Test Report on the Module statistics_lib.distribution_classes

Conventions

Each test is defined following the same format. Each test receives a unique test identifier and a reference to the ID(s) of the requirements it covers (if applicable). The goal of the test is described to clarify what is to be tested. The test steps are described in brief but clear instructions. For each test it is defined what the expected results are for the test to pass. Finally, the test result is given, this can be only pass or fail.

The test format is as follows:

Test Identifier: TEST-[I/A/D/T]-XYZ

Requirement ID(s): REQ-uvw-xyz

Verification method: I/A/D/T

Test goal: Description of what is to be tested

Expected result: What test result is expected for the test to pass

Test steps: Step by step instructions on how to perform the test

Test result: PASS/FAIL

The test ID starts with the fixed prefix 'TEST'. The prefix is followed by a single letter, which defines the test type / verification method. The last part of the ID is a 3-digits *hexadecimal* number (0..9|A..F), with the first digit identifing the module, the second digit identifing a class / function, and the last digit - the test ordering number for this object. E.g. 'TEST-T-112'. Each test type has its own counter, thus 'TEST-T-112' and 'TEST-A-112' tests are different entities, but they refer to the same object (class or function) within the same module.

The verification method for a requirement is given by a single letter according to the table below:

Term	**Definition**
Inspection (I)	Control or visual verification
Analysis (A)	Verification based upon analytical evidences
Test (T)	Verification of quantitative characteristics with quantitative measurement
Demonstration (D)	Verification of operational characteristics without quantitative measurement

Tests definition (Analysis)

Test Identifier: TEST-A-600

Requirement ID(s): REQ-FUN-600

Verification method: A

Test goal: All required functionality is implemented and performs correctly.

Expected result: All required classes implenting continuous random distributions are present and function as expected, i.e. all TEST-T-6xy tests defined in this document are passed.

Test steps: Analyze the source code of the module inverse_distrbutions as well as of the unit-test module /Tests/UT006_inverse_distributions. Execute the mentioned unit-test module.

Test result: PASS

Tests definition (Test)

Test Identifier: TEST-T-600

Requirement ID(s): REQ-FUN-601

Verification method: T

Test goal: Instantiation of a class - assignment of parameters.

Expected result: A class being tested can be instantiated, provided that the required number of the required data type and value's range attributes are passed into the initialization method. The passed arguments are properly assigned to the parameters of the distribution.

Test steps: Instantiate the class being tested with random but proper values of the parameters. Check that no exception is raised. Check that the parameters of the distribution are properly assiged, and the statistical properties of the distribution are as expected for the given parametes. Repeat the process several time. This test should be performed with all implemented distribution classes.

Test result: PASS

Test Identifier: TEST-T-601

Requirement ID(s): REQ-FUN-601

Verification method: T

Test goal: Instantiation of the class - all required attributes are present.

Expected result: All required attributes: methods and properties - are present in an instance of the class regardless of the passed parameters of the distribution, as long as they are of the proper values.

Test steps: Instantiate the class being tested with random but proper values of the parameters. Check that all requried attributes are present. Repeat the process several time. This test should be performed with all implemented distribution classes.

Test result: PASS

Test Identifier: TEST-T-602

Requirement ID(s): REQ-FUN-602

Verification method: T

Test goal: Variability of the distribution parameters

Expected result: The parameters of the distribution can be modified at any time via the respective setter properties. The statistical properties of the distribution change accordingly.

Test steps: Instantiate the class being tested with random but proper values of the parameters. Assign new proper value to the all parameters of the distribution. Check that the parameters are changed, and the statistical properties are calculated as expected for the set parameters. Repeat several times with different (random) values of the parameters. This test should be performed with all implemented distribution classes.

Test result: PASS

Test Identifier: TEST-T-603

Requirement ID(s): REQ-FUN-603

Verification method: T

Test goal: Read-only access to the statistical properties of the distribution

Expected result: The class' properties representing the statistical properties of the distribution, can be accessed for reading, but they cannot be deleted or modified (assigned to) on an instance of that class.

