Package **limSolve** , solving linear inverse models in R

Karline Soetaert, Karel Van den Meersche and Dick van Oevelen
Centre for Estuarine and Marine Ecology
Netherlands Institute of Ecology
The Netherlands

Abstract

R package **limSolve** (Soetaert, Van den Meersche, and van Oevelen 2008) solves linear inverse models (LIM), consisting of linear equality and or linear inequality conditions, which may be supplemented with approximate linear equations, or a target (cost, profit) function. Depending on the determinacy of these models, they can be solved by least squares or linear programming techniques, by calculating ranges of unknowns or by randomly sampling the feasible solution space.

Amongst the possible scientific applications are: food web quantification (ecology), flux balance analysis (e.g. quantification of metabolic networks, systems biology), compositional estimation (ecology, chemistry,...), and operations research problems. Package **limSolve** contains examples of these four application domains.

Keywords: Linear inverse models, food web models, flux balance analysis, linear programming, quadratic programming, R.

1. Introduction

In matrix notation, linear inverse problems are defined as: [1]

$$\mathbf{A} \cdot \mathbf{x} \simeq \mathbf{b} \qquad (1)$$
$$\mathbf{E} \cdot \mathbf{x} = \mathbf{f} \qquad (2)$$
$$\mathbf{G} \cdot \mathbf{x} \geq \mathbf{h} \qquad (3)$$

There are three sets of linear equations: equalities that have to be met as closely as possible (1), equalities that have to be met exactly (2) and inequalities (3).

Depending on the active set of equalities (2) and constraints (3), the system may either be underdetermined, even determined, or overdetermined. Solving these problems requires different mathematical techniques.

2. Even determined systems

An even determined problem has as many (independent and consistent) equations as unknowns. There is only one solution that satisfies the equations exactly.

[1] notations: vectors and matrices are in **bold**; scalars in normal font. Vectors are indicated with a small letter; matrices with capital letter.

Even determined systems that do not comprise inequalities, can be solved with R function `solve`, or -more generally- with **limSolve** function `Solve`. The latter is based on the Moore-Penrose generalised inverse method, and can solve any linear system of equations. In case the model is even determined, and if $\mathbf{E}$ is square and positive definite, `Solve` returns the same solution as function `solve`. The function uses function `ginv` from package **MASS** (Venables and Ripley 2002).

Consider the following set of linear equations:

$$\begin{array}{llll} 3 \cdot x_1 & +2 \cdot x_2 & +x_3 & = & 2 \\ x_1 & & & = & 1 \\ 2 \cdot x_1 & & +2 \cdot x_3 & = & 8 \end{array}$$

which, in matrix notation is:

$$\begin{bmatrix} 3 & 2 & 1 \\ 1 & 0 & 0 \\ 2 & 0 & 2 \end{bmatrix} \cdot \mathbf{X} = \begin{bmatrix} 2 \\ 1 \\ 8 \end{bmatrix}$$

where $\mathbf{X} = [x_1, x_2, x_3]^T$.

In R we write:

```
> E <- matrix(nrow=3,ncol=3,
+              data=c(3,1,2,2,0,0,1,0,2))
> F <- c(2,1,8)
> solve(E,F)

[1]  1 -2  3

> Solve(E,F)

[1]  1 -2  3
```

In the next example, an additional equation, which is a linear comination of the first two is added to the model (i.e. $eq_4 = eq_1 + eq_2$). This problem is therefore equivalent to the previous one; it is even determined although it contains 4 equations and only 3 unknowns.

As the input matrix is not square, this model can only be solved with function `Solve`

$$\begin{bmatrix} 3 & 2 & 1 \\ 1 & 0 & 0 \\ 2 & 0 & 2 \\ 4 & 2 & 1 \end{bmatrix} \cdot \mathbf{X} = \begin{bmatrix} 2 \\ 1 \\ 8 \\ 3 \end{bmatrix}$$

```
> E2 <- rbind(E,c(4,2,1))
> F2 <- c(F,3)
> #solve(E2,F2) # error
> Solve(E2,F2)
```

```
[1]  1 -2  3
```

3. Overdetermined systems

Overdetermined linear systems contain more independent equations than unknowns. In this case, there is only one solution in the least squares sense, i.e. a solution that satisfies:
$\min_x \|\mathbf{A} \cdot \mathbf{x} - \mathbf{b}\|^2$.

