Creating GeoMaps in D3

Overview

Goal:
1. To Create California Population Density Map from the very beginning.
2. To Create any US State Population Density Map from the very beginning.

Additional Subgoals are:
1. Learn how to simplify Shapefiles (discussed in Scott Murray's Book)
2. Learn how to Convert from Shapefiles to GeoJson/Topojson files
3. Learn how to view/modify Shapefiles, GeoJson, Topojson Files

In order to achieve these goals and subgoals, you will need to install several software on your laptop irrespective of the platform: Mac, Windows, Linux. The main softwares are: Node, npm, shapefile, ndjson-cli, and d3-geo-projection (These are discussed well in Command0line cartography 4-part tutorial by Mike Bostock). Additional recommended softwares are Xcode and Homebrew (for Mac only), Mapshaper, and GDAL.

Advanced Goal (optional):
1. Ability to find and convert shapefiles at high resolution to Topojson files of any region in the world, and
2. Ability to find socio-economic data related to above geography and join the data with the above geography for D3 Visualization

Reference: Command-line cartography Four Part Tutorial by Mike Bostock
https://medium.com/@mbostock/command-line-cartography-part-1-897aa8f8ca2c

Section 1: Installing Software (and verifying that they work)
It has been reported that these methods are somewhat tricky, the user experiences are not consistent with each other (Scott Murray remarks, "seriously why this isn't working omg please work this time or i will freak out", page 319) and sometimes require workaround to ensure that these softwares work properly. If you struggle installing these softwares, please document your challenges and how you overcame them. If your writeup is insightful, then you will be rewarded with bonus points.

Apple/Mac Platform
1. Xcode (https://developer.apple.com/xcode/download/)
download and install accepting the license agreement otherwise later program homebrew will complain. Xcode is an integrated development environment for a suite of software development tools developed by Apple for developing software. This tool is needed in order to work with homebrew.

2. Homebrew (Open Terminal on Mac OSX (Applications -> Utilities -> Terminal)
Then download Homebrew Package from Mac OS X: http://brew.sh/
by pasting the command provided on the website.
To confirm installation, type brew in Terminal, this should return a list of brew commands.

Homebrew is a free and open source software package management system and simplifies the installation of software on Apple's OS X operating system.

3. (optional) GDAL (Type the following command into Terminal: brew install gdal)
After this installation, the terminal displays the version of GDAL installed.

GDAL is a computer software library for reading, writing raster and vector geospatial data formats. This library is needed to work with shapefiles. GDAL supports ogr2ogr command that is extremely helpful.

To confirm that the installations are happening without problems,
Type which ogr2ogr. This should return /usr/local/bin/ogr2ogr

4. Install npm (node package manager)
Go to the following link, then download and install the node.js package.
https://nodejs.org/en

Type the following to check which versions of node and npm do you have:
node -v
npm –v
If you do not have the current versions, you may type the following to get the latest versions.
sudo npm install npm -g

5. shapefile (this package includes command: shp2json)
Type the following command into Terminal:
npm install -g shapefile
If this does not work and it is a permission problem, try,
sudo npm install -g shapefile

To confirm that the installation is happening without problems,
Type which shp2json. This should return /usr/local/bin/shp2json

6. ndjson-cli (this package includes command: ndjson-split, ndjson-map, ndjson-cat, ndjson-join, ndjson-redue)
Type the following command into Terminal:
npm install -g ndjson-cli
If this does not work and it is a permission problem, try,
sudo npm install -g ndjson-cli

7. d3-geo-projection (this package includes command: geoproject)
Type the following command into Terminal:
npm install -g d3-geo-projection
If this does not work and it is a permission problem, try,
sudo npm install -g d3-geo-projection

After installations of these software, you are ready to roll!

Windows Platform
You don't need Xcode or homebrew (those are specific for Apple/Mac platforms). You can follow the instructions from Step 4 above as long as you use either GitBash, Cygwin, Bash or Ubuntu on Windows, or some other bash emulator. Be sure to run commands with a -g when specified and install things globally. If running on Bash on Ubuntu on Windows, the folder where your files are stored is C:\Users\$username$\AppData\Local\lxss\home
where $username$ is your Windows username. This is where you can add/remove files if needed.

Linux-Based Machines
(Following has been tried successfully on Ubuntu 16.04)
1. Install Node and npm
Go to the following link, then follow the instruction for your Linux OS to install Node and npm.
For example, if you are using Ubuntu:
sudo apt-get install curl
curl -sL https://deb.nodesource.com/setup_8.x | sudo -E bash -
sudo apt-get install -y nodejs

Type the following to check which versions of node and npm do you have:
node -v
npm –v
If you do not have the current versions, you may type the following to get the latest versions.
sudo npm install npm -g

2. shapefile (this package includes command: shp2json)
Type the following command into Terminal:
npm install -g shapefile
If this does not work and it is a permission problem, try,
sudo npm install -g shapefile

To confirm that the installation is happening without problems,
Type which shp2json. This should return /usr/bin/shp2json

3. ndjson-cli (this package includes command: ndjson-split, ndjson-map, ndjson-cat, ndjson-join, ndjson-redue)
Type the following command into Terminal:
npm install -g ndjson-cli
If this does not work and it is a permission problem, try,
sudo npm install -g ndjson-cli

4. d3-geo-projection (this package includes command: geoproject)
Type the following command into Terminal:
npm install -g d3-geo-projection
If this does not work and it is a permission problem, try,
sudo npm install -g d3-geo-projection

Section 2: Creating California (and other US States) Population Density Visualization
(Creating Topojson files from Shapefiles and Associate Geocoded Data)

We will adopt a three-step process.

1. First, we will use US Census Bureau website to obtain a shapefile and convert it into a topojson file using command line.

2. Second, we will use US Census Bureau's developer API to obtain geocoded population density data that can be joined into the topojson file for they share a common geocoded ID.

3. Join the geocoded data into the topojson file using command line.

We will illustrate this entire 3-step process by downloading the California **Census Tract** shapefile, convert it into topojson, and then join population data for California Census Tracts.

Step 1 (California Census-Tract, County, and State Boundary Data):

1. Go to the following US Census Website titled "Cartographic Boundary Shapefiles":

https://www.census.gov/geo/maps-data/data/tiger-cart-boundary.html

2. Click on "Census Tract" under State-based files.

3. Select the year "2014", and then select state "California" and the file, cb_2014_06_tract_500k.zip, will be prompt to download.

4. Move the zip file into a directory titled "CaliforniaMapping". Unzip the file. You will see see several files including one with .shp extension. This is the shape file.

5. To convert the shp file into topojson file, open the Terminal (Applications -> Utilities -> Terminal). Redirect the working directory to "CaliforniaMapping". Now type the following command:

```
shp2json cb_2014_06_tract_500k.shp -o ca.json
```

This should create ca.json file in the directory. This topojson file is ready for rendering. You can preview it on "mapshaper.org".

5. To avoid expensive trigonometric operations at runtime and therefore the resulting GeoJSON renders much faster, we should apply a geographic projection. Type the following in terminal:

```
geoproject 'd3.geoConicEqualArea().parallels([34, 40.5]).rotate([120, 0]).fitSize([960, 960], d)' < ca.json > ca-albers.json
```

This d3.geoConicEqualArea is California Albers projection, which is appropriate for showing California. This should create ca-albers.json file in the directory.
For other states, you could try d3-stateplane or search spatialreference.org to find out what projection to use (discussed later).

Step 2 (California Population Density Data):

Download the population data from US community survey for 2014 using Census Bureau's developer API. Use any tool (your browser, curl, wget) to download the json file using following URL:

http://api.census.gov/data/2014/acs5?get=B01003_001E&for=tract:*&in=state:06

Save the file with name "cb_2014_06_tract_B01003.json".

Step 3 (Join Geography Data with Population Density Data):

1. To convert a GeoJSON feature collection to a newline-delimited stream of GeoJSON features (NDJSON stream), use ndjson-split:

```
ndjson-split 'd.features' \
  < ca-albers.json \
  > ca-albers.ndjson
```

2. To set each feature's id using ndjson-map:

```
ndjson-map 'd.id = d.properties.GEOID.slice(2), d' \
  < ca-albers.ndjson \
  > ca-albers-id.ndjson
```

This will create a new key named "id" for each feature, and the value is the GEOID without first 2 digits (which stands for the state ID and therefore redundant). This id value shall match the id in the geocoded data (population survey data).

