Journal of Statistical Software
April 2009, Volume 30, Code Snippet 1. http://www.jstatsoft.org/

xsample(): An R Function for Sampling Linear Inverse Problems

Karel Van den Meersche
NIOO-CEME Yerseke

Karline Soetaert
NIOO-CEME Yerseke

Dick Van Oevelen
NIOO-CEME Yerseke

Abstract

An R function is implemented that uses Markov chain Monte Carlo (MCMC) algorithms to uniformly sample the feasible region of constrained linear problems. Two existing hit-and-run sampling algorithms are implemented, together with a new algorithm where an MCMC step reflects on the inequality constraints. The new algorithm is more robust compared to the hit-and-run methods, at a small cost of increased calculation time.

Keywords: linear modeling, underdetermined systems, Markov chain, R.

1. Introduction

In linear programming and system theory, a linear model is conventionally written in matrix notation as[1] $\mathbf{Ax} = \mathbf{b} + \epsilon$, with $\mathbf{x}$ a vector of unknowns, and ϵ an error vector. Additional equality and inequality constraints can be present, leading to a general formulation:

$$\begin{cases} \mathbf{Ax} = \mathbf{b} + \epsilon \\ \mathbf{Ex} = \mathbf{f} \\ \mathbf{Gx} \geq \mathbf{h} \end{cases} \tag{1}$$

This kind of problems are usually overdetermined, meaning that there is no solution for which $\epsilon = 0$. They can then be solved with quadratic programming (Lawson and Hanson 1995) techniques, in which case a norm of the error term $\epsilon = \mathbf{Ax} - \mathbf{b}$ is minimized, for example

[1]Notations: Vectors and matrices are in bold; scalars in normal font. Vectors are indicated with a small letter; matrices with capital letter. Indices between brackets indicate elements of vectors (as in $\mathbf{a}_{(i)}$) or matrices (as in $\mathbf{A}_{(i,j)}$). Rows or columns of matrices are indicated as $\mathbf{A}_{(i,)}$ (rows) or $\mathbf{A}_{(,j)}$ (columns). Indices without brackets ($\mathbf{q_1}$, $\mathbf{q_2}$) indicate vectors that are subsequent in a random walk.

the sum of squares $\sum \epsilon^2$. This is a constrained linear regression problem: parameters $\mathbf{x}$ are subject to the constraints $\mathbf{Ex} = \mathbf{F}$ and $\mathbf{Gx} \geq \mathbf{h}$.

In many real-life applications with a general lack of data, the linear model (1) is underdetermined. Some examples include metabolic flux analysis in systems biology (Edwards, Covert, and Palsson 2002), food web modeling (Vezina and Platt 1988), biogeochemical modeling of the oceans, and the identification of food sources in a grazer's diet using stable isotope data (Phillips and Gregg 2003). Applications in other fields may be found as well.

We define the feasible region of linear problem (1), L, as the part of the parameter space that contains all solutions of the reduced problem

$$\begin{cases} \mathbf{Ex} = \mathbf{f} \\ \mathbf{Gx} \geq \mathbf{h} \end{cases} \tag{2}$$

Algorithms that sample the feasible region of an underdetermined linear problem in a uniform way, have already been described in the literature (Smith 1984). Here we introduce an R function that includes these algorithms in addition to an algorithm developed by the authors, that is more stable in high-dimensional situations. The implemented function returns a sample set that is uniformly distributed over the feasible region of equation set (2) when $\mathbf{A}$ and $\mathbf{b}$ are lacking.

The model can also contain a number of linear equations $\mathbf{Ax} = \mathbf{b} + \epsilon$ with an error ϵ in the data vector $\mathbf{b}$. In that case, the generated sample set is restricted to the feasible region defined by (2), but is not uniformly distributed.

When equation (1) is underdetermined, there exist solutions for which $\epsilon = 0$, i.e. the model $\mathbf{Ax}$ can fit the data $\mathbf{b}$ exactly. Here, we assume that ϵ is normally distributed, i.e. $\epsilon \sim N(0, \mathbf{s})$.

In the absence of inequality conditions, it is straightforward to construct a series of samples $\mathbf{x}$ for which $\mathbf{Ax} - \mathbf{b} = \epsilon$ has the proposed distribution. However, when $\mathbf{x}$ is subject to inequality constraints ($\mathbf{Gx} \geq \mathbf{h}$), ϵ cannot be normally distributed.

Instead, a truncated normal distribution is proposed for $\mathbf{x}$:

$$p(\mathbf{x}) \propto e^{-\frac{1}{2}(\mathbf{Ax}-\mathbf{b})^\top \mathbf{W}^2 (\mathbf{Ax}-\mathbf{b})} \quad \text{if} \quad \mathbf{x} \in L \quad ; \quad p(\mathbf{x}) = 0 \quad \text{if} \quad \mathbf{x} \notin L \tag{3}$$

where the weight matrix $\mathbf{W} = \text{diag}(\mathbf{s}^{-1})$. This formulation penalizes samples $\mathbf{x}$ when $||\mathbf{Ax} - \mathbf{b}||$ increases, and leads to a normal distribution of $\mathbf{Ax} - \mathbf{b} \sim N(0, \mathbf{s})$ when there are no constraints.

Equation (1) is overdetermined when there is no exact fit $\mathbf{Ax} = \mathbf{b}$. ϵ then represents a model error term rather than uncertainties in the data:

$$p(\mathbf{x}) \propto e^{-\frac{1}{2}\sigma^{-2}(\mathbf{Ax}-\mathbf{b})^\top \mathbf{W}^2 (\mathbf{Ax}-\mathbf{b})} \quad \text{if} \quad \mathbf{x} \in L \quad ; \quad p(\mathbf{x}) = 0 \quad \text{if} \quad \mathbf{x} \notin L \tag{4}$$

Here the model standard deviation σ is a scalar parameter that is estimated together with the other parameters $\mathbf{x}$ (Gelman, Carlin, Stern, and Rubin 2004). In the absence of inequality constraints, the mean estimate of σ equals the standard deviation of the residuals of a weighted linear regression.

