data-vis\nutrient-intakes.js

```
//------------------------------ START NEW CODE ---------------------------------------
-//
function NutrientsTimeSeries() {

  // Name for the visualisation to appear in the menu bar.
  this.name = 'Vitamins Intake TimeSeries';

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

  // Title to display above the plot.
  this.title = 'Vitamins Instake';

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

  this.colors = [];

  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 * 14,
    rightMargin: width - marginSize,
    topMargin: marginSize + 200,
    bottomMargin: height - marginSize * 3,
    pad: 5,

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

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

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

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

```

```
51    // Property to represent whether data has been loaded.
52    this.loaded = false;
53
54    // Preload the data. This function is called automatically by the gallery when a
      visualisation is added.
55    this.preload = function() {
56      var self = this;
57      this.data = loadTable(
58        './data/food/avg-intake-weighted-reference-nutrient-intakes.csv', 'csv', 'header',
59        // Callback function to set the value
60        // this.loaded to true.
61        function(table) {
62          self.loaded = true;
63        });
64
65    };
66
67    this.setup = function() {
68      // Font defaults.
69      textSize(16);
70
71      // Set min and max years: assumes data is sorted by date.
72      this.startYear = Number(this.data.columns[1]);
73      // this.endYear = this.data.getNum(this.data.getRowCount() - 1, 'year');
74      this.endYear = Number(this.data.columns[this.data.columns.length - 1]);
75
76      // Colors
77      for (var i = 0; i < this.data.getRowCount(); i++) {
78        this.colors.push(color(random(0, 255), random(0, 255), random(0, 255)));
79      }
80
81      // Find min and max
82      this.minValue = 0;
83      this.maxValue = 45;
84    };
85
86    this.destroy = function() {
87    };
88
89    this.draw = function() {
90      if (!this.loaded) {
91        console.log('Data not yet loaded');
92        return;
93      }
94
95      // Draw the title above the plot.
96      this.drawTitle();
97
98      // Draw all y-axis labels.
99      drawYAxisTickLabels(this.minValue,
100                          this.maxValue,
101                          this.layout,
102                          this.mapPayGapToHeight.bind(this),
103                          0);
```

```
104
105     // Draw x and y axis.
106     drawAxis(this.layout);
107
108     // Draw x and y axis labels.
109     drawAxisLabels(this.xAxisLabel,
110                    this.y1AxisLabel,
111                    this.y2AxisLabel,
112                    this.layout);
113
114     // Plot all pay gaps between startYear and endYear using the width of the canvas minus
    margins.
115     var previous;
116     var numYears = this.endYear - this.startYear;
117
118     // Loop over all rows and draw a line from the previous value to the current.
119     for (var i = 0; i < this.data.getRowCount(); i++) {
120
121       let row = this.data.getRow(i);
122       let previous = null;
123       let l = row.getString(0);
124
125       for(var j = 1; j < numYears; j++) {
126         var current = {
127           'year': this.startYear + j - 1,
128           'value': row.getNum(j)
129         };
130
131         if (previous != null) {
132           // Draw line segment connecting previous year to current
133           // year pay gap.
134           stroke(this.colors[i]);
135           line(this.mapYearToWidth(previous.year),
136               this.mapPayGapToHeight(previous.value),
137               this.mapYearToWidth(current.year),
138               this.mapPayGapToHeight(current.value));
139
140           // The number of x-axis labels to skip so that only
141           // numXTickLabels are drawn.
142           var xLabelSkip = ceil(numYears / this.layout.numXTickLabels);
143
144           // Draw the tick label marking the start of the previous year.
145           if (i % xLabelSkip == 0) {
146             drawXAxisTickLabel(previous.year, this.layout,
147                                this.mapYearToWidth.bind(this));
148           }
149         }
150         else {
151           noStroke();
152           fill(this.colors[i]);
153           text(l, 630, this.mapPayGapToHeight(current.value))
154         }
155       // 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;
      }
    }
  };

  this.drawTitle = function() {
    fill(0);
    noStroke();
    textAlign('center', 'center');

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

  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.minValue,
               this.maxValue,
               this.layout.bottomMargin, // Smaller pay gap at bottom.
               this.layout.topMargin);   // Bigger pay gap at top.
  };
}
//------------------------------ END NEW CODE ----------------------------------------//
```