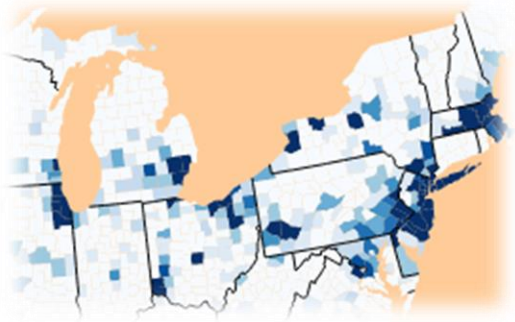
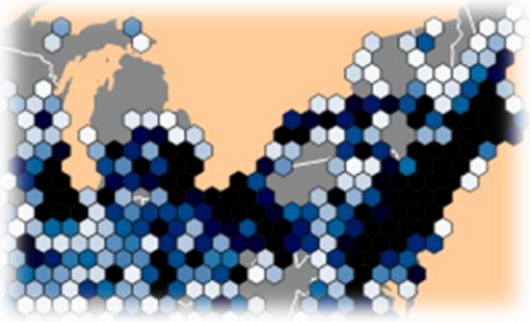
Part II: Programming Geo-Data Visualizations

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





Agenda

Overview	8min
Preparing the Data	2min
Recipe 1: Server-side Rendering	15min
Recipe 2: Data-Driven Documents	15min
Recipe 3: Visualizing Time	5 min
Coffee Break	30 min
Recipe 4: Draw-it-yourself	10min
Recipe 5: Route Visualization	5min
Bonus Recipe: Scripting	

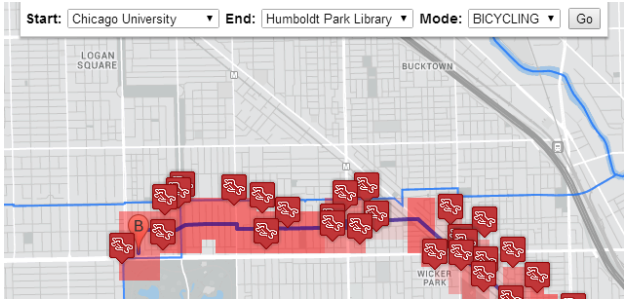
Why Program Thematic Maps?

Exploratory data analysis

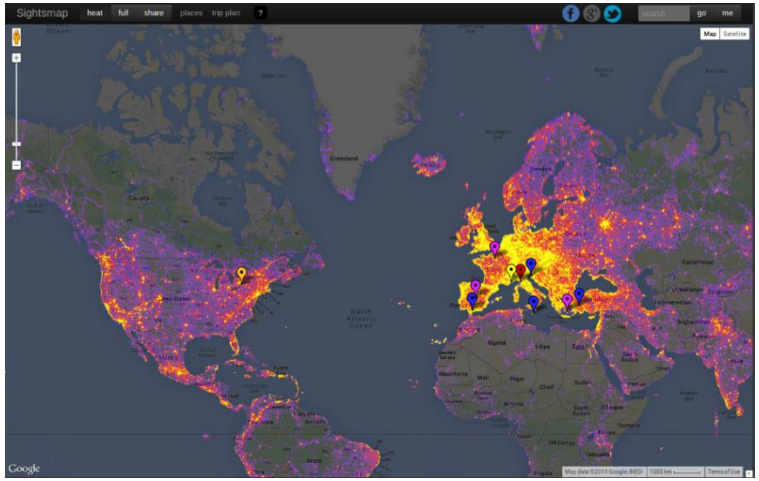
Dynamic rendering

Scalability

Interactivity



Inspiration



http://www.sightsmap.com

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

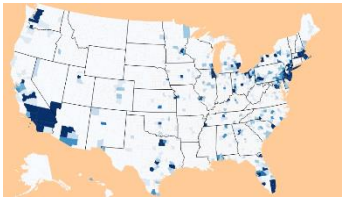
Explanatory Visualization Guides:

- http://www.edwardtufte.com/tufte/
- http://www.ted.com/talks/hans_rosling_shows_the_best_stats_you_ve_ever_seen
- http://blog.visual.ly/10-things-you-can-learn-from-the-new-york-times-data-visualizations

Approaches

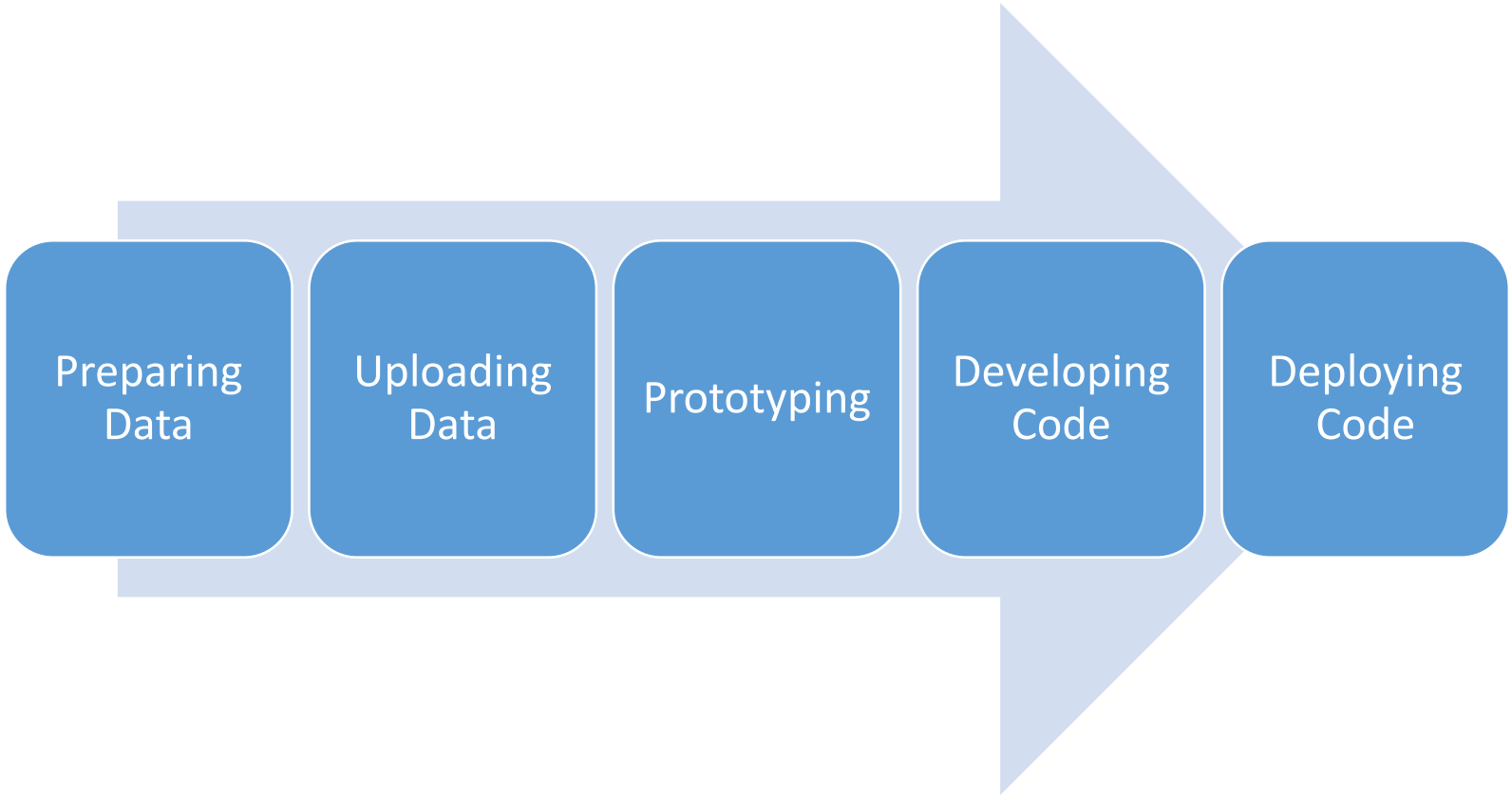
- Direct Plotting (DP) Marker positioning
  + simple, flexible
  - only handles few points, slow, point occlusion

