

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

```
1  <!DOCTYPE html>
2  <html lang="en">
3    <head>
4      <meta charset="utf-8">
5      <meta http-equiv="X-UA-Compatible" content="IE=edge">
6      <meta name="viewport" content="width=device-width, initial-scale=1">
7      <title>Running Data Visualisation</title>
8
9      <!-- Libraries -->
10     <script src="lib/p5.min.js"></script>
11
12     <link rel="stylesheet" href="style.css">
13
14     <!-- Main sketch file -->
15     <script src="sketch.js"></script>
16
17     <!-- Helper functions and core components (from template) -->
18     <script src="helper-functions.js"></script>
19     <script src="gallery.js"></script>
20     <script src="pie-chart.js"></script>
21
22     <!-- Start of my own code -->
23     <!-- New reusable chart constructors -->
24     <script src="bar-chart.js"></script>
25     <script src="scatter-plot.js"></script>
26
27     <!-- New running data visualisations -->
28     <script src="monthly-distance.js"></script>
29     <script src="pace-progress.js"></script>
30     <script src="activity-types.js"></script>
31     <script src="heartrate-vs-pace.js"></script>
32     <!-- End of my own code -->
33   </head>
34   <body>
35     <div id="app" class="container">
36       <ul id="visuals-menu"></ul>
37     </div>
38   </body>
39 </html>
40
```

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

```
1  /* Start of my own code */
2  function BarChart(layout) {
3      this.layout = layout;
4      this.barWidth = 30;
5      this.barGap = 10;
6
7      this.draw = function(data, labels, colours, title) {
8          if (data.length == 0) {
9              console.log('Data has length zero!');
10             return;
11         }
12
13         if (labels.length != data.length) {
14             console.log('Data and labels arrays must be the same length!');
15             return;
16         }
17
18         var plotWidth = this.layout.rightMargin - this.layout.leftMargin;
19         var plotHeight = this.layout.bottomMargin - this.layout.topMargin;
20         this.barWidth = (plotWidth - (data.length + 1) * this.barGap) / data.length;
21         var maxValue = max(data);
22
23         if (title) {
24             noStroke();
25             fill(0);
26             textAlign('center', 'center');
27             textSize(16);
28             text(title,
29                 (this.layout.leftMargin + this.layout.rightMargin) / 2,
30                 this.layout.topMargin - 20);
31         }
32
33         for (var i = 0; i < data.length; i++) {
34             var x = this.layout.leftMargin + this.barGap + i * (this.barWidth +
this.barGap);
35             var barHeight = map(data[i], 0, maxValue, 0, plotHeight);
36             var y = this.layout.bottomMargin - barHeight;
37
38             if (Array.isArray(colours)) {
39                 fill(colours[i % colours.length]);
40             } else {
41                 fill(colours);
42             }
43
44             stroke(0);
45             strokeWeight(1);
46             rect(x, y, this.barWidth, barHeight);
47
48             noStroke();
49             fill(0);
50             textAlign('center', 'bottom');
51             textSize(10);
```

```
52     text(data[i].toFixed(1), x + this.barWidth / 2, y - 5);
53
54     push();
55     translate(x + this.barWidth / 2, this.layout.bottomMargin + 10);
56     rotate(radians(-45));
57     textAlign('right', 'center');
58     textSize(10);
59     text(labels[i], 0, 0);
60     pop();
61 }
62 };
63
64 this.drawYAxisTickLabels = function(minValue, maxValue, numTicks, decimalPlaces)
65 {
66     var range = maxValue - minValue;
67     var tickStep = range / numTicks;
68
69     fill(0);
70     noStroke();
71     textAlign('right', 'center');
72     textSize(12);
73
74     for (var i = 0; i <= numTicks; i++) {
75         var value = minValue + (i * tickStep);
76         var y = map(value, minValue, maxValue,
77                     this.layout.bottomMargin,
78                     this.layout.topMargin);
79
80         text(value.toFixed(decimalPlaces),
81              this.layout.leftMargin - 10,
82              y);
83
84         stroke(220);
85         strokeWeight(1);
86         line(this.layout.leftMargin, y,
87              this.layout.rightMargin, y);
88     }
89 };
90 /* End of my own code */
91
```

