NAME: Statistics and Distributions v. 1.02
AKA: stats_dist.avx
Last Modified: October 12, 2003

TITLE: Statistical Distributions and Summary Data

TOPICS: Statistics, Distributions, t, F, logistic, normal, skewness, kurtosis, binomial, probability, critical, Poisson, chi-square

AUTHOR: Jeff Jenness
GIS Analysis and Application Design
Jenness Enterprises, www.jennessent.com
jeffj@jennessent.com

DESCRIPTION: This extension gives you the ability to calculate a wider range of summary statistical data than the basic set provided by ESRI, and also the power to generate probability values and critical values from a wide range of statistical distributions.

Summary Statistics: From any numeric field in a table, this function will calculate the Mean, Standard Error of the Mean, Confidence Intervals, Minimum, 1st Quartile, Median, 3rd Quartile, Maximum, Variance, Standard Deviation, Average Absolute Deviation, Skewness (normal and Fisher's G1), Kurtosis (normal and Fisher's G2), Number of Records, Number of Null Values, and Total Sum.

Probability Calculators: This function will allow you to calculate the probability, cumulative probability and inverse probability (i.e. given a cumulative probability, calculate the corresponding critical value) of a wide range of statistical distributions, including the *Beta, Binomial, Cauchy, Chi-Square, Exponential, F, Logistic, LogNormal, Normal, Poisson, Student's T* and *Weibull* distributions. This function is available as a general calculator that remains open until you are finished with it, or as a Table tool that performs the calculations on all selected records in a table.

REQUIRES: This extension requires that the Dialog Designer DLL file be available in your ArcView/BIN32 directory (or $AVBIN/avdlog.dll), which it almost certainly is if you're running AV3.1 or higher. You don't have to load the Dialog Designer; it only has to be available. If you are running ArcView 3.0a, you can download the appropriate files for free from ESRI, at:

http://www.esri.com/software/arcview/extensions/dialog/index.html

UPDATES: The 1.01 update fixes a typo in the "Average Absolute Deviation" calculation which caused it not to be divided by N, and gives more programming functions to Avenue developers. Version 1.02 adds "Range" to the list of calculated statistics. The March 5 update makes no changes to the extension, but rather corrects some errors in the manual pertaining to generating summary statistics with Avenue. The October 12 update corrects some typos in the manual on formulas for the PDF and IDF Normal and LogNormal functions.

GENERAL INSTRUCTIONS:

1: Begin by placing the "stats_dist.avx" file into the ArcView extensions directory (../../Av_gis30/Arcview/ext32/).

2: After starting ArcView, load the extension by clicking on **File → Extensions… ,** scrolling down through the list of available extensions, and then clicking on the checkbox next to the extension called "Statistics and Distributions"

Table of Contents:

An on-line version of this manual may be viewed at:

http://www.jennessent.com/arcview/stats_dist.htm

Enjoy! Please contact the author if you have problems or find bugs.

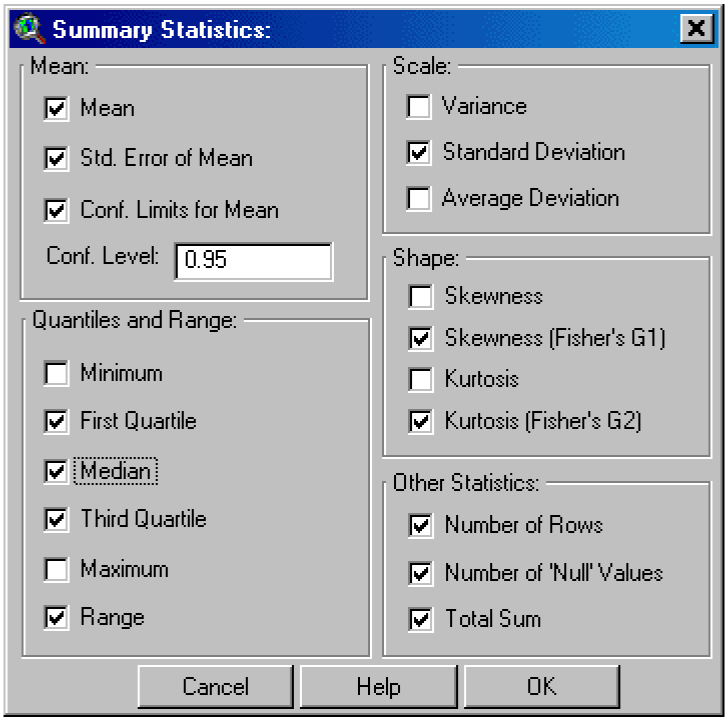
Jeff Jenness jeffj@jennessent.com
3020 N. Schevene Blvd.
Flagstaff, AZ 86004
USA

Please visit Jenness Enterprises ArcView Extensions site for more ArcView Extensions and other software by the author. We also offer customized ArcView-based GIS consultation services to help you meet your specific data analysis and application development needs.

Field Summary Statistics:

This function provides similar values to those produced by the "Summarize" function in the "Field" menu, except with a few more options and a higher level of precision. Open the "Summary Statistics" dialog by opening your table, selecting the field you are interested in, and then clicking on the $\Sigma\bar{x}$ button. This button will not be enabled unless a numeric field has been selected. Then, select your summary statistics and click "OK". The output will open in a Report window and the data will automatically be copied to the clipboard for you to paste into other applications.

This tool is designed with a dialog interface to make it easy for users to get and view a variety of statistics describing a set of data. This tool can also be accessed with Avenue code, which enables more advanced users to pass these statistics to variables, and then use the calculated values in other places. Please review Appendix A, Calculating Summary Statistics with Avenue, for details on how to do this.



Mean:
☑ Mean
☑ Std. Error of Mean
☑ Conf. Limits for Mean
Conf. Level: 0.95

Quantiles and Range:
☐ Minimum
☑ First Quartile
☑ Median
☑ Third Quartile
☐ Maximum
☑ Range

Scale:
☐ Variance
☑ Standard Deviation
☐ Average Deviation

Shape:
☐ Skewness
☑ Skewness (Fisher's G1)
☐ Kurtosis
☑ Kurtosis (Fisher's G2)

Other Statistics:
☑ Number of Rows
☑ Number of 'Null' Values
☑ Total Sum

1) Mean: Calculated as: $\dfrac{\sum x}{n}$

2) Std Error of the Mean: Calculated as: $s\{\bar{Y}\} = \dfrac{s}{\sqrt{n}}$, where s = sample standard deviation

3) Confidence Interval: The confidence limits for population mean μ with a confidence coefficient $(1 - \alpha)$, given a sample population mean $\bar{X}$, are calculated as:

$$\bar{X} \pm t_{\left(1-\alpha/2;n-1\right)} s\{\bar{Y}\},$$

where $s\{\bar{Y}\} = \frac{s}{\sqrt{n}}$, s = Sample Standard Deviation

and $t_{\left(1-\alpha/2;n-1\right)}$ = value from the t distribution at $p = \left(1-\alpha/2\right)$ and $n-1$ degrees of freedom.

4) Minimum: The lowest value in the data set.

5) Quartiles and Median: Those values at which at most (*P*)% of the data lie below the value and at most (1 – *P*)% of the data lie above the value. There are different ways to calculate quartile values, which produce similar but different results. Some methods draw the quartile values from the data set itself, so that the value called the "quartile" will always be found in the data. This script uses a different method which occasionally calculates a quartile value which represents the midpoint between two values from the data set, applying the following algorithm:

Assuming the data have been sorted from lowest to highest:

$$\text{Quartile 1 Index} = (N+1)\times 0.25 = Q(1)$$
$$\text{Quartile 2 Index} = (N+1)\times 0.50 = Q(2)$$
$$\text{Quartile 3 Index} = (N+1)\times 0.75 = Q(3)$$

If $Q(N)$ is an integer, then:

$$Quartile = Q(N)^{th}\ Value$$

If $Q(N)$ is not an integer, then:

R = that value at which $R < N < R+1$, and

$$Quartile = \frac{\left(Q(R)^{th}\ Value\right) + \left(Q(R+1)^{th}\ Value\right)}{2}$$

6) Maximum: The highest value in the data set.

