Requirements for the Module statistics_lib.distribution_classes

Conventions

Requirements listed in this document are constructed according to the following structure:

Requirement ID: REQ-UVW-XYZ

Title: Title / name of the requirement

Description: Description / definition of the requirement

Verification Method: I / A / T / D

The requirement ID starts with the fixed prefix 'REQ'. The prefix is followed by 3 letters abbreviation (in here 'UVW'), which defines the requiement type - e.g. 'FUN' for a functional and capability requirement, 'AWM' for an alarm, warnings and operator messages, etc. 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 requirement ordering number for this object. E.g. 'REQ-FUN-112'. Each requirement type has its own counter, thus 'REQ-FUN-112' and 'REQ-AWN-112' requirements 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

Functional and capability requirements

Requirement ID: REQ-FUN-400

Title: Functionality implemented by the module (scope)

Description: The module should provide classes implementing the following continuous and discrete random distribution using the definitions in DE002 document:

- Generic Gauss (normal) distribution $\mathbb{N}(\mu, \sigma)$
- Z-distribution distribution $\mathbb{Z} = \mathbb{N}(0, 1)$
- Student`s t-distribution
- χ^2-distribution
- F-distribution
- Poisson distribution

- Binomial distribution

If other distriutions described in DE002 document are also implemented, they shoud function as expected and have the same API as the rest of the classes.

Verification Method: A

Requirement ID: REQ-FUN-401

Title: Instantiation of a random distribution class

Description: All classes, except for Z-distribution, should accept and require as many aruments of the initialization method (during instantiation) as they have parameters, defining the distribition. These arguments must be of the proper data type and of the accepted values range, as the specific distribution is defined for. The exception is Z-distribution, which has no parameters.

Verification Method: T

Requirement ID: REQ-FUN-402

Title: Access to the parameters of a random distribution

Description: All parameters of the distribution should be accessible via a respective getter property. It should be possible to change the value of each parameter via a respective setter property, as long as the assigment value is of the proper type and value. The exception is Z-distribution, which has no parameters.

Verification Method: T

Requirement ID: REQ-FUN-403

Title: Required statistical properties

Description: All distribution classes should provide the following read-only properties representing the statistical properties of the distribution:

- Mean
- Max
- Min
- Median
- Q1
- Q3
- Var
- Sigma
- Skew
- Kurt

The exception is generic Gaussian distribution, for which Mean and Sigma are also the parameters of the distribution, therefore they must be read-write properties. These properties should have floating point type, which values match the definitions in DE002 document.

Verification Method: T

Requirement ID: REQ-FUN-404

Title: Probability density / mass function

Description: All classes should have instance method *pdf*(), which returns the value of the probability density function for a continuous distribution and of the probability mass function for a discrete distribution if the passed argument is an integer or floating point number (real) and of the value for which the distribution is defined. In the case of the real number argument with the value outside the acceptable range - zero value (0) should be returned instead of rising an exception, which includes not integer (floating point) number argument for a discrete distribution.

Verification Method: T

Requirement ID: REQ-FUN-405

Title: Cummulative probability function

Description: All classes should have instance method *cdf*(), which returns the value of the cummulative probability density function. For a continuous distribution it should be a continuous function. For a discrete distribution it should be a step function with a constant value on the semi-open interval $\left[x_k, x_{k+1}\right)$. It should accept and real number value as its argument. If it is less than the minimum of the aceeptance range the returned value should be zero (0), and one (1) if the value is greater than the maximum of the acceptance range.

Verification Method: T

Requirement ID: REQ-FUN-406

Title: Quantile function

Description: All classes should have instance method *qf*(), which returns the value of the quantile function, which is inverse to the cummulative probability density function. It should accept only floating point numbers in the open range (0, 1). For a continuous distribution it should return a floating point number within the range for which the distribution is defined. For a discrete distribution it should return a floating point number, which can be equal to any of the integer values, for which the distribution is defined, or being in between two consecutive integer within the distribution acceptance range. In the special case when p < pdf(min(X)) == cdf(min(X)), min(X) - 1 < qf(p) < min(X). The second function *getQuantile*(k, m) should be defined for 0 < k : **int** < m : **int** and return *qf*(k/m).

Verification Method: T

Requirement ID: REQ-FUN-407

Title: Generation of histogram of distribution

Description: All classes should have instance method *getHistogram*(), which should calculate a histrogram of the distribution as follows:

- the arguments of the method are: central value of the minimal bin X_{min}, central value of the maximal bin X_{max} and the number of bins *N* > 1
- each bin has the same width $S = \frac{X_{max} - X_{min}}{N-1}$
- the bins are indexed from 0, and the central value of each bin is $x_k = X_{min} + k * S$, where $0 \leq k \leq N-1$
- for the k-th bin the value is calculated as $cdf(x_k + S/2) - cdf(x_k - S/2)$
- the result is returned as a tuple of 2-tuples with each nested tuple being a pair of the central value and the frequency for the given bin, where the bins are sorted in the ascending order of the central values

Verification Method: D

Requirement ID: REQ-FUN-408

Title: Generation of a random number

Description: All classes should have instance method w/o arguments *random*(), which returns a random value (floating point for continuous and integer for discrete distribution), with those numbers being distributed with the respective random disribution.

Verification Method: D

Alarms, warnings and operator messages

Requirement ID: REQ-AWM-400

Title: Improper type of an agrument

Description: The **TypeError** or its sub-class should be raised if an improper data type argument is passed into any parametric method of any class, including the initialization / instantiation method, and assigment to a setter property. Specifically,

- Instantiation of the classes and changing the distribution parameters via setter properties
  - Gaussian - mean or sigma are not **int** or **float**
  - Student's - degrees of freedom is not **int** or **float**
  - Chi-square - degrees of freedom is not **int** or **float**
  - F - either of the degress of freedom is not **int** or **float**
  - Exponential - rate is not **int** or **float**
  - Gamma - either shape or rate are not **int** or **float**
  - Erlang - shape is not **int** or rate is not **int** or **float**
  - Poisson - rate is not **int** or **float**
  - Binomial - probability is not **int** or **float**, or the number of trials is not **int**
  - Geometrical - probability is not **int** or **float**
  - Hypergeometrical - either of the parameters is not **int**
- pdf() and cdf() - the argument is not **int** or **float**
- qf() - the argument is not **float**

- getQuantile(k, m) - either of the arguments is not **int**
- getHistogram(min, max, NBins) - either min or max is not **int** or **float**, or NBins is not **int**

Verification Method: T

Requirement ID: REQ-AWM-401

Title: Improper value of an agrument

Description: The **ValueError** or its sub-class should be raised if an argument of a proper data type, but of unacceptable value is passed into any parametric method of any class, including the initialization / instantiation method, and assigment to a setter property. Specifically,

- Instantiation of the classes and changing the distribution parameters via setter properties
    - Gaussian - sigma is <= 0
    - Student's - degrees of freedom is <= 0
    - Chi-square - degrees of freedom is <= 0
    - F - either of the degress of freedom is <= 0
    - Exponential - rate is <= 0
    - Gamma - either shape or rate is <= 0
    - Erlang - shape or rate is <= 0
    - Poisson - rate is <= 0
    - Binomial - probability is not in the interval [0, 1], or number of trials is < 0
    - Geometrical - probability is not in the interval [0, 1]
    - Hypergeometrical - either of the parameters is not < 0, or number of 'succeess' objects or trials is > than total number of objects
- 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

Verification Method: T