Package 'sorcering'

October 13, 2025

Type Package

Title Soil Organic Carbon and CN Ratio Driven Nitrogen Modelling Framework

Version 1.2.1

Date 2025-10-13

Author Marc Scherstjanoi [aut, cre], Rene Dechow [aut]

Maintainer Marc Scherstjanoi <marc.scherstjanoi@thuenen.de>

Description Can be used to model the fate of soil organic carbon and soil organic nitrogen and to calculate N mineralisation rates. Provides a framework that numerically solves differential equations of soil organic carbon models based on first-order kinetics and extends these models to include the nitrogen component. The name 'sorcering' is an acronym for 'Soil ORganic Carbon & CN Ratio drIven Nitrogen modellinG framework'.

LazyData true

Depends R (>= 3.5.0)

License GPL (>= 2)

Imports Rcpp (>= 1.0.6), mathjaxr, Rdpack

LinkingTo Rcpp, RcppArmadillo

RdMacros mathjaxr, Rdpack

BuildManual TRUE

Contents

`fget_A_RothC` *RothC Transfer Matrix Building Function*

Description

Builds a RothC transfer matrix. Parameters taken from Coleman and Jenkinson (1996).

Usage

```
fget_A_RothC(    clay = 23.4
)
```

Arguments

`clay`	double. Soil clay content in %.

Value

`fget_A_RothC()` returns a 5×5 matrix that contains RothC specific carbon transfer parameters based on clay content.

Author(s)

Marc Scherstjanoi <marc.scherstjanoi@thuenen.de>, Rene Dechow

References

Coleman K, Jenkinson DS (1996). "RothC-26.3 - A Model for the turnover of carbon in soil." In Powlson DS, Smith P, Smith JU (eds.), *Evaluation of Soil Organic Matter Models*, 237–246. ISBN 978-3-642-61094-3.

See Also

`sorcering`.

Examples

```
fget_A_RothC(clay=30)
```

meas_data_ex *Measured Data Input*

Description

Fictional input data. Contains a matrix of three columns. Column 1: Measuremnt time. Column 2: SOC [t/ha]. Column 3: SON [t/ha].

Usage

```
meas_data_ex
```

Format

A matrix containing 3 columns

See Also

sorcering.

RothC_C0_ex *Initial Soil Organic Carbon Data for RothC*

Description

Fictional initial soil organic carbon for the RothC SOC model

Usage

```
RothC_C0_ex
```

Format

A vector containing five numeric entries

See Also

sorcering.

RothC_Cin_ex *Carbon Input Data for RothC*

Description

Fictional carbon input for the RothC SOC model. Columns stand for pools and rows for simulation time steps.

Usage

```
RothC_Cin_ex
```

Format

A matrix of 5 columns and 60 rows

See Also

sorcering.

RothC_Cin_ex_sl *Carbon Input Data for RothC using multiple sites*

Description

Fictional carbon input for the RothC SOC model. The input data constists of a list of 3 matrices. Matrix columns stand for pools and matrix rows for simulation time steps.

Usage

```
RothC_Cin_ex_sl
```

Format

A list of 3 matrices with 5 columns and 60 rows each

See Also

sorcering.

RothC_Cin_ex_sl_spin *Carbon Input Data for RothC using multiple sites with spinup run*

Description

Fictional carbon input for the RothC SOC model. The input data constists of a list of 3 matrices. Matrix columns stand for pools and matrix rows for simulation time steps.

Usage

```
RothC_Cin_ex_sl_spin
```

Format

A list of 3 matrices with 5 columns and 12 rows each

See Also

sorcering.

RothC_env_in_ex *Environmental Input Data for RothC*

Description

Fictional environmental input for RothC. Rows stand for simulation time steps. Column 1: atmospheric temperature [Celsius degrees]. Column 2: precipitation [mm]. Column 3: evapotranspiration [mm]. Column 4: zeros or ones, the latter indicate time steps with growing crops.

Usage

```
RothC_env_in_ex
```

Format

A matrix of 4 columns and 60 rows

See Also

sorcering.

RothC_N0_ex *Initial Soil Organic Nitrogen Data for RothC*

Description

Fictional initial soil organic nitrogen for the RothC SOC model

Usage

```
RothC_N0_ex
```

Format

A vector containing five numeric entries

See Also

sorcering.

RothC_Nin_ex *Nitrogen Input Data for RothC*

Description

Fictional nitrogen input for the RothC SOC model. Columns stand for pools and rows for simulation time steps.

Usage

```
RothC_Nin_ex
```

Format

A matrix of 5 columns and 60 rows

See Also

sorcering.

RothC_Nin_ex_sl *Nitrogen Input Data for RothC using multiple sites*

Description

Fictional nitrogen input for the RothC SOC model. The input data constists of a list of 3 matrices. Matrix columns stand for pools and matrix rows for simulation time steps.

Usage

```
RothC_Nin_ex_sl
```

Format

A list of 3 matrices with 5 columns and 60 rows each

See Also

sorcering.

RothC_Nin_ex_sl_spin *Nitrogen Input Data for RothC using multiple sites with spinup run*

Description

Fictional nitrogen input for the RothC SOC model. The input data constists of a list of 3 matrices. Matrix columns stand for pools and matrix rows for simulation time steps.

Usage

```
RothC_Nin_ex_sl_spin
```

Format

A list of 3 matrices with 5 columns and 12 rows each

See Also

sorcering.

RothC_site_ex *Environmental Input Data for RothC*

Description

Fictional site information for RothC. Contains information to calculate xi. Vector of length 4: sample depth [mm], clay content [or 1, 0 if unknown or if black sand method is not desired) and CN ratio (0 if unknown, but then either C0 and N0 must be defined or calcC0 = TRUE and calcN0 = TRUE).

Usage

```
RothC_site_ex
```

Format

A vector containing 4 numeric entries

See Also

sorcering.

RothC_xi_ex *Environmental Factors Data for RothC*

Description

Fictional environmental factors for the RothC SOC model. Columns stand for pools and rows for simulation time steps.

Usage

```
RothC_xi_ex
```

Format

A matrix of 5 columns and 60 rows

See Also

sorcering.

sorcering *Soil ORganic Carbon & CN Ratio drIven Nitrogen modellinG framework*

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. An SOC model can be simply defined or a predefined existing SOC model can be chosen and then 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) either the initial distributions of C and N among model pools as a direct input or time series of at least three C and N measurement points with which these initial distributions can be calculated using linear regression, and (3) time series of C and N inputs and rate modifying environmental factors. In case a predefined SOC model is used, instead of model parameters and time series of rate modifying factors, model-specific environmental and stand data must be passed for the calculation of decomposition and transfer rates. The fourth-order Runge-Kutta algorithm is used to numerically solve the system of differential equations.

Usage

```
sorcering(    A = NULL,
              tsteps = "monthly",
              C0 = NULL,
              N0 = NULL,
              Cin = NULL,
              Nin = NULL,
              Cin_wood = NULL,
              Nin_wood = NULL,
              wood_diam = NULL,
              xi = NULL,
              env_in = NULL,
              site = NULL,
              theta = NULL,
              theta_unc = NULL,
              theta_n_unc = 1,
              meas_data = NULL,
              A_sl = NULL,
              C0_sl = NULL,
              N0_sl = NULL,
              Cin_sl = NULL,
              Nin_sl = NULL,
              Cin_wood_sl = NULL,
              Nin_wood_sl = NULL,
              wood_diam_sl = NULL,
```

```
xi_sl = NULL,
env_in_sl = NULL,
site_sