Ircam
documentation

- Etudes et recherches
- Œuvres musicales
- Logiciels

# PatchWork

## Alea Library

reference

*Third English Edition, April 1996*

IRCAM Centre Georges Pompidou

This manual was written by Mikhail Malt,
in collaboration with Curtis Roads,
translated by Frank Pecquet, and
produced under the editorial responsibility of
Marc Battier - Marketing Office, Ircam.

Patchwork was conceived and programmed by
Mikael Laurson, Camilo Rueda, and Jacques Duthen.

The Alea library conceived and programmed by Mikhail Malt.

3rd edition of the documentation, April 1996.
This documentation corresponds to version 2.0 or higher of PatchWork, and version 1.0 of the Alea library.

Ircam
1, place Igor-Stravinsky
F-75004 Paris
Tel. (33) (1) 44 78 49 62
Fax (33) (1) 42 77 29 47
E-mail ircam-doc@ircam.fr

# IRCAM Users' group

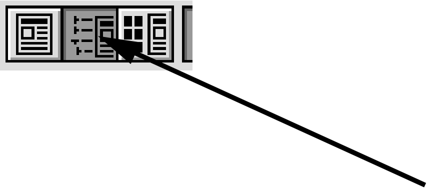
The use of this software and its documentation is restricted to members of the Ircam software users' group. For any supplementary information, contact:

Département de la Valorisation
Ircam
Place Stravinsky, F-75004 Paris

Tel. (1) 44 78 49 62
Fax (1) 42 77 29 47
E-mail: bousac@ircam.fr

Send comments or suggestions to the editor:

E-mail: bam@ircam.fr
Mail: Marc Battier,
Ircam, Département de la Valorisation
Place Stravinsky, F-75004 Paris



To see the table of contents of this manual, click on the Bookmark Button located in the Viewing section of the Adobe Acrobat Reader toolbar.

# Contents

# Résumé

La librairie Alea pour PatchWork regroupe un grand nombre de fonctions destinées à la génération de nombres selon des procédures adaptées de modèles stochastiques.

Les fonctions sont classées en quatre catégories :

**Distributions**

**Sequences**

**Random walks**

**Tools**

Le lecteur peut, si besoin est, se reporter aux ouvrages figurant dans la bibliographie.

# Introduction

The Alea library consists of a series of PatchWork modules to generate numbers according to several stochastic models. The library is divided into four parts:

**Distributions**

**Sequences**

**Random walks**

**Tools**

Most of these module boxes have been designed to remain very close to the basic mathematical models, since all musical applications generate a series of constraints that could jeopardize the coherency between the model and its practical use.

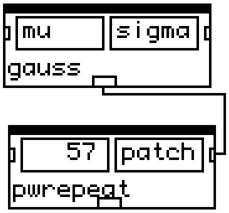
All probability distribution models are validated by The Law of Large Numbers. This means that we cannot really be sure that a series of algorithmically-generated numbers follows a specific probability distribution if this series is not long enough, or not infinite. Another  problem is the lack of boundaries in several of these models. Users may need to constrain the model with specific boundaries.

From these considerations one may understand that any kind of constraint—number of elements, scales, boundaries, etc.—could contradict the intent of the underlying stochastic model. Therefore, users should make their own choices and set their constraints according to their needs and priorities.

It is advised to consult the books by Ames (1991), Dodge (1985), and Lorrain (1980) if you want to learn about the models presented herein.

# Distributions

Distributions modules generate numbers according to a specific function distribution. Each of these modules only generates one value at a time. In order to generate a series of values, that is, a list, one can link these modules to a module called **pwrepeat**:



Some bounded distributions have scaling parameters in order to stretch, compress, and shift the resulting spectrum. These parameters are:

*alpha* (scale factor)

given the generating interval

*inter* = *upper boundary* **-** *lower boundary*

*new interval* = *inter* * *alpha*

therefore,

*alpha* < 1: interval compressed

*alpha* > 1: interval stretched

*beta* (shift factor)

Interval transformation

from *upper boundary* .....*lower boundary*

to

*upper boundary*+**beta** .....l*owerboundary*+**beta**

# ran *Generates pseudorandom number [0, 1] inclusive*

### Syntax

(alea::ran )

### Inputs

none

### Output

float

Generates a variable *x* uniformly distributed, such as 0 =< *x* =< 1.

