Line\_graph.js

// set the dimensions and margins of the graph

var margin = {

top: 0,

right: 20,

bottom: 10,

left: 40

},

width = 600 - margin.left - margin.right,

height = 400 - margin.top - margin.bottom;

// parse the date / time

var parseTime = d3.timeParse("%Y");

// set the ranges

var x = d3.scaleTime().range([0, width]);

var y = d3.scaleLinear().range([height, 0]);

// define the line

var valueline = d3.line()

.x(function(d) {return x(d.years);})

.y(function(d) {return y(d.Chinatown);});

// append the svg obgect to the body of the page

// appends a 'group' element to 'svg'

// moves the 'group' element to the top left margin

var svg = d3.select("#chart")

.attr("width", width + margin.left + margin.right)

.attr("height", height + margin.top + margin.bottom);

// Get the data

d3.csv("./data/2010-2017\_review.csv", function(error, data) {

if (error) throw error;

//format the data

data.forEach(function(d) {

d.year = parseTime(d.years);

d.Chinatown = +d.Chinatown;

});

// Scale the range of the data

x.domain(d3.extent(data, function(d) {

return d.years;

}));

y.domain([0, d3.max(data, function(d) {

return d.Chinatown;

})]).nice();

var tooltip = d3.select("body")

.append("div")

.append("class", "tooltip")

.append("opacity", 1)

.style("background-color", "white")

.style("border", "solid")

.style("border-width", "2px")

.style("border-radius", "5px")

.style("padding", "5px")

.style("position", "absolute")

.style("z-index", "10")

.style("visibility", "hidden");

// Add the valueline path.

svg.append("path")

.data([data])

.attr("class", "line2")

.attr("d", valueline)

.attr("transform", "translate(62" + ", 70" + ")")

.on("mouseover", function(d,i) {

tooltip

.style("opacity", 1)

.style("visibility", "visible")

.html(d.count);}) ;

// Add the X Axis

var marginB = 460;

svg.append("g")

.attr("transform", "translate(60," + marginB + ")")

.call(d3.axisBottom(x)

.tickFormat(d3.format(".4r")));

svg.append("text")

.attr("class", "axisLabel")

.attr("transform",

"translate(" + (width / 2.5 + 110) + " ," +

(height + 110) + ")")

.style("text-anchor", "middle")

.text("Years");

svg.append("g")

.attr("class", "y axis")

.attr("transform", "translate(60" + ", 70" + ")")

.call(d3.axisLeft(y)

.ticks(5));

svg.append("text")

.attr("class", "axisLabel")

.attr("transform", "rotate(-90)")

.attr("y", margin.left - 40)

.attr("x", 0 - (height / 2) - 78)

.attr("dy", "1em")

.style("text-anchor", "middle")

.text("Number of Reviews");

drawChart(field);

});

var drawChart = function(field) {

d3.select("#chart").select(".y.axis").remove();

d3.select("#chart").select("path.line2").remove();

svg.select("text")

.attr("x", width/2)

.attr("y", -400)

.text(field);

// Get the data

d3.csv("./data/2010-2017\_review.csv", function(error, data) {

if (error) throw error;

//format the data

data.forEach(function(d) {

d.year = parseTime(d.years);

d[field] = +d[field];

});

x.domain(d3.extent(data, function(d) {

return d.years;

}));

y.domain([0, d3.max(data, function(d) {

return d[field];

})]).nice();

svg.append("g")

.attr("class", "y axis")

.attr("transform", "translate(65" + ", 70" + ")")

.call(d3.axisLeft(y)

.ticks(5));

var valueline = d3.line()

.x(function(d) {return x(d.years);})

.y(function(d) {return y(d[field]);});

// Add the valueline path.

svg.append("path")

.data([data])

.attr("class", "line2")

.attr("d", valueline)

.attr("transform", "translate(62" + ", 70" + ")");

})};

**Sf\_chrolopeth.js**

// load data with queue

var url1 = "./data/neighborhood.geojson";

var url2 = "./data/listing\_count.json";

var url3 = "./data/2010-2017\_review.csv";

var q = d3\_queue.queue(1)

.defer(d3.json, url1)

.defer(d3.json, url2)

//.defer(d3.csv, url3)

.awaitAll(draw);

function draw(error, data) {

"use strict";

// important: First argument it expects is error

if (error) throw error;

