

S. Rinzivillo – rinzivillo@isti.cnr.it

DATA VISUALIZATION AND VISUAL ANALYTICS

SPATIAL DATA AND GEOGRAPHY

Objective

- To show **spatial distribution** of data
- To show **relative positions** of data components
- Thematic maps
 - Mapping to attribute data (quantitative and qualitative) on a map
 - Geometry linked to fixed geographical position

Map Design

- Projection
 - Map curved 3D objects to a place
- Scale
 - Reduction of a map to the available space
- Symbolization and themes
 - Equivalent to encoding with Visual Variables



SCALE

Map Scale

- Defines as the ratio between a distance on the map and the corresponding distance on the Earth
 - Usually expressed as verbal ratio
 - 1:100
 - Distance on the map is always expressed as one
 - The ratio is dimensionless
 - The larger the fraction, the greater the map's details

Map scale (1:50,000,000)



Map Scale (1:6,500,000)



Map Scale (1:1,500,000)



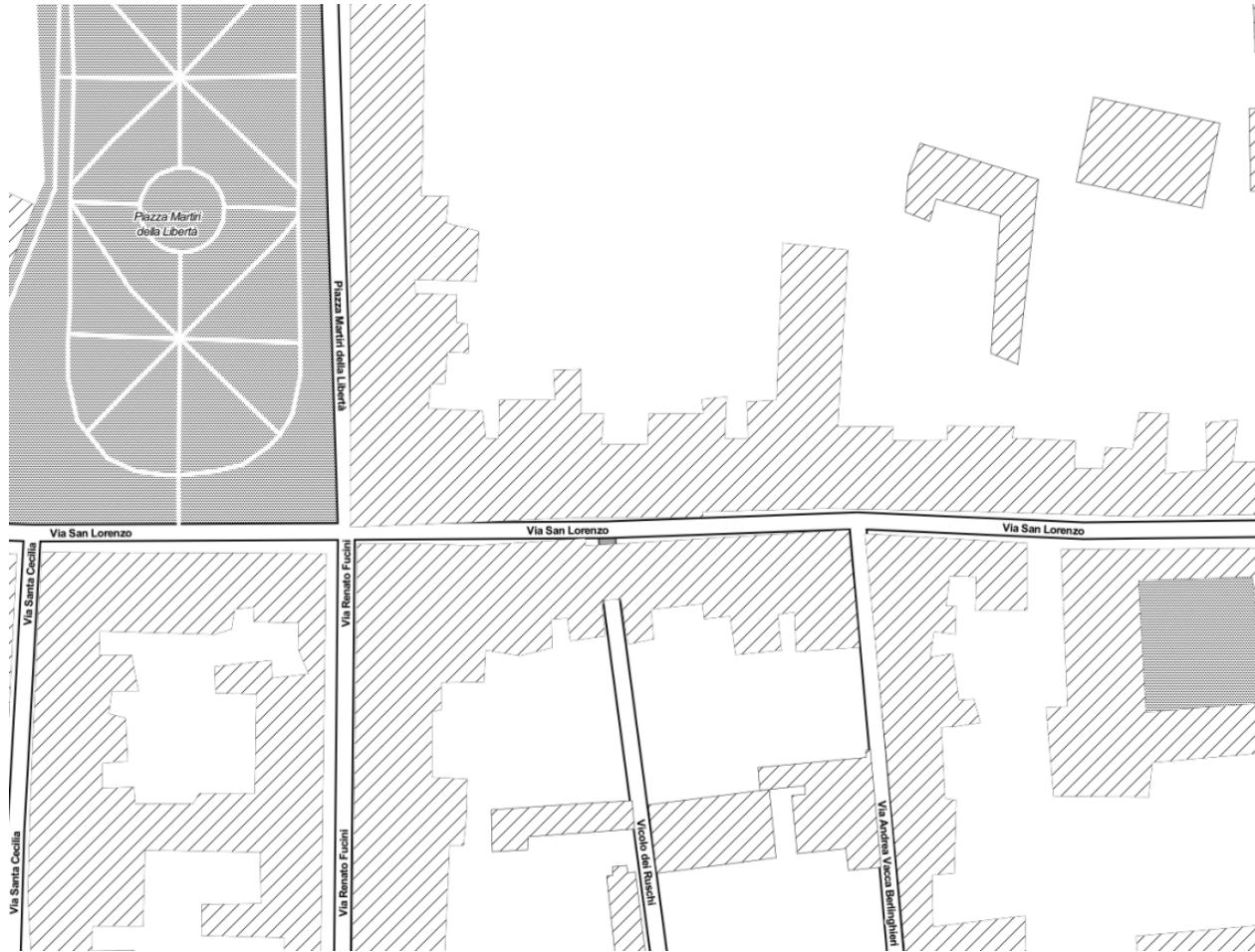
Map Scale (1:100,000)



Map Scale (1:10,000)



Map Scale (1:1,000)



PROJECTIONS

Cartography as Art









The New World

- New challenges for geographers
- Since XVI century new methods to represent geography
- From plane to globe

Basic components...

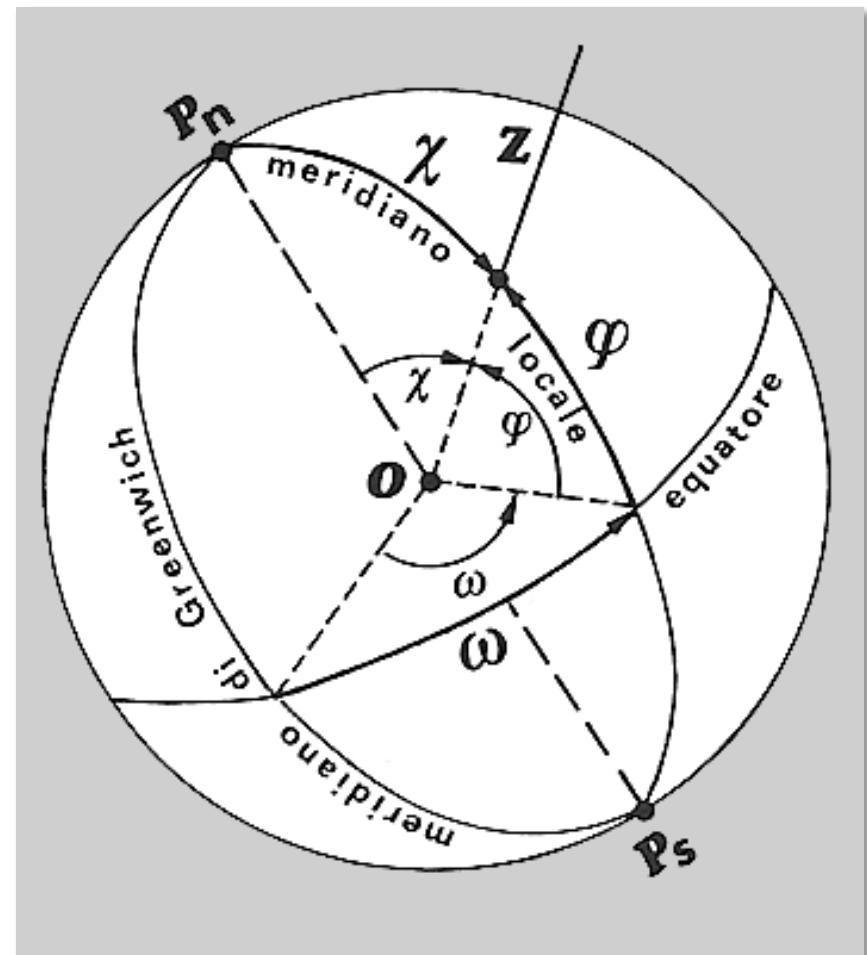
- A reference system
- A set of coordinates

Reference System

- Univocally determine a position in 3D (2D+1D)
- Need for a simple model:
 - Mathematically tractable: surface
 - Link to physical world
- Typical surfaces:
 - Sphere
 - Ellipsoid (spheroid)
 - Geoid

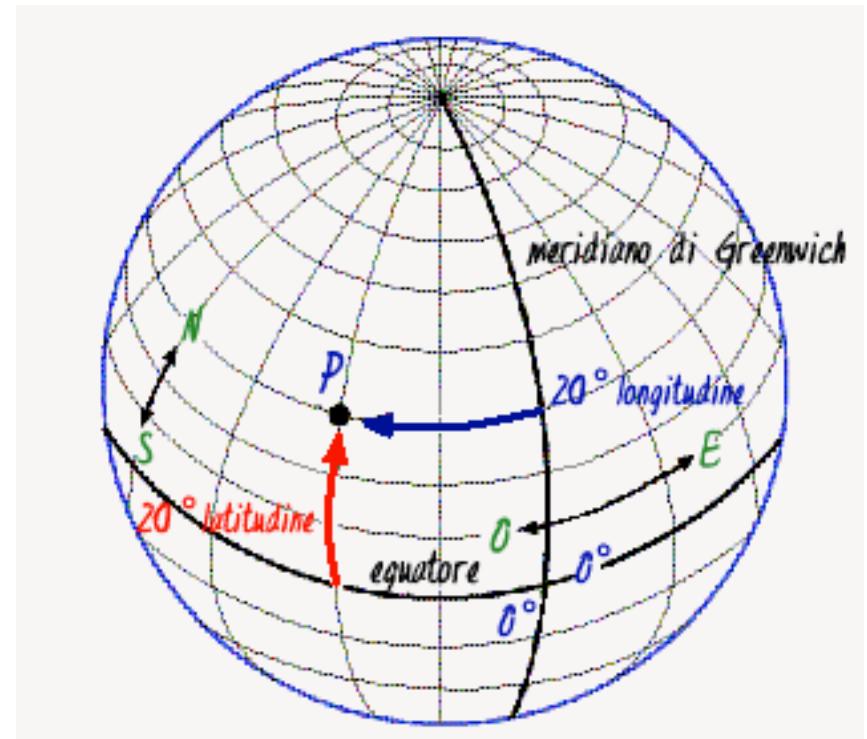
Coordinate (2D+1D)

- Position relative to the reference system
- Angular coordinates
 - Longitude
 - Latitude
- Altitude as offset from the reference point



Latitude and Longitude

- Latitude: angular distance from equator
- Longitude: angular distance from central meridian

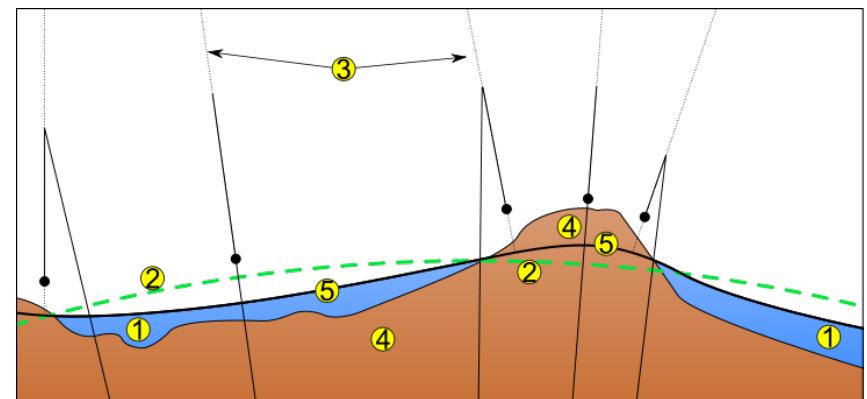


Which Reference System?