3. The survey data file “cb_2014_06_tract_B01003.json” is a JSON array. To convert it to an NDJSON stream, type the following into terminal:

```
ndjson-cat cb_2014_06_tract_B01003.json \
  | ndjson-split 'd.slice(1)' \
  | ndjson-map '{id: d[2] + d[3], B01003: +d[0]}' \
  > cb_2014_06_tract_B01003.ndjson
```

The command ndjson-cat is to remove the newlines. The command ndjson-split is to separate the array into multiple lines. And the command ndjson-map is to reformat each line as an object.

4. Join the population data to the geometry data using ndjson-join. Type the following into terminal:

```
ndjson-join 'd.id' \
  ca-albers-id.ndjson \
  cb_2014_06_tract_B01003.ndjson \
  > ca-albers-join.ndjson
```

5. Compute the only useful data, the population density, using population divided by land area. And remove the additional properties we no longer need. Type the following into terminal:

```
ndjson-map 'd[0].properties = {density: Math.floor(d[1].B01003 / d[0].properties.ALAND
* 2589975.2356)}, d[0]' \
  < ca-albers-join.ndjson \
  > ca-albers-density.ndjson
```

6. Finally, convert back to GeoJSON, use ndjson-reduce and ndjson-map. Type the following into terminal:

```
ndjson-reduce \
  < ca-albers-density.ndjson \
  | ndjson-map '{type: "FeatureCollection", features: d}' \
  > ca-albers-density.json
```

This would create the “ca-albers-density.json” which is a topojson file with geocoded data included. We could solely use this file to render visualization in the future. Try it out on “mapshaper.org” or your D3 project!

Now, that you have the joined geography-population density data, you can use Mike Bostock’s code to visualize California population density data:

https://bl.ocks.org/mbostock/5562380

How to use geocoded data along with NDJSON files to create visualization in D3 (using only command line)?

Given that we have already produced the "ca-albers-density.ndjson" file at previous tutorial (Step3 #5), we can use the density values to fill each state with appropriate color. For example, in a choropleth, we allocate darker color to state with higher density value. (See the Command-line Cartography Part 2 for full details.)

You would need to install the "d3" command line package to proceed. The d3 package contains several predefined color schemes that helps reflect the density values with proper color. Type the following command into Terminal:

```
npm install -g d3
```

If this does not work and it is a permission problem, try,

```
sudo npm install -g d3
```

Next, use ndjson-map to define a fill property in the NDJSON, using a sequential scale with the Viridis color scheme:

```
ndjson-map -r d3 \
  '(d.properties.fill = d3.scaleSequential(d3.interpolateViridis).domain([0,
  4000])(d.properties.density), d)' \
 < ca-albers-density.ndjson \
 > ca-albers-color.ndjson
```

Next, convert the NDJSON to SVG using geo2svg:

```
geo2svg -n --stroke none -p 1 -w 960 -h 960 \
 < ca-albers-color.ndjson \
 > ca-albers-color.svg
```

Now, you can use any browser to open the SVG file to see the colored map.

Instructions on how to project to a US State (other than California) Go to the webpage SpatialReference.org. Search your state and try to find the entire state. If you can't try to find if there's a central portion (e.g. New York Central). If you can't find that either pick either East and West and click that one. Then click the Proj4 link on the page.

You should get the result
+proj=aea +lat_1=**val1** +lat_2=**val2** +lat_0=**val3** +lon_0=**val4** +x_0=0 +y_0=-4000000 +ellps=clrk66 +datum=NAD27 +units=m +no_defs.

You must edit the following command with the vals above
geoproject 'd3.geoConicEqualArea()**.parallels([*val1*, *val2*]).rotate([-*val4*, -*val3*])**.fitSize([960, 960], d)' < ca.json > ca-albers.json.

That should be the projection needed for your file.
Note: Some of the values (e.g. lat_1 and lat_2 may not be available, in case that occurs, just ignore that field.

