# Module statistics_lib.base_functions Reference

## Scope

This document describes the intended usage, design and implementation of the functionality implemented in the module **ordered_functions** of the library **statistics_lib**. The API reference is also provided.

This module provides functions for calculation of:

- 1D statstics
  - The minimal value within the sample - *GetMix*()
  - The maximal value within the sample - *GetMax*()
  - Median value of the sample - *GetMedian*()
  - The first quartile of the sample - *GetFirstQuartile*()
  - The third quartile of the sample - *GetThirdQuartile*()
  - Generic k-th of m-quantiles - *GetQuantile*()
  - Histogram of the sample's distribution - *GetHistogram*()
  - Mode(s) of the sample's distribution - *GetModes*()
- 2D statistics
  - Spearman rank correlation coefficient $\rho$ - *GetSpearman*()
  - Kendall rank correlation coefficient $\tau$-*b* - *GetKendall*()

## Intended Use and Functionality

The moment of distribution based statistical properties provide *rough* estimators of the shape of the sample data distribution. Arithmetic mean is an estimator of the position of the center of the data, variance (or standard deviation) - of the (squared) width of the distribution, skewness - of the symmetry of the distribution, and (excess) kurtosis - of the relative weight of the 'tails' with respect to the 'core' of the distribution.

The moment-based properties are sensitive to the 'far' outliers in the data, and because of that fact many statistical tests use median and inter-quartile distance as more 'robust' estimators of the 'average' and spread of the distribution. The module **ordered_functions** provides the functions for the calculation of such 'robust' properties:

- *Median* (or the *second quartile*, Q2) of the data as an estimator of the 'average' value
- The *first quartile* Q1 and the *third quartile* Q3 of the data, which are used for:
  - Estimation of the distribution width as *inter-quartile* distance as (Q3 - Q1) / 2
  - Estimation of the symmetry of the distribution by comparison of (Q3 - Q2) and (Q2 - Q1)

Furthermore, the sample data distribution's shape can be compared to the shape of another sample or model distribution using a histogram or Q-Q plot, which uses *quantiles* of the distribution. The calculation of the *histogram* and of a *generic quantile* of the distribution is impelemented as well.

Finally, the Pearson's correlation coefficient *r* shows how well relation between two random variables is described by a linear dependency as $y_i = a \times x_i + b$ . However, if two random variables are in a more complex relation $y_i = f(x_i)$ , where *f*() is monotonic but non-linear function, the rank correlation (like

Spearman or Kendall algorithms) is a better tool to detect the systematic relation between the variables than the Person's correlation.

# Design and Implementation

Similar to the module **base_function** all functions defined in the module **ordered_functions** are designed to to accept a generic sequence (or two sequences) containing only real number (integer or floating point) and / or instances of a class implementing measurements with uncertainty, which is expected to be API compatible with the **phyqus_lib.base_classes.MeasuredValue** class, i.e. to have fields / properties *Value* and *SE*. A different type of the input data set results in a **TypeError** typed exception. An empty data sequence, or unequal length X- and Y-data sequences result in **ValueError** typed exception. The input data sanity check is optional, and it can be skipped if the input data is quaranteed to be proper.

The moment related statistical properties calculation (**base_function** module) has a constant computation tine complexity of O(N), regardless of the type and power of the moment. The functions in the **ordered_functions** module, on the other hand, have various computation time complexity.

For instance, finding the minimal and maximal values in the sample requires iteration through the entire sample, thus the complexity is O(N). The calculation of the median, quartiles, generic quantiles and Spearman correlation coefficient require sorting of the data as the slowest part, thus the complexity is O(N*log(N)). If the input data is not only proper, but it is already sorted in the ascending order, the computation time complexity of finding min, max, median, Q1, Q3 or any generic quantile is reduced to accessing one or two elements of a list, in which case the computational time complexity is reduced to O(1).

Calculation of a histogram or the mode(s) of a distribution does not require sorting but the entire sample must be iterated through, thus the complexity is O(N) regardless of the input data being already sorted or not.

The Spearman rank correlation and Kendall rank correlation algorithms require the 2-D data set in its natural order, amd the time complexity of these algorithm is O(N*log(N)) and O(N^2) respectively.

Therefore, if the quantile (or quartile in particular) calculation is required more then once, e.g. for the calculation of inter-quartile distance (IQD) or Q-Q plot, it is beneficial to sanitize the input data and produce its sorted representation before-hands, and perform the calculations on the already sorted data using the keyword flag argument *DoCheck* = **False** to save the computation time. The drawback is, of course, the increase of the amount of the used computer memory.