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

```
1  /* Start of my own code */
2  function ScatterPlot(layout) {
3      this.layout = layout;
4      this.pointSize = 8;
5
6      this.draw = function(xData, yData, xRange, yRange, colour, labels) {
7          if (xData.length == 0 || yData.length == 0) {
8              console.log('Data has length zero!');
9              return;
10         }
11
12         if (xData.length != yData.length) {
13             console.log('X and Y data arrays must be the same length!');
14             return;
15         }
16
17         for (var i = 0; i < xData.length; i++) {
18             var x = this.mapToX(xData[i], xRange.min, xRange.max);
19             var y = this.mapToY(yData[i], yRange.min, yRange.max);
20             var isHovered = dist(mouseX, mouseY, x, y) < this.pointSize;
21
22             if (isHovered) {
23                 fill(255, 100, 100);
24                 noStroke();
25                 ellipse(x, y, this.pointSize * 1.5, this.pointSize * 1.5);
26                 this.drawTooltip(x, y, xData[i], yData[i], labels ? labels[i] : null);
27             } else {
28                 fill(colour);
29                 stroke(0);
30                 strokeWeight(0.5);
31                 ellipse(x, y, this.pointSize, this.pointSize);
32             }
33         }
34     };
35
36     this.mapToX = function(value, minVal, maxVal) {
37         return map(value, minVal, maxVal,
38             this.layout.leftMargin,
39             this.layout.rightMargin);
40     };
41
42     this.mapToY = function(value, minVal, maxVal) {
43         return map(value, minVal, maxVal,
44             this.layout.bottomMargin,
45             this.layout.topMargin);
46     };
47
48     this.drawTooltip = function(x, y, xVal, yVal, label) {
49         var tooltipWidth = 120;
50         var tooltipHeight = 50;
51         var tooltipX = x + 10;
```

```
52     var tooltipY = y - tooltipHeight - 10;
53
54     if (tooltipX + tooltipWidth > this.layout.rightMargin) {
55         tooltipX = x - tooltipWidth - 10;
56     }
57     if (tooltipY < this.layout.topMargin) {
58         tooltipY = y + 10;
59     }
60
61     fill(255, 255, 255, 230);
62     stroke(100);
63     strokeWeight(1);
64     rect(tooltipX, tooltipY, tooltipWidth, tooltipHeight, 5);
65
66     fill(0);
67     noStroke();
68     textAlign('left', 'top');
69     textSize(10);
70
71     var yText = 'Pace: ' + this.formatPace(yVal) + '/km';
72     var xText = 'HR: ' + xVal.toFixed(0) + ' bpm';
73
74     text(xText, tooltipX + 5, tooltipY + 5);
75     text(yText, tooltipX + 5, tooltipY + 20);
76
77     if (label) {
78         text(label, tooltipX + 5, tooltipY + 35);
79     }
80 };
81
82 this.formatPace = function(paceMinutes) {
83     var minutes = Math.floor(paceMinutes);
84     var seconds = Math.round((paceMinutes - minutes) * 60);
85     if (seconds < 10) {
86         return minutes + ':0' + seconds;
87     }
88     return minutes + ':' + seconds;
89 };
90
91 this.drawXAxisTicks = function(minVal, maxVal, numTicks, label) {
92     var range = maxVal - minVal;
93     var step = range / numTicks;
94
95     fill(0);
96     noStroke();
97     textAlign('center', 'top');
98     textSize(10);
99
100    for (var i = 0; i <= numTicks; i++) {
101        var value = minVal + (i * step);
102        var x = this.mapToX(value, minVal, maxVal);
103
104        stroke(0);
105        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