# ran01 *Generates pseudorandom number [0, 1] exclusive*

### Syntax

(alea::ran01)

### Inputs

none

### Output

float

Generates a variable $x$ uniformly distributed, such that $0 < x < 1$.

# choix *Selects between two alternatives*

**Syntax**

(alea::choix x1 x2 px1)

Inputs

| | |
|---|---|
| *x1* | numbers? |
| *x2* | numbers? |
| *Px1* | numbers? |

**Output**

numbers?

Selects between two alternatives *x1* and *x2*, with complementary probabilities:

Probability of *x1 = px1,*

Probability of *x2* = 1-*px1*.

# **choix multiple** *Selects among several alternatives*

**Syntax**

(alea::choixmultiple vectprob &rest ;listobjets )

**Inputs**

| | |
|---|---|
| *vectprob* | numbers? |
| *listobjets* | list |

**Output**

list

Choice between severals alternatives (*listobjets)* from a vector of probability *vectprob*. This box is extensible, and the optional inputs must be open.

# **distexp** *exponential distribution*

**Syntax**

(alea::distexp   lambda )

**Inputs**

| | |
|---|---|
| *lambda* | fix/float |

**Output**

float

| | |
|---|---|
| lower boundary | 0 |
| upper boundary | +∞ |

Generates a number with an exponential distribution of density *lambda*.

# expobi *bilateral exponential distribution*

### Syntax

(alea::expobi lambda mu)

### Inputs

| | |
|---|---|
| *lambda* | fix/float |
| *mu* | fix/float |

### Output

float

| | |
|---|---|
| lower boundary | **-**∞ |
| upper boundary | +∞ |

Generates a number with a bilateral exponential distribution of density *lambda* with an average *mu.*

# distilin *linear distribution*

### Syntax

(alea::distilin g)

### Inputs

| | |
|---|---|
| *g* | fix/float |

### Output

float

| | |
|---|---|
| lower boundary | 0 |
| upper boundary | g |

Generates a number with a linear distribution of parameter *g*.

# **distcauchy** *cauchy distribution*

### Syntax

(alea::cauchy  alpha)

### Inputs

| | |
|---|---|
| *alpha* | fix/float |

### Output

float

| | |
|---|---|
| lower boundary | -∞ |
| upper boundary | +∞ |

Generates a number with a cauchy distribution with a parameter *alpha* .

# distlog *logistic distribution*

### Syntax

(alea::distlog alpha beta)

### Inputs

| | |
|---|---|
| *alpha* | fix/float |
| *beta* | fix/float |

### Output

float

| | |
|---|---|
| lower boundary- | ∞ |
| upper boundary | +∞ |

Generates a number with a logistic distribution of parameters *alpha* (the dispersion is proportionally inversed to *alpha*) and *beta*. The "mode" is located at (-*beta/alpha*).

# **distcshp** *hyperbolic cosinus distribution*

**Syntax**

(alea::distcshp alpha beta)

**Inputs**

| | |
|---|---|
| *alpha* | fix/float |
| *beta* | fix/float |

**Output**

float

| | |
|---|---|
| lower boundary | -∞ |
| upper boundary | +∞ |

Generates a number with a hyperbolic cosinus distribution. *alpha* is a scaling factor and *beta* a shifting factor.

# **distarsin** *arc sinus distribution*

**Syntax**

(alea::distarsin alpha beta)

**Inputs**

| | |
|---|---|
| *alpha* | fix/float |
| *beta* | fix/float |

**Output**

float

| | |
|---|---|
| lower boundary | beta |
| upper boundary | alpha |

Generates a number with a arc sinus distribution. *alpha* is a scaling factor and *beta* a shifting factor. For *alpha*= 1 and *beta*= 0, this distribution is identical to the Beta distribution .

