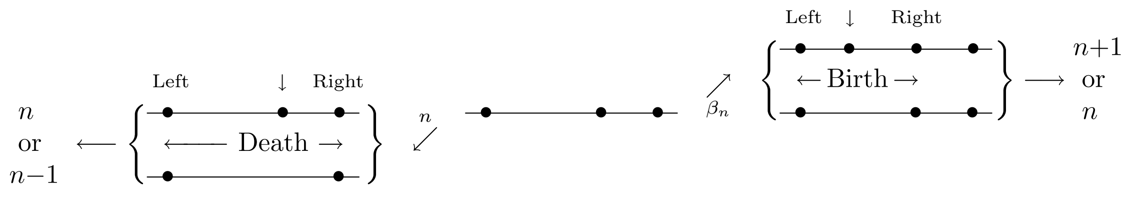
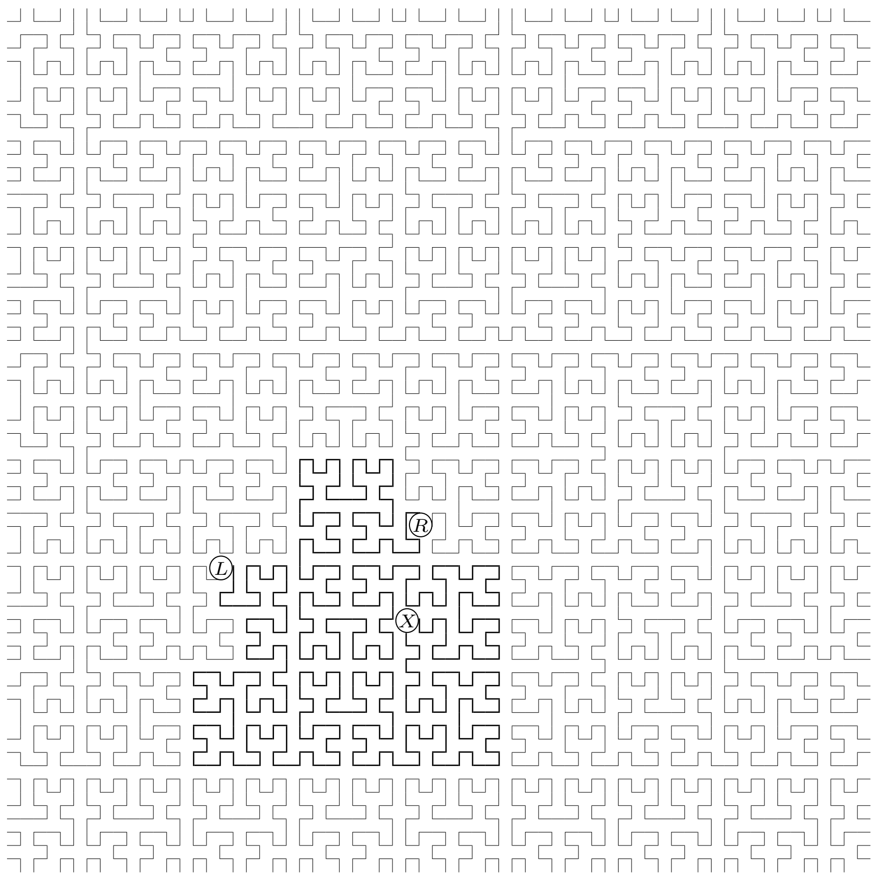
BayeSys and MassInf





```
..o..oo....oo.oo.....oo.ooo....ooooooo..ooooo...ooo.o...o...o....o...
       ←—|—→        ←—|—→     ←|→    ←|→     ←———|———→
```

John Skilling Maximum Entropy Data Consultants Ltd (est.1981) February 2003

2. Overview of Inference

For us, inference is the art of recovering an object θ from data D. Usually, the object will be something quite complicated, perhaps a spectrum or an image, having at least several and maybe thousands or millions of degrees of freedom. The data, acquired through some experiment R so that

$$D \approx R(\theta)$$

will nearly always be incomplete (fewer components than θ has) or noisy (inexact), or somehow inadequate to fix θ unambiguously. Methods of inference fall into three increasingly sophisticated classes, "inversion", "regularisation" and "probabilistic".

I can illustrate these with the population of the four countries (England, Scotland, Wales and Ireland) that comprise the British Isles. I have received data (mythical, of course) that of the entire population of 10000, 8000 live in the east (England and Scotland), and 7500 live in the south (England and Wales). ¿From these three numbers, I have to estimate the populations $\theta = (\theta_1, \theta_2, \theta_3, \theta_4)$ of the four countries.

Ireland $= \theta_4$	Scotland $= \theta_2$	2500
Wales $= \theta_3$	England $= \theta_1$	7500
2000	8000	10000

In algebraic terms,

$$D = \begin{bmatrix} 2500 \\ 7500 \\ 2000 \\ 8000 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} \theta_1 \\ \theta_2 \\ \theta_3 \\ \theta_4 \end{bmatrix} = R\theta.$$

2.1. INVERSION

Inversion in its pure form inverts R to obtain the estimate $\hat{\theta} = R^{-1}D$, which might work except that R is here (and in general) singular so that there is no inverse. A common fixup is to use a pseudo-inverse instead, so that $\hat{\theta} = XD$ where X approximates the inverse of R. But the data which we would have obtained if this were true are RXD which, insofar as X is **not** the inverse of R, differ from the true data, showing $\hat{\theta}$ cannot be correct. So much for inversion. This method should only be used when the results are essentially unambiguous, or don't matter much.

2.2. REGULARISATION

In my illustrative problem, simple logic shows that the data leave only one degree of freedom. Let this be the Irish population x. Then

$$\theta_4 = x, \quad \theta_3 = 2000 - x, \quad \theta_2 = 2500 - x, \quad \theta_1 = 5500 + x,$$

and this satisfies the data for any x. Regularisation consists of finding the "best" result from among all those that agree with the data, as defined by maximising some regularisation function $\Phi(\theta)$ representing quality. The commonest such method is least squares, which here fixes x by minimising

$$\theta_1^2 + \theta_2^2 + \theta_3^2 + \theta_4^2 = -\Phi(\theta)$$

subject to agreement with the data. Unfortunately the selected value of x is -250, indicating that the population of Ireland is negative. Really? Least squares is not always best!

For the sort of positive distribution being considered, maximum entropy (maximise $\Phi = -\sum \theta_i \log \theta_i$) is useful. Negative populations are prohibited by the logarithm, and the result

Ireland $= 500$	Scotland $= 2000$
Wales $= 1500$	England $= 6000$

has the 1:3 north:south ratio nicely independent of longitude, and the 1:4 east:west ratio nicely independent of latitude. In fact, entropy is the only function yielding this general symmetry, which may well account for the high quality of many maximum-entropy reconstructions. But questions remain. How should we account for noise in the data? What about nuisance parameters? How uncertain is the "best" result?

2.3. PROBABILITIES

In 1946, R.T. Cox [ref] proved that the only way of doing inference consistently is through probability calculus and seeking the posterior probability distribution $\Pr(\theta \mid D)$ of result θ, given data D. We start by assuming a prior distribution representing what we think θ might be, and modulate this with how likely the data measurements would then be. This gives the joint distribution of θ and D, which can be alternatively expanded as the posterior we want, scaled by a coefficient called the "evidence".

$$\begin{array}{ccccc} & & \text{assumptions \& measurements} & \Rightarrow & \text{inference} \\ \Pr(\theta, D \mid I) & = & \Pr(\theta \mid I) \times \Pr(D \mid \theta, I) & = & \Pr(\theta \mid D, I) \times \Pr(D \mid I) \\ \text{Joint} & & \text{Prior} \times \text{Likelihood} & & \text{Posterior} \times \text{Evidence} \end{array}$$

That's the famous, and stunningly elementary, **Bayes' Theorem**. In writing it, I have explicitly included the dependence on whatever ancillary information I is to hand (though in practice one often omits the symbol when the context is understood).

Observe that the evidence amounts to the likelihood for this information. We would use exactly the same Bayes' Theorem to compare different ancillary assumptions I (under more general background information J) as we do here to compare different parameter values θ: it is only the identifications of the symbols that would change. If we don't have any alternatives for I, the evidence is useless to us, though in principle we should still calculate it and publish it in case somebody else can think of an alternative that might explain the data better. Or worse, if the alternative evidence was smaller. Always calculate the evidence (its units are inverse data) as an integral part of Bayesian inference. Few people do this, which is a pity.

2.4. PRIOR PROBABILITIES

Before we even start to use the data, we need to assign a prior probability distribution $\pi(\theta) = \Pr(\theta \mid I)$ to everything relevant that we don't know. I sometimes remark that I don't know what I am talking about — until I have defined a prior. We have complete freedom to set π, subject only to it being non-negative and summing to unity (so that it can actually **be** a probability distribution). In practice, we will assign π by using such desiderata as symmetry, simplicity, and reasonable behaviour, according to our experience and judgment. We can also use the posterior distribution from previous observations as our current prior. (That's what "doing inference consistently" means. It doesn't matter whether we use dataset D_1 first then D_2, or the other way round, or use them both together. Each yields the same posterior.) Yes, the assignment of prior is subjective. That's the way it is! But, after acquiring data, we can compare evidence values, so the subjective assumptions can be objectively compared.

In my illustrative population example, we need to assign a prior over the four non-negative populations. In fact, we may as well split the British Isles into arbitrary fractions f_i instead of restricting to four quarters. One choice, which can be used at any resolution and is quite often suggested (though seldom by me), is a gamma distribution

$$\pi(\theta) \propto \prod \theta_i^{-1+cf_i} e^{-\alpha\theta_i}$$

where α and c are constants. Priors nearly always involve constants like these (grandiosely called "hyper-parameters"). Interpreting them and setting them appropriately is part of the art.

Given the success of maximum entropy in regularisation, it is tempting to try

$$\pi(\theta) \propto \exp(-\theta_1 \log\theta_1 - \theta_2 \log\theta_2 - \theta_3 \log\theta_3 - \theta_4 \log\theta_4)$$

or some close variant. However, integration shows that the implied prior on the total population $\Theta = \theta_1 + \theta_2 + \theta_3 + \theta_4$ is nothing like $\exp(-\Theta \log \Theta)$, whereas the gamma distribution would have remained intact. Worse, the entropy prior cannot be subdivided. There is no distribution $p(\cdot)$ which could be applied independently to northern and southern England and then integrated to give a prior like $\exp(-\theta \log \theta)$ for the population of England as a whole. Entropy is a good regulariser but, like most functions, it does not translate into a good prior.

2.4.1. *Atomic priors*

One useful way of breaking down the complexity of a problem is to construct the object θ as some number of *a-priori*-equivalent "atoms", which are scattered randomly over the domain of interest. These atoms are given whatever extra attributes are needed to model the object. In the population example, an atom might represent a person, in which case those particular data would require 10000 atoms. Or an atom might represent a census unit of 500 people, in which case only 20 would be needed. Or an atom might represent a tribe whose size was drawn from some distribution such as an exponential: this would often be better because of the flexibility involved, and might require even fewer atoms. Whatever an atom represents, and however many there are, letting them fall randomly over the domain ensures that we can compute on any scale.

Atomic priors give a structure-based description. The computer only needs to deal with the amount of structure that is actually required, because the number of atoms is usually allowed to vary. Spatial resolution, in the form of accuracy of location of the atoms, is "free", because each atom is automatically held to the arithmetical precision of the hardware. Algebraically-continuous priors like the gamma distribution, by contrast, give a cell-based description. To use them, we have to divide the spatial domain into as many cells as we are ever likely to need, and be prepared to compute them all. Resolution is directly limited by the computer memory and processor time. This comparison between atomic and continuous priors is rather analogous to the comparison between a Monte Carlo representation by samples and a full covering of the entire space. In each case, the former is practical, while the latter is likely not. And there is no loss of generality. If required, we could let the whole θ be a single atom having attributes for every cell, which would reproduce a cell-based prior.

To define an atomic prior, we assign distributions for the number n of atoms, and for their attributes. Typical priors for n are

$$\pi(n) = \text{constant}$$

between a minimum and maximum number (where equality makes n fixed), or Poisson

$$\pi(n) = e^{-\alpha}\alpha^n/n!$$

(binomial if a maximum is imposed), or geometric

$$\pi(n) \propto c^n \quad \text{with } c < 1$$

(which is wider than the Poisson). As a technical note, only the Poisson assignment is "infinitely divisible", meaning that it can be accumulated from arbitrarily small but fully independent subdivisions of the domain — remember that Poisson distributions combine into another Poisson distribution with summed mean. If computed at small scale, the other forms need small correlations to make the total number correctly distributed. I don't think that matters at all. Indeed, I usually prefer the less-committal geometric assignment.

2.4.2. *Coordinates*

As for the attribute coordinates, their priors will depend on what the attributes are. Importantly, there are many applications where an atom has very few attributes, such as image reconstruction where an atom has only position (x, y), intensity, and possibly shape. It is then much easier to find an acceptable and useful new position for an atom than it would be for the object as a whole, simply because of the huge reduction in dimensionality. (*Divide and conquer*, as the slogan has it.)

An important case is where an attribute measures an additive quantity z such as population number, or power of signal, or brightness of radiation. For such intensities, an exponential distribution is often convenient,

$$\pi(z) = c\,e^{-cz}, \quad c = \text{constant}.$$

Equivalently, the cumulant distribution $\theta = \int_0^z \pi(\zeta)d\zeta = 1 - e^{-cz}$ is (by construction) uniformly distributed with $\pi(\theta) = 1$ in $[0, 1]$. This form can be imposed quite generally. Even if there are several (d) attributes of an atom, it is always possible to squash the original prior into uniformity $\pi(\theta) = 1$ over the unit hypercube

$[0,1]^d$. I recommend this discipline: there is no loss of generality and numerical exploration is likely to be easier.

2.5. SAMPLING

Only when θ has very few degrees of freedom can we hope to explore "all" θ to find the posterior "everywhere". Instead, the modern approach is to characterise the posterior in a Monte Carlo sense, by taking a dozen or more random-sample objects $\tilde{\theta}$ from it.

