DOI: 10.14714/CP76.1252

Census Mapping Mashup

Paul Hunt
University of Nebraska Omaha
phunt@unomaha.edu

By mandate, the United States Census Bureau compiles and distributes data on the American population. Open data initiatives have made it possible for users to access and analyze data with simple web-based tools. A new method for requesting data from the Census Bureau is described here, along with two different mapping mashups. Using the technology described in this article, a simple web mapping interface could unlock vast amounts of available data for user exploration.

INTRODUCTION

The United States Census Bureau collects and maintains a large, widely-used collection of data on general demographic, social, housing, and economic characteristics. Recently, they made their data available through an Application Programming Interface (API) (US Census Bureau 2014). Their API is a mechanism to access Census data through a set of web technologies referred to as AJAX (Asynchronous JavaScript and XML), which facilitate the continuous sending and receiving of data between client and server environments (Powell 2008). The main advantage of using the API is that the Census data are available for use without the need to store them on a local computer: they remain in the cloud.

JavaScript is the main programming language used with APIs; most are completely written in JavaScript. Interpreted by the web browser, JavaScript is also the most widely used programming language for web development (Raasch 2013). As an interpreted computer language, there are no software packages that need to be installed, nor any special server-side setup. It is an efficient and simple solution for web-based application development and hosting.

The Esri JavaScript API (Esri 2014a) allows developers to use web services based on spatial data servers that implement their software. The Esri software packages and API are popular with government enterprise GIS systems. Data Driven Documents (D3) is another powerful set of JavaScript functions used to visualize large datasets. Both Esri and D3 provide JavaScript-based environments for creating mapping *mashups*, which are the pulling together of various online resources (Batty, et al. 2010).

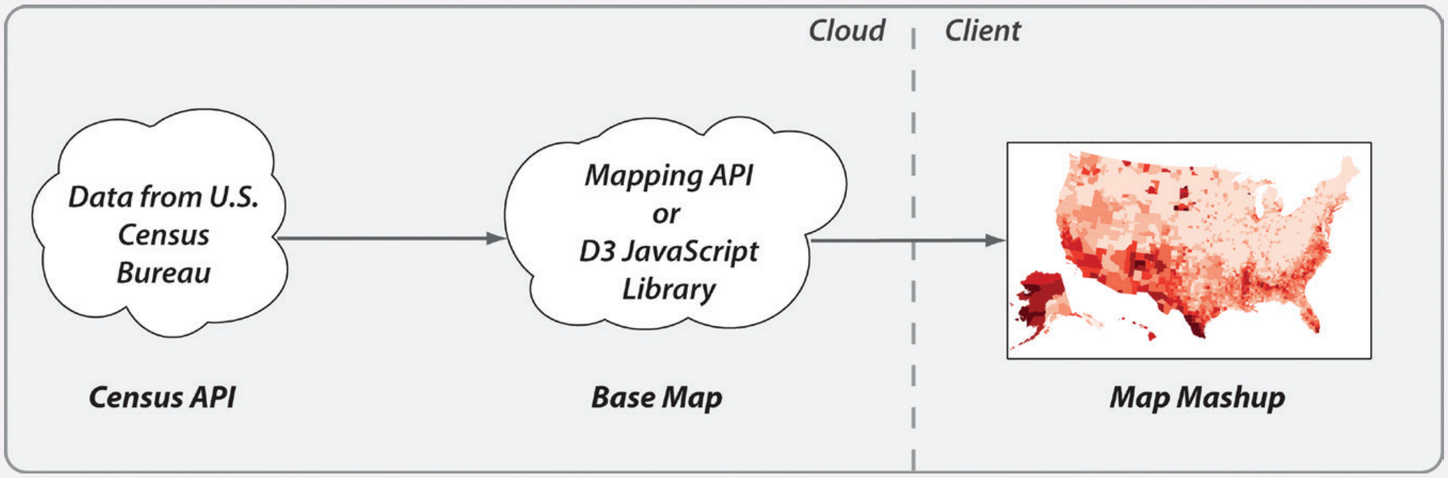

Cloud
Client
Data from U.S. Census Bureau
Mapping API or D3 JavaScript Library
Census API
Base Map
Map Mashup

Figure 1: Representation of AJAX and JavaScript consuming web resources and a mashup of the attribute and spatial data in the client.

