

# The DISTRIBUTIONS Reference Manual

---

Statistical distributions and related functions, version 1.0.0

Steven Nunez <steve@symbolics.tech>

---

This manual was generated automatically by Declt 4.0b2.

Copyright © 2019-2022 Steven Nunez

Permission is granted to make and distribute verbatim copies of this manual provided the copyright notice and this permission notice are preserved on all copies.

Permission is granted to copy and distribute modified versions of this manual under the conditions for verbatim copying, provided also that the section entitled “Copying” is included exactly as in the original.

Permission is granted to copy and distribute translations of this manual into another language, under the above conditions for modified versions, except that this permission notice may be translated as well.

# Table of Contents

<b>Copying</b>	<b>1</b>
<b>1 Systems</b>	<b>3</b>
1.1 distributions	3
<b>2 Files</b>	<b>5</b>
2.1 Lisp	5
2.1.1 distributions/distributions.asd	5
2.1.2 distributions/packages.lisp	5
2.1.3 distributions/internals.lisp	5
2.1.4 distributions/generator.lisp	5
2.1.5 distributions/simple-multiplicative-congruential-generators.lisp	6
2.1.6 distributions/defs.lisp	7
2.1.7 distributions/generics.lisp	7
2.1.8 distributions/discrete.lisp	7
2.1.9 distributions/uniform.lisp	8
2.1.10 distributions/exponential.lisp	9
2.1.11 distributions/normal.lisp	9
2.1.12 distributions/log-normal.lisp	10
2.1.13 distributions/truncated-normal.lisp	11
2.1.14 distributions/t-distribution.lisp	12
2.1.15 distributions/gamma.lisp	12
2.1.16 distributions/chi-square.lisp	13
2.1.17 distributions/beta.lisp	14
2.1.18 distributions/rayleigh.lisp	14
2.1.19 distributions/bernoulli.lisp	15
2.1.20 distributions/binomial.lisp	15
2.1.21 distributions/geometric.lisp	16
2.1.22 distributions/poisson.lisp	16
<b>3 Packages</b>	<b>17</b>
3.1 distributions.internals	17
3.2 distributions	17
<b>4 Definitions</b>	<b>23</b>
4.1 Public Interface	23
4.1.1 Macros	23
4.1.2 Ordinary functions	23
4.1.3 Generic functions	27
4.1.4 Standalone methods	34
4.1.5 Structures	34
4.1.6 Classes	42
4.1.7 Types	44
4.2 Internals	44
4.2.1 Constants	44
4.2.2 Special variables	44

4.2.3	Macros .....	44
4.2.4	Ordinary functions .....	45
4.2.5	Generic functions .....	56
4.2.6	Structures .....	59
4.2.7	Classes .....	61
<b>Appendix A Indexes .....</b>		<b>63</b>
A.1	Concepts .....	63
A.2	Functions .....	64
A.3	Variables .....	68
A.4	Data types .....	69

## Copying

This program is distributed under the terms of the Microsoft Public License.



# 1 Systems

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

## 1.1 distributions

Random numbers and distributions

### Long Name

Statistical distributions and related functions

**Author** Steven Nunez <steve@symbolics.tech>

### Home Page

<https://lisp-stat.dev/docs/manuals/distributions/>

### Source Control

(GIT <https://github.com/Lisp-Stat/distributions.git>)

### Bug Tracker

<https://github.com/Lisp-Stat/distributions/issues>

**License** MS-PL

### Long Description

The Distributions package provides a collection of probabilistic distributions and related functions including:

- Sampling from distributions
- Moments (e.g mean, variance, skewness, and kurtosis), entropy, and other properties
- Probability density/mass functions (pdf) and their logarithm (logpdf)
- Moment-generating functions and characteristic functions
- Maximum likelihood estimation
- Distribution composition and derived distributions

**Version** 1.0.0

### Dependencies

- alexandria (system).
- anaphora (system).
- array-operations (system).
- cephes (system).
- num-utils (system).
- special-functions (system).
- let-plus (system).
- float-features (system).

**Source** [distributions.asd], page 5.

### Child Components

- [packages.lisp], page 5 (file).
- [internals.lisp], page 5 (file).
- [generator.lisp], page 5 (file).
- [simple-multiplicative-congruential-generators.lisp], page 6 (file).
- [defs.lisp], page 7 (file).

- `[generics.lisp]`, page 7 (file).
- `[discrete.lisp]`, page 7 (file).
- `[uniform.lisp]`, page 8 (file).
- `[exponential.lisp]`, page 9 (file).
- `[normal.lisp]`, page 9 (file).
- `[log-normal.lisp]`, page 10 (file).
- `[truncated-normal.lisp]`, page 11 (file).
- `[t-distribution.lisp]`, page 12 (file).
- `[gamma.lisp]`, page 12 (file).
- `[chi-square.lisp]`, page 13 (file).
- `[beta.lisp]`, page 14 (file).
- `[rayleigh.lisp]`, page 14 (file).
- `[bernoulli.lisp]`, page 15 (file).
- `[binomial.lisp]`, page 15 (file).
- `[geometric.lisp]`, page 16 (file).
- `[poisson.lisp]`, page 16 (file).



## 2 Files

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

### 2.1 Lisp

#### 2.1.1 distributions/distributions.asd

**Source** [distributions.asd], page 5.

**Parent Component**  
[distributions], page 3 (system).

**ASDF Systems**  
[distributions], page 3.

#### 2.1.2 distributions/packages.lisp

**Source** [distributions.asd], page 5.

**Parent Component**  
[distributions], page 3 (system).

**Packages**

- [distributions.internals], page 17.
- [distributions], page 17.

#### 2.1.3 distributions/internals.lisp

**Dependency**  
[packages.lisp], page 5 (file).

**Source** [distributions.asd], page 5.

**Parent Component**  
[distributions], page 3 (system).

**Public Interface**

- [as-float], page 23 (function).
- [as-float-probabilities], page 23 (function).
- [as-float-vector], page 23 (function).
- [float-vector], page 44 (type).
- [internal-float], page 44 (type).
- [maybe-ignore-constant], page 23 (macro).
- [try], page 23 (macro).
- [with-floats], page 23 (macro).

#### 2.1.4 distributions/generator.lisp

**Dependency**  
[internals.lisp], page 5 (file).

**Source** [distributions.asd], page 5.

**Parent Component**  
[distributions], page 3 (system).

**Public Interface**

- `[generator]`, page 42 (class).
- `[initialize-instance]`, page 34 (method).
- `[make-generator]`, page 25 (function).
- `[next]`, page 25 (function).

**Internals**

- `[%next-double-float]`, page 45 (function).
- `[%next-integer]`, page 45 (function).
- `[%next-single-float]`, page 45 (function).
- `[*default-generator-type*]`, page 44 (special variable).
- `[chunk-length]`, page 56 (reader method).
- `[copy-state]`, page 57 (generic function).
- `[default-seed]`, page 57 (reader method).
- `[generate-seed]`, page 47 (function).
- `[generate-state]`, page 57 (generic function).
- `[state]`, page 59 (reader method).
- `[(setf state)]`, page 59 (writer method).

**2.1.5 distributions/simple-multiplicative-congruential-generators.lisp****Dependency**

`[generator.lisp]`, page 5 (file).

**Source**

`[distributions.asd]`, page 5.

**Parent Component**

`[distributions]`, page 3 (system).

**Public Interface**

- `[borosh13]`, page 42 (class).
- `[initialize-instance]`, page 34 (method).
- `[randu]`, page 43 (class).
- `[transputer]`, page 43 (class).
- `[waterman14]`, page 44 (class).

**Internals**

- `[a]`, page 56 (reader method).
- `[clone]`, page 56 (method).
- `[generate-state]`, page 57 (method).
- `[m]`, page 57 (reader method).
- `[next-chunk]`, page 58 (method).
- `[next-real]`, page 58 (method).
- `[simple-multiplicative-congruential]`, page 61 (class).

### 2.1.6 distributions/defs.lisp

**Dependency**

[simple-multiplicative-congruential-generators.lisp], page 6 (file).

**Source** [distributions.asd], page 5.

**Parent Component**

[distributions], page 3 (system).

**Public Interface**

- [cdf], page 28 (generic function).
- [draw], page 28 (generic function).
- [generator], page 29 (generic function).
- [log-pdf], page 30 (generic function).
- [mean], page 30 (generic function).
- [pdf], page 25 (function).
- [variance], page 33 (generic function).

**Internals**

- [check-probability], page 46 (function).
- [define-rv], page 44 (macro).

### 2.1.7 distributions/generics.lisp

**Dependency**

[defs.lisp], page 7 (file).

**Source** [distributions.asd], page 5.

**Parent Component**

[distributions], page 3 (system).

**Public Interface**

- [nu], page 31 (generic function).
- [standard-deviation], page 33 (generic function).

**Internals**

- [copy-r-univariate], page 47 (function).
- [make-r-univariate], page 50 (function).
- [r-univariate], page 60 (structure).
- [r-univariate-p], page 55 (function).
- [s^2], page 58 (generic function).

### 2.1.8 distributions/discrete.lisp

**Dependency**

[generics.lisp], page 7 (file).

**Source** [distributions.asd], page 5.

**Parent Component**

[distributions], page 3 (system).

**Public Interface**

- [cdf], page 28 (method).

- [distinct-random-integers], page 23 (function).
- [distinct-random-integers-dense], page 24 (function).
- [draw], page 29 (method).
- [log-pdf], page 30 (method).
- [mean], page 31 (method).
- [probabilities], page 32 (method).
- [r-discrete], page 26 (function).
- [r-discrete], page 36 (structure).
- [variance], page 34 (method).

### Internals

- [copy-r-discrete], page 46 (function).
- [make-r-discrete], page 49 (function).
- [r-discrete-alias], page 51 (reader).
- [(setf r-discrete-alias)], page 51 (writer).
- [r-discrete-n-float], page 52 (reader).
- [(setf r-discrete-n-float)], page 52 (writer).
- [r-discrete-p], page 52 (function).
- [r-discrete-prob], page 52 (reader).
- [(setf r-discrete-prob)], page 52 (writer).
- [r-discrete-probabilities], page 52 (reader).
- [(setf r-discrete-probabilities)], page 52 (writer).

## 2.1.9 distributions/uniform.lisp

### Dependency

[discrete.lisp], page 7 (file).

### Source

[distributions.asd], page 5.

### Parent Component

[distributions], page 3 (system).

### Public Interface

- [cdf], page 28 (method).
- [draw], page 29 (method).
- [draw-uniform], page 25 (function).
- [left], page 30 (method).
- [log-pdf], page 30 (method).
- [mean], page 31 (method).
- [quantile], page 32 (method).
- [r-uniform], page 27 (function).
- [r-uniform], page 41 (structure).
- [right], page 32 (method).
- [variance], page 34 (method).

### Internals

- [copy-r-uniform], page 47 (function).

- `[make-r-uniform]`, page 50 (function).
- `[r-uniform-left]`, page 55 (reader).
- `[(setf r-uniform-left)]`, page 55 (writer).
- `[r-uniform-p]`, page 55 (function).
- `[r-uniform-right]`, page 55 (reader).
- `[(setf r-uniform-right)]`, page 55 (writer).
- `[r-uniform-width]`, page 55 (reader).
- `[(setf r-uniform-width)]`, page 55 (writer).

### 2.1.10 distributions/exponential.lisp

#### Dependency

`[uniform.lisp]`, page 8 (file).

#### Source

`[distributions.asd]`, page 5.

#### Parent Component

`[distributions]`, page 3 (system).

#### Public Interface

- `[cdf]`, page 28 (method).
- `[draw]`, page 29 (method).
- `[draw-exponential]`, page 24 (function).
- `[draw-standard-exponential]`, page 24 (function).
- `[log-pdf]`, page 30 (method).
- `[mean]`, page 31 (method).
- `[quantile]`, page 32 (method).
- `[r-exponential]`, page 26 (function).
- `[r-exponential]`, page 36 (structure).
- `[rate]`, page 32 (method).
- `[variance]`, page 34 (method).

#### Internals

- `[copy-r-exponential]`, page 46 (function).
- `[make-r-exponential]`, page 49 (function).
- `[r-exponential-p]`, page 52 (function).
- `[r-exponential-rate]`, page 52 (reader).
- `[(setf r-exponential-rate)]`, page 52 (writer).

### 2.1.11 distributions/normal.lisp

#### Dependency

`[exponential.lisp]`, page 9 (file).

#### Source

`[distributions.asd]`, page 5.

#### Parent Component

`[distributions]`, page 3 (system).

#### Public Interface

- `[cdf]`, page 28 (method).
- `[draw]`, page 29 (method).

- `[draw-standard-normal]`, page 25 (function).
- `[from-standard-normal]`, page 25 (function).
- `[log-pdf]`, page 30 (method).
- `[mean]`, page 31 (method).
- `[quantile]`, page 32 (method).
- `[r-normal]`, page 26 (function).
- `[r-normal]`, page 39 (structure).
- `[to-standard-normal]`, page 27 (function).
- `[variance]`, page 34 (method).

#### Internals

- `[+normal-log-pdf-constant+]`, page 44 (constant).
- `[cdf-normal%]`, page 46 (function).
- `[copy-r-normal]`, page 47 (function).
- `[make-r-normal]`, page 49 (function).
- `[pdf-normal%]`, page 50 (function).
- `[quantile-normal%]`, page 50 (function).
- `[r-normal-mean]`, page 54 (reader).
- `[(setf r-normal-mean)]`, page 54 (writer).
- `[r-normal-p]`, page 54 (function).
- `[r-normal-sd]`, page 54 (reader).
- `[(setf r-normal-sd)]`, page 54 (writer).
- `[sd]`, page 58 (method).

### 2.1.12 distributions/log-normal.lisp

#### Dependency

`[normal.lisp]`, page 9 (file).

#### Source

`[distributions.asd]`, page 5.

#### Parent Component

`[distributions]`, page 3 (system).

#### Public Interface

- `[cdf]`, page 28 (method).
- `[draw]`, page 29 (method).
- `[log-pdf]`, page 30 (method).
- `[mean]`, page 31 (method).
- `[quantile]`, page 32 (method).
- `[r-log-normal]`, page 26 (function).
- `[r-log-normal]`, page 39 (structure).
- `[variance]`, page 34 (method).

#### Internals

- `[copy-r-log-normal]`, page 47 (function).
- `[make-r-log-normal]`, page 49 (function).
- `[r-log-normal-log-mean]`, page 53 (reader).

- `[(setf r-log-normal-log-mean)]`, page 53 (writer).
- `[r-log-normal-log-sd]`, page 53 (reader).
- `[(setf r-log-normal-log-sd)]`, page 53 (writer).
- `[r-log-normal-p]`, page 54 (function).

### 2.1.13 distributions/truncated-normal.lisp

#### Dependency

`[log-normal.lisp]`, page 10 (file).

#### Source

`[distributions.asd]`, page 5.

#### Parent Component

`[distributions]`, page 3 (system).

#### Public Interface

- `[cdf]`, page 28 (method).
- `[draw]`, page 29 (method).
- `[log-pdf]`, page 30 (method).
- `[mean]`, page 31 (method).
- `[quantile]`, page 32 (method).
- `[r-truncated-normal]`, page 27 (function).
- `[variance]`, page 33 (method).

#### Internals

- `[copy-left-truncated-normal]`, page 46 (function).
- `[draw-left-truncated-standard-normal]`, page 47 (function).
- `[left-truncated-normal]`, page 48 (function).
- `[left-truncated-normal]`, page 59 (structure).
- `[left-truncated-normal-alpha]`, page 48 (reader).
- `[(setf left-truncated-normal-alpha)]`, page 48 (writer).
- `[left-truncated-normal-left]`, page 48 (reader).
- `[(setf left-truncated-normal-left)]`, page 48 (writer).
- `[left-truncated-normal-left-standardized]`, page 48 (reader).
- `[(setf left-truncated-normal-left-standardized)]`, page 48 (writer).
- `[left-truncated-normal-m0]`, page 48 (reader).
- `[(setf left-truncated-normal-m0)]`, page 48 (writer).
- `[left-truncated-normal-mu]`, page 48 (reader).
- `[(setf left-truncated-normal-mu)]`, page 48 (writer).
- `[left-truncated-normal-p]`, page 48 (function).
- `[left-truncated-normal-sigma]`, page 48 (reader).
- `[(setf left-truncated-normal-sigma)]`, page 48 (writer).
- `[make-left-truncated-normal]`, page 49 (function).
- `[truncated-normal-moments%]`, page 56 (function).
- `[truncated-normal-optimal-alpha]`, page 56 (function).

### 2.1.14 distributions/t-distribution.lisp

#### Dependency

[truncated-normal.lisp], page 11 (file).

#### Source

[distributions.asd], page 5.