It is fortunate but true that these dozen samples will very likely answer any particular question about θ with adequate precision. The point is that each sample object yields an independent value $\tilde{Q} = Q(\tilde{\theta})$ for the scalar quantity Q being sought. Only occasionally will these be badly biassed overall. For example, there is only about a 1 in 2000 chance that their average $\langle\tilde{Q}\rangle$ will be more than one standard deviation from the true mean of Q (for Gaussian statistics), and this chance drops sharply as more samples are taken.

Note that any single θ selected from the posterior should be treated with care. Suppose for instance that the posterior requires θ to lie on a circle. Then the mean (an obvious candidate for presentation) will lie inside the circle, which is supposed to be prohibited! A median is predicated upon being able to order the values of θ, which really only makes sense if θ is restricted to one dimension. The mode, obtained by maximising the posterior (in a method sometimes glorified with the acronym MAPP for maximum a-posteriori probability), moves if θ is re-parameterised, because squeezing θ somewhere increases the posterior there to compensate. Hence the mode lacks an invariance we often want. In fact the dozen or more sample objects $\tilde{\theta}$ seem to be the only faithful representation that is generally accessible.

3. Markov chain Monte Carlo (MCMC)

Markov chain Monte Carlo algorithms are the preferred method for practical inference. The only generally faithful way of representing the posterior distribution is through sampling, which implies a Monte Carlo method. And the posterior can only be reached in practice through a Markov chain of intermediate states, each learning from the previous one(s). Hence MCMC.

Let our object θ have available states $i = 1, 2, \ldots$. A MCMC algorithm for exploring θ is identified by the transitions $\mathbf{T}$ that it can make, with their probabilities.

$$i \to j \text{ with probability } T_{ji} = \Pr(j \mid i)$$

Technically, a Markov transition could also involve a memory of earlier states. However, transitions may (and often do) already involve a sophisticated history of intermediate trial or actual states, so the restriction is more apparent than real.

Suppose our current knowledge of θ is probabilistic: $\Pr(\theta_i) = p_i$. Repeatedly applying the algorithm will yield $\mathbf{p} \to \mathbf{Tp} \to \mathbf{T}^2\mathbf{p} \to \mathbf{T}^3\mathbf{p} \to \cdots$. Eventually, $\mathbf{p}$ will converge to the principal eigenvector of $\mathbf{T}$ with greatest eigenvalue (which, because the components of $\mathbf{p}$ always sum to the same unit total, must be 1). Provided the algorithm is aperiodic (so that it doesn't just bounce) and "irreducible" (so that every state is eventually accessible from any other), this principal eigenvector is unique. Elements of randomness soon ensure that an algorithm is aperiodic, and if it happens that chains of successive $\mathbf{p}$ reduce into disjoint un-mixed domains, we will include extra transition routes to join all such domains. Hence it is usually straightforward to satisfy these conditions. We will start by designing algorithms which target the prior $\Pr(\theta) = \pi(\theta)$, for which we need a transition matrix whose principal eigenvector is $\boldsymbol{\pi}$. Eigenvectors are somewhat remote from the components of a matrix, being expensive to compute. However, any matrix with the property of "**detailed balance**"

$$T_{ji}/T_{ij} = \pi_j/\pi_i \quad \text{for all } i,\, j$$

will suffice. Thinking physically, we see that if we start correctly with π_i objects in state i and π_j in state j, then on applying $\mathbf{T}$ the same number ($T_{ji}\pi_i$) will pass from i to j as pass from j to i ($T_{ij}\pi_j$), leaving the correct assignment intact.

$$\begin{tikzcd}
\boxed{i} \arrow[r, shift left, "T_{ji}\pi_i"] & \boxed{j} \arrow[l, shift left, "T_{ij}\pi_j"]
\end{tikzcd}$$

The corresponding algebraic proof is $(\mathbf{T}\boldsymbol{\pi})_j = \sum_i T_{ji}\pi_i = \sum_i T_{ij}\pi_j = (\sum_i T_{ij})\pi_j = \pi_j$. Although not essential, detailed balance is the key to constructing useful transition matrices. We start with algorithms that explore the prior faithfully, leaving the extra complication of likelihood factors until later.

3.1. THE NUMBER OF ATOMS

Typical priors $\Pr(n)$ for the number of atoms are

$$\pi(n) = \begin{cases} 1/N, & \text{for } 0 \le n < N \quad \text{(uniform);} \\ e^{-\alpha}\alpha^n/n!, & \text{for } n \ge 0 \quad \text{(Poisson);} \\ c^n, & \text{for } n \ge 0 \quad \text{(geometric).} \end{cases}$$

and we seek an algorithm that faithfully targets any such prior. The natural unit of change is just one atom at a time, so the only non-zero transitions T_{ji} will be $j = i + 1$ ("birth" of an atom), $j = i - 1$ ("death" of an atom), and $j = i$ (no change). Detailed balance fixes the ratios between birth and death but leaves their overall magnitudes free.

I choose to let each atom decay with unit mean lifetime, so that the death rate is set as

$$T_{n-1,n} = n\, dt$$

in infinitesimal interval dt of artificial time. The rationale for this is that most of the atoms will have been changed after unit time. Regular $\mathcal{O}(1)$ timesteps thus give a natural sampling period τ for our multi-atom object. Sampling more frequently would make successive objects too similar, whereas sampling less often might equilibrate wastefully between samples. Detailed balance implies a corresponding birth rate

$$T_{n+1,n} = \beta_n \, dt\,, \qquad \beta_n = (n+1)\, \pi_{n+1}/\pi_n.$$

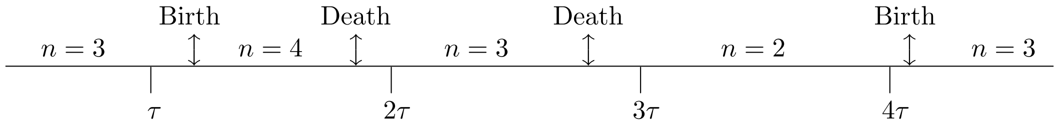
For the three typical priors, these birth rates are

$$\beta_n = \begin{cases} n+1, & \text{for } n < N-1, \text{ else } 0 \quad \text{(uniform);} \\ \alpha, & \text{(Poisson);} \\ (n+1)\, c, & \text{(geometric).} \end{cases}$$

Starting with (say) n atoms, the time to the next event is exponentially distributed

$$\Pr(\Delta t) = (\beta_n + n)\, e^{-(\beta_n + n)\, \Delta t}$$

and when that event occurs, it is either birth or death in ratio $\beta_n : n$. Thus, in the following example, samples are taken at uniform times $\tau, 2\tau, 3\tau, 4\tau, \ldots$ when n happens to be $3, 3, 2, 2, \ldots$.



3.2. COORDINATES

As recommended above, we require the prior to be uniform $\pi(\theta) = 1$ over the unit hypercube $[0,1]^d$. In one dimension, this domain is just the unit interval $0 < \theta < 1$. Within the computer, of course, the coordinate will come from a finite set determined by the finite precision of the hardware. This observation suggests a "modern digital" style of treatment. My convention, taking the computer word length to be B bits (usually 32) is to let the available values be odd multiples of $2^{-(B+1)}$, labelled by `unsigned` (*i.e.* non-negative) integers from 0 to $2^B - 1$:

$$\theta_k = 2^{-B}(k + \tfrac{1}{2}).$$

This makes all states *a-priori*-equivalent, and keeps $1-\theta$ always paired with θ. Helpfully, the states are symmetrically away from the boundaries $\theta = 0$ and $\theta = 1$, and centre $\theta = \frac{1}{2}$, which might be special. The rules of integer arithmetic (modulo 2^B) ensure that θ is wraparound continuous, which is never harmful and sometimes appropriate.

In one dimension, the obvious "analogue" transition scheme is to select $\delta\theta$ at some appropriate scale (to be assigned somehow), and then use it to either increment or decrement θ:

$$\theta \;\leftarrow\; \theta \pm \delta\theta \pmod 1.$$

In the underlying integer representation, with modulo 2^B arithmetic understood,

$$k \;\leftarrow\; k \pm \delta k.$$

With increment and decrement being equi-probable, the algorithm is clearly in detailed balance over a uniform prior, no matter how $\delta\theta$ was set. An alternative "digital" transition scheme is to decide on some number b up to B, and then randomise the low order b bits of k:

$$k \;\leftarrow\; k \vee \texttt{Uniform}[\,0, 2^b)$$

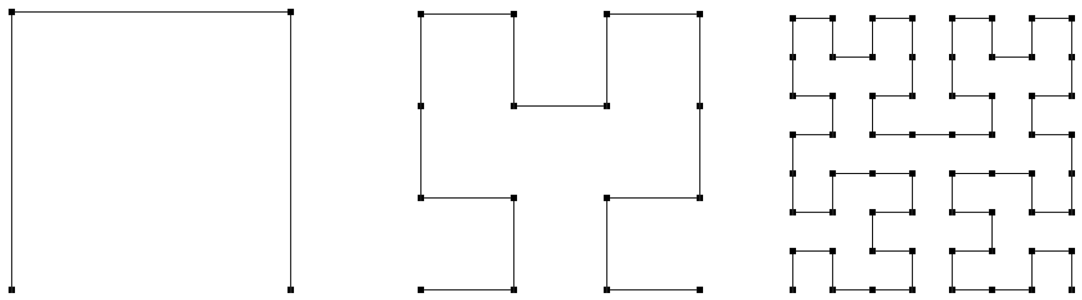
where "$\vee$" represents binary exclusive-or, here with a random integer less than 2^b. Extra randomisation at the same scale is achieved by taking the coordinate relative to some non-zero origin o.

$$k \;\leftarrow\; \big(\,(k-o) \vee \texttt{Uniform}[\,0, 2^b)\,\big) + o$$

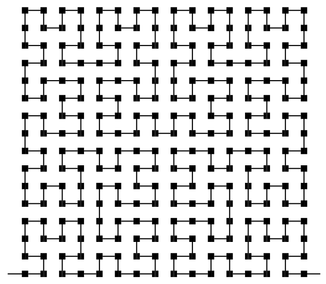
Advantages of this digital algorithm will become clear later.

When θ has d dimensions (more than one), we can fill the unit hypercube with a space-filling Peano curve, thus reducing the topology to a one-dimensional coordinate along the curve. The hypercube contains $(2^B)^d$ digital points, so each point along the curve is labelled by an integer with Bd bits (d words). The diagrams below show Peano curves in two dimensions.

A Peano curve can be constructed recursively from its generator. At the top level the generator is a path around the 2^d corners of a hypercube, using segments directed parallel to the axes. In computer parlance, this path is a "Gray code" for d-bit integers. At the next (second) level, smaller copies of each generator are placed at each first-level point, oriented to keep the ends of successive generators adjacent. There are now 2^{2d} points along the path. At the third level, yet smaller copies of each generator are placed at each second-level point, again oriented to keep the ends of successive generators adjacent. By now, the path has 2^{3d} points.



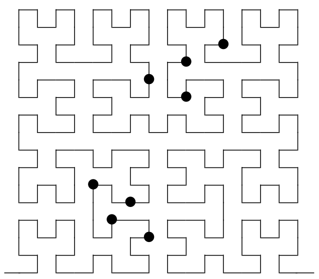
And so on, at finer and finer scales until all Bd bits have been used. Each stage adds resolution without changing the existing larger-scale pattern: with 4-bit axes $0, 1, \ldots, 15$ there are $16^d = 16^2 = 256$ points.



A Peano curve preserves locality, meaning that contiguity along the line implies contiguity in space (though not conversely). In fact, the Peano curve has the greatest degree of locality possible for a space-filling curve. It follows from locality that a Peano curve also preserves continuity, meaning that a function that is continuous in space must also be continuous along the line (though not conversely). Differentiability, though, is not preserved. Even so, there are useful methods for one-dimensional exploration which can immediately be applied in d dimensions, simply by invoking extended-precision integer arithmetic.

A length of line that occupies a fraction f of the total necessarily fills that same fraction f of the hypercube volume. The winding pattern spirals out very roughly isotropically, so that each coordinate ranges over something like $f^{1/d}$ as the length is traversed. Winding a one-dimensional line into a d-dimensional volume, though, necessarily places some points that are close in space far apart on the line. With the very precise and binary Peano pattern, there is only one crossing of the central line $\theta_1 = \frac{1}{2}$ (vertical in the diagrams), so that points on opposite sides of that line are mostly far apart on the Peano curve. Similarly, there is only one crossing of the (wraparound continuous) abscissa $\theta_1 = 0$, and none at all of the base $\theta_2 = 0$. These barriers to free spatial movement are made harmless by moving them around after each observation interval τ, re-randomising the origin of the Peano curve within the hypercube and re-permuting its orientation.

One-dimensional representation along a line implies an ordering of locations, so that an atom's neighbours can be quickly identified by keeping a linked list. Neighbouring atoms along the line may not be quite the closest in space, but they still turn out to be useful because they identify pairs of atoms that may be similarly related to the data, and whose behaviour may thereby be correlated.



In order to keep the ordering of atoms unambiguous, I accept a restriction that not more than one atom may occupy a single point. There is a huge number of points, so this is only a technicality.

3.3. ROLE OF LIKELIHOOD

We have, so far, discussed the form of the prior, and settled on a flat prior over the unit hypercube for the attributes of each of a variable number of atoms. For exploration of the prior by MCMC methods, we have introduced birth and death rates for changing the number of atoms, and movement along a Peano curve to change their positions. It is now time to introduce the data.

To correct a MCMC algorithm and make it converge to the required posterior instead of merely to the prior, its transition probabilities need to be adjusted. Suppose that transitions are performed, not definitively, but probabilistically according to acceptance probabilities A. This will reduce an effective transition rate from T_{ji} to $A_{ji}T_{ji}$. Detailed balance should conform to the posterior $L(\theta)\pi(\theta)$, so we require

$$\frac{A_{ji}T_{ji}}{A_{ij}T_{ij}} = \frac{L(\theta_j)\pi(\theta_j)}{L(\theta_i)\pi(\theta_i)}.$$

Because the basic transition scheme T will already conform to the prior $\big(T_{ji}/T_{ij} = \pi(\theta_j)/\pi(\theta_i)\big)$, the acceptance probabilities must simply be in ratio of the likelihoods

$$\frac{A_{ji}}{A_{ij}} = \frac{L(\theta_j)}{L(\theta_i)}.$$

To avoid wasting resources, we want to accept as often as possible, so we set whichever acceptance would be larger to its greatest permitted value of 1. Hence

$$A_{ji} = \min\left(1,\ \frac{L(\theta_j)}{L(\theta_i)}\right)$$

which is equivalent to the rule:

$$\text{“Accept transition } i \to j \text{ if and only if } L(j) \geq \texttt{Uniform}\big(0, L(i)\big)\text{”.}$$

Although delightfully simple, this method (due to Metropolis [ref] as generalised by Hastings [ref]) can be inefficient. The worst difficulty lies in how to choose the magnitude of change $\delta\theta$ in the transitions. If $\delta\theta$ is set too small, diffusion of position will be un-necessarily (and quadratically) slow. If $\delta\theta$ is set too large, nearly every proposal will be rejected, and the procedure effectively stops — without having converged. We need a way of ensuring that $\delta\theta$ has the correct scale.

