# Comparing Code and Results of Basic Plotting across Programs

David M Vermillion
davidmvermillion.com
October 5th, 2020

## Abstract

This project exists as a quick reference guide for the work required to achieve simple graphs while showing how common programs develop those graphs. Some graphs are created using a GUI and some using command line coding. More customization is possible than is shown in this document. This document shows the approximate workload and default or near-default design philosophies of each program. Graphing examples are shown using Google Sheets, Excel, LibreCalc, Matlab, Mathematica, ggplot, ggplot2, Gluviz, GNU Octave, gnuplot, Matplotlib, Pandas, Past, R Studio, and Sagemath. The dataset used for this project was created in Google Sheets using the code snippet `=RANDBETWEEN(1,100)` across a swath of cells 2 rows wide and 50 columns deep. The column to the left was set from 1 to 50 to label the counts of each data series, resulting in a file with 3 columns and 50 rows. This instructional document was created using LibreOffice Writer in an .odt file before being exported as a PDF for distribution to maintain formatting across viewing devices.

## Sheets

Using the originating spreadsheet, the first graph (**Figure 1**) was a default scatterplot created using Google Sheets (Insert > Chart) with two trendlines added (Edit Chart > Customize > Series > Trendline > Linear > Line Opacity = 100%) and downloaded as an SVG. Following the path (Customize > Chart & axis titles), the chart title was set as displayed under the chart title drop-down. In the same drop-down, the horizontal and vertical axes were set as displayed. Under the Series tab, after selecting the individual series set, (Label > Use Equation) was chosen as the name for the lines. All other default settings were kept. All graph programming was made using the GUI.

**Figure 2** is obtained by tweaking the existing chart to create a histogram as shown. Under Setup, the Chart Type drop-down list contains a histogram option. Under Data Range, the data was changed from `A1:C50` to `B1:C50` to filter out the existing count column. Under (Customize > Histogram), the bucket size was manually set to 15. The vertical axis title (same path as the last chart) was set to Frequency. The horizontal axis title was reset to Data Value. This clearly shows a skewing toward the bottom values for both sets of *random* numbers. No further customizations were made. All changes made were executed through the GUI.

