data-vis\climate-change.js

```javascript
function ClimateChange() {

  // Name for the visualisation to appear in the menu bar.
  this.name = 'Climate Change';

  // Each visualisation must have a unique ID with no special
  // characters.
  this.id = 'climate-change';

  // Names for each axis.
  this.xAxisLabel = 'year';
  this.y1AxisLabel = '℃';
  this.y2AxisLabel = "";

  var marginSize = 35;

  // 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 * 12,
    rightMargin: width - marginSize - 100,
    topMargin: marginSize * 7,
    bottomMargin: height - marginSize * 4,
    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: 8,
    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/surface-temperature/surface-temperature.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);
    textAlign('center', 'center');

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

    // Find min and max temperature for mapping to canvas height.
    this.minTemperature = min(this.data.getColumn('temperature'));
    this.maxTemperature = max(this.data.getColumn('temperature'));

    // Find mean temperature to plot average marker.
    this.meanTemperature = mean(this.data.getColumn('temperature'));

    // Count the number of frames drawn since the visualisation
    // started so that we can animate the plot.
    this.frameCount = 0;

    // Create sliders to control start and end years. Default to
    // visualise full range.
    this.startSlider = createSlider(this.minYear,
                                    this.maxYear - 1,
                                    this.minYear,
                                    1);
    this.startSlider.position(400, 10);

    this.endSlider = createSlider(this.minYear + 1,
                                  this.maxYear,
                                  this.maxYear,
                                  1);
    this.endSlider.position(600, 10);
  };

  this.destroy = function() {
    this.startSlider.remove();
    this.endSlider.remove();
  };

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

    // Prevent slider ranges overlapping.
    if (this.startSlider.value() >= this.endSlider.value()) {
      this.startSlider.value(this.endSlider.value() - 1);
    }
    this.startYear = this.startSlider.value();
    this.endYear = this.endSlider.value();

    // Draw all y-axis tick labels.
    drawYAxisTickLabels(this.minTemperature,
                        this.maxTemperature,
                        this.layout,
                        this.mapTemperatureToHeight.bind(this),
                        1);

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

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

    // Plot average line.
    stroke(200);
    strokeWeight(1);
    line(this.layout.leftMargin,
         this.mapTemperatureToHeight(this.meanTemperature),
         this.layout.rightMargin,
         this.mapTemperatureToHeight(this.meanTemperature));

    // Plot all temperatures between startYear and endYear using the
    // width of the canvas minus margins.
    var previous;
    var numYears = this.endYear - this.startYear;
    var segmentWidth = this.layout.plotWidth() / numYears;

    // Count the number of years plotted each frame to create
    // animation effect.
    var yearCount = 0;

    // Loop over all rows but only plot those in range.
    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'),
        'temperature': this.data.getNum(i, 'temperature')
      };

      if (previous != null
          && current.year > this.startYear
          && current.year <= this.endYear) {

        // Draw background gradient to represent colour temperature of
        // the current year.
        noStroke();
        fill(this.mapTemperatureToColour(current.temperature));
        rect(this.mapYearToWidth(previous.year),
             this.layout.topMargin,
             segmentWidth,
             this.layout.plotHeight());

        // Draw line segment connecting previous year to current
        // year temperature.
        stroke(0);
        line(this.mapYearToWidth(previous.year),
             this.mapTemperatureToHeight(previous.temperature),
             this.mapYearToWidth(current.year),
             this.mapTemperatureToHeight(current.temperature));

        // 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 (yearCount % xLabelSkip == 0) {
          drawXAxisTickLabel(previous.year, this.layout,
                             this.mapYearToWidth.bind(this));
        }

        // When six or fewer years are displayed also draw the final
        // year x tick label.
        if ((numYears <= 6
             && yearCount == numYears - 1)) {
          drawXAxisTickLabel(current.year, this.layout,
                             this.mapYearToWidth.bind(this));
        }

        yearCount++;
      }

      // Stop drawing this frame when the number of years drawn is
      // equal to the frame count. This creates the animated effect
      // over successive frames.
      if (yearCount >= this.frameCount) {
        break;
      }

      // 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 = current;
    }

    // Count the number of frames since this visualisation
    // started. This is used in creating the animation effect and to
    // stop the main p5 draw loop when all years have been drawn.
    this.frameCount++;

    // Stop animation when all years have been drawn.
    if (this.frameCount >= numYears) {
      //noLoop();
    }
  };

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

  this.mapTemperatureToHeight = function(value) {
    return map(value,
               this.minTemperature,
               this.maxTemperature,
               this.layout.bottomMargin, // Lower temperature at bottom.
               this.layout.topMargin);   // Higher temperature at top.
  };

  this.mapTemperatureToColour = function(value) {
    var red =  map(value,
                   this.minTemperature,
                   this.maxTemperature,
                   0,
                   255);
    var blue = 255 - red;
    return color(red, 0, blue, 100);
  };
}

```