Two mashup methods are demonstrated here for mapping Census data obtained from the Census Bureau API. In the first, the Esri JavaScript API is used to obtain a basemap from the US Census Bureau TIGERweb spatial data server (see Figure 1). In the second method, the base layer for D3 is provided in GeoJSON or TopoJSON format, which are compatible with JavaScript.

BACKGROUND

For mapping, most APIs are based on a technology called REST (Representational State Transfer) web services, which manipulate representations of web resources using a uniform set of stateless operations (Fielding 2000). One of the most popular map service architectures is the tiled web map service. This type of REST service involves a series of predefined static map tiles, produced for various scales, that populate the web map based on user interaction. There are additional REST mapping services, both raster and vector, that are the primary technology behind the plethora of Multi-Scale Panable maps available on the web (Peterson 2014). These mapping APIs provide the base layer of most map mashups.

There are alternatives for creating web map mashups. The D3 JavaScript Library can render a projected, SVG map element (base layer) that can be used to visualize spatial data (Cuesta 2013). D3 is designed for the creation of dynamic visualizations. In addition to mapping APIs, there are also APIs that revolve around accessing data. Data APIs focus on delivering specific queried data or streaming updated data. In a map mashup, these attribute (thematic) data sources, referred to as operational layers, are overlaid on top of the base layers.

CENSUS DATA API

As a product of the US Government, Census data have always been available free of charge. In the Internet era, there have been two main ways to access the hundreds of tables and thousands of spatial data elements derived from the Census. First, there are web-based query and extraction methods, the latest being the American Factfinder. Secondly, there is direct FTP access to the data that can then be processed locally within a relational database management system.

To promote access and innovation, the Census Bureau released an API in 2012 that would allow users to access data through properly formatted HTTP requests. This allows for AJAX methods to request and use data on-the-fly within a mashup environment. The Census API opens the door for alternative methods for developing mapping applications.

To make a data request, you must have a properly formatted HTTP string. This string begins with the Census API website, "api.census.gov/data." Next, the dataset you are querying is specified (i.e., Census 1990, 2000, 2010 or ACS aggregate datasets). After this, you must provide your key, which is required to access the Census API and can be obtained by completing a short form at www.census.gov/developers/tos/key_request.html. Finally, the combination of variables and the spatial units are specified. Queries can be made directly in your web browser by pasting into your address bar the example requests listed in Example 1 (you will need to use your own key provided by the Census). A partial output of the first request is shown in Example 2: a two-dimensional array of data in the JSON (JavaScript Object Notation) format. The first row provides column names and subsequent rows contain the data values. The data in this array can then be mapped.

The Census API allows for up to 50 variables to be queried in a single request; a series of requests can unlock a vast amount of Census data for user mapping and analysis. In order to implement the Census API in a JavaScript mashup, the jQuery library is needed. JQuery is a free and widely adopted JavaScript library that has the built-in functions necessary for accomplishing common AJAX requests (Powell 2008).

1) URL for 2010 Census SF1 total population and name by for all states:

http://api.census.gov/data/2010/sf1?key=b48301d897146e8f8efd9bef3c6eb1fcb864cf&get=P0010001,NAME&-for=state:*

2) URL for ACS 2010 5 Year data for Total Population for California and New York:

http://api.census.gov/data/2010/acs5?key=b48301d897146e8f8efd9bef3c6eb1fcb-864cf&get=B02001_001E,NAME&for=state:06,36

3) URL for ACS 2011 5 Year data for Gross Rent as a % of Household Income, 10.0 to 14.9 percent for all counties in CA:

http://api.census.gov/data/2011/acs5?key=b48301d897146e8f8efd9bef3c6eb1fcb864cf&get=B25070_003E,NAME&for=county:*&in=state:06

4) URL for 2010 Census SF1 white population of 12 year olds in Alabama:

http://api.census.gov/data/2010/sf1?key=[user key]&get=PCT012A015,PCT012A119&for=state:01

Example 1: Example Census API requests.

REFORMATTING

Before the data can be mapped, they must first be properly formatted. The two-dimensional array has certain limitations and is more usable if it is reformatted as key-and-value paired objects. In other words, we need to restructure the data from an array of individual elements into an array of record-like objects. The function shown in Example 3 produces the output shown in Example 4.

