modules/CM1010 Introduction to Programming_II/week12/midterm/project/index.html

```html
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <title>Running Data Visualisation</title>

    <!-- Libraries -->
    <script src="lib/p5.min.js"></script>

    <link rel="stylesheet" href="style.css">

    <!-- Main sketch file -->
    <script src="sketch.js"></script>

    <!-- Helper functions and core components (from template) -->
    <script src="helper-functions.js"></script>
    <script src="gallery.js"></script>
    <script src="pie-chart.js"></script>

    <!-- Start of my own code -->
    <!-- New reusable chart constructors -->
    <script src="bar-chart.js"></script>
    <script src="scatter-plot.js"></script>

    <!-- New running data visualisations -->
    <script src="monthly-distance.js"></script>
    <script src="pace-progress.js"></script>
    <script src="activity-types.js"></script>
    <script src="heartrate-vs-pace.js"></script>
    <!-- End of my own code -->
  </head>
  <body>
    <div id="app" class="container">
      <ul id="visuals-menu"></ul>
    </div>
  </body>
</html>

```

modules/CM1010 Introduction to Programming_II/week12/midterm/project/sketch.js

```
1
2  // Global variable to store the gallery object. The gallery object is
3  // a container for all the visualisations.
4  var gallery;
5
6  function setup() {
7    // Create a canvas to fill the content div from index.html.
8    var c = createCanvas(1024, 576);
9    c.parent('app');
10
11   // Create a new gallery object.
12   gallery = new Gallery();
13
14   /* Start of my own code */
15   // Add running data visualisations
16   gallery.addVisual(new MonthlyDistance());
17   gallery.addVisual(new PaceProgress());
18   gallery.addVisual(new ActivityTypes());
19   gallery.addVisual(new HeartRateVsPace());
20   /* End of my own code */
21 }
22
23 function draw() {
24   background(255);
25   if (gallery.selectedVisual != null) {
26     gallery.selectedVisual.draw();
27   }
28 }
29
```

modules/CM1010 Introduction to Programming_II/week12/midterm/project/bar-chart.js

```
/* Start of my own code */
function BarChart(layout) {
  this.layout = layout;
  this.barWidth = 30;
  this.barGap = 10;

  this.draw = function(data, labels, colours, title) {
    if (data.length == 0) {
      console.log('Data has length zero!');
      return;
    }

    if (labels.length != data.length) {
      console.log('Data and labels arrays must be the same length!');
      return;
    }

    var plotWidth = this.layout.rightMargin - this.layout.leftMargin;
    var plotHeight = this.layout.bottomMargin - this.layout.topMargin;
    this.barWidth = (plotWidth - (data.length + 1) * this.barGap) / data.length;
    var maxValue = max(data);

    if (title) {
      noStroke();
      fill(0);
      textAlign('center', 'center');
      textSize(16);
      text(title,
           (this.layout.leftMargin + this.layout.rightMargin) / 2,
           this.layout.topMargin - 20);
    }

    for (var i = 0; i < data.length; i++) {
      var x = this.layout.leftMargin + this.barGap + i * (this.barWidth +
this.barGap);
      var barHeight = map(data[i], 0, maxValue, 0, plotHeight);
      var y = this.layout.bottomMargin - barHeight;

      if (Array.isArray(colours)) {
        fill(colours[i % colours.length]);
      } else {
        fill(colours);
      }

      stroke(0);
      strokeWeight(1);
      rect(x, y, this.barWidth, barHeight);

      noStroke();
      fill(0);
      textAlign('center', 'bottom');
      textSize(10);
```

```
      text(data[i].toFixed(1), x + this.barWidth / 2, y - 5);

      push();
      translate(x + this.barWidth / 2, this.layout.bottomMargin + 10);
      rotate(radians(-45));
      textAlign('right', 'center');
      textSize(10);
      text(labels[i], 0, 0);
      pop();
    }
  };

  this.drawYAxisTickLabels = function(minValue, maxValue, numTicks, decimalPlaces)
{
    var range = maxValue - minValue;
    var tickStep = range / numTicks;

    fill(0);
    noStroke();
    textAlign('right', 'center');
    textSize(12);

    for (var i = 0; i <= numTicks; i++) {
      var value = minValue + (i * tickStep);
      var y = map(value, minValue, maxValue,
                  this.layout.bottomMargin,
                  this.layout.topMargin);

      text(value.toFixed(decimalPlaces),
           this.layout.leftMargin - 10,
           y);

      stroke(220);
      strokeWeight(1);
      line(this.layout.leftMargin, y,
           this.layout.rightMargin, y);
    }
  };
}
/* End of my own code */