7) Range: Maximum - Minimum

8) Variance: Calculated as: $\text{Variance} = \frac{\sum_{i=1}^{n}(y_i - \bar{y})^2}{n-1}$

9) Standard Deviation: Calculated as: $\text{Std. Deviation} = \sqrt{\frac{\sum_{i=1}^{n}(y_i - \bar{y})^2}{n-1}}$

10) Average Deviation: Calculated as: $\text{Avg. Deviation} = \frac{\sum_{i=1}^{n}\left|y_i - \bar{y}\right|}{n}$

11) Skewness: Measures the degree of asymmetry of the sample data around the mean.

$$\text{Skewness} = \frac{m_3}{m_2^{3/2}},$$

$$\text{where: } m_2 = 2^{\text{nd}} \text{ moment} = \frac{\sum_{i=1}^{n}(y_i - \bar{y})^2}{n}$$

$$\text{and: } m_3 = 3^{\text{rd}} \text{ moment} = \frac{\sum_{i=1}^{n}(y_i - \bar{y})^3}{n}$$

12) <u>Skewness (Fisher's G1):</u> There are alternative methods to calculate *skewness* measures of the data. S-PLUS uses the *Fisher's G1* variation, calculated as:

$$\text{Fisher's G1} = \frac{b_1\sqrt{n(n-1)}}{n-2}$$

$$\text{where: } b_1 = \frac{m_3}{m_2^{3/2}} \quad \text{(standard measure of skewness),}$$

$$\text{and: } m_2 = 2^{\text{nd}} \text{ moment} = \frac{\sum_{i=1}^{n}(y_i - \bar{y})^2}{n}$$

$$\text{and: } m_3 = 3^{\text{rd}} \text{ moment} = \frac{\sum_{i=1}^{n}(y_i - \bar{y})^3}{n}$$

13) <u>Kurtosis:</u> Measures the "peakedness" or "pointedness" in a distribution, and calculated as:

$$\text{Kurtosis} = \frac{m_4}{m_2^{2}},$$

$$\text{where: } m_2 = 2^{\text{nd}} \text{ moment} = \frac{\sum_{i=1}^{n}(y_i - \bar{y})^2}{n}$$

$$\text{and: } m_4 = 4^{\text{th}} \text{ moment} = \frac{\sum_{i=1}^{n}(y_i - \bar{y})^4}{n}$$

14) Kurtosis (Fisher's G2): As with *Skewness*, there are alternative ways to calculate kurtosis. S-PLUS uses the *Fisher's G2* variation, calculated as:

$$\text{Fisher's G2} = \frac{(n+1)(n-1)}{(n-2)(n-3)}\left[b_2 - \frac{3(n-1)}{n+1}\right]$$

$$\text{where: } b_2 = \frac{m_4}{m_2^{\,2}} \quad \text{(standard measure of kurtosis),}$$

$$\text{and: } m_2 = 2^{\text{nd}} \text{ moment} = \frac{\sum_{i=1}^{n}(y_i - \bar{y})^2}{n}$$

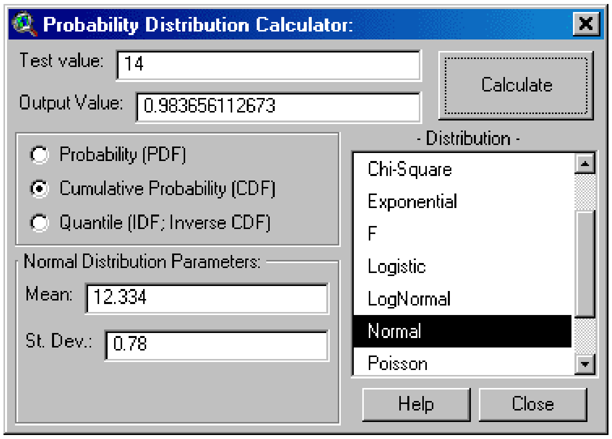
$$\text{and: } m_4 = 4^{\text{th}} \text{ moment} = \frac{\sum_{i=1}^{n}(y_i - \bar{y})^4}{n}$$

15) Number of Rows: The total number of rows of data examined during the analysis.

16) Number of 'Null' Values: The total number of cases of missing data. These are represented as "null" numbers in the table, which are different than zeros.

17) Total Sum: Sum of all non-null values, calculated as: $\sum x$

Probability Distribution Calculators:


Probability Distribution Calculator:

Test value: 14

Output Value: 0.983656112673

○ Probability (PDF)
● Cumulative Probability (CDF)
○ Quantile (IDF; Inverse CDF)

Normal Distribution Parameters:

Mean: 12.334

St. Dev.: 0.78

Distribution: Normal

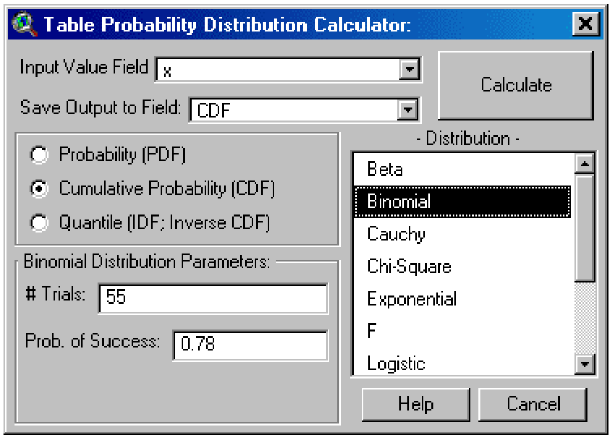


Table Probability Distribution Calculator:

Input Value Field: x

Save Output to Field: CDF

○ Probability (PDF)
● Cumulative Probability (CDF)
○ Quantile (IDF; Inverse CDF)

Binomial Distribution Parameters:

Trials: 55

Prob. of Success: 0.78

Distribution: Binomial

This extension includes two versions of a Probability Distribution Calculator, each of which calculate distribution data based on a variety of distributions and parameters. The **Probability Distribution Calculator** is started from within a View, and is opened by clicking on the button in the View toolbar. You simply enter the input and parameter values, specify whether you are calculating Probability, Cumulative Probability or Quantile values, and click "Calculate", and the result appears in the "Output Value" window. This calculator stays open until you close it and you can leave it open as you do other things in ArcView.

The "Table Probability Distribution Calculator" is designed to work on all selected records in a table, applying the distribution parameters to each value and saving the results to a field in that table. This calculator is opened from within a Table by clicking on the button in the Table toolbar. Select the field containing the "Input" values, then decide whether to create a new field or use an existing field to save the "Output" values, then click "Calculate" to generate distribution values for all selected records. The window stays open until you click "Calculate" or "Cancel".

The Distribution functions included with this extension may be grouped in 3 categories. In general, the *Probability Density Functions* return the probability that the Test Value = X given that particular distribution. The *Cumulative Distribution Functions* return the probability that the Test Value $\leq X$, given that particular distribution. The *Quantile Functions* (sometimes referred to as *Inverse Density Functions* or *Percent Point Functions*) return the Value X at which $P(X)$ = [specified probability], given that particular distribution.

Functions and Probability Distributions

Distribution	Probability Density Function	Cumulative Distribution Function	Quantile Function
Beta	PDF_Beta	CDF_Beta	IDF_Beta
Binomial	PDF_Binomial	CDF_Binomial	IDF_Binomial
Cauchy	PDF_Cauchy	CDF_Cauchy	IDF_Cauchy
Chi-Square	PDF_ChiSquare	CDF_ChiSquare	IDF_ChiSquare

Exponential	PDF_Exp	CDF_Exp	IDF_Exp
F	PDF_F	CDF_F	IDF_F
Logistic	PDF_Logistic	CDF_Logistic	IDF_Logistic
LogNormal	PDF_LogNormal	CDF_LogNormal	IDF_LogNormal
Normal	PDF_Normal_Simpsons	CDF_Normal	IDF_Normal
Poisson	PDF_Poisson	CDF_Poisson	IDF_Poisson
Student's T	PDF_StudentsT	CDF_StudentsT	IDF_StudentsT
Weibull	PDF_Weibull	CDF_Weibull	IDF_Weibull

Equations for each function are included in the Appendix B: Distribution Functions, Parameters and Usages, but some them do not have closed formulas that can be calculated and therefore must be computed numerically. The author refers interested persons to the references to find source code and computational methods of calculating these functions. The author especially recommends Croarkin & Tobias (date unknown) and McLaughlin (2001) for illustrations of the various distributions, and Press et al. (1988-1997) and Burkardt (2001) for computational methods. These sources are all available on-line.

The descriptions in Appendix B include 4 methods of utilizing each function. The first method describes how to use the *Probability Distribution Calculators* to calculate values. There are three additional methods available for programmers who want to access the functions through *Avenue* code. Simply copy the line of code exactly as written, substituting your parameter variable names in the proper places.

Avenue Functions:

1) The first *Avenue* option sends your parameters to a central script called "Jennessent.DistFunc", which checks for possible errors in the parameters (using a negative value for Degrees of Freedom, for example). If the script finds such values, it will halt operation and alert you to the problem. If it doesn't find errors, it forwards your parameters to the appropriate script and returns the result to you. Users may want to review the script "Jennessent.ProbDlogCalculate" for an example of this option. IMPORTANT: Users should also be aware that this script only checks to see if the input values follow the rules described in Appendix B. It doesn't check for basic errors such as sending a non-numeric value to the script.

2) The second *Avenue* option is very similar to the first in that it sends your parameters to a central script to check for errors ("Jennessent.TableDistFunc" in this case), but it doesn't halt operation if it finds an error. Rather, it returns an error message (in *String* format) detailing the problem. The author recommends this option for cases in which the user wants to conduct calculations on a series of values (i.e. records in a table), but doesn't want the function to stop if it finds an illegal value (possibly a record with no data). This option would allow the user to insert an "if-then" statement in their code to check if the result is a string or a number. Numerical responses would indicate successful calculations while String responses could be appended to a running report of unsuccessful

calculations. Users may want to review the script “Jennessent.ProbTabDlogCalculate” for an example of this option.

3) If you’d like to skip the error-checking routines, use the third *Avenue* option to send your parameters to the relevant script directly.

Appendix A: Calculating Summary Statistics with Avenue

The Summary Statistics tool collects a series of True/False and Numerical parameters from the user and sends them to a script called “Jennessent.CalcFieldStats”, which does the necessary calculations and returns a list of results. The tool then prints those results up in a Report window for the user.

Avenue programmers can bypass the dialog and send values to the script directly if they wish, and then they will have the desired statistics directly available to them in a list. For example, many statistical calculations require such things as means, standard deviations, variances, quartiles, etc. The user may want to generate these values early in a script and then use them in later calculations. The “Jennessent.CalcFieldStats” script makes it simple to generate such values from data in a table.

