Part II: Programming Geo-Data Visualizations

http://patompa.github.io/geovizdev/
## Agenda

<table>
<thead>
<tr>
<th>Topic</th>
<th>Duration</th>
</tr>
</thead>
<tbody>
<tr>
<td>Overview</td>
<td>8 min</td>
</tr>
<tr>
<td>Preparing the Data</td>
<td>2 min</td>
</tr>
<tr>
<td>Recipe 1: Server-side Rendering</td>
<td>15 min</td>
</tr>
<tr>
<td>Recipe 2: Data-Driven Documents</td>
<td>15 min</td>
</tr>
<tr>
<td>Recipe 3: Visualizing Time</td>
<td>5 min</td>
</tr>
<tr>
<td>Coffee Break</td>
<td>30 min</td>
</tr>
<tr>
<td>Recipe 4: Draw-it-yourself</td>
<td>10 min</td>
</tr>
<tr>
<td>Recipe 5: Route Visualization</td>
<td>5 min</td>
</tr>
<tr>
<td>Bonus Recipe: Scripting</td>
<td></td>
</tr>
</tbody>
</table>
Why Program Thematic Maps?

- Exploratory data analysis
- Dynamic rendering
- Scalability
- Interactivity
Inspiration

Explanatory Visualization Guides:

- [http://www.ted.com/talks/hans_rosling_shows_the_best_stats_you_ve_ever_seen](http://www.ted.com/talks/hans_rosling_shows_the_best_stats_you_ve_ever_seen)

http://www.sightsmap.com

http://www.facebook.com/notes/facebook-engineering/visualizing-friendships/469716398919
Approaches

• Direct Plotting (DP) Marker positioning
  + simple, flexible
  - only handles few points, slow, point occlusion

• Area Aggregation (AA) Choro- and Isopleths
  + handles many points, fast, easy to interpret
  - relies on geocoding, may misrepresent areas

• Heatmap (HM) Radial diffusion and blending
  + discovers hotspots, no point occlusion, handles many points
  - slow, could be hard to read, artificial gradients
Programming a Thematic Map

- Preparing Data
- Uploading Data
- Prototyping
- Developing Code
- Deploying Code
Preparing Data

- Google REST API (rate limit):
  https://developers.google.com/maps/documentation/geocoding/

- Geonames (download):
  http://download.geonames.org/export/dump/

- Adding more
  Area to census data (FIPS to population, income etc)

For more details see:
https://github.com/patompa/geovizdev/blob/master/utils/addlocation.py
Recipe 1: Server-side Rendering

Recipe 2: Data-driven Documents

Recipe 3: Visualizing Time

Recipe 4: Draw-It-Yourself

Recipe 5: Route Visualization
Recipe 1: Server-side Rendering

http://patompa.github.io/geovizdev/demos/fusionheat/

Types: DP, HM
Tools: Fusion Tables, Google Maps, Stamen Tiles
Key Ideas: Many points, Pre-render images on Server, Hosted server, No prep-work

Step 1: Upload to Google Drive FusionTable
Step 2: Write Javascript with Google Maps API
R1 Step 1: Upload to FusionTable
R1 Step 2: Write Javascript

```javascript
var mapOptions = {
  zoom: zoom,
  center: center,
  disableDefaultUI: true,
  mapTypeId: stamenlayer,
  mapTypeControlOptions: {
    mapTypeIds: [stamenlayer]
  }
};

map = new google.maps.Map(document.getElementById('map-canvas'), mapOptions);
var layer = new google.maps.FusionTablesLayer({
  query: {
    select: 'location',
    from: '1AG4tCmC0CRUMQ4KECBpWePRuq_hbMwHHT_6OD40'
  },
  heatmap: {
    enabled: true
  }
});
layer.setMap(map);
var stamenMap = new google.maps.StamenMapType(stamenlayer);
map.mapTypes.set(stamenlayer, stamenMap);
```

For more details see:
Recipe 1: Server-side Rendering

Recipe 2: Data-driven Documents

Recipe 3: Visualizing Time

Recipe 4: Draw-It-Yourself

Recipe 5: Route Visualization
Recipe 2: Data-driven Documents

http://patompa.github.io/geovizdev/demos/d3/

Step 1: Aggregate by County
Step 2: Get TopoJSON Area polygons
Step 3: Create Choropleth

Types: DP, AA
Tools: D3, Tableau Public
Key Ideas: Tie data to DOM, use SVG for speed and interactivity
R2 Step 2-3: Aggregate by County and Get Area Polygons