- Earth present a complex surface, results of gravity, magnetical forces and different densities
- Mathematic representation is very complex

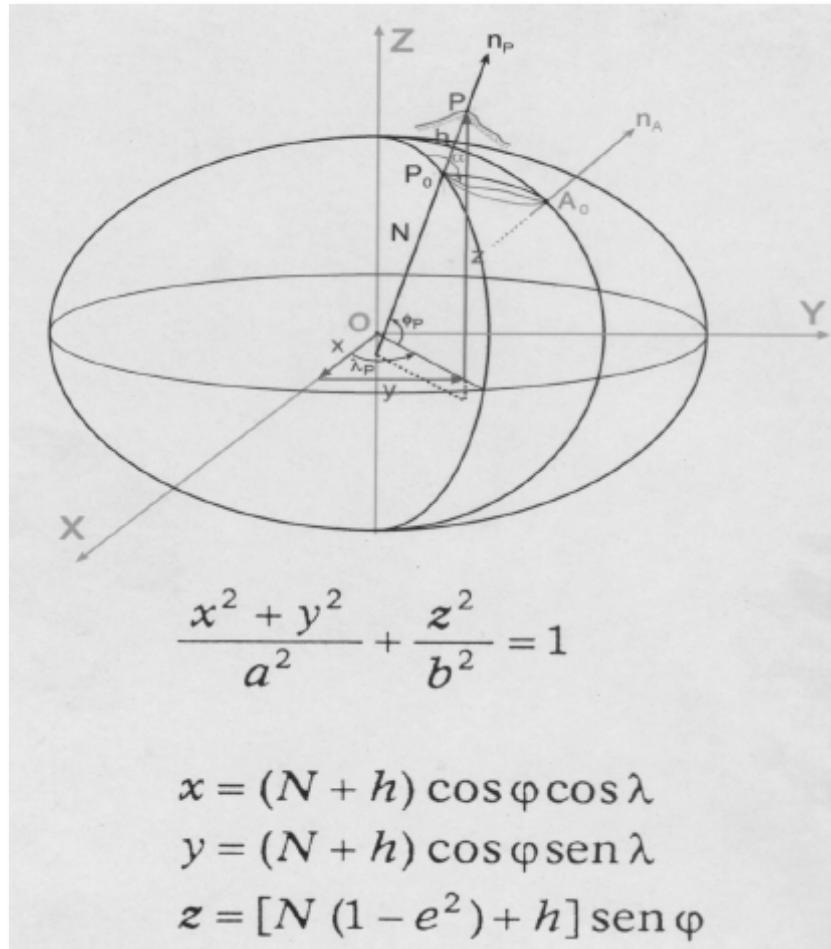
Geoid

- Geoid: surface where gravity is constant in each point
- Average surface of seas



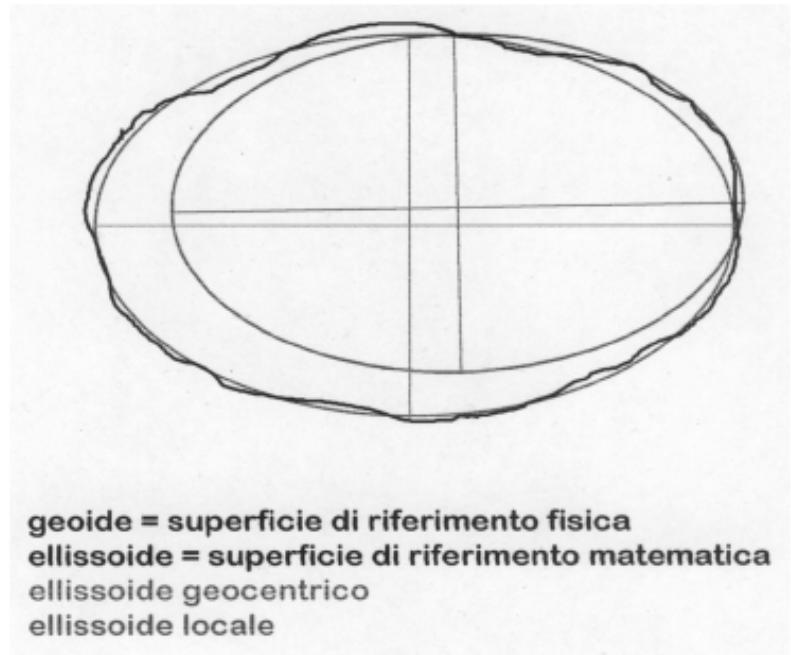
Which Reference System?

- Ellipsoid: clear and easy mathematic definition
- Easy to define a position of a point in the space
- Low differences with the real geoid ($\sim 40m$)



Datum

- An ellipsoid is univocally determined by 8 parameters (named Datum)
 - 2 shape parameted:
 - Equatorial radius
 - Polar radius
 - 6 parameters for position and orientation



geoide = superficie di riferimento fisica

ellissoide = superficie di riferimento matematica

ellissoide geocentrico

ellissoide locale

Which Datum?

- Diffusion of GPS systems: WGS84 (World Geodetic System 1984)
- Many local cartograph systems use local defined datum
 - In Europe, datum ED50 (European Datum 1950, Ellisseoide di Hayford) is largely used
- All datum can be mapped/translated to WGS84

Projections

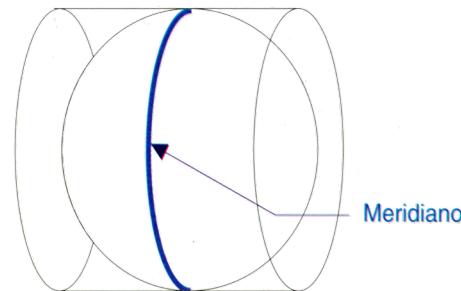
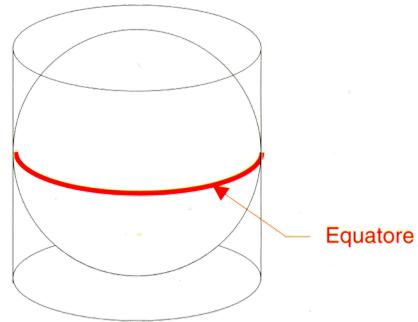
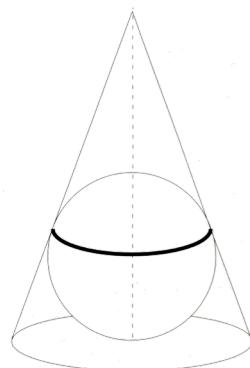
- Cartographic projections maps coordinates from the ellipsoid to the plane
- A direct mapping is not feasible without introducing deformations
- Families of mapping that preserve:
 - Angles (conformal projection)
 - Surfaces (equal area projection)
 - Minimizing both

Projections

- Each projection assume a precise datum
- For example, UTM projection uses datum WGS84 and ED50

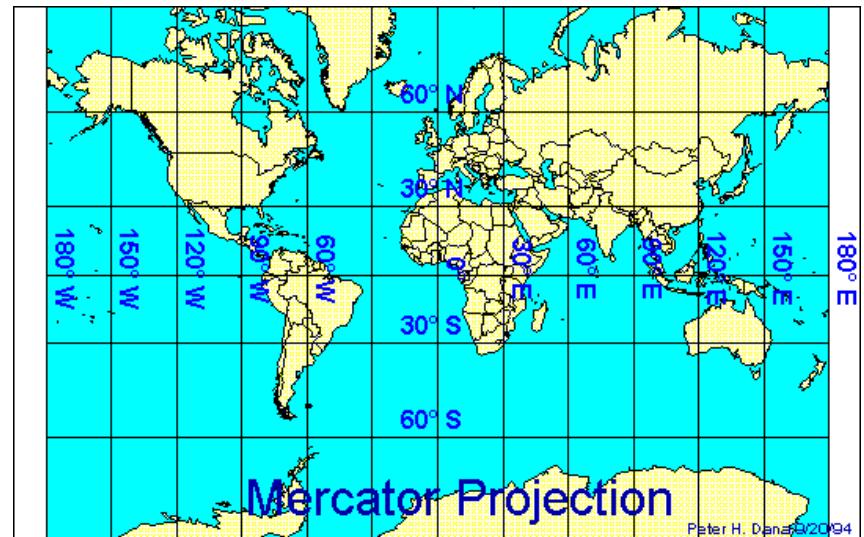
Projections

- Three different types
 - Azimuthal: projection plane is tangent to a point on the earth
 - Conic: points are projected on a cone
 - Cylindrical: points are projected on a cylinder



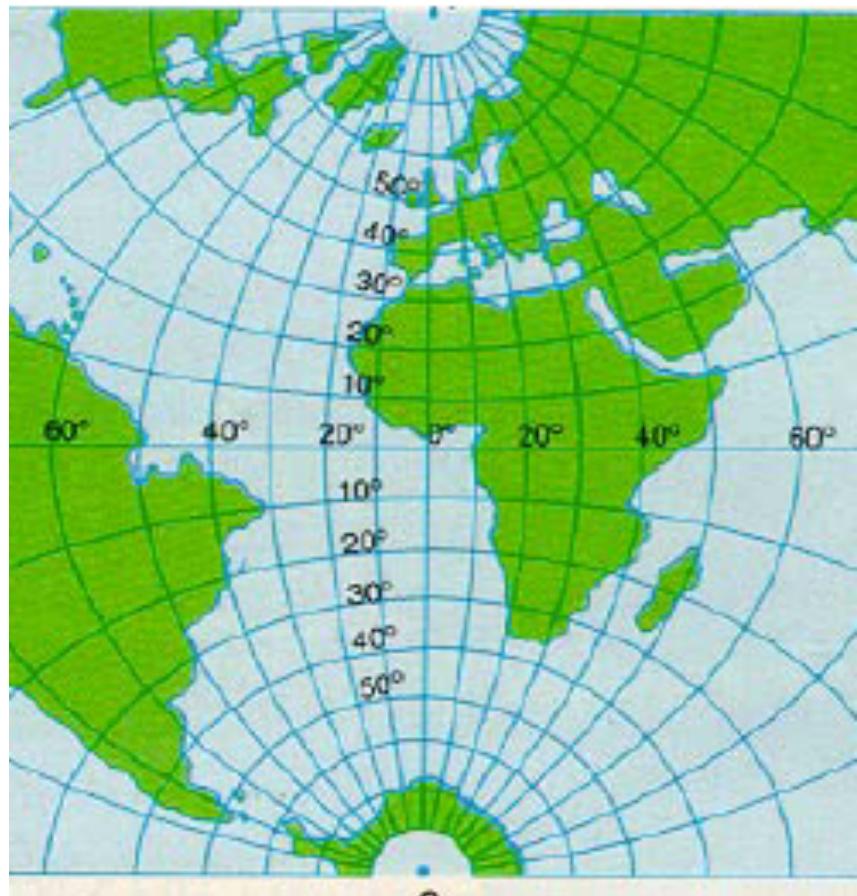
Mercator Projection

- Cylindrical projections
- Cylinder tangent to equator
- Meridians are parallel
- Low distortion for tropical zones



UTM (Universal Transverse Mercator)