```
1  /* Start of my own code */
2  function MonthlyDistance() {
3      this.name = 'Monthly Distance';
4      this.id = 'monthly-distance';
5      this.loaded = false;
6
7      var marginSize = 35;
8      this.layout = {
9          marginSize: marginSize,
10         leftMargin: marginSize * 2,
11         rightMargin: width - marginSize,
12         topMargin: marginSize * 2,
13         bottomMargin: height - marginSize * 3,
14         pad: 5,
15         plotWidth: function() {
16             return this.rightMargin - this.leftMargin;
17         },
18         plotHeight: function() {
19             return this.bottomMargin - this.topMargin;
20         },
21         grid: true,
22         numYTickLabels: 6
23     };
24
25     this.barChart = new BarChart(this.layout);
26
27     this.preload = function() {
28         var self = this;
29         this.data = loadTable(
30             './data/running/activities.csv', 'csv', 'header',
31             function(table) {
32                 self.loaded = true;
33             });
34     };
35
36     this.setup = function() {
37         if (!this.loaded) {
38             console.log('Data not yet loaded');
39             return;
40         }
41         this.monthlyData = this.aggregateByMonth();
42     };
43
44     this.destroy = function() {};
45
46     this.aggregateByMonth = function() {
47         var months = {};
48
49         for (var i = 0; i < this.data.getRowCount(); i++) {
50             var dateStr = this.data.getString(i, 'date');
51             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,
108     0);
109
110     var barColour = color(66, 133, 244);
111     this.barChart.draw(
112         this.monthlyData.values,
113         this.monthlyData.labels,
114         barColour,
115         null
116     );
117
118     var totalDistance = this.monthlyData.values.reduce(function(a, b) {
119         return a + b;
120     }, 0);
121
122     textSize(12);
123     textAlign('left', 'top');
124     fill(100);
125     text('Total Distance: ' + totalDistance.toFixed(1) + ' km',
126         this.layout.leftMargin,
127         this.layout.bottomMargin + 60);
128 }
129 /* End of my own code */
130
```

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

```
1  /* Start of my own code */
2  function PaceProgress() {
3      this.name = 'Pace Progress';
4      this.id = 'pace-progress';
5      this.title = 'Running Pace Progress Over Time';
6      this.xAxisLabel = 'Date';
7      this.yAxisLabel = 'Pace (min/km)';
8
9      var marginSize = 35;
10     this.layout = {
11         marginSize: marginSize,
12         leftMargin: marginSize * 2,
13         rightMargin: width - marginSize,
14         topMargin: marginSize * 2,
15         bottomMargin: height - marginSize * 2,
16         pad: 5,
17         plotWidth: function() {
18             return this.rightMargin - this.leftMargin;
19         },
20         plotHeight: function() {
21             return this.bottomMargin - this.topMargin;
22         },
23         grid: true,
24         numXTickLabels: 8,
25         numYTickLabels: 8
26     };
27
28     this.loaded = false;
29
30     this.preload = function() {
31         var self = this;
32         this.data = loadTable(
33             './data/running/activities.csv', 'csv', 'header',
34             function(table) {
35                 self.loaded = true;
36             });
37     };
38
39     this.setup = function() {
40         if (!this.loaded) {
41             console.log('Data not yet loaded');
42             return;
43         }
44
45         this.processedData = this.processData();
46         this.startDate = this.processedData.dates[0];
47         this.endDate = this.processedData.dates[this.processedData.dates.length - 1];
48         this.minPace = Math.floor(min(this.processedData.paces)) - 0.5;
49         this.maxPace = Math.ceil(max(this.processedData.paces)) + 0.5;
50     };
51 }
```

