FLŌSim Documentation

December 10, 2025

Contents

1 Add-in Installation

The plugin can be added on both Windows and Mac operating systems. For Mac users, Excel 2021 or later, or Excel 365 is required for full compatibility with `flosim.xlam` file.

Note: This add-in is not supported in Excel for the Web (browser-based version). Only desktop installations are supported.

1.1 Windows PC

1. **Unblock the file (required)**:
   (a) Close Excel completely.
   (b) Right-click the `.xlam` file and choose **Properties**.
   (c) At the bottom of the **General** tab, check the box **Unblock** ("This file came from another computer and might be blocked").
   (d) Click **Apply** and **OK**.
2. **Disable Protected View for files from trusted locations:**

(a) Open Excel.

(b) Go to **File** → **Options**.

(c) Select **Trust Center** → **Trust Center Settings**.

(d) Choose **Protected View**.

(e) Uncheck (recommended for add-ins only):

- **Enable Protected View for files originating from the internet**
- **Enable Protected View for files located in potentially unsafe locations**

(f) Click **OK**.

Note: Add-ins should never open in Protected View. If they do, macros will stay disabled regardless of other Trust Center settings.

3. (Recommended) **Move the unblocked file to a trusted folder**, such as:

   - `%APPDATA%\Microsoft\AddIns`
   - or any folder you explicitly trust e.g. `Documents`

4. **Add the folder as a Trusted Location:**

   (a) Open Excel.

   (b) Go to **File** → **Options**.

   (c) Select **Trust Center** → **Trust Center Settings**.

   (d) Choose **Trusted Locations**.

   (e) Click **Add new location**, browse to the folder where the add-in is stored, and confirm.

5. **Enable macros (if required):**

   (a) In **Trust Center Settings**, go to **Macro Settings**.

   (b) Choose either:

- **Disable all macros except digitally signed macros**, or
- **Enable all macros** (least restrictive)

(c) Click **OK**.

6. **Load the add-in:**

   (a) Go to **File** → **Options**.

   (b) Select **Add-ins**.

   (c) At the bottom, set **Manage** to **Excel Add-ins** and click **Go**.

   (d) Click **Browse** and select the `.xlam` file in your trusted folder.

   (e) Ensure the checkbox next to the add-in name is checked.

7. **Verify installation**: In a blank cell, try a function such as

   $$= \texttt{norm()}$$

   If the add-in is installed and trusted, the function will evaluate normally (no #NAME? error).

1.2 Mac OS

1. Open Microsoft Excel.
2. Go to the menu bar and click **Tools** → **Excel Add-ins**.
3. In the Add-ins dialog, click the **Browse** button.
4. Locate and select the `.xlam` file, then click **Open**.
5. Make sure the checkbox next to the add-in name is checked.
6. Close the dialog. The add-in should now be active.
7. **Verify installation**: In a blank cell, try a function such as

   $$= \texttt{norm()}$$

If the add-in is installed and trusted, the function will evaluate normally (no #NAME? error).

Note: Macros must be enabled in Excel Preferences under **Security** to allow the add-in to function properly.

2 FLŌSim Ribbon Overview

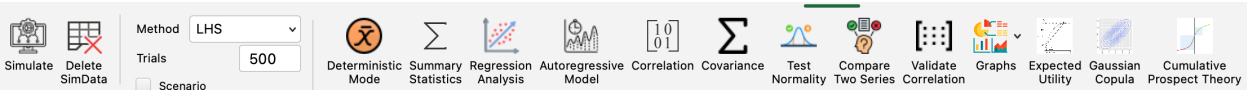


Figure 1: *Custom FLOSim Ribbon Interface in Excel*

The ribbon includes the following buttons and dropdowns:

1. **Simulate**
2. **Delete SimData**
3. **Method** (Dropdown: LHS, Monte Carlo)
4. **Trials** (Numeric input)
5. **Scenario** (Checkbox)
6. **Deterministic Mode**
7. **Summary Statistics**
8. **Regression Analysis**
9. **AR Model**
10. **Correlation**
11. **Covariance**
12. **Test Normality**
13. **Compare Two Series**
14. **Validate Correlation**
15. **Graphs**
    (a) **PDF**

(b) **CDF**

(c) **Stoplight**

(d) **Fan Graph**

(e) **Twoway Scatter**

16. **Expected Utility**

17. **Gaussian Copula**

18. **Cumulative Prospect Theory**

1. Simulate

This button runs the simulation engine based on the selected method (either Latin Hypercube Sampling or Monte Carlo) and the user-defined model inputs.

Upon clicking the `Simulate` button, the following steps occur:

1. A dialog box prompts to **select the cell or range** to simulate.
2. A second input box appears asking to optionally:
   - Specify the label location (`none`, `above`, or `left`) to capture variable names.
   - Enter a custom name for the output sheet (e.g., `SimData`).

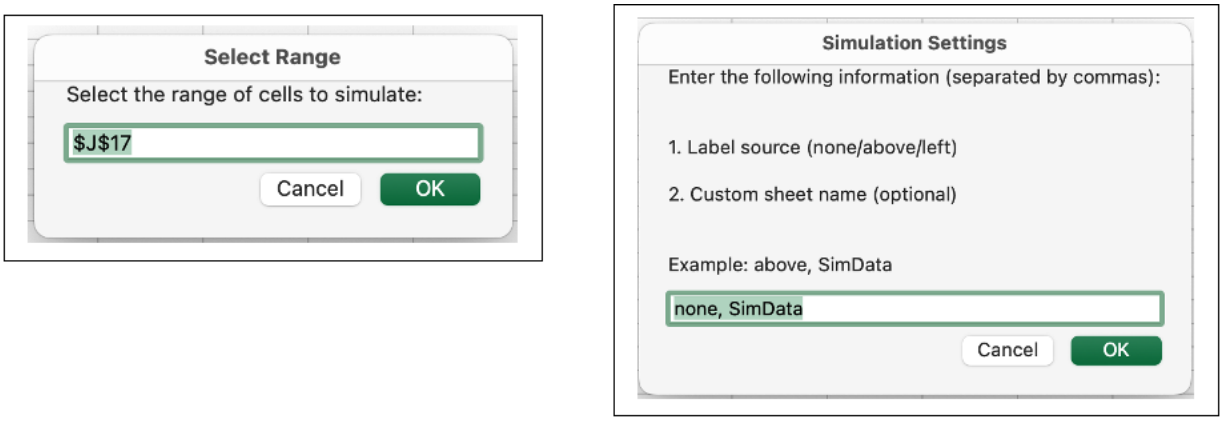


Figure 2: *Simulation settings*

3. The simulation engine will then:

    - Identify and simulate values for the selected range.
    - Apply the chosen sampling method (LHS or MC).
    - Create a new worksheet with results, including trial-level data and summary statistics.

2. Delete SimData

This button clears any previously generated simulation worksheets that begin with the prefix `Sim`. Specifically, it will:

- Locate all worksheets with names that begin with "Sim" (e.g., `SimData`, `SimSummary`).
- Prompt the user for confirmation before deletion.
- Permanently remove the identified sheets from the workbook.

Use this feature to clean up previous results before running a new simulation to avoid confusion or clutter.

3. Method

This dropdown menu allows the user to select the desired sampling method for the simulation. The available options are:

- **LHS** (Latin Hypercube Sampling): A stratified sampling technique that divides each input parameter's range into equal probability intervals and ensures exactly one sample from each interval. This provides more uniform coverage and requires fewer samples to achieve reliable statistical estimates. Recommended for most use cases due to higher sampling efficiency.
- **MC** (Monte Carlo): Traditional random sampling where each sample is drawn independently and purely at random from the input distributions. While simple to implement, it can produce clustering and gaps, requiring more samples for reliable statistical estimates.

The default selection is **LHS**. This setting influences how the random draws are generated when the `Simulate` button is clicked.

4. Trials

Number of simulated trials to be conducted. The default is 500.

5. Scenario

The Scenario tool is useful to simulate alternative outcomes by substituting multiple sets of values for one or more input variables. It is designed to work in combination with the `=SCENARIO()` function, which must be applied to each input cell where alternative values are defined.

When the "Scenario" checkbox is selected in the Ribbon, the simulation is run separately for each column of the scenario table, holding the underlying random numbers constant across all scenarios. This ensures that any variation in the results is due solely to differences in input values (e.g., mean or standard deviation), rather than random noise, making scenario comparisons more meaningful and reproducible.

Example The example below illustrates how the `SCENARIO()` function can be used to implement scenario-based simulation.

- Cells `C3` and `C4` are the inputs to a normal distribution, specifying the mean and standard deviation, respectively.
- These cells use the `=SCENARIO()` function to point to the corresponding rows in the scenario table `E3:G4`.
- The scenario table defines three cases:
    1. Scenario 1: mean = 100, standard deviation = 20
    2. Scenario 2: mean = 105, standard deviation = 30
    3. Scenario 3: mean = 110, standard deviation = 35
- The simulated value in `C7` is calculated as `=NORM(C3,C4)`.

When simulation is run with the Scenario option enabled, a separate column of simulated results will be generated for each of the three scenarios, using the same underlying uniform draw to ensure fair comparison.

	A	B	C	D	E	F	G
1							
2					Scenario 1	Scenario 2	Scenario 3
3		Mean	100	=scenario(E3:G3)	100	105	110
4		St. Dev	20	=scenario(E4:G4)	20	30	35
5							
6							
7		Y	92.06066	=norm(C3,C4)			

Figure 3: *Example: Scenario-based simulation using the `SCENARIO()` function to vary distribution parameters.*

6. Deterministic Mode

The deterministic mode sets all uniform standard deviates to a fixed value of 0.5, effectively replacing stochastic variables with constant values. This feature is especially useful for debugging models, validating formulas, or ensuring consistent comparisons across different spreadsheets.

7. Summary Statistics

The "Summary Statistics" tool allows to quickly generate a descriptive summary of a selected data series. It creates descriptive univariate statistics in a user-specified output location.

Workflow:

1. The user is prompted to select a range of numeric data series (**without headers**).
2. A second prompt asks for:
    - The desired position of the label (e.g., "above"),
    - The output range output cell (e.g., "E2").

Displayed Statistics:

- **Mean (Average)**: `=AVERAGE(range)`
- **Median**: `=MEDIAN(range)`
- **Minimum**: `=MIN(range)`

- **Maximum**: `=MAX(range)`
- **Standard Deviation (Sample)**: `=STDEV.S(range)`
- **25th Percentile (Q1)**: `=PERCENTILE.INC(range, 0.25)`
- **75th Percentile (Q3)**: `=PERCENTILE.INC(range, 0.75)`

The demo example is shown in the figure below. The first prompt asks the user to select the data series (excluding labels), while the second prompt requests the label position and output cell. Once these inputs are provided, the summary statistics are automatically displayed in the specified location.

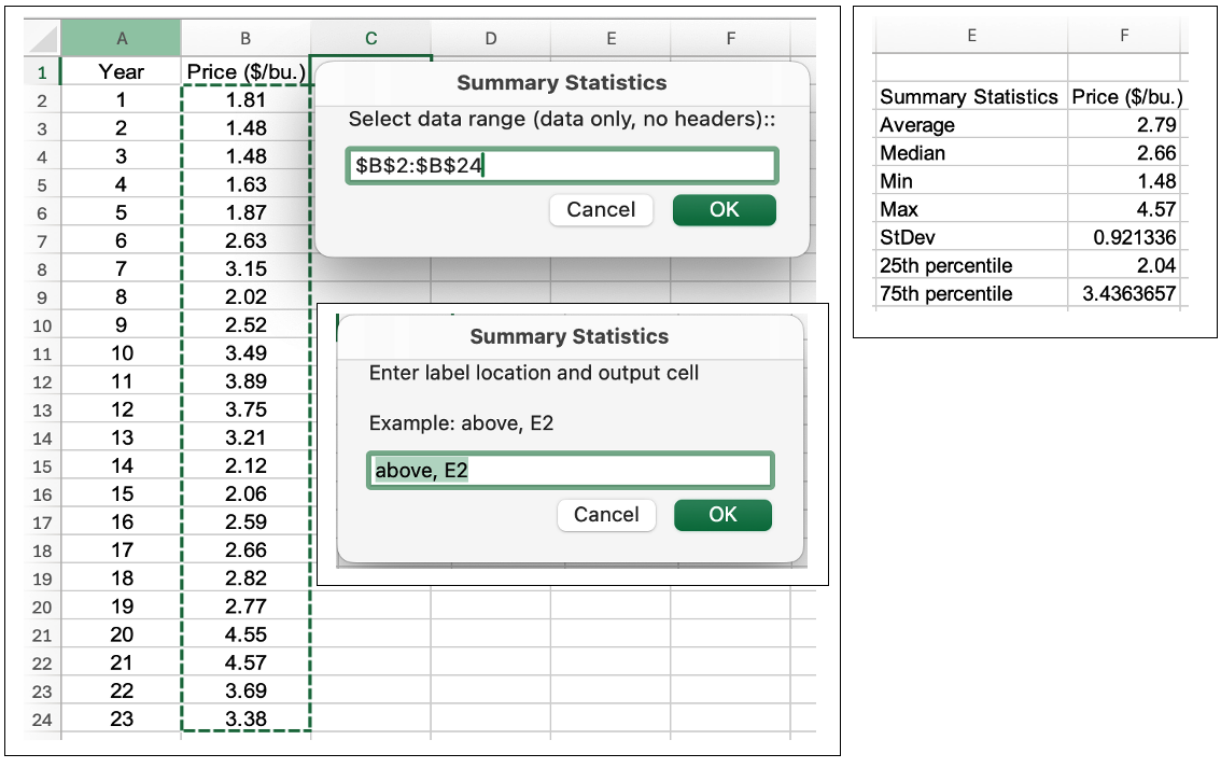


Figure 4: *Summary Statistics*

8. Regression Analysis

This tool performs a linear regression analysis between a dependent variable and one or more independent variables. It runs an ordinary least squares (OLS) regression and reports key diagnostics in the worksheet. A constant term (intercept) is included by default.

Workflow: Upon clicking the **Regression Analysis** button, the user is prompted through three steps:

1. Select the **dependent variable (Y)** range (data only, no headers).
2. Select the **independent variable(s) (X)** range (data only, no headers).
3. Select the **output cell** where results will be written.

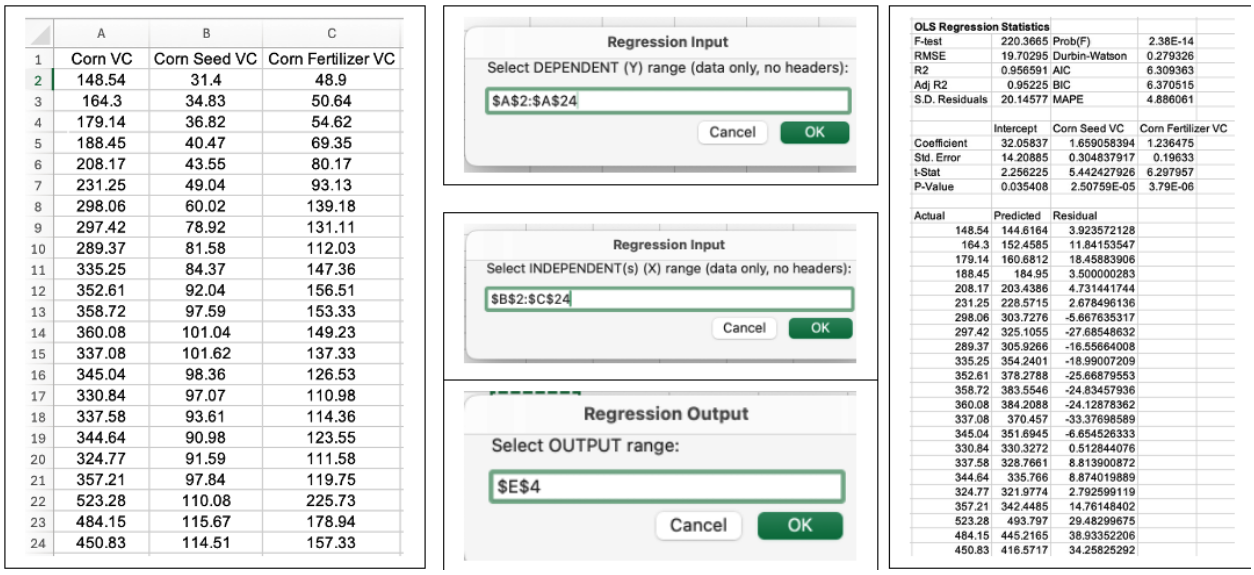
Output: The regression output is displayed beginning at the selected cell. The output includes:

- Estimated regression coefficients and standard errors
- t-statistics and p-values for inference
- Goodness-of-fit measures: R^2, adjusted R^2
- Model performance metrics: Root mean squared error (RMSE), mean absolute percentage error (MAPE), residual standard deviation
- Model selection criteria (scaled): Akaike Information Criterion (AIC) and Bayesian Information Criterion (BIC)
- Diagnostic statistics: F-statistic, its p-value, and Durbin-Watson statistic
- Predicted values and residuals

Variable Labels: If variable headers are present in the row directly above the X-range, they are automatically detected and used in the output table. Otherwise, default labels like `Var 1`, `Var 2`, etc., are applied.

The demo example is illustrated in the figure below. The first prompt asks the user to select the dependent variable data range (excluding labels), followed by a second prompt to select the independent variable(s). The third prompt requests the user to specify the output

cell where the results will be displayed. Once all inputs are provided, the regression output is generated in the specified location.



	A	B	C
1	Corn VC	Corn Seed VC	Corn Fertilizer VC
2	148.54	31.4	48.9
3	164.3	34.83	50.64
4	179.14	36.82	54.62
5	188.45	40.47	69.35
6	208.17	43.55	80.17
7	231.25	49.04	93.13
8	298.06	60.02	139.18
9	297.42	78.92	131.11
10	289.37	81.58	112.03
11	335.25	84.37	147.36
12	352.61	92.04	156.51
13	358.72	97.59	153.33
14	360.08	101.04	149.23
15	337.08	101.62	137.33
16	345.04	98.36	126.53
17	330.84	97.07	110.98
18	337.58	93.61	114.36
19	344.64	90.98	123.55
20	324.77	91.59	111.58
21	357.21	97.84	119.75
22	523.28	110.08	225.73
23	484.15	115.67	178.94
24	450.83	114.51	157.33

OLS Regression Statistics			
F-test	220.3665	Prob(F)	2.38E-14
RMSE	19.70295	Durbin-Watson	0.279326
R2	0.956591	AIC	6.309363
Adj R2	0.95225	BIC	6.370515
S.D. Residuals	20.14577	MAPE	4.886061
	Intercept	Corn Seed VC	Corn Fertilizer VC
Coefficient	32.05837	1.659058394	1.236475
Std. Error	14.20885	0.304837917	0.19633
t-Stat	2.256225	5.442427926	6.297957
P-Value	0.035408	2.50759E-05	3.79E-06
Actual	Predicted	Residual	
148.54	144.6164	3.923572128	
164.3	152.4585	11.84153547	
179.14	160.6812	18.45883906	
188.45	184.95	3.500000283	
208.17	203.4386	4.731441744	
231.25	228.5715	2.678496136	
298.06	303.7276	-5.667635317	
297.42	325.1055	-27.68548632	
289.37	305.9266	-16.55664008	
335.25	354.2401	-18.99007209	
352.61	378.2788	-25.66879553	
358.72	383.5546	-24.83457936	
360.08	384.2088	-24.12878362	
337.08	370.457	-33.37698589	
345.04	351.6945	-6.654526333	
330.84	330.3272	0.512844076	
337.58	328.7661	8.813900872	
344.64	335.766	8.874019889	
324.77	321.9774	2.792599119	
357.21	342.4485	14.76148402	
523.28	493.797	29.48299675	
484.15	445.2165	38.93352206	
450.83	416.5717	34.25825292	

Figure 5: *Regression Analysis*

9. AR Model

This tool fits an autoregressive (AR) time series model with optional differencing. It estimates the model parameters, forecasts future values, and presents residual diagnostics and a forecast chart in the worksheet.

Workflow: Upon clicking the **AR Model** button, the user is guided through two prompts:

1. Select the **data range** (one column, no headers). This is the historical time series to be modeled (not differences).

2. Enter four parameters as a comma-separated string in the format: `differencing, lags, forecast horizon, output cell`. **Example:** `1,3,5,E2`

   - `differencing` - order of differencing (e.g., 0, 1, or 2)
   - `lags` - number of autoregressive lags to include. The optimal number of lags can be determined using the sections 4.5, 4.6, and 4.7.

- `forecast horizon` - number of future steps to forecast
- `output cell` - output cell location for the output report

Output: The tool generates an AR model report starting from the selected output cell, including:

- A title indicating model structure (e.g., "AR model with 3 lags and 1 difference")
- Coefficient table:
  - Intercept and AR lag coefficients
  - Standard errors and t-statistics
- Forecast table:
  - Period index and corresponding forecast values
- Residual output for in-sample fitted values
- Line chart of historical and forecasted values with shaded forecast region
- Model diagnostics:
  - Standard deviation of residuals
  - Sum of squared errors (SSE)
  - Number of observations used
  - R^2 and adjusted R^2
  - Durbin-Watson statistic

The demo example is illustrated in the figure below. The first prompt asks the user to select the time series data range (original values, not differenced, and excluding any labels). The second prompt then requests the number of differences, the autoregressive lag order, the forecast horizon (number of steps), and the desired output location. Once all inputs are provided (in this case, 1 difference, 3 lags, and a 5-step forecast), the AR model output is generated at the specified location.

10. Correlation

This tool computes and displays the Pearson correlation matrix for a selected data range.

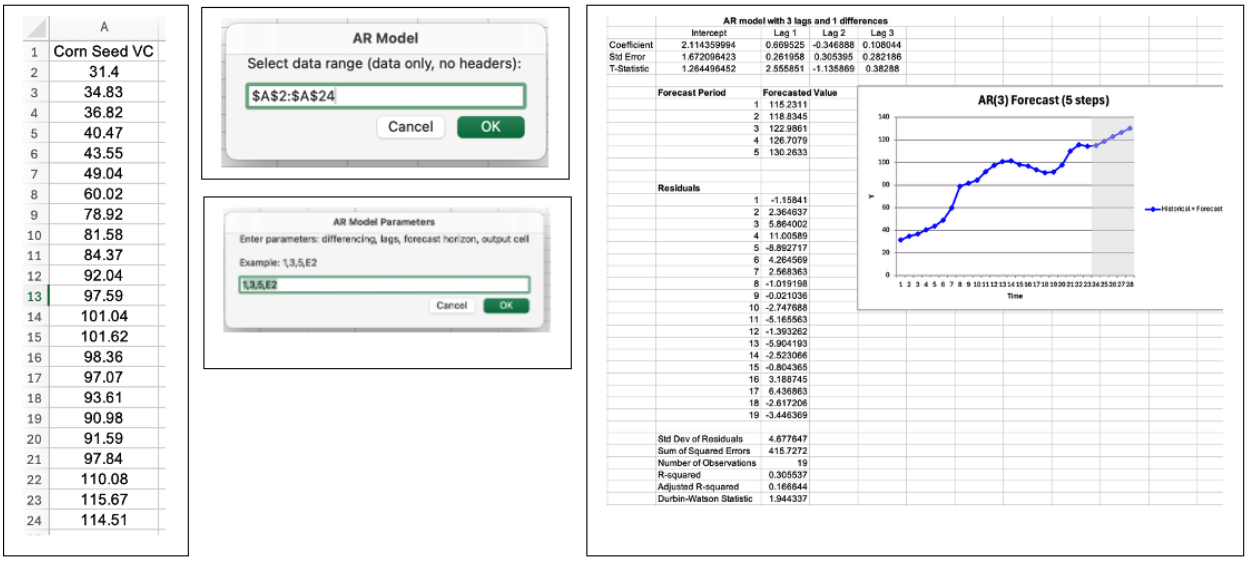


Figure 6: *AR Model*

Workflow: Upon clicking the **Correlation Matrix** button, the user is guided through two prompts:

1. Select the **data range** (numerical data only, no headers). Each column represents a variable.

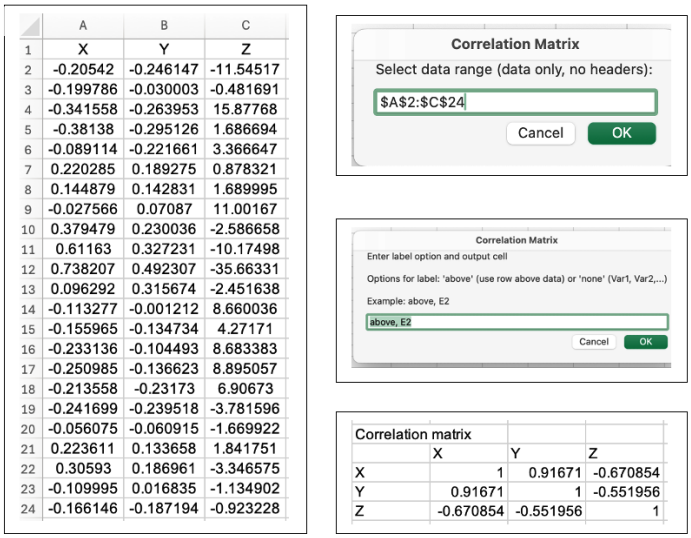
2. Enter the **labeling option** and **output cell**, as a comma-separated string. **Example:** `above, E2`

    - `above` - uses the row above the data range as variable labels
    - `none` - automatically assigns labels as `Var1`, `Var2`, etc.
    - `E2` - specifies the output cell

Output: The tool produces a correlation matrix beginning at the selected output cell, including:

- A title: `Correlation matrix`
- Variable labels on both rows and columns
- Symmetric matrix of pairwise Pearson correlation coefficients between all columns in the selected data range

The demo example is illustrated in the figure below. The first prompt asks the user to select the data range (excluding any labels). The second prompt then requests the label option and the output location for displaying the results. Once all inputs are provided, the correlation matrix is generated at the specified location



	A	B	C
1	X	Y	Z
2	-0.20542	-0.246147	-11.54517
3	-0.199786	-0.030003	-0.481691
4	-0.341558	-0.263953	15.87768
5	-0.38138	-0.295126	1.686694
6	-0.089114	-0.221661	3.366647
7	0.220285	0.189275	0.878321
8	0.144879	0.142831	1.689995
9	-0.027566	0.07087	11.00167
10	0.379479	0.230036	-2.586658
11	0.61163	0.327231	-10.17498
12	0.738207	0.492307	-35.66331
13	0.096292	0.315674	-2.451638
14	-0.113277	-0.001212	8.660036
15	-0.155965	-0.134734	4.27171
16	-0.233136	-0.104493	8.683383
17	-0.250985	-0.136623	8.895057
18	-0.213558	-0.23173	6.90673
19	-0.241699	-0.239518	-3.781596
20	-0.056075	-0.060915	-1.669922
21	0.223611	0.133658	1.841751
22	0.30593	0.186961	-3.346575
23	-0.109995	0.016835	-1.134902
24	-0.166146	-0.187194	-0.923228

Correlation matrix			
	X	Y	Z
X	1	0.91671	-0.670854
Y	0.91671	1	-0.551956
Z	-0.670854	-0.551956	1

Figure 7: *Correlation matrix*

11. Covariance

This tool computes and displays the covariance matrix for a selected dataset.