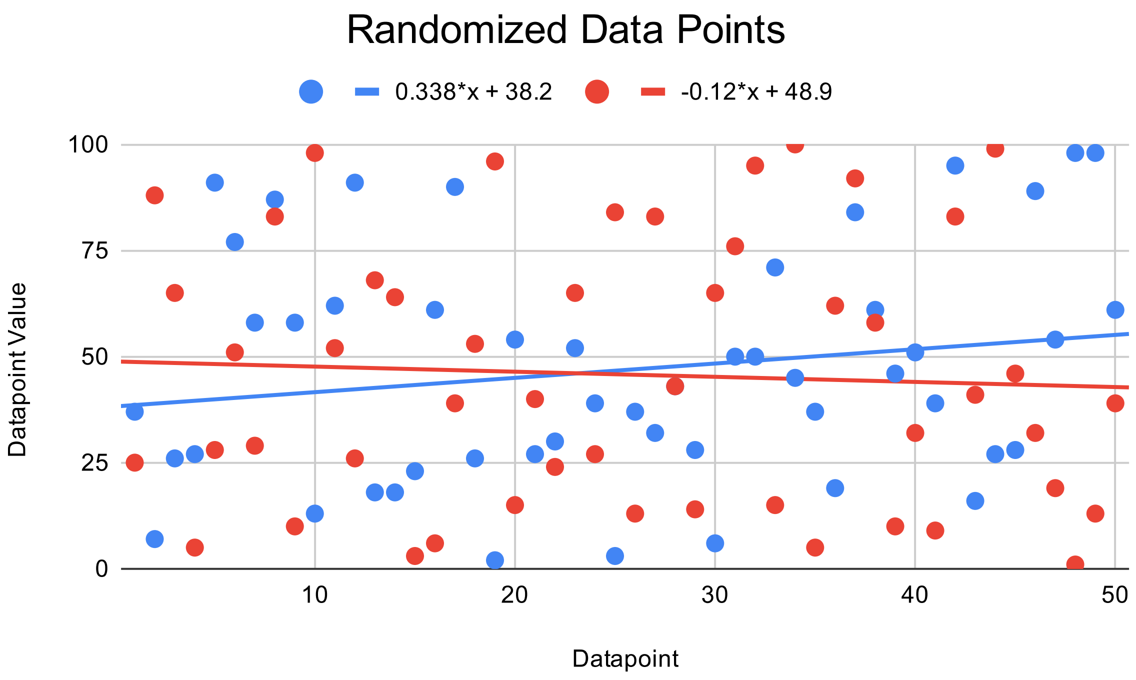

*Figure 1: Google Sheets Scatterplot with Trendlines and Labels*

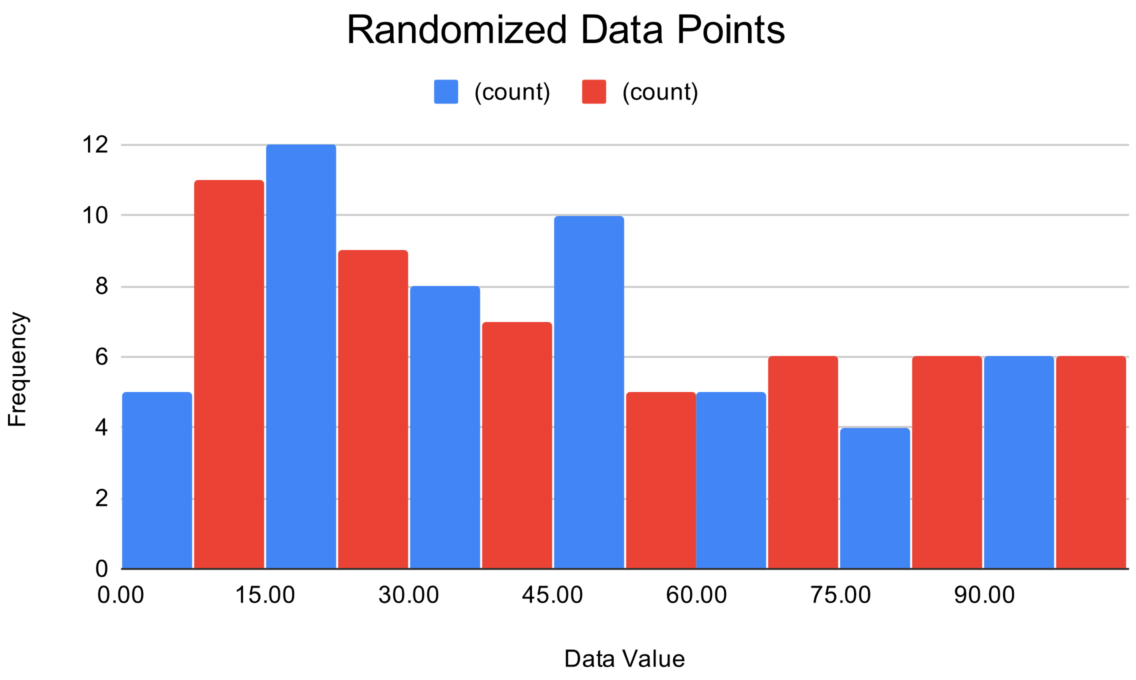

*Figure 2: Google Sheets Histogram*

## Excel

The Google Sheets dataset was downloaded as a .xlsx file for use in Microsoft Excel. Following the menu options (Ribbon Bar > Insert > Scatter) populated a scatterplot from the selected range. Double-clicking the default chart title allowed it to be changed as shown. Clicking the GUI to add a chart element and selecting Axis Titles populated default titles which could then be changed through a double-click action. On the Ribbon Bar element for Chart Design, the following path inserted trendlines. (Add Chart Element > Trendline > Linear). The process was repeated for Series 1 and Series 2. Selecting the trendlines and going to the Format Trendline Window, the button was selected to display the equation on the chart. The default placement had to be changed for readability. Under label options in the same window, a side button opens for Text Options. Color selection is then possible. This was changed from the default color (same as other labels) to match the series colors. The formula bar displayed `=SERIES(,Data!$A$1:$A$50,Data!$C$1:$C$50,2)` as the series selection while checking Series 2 on the graph. No further changes were made. All settings were completed using the GUI. Excel does not allow directly exporting graphs. Therefore they must be copied/pasted through the computer clipboard to a different program. For this demonstration, **Figure 3** was pasted into Inkscape to save as an SVG. The dotted trendlines display in Excel and Inkscape as ~ 20% of the series dot size. They currently show as ~ 80% of the series dot size in this document.

Selecting all three columns, a histogram was created by the same path of (Ribbon Bar > Insert > Histogram). This time the graph was proportionally scaled by clicking the right top corner and dragging while holding down the shift key. Axis titles were inserted as before. The data series had to be readjusted by following the path (Ribbon Bar > Chart Design > Select Data). Column A was renamed as Positions, Column B  as Series 1, and Column C as Series 2. Series 2 data selection was coded as `=Data!$C$18:$C$50` as displayed during the selection process. Positions was also placed into the Horizontal (Category) Axis Labels. After selecting the horizontal axis, going to the window with (Format Axis > Axis Options > Axis Options) the bin width was manually adjusted to 15. This resulted in **Figure 4**. It therefore appears that default histogram plotting of multivariate data is not currently an option in Excel. However, the workaround is explained in the following paragraph.

Utilizing hints from this 2010 University of Oregon class handout, the following steps were devised after creating a second data sheet for continuity: First, Series 1 and Series 2 stats were placed as described using the `MIN, AVERAGE, MAX`, and `STDEV.S` functions. Next, bins were set in increments of 20 from 0 to 100. The count formula was entered with this as a sample:

```
=COUNTIF($B$2:$B$51,">="&E8)-COUNTIF($B$2:$B$51,">="&E9)
```

A clustered column was selected from the ribbon bar for inserting charts with data from each series counts included, but not the bins. The chart title was changed for consistency. Right clicking on the chart, Select Data was chosen. Editing the Horizontal (Category) Axis Labels brought up a window to select the range, which allowed the Bin values to be selected. Right clicking one of the bars, the menu option Format Data Series brought up an option to change the Gap Width from the default 150% to 70% for this demo. Horizontal and vertical labels were created and changed as previously discussed. **Figure 5** was exported as the past Excel charts were.