The R (R Development Core Team 2008) function `xsample()` is currently part of the **limSolve** package (Soetaert, Van den Meersche, and van Oevelen 2009), available under the GPL (General Public License) from the Comprehensive R Archive Network at `http://CRAN.R-project.`

org/. **limSolve** contains several tools for linear inverse modeling. Function `xsample()` takes the matrices $\mathbf{A}$, $\mathbf{E}$, $\mathbf{G}$ and the vectors $\mathbf{b}$, $\mathbf{f}$, $\mathbf{h}$ as input, together with a vector of standard deviations for $\mathbf{b}$ and a number of technical input parameters. In the next sections, the function and contained algorithms are explained, and some examples are provided.

2. Method

The `xsample()` function aims to produce a sample set of vectors $\mathbf{x}$ that fulfill a number of equality constraints, and are confined by a number of inequality constraints. They are either uniformly distributed within their feasible region, or their distribution depends on the value of linear combinations $\mathbf{Ax}$. This is done in two steps: (1) eliminate the equality constraints $\mathbf{Ex} = \mathbf{f}$ and (2) perform a random walk on the reduced problem.

2.1. Step 1: Eliminate equality constraints

The elements $x_{(i)}$ of $\mathbf{x}$ are not linearly independent; they are coupled through the equations in $\mathbf{Ex} = \mathbf{f}$. They are first linearly transformed to a vector $\mathbf{q}$ for which all elements $q_{(i)}$ are linearly independent. If solutions exist for the equations in (2) and a vector $\mathbf{x_0}$ is a particular solution of $\mathbf{Ex} = \mathbf{f}$, then all solutions $\mathbf{x}$ can be written as:

$$\mathbf{x} = \mathbf{x_0} + \mathbf{Zq} \tag{5}$$

$\mathbf{Z}$ is an orthonormal matrix, obtained from the QR-decomposition or singular value decomposition of $\mathbf{E}$ (Press, Teukolsky, Vetterling, and Flannery 1992), and serves as a basis for the null space of $\mathbf{E}$: $\mathbf{Z}^\top\mathbf{Z} = \mathbf{I}$ and $\mathbf{EZ} = \mathbf{0}$.

There are no equality constraints for the elements in $\mathbf{q}$. Thus, the problem is reduced to:

$$\begin{cases} \mathbf{A}'\mathbf{q} - \mathbf{b}' = \epsilon \\ \mathbf{G}'\mathbf{q} - \mathbf{h}' \geq 0 \end{cases} \tag{6}$$

with $\mathbf{A}' = \mathbf{AZ}$, $\mathbf{b}' = \mathbf{Ap} - \mathbf{b}$, $\mathbf{G}' = \mathbf{GZ}$ and $\mathbf{h}' = \mathbf{Gx_0} - \mathbf{h}$. In `xsample()`, a particular solution $\mathbf{x_0}$ of $\mathbf{Ex} = \mathbf{f}$ can either be provided as one of the input parameters or be calculated by `xsample()` as a particular solution using the Least Squares with Equalities and Inequalities (LSEI) algorithm (Haskell and Hanson 1981), available in the **limSolve** package as `lsei()`.

Because $\mathbf{p}$ meets the inequality constraints $\mathbf{Gp} \geq \mathbf{h}$, there is already one trivial solution of $\mathbf{q}$: the null vector $\mathbf{0}$. From this point, new points are sequentially sampled.

We want to know which distribution of $\mathbf{q}$ is necessary to obtain the targeted distribution of the sample set $\mathbf{x}$. If a vector $\mathbf{x}(\mathbf{q})$ is a function of $\mathbf{q}$, the PDF (probability density function) of $\mathbf{q}$ is a product of the PDF of $\mathbf{x}$ and the Jacobian determinant:

$$p(\mathbf{q}) = p(\mathbf{x})||\frac{\partial \mathbf{x}}{\partial \mathbf{q}}|| \tag{7}$$

In this case, as $\mathbf{Z}$ is orthonormal, the Jacobian is $||\frac{\partial \mathbf{x}}{\partial \mathbf{q}}|| = |\mathbf{Z}| = 1$. Therefore $p(\mathbf{x}) = p(\mathbf{q})$. This means that if $\mathbf{q}$ is sampled uniformly, then $\mathbf{x}$ is too.

2.2. Step 2: Random walk

Markov chain Monte Carlo (MCMC)

What's left to do, is to properly sample $\mathbf{q}$. This can be done numerically using an MCMC random walk. Especially for high-dimensional problems, this is more efficient than a grid-based approach. The Metropolis algorithm (Roberts 1996) produces a series of samples whose distribution approaches an underlying target distribution. In `xsample()`, new samples $\mathbf{q_2}$ are drawn randomly from a jump distribution with PDF $j(.|\mathbf{q_1})$ that only depends on the previously accepted point $\mathbf{q_1}$. The new sample point $\mathbf{q_2}$ is either accepted or rejected based on the following criterion:

$$\text{if} \quad r \leq \frac{p(\mathbf{q_2})}{p(\mathbf{q_1})} \quad \text{accept } \mathbf{q_2} \quad \text{else} \quad \text{keep } \mathbf{q_1} \tag{8}$$

with $0 < r \leq 1$ and $p(\cdot)$ the PDF of the target distribution. The only prerequisite for the sample distribution to converge to the target distribution with PDF $p(\cdot)$, is that the jump distribution from which a new sample is drawn, is symmetrical in the following sense: the probability to jump from $\mathbf{q_1}$ to $\mathbf{q_2}$, $j(\mathbf{q_2}|\mathbf{q_1})$, has to be the same as the probability to jump from $\mathbf{q_2}$ to $\mathbf{q_1}$, $j(\mathbf{q_1}|\mathbf{q_2})$. Three different jump distributions are implemented and are discussed further below.