```

modules/CM1010 Introduction to Programming_II/week12/midterm/project/scatter-plot.js

```
/* Start of my own code */
function ScatterPlot(layout) {
  this.layout = layout;
  this.pointSize = 8;

  this.draw = function(xData, yData, xRange, yRange, colour, labels) {
    if (xData.length == 0 || yData.length == 0) {
      console.log('Data has length zero!');
      return;
    }

    if (xData.length != yData.length) {
      console.log('X and Y data arrays must be the same length!');
      return;
    }

    for (var i = 0; i < xData.length; i++) {
      var x = this.mapToX(xData[i], xRange.min, xRange.max);
      var y = this.mapToY(yData[i], yRange.min, yRange.max);
      var isHovered = dist(mouseX, mouseY, x, y) < this.pointSize;

      if (isHovered) {
        fill(255, 100, 100);
        noStroke();
        ellipse(x, y, this.pointSize * 1.5, this.pointSize * 1.5);
        this.drawTooltip(x, y, xData[i], yData[i], labels ? labels[i] : null);
      } else {
        fill(colour);
        stroke(0);
        strokeWeight(0.5);
        ellipse(x, y, this.pointSize, this.pointSize);
      }
    }
  };

  this.mapToX = function(value, minVal, maxVal) {
    return map(value, minVal, maxVal,
               this.layout.leftMargin,
               this.layout.rightMargin);
  };

  this.mapToY = function(value, minVal, maxVal) {
    return map(value, minVal, maxVal,
               this.layout.bottomMargin,
               this.layout.topMargin);
  };

  this.drawTooltip = function(x, y, xVal, yVal, label) {
    var tooltipWidth = 120;
    var tooltipHeight = 50;
    var tooltipX = x + 10;
```

```
    var tooltipY = y - tooltipHeight - 10;

    if (tooltipX + tooltipWidth > this.layout.rightMargin) {
      tooltipX = x - tooltipWidth - 10;
    }
    if (tooltipY < this.layout.topMargin) {
      tooltipY = y + 10;
    }

    fill(255, 255, 255, 230);
    stroke(100);
    strokeWeight(1);
    rect(tooltipX, tooltipY, tooltipWidth, tooltipHeight, 5);

    fill(0);
    noStroke();
    textAlign('left', 'top');
    textSize(10);

    var yText = 'Pace: ' + this.formatPace(yVal) + '/km';
    var xText = 'HR: ' + xVal.toFixed(0) + ' bpm';

    text(xText, tooltipX + 5, tooltipY + 5);
    text(yText, tooltipX + 5, tooltipY + 20);

    if (label) {
      text(label, tooltipX + 5, tooltipY + 35);
    }
  };

  this.formatPace = function(paceMinutes) {
    var minutes = Math.floor(paceMinutes);
    var seconds = Math.round((paceMinutes - minutes) * 60);
    if (seconds < 10) {
      return minutes + ':0' + seconds;
    }
    return minutes + ':' + seconds;
  };

  this.drawXAxisTicks = function(minVal, maxVal, numTicks, label) {
    var range = maxVal - minVal;
    var step = range / numTicks;

    fill(0);
    noStroke();
    textAlign('center', 'top');
    textSize(10);

    for (var i = 0; i <= numTicks; i++) {
      var value = minVal + (i * step);
      var x = this.mapToX(value, minVal, maxVal);

      stroke(0);
      line(x, this.layout.bottomMargin, x, this.layout.bottomMargin + 5);
```

```
106
107       noStroke();
108       text(value.toFixed(0), x, this.layout.bottomMargin + 8);
109
110       stroke(220);
111       line(x, this.layout.topMargin, x, this.layout.bottomMargin);
112     }
113
114     textSize(12);
115     text(label,
116          (this.layout.leftMargin + this.layout.rightMargin) / 2,
117          this.layout.bottomMargin + 30);
118   };
119
120   this.drawYAxisTicks = function(minVal, maxVal, numTicks, label) {
121     var range = maxVal - minVal;
122     var step = range / numTicks;
123
124     fill(0);
125     noStroke();
126     textAlign('right', 'center');
127     textSize(10);
128
129     for (var i = 0; i <= numTicks; i++) {
130       var value = minVal + (i * step);
131       var y = this.mapToY(value, minVal, maxVal);
132
133       text(this.formatPace(value), this.layout.leftMargin - 8, y);
134
135       stroke(220);
136       line(this.layout.leftMargin, y, this.layout.rightMargin, y);
137     }
138
139     push();
140     translate(this.layout.leftMargin - 45, height / 2);
141     rotate(-PI / 2);
142     textAlign('center', 'center');
143     textSize(12);
144     text(label, 0, 0);
145     pop();
146   };
147 }
148 /* End of my own code */
149
```