# poison *Poisson distribution*

### Syntax

(alea::poisson  lmbd)

### Inputs

| | |
|---|---|
| *lmbd* | fix/float |

### Output

float

| | |
|---|---|
| lower boundary | 0 |
| upper boundary | +∞ |

Generates a number with a Poisson distribution of average *lmbd*.

# **triang** *triangular distribution*

### Syntax

(alea::triang  alpha beta)

### Inputs

| | |
|---|---|
| *alpha* | fix/float |
| *beta* | fix/float |

### Output

float

| | |
|---|---|
| lower boundary | beta |
| upper boundary | alpha |

Generates a number with a triangular distribution of average .5 (for *alpha*=1 and *beta*=0) ,where *alpha* is a scaling factor and *beta* a shifting factor.

# **gauss** *gaussian distribution*

### Syntax

(alea::gauss mu sigma)

### Inputs

| | |
|---|---|
| *mu* | fix/float |
| *sigma* | fix/float |

### Output

float

| | |
|---|---|
| lower boundary | -∞ ,i.e.,-> -6*sigma |
| upper boundary | +∞ ,i.e.,-> +6*sigma |

Generates a number with a gaussian distribution of an average *mu* and a bandwidth *sigma*. It is important to know that the gaussian distribution is not bounded, and that 99.74 % of the results falls between -3*sigma and +3*sigma, but for the present algorithm, the results will be bounded between -6*sigma and +6*sigma. In most cases this aproximation is acceptable, since only two results out of a billion fall out of these limits in a true gaussian processus.

# **weilbull** *Weilbull distribution*

### Syntax

(alea::weilbull s t)

### Inputs

| | |
|---|---|
| *s* | fix/float |
| *t* | fix/float |

### Output

float

| | |
|---|---|
| lower boundary | -∞ |
| upper boundary | +∞ |

Generates a number with a Weilbull distribution of parameters *s* and *t*, where *s* is a horizontal scaling factor and *t* controls the distribution morphology. For *t*=3.2, this distribution approaches the gaussian distribution. 99.9 % of the results are below s*6.9 $^{(1/t)}$.

# gamma *Gamma distribution*

**Syntax**

(alea::gamma nu)

**Inputs**

| | |
|---|---|
| *nu* | fix >0 |

**Output**

float

| | |
|---|---|
| lower boundary | 0 |
| upper boundary | +∞ |

Generates a number with a gamma distribution of parameter *nu*.

Since the present algorithm only functions witth *nu* as an integer number, for floating numbers one need multiply the variable which is generated by some factor that transforms nu1 to nu2: nu2 = nu1 * factor.

# **beta** *beta distribution*

**Syntax**

(alea::beta a b)

**Inputs**

| | |
|---|---|
| *a* | fix/float |
| *b* | fix/float |

**Output**

float

| | |
|---|---|
| lower boundary | 0 |
| upper boundary | 1 |

Generates a number with a beta distribution. For *a* = *b* = 1 the result is a continuous uniform distribution, for *a* and *b* greater than 1 the result is similar to a gaussian distribution.

# Sequences

This package contains four modules that generate lists of pitches (in midicents) as an example of application of three distributions: *uniform*, *linear* and *triangular*.

## not-centr *Returns an aleatoric pitch value*

### Syntax

(alea::not-centr nc intv)

### Inputs

| | |
|---|---|
| *nc:* | midicents |
| *intv:* | fix |

### Output

chord

**not-centr** returns an aleatoric pitch value between *nc*-*intv* and *nc+intv*.

# alea-seq *aleatoric sequence of uniform distribution*

### Syntax

(alea::alea-seq nc intv long)

### Inputs

| | |
|---|---|
| *nc:* | midicents |
| *intv:* | fix |
| *long:* | fix>0 |

### Output

chord

Aleatoric sequence of a uniform distribution, between *nc*-*intv* and *nc+intv* with a length *long*.

# **linea-seq** *aleatoric sequence of a linear distribution*

### Syntax

(alea::linea-seq limsup liminf long)

### Inputs

| | |
|---|---|
| *limsup* | midicents |
| *liminf* | midicents |
| *long* | fix>0 |

### Output

chord

Generates aleatoric sequence of a linear distribution, between *lim.inf* and *lim.sup* with a length *long*.