The least squares solution can be singled out by function *lsei* (*l*east *s*quares with *e*qualities and *i*nequalities). This function also returns the parameter covariance matrix, which gives indication on the confidence interval and relations among the estimated unknowns.

If there are no inequalities, then the least squares solution can also be estimated with `Solve`.

The following problem:

$$\begin{bmatrix} 3 & 2 & 1 \\ 1 & 0 & 0 \\ 2 & 0 & 2 \\ 0 & 1 & 0 \end{bmatrix} \cdot \mathbf{X} = \begin{bmatrix} 2 \\ 1 \\ 8 \\ 3 \end{bmatrix}$$

is solved in R as follows:

```
> A <- matrix(nrow=4,ncol=3,
+             data=c(3,1,2,0,2,0,0,1,1,0,2,0))
> B <- c(2,1,8,3)
> lsei(A=A,B=B)

$X
[1] -1.1621622  0.8378378  4.8918919

$residualNorm
[1] 0

$solutionNorm
[1] 10.81081

$IsError
[1] FALSE

$type
[1] "lsei"
```

Here the *residualNorm* is the sum of absolute values of the residuals of the equalities that have to be met exactly and of the violated inequalities. As in this case, there are none of those, this quantity is 0. The *solutionNorm* is the value of the minimised quadratic function at the solution, i.e. the value of $\|\mathbf{A} \cdot \mathbf{x} - \mathbf{b}\|^2$.

Alternatively, the problem can be solved by `Solve`:

```
> Solve(A,B)
```

```
[1] -1.1621622  0.8378378  4.8918919
```

If, in addition to the equalities, there are also inequalities, then `lsei` is the only method to find the least squares solution.

With the following inequalities:

$$\begin{aligned} x_1 - 2 \cdot x_2 &< 3 \\ x_1 - x_3 &> 2 \end{aligned}$$

the R-code becomes:

```
> G <- matrix(nrow=2, ncol=3, byrow=TRUE,
+             data=c(-1,2,0,1,0,-1))
> H <- c(-3,2)
> lsei(A=A,B=B,G=G,H=H)

$X
[1]  2.04142012 -0.47928994  0.04142012

$residualNorm
[1] 1.776357e-15

$solutionNorm
[1] 38.17751

$IsError
[1] FALSE

$type
[1] "lsei"
```

It is also possible to merge equalities that have to be matched exactly with equations that have to be approximated as closely as possible:

```
> E <- c(0,1,0)
> F <- 3
> lsei(E=E,F=F,A=A,B=B,G=G,H=H)

$X
[1]  1.2424242  3.0000000 -0.7575758

$residualNorm
[1] 2.220446e-15

$solutionNorm
[1] 98.0606
```

```
$IsError
[1] FALSE

$type
[1] "lsei"
```

Note that for the case above, it is also possible to perform Bayesian sampling, using **limSolve** function `xsample` (see below).

4. Undetermined systems

Underdetermined linear systems contain less independent equations than unknowns. If the equations are consistent, there exist an infinite amount of solutions. To solve such models, there are several options:

- `ldei` - finds the "*l*east *d*istance" (or parsimonious) solution, i.e. the one where the sum of squared unknowns is minimal
- `lsei`- minimises some other set of linear functions ($\mathbf{A} \cdot \mathbf{x} \simeq \mathbf{b}$) in a least squares sense.
- `linp` - finds the solution where **one** linear function (i.e. the sum of flows) is either minimized (a "cost" function) or maximized (a "profit" function). Uses linear programming.
- `xranges` - finds the possible ranges ([min,max]) for each unknown.
- `xsample` - randomly samples the solution space in a Bayesian way. This method returns the conditional probability density function for each unknown.

4.1. Equalities only

We start with an example including only equalities.

$$\begin{array}{llllll} 3 \cdot x_1 & +2 \cdot x_2 & +x_3 & = & 2 \\ x_1 & & & = & 1 \end{array}$$

Functions `Solve` and `ldei` retrieve the parsimonious solution.