modules/CM1010 Introduction to Programming_II/week12/midterm/project/monthly-distance.js

```
/* Start of my own code */
function MonthlyDistance() {
  this.name = 'Monthly Distance';
  this.id = 'monthly-distance';
  this.loaded = false;

  var marginSize = 35;
  this.layout = {
    marginSize: marginSize,
    leftMargin: marginSize * 2,
    rightMargin: width - marginSize,
    topMargin: marginSize * 2,
    bottomMargin: height - marginSize * 3,
    pad: 5,
    plotWidth: function() {
      return this.rightMargin - this.leftMargin;
    },
    plotHeight: function() {
      return this.bottomMargin - this.topMargin;
    },
    grid: true,
    numYTickLabels: 6
  };

  this.barChart = new BarChart(this.layout);

  this.preload = function() {
    var self = this;
    this.data = loadTable(
      './data/running/activities.csv', 'csv', 'header',
      function(table) {
        self.loaded = true;
      });
  };

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

  this.destroy = function() {};

  this.aggregateByMonth = function() {
    var months = {};

    for (var i = 0; i < this.data.getRowCount(); i++) {
      var dateStr = this.data.getString(i, 'date');
      var distance = this.data.getNum(i, 'distance');
```

```
52        var yearMonth = dateStr.substring(0, 7);
53
54        if (months[yearMonth]) {
55          months[yearMonth] += distance;
56        } else {
57          months[yearMonth] = distance;
58        }
59      }
60
61      var sortedMonths = Object.keys(months).sort();
62      var labels = [];
63      var values = [];
64
65      for (var j = 0; j < sortedMonths.length; j++) {
66        var ym = sortedMonths[j];
67        var parts = ym.split('-');
68        var monthNames = ['Jan', 'Feb', 'Mar', 'Apr', 'May', 'Jun',
69                          'Jul', 'Aug', 'Sep', 'Oct', 'Nov', 'Dec'];
70        var monthLabel = monthNames[parseInt(parts[1]) - 1] + ' ' +
parts[0].substring(2);
71        labels.push(monthLabel);
72        values.push(months[ym]);
73      }
74
75      return { labels: labels, values: values };
76    };
77
78    this.draw = function() {
79      if (!this.loaded) {
80        console.log('Data not yet loaded');
81        return;
82      }
83
84      fill(0);
85      noStroke();
86      textSize(20);
87      textAlign('center', 'center');
88      text('Monthly Running Distance',
89           (this.layout.leftMargin + this.layout.rightMargin) / 2,
90           this.layout.topMargin / 2);
91
92      push();
93      translate(this.layout.leftMargin / 3, height / 2);
94      rotate(-PI / 2);
95      textSize(14);
96      text('Distance (km)', 0, 0);
97      pop();
98
99      stroke(0);
100     strokeWeight(1);
101     line(this.layout.leftMargin, this.layout.topMargin,
102          this.layout.leftMargin, this.layout.bottomMargin);
103     line(this.layout.leftMargin, this.layout.bottomMargin,
104          this.layout.rightMargin, this.layout.bottomMargin);
```

```
105
106     var maxDistance = max(this.monthlyData.values);
107     this.barChart.drawYAxisTickLabels(0, maxDistance, this.layout.numYTickLabels,
    0);
108
109     var barColour = color(66, 133, 244);
110     this.barChart.draw(
111       this.monthlyData.values,
112       this.monthlyData.labels,
113       barColour,
114       null
115     );
116
117     var totalDistance = this.monthlyData.values.reduce(function(a, b) {
118       return a + b;
119     }, 0);
120
121     textSize(12);
122     textAlign('left', 'top');
123     fill(100);
124     text('Total Distance: ' + totalDistance.toFixed(1) + ' km',
125          this.layout.leftMargin,
126          this.layout.bottomMargin + 60);
127   };
128 }
129 /* End of my own code */
130
```