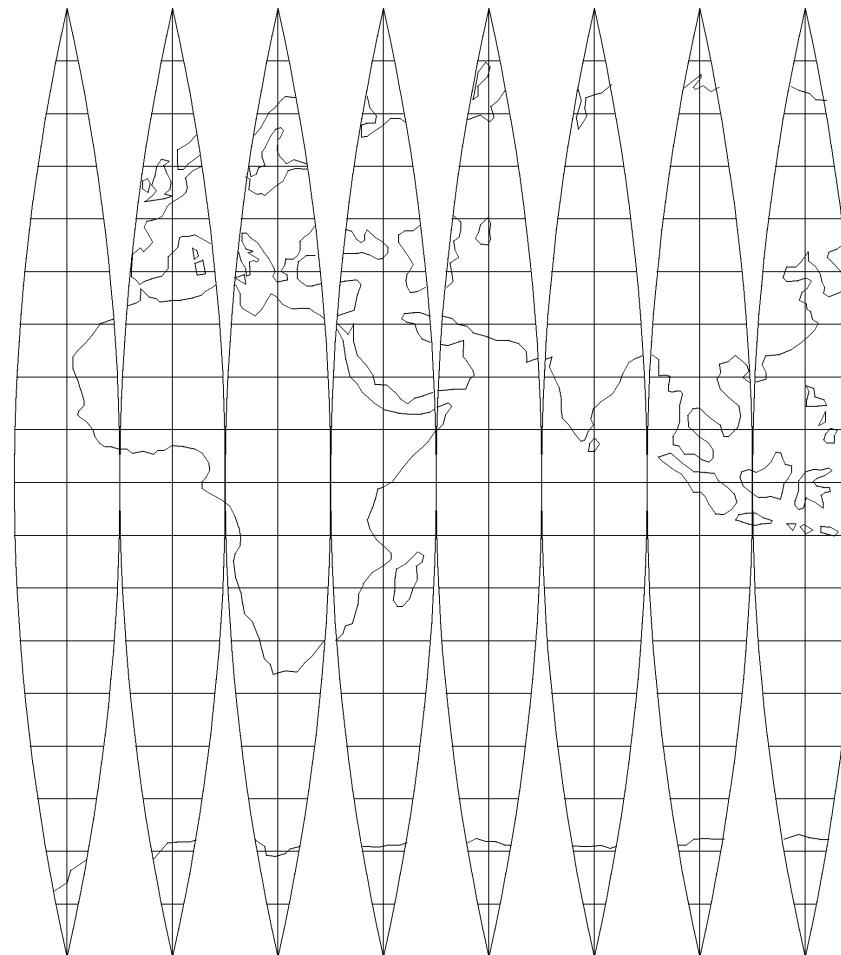
- Transverse Mercator Projection
- Cylinder tangent to one of the meridians
- Low deformation around the reference meridian



UTM Projection

- Minimize distortion
 - Each projection is limited to a zone of 6 degrees
 - Central meridian is contracted by 0.9996
 - To ensure positive coordinates, each zone has a false easting origin at 500000 m on the east of central meridian
 - Projection is limited to latitudes between -80 N and +80 N

UTM Zones



UTM zones in Italy

- Italy is covered by zones 32, 33 e 34



Reference Systems: catalogue

Spatial Reference epsg projection 3003 - monte mario / italy zone 1

[Home](#) | [Upload Your Own](#) | [List user-contributed references](#) | [List all references](#) [Search](#)

Previous: [EPSG:3002: Makassar / NEIEZ](#) | Next: [EPSG:3004: Monte Mario / Italy zone 2](#) [Link to this Page](#)

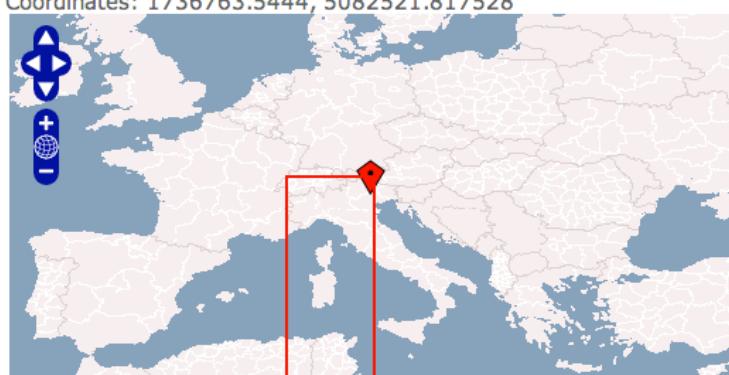
EPSG:3003

Monte Mario / Italy zone 1 ([Google it](#))

- **WGS84 Bounds:** 6.6500, 8.8000, 12.0000, 47.0500
- **Projected Bounds:** 1241482.0019, 973563.1609, 1830078.9331, 5215189.0853
- **Scope:** Large and medium scale topographic mapping and engineering survey.
- **Last Revised:** 2005-05-27
- **Area:** Italy - west of 12°E

- [Well Known Text as HTML](#)
- [Human-Readable OGC WKT](#)
- [Proj4](#)
- [OGC WKT](#)
- [JSON](#)
- [GML](#)
- [ESRI WKT](#)
- [.PRJ File](#)
- [USGS](#)
- [MapServer Mapfile | Python](#)
- [Mapnik XML | Python](#)
- [GeoServer](#)
- [PostGIS spatial_ref_sys INSERT statement](#)

Input Coordinates: 12.049609375, 45.8546875 Output Coordinates: 1736763.5444, 5082521.817528



D3.js references

- d3.geo API reference
 - <https://github.com/mbostock/d3/wiki/Geo-Projections>
- topojson API reference
 - <https://github.com/mbostock/topojson/wiki/API-Reference>

CARTOGRAFIA IN D3

Dati Geografici - GeoJSON

- Describe i poligoni geografici tramite oggetti javascript
- Organizzato in tre proprietà:
 - type (di solito: “FeatureCollection”)
 - crs (Coordinate Reference System)
 - features (elenco di oggetti geografici)
- Ogni Feature è organizzata in tre proprietà:
 - type (di solito “Feature”)
 - properties (dizionario di attributi della feature)
 - geometry (descrittore della geometria)

Dai dati a SVG

- Creazione di una **proiezione** per trasformare le **coordinate geografiche** in coordinate sullo schermo
 - Definizione di una **scala**, una **translazione**, eventuale **rotazione**
 - Oggetto: `d3.geo.projection`
- Creazione di una definizione di path per il disegno di un contorno
 - Oggetto: `d3.geo.path`

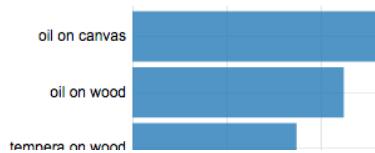
Example: Map of paintings

Geography of Paintings

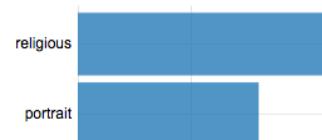


Location: Galleria degli Uffizi

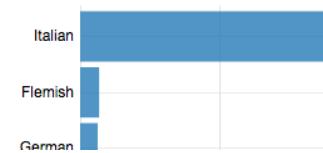
Technique



Type



School



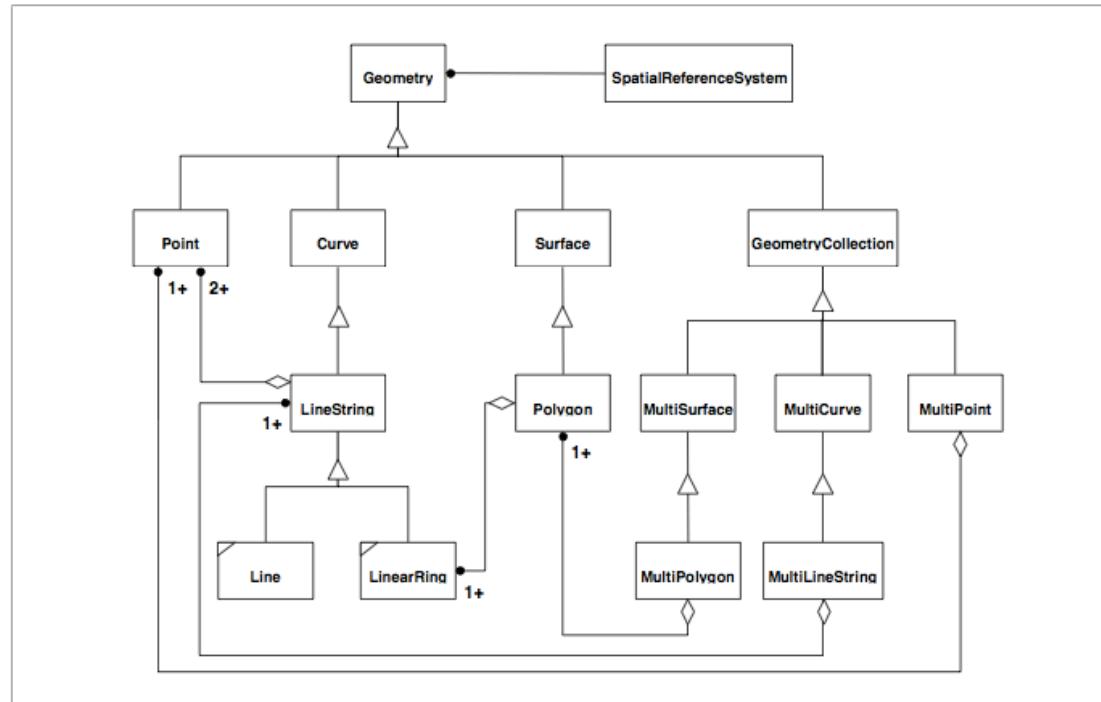
Geometries and Standards

Open GIS Consortium – OGC

- Consortium to define protocols to transmit territorial and geographic information