In this format, the data can be accessed more efficiently for database operations within a coding environment. As a two-dimensional array, the data would have to be referenced as a numeric `[row]`, `[column]` of table elements. After reformatting, data items can be referenced by name in a `{key: value}` pair and more easily mapped using JavaScript.

```
[ [ "P0010001" , "NAME" , "state" ],
[ "710231" , "Alaska" , "02" ],
[ "4779736" , "Alabama" , "01" ],
[ "2915918" , "Arkansas" , "05" ],
[ "6392017" , "Arizona" , "04" ],
[ "37253956" , "California" , "06" ], ...]
```

Example 2: Results from the first request. This query has returned the total population (P0010001), state name (NAME), and Federal Information Processing Standards (FIPS) code (state). The FIPS code uniquely identifies the spatial unit and is used for joining the attribute data to the spatial components for mapping.

MAPPING

In order to map the Census data, we need to acquire a basemap layer. After this, we dynamically join the reformatted Census data to their corresponding spatial counterparts as attributes. The map symbology (such as a choropleth) is then made based on those joined Census attributes.

MAPPING WITH THE ESRI JAVASCRIPT API

The Census' TIGERweb spatial data platform is based on the Esri ArcGIS Server software, which serves Open Geospatial Consortium (OGC) compliant spatial data as web services. Since the data are OGC compliant, a mash-up could be done by using the OpenLayers API, an open source JavaScript API used for web mapping and consuming spatial data services. However, since the TIGERweb services are natively using Esri software, it is simpler to use Esri's freely-provided API.

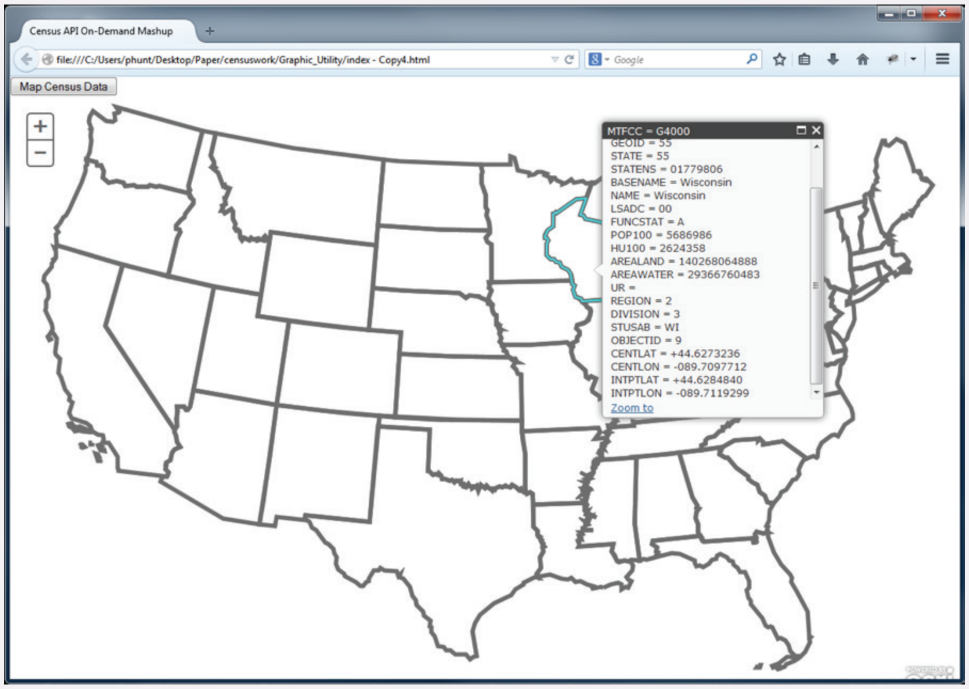
To demonstrate the use of dual web services, the example here has two parts. First, the page is loaded with spatial

```
function getCensusData(){
  //JQuery AJAX function getting data from the Census API
  //and call a return function processing the Census JSON object in ‘data’ variable
  $.getJSON(“http://api.census.gov/data/2010/sf1?key=[user key]&get=P0010001,P0050010,NAME&for=state:*”,
  function(data){
      var keys = data[0]; //extract the first row of the returned 2d array that are the column headers
      var values = data; //copy the array
      values.splice(0,1); //delete the first row of headers in the copied array
      arrayCensus = []; //create a new array to store the formatted object outputs
      //nested loops combining the column header with appropriate values as {key:value} pair objects
      for(var i = 0 ; i < values.length; i++){
            var obj = {};
            for(var j = 0 ; j < values[i].length; j++){
                  obj[keys[j]] = values[i][j];
            }
            arrayCensus.push(obj);
     }
  });
}
```