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

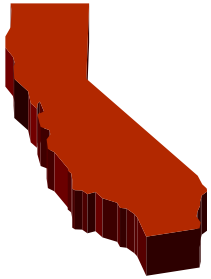


- Heatmap (HM) Radial diffusion and blending
  + discovers hotspots, no point occlusion, handles many points
  - slow, could be hard to read, artificial gradients

Programming a Thematic Map



Preparing Data

CA 

Geocoding (DP,HM)

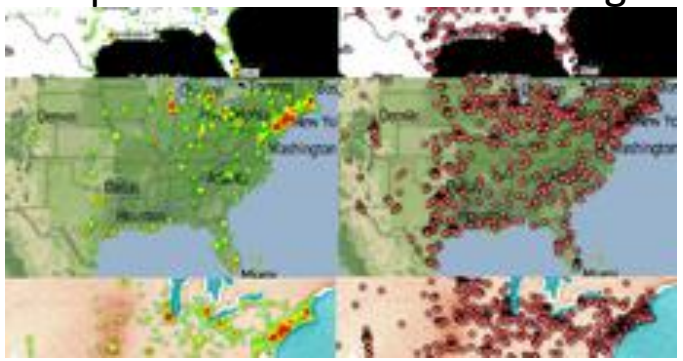
Reverse Geocoding (AA)

37.75, -122.42

- 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

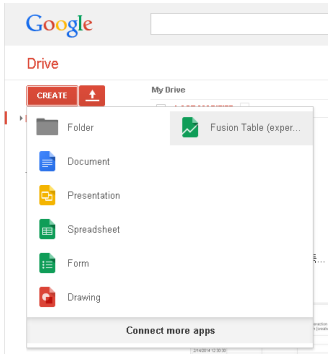
http://patompa.github.io/geovizdev/demos/fusionheat/

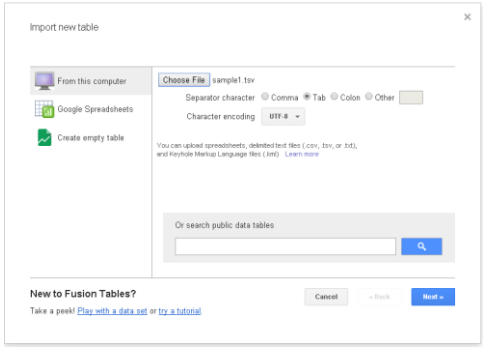
Types: DP, HM
Tools: Fusion Tables, Google Maps, Stamen Tiles
Key Ideas: Many points, Pre-render images on Server, Hosted server, No prep-work

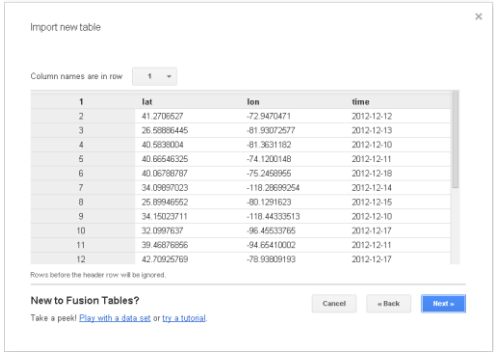
Step 1: Upload to Google Drive FusionTable

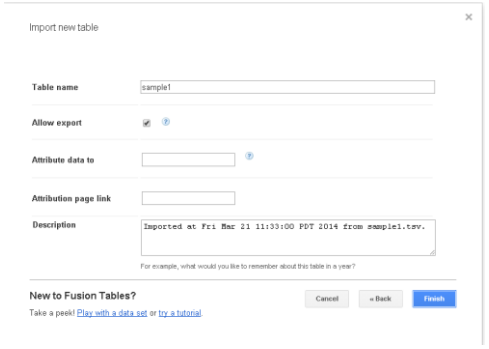
Step 2: Write Javascript with Google Maps API