**Mote**: there is no computational time gain in using the already sorted data in the computation of a histogram or the mode(s) of the distribution, whereas the Spearman and Kendall rank correlation computation MUST be performed on the not sorted data.

## Implementation notes

Calculation of any generic quantile, including Q1 and Q3, as well as of the median (Q2) in the case of the even sequence length is based on the linear interpolation between the value of the elements before and after the quantile cut-point.

Calculation of any quantile, including Q1 and Q3 but exluding median (Q2) requires the sequence being, at least, 2 elements long.

Calculation of the k-th of m-quantile of a sample can be performed with any sample's length N >= 2, however, it is meaningful only if N >= m.

The 0th of m-quantile is always the minimal value in the sample. Similarly, the m-th of m-qunatile is the maximal value in the sample. In other words, the sample is treated as the entire population.

The Spearman rank correlation coefficient is calculated as the Pearson's correlation coreffient of the *fractional ranks* of the input X and Y data.

The Kendall rank correlation coefficient is calculated using $\tau$-*b* algorithm, i.e. accounting for the ties.

For more information consult the DE001 document.

Implementation of the fractional ranks relies on the preservation of the insertion order into a **dict** in Python v3.7+; if the code is executed in an earlier version of the Python 3 interpreter, the additional sorting of the keys is applied.

## Edge-cases

The median of a sample containg only a single value is this value itself.

Both the Spearman and Kendall rank correlation coefficients are 1 if the input sequences are both of 1 element long.

During the calculation of a histogram of the sample's distribution the values in the vicinity of the bins' boundary can fall to either the left or right bin due to the rounding error, with the total number of the elements in all bins being preserved. This may cause visible changes in the shape of the histogram with minor variation of the bins size or number of bins.

## API conventions

Similar to the functions defined in the module **base_functions** (see UD001 document) the functions of this module accept the (optional) keyword-only arguments *SkipFrames* (defaults to 1, should be a positive integer) and *DoCheck* (defaults to **True**, should be any data type, which can be used in the Boolean context). The *SkipFrames* argument is used for truncation of the exception traceback (see module *base_exceptions* in the library **introspection_lib**) for the purposes of debuging and errors logging, when it is more important to find the place of the improper input data when to show all the path to where the exception is raised. The second keyword-only argument *DoCheck* is used only for the optimization as in avoiding redundant data sanity checks, data types convertion and sorting. The argument *DoCheck* = **False** should be passed only if the input is quaranteed to be:

- a sequence of only real numbers (**int** or **float**)
- sorted in the ascending order, except for the functions *GetHistogram*(), *GetModes*() and *GetKendals*(), for which the sorting is not required

The performance of the *GetHistogram*() function can be modified by the keyword-only arguments *NBins* and *BinSize* as follows:

- If the both arguments are not passed, or **None** the number of bins is set to the default value of 20
- If the *NBins* argument is provided as a positive integer, the *BinSize* argument is ignored, and the number of bins is set to the value of *NBins* argument

- If *NBins* argument is not passed, or it is **None**, whereas the *BinSize* argument is a positive integer or floating point number the width of each bin is set to the value of *BinSize* argument

With the calculations driven by the fixed number of bins the bin width is chosen such, that the minimal and maximal values found in the data sample are centered to the middle of the left-most and right-most bins of the histogram. If the bin width is fixed instead, the arithmetic mean of the sample is centered to the middle of its respective bin, and the number of bins to the left and to the right of it is selected such, than the minimal and maximal values found in the sample fall into the left-most and right-most bins respectively.

# API Reference

## Functions

**GetMin**(Data, *, SkipFrames = 1, DoCheck = True)

*Signature*:

seq(int OR float OR phyqus_lib.base_classes.MeasuredValue)/, *, int > 0, bool/ -> int OR float

*Args*:

- *Data*: **seq**(**int**0 OR **float** OR **phyqus_lib.base_classes.MeasuredValue**); a sequence of real numbers or 'measurements with uncertainty'
- *SkipFrames*: (keyword) **int** > 0; how many frames to hide in the exception traceback, defaults to 1
- *DoCheck*: (keyword) **bool**; flag if to perform the input data sanity check and convert the mixed sequence into a list of only real numbers

*Returns*:

**int** OR **float**: the calculated minimal value within the sample

*Raises*:

- **UT_TypeError**: mandatory argument is not a sequence of real numbers or measurements with uncertainty, OR any keyword argument is of improper type
- **UT_ValueError**: passed mandatory sequence is empty, OR any keyword argument is of the proper type but unacceptable value

*Description*:

Calculates the minimum value of a mixed sequence of real numbers and the measurements with uncertainty. Computation speed is O(N), unless the passed sequence is already sorted in ascending order sequence of real numbers, which is indicated by the keyword argument *DoCheck* = **False**, in which case the calculation speed is O(1).