It is slightly more complex to select the parsimonious solution using `lsei`. Here the approximate equations ($\mathbf{A}$, the identity matrix, and $\mathbf{b}$) have to be specified.

```
> E <- matrix(nrow=2, ncol=3,
+             data=c(3,1,2,0,1,0))
> F <- c(2,1)
> Solve(E,F)

[1]  1.0 -0.4 -0.2
```

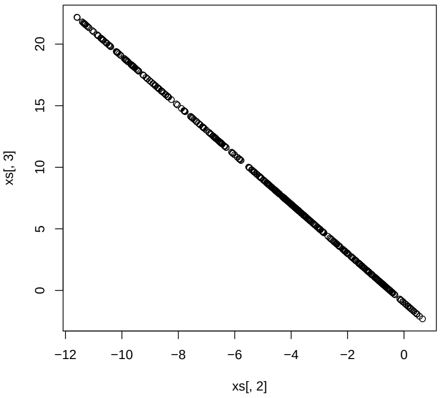

Figure 1: Random sample of the underdetermined system including only equalities. See text for explanation.

```
> ldei(E=E,F=F)$X

[1]  1.0 -0.4 -0.2

> lsei(E=E,F=F,A=diag(3),B=rep(0,3))$X

[1]  1.0 -0.4 -0.2
```

In this particular case, it is also possible to randomly sample the solution space. This demonstrates that all valid solutions of x_2 and x_3 are located on a line (figure 1).

```
> xs <- xsample(E=E,F=F,iter=500)$X
> plot(xs[,2],xs[,3])
```

4.2. Equalities and inequalities

The next example also includes inequalities:

Consider the following set of linear equations:

$$\begin{array}{lllll} 3\cdot x_1 & +2\cdot x_2 & +x_3 & +4\cdot x_4 & = & 2 \\ x_1 & +x_2 & +x_3 & +x_4 & = & 2 \end{array}$$

complemented with the inequalities:

$$\begin{array}{lllll} 2\cdot x_1 & +x_2 & +x_3 & +x_4 & \geq & -1 \\ -1\cdot x_1 & +3\cdot x_2 & +2\cdot x_3 & +x_4 & \geq & 2 \\ -1\cdot x_1 & & +x_3 & & \geq & 1 \end{array}$$

As before, the parsimonious solution can be found by functions `ldei` and `lsei`.

```
> E <- matrix(ncol=4, byrow=TRUE,
+             data=c(3,2,1,4,1,1,1,1))
> F <- c(2,2)
> G <-matrix(ncol=4,byrow=TRUE,
+            data=c(2,1,1,1,-1,3,2,1,-1,0,1,0))
> H <- c(-1, 2, 1)
> ldei(E,F,G=G,H=H)$X

[1]  0.2  0.8  1.4 -0.4

> (pars<-lsei(E,F,G=G,H=H,A=diag(nr=4),B=rep(0,4))$X)

[1]  0.2  0.8  1.4 -0.4
```

We can also estimate the minimal and maximal values of all unknowns using function `xranges`. The results are conveniently plotted using R function `dotchart` (figure 2)

```
> (xr<-xranges(E,F,G=G,H=H))

       min  max
[1,] -3.00 0.80
[2,] -6.75 7.50
[3,] -2.00 7.50
[4,] -0.50 4.25

> dotchart(pars,xlim=range(c(pars,xr)),label=paste("x",1:4,""))
> segments(x0=xr[,1],x1=xr[,2],y0=1:nrow(xr),y1=1:nrow(xr))
```

Function `xsample` randomly samples the underdetermined problem. For small problems, the coordinates directions algorithm ("cda") is a good choice.

We use R function `pairs`, with a density function on the diagonal, to visualise the output (figure 3).

```
> panel.dens <- function(x, ...)
+ {
+     usr <- par("usr"); on.exit(par(usr))
+     par(usr = c(usr[1:2], 0, 1.5) )
+     DD <- density(x); DD$y<- DD$y/max(DD$y)
+     polygon(DD,col="grey")
+ }
> xs <- xsample(E=E,F=F,G=G,H=H,type="cda")$X
> pairs(xs,pch=".",cex=2,upper.panel=NULL,diag.panel=panel.dens)
```

4.3. Equalities, inequalities and approximate equations

The following problem

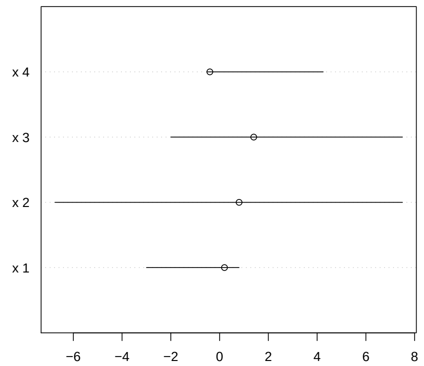

Figure 2: Parsimonious solution and ranges of the underdetermined system including equalities and inequalities. See text for explanation.