Example 3: Sample function that requests data using the Census API and then reformats the results into a usable array of objects that can be mapped.

```
[ { "P0010001" : "4779736" , "NAME" : "Alabama" , "state" : "01" },
{ "P0010001" : "710231" , "NAME" : "Alaska" , "state" : "02" },
{ "P0010001" : "6392017" , "NAME" : "Arizona" , "state" : "04" },
{ "P0010001" : "2915918" , "NAME" : "Arkansas" , "state" : "05" },
{ "P0010001" : "37253956" , "NAME" : "California" , "state" : "06" }, ...]
```

Example 4: The reformatted Census API request, now an array of objects and ready to be used as a mashup with either the Esri API or D3.

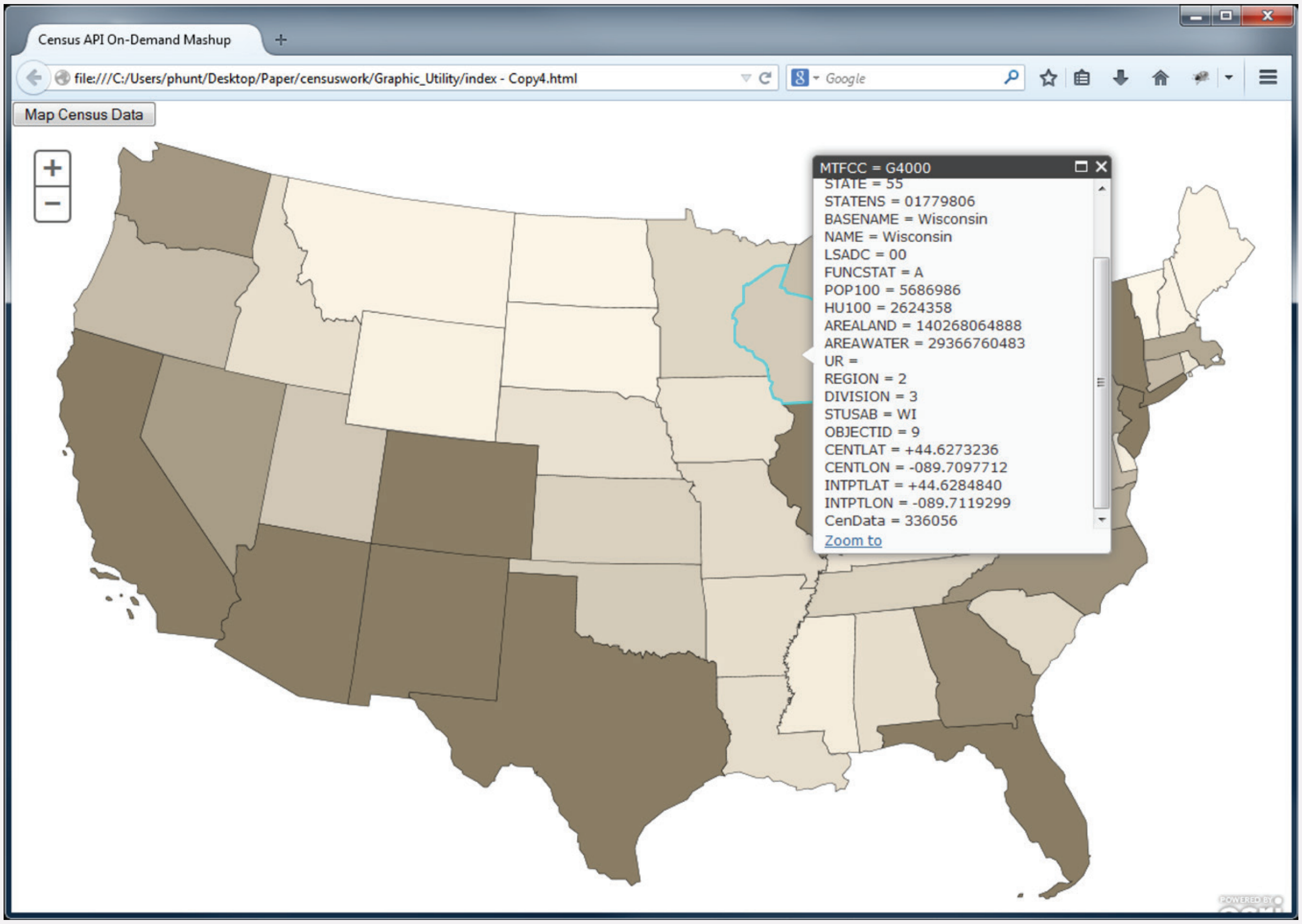

Census API On-Demand Mashup
file:///C:/Users/phunt/Desktop/Paper/censuswork/Graphic_Utility/index - Copy4.html
Google
Map Census Data
Zoom to

MTFCC = G4000
GEOID = 55
STATE = 55
STATENS = 01779806
BASENAME = Wisconsin
NAME = Wisconsin
LSADC = 00
FUNCSTAT = A
POP100 = 5686986
HU100 = 2624358
AREALAND = 140268064888
AREAWATER = 29366760483
UR =
REGION = 2
DIVISION = 3
STUSAB = WI
OBJECTID = 9
CENTLAT = +44.6273236
CENTLON = -089.7097712
INTPTLAT = +44.6284840
INTPTLON = -089.7119299

Figure 2: Initial loading of web page with the TIGERweb services.

data—in this case, data for the continental 48 states—derived from the TIGERweb web mapping services that is being consumed locally. In Figure 2, a feature has been selected to show the default popup window included with the Esri API. Second, there is a button marked “Map Census Data” in the top left corner of the page. Clicking it invokes the Census Data API AJAX request; when this button is pressed, the Census data are requested, reformatted, spatially joined, and symbolized. The resultant map is shown in Figure 3.

Example 5 shows the combination of Census and Esri code needed. Notice in Figure 3 that there is an additional


Census API On-Demand Mashup
