Map of the USA

Basics of data loading, visualization and interaction. We will visualize dataset for 50 countries of USA on a map.

Adding the map to the sketch

Find *map.png* in file explorer and Drag and Drop it to the Processing widow – i.e., your sketch. The map file will be automatically linked with your sketch. Alternatively, you can add files through *Menu -> Sketch -> Add file.*

Drawing the map

Class *PImage* is a container for images. Create the image object in the *setup()* function. Use *loadImage()* function to which you pass the file name as the attribute. Then draw it in the *draw()* at the position (0,0), so the image will fill the whole window.

```
PImage mapImage;
void setup( ) {
size(640, 400);
mapImage = loadImage("map.png");
}
void draw( ) {
background(255);
image(mapImage, 0, 0);
}
```

Specification of the points on map

In the file *locations.tsv* contains points specifying the center of each state. We will need to load the data into our sketch. For this we can use the code from the file *Table.pde*. The class *Table* loads the data file as a grid of rows and columns. For access to the loaded float numbers we can use the following funcion:

```
table.getFloat(row, column)
```

The rows and columns are numbered from 0, so the column headers (if present) will have attribute *row* equal to 0.

We will create object *locationTable* and for reading the positions of the states we will use function *locationTable.getFloat().* Then we will draw circles (ellipse with the equal width and height) at these locations. Updated code:

```
PImage mapImage;
Table locationTable;
int rowCount;

void setup( ) {
      size(640, 400);
      mapImage = loadImage("map.png");
      // Make a data table from a file that contains
```

```
      // the coordinates of each state.
      locationTable = new Table("locations.tsv");
      // The row count will be used a lot, so store it globally.
      rowCount = locationTable.getRowCount();
}
void draw( ) {
      background(255);
      image(mapImage, 0, 0);
      // Drawing attributes for the ellipses. They affect
      //all subsequent drawings.
      smooth();
      fill(192, 0, 0);
      noStroke();
      // Loop through the rows of the locations file and draw the points.
      for (int row = 0; row < rowCount; row++) {
      float x = locationTable.getFloat(row, 1); // column 1
      float y = locationTable.getFloat(row, 2); // column 2
      ellipse(x, y, 9, 9);
      }
}
```

Data visualization

Now we have the circles, but we would like to map some date onto their visual variables – color and size. We will first try to modify their size based on values in our dataset.

We will add another *Table* object *dataTable* and load data from the file *random.tsv*. In order to be able to map the values onto the ellipse size, we first need to identify the range of the data – minimum and maximum values.

We will firs initialize the variables *dataMin* and *dataMax* with the inbuilt constants *MIN_FLOAT* a *MAX_FLOAT* to ensure that these values will be overwritten with the first larger/smaller value from the dataset. We can do the minimum and maxim search in the *setup()* function:

```
Table dataTable;
float dataMin = MAX_FLOAT;
float dataMax = MIN_FLOAT;

// Read the data table.
dataTable = new Table("random.tsv");
// Find the minimum and maximum values.
for (int row = 0; row < rowCount; row++) {
      float value = dataTable.getFloat(row, 1);
      if (value > dataMax) {
            dataMax = value;
      }
      if (value < dataMin) {
            dataMin = value;
      }
}
```

In the end we will want to visualize the loaded datapoints for each state. We will create a function *drawData()* which will take the x and y coordinates of the state and the state abbreviation as its

parameters. The state abbreviation is located in the first column, so it is considered to be a row header or a row name and can be accessed with function *getRowName()* which takes row index as the argument. We can also pass the row name name as an argument to a *getFloat()* function instead of row index.

To map the loaded values to the ellipse size within some reasonable range we can use the map() function with the following arguments: value to be re-mapped, original data range minimum, original data range maximum, target range minimum, target range maximum.

```
void draw( ) {
      background(255);
      image(mapImage, 0, 0);
      smooth( );
      fill(192, 0, 0);
      noStroke( );
      for (int row = 0; row < rowCount; row++) {
            // getRowName is equivalent to getString(row, 0)
            String abbrev = dataTable.getRowName(row);
            float x = locationTable.getFloat(abbrev, 1);
            float y = locationTable.getFloat(abbrev, 2);
            drawData(x, y, abbrev);
      }
}

// Map the size of the ellipse to the data value
void drawData(float x, float y, String abbrev) {
      // Get data value for state
      float value = dataTable.getFloat(abbrev, 1);
      // Re-map the value to a number between 2 and 40
      float mapped = map(value, dataMin, dataMax, 2, 40);
      // Draw an ellipse for this item
      ellipse(x, y, mapped, mapped);
}
```

Exercise

Try to modify the code so the values will be mapped on color instead of the size of the points.