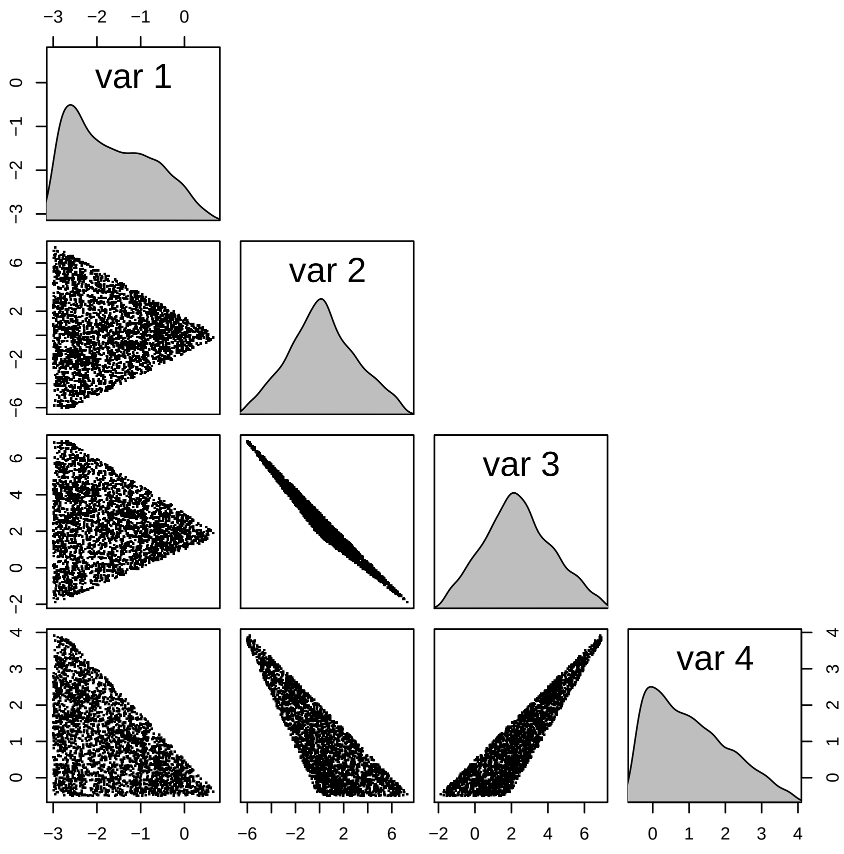

Figure 3: Random sample of the underdetermined system including equalities and inequalities. See text for explanation.

$$
\begin{array}{lllllll}
3 \cdot x_1 & +2 \cdot x_2 & +x_3 & +4 \cdot x_4 & = & 2 \\
x_1 & +x_2 & +x_3 & +x_4 & = & 2 \\
\\
2 \cdot x_1 & +x_2 & +x_3 & +x_4 & \geq & -1 \\
-1 \cdot x_1 & +3 \cdot x_2 & +2 \cdot x_3 & +x_4 & \geq & 2 \\
-1 \cdot x_1 & & +x_3 & & \geq & 1 \\
\\
2 \cdot x_1 & +2 \cdot x_2 & +x_3 & +6 \cdot x_4 & \simeq & 1 \\
x_1 & -x_2 & +x_3 & -x_4 & \simeq & 2
\end{array}
$$

is implemented and solved in R as:

```
> A <- matrix(ncol=4, byrow=TRUE,
+             data=c(2,2,1,6,1,-1,1,-1))
> B <- c(1,2)
> lsei(E,F,G=G,H=H,A=A,B=B)$X
```

```
[1]  0.3333333  0.3333333  1.6666667 -0.3333333
```

Function `xsample` randomly samples the underdetermined problem (using the metropolis algorithm), selecting likely values given the approximate equations. The probability distribution of the latter is assumed Gaussian, with given standard deviation (argument `sdB`, here assumed 1).