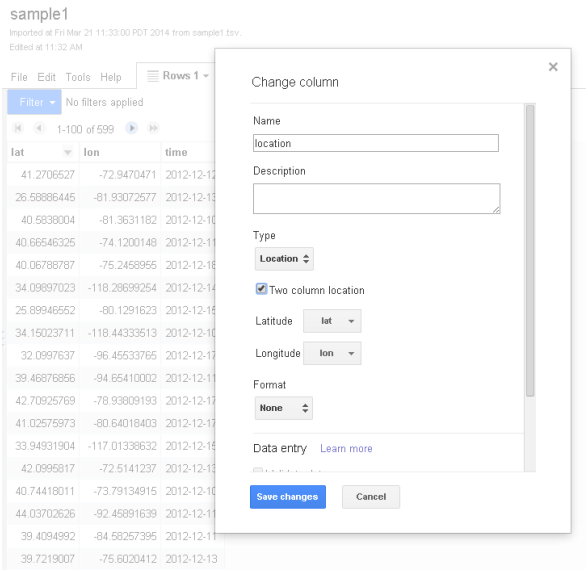
R1 Step 1: Upload to FusionTable

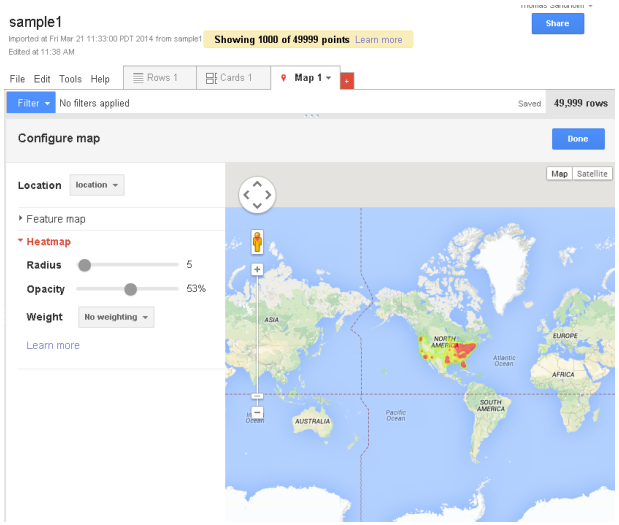


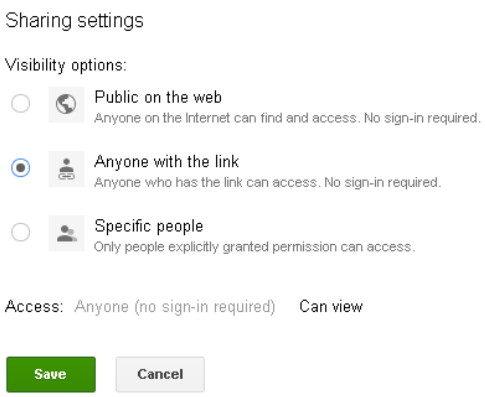










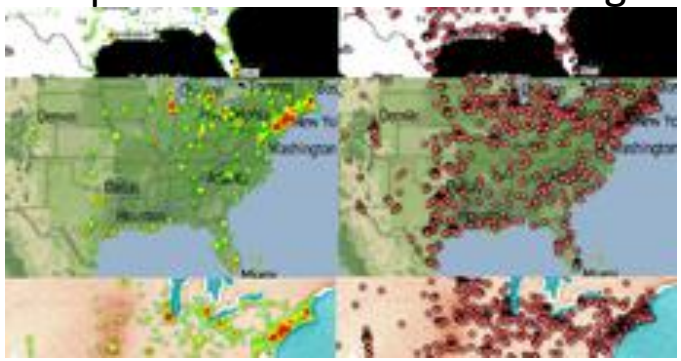


R1 Step 2: Write Javascript

```
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:
https://github.com/patompa/geovizdev/blob/master/fusionheat/index.html

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/

Types: DP, AA
Tools: D3, Tableau Public
Key Ideas: Tie data to DOM, use SVG for speed and interactivity

Step 1: Aggregate by County

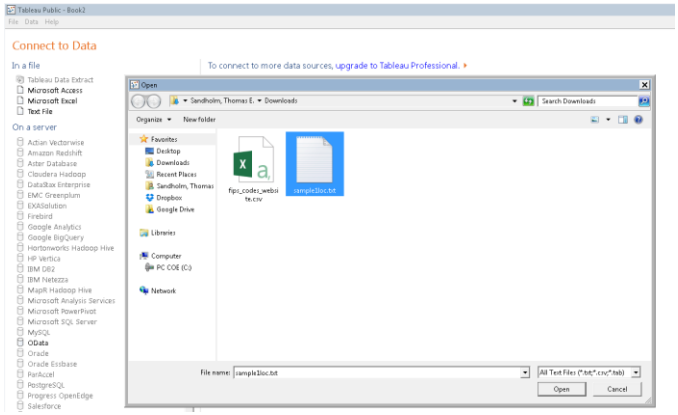
Step 2: Get TopoJSON Area polygons

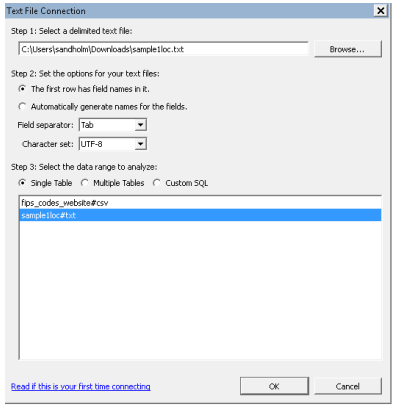
Step 3: Create Choropleth

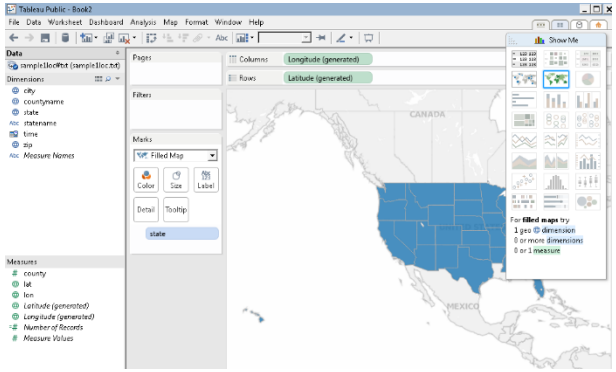
R2 Step 2-3: Aggregate by County and Get Area Polygons

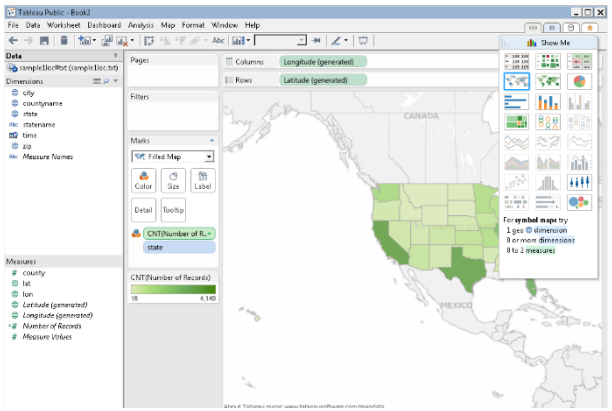
- In D3 manual aggregation and map drawing is needed (Tableau Public does this for you)
- TopoJSON, more efficient GeoJSON format used by D3 to draw maps
- US county/state available at: http://bl.ocks.org/mbostock/raw/4090846/us.json
- Area polygons may also be created from GIS tools and converted from public shape files, see: http://bost.ocks.org/mike/map/

R2 Step 3: Create Tableau Public Choropleth









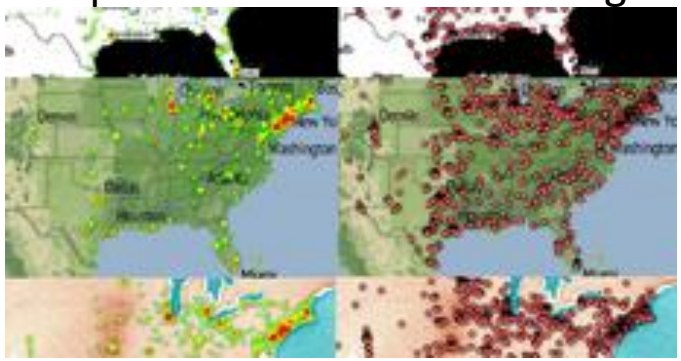
For more details see:
http://public.tableausoftware.com/views/TweetDensity/State

R2 Step 3: Create D3 Choropleth

```
queue()
    .defer(d3.json, "us.json")
    .defer(d3.tsv, "../utils/sample1loccounty.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/patompa/geovizdev/blob/master/d3/index.html
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