#### Parent Component

[distributions], page 3 (system).

#### Public Interface

- [draw], page 29 (method).
- [draw-standard-t], page 25 (function).
- [mean], page 31 (method).
- [nu], page 32 (method).
- [r-t], page 27 (function).
- [r-t], page 40 (structure).
- [scale], page 33 (method).
- [t-scale-to-variance-coefficient], page 27 (function).
- [variance], page 33 (method).

#### Internals

- [copy-r-t], page 47 (function).
- [make-r-t], page 50 (function).
- [r-t-mean], page 54 (reader).
- [(setf r-t-mean)], page 54 (writer).
- [r-t-nu], page 54 (reader).
- [(setf r-t-nu)], page 54 (writer).
- [r-t-p], page 55 (function).
- [r-t-scale], page 55 (reader).
- [(setf r-t-scale)], page 55 (writer).

### 2.1.15 distributions/gamma.lisp

#### Dependency

[t-distribution.lisp], page 12 (file).

#### Source

[distributions.asd], page 5.

#### Parent Component

[distributions], page 3 (system).

#### Public Interface

- [alpha], page 27 (method).
- [alpha], page 27 (method).
- [beta], page 28 (method).
- [beta], page 28 (method).
- [cdf], page 28 (method).
- [draw], page 29 (method).
- [draw], page 29 (method).
- [log-pdf], page 30 (method).



- [log-pdf], page 30 (method).
- [mean], page 31 (method).
- [mean], page 31 (method).
- [num=], page 34 (method).
- [quantile], page 32 (method).
- [r-gamma], page 26 (function).
- [r-gamma], page 37 (structure).
- [r-inverse-gamma], page 26 (function).
- [r-inverse-gamma], page 38 (structure).
- [variance], page 33 (method).
- [variance], page 33 (method).

### Internals

- [cdf-gamma%], page 45 (function).
- [cdf-gamma%+], page 45 (function).
- [copy-r-gamma], page 46 (function).
- [copy-r-inverse-gamma], page 46 (function).
- [draw-standard-gamma1], page 47 (function).
- [make-r-gamma], page 49 (function).
- [make-r-inverse-gamma], page 49 (function).
- [pdf-gamma], page 50 (function).
- [pdf-gamma%], page 50 (function).
- [pdf-gamma\*], page 50 (function).
- [pdf-gamma+], page 50 (function).
- [r-gamma-alpha], page 52 (reader).
- [(setf r-gamma-alpha)], page 52 (writer).
- [r-gamma-beta], page 52 (reader).
- [(setf r-gamma-beta)], page 52 (writer).
- [r-gamma-p], page 53 (function).
- [r-inverse-gamma-alpha], page 53 (reader).
- [(setf r-inverse-gamma-alpha)], page 53 (writer).
- [r-inverse-gamma-beta], page 53 (reader).
- [(setf r-inverse-gamma-beta)], page 53 (writer).
- [r-inverse-gamma-p], page 53 (function).
- [standard-gamma1-d-c], page 55 (function).

### 2.1.16 distributions/chi-square.lisp

#### Dependency

[gamma.lisp], page 12 (file).

#### Source

[distributions.asd], page 5.

#### Parent Component

[distributions], page 3 (system).

#### Public Interface

- [nu], page 31 (method).

- [nu], page 32 (method).
- [r-chi-square], page 26 (function).
- [r-inverse-chi-square], page 26 (function).

**Internals** [s<sup>2</sup>], page 58 (method).

### 2.1.17 distributions/beta.lisp

#### Dependency

[chi-square.lisp], page 13 (file).

**Source** [distributions.asd], page 5.

#### Parent Component

[distributions], page 3 (system).

#### Public Interface

- [alpha], page 27 (method).
- [beta], page 27 (method).
- [draw], page 29 (method).
- [mean], page 31 (method).
- [quantile], page 32 (method).
- [r-beta], page 26 (function).
- [r-beta], page 35 (structure).
- [variance], page 33 (method).

#### Internals

- [copy-r-beta], page 46 (function).
- [make-r-beta], page 49 (function).
- [r-beta-alpha], page 51 (reader).
- [(setf r-beta-alpha)], page 51 (writer).
- [r-beta-beta], page 51 (reader).
- [(setf r-beta-beta)], page 51 (writer).
- [r-beta-p], page 51 (function).

### 2.1.18 distributions/rayleigh.lisp

#### Dependency

[beta.lisp], page 14 (file).

**Source** [distributions.asd], page 5.

#### Parent Component

[distributions], page 3 (system).

#### Public Interface

- [cdf], page 28 (method).
- [draw], page 29 (method).
- [draw-rayleigh], page 24 (function).
- [mean], page 31 (method).
- [r-rayleigh], page 27 (function).
- [r-rayleigh], page 40 (structure).
- [scale], page 33 (method).

- [variance], page 33 (method).

#### Internals

- [copy-r-rayleigh], page 47 (function).
- [make-r-rayleigh], page 49 (function).
- [r-rayleigh-p], page 54 (function).
- [r-rayleigh-scale], page 54 (reader).
- [(setf r-rayleigh-scale)], page 54 (writer).

### 2.1.19 distributions/bernoulli.lisp

#### Dependency

[rayleigh.lisp], page 14 (file).

#### Source

[distributions.asd], page 5.

#### Parent Component

[distributions], page 3 (system).

#### Public Interface

- [cdf], page 28 (method).
- [draw], page 29 (method).
- [draw-bernoulli], page 24 (function).
- [mean], page 31 (method).
- [r-bernoulli], page 25 (function).
- [r-bernoulli], page 34 (structure).
- [variance], page 33 (method).

#### Internals

- [copy-r-bernoulli], page 46 (function).
- [draw-bernoulli-bit], page 47 (function).
- [make-r-bernoulli], page 49 (function).
- [pr], page 58 (method).
- [r-bernoulli-p], page 50 (function).
- [r-bernoulli-pr], page 51 (reader).
- [(setf r-bernoulli-pr)], page 51 (writer).

### 2.1.20 distributions/binomial.lisp

#### Dependency

[bernoulli.lisp], page 15 (file).

#### Source

[distributions.asd], page 5.

#### Parent Component

[distributions], page 3 (system).

#### Public Interface

- [draw], page 28 (method).
- [draw-binomial], page 24 (function).
- [mean], page 31 (method).
- [r-binomial], page 26 (function).
- [r-binomial], page 35 (structure).

- [variance], page 33 (method).

#### Internals

- [copy-r-binomial], page 46 (function).
- [make-r-binomial], page 49 (function).
- [n], page 58 (method).
- [pr], page 58 (method).
- [r-binomial-n], page 51 (reader).
- [(setf r-binomial-n)], page 51 (writer).
- [r-binomial-p], page 51 (function).
- [r-binomial-pr], page 51 (reader).
- [(setf r-binomial-pr)], page 51 (writer).

### 2.1.21 distributions/geometric.lisp

#### Dependency

[binomial.lisp], page 15 (file).

#### Source

[distributions.asd], page 5.

#### Parent Component

[distributions], page 3 (system).

#### Public Interface

- [draw], page 28 (method).
- [draw-geometric], page 24 (function).
- [mean], page 31 (method).
- [r-geometric], page 26 (function).
- [r-geometric], page 38 (structure).
- [variance], page 33 (method).

#### Internals

- [copy-r-geometric], page 46 (function).
- [make-r-geometric], page 49 (function).
- [pr], page 58 (method).
- [r-geometric-p], page 53 (function).
- [r-geometric-pr], page 53 (reader).
- [(setf r-geometric-pr)], page 53 (writer).

### 2.1.22 distributions/poisson.lisp

#### Dependency

[geometric.lisp], page 16 (file).

#### Source

[distributions.asd], page 5.

#### Parent Component

[distributions], page 3 (system).

#### Public Interface

[draw-poisson], page 24 (function).

## 3 Packages

Packages are listed by definition order.

### 3.1 `distributions.internals`

**Source**      `[packages.lisp]`, page 5.

**Use List**

- `alexandria`.
- `common-lisp`.
- `let-plus`.

**Used By List**

`[distributions]`, page 17.

**Public Interface**

- `[as-float]`, page 23 (function).
- `[as-float-probabilities]`, page 23 (function).
- `[as-float-vector]`, page 23 (function).
- `[float-vector]`, page 44 (type).
- `[internal-float]`, page 44 (type).
- `[maybe-ignore-constant]`, page 23 (macro).
- `[try]`, page 23 (macro).
- `[with-floats]`, page 23 (macro).

### 3.2 `distributions`

**Source**      `[packages.lisp]`, page 5.

**Use List**

- `alexandria`.
- `anaphora`.
- `common-lisp`.
- `[distributions.internals]`, page 17.
- `let-plus`.
- `num-utils.elementwise`.
- `num-utils.matrix`.
- `num-utils.num=`.
- `special-functions`.

**Public Interface**

- `[alpha]`, page 27 (generic function).
- `[beta]`, page 27 (generic function).
- `[borosh13]`, page 42 (class).
- `[cdf]`, page 28 (generic function).
- `[distinct-random-integers]`, page 23 (function).
- `[distinct-random-integers-dense]`, page 24 (function).
- `[draw]`, page 28 (generic function).

- `[draw-bernoulli]`, page 24 (function).
- `[draw-binomial]`, page 24 (function).
- `[draw-exponential]`, page 24 (function).
- `[draw-geometric]`, page 24 (function).
- `[draw-poisson]`, page 24 (function).
- `[draw-rayleigh]`, page 24 (function).
- `[draw-standard-exponential]`, page 24 (function).
- `[draw-standard-normal]`, page 25 (function).
- `[draw-standard-t]`, page 25 (function).
- `[draw-uniform]`, page 25 (function).
- `[from-standard-normal]`, page 25 (function).
- `[generator]`, page 29 (generic function).
- `[generator]`, page 42 (class).
- `[left]`, page 29 (generic function).
- `[log-pdf]`, page 30 (generic function).
- `[make-generator]`, page 25 (function).
- `[mean]`, page 30 (generic function).
- `[next]`, page 25 (function).
- `[nu]`, page 31 (generic function).
- `[pdf]`, page 25 (function).
- `[probabilities]`, page 32 (generic function).
- `[quantile]`, page 32 (generic function).
- `[r-bernoulli]`, page 25 (function).
- `[r-bernoulli]`, page 34 (structure).
- `[r-beta]`, page 26 (function).
- `[r-beta]`, page 35 (structure).
- `[r-binomial]`, page 26 (function).
- `[r-binomial]`, page 35 (structure).
- `[r-chi-square]`, page 26 (function).
- `[r-discrete]`, page 26 (function).
- `[r-discrete]`, page 36 (structure).
- `[r-exponential]`, page 26 (function).
- `[r-exponential]`, page 36 (structure).
- `[r-gamma]`, page 26 (function).
- `[r-gamma]`, page 37 (structure).
- `[r-geometric]`, page 26 (function).
- `[r-geometric]`, page 38 (structure).
- `[r-inverse-chi-square]`, page 26 (function).
- `[r-inverse-gamma]`, page 26 (function).
- `[r-inverse-gamma]`, page 38 (structure).
- `[r-log-normal]`, page 26 (function).
- `[r-log-normal]`, page 39 (structure).
- `[r-normal]`, page 26 (function).

- `[r-normal]`, page 39 (structure).
- `[r-rayleigh]`, page 27 (function).
- `[r-rayleigh]`, page 40 (structure).
- `[r-t]`, page 27 (function).
- `[r-t]`, page 40 (structure).
- `[r-truncated-normal]`, page 27 (function).
- `[r-uniform]`, page 27 (function).
- `[r-uniform]`, page 41 (structure).
- `[randu]`, page 43 (class).
- `[rate]`, page 32 (generic function).
- `[right]`, page 32 (generic function).
- `[scale]`, page 33 (generic function).
- `[standard-deviation]`, page 33 (generic function).
- `[t-scale-to-variance-coefficient]`, page 27 (function).
- `[to-standard-normal]`, page 27 (function).
- `[transputer]`, page 43 (class).
- `[variance]`, page 33 (generic function).
- `[waterman14]`, page 44 (class).

## Internals

- `[%next-double-float]`, page 45 (function).
- `[%next-integer]`, page 45 (function).
- `[%next-single-float]`, page 45 (function).
- `[*default-generator-type*]`, page 44 (special variable).
- `[+normal-log-pdf-constant+]`, page 44 (constant).
- `[a]`, page 56 (generic reader).
- `[cdf-gamma%]`, page 45 (function).
- `[cdf-gamma%+]`, page 45 (function).
- `[cdf-normal%]`, page 46 (function).
- `[check-probability]`, page 46 (function).
- `[chunk-length]`, page 56 (generic reader).
- `[clone]`, page 56 (generic function).
- `[copy-left-truncated-normal]`, page 46 (function).
- `[copy-r-bernoulli]`, page 46 (function).
- `[copy-r-beta]`, page 46 (function).
- `[copy-r-binomial]`, page 46 (function).
- `[copy-r-discrete]`, page 46 (function).
- `[copy-r-exponential]`, page 46 (function).
- `[copy-r-gamma]`, page 46 (function).
- `[copy-r-geometric]`, page 46 (function).
- `[copy-r-inverse-gamma]`, page 46 (function).
- `[copy-r-log-normal]`, page 47 (function).
- `[copy-r-normal]`, page 47 (function).

- [copy-r-rayleigh], page 47 (function).
- [copy-r-t], page 47 (function).
- [copy-r-uniform], page 47 (function).
- [copy-r-univariate], page 47 (function).
- [copy-state], page 57 (generic function).
- [default-seed], page 57 (generic reader).
- [define-rv], page 44 (macro).
- [draw-bernoulli-bit], page 47 (function).
- [draw-left-truncated-standard-normal], page 47 (function).
- [draw-standard-gamma1], page 47 (function).
- [generate-seed], page 47 (function).
- [generate-state], page 57 (generic function).
- [left-truncated-normal], page 48 (function).
- [left-truncated-normal], page 59 (structure).
- [left-truncated-normal-alpha], page 48 (reader).
- [(setf left-truncated-normal-alpha)], page 48 (writer).
- [left-truncated-normal-left], page 48 (reader).
- [(setf left-truncated-normal-left)], page 48 (writer).
- [left-truncated-normal-left-standardized], page 48 (reader).
- [(setf left-truncated-normal-left-standardized)], page 48 (writer).
- [left-truncated-normal-m0], page 48 (reader).
- [(setf left-truncated-normal-m0)], page 48 (writer).
- [left-truncated-normal-mu], page 48 (reader).
- [(setf left-truncated-normal-mu)], page 48 (writer).
- [left-truncated-normal-p], page 48 (function).
- [left-truncated-normal-sigma], page 48 (reader).
- [(setf left-truncated-normal-sigma)], page 48 (writer).
- [m], page 57 (generic reader).
- [make-left-truncated-normal], page 49 (function).
- [make-r-bernoulli], page 49 (function).
- [make-r-beta], page 49 (function).
- [make-r-binomial], page 49 (function).
- [make-r-discrete], page 49 (function).
- [make-r-exponential], page 49 (function).
- [make-r-gamma], page 49 (function).
- [make-r-geometric], page 49 (function).
- [make-r-inverse-gamma], page 49 (function).
- [make-r-log-normal], page 49 (function).
- [make-r-normal], page 49 (function).
- [make-r-rayleigh], page 49 (function).
- [make-r-t], page 50 (function).
- [make-r-uniform], page 50 (function).
- [make-r-univariate], page 50 (function).



- `[n]`, page 57 (generic function).
- `[next-chunk]`, page 58 (generic function).
- `[next-real]`, page 58 (generic function).
- `[pdf-gamma]`, page 50 (function).
- `[pdf-gamma%]`, page 50 (function).
- `[pdf-gamma*]`, page 50 (function).
- `[pdf-gamma+]`, page 50 (function).
- `[pdf-normal%]`, page 50 (function).
- `[pr]`, page 58 (generic function).
- `[quantile-normal%]`, page 50 (function).
- `[r-bernoulli-p]`, page 50 (function).
- `[r-bernoulli-pr]`, page 51 (reader).
- `[(setf r-bernoulli-pr)]`, page 51 (writer).
- `[r-beta-alpha]`, page 51 (reader).
- `[(setf r-beta-alpha)]`, page 51 (writer).
- `[r-beta-beta]`, page 51 (reader).
- `[(setf r-beta-beta)]`, page 51 (writer).
- `[r-beta-p]`, page 51 (function).
- `[r-binomial-n]`, page 51 (reader).
- `[(setf r-binomial-n)]`, page 51 (writer).
- `[r-binomial-p]`, page 51 (function).
- `[r-binomial-pr]`, page 51 (reader).
- `[(setf r-binomial-pr)]`, page 51 (writer).
- `[r-discrete-alias]`, page 51 (reader).
- `[(setf r-discrete-alias)]`, page 51 (writer).
- `[r-discrete-n-float]`, page 52 (reader).
- `[(setf r-discrete-n-float)]`, page 52 (writer).
- `[r-discrete-p]`, page 52 (function).
- `[r-discrete-prob]`, page 52 (reader).
- `[(setf r-discrete-prob)]`, page 52 (writer).
- `[r-discrete-probabilities]`, page 52 (reader).
- `[(setf r-discrete-probabilities)]`, page 52 (writer).
- `[r-exponential-p]`, page 52 (function).
- `[r-exponential-rate]`, page 52 (reader).
- `[(setf r-exponential-rate)]`, page 52 (writer).
- `[r-gamma-alpha]`, page 52 (reader).
- `[(setf r-gamma-alpha)]`, page 52 (writer).
- `[r-gamma-beta]`, page 52 (reader).
- `[(setf r-gamma-beta)]`, page 52 (writer).
- `[r-gamma-p]`, page 53 (function).
- `[r-geometric-p]`, page 53 (function).
- `[r-geometric-pr]`, page 53 (reader).
- `[(setf r-geometric-pr)]`, page 53 (writer).