For example, Illinois would look like the following:
Proj4 for Illinois West
+proj=tmerc +lat_0=**36.66666666666666** +lon_0=**-90.16666666666667** +k=0.999941177
+x_0=699999.9999898402 +y_0=0 +ellps=GRS80 +datum=NAD83
+to_meter=0.3048006096012192 +no_defs

This becomes:
geoproject 'd3.geoTransverseMercator().rotate([**90.166667, -36.666667**]).fitSize([960, 960], d)' < il.json > il-epsg3436.json

Since the Proj4 does not have lat_1 or lat_2, the parallels are not needed so we only rotate.

References

1. Topojson command Line Reference:
   https://github.com/mbostock/topojson/wiki/Command-Line-Reference
2. US Atlas Repository by Mike Bostock: https://github.com/mbostock/us-atlas
   Makefile on this website is quite useful
3. Generate world atlas TopoJSON files from Natural Earth data:
   https://github.com/mbostock/world-atlas

Section 3: Tools for Viewing/Mainpulating Shapefiles/GeoJson/Topojson files

Resolution and Simplification
Resolution of shape files may be available at different resolutions. You may want data at very high resolutions for some applications. For some other applications, one may need to simplify the data. Topojson command may be used to simplify the resolution. This simplification algorithm is illustrated at the following website: https://bost.ocks.org/mike/simplify/
Scott Murray discusses various methods of simplifying shape files.

Viewing GeoJson, Topojson, or Shape files: Mapshaper
Mapshaper (http://mapshapper.org) by Matt Bloch. Use this software to view json files.
A tool for reading, writing, and viewing shapefiles, geoJson, and Topojson files. Just drag and drop your files onto this website.

GDAL (advanced):

GDAL is a computer software library for reading, writing raster and vector geospatial data formats. This library is needed to work with shapefiles. GDAL supports ogr2ogr command that is extremely helpful.

Section 4: Creating New Geomaps

In order to create a GeoMap Visualization in D3, you need two items:
1. Polygonal Boundaries of Geography in Topojson (or Geojson format)
2. Geocoded Data (that you want to visualize) that is data associated with some underlying geography
In this handout, we address the two main questions:
Q1: Where do we find or how do we create polygonal boundaries (json files)?
Q2: Where do we find geocoded data?

We revisit some examples of Geomapping in D3.
Example 1: **Population density** data for all the 7,049 census tracts of **California** with CensusTract ID, along with polygonal boundaries of all the census tracts of California in json format with CensusTract ID
https://bl.ocks.org/mbostock/5562380
(uses ca.json file that was created from shapefiles and ogr2ogr command in GDAL to create California census tracts map with boundaries of counties as well as census tracts. Also describes how geocoded data was extracted from US Census Bureau)

Example 2: Unemoplyment rates in more than 3,000 counties to create a **Chloropleth map of US** with geocoded counties, and polygonal boundaries of the counties in json format with County ID
http://bl.ocks.org/mbostock/4060606 (uses us-10m.json)

How to obtain json files

There are two most likely ways of obtaining json files:

1. **Publicly Available**: The easiest way to get your hands on json files is if somebody else has created them for you and has made them readily available to you.

We have the following **Geojson** files:
1. us-states.json (provided by Scott Muarry with his sample code)
[includes Alaska, Hawaii, and Puerto Rico] works perfectly only 3 line intersections
2. oceans.json (provided by Scott Murray with his sample code) works perfectly
3. countries.geo.json (worls map with 180 polygons; looks great); also has individual boundaries of a large number of countries
https://github.com/johan/world.geo.json
US/World GeoJSON data by Johan
This website has a directory "countries".
Inside this directory, there is data for US counties by state.
It also has data for many countries.
The data is "dubious". The website states that, drag these data sets to
http://bl.ocks.org/1431429 and paint the globe

We have the following **Topojson** files:

1. **USMainLandCounties.json** (works perfectly)
2. https://github.com/mbostock/topojson
Examples directory contains following three TopoJSON files
us-10m.json (has counties, states, and land including Alaska, Hawaii, and Puerto Rico) with only 82 line intersections; usable
world-110m.json (177 polygons with 158 line interscetions)
world-50m.json (241 polygons with 369 line intersections)
3. http://bost.ocks.org/mike/map/
uk.json (provided by Mike Bostock: looks great)
4. Following city files
SF (Neighborhoods with 487 line intersections)
LA (Neighborhoods with 1545 line intersections)
NY (Neighborhoods with 270 line interscetions)
5. We created the following jsons following Option 1 (described below)
usa-nation (has 2232 line intersections; ugly at edges)
usa-states (has 2328 line intersections: ugly at some boundaries)
usa-counties (has 3317 line intersections: ugly)
california-state.json (has 2 line intersection, only CA state boundary; useless by itself)
california-counties.json (has 11 line intersections; barely visible; usable)
(above 2 files for 13 states including California, Illinois, Washington DC)
6. **ca.json** (used in California Population Density: has 58 CA counties and 8000+ census tracts)
7. India.json, asean.json, middleeast.json, brics.json (individual countries)
8. India.json with states boundaries

2. **Create from Shapefiles**: Most often, polygonal boundaries of underlying geography is available as "shapefiles" from various agencies, organizations such as US Census Bureau, Cities, Government websites etc. One then needs to learn how to convert the shape files to json files. In addition, one may need to **Extract Subdata from Shapefiles**: often the shape files is a superset of data, for example, polygonal boundaries of all the 66,438 census tracts of the US and one may only be interested in California counties.

Where to find shapefiles?
US Census Bureau
https://www.census.gov/geo/maps-data/data/tiger-cart-boundary.html
https://www.census.gov/geo/maps-data/data/tiger-line.html
https://www.census.gov/geo/maps-data/
https://www.census.gov/geo/maps-data/tiger.html

Natural Earth
http://www.naturalearthdata.com

City Census Files
This website has all the LA Data http://boundaries.latimes.com/sets/
This website has all the SF Data https://data.sfgov.org/data
This website has the NY Data https://data.cityofnewyork.us/data

We have created TopoJSON files from the data from the websites above of city neighborhoods. **Note** The red dots can be ignored, they wont disrupt the visual.

Zillow websitehttp://www.zillow.com/howto/api/neighborhood-boundaries.htm
has boundaries of all the major cities of each state. However, the shapefiles Zillow provides a little different and might need some editing.

India
https://github.com/datameet/maps
The Indian states are divided into districts, whose shapefiles can be obtained from https://github.com/datameet/maps/tree/master/Districts/Census_2011
District-wise population data and area can be obtained fromhttps://en.wikipedia.org/wiki/List_of_districts_in_India
Other sources are http://opengovernanceindia.org/leapaf/district-wise-population-in-india-2011-census
Also some data is available from https://data.gov.in/keywords/district-0 (not sure how good this data is because shapefiles are different than those in population data).
You may be able to use wikipedia data (the page has references which can be used to download data), but some of these references do not have good data. Rather, web scraping could be done to get data from wikipedia page.

Another way to obtain US Shapefiles

(Creating Shapefiles after Cloning US-Atlas Repository by Bostock)

Follow Mike Bostock's us-atlas github repository, clone the repository and then follow makefile commands to generate shapefiles for us states including census tracts and many other types. Get geo-coded data from census.gov and use topojson commands to join the data with shape files and generate json files.

If you already have git installed, the type the following commands one at a time in Terminal to clone US-Atlas repository by Mike Bostock. We will use this repository to extract California shape files.

1. git clone https://github.com/mbostock/us-atlas.git
2. cd us-atlas
3. npm install

This last command should create a directory called node_modules with 3 sub-directories within it titled "d3", "topojson" and "shapefile". If this last command does not work, visit the following github to try out fixes or google to figure out if you can overcome this problem. This problem is related to install;lted of npm package and node.js installation. If those installations went smoothly, you are unlikely to encounter any error.
https://github.com/mbostock/us-atlas

Creating California ShapeFile

To create a California TopoJSON file, we follow steps as described in the Makefile in Mike Bostock's GitHub Repository https://github.com/mbostock/us-atlas.git

1. Type make shp/ca/states.shp

The above command should have created a directory called shp which has two subdirectories within it titled "ca" and "us".