modules/CM1010 Introduction to Programming_II/week12/midterm/project/pace-progress.js

```
/* Start of my own code */
function PaceProgress() {
  this.name = 'Pace Progress';
  this.id = 'pace-progress';
  this.title = 'Running Pace Progress Over Time';
  this.xAxisLabel = 'Date';
  this.yAxisLabel = 'Pace (min/km)';

  var marginSize = 35;
  this.layout = {
    marginSize: marginSize,
    leftMargin: marginSize * 2,
    rightMargin: width - marginSize,
    topMargin: marginSize * 2,
    bottomMargin: height - marginSize * 2,
    pad: 5,
    plotWidth: function() {
      return this.rightMargin - this.leftMargin;
    },
    plotHeight: function() {
      return this.bottomMargin - this.topMargin;
    },
    grid: true,
    numXTickLabels: 8,
    numYTickLabels: 8
  };

  this.loaded = false;

  this.preload = function() {
    var self = this;
    this.data = loadTable(
      './data/running/activities.csv', 'csv', 'header',
      function(table) {
        self.loaded = true;
      });
  };

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

    this.processedData = this.processData();
    this.startDate = this.processedData.dates[0];
    this.endDate = this.processedData.dates[this.processedData.dates.length - 1];
    this.minPace = Math.floor(min(this.processedData.paces)) - 0.5;
    this.maxPace = Math.ceil(max(this.processedData.paces)) + 0.5;
  };