Workflow: Upon clicking the **Covariance Matrix** button, the user is guided through two prompts:

1. Select the **data range** (numerical values only, no headers). Each column should represent a separate variable.

2. Enter the **label option** and the **output cell** as a comma-separated string.
   Example: `above, E2`

   - `above` - uses the row directly above the selected data as variable names
   - `none` - automatically assigns default labels `Var1`, `Var2`, etc.
   - `E2` - specifies the output cell for matrix placement

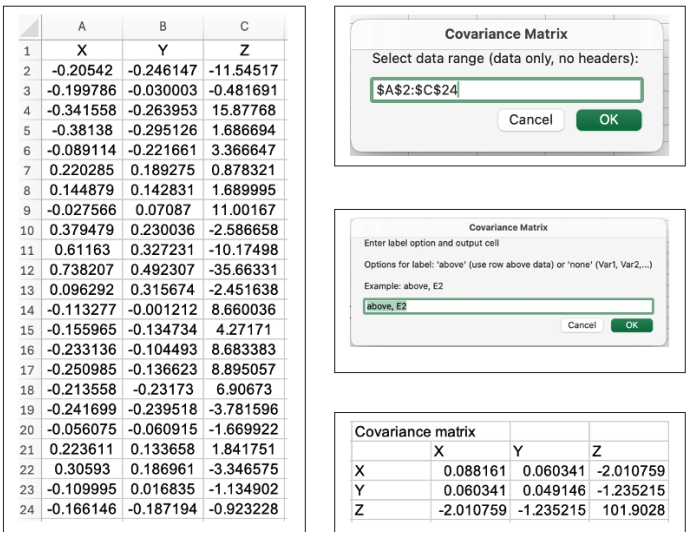
Output: The tool generates a covariance matrix starting from the selected output cell, including:

- A title: `Covariance matrix`
- Variable names along the top row and leftmost column
- Symmetric matrix of pairwise covariances among all variables in the selected range

The demo example is illustrated in the figure below. The first prompt asks the user to select the data range (excluding any labels). The second prompt then requests the label option and the output location for displaying the results. Once all inputs are provided, the covariance matrix is generated at the specified location.

12. Test Normality

This tool performs five normality tests described in Section 6.2 on a selected dataset and shows the results in a summary table. For each test, it provides the test statistic and the corresponding p-value to help decide whether the data is normally distributed.



	A	B	C
1	X	Y	Z
2	-0.20542	-0.246147	-11.54517
3	-0.199786	-0.030003	-0.481691
4	-0.341558	-0.263953	15.87768
5	-0.38138	-0.295126	1.686694
6	-0.089114	-0.221661	3.366647
7	0.220285	0.189275	0.878321
8	0.144879	0.142831	1.689995
9	-0.027566	0.07087	11.00167
10	0.379479	0.230036	-2.586658
11	0.61163	0.327231	-10.17498
12	0.738207	0.492307	-35.66331
13	0.096292	0.315674	-2.451638
14	-0.113277	-0.001212	8.660036
15	-0.155965	-0.134734	4.27171
16	-0.233136	-0.104493	8.683383
17	-0.250985	-0.136623	8.895057
18	-0.213558	-0.23173	6.90673
19	-0.241699	-0.239518	-3.781596
20	-0.056075	-0.060915	-1.669922
21	0.223611	0.133658	1.841751
22	0.30593	0.186961	-3.346575
23	-0.109995	0.016835	-1.134902
24	-0.166146	-0.187194	-0.923228

Covariance matrix			
	X	Y	Z
X	0.088161	0.060341	-2.010759
Y	0.060341	0.049146	-1.235215
Z	-2.010759	-1.235215	101.9028

Figure 8: *Covariance matrix*

Workflow: After clicking the **Normality Test** button, the user is asked to:

1. Select the data range (a single column of numbers, without headers).
2. Select the output location where the results should be written.

Output: The tool displays:

- The significance level (default $\alpha = 0.05$)
- Test names, test statistics, p-values (or critical values), and a conclusion string for normality

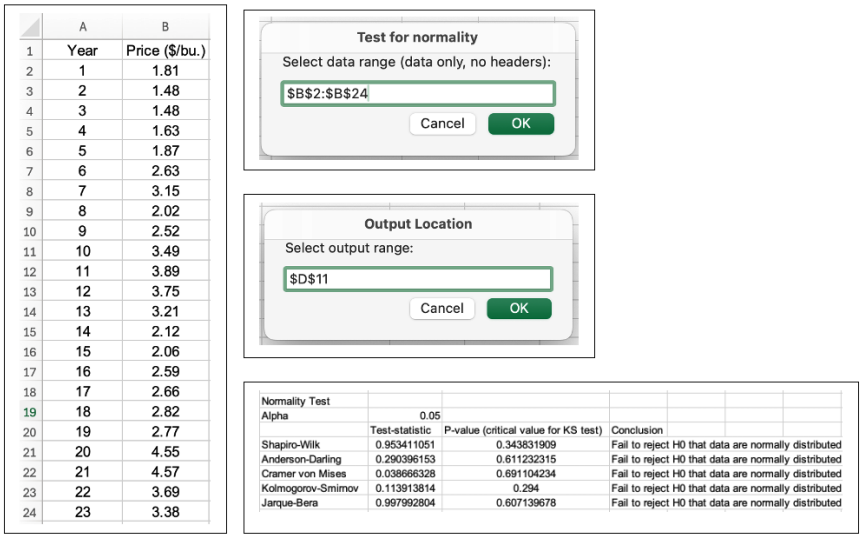
Included Tests:

- Shapiro-Wilk
- Anderson-Darling
- Cramer-von Mises
- Kolmogorov-Smirnov
- Jarque-Bera

Notes:

- The tests are applied to the original sample; no transformation or standardization is performed by default.
- The significance level can be manually changed in the worksheet by editing the `Alpha` cell.
- For details on each individual test, including formulas and statistical interpretation, refer to Sections 6.2.1, 6.2.2, 6.2.3, 6.2.4, and 6.2.5.

The demo example is illustrated in the figure below. The first prompt asks the user to select the data series (excluding any labels). The second prompt then requests the output location for displaying the results. Once all inputs are provided, the normality test results are generated at the specified location.



	A	B
1	Year	Price ($/bu.)
2	1	1.81
3	2	1.48
4	3	1.48
5	4	1.63
6	5	1.87
7	6	2.63
8	7	3.15
9	8	2.02
10	9	2.52
11	10	3.49
12	11	3.89
13	12	3.75
14	13	3.21
15	14	2.12
16	15	2.06
17	16	2.59
18	17	2.66
19	18	2.82
20	19	2.77
21	20	4.55
22	21	4.57
23	22	3.69
24	23	3.38

Normality Test			
Alpha	0.05		
	Test-statistic	P-value (critical value for KS test)	Conclusion
Shapiro-Wilk	0.953411051	0.343831909	Fail to reject H0 that data are normally distributed
Anderson-Darling	0.290396153	0.611232315	Fail to reject H0 that data are normally distributed
Cramer von Mises	0.038666328	0.691104234	Fail to reject H0 that data are normally distributed
Kolmogorov-Smirnov	0.113913814	0.294	Fail to reject H0 that data are normally distributed
Jarque-Bera	0.997992804	0.607139678	Fail to reject H0 that data are normally distributed

Figure 9: *Test for Normality*

13. Compare Two Series

Purpose: This subroutine performs a statistical comparison of two datasets - typically historical and simulated values - using appropriate univariate or multivariate hypothesis tests based on dimensionality. Univariate tests include Welch's t-test and an F-test for means and variances, while multivariate tests include Hotelling's T^2, Box's M, and a joint likelihood ratio test for homogeneity of means and covariances.

User Workflow:

1. Prompts the user to select the first data range (e.g., simulated series).
2. Prompts the user to select the second data range (e.g., historical series).
3. Prompts the user to select the output cell location.

Operation:

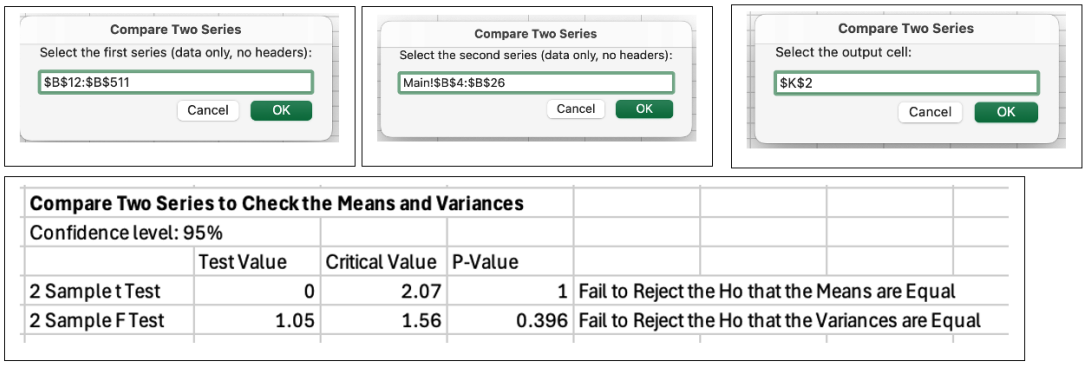
- **Univariate Case (when selecting only 1 Column):**
  - **Welch's Two-Sample *t*-Test** for comparing means (see section 6.3)
  - **F-Test** for comparing variances (see section 6.4)
- **Multivariate Case (when selecting 2 or more columns):**
  - **Hotelling's** T^2 test for mean vector equality (see section 6.5)
  - **Box's** M test for covariance matrix equality (see section 6.5)
  - **Complete Homogeneity Test** based on log-likelihood comparison of joint means and covariances (see section 6.5)

Output: Test results are displayed in the user selected output range, including the confidence level, test name, statistic, critical value, p-value, and decision outcome. Both input ranges must be aligned in shape (same number of columns).

The example in the figure below illustrates how the tool works:

- The first row shows the three prompts. First, the user selects the initial data range (without labels). Second, they select the comparison data range (also without labels). Third, they choose where to display the results. After these inputs, the results are generated at the chosen location.

- The second row shows an example where only one column is validated - comparing one simulated series to one historical series.
- The third row shows an example with multiple columns being validated.



Compare Two Series to Check the Means and Variances				
Confidence level: 95%				
	Test Value	Critical Value	P-Value	
2 Sample t Test	0	2.07	1	Fail to Reject the Ho that the Means are Equal
2 Sample F Test	1.05	1.56	0.396	Fail to Reject the Ho that the Variances are Equal

Compare Two Series to Check the Means and Variances				
Confidence level: 95.00%				
Test	Test Value	Critical Value	P-Value	Decision
2 Sample Hotelling T2 Test	1.1517	12.8199	0.9796	Fail to Reject the Ho that the Mean Vectors are Equal
Box's M Test	9.374	32.6706	0.986	Fail to Reject the Ho that the Covariance Matrices are Equal
Complete Homogeneity Test	10.5636	40.1133	0.9981	Fail to Reject the Ho that the Mean Vectors and Covariance Matrices are Equivalent, Respectively
*Based on asymptotic distribution				

Figure 10: *Compare Two Series*

14. Validate Correlation

This tool compares the simulated correlation matrix from a dataset to a historical (or reference) correlation matrix, and highlights statistically significant differences using a z-test. It helps identify whether simulated and historical correlation values are statistically equal.

Workflow: After clicking the **Validate Correlation** button, the user is prompted to:

1. Select the **data range** (simulated data range, data only, no headers).
2. Select the **historical correlation matrix** (a square matrix of the same column size, without labels).
3. Select the **output location** where the validation results will be displayed.

Procedure: The tool calculates the sample correlation matrix from the simulated data and compares it to the user-provided historical correlation matrix. For each pair of variables, it performs a z-test for the difference in correlation coefficients:

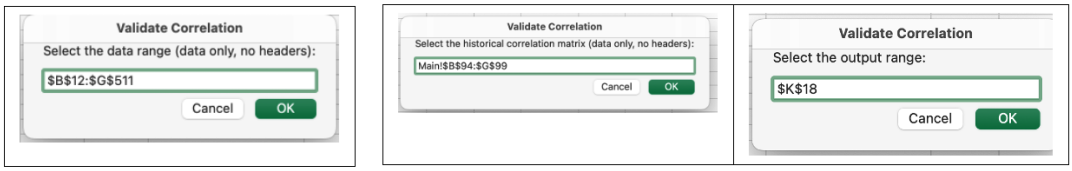
- Computes a z-score using the Fisher transformation
- Applies a two-tailed test at a fixed significance level ($\alpha = 0.01$)
- Highlights cells where the difference is statistically significant ($|z| > z_{\alpha/2}$)

Output: The output table includes:

- The selected confidence level and corresponding critical z-value displayed at the top.
- A matrix of absolute z-statistics comparing each pairwise correlation from the dataset against the historical matrix.
- Each cell in the lower triangle of the matrix shows the z-value for the test of difference in correlations.
- Cells where the absolute z-statistic exceeds the critical value are automatically highlighted in bold to indicate a significant difference.

The example in the figure below illustrates how the tool works:

- The first row shows the three user prompts. First, the user selects the simulated data range (excluding labels). Second, the historical correlation matrix is selected (also excluding labels). Third, the user specifies the output location where the results will be displayed. Once these inputs are provided, the validation output is generated at the selected location.
- The second row displays the validation results. It includes the confidence level and critical value at the top, followed by the individual z-statistics comparing each pair of variables between the two matrices.



Confidence Level	99.00%	Critical Value	2.58		
Test Corr Z	Corn price deviates	Soybean price deviates	Corn yield	Soybean yield	VC corn
Soybean price deviates	1.69				
Corn yield	0.16	0.07			
Soybean yield	1.36	0.92	0.65		
VC corn	0.63	0.65	0.6	0.71	
VC soybean	0.14	0.01	0.39	0.27	0.1

Figure 11: *Validate Correlation*

15. Graphs

2.0.1 15.a. PDF

This tool creates a probability density function (PDF) plot using kernel density estimation (KDE) for one or more simulated data series. It helps visualize the distribution of the selected data and outputs the corresponding chart. The procedure selects an appropriate bandwidth using a Silverman-like rule of thumb. For a single column input, vertical lines are added to mark the sample mean and the 2.5% and 97.5% quantiles.

Workflow: After clicking the **PDF Plot** button, the user is prompted to:

1. Select the **data range** (numerical values only, no headers).
2. Enter the **labeling option** and **output location**, separated by a comma.
   Example: `above, E2`
   - `above` - uses the row above the data range as labels
   - `none` - uses default labels such as `Series 1`, `Series 2`, etc.
   - `E2` - cell where the results will be placed and the chart aligned

Output: The following outputs are generated:

- A line chart showing the KDE estimate(s)
- Vertical markers for the mean and 95% interval (if applicable)

- A table to the right of the output cell, containing:
  - Grid of `X` values used for the KDE
  - Corresponding `PDF` values

Notes:

- The KDE is based on a Gaussian kernel and a rule-based bandwidth estimate.
- Each column in the selected range is treated independently and plotted as a separate series.

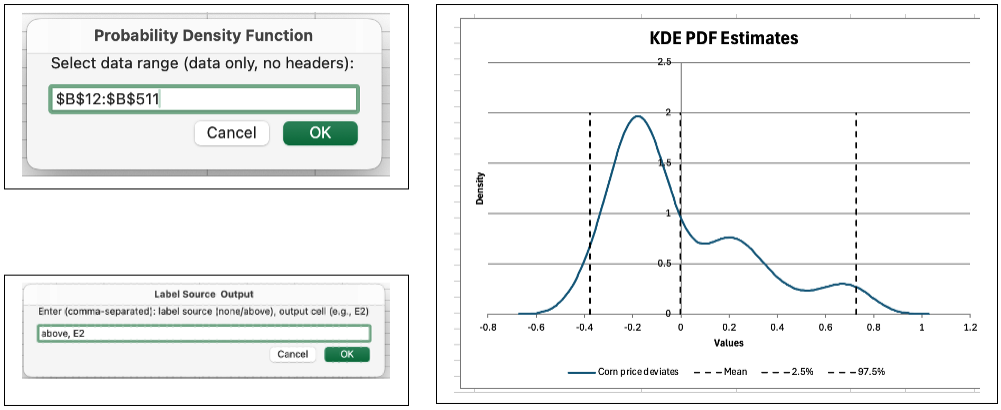


Figure 12: *Probability Density Function*

15.b. CDF

This tool creates an empirical cumulative distribution function (CDF) plot for one or more simulated data series. It helps visualize the probability distribution of each variable by showing how the cumulative probability increases with data values.

Workflow: After clicking the **CDF Plot** button, the user is prompted to:

1. Select the **data range** (numerical values only, no headers).
2. Enter the **labeling option** and **output location**, separated by a comma.
   Example: `above, E2`
   - `above` - uses the row above the data as column labels
   - `none` - applies default labels like `Series 1`, `Series 2`, etc.
   - `E2` - specifies the top-left output cell for the results and chart

Procedure: The tool calculates the empirical CDF as follows:

- Reads all valid numeric data from each column
- Sorts the values to construct the empirical distribution
- Calculates the cumulative probability at each data point
- Each data column is treated independently and plotted as a separate line on the same chart

Output: The following results are generated:

- A line chart titled `Empirical CDFs`, displaying one curve per series
- A corresponding table to the right of the output cell, listing:
  - Sorted `X` values for each series
  - Corresponding cumulative probabilities (from 0 to 1)

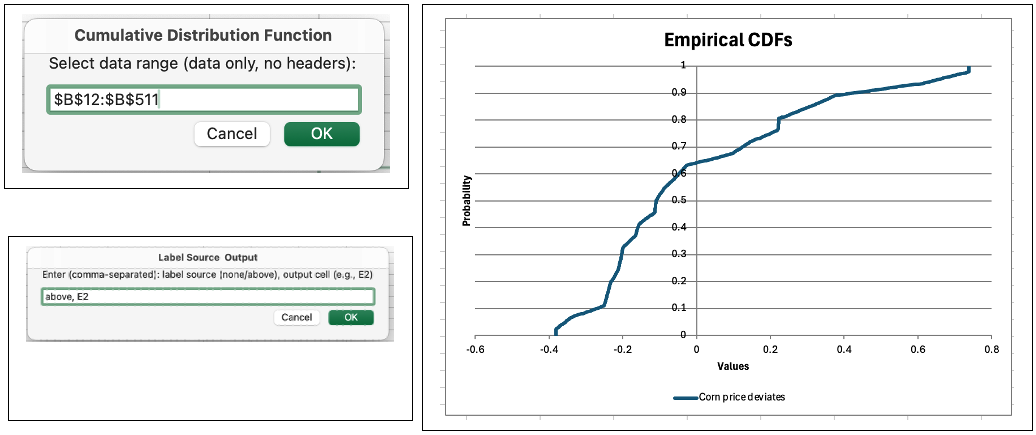


Figure 13: *Cumulative Distribution Function*

15.c. Stoplight

This tool creates a stacked column chart that visually summarizes the distribution of simulated values across three categories - red, yellow, and green - based on user-defined threshold values. The stoplight chart is useful for quickly identifying risk levels by showing the proportion of simulated values falling within each threshold category across multiple scenarios.

Workflow: After clicking the **Stoplight Chart** button, the user is prompted to:

1. Select the **data range** (numerical values only, no headers).
2. Enter the following four values, separated by commas:
   - **Lower bound**: values below this threshold are marked `Red`
   - **Upper bound**: values above this threshold are marked `Green`
   - **Label source**: either `above` or `none`
   - **Output cell**: Output range location for the chart placement

   Example input: `0, 100, above, E2`

Procedure: The tool examines each column (series) in the selected range and counts how many values fall into each of the three categories:

- **Red:** Values less than the lower bound

- **Yellow:** Values between the lower and upper bounds (inclusive)
- **Green:** Values greater than the upper bound

It then calculates the proportion of values in each category and builds a stacked column chart with labeled segments.

Output: The output is a stacked column chart with:

- One column per data series (e.g., scenario)
- Red, yellow, and green sections representing proportion of values by threshold
- Percentage labels displayed on each segment for comparison

Notes:

- Series names are automatically taken from the row above the data if `above` is selected; otherwise, generic labels such as `Option 1`, `Option 2`, etc. are used.
- The y-axis is fixed from 0 to 1 since the chart represents proportions.

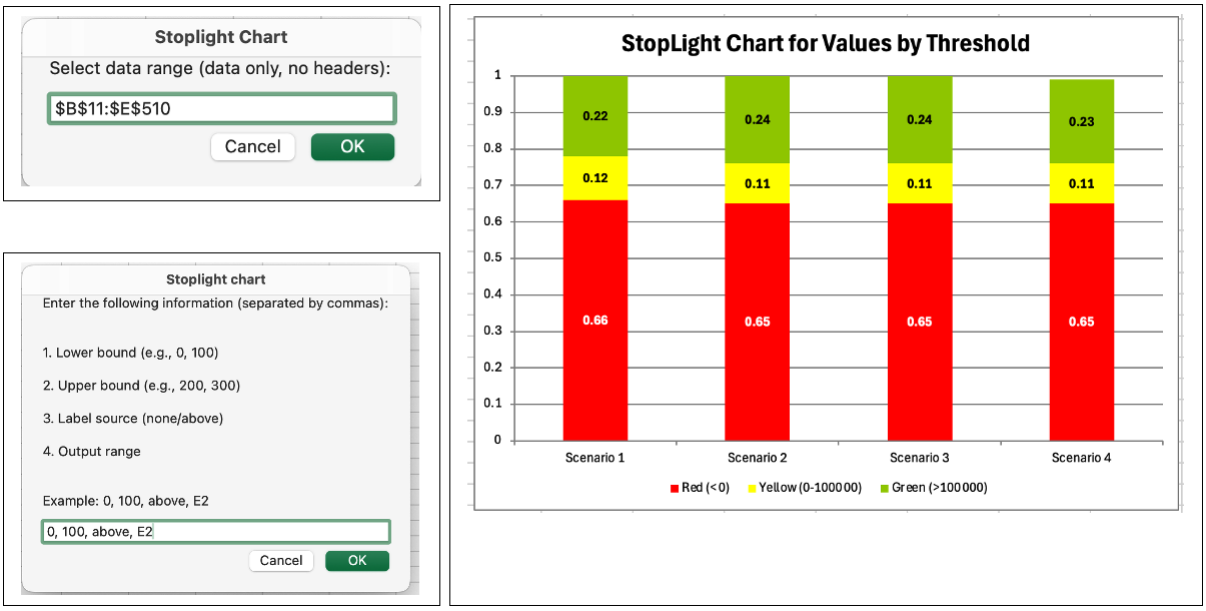


Figure 14: *Stoplight Chart*

15.d. Fan Graph

This tool creates a line chart with percentile bands (fan graph) to visualize the uncertainty in simulated results across multiple time periods or categories. The fan graph shows how distributions widen over time by displaying the 5th, 25th, 75th, and 95th percentiles along with the average.

Workflow: After clicking the **Fan Graph** button, the user is prompted to:

1. Select the **data range** (simulation trials in rows, periods in columns; no headers).
2. Enter the **output cell** (where the chart should be placed).

Procedure: For each column (representing a time period or category), the tool calculates:

- **Average**: mean value of the column
- **5th Percentile**: lower bound of the fan
- **25th Percentile**: inner lower bound
- **75th Percentile**: inner upper bound
- **95th Percentile**: upper bound of the fan

Labels for each period are taken from the row above the selected data range. If labels are missing, generic labels (`Year 1`, `Year 2`, etc.) are automatically assigned.

Output: The output is a line chart with:

- Solid black line for the **Average**
- Red dashed line for the **5th Percentile**
- Blue dashed line for the **25th Percentile**
- Green dashed line for the **75th Percentile**
- Dark red dashed line for the **95th Percentile**
- X-axis labeled by period (e.g., `Year 1`, `Year 2`, `Year 3`)
- Y-axis displaying numerical values with gridlines for readability

This makes it easy to see how uncertainty expands over time and compare the central tendency (average) against the distribution tails.

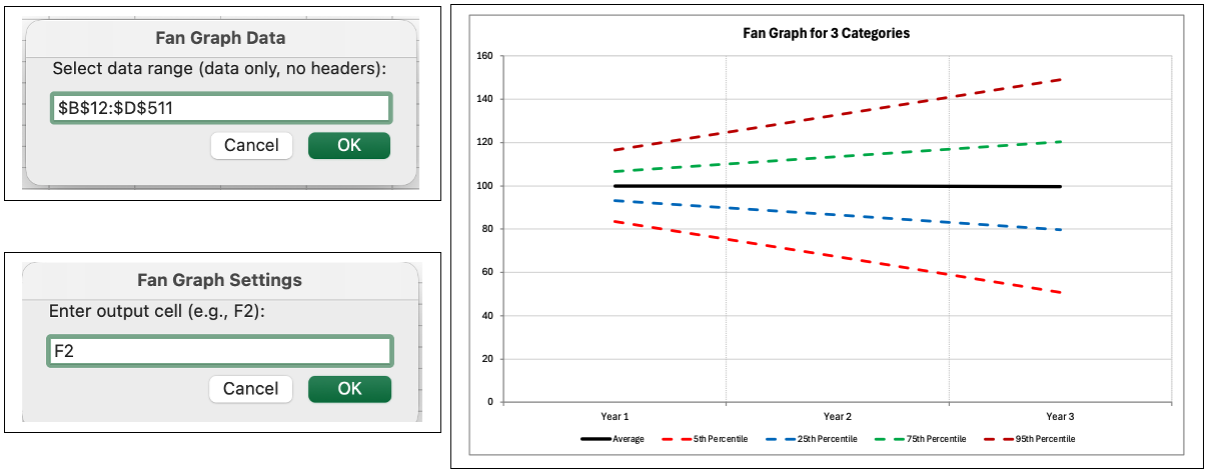


Figure 15: *Fan Graph with average and percentile lines*

15.e. Two–Way Scatter (Actual vs Simulated)

This tool creates an **XY scatter** chart that overlays two point clouds - *Actual* and *Simulated* - to visually compare the joint (X,Y) relationship from observed data against simulated output.

