The More You Snow

Chloe Kurzon, Andrew Moran, and Jay Chakravarty

Final Project CS171 (Spring 2015)

Process Book

CS171 Project Proposal

Background and Motivation

Snowfall has been mentioned by several leading environmental agencies as an important indicator of climate change globally. In light of recent snow events in Boston (i.e. Boston having its snowiest year on record), we thought it would be interesting to visualize changes in snow patterns over time throughout the U.S. to see if recent patterns are actually abnormal, and could therefore indicate real changes in the earth's climate. We want to create a visualization that makes it easy for users to compare changes in snow patterns for the entire U.S. and smaller regions over time, and relative to each other. To accomplish this task, we plan on making use of the NASA MODIS dataset, which uses satellite imagery to create a dataset that provides snowfall data on the latitudinal/longitudinal level. In addition to relaying interesting information about snowfall, we decided to pursue working with the NASA MODIS dataset because we thought it presented interesting challenges related to visualizing large datasets. This dataset is also particularly appropriate for this task because it provides the data in a format that will make it easy to create a comprehensive map-view of snowfall in the United States. Additionally, one of our team members, Andrew Moran, is currently involved in a research project related to the utilization of this dataset.

Project Objectives

Our main goal is to create a detailed visualization of snowfall in the US from satellite imagery data. We want a very interactive and intuitive interface that makes it easy to discover snowfall patterns over time. With current concerns of climate change, we thought that users can potentially predict weather patterns based on previous measures in snowfall amount, coverage, duration, etc. This could eventually extend to making comparisons and drawing conclusions of snowfall in other regions of the globe. We plan to achieve this using a map-based layout. Remaining in the geospatial domain will allow us to experience the added utility of superimposing a large dataset on a well-known geographical display. The user will be given the freedom to browse any region of interest and request more detail via added navigation features (zooming, panning, etc). Creating our own map interface will give us more insight on how other common navigation tools work (e.g Google maps, Open StreetMap). In addition to usability, we plan to exploit the benefits of representing data in a tile-based format. This format allows data to be queried and aggregated faster, providing a rich/high resolution display with the reduced time cost. Also, this representation will allow our visualizer to remain running smoothly without the loss of detail. Enhanced performance is a large benefit, especially is there is potential to use this visualizer for other datasets.

Data

Data will be acquired from the NASA MODIS site. The NASA MODIS is a satellite instrument that records satellite imagery data. This data is stored in a 3D array indexed by latitude, longitude and time (taken from one week of MODIS measurements). We may limit the time range to minimize the scope of the data we are working with. To start, we will be

visualizing global snowfall coverage. However, this can potentially be extended to other satellite imagery information such as vegetation estimates on land, phytoplankton populations in the ocean, etc. (Battle et al., unpublished, see attached)

