The STATISTICS Reference Manual

Consolidated Common Lisp statistical functions, version 1.0.0

Steve Nunez `<steve@symbolics.tech>`

This manual was generated automatically by Declt 4.0b2.

Table of Contents

Copying

1 Systems

The main system appears first, followed by any subsystem dependency.

1.1 `statistics`

A consolidated system of statistical functions

Long Name
Consolidated Common Lisp statistical functions

Author Steve Nunez `<steve@symbolics.tech>`

Home Page
`https://lisp-stat.dev/`

Source Control
`(GIT https://github.com/Lisp-Stat/statistics.git)`

Bug Tracker
`https://github.com/Lisp-Stat/statistics/issues`

License MS-PL

Long Description
This system is a consolidation of three Common Lisp statistics libraries:
- Tamas Papp's library, focusing on central moments
- Larry Hungers general statistical library
- Gary Warren King's (GWK) general statistical library, cl-mathstats

As of Q3 2022, CL-MATHSTATS is usable with Lisp-Stat, but not incorporated. This is due to it being rather deeply embedded into its own ecosystem of utilities libraries (metatilities-base and cl-containers and the lift test framework) that have in some cases been superseded by alexandria, anaphora or numerical-utilities. In short, we recommend using CL-MATHSTATS when you need to, recognising that you'll be hauling in a parallel system of math, statistics and utilities. Long term, we're working to port CL-MATHSTATS on a case-by-case basis.

Version 1.0.0

Dependencies
- `anaphora` (system).
- `alexandria` (system).
- `distributions` (system).
- `let-plus` (system).
- `num-utils` (system).
- `org.tfeb.conduit-packages` (system).

Source [`statistics.asd`], page 5.

Child Components
- [`license`], page 10 (file).
- [`lh-statistics.lisp`], page 5 (file).
- [`nu-statistics.lisp`], page 7 (file).
- [`ls-statistics.lisp`], page 9 (file).
- [`pkgdcl.lisp`], page 10 (file).

2 Files

Files are sorted by type and then listed depth-first from the systems components trees.

2.1 Lisp

2.1.1 statistics/statistics.asd

Source [statistics.asd], page 5.

Parent Component
[statistics], page 3 (system).

ASDF Systems
[statistics], page 3.

2.1.2 statistics/lh-statistics.lisp

Source [statistics.asd], page 5.

Parent Component
[statistics], page 3 (system).

Packages [lh.statistics], page 11.

Public Interface

- [bin-and-count], page 17 (function).
- [binomial-cumulative-probability], page 17 (function).
- [binomial-ge-probability], page 17 (function).
- [binomial-probability], page 17 (function).
- [binomial-probability-ci], page 18 (function).
- [binomial-test-one-sample], page 18 (function).
- [binomial-test-one-sample-sse], page 18 (function).
- [binomial-test-paired-sse], page 18 (function).
- [binomial-test-two-sample], page 18 (function).
- [binomial-test-two-sample-sse], page 18 (function).
- [chi-square], page 19 (function).
- [chi-square-cdf], page 19 (function).
- [chi-square-test-for-trend], page 19 (function).
- [chi-square-test-one-sample], page 19 (function).
- [chi-square-test-rxc], page 19 (function).
- [choose], page 19 (function).
- [coefficient-of-variation], page 19 (function).
- [convert-to-standard-normal], page 19 (function).
- [correlation-coefficient], page 20 (function).
- [correlation-sse], page 20 (function).
- [correlation-test-two-sample], page 20 (function).
- [correlation-test-two-sample-on-sequences], page 20 (function).
- [f-significance], page 21 (function).
- [f-test], page 21 (function).

- [`false-discovery-correction`], page 21 (function).
- [`fisher-exact-test`], page 21 (function).
- [`fisher-z-transform`], page 21 (function).
- [`geometric-mean`], page 21 (function).
- [`linear-regression`], page 22 (function).
- [`mcnemars-test`], page 22 (function).
- [`mean`], page 22 (function).
- [`mean-sd-n`], page 22 (function).
- [`median`], page 23 (function).
- [`mode`], page 23 (function).
- [`normal-mean-ci`], page 23 (function).
- [`normal-mean-ci-on-sequence`], page 23 (function).
- [`normal-pdf`], page 23 (function).
- [`normal-sd-ci`], page 23 (function).
- [`normal-sd-ci-on-sequence`], page 23 (function).
- [`normal-variance-ci`], page 23 (function).
- [`normal-variance-ci-on-sequence`], page 24 (function).
- [`percentile`], page 24 (function).
- [`permutations`], page 24 (function).
- [`phi`], page 24 (function).
- [`poisson-cumulative-probability`], page 24 (function).
- [`poisson-ge-probability`], page 24 (function).
- [`poisson-mu-ci`], page 24 (function).
- [`poisson-probability`], page 24 (function).
- [`poisson-test-one-sample`], page 25 (function).
- [`random-normal`], page 25 (function).
- [`random-pick`], page 25 (function).
- [`random-sample`], page 25 (function).
- [`round-float`], page 25 (function).
- [`sample-range`], page 25 (function).
- [`sd`], page 25 (function).
- [`sign-test`], page 25 (function).
- [`sign-test-on-sequences`], page 26 (function).
- [`spearman-rank-correlation`], page 26 (function).
- [`square`], page 17 (macro).
- [`standard-deviation`], page 26 (function).
- [`standard-error-of-the-mean`], page 26 (function).
- [`t-distribution`], page 26 (function).
- [`t-significance`], page 27 (function).
- [`t-test-one-sample`], page 27 (function).
- [`t-test-one-sample-on-sequence`], page 27 (function).
- [`t-test-one-sample-sse`], page 27 (function).
- [`t-test-paired`], page 27 (function).

- [`t-test-paired-on-sequences`], page 27 (function).
- [`t-test-paired-sse`], page 27 (function).
- [`t-test-two-sample`], page 28 (function).
- [`t-test-two-sample-on-sequences`], page 28 (function).
- [`t-test-two-sample-sse`], page 28 (function).
- [`test-variables`], page 17 (macro).
- [`variance`], page 28 (function).
- [`wilcoxon-signed-rank-test`], page 29 (function).
- [`wilcoxon-signed-rank-test-on-sequences`], page 29 (function).
- [`z`], page 29 (function).
- [`z-test`], page 29 (function).
- [`z-test-on-sequence`], page 29 (function).

Internals

- [`average-rank`], page 37 (function).
- [`beta-incomplete`], page 37 (function).
- [`binomial-le-probability`], page 37 (function).
- [`error-function`], page 38 (function).
- [`error-function-complement`], page 38 (function).
- [`factorial`], page 38 (function).
- [`find-critical-value`], page 38 (function).
- [`gamma-incomplete`], page 38 (function).
- [`gamma-ln`], page 39 (function).
- [`round-up`], page 39 (function).
- [`safe-exp`], page 39 (function).
- [`sign`], page 39 (function).
- [`underflow-goes-to-zero`], page 36 (macro).

2.1.3 `statistics/nu-statistics.lisp`

Source [`statistics.asd`], page 5.

Parent Component
[`statistics`], page 3 (system).

Packages [`nu.statistics`], page 13.

Public Interface

- [`*central-sample-moments-default-degree*`], page 17 (special variable).
- [`add`], page 29 (generic function).
- [`as-alist`], page 33 (method).
- [`central-m2`], page 30 (generic function).
- [`central-m3`], page 30 (generic function).
- [`central-m4`], page 30 (generic function).
- [`central-sample-moments`], page 30 (generic function).
- [`central-sample-moments`], page 34 (structure).
- [`central-sample-moments-degree`], page 18 (function).

- [cross-tabulate], page 20 (function).
- [empirical-quantile], page 20 (function).
- [empirical-quantile-probabilities], page 20 (function).
- [empty-accumulator], page 33 (condition).
- [ensure-sorted-reals], page 31 (generic function).
- [ensure-sorted-vector], page 20 (function).
- [information-not-collected-in-accumulator], page 33 (condition).
- [kurtosis], page 31 (generic function).
- [make-sparse-counter], page 22 (function).
- [mean], page 31 (generic function).
- [median], page 31 (generic function).
- [not-enough-elements-in-accumulator], page 34 (condition).
- [num=], page 33 (method).
- [pool], page 25 (function).
- [print-object], page 33 (method).
- [print-object], page 33 (method).
- [quantile], page 32 (generic function).
- [quantiles], page 32 (generic function).
- [sd], page 32 (generic function).
- [skewness], page 32 (generic function).
- [sorted-reals], page 35 (structure).
- [sorted-reals-elements], page 26 (function).
- [sparse-counter], page 36 (structure).
- [sparse-counter-count], page 26 (function).
- [sparse-counter-table], page 26 (reader).
- [tabulate], page 28 (function).
- [tally], page 32 (generic function).
- [variance], page 33 (generic function).
- [weighted-quantiles], page 29 (function).