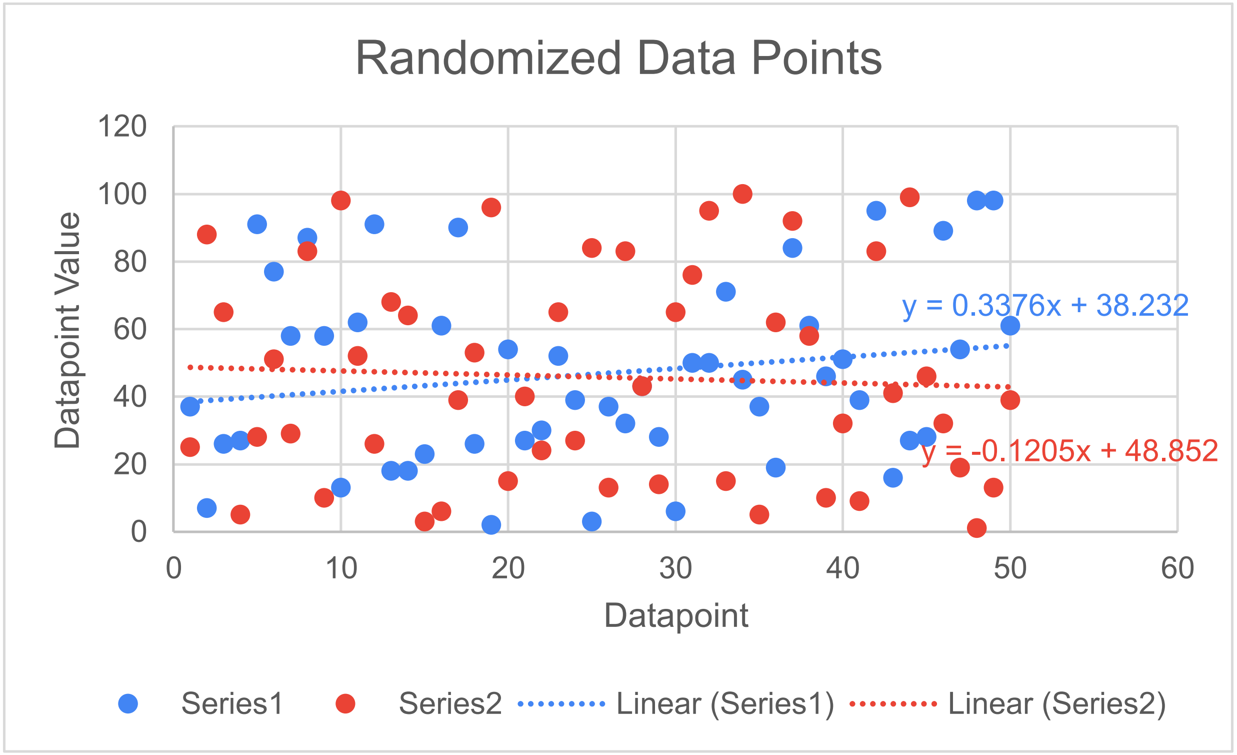

*Figure 3: Excel Scatterplot*

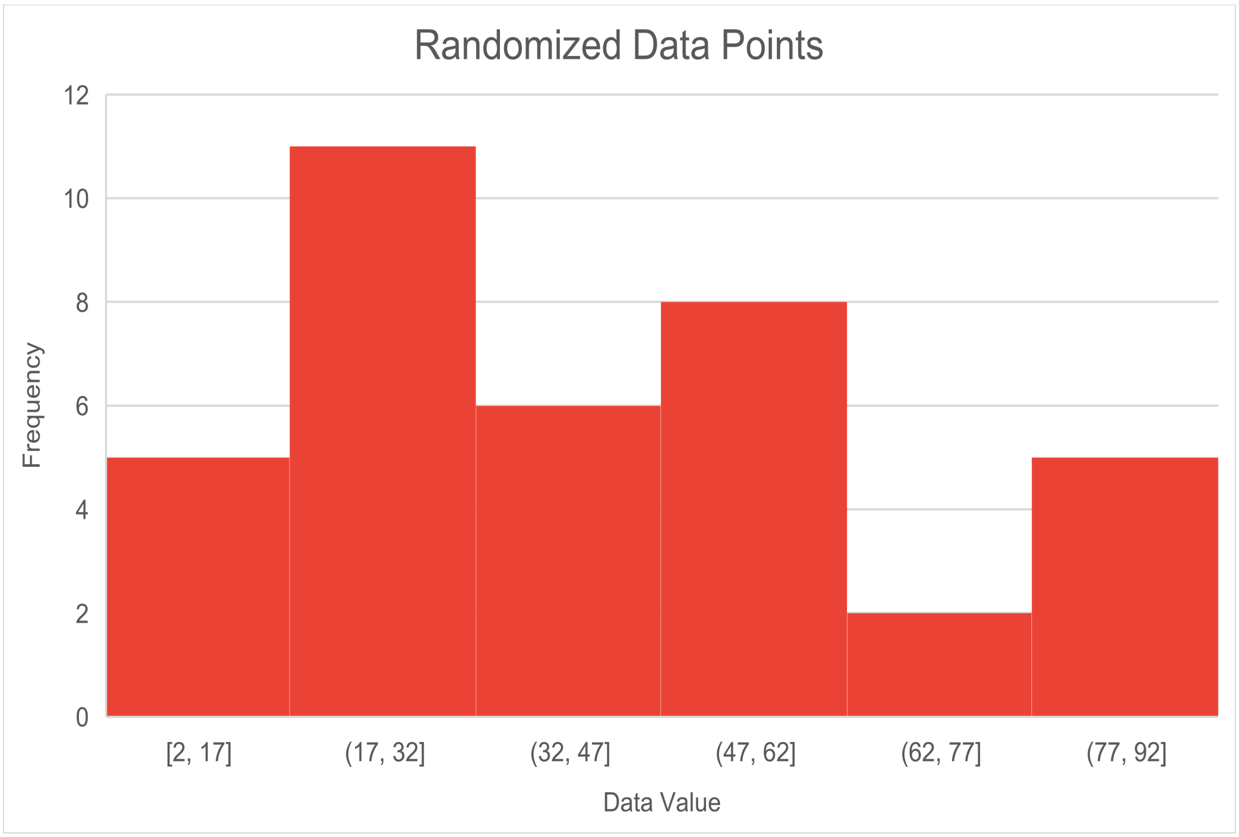

*Figure 4: Excel Default Histogram*

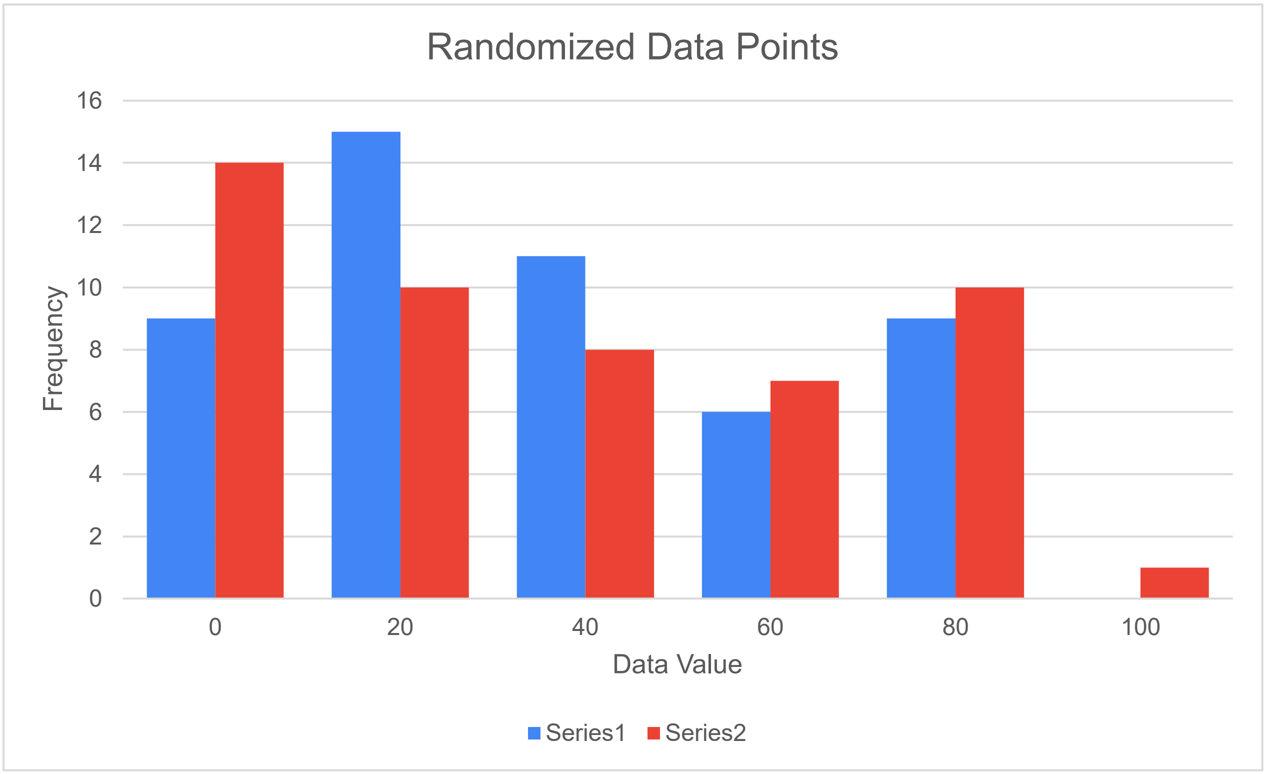

*Figure 5: Excel Histogram with Two Data Series on Same Chart*

## LibreCalc

A fresh OpenDocument Sheet was downloaded from Google Sheets to start. This exported the entire sheet with both tabs, so the frozen version was retained. Left click row 1 > Insert Rows Above was chosen to give the data set space to be labeled as the titles used in the Excel Spreadsheet. After selecting the entire data set, the menu path Insert > Chart was followed to open a chart. A 4-step Chart Wizard was opened to create the desired chart. XY (Scatter) was chosen to create a scatterplot. Under the Data Range selection, the second checkbox selection was made so that the first column and row were the labels. Under chart elements, the Title, X Axis, and Y Axis fields were populated as shown. After selecting Finish, each data series was selected. Following Insert > Trend Line, a linear regression was chosen and Show Equation was toggled on. Double-clicking the equation opens another window. Font Effects > Font Color allowed the option to change the color of the equations so they could be distinguished. The equation decimal places were changed by right-clicking the equation > Format Trendline Equation… > Numbers and changing the decimal places to 2. Right-clicking the chart (after exiting the editing phase) and selecting Export As Image allowed the SVG to be easily exported. **Figure 6** is the result.

