Problem Set 04: Cumulative Distribution Plots & User-Defined Functions

New Learning Objectives under Evaluation

10.00 Create and interpret cumulative distribution plots

Learning Objective	Evidence
10.01 Compute the relative fractional values given the bin intervals	Call the `histogramRight` function (with the correct input arguments) to generate the histogram properties Determine the frequencies for each bin in a right-bin inclusive histogram Determine the total number of data points accounted for in the overall histogram Calculate the fractional values by dividing the frequency in each bin by the total number of data points accounted for in the overall histogram
10.02 Compute cumulative fractional values given the bin intervals	Correct syntax for cumsum function Perform the cumulative sum to get vector of cumulative fractional values Start the cumulative sum vector at 0
10.03 Create a cumulative distribution plot using the companion histogram's bin right edges	Correct syntax for the plot command: plot(x, y, 'line/marker formatting') Independent variable (x) is the bin edge values from a right-bin inclusive histogram Dependent variable (y) is the cumulative fractional values corresponding to the right-bin inclusive histogram Correct use of data markers and lines: data markers (for the bin edges) with an overlaid line (for the model)
10.04 Format a cumulative distribution plot for technical presentation	Correct syntax for title Correct syntax for xlabel Correct syntax for ylabel Descriptive title that references the problem context and x-variable data Clear x-axis label with units Clear y-axis label that is cumulative fractional value y-axis scale range of 0 to 1 x-axis scales that match each other, when using subplots to compare data Color and marker/line style(s) that are as specified or distinctive (when multiple data sets) Proper formatting of a legend, when multiple data sets and/or models Gridlines

Problem Set 04: Cumulative Distribution Plots & User-Defined Functions

New Learning Objectives under Evaluation

Learning Objective	Evidence
10.05 Determine the likelihood of event occurrences using a cumulative distribution plot	Determine the likelihood of an occurrence of a value: • less than specified criteria • greater than specified criteria • between specified criteria reading the fractional value for given data point Clear explanation of how the likelihood is determined
10.06 Estimate and/or describe the process for determining the characteristics of the underlying data set from a cumulative distribution plot	Estimate the median of the data by reading the CDP at 0.5 cumulative fractional value (within 2% of solution answer) Estimate the range of the data by reading the CDP at 0 and 1 cumulative fractional values (within 2% of solution answer) Clear description of a process for determining the median Clear description of a process for determining the range
10.07 Determine the data distribution type from the shape of a cumulative distribution plot	Identify the shape of the distribution (uniform, unimodal, bimodal, normal, etc) Justify shape identification Identify the skew of the distribution (positive, negative, undefined, etc) Justify skew identification
10.08 Draw inferences from the analysis of data with evidence from a cumulative distribution	For a given data set and a problem context, appropriately use a cumulative distribution as described in 10.05 – 10.07 to draw conclusions.

Problem Set 04: Cumulative Distribution Plots & User-Defined Functions

New Learning Objectives under Evaluation

11.00 Create and execute a user-defined function

Learning Objective	Evidence
11.01 Describe at least two reasons why MATLAB user-defined functions as opposed to scripts are used	Recognition that UDFs enable one to create an easily re-usable piece of code Recognition that UDFs can be shared with others without them having to know what variables were used by the author Recognition that UDFs enable a larger program to broken into smaller parts that can be more easily tested & debugged Recognition that UDFs allow team members to work on separate parts of a larger program with less coordination
11.02 Describe three ways a user-defined function is different from a script	Recognition that the first line of a UDF is the function definition line; the first line of a script can be any executable line of code Recognition that the variables created in a UDF are not available in the Workspace; all variables created in script are available in the Workspace Recognition that a UDF must be called from the command line or from within another function or script; the green run button will not work for a UDF that has input arguments
11.03 Create a user-defined function that adheres to programming standards	Help lines contain input and output argument definitions, with units as appropriate Help lines contain concise description of the program Help lines show the call to the function Complete programmer and contributor information in the header (names and emails) Complete problem details including assignment number, problem number Code items are in the correct section (e.g. Initialization, Calculations, …) Computed values are assigned to variables Code blocks have explanatory comments Variables have commented definitions and units Minimal use of hardcoding

Problem Set 04: Cumulative Distribution Plots & User-Defined Functions

New Learning Objectives under Evaluation

Learning Objective	Evidence
11.04 Construct an appropriate function definition line	Correct syntax for a function: • `function [output1,…,outputN] = function_name(input1,…,inputM)` • Function starts with the keyword `function` • Order is output arguments, equal sign, function name, input arguments • Functions with no inputs have no input list; use of ( ) is optional • Functions with no output arguments have no output list and no equal sign • Multiple output arguments are listed inside square brackets, separated by spaces or commas • Multiple input arguments are listed inside parentheses and are comma-separated Function definition line is the first line in the function file (above help lines) Function file name matches the function name in the definition line Input arguments must meet the problem specifications (with no extraneous input arguments) or be appropriate for the purpose of the function Output arguments must meet the problem specifications or be appropriate for the purpose of the function Output arguments must be assigned within the function code
11.05 Match the variables names used in the function definition line to those used in the function code	All input arguments are used in the code All input arguments necessary to perform computations are provided in the function definition Input arguments are not overwritten (e.g. by hardcoded values) before being used in calculations All output arguments are appropriately assigned in the function code

Problem Set 04: Cumulative Distribution Plots & User-Defined Functions

New Learning Objectives under Evaluation

Learning Objective	Evidence
11.06 Execute a user-defined function	Correct syntax to call a function: • `[output1,…,outputN] = function_name(input1,…,inputM)` • Call does not contain keyword `function` • Order is output arguments, equal sign, function name, input arguments, with output arguments and equal sign being optional for a no-output function • Functions with no inputs have no input list; use of ( ) is optional • Functions with no output arguments have no output list and no equal sign • Multiple output arguments are listed inside square brackets, separated by spaces or commas • Multiple input arguments are listed inside parentheses and are comma-separated Calls the correct function filename Number of input arguments matches the number required by the function Input argument list corresponds to the function’s expected inputs Number of output argument(s) matches the number required by the function Output argument list corresponds to the function’s expected outputs
11.07 Create test cases to evaluate a user-defined function	Running the UDF with a variety of reasonable values for each input argument to ensure no computation or execution errors occur Running the UDF with both scalar and array input arguments to ensure no errors occur
11.08 Convert a script to a user-defined function	First line of code is a function definition line Replacement of script header with function header Removal of hardcoded variable assignments for all variables in the input argument list