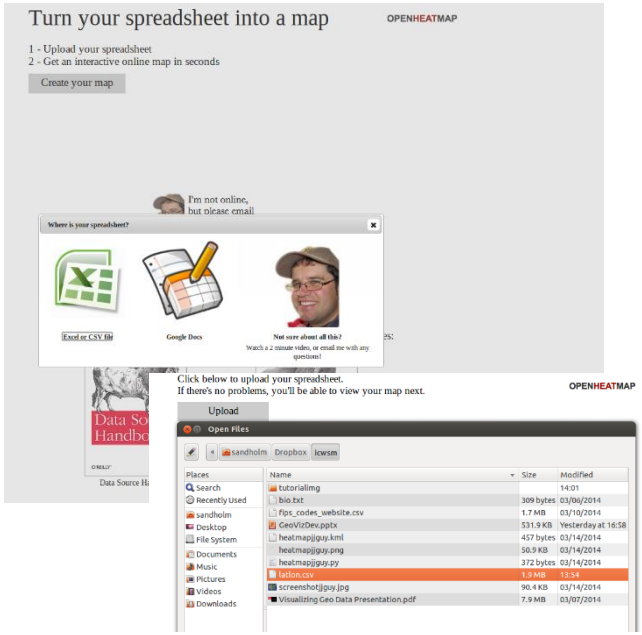
Step 2: Prototype with OHM web tool

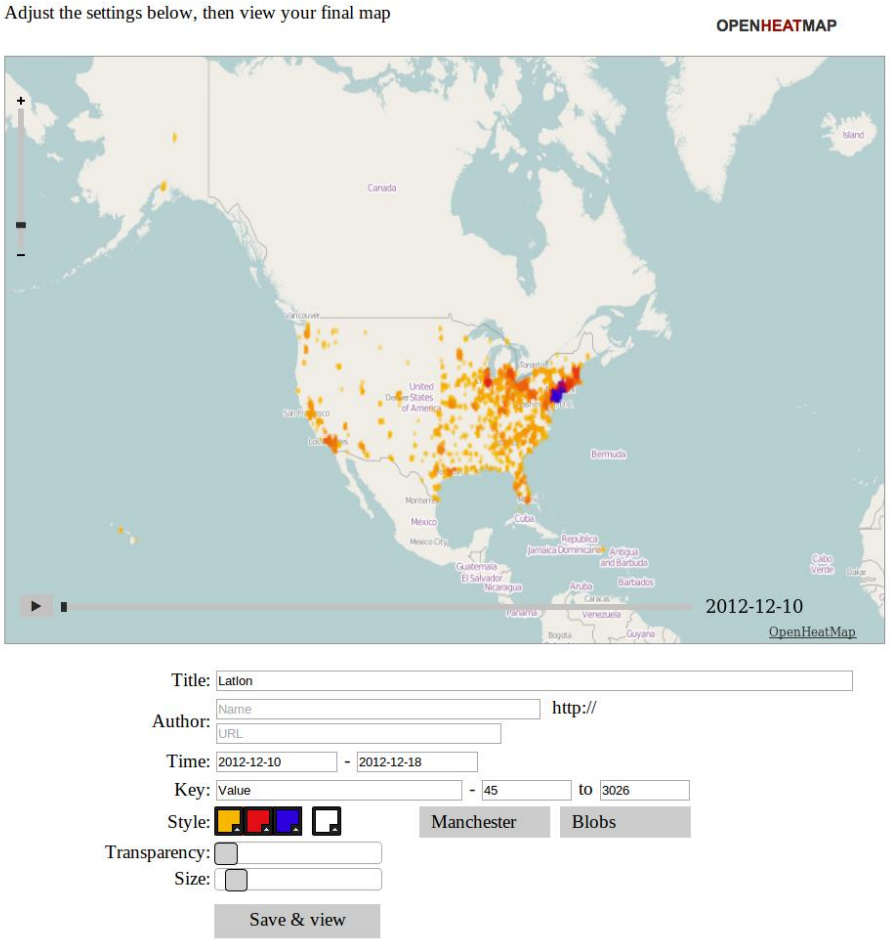
Step 3: Write OHM Javascript

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 100x100mile 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/

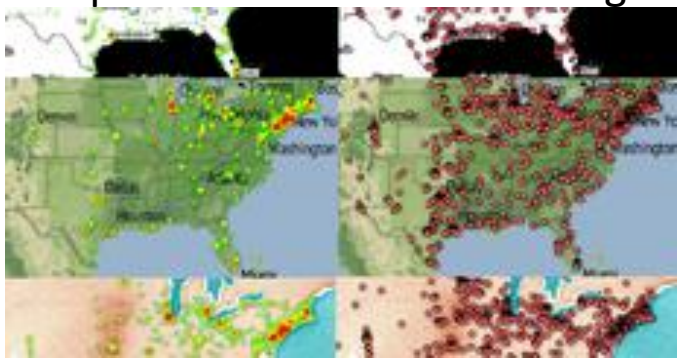
R3 Step 3: Write OHM Javascript

```
$('#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:
https://github.com/patompa/geovizdev/blob/master/ohm/heat.html

Coffee

Recipe 1: Server-side Rendering



Recipe 2: Data-driven Documents

Recipe 3: Visualizing Time

Recipe 4: Draw-It-Yourself

Recipe 5: Route Visualization

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

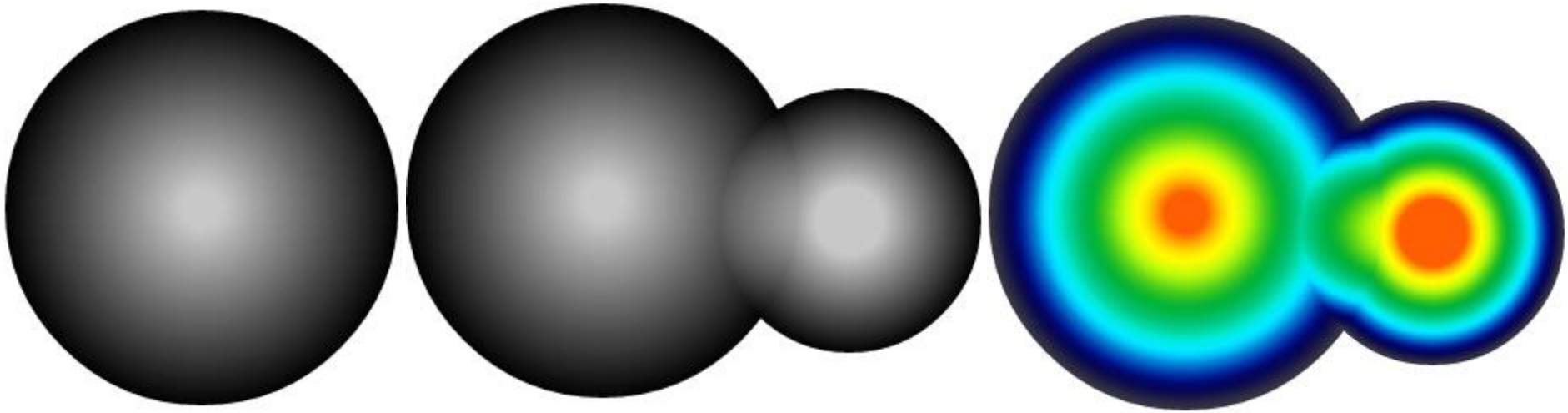
Step 2: Render heatmap

R4 Step 1: Draw D3 Map and Reuse Projection