Workflow: After clicking the **Twoway Scatter** button, the user is prompted to:

1. Select the **ACTUAL** data range (two columns: no headers).
2. Select the **SIMULATED** data range (two columns: no headers).
3. Enter the **output cell** for chart placement (e.g., `F2`).

Procedure:

- The macro validates that each selected range has exactly two columns.
- It creates an XY scatter chart (markers only) and adds:
  - **Actual** series using the first selection (`X` = column 1, `Y` = column 2).
  - **Simulated** series using the second selection (`X` = column 1, `Y` = column 2).
- Gridlines and a legend are added for readability. The chart title defaults to *Actual vs Simulated*.

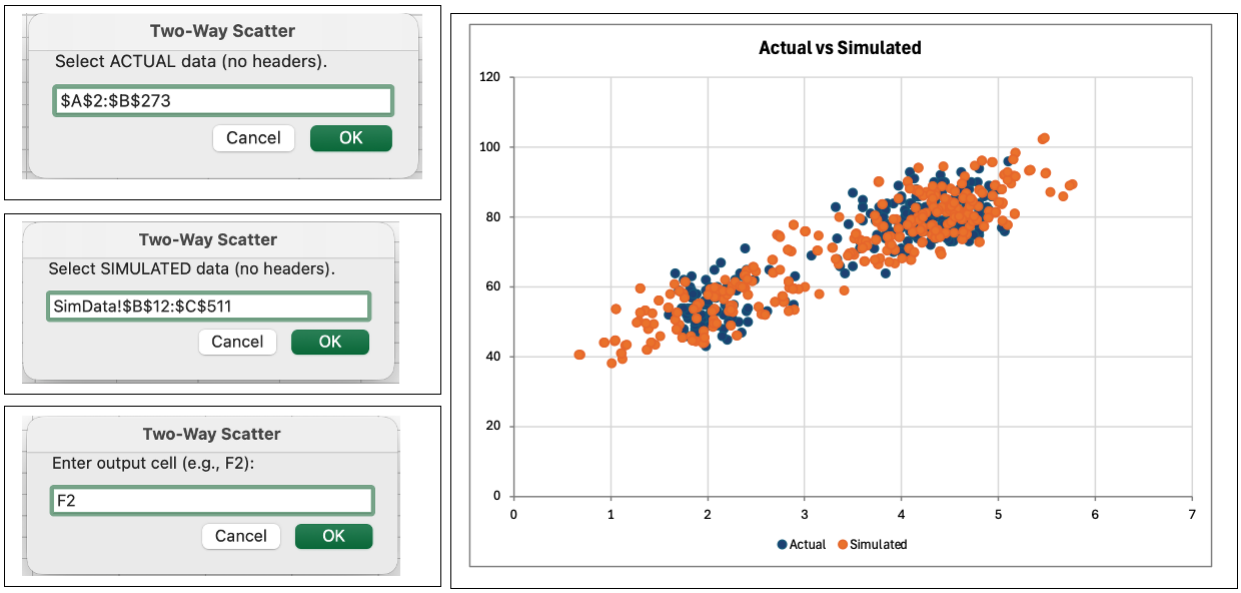


Figure 16: *Two-Way Scatter comparing Actual (blue) and Simulated (orange).*

16. Expected Utility

This tool calculates and visualizes certainty equivalent (CE) values based on expected utility theory. It evaluates CE values across a range of relative risk aversion coefficients (RRACs), using either the power or negative exponential utility function, and displays the results in both tabular and graphical formats.

Workflow: After clicking the **Expected Utility** button, the user is prompted through three inputs:

1. Select the **input data range** containing simulated outcome values (e.g., across multiple scenarios or distributions).

2. Select a 4-cell **parameter range** with the following:

   - Lower RRAC value (e.g., 0)
   - Upper RRAC value (e.g., 4)
   - Initial wealth (must be greater than zero)
   - Utility function type: `1` for Power utility, `2` for Negative Exponential utility, `3` for Normalized Log utility

3. Select the **output location** for displaying the results.

Output: Once the inputs are provided, the tool generates:

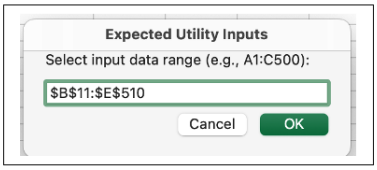
- A table with 25 equally spaced RRAC values between the specified lower and upper bounds in the first column.
- The remaining columns show the corresponding certainty equivalent (CE) values for each scenario, calculated using the selected utility function.
- A line chart titled *Certainty Equivalent Values*, with:
  - The horizontal axis showing the 25 RRAC values (levels of risk aversion)
  - The vertical axis showing the corresponding CE values
  - One line per scenario to compare how CE declines with increasing risk aversion

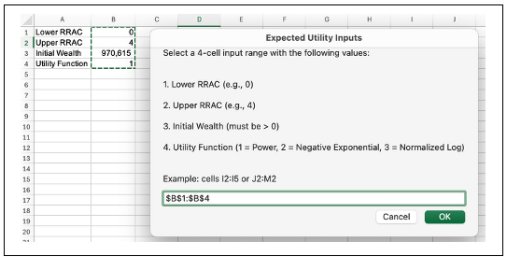
The figure below demonstrates how the tool works:

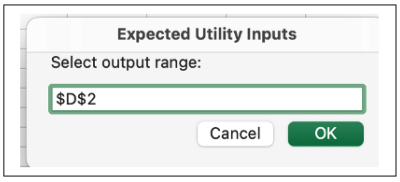
- The top row shows the three user prompts: the input data range, the 4 utility parameters, and the output cell.
- The table displays CE values calculated from the selected utility function across the specified RRAC interval.
- The chart on the right visualizes the CE values for each scenario as the degree of risk aversion increases.
- The y-axis of the chart represents CE values, while the x-axis shows increasing RRAC values.

Utility Function Details:

- The values are calculated using the `=SERF(...)` function (see section 8).
- For the Power utility function, see Section 8.1
- For the Negative Exponential utility function, see Section 8.2
- For the Normalized Log utility function, see Sectio 8.3
- For certainty equivalent calculations see Sections 8.1, 8.2 and 8.3.







Relative RAC	Scenario 1	Scenario 2	Scenario 3	Scenario 4
0.0	762,869	763,759	761,387	758,865
0.2	760,376	760,980	758,426	755,782
0.3	757,943	758,267	755,530	752,763
0.5	755,573	755,619	752,702	749,811
0.7	753,262	753,036	749,940	746,925
0.8	751,012	750,517	747,243	744,105
1.0	748,820	748,062	744,612	741,350
1.2	746,686	745,670	742,045	738,659
1.3	744,608	743,338	739,541	736,031
1.5	742,586	741,067	737,099	733,465
1.7	740,618	738,855	734,719	730,961
1.8	738,704	736,701	732,398	728,517
2.0	736,841	734,604	730,136	726,131
2.2	735,029	732,562	727,931	723,803
2.3	733,266	730,574	725,782	721,532
2.5	731,551	728,639	723,688	719,316
2.7	729,883	726,755	721,648	717,153
2.8	728,260	724,921	719,660	715,043
3.0	726,681	723,136	717,722	712,984
3.2	725,146	721,398	715,834	710,975
3.3	723,652	719,706	713,994	709,014
3.5	722,199	718,060	712,200	707,101
3.7	720,785	716,456	710,452	705,233
3.8	719,409	714,895	708,748	703,410
4.0	718,070	713,374	707,087	701,630

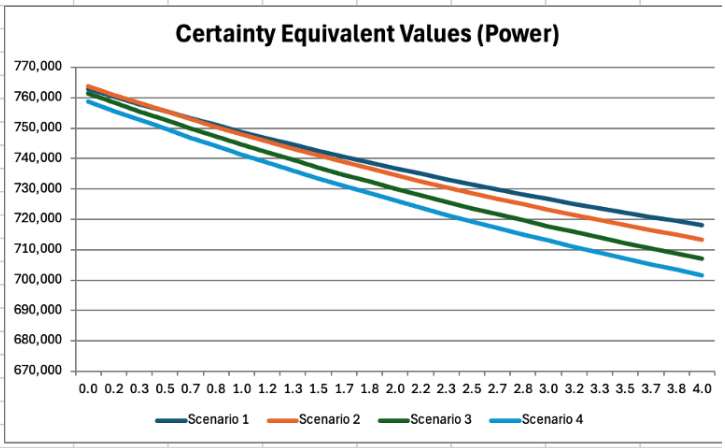


Figure 17: *Expected Utility*

17. Gaussian Copula

Overview

The `Gaussian Copula` ribbon button provides a tool for simulating a single multivariate draw from a set of correlated variables using the `CopulaMixed` function. This function performs Gaussian copula-based simulation with support for mixed marginal distributions. For technical details, see Section 5.5. The process requires three prompts.

Steps

1. **Select Data Range:** A rectangular range with historical data (excluding headers). Each column represents a variable.

2. **Select Marginal Types:** A single-row range with one string per column, indicating the marginal distribution type. Supported values are:

   - `Normal`
   - `Uniform`
   - `Beta`
   - `Empirical`
   - `KDE`

   If left blank, defaults to `Normal`. The values are case insensitive.

3. **Select Output Cell:** A single cell where the result will be written as a horizontal array formula (e.g., `=CopulaMixed(...)`).

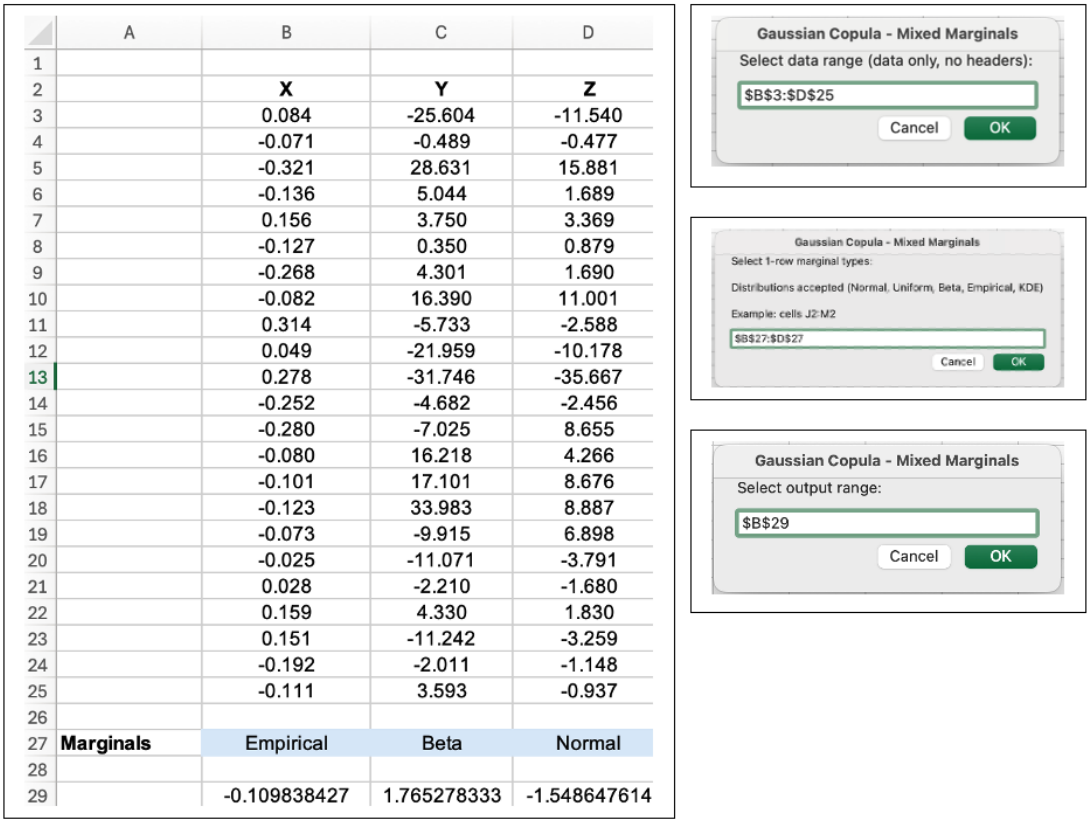
Output

The tool writes a single-row, $1 \times k$ array corresponding to one multivariate draw. The formula uses a spilling array of the form:

```
=CopulaMixed(dataRange, marginalTypes)
```

Example The example below demonstrates how the tool works:

- The three prompt boxes on the right show how the user selects: (i) the historical input data range, (ii) the marginal types, and (iii) the output location.
- The table on the left contains the original data (columns `X`, `Y`, and `Z`) and the marginal distribution types specified in the row labeled `Marginals`.
- The final row shows the simulated 1×3 draw returned by the copula-based simulation with mixed marginals.
- This result is generated by a single formula that automatically updates on recalculation (`F9`).



	A	B	C	D
1				
2		**X**	**Y**	**Z**
3		0.084	-25.604	-11.540
4		-0.071	-0.489	-0.477
5		-0.321	28.631	15.881
6		-0.136	5.044	1.689
7		0.156	3.750	3.369
8		-0.127	0.350	0.879
9		-0.268	4.301	1.690
10		-0.082	16.390	11.001
11		0.314	-5.733	-2.588
12		0.049	-21.959	-10.178
13		0.278	-31.746	-35.667
14		-0.252	-4.682	-2.456
15		-0.280	-7.025	8.655
16		-0.080	16.218	4.266
17		-0.101	17.101	8.676
18		-0.123	33.983	8.887
19		-0.073	-9.915	6.898
20		-0.025	-11.071	-3.791
21		0.028	-2.210	-1.680
22		0.159	4.330	1.830
23		0.151	-11.242	-3.259
24		-0.192	-2.011	-1.148
25		-0.111	3.593	-0.937
26				
27	**Marginals**	Empirical	Beta	Normal
28				
29		-0.109838427	1.765278333	-1.548647614

Figure 18: *Example: Gaussian Copula*

18. Cumulative Prospect Theory

The `CPT` ribbon runs the cumulative prospect theory calculation without typing the formula manually. It walks the user through three prompts to select the data, enter the key parameters, and choose where to place the output. Once completed, the formula is automatically added to the worksheet. For more details on how the `CPT` function works, see Section 9.

Usage Instructions: Upon clicking the `CPT` button in the Excel ribbon, the user is prompted to:

1. **Select a range of outcome values**, typically a multi column of simulated results, each column representing a specific scenario (e.g., `B11:F510`).
2. **Select a 4-cell range of parameter values**, corresponding to:
   - `gammaGain` (probability weighting for gains)
   - `gammaLoss` (probability weighting for losses)
   - `lambda` (loss aversion coefficient)
   - `alpha` (power utility exponent)
3. **Enter four comma-separated settings** into a text prompt:
   (a) Weighting function: `prelec` or `tk`
   (b) Include certainty equivalent? (`TRUE` or `FALSE`)
   (c) Include headers? (`TRUE` or `FALSE`)
   (d) Output cell address (e.g., `E3`)

Output: The macro automatically inserts the corresponding `=CPT(...)` formula into the specified output cell. The number of returned rows and columns depends on the selected options for certainty equivalent and headers.

Example The figure below demonstrates how the `CPT` ribbon button works in Excel:

- The user is guided through three prompts:

1. Selection of the input data range (e.g., 500 simulated trials across 5 scenarios).
2. Selection of a 4-cell parameter block: `gammaGain`, `gammaLoss`, `lambda`, and `alpha` (shown in cells `I2:I5`).
3. Manual entry of a 4-value configuration string: weighting function, include CE, include headers, and output cell (e.g., `prelec, TRUE, FALSE, H10`).

- The function automatically outputs CPT values and certainty equivalents, depending on the selected options, directly into the target output range.
- Each CPT value is calculated using the selected probability weighting function (`prelec` in this case) and power utility parameters.
- The formula bar shows the resulting `=CPT(...)` formula for each scenario. It dynamically references the data and parameter cells.

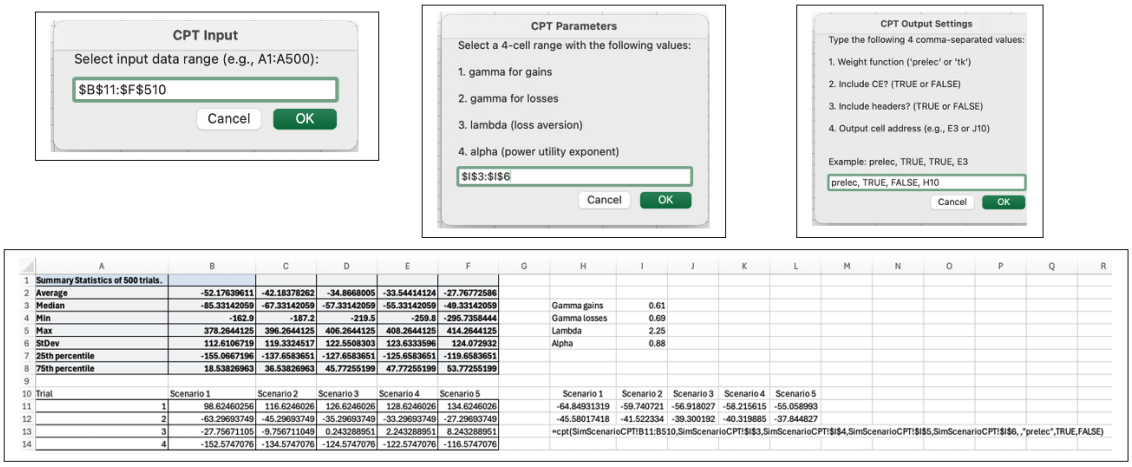


Figure 19: *Example use of the `CPT` ribbon button to evaluate multiple scenarios.*

Note:

- Ensure the input data is structured with **scenarios in columns** and **trials in rows**.
- Parameter values must be provided in the exact order listed above: γ^+, γ^-, λ, and α.
- The choice of weighting function and parameter values can substantially affect the output. Users are encouraged to consult relevant literature for guidance on selecting appropriate values.
- For detailed customization options and technical explanation, see Section 9.

3 Random Numbers and Probability Distributions

3.1 Parametric Distributions

3.1.1 Bernoulli

The `BERNOULLI` function generates a binary outcome (0 or 1) from a Bernoulli distribution with a user-specified probability of success.

Function Signature:

```
=BERNOULLI(probability, [usd])
```

Inputs:

- `probability`: The probability of success (p), where $0 \leq p \leq 1$.
- `usd` (optional): A user-supplied uniform deviate on $(0, 1)$. If omitted, one is generated internally.

Output: Returns:

$$X = \begin{cases} 1, & \text{if } U \leq p \\ 0, & \text{otherwise} \end{cases}$$

where $U \sim \text{Uniform}(0, 1)$.

Example: Simulating 1 trial of a Bernoulli distribution with success probability 0.3:

```
=BERNOULLI(0.3)
```

Each evaluation returns either 0 or 1, depending on the comparison between the probability input and the uniform draw.

	A	B	C	D
1	Probability	0.3		
2	Uniform draw	0.120613	=UNIFORM()	
3				
4		1	=bernoulli(B1, B2)	

Figure 20: *Example: Bernoulli distribution applied with probability=0.3*

3.1.2 Beta

The `BETA` function generates a random value from a scaled Beta distribution over a specified interval $[\min, \max]$, using inverse transform sampling.

Distribution Overview: The Beta distribution is a flexible two-parameter distribution commonly used to model variables bounded between two finite values. The probability density is defined on $[0, 1]$ for standard Beta, and can be rescaled to $[\min, \max]$:

$$X = \min + (\max - \min) \cdot B, \quad B \sim \text{Beta}(\alpha, \beta)$$

Function Signature:

```
=BETA(alpha, beta, min, max, [usd])
```

Inputs:

- `alpha`, `beta`: Shape parameters ($\alpha > 0$, $\beta > 0$).
- `min`, `max`: Lower and upper bounds for scaling.
- `usd` (optional): A user-supplied uniform deviate on $(0, 1)$. If omitted, one is generated internally.

Output: Returns a single draw from the scaled Beta distribution using:

$$X = \min + (\max - \min) \cdot B, \quad B \sim \text{Beta}(\alpha, \beta)$$

Parameter Estimation: `FITBETA_MOM` The `FITBETA_MOM` function estimates Beta distribution parameters from raw data using the method of moments. It returns a 4-row, 2-column array with:

- Estimated $\hat{\alpha}$ and $\hat{\beta}$
- Minimum and maximum values of the data (used for scaling)

Estimation Procedure:

1. Let y_i denote raw data points.

2. Calculate $\min(y)$ and $\max(y)$.

3. Rescale data into $(0, 1)$ with:

$$x_i = \frac{y_i - \min + \varepsilon}{\max - \min + 2\varepsilon}, \quad \varepsilon \ll 1$$

4. Calculate sample mean $\bar{x}$ and variance s^2.

5. Estimate:

$$\hat{\alpha} = \bar{x}\left(\frac{\bar{x}(1-\bar{x})}{s^2} - 1\right), \quad \hat{\beta} = (1-\bar{x})\left(\frac{\bar{x}(1-\bar{x})}{s^2} - 1\right)$$

If $s^2 = 0$, the function defaults to $\alpha = \beta = 1$.

Example: The figure below demonstrates how `FITBETA_MOM` can be used to fit a Beta distribution to data (column A), and how the estimated parameters are used to generate one random draw using the `BETA` function.

```
=FITBETA_MOM(A2:A24)
=BETA(C3, C4, C5, C6)
```

- Column A contains the observed sample.
- Cells C3–C6 contain $\hat{\alpha}$, $\hat{\beta}$, $\min(y)$, and $\max(y)$.
- Cell C8 calculates a single simulated draw using the fitted Beta distribution.

	A	B	C	D	E
1	Y				
2	-11.55	**Beta parameters estimation**			
3	-0.48	alpha	3.153017223	=FITBETA_MOM(A2:A24)	
4	15.88	beta	1.403756702		
5	1.69	min	-35.66331111		
6	3.37	max	15.87768372		
7	0.88				
8	1.69	-3.942911	=beta(C3,C4,C5,C6)		
9	11.00				
10	-2.59				
11	-10.17				
12	-35.66				
13	-2.45				
14	8.66				
15	4.27				
16	8.68				
17	8.90				
18	6.91				
19	-3.78				
20	-1.67				
21	1.84				
22	-3.35				
23	-1.13				
24	-0.92				

Figure 21: *Example: Fitting and Sampling from a Beta Distribution using* `FITBETA_MOM` *and* `BETA`

3.1.3 Gamma (3-Parameter)

The `GAMMA_INV` function generates a random value from a **shifted Gamma distribution** with shape, scale, and location parameters using inverse transform sampling.

Distribution Overview: The 3-parameter Gamma distribution is defined over $(\text{location}, \infty)$ and generalizes the standard Gamma by adding a location (shift) parameter:

$$X \sim \text{Gamma}(\alpha, \theta, \text{location}) \iff X - \text{location} \sim \text{Gamma}(\alpha, \theta)$$

Its probability density function (PDF) is:

$$f_X(x) = \begin{cases} 0 & x \leq \text{location} \\ \dfrac{(x - \text{location})^{\alpha-1} e^{-(x-\text{location})/\theta}}{\Gamma(\alpha)\,\theta^{\alpha}} & x > \text{location} \end{cases}$$

Function Signature:

```
=GAMMA_INV(shape, scaleParam, location, [usd])
```

Inputs:

- `shape`: Shape parameter $\alpha > 0$
- `scaleParam`: Scale parameter $\theta > 0$
- `location`: Location (shift) parameter
- `usd` (optional): Uniform standard deviate. If omitted, one is generated internally.

Output: Returns:

$$X = \text{Gamma}^{-1}(U; \alpha, \theta) + \text{location}$$

Notes:

- If `shape` ≤ 0 or `scale` ≤ 0, the function returns `#N/A`.

Parameter Estimation: The `FITGAMMA_MOM` function estimates the parameters of a 3-parameter Gamma distribution using the method of moments. It shifts the data so all values are strictly positive, then fits a 2-parameter Gamma distribution to the shifted data.

Function Signature:

```
=FITGAMMA_MOM(dataRange)
```

Inputs:

- `dataRange`: A one-column range of numerical values (can include negative values).

Output: Returns a 3-row, 2-column array:

- Row 1: Label `"shape"` and estimate $\hat{\alpha}$
- Row 2: Label `"scale"` and estimate $\hat{\theta}$
- Row 3: Label `"location"` and the shift amount $\hat{\text{location}}$

Important: The location parameter represents the shift amount. To recover the original data scale from a generated value X, use: $X_{\text{original}} = X_{\text{gamma}} + \text{location}$

Estimation Procedure:

1. Define the shift amount:

$$\text{location} = \min(y_i) - \varepsilon, \quad \varepsilon = 10^{-6}$$

   where location represents the amount subtracted from the original data.