Incidentally, it is now apparent why attention is universally focussed on pair-wise transitions rather than on longer cycles

$$\begin{tikzcd} \boxed{i} \arrow[r] & \boxed{j} \arrow[d] \\ \boxed{l} \arrow[u] & \boxed{k} \arrow[l] \end{tikzcd}$$

that might seem to offer systematic exploration instead of diffusion. The flux of samples around such a cycle is limited by the relative probability of its least likely state. With larger cycles, this will be an ever-smaller fraction of the highest probability where most of the samples will reside, so the flux decreases and the cycle offers no gain. The smallest non-trivial cycle (length 2) is best.

3.4. SLICE SAMPLING

Let the initial state, with likelihood L_0, be represented by a B-bit integer k from domain $\mathcal{D}_0$. Set an acceptance level

$$a \in \texttt{Uniform}(0, L_0)$$

and randomise all B bits of k to reach

$$j_0 = k \vee \texttt{Uniform}[\,0, 2^B)$$

in domain $\mathcal{D}_0$. According to Metropolis-Hastings, we are entitled to accept any new trial state j whose likelihood exceeds a, provided it was generated symmetrically with the reverse transition $j \to k$ being just as probable as the forwards $k \to j$. So, if the likelihood for j_0 is greater than a, accept it.

If not, halve the domain size to $\mathcal{D}_1 \subset \mathcal{D}_0$ by randomising only the lowest $B-1$ bits,

$$j_1 = k \vee \texttt{Uniform}[\,0, 2^{B-1}),$$

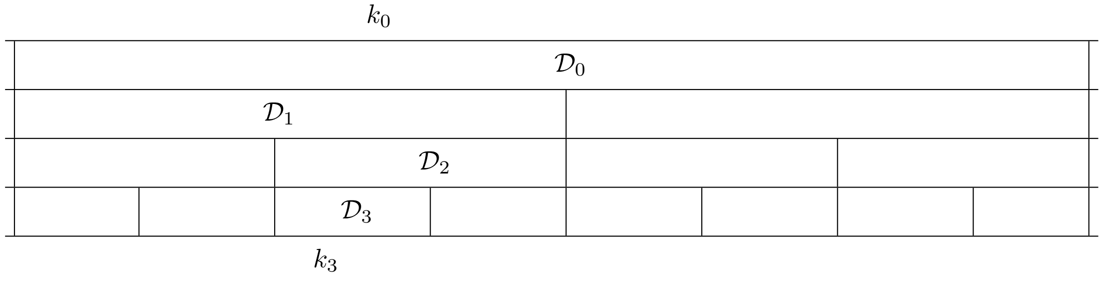
keeping the top bit intact. The transition is symmetric because both k and j_1 lie in the same domain $\mathcal{D}_1$, so $j_1 \to k$ is just as likely as $k \to j_1$ (with probability $1/2^{B-1}$ as it happens). So, if the likelihood for j_1 is greater than a, accept it.

If not, halve the domain size again to $\mathcal{D}_2$ by randomising only the lowest $B-2$ bits,

$$j_2 = k \vee \texttt{Uniform}[\,0, 2^{B-2}).$$

This transition too is symmetric, because k and j_2 both lie in $\mathcal{D}_2$ so $j_2 \to k$ is just as likely as $k \to j_2$, and each direction had the same chance of being aborted at the earlier stages. (This would no longer be true if j_2 had been obtained by addition or subtraction instead of bit-randomisation, because k and j_2 would have gone to each other via different ranges and would hence have had different chances of aborting.) So, if the likelihood for j_2 is greater than a, accept it.

If not, keep going by randomising fewer and fewer low-order bits, until an acceptable likelihood is reached. At worst, this procedure terminates (after B trials) with the original state, necessarily acceptable because $L_0 \geq a$. More likely, an acceptable state will be found somewhere around the scale of the controlling likelihood function, at which $\Delta \log L = \mathcal{O}(1)$. All we ask of the likelihood is that it be a reasonably continuous function of position, so that there **is** an acceptable scale.



There appear to be barriers to free movement at special places such as $\theta = \frac{1}{2}$ (where passage between adjacent integers $0111111\cdots$ and $1000000\cdots$ requires all $B-1$ low bits to change). As elsewhere, these can be made harmless by carrying out the procedure with respect to a randomly offset origin.

There is nothing special about the halving procedure just described, other than being simple to implement in integer arithmetic. Any set of pre-defined nested domains would suffice: slice sampling was introduced by Neal [ref] in this more general form. Actually, Neal started his slice-sampling algorithm from some intermediate length scale, and could step outward by widening the domain as well as inward by shrinkage. In a multi-atom environment, that is less important. Each atom has left and right neighbours along the

Peano curve, and their location will often give sensible limits on that atom's movement. A location beyond can be considered "out-of-range", and rejected at once. Stepping out is no longer particularly necessary, because the early steps inward from the full number of bits mostly involve merely the trivial overhead of checking whether a trial location is in range.

The following fragment of pseudo-code illustrates the basic implementation of slice-sampling. The entry position is represented by the Bd-bit integer k.

```
$b \leftarrow B \times d$                                          full number of bits
$o \leftarrow \texttt{Uniform}[0, 2^b)$                             random origin
$a \leftarrow \texttt{Uniform}\big(0,\ L(k)\big)$                   level of acceptance probability
do{                                                                 loop
    $j \leftarrow \Big((k-o) \vee \texttt{Uniform}[0,2^b)\Big) + o$     trial position around $k$
    $b \leftarrow b - 1$                                            shrink interval around $k$
}while( $j$ is out-of-range or $L(j) < a$ )                         until $j$ is in range and if so is acceptably probable
```

The fact that the position almost certainly changes ($j \neq k$) more than justifies the mild extra cost of slice sampling, as opposed to straightforward Metropolis-Hastings rejection.

4. Annealing

In any large application, it is true almost by definition that the significant "volume" of the posterior occupies only a tiny fraction of the volume of the prior. Technically, the **information**, or negative entropy

$$H = +\int P(\theta)\log\bigl(P(\theta)/\pi(\theta)\bigr)d\theta, \quad P = \text{posterior},\ \pi = \text{prior},$$

is likely to be much bigger than unity, in keeping with the dimensionality of the problem. It is then difficult to jump directly from the prior to the posterior, which is why we need a Markov chain. Practically all samples from the prior will lie out on the far tails of the posterior, where the local structure may give poor guidance about how to approach the isolated peak(s). To avoid this, we divide the application into subsidiary steps within which H does not increase too much. We arrange that there is significant overlap between the initial and final distributions of each step, and the computation can then proceed in reasonable safety.

The traditional problem of locating a needle in a haystack is analogous: the needle occupies 1 unit in a haystack of volume $V = \exp(H)$. Instead of a direct search, which would take V or so trials, we divide the haystack into halves, quarters *etc*, thus managing to locate the needle in $\log(V) = H$ steps instead.

Although there are many possible paths between the prior and the posterior, one particular path is specially sympathetic to the formalism. I have tried a couple of others, but found no advantage. Let $L(\theta)$ be the likelihood

$$\Pr(D \mid \theta) = L(\theta)$$

and let λ be a numerical coefficient. Instead of working with L directly, we work with a modified likelihood $L^\lambda(\theta)$ which induces a modified posterior proportional to $L^\lambda(\theta)\pi(\theta)$.

Setting $\lambda = 0$ switches the likelihood off ($L^0 = 1$), so the modified posterior is simply the prior. Setting $\lambda = 1$ switches the likelihood on (L^1 = likelihood), so the modified posterior is simply the true posterior. And, by increasing λ gently from 0 to 1, we can divide the application into small, safe steps. In fact we can continue further: making λ even larger makes the modified posterior sharpen around the point(s) of maximum likelihood.

$$\begin{array}{ccccc} \lambda = 0 & \longrightarrow & \lambda = 1 & \longrightarrow & \lambda = \infty \\ \text{prior} & & \text{posterior} & & \text{maximum} \end{array}$$

We can thus use the same program for **both** Bayesian sampling of the posterior **and** maximum-likelihood, or indeed for maximising an arbitrary function.

In a thermodynamic analogy that is productive of terminology, λ is inverse temperature $1/T$ (so we call it "coolness"), and $-\log L$ is energy $\mathcal{E}$, whence the modified likelihood factor becomes the familiar $\exp(-\mathcal{E}/kT)$. This is why passage along this particular path is called "annealing". We also talk of a set of objects $\{\theta\}$ being "at equilibrium" if they sample the modified posterior faithfully.

Annealing also allows us to calculate the values of evidence E and information H. We generalise E, the evidence

$$E = \Pr(D) = \int \Pr(D \mid \theta)\Pr(\theta)d\theta = \int L\, d\pi,$$

to

$$E(\lambda) = \int L^\lambda d\pi$$

which differentiates to give

$$\frac{d(\log E)}{d\lambda} = \frac{\int L^\lambda \log L\, d\pi}{\int L^\lambda d\pi} = \langle \log L \rangle_\lambda \,.$$

As the central expression defines, the angle-brackets here denote averaging over the posterior as modified to coolness λ. The purpose of annealing is to calculate the posterior (by sampling) at unit coolness, so *a fortiori* we can sample at intermediate coolnesses to pick up the required averages. And, because $E(0) = \int d\pi = 1$, the evidence value we seek is simply their sum

$$E = E(1) = \exp \int_0^1 \frac{d(\log E)}{d\lambda}\, d\lambda = \exp \int_0^1 \langle \log L \rangle_\lambda d\lambda.$$

(I was startled by this result when I stumbled across it, and enthusiastically took it to the late – and widely missed – Edwin Jaynes. He looked at it and remarked that the formula was "well known to those who know these things". Thank you Ed. Such identities are indeed commonplace in thermodynamics. In numerical work, the method is called "thermodynamic integration".) A similarly short derivation yields

$$H(\lambda) = \lambda \langle \log L \rangle_\lambda - \log E(\lambda),$$

with $\lambda = 1$ being the case Bayesians usually want.

How should we control λ? One way of proceeding is to set up some chain of values of λ, and let the object(s) diffuse up and down, as well as equilibrating in θ at the current coolness [ref from Julian Besag]. If the chain is appropriate (so that the beginning and end of each link overlap well enough), and if the links are weighted by their correct evidence values, then any object that successfully wanders from the prior end of the chain to the posterior end, and back, is guaranteed to have given a faithful sample of the posterior. That sounds attractive. But the evidence values are numerical, and rely upon adequate equilibration of θ, so the guarantee is not what it seems. If an implementation failed to move θ at all, it would still appear to work. A variant, "replica exchange", starts by putting an object on each link of the chain, and letting them exchange position as they equilibrate. This avoids having to pre-acquire evidence values, but it starts far from equilibrium because at the beginning we only know how to sample θ at the prior end. Replica exchange, like everything else, relies on successful equilibration in θ.

I prefer not to control λ by diffusion. The number of coolness values will be H or so, so that an object will take H^2 iterates to diffuse along the chain. This seems wasteful by a large factor H. I think it is better to cool systematically. And I think it wise to take a hint from physics, where the efficient route to equilibrium is usually through slow cooling, trying to remain always close to equilibrium. All too often, systems lock into metastable states if they are quenched too rapidly.

A successful program, then, needs an "annealing schedule" that defines how fast the coolness λ may increase. Crudely, the change in λ per cooling step should presumably allow the relative information between its beginning and end to be $\mathcal{O}(1)$, hence the suggestion [Otten & van Ginneken 1984 quoted in Radford Neal (1993) technical report p.90] that entropy should decrease at a roughly constant rate, whatever rate that may be. But I think that you, the user, should have control of the overall numerical rate, if only because applications differ in their difficulty.

4.1. SELECTIVE ANNEALING

Let us use an ensemble of $\mathcal{N}$ member objects, perhaps $10 - 100$ of them, with the k^{th} having likelihood L_k. Suppose this is in equilibrium at coolness λ. We now wish to cool by an additional $\delta\lambda$. This could be done by weighting the objects by

$$w_k \propto {L_k}^{\delta\lambda}$$

or, with normalisation to $\langle w \rangle = 1$,

$$w_k = (L_k/\overline{L})^{\delta\lambda}, \quad \overline{L} = \left\langle L^{\delta\lambda} \right\rangle^{1/\delta\lambda}.$$

Weighted averages over the objects would then be faithful to the new coolness. However, log-likelihoods are usually large, so that the weights would become very non-uniform after significant cooling, leading to gross numerical inefficiency.

Instead, draw $\mathcal{N}$ new samples from the weighted ensemble, ensuring a mean multiplicity $\langle n_k \rangle = w_k$ for each original object. The members of the new ensemble will then have equal weight again. To reduce changes in the ensemble, it is desirable to keep the (integer) number of copies n_k as close as possible to its mean, being either the integer immediately below or immediately above $\langle n_k \rangle$. It is further desirable to treat similar objects as similarly as possible. For example, if 10 objects have $\langle n \rangle = 0{\cdot}3$, we should keep exactly 3 of them, omitting the other 7. Likewise, if 10 objects have $\langle n \rangle = 2{\cdot}4$, we should keep exactly 24 of them, taking either 2 or 3 copies of each.