This option is a little simpler than the standard Avenue method for generating statistics, which is to create a new file on the hard drive and then use the “Summarize” request to save statistics to that file. It also offers a larger variety of statistical output, including such things as Confidence Intervals, Standard Error of the Mean, Average Deviation, and Kurtosis/Skewness values. This option is also a little slower on large datasets, however, and it doesn’t divide up the dataset into subsets like “Summarize” does.

The function can be used with just a few lines of code:

```
ListOfResults = av.Run(“Jennessent.CalcFieldStats”, {ListOfInputParameters,
      theVTab, theField})
```

The object “`theVTab`” is a VTab object containing your data, and “`theField`” is a Field object in the VTab, reflecting the field you want to calculate statistics on.

The “`ListOfInputParameters`” must contain 20 values, most of which are Boolean (true/false) reflecting whether you want that particular statistic calculated:

```
ListOfInputParameters = {CalcMean, CalcSEMean, CalcConInt, Con_Level,
   CalcMinimum, Calc1stQuart, CalcMedian, Calc3rdQuart, CalcMaximum,
   CalcVariance, CalcStandDev, CalcAvgDev, CalcSkewness, CalcSkewFish,
   CalcKurtosis, CalcKurtFish, CalcCount, CalcNumNull, CalcSum, CalcRange}
```

Where:

`CalcMean`: **Boolean**, *True* if you want to calculate the mean.
`CalcSEMean`: **Boolean**, *True* if you want to calculate the standard error of the mean.
`CalcConInt`: **Boolean**, *True* if you want to calculate confidence intervals of the mean.
`Con_Level`: **Number**, $0 \le p \le 1$, where p = probability = $(1 - \alpha)$
`CalcMinimum`: **Boolean**, *True* if you want to calculate the minimum value.
`Calc1stQuart`: **Boolean**, *True* if you want to calculate the 1st quartile.
`CalcMedian`: **Boolean**, *True* if you want to calculate the median.
`Calc3rdQuart`: **Boolean**, *True* if you want to calculate the 3rd quartile.
`CalcMaximum`: **Boolean**, *True* if you want to calculate the maximum value.
`CalcVariance`: **Boolean**, *True* if you want to calculate the variance.
`CalcStandDev`: **Boolean**, *True* if you want to calculate the standard deviation.
`CalcAvgDev`: **Boolean**, *True* if you want to calculate the absolute average deviation.
`CalcSkewness`: **Boolean**, *True* if you want to calculate the standard skewness.

`CalcSkewFish:` **Boolean**, *True* if you want to calculate the Fisher’s G1 skewness.
`CalcKurtosis:` **Boolean**, *True* if you want to calculate the standard kurtosis.
`CalcKurtFish:` **Boolean**, *True* if you want to calculate the Fisher’s G2 kurtosis.
`CalcCount:` **Boolean**, *True* if you want to calculate the total number of rows of data.
`CalcNumNull:` **Boolean**, *True* if you want to calculate the number of null values.
`CalcSum:` **Boolean**, *True* if you want to calculate the sum.
`CalcRange:` **Boolean**, *True* if you want to calculate the Range.

When the script finishes, it will return a list of 18 values to you representing the various statistics you requested. If you did not request a particular statistic, then it will not be calculated and the return list will contain a “nil” object in it’s place. Note that if you requested a confidence interval, the upper and lower levels are returned as a separate list (3rd object in the Return List).

Return list:
```
{Mean, Standard Error of Mean, {Lower Confidence Level,
 Upper Confidence Level}, Minimum, 1st Quartile, Median, 3rd
 Quartile, Maximum, Variance, Standard Deviation, Skewness,
 Fisher’s GI Skewness, Kurtosis, Fisher’s G2 Kurtosis,
 Record Count, Number of Null Values, Sum, Range}
```

For example: If you had a table of population demographic data containing a field of Annual Income values, and you were interested in the mean annual income plus a 95% confidence interval around that mean, then you would set up the code as follows:

```
theDemographyVTab = YourTable.GetVTab

theField = theDemographyVTab.FindField(“Income”)

theInputParameters = {True, False, True, 0.95, False, False, False, False,
      False, False, False, False, False, False, False, False, False, False,
      False, False}

theReturnList = av.Run(“Jennessent.CalcFieldStats”, {theInputParameters,
      theDemographyVTab , theField})

theMeanIncome = theReturnList.Get(0)

theLowerConfidenceLimit = theReturnList.Get(2).Get(0)

theUpperConfidenceLimit = theReturnList.Get(2).Get(1)
```

All the objects in “`theReturnList`” will be “nil” objects except for the ones at indices 0 and 2. The Mean will be at index 0, the Lower 95% Confidence Limit will be the first item in index 2, and the Upper 95% Confidence Limit will be the second item in index 2.

In general, all the possible statistics can be obtained with the following lines of code. Simply copy and paste the appropriate lines into your script:

```
theMean = theReturnList.Get(0)

theSEMean = theReturnList.Get(1)

if (Calculating_Confidence_Intervals)

      LowerCI = theReturnList.Get(2).Get(0)

      UpperCI = theReturnList.Get(2).Get(1)

end

theMinimum = theReturnList.Get(3)

theQ1 = theReturnList.Get(4)
```

```
theMedian = theReturnList.Get(5)
theQ3 = theReturnList.Get(6)
theMaximum = theReturnList.Get(7)
theVar = theReturnList.Get(8)
theStdDev = theReturnList.Get(9)
theAvgDev = theReturnList.Get(10)
theSkew = theReturnList.Get(11)
theFisherSkew = theReturnList.Get(12)
theKurt = theReturnList.Get(13)
theFisherKurt = theReturnList.Get(14)
theCount = theReturnList.Get(15)
theNumberNull = theReturnList.Get(16)
theSum = theReturnList.Get(17)
theRange = theReturnList.Get(18)
```

Appendix B: Functions, Parameters and Usages

Probability Density Functions:

1. **PDF_Beta:** This function returns the probability that the Test Value = *X*, assuming a Beta distribution and the specified Shape parameters. This is the standardized Beta function, where Location = 0 and Scale (upper bound) = 1. According to McLaughlin (2001), parameters *Shape1* and *Shape2* can be any positive value, but they rarely exceed 10. The function becomes nearly flat if the values get much larger than this.
   a) Parameters:
      i) Test Value: Number
      ii) Shape1: Number > 0
      iii) Shape2: Number > 0
   b) Usages:
      i) From “Probability Distribution Calculator”, select “Probability (PDF)” and *Beta* distribution.
      ii) (*Avenue*): theProb = av.Run("Jennessent.DistFunc", {“PDF_Beta”, {Test Value, Shape1, Shape2}})
      iii) (*Avenue*): theProb = av.Run("Jennessent.TableDistFunc", {“PDF_Beta”, {Test Value, Shape1, Shape2}})
      iv) (*Avenue*): theProb = av.Run(“Jennessent.PDF_Beta”, {Test Value, Shape1, Shape2})
   c) Function:

$$\text{Beta PDF} = \frac{\Gamma(S_1 + S_2)}{\Gamma(S_1)\Gamma(S_2)} y^{S_1 - 1}(1 - y)^{S_2 - 1}$$

where: y = Test Value, S_1 = Shape 1, S_2 = Shape 2

and: $\Gamma(x) = \int_0^\infty t^{x-1} e^{-1} dt$

2. **PDF_Binomial:** The Binomial distribution is used when there are exactly two mutually exclusive outcomes of a trial. This function returns the probability of getting *X* successes out of *N* trials, given a probability of success = *P*.
   a) Parameters:
      i) # Successes: Integer ≥ 0
      ii) # Trials: Integer ≥ 2, # Successes
      iii) Probability of Success: Number ($0 \geq p \geq 1$)
   b) Usages:
      i) From "Probability Distribution Calculator", select "Probability (PDF)" and *Binomial* distribution.
      ii) (*Avenue*): theProb = av.Run("Jennessent.DistFunc", {“PDF_Binomial”, {#Success, #Trials, Probability of Success}})
      iii) (*Avenue*): theProb = av.Run("Jennessent.TableDistFunc", {“PDF_Binomial”, {#Success, #Trials, Probability of Success}})

iv) (*Avenue*): theProb = av.Run(“Jennessent.PDF_Binomial”, {#Success, #Trials, Probability or Success}})

c) Function: $\text{Binomial PDF} = \left(\frac{B!}{y!(B-y)!}\right)A^{y}(1-A)^{B-y}$

where: y = #Successes, A = Probability of Success, B = #Trials

3. **PDF_Cauchy:** This function returns the probability that the Test Value = *X*, assuming a *Cauchy* distribution with the specified mean and standard deviation. The *Standardized Cauchy distribution* is that with *Location = 0* and *Scale = 1.*
   a) Parameters:
      i) Test Value: Number
      ii) Location: Number
      iii) Scale: Number > 0
   b) Usages:
      i) From "Probability Distribution Calculator", select "Probability (PDF)" and *Cauchy* distribution.
      ii) (*Avenue*): theProb = av.Run("Jennessent.DistFunc”, {“PDF_Cauchy”, {Test Value, Location, Scale}})
      iii) (*Avenue*): theProb = av.Run("Jennessent.TableDistFunc”, {“PDF_Cauchy”, {Test Value, Location, Scale}})
      iv) *(Avenue*): theProb = av.Run("Jennessent.PDF_Cauchy", {Test Value, Location, Scale})

   c) Function: $\text{Cauchy PDF} = \frac{1}{\pi B\left[1+\left(\frac{y-A}{B}\right)^{2}\right]}$