2. Shift the data to positive domain: $x_i = y_i - \text{location}$

3. Calculate sample mean $\bar{x}$ and variance s^2 of the shifted data

4. Estimate parameters using method of moments:

$$\hat{\theta} = \frac{s^2}{\bar{x}}, \quad \hat{\alpha} = \frac{\bar{x}^2}{s^2}$$

Example: The example below shows data with values ranging from -0.32 to 0.31. To estimate the parameters and generate a stowchastic draw, two functions are used. The `FITGAMMA_MOM` returns shape, scale, location parameters. The `GAMM_INV` generates random value using estimated parameters.

```
=FITGAMMA_MOM(A2:A24)
=GAMMA_INV(D2, D3, D4)
```

- Column A: Original observed data (including negative values)
- D2: Estimated shape parameter $\hat{\alpha}$
- D3: Estimated scale parameter $\hat{\theta}$
- D4: Estimated location parameter $\hat{\text{location}} \approx -0.32 - 10^{-6}$
- D7: Simulated draw from the shifted Gamma distribution (will be on original data scale)

	A	B	C	D	E	F	G
1	Y						
2	0.08		shape	2.531362	=FITGAMMA_MOM(A2:A24)		
3	-0.07		scale	0.109197			
4	-0.32		location	-0.32098			
5	-0.14						
6	0.16						
7	-0.13		Gamma	-0.082975	=GAMMA_INV(D2,D3,D4)		
8	-0.27						
9	-0.08						
10	0.31						
11	0.05						
12	0.28						
13	-0.25						
14	-0.28						
15	-0.08						
16	-0.10						
17	-0.12						
18	-0.07						
19	-0.03						
20	0.03						
21	0.16						
22	0.15						
23	-0.19						
24	-0.11						

Figure 22: *Example: Estimation and Sampling from a Shifted Gamma Distribution using* `FITGAMMA_MOM` *and* `GAMMA_INV`

3.1.4 GRKS

GRKS distribution was developed by Gray, Richardson, Klose, and Schumann (Richardson, 2008). The `GRKS` function generates random values from a piecewise approximation based on the standard normal distribution. The distribution uses three user-defined points - minimum, midpoint, and maximum - to simulate subjective probability distributions based on minimal input data (Richardson, 2008).

Function Signature:

```
=GRKS(minimum, midpoint, maximum, [usd])
```

Inputs:

- `minimum`: The lowest plausible value of the variable.
- `midpoint`: The most likely value (mode or peak).
- `maximum`: The highest plausible value.
- `usd` (optional): A user-supplied uniform value on $(0, 1)$. If omitted, one is internally generated.

Output: The estimation procedure follows Richardson (2008, Chapter 5, p. 3-4). The function returns a value drawn from a custom distribution derived by aligning a 13-point scale to standard normal quantiles. The distribution is piecewise linear between the following:

- 13 z-scores spanning $(-\infty, +\infty)$ with breakpoints at -2.5, -2, -1.5, ..., $+2.5$.
- 13 corresponding x-values, defined using:
  - Lower interval width: $\frac{\text{midpoint}-\text{minimum}}{4}$
  - Upper interval width: $\frac{\text{maximum}-\text{midpoint}}{4}$
  - Intervals extend beyond the minimum and maximum to match tails of the normal distribution. That is, 2.2 percent of the simulated observations are less than the minimum and 2.2 percent are greater than the maximum (Richardson, 2008).

The generated value is interpolated between the bracketing percentiles based on the input (or generated) uniform draw U.

How it Works:

1. Constructs a 13-point support vector X_i from minimum, midpoint, and maximum.
2. Assigns each X_i a cumulative probability $P(X_i)$ based on the standard normal CDF at z_i.
3. For a given uniform deviate $U \in (0, 1)$, the function locates the bracket (X_{i-1}, X_i) such that $P(X_{i-1}) \leq U \leq P(X_i)$.
4. It interpolates linearly between those two X-values to return the random value.

Example: The example below illustrates a single draw from the GRKS distribution using the parameters:

- Minimum = 5
- Midpoint = 10
- Maximum = 17

The uniform deviate is omitted and generated internally.

	A	B	C	D
1	Minimum	5		
2	Midpoint	10		
3	Maximum	17		
4				
5		9.971444	=grks(B1, B2, B3)	

Figure 23: *Example: GRKS distribution applied with (min, midpoint, max) = (5, 10, 17).*

Remarks:

- Unlike traditional distributions, GRKS does not rely on fitting a named statistical family. Instead, it generates a smooth approximation based on user-defined minimum, midpoint, and maximum values mapped to standard normal cumulative probabilities. This structure makes it especially valuable in expert-driven or judgment-based simulations where formal parameter estimation is not feasible.
- GRKS is particularly useful when historical data is limited or unavailable. In such cases, users can provide plausible lower and upper bounds along with a most-likely value (mode), enabling simulation of uncertainty even in the absence of detailed distributional information.

References

- Richardson, J.W. (2008). *Simulation for Applied Risk Management with an Introduction to SIMETAR*. Department of Agricultural Economics. Texas A&M University.

3.1.5 Gumbel (Extreme Value Type I)

The `GUMBEL` function generates a random value from a **Gumbel distribution** using inverse transform sampling. This distribution models the maximum (or minimum) of a large number of i.i.d. variables and is commonly used in extreme value theory.

Distribution Overview: The Gumbel distribution is defined over $(-\infty, \infty)$ and is characterized by a location parameter μ and a scale parameter $b > 0$:

$$X \sim \text{Gumbel}(\mu, b)$$

Its probability density function (PDF) is:

$$f_X(x) = \frac{1}{b} \exp\left(-\frac{x-\mu}{b} - \exp\left(-\frac{x-\mu}{b}\right)\right)$$

Its cumulative distribution function (CDF) is:

$$F_X(x) = \exp\left(-\exp\left(-\frac{x-\mu}{b}\right)\right)$$

Function Signature:

```
=GUMBEL(location, scaleParam, [usd])
```

Inputs:

- `location`: Location parameter μ (controls the center of the distribution)
- `scaleParam`: Scale parameter $b > 0$ (controls the spread)
- `usd` (optional): Uniform standard deviate. If omitted, the function generates one internally.

Output: Returns a value sampled using the inverse transform:

$$X = \mu - b \cdot \ln\left(-\ln(U)\right)$$

Parameter Estimation: The `FITGUMBEL_MOM` function estimates the location and scale parameters of a Gumbel distribution using the method of moments. It uses the known mean and variance formulas of the Gumbel distribution to derive parameter estimates.

Function Signature:

```
=FITGUMBEL_MOM(dataRange)
```

Inputs:

- `dataRange`: A one-column range of numerical values.

Output: Returns a 2-row, 2-column array:

- Row 1: Label `"location"` and estimate $\hat{\mu}$
- Row 2: Label `"scale"` and estimate $\hat{b}$

Estimation Procedure: The method of moments uses the following relationships:

$$\mathbb{E}[X] = \mu + b \cdot \gamma, \qquad \mathrm{Var}(X) = \frac{\pi^2}{6} b^2$$

where $\gamma \approx 0.5772$ is the Euler–Mascheroni constant.

Steps:

1. Calculate sample mean $\bar{x}$ and sample variance s^2
2. Estimate scale and location:

$$\hat{b} = \sqrt{\frac{6s^2}{\pi^2}}, \qquad \hat{\mu} = \bar{x} - \hat{b} \cdot \gamma$$

Example: The figure below demonstrates how `FITGUMBEL_MOM` is used to estimate parameters from raw data (Column A), and how those parameters are passed to `GUMBEL` to generate simulated draws.

```
=FITGUMBEL_MOM(A2:A24)
=GUMBEL(D2, D3)
```

- Column A: Original observed data.
- D2: Estimated location parameter $\hat{\mu}$.
- D3: Estimated scale parameter $\hat{b}$.
- D7: Simulated draw from the Gumbel distribution.

	A	B	C	D	E	F	G
1	Y						
2	-25.60		location	-7.112497	=FITGUMBEL_MOM(A2:A24)		
3	-0.49		scale	12.32208			
4	28.63						
5	5.04						
6	3.75						
7	0.35		Gumbel	7.314283	=GUMBEL(D2,D3)		
8	4.30						
9	16.39						
10	-5.73						
11	-21.96						
12	-31.75						
13	-4.68						
14	-7.03						
15	16.22						
16	17.10						
17	33.98						
18	-9.92						
19	-11.07						
20	-2.21						
21	4.33						
22	-11.24						
23	-2.01						
24	3.59						

Figure 24: *Example: Estimation and Sampling from a Gumbel Distribution using* `FITGUMBEL_MOM` *and* `GUMBEL`

Notes:

- The Gumbel distribution is skewed (right-skewed by default).
- It is suitable for modeling extreme maximum values (e.g., rainfall, temperatures).
- The variance is fixed relative to the scale via $\text{Var}(X) = \frac{\pi^2}{6}b^2$.
- The standard Gumbel distribution models extreme *maximum* values and is right-skewed. If data instead represent extreme *minimums* and are left-skewed, Gumbel distribution can be fit by first multiplying each value by -1. Fit the Gumbel distribution to the reflected data, and then multiply the simulated values by -1 to convert them back to the original (minimum) direction.

3.1.6 Johnson's SU

The `JOHNSONSU` function generates a random value from a Johnson's SU distribution using inverse transform sampling.

Distribution Overview: The Johnson's SU distribution is a flexible four-parameter distribution that can model a wide variety of shapes, including symmetric, skewed, light-tailed,

and heavy-tailed distributions. The distribution is particularly useful in modeling crop yield residuals, which often exhibit asymmentry (e.g., bad years cause large negative residuals). The distribution is defined by transforming a standard normal variable $Z \sim N(0,1)$ using the hyperbolic sine function:

$$X = \xi + \sigma \cdot \sinh\left(\frac{Z - \gamma}{\delta}\right)$$

where $\sinh(u) = \frac{e^u - e^{-u}}{2}$ is the hyperbolic sine function.

Function Signature:

```
=JOHNSONSU(a, b, loc, scale, [usd])
```

Inputs:

- `a` (γ): Shape parameter controlling skewness (can be any real number).
- `b` (δ): Shape parameter controlling tail behavior ($\delta > 0$).
- `loc` (ξ): Location parameter (can be any real number).
- `scale` (σ): Scale parameter ($\sigma > 0$).
- `usd` (optional): A user-supplied uniform deviate on $(0,1)$. If omitted, one is generated internally.

Output: Returns a single draw from the Johnson's SU distribution using:

$$X = \xi + \sigma \cdot \sinh\left(\frac{\Phi^{-1}(U) - \gamma}{\delta}\right)$$

where U is uniform on $(0,1)$ and Φ^{-1} is the inverse standard normal CDF.

Parameter Estimation: The `FITJOHNSONSU_MLE` function estimates Johnson's SU distribution parameters from raw data using Maximum Likelihood Estimation (MLE). It returns a 4-row, 2-column array with:

- Estimated shape parameter $\hat{a}$ ($\hat{\gamma}$)
- Estimated shape parameter $\hat{b}$ ($\hat{\delta}$)
- Estimated location parameter $\hat{\text{loc}}$ ($\hat{\xi}$)
- Estimated scale parameter $\hat{\text{scale}}$ ($\hat{\sigma}$)

Parameter Interpretation:

- **Shape parameter** a **(**γ**)**: Controls skewness
  - $a = 0$: Symmetric distribution
  - $a > 0$: Right-skewed (positive skew)
  - $a < 0$: Left-skewed (negative skew)
- **Shape parameter** b **(**δ**)**: Controls tail behavior and kurtosis
  - Larger b: Lighter tails, more normal-like
  - Smaller b: Heavier tails, more extreme values
- **Location parameter loc (**ξ**)**: Shifts the distribution horizontally
- **Scale parameter scale (**σ**)**: Controls the spread of the distribution

Example: The figure below demonstrates how `FITJOHNSONSU_MLE` can be used to fit a Johnson SU distribution to data (column A), and how the estimated parameters are used to generate one random draw using the `JOHNSONSU` function.

```
=FITJOHNSONSU_MLE(A2:A24)
=JOHNSONSU(C3, C4, C5, C6)
```

- Column A: observed data
- Cells C3:C6: estimated values $\hat{a}, \hat{b}, \hat{\xi}, \hat{\sigma}$
- Cell C8 calculates a single simulated draw using the fitted Johnson SU distribution.

Notes:

- MLE is performed using the Nelder-Mead simplex algorithm.
- The multi-start approach is implemented to improve global convergance for reliable parameter estimation, especially with complex or multimodal data.
- The methodology is adapted from SciPy's `scipy.stats.johnsonsu.fit()` function and the results should be similar to those of SciPy's.

	A	B	C	D	E
1	Y				
2	-11.55	**Johnson's SU parameters estimation**			
3	-0.48	a	0.13855595	=FitJohnsonSU_MLE(A2:A24)	
4	15.88	b	1.029524624		
5	1.69	loc	1.727457312		
6	3.37	scale	5.894987139		
7	0.88				
8	1.69	-3.397577	=JohnsonSU(C3,C4,C5,C6)		
9	11.00				
10	-2.59				
11	-10.17				
12	-35.66				
13	-2.45				
14	8.66				
15	4.27				
16	8.68				
17	8.90				
18	6.91				
19	-3.78				
20	-1.67				
21	1.84				
22	-3.35				
23	-1.13				
24	-0.92				

Figure 25: *Example: Fitting and Sampling from a Johnson SU Distribution using* `FITJOHNSONSU_MLE` *and* `JOHNSONSU`

3.1.7 Laplace (Double Exponential)

The `LAPLACE` function generates a random value from a **Laplace distribution** with location and scale parameters using inverse transform sampling.

Distribution Overview: The Laplace distribution, also known as the double exponential distribution, is defined over $(-\infty, \infty)$ and is characterized by two parameters:

$$X \sim \text{Laplace}(\mu, b)$$

Its probability density function (PDF) is:

$$f_X(x) = \frac{1}{2b} \exp\left(-\frac{|x-\mu|}{b}\right), \quad x \in \mathbb{R}$$

Its cumulative distribution function (CDF) is:

$$F_X(x) = \begin{cases} \frac{1}{2}\exp\left(\frac{x-\mu}{b}\right) & x < \mu \\ 1 - \frac{1}{2}\exp\left(-\frac{x-\mu}{b}\right) & x \geq \mu \end{cases}$$

Function Signature:

```
=LAPLACE(mu, b, [usd])
```

Inputs:

- `mu`: Location parameter μ (median of the distribution)
- `b`: Scale parameter $b > 0$ (controls spread)
- `usd` (optional): Uniform standard deviate. If omitted, one is generated internally.

Output: Returns a random value using inverse transform sampling:

$$X = \begin{cases} \mu + b\ln(2U) & U < 0.5 \\ \mu - b\ln(2 - 2U) & U \geq 0.5 \end{cases}$$

Parameter Estimation: The `FITLAPLACE_MOM` function estimates the parameters of a Laplace distribution using the method of moments. The location parameter is estimated by the sample median, and the scale parameter by the mean absolute deviation from the median.

Function Signature:

```
=FITLAPLACE_MOM(dataRange)
```

Inputs:

- `dataRange`: A one-column range of numerical values.

Output: Returns a 2-row, 2-column array:

- Row 1: Label `"mu"` and estimate $\hat{\mu}$
- Row 2: Label `"b"` and estimate $\hat{b}$

Estimation Procedure:

1. Estimate location parameter using the sample median:

$$\hat{\mu} = \text{median}(y_1, y_2, \ldots, y_n)$$

2. Estimate scale parameter using mean absolute deviation from median:

$$\hat{b} = \frac{1}{n} \sum_{i=1}^{n} |y_i - \hat{\mu}|$$

Example: The figure below demonstrates how `FITLAPLACE_MOM` is used to estimate parameters from raw data (Column A), and how those parameters are passed to `LAPLACE` to generate simulated draws.

```
=FITLAPLACE_MOM(A2:A24)
=LAPLACE(D2, D3)
```

- Column A: Original observed data.
- D2: Estimated location parameter $\hat{\mu}$ (median).
- D4: Estimated scale parameter $\hat{b}$ (mean absolute deviation).
- D7: Simulated draw from the Laplace distribution.

	A	B	C	D	E	F	G
1	Y						
2	-25.60		mu	-0.489459	=FITLAPLACE_MOM(A2:A24)		
3	-0.49		b	11.60392			
4	28.63						
5	5.04						
6	3.75						
7	0.35		Laplace	1.487651	=LAPLACE(D2,D3)		
8	4.30						
9	16.39						
10	-5.73						
11	-21.96						
12	-31.75						
13	-4.68						
14	-7.03						
15	16.22						
16	17.10						
17	33.98						
18	-9.92						
19	-11.07						
20	-2.21						
21	4.33						
22	-11.24						
23	-2.01						
24	3.59						

Figure 26: *Example: Estimation and Sampling from a Laplace Distribution using `FITLAPLACE_MOM` and `LAPLACE`*

Notes:

- The median equals the location parameter: $\text{median}(X) = \mu$
- The mean absolute deviation from median equals the scale parameter: $E[|X - \mu|] = b$
- The Laplace distribution is symmetric around μ with mean μ and variance $2b^2$.
- It has heavier tails than the normal distribution, making it useful for modeling outlier-prone data.
- Considering the distributions is always symmetric around the median, if the data are are positively or negatively skewed, Laplace is inappropriate.

3.1.8 Lognormal

The `LOGNORM` function generates a random value from a lognormal distribution using inverse transform sampling. The lognormal distribution is defined such that:

$$X \sim \text{Lognormal}(\mu, \sigma^2) \quad \text{implies} \quad \log X \sim \mathcal{N}(\mu, \sigma^2)$$

Distribution Overview: The lognormal distribution is commonly used to model positive, right-skewed data. It is defined over $(0, \infty)$ and is characterized by the mean and standard deviation of the logarithm of the variable.

Function Signature:

```
=LOGNORM(mu, sigma, [usd])
```

Inputs:

- `mu`: The mean of the underlying normal distribution, $\mu = \mathbb{E}[\log X]$.
- `sigma`: The standard deviation of the underlying normal distribution.
- `usd` (optional): A user-supplied uniform deviate from $(0, 1)$. If omitted, the function generates one internally.

Output: Returns a single draw from the lognormal distribution using:

$$X = \exp\left(\mu + \sigma \cdot \Phi^{-1}(U)\right)$$

where Φ^{-1} is the inverse standard normal CDF, and $U \sim \text{Uniform}(0, 1)$.

Notes:

- If `sigma` ≤ 0, the function returns `#N/A`.

Parameter Estimation: `FITLOGNORM_MOM` The `FITLOGNORM_MOM` function estimates the μ and σ parameters of a lognormal distribution from raw data using the method of moments.

Function Signature:

```
=FITLOGNORM_MOM(dataRange)
```

Inputs:

- `dataRange`: A single-column range containing strictly positive values assumed to follow a lognormal distribution.

Output: Returns a 2-row, 2-column array:

- Row 1: Label `"mu"` and estimated value $\hat{\mu}$
- Row 2: Label `"sigma"` and estimated value $\hat{\sigma}$

Estimation Procedure: Given a dataset $\{x_1, x_2, \ldots, x_n\}$:

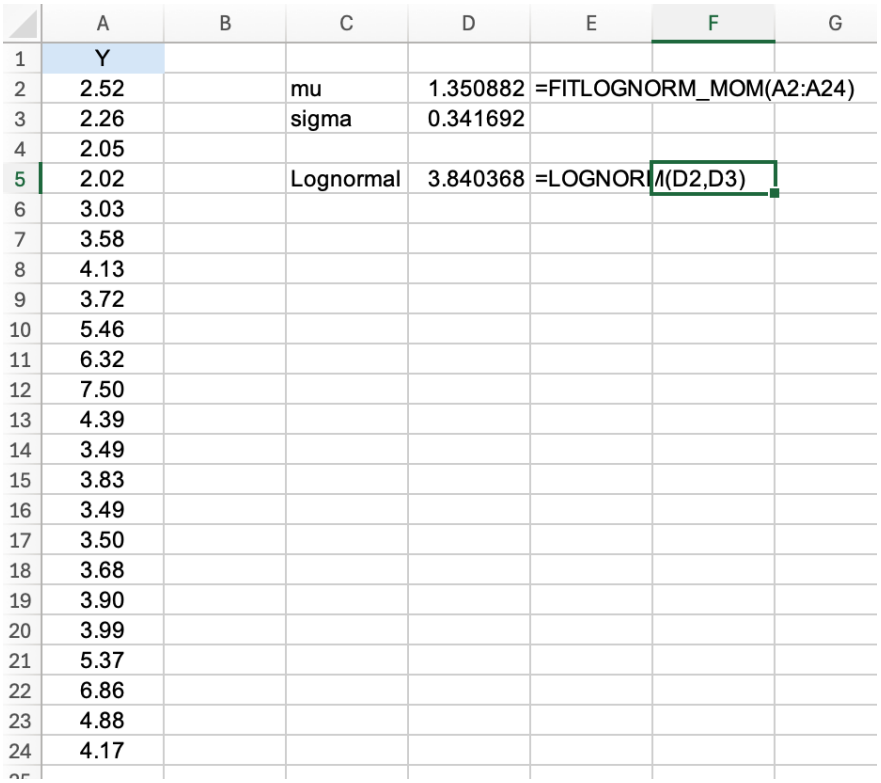
1. Compute the sample mean $\bar{x}$ and variance s^2.
2. Estimate:
$$\hat{\sigma} = \sqrt{\log\left(1 + \frac{s^2}{\bar{x}^2}\right)}, \quad \hat{\mu} = \log(\bar{x}) - \frac{1}{2}\hat{\sigma}^2$$

If the sample mean or variance is non-positive, the function returns `#NUM!`.

Example: The figure below demonstrates how `FITLOGNORM_MOM` can be used to fit a lognormal distribution to positive data (column A), and how the estimated parameters are used to generate a single random draw using the `LOGNORM` function.

```
=FITLOGNORM_MOM(A2:A24)
=LOGNORM(D3, D4)
```

- Column A contains observed lognormal-like values.
- Cells D3–D4 contain $\hat{\mu}$ and $\hat{\sigma}$.
- Cell D5 uses the estimated parameters to generate a simulated draw.



	A	B	C	D	E	F	G
1	Y						
2	2.52		mu	1.350882	=FITLOGNORM_MOM(A2:A24)		
3	2.26		sigma	0.341692			
4	2.05						
5	2.02		Lognormal	3.840368	=LOGNORM(D2,D3)		
6	3.03						
7	3.58						
8	4.13						
9	3.72						
10	5.46						
11	6.32						
12	7.50						
13	4.39						
14	3.49						
15	3.83						
16	3.49						
17	3.50						
18	3.68						
19	3.90						
20	3.99						
21	5.37						
22	6.86						
23	4.88						
24	4.17						

Figure 27: *Example: Fitting and Sampling from a Lognormal Distribution using* `FITLOGNORM_MOM` *and* `LOGNORM`

3.1.9 Normal

The `NORM` function generates random values from a Normal distribution using inverse transform sampling:

$$X \sim \mathcal{N}(\mu, \sigma^2)$$

Function Signature:

```
=NORM([mean], [stDev], [usd])
```

Inputs:

- `mean` (optional): The mean of the normal distribution. Default = 0.
- `stDev` (optional): The standard deviation of the distribution. Must be non-negative. Default = 1.
- `usd` (optional): A user-supplied value from the uniform distribution on $(0, 1)$. If omitted, the function generates one internally.

Output: Returns a normally distributed value calculated as:

$$X = \mu + \sigma \cdot \Phi^{-1}(U)$$

Where:

- Φ^{-1} is the inverse cumulative distribution function of the standard normal distribution,
- $U \sim \text{Uniform}(0, 1)$ is the input (or internally generated) uniform draw.

Example: The figure below illustrates one iteration of sampling from a normal distribution with a mean of 10 and a standard deviation of 3. In this example, the uniform standard deviate (`usd`) is omitted, and the function generates it internally.

	A	B	C
1	Average	10	
2	Standard Deviation	3	
3			
4		6.375079	=norm(B1, B2)

Figure 28: *Simulated Normal Distribution Output (`mean` = 10, `stDev` = 3)*

Notes:

- If `stDev` < 0, the function returns `#N/A` and displays an error message.
- If `stDev` $= 0$, the output is always equal to the mean.

3.1.10 Triangular

The `TRIANGULAR` function generates a random value from a Triangular distribution defined by user-supplied minimum, mode, and maximum values. It is commonly used in simulation when limited information is available but reasonable estimates for bounds and most likely outcome can be provided.

Function Signature:

```
=TRIANGULAR(min, mode, max, [usd])
```

Inputs:

- `min`: Minimum possible value of the distribution.
- `mode`: Most likely (peak) value where the distribution is highest.
- `max`: Maximum possible value of the distribution.
- `usd` (optional): A user-supplied uniform deviate on $(0, 1)$. If omitted, one is generated internally.

Output: Returns a simulated value from the Triangular distribution using inverse transform sampling:

$$X = \begin{cases} \min + \sqrt{U \cdot (\max - \min) \cdot (\text{mode} - \min)}, & \text{if } U < C \\ \max - \sqrt{(1 - U) \cdot (\max - \min) \cdot (\max - \text{mode})}, & \text{otherwise} \end{cases}$$

where:

- $U \sim \text{Uniform}(0, 1)$,
- $C = \frac{\text{mode} - \min}{\max - \min}$ is the cumulative probability at the mode.

Notes:

- Unlike GRKS (see section 3.1.4), the Triangular distribution never reaches the exact minimum or maximum values during simulation.

Example: The example below shows one simulated iteration from a Triangular distribution with `min = 5`, `mode = 10`, and `max = 17`. The `usd` argument is omitted and generated internally.

	A	B	C	D	E
1	Minimum	5			
2	Mode	10			
3	Maximum	17			
4					
5		9.9719	=TRIANGULAR(B1,B2,B3)		

Figure 29: *Triangular distribution simulation with parameters (min = 5, mode = 10, max = 17)*

3.1.11 Uniform

The `UNIFORM` function generates random values from a continuous Uniform distribution using inverse transform sampling. It linearly maps a uniform deviate from $(0, 1)$ to the specified interval $[a, b]$.

Function Signature:

```
=UNIFORM([min], [max], [usd])
```

Inputs:

- `min` (optional): The lower bound of the distribution. Default = 0.
- `max` (optional): The upper bound of the distribution. Default = 1.
- `usd` (optional): A user-supplied uniform draw from $(0, 1)$. If omitted, one is internally generated.

If `max` is less than `min`, the function automatically swaps the two values to ensure a valid range.

Output: Returns a uniformly distributed value computed as:

$$X = a + (b - a) \cdot U$$

Where:

- a is the minimum,
- b is the maximum,
- $U \sim \text{Uniform}(0, 1)$ is the uniform draw.

Example: The figure below shows a single iteration of simulating a value from a uniform distribution with parameters `min` $= 10$ and `max` $= 20$. In this example, the uniform standard deviate (`usd`) is omitted and generated internally by the function.

	A	B	C	D
1	Minimum	10		
2	Maximum	20		
3				
4		17.05548	=UNIFORM(B1,B2)	

Figure 30: *Simulated Uniform Distribution Output (`minimum` = 10, `maximum` = 20)*

3.2 Non-parametric Distributions

3.2.1 Empirical

The `EMPIRICAL` function generates a simulated value from a non-parametric empirical distribution using inverse transform sampling.

