## CONTENTS

1 Usage ................................................. 1
   1.1 Getting the Library ..................................... 1

2 The STATISTICS library ........................................ 3
   2.1 The STATISTICS module ..................................... 3

3 Copyright ........................................... 15

API Index ............................................ 17

Index ............................................... 19
CHAPTER ONE

USAGE

1.1 Getting the Library

The statistics libraries is available from GitHub.
The STATISTICS library provides some basic statistical functions along with optimized implementations.

Note: Currently, the statistical functions are only available for limited vectors of <double-float> values. This is expected to change in the future.

2.1 The STATISTICS module

2.1.1 Types

<double-float-vector> Type
A <vector> that only contains <double-float> values.

Equivalent limited(<vector>, of: <double-float>)

Discussion
A <vector> that only contains <double-float> values.

This type is used for implementations of statistical functions which are specialized for <double-float> values.

See also
• <double-float?-vector>
• <numeric-sequence>

<double-float?-vector> Type
A <vector> that contains values that are either <double-float> or #f.
Equivalent limited(<vector>, of: false-or(<double-float>))

Discussion

A <vector> that contains values that are either <double-float> or #f.
This type is used for implementations of statistical functions which may need to handle missing data. By using a separate type from <double-float-vector>, the implementation can limit any overhead from handling missing values to only being applied where it is needed.

Note: Implementations of the statistical functions which handle missing data have not yet been provided.

See also

- <double-float-vector>
- <numeric-sequence>

<numeric-sequence> Type

Equivalent type-union(<double-float-vector>, <double-float?-vector>)

See also

- <double-float-vector>
- <double-float?-vector>

2.1.2 Coercion Functions

double-float-vector Function
Utility function for converting a sequence that contains only <double-float> values to a <double-float-vector> for use with the optimized implementations of the basic statistical functions.

Signature double-float-vector (seq) => (vec)

Parameters

- seq – An instance of <sequence>.

Values

- vec – An instance of <double-float-vector>.

Example

let dv = double-float-vector(#[1.0d0, 2.0d0, 3.0d0]);

2.1.3 Extrema

maximum Open Generic function
Returns the maximum value from a numeric sequence.

Signature maximum (sample) => (maximum)

Parameters

- sample – An instance of <numeric-sequence>.

Values
• **maximum** – An instance of `<number>`.

**Example** Assuming that `dv` contains the values `[1.0d0, -1.0d0, 2.0d0]`:

```lisp
? maximum(dv)
=> 2.0d0
```

**See also**

• `maximum/trimmed`
• `minimum`
• `minimum/trimmed`
• `minimum+maximum`

### `maximum(<double-float-vector>)` Sealed Method

A specialized implementation of `maximum` for `<double-float>`.

**Parameters**

• `sample` – An instance of `<double-float-vector>`.

**Values**

• `maximum` – An instance of `<double-float>`.

### `maximum/trimmed` Open Generic function

Returns the maximum value from a numeric sequence that is below (or optionally equal to) an upper limit.

**Signature** `maximum/trimmed (sample upper-limit #key inclusive?) => (maximum)`

**Parameters**

• `sample` – An instance of `<numeric-sequence>`.

• `upper-limit` – An instance of `<number>`.

• `inclusive? (#key)` – An instance of `<boolean>`. Default value is `#t`.

**Values**

• `maximum` – An instance of `<number>`.

**Discussion**

Returns the maximum value from a numeric sequence that is below (or optionally equal to) an upper limit.

If `inclusive?` is true (the default), then values equal to the `upper-limit` are included when calculating the maximum value.

**Example** Assuming that `dv` contains the values `[1.0d0, 2.0d0, 3.0d0, 4.0d0]`:

```lisp
? maximum/trimmed(dv, 3.0d0, inclusive?: #t)
=> 3.0d0

? maximum/trimmed(dv, 3.0d0, inclusive?: #f)
=> 2.0d0
```

**See also**

• `maximum`
• `minimum`
• minimum/trimmed
• minimum+maximum

maximum/trimmed(<double-float-vector>, <double-float>) Sealed Method
A specialized implementation of maximum/trimmed for <double-float>.

