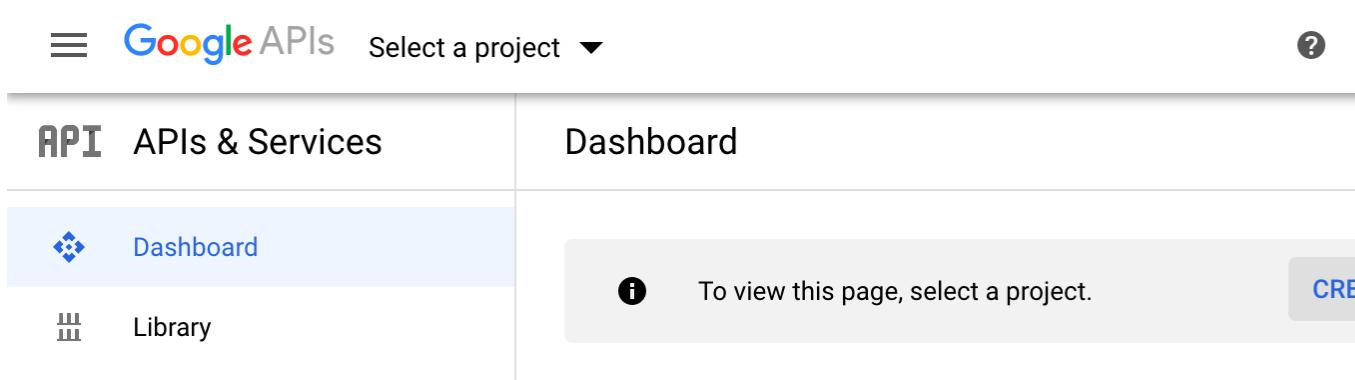


Figure 11.11: Select *Create a Project* or use the menu to select a new project.

3. In the next screen, give your new project a meaningful short name to remind you of its purpose, such as `handsondataviz`. You do not need to create an organization or parent folder. Then click *Create*, as shown in Figure 11.12.

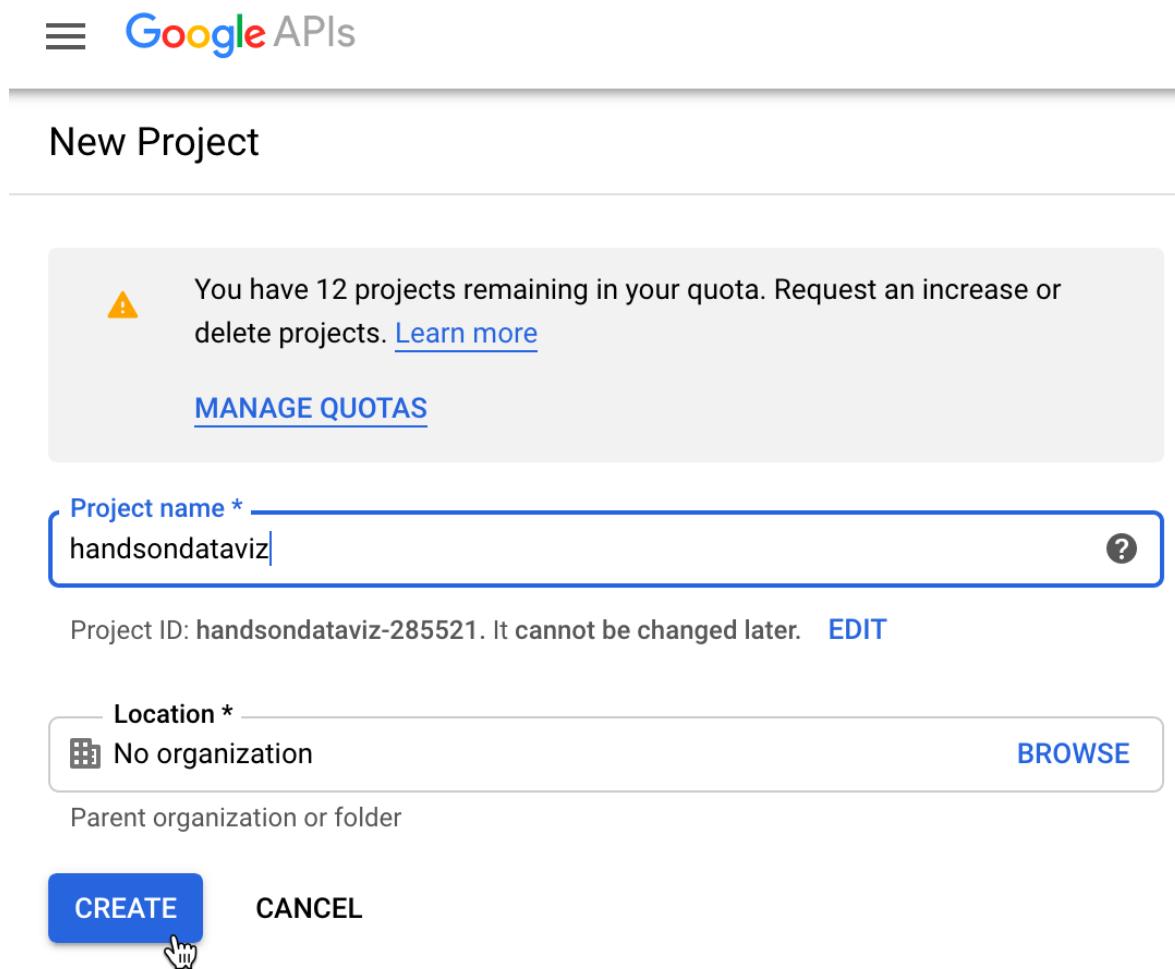


Figure 11.12: Give your project a meaningful short name.

4. In the next screen, press the *+ Enable APIs and Services* at the top of the menu, as shown in Figure 11.13. Make sure that your new project name appears near the top.

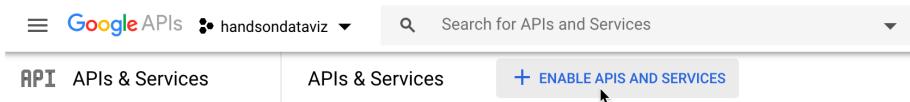


Figure 11.13: Press the *+ Enable APIs and Services* button.

5. In the next screen, enter *Google Sheets* into the search bar, and select this result, as shown in Figure 11.14.

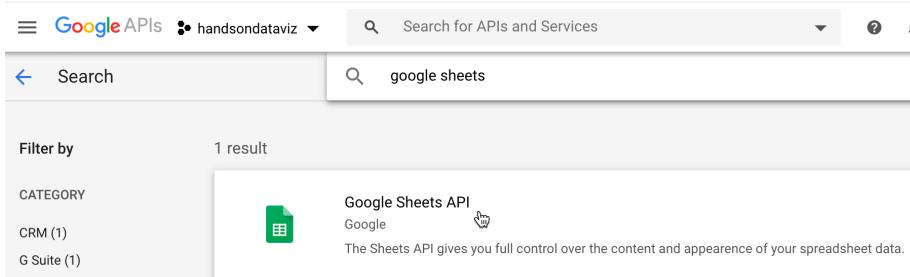


Figure 11.14: Search for *Google Sheets* and select this result.

6. In the next screen, select the *Enable* button to turn on the Google Sheets API for your project, as shown in Figure 11.15.
7. In the left sidebar menu, click *Credentials*, then click *+ Create Credentials* and select *API key*, as shown in Figure 11.16.
8. In the next screen, the console will generate your API key. Copy it, then press *Restrict key*, as shown in Figure 11.17.
9. In the new window, under *API restrictions*, choose the *Restrict key* radio button. In the dropdown that appears, choose *Google Sheets API*, then click *Save*, as shown in Figure 11.18.

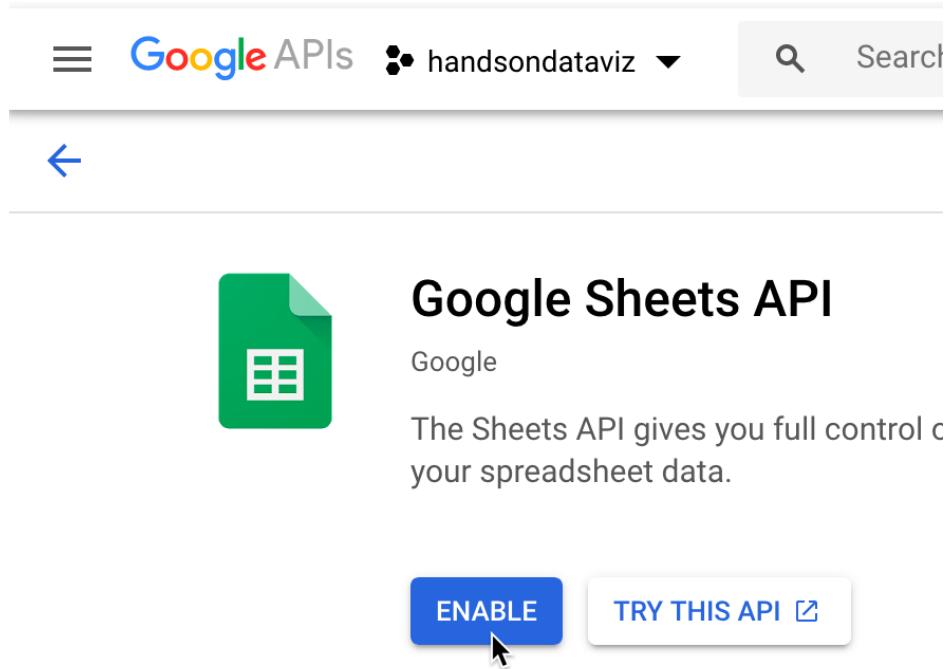


Figure 11.15: Select the *Enable* button for Google Sheets API.

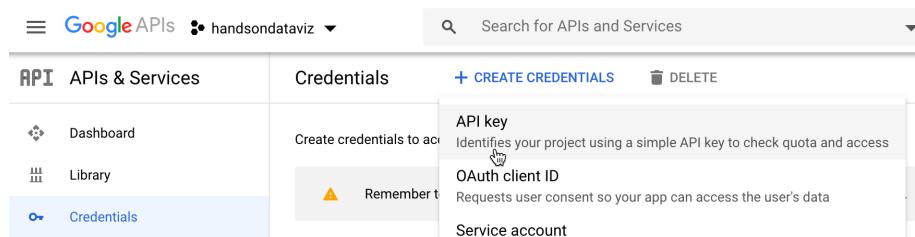


Figure 11.16: Select *Credentials - Create Credentials - API key*.

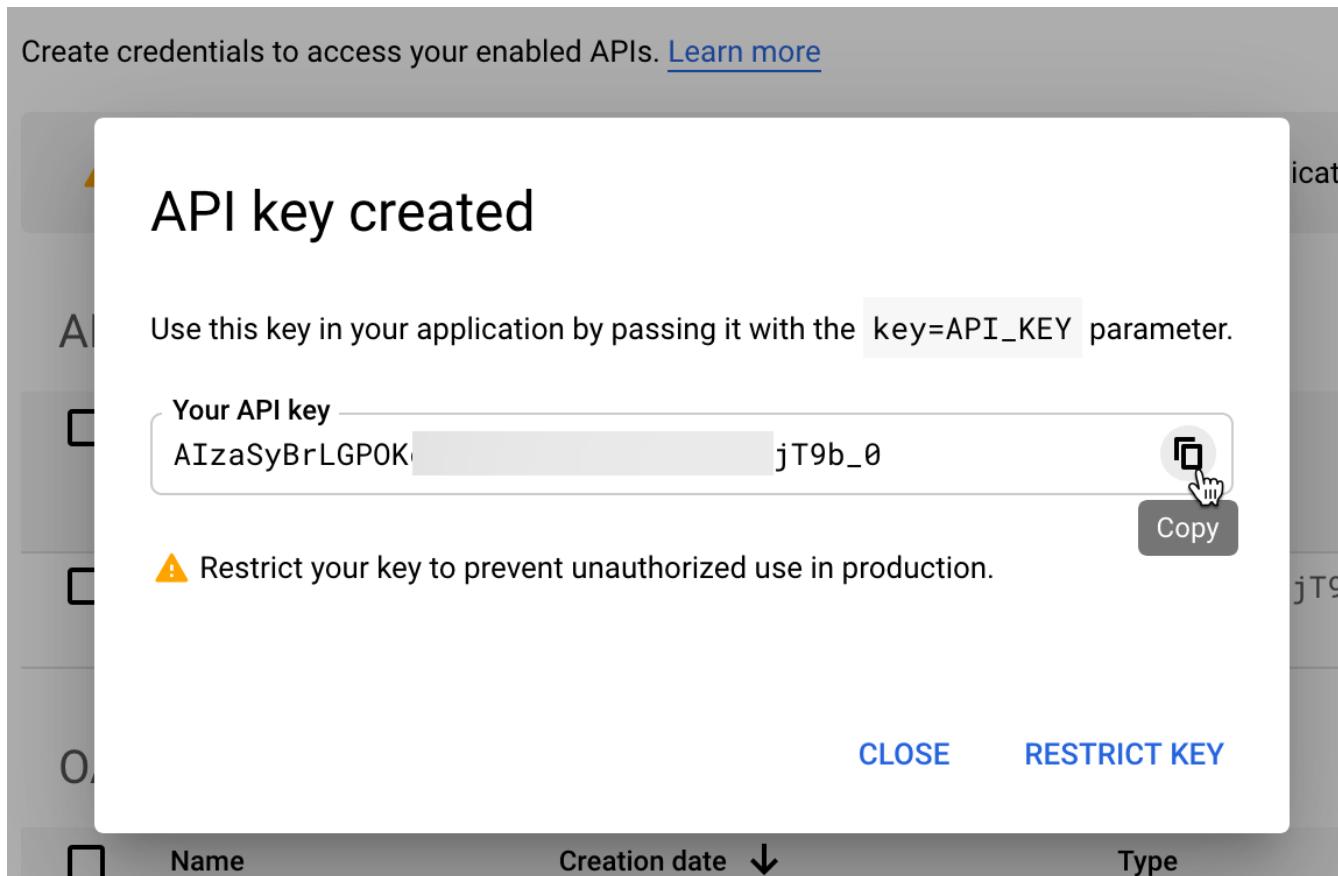


Figure 11.17: Copy your API key and press *Restrict key*.

