Working with Geospatial Data

Ben Sullins
DATA GEEK

@bensullins www.bensullins.com

What You'll Learn

- Geospatial Data Concepts
- Finding Geospatial Data
- Working with GeoJSON
- Working with TopoJSON
- Final Project Data

Cogsley Services Inc.

Founded in 2008

Technical Consulting Firm

Needs to understand customer locations for targeted marketing efforts

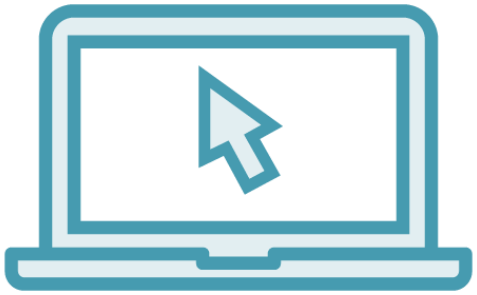
- Exported CSV Sales Data

- Need to combine with map data

Geospatial Data Concepts

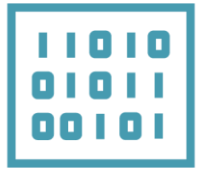
Geospatial Data Concepts

Shapefiles

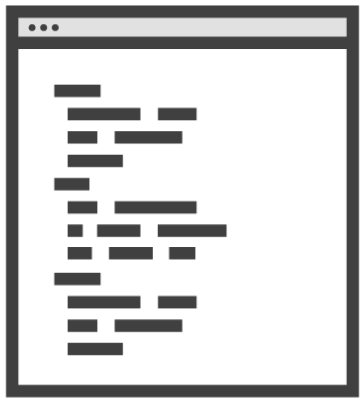


GDAL

GeoJSON



TopoJSON



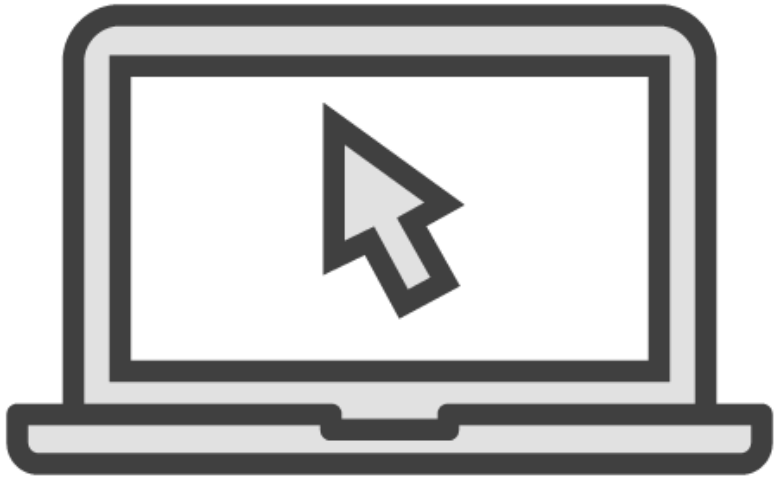
Shapefiles:

A geospatial vector format used by many GIS software packages

.shp — shape format; the feature geometry itself

.shx — shape index format, allows seeking fwd and backwards quickly

.dbf — attribute format; columnar attributes for each shape



GDAL:

Geospatial Data Abstraction Library

Raster and Vector formats

Open Source

Utility Programs (ex. ogr2ogr)

Open Source Geospatial Foundation

GeoJSON and TopoJSON

GeoJSON

Discrete Geometry Objects

Feature Collections

Name/Value Paris

TopoJSON

Extension of GeoJSON

Topology Encoding

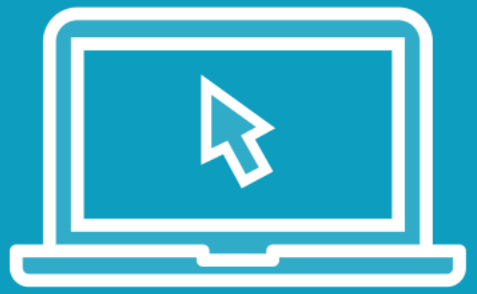
Shared Line Segments (arcs)

Eliminate Redundancy

~80% smaller files

Finding Geospatial Data

Demo



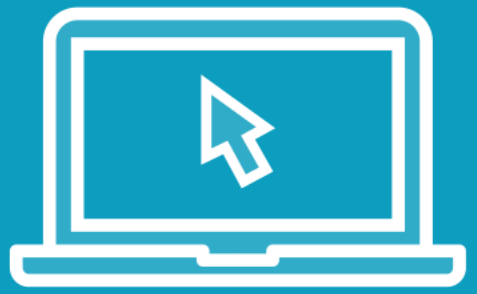
Review popular online sources

Choosing a Resolution

Download Shape Files

Working with GeoJSON

Demo



Installing Tools

Checking our Work

Converting Shapefiles

Clipping Shapefiles

Filtering Shapefiles

Installing Tools

```
# Mac OSX El Capitan (10.11.4)
# Homebrew already installed
$ brew install gdal
# Check Versions
$ which ogr2ogr
$ which topojson
```