These desiderata can be satisfied by ordering the L_k into increasing (or decreasing) order: for example with $\mathcal{N} = 4$ we might have:-

Object k		1		2		3		4	
Weight w		0·5		0·7		1·0		1·8	
Cumulant weight	0		0·5		1·2		2·2		4

We then draw an initialising random variable $r \in \texttt{Uniform}(0,1)$ to set a sequence $\{r,\, r+1,\, r+2, \ldots\}$. Whenever one of the sequence intersects the cumulant weight, we draw that one as a new object. For example, if $r = 0{\cdot}4$ the sequence would be $\{0{\cdot}4,\ 1{\cdot}4,\ 2{\cdot}4,\ 3{\cdot}4\}$, which intersects the cumulants at $k = \{1, 3, 4, 4\}$. The least likely object 1 is drawn once, object 2 is deleted, object 3 is taken once, and the most likely object 4 is taken twice. The net effect is to copy object 4 onto object 2, duplicating the former while deleting the latter. Depending on the value of r (random between 0 and 1), the various multiplicities would have been:-

	1	2	3	4	Object k
$r = 1$ ⋮ $r = 0{\cdot}5$	0	1	1	2	Actual multiplicities n
$r = 0{\cdot}2$	1	0	1	2	
$r = 0$	1	1	1	1	
	0·5	0·7	1·0	1·8	Mean multiplicities $\langle n \rangle$

Necessarily, the mean multiplicities agree with the quoted weights. And, provided the mean is correct, there is no harm in thus imposing some correlations.

I suggest that $\delta\lambda$ should be chosen so that in one cooling step no object is copied by more than some maximum — to be supplied by you the user as a floating-point number `Rate` (it need not be an integer, and will often be set less than 1 for safety).

$$w_{\max} = 1 + \texttt{Rate} \qquad \Rightarrow \qquad \delta\lambda$$

$w_{\max}$ is a monotonically increasing function of $\delta\lambda$, so this equation is easily soluble numerically. However, accidental coalescence of all the likelihoods L_k could make $\delta\lambda$ damagingly large. To make this less likely even when the ensemble is small, we can include a dozen or so extra likelihoods taken from recent ensembles. Even if these are a little atypical for the current coolness, it won't matter because the enhanced range will merely slow the cooling down a little, adding to safety. As a second, probably un-necessary, line of defence, $\delta\lambda$ can be prevented from more than doubling between cooling steps. Such reductions in $\delta\lambda$ stabilise the annealing schedule. They only matter when $\mathcal{N}$ is fairly small, but they enable the schedule to operate correctly even if $\mathcal{N}$ is only 1 (when copying objects is impossible). Finally, the user can impose whatever limit on coolness is appropriate, either 1 for Bayesian calculations, or no limit for maximisation.

4.1.1. *Imperfections*

Selective annealing cannot cool perfectly, because the ensemble is only finite. For a start, the member objects must lose some independence through the duplications, so that after 2 or more cooling steps the ensemble is no longer quite at equilibrium. So annealing must still be accompanied by equilibration over θ: there is no escape from that.

If $\delta\lambda$ were to be set large, the ensemble still could not anneal further than to the largest likelihood in its current membership, and this limits the amount of annealing that appears to take place. Also, the estimate of $\overline{L}$ that underlies the weights fluctuates statistically by $\mathcal{N}^{-1/2}$, and this induces a $\mathcal{O}(\mathcal{N}^{-1})$ systematic reduction in apparent cooling. Just as in elementary statistics when we estimate variance as $\sum_1^{\mathcal{N}} (\delta x)^2/(\mathcal{N}-1)$ instead of naïvely dividing by $\mathcal{N}$, we have the fixup

$$\delta\lambda^{(\text{apparent})} = \frac{\mathcal{N}-1}{\mathcal{N}}\,\delta\lambda.$$

Presumably it is slightly better to use this apparent cooling in the actual updating of λ, though the effect is invisible if the exploration of θ is reasonably efficient.

4.1.2. *Properties*

The most important feature of selective annealing is that objects in the ensemble jump from "bad" low-likelihood positions to such "good" high-likelihood places as have currently been found. Hence objects can escape from local maxima of likelihood without having to find an exit geometrically, or by tunnelling. In a problem having several likelihood peaks, only one object needs to find the "right" peak for all the others to copy across in due course. And, because of the regular nature of copying in this implementation, no "good" object ($L \geq \overline{L}$) is ever destroyed.

Qualitatively, the cooling speed $\delta\lambda$ per step varies inversely with the log-likelihood range. The general formula is non-linear, but the small-speed limit is

$$\delta\lambda = \texttt{Rate} \,/\, (\log L_{\max} - \log \overline{L}).$$

Thus, whenever the likelihoods are widely separated because of a phase change or other difficulty in the application, the cooling rate automatically slows in proportion to compensate. Cooling also slows if one object suddenly acquires a much higher likelihood than the others. Again, the program "recognises" that something interesting has happened, and takes care to slow down.

In the small-speed limit, the relative information between the beginning and end of a step depends on the standard deviation of the log-likelihood as

$$\delta H \simeq \big((\log L)_{\text{r.m.s.}}\, \delta\lambda\big)^2 \sim \texttt{Rate}^2.$$

Thus selective annealing is broadly in line with the hope that this relative information should be held more-or-less constant. However, I prefer to fix the cooling by selective annealing because (*a*) it has a direct computational interpretation in terms of copy operations, and (*b*) it focusses on the single most probable object, which must be right when that one happens to be unusual.

4.2. COMPARISON WITH STATISTICAL THERMODYNAMICS

Statistical thermodynamics is the application of probabilistic methods to large physical systems, and some of its formulas correspond closely with ours. Our basic Bayesian formulas are:

Prior	:	$\pi = \pi(\theta)$ where symbol θ includes number of atoms N and all attributes
Likelihood	:	$L = L(\theta)$ associated with coolness λ
Evidence	:	$E = E(\lambda) = \int \pi L^\lambda d\theta$
Annealed posterior	:	$P = P_\lambda(\theta) = \pi\, L^\lambda / E$
Information=−Entropy	:	$H = H(\lambda) = -\log E + \lambda\, d\log E/d\lambda = +\int P \log P\, d\theta$
	:	$\langle \log L\rangle = d\, \log E/d\lambda = \int P \log L\, d\theta$
	:	$d\log E = \langle \log L\rangle d\lambda$
	:	$H = -\log E + \lambda \langle \log L\rangle$

These are strikingly similar to formulas for a canonical (fixed N) ensemble of $\mathcal{V}$-state systems.

List of states	:	$j = 1, 2, \ldots, \mathcal{V}$
Energy states	:	$\mathcal{E}_j$ associated with temperature T
Canonical partition function	:	$Q = Q(T) = \sum \exp(-\mathcal{E}_j/T)$
Probability of state	:	$P = P(j \mid T) = \exp(-\mathcal{E}_j/T)\,/\,Q$
Entropy	:	$S = S(T) = \log Q + T\, d\log Q/dT = -\sum P(j) \log P(j)$
Energy	:	$\langle \mathcal{E}\rangle = T^2\, d\log Q/dT = \sum P(j)\mathcal{E}_j$
	:	$d\log Q = \langle \mathcal{E}\rangle\, dT/T^2$
	:	$S = \log Q + \langle \mathcal{E}\rangle/T$

This invites the following identifications:

Thermodynamic		Bayesian
j	$\longleftrightarrow$	θ
$\mathcal{E}_j$	$\longleftrightarrow$	$-\log L(\theta)$
$\langle\mathcal{E}\rangle$	$\longleftrightarrow$	$-\langle\log L\rangle$
$Q\,/\,\mathcal{V}$	$\longleftrightarrow$	E
$1\,/\,T$	$\longleftrightarrow$	λ
$S-\log\mathcal{V}$	$\longleftrightarrow$	$-H$

Statistical thermodynamics being a well-developed discipline, it may contain other techniques of use to us. In particular, there is a grand canonical ensemble which explicitly allows N to vary according to a chemical potential μ. Actually, the list of states in the canonical ensemble could already include differing numbers N of component parts, so from our point of view the grand canonical ensemble may offer nothing new. Let's see. The thermodynamic formulas for the grand canonical ensemble are:

List of states	:	$j = 1, 2, \ldots$ for each number of atoms N
Energy states	:	$\mathcal{E}_j(N)$ with temperature T
Grand canonical partition function	:	$\Xi = \Xi(T,\mu) = \sum_{N,j}\exp(-(\mathcal{E}_j(N) - N\mu)/T)$
Probability of state	:	$P = P(N, j \mid T,\mu) = \exp(-(\mathcal{E}_j(N) - N\mu)/T)\,/\,\Xi$
Entropy	:	$S = S(T,\mu) = \log\Xi + T\,\partial\log\Xi/\partial T = -\sum P(N,j)\log P(N,j)$
Number	:	$\langle N\rangle = T\,\partial\log\Xi/\partial\mu = \sum P(N,j)N$
Energy	:	$\langle\mathcal{E}\rangle = T(\mu\,\partial\log\Xi/\partial\mu + T\,\partial\log\Xi/\partial T) = \sum P(N,j)\mathcal{E}_j(N)$
	:	$S = \log\Xi + (\langle\mathcal{E}\rangle - \mu\langle\mathcal{N}\rangle)/T$

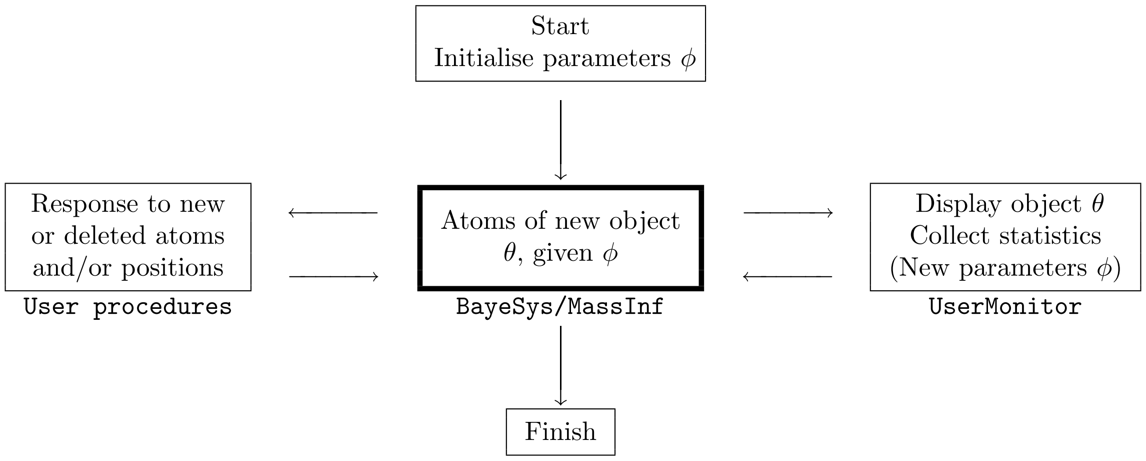
The two "potentials" T and μ allow a variety of "annealing" paths through the two-dimensional (T,μ) space, and it looks as if this could enable more direct control over the number of atoms. The geometric prior $\Pr(N) = \pi(N) = c^N$ in particular looks attractively similar to the chemical potential factor $e^{N\mu/T}$. However, in a Bayesian calculation powering the prior π is quite different from powering the likelihood L. For a start, it interferes with normalisation, which matters for a prior but not for a likelihood. At a more formal level, the prior represents our prior assumptions and is a **measure**. It is an intrinsically additive quantity, which can only appear linearly, as in $\int\pi(\theta)\ldots d\theta = \int\ldots d\pi$. If we were, regardless, to raise π to (say) the zeroth power, we would in effect be assigning uniform weight with respect to the coordinates. But coordinates are arbitrary, raising a difficulty about consistent treatment. And, in the case of a number of atoms, we certainly do **not** want to assign uniform weight over a possibly unlimited number. In our Bayesian context, different numbers of atoms are **not** *a-priori*-equivalent. So we do **not** want the powering coefficient μ/T to vanish. That sits uneasily with wanting T to anneal downwards from ∞, and suggests annealing with μ/T held constant. Which removes the whole point of considering the grand canonical ensemble in the first place, because we already have $\mu/T = \log c =$ constant as one of our priors.

Out of interest, I have tried "annealing" μ instead of λ, starting with $\mu = -\infty$ at which no atoms are present, and gradually allowing more atoms to appear. The method was hopelessly inefficient, because the full force of the likelihood was seen by each fresh atom, and this was wholly unsympathetic to easy movement. Finally, it would be permissible to factorise the likelihood into two or more factors $L = L_1L_2\cdots$, each with its own coolness $\lambda_1, \lambda_2, \cdots$, and to take an arbitrary path from coolnesses all 0 to all 1. But I see no advantage.

5. The BayeSys program

The following parts of this manual document what is actually implemented, which for various reasons can differ in detail from the foregoing tutorial material.

BayeSys (an acronym for **Baye**sian **Sys**tem) is a program for sampling the posterior of a system having an atomic prior, along with calculating the evidence value. It incorporates several engines for creating, destroying, and moving the atoms efficiently. As it attempts to modify the atoms, it passes each suggested modification to your user-procedures, which inform it about the associated likelihood change. In the light of this, BayeSys accepts, rejects, or changes its suggestion. After each complete iterate, when each atom has likely been changed, BayeSys passes all its atoms back to your procedure `UserMonitor`, for display of the current ensemble of sample objects $\{\theta\}$, and collection of any statistics you want. At this point, you may want to re-calibrate any "nuisance" parameters ϕ that had to be assigned. More properly, you should re-sample them from their posterior probability as calculated in accordance with the current object they are related to. This alternate sampling of relevant parameters (here θ and ϕ) is known as "Gibbs sampling" [ref. Geman and Geman]. It ensures that the joint posterior of atoms and nuisance parameters will be explored faithfully, which is how nuisance parameters are correctly estimated and eliminated.


Start
Initialise parameters ϕ
Response to new or deleted atoms and/or positions
User procedures
Atoms of new object θ, given ϕ
BayeSys/MassInf
Display object θ
Collect statistics
(New parameters ϕ)
UserMonitor
Finish

It is also your responsibility to demand that the computation finish — by signalling from `UserMonitor`.