SFS – Simple Feature Specification

- Definition of an abstract data type:
 - Geometry

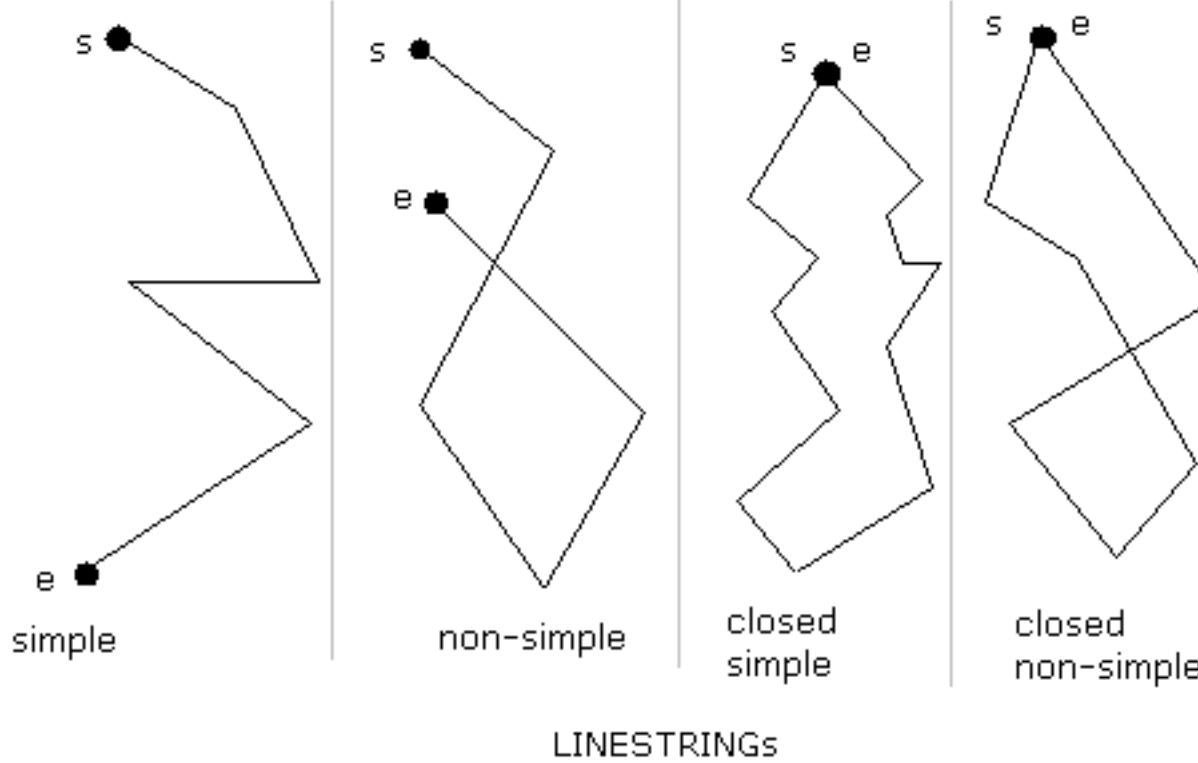


Point

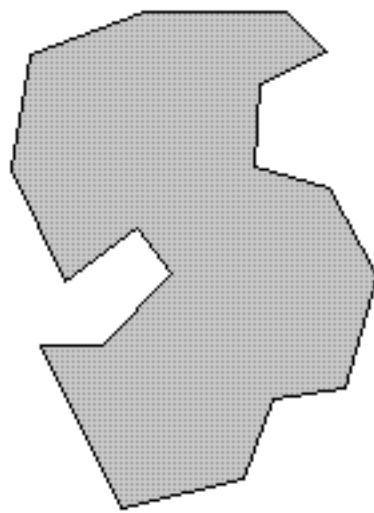


POINT

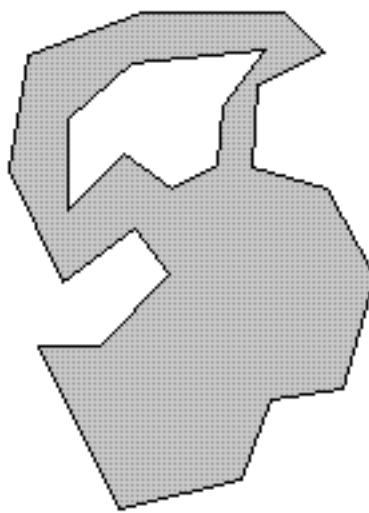
Linestring



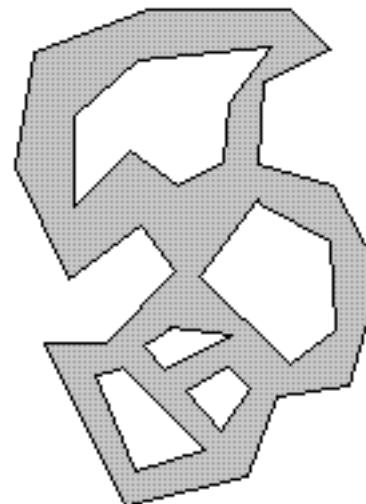
Polygon



exterior ring
no interior rings



exterior ring
1 interior ring



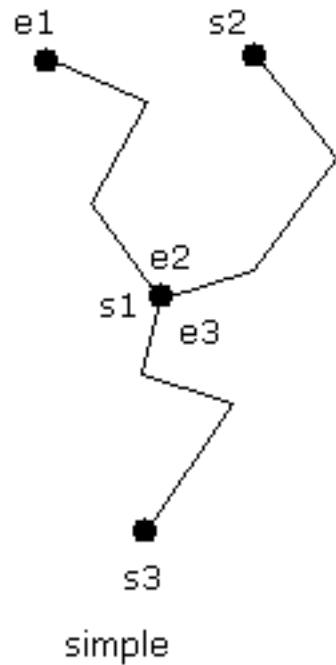
exterior ring
5 interior rings

POLYGONS

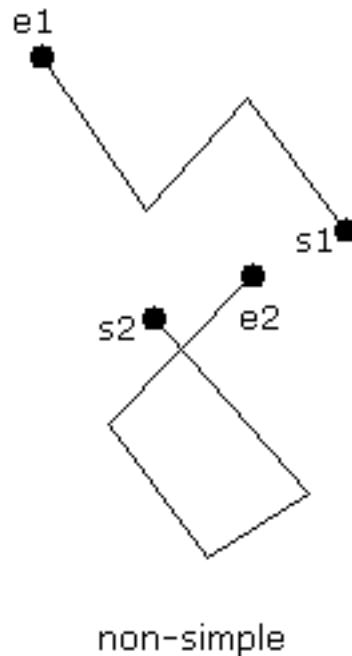
Multipoint, Multilinestring



MULTIPOINT



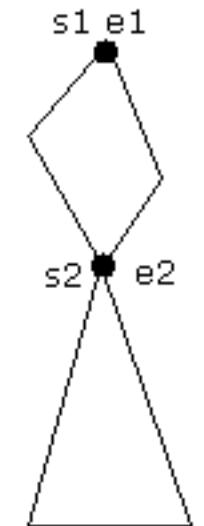
simple



non-simple



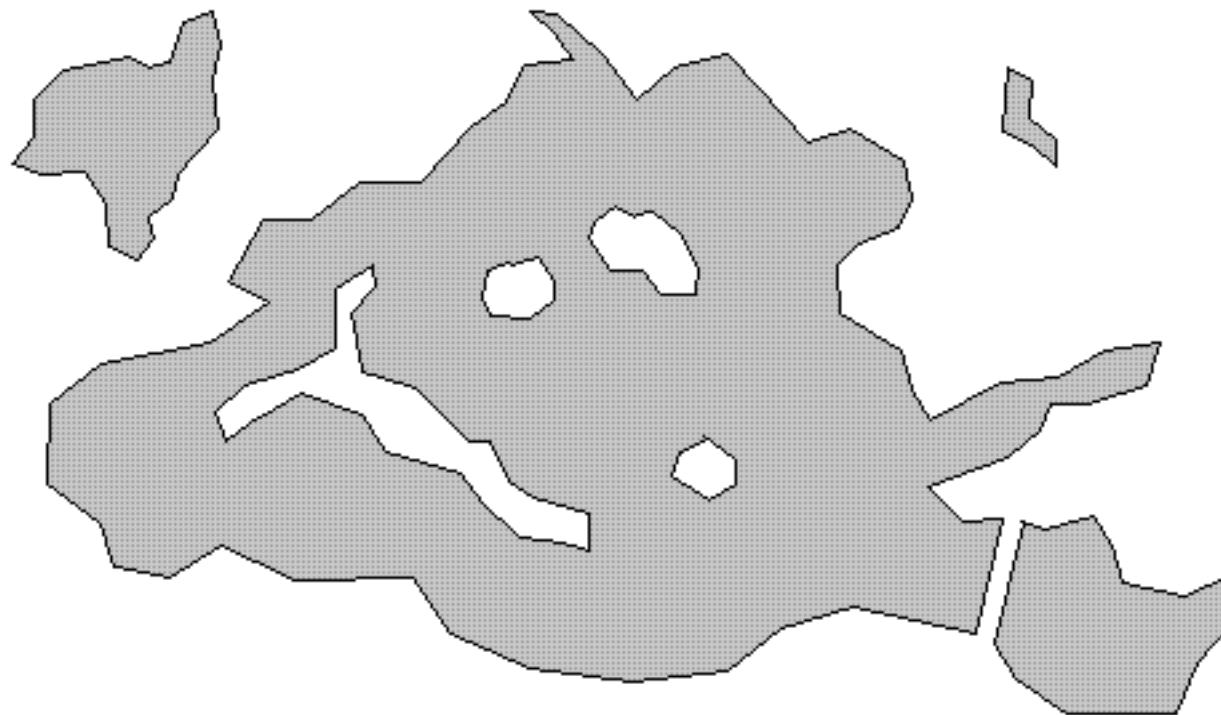
closed
simple



closed
non-simple

MULTILINESTRINGS

Multipolygon



MULTIPOINT

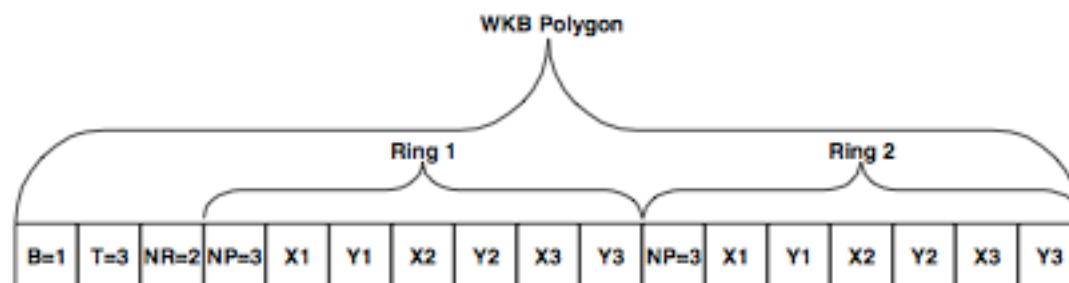
WKT – Well Known Text Format

- POINT(123.45 543.21)
- LINESTRING(100.0 200.0, 201.5 102.5, 1234.56 123.89)
- POLYGON((101.23 171.82, 201.32 101.5, 215.7 201.953, 101.23 171.82))
- POLYGON((10 10, 20 10, 20 20, 10 20, 10 10),(13 13, 17 13, 17 17, 13 17, 13 13))
- MULTILINESTRING((1 2, 3 4), (5 6, 7 8, 9 10), (11 12, 13 14))
- MULTIPOLYGON(((0 0,10 20,30 40,0 0),(1 1,2 2,3 3,1 1)),((100 100,110 110,120 120,100 100)))
- GEOMETRYCOLLECTION(POINT(1 1), LINESTRING(4 5, 6 7, 8 9), POINT(30 30))

WKB – Well Known Binary Format

- Compact representation
 - Useful to store geometries on DBMS

```
WKBPolygon {
    byte          byteOrder;
    uint32        wkbType;                                // 3
    uint32        numRings;
    LinearRing    rings[numRings];
}
```



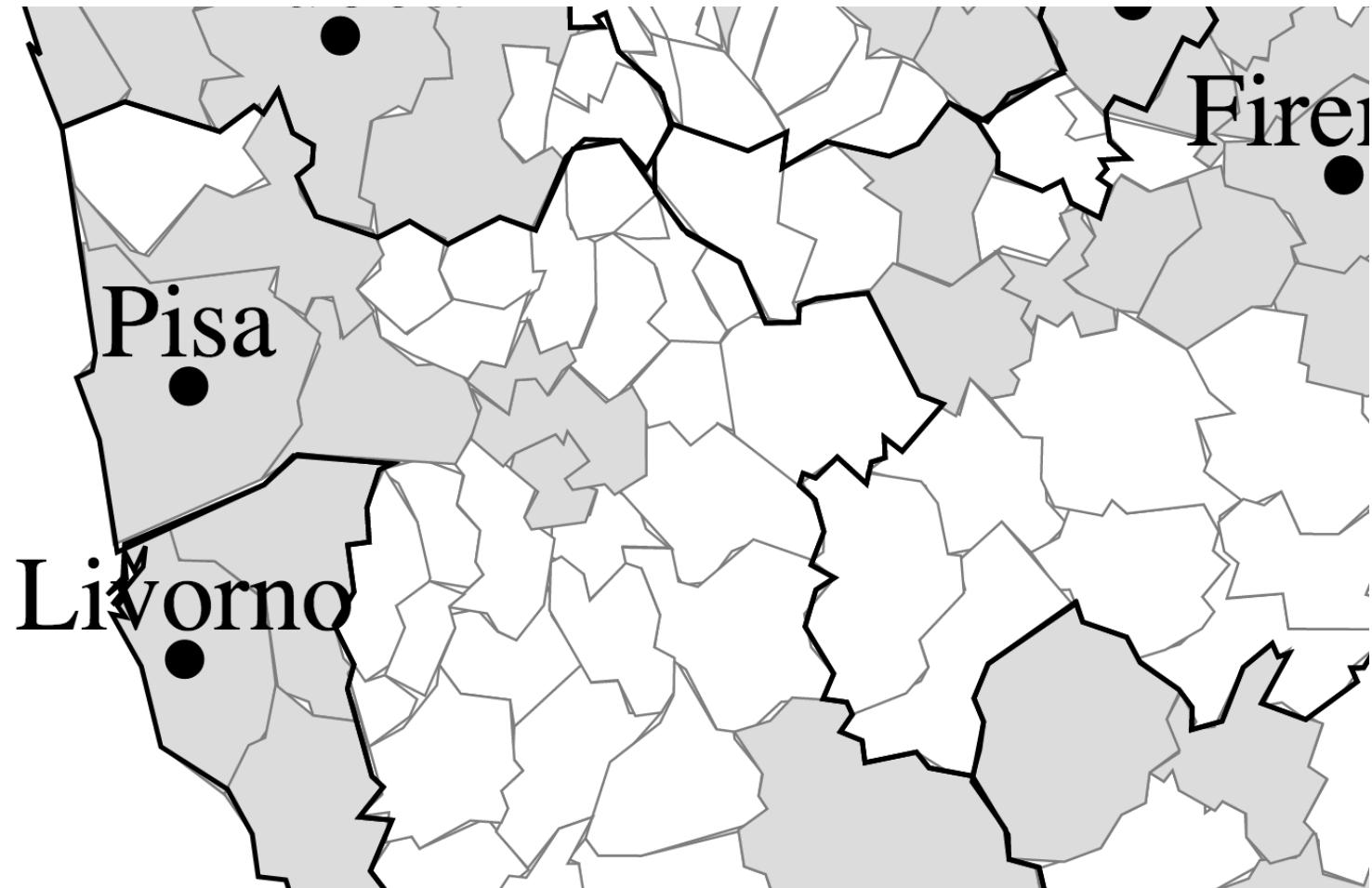
GeoJSON format

```
{ "type": "FeatureCollection",
  "features": [
    { "type": "Feature",
      "geometry": { "type": "Point", "coordinates": [102.0, 0.5]},
      "properties": { "prop0": "value0" }
    },
    { "type": "Feature",
      "geometry": {
        "type": "LineString",
        "coordinates": [
          [102.0, 0.0], [103.0, 1.0], [104.0, 0.0], [105.0, 1.0]
        ]
      },
      "properties": {
        "prop0": "value0",
        "prop1": 0.0
      }
    },
    { "type": "Feature",
      "geometry": {
        "type": "Polygon",
        "coordinates": [
          [ [100.0, 0.0], [101.0, 0.0], [101.0, 1.0],
            [100.0, 1.0], [100.0, 0.0] ]
        ]
      },
      "properties": {
        "prop0": "value0",
        "prop1": { "this": "that" }
      }
    }
  ]
}
```