The screenshot shows the Google APIs console interface. On the left, a sidebar lists various API services: Dashboard, Library, Credentials (which is selected and highlighted in blue), OAuth consent screen, Domain verification, and Page usage agreements. The main content area has a header "Restrict and rename API key" with a back arrow and a "REGENERATE" button. Below this, there are two sections: "Key restrictions" and "Application restrictions". In "Key restrictions", a warning message states: "This key is unrestricted. Restrictions help prevent unauthorized use and quota theft." In "Application restrictions", it says: "An application restriction controls which websites, IP addresses, or applications can use your API key. You can set one application restriction per key." A radio button is selected for "None". In the "API restrictions" section, there is a note: "API restrictions specify the enabled APIs that this key can call". Two radio button options are shown: "Don't restrict key" (unchecked) and "Restrict key" (checked). A dropdown menu titled "Type to filter" is open, showing "Google Sheets API" with a checked checkbox and a cursor hovering over the close button. At the bottom, a note says: "Note: It may take up to 5 minutes for settings to take effect".

Figure 11.18: Choose *API restrictions* - *Restrict key* - *Google Sheets API*

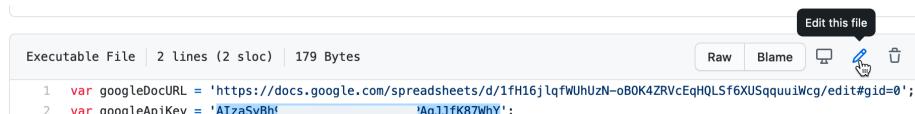


Figure 11.19: Paste in *your* Google Sheets API key to replace *our* key.

10. In your Leaflet map code on your GitHub repo, open the `google-doc-url.js` file, click the pencil symbol to edit it, and paste in *your* Google Sheets API key to replace *our* key, as shown in Figure 11.19. Be careful not to erase the single-quote marks or the semicolon. Scroll down to *Commit* your changes.

You might receive a notification from GitHub stating that you have an exposed API key, but don't worry. This key can only be used with Google Sheets, you received it for free, and you did not attach any billing information to it, so Google cannot charge you for its use.

Now that you've learned how to create a Google Sheets API key to use with Leaflet Maps with Google Sheets or Leaflet Storymaps with Google Sheets, in the next sections you'll learn more about other types of Leaflet map templates.

Leaflet Maps with CSV Data

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

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

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

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

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

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

Video

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

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

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

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

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

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

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

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

- 6) Scroll up to the top, and click on your repo name to go back to its main page.
- 7) At the top level of your repo main page, click on README.md, and click the pencil icon to edit this file, written in easy-to-read Markdown code.
- 8) Delete the link to the current live site, and paste in the link to your site. Scroll down and Commit to save your edits.
- 9) On your repo main page, right-click on the link to your published site to open in a new tab. **Be patient** during busy periods, because your website may take up to 1 minute to appear the first time.

C) Modify your map title and add layer controls

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

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

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

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

```
// Set up initial map center and zoom level
var map = L.map('map', {
  center: [41.77, -72.69], // EDIT latitude, longitude to re-center map
  zoom: 12, // EDIT from 1 to 18 -- decrease to zoom out, increase to zoom in
  scrollWheelZoom: false
});
```

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

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

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

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

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

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

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

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

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

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

```
/* Display a blue point marker with pop-up text */
L.marker([41.77, -72.69]).addTo(map) // EDIT latitude, longitude to re-position marker
.bindPopup("Insert pop-up text here"); // EDIT pop-up text message
```

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

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

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

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

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

Leaflet Maps with Open Data API

TODO:

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

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

Why pair Leaflet maps with Socrata data?

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

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

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

About Socrata API endpoints

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



Figure 11.20: Police Incidents dataset on Hartford Open Data portal

Copy the API endpoint. The default version is JSON.

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

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

JSON Raw Data Headers

Save Copy Collapse All Filter JSON

```

  ↳ 8: {…}
  ↳ 9: {…}
  ↳ 10: {…}
  ↳ 11:
    case_number: "5000007"
    date: "2005-01-01T00:00:00.000"
    time_24hr: "0000"
    address: "CHURCH ST & TRUMBULL ST"
    ucr_1_category: "32* - PROPERTY DAMAGE ACCIDENT"
    ucr_1_description: "PROP DAM ACC"
    ucr_1_code: "3224"
    ucr_2_category: "23* - DRIVING LAWS"
    ucr_2_description: "FOLL TOO CLOSE"
    ucr_2_code: "2334"
    neighborhood: "DOWNTOWN"
    ↳ geom: {…}
    :@computed_region_ugzy_yqsh: "19"
    :@computed_region_35zh_8fiz: "10"
    :@computed_region_2vdc_22if: "15050"
    :@computed_region_haf6_6xye: "1041"
  ↳ 12: {…}
  ↳ 13: {…}
  ↳ 14: {…}
  ↳ 15: {…}
  ↳ 16: {…}

```

Figure 11.21: Formatted JSON example in Firefox

JSON Raw Data Headers

Save Copy Pretty Print

```

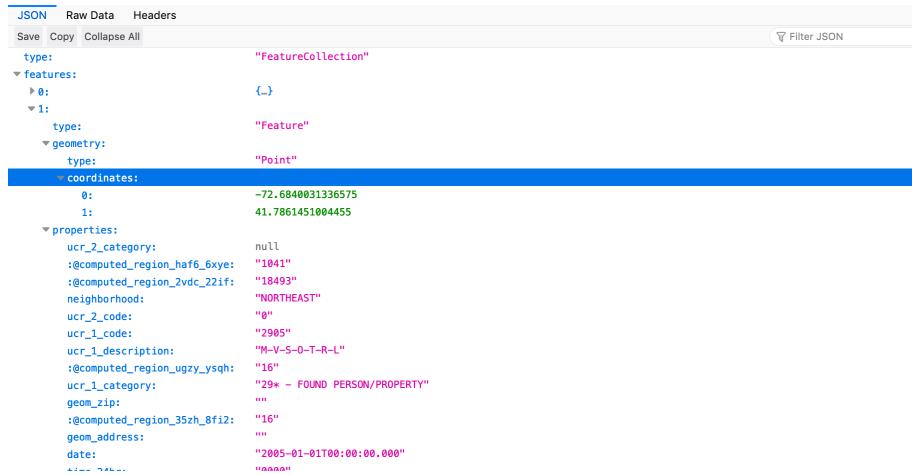
[{"case_number": "9810396", "date": "2005-01-01T00:00:00.000", "time_24hr": "0000", "address": "56 VINE ST", "ucr_1_category": "55* - REPORT-RELATED", "ucr_1_description": "CASE DRAWN IN ERROR", "ucr_1_code": "5520", "ucr_2_code": "0", "neighborhood": "UPPER ALBANY", "geom": {"latitude": "41.7809708152311", "longitude": "-72.6881141026066"}, "human_address": {"address": "", "city": "", "state": "", "zip": ""}, "@computed_region_ugzy_yqsh": "15", "@computed_region_2vdc_22if": "18493", "@computed_region_haf6_6xye": "1041", {"case_number": "5000008", "date": "2005-01-01T00:00:00.000", "time_24hr": "0000", "address": "14 GILMAN ST", "ucr_1_category": "29* - FOUND PERSON", "ucr_1_description": "UCR 1 DESCRIPTION", "ucr_1_code": "5510", "ucr_2_code": "0", "neighborhood": "DOWNTOWN", "geom": {"latitude": "41.7861451004455", "longitude": "-72.6840931336575"}, "human_address": {"address": "", "city": "", "state": "", "zip": ""}, "@computed_region_ugzy_yqsh": "16", "@computed_region_35zh_8fiz": "16", "@computed_region_2vdc_22if": "18493", "@computed_region_haf6_6xye": "1041", {"case_number": "5000024", "date": "2005-01-01T00:00:00.000", "time_24hr": "0000", "address": "115 ASYLUM ST", "ucr_1_category": "55* - REPORT-RELATED", "ucr_1_description": "CASE UNRELATED TO", "ucr_1_code": "5510", "ucr_2_code": "0", "neighborhood": "DOWNTOWN", "geom": {"latitude": "41.7669464534884", "longitude": "-72.6753377122773"}, "human_address": {"address": "", "city": "", "state": "", "zip": ""}, "@computed_region_ugzy_yqsh": "17", "@computed_region_35zh_8fiz": "17", "@computed_region_2vdc_22if": "18493", "@computed_region_haf6_6xye": "1041", {"case_number": "5000009", "date": "2005-01-01T00:00:00.000", "time_24hr": "0000", "address": "127 IRVING ST", "ucr_1_category": "34* - OTHER ACCIDENT", "ucr_1_description": "OCC-INJ-POLICE", "ucr_1_code": "3448", "ucr_2_code": "0", "neighborhood": "UPPER ALBANY", "geom": {"latitude": "41.78011122575092", "longitude": "-72.68611838320887"}, "human_address": {"address": "", "city": "", "state": "", "zip": ""}, "@computed_region_ugzy_yqsh": "18", "@computed_region_35zh_8fiz": "18", "@computed_region_2vdc_22if": "18494", "@computed_region_haf6_6xye": "1041", {"case_number": "5000022", "date": "2005-01-01T00:00:00.000", "time_24hr": "0005", "address": "29 ANNAWAN ST", "ucr_1_category": "19* - CRIMES AGAINST THE PUBLIC", "ucr_1_description": "BREACH-PEACE", "ucr_1_code": "1901", "ucr_2_code": "19", "neighborhood": "BARRY SQUARE", "geom": {"latitude": "41.7492666840371", "longitude": "-72.6754861409539"}, "human_address": {"address": "", "city": "", "state": "", "zip": ""}, "@computed_region_ugzy_yqsh": "19", "@computed_region_35zh_8fiz": "19", "@computed_region_2vdc_22if": "18494", "@computed_region_haf6_6xye": "1041", {"case_number": "5000046", "date": "2005-01-01T00:00:00.000", "time_24hr": "0005", "address": "FOOT GUARD PL & HOADLEY PL", "ucr_1_category": "66-LARCENY", "ucr_1_description": "LARC3-FROM M/V", "ucr_1_code": "1909", "ucr_2_code": "24* - MOTOR VEHICLE LAWS", "ucr_2_description": "MISUSE OF PLATE", "ucr_2_code": "3503", "neighborhood": "DOWNTOWN", "geom": {"latitude": "41.7376529965414", "longitude": "-72.6781666787566"}, "human_address": {"address": "", "city": "", "state": "", "zip": ""}, "@computed_region_ugzy_yqsh": "20", "@computed_region_35zh_8fiz": "20", "@computed_region_2vdc_22if": "18494", "@computed_region_haf6_6xye": "1041", {"case_number": "5000009", "date": "2005-01-01T00:00:00.000", "time_24hr": "0010", "address": "949 ALBANY AV", "ucr_1_category": "5211 - SHOTS FIRED UNCONFIRMED", "ucr_1_description": "SHOTS FIRED UNCONFIRMED", "ucr_1_code": "5211", "ucr_2_code": "0", "neighborhood": "UPPER ALBANY", "geom": {"latitude": "41.7803325207027", "longitude": "-72.6524460975509"}, "human_address": {"address": "", "city": "", "state": "", "zip": ""}, "@computed_region_ugzy_yqsh": "21", "@computed_region_35zh_8fiz": "21", "@computed_region_2vdc_22if": "18494", "@computed_region_haf6_6xye": "1041"
]

```

Figure 11.22: Unformatted JSON example in Firefox

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

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



The screenshot shows a JSON object representing a FeatureCollection. It includes features (0 and 1), each with a geometry (Point) and coordinates (-72.6840311336575, 41.7861451004455). The properties section contains various Socrata-specific fields like ucr_2_category, ucr_2_code, and date.

```

{
  "type": "FeatureCollection",
  "features": [
    {
      "type": "Feature",
      "geometry": {
        "type": "Point",
        "coordinates": [
          -72.6840311336575,
          41.7861451004455
        ]
      },
      "properties": {
        "ucr_2_category": null,
        "ucr_2_code": "1841",
        "neighborhood": "NORTHEAST",
        "ucr_1_code": "2905",
        "ucr_1_description": "M-V-S-O-T-R-L",
        "date": "2005-01-01T00:00:00.000"
      }
    }
  ]
}