In absence of matrix $\mathbf{A}$ and vector $\mathbf{b}$, the target distribution of $\mathbf{q}$ is uniform and thus:

$$\begin{aligned} &\text{if} \quad \mathbf{G'q_2} \geq \mathbf{h} \quad (\frac{p(\mathbf{q_2})}{p(\mathbf{q_1})} = 1 \quad \Rightarrow \quad \text{accept } \mathbf{q_2} \\ &\text{else} \quad p(\mathbf{q_2}) = 0 \quad \Rightarrow \quad \text{reject } \mathbf{q_2} \end{aligned} \tag{9}$$

If $\mathbf{A}$ and $\mathbf{b}$ are present, combining equations (4), (6) and (7):

$$\begin{aligned} &\text{if} \quad \mathbf{G'q} \geq \mathbf{h} \quad p(\mathbf{q}) \propto e^{-\frac{1}{2}\sigma^{-2}(\mathbf{A'q}-\mathbf{b'})^\top W^2(\mathbf{A'q}-\mathbf{b'})} \\ &\text{else} \quad p(\mathbf{q}) = 0 \end{aligned} \tag{10}$$

The expression for fixed standard deviations is easily obtained from (3) by setting $\sigma = 1$ and $\mathbf{W} = \text{diag}(\mathbf{s}^{-1})$. Otherwise, σ is estimated from fitting of the unconstrained model $\mathbf{Ax} - \mathbf{b} \sim N(0, \sigma)$.

Sampling the feasible region

New samples in the MCMC are taken from a symmetric jump distribution. A major challenge is to only sample points that fulfill the inequality constraints. Three algorithms that ensure this, are discussed in the next paragraphs. As a consequence, the sample set of vectors $\mathbf{q}$ and the derived sample set of vector $\mathbf{x}$, has a distribution that is bounded by the inequality constraints.

In a euclidean space, every inequality constraint defines a boundary of the feasible subspace. Each boundary can be considered a multidimensional plane (a hyperplane). One side of the hyperplane is the feasible range, where the inequality is fulfilled. The other side of the hyperplane is non-feasible. The hyperplanes are defined by the following set of equations:

$$\mathbf{G'}_{(,i)}\mathbf{q} - h'_{(i)} = 0 \quad \forall i \tag{11}$$

Three jump algorithms for selecting new points $\mathbf{q_2}$ were implemented: Two hit-and-run algorithms (Smith 1984): the random directions and coordinates directions algorithms and a novel mirror algorithm that uses the inequality bounds as reflective planes. All three algorithms produce sample points that fulfill all inequality constraints, and they fulfill the symmetry prerequisite for the Metropolis algorithm.

Random directions algorithm (rda)

The random directions algorithm (Smith 1984) consists of two steps: first a random direction is selected by drawing and normalizing a randomly distributed vector. Starting point and direction define a line in solution space. Then the intersections of this line with the hyperplanes defined by the inequality constraints are determined. A new point is then sampled uniformly along the line segment that fulfills all inequalities.

Coordinates directions algorithm (cda)

The only difference with the random directions algorithm, is that the coordinates directions algorithm (Smith 1984) starts with selecting a direction along one of the coordinate axes. This leads to a simpler formulation of the algorithm.

The mirror algorithm

The mirror algorithm was inspired by the reflections in mirrors and uses the inequality constraints as reflecting planes. New samples are taken from a normal jump distribution with $\mathbf{q_1}$ as average and a fixed standard deviation, called the jump length. With an increasing number of inequality constraints, more and more samples from an unmodified normal distribution will be situated outside of the feasible region and have to be rejected based on criterion (8). While this is a correct approach and the sample distribution will also converge to the targeted distribution, it is inefficient because many points are rejected. We propose an alternative sampling routine that uses the inequalities to ensure that every newly sampled point is situated in the feasible region.

If $\mathbf{q_1}$ is a point for which the inequality constraints are fulfilled, a new point $\mathbf{q_2}$ can be sampled in the following way: first $\mathbf{q_{2-0}}$ is sampled from a normal distribution in the unrestricted space, ignoring all inequality constraints:

$$\mathbf{q_{2-0}} = \mathbf{q_1} + \eta \tag{12}$$

with η drawn from a normal distribution with mean 0 and a fixed standard deviation. If $\mathbf{q_{2-0}}$ is in the feasible range (all inequalities are met), $\mathbf{q_{2-0}}$ is accepted as a sample point $\mathbf{q_2}$ and evaluated in the Metropolis algorithm (8).

If some inequalities are violated (Figure 1), then the new point $\mathbf{q_{2-0}}$ is mirrored consecutively in the hyperplanes representing the unmet inequalities: the line segment $\mathbf{q_1} \rightarrow \mathbf{q_{2-0}}$ crosses these hyperplanes. For each hyperplane, a scalar $\alpha_{(i)}$ can be calculated for which

$$(\mathbf{G'})_{(,i)}(\mathbf{q_1} + \alpha_{(i)}\eta) + h'_{(i)} = 0 \tag{13}$$

with $\eta = \mathbf{q_{2-0}} - \mathbf{q_1}$. The hyperplane with the smallest non-negative $\alpha_{(i)}$, call it $\alpha_{(s)}$, is the hyperplane that is crossed first by the line segment. $\mathbf{q_{2-0}}$ is mirrored around this hyperplane.