```
52   this.destroy = function() {};  
53  
54   this.paceToMinutes = function(paceStr) {  
55     var parts = paceStr.split(':');  
56     if (parts.length == 2) {  
57       return parseInt(parts[0]) + parseInt(parts[1]) / 60;  
58     }  
59     return 0;  
60   };  
61  
62   this.processData = function() {  
63     var dates = [];  
64     var paces = [];  
65     var distances = [];  
66  
67     for (var i = 0; i < this.data.getRowCount(); i++) {  
68       var activityType = this.data.getString(i, 'activity_type');  
69  
70       if (activityType === 'Run') {  
71         var dateStr = this.data.getString(i, 'date');  
72         var paceStr = this.data.getString(i, 'avg_pace');  
73         var distance = this.data.getNum(i, 'distance');  
74  
75         if (distance >= 3) {  
76           var paceMinutes = this.paceToMinutes(paceStr);  
77  
78           if (paceMinutes >= 4 && paceMinutes <= 10) {  
79             dates.push(new Date(dateStr));  
80             paces.push(paceMinutes);  
81             distances.push(distance);  
82           }  
83         }  
84       }  
85     }  
86  
87     var combined = [];  
88     for (var j = 0; j < dates.length; j++) {  
89       combined.push({  
90         date: dates[j],  
91         pace: paces[j],  
92         distance: distances[j]  
93       });  
94     }  
95     combined.sort(function(a, b) {  
96       return a.date - b.date;  
97     });  
98  
99     var sortedDates = [];  
100    var sortedPaces = [];  
101    var sortedDistances = [];  
102    for (var k = 0; k < combined.length; k++) {  
103      sortedDates.push(combined[k].date);  
104      sortedPaces.push(combined[k].pace);  
105      sortedDistances.push(combined[k].distance);
```

```
106     }
107
108     return {
109         dates: sortedDates,
110         paces: sortedPaces,
111         distances: sortedDistances
112     };
113 };
114
115 // Moving average smooths out short-term fluctuations
116 this.calculateMovingAverage = function(data, windowSize) {
117     var result = [];
118     for (var i = 0; i < data.length; i++) {
119         var start = Math.max(0, i - windowSize + 1);
120         var sum = 0;
121         var count = 0;
122         for (var j = start; j <= i; j++) {
123             sum += data[j];
124             count++;
125         }
126         result.push(sum / count);
127     }
128     return result;
129 };
130
131 this.draw = function() {
132     if (!this.loaded) {
133         console.log('Data not yet loaded');
134         return;
135     }
136
137     this.drawTitle();
138
139     drawYAxisTickLabels(this.minPace,
140                         this.maxPace,
141                         this.layout,
142                         this.mapPaceToHeight.bind(this),
143                         1);
144
145     drawAxis(this.layout);
146     drawAxisLabels(this.xAxisLabel, this.yAxisLabel, this.layout);
147
148     var dates = this.processedData.dates;
149     var paces = this.processedData.paces;
150
151     for (var i = 0; i < dates.length; i++) {
152         var x = this.mapDateToWidth(dates[i]);
153         var y = this.mapPaceToHeight(paces[i]);
154
155         fill(100, 149, 237, 150);
156         noStroke();
157         ellipse(x, y, 6, 6);
158     }
159 }
```

