data-vis\ai-check.js

```
//------------------------------ START NEW CODE ---------------------------------------
-//
function aiCheck() {
    // Name for the visualisation to appear in the menu bar.
    this.name = 'AI Validation';

    // Each visualisation must have a unique ID with no special characters.
    this.id = 'ai usage';

    // Title to display above the plot.
    this.title = 'how often does people double-check AI generated information!';

    // 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/survey/ai_double_check_survey.csv', 'csv', 'header',
        // Callback function to set the value this.loaded to true.
        function(table) {
            self.loaded = true;
        });
    };

    this.setup = function() {
        // Collecting quantity values from quantity column.
        this.quantity = this.data.getColumn("Quantity").map(Number);
        this.doubleCheck = this.data.getColumn("How often do you double-check AI-generated
information for accuracy?");

        // Create array to store bubbles.
        this.aBubble = [];

        // Loop through each quantity and create bubble object.
        this.createBubbles(9);
    };

    this.destroy = function() {
    };

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


        // Anonymous function to draw text.
        this.drawText();
```

```
50
51          // Loop to create a bubble for every quantity and avoid letting them touch each
    other. NESTED LOOP! O(n²).
52          for (let i = 0; i < this.aBubble.length; i++) {
53              for (let j = 0; j < this.aBubble.length; j++) {
54                  if (i !== j) {
55                      let a = this.aBubble[i];
56                      let b = this.aBubble[j];
57
58                      let dx = a.x - b.x;
59                      let dy = a.y - b.y;
60                      let distBetween = sqrt(dx * dx + dy * dy);
61                      let minDist = (a.size + b.size) / 2;
62
63                      if (distBetween < minDist) {
64                          let angle = atan2(dy, dx);
65                          let overlap = minDist - distBetween;
66
67                          // Pushes bubble i out of bubble j.
68                          a.x += cos(angle) * (overlap / 2) * 20;
69                          a.y += sin(angle) * (overlap / 2) * 20;
70                          b.x -= cos(angle) * (overlap / 2) * 20;
71                          b.y -= sin(angle) * (overlap / 2) * 20;
72                      }
73                  }
74              }
75
76              // Function to create and update each bubble.
77              this.updateAndDrawBubble(i);
78          }
79
80          // Function to draw Legend.
81          this.draw5Legend(
82              450,
83              740,
84              "About half the time",
85              "Always",
86              "Most of the time",
87              "Never",
88              "Sometimes"
89          )
90      }
91
92      // Function to draw the text from the "user", according to the tittle.
93      this.drawText = function() {
94          // Draw user message
95          let message_ai_usage = "Check out "
96
97          push();
98          textSize(30);
99          textAlign(LEFT, TOP);
100         textFont(robotoFont);
101         fill(0);
102         noStroke();
```

```
        text(message_ai_usage, 440, 125.5);
        textFont(robotoFontBold);
        text(this.title, 570, 125.5)
        pop();
    }

    // Function to create the bubbles.
    this.createBubbles = function (sizeValue) {

        // Colors.
        let bubbleColors = ['#FFCD6E', '#688E26', '#1B4D3E', '#C8102E', '#C99A00'];

        // Loop for bubbles.
        for (let i = 0; i < this.data.getRowCount(); i++) {
            let size = this.quantity[i] * sizeValue;

            // Create one bubble object.
            let bubble = {
                x: random(500, 1200),
                y: random(300, 600),
                // x: 400 + i * size * 0.9,
                // y: 400 + i * size * 0.4,
                velocityX: random(-0.2, + 0.2),
                velocityY: random(-0.2, + 0.2),
                size: size,
                color: (bubbleColors[i])
            };

            // Add bubble to array.
            this.aBubble.push(bubble);
        }

    }

    // Function to create and update the bubbles.
    this.updateAndDrawBubble = function (i) {
        // Update position.
        this.aBubble[i].x += this.aBubble[i].velocityX;
        this.aBubble[i].y += this.aBubble[i].velocityY;

        // Interaction with mouse.