```
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

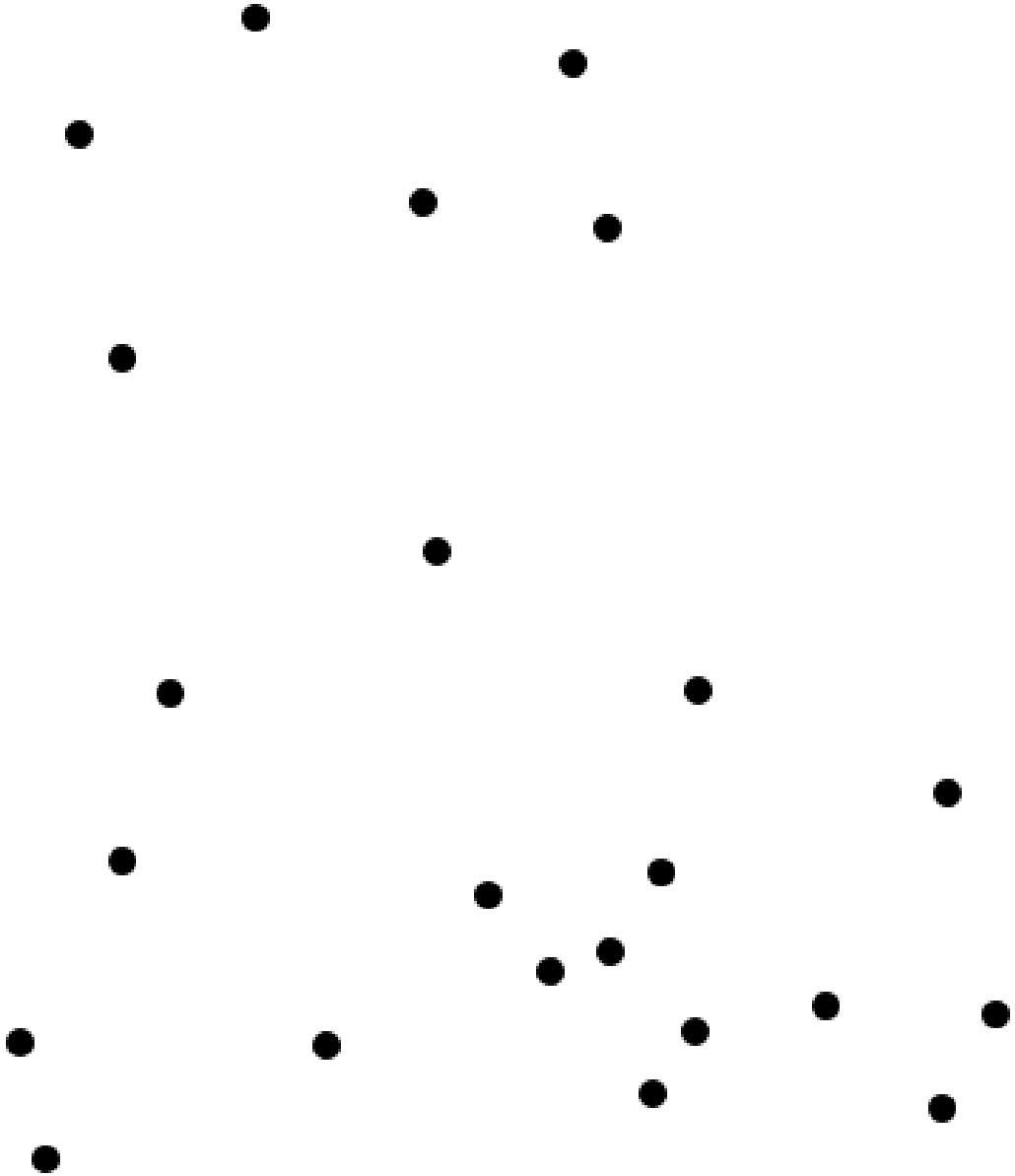


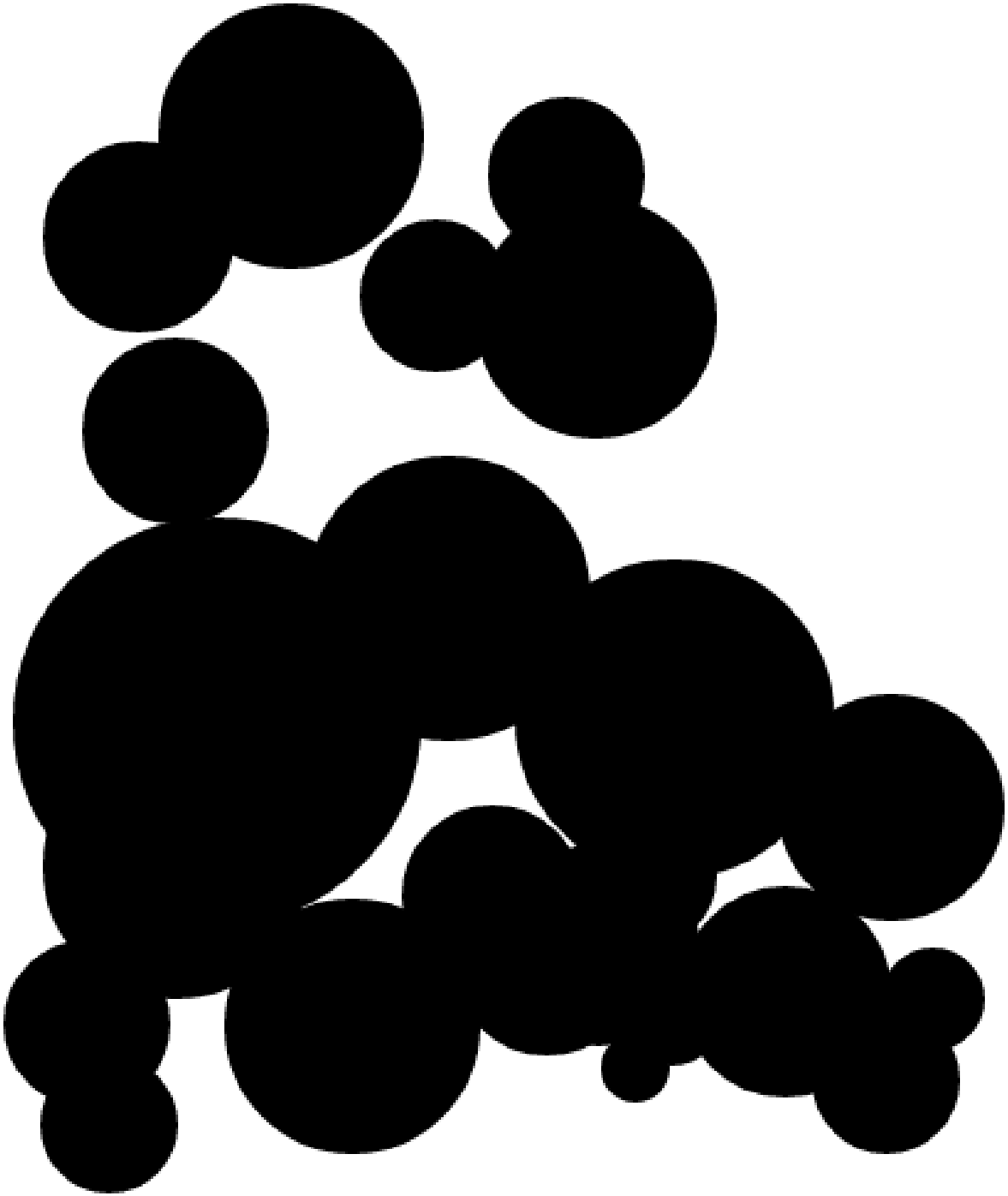
1. Draw Grayscale Circle with Radial Gradient

