extended R documentation

of function `sorcering()`

from package `'sorcering'`

July 14, 2021

Version	0.9.2
Title	SORCERING: Soil ORganic Carbon & CN Ratio drIven Nitrogen modellinG framework
Authors	Marc Scherstjanoi* & René Dechow◇ * Thünen Institute of Forest Ecosystems, Eberswalde, Germany ◇ Thünen Institute of Climate-Smart Agriculture, Braunschweig, Germany
Maintainer	Marc Scherstjanoi <marc.scherstjanoi@thuenen.de>
LazyData	true
Depends	R (>= 3.5.0)
License	GLP (>= 2)
Imports	Rcpp (>= 1.0.6)
LinkingTo	Rcpp, RcppArmadillo

Description

`SORCERING` can be used to model the fate of soil organic carbon (SOC) and soil organic nitrogen (SON) and to calculate N mineralisation rates. It provides a framework that numerically solves differential equations of SOC models based on first-order kinetics. Thus, SOC models can be simply defined and run to predict the temporal development of SOC. Beyond this, `SORCERING` determines the fluxes of SON and N mineralisation / immobilisation. Basic inputs are (1) the model parameters of a given SOC model expressed as the C transfer matrix (including information on decomposition and transfer rates between model pools), (2) the initial distributions of C and N among model pools and (3) time series of C and N inputs and rate modulating environmental factors. The fourth-order Runge-Kutta algorithm is used to numerically solve the system of differential equations.

Usage

```
sorcering(      A = NULL,
                t_sim = 2,
                tsteps = "monthly",
                C0 = NULL,
                N0 = NULL,
                Cin = NULL,
                Nin = NULL,
                xi = NULL,
                A = NULL,
                calcN = FALSE,
                calcNbalance = FALSE)
```

Arguments

`A`	transfer matrix. Defines number of pools, decomposition and transfer rates. Must be a square matrix. n $\times$ n elements with n = number of pools. Diagonal values are decomposition rates [yr^{-1}]. Off-diagonals represent the transfer between pools [yr^{-1}].
`t_sim`	number of simulation time steps. Must correspond to the number of rows of `Cin`, `Nin` and `xi`.
`tsteps`	character indicating the type of simulation time steps. valid options are `annually`, `monthly` (recommended) or `weekly`.
`C0`	vector with a length equal to the number of pools. Contains initial soil organic carbon per pool [tC ha^{-1}]. If `NULL`, filled with zeros.
`N0`	vector with a length equal to the number of pools. Contains initial soil organic nitrogen per pool [tN ha^{-1}]. If `NULL`, filled with zeros. Only used when `calcN = TRUE`
`Cin`	matrix with a number of columns equal to the number of pools and a number of rows corresponding to `t_sim`. Contains carbon input per pool and time step [tC ha^{-1}]. If `NULL`, filled with zeros.
`Nin`	matrix with a number of columns equal to the number of pools and a number of rows corresponding to `t_sim`. Contains nitrogen input per pool and time step [tN ha^{-1}]. If `NULL`, filled with zeros. Only used when `calcN = TRUE` Must be `>0` where `Cin>0`.
`xi`	matrix with a number of columns equal to the number of pools and a number of rows corresponding to `t_sim`. Contains environmental factors. If `NULL`, filled with ones.
`calcN`	logical indicating whether the development of soil organic nitrogen should be simulated.
`calcNbalance`	logical indicating whether the balance of nitrogen cycling should be calculated.

Value

`sorcering()` returns a list object with components:

`C`	array with a number of rows corresponding to `t_sim` and a number of columns equal to the number of pools. Contains soil organic carbon [tC ha^{-1}].
`N`	array with a number of rows corresponding to `t_sim` and a number of columns equal to the number of pools. Contains soil organic nitrogen [tN ha^{-1}]. Only generated if `calcN = TRUE`
`Nloss`	array with a number of rows corresponding to `t_sim` and a number of columns equal to the number of pools. Contains nitrogen losses [tN ha^{-1}]. Only generated if `calcN = TRUE`.
`Nmin`	array with a number of rows corresponding to `t_sim` and a number of columns equal to the number of pools. Contains nitrogen mineralisation [tN ha^{-1}]. If values are negative, nitrogen immobilisation exceeds mineralisation. Only generated if `calcN = TRUE`.
`Nmin.sink.1, ..., Nmin.sink.n`	arrays with a number of rows corresponding to `t_sim` and a number of columns equal to the number of pools `n`. Contain pool-specific nitrogen mineralisation sinks [tN ha^{-1}] (from the pool according to variable index [1, ..., n] to the pool according to column number). If the sink is the pool itself (index equals column number) the amount of decomposition is recorded. Only generated if `calcN = TRUE`.
`Nbalance`	array with a number of rows corresponding to `t_sim` and three columns. Contains information on overall N changes in the soil between two time steps (first column) and information on total system N balance calculated based on total `Nloss` (second column) and based on total `Nmin` (third column) [tN ha^{-1}]. Only generated if `calcN = TRUE` and `calcNbalance = TRUE`.

