Developing Interactive Mathematical Activities in JavaScript

G14PJA = MATH4041

Mathematics 3rd Year Project

Autumn 2021/22

School of Mathematical Sciences

University of Nottingham

Thomas Hanson

Supervisor: Dr. Edward Hall

Assessment type: Investigation

I have read and understood the School and University guidelines on plagiarism. I confirm that this work is my own, apart from the acknowledged references.

Abstract

This report documents the creation of two mathematical tools/activities using HyperText Markup Language (HTML), JavaScript, and Cascading Style Sheets (CSS), designed to aid understanding of statistical/probabilistic concepts. The activities are in the form of web pages making them accessible to most students as they will not need any specialised software.

The activities created are:

1. A visual representation of the normal probability density function. The user is able to change the mean and standard deviation and visualise how this affects the graph.

2. A demonstration of the central limit theorem using multiple distributions. The user can select which distribution they would like to generate samples from. Samples are then generated and the mean values are displayed on a histogram. This tool is aimed at both A-Level and first-year university students.

Contents

1 Introduction

The aim of this project is to create two interactive tools/activities for use in schools in order to aid understanding of probabilistic/statistical concepts.

In this report, section 2 discusses the benefits of interactivity in learning as well as the technologies used to create the activities. Section 3 covers the creation of the first activity, based around the normal distribution and section 4 covers the second activity, based around the central limit theorem. The full source code for both tools are given in appendix A and B. Section 5 sums up the project and outlines further work that can be carried out.

2 Background

2.1 Interactive Learning

Interactive learning has always been fundamental to education, encouraging students to actively engage with a topic. Use of technology has further developed interactivity in recent years due to the increased availability of ICT in schools and at home as well as the rise in on-line teaching due to the COVID-19 pandemic.

In chapter 4 of [2], Edgar Dale introduces the concept of the *Cone of Experience* (figure 1), a model which describes the concreteness of various methods of educational delivery. At the top of the cone, are the least concrete methods of teaching such as *verbal symbols* and *visual symbols* - less interactive methods. Methods towards the bottom of the cone which are more interactive, such as *demonstrations* and *direct purposeful experiences*, are more concrete.

Whilst the the world wide web remained decades away from invention when this model was created in 1969, the activities developed in this project would likely be categorised as contrived experiences or direct purposeful experiences as they allow students to "learn by doing" ([2], p111), albeit in a somewhat abstract way. In either category, the activities are at the bottom of the cone meaning they are highly effective.

Furthermore, on pages 10 and 11 of [3], Jerome S. Bruner describes three main methods of learning - *enactive*, *iconic*, and *symbolic*. Enactive learning being learning through actions,

Figure 1: The Cone of Experience - [2], p107

iconic being learning through visual imagery, and symbolic being learning through words and language. Bruner goes on to describe how each method of learning develops understanding of a topic in a different way. Therefore, to be as effective as possible, the activities should explore all three areas, with interactive elements and exercises covering the enactive method, graphs and other imagery covering the iconic method, and explanations and labels covering the symbolic method.

2.2 HTML, JavaScript, and CSS

In order to create a web page, three coding languages are required to interact with each other.

- *HTML (HyperText Markup Language)* is responsible for the main structure of the web page.
- *JavaScript* is responsible for adding higher-level functionality including facilitating input and output, data manipulation. At the time of writing, 97.8% of all websites use

JavaScript [7] making it a core web technology.

- *CSS (Cascading Style Sheets)* is a language responsible for the design and presentation of the web page. For example, whilst HTML would be responsible for adding a box around a paragraph of text, CSS could be used to specify the colour of the box or whether it has rounded corners.

Applying this to our activities, HTML will be used to define the overall structure such as adding buttons to interact with the activity, titles, and labels; JavaScript will be used to add the functionality of the activity, for example drawing graphs or performing calculations; and CSS will be used to ensure the web page is user-friendly.

2.3 Other Technologies

There are other technologies which would allow similar activities to be created including Desmos [4], Geogebra [5], and Autograph [6]. Using these technologies may make the creation of such activities easier with some also adding features such as the ability to assign certain activities to students, however, this adds a barrier to entry as using the tools may require additional downloads, signing in to an account, or difficulty accessing the site for example during an outage. By using HTML, JavaScript, and CSS, the activities will have fewer barriers and can either be hosted on a web site or accessed by opening the HTML files.

3 Activity 1: Normal PDF

3.1 Overview

The first activity is based around the normal distribution. From personal experience, A-Level students often find it difficult to visualise how changes to the mean, μ, and standard deviation, σ of a normal distribution affect its probability density function (PDF). In this section, we create a tool which displays a graph of the normal PDF and allows the user to change the parameters. The graph will then update to show how these changes affect it.

Required features of this activity:

- PDF of Normal distribution drawn on screen
- Ability for the user to change the parameters of the distribution, affecting the graph accordingly
- Tool to calculate x given α such that either $P(X \leq x) = \alpha$ or $P(X \geq x) = \alpha$
- Tool to calculate α given x such that either $P(X \leq x) = \alpha$ or $P(X \geq x) = \alpha$
- Ability to show the above values of α and x on the graph via shading

3.2 Plotting the Normal PDF

First we plot a standard normal PDF. To do this, we create a function which calculates the value of the PDF at a given point, x, using the standard formula

$$p_X(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}.$$

For now, μ and σ will be hard-coded as 0 and 1, respectively, as this reduces the scope for errors in our code. We will change this later when we add user input. In JavaScript, we use the inbuilt *Math* object to perform several of the required operations including square root, powers, and recalling the value of e. An example of this function implemented in JavaScript is shown in listing 1 below.

Listing 1: Normal PDF function

```
function p(x){
// Hard code mean and sd
  var mean = 0;
  var sd = 1;
// Calculate fraction in front of exponential
  var o2pi = 1/(Math.sqrt(2*Math.PI)*sd);
// Calculate exponential part of the formula
```

```
  var ez = Math.exp(-(Math.pow((x - m)/s, 2))/2); // Calculates e^(-(z^2)/2)
// Return the two values multiplied together
  return (o2pi * ex);
}
```

This function is then iteratively called to build an array of data points which can be plotted. A good balance between computation speed and precision is calculating the points in increments of 0.1 for $x \in [-5, 5]$, giving us 10001 points.

To draw the graph, we will use an open-source package called Chart.js [8]. Following the documentation on chartjs.org, we create a line chart using the data points that were generated from our function, as shown in figure 2.

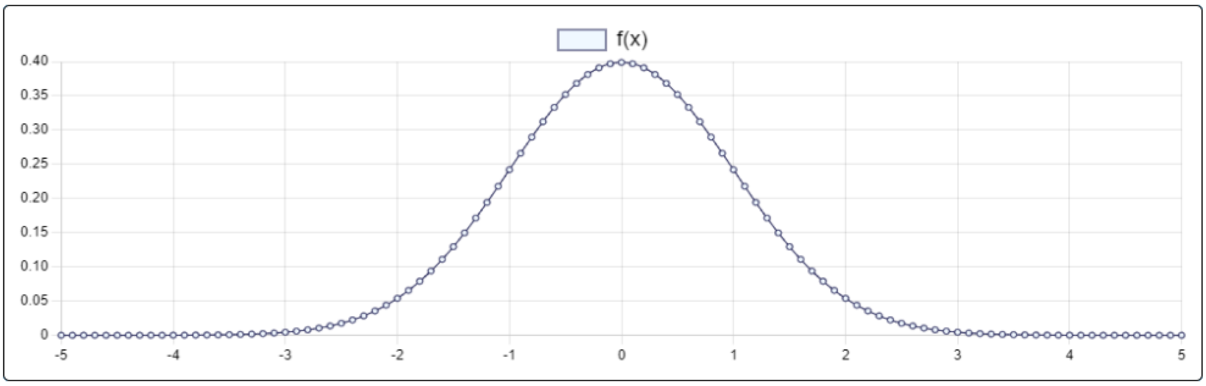


Figure 2: PDF of a $N(0, 1)$ distribution, drawn in Chart.js

We will now add the ability to change μ and σ. First we must add boxes for user input. We could instead use sliders, which would be easier to use on a tablet or other touch-screen device, but input boxes will allow for more precise control. We then add a condition for the boxes to call the graph drawing function when the contents of either box changes using the *onInput* event, as shown in listing 2 below.

Listing 2: Mean and standard deviation input boxes

```
<!-- HTML code for mean and standard deviation input -->
<input onInput="createGraph()" type="text" name="mean" value="0">
<input onInput="createGraph()" type="text" name="sd" value="1">
```

The numeric values entered in the boxes are passed to the graphing function from listing 1 which is edited to use these values instead of the hard-coded μ and σ. The boxes have default values which create the standard uniform distribution when the page is first opened. Adding labels and CSS styling, our page is as shown in figure 3. A reset button was also added to allow the user to revert back to the default values. This is done by simply refreshing the page.

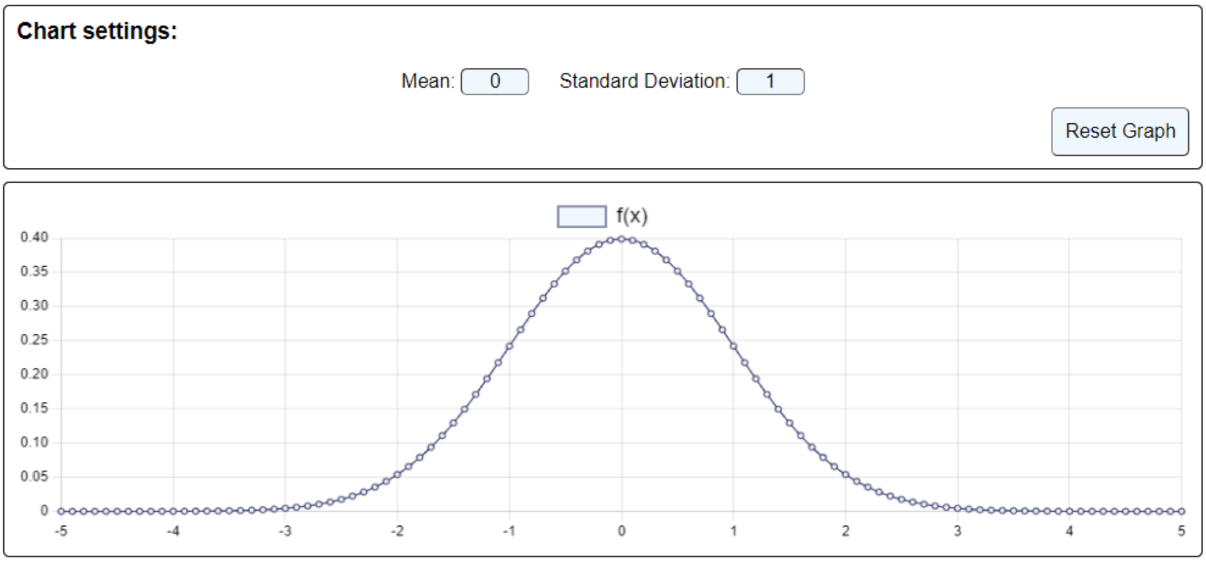


Figure 3: Added input boxes to allow user to change parameters

3.3 Probability Calculator

We now wish to add a feature which, when $X \sim N(\mu, \sigma^2)$ as defined by the user, can calculate α given x or x given α such that either $P(X \leq x) = \alpha$ or $P(X \geq x) = \alpha$, depending on user input. To change the inequality sign, the user will be able to open a drop-down menu and select the desired symbol. The parameter being calculated will be determined by which parameter the user has input.

To calculate α given x, x is first standardised using the transformation $x \mapsto \frac{x-\mu}{\sigma}$ in order to simplify the calculations. We then use the fact that, if $X \sim N(0, 1^2)$, $P(X \leq 0) = 0.5$ and can use an appropriate quadrature rule to calculate

$$P(0.5 \leq X \leq x) = \int_{0.5}^{x} p_X(t)\, dt$$

or

$$P(x \leq X \leq 0.5) = \int_{x}^{0.5} p_X(t)\, dt$$

depending on whether $x > 0.5$ or $x < 0.5$ respectively, which can then be added to or subtracted from 0.5 to obtain α. Note that we have

$$P(x \leq X \leq 0.5) = \int_{x}^{0.5} p_X(t)dt = -\int_{0.5}^{x} p_X(t)\, dt \implies \int_{0.5}^{x} p_X(t)\, dt = -[P(x \leq X \leq 0.5)]$$

therefore

$$0.5 + P(0.5 \leq X \leq x) = 0.5 + \int_{0.5}^{x} p_X(t)\, dt = 0.5 - [P(x \leq X \leq 0.5)]$$

hence we can simply calculate α by computing

$$0.5 + \int_{0.5}^{x} p_X(t)\, dt.$$

The quadrature rule used to calculate the integrals here was the composite trapezium rule [9] where a region $[a,\, b]$ is decided into n subintervals/strips with the width of each subinterval $h = \frac{b-a}{2n}$. We also define $x_j = a + j \cdot h$, then use the approximation

$$\int_{a}^{b} f(x)\, dx \approx \frac{h}{2}\left[f(a) + 2\left[\sum_{j=1}^{n-1} f(x_j)\right] + f(b)\right].$$

The composite trapezium rule was chosen for a few reasons. Firstly, it will be relatively fast and simple to calculate algorithmically with some simple pseudocode provided below.