• In D3 manual aggregation and map drawing is needed (Tableau Public does this for you)

• TopoJSON, more efficient GeoJSON format used by D3 to draw maps

• US county/state available at:
  
  http://bl.ocks.org/mbostock/raw/4090846/us.json

• Area polygons may also be created from GIS tools and converted from public shape files, see:
  
  http://bost.ocks.org/mike/map/
R2 Step 3: Create Tableau Public Choropleth

For more details see:
http://public.tableausoftware.com/views/TweetDensity/State
R2 Step 3: Create D3 Choropleth

```javascript
queue()
  .defer(d3.json, "us.json")
  .defer(d3.tsv, "/utils/samplelococcounty.tsv", function(d) {
    rateById.set(d.county, +d.count);
  })
  .await(ready);

function ready(error, us) {
  svg.append("g")
    .attr("class", "counties")
    .selectAll("path")
      .data(topojson.feature(us, us.objects.counties).features)
      .enter().append("path")
      .attr("class", function(d) { return quantize(rateById.get(d.id)); })
      .attr("d", path);

  svg.append("path")
    .datum(topojson.mesh(us, us.objects.states, function(a, b) { return a !== b; }))
    .attr("class", "states")
    .attr("d", path);
}
```

For more details see:

https://github.com/mbostock/topojson/wiki/API-Reference
Recipe 1: Server-side Rendering

Recipe 2: Data-driven Documents

Recipe 3: Visualizing Time

Recipe 4: Draw-It-Yourself

Recipe 5: Route Visualization
Recipe 3: Visualizing Time

http://patompa.github.io/geovizdev/demos/ohm/

Types: AA, HM
Tools: Open Heat Map
Key Ideas: Show heatmap evolution over time

Step 1: Compute heat
Step 2: Prototype with OHM web tool
Step 3: Write OHM Javascript
R3 Step 1: Compute Heat

• OHM does not support heatmap blending (color aggregation)!
• Latitude, Longitude values need to have a heat value
• Fake point heat using geohash aggregation
• Each point has the heat based on number of points within same 100x100 mile geohash grid. See: https://github.com/patompa/geovizdev/blob/master/ohm/latlondens.py
R3 Step 2: Prototype with OHM Web Tool

For more details see: http://www.openheatmap.com/
R3 Step 3: Write OHM Javascript

```javascript
$(
    '#openheatmap_container'
).insertOpenHeatMap({
    width: 800,
    height: 600,
    source: 'openheatmap.swf'
});

var map = $.getOpenHeatMap();
map.setLatLonViewingArea(50, -126.58, 15, -66.73);
map.loadWaysFromFile('http://static.openheatmap.com/us_counties.osm');
map.loadValuesFromFile('latlon.csv');
map.setSetting('show_map_tiles', true);
map.setSetting('gradient_value_min', 5);
map.setSetting('gradient_value_max', 500);
map.setSetting('is_gradient_value_range_set', true);
map.setSetting('point_blob_radius', 0.2);
```

For more details see:
Coffee
Recipe 4: Draw-It-Yourself

http://patompa.github.io/geovizdev/demos/canvas/

Types: HM
Tools: HTML5 Canvas, D3
Key Ideas: Draw heatmap yourself with canvas and position on D3 map for maximum customizability

Step 1: Draw D3 Map and reuse projection
Step 2: Render heatmap
R4 Step 1: Draw D3 Map and Reuse Projection

```javascript
var projection = d3.geo.albersUsa()
    .scale(1000)
    .translate([width / 2, height / 2]);

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

d3.json("../d3/us.json", function(error, us) {
    svg.insert("path", ".graticule")
        .datum(topojson.feature(us, us.objects.land))
        .attr("class", "land")
        .attr("d", path);

    xy = projection([lon, lat])

    var ctx = myCanvas.getContext("2d");
    ctx.beginPath();
    ctx.arc(xy[0], xy[1], r, 0, 2 * Math.PI, false);
    ctx.fill();
```

For more details see:
https://github.com/patompa/geovizdev/blob/master/canvas/map.html
R4 Step 2: Render Heatmap

1. Draw Grayscale Circle with Radial Gradient
2. Blend points by adding pixel RGB values
3. Compute pixel luminance and colorize using 255-scale palette

For more details see:
https://github.com/patompa/geovizdev/blob/master/canvas/geo.js
Gaussian Blur

\[ G(x, y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}} \]