- [r-inverse-gamma-alpha], page 53 (reader).
- [(setf r-inverse-gamma-alpha)], page 53 (writer).
- [r-inverse-gamma-beta], page 53 (reader).
- [(setf r-inverse-gamma-beta)], page 53 (writer).
- [r-inverse-gamma-p], page 53 (function).
- [r-log-normal-log-mean], page 53 (reader).
- [(setf r-log-normal-log-mean)], page 53 (writer).
- [r-log-normal-log-sd], page 53 (reader).
- [(setf r-log-normal-log-sd)], page 53 (writer).
- [r-log-normal-p], page 54 (function).
- [r-normal-mean], page 54 (reader).
- [(setf r-normal-mean)], page 54 (writer).
- [r-normal-p], page 54 (function).
- [r-normal-sd], page 54 (reader).
- [(setf r-normal-sd)], page 54 (writer).
- [r-rayleigh-p], page 54 (function).
- [r-rayleigh-scale], page 54 (reader).
- [(setf r-rayleigh-scale)], page 54 (writer).
- [r-t-mean], page 54 (reader).
- [(setf r-t-mean)], page 54 (writer).
- [r-t-nu], page 54 (reader).
- [(setf r-t-nu)], page 54 (writer).
- [r-t-p], page 55 (function).
- [r-t-scale], page 55 (reader).
- [(setf r-t-scale)], page 55 (writer).
- [r-uniform-left], page 55 (reader).
- [(setf r-uniform-left)], page 55 (writer).
- [r-uniform-p], page 55 (function).
- [r-uniform-right], page 55 (reader).
- [(setf r-uniform-right)], page 55 (writer).
- [r-uniform-width], page 55 (reader).
- [(setf r-uniform-width)], page 55 (writer).
- [r-univariate], page 60 (structure).
- [r-univariate-p], page 55 (function).
- [s^2], page 58 (generic function).
- [sd], page 58 (generic function).
- [simple-multiplicative-congruential], page 61 (class).
- [standard-gamma1-d-c], page 55 (function).
- [state], page 59 (generic reader).
- [(setf state)], page 59 (generic writer).
- [truncated-normal-moments%], page 56 (function).
- [truncated-normal-optimal-alpha], page 56 (function).

## 4 Definitions

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

### 4.1 Public Interface

#### 4.1.1 Macros

**maybe-ignore-constant** (*ignore-constant? value constant*) [Macro]

Handle a constant that is calculated only when IGNORE-CONSTANT? is NIL and VALUE is not negative infinity (represented by NIL).

**Package** [distributions.internals], page 17.

**Source** [internals.lisp], page 5.

**try** ((**&rest** *bindings*) *condition value*) [Macro]

Evaluate bindings (expanding into LET+, so all features can be used) until condition is satisfied, then return value.

**Package** [distributions.internals], page 17.

**Source** [internals.lisp], page 5.

**with-floats** ((**&rest** *variables*) **&body** *body*) [Macro]

Rebind each variable, coerced to the internal float type used by DISTRIBUTIONS.

**Package** [distributions.internals], page 17.

**Source** [internals.lisp], page 5.

#### 4.1.2 Ordinary functions

**as-float** (*x*) [Function]

Return the argument coerced to the DISTRIBUTIONS library's internal float type.

**Package** [distributions.internals], page 17.

**Source** [internals.lisp], page 5.

**as-float-probabilities** (*vector*) [Function]

Normalize vector as probabilities, assert that all are positive, return them as a VECTOR-DOUBLE-FLOAT. Vector is always copied.

**Package** [distributions.internals], page 17.

**Source** [internals.lisp], page 5.

**as-float-vector** (*vector &key copy?*) [Function]

Return VECTOR converted to another vector with elements converted to INTERNAL-FLOAT if necessary. When COPY?, the vector is always copied.

**Package** [distributions.internals], page 17.

**Source** [internals.lisp], page 5.

**distinct-random-integers** (*count limit &key rng*) [Function]

Return a vector of COUNT distinct random integers, in increasing order, drawn from the uniform discrete distribution on {0 , ..., limit-1}.

**Package** [distributions], page 17.

**Source** [discrete.lisp], page 7.

- distinct-random-integers-dense** (*count limit &key rng*) [Function]  
 Implementation of DISTINCT-RANDOM-INTEGERS when count/limit is (relatively) high.  
 Implements algorithm S from @cite{taocp3}, p 142.  
**Package** [distributions], page 17.  
**Source** [discrete.lisp], page 7.
- draw-bernoulli** (*p &key rng*) [Function]  
 Return T with probability p, otherwise NIL. Rationals are handled exactly.  
**Package** [distributions], page 17.  
**Source** [bernoulli.lisp], page 15.
- draw-binomial** (*p n &key rng*) [Function]  
 Return the number of successes out of N Bernoulli trials with probability of success P.  
**Package** [distributions], page 17.  
**Source** [binomial.lisp], page 15.
- draw-exponential** (*rate &key rng*) [Function]  
 Return a random variable from the Exponential(rate) distribution which has density  $\text{rate} \cdot \exp(-\text{rate} \cdot x)$  for  $x \geq 0$  and 0 for  $x < 0$ .  $\text{rate} > 0$ .  
**Package** [distributions], page 17.  
**Source** [exponential.lisp], page 9.
- draw-geometric** (*p &key rng*) [Function]  
 Return the number of Bernoulli trials, with probability of success P, that were needed to reach the first success. This is  $\geq 1$ .  
**Package** [distributions], page 17.  
**Source** [geometric.lisp], page 16.
- draw-poisson** (*lamda &key rng*) [Function]  
 Return the number of events that occur with probability LAMDA. The algorithm is from Donald E. Knuth (1969). Seminumerical Algorithms. The Art of Computer Programming, Volume 2. Addison Wesley. WARNING: It's simple but only linear in the return value K and is numerically unstable for large LAMDA.  
**Package** [distributions], page 17.  
**Source** [poisson.lisp], page 16.
- draw-rayleigh** (*scale &key rng*) [Function]  
 Return a random variable from the Rayleigh(scale) distribution, where  $\text{scale} > 0$  and density  $x \cdot \exp(-x^2 / (2 \cdot \text{scale}^2)) / \text{scale}^2$  for  $x \geq 0$  and 0 for  $x < 0$ .  
**Package** [distributions], page 17.  
**Source** [rayleigh.lisp], page 14.
- draw-standard-exponential** (*&key rng*) [Function]  
 Return a random variable from the Exponential(1) distribution, which has density  $\exp(-x)$  for  $x \geq 0$  and 0 for  $x < 0$ .  
**Package** [distributions], page 17.  
**Source** [exponential.lisp], page 9.

- draw-standard-normal** (*&key rng*) [Function]  
Draw a random number from  $N(0,1)$ .  
**Package** [distributions], page 17.  
**Source** [normal.lisp], page 9.
- draw-standard-t** (*nu &key rng*) [Function]  
Draw a standard T random variate, with NU degrees of freedom.  
**Package** [distributions], page 17.  
**Source** [t-distribution.lisp], page 12.
- draw-uniform** (*left right &key rng*) [Function]  
Return a random variable from the uniform distribution between LEFT and RIGHT. It's type is the same as that of (- LEFT RIGHT).  
**Package** [distributions], page 17.  
**Source** [uniform.lisp], page 8.
- from-standard-normal** (*x mu sigma*) [Function]  
Scale x from standard normal.  
**Package** [distributions], page 17.  
**Source** [normal.lisp], page 9.
- make-generator** (*&key seed type*) [Function]  
Make a random number generator object. SEED can be any of NIL, T, an other generator, an integer, or any type of seed that a generator of type TYPE supports: - NIL: the generator's STD-SEED is used;  
- T: a random seed is used;  
- a generator: a clone is returned;  
- otherwise: SEED is used as depends on the generator.  
**Package** [distributions], page 17.  
**Source** [generator.lisp], page 5.
- next** (*limit &optional rng*) [Function]  
Generates a uniformly distributed pseudo-random number greater than or equal to zero and less than LIMIT. RNG, if supplied, is the random generator to use.  
**Package** [distributions], page 17.  
**Source** [generator.lisp], page 5.
- pdf** (*rv x &optional ignore-constant?*) [Function]  
Probability distribution function of RANDOM-VARIABLE at X. See LOG-PDF for the semantics of IGNORE-CONSTANT?.  
**Package** [distributions], page 17.  
**Source** [defs.lisp], page 7.
- r-bernoulli** (*pr*) [Function]  
**Package** [distributions], page 17.  
**Source** [bernoulli.lisp], page 15.

<b>r-beta</b> ( <i>alpha beta</i> )	[Function]
<b>Package</b> [distributions], page 17.	
<b>Source</b> [beta.lisp], page 14.	
<b>r-binomial</b> ( <i>pr n</i> )	[Function]
<b>Package</b> [distributions], page 17.	
<b>Source</b> [binomial.lisp], page 15.	
<b>r-chi-square</b> ( <i>nu</i> )	[Function]
Chi-square distribution with NU degrees of freedom.	
<b>Package</b> [distributions], page 17.	
<b>Source</b> [chi-square.lisp], page 13.	
<b>r-discrete</b> ( <i>probabilities</i> )	[Function]
<b>Package</b> [distributions], page 17.	
<b>Source</b> [discrete.lisp], page 7.	
<b>r-exponential</b> ( <i>rate</i> )	[Function]
<b>Package</b> [distributions], page 17.	
<b>Source</b> [exponential.lisp], page 9.	
<b>r-gamma</b> ( <i>alpha beta</i> )	[Function]
<b>Package</b> [distributions], page 17.	
<b>Source</b> [gamma.lisp], page 12.	
<b>r-geometric</b> ( <i>pr</i> )	[Function]
<b>Package</b> [distributions], page 17.	
<b>Source</b> [geometric.lisp], page 16.	
<b>r-inverse-chi-square</b> ( <i>nu &amp;optional s<sup>2</sup></i> )	[Function]
Generalized inverse chi-square distribution. Reparametrized to INVERSE-GAMMA.	
<b>Package</b> [distributions], page 17.	
<b>Source</b> [chi-square.lisp], page 13.	
<b>r-inverse-gamma</b> ( <i>alpha beta</i> )	[Function]
<b>Package</b> [distributions], page 17.	
<b>Source</b> [gamma.lisp], page 12.	
<b>r-log-normal</b> ( <i>log-mean log-sd</i> )	[Function]
<b>Package</b> [distributions], page 17.	
<b>Source</b> [log-normal.lisp], page 10.	
<b>r-normal</b> ( <i>&amp;optional mean variance</i> )	[Function]
<b>Package</b> [distributions], page 17.	
<b>Source</b> [normal.lisp], page 9.	

**r-rayleigh** (*scale*) [Function]

**Package** [distributions], page 17.

**Source** [rayleigh.lisp], page 14.

**r-t** (*mean scale nu*) [Function]

**Package** [distributions], page 17.

**Source** [t-distribution.lisp], page 12.

**r-truncated-normal** (*left right &optional mu sigma*) [Function]

Truncated normal distribution. If LEFT or RIGHT is NIL, it corresponds to -/+ infinity.

**Package** [distributions], page 17.

**Source** [truncated-normal.lisp], page 11.

**r-uniform** (*left right*) [Function]

**Package** [distributions], page 17.

**Source** [uniform.lisp], page 8.

**t-scale-to-variance-coefficient** (*nu*) [Function]

Return the coefficient that multiplies the Sigma matrix or the squared scale to get the variance of a (multivariate) Student-T distribution. Also checks that  $\nu > 2$ , ie the variance is defined.

**Package** [distributions], page 17.

**Source** [t-distribution.lisp], page 12.

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

Scale x to standard normal.

**Package** [distributions], page 17.

**Source** [normal.lisp], page 9.

### 4.1.3 Generic functions

**alpha** (*r-gamma0*) [Generic Function]

**Package** [distributions], page 17.

**Methods**

**alpha** ((*r-beta0* [**r-beta**], page 35)) [Method]

**Source** [beta.lisp], page 14.

**alpha** ((*r-inverse-gamma0* [**r-inverse-gamma**], page 38)) [Method]

**Source** [gamma.lisp], page 12.

**alpha** ((*r-gamma0* [**r-gamma**], page 37)) [Method]

**Source** [gamma.lisp], page 12.

**beta** (*r-gamma0*) [Generic Function]

**Package** [distributions], page 17.

**Methods**

**beta** ((*r-beta0* [**r-beta**], page 35)) [Method]

**Source** [beta.lisp], page 14.

- beta** ((*r-inverse-gamma0* [*r-inverse-gamma*], page 38)) [Method]  
**Source** [gamma.lisp], page 12.
- beta** ((*r-gamma0* [*r-gamma*], page 37)) [Method]  
**Source** [gamma.lisp], page 12.
- cdf** (*random-variable* x) [Generic Function]  
 Cumulative distribution function of RANDOM-VARIABLE at X.  
**Package** [distributions], page 17.  
**Source** [defs.lisp], page 7.  
**Methods**
- cdf** ((*r-bernoulli0* [*r-bernoulli*], page 34) x) [Method]  
**Source** [bernoulli.lisp], page 15.
- cdf** ((*r-rayleigh0* [*r-rayleigh*], page 40) x) [Method]  
**Source** [rayleigh.lisp], page 14.
- cdf** ((*r-gamma0* [*r-gamma*], page 37) x) [Method]  
**Source** [gamma.lisp], page 12.
- cdf** ((*left-truncated-normal0* [*left-truncated-normal*], page 59) x) [Method]  
**Source** [truncated-normal.lisp], page 11.
- cdf** ((*r-log-normal0* [*r-log-normal*], page 39) x) [Method]  
**Source** [log-normal.lisp], page 10.
- cdf** ((*r-normal0* [*r-normal*], page 39) x) [Method]  
**Source** [normal.lisp], page 9.
- cdf** ((*r-exponential0* [*r-exponential*], page 36) x) [Method]  
**Source** [exponential.lisp], page 9.
- cdf** ((*r-uniform0* [*r-uniform*], page 41) x) [Method]  
**Source** [uniform.lisp], page 8.
- cdf** ((*instance* [*r-discrete*], page 36) i) [Method]  
**Source** [discrete.lisp], page 7.
- draw** (*random-variable* &key rng &allow-other-keys) [Generic Function]  
 Draw random variates. Can also be used on generators.  
**Package** [distributions], page 17.  
**Source** [defs.lisp], page 7.  
**Methods**
- draw** ((*r-geometric0* [*r-geometric*], page 38) &key rng) [Method]  
**Source** [geometric.lisp], page 16.
- draw** ((*r-binomial0* [*r-binomial*], page 35) &key rng) [Method]  
**Source** [binomial.lisp], page 15.