TopoJSON

- Extends GeoJSON and encodes topology
- Shared lines are represented as arcs
- Reduce redundancy and decrease file size
- Topology can be exploited in specific applications

Example, without topology



Line Simplification

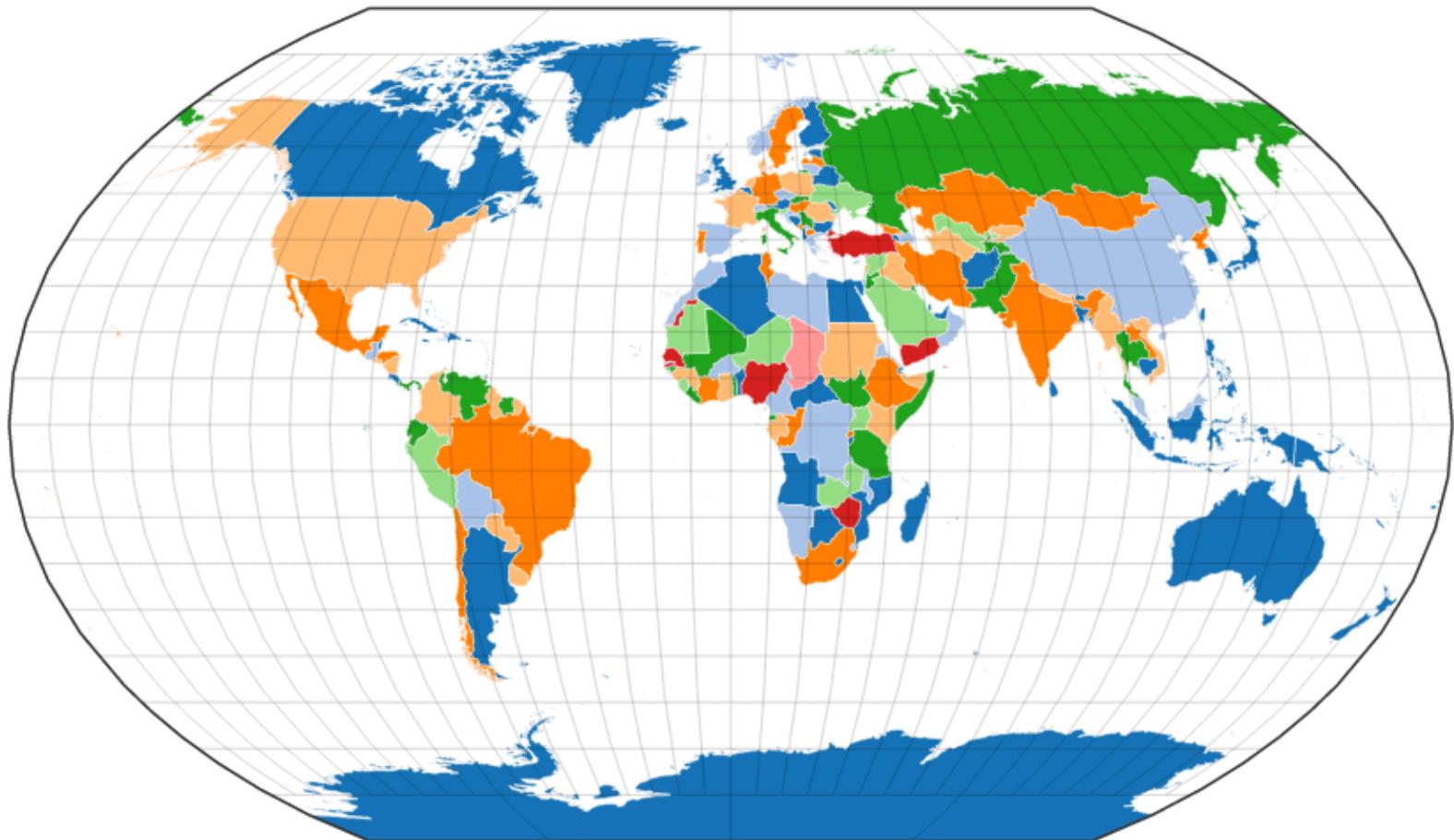
Line Simplification



$0.0053\text{px}^2 / 74.05\%$

<https://bost.ocks.org/mike/simplify/>

Color Mapping



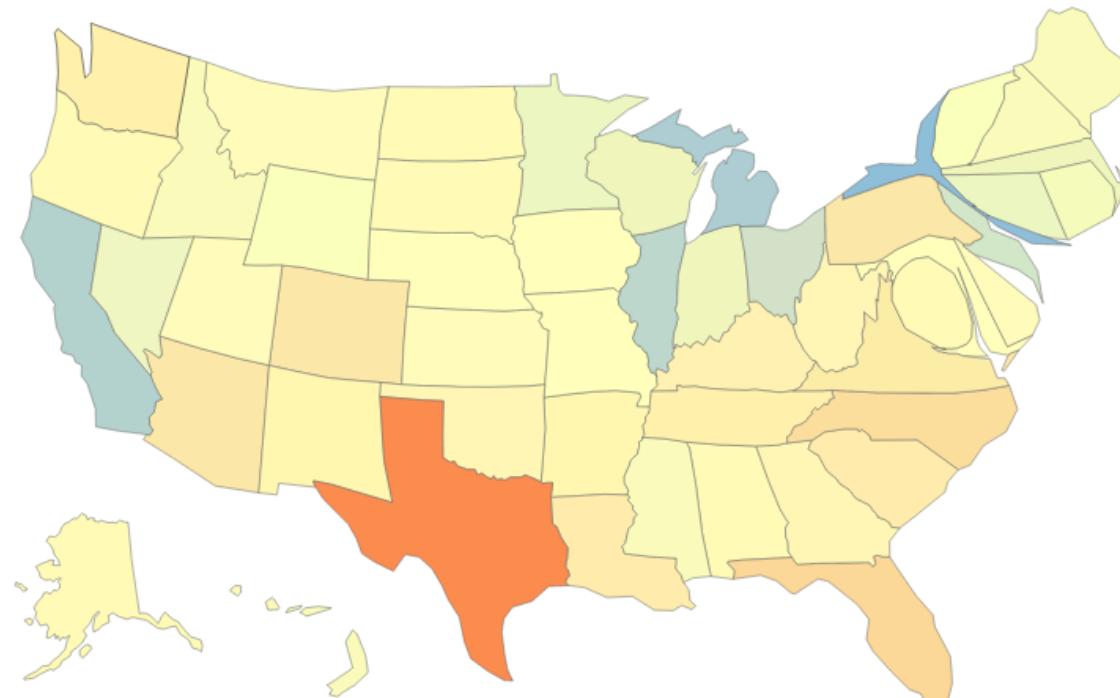
<http://bl.ocks.org/jasondavies/4188334>

