Exercise 2: Basic Web Mapping with GeoJSON

Goal

The goal of this exercise is to create a basic web map using data converted to GeoJSON.

Part 1: Unpack the Sample Geodata

Please unpack the contents of the “geodata_eu.zip” (sample geodata for your exercises) archive into the C:\workspace directory of your computer.

Then briefly study the sample datasets by opening them in QGIS (an open source desktop GIS software). You can use the version of QGIS which is installed locally on your computer (from “Program Files”).

(Note: in case you would like to redo the exercises at home on your own computer, you may download QGIS from www.qgis.org).

Part 2: Convert Data to GeoJSON and Add them to your Map

Load in QGIS the vector datasets eu_cities.shp and eu_countries.shp from the uncompressed “eu_geodata” folder. Then, convert these two datasets into the GeoJSON format by right clicking on the layer in QGIS, then choosing the contextual menu “Save as...” and selecting GeoJSON as the output format.

There are different ways of loading the exported datasets in the web map. We will include this data in a static manner, by using the fact that GeoJSON is in fact JavaScript.

First, change the extension of the exported eu_cities GeoJSON file to “.js”. Then, open this file in a text editor and add the GeoJSON content to a variable as shown below:

```
var eu_cities = {content of the exported geojson file};
```

Next, copy or move the resulted javascript file inside the mymap folder. It is recommended that you place this data in separate folder in your web map project, for example in a “geojson” folder that you will have to create. Then open your map (“index.html”) in a text editor and link the resulted javascript file containing the eu_cities data:

```
<!-- GeoJSON as JS file -->
<script type="text/javascript" src="geojson/eu_cities.js"></script>
```

At this point, the data can be added as an additional layer to the map:

```
var eu_cities = L.geoJson(eu_cities).addTo(map);
```

Then please use a similar procedure to add the eu_countries_them data (that you also previously exported to GeoJSON) to the map.

Finally, check the result in the web browser.

Part 3: Add a Layer Control

The goal of this part of the exercise is to add the capability to turn your layers on and off.

In order to achieve this, please first consult the Leaflet API documentation (available at http://leafletjs.com/reference.html), search for the subsection “Controls” and click on the “Layers” link to jump to the corresponding documentation for *Control.Layers*. Read the documentation and eventually also briefly check the source of the detailed example linked on the documentation page (http://leafletjs.com/examples/layers-control.html). Please also make sure that you read about the options available for the layer control, such as “position”, “collapsed”, or “autoZIndex”.

Armed with this information, now add a layer control to your map similarly as shown below:

```
// Add the layer control
// see http://leafletjs.com/reference.html#control-layers
var baseLayers = {
    "Base": baseLayer
    };
var overlays = {
    "Countries": eu_countries_them,
    "Cities": eu_cities
    };
L.control.layers(baseLayers, overlays, {
    position: 'topright',
    collapsed:false,
    autoZIndex: true
    }).addTo(map);
```

Part 4: Define a Choropleth Map for the eu_countries_them Layer

First of all, please consult the step-by-step Leaflet example about how to create choropleth maps with Leaflet at http://leaflet.com/examples/choropleth.html. Please pay particular attention to the functions “getColor(d)” and “style(feature)”, and notice how the later function is using the former to define the “fillColor” of each feature. Finally, notice how the style function is used before adding the layer to the map.

Now, as you may have already noticed, the eu_countries_them data that you have previously included in your map contains some attribute data that we can visualize as a choropleth map. The attribute that we choose to visualize is the percentage of the renewable energy for each country in Europe (“PERC_RENEW”).

Spoiler:

```
 // Define the different classes and the appropriate colors
function getColor(percentage){
    if (percentage >0 && percentage <=10){return "#ffffcc"}
    else if (percentage >10 && percentage <=20){return "#c2e699"}
    else if (percentage >20 && percentage <=30){return "#78c679"}
    else if (percentage >30){return "#238443"}
    else {return '#ffffff'}
}
// Define the style of the polygons
function style(feature){
    return {
        fillColor: getColor(feature.properties.PERC_RENEW),
        color: '#ffffff',
        weight: 1,
        opacity: 1,
        fillOpacity: 0.8
    };
}
//Modify the layer definition to include the style
var eu_countries_them = L.geoJson(eu_countries_them, {
    style:style}).addTo(map);
```

Part 5: Adding Interactivity (optional)

In this part of the exercise you will add some interactivity to your choropleth map. For this, follow the instructions from the sections “Adding Interaction” and “Custom Info Control” available in the same Leaflet example that you consulted for the part 4 of the exercise (http://leaflet.com/examples/choropleth.html). If needed, also search for, and read about additional information related to javascript “events”, “event listeners” and Cascading Style Sheets (CSS).

If you do not finish the exercise in time, please feel free to consult the “spoiler” from the next page or the sample solution for exercise 2.

Spoiler:

```
// Info done using a custom control
var info = L.control({position:'bottomright'});

info.onAdd = function (map) {
    this._div = L.DomUtil.create('div', 'info'); // create a div with a class "info"
    this.update();
    return this._div;
};
// method that we will use to update the control based on feature properties passed
info.update = function (props) {
    this._div.innerHTML = '<h4>Renewable energy  </h4>' +  (props ?
        '<b>' + props.CTRY_NAME + '</b><br />' + props.PERC_RENEW + ' %'
        : 'Hover over a country <br /> %');
};
info.addTo(map);

// Highlight functions: event listeners for setting and resetting the style
function highlightFeature(e) {
    var layer = e.target;
    layer.setStyle({
        weight: 5,
        color: '#DD6565',
        dashArray: '',
        fillOpacity: 0.6
    });
    if (!L.Browser.ie && !L.Browser.opera) {
        layer.bringToFront();
    }
        info.update(layer.feature.properties); // added afterwards
}
function resetHighlight(e) {
    eu_countries_them.resetStyle(e.target);
    info.update();// added afterwards
}
// Zoom in: event listener for click
function zoomToFeature(e) {
    map.fitBounds(e.target.getBounds());
}

// Using the onEachFeature option to add the listeners to our layer
function onEachFeature(feature, layer) {
    layer.on({
        mouseover: highlightFeature,
        mouseout: resetHighlight,
        click: zoomToFeature
    });
}

// Modify the layer definition
var eu_countries_them = L.geoJson(eu_countries_them, {
    style:style,
    onEachFeature:onEachFeature
    }).addTo(map);
```

CSS (to add in the index.html file within the <style></style> tag

```
/* Defines the style of the info element */
.info{
    padding:6px 8px;
    font:14px/16px Arial, Helvetica, sans-serif;
    background:white;
    background:rgba(255, 255, 255, 0.8);
    box-shadow:0 0 15px rgba(0, 0, 0, 0.2);
    border-radius:5px;
}
.info h4{
    margin:0 0 5px;
    color:#777;
}
```

Discussion of the Exercise

Congratulations, you have acquired the skills to read the documentation of a mapping framework and to create web maps using data converted to the GeoJSON format.

Additional Things to Try at Home:

In order to further improve your skills, here is a list of additional things you can try later on:

- add a legend for the choropleth (see http://leafletjs.com/examples/choropleth.html)
- define custom markers for the GeoJSON points (see the pointToLayer option from http://leafletjs.com/reference.html#geojson and how to define a custom marker http://leafletjs.com/reference.html#marker)
- label the cities (see https://github.com/Leaflet/Leaflet.label)