Internals

- [&sorted-reals], page 36 (macro).
- [&sorted-reals-r/o], page 36 (macro).
- [central-sample-moments-m], page 37 (reader).
- [(setf central-sample-moments-m)], page 37 (writer).
- [central-sample-moments-p], page 37 (function).
- [central-sample-moments-s2], page 37 (reader).
- [(setf central-sample-moments-s2)], page 37 (writer).
- [central-sample-moments-s3], page 37 (reader).
- [(setf central-sample-moments-s3)], page 37 (writer).
- [central-sample-moments-s4], page 37 (reader).
- [(setf central-sample-moments-s4)], page 37 (writer).
- [central-sample-moments-w], page 38 (function).

- [(setf central-sample-moments-w)], page 38 (function).
- [copy-central-sample-moments], page 38 (function).
- [copy-sorted-reals], page 38 (function).
- [copy-sparse-counter], page 38 (function).
- [copy-tally-mixin], page 38 (function).
- [define-central-sample-moment], page 36 (macro).
- [make-central-sample-moments], page 39 (function).
- [make-sorted-reals], page 39 (function).
- [make-sparse-counter%], page 39 (function).
- [make-tally-mixin], page 39 (function).
- [pool2], page 40 (generic function).
- [sort-reals], page 39 (function).
- [sorted-reals-ordered-elements], page 39 (reader).
- [(setf sorted-reals-ordered-elements)], page 39 (writer).
- [sorted-reals-p], page 40 (function).
- [sorted-reals-unordered-elements], page 40 (reader).
- [(setf sorted-reals-unordered-elements)], page 40 (writer).
- [sparse-counter-p], page 40 (function).
- [tally-mixin], page 41 (structure).
- [tally-mixin-p], page 40 (function).
- [tally-mixin-w], page 40 (reader).
- [(setf tally-mixin-w)], page 40 (writer).
- [weighted-empirical-quantile], page 40 (function).
- [weighted-quantile-p-table], page 40 (function).

2.1.4 statistics/ls-statistics.lisp

Source [statistics.asd], page 5.

Parent Component
[statistics], page 3 (system).

Packages [ls.statistics], page 15.

Public Interface
- [fivenum], page 21 (function).
- [interquartile-range], page 22 (function).
- [mean], page 22 (function).
- [variance], page 28 (function).

2.1.5 statistics/pkgdcl.lisp

Source [statistics.asd], page 5.

Parent Component
[statistics], page 3 (system).

Packages [statistics-1], page 15.

2.2 Static

2.2.1 `statistics/license`

Source [`statistics.asd`], page 5.

Parent Component
[`statistics`], page 3 (system).

3 Packages

Packages are listed by definition order.

3.1 lh.statistics

The formulas and methods used are largely taken from Bernard Rosner, *Fundamentals of Biostatistics* 5th Edition. 'Rosner x' is a page number. Some numeric functions were taken from CLASP, a 1994 common lisp package that implemented some of the statistical functions from *Numeric recipes in C* For CLASP functions, see copyright notice below.

These abreviations used in function and variable names:
ci = confidence interval
cdf = cumulative density function
ge = greater than or equal to
le = less than or equal to
pdf = probability density function
sd = standard deviation
rxc = rows by columns
sse = sample size estimate

Source [`lh-statistics.lisp`], page 5.

Use List `common-lisp`.

Public Interface

- [`bin-and-count`], page 17 (function).
- [`binomial-cumulative-probability`], page 17 (function).
- [`binomial-ge-probability`], page 17 (function).
- [`binomial-probability`], page 17 (function).
- [`binomial-probability-ci`], page 18 (function).
- [`binomial-test-one-sample`], page 18 (function).
- [`binomial-test-one-sample-sse`], page 18 (function).
- [`binomial-test-paired-sse`], page 18 (function).
- [`binomial-test-two-sample`], page 18 (function).
- [`binomial-test-two-sample-sse`], page 18 (function).
- [`chi-square`], page 19 (function).
- [`chi-square-cdf`], page 19 (function).
- [`chi-square-test-for-trend`], page 19 (function).
- [`chi-square-test-one-sample`], page 19 (function).
- [`chi-square-test-rxc`], page 19 (function).
- [`choose`], page 19 (function).
- [`coefficient-of-variation`], page 19 (function).
- [`convert-to-standard-normal`], page 19 (function).
- [`correlation-coefficient`], page 20 (function).
- [`correlation-sse`], page 20 (function).
- [`correlation-test-two-sample`], page 20 (function).
- [`correlation-test-two-sample-on-sequences`], page 20 (function).
- [`f-significance`], page 21 (function).
- [`f-test`], page 21 (function).

- [false-discovery-correction], page 21 (function).
- [fisher-exact-test], page 21 (function).
- [fisher-z-transform], page 21 (function).
- [geometric-mean], page 21 (function).
- [linear-regression], page 22 (function).
- [mcnemars-test], page 22 (function).
- [mean], page 22 (function).
- [mean-sd-n], page 22 (function).
- [median], page 23 (function).
- [mode], page 23 (function).
- [normal-mean-ci], page 23 (function).
- [normal-mean-ci-on-sequence], page 23 (function).
- [normal-pdf], page 23 (function).
- [normal-sd-ci], page 23 (function).
- [normal-sd-ci-on-sequence], page 23 (function).
- [normal-variance-ci], page 23 (function).
- [normal-variance-ci-on-sequence], page 24 (function).
- [percentile], page 24 (function).
- [permutations], page 24 (function).
- [phi], page 24 (function).
- [poisson-cumulative-probability], page 24 (function).
- [poisson-ge-probability], page 24 (function).
- [poisson-mu-ci], page 24 (function).
- [poisson-probability], page 24 (function).
- [poisson-test-one-sample], page 25 (function).
- [random-normal], page 25 (function).
- [random-pick], page 25 (function).
- [random-sample], page 25 (function).
- [round-float], page 25 (function).
- [sample-range], page 25 (function).
- [sd], page 25 (function).
- [sign-test], page 25 (function).
- [sign-test-on-sequences], page 26 (function).
- [spearman-rank-correlation], page 26 (function).
- [square], page 17 (macro).
- [standard-deviation], page 26 (function).
- [standard-error-of-the-mean], page 26 (function).
- [t-distribution], page 26 (function).
- [t-significance], page 27 (function).
- [t-test-one-sample], page 27 (function).
- [t-test-one-sample-on-sequence], page 27 (function).
- [t-test-one-sample-sse], page 27 (function).
- [t-test-paired], page 27 (function).

- [t-test-paired-on-sequences], page 27 (function).
- [t-test-paired-sse], page 27 (function).
- [t-test-two-sample], page 28 (function).
- [t-test-two-sample-on-sequences], page 28 (function).
- [t-test-two-sample-sse], page 28 (function).
- [test-variables], page 17 (macro).
- [variance], page 28 (function).
- [wilcoxon-signed-rank-test], page 29 (function).
- [wilcoxon-signed-rank-test-on-sequences], page 29 (function).
- [z], page 29 (function).
- [z-test], page 29 (function).
- [z-test-on-sequence], page 29 (function).

Internals

- [average-rank], page 37 (function).
- [beta-incomplete], page 37 (function).
- [binomial-le-probability], page 37 (function).
- [error-function], page 38 (function).
- [error-function-complement], page 38 (function).
- [factorial], page 38 (function).
- [find-critical-value], page 38 (function).
- [gamma-incomplete], page 38 (function).
- [gamma-ln], page 39 (function).
- [round-up], page 39 (function).
- [safe-exp], page 39 (function).
- [sign], page 39 (function).
- [underflow-goes-to-zero], page 36 (macro).

3.2 nu.statistics

Source [nu-statistics.lisp], page 7.