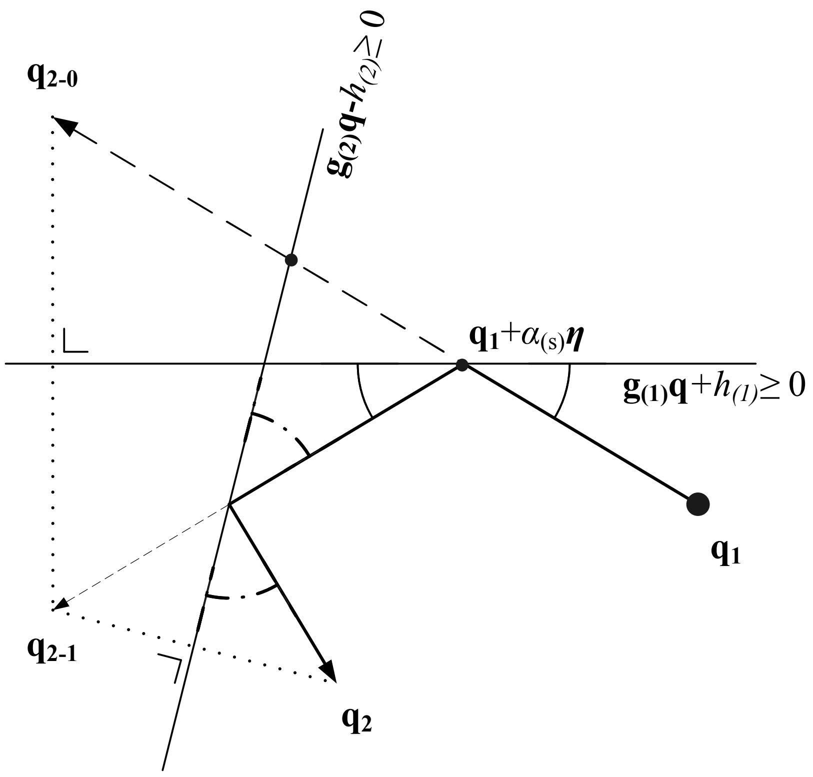
$\mathbf{q_{2\text{-}0}}$
$\mathbf{g_{(2)}q}\text{-}h_{(2)}\geq 0$
$\mathbf{q_1}+\alpha_{(s)}\boldsymbol{\eta}$
$\mathbf{g_{(1)}q}+h_{(1)}\geq 0$
$\mathbf{q_1}$
$\mathbf{q_{2\text{-}1}}$
$\mathbf{q_2}$

Figure 1: MCMC jump with inequality constraints functioning as mirrors. See text for explanation.

If the new point ($\mathbf{q_{2-1}}$ in Figure 1) still has unmet inequalities, a new set of $\alpha_{(i)}$'s is calculated from the line segment between the new point and the intersection of the previous line segment and the first hyperplane, i.e., $\mathbf{q_1}+\alpha_{(s)}\eta$. $\mathbf{q_{2-1}}$ is again reflected in the hyperplane with smallest non-negative $\alpha_{(i)}$. This is repeated until all inequalities are met. The resulting point $\mathbf{q_2}$ is in the feasible subspace and is accepted as a new sample point.

In most cases, the directional algorithms and the mirror algorithm converge to the same distributional result. However, we found that especially in high-dimensional problems, the mirror algorithm is still able to move away from the initial particular solution when the directional algorithms fail to do so. One possible explanation for this can be found in the initialisation of the MCMC with LSEI. LSEI often returns a solution in a corner of the feasible region, at the intersection of inequality constraints. In some circomstances, the line segment used by a random directions algorithm has then length zero and the algorithm fails to move away from the initial point.

In the mirror algorithm, η is drawn from a normal distribution with zero mean and a set of fixed standard deviations, which we call the jump lengths of the Markov chain. These jump lengths have a significant influence on the efficiency of the mirror algorithm, as they define the distance covered within the solution space in one iteration, but also the number of reflections in the solution boundaries. They can be set manually with the parameter `jmp` in `xsample()`. When sampling the feasible region uniformly, a suitable jump length is often in the same order of magnitude as the ranges of the unknowns.

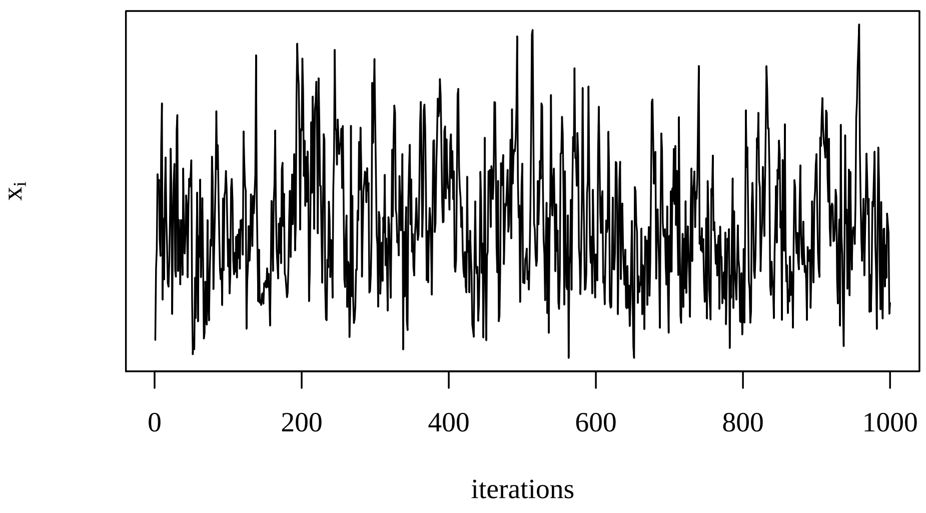

Figure 2: A good random walk of a parameter x_i, using `xsample()` with 1000 iterations.

When the default parameter setting `jmp = NULL` is used, a jump length is calculated internally, which gives quick and suitable results in most cases. Sometimes, these internally calculated jump lengths are too large, and the calculation time is too long. One can then turn to manually setting small jump lengths and gradually increasing them, until all elements in $\mathbf{x}$ are properly sampled. This can be checked by looking at the trace of the elements $\mathbf{x}_{(i)}$, which need to have an obviously random pattern, as illustrated in Figure 2.