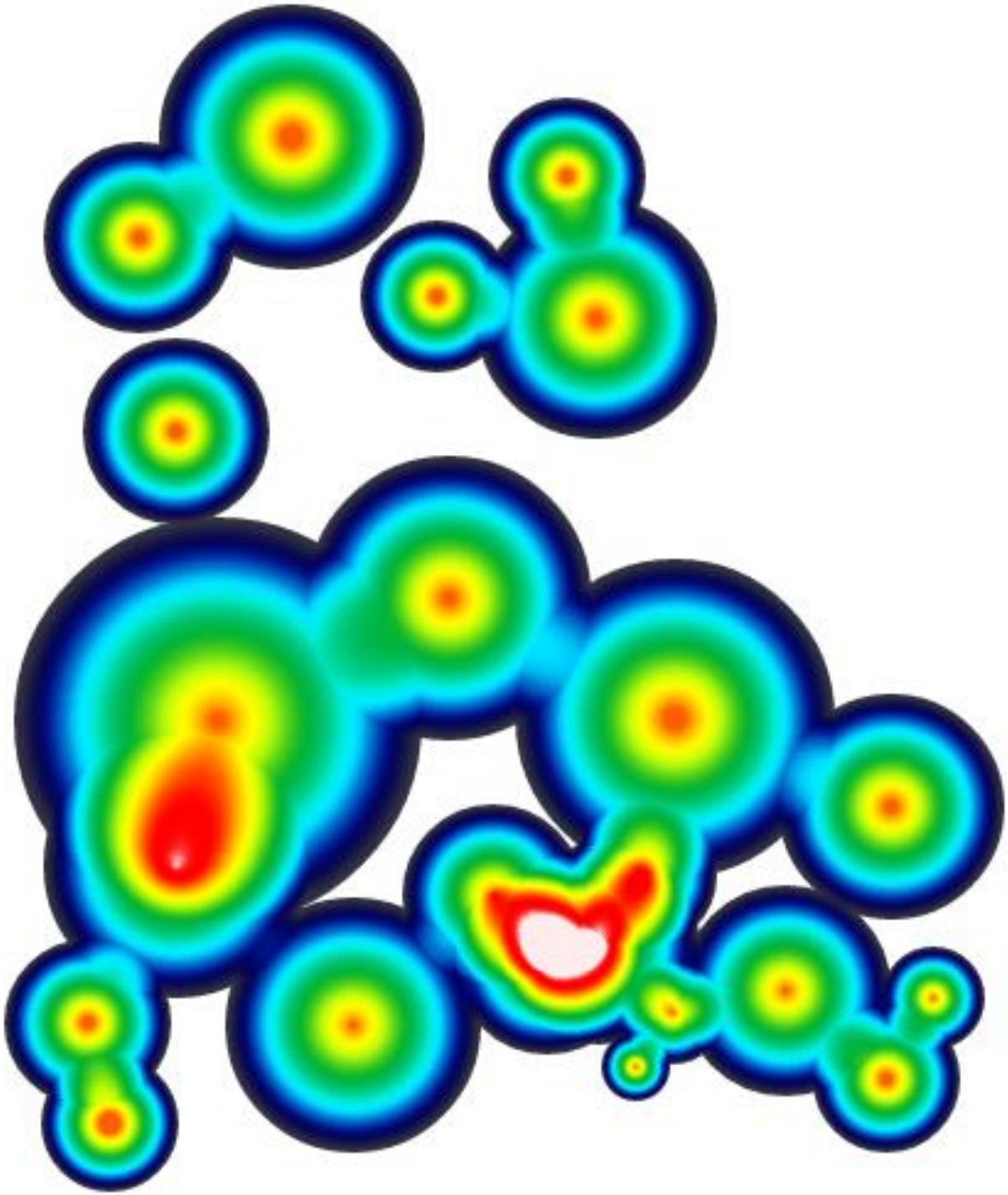
2. Blend points by adding pixel RGB values

3. Compute pixel luminance and colorize using 255-scale palette

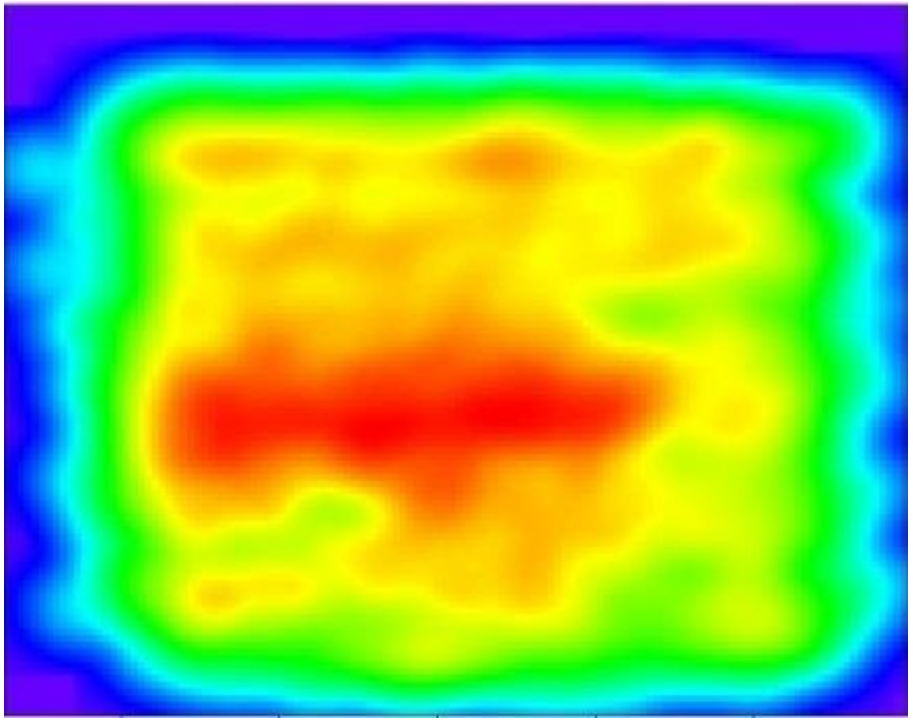
For more details see:
https://github.com/patompa/geovizdev/blob/master/canvas/geo.js