```

```
  this.destroy = function() {};

  this.paceToMinutes = function(paceStr) {
    var parts = paceStr.split(':');
    if (parts.length == 2) {
      return parseInt(parts[0]) + parseInt(parts[1]) / 60;
    }
    return 0;
  };

  this.processData = function() {
    var dates = [];
    var paces = [];
    var distances = [];

    for (var i = 0; i < this.data.getRowCount(); i++) {
      var activityType = this.data.getString(i, 'activity_type');

      if (activityType === 'Run') {
        var dateStr = this.data.getString(i, 'date');
        var paceStr = this.data.getString(i, 'avg_pace');
        var distance = this.data.getNum(i, 'distance');

        if (distance >= 3) {
          var paceMinutes = this.paceToMinutes(paceStr);

          if (paceMinutes >= 4 && paceMinutes <= 10) {
            dates.push(new Date(dateStr));
            paces.push(paceMinutes);
            distances.push(distance);
          }
        }
      }
    }

    var combined = [];
    for (var j = 0; j < dates.length; j++) {
      combined.push({
        date: dates[j],
        pace: paces[j],
        distance: distances[j]
      });
    }
    combined.sort(function(a, b) {
      return a.date - b.date;
    });

    var sortedDates = [];
    var sortedPaces = [];
    var sortedDistances = [];
    for (var k = 0; k < combined.length; k++) {
      sortedDates.push(combined[k].date);
      sortedPaces.push(combined[k].pace);
      sortedDistances.push(combined[k].distance);
```

```
    }

    return {
      dates: sortedDates,
      paces: sortedPaces,
      distances: sortedDistances
    };
  };

  // Moving average smooths out short-term fluctuations
  this.calculateMovingAverage = function(data, windowSize) {
    var result = [];
    for (var i = 0; i < data.length; i++) {
      var start = Math.max(0, i - windowSize + 1);
      var sum = 0;
      var count = 0;
      for (var j = start; j <= i; j++) {
        sum += data[j];
        count++;
      }
      result.push(sum / count);
    }
    return result;
  };

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

    this.drawTitle();

    drawYAxisTickLabels(this.minPace,
                        this.maxPace,
                        this.layout,
                        this.mapPaceToHeight.bind(this),
                        1);

    drawAxis(this.layout);
    drawAxisLabels(this.xAxisLabel, this.yAxisLabel, this.layout);

    var dates = this.processedData.dates;
    var paces = this.processedData.paces;

    for (var i = 0; i < dates.length; i++) {
      var x = this.mapDateToWidth(dates[i]);
      var y = this.mapPaceToHeight(paces[i]);

      fill(100, 149, 237, 150);
      noStroke();
      ellipse(x, y, 6, 6);
    }

```

```
    var movingAvg = this.calculateMovingAverage(paces, 10);
    stroke(220, 20, 60);
    strokeWeight(2);
    noFill();
    beginShape();
    for (var j = 0; j < dates.length; j++) {
      var trendX = this.mapDateToWidth(dates[j]);
      var trendY = this.mapPaceToHeight(movingAvg[j]);
      vertex(trendX, trendY);
    }
    endShape();

    this.drawXAxisLabels();
    this.drawLegend();

    fill(100);
    noStroke();
    textSize(10);
    textAlign('left', 'top');
    text('Note: Lower pace = faster running',
         this.layout.leftMargin,
         this.layout.bottomMargin + 45);
  };

  this.drawTitle = function() {
    fill(0);
    noStroke();
    textAlign('center', 'center');
    textSize(18);
    text(this.title,
         (this.layout.plotWidth() / 2) + this.layout.leftMargin,
         this.layout.topMargin / 2);
  };

  this.drawXAxisLabels = function() {
    var dates = this.processedData.dates;
    var numLabels = Math.min(this.layout.numXTickLabels, dates.length);
    var step = Math.floor(dates.length / numLabels);

    fill(0);
    noStroke();
    textSize(10);
    textAlign('center', 'top');

    for (var i = 0; i < dates.length; i += step) {
      var x = this.mapDateToWidth(dates[i]);
      var dateLabel = this.formatDate(dates[i]);

      stroke(0);
      line(x, this.layout.bottomMargin, x, this.layout.bottomMargin + 5);

      noStroke();
      text(dateLabel, x, this.layout.bottomMargin + 8);
    }
```

```
  };

  this.formatDate = function(date) {
    var months = ['Jan', 'Feb', 'Mar', 'Apr', 'May', 'Jun',
                  'Jul', 'Aug', 'Sep', 'Oct', 'Nov', 'Dec'];
    return months[date.getMonth()] + ' ' + (date.getFullYear() % 100);
  };

  this.drawLegend = function() {
    var legendX = this.layout.rightMargin - 150;
    var legendY = this.layout.topMargin + 10;

    fill(100, 149, 237);
    noStroke();
    ellipse(legendX, legendY, 6, 6);
    fill(0);
    textAlign('left', 'center');
    textSize(11);
    text('Individual runs', legendX + 10, legendY);

    stroke(220, 20, 60);
    strokeWeight(2);
    line(legendX - 10, legendY + 20, legendX + 10, legendY + 20);
    noStroke();
    fill(0);
    text('10-run moving avg', legendX + 15, legendY + 20);
  };

  this.mapDateToWidth = function(date) {
    var startTime = this.startDate.getTime();
    var endTime = this.endDate.getTime();
    return map(date.getTime(),
               startTime,
               endTime,
               this.layout.leftMargin,
               this.layout.rightMargin);
  };

  this.mapPaceToHeight = function(value) {
    return map(value,
               this.minPace,
               this.maxPace,
               this.layout.topMargin,
               this.layout.bottomMargin);
  };
}
/* End of my own code */