   where: y = Test Value, A = Location, B = Scale

4. **PDF_ChiSquare:** This function returns the probability that the Test Value = *X*, assuming a *Chi-Square* distribution with the specified Degrees of Freedom. The *Chi-Square* distribution results when *v* (where *v* = Degrees of Freedom) independent variables with standard normal distributions are squared and summed (Croarkin & Tobias, Date unknown).
   a) Parameters:
      i) Test Value: Number ≥ 0
      ii) Degrees of Freedom: Number > 0
   b) Usages:
      i) From "Probability Distribution Calculator", select "Probability (PDF)" and *Chi-Square* distribution.
      ii) (*Avenue*): theProb = av.Run("Jennessent.DistFunc”, {“PDF_ChiSquare”, {Test Value, DF}})
      iii) (*Avenue*): theProb = av.Run("Jennessent.TableDistFunc”, {“PDF_ChiSquare”, {Test Value, DF}})
      iv) *(Avenue*): theProb = av.Run("Jennessent.PDF_ChiSquare", {Test Value, DF})

c) Function: $$\text{Chi-Square PDF} = \frac{e^{\frac{-y}{2}} x^{\frac{v}{2}-1}}{2^{\frac{v}{2}} \Gamma\left(\frac{v}{2}\right)}$$

where: y = Test Value, S_1 = Shape 1, S_2 = Shape 2

and: $\Gamma(x) = \int_0^\infty t^{x-1} e^{-1} dt$

5. **PDF_Exp:** This function returns the probability that the Test Value = *X*, assuming an *Exponential* distribution with the specified mean. This script uses the 1-parameter version of the equation (i.e. assuming *Location* = 0). The *Standard Exponential Distribution* is that which has *Mean = 1*.
   a) Parameters:
      i) Test Value: Number ≥ 0
      ii) Mean: Number > 0
   b) Usages:
      i) From "Probability Distribution Calculator", select "Probability (PDF)" and *Exponential* distribution.
      ii) (*Avenue*): theProb = av.Run("Jennessent.DistFunc", {"PDF_Exp", {Test Value, Mean}})
      iii) (*Avenue*): theProb = av.Run("Jennessent.TableDistFunc", {"PDF_Exp", {Test Value, Mean}})
      iv) *(Avenue*): theProb = av.Run("Jennessent.PDF_Exp", {Test Value, Mean})

   c) Function: $\text{Exponential PDF} = \frac{1}{\beta} e^{-\frac{x}{\beta}}$

   where: x = Test Value, β = Mean (or Scale Parameter)

6. **PDF_F:** This function returns the probability that the Test Value = *X*, assuming an *F* distribution with the specified Degrees of Freedom. The *F* distribution is the ratio of two *Chi-Square* distributions with ratios v_1 and v_2 respectively.
   a) Parameters:
      i) Test Value: Number ≥ 1
      ii) 1st Degrees of Freedom: Number > 1
      iii) 2nd Degrees of Freedom: Number > 1
   b) Usages:
      i) From "Probability Distribution Calculator", select "Probability (PDF)" and *F* distribution.
      ii) (*Avenue*): theProb = av.Run("Jennessent.DistFunc", {"PDF_F", {Test Value, DF1, DF2}})
      iii) (*Avenue*): theProb = av.Run("Jennessent.TableDistFunc", {"PDF_F", {Test Value, DF1, DF2}})
      iv) *(Avenue*): theProb = av.Run("Jennessent.PDF_F", {Test Value, DF1, DF2})

c) Function: $$\text{F PDF} = \frac{\Gamma\left(\frac{v_1+v_2}{2}\right)\left(\frac{v_1}{v_2}\right)^{\frac{v_1}{2}} x^{\frac{v_1}{2}-1}}{\Gamma\left(\frac{v_1}{2}\right)\Gamma\left(\frac{v_1}{2}\right)\left(1+\frac{v_1 x}{v_2}\right)^{\frac{v_1+v_2}{2}}}$$

where: x = Test Value, v_1 = DF1, v_2 = DF2

and: $\Gamma(x) = \int_0^\infty t^{x-1}e^{-1}dt$

7. **PDF_Logistic:** This function returns the probability that the Test Value = *X*, assuming a *Logistic* distribution with the specified mean and scale.
   a) Parameters:
      i) Test Value: Number
      ii) Mean: Number
      iii) Scale: Number > 0
   b) Usages:
      i) From "Probability Distribution Calculator", select "Probability (PDF)" and *Logistic* distribution.
      ii) (*Avenue*): theProb = av.Run("Jennessent.DistFunc", {"PDF_Logistic", {Test Value, Mean, Scale}})
      iii) (*Avenue*): theProb = av.Run("Jennessent.TableDistFunc", {"PDF_Logistic", {Test Value, Mean, Scale}})
      iv) *(Avenue*): theProb = av.Run("Jennessent.PDF_Logistic", {Test Value, Mean, Scale})

   c) Function: $$\text{Logistic PDF} = \frac{1}{B}\frac{\exp\left(\frac{y-A}{B}\right)}{\left[1+\exp\left(\frac{y-A}{B}\right)\right]^2}$$

   where: y = Test Value, A = Mean, B = Scale

8. **PDF_LogNormal:** This function returns the probability that the Test Value = *X*, assuming a *LogNormal* distribution with the specified mean and scale. A *LogNormal* distribution occurs when natural logarithms of variable *X* are normally distributed. The *Standard LogNormal Distribution* is that with Mean = 0 and Scale = 1. The *2-Parameter LogNormal Distribution* is that with Mean = 0.
   a) Parameters:
      i) Test Value: Number ≥ 0
      ii) Mean: Number > 0
      iii) Scale: Number > 0
   b) Usages:

   - i) From "Probability Distribution Calculator", select "Probability (PDF)" and *LogNormal* distribution.
   - ii) (*Avenue*): theProb = av.Run("Jennessent.DistFunc", {"PDF_LogNormal", {Test Value, Mean, Scale}})
   - iii) (*Avenue*): theProb = av.Run("Jennessent.TableDistFunc", {"PDF_LogNormal", {Test Value, Mean, Scale}})
   - iv) *(Avenue*): theProb = av.Run("Jennessent.PDF_LogNormal", {Test Value, Mean, Scale})

c) Function: $\text{LogNormal PDF} = \frac{1}{B\sqrt{2\pi}} \exp\left(-\frac{1}{2}\left(\frac{\ln(y)-A}{B}\right)^2\right)$

where: y = Test Value, A = Mean, B = Scale

9. PDF_Normal: This function returns the probability that the Test Value = *X*, assuming a *Normal* distribution with the specified mean and standard deviation. The *Standard Normal Distribution* is that with Mean = 0 and Standard Deviation = 1.

a) Parameters:
   - i) Test Value: Number
   - ii) Mean: Number
   - iii) Standard Deviation: Number > 0

b) Usages:
   - i) From "Probability Distribution Calculator", select "Probability (PDF)" and *Normal* distribution.
   - ii) (*Avenue*): theProb = av.Run("Jennessent.DistFunc", {"PDF_Normal", {Test Value, Mean, St. Dev.}})
   - iii) (*Avenue*): theProb = av.Run("Jennessent.TableDistFunc", {"PDF_Normal", {Test Value, Mean, St. Dev.}})
   - iv) *(Avenue*): theProb = av.Run("Jennessent.PDF_Normal", {Test Value, Mean, St. Dev.})

c) Function: $\text{Normal PDF} = \frac{1}{B\sqrt{2\pi}} \exp\left(-\frac{1}{2}\left(\frac{y-A}{B}\right)^2\right)$

where: y = Test Value, A = Mean, B = Scale

10. PDF_Poisson: This function returns the probability that the Specified Number of Events = *X*, assuming a *Poisson* distribution with the specified mean.

a) Parameters:
   - i) # Events: Integer ≥ 0
   - ii) Mean: Number > 0

b) Usages:
   - i) From "Probability Distribution Calculator", select "Probability (PDF)" and *Poisson* distribution.

        ii) (*Avenue*): theProb = av.Run("Jennessent.DistFunc”, {“PDF_Poisson”, {# Events, Mean}})
        iii) (*Avenue*): theProb = av.Run("Jennessent.TableDistFunc”, {“PDF_Poisson”, {# Events, Mean}})
        iv) *(Avenue*): theProb = av.Run("Jennessent.PDF_Poisson", {# Events, Mean})

    c) Function: Poisson PDF $= \dfrac{\exp^{-A} A^{y}}{y!}$

where: y = Test value, A = Expectation (mean)

11. PDF_StudentsT: This function returns the probability that the Test Value = *X*, assuming a *Students T* distribution with the specified Degrees of Freedom. A *Student’s T* distribution with 1df is a *Cauchy* Distribution, and it approaches a *Normal* distribution when DF>30. Various sources recommend using the *Normal* distribution if DF>100.