**GetMax**(Data, *, SkipFrames = 1, DoCheck = True)

*Signature*:

seq(int OR float OR phyqus_lib.base_classes.MeasuredValue)/, *, int > 0, bool/ -> int OR float

*Args*:

- *Data*: **seq**(**int**0 OR **float** OR **phyqus_lib.base_classes.MeasuredValue**); a sequence of real numbers or 'measurements with uncertainty'
- *SkipFrames*: (keyword) **int** > 0; how many frames to hide in the exception traceback, defaults to 1
- *DoCheck*: (keyword) **bool**; flag if to perform the input data sanity check and convert the mixed sequence into a list of only real numbers

*Returns*:

**int** OR **float**: the calculated maximal value within the sample

*Raises*:

- **UT_TypeError**: mandatory argument is not a sequence of real numbers or measurements with uncertainty, OR any keyword argument is of improper type
- **UT_ValueError**: passed mandatory sequence is empty, OR any keyword argument is of the proper type but unacceptable value

*Description*:

Calculates the maximum value of a mixed sequence of real numbers and the measurements with uncertainty. Computation speed is O(N), unless the passed sequence is already sorted in ascending order sequence of real numbers, which is indicated by the keyword argument *DoCheck* = **False**, in which case the calculation speed is O(1).

**GetMedian**(Data, *, SkipFrames = 1, DoCheck = True)

*Signature*:

seq(int OR float OR phyqus_lib.base_classes.MeasuredValue)/, *, int > 0, bool/ -> int OR float

*Args*:

- *Data*: **seq**(**int**0 OR **float** OR **phyqus_lib.base_classes.MeasuredValue**); a sequence of real numbers or 'measurements with uncertainty'
- *SkipFrames*: (keyword) **int** > 0; how many frames to hide in the exception traceback, defaults to 1
- *DoCheck*: (keyword) **bool**; flag if to perform the input data sanity check and convert the mixed sequence into a list of only real numbers, and sort the values

*Returns*:

**int** OR **float**: the calculated median value of the sample

*Raises*:

- **UT_TypeError**: mandatory argument is not a sequence of real numbers or measurements with uncertainty, OR any keyword argument is of improper type
- **UT_ValueError**: passed mandatory sequence is empty, OR any keyword argument is of the proper type but unacceptable value

*Description*:

Calculates the median value of a mixed sequence of real numbers and the measurements with uncertainty. Computation speed is O(N*log(N)), unless the passed sequence is already sorted in ascending order sequence of real numbers, which is indicated by the keyword argument *DoCheck* = **False**, in which case the calculation speed is O(1).

**GetFirstQuartle**(Data, *, SkipFrames = 1, DoCheck = True)

*Signature*:

seq(int OR float OR phyqus_lib.base_classes.MeasuredValue)/, *, int > 0, bool/ -> int OR float

*Args*:

- *Data*: **seq**(**int**0 OR **float** OR **phyqus_lib.base_classes.MeasuredValue**); a sequence of real numbers or 'measurements with uncertainty'
- *SkipFrames*: (keyword) **int** > 0; how many frames to hide in the exception traceback, defaults to 1
- *DoCheck*: (keyword) **bool**; flag if to perform the input data sanity check and convert the mixed sequence into a list of only real numbers, and sort the values

*Returns*:

**int** OR **float**: the calculated first quartile of the sample

*Raises*:

- **UT_TypeError**: mandatory argument is not a sequence of real numbers or measurements with uncertainty, OR any keyword argument is of improper type
- **UT_ValueError**: passed mandatory sequence is shorter than 2 elements, OR any keyword argument is of the proper type but unacceptable value

*Description*:

Calculates the first quartile value of a mixed sequence of real numbers and the measurements with uncertainty. Computation speed is O(N*log(N)), unless the passed sequence is already sorted in ascending order sequence of real numbers, which is indicated by the keyword argument *DoCheck* = **False**, in which case the calculation speed is O(1).