Cartograms

Cartograms with d3 & TopoJSON

Scale by in calculated in 0.1 seconds

Washington: +3,626



<http://prag.ma/code/d3-cartogram/>

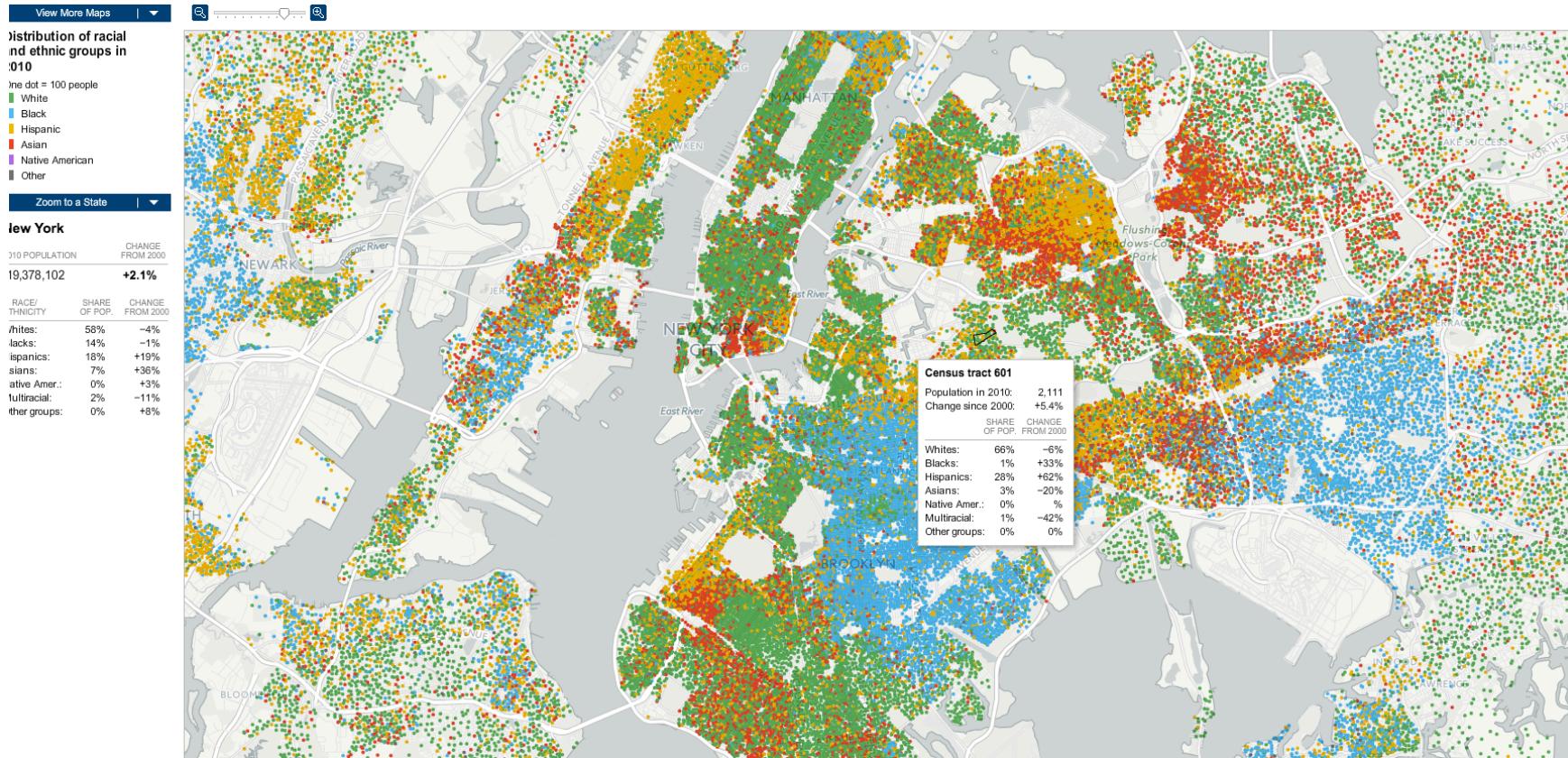
Dot distribution

The New York Times

Mapping the 2010 U.S. Census

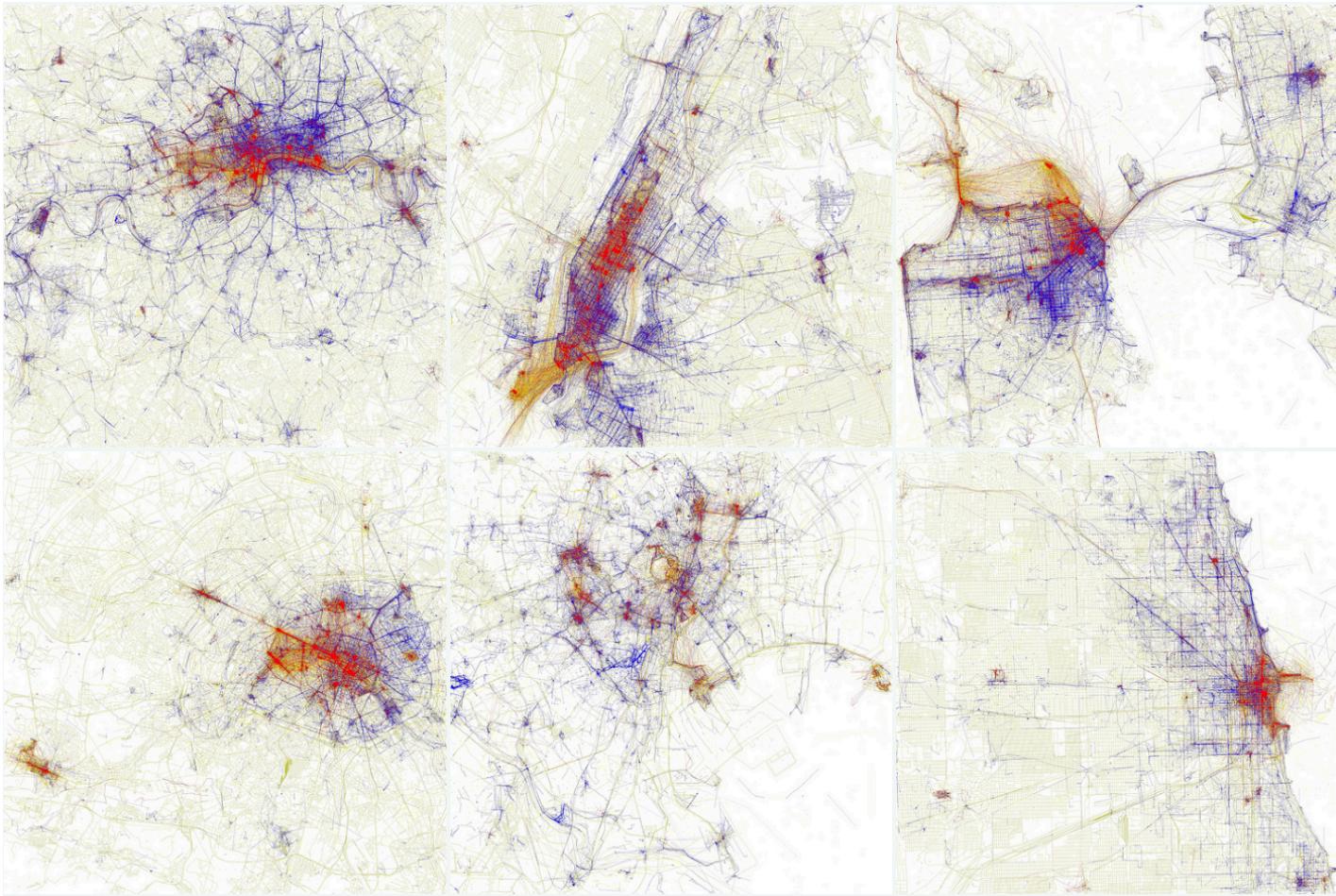
Browse population growth and decline, changes in racial and ethnic concentrations and patterns of housing development.

Share this view on [Twitter](#) or [Facebook](#)



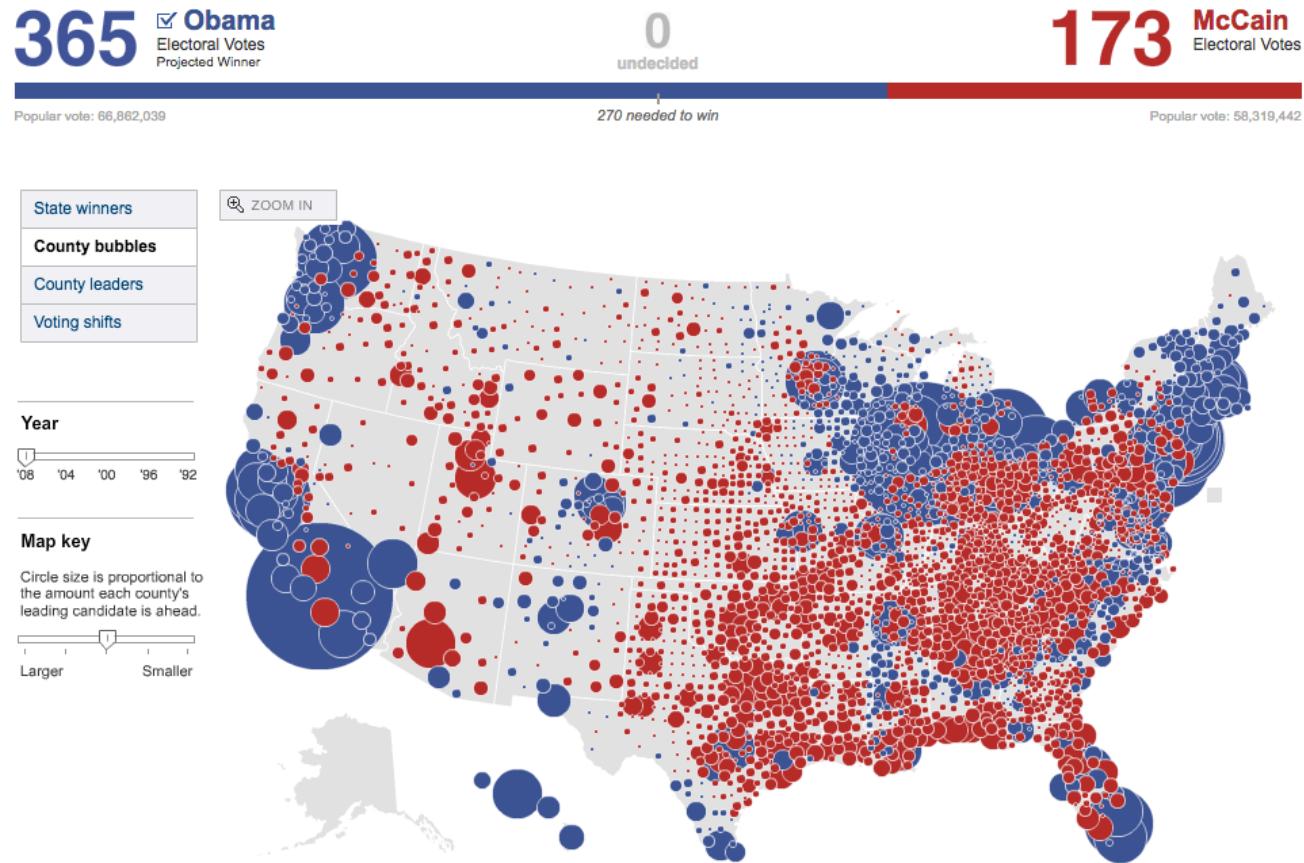
<http://projects.nytimes.com/census/2010/map>

Lines distribution



<https://www.flickr.com/photos/walkingsf/albums/72157624209158632>

Graduated Symbol Map



<http://elections.nytimes.com/2008/results/president/map.html>

Tile Map Server

Tile Map Server

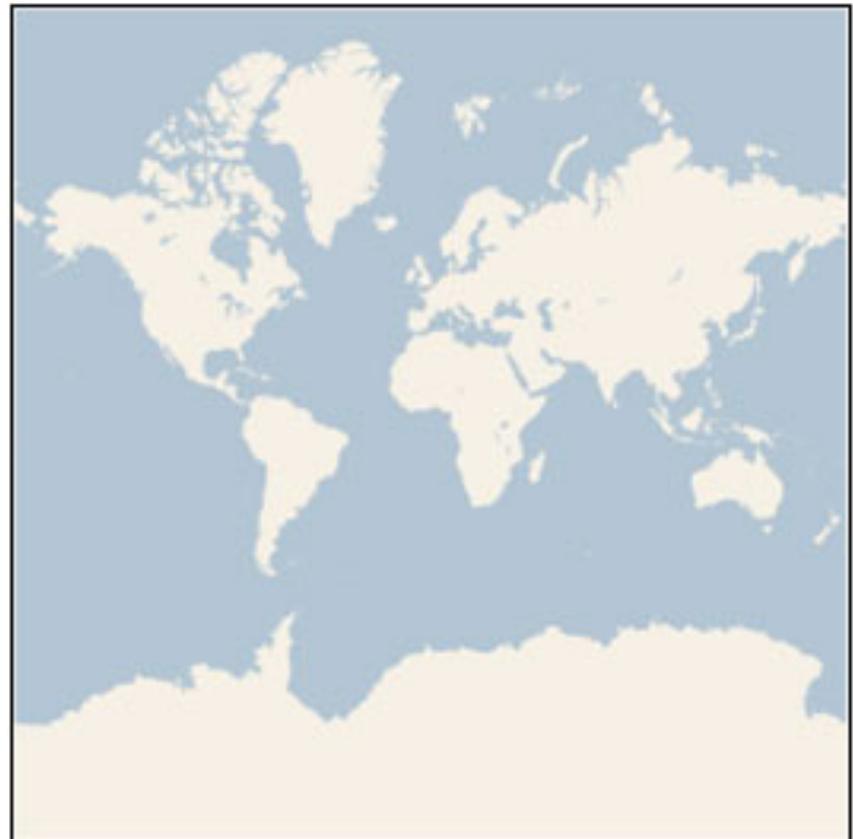
- An efficient solution to publish maps on the web
 - Complexity in space (rather than in time)
 - Used by many providers:
 - Google Maps, Bing, Yahoo Maps, OpenStreetMaps
- Maps is generated once for all level of zoom and then sliced into **tiles**
- A map for a finite set of zoom levels

Tile Map Server (2)

- To simplify coordinate mapping: cylindrical projection
- Two main reference systems:
 - Sphere Mercator (53004)
 - World Mercator (54004)
- Mercator Cyndric projection
 - Meridians are parallels
 - Conformal (preserves shapes)
 - Preserves directions