1. Set $ans = 0$
2. Calculate array $[a = x_0, x_1, x_2, ..., b = x_n]$ // *Using* $x_j = a + j \cdot h$
3. For $i = 0$ to $i = x_{n-1}$
4. Set $ans = ans + x_i + x_i + 1$
5. EndFor

6. Set $ans = \frac{h}{2} ans$ // *ans is final Answer*

Secondly, whilst there are numerical methods which may be more accurate, such as *Simpson's rule*, our answer does not have to be calculated to a large number of decimal points (as it needs to fit in the text box!) hence the simplicity of the trapezium rule is more valuable in our case. It was found that a maximum of around 1000000 strips could be used before there was a noticeable slowdown in the calculation. For this calculation, we assume that we are calculating α such that $P(X \leq x) = \alpha$. If this is not the case (i.e. we were supposed to calculate α such that $P(X \geq x) = \alpha$), we return $1 - \alpha$ instead. Example code is shown in listing 3.

Listing 3: Calculating α given x

```
function calcProb(x){
// Extract parameters from inputs
    const form = document.getElementById("opt");
    var mean = parseFloat(form.elements["mean"].value);
    var sd = parseFloat(form.elements["sd"].value);
    var type = form.elements["type"].value;
    x = (x - mean)/sd; // Standardise x
// We start with the knowledge that P(X<=0)=0.5 for a standard
   normal distribution
    var p = 0.5;
    if (x>=0){
        p = p + integ(0, x);
    }else{
        p = p - integ(x, 0);
    }
  if (p>1){
        p=1;
    }
    if(type == "2"){
        p = 1 - p;
```

```
    }
    p = Math.round( p * 10000 ) / 10000; // Round to 4dp
    document.getElementById("prob").value = p;
}

function integ(a, b){
\\ Trapezium rule
    var p = 0;
    var strips = 1000000;
    var h = (b - a)/strips;
    for (var i = 0; i < strips; i++) {
            p = p + (h/2)*(phi(a + (i*h), 0, 1) + phi(a + ((i+1)*h), 0, 1));
        }
    return p;
}
```

To calculate x given α, we first check if we were asked to calculate α such that $P(X \geq x) = \alpha$, replacing α with $1 - \alpha$ if so, as we have

$$P(X \geq x) = \alpha \iff P(X \leq x) = 1 - P(X \geq x) = 1 - \alpha.$$

Now the problem is in the form $P(X \leq x) = \alpha$ (if it wasn't already) so we can again use the knowledge that if $X \sim N(0, 1^2)$, $P(X \leq 0) = 0.5$ in addition to the fact that $P(X \leq -4) \approx 0$ and $P(X \leq 4) \approx 1$ to deduce that our desired value of x will lie somewhere in $(-4, 0)$ if $\alpha < 0.5$ or $(0, 4)$ if $\alpha > 0.5$ (that is unless $\alpha = \pm 1$, in which case we set $x = \pm \inf$, respectively). The bisection (root-finding) method can then be used to solve $f(x) := p_X(x) - \alpha = 0$, obtaining an approximation for x. This method works by calculating the midpoint, p, of the upper and lower bounds, a and b, respectively (initially -4 and 4) then calculating $f(p)$. If $f(p)$ has the same sign as $f(a)$ the upper bound, then we overwrite a with p. Otherwise, we overwrite b with p. The method then repeats until $f(p) = 0$ and hence the root of f is p [10]. Finally, we must ensure the obtained value of x is correct for the given distribution by returning $(\sigma x) + \mu$. Example code is shown in listing 4. Note input validation has been excluded in order to keep the listing compact.

Listing 4: Calculating x given α

```
function calcVal(c){
// Special cases if probability is 1 or 0
  else if((c==1)){
    document.getElementById("val").value = "inf"
    return 0 // Stops the function prematurely
  }else if((c==0)){
    document.getElementById("val").value = "-inf"
    return 0
  }
// Extract parameters from inputs
  const form = document.getElementById("opt");
  var mean = parseFloat(form.elements["mean"].value);
  var sd = parseFloat(form.elements["sd"].value);
  var type = form.elements["type"].value;
// Replace alpha (c) with 1-alpha if greater than sign selected
  if(type == "2"){
    c = 1 - c
  }
// We start with the knowledge that P(X<=0)=0.5 for a standard
   normal distribution
  var x = 0;
  var p = 0.5;
  var y = 4*((c-0.5)/Math.abs(c-0.5));
  if(y<0){
    var temp = x;
    x = y;
```

```
    y = temp;
  }
  var last = 0;
  var z = 0;
  var cycles = 0;
  while (Math.round((p*10000)-(c*10000))!=0){ // ie while p and c are not equal when rounded to 4dp
    z = (x+y)/2
// Estimate P(X<=z)
    if (z > last){
      p = p + integ(x,z)
        }else{
      p = p - integ(z,y)
    }
// Bisection method
    if (p > c){
      y = z;
      last = y;
    }else{
      x = z;
      last = x;
    }
  }
  // Return scaled value
  z = (sd*z) + mean;
  document.getElementById("val").value = Math.round(z * 1000) / 1000; // Round to 4dp
}
```

Problems may arise if the user inputs certain values, for example $\alpha = 0.9999997119$ as first, $P(X \leq 4.999) = 0.9999997119$ so the code will essentially be trying to find 4.999 in between 0 and 4 and secondly, 4.999 is very precise meaning it may take many iterations (and hence time) to find. These issues can be resolved by restricting the number of decimal places the user can input to 4 (as the maximum possible value then gives $P(X \leq 3.75) = 0.9999$) and halting the calculation when the calculated and actual values are the equal when rounded to 4 decimal, respectively.

We will now add the ability to show the calculated probability on the graph, visually linking the probabilities with the pdf. To do this, radio buttons can be added to allow the user to show or hide the shaded area. If the "show" option is selected, the graph is redrawn with an additional, identical graph being drawn up to the appropriate value of x with shading underneath, as shown in figure 4

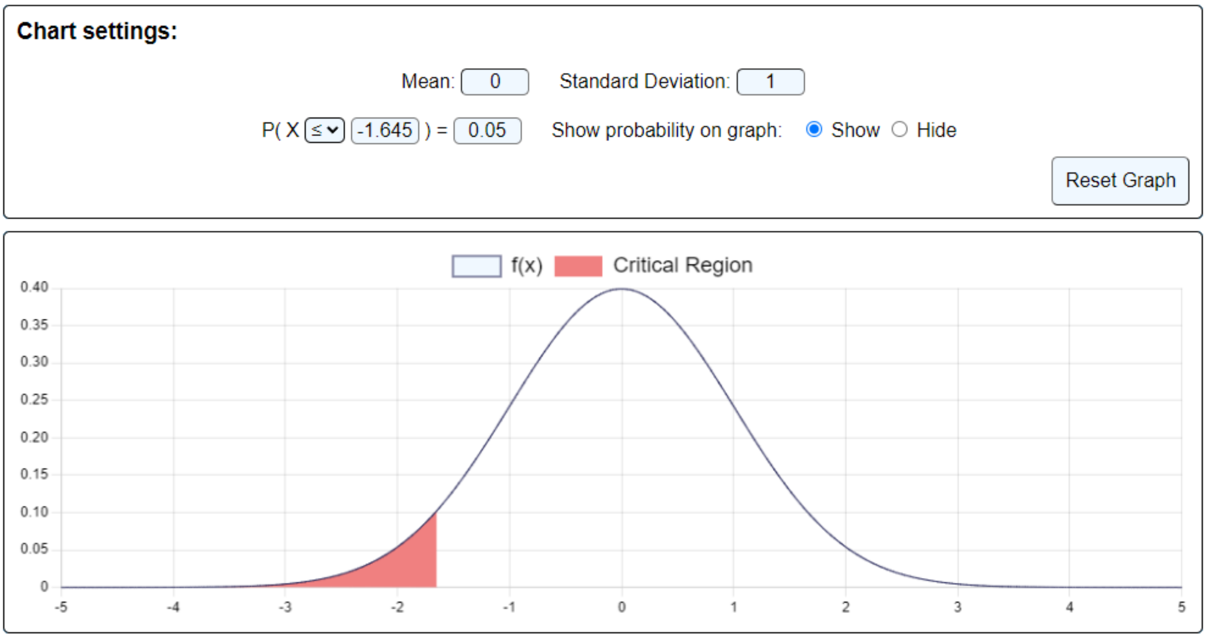
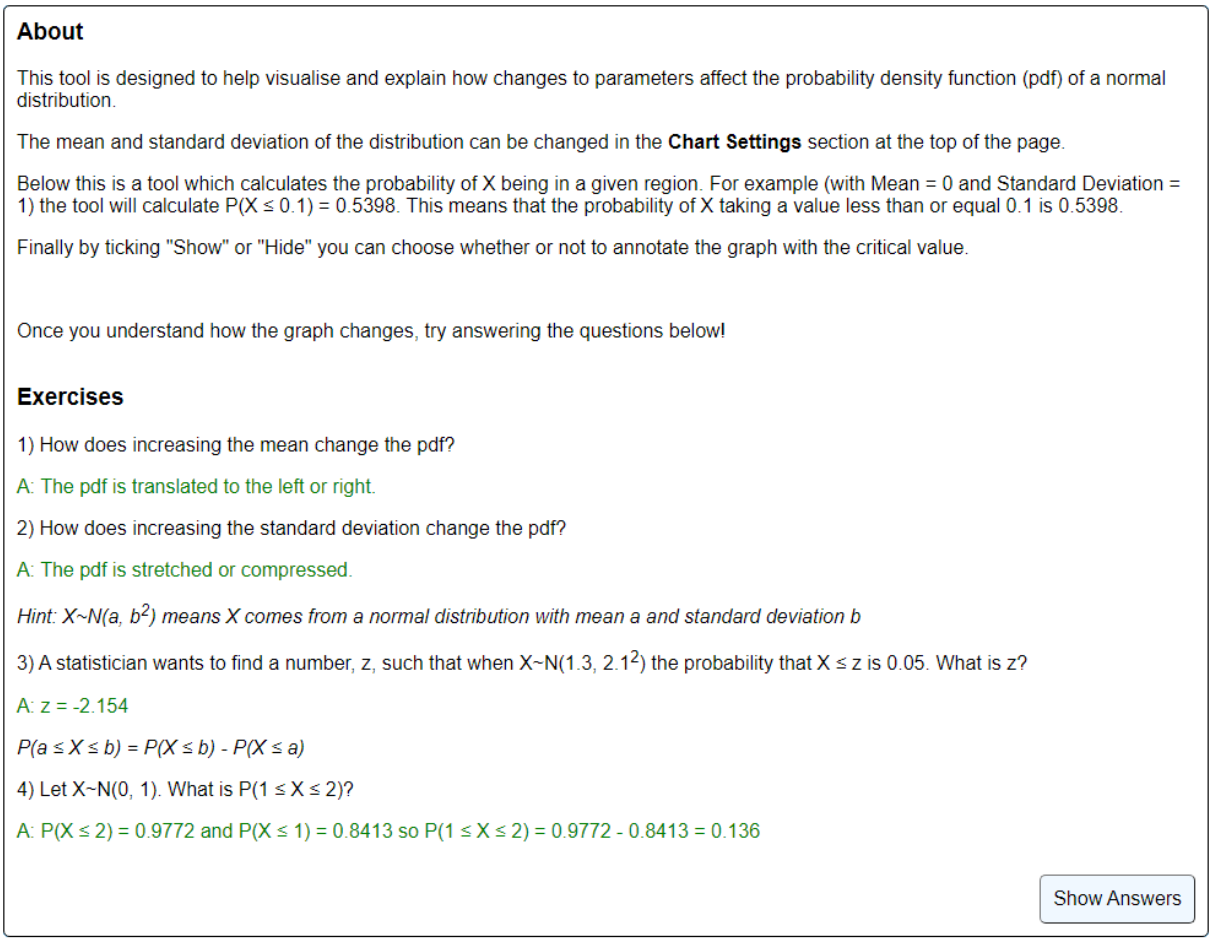


Figure 4: Added probability calculation feature and corresponding shaded region

3.4 Feedback

An explanation of how the tool works and questions aimed at A-Level students were added, as shown in figure 5, with the idea that students will be able to learn how the tool functions and then use the tool to attempt the exercises, checking their answers using the *Show Answers* button.