<code>draw ((r-bernoulli0 [r-bernoulli], page 34) &amp;key rng)</code>	[Method]
Source [bernoulli.lisp], page 15.	
<code>draw ((r-rayleigh0 [r-rayleigh], page 40) &amp;key rng)</code>	[Method]
Source [rayleigh.lisp], page 14.	
<code>draw ((r-beta0 [r-beta], page 35) &amp;key rng)</code>	[Method]
Source [beta.lisp], page 14.	
<code>draw ((r-inverse-gamma0 [r-inverse-gamma], page 38) &amp;key rng)</code>	[Method]
Source [gamma.lisp], page 12.	
<code>draw ((r-gamma0 [r-gamma], page 37) &amp;key rng)</code>	[Method]
Source [gamma.lisp], page 12.	
<code>draw ((r-t0 [r-t], page 40) &amp;key rng)</code>	[Method]
Source [t-distribution.lisp], page 12.	
<code>draw ((left-truncated-normal0 [left-truncated-normal], page 59) &amp;key rng)</code>	[Method]
Source [truncated-normal.lisp], page 11.	
<code>draw ((r-log-normal0 [r-log-normal], page 39) &amp;key rng)</code>	[Method]
Source [log-normal.lisp], page 10.	
<code>draw ((r-normal0 [r-normal], page 39) &amp;key rng)</code>	[Method]
Source [normal.lisp], page 9.	
<code>draw ((r-exponential0 [r-exponential], page 36) &amp;key rng)</code>	[Method]
Source [exponential.lisp], page 9.	
<code>draw ((r-uniform0 [r-uniform], page 41) &amp;key rng)</code>	[Method]
Source [uniform.lisp], page 8.	
<code>draw ((instance [r-discrete], page 36) &amp;key rng)</code>	[Method]
Source [discrete.lisp], page 7.	
<code>draw ((function function) &amp;key rng)</code>	[Method]
<b>generator</b> ( <i>random-variable</i> &key rng)	[Generic Function]
Return a closure that returns random draws.	
Package [distributions], page 17.	
Source [defs.lisp], page 7.	
Methods	
generator ( <i>random-variable</i> &key rng)	[Method]
<b>left</b> ( <i>r-uniform0</i> )	[Generic Function]
Package [distributions], page 17.	
Methods	

`left ((r-uniform0 [r-uniform], page 41))` [Method]

**Source** [uniform.lisp], page 8.

**log-pdf** (*random-variable* *x* **&optional** *ignore-constant?*) [Generic Function]

Log of probability distribution function of RANDOM-VARIABLE at X. NIL corresponds to log(-infinity). When IGNORE-CONSTANT?, the result may be shifted by an arbitrary real constant that does not change between calls of the same RANDOM-VARIABLE. This may save computation, and is useful for MCMC methods, etc.

**Package** [distributions], page 17.

**Source** [defs.lisp], page 7.

#### Methods

`log-pdf ((r-inverse-gamma0 [r-inverse-gamma], page 38) x &optional ignore-constant?)` [Method]

**Source** [gamma.lisp], page 12.

`log-pdf ((r-gamma0 [r-gamma], page 37) x &optional ignore-constant?)` [Method]

**Source** [gamma.lisp], page 12.

`log-pdf ((left-truncated-normal0 [left-truncated-normal], page 59) x &optional ignore-constant?)` [Method]

**Source** [truncated-normal.lisp], page 11.

`log-pdf ((r-log-normal0 [r-log-normal], page 39) x &optional ignore-constant?)` [Method]

**Source** [log-normal.lisp], page 10.

`log-pdf ((r-normal0 [r-normal], page 39) x &optional ignore-constant?)` [Method]

**Source** [normal.lisp], page 9.

`log-pdf ((r-exponential0 [r-exponential], page 36) x &optional ignore-constant?)` [Method]

**Source** [exponential.lisp], page 9.

`log-pdf ((r-uniform0 [r-uniform], page 41) x &optional ignore-constant?)` [Method]

**Source** [uniform.lisp], page 8.

`log-pdf ((instance [r-discrete], page 36) i &optional ignore-constant?)` [Method]

**Source** [discrete.lisp], page 7.

**mean** (*random-variable*) [Generic Function]

Mean of random variable.

**Package** [distributions], page 17.

**Source** [defs.lisp], page 7.

#### Methods

<code>mean ((r-geometric0 [r-geometric], page 38))</code>	[Method]
<b>Source</b> [geometric.lisp], page 16.	
<code>mean ((r-binomial0 [r-binomial], page 35))</code>	[Method]
<b>Source</b> [binomial.lisp], page 15.	
<code>mean ((r-bernoulli0 [r-bernoulli], page 34))</code>	[Method]
<b>Source</b> [bernoulli.lisp], page 15.	
<code>mean ((r-rayleigh0 [r-rayleigh], page 40))</code>	[Method]
<b>Source</b> [rayleigh.lisp], page 14.	
<code>mean ((r-beta0 [r-beta], page 35))</code>	[Method]
<b>Source</b> [beta.lisp], page 14.	
<code>mean ((r-inverse-gamma0 [r-inverse-gamma], page 38))</code>	[Method]
<b>Source</b> [gamma.lisp], page 12.	
<code>mean ((r-gamma0 [r-gamma], page 37))</code>	[Method]
<b>Source</b> [gamma.lisp], page 12.	
<code>mean ((r-t0 [r-t], page 40))</code>	[Method]
<b>Source</b> [t-distribution.lisp], page 12.	
<code>mean ((left-truncated-normal0 [left-truncated-normal], page 59))</code>	[Method]
<b>Source</b> [truncated-normal.lisp], page 11.	
<code>mean ((r-log-normal0 [r-log-normal], page 39))</code>	[Method]
<b>Source</b> [log-normal.lisp], page 10.	
<code>mean ((r-normal0 [r-normal], page 39))</code>	[Method]
<b>Source</b> [normal.lisp], page 9.	
<code>mean ((r-exponential0 [r-exponential], page 36))</code>	[Method]
<b>Source</b> [exponential.lisp], page 9.	
<code>mean ((r-uniform0 [r-uniform], page 41))</code>	[Method]
<b>Source</b> [uniform.lisp], page 8.	
<code>mean ((instance [r-discrete], page 36))</code>	[Method]
<b>Source</b> [discrete.lisp], page 7.	
<b>nu</b> ( <i>distribution</i> )	[Generic Function]
Return the degrees of freedom when applicable.	
<b>Package</b> [distributions], page 17.	
<b>Source</b> [generics.lisp], page 7.	
<b>Methods</b>	
<code>nu ((r-inverse-gamma [r-inverse-gamma], page 38))</code>	[Method]
<b>Source</b> [chi-square.lisp], page 13.	

- `nu ((r-gamma [r-gamma], page 37))` [Method]  
**Source** [chi-square.lisp], page 13.
- `nu ((r-t0 [r-t], page 40))` [Method]  
**Source** [t-distribution.lisp], page 12.
- `probabilities (instance)` [Generic Function]  
**Package** [distributions], page 17.  
**Methods**
- `probabilities ((instance [r-discrete], page 36))` [Method]  
**Source** [discrete.lisp], page 7.
- `quantile (r-uniform0 p)` [Generic Function]  
**Package** [distributions], page 17.  
**Methods**
- `quantile ((r-beta0 [r-beta], page 35) q)` [Method]  
**Source** [beta.lisp], page 14.
- `quantile ((r-gamma0 [r-gamma], page 37) q)` [Method]  
**Source** [gamma.lisp], page 12.
- `quantile ((left-truncated-normal0 [left-truncated-normal], page 59) q)` [Method]  
**Source** [truncated-normal.lisp], page 11.
- `quantile ((r-log-normal0 [r-log-normal], page 39) q)` [Method]  
**Source** [log-normal.lisp], page 10.
- `quantile ((r-normal0 [r-normal], page 39) q)` [Method]  
**Source** [normal.lisp], page 9.
- `quantile ((r-exponential0 [r-exponential], page 36) p)` [Method]  
**Source** [exponential.lisp], page 9.
- `quantile ((r-uniform0 [r-uniform], page 41) p)` [Method]  
**Source** [uniform.lisp], page 8.
- `rate (r-exponential0)` [Generic Function]  
**Package** [distributions], page 17.  
**Methods**
- `rate ((r-exponential0 [r-exponential], page 36))` [Method]  
**Source** [exponential.lisp], page 9.
- `right (r-uniform0)` [Generic Function]  
**Package** [distributions], page 17.  
**Methods**
- `right ((r-uniform0 [r-uniform], page 41))` [Method]  
**Source** [uniform.lisp], page 8.

<b>scale</b> ( <i>r-t0</i> )	[Generic Function]
<b>Package</b> [distributions], page 17.	
<b>Methods</b>	
scale (( <i>r-rayleigh0</i> [ <i>r-rayleigh</i> ], page 40))	[Method]
<b>Source</b> [rayleigh.lisp], page 14.	
scale (( <i>r-t0</i> [ <i>r-t</i> ], page 40))	[Method]
<b>Source</b> [t-distribution.lisp], page 12.	
<b>standard-deviation</b> ( <i>random-variable</i> )	[Generic Function]
Standard deviation of random variable.	
<b>Package</b> [distributions], page 17.	
<b>Source</b> [generics.lisp], page 7.	
<b>Methods</b>	
standard-deviation (( <i>random-variable</i> [ <i>r-univariate</i> ], page 60))	[Method]
<b>variance</b> ( <i>random-variable</i> )	[Generic Function]
Variance of random variable.	
<b>Package</b> [distributions], page 17.	
<b>Source</b> [defs.lisp], page 7.	
<b>Methods</b>	
variance (( <i>r-geometric0</i> [ <i>r-geometric</i> ], page 38))	[Method]
<b>Source</b> [geometric.lisp], page 16.	
variance (( <i>r-binomial0</i> [ <i>r-binomial</i> ], page 35))	[Method]
<b>Source</b> [binomial.lisp], page 15.	
variance (( <i>r-bernoulli0</i> [ <i>r-bernoulli</i> ], page 34))	[Method]
<b>Source</b> [bernoulli.lisp], page 15.	
variance (( <i>r-rayleigh0</i> [ <i>r-rayleigh</i> ], page 40))	[Method]
<b>Source</b> [rayleigh.lisp], page 14.	
variance (( <i>r-beta0</i> [ <i>r-beta</i> ], page 35))	[Method]
<b>Source</b> [beta.lisp], page 14.	
variance (( <i>r-inverse-gamma0</i> [ <i>r-inverse-gamma</i> ], page 38))	[Method]
<b>Source</b> [gamma.lisp], page 12.	
variance (( <i>r-gamma0</i> [ <i>r-gamma</i> ], page 37))	[Method]
<b>Source</b> [gamma.lisp], page 12.	
variance (( <i>r-t0</i> [ <i>r-t</i> ], page 40))	[Method]
<b>Source</b> [t-distribution.lisp], page 12.	
variance (( <i>left-truncated-normal0</i> [ <i>left-truncated-normal</i> ], page 59))	[Method]
<b>Source</b> [truncated-normal.lisp], page 11.	

<code>variance ((<i>r-log-normal0</i> [<i>r-log-normal</i>], page 39))</code>	[Method]
<b>Source</b> <code>[log-normal.lisp]</code> , page 10.	
<code>variance ((<i>r-normal0</i> [<i>r-normal</i>], page 39))</code>	[Method]
<b>Source</b> <code>[normal.lisp]</code> , page 9.	
<code>variance ((<i>r-exponential0</i> [<i>r-exponential</i>], page 36))</code>	[Method]
<b>Source</b> <code>[exponential.lisp]</code> , page 9.	
<code>variance ((<i>r-uniform0</i> [<i>r-uniform</i>], page 41))</code>	[Method]
<b>Source</b> <code>[uniform.lisp]</code> , page 8.	
<code>variance ((<i>instance</i> [<i>r-discrete</i>], page 36))</code>	[Method]
<b>Source</b> <code>[discrete.lisp]</code> , page 7.	

#### 4.1.4 Standalone methods

<code>initialize-instance :after ((<i>rng</i> [<i>generator</i>], page 42) <b>&amp;key</b> <i>seed</i> <b>&amp;allow-other-keys</b>)</code>	[Method]
<b>Source</b> <code>[generator.lisp]</code> , page 5.	
<code>initialize-instance :after ((<i>self</i> [<i>simple-multiplicative-congruential</i>], page 61) <b>&amp;key</b> <b>&amp;allow-other-keys</b>)</code>	[Method]
<b>Source</b> <code>[simple-multiplicative-congruential-generators.lisp]</code> , page 6.	
<code>num= ((<i>a1</i> [<i>r-inverse-gamma</i>], page 38) (<i>b2</i> [<i>r-inverse-gamma</i>], page 38) <b>&amp;optional</b> <i>tolerance3</i>)</code>	[Method]
<b>Package</b> <code>num-utils.num=.</code>	
<b>Source</b> <code>[gamma.lisp]</code> , page 12.	

#### 4.1.5 Structures

<code>r-bernoulli</code>	[Structure]
Bernoulli( <i>pr</i> ) distribution, with probability <i>PR</i> for success and 1- <i>PR</i> for failure.	
<b>Package</b> <code>[distributions]</code> , page 17.	
<b>Source</b> <code>[bernoulli.lisp]</code> , page 15.	
<b>Direct superclasses</b>	
<code>[r-univariate]</code> , page 60.	
<b>Direct methods</b>	
<ul style="list-style-type: none"> <li>• <code>[cdf]</code>, page 28.</li> <li>• <code>[draw]</code>, page 29.</li> <li>• <code>[mean]</code>, page 31.</li> <li>• <code>[pr]</code>, page 58.</li> <li>• <code>[variance]</code>, page 33.</li> </ul>	
<b>Direct slots</b>	
<code>pr</code>	[Slot]
<b>Type</b>	<code>distributions.internals:internal-float</code>
<b>Readers</b>	<code>[r-bernoulli-pr]</code> , page 51.
<b>Writers</b>	<code>[(setf r-bernoulli-pr)]</code> , page 51.

**r-beta** [Structure]  
Beta(alpha,beta) distribution, with density proportional to  $x^{(\alpha-1)}(1-x)^{(\beta-1)}$ .

**Package** [distributions], page 17.

**Source** [beta.lisp], page 14.

**Direct superclasses**  
[r-univariate], page 60.

**Direct methods**

- [alpha], page 27.
- [beta], page 27.
- [draw], page 29.
- [mean], page 31.
- [quantile], page 32.
- [variance], page 33.

**Direct slots**

alpha [Slot]

**Type** distributions.internals:internal-float

**Readers** [r-beta-alpha], page 51.

**Writers** [(setf r-beta-alpha)], page 51.

beta [Slot]

**Type** distributions.internals:internal-float

**Readers** [r-beta-beta], page 51.

**Writers** [(setf r-beta-beta)], page 51.

**r-binomial** [Structure]  
Binomial(pr,n) distribution, with N Bernoulli trials with probability PR for success.

**Package** [distributions], page 17.

**Source** [binomial.lisp], page 15.

**Direct superclasses**  
[r-univariate], page 60.

**Direct methods**

- [draw], page 28.
- [mean], page 31.
- [n], page 58.
- [pr], page 58.
- [variance], page 33.

**Direct slots**

pr [Slot]

**Type** distributions.internals:internal-float

**Readers** [r-binomial-pr], page 51.

**Writers** [(setf r-binomial-pr)], page 51.

<b>n</b>		[Slot]
<b>Type</b>	integer	
<b>Readers</b>	[r-binomial-n], page 51.	
<b>Writers</b>	[(setf r-binomial-n)], page 51.	
<b>r-discrete</b>		[Structure]
	Discrete probabilities.	
<b>Package</b>	[distributions], page 17.	
<b>Source</b>	[discrete.lisp], page 7.	
<b>Direct superclasses</b>	[r-univariate], page 60.	
<b>Direct methods</b>	<ul style="list-style-type: none"> <li>• [cdf], page 28.</li> <li>• [draw], page 29.</li> <li>• [log-pdf], page 30.</li> <li>• [mean], page 31.</li> <li>• [probabilities], page 32.</li> <li>• [variance], page 34.</li> </ul>	
<b>Direct slots</b>		
<b>probabilities</b>		[Slot]
<b>Type</b>	distributions.internals:float-vector	
<b>Readers</b>	[r-discrete-probabilities], page 52.	
<b>Writers</b>	[(setf r-discrete-probabilities)], page 52.	
<b>prob</b>		[Slot]
<b>Type</b>	distributions.internals:float-vector	
<b>Readers</b>	[r-discrete-prob], page 52.	
<b>Writers</b>	[(setf r-discrete-prob)], page 52.	
<b>alias</b>		[Slot]
<b>Type</b>	(simple-array fixnum (*))	
<b>Readers</b>	[r-discrete-alias], page 51.	
<b>Writers</b>	[(setf r-discrete-alias)], page 51.	
<b>n-float</b>		[Slot]
<b>Type</b>	distributions.internals:internal-float	
<b>Readers</b>	[r-discrete-n-float], page 52.	
<b>Writers</b>	[(setf r-discrete-n-float)], page 52.	
<b>r-exponential</b>		[Structure]
	Exponential(rate) distribution, with density $\text{rate} \cdot \exp(-\text{rate} \cdot x)$ for $x \geq 0$ and 0 for $x < 0$ . rate > 0.	
<b>Package</b>	[distributions], page 17.	



**Source** [exponential.lisp], page 9.

**Direct superclasses**  
[r-univariate], page 60.

**Direct methods**

- [cdf], page 28.
- [draw], page 29.
- [log-pdf], page 30.
- [mean], page 31.
- [quantile], page 32.
- [rate], page 32.
- [variance], page 34.

**Direct slots**

rate	[Slot]
<b>Type</b>	distributions.internals:internal-float
<b>Readers</b>	[r-exponential-rate], page 52.
<b>Writers</b>	[(setf r-exponential-rate)], page 52.