    a) Parameters:
        i) Test Value: Number
        ii) Degrees of Freedom: Number > 0
    b) Usages:
        i) From "Probability Distribution Calculator", select "Probability (PDF)" and *Student’s T* distribution.
        ii) (*Avenue*): theProb = av.Run("Jennessent.DistFunc”, {“PDF_StudentsT”, {Test Value, DF}})
        iii) (*Avenue*): theProb = av.Run("Jennessent.TableDistFunc”, {“PDF_StudentsT”, {Test Value, DF}})
        iv) *(Avenue*): theProb = av.Run("Jennessent.PDF_StudentsT", {Test Value, DF})

    c) Function:

$$\text{Student's T PDF} = \frac{\Gamma\left(\dfrac{v+1}{2}\right)}{\sqrt{\pi v}\,\Gamma\left(\dfrac{v}{2}\right)}\left[1+\frac{y^{2}}{v}\right]^{-\frac{v+1}{2}}$$

where: y = Test Value, v = Degrees of Freedom

and: $\Gamma(x) = \int_{0}^{\infty} t^{x-1} e^{-1} dt$

12. PDF_Weibull: This function returns the probability that the Test Value = *X*, assuming a *Weibull* distribution with the specified Location, Scale and Shape parameters. The *Standardized Weibull Distribution* is that with *Location = 0* and *Scale = 1.* The *2-Parameter Weibull Distribution* is that with *Location = 0.*

    a) Parameters:
        i) Test Value: Number > Location
        ii) Location: Number
        iii) Scale: Number > 0
        iv) Shape: Number > 0
    b) Usages:

    i) From "Probability Distribution Calculator", select "Probability (PDF)" and *Weibull* distribution.
    ii) (*Avenue*): theProb = av.Run("Jennessent.DistFunc", {"PDF_Weibull", {Test Value, Location, Scale, Number}})
    iii) (*Avenue*): theProb = av.Run("Jennessent.TableDistFunc", {"PDF_Weibull", {Test Value, Location, Scale, Number}})
    iv) (*Avenue*): theProb = av.Run("Jennessent.PDF_Weibull", {Test Value, Location, Scale, Number})

c) Function: Weibull PDF $= \left(\frac{C}{B}\right)\left(\frac{y-A}{B}\right)^{C-1} \exp^{\left(-\left(\frac{y-A}{b}\right)^{2}\right)}$

where: y = Test Value, A = Location, B = Scale, C = Shape

Cumulative Distribution Functions:

1. **CDF_Beta:** This function returns the probability that the Test Value ≤ *X*, assuming a Beta distribution with the specified Shape parameters. This is the *Standardized Beta function*, where Location = 0 and Scale (upper bound) = 1. According to McLaughlin (2001), parameters *Shape1* and *Shape2* can be any positive value, but they rarely exceed 10. The function becomes nearly flat if the values get much larger than this.
    a) Parameters:
        i) Test Value: Number
        ii) Shape1: Number > 0
        iii) Shape2: Number > 0
    b) Usages:
        i) From "Probability Distribution Calculator", select "Cumulative Probability (CDF)" and *Beta* distribution.
        ii) (*Avenue*): theProb = av.Run("Jennessent.DistFunc", {"CDF_Beta", {Test Value, Shape1, Shape2}})
        iii) (*Avenue*): theProb = av.Run("Jennessent.TableDistFunc", {"CDF_Beta", {Test Value, Shape1, Shape2}})
        iv) (*Avenue*): theProb = av.Run("Jennessent.CDF_Beta", {Test Value, Shape1, Shape2})

Beta CDF $= I\left(y, S_1, S_2\right)$ (From Press et al, 1997)

c) Function: where: y = Test Value, S_1 = Shape 1, S_2 = Shape 2

and: $I\left(x,a,b\right) \equiv \frac{B_x\left(a,b\right)}{B\left(a,b\right)} \equiv \frac{1}{B\left(a,b\right)} \int_0^x t^{a-1}(1-t)^{b-1}\,dt$

and: $B\left(a,b\right) = \int_0^1 t^{a-1}(1-t)^{b-1}\,dt$

2. **CDF_Binomial:** The *Binomial* distribution is used when there are exactly two mutually exclusive outcomes of a trial. This function returns the probability of getting ≤ *X* successes out of *N* trials, given a probability of success= *P*.
   a) Parameters:
      i) # Successes: Integer ≥ 0
      ii) # Trials: Integer ≥ 2, # Successes
      iii) Probability of Success: Number ($0 \geq p \geq 1$)
   b) Usages:
      i) From "Probability Distribution Calculator", select "Cumulative Probability (CDF)" and *Binomial* distribution.
      ii) (*Avenue*): theProb = av.Run("Jennessent.DistFunc", {"CDF_Binomial", {#Success, #Trials, Probability of Success}})
      iii) (*Avenue*): theProb = av.Run("Jennessent.TableDistFunc", {"CDF_Binomial", {#Success, #Trials, Probability of Success}})
      iv) *(Avenue*): theProb = av.Run("Jennessent.CDF_Binomial", {#Success, #Trials, Probability of Success})

   c) Function: $\text{Binomial CDF} = \sum_{i=1}^{y} \left(\frac{B!}{i!(B-i)!} \right) A^i (1-A)^{B-i}$

      where: y = #Successes, A = Probability of Success, B = #Trials

3. **CDF_Cauchy:** This function returns the probability that the Test Value ≤ *X*, assuming a *Cauchy* distribution with the specified Location and Scale parameters. The *Standardized Cauchy distribution* has *Location = 0* and *Scale = 1.*
   a) Parameters:
      i) Test Value: Number
      ii) Location: Number
      iii) Scale: Number > 0
   b) Usages:
      i) From "Probability Distribution Calculator", select "Cumulative Probability (CDF)" and *Cauchy* distribution.
      ii) (*Avenue*): theProb = av.Run("Jennessent.DistFunc", {"CDF_Cauchy", {Test Value, Location, Scale}})
      iii) (*Avenue*): theProb = av.Run("Jennessent.TableDistFunc", {"CDF_Cauchy", {Test Value, Location, Scale}})
      iv) *(Avenue*): theProb = av.Run("Jennessent.CDF_Cauchy", {Test Value, Location, Scale})

   c) Function: $\text{Cauchy CDF} = \frac{1}{2} + \frac{1}{\pi} \tan^{-1} \left(\frac{y-A}{B} \right)$

      where: y = Test Value, A = Location, B = Scale

4. **CDF_ChiSquare:** This function returns the probability that the Test Value ≤ *X*, assuming a *Chi-Square* distribution with the specified Degrees of Freedom. The *Chi-Square* distribution

results when *v* (where *v* = Degrees of Freedom) independent variables with standard normal distributions are squared and summed (Croarkin & Tobias, Date unknown).

a) Parameters:
   - i) Test Value: Number ≥ 0
   - ii) Degrees of Freedom: Number > 0

b) Usages:
   - i) From "Probability Distribution Calculator", select "Cumulative Probability (CDF)" and *Chi-Square* distribution.
   - ii) (*Avenue*): theProb = av.Run("Jennessent.DistFunc”, {“CDF_ChiSquare”, {Test Value, DF}})
   - iii) (*Avenue*): theProb = av.Run("Jennessent.TableDistFunc”, {“CDF_ChiSquare”, {Test Value, DF}})
   - iv) *(Avenue*): theProb = av.Run("Jennessent.CDF_ChiSquare", {Test Value, DF})

$$\text{Chi-Square CDF} = \frac{\gamma\left(\frac{v}{2}, \frac{x}{2}\right)}{\Gamma\left(\frac{v}{2}\right)}$$

c) Function: where: y = Test Value, S_1 = Shape 1, S_2 = Shape 2

and: $\Gamma(x) = \int_0^\infty t^{x-1} e^{-1} dt$

and: $\gamma(x,y) = \int_0^y t^{x-1} e^{-1} dt$

5. **CDF_Exp:** This function returns the probability that the Test Value ≤ *X*, assuming an *Exponential* distribution with the specified mean. This script uses the 1-parameter version of the equation (i.e. assuming *Location* = 0). The *Standard Exponential Distribution* is that which has *Mean = 1.*

a) Parameters:
   - i) Test Value: Number ≥ 0
   - ii) Mean: Number > 0

b) Usages:
   - i) From "Probability Distribution Calculator", select "Cumulative Probability (CDF)" and *Exponential* distribution.
   - ii) (*Avenue*): theProb = av.Run("Jennessent.DistFunc”, {“CDF_Exp”, {Test Value, Mean}})
   - iii) (*Avenue*): theProb = av.Run("Jennessent.TableDistFunc”, {“CDF_Exp”, {Test Value, Mean}})
   - iv) *(Avenue*): theProb = av.Run("Jennessent.CDF_Exp", {Test Value, Mean})

c) Function: $\text{Exponential CDF} = 1 - e^{-\frac{x}{\beta}}$

where: x = Test value, β = Mean (or Scale Parameter)

6. **CDF_F:** This function returns the probability that the Test Value ≤ *X*, assuming an *F* distribution with the specified Degrees of Freedom. The *F* distribution is the ratio of two *Chi-Square* distributions with ratios v_1 and v_2 respectively.
   a) Parameters:
      i) Test Value: Number ≥ 1
      ii) 1st Degrees of Freedom: Number > 1
      iii) 2nd Degrees of Freedom: Number > 1
   b) Usages:
      i) From "Probability Distribution Calculator", select "Cumulative Probability (CDF)" and *F* distribution.
      ii) (*Avenue*): theProb = av.Run("Jennessent.DistFunc", {"CDF_F", {Test Value, DF1, DF2}})
      iii) (*Avenue*): theProb = av.Run("Jennessent.TableDistFunc", {"CDF_F", {Test Value, DF1, DF2}})
      iv) (*Avenue*): theProb = av.Run("Jennessent.CDF_F", {Test Value, DF1, DF2})

$$\text{F CDF} = 1 - I\left(k, \frac{v_1}{2}, \frac{v_2}{2}\right)$$

where: $k = \left(\dfrac{v_2}{v_2 + v_1 y}\right)$

   c) Function: and: $y = \text{Test Value}$, $S_1 = \text{Shape 1}$, $S_2 = \text{Shape 2}$

and: $I(x,a,b) \equiv \dfrac{B_x(a,b)}{B(a,b)} \equiv \dfrac{1}{B(a,b)} \displaystyle\int_0^x t^{a-1}(1-t)^{b-1}\,dt$

and: $B(a,b) = \displaystyle\int_0^1 t^{a-1}(1-t)^{b-1}\,dt$

(From Croarkin & Tobias, Date Unknown; Press et al, 1997)