The tool was then shown to a science educator in Doncaster to gain feedback from an education professional. She commented that the tool was clearly laid out, easy to follow, and



About

This tool is designed to help visualise and explain how changes to parameters affect the probability density function (pdf) of a normal distribution.

The mean and standard deviation of the distribution can be changed in the **Chart Settings** section at the top of the page.

Below this is a tool which calculates the probability of X being in a given region. For example (with Mean = 0 and Standard Deviation = 1) the tool will calculate $P(X \leq 0.1) = 0.5398$. This means that the probability of X taking a value less than or equal 0.1 is 0.5398.

Finally by ticking "Show" or "Hide" you can choose whether or not to annotate the graph with the critical value.

Once you understand how the graph changes, try answering the questions below!

Exercises

1) How does increasing the mean change the pdf?

A: The pdf is translated to the left or right.

2) How does increasing the standard deviation change the pdf?

A: The pdf is stretched or compressed.

Hint: $X \sim N(a, b^2)$ means X comes from a normal distribution with mean a and standard deviation b

3) A statistician wants to find a number, z, such that when $X \sim N(1.3, 2.1^2)$ the probability that $X \leq z$ is 0.05. What is z?

A: z = -2.154

$P(a \leq X \leq b) = P(X \leq b) - P(X \leq a)$

4) Let $X \sim N(0, 1)$. What is $P(1 \leq X \leq 2)$?

A: $P(X \leq 2) = 0.9772$ and $P(X \leq 1) = 0.8413$ so $P(1 \leq X \leq 2) = 0.9772 - 0.8413 = 0.136$

Show Answers

Figure 5: About section and student exercises

simple to use. She suggested that the about section might be better being located at the top of the page so that students aren't overwhelmed at seeing the unfamiliar input boxes and settings straight away. She also suggested adding a drop-down to the about section so that students don't have to scroll back and forth between the exercises and the tool itself.

4 Activity 2: The Central Limit Theorem

4.1 Overview

The Central Limit Theorem (CLT) is another topic which many students find difficult to understand, but which lends itself quite well to a visual explanation. This tool will allow the user to select a probability distribution to be sampled from. The mean of each sample will be calculated and displayed on a histogram.

Suppose we have independent and identically distributed variables, $X_1, X_2, ...$, we can define their *partial sums*, $S_n = \sum_{i=1}^{n} X_i \quad (n = 1, 2, ...)$ It is assumed that $\mathrm{E[X_1]} = \mu$ and $\mathrm{var}(X_1) = \sigma^2$ are finite. The CLT then states that as $n \to \infty$,

$$P\left(\frac{S_n - n\mu}{\sigma\sqrt{n}} \leq z\right) \to P(Z \leq z) \tag{4.1}$$

where $Z \sim N(0, 1^2)$. If we standardise S_n, defining $Z_n = (S_n - n\mu)/\sigma\sqrt{n}$, then we have

$$P(Z_n \leq z) \to P(Z \leq z)$$

which is often written as $Z_n \xrightarrow{D} Z$. [11]

For this activity, this means that as the number of samples $n \to \infty$ our histogram of sample means will approximate a normal distribution with $\mu = \mathrm{E[X_1]}$ and $\sigma^2 = \mathrm{var}(X_1)$.

Required features of this activity:

- Ability to select from a list of distributions
- Ability to change parameters of the chosen distribution
- Ability to sample a chosen number of points from the distribution
- Mean values of previous samples shown on histogram

4.2 Basic Function and Uniform Distribution

In this section we will create the main functionality of the tool using the uniform distribution. JavaScript has an in-built function to generate samples from the uniform distribution in the form of the *Math.random()* function which returns a number generated from a U(0,1) distribution.

In most computer systems, "random" numbers are actually long sequences of numbers produced using deterministic process, but which appear to be random. Most programming languages use a process called seeding where each "seed" corresponds to a specific number sequence [12]. JavaScript is no exception [13] and while the exact method used is dependent on the implementation, the function does produce values which are approximately uniform [14].

First we draw the PDF of a uniform distribution using similar methods as we have previously. This time we will use JSXGraph [15] to create our plot as it will allow us to more easily annotate the sample points later on.

As in 3.2, we create a function which calculates the value of the uniform PDF at a given point, x, using the standard formula

$$p_X(x) = \begin{cases} \frac{1}{b-a} & \text{if } a \geq x \geq b \\ 0 & \text{otherwise.} \end{cases}$$

An example function is shown in listing 5 below.

Listing 5: Uniform PDF function

```
function uni(x,a,b){
// Check for second half of formula
  if(x>b || x<a){
    return 0
  }
// If not outside of range
  return 1/(b-a);
}
```

We then use this function to create an array of points to plot, this time using JSXGraph, as shown in figure 6.

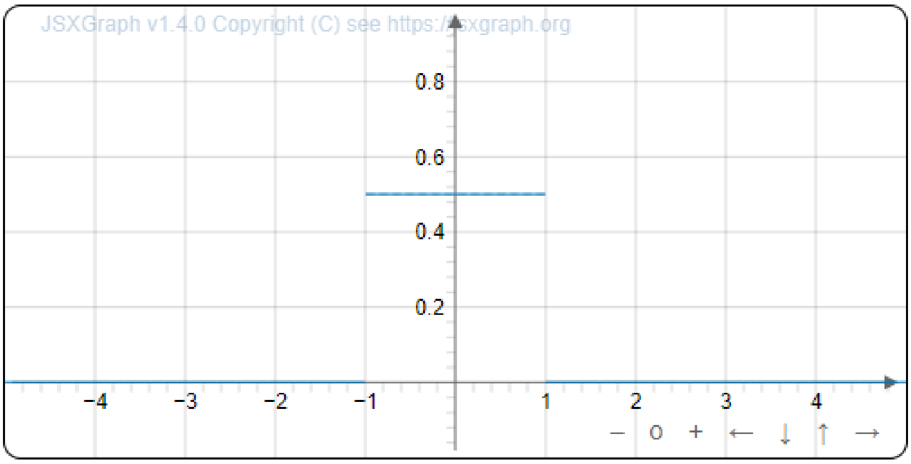


Figure 6: PDF of $U(-1, 1)$ distribution, drawn in JSXGraph

Now we can create a function which uses *Math.random()* to sample from the uniform distribution, scaling using parameters a and b (which are hard-coded for now to reduce scope for errors) as in listing 6.

Listing 6: Uniform distribution sampling function

```
function unigen(){
  var a=-1;
  var b=1;
// Use Math.random to sample from U(0,1)
  v = a + (Math.random()*(b-a));
  return v;
}
```

Each time we wish to take a sample, we can run the function the required amount of times using a *for* loop as in line 8 of listing 7.

Listing 7: Sampling options

```
1 <!-- Slider to select number of points per sample -->
2 <input type="range" value="10" min="2" max="500" id = "sslide"
3 <!-- Change label when number of points changes -->
4   oninput="document.getElementById('sPoints').innerHTML = this.
      value"/>
```

```
<!-- Label to display number of points per sample -->
<label id="sPoints">10</label> <label>points per sample</label>
<!-- Button to generate sample -->
<button type="button" onclick="for (var i = 0; i < document.getElementById('sPoints').innerHTML; i++) {unigen()};">Generate 1 Sample</button>
```

With the options in listing 7 added, our activity is now as shown in figure 7.

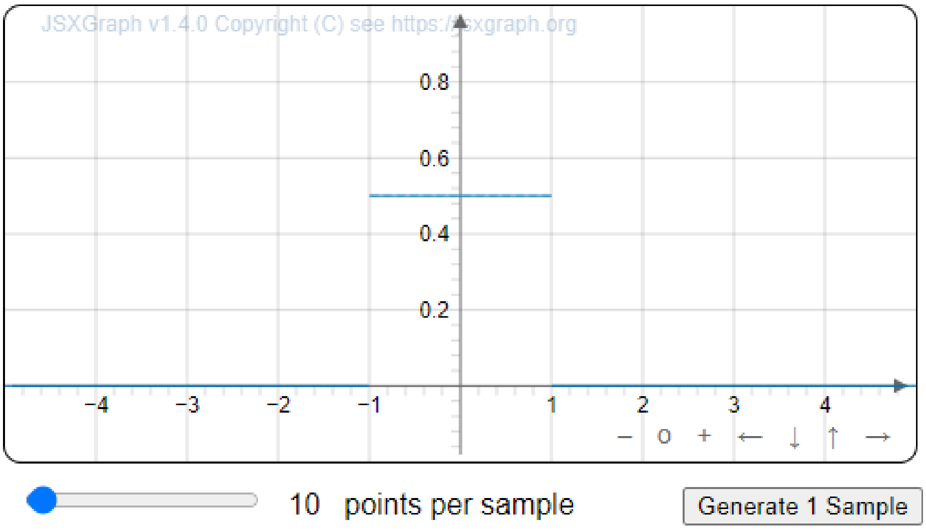


Figure 7: Added ability to sample from the distribution

At this point, our method for generating the sample is not very useful. Not only is the code messy, but we will not be able to generate from other samples later on and currently can't use the generated values as they are not stored. Therefore, we create a function which will be able to select the appropriate distribution to sample from, store the values generated, and display them on the graph. An example is shown in listing 8.

Listing 8: Sampler function

```
function sampler(size){
// Remove previous points from graph if they exist
  try{
  board.removeObject(gps)
  }finally{}
```

```
// Get the correct distribution from the heading on the page
  distribution =  document.getElementById("dist").innerHTML.toLowerCase()
// Sample appropriate number of times
  var total = 0;
  for (var i = 0; i < size; i++) {
    switch(distribution){
      case "uniform":
            a = parseFloat(document.getElementById("a").value);
            b = parseFloat(document.getElementById("b").value);
            pt = unigen(a,b);
            break;
      }
      total=total+pt // Add points to total
// Add point to graph
  apoint = board.create("point" , [pt, 0]).setLabelText("");
// Store points in an array so they can be removed before the next sample
    gps.push(apoint);
    }
    mean = total/size // Store sample mean
}
```

Now when the distribution is sampled, the points are shown on the graph, as in figure 8, and the sample mean is calculated to be plotted on the histogram.

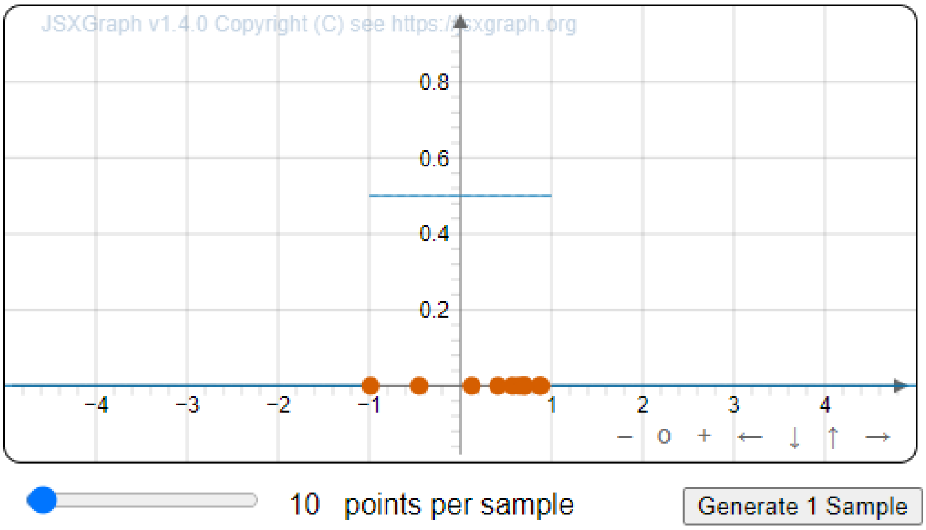


Figure 8: Sample points annotated on graph

To plot our histogram, we will use Plotly [16], redrawing the histogram after each sample. Figure 9 shows the histogram after 7500 samples. An extra slider was added to select how many samples should be taken at one time, in order to speed up the process, as well as a counter to show how many samples have been taken so far, and a reset button to allow the user to start the experiment again. The parameter inputs were also changed to call the resetting function when they are changed.