Parameters
- sample – An instance of <double-float-vector>.
- upper-limit – An instance of <double-float>.
- inclusive? (#key) – An instance of <boolean>.

Values
- maximum – An instance of <double-float>.

minimum Open Generic function
Returns the minimum value from a numeric sequence.

Signature  minimum (sample) => (minimum)

Parameters
- sample – An instance of <numeric-sequence>.

Values
- minimum – An instance of <number>.

Example  Assuming that dv contains the values #[1.0d0, -1.0d0, 2.0d0]:

? minimum(dv)
=> -1.0d0

See also
- maximum
- maximum/trimmed
- minimum/trimmed
- minimum+maximum

minimum(<double-float-vector>) Sealed Method
A specialized implementation of minimum for <double-float>.

Parameters
- sample – An instance of <double-float-vector>.

Values
- minimum – An instance of <double-float>.

minimum/trimmed Open Generic function
Returns the minimum value from a numeric sequence that is over (or optionally equal to) a lower-limit.

Signature  minimum/trimmed (sample lower-limit #key inclusive?) => (minimum)

Parameters
- sample – An instance of <numeric-sequence>.
- lower-limit – An instance of <number>. 
• **inclusive?** (#key) – An instance of `<boolean>`.

**Values**

• **minimum** – An instance of `<number>`.

**Discussion**

Returns the minimum value from a numeric sequence that is over (or optionally equal to) a lower-limit.

If `inclusive?` is true (the default), then values equal to the lower-limit are included when calculating the minimum value.

**Example** Assuming that `dv` contains the values `[1.0d0, 2.0d0, 3.0d0, 4.0d0]`:

```plaintext
? minimum/trimmed(dv, 2.0d0, inclusive?: #t)
=> 2.0d0

? minimum/trimmed(dv, 2.0d0, inclusive?: #f)
=> 3.0d0
```

**See also**

• `maximum`
• `maximum/trimmed`
• `minimum`
• `minimum+maximum`

**minimum/trimmed** (<double-float-vector>, <double-float>) Sealed Method

A specialized implementation of `minimum/trimmed` for `<double-float>`.

**Parameters**

• **sample** – An instance of `<double-float-vector>`.
• **lower-limit** – An instance of `<double-float>`.
• **inclusive?** (#key) – An instance of `<boolean>`.

**Values**

• **minimum** – An instance of `<double-float>`.

**minimum+maximum** Open Generic function

Returns both the minimum and maximum values within a numeric sequence.

**Signature** `minimum+maximum (sample) => (minimum maximum)`

**Parameters**

• **sample** – An instance of `<numeric-sequence>`.

**Values**

• **minimum** – An instance of `<number>`.
• **maximum** – An instance of `<number>`.

**Example** Assuming that `dv` contains the values `[1.0d0, -1.0d0, 2.0d0]`:

```plaintext
? minimum+maximum(dv)
=> values(-1.0d0, 2.0d0)
```
See also

• maximum
• maximum/trimmed
• minimum
• minimum/trimmed

minimum+maximum(<double-float-vector>) Sealed Method
A specialized implementation of minimum+maximum for <double-float>.

Parameters
• sample – An instance of <double-float-vector>.

Values
• minimum – An instance of <double-float>.
• maximum – An instance of <double-float>.

2.1.4 Means

mean/arithmetic Open Generic function
Returns the arithmetic mean of a numeric sequence.

Signature mean/arithmetic (sample) => (mean)

Parameters
• sample – An instance of <numeric-sequence>.

Values
• mean – An instance of <number>.

Discussion
Returns the arithmetic mean of a numeric sequence.