```

Figure 11.23: Formatted GeoJSON example in Firefox

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

Register for Socrata App Token

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

Demonstration Maps

GeoJSON endpoint with circle markers and tooltips

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

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

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

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

Multiple Socrata datasets with Leaflet control layers legend

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

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

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

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

Thanks to

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

TODO: blend this section into the one above

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

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

How it works

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



Figure 11.24: Screenshot: Sample API endpoint in Socrata open data repo

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

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

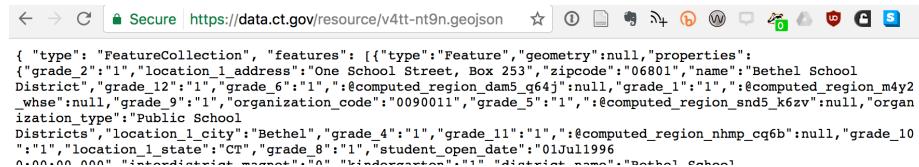


Figure 11.25: Screenshot: API endpoint with .geojson suffix in Chrome browser

- 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=QVVY3I72SVPbxBYI")
  var geoJsonLayer = L.geoJson(data, {
    pointToLayer: function( feature, latlng ) {
      var circle = L.circleMarker(latlng, {
        radius: 6,
        fillColor: "blue",
        color: "blue",
        weight: 2,
        opacity: 1,
        fillOpacity: 0.7
      });
      circle.bindPopup(feature.properties.name + '<br>' + feature.properties.district_name);
      return circle;
    }
  }).addTo(map); // display by default
  controlLayers.addOverlay(geoJsonLayer, 'Public Schools (CT Open Data-Socrata)');
});
```

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

Chapter 12

Transform Your Map Data

All maps, including interactive web maps, are made up of different layers. These are background basemaps, colored or shaded polygons (also known as *choropleth* layers), lines, and point data that are often represented as markers.

In this chapter, we will look at multiple ways to convert and edit geospatial data to create layers (files) that you can use in your favorite mapping tools.

We will begin by looking at the process of geocoding, or transforming human-friendly address lines into points that can be plotted on the map (see Figure 12.1 for inspiration). We will then talk about polygons and why you should normalize your data before creating choropleth maps. These map transformations happen inside spreadsheets, so you won't directly deal with map data until you are halfway through the chapter.

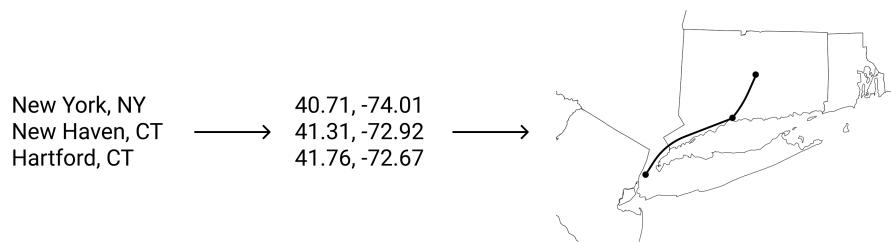


Figure 12.1: To map addresses, you need to geocode them first.

Before you can dive into creating shapes and dealing with boundaries in the map, we will introduce various file formats (most notably GeoJSON) and talk about geospatial data in general. You will learn that map data can be raster and vector, that geospatial data consists of location and attribute components, and how GeoJSON is different from Shapefiles and other geographical data formats.

With our tutorials, you will learn how to convert or draw your own layer of map polygons or polylines on top of satellite imagery using the GeoJson.io tool, and also how to edit geospatial data and join it with spreadsheet data using the Mapshaper tool. Both are powerful, web-based open-source geodata tools that for common tasks can substitute for more complex geographic information system tools, such as ArcGIS or QGIS. Finally, you'll also learn how to georectify a digitized map to display as a background overlay using the MapWarper tool.

By the end of this chapter, you should feel much more confident navigating the overwhelming world of geospatial data.

Geocode Locations into Coordinates with US Census or Google

Before addresses can be mapped, they need to be geocoded. Geocoding is a process of transforming a human-readable address, such as *300 Summit St, Hartford, CT* into a latitude-longitude pair, such as *41.747,-72.692*. These numbers are x- and y-coordinates that maps understand.

If you have just a few addresses, it might be faster to geocode them with Google Maps. Search for an address, right-click on that point, and select *What's here?* to reveal a popup window with its latitude and longitude, as shown in Figure 12.2. You can copy and paste the coordinates into your spreadsheet. Similar tools also geocode one place at a time, such as LatLong.net.

But what if you need to geocode dozens, hundreds, or even thousands of addresses? In this section, we will look at two ways to geocode larger lists of addresses. First, you'll learn how to use our custom-built Google Sheets Geocoder, which lets you convert addresses using Google Geocoder (available pretty much worldwide) and the US Census Geocoder (for US addresses only). Second, you'll learn how to use a stand-alone US Census Geocoder that allows you to upload a file with up to 10,000 addresses within the US, and download geocoded results.

Note: Using Google Maps Geocoder within Google Sheets (App Script) does not require an API key. In the past, free tier was restricted by 1,000 geocoding requests in 24 hours. Since 2018, use quotas are unclear, but we believe the new limit is up to 50 requests per minute.

Geocode addresses with Google Sheets Geocoder

The Google Sheets Geocoder script lives inside a special Google Sheet that you should *copy* to your own Google Drive (you don't need editing access, just go to *File > Make a copy*).

The spreadsheet contains six columns. Populate the first column, *Location*, with your addresses. The remaining five columns will be filled by the geocoding

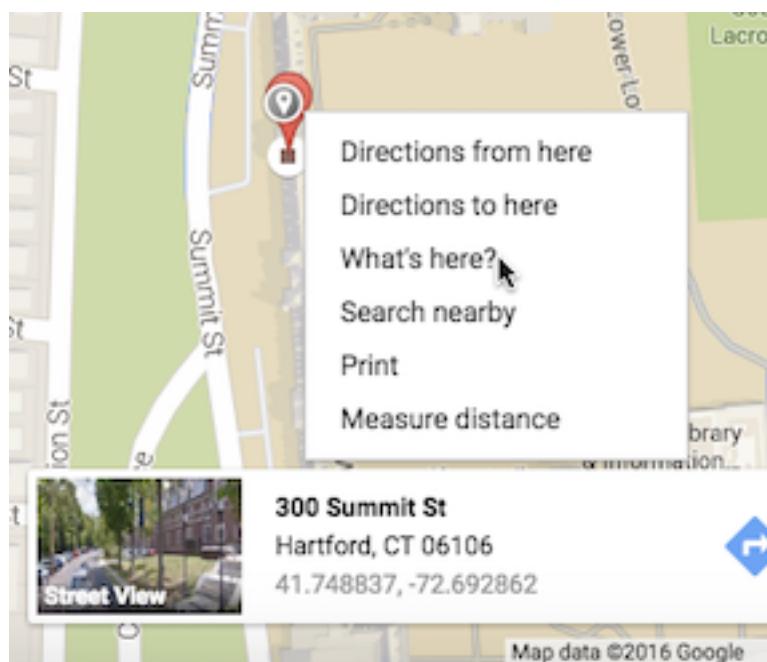


Figure 12.2: To geocode one address, search in Google Maps and right-click *What's here?* to show coordinates.

script. Select all six columns, go to *Geocoder* in the menu, and choose which geocoding utility to use, like is shown in Figure 12.3.

A	B	C	D	E	F
Location	Latitude	Longitude	Found	Quality	Source
300 Summit Street, Hartford, CT 06106					
4 Frederick Rd, West Hartford CT					
East Capitol St NE & First St SE, Washington, DC 20004					
2329 West Mall, Vancouver, BC V6T 1Z4, Canada					
Av. Eugenio Garza Sada 2501 Sur, 64849 Monterrey, N.L., Mexico					

Figure 12.3: Put addresses in the first column, and use Geocoder to fill in the remaining five.

Note: If your address data is split into multiple columns (such as *Street*, *City*, and *State*), revisit Clean Data with Spreadsheets section to remind yourself how to “glue” multiple cells into one.

The first time you run the Geocoder script, it will ask for your permission. TODO: since we still need Google to authorize our script, you will need to click *Advanced* and then *Go to Geocoder (unsafe)* near the bottom of the next screen. Trust us – our script is safe to use! The code is open-source and is available on GitHub, so you or your programmer friend can make sure it doesn’t steal your personal data. Sorry for the inconvenience.

Once the script finishes executing, you will get a pop-up notification that will tell you how many addresses were successfully geocoded, and how many failed. Inspect *Found* and *Quality* columns to ensure the geocoder matched your addresses correctly. Then look at the failed addresses and see if you can spot problems with them. For tips about using Google geocoder, see documentation.

Note: The Geocoder plugin is a small Apps Script program that is connected to your Google sheet. It sends your addresses to either US Census Geocoder, or Google Geocoding API, and gets geocoded results as a response.

Geocode US addresses to census tracts with Google Sheets Geocoder

You can use a modified version of the Google Sheets Geocoder, available in its own spreadsheet, to assign census tracts and GeoIDs to addresses within the United States.

A GeoID is a unique identifier of a place according to the US Census. A sample 15-digit GeoID, 090035245022001, consists of a state (09), followed by county (003), followed by census tract (524502, or more conventional 5245.02), followed by a census block group (2), and finally a census block (001).

Make a copy of the template spreadsheet into your own Google Drive by going to *File > Make a copy*.

You only need to populate the first column, *Location*. The rest seven columns will be populated by the Geocoder. Similar to the previous template, select all eight columns, and go to *Geocoder > US Census 2010 Geographies*, like is shown in Figure 12.4.

If you run this script for the first time, Google Sheets will ask you for permission to run, and will possibly warn you that this script is unsafe. Once again, you shouldn't worry. The plugin is open-source and you can inspect it to make sure it doesn't steal or retain your personal data.

	A	B	C	D	E	F	G	H
1	Location	Latitude	Longitude	Found	Quality	Source	GeoID	Tract
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								

Figure 12.4: Put addresses in the first column, and use Geocoder to fill in the remaining seven.

Insert Google Sheets Geocoder script into your own spreadsheet

If you don't want to make a copy of the Google Sheet templates from the previous examples, you can insert the open-source Geocoder scripts into your own Google sheet.

1. In your personal Google spreadsheet, go to *Tools > Script Editor*. This should open up a new tab.
2. Replace the empty `function myFunction()` with the contents of `geocoder-census-google.gs` from the plugin's repo on GitHub.
3. In Script Editor, click *File > Save*. An *Edit Project Name* window will pop up, where you should give the script a meaningful name, such as "Geocoder".
4. Close the Script Editor, and go back to your spreadsheet. Refresh and wait for a couple of seconds. *Geocoder* should appear in the menu.

Geocode up to 10,000 US addresses with US Census Geocoder

One of the fastest ways to geocode up to 10,000 US addresses at a time is to create a CSV file with 5 columns and upload it to Address Batch form of the US

Census Geocoder. In the menu on the left-hand side, you can switch from *Find Locations* to *Find Geographies* if you wish to include census tract and GeoID data in addition to the coordinates.

Your CSV file **must not contain a header row**. It needs to be formatted the following way:

```
| 1 | 300 Summit St | Hartford | CT | 06106 |
| 2 | 1012 Broad St | Hartford | CT | 06106 |
```

Here, the first column is unique IDs (make sure it is unique to each address, but they don't have to start at 1 or be in an increasing order). The second column is street address. The third column is city. Column four is state, and the final fifth column is zip code.

Upload the file using the *Browse...* button of *Select Address File*, use *Public_AR_Current Benchmark*, and hit *Get Results*.

In a few moments (it usually takes longer for larger files), the tool will return a file named *GeocodeResults.csv* with geocoded results. Save it, and inspect it in your favorite spreadsheet tool. The resulting file is an eight-column CSV file with the original ID and address, match type (exact, non-exact, tie, or no match), and latitude/longitude coordinates.

Getting a *tie* matching means there are multiple possible results for your address. To see all possible matches of an address that got a *tie*, use *One Line* or *Address* tools in the left-hand side menu and search for that address.

Tip: If you see some unmatched addresses, use a filtering functionality of your spreadsheet to filter for unmatched addresses, then manually correct them, save as a separate CSV file, and re-upload. You can use the US Census Geocoder as many times as you want, as long as a single file doesn't exceed 10,000 records.

In reality only the first two columns, *unique ID* and *street address*, are required for the US Census Geocoder to accept your file for processing. City, state, and zip code values may be left blank if you don't have that data. But to ensure you get exact matches, you should provide as much data as is available to you.

If your data lacks ID values, you can create a column of consecutive numbers. See Calculate with Formulas and Functions section of this book to see how.

Make sure your street addresses don't contain city, state, and zip code data. If they do, use splitting text to columns technique, described in the Clean Data with Spreadsheets section of the book, to get rid of that extra data. But if your street addresses contain apartment numbers, you can leave them in.

Note: US Census Geocoder has a comprehensive overview and documentation that you can refer to if you encounter issues not covered here.

If for some reason you cannot geocode address-level data, but you need to produce some mapping output, you can use pivot tables to get counts of points

for specific areas, such as towns or states. In the next section, we will look at hospital addresses in the US and how we can count them by state using pivot tables.

Pivot Address-Level Point Data into Polygon Data

If you deal with geographical data, you may find yourself in a situation where you have a list of addresses which need to be counted (*aggregated*) by area and displayed as a polygon map. In this case, a simple pivot table in a spreadsheet software can solve the problem.

Note: A special case of a polygon map is a *choropleth* map, which represents polygons that are colored in a particular way to represent underlying values. A lot of polygon maps end up being *choropleth* maps, so we will be using this term a lot in this book.

Let's take a look at a list of all hospitals that are registered with the Medicare program in the United States. The dataset is stored and displayed by Socrata, a web database popular among government agencies and city administrations. This particular dataset has information on each hospital's name, location (nicely divided into Address, City, State, and ZIP Code columns), a phone number and some other indicators, such as mortality and patient experience.

Now, imagine you are given a task to create a choropleth map of total hospitals by US state. Instead of showing individual hospitals as points (as in Figure 12.5a), you want darker shades of blue to represent states with more hospitals (as in Figure 12.5b).

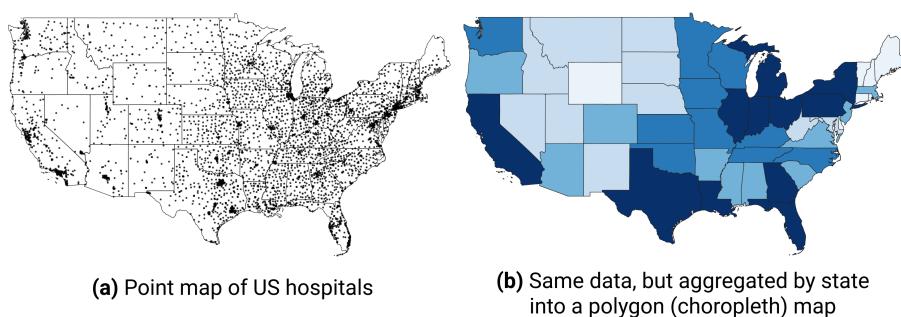


Figure 12.5: You can count addresses by state (or other area) to produce polygon, or choropleth, maps instead of point maps.

First, save the database to your local machine by going to *Export > Download > CSV* of Socrata interface. Figure 12.6 shows where you can find the Export button.

Figure 12.6: In Socrata, you can export the entire dataset as a CSV.

Next, open the file in your favorite spreadsheet tool. If you use Google Sheets, use *File > Import > Upload* to import CSV data. Make sure your address columns are present, and move on to creating a pivot table (in Google Sheets, go to *Data > Pivot table*, make sure the entire data range is selected, and click *Create*). In the pivot table, set *Rows* to *State*, because we want to get counts by state. Next, set pivot table's *Values* to *State*—or really any other column that has no missing values—and choose *Summarize by: COUNTA*. Voila!

Your aggregated dataset is ready, so save it as a CSV. If you use Google Sheets, go to *File > Download > Comma-separated values (.csv, current sheet)*. You can now merge this dataset with your polygons manually using editing capabilities of GeoJson.io, or merge it all in one go using powerful Mapshaper.

We will introduce both tools in the next few sections. But before we do that, let's talk about data normalization and why showing counts of hospitals per state doesn't really tell a good story.

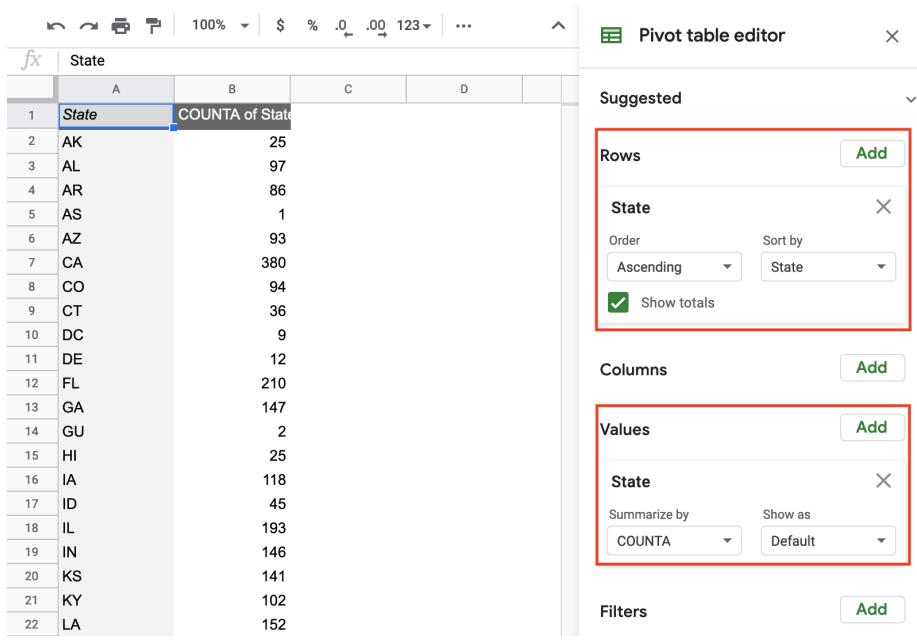


Figure 12.7: Use pivot tables in any spreadsheet software to count addresses per area (such as state, county, or zip code).

Normalize Data to Create Meaningful Choropleth Maps

Choropleth maps are best when they represent relative, not absolute values. Consider two maps shown in Figure 12.8. They both are about Covid-19 cases in the US states (excluding Alaska and Hawaii) as of June 26, 2020. Figure 12.8a shows total number of recorded cases per state, and Figure 12.8b shows Covid-19 cases adjusted by the state's population. Darker colors represent higher values. Do you notice any differences in spatial patterns?

Both maps show Covid-19 data collected by the New York Times and published on GitHub. In the map in Figure 12.8b, we normalized (divided) values by population in each state, according to the 2018 US Census American Community Survey, the most recent data available on the day of writing. We didn't add legends and other important cartographic elements so that you can better focus on interpreting spatial patterns. In both cases, we used Jenks natural breaks for classification.

What are the worst-hit states according to the map showing total Covid-19 counts (shown in Figure 12.8a)? If you are familiar with the US geography, you can quickly tell that these are New York, New Jersey, Massachusetts, Florida,

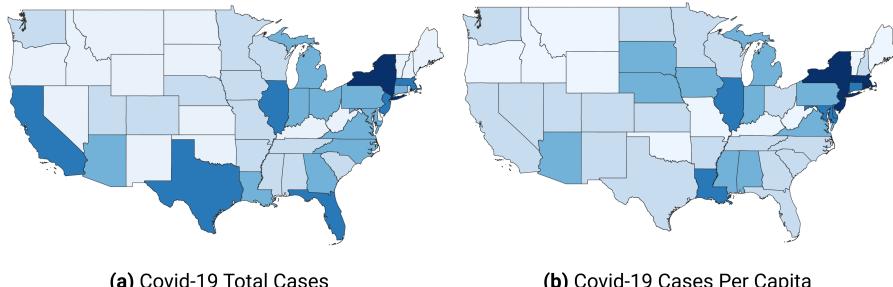


Figure 12.8: Choropleth maps work best with normalized values.

Illinois, Texas, and California. But five of these happen to be some of the most populous states in the US, so it makes sense that they will also have higher Covid-19 cases.

Now, how about the map in Figure 12.8b? You can see that New York and its neighbors, including New Jersey and Massachusetts, have by far the highest rates per capita (per person), which we saw in the first map. But you can also see that in fact California, Texas, and Florida were impacted to a lesser extent than the map on the left had suggested. So the map with per-capita values is a much better illustration to the story about New York being the *first* epicenter of the Covid-19 crisis in the United States.

Different ways to normalize data

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

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

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

Unlike counts, most *measured* variables do not need normalization because they belong to a scale. For example, median age (the age of the “middle” person in a population, when sorted from youngest to oldest) can be directly compared among populations. We know that humans live anywhere between 0 and 120.

years or so, and we wouldn't expect median ages to be vastly different from one country to another (maybe twice, but not tenfold). Median incomes, if measured in the same currency, also belong to the same scale and can be compared directly.

How not to normalize values

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

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

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

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

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

Convert to GeoJSON format

Geospatial data comes in an overwhelming number of file formats. We will tell you about a few most common ones so that you have a general idea of what tools you can use to work with them. But before we do that, let's talk about the basics of geospatial (map) data.

About geospatial data

The first thing to know about geospatial data is that it consists of two components, *location* and *attribute*. When you use Google Maps to search for a restaurant, you get a red marker on the screen that points to the latitude and

longitude of the physical location of the restaurant in the real world. These latitude and longitude (two numbers) are your location component. The name of the restaurant, its human-friendly address, and guest reviews are the attributes, which bring value to your location data.

Second, geospatial data can be *raster* or *vector*, as illustrated in Figure 12.9. Raster data, as shown in Figure 12.9a, is a grid of cells (“pixels”) of a certain size (for example, 1 meter by 1 meter). For example, satellite images of the Earth that you see on Google Maps are raster geospatial data. Each pixel contains the color of Earth that satellite cameras were able to capture. People and algorithms can then use raster data (images) to create outlines of buildings, lakes, roads, and other objects. These outlines become vector data. For example, most of OpenStreetMap was built by volunteers tracing outlines of objects from satellite images.

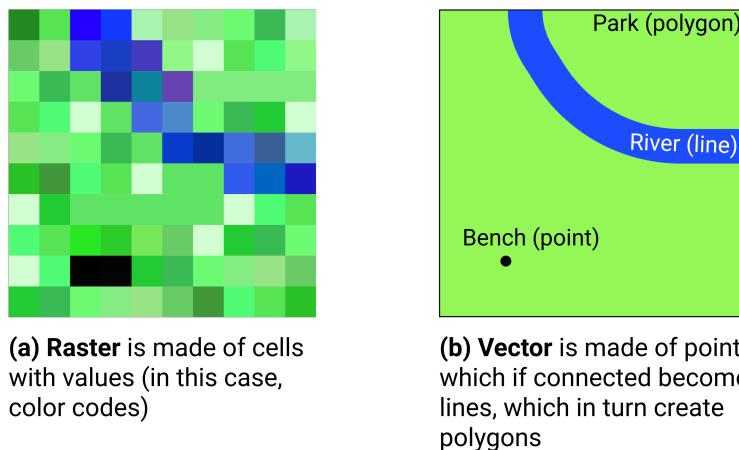


Figure 12.9: Geospatial data can be raster or vector.

In this book, we will focus on vector data, which is based on points, lines, and polygons, as shown in Figure 12.9b. Vector data can be much more precise than raster data, because points’ coordinates can be expressed with precise decimals. In addition, vector data can contain as much extra *attribute* information about each object as desired, whereas raster data is generally limited to 1 value per cell, whether it is the Earth color, or temperature, or altitude. Moreover, vector map files are usually much smaller in size than raster ones.

Let’s take a look at some of the most common vector file formats.

GeoJSON

GeoJSON is a newer, popular open format for map data that comes in `.geojson`

or `.json` files. It was first developed in 2008, and then standardized in 2016 by the Internet Engineering Task Force (IETF). The code snippet below represents a single point (feature) with latitude of 41.76 and longitude of -72.67 in GeoJSON format. That point has a *name* attribute (property) whose value is *Hartford*.