```

modules/CM1010 Introduction to Programming_II/week12/midterm/project/activity-types.js

```
/* Start of my own code */
function ActivityTypes() {
  this.name = 'Activity Types';
  this.id = 'activity-types';
  this.loaded = false;

  this.pie = new PieChart(width / 2, height / 2, width * 0.35);

  this.preload = function() {
    var self = this;
    this.data = loadTable(
      './data/running/activities.csv', 'csv', 'header',
      function(table) {
        self.loaded = true;
      });
  };

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

    this.activityCounts = this.countActivityTypes();
    this.distanceByType = this.sumDistanceByType();

    this.select = createSelect();
    this.select.position(350, 40);
    this.select.option('By Count');
    this.select.option('By Distance');
  };

  this.destroy = function() {
    this.select.remove();
  };

  this.countActivityTypes = function() {
    var counts = {};

    for (var i = 0; i < this.data.getRowCount(); i++) {
      var activityType = this.data.getString(i, 'activity_type');

      if (counts[activityType]) {
        counts[activityType]++;
      } else {
        counts[activityType] = 1;
      }
    }

    return counts;
  };
```

```
52
53    this.sumDistanceByType = function() {
54      var distances = {};
55
56      for (var i = 0; i < this.data.getRowCount(); i++) {
57        var activityType = this.data.getString(i, 'activity_type');
58        var distance = this.data.getNum(i, 'distance');
59
60        if (distances[activityType]) {
61          distances[activityType] += distance;
62        } else {
63          distances[activityType] = distance;
64        }
65      }
66
67      return distances;
68    };
69
70    this.draw = function() {
71      if (!this.loaded) {
72        console.log('Data not yet loaded');
73        return;
74      }
75
76      var viewMode = this.select.value();
77      var dataToShow;
78      var title;
79
80      if (viewMode === 'By Count') {
81        dataToShow = this.activityCounts;
82        title = 'Activity Types by Count';
83      } else {
84        dataToShow = this.distanceByType;
85        title = 'Activity Types by Distance (km)';
86      }
87
88      var labels = Object.keys(dataToShow);
89      var values = [];
90      for (var i = 0; i < labels.length; i++) {
91        values.push(dataToShow[labels[i]]);
92      }
93
94      var colours = [
95        color(66, 133, 244),
96        color(52, 168, 83),
97        color(251, 188, 5)
98      ];
99
100     this.pie.draw(values, labels, colours, title);
101     this.drawStats(dataToShow, viewMode);
102   };
103
104   this.drawStats = function(data, viewMode) {
105     var labels = Object.keys(data);
```

```
    var total = 0;
    for (var i = 0; i < labels.length; i++) {
      total += data[labels[i]];
    }

    var statsX = 80;
    var statsY = height - 100;

    fill(0);
    noStroke();
    textAlign('left', 'top');
    textSize(12);

    if (viewMode === 'By Count') {
      text('Total Activities: ' + total, statsX, statsY);
    } else {
      text('Total Distance: ' + total.toFixed(1) + ' km', statsX, statsY);
    }

    textSize(11);
    for (var j = 0; j < labels.length; j++) {
      var percentage = ((data[labels[j]] / total) * 100).toFixed(1);
      var valueStr = viewMode === 'By Count'
        ? data[labels[j]] + ' activities'
        : data[labels[j]].toFixed(1) + ' km';
      text(labels[j] + ': ' + valueStr + ' (' + percentage + '%)',
           statsX, statsY + 18 + (j * 16));
    }
  };
}
/* End of my own code */