Overview: Non-parametric empirical distributions are typically used when the available sample is too small or irregular to reliably estimate the parameters of a standard parametric distribution (Richardson, 2008). In this approach, the distribution's shape is entirely determined by the observed data, and its support is bounded by the minimum and maximum observed values.

Although the input data are discrete, the `EMPIRICAL` function applies linear interpolation between sorted data points, effectively producing a continuous cumulative distribution function (CDF). This method maintains the shape of the historical data while allowing for smooth interpolation.

The empirical CDF construction and interpolation procedure follow the approach described in Richardson (2008, Chapter 5, p. 15).

Function Signature:

```
=EMPIRICAL(dataRange, [usd])
```

Inputs:

- `dataRange`: A one-column range containing the sample data.
- `usd` (optional): A user-supplied uniform standard deviate on $(0, 1)$. If omitted, the function draws one internally.

Output: Returns a random draw interpolated from the empirical CDF. The function operates as follows:

1. Sorts the input data in ascending order.

2. Constructs pseudo-boundary values to extend the range:

$$X_{\text{pseudo-min}} = 0.9999 \cdot \min(X), \quad X_{\text{pseudo-max}} = 1.0001 \cdot \max(X)$$

This ensures that values slightly below or above the observed extremes can still be bracketed during interpolation.

3. Assigns cumulative probabilities to each value:

$$F(x_i) = \frac{i - 0.5}{n}, \quad i = 1, \ldots, n$$

with 0 and 1 assigned to the pseudo-min and pseudo-max respectively.

4. For a uniform deviate $u \in (0, 1)$, locate the bounding values F_L and F_U and linearly interpolate:

$$X = X_L + (X_U - X_L) \cdot \frac{u - F_L}{F_U - F_L}$$

Example: The example below illustrates how the `EMPIRICAL` function works using a historical sample that has only 11 observations. The uniform deviate is omitted and generated internally.

	A	B	C	D
1	Y			
2	-11.55	12.42337	=EMPIRICAL(A2:A12)	
3	-0.48			
4	15.88			
5	1.69			
6	3.37			
7	0.88			
8	1.69			
9	11.00			
10	-2.59			
11	-10.17			
12	-35.66			

Figure 31: *Example of sampling from an empirical distribution using `EMPIRICAL`.*

Notes:

- The interpolation procedure ensures a smooth CDF while remaining strictly within the bounds of the observed data.
- This method is especially useful in risk modeling or simulation when the dataset is small, skewed, or contains outliers not well captured by parametric distributions.

References

- Richardson, J.W. (2008). *Simulation for Applied Risk Management with an Introduction to SIMETAR*. Department of Agricultural Economics. Texas A&M University.

3.2.2 Kernel Density Estimation (KDE)

The `KDE` function generates a random value from a kernel density estimate (KDE) constructed from the data series. It applies non-parametric smoothing using a selected kernel function and bandwidth. The result is a continuous approximation of the empirical distribution based on inverse transform sampling from the KDE-implied CDF.

Function Signature:

```
=KDE(dataRange, [customBandwidth], [kernelType], [kernelScale], [usd], [numGrid])
```

Inputs:

- `dataRange` (required): Range containing numeric data series.
- `customBandwidth` (optional): Custom bandwidth value. If omitted, Silverman's rule of thumb is used.
- `kernelType` (optional): Specifies the smoothing kernel to use. One of:
  - `"Gaussian"` (default): $K(u) = \frac{1}{\sqrt{2\pi}} e^{-u^2/2}$
  - `"Triangular"`:
$$K(u) = \begin{cases} 1 - |u|, & \text{if } |u| \leq 1 \\ 0, & \text{otherwise} \end{cases}$$
  - `"Epanechnikov"`:
$$K(u) = \begin{cases} \frac{3}{4}(1 - u^2), \text{if } |u| \leq 1 \\ 0, & \text{otherwise} \end{cases}$$
- `kernelScale` (optional): Scalar applied to the calculated bandwidth (default = 0.9).
- `usd` (optional): Uniform value in $(0, 1)$ for inverse transform sampling. If omitted, one is generated internally.
- `numGrid` (optional): Number of grid points used to approximate the density and CDF (default = 200).

Output: Returns a single simulated value based on the KDE-smoothed distribution.

Method and Sampling Overview:

1. The bandwidth h is either provided by the user or calculated automatically using:

$$h = \texttt{kernelScale} \cdot \min(\sigma, \text{IQR}/1.34) \cdot n^{-1/5}$$

where `kernelScale` is a tuning parameter, σ is the standard deviation, IQR is the interquartile range, and n is the number of data points.

2. A smoothed probability density function (PDF) is estimated on a regular grid of points $\{x_1, x_2, \ldots, x_G\}$, where G is the number of grid points (default: 200), using:

$$\hat{f}(x_i) = \frac{1}{nh} \sum_{j=1}^{n} K\left(\frac{x_i - x_j}{h}\right)$$

where $K(\cdot)$ is the selected kernel function (e.g., Gaussian, Triangular, or Epanechnikov).

3. The cumulative distribution function (CDF) is calculated from the PDF using the trapezoidal rule:

$$F(x_i) = \sum_{j=2}^{i} f(x_j) \cdot (x_j - x_{j-1})$$

4. To generate a random draw, the function uses a uniform value $u \sim \text{Uniform}(0, 1)$, finds the interval where it fits:

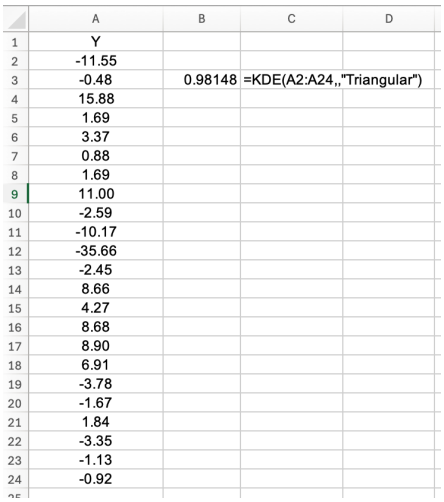
$$F(x_{i-1}) \leq u \leq F(x_i)$$

and returns a simulated value by interpolating between the two grid points:

$$x^* = x_{i-1} + \left(\frac{u - F(x_{i-1})}{F(x_i) - F(x_{i-1})}\right) \cdot (x_i - x_{i-1})$$

Notes:

- This KDE implementation is non-parametric and flexible.
- Results may vary depending on the chosen kernel type and are especially sensitive to the selected bandwidth. A smaller bandwidth can lead to overfitting (too spiky), while a larger bandwidth can oversmooth the distribution.
- The `KDE` method is especially useful when no known parametric distribution fits the data well.



	A	B	C	D
1	Y			
2	-11.55			
3	-0.48	0.98148	=KDE(A2:A24,,"Triangular")	
4	15.88			
5	1.69			
6	3.37			
7	0.88			
8	1.69			
9	11.00			
10	-2.59			
11	-10.17			
12	-35.66			
13	-2.45			
14	8.66			
15	4.27			
16	8.68			
17	8.90			
18	6.91			
19	-3.78			
20	-1.67			
21	1.84			
22	-3.35			
23	-1.13			
24	-0.92			

Figure 32: *Example of simulated draw from a kernel density estimate using the* `KDE` *function with triangular kernel*

4 Time Series

4.1 Dickey-Fuller Unit Root Test

Function: `DFTEST`

Overview

The `DFTEST` function implements the **Dickey-Fuller test** to evaluate whether a given univariate time series contains a unit root, indicating non-stationarity. The test is based on an ordinary least squares (OLS) regression of the first difference of the series on its lagged level (and optionally a time trend), and tests whether the lagged level coefficient is statistically different from zero.

This test is useful in time series analysis to diagnose the need for differencing prior to modeling with AR or similar models. The null hypothesis is that the time series has a unit root (i.e., is non-stationary), while the alternative is that it is trend-stationary or mean-reverting.

Regression Equation

The Dickey–Fuller regression form is:

$$\Delta y_t = \alpha + \beta y_{t-1} + \gamma t + \varepsilon_t$$

where:

- $\Delta y_t = y_t - y_{t-1}$ is the first difference
- α is the intercept
- β is the coefficient on the lagged level y_{t-1}
- γ is the coefficient on the deterministic time trend (optional)

The null hypothesis is:

$$H_0 : \beta = 0 \quad \text{(unit root exists)}$$

against the alternative:

$$H_1 : \beta < 0 \quad \text{(stationary process)}$$

Estimation Method

1. If `diffOrder` > 0, the series is differenced prior to testing.
2. OLS is used to regress Δy_t on y_{t-1} and an intercept.
3. Optionally, a deterministic time trend t is included.
4. The t-statistic for the coefficient β is returned.

Function Signature

```
=DFTEST(dataRange, [diffOrder], [timeTrend])
```

Arguments:

- `dataRange` (required). A univariate numeric time series.
- `diffOrder` (optional). Number of differences to apply before estimation (default = 0).
- `timeTrend` (optional). If `True`, includes a deterministic trend term in the regression (default = False).

Returns

- A single value: the t-statistic associated with the coefficient on y_{t-1}.

Example The example below demonstrates the estimation of Dickey-Fuller test statistics with 1 difference and no deterministic time trend.

	A	B	C	D
1	Year	Price ($/bu.)		
2	1	1.81		
3	2	1.48		
4	3	1.48		
5	4	1.63		
6	5	1.87		
7	6	2.63		
8	7	3.15		
9	8	2.02		
10	9	2.52		
11	10	3.49		
12	11	3.89		
13	12	3.75		
14	13	3.21		
15	14	2.12		
16	15	2.06		
17	16	2.59		
18	17	2.66		
19	18	2.82		
20	19	2.77		
21	20	4.55		
22	21	4.57		
23	22	3.69		
24	23	3.38		
25				
26	Dickey-Fuller test	-3.8945219	=DFTEST(B2:B24,1,FALSE)	

Figure 33: *Example of Dickey-Fuller test with 1 difference and no time trend*

Assumptions and Notes

- Critical values are non-standard and depend on the inclusion of an intercept and trend.
- The result should be compared to Dickey–Fuller critical values from literature for inference.
- The test assumes homoskedastic errors and no serial correlation beyond the first lag.

References

- Dickey, D.A. & Fuller, W.A. (1979). "Distribution of the Estimators for Autoregressive Time Series with a Unit Root." *Journal of the American Statistical Association.*
- Hamilton, J.D. (1994). *Time Series Analysis.* Princeton University Press.
- Enders, W. (2014). *Applied Econometric Time Series.* 4th ed., Wiley.

4.2 Autocorrelation Function (ACF)

Function: `AUTOCORR`

Overview

The `AUTOCORR` function calculates the sample autocorrelation coefficient at a specified lag k for a (possibly differenced) univariate time series. It includes the Bartlett-adjusted standard error and associated t-statistic.

Definition

Let $X = \{x_1, x_2, \ldots, x_n\}$ be the input series after d differences. The autocorrelation at lag k is:

$$r_k = \frac{\sum_{t=1}^{n-k}(x_t - \bar{x})(x_{t+k} - \bar{x})}{\sum_{t=1}^{n}(x_t - \bar{x})^2}$$

The standard error is estimated using the Bartlett formula:

$$\mathrm{SE}(r_k) \approx \sqrt{\frac{1 + 2\sum_{j=1}^{k-1} r_j^2}{n}}$$

Function Signature

```
=AUTOCORR(dataRange, lag, numDiff)
```

Arguments:

- `dataRange` (required). A univariate numeric range.
- `lag` (required). Positive integer indicating the lag.
- `numDiff` (required). Number of differences to apply.

Returns An array with:

- Sample autocorrelation r_k
- Bartlett-corrected standard error
- t-statistic (r_k/SE)

Example The example below displays the sample autocorrelation function, implemented with 3 lags and 1 difference. It outputs the sample autocorrelation coefficient, Bartlett-corrected standard error, and the corresponding t-statistics value.

	A	B	C	D	E
1	Year	Price ($/bu.)			
2	1	1.81			
3	2	1.48			
4	3	1.48			
5	4	1.63			
6	5	1.87			
7	6	2.63			
8	7	3.15			
9	8	2.02			
10	9	2.52			
11	10	3.49			
12	11	3.89			
13	12	3.75			
14	13	3.21			
15	14	2.12			
16	15	2.06			
17	16	2.59			
18	17	2.66			
19	18	2.82			
20	19	2.77			
21	20	4.55			
22	21	4.57			
23	22	3.69			
24	23	3.38			
25					
26	Lag	Difference	Sample autocorrelation	Standard-error	t-statistics
27	3	1	-0.198133142	0.243121591	-0.814954941
28			=AUTOCORR(B2:B24, A27, B27)		

Figure 34: *Example of sample autocorrelation test with 3 lags and 1 difference*

Note: For Excel 2021 and later versions, including Microsoft 365, the function returns a spilled array automatically. For earlier versions, use `Ctrl + Shift + Enter` *(Windows) or* `Cmd + Shift + Return` *(Mac) to enter the formula as an array.*

References

- Box, G.E.P., Jenkins, G.M., Reinsel, G.C., & Ljung, G.M. (2016). *Time Series Analysis: Forecasting and Control*. Wiley.

4.3 Partial Autocorrelation Function (PACF)

Function: `PAUTOCORR`

Overview

`PAUTOCORR` estimates the partial autocorrelation coefficient at lag k via OLS regression, controlling for lags 1 through $k-1$. The function also returns the standard error and t-statistic for significance testing.

Definition

The partial autocorrelation at lag k is the coefficient ϕ_{kk} from the regression:

$$x_t = \beta_0 + \sum_{j=1}^{k} \beta_j x_{t-j} + \varepsilon_t$$

with $\phi_{kk} = \beta_k$.

Function Signature

```
=PAUTOCORR(dataRange, lag, [numDiff])
```

Arguments:

- `dataRange` (required). A univariate numeric range.
- `lag` (required). Positive integer indicating the lag.
- `numDiff` (optional). Number of differences to apply (default = 0).

Returns

- Partial autocorrelation ϕ_{kk}
- Standard error
- t-statistic

Example The example below displays the partial autocorrelation function, implemented with 2 lags and 1 difference. It outputs the partial autocorrelation coefficient, standard error, and the corresponding t-statistics value.

	A	B	C	D	E
1	Year	Price ($/bu.)			
2	1	1.81			
3	2	1.48			
4	3	1.48			
5	4	1.63			
6	5	1.87			
7	6	2.63			
8	7	3.15			
9	8	2.02			
10	9	2.52			
11	10	3.49			
12	11	3.89			
13	12	3.75			
14	13	3.21			
15	14	2.12			
16	15	2.06			
17	16	2.59			
18	17	2.66			
19	18	2.82			
20	19	2.77			
21	20	4.55			
22	21	4.57			
23	22	3.69			
24	23	3.38			
25					
26	Lag	Difference	Partial autocorrelation	Standard-error	t-statistics
27	2	1	-0.436679974	0.232329637	-1.879570679
28			=PAUTOCORR(B2:B24, A27, B27)		

Figure 35: *Example of partial autocorrelation test with 2 lags and 1 difference*

Note: For Excel 2021 and later versions, including Microsoft 365, the function returns a spilled array automatically. For earlier versions, use `Ctrl + Shift + Enter` *(Windows) or* `Cmd + Shift + Return` *(Mac) to enter the formula as an array.*

References

- Box, G.E.P., Jenkins, G.M., Reinsel, G.C., & Ljung, G.M. (2016). *Time Series Analysis: Forecasting and Control*. Wiley.
- Hyndman, R.J., & Athanasopoulos, G. (2018). *Forecasting: Principles and Practice*.

4.4 Partial Autocorrelation via Yule-Walker

Function: `PAUTOCORR_YW`

Overview

The `PAUTOCORR_YW` function estimates the partial autocorrelation coefficient (PACF) at a given lag using the **Yule-Walker equations** and the **Durbin–Levinson recursion**. Unlike regression-based PACF estimators, which perform separate regressions for each lag, this method recursively calculates PACF values using only the sample autocorrelations. It is computationally more efficient, especially for high-order AR models (Box et al. (2016)).

Definition

Given the sample autocorrelations $\{r_1, \ldots, r_k\}$ from a (demeaned) time series $\{x_t\}$, the Yule-Walker equations form a Toeplitz system of the form:

$$\begin{bmatrix} 1 & r_1 & \ldots & r_{k-1} \\ r_1 & 1 & \ldots & r_{k-2} \\ \vdots & \vdots & \ddots & \vdots \\ r_{k-1} & r_{k-2} & \ldots & 1 \end{bmatrix} \begin{bmatrix} \phi_{k1} \\ \phi_{k2} \\ \vdots \\ \phi_{kk} \end{bmatrix} = \begin{bmatrix} r_1 \\ r_2 \\ \vdots \\ r_k \end{bmatrix}$$

The solution vector $\boldsymbol{\phi}_k = (\phi_{k1}, \ldots, \phi_{kk})^\top$ gives the autoregressive coefficients of an AR(k) process, where the final element ϕ_{kk} is the PACF at lag k.

Estimation Method

PACF values are calculated using the **Durbin–Levinson recursion**, a numerically stable recursive algorithm that solves the Yule-Walker equations efficiently without matrix inversion. It also provides backward-updated PACF values ϕ_{kj} for $j < k$.

Standard Error and t-Statistic

For large samples, the standard error of ϕ_{kk} is approximated by:

$$\mathrm{SE}(\phi_{kk}) \approx \frac{1}{\sqrt{n}}$$

The function also reports a t-statistic:

$$t = \frac{\phi_{kk}}{\mathrm{SE}(\phi_{kk})}$$

for hypothesis testing of $\phi_{kk} = 0$.

Function Signature

```
=PAUTOCORR_YW(dataRange, lag, [numDiff], [sampleAdjusted])
```

Arguments

- `dataRange`: Excel range of numeric time series values.
- `lag`: Desired lag k for which the PACF is computed.
- `numDiff` (optional): Number of differences to apply (default = 0).
- `sampleAdjusted` (optional): Use $(n-k)$ denominator in autocovariance if `True` (default = True).

Returns

- PACF at lag k (ϕ_{kk})
- Standard error
- t-statistic

Example The example below displays the partial autocorrelation function using the Yule-Walker equations, implemented with 2 lags and 1 difference. It outputs the partial autocorrelation coefficient, standard error, and the corresponding t-statistics value.

	A	B	C	D	E
1	Year	Price ($/bu.)			
2	1	1.81			
3	2	1.48			
4	3	1.48			
5	4	1.63			
6	5	1.87			
7	6	2.63			
8	7	3.15			
9	8	2.02			
10	9	2.52			
11	10	3.49			
12	11	3.89			
13	12	3.75			
14	13	3.21			
15	14	2.12			
16	15	2.06			
17	16	2.59			
18	17	2.66			
19	18	2.82			
20	19	2.77			
21	20	4.55			
22	21	4.57			
23	22	3.69			
24	23	3.38			
25					
26	Lag	Difference	Partial autocorrelation	Standard-error	t-statistics
27	2	1	-0.427762946	0.213200716	-2.006386063
28			=PAUTOCORR_YW(B2:B24, A27, B27)		

Figure 36: *Example of partial autocorrelation test using Yule-Walker equations with 2 lags and 1 difference*

Note: For Excel 2021 and later versions, including Microsoft 365, the function returns a spilled array automatically. For earlier versions, use `Ctrl + Shift + Enter` *(Windows) or* `Cmd + Shift + Return` *(Mac) to enter the formula as an array.*

References

- Box, G.E.P., Jenkins, G.M., Reinsel, G.C., & Ljung, G.M. (2016). *Time Series Analysis: Forecasting and Control*. Wiley.
- Brockwell, P. J., & Davis, R. A. (2016). *Introduction to Time Series and Forecasting*. Springer.
- Durbin, J., & Levinson, N. (1951). "Orthogonal Series and Prediction." *Journal of the Royal Statistical Society, Series B*.

4.5 ACF Series Table

Function: `ACF_SERIES`

Overview

`ACF_SERIES` generates a table of autocorrelation coefficients, standard errors, and t-statistics up to a user-specified maximum lag.

Function Signature

```
Function ACF_SERIES(dataRange, maxLag, [numDiff])
```

Arguments

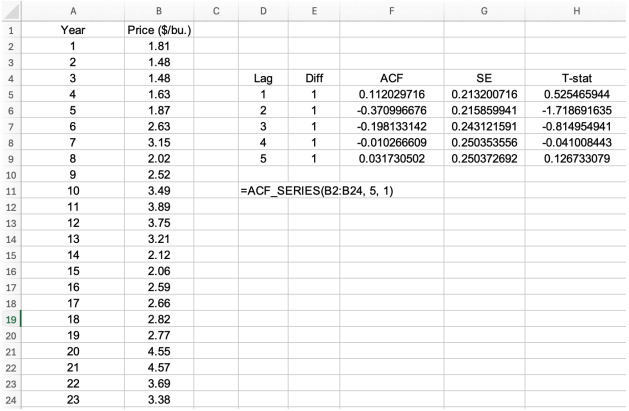
- `dataRange`: Excel range of numeric time series values.
- `maxLag`: Desired maximum number of lag k for which the `AUTOCORR` is calculated (see 4.2).
- `numDiff` (optional): Number of differences to apply (default = 0).

Returns A 2D array (with headers) of:

- Lag number
- Number of differences applied
- Autocorrelation
- Standard error

- t-statistic

Example The example below displays the sample autocorrelation series function, estimated with up to 5 lags and one difference. It outputs the lag numbers (1-5), the number of differences (1), and the corresponding sample autocorrelation coefficients, standard errors, and t-statistic values.



	A	B	C	D	E	F	G	H
1	Year	Price ($/bu.)						
2	1	1.81						
3	2	1.48						
4	3	1.48		Lag	Diff	ACF	SE	T-stat
5	4	1.63		1	1	0.112029716	0.213200716	0.525465944
6	5	1.87		2	1	-0.370996676	0.215859941	-1.718691635
7	6	2.63		3	1	-0.198133142	0.243121591	-0.814954941
8	7	3.15		4	1	-0.010266609	0.250353556	-0.041008443
9	8	2.02		5	1	0.031730502	0.250372692	0.126733079
10	9	2.52						
11	10	3.49		=ACF_SERIES(B2:B24, 5, 1)				
12	11	3.89						
13	12	3.75						
14	13	3.21						
15	14	2.12						
16	15	2.06						
17	16	2.59						
18	17	2.66						
19	18	2.82						
20	19	2.77						
21	20	4.55						
22	21	4.57						
23	22	3.69						
24	23	3.38						

Figure 37: *Example of sample autocorrelation series tests with up to 5 lags and 1 difference*

Note: For Excel 2021 and later versions, including Microsoft 365, the function returns a spilled array automatically. For earlier versions, use `Ctrl + Shift + Enter` *(Windows) or* `Cmd + Shift + Return` *(Mac) to enter the formula as an array.*

4.6 PACF Series Table

Function: `PACF_SERIES`

Overview

`PACF_SERIES` outputs a table of PACF values up to a user-defined maximum lag, using regression-based estimation.

Function Signature

```
=PACF_SERIES(dataRange, maxLag, [numDiff])
```

Arguments

- `dataRange`: Excel range of numeric time series values.
- `maxLag`: Desired maximum number of lag k for which the `PAUTOCORR` is calculated (see 4.3).
- `numDiff` (optional): Number of differences to apply (default = 0).

Returns Table with:

- Lag number
- Differencing order
- PACF coefficient
- Standard error
- t-statistic

Example The example below displays the partial autocorrelation series function, estimated with up to 5 lags and one difference. It outputs the lag numbers (1-5), the number of differences (1), and the corresponding partial autocorrelation coefficients, standard errors, and t-statistic values.

	A	B	C	D	E	F	G	H
1	Year	Price ($/bu.)						
2	1	1.81						
3	2	1.48						
4	3	1.48		Lag	Diff	PACF	SE	T-stat
5	4	1.63		1	1	0.113083687	0.227734324	0.496559696
6	5	1.87		2	1	-0.436679974	0.232329637	-1.879570679
7	6	2.63		3	1	-0.171160141	0.26850852	-0.637447709
8	7	3.15		4	1	-0.138329264	0.338893047	-0.408179704
9	8	2.02		5	1	-0.106367127	0.367641604	-0.289322879
10	9	2.52						
11	10	3.49		=PACF_SERIES(B2:B24, 5, 1)				
12	11	3.89						
13	12	3.75						
14	13	3.21						
15	14	2.12						
16	15	2.06						
17	16	2.59						
18	17	2.66						
19	18	2.82						
20	19	2.77						
21	20	4.55						
22	21	4.57						
23	22	3.69						
24	23	3.38						

Figure 38: *Example of partial autocorrelation series tests with up to 5 lags and 1 difference*

Note: For Excel 2021 and later versions, including Microsoft 365, the function returns a spilled array automatically. For earlier versions, use `Ctrl + Shift + Enter` *(Windows) or* `Cmd + Shift + Return` *(Mac) to enter the formula as an array.*

4.7 Partial Autocorrelation Series via Yule-Walker

Function: `PACF_SERIES_YW`

Overview

The `PACF_SERIES_YW` function estimates the partial autocorrelation function (PACF) for a sequence of lags using the **Yule-Walker method** described in 4.4.

Function Signature

```
=PACF_SERIES_YW(dataRange, maxLag, [numDiff], [sampleAdjusted])
```

Arguments

- `dataRange`] (required). A univariate numeric time series.
- `maxLag` (required). Maximum lag up to which `PAUTOCORR_YW` values are calculated (see 4.4).
- `numDiff` (optional). Number of differences applied before estimation (default = 0).
- `sampleAdjusted` (optional). If `True`, use $(n - k)$ as denominator when estimating autocovariances (default = True).

Returns A table with:

- Lag number
- Differencing order
- PACF at lag k (via Yule-Walker)
- Standard error $\approx 1/\sqrt{n}$
- t-statistic

Example The example below displays the partial autocorrelation series function via Yule-Walker equations, estimated with up to 5 lags and one difference. It outputs the lag numbers (1-5), the number of differences (1), and the corresponding partial autocorrelation coefficients, standard errors, and t-statistic values.