// initialize the Bayview as the default neighborhood

var field = "Bayview";

//drawChart(field);

var margin = 50,

width = 450 - margin,

height = 500 - margin;

var colorScale = d3.scaleThreshold()

.domain([1, 25, 50, 100, 200, 300, 700])

//.range(d3.schemeReds[7]);

.range(["#fff7ec","#fee8c8","#fdd49e","#fdbb84","#fc8d59","#ef6548","#d7301f"]);

// create a projection properly scaled for SF

var projection = d3.geoMercator()

.center([-122.433701, 37.767683])

.scale(175000)

.translate([width / 1.5, height / 1.74]);

// create a path to draw the neighborhoods

var path = d3.geoPath()

.projection(projection);

var state = null;

// create and append the map of SF neighborhoods

var map = d3.select('#map')

.selectAll('path')

.data(data[0].features)

.enter()

.append('path')

.attr("cx", function(d) {return d.x;})

.attr('d', path)

.style('stroke', 'black')

.style('stroke-width', 0.75)

.on("click", function(d) {

d3.selectAll("path").style("opacity", 0.2);

if (state == 0) {

d3.select(this).style("opacity", 1);

state = 1;

} else {

d3.selectAll("path").style("opacity", 1);

state = 0;

}

var region = d.properties.neighbourhood;

var listing = d.properties.count;

d3.select("#map path." + region);

//d3.select("#map path." + listing);

drawChart(region);

} );

var tooltip = d3.select("body")

.append("div")

.append("class", "tooltip")

.append("opacity", 1)

.style("background-color", "white")

.style("border", "solid")

.style("border-width", "2px")

.style("border-radius", "5px")

.style("padding", "5px")

.style("position", "absolute")

.style("z-index", "10")

.style("visibility", "hidden");

var threshold = d3.scaleThreshold()

.domain([1, 25, 50, 100, 200, 300, 700])

.range(["#fff7ec","#fee8c8","#fdd49e","#fdbb84","#fc8d59","#ef6548","#d7301f"]);

var x = d3.scaleLinear()

.domain([1, 700])

.range([0, 240]);

var xAxis = d3.axisBottom(x)

.scale(x)

//.orient("bottom")

.tickSize(7)

.tickValues(threshold.domain());

var g = d3.select('g');//.call(xAxis)

//.attr("x", 0 - (height / 2));

g.select(".domain");

g.selectAll("rect")

.data(threshold.range().map(function(color) {

var d = threshold.invertExtent(color);

if (d[0] == null) d[0] = x.domain()[0];

if (d[1] == null) d[1] = x.domain()[1];

return d;

}))

.enter().insert("rect", ".tick")

.attr("height", 8)

.attr("y", 80)

.attr("x", function(d) { return x(d[0]); })

.attr("width", function(d) { return x(d[1]) - x(d[0]); })

.attr("fill", function(d) { return threshold(d[0]); })

g.append("text")

.attr("fill", "#000")

.attr("font-weight", "bold")

.attr("text-anchor", "start")

.attr("y", 75)

.text("Number of Avaible Airbnb Listings");

var mouseover = function(d,i) {

tooltip

.style("opacity", 1)

.style("visibility", "visible")

.html("State: " + d + " Listings: " + i);

}

var mousemove = function(d,i) {

tooltip

.style("top", (d3.event.pageY-10)+"px")

.style("left",(d3.event.pageX+10)+"px");

}

var mouseleave = function(d,i) {

tooltip

.style("opacity", 0)

d3.select(this)

.style("stroke", "none")

.style("visibility", "hidden");

}

// normalize neighborhood names

map.datum(function(d) {

var normalized = d.properties.neighbourhood

.replace(/ /g, '\_')

.replace(/\//g, '\_');

d.properties.neighbourhood = normalized;

d.count = data[1][d.properties.neighbourhood];

return d;

});

// // add the neighborhood name as its class

map

.attr('class', function(d) {

return d.properties.neighbourhood;

})

.attr("fill", function(d) {

return colorScale(d.count); })

.on("mouseover", function(d, i) {return mouseover(d.properties.neighbourhood, d.count); })

.on("mousemove", function(d, i) {return mousemove(d.properties.neighbourhood, d.count); })

.on("mouseleave", function(d, i) {return mouseleave(d.properties.neighbourhood, d.count); })

.attr("transform", "translate(60" + ", 50" + ")");

}