No native histogram option exists, so similar workaround steps were performed to create it as shown for Excel. First, Series 1 and Series 2 stats were placed as described using the `MIN, AVERAGE, MAX`, and `STDEV.S` functions. Next, bins were set in increments of 20 from 0 to 100 using the extension handle and double-clicking. The count formula was entered with this as a sample:

```
=COUNTIF($B$2:$B$51,">="&E8)-COUNTIF($B$2:$B$51,">="&E9)
```

Selecting Insert > Chart > Column allowed the creation of the standard histogram. All previous adjustments were made as described. To adjust the spacing, right-click a bar, select Format Data Series, and set spacing to 70%. Results shown in **Figure 7**.

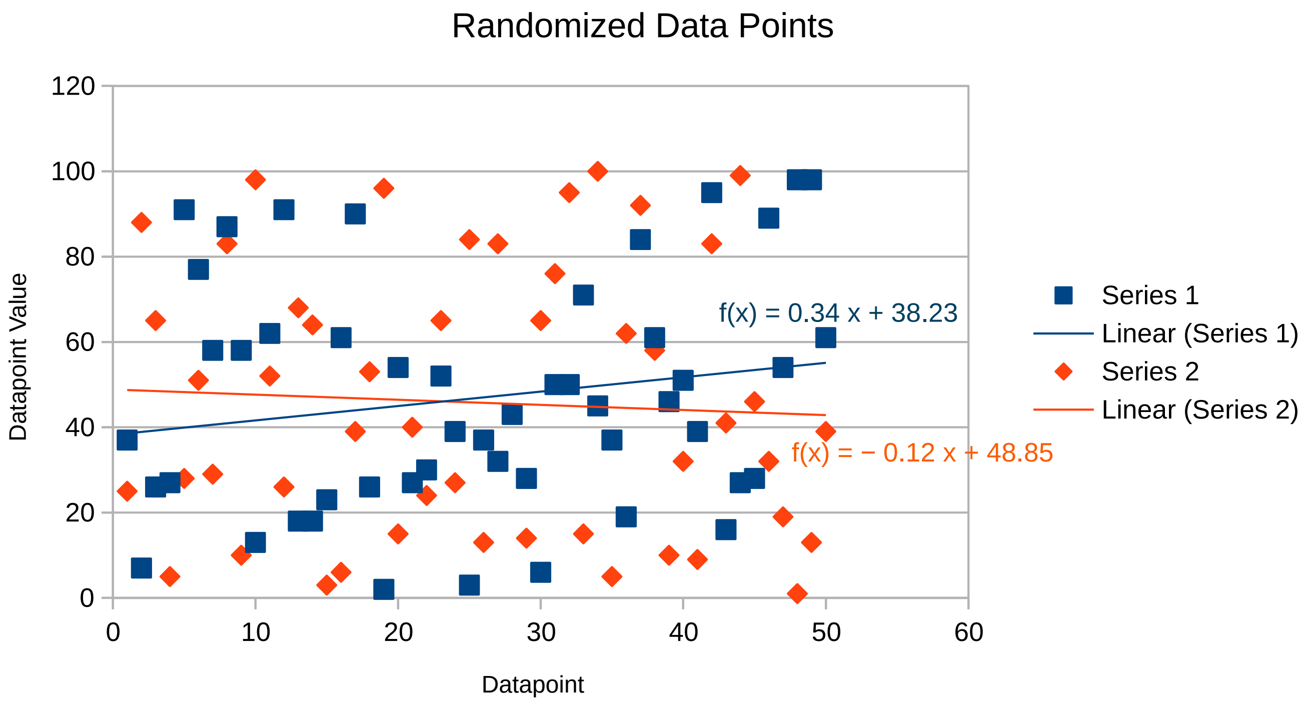

*Figure 6: LibreCalc Scatterplot*

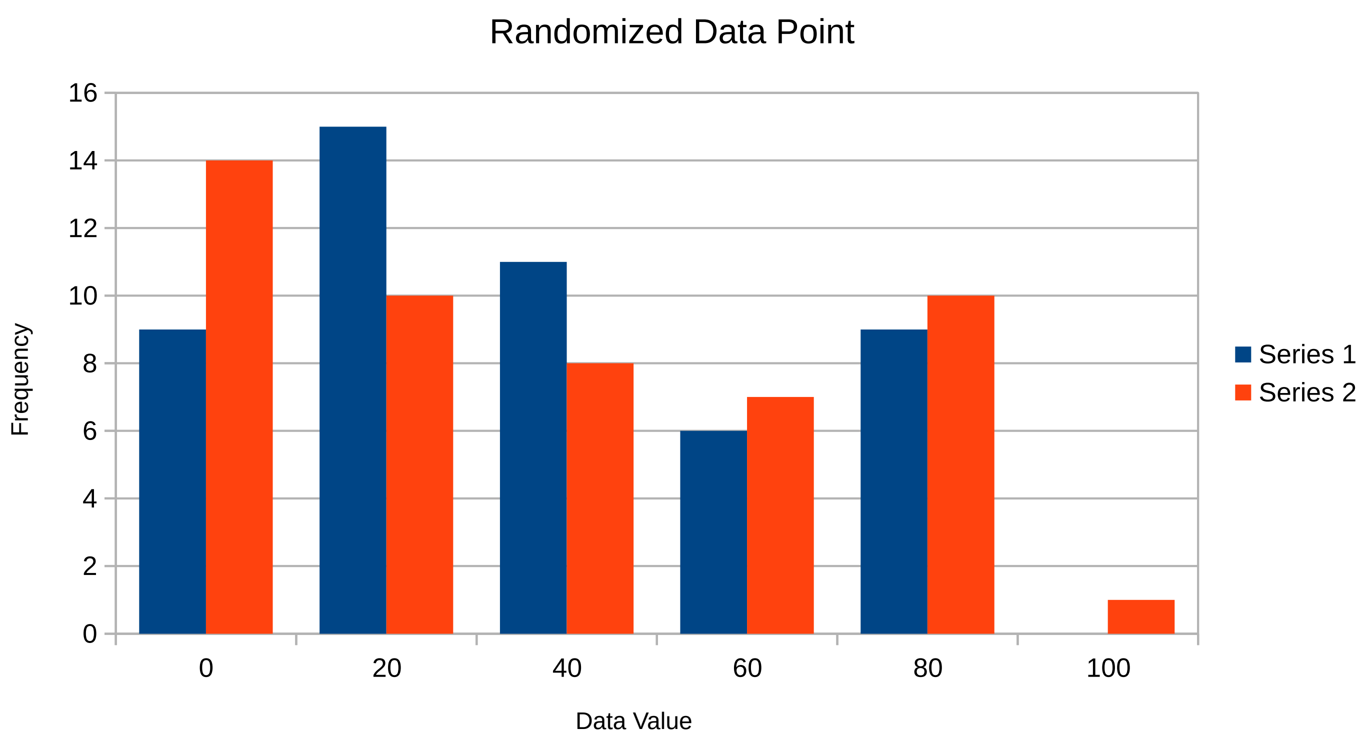

*Figure 7: LibreCalc Histogram*

## MATLAB

Opening Matlab, a button exists to Import Data. Selecting it and navigating to the CSV, allowed it to be imported. Selecting the Import button imported the data as a 50x3 table. Double-selecting the value number opened the table. From there, selecting the rows and right-clicking gave the option to follow Create New Workspace Variable From Selection > New Numeric Array. These were then renamed using `Labels = VarName1; Series_1 = VarName2; Series_2 = VarName3;`. From there, the following code was used to create **Figure 8**. The equations require additional string work to automate. Some level of creating a string, splitting it, resorting it, shrinking it, and splicing it will be required. This will be found in a later iteration, or as a shout-out if someone wants to drop the necessary code for this project.

```
% scatterplot
scatter(Labels, Series_1, 'b'), xlabel('Datapoint'), ylabel('Datapoint Value'),
title('Randomized Data Points')
hold on
scatter(Labels, Series_2, 'r'), xlabel('Datapoint'), ylabel('Datapoint Value'),
title('Randomized Data Points')
format long
X = [ones(length(Labels),1) Labels];
% b represents the equation in intercept + slope form
b1 = X\Series_1;
```

```
% b1 = 38.231836734693879 + 0.337575030012005x
% lm1 = linear model 1
lm1 = X*b1;
b2 = X\Series_2;
% b2 = 48.852244897959189 -0.120480192076831x
lm2 = X*b2;
plot(Labels, lm1, 'b', Labels, lm2, 'r')
legend({'Series 1', 'Series 2', 'y = 0.34x + 38.23', 'y = -0.12x + 48.85'},
'Location', 'best')
hold off
```

Matlab allows the creation of superimposed histograms through its native histogram function. The following code generated **Figure 9**.

```
h1 = histogram(Series_1);
h1.BinWidth = 15;
hold on
xlabel('Data Value'), ylabel('Frequency'), title('Randomized Data Points')
h2 = histogram(Series_2);
h2.BinWidth = 15;
legend('Series 1', 'Series 2')
legend('boxoff')
hold off
```