Commonly known as just ‘mean’ or ‘average’, the arithmetic mean is the sum of the values of
the sequence, divided by the number of values in the sequence. It is distinct from other ways of
calculating a mean such as those provided by mean/geometric and mean/harmonic.

A simple (and slightly faster) naive implementation of the arithmetic mean is subject to nu-
merical inaccuracy. This implementation follows the method presented by Knuth in The Art of

Equivalent The arithmetic mean is given by:

\[ \frac{1}{n} \sum_{i=1}^{n} x_i \]

Our implementation is computed as follows:

\[ m_1 = x_1 \]
\[ m_k = m_{k-1} + \frac{x_k - m_{k-1}}{k} \]

Example Assuming that dv contains the values #\{1.0d0, 2.0d0, 8.0d0, 9.0d0\}:
See also

- `mean/fast`
- `mean/geometric`
- `mean/harmonic`
- `standard-scores`
- Arithmetic Mean on Wikipedia

### `mean/arithmetic(<double-float-vector>)` Sealed Method

A specialized implementation of `mean/arithmetic` for `<double-float>`.

**Parameters**

- `sample` – An instance of `<double-float-vector>`.

**Values**

- `mean` – An instance of `<double-float>`.

### `mean/fast` Open Generic function

Returns the arithmetic mean of a numeric sequence.

**Signature**

`mean/fast (sample) => (mean)`

**Parameters**

- `sample` – An instance of `<numeric-sequence>`.

**Values**

- `mean` – An instance of `<number>`.

**Discussion**

Returns the arithmetic mean of a numeric sequence.

This differs from `mean/arithmetic` by using a naive algorithm that is slightly faster, but subject to numerical inaccuracy. You should only use this function if you’re aware of the risks.

**Equivalent**

\[
\frac{1}{n} \sum_{i=1}^{n} x_i
\]

**Example**

Assuming that `dv` contains the values `[1.0d0, 2.0d0, 8.0d0, 9.0d0]`:

```
? mean/arithmetic(dv) => 5.25d0
```

See also

- `mean/arithmetic`
- `mean/geometric`
- `mean/harmonic`

### `mean/fast (<double-float-vector>)` Sealed Method

A specialized implementation of `mean/fast` for `<double-float>`.

**Parameters**

- `sample` – An instance of `<double-float-vector>`. 
Values

- mean – An instance of <double-float>.

**mean/geometric** Open Generic function
Returns the geometric mean of a numeric sequence.

**Signature** mean/geometric (sample) => (mean)

**Parameters**
- sample – An instance of <numeric-sequence>.

**Values**
- mean – An instance of <number>.

**Discussion**
Returns the geometric mean of a numeric sequence.

For greater numerical accuracy, our implementation is based on the exponentiation of the arithmetic mean of the natural logarithm of each value in sample.

**Equivalent** The geometric mean is given by:
\[
\left( \prod_{i=1}^{n} a_i \right)^{1/n}
\]

Our implementation is computed as follows:
\[
\exp \left[ \frac{1}{n} \sum_{i=1}^{n} \ln a_i \right]
\]

**Example** Assuming that \(dv\) contains the values \([2.0d0, 4.0d0, 8.0d0]\):

```lisp
? mean/geometric(dv)
=> 4.0d0
```

**See also**
- mean/arithmetic
- mean/fast
- mean/harmonic
- Geometric Mean on Wikipedia

**mean/geometric(<double-float-vector>)** Sealed Method
A specialized implementation of mean/geometric for <double-float>.

**Parameters**
- sample – An instance of <double-float-vector>.

**Values**
- mean – An instance of <double-float>.

**mean/harmonic** Open Generic function
Returns the harmonic mean of a numeric sequence.

**Signature** mean/harmonic (sample) => (mean)
Parameters

• **sample** – An instance of `<numeric-sequence>.

Values

• **mean** – An instance of `<number>.

Discussion

Returns the harmonic mean of a numeric sequence.