Details

`SORCERING` is a general model framework to describe soil organic carbon (SOC) dynamics and soil organic nitrogen (SON) dynamics based on models of first-order kinetics. It can be applied to any given SOC first-order kinetics model. The approach has already been successfully tested to describe SOC dynamics of Yasso (Tuomi et al. 2009), RothC (Coleman and Jenkinson 1996) and C-Tool (Taghizadeh-Toosi et al. 2014). Therefore, `SORCERING` is a lightweight alternative to the widely developed and multifunctional R package SoilR (Sierra et al. 2012, Sierra and Mueller 2014). Moreover, it additionally offers the possibility of modelling N immobilisation and mineralisation by enhancing given SOC models by an additional N module.

The following is a description of each element calculated, which also corresponds to the output values (see section Value).

C

`SORCERING` calculates SOC applying a given SOC model for every simulation time step defined by `tsteps` and `t_sim`. SOC models applied here are defined by a number of pools, each characterised by specific decomposition and turnover rates. The underlying equation of first-order kinetics defines the change of SOC concentration in time as:

$$\frac{dC(t)}{dt} = Cin(t) + A_e(t) \cdot C(t) \tag{1}$$

The boundary condition $Cin(t)$ must be defined beforehand. $A_e(t)$ is composed of transfer matrix A and the model-specific generated environmental factor $xi(t)$:

$$\begin{aligned} A_e(t) &= \left(A^T \cdot xi(t)\right)^T \\ &= A \cdot diag(xi(t)) \end{aligned} \tag{2}$$

Eq. 1 is valid for scalar values of A, C, xi and Cin, as well as for a square matrix A with side length of number of SOC pools n and related one dimensional vectors C, xi and Cin with length n. Each element of C, xi and Cin and each row and column of A thus stands for a specific pool. Off-diagonal elements of A describe SOC fluxes and diagonal elements describe SOC decomposition. Analytical solutions of eq. 1 are exponential functions and can be very complex with A containing more off-diagonals, i.e. more types of SOC transfer among pools. Therefore, numerical solutions are an efficient way to solve the resulting complex equation system. In `SORCERING`, this equation system is solved by applying the fourth-order Runge-Kutta method:

$$C(t) = C(t-1) + \frac{1}{6}(K_1 + 2K_2 + 2K_3 + K_4) \tag{3}$$

with

$$\begin{aligned} K_1 =& Cin(t-1) + A \cdot diag(xi(t-1)) \cdot C(t-1) \\ K_2 =& Cin(t-1) + A \cdot diag\left(\frac{xi(t-1)+xi(t)}{2}\right) \cdot \left(C(t-1) + \frac{K_1}{2}\right) \\ K_3 =& Cin(t-1) + A \cdot diag\left(\frac{xi(t-1)+xi(t)}{2}\right) \cdot \left(C(t-1) + \frac{K_2}{2}\right) \\ K_4 =& Cin(t-1) + A \cdot diag(xi(t)) \cdot (C(t-1) + K_3) \end{aligned} \tag{4}$$

Initial conditions must be defined for every SOC pool by `C0`. For more information on the functioning of and other possibilities for solving first-order kinetics SOC models see Sierra et al. (2012).

N

As an extension to SOC modelling, `SORCERING` allows the modelling of SON coupled to the modelling of SOC. Its implementation is based on the following simplifying assumptions: (1) Nitrogen transfer and turnover rates are equal to carbon rates. (2) There is no N limitation in the soil, i.e. mineral N is always available for N immobilisation processes. (3) CN ratios of single pools are only affected by external inputs of N and C. The transfer of organic matter among pools does not affect CN ratios. As for SOC, the development of SON depends on initial and boundary conditions: `N0` and `Nin`.