```

modules/CM1010 Introduction to Programming_II/week12/midterm/project/heartrate-vs-pace.js

```
/* Start of my own code */
function HeartRateVsPace() {
  this.name = 'Heart Rate vs Pace';
  this.id = 'heartrate-vs-pace';
  this.title = 'Heart Rate vs Running Pace';

  var marginSize = 35;
  this.layout = {
    marginSize: marginSize,
    leftMargin: marginSize * 2.5,
    rightMargin: width - marginSize,
    topMargin: marginSize * 2,
    bottomMargin: height - marginSize * 2.5,
    pad: 5,
    plotWidth: function() {
      return this.rightMargin - this.leftMargin;
    },
    plotHeight: function() {
      return this.bottomMargin - this.topMargin;
    },
    grid: true,
    numXTickLabels: 6,
    numYTickLabels: 6
  };

  this.scatterPlot = new ScatterPlot(this.layout);
  this.loaded = false;

  this.preload = function() {
    var self = this;
    this.data = loadTable(
      './data/running/activities.csv', 'csv', 'header',
      function(table) {
        self.loaded = true;
      });
  };

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

    this.processedData = this.processData();

    this.xRange = {
      min: Math.floor(min(this.processedData.heartRates) / 10) * 10 - 5,
      max: Math.ceil(max(this.processedData.heartRates) / 10) * 10 + 5
    };

    this.yRange = {
```

```
      min: Math.floor(min(this.processedData.paces) * 2) / 2 - 0.5,
      max: Math.ceil(max(this.processedData.paces) * 2) / 2 + 0.5
    };
  };

  this.destroy = function() {};

  this.paceToMinutes = function(paceStr) {
    var parts = paceStr.split(':');
    if (parts.length == 2) {
      return parseInt(parts[0]) + parseInt(parts[1]) / 60;
    }
    return 0;
  };

  this.processData = function() {
    var heartRates = [];
    var paces = [];
    var labels = [];

    for (var i = 0; i < this.data.getRowCount(); i++) {
      var activityType = this.data.getString(i, 'activity_type');

      if (activityType === 'Run') {
        var heartRate = this.data.getNum(i, 'avg_heart_rate');
        var paceStr = this.data.getString(i, 'avg_pace');
        var distance = this.data.getNum(i, 'distance');
        var date = this.data.getString(i, 'date');

        if (heartRate > 0 && distance >= 3) {
          var paceMinutes = this.paceToMinutes(paceStr);

          if (paceMinutes >= 5 && paceMinutes <= 9) {
            heartRates.push(heartRate);
            paces.push(paceMinutes);
            labels.push(date + ' - ' + distance.toFixed(1) + 'km');
          }
        }
      }
    }

    return {
      heartRates: heartRates,
      paces: paces,
      labels: labels
    };
  };

  // Pearson correlation coefficient: measures linear relationship between two variables
  this.calculateCorrelation = function(x, y) {
    var n = x.length;
    var sumX = 0, sumY = 0, sumXY = 0, sumX2 = 0, sumY2 = 0;

