using System;

using System.Windows.Forms;

namespace Bin\_Program

{

public partial class Bin : Form

{

Random rand = new Random();

public int GetRandomUniform(int min, int max)

{

int uniform = rand.Next(min, max + 1);

return uniform;

}

public double GetRandomNormal(double mean, double stdv)

{

double r = rand.NextDouble();

double phi = rand.NextDouble();

double z = (Math.Cos(2 \* Math.PI \* r)) \* (Math.Sqrt(-2 \* Math.Log(phi)));

double x = z \* stdv + mean;

return x;

}

public int getBinIndex(double mini, double maxi, int numbins, double valuetobin)

{

int binIndex = (int)Math.Ceiling((valuetobin - mini) \* (numbins / (maxi - mini)));

return binIndex;

}

public Bin()

{

InitializeComponent();

}

private void Form1\_Load(object sender, EventArgs e)

{

}

private void chart1\_Click(object sender, EventArgs e)

{

}

private void textBox2\_TextChanged(object sender, EventArgs e)

{

}

private void textBox4\_TextChanged(object sender, EventArgs e)

{

}

private void radioButton4\_CheckedChanged(object sender, EventArgs e)

{

if (rdoUniPV.Checked)

{

lblMeanMinPV.Text = "Min";

lblStMaxPV.Text = "Max";

}

else

{

lblMeanMinPV.Text = "Mean:";

lblStMaxPV.Text = "Standard Dev:";

}

}

private void label1\_Click(object sender, EventArgs e)

{

}

private void lblNumBins\_Click(object sender, EventArgs e)

{

}

private void textBox3\_TextChanged(object sender, EventArgs e)

{

}

private void rdoNormN\_CheckedChanged(object sender, EventArgs e)

{

if (rdoNormN.Checked)

{

lblMinMeanN.Text = "Mean:";

lblStMaxN.Text = "Standard Dev:";

}

else

{

lblMinMeanN.Text = "Min:";

lblStMaxN.Text = "Max:";

}

}

private void btnRunSim\_Click(object sender, EventArgs e)

{

try

{

chtBin.Series[0].Points.Clear();

chtBin.ChartAreas[0].AxisX.CustomLabels.Clear();

int i, numBins;

double nMinMean, nMaxSt, PvMeanMin, PvStdMax;

//^variables are named in such a way so that it accounts for both normal and uniform distribtion of n and Pv

i = int.Parse(txtNumI.Text);

numBins = int.Parse(txtNumBins.Text);

nMinMean = double.Parse(txtMeanN.Text);

nMaxSt = double.Parse(txtStDvN.Text);

PvMeanMin = double.Parse(txtMeanPV.Text);

PvStdMax = double.Parse(txtStDvPV.Text);

double PvMin, PvMax, PtMin, PtMax;

double nMin, nMax;

double binWidth;

string[] binLabels = new string[numBins];

//^will be used to store bin labels, which will then be added in the chart

int[] binValues = new int[numBins];

double PtTotal = 0;

double PtAvg = 0;

if (rdoNormPV.Checked && rdoUniN.Checked) //For Pv normal and uniform N

{

PvMin = PvMeanMin - (3 \* PvStdMax);

PvMax = PvMeanMin + (3 \* PvStdMax);

PtMin = PvMin \* nMinMean;

PtMax = PvMax \* nMaxSt;

for (int j = 0; j < i; j++)

{

int n = GetRandomUniform((int)nMinMean, (int)nMaxSt);

double Pv = GetRandomNormal(PvMeanMin, PvStdMax);

double Pt = Pv \* n;

PtTotal += Pt;

int binIndex = getBinIndex(PtMin, PtMax, numBins, Pt);

if (binIndex > 0 && binIndex <= numBins)

{

binValues[binIndex - 1] += 1;

}

}

}

else if (rdoNormPV.Checked && rdoNormN.Checked) // For normal on Pv and n

{

PvMin = PvMeanMin - (3 \* PvStdMax);

PvMax = PvMeanMin + (3 \* PvStdMax);

nMin = nMinMean - (3 \* nMaxSt);

nMax = nMinMean + (3 \* nMaxSt);

PtMin = PvMin \* nMin;

PtMax = PvMax \* nMax;

for (int j = 0; j < i; j++)

{

double n = GetRandomNormal(nMinMean, nMaxSt);

double Pv = GetRandomNormal(PvMeanMin, PvStdMax);

double Pt = Pv \* n;

PtTotal += Pt;

int binIndex = getBinIndex(PtMin, PtMax, numBins, Pt);

if (binIndex > 0 && binIndex <= numBins)

{

binValues[binIndex - 1] += 1;

}

}

}

else if (rdoUniPV.Checked && rdoUniN.Checked) // for uniform on Pv and n

{

PtMin = PvMeanMin \* nMinMean;

PtMax = PvStdMax \* nMaxSt;

for (int j = 0; j < i; j++)

{

int n = GetRandomUniform((int)nMinMean, (int)nMaxSt);

int Pv = GetRandomUniform((int)PvMeanMin, (int)PvStdMax);

double Pt = Pv \* n;

PtTotal += Pt;

int binIndex = getBinIndex(PtMin, PtMax, numBins, Pt);

if (binIndex > 0 && binIndex <= numBins)

{

binValues[binIndex - 1] += 1;

}

}

}

else //for uniform Pv and normal n

{

nMin = nMinMean - (3 \* nMaxSt);

nMax = nMinMean + (3 \* nMaxSt);

PtMin = PvMeanMin \* nMin;

PtMax = PvStdMax \* nMax;

for (int j = 0; j < i; j++)

{

double n = GetRandomNormal(nMinMean, nMaxSt);

int Pv = GetRandomUniform((int)PvMeanMin, (int)PvStdMax);

double Pt = Pv \* n;

PtTotal += Pt;

int binIndex = getBinIndex(PtMin, PtMax, numBins, Pt);

if (binIndex > 0 && binIndex <= numBins)

{

binValues[binIndex - 1] += 1;

}

}

}

PtAvg += PtTotal / i;

PtAvg = Math.Round(PtAvg, 2);

txtAvg.Text = PtAvg.ToString();

binWidth = (PtMax - PtMin) / numBins;

for (int j = 0; j < numBins; j++)

{

chtBin.Series[0].Points.AddXY(j + 1, binValues[j]);

binLabels[j] = $"{(PtMin + j \* binWidth):F1}-{(PtMin + (j + 1) \* binWidth):F1}";

//^Create Bin Labels in the format 'LowerBound-UpperBound'

chtBin.ChartAreas[0].AxisX.CustomLabels.Add(j + 1 - 0.5, j + 2 - 0.5, binLabels[j]);

//^add them to the x-axis and subtract each increment by 0.5 so that the labels are

properly centered on the bins

}

}

catch

{

MessageBox.Show("Error parsing values");

}

}

private void txtNumBins\_TextChanged(object sender, EventArgs e)

{

}

private void lblNumI\_Click(object sender, EventArgs e)

{

}

private void btnExit\_Click(object sender, EventArgs e)

{

Application.Exit();

}

}

}