Test steps: Instantiate the class being tested with random but proper values of the parameters. Attemt to modify and to delete each of the properties, unless it is also a parameter of the distribution (e.g. mean and sigma for Gaussian). Check that AttributeError or its sub-class is raised each time. This test should be performed with all implemented distribution classes.

Test result: PASS

Test Identifier: TEST-T-604

Requirement ID(s): REQ-FUN-604

Verification method: T

Test goal: Proper implementation of pdf() method.

Expected result: The method pdf() accepts any real number as its single argument and returns a non-negative real number, which is:

- a positive value of the probability density function at this value of the random variable as long as it is within the range of the accepted values (Min to Max) for the distibution
- zero (0) for the values outside the accepted range

Test steps: Instantiate the class being tested with random but proper values of the parameters. Call the method with different random values of its argument. Check that the returned results is as expected (see

DE004 document). The returned values should be checked either against 'manually calculated' ones. Change the parameters of the distribution and repeat the test.

This test should be performed with all implemented distribution classes.

Test result: PASS

Test Identifier: TEST-T-605

Requirement ID(s): REQ-FUN-605

Verification method: T

Test goal: Proper implementation of cdf() method.

Expected result: The method cdf() accepts any real number as its single argument and returns a non-negative real number, which is the value of the cummulative density function, which is a monotonically growing from 0 to 1 function for the values within the range of the accepted values (Min to Max) for the distibution, and strictly 0 for values <= Min.

Test steps: Instantiate the class being tested with random but proper values of the parameters. Call the method with different random values of its argument. Check that the returned results is as expected (see DE004 document). The returned values should be checked either against 'manually calculated' ones. Change the parameters of the distributio and repeat the test.

This test should be performed with all implemented distribution classes.

Test result: PASS

Test Identifier: TEST-T-606

Requirement ID(s): REQ-FUN-606

Verification method: T

Test goal: Proper implementation of qf() and getQuantile() methods.

Expected result: The method qf() accepts any floating point number p in the range (0, 1) and returns a real number x, for which the following relations are valid: Min < x < Max (returned value is within the range of the random variable) and cdf(x) = p.

The method getQuantile(k, m) accepts two positive integer numbers 0 < k < m and returns the value equal to qf(k/m).

Test steps: Instantiate the class being tested with random but proper values of the parameters. Call the qf() method with different random values of its argument. Check that the returned results is as expected (see DE004 document). The returned values should be checked either against 'manually calculated' ones (for the simple formulas) or using the 'inverse' relation between QF and the already tested CDF.

Call the method getQuantile(k, m) with different random values of the arguments and compare the returned value with the result of the call qf(k/m).

Change the parameters of the distributio and repeat the test.

This test should be performed with all implemented distribution classes.

Test result: PASS

Test Identifier: TEST-T-607

Requirement ID(s): REQ-AWM-600

Verification method: T

Test goal: Rejection of the arguments of improper data types.

Expected result: TypeError or its sub-class exception is raised upon passing improper data type argument(s) to:

- Initialization method - not a real number for all classes, not an integer or floating point
- pdf() and cdf() methods - not a real number for all classes
- qf() - not a floating point number for all classes
- getQuantile() - not an integer (any of the two argument) - for all classes
- getHistogram() - not a real number for the first two arguments, not an integer for the third argument - all classes

Concerning the assigment to the setter properties (parameters of the distribution) the same rules are applied as for the instantiation of the class.

Test steps: Try to instantiate the class being tested with different improper data types of each of the parameters in turn. Check that the expected exception is raised each time. Repeat with the different improper data types.

Instantiate the class being tested with random but proper values of the parameters. Try to assign different improper data types values to the setter properties. Check that the expected exception is raised each time. Repeat with the different improper data types. Try to pass different improper data type argument(s) of the listed methods. Check that the expected exception is raised each time. Repeat with the different improper data types. This test should be performed with all implemented distribution classes.