Note that the hit-and-run algorithms rda and cda only work if G and H define a bounded feasible region. In an open or half open space, these algorithms will generate error messages because they draw from a uniform distribution confined by this feasible region.The mirror algorithm is not affected by this problem because new samples are drawn from a normal distribution instead of a uniform distribution.

3. Using the `xsample()` function in R

The default input for the `xsample()` function in R is:

```
xsample(A = NULL, B = NULL, E = NULL, F = NULL, G = NULL, H = NULL,
        sdB = 1, iter = 3000, outputlength = iter, burninlength = NULL,
        type = "mirror", jmp = NULL, tol = sqrt(.Machine$double.eps),
        x0 = NULL, fulloutput = FALSE, test = TRUE)
```

with the following arguments:

- `A` and `B`: a numeric matrix and a vector that contain the coefficients of the equations used to penalize the samples according to a normal distribution: $\mathbf{Ax} = \mathbf{b} + \epsilon$.

- `E` and `F`: a numeric matrix and a vector that contain the coefficients of the equality constraints, $\mathbf{Ex} = \mathbf{f}$.
- `G` and `H`: a numeric matrix and a vector containing the coefficients of the inequality constraints, $\mathbf{Gx} \geq \mathbf{h}$.
- `sdB`: a vector with fixed standard deviations of $\mathbf{b}$.
- `iter`: the number of iterations.
- `outputlength`: the number of samples of $\mathbf{x}$ in the output; this value is smaller than or equal to `iter`.
- `burninlength`: the number of initial iterations that are not included in the output.
- `type`: the algorithm used to sample new points: one of: `"mirror"`, (mirroring algorithm), `"rda"` (random directions algorithm) or `"cda"` (coordinates directions algorithm)
- `jmp`: jump length of the transformed variables $\mathbf{q}$: $\mathbf{x} = \mathbf{x_0} + \mathbf{Zq}$ (only if `type == "mirror"`); if not provided, a reasonable jump length is calculated internally.
- `tol`: tolerance for equality and inequality constraints.
- `x0`: initial (particular) solution $\mathbf{x_0}$ for the equality and inequality constraints.
- `fulloutput`: if `TRUE`, the transformed variables $\mathbf{q}$ are included in the output.
- `test`: if `TRUE`, `xsample()` will test for hidden equalities (i.e. equalities that are implemented as a set of two inequalities). This may be necessary for large problems, but slows down execution a bit.

The returned value of `xsample()` is a list containing:

- `X`: a numerical matrix with `outputlength` rows containing the samples.
- `acceptedratio`: ratio of acceptance (i.e. the ratio of the accepted runs in the MCMC over the total number of iterations).
- `Q`: only returned if `fulloutput = TRUE`: a numerical matrix with `outputlength` rows containing the transformed samples $\mathbf{q}$.
- `p`: only returned if `fulloutput = TRUE`: probability vector for all samples (one value for each row of `X`)
- `jmp`: the jump length used for the random walk. Can be used to check the automatically generated jump length.

4. Applications of `xsample()` and interpretations of the results

`xsample()` is primarily aimed at underdetermined systems. It samples their solution space and in addition, attempts to account for error in the data vector $\mathbf{b}$. However, it also works if the input is overdetermined. A model error can then be estimated from the difference between model and data as well. In this section, we illustrate two underdetermined scenarios for which `xsample()` may be used: underdetermined problems with `A = B = NULL`, underdetermined problems with `A` and `B` not `NULL`. We also briefly discuss the use of `xsample()` for overdetermined problems.

4.1. `A = NULL` and `B = NULL`: The mink diet composition

The most straightforward application of `xsample()` is to uniformly sample underdetermined systems with a set of equality constraints `(E,F)` and inequality constraints `(G,H)`, and no input `A` or `B`. The first example is a low-dimensional, underdetermined problem estimating the relative proportion of food sources of a predator based on stable isotope data.

The mink data set, from Ben-David, Hanley, Klein, and Schell (1997) is a small data set used in the program Isosource, which estimates the contributions of food sources to the diet of a consumer, based on stable isotope compositions (Phillips and Gregg 2003). Isosource searches all possible compositions with a grid-based approach and retains those compositions that are sufficiently close to the equality constraints imposed by the isotopic data. Our method obtains similar results, but obtains exact solutions instead of approximations. It also searches the solution space in a more direct and efficient way.

Stable isotope ratios of carbon ($\delta^{13}C$) and nitrogen ($\delta^{15}N$) in a predator, the Canadian Mink,

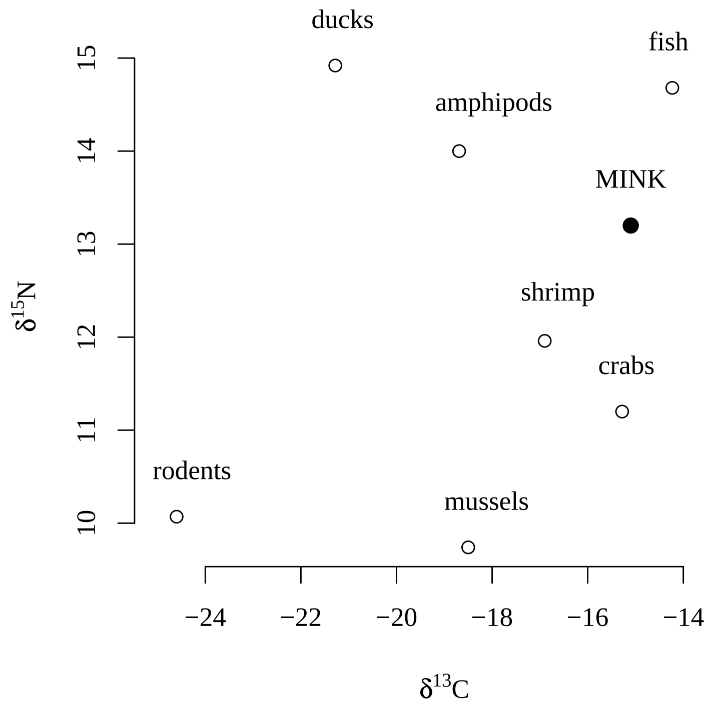

Figure 3: Isotopic composition of Canadian mink and seven candidate food sources.