	A	B	C	D	E	F	G	H
1	Year	Price ($/bu.)						
2	1	1.81						
3	2	1.48						
4	3	1.48		Lag	Diff	PACF	SE	T-stat
5	4	1.63		1	1	0.117364464	0.213200716	0.550488132
6	5	1.87		2	1	-0.427762946	0.213200716	-2.006386063
7	6	2.63		3	1	-0.137545091	0.213200716	-0.645143664
8	7	3.15		4	1	-0.172826258	0.213200716	-0.810627005
9	8	2.02		5	1	-0.123802562	0.213200716	-0.580685489
10	9	2.52						
11	10	3.49		=PACF_SERIES_YW(B2:B24, 5, 1)				
12	11	3.89						
13	12	3.75						
14	13	3.21						
15	14	2.12						
16	15	2.06						
17	16	2.59						
18	17	2.66						
19	18	2.82						
20	19	2.77						
21	20	4.55						
22	21	4.57						
23	22	3.69						
24	23	3.38						

Figure 39: *Example of partial autocorrelation series tests via Yule-Walker equations with up to 5 lags and 1 difference*

Note: For Excel 2021 and later versions, including Microsoft 365, the function returns a spilled array automatically. For earlier versions, use `Ctrl + Shift + Enter` *(Windows) or* `Cmd + Shift + Return` *(Mac) to enter the formula as an array.*

5 Modeling Dependence

5.1 Correlated Standard Normal Deviates

Function: `CSND`

Overview

The `CSND` function generates a vector of correlated standard normal deviates using a specified correlation matrix.

Function Signature

```
=CSND(correlMatrix)
```

Inputs

- `correlMatrix`: A $k \times k$ symmetric and positive semi-definite matrix representing the desired correlation structure.

Method Description

1. Perform a Cholesky decomposition:

$$\mathbf{L}\mathbf{L}^\top = \mathbf{R}$$

   where $\mathbf{R}$ is the correlation matrix and $\mathbf{L}$ is lower triangular.

2. Generate a vector $\mathbf{z} \sim \mathcal{N}(0, \mathbf{I}_k)$ of k independent standard normal values.

3. Calculate the correlated standard normals:

$$\mathbf{x} = \mathbf{L}\mathbf{z}$$

   The output $\mathbf{x}$ will have the specified correlation structure $\mathbf{R}$.

Output

Returns a $1 \times k$ array of correlated standard normal deviates.

Notes

- The Cholesky decomposition assumes that the correlation matrix is symmetric and positive semi-definite. If not, the `CHOLESKY` function applies shrinkage under the hood to proceed.

Example The example below shows the `CSND` function applied to a 3x3 correlation matrix.

	A	B	C	D
1	Correlation matrix			
2		X	Y	Z
3	Z	1	-0.634615	-0.390651
4	Y	-0.634615	1	0.576488
5	Z	-0.390651	0.576488	1
6				
7				
8	Correlated standard normal deviates			
9		-0.8906	0.4023	1.2709
10		=csnd(B3:D5)		

Figure 40: *Example: Correlated Standard Normal Deviates*

5.2 Correlated Uniform Standard Deviates

Function: `CUSD`

Overview

The `CUSD` function transforms the correlated standard normal deviates from `CSND` into correlated uniform standard deviates using the standard normal cumulative distribution function (CDF).

Function Signature

```
=CUSD(correlMatrix)
```

Inputs

- `correlMatrix`: A $k \times k$ correlation matrix.

Method Description

1. Call `CSND` to obtain a $k \times 1$ vector of correlated standard normal deviates $\mathbf{x}$.

2. Apply the standard normal cumulative distribution function (CDF) to each component:

$$u_i = \Phi(x_i)$$

where $\Phi(\cdot)$ is the CDF of the standard normal distribution.

Output

Returns a $1 \times k$ array of correlated uniform standard deviates (in the range $(0, 1)$).

Use Case

The correlated uniform standard deviates are useful in a wide range of applications, including Gaussian copula-based simulations with mixed marginals (e.g., lognormal, beta, empirical), while preserving historical inter-variable dependencies.

Example The example below shows the `CUSD` function applied to a 3x3 correlation matrix.

	A	B	C	D
1	Correlation matrix			
2		X	Y	Z
3	Z	1	-0.634615	-0.390651
4	Y	-0.634615	1	0.576488
5	Z	-0.390651	0.576488	1
6				
7				
8	Correlated uniform standard deviates			
9		0.1101	0.6216	0.3492
10		=cusd(B3:D5)		

Figure 41: *Example: Correlated Uniform Standard Deviates*

5.3 Multivariate Normal

Function: `MVNORM`

Overview

The `MVNORM` function generates a multivariate draw from a *multivariate normal distribution* with a user-specified covariance matrix and optional mean vector. It simulates a vector of correlated normal random variables that preserve the variance and covariance structure defined by the input covariance matrix, centered at the specified means.

Function Signature

```
=MVNORM(covarMatrix, [meanVectors])
```

Inputs

- `covarMatrix`: A symmetric $k \times k$ matrix representing the desired covariance structure among k variables.
- `meanVectors` (optional): A $1 \times k$ row vector or $k \times 1$ column vector of means $\boldsymbol{\mu}$. If not provided, the function uses zero means for all variables (i.e., $\boldsymbol{\mu} = \mathbf{0}$).

Method Description

The function follows a standard procedure for simulating from a multivariate normal distribution:

1. Let $\boldsymbol{\Sigma}$ denote the user-supplied $k \times k$ covariance matrix and $\boldsymbol{\mu}$ denote the mean vector.
2. Generate the lower triangular matrix $\mathbf{L}$ from Cholesky decomposition:

$$\boldsymbol{\Sigma} = \mathbf{L}\mathbf{L}^\top$$

3. Generate a vector of k independent standard normal deviates:

$$\mathbf{z} \sim \mathcal{N}(0, \mathbf{I}_k)$$

4. Multiply to induce correlation and add the mean vector:

$$\mathbf{x} = \mathbf{L}\mathbf{z} + \boldsymbol{\mu}$$

5. The result $\mathbf{x}$ is a vector of correlated normal values with covariance $\boldsymbol{\Sigma}$ and mean $\boldsymbol{\mu}$.

Example

The example below shows how the `MVNORM` function uses a 3×3 covariance matrix to generate a single draw from a trivariate normal distribution. When the mean vector is not provided

(or is zero), the multiplication step involves:

$$\mathbf{x} = \begin{bmatrix} l_{11} & 0 & 0 \\ l_{21} & l_{22} & 0 \\ l_{31} & l_{32} & l_{33} \end{bmatrix} \begin{bmatrix} z_1 \\ z_2 \\ z_3 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}$$

For a non-zero mean vector $\boldsymbol{\mu} = [\mu_1, \mu_2, \mu_3]^\top$, the calculation becomes:

$$\mathbf{x} = \begin{bmatrix} l_{11} & 0 & 0 \\ l_{21} & l_{22} & 0 \\ l_{31} & l_{32} & l_{33} \end{bmatrix} \begin{bmatrix} z_1 \\ z_2 \\ z_3 \end{bmatrix} + \begin{bmatrix} \mu_1 \\ \mu_2 \\ \mu_3 \end{bmatrix} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}$$

The `MVNORM` function automatically calculates the lower triangular matrix $\mathbf{L}$ from the input covariance matrix, generates a vector of independent standard normal deviates $\mathbf{z}$, and calculates the multivariate normal draw as $\mathbf{x} = \mathbf{L}\mathbf{z} + \boldsymbol{\mu}$. The result is returned as a $1 \times k$ row vector for display in Excel.

	A	B	C	D
1		X	Y	Z
2		-0.205	-0.246	-11.545
3		-0.200	-0.030	-0.482
4		-0.342	-0.264	15.878
5		-0.381	-0.295	1.687
6		-0.089	-0.222	3.367
7		0.220	0.189	0.878
8		0.145	0.143	1.690
9		-0.028	0.071	11.002
10		0.379	0.230	-2.587
11		0.612	0.327	-10.175
12		0.738	0.492	-35.663
13				
14				
15	**Covariance matrix**			
16		X	Y	Z
17	X	0.14198586	0.0959758	-3.52908805
18	Y	0.09597579	0.0716774	-2.19399159
19	Z	-3.5290881	-2.1939916	184.5337961
20				
21				
22	**Multivariate normal draw**			
23		X	Y	Z
24		-0.0558627	0.1111225	-2.61821373
25		=MVNORM(B17:D19)		

Figure 42: *Example: A simulated draw from `MVNORM` using a 3×3 covariance matrix with zero mean vector.*

Notes

- The input covariance matrix must be symmetric and at least positive semi-definite. If the matrix is not strictly positive definite, the `MVNORM` function uses a shrinkage-based adjustment internally to ensure that Cholesky decomposition can proceed.
- The result is returned as a $1 \times k$ row vector.
- If the mean vector is omitted, all means default to zero ($\boldsymbol{\mu} = \mathbf{0}$).

5.4 Copula-Based Simulation with KDE Marginals

Function: `CopulaKDE`

Overview

The `CopulaKDE` function generates a single multivariate simulation from a set of correlated variables using a *Gaussian copula* with KDE marginals. This method allows incorporating dependence among random variables when no parametric form is assumed for the marginals.

For details on how KDE marginals are estimated, see Section 3.2.2.

Function Signature

```
=CopulaKDE(dataRange, [customBandwidth], [kernelType], [kernelScale])
```

Inputs

- `dataRange`: A range with n rows and k columns of historical data. Each column represents one variable.
- `customBandwidth` (optional): A single value, comma-separated list, or range with k rows specifying custom KDE bandwidths.
- `kernelType` (optional): A single string, comma-separated list, or range with k rows specifying kernel types:
  - `"Gaussian"`
  - `"Triangular"`
  - `"Epanechnikov"`

- `kernelScale` (optional): A single value, comma-separated list, or range with k rows scaling the bandwidth as described in Section 3.2.2.

Simulation Procedure

1. Generate the empirical correlation matrix Σ from `dataRange`.
2. Perform Cholesky decomposition $\Sigma = LL^\top$.
3. Generate independent standard normal values $Z \sim \mathcal{N}(0, I_k)$.
4. Transform to correlated normals: $S = LZ$.
5. Convert each S_j to a correlated uniform standard deviate $cusd_j = \Phi(S_j)$ using the standard normal CDF.
6. For each column j, use `KDE` to simulate from the marginal distribution:

$$x_j = \texttt{KDE}(X_j, h_j, K_j, \alpha_j, cusd_j)$$

Output

Returns a $1 \times k$ array representing a single multivariate draw that:

- Matches the marginal distribution of each variable via KDE
- Preserves the historical dependence structure via Gaussian copula

Notes

- If no kernel or bandwidth is specified, defaults are: Gaussian kernel and Silverman's rule with scale 0.9.
- Inputs such as `kernelType`, `customBandwidth`, and `kernelScale` can vary by column via comma-separated strings or horizontal ranges.
- This method preserves rank-based dependence and supports asymmetric, skewed, or multimodal marginals.

Example

The example below shows a 3-variable simulation using `CopulaKDE`, where each variable uses a different kernel type (Gaussian, Triangular, and Epanechnikov).

	A	B	C	D
1		X	Y	Z
2		-0.205	-0.246	-11.545
3		-0.200	-0.030	-0.482
4		-0.342	-0.264	15.878
5		-0.381	-0.295	1.687
6		-0.089	-0.222	3.367
7		0.220	0.189	0.878
8		0.145	0.143	1.690
9		-0.028	0.071	11.002
10		0.379	0.230	-2.587
11		0.612	0.327	-10.175
12		0.738	0.492	-35.663
13				
14	Kernel type	Gaussian	Triangular	Epanechnikov
15				
16	Copula KDE draw	0.06700724	0.1544588	1.883036181
17		=CopulaKDE(B2:D12,, B14:D14)		

Figure 43: *Example: Simulating a multivariate draw using `CopulaKDE` with kernel-specific marginals.*

Assumptions

- The copula assumption is elliptical, based on Gaussian dependence.
- Marginals are assumed independent during KDE estimation; joint dependence is applied post hoc.
- This is not a multivariate kernel density estimator (MVKDE), which smooths directly in joint space using multivariate kernels. For MVKDE estimation procedures see section 5.7.

5.5 Copula-Based Simulation with Mixed Marginals

Function: `CopulaMixed`

Overview

The `CopulaMixed` function simulates a single multivariate draw from a correlated distribution using a *Gaussian copula* and user-specified marginal distributions. This allows users to flexibly combine empirical, parametric, and uniform marginal types while preserving the correlation structure found in historical data.

Function Signature

```
=CopulaMixed(dataRange, [marginalTypes])
```

Inputs

- `dataRange`: A range with n rows and k columns of historical data. Each column represents one variable.
- `marginalTypes` (optional): A horizontal range or comma-separated string of k values specifying the marginal distribution for each column. Accepted types (case-insensitive):
  - `"normal"`: Fits a normal distribution via sample mean and standard deviation (see Section 3.1.9).
  - `"empirical"`: Uses empirical distribution via rank interpolation (see Section 3.2.1).
  - `"uniform"`: Assumes data are uniformly distributed between column minimum and maximum (see Section 3.1.11).
  - `"beta"`: Fits a Beta distribution using method-of-moments after rescaling to $(0, 1)$ (see Section 3.1.2).
- If `marginalTypes` is omitted, all marginals default to `"normal"`.

Simulation Procedure

The simulation proceeds in the following steps:

1. **Transform historical data to uniform standard deviates (U-space)**:

   - For each column of `dataRange`, apply the CDF corresponding to the chosen marginal type (empirical, normal, uniform, beta, etc.) to convert raw data into historical uniform values $U_{ij} \in (0, 1)$.

2. **Standardize U-space into Gaussian Z-space**:

   - Convert each uniform standard deviate U_{ij} to a standard normal score Z_{ij} via the inverse standard normal CDF: $Z_{ij} = \Phi^{-1}(U_{ij})$.

3. **Estimate the dependence structure**:

   - Calculate the correlation matrix Σ from the historical Z matrix.

4. **Generate a new correlated sample**:

   (a) Perform Cholesky decomposition: $\Sigma = LL^\top$.

   (b) Generate independent standard normals $Z \sim \mathcal{N}(0, I_k)$.

   (c) Calculate correlated standard normal deviates via $S = LZ$.

   (d) Convert S_j to correlated uniform standard deviates: $CUSD_j = \Phi(S_j)$.

5. **Simulate from marginals using inverse transform sampling**:

   - For each column j, apply the inverse CDF corresponding to the selected marginal type to transform $CUSD_j$ into a simulated value:

$$X_j^{\text{sim}} = F_j^{-1}(CUSD_j)$$

Output

Returns a $1 \times k$ array representing a single multivariate draw where:

- The marginal distribution of each variable matches the user-specified type.
- The correlation structure mimics the historical dependence via Gaussian copula.

Notes

- If `marginalTypes` has fewer than k entries, the remaining columns default to `"normal"`.

Example

The following example simulates a 3-variable draw where:

- Column 1 (X) is modeled with empirical distribution,
- Column 2 (Y) uses a beta distribution,
- Column 3 (Z) assumes a normal distribution.

	A	B	C	D
1				
2		**X**	**Y**	**Z**
3		0.084	-25.604	-11.540
4		-0.071	-0.489	-0.477
5		-0.321	28.631	15.881
6		-0.136	5.044	1.689
7		0.156	3.750	3.369
8		-0.127	0.350	0.879
9		-0.268	4.301	1.690
10		-0.082	16.390	11.001
11		0.314	-5.733	-2.588
12		0.049	-21.959	-10.178
13		0.278	-31.746	-35.667
14		-0.252	-4.682	-2.456
15		-0.280	-7.025	8.655
16		-0.080	16.218	4.266
17		-0.101	17.101	8.676
18		-0.123	33.983	8.887
19		-0.073	-9.915	6.898
20		-0.025	-11.071	-3.791
21		0.028	-2.210	-1.680
22		0.159	4.330	1.830
23		0.151	-11.242	-3.259
24		-0.192	-2.011	-1.148
25		-0.111	3.593	-0.937
26				
27	**Marginals**	Empirical	Beta	Normal
28				
29		-0.101167026	3.572306653	4.613181836
30		=CopulaMixed(B3:D25, B27:D27)		

Figure 44: *Example: Multivariate simulation using* ***`CopulaMixed`*** *with mixed marginal types.*

Assumptions

- The copula is Gaussian (elliptical dependence).
- Marginal fitting is done independently before copula dependence is imposed.
- The method does not model tail dependence beyond that captured by the correlation matrix.

5.6 Multivariate Empirical

Function: `MVEMPIRICAL`

Overview

The `MVEMPIRICAL` function generates a single multivariate simulation from historical data by preserving the empirical marginal distributions and the observed correlation structure. The procedure closely follows the method described in Richardson, Klose, and Gray (2000). This approach is particularly useful for modeling dependence among random variables without assuming any parametric or kernel-based form for the marginals.

This is a nonparametric alternative to `CopulaKDE` where empirical distributions are used instead of kernel density estimates. For details on empirical distribution construction, refer to Section 3.2.1.

Function Signature

```
=MVEMPIRICAL(dataRange)
```

Inputs

- `dataRange`: Data with n rows and k columns. Each column represents one variable.

Method Description

The procedure follows these steps:

1. Generate the empirical correlation matrix Σ from `dataRange`.
2. Perform Cholesky decomposition $\Sigma = LL^\top$.
3. Generate a vector of independent standard normal values: $Z \sim \mathcal{N}(0, I_k)$.
4. Transform Z into correlated normals: $S = LZ$.
5. Convert each S_j to a correlated uniform standard deviate $cusd_j = \Phi(S_j)$ using the standard normal CDF Φ.

6. Use each $cusd_j$ to draw a sample from the empirical distribution of the corresponding column via:

$$x_j = \texttt{EMPIRICAL}(X_j, cusd_j)$$

Output

Returns a $1 \times k$ array:

- Each value x_j is a simulated value drawn from the empirical distribution of the jth variable.
- The dependence structure among columns is preserved through the Gaussian copula.

Example

The example below shows a 3-variable simulation using `MVEMPIRICAL`.

	A	B	C	D
1		X	Y	Z
2		-0.205	-0.246	-11.545
3		-0.200	-0.030	-0.482
4		-0.342	-0.264	15.878
5		-0.381	-0.295	1.687
6		-0.089	-0.222	3.367
7		0.220	0.189	0.878
8		0.145	0.143	1.690
9		-0.028	0.071	11.002
10		0.379	0.230	-2.587
11		0.612	0.327	-10.175
12		0.738	0.492	-35.663
13				
14	Multivariate empirical draw	-0.307752	-0.2668023	2.062439823
15		=MVEMPIRICAL(B2:D12)		

Figure 45: *Example: Simulating a multivariate empirical draw*

References

- Richardson, J.W., Klose, S.L, & Gray, A.W. (2000). "An Applied Procedure for Estimating and Simulating Multivariate Empirical (MVE) Probability Distributions In Farm-Level Risk Assessment and Policy Analysis". *Journal of Agricultural and Applied Economics* 32(2):299-315.

5.7 Multivariate Kernel Density Estimation (MVKDE)

Function: `MVKDE`

Overview

The `MVKDE` function generates a random sample from a **multivariate kernel density estimate** (KDE) using Gaussian kernels. Sampling is performed in standardized space, and the result is transformed back to the original units. Multivariate KDE is a nonparametric estimator of an unknown multivariate density. Given M observations of K variables, the estimator is

$$\hat{f}(\mathbf{v} \mid \mathbf{H}, \breve{\mathbf{V}}) \;=\; \frac{1}{M} \sum_{m=1}^{M} |\mathbf{H}|^{-\frac{1}{2}} \, \kappa\Big(\mathbf{H}^{-1/2}(\mathbf{v} - \breve{\mathbf{V}}_m)\Big) \,,$$

where κ is a symmetric kernel with mean $\mathbf{0}$. With a multivariate standard normal kernel,

$$\hat{f}(\mathbf{v} \mid \mathbf{H}, \breve{\mathbf{V}}) = \frac{1}{M} \sum_{m=1}^{M} |\mathbf{H}|^{-1/2} (2\pi)^{-K/2} \exp\Big(-\tfrac{1}{2}(\mathbf{v} - \breve{\mathbf{V}}_m)^\top \mathbf{H}^{-1} (\mathbf{v} - \breve{\mathbf{V}}_m)\Big) \,.$$

This function draws from the KDE by:

- selecting a random observation as a kernel center,
- adding correlated Gaussian noise with covariance $\mathbf{H}$ (in standardized space),
- mapping the result back to the original scale.

Function Signature:

```
=MVKDE(dataRange, [bandwidth], [usd], [Hopt])
```

Inputs:

- `dataRange`: An $M \times K$ range (rows = observations, columns = variables).
- `bandwidth` (optional): `"scott"` (default) or `"silverman"` for rule-of-thumb diagonal bandwidths in standardized space. Ignored if `Hopt` is supplied.
- `usd` (optional): A uniform standard deviate. If omitted, the one is generated internally.
- `Hopt` (optional): A $K \times K$ custom bandwidth matrix in *original* units. If supplied, it overrides `bandwidth`.

Output: Returns a $1 \times K$ row vector (spills across columns) representing a single draw from the MVKDE in original units.

Methodology:

1. **Standardize the data** using population statistics (divide by M):

$$z_{ij} = \frac{x_{ij} - \bar{x}_j}{\hat{\sigma}_j}, \qquad \bar{x}_j = \frac{1}{M}\sum_{i=1}^{M} x_{ij}, \quad \hat{\sigma}_j = \sqrt{\frac{1}{M}\sum_{i=1}^{M}(x_{ij} - \bar{x}_j)^2}.$$

   (Columns with zero variance use a fallback $\hat{\sigma}_j = 1$.) Let $D = \mathrm{diag}(\hat{\sigma}_1, \ldots, \hat{\sigma}_K)$.

2. **Form the bandwidth in standardized space.** Denote s_j^* as the population standard deviation of standardized column j (typically $s_j^* \approx 1$).

$$\text{Scott (diag):} \quad \sqrt{H_{jj}^*} \;=\; s_j^*\, M^{-1/(K+4)},$$

$$\text{Silverman (diag):} \quad \sqrt{H_{jj}^*} \;=\; s_j^*\, \big(M(K+2)/4\big)^{-1/(K+4)}.$$

   If a custom matrix `Hopt` is supplied in original units, map it into standardized space via

$$\mathbf{H}^* \;=\; D^{-1}\, \mathbf{H}_{\text{opt}}\, D^{-1}.$$

   (Rule-of-thumb bandwidths are diagonal; off-diagonal elements are set to zero.)

3. **Sample in standardized space.** Pick a center $\boldsymbol{\mu}$ from the standardized data, draw $\mathbf{Z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$, compute the Cholesky $\mathbf{H}^* = LL^\top$, and set

$$\mathbf{X}^* \;=\; \boldsymbol{\mu} + L\mathbf{Z}.$$

4. **Un-standardize to original units:**

$$\mathbf{X} \;=\; \bar{\mathbf{x}} + D\,\mathbf{X}^*.$$

Bandwidth Selection (details):

- **Scott's rule (default, standardized diag):**

$$H_{jj}^* \;=\; \big(s_j^*\, M^{-1/(K+4)}\big)^2.$$

- **Silverman's rule (standardized diag):**

$$H^*_{jj} \;=\; \left(s^*_j \left(M(K+2)/4\right)^{-1/(K+4)}\right)^2.$$

- **Special case (K=2):** Silverman reduces to Scott because the ratio

$$\frac{H^*_{\text{Silverman}}}{H^*_{\text{Scott}}} \;=\; \left(\frac{4}{K+2}\right)^{\frac{2}{K+4}}$$

  equals 1 when $K = 2$.

- **Custom bandwidth:** If `Hopt` is provided in original units, it is mapped as $\mathbf{H}^* = D^{-1}\mathbf{H}_{\text{opt}}D^{-1}$ and used directly (overriding the rules).

Example: The figure below shows a sample Excel implementation with three variables `X`, `Y`, and `Z`, and 23 observations. The function generates a single multivariate draw from the KDE using Scott's rule.

```
=MVKDE(B3:D25, "SCOTT")
```

- Cells `B3:D25` contain the input data
- Row `B27:D27` shows the simulated multivariate draw

	A	B	C	D
1				
2		**X**	**Y**	**Z**
3		0.084	-25.604	-11.540
4		-0.071	-0.489	-0.477
5		-0.321	28.631	15.881
6		-0.136	5.044	1.689
7		0.156	3.750	3.369
8		-0.127	0.350	0.879
9		-0.268	4.301	1.690
10		-0.082	16.390	11.001
11		0.314	-5.733	-2.588
12		0.049	-21.959	-10.178
13		0.278	-31.746	-35.667
14		-0.252	-4.682	-2.456
15		-0.280	-7.025	8.655
16		-0.080	16.218	4.266
17		-0.101	17.101	8.676
18		-0.123	33.983	8.887
19		-0.073	-9.915	6.898
20		-0.025	-11.071	-3.791
21		0.028	-2.210	-1.680
22		0.159	4.330	1.830
23		0.151	-11.242	-3.259
24		-0.192	-2.011	-1.148
25		-0.111	3.593	-0.937
26				
27	**Multivariate KDE**	0.155376917	-29.23946999	-13.15322457
28		=MVKDE(B3:D25, "SCOTT")		

Figure 46: *Example: Sampling from Multivariate KDE using* `MVKDE` *with Scott's Rule*

Notes:

- Works for any dimension $K \geq 1$; rule-of-thumb bandwidths are *diagonal in standardized space.*
- Uses population mean/standard deviation for standardization and bandwidth computation.
- Constant columns are handled safely (fallback $\hat{\sigma}_j = 1$).
- Based on `funcsim`: `https://github.com/h-bryant/funcsim`.

6 Validation

6.1 Screening Diagnostics

Function: `ScreenSeries`

Overview

The `ScreenSeries` function conducts a series of diagnostic tests to determine whether a given univariate data series meets key assumptions required for simulation or further modeling. Specifically, it checks for:

- Stability of the mean (absence of structural break),
- Homoskedasticity (constant variance),
- Lack of serial correlation (no autocorrelation),
- Overall white noise behavior using the Ljung-Box test.

Each test is reported along with its critical value, test statistic, p-value, and a binary PASS/FAIL decision.

Function Signature

```
=ScreenSeries(dataRange, [maxLag], [numDiff], [alpha])
```

Arguments:

- `dataRange` (required). A numeric range representing a univariate time series.
- `maxLag` (optional). Maximum lag to use in autocorrelation and Ljung-Box tests. Default = 5.
- `numDiff` (optional). Number of differences to apply prior to autocorrelation calculations. Default = 0.
- `alpha` (optional). Significance level used to determine critical values. Default = 0.05.