        let d = dist(mouseX, mouseY, this.aBubble[i].x, this.aBubble[i].y);
        if (d < this.aBubble[i].size / 2) {
            // Calculate the distance between the mouse and the aBubble (in x and y).
            let dx = this.aBubble[i].x - mouseX;
            let dy = this.aBubble[i].y - mouseY;

            // Calculate the angle of the vector.
            let angle = atan2(dy, dx);

            // Tranform how much to walk in x and y, considering cos to the movement in x and and sin the movement in y.
            this.aBubble[i].x += cos(angle) * 4;
            this.aBubble[i].y += sin(angle) * 4;
```

```
        }

        // Boundary reflection for x on the left.
        if (this.aBubble[i].x < (menuLeft.x + menuLeft.w + 60 + this.aBubble[i].size / 2))
{
            this.aBubble[i].velocityX *= -1;
            this.aBubble[i].x = menuLeft.x + menuLeft.w + 60 + this.aBubble[i].size / 2
        }

        // Boundary reflection for x on the right.
        if (this.aBubble[i].x > (menuLeft.x + menuLeft.w + 60 + width - 420 -
this.aBubble[i].size / 2)) {
            this.aBubble[i].velocityX *= -1;
            this.aBubble[i].x = menuLeft.x + menuLeft.w + 60 + width - 420 -
this.aBubble[i].size / 2;
        }

        // Boundary reflection for y on top.
        if (this.aBubble[i].y < menuTop.h + 80 + this.aBubble[i].size / 2) {
            this.aBubble[i].velocityY *= -1;
            this.aBubble[i].y = menuTop.h + 80 + this.aBubble[i].size / 2;
        }

        // Boundary reflection for y on the bottom.
        if (this.aBubble[i].y > menuTop.w - 805 + menuTop.h + 80 - (this.aBubble[i].size /
2) - 120) {
            this.aBubble[i].velocityY *= -1;
            this.aBubble[i].y = menuTop.w - 805 + menuTop.h + 80 - (this.aBubble[i].size /
2) - 120;
        }

        // Draw.
        fill(this.aBubble[i].color);
        noStroke();
        ellipse(this.aBubble[i].x,
                this.aBubble[i].y,
                this.aBubble[i].size,
                this.aBubble[i].size);

        textAlign(CENTER, CENTER);
        textFont(robotoFont);
        textSize(this.aBubble[i].size * 0.3);
        fill(255);
    text(this.quantity[i] + "%", this.aBubble[i].x, this.aBubble[i].y);
    };

    // Function to draw the legend.
    this.draw5Legend = function (xPos, yPos, labelText1, labelText2, labelText3,
labelText4, labelText5) {
        rightOffset = 300;
        textFont(robotoFont);
        textAlign(LEFT, CENTER);
        textSize(16);
        noFill();
```

```
        noStroke();

        fill('#FFCD6E');
        rect(xPos + rightOffset, yPos, 15, 15, 3);
        fill(0);
        text(labelText1, xPos + 25 + rightOffset, yPos + 6);

        fill('#688E26');
        rect(xPos + 160 + rightOffset, yPos, 15, 15, 3);
        fill(0);
        text(labelText2, xPos + 180 + rightOffset, yPos + 6);

        fill('#1B4D3E');
        rect(xPos + 240 + rightOffset, yPos, 15, 15, 3);
        fill(0);
        text(labelText3, xPos + 260 + rightOffset, yPos + 6);

        fill('#C8102E');
        rect(xPos + 380 + rightOffset, yPos, 15, 15, 3);
        fill(0);
        text(labelText4, xPos + 400 + rightOffset, yPos + 6);

        fill('#C99A00');
        rect(xPos + 460 + rightOffset, yPos, 15, 15, 3);
        fill(0);
        text(labelText5, xPos + 480 + rightOffset, yPos + 6);
    }
}
//------------------------------ END NEW CODE ----------------------------------------//
```