**r-gamma** [Structure]  
Gamma(alpha,beta) distribution, with density proportional to  $x^{(\text{alpha}-1)} \exp(-x*\text{beta})$ . Alpha and beta are known as shape and inverse scale (or rate) parameters, respectively.

**Package** [distributions], page 17.

**Source** [gamma.lisp], page 12.

**Direct superclasses**  
[r-univariate], page 60.

**Direct methods**

- [alpha], page 27.
- [beta], page 28.
- [cdf], page 28.
- [draw], page 29.
- [log-pdf], page 30.
- [mean], page 31.
- [nu], page 32.
- [quantile], page 32.
- [variance], page 33.

**Direct slots**

alpha	[Slot]
<b>Type</b>	distributions.internals:internal-float
<b>Readers</b>	[r-gamma-alpha], page 52.
<b>Writers</b>	[(setf r-gamma-alpha)], page 52.
beta	[Slot]
<b>Type</b>	distributions.internals:internal-float
<b>Readers</b>	[r-gamma-beta], page 52.
<b>Writers</b>	[(setf r-gamma-beta)], page 52.

**r-geometric** [Structure]

Geometric(pr) distribution.

**Package** [distributions], page 17.

**Source** [geometric.lisp], page 16.

**Direct superclasses**

[r-univariate], page 60.

**Direct methods**

- [draw], page 28.
- [mean], page 31.
- [pr], page 58.
- [variance], page 33.

**Direct slots**

**pr** [Slot]

**Type** distributions.internals:internal-float

**Readers** [r-geometric-pr], page 53.

**Writers** [(setf r-geometric-pr)], page 53.

**r-inverse-gamma** [Structure]

Inverse-Gamma(alpha,beta) distribution, with density  $p(x)$  proportional to  $x^{-(\alpha+1)} \exp(-\beta/x)$

**Package** [distributions], page 17.

**Source** [gamma.lisp], page 12.

**Direct superclasses**

[r-univariate], page 60.

**Direct methods**

- [alpha], page 27.
- [beta], page 28.
- [draw], page 29.
- [log-pdf], page 30.
- [mean], page 31.
- [nu], page 31.
- [num=], page 34.
- [s^2], page 58.
- [variance], page 33.

**Direct slots**

**alpha** [Slot]

**Type** distributions.internals:internal-float

**Readers** [r-inverse-gamma-alpha], page 53.

**Writers** [(setf r-inverse-gamma-alpha)], page 53.

**beta** [Slot]

**Type** distributions.internals:internal-float

**Readers** [r-inverse-gamma-beta], page 53.

**Writers** [(setf r-inverse-gamma-beta)], page 53.

**r-log-normal** [Structure]

Log-normal distribution with location log-mean and scale log-sd.

**Package** [distributions], page 17.

**Source** [log-normal.lisp], page 10.

**Direct superclasses**

[r-univariate], page 60.

**Direct methods**

- [cdf], page 28.
- [draw], page 29.
- [log-pdf], page 30.
- [mean], page 31.
- [quantile], page 32.
- [variance], page 34.

**Direct slots**

**log-mean** [Slot]

**Type** distributions.internals:internal-float

**Readers** [r-log-normal-log-mean], page 53.

**Writers** [(setf r-log-normal-log-mean)], page 53.

**log-sd** [Slot]

**Type** distributions.internals:internal-float

**Readers** [r-log-normal-log-sd], page 53.

**Writers** [(setf r-log-normal-log-sd)], page 53.

**r-normal** [Structure]

Normal(mean,variance) distribution.

**Package** [distributions], page 17.

**Source** [normal.lisp], page 9.

**Direct superclasses**

[r-univariate], page 60.

**Direct methods**

- [cdf], page 28.
- [draw], page 29.
- [log-pdf], page 30.
- [mean], page 31.
- [quantile], page 32.
- [sd], page 58.
- [variance], page 34.

**Direct slots**

**mean** [Slot]

**Type** distributions.internals:internal-float

**Readers** [r-normal-mean], page 54.

**Writers** [(setf r-normal-mean)], page 54.

**sd** [Slot]

**Type** `distributions.internals:internal-float`

**Readers** `[r-normal-sd]`, page 54.

**Writers** `[(setf r-normal-sd)]`, page 54.

**r-rayleigh** [Structure]

Rayleigh(scale) distribution with scale > 0 and density  $x * \exp(-x^2 / (2 \text{scale}^2)) / \text{scale}^2$  for  $x \geq 0$  and 0 for  $x < 0$ .

**Package** `[distributions]`, page 17.

**Source** `[rayleigh.lisp]`, page 14.

**Direct superclasses**  
`[r-univariate]`, page 60.

#### Direct methods

- `[cdf]`, page 28.
- `[draw]`, page 29.
- `[mean]`, page 31.
- `[scale]`, page 33.
- `[variance]`, page 33.

#### Direct slots

**scale** [Slot]

**Type** `distributions.internals:internal-float`

**Readers** `[r-rayleigh-scale]`, page 54.

**Writers** `[(setf r-rayleigh-scale)]`, page 54.

**r-t** [Structure]

T(mean,scale,nu) random variate.

**Package** `[distributions]`, page 17.

**Source** `[t-distribution.lisp]`, page 12.

**Direct superclasses**  
`[r-univariate]`, page 60.

#### Direct methods

- `[draw]`, page 29.
- `[mean]`, page 31.
- `[nu]`, page 32.
- `[scale]`, page 33.
- `[variance]`, page 33.

#### Direct slots

**mean** [Slot]

**Type** `distributions.internals:internal-float`

**Readers** `[r-t-mean]`, page 54.

**Writers** `[(setf r-t-mean)]`, page 54.

<b>scale</b>		[Slot]
<b>Type</b>	distributions.internals:internal-float	
<b>Readers</b>	[r-t-scale], page 55.	
<b>Writers</b>	[(setf r-t-scale)], page 55.	
<b>nu</b>		[Slot]
<b>Type</b>	distributions.internals:internal-float	
<b>Readers</b>	[r-t-nu], page 54.	
<b>Writers</b>	[(setf r-t-nu)], page 54.	
<b>r-uniform</b>		[Structure]
Uniform(left,right) distribution.		
<b>Package</b>	[distributions], page 17.	
<b>Source</b>	[uniform.lisp], page 8.	
<b>Direct superclasses</b>	[r-univariate], page 60.	
<b>Direct methods</b>	<ul style="list-style-type: none"> <li>• [cdf], page 28.</li> <li>• [draw], page 29.</li> <li>• [left], page 30.</li> <li>• [log-pdf], page 30.</li> <li>• [mean], page 31.</li> <li>• [quantile], page 32.</li> <li>• [right], page 32.</li> <li>• [variance], page 34.</li> </ul>	
<b>Direct slots</b>		
<b>left</b>		[Slot]
<b>Type</b>	distributions.internals:internal-float	
<b>Readers</b>	[r-uniform-left], page 55.	
<b>Writers</b>	[(setf r-uniform-left)], page 55.	
<b>right</b>		[Slot]
<b>Type</b>	distributions.internals:internal-float	
<b>Readers</b>	[r-uniform-right], page 55.	
<b>Writers</b>	[(setf r-uniform-right)], page 55.	
<b>width</b>		[Slot]
<b>Type</b>	distributions.internals:internal-float	
<b>Readers</b>	[r-uniform-width], page 55.	
<b>Writers</b>	[(setf r-uniform-width)], page 55.	

### 4.1.6 Classes

**borosh13** [Class]

Donald E. Knuth's Borosh-Niederreiter, The Art of Computer Programming, Volume 2, Third Edition, Addison-Wesley, pp 106-108.

**Package** [distributions], page 17.

**Source** [simple-multiplicative-congruential-generators.lisp], page 6.

**Direct superclasses**  
[simple-multiplicative-congruential], page 61.

**Direct slots**

**a** [Slot]

**Initform** 1812433253

**chunk-length** [Slot]

**Initform** 32

**generator** [Class]

Base class for random number generators.

**Package** [distributions], page 17.

**Source** [generator.lisp], page 5.

**Direct subclasses**  
[simple-multiplicative-congruential], page 61.

**Direct methods**

- [chunk-length], page 56.
- [copy-state], page 57.
- [default-seed], page 57.
- [generate-state], page 57.
- [initialize-instance], page 34.
- [(setf state)], page 59.
- [state], page 59.

**Direct slots**

**state** [Slot]

All information needed by the generator to create the next chunk of random bits. This state is modified after each call to NEXT-CHUNK.

**Initargs** :state

**Readers** [state], page 59.

**Writers** [(setf state)], page 59.

**min** [Slot]

The minimum value return by NEXT-CHUNK.

**Package** common-lisp.

**Initargs** :min

**max** [Slot]

The maximum value return by NEXT-CHUNK.

**Package** common-lisp.

**Initargs** :max

**chunk-length** [Slot]

The length in bits of the integer returned by NEXT-CHUNK.

**Readers** [chunk-length], page 56.

**Writers** *This slot is read-only.*

**default-seed** [Slot]

The seed used by default, when the seed is NIL.

**Initform** 0

**Readers** [default-seed], page 57.

**Writers** *This slot is read-only.*

**randu** [Class]

The poor IBM randu generator. Park and Miller, Random Number Generators: Good ones are hard to find, Communications of the ACM, October 1988, Volume 31, No 10, pp 1192-1201.

**Package** [distributions], page 17.

**Source** [simple-multiplicative-congruential-generators.lisp], page 6.

**Direct superclasses**  
[simple-multiplicative-congruential], page 61.

**Direct slots**

**a** [Slot]

**Initform** 65539

**chunk-length** [Slot]

**Initform** 31

**transputer** [Class]

INMOS Transputer Development System generator.

**Package** [distributions], page 17.

**Source** [simple-multiplicative-congruential-generators.lisp], page 6.

**Direct superclasses**  
[simple-multiplicative-congruential], page 61.

**Direct slots**

**a** [Slot]

**Initform** 1664525

**chunk-length** [Slot]

**Initform** 32

**waterman14** [Class]  
 Donald E. Knuth's Waterman, The Art of Computer Programming, Volume 2, Third Edition, Addison-Wesley, pp 106-108.

**Package** [distributions], page 17.

**Source** [simple-multiplicative-congruential-generators.lisp], page 6.

**Direct superclasses**  
 [simple-multiplicative-congruential], page 61.

**Direct slots**

**a** [Slot]

**Initform** 1566083941

**chunk-length** [Slot]

**Initform** 32

### 4.1.7 Types

**float-vector** (*&optional n*) [Type]

**Package** [distributions.internals], page 17.

**Source** [internals.lisp], page 5.

**internal-float** (*&optional lower-limit upper-limit*) [Type]

Type used for internal representation of floats in the DISTRIBUTIONS library.

**Package** [distributions.internals], page 17.

**Source** [internals.lisp], page 5.

## 4.2 Internals

### 4.2.1 Constants

**+normal-log-pdf-constant+** [Constant]

Normalizing constant for a standard normal PDF.

**Package** [distributions], page 17.

**Source** [normal.lisp], page 9.

### 4.2.2 Special variables

**\*default-generator-type\*** [Special Variable]

**Package** [distributions], page 17.

**Source** [generator.lisp], page 5.

### 4.2.3 Macros

**define-rv** (*name constructor-lambda-list options slots constructor-form* [Macro]  
*&rest methods*)

Define a random variable, abstracting from the representation. Syntax:

NAME is a symbol, and will name the class and the creation function.



CONSTRUCTOR-LAMBDA-LIST will be used to wrap the CONSTRUCTOR-FORM, which can use the locally define macro (MAKE :slot-name value1 ...) to initialize slots.

SLOTS is a list of (slot-name &key type read-only reader) slot specifications. When READER is T, SLOT-NAME is used instead, otherwise a method is defined using the given symbol.

OPTIONS is (&key documentation instance), the default instance is a gensym.

METHODS are (function-name lambda-list &body body), with (INSTANCE NAME) prepended to the lambda-list, ie the instance is accessible using INSTANCE. Also, within BODY, slots are accessible by their names.

**Package** [distributions], page 17.

**Source** [defs.lisp], page 7.

#### 4.2.4 Ordinary functions

**%next-double-float** (*limit rng*) [Function]

Handle the single or double float case of RANDOM. We generate a float in [0d0, 1d0) by clobbering the mantissa of 1d0 with random bits (52 bits); this yields a number in [1d0, 2d0). Then 1d0 is subtracted.

**Package** [distributions], page 17.

**Source** [generator.lisp], page 5.

**%next-integer** (*limit rng*) [Function]

Generates an integer greater than or equal to zero and less than LIMIT. Successive chunks are concatenated without overlap to construct integers larger than a single chunk. The return value has this property: If two integers are generated from the same RNG with LIMIT equal to  $2^m$  and  $2^n$ , respectively, then bit  $k$  is the same in both integers for  $0 \leq k < \min(m, n)$ . Each call to %NEXT-INTEGER consumes at least one chunk; bits left over from previous chunks are not re-used.

**Package** [distributions], page 17.

**Source** [generator.lisp], page 5.

**%next-single-float** (*limit rng*) [Function]

Handle the single or double float case of RANDOM. We generate a float in [0f0, 1f0) by clobbering the mantissa of 1f0 with random bits (23 bits); this yields a number in [1f0, 2f0). Then 1f0 is subtracted.

**Package** [distributions], page 17.

**Source** [generator.lisp], page 5.

**cdf-gamma%** (*x shape &key rate scale upper-tail log*) [Function]

CDF of Gamma with parameterisation like that of R pgamma

**Package** [distributions], page 17.

**Source** [gamma.lisp], page 12.

**cdf-gamma%+** (*x k  $\theta$* ) [Function]

Return the cumulative gamma distribution function, shape  $k > 0$ , scale  $\theta > 0$

**Package** [distributions], page 17.

**Source** [gamma.lisp], page 12.

- `cdf-normal%` (*x mu sigma*) [Function]  
Internal function for normal CDF.  
**Package** [distributions], page 17.  
**Source** [normal.lisp], page 9.
- `check-probability` (*p &optional open*) [Function]  
Assert that P is a probability (ie a real number between 0 and 1). When OPEN is given, it is checked that p is not 0 (:LEFT), 1 (:RIGHT), or 0/1 (:BOTH).  
**Package** [distributions], page 17.  
**Source** [defs.lisp], page 7.
- `copy-left-truncated-normal` (*instance*) [Function]  
**Package** [distributions], page 17.  
**Source** [truncated-normal.lisp], page 11.
- `copy-r-bernoulli` (*instance*) [Function]  
**Package** [distributions], page 17.  
**Source** [bernoulli.lisp], page 15.
- `copy-r-beta` (*instance*) [Function]  
**Package** [distributions], page 17.  
**Source** [beta.lisp], page 14.
- `copy-r-binomial` (*instance*) [Function]  
**Package** [distributions], page 17.  
**Source** [binomial.lisp], page 15.
- `copy-r-discrete` (*instance*) [Function]  
**Package** [distributions], page 17.  
**Source** [discrete.lisp], page 7.
- `copy-r-exponential` (*instance*) [Function]  
**Package** [distributions], page 17.  
**Source** [exponential.lisp], page 9.
- `copy-r-gamma` (*instance*) [Function]  
**Package** [distributions], page 17.  
**Source** [gamma.lisp], page 12.
- `copy-r-geometric` (*instance*) [Function]  
**Package** [distributions], page 17.  
**Source** [geometric.lisp], page 16.
- `copy-r-inverse-gamma` (*instance*) [Function]  
**Package** [distributions], page 17.  
**Source** [gamma.lisp], page 12.

- `copy-r-log-normal` (*instance*) [Function]  
  **Package** [distributions], page 17.  
  **Source** [log-normal.lisp], page 10.
- `copy-r-normal` (*instance*) [Function]  
  **Package** [distributions], page 17.  
  **Source** [normal.lisp], page 9.
- `copy-r-rayleigh` (*instance*) [Function]  
  **Package** [distributions], page 17.  
  **Source** [rayleigh.lisp], page 14.
- `copy-r-t` (*instance*) [Function]  
  **Package** [distributions], page 17.  
  **Source** [t-distribution.lisp], page 12.
- `copy-r-uniform` (*instance*) [Function]  
  **Package** [distributions], page 17.  
  **Source** [uniform.lisp], page 8.
- `copy-r-univariate` (*instance*) [Function]  
  **Package** [distributions], page 17.  
  **Source** [generics.lisp], page 7.
- `draw-bernoulli-bit` (*p &key rng*) [Function]  
  **Package** [distributions], page 17.  
  **Source** [bernoulli.lisp], page 15.
- `draw-left-truncated-standard-normal` (*left alpha &key rng*) [Function]  
  Draw a left truncated standard normal, using an  $\text{Exp}(\alpha, \text{left})$  distribution. LEFT is the standardized boundary, ALPHA should be calculated with TRUNCATED-NORMAL-OPTIMAL-ALPHA.  
  **Package** [distributions], page 17.  
  **Source** [truncated-normal.lisp], page 11.
- `draw-standard-gamma1` (*alpha d c &key rng*) [Function]  
  Return a standard gamma variate ( $\beta=1$ ) with shape parameter  $\alpha \geq 1$ . See Marsaglia and Tsang (2004). You should precalculate d and c using the utility function above.  
  **Package** [distributions], page 17.  
  **Source** [gamma.lisp], page 12.
- `generate-seed` () [Function]  
  Return a 64-bit random seed, based on current time.  
  **Package** [distributions], page 17.  
  **Source** [generator.lisp], page 5.