file:///C:/Users/phunt/Desktop/Paper/censuswork/Graphic_Utility/index - Copy4.html
Google
Map Census Data
Zoom to

```
MTFCC = G4000
STATE = 55
STATENS = 01779806
BASENAME = Wisconsin
NAME = Wisconsin
LSADC = 00
FUNCSTAT = A
POP100 = 5686986
HU100 = 2624358
AREALAND = 140268064888
AREAWATER = 29366760483
UR =
REGION = 2
DIVISION = 3
STUSAB = WI
OBJECTID = 9
CENTLAT = +44.6273236
CENTLON = -089.7097712
INTPTLAT = +44.6284840
INTPTLON = -089.7119299
CenData = 336056
```

Figure 3: Map showing Total Hispanic Population (calculated CenData attribute) after Census data request, reformat, join, and symbolization.

```
<html>
   <head>
      <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
      <meta name="viewport" content="initial-scale=1, maximum-scale=1,user-scalable=no">
      <title> Census API On-Demand Mashup </title>
      <link rel="stylesheet" href="http://js.arcgis.com/3.8/js/esri/css/esri.css">
      //link to JQuery library
      <script src="http://ajax.googleapis.com/ajax/libs/jquery/2.1.0/jquery.min.js"></script>
      <script src="getCensusData.js"></script>//link to Code 3 example
      <style> html, body, #map {height: 100%; width: 100%; margin: 0; padding: 0;} </style>
      <script src="http://js.arcgis.com/3.8/"></script>//link to ESRI API
      <script>
         require ([
            "dojo/parser", "dojo/dom-construct", "dojo/json", "dojo/_base/array", "dojo/_base/connect", "dojo/number",
            "esri/map", "esri/layers/FeatureLayer", "esri/geometry/Extent", "esri/InfoTemplate" ,
            "esri/renderers/SimpleRenderer", "dojo/_base/Color", "esri/symbols/SimpleFillSymbol" ,
            "esri/symbols/SimpleLineSymbol", "dojo/domReady!"
         ], function (
            parser, domConstruct, JSON, arr, conn, number, Map, FeatureLayer, Extent, InfoTemplate, SimpleRenderer,
            Color, SimpleFillSymbol, SimpleLineSymbol
```