7. **CDF_Logistic:** This function returns the probability that the Test Value ≤ *X*, assuming a *Logistic* distribution with the specified mean and scale.
   a) Parameters:
      i) Test Value: Number
      ii) Mean: Number
      iii) Scale: Number > 0
   b) Usages:
      i) From "Probability Distribution Calculator", select "Cumulative Probability (CDF)" and *Logistic* distribution.
      ii) (*Avenue*): theProb = av.Run("Jennessent.DistFunc", {"CDF_Logistic", {Test Value, Mean, Scale}})
      iii) (*Avenue*): theProb = av.Run("Jennessent.TableDistFunc", {"CDF_Logistic", {Test Value, Mean, Scale}})
      iv) (*Avenue*): theProb = av.Run("Jennessent.CDF_Logistic", {Test Value, Mean, Scale})

c) Function: $\text{Logistic CDF} = \dfrac{1}{1+\exp\left(\dfrac{A-y}{B}\right)}$

where: y = Test Value, A = Mean, B = Scale

8. **CDF_LogNormal:** This function returns the probability that the Test Value ≤ *X*, assuming a *LogNormal* distribution with the specified mean and scale. A *LogNormal* distribution occurs when natural logarithms of variable *X* are normally distributed. The *Standard LogNormal Distribution* is that with Mean = 0 and Scale = 1. The *2-Parameter LogNormal Distribution* is that with Mean = 0.
   a) Parameters:
      i) Test Value: Number ≥ 0
      ii) Mean: Number > 0
      iii) Scale: Number > 0
   b) Usages:
      i) From "Probability Distribution Calculator", select "Cumulative Probability (CDF)" and *LogNormal* distribution.
      ii) (*Avenue*): theProb = av.Run("Jennessent.DistFunc", {"CDF_LogNormal, {Test Value, Mean, Scale}})
      iii) (*Avenue*): theProb = av.Run("Jennessent.TableDistFunc", {"CDF_LogNormal, {Test Value, Mean, Scale}})
      iv) *(Avenue*): theProb = av.Run("Jennessent.CDF_LogNormal", {Test Value, Mean, Scale})
   c) Function:

$$\text{LogNormal CDF} = \Phi\left(\frac{\ln(y)-A}{B}\right)$$

where: y = Test Value, A = Mean, B = Scale

and: $\Phi(x)$ = Cumulative Distribution Function of the Normal Distribution

9. **CDF_Normal_Simpsons:** This function returns the probability that the Test Value ≤ *X*, assuming a *Normal* distribution with the specified mean and standard deviation. Because the formula for this function does not exist in a closed form, it must be computed numerically. This script uses the *Simpson's* approximation method (Stewart 1998, p. 421-424) to calculate a highly accurate estimate of the Normal cumulative distribution function (accuracy to > 12 decimal places). The *Standard Normal Distribution* is that with Mean = 0 and Standard Deviation = 1.
   a) Parameters:
      i) Test Value: Number
      ii) Mean: Number
      iii) Standard Deviation: Number > 0

b) Usages:
   - i) From "Probability Distribution Calculator", select "Cumulative Probability (CDF)" and *Normal* distribution.
   - ii) (*Avenue*): theProb = av.Run("Jennessent.DistFunc”, {“CDF_Normal_Simpsons, {Test Value, Mean, St. Dev.}})
   - iii) (*Avenue*): theProb = av.Run("Jennessent.TableDistFunc”, {“CDF_Normal_Simpsons, {Test Value, Mean, St. Dev.}})
   - iv) (*Avenue*): theProb = av.Run("Jennessent.CDF_Normal_Simpsons", {Test Value, Mean, St. Dev.})

c) Function:

$$\text{Normal CDF} = \Phi\left(\frac{y-A}{B}\right)$$

where: y = Test Value, A = Mean, B = Scale

and: $\Phi(x)$ = Cumulative Distribution Function of the Normal Distribution

10. CDF_Poisson: This function returns the probability that the specified Number of Events will be ≤ *X*, assuming a *Poisson* distribution with the specified mean.

a) Parameters:
   - i) # Events: Integer ≥ 0
   - ii) Mean: Number > 0

b) Usages:
   - i) From "Probability Distribution Calculator", select "Cumulative Probability (CDF)" and *Poisson* distribution.
   - ii) (*Avenue*): theProb = av.Run("Jennessent.DistFunc”, {“CDF_Poisson, {# Events, Mean}})
   - iii) (*Avenue*): theProb = av.Run("Jennessent.TableDistFunc”, {“CDF_Poisson, {# Events, Mean}})
   - iv) (*Avenue*): theProb = av.Run("Jennessent.CDF_Poisson", {# Events, Mean})

$$\text{Poisson CDF} = \frac{\gamma(y,A)}{\Gamma(y)}$$

c) Function: where: y = Test value, A = Expectation (mean)

and: $\Gamma(x) = \int_0^\infty t^{x-1} e^{-1} dt$

and: $\gamma(x,y) = \int_0^y t^{x-1} e^{-1} dt$

11. CDF_StudentsT: This function returns the probability that the Test Value ≤ *X*, assuming a *Students T* distribution with the specified Degrees of Freedom. A *Student’s T* distribution with 1df is a *Cauchy* Distribution, and it approaches a *Normal* distribution when DF>30. Various sources recommend using the *Normal* distribution if DF>100.

a) Parameters:
   - i) Test Value: Number

    ii) Degrees of Freedom: Number > 0

b) Usages:

    i) From "Probability Distribution Calculator", select "Cumulative Probability (CDF)" and *Student's T* distribution.
    ii) (*Avenue*): theProb = av.Run("Jennessent.DistFunc”, {“CDF_StudentsT, {Test Value, DF}})
    iii) (*Avenue*): theProb = av.Run("Jennessent.TableDistFunc”, {“CDF_StudentsT, {Test Value, DF}})
    iv) (*Avenue*): theProb = av.Run("Jennessent.CDF_StudentsT", {Test Value, DF})

c) Function: The CDF_StudentsT T Function is dependent on whether the test value is positive or negative:

$$\text{Student's T CDF} = \begin{cases} \frac{1}{2} I\left(\frac{v}{v+y^2}, \frac{v}{2}, \frac{1}{2}\right), & t \equiv y \le 0 \\ 1 - \frac{1}{2} I\left(\frac{v}{v+y^2}, \frac{v}{2}, \frac{1}{2}\right), & t \equiv y > 0 \end{cases}$$

where: y = Test Value, v = Degrees of Freedom

and: $I(x,a,b) \equiv \frac{B_x(a,b)}{B(a,b)} \equiv \frac{1}{B(a,b)} \int_0^x t^{a-1}(1-t)^{b-1} dt$

and: $B(a,b) = \int_0^1 t^{a-1}(1-t)^{b-1} dt$

12. CDF_Weibull: This function returns the probability that the Test Value ≤ *X*, assuming a *Weibull* distribution with the specified Location, Scale and Shape parameters. The *Standardized Weibull Distribution* is that with *Location = 0* and *Scale = 1.* The *2-Parameter Weibull Distribution* is that with *Location = 0.*

a) Parameters:

    i) Test Value: Number > Location
    ii) Location: Number
    iii) Scale: Number > 0
    iv) Shape: Number > 0

b) Usages:

    i) From "Probability Distribution Calculator", select "Cumulative Probability (CDF)" and *Weibull* distribution.
    ii) (*Avenue*): theProb = av.Run("Jennessent.DistFunc”, {“CDF_Weibull, {Test Value, Location, Scale, Number}})
    iii) (*Avenue*): theProb = av.Run("Jennessent.TableDistFunc”, {“CDF_Weibull, {Test Value, Location, Scale, Number}})
    iv) (*Avenue*): theProb = av.Run("Jennessent.CDF_Weibull", {Test Value, Location, Scale, Number})

c) Function: $\text{Weibull CDF} = 1 - \exp^{\left(-\left(\frac{y-A}{B}\right)^{C}\right)}$

where: y = Test Value, A = Location, B = Scale, C = Shape

Quantiles (also referred to as Inverse Density Functions or Percent Point Functions).