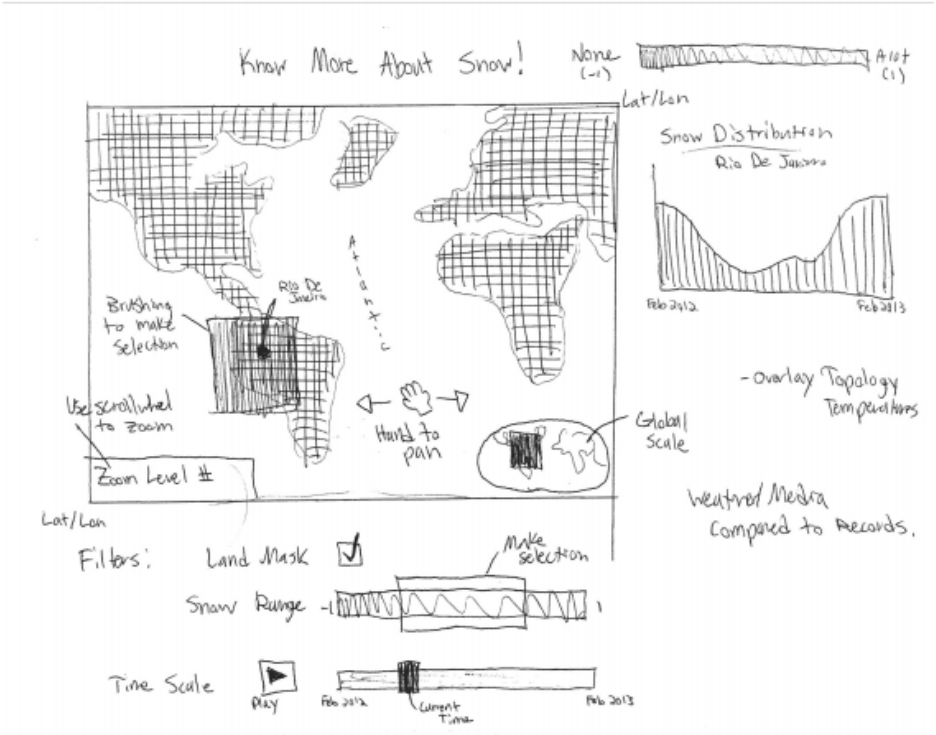
Data Processing

Not much data processing needs to be done besides reformatting the MODIS data into a tile format to help with zooming. We will be using numerical, array-based data so extracting information would not require additional processing. If we extend to non-integer based data, we will utilize Google Refine. (Battle et al., unpublished, see attached).

Visualization

For our visualization, we plan to have a map of the US (with the plan of expanding to a map of the entire world, if we have time). We will use a pixel-based method where each pixel is colored according to the amount of snowfall in that lat/lon region. In our snowfall example, pixels are colored on a gradient corresponding to a low or high NSDI (Normalized Snow Difference Index) value. Similar to a Google Maps view, users can navigate by zooming & panning. Additional widgets (e.g., scales, radio buttons) on the bottom of the screen will help filter pixels by various characteristics recorded in the dataset (e.g., whether it is a land/ocean pixel, min/max snowfall). The map view will also include an additional view on the side that includes a graph of historical snowfall data for a selected region. Additionally, we will include a table-view of the data set with columns sortable by continent, country, and snowfall level.

Initial Sketch:



Must-Have Features

- Zooming
- Panning
- Filtering - Choose min/max snowfall and only view regions that fall within this range; land/sea mask
- Selection/Brushing - change the view of a graph of the snowfall for the selected region.
- Tool-tips (hover over a pixel to view info specific to the tile/pixel)
- Timeline slider/play button
- Sortable table with data
- Only a map of U.S.

Optional Features

- Overlay height map data with NDSI data (similar to http://colorbrewer2.org/)
- Expand map to entire world
- Overlay of country borders
- Interchangeable MODIS data (Snowfall, vegetation, etc)
- Secondary map to show zoom level in relation to global scale
- Overlay of other relevant data sets (e.g. agriculture, wind speeds, weather-related diseases)
- Pre-set zoom features (e.g. person can select a country or continent from a drop down menu and view the zoomed version automatically on the map).
- Aggregate data by country and/or continent
- Small-secondary map that shows current zoomed-in position relative to the global map as a whole.
- Option to change the relative color scheme of a selected region (e.g. only scale based on the selected region vs. scaling for the whole globe).

Project Schedule

We plan to meet the milestones outlined on the course project page. Below is our project outline in accordance to key dates/milestones.

- W1 (Saturday, 04/11/15): Setup Repo for Development - Server/Client/Data
- W2 (Friday, 04/17/15): Simple Map-Based Visualization (color-coded by snowfall with tooltips), table with sortable columns **Milestone 1**
- W3 (Friday, 04/24/15): Fundamental Navigation (zooming,panning), time slider, filter by max/min snowfall **TF Review**
- W4 (Friday, 05/1/15): Feature Implementation (aggregation, filtering), additional views (graph of snowfall for selected region, separate view of zoomed in area).
- Tuesday, (05/5/15): any additional optional features we have not yet implemented, video
- **Final Submission**

4/7/15 - Meeting after class

[entry by amoran]

Today our team discussed about data representation/acquisition. Andrew met with an outside class member working on a related project about potentially integrating NASA MODIS data. It is up to the team's discretion how to format the dense dataset so it is easily readable/parable. We will come up with a basic JSON format and change it as we start developing our prototype. Meanwhile, we will continue to explore map visualizations. In particular, D3 implementations.