2. Type make shp/ca/counties.shp (Note * 'ca' can be replaced with any state id). This creates a map with CA counties within "ca" directory titled "counties.shp".

Remarks

1. **og2ogr command** was used by Mike Bostock
to reproject the shapefiles to the California Albers projection:

ogr2ogr \ -f 'ESRI Shapefile' \ -t_srs 'EPSG:3310' \ tracts.shp \ shp/ca/tracts.shp
ogr2ogr command is available in GDAL
Ogr2ogr: http://www.gdal.org/ogr2ogr.html

2. Tool to covert csv files to json:
http://www.csvjson.com/csv2json

3. To set up a web counter, follow the tutorial on
https://support.google.com/analytics#topic=3544906

Section 5: Additional Examples of GeoSpatial mapping using D3 V4

Examples on the Internet

2016 Election results by county with votes differences between two parties
http://bl.ocks.org/sarubenfeld/9390b66fc33eecf73b1c44d06ab0e0bf

This map uses 8 colors to classify 8 different tracts for all tracts in bay area based on the study of urban displacement.
http://www.urbandisplacement.org/map/sf

A U.S. map showing gun traces of Chicago across the country
https://bl.ocks.org/sarubenfeld/724f0b986364e780a3333c9082f6585a

Create a night view of Japan by re-coloring the geo map.
https://bl.ocks.org/sarubenfeld/a45ec922b0afc039568fcceb16c285bc

This is a flat world map with geo satellite positions added on.
http://bl.ocks.org/rob4acre/26138ad26ed2a16a0fc326a56252b630

This is a world map with each country colored by its population.
http://bl.ocks.org/micahstubbs/8e15870eb432a21f0bc4d3d527b2d14f

This is a U.S. map, with states colored by whether the author has lived in, and cities marked and sized by how long the author had stayed in.
http://bl.ocks.org/dbetebenner/9b537c237b6ac96b107f259c98e62b81

This world map can be dragged and viewed differently based on Furuti projection.
https://bl.ocks.org/Fil/14ddff5e46b6fe9341dae91c3c83304b

Roads in San Francisco are shown line by line in purple.
http://bl.ocks.org/sarubenfeld/58c7b480d6dd72b5f64943db6cf03e2b

23 Examples from previous classes on D3 at UCSC

San Francisco vs LA
https://sureshlodha.github.io/CMPS165_Winter15_FinalProjects/SFvsLA/

California Quality of Life: Education, Health, and Poverty
https://sureshlodha.github.io/CMPS165_Spring2016_FinalProjects/projects/CALife/index.html

Renewable Water Resources in the World
https://sureshlodha.github.io/CMPS165_Spring2016_FinalProjects/projects/water/index.html

Human Trafficking: Sex Trade and Forced Labor
https://sureshlodha.github.io/CMPS165_Spring2016_FinalProjects/projects/humanTrafficking/index.html

New York vs Chicago
https://sureshlodha.github.io/CMPS165_Spring2016_FinalProjects/projects/NYvsChicago/index.html

Map of Gods
https://sureshlodha.github.io/CMPS165_Winter15_FinalProjects/MapOfGods/

Comparison of Indian States: Population, Literacy, Employment, and Crime
https://sureshlodha.github.io/CMPS263_Win2017/IndianStatesComparison/

When a CS graduate student can buy a 70m^2 home in Beijing
https://sureshlodha.github.io/CMPS263_Win2017/BuyingHomeBeijing/

China Population Density, 2014 (Provinces and Counties)
https://sureshlodha.github.io/CMPS263_Win2017/ChinaPopulationDensity/

All of the following can be found on:
https://sureshlodha.github.io/CMPS165_Fall2016_FinalProjects/

Out of School Children in India

Wildfires in California

USA: National Parks Visitor Count

US Population Density: Census Tract

Australia Topology

HIV in Africa

Syrian Conflict

Clan System in Scotland

Europe; Renewable Energy

Taiwan: Political Views Vary over Time

AEAN: Human Development Index

Birth-Death Rate in Japan

China: Population and GDP

Urban Displacement in Santa Cruz County