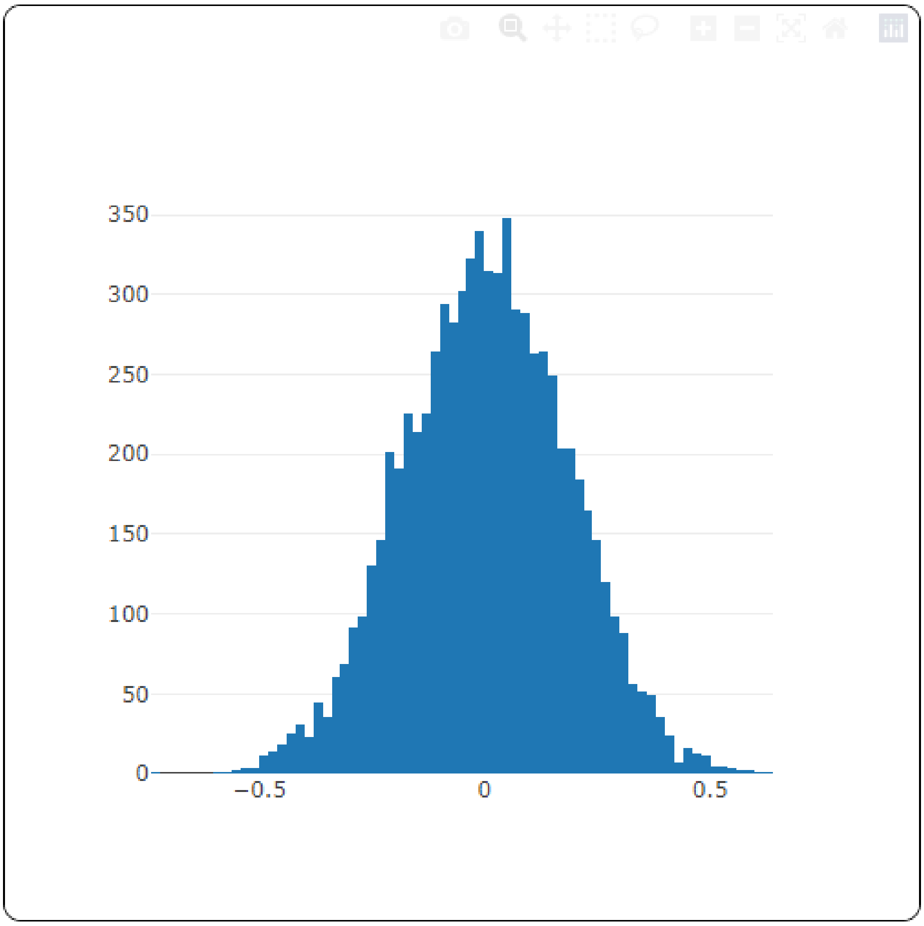


Figure 9: Histogram of sample means for 7500 samples from $U(-1, 1)$ distribution

Testing the code at this stage, the tool clearly shows how, as the number of samples increases, the histogram tends towards being a normal distribution.

4.3 Normal Distribution

Now we extend the activity to other distributions, beginning with the normal distribution. There are several methods of computing a sample from the normal distribution. We could use a method called *inverse transform sampling* [17] which involves generating a value, z, from a uniform distribution, $Z \sim U(0, 1)$. We then find the largest value of x such that when $X \sim N(\mu, \sigma^2)$, $P(X \geq x) \leq z$. In other words, we randomly choose a value, z, between 0 and 1 and return the value of x such that the probability of obtaining a lower value is exactly z. This method is theoretically relatively simple, however it would require us to use the bisection method, as in 3.3 in order to calculate x which may be computationally slow as the method will have to be performed for every value.

An alternative method is the *Box–Muller transform* [18]. In this method, two independent samples, U_1 and U_2 are taken, again from the uniform distribution $U_1, U_2 \sim U(0,1)$. These variables are then used to define $Z_0 = \sqrt{-2\log U_1}\cos(2\pi U_2)$ and $Z_1 = \sqrt{-2\log U_1}\sin(2\pi U_2)$ which are independent random variables such that $Z_0, Z_1 \sim N(0, 1^2)$. The Box–Muller transform method is also simple to implement, but has the added benefit that it is more computationally efficient than inverse transform sampling, hence we will use this method.

Listing 9 shows an example function which uses the Box-Muller transform method to generate a random value from a normal distribution.

Listing 9: Box-Muller transform

```
function stdNorm(mean, sd) {
  var u = 0
  var v = 0;
  while(u == 0){ // While loop ensures (0,1) not [0,1)
    u = Math.random();
  }
  while(v == 0){
    v = Math.random();
  }
  // Transform to make Z_0
  var val = (Math.sqrt( -2 * Math.log( u ) ) * Math.cos( 2 * Math.PI * v ));
  // Return value, scaled by mean and sd
  return (sd*val) + mean;
}
```

The sampler function in listing 8 was also updated, adding a case for if the normal distribution was selected. With added labels, the tool was as shown in figure 10.

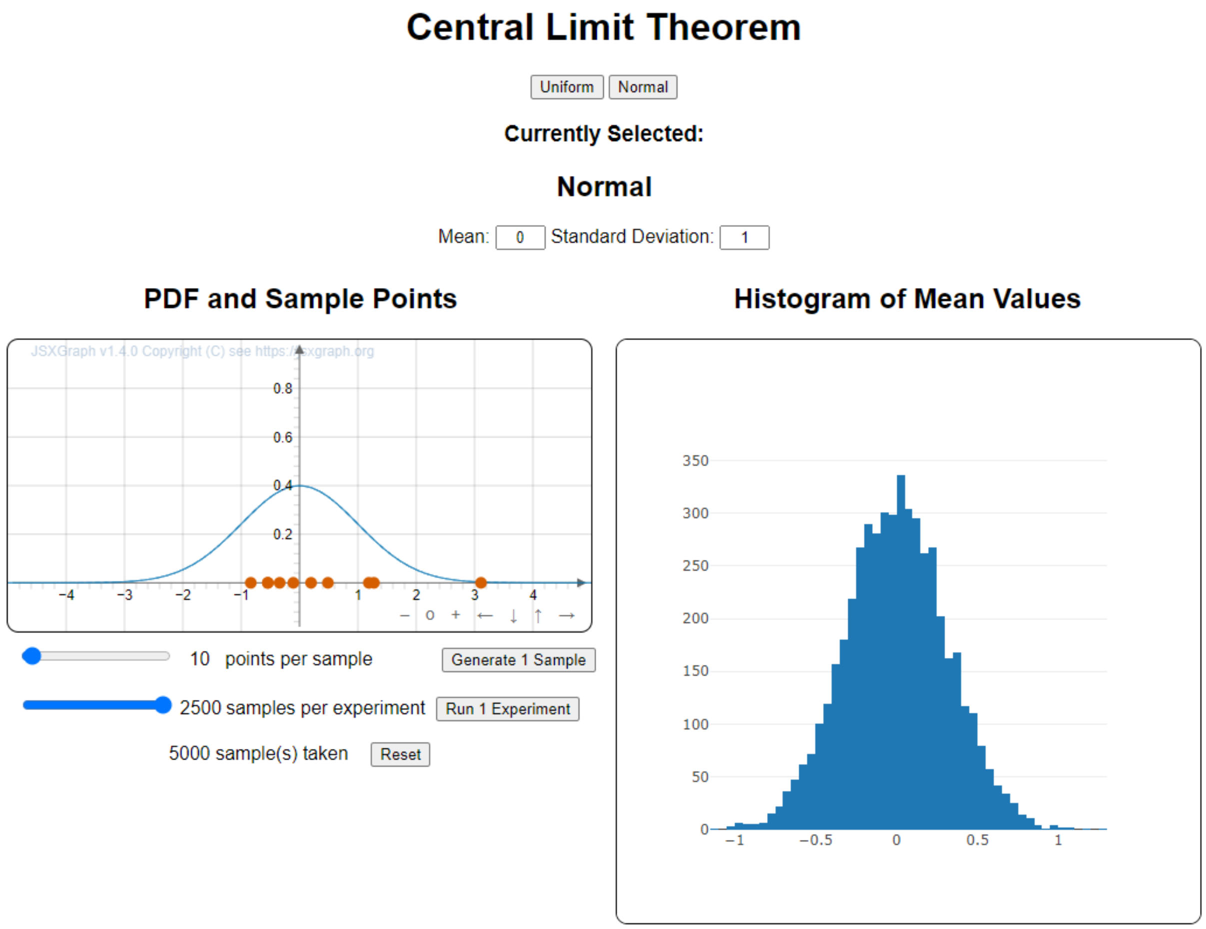


Figure 10: Activity with normal distribution ($\mu = 0$, $\sigma = 1$) selected and 5000 samples taken of size 10

4.4 Exponential Distribution

Now we wish to add the exponential distribution to our activity. As previous, we begin by drawing the PDF, first creating a function which calculates the value of the exponential PDF at a point, x, using the formula

$$p_X(x) = \begin{cases} \lambda e^{-\lambda x} & \text{if } x > 0 \\ 0 & \text{otherwise.} \end{cases}$$

An example function is shown in listing 10.

Listing 10: Exponential PDF

```
function exp(x, lam){
  if(x>0){
```

```
    return lam*Math.exp(-lam*x);
  }else{
    return 0;
  }
}
```

The function was then used to draw the PDF in JSXGraph, as shown in figure 11.

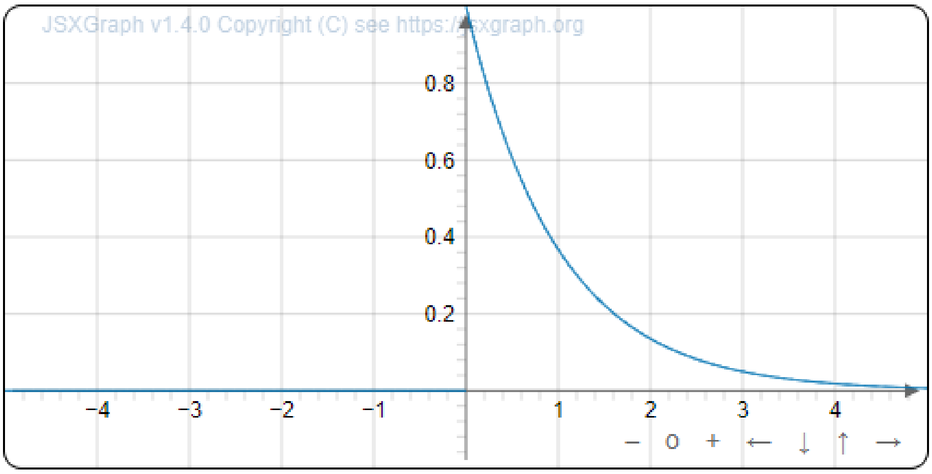


Figure 11: Exponential PDF drawin in JSXGraph

To generate samples from the distribution, we could use inverse transform sampling as mentioned in 4.3, however now that we have the ability to sample from normal distributions, we can instead use the *Metropolis-Hastings algorithm* (MHA). The MHA is an algorithm used to sample from a distribution with pdf $\pi(x)$. Beginning at a value, x, it uses a proposal distribution, $q(x, y) = q(y|x)$ to suggest a value, y to move to. The move is accepted according to *Metropolis-Hastings acceptance probability* given by

$$\alpha = \min\left(1, \frac{\pi(y)q(y, x)}{\pi(x)q(x, y)}\right). \tag{4.2}$$

This creates a Markov chain where we set

$$X_{t+1} = \begin{cases} y & \text{with probability } \alpha(x, y) \\ x & \text{with probability } 1 - \alpha(x, y). \end{cases}$$

The final value of the chain is then our sample value. Whilst the algorithm works for any arbitrary $q(x, y)$, a common choice is $q(x, y) \sim N(x, \sigma^2)$. In this case the algorithm is known as known as a *Gaussian Random Walk*. As $q(x, y) = q(y, x)$, equation 4.2 becomes

$$\alpha = \min\left(1, \frac{\pi(y)}{\pi(x)}\right).$$

We can in theory choose any value of σ, but we should be careful not to choose a value which is too large or too small to avoid making jumps which are too big or too small, respectively. [19]

An example of the MHA is given in listing 11. Initally, $\sigma = 1$ was chosen, arbitrarily. It was found that around 1000 moves was the maximum that can be done without the tool being too slow.