The harmonic mean is the reciprocal of the arithmetic mean of the reciprocals of the values of the sequence.

Equivalent

The harmonic mean is given by:

\[
\sum_{i=1}^{n} \frac{1}{x_i}
\]

See also

• `mean/arithetic`
• `mean/fast`
• `mean/geometric`
• Harmonic Mean on Wikipedia

**mean/harmonic(<double-float-vector>)** Sealed Method
A specialized implementation of `mean/harmonic` for `<double-float>.

Parameters

• **sample** – An instance of `<double-float-vector>.

Values

• **mean** – An instance of `<double-float>.

---

### 2.1.5 Scaling

**scale** Open Generic function

**Signature**

\[ \text{scale} \ (\text{sample} \ \text{lower-bound} \ \text{upper-bound}) \ \Rightarrow \ (\text{res}) \]

**Parameters**

• **sample** – An instance of `<numeric-sequence>.
• **lower-bound** – An instance of `<number>.
• **upper-bound** – An instance of `<number>.

**Values**

• **res** – An instance of `<numeric-sequence>.

**scale(<double-float-vector>, <double-float>, <double-float>)** Sealed Method
A specialized implementation of `scale` for `<double-float>.

**Parameters**

• **sample** – An instance of `<double-float-vector>.
• **lower-bound** – An instance of `<double-float>.
• **upper-bound** – An instance of `<double-float>`.

**Values**

• **res** – An instance of `<double-float-vector>`.

### 2.1.6 Variance and Deviation

**standard-deviation/population** Open Generic function

**Signature** standard-deviation/population (sample) => (standard-deviation)

**Parameters**

• **sample** – An instance of `<numeric-sequence>`.

**Values**

• **standard-deviation** – An instance of `<number>`.

**See also**

• variance/population
  • variance/sample
  • standard-deviation/sample
  • standard-scores

**standard-deviation/population (<double-float-vector>)** Sealed Method

A specialized implementation of `standard-deviation/population` for `<double-float>`.

**Parameters**

• **sample** – An instance of `<double-float-vector>`.

**Values**

• **standard-deviation** – An instance of `<double-float>`.

**standard-deviation/sample** Open Generic function

**Signature** standard-deviation/sample (sample) => (standard-deviation)

**Parameters**

• **sample** – An instance of `<numeric-sequence>`.

**Values**

• **standard-deviation** – An instance of `<number>`.

**Discussion** The standard-deviation calculation for a sample, rather than a complete population, uses `sample.size - 1` rather than the sample size. This is Bessel’s Correction.

**See also**

• variance/population
  • variance/sample
  • standard-deviation/population

**standard-deviation/sample (<double-float-vector>)** Sealed Method

A specialized implementation of `standard-deviation/sample` for `<double-float>`.

**Parameters**
• **sample** – An instance of `<double-float-vector>`.

Values

• **standard-deviation** – An instance of `<double-float>`.

**variance/population** Open Generic function

Signature variance/population (sample) => (variance)

Parameters

• **sample** – An instance of `<numeric-sequence>`.

Values

• **variance** – An instance of `<number>`.

See also

• variance/sample
  • standard-deviation/population
  • standard-deviation/sample

**variance/population(<double-float-vector>)** Sealed Method

A specialized implementation of variance/population for `<double-float>`.

Parameters

• **sample** – An instance of `<double-float-vector>`.

Values

• **variance** – An instance of `<double-float>`.

**variance/sample** Open Generic function

Signature variance/sample (sample) => (variance)

Parameters

• **sample** – An instance of `<numeric-sequence>`.

Values

• **variance** – An instance of `<number>`.

See also

• variance/population
  • standard-deviation/population
  • standard-deviation/sample

**variance/sample(<double-float-vector>)** Sealed Method

A specialized implementation of variance/sample for `<double-float>`.

Parameters

• **sample** – An instance of `<double-float-vector>`.