Test result: PASS

Test Identifier: TEST-T-608

Requirement ID(s): REQ-AWM-601

Verification method: T

Test goal: Rejection of the arguments of proper data types but improper values.

Expected result: TypeError or its sub-class exception is raised upon passing proper data type argument(s) of inapproptiate values to:

- Instantiation of the classes

    - Levy and Cauchy distributions - scale parameter is not positive, i.e. <= 0
    - Other distributions - any of the parameters is not positive, i.e. <= 0
- qf() - the argument is not within (0, 1) range
- getQuantile(k, m) - either of the arguments is <= 0, or k >= m
- getHistogram(min, max, NBins) - min >= max , or NBins is <= 1

Concerning the assigment to the setter properties (parameters of the distribution) the same rules are applied as for the instantiation of the class.

Test steps: Try to instantiate the class being tested with different improper values (but proper data type) of each of the parameters in turn. Check that the expected exception is raised each time. Repeat with the different improper values.

Instantiate the class being tested with random but proper values of the parameters. Try to assign different improper values (but proper data type) to the setter properties. Check that the expected exception is raised each time. Repeat with the different improper values. Try to pass different improper values (but proper data type) argument(s) of the listed methods. Check that the expected exception is raised each time. Repeat with the different improper values. This test should be performed with all implemented distribution classes.

Test result: PASS

Tests definition (Demonstration)

Test Identifier: TEST-D-600

Requirement ID(s): REQ-FUN-607, REQ-FUN-608

Verification method: D

Test goal: Check implementation of the methods *random*() and *getHistogram*()

Expected result: A historgam of the distribution itself (method *getHistogram*()) is similar (almost equal) in shape to the step-wise approximation of the shape of PDF of the distribution. A histogram of a large sample of random numbers generated by the corresponding distribution class resembles the both previous shapes.

Test steps: Perform the following procedure with all distributions to be tested.

- Instantiate the respective class with arbitrary chosen parameters
- Generate 10000 random number with this instance. Because of the 'fat tails', store in a sequence only those values, which are in the range [1.5 * Q1 - 0.5 * Q3, 1.5 * Q3 - 0.5 * Q1], where Q1 and Q3 are the first and the third quartiles of the model distribution; and discard the other values
- Instantiate 1D statistics class (module *data_classes*) with that sequence, which can be less than the intented 10000 elements
- Generate a histogram of the sample with the default 20 bins
- Normalize the values (frequences) in each bin by the intented length of the sequence 10000
- Determine the minimum and maximum central values of the bins and bin width
- Generate the histrogram of the entire distribution (tested class) using the determined minimum and maximum central values and the number of bins - 20

- For each bin with the cental value x_k (and the same width S) manually calculate the value $S * [pdf(x_k - s/2) + pdf(x_k) + pdf(x_k + s/2)]/3$
- Print out the results of the calculation (3 histograms) on the screen as 6-columns table and compare

This test should be performed with all implemented distribution classes.

Test result: PASS

Traceability

For traceability the relation between tests and requirements is summarized in the table below:

Requirement ID	Covered in test(s)	Verified [YES/NO])
REQ-FUN-600	TEST-A-600	YES
REQ-FUN-601	TEST-T-600, TEST-T-601	YES
REQ-FUN-602	TEST-T-602	YES
REQ-FUN-603	TEST-T-603	YES
REQ-FUN-604	TEST-T-604	YES
REQ-FUN-605	TEST-T-605	YES
REQ-FUN-606	TEST-T-606	YES
REQ-FUN-607	TEST-D-600	YES
REQ-FUN-608	TEST-D-600	YES
REQ-AWM-600	TEST-T-607	YES
REQ-AWM-601	TEST-T-608	YES

Software ready for production [YES/NO]	Rationale
YES	All tests are passed