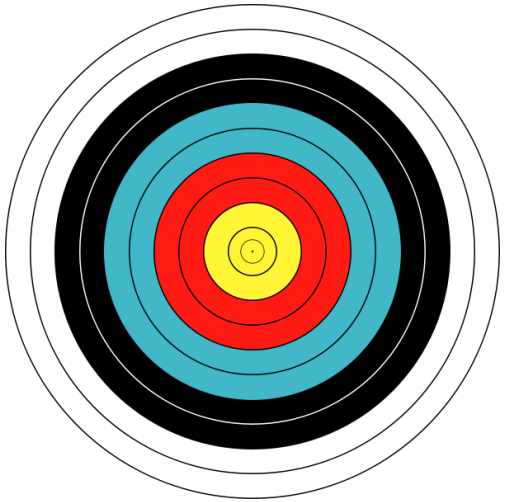




Gaussian Blur

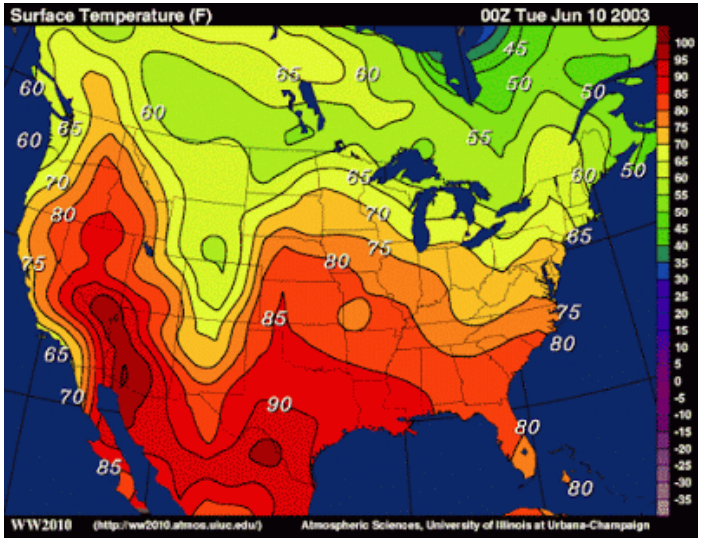


From http://finance.yendor.com/etfviz/2008/0301

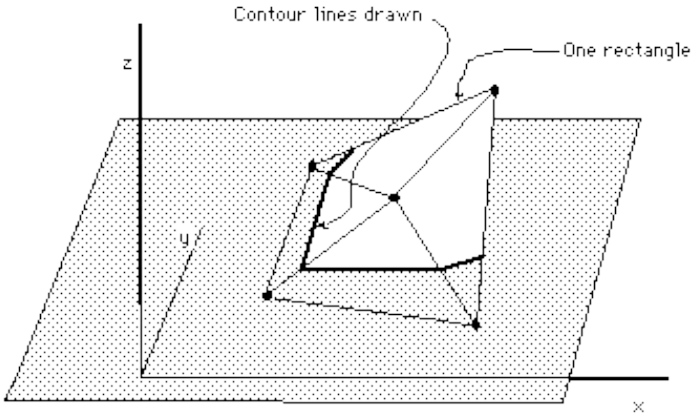


$$G(x,y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}$$

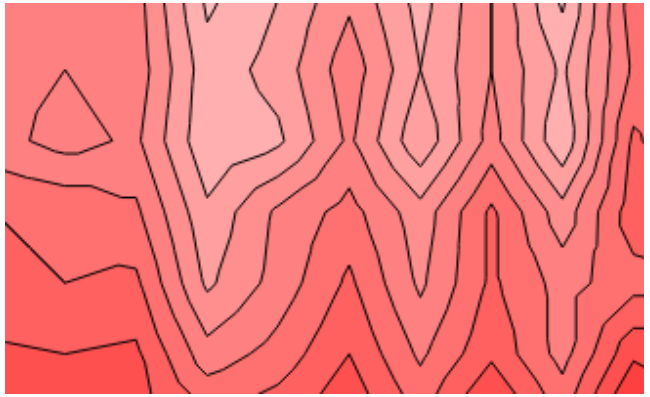
Isopleths or Contour Maps



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

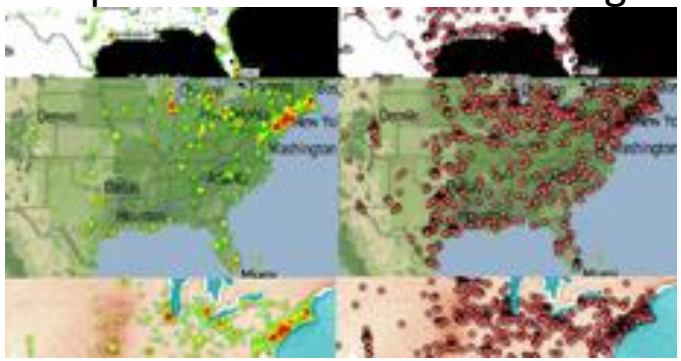


From http://paulbourke.net/papers/conrec/



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

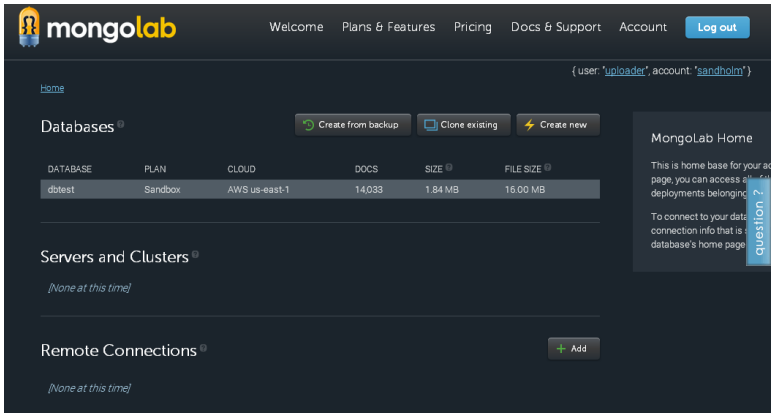
Step 1: Upload JSON to Mongolab

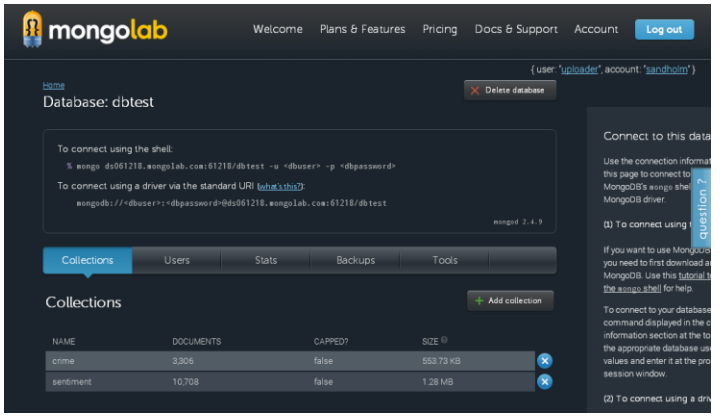
Step 2: Routebox Google Directions path

Step 3: Pull data and visualize

Based on WWW'14 Demo: http://mia.kaist.ac.kr/project/socroutes/

R5 Step 1: Upload to Mongolab





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

crime.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": -87.638676258693792}
{"lat": 41.765808860199698, "text": "12/14/2012 11:40:00 PM ROBBERY STRONGARM - NO WEAPON", "lon": -87.615813855691911}
```

sentiment.json

