```
'This is the code to generate the data presented in Figures 1 and 2 of the the paper
'"Induction with and without natural properties: a new approach to the New Riddle of Induction".
'The program was created in Visual Studio 2019.
'To run the program, create a windows form application with single button "Button 1", paste this code into Form1.vb, and place a breakpoint at line 66 (End Sub).
'After executing the program, press Button 1.
'Once the program breaks at line 66, data of the type presented in Figure 1 is stored in the array "Mean_Error_at_Freq",
'and data of the type presented in Figure 2 is stored in the array "Mean_Error_for_Grue_at_Freq".

Imports System.Math

Public Class Form1

    Dim Population_Size As Integer = 10000
    Dim Sample_Size As Double = 100
    Dim Big_Loop_Size As Integer = 1000000 'Number of samples taken for each possible frequency of G in the population
    Dim Granularity As Integer = 100 'Determines what possible frequencies of Gs in the population are computed. For maximum Granularity to Population_Size. See line 33.

    Dim RandomClass As New Random()
    Dim RandomNumber As Double

    Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button1.Click

        Dim Number_of_Gs As Integer
        Dim Remaining_Population As Integer
        Dim G_Count As Integer
        Dim Mean_Error_at_Freq(Granularity) As Double
        Dim Mean_Error_for_Grue_at_Freq(Granularity) As Double
        Dim Grue_Freq As Double

        For freq = 0 To Granularity 'Loops through different possible frequencies of Gs in the population. See line 42.

            Mean_Error_at_Freq(freq) = 0
            Mean_Error_for_Grue_at_Freq(freq) = 0

            For runs = 1 To Big_Loop_Size

                Remaining_Population = Population_Size
                Number_of_Gs = (freq / Granularity) * Population_Size
                G_Count = 0

                For sample_item = 1 To Sample_Size
                    RandomNumber = RandomClass.NextDouble()
                    RandomNumber = RandomNumber * Remaining_Population
                    If RandomNumber <= Number_of_Gs Then
                        G_Count = G_Count + 1
                        Number_of_Gs = Number_of_Gs - 1
                    End If
                    Remaining_Population = Remaining_Population - 1

                Next

                Grue_Freq = (G_Count + (Population_Size - Sample_Size) - (Number_of_Gs - G_Count)) / Population_Size

                Mean_Error_at_Freq(freq) = Mean_Error_at_Freq(freq) + Sqrt(((freq / Granularity) - (G_Count / Sample_Size)) * ((freq / Granularity) - (G_Count / Sample_Size)))
                Mean_Error_for_Grue_at_Freq(freq) = Mean_Error_for_Grue_at_Freq(freq) + Sqrt((Grue_Freq - (G_Count / Sample_Size)) * (Grue_Freq - (G_Count / Sample_Size)))

            Next

            Mean_Error_at_Freq(freq) = Mean_Error_at_Freq(freq) / Big_Loop_Size
            Mean_Error_for_Grue_at_Freq(freq) = Mean_Error_for_Grue_at_Freq(freq) / Big_Loop_Size

        Next


    End Sub 'Put break point here.


End Class
```