The jump length (argument `jmp`) is finetuned such that a sufficient number of trials ($\tilde{3}0\%$), but not too many, is accepted. Note how the ultimate distribution is determined both by the inequality constraints (the shart edges) as well as by the approximate equations (figure 4).

```
> panel.dens <- function(x, ...)
+ {
+     usr <- par("usr"); on.exit(par(usr))
+     par(usr = c(usr[1:2], 0, 1.5) )
+     DD <- density(x); DD$y<- DD$y/max(DD$y)
+     polygon(DD,col="grey")
+ }
> xs <- xsample(E=E,F=F,G=G,H=H,A=A,B=B,jmp=0.5, sdB=1)$X
> pairs(xs,pch=".",cex=2,upper.panel=NULL,diag.panel=panel.dens)
```

4.4. Equalities and inequalities and a target function

In previous section, the parsimonious solution, i.e. the solution for which the sum of squared values is minimal was selected. Another way to single out one solution out of the infinite amount of valid solutions is by minimising or maximising a linear target function, using linear programming. For instance,

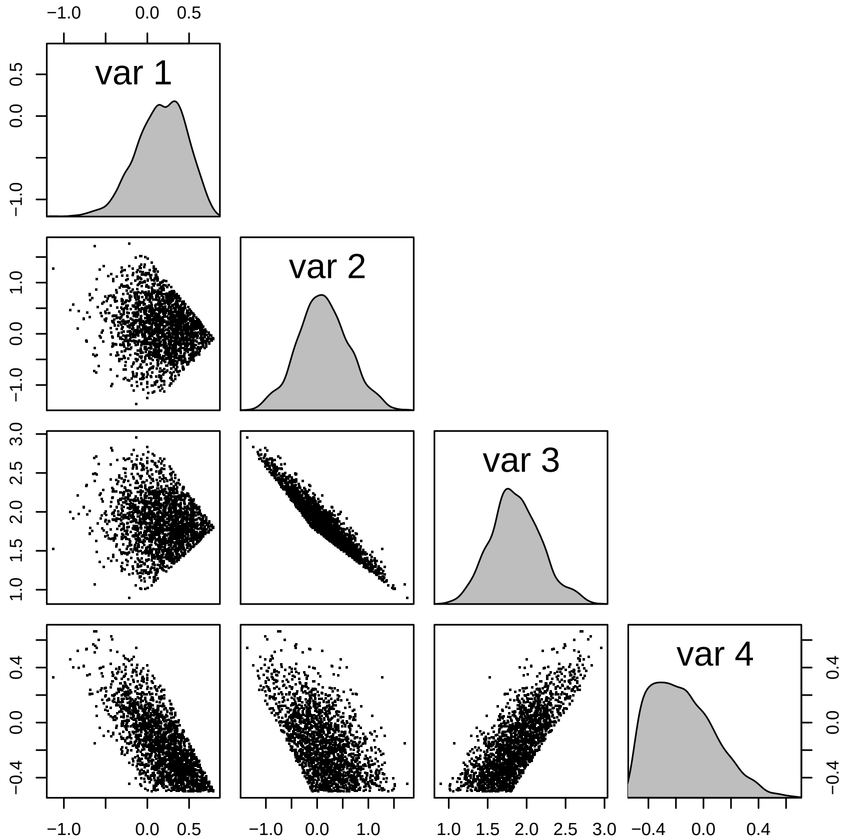

Figure 4: Random sample of the underdetermined system including equalities, inequalities, and approximate equations. See text for explanation.

$$\min(x_1 + 2 \cdot x_2 - 1 \cdot x_3 + 4 \cdot x_4)$$

subject to :

$$\begin{array}{lllll}
3 \cdot x_1 & +2 \cdot x_2 & +x_3 & +4 \cdot x_4 & = & 2 \\
x_1 & +x_2 & +x_3 & +x_4 & = & 2 \\
 & & & & & \\
2 \cdot x_1 & +x_2 & +x_3 & +x_4 & \geq & -1 \\
-1 \cdot x_1 & +3 \cdot x_2 & +2 \cdot x_3 & +x_4 & \geq & 2 \\
-1 \cdot x_1 & & +x_3 & & \geq & 1 \\
 & & & & & \\
x_i & \geq & 0 & \forall i & &
\end{array}$$

is implemented in R as:

```
> E <- matrix(ncol=4, byrow=TRUE,
+             data=c(3,2,1,4,1,1,1,1))
> F <- c(2,2)
> G <-matrix(ncol=4,byrow=TRUE,
+            data=c(2,1,1,1,-1,3,2,1,-1,0,1,0))
> H <- c(-1, 2, 1)
> Cost <- c(1,2,-1,4)
> linp(E,F,G,H,Cost)

$X
[1] 0 0 2 0

$residualNorm
[1] 0

$solutionNorm
[1] -2

$IsError
[1] FALSE

$type
[1] "simplex"
```

The positivity requirement ($x_i \geq 0$) is -by default- part of the linear programming problem.