From http://finance.yendor.com/etfviz/2008/0301
Isopleths or Contour Maps

From http://enb110-ert-2012.blogspot.com/2012/08/maps-chloropleth-map-is-used-as-way-to.html

From http://paulbourke.net/papers/conrec/

See https://github.com/jasondavies/conrec.js
Recipe 1: Server-side Rendering

Recipe 2: Data-driven Documents

Recipe 3: Visualizing Time

Recipe 4: Draw-It-Yourself

Recipe 5: Route Visualization
Recipe 5: Route Visualization

http://patompa.github.io/geovizdev/demos/route/

Types: AA, DP
Tools: Google Directions API, Mongolab, RouteBoxer
Key Ideas: Box route and pull in points on demand for area aggregation and direct plotting

Step 1: Upload JSON to Mongolab
Step 2: Routebox Google Directions path
Step 3: Pull data and visualize

Based on WWW’14 Demo: http://mia.kaist.ac.kr/project/socroutes/
R5 Step 1: Upload to Mongolab

```bash
mongoimport -h ds061218.mongolab.com:61218 -d <db name> -c <collection name> -u user -p pwd --file <json file>
```

**crime.json**

```json
{
"lat": 41.752069205715991, "text": "12/14/2012 11:58:00 PM CRIMINAL DAMAGE TO CITY OF CHICAGO PROPERTY", "lon": -87.644229677461581
}
{
"lat": 41.88162468747845, "text": "12/14/2012 11:56:00 PM BATTERY DOMESTIC BATTERY SIMPLE", "lon": -87.75154695794852
}
{
"lat": 41.867305215905006, "text": "12/14/2012 11:50:00 PM CRIMINAL DAMAGE TO VEHICLE", "lon": -87.715304610287035
}
{
"lat": 41.908977645619956, "text": "12/14/2012 11:45:00 PM BATTERY DOMESTIC BATTERY SIMPLE", "lon": -87.63867258693792
}
{
"lat": 41.765808860199698, "text": "12/14/2012 11:40:00 PM ROBBERY STRONGARM - NO WEAPON", "lon": -87.615813855691911
}
```

**sentiment.json**

```json
{
"lat": 41.828851290000003, "lon": -87.682199550000007, "sentiment": -0.80000000000000004
}
{
"lat": 41.91660289, "lon": -87.687397199999996, "sentiment": -0.75
}
{
"lat": 41.908128220000002, "lon": -87.694693700000002, "sentiment": -0.75
}
{
"lat": 41.911173400000003, "lon": -87.641940059999996, "sentiment": 0.6666670000000001
}
{
"lat": 41.883652580000008, "lon": -87.630246880000001, "sentiment": 0.0
}
{
"lat": 41.973033239999999, "lon": -87.659767509999995, "sentiment": 0.0
}
```
R5 Step 2: Routebox Google Directions Path

```javascript
var polyOptions = {
  strokeColor: '#29088A',
  strokeOpacity: 0.7,
  strokeWeight: 4
};
var rendererOptions = {
  draggable: true,
  suppressBicyclingLayer: true,
  polylineOptions: polyOptions,
};
var directionsDisplay = new google.maps.DirectionsRenderer(rendererOptions);
var on_path = directionsDisplay.getDirections().routes[0].overview_path;
var routeBoxer = new RouteBoxer();
var boxes = routeBoxer.box(on_path, distance);
```

For more details see:
http://google-maps-utility-library-v3.googlecode.com/svn/trunk/routeboxer/docs/examples.html
R5 Step 3: Pull Data and Visualize

```javascript
var swlat = box.getSouthWest().lat();
var swlon = box.getSouthWest().lng();
var nelat = box.getNorthEast().lat();
var nelon = box.getNorthEast().lng();
query({"lat": {"$gt": swlat,"$lt":nelat},"lon": {"$gt":swlon,"$lt":nelon}},
  'sentiment',function (data) {
    for (var i=0; i< data.length; i++) {
      ...
    } });

var boxpolys = new Array(boxes.length);
for (var i = 0; i < boxes.length; i++) {
  boxpolys[i] = new google.maps.Rectangle({
    bounds: boxes[i],
    fillOpacity: Math.abs(sentimentValue[i]),
    strokeOpacity: 0.0,
    fillColor: sentimentColor,
    strokeWeight: 1,
    map: map,
    clickable: false
  });
}
```

For more details see:
https://github.com/patompa/geovizdev/blob/master/route/mongo.js
Parting Thoughts