and seven candidate food sources are provided (Figure 3). The data are given without error estimate, and are considered exact. As the isotopic composition of the predator is a simple weighted average of the compositions of its food sources, and the fractions sum to one, there are three equality constraints. Also, all fractions have to be positive, which gives seven inequality constraints. In matrix notation this can be written as:

$$\begin{cases} \begin{bmatrix} \delta^{15}N_1 & \cdots & \delta^{15}N_7 \\ \delta^{13}C_1 & \cdots & \delta^{13}C_7 \\ 1 & \cdots & 1 \end{bmatrix} \cdot \begin{bmatrix} \alpha_1 \\ \vdots \\ \alpha_7 \end{bmatrix} = \begin{bmatrix} \delta^{15}N_{\text{Mink}} \\ \delta^{13}C_{\text{Mink}} \\ 1 \end{bmatrix} \\ \mathbf{I} \cdot \alpha \geq \mathbf{0} \quad ; \quad \alpha = (\alpha_1, ..., \alpha_7) \end{cases} \tag{14}$$

Filling in the numbers, this results into:

$$\begin{cases} \begin{bmatrix} 14.7 & 9.7 & 11.2 & 12.0 & 10.1 & 14.0 & 14.9 \\ -14.2 & -18.5 & -15.3 & -16.9 & -24.6 & -18.7 & -21.3 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} \cdot \begin{bmatrix} \alpha_{\text{fish}} \\ \alpha_{\text{mussels}} \\ \alpha_{\text{crabs}} \\ \alpha_{\text{shrimp}} \\ \alpha_{\text{rodents}} \\ \alpha_{\text{amphipods}} \\ \alpha_{\text{ducks}} \end{bmatrix} = \begin{bmatrix} 13.2 \\ -15.1 \\ 1 \end{bmatrix} \\ \alpha \geq \mathbf{0} \quad ; \quad \alpha = (\alpha_{\text{fish}}, ..., \alpha_{\text{ducks}}) \end{cases} \tag{15}$$

The problem has three equations with seven unknowns and is thus underdetermined. The first two equations from (15) are implemented in **limSolve** as dataset `Minkdiet`. A set of samples that are uniformly distributed over the solution space defined by (15), are generated in R as follows:

```
R> xs <- xsample(E = rbind(Minkdiet$Prey, rep(1,7)),
+    F = c(Minkdiet$Mink,1), G = diag(7), H = rep(0,7), iter = 3000)
R> pairs(xs$X)
```

The results using the mirror algorithm are shown in Figure 4. With only 3000 jumps, the solution space is sampled thoroughly and the conclusion is that fish and crab are the most probable food sources for the mink. These results match the results presented by Phillips and Gregg (2003), but are obtained in a more straightforward manner.

When the only information we have on a set of parameters are a number of linear relationships and parameter ranges, as in this example, then to our best knowledge, every parameter set that fulfills these relationships has equal probability of occurring. The samples produced by `xsample()` are distributed uniformly over the feasible region of these parameters. They can be used to estimate value ranges, quantiles and means of the expected parameter values.

In the diagonal of Figure 4, histograms of the parameter values are shown. While the probability of each parameter set is equal, this is obviously not the case for the marginal distributions of the individual parameters.