```

```
    for (var i = 0; i < n; i++) {
      sumX += x[i];
      sumY += y[i];
      sumXY += x[i] * y[i];
      sumX2 += x[i] * x[i];
      sumY2 += y[i] * y[i];
    }

    var numerator = (n * sumXY) - (sumX * sumY);
    var denominator = Math.sqrt(((n * sumX2) - (sumX * sumX)) *
                                ((n * sumY2) - (sumY * sumY)));

    if (denominator === 0) return 0;
    return numerator / denominator;
  };

  // Least squares linear regression: finds best fit line y = mx + b
  this.calculateRegression = function(x, y) {
    var n = x.length;
    var sumX = 0, sumY = 0, sumXY = 0, sumX2 = 0;

    for (var i = 0; i < n; i++) {
      sumX += x[i];
      sumY += y[i];
      sumXY += x[i] * y[i];
      sumX2 += x[i] * x[i];
    }

    var slope = ((n * sumXY) - (sumX * sumY)) / ((n * sumX2) - (sumX * sumX));
    var intercept = (sumY - (slope * sumX)) / n;

    return { slope: slope, intercept: intercept };
  };

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

    fill(0);
    noStroke();
    textAlign('center', 'center');
    textSize(18);
    text(this.title,
         (this.layout.leftMargin + this.layout.rightMargin) / 2,
         this.layout.topMargin / 2);

    stroke(0);
    strokeWeight(1);
    line(this.layout.leftMargin, this.layout.topMargin,
         this.layout.leftMargin, this.layout.bottomMargin);
    line(this.layout.leftMargin, this.layout.bottomMargin,
         this.layout.rightMargin, this.layout.bottomMargin);
```

```
159
160     this.scatterPlot.drawXAxisTicks(this.xRange.min, this.xRange.max,
161                                     this.layout.numXTickLabels,
162                                     'Average Heart Rate (bpm)');
163     this.scatterPlot.drawYAxisTicks(this.yRange.min, this.yRange.max,
164                                     this.layout.numYTickLabels,
165                                     'Pace (min/km)');
166
167     var regression = this.calculateRegression(
168       this.processedData.heartRates,
169       this.processedData.paces
170     );
171     this.drawTrendLine(regression);
172
173     var pointColour = color(66, 133, 244, 180);
174     this.scatterPlot.draw(
175       this.processedData.heartRates,
176       this.processedData.paces,
177       this.xRange,
178       this.yRange,
179       pointColour,
180       this.processedData.labels
181     );
182
183     this.drawCorrelationInfo();
184
185     fill(100);
186     textSize(10);
187     textAlign('left', 'top');
188     text('Hover over points to see details',
189          this.layout.leftMargin,
190          this.layout.bottomMargin + 50);
191   };
192
193   this.drawTrendLine = function(regression) {
194     var x1 = this.xRange.min;
195     var x2 = this.xRange.max;
196     var y1 = regression.slope * x1 + regression.intercept;
197     var y2 = regression.slope * x2 + regression.intercept;
198
199     y1 = constrain(y1, this.yRange.min, this.yRange.max);
200     y2 = constrain(y2, this.yRange.min, this.yRange.max);
201
202     var screenX1 = this.scatterPlot.mapToX(x1, this.xRange.min, this.xRange.max);
203     var screenY1 = this.scatterPlot.mapToY(y1, this.yRange.min, this.yRange.max);
204     var screenX2 = this.scatterPlot.mapToX(x2, this.xRange.min, this.xRange.max);
205     var screenY2 = this.scatterPlot.mapToY(y2, this.yRange.min, this.yRange.max);
206
207     stroke(220, 20, 60);
208     strokeWeight(2);
209     line(screenX1, screenY1, screenX2, screenY2);
210   };
211
212   this.drawCorrelationInfo = function() {
```

```
    var correlation = this.calculateCorrelation(
      this.processedData.heartRates,
      this.processedData.paces
    );

    var infoX = this.layout.rightMargin - 180;
    var infoY = this.layout.topMargin + 10;

    fill(255, 255, 255, 200);
    stroke(200);
    strokeWeight(1);
    rect(infoX - 10, infoY - 5, 180, 70, 5);

    fill(0);
    noStroke();
    textAlign('left', 'top');
    textSize(11);

    text('Correlation: ' + correlation.toFixed(3), infoX, infoY);
    text('Data points: ' + this.processedData.heartRates.length, infoX, infoY +
16);

    textSize(10);
    fill(100);
    var interpretation;
    if (correlation < -0.3) {
      interpretation = 'Negative correlation: higher HR,';
      text(interpretation, infoX, infoY + 36);
      text('faster pace (lower number)', infoX, infoY + 48);
    } else if (correlation > 0.3) {
      interpretation = 'Positive correlation: higher HR,';
      text(interpretation, infoX, infoY + 36);
      text('slower pace', infoX, infoY + 48);
    } else {
      text('Weak correlation', infoX, infoY + 36);
    }
  };
}
/* End of my own code */

```