

Litter Power Package: List of Objects &c.

Discrete Distributions

Object	Distributions	Output Range	Options	Parameters (defaults, ranges)	Availability
lp.bernie	Bernoulli Binary Choice (n = 1)	0 .. n		n [1]: Positive integer, number of Bernoulli trials p [0.5]: Probability (i.e between 0 and 1) of positive outcome	Starter
lp.dicey	Dice	0 .. n * m		n [2]: Positive integer, number of dice m [6]: Faces on each die Optional list of integers representing "weights" of each face on the die. All faces for which no individual weighting has been specified are equiprobable	Pro
lp.ernie	Arbitrary distributions using the "finite urn" model			L [128]: Length of cycle Other messages: const int, refer symbol (refers to table with data)	Pro
lp.ginger	I Ching Produces both "present" and "future" oracles. Supports yarrow stick and coin toss methods.	1 .. 64	coin (default) yarrow	None	Starter
lp.lili	Uniform distribution over long integers. Well, sort of uniform. Based on the now largely obsolete Linear Congruence method for generating pseudo-random numbers. Allows you to play with the parameters. Find the shortest cycle!	0 .. mod.		mul [65539]: Seed multiplier between successive calls add [0]: Constant added to (seed * mul) at each successive call mod [4294967296]: Modulo base used. 0 is interpreted as 2^32. seed [1]: Initial seed for linear congruence pseudo-random number generator The default values give the same random numbers as Max.	Pro
lp.pfishie	Poisson	0 .. ∞		lambda [1]: Positive real, is both mean and standard deviation	Pro
lp.tata	Uniform distribution over long integers. Based on the Taus88 (Tausworthe) algorithm: flat distribution, cycle of just under 2^88, all bits random, robust seeding. All that, and faster than the traditional linear congruence method!	min .. max	(See parameters. Note that the range is defined inclusively)	min [-2,147,483,648]: integer less than max max [2,147,483,647]: integer greater than min If only one integer is given as an initialization argument and it is positive, the range is set from 0 to the specified value. If this integer is negative, the maximum is taken as 0.	Starter
lp.titi	Uniform distribution over long integers. Based on M. Matsumoto's TT800 algorithm: flat distribution, cycle of 2^800 - 1, all bits random, robust seeding.	min .. max	(See parameters. Note that the range is defined inclusively)	min [-2,147,483,648]: integer less than max max [2,147,483,647]: integer greater than min If only one integer is given as an initialization argument and it is positive, the range is set from 0 to the specified value. If this integer is negative, the maximum is taken as 0.	Pro

NB: Output from any of these distributions can be scaled/mapped to non-standard values with lp.scampi

Continuous Distributions

Object	Distributions	Output Range	Options	Parameters (defaults, ranges)	Availability
lp.abbie	arc sine (a = 0.5 and b = 0.5) beta	0 .. 1		a [0.5]: Positive real; increase tendency towards 0 b [0.5]: Positive real; increase tendency towards 1	Pro
lp.chichi	Chi Square	0 .. ∞		f [1]: Positive integer, degrees of freedom	Pro
lp.coshy	Cauchy Positive Cauchy Negative Cauchy	-∞ .. ∞ 0 .. ∞ -∞ .. 0	sym (default) pos neg	tau [1]: Positive real, effectively a scaling factor	Pro
lp.expo	Exponential Bilateral ("[First Law of] Laplace") Negative Exponential	0 .. ∞ -∞ .. ∞ -∞ .. 0	pos (default) sym neg	lambda [1]: Positive real, influences mean and standard deviation	Pro
lp.fishie	Fisher	0 .. ∞		f1 and f2 [both 1]: Positive integers, two independent degrees of freedom parameters	Pro
lp.gammer	Gamma/Erlang distribution	0 .. ∞		nu [1]: Positive real. If nu is an integral value, the Erlang distribution is produced. lambda [1]: Positive real	Pro
lp.grrr	"Gray" noise	0 .. 1		None	Pro
lp.hyppie	Hyperbolic Cosine	-∞ .. ∞		None	Pro
lp.linnie	Linear (decreasing) Triangular Linear (increasing)	0 .. 1	neg (default) sym pos	None Note: the option names refer to the slope of the distribution. Does that help?	Starter
lp.loggie	Logistic	-∞ .. ∞		alpha [1]: Positive real beta [0]: Non-negative real	Pro
lp.lonnie	Log-normal	(0) .. ∞		mu*[1]: Positive real sigma* [1]: Positive real	Pro
lp.norm	Normalized Gauss distribution (with mean at 0 and standard deviation of 1)	-∞ .. ∞		mu[0]: any real value sigma [1]: Positive real (0 is allowed but counter-productive; negative values give the same results as the complementary positive value)	Starter
lp.pfff	Brownian motion ("brown" noise)	0 .. 1			Starter
lp.schhh	Uniform ("white" noise)	0 .. 1		nn [0]: Integer from 0 to 31; a noisiness factor (number of least significant bits masked out before calculating result)	Starter
lp.sss	1/f ("pink" noise: Voss/Gardner algorithm)	0 .. 1		nn [0]: Integer from 0 to 29 a noisiness factor (number of least significant bits masked out before calculating result).	Starter
lp.stu	Student's "T" Distribution	-∞ .. ∞		f [1]: Positive integer, degrees of freedom	Pro

lp.y	Weibull/Rayleigh distributions	0 .. ∞		s [1]: Positive real t [1]: Positive real	<i>Pro</i>
lp.zzz	1/f (McCartney's improved algorithm for pink noise)	0 .. 1		nn [0]: Integer from 0 to 29 a noisiness factor (number of least significant bits masked out before calculating result).	<i>Pro</i>