By Saturday:

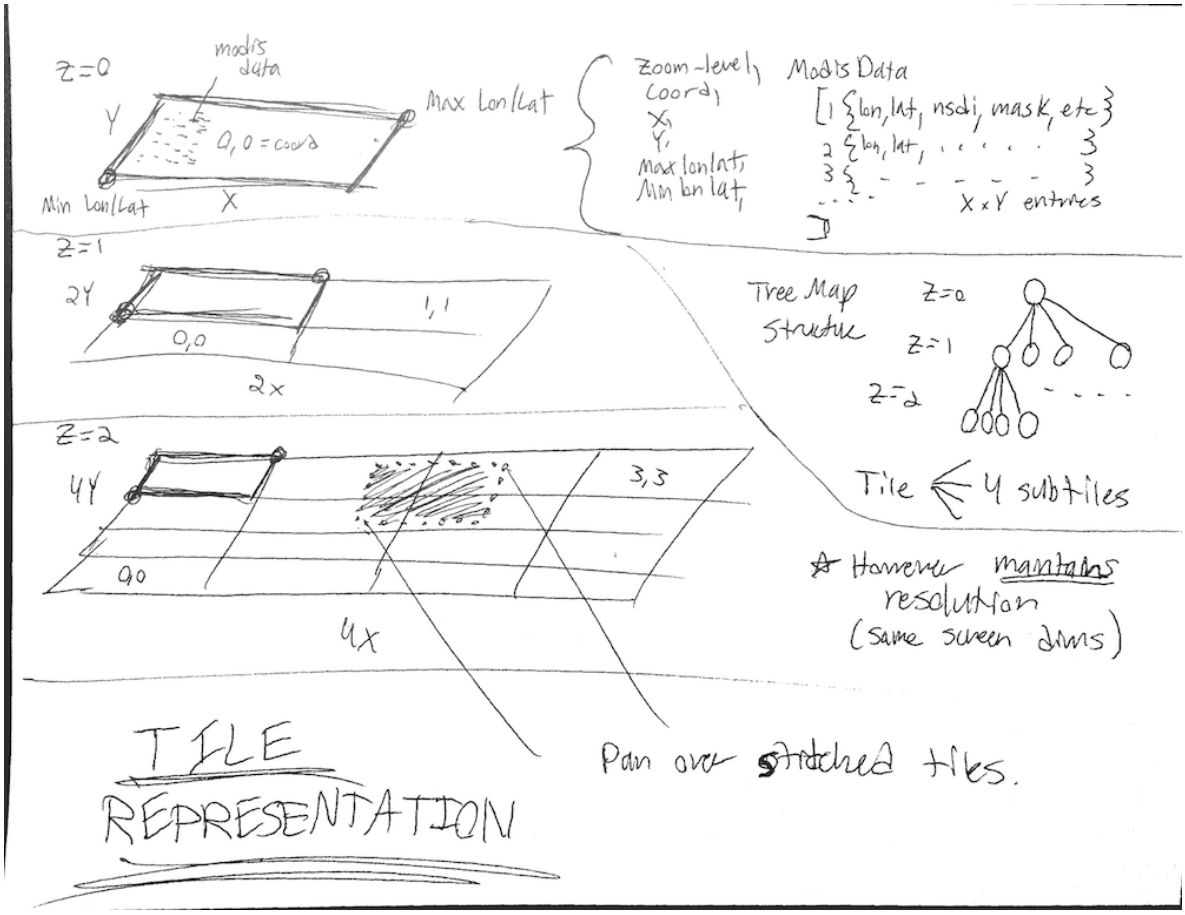
Andrew plans to have a data format template, a written script for JSON request from server, and a small sample ready. Potentially reach out to TF on suggestions for dense data rendering on browser (potential issues such as lagging, etc)

Chloe and Jay will explore current D3 map projections/tools and how to bind data to simple map layouts.

4/8/15 - Data/Tile Representation

[entry by amoran]

Continued to think about data/tile representation. Below is a sketch on how I picture the tile scheme working



It will be a recursive treemap representation where tiles remain the same size but become aggregated the higher level zoom level you are. Just to start of with, each tile will have a JSON format like the following:

```
id: {dataset: “modis”,
"zoom": 0,
"coord" : [0,0]},
“dims”: [“lat”,”lon”,”time”],
“resolution” : [1024,1024,0]
"min" : [0,0,0],
"max" : [1024, 1024,0],
"data": [
{ "lon" : 0,
```

```
  "lat" : 0,
  "avg_nsdi" : 0.5,
  "min_nsdi" : -1,
  "max_nsdi" : 1,
  "count_nsdi" : 100,
  "mask" : 0
},
{}, {}, {}, {}, {}          // ... 1024 X 1024 entries
],
“parent” : {#id}
“neighbors” : [{#id}, {#id}, {#id}, {#id} ]
```