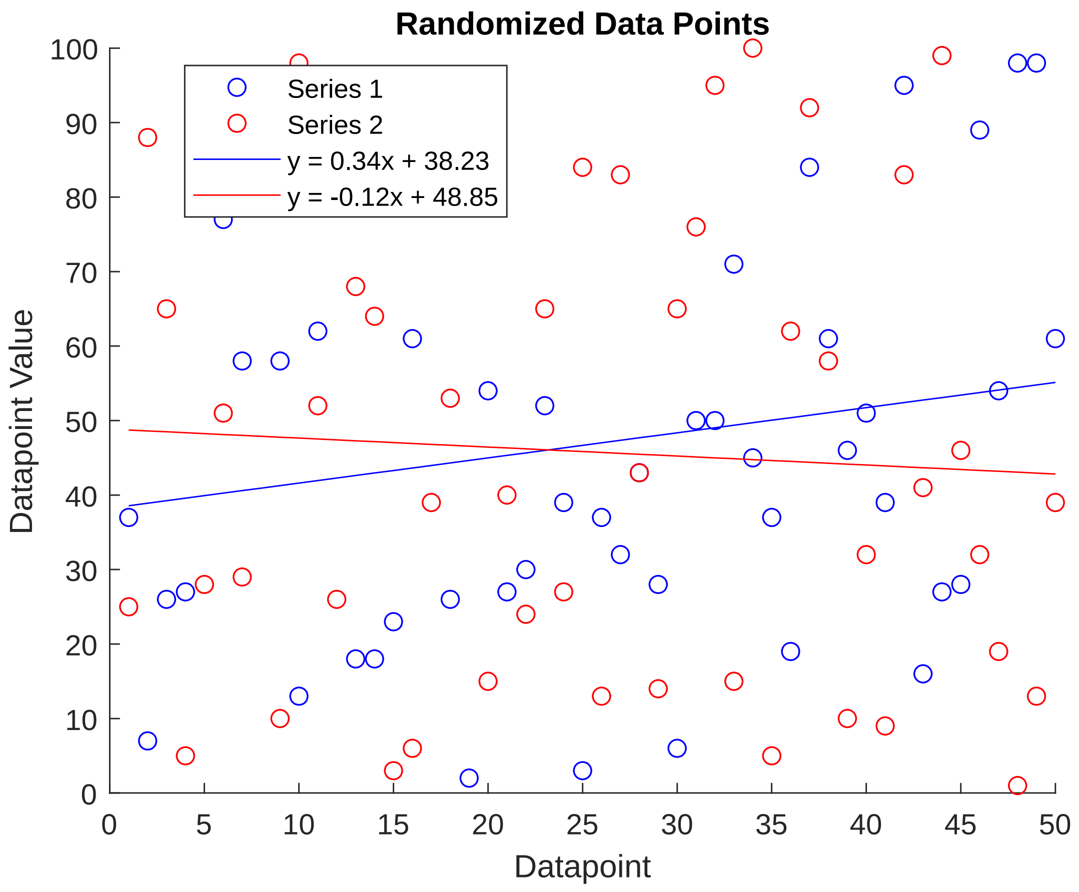

*Figure 8: Matlab Scatterplot*

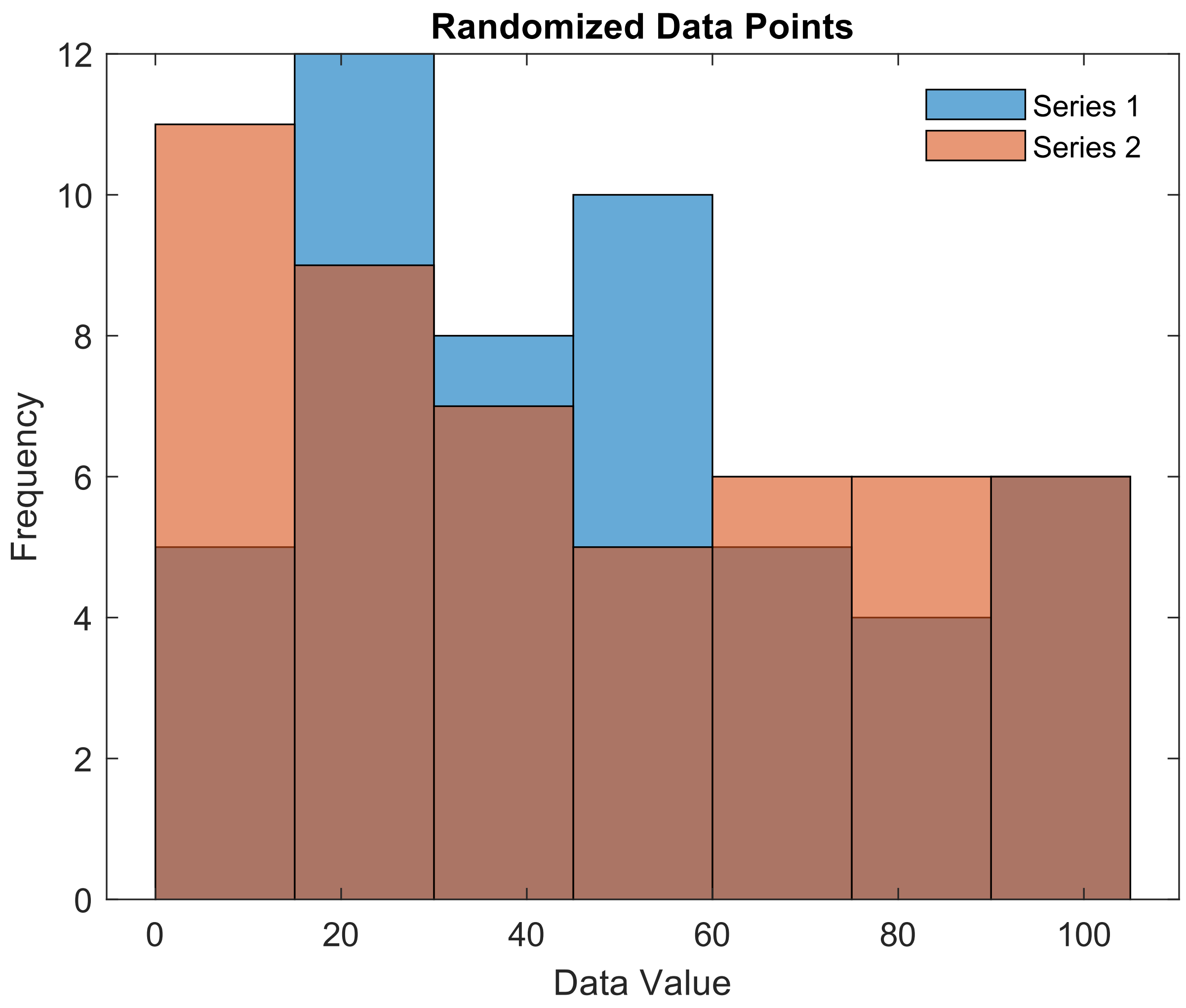

*Figure 9: Matlab Histogram*

## MATHEMATICA

Data imported using the following code:

```
A = Import[
  "C:\Users\david\Documents\GitHub\Chart_Comparisons\Seeded_Values_\
for_Comparison_Project.csv", {"Data", All, 2}]
B = Import[
  "C:\Users\david\Documents\GitHub\Chart_Comparisons\Seeded_Values_\
for_Comparison_Project.csv", {"Data", All, 3}]
```

Best fit lines created using the following code:

```
line1 = Fit[A, {1, x}, x]

line2 = Fit[B, {1, x}, x]
```

Scatterplot made using this code and displayed as **Figure 10:**

```
scatterplot =
 Show[ListPlot[Legended[A, "Series 1"],
   PlotMarkers -> {Automatic, Medium}, PlotStyle -> Red],
  ListPlot[Legended[B, "Series 2"],
   PlotMarkers -> {Automatic, Medium}, PlotStyle -> Blue],
  Plot[{Legended[line2, "-0.12x + 48.9"],
    Legended[line1, "0.34x + 38.2"]}, {x, 0, 50}],
  AxesLabel -> {HoldForm[Datapoint], HoldForm[Datapoint Value]},
  PlotLabel ->
   RawBoxes[
    RowBox[{RowBox[{"Randomized", " ", "Data", " ", "Points"}]}]],
  LabelStyle -> {GrayLevel[0]}]

Export["MathematicaScatterplotExport.svg", scatterplot,
 ImageResolution -> 2000]
```

To develop the histogram displayed in **Figure 11**, the following code was used:

```
H = Histogram[{Legended[A, "Series 1"], Legended[B, "Series 2"]},
  AxesLabel -> {HoldForm[Data Value], HoldForm[Frequency]},
  PlotLabel ->
   RawBoxes[
    RowBox[{RowBox[{"Randomized", " ", "Data", " ", "Points"}]}]],
  LabelStyle -> {GrayLevel[0]}]

Export["MathematicaHistogram.svg", H, ImageResolution -> 2000]
```