**GetThirdQuartle**(Data, *, SkipFrames = 1, DoCheck = True)

*Signature*:

seq(int OR float OR phyqus_lib.base_classes.MeasuredValue)/, *, int > 0, bool/ -> int OR float

*Args*:

- *Data*: **seq**(**int**0 OR **float** OR **phyqus_lib.base_classes.MeasuredValue**); a sequence of real numbers or 'measurements with uncertainty'
- *SkipFrames*: (keyword) **int** > 0; how many frames to hide in the exception traceback, defaults to 1
- *DoCheck*: (keyword) **bool**; flag if to perform the input data sanity check and convert the mixed sequence into a list of only real numbers, and sort the values

*Returns*:

**int** OR **float**: the calculated third quartile of the sample

*Raises*:

- **UT_TypeError**: mandatory argument is not a sequence of real numbers or measurements with uncertainty, OR any keyword argument is of improper type
- **UT_ValueError**: passed mandatory sequence is shorter than 2 elements, OR any keyword argument is of the proper type but unacceptable value

*Description*:

Calculates the third quartile value of a mixed sequence of real numbers and the measurements with uncertainty. Computation speed is O(\N*log(N)), unless the passed sequence is already sorted in ascending order sequence of real numbers, which is indicated by the keyword argument *DoCheck* = **False**, in which case the calculation speed is O(1).

**GetQuantile**(Data, k, m, *, SkipFrames = 1, DoCheck = True)

*Signature*:

seq(int OR float OR phyqus_lib.base_classes.MeasuredValue), int >= 0, int > 0/, *, int > 0, bool/ -> int OR float

*Args*:

- *Data*: **seq**(**int**0 OR **float** OR **phyqus_lib.base_classes.MeasuredValue**); a sequence of real numbers or 'measurements with uncertainty'
- *k*: **int** >= 0; the quantile index, between 0 and m inclusively
- *m*: **int** > 0; the total number of quantiles
- *SkipFrames*: (keyword) **int** > 0; how many frames to hide in the exception traceback, defaults to 1
- *DoCheck*: (keyword) **bool**; flag if to perform the input data sanity check and convert the mixed sequence into a list of only real numbers, and sort the values

*Returns*:

**int** OR **float**: the calculated k-th of m-quantile of the sample

*Raises*:

- **UT_TypeError**: mandatory argument is not a sequence of real numbers or measurements with uncertainty, OR any keyword argument is of improper type, OR quantile index is not an integer, OR the total number of quantiles is not an integer
- **UT_ValueError**: passed mandatory sequence is shorter than 2 elements, OR any keyword argument is of the proper type but unacceptable value OR the total number of quantilies is negative integer or zero, OR the quantile index is negative integer or integer greater than the total number of qunatiles

*Description*:

Calculates the k-th of m-quantile value of a mixed sequence of real numbers and the measurements with uncertainty. Computation speed is O(N*log(N)), unless the passed sequence is already sorted in ascending order sequence of real numbers, which is indicated by the keyword argument *DoCheck* = **False**, in which case the calculation speed is O(1).

**GetHistogram**(Data, *, NBins=None, BinSize=None, SkipFrames=1, DoCheck=True)

*Signature*:

seq(int OR float OR phyqus_lib.base_classes.MeasuredValue)/, *, int > 0 OR None, int > 0 OR float > 0 OR None, int > 0, bool/ -> dict(int OR float -> int >=0)

*Args*:

- *Data*: **seq**(**int**0 OR **float** OR **phyqus_lib.base_classes.MeasuredValue**); a sequence of real numbers or 'measurements with uncertainty'
- *NBins*: (keyword) **int** > 0 OR **None**; the desired number of bins
- *BinSize*: (keyword) **int** > 0 OR **float** > 0 OR **None**; the desired bin size, ignored is NBins is passed as not None value
- *SkipFrames*: (keyword) **int** > 0; how many frames to hide in the exception traceback, defaults to 1
- *DoCheck*: (keyword) **bool**; flag if to perform the input data sanity check and convert the mixed sequence into a list of only real numbers

*Returns*:

**dict**(**int** OR **float** -> **int** >= 0): the calculated histogram of the sample's distribution

*Raises*:

- **UT_TypeError**: mandatory argument is not a sequence of real numbers or measurements with uncertainty, OR any keyword argument is of improper type
- **UT_ValueError**: passed mandatory sequence is empty, OR any keyword argument is of the proper type but unacceptable value

*Description*:

Calculates the histogram of number of apperance of 'mean' values belonging to the respective bins for the data sample. Either total number of bins OR the desired bin width can be specified, where number of bins takes the precedence. When neither value is defined, the default number of bins is 20. Computation speed is always O(N).