Example 5: A mashup using both the Census and Esri JavaScript APIs to map census data. Continued next page.

```
        ) {var bounds=new Extent ({//Set spatial extent of map and coordinate system
          "xmin" : -2473966, "ymin" : -2231235, "xmax" : 2421565, "ymax" : 1922548 ,
          "spatialReference":{"wkid" : 102004}
        });
        var map=new Map("map", {extent : bounds, center : [-96, 41]});
        //Define States Layer from the Census TIGERweb rest Services
        var urlSTE=
          "http://tigerweb.geo.census.gov/arcgis/rest/services/Census2010/tigerWMS_Census2010/MapServer/88";
        var templateSTE=new esri.InfoTemplate ("${*}");
        var CensusSTE=new FeatureLayer(urlSTE, {
          mode : FeatureLayer.MODE_ONDEMAND, outFields:["*"], infoTemplate : templateSTE ,
        });
        map.addLayer (CensusSTE); //add layers to map
        $ ('#submit').click (function(){
          getCensusData ();//get and format census data with Example 3 example
          alert ("Getting census data....");
          DemMin=new Number; DemMax=new Number;
          //loop through each census object and set min and max value
          arr.forEach(arrayCensus, function(C){
            if(C.P0050010 < DemMin){DemMin=C.P0050010 ;}
            if(C.P0050010 > DemMax){DemMax=C.P0050010 ;}
            //for each census object,loop through spatial object and join data, add attribute called CenData
            //CenData represents the Census Variable P0050010 (Total Hispanic Population)
            arr.forEach(CensusSTE.graphics, function(G){
              if(G.attributes.GEOID==C.state){
                G.attributes.CenData=C.P0050010 ;
                return false ;
              }
            });
          });
          var renderer=new SimpleRenderer
            (new SimpleFillSymbol().setOutline(new SimpleLineSymbol().setWidth (0.5)));
              renderer.setColorInfo ({//set classification values and symbology color information
              field : "CenData", minDataValue : DemMin, maxDataValue : DemMax,
              colors : [new Color ([253, 245, 230]), new Color ([139, 126, 102])]
            });
          CensusSTE.setRenderer(renderer);
          CensusSTE.redraw ();
        });
      });
    </script>
  </head>
  <body> <div id="map"><button id="submit"> Map Census Data </button> </div> </body>
  </html>
```