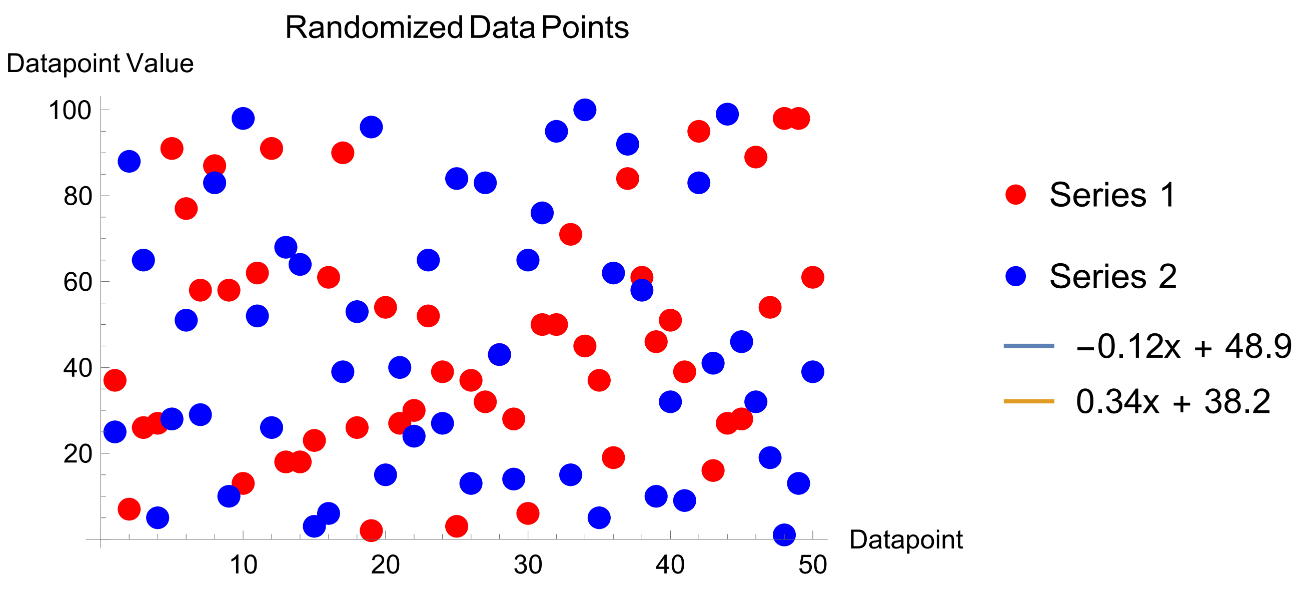

*Figure 10: Mathematica Scatterplot*

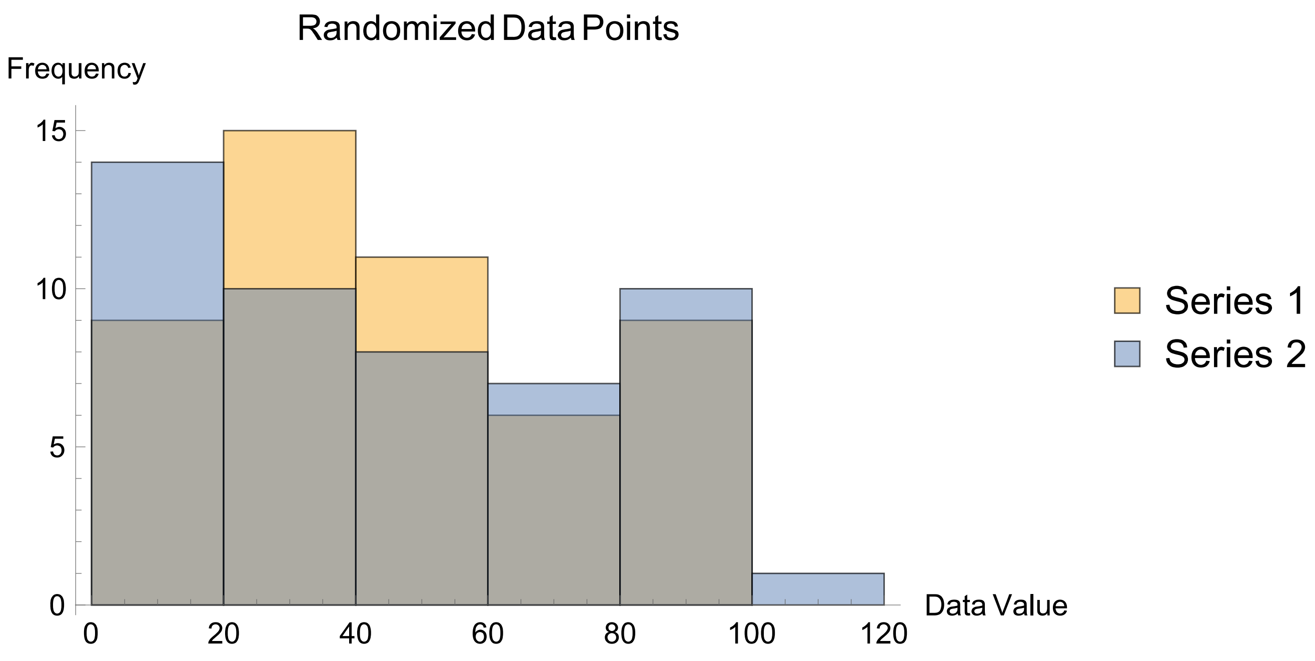

*Figure 11: Mathematica Histogram*

## GGPLOT

ggplot is a Python package based off the R package ggplot2, which is demonstrated in the next section, effectively making it a Python wrapper of an R package. For ggplot, the Anaconda IDE is utilized. Some level of further debugging may be required for packages to work correctly. Will proceed with ggplot2 next and return to ggplot later.

```
ggplot(a, aes(x = Datapoint, y = value, color = variable)) +

  geom_point(aes(y = Series1, col = 'Series1')) +

  geom_point(aes(y = Series2, col = 'Series2')) +

  xlab('Datapoint') +

  ylab('Datapoint Value')
```

