Test Report on the Module statistics_lib.special_functions

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-500

Requirement ID(s): REQ-FUN-500

Verification method: A

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

Expected result: The required special functions are present and perform as expected, i.e. all TEST-T-5xy tests defined in this document are passed.

Test steps: Analyze the source code of the module special_functions as well as of the unit-test module /Tests/UT005_special_functions. Execute the mentioned unit-test module.

Test result: PASS

Tests definition (Test)

Test Identifier: TEST-T-500

Requirement ID(s): REQ-AWM-500

Verification method: T

Test goal: TypeError sub-class exception is raised in response to the improper data type of an argument.

Expected result: The said exception is raised with

- Any data type of argument(s) except **int** for *permutation*() and *combinations*() functions
- Any data type of argument(s) except **int** or **float** for all other functions

Test steps: This test should be implemented as a method *test_TypeError* of all unit-test classes testing a specific function. The test is simple - try to call the function being tested and pass an inappropriate data type value as the value one of the arguments, whereas the other arguments are of the acceptable data type and value. Check that the expected exception is raised. Repeat this process with a number of inappropriate data type for the same argument. Also apply the same checks for each of the other arguments. Also try to pass all arguments of an improper data type simultaneously.

Test result: PASS

Test Identifier: TEST-T-501

Requirement ID(s): REQ-AWM-501

Verification method: T

Test goal: ValueError sub-class exception is raised in response to the proper data type of an argument with unacceptable value.

Expected result: The said exception is raised with

- Any of the arguments is < 0 for for *permutation*() and *combinations*() functions, OR
- k > n for *permutation*() and *combinations*() functions
- The argument of *inv_erf*() function is >= 1 or <= -1
- The second or / and the third arguments (x and y) of the incomplete beta functions is <= 0, OR
- The first argument (z) of the the incomplete beta functions is not in the range [0, 1]

- The first argument (x) of the incomplete gamma functions is <= 0, OR
- The second argument (y) of the incomplete gamma functions is < 0 (not logarithmic), OR
- The second argument (y) of the logarithmic incomplete gamma functions is <= 0

Test steps: This test should be implemented as a method *test_ValueError* of all unit-test classes testing a specific function. The test is simple - try to call the function being tested and pass an inappropriate value of the proper data type as the value one of the arguments, whereas the other arguments are of the acceptable data type and value. Check that the expected exception is raised. Repeat this process with a number of inappropriate vlaues for the same argument. Also apply the same checks for each of the other arguments. Also try to pass all arguments of an improper value simultaneously.

Test result: PASS

Test Identifier: TEST-T-510

Requirement ID(s): REQ-FUN-510

Verification method: T

Test goal: Implementation of the k-permutation function.

Expected result: The function *permutation*(n, k) with proper arguments 0 <= k: **int** <= n: **int** returns the value of n! / (n-k)!, which is a positive integer.

Test steps: This test is implemented as method *test_Ok*() of the unit-test class **Test_permutation**. Call the function being tested with a number of combination of values of the arguments, for which the expected result is known (manually calculated in advance). Compare the returned values with the expected ones. Also call the same function with a number of random values of arguments within the acceptable range, and check that the function returns a positive **int** number, which value is the same as calculated using *math.factorial*() function calls.

Test result: PASS

Test Identifier: TEST-T-520

Requirement ID(s): REQ-FUN-520

Verification method: T

Test goal: Implementation of the combinations function.

Expected result: The function *combination*(n, k) with proper arguments 0 <= k: **int** <= n: **int** returns the value of n! / [k! (n-k)!], which is a positive integer.

Test steps: This test is implemented as method *test_Ok*() of the unit-test class **Test_combination**. Call the function being tested with a number of combination of values of the arguments, for which the expected result is known (manually calculated in advance). Compare the returned values with the expected ones. Also call the same function with a number of random values of arguments within the acceptable range, and check that the function returns a positive **int** number, which value is the same as calculated using *math.factorial*() function calls.

Test result: PASS

Test Identifier: TEST-T-530

Requirement ID(s): REQ-FUN-530

Verification method: T

Test goal: Implementation of inverse error function.

Expected result: The function *inv_erf*(p) accepts a floating point number in the range (0, 1) and returns a floating point value $x \in (-\infin, +\infin)$ such that erf(x)= p, which is equivalent to $x = \mathtt{erf}^{-1}(p)$.

Test steps: This test is implemented as method *test_Ok*() of the unit-test class **Test_inv_erf**. Call the function being tested with a number of combination of values of the arguments, for which the expected result is known (either using published tables or on-line calculators, the source should be indicated in the unit-test method description). Compare the returned values with the expected ones. Also call the same function with a number of random values of arguments within the acceptable range, and check that the function returns a **float** number, and that the tested function is indeed inverse to the *math.efr*() implementation with the default (7 places) floating point precision.