Use List

- alexandria.
- anaphora.
- common-lisp.
- let-plus.
- num-utils.arithmetic.
- num-utils.num=.
- num-utils.utilities.

Public Interface

- [*central-sample-moments-default-degree*], page 17 (special variable).
- [add], page 29 (generic function).
- [central-m2], page 30 (generic function).
- [central-m3], page 30 (generic function).

- [central-sample-moments-w], page 38 (function).
- [(setf central-sample-moments-w)], page 38 (function).
- [copy-central-sample-moments], page 38 (function).
- [copy-sorted-reals], page 38 (function).
- [copy-sparse-counter], page 38 (function).
- [copy-tally-mixin], page 38 (function).
- [define-central-sample-moment], page 36 (macro).
- [make-central-sample-moments], page 39 (function).
- [make-sorted-reals], page 39 (function).
- [make-sparse-counter%], page 39 (function).
- [make-tally-mixin], page 39 (function).
- [pool2], page 40 (generic function).
- [sort-reals], page 39 (function).
- [sorted-reals-ordered-elements], page 39 (reader).
- [(setf sorted-reals-ordered-elements)], page 39 (writer).
- [sorted-reals-p], page 40 (function).
- [sorted-reals-unordered-elements], page 40 (reader).
- [(setf sorted-reals-unordered-elements)], page 40 (writer).
- [sparse-counter-p], page 40 (function).
- [tally-mixin], page 41 (structure).
- [tally-mixin-p], page 40 (function).
- [tally-mixin-w], page 40 (reader).
- [(setf tally-mixin-w)], page 40 (writer).
- [weighted-empirical-quantile], page 40 (function).
- [weighted-quantile-p-table], page 40 (function).

3.3 statistics-1

Source [pkgdcl.lisp], page 10.

3.4 ls.statistics

Source [ls-statistics.lisp], page 9.

Use List common-lisp.

Public Interface

- [fivenum], page 21 (function).
- [interquartile-range], page 22 (function).
- [mean], page 22 (function).
- [variance], page 28 (function).

4 Definitions

Definitions are sorted by export status, category, package, and then by lexicographic order.

4.1 Public Interface

4.1.1 Special variables

`*central-sample-moments-default-degree*` [Special Variable]

Default degree for (weighted) central sample moments.

Package [`nu.statistics`], page 13.

Source [`nu-statistics.lisp`], page 7.

4.1.2 Macros

`square` (*x*) [Macro]

Package [`lh.statistics`], page 11.

Source [`lh-statistics.lisp`], page 5.

`test-variables` (**&rest** *args*) [Macro]

Package [`lh.statistics`], page 11.

Source [`lh-statistics.lisp`], page 5.

4.1.3 Ordinary functions

`bin-and-count` (*sequence n*) [Function]

Make N equal width bins and count the number of elements of sequence that belong in each.

Package [`lh.statistics`], page 11.

Source [`lh-statistics.lisp`], page 5.

`binomial-cumulative-probability` (*n k p*) [Function]

Return P(X<k) for X a binomial random variable with parameters n & p. Bionomial expecations for fewer than k events in N trials, each having probability p. This is also known as probability mass function (PMF), the probability of getting exactly k successes in n independent Bernoulli trials.

Package [`lh.statistics`], page 11.

Source [`lh-statistics.lisp`], page 5.

`binomial-ge-probability` (*n k p*) [Function]

The probability of k or more occurances in N events, each with probability p.

Package [`lh.statistics`], page 11.

Source [`lh-statistics.lisp`], page 5.

`binomial-probability` (*n k p*) [Function]

Return P(X=k) for X a binomial random variable with parameters n & p. Binomial expectations for seeing k events in N trials, each having probability p. Use the Poisson approximation if N>100 and P<0.01.

Package [`lh.statistics`], page 11.

Source [`lh-statistics.lisp`], page 5.

`binomial-probability-ci` (*n p alpha* **&key** *exact?*) [Function]

Confidence intervals on a binomial probability. If a binomial probability of p has been observed in N trials, what is the 1-alpha confidence interval around p? Approximate (using normal theory approximation) when npq >= 10 unless told otherwise

Package [`lh.statistics`], page 11.

Source [`lh-statistics.lisp`], page 5.

`binomial-test-one-sample` (*p-hat n p* **&key** *tails exact?*) [Function]

The significance of a one sample test for the equality of an observed probability p-hat to an expected probability p under a binomial distribution with N observations. Use the normal theory approximation if n*p*(1-p) > 10 (unless the exact flag is true).

Package [`lh.statistics`], page 11.

Source [`lh-statistics.lisp`], page 5.

`binomial-test-one-sample-sse` (*p-estimated p-null* **&key** *alpha 1-beta tails*) [Function]

Returns the number of subjects needed to test whether an observed probability is significantly different from a particular binomial null hypothesis with a significance alpha and a power 1-beta.

Package [`lh.statistics`], page 11.

Source [`lh-statistics.lisp`], page 5.

`binomial-test-paired-sse` (*pd pa* **&key** *alpha 1-beta tails*) [Function]

Sample size estimate for the McNemar (discordant pairs) test. Pd is the projected proportion of discordant pairs among all pairs, and Pa is the projected proportion of type A pairs among discordant pairs. alpha, 1-beta and tails are as above. Returns the number of individuals necessary; that is twice the number of matched pairs necessary.

Package [`lh.statistics`], page 11.

Source [`lh-statistics.lisp`], page 5.

`binomial-test-two-sample` (*p-hat1 n1 p-hat2 n2* **&key** *tails exact?*) [Function]

Are the observed probabilities of an event (p-hat1 and p-hat2) in N1/N2 trials different? The normal theory method implemented here. The exact test is Fisher's contingency table method, below.

Package [`lh.statistics`], page 11.

Source [`lh-statistics.lisp`], page 5.

`binomial-test-two-sample-sse` (*p1 p2* **&key** *alpha sample-ratio 1-beta tails*) [Function]

The number of subjects needed to test if two binomial probabilities are different at a given significance alpha and power 1-beta. The sample sizes can be unequal; the p2 sample is sample-sse-ratio * the size of the p1 sample. It can be a one tailed or two tailed test.

Package [`lh.statistics`], page 11.

Source [`lh-statistics.lisp`], page 5.

`central-sample-moments-degree` (*central-sample-moments*) [Function]

Return the degree of CENTRAL-SAMPLE-MOMENTS.

Package [`nu.statistics`], page 13.

Source [`nu-statistics.lisp`], page 7.

`chi-square` (*dof percentile*) [Function]

;; Returns the point which is the indicated percentile in the Chi Square distribution with dof degrees of freedom.

Package [`lh.statistics`], page 11.

Source [`lh-statistics.lisp`], page 5.

`chi-square-cdf` (*x dof*) [Function]

Chi-square-cdf computes the left hand tail area under the chi square distribution under dof degrees of freedom up to X.

Package [`lh.statistics`], page 11.

Source [`lh-statistics.lisp`], page 5.

`chi-square-test-for-trend` (*row1-counts row2-counts* **&optional** *scores*) [Function]

This test works on a 2xk table and assesses if there is an increasing or decreasing trend. Arguments are equal sized lists counts. Optionally, provide a list of scores, which represent some numeric attribute of the group. If not provided, scores are assumed to be 1 to k.

Package [`lh.statistics`], page 11.

Source [`lh-statistics.lisp`], page 5.

`chi-square-test-one-sample` (*variance n sigma-squared* **&key** *tails*) [Function]

The significance of a one sample Chi square test for the variance of a normal distribution. Variance is the observed variance, N is the number of observations, and sigma-squared is the test variance.

Package [`lh.statistics`], page 11.

Source [`lh-statistics.lisp`], page 5.

`chi-square-test-rxc` (*contingency-table*) [Function]

Takes contingency-table, an RxC array, and returns the significance of the relationship between the row variable and the column variable. Any difference in proportion will cause this test to be significant – consider using the test for trend instead if you are looking for a consistent change.

Package [`lh.statistics`], page 11.

Source [`lh-statistics.lisp`], page 5.

`choose` (*n k*) [Function]

How may ways to take n things taken k at a time, when order doesn't matter

Package [`lh.statistics`], page 11.

Source [`lh-statistics.lisp`], page 5.

`coefficient-of-variation` (*sequence*) [Function]

Return coefficient of variation

Package [`lh.statistics`], page 11.

