ulm university universität
uulm

Universität Ulm | 89069 Ulm | Germany

Fakultät für Ingenieurwissenschaften,
Informatik und Psychologie
Institut für Medieninformatik
Forschungsgruppe Visual Computing

Projektionen in D3.js

Proseminararbeit an der Universität Ulm

Vorgelegt von:
Lukas Pellot
lukas.pellot@uni-ulm.de

Gutachter:
Prof. Dr. Timo Ropinski

Betreuer:
Julian Kreiser

Wintersemester 2017/18

Fassung vom 21. Februar 2018

Satz: PDF-LaTeX 2$_\varepsilon$

Inhaltsverzeichnis

1 Einleitung **1**

2 Einführung in Projektionen **2**
2.1 Allgemeines . . . 2
2.2 Kategorisierung von Projektionen . . . 2

3 Projektionen in D3.js **6**
3.1 GeoJSON . . . 6
3.1.1 Genereller Aufbau einer JSON-Datei . . . 6
3.1.2 Repräsentation eines einzelnen Elements in GeoJSON . . . 7
3.1.3 Repräsentation von Koordinaten in GeoJSON . . . 7
3.1.4 Mögliche Elemente in GeoJSON . . . 8
3.1.5 Gegenüberstellung mit TopoJSON . . . 8
3.2 Projektionsfunktion . . . 9
3.3 Pfadfunktion . . . 10

4 Erstellen einer Landkarte in D3.js **12**
4.1 Allgemeines . . . 12
4.2 Anlegen von Projektions- und Pfadfunktion . . . 12
4.3 Einlesen und Verarbeiten der zu projizierenden Daten . . . 13

5 Choreoplethen-Karten **14**
5.1 Allgemeines . . . 14
5.2 Implementierung in D3.js . . . 14
5.2.1 Problematik bzgl. bisherigem Ansatz . . . 14
5.2.2 Implementierungsansatz zur Bewältigung . . . 14
5.2.3 Einfärbung der Landkarte . . . 15

6 Ausblick **17**

1 Einleitung

Seit jeher werden Landkarten genutzt, um die topologische Beschaffenheit der Welt sowie, vor allem in Zeiten der Globalisierung, (über-)regionale Sachverhalte und Statistiken grafisch aufbereitet darzustellen.

Allerdings bestehen zwei Problematiken. Dreidimensionale Strukturen wie Globen, und damit auch deren Oberflächen, können aufgrund des "Verlusteseiner Dimension grundsätzlich nur verzerrt auf zweidimensionalen Strukturen wie Papier oder Computer-Bildschirmen wiedergegeben werden. Außerdem gibt es unzählige Möglichkeiten, eine solche Verzerrung durchzuführen. Die Vereinheitlichung einer solchen Verzerrung in Form einer Funktion nennt man Projektion.

Die vorliegende Ausarbeitung beschäftigt sich mit solchen Projektionen. Nach einer theoretischen Einführung zu Projektionen wird gezeigt, wie Landkarten und Choreoplethenkarten in der JavaScript-Bibliothek D3.js angefertigt werden können.

2 Einführung in Projektionen

2.1 Allgemeines

Will man einen Ausschnitt der Oberfläche einer Sphäre, beispielsweise ein Land auf dem Erdball, auf einer ebenen Oberfläche wie einem Bildschirm oder einem Blatt Papier darstellen, so muss eine Funktion aufgestellt werden, die einen dreidimensionalen Punkt (üblicherweise erfolgt die Darstellung hier durch ein Paar aus Breiten- und Längengrad) in einen zweidimensionalen Punkt umwandelt. Eine solche Funktion nennt man Projektion.

Hierbei ist zu beachten, dass mit dem Verlust einer räumlichen Dimension bei der Anwendung einer Projektion stets ein gewisser Informationsverlust beziehungsweise eine -verfälschung einhergeht. Betroffen sein können hierbei der Flächeninhalt der projizierten Struktur, die Länge einer Strecke innerhalb dieser, oder Winkel innerhalb eines Streckenverlaufs. Bewerkstelligt es eine Projektion, bei mehreren projizierten Strukturen eine der genannten Eigenschaften für alle Strukturen um einen konstanten Faktor verzerrt (üblicherweise verkleinert) wiederzugeben, wird diese Projektion flächen-/längen-/winkeltreu genannt.

2.2 Kategorisierung von Projektionen

Projektionen können in folgende Kategorien eingeteilt werden:

Azimutalprojektion Bei einer Azimutalprojektion wird die Sphäre ohne Zwischenschritte direkt auf eine Ebene projiziert.

Abbildung 2.1: Flächentreue Azimutalprojektion

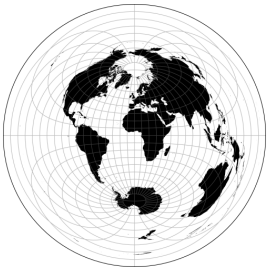

Abbildung 2.2: Längentreue Azimutalprojektion

Konische Projektion Bei einer konischen Projektion wird die Sphäre zunächst auf einen Kegel projiziert. Dieser wird dann aufgeschnitten und auf eine Ebene ausgerollt.

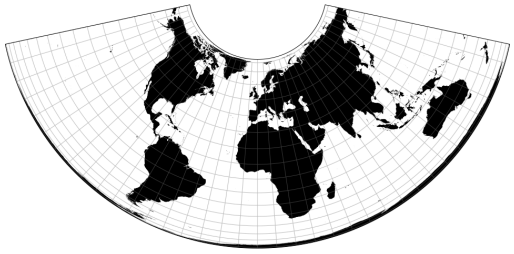

Abbildung 2.3: Flächentreue konische Projektion

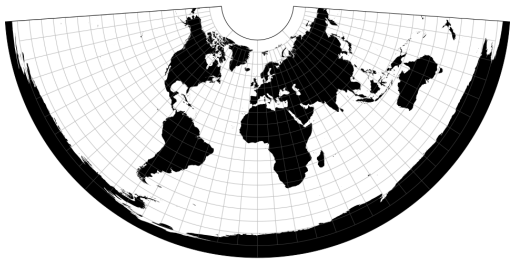

Abbildung 2.4: Längentreue konische Projektion

Zylindrische Projektion Bei einer zylindrischen Projektion wird die Sphäre zunächst auf einen Zylinder projiziert. Dieser wird dann aufgeschnitten und auf eine Ebene ausgerollt.

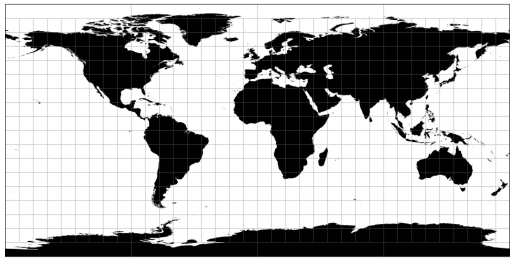

Abbildung 2.5: Längentreue Zylinderprojektion

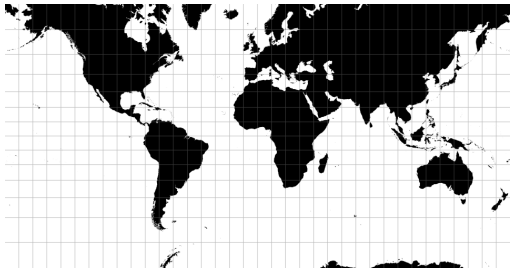

Abbildung 2.6: Winkeltreue Zylinderprojektion

Pseudozylindrische Projektion Pseudozylindrische Projektionen stellen eine Verallgemeinerung zylindrischer Projektionen dar. Breitengrade werden ebenso als parallele Linien projiziert, allerdings sind Höhengrade der konkreten Projektion entsprechend geneigt.

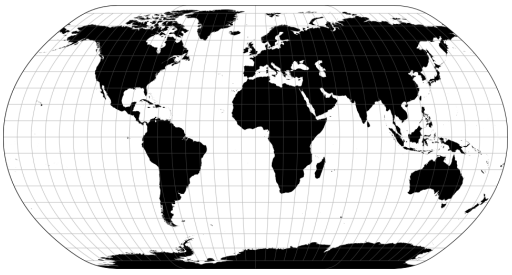

Abbildung 2.7: „Natural Earth“-Projektion

Zusammengesetzte Projektion Zusammengesetzte Projektionen kombinieren verschiedene der oben genannten Projektionen. Im Gegensatz zu den anderen Kategorien werden zusammengesetzte Projektionen selten für die Darstellung der gesamten Welt, sondern häufiger lediglich für Ausschnitte ebendieser genutzt.