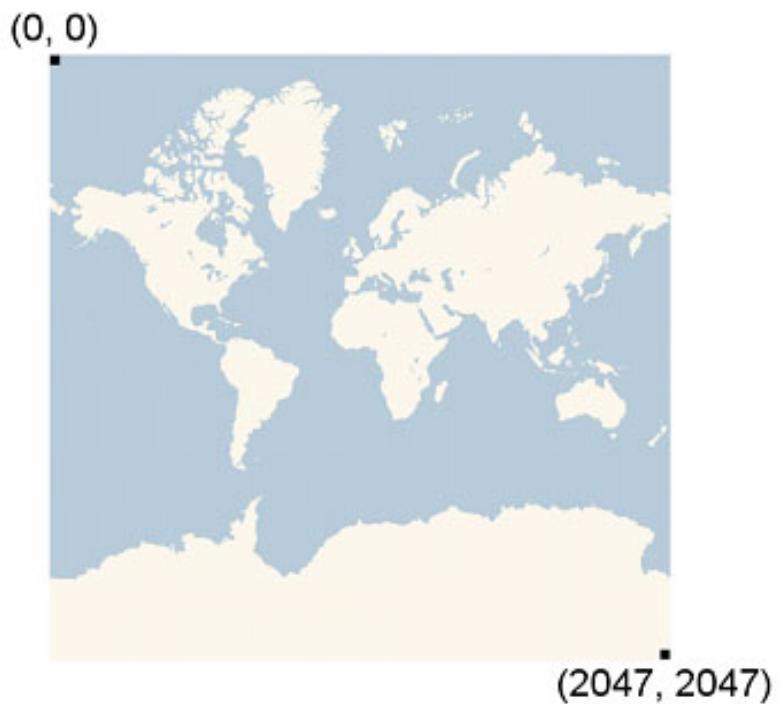
Tile Map Server (3): scale

- Hierarchy division of plane
- Every tile (any zoom) has a fixed dimension: 256x256
- Each zoom level increases (doubles) the number of tiles
- At level 1: only 4 tiles



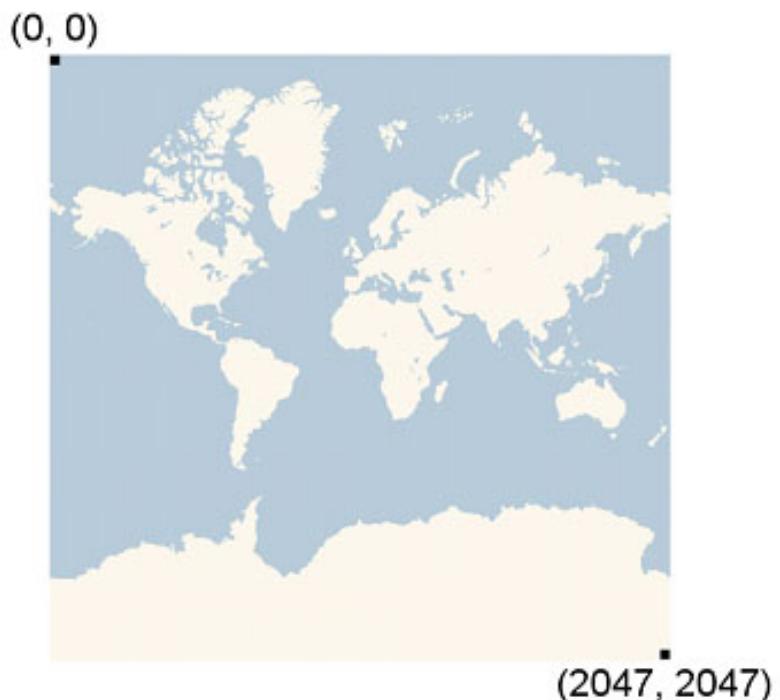
Tile Map Server (4): scale

- At each zoom level, the number of tiles increases
 - Every tile at level n generates 4 tiles at level $n+1$
- At level n tiles cover $256 * 2^n$ pixel
- For example, at level 3 map has a side $256 * 2^3 = 2048$ pixel



Tile Map Server (5): coordinates

- Given coordinate (lat,lon) and zoom level n, how to determine position on the image?
- Which tile correspond to coordinate?



```
pixelX = ((longitude + 180) / 360) * 256 * 2^level
```

```
pixelY = (1 - log(tan(Latitude)+ sinh(Latitude)) / pi))/2 * 256 * 2 ^ level
```

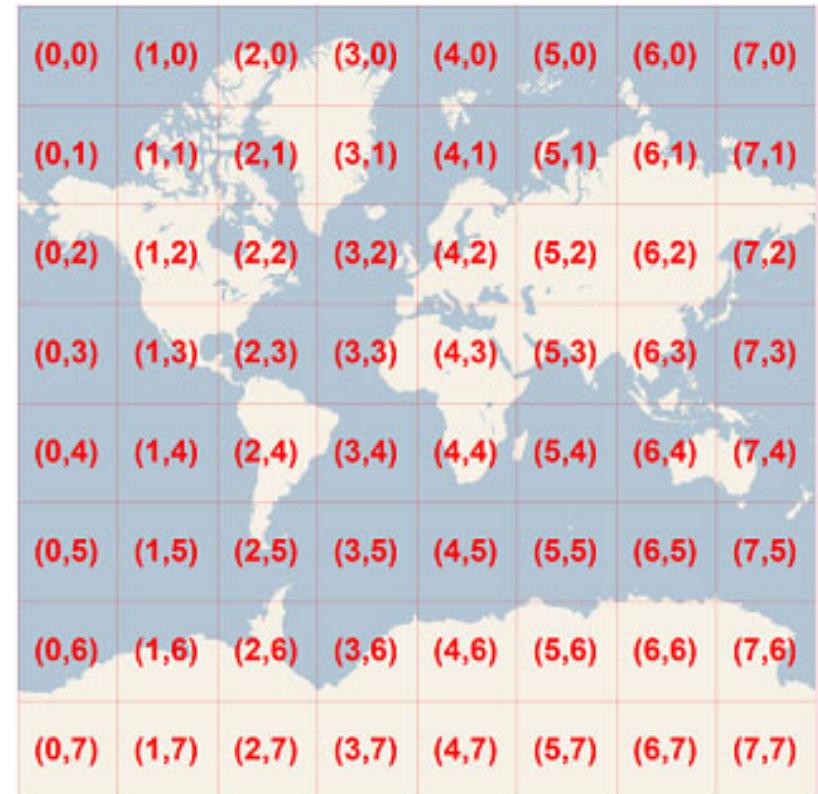
Tile Map Server (6): tile number

- Given pixelX e pixelY
- Which tile contains that pixel?

$\text{tileX} = \text{floor}(\text{pixelX} / 256)$

$\text{tileY} = \text{floor}(\text{pixelY} / 256)$

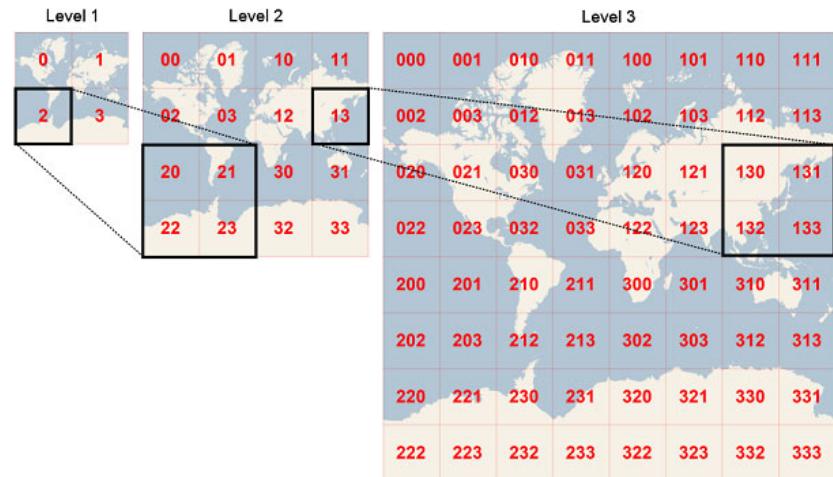
- URL to tile:
 - `/zoom/tx/ty`
 - quadkey



<http://otile2.mqcdn.com/tiles/1.0.0/osm/1/0/0.png>

Tile Map Server (7): quadkey

- Used by Bing
- Length of the key corresponds to the zoom level



tileX = 3 = 011

tileY = 5 = 101

quadkey = 100111 = 2134 = "213"

Tile Map Server (8): zoom in

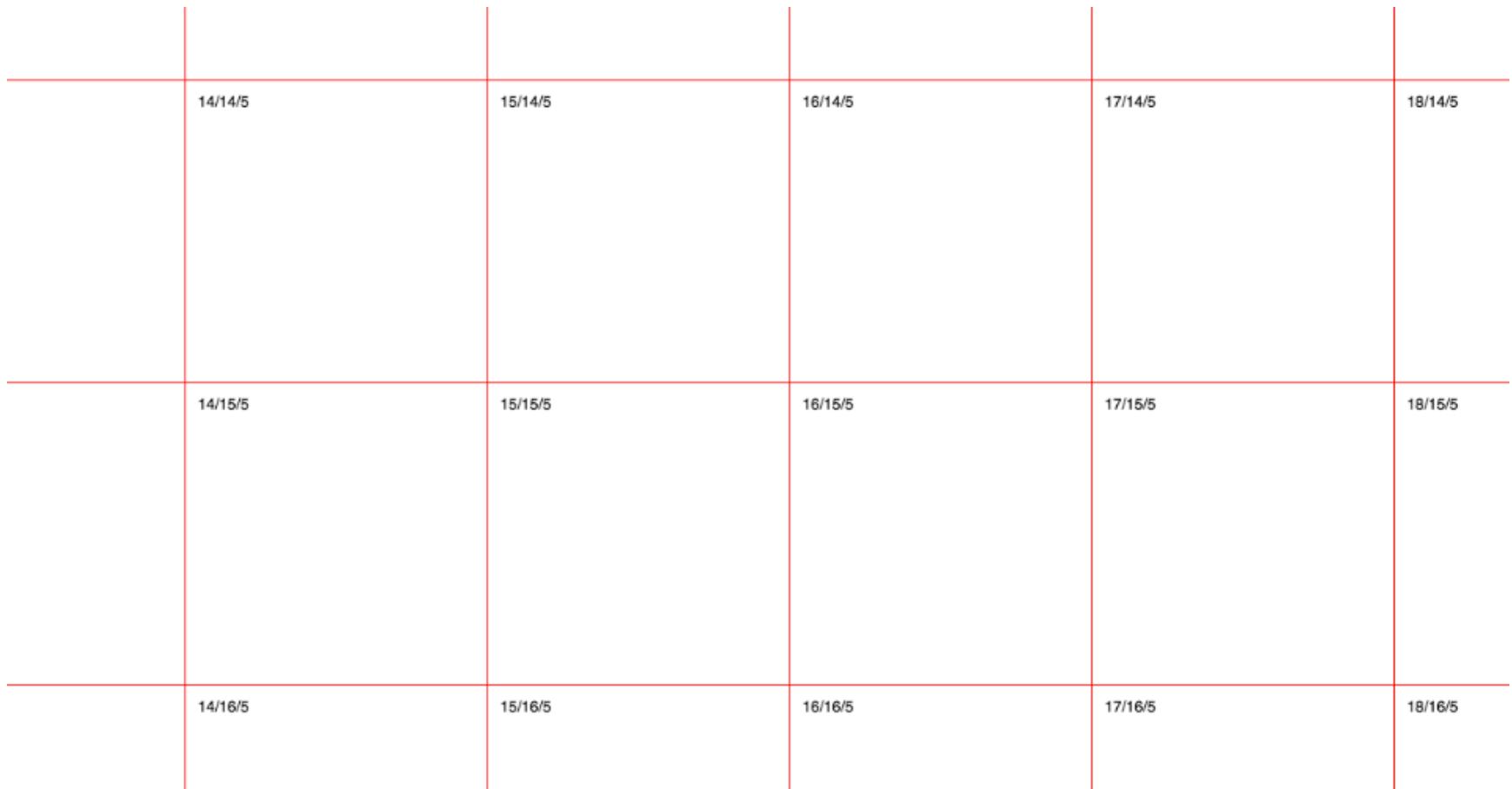
- Given a tile a zoom level n
- Successive tile at level $n+1$ are:
 - $2x, 2y$
 - $2x+1, 2y$
 - $2x, 2y+1$
 - $2x+1, 2y+1$



/3/4/2.png



Tile Viewer Example



<https://bl.ocks.org/mbostock/f9f91fd9148bdc5aa6db>



TILE SERVICES

Leaflet.js

 An Open-Source JavaScript Library for Mobile-Friendly Interactive Maps

Overview Features Tutorials API Download Plugins Blog GitHub Twitter Forum

Leaflet is a modern open-source JavaScript library for mobile-friendly interactive maps. It is developed by [Vladimir Agafonkin](#) with a team of dedicated [contributors](#). Weighing just about 33 KB of JS, it has all the [features](#) most developers ever need for online maps.