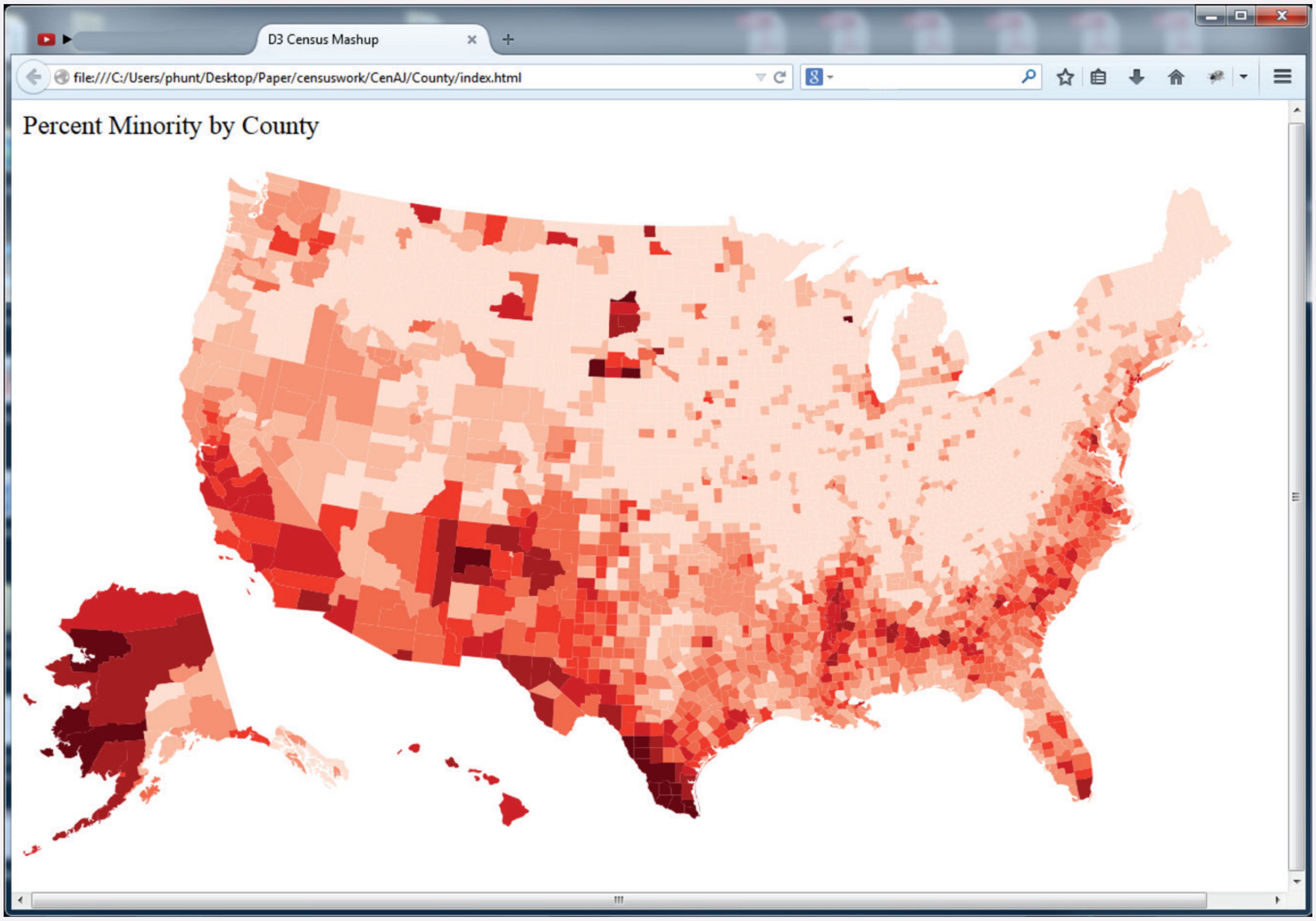
Example 5, continued.

attribute at the bottom named CenData in the popup window. The feature symbology is a gradient between two colors assigned to the minimum and maximum CenData attribute values (outlined in the code).

MAPPING WITH THE D3 JAVASCRIPT LIBRARY

The D3/Census Bureau mashup is essentially the same as above, with two exceptions: the D3 JavaScript library requires data defined in either the GeoJSON or TopoJSON formats, and D3 has its own AJAX functionality—the jQuery library is not needed. Example 6 outlines the process to render the map in Figure 4. To demonstrate dynamic field calculation with Census data, percent values are calculated on-the-fly.

D3 has more robust visualization capabilities because of its unique data handling functions. For example, the quantize classification utilized in Example 6 removes outliers in the data range that could skew the color values of the map symbology. Furthermore, we are using ColorBrewer (www.colorbrewer.org)'s 9-class "Reds" scheme, which is integrated into D3.


D3 Census Mashup
file:///C:/Users/phunt/Desktop/Paper/censuswork/CenAJ/County/index.html

Percent Minority by County

Figure 4: A mashup map produced with the Census Bureau API and the D3 JavaScript library displaying percent-minority population.

```
<html lang ="en" >
   <head>
      <title> D3 Census Mashup </title>
      <script type ="text/javascript" src ="d3/d3.v3.js" ></script> //link to D3 library
      //link to FormatCounty(), similiar to Example 3
      <script type ="text/javascript" src ="d3/Paul_minMaster.js" ></script>
   </head>
<body> <big> Percent Minority by County
   <script type ="text/javascript">
      var w=1000; var h=700 ;
      var projection=d3.geo.albersUsa().translate([w/2, h/2]) //Define map projection
      var path=d3.geo.path().projection(projection);
      var color=d3.scale.quantize() //Define quantize scale to sort data values into classes of color
         //Colors from ColorBrewer.js, included in the D3 download
         .range(["#fff5f0", "#fee0d2", "#fcbba1", "#fc9272", "#fb6a4a",
            "#ef3b2c", "#cb181d", "#a50f15", "#67000d"]);
      var svg=d3.select("body"). append("svg"). attr("width", w).attr("height", h); //Create SVG element
      //ajax the Census data - P0010001=Population, P0050003=Non Hispanic White Population
      d3.json("http://api.census.gov/data/2010/sf1?key=[userkey]&get=P0010001,P0050003,NAME&for=county:* ",
         function(data) {
            //call function to format data into array of objects like in Example 3
```