5.1. THE BAYESYS PRIOR

As programmed in BayeSys, the prior $\Pr(n)$ for the number of atoms can be styled "uniform", "Poisson" or "geometric", according to the sign $(0, +, -)$ of the input parameter `Alpha` $= \alpha$. In each case, the distribution is qualified by a minimum number M and a maximum number N supplied through parameters `MinAtoms` and `MaxAtoms`,

$$M \leq n \leq N.$$

M is given directly by `MinAtoms`. It must be positive $(1, 2, \ldots)$ and not 0 (or negative). This means that BayeSys excludes the null object with no atoms — a restriction imposed for the convenience of both author and user. If you the user really want to include the null object, you can treat it as an alternative prior hypothesis, weighted as always through Bayes' theorem with

$$\begin{aligned}\Pr(D \mid n > 0) &= \text{Evidence from BayeSys (given as its logarithm);}\\ \Pr(D \mid n = 0) &= \text{Likelihood(Data} \mid \text{null object).}\end{aligned}$$

N is given by `MaxAtoms`, usually directly, except that `MaxAtoms` $= 0$ codes for $N = \infty$ (absence of a maximum). Omitting the maximum is prohibited for a uniform prior, which would be improper. N must, obviously, be at least as large as M. Equality is allowed, and forces BayeSys to use exactly that number of atoms.

The "uniform" prior (`Alpha` $= 0$) is

$$\Pr(n \mid \alpha = 0) \;=\; (N - M + 1)^{-1} \qquad \text{for } M \le n \le N$$

with mean and variance

$$\langle n \rangle = \tfrac{1}{2}(N + M), \qquad \operatorname{var}(n) = \tfrac{1}{12}(N - M)(N - M + 2).$$

The "Poisson" prior is specified by `Alpha` $= \alpha > 0$. With finite maximum, the implementation is binomial, offset by the minimum number.

$$\Pr(n \mid \alpha > 0) \;=\; \frac{(N - M)!}{(n - M)!\,(N - n)!}\, q^{n-M} (1 - q)^{N-n}\ , \qquad q = \frac{\alpha}{\alpha + N - M}$$

Supplying the binomial rate through `Alpha` $= \alpha$ rather than directly as q makes it impossible to supply an unusable value $q > 1$. The binomial mean and variance are

$$\langle n \rangle = (1 - q)M + qN, \qquad \operatorname{var}(n) = (N - M)\, q(1 - q).$$

If the maximum is omitted, the binomial reduces to Poisson, offset by the minimum number.

$$\Pr(n \mid \alpha > 0) = e^{-\alpha} \alpha^{(n-M)} / (n - M)!\,, \qquad \langle n \rangle = M + \alpha, \qquad \operatorname{var}(n) = \alpha$$

The "geometric" prior is specified by `Alpha` $= \alpha < 0$, exhausting all the setting possibilities. It is

$$\Pr(n \mid \alpha < 0) = \frac{1 - c}{1 - c^{N-M+1}}\, c^{n-M}, \qquad c = \frac{|\alpha|}{|\alpha| + 1}.$$

Again, supplying the ratio through `Alpha` in this way makes it impossible to supply a ratio $c > 1$ which could be unusable. The limit $\alpha \to -\infty$ is the uniform case alternatively coded as `Alpha` $= 0$. If the maximum is omitted, the geometric prior is no longer truncated and is

$$\Pr(n \mid \alpha < 0) = (1 - c)\, c^{n-M}, \qquad c = \frac{|\alpha|}{|\alpha| + 1},$$
$$\langle n \rangle = M + |\alpha|, \qquad \operatorname{var}(n) = |\alpha| \big(|\alpha| + 1\big).$$

Thus, as for the Poisson prior, α directly yields the suggested number of atoms.

Your selection of prior should be influenced by your expectation of the degree of complexity $\langle n \rangle$ that you expect in the object, and by the uncertainty $\sqrt{\operatorname{var}(n)}$ of that estimate. For many applications, I prefer to keep options open by omitting the maximum number of atoms and setting the minimum to 1. Then `Alpha` can be set to the number of atoms I expect to see, usually made negative because the geometric distribution is less committal than the Poisson ($\alpha \pm \alpha$ instead of $\alpha \pm \sqrt{\alpha}$). However, the precise settings ought not to influence the results very much. If they do, seek a reason because something may be amiss.

The last parameter needed to define the prior is `Ndim`, the number of attributes per atom, otherwise known as the dimensionality d. This has to be a positive integer. Each attribute or coordinate is restricted to the range $[\,0, 1]$, and the prior is uniform over the unit hypercube $[\,0, 1]^d$. The Peano-curve transformations of the hypercube are handled within BayeSys, and need not concern the applications programmer. BayeSys will provide you with suggested positions for its atoms, and in fact each coordinate will always be an odd multiple of 2^{-33} (assuming a 32-bit word length). You do not have to provide any positions yourself, but you can instruct your user-procedures to reject particular positions. If you do this, BayeSys will omit those parts of the hypercube from your prior.

5.2. THE BAYESYS ENGINES

BayeSys has several MCMC exploration algorithms or “engines”, currently LifeStory1, LifeStory2, GuidedWalk, Leapfrog1, Leapfrog2, Chameleon1 and Chameleon2. The LifeStory engines control the birth and death of atoms, and allow their movement along the space-filling curve. The GuidedWalk and Leapfrog engines allow atoms to move geometrically within the coordinate hypercube, thus letting the ensemble learn about and use the shape of the likelihood function. The Chameleon engines let atoms jump between the objects in your ensemble, thus allowing the objects to communicate. You can control the engines by setting the 7-bit-integer input parameter `Method` as follows.

	high						low	
`Method =`	GuidedWalk	Leapfrog2	Leapfrog1	Chameleon2	Chameleon1	LifeStory2	Peano	1 ON
	off	off	off	off	off	LifeStory1	raster	0 OFF
	64	32	16	8	4	2	1	

The lowest bit of `Method` switches between topologies. If the bit is ON the Peano curve is used, otherwise a simple raster is used (with attribute coordinates $0, 1, 2, \ldots$ taking progressively lower precedence). In one dimension, the two topologies are equivalent, so the switch has no effect. Generally, I recommend using the Peano topology unless you have a specific contrary need.

The next-to-lowest bit of `Method` switches between LifeStory1 and its more sophisticated form LifeStory2. These two are the only engines in BayeSys that are guaranteed to mix all the states irreducibly, so one of them must be present, and you are given no way of switching them both off. Generally, I recommend the extra power of LifeStory2, although its iterations are more expensive.

The highest five bits of `Mathod` switch the other individual engines on or off. For example, `Method`=27 (= 16+8+2+1) uses Leapfrog1 and Chameleon2, with LifeStory2 but not LifeStory1, in the context of Peano topology. Future development may yield new engines switched by higher bits.

None of the `Method` settings should change the ultimate statistical properties of the results, because BayeSys always aims to explore the posterior. It is only the efficiency of that exploration which changes. I suggest the strategy of developing an application with all bits ON (*i.e.* `Method` = 127 or equivalently `Method` = −1). Then, if you need to save computer resources, try switching bits OFF as long as your results are undamaged. Remember that you can control the speed of annealing with parameter `Rate`.

All the engines operate in the environment of an ensemble of $\mathcal{N}$ objects (input as parameter `ENSEMBLE`), each being a sample from the posterior distribution, as currently annealed. The LifeStory engines operate on just one object at a time, but the others can use two or even three. To control this, we consider the ensemble to be a single supersystem $\Theta = \{\theta_1, \theta_2, \ldots, \theta_{\mathcal{N}}\}$ with its own prior

$$\pi(\Theta) = \pi(\theta_1)\pi(\theta_2)\cdots\pi(\theta_{\mathcal{N}})$$

and its own likelihood

$$\mathcal{L}(\Theta) = L(\theta_1)L(\theta_2)\cdots L(\theta_{\mathcal{N}})$$

giving its own posterior

$$\mathcal{P}(\Theta) = P(\theta_1)P(\theta_2)\cdots P(\theta_{\mathcal{N}})$$

which factorises cleanly into the required posteriors of the constituent objects. Equilibrating just one object at a time is like exploring Θ by simple Gibbs sampling, but the supersystem allows more general exploration. Dimensionality need not be a curse; here it is an opportunity.

There is a reason for having several engines. Any individual engine can be thwarted by a particular form of likelihood. Even if provably convergent, it may be impractically slow. But a different engine might overcome the defect, only to be thwarted elsewhere. The combination of both engines will only be defeated by likelihoods that defeat each. BayeSys runs with up to six engines, and roughly balances its computation time between them. At worst, the cost of running six engines instead of the best alone is only a factor of six. The payoff comes when one of the engines rescues the system from a location that would trap the others.

We give no general guarantee of convergence. A sufficiently perverse likelihood will defeat any engine, even if composite, and you may never be aware of it. Indeed the scope for difficulty is so wide that “most” conceivable likelihoods will defeat any engine we are ever likely to build. On the other hand, our actual applications involve data that we feel will be interpretable, otherwise the data would not have been collected in the first place. We specialise in soluble problems!

5.2.1. *LifeStory1*

The LifeStory1 engine operates on just one object in the ensemble, and combines the processes of birth, death, and movement – hence the name – in a rather natural way. As explained above, the transition scheme for exploring the prior number n of atoms uses a birth rate β_n and death rate n per unit artificial time. When an event occurs, being birth or death in ratio $\beta_n : n$, the atomic number is to be incremented or decremented as appropriate.

$$\begin{tikzcd} & & \boxed{n+1} \\ & \boxed{n} \arrow[ur, "\beta_n"'] \arrow[dl, "n"'] & \\ \boxed{n-1} & & \end{tikzcd}$$

More slowly, but with equal validity, we can change the atomic number with probability 50%, leaving the indeterminate composite to be resolved afterwards.

$$\begin{array}{c} n\\ \text{or}\\ n-1\end{array} \longleftarrow \left\{\begin{array}{c}\boxed{n}\\ \text{and}\\ \boxed{n-1}\end{array}\right\} \overset{n}{\swarrow} \boxed{n} \underset{\beta_n}{\nearrow} \left\{\begin{array}{c}\boxed{n+1}\\ \text{and}\\ \boxed{n}\end{array}\right\} \longrightarrow \begin{array}{c} n+1\\ \text{or}\\ n\end{array}$$

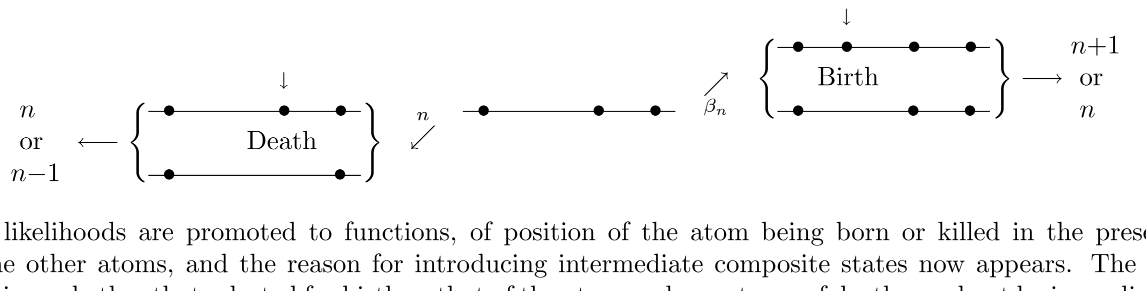
With likelihood factors L_j for j atoms, we proceed to sample from a birth composite, which has likelihood $L_{\text{birth}} = \frac{1}{2}(L_n + L_{n+1})$, according to the relative individual likelihoods:

$$\begin{aligned} \Pr(n+1 \mid \text{birth composite}) &= L_{n+1}/(L_n + L_{n+1}), \\ \Pr(\quad n \quad \mid \text{birth composite}) &= L_n \quad /(L_n + L_{n+1}). \end{aligned}$$

Likewise for a death composite of likelihood $L_{\text{death}} = \frac{1}{2}(L_n + L_{n-1})$:

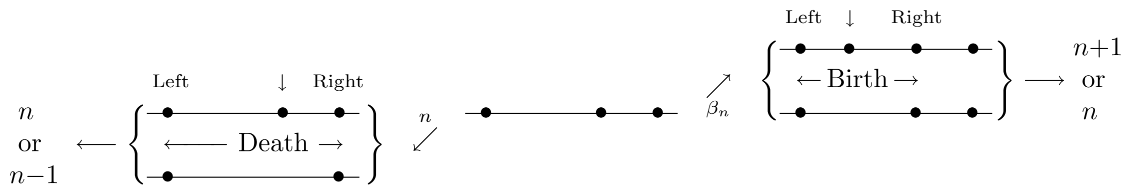
$$\begin{aligned} \Pr(\quad n \quad \mid \text{death composite}) &= L_n \quad /(L_n + L_{n-1}), \\ \Pr(n-1 \mid \text{death composite}) &= L_{n-1}/(L_n + L_{n-1}). \end{aligned}$$

When atoms have attributes, birth of an atom is accompanied by selection of a random coordinate θ, and death is accompanied by selection of a random identifiable-by-location atom, as flagged by downward arrows in the diagram below.



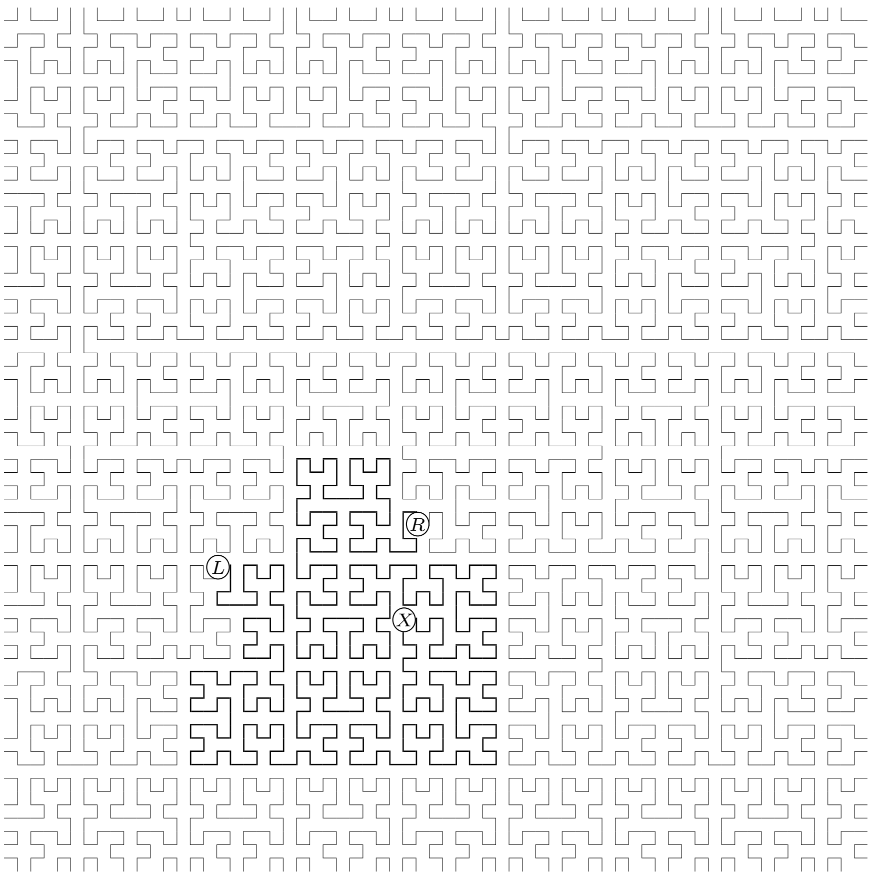
The likelihoods are promoted to functions, of position of the atom being born or killed in the presence of the other atoms, and the reason for introducing intermediate composite states now appears. The new position, whether that selected for birth or that of the atom under sentence of death, need not be immediately accepted or rejected by Metropolis-Hastings on the basis of that position only. Instead, movement is almost guaranteed by using slice sampling to adjust the composite state's position, using whichever of L_{birth} and

L_{death} is appropriate. Slice sampling centres on the original selected position, and for efficiency it is carried out between the left and right neighbour atoms of that initial choice.