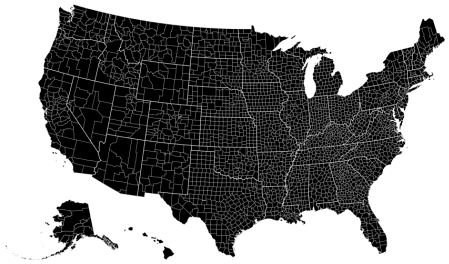

Abbildung 2.8: Albers-USA-Projektion

Hier wird die Albers-USA-Projektion gezeigt, die den Hauptteil der USA als konische Projektion, allerdings aus Platzgründen Alaska und Hawaii in der linken unteren Ecke um einen Faktor von etwa 0.35 verkleinert als zylindrische Projektion darstellt.

Anmerkung Bei Betrachtung der Abbildungen fällt auf, dass keine der Kategorien Treue hinsichtlich einer Eigenschaft bedingt oder fördert, und diese lediglich der Optik dienlich sind.

3 Projektionen in D3.js

Projektionen in D3.js haben drei wesentliche Bestandteile, welche im Folgenden genauer erläutert werden:

- Geographische Daten
  Müssen im Format GeoJSON oder TopoJSON vorliegen
- Projektionsfunktion
  Dient zur Projektion eines einzelnen Punktes
- Pfadfunktion
  Verallgemeinert die Projektion eines einzelnen Punktes auf die Projektion einer topologischen Struktur

3.1 GeoJSON

GeoJSON ist ein offenes Format, um geografische Daten zu repräsentieren. Es verwendet die JavaScript Object Notation (kurz: JSON) und ist aufgrund der hohen Verbreitung und Unterstützung durch viele Programmiersprachen im Allgemeinen und JavaScript im Speziellen bestens für die Verarbeitung der enthaltenen Daten geeignet.

Wie bei jedem JSON-Objekt besteht auch ein GeoJSON-Objekt aus diversen Schlüssel-Wert-Paaren. Es existieren zwingend erforderliche Schlüssel, die Werte zu diesen Schlüsseln sind allerdings variabel bzw. können einem gewissen Wertebereich oder einer Menge entstammen. Welche Einschränkungen dabei vorliegen, wird im Folgenden erläutert.

3.1.1 Genereller Aufbau einer JSON-Datei

Der erste erforderliche Schlüssel ist "type". Je nachdem, ob man lediglich ein einzelnes Element oder mehrere Elemente repräsentieren will, kann er einen von zwei Werten annehmen.

- Soll lediglich ein einzelnes Element repräsentiert werden, muss "type"den Wert "Feature"haben.

In diesem Fall muss außerdem der zweite Schlüssel zwingend "geometry"lauten Dessen Wert ist dann die GeoJSON-konforme Repräsentation eines einzelnen Elements.

- Sollen mehrere Elemente, wie verschiedene Länder eines Kontinents, repräsentiert werden, muss "type"den Wert "FeatureCollection"haben.
  In diesem Fall muss außerdem der zweite Schlüssel zwingend "features"lauten. Dessen Wert ist ein Array an einzelnen Elementen, also GeoJSON-Objekten, bei denen der Schlüssel "type"den Wert "Feature"hat).

Optional kann außerdem auf dieser Ebene der Schlüssel "bboxängelegt werden. Dessen Wert ist dann ein Array mit vier Elementen, wobei die ersten zwei Elemente den südwestlichen und die zweiten zwei Elemente den nordöstlichen Eckpunkt eines Rechtecks darstellen, das alle Punkte des repräsentierten Elements beziehungsweiser aller zu repräsentierenden Elemente beinhaltet.

3.1.2 Repräsentation eines einzelnen Elements in GeoJSON

Zur Repräsentation eines einzelnen Elements in GeoJSON werden mindestens zwei Schlüssel benötigt. Diese lauten:

- "type"
  Der Wert dieses Schlüssels ist der Typ des zu repräsentierenden Elements als String.

- "coordinates"
  Der Wert dieses Schlüssels sind die Koordinaten, die das entsprechende Element repräsentieren.

Optional kann außerdem der Schlüssel "properties"hinzugefügt werden, dessen Wert kann wiederum ein beliebiges JSON-Objekt sein. So wird unter Anderem das Setzen beliebiger Kennzahlen bezüglich des entsprechenden Objekts ermöglicht (Beispiel: Einwohnerzahl oder Bruttoinlandsprodukt pro Land).

3.1.3 Repräsentation von Koordinaten in GeoJSON

Geographische Koordinaten werden in GeoJSON als Array mit zwei Werten in der Form [Längengrad, Breitengrad], im Folgenden auch als Koordinatenpaar bezeichnet, definiert. Zu beachten ist hier, dass Längen- und Breitengrad im Vergleich zu sonstigen Anwendungen und Notationen vertauscht werden.

3.1.4 Mögliche Elemente in GeoJSON

GeoJSON unterstützt folgende Elemente:

- Point
  - Wird genutzt, um einen einzelnen Punkt zu repräsentieren
  - Koordinaten: Koordinatenpaar
- LineString
  - Wird genutzt, um (geknickte) Strecken(-verläufe) darzustellen
  - Koordinaten: Array mit beliebig vielen Koordinatenpaaren
- Polygon
  - Wird genutzt, um geschlossene Flächen darzustellen
  - Koordinaten: Array mit bis zu zwei Arrays, die wiederum beliebig viele Koordinatenpaare beinhalten
    Zweiter Satz an Koordinatenpaaren wird genutzt, um "Löcherïn der Struktur, den der erste Satz beschreibt, wiederzuspiegeln

Außerdem kann jedes einteilige Element auch mit anderen einteiligen Elementen vom selben Typ dargestellt werden. Dem Namen des Elements wird dann "Multi"vorangestellt (also MultiPoint, MultiLineString, MultiPolygon), die Koordinaten bestehen dann aus einem Array, dessen Elemente Koordinaten gemäß den Vorgaben des entsprechenden einteiligen Elements sind.

3.1.5 Gegenüberstellung mit TopoJSON

TopoJSON stellt einen alternativen Weg dar, um geographische Daten standardisiert zu speichern. Es lassen sich folgende Unterschiede sowie Vor- und Nachteile feststellen:

	GeoJSON	TopoJSON
Speicherung von Strukturen	• Explizite Speicherung des Pfades für jedes Element • Pfade werden bei aneinandergrenzenden Elementen mehrfach gespeichert	• Alle genutzten Pfade werden genau einmal gespeichert • Zusätzlich Information für jedes Element, welche Pfade es nutzt
Vorteil	Einfachere Datenstruktur	Kleinere Dateigröße
Nachteil	Ggf. wesentlich umfangreichere Datei	• Komplexere Dateistruktur • Ggf. weitere Bibliotheken zur Verarbeitung erforderlich

3.2 Projektionsfunktion

In D3.js wird die Funktionalität, einen bestimmten Punkt von einer Sphäre auf eine Ebene zu projizieren, von einer sogenannten Projektionsfunktion übernommen. Diverse etablierte Projektionen wie die Merkator- oder die Rektangularprojektion werden von D3.js bereits 'ab Werk' im Modul d3-geo mitgeliefert. Viele weitere (exotischere) Projektionen sind ebenfalls vordefiniert und können über das Paket d3-geo-projections nachgeladen werden.

Auf Projektionen können diverse Funktionen aufgerufen werden, um die Ausgabe zu gestalten. Es folgt ein Auszug aus diesen:

- projection(point)
  - Gibt ein Array mit den Koordinaten des projizierten Punktes (in Pixel) zurück
  - point: Koordinatenpaar
- projection.invert(point)
  - 'Umkehrfunktion' zu projection(point)
- projection.translate([tx, ty])

  - Versetzt die Koordinaten des projizierten Punktes um die Koordinaten tx und ty entlang der entsprechenden Achsen
  - Versatz bezieht sich auf Mittelpunkt der Projektion
  - Falls keine Werte übergeben werden, wird der aktuell festgelegte Versatz zurückgegeben (Standardwert: [480, 250])

- projection.center([longitude, latitude])
  - Setzt den Mittelpunkt der Projektion auf den übergebenen Punkt um
  - Falls keine Werte übergeben werden, wird der aktuell festgelegte Mittelpunkt zurückgegeben (Standardwert: [0, 0])

- projection.scale(scale)
  - Skaliert die Koordinaten des projizierten Punktes um den übergebenen Faktor
  - Falls kein Wert übergeben wird, wird der aktuell festgelegte Skalierungsfaktor zurückgegeben

3.3 Pfadfunktion

Die Pfadfunktion dient in D3.js dazu, statt lediglich der Projektion einzelner Punkte die Projektion beliebiger (in GeoJSON oder TopoJSON vorhandener) Strukturen zu ermöglichen.