link to retrieve data:
http://modis.csail.mit.edu:11001/fetch/0_1_0

tile id format for link: “.../fetch/<zoom>_<xcoord>_<ycoord>”

current implementation (1 tile):
Note: min max = range of tile boundaries ("min": [0, 0], "max": [3599, 1799])
lows/highs = upper/lower range of the given tile ("highs": [149, 149], "highs": [149, 149])
tile “0_0_0” has (-179,-89) as its lowest coord range
tile “0_23_11” has (179,89) as its highest coord range

from python:
```
“    return {
        'id':None,
        'attrs':[],
        'dtype':{},
        'data':[],
        'parent':[],
        'neighbors':[]
    }”
```

data sample:
"neighbors": [{"arrayname": "ndsi_agg_04_10_2015", "min": [0, 0], "max": [3599, 1799], "highs": [149, 149], "zoom": 0, "dims": ["longitude_e4ndsi_agg_04_10_2015", "latitude_e4ndsi_agg_04_10_2015"], "coords": [1, 0], "lows": [0, 0], "resolution": [50, 50]}, {"arrayname": "ndsi_agg_04_10_2015", "min": [0, 0], "max": [3599, 1799], "highs": [149, 149], "zoom": 0, "dims": ["longitude_e4ndsi_agg_04_10_2015", "latitude_e4ndsi_agg_04_10_2015"], "coords": [0, 1], "lows": [0, 0], "resolution": [50, 50]}],

"parent": [],

"dtype": {"min_land_sea_mask": "uint8", "min_lat": "double", "max_ndsi": "double", "longitude_e4ndsi_agg_04_10_2015": "int64", "min_ndsi": "double", "avg_ndsi": "double", "latitude_e4ndsi_agg_04_10_2015": "int64", "max_lat": "double", "max_land_sea_mask": "uint8", "min_lon": "double", "max_lon": "double", "avg_lat": "double", "avg_lon": "double"},

"attrs": ["avg_ndsi", "max_ndsi", "min_ndsi", "max_land_sea_mask", "min_land_sea_mask", "min_lat", "max_lat", "avg_lat", "max_lon", "min_lon", "avg_lon"],

"data": [
{"min_land_sea_mask": 1, "min_lat": -89.9748, "max_ndsi": 0.4697842395345295, "longitude_e4ndsi_agg_04_10_2015": 0, "min_ndsi": 0.12016576445557625, "avg_ndsi": 0.3397024633537475, "latitude_e4ndsi_agg_04_10_2015": 0, "max_lat": -89.7007, "max_land_sea_mask": 1, "min_lon": -179.998, "max_lon": -179.7068, "avg_lat": -89.84217620830057, "avg_lon": -179.84690677448958},
{"min_land_sea_mask": 0, "min_lat": -84.3, "max_ndsi": 0.9070968514112915, "longitude_e4ndsi_agg_04_10_2015": 0, "min_ndsi": 0.27715124031127036, "avg_ndsi": 0.7195162319970785, "latitude_e4ndsi_agg_04_10_2015": 19, "max_lat": -84.0001, "max_land_sea_mask": 0, "min_lon": -165.0, "max_lon": -164.7001, "avg_lat": -84.1502631182468, "avg_lon": -164.84991795994148}
],

"id": {"arrayname": "ndsi_agg_04_10_2015", "min": [0, 0], "max": [3599, 1799], "highs": [149, 149], "zoom": 0, "dims": ["longitude_e4ndsi_agg_04_10_2015", "latitude_e4ndsi_agg_04_10_2015"], "coords": [0, 0], "lows": [0, 0], "resolution": [50, 50]}

land sea mask definitions

Found this here:
http://lgge.osug.fr/~picard/enseignement/fichiers/Teledetection_M1/TPTemperatureSurface/MOD03.geolocation.fs

DN values:

0: Shallow Ocean (Ocean <5k from coast OR <50m deep).
1: Land (not anything else).
2: Ocean Coastlines and Lake Shorelines.
3: Shallow Inland Water (Inland Water < 5km from shore OR < 50m deep).
4: Ephemeral (intermittent) Water.
5: Deep Inland Water (Inland water > 5km from shoreline AND > 50m deep).
6: Moderate or Continental Ocean (Ocean > 5km from coast AND > 50m deep AND < 500m deep).
7: Deep Ocean (Ocean > 500m deep).