The birth or death of a complete atom might well cause significant change in the likelihood, and it is wise to couple the opportunity with a wider view of the possibilities by allowing movement. If a birth event succeeds in incrementing the number of atoms, all well and good. A change has been made, and slice sampling has likely improved the suggested location. If it fails, then there has been no change, which is regrettable. If a death event succeeds in decrementing the number of atoms, a change has been made. If not, then at least the selected atom will almost certainly have been moved. So, overall, LifeStory1 should make useful changes at least half the time.

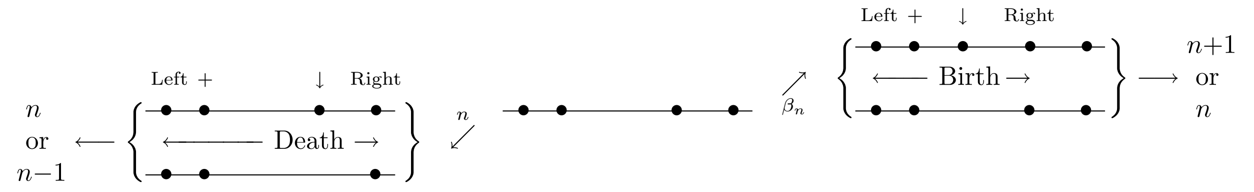
The illustration below illustrates the range allowed to a selected location X, constrained to move along the Peano curve (with origin randomised to $(5, 7)$ on a wraparound $2^6 \times 2^6$ grid) between its left and right neighbours L and R. With respect to the coordinates, this range is very roughly circular out to the distance of the neighbours.



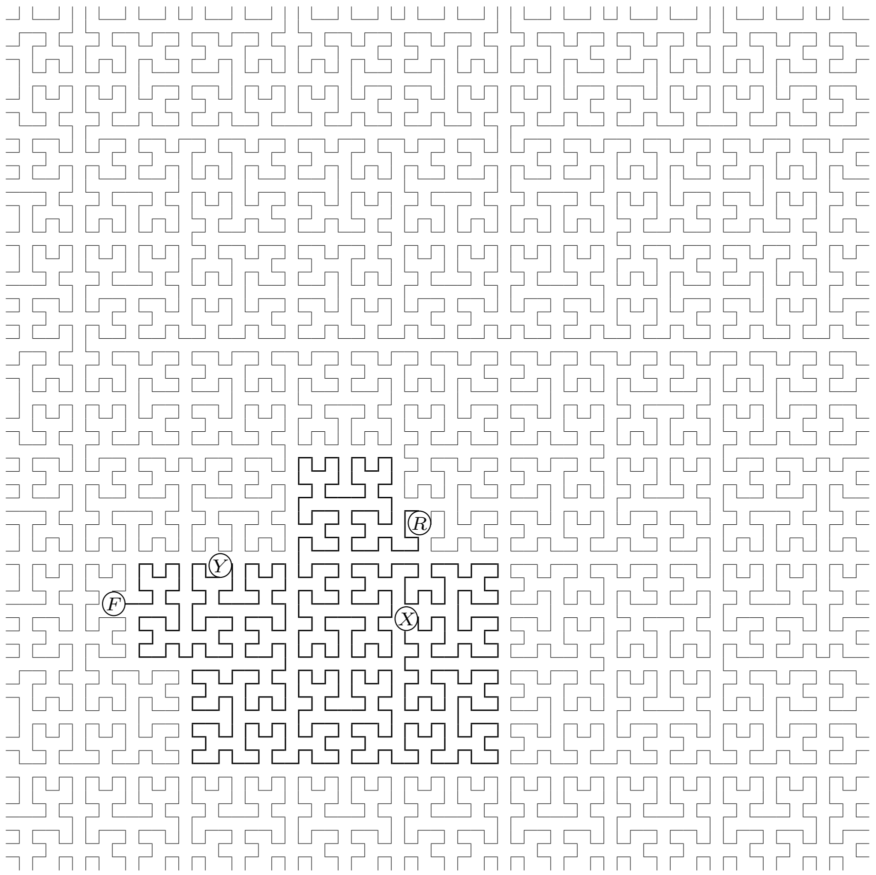
5.2.2. *LifeStory2*

The LifeStory2 engine, like LifeStory1, operates on just one object, and combines the processes of birth, death, and movement. The difference is that one of the neighbouring atoms is also allowed to move. Because they are indivisible changes, birth and death are often discriminated against by the associated changes in likelihood. The existing n atoms may have equilibrated as best they can, so that inserting or deleting a complete atom might always be unlikely, even though re-equilibration with $n \pm 1$ atoms might make the change acceptable or even preferable. In this situation, LifeStory1 is thwarted.

LifeStory2 allows a neighbouring atom to move aside to make room for the insertion, or move closer to compensate the deletion, so that the engine can jump around the barrier and equilibrate the number of atoms more effectively. In this way, a locally dominant constraint (such as mean position or total intensity) can remain satisfied even while an atom is created or destroyed.



The diagram shows the leftward neighbour "+" of the selected birth or death position "↓" being included in the process: inclusion of the rightward neighbour would be equally likely. In the upper state of either composite there are now two atoms that can be moved around, and in the lower state there is one. Slice sampling can equally well be carried out on two coordinates as on one. It centres on the original selected positions, and for efficiency it is done between the left and right neighbour atoms of those initial choices, as shown by the horizontal range arrows. In the 2-dimensional illustration, both X and its left neighbour (now called Y) can move between the further-left neighbour F and the original right neighbour R, allowing synergy between X and Y across a somewhat greater range.



If a birth event succeeds in incrementing the number of atoms, a good change has been made, with likely improvement to the suggested random location, and compensatory movement of a neighbour. Even if it fails, the neighbour will almost certainly have moved, so some change will have occurred. If a death event succeeds in decrementing the number of atoms, a neighbour will have moved to compensate, allowing the change to be more sympathetic to the data (hence more favoured). If not, then at least the selected atom and a neighbour will almost certainly have both moved. So, overall, LifeStory2 should almost always make useful changes, and thereby should be at least twice as powerful as LifeStory1.

One can envisage generalising LifeStory2 by allowing yet more atoms to move. However, bringing in more atoms will expand the domain they cover, and may well bring in additional constraints. Slice sampling would have to work harder to find an acceptable pattern, and the movements would be correspondingly smaller. My judgment is that LifeStory2 is often likely to be about best.

5.2.3. *GuidedWalk*

The LifeStory engines explore a very roughly spherical domain about one or two selected positions. Especially in many dimensions, where the likelihood function is increasingly capable of constraining an atom anisotropically, this can be inefficient. An atom is allowed to move only a short distance along strongly-constrained directions, and this restriction carries over to weakly-constrained directions if exploration is isotropic, making their exploration slow. As it happens, the number of attributes (dimensions) ascribed to an atom is often small, so the mere fact of using an atomic prior much reduces the difficulty: we seek to control only one or two atoms at a time, not the entire object. Even so, it may be useful to avoid the inefficiency by using the local shape of the likelihood function. In "classical analogue" style, high-order methods such as conjugate gradient (for maximisation) and hybrid Monte Carlo (for probabilistic exploration) come to mind. These methods use a sequence of intermediate evaluations (of gradient as well as value of likelihood) to discover and incorporate the local shape.

Alternatively, we could simply use the ensemble. Let X be a randomly-selected atom that we wish to move, in accordance with the local likelihood function. We may presume that the existence of X at its particular location already represents some feature of the likelihood, in which case other objects should also have atoms in similar location representing the same feature. Let L and R be left and right neighbour atoms of the position of X, drawn from different objects. If, indeed, each object contains just one atom for the feature in question, then that will lie fairly closely left of X along the Peano curve half the time, and fairly closely right of X the other half. So L from its object should be the appropriate corresponding atom about half the time, and so should R from its object. Even if the precise correspondence and symmetry are relaxed, there should remain an appreciable $\mathcal{O}(1)$ probability $\mathcal{P}$ that L and R have a similar environment to X. The offset vector $(\mathbf{R}-\mathbf{L})$ will then be obedient to the extent and shape of the local likelihood function. Hence it can be suggested as an appropriate increment for the position of X, yielding a new trial location

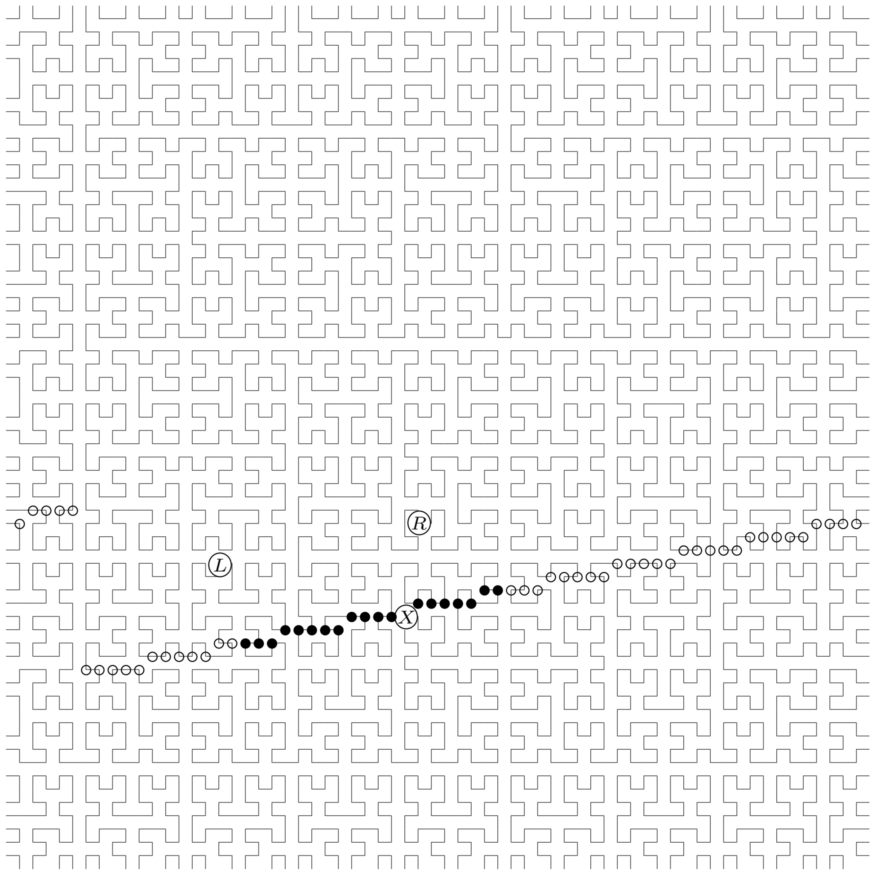
$$\mathbf{X}^{\text{trial}} = \mathbf{X} + s(\mathbf{R}-\mathbf{L}), \qquad s = \mathcal{O}(1).$$

Instead of a quasi-isotropic random walk, we have a **guided** walk in the direction $\pm(\mathbf{R}-\mathbf{L})$.

As usual, the scale s of the change cannot be too large. Larger values of $|s|$ increase the risk that the neighbours of X^{trial} will no longer be L and R, breaking detailed balance. With randomly scattered atoms and $|s| = 1$, the neighbours will only be correct with probability $\mathcal{O}(2^{-d})$, basically because there are 2^d quadrants in which other atoms might intervene. That would limit $|s|$ to $\frac{1}{2}$ or so, which reduces the potentially interfering atoms by the compensating volumetric factor 2^d. Moreover, if there are ν Gaussian constraints active on an atom, then $|s|$ is limited to about $\nu^{-1/2}$, otherwise the trial location will usually be rejected. However, if the local atoms do suffice to represent a local likelihood feature, then we might be allowed to go further, half way out to the next nearby feature. It is hard to make this sort of argument precise, and fortunately we can use slice sampling to keep s nearly optimal, whatever the restrictions.

Suppose there is one (or more) direction that is completely unconstrained, so that the likelihood function is extremely anisotropic. At any stage in the computation, the ensemble will have its corresponding atoms $X, L, R, \ldots$ distributed somehow along this axis, say with variance $\|\delta x\|^2 = \sigma^2$. After a step of the guided walk with $s = \nu^{-1/2}$, the variance of the newly accepted X^{trial} will be $(1+2s^2)\sigma^2 = (1+2/\nu)\sigma^2$, greater than before by a factor $1 + 2/\nu$. Hence the spread of atoms along the unconstrained direction(s) increases **exponentially**, with $\|\delta x\|$ gaining a factor like $e^{\mathcal{P}}$ after ν ensemble-wide applications of the guided walk. Exponential growth occurs because the algorithm is invariant under affine transformation, so it automatically adjusts to whatever scales are operative (until supposedly-corresponding atoms become so distant that they can no longer be identified as such). Of course, we do not expect to recognise exponential behaviour in realistic applications. What we may see, though, is easy exploration of badly conditioned likelihood functions.