Source [`lh-statistics.lisp`], page 5.

`convert-to-standard-normal` (*x mu sigma*) [Function]

Convert X from a Normal distribution with mean mu and variance sigma to standard normal

Package [`lh.statistics`], page 11.

Source [`lh-statistics.lisp`], page 5.

`correlation-coefficient` (*points*) [Function]

Pearson correlation

Package [`lh.statistics`], page 11.

Source [`lh-statistics.lisp`], page 5.

`correlation-sse` (*rho* **&key** *alpha 1-beta*) [Function]

Returns the size of a sample necessary to find a correlation of expected value rho with significance alpha and power 1-beta.

Package [`lh.statistics`], page 11.

Source [`lh-statistics.lisp`], page 5.

`correlation-test-two-sample` (*r1 n1 r2 n2* **&key** *tails*) [Function]

Test if two correlation coefficients are different. Users Fisher's Z test.

Package [`lh.statistics`], page 11.

Source [`lh-statistics.lisp`], page 5.

`correlation-test-two-sample-on-sequences` (*points1 points2* **&key** *tails*) [Function]

Package [`lh.statistics`], page 11.

Source [`lh-statistics.lisp`], page 5.

`cross-tabulate` (*sequence1 sequence2* **&key** *test*) [Function]

Cross-tabulate two sequences (using a SPARSE-COUNTER with the given TEST). TEST is used to compare conses.

Package [`nu.statistics`], page 13.

Source [`nu-statistics.lisp`], page 7.

`empirical-quantile` (*sorted-vector q*) [Function]

Return the empirical quantile of a vector of real numbers, sorted in ascending order (not checked). Uses a 0.5 correction.

Package [`nu.statistics`], page 13.

Source [`nu-statistics.lisp`], page 7.

`empirical-quantile-probabilities` (*n*) [Function]

Probabilities that correspond to the empirical quantiles of a vector of length N. That is to say,

(== (quantiles sample (empirical-quantile-probabilities (length sample)))
sample)

for any vector SAMPLE.

Package [`nu.statistics`], page 13.

Source [`nu-statistics.lisp`], page 7.

`ensure-sorted-vector` (*object*) [Function]

Return the elements of OBJECT as a vector (or reals) sorted in ascending order.

Package [`nu.statistics`], page 13.

Source [`nu-statistics.lisp`], page 7.

`f-significance` (*f-statistic numerator-dof denominator-dof* **&optional** *one-tailed-p*) [Function]

Adopted from CLASP, but changed to handle F < 1 correctly in the one-tailed case. The 'f-statistic' must be a positive number. The degrees of freedom arguments must be positive integers. The 'one-tailed-p' argument is treated as a boolean.

Package [`lh.statistics`], page 11.

Source [`lh-statistics.lisp`], page 5.

`f-test` (*variance1 n1 variance2 n2* **&key** *tails*) [Function]

F test for the equality of two variances

Package [`lh.statistics`], page 11.

Source [`lh-statistics.lisp`], page 5.

`false-discovery-correction` (*p-values* **&key** *rate*) [Function]

A multiple testing correction that is less conservative than Bonferroni. Takes a list of p-values and a false discovery rate, and returns the number of p-values that are likely to be good enough to reject the null at that rate. Returns a second value which is the p-value cutoff.

Package [`lh.statistics`], page 11.

Source [`lh-statistics.lisp`], page 5.

`fisher-exact-test` (*contingency-table* **&key** *tails*) [Function]

Fisher's exact test. Gives a p value for a particular 2x2 contingency table

Package [`lh.statistics`], page 11.

Source [`lh-statistics.lisp`], page 5.

`fisher-z-transform` (*r*) [Function]

Transforms the correlation coefficient to an approximately normal distribution.

Package [`lh.statistics`], page 11.

Source [`lh-statistics.lisp`], page 5.

`fivenum` (*x* **&key** *tukey*) [Function]

By default, returns the five number summary (min, 1st quartile, median, 3rd quartile, max) of the elements X. If the keyword :tukey is set to a non-nil value, Tukey's fivenum summary is computed instead.

Package [`ls.statistics`], page 15.

Source [`ls-statistics.lisp`], page 9.

`geometric-mean` (*sequence* **&optional** *base*) [Function]

Returns the geometric mean of SEQUENCE
The geometric mean is a mean or average, which indicates the central tendency or typical value of a set of numbers by using the product of their values (as opposed to the arithmetic mean which uses their sum)

Package [`lh.statistics`], page 11.

Source [`lh-statistics.lisp`], page 5.

`interquartile-range` (*x*) [Function]

Returns the interquartile range of the elements of X.

Package [`ls.statistics`], page 15.

Source [`ls-statistics.lisp`], page 9.

`linear-regression` (*points*) [Function]

Computes the regression equation for a least squares fit of a line to a sequence of points (each a list of two numbers, e.g. '((1.0 0.1) (2.0 0.2))) and report the intercept, slope, correlation coefficient r, R^2, and the significance of the difference of the slope from 0.

Package [`lh.statistics`], page 11.

Source [`lh-statistics.lisp`], page 5.

`make-sparse-counter` (**&key** *test*) [Function]

Create a sparse counter. Elements are compared with TEST (should be accepted by HASH-TABLE).

Package [`nu.statistics`], page 13.

Source [`nu-statistics.lisp`], page 7.

`mcnemars-test` (*a-discordant-count b-discordant-count* **&key** *exact?*) [Function]

McNemar's test for correlated proportions, used for longitudinal studies. Look only at the number of discordant pairs (one treatment is effective and the other is not). If the two treatments are A and B, a-discordant-count is the number where A worked and B did not, and b-discordant-count is the number where B worked and A did not.

Package [`lh.statistics`], page 11.

Source [`lh-statistics.lisp`], page 5.

`mean` (*sequence*) [Function]

Returns the mean of SEQUENCE

Package [`lh.statistics`], page 11.

Source [`lh-statistics.lisp`], page 5.

`mean` (*object* **&key** *weights*) [Function]

Return the mean of OBJECT. OBJECT must be either a sequence of numbers, a sequence of BOOLEAN or a DISTRIBUTION object.

A sequence of BOOLEAN is converted to a BIT-VECTOR and the mean of it returned. This gives you the ratio of TRUE/FALSE values in the sequence (which is most often interpreted as a probability).

For samples (numeric-vectors), normalized by the weight-1 (and thus unbiased if certain assumptions hold, eg weights that count frequencies.

Package [`ls.statistics`], page 15.

Source [`ls-statistics.lisp`], page 9.

`mean-sd-n` (*sequence*) [Function]

A combined calculation that is often useful. Takes a sequence and returns three values: mean, standard deviation and N.

Package [`lh.statistics`], page 11.

Source [`lh-statistics.lisp`], page 5.

`median` (*sequence*) [Function]

Returns the median of SEQUENCE

Package [`lh.statistics`], page 11.

Source [`lh-statistics.lisp`], page 5.

`mode` (*sequence*) [Function]

Returns two values: a list of the modes and the number of times they occur

Package [`lh.statistics`], page 11.

Source [`lh-statistics.lisp`], page 5.

`normal-mean-ci` (*mean sd n alpha*) [Function]

Confidence interval for the mean of a normal distribution.

The 1-alpha percent confidence interval on the mean of a normal distribution with parameters mean, sd & n.

Package [`lh.statistics`], page 11.

Source [`lh-statistics.lisp`], page 5.

`normal-mean-ci-on-sequence` (*sequence alpha*) [Function]

The 1-alpha confidence interval on the mean of a sequence of numbers drawn from a Normal distribution.

Package [`lh.statistics`], page 11.

Source [`lh-statistics.lisp`], page 5.

`normal-pdf` (*x mu sigma*) [Function]

The probability density function (PDF) for a normal distribution with mean mu and variance sigma at point x.

Package [`lh.statistics`], page 11.

Source [`lh-statistics.lisp`], page 5.

`normal-sd-ci` (*sd n alpha*) [Function]

The 1-alpha confidence interval on the standard deviation of a sequence of numbers drawn from a Normal distribution.

Package [`lh.statistics`], page 11.

Source [`lh-statistics.lisp`], page 5.

`normal-sd-ci-on-sequence` (*sequence alpha*) [Function]

Package [`lh.statistics`], page 11.

Source [`lh-statistics.lisp`], page 5.