4/13/15 - First Meeting With TF

[entry by ckurzon]

- Daniel likes our project idea but is concerned about the difficulty with implementing the tile architecture. He thinks zooming with tiles will be extremely difficult
- Daniel suggests just implementing a global map with interactive and not implementing zoom unless we have extra time.
- Daniel is concerned about datasize. We need to make sure that on each load of the data, no more than 1 MB is loaded
- Daniel has concerns that the backend of our project is coming from an MIT Grad student who is not ultimately being graded for this project
- Need to remember that this project is focused on the visualization and front end, not on having a complex backend
- We realize during the meeting that we might not have a time component for our dataset (i.e. the data we are receiving might just be for one point in time).
    - This severely restricts our options for interactivity. Daniel is concerned about the number of views we can implement with only snowfall data for one point in time
    - Andrew is going to talk to the MIT grad student who is providing us the data to see if we can get a time component for it.
- Daniel has data concerns and states that we need to have a function to quickly request snowfall data by latitude and longitude by friday if he is going to feel comfortable with the state of our project
- Following the TF meeting, we decide to consider switching datasets and/or projects, pending Andrew's discussion with the MIT grad student who is providing us with data.

Update:

- Andrew talks to the MIT grad student who is providing us with this data, and ultimately decides that this dataset is not the best for our project considering the amount of time we have
- Team meeting is scheduled for tomorrow (Tuesday) to discuss potentially switching projects
- Daniel emails us a suggestion for a different dataset for snowfall data, that is on a much smaller scale than the one we were considering using before, from the NCDC: https://www.ncdc.noaa.gov/snow-and-ice/daily-snow/

4/14/15-Team Meeting and Design Studio

[entry by ckurzon]

- Team meeting to discuss switching project ideas because we don't think our current dataset is ideal for this project
- Consider switching to World Health Organization data, but ultimately decide to use the new dataset Daniel recommended and stick with our original sketches for thimplementation
- Unsure if we want to focus on snowfall data or cumulative snow data (can do both with new dataset)
    - Could possibly represent cumulative snow data with a 3D map
    - Could allow users to toggle between both
- To-Dos:
    - Andrew: By Wednesday night, Andrew will download all of data from NCDC and merge it into one tsv file that is organized in logical manner for parsing
    - Jay: will work on loading in the data
    - Chloe: make a draft of a static map visualization encoding the new dataset by meeting on friday
- Design studio feedback summary:
    - should probably stay away from 3D maps
    - Need to come up with a tool to compare different counties-possibly through a bar graph when brushing
    - Need to add a feature that allows users to aggregate by county or state

4/15/15- Map Projections

[entry by jchakravarty]

d3 map projections:

It is important that we find a d3 map projection appropriate to our dataset. One possible option is d3.geo.mercator() which displays the entire world without curvature.

The spherical Mercator projection is commonly used by tiled mapping libraries (such as OpenLayers and Leaflet). For an example displaying raster tiles with the Mercator projection, see the d3.geo.tile plugin.

Mercator Script:

```
<script type="text/javascript" src="d3/d3.v3.js"></script>
<script src="js/topojson.v0.min.js"></script>
<script>
var width = 960,
    height = 500;

var projection = d3.geo.mercator()
    .center([0, 5 ])
    .scale(900)
    .rotate([-180,0]);

var svg = d3.select("body").append("svg")
    .attr("width", width)
    .attr("height", height);

var path = d3.geo.path()
    .projection(projection);

var g = svg.append("g");

d3.json("json/world-110m2.json", function(error, topology) {
    g.selectAll("path")
      .data(topojson.object(topology, topology.objects.countries)
          .geometries)
    .enter()
```

```
    .append("path")
    .attr("d", path)
});

</script>
```

However, now that we are using the tabular U.S. daily snowfall and snowdepth data from the National Oceanic and Atmospheric Administration, it will be more appropriate to use a map projection displaying solely the U.S.

Furthermore, given the area distortion potentially introduced by the mercator projection, it will create issues for choropleths (map in which areas are shaded or patterned in proportion to the measurement of a statistical variable being displayed on the map, such as, in our case, snowfall and snowdepth).

