Purpose of the program:

The purpose of the program is to provide COVID-19 statistics from Europe, and the purpose of the library that the map is based on is to make it easy to make interactive maps.

Functionality:

With the library it is possible to make an interactive map with Python as shown in the first part of the video. In the second part, we see that an infobox can be shown when hovering over countries on the map to show COVID-19 data.

Inputs and outputs:

The input of the library portion is the Python code used to create the map as well as data and a shapefile. The output of the library is a Javascript file containing a variable with all the map information that can be displayed in a web browser. The input of the interactive map is which countries the user hovers over. The output is the corresponding infobox with COVID-19 data.

List being defined and having values added to it:

var textPopUpList = [];

var textPopUpText;

//Sort text containers into lines

for(var i in infobox\_text){

while(textPopUpList.length <= infobox\_text[i]["line"]){

textPopUpList.push([]);

}

textPopUpText = {};

textPopUpText["text"] = infobox\_text[i]["text"];

textPopUpText["type"] = infobox\_text[i]["type"];

textPopUpText["style"] = infobox\_styles\_text[i];

textPopUpList[infobox\_text[i]["line"]].push(textPopUpText);

List being used:

for(i in textPopUpList){

lineWidth = 0;

maxHeight = 0;

iReverse = numberLines - 1 - i;

for(var j in textPopUpList[iReverse]){

textPopUp = document.createElementNS("http://www.w3.org/2000/svg","text");

if(textPopUpList[iReverse][j]["type"] == "text"){

textPopUp.innerHTML = textPopUpList[iReverse][j]["text"].replace(" ","&nbsp;");

} else {

textPopUp.innerHTML = eval(textPopUpList[iReverse][j]["text"]);

}

//Calculate x and y of text

textX = event.pageX - 72 + lineWidth;

textY = event.pageY - 65 - ttlHeight;

//Style

textColorDict = textPopUpList[iReverse][j]["style"];

The list name is textPopUpList.

Each list item represents a line in the infobox. Because each line may contain multiple text containers, the each list item is itself a list of the text containers in a given line. (The reason for this is allowing both data and text to be in the same line as well as changing colors or fonts.)

The list manages complexity by sorting all the text containers by line. Without it calculating the location of each text container would be almost impossible because even though the height would be able to be calculated, the width would be quite hard since it would not be next to the previous text container.

Procedure:

function showInfobox(c){

//Infobox

var infobox\_styles = data["infobox"]["style"];

var infobox\_styles\_text = infobox\_styles["text"];

var infobox\_text = data["infobox"]["text"];

//Text

var textPopUpParent = document.getElementById("dataInfo");

var textPopUpList = [];

var textPopUpText;

//Sort text containers into lines

for(var i in infobox\_text){

while(textPopUpList.length <= infobox\_text[i]["line"]){

textPopUpList.push([]);

}

textPopUpText = {};

textPopUpText["text"] = infobox\_text[i]["text"];

textPopUpText["type"] = infobox\_text[i]["type"];

textPopUpText["style"] = infobox\_styles\_text[i];

textPopUpList[infobox\_text[i]["line"]].push(textPopUpText);

}

//Create text

var textX;

var textY;

var numberLines = textPopUpList.length;

var lineWidth;

var ttlHeight = 0;

var maxWidth = 0;

var maxHeight;

var bbox;

var textColorHTML;

var textColorDict;

var font\_size;

var font;

var iReverse;

var textPopUpGroup = document.getElementById("dataInfo");

var textPopUpDelete = document.getElementsByClassName("infoboxText")

//Delete old text

while(textPopUpDelete.length > 0){

textPopUpDelete[0].parentNode.removeChild(textPopUpDelete[0]);

}

for(i in textPopUpList){

lineWidth = 0;

maxHeight = 0;

iReverse = numberLines - 1 - i;

for(var j in textPopUpList[iReverse]){

textPopUp = document.createElementNS("http://www.w3.org/2000/svg","text");

if(textPopUpList[iReverse][j]["type"] == "text"){

textPopUp.innerHTML = textPopUpList[iReverse][j]["text"].replace(" ","&nbsp;");

} else {

textPopUp.innerHTML = eval(textPopUpList[iReverse][j]["text"]);

}

//Calculate x and y of text

textX = event.pageX - 72 + lineWidth;

textY = event.pageY - 65 - ttlHeight;

//Style

textColorDict = textPopUpList[iReverse][j]["style"];

textColorHTML = "rgb(" + textColorDict["r"] + "," + textColorDict["g"] + "," + textColorDict["b"] + ");"

font\_size = textColorDict["size"];

font\_family = textColorDict["font"];

//Set property and style

textPopUp.setAttributeNS(null,"x",textX);

textPopUp.setAttributeNS(null,"y",textY);

textPopUp.setAttributeNS(null,"font-family",font\_family);

textPopUp.setAttributeNS(null,"font-size",font\_size);

textPopUp.setAttributeNS(null,"fill",textColorHTML);

textPopUp.setAttributeNS(null,"class","infoboxText");

textPopUpGroup.appendChild(textPopUp);

//Adjust future elements by accounting for element width

bbox = textPopUp.getBBox();

lineWidth = lineWidth + bbox.width;

if(bbox.height > maxHeight){

maxHeight = bbox.height;

}

}

if(lineWidth > maxWidth){

maxWidth = lineWidth;

}

ttlHeight = ttlHeight + maxHeight;

}

//Box

var boxPopUp = document.getElementById("dataInfoBox");

boxPopUp.setAttribute("x",event.pageX-82);

boxPopUp.setAttribute("y",event.pageY-65-ttlHeight);

boxPopUp.setAttribute("height",ttlHeight+10);

boxPopUp.setAttribute("width",maxWidth+20);

//Set stroke width

boxPopUp["style"]["stroke-width"] = infobox\_styles["border"]["stroke-width"];

//Fill colors

var red = infobox\_styles["color"]["r"];

var green = infobox\_styles["color"]["g"];

var blue = infobox\_styles["color"]["b"];

var color = "rgb(" + red + "," + green + "," + blue + ")";

boxPopUp.setAttribute("fill",color);

//Border colors

red = infobox\_styles["border"]["color"]["r"];

green = infobox\_styles["border"]["color"]["g"];

blue = infobox\_styles["border"]["color"]["b"];

color = "rgb(" + red + "," + green + "," + blue + ")";

boxPopUp.style.stroke = color;

//Make visible

boxPopUp.style.visibility = "visible";

}

Procedure being called:

mySVGMap['inputs']['onHover'] = 'showInfobox([COUNTRY]);'

This Python code automatically calls the procedure for all countries.

The procedure draws an infobox in SVG on the map.

First, it iterates through all the text containers to sort them into lines. Then, all the text previously in the infobox (if any) is deleted. Next, each line of the infobox is iterated through. In each line, each text container is iterated through. The location of the text is determined based on the height of the line as well as the location of the previous text container. The font, font size, and color are set. The widest line as well as the height of all the lines are noted for use when drawing the box around the text, which happens after the lines are looped through. The box is then drawn as the color is set.

The procedure is called when Iceland is hovered over. The parameter is a number corresponding to the index of Iceland in the territory list. Both calls are set automatically in the Python portion of the code.

The procedure is called when Norway is hovered over. The parameter is a number corresponding to the index of Norway in the territory list. Both calls are set automatically in the Python portion of the code.

The first call tests the condition that the data retrieved must be for Iceland.

The second call tests the condition that the data retrieved must be for Norway.

The result of the first call is an infobox showing COVID-19 data from Iceland.

The result of the second call is an infobox showing COVID-19 data from Norway.