In case the unknowns need not be positive, they are rewritten as the difference of two positive unknowns as follows:

$$x_i = {x_i}^a - {x_i}^b$$

where both ${x_i}^a$ and ${x_i}^b$ are positive quantities, but where x_i can be negative (if ${x_i}^a < {x_i}^b$. The linear programming problem is then rewritten as a function of these new unknowns as follows:

```
> EE <- cbind(E,-E)
> GG <- cbind(G,-G)
> CCost <- c(Cost, -Cost)
> lp <-linp(E=EE,F=F,G=GG, H= H, Cost=CCost)
> X <- lp$X[1:4]-lp$X[5:8]
> X
```

```
[1] -3.00 -6.75  7.50  4.25
```

```
> lp$solutionNorm
```

```
[1] -7
```

5. datasets

There are five example applications in package **limSolve** :

- `Blending`. In this -underdetermined- problem, an optimal composition of a feeding mix is sought such that production costs are minimised subject to minimal nutrient constraints. The problem consists of one equality and 4 inequality conditions, and a cost function. It is solved by linear programming (`linp`). Feasible ranges are estimated (`xranges`) and feasible solutions generated (`xsample`)

- `Chemtax`. This is an overdetermined linear inverse problem, where the algal composition of a (field) sample is estimated based on (experimentally-determined) pigment biomarkers. Biologists may know this problem as "Chemtax" (Mackey, Mackey, Higgins, and Wright 1996). See also R -package **BCE** (Van den Meersche, Soetaert, and Middelburg 2008), (Van den Meersche and Soetaert 2008). The problem contains 8 unknowns; it consists of 1 equality, 12 approximate equations, and 8 inequalities. It is solved using `lsei` and `xsample`.

- `Minkdiet`. This is another -underdetermined- compositional estimation problem, where the diet composition of Southeast Alaskan Mink is estimated, based on the C and N isotopic ratios of Mink and of its prey items (Ben-David, Hanley, Klein, and Schell 1997). The problem consists of 7 unknowns, 3 equations, and 7 inequalities

- `RigaWeb`. This is a food web problem, where food web flows of the Gulf of Riga planktonic food web in Spring are quantified (Donali, Olli, Heiskanen, and Andersen 1999). This -underdetermined- model comprises 26 unknowns, 14 equalities, and 45 inequalities. It is solved by `lsei`, `xranges`, and `xsample`.

- `E_coli`. This is a flux balance problem, estimating the core metabolic fluxes of Escherichia coli (Edwards, Covert, and Palsson 2002). It is the largest example included in **limSolve** . There are 70 unknowns, 54 equalities and 62 inequalities, and one function to maximise. This model is solved using `lsei`, `linp`, `xranges`, and `xsample`.

6. Notes

Package **limSolve** provides FORTRAN implementations of:

- the least distance algorithms from (Lawson and Hanson 1974) (`ldp`, `ldei`, `nnls`).
- the least squares with equality and inequality algorithms from (Haskell and Hanson 1978) (`lsei`).
- a solver for banded linear systems from LINPACK ((Dongarra, Bunch, Moler, and Stewart 1979)).
- a solver for tridiagonal linear systems (own implementation).

In addition, the package provides a wrapper around functions:

- function `lp` from package **lpSolve** (Berkelaar *et al.* 2007)
- function `solve.QP` from package **quadprog** (Weingessel 2007)

This way, similar input can be used to solve least distance, least squares and linear programming problems. Note that the packages **lpSolve** and **quadprog** provide more options than used here.

For solving linear programming, `lp` from **lpSolve** is the only algorithm included. (It is the most robust linear programming algorithm we are familiar with). For quadratic programming, we added the code `lsei`, which in our experience often gives a valid solution whereas other functions (including `solve.QP`) may fail.

Finalisation of this package was done using R-Forge (`http://r-forge.r-project.org/`), the framework for R project developers, based on GForge and tortoiseSVN (`http://tortoisesvn.net/`) for (sub) version control.

This vignette was created using Sweave (Leisch 2002).

