Test Report on the Module statistics_lib.base_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-100

Requirement ID(s): REQ-FUN-100

Verification method: A

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

Expected result: The following functions are present:

- Calculation of the mean
- Calculation of the sample and population variance
- Calculation of the sample and population standard deviation
- Calculation of the standard error of the mean
- Calculation of the mean squared uncertainty of the points in the sample
- Calculation of the total error / uncertainty of the mean
- Calculation of the sample and population skewness
- Calculation of the sample and population excess kurtosis
- Calculation of the generic Nth central / non-central, normalized / not normalized momentum
- Calculation of covariance of two samples of the same length
- Calculation of the Pearson's r correlation coefficient for two samples of the same length
- Calculation of the generic Nth-Mth central / non-central, normalized / not normalized co-momentum

All these function pass TEST-T-100, TEST-T-101 and TEST-T-102.

Test steps: Analyze the source code of the module base_functions as well as of the unit-test module /Tests/UT001_base_functions. Execute the mentioned unit-test module.

Test result: PASS

Tests definition (Test)

Test Identifier: TEST-T-100

Requirement ID(s): REQ-FUN-101

Verification method: T

Test goal: Check the acceptable input data types, the type of the returned value and the correctness of the calculation of the returned value.

Expected result: All 1D statistics functions accept any non-empty sequence of either integer numbers, or floating point numbers, or instances of a class compatible with **phyqus_lib.base_classes.MeasuredValue**, i.e. having data attributes *Value* and *SE*, or any mixture of these three data types. All 2D statistics functions accept two such sequences of equal length. All functions return either a floating point or an integer value.

In all functions, except for the mean squared uncertanity and total error, any instance of 'measurement with uncertainty' is treated as a real number equal to its *Value* attribute, i.e. (x_i, z_i) -> x_i. In the function calculating the mean squared uncertanity each real number is considered to be an instance of 'measurement with uncertainty', where uncertainty is zero, i.e. x_i -> (x_i, 0). The total error calculation function uses the both described data type casting mechanisms to find the respective parts of the total error / uncertainty.

The calculatins performed by the functions are correct, as is verified by comparison with the results returned by the functions from the Stand Library module *statistics*:

- Arithmetic mean (average) -> *mean*()
- Variance -> *pvariance*() - without Bessel correction (population), and *variance*() - with Bessel correction (sample)
- Standard deviation -> *pstdev*() - without Bessel correction (population), and *stdev*() - with Bessel correction (sample)
- Covariance -> *covariance*() (Python v3.10+, otherwise - use manual calculations)
- Pearson correlation -> *correlation*() (Python v3.10+, otherwise - use manual calculations)

Other functions can be verified using varaible(s) substitution rule $\left(\frac{x-a_x}{b_x}\right)^{N>0}\left(\frac{y-a_y}{b_y}\right)^{M\geq0} \rightarrow z$ and calculating the arithmetic mean of the substitution variable.

Test steps: Preparation

- Generate a random length lists of:
    - All integers (positive and negative are allowed) as *AllInt*
    - All floating point numbers (positive and negative are allowed) as *AllFloat*
    - A random mixture of integers and floating point numbers (positive and negative are allowed) as *Mixed*
- Generate a list of non-negative floating point numbers of the length equal the largest length of the lists above as *Errors*
- Construct the lists of instances of **phyqus_lib.base_classes.MeasuredValue** using:
    - Values from *AllInt* as the 'means' and the same index (position) values from *Errors* as the ucertainties
    - Values from *AllFloat* as the 'means' and the same index (position) values from *Errors* as the ucertainties
    - Values from *Mixed* as the 'means' and the same index (position) values from *Errors* as the ucertainties
- Construct a mixed list of real numbers and **phyqus_lib.base_classes.MeasuredValue** instances from *Mixed* list, where some elements from the original list are randomly converted into 'measurement with uncertainty' using the respective (same index) element from *Errors* as the uncertainty

Pass each of the created lists into the function being tested. Check that no exception is raised. Check that the returned value is a real number. Check that the returned value is (almost) equal to the expected result using the Standard Library and the respective 'real numbers' lists.

Repeat the test coverting the same lists into tuples before passing into the function being tested.

In case of of the 2D statistic functions 2 sets of the lists of these data types should be generated. Before passing into a function, the longest of the two sequence argumnets must be truncated to match the length of the shortest one.

Test result: PASS

Test Identifier: TEST-T-101

Requirement ID(s): REQ-AWM-100

Verification method: T

Test goal: A sub-class of **TypeError** exception is raised in response to the improper type of the input data.

Expected result: Such an exception is raised if, at least, one of the data set arguments is not a flat sequence of real numbers and / or measurements with uncertainties, or any of the moment powers is not an integer.

Test steps: Try to call the funcion being tested with an appropriate data type argument. Check that the expected exception is raised.

Test result: PASS

Test Identifier: TEST-T-102

Requirement ID(s): REQ-AWM-101

Verification method: T

Test goal:A sub-class of **ValueError** exception is raised in response to the improper values of the proper type of the input data.

Expected result: Such an exception is raised in the following cases:

- At least, one of the sequence data set arguments is an empty sequence
- Inequal length of the sequence arguments of a 2D statistics function (in general)
- Integer but zero or negative momentum power
- Too short sequence for the Bessel corrected functions:
    - N = 1 for variance and standard deviation
    - N < 3 for skewness
    - N < 4 for kurtosis

Test steps: Try to call the funcion being tested with an appropriate argument (see above). Check that the expected exception is raised.

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-100	TEST-A-100	YES
REQ-FUN-101	TEST-T-100	YES
REQ-AWM-100	TEST-T-101	YES
REQ-AWM-101	TEST-T-102	YES

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