1. **IDF_Beta:** This function takes the specified probability and returns the value *X*, such that *P*(*X*) = *P-value*, given the Beta distribution with the two specified Shape parameters. Because the formula for this function does not exist in a closed form, it must be computed numerically. This script arrives at the *X*-value through an iterative process, repeatedly testing *X*-values with the *CDF_Beta* function until it arrives at *P-value* that is within $1x10^{-12}$ units from the specified *P-value* (this usually takes between 30-60 iterations).
   a) Parameters:
      i) P-value: Number ($0 \geq p \geq 1$)
      ii) Shape1: Number > 0
      iii) Shape2: Number > 0
   b) Usages:
      i) From "Probability Distribution Calculator", select "Quantile (IDF; Inverse CDF)" and *Beta* distribution.
      ii) (*Avenue*): theX = av.Run("Jennessent.DistFunc”, {“IDF_Beta, {P-value, Shape1, Shape2}})
      iii) (*Avenue*): theX = av.Run("Jennessent.TableDistFunc”, {“IDF_Beta, {P-value, Shape1, Shape2}})
      iv) (*Avenue*): theX = av.Run("Jennessent.IDF_Beta", {P-value, Shape1, Shape2})
   c) Function:

$$\text{Beta IDF} = \int_0^y \frac{\Gamma(S_1+S_2)}{\Gamma(S_1)\Gamma(S_2)} y^{S_1-1}(1-y)^{S_2-1}$$

where: y = Test Value, S_1 = Shape 1, S_2 = Shape 2

and: $\Gamma(x) = \int_0^\infty t^{x-1} e^{-1} dt$

2. **IDF_Binomial:** This function takes the specified probability and returns the value *X* such that the Probability of getting (*X* – 1) successes ≤ the Specified Probability. This function takes an iterative approach to finding the correct *X* value, repeatedly trying different values of *X* until it reaches the correct one. This iterative process rarely takes more than 25 repetitions.
   a) Parameters:
      i) P-value = Number ($0 \geq p \geq 1$)
      ii) # Trials = Integer ≥ 2
      iii) Probability of Success = Number ($0 \geq p \geq 1$)Usages:
   b) Usages:
      i) From "Probability Distribution Calculator", select "Quantile (IDF; Inverse CDF)" and *Binomial* distribution.
      ii) (*Avenue*): theX = av.Run("Jennessent.DistFunc”, {“IDF_Binomial, {P-value, NumTrials, Probability of Success}})
      iii) (*Avenue*): theX = av.Run("Jennessent.TableDistFunc”, {“IDF_Binomial, {P-value, NumTrials, Probability of Success}})

    - iv) *(Avenue)*: theX = av.Run("Jennessent.IDF_Binomial", {P-value, NumTrials, Probability of Success})

   c) Function: Binomial IDF: Iterative Process, repeatedly testing values of y, such that:

$$p = \sum_{i=1}^{y} \left(\frac{B!}{i!(B-i)!} \right) A^{i} (1-A)^{B-i}$$

where: y = #Successes, A = Probability of Success, B = #Trials

Until: $P(y-1) \leq$ User-Specified probability

3. **IDF_Cauchy:** This function takes the specified probability and returns the value *X*, such that *P*(*X*) = *P-value*, given the Cauchy distribution with the specified location and scale parameters. The *Standardized Cauchy distribution* has *Location = 0* and *Scale = 1.*
   a) Parameters:
      - i) P-value: Number ($0 \geq p \geq 1$)
      - ii) Location: Number
      - iii) Scale: Number > 0
   b) Usages:
      - i) From "Probability Distribution Calculator", select "Quantile (IDF; Inverse CDF)" and *Cauchy* distribution.
      - ii) (*Avenue*): theX = av.Run("Jennessent.DistFunc", {"IDF_Cauchy, {P-value, location, Scale}})
      - iii) (*Avenue*): theX = av.Run("Jennessent.TableDistFunc", {"IDF_Cauchy, {P-value, location, Scale}})
      - iv) *(Avenue)*: theX = av.Run("Jennessent.IDF_Cauchy", {P-value, Location, Scale})

   c) Function: Cauchy IDF = $A - \frac{B}{\tan(\pi p)}$

where: A = Location, B = Scale, p = Probability

4. **IDF_ChiSquare:** This function takes the specified probability and returns the value *X*, such that *P*(*X*) = *P-value*, given the Chi-Square distribution with the specified Degrees of Freedom. Because the formula for this function does not exist in a closed form, it must be computed numerically. This script arrives at the *X*-value through an iterative process, repeatedly testing *X*-values with the *CDF_ChiSquare* function until it arrives at *P-value* that is within $1x10^{-12}$ units from the specified *P-value* (this usually takes between 30-60 iterations). The *Chi-Square* distribution results when *v* (where *v* = Degrees of Freedom) independent variables with standard normal distributions are squared and summed (Croarkin & Tobias, Date unknown).
   a) Parameters:
      - i) P-value: Number ($0 \geq p \geq 1$)
      - ii) Degrees of Freedom: Number
   b) Usages:

    i) From "Probability Distribution Calculator", select "Quantile (IDF; Inverse CDF)" and *Chi-Square* distribution.
    ii) (*Avenue*): theX = av.Run("Jennessent.DistFunc”, {“IDF_ChiSquare, {P-Value, F}})
    iii) (*Avenue*): theX = av.Run("Jennessent.TableDistFunc”, {“IDF_ChiSquare, {P-Value, F}})
    iv) *(Avenue*): theX = av.Run("Jennessent.IDF_ChiSquare", {P-Value, DF})

c) Function:

$$\text{Chi-Square IDF} = \int_0^y \frac{e^{\frac{-y}{2}} x^{\frac{v}{2}-1}}{2^{\frac{v}{2}}\Gamma\left(\frac{v}{2}\right)} dy$$

where: y = Test Value, S_1 = Shape 1, S_2 = Shape 2

and: $\Gamma(x) = \int_0^\infty t^{x-1} e^{-1} dt$

5. **IDF_Exp:** This function takes the specified probability and returns the value *X*, such that *P*(*X*) = *P-value*, given the Exponential distribution with the specified mean. This script uses the 1-parameter version of the equation (i.e. assuming *Location* = 0). The *Standard Exponential Distribution* is that which has *Mean = 1*.
    a) Parameters:
        i) P-value: Number ($0 \geq p \geq 1$)
        ii) Mean: Number > 0
    b) Usages:
        i) From "Probability Distribution Calculator", select "Quantile (IDF; Inverse CDF)" and *Exponential* distribution.
        ii) (*Avenue*): theX = av.Run("Jennessent.DistFunc”, {“IDF_Exp, {P-value, Mean}})
        iii) (*Avenue*): theX = av.Run("Jennessent.TableDistFunc”, {“IDF_Exp, {P-value, Mean}})
        iv) *(Avenue*): theX = av.Run("Jennessent.IDF_Exp", {P-value, Mean})
    c) Function:

$$\text{Exponential IDF} = -\beta \ln(1-p)$$

where: β = Mean (or Scale Parameter)

and: p = Specified Probability

6. **IDF_F:** This function takes the specified probability and returns the value *X*, such that *P*(*X*) = *P-value*, given the *F* distribution with the specified Degrees of Freedom. Because the formula for this function does not exist in a closed form, it must be computed numerically. This script arrives at the *X*-value through an iterative process, repeatedly testing *X*-values with the *CDF_F* function until it arrives at *P-value* that is within $1x10^{-12}$ units from the specified *P-value* (this usually takes between 30-60 iterations). The *F* distribution is the ratio of two *Chi-Square* distributions with ratios v_1 and v_2 respectively.
    a) Parameters:
        i) Test Value: Number ≥ 1
        ii) 1st Degrees of Freedom: Number > 1

    iii) 2nd Degrees of Freedom: Number > 1
  b) Usages:
    i) From "Probability Distribution Calculator", select "Quantile (IDF; Inverse CDF)" and *F* distribution.
    ii) (*Avenue*): theX = av.Run("Jennessent.DistFunc”, {“IDF_F, {P-value, DF1, DF2}})
    iii) (*Avenue*): theX = av.Run("Jennessent.TableDistFunc”, {“IDF_F, {P-value, DF1, DF2}})
    iv) *(Avenue*): theX = av.Run("Jennessent.IDF_F", {P-value, DF1, DF2})

  c) Function: $$\text{F IDF} = \int_0^x \left(\frac{\Gamma\left(\frac{v_1+v_2}{2}\right)\left(\frac{v_1}{v_2}\right)^{\frac{v_1}{2}} x^{\frac{v_1}{2}-1}}{\Gamma\left(\frac{v_1}{2}\right)\Gamma\left(\frac{v_1}{2}\right)\left(1+\frac{v_1 x}{v_2}\right)^{\frac{v_1+v_2}{2}}} \right) dx$$

where: x = Test Value, v_1 = DF1, v_2 = DF2

and: $\Gamma(x) = \int_0^\infty t^{x-1} e^{-1} dt$

7. **IDF_Logistic:** This function takes the specified probability and returns the value *X*, such that *P*(*X*) = *P-value*, given the *Logistic* distribution with the specified mean and scale parameters.
  a) Parameters:
    i) P-value: Number ($0 \geq p \geq 1$)
    ii) Mean: Number
    iii) Scale: Number > 0
  b) Usages:
    i) From "Probability Distribution Calculator", select "Quantile (IDF; Inverse CDF)" and *Logistic* distribution.
    ii) (*Avenue*): theX = av.Run("Jennessent.DistFunc”, {“IDF_Logistic, {P-value, Mean, Scale}})
    iii) (*Avenue*): theX = av.Run("Jennessent.TableDistFunc”, {“IDF_Logistic, {P-value, Mean, Scale}})
    iv) *(Avenue*): theX = av.Run("Jennessent.IDF_Logistic", {P-value, Mean, Scale})

  c) Function: Logistic IDF = $A + B \ln\left(\frac{p}{1-p}\right)$

where: p = Probability, A = Mean, B = Scale

8. **IDF_LogNormal:** This function takes the specified probability and returns the value *X*, such that *P*(*X*) = *P-value*, given the *LogNormal* distribution with the specified mean and scale parameters. Because the formula for this function does not exist in a closed form, it must be computed numerically. This script arrives at the *X*-value through an iterative process, repeatedly testing *X*-values with the *CDF_LogNormal* function until it arrives at *P-value* that

is within *1x10* $^{-12}$ units from the specified *P-value* (this usually takes between 30-60 iterations). A *LogNormal* distribution occurs when natural logarithms of variable *X* are normally distributed. The *Standard LogNormal Distribution* is that with Mean = 0 and Scale = 1. The *2-Parameter LogNormal Distribution* is that with Mean = 0.