Listing 11: Metropolis-Hastings Algorithm for exponential distribution

```
var x = 1;
var y = 1;
var alpha = 0;
var p=0;
for (var i = 0; i < 1000; i++) {
  y = stdNorm(x, 1) // Propose y
  alpha = Math.min(1, (exp(y, lam))/(exp(x, lam))) // Calculate
     MH acceptance probability
  p = Math.random();
  if(p<alpha){ // Accept with probability alpha
    x = y;
    }
  }
  return x;
}
```

To decide on a value of σ, the algorithm was rewritten in *R* [20], allowing us to assess how well our choice of σ performs. The two metrics used to evaluate the performance were *trace plots* and *histograms*. Trace plots show the value of X_t at each time, t. If our choice of

σ is a good one, the trace plot should "settle down" and reach equilibrium. The histograms compare the frequency of the sample values, X_t against the exponential PDF using the inbuilt *dnorm* function in R.

As mentioned, $\sigma = 1$ was chosen initially and, with $\lambda = 1$, the trace plot (figure 12) reaches equilibrium quickly.

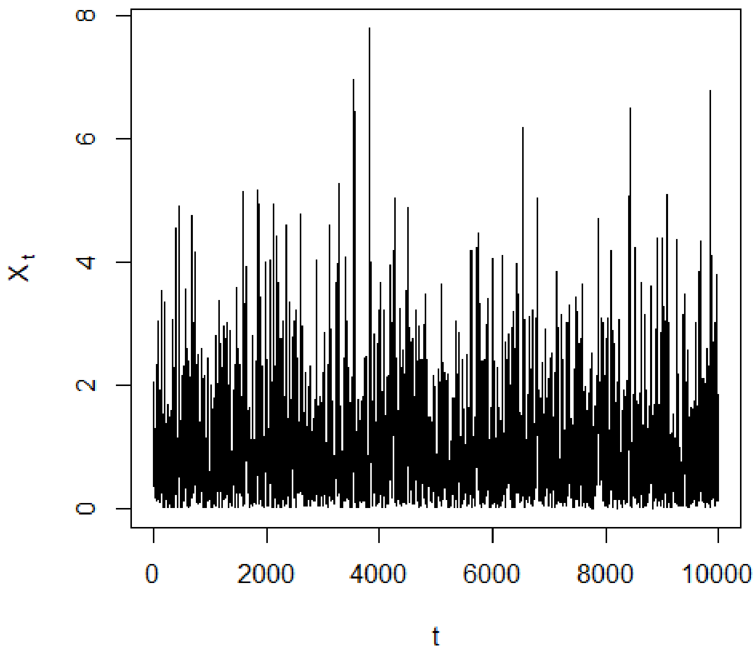


Figure 12: Trace plot with $\sigma = 1$, $\lambda = 1$

However, when we change λ to a more extreme value such as $\lambda = 0.05$ (figure 13), the trace plot does not reach equilibrium even after 10000 moves.

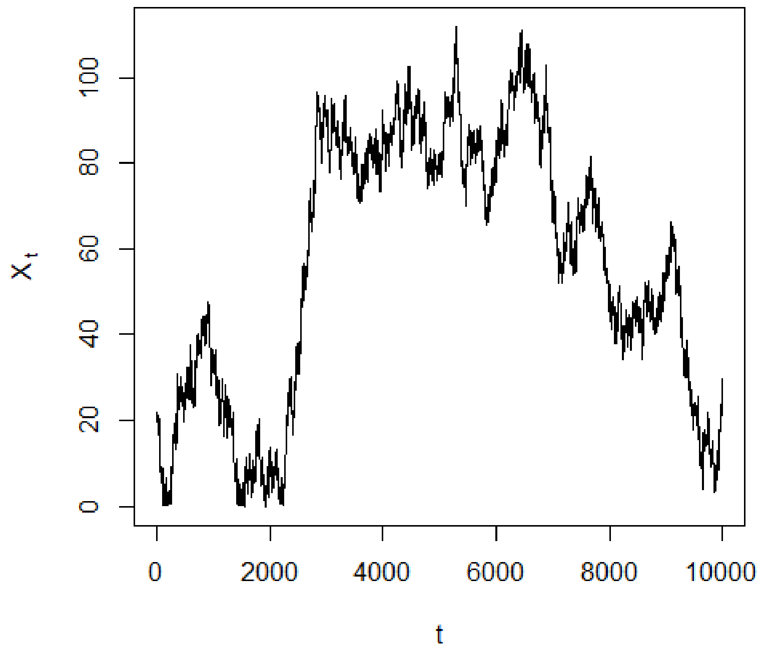


Figure 13: Trace plot with $\sigma = 1$, $\lambda = 0.05$

This shows that σ cannot simply be a constant as the user can input any $\lambda > 0$. After experimenting with a few other constants, it was observed that higher values of σ performed well with lower values of λ, and vice versa. Therefore we want σ to be a decreasing function of λ with $\sigma(\lambda = 1) = 1$ (as $\sigma = 1$ worked well when $\lambda = 1$). Therefore $\sigma = 1/\lambda$ was tested next. As can be seen in figure 14, this value of σ performed well for $\lambda = 0.05$, reaching equilibrium quickly.

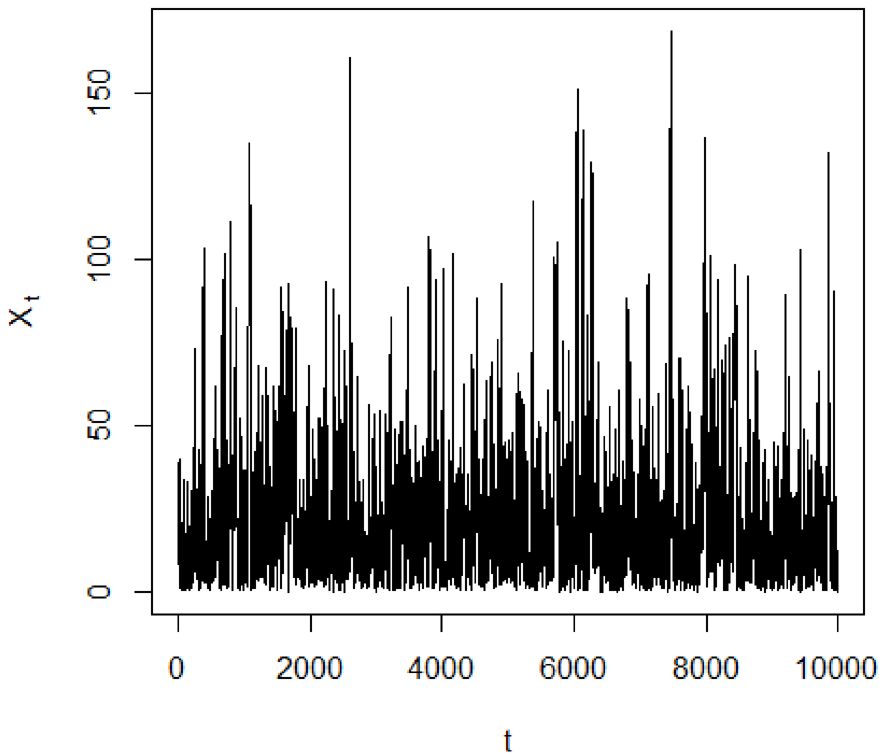


Figure 14: Trace plot with $\sigma = 1/\lambda$, $\lambda = 0.05$

Testing with extreme high values, such as $\lambda = 500$, also gave good results as can be seen in figure 15.

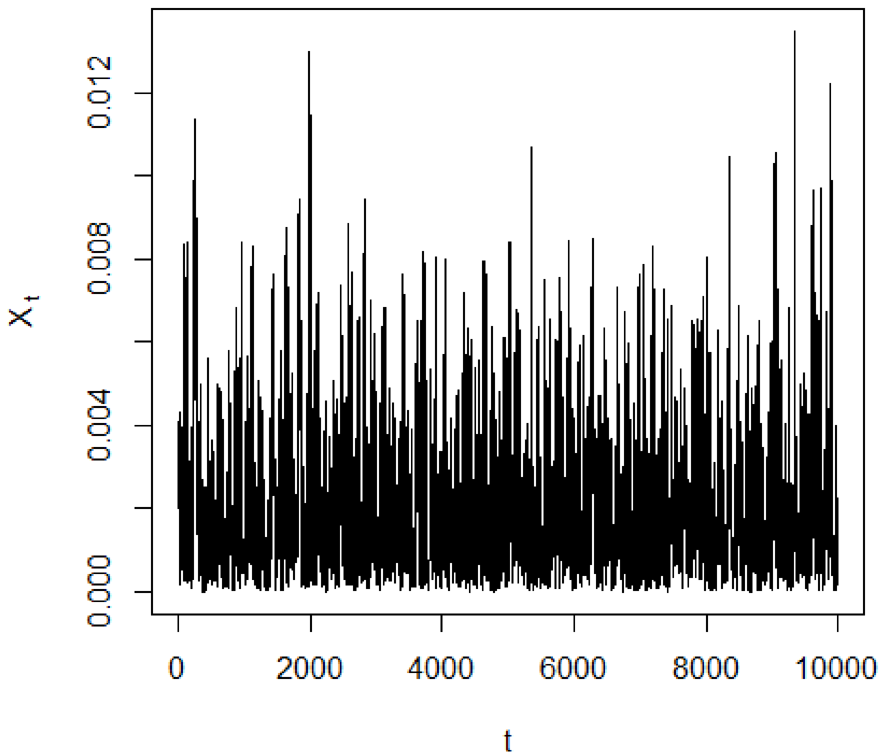


Figure 15: Trace plot with $\sigma = 1/\lambda$, $\lambda = 500$

Now that a value of σ was found that reaches equilibrium quickly, the sample values were compared against the true values using *dnorm*. As can be seen in figures 16, 17, and 18, the sample values matched closely with the true values for $\sigma = 1$ as well as for more extreme values $\sigma = 0.05,\, 500$.