```
160     var movingAvg = this.calculateMovingAverage(paces, 10);
161     stroke(220, 20, 60);
162     strokeWeight(2);
163     noFill();
164     beginShape();
165     for (var j = 0; j < dates.length; j++) {
166         var trendX = this.mapDateToWidth(dates[j]);
167         var trendY = this.mapPaceToHeight(movingAvg[j]);
168         vertex(trendX, trendY);
169     }
170     endShape();
171
172     this.drawXAxisLabels();
173     this.drawLegend();
174
175     fill(100);
176     noStroke();
177     textSize(10);
178     textAlign('left', 'top');
179     text('Note: Lower pace = faster running',
180         this.layout.leftMargin,
181         this.layout.bottomMargin + 45);
182 };
183
184 this.drawTitle = function() {
185     fill(0);
186     noStroke();
187     textAlign('center', 'center');
188     textSize(18);
189     text(this.title,
190         (this.layout.plotWidth() / 2) + this.layout.leftMargin,
191         this.layout.topMargin / 2);
192 };
193
194 this.drawXAxisLabels = function() {
195     var dates = this.processedData.dates;
196     var numLabels = Math.min(this.layout.numXTickLabels, dates.length);
197     var step = Math.floor(dates.length / numLabels);
198
199     fill(0);
200     noStroke();
201     textSize(10);
202     textAlign('center', 'top');
203
204     for (var i = 0; i < dates.length; i += step) {
205         var x = this.mapDateToWidth(dates[i]);
206         var dateLabel = this.formatDate(dates[i]);
207
208         stroke(0);
209         line(x, this.layout.bottomMargin, x, this.layout.bottomMargin + 5);
210
211         noStroke();
212         text(dateLabel, x, this.layout.bottomMargin + 8);
213     }
```

```
214     };
215
216     this.formatDate = function(date) {
217         var months = ['Jan', 'Feb', 'Mar', 'Apr', 'May', 'Jun',
218             'Jul', 'Aug', 'Sep', 'Oct', 'Nov', 'Dec'];
219         return months[date.getMonth()] + ' ' + (date.getFullYear() % 100);
220     };
221
222     this.drawLegend = function() {
223         var legendX = this.layout.rightMargin - 150;
224         var legendY = this.layout.topMargin + 10;
225
226         fill(100, 149, 237);
227         noStroke();
228         ellipse(legendX, legendY, 6, 6);
229         fill(0);
230         textAlign('left', 'center');
231         textSize(11);
232         text('Individual runs', legendX + 10, legendY);
233
234         stroke(220, 20, 60);
235         strokeWeight(2);
236         line(legendX - 10, legendY + 20, legendX + 10, legendY + 20);
237         noStroke();
238         fill(0);
239         text('10-run moving avg', legendX + 15, legendY + 20);
240     };
241
242     this.mapDateToWidth = function(date) {
243         var startTime = this.startDate.getTime();
244         var endTime = this.endDate.getTime();
245         return map(date.getTime(),
246             startTime,
247             endTime,
248             this.layout.leftMargin,
249             this.layout.rightMargin);
250     };
251
252     this.mapPaceToHeight = function(value) {
253         return map(value,
254             this.minPace,
255             this.maxPace,
256             this.layout.topMargin,
257             this.layout.bottomMargin);
258     };
259 }
260 /* End of my own code */
261
```

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

```
1  /* Start of my own code */
2  function ActivityTypes() {
3      this.name = 'Activity Types';
4      this.id = 'activity-types';
5      this.loaded = false;
6
7      this.pie = new PieChart(width / 2, height / 2, width * 0.35);
8
9      this.preload = function() {
10         var self = this;
11         this.data = loadTable(
12             './data/running/activities.csv', 'csv', 'header',
13             function(table) {
14                 self.loaded = true;
15             });
16     };
17
18     this.setup = function() {
19         if (!this.loaded) {
20             console.log('Data not yet loaded');
21             return;
22         }
23
24         this.activityCounts = this.countActivityTypes();
25         this.distanceByType = this.sumDistanceByType();
26
27         this.select = createSelect();
28         this.select.position(350, 40);
29         this.select.option('By Count');
30         this.select.option('By Distance');
31     };
32
33     this.destroy = function() {
34         this.select.remove();
35     };
36
37     this.countActivityTypes = function() {
38         var counts = {};
39
40         for (var i = 0; i < this.data.getRowCount(); i++) {
41             var activityType = this.data.getString(i, 'activity_type');
42
43             if (counts[activityType]) {
44                 counts[activityType]++;
45             } else {
46                 counts[activityType] = 1;
47             }
48         }
49
50         return counts;
51     };
```