# **triang-seq** *aleatoric sequence of a triangular distribution*

### Syntax

(alea::linea-seq  limsup  liminf  long)

### Inputs

| | |
|---|---|
| *limsup* | midicents |
| *liminf* | midicents |
| *long* | fix>0 |

### Output

chord

Aleatoric sequence of a triangular distribution, between *lim.inf* and *lim.sup* with a length *long*.

# Random Walks

This package contains four modules that generate lists of numeric sequences that can be associated with the following models:

*Simple random walk*

*Brownian motion*

*One-over-f (1/f) distributions*

*Markov chains*

Since the simple random walk and the brownian movement generate nonbounded sequences, as a constraint there is a model of elastic boundaries. When a value goes over the boundary it is automatically recalculated as it would be elastically reflected by this boundary.

## randwalk1 *random walk*

### Syntax

(alea::randwalk1 nc bsup binf pamax long)

### Inputs

| | |
|---|---|
| *nc* | fix/float |
| *bsup* | fix/float |
| *binf* | fix/float |
| *pamax* | fix |
| *long* | fix |

### Output

list

Generates a random walk, where *pamax* indicates the maximal step of the random walk, *nc* the initial value, *long* the length of the sequence and *binf* and *bsup* the lower boundary and the upper boundary.
Warning: boundaries are considered elastic.

# **randwalk2** *random walk*

### Syntax

(alea::randwalk2 nc bsup binf pamax long)

### Inputs

| | |
|---|---|
| *nc* | midic |
| *bsup* | midic |
| *binf* | midic |
| *pamax* | fix |
| *long* | fix |

### Output

list

Random walk, where *pamax* indicates the maximal step of the random walk in semi-tones, *nc* the initial value in midicents, *long* the length of the sequence and *binf* and *bsup* the lower boundary and the upper boundary. Warning: boundaries are considered as elastic.

# randwalkx *random walk*

### Syntax

(alea::randwalkx nc bsup binf pamax prox long)

### Inputs

| | |
|---|---|
| *nc* | midic |
| *bsup* | midic |
| *binf* | midic |
| *pamax* | fix |
| *prox* | fix>0 |
| *long* | fix |

### Output

list

Random walk, where *pamax* indicates the maximal step of the random walk according to the index *prox*.

If *prox* = 1, *pamax* is in semi-tone,

If *prox* = 2, *pamax* is in quarter-tone,

If *prox* = 4, *pamax* is in eight-tone,

i.e. *pamax* will change into 1/*prox* of one semi-tone, *nc* the initial value in midicents, *long* the length of the sequence and *binf* and *bsup* the lower boundary and the upper boundary. Warning: boundaries are considered elastic.

# **brownian1** *brownian motion*

### Syntax

(alea::brownian1 nc bsup binf sigma long)

### Inputs

| | |
|---|---|
| *nc* | fix/float |
| *bsup* | fix/float |
| *binf* | fix/float |
| *sigma* | fix/float |
| *long* | fix>0 |

### Output

list

Random walk, simulation of the brownian motion. *sigma* is associated with the *bandwidth* of the gaussian distribution, *nc* the initial value , *long* the length of the sequence and *binf* and *bsup* the lower boundary and the upper boundary. Warning: boundaries are considered elastic.

# **brownian2** *brownian motion*

### Syntax

(alea::brownian2 nc bsup binf sigma long)

### Inputs

| | |
|---|---|
| *nc* | midic |
| *bsup* | midic |
| *binf* | midic |
| *sigma* | fix/float |
| *long* | fix>0 |

### Output

list

Random walk, simulation of the brownian motion. *sigma* is associated with the *bandwidth* of the gaussian distribution, *nc* the initial value in midicents, *long* the length of the sequence and *binf* and *bsup* the lower boundary and the upper boundary in midicents;. Warning: boudaries are considered elastic.