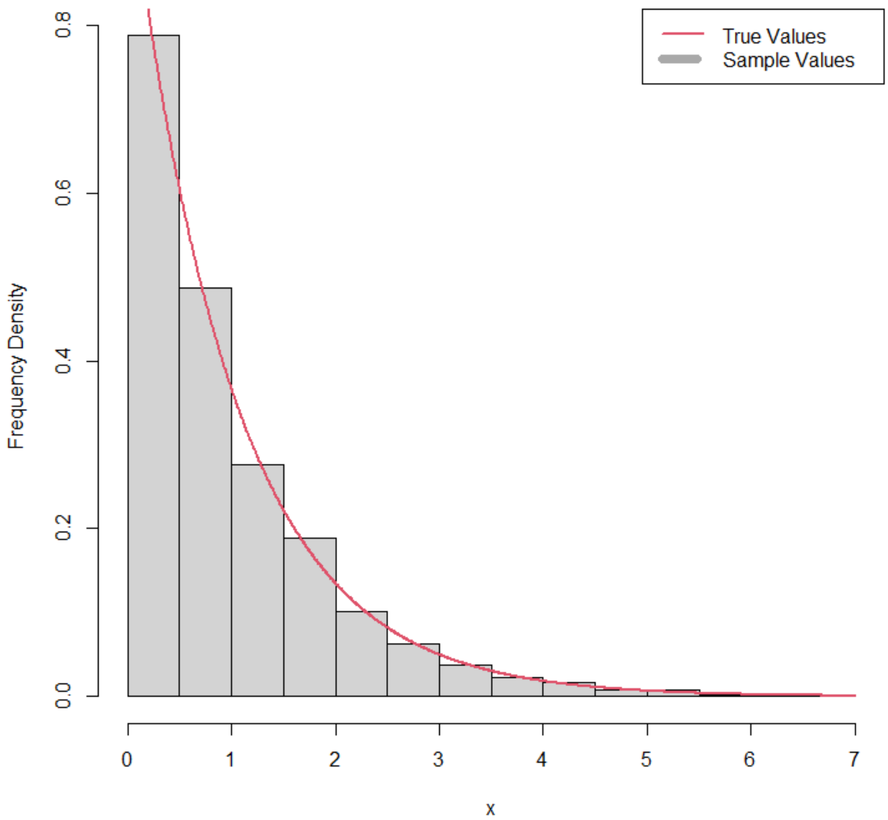


Figure 16: Histogram with $\sigma = 1/\lambda$, $\lambda = 1$

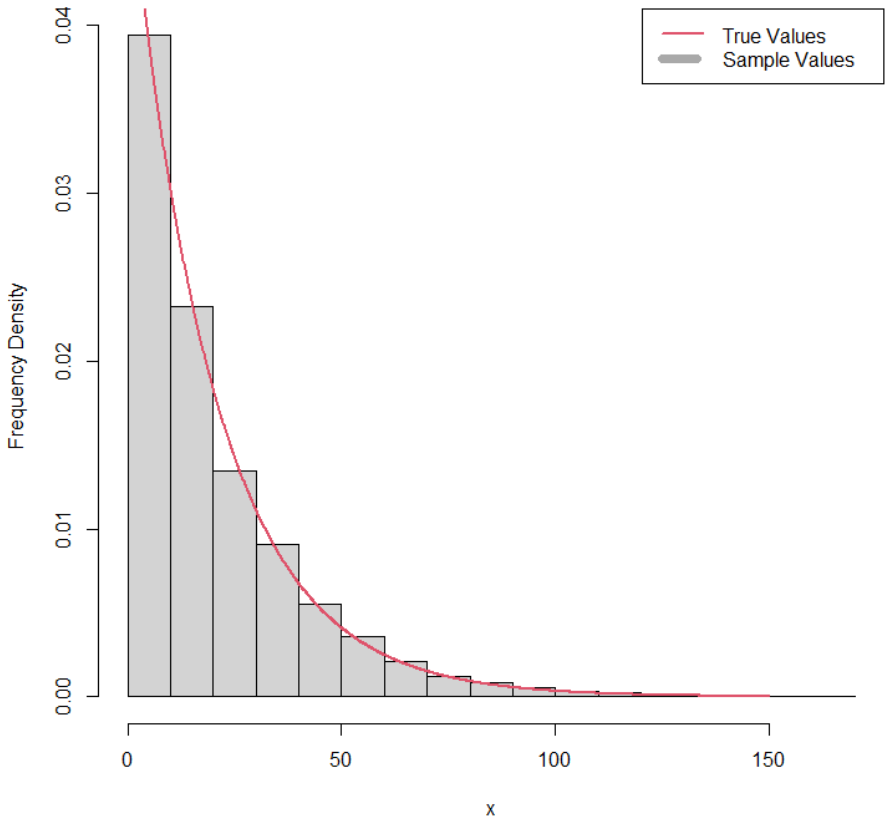


Figure 17: Histogram with $\sigma = 1/\lambda$, $\lambda = 0.05$

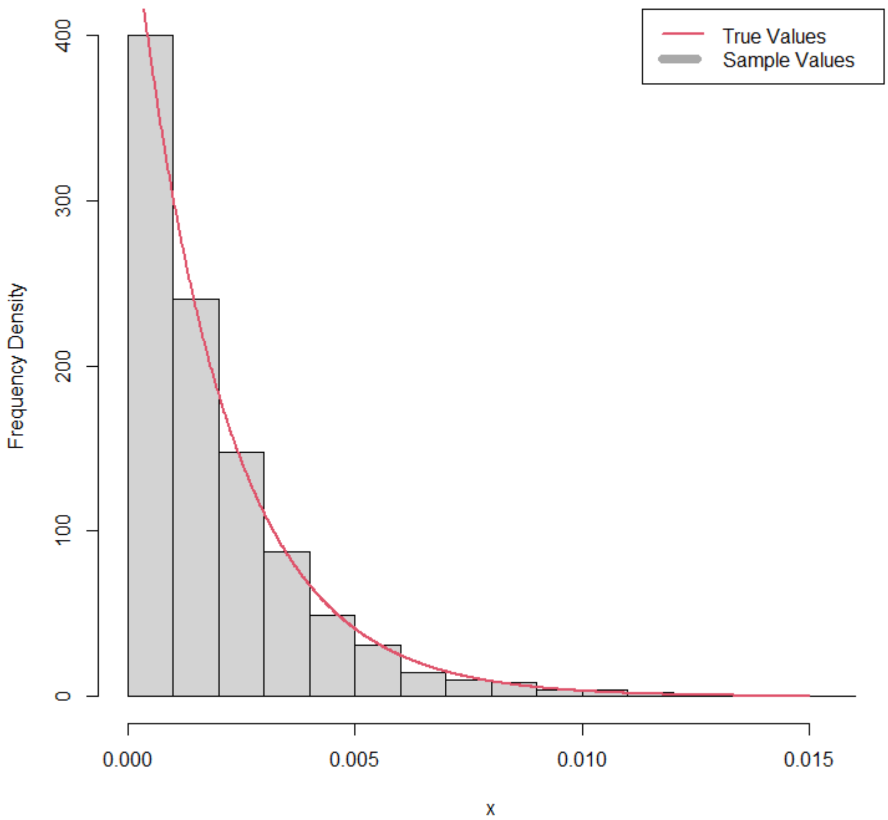


Figure 18: Histogram with $\sigma = 1/\lambda$, $\lambda = 500$

Therefore, it was decided to use $\sigma = 1/\lambda$ in this implementation. Listing 11 was updated to use this value of σ, and after updating the sampler function in listing 8 to add the case for the exponential distribution and adding required labels and inputs, the tool was as shown in figure 19

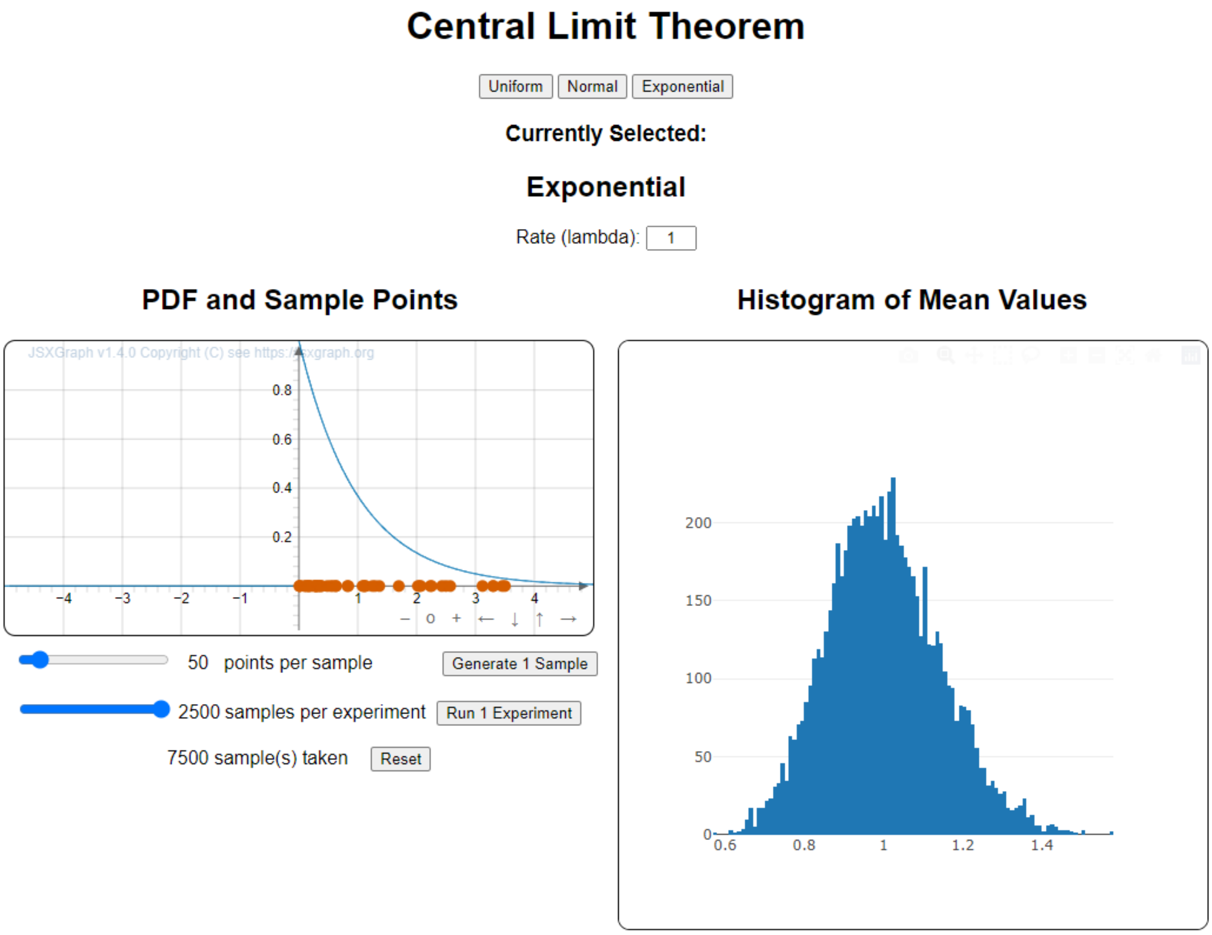


Figure 19: Activity with exponential distribution selected ($\lambda = 1$) and 7500 samples taken of size 50

4.5 Feedback

As in 3.4, the tool was shown to a science educator in Doncaster to gain feedback.

She commented that this tool was simple to use and follow once she understood how to use it, but suggested adding a small about section, like in the tool from section 3. She also commented that the tool was effective as her understanding of the central limit theorem was relatively weak before using the tool, but said understood the basics afterwards.

5 Conclusions

The two activities documented in this report make use of technology as a valuable learning tool, allowing students to explore concepts in new ways. The activities combine different methods of learning to give students a greater understanding of the relevant topic.

Further work on this project would include improving the activities created here by acting upon the feedback that was given in sections 3.4 and 4.5, trialling the activities in a school environment to explore how students respond, and exploring the endless possibilities that these technologies provide by creating additional activities.