NB: Output from any of these distributions can be scaled/mapped to other values (including integers) with lpp.scampi

Signal Generators

Object	Distributions	Output Range	Options	Parameters (defaults, ranges)	
lp.frrr~	Low frequency noise	signal (-1 .. 1)		baseFreq [1000]: Approx. frequency of center freq. (always rounded to integral division of sample rate) Interp [0]: Order of interpolation between generated random values. Range from 0 to 2.	<i>Pro</i>
lp.grrr~	Gray noise	signal (-1 .. 1)			<i>Pro</i>
lp.lll~	Linear Congruence Noise	signal (-1 .. 1)		mul add: mod[0]: 0 is interpreted as 2^32. seed [1]: Initial seed for linear congruence pseudo-random number generator	<i>Pro</i>
lp.pfff~	Brown noise	signal (-1 .. 1)		nn [0]: Granularity of noise (bit-masking)	<i>Starter</i>
lp.phhh~	Black noise	signal (-1 .. 1)		nn [0]: Granularity of noise (bit-masking)	<i>Pro</i>
lp.ppp~	Dust/popcorn noise	signal (-1 .. 1)	pos (default) sym neg	density [100]: average number of "pops" per second. popWidth [1]: number of samples before and after pop peak.	<i>Pro</i>
lp.shhh~	White noise	signal (-1 .. 1)		nn [0]: Granularity of noise (bit-masking)	<i>Starter</i>
lp.sss~	Pink noise (Original Voss algorithm)	signal (-1 .. 1)		nn [0]: Granularity of noise (bit-masking)	<i>Starter</i>
lp.zzz~	Pink noise (McCartney's algorithm)	signal (-1 .. 1)		nn [0]: Granularity of noise (bit-masking)	<i>Pro</i>

Mutation Processors

Object	Distributions	Output Range	Options	Parameters (defaults, ranges)	
lp.flim	Floating point interval mutation	float	usim, uuim, isim, iuim, lcm, wcm		<i>Pro</i>
lp.frim~	Frequency-domain interval mutation	signal	usim, uuim, isim, iuim, lcm, wcm		<i>Pro</i>
lp.intim	Integer interval mutation	integer	usim, uuim, isim, iuim, lcm, wcm		<i>Pro</i>
lp.tim~	Time-domain interval mutation	signal	usim, uuim, isim, iuim, lcm, wcm		<i>Starter</i>

Utilities

Object	Description	Options	Parameters (defaults, ranges)	
lp.c2p~	Cartersian to polar coordinate conversion (compatible with MSP1 and MSP2)		None	<i>Starter</i>
lp.grl~	Phase unwrapping		FFT Size	<i>Pro</i>
lp.i	Posts fortunes from I Ching to the Max window. Can be used with the ginger object.		None	<i>Pro</i>
lp.kg	Maps I Ching results (from the ginger object) to some number (2 to 63) of different items. Bang generates a new distribution		n [2]: Integer between 2 and 63, number of different choices to make.	<i>Pro</i>
lp.p2c~	Polar to Cartesian coordinate conversion (compatible with MSP1 and MSP2)		None	<i>Starter</i>
lp.scampf	Scale, map, and limit floating point values	clip reflect wrap split stet	scale [128]: Any value; input 1 maps to offset + scale. (default value chosen for convenient mapping of MIDI data) offset [0]: Any value that 0 input maps to. limit [stet]: any of the limiting options clip, reflect, wrap, split, or stet. min [0]: Minimum output value (ignored if limit option is set to stet). max [1]: Maximum output value (ignored if limit option is set to stet).	<i>Starter</i>
lp.scampi	Scale, map, and limit values	clip reflect wrap split stet trunc round floor ceil	scale [1/128]: Any value; input 1 maps to offset + scale. (default value chosen for convenient mapping of MIDI data) offset [0]: Any value that 0 input maps to. limit [stet]: any of the limiting options clip, reflect, wrap, split, or stet. min [0]: Minimum output value (ignored if limit option is set to stet). max [1]: Maximum output value (ignored if limit option is set to stet). integer-conversion [trunc]: Integer processing truncates by default, but scampi understands floor, ceil, round, and trunc messages to specify direction.	<i>Starter</i>
lp.scamp~	Scale, map, and limit signals	clip reflect wrap stet	scale [-6dB]: Any value; input 1 maps to offset + scale. offset [0]: Any value that 0 input maps to. limit [stet]: any of the limiting options clip, reflect, wrap or stet. (Note that lp.scampi~ does not support split) min [-1]: Minimum output value (ignored if limit option is set to stet). max [1]: Maximum output value (ignored if limit option is set to stet).	<i>Pro</i>