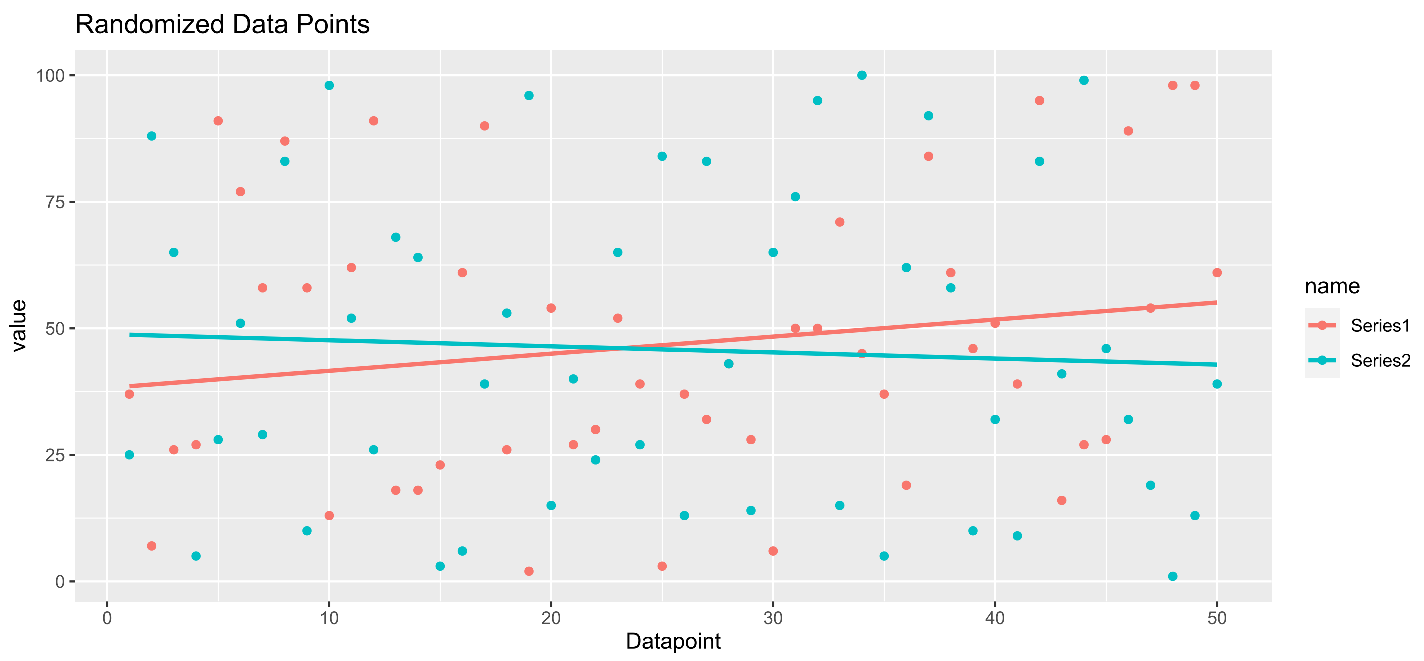

*Figure 12: ggplot Scatterplot (not)*

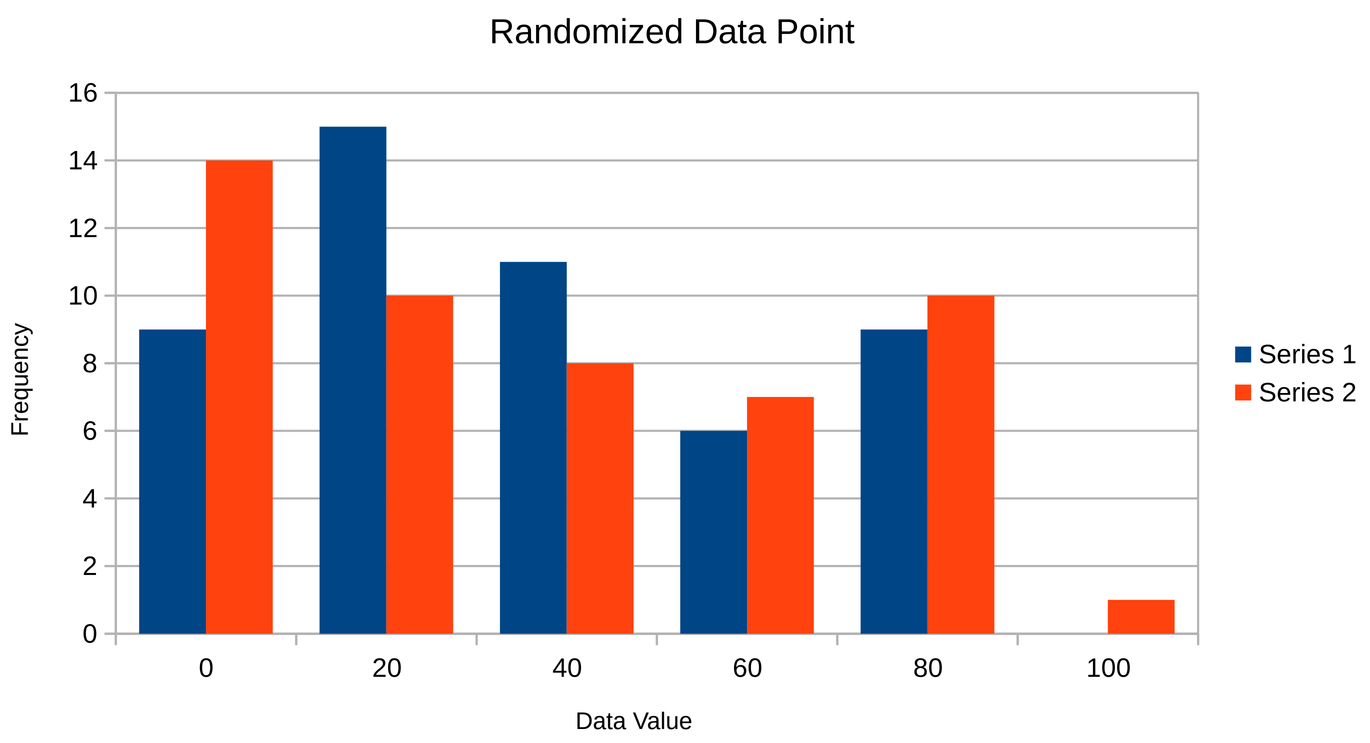

*Figure 13: Placeholder histogram*

## GGPLOT2

ggplot2 is an R package. This shows how to create both a scatterplot and histogram using RStudio. As a code-forward method, the following steps were taken within an R script to create a scatterplot (Figure 12) and a histogram (Figure 13).

```
# Load the tidyverse, which includes ggplot2

library('tidyverse')

names <- c("Datapoint", "Series1","Series2")

# Import CSV as tibble

a <-
read_csv("C:/Users/david/Documents/GitHub/Chart_Comparisons/Seeded_Values_for_Com
parison_Project.csv", col_names = names)

# Create the plot

my.formula <- y ~ x # Required for linear equation listing

a %>% pivot_longer(-Datapoint) %>%

  ggplot(aes(x = Datapoint, y = value, color = name, group = name)) +

  geom_point() +

  geom_smooth(method = lm, se = FALSE) +

  labs(title = 'Randomized Data Points', xlab = 'Datapoint', ylab = 'Datapoint
Value') +

  stat_poly_eq(formula = my.formula, aes(label =
paste(..eq.label.., ..rr.label.., sep = "~~~")), parse = TRUE)
```

The above code was created with input from Duck (Stackoverflow) to assist with debugging how to make the linear fit show up as desired.

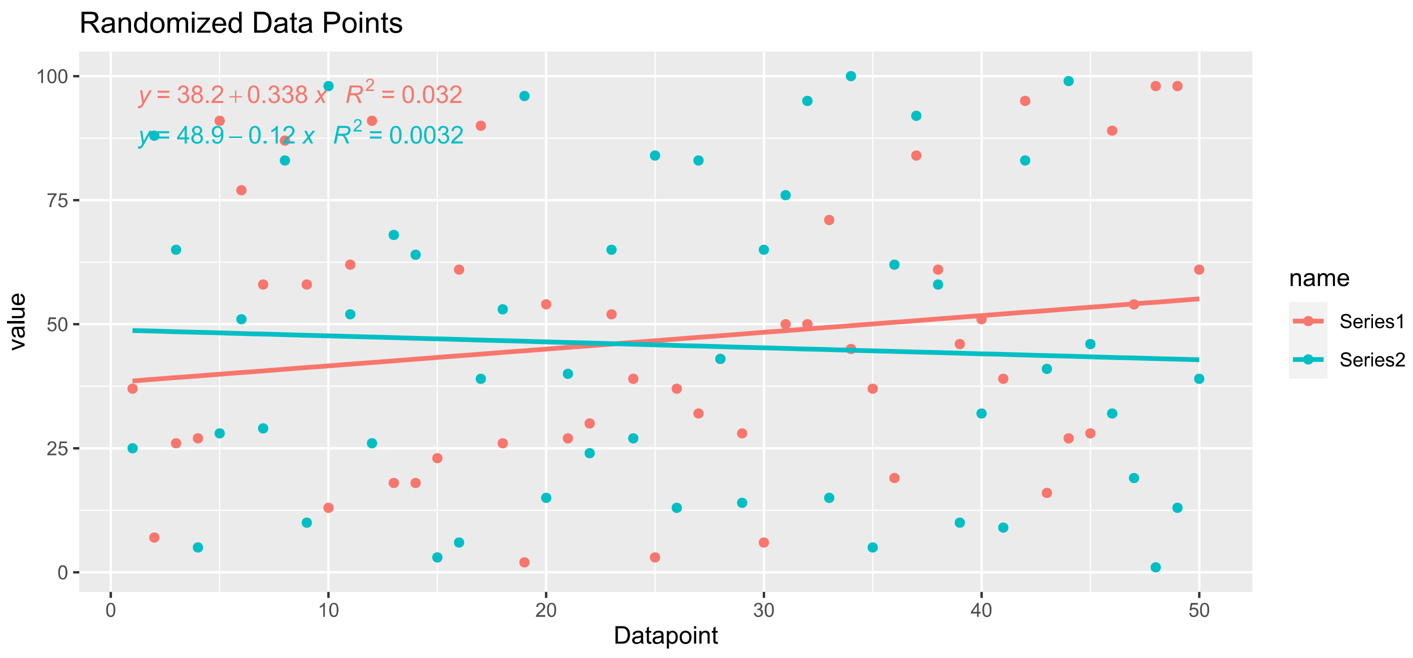

*Figure 14: ggplot2 Scatterplot*