Given the amount of SOC decomposed

$$C_{decomp} = C(t-1) + Cin - C(t) \tag{5}$$

and the amount of SON decomposed

$$N_{decomp} = N(t-1) + Nin - N(t) \tag{6}$$

between time points t and $t-1$, and assuming proportional C and N decomposition rates

$$\frac{C_{decomp}}{C(t)} = \frac{N_{decomp}}{N(t)} \tag{7}$$

the amount of SON at each simulation time step is calculated as:

$$N(t) = \frac{N(t-1) + Nin}{\left(\frac{C_{decomp}}{C(t)} + 1\right)} \tag{8}$$

`Nloss, Nmin, Nmin.sink⟨1⟩, ..., Nmin.sink⟨n⟩`

Along with modelling SON, further quantities are determined. Nitrogen losses are calculated as:

$$Nloss(t) = N(t-1) + Nin(t-1) - N(t) \tag{9}$$

In contrast, mineralisation rates contain information about sources and sinks of SON. Pool-specific N mineralisation $Nmin.sink\,\langle j \rangle$ and N mineralisation *Nmin* are related the following:

$$Nmin_j(t) = \sum_{p=1}^{n} Nmin.sink\,\langle j \rangle_p\,(t) \tag{10}$$

for each simulation time point t, each pool $j = 1, ..., n$ and each pool $p = 1, ..., n$ and n total pools. Or in other words, the row sum of $Nmin.sink\,\langle j \rangle$ at one simulation time point equals the j[th] column of *Nmin* at that time point. Mineralisation rates and sinks are read from a mineralisation rates matrix *Nmin.mat*:

$$Nmin_1(t), ..., Nmin_n(t) = \sum_{i=1}^{n} Nmin.mat_{i,1}(t), ..., \sum_{i=1}^{n} Nmin.mat_{i,n}(t) \tag{11}$$

$$Nmin.sink\,\langle j \rangle_1\,(t), ..., Nmin.sink\,\langle j \rangle_n\,(t) = Nmin.mat_{j,1}(t), ..., Nmin.mat_{j,n}(t) \tag{12}$$

and using the fourth-order Runge-Kutta Method *Nmin.mat* at time point t is calculated as:

$$Nmin.mat(t) = -\frac{1}{6}(Kn_1 + 2Kn_2 + 2Kn_3 + Kn_4) \tag{13}$$

with

$$\begin{aligned}
Kn_1 &= (A \cdot diag(xi(t-1) \cdot C(t-1)))^T \cdot diag\left(\frac{1}{CN(t)}\right) \\
Kn_2 &= \left(A \cdot diag\left(\frac{xi(t-1) + xi(t)}{2} \cdot \left(C(t-1) + \frac{K_1}{2}\right)\right)\right)^T \cdot diag\left(\frac{1}{CN(t)}\right) \\
Kn_3 &= \left(A \cdot diag\left(\frac{xi(t-1) + xi(t)}{2} \cdot \left(C(t-1) + \frac{K_2}{2}\right)\right)\right)^T \cdot diag\left(\frac{1}{CN(t)}\right) \\
Kn_4 &= (A \cdot diag(xi(t) \cdot (C(t-1) + K_3)))^T \cdot diag\left(\frac{1}{CN(t)}\right)
\end{aligned} \tag{14}$$

and

$$CN(t) = \begin{cases} \frac{C(t)}{N(t)}, & \forall n \in N(t) > 0 \\ n.c., & \text{otherwise} \end{cases} \tag{15}$$

and K_1 - K_3 taken from eq. system (4). Note that $Kn_{1..4}$ are matrices and $K_{1..3}$ are vectors.

As changes in SON must match the sums of all mineralisation paths, the sums over soil pools of `Nloss` and `Nmin`, respectively, must be approximately equal for all simulation time points:

$$\sum_{p=1}^{n} Nloss_p(t) \approx \sum_{p=1}^{n} Nmin_p(t) \quad \forall t \in tseq \tag{16}$$

A verification of this relation is given by `Nbalance` (see below).