A better option may be d3.geo.conicEqualArea() which is more suitable to choropleths given that it preserves the relative areas of geographic features.

This projection also includes county lines which may be appropriate divisions by which the shade the map. Luckily, the data already includes the county of every weather station at which metrics have been recorded so the process of aggregating snowfall and snowdepth by county will be relatively straightforward.

4/15/15- Parsing the New Dataset
[entry by andrewmo]

- Due to foreseeable issues with the tiling schema and more overhead required to work with the complicated dataset, we decided to work with a new one.
- Thanks to Daniel's advice we found the below dataset to use for recorded snowfall: http://www.ncdc.noaa.gov/snow-and-ice/daily-snow/ (sample shown below)

State: AL
Lat Lon COOP# StnID State City/Station Name County Elev Apr 1 Apr 2 Apr 3 Apr 4 Apr 5 Apr 6 Apr 7 Apr 8 Apr 9 Apr10 Apr11 Apr12 Apr13 Apr14
33.59 -85.86 010272 AL ANNISTON ARPT ASOS CALHOUN 594 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 -9999.000

- Working with this dataset has its pros and cons.
    - Pros: Many attributes, dates back 10 years, many counties, lat/lon points
    - Cons: .txt file, needs to be reformatted to be utilized in d3. I decided to write a python script to convert this text file to JSON/CSV.

Parsing Implementation:

- Created a python script that grabs .txt from url
- I iterate through all desired months and years to get a unique snowfall/depth for that date. (update string url)
- First I want to remove unnecessary lines:
    - newlines, current state, headers
- Next, I thought that all I had to do was split each line by spacing, however, some values are names that are multiple words, etc. I discovered that each value had a fixed amount of spacing for that header field. EUREKA!
- Read so many characters for that line and pointer shows which header value you are reading.
- Something else was that not all months had the same number of days. The .txt file only showed days leading up to the end of the month. For consistency I made an empty placeholder for all possible days and did not populate the days field if it did not exist (e.g. Feb 30)
- Lastly I stripped any additional padding on either side of the desired field value and separated each line by a tab and wrote that line to a new file.
- Each file is about 115MB (Aggregated for all months all states for 10 years)

Added Files:

counties.json (not sure if necessary)

```
[
        {state: “Alabama”,
        counties: [ … ],
        abbrev: “AL”
        },
        …
]
```

(A mapping of states with their abbreviation and a list of all county names)

snfl.tsv (same idea for sndpth.tsv)

- .tsv instead of json so dont have to do with brackets, commas, spacing, etc
- uncollected data is empty represented as empty string “”
    - This happens for no ids for stations and days that do not have full 31 days.
- invalid data is shown as -9999.000
- not sure what range of values are just yet (most extreme)
- similar to previous psets need to aggregate by month/year/state/etc to get other insight

Below is a sample of the formatted data:

latitude		longitude		coop	id	state	station	county	elevation		year	month
day1	day2	day3	day4	day5	day6	day7	day8	day9	day10	day11	day12	day13
day14	day15	day16	day17	day18	day19	day20	day21	day22	day23	day24	day25	day26
day27	day28	day29	day30	day31								
34.74	-87.60	015749			AL	MUSCLE SHOALS AP			COLBERT		540	
2005	07	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
0.000	0.000	0.000	0.000	0.000	0.000	0.000						

Look at https://github.com/mbostock/d3/wiki/CSV to utilize delimited files in d3

Worth mentioning

- Moving forward, Below are some example of county maps I believe can be useful.
    - http://vizhub.healthdata.org/us-health-map/
    - http://code.minnpost.com/simple-map-d3/
    - http://bl.ocks.org/mbostock/4060606
    - http://bl.ocks.org/rwaldin/6244803
    - http://scribu.net/blog/3d-maps-using-d3-and-three.js.html
- I think it is good to note down some libraries we should utilize for development (Libraries to use):
    - topojson
    - d3.js
    - d3-threeD.js
    - three.js

- colorbrewer
- bootstrap

4/16/15 - Loading Data
[entry by jchakravarty]

In order to upload the tsv file we will need to use:

d3.tsv(url[, accessor][, callback])
Issues an HTTP GET request for the comma-separated values (CSV) file at the specified url.