```
{"lat": 41.828851290000003, "lon": -87.682199550000007, "sentiment": -0.80000000000000004}
{"lat": 41.91660289, "lon": -87.687379719999996, "sentiment": -0.75}
{"lat": 41.908128220000002, "lon": -87.694693700000002, "sentiment": -0.75}
{"lat": 41.911173400000003, "lon": -87.641940059999996, "sentiment": 0.66666700000000001}
{"lat": 41.883652580000003, "lon": -87.630246880000001, "sentiment": 0.0}
{"lat": 41.973033239999999, "lon": -87.659767509999995, "sentiment": 0.0}
```

R5 Step 2: Routebox Google Directions Path

```
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();
boxes = routeBoxer.box(on_path, distance);
```

For more details see:
https://github.com/patompa/geovizdev/blob/master/route/index.html
http://google-maps-utility-library-v3.googlecode.com/svn/trunk/routeboxer/docs/examples.html

R5 Step 3: Pull Data and Visualize

```
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/
  - Aggregate, discretize and bin with care
  - Projections from 3D to 2D lie

Tool Summary

Tool	Pros	Cons
R1: Fusion Tables	Server rendering for scalability. Heatmap and DP support.	Need to convert geodata into tabular form. Very limited configuration options for heatmap rendering. Data upload through browser may be slow.
R2: D3	Large collection of pre-drawn maps. Efficient map drawing and projection. Interactivity built into the browser. Styling built into the browser.	Steep learning curve. Client side rendering may be slow.
R2: Tableau Public	Fast to prototype DP and AA maps.	No programmability. Windows only. Strange saving behavior (save to web only).
R3: Open Heat Map	Online tool to quickly visualize time evolution. Minimal programming needed due to standard column name design.	Flash based. Need to rename data columns. No heatmap without aggregation.
R4: Canvas	Easy to program. Customization unlimited.	Scaling not as flexible as with SVG.
R5: RouteBoxer	Works well with bounding box queries. Easy to visualize.	May introduce artificial areas and gradients.
R5: MongoDB	Works well with json data such as Twitter dumps.	Rate limited for commercial use.

Tool Index

Tool	Purpose	Reference
D3	Web visualization mostly focused on AA, very low level pixel-by-pixel control.	https://github.com/mbostock/d3 http://bl.ocks.org/mbostock/4060606 http://chimera.labs.oreilly.com/books/1230000000345/index.html
Google Maps	Basic Marker based overlays (DP) for Web visualizations.	https://developers.google.com/maps/documentation/javascript/
Google FusionTables	Online Table that can be accessed from Javascript and supports server rendered HM and DP.	https://developers.google.com/maps/documentation/javascript/examples/layer-fusiontables-heatmap
Google Geocharts	AA for countries and regions	https://developers.google.com/chart/interactive/docs/gallery/geochart
Stamen	OpenStreetMap based Map Tiles	http://maps.stamen.com/
Python Heatmaps (jjguy)	Google Earth Compatible Heatmaps. Based on Gheat.	http://jjguy.com/heatmap/ https://code.google.com/p/gheat/
Python Heatmaps (sethoscope)	OpenStreetMap based Heatmaps	http://www.sethoscope.net/heatmap/
Mongolab	Like Fusion Tables but for a json database instead of a table	https://mongolab.com/welcome/
Tableau Public	Similar to D3 in functionality but UI based. County, State and City AA.	http://www.tableausoftware.com/public/

Tool Index Continued

Tool	Purpose	Reference
Open Heat Map	Quick HeatMaps with web customization and API. Basic support for time animation.	http://www.openheatmap.com/
Open Street Map Visualization Toolkit for Python	OpenStreetMap tile based visualizations in python, used by stethoscope heatmaps, see above	http://cbick.github.io/osmviz/html/index.html
RouteBoxer	Computes boxes around routes using google directions api to simplify area aggregation and lookup	http://google-maps-utility-library-v3.googlecode.com/svn/trunk/routeboxer/docs/examples.html
Google Directions API	Turn by Turn direction customization on the Web	https://developers.google.com/maps/documentation/directions/
HTML5 Canvas	Powerful 2D drawing directly in browser	http://diveintohtml5.info/canvas.html
Google Earth	Allows KLM aligned image overlays, to fit image heatmaps to geo maps	http://www.google.com/earth/
Visualization Tool Guide	Review of 30+ visualization tools, many of which support geo mapping	http://www.computerworld.com/s/article/9214755/Chart_and_image_gallery_30_free_tools_for_data_visualization_and_analysis
Map Types Guide	Guide to tools for different map types	http://guides.library.duke.edu/vis_types

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)

Microsoft®
Research
微软亚洲研究院

Bonus

Bonus Recipe: Scripting

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

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

Step 1: Generate overlay image and KML

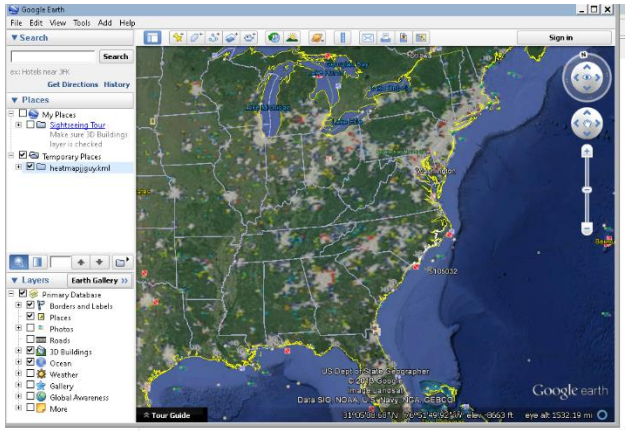
Step 2: View in Google Earth

Step 3: Generate standalone images with OSM

R4 Step 1-2: Generate Overlay Image and KML and View in Google Earth

```
f= open('sample1.coord').read().split('\n')
pts = []
for line in f:
   coords = line.strip().split('\t')
   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/patompa/geovizdev/blob/master/script/heatmapjjguy.py
http://jjguy.com/heatmap/

R4 Step 3: Generate Standalone Images with OSM

```
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:
http://cbick.github.io/osmviz/html/index.html

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