To implement GuidedWalk, we extend the vector $(\mathbf{R}-\mathbf{L})$ across the unit hypercube, starting at the arbitrary origin of the current space-filling Peano curve and continuing until the most-rapidly-changing coordinate (ξ say) increments or decrements by 1. On the digital B-bit grid, this defines a "staircase" of 2^B points parameterised by ξ. This is displaced along the less-rapidly-changing axes until it passes through $\mathbf{X}$, as shown by the circles in the illustration below (note the horizontal wraparound of 5 points caused by the Peano origin being randomised to $(5,7)$).



Filled circles identify those points of the staircase which lie on that part of the Peano curve for which L and R remain the left and right neighbours in their own objects. These are the trial points that are in detailed balance with X. To select one, we use slice sampling along the staircase, which almost guarantees acceptable movement along what is appreciably often a useful direction.

5.2.4. *Leapfrog1 and Leapfrog2*

There are simpler ways of using the neighbours L and R of a selected atom X. Leapfrog1 uses just one leftward or rightward neighbour, L or R, preferably from an object different from X though it need not be. It takes

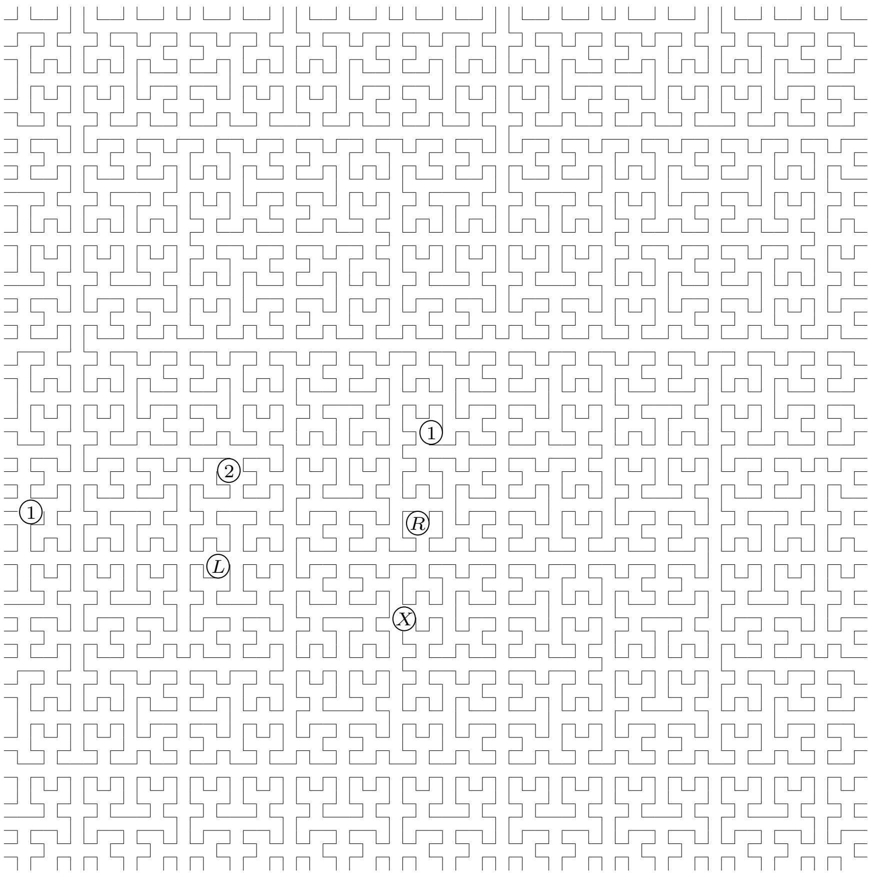
$$\mathbf{X}^{(1)} = 2\mathbf{L} - \mathbf{X} \qquad \text{or} \qquad \mathbf{X}^{(1)} = 2\mathbf{R} - \mathbf{X}$$

as a trial location, inverting X through L or R without any tunable coefficient. The trial location must be in detailed balance with X. Hence if L or R, in its member, is the left (right) neighbour of the location of X, then it must also be the right (left) neighbour of $X^{(1)}$ without any other intervening atoms. Whether this is likely depends on how atoms are distributed locally. Provided this neighbourhood condition holds, the Metropolis-Hastings rule can be used to accept or reject the suggestion on the basis of its likelihood relative to X.

Leapfrog2 uses two neighbours, L to the left and R to the right, preferably drawn from objects other than X, but they could be both from the same as X. It takes

$$\mathbf{X}^{(2)} = \mathbf{L} + \mathbf{R} - \mathbf{X}$$

as a trial location, inverting X through the midpoint of L and R (which may be usefully closer to the centre of the local feature than either L or R individually). The trial location will be in detailed balance if L remains its leftward and R remains its rightward neighbour. Provided this neighbourhood condition holds, the Metropolis-Hastings rule can again be used to accept or reject the suggestion on the basis of its likelihood relative to X. In the illustration below, Leapfrog1 can invert X to either of the points marked ① , and Leapfrog2 can invert X to ② .

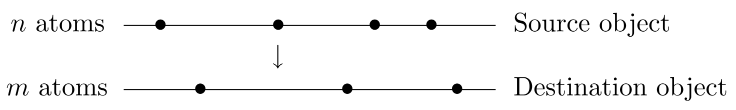


Like GuidedWalk, Leapfrog1 and Leapfrog2 are capable of increasing the spread of atoms along unconstrained directions **exponentially**. And, lacking the slice-sampling loop, they are simpler and faster to compute. However, if the atoms are subject to several constraints, the suggested changes will usually be rejected. Supposing that the local atoms are distributed in accordance with a locally Gaussian likelihood having ν constraints, each atom's chisquared misfit should be $\chi^2 \sim \nu$. But, according to the generating formulas, the misfits at the trial locations are likely to be larger: 5ν for Leapfrog1 and 3ν for Leapfrog2. The net effect is that the acceptance rate drops exponentially with the number of constraints, roughly like $e^{-0.4\nu}$ for Leapfrog1 and $e^{-0.25\nu}$ for Leapfrog2. This means that the Leapfrog engines can only operate well when the individual atoms have just a few constraints, but that will certainly be true if their number of attributes d is small.

No matter how long they are run for, the Leapfrog engines can only move atoms around the discrete lattice of points being integer combinations of the original positions. Whether or not this pattern of points is technically irreducible depends on Diophantine subtleties of whether the offsets in this lattice are co-prime in each dimension with the number of grid points, here 2^B. So the test of irreducibility may be less clear than one might suppose. In practice, we play safe and consider the Leapfrog engines to lack the irreducible property, resolving always to use them alongside irreducible engines like LifeStory which do provide full exploration.

5.2.5. *Chameleon1*

The Chameleon1 engine operates on two objects. It tries to make a randomly chosen atom jump from one object to another.



The idea is that an atom might be better placed in an object other than its source, especially if a useful patch of the hypercube is only just being discovered. Occasionally this could help the destination object out of a trap, so the engine could be worth including even if most of its transitions were rejected. Chameleon1 cannot be used on its own because its transitions are not irreducible: it can only use existing atom positions and cannot generate all the other possibilities. Hence it must be used with an irreducible engine such as LifeStory.

Care is needed with detailed balance. Let the original ensemble state "i" have n atoms in the source object and m in the destination. This occupancy has prior probability $\pi(n)\,\pi(m)$. The destination state "j" will have $n-1$ and $m+1$ atoms respectively, with prior probability $\pi(n-1)\,\pi(m+1)$. We require balance between the forward and backward transitions between these states.

$$\begin{tikzcd} n \text{ atoms} & & n{-}1 \text{ atoms} \\ \boxed{i} \arrow[rr, shift left, "T_{ji}\,\pi(n)\,\pi(m)"] & & \boxed{j} \arrow[ll, shift left, "T_{ij}\,\pi(n{-}1)\,\pi(m{+}1)"] \\ m \text{ atoms} & & m{+}1 \text{ atoms} \end{tikzcd}$$

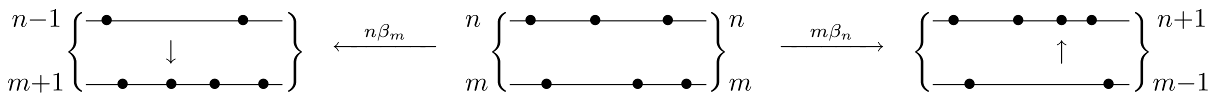
Hence the transition ratios must obey

$$\frac{T_{ji}}{T_{ij}} = \frac{\pi(n-1)\,\pi(m+1)}{\pi(n)\,\pi(m)} = \frac{\beta_m}{m+1}\,\frac{n}{\beta_{n-1}}$$

(remembering the birth and death rates for faithfully sampling the prior). This is implemented with a forwards rate

$$T_{ji} = n\beta_m\,dt$$

in infinitesimal interval dt of artificial time, implying the balancing backwards rate $T_{ij} = (m+1)\beta_{n-1}\,dt$. Starting with two objects with n and m atoms, jumps of an atom from n to m occur at rate $n\beta_m$, whereas jumps the other way from m to n occur at rate $m\beta_n$.



The time to the next event is exponentially distributed as

$$\Pr(\Delta t) = (n\beta_m + m\beta_n)\,\exp\big(-(n\beta_m + m\beta_n)\,\Delta t\big).$$

When that event occurs, it is either n-to-m or m-to-n in ratio $n\beta_m : m\beta_n$. Using a regular interval $\tau = \mathcal{O}(n+m)^{-1}$ between observations offers each atom an $\mathcal{O}(1)$ chance of jumping. This is the natural "period" for which to run Chameleon1 between a given pair of objects, randomly selected from the ensemble.

A suggested jump is accepted or rejected on the basis of the ensemble likelihood

$$\mathcal{L} \;=\; \prod_{\text{objects}} L(\text{object})$$

in which the only changeable factors are the likelihoods of the source and destination. According to the Metropolis-Hastings rule, a jump is accepted if

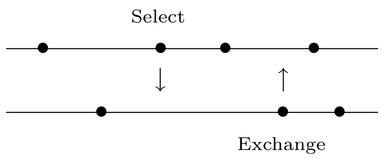
$$\mathcal{L}_{\text{new}} \geq \texttt{Uniform}(0, \mathcal{L}_{\text{old}})$$

and rejected otherwise.

It would be possible to use slice sampling to merge an atom's jump between objects with movement along the Peano line, as was done in LifeStory for birth and death. However, even that version would still be constrained to a fixed total number of atoms in the ensemble, so the engine by itself would only be cleanly irreducible and faithful to the prior for individual objects if the ensemble was very large. I do not expect the extra complication would yield any worthwhile advantage in power.

5.2.6. *Chameleon2*

In the Chameleon2 engine, pairs of atoms exchange their ensemble membership. Equivalently, they exchange positions.



To encourage synergy between the jumps, the chosen atoms should be close together, so that the exchange atom should be a neighbour (in its own object) of the position of the atom first selected. Detailed balance is then trivially assured. The transitions do not affect the number of atoms in either object, so they stay faithful to the prior without needing to adjust the relative rates. As was done for Chameleon1, it seems appropriate for an iterate to operate the engine on a given pair of objects for long enough that each atom is offered an $\mathcal{O}(1)$ chance of jumping.

Suggested jumps are again accepted or rejected on the basis of the ensemble likelihood, according to whether or not

$$\mathcal{L}_{\text{new}} \geq \texttt{Uniform}(0, \mathcal{L}_{\text{old}})\,.$$

6. Massive Inference (MassInf)

MassInf (an acronym for point-**Mass** atoms in **Inf**erence, originally with particular reference to **mass** spectrometry) is an extension to BayeSys, for use when attributes include intensity parameters which relate to **linear** data having Gaussian errors. In this special but quite common case, the intensity parameters can be processed semi-analytically, which means less load on the program, and enhanced power. Each atom in the prior model has intensity coordinates z which are treated separately from the other attributes θ. For example, the atoms for a black-and-white image would have a single intensity coordinate representing brightness, whereas atoms for a colour image might have three, one for each primary colour red, green, blue. The number of such intensities is called `Valency` as a mnemonic for the number of ways the atom can bond to data. For imagery the intensities would be positive, though in other applications z might take either sign.

6.1. MASSINF PRIORS

Massive inference allows a choice of four priors, selected by the `MassInf` parameter. In each case, the intensity is factorised as

$$z = \zeta q$$

where q is a dimensional unit of intensity (common to all the atomic intensities in any single object), and ζ is the individual dimensionless coefficient. The prior on q is assigned as

$$\pi(q) = q_0^{-2}\, q\, e^{-q/q_0}$$

where q_0 is an initial global guess about the expected magnitudes of the atomic intensities, and the "xe^{-x}" form is chosen to be tractable whilst keeping q away from both 0 and ∞.

The hyperparameter q_0 should have little effect, because q will usually be influenced much more strongly by the numerous data. But it must be given, and MassInf obtains it by peeking at the magnitude of the data. To be absolutely consistent with the strictest Bayesian paradigm, any dimensional dataset should be accompanied by a similarly dimensioned constant giving the expected size of phenomenon being observed. Nobody ever provides this, so nobody ought to object to the Bayesian analyst cheating a little, and peeking at the data to guess it.

The four MassInf priors for the relative intensities are:-

$$\pi(\zeta) = \begin{cases} \delta(\zeta - 1), & \texttt{MassInf} = 0 \text{ (``monkeys'', degenerate case with } \zeta \text{ fixed);} \\ \exp(-\zeta), & \texttt{MassInf} = 1 \text{ (``positive'', } \zeta > 0\text{, the commonest case);} \\ \frac{1}{2}\exp(-|\zeta|), & \texttt{MassInf} = 2 \text{ (``positive-negative'', } \zeta \text{ has either sign);} \\ \exp(-\frac{1}{2}\zeta^2)/\sqrt{2\pi}, & \texttt{MassInf} = 3 \text{ (``Gaussian'', } \zeta \text{ has either sign).} \end{cases}$$

The "monkey" prior (`MassInf` = 0) models a team of monkeys throwing balls of equal magnitude q at the unit box defined by the remaining coordinates θ. It was the original motivation for maximum entropy image reconstruction, as developed with the Cambridge group. On dividing the box into cells, the probability that N balls will be distributed as $(n_1, n_2, \ldots)$ is given by the degeneracy

$$\Pr(\mathbf{n} \mid N) \propto \frac{N!}{n_1!\, n_2! \, \ldots} = \Omega,$$

which suggests using the entropy $S = \log \Omega$ as a prior for intensity. Taking the model literally, though, imposes an unfortunate digitisation of intensity (to integer multiples of q), and as noted in the Overview the smoothed entropy form does not quite work as a Bayesian prior. Maximum entropy is a regularisation method, albeit with strong Bayesian connections, whilst the monkey prior is fully Bayesian.