Using:

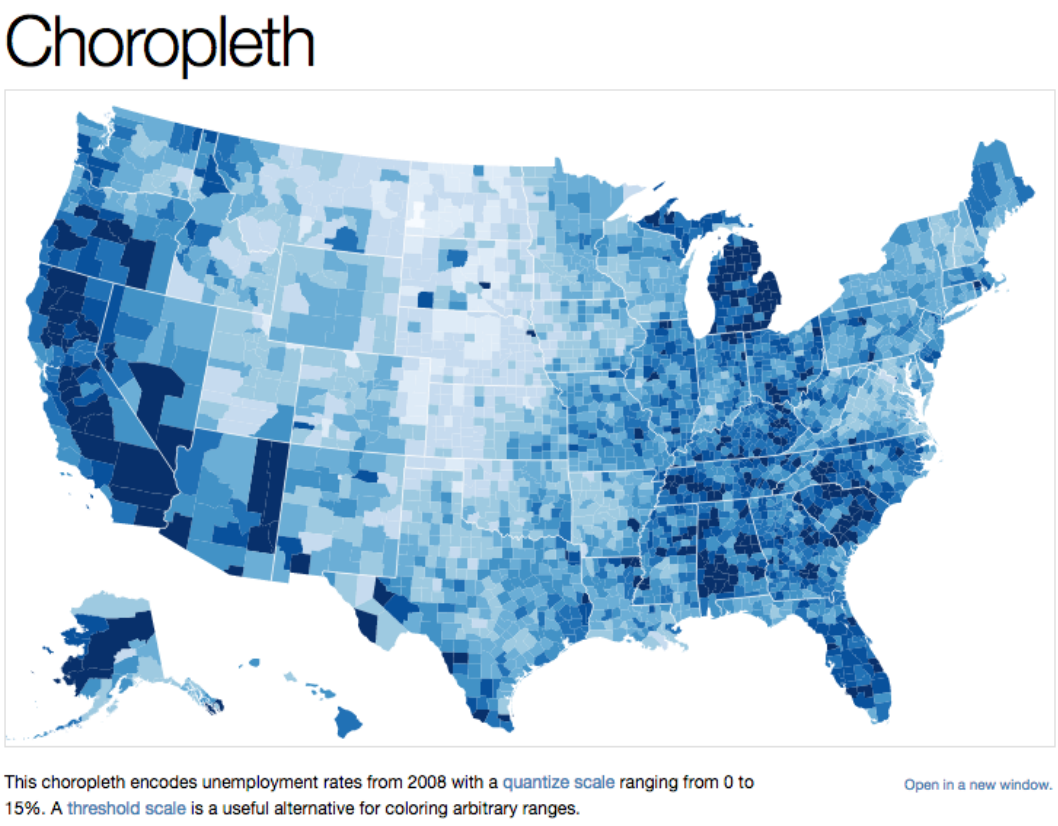
```
var downloaded_data;

d3.tsv(data/sndpth.tsv, function (error, data) {
        downloaded_data = data;
});
```

The data is available via console.log(downloaded_data);