```
{
  "type": "Feature",
  "geometry": {
    "type": "Point",
    "coordinates": [-72.67, 41.76]
  },
  "properties": {
    "name": "Hartford"
  }
}
```

The simplicity and readability of GeoJSON allows you to edit it even in the most simple text editor. We strongly recommend you use and share your map data in GeoJSON. Web-based maps, such as those built with Leaflet, Mapbox, Google Maps JS API, and Carto, as well as ArcGIS and QGIS all support GeoJSON. By having your geospatial data stored and shared in GeoJSON, you ensure you can use it on the web with nearly any mapping tool. You can also be confident that other people will be able to use and extract data from the file without bulky and often expensive GIS software installed.

Also, your GitHub repository will automatically display any GeoJSON files in a map view, like is shown in Figure 12.10.

Warning: In GeoJSON, coordinates are ordered in *longitude-latitude* format, the same as X-Y coordinates in mathematics. This is the opposite of Google Maps and some other web map tools, which order values as *latitude-longitude*. For example, *Hartford, Conn.* is located at (-72.67, 41.76) according to GeoJSON, but at (41.76, -72.67) in Google Maps. Neither notation is right or wrong, just make sure you know which one you are dealing with. Tom MacWright created a great summary table showing lat/lon order of different geospatial formats and technologies.

Shapefiles

The shapefile format was created in the 1990s by Esri, the company that develops ArcGIS software. Shapefiles typically appear as a folder of subfiles with suffixes such as `.shp`, `.shx`, and `.dbf`. The folder with shapefiles is often compressed in a `.zip` file.

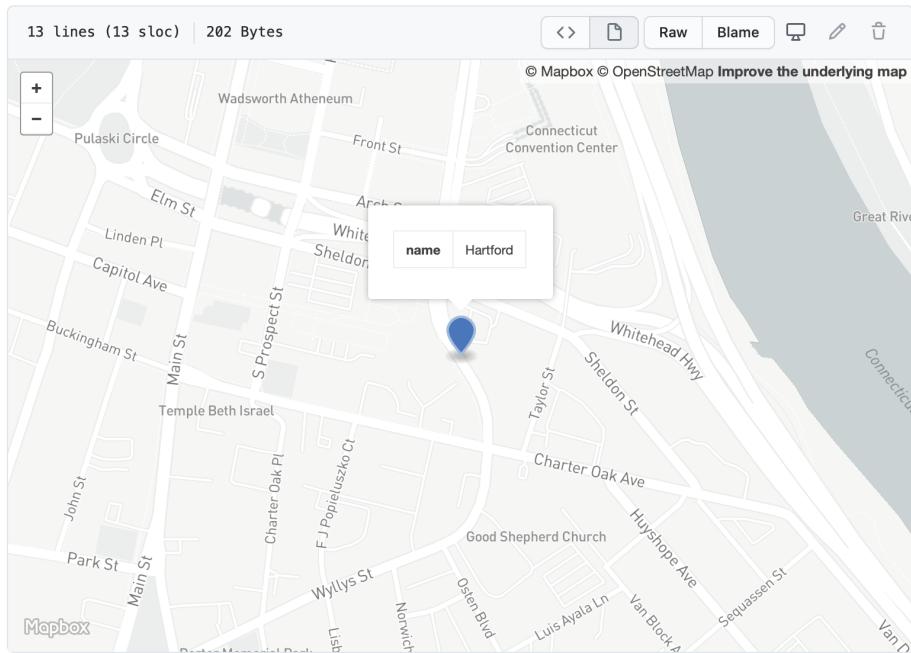


Figure 12.10: GitHub can show previews of GeoJSON files stored in repositories.

Although government agencies commonly distribute map data in shapefile format, the standard tools for editing these files—ArcGIS and its free and open-source cousin, QGIS—are not as easy to learn as other tools in this book. For this reason, we recommend converting shapefiles into GeoJSON files if possible. Mapshaper, discussed a bit later in the chapter, can perform such conversion.

GPS Exchange Format (GPX)

If you ever exported your Strava run or a bike ride from a GPS device, chances are you ended up with a .gpx file. GPX is an open standard and is based on XML markup language. Like GeoJSON, you can inspect a GPX file in any simple text editor to see its contents. Most likely, you will see a collection timestamps and latitude/longitude coordinates of the recording GPS device at that particular time. You should be able to convert GPX to GeoJSON with GeoJson.io utility discussed later in this chapter.

Keyhole Markup Language (or KML)

The KML format rose in popularity during the late 2000s. It was developed for Google Earth, a free and user-friendly tool that allowed many people to view

and edit two- and three-dimensional geographic data. KML files were often used with maps powered by Google Fusion Tables, but that became history in late 2019. GeoJson.io should be able to convert your KML file into a GeoJSON.

Sometimes .kml files are distributed in a compressed .kmz format. See Converting from KMZ to KML format section of this book to learn to convert.

MapInfo TAB

Similar to Esri's shapefiles, MapInfo's TAB format comes as a folder with .tab, .dat, .ind, and some other files. It is a proprietary format created and supported by MapInfo, Esri's competitor, and is designed to work well with MapInfo Pro GIS software. Unfortunately, you will most likely need MapInfo Pro, QGIS, or ArcGIS to re-save these as GeoJSON or a Shapefile.

We've mentioned only a handful of the most common geospatial file formats. There is a myriad of other, less known formats for both raster and vector data. Remember that GeoJSON is one of the best, most universal formats for your *vector* data, and we strongly recommend to store and share your map data in GeoJSON. In the next section, we will look at free online tools to create, convert, join, crop, and in other ways manipulate GeoJSON files.

GeoJson.io to Convert, Edit, and Create Map Data

GeoJson.io is a popular open-source web tool to convert, edit, and create GeoJSON files. The tool was originally developed by Tom MacWright in 2013 and quickly became a go-to tool for geospatial practitioners.

In this tutorial, we will show you how to convert existing KML, GPX, TopoJSON, and even CSV files with lat/lon data into GeoJSON files. We will also look at editing attribute data and adding new features to GeoJSON files, and creating them from scratch by tracing satellite imagery.

Convert KML, GPX, and other formats into GeoJSON

Navigate to GeoJson.io. You will see a map on the left, and a Table/JSON attribute view area on the right. At the start, it represents an empty feature collection (features are your points, lines, and polygons).

Drag and drop your geospatial data file into the map area on the left. Alternatively, you can also import a file from *Open > File* menu. If you don't have a geospatial file, download Hartford parks in KML format. If GeoJson.io was able to recognize and import the file, you will see a green popup message in the

upper-left corner saying how many features (in case of Hartford parks, polygons) were imported. Figure 12.11 shows us that 62 features were imported from the sample Hartford parks file. You can see that the polygons appeared on top of the Mapbox world layer.

Note: If GeoJson.io couldn't import your file, you will see a red popup saying it "Could not detect file type". You will need to use a different tool, such as Mapshaper or QGIS, to convert your file to GeoJSON.

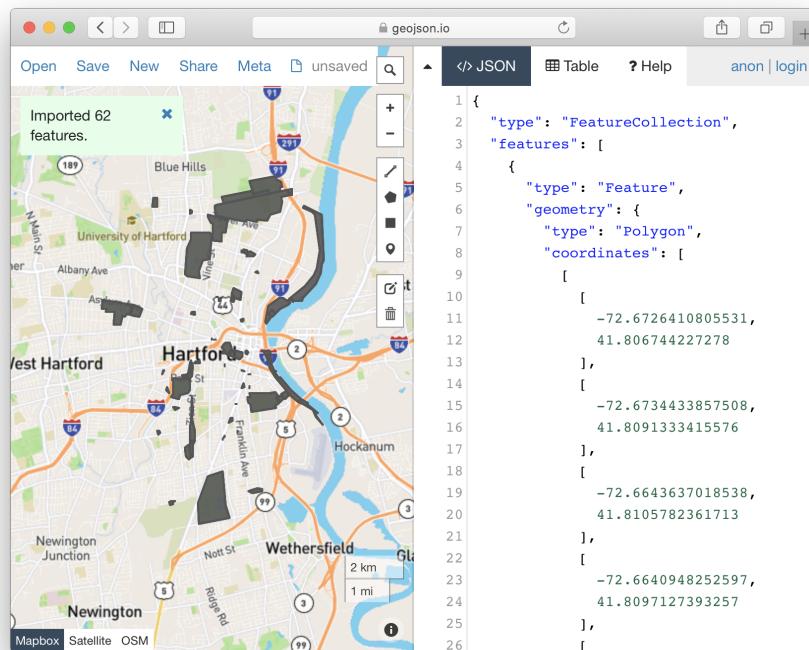


Figure 12.11: GeoJson.io successfully imported Hartford parks KML file.

You can now save your file to GeoJSON. Go to *Save > GeoJSON* to download a converted GeoJSON file to your computer.

Create GeoJSON from a CSV file

GeoJson.io can transform a spreadsheet with *latitude* (or *lat*) and *longitude* (or *lon*) columns into a GeoJSON file of point features. Each row in the spreadsheet becomes its own point, and all columns other than *lat* and *lon* become *attributes* (or *properties*) of point features. An example of such spreadsheet is shown in Figure 12.12. You can download it for the exercise.

A	B	C	D	E
town	lat	lon	community_type	wiki_link
Hartford	41.76	-72.67	urban core	https://en.wikipedia.org/wiki/Hartford,_Connecticut
Bloomfield	41.85	-72.73	urban periphery	https://en.wikipedia.org/wiki/Bloomfield,_Connecticut
West Hartford	41.76	-72.75	urban periphery	https://en.wikipedia.org/wiki/West_Hartford,_Connecticut
Wethersfield	41.71	-72.65	urban periphery	https://en.wikipedia.org/wiki/Wethersfield,_Connecticut
Avon	41.79	-72.86	suburban	https://en.wikipedia.org/wiki/Avon,_Connecticut
Glastonbury	41.69	-72.54	suburban	https://en.wikipedia.org/wiki/Glastonbury,_Connecticut

Figure 12.12: A spreadsheet with lat/lon columns can be transformed into a GeoJSON with point features.

1. Save your spreadsheet as a CSV file, and drag-and-drop it to the map area of GeoJson.io. You should see a green popup in the upper-left corner notifying you how many features were imported.

Note: If you had some data on the map already, GeoJson.io wouldn't erase anything but instead would add point features to the existing map.

2. Click on a marker to see a popup with point properties. If you used the sample file with towns around Hartford, you will see *town*, *community_type*, and *wiki_link* features in addition to the tool's default *marker-color*, *marker-size*, and *marker-symbol* fields.

Tip: The popup is interactive, and you can click and edit each property (including property names). You can also add a new property by clicking the *Add row* button. You can delete the marker by clicking *Delete feature* button.

3. Click *Save* to record all marker changes to the GeoJSON. This will close the popup window, and you will see updated markers in the JSON tab to the right of the map.
4. It may be quicker to view all data as a table instead of dealing with individual marker popups. In the *Table* tab to the right of the map, you can add, rename, and remove columns from *all* features (markers) at once. Table cells are also modifiable, so you can edit your data there.
5. Once you are happy with your map data, go to *Save > GeoJSON* to download the result to your computer. You can also log into GeoJson.io with your GitHub account and save directly to your repository.

Create a GeoJSON from scratch using drawing tools

GeoJson.io lets you create geospatial files from scratch, using simple drawing tools to put markers (points), lines, and polygons to appropriate locations. These are useful when you have no original file to work with. The following steps will show you how to create a new GeoJSON file and add markers, lines, and polygons to it.

1. Open GeoJson.io and in the lower-left corner switch from Mapbox (vector tiles) to Satellite.
 2. In the upper-right corner of the map, use the Search tool to find the area you're interested in mapping. For this exercise, we will use tennis courts at Trinity College, Hartford, as shown in Figure 12.13.

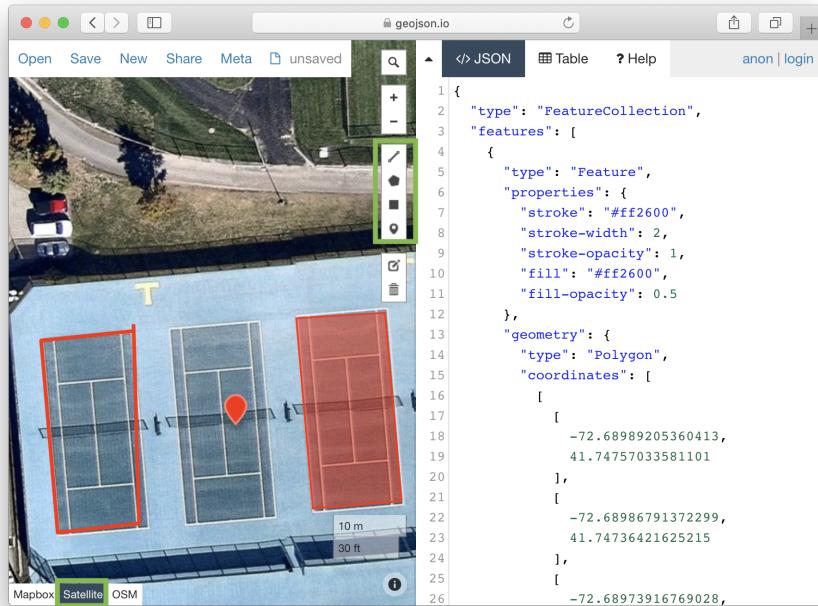


Figure 12.13: Use drawing tools to create points, lines, and polygons in GeoJSON.io.

3. In the toolbar, you have a choice of four drawing tools: a polyline (which is a series of points connected by lines, but not closed like a polygon), a polygon, a rectangle (which is just an instance of a polygon), and a marker (point). Let's start by creating a marker.
 4. Click on the *Draw a marker* button, and click anywhere on the map to place it. You will see a gray marker that is now part of your map. You can modify its properties, or delete it in the interactive pop-up.
 5. Next, choose *Draw a polyline* and click on multiple locations in the map to see connected lines appearing. To finish and create a feature, click again on the final point. Polylines are generally used for roads and paths.
 6. Drawing a polygon is similar to drawing a polyline, except that you need to complete the feature by making your final point at the same location

as your initial point. Polygons are used to define object boundaries, from continents to buildings, cars, and anything that has significant dimensions.

7. Use *Edit layers* tool (the one above *Delete*) to move a marker to a better position, or adjust the shapes of your features.

Once you are done creating features and their physical boundaries, it is time to add meaningful attribution data. Use the interactive popups or the Table view to give objects names and other qualities. When finished, save the GeoJSON to your computer.

Drawing tools can be used to correct your existing GeoJSON files. For example, if you created a GeoJSON from a CSV file, you might decide to move some markers with *Edit layers* tool instead of modifying their latitude and longitude values. Or you might decide that your polygons (eg those representing Hartford parks) are too “simplified”, and make them more precise with the satellite imagery.

In the next section, we will introduce Mapshaper, another free online tool to convert and modify geospatial files.

Mapshaper to Convert, Edit, and Join Data

Like GeoJson.io, Mapshaper is a free, open-source editor that can convert geospatial files, edit attribute data, filter and dissolve features, simplify boundaries to make files smaller, and many more. Unlike GeoJson.io, Mapshaper doesn’t have drawing tools, so you won’t be able to create geospatial files from scratch.

Mapshaper is developed and maintained by Matthew Bloch on GitHub. It is written in JavaScript, so we recommend you use a recent version of Firefox or Chrome.

This free and easy-to-learn Mapshaper web tool has replaced *many* of our map preparation tasks that previously required expensive and hard-to-learn ArcGIS software, or its free but still-challenging-to-learn cousin, QGIS. Even advanced GIS users may discover Mapshaper to be a quick alternative for some common but time-consuming tasks.

Import, convert, and export map boundary files

You can use Mapshaper to convert between geospatial file formats. Unlike GeoJson.io, Mapshaper also supports Esri Shapefiles (which is a folder of individual files with the same name, but different file extensions), so you can easily convert a Shapefile into a web-friendly GeoJSON. In the following steps, we will convert a geospatial file by import it to Mapshaper, and then export it as a different file type.

1. Navigate to Mapshaper.org. The start page is two large drag-and-drop zones which you can use to import your file. The smaller area at the bottom, *Quick import*, uses default import settings and is a good way to begin.
2. Drag and drop your geospatial file to the *Quick import* area, or use our sample Shapefile of US state boundaries. This is a `.zip` archive which contains a folder with all necessary files.

Note: If you want to import a folder, you need to either select all files inside that folder and drop them all together to the import area, or create a `.zip` archive.

3. Each imported file becomes a layer, and is accessible from the dropdown menu in the top-middle of the browser window. There, you can see how many features each layer has, toggle their visibility, or delete them.
4. To export, go to *Export* in the upper-right corner, and select a desired file format. The choice of export formats is shown in Figure 12.14. As of July 2020, these are Shapefile, GeoJSON, TopoJSON (similar to GeoJSON, but with topographical data), JSON records, CSV, or SVG (Scalable Vector Graphics, for web and print). If you export more than one layer at a time, Mapshaper will archive them first, and you will download an `output.zip` that contains all exported layers.

Tip: Mapshaper doesn't work with KML or KMZ files, but you can use GeoJSON.io to convert these.

Edit data for specific polygons

You can edit attribute data of individual polygons (and also points and lines) in Mapshaper. Figure 12.15 shows you how.

1. Import the file whose polygon attributes you want to edit.
2. Under the cursor tool, select *edit attributes*.
3. Click on the polygon you want to edit. A pop-up will appear in the upper-left corner listing all attributes and values of the polygon.
4. Click on any value (underlined, in blue) and edit it.
5. When you are done, export your geospatial file by clicking *Export* and choosing the desired file format.

Simplify map boundaries to reduce file size

You may not need precise and detailed map boundaries for data visualization projects where zoomed-out geographies are shown. Detailed boundaries are heavy, and may slow down your web maps.

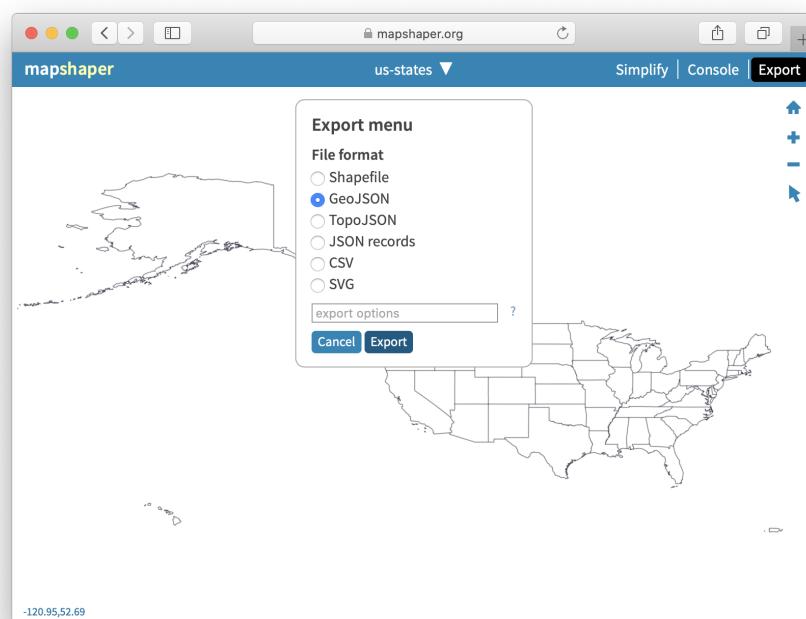


Figure 12.14: You can use Mapshaper to quickly convert between geospatial file formats.

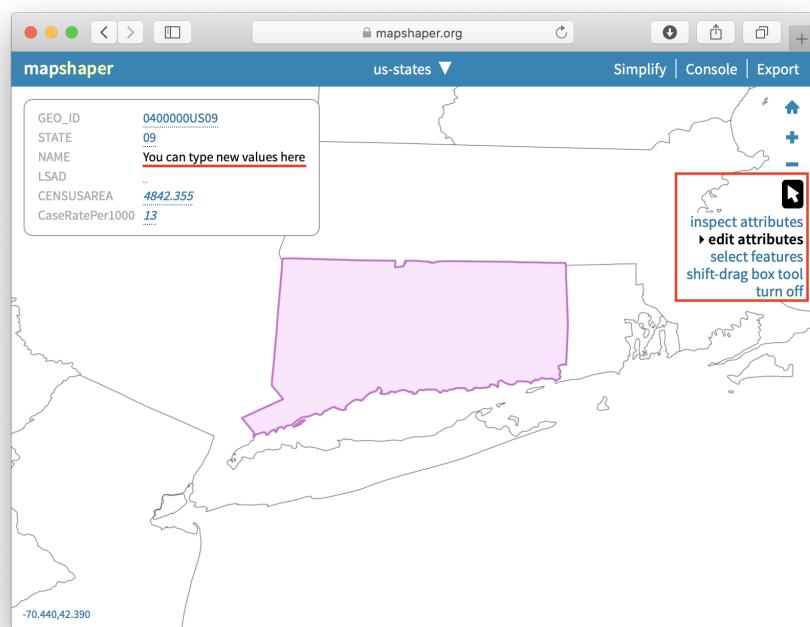


Figure 12.15: Use *edit attributes* tool (under Cursor tool) to edit attributes of polygons, lines, and points.

Consider two maps of the contiguous US states (also known as *the lower 48*, the term Ilya learned in 2018 while travelling in Alaska), shown in Figure 12.16. The map in Figure 12.16a is more detailed and is about 230 kilobytes, but the map in Figure 12.16b is only 37 kilobytes, 6 times smaller!



Figure 12.16: Consider simplifying geometries with Mapshaper to make your web maps faster.

To simplify map boundaries in Mapshaper, follow the steps below.

1. Import your geo file to Mapshaper. You can use the sample contiguous US states GeoJSON.
2. Click the *Simplify* button in the upper-right corner. The Simplification menu will appear, where you can choose one of three methods. We recommend checking *prevent shape removal*, and leaving the default *Visvalingam / weighted area*. Click *Apply*.
3. You will see a slider with 100% appear on top (Figure 12.17), replacing the layer selection dropdown. Move the slider to the right and see the map simplify its shape as you go. Stop when you think the map looks appropriate (when the shapes are still recognizable).
4. Mapshaper may suggest to repair line intersections in the upper-left corner. Click *Repair*.
5. You can now export your file using the *Export* feature.

Tip: You may find the US shape a bit unusual and vertically “shrunk”. In **Console**, type `-proj EPSG:3857` to change projection to Web Mercator, which is more common.

Dissolve internal polygons to create an outline map

Mapshaper’s most powerful tools are available through the *Console*, which allows you to type commands for common map editing tasks. One of such tasks is to create an outline map by removing the internal boundaries. For example,

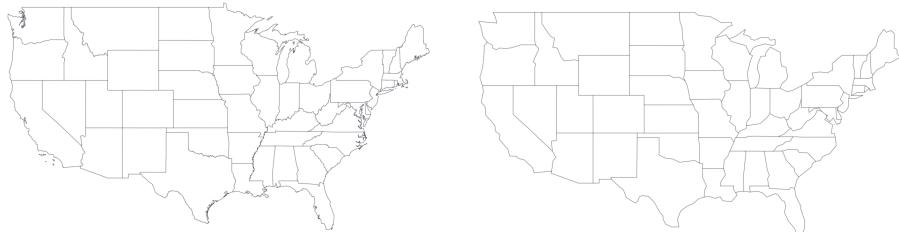


Figure 12.17: Use Simplify & Repair tools in Mapshaper.

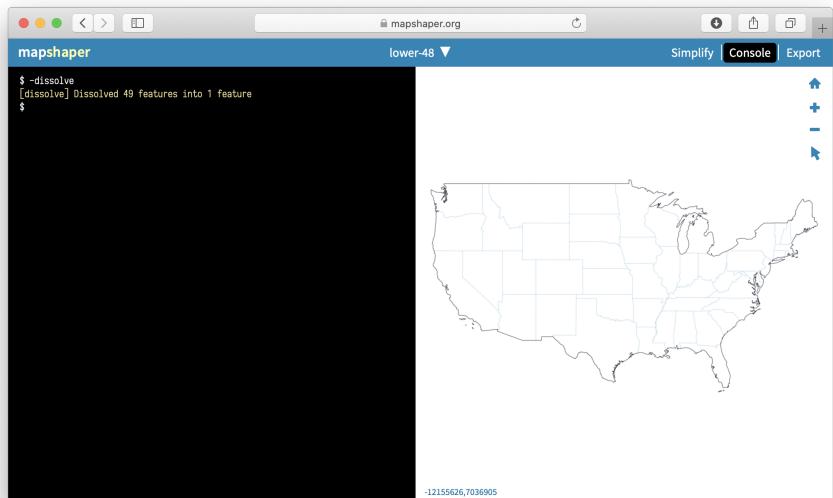


Figure 12.18: Mapshaper lets you dissolve boundaries to create an outline shape.

you can dissolve state boundaries of the US map in the previous exercise to get the outline of the country, like is shown in Figure 12.18.

Click the Console button, which opens a window to type in commands. Enter the command below, then press return (Enter).

```
-dissolve
```

You will see that internal boundaries became lighter color, and that's Mapshaper's way of saying they no longer exist. You can now export your outline shape.

Clip a map to match an outline layer

The state of Connecticut consists of 8 counties, which in turn are divided into towns. There are a total of 169 towns in Connecticut. Imagine you are given a boundary file of all 169 towns, and the outline of Hartford county. You need to "cut" the original towns map to only include those towns that fall within Hartford county.

Mapshaper allows you to do just that using one simple `-clip` command.

1. Import two boundary files into Mapshaper. One is the larger one that is being clipped (if you use sample files, *ct-towns*), and one is the desired final shape (*hartfordcounty-outline*). The latter is what ArcGIS calls the "clip feature".
2. Make sure your active layer is set to the map you are clipping (*ct-towns*).
3. In the *Console*, type `-clip` followed by the name of your clip layer, like that:

```
-clip hartfordcounty-outline
```

4. You should see your active layer got clipped. Sometimes you end up with tiny "slivers" of clipped areas that remain alongside the borders. If that is the case, use the `-filter-slivers` command to remove them, like that:

```
-clip hartfordcounty-outline -filter-slivers
```

5. Your Mapshaper state should look like pictured in Figure 12.19. You can now save the file on your computer using the *Export* button.

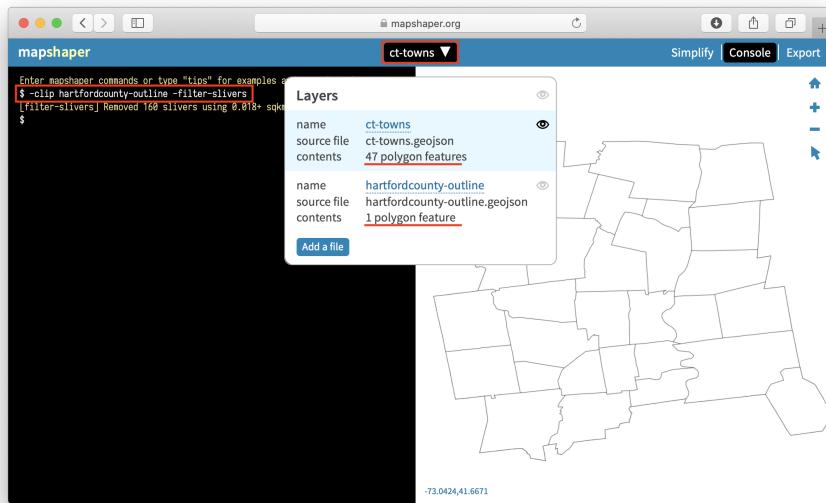


Figure 12.19: When clipping, make sure your active layer is the one being clipped (with many features), not the clipping feature itself.

Remove unwanted data fields

Sometimes map features, such as polygons, lines, and points, contain unwanted *attributes* (or fields, or columns) that you may want to remove. In the *Console*, type the `-filter-fields` editing command to remove unnecessary fields.

For example, remove all fields except *town*:

```
-filter-fields town
```

If you want to leave more than one field, separate them by a comma, but without spaces, like that:

```
-filter-fields town,state
```

Warning: If you leave a space after comma, you will get a *Command expects a single value* error.

Join spreadsheet data with polygon map

Combining spreadsheet data with geographical boundaries is a common task for geospatial practitioners. Imagine you have a file with Connecticut town

boundaries, and you want to add population data to each of them in order to build a choropleth map.

Mapshaper provides a powerful `-join` command to join such files. Remember that you need some common keys in both datasets (such as *town name*, or *state*, or *country*) in order to join files. Otherwise Mapshaper has no way of knowing which numbers belong to which polygons.

1. Import both geospatial file and a CSV dataset into Mapshaper using Quick import box.
2. Make sure both files appear in the drop-down list of layers. Your CSV data will be shown as something that resembles a table. Use the *Cursor > inspect attributes* tool to make sure the data is imported correctly. If you use the sample CT files, note that the *ct-towns* layer has *name* attribute with the name of the town, and *ct-towns-popdensity* has town names in the *town* column.
3. Make your geospatial layer (*ct-towns*) is the one active.
4. Open the *Console*, and use the `-join` command, like this:

```
-join ct-towns-popdensity keys=name,town
```

where *ct-towns-popdensity* is the CSV layer you are merging with, and **keys** are the attributes that contain values to join by. In case with our sample files, these would be town names which are stored in **name** attribute of the map file, and **town** column of the CSV file.

5. You will see a message in the console notifying you if join was performed successfully, or if Mapshaper encountered any errors.
6. Use the *Cursor > inspect attributes* tool to make sure you see CSV columns as fields of your polygons, like is shown in Figure 12.20.
7. You can now save the file to your computer by clicking the *Export* button.

Tip: To avoid confusion, it may be useful to re-name your CSV column that contains key values to match the key attribute name of your map. In our example, you would rename *town* column to *name* column in the CSV, and your command would end with `keys=name,name`.

Do you remember aggregating address-level point records of hospitals into hospital counts per state discussed earlier in this chapter? Now is a good time to find that .CSV file and practice your merging skills.

Count points in polygons with Mapshaper

Mapshaper lets you count points in polygons, and record that number in polygon attributes using `-join` command.

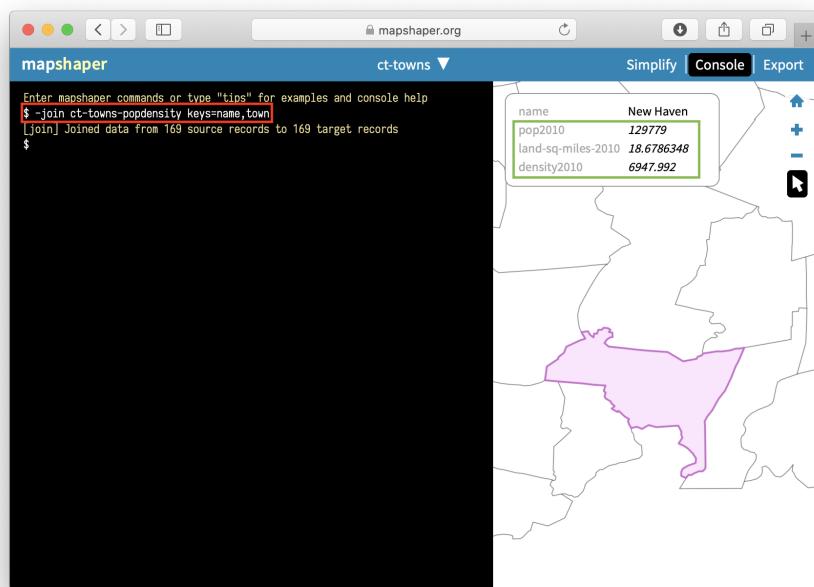


Figure 12.20: Mapshaper lets you join spatial and CSV files using common keys (for example, town names).

1. Import two geospatial files, one containing polygon boundaries (for example, US state boundaries), and another containing points that you want to aggregate (for example, hospitals in the US).
2. Make sure your polygons (not points) layer is active by selecting it from the dropdown menu.
3. In the *Console*, perform `-join` using a `count()` function, like this:

```
-join hospitals-points calc='hospitals = count()' fields=
```

This command tells Mapshaper to count points inside *hospitals-points* layer and record them as *hospitals* attribute of the polygons. The `fields=` part tells Mapshaper to not copy any fields from the points, because we are performing many-to-one matching (many hospitals per state, in our case).

4. Use the *Cursor > inspect attributes* tool to make sure polygons obtained a new field with the recorded count of points, like is shown in Figure 12.21.
5. Save the new file using *Export* button and choosing the desired output format.

More about joins

From the “Count points in polygons with Mapshaper” section of this chapter, you should recall that you do not need to specify *keys* if you want to perform join based on geographical locations between two geospatial layers (one being points, the other is polygons). If one of your files is a CSV, you need *keys*.

If you don’t have a CSV table that matches the columns in your boundary map data, you can easily create one. Upload the boundary map to Mapshaper, and export in CSV format. Open the downloaded file in any spreadsheet tool. To match data columns in the CSV spreadsheet, use the VLOOKUP function.

In real life, you will rarely have perfect files with one-to-one matches, so you might want to have more information about which features didn’t get matched so that you can fix your data. Mapshaper helps you keep track of data that is not properly joined or matched. For example, if the polygon map contains 169 features (one for each town in Connecticut), but the CSV table contains only 168 rows of data, Mapshaper will join all of those with matching keys, and then display this message:

```
[join] Joined data from 168 source records to 168 target records
[join] 1/169 target records received no data
[join] 1/169 source records could not be joined
```

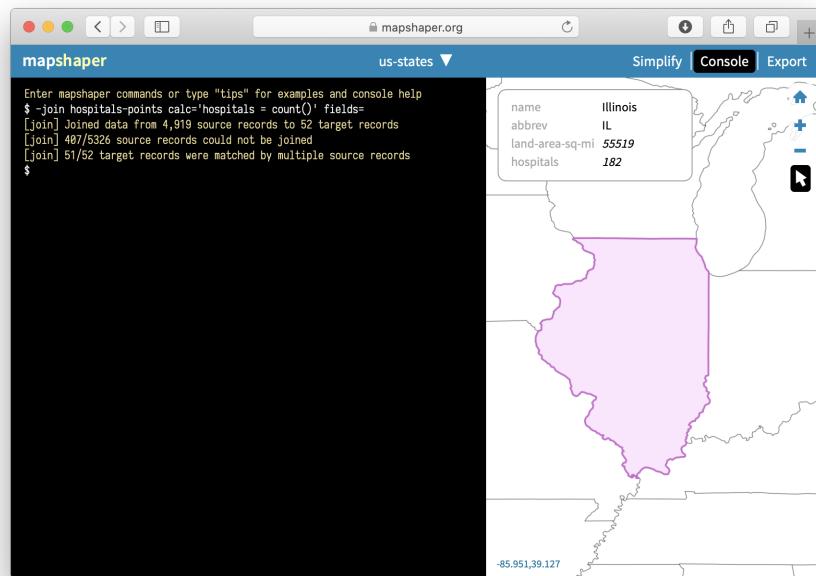


Figure 12.21: Mapshaper's `-join` can count points in polygons.

To get more details on which values were not joined, add `unjoined unmatched -info` flags to your join command, like this:

```
-join ct-towns-popdensity keys=name,town unjoined unmatched -info
```

The `unjoined` flag saves a copy of each unjoined record from the source table into another layer named *unjoined*. The `unmatched` flag saves a copy of each unmatched record from the target table to a new layer named *unmatched*. And the `-info` flag outputs some additional information about the joining procedure to the console.

Merge selected polygons with join and dissolve commands

In Mapshaper, you can merge selected polygons into larger “clusters” using `-join` and `-dissolve` commands.

Imagine that you are employed by the CT Department of Public Health, and your task is to divide 169 towns into 20 so-called Health Districts and produce a new geospatial file. By the way, health districts are a real thing in Connecticut.

You should begin by creating a *crosswalk* of towns and their health districts. Computer scientists and those working with data often use the term *crosswalk* to describe some kind of matching between two sets of data, such as zipcodes and towns where they are located. In our case, the crosswalk can be as simple as a two-column CSV list of a town and its district, each on a new line. Because your boss didn’t give you a list of towns in a spreadsheet format, but instead a GeoJSON file with town boundaries, let’s extract a list of towns from it.

1. Import `ct-towns.geojson` to Mapshaper using Quick import box.
2. You can use the *Cursor > inspect attributes* tool to see that each polygon has a `name` attribute with the name of the town.
3. Save attribute data as a CSV file using *Export* button. Open the file in any spreadsheet tool. You will see that your data is a one-column file with a `*name&` column that lists 169 towns.
4. Create a second column titled *merged* and copy-paste values from the first, `name` column. At this point your spreadsheet contains two columns with the same values.
5. Pick a few towns, for example *West Hartford* and *Bloomfield*, and assign “*Bloomfield-West Hartford*” to their *merged* column, like is shown in Figure 12.22. You may stop right here and move to the next step, or keep assigning district names to a few other neighboring towns.
6. Save this new file as `ct-towns-merged.csv`, and drag-and-drop it to Mapshaper on top of your `ct-towns` layer. Click *Import*.

	A	B	C
1	name	merged	
2	Bloomfield	Bloomfield-West Hartford	
3	West Hartford	Bloomfield-West Hartford	
4	Bethel	Bethel	
5	Bridgeport	Bridgeport	
6	Brookfield	Brookfield	
7	Danbury	Danbury	
8	Darien	Darien	
9	Easton	Easton	
10	Fairfield	Fairfield	
11	Greenwich	Greenwich	

Figure 12.22: Create a two-column crosswalk of towns and which districts they should be merged to.

7. This new CSV layer will be added as *ct-towns-merged* and will appear as a series of table cells. From the dropdown menu, select *ct-towns* to get back to your map.
8. Now you are ready to merge certain towns into districts according to your uploaded CSV file. Open the *Console*, and type:

```
-join ct-towns-merged keys=name, name
```

to join the CSV layer with the boundaries layer that you see on the screen, followed by

```
-dissolve merged
```

to dissolve polygons of towns according to the *merged* column of the CSV file.

In our example, only Bloomfield and West Hartford are dissolved into a combined “Bloomfield-West Hartford” regional health district (with the shared boundary between towns becoming grayed out), and all of the other polygons remain the same. Figure 12.23 shows the final result.

You can inspect attribute data of polygons using *Cursor > inspect attributes* tool, and save the resulting file using the *Export* button.

Learn more advanced MapShaper methods

There are many more commands within Mapshaper that are worth exploring if you are serious about GIS, such as changing projections, filtering features using

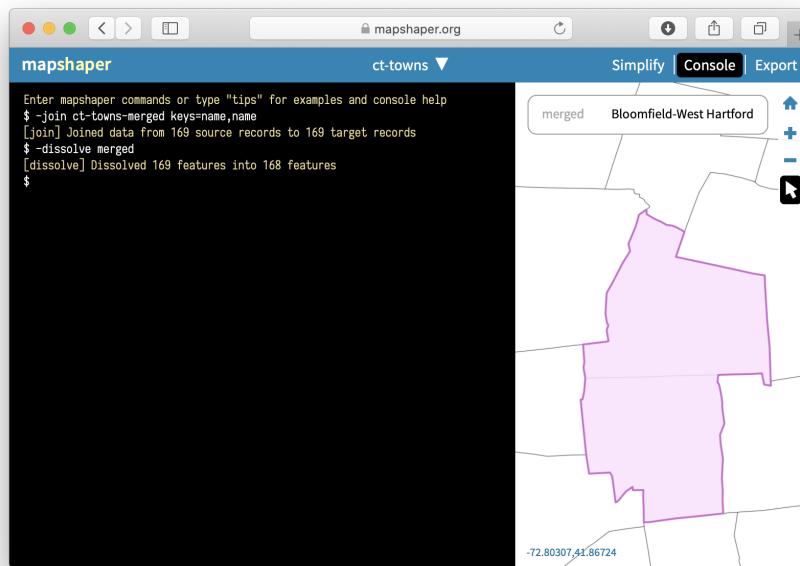


Figure 12.23: Merge polygons based on a predefined crosswalk.

JavaScript expressions, assigning colors to polygons based on values, and many more. Explore the Wiki of Mapshaper project on GitHub for more commands and examples.

Georectify a Digitized Map with MapWarper

TODO: write this section about using MapWarper, a tool created and hosted by Tim Waters, to upload and georectify a scanned map. This means to properly position a scanned map based on standard coordinates, so that you can place it as an overlay on an interactive map, such as Leaflet Storymaps with Google Sheets.

Anyone can upload and georectify a map on the developer's public site at <http://mapwarper.net>, and also see how it's used by organizations such as the New York Public Library at <http://maps.nypl.org>.

Warning: MapWarper is a wonderful open-source tool and platform, but service may be interrupted. As of July 2020, the site warns: "Ran out of disk space. Maps older than 2 years will need re-warping to work. Downtime will happen again."

Convert a Compressed KMZ file to KML format

In the previous sections, we looked at using Geojson.io and Mapshaper to convert geospatial files. However, not all file types can be converted with these tools. This chapter shows a particular example of a commonly requested .kmz <-> .kml pair conversion with Google Earth Pro.

KMZ is a compressed version of a KML file, and the easiest way to convert between the two is to use free Google Earth Pro (if you remember from *Convert to GeoJSON format* section, KML is a native format of Google Earth).

1. Download and install Google Earth Pro for desktop.
2. Double-click on any .kmz file to open it in Google Earth Pro. Alternatively, open Google Earth Pro first, and go to *File > Open* and choose your KMZ file.
3. Right-click (or control-click) on the KMZ layer under Places menu, and select *Save Place As...*, like is shown in Figure 12.24.
4. In the dropdown menu of *Save file...* window, choose KML format, like is shown in Figure 12.25.

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

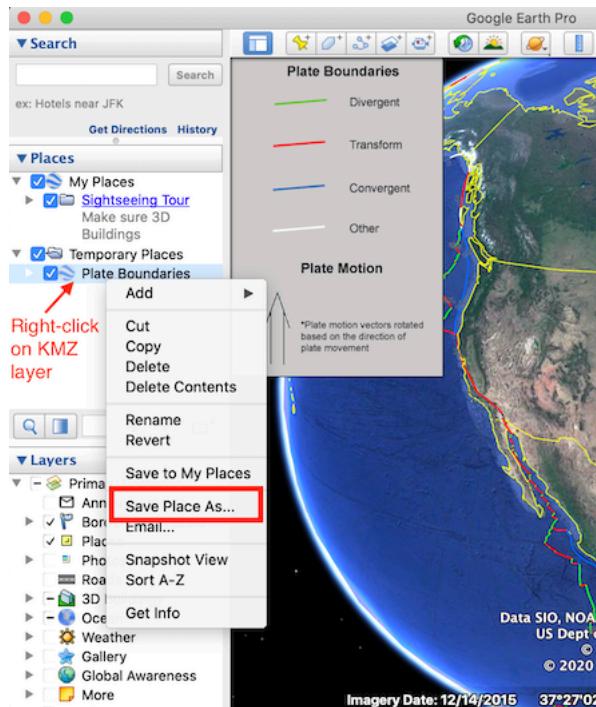


Figure 12.24: In Google Earth Pro, right-click the KMZ layer and choose *Save Place As...*.

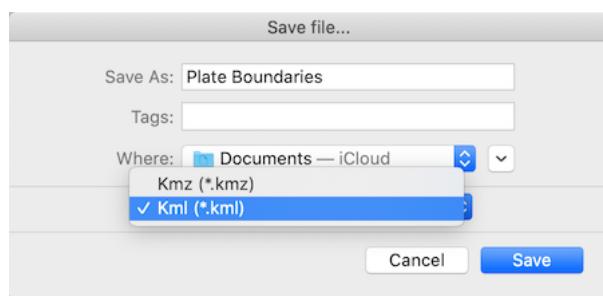


Figure 12.25: Save as KML, not KMZ.

Summary

In this chapter, you learned to use pivot tables to count addresses (points) by geographical area, such as states or cities (polygons). You learned that geospatial data can be vector or raster. The best file format to store, share, and use vector data is GeoJSON. You can use GeoJson.io to create, edit, or convert geospatial files inside your browser. Mapshaper is another online tool to convert, simplify, join or crop geospatial data.

In the following chapter, we will talk detecting bias in charts and maps. so that you become a better storyteller and a more critical reader.

Chapter 13

Detect Bias in Data Stories

TODO: Rewrite chapter

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

Sections in this chapter:

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

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.

TODO: convert all to code-chunk iframes

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 13.1: 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...

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

Chapter 14

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
- Integrate Story with Your Data
- Write Clearly about What You Visualize
- Acknowledge Limitations of the Data
- Credit Data Sources and Collaborators

Credit sources and collaborators on dataviz products and readme files

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

Appendix A

Fix Common Mistakes

TODO: Rewrite appendix to focus more broadly on “common mistakes” not just “code errors”

Creating your data visualizations through code templates hosted on GitHub has multiple advantages over drag-and-drop tools. Coding gives you more power to customize their appearance and interactive features, and to control where your data and products reside online. But there’s also a trade-off. Code can “break” and leave you staring at a blank screen. Sometimes problems happens through no fault of your own, such as when a “code dependency” to an online background map or code library is unexpectedly interrupted. But more often it seems that problems arise because we make simple mistakes that break our own code. Whatever the cause, one big drawback of working with code is that you’re also responsible for fixing it.

We designed this section as a guide to help new coders diagnose and solve common errors when working with code templates on GitHub. We understand the feeling you experience when a simple typo—such as a misplaced semicolon (;)—makes your data visualization disappear from the screen. Finding the source of the problem can be very frustrating. But breaking your code—and figuring out how to fix it—also can be a great way to learn, because trial-and-error on a computer often provides immediate feedback that supports the learning process and develops our thinking.

TODO: Reorganize contents, perhaps using this outline?

- Problems with Mac computers
- Problems with data tables
- Problems with iframes (since this chapter appears before code templates)
- Problems with GitHub forking and hosting
- Problems with code templates

Problems with Mac computers: cannot see filename extension

Several tools in this book will not work properly if your computer does not display the filename extensions, meaning the abbreviated file format that appears after the period, such as `data.csv` or `map.geojson`. The Mac computer operating system hides these by default, so you need to turn them on by going to `Finder > Preferences > Advanced`, and check the box to *Show all filename extensions*, as shown in Figure A.1.

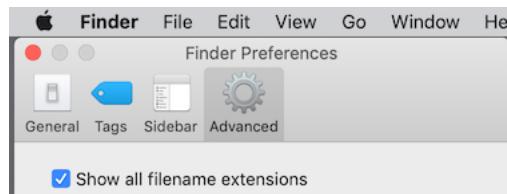


Figure A.1: On a Mac, go to *Finder* then *Preferences* then *Advanced* and check the box to *Show all filename extensions*.

Problems with data tables

Avoid typing blank spaces after column headers—or any spreadsheet entries—since some data visualization tools will not match them with headers lacking a blank character.

	A	B	C	D
1	name	all2015	healthcare2015	manufactur
2	Andover	189	36	67
3	Avon	2536	1866	391
4	Berlin	4907	1159	2822
5	Bloomfield			3688
6	Bolton	488	60	269

Problems with iframes

My iframe does not appear in my web page

- Go back to the Embed tutorials in this book to double-check the directions
- Items listed in your iframe (such as the URL, width, or height) should be enclosed inside straight quotation marks (single or double)
 - BROKEN iframe (missing quotation marks for width and height)

```
<iframe src="https://handsondataviz.github.io/leaflet-map-simple"
width=90% height=350></iframe>
```

- FIXED iframe (with correct quotation marks)