Wie auf Projektionsfunktionen können auch auf Pfadfunktionen diverse Funktionen aufgerufen werden. Es folgt ein Auszug aus diesen:

- d3.geoPath(projection)
  - Erzeugt eine neue Pfadfunktion, die gemäß der übergebenen Projektionsfunktion Elemente abbildet
  - Wird keine Projektionsfunktion übergeben, wird standardmäßig d3.geoAlbersUSA() (die Albers-USA-Projektion) genutzt

- path.projection(projection)
  - Ändert die von path genutzte Projektionsfunktion auf die übergebene
  - Wird keine Projektionsfunktion übergeben, wird stattdessen die aktuell genutzte Projektionsfunktion übergeben

- path(object)

– zeichnet das übergebene GeoJSON-/TopoJSON-Objekt gemäß der gesetzten Projektionsfunktion als SVG-Pfad

4 Erstellen einer Landkarte in D3.js

Im Folgenden wird gezeigt, wie man eine einfache Landkarte in D3.js erstellen kann. Dieser Prozess kann in drei logische Abschnitte aufgeteilt werden.

4.1 Allgemeines

Zu Beginn müssen ein allgemeine Werte wie Breite und Höhe des Bereichs auf dem Bildschirm sowie ein svg-Tag, in den die Landkarte projiziert werden soll, festgelegt werden.

```
var width = 1440;
var height = 750;

var svg = d3.select("body")
      .append("svg")
      .classed("map", true);

var background = svg.append("rect")
      .attr("width", width)
      .attr("height", height)
      .classed("background", true);
```

4.2 Anlegen von Projektions- und Pfadfunktion

Danach müssen die in beschriebenen Projektions- und Pfadfunktionen angelegt werden, mithilfe derer dann die Landkarte gezeichnet werden kann.

```
var projection = d3.geoEquirectangular()
      .scale(width / 2 / Math.PI)
      .translate([width / 2, height / 2]);
```

```
var path = d3.geoPath()
      .projection(projection);
```

4.3 Einlesen und Verarbeiten der zu projizierenden Daten

Zuletzt muss noch die Datei, welche die zu projizierenden Daten repräsentiert, eingelesen und verarbeitet werden.

```
var url = "..."; // URL, die auf die zu verarbeitende Datei verweist

d3.json(url, function(geojson){
   svg.append("path")
         .attr("d", path(geojson));
});
```

Die entstehende Landkarte hat dieses Aussehen:

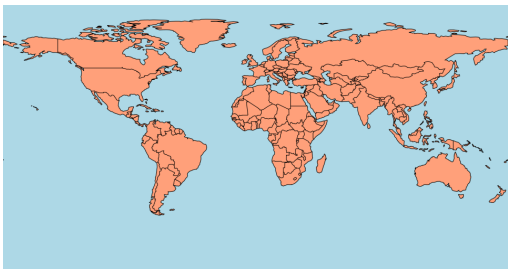

Abbildung 4.1: „Natural Earth“-Projektion

5 Choreoplethen-Karten

5.1 Allgemeines

Choreoplethenkarten unterscheiden sich von gewöhnlichen/einfachen Landkarten darin, dass bestimmte Abschnitte einer Karten einer bestimmten statistischen Variablen entsprechend eingefärbt werden. Daher rührt auch ihre alternative Bezeichnung als 'Thematische Karten'.

Nützlich ist diese Art einer Landkarte vor allem, um zu zeigen, wie Messungen/Werte in bestimmten geographischen Regionen variieren, sowie daraus gegebenenfalls Schlüsse zu ziehen. Ein Beispiel hierfür wäre die Anzahl erwerbsloser Personen in Relation zur Einwohnerzahl für jeden Landkreis in Deutschland.

5.2 Implementierung in D3.js

5.2.1 Problematik bzgl. bisherigem Ansatz

Beim Versuch, eine solche Choreoplethenkarten in D3.js zu implementieren, tritt ein Problem mit der bisherigen Implementierung auf. Diese bildet zwar alle Länder ab, nutzt hierzu aber lediglich einen einzelnen Pfad, welcher folglich auch nur mit einer einzelnen Farbe gefüllt werden kann, was die Vergleichbarkeit von Regionen unmöglich macht.