4/16/15-Working on Mapping Data

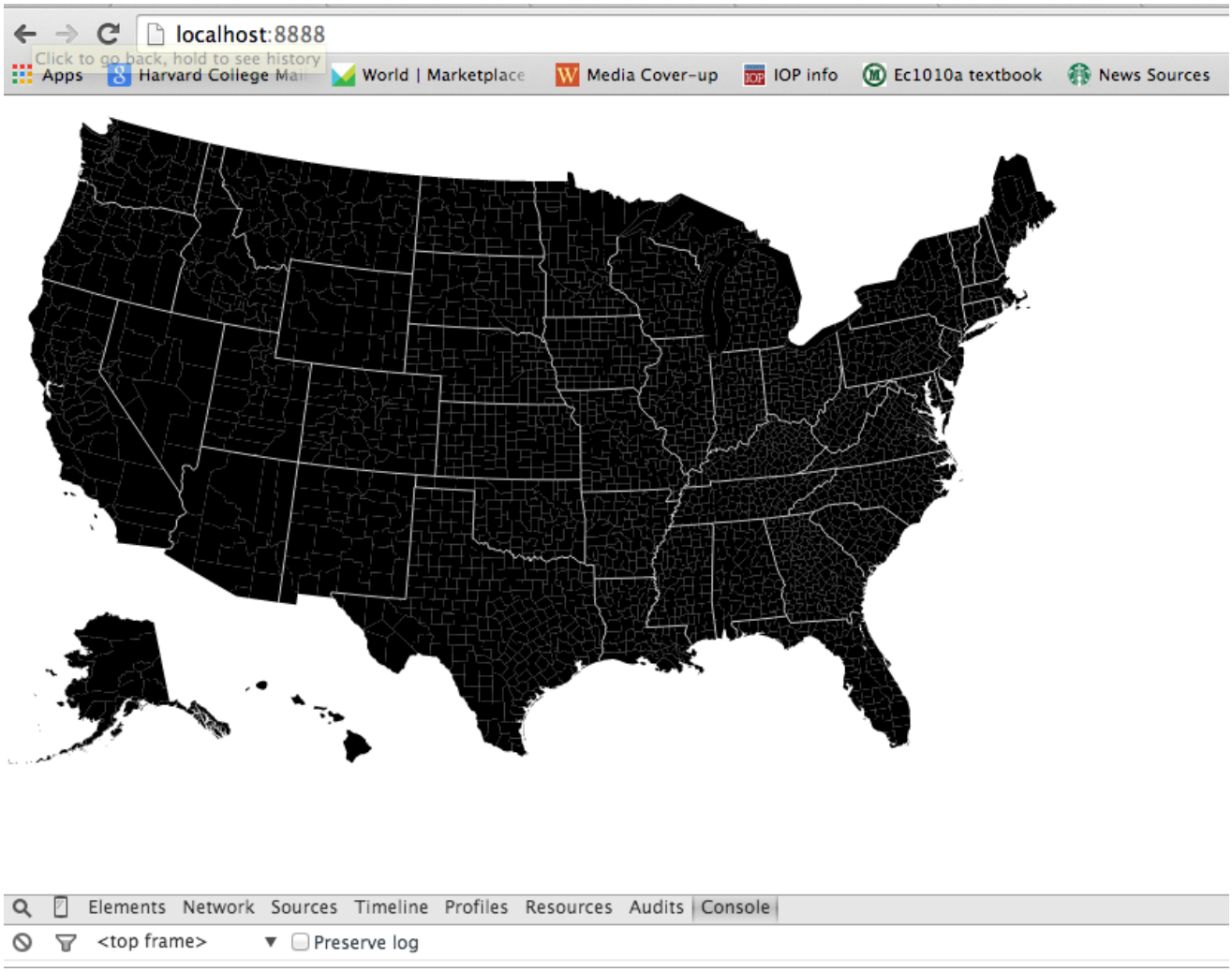
[entry by ckurzon]

Goal: To implement a choropleth U.S. map of a snapshot of the dataset that Andrew has parsed. The color coding should be based on the county's snowfall on the selected day. I want my map to look something like this:



Source: http://bl.ocks.org/mbostock/4060606

Methods: I first copy the code from this example. I noticed that the tsv file used has counties in the form of IDs. After looking at several other D3 choropleth maps, I discover that this ID scheme for counties is standard. I learn that this ID scheme is referred to as “FIPS”. This is a problem because our dataset only has county names and no county ids. I find a text file online that gives the FIPS for each individual county at the following link:
http://www2.census.gov/geo/docs/reference/codes/files/national_county.txt
I import this into excel and use a vlookup to match the entries in our dataset with their FIPS code.
After completing this, I then map the data using January 1st 2015 snowfall. The resulting map looks like this:



There’s clearly an issue with either the data or the color scale, but I was not able to figure it out at this time.

Remaining To-Dos:

- color scale of the map is not working for some reason, map is all just one color
- need to figure out how we want to deal with missing values in the dataset
- At the moment, some counties have multiple weather stations and we need to average out this weather station data so that each county only has one data point representing it.

4/17/15- Team Meeting

[entry by ckurzon]

- We have a map display of a static portion of our visualization, and next need to work on color-coding it and on making the visualization interactive
- Task assignments due on Monday, 4/20/15:
    - Andrew: Fix the color scheme of the map implementation; create the layout for the svg elements (i.e., map out where the different views should go)
    - Jay: Make a function that aggregates the data by year, e.g. function takes in year as an argument, and then sums the snowfall data for each county in that year. Add a time slider.
    - Chloe: Make a static histogram of snowfall by county
- Will next meet on Sunday to do a progress check and see if people are running into issues
- Will combine all of our separate parts on monday/tuesday to have an interactive version by TF meeting on Wednesday.