`normal-variance-ci` (*variance n alpha*) [Function]

The 1-alpha confidence interval on the variance of a sequence of numbers drawn from a Normal distribution.

Package [`lh.statistics`], page 11.

Source [`lh-statistics.lisp`], page 5.

`normal-variance-ci-on-sequence` (*sequence alpha*) [Function]

Package [`lh.statistics`], page 11.

Source [`lh-statistics.lisp`], page 5.

`percentile` (*sequence percent*) [Function]

Return an element from SEQUENCE at percentile PERCENT This function is also known as quantile.

Package [`lh.statistics`], page 11.

Source [`lh-statistics.lisp`], page 5.

`permutations` (*n k*) [Function]

How many ways to take n things taken k at a time, when order matters

Package [`lh.statistics`], page 11.

Source [`lh-statistics.lisp`], page 5.

`phi` (*x*) [Function]

the CDF of standard normal distribution

Package [`lh.statistics`], page 11.

Source [`lh-statistics.lisp`], page 5.

`poisson-cumulative-probability` (*mu k*) [Function]

Probability of seeing fewer than K events over a time period when the expected number events over that time is mu.

Package [`lh.statistics`], page 11.

Source [`lh-statistics.lisp`], page 5.

`poisson-ge-probability` (*mu x*) [Function]

Probability of X or more events when expected is mu.

Package [`lh.statistics`], page 11.

Source [`lh-statistics.lisp`], page 5.

`poisson-mu-ci` (*x alpha*) [Function]

Confidence interval for the Poisson parameter mu

Given x observations in a unit of time, what is the 1-alpha confidence interval on the Poisson parameter mu (= lambda*T)?

Since find-critical-value assumes that the function is monotonic increasing, adjust the value we are looking for taking advantage of reflectiveness

Package [`lh.statistics`], page 11.

Source [`lh-statistics.lisp`], page 5.

`poisson-probability` (*mu k*) [Function]

Probability of seeing k events over a time period when the expected number of events over that time is mu.

Package [`lh.statistics`], page 11.

Source [`lh-statistics.lisp`], page 5.

`poisson-test-one-sample` (*observed mu* **&key** *tails approximate?*) [Function]

The significance of a one sample test for the equality of an observed number of events (observed) and an expected number mu under the poisson distribution. Normal theory approximation is not that great, so don't use it unless told.

Package [`lh.statistics`], page 11.

Source [`lh-statistics.lisp`], page 5.

`pool` (**&rest** *accumulators*) [Function]

Pool ACCUMULATORS.

Package [`nu.statistics`], page 13.

Source [`nu-statistics.lisp`], page 7.

`random-normal` (**&key** *mean sd*) [Function]

Returns a random number with mean and standard-distribution as specified.

Package [`lh.statistics`], page 11.

Source [`lh-statistics.lisp`], page 5.

`random-pick` (*sequence*) [Function]

Random selection from sequence

Package [`lh.statistics`], page 11.

Source [`lh-statistics.lisp`], page 5.

`random-sample` (*n sequence*) [Function]

Return a random sample of size N from sequence, without replacement. If N is equal to or greater than the length of the sequence, return the entire sequence.

Package [`lh.statistics`], page 11.

Source [`lh-statistics.lisp`], page 5.

`round-float` (*x* **&key** *precision*) [Function]

Rounds a floating point number to a specified number of digits precision.

Package [`lh.statistics`], page 11.

Source [`lh-statistics.lisp`], page 5.

`sample-range` (*sequence*) [Function]

Return the difference between the largest and smallest values in SEQUENCE

Package [`lh.statistics`], page 11.

Source [`lh-statistics.lisp`], page 5.

`sd` (*sequence*) [Function]

Package [`lh.statistics`], page 11.

Source [`lh-statistics.lisp`], page 5.

`sign-test` (*plus-count minus-count* **&key** *exact? tails*) [Function]

Really just a special case of the binomial one sample test with p = 1/2. The normal theory version has a correction factor to make it a better approximation.

Package [`lh.statistics`], page 11.

Source [`lh-statistics.lisp`], page 5.

`sign-test-on-sequences` (*sequence1 sequence2* **&key** *exact? tails*) [Function]
Same as SIGN-TEST, but takes two sequences and tests whether the entries in one are different (greater or less) than the other.

Package [`lh.statistics`], page 11.

Source [`lh-statistics.lisp`], page 5.

`sorted-reals-elements` (*sorted-reals*) [Function]

Package [`nu.statistics`], page 13.

Source [`nu-statistics.lisp`], page 7.

`sparse-counter-count` (*sparse-counter object*) [Function]
Return the count for OBJECT.

Package [`nu.statistics`], page 13.

Source [`nu-statistics.lisp`], page 7.

`sparse-counter-table` (*instance*) [Reader]

Package [`nu.statistics`], page 13.

Source [`nu-statistics.lisp`], page 7.

Target Slot
[`table`], page 36.

`spearman-rank-correlation` (*points*) [Function]
Spearman rank correlation computes the relationship between a pair of variables when one or both are either ordinal or have a distribution that is far from normal. It takes a list of points (same format as linear-regression) and returns the spearman rank correlation coefficient and its significance.

Package [`lh.statistics`], page 11.

Source [`lh-statistics.lisp`], page 5.

`standard-deviation` (*sequence*) [Function]
Return the standard deviation of SEQUENCE

Package [`lh.statistics`], page 11.

Source [`lh-statistics.lisp`], page 5.

`standard-error-of-the-mean` (*sequence*) [Function]
Return the estimated standard deviation obtained from a set of sample means from repeated samples

Package [`lh.statistics`], page 11.

Source [`lh-statistics.lisp`], page 5.

`t-distribution` (*dof percentile*) [Function]
Returns the point which is the indicated percentile in the T distribution with dof degrees of freedom

Package [`lh.statistics`], page 11.

Source [`lh-statistics.lisp`], page 5.

`t-significance` (*t-statistic dof* **&key** *tails*) [Function]

Adopted from CLASP 1.4.3, http://eksl-www.cs.umass.edu/clasp.html

Package [`lh.statistics`], page 11.

Source [`lh-statistics.lisp`], page 5.

`t-test-one-sample` (*x-bar sd n mu* **&key** *tails*) [Function]

The significance of a one sample T test for the mean of a normal distribution with unknown variance. X-bar is the observed mean, sd is the observed standard deviation, N is the number of observations and mu is the test mean.

Package [`lh.statistics`], page 11.

Source [`lh-statistics.lisp`], page 5.

`t-test-one-sample-on-sequence` (*sequence mu* **&key** *tails*) [Function]

The significance of a one sample T test for the mean of a normal sequence of numbers with unknown variance. X-bar is the observed mean, sd is the observed standard deviation, N is the number of observations and mu is the test mean.

Package [`lh.statistics`], page 11.

Source [`lh-statistics.lisp`], page 5.

`t-test-one-sample-sse` (*mu mu-null variance* **&key** *alpha 1-beta tails*) [Function]

Returns the number of subjects needed to test whether the mean of a normally distributed sample mu is different from a null hypothesis mean mu-null and variance variance, with alpha, 1-beta and tails as specified.

Package [`lh.statistics`], page 11.

Source [`lh-statistics.lisp`], page 5.

`t-test-paired` (*d-bar sd n* **&key** *tails*) [Function]

The significance of a paired t test for the means of two normal distributions in a longitudinal study. D-bar is the mean difference, sd is the standard deviation of the differences, N is the number of pairs.

Package [`lh.statistics`], page 11.

Source [`lh-statistics.lisp`], page 5.

`t-test-paired-on-sequences` (*before after* **&key** *tails*) [Function]

The significance of a paired t test for means of two normal distributions in a longitudinal study. Before is a sequence of before values, after is the sequence of paired after values (which must be the same length as the before sequence).

Package [`lh.statistics`], page 11.

Source [`lh-statistics.lisp`], page 5.

`t-test-paired-sse` (*difference-mu difference-variance* **&key** *alpha 1-beta tails*) [Function]

Returns the number of subjects needed to test whether the differences with mean difference-mu and variance difference-variance, with alpha, 1-beta and tails as specified.

Package [`lh.statistics`], page 11.

Source [`lh-statistics.lisp`], page 5.