<code>left-truncated-normal</code> ( <i>mu sigma left</i> )	[Function]
<b>Package</b> [distributions], page 17.	
<b>Source</b> [truncated-normal.lisp], page 11.	
<code>left-truncated-normal-alpha</code> ( <i>instance</i> )	[Reader]
<code>(setf left-truncated-normal-alpha)</code> ( <i>instance</i> )	[Writer]
<b>Package</b> [distributions], page 17.	
<b>Source</b> [truncated-normal.lisp], page 11.	
<b>Target Slot</b>	
[alpha], page 60.	
<code>left-truncated-normal-left</code> ( <i>instance</i> )	[Reader]
<code>(setf left-truncated-normal-left)</code> ( <i>instance</i> )	[Writer]
<b>Package</b> [distributions], page 17.	
<b>Source</b> [truncated-normal.lisp], page 11.	
<b>Target Slot</b>	
[left], page 59.	
<code>left-truncated-normal-left-standardized</code> ( <i>instance</i> )	[Reader]
<code>(setf left-truncated-normal-left-standardized)</code> ( <i>instance</i> )	[Writer]
<b>Package</b> [distributions], page 17.	
<b>Source</b> [truncated-normal.lisp], page 11.	
<b>Target Slot</b>	
[left-standardized], page 60.	
<code>left-truncated-normal-m0</code> ( <i>instance</i> )	[Reader]
<code>(setf left-truncated-normal-m0)</code> ( <i>instance</i> )	[Writer]
<b>Package</b> [distributions], page 17.	
<b>Source</b> [truncated-normal.lisp], page 11.	
<b>Target Slot</b>	
[m0], page 60.	
<code>left-truncated-normal-mu</code> ( <i>instance</i> )	[Reader]
<code>(setf left-truncated-normal-mu)</code> ( <i>instance</i> )	[Writer]
<b>Package</b> [distributions], page 17.	
<b>Source</b> [truncated-normal.lisp], page 11.	
<b>Target Slot</b>	
[mu], page 59.	
<code>left-truncated-normal-p</code> ( <i>object</i> )	[Function]
<b>Package</b> [distributions], page 17.	
<b>Source</b> [truncated-normal.lisp], page 11.	
<code>left-truncated-normal-sigma</code> ( <i>instance</i> )	[Reader]
<code>(setf left-truncated-normal-sigma)</code> ( <i>instance</i> )	[Writer]
<b>Package</b> [distributions], page 17.	
<b>Source</b> [truncated-normal.lisp], page 11.	
<b>Target Slot</b>	
[sigma], page 59.	

<code>make-left-truncated-normal</code>	<code>(&amp;key mu sigma left left-standardized m0 alpha)</code>	[Function]
<b>Package</b>	[distributions], page 17.	
<b>Source</b>	[truncated-normal.lisp], page 11.	
<code>make-r-bernoulli</code>	<code>(&amp;key pr)</code>	[Function]
<b>Package</b>	[distributions], page 17.	
<b>Source</b>	[bernoulli.lisp], page 15.	
<code>make-r-beta</code>	<code>(&amp;key alpha beta)</code>	[Function]
<b>Package</b>	[distributions], page 17.	
<b>Source</b>	[beta.lisp], page 14.	
<code>make-r-binomial</code>	<code>(&amp;key pr n)</code>	[Function]
<b>Package</b>	[distributions], page 17.	
<b>Source</b>	[binomial.lisp], page 15.	
<code>make-r-discrete</code>	<code>(&amp;key probabilities prob alias n-float)</code>	[Function]
<b>Package</b>	[distributions], page 17.	
<b>Source</b>	[discrete.lisp], page 7.	
<code>make-r-exponential</code>	<code>(&amp;key rate)</code>	[Function]
<b>Package</b>	[distributions], page 17.	
<b>Source</b>	[exponential.lisp], page 9.	
<code>make-r-gamma</code>	<code>(&amp;key alpha beta)</code>	[Function]
<b>Package</b>	[distributions], page 17.	
<b>Source</b>	[gamma.lisp], page 12.	
<code>make-r-geometric</code>	<code>(&amp;key pr)</code>	[Function]
<b>Package</b>	[distributions], page 17.	
<b>Source</b>	[geometric.lisp], page 16.	
<code>make-r-inverse-gamma</code>	<code>(&amp;key alpha beta)</code>	[Function]
<b>Package</b>	[distributions], page 17.	
<b>Source</b>	[gamma.lisp], page 12.	
<code>make-r-log-normal</code>	<code>(&amp;key log-mean log-sd)</code>	[Function]
<b>Package</b>	[distributions], page 17.	
<b>Source</b>	[log-normal.lisp], page 10.	
<code>make-r-normal</code>	<code>(&amp;key mean sd)</code>	[Function]
<b>Package</b>	[distributions], page 17.	
<b>Source</b>	[normal.lisp], page 9.	
<code>make-r-rayleigh</code>	<code>(&amp;key scale)</code>	[Function]
<b>Package</b>	[distributions], page 17.	
<b>Source</b>	[rayleigh.lisp], page 14.	

- make-r-t** (*&key mean scale nu*) [Function]  
**Package** [distributions], page 17.  
**Source** [t-distribution.lisp], page 12.
- make-r-uniform** (*&key left right width*) [Function]  
**Package** [distributions], page 17.  
**Source** [uniform.lisp], page 8.
- make-r-univariate** (*&key*) [Function]  
**Package** [distributions], page 17.  
**Source** [generics.lisp], page 7.
- pdf-gamma** (*x a b*) [Function]  
 Return the probability density function of a gamma distribution with shape  $a > 0$ , scale  $b > 0$   
 Returns:  $x^{(a-1)} \exp(-x/b) / \text{gamma}(a) / b^a$ ,  $x > 0$   
**Package** [distributions], page 17.  
**Source** [gamma.lisp], page 12.
- pdf-gamma%** (*x shape scale*) [Function]  
**Package** [distributions], page 17.  
**Source** [gamma.lisp], page 12.
- pdf-gamma\*** (*x shape scale*) [Function]  
**Package** [distributions], page 17.  
**Source** [gamma.lisp], page 12.
- pdf-gamma+** (*x k  $\theta$* ) [Function]  
 Return the probability density function where:  
  
 K is the shape of the distribution  
 $\theta$  (theta) is the scale  
 X is the random variate  
**Package** [distributions], page 17.  
**Source** [gamma.lisp], page 12.
- pdf-normal%** (*x &key mu sigma*) [Function]  
 Direct calculation of the Probability Density of the normal distribution.  
**Package** [distributions], page 17.  
**Source** [normal.lisp], page 9.
- quantile-normal%** (*q mu sigma*) [Function]  
 Internal function for normal quantile.  
**Package** [distributions], page 17.  
**Source** [normal.lisp], page 9.
- r-bernoulli-p** (*object*) [Function]  
**Package** [distributions], page 17.  
**Source** [bernoulli.lisp], page 15.

<code>r-bernoulli-pr</code> ( <i>instance</i> )	[Reader]
<code>(setf r-bernoulli-pr)</code> ( <i>instance</i> )	[Writer]
Package [distributions], page 17.	
Source [bernoulli.lisp], page 15.	
Target Slot [pr], page 34.	
<code>r-beta-alpha</code> ( <i>instance</i> )	[Reader]
<code>(setf r-beta-alpha)</code> ( <i>instance</i> )	[Writer]
Package [distributions], page 17.	
Source [beta.lisp], page 14.	
Target Slot [alpha], page 35.	
<code>r-beta-beta</code> ( <i>instance</i> )	[Reader]
<code>(setf r-beta-beta)</code> ( <i>instance</i> )	[Writer]
Package [distributions], page 17.	
Source [beta.lisp], page 14.	
Target Slot [beta], page 35.	
<code>r-beta-p</code> ( <i>object</i> )	[Function]
Package [distributions], page 17.	
Source [beta.lisp], page 14.	
<code>r-binomial-n</code> ( <i>instance</i> )	[Reader]
<code>(setf r-binomial-n)</code> ( <i>instance</i> )	[Writer]
Package [distributions], page 17.	
Source [binomial.lisp], page 15.	
Target Slot [n], page 36.	
<code>r-binomial-p</code> ( <i>object</i> )	[Function]
Package [distributions], page 17.	
Source [binomial.lisp], page 15.	
<code>r-binomial-pr</code> ( <i>instance</i> )	[Reader]
<code>(setf r-binomial-pr)</code> ( <i>instance</i> )	[Writer]
Package [distributions], page 17.	
Source [binomial.lisp], page 15.	
Target Slot [pr], page 35.	
<code>r-discrete-alias</code> ( <i>instance</i> )	[Reader]
<code>(setf r-discrete-alias)</code> ( <i>instance</i> )	[Writer]
Package [distributions], page 17.	
Source [discrete.lisp], page 7.	
Target Slot [alias], page 36.	

<code>r-discrete-n-float</code> ( <i>instance</i> )	[Reader]
<code>(setf r-discrete-n-float)</code> ( <i>instance</i> )	[Writer]
Package [distributions], page 17.	
Source [discrete.lisp], page 7.	
Target Slot [n-float], page 36.	
<code>r-discrete-p</code> ( <i>object</i> )	[Function]
Package [distributions], page 17.	
Source [discrete.lisp], page 7.	
<code>r-discrete-prob</code> ( <i>instance</i> )	[Reader]
<code>(setf r-discrete-prob)</code> ( <i>instance</i> )	[Writer]
Package [distributions], page 17.	
Source [discrete.lisp], page 7.	
Target Slot [prob], page 36.	
<code>r-discrete-probabilities</code> ( <i>instance</i> )	[Reader]
<code>(setf r-discrete-probabilities)</code> ( <i>instance</i> )	[Writer]
Package [distributions], page 17.	
Source [discrete.lisp], page 7.	
Target Slot [probabilities], page 36.	
<code>r-exponential-p</code> ( <i>object</i> )	[Function]
Package [distributions], page 17.	
Source [exponential.lisp], page 9.	
<code>r-exponential-rate</code> ( <i>instance</i> )	[Reader]
<code>(setf r-exponential-rate)</code> ( <i>instance</i> )	[Writer]
Package [distributions], page 17.	
Source [exponential.lisp], page 9.	
Target Slot [rate], page 37.	
<code>r-gamma-alpha</code> ( <i>instance</i> )	[Reader]
<code>(setf r-gamma-alpha)</code> ( <i>instance</i> )	[Writer]
Package [distributions], page 17.	
Source [gamma.lisp], page 12.	
Target Slot [alpha], page 37.	
<code>r-gamma-beta</code> ( <i>instance</i> )	[Reader]
<code>(setf r-gamma-beta)</code> ( <i>instance</i> )	[Writer]
Package [distributions], page 17.	
Source [gamma.lisp], page 12.	
Target Slot [beta], page 37.	



<code>r-gamma-p</code> ( <i>object</i> )	[Function]
<b>Package</b> [distributions], page 17.	
<b>Source</b> [gamma.lisp], page 12.	
<code>r-geometric-p</code> ( <i>object</i> )	[Function]
<b>Package</b> [distributions], page 17.	
<b>Source</b> [geometric.lisp], page 16.	
<code>r-geometric-pr</code> ( <i>instance</i> )	[Reader]
( <code>setf r-geometric-pr</code> ) ( <i>instance</i> )	[Writer]
<b>Package</b> [distributions], page 17.	
<b>Source</b> [geometric.lisp], page 16.	
<b>Target Slot</b>	
[pr], page 38.	
<code>r-inverse-gamma-alpha</code> ( <i>instance</i> )	[Reader]
( <code>setf r-inverse-gamma-alpha</code> ) ( <i>instance</i> )	[Writer]
<b>Package</b> [distributions], page 17.	
<b>Source</b> [gamma.lisp], page 12.	
<b>Target Slot</b>	
[alpha], page 38.	
<code>r-inverse-gamma-beta</code> ( <i>instance</i> )	[Reader]
( <code>setf r-inverse-gamma-beta</code> ) ( <i>instance</i> )	[Writer]
<b>Package</b> [distributions], page 17.	
<b>Source</b> [gamma.lisp], page 12.	
<b>Target Slot</b>	
[beta], page 38.	
<code>r-inverse-gamma-p</code> ( <i>object</i> )	[Function]
<b>Package</b> [distributions], page 17.	
<b>Source</b> [gamma.lisp], page 12.	
<code>r-log-normal-log-mean</code> ( <i>instance</i> )	[Reader]
( <code>setf r-log-normal-log-mean</code> ) ( <i>instance</i> )	[Writer]
<b>Package</b> [distributions], page 17.	
<b>Source</b> [log-normal.lisp], page 10.	
<b>Target Slot</b>	
[log-mean], page 39.	
<code>r-log-normal-log-sd</code> ( <i>instance</i> )	[Reader]
( <code>setf r-log-normal-log-sd</code> ) ( <i>instance</i> )	[Writer]
<b>Package</b> [distributions], page 17.	
<b>Source</b> [log-normal.lisp], page 10.	
<b>Target Slot</b>	
[log-sd], page 39.	

<code>r-log-normal-p</code> ( <i>object</i> )	[Function]
<b>Package</b> [distributions], page 17.	
<b>Source</b> [log-normal.lisp], page 10.	
<code>r-normal-mean</code> ( <i>instance</i> )	[Reader]
( <code>setf r-normal-mean</code> ) ( <i>instance</i> )	[Writer]
<b>Package</b> [distributions], page 17.	
<b>Source</b> [normal.lisp], page 9.	
<b>Target Slot</b>	
[mean], page 39.	
<code>r-normal-p</code> ( <i>object</i> )	[Function]
<b>Package</b> [distributions], page 17.	
<b>Source</b> [normal.lisp], page 9.	
<code>r-normal-sd</code> ( <i>instance</i> )	[Reader]
( <code>setf r-normal-sd</code> ) ( <i>instance</i> )	[Writer]
<b>Package</b> [distributions], page 17.	
<b>Source</b> [normal.lisp], page 9.	
<b>Target Slot</b>	
[sd], page 40.	
<code>r-rayleigh-p</code> ( <i>object</i> )	[Function]
<b>Package</b> [distributions], page 17.	
<b>Source</b> [rayleigh.lisp], page 14.	
<code>r-rayleigh-scale</code> ( <i>instance</i> )	[Reader]
( <code>setf r-rayleigh-scale</code> ) ( <i>instance</i> )	[Writer]
<b>Package</b> [distributions], page 17.	
<b>Source</b> [rayleigh.lisp], page 14.	
<b>Target Slot</b>	
[scale], page 40.	
<code>r-t-mean</code> ( <i>instance</i> )	[Reader]
( <code>setf r-t-mean</code> ) ( <i>instance</i> )	[Writer]
<b>Package</b> [distributions], page 17.	
<b>Source</b> [t-distribution.lisp], page 12.	
<b>Target Slot</b>	
[mean], page 40.	
<code>r-t-nu</code> ( <i>instance</i> )	[Reader]
( <code>setf r-t-nu</code> ) ( <i>instance</i> )	[Writer]
<b>Package</b> [distributions], page 17.	
<b>Source</b> [t-distribution.lisp], page 12.	
<b>Target Slot</b>	
[nu], page 41.	