• Pick a tool based on
  • Visualization Types supported
  • Size of your data set
  • Programmability
  • Online or Offline
  • Interactive or Static

• Word of caution
  • Pick colors carefully [http://colorbrewer2.org/](http://colorbrewer2.org/)
  • Aggregate, discretize and bin with care
  • Projections from 3D to 2D lie
# Tool Summary

<table>
<thead>
<tr>
<th>Tool</th>
<th>Pros</th>
<th>Cons</th>
</tr>
</thead>
<tbody>
<tr>
<td>R1: Fusion Tables</td>
<td>Server rendering for scalability. Heatmap and DP support.</td>
<td>Need to convert geodata into tabular form. Very limited configuration options for heatmap rendering. Data upload through browser may be slow.</td>
</tr>
<tr>
<td>R2: D3</td>
<td>Large collection of pre-drawn maps. Efficient map drawing and projection. Interactivity built into the browser. Styling built into the browser.</td>
<td>Steep learning curve. Client side rendering may be slow.</td>
</tr>
<tr>
<td>R2: Tableau Public</td>
<td>Fast to prototype DP and AA maps.</td>
<td>No programmability. Windows only. Strange saving behavior (save to web only).</td>
</tr>
<tr>
<td>R3: Open Heat Map</td>
<td>Online tool to quickly visualize time evolution. Minimal programming needed due to standard column name design.</td>
<td>Flash based. Need to rename data columns. No heatmap without aggregation.</td>
</tr>
<tr>
<td>R4: Canvas</td>
<td>Easy to program. Customization unlimited.</td>
<td>Scaling not as flexible as with SVG.</td>
</tr>
<tr>
<td>R5: RouteBoxer</td>
<td>Works well with bounding box queries. Easy to visualize.</td>
<td>May introduce artificial areas and gradients.</td>
</tr>
<tr>
<td>R5: MongoDB</td>
<td>Works well with json data such as Twitter dumps.</td>
<td>Rate limited for commercial use.</td>
</tr>
<tr>
<td>Tool</td>
<td>Purpose</td>
<td>Reference</td>
</tr>
<tr>
<td>---------------------</td>
<td>-------------------------------------------------------------------------</td>
<td>---------------------------------------------------------------------------</td>
</tr>
<tr>
<td>D3</td>
<td>Web visualization mostly focused on AA, very low level pixel-by-pixel</td>
<td><a href="https://github.com/mbostock/d3">https://github.com/mbostock/d3</a></td>
</tr>
<tr>
<td></td>
<td>control.</td>
<td><a href="http://bl.ocks.org/mbostock/4060606">http://bl.ocks.org/mbostock/4060606</a></td>
</tr>
<tr>
<td></td>
<td></td>
<td><a href="http://chimera.labs.oreilly.com/books/123000000345/index.html">http://chimera.labs.oreilly.com/books/123000000345/index.html</a></td>
</tr>
<tr>
<td>Google Maps</td>
<td>Basic Marker based overlays (DP) for Web visualizations.</td>
<td><a href="https://developers.google.com/maps/documentation/javascript/">https://developers.google.com/maps/documentation/javascript/</a></td>
</tr>
<tr>
<td>Google FusionTables</td>
<td>Online Table that can be accessed from Javascript and supports server</td>
<td><a href="http://chimera.labs.oreilly.com/books/123000000345/index.html">http://chimera.labs.oreilly.com/books/123000000345/index.html</a></td>
</tr>
<tr>
<td></td>
<td>rendered HM and DP.</td>
<td></td>
</tr>
<tr>
<td>Google Geocharts</td>
<td>AA for countries and regions</td>
<td><a href="https://developers.google.com/chart/interactive/docs/gallery/geochart">https://developers.google.com/chart/interactive/docs/gallery/geochart</a></td>
</tr>
<tr>
<td>Stamen</td>
<td>OpenStreetMap based Map Tiles</td>
<td><a href="http://maps.stamen.com/">http://maps.stamen.com/</a></td>
</tr>
<tr>
<td></td>
<td></td>
<td><a href="https://code.google.com/p/gheat/">https://code.google.com/p/gheat/</a></td>
</tr>
<tr>
<td>Python Heatmaps (sethoscope)</td>
<td>OpenStreetMap based Heatmaps</td>
<td><a href="http://www.sethoscope.net/heatmap/">http://www.sethoscope.net/heatmap/</a></td>
</tr>
<tr>
<td>Mongolab</td>
<td>Like Fusion Tables but for a json database instead of a table</td>
<td><a href="https://mongolab.com/welcome/">https://mongolab.com/welcome/</a></td>
</tr>
<tr>
<td>Tableau Public</td>
<td>Similar to D3 in functionality but UI based. County, State and City AA.</td>
<td><a href="http://www.tableausoftware.com/public/">http://www.tableausoftware.com/public/</a></td>
</tr>
<tr>
<td>Tool</td>
<td>Purpose</td>
<td>Reference</td>
</tr>
<tr>
<td>-------------------------------------------</td>
<td>-------------------------------------------------------------------------</td>
<td>----------------------------------------------------------------</td>
</tr>
<tr>
<td>Open Street Map Visualization Toolkit for Python</td>
<td>OpenStreetMap tile based visualizations in python, used by stethoscope heatmaps, see above</td>
<td><a href="http://cbick.github.io/osmviz/html/index.html">http://cbick.github.io/osmviz/html/index.html</a></td>
</tr>
<tr>
<td>RouteBoxer</td>
<td>Computes boxes around routes using google directions api to simplify area aggregation and lookup</td>
<td><a href="http://google-maps-utility-library-v3.googlecode.com/svn/trunk/routeboxer/docs/examples.html">http://google-maps-utility-library-v3.googlecode.com/svn/trunk/routeboxer/docs/examples.html</a></td>
</tr>
<tr>
<td>Google Directions API</td>
<td>Turn by Turn direction customization on the Web</td>
<td><a href="https://developers.google.com/maps/documentation/directions/">https://developers.google.com/maps/documentation/directions/</a></td>
</tr>
<tr>
<td>HTML5 Canvas</td>
<td>Powerful 2D drawing directly in browser</td>
<td><a href="http://diveintohtml5.info.canvas.html">http://diveintohtml5.info.canvas.html</a></td>
</tr>
<tr>
<td>Google Earth</td>
<td>Allows KLM aligned image overlays, to fit image heatmaps to geo maps</td>
<td><a href="http://www.google.com/earth/">http://www.google.com/earth/</a></td>
</tr>
<tr>
<td>Visualization Tool Guide</td>
<td>Review of 30+ visualization tools, many of which support geo mapping</td>
<td><a href="http://www.computerworld.com/s/article/9214755/Chart_and_image_gallery_30_free_tools_for_data_visualization_and_analysis">http://www.computerworld.com/s/article/9214755/Chart_and_image_gallery_30_free_tools_for_data_visualization_and_analysis</a></td>
</tr>
<tr>
<td>Map Types Guide</td>
<td>Guide to tools for different map types</td>
<td><a href="http://guides.library.duke.edu/vis_types">http://guides.library.duke.edu/vis_types</a></td>
</tr>
</tbody>
</table>
Acknowledgement