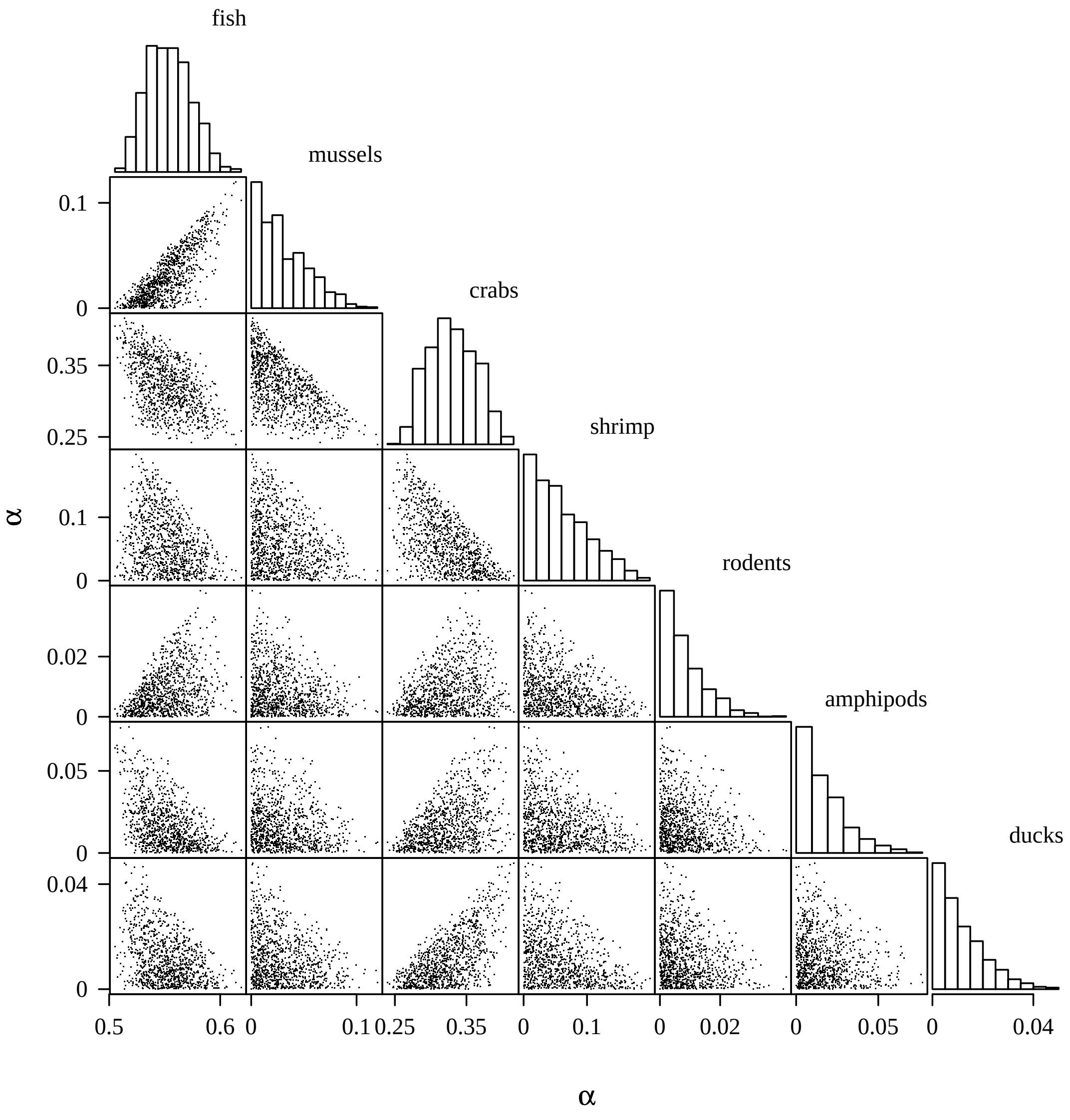

Figure 4: 3000 MCMC samples of the solution space of the mink diet problem: The lower half of the figure contains pairwise scatter plots of the fractions of the different food sources contributing to the diet. In the diagonal are density plots of the marginal distributions of these fractions.

4.2. An underdetermined problem with `A` and `B` not `NULL`

In the Mink example, all data were treated as exact values without uncertainties. As a result, the feasible region was sampled uniformly. `xsample()` allows for introducing some uncertainty in the equations via the parameters `A`, `B` and `sdB` (corresponding to $\mathbf{A}$, $\mathbf{b}$ and $\mathbf{s}$, respectively). The implications were already discussed in the introduction. We illustrate the effect of this in a very simple problem: consider two positive fluxes $x_{(1)}$ and $x_{(2)}$. If the sum of two fluxes

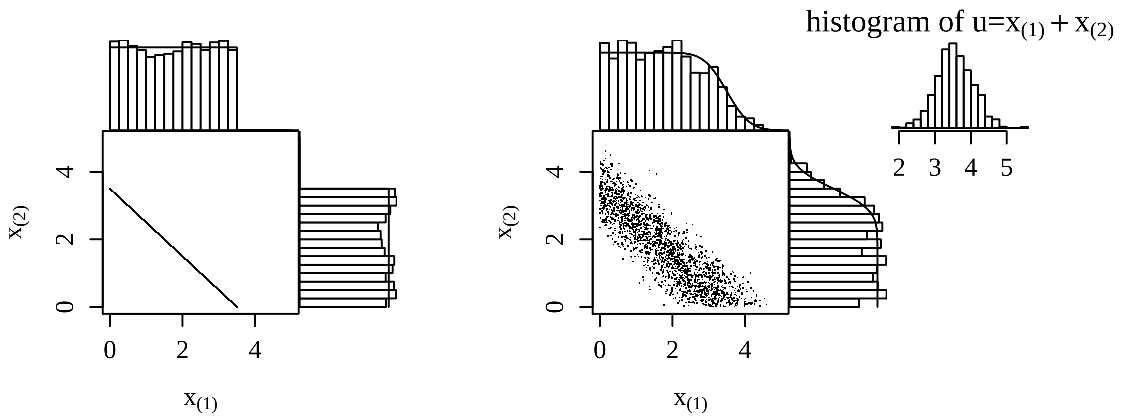

Figure 5: Illustration of the effect of equations $\mathbf{Ax} - \mathbf{b} = \epsilon$ on the sample distribution: two parameters with a fixed sum (left panel) versus two parameters with sum subject to variation (right panel). Histograms of the marginal distributions are plotted in the margins. See text for details.

is considered exactly 3.5, then the `xsample()` input becomes:

$$\begin{cases} \mathtt{E} = \begin{bmatrix} 1 & 1 \end{bmatrix} & ; \quad \mathtt{F} = \begin{bmatrix} 3.5 \end{bmatrix} \\ \mathtt{G} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} & ; \quad \mathtt{H} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \end{cases} \tag{16}$$

and the solution is trivial: the estimates of $x_{(1)}$ and $x_{(2)}$ each have a uniform distribution between 0 and 3.5, while their sum is constant. The result is shown in the left panel of Figure 5.

Now consider a normal distribution for the sum of $\mathbf{x}$: $u = x_{(1)} + x_{(2)} \sim N(3.5, 5)$. As argued in the introduction, this is not possible because $u \geq 0$, and expression (3) is used instead:

$$p(\mathbf{x}) \propto e^{-\frac{1}{2}(\frac{x_{(1)}+x_{(2)}-3.5}{0.5})^2} \quad \text{if} \quad \mathbf{x} \geq 0 \tag{17}$$

The probability outside the feasible region ($x_{(1)} < 0$ or $x_{(2)} < 0$), is still assumed zero.

The `xsample()` input then becomes:

$$\begin{cases} \mathtt{A} = \begin{bmatrix} 1 & 1 \end{bmatrix} & ; \quad \mathtt{B} = \begin{bmatrix} 3.5 \end{bmatrix} \quad ; \quad \mathtt{sdB} = \begin{bmatrix} 0.5 \end{bmatrix} \\ \mathtt{G} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} & ; \quad \mathtt{H} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \end{cases} \tag{18}$$

The results are illustrated in the right panel of Figure 5. Although $p(\mathbf{x})$ is highest when $x_{(1)} + x_{(2)} = 3.5$, this is not the case for the distribution of the sum itself. Because of the shape of the feasible region, there are more feasible vectors $\mathbf{x}$ for larger values of $u = x_{(1)} + x_{(2)}$ (Figure 5), causing the distribution of u to be skewed towards these larger values.