# achorripsis

### Syntax

(alea::achorripsis nc bsup binf g long)

### Inputs

| | |
|---|---|
| *nc* | midic |
| *bsup* | midic |
| *binf* | midic |
| *g* | fix/float |
| *long* | fix>0 |

### Output

list

Random walk is based on the model used by Iannis Xenakis in *Achorripsis.*

(*g* is associated to the linear distribution), *nc* the initial value in midicents, *long* the length of the sequence and *binf* and *bsup* the lower boundary and the upper boundary in midicents. Warning: boundaries are considered as elastic.

# i1/f

## *Generates a value according to a 1/f distribution*

### Syntax

(alea::i1/f last n)

### Input

| | |
|---|---|
| *last* | fix |
| *n* | fix |

### Output

list

Generates a value according to the 1/*f* distribution with a first value *last*, and where *n* is a parameter which defines the output scale values, i.e., the values will fall between 0 and ($2^{n -1}$).

For example,

If n=2 the values fall between 0 and 3,
If n=4 the values fall between 0 and 15,
If n=7 the values fall between 0 and 127.

# seq1/f *Generates list according to a 1/f distribution*

### Syntax

(alea::seq1/f last n)

### Inputs

| | |
|---|---|
| *last* | fix |
| *n* | fix |
| *long* | fix>0 |

### Output

list

Generates a list of values according to the 1/f distribution where *prim* is the initial value, *long* the length of the sequence, and where *n* is a parameter which defines the output scale values. That is, the values will fall between 0 and ($2^{n -1}$). For example,

If n=2 the values fall between 0 and 3,

If n=4 the values fall between 0 and 15,

If n=7 the values fall between 0 and 127.

## markov1 *Generates an index from a markovian matrix*

### Syntax

(alea::markov1 prim l)

### Inputs

| | |
|---|---|
| *prim* | fix>0 |
| *l* | list |

### Output

numbers?

Generation of an index from a markovian matrix written in the list *l*, where *prim* is the first element of the resulting sequence. In the list *l* the transition is considered as made from the element of the row to the element of the column.

The list *l* is a double list of the form:

((1 2 3 4) (5 6 7 8 ) (9 10 11 12) (13 14 15 16 ))

where each sublist corresponds to a row. For example, the element *a34*, that is, the third row and fourth column, is element *12*. In the case where this matrix is a markov matrix, element *a34* corresponds to a transition from the element with index *3* to the element with index *4*.

# markov2 *Generates a list using a Markov matrix*

### Syntax

(alea::markov2 prim l long)

### Inputs

| | |
|---|---|
| *prim* | fix > 0 |
| *l* | list |
| *long* | fix > 0 |

### Output

numbers?

Generates a sequence of length *long* from a markovian matrix written in the list *l*, where *prim* is the first element of the resulting sequence.

In the list *l* the transition is considered as made from the element of the line to the element of the column.

The list *l* is a double list of the form:

((1 2 3 4) (5 6 7 8 ) (9 10 11 12) (13 14 15 16 ))

where each sublist corresponds to a row. For example, the element *a34*, that is, the third row and fourth column, is element *12*. In the case where this matrix is a markov matrix, element *a34* corresponds to a transition from the element with index *3* to the element with index *4*.

# Tools

These tools for filtering and scaling results have been designed to work in direct connection to the output of the distribution modules.

## filtre1 *Filters datum with elastic boundaries*

**Syntax**

(alea::filtre1 x bsup binf)

**Inputs**

| | |
|---|---|
| *x* | fix/float |
| *bsup* | fix/float |
| *binf* | fix/float |

**Output**

list

Filter datum *x* using the elastic boundaries *bsup* and *binf*.

# filtre2 *Filters datum with absorbant boundaries*

### Syntax

(alea::filtre2 x bsup binf)

### Inputs

| | |
|---|---|
| *x* | fix/float |
| *bsup* | fix/float |
| *binf* | fix/float |

### Output

list

Filter datum using absorbant boundaries.

# filtre3 *Filters a list using elastic boundaries*

### Syntax

(alea::filtre3 x bsup binf)

### Inputs

| | |
|---|---|
| *x* | numbers? |
| *bsup* | numbers? |
| *binf* | numbers? |

### Output

list

Filters a list of data using elastic boundaries.

# filtre4 *Filters a list using absorbant boundaries*

## Syntax

(alea::filtre4 x bsup binf)

## Inputs

| | |
|---|---|
| *x* | numbers? |
| *bsup* | numbers? |
| *binf* | numbers? |

## Output

list

Filters a list of data with absorbant boundaries.

# zoom1 *Transforms linearly the variation range of a variable between 0 and 1*

### Syntax

(alea::zoom1  x bsup2 binf2)

### Inputs

| | |
|---|---|
| x | fix/float |
| *bsup2* | fix/float |
| *binf2* | fix/float |

### Output

float

Linearly transforms the variation range of a variable *x* which is located between 0 and 1 in a range between *binf2* and *bsup2*.

## zoom2 *Transforms linearly the variation range of a variable*

### Syntax

(alea::zoom2  x  bsup1 binf1 bsup2 binf2)

### Inputs

| | |
|---|---|
| x | fix/float |
| *bsup1* | fix/float |
| *binf1* | fix/float |
| *bsup2* | fix/float |
| *binf2* | fix/float |

### Output

float

Linearly transforms the variation range of a variable *x* that is located between *binf1* and *bsup1* in a range between *binf2* and *bsup2*.

# zoom3 *Transforms linearly the variation range of a list*

### Syntax

(alea::zoom3  x bsup2 binf2)

### Inputs

| | |
|---|---|
| *x* | numbers? |
| *bsup2* | fix/float |
| *binf2* | fix/float |

### Output

float

Linearly transforms the variation range of a list of variables *x*, which are located between 0 and 1 in a range between *bsup2* and *binf2.*

# zoom4

*Transforms linearly the variation range of a list within a range*

**Syntax**

(alea::zoom4 x bsup1 binf1 bsup2 binf2)

**Inputs**

| | |
|---|---|
| x | numbers? |
| *bsup1* | fix/float |
| *binf1* | fix/float |
| *bsup2* | fix/float |
| *binf2* | fix/float |

**Output**

float

Linearly transforms the variation range of a list of variables *x*, which is located between *binf1* and *bsup1* in a range between *binf2* and *bsup2*.

# Bibliography

AMES, Charles (1991) - "A Catalog of Statistical Distributions: The Techniques for Transforming Random, Determinate and Chaotic Sequences." *Leonardo Music Journal* 1(1): 55-70.

BAFFIONI, C. and GUERRA, F. and LAURA, T. L. 1989. "The Theory of Stochastic Processes and Dynamical Systems as a Basis for Models of Musical Structures", in Atti del Convegno di Modena.

BOLOGNESI, Tomaso. 1979. "Composizione Automatica: Dalla musica 1/*f* alla Musica Autosimile", in *Atti del 3° Colloquio di Informatica Musicale*. Univ. di Padova, 2 a 3 aprile 1979.

DALMENICO, A. D. ; CHABERT, J. L. and CHEMLA, K. 1992. *Chaos et Déterminisme*. Ed. du Seuil, Paris.

DODGE, C. and JERSE, T. 1985. *Computer Music Synthesis*. New York: Schirmer Books.

LEGALL, Jean-François (1989) - "Introduction to the Brownian motion", in Gazette des Mathématiciens, avril 1989, n° 40, Société Mathématique de France.

LORRAIN, Denis. 1980. "A panoply of stochastic 'cannons'." *Computer Music Journal* 4(1). Reprinted with corrections in C. Roads, ed. 1989. *The Music Machine.* Cambridge, Massachusetts. pp. 351-379.

RUELLE, David. 1991. *Hasard et Chaos.* Paris: Ed. Odile Jacob.

SIDDALL, J. and SIDDALL. C. (1977) - "New developments in Stochastic Computer Music" In C. Roads, ed. *Proceedings of the 1977 International Computer Music Conference.* San Francisco: International Computer Music Association.

STEWART, Ian. 1992. *Dieu joue-t-il aux dés?* Paris: Flammarion.

VOSS, Richard F. and CLARKE, John. 1978. "1/f noise in Music: Music from 1/f Noise", in *J. Acoust. Soc. America*, vol 63, n° 1, January 1978.

XENAKIS, Iannis. 1981. *Musiques Formelles.* Paris: Stock Musique.

# Index