data-vis\pay-gap-1997-2017.js

```
function PayGapTimeSeries() {

  // Name for the visualisation to appear in the menu bar.
  this.name = 'Gender Pay GAP';

  // Each visualisation must have a unique ID with no special characters.
  this.id = 'pay-gap-timeseries';

  // Title to display above the plot.
  this.title = 'Gender Pay GAP chart!';

  // Names for each axis.
  this.xAxisLabel = "Year";
  this.y1AxisLabel = "Percentage of GAP";
  this.y2AxisLabel = "Median per Gender"

  // Define margin
  var marginSize = 140;

  // Layout object to store all common plot layout parameters and methods.
  this.layout = {
    marginSize: marginSize,

    // Locations of margin positions. Left and bottom have double margin size due to axis and tick labels.
    leftMargin: marginSize * 3.2,
    rightMargin: width - marginSize,
    topMargin: marginSize * 2,
    bottomMargin: height - marginSize * 0.8,
    pad: 5,

    plotWidth: function() {
      return this.rightMargin - this.leftMargin;
    },

    plotHeight: function() {
      return this.bottomMargin - this.topMargin;
    },

    // Boolean to enable/disable background grid.
    grid: false,

    // Number of axis tick labels to draw so that they are not drawn on top of one another.
    numXTickLabels: 10,
    numYTickLabels: 8,
  };

  // Property to represent whether data has been loaded.
  this.loaded = false;

  // Preload the data. This function is called automatically by the gallery when a
visualisation is added.
  this.preload = function() {
    var self = this;
    this.data = loadTable(
      './data/pay-gap/all-employees-hourly-pay-by-gender-1997-2017.csv', 'csv', 'header',
      // Callback function to set the value
      // this.loaded to true.
      function(table) {
        self.loaded = true;
      });

  };

  this.setup = function() {
    // Font defaults.
    textSize(16);

    // Set min and max years: assumes data is sorted by date.
    this.startYear = this.data.getNum(0, 'year');
    this.endYear = this.data.getNum(this.data.getRowCount() - 1, 'year');

    // Find min and max pay gap for mapping to canvas height.
    this.minPayGap = 0;         // Pay equality (zero pay gap).
    this.maxPayGap = max(this.data.getColumn('pay_gap'));

    // Find min and max average per gender
    this.minMedian = 0;
    this.maxMedianMale = max(this.data.getColumn('median_male'));
    this.maxMedianFemale = max(this.data.getColumn('median_female'));
    this.maxMedian = max(this.maxMedianMale, this.maxMedianFemale);

    // Create variable do draw bars and line smoothly
    this.animationProgress = 0;
  };

  this.destroy = function() {
  };

  this.draw = function() {
    if (!this.loaded) {
      console.log('Data not yet loaded');
      return;
    }

    // Draw text
    this.drawText();

    // Draw all y-axis labels.
    drawYAxisTickLabels(this.minPayGap,
                        this.maxPayGap,
                        this.layout,
                        this.mapPayGapToHeight.bind(this),
                        0);

    // Draw all y-axis labels.
    drawY2AxisTickLabels(this.minMedian,
                         this.maxMedian,
                         this.layout,
                         this.mapMedianToHeight.bind(this),
                         0);

    // Draw x and y axis.
    drawAxis(this.layout, 0, true);

    // Draw x and y axis labels.
    drawAxisLabels(this.xAxisLabel,
                   this.y1AxisLabel,
                   this.y2AxisLabel,
                   this.layout);

    // Draw legend
    this.drawLegend(1100, 740, [
    {text: "Woman", color: '#C8102E'},
    {text: "Men", color: '#002147'}
    ]);

    // Plot all pay gaps between startYear and endYear using the width of the canvas minus margins.
    var previous;
    var numYears = this.endYear - this.startYear;

    // Loop over all rows and draw a line from the previous value to the current.
    for (var i = 0; i < this.data.getRowCount(); i++) {

      // Create an object to store data for the current year.
      var current = {
        // Convert strings to numbers.
        'year': this.data.getNum(i, 'year'),
        'payGap': this.data.getNum(i, 'pay_gap'),

//-------------------------------- START NEW CODE -----------------------------------------//
        'male': this.data.getNum(i, 'median_male'), // First attribute is row, second is column. So: row i, column median_male.
        'female': this.data.getNum(i, 'median_female')
//-------------------------------- END NEW CODE -----------------------------------------//
      };

      if (previous != null) {

//-------------------------------- START NEW CODE -----------------------------------------//
        if (current.year != 2017) { // NOT DRAWING YEAR 2017 FOR BETTER VISUALIZATION!
          let barWidth = 25;

          // Draw animation progress for bar
          this.animationProgress += 0.0010;
          this.animationProgress = min(this.animationProgress, 1);

          // Draw bars male
          fill('#002147');
          noStroke();
          rect(this.mapYearToWidth(current.year) - barWidth/2,
               this.layout.bottomMargin,
               barWidth,
               -(this.layout.bottomMargin - this.mapMedianToHeight(current.male)) * this.animationProgress)

          // Draw bars female
          fill('#C8102E');
          noStroke();
          rect(this.mapYearToWidth(current.year) - barWidth/2,
              this.layout.bottomMargin,
              barWidth,
              -(this.layout.bottomMargin - this.mapMedianToHeight(current.female)) * this.animationProgress);

          // Draw line segment connecting previous year to current
          // year pay gap.
          stroke(0);
          strokeWeight(5);
          line(this.mapYearToWidth(previous.year),
              this.mapPayGapToHeight(previous.payGap),
              this.mapYearToWidth(current.year),
              this.mapPayGapToHeight(current.payGap));
        }
//------------------------------- END NEW CODE ----------------------------------------//

        // The number of x-axis labels to skip so that only
        // numXTickLabels are drawn.
        var xLabelSkip = ceil(numYears / this.layout.numXTickLabels);

        // Draw the tick label marking the start of the previous year.
        if (i % xLabelSkip == 0) {
          drawXAxisTickLabel(previous.year, this.layout,
          this.mapYearToWidth.bind(this));
        }
      }

//------------------------------- START NEW CODE ----------------------------------------//
      // Assign current year to previous year so that it is available during the next iteration of this loop to give us the start position of the next line segment.
      previous = {
      year: current.year,
      payGap: current.payGap,
      male: current.male,
      female: current.female
      };
//------------------------------- END NEW CODE ----------------------------------------//

```

```
}
  };

  // REMOVED FUNCTION!!!
  this.drawTitle = function() {
    fill(0);
    noStroke();
    textAlign('center', 'center');
    textFont(robotoFont);
    textSize(34);

    text(this.title,
         (this.layout.plotWidth() / 2) + this.layout.leftMargin,
         this.layout.topMargin - (this.layout.marginSize / 2.6));
  };

  this.mapYearToWidth = function(value) {
    return map(value,
                this.startYear,
                this.endYear,
                this.layout.leftMargin,   // Draw left-to-right from margin.
                this.layout.rightMargin);
  };

  this.mapPayGapToHeight = function(value) {
    return map(value,
                this.minPayGap,
                this.maxPayGap,
                this.layout.bottomMargin, // Smaller pay gap at bottom.
                this.layout.topMargin);   // Bigger pay gap at top.
  };

//------------------------------ START NEW CODE ----------------------------------------
-//
  this.mapMedianToHeight = function(value) {
    return map(value,
                this.minMedian,
                this.maxMedian,
                this.layout.bottomMargin,
                this.layout.topMargin);
  };

  this.drawText = function() {
    // Draw user message
    let message_pay_gap = "Check out the "

    push();
    textSize(30);
    textAlign(LEFT, TOP);
    textFont(robotoFont);
    fill(0);
    noStroke();
    text(message_pay_gap, 440, 125.5);
    textFont(robotoFontBold);
```

```
      text(this.title, 612, 125.5)
      pop();
      }

    this.drawLegend = function(xPos, yPos, labels) {
    // How to use:
    //   draw5Legend(100, 50, [
    //   { text: "Always", color: '#C8102E' },
    //   { text: "Often", color: '#002147' },
    //   { text: "Sometimes", color: '#FFCD6E' },
    //   { text: "Rarely", color: '#333333' },
    //   { text: "Never", color: '#91A7D2' }
    // ]);


    // Space per block
    let spacing = 145;
    let boxSize = 15;

    // Font
    textFont(robotoFont);
    textAlign(LEFT, CENTER);
    textSize(16);
    noStroke();

    for (let i = 0; i < labels.length; i++) {
      //
      let currentX = xPos + i * spacing;

      //
      fill(labels[i].color);
      rect(currentX, yPos, boxSize, boxSize, 3);

      //
      fill(0);
      text(labels[i].text, currentX + boxSize + 10, yPos + boxSize / 2);
    }
    }
  //------------------------------- END NEW CODE -----------------------------------------//
  }

```