<b>r-t-p</b> ( <i>object</i> )	[Function]
<b>Package</b> [distributions], page 17.	
<b>Source</b> [t-distribution.lisp], page 12.	
<b>r-t-scale</b> ( <i>instance</i> )	[Reader]
(setf r-t-scale) ( <i>instance</i> )	[Writer]
<b>Package</b> [distributions], page 17.	
<b>Source</b> [t-distribution.lisp], page 12.	
<b>Target Slot</b>	
[ <i>scale</i> ], page 41.	
<b>r-uniform-left</b> ( <i>instance</i> )	[Reader]
(setf r-uniform-left) ( <i>instance</i> )	[Writer]
<b>Package</b> [distributions], page 17.	
<b>Source</b> [uniform.lisp], page 8.	
<b>Target Slot</b>	
[ <i>left</i> ], page 41.	
<b>r-uniform-p</b> ( <i>object</i> )	[Function]
<b>Package</b> [distributions], page 17.	
<b>Source</b> [uniform.lisp], page 8.	
<b>r-uniform-right</b> ( <i>instance</i> )	[Reader]
(setf r-uniform-right) ( <i>instance</i> )	[Writer]
<b>Package</b> [distributions], page 17.	
<b>Source</b> [uniform.lisp], page 8.	
<b>Target Slot</b>	
[ <i>right</i> ], page 41.	
<b>r-uniform-width</b> ( <i>instance</i> )	[Reader]
(setf r-uniform-width) ( <i>instance</i> )	[Writer]
<b>Package</b> [distributions], page 17.	
<b>Source</b> [uniform.lisp], page 8.	
<b>Target Slot</b>	
[ <i>width</i> ], page 41.	
<b>r-univariate-p</b> ( <i>object</i> )	[Function]
<b>Package</b> [distributions], page 17.	
<b>Source</b> [generics.lisp], page 7.	
<b>standard-gamma1-d-c</b> ( <i>alpha</i> )	[Function]
Return precalculated constants (values d c), useful for drawing from a gamma distribution.	
<b>Package</b> [distributions], page 17.	
<b>Source</b> [gamma.lisp], page 12.	

**truncated-normal-moments%** (*n mu sigma left right &optional m0*) [Function]

N=0 gives the total mass of the truncated normal, used for normalization, N=1 the mean, and N=2 the variance. where  $p(x)$  is the normal density. When LEFT or RIGHT are NIL, they are taken to be - or + infinity, respectively. M0 may be provided for efficiency if would be calculated multiple times. The formulas are from Jawitz (2004).

**Package** [distributions], page 17.

**Source** [truncated-normal.lisp], page 11.

**truncated-normal-optimal-alpha** (*left*) [Function]

Calculate optimal exponential parameter for left-truncated normals. LEFT is the standardized boundary.

**Package** [distributions], page 17.

**Source** [truncated-normal.lisp], page 11.

## 4.2.5 Generic functions

**a** (*object*) [Generic Reader]

**Package** [distributions], page 17.

**Methods**

**a** ((*simple-multiplicative-congruential* [Reader Method]  
[*simple-multiplicative-congruential*], page 61))

The multiplier of the sequence  $x(n+1) = A * x(n) \bmod M$ .

**Source** [simple-multiplicative-congruential-generators.lisp],  
page 6.

**Target Slot**  
[a], page 61.

**chunk-length** (*object*) [Generic Reader]

**Package** [distributions], page 17.

**Methods**

**chunk-length** ((*generator* [generator], page 42)) [Reader Method]

The length in bits of the integer returned by NEXT-CHUNK.

**Source** [generator.lisp], page 5.

**Target Slot**  
[chunk-length], page 43.

**clone** (*self*) [Generic Function]

**Package** [distributions], page 17.

**Methods**

**clone** ((*self* [simple-multiplicative-congruential], [Method]  
page 61))

**Source** [simple-multiplicative-congruential-generators.lisp],  
page 6.

**copy-state** (*rng*) [Generic Function]  
 Return a deep copy of RNG. The stream of random numbers drawn from RNG and its clone should be the same (given you draw according to the same distributions).

**Package** [distributions], page 17.

**Source** [generator.lisp], page 5.

**Methods**

`copy-state ((rng [generator], page 42))` [Method]

`copy-state ((rng random-state))` [Method]

**default-seed** (*object*) [Generic Reader]

**Package** [distributions], page 17.

**Methods**

`default-seed ((generator [generator], page 42))` [Reader Method]  
 The seed used by default, when the seed is NIL.

**Source** [generator.lisp], page 5.

**Target Slot**

[default-seed], page 43.

**generate-state** (*rng seed*) [Generic Function]

Return a state for a generator of RNG's type using seed.

**Package** [distributions], page 17.

**Source** [generator.lisp], page 5.

**Methods**

`generate-state ((self [simple-multiplicative-congruential], page 61) seed)` [Method]

**Source** [simple-multiplicative-congruential-generators.lisp],  
 page 6.

`generate-state ((rng [generator], page 42) seed)` [Method]

**m** (*object*) [Generic Reader]

**Package** [distributions], page 17.

**Methods**

`m ((simple-multiplicative-congruential [simple-multiplicative-congruential], page 61))` [Reader Method]

The modulo of the sequence  $x(n+1) = A * x(n) \bmod M$ .

**Source** [simple-multiplicative-congruential-generators.lisp],  
 page 6.

**Target Slot**

[m], page 61.

**n** (*r-binomial0*) [Generic Function]

**Package** [distributions], page 17.

**Methods**

- n** ((*r-binomial0* [*r-binomial*], page 35)) [Method]  
**Source** [binomial.lisp], page 15.
- next-chunk** (*self*) [Generic Function]  
**Package** [distributions], page 17.  
**Methods**
- next-chunk** ((*self* [simple-multiplicative-congruential], page 61)) [Method]  
**Source** [simple-multiplicative-congruential-generators.lisp], page 6.
- next-real** (*self*) [Generic Function]  
**Package** [distributions], page 17.  
**Methods**
- next-real** ((*self* [simple-multiplicative-congruential], page 61)) [Method]  
**Source** [simple-multiplicative-congruential-generators.lisp], page 6.
- pr** (*r-bernoulli0*) [Generic Function]  
**Package** [distributions], page 17.  
**Methods**
- pr** ((*r-geometric0* [*r-geometric*], page 38)) [Method]  
**Source** [geometric.lisp], page 16.
- pr** ((*r-binomial0* [*r-binomial*], page 35)) [Method]  
**Source** [binomial.lisp], page 15.
- pr** ((*r-bernoulli0* [*r-bernoulli*], page 34)) [Method]  
**Source** [bernoulli.lisp], page 15.
- s<sup>2</sup>** (*distribution*) [Generic Function]  
Return the scale when applicable.  
**Package** [distributions], page 17.  
**Source** [generics.lisp], page 7.  
**Methods**
- s<sup>2</sup>** ((*r-inverse-gamma* [*r-inverse-gamma*], page 38)) [Method]  
**Source** [chi-square.lisp], page 13.
- sd** (*r-normal0*) [Generic Function]  
**Package** [distributions], page 17.  
**Methods**
- sd** ((*r-normal0* [*r-normal*], page 39)) [Method]  
**Source** [normal.lisp], page 9.

`state` (*object*) [Generic Reader]  
`(setf state)` (*object*) [Generic Writer]

**Package** [distributions], page 17.

#### Methods

`state` ((*generator* [*generator*], page 42)) [Reader Method]

`(setf state)` ((*generator* [*generator*], page 42)) [Writer Method]

All information needed by the generator to create the next chunk of random bits. This state is modified after each call to NEXT-CHUNK.

**Source** [generator.lisp], page 5.

#### Target Slot

[state], page 42.

## 4.2.6 Structures

`left-truncated-normal` [Structure]

Truncated normal distribution with given mu and sigma (corresponds to the mean and standard deviation in the untruncated case, respectively), on the interval [left, infinity).

**Package** [distributions], page 17.

**Source** [truncated-normal.lisp], page 11.

#### Direct superclasses

[r-univariate], page 60.

#### Direct methods

- [cdf], page 28.
- [draw], page 29.
- [log-pdf], page 30.
- [mean], page 31.
- [quantile], page 32.
- [variance], page 33.

#### Direct slots

`mu` [Slot]

**Type** distributions.internals:internal-float

**Readers** [left-truncated-normal-mu], page 48.

**Writers** [(setf left-truncated-normal-mu)], page 48.

`sigma` [Slot]

**Type** distributions.internals:internal-float

**Readers** [left-truncated-normal-sigma], page 48.

**Writers** [(setf left-truncated-normal-sigma)], page 48.

`left` [Slot]

**Type** distributions.internals:internal-float

**Readers** [left-truncated-normal-left], page 48.

**Writers** [(setf left-truncated-normal-left)], page 48.

**left-standardized** [Slot]

**Type** `distributions.internals:internal-float`

**Readers** `[left-truncated-normal-left-standardized]`, page 48.

**Writers** `[(setf left-truncated-normal-left-standardized)]`, page 48.

**m0** [Slot]

**Type** `distributions.internals:internal-float`

**Readers** `[left-truncated-normal-m0]`, page 48.

**Writers** `[(setf left-truncated-normal-m0)]`, page 48.

**alpha** [Slot]

**Type** `distributions.internals:internal-float`

**Readers** `[left-truncated-normal-alpha]`, page 48.

**Writers** `[(setf left-truncated-normal-alpha)]`, page 48.

**r-univariate** [Structure]

Univariate distribution.

**Package** `[distributions]`, page 17.

**Source** `[generics.lisp]`, page 7.

**Direct superclasses**  
`structure-object`.

**Direct subclasses**

- `[left-truncated-normal]`, page 59.
- `[r-bernoulli]`, page 34.
- `[r-beta]`, page 35.
- `[r-binomial]`, page 35.
- `[r-discrete]`, page 36.
- `[r-exponential]`, page 36.
- `[r-gamma]`, page 37.
- `[r-geometric]`, page 38.
- `[r-inverse-gamma]`, page 38.
- `[r-log-normal]`, page 39.
- `[r-normal]`, page 39.
- `[r-rayleigh]`, page 40.
- `[r-t]`, page 40.
- `[r-uniform]`, page 41.

**Direct methods**

`[standard-deviation]`, page 33.



### 4.2.7 Classes

**simple-multiplicative-congruential** [Class]

A multiplicative congruential generator generates the sequence  $x(n+1) =$

$A * x(n) \bmod M$  and uses the seed as  $x(1)$ . A simple multiplicative congruential generator is a multiplicative congruential generator with  $M$  a power of 2. This allows to implement the modulo operation as a bitwise and operation of  $M-1$ , which is also the maximum value of a random chunk.

**Package** [distributions], page 17.

**Source** [simple-multiplicative-congruential-generators.lisp], page 6.

**Direct superclasses**  
[generator], page 42.

**Direct subclasses**

- [borosh13], page 42.
- [randu], page 43.
- [transputer], page 43.
- [waterman14], page 44.

**Direct methods**

- [a], page 56.
- [clone], page 56.
- [generate-state], page 57.
- [initialize-instance], page 34.
- [m], page 57.
- [next-chunk], page 58.
- [next-real], page 58.

**Direct slots**

**default-seed** [Slot]

**Initform** 1

**a** [Slot]

The multiplier of the sequence  $x(n+1) = A * x(n) \bmod M$ .

**Readers** [a], page 56.

**Writers** *This slot is read-only.*

**m** [Slot]

The modulo of the sequence  $x(n+1) = A * x(n) \bmod M$ .

**Readers** [m], page 57.

**Writers** *This slot is read-only.*



## Appendix A Indexes

### A.1 Concepts

(Index is nonexistent)

## A.2 Functions

### %

%next-double-float .....	45
%next-integer .....	45
%next-single-float .....	45

### (

(setf left-truncated-normal-alpha) .....	48
(setf left-truncated-normal-left) .....	48
(setf left-truncated-normal-left-standardized) ..	48
(setf left-truncated-normal-m0) .....	48
(setf left-truncated-normal-mu) .....	48
(setf left-truncated-normal-sigma) .....	48
(setf r-bernoulli-pr) .....	51
(setf r-beta-alpha) .....	51
(setf r-beta-beta) .....	51
(setf r-binomial-n) .....	51
(setf r-binomial-pr) .....	51
(setf r-discrete-alias) .....	51
(setf r-discrete-n-float) .....	52
(setf r-discrete-prob) .....	52
(setf r-discrete-probabilities) .....	52
(setf r-exponential-rate) .....	52
(setf r-gamma-alpha) .....	52
(setf r-gamma-beta) .....	52
(setf r-geometric-pr) .....	53
(setf r-inverse-gamma-alpha) .....	53
(setf r-inverse-gamma-beta) .....	53
(setf r-log-normal-log-mean) .....	53
(setf r-log-normal-log-sd) .....	53
(setf r-normal-mean) .....	54
(setf r-normal-sd) .....	54
(setf r-rayleigh-scale) .....	54
(setf r-t-mean) .....	54
(setf r-t-nu) .....	54
(setf r-t-scale) .....	55
(setf r-uniform-left) .....	55
(setf r-uniform-right) .....	55
(setf r-uniform-width) .....	55
(setf state) .....	59

### A

a .....	56
alpha .....	27
as-float .....	23
as-float-probabilities .....	23
as-float-vector .....	23

### B

beta .....	27, 28
------------	--------

### C

cdf .....	28
cdf-gamma% .....	45
cdf-gamma%+ .....	45
cdf-normal% .....	46
check-probability .....	46
chunk-length .....	56
clone .....	56
copy-left-truncated-normal .....	46
copy-r-bernoulli .....	46
copy-r-beta .....	46
copy-r-binomial .....	46
copy-r-discrete .....	46
copy-r-exponential .....	46
copy-r-gamma .....	46
copy-r-geometric .....	46
copy-r-inverse-gamma .....	46
copy-r-log-normal .....	47
copy-r-normal .....	47
copy-r-rayleigh .....	47
copy-r-t .....	47
copy-r-uniform .....	47
copy-r-univariate .....	47
copy-state .....	57

### D

default-seed .....	57
define-rv .....	44
distinct-random-integers .....	23
distinct-random-integers-dense .....	24
draw .....	28, 29
draw-bernoulli .....	24
draw-bernoulli-bit .....	47
draw-binomial .....	24
draw-exponential .....	24
draw-geometric .....	24
draw-left-truncated-standard-normal .....	47
draw-poisson .....	24
draw-rayleigh .....	24
draw-standard-exponential .....	24
draw-standard-gamma1 .....	47
draw-standard-normal .....	25
draw-standard-t .....	25
draw-uniform .....	25

### F

from-standard-normal .....	25
Function, %next-double-float .....	45
Function, %next-integer .....	45
Function, %next-single-float .....	45
Function, (setf left-truncated-normal-alpha) .....	48
Function, (setf left-truncated-normal-left) ..	48
Function, (setf left-truncated-normal-left-standardized) ..	48
Function, (setf left-truncated-normal-m0) .....	48
Function, (setf left-truncated-normal-mu) .....	48
Function, (setf left-truncated-normal-sigma) .....	48