Converting Shape Files

```
# ogr2ogr -f <format> <output> <input>
$ ogr2ogr
    -f GeoJSON // output format
    counties.json // output file
    build/cb_2014_us_county_20m.shp // input file
```

Clipping Shape Files

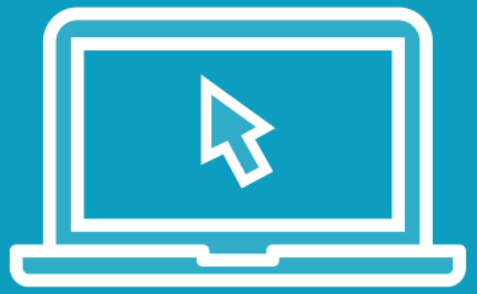
```
# Browse to http://bit.ly/flicker-geo-api for bounding box
$ ogr2ogr
    -f GeoJSON // output format
    counties-clipped.json // output file
    build/cb_2014_us_county_20m.shp // input file
    -clipsrc -124.4096 32.5343 -114.1308 42.0095 // bounding box
```

Filtering Shape Files

```
# Browse to http://bit.ly/usa-fips for fips codes
$ ogr2ogr
    -f GeoJSON // output format
    counties-filtered.json // output file
    build/cb_2014_us_county_20m.shp // input file
    -where "STATEFP='06'" // filtering
```

Working with topojson

Demo



Convert GeoJSON to topojson

Adding a D3 Projection

Simplifying the Output

Converting GeoJSON to topojson

```
# topojson -o <output file> <input file>
$ topojson
    -o topo-counties.json //-o <output file>
    build/cb_2014_us_county_20m.shp // <input file>
```

Adding a D3 Projection

```
# --projection project spherical input geometry using a D3 geographic
projection

$ topojson

    -o topo-counties-projected.json // <output file>

    --projection='width = 960, height = 600, d3.geo.albersUsa()

        .scale(1280)

        .translate([width / 2, height / 2])' // D3 projection

    build/cb_2014_us_county_20m.shp //input file
```

Simplifying Output

```
# --simplify precision threshold for Visvalingam simplification
# example: https://bost.ocks.org/mike/simplify/
$ topojson
    -o topo-counties-simplified.json // <output file>
    --projection='width = 960, height = 600, d3.geo.albersUsa()
        .scale(1280)
        .translate([width / 2, height / 2])' // D3 projection
    --simplify=.5 // simplify parameter
    -- counties=build/cb_2014_us_county_20m.shp //retain all counties
```

Final Project Data

Demo

Creating a Makefile

Acquiring Data

Merging CSV and topojson

Building states.json

Building us.json

Creating a Makefile

```
$ nano Makefile

    all: us.json #run us.json and all dependencies
    clean:  rm -rf -- us.json build #function to rebuild from scratch
    .PHONY: all clean #define recipe names to be executed
    recipe1:…
    recipe2:…
    etc:…
```

Acquiring Data

```
#unzip the file
build/gz_2010_us_050_00_20m.shp: build/gz_2010_us_050_00_20m.zip
    unzip -od $(dir $@) $<
    touch $@
#download the shapefile
build/gz_2010_us_050_00_20m.zip:
    mkdir -p $(dir $@)
    curl -o $@ http://www2.census.gov/geo/tiger/GENZ2010/$(notdir $@)
```

Merging CSV and topojson

```
build/counties.json: build/gz_2010_us_050_00_20m.shp profit-by-county.csv
    node_modules/.bin/topojson \
    -o $@ \
    --id-property='STATE+COUNTY,id' \
    --external-properties='profit-by-county.csv' \
    --properties='profit=+profit,name=label,recno=id,orders=+orders' \
    --projection='width = 960, height = 600, d3.geo.albersUsa() \
        .scale(1280) \
        .translate([width / 2, height / 2])' \
    --simplify=.5 \
    -- counties=$<
```

Creating States Layer

```
build/states.json: build/counties.json
    node_modules/.bin/topojson-merge \
    -o $@ \ # build/states.json
    --in-object=counties \
    --out-object=states \
    --key='d.id.substring(0, 2)' \
    -- $< # build/counties.json
```

Creating Nation Layer

```
us.json: build/states.json
    node_modules/.bin/topojson-merge \
        -o $@ \ # us.json
        --in-object=states \
        --out-object=nation \
        -- $< # build/states.json
```

Where to Find More

Where to Find More

Pluralsight Courses

- D3.js Fundamentals

External

- http://bit.ly/topojson-ref
- http://bit.ly/census-maps
- http://bit.ly/natural-earth
- http://bit.ly/flicker-geo-api
- https://github.com/mbostock/us-atlas