# Version 0.1: Template

// create svg canvas

const canvHeight = 600, canvWidth = 960;

const svg = d3.select("body").append("svg")

.attr("width", canvWidth)

.attr("height", canvHeight)

.style("border", "1px solid");

// calc the width and height depending on margins.

const margin = {top: 50, right: 80, bottom: 50, left: 60};

const width = canvWidth - margin.left - margin.right;

const height = canvHeight - margin.top - margin.bottom;

// create parent group and add left and top margin

const g = svg.append("g")

.attr("id", "chart-area")

.attr("transform", `translate(${margin.left},${margin.top})`);

// chart title

svg.append("text")

.attr("id", "chart-title")

.attr("y", 0)

.attr("x", margin.left)

.attr("dy", "1.5em")

.text("Switzerland");

//-----------------------------------------------------------

function doPlot() {

//adapt from https://bl.ocks.org/mbostock/4207744

}

doPlot();

# Version 0.2: Copy & Adapt

//-----------------------------------------------------------

function doPlot() {

// adapt from https://bl.ocks.org/mbostock/4207744

var projection = d3.geoAlbers() // Albers is best at lat 45°

.rotate([0, 0]) // rotate around globe by lat and long

.center([8.3, 46.8]) // lat and long in degrees

.scale(16000) // zoom into small switzerland, depends on  
 // the projection

.translate([width / 2, height / 2]) // move to center of map

.precision(.1);

// change to d3-fetch

d3.json("./data/readme-swiss.json").then(function(topology) {

// convert TopoJSON topology to GeoJSON

var cantons = topojson.feature(topology,  
 topology.objects.cantons);

//console.log(topology, cantons);

var pathGenerator = d3.geoPath().projection(projection);

g.append("path")

.datum(cantons)

.attr("class", "canton")

.attr("d", pathGenerator);

g.append("path")

// Create a GeoJSON MultiLineString for cantons

.datum(topojson.mesh(topology, topology.objects.cantons))

.attr("class", "canton-boundary")

.attr("d", pathGenerator);

g.selectAll("text")

.data(cantons.features)

.enter().append("text")

.attr("transform",   
 d => "translate(" + pathGenerator.centroid(d) + ")")

.attr("dy", ".35em")

.text(d => d.properties.name);

});

}

doPlot();

# Version 0.3: Introduce cantons list

// change to d3-fetch

d3.json("./data/readme-swiss.json").then(function(topology) {

var cantons = topojson.feature(topology,  
 topology.objects.cantons);

var pathGenerator = d3.geoPath().projection(projection);

//g.append("path")

// .datum(cantons)

// .attr("class", "canton")

// .attr("d", pathGenerator);

var cant = g.selectAll("path.canton")

.data(cantons.features)

.enter()

.append("path")

.attr("id", d=> d.id)

.attr("class", "canton")

.attr("d", pathGenerator);

g.append("path")

.datum(topojson.mesh(topology, topology.objects.cantons))

.attr("class", "canton-boundary")

.attr("d", pathGenerator);

g.selectAll("text")

.data(cantons.features)

.enter().append("text")

.attr("transform",   
 d => "translate(" + pathGenerator.centroid(d) + ")")

.attr("dy", ".35em")

.text(d => d.properties.name);

});

}

# Version 0.4: Mouse handlers

//-----------------------------------------------------------

// Create Event Handlers for mouse

function mouseover(country) {

country.style("fill", "orange");

}

function mouseout(country) {

country.style("fill", "white");

}

function doPlot() {

...

// change to d3-fetch

d3.json("./data/readme-swiss.json").then(function(topology) {

var cantons = topojson.feature(topology,  
 topology.objects.cantons);

var pathGenerator = d3.geoPath().projection(projection);

//g.append("path")

// .datum(cantons)

// .attr("class", "canton")

// .attr("d", pathGenerator);

var cant = g.selectAll("path.canton")

.data(cantons.features)

.enter()

.append("path")

.attr("id", d=> d.id)

.attr("class", "canton")

.attr("d", pathGenerator);

cant.on("mouseover", (event,d) => mouseover(d3.select("#" + d.id)));

cant.on("mouseout", (event,d) => mouseout(d3.select("#" + d.id)));

...

