Nathan Mitchell

Geog:3540 Assignment 1 – Cheat Sheet

February 15, 2021

**Transforming Spatial Data**

Shapefile to topoJson

1. Open a browser and navigate to <https://mapshaper.org/>.

2. Click the “select” button in the center of the page and select the location of the desired shapefile.

4. Leave “detect line intersections” checked and hit “import”.

5. A display of the shapefile should appear. If the file is too large, choosing “simplify” can help reduce the size.

7. Select “export” at the top right of the screen and select “topoJSON” and click the export button.

8. The topoJSON will download to your computer, which can be later accessed and manipulated

DBF to CSV

1. Using mapshaper once again, click the “select” button and navigate to the location of the shapefile.

2. Select the file ending in .dbf and click “import”, the screen should display some boxes.

3. Find the “export” button again.

4. This time, select the “CSV” option and hit export.

**Upload and Read JSON on Observable**

1. Navigate to <https://observablehq.com> and edit/create a notebook.

2. Using the three dots in the upper right, select “file attachments”.

3. Select the json that was downloaded earlier and upload the file.

4. To access these files through the program, they must be plugged into the kernels. In their own kernels, name the notebook and set “d3” equal to: “require("d3@5")”.

5. Create another kernel and enter “topojson = require(“topojson-client@3”)” to give value to the topojson variable.

6. To insert the topojson file, enter “Cities = FileAttachment("eu\_cities.json").json()” into a new kernel. Replace “Cities” with corresponding variable name and “eu\_cities” with the name of the json file.

**Upload and Read CSV on Observable**

1. Once the “Upload and Read JSON on Observable” steps are complete, the CSV can be added to illuminate the features of the dataset.

2. In a new kernel, beneath the ones created in the previous steps, enter “csv\_data = d3.csvParse(await FileAttachment("eu\_cities.csv").text())”. Replace “csv\_data” with corresponding variable name and “eu\_cities” with the name of the csv file.

**Creating a new variable**

1. Using the previous kernel’s code and created variable, it is possible to use existing variables from the CSV to alter the new variable. In this case Population is taken over Population Rank to normalize the data.

2. In the CSV import kernel, append “({FIPS, POP, POP\_RANK}) => [FIPS, [+POP,+POP/+POP\_RANK]]” as a second argument. The kernel should now look like: “csv\_data = d3.csvParse(await FileAttachment("eu\_cities.csv").text(), ({FIPS, POP, POP\_RANK}) => [FIPS, [+POP,+POP/+POP\_RANK]])”.

3. Make sure the variable names are correct, the code runs, and the notebook is published.

**Observable Notebook:** https://observablehq.com/@njmitchell20/cs-3540-assignment-1