`t-test-two-sample` (*x-bar1 sd1 n1 x-bar2 sd2 n2* **&key** *variances-equal? variance-significance-cutoff tails*) [Function]

The significance of the difference of two means (x-bar1 and x-bar2) with standard deviations sd1 and sd2, and sample sizes n1 and n2 respectively. The form of the two sample t test depends on whether the sample variances are equal or not. If the variable variances-equal? is :test, then we use an F test and the variance-significance-cutoff to determine if they are equal. If the variances are equal, then we use the two sample t test for equal variances. If they are not equal, we use the Satterthwaite method, which has good type I error properties (at the loss of some power).

Package [`lh.statistics`], page 11.

Source [`lh-statistics.lisp`], page 5.

`t-test-two-sample-on-sequences` (*sequence1 sequence2* **&key** *variance-significance-cutoff tails*) [Function]

The significance of the difference of two means of SEQUENCE1 and SEQUENCE2 with standard deviations sd1 and sd2, and sample sizes n1 and n2 respectively. The form of the two sample t test depends on whether the sample variances are equal or not. If the variable variances-equal? is :test, then we use an F test and the variance-significance-cutoff to determine if they are equal. If the variances are equal, then we use the two sample t test for equal variances. If they are not equal, we use the Satterthwaite method.

Package [`lh.statistics`], page 11.

Source [`lh-statistics.lisp`], page 5.

`t-test-two-sample-sse` (*mu1 variance1 mu2 variance2* **&key** *sample-ratio alpha 1-beta tails*) [Function]

Returns the number of subjects needed to test whether the mean mu1 of a normally distributed sample (with variance variance1) is different from a second sample with mean mu2 and variance variance2, with alpha, 1-beta and tails as specified. It is also possible to set a sample size ratio of sample 1 to sample 2.

Package [`lh.statistics`], page 11.

Source [`lh-statistics.lisp`], page 5.

`tabulate` (*sequence* **&key** *test*) [Function]

Tabulate a sequence (using a SPARSE-COUNTER with the given TEST).

Package [`nu.statistics`], page 13.

Source [`nu-statistics.lisp`], page 7.

`variance` (*sequence*) [Function]

Return variance of SEQUENCE

Package [`lh.statistics`], page 11.

Source [`lh-statistics.lisp`], page 5.

`variance` (*object* **&key** *weights biased?*) [Function]

Variance of OBJECT. For samples, normalized by the weight-1 (and thus unbiased if certain assumptions hold, e.g. weights that count frequencies).

Note that alexandria's default for variance will return biased variance. We change that here for consistency. If you want a biased variance use alexandria:variance directly.

Package [`ls.statistics`], page 15.

Source [`ls-statistics.lisp`], page 9.

`weighted-quantiles` (*values weights qs*) [Function]

Calculate quantiles QS of weighted observations. Uses a 0.5 correction.

Package [`nu.statistics`], page 13.

Source [`nu-statistics.lisp`], page 7.

`wilcoxon-signed-rank-test` (*differences* **&optional** *tails*) [Function]

A test on the ranking of positive and negative differences (are the positive differences significantly larger/smaller than the negative ones). Assumes a continuous and symmetric distribution of differences, although not a normal one. This is the normal theory approximation, which is only valid when N > 15. This test is equivalent to the Mann-Whitney test.

Package [`lh.statistics`], page 11.

Source [`lh-statistics.lisp`], page 5.

`wilcoxon-signed-rank-test-on-sequences` (*sequence1 sequence2* **&optional** *tails*) [Function]

Package [`lh.statistics`], page 11.

Source [`lh-statistics.lisp`], page 5.

`z` (*percentile* **&key** *epsilon*) [Function]

The inverse normal function, P(X<Zu) = u where X is distributed as the standard normal. Uses binary search.(

Package [`lh.statistics`], page 11.

Source [`lh-statistics.lisp`], page 5.

`z-test` (*x-bar n* **&key** *mu sigma tails*) [Function]

The significance of a one sample Z test for the mean of a normal distribution with known variance. mu is the null hypothesis mean, x-bar is the observed mean, sigma is the standard deviation and N is the number of observations. If tails is :both, the significance of a difference between x-bar and mu. If tails is :positive, the significance of x-bar is greater than mu, and if tails is :negative, the significance of x-bar being less than mu. Returns a p value.

Package [`lh.statistics`], page 11.

Source [`lh-statistics.lisp`], page 5.

`z-test-on-sequence` (*sequence* **&key** *mu sigma tails*) [Function]

Package [`lh.statistics`], page 11.

Source [`lh-statistics.lisp`], page 5.

4.1.4 Generic functions

`add` (*accumulator object* **&key** *weight*) [Generic Function]

Add OBJECT to ACCUMULATOR. Return OBJECT. NILs are ignored by the accumulator, unless a specialized method decides otherwise. Keywords may be used to specify additional information (eg weight).

Package [`nu.statistics`], page 13.

Source [`nu-statistics.lisp`], page 7.

Methods

`add` ((*accumulator* [`sparse-counter`], *page 36*) *object* **&key** *weight*) [Method]

Increments the count of OBJECT in SPARSE-COUNTER, optionally with a weight

`add` ((*accumulator* [`sorted-reals`], *page 35*) *object* **&key**) [Method]

`add` ((*moments* [`central-sample-moments`], *page 34*) (*y* `real`) **&key** *weight*) [Method]

`add` (*accumulator* (*object* `null`) **&key**) [Method]

`central-m2` (*object* **&key** *weights*) [Generic Function]

Second central moment. For samples, normalized by the total weight (and thus not the unbiased estimator, see VARIANCE).

Package [`nu.statistics`], page 13.

Source [`nu-statistics.lisp`], page 7.

Methods

`central-m2` (*object* **&key** *weights*) [Method]

`central-m2` ((*object* [`central-sample-moments`], *page 34*) **&key** *weights*) [Method]

`central-m3` (*object* **&key** *weights*) [Generic Function]

Third central moment.

Package [`nu.statistics`], page 13.

Source [`nu-statistics.lisp`], page 7.

Methods

`central-m3` (*object* **&key** *weights*) [Method]

`central-m3` ((*object* [`central-sample-moments`], *page 34*) **&key** *weights*) [Method]

`central-m4` (*object* **&key** *weights*) [Generic Function]

Fourth central moment.

Package [`nu.statistics`], page 13.

Source [`nu-statistics.lisp`], page 7.

Methods

`central-m4` (*object* **&key** *weights*) [Method]

`central-m4` ((*object* [`central-sample-moments`], *page 34*) **&key** *weights*) [Method]

`central-sample-moments` (*object* **&key** *degree weights*) [Generic Function]

Return a CENTRAL-SAMPLE-MOMENTS object that allows the calculation of the central sample moments of OBJECT up to the given DEGREE.

When WEIGHTS are given, they need to be a sequence of matching length.

Package [`nu.statistics`], page 13.

Source [`nu-statistics.lisp`], page 7.

Methods

`central-sample-moments` ((*object* `null`) **&key** *degree weights*) [Method]

`central-sample-moments` ((*moments* [`central-sample-moments`], *page 34*) **&key** *degree weights*) [Method]

`central-sample-moments` ((*sequence* `sequence`) **&key** *degree weights*) [Method]

`ensure-sorted-reals` (*object*) [Generic Function]

Return the contents of OBJECT as a SORTED-REALS.

Package [`nu.statistics`], page 13.

Source [`nu-statistics.lisp`], page 7.

Methods

`ensure-sorted-reals` ((*sorted-reals* [`sorted-reals`], *page 35*)) [Method]

`ensure-sorted-reals` ((*array* `array`)) [Method]

`ensure-sorted-reals` ((*list* `list`)) [Method]

`kurtosis` (*object* **&key** *weights*) [Generic Function]

Kurtosis FIXME talk about bias, maybe implement unbiased?

Package [`nu.statistics`], page 13.

Source [`nu-statistics.lisp`], page 7.

Methods

`kurtosis` (*object* **&key** *weights*) [Method]

`kurtosis` ((*object* [`central-sample-moments`], *page 34*) **&key** *weights*) [Method]

`mean` (*object* **&key** *weights*) [Generic Function]

The mean of elements in OBJECT.

Package [`nu.statistics`], page 13.

Source [`nu-statistics.lisp`], page 7.