Nbalance

The overall N change between two time steps is calculated as:

$$\Delta N(t) = \sum_{p=1}^{n} N_p(t-1) - \sum_{p=1}^{n} N_p(t) \tag{17}$$

The total system N balance serves as a verification output. Both of the following equations should give results close to zero:

$$N_{bal1}(t) = \sum_{p=1}^{n} Nin_p(t-1) + \Delta N(t) - \sum_{p=1}^{n} Nloss_p(t) \approx 0 \tag{18}$$

$$N_{bal2}(t) = \sum_{p=1}^{n} Nin_p(t-1) + \Delta N(t) - \sum_{p=1}^{n} Nmin_p(t) \approx 0 \tag{19}$$

Package Building Information

The SORCERING code was written in C++ using the R packages Rcpp (Eddelbuettel et al. 2021a) and RcppArmadillo (Eddelbuettel et al. 2021b).

Example

```
#EXAMPLE OF RothC application with fictional input

#1. Input

data(Cin_ex, Nin_ex, N0_ex, C0_ex, xi_ex) #fictional data
A_RothC <- fget_A_RothC(clay=30) #create transfer matrix for RothC

#2. simulation

out <- sorcering(A=A_RothC, t_sim=60, Cin=Cin_ex, Nin=Nin_ex,
N0=N0_ex, C0=C0_ex, xi=xi_ex, calcN=TRUE, tsteps="monthly")

#3. results

#output structure summary
summary(out)

#sample plot
oldpar <- par(no.readonly = TRUE) #save old par
par(mfrow=c(1,1),mar=c(4,4,1,4))
plot(rowSums(out$N),axes=FALSE, col=1, cex.lab=2,xlab="",ylab="",ylim=c(0,9),pch=20)
par(new=TRUE)
plot(rowSums(Cin_ex)/rowSums(Nin_ex),
  axes=FALSE,col=2, cex.lab=2,xlab="",ylab="",ylim=c(0,60),pch=20)
axis(side=2, pos = 0,
  labels = (0:6) 1.5, at=(0:6) 10, hadj=1, padj = 0.5, cex.axis=2,las=1,col.axis=1)
axis(side=4, pos = 60,
  labels = (0:6) 10, at=(0:6) 10, hadj=0, padj = 0.5, cex.axis=2, las=1,col.axis=2)
axis(side=1, pos = 0,
  labels =  (0:6) 10 , at=(0:6) 10, hadj=0.5, padj = 0, cex.axis=2)
title(ylab="total N", line=2, cex.lab=2)
title(ylab="C input / N input", line=-30, cex.lab=2,col.lab=2)
title(xlab="time", line= 2, cex.lab=2)
par(oldpar) #back to old par
```

References

Coleman, K., Jenkinson, D.S., 1996. Rothc-26.3 - a model for the turnover of carbon in soil, in: Powlson, D.S., Smith, P., Smith, J.U. (Eds.), Evaluation of Soil Organic Matter Models, Springer Berlin Heidelberg, Berlin, Heidelberg. pp. 237–246.

Eddelbuettel, D., Francois, R., Allaire, J., Ushey, K., Kou, Q., Russell, N., Bates, D., Chambers, J., 2021a. Rcpp: Seamless R and C++ Integration. R package version 1.0.6, https://cran.r-project.org/web/packages/Rcpp/index.html.

Eddelbuettel, D., Francois, R., Bates, D., Ni, B., 2021b. RcppArmadillo: 'Rcpp' Integration for the 'Armadillo' Templated Linear Algebra Library. R package version 0.10.4.0.0, https://cran.r-project.org/web/packages/RcppArmadillo/index.html.

Sierra, C.A., Mueller, M., 2014. SoilR: Models of Soil Organic Matter Decomposition. R package version 1.1-23, https://cran.r-project.org/web/packages/mathjaxr/index.html.

Sierra, C.A., Müller, M., Trumbore, S.E., 2012. Models of soil organic matter decomposition: the SoilR package, version 1.0. Geoscientific Model Development 5, 1045–1060.

Taghizadeh-Toosi, A., Christensen, B.T., Hutchings, N.J., Vejlin, J., Kätterer, T., Glendining, M., Olesen, J.E., 2014. C-TOOL: A simple model for simulating whole-profile carbon storage in temperate agricultural soils. Ecological Modelling 292, 11 – 25.

Tuomi, M., Thum, T., Järvinen, H., Fronzek, S., Berg, B., Harmon, M., Trofymow, J., Sevanto, S., Liski, J., 2009. Leaf litter decomposition—Estimates of global variability based on Yasso07 model. Ecological Modelling 220, 3362 – 3371.