Example 6: Code to combine Census API with the D3 JavaScript library. Continued next page.

```
          var arrayCensus=FormatCounty(data);
          //loop through census data; create and calculate new normalized variable PerMin(percent minority)
          for(var i =0 ; i < arrayCensus.length ; i ++){
             arrayCensus[i].PerMin=1 -(arrayCensus[i].P0050003 / arrayCensus[i].P0010001);
          }
          //loop through dataset and calculate input domain min/max value for Quantize color scale
          var min=parseFloat(arrayCensus[0].PerMin); var max=parseFloat(arrayCensus[0].PerMin);
          for(var i=0 ; i < arrayCensus.length ; i ++) {
             pop=parseFloat(arrayCensus[i]. PerMin);
             if(pop < min) min=pop ;} if(pop > max){max=pop ;}
          }
          color.domain([min , max]); //Set the range of values for Quantize classification
          d3.json("us-counties.json", function(json) { //Load in GeoJSON data
             //Merge census data and GeoJSON; Loop through once for each data value
             for(var i=0 ; i < data.length ; i ++) {
                var dataCnty=arrayCensus[i].GEOID;
                var dataValue=parseFloat(arrayCensus[i].PerMin); //Grab data value, and convert from string to float
                //for each census data value loop through counties, find the corresponding county inside the GeoJSON
                for(var j=0; j < json.features.length ; j ++) {
                   var jsonCnty=json.features[j].id;
                   if(dataCnty == jsonCnty) {
                         json.features[j].properties.value=dataValue ; //Copy the data value into the GeoJSON
                         break ;
             } } }
                svg.selectAll("path") //Bind data and SVG, create one path per GeoJSON feature
             .data(json.features)
             .enter()
             .append("path")
             .attr("d" , path)
             .style("fill", function(d) {
                   //Use defined 'color()' Quantize scale to symboloze the fill based on census data value
                   var value=d.properties.value ;
                   if(value){return color(value);}
                   else{return "#ccc" ;}
      });   });   });
      </script>
   </body>
</html>
```

Example 6, continued.

SUMMARY

A VARIETY OF APIs and JavaScript can be used to access Census data for on-the-fly mapping. With some additional work, an interface that allows the user to specify Census attributes, color scheme, classification, and spatial units (i.e., state, county, tract, etc.) could be implemented. This would unlock the potential of the Census data for spatial analysis.

Furthermore, the capabilities of D3 could be incorporated into the Esri JavaScript API to enhance how the data is visualized. There are current examples of D3 being used with Esri API on the Esri developers website (Esri 2014b). Using the technology described in this article, a simple web mapping interface could permit users to more easily explore the vast amount of data available from the Census.

REFERENCES

Batty, Michael, Andrew Hudson-Smith, Richard Milton, and Andrew Crooks. 2010. "Map Mashups, Web 2.0 and the GIS Revolution." *Annals of GIS* 16(1): 1–13. doi:10.1080/19475681003700831.

Cuesta, Hector. 2013. *Practical Data Analysis*. Birmingham, UK: Packt Publishing.

Esri. 2014a. *ArcGIS API for JavaScript*. Accessed January 21. https://developers.arcgis.com/javascript/jsapi/.

———. 2014b. *SVG and CSS using D3*. Accessed April 29. **https://developers.arcgis.com/javascript/jssamples/styling_svg_quantize.html**.

Fielding, Roy Thomas. 2000. "Architectural Styles and the Design of Network-based Software Architectures." PhD diss., University of California, Irvine.

Peterson, Michael P. 2014. *Mapping in the Cloud*. New York: Guilford Publications.

Powell, Thomas A. 2008. *Ajax: the complete reference*. New York: McGraw-Hill.

Raasch, Jon. 2013. *JavaScript Programming Pushing the Limits: Advanced Application Development with JavaScript & HTML5.* Chichester, England: Wiley.

US Census Bureau. 2014. *Access Data with the Census API*. Accessed January 21. **http://www.census.gov/developers/**.