Man muss also zunächst die Implementierung so abändern, dass statt einem Pfad für alle Länder ein Pfad pro Land gezeichnet wird. Möglich wird dies, wenn man statt dem übergebenen GeoJSON-Objekt als Ganzes nur das "featuresArray unter Zuhilfenahme des Enter-Update-Exit-Patterns an die Funktion übergibt.

5.2.2 Implementierungsansatz zur Bewältigung

Der allgemeine Teil und der Teil zum Anlegen von Projektions- und Pfadfunktion gestalten sich wie in 4.1 und 4.2 beschrieben. Danach folgt das veränderte Einlesen der Datei wie folgt:

```
var url = "...";
```

```
d3.json(url, function(geojson){
   svg.selectAll("path")
         .data(geojson.features)
         .enter()
         .append("path")
         .attr("d", path);
})
```

5.2.3 Einfärbung der Landkarte

Um nun noch gemäß der Aufgabenstellung jedes Land entsprechend seiner Kennzahl einfärben zu können, muss eine Farbskala erstellt werden. Beim Zeichnen der Pfade wird dann die jeweilige Füllfarbe mithilfe des Werts der entsprechenden Kennzahl und der Farbskala berechnet und gesetzt.

Der allgemeine Teil und der Teil zum Anlegen von Projektions- und Pfadfunktion gestalten sich erneut wie in 4.1 und 4.2 beschrieben. Danach folgt das Einlesen der Datei und Einfärben der Pfade wie folgt (Kennzahl ist das Bruttoinlandsprodukt jeden Landes, das in der GeoJSON-Datei über den optionalen Wert "properties"mit übergeben wird)

```
var url = "...";

d3.json(url, function(geojson){
   var features = geojson.features;

   var min = d3.min(features, function(d){
      return d.properties.gdp_md_est;
   });

   var max = d3.max(features, function(d){
      return d.properties.gdp_md_est;
   });

   var color = d3.scaleLinear()
         .domain([min, max])
         .range(["white", "crimson"]);

   svg.selectAll("path")
```

```
            .data(features)
            .enter()
            .append("path")
            .attr("d", path)
            .style("fill", function(d){
               return color(d.properties.gdp_md_est);
            });
});
```

Die entstehende Landkarte hat dieses Aussehen:

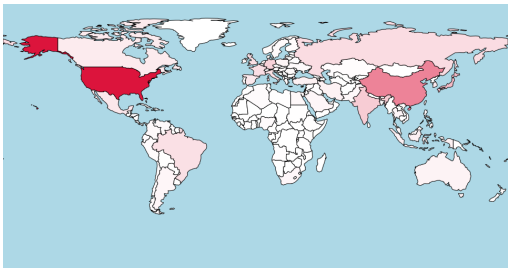

Abbildung 5.1: „Natural Earth“-Projektion

6 Ausblick

Wie in dieser Arbeit gezeigt wurde, kann in D3.js ohne großen Aufwand eine Landkarte erstellt und nach Belieben eingefärbt werden. Abschließend gilt zu sagen, dass die Möglichkeit besteht, Landkarten jeder Art in D3.js beliebig zu gestalten, und man keinesfalls auf rein optische Vorgänge beschränkt ist.

So ist zum Beispiel eine interaktive Gestaltung der Karte innerhalb von D3.js möglich, indem man gezeichneten Pfaden Maus- oder sonstige Ereignisse und deren Verarbeitung zuweist. In entsprechender Fachliteratur wie der offiziellen Dokumentation finden sich viele weitere Ansätze zur Gestaltung.

Literaturverzeichnis

[1] M. Bostock, *Basic D3 map (Mercator) in d3v4*, 2018 (accessed: 22.02.2018). [Online]. Available: http://bl.ocks.org/almccon/fe445f1d6b177fd0946800a48aa59c71

[2] ——, *D3 API documentation*, 2018 (accessed: 22.02.2018). [Online]. Available: https://github.com/d3/d3/blob/master/API.md#geographies-d3-geo

[3] J. A. G. Gomez, *GeoJson map of Colombia*, 2018 (accessed: 22.02.2018). [Online]. Available: https://bl.ocks.org/john-guerra/43c7656821069d00dcbc