Values

• **variance** – An instance of `<double-float>`.

**standard-scores** Open Generic function

Signature standard-scores (population) => (scores)
Parameters

- **population** – An instance of `<numeric-sequence>`.

Values

- **scores** – An instance of `<numeric-sequence>`.

Equivalent

The standard score of a value in a sequence is given by:

\[ z = \frac{X - \mu}{\sigma} \]

Where:

- \( \mu \) is the mean of the population
- \( \sigma \) is the standard deviation of the population

See also

- `mean/arithmetic`
- `standard-deviation/population`

**standard-scores (<double-float-vector>) Sealed Method**

A specialized implementation of `standard-scores` for `<double-float>`.

Parameters

- **population** – An instance of `<double-float-vector>`.

Values

- **scores** – An instance of `<double-float-vector>`.
Copyright (c) 2015 Bruce Mitchener, Jr.

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the “Software”), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED “AS IS”, WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
### D
- `<double-float-vector>` *(type)*, 3
- `<double-float?-vector>` *(type)*, 3
- `double-float-vector` *(function)*, 4

### M
- `statistics:statistics:maximum` *(generic function)*, 4
- `statistics:statistics:maximum/trimmed` *(generic function)*, 5
- `statistics:statistics:mean/arithmetic` *(generic function)*, 8
- `statistics:statistics:mean/fast` *(generic function)*, 9
- `statistics:statistics:mean/geometric` *(generic function)*, 10
- `statistics:statistics:mean/geometric` *(<double-float-vector>)* *(method)*, 10
- `statistics:statistics:mean/harmonic` *(generic function)*, 10
- `statistics:statistics:minimum` *(generic function)*, 6
- `statistics:statistics:minimum` *(<double-float-vector>)* *(method)*, 6
- `statistics:statistics:minimum+maximum` *(generic function)*, 7
- `statistics:statistics:minimum+maximum` *(<double-float-vector>)* *(method)*, 8
- `statistics:statistics:minimum/trimmed` *(generic function)*, 6

### N
- `<numeric-sequence>` *(type)*, 4

### S
- `statistics:statistics:scale` *(generic function)*, 11
- `statistics:statistics:scale` *(<double-float-vector>)* *(method)*, 11
- `statistics:statistics:standard-deviation/population` *(generic function)*, 12
- `statistics:statistics:standard-deviation/sample` *(generic function)*, 12
- `statistics:statistics:standard-scores` *(generic function)*, 13

### V
- `statistics:statistics:variance/population` *(generic function)*, 13
- `statistics:statistics:variance/sample` *(generic function)*, 13
INDEX

Symbols
<double-float-vector>, 3
<double-float?-vector>, 3
<num-numeric-sequence>, 4

A
average, 8

D
double-float-vector, 4

M
maximum, 4
  maximum(<double-float-vector>), 5
maximum/trimmed, 5
  maximum/trimmed(<double-float-vector>), <double-float>, 6
mean, 8
mean/arithmetic, 8
  mean/arithmetic(<double-float-vector>), 9
mean/fast, 9
  mean/fast(<double-float-vector>), 9
mean/geometric, 10
  mean/geometric(<double-float-vector>), 10
mean/harmonic, 10
  mean/harmonic(<double-float-vector>), 11
minimum, 6
  minimum(<double-float-vector>), 6
minimum+maximum, 7
  minimum+maximum(<double-float-vector>), 8
minimum/trimmed, 6
  minimum/trimmed(<double-float-vector>), <double-float>, 7

S
cscale, 11
  scale(<double-float-vector>, <double-float>, <double-float>), 11
standard-deviation/population, 12
  standard-deviation/population(<double-float-vector>), 12
standard-deviation/sample, 12

V
variance/population, 13
  variance/population(<double-float-vector>), 13
variance/sample, 13
  variance/sample(<double-float-vector>), 13

Z
z-scores, 13