• Dr. Yu Zheng (MSRA), Dr. Xin Xie (MSRA)
• Prof. Shou-De Lin (National Taiwan University)
• Prof. Ee-Peng Lim (Singapore Management University)
• Prof. Jure Leskovec (Stanford University) & SNAP Datasets
• Prof. Dongman Lee (KAIST), Prof. Meeyoung Cha (KAIST)
Bonus
Bonus Recipe: Scripting

https://github.com/patompa/geovizdev/tree/master/script

Step 1: Generate overlay image and KML
Step 2: View in Google Earth
Step 3: Generate standalone images with OSM

Types: HM
Tools: Python jjguy, sethoscope heatmap, Google Earth
Key Ideas: Generate maps from command-line or API to integrate with backend or native app, no web server needed
R4 Step 1-2: Generate Overlay Image and KML and View in Google Earth

```python
f = open('sample1.coord').read().split('
')
pts = []
for line in f:
    coords = line.strip().split('	')
    if len(coords) < 2:
        continue
    pts.append((float(coords[1]), float(coords[0])))

hm = heatmap.Heatmap()
img = hm.heatmap(pts, dotsize=3)
hm.saveKML("heatmapjjguy.kml")
```

Open KML and overlay image with same name in Google Earth

For more details see:
https://github.com/patompag/geomdenv/blob/master/script/heatmapjjguy.py
http://jjguy.com/heatmap/
R4 Step 3: Generate Standalone Images with OSM

```python
python heatmap.py -r 4
    -p sample1.coord
    -o heatmapseth.png
    --height 800
    --osm
    -B 0.8
    --osm_base http://b.tile.stamen.com/toner
```

Requires OSM visualization toolkit for python:

For more details see:
http://www.sethoscope.net/heatmap/