```
<iframe src="https://handsondataviz.github.io/leaflet-map-simple"
width="90%" height="350"></iframe>
```

- Use only `https` (the extra ‘s’ means ‘secure’), not `http`. Some web browsers will block content if it mixes http and https resources, and some code templates in this book require https.

**<iframe src="http://ik
Change to https**

Correct

<iframe src="https://

Figure A.2: Screenshot: Replace http with https

- Use only straight quotes, not curly quotes. Avoid pasting text from a word-processor into GitHub, which can accidentally carry over curly quotes. Typing directly into the GitHub editor will create straight quotes.

TODO: Test one way to fix GitHub errors by going into the commits and going back to a previous version of the code. Is this possible in the web version?

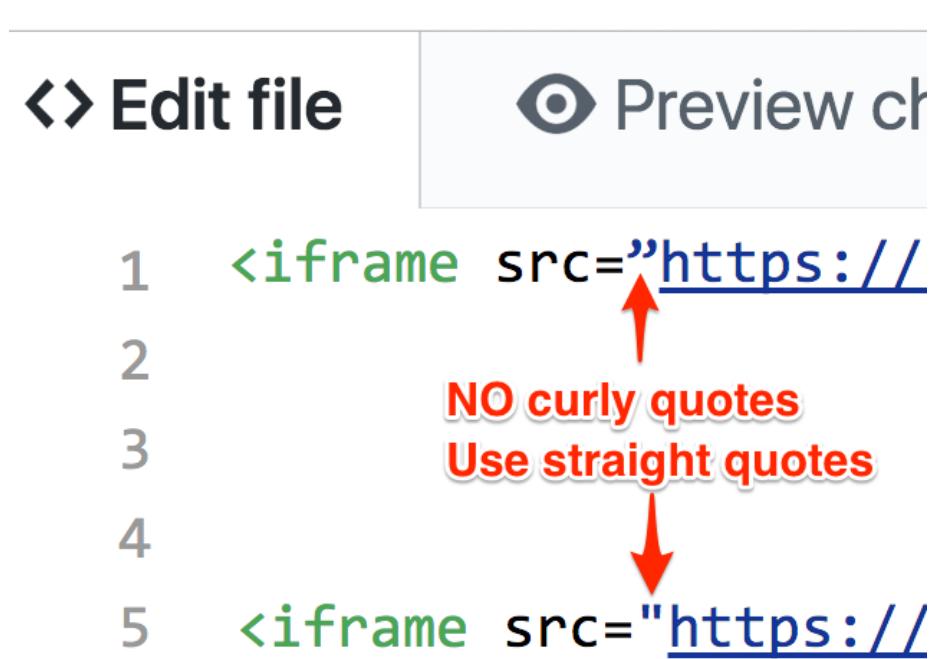


Figure A.3: Screenshot: Curly quotes versus straight quotes

Safely Delete your GitHub Repo and Start Over

If you need to delete your GitHub repo and start over, here's a simple way to safely save your work:

- Go to the top-level of your GitHub repository, similar to <https://github.com/USERNAME/REPOSITORY>
- Click the green “Clone or Download” button, and select Download Zip to receive a compressed folder of your repo contents on your computer.
- In your GitHub repo, click on Settings (upper-right area) and scroll down to Delete This Repository.
- To prevent accidental deletions, GitHub requires you to type in the REPOSITORY name.
- Now you can start over in one of these ways:
 - If you wish to Create a Simple Web Page with GitHub Pages, follow that tutorial again.
 - OR
 - Fork another copy of the original GitHub repository to your account. After you create your copy, if you wish to add selected files that you previously downloaded to your computer, follow directions to Upload Code with GitHub in the second half of this tutorial in this book

Problems with Creating a Simple Web Page with GitHub Pages

If you followed the Create a Simple Web Page with GitHub Pages tutorial, it should have created two web links (or URLs):

- your code repository, in this format: <https://github.com/USERNAME/REPOSITORY>
- your published web page, in this format: <https://USERNAME.github.io/REPOSITORY>

Be sure to insert your GitHub username, and your GitHub repository name, in the general formats above.

These URLs are NOT case-sensitive, which means that <https://github.com/USERNAME> and <https://gitub.com/username> point to the same location.

My simple GitHub web page does not appear

- Make sure that you are pointing to the correct URL for your published web page, in the format shown above.
- Be patient. During busy periods on GitHub, it may take up to 1 minute for new content to appear in your browser.

- **MOVE UP** If your map does *not* appear right away, wait up to 30 seconds for GitHub Pages to finish processing your edits. Then give your browser a “hard refresh” to bypass any saved content in your cache and re-download the entire web page from the server, using one of these key combinations:
 - Ctrl + F5 (most browsers for Windows or Linux)
 - Command + Shift + R (Chrome or Firefox for Mac)
 - Shift + Reload button toolbar (Safari for Mac)
 - Ctrl + Shift + Backspace (on Chromebook)
- Test the link to your published web page in a different browser. If you normally use Chrome, try Firefox.
- On rare occasions, the GitHub service or GitHub Pages feature may be down. Check <https://status.github.com>.

My simple GitHub web page does not display my iframe

- If you followed the Create a Simple Web Page with GitHub Pages tutorial and inserted an iframe in the README.md file, it will appear in your published web page, under these conditions:
 - Ideally, your README.md should be the ONLY file in this GitHub repository
 - Any other files in your repo (such as index.html, default.html, or index.md) will block the iframe HTML code in your README.md from being published on the web. If you accidentally selected a GitHub Pages Theme, you need to delete any extra files it created: click each file, select trash can to delete it, and commit changes.

Problems with Leaflet Maps with Google Sheets template

My map does not appear

- 1) Confirm that you have completed all of the key steps in the Leaflet Maps with Google Sheets tutorial in this book, especially these:
 - Sign in to Google and File > Make a Copy of the Google Sheet to your Google Drive.
 - File > Publish your Google Sheet (Jack often forgets this key step!)
 - Copy your Google Sheet web address from top of your browser (usually ends with ...XYZ/edit#gid=0) and paste into your `google-doc-url.js` file in your GitHub repo. Do NOT copy the *Published* web address (which usually ends with ...XYZ/pubhtml)

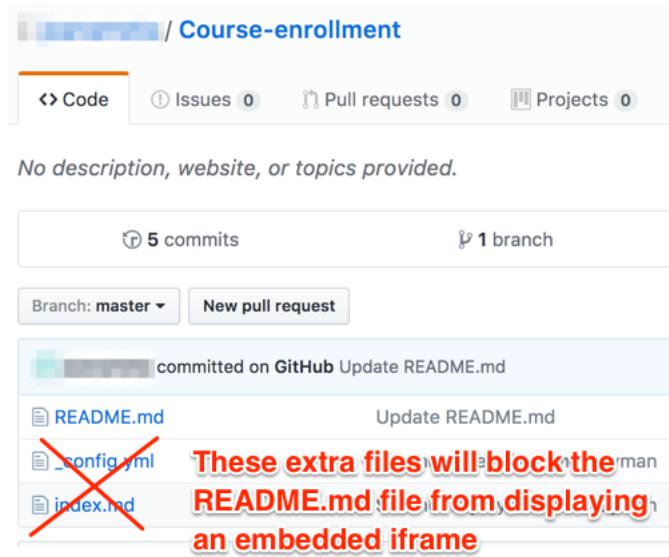


Figure A.4: Screenshot: Extra files in GitHub repo will block iframe in your README

- When you paste your Google Sheet web address into `google-doc-url.js`, be careful not to erase single-quote marks or semicolon
 - Go to your live map link, which should follow this format: `https://USERNAME.github.io/REPOSITORY`, refresh the browser, and wait at least 30 seconds.
- 2) Check your Google Sheet for errors:
 - Do NOT rename column headers (in row 1) of any sheet, because the Leaflet Map code looks for these exact words.
 - Do NOT rename row labels (in column A) of any sheet, due to the same reason above.
 - In your Points tab, DO NOT leave any blank rows
 - 3) Confirm on GitHub Status (<https://status.github.com/>) that all systems are operational.
 - 4) If you cannot find the problem, go to the top of this page to Safely Delete Your GitHub Repo and Start Over. Also, make a new copy of the Google Sheet template, give it a new name, and copy data from your old sheet using File > Paste Special > Values Only.

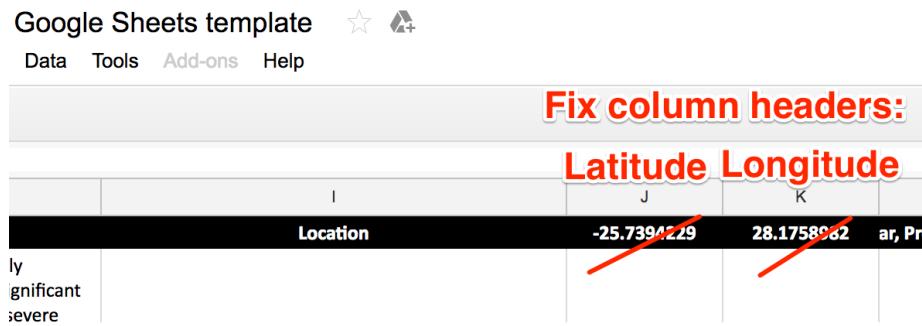


Figure A.5: Screenshot: User accidentally renamed column headers in the Points tab

Problems with Chart.js code templates

Chart displays old data If you upload new data to your Chart.js code template on GitHub Pages, and it does not appear in your browser after refreshing and waiting up to one minute, then GitHub Pages is probably not the cause of the problem. Instead, some browsers continue to show “old” Chart.js in the web cache. The simplest solution is to File > Quit your browser and re-open the link to your Chart.js

TODO: Our Chart.js templates appear blank (just text, no chart) when viewed in the local browser. But Leaflet maps appear mostly or partially complete. Why is this, and how should we inform readers about this? Discuss with Ilya

Solve Problems with Browser Developer Tools

Peek inside any website and view the web code under the hood with the browser developer tools.

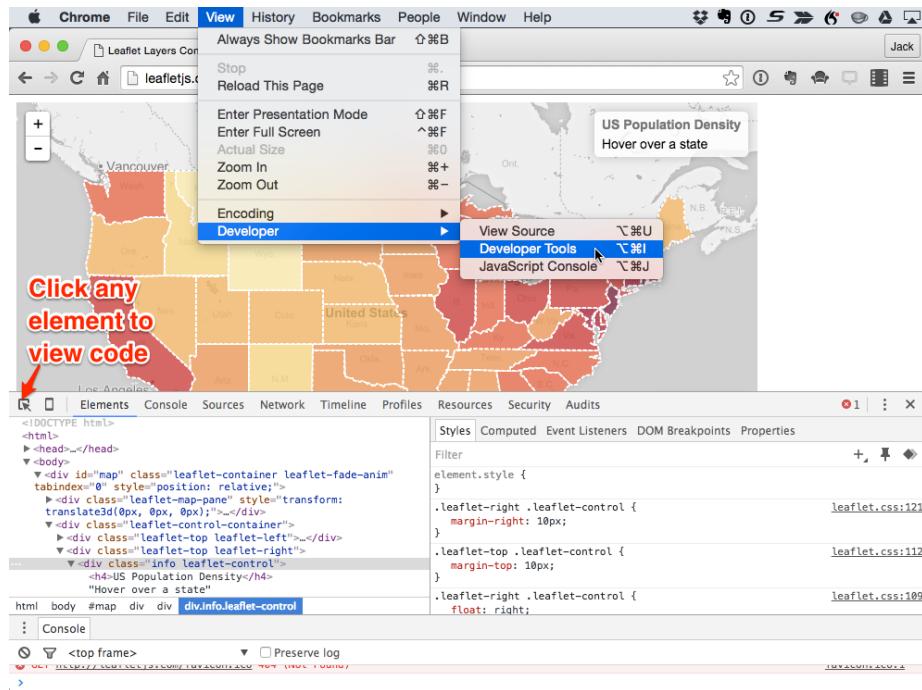
In Chrome for Mac, go to View > Developer > Developer Tools

The screenshot shows a Microsoft Excel spreadsheet with the title "Leaflet Maps with Google Sheets" in the top bar. The menu bar includes File, Edit, View, Insert, Format, and a currency symbol. Below the menu is a toolbar with icons for print, refresh, and other functions, along with a currency symbol and a date/time field.

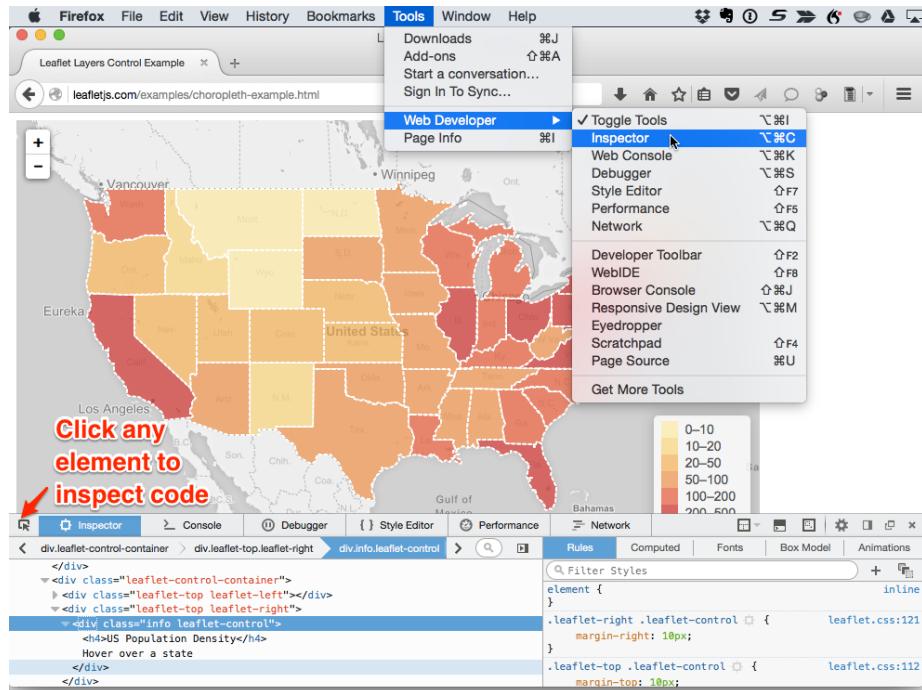
The spreadsheet contains a table with 12 rows. The first row has a green header labeled "Setting". Rows 2 through 9 have a light purple background, while rows 10 through 12 have a light green background. Row 10 is labeled "Map Settings". The data in the table is as follows:

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

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



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



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

Dougherty, Jack, Jeffrey Harrelson, Laura Maloney, Drew Murphy, Russell Smith, Michael Snow, and Diane Zannoni. "School Choice in Suburbia: Test Scores, Race, and Housing Markets." *American Journal of Education* 115, no. 4 (August 2009): 523–48. http://digitalrepository.trincoll.edu/cssp_papers/1.

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