**GetModes**(Data, *, SkipFrames = 1, DoCheck = True)

*Signature*:

seq(int OR float OR phyqus_lib.base_classes.MeasuredValue)/, *, int > 0, bool/ -> list(int OR float)

*Args*:

- *Data*: **seq**(**int**0 OR **float** OR **phyqus_lib.base_classes.MeasuredValue**); a sequence of real numbers or 'measurements with uncertainty'
- *SkipFrames*: (keyword) **int** > 0; how many frames to hide in the exception traceback, defaults to 1
- *DoCheck*: (keyword) **bool**; flag if to perform the input data sanity check and convert the mixed sequence into a list of only real numbers

*Returns*:

**list**(**int** OR **float**): the calculated mode(s) of the sample's distribution

*Raises*:

- **UT_TypeError**: mandatory argument is not a sequence of real numbers or measurements with uncertainty, OR any keyword argument is of improper type
- **UT_ValueError**: passed mandatory sequence is empty, OR any keyword argument is of the proper type but unacceptable value

*Description*:

Calculates the mode(s) of a mixed sequence of real numbers and the measurements with uncertainty. Computation speed is always O(N).

**GetSpearman**(DataX, DataY, *, SkipFrames = 1, DoCheck = True)

*Signature*:

seq(int OR float OR phyqus_lib.base_classes.MeasuredValue)/, seq(int OR float OR phyqus_lib.base_classes.MeasuredValue)/, *, int > 0, bool/ -> int OR float

*Args*:

- *DataX*: **seq**(**int**0 OR **float** OR **phyqus_lib.base_classes.MeasuredValue**); a sequence of real numbers or 'measurements with uncertainty' as X data sequence
- *DataY*: **seq**(**int**0 OR **float** OR **phyqus_lib.base_classes.MeasuredValue**); a sequence of real numbers or 'measurements with uncertainty' as Y data sequence
- *SkipFrames*: (keyword) **int** > 0; how many frames to hide in the exception traceback, defaults to 1
- *DoCheck*: (keyword) **bool**; flag if to perform the input data sanity check and convert the mixed sequences into the lists of only real numbers

*Returns*:

**int** OR **float**: the calculated Spearman rank correlation coefficient

*Raises*:

- **UT_TypeError**: any of mandatory data arguments is not a sequence of real numbers or measurements with uncertainty, OR any keyword argument is of improper type
- **UT_ValueError**: any of the passed mandatory sequence is empty, OR any keyword argument is of the proper type but unacceptable value, OR the X and Y sequences are of different length

*Description*:

Calculates the Spearman rank correlation coeffificent of the paired mixed sequences of real numbers and the measurements with uncertainty. Computation speed is always O(N*log(N)).

**GetKendall**(DataX, DataY, *, SkipFrames = 1, DoCheck = True)

*Signature*:

seq(int OR float OR phyqus_lib.base_classes.MeasuredValue)/, seq(int OR float OR phyqus_lib.base_classes.MeasuredValue)/, *, int > 0, bool/ -> int OR float

*Args*:

- *DataX*: **seq**(**int**0 OR **float** OR **phyqus_lib.base_classes.MeasuredValue**); a sequence of real numbers or 'measurements with uncertainty' as X data sequence
- *DataY*: **seq**(**int**0 OR **float** OR **phyqus_lib.base_classes.MeasuredValue**); a sequence of real numbers or 'measurements with uncertainty' as Y data sequence
- *SkipFrames*: (keyword) **int** > 0; how many frames to hide in the exception traceback, defaults to 1
- *DoCheck*: (keyword) **bool**; flag if to perform the input data sanity check and convert the mixed sequences into the lists of only real numbers

*Returns*:

**int** OR **float**: the calculated Kendall rank correlation coefficient

*Raises*:

- **UT_TypeError**: any of mandatory data arguments is not a sequence of real numbers or measurements with uncertainty, OR any keyword argument is of improper type
- **UT_ValueError**: any of the passed mandatory sequence is empty, OR any keyword argument is of the proper type but unacceptable value, OR the X and Y sequences are of different length

*Description*:

Calculates the Kendall rank correlation coeffificent of the paired mixed sequences of real numbers and the measurements with uncertainty. Computation speed is always O(N^2).