Generally, the shape of the feasible region will influence the distribution of $\mathbf{Ax}$ in a complex way which makes it difficult to produce a sample set $\mathbf{x}$ for which $\mathbf{Ax}$ has a predictable distribution.

4.3. Applying `xsample()` on overdetermined systems

In case the problem is overdetermined, a set of samples can also be generated from bootstrapping a constrained linear regression, e.g., using `orlm()` from the R package **ic.infer** (Groemping 2008). The results will however be different: samples that are located outside of the feasible region in the unconstrained model, are mapped on the borders of the feasible region. This is consistent with formulations of the underlying distribution of the parameters in a constrained linear regression (Shapiro 1988).

In contrast, `xsample()` will ignore the points outside of the feasible region and thus lead to a truncated normal distribution. The distribution of the samples can therefore not be interpreted as a probability distribution of the parameter estimate in a constrained linear regression. One should always be aware of these differences.

5. Concluding remarks

The `xsample()` algorithm successfully and efficiently produces a sample set that is uniformly distributed over the feasible region of an underdetermined linear problem. The sample set can also be non-uniformly distributed around an extra set of linear equations, and has a truncated normal distribution when the problem is overdetermined.

Quick convergence of the sample set to the target distribution (3) or (4) is one of the advantages of the algorithm. Convergence can be tested using tools for output analysis and diagnosis of MCMC, such as the R package **coda** (Plummer, Best, Cowles, and Vines 2009).

Calculation times are in most cases in the order of seconds to minutes on a modern pc. This makes the function an attractive alternative to linear or quadratic programming, offering a more complete answer to underdetermined linear problems and complementing the more traditional approaches.

Acknowledgments

We like to thank Prof. Dr. Carlo Heip and Prof. Dr. Jack Middelburg for guidance and for providing research facilities, and an anonymous reviewer for providing feedback. We also like to thank Dr. Dave Callaghan for proofreading and useful suggestions. This research was supported by a grant from the Flemish Fund for Scientific Research (FWO), extra funding from Ghent University and by the HERMES project, EC contract no GOCE-CT-2005-511234, funded by the European Commission's Sixth Framework Program under the priority 'Sustainable Development, Global Change and Ecosystems'. This is contribution 4516 from the Netherlands Institute of Ecology.

References

Ben-David M, Hanley TA, Klein DR, Schell DM (1997). "Seasonal Changes in Diets of Coastal and Riverine Mink: The Role of Spawning Pacific Salmon." *Canadian Journal of Zoology*, **75**, 803–811.

Edwards JS, Covert M, Palsson B (2002). "Metabolic Modeling of Microbes: The Flux Balance Approach." *Environmental Microbiology*, **4**(3), 133–140.

Gelman A, Carlin JB, Stern HS, Rubin DB (2004). *Bayesian Data Analysis*. 2nd edition. Chapman & Hall, London.

Groemping U (2008). ***ic.infer**: Inequality Constrained Inference in Linear Normal Situations*. R package version 1.0-6, URL `http://CRAN.R-project.org/package=ic.infer`.

Haskell KH, Hanson RJ (1981). "An Algorithm for Linear Least-Squares Problems with Equality and Non-Negativity Constraints." *Mathematical Programming*, **21**(1), 98–118.

Lawson CL, Hanson RJ (1995). *Solving Least Squares Problems*. 3rd edition. SIAM.

Phillips DL, Gregg JW (2003). "Source Partitioning Using Stable Isotopes: Coping with Too Many Sources." *Oecologia*, **136**(2), 261–269.

Plummer M, Best N, Cowles K, Vines K (2009). ***coda**: Output Analysis and Diagnostics for MCMC*. R package version 0.13-4, URL `http://CRAN.R-project.org/package=coda`.

Press WH, Teukolsky SA, Vetterling WT, Flannery BP (1992). *Numerical Recipes in Fortran: The Art of Scientific Computing*. Cambridge University Press.

R Development Core Team (2008). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria. ISBN 3-900051-07-0, URL `http://www.R-project.org/`.

Roberts GO (1996). "Markov Chain Concepts Related to Sampling Algorithms." In WR Gilks, S Richardson, DJ Spiegelhalter (eds.), *Markov Chain Monte Carlo in Practice*, pp. 45–58. Chapman and Hall.

Shapiro A (1988). "Towards a Unified Theory of Inequality Constrained Testing in Multivariate Analysis." *International Statistical Review*, **56**(1), 49–62.

Smith RL (1984). "Efficient Monte-Carlo Procedures for Generating Points Uniformly Distributed over Bounded Regions." *Operations Research*, **32**(6), 1296–1308.

Soetaert K, Van den Meersche K, van Oevelen D (2009). ***limSolve**: Solving Linear Inverse Models*. R package version 1.5, URL `http://CRAN.R-project.org/package=limSolve`.

Vezina AF, Platt T (1988). "Food Web Dynamics in the Ocean 1. Best-Estimates of Flow Networks Using Inverse Methods." *Marine Ecology-Progress Series*, **42**(3), 269–287.

Affiliation:

Karel Van den Meersche
Centre for Estuarine and Marine Ecology (CEME)
Netherlands Institute for Ecology (NIOO)
4401 NT Yerseke, The Netherlands
E-mail: k.vdmeersche@nioo.knaw.nl
URL: http://www.nioo.knaw.nl/users/kvdmeersche/

Journal of Statistical Software http://www.jstatsoft.org/
published by the American Statistical Association http://www.amstat.org/

Volume 30, Code Snippet 1 *Submitted:* 2008-05-26
April 2009 *Accepted:* 2009-03-31