The package is on CRAN, the R-archive website ((R Development Core Team 2008))

More examples can be found in the demo of package **limSolve** ("demo(limSolve"))

Another R -package, **LIM** ((Soetaert and van Oevelen 2008)) [2] is designed for reading and solving linear inverse models (LIM). The model problem is formulated in text files in a way that is natural and comprehensible. **LIM** then converts this input into the required linear equality and inequality conditions, which can be solved by the functions in package **limSolve** .

The various functions in **limSolve** are given in table (1).

[2] at this time, this package may not yet be on CRAN; it may be found on R-forge

Table 1: Summary of the functions in package **limSolve**

Function	Description
Solve	Finds the generalised inverse solution of $A \cdot x = b$
Solve.banded	Solves a banded system of linear equations
Solve.tridiag	Solves a tridiagonal system of linear equations
ldei	Least distance programming with equality and inequality conditions
ldp	Least distance programming (only inequality conditions)
linp	Linear programming
lsei	Least squares with equality and inequality conditions
nnls	Nonnegative least squares
resolution	Row and column resolution of a matrix
xranges	Calculates ranges of unknowns
varranges	Calculates ranges of variables (linear combinations of unknonws)
xsample	Randomly samples a linear inverse problem for the unknowns
varsample	Randomly samples a linear inverse problem for the inverse variables

References

Ben-David M, Hanley T, Klein D, Schell D (1997). "Seasonal changes in diets of coastal and riverine mink: the role of spawning Pacific salmon." *Canadian Journal of Zoology*, **75**, 803–811.

Berkelaar M, *et al.* (2007). *lpSolve: Interface to Lp_solve v. 5.5 to solve linear or integer programs*. R package version 5.5.8.

Donali E, Olli K, Heiskanen AS, Andersen T (1999). "Carbon flow patterns in the planktonic food web of the Gulf of Riga, the Baltic Sea: a reconstruction by the inverse method." *Journal of Marine Systems*, **23**, 251–268.

Dongarra J, Bunch J, Moler C, Stewart G (1979). *LINPACK Users Guide*. SIAM.

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

Haskell KH, Hanson RJ (1978). *An algorithm for linear least squares problems with equality and nonnegativity constraints*. Report SAND77-0552.

Lawson C, Hanson R (1974). *Solving Least Squares Problems*. Prentice-Hall.

Leisch F (2002). "Sweave: Dynamic Generation of Statistical Reports Using Literate Data Analysis." In W Härdle, B Rönz (eds.), "Compstat 2002 — Proceedings in Computational Statistics," pp. 575–580. Physica Verlag, Heidelberg. ISBN 3-7908-1517-9, URL `http://www.stat.uni-muenchen.de/~leisch/Sweave`.

Mackey M, Mackey D, Higgins H, Wright S (1996). "CHEMTAX - A program for estimating class abundances from chemical markers: Application to HPLC measurements of phytoplankton." *Marine Ecology-Progress Series*, **144**, 265–283.

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`.

Soetaert K, Van den Meersche K, van Oevelen D (2008). *limSolve: Solving linear inverse models*. R package version 1.2.

Soetaert K, van Oevelen D (2008). *LIM: Linear Inverse Modelling*. R package version 1.1.

Van den Meersche K, Soetaert K (2008). *BCE: Bayesian composition estimator: estimating sample (taxonomic) composition from biomarker data*. R package version 1.1.

Van den Meersche K, Soetaert K, Middelburg J (2008). "A Bayesian compositional estimator for microbial taxonomy based on biomarkers." *Limnology and Oceanography Methods*, pp. 190–199.

Venables WN, Ripley BD (2002). *Modern Applied Statistics with S*. Springer, New York, fourth edition. ISBN 0-387-95457-0, URL `http://www.stats.ox.ac.uk/pub/MASS4`.

Weingessel A (2007). *quadprog: Functions to solve Quadratic Programming Problems.S original by Berwin A. Turlach R port by Andreas Weingessel*. R package version 1.4-11.

Affiliation:

Karline Soetaert, Karel Van den Meersche, Dick van Oevelen
Centre for Estuarine and Marine Ecology (CEME)
Netherlands Institute of Ecology (NIOO)
4401 NT Yerseke, Netherlands E-mail: `k.soetaert@nioo.knaw.nl`
URL: `http://www.nioo.knaw.nl/ppages/ksoetaert`