```
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);
```

```
106     var total = 0;
107     for (var i = 0; i < labels.length; i++) {
108         total += data[labels[i]];
109     }
110
111     var statsX = 80;
112     var statsY = height - 100;
113
114     fill(0);
115     noStroke();
116     textAlign('left', 'top');
117     textSize(12);
118
119     if (viewMode === 'By Count') {
120         text('Total Activities: ' + total, statsX, statsY);
121     } else {
122         text('Total Distance: ' + total.toFixed(1) + ' km', statsX, statsY);
123     }
124
125     textSize(11);
126     for (var j = 0; j < labels.length; j++) {
127         var percentage = ((data[labels[j]] / total) * 100).toFixed(1);
128         var valueStr = viewMode === 'By Count'
129             ? data[labels[j]] + ' activities'
130             : data[labels[j]].toFixed(1) + ' km';
131         text(labels[j] + ': ' + valueStr + ' (' + percentage + '%)',
132             statsX, statsY + 18 + (j * 16));
133     }
134 };
135 }
136 /* End of my own code */
137
```

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

```
1  /* Start of my own code */
2  function HeartRateVsPace() {
3      this.name = 'Heart Rate vs Pace';
4      this.id = 'heartrate-vs-pace';
5      this.title = 'Heart Rate vs Running Pace';
6
7      var marginSize = 35;
8      this.layout = {
9          marginSize: marginSize,
10         leftMargin: marginSize * 2.5,
11         rightMargin: width - marginSize,
12         topMargin: marginSize * 2,
13         bottomMargin: height - marginSize * 2.5,
14         pad: 5,
15         plotWidth: function() {
16             return this.rightMargin - this.leftMargin;
17         },
18         plotHeight: function() {
19             return this.bottomMargin - this.topMargin;
20         },
21         grid: true,
22         numXTickLabels: 6,
23         numYTickLabels: 6
24     };
25
26     this.scatterPlot = new ScatterPlot(this.layout);
27     this.loaded = false;
28
29     this.preload = function() {
30         var self = this;
31         this.data = loadTable(
32             './data/running/activities.csv', 'csv', 'header',
33             function(table) {
34                 self.loaded = true;
35             });
36     };
37
38     this.setup = function() {
39         if (!this.loaded) {
40             console.log('Data not yet loaded');
41             return;
42         }
43
44         this.processedData = this.processData();
45
46         this.xRange = {
47             min: Math.floor(min(this.processedData.heartRates) / 10) * 10 - 5,
48             max: Math.ceil(max(this.processedData.heartRates) / 10) * 10 + 5
49         };
50
51         this.yRange = {
```

```
52     min: Math.floor(min(this.processedData.paces) * 2) / 2 - 0.5,
53     max: Math.ceil(max(this.processedData.paces) * 2) / 2 + 0.5
54 };
55 };
56
57 this.destroy = function() {};
58
59 this.paceToMinutes = function(paceStr) {
60     var parts = paceStr.split(':');
61     if (parts.length == 2) {
62         return parseInt(parts[0]) + parseInt(parts[1]) / 60;
63     }
64     return 0;
65 };
66
67 this.processData = function() {
68     var heartRates = [];
69     var paces = [];
70     var labels = [];
71
72     for (var i = 0; i < this.data.getRowCount(); i++) {
73         var activityType = this.data.getString(i, 'activity_type');
74
75         if (activityType === 'Run') {
76             var heartRate = this.data.getNum(i, 'avg_heart_rate');
77             var paceStr = this.data.getString(i, 'avg_pace');
78             var distance = this.data.getNum(i, 'distance');
79             var date = this.data.getString(i, 'date');
80
81             if (heartRate > 0 && distance >= 3) {
82                 var paceMinutes = this.paceToMinutes(paceStr);
83
84                 if (paceMinutes >= 5 && paceMinutes <= 9) {
85                     heartRates.push(heartRate);
86                     paces.push(paceMinutes);
87                     labels.push(date + ' - ' + distance.toFixed(1) + 'km');
88                 }
89             }
90         }
91     }
92
93     return {
94         heartRates: heartRates,
95         paces: paces,
96         labels: labels
97     };
98 };
99
100 // Pearson correlation coefficient: measures linear relationship between two
    variables
101 this.calculateCorrelation = function(x, y) {
102     var n = x.length;
103     var sumX = 0, sumY = 0, sumXY = 0, sumX2 = 0, sumY2 = 0;
104
```