Methods

`mean` (*object* **&key** *weights*) [Method]

`mean` ((*object* [`central-sample-moments`], *page 34*) **&key** *weights*) [Method]

`median` (*object*) [Generic Function]

Median of OBJECT.

Package [`nu.statistics`], page 13.

Source [`nu-statistics.lisp`], page 7.

Methods

`median` ((*sample* `sequence`)) [Method]

Returns median of SAMPLE. SAMPLE must be a sequence of real numbers.

`median` (*object*) [Method]

`quantile` (*object q*) [Generic Function]

Return an element at quantile Q. May be an interpolation or an approximation, depending on OBJECT and Q. NOTE: Extensions should define methods for QUANTILES, not QUANTILE.

Package [`nu.statistics`], page 13.

Source [`nu-statistics.lisp`], page 7.

Methods

`quantile` ((*object* `sequence`) *q*) [Method]

`quantile` ((*object* `r-univariate`) *q*) [Method]

`quantile` (*object q*) [Method]

`quantiles` (*object qs*) [Generic Function]

Multiple quantiles (see QUANTILE). NOTE: Extensions should define methods for QUANTILES, not QUANTILE.

Package [`nu.statistics`], page 13.

Source [`nu-statistics.lisp`], page 7.

Methods

`quantiles` ((*object* `sequence`) *qs*) [Method]

`quantiles` ((*accumulator [*`sorted-reals`*], page 35*) *q*) [Method]

`sd` (*object* **&key** *weights*) [Generic Function]

Standard deviation. For samples, the square root of the unbiased estimator (see VARIANCE).

Package [`nu.statistics`], page 13.

Source [`nu-statistics.lisp`], page 7.

Methods

`sd` (*object* **&key** *weights*) [Method]

`skewness` (*object* **&key** *weights*) [Generic Function]

Skewness FIXME talk about bias, maybe implement unbiased?

Package [`nu.statistics`], page 13.

Source [`nu-statistics.lisp`], page 7.

Methods

`skewness` (*object* **&key** *weights*) [Method]

`skewness` ((*object [*`central-sample-moments`*], page 34*) **&key** *weights*) [Method]

`tally` (*accumulator*) [Generic Function]

The total weight of elements in ACCUMULATOR.

Package [`nu.statistics`], page 13.

Source [`nu-statistics.lisp`], page 7.

Methods

`tally` ((*accumulator [*`sparse-counter`*], page 36*)) [Method]

Return the total 'weight' of the accumulator `tally` ((*accumulator* [`tally-mixin`], *page 41*)) [Method]

`variance` (*object* **&key** *weights*) [Generic Function]

Variance of OBJECT. For samples, normalized by the weight-1 (and thus unbiased if certain assumptions hold, eg weights that count frequencies).

Package [`nu.statistics`], page 13.

Source [`nu-statistics.lisp`], page 7.

Methods

`variance` (*object* **&key** *weights*) [Method]

`variance` ((*object* [`central-sample-moments`], *page 34*) **&key** *weights*) [Method]

4.1.5 Standalone methods

`as-alist` ((*object* [`sparse-counter`], *page 36*)) [Method]

Return (OBJECT . COUNT) pairs as an alist.

Package `num-utils.utilities`.

Source [`nu-statistics.lisp`], page 7.

`num=` ((*a* [`central-sample-moments`], *page 34*) (*b* [`central-sample-moments`], *page 34*) **&optional** *tolerance*) [Method]

Package `num-utils.num=`.

Source [`nu-statistics.lisp`], page 7.

`print-object` ((*acc* [`sorted-reals`], *page 35*) *stream*) [Method]

Source [`nu-statistics.lisp`], page 7.

`print-object` ((*sparse-counter* [`sparse-counter`], *page 36*) *stream*) [Method]

Source [`nu-statistics.lisp`], page 7.

4.1.6 Conditions

`empty-accumulator` [Condition]

Package [`nu.statistics`], page 13.

Source [`nu-statistics.lisp`], page 7.

Direct superclasses

`error`.

`information-not-collected-in-accumulator` [Condition]

Package [`nu.statistics`], page 13.

Source [`nu-statistics.lisp`], page 7.

Direct superclasses

`error`.

`not-enough-elements-in-accumulator` [Condition]

Package [`nu.statistics`], page 13.

Source [`nu-statistics.lisp`], page 7.

Direct superclasses

`error`.

4.1.7 Structures

`central-sample-moments` [Structure]

Central sample moments calculated on-line/single-pass.

M weighted mean
S2 weighted sum of squared deviations from the mean, not calculated when NIL S3 weighted sum of cubed deviations from the mean, not calculated when NIL S4 weighted sum of 4th power deviations from the mean, not calculated when NIL

Allows on-line, numerically stable calculation of moments. See cite{bennett2009numerically} and cite{pebay2008formulas} for the description of the algorithm. M_2, ..., M_4 in the paper are s2, ..., s4 in the code.

Package [`nu.statistics`], page 13.

Source [`nu-statistics.lisp`], page 7.

Direct superclasses
[`tally-mixin`], page 41.

Direct methods
- [`add`], page 30.
- [`central-m2`], page 30.
- [`central-m3`], page 30.
- [`central-m4`], page 30.
- [`central-sample-moments`], page 31.
- [`kurtosis`], page 31.
- [`mean`], page 31.
- [`num=`], page 33.
- [`pool2`], page 40.
- [`skewness`], page 32.
- [`variance`], page 33.

Direct slots

`m` [Slot]

Type `real`

Initform `0.0d0`

Readers [`central-sample-moments-m`], page 37.

Writers [`(setf central-sample-moments-m)`], page 37.

`s2` [Slot]

Type `(or (real 0) null)`

Initform `0.0d0`

Readers [`central-sample-moments-s2`], page 37.

Writers [`(setf central-sample-moments-s2)`], page 37.

`s3` [Slot]

Type `(or real null)`

Initform `0.0d0`

Readers [central-sample-moments-s3], page 37.

Writers [(setf central-sample-moments-s3)], page 37.

`s4` [Slot]

Type `(or (real 0) null)`

Initform `0.0d0`

Readers [central-sample-moments-s4], page 37.

Writers [(setf central-sample-moments-s4)], page 37.

`sorted-reals` [Structure]

Accumulator which sorts elements. ELEMENTS return the sorted elements.

Package [nu.statistics], page 13.

Source [nu-statistics.lisp], page 7.

Direct superclasses
`structure-object`.

Direct methods
- [add], page 30.
- [ensure-sorted-reals], page 31.
- [print-object], page 33.
- [quantiles], page 32.

Direct slots

`ordered-elements` [Slot]

Type `vector`

Initform `#()`

Readers [sorted-reals-ordered-elements], page 39.

Writers [(setf sorted-reals-ordered-elements)], page 39.

`unordered-elements` [Slot]

Type `list`

Readers [sorted-reals-unordered-elements], page 40.

Writers [(setf sorted-reals-unordered-elements)], page 40.

`sparse-counter` [Structure]

Package [nu.statistics], page 13.

Source [nu-statistics.lisp], page 7.

Direct superclasses
`structure-object`.

Direct methods
- [add], page 30.
- [as-alist], page 33.
- [print-object], page 33.
- [tally], page 33.

Direct slots

`table` [Slot]

Type `hash-table`

Readers [`sparse-counter-table`], page 26.

Writers *This slot is read-only.*

4.2 Internals

4.2.1 Macros

`&sorted-reals` (*ordered-elements unordered-elements*) [Macro]

LET+ form for slots of the structure SORTED-REALS.

Package [`nu.statistics`], page 13.

Source [`nu-statistics.lisp`], page 7.

`&sorted-reals-r/o` (*ordered-elements unordered-elements*) [Macro]

LET+ form for slots of the structure SORTED-REALS. Read-only.

Package [`nu.statistics`], page 13.

Source [`nu-statistics.lisp`], page 7.

`define-central-sample-moment` (*function* (*variable degree*) **&body** *body*) [Macro]

FIXME documentation, factor out general part

Package [`nu.statistics`], page 13.

Source [`nu-statistics.lisp`], page 7.

`underflow-goes-to-zero` (**&body** *body*) [Macro]

Protects against floating point underflow errors and sets the value to 0.0 instead.

Package [`lh.statistics`], page 11.

Source [`lh-statistics.lisp`], page 5.

