# `xsample()`: an R Function for Sampling Linear Inverse Problems

**Karel Van den Meersche**
Universiteit Gent

**Karline Soetaert**
NIOZ Yerseke

**Dick Van Oevelen**
NIOZ Yerseke

**Abstract**

The R function `xsample()` uses Markov Chain Monte Carlo (MCMC) algorithms to uniformly sample the feasible region of constrained linear problems. It contains two hit-and-run sampling algorithms, together with a "mirror" algorithm where an MCMC step reflects on the inequality constraints.

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

## 1. Introduction

This vignette is based on a publication with the same title in Journal of Statistical Software (Van den Meersche, Soetaert, and Van Oevelen 2009). It includes parts of the introduction and the method section of that publication, omitting the examples. It may be updated and extended in the future as the package develops. For now we refer to the publication for the most up-to-date and complete documentation of the function `xsample()`.

`xsample()` is an R function that instead of optimizing a linear problem, returns a sample set that has a uniform or a truncated normal distribution bounded by a set of inequality constraints.

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

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

This kind of problems are usually overdetermined, meaning that there is no solution for which $\varepsilon = 0$. They can then be solved with quadratic programming (Lawson and Hanson 1995) techniques, in which case a norm of the error term $\varepsilon = \mathbf{Ax} - \mathbf{b}$ is minimized, for example the sum of squares $\sum \varepsilon^2$. This is a constrained linear regression problem: parameters $\mathbf{x}$ are subject to the constraints $\mathbf{Ex} = \mathbf{F}$ and $\mathbf{Gx} \geq \mathbf{h}$.

[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.

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 $\mathbf{Ax} = \mathbf{b} + \varepsilon$ with an error $\varepsilon$ 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 $\varepsilon = 0$, i.e. the model $\mathbf{Ax}$ can fit the data $\mathbf{b}$ exactly. Here, we assume that $\varepsilon$ is normally distributed, i.e. $\varepsilon \sim N(0, \mathbf{s})$.

In the absence of inequality conditions, it is straightforward to construct a series of samples $\mathbf{x}$ for which $\mathbf{Ax} - \mathbf{b} = \varepsilon$ has the proposed distribution. However, when $\mathbf{x}$ is subject to inequality constraints ($\mathbf{Gx} \geq \mathbf{h}$), $\varepsilon$ 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 $\mathbf{W} = diag(\mathbf{s}^{-1})$. 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 $\mathbf{Ax} = \mathbf{b}$. $\varepsilon$ 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 $\sigma$ 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 $\sigma$ 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 (CRAN, `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:

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

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 $\mathbf{W} = diag(\mathbf{s}^{-1})$. Otherwise, $\sigma$ 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.

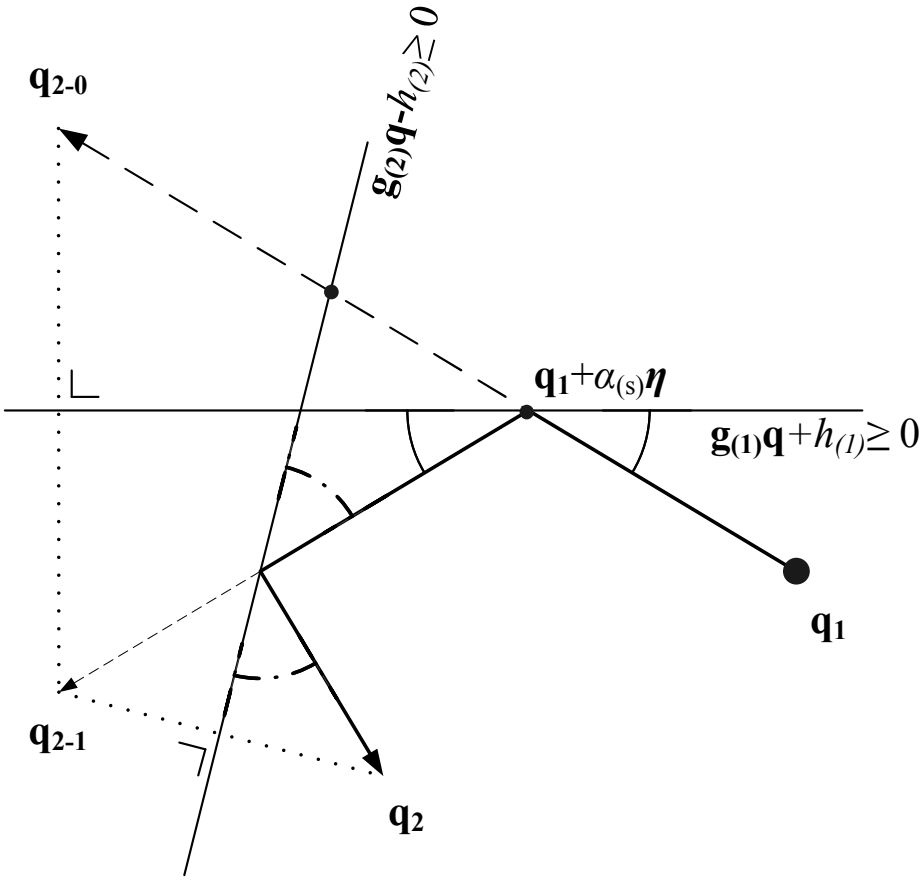


Figure 1: MCMC jump with inequality constraints functioning as mirrors. See text for explanation.

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 $\eta$ 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. 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.

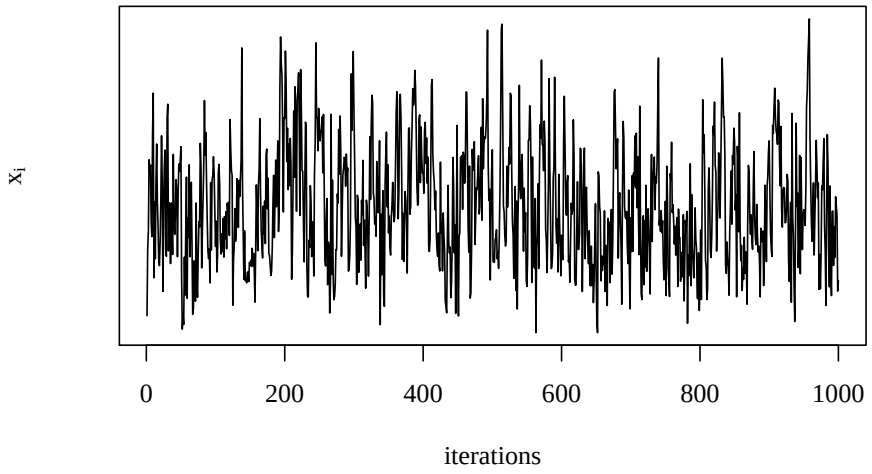


Figure 2: A good random walk of a parameter $x_i$, using `xsample()` with 1000 iterations.

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, $\eta$ 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.

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.

## References

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.

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.

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.

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 1.4, URL http://CRAN.R-project.org/package=limSolve.

Van den Meersche K, Soetaert K, Van Oevelen D (2009). "`xsample()`: An R Function for Sampling Linear Inverse Problems." *Journal of Statistical Software, Code Snippets*, **30**(1), 1–15. URL http://www.jstatsoft.org/v30/c01/.

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:

Karline Soetaert, Karel Van den Meersche, Dick van Oevelen
Royal Netherlands Institute of Sea Research (NIOZ)
4401 NT Yerseke, Netherlands E-mail: karline.soetaert@nioz.nl
URL: http://www.nioz.nl