```
105     for (var i = 0; i < n; i++) {
106         sumX += x[i];
107         sumY += y[i];
108         sumXY += x[i] * y[i];
109         sumX2 += x[i] * x[i];
110         sumY2 += y[i] * y[i];
111     }
112
113     var numerator = (n * sumXY) - (sumX * sumY);
114     var denominator = Math.sqrt(((n * sumX2) - (sumX * sumX)) *
115                                 ((n * sumY2) - (sumY * sumY)));
116
117     if (denominator === 0) return 0;
118     return numerator / denominator;
119 };
120
121 // Least squares linear regression: finds best fit line y = mx + b
122 this.calculateRegression = function(x, y) {
123     var n = x.length;
124     var sumX = 0, sumY = 0, sumXY = 0, sumX2 = 0;
125
126     for (var i = 0; i < n; i++) {
127         sumX += x[i];
128         sumY += y[i];
129         sumXY += x[i] * y[i];
130         sumX2 += x[i] * x[i];
131     }
132
133     var slope = ((n * sumXY) - (sumX * sumY)) / ((n * sumX2) - (sumX * sumX));
134     var intercept = (sumY - (slope * sumX)) / n;
135
136     return { slope: slope, intercept: intercept };
137 };
138
139 this.draw = function() {
140     if (!this.loaded) {
141         console.log('Data not yet loaded');
142         return;
143     }
144
145     fill(0);
146     noStroke();
147     textAlign('center', 'center');
148     textSize(18);
149     text(this.title,
150          (this.layout.leftMargin + this.layout.rightMargin) / 2,
151          this.layout.topMargin / 2);
152
153     stroke(0);
154     strokeWeight(1);
155     line(this.layout.leftMargin, this.layout.topMargin,
156          this.layout.leftMargin, this.layout.bottomMargin);
157     line(this.layout.leftMargin, this.layout.bottomMargin,
158          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() {
```

```
213     var correlation = this.calculateCorrelation(  
214         this.processedData.heartRates,  
215         this.processedData.paces  
216     );  
217  
218     var infoX = this.layout.rightMargin - 180;  
219     var infoY = this.layout.topMargin + 10;  
220  
221     fill(255, 255, 255, 200);  
222     stroke(200);  
223     strokeWeight(1);  
224     rect(infoX - 10, infoY - 5, 180, 70, 5);  
225  
226     fill(0);  
227     noStroke();  
228     textAlign('left', 'top');  
229     textSize(11);  
230  
231     text('Correlation: ' + correlation.toFixed(3), infoX, infoY);  
232     text('Data points: ' + this.processedData.heartRates.length, infoX, infoY +  
233 16);  
234  
235     textSize(10);  
236     fill(100);  
237     var interpretation;  
238     if (correlation < -0.3) {  
239         interpretation = 'Negative correlation: higher HR,';  
240         text(interpretation, infoX, infoY + 36);  
241         text('faster pace (lower number)', infoX, infoY + 48);  
242     } else if (correlation > 0.3) {  
243         interpretation = 'Positive correlation: higher HR,';  
244         text(interpretation, infoX, infoY + 36);  
245         text('slower pace', infoX, infoY + 48);  
246     } else {  
247         text('Weak correlation', infoX, infoY + 36);  
248     }  
249 }  
250 /* End of my own code */  
251
```