4.2.2 Ordinary functions

`average-rank` (*value sorted-values*) [Function]

Average rank calculation for non-parametric tests. Ranks are 1 based, but lisp is 0 based, so add 1!

Package [`lh.statistics`], page 11.

Source [`lh-statistics.lisp`], page 5.

`beta-incomplete` (*a b x*) [Function]

Adopted from CLASP 1.4.3, http://eksl-www.cs.umass.edu/clasp.html

Package [`lh.statistics`], page 11.

Source [`lh-statistics.lisp`], page 5.

`binomial-le-probability` (*n k p*) [Function]

Package [`lh.statistics`], page 11.

Source [`lh-statistics.lisp`], page 5.

`central-sample-moments-m` (*instance*) [Reader]
`(setf central-sample-moments-m)` (*instance*) [Writer]

Package [`nu.statistics`], page 13.

Source [`nu-statistics.lisp`], page 7.

Target Slot
[`m`], page 34.

`central-sample-moments-p` (*object*) [Function]

Package [`nu.statistics`], page 13.

Source [`nu-statistics.lisp`], page 7.

`central-sample-moments-s2` (*instance*) [Reader]
`(setf central-sample-moments-s2)` (*instance*) [Writer]

Package [`nu.statistics`], page 13.

Source [`nu-statistics.lisp`], page 7.

Target Slot
[`s2`], page 34.

`central-sample-moments-s3` (*instance*) [Reader]
`(setf central-sample-moments-s3)` (*instance*) [Writer]

Package [`nu.statistics`], page 13.

Source [`nu-statistics.lisp`], page 7.

Target Slot
[`s3`], page 35.

`central-sample-moments-s4` (*instance*) [Reader]
`(setf central-sample-moments-s4)` (*instance*) [Writer]

Package [`nu.statistics`], page 13.

Source [`nu-statistics.lisp`], page 7.

Target Slot
[`s4`], page 35.

`central-sample-moments-w` (*instance*) [Function]

Package [`nu.statistics`], page 13.

Source [`nu-statistics.lisp`], page 7.

`(setf central-sample-moments-w)` (*instance*) [Function]

Package [`nu.statistics`], page 13.

Source [`nu-statistics.lisp`], page 7.

`copy-central-sample-moments` (*instance*) [Function]

Package [`nu.statistics`], page 13.

Source [`nu-statistics.lisp`], page 7.

`copy-sorted-reals` (*instance*) [Function]

Package [`nu.statistics`], page 13.

Source [`nu-statistics.lisp`], page 7.

`copy-sparse-counter` (*instance*) [Function]

Package [`nu.statistics`], page 13.

Source [`nu-statistics.lisp`], page 7.

`copy-tally-mixin` (*instance*) [Function]

Package [`nu.statistics`], page 13.

Source [`nu-statistics.lisp`], page 7.

`error-function` (*x*) [Function]

Adopted from CLASP 1.4.3, http://eksl-www.cs.umass.edu/clasp.html

Package [`lh.statistics`], page 11.

Source [`lh-statistics.lisp`], page 5.

`error-function-complement` (*x*) [Function]

Adopted from CLASP 1.4.3, http://eksl-www.cs.umass.edu/clasp.html

Package [`lh.statistics`], page 11.

Source [`lh-statistics.lisp`], page 5.

`factorial` (*number*) [Function]

Package [`lh.statistics`], page 11.

Source [`lh-statistics.lisp`], page 5.

`find-critical-value` (*p-function p-value* **&optional** *x-tolerance y-tolerance*) [Function]

Adopted from CLASP 1.4.3, http://eksl-www.cs.umass.edu/clasp.html

Package [`lh.statistics`], page 11.

Source [`lh-statistics.lisp`], page 5.

`gamma-incomplete` (*a x*) [Function]

Adopted from CLASP 1.4.3, http://eksl-www.cs.umass.edu/clasp.html

Package [`lh.statistics`], page 11.

Source [`lh-statistics.lisp`], page 5.

`gamma-ln` (*x*) [Function]

Adopted from CLASP 1.4.3, http://eksl-www.cs.umass.edu/clasp.html

Package [`lh.statistics`], page 11.

Source [`lh-statistics.lisp`], page 5.

`make-central-sample-moments` (**&key** *w m s2 s3 s4*) [Function]

Package [`nu.statistics`], page 13.

Source [`nu-statistics.lisp`], page 7.

`make-sorted-reals` (**&key** *ordered-elements unordered-elements*) [Function]

Package [`nu.statistics`], page 13.

Source [`nu-statistics.lisp`], page 7.

`make-sparse-counter%` (**&key** *table*) [Function]

Package [`nu.statistics`], page 13.

Source [`nu-statistics.lisp`], page 7.

`make-tally-mixin` (**&key** *w*) [Function]

Package [`nu.statistics`], page 13.

Source [`nu-statistics.lisp`], page 7.

`round-up` (*x*) [Function]

Package [`lh.statistics`], page 11.

Source [`lh-statistics.lisp`], page 5.

`safe-exp` (*x*) [Function]

Eliminates floating point underflow for the exponential function. Instead, it just returns 0.0d0

Package [`lh.statistics`], page 11.

Source [`lh-statistics.lisp`], page 5.

`sign` (*x*) [Function]

Package [`lh.statistics`], page 11.

Source [`lh-statistics.lisp`], page 5.

`sort-reals` (*sequence*) [Function]

Return a SORTED-REALS structure.

Package [`nu.statistics`], page 13.

Source [`nu-statistics.lisp`], page 7.

`sorted-reals-ordered-elements` (*instance*) [Reader]
`(setf sorted-reals-ordered-elements)` (*instance*) [Writer]

Package [`nu.statistics`], page 13.

Source [`nu-statistics.lisp`], page 7.

Target Slot
[`ordered-elements`], page 35.

`sorted-reals-p` (*object*) [Function]

Package [`nu.statistics`], page 13.

Source [`nu-statistics.lisp`], page 7.

`sorted-reals-unordered-elements` (*instance*) [Reader]
`(setf sorted-reals-unordered-elements)` (*instance*) [Writer]

Package [`nu.statistics`], page 13.

Source [`nu-statistics.lisp`], page 7.

Target Slot
[`unordered-elements`], page 35.

`sparse-counter-p` (*object*) [Function]

Package [`nu.statistics`], page 13.

Source [`nu-statistics.lisp`], page 7.

`tally-mixin-p` (*object*) [Function]

Package [`nu.statistics`], page 13.

Source [`nu-statistics.lisp`], page 7.

`tally-mixin-w` (*instance*) [Reader]
`(setf tally-mixin-w)` (*instance*) [Writer]

Package [`nu.statistics`], page 13.

Source [`nu-statistics.lisp`], page 7.

Target Slot
[`w`], page 41.

`weighted-empirical-quantile` (*sorted-reals p-table q*) [Function]
Return the empirical quantile of a vector of real numbers, sorted in ascending order (not checked). Uses a 0.5 correction.

Package [`nu.statistics`], page 13.

Source [`nu-statistics.lisp`], page 7.

`weighted-quantile-p-table` (*weights*) [Function]
Return table of probability brackets for weighted quantile calculations., built from the weights (which should be positive reals, not checked). Uses a 0.5 correction.

Package [`nu.statistics`], page 13.

Source [`nu-statistics.lisp`], page 7.

4.2.3 Generic functions

`pool2` (*accumulator1 accumulator2*) [Generic Function]
Pool two accumulators. When they are of a different type, the resulting accumulator will be downgraded to the level afforded by the information available in the accumulators.

Package [`nu.statistics`], page 13.

Source [`nu-statistics.lisp`], page 7.

Methods

`pool2` ((*moments-a [*`central-sample-moments`*], page 34*) (*moments-b [*`central-sample-moments`*], page 34*)) [Method]

4.2.4 Structures

`tally-mixin` [Structure]
Mixin structure that contains a tally. Not exported. W is the total weight.

Package [`nu.statistics`], page 13.

Source [`nu-statistics.lisp`], page 7.

Direct superclasses
`structure-object`.

Direct subclasses
[`central-sample-moments`], page 34.

Direct methods
[`tally`], page 33.

Direct slots

`w` [Slot]

Type	`(real 0)`
Initform	`0`
Readers	[`tally-mixin-w`], page 40.
Writers	[`(setf tally-mixin-w)`], page 40.

Appendix A Indexes

A.1 Concepts

(Index is nonexistent)