A Code for section 3

Listing 12: HTML code for section 3

```html
<!DOCTYPE html>
<html lang="en">
    <head>
        <meta charset="UTF-8" />
        <title>Normal Distribution</title>
        <link rel="stylesheet" type="text/css" href="index.css"/>
    </head>
    <body>
        <script src="https://cdn.jsdelivr.net/npm/chart.js@3.5.1/dist/chart.min.js"></script>
        <main>
            <h1>Normal Distribution</h1>
            <div id=chartSettings>
                <h3>Chart settings:</h3>

                <form id="opt">
                    <div id="inputs" style="text-align: center">
                        <span class="sett">
                            Mean:
                            <input oninput="calcProb(getElementById('val').value), createGraph()" type="text" name="mean" value="0" >
                        </span>
                        <span class="sett">
                            Standard Deviation:
                            <input oninput="calcProb(getElementById('val').value),
```

```
createGraph()" type="text" name="sd
" value="1">
</span>
<br><br>

<span class="sett">
    <span id="probability">
        P( X
        <select onchange="calcProb(
            getElementById('val').value)"
            id="type">
            <option value="1">&leq;
                </option>
            <option value="2">&geq;
                </option>
        </select>
        <input oninput="calcProb(
            getElementById('val').value)"
            id = "val" type="text" name="
            tail" value="0"/>
        ) = <input onchange="calcVal(
            getElementById('prob').value)"
            id = "prob" type="text" name="
            value" value="0.5"/>
    </span>
</span>

Show probability on graph:
<span onchange="calcProb(getElementById('
    val').value)" class="sett">
    <input class = "radio" type="radio"
```

```
                        id="show" name="sig" value="show">
                    <label for="show">Show</label>
                    <input class = "radio" type="radio"
                        id="hide" name="sig" value="hide"
                        checked="checked">
                    <label for="hide">Hide</label>
                </span>
            </div>

            <div id="buttons" style="text-align: right">
                <button type="button" onclick="window.location.reload()">Reset Graph</button>
            </div>
        </form>
    </div>

    <div id="canvas">
        <canvas id="myChart" height="100%"></canvas>
    </div>

    <div id="expl">
        <h3>About</h3>
        <p>This tool is designed to help visualise and explain how changes to parameters affect the
            probability density function (pdf) of a normal distribution.</p>

        <p>The mean and standard deviation of the distribution can be changed in the <b>Chart Settings</b>
            section at the top of the page.</p>
```

```html

            <p>Below this is a tool which calculates the probability of X being in a given region.
             For example (with Mean = 0 and Standard Deviation = 1) the tool will calculate P(X &leq; 0.1) = 0.5398.
             This means that the probability of X taking a value less than or equal 0.1 is 0.5398.</p>

            <p>Finally by ticking "Show" or "Hide" you can choose whether or not to annotate the graph with the critical value.</p>

            <br>

            <p>Once you understand how the graph changes, try answering the questions below!</p>

            <br>
            <h3>Exercises</h3>
            <p>1) How does increasing the mean change the pdf?</p>
            <p id="answer1" class="answer"> </p>
            <p>2) How does increasing the standard deviation change the pdf?</p>
            <p id="answer2" class="answer"> </p>
            <p><i>Hint: X~N(a, b<sup>2</sup>) means X comes from a normal distribution with mean a and standard
             deviation b</i></p>
```

```
            <p>3) A statistician wants to find a number, z, such that when X~N(1.3, 2.1<sup>2</sup>) the
                probability that X &leq; z is 0.05. What is z?
            </p>
            <p id="answer3" class="answer"> </p>
            <p><i>P(a &leq; X &leq; b) = P(X &leq; b) - P(X &leq; a)</i></p>
            <p>4) Let X~N(0, 1). What is P(1 &leq; X &leq; 2)?
            </p>
            <p id="answer4" class="answer"> </p>

            <div id="buttons" style="text-align: right">
                <button type="button" onclick="showAnswers()">Show Answers</button>
            </div>
        </div>

        <script>
            Chart.defaults.color = "#000"; // Set chart text to black

            function phi(z, mean, sd){
                var o2pi = 1/(Math.sqrt(2*Math.PI)*sd); // Calculates 1/sqrt(2*pi)
                var ez = Math.pow(Math.E,(-(Math.pow((z - mean)/sd, 2))/2)); // Calculates e^(-(z^2)/2)
                return ((o2pi * ez));  // Multiply them together to caclulate pdf
```

```
        }

        function createGraph(){
            const form = document.getElementById("opt");
            var mean = parseFloat(form.elements["mean"].value);
            var sd = parseFloat(form.elements["sd"].value);
            var hide = document.getElementById("hide").checked;
            var val = parseFloat(document.getElementById("val").value);
            var type = form.elements["type"].value;

            var data1 = [];
            var data2 = [];
            var labels1 = [];
            var bound = 5000;

            // Form validation
            if (sd < 0){
                alert("Standard deviation must be greater than 0");
                form.elements["sd"].value = "1";
                sd = 1;
            }

            if (Math.abs(mean) > 5){
                alert("Mean must be between 5 and -5");
                form.elements["mean"].value = "0";
                mean = 0;
```

```
                    }

                    var theData = []

                    if (hide==true){
                        for (var i = -bound; i <= bound; i+=100) {
                            var t = i/1000;
                            data1[i+bound] = phi(t, mean, sd);
                            labels1[i+bound] = t;
                        }

                        theData = [{
                        type: "line",
                            label: "f(x)",
                            data: data1,
                            pointRadius: 2.5,
                            fill: false,
                            borderColor: "rgb(25,26,79)",
                            backgroundColor: "rgb(240,248,255)",
                            tension: 0.4,
                            spanGaps: true,
                            borderWidth: 1
                        }]
                        console.log((data1))
                    }

                    if (hide == false){
                        if(type=="1"){
                            for (var i = -bound; i <= val*1000; i+=50) {
```

```
                var t = i/1000;
                data2[i+bound] = phi(t, mean, sd);
                labels1[i+bound] = t;
            }
        }else{
            for (var i = val*1000; i <= bound; i+=50) {
                var t = i/1000;
                data2[i+bound] = phi(t, mean, sd);
                labels1[i+bound] = t;
            }
        }



        for (var i = -bound; i <= bound; i+=50) {
            var t = i/1000;
            data1[i+bound] = phi(t, mean, sd);
            labels1[i+bound] = t;
        }

        theData = [{
        type: "line",
            label: "f(x)",
            data: data1,
            pointRadius: 0,
            fill: false,
            borderColor: "rgb(25,26,79)",
            backgroundColor: "rgb(240,248,255)",
```

```
                    tension: 0.4,
                    spanGaps: true,
                    borderWidth: 1
                },{
                // Data for critical region
                type: "line",
                    label: "Critical Region",
                    data: data2,
                    pointRadius: 0,
                    fill: true,
                    borderColor: "rgb(25,26,79, 0)",
                    backgroundColor: "rgb(240,128,128)",
                    tension: 0.4,
                    spanGaps: true,
                    borderWidth: 1
                }]

            }


            var chartStatus = Chart.getChart("myChart");
            if (chartStatus != undefined) {
                chartStatus.destroy(); // Destroy old
                    graph
            }
            var ctx = document.getElementById("myChart").
              getContext("2d");

            var myChart = new Chart(ctx, {
            data: {
                labels: labels1,
```

```
                datasets: theData
            },
            options: {
                scales: {

                    x: {
                        type: "linear",
                    },

                    y: {
                        type: "linear",
                        beginAtZero: true,
                    },
                },

                plugins: {
                    legend: {
                        labels: {
                            font: {
                                size: 18,
                                family: "Arial, Helvetica
                                   , sans-serif"
                            }
                        }
                    },
                }
            }
        })
    }

    // calcVal calculates the critical value x such that
```

```
            P(X <= x) = c for a given c
        function calcVal(c){
            if((c>1 || c<0)){
                window.alert("Quantile must be between 0 and
                    1");
                document.getElementById("val").value = "";
                document.getElementById("prob").value = "";
                return 0;
            }

            if(!Number.isInteger(c*10000)){
                window.alert("Please input a maximum of 4
                    decimal places");
                document.getElementById("prob").value = Math.
                    floor(c*10000)/10000;
                return 0;
            }

            // Special cases if probability is 1 or 0
            else if((c==1)){
                document.getElementById("val").value = "inf"
                return 0 // Stops the function prematurely
            }else if((c==0)){
                document.getElementById("val").value = "-inf"
                return 0
            }
        // Extract parameters from inputs
            const form = document.getElementById("opt");
            var mean = parseFloat(form.elements["mean"].value
                );
            var sd = parseFloat(form.elements["sd"].value);
```

```
            var type = form.elements["type"].value;
        // Replace alpha (c) with 1-alpha if greater than
           sign selected
            if(type == "2"){
                c = 1 - c
            }
        // We start with the knowledge that P(X<=0)=0.5 for a
            standard normal distribution
            var x = 0;
            var p = 0.5;
            var y = 4*((c-0.5)/Math.abs(c-0.5));
            if(y<0){
                var temp = x;
                x = y;
                y = temp;
            }
            var last = 0;
            var z = 0;
            var cycles = 0;
            while (Math.round((p*10000)-(c*10000))!=0){ // ie
                while p and c are not equal when rounded to 4
              dp
                z = (x+y)/2
        // Estimate P(X<=z)
                if (z > last){
                    p = p + integ(x,z)
                }else{
                    p = p - integ(z,y)
                }
        // Bisection method
                if (p > c){
```

```
            y = z;
            last = y;
        }else{
            x = z;
            last = x;
        }
    }
    // Return scaled value
    z = (sd*z) + mean;
    document.getElementById("val").value = Math.round(z * 1000) / 1000; // Round to 4dp
    createGraph()
}
// calcProb calculates the probability that X <= x for a given x
function calcProb(x){

    const form = document.getElementById("opt");
    var mean = parseFloat(form.elements["mean"].value);
    var sd = parseFloat(form.elements["sd"].value);
    var type = form.elements["type"].value;

    x = (x - mean)/sd;
    // We start with the knowledge that P(X<=0)=0.5 for a standard normal distribution
    var p = 0.5;

    if (x>=0){
        p = p + integ(0, x);
    }
```

```
        else{
            p = p - integ(x, 0);
        }

        if (p>1){
            p=1;
        }

        if(type == "2"){
            p = 1 - p;
        }

        p = Math.round( p * 10000 ) / 10000;

        document.getElementById("prob").value = p;
        createGraph();
    }

    function integ(a, b){
        var p = 0;
        var strips = 1000000;
        var h = (b - a)/strips;
        for (var i = 0; i < strips; i++) {
                p = p + (h/2)*(phi(a + (i*h), 0, 1) + phi
                    (a + ((i+1)*h), 0, 1));
            }
        return p;
    }

    function showAnswers(){
```

```
                var a1 = document.getElementById("answer1");
                a1.innerText = "A: The pdf is translated to the left or right.";

                var a2 = document.getElementById("answer2");
                a2.innerText = "A: The pdf is stretched or compressed.";

                var a3 = document.getElementById("answer3");
                a3.innerText = "A: z = -2.154";

                var a4 = document.getElementById("answer4");
                a4.innerText = "A:  P(X <= 2) = 0.9772 and  P(X <= 1) = 0.8413 so P(1 <= X <= 2) = 0.9772 - 0.8413 = 0.136";
            }

            createGraph();
        </script>
    </main>
  </body>
</html>
```

Listing 13: CSS code for section 3

```
body{
    background-color: lightskyblue;
    font-family: Arial, Helvetica, sans-serif;
    padding: 10px 8px;

}

input{
    width:50px;
    font-size: 16px;
}

h1{
    color: black;
    text-align: center;
}

h3{
    margin-top: 0;
}

input{
    border-width: 1px;
    border-radius: 5px;
    width: 50px;
    text-align: center;
    background-color: aliceblue;
}

select{
```

```
    border: solid;
    border-width: 1px;
    border-radius: 5px;
    width: auto;
    font-size: 16px;
    text-align: center;
    background: aliceblue;
}

button{
    background-color: aliceblue;
    border-width: 1px;
    border-radius: 5px;
    padding: 10px;
    text-align: center;
    text-decoration: none;
    display: inline-block;
    font-size: 16px;
    transition-duration: 0.1s;
}

button:hover{
    background-color: lightskyblue;
}

button:active{
    transform: translateY(3px);
}


#buttons{
```

```
    padding-top: 10px;
}

#chartSettings{
    padding: 10px;
    margin: 10px 10%;
    background-color: white;
    border-style: solid;
    border-color: black;
    border-width: 1px;
    border-radius: 5px;
}

#canvas{
    height: fit-content;
    margin: 10px 10%;
    padding: 10px auto;
    background: white;
    border-style: solid;
    border-color: black;
    border-width: 1px;
    border-radius: 5px;
}

#expl{
    padding: 10px;
    margin: 10px 10%;
    background-color: white;
    border-style: solid;
    border-color: black;
    border-width: 1px;
```

```
    border-radius: 5px;
}

.sett{
    width: 100px;
    padding: 8px 10px;
    text-align: right;
}

#myChart{
    font-size: 16px;
    padding: 10px;
}

#probability{

    padding: 1px 10px;
    border-style: none;
    border-width: 1px;
    border-radius: 5px;
    width: 50px;
    text-align: center;
    background-color: white;
}

#val{
    border-style: solid;
    border-width: 1px;
    border-radius: 5px;
    width: 50px;
    text-align: center;
```

```
    background-color: aliceblue;
}

#prob{
    border-style: solid;
    border-width: 1px;
    border-radius: 5px;
    width: 50px;
    text-align: center;
    background-color: aliceblue;
}

.radio{
    padding: 0px;
    width: auto;
}

.answer{
    color: green;
}

header{
    width: 100%;
    padding: 0px;
    margin: 0px;
}

footer{
    text-align: center;
    padding: 0px;
}
```