Returns A 5-row x 5-column array with labeled results for each diagnostic test. The columns are:

- Test name,
- Critical value (based on significance level),
- Test statistic,
- p-value (if available),
- PASS/FAIL decision.

Diagnostic Tests and Formulas

1. Mean Stability (t-Test). Tests whether the series has a structural break in the mean. The sample is split into two halves, and a pooled two-sample t-test is performed:

$$t = \frac{|\bar{x}_1 - \bar{x}_2|}{\sqrt{\frac{s_1^2}{n_1} + \frac{s_2^2}{n_2}}}$$

where $\bar{x}_1, \bar{x}_2$ and s_1^2, s_2^2 are the means and variances of each half. The test uses degrees of freedom $df = n_1 + n_2 - 2$.

2. Variance Stability (F-Test). Tests for homoskedasticity by comparing the variances of the two halves:

$$F = \frac{\max(s_1^2, s_2^2)}{\min(s_1^2, s_2^2)}$$

A two-tailed F-test is used with degrees of freedom (df_1, df_2) from each sample.

3. Maximum Autocorrelation t-Statistic. Computes the t-statistic for sample autocorrelation at lags 1 through h, where $h =$ `maxLag`. The maximum absolute t-value is reported:

$$t_k = \frac{r_k}{\mathrm{SE}(r_k)}$$

with r_k as the lag-k sample autocorrelation and SE estimated via Bartlett's formula.

4. Ljung-Box Q-Test. Performs a joint test for zero autocorrelation up to lag h:

$$Q = n(n+2) \sum_{k=1}^{h} \frac{r_k^2}{n-k}$$

Under the null hypothesis of no autocorrelation, Q follows a chi-square distribution with h degrees of freedom.

Example

The following example tests a sample of 23 time series values using a default 5-lag structure and default $\alpha = 0.05$:

```
=ScreenSeries(A2:A24)
```

	A	B	C	D	E	F	G
1	Y						
2	0.08358143						
3	-0.07142839		Test Name	Critical Value	Test Statistic	p-Value	Decision
4	-0.3209785		Mean Stability (Two-Sample t-Test)	2.080	0.8597	0.3996	PASS
5	-0.13614477		Variance Stability (F-Test for Equality of Variances)	3.526	2.2482	0.2000	PASS
6	0.15641202		Max ACF |t-stat| (lag 1-5)	2.080	-1.3352	0.1961	PASS
7	-0.12731907		Ljung-Box Q (lag 5)	p > 0.050	4.6702	0.4574	PASS
8	-0.26809148						
9	-0.08199897		=ScreenSeries(A2:A24)				
10	0.31423754						
11	0.04925379						
12	0.27841112						
13	-0.25245737						
14	-0.28049297						
15	-0.08024353						
16	-0.10076545						
17	-0.12348106						
18	-0.07333127						
19	-0.02531781						
20	0.02795608						
21	0.15912891						
22	0.15075509						
23	-0.19150061						
24	-0.11109941						

Figure 47: *Example output of the* `ScreenSeries` *function with labeled results.*

Note: This function returns a spilled array in Excel 365+. For older Excel versions, use `Ctrl+Shift+Enter`*.*

References

- Box, G.E.P., Jenkins, G.M., Reinsel, G.C., & Ljung, G.M. (2016). *Time Series Analysis: Forecasting and Control*. Wiley.
- Greene, W.H. (2018). *Econometric Analysis* (8th ed.). Pearson.

6.2 Test for Normality

6.2.1 Shapiro-Wilk Test for Normality

Function: `SHAPIROWILK`

Overview

The `SHAPIROWILK` function implements the **Shapiro-Wilk test** to assess whether a sample is drawn from a normally distributed population. It is widely considered one of the most powerful tests for normality, particularly effective for small to moderate sample sizes.

The null hypothesis is that the data are normally distributed. A low p-value indicates that the sample deviates significantly from normality.

Test Statistic

The test statistic W is computed as:

$$W = \frac{\left(\sum_{i=1}^{m} a_i (x_{n-i+1} - x_i)\right)^2}{\sum_{i=1}^{n} (x_i - \bar{x})^2}$$

where:

- x_i are the ordered sample values ($x_1 \leq x_2 \leq \cdots \leq x_n$)
- $\bar{x}$ is the sample mean
- a_i are pre-calculated Shapiro-Wilk coefficients derived from expected values of order statistics under normality; this implementation uses published tables from Conover (1999), Table A16.
- $m = \lfloor n/2 \rfloor$ is the number of symmetric pairs used; each a_i is applied to the difference between values at positions x_i and x_{n-i+1}

The numerator captures the symmetry of the distribution's tails, while the denominator measures overall variance. A value of W near 1 suggests normality; lower values indicate departures from normality.

Estimation Method

1. The data are sorted and centered by their mean.
2. Symmetric weights a_i are applied to the tails of the distribution.
3. The W statistic is calculated using the formula above.
4. The p-value is approximated using Royston's (1995) algorithm, which transforms W and applies a normal approximation.
5. A decision is returned based on whether $p < \alpha$.

Function Signature

```
=SHAPIROWILK(dataRange, [alpha])
```

Arguments

- `dataRange` (required). A range of numeric values (sample size n must be between 3 and 50).
- `alpha` (optional). Significance level for hypothesis testing (default = 0.05).

Returns An array containing:

- The test statistic W
- The p-value associated with W
- A decision string: "Reject H_0 that data are normally distributed" or "Fail to reject H_0"

Example The example below shows the `SHAPIROWILK` function applied to a data series with significance level $\alpha = 0.05$. The first output is the test statistic, the second is the corresponding p-value, and the third is the decision result as a string (e.g., "Reject" or "Fail to Reject").

	A	B	C	D	E	F	G	H	I
1		Y							
2	1	5.049298							
3	2	8.283555		0.935802	0.145945	Fail to reject H0 that data are normally distributed			
4	3	7.166893		=SHAPIROWILK(B2:B24,0.05)					
5	4	5.95053							
6	5	3.90633							
7	6	0.065137							
8	7	-9.674403							
9	8	-20.4396							
10	9	-11.18239							
11	10	-12.61458							
12	11	-11.84254							
13	12	-11.58898							
14	13	-8.609179							
15	14	-12.9468							
16	15	-6.02482							
17	16	0.434547							
18	17	14.20449							
19	18	10.79267							
20	19	10.18514							
21	20	10.05754							
22	21	-2.127549							
23	22	15.32632							
24	23	15.62838							

Figure 48: *Example: Shapiwo-Wilk Test for Normality*

Assumptions and Notes

- Valid for sample sizes between 3 and 50. Outside this range, the function returns an error.
- The p-value is based on an approximation using the Royston (1995) method. For higher-precision results, use implementations such as `scipy.stats.shapiro` in Python, which rely on Royston's original Fortran algorithm and provide reliable p-values for sample sizes up to 5000 (Razali and Wah (2011)).
- The test assumes independent and identically distributed (i.i.d.) observations.

References

- Conover, W.J. (1999). *Practical Nonparametric Statistics*. 3rd ed., Wiley, Table A16.
- Shapiro, S.S., & Wilk, M.B. (1965). "An analysis of variance test for normality (complete samples)." *Biometrika*, 52(3/4), 591–611.
- Royston, P. (1995). "A remark on AS 181: The W test for normality." *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, 44(4), 547–551.
- Razali, N.M., & Wah, Y.B. (2011). "Power comparisons of Shapiro-Wilk, Kolmogorov-Smirnov, Lilliefors, and Anderson-Darling tests." *Journal of Statistical Modeling and*

Analytics, 2(1), 21-33.

6.2.2 Kolmogorov-Smirnov Test for Normality

Function: `KSNORMAL`

Overview

The `KSNORMAL` function performs the **Kolmogorov-Smirnov (K-S) test** to evaluate whether a sample of data is drawn from a normal distribution. The test compares the empirical cumulative distribution function (ECDF) of the sample to the cumulative distribution function (CDF) of a theoretical normal distribution with the same sample mean and standard deviation.

The null hypothesis is that the data are normally distributed. A large maximum difference between the two distributions implies a deviation from normality.

Test Statistic

The K-S statistic is defined as:

$$D = \max_{1 \leq i \leq n} |F_n(x_i) - F(x_i)|$$

where:

- $F_n(x_i) = \frac{i}{n}$ is the empirical CDF at the i-th ordered value
- $F(x_i)$ is the theoretical CDF from a normal distribution with sample mean μ and sample standard deviation σ
- D is the maximum absolute difference between the ECDF and the normal CDF

Estimation Method

1. The method involves sorting the sample values into ascending order.

2. Calculate the sample mean μ and sample standard deviation σ.

3. For each sorted value x_i, calculate:

$$\text{Empirical } F_n(x_i) = \frac{i}{n}, \quad \text{Theoretical } F(x_i) = \Phi\left(\frac{x_i - \mu}{\sigma}\right)$$

4. Calculate the test statistic $D = \max |F_n(x_i) - F(x_i)|$

5. Compare D against a critical value obtained from standard K-S tables or asymptotic approximations.

Function Signature

```
=KSNORMAL(dataRange, [alpha])
```

Arguments

- `dataRange` (required). A range of numeric values (sample size n must be between 3 and 50).
- `alpha` (optional). Significance level for hypothesis testing (default = 0.05).

Returns An array containing:

- The D test statistic (maximum deviation between empirical and normal CDF)
- The critical value for the given sample size and α
- A decision string: "Reject H_0 that data are normally distributed" or "Fail to reject H_0"

Example The example below shows the `KSNORMAL` function applied to a data series with significance level $\alpha = 0.05$. The first output is the test statistic, the second is the corresponding critical value, and the third is the decision result as a string (e.g., "Reject" or "Fail to Reject").

	A	B	C	D	E	F	G	H	I
1		Y							
2	1	5.049298							
3	2	8.283555		0.135813	0.294	Fail to reject H0 that data are normally distributed			
4	3	7.166893		=KSNORMAL(B2:B24, 0.05)					
5	4	5.95053							
6	5	3.90633							
7	6	0.065137							
8	7	-9.674403							
9	8	-20.4396							
10	9	-11.18239							
11	10	-12.61458							
12	11	-11.84254							
13	12	-11.58898							
14	13	-8.609179							
15	14	-12.9468							
16	15	-6.02482							
17	16	0.434547							
18	17	14.20449							
19	18	10.79267							
20	19	10.18514							
21	20	10.05754							
22	21	-2.127549							
23	22	15.32632							
24	23	15.62838							

Figure 49: *Example: Kolmogorov-Smirnov Test for Normality*

Assumptions and Notes

- The test is applied to data with parameters μ and σ estimated from the sample.
- The implementation uses exact tabulated critical values for $n \leq 50$, and asymptotic approximations for larger n:

$$D_{\text{crit}} = \frac{c(\alpha)}{\sqrt{n}}$$

  with $c(\alpha)$ values taken from standard K-S tables.
- This version compares against a normal distribution, not a fully specified distribution (i.e., it is *Lilliefors-type* test).
- The K-S test is generally considered less powerful than the Shapiro-Wilk test, especially for small to moderate sample sizes. As shown in Razali and Wah (2011), it often fails to reject the null hypothesis even when the data deviate meaningfully from normality. This is because the K-S test is sensitive to deviations in the center of the distribution but less so in the tails.

References

- Kolmogorov, A.N. (1933). "Sulla determinazione empirica di una legge di distribuzione." *Giornale dell'Istituto Italiano degli Attuari.*

- Smirnov, N. (1948). "Table for estimating the goodness of fit of empirical distributions." *Annals of Mathematical Statistics*, 19(2), 279-281.
- Razali, N.M., & Wah, Y.B. (2011). "Power comparisons of Shapiro-Wilk, Kolmogorov-Smirnov, Lilliefors, and Anderson-Darling tests." *Journal of Statistical Modeling and Analytics*, 2(1), 21-33.

6.2.3 Anderson-Darling Test for Normality

Function: `ANDERSONDARLING`

Overview

The `ANDERSONDARLING` function implements the **Anderson-Darling test** for normality. This test enhances the Kolmogorov-Smirnov approach by giving more weight to the tails of the distribution, where deviations from normality often occur (Razali and Wah (2011)).

The null hypothesis is that the data follow a normal distribution. Compared to other normality tests, Anderson-Darling is especially sensitive to tail departures and performs well in both small and large samples.

Test Statistic

The Anderson-Darling statistic A^2 is calculated as:

$$A^2 = -n - \frac{1}{n}\sum_{i=1}^{n}(2i-1)\left[\ln F(x_i) + \ln(1 - F(x_{n+1-i}))\right]$$

where:

- n is the sample size
- x_i are the ordered sample values ($x_1 \leq x_2 \leq \cdots \leq x_n$)
- $F(x_i)$ is the cumulative distribution function (CDF) of the standard normal distribution evaluated at the standardized value $z_i = \frac{x_i - \bar{x}}{s}$

An optional adjustment is applied for small sample sizes following D'Agostino and Stephens (1986):

$$A_s^2 = A^2 \cdot \left(1 + \frac{0.75}{n} + \frac{2.25}{n^2}\right)$$

This correction makes the statistic more accurate when n is small (D'Agostino and Stephens (1986)).

Function Signature

```
=ANDERSONDARLING(dataRange,[alpha], [adjustForSampleSize])
```

Arguments

- `dataRange` (required). A numeric range of observations to test.
- `alpha` (optional) Significance level (default = 0.05).
- `adjustForSampleSize` (optional). Whether to apply the small-sample correction (default = `True`).

Returns An array containing:

- The adjusted or unadjusted Anderson-Darling statistic (A_s^2 or A^2)
- The estimated p-value
- A decision string: "Reject H_0 that data are normally distributed" or "Fail to reject H_0"
- A string indicating whether the statistic is adjusted or unadjusted

Example The example below shows the `ANDERSONDARLING` function applied to a data series with significance level $\alpha = 0.05$. The first output is the test statistic, the second is the corresponding p-value, the third is the decision result as a string (e.g., "Reject" or "Fail to Reject"), and the fourth is a string indicating whether the statistic is adjusted or not.

	A	B	C	D	E	F	G
1		Y					
2	1	5.049298					
3	2	8.283555		0.582915	0.129315	Fail to reject H0 that data are normally distributed	Adjusted
4	3	7.166893		=ANDERSONDARLING(B2:B24, 0.05)			
5	4	5.95053					
6	5	3.90633					
7	6	0.065137					
8	7	-9.674403					
9	8	-20.4396					
10	9	-11.18239					
11	10	-12.61458					
12	11	-11.84254					
13	12	-11.58898					
14	13	-8.609179					
15	14	-12.9468					
16	15	-6.02482					
17	16	0.434547					
18	17	14.20449					
19	18	10.79267					
20	19	10.18514					
21	20	10.05754					
22	21	-2.127549					
23	22	15.32632					
24	23	15.62838					

Figure 50: *Example: Anderson-Darling Test for Normality*

Assumptions and Notes

- The test assumes that the data are independent and identically distributed (i.i.d.).
- This implementation uses an approximation to the p-value based on thresholds provided in statistical literature.
- The small-sample adjustment (recommended by D'Agostino and Stephens (1986)) is automatically applied unless specified otherwise.
- Compared to other tests, the Anderson-Darling test is more sensitive to deviations in the tails of the distribution. For example, Razali and Wah (2011) show that the Anderson-Darling test consistently has higher power than the Kolmogorov-Smirnov test and is competitive with or slightly below the Shapiro-Wilk test in many scenarios.

References

- Anderson, T.W., & Darling, D.A. (1952). "Asymptotic theory of certain 'goodness-of-fit' criteria based on stochastic processes." *Annals of Mathematical Statistics*, 23(2), 193-212.
- D'Agostino, R.B., & Stephens, M.A. (1986). *Goodness-of-Fit Techniques*. Vol. 68. CRC press.

- Razali, N.M., & Wah, Y.B. (2011). "Power comparisons of Shapiro-Wilk, Kolmogorov-Smirnov, Lilliefors, and Anderson-Darling tests." *Journal of Statistical Modeling and Analytics*, 2(1), 21-33.

6.2.4 Cramer-von Mises Test for Normality

Function: `CRAMERVONMISES`

Overview

The `CRAMERVONMISES` function implements the **Cramer-von Mises (CvM) test** for assessing whether a dataset follows a normal distribution. It is a goodness-of-fit test that compares the empirical distribution of the data to the expected normal distribution by summing squared differences across the entire range. Unlike the Kolmogorov-Smirnov test, which focuses on the maximum deviation, the Cramer-von Mises test gives equal weight to all parts of the distribution.

The null hypothesis is that the data are drawn from a normal distribution with unknown mean and variance.

Test Statistic

The Cramer-von Mises test statistic W^2 is calculated as:

$$W^2 = \sum_{i=1}^{n} \left[F(z_i) - \frac{2i-1}{2n} \right]^2 + \frac{1}{12n}$$

where:

- x_i are the ordered sample values
- $z_i = \frac{x_i - \bar{x}}{s}$ is the standardized value of x_i
- $F(z_i)$ is the cumulative distribution function (CDF) of the standard normal distribution evaluated at z_i
- n is the sample size

This statistic measures the squared distance between the empirical distribution function and the hypothesized normal CDF, with a correction term $\frac{1}{12n}$ to account for bias in finite samples.

Function Signature

```
=CRAMERVONMISES(dataRange, [alpha])
```

Arguments

- `dataRange` (required). A numeric range of observations to test.
- `alpha` (optional) Significance level (default = 0.05).

Returns An array containing:

- The W^2 test statistic
- The approximate p-value
- A decision string: "Reject H_0 that data are normally distributed" or "Fail to reject H_0"

Example The example below shows the `CRAMERVONMISES` function applied to a data series with significance level $\alpha = 0.05$. The first output is the test statistic, the second is the approximated p-value, and the third is the decision result as a string (e.g., "Reject" or "Fail to Reject").

	A	B	C	D	E	F
1		Y				
2	1	5.049298				
3	2	8.283555		0.09084	0.141335	Fail to reject H0 that data are normally distributed
4	3	7.166893		=CRAMERVONMISES(B2:B24, 0.05)		
5	4	5.95053				
6	5	3.90633				
7	6	0.065137				
8	7	-9.674403				
9	8	-20.4396				
10	9	-11.18239				
11	10	-12.61458				
12	11	-11.84254				
13	12	-11.58898				
14	13	-8.609179				
15	14	-12.9468				
16	15	-6.02482				
17	16	0.434547				
18	17	14.20449				
19	18	10.79267				
20	19	10.18514				
21	20	10.05754				
22	21	-2.127549				
23	22	15.32632				
24	23	15.62838				

Figure 51: *Example: Cramer - von Mises Test for Normality*

Assumptions and Notes

- This implementation assumes that mean and variance are unknown and estimated from the data.
- The p-value is calculated using piecewise polynomial approximations from Table 4.9 in D'Agostino and Stephens (1986), which are appropriate when parameters are estimated.
- The Cramer-von Mises test is generally less sensitive to tail deviations than the Anderson-Darling test, but more evenly sensitive across the entire distribution (D'Agostino and Stephens (1986)).

References

- D'Agostino, R.B., & Stephens, M.A. (1986). *Goodness-of-Fit Techniques*. Vol. 68. CRC press.

6.2.5 Jarque-Bera Test for Normality

Function: `JARQUEBERA`

Overview

The `JARQUEBERA` function implements the **Jarque-Bera** (JB) test for normality, a popular method used in econometrics to evaluate whether a dataset's skewness and kurtosis match those of a normal distribution.

The null hypothesis is that the data are drawn from a normally distributed population. The test jointly examines whether the sample skewness is zero and the excess kurtosis is zero. It is most appropriate for medium to large sample sizes.

Test Statistic

The JB statistic is defined as:

$$\text{JB} = \frac{n}{6}\left(S^2 + \frac{(K-3)^2}{4}\right)$$

where:

- n is the sample size
- S is the sample skewness
- K is the sample kurtosis

Under the null hypothesis of normality, the JB statistic asymptotically follows a chi-squared distribution with 2 degrees of freedom:

$$\text{JB} \sim \chi^2_2$$

A high JB value implies departure from normality due to either skewness, kurtosis, or both.

Function Signature

```
=JARQUEBERA(dataRange, [alpha])
```

Arguments

- `dataRange` (required). A numeric range of observations to test.
- `alpha` (optional) Significance level (default = 0.05).

Returns An array containing:

- The JB test statistic
- The p-value associated with the statistic
- A decision string: "Reject H_0 that data are normally distributed" or "Fail to reject H_0"

Example The example below shows the `JARQUEBERA` function applied to a data series with significance level $\alpha = 0.05$. The first output is the test statistic, the second is the corresponding p-value, and the third is the decision result as a string (e.g., "Reject" or "Fail to Reject").

	A	B	C	D	E	F
1		Y				
2	1	5.049298				
3	2	8.283555		1.606592	0.44785	Fail to reject H0 that data are normally distributed
4	3	7.166893		=JARQUEBERA(B2:B24, 0.05)		
5	4	5.95053				
6	5	3.90633				
7	6	0.065137				
8	7	-9.674403				
9	8	-20.4396				
10	9	-11.18239				
11	10	-12.61458				
12	11	-11.84254				
13	12	-11.58898				
14	13	-8.609179				
15	14	-12.9468				
16	15	-6.02482				
17	16	0.434547				
18	17	14.20449				
19	18	10.79267				
20	19	10.18514				
21	20	10.05754				
22	21	-2.127549				
23	22	15.32632				
24	23	15.62838				

Figure 52: *Example: Jarque-Bera Test for Normality*

Assumptions and Notes

- The test assumes the data are independently and identically distributed (i.i.d.).
- The accuracy of the chi-squared approximation improves with larger sample sizes.

References

- Jarque, C.M., & Bera, A.K. (1987). "A Test for Normality of Observations and Regression Residuals." *International Statistical Review*, 55(2), 163-172.
- Gujarati, D.N. & D.C. Porter. (2009). *Basic Econometrics*, 5th ed. McGraw-Hill.

6.3 Validating Simulated Means Against Historical Data Using Welch's Two-Sample t-Test

Function: `CompareMeans`

Overview

The `CompareMeans` function is used to validate the similarity between simulated and historical series by comparing their sample means. The function implements **Welch's two-sample t-test**, a statistical test used to determine whether the means of two independent samples

are significantly different. Unlike the standard Student's t-test, Welch's test does not assume equal variances between the two groups, making it more suitable in practical applications.

Test Statistic

The Welch t-statistic is calculated as:

$$t = \frac{\bar{X}_1 - \bar{X}_2}{\sqrt{\frac{s_1^2}{n_1} + \frac{s_2^2}{n_2}}}$$

where:

- $\bar{X}_1, \bar{X}_2$ are the sample means
- s_1^2, s_2^2 are the sample variances
- n_1, n_2 are the sample sizes

The degrees of freedom are estimated using the Welch-Satterthwaite equation:

$$df = \frac{\left(\frac{s_1^2}{n_1} + \frac{s_2^2}{n_2}\right)^2}{\frac{(s_1^2/n_1)^2}{n_1 - 1} + \frac{(s_2^2/n_2)^2}{n_2 - 1}}$$

VBA Function Signature

```
=CompareMeans(dataRange1, dataRange2, [alpha])
```

Arguments

- `dataRange1` (required). A numeric range representing the first sample.
- `dataRange2` (required). A numeric range representing the second sample.
- `alpha` (optional). Significance level for the hypothesis test (default = 0.05).

Returns A 1-row summary table:

- Test name (e.g., "2 Sample t Test")
- The calculated t-statistic (rounded)
- The critical t value
- The two-tailed p-value
- A decision string: "Reject H_0 that the Means are Equal" or "Fail to Reject H_0"

Example The example below shows the `COMPAREMEANS` function applied to historical and simulated data series using a significance level of $\alpha = 0.05$. Only the first 10 historical and simulated series are displayed to conserve space. The function outputs: (1) the test name, (2) the calculated t-statistic, (3) the critical t-value, (4) the two-tailed p-value, and (5) the decision result as a string (e.g., "Reject" or "Fail to Reject").

	A	B	C	D	E	F	G	H	I	J	K	L
2	Average	-0.00774872										
3	Median	-0.105368649										
4	Minimum	-0.486526743										
5	Maximum	0.850363189										
6	Standard Deviation	0.296398472										
7	25th percentile	-0.220772398										
8	75th percentile	0.180321373										
9												
10												
11	**Trials**	**Simulated series**	**Historical series**		2 Sample t Test	-0.08	2.06	0.939	Fail to Reject the Ho that the Means are Equal			
12	1	-0.29417691	-0.205420296		=CompareMeans(B12:B511,C12:C34, 0.05)							
13	2	-0.046435319	-0.199785655									
14	3	-0.095955186	-0.341557859									
15	4	-0.246905863	-0.38138028									
16	5	0.048004634	-0.089114305									
17	6	0.284624594	0.220285261									
18	7	0.256066553	0.144879318									
19	8	-0.139628413	-0.027565747									
20	9	0.334792056	0.379479487									
21	10	-0.149891755	0.611630003									

Figure 53: *Example: Two sample t test to compare means*

Assumptions and Notes

- The p-value is based on the two-tailed test.
- The test is sensitive to extreme outliers and assumes independence of observations.

References

- Welch, B.L. (1947). "The generalization of Student's problem when several different population variances are involved." *Biometrika*, 34(1/2), 28-35.
- Richardson, J. W., K. Schumann, & P. Feldman. (2004). *Technical Guide for Simetar*. College Station, Texas.
- Wikipedia: Welch's t-test

6.4 Validating Simulated Variances Against Historical Variances Using F-test

Function: `CompareVariances`

Overview

The `CompareVariances` function implements the **F-test** to determine whether the variances of historical and simulated series are equal.

Test Statistic

The F-statistic is defined as the ratio of the larger sample variance to the smaller:

$$F = \frac{s_1^2}{s_2^2}$$

where:

- s_1^2 and s_2^2 are the sample variances of the two groups (with $s_1^2 \geq s_2^2$)
- $df_1 = n_1 - 1$, $df_2 = n_2 - 1$ are the degrees of freedom for the numerator and denominator

The statistic follows an F-distribution under the null hypothesis that the two variances are equal.

VBA Function Signature

```
=CompareVariances(dataRange1, dataRange2, [alpha])
```

Arguments

- `dataRange1` (required). A numeric range representing the first sample.
- `dataRange2` (required). A numeric range representing the second sample.
- `alpha` (optional). Significance level for the hypothesis test (default = 0.05).