Test result: PASS

Test Identifier: TEST-T-540

Requirement ID(s): REQ-FUN-540

Verification method: T

Test goal: Implementation of the beta functions.

Expected result: The functions *beta*(x, y) and *log_beta*(x, y) accept two positive integer or floating point number arguments and return the values of the special function *beta* $B(x,y) = \Gamma(x)\Gamma(y)/\Gamma(x+y)$ and its natural logarithm ln(B(x, y)) respectively.

Test steps: This test is implemented as method *test_Ok*() of the unit-test classes **Test_beta** and **Test_log_beta**. Call the function being tested with a number of combination of values of the arguments, for which the expected result is known (either using published tables or on-line calculators, the source should be indicated in the unit-test method description). Compare the returned values with the expected ones. Also call the same function with a number of random values of arguments within the acceptable range, and check that the function returns a **float** number within the expected range.

Test result: PASS

Test Identifier: TEST-T-550

Requirement ID(s): REQ-FUN-550

Verification method: T

Test goal: Implementation of the incomplete beta functions.

Expected result: The functions *beta_incomplete*(z, x, y), *beta_incomplete_reg*(z, x, y) and *log_beta_incomplete*(z, x, y) accept the first argument being a real number in the range (0, 1] and two positive integer or floating point number arguments x and y and return the values of the special functions *incomplete beta B(z; x, y)*, *regularized incomplete beta* $I_z(x,y) = B(z;x,y)/B(x,y)$ and the natural logarithm ln(B(z; x, y)) respectively. In addition, the special case z=0 should be supported as *beta_incomplete*(0, x, y) = 0 and *beta_incomplete_reg*(0, x, y) = 0.

Test steps: This test is implemented as method *test_Ok*() of the unit-test classes **Test_beta_incomplete**, **Test_log_beta_incomplete** and **Test_beta_incomplete_reg**. Call the function being tested with a number of combination of values of the arguments, for which the expected result is known (either using published tables or on-line calculators, the source should be indicated in the unit-test method description). Compare the returned values with the expected ones. Also call the same function with a number of random values of arguments within the acceptable range, and check that the function returns a **float** number within the expected range.

Test result: PASS

Test Identifier: TEST-T-560

Requirement ID(s): REQ-FUN-560

Verification method: T

Test goal: Implementation of the incomplete gamma functions.

Expected result: The functions *lower_gamma*(x, y), *lower_gamma_reg*(x, y) and *log_lower_gamma*(x, y) accept two positive integer or floating point number arguments and return the values of the special functions *lower incomplete gamma* $\gamma(x,y)$, *regularized lower incomplete gamma* $P(x,y) = \gamma(x,y)/\Gamma(x)$ and the natural logarithm $\mathtt{ln}(\gamma(x,y))$ respectively. In addition, the special case y=0 should be supported as *lower_gamma*(x, 0) = 0 and *lower_gamma_reg*(x, 0) = 0.

The functions *upper_gamma*(x, y), *upper_gamma_reg*(x, y) and *log_upper_gamma*(x, y) accept two positive integer or floating point number arguments and return the values of the special functions *upper incomplete gamma* $\Gamma(x,y)$, *regularized upper incomplete gamma* $Q(x,y) = \Gamma(x,y)/\Gamma(x)$ and the natural logarithm $\mathtt{ln}(\Gamma(x,y))$ respectively. In addition, the special case y=0 should be supported as *upper_gamma*(x, 0) = *math.gamma*(x) and *upper_gamma_reg*(x, 0) = 1.

Test steps: This test is implemented as method *test_Ok*() of the unit-test classes **Test_lower_gamma**, **Test_log_lower_gamma**, **Test_lower_gamma_reg**, **Test_upper_gamma**, **Test_log_upper_gamma** and **Test_upper_gamma_reg**. Call the function being tested with a number of combination of values of the arguments, for which the expected result is known (either using published tables or on-line calculators, the source should be indicated in the unit-test method description). Compare the returned values with the expected ones. Also call the same function with a number of random values of arguments within the acceptable range, and check that the function returns a **float** number within the expected range.

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-500	TEST-A-500	YES
REQ-FUN-510	TEST-T-510	YES
REQ-FUN-520	TEST-T-520	YES
REQ-FUN-530	TEST-T-530	YES
REQ-FUN-540	TEST-T-540	YES
REQ-FUN-550	TEST-T-550	YES
REQ-FUN-560	TEST-T-560	YES
REQ-AWM-500	TEST-T-500	YES
REQ-AWM-501	TEST-T-501	YES

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