Leaflet is designed with *simplicity, performance and usability* in mind. It works efficiently across all major desktop and mobile platforms out of the box, taking advantage of HTML5 and CSS3 on modern browsers while still being accessible on older ones. It can be extended with a huge amount of [plugins](#), has a beautiful, easy to use and [well-documented API](#) and a simple, readable [source code](#) that is a joy to [contribute](#) to.

Used by: Flickr foursquare Pinterest craigslist Data.gov IGN Wikimedia OSM Meetup WSJ Mapbox CartoDB GIS Cloud ...



A pretty CSS3 popup.
Easily customizable.

Leaflet | © OpenStreetMap contributors

Leaflet.js - APIs

An Open-Source JavaScript Library for Mobile-Friendly Interactive Maps

Overview	Features	Tutorials	API	Download	Plugins	Blog		 GitHub	 Twitter	 Forum
----------	----------	-----------	-----	----------	---------	------	--	--	---	---

Map

[Usage example](#)

[Creation](#)

[Options](#)

[Events](#)

Map Methods

[For modifying map state](#)

[For getting map state](#)

[For layers and controls](#)

[Conversion methods](#)

[Other methods](#)

Map Misc

[Properties](#)

[Panes](#)

UI Layers

[Marker](#)

[Popup](#)

Raster Layers

[TileLayer](#)

[TileLayer.WMS](#)

[TileLayer.Canvas](#)

[ImageOverlay](#)

Vector Layers

[Path](#)

[Polyline](#)

[MultiPolyline](#)

[Polygon](#)

[MultiPolygon](#)

[Rectangle](#)

[Circle](#)

Other Layers

[LayerGroup](#)

[FeatureGroup](#)

[GeoJSON](#)

Basic Types

[Point](#)

[Bounds](#)

[Icon](#)

[Path](#)

[DivIcon](#)

Controls

[Control](#)

[Zoom](#)

[Attribution](#)

[Layers](#)

Events

[Event methods](#)

[Event objects](#)

[Class](#)

[Browser](#)

[Util](#)

[Transformation](#)

[LineUtil](#)

[PolyUtil](#)

DOM Utility

[DomEvent](#)

[DomUtil](#)

[PosAnimation](#)

[Draggable](#)

Interfaces

[IHandler](#)

[ILayer](#)

[IControl](#)

[IProjection](#)

[ICRS](#)

[Misc](#)

[global switches](#)

[noConflict](#)

[version](#)

Leaflet.js

- A valid tool to provide tile-based maps
 - Open Source
 - Open Data (<http://tools.geofabrik.de/mc/>)
 - Free
- Easy to use API
- Lightweight lib (only 64k)
- Support mobile applications
- Alternative to Google Maps
(<http://flink.com.au/tips-tricks/27-reasons-not-to-use-google-maps>)

Free Tiles Providers

- OpenStreetMap
 - Some issues for high traffic services
- MapQuest Open License
 - Free, by attribution
 - Special configuration for heavy usage
- MapBox
 - Free tier
 - Customizable design (see next slide)
 - Same family as Leaflet.js



Access blocked

This application is blocked for overusing OpenStreetMap's volunteer-run servers:
wiki.osm.org/Blocked

Commercial Tile Providers

- CloudMade
 - Mirror of OSM data till few years ago
 - Leaflet was born here
 - \$30 per 1M tiles
- MapBox
 - Free for low traffic
 - \$30 for 900k tiles

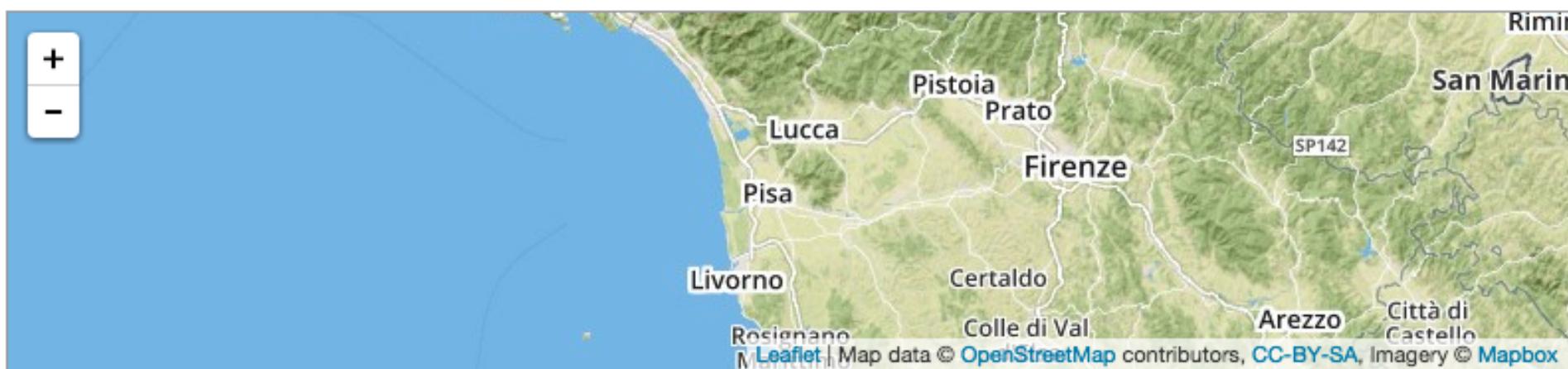
Easy to install/use

- HTML (Setting the stage)
 - Link CSS (via CDN)
 - `<link rel="stylesheet" href="http://cdnjs.cloudflare.com/ajax/libs/leaflet/0.7.3/leaflet.css" />`
 - Link JS (via CDN)
 - `<script src="http://cdnjs.cloudflare.com/ajax/libs/leaflet/0.7.3/leaflet.js"></script>`
 - Create a div to contain the map
 - `<div id="map"></div>`
 - Set height for the container
 - `#map { height: 180px; }`

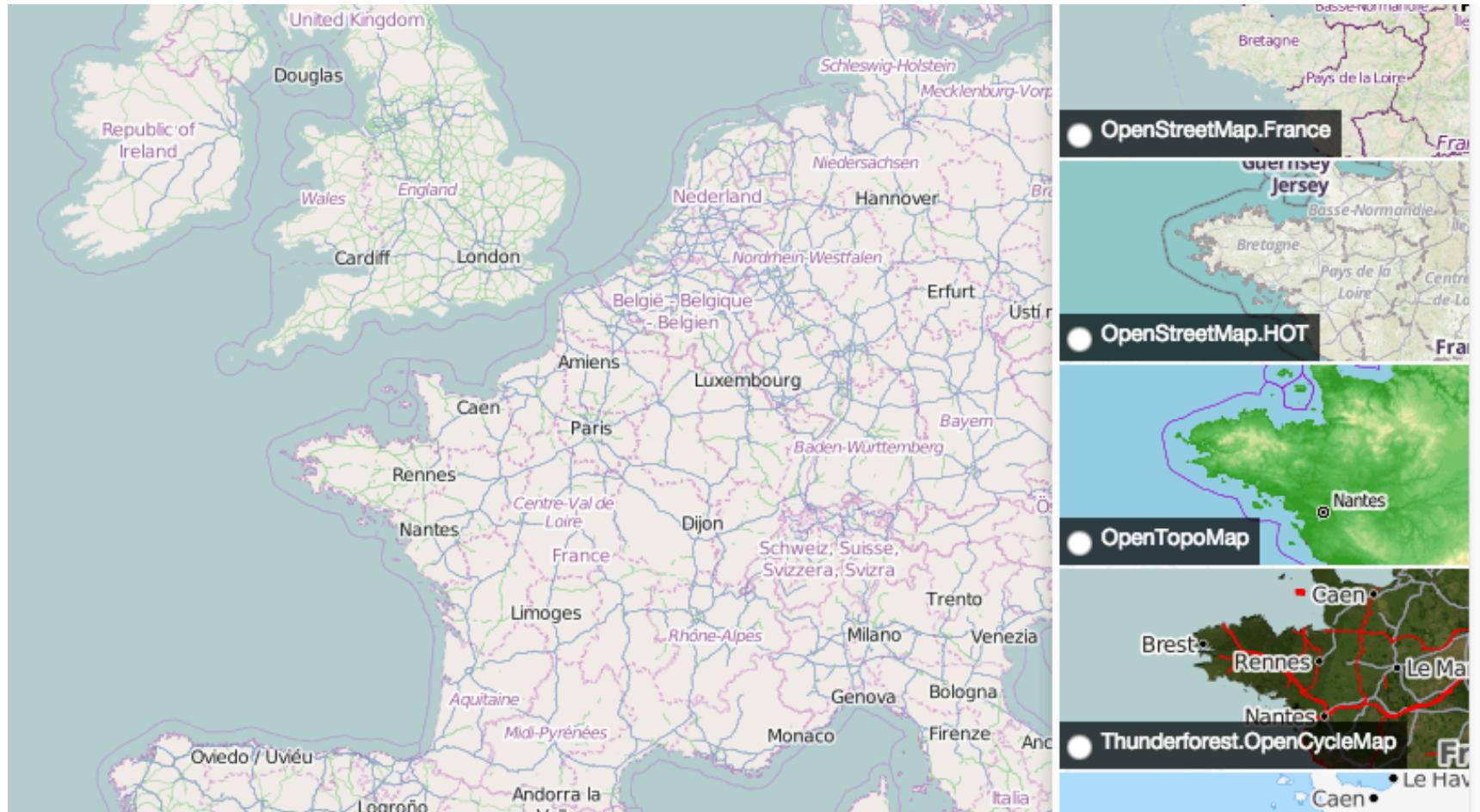
Easy to install/use

- Create an object to handle the map

- ```
var map = L.map('map').setView([51.505, -0.09], 13);
```



# Tile map providers



<http://leaflet-extras.github.io/leaflet-providers/preview/>

<http://maps.stamen.com>

# Markers and geometries

- var marker = L.marker([51.5, -0.09]).addTo(map);
- var circle = L.circle([51.508, -0.11], 500, {  
    color: 'red',  
    fillColor: '#f03',  
    fillOpacity: 0.5  
}).addTo(map);
- var polygon = L.polygon([  
    [51.509, -0.08],  
    [51.503, -0.06],  
    [51.51, -0.047]  
]).addTo(map);

# Interactions

- marker.bindPopup("<b>Hello world!</b><br>I am a popup.").openPopup();
- circle.bindPopup("I am a circle.");
- polygon.bindPopup("I am a polygon.");

# Event handling

- ```
function onMapClick(e) {  
    alert("You clicked the map at " + e.latlng);  
}  
map.on('click', onMapClick);
```
- ```
var popup = L.popup();
function onMapClick(e) {
 popup
 .setLatLng(e.latlng)
 .setContent("You clicked the map at " + e.latlng.toString())
 .openOn(map);
}
map.on('click', onMapClick);
```

# Other Examples

- Mobile app
  - <http://leafletjs.com/examples/mobile.html>
- GeoJSON
  - <http://leafletjs.com/examples/geojson.html>
  - <http://geojson.io/>
- Tutorials
  - <http://leafletjs.com/examples.html>