Function, (setf r-bernoulli-pr) .....	51	Function, generate-seed .....	47
Function, (setf r-beta-alpha) .....	51	Function, left-truncated-normal .....	48
Function, (setf r-beta-beta) .....	51	Function, left-truncated-normal-alpha .....	48
Function, (setf r-binomial-n) .....	51	Function, left-truncated-normal-left .....	48
Function, (setf r-binomial-pr) .....	51	Function, left-truncated-normal-left-standardized .....	48
Function, (setf r-discrete-alias) .....	51	Function, left-truncated-normal-m0 .....	48
Function, (setf r-discrete-n-float) .....	52	Function, left-truncated-normal-mu .....	48
Function, (setf r-discrete-prob) .....	52	Function, left-truncated-normal-p .....	48
Function, (setf r-discrete-probabilities) .....	52	Function, left-truncated-normal-sigma .....	48
Function, (setf r-exponential-rate) .....	52	Function, make-generator .....	25
Function, (setf r-gamma-alpha) .....	52	Function, make-left-truncated-normal .....	49
Function, (setf r-gamma-beta) .....	52	Function, make-r-bernoulli .....	49
Function, (setf r-geometric-pr) .....	53	Function, make-r-beta .....	49
Function, (setf r-inverse-gamma-alpha) .....	53	Function, make-r-binomial .....	49
Function, (setf r-inverse-gamma-beta) .....	53	Function, make-r-discrete .....	49
Function, (setf r-log-normal-log-mean) .....	53	Function, make-r-exponential .....	49
Function, (setf r-log-normal-log-sd) .....	53	Function, make-r-gamma .....	49
Function, (setf r-normal-mean) .....	54	Function, make-r-geometric .....	49
Function, (setf r-normal-sd) .....	54	Function, make-r-inverse-gamma .....	49
Function, (setf r-rayleigh-scale) .....	54	Function, make-r-log-normal .....	49
Function, (setf r-t-mean) .....	54	Function, make-r-normal .....	49
Function, (setf r-t-nu) .....	54	Function, make-r-rayleigh .....	49
Function, (setf r-t-scale) .....	55	Function, make-r-t .....	50
Function, (setf r-uniform-left) .....	55	Function, make-r-uniform .....	50
Function, (setf r-uniform-right) .....	55	Function, make-r-univariate .....	50
Function, (setf r-uniform-width) .....	55	Function, next .....	25
Function, as-float .....	23	Function, pdf .....	25
Function, as-float-probabilities .....	23	Function, pdf-gamma .....	50
Function, as-float-vector .....	23	Function, pdf-gamma% .....	50
Function, cdf-gamma% .....	45	Function, pdf-gamma* .....	50
Function, cdf-gamma%+ .....	45	Function, pdf-gamma+ .....	50
Function, cdf-normal% .....	46	Function, pdf-normal% .....	50
Function, check-probability .....	46	Function, quantile-normal% .....	50
Function, copy-left-truncated-normal .....	46	Function, r-bernoulli .....	25
Function, copy-r-bernoulli .....	46	Function, r-bernoulli-p .....	50
Function, copy-r-beta .....	46	Function, r-bernoulli-pr .....	51
Function, copy-r-binomial .....	46	Function, r-beta .....	26
Function, copy-r-discrete .....	46	Function, r-beta-alpha .....	51
Function, copy-r-exponential .....	46	Function, r-beta-beta .....	51
Function, copy-r-gamma .....	46	Function, r-beta-p .....	51
Function, copy-r-geometric .....	46	Function, r-binomial .....	26
Function, copy-r-inverse-gamma .....	46	Function, r-binomial-n .....	51
Function, copy-r-log-normal .....	47	Function, r-binomial-p .....	51
Function, copy-r-normal .....	47	Function, r-binomial-pr .....	51
Function, copy-r-rayleigh .....	47	Function, r-chi-square .....	26
Function, copy-r-t .....	47	Function, r-discrete .....	26
Function, copy-r-uniform .....	47	Function, r-discrete-alias .....	51
Function, copy-r-univariate .....	47	Function, r-discrete-n-float .....	52
Function, distinct-random-integers .....	23	Function, r-discrete-p .....	52
Function, distinct-random-integers-dense .....	24	Function, r-discrete-prob .....	52
Function, draw-bernoulli .....	24	Function, r-discrete-probabilities .....	52
Function, draw-bernoulli-bit .....	47	Function, r-exponential .....	26
Function, draw-binomial .....	24	Function, r-exponential-p .....	52
Function, draw-exponential .....	24	Function, r-exponential-rate .....	52
Function, draw-geometric .....	24	Function, r-gamma .....	26
Function, draw-left-truncated-standard-normal .....	47	Function, r-gamma-alpha .....	52
Function, draw-poisson .....	24	Function, r-gamma-beta .....	52
Function, draw-rayleigh .....	24	Function, r-gamma-p .....	53
Function, draw-standard-exponential .....	24	Function, r-geometric .....	26
Function, draw-standard-gamma1 .....	47	Function, r-geometric-p .....	53
Function, draw-standard-normal .....	25	Function, r-geometric-pr .....	53
Function, draw-standard-t .....	25	Function, r-inverse-chi-square .....	26
Function, draw-uniform .....	25	Function, r-inverse-gamma .....	26
Function, from-standard-normal .....	25	Function, r-inverse-gamma-alpha .....	53

Function, <code>r-inverse-gamma-beta</code> .....	53
Function, <code>r-inverse-gamma-p</code> .....	53
Function, <code>r-log-normal</code> .....	26
Function, <code>r-log-normal-log-mean</code> .....	53
Function, <code>r-log-normal-log-sd</code> .....	53
Function, <code>r-log-normal-p</code> .....	54
Function, <code>r-normal</code> .....	26
Function, <code>r-normal-mean</code> .....	54
Function, <code>r-normal-p</code> .....	54
Function, <code>r-normal-sd</code> .....	54
Function, <code>r-rayleigh</code> .....	27
Function, <code>r-rayleigh-p</code> .....	54
Function, <code>r-rayleigh-scale</code> .....	54
Function, <code>r-t</code> .....	27
Function, <code>r-t-mean</code> .....	54
Function, <code>r-t-nu</code> .....	54
Function, <code>r-t-p</code> .....	55
Function, <code>r-t-scale</code> .....	55
Function, <code>r-truncated-normal</code> .....	27
Function, <code>r-uniform</code> .....	27
Function, <code>r-uniform-left</code> .....	55
Function, <code>r-uniform-p</code> .....	55
Function, <code>r-uniform-right</code> .....	55
Function, <code>r-uniform-width</code> .....	55
Function, <code>r-univariate-p</code> .....	55
Function, <code>standard-gamma1-d-c</code> .....	55
Function, <code>t-scale-to-variance-coefficient</code> ....	27
Function, <code>to-standard-normal</code> .....	27
Function, <code>truncated-normal-moments%</code> .....	56
Function, <code>truncated-normal-optimal-alpha</code> ....	56

## G

<code>generate-seed</code> .....	47
<code>generate-state</code> .....	57
<code>generator</code> .....	29
Generic Function, <code>(setf state)</code> .....	59
Generic Function, <code>a</code> .....	56
Generic Function, <code>alpha</code> .....	27
Generic Function, <code>beta</code> .....	27
Generic Function, <code>cdf</code> .....	28
Generic Function, <code>chunk-length</code> .....	56
Generic Function, <code>clone</code> .....	56
Generic Function, <code>copy-state</code> .....	57
Generic Function, <code>default-seed</code> .....	57
Generic Function, <code>draw</code> .....	28
Generic Function, <code>generate-state</code> .....	57
Generic Function, <code>generator</code> .....	29
Generic Function, <code>left</code> .....	29
Generic Function, <code>log-pdf</code> .....	30
Generic Function, <code>m</code> .....	57
Generic Function, <code>mean</code> .....	30
Generic Function, <code>n</code> .....	57
Generic Function, <code>next-chunk</code> .....	58
Generic Function, <code>next-real</code> .....	58
Generic Function, <code>nu</code> .....	31
Generic Function, <code>pr</code> .....	58
Generic Function, <code>probabilities</code> .....	32
Generic Function, <code>quantile</code> .....	32
Generic Function, <code>rate</code> .....	32
Generic Function, <code>right</code> .....	32
Generic Function, <code>s<sup>2</sup></code> .....	58
Generic Function, <code>scale</code> .....	33
Generic Function, <code>sd</code> .....	58

Generic Function, <code>standard-deviation</code> .....	33
Generic Function, <code>state</code> .....	59
Generic Function, <code>variance</code> .....	33

## I

<code>initialize-instance</code> .....	34
--	----

## L

<code>left</code> .....	29, 30
<code>left-truncated-normal</code> .....	48
<code>left-truncated-normal-alpha</code> .....	48
<code>left-truncated-normal-left</code> .....	48
<code>left-truncated-normal-left-standardized</code> ....	48
<code>left-truncated-normal-m0</code> .....	48
<code>left-truncated-normal-mu</code> .....	48
<code>left-truncated-normal-p</code> .....	48
<code>left-truncated-normal-sigma</code> .....	48
<code>log-pdf</code> .....	30

## M

<code>m</code> .....	57
Macro, <code>define-rv</code> .....	44
Macro, <code>maybe-ignore-constant</code> .....	23
Macro, <code>try</code> .....	23
Macro, <code>with-floats</code> .....	23
<code>make-generator</code> .....	25
<code>make-left-truncated-normal</code> .....	49
<code>make-r-bernoulli</code> .....	49
<code>make-r-beta</code> .....	49
<code>make-r-binomial</code> .....	49
<code>make-r-discrete</code> .....	49
<code>make-r-exponential</code> .....	49
<code>make-r-gamma</code> .....	49
<code>make-r-geometric</code> .....	49
<code>make-r-inverse-gamma</code> .....	49
<code>make-r-log-normal</code> .....	49
<code>make-r-normal</code> .....	49
<code>make-r-rayleigh</code> .....	49
<code>make-r-t</code> .....	50
<code>make-r-uniform</code> .....	50
<code>make-r-univariate</code> .....	50
<code>maybe-ignore-constant</code> .....	23
<code>mean</code> .....	30, 31
Method, <code>(setf state)</code> .....	59
Method, <code>a</code> .....	56
Method, <code>alpha</code> .....	27
Method, <code>beta</code> .....	27, 28
Method, <code>cdf</code> .....	28
Method, <code>chunk-length</code> .....	56
Method, <code>clone</code> .....	56
Method, <code>copy-state</code> .....	57
Method, <code>default-seed</code> .....	57
Method, <code>draw</code> .....	28, 29
Method, <code>generate-state</code> .....	57
Method, <code>generator</code> .....	29
Method, <code>initialize-instance</code> .....	34
Method, <code>left</code> .....	30
Method, <code>log-pdf</code> .....	30
Method, <code>m</code> .....	57
Method, <code>mean</code> .....	31
Method, <code>n</code> .....	58

Method, next-chunk .....	58
Method, next-real .....	58
Method, nu .....	31, 32
Method, num= .....	34
Method, pr .....	58
Method, probabilities .....	32
Method, quantile .....	32
Method, rate .....	32
Method, right .....	32
Method, s <sup>2</sup> .....	58
Method, scale .....	33
Method, sd .....	58
Method, standard-deviation .....	33
Method, state .....	59
Method, variance .....	33, 34

## N

n .....	57, 58
next .....	25
next-chunk .....	58
next-real .....	58
nu .....	31, 32
num= .....	34

## P

pdf .....	25
pdf-gamma .....	50
pdf-gamma% .....	50
pdf-gamma* .....	50
pdf-gamma+ .....	50
pdf-normal% .....	50
pr .....	58
probabilities .....	32

## Q

quantile .....	32
quantile-normal% .....	50

## R

r-bernoulli .....	25
r-bernoulli-p .....	50
r-bernoulli-pr .....	51
r-beta .....	26
r-beta-alpha .....	51
r-beta-beta .....	51
r-beta-p .....	51
r-binomial .....	26
r-binomial-n .....	51
r-binomial-p .....	51
r-binomial-pr .....	51
r-chi-square .....	26
r-discrete .....	26
r-discrete-alias .....	51
r-discrete-n-float .....	52
r-discrete-p .....	52
r-discrete-prob .....	52
r-discrete-probabilities .....	52
r-exponential .....	26
r-exponential-p .....	52

r-exponential-rate .....	52
r-gamma .....	26
r-gamma-alpha .....	52
r-gamma-beta .....	52
r-gamma-p .....	53
r-geometric .....	26
r-geometric-p .....	53
r-geometric-pr .....	53
r-inverse-chi-square .....	26
r-inverse-gamma .....	26
r-inverse-gamma-alpha .....	53
r-inverse-gamma-beta .....	53
r-inverse-gamma-p .....	53
r-log-normal .....	26
r-log-normal-log-mean .....	53
r-log-normal-log-sd .....	53
r-log-normal-p .....	54
r-normal .....	26
r-normal-mean .....	54
r-normal-p .....	54
r-normal-sd .....	54
r-rayleigh .....	27
r-rayleigh-p .....	54
r-rayleigh-scale .....	54
r-t .....	27
r-t-mean .....	54
r-t-nu .....	54
r-t-p .....	55
r-t-scale .....	55
r-truncated-normal .....	27
r-uniform .....	27
r-uniform-left .....	55
r-uniform-p .....	55
r-uniform-right .....	55
r-uniform-width .....	55
r-univariate-p .....	55
rate .....	32
right .....	32

## S

s <sup>2</sup> .....	58
scale .....	33
sd .....	58
standard-deviation .....	33
standard-gamma1-d-c .....	55
state .....	59

## T

t-scale-to-variance-coefficient .....	27
to-standard-normal .....	27
truncated-normal-moments% .....	56
truncated-normal-optimal-alpha .....	56
try .....	23

## V

variance .....	33, 34
----------------	--------

## W

with-floats .....	23
-------------------	----

## A.3 Variables

**\***

`*default-generator-type*` ..... 44

**+**

`+normal-log-pdf-constant+` ..... 44

**A**

`a` ..... 42, 43, 44, 61

`alias` ..... 36

`alpha` ..... 35, 37, 38, 60

**B**

`beta` ..... 35, 37, 38

**C**

`chunk-length` ..... 42, 43, 44

`Constant, +normal-log-pdf-constant+` ..... 44

**D**

`default-seed` ..... 43, 61

**L**

`left` ..... 41, 59

`left-standardized` ..... 60

`log-mean` ..... 39

`log-sd` ..... 39

**M**

`m` ..... 61

`m0` ..... 60

`max` ..... 43

`mean` ..... 39, 40

`min` ..... 42

`mu` ..... 59

**N**

`n` ..... 36

`n-float` ..... 36

`nu` ..... 41

**P**

`pr` ..... 34, 35, 38

`prob` ..... 36

`probabilities` ..... 36

**R**

`rate` ..... 37

`right` ..... 41

**S**

`scale` ..... 40, 41

`sd` ..... 40

`sigma` ..... 59

`Slot, a` ..... 42, 43, 44, 61

`Slot, alias` ..... 36

`Slot, alpha` ..... 35, 37, 38, 60

`Slot, beta` ..... 35, 37, 38

`Slot, chunk-length` ..... 42, 43, 44

`Slot, default-seed` ..... 43, 61

`Slot, left` ..... 41, 59

`Slot, left-standardized` ..... 60

`Slot, log-mean` ..... 39

`Slot, log-sd` ..... 39

`Slot, m` ..... 61

`Slot, m0` ..... 60

`Slot, max` ..... 43

`Slot, mean` ..... 39, 40

`Slot, min` ..... 42

`Slot, mu` ..... 59

`Slot, n` ..... 36

`Slot, n-float` ..... 36

`Slot, nu` ..... 41

`Slot, pr` ..... 34, 35, 38

`Slot, prob` ..... 36

`Slot, probabilities` ..... 36

`Slot, rate` ..... 37

`Slot, right` ..... 41

`Slot, scale` ..... 40, 41

`Slot, sd` ..... 40

`Slot, sigma` ..... 59

`Slot, state` ..... 42

`Slot, width` ..... 41

`Special Variable, *default-generator-type*` ..... 44

`state` ..... 42

**W**

`width` ..... 41



## A.4 Data types

### B

bernoulli.lisp.....	15
beta.lisp.....	14
binomial.lisp.....	15
borosh13.....	42

### C

chi-square.lisp.....	13
Class, borosh13.....	42
Class, generator.....	42
Class, randu.....	43
Class, simple-multiplicative-congruential.....	61
Class, transputer.....	43
Class, waterman14.....	44

### D

defs.lisp.....	7
discrete.lisp.....	7
distributions.....	3, 17
distributions.asd.....	5
distributions.internals.....	17

### E

exponential.lisp.....	9
-----------------------	---

### F

File, bernoulli.lisp.....	15
File, beta.lisp.....	14
File, binomial.lisp.....	15
File, chi-square.lisp.....	13
File, defs.lisp.....	7
File, discrete.lisp.....	7
File, distributions.asd.....	5
File, exponential.lisp.....	9
File, gamma.lisp.....	12
File, generator.lisp.....	5
File, generics.lisp.....	7
File, geometric.lisp.....	16
File, internals.lisp.....	5
File, log-normal.lisp.....	10
File, normal.lisp.....	9
File, packages.lisp.....	5
File, poisson.lisp.....	16
File, rayleigh.lisp.....	14
File, simple-multiplicative-congruential-generators.lisp.....	6
File, t-distribution.lisp.....	12
File, truncated-normal.lisp.....	11
File, uniform.lisp.....	8
float-vector.....	44

### G

gamma.lisp.....	12
generator.....	42
generator.lisp.....	5
generics.lisp.....	7
geometric.lisp.....	16

### I

internal-float.....	44
internals.lisp.....	5

### L

left-truncated-normal.....	59
log-normal.lisp.....	10

### N

normal.lisp.....	9
------------------	---

### P

Package, distributions.....	17
Package, distributions.internals.....	17
packages.lisp.....	5
poisson.lisp.....	16

### R

r-bernoulli.....	34
r-beta.....	35
r-binomial.....	35
r-discrete.....	36
r-exponential.....	36
r-gamma.....	37
r-geometric.....	38
r-inverse-gamma.....	38
r-log-normal.....	39
r-normal.....	39
r-rayleigh.....	40
r-t.....	40
r-uniform.....	41
r-univariate.....	60
randu.....	43
rayleigh.lisp.....	14

**S**

simple-multiplicative-congruential.....	61
simple-multiplicative- congruential-generators.lisp.....	6
Structure, left-truncated-normal.....	59
Structure, r-bernoulli.....	34
Structure, r-beta.....	35
Structure, r-binomial.....	35
Structure, r-discrete.....	36
Structure, r-exponential.....	36
Structure, r-gamma.....	37
Structure, r-geometric.....	38
Structure, r-inverse-gamma.....	38
Structure, r-log-normal.....	39
Structure, r-normal.....	39
Structure, r-rayleigh.....	40
Structure, r-t.....	40
Structure, r-uniform.....	41

Structure, r-univariate.....	60
System, distributions.....	3

**T**

t-distribution.lisp.....	12
transputer.....	43
truncated-normal.lisp.....	11
Type, float-vector.....	44
Type, internal-float.....	44

**U**

uniform.lisp.....	8
-------------------	---

**W**

waterman14.....	44
-----------------	----