B Code for section 4

Listing 14: HTML code for section 4

```
<!DOCTYPE html>
<html lang="en">
   <head>
       <meta charset="UTF-8" />
       <title>Central Limit theorem</title>
       <script type="text/javascript" charset="UTF-8" src="https://cdn.jsdelivr.net/npm/jsxgraph/distrib/jsxgraphcore.js"></script>
       <link rel="stylesheet" type="text/css" href="https://cdn.jsdelivr.net/npm/jsxgraph/distrib/jsxgraph.css" />
       <script src='https://cdn.plot.ly/plotly-2.6.3.min.js'></script>

   </head>

   <body style="font-family: Arial, Helvetica, sans-serif;">
    <header style="text-align: center;">
        <h1>Central Limit Theorem</h1>
    </header>

    <main style="text-align: center;">
        <div>
            <div>
                <button type="button" onclick="uniSelector()">Uniform</button>
                <button type="button" onclick="normSelector()">Normal</button>
                <button type="button" onclick="expSelector()"
```

```
                  >Exponential</button>
            </div>

            <div>
                <h3>Currently Selected:</h3>
                <h2 id="dist"></h2>
            </div>
                <div id="params">
                </div>
            <br>
        </div>

        <div style="display: flex; text-align: center; justify-content: center;">
            <h2 style="width:500px; margin:10px;">PDF and Sample Points</h2>
            <h2 style="width:500px; margin:10px;">Histogram of Mean Values</h2>
        </div>

        <div style="display: flex; text-align: center; justify-content: center;">
            <span>
                <div id="box1" class="jxgbox" style="width:500px; height:250px; margin:10px; border-color: black;"></div>
                <form>
                    <span>
                        <input type="range" value="50" min="2" max="500" id = "sslide"
                        style="margin-left: 15px;"
```

```
                    oninput="document.getElementById('sPoints
                      ').innerHTML = this.value"
                    onChange="plotter()"/>
                    <label id="sPoints" style="display:
                      inline-block; width: 35px;">50</label>
                      <label style="display: inline-block;
                      padding-right: 55px;">points per sample
                      </label>
                </span>
                <button type="button" onclick="sampler(1);"
                   >Generate 1 Sample</button>
                <br>
                <br>
                <span>
                    <input type="range" value="10" min="2"
                      max="2500" id = "n"
                    oninput="document.getElementById('genn').
                      innerHTML = this.value"/>
                    <label id="genn" style="display:
                      inline-block; width: 35px;">10</label>
                      <label style="display: inline-block;
                      padding-right: 5px;">samples per
                      experiment</label>
                    <button type="button" onclick="sampler(
                      document.getElementById('n').value);">
                    Run 1 Experiment</button>
                </span>
                <br>
                <br>
                <span>
                    <label id="sampleCount">0</label> <label
```

```
                            style="display: inline-block;
                            padding-right: 15px;">sample(s) taken
                            </label>
                        <button type="button" onclick="plotter()"
                            >Reset</button>
                    </span>
                </form>
            </span>
            <div id="box2" class="jxgbox" style="width:500px;
               height:500px; margin:10px; border-color: black;">
               </div>
        </div>

        <script>
            var points;
            var pt;
            var mean;
            var sd;
            var a;
            var b;
            var points = [];
            var gps = [];

            function clearBoard(){
                try{
                    board.removeObject(gps)
                }
                finally{
                    Plotly.newPlot("box2", []);
                    window.plot.remove();
                    means = []
```

```
                    document.getElementById("sampleCount").innerHTML = 0;
                }
            }

            // Normal Functions

            // Box Muller transform
            function stdNorm(m, s) {
                var u = 0
                var v = 0;
                while(u == 0){ // While loop ensures (0,1) not [0,1)
                    u = Math.random();
                }
                while(v == 0){
                    v = Math.random();
                }
                // Transform to make Z0
                var val = (Math.sqrt( -2 * Math.log( u ) ) * Math.cos( 2 * Math.PI * v ));
                // Return value, scaled by mean and sd
                return (s*val) + m;
            }

            function phi(x, m, s){
                var o2pi = 1/(Math.sqrt(2*Math.PI)*s); // Calculates 1/sqrt(2*pi)
                var ez = Math.exp(-(Math.pow((x - m)/s, 2))/2); // Calculates e^(-(z^2)/2)
                return ((o2pi * ez));  // Multiply them together
```

```
                to caclulate pdf
        }

        // Uniform Functions

        function unigen(a,b){
            v = a + (Math.random()*(b-a));
            return v
        }

        function uni(x,a,b){
            if(x>b || x<a){
                return 0
            }
            return 1/(b-a);
        }

        //Exponential Functions

        function expGen(lam) {
            //Metropolis-Hastings
            var x = 1;
            var y = 1;
            var alpha = 0;
            var p=0;
            for (var i = 0; i < 100; i++) {
                y = stdNorm(x, 1/lam)
                alpha = Math.min(1, (exp(y, lam))/(exp(x, lam
                   )))
                p = Math.random();
                if(p<alpha){
```

```
                        x = y;
                    }
                }
                return x;
            }

            function exp(x, lam){
                if(x>0){
                    return lam*Math.exp(-lam*x);
                }else{
                    return 0;
                }
            }


            // Draws appropriate graph

            function plotter(){
                clearBoard()
                distribution =  document.getElementById("dist").
                   innerHTML.toLowerCase()

                switch(distribution){
                    case "normal":
                        mean = parseFloat(document.getElementById
                           ("mean").value);
                        sd = parseFloat(document.getElementById("
                           sd").value);
                        window.plot = board.create("functiongraph
                           ", [function(x){return phi(x, mean, sd)
                           ;}]);
```

```
                        break;
                    case "uniform":
                        a = parseFloat(document.getElementById("a").value);
                        b = parseFloat(document.getElementById("b").value);
                        window.plot = board.create("functiongraph", [function(x){return uni(x,a,b);}]);
                        break;
                    case "exponential":
                        lam = parseFloat(document.getElementById("lam").value);
                        window.plot = board.create("functiongraph", [function(x){return exp(x,lam);}]);
                        break;
                }
            }

            function sampler(runs){
                distribution =  document.getElementById("dist").innerHTML.toLowerCase()
                size=document.getElementById("sslide").value;
                try{
                    board.removeObject(gps)
                }finally{}7

                for (var index = 0; index < runs; index++) {
                        total = 0;
                            for (var i = 0; i < size; i++) {
                                switch(distribution){
                                    case "normal":
```

```
                mean = parseFloat(document.getElementById("mean").value);
                sd = parseFloat(document.getElementById("sd").value);
                pt = stdNorm(mean, sd);
                break;
            case "uniform":
                a = parseFloat(document.getElementById("a").value);
                b = parseFloat(document.getElementById("b").value);
                if(a>b){
                    alert("Cannot generate sample with a>b")
                    index = runs;
                    i = size;
                    break;
                }
                pt = unigen(a,b);
                break;
            case "exponential":
                lam = parseFloat(document.getElementById("lam").value);
                if(lam<=0){
                    alert("Cannot
```

```
                        generate sample
                        with rate less than
                         or equal to 0")
                      index = runs;
                      i = size;
                      break;
                    }
                    pt = expGen(lam);
                    break;
                }
                total = total + pt;
                if (index===(runs-1)){
                  apoint = board.create("point"
                    , [pt, 0]).setLabelText(""
                    );
                  gps.push(apoint);
                }
              }

              sampleVal = parseFloat(document.
                getElementById("sampleCount").
                innerHTML)
              document.getElementById("sampleCount"
                ).innerHTML = sampleVal + 1

              smean = (total/size);
              means.push(smean);
          }

        trace = {
          x: means,
```

```
                    type: 'histogram',
                };
                data = [trace];
                Plotly.newPlot("box2", data);
            }

            // Selector functions

            function normSelector(){
                board.removeObjects
                means=[]
                Plotly.newPlot("box2", []);
                Plotly.removeObject
                document.getElementById("params").innerHTML = "";
                document.getElementById("dist").innerHTML = "
                   Normal";

                var meanLab = document.createElement("TEXT");
                meanLab.innerHTML = "Mean: "
                document.getElementById("params").appendChild(
                   meanLab);
                var meanIn = document.createElement("INPUT");
                meanIn.addEventListener("input", function(){
                   plotter()}, false)
                meanIn.id = "mean"
                meanIn.value = 0
                meanIn.style = "text-align: center; width: 35px;"
                document.getElementById("params").appendChild(
                   meanIn);

                var sdLab = document.createElement("TEXT");
```

```
            sdLab.innerHTML = " Standard Deviation: "
            document.getElementById("params").appendChild(
               sdLab);
            var sdIn = document.createElement("INPUT");
            sdIn.id = "sd"
            sdIn.value = 1
            sdIn.style = "text-align: center; width: 35px;"
            sdIn.addEventListener("input", function(){plotter
               ()}, false)
            document.getElementById("params").appendChild(
               sdIn);
            plotter();
        }

        function uniSelector(){
            document.getElementById("params").innerHTML = "";
            document.getElementById("dist").innerHTML = "
               Uniform";

            var aLab = document.createElement("TEXT");
            aLab.innerHTML = "a: "
            document.getElementById("params").appendChild(
               aLab);
            var aIn = document.createElement("INPUT");
            aIn.value = -1;
            aIn.id="a"
            aIn.style = "text-align: center; width: 35px;"
            aIn.addEventListener("input", function(){plotter
               ()}, false)
            document.getElementById("params").appendChild(aIn
               );
```

```

        var bLab = document.createElement("TEXT");
        bLab.innerHTML = " b: "
        document.getElementById("params").appendChild(
            bLab);
        var bIn = document.createElement("INPUT");
        bIn.value = 1;
        bIn.id="b"
        bIn.style = "text-align: center; width: 35px;"
        bIn.addEventListener("input", function(){plotter
            ()}, false)
        document.getElementById("params").appendChild(bIn
            );
        plotter();
    }

    function expSelector(){
        document.getElementById("params").innerHTML = "";
        document.getElementById("dist").innerHTML = "
            Exponential";

        var lamLab = document.createElement("TEXT");
        lamLab.innerHTML = "Rate (lambda): "
        document.getElementById("params").appendChild(
            lamLab);
        var lamIn = document.createElement("INPUT");
        lamIn.value = 1;
        lamIn.id="lam"
        lamIn.style = "text-align: center; width: 35px;"
        lamIn.addEventListener("input", function(){
            plotter()}, false)
```

```
                document.getElementById("params").appendChild(
                    lamIn);
                plotter();
            }

            var board = JXG.JSXGraph.initBoard("box1", {
                boundingbox: [-5, 1, 5, -0.2], axis:true
            });

            var plot = board.create("functiongraph", [function(x)
               {return phi(x);}]);

            normSelector()

            </script>
        </main>
    </body>
</html>
```

References

[1] Cristiane Neri Nobre, Magali Rezende Gouvêa Meireles, Niltom Vieira Junior, Mônica Neli de Resende, Lucivânia Ester da Costa, Rejane Corrêa da Rocha, *The Use of Geogebra Software as a Calculus Teaching and Learning Tool*,
Vilnius University, Information in Education, 2016, Vol.15, No. 2, 253-267,
Available at: files.eric.ed.gov/fulltext/EJ1117161.pdf [Accessed January 7, 2022]

[2] Edgar Dale, *Audiovisual Methods in Teaching*,
3rd Edition, Dryden Press, 1969

[3] Jerome S. Bruner, *Toward a Theory of Instruction*,
Belknap Press of Harvard University, 1966

[4] Desmos, *Desmos*,
Available at: teacher.desmos.com [Accessed January 6, 2022]

[5] The GeoGebra Group, *GeoGebra*,
Available at: geogebra.org [Accessed January 6, 2022]

[6] La Salle Education, *Autograph*,
Available at: completemaths.com/autograph [Accessed January 6, 2022]

[7] Q-Success, *W3Techs*,
Available at: w3techs.com [Accessed January 6, 2022]

[8] Chart.js Team and Contributors, *Chart.js*,
Available at: chartjs.org [Accessed October 19, 2021]

[9] Dr. Edward Hall, *Introduction to Scientific Computation, Lecture 12*,
February 22 2021, School of Mathematical Sciences, University of Nottingham.

[10] Dr. Kris van der Zee, *Introduction to Scientific Computation, Lecture 1*,
September 30 2020, School of Mathematical Sciences, University of Nottingham.

[11] Dr. David Sirl, *Probability Models and Methods*, p70, Autumn Semester Notes 2020/21 School of Mathematical Sciences, University of Nottingham.

[12] Prof. Ender Özcan, *Artificial Intelligence Methods, Lecture 1*, February 3 2021, School of Computer Science, University of Nottingham.

[13] Mozilla and Individual Contributors, *MDN Web Docs*, Available at: developer.mozilla.org [Accessed January 4, 2022]

[14] TC39, *ECMAScript 2022 Language Specification*, Available at: tc39.es/ecma262 [Accessed January 4, 2022]

[15] Matthias Ehmann, Michael Gerhäuser, Carsten Miller, Alfred Wassermann, Bianca Valentin, and Peter Wilfahrt, *JSXGraph*, Available at: jsxgraph.org [Accessed December 4, 2021]

[16] Plotly Team, *Plotly Graphing Libraries*, Available at: plotly.com [Accessed December 4, 2021]

[17] Wikipedia, *Inverse Transform Sampling*, Available at: wikipedia.org/wiki/Inverse_transform_sampling [Accessed December 4, 2021]

[18] Wikipedia, *Box–Muller Transform*, Available at: wikipedia.org/wiki/Box-Muller_transform [Accessed December 4, 2021]

[19] Prof. Theo Kypraios, *Statistical Inference*, p70, Lecture Notes 2020/21 School of Mathematical Science, University of Nottingham.

[20] R Core Team, *R*, Available at: r-project.org [Accessed January 8, 2021].

[21] Code listing style based on a post by Benny Neugebauer on tex.stackexchange.com [Accessed December 4, 2021]