});

}

doPlot();

# Version 0.5: Linked context plot

//-----------------------------------------------------------

// arc generator for donut plot

const arc = d3.arc()

.innerRadius(25)

.outerRadius(45);

const contextHolder = createContextHolder();

// Create Event Handlers for mouse

function mouseover(country) {

//country.style("fill", "orange");

contextHolder.select("path")

.attr("d", arc({startAngle: 0,   
 endAngle: d3.randomUniform(0, 2)() \* Math.PI}));

}

function mouseout(country) {

//country.style("fill", "white");

}

// create small context rectangle

// <g id="context-holder" transform="...">

// <rect width="100" height="100" />

// <path transform="translate(50,50)"></path>

// </g>

function createContextHolder() {

const contextHolder = g.append("g")

.attr("id", "context-holder")

.attr("transform", `translate(${width-80},${height-80})`);

contextHolder.append("rect")

.attr("width", 100)

.attr("height", 100);

contextHolder.append("path")

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

return contextHolder;

}

# Version 1.0: Add external data

//-----------------------------------------------------------

// arc generator for donut plot

const arc = d3.arc()

.innerRadius(25)

.outerRadius(45);

const contextHolder = createContextHolder();

// Create Event Handlers for mouse

function mouseover(citizens, cantonId) {

var percent = citizens[cantonId] / citizens["Schweiz"];

contextHolder.select("path")

.attr("d", arc({startAngle: 0,   
 endAngle: percent \* 2 \* Math.PI}));

d3.select("#context-label")

.text("" + Math.round(percent \*100, 0)+ "%");

//country.style("fill", "orange");

}

function mouseout(citizen, cantonId) {

contextHolder.select("path")

.attr("d", arc({ startAngle: 0, endAngle: 0}));

d3.select("#context-label").text("");

//country.style("fill", "white");

}

// create small context rectangle

// <g id="context-holder" transform="...">

// <rect width="100" height="100" />

// <path transform="translate(50,50)"></path>

// </g>

function createContextHolder() {

const contextHolder = g.append("g")

.attr("id", "context-holder")

.attr("transform", `translate(${width-80},${height-80})`);

contextHolder.append("rect")

.attr("width", 100)

.attr("height", 100);

contextHolder.append("path")

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

contextHolder.append("text")

.attr("id", "context-label")

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

return contextHolder;

}

function doPlot() {

// adapt from https://bl.ocks.org/mbostock/4207744

var projection = d3.geoAlbers() // Albers is best at lat 45°

.rotate([0, 0]) // rotate around globe by lat and long

.center([8.3, 46.8]) // lat and long in degrees

.scale(16000) // zoom into small switzerland

.translate([width / 2, height / 2]) // move to center of map

.precision(.1);

Promise.all([

d3.json("./data/readme-swiss.json"),

d3.csv("./data/swiss-cantons.csv")

]).then(function(data) {

var topology = data[0];

var citizens = data[1];

var cantons = topojson.feature(topology,   
 topology.objects.cantons);

// console.log(topology);

var pathGenerator = d3.geoPath().projection(projection);

var cant = g.selectAll("path.canton")

.data(cantons.features)

.enter()

.append("path")

.attr("id", d=> d.id)

.attr("class", "canton")

.attr("d", pathGenerator);

cant.on("mouseover", d => mouseover(citizens[0], d.id));

cant.on("mouseout", d => mouseout(citizens.columns[0], d.id));

g.append("path")

.datum(topojson.mesh(topology, topology.objects.cantons))

.attr("class", "canton-boundary")

.attr("d", pathGenerator);

g.selectAll("text.canton-label")

.data(cantons.features)

.enter().append("text")

.attr("class", "canton-label")

.attr("transform",

d => "translate(" + pathGenerator.centroid(d) + ")")

.attr("dy", ".35em")

.text(d => d.properties.name);

});

}