Returns A 1-row array with:

- Test name (e.g., "2 Sample F Test")
- The calculated F-statistic (rounded)
- The critical value from the F-distribution
- The right-tail p-value
- A decision string: "Reject H_0 that the Variances are Equal" or "Fail to Reject H_0"

Example The example below shows the `COMPAREVARIANCES` function applied to historical and simulated data series using a significance level of $\alpha = 0.05$. Only the first 10 historical and simulated series are displayed to conserve space. The function outputs: (1) the test name, (2) the calculated F-statistic, (3) the critical value from F-distribution, (4) the right-tail p-value, and (5) the decision result as a string (e.g., "Reject" or "Fail to Reject").

	A	B	C	D	E	F	G	H	I	J	K	L
2	Average	-0.00774872										
3	Median	-0.105368649										
4	Minimum	-0.486526743										
5	Maximum	0.850363189										
6	Standard Deviation	0.296398472										
7	25th percentile	-0.220772398										
8	75th percentile	0.180321373										
9												
10												
11	**Trials**	**Simulated series**	**Historical series**		2 Sample F Test	1	1.56	0.458	Fail to Reject the Ho that the Variances are Equal			
12	1	-0.29417691	-0.205420296		=CompareVariances(B12:B511, C12:C34, 0.05)							
13	2	-0.046435319	-0.199785655									
14	3	-0.095955186	-0.341557859									
15	4	-0.246905863	-0.38138028									
16	5	0.048004634	-0.089114305									
17	6	0.284624594	0.220285261									
18	7	0.256066553	0.144879318									
19	8	-0.139628413	-0.027565747									
20	9	0.334792056	0.379479487									
21	10	-0.149891755	0.611630003									

Figure 54: *Example: Two sample F test to compare variances*

Assumptions and Notes

- The F-test assumes that both samples are drawn from normally distributed populations.
- The test is sensitive to deviations from normality, particularly when sample sizes are small.
- The test only detects variance inequality, not which group has greater dispersion.

References

- D'Agostino, R.B., & Stephens, M.A. (1986). *Goodness-of-Fit Techniques*. Vol. 68. CRC press.
- Richardson, J. W., K. Schumann, & P. Feldman. (2004). *Technical Guide for Simetar*. College Station, Texas.
- Wikipedia: F-test of equality of variances

6.5 Multivariate Tests: Hotelling's T^2, Box's M, and Complete Homogeneity

Function: `MultivariateTests`

Overview

The `MultivariateTests` function performs three multivariate statistical tests to jointly assess whether two samples-typically historical and simulated data-differ in terms of their mean vectors, covariance matrices, or both.

Specifically, the function implements:

- **Hotelling's Two-Sample T^2 Test** - to test equality of mean vectors.
- **Box's M Test** - to test equality of covariance matrices.
- **Complete Homogeneity Test** - a likelihood ratio test to evaluate whether both the means and covariances are equivalent.

Each test produces a test statistic, a critical value at the specified significance level, a p-value, and a decision string regarding the null hypothesis.

Test 1: Hotelling's T^2 Test

This test evaluates whether the mean vectors of two multivariate samples are equal, assuming equal covariance matrices:

$$T^2 = \frac{n_1 n_2}{n_1 + n_2}(\bar{x}_1 - \bar{x}_2)' S_p^{-1}(\bar{x}_1 - \bar{x}_2)$$

where:

- $\bar{x}_1, \bar{x}_2$ are sample mean vectors
- S_p is the pooled covariance matrix
- n_1, n_2 are the sample sizes

This statistic is converted to an F-distribution with degrees of freedom p and n_1+n_2-p-1 for hypothesis testing.

Test 2: Box's M Test

Box's M test evaluates whether the covariance matrices of two multivariate samples are equal. The test statistic is:

$$M = (n_1 + n_2 - 2) \ln |S_p| - (n_1 - 1) \ln |S_1| - (n_2 - 1) \ln |S_2|$$

where S_1, S_2 are the sample covariance matrices and S_p is the pooled covariance. A correction factor is applied for small samples, and the test statistic approximately follows a chi-squared distribution with:

$$\text{df} = \frac{p(p+1)}{2}$$

Test 3: Complete Homogeneity Test

This likelihood ratio test evaluates whether the two multivariate samples come from the same distribution, i.e., they have both equal means and equal covariance matrices (Anderson, 2003). It is based on the log-likelihood ratio:

$$\Lambda = -2 \left(\log L_0 - \log L_1 \right)$$

where:

- $\log L_0$: log-likelihood under H_0 (equal means and covariances)
- $\log L_1$: log-likelihood under H_1 (different means and/or covariances)

This statistic also follows a chi-squared distribution with:

$$\text{df} = p + \frac{p(p+1)}{2}$$

VBA Function Signature

```
=MultivariateTests(rangeSim, rangeHist,[alpha])
```

Arguments

- `rangeSim` (required). A range of simulated data, where rows are observations and columns are variables.
- `rangeHist` (required). A range of historical data, with matching number of columns.
- `alpha` (optional). Significance level for hypothesis testing (default = 0.05).

Returns A 4-row summary array with the following tests:

- Row 1: Column headers - `"Test", "Test Value", "Critical Value", "P-Value", "Decision"`
- Row 2: Hotelling's T^2 test summary
- Row 3: Box's M test summary
- Row 4: Complete homogeneity test summary

Example The example below shows the `MULTIVARIATETESTS` function applied to historical and simulated data series using a significance level of $\alpha = 0.05$. Only the first 10 historical and simulated series are displayed to conserve space.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
1	**LH Sampling with 500 trials.**															
2	Average	-0.007867727	-0.005776139													
3	Median	-0.105326013	-0.03883667		Test	Test Value	Critical Valu	P-Value	Decision							
4	Minimum	-0.503315763	-0.388799272		2 Sample Hotelling T2 Test	0.0066	6.0377	0.9967	Fail to Reject the Ho that the Mean Vectors are Equal							
5	Maximum	0.859946937	0.568942306		Box's M Test	0.4843	7.8147	0.9223	Fail to Reject the Ho that the Covariance Matrices are Equal							
6	Standard Deviation	0.296342234	0.220638623		Complete Homogeneity Test	0.6727	11.0705	0.9844	Fail to Reject the Ho that the Mean Vectors and Covariance Matrices are Equivalent, Respectively							
7	25th percentile	-0.220282623	-0.198787786													
8	75th percentile	0.180279138	0.153200859		=MultivariateTests(B12:C511, D12:E34, 0.05)											
9																
10																
11	**Trials**	**Simulated series 1**	**Simulated series 2**	**Historical series 1**	**Historical series 2**											
12	1	0.138061453	0.165588518	-0.205420296	-0.246147463											
13	2	-0.273683483	-0.23930333	-0.199785655	-0.030003239											
14	3	-0.089324746	0.140427212	-0.341557859	-0.263953371											
15	4	0.314245657	0.228152583	-0.38138028	-0.295125865											
16	5	0.229130469	0.156842625	-0.089114305	-0.221661372											
17	6	0.556664797	0.249058301	0.220285261	0.189274976											
18	7	0.01380091	0.059051593	0.144879318	0.142830606											
19	8	-0.120703523	-0.211549393	-0.027565747	0.07086993											
20	9	0.74936172	0.512420979	0.379479487	0.230036485											
21	10	-0.357018856	-0.292872035	0.611630003	0.327231217											

Figure 55: *Example: Multivariate tests to compare vector of means and variances*

Assumptions and Notes

- All tests assume multivariate normality.

References

- Anderson, T.W. (2003). *An Introduction to Multivariate Statistical Analysis*, 3rd ed., Wiley.
- Johnson, R.A., & Wichern, D.W. (2007). *Applied Multivariate Statistical Analysis*, 6th ed., Pearson.
- Richardson, J. W., K. Schumann, & P. Feldman. (2004). *Technical Guide for Simetar*. College Station, Texas.

7 Wasserstein Distance

Function: `WassersteinDistance`

Overview

The `WassersteinDistance` is useful for comparing historical data against simulated values to assess distributional similarity. The function calculates the **first-order Wasserstein distance** (also known as the *Earth Mover's Distance*) between two empirical distributions. This metric captures how different two distributions are by measuring the area between their cumulative distribution functions (ECDFs).

Definition

Let $F_n(x)$ and $G_m(x)$ denote the empirical cumulative distribution functions of two samples:

$$X = \{x_1, \ldots, x_n\}, \quad Y = \{y_1, \ldots, y_m\}$$

The first-order Wasserstein distance is given by:

$$W_1(F_n, G_m) = \int_{-\infty}^{\infty} |F_n(x) - G_m(x)| \, dx$$

In this implementation, the integral is approximated numerically via the trapezoidal rule, evaluated over 1000 evenly spaced grid points across the union of support $[\min(X \cup Y), \max(X \cup Y)]$.

Implementation Highlights

- Input ranges are converted to sorted arrays of numeric values.
- ECDFs are estimated incrementally using an optimized *two-pointer* traversal technique.
- The difference in ECDFs is numerically integrated using the trapezoidal rule.
- The result is a single non-negative scalar indicating distributional dissimilarity.

Function Signature

```
=WassersteinDistance(dataRange1, dataRange2)
```

Arguments

- `dataRange1` (required). Data series (row or column vector) representing the first empirical distribution (e.g., historical or baseline series).
- `dataRange2` (required). Data series (row or column vector) representing the second empirical distribution (e.g., simulated series).

Returns A scalar representing the Wasserstein-1 distance.

- A value of 0 indicates that the two empirical distributions are identical.
- Larger values reflect greater differences between the distributions.

Example The example below demonstrates the `WassersteinDistance` function applied to historical data and two simulated series-one generated from a Beta distribution and the other from a Normal distribution. To conserve space, only the first 10 values from each series are shown.

	A	B	C	D	E	F	G	H	I	J	K
1	**LH Sampling with 500 trials.**										
2	Average	0.000523702	0.01285929								
3	Median	-0.962091161	0.04267697								
4	Minimum	-33.34495079	-61.768707								
5	Maximum	38.59350008	62.6363288								
6	Standard Deviation	20.1702581	20.1698766								
7	25th percentile	-17.40125022	-13.573147								
8	75th percentile	16.65559879	13.6142426								
9											
10											
11	**Trials**	**Beta**	**Normal**	**Hisotrical series**		**Beta**	3.310369	=WassersteinDistance($D$12:$D$34, B12:B511)			
12	1	-27.63682122	-8.2851172	3.923572128		**Normal**	3.42936	=WassersteinDistance($D$12:$D$34, C12:C511)			
13	2	-31.76767366	-12.246974	11.84153547							
14	3	25.90787943	-5.2406567	18.45883906							
15	4	10.79315537	20.4538667	3.500000283							
16	5	-22.3481361	-20.869738	4.731441744							
17	6	-3.862952257	-6.6091994	2.678496136							
18	7	9.490286952	-13.195917	-5.667635317							
19	8	-21.07586959	11.4666694	-27.68548632							
20	9	20.78655934	-17.396888	-16.55664008							
21	10	-4.15160533	12.0880677	-18.99007209							

Figure 56: *Example: Wasserstein distance comparison between simulated Beta and Normal distributions and the historical data series*

Assumptions and Scope

- Both inputs are assumed to represent univariate, real-valued empirical samples.
- The function is non-parametric and does not require distributional assumptions.

Use Cases

- Comparing simulated data to historical observations.
- Comparing the individual series simulated with different parametric and non-parametric distributions against the historical observations.
- Quantifying the goodness-of-fit of empirical or parametric distributions.

References

- Villani, C. (2009). *Optimal Transport: Old and New*. Springer.
- Panaretos, V. M., & Zemel, Y. (2020). *An Invitation to Statistics in Wasserstein Space*. Springer.

8 Stochastic Efficiency with Respect to a Function

Stochastic efficiency with respect a function (SERF), developed by Hardaker et al. (2004), calculates certainty equivalents (CEs) for a set of stochastic outcomes using expected utility. It evaluates CE values for 25 evenly spaced relative risk aversion coefficients (RRACs) between a user-defined lower and upper bound. The function supports three utility specifications and returns a dynamic array formatted as a table.

Function Signature:

```
=SERF(dataRange, lowerBound, upperBound, initialWealth, [utilityFunc])
```

Inputs:

- `dataRange`: A 2D range of simulated values (each column corresponds to a scenario).
- `lowerBound`: Minimum RRAC (e.g., 0).
- `upperBound`: Maximum RRAC (e.g., 4).
- `initialWealth`: Wealth level required for utility evaluation (must be positive).
- `utilityFunc` (optional): Utility function type:
  - `1` - Power (CRRA)
  - `2` - Negative Exponential (CARA)
  - `3` - Normalized Log

Output: The function returns a $(25 + 1) \times (n + 1)$ table:

- First column: 25 RRAC values between the lower and upper bounds
- Remaining columns: CE values for each scenario (one per column in the original data)

Procedure Overview:

1. All outcome values are shifted (if needed) to ensure they are strictly positive.
2. For each RRAC level, CE is calculated for each scenario using the selected utility function.
3. The computed CE values are then shifted back and returned in a tabular format.

Certainty Equivalent Computation: Shift and Scaling

Before calculating certainty equivalents (CEs), all simulated outcome values are **shifted by a global constant** to ensure that every value across all scenarios is strictly positive. This is essential for utility functions that are undefined for non-positive values (e.g., Power, Log).

Global Shift: Let y_{ij} represent the i-th simulated value in the j-th scenario. The global minimum across all cells is calculated as:

$$\text{globalMin} = \min_{i,j} \{y_{ij}\}$$

If globalMin ≤ 0, a shift is applied:

$$\tilde{y}_{ij} = y_{ij} + |\text{globalMin}| + 1$$

This guarantees that all shifted values $\tilde{y}_{ij} > 0$.

Stability Scaling: In addition to the global shift, utility-specific CE calculations apply internal rescaling using the column-wise minimum:

$$\min(\tilde{y}) = \min_{i} \{\tilde{y}_i\}$$

This internal scaling is used for numerical stability during the CE calculation and is not related to the global shift applied above.

8.1 Power Utility (CRRA)

The Power utility function - also known as the Constant Relative Risk Aversion (CRRA) utility - is widely used for modeling decreasing marginal utility of income.

Utility Function:

$$U(y) = \begin{cases} \frac{y^{1-r}-1}{1-r}, & r \neq 1 \\ \log(y), & r = 1 \end{cases} \quad \text{where } y > 0$$

Normalization: As mentioned above, each outcome is rescaled to ensure numerical stability:

$$\tilde{y}_{i(\text{scaled})} = \frac{\tilde{y}_i}{\min(\tilde{y})}$$

Expected Utility: The expected utility is calculated from the normalized values:

$$EU = \frac{1}{n} \sum_{i=1}^{n} U(\tilde{y}_{i(\text{scaled})})$$

Inverse Utility (Certainty Equivalent): To recover the CE, the inverse of the utility function is applied:

$$U^{-1}(EU) = \begin{cases} \exp(EU), & r = 1 \\ [1 + (1 - r) \cdot EU]^{\frac{1}{1-r}}, & r \neq 1 \quad (\text{if base} > 0) \end{cases}$$

Final Certainty Equivalent: The CE is then rescaled to its original level:

$$\text{CE} = \min(y) \cdot U^{-1}(EU)$$

8.2 Negative Exponential Utility (CARA)

The CARA utility function is defined as:

$$U(y) = 1 - \exp(-\lambda y), \quad \lambda = \frac{r}{w}$$

where r is the RRAC, w is the initial wealth and λ is the constant absolute risk aversion (CARA) coefficient.

Certainty Equivalent: Given the expected utility $EU = \frac{1}{n} \sum_{i=1}^{n} U(y_i)$, the certainty equivalent is calculated as:

$$\text{CE} = -\frac{1}{\lambda} \log(1 - EU), \quad \text{where } 0 < EU < 1$$

8.3 Normalized Log Utility

The normalized log utility function is an alternative to the standard CRRA and CARA forms (see Rachlin 1992; Scholten and Read 2010, 2014; Bouchouicha and Vieider 2017). It offers a smooth transition around zero risk aversion and avoids undefined values for low or small outcomes by applying normalization. The function is defined as:

Utility Function:

$$U(y) = \begin{cases} \frac{\log(1+ry)}{r}, & r \neq 0 \\ y, & r = 0 \end{cases} \quad \text{where } y > 0$$

This formulation maintains approximate linearity near $r = 0$ and avoids issues with small positive values.

Normalization Procedure: As with the Power utility function, the input values y_i are rescaled to enhance numerical stability:

$$\tilde{y}_{i(\text{scaled})} = \frac{\tilde{y}_i}{\min(\tilde{y})}$$

Expected Utility: The expected utility of the normalized values is calculated as:

$$EU = \frac{1}{n} \sum_{i=1}^{n} U(\tilde{y}_{i(\text{scaled})})$$

Certainty Equivalent: To recover the certainty equivalent (CE) from expected utility, the inverse of the normalized utility is used:

$$U^{-1}(EU) = \begin{cases} EU, & r = 0 \\ \frac{\exp(r \cdot EU) - 1}{r}, & r \neq 0 \end{cases}$$

The final CE is rescaled to the original domain:

$$\text{CE} = \min(y) \cdot U^{-1}(EU) = \min(y) \cdot \begin{cases} EU, & r = 0 \\ \frac{\exp(r \cdot EU) - 1}{r}, & r \neq 0 \end{cases}$$

Illustrative Example: Using `SERF` in Excel

The figure below shows an example of the `SERF` function applied in an Excel worksheet. The formula:

```
=SERF(SimScenario!B11:E510, B1, B2, B3, B4)
```

is used to calculate certainty equivalents across four scenarios using the Power utility function.

Inputs:

- `SimScenario!B11:E510` - simulated data across four scenarios.
- `B1` - Lower RRAC = 0
- `B2` - Upper RRAC = 4
- `B3` - Initial Wealth = 970,615
- `B4` - Utility Function = 1 (Power utility)

Outputs:

- The first column lists the 25 evenly spaced relative risk aversion coefficients (RRACs).
- Each subsequent column displays the certainty equivalent (CE) for one scenario based on the specified utility function and level of risk aversion.
- The CE values decline monotonically with increasing RRAC, reflecting greater aversion to risk.

Note: For Excel 2021 and later versions, including Microsoft 365, the function returns a spilled array automatically. For earlier versions, use `Ctrl + Shift + Enter` *(Windows) or* `Cmd + Shift + Return` *(Mac) to enter the formula as an array.*

	A	B	C	D	E	F	G	H
1	Lower RRAC	0						
2	Upper RRAC	4		=SERF(SimScenario!B11:E510, B1, B2, B3, B4)				
3	Initial Wealth	970,615						
4	Utility Function	1		Relative RAC	Scenario 1	Scenario 2	Scenario 3	Scenario 4
5				0	762870.4	763760.4	761389.2	758867.3
6				0.167	759895.5	760428.5	757806.6	755103.9
7				0.333	756989.6	757166.9	754292.8	751405.5
8				0.5	754153.1	753977	750849.6	747774.2
9				0.667	751386.5	750859.8	747478.3	744211.7
10				0.833	748690.1	747815.9	744180.3	740719.6
11				1	746063.7	744846	740956.3	737298.9
12				1.167	743507.2	741950.1	737807.1	733950.6
13				1.333	741020.1	739128.4	734732.9	730675.3
14				1.5	738601.8	736380.6	731734	727473.4
15				1.667	736251.7	733706.3	728810	724344.9
16				1.833	733968.7	731104.8	725960.8	721289.8
17				2	731752	728575.3	723185.8	718307.8
18				2.167	729600.3	726117	720484.3	715398.4
19				2.333	727512.5	723728.8	717855.3	712560.7
20				2.5	725487.2	721409.3	715297.9	709794.1
21				2.667	723523.1	719157.5	712810.8	707097.4
22				2.833	721618.7	716971.7	710392.9	704469.6
23				3	719772.6	714850.7	708042.6	701909.4
24				3.167	717983.3	712792.8	705758.7	699415.5
25				3.333	716249.2	710796.6	703539.5	696986.5
26				3.5	714568.8	708860.3	701383.5	694621
27				3.667	712940.5	706982.3	699289.1	692317.3
28				3.833	711362.8	705161.1	697254.7	690073.9
29				4	709834.1	703394.9	695278.5	687889.3

Figure 57: *Example output of the `SERF` function applied to four simulated scenarios using Power utility.*

References

- Bouchouicha, R., & Vieider, F. M. (2017). *Accommodating stake effects under prospect theory.* Journal of Risk and Uncertainty, 55, 1-28.
- Hardaker, J. B., Richardson, J. W., Lien, G., & Schumann, K. D. (2004). *Stochastic efficiency analysis with risk aversion bounds: a simplified approach.* Australian Journal of Agricultural and Resource Economics, 48(2), 253-270.
- Rachlin, H. (1992). *Diminishing marginal value as delay discounting.* Journal of the experimental analysis of behavior, 57(3), 407-415.
- Scholten, M., & Read, D. (2010). *The psychology of intertemporal tradeoffs.* Psychological review, 117(3), 925.
- Scholten, M., Read, D., & Sanborn, A. (2014). *Weighing outcomes by time or against time? Evaluation rules in intertemporal choice.* Cognitive science, 38(3), 399-438.

9 Cumulative Prospect Theory (CPT)

Cumulative Prospect Theory (CPT), originally proposed by Tversky and Kahneman (1992), models decision-making under risk by incorporating probability weighting and asymmetric valuation of gains and losses. This Excel function calculates the CPT value and the implied certainty equivalent (CE) for a range of outcomes (gains and losses) based on user-specified weighting and utility parameters.

Function Signature:

```
=CPT(dataRange, gammaGain, gammaLoss, lambda, alpha, [probsRange], [weightFunc], [returnCE], [includeHeaders])
```

Inputs:

- `dataRange`: A 1D or 2D range of numeric outcomes (e.g., simulation results).
- `gammaGain`: Distortion parameter for probability weighting of gains (typically between 0.28 and 1).
- `gammaLoss`: Distortion parameter for losses.
- `lambda`: Loss aversion coefficient ($\lambda \geq 1$).
- `alpha`: Power utility curvature parameter ($0 < \alpha \leq 1$).
- `probsRange` (optional): A range of explicit probabilities associated with each outcome (must sum to 1).
- `weightFunc` (optional): Weighting function to use:
  - `"prelec"` (Prelec (1998) weighting function, default)
  - `"tk"` (Tversky-Kahneman (1992) weighting function)
- `returnCE` (optional): If `TRUE` (default), also returns the certainty equivalent.
- `includeHeaders` (optional): If `TRUE` (default), outputs include row labels.

Output: The function returns a 1×2 or 2×2 table:

- CPT value (weighted utility)

- Certainty equivalent (inverse utility of CPT value)
- Optional: Row labels if `includeHeaders` is `TRUE`

Procedure Overview:

1. Outcomes are separated into gains and losses.
2. Cumulative probabilities are assigned by rank.
3. Decision weights are calculated using the selected weighting function.
4. Power utility is applied to each outcome.
5. The CPT value is calculated as the weighted sum of utilities.
6. The certainty equivalent is recovered via the inverse of the utility function.

Power Utility with Loss Aversion

CPT uses a power utility function that treats gains and losses asymmetrically via a loss aversion factor λ:

$$U(x) = \begin{cases} x^{\alpha}, & x \geq 0 \\ -\lambda(-x)^{\alpha}, & x < 0 \end{cases} \quad \text{for } 0 < \alpha \leq 1,\ \lambda \geq 1$$

Inverse Utility (Certainty Equivalent): To compute the certainty equivalent, the inverse of the utility function is applied:

$$U^{-1}(v) = \begin{cases} v^{1/\alpha}, & v \geq 0 \\ -\left(\frac{-v}{\lambda}\right)^{1/\alpha}, & v < 0 \end{cases}$$

Probability Weighting

CPT transforms cumulative probabilities using a nonlinear weighting function. Two options are supported:

1. Prelec (1998) Function:

$$w(p) = \exp\left(-(-\ln p)^{\gamma}\right), \quad \text{for } 0 < p \leq 1,\ \gamma \in (0, 1]$$

2. Tversky-Kahneman (1992) Function:

$$w(p) = \frac{p^{\gamma}}{[p^{\gamma} + (1-p)^{\gamma}]^{1/\gamma}}$$

Each outcome receives a decision weight equal to the difference in weighted cumulative probabilities:

$$\pi_i = w(p_i^{\text{high}}) - w(p_i^{\text{low}})$$

Final CPT Value: The final CPT value is the weighted sum:

$$\text{CPT} = \sum_i \pi_i \cdot U(x_i)$$

Certainty Equivalent: The CE is the outcome value that yields the same utility as the CPT value:

$$\text{CE} = U^{-1}(\text{CPT})$$

Illustrative Example

The example below illustrates how the `CPT` function is used in a worksheet:

- The user selects a column of outcome values (e.g., a set of simulated returns).
- Four required parameters - `gammaGain`, `gammaLoss`, `lambda`, and `alpha` - are supplied directly in the formula, typically as cell references (e.g., `D2`, `D3`, `D4`, `D5`).
- Optional arguments include: a range of probabilities, the probability weighting function (`"prelec"` or `"tk"`), a flag to include the certainty equivalent, and a flag to include row headers.
- The function returns either the CPT value alone or both the CPT value and certainty equivalent, depending on the `returnCE` argument.

```
=CPT(A2:A5, D2, D3, D4, D5)
```

Inputs:

- `A2:A5` – Outcome values (gains and losses)
- `D2` – Probability distortion parameter for gains
- `D3` – Probability distortion parameter for losses
- `D4` – Loss aversion parameter
- `D5` – Curvature parameter
- The remaining arguments are left unspecified, in which case the function uses their default values.

Outputs:

- First row: `"CPT value"` and its result
- Second row: `"Certainty equivalent"` and its value

	A	B	C	D	E	F	G
1	Outcomes						
2	100		Gamma gain	0.61			
3	-50		Gamma loss	0.69			
4	75		Lambda	2.25			
5	-30		Alpha	0.88			
6							
7							
8			CPT value	-4.043776	=cpt(A2:A5, D2, D3, D4, D5)		
9			Certainty equivalent	-1.946809			

Figure 58: *Example output of the `CPT` function using Prelec weighting and power utility.*

References

- Tversky, A., & Kahneman, D. (1992). *Advances in prospect theory: Cumulative representation of uncertainty.* Journal of Risk and Uncertainty, 5(4), 297-323.
- Prelec, D. (1998). *The probability weighting function.* Econometrica, 66(3), 497-527.