a) Parameters:
   - i) P-value: Number ($0 \geq p \geq 1$)
   - ii) Mean: Number > 0
   - iii) Scale: Number > 0

b) Usages:
   - i) From "Probability Distribution Calculator", select "Quantile (IDF; Inverse CDF)" and *LogNormal* distribution.
   - ii) (*Avenue*): theX = av.Run("Jennessent.DistFunc", {"IDF_LogNormal, {P-value, Mean, Scale}})
   - iii) (*Avenue*): theX = av.Run("Jennessent.TableDistFunc", {"IDF_LogNormal, {P-value, Mean, Scale}})
   - iv) *(Avenue*): theX = av.Run("Jennessent.IDF_LogNormal", {P-value, Mean, Scale})

c) Function: LogNormal IDF = $\int_0^y \left(\frac{1}{B\sqrt{2\pi}} \exp\left(-\frac{1}{2} \left(\frac{\ln(y) - A}{B} \right)^2 \right) \right) dy$

where: y = Test Value, A = Mean, B = Scale

9. **IDF_Normal:** This function takes the specified probability and returns the value *X*, such that *P*(*X*) = *P-value*, given the *Normal* distribution with the specified mean and standard deviation. Because the formula for this function does not exist in a closed form, it must be computed numerically. This script arrives at the *X*-value through an iterative process, repeatedly testing *X*-values with the *CDF_Normal_Simpsons* function until it arrives at *P-value* that is within *1x10* $^{-12}$ units from the specified *P-value* (this usually takes between 30-60 iterations). Furthermore, there is no closed formula for calculating the Normal cumulative distribution function, so this script uses the *Simpson's* approximation method (Stewart 1998, p. 421-424) to calculate a highly accurate estimate (accuracy to > 12 decimal places). The *Standard Normal Distribution* is that with Mean = 0 and Standard Deviation = 1.

a) Parameters:
   - i) P-value: Number ($0 \geq p \geq 1$)
   - ii) Mean: Number
   - iii) Standard Deviation: Number > 0

b) Usages:
   - i) From "Probability Distribution Calculator", select "Quantile (IDF; Inverse CDF)" and *Normal* distribution.
   - ii) (*Avenue*): theX = av.Run("Jennessent.DistFunc", {"IDF_Normal, {P-value, Mean, St. Dev.}})
   - iii) (*Avenue*): theX = av.Run("Jennessent.TableDistFunc", {"IDF_Normal, {P-value, Mean, St. Dev.}})
   - iv) *(Avenue*): theX = av.Run("Jennessent.IDF_Normal", {P-value, Mean, St. Dev.})

c) Function: Normal IDF = $\int_0^y \left(\frac{1}{B\sqrt{2\pi}} \exp\left(-\frac{1}{2}\left(\frac{y-A}{B}\right)^2 \right) \right) dy$

where: y = Test Value, A = Mean, B = Scale

10. **IDF_Poisson:** This function takes the specified probability and returns the value *X* such that the Probability of getting (*X* – 1) events ≤ the Specified Probability. This function takes an iterative approach to finding the correct *X* value, repeatedly trying different values of *X* until it reaches the correct one.
    a) Parameters:
        i) P-value: Number ($0 \geq p \geq 1$)
        ii) Mean: Number > 0
    b) Usages:
        i) From "Probability Distribution Calculator", select "Quantile (IDF; Inverse CDF)" and *Poisson* distribution.
        ii) (*Avenue*): theProb = av.Run("Jennessent.DistFunc", {"IDF_Poisson, {P-value, Mean}})
        iii) (*Avenue*): theProb = av.Run("Jennessent.TableDistFunc", {"IDF_Poisson, {P-value, Mean}})
        iv) *(Avenue*): theProb = av.Run("Jennessent.IDF_Poisson", {P-value, Mean})

    c) Function:

Poisson IDF: Iterative Process, repeatedly testing values of y, such that:

$$p = \frac{\gamma(y, A)}{\Gamma(y)}$$

where: y = Test value, A = Expectation (mean)

and: $\Gamma(x) = \int_0^\infty t^{x-1} e^{-1} dt$

and: $\gamma(x, y) = \int_0^y t^{x-1} e^{-1} dt$

Until: $P(y-1) \leq$ User-Specified probability

11. **IDF_StudentsT:** This function takes the specified probability and returns the value *X*, such that *P*(*X*) = *P-value*, given the *Student's T* distribution with the specified Degrees of Freedom. Because the formula for this function does not exist in a closed form, it must be computed numerically. This script arrives at the *X*-value through an iterative process, repeatedly testing *X*-values with the *CDF_StudentsT* function until it arrives at *P-value* that is within $1x10^{-12}$ units from the specified *P-value* (this usually takes between 30-60 iterations). A *Student's T* distribution with 1df is a *Cauchy* Distribution, and it approaches a *Normal* distribution when DF>30. Various sources recommend using the *Normal* distribution if DF>100.
    a) Parameters:
        i) P-value: Number ($0 \geq p \geq 1$)
        ii) Degrees of Freedom: Number > 0

b) Usages:
  i) From "Probability Distribution Calculator", select "Quantile (IDF; Inverse CDF)" and *Student's T* distribution.
  ii) (*Avenue*): theX = av.Run("Jennessent.DistFunc", {"IDF_StudentsT, {P-value, DF}})
  iii) (*Avenue*): theX = av.Run("Jennessent.TableDistFunc", {"IDF_StudentsT, {P-value, DF}})
  iv) *(Avenue*): theX = av.Run("Jennessent.IDF_StudentsT", {P-value, DF})

c) Function:

$$\text{Student's T IDF} = \int_{-\infty}^{y} \frac{\Gamma\left(\frac{v+1}{2}\right)}{\sqrt{\pi v}\,\Gamma\left(\frac{v}{2}\right)} \left[1+\frac{y^2}{v}\right]^{-\frac{v+1}{2}} dy$$

where: y = Test Value, v = Degrees of Freedom

and: $\Gamma(x) = \int_0^\infty t^{x-1} e^{-1} dt$

12. IDF_Weibull: This function takes the specified probability and returns the value *X*, such that *P*(*X*) = *P-value*, given the *Weibull* distribution with the specified Location, Scale and Shape parameters. The *Standardized Weibull Distribution* is that with *Location = 0* and *Scale = 1*. The *2-Parameter Weibull Distribution* is that with *Location = 0*.

a) Parameters:
  i) Test Value: Number > Location
  ii) Location: Number
  iii) Scale: Number > 0
  iv) Shape: Number > 0

b) Usages:
  i) From "Probability Distribution Calculator", select "Quantile (IDF; Inverse CDF)" and *Weibull* distribution.
  ii) (*Avenue*): theX = av.Run("Jennessent.DistFunc", {"IDF_Weibull, {P-value, Location, Scale, Number}})
  iii) (*Avenue*): theX = av.Run("Jennessent.TableDistFunc", {"IDF_Weibull, {P-value, Location, Scale, Number}})
  iv) *(Avenue*): theX = av.Run("Jennessent.IDF_Weibull", {P-value, Location, Scale, Number})

c) Function: Weibull IDF = $A + B\sqrt[C]{-\ln p}$

where: p = Probability, A = Location, B = Scale, C = Shape

References

Abramowitz, Milton; Stegen, Irene A. (1972). Handbook of Mathematical Functions.

Burkardt, John. (2001). PROB - Probability Density Functions. Sample Fortran code for calculating density functions. Available on-line at http://www.psc.edu/~burkardt/src/prob/prob.html.

Croarkin, Carrol; Tobias, Paul. (Date Unknown). Engineering and Statistics Handbook, available on-line at: (http://www.itl.nist.gov/div898/handbook/). National Institute of Standards and Technology.

Jeffrey, Alan. (2000). Handbook of Mathematical Formulas and Integrals (2nd Ed). Academic Press

McLaughlin, Michael P. (2001). Regress+, Appendix A. A Compendium of Common Probability Distributions (version 2.3). Available on-line at: http://www.causaScientia.org/math_stat/Dists/Compendium.pdf

Neter, John; Wasserman, William; and Kutner, Michael H. (1990). Applied Linear Statistical Models (3rd Edition). Published by Irwin

Ott, R. Lyman. (1993). An Introduction to Statistical Methods and Data Analysis (4th Ed). Duxbury Press.

Press, William H.; Teukolsky, Saul A.; Vetterling William T.; and Flannery, Brian P. (1988-1997) Numerical Recipes in C; the Art of Scientific Computing (2nd Ed). Cambridge University Press. (ISBN 0-521-43108-5). (http://lib-www.lanl.gov/numerical/bookcpdf.html - See Chapter 6, sections 1-4)

S-Plus Help Files (2001). S-Plus 6 for Windows. Product Information available at:

http://www.insightful.com/support/splus60win/default.asp

S-PLUS 6 for Windows Guide to Statistics, Volume 1, Insightful Corporation, Seattle, WA.

SPSS Help Files. (1999). SPSS for Windows Release 9. Product Information available at:

http://www.spss.com/

Stewart, James. (1998). Calculus, Concepts and Contexts. Brooks/Cole Publishing Company. p. 421-424