The "positive" prior (`MassInf` = 1) avoids the monkey model's digitisation by letting the balls be of variable magnitude, as described by their exponential prior. Although the BayeSys/MassInf system need not and does not do this, the unit box of remaining coordinates θ can be divided into cells. Choosing a Poisson prior of mean α for the total number of atoms, a cell of size $\delta\theta$ will contain r atoms, distributed as Poisson with mean $\mu = \alpha\, \delta\theta$.

$$\Pr(r) = e^{-\mu}\mu^r / r!, \qquad r = 0, 1, 2, \ldots$$

With the intensity of each of the r atoms distributed as

$$\Pr(\zeta_i) = \exp(-\zeta_i), \qquad i = 1, 2, \ldots, r$$

the distribution for the total intensity in the cell is

$$\Pr(\zeta \mid r) = \begin{cases} \delta(\zeta), & r = 0; \\ e^{-\zeta}\,\zeta^{r-1}/(r-1)!, & r > 0. \end{cases}$$

and the net effect is that the total intensity in the cell is distributed as

$$\Pr(\zeta) = \sum_{r=0}^{\infty} \Pr(\zeta \mid r)\Pr(r) = e^{-\mu}\big(\delta(\zeta) + e^{-\zeta}\sqrt{\mu/\zeta}\; I_1(2\sqrt{\mu\zeta}\,)\big)$$

where I_1 is the first-order Bessel function. It is intellectually satisfying to find that the number of atoms can be summed away analytically, albeit at the cost of doing a cell-wise computation afterwards. It is also amusing to note that, almost regardless of the data, the most probable (MAPP) inferred intensity would thereby become precisely zero because of the sole surviving delta function — a delicious counter-example to MAPP estimation. Pursuing the algebraic development beyond these remarks takes us too far afield, into the realms of measure theory and Lévy-Khinchin representations.

The "positive-negative" prior (`MassInf` = 2) is the natural generalisation of the positive-only prior for applications where z may take either sign. Its cusp at zero is no disadvantage; it helps to reduce the intensity of faint atoms of doubtful significance.

The "Gaussian" prior (`MassInf` = 3) is an alternative prior when z may be of either sign. Its main disadvantage over "positive-negative" is that the expected intensity q is close to the r.m.s. value, so tends to be dragged up by any unusually large intensities. This leaves the fainter bulk of the intensities less well controlled, whereas the largest intensities are over-controlled and pulled back by the severe Gaussian tail. Objects of large dynamic range are thus recovered less well. The effect shows up quantitatively as a poorer (lower) value of the evidence for such objects, reflecting the improbability of having most intensities well below the r.m.s. magnitude, and some well above. On the other hand, the Gaussian prior $\exp(-x^2 - y^2)$ on two intensities x and y serves as a circularly symmetric prior for a complex $z = x + iy$, whereas the positive-negative $\exp(-|x| - |y|)$ does not.

6.2. MASSINF LIKELIHOOD

For MassInf to be used, the likelihood should factorise

$$\Pr(D \mid \text{atoms}) = L_{\text{Bayes}}(\theta)\, L_{\text{MassInf}}(z \mid \theta)$$

so that the intensity-dependent part L_{MassInf} is Gaussian, as from linear data.

An atom of unit intensity at position θ has a footprint (or "point-spread-function" or "Green's function") $\mathbf{f}(\theta)$ over the data, several such if it has several intensities. The footprints of the constituent atoms combine linearly to produce the mock data

$$\mathbf{F} = \sum_{\text{atoms}} z_i\, \mathbf{f}(\theta_i)$$

as a list of numbers which compare with the actual data $\mathbf{D}$. To keep the notation clean, we incorporate the standard deviation uncertainties σ into the definitions

$$\mathbf{X}\cdot\mathbf{Y} = \sum X_k Y_k/\sigma_k^2, \qquad \|\mathbf{X}\|^2 = \mathbf{X}\cdot\mathbf{X}$$

of inner product and 2-norm. We then write the chisquared misfit as

$$\chi^2(\mathbf{F}) = \|\mathbf{F} - \mathbf{D}\|^2$$

and this gives the likelihood

$$L_{\text{MassInf}}(\mathbf{F}) = \mathcal{Z}^{-1} e^{-\chi^2/2}$$

where $\mathcal{Z} = \prod \sqrt{2\pi\sigma_k^2}$ is the dimensional normalisation.

MassInf uses these formulas to calculate the likelihood efficiently. When an atom is created (with new intensity z) or destroyed (by removing its intensity) or moved (by destroying it at the old location then creating it at the new), the program updates the mock data by

$$\Delta\mathbf{F} = \mathbf{f}(\theta)\,\Delta z$$

and the likelihood by

$$\Delta(\log L_{\mathrm{MassInf}}) = -\Delta\mathbf{F}\cdot(\tfrac{1}{2}\Delta\mathbf{F} + \mathbf{F} - \mathbf{D})$$

which should be faster than re-calculating the factor from scratch.

The accompanying factor L_{Bayes} is often just 1 and ignorable, but it need not be. For example, the data might be intensities whose locations are themselves observed uncertainly, in which case the identification of an atom's footprint would become probabilistic, involving a L_{Bayes} "evidence" factor for its plausibility.

6.3. MASSINF UNIT OF INTENSITY

MassInf uses Gibbs sampling to explore the joint distribution of q and the other variables, alternating between re-calibrating q and performing the rest of its calculation to re-sample the atoms. At the end of each iterate, before returning the updated ensemble to you as a new set of objects, MassInf re-samples q. Temporarily write the mock data as $\mathbf{F}(z) = q\mathbf{F}^*(\zeta)$ to make explicit its dimensional scaling with q. With all other variables (including ζ) fixed, the q-dependence of the likelihood is

$$\Pr(D \mid q, \ldots) \;\propto\; \exp(\,q\mathbf{F}^*\cdot\mathbf{D} - \tfrac{1}{2}q^2\mathbf{F}^*\cdot\mathbf{F}^*)$$

and this combines with the prior for q to give a joint distribution

$$\Pr(D, q \mid \ldots) \;\propto\; q\exp(-q/q_0 + q\mathbf{F}^*\cdot\mathbf{D} - \tfrac{1}{2}q^2\mathbf{F}^*\cdot\mathbf{F}^*)$$

which in turn is proportional to the posterior $\Pr(q \mid D, \ldots)$. To re-calibrate q, MassInf re-samples from this "$x \times \mathrm{Gaussian}(x)$" distribution. That deals with q. For the rest of the calculation, q is merely a constant that can be included in the appropriate prior for z:

$$\pi(z) = \begin{cases} \delta(z-q), & \texttt{MassInf} = 0; \\ q^{-1}\exp(-z/q), & \texttt{MassInf} = 1; \\ \tfrac{1}{2}q^{-1}\exp(-|z|/q), & \texttt{MassInf} = 2; \\ \exp(-\tfrac{1}{2}z^2/q^2)/\sqrt{2\pi q^2}, & \texttt{MassInf} = 3. \end{cases}$$

6.4. MASSINF INTENSITIES

The BayeSys engines modify only one, or sometimes two, atoms at a time. For exposition, we first consider modifying only one atom (say the k^{th}), in the single-intensity case `Valency` = 1. Let

$$\mathbf{F}^- = \sum_{i \neq k} z_i \mathbf{f}(\theta_i)$$

be the mock data from all the other atoms, with the selected k^{th} removed. With all other variables fixed, the dependence of the likelihood on the selected intensity z_k is

$$\Pr(D \mid z_k, \theta_k, \ldots) \;\propto\; \exp\big(-z_k(\mathbf{F}^- - \mathbf{D})\cdot\mathbf{f}(\theta_k) - \tfrac{1}{2}\,z_k^2\,\mathbf{f}(\theta_k)\cdot\mathbf{f}(\theta_k)\big)$$

and this combines with the "positive" prior for z_k to give a joint distribution

$$\Pr(D, z_k \mid \theta_k, \ldots) \;\propto\; \exp\big(-z_k/q - z_k(\mathbf{F}^- - \mathbf{D})\cdot\mathbf{f}(\theta_k) - \tfrac{1}{2}\,z_k^2\,\mathbf{f}(\theta_k)\cdot\mathbf{f}(\theta_k)\big)$$

which in turn is proportional to the posterior $\Pr(z_k \mid D, \theta_k, \ldots)$. Hence we can reset z_k by sampling directly from this "truncated Gaussian(x)" distribution: we don't have to guess trial values which may be rejected. The other MassInf priors give similarly tractable distributions. Moreover, the joint distribution is **integrable** to an error function.

$$\Pr(D \mid \theta_k, \ldots) = \int_0^\infty \Pr(D, z_k \mid \theta_k, \ldots)\, dz_k = \int_0^\infty \text{Gaussian}(z)\, dz \;\Rightarrow\; \text{erf(coefficients)}$$

This is an explicit expression for the likelihood (or, if you prefer, the evidence) of θ_k alone. Hence we can explore the location θ on its own, with the standard BayeSys engines, **without** having to find the intensity z at the same time. Dimensionality is reduced and exploratory power is improved. Implicitly, all intensity values are explored in parallel, and we delay picking an intensity until **after** a new acceptable location has been found.

$$\begin{tikzcd}
\bullet \arrow[dr, "\text{kill}"'] & & \begin{bmatrix}\cdot\\ \cdot\\ \cdot\\ \cdot\\ \cdot\end{bmatrix} \arrow[r, "\delta\theta"] & \begin{bmatrix}\cdot\\ \cdot\\ \cdot\\ \cdot\\ \cdot\end{bmatrix} \arrow[r, "\text{pick}"] & \bullet \\
& 0 \arrow[ur, "\text{sum}"] & & & \\
\text{Old } z & 0 & \text{All } z & \text{All } z & \text{New } z
\end{tikzcd}$$

Also, Gaussian forms over the positive quadrant remain tractable in 2 dimensions, so that the LifeStory2 engine (which moves two atoms in the same object) still works. The intensities of both atoms integrate out together, and are picked together after the new locations have been found.

If an atom has several intensities (`Valency` > 1), the analysis continues to hold, with the z-integrals being promoted to more dimensions. However, the analytic evaluation of these relies on separability. So, for practical implementation, an atom's footprints over the data must not overlap. Thus you could not use MassInf if you had red, green, blue footprints from the same atom blurred into the same measurements. If you want to use the LifeStory2 engine with `Valency` > 1, there is another restriction, that a single footprint from one atom must not overlap more than one of the footprints from another.

To run MassInf, all you need to supply are the data D and their uncertainties σ, along with a procedure to give the footprint(s) $\mathbf{f}(\theta)$ at a trial position θ. The program takes care of the rest. MassInf can also auto-scale your uncertainties σ for you, if their absolute scale is unknown.

7. Display of results

A posterior object derived from an atomic prior is intrinsically spiky. As a function of θ it is a sum of delta functions, one at each location of an atom. Likewise, any accumulation of objects remains a set of delta functions: summing over many objects merely produces more. Yet most likelihood functions are spatially smooth because data have finite resolution. So the actual posterior ought to be correspondingly smooth. It is true that, for any pre-specified resolution, long-term averaging will eventually produce a smooth result, but only after large numbers of atoms have chanced to fall into each (small) cell $\delta\theta$. This can take far too long to compute directly. For display purposes, we want to show a limited number of atoms as a smooth function. Note that this is not part of Bayesian analysis: it is openly a matter of professional communication and aesthetics.

BayeSys gives you some guidance. Each atom it produces is accompanied by an estimate of its width, in the form of that fraction f of the hypercube's volume that the atom could plausibly range through. You may use this estimate as you please, depending upon your current application. In one dimension ($d = 1$), the hypercube is just the unit interval $[0, 1]$, and f is a range $\delta\theta$ within it. For display, an atom will presumably be shown as some sort of bell-shaped curve of correct area, correct centre, and width related to f. In two dimensions ($d = 2$), it would be natural to give each coordinate (θ_1, θ_2) a width $f^{1/2}$. Generally, in d dimensions, a width $f^{1/d}$ might be appropriate so that the volume remains f. On the other hand, you may wish to divide f differently among the various dimensions, if these represent intrinsically different quantities. You decide.

To find f, BayeSys keeps a historical log of its atoms, biassed towards the more recent and presumably better-equilibrated ones. These many atoms are all put onto a standard Peano curve, along which their mean density

$$\rho(x) \propto \frac{\text{number of atoms in interval}}{\text{length of interval } \Delta x}\ , \quad \text{normalised to } \textstyle\int \rho(x)dx = 1,$$

is tolerably well defined over intervals wide enough to cover at least several atoms. When a new object is produced, having n atoms, each may be expected to range over a fraction $1/n$ of the atoms in the log. Perhaps half of this may be assigned to randomness, yielding a width

$$f = \delta x = 1/(2\rho n)$$

expressed as a fraction of the length of the Peano line, or equivalently as a fraction of the volume of the hypercube. This can conveniently be computed as a fixed atom-count across the accumulated log.

In "important" parts of the posterior, where accumulated atoms congregate closely, the width is small. It is less than the size of the feature, so does not appreciably degrade the delineation of that part. In unimportant parts of the posterior, where a few atoms may wander about weakly constrained in some broad background, the width becomes large. Such stray atoms are displayed as wide and shallow, so that they no longer obtrude upon the eye as isolated sharp spikes. As a technical refinement, BayeSys uses whichever of the leftward-looking and rightward-looking estimates of ρ is larger. This narrows the width so as to prevent atoms near the edge of an important domain from spilling out into the background, which could degrade the display's visible resolution if it were allowed.

The diagram illustrates a log of atoms "o" placed on the Peano line ". . .". Below this are five atoms of a current object "|", and sufficient atoms are in the log that the fixed atom-count is 2 (in practice the count would be more because the log would be fuller, and the log would be sparse among the huge number of points on the Peano line). Arrows in the diagram show how the widths are constructed, by counting 2 atoms across the log to whichever side is closer.

```
..o..oo....oo.oo.....oo.ooo....ooooooo..ooooo...ooo.o...o...o....o...
         ←──|──→      ←──|──→    ←─|─→   ←─|─→     ←────|─────→
```

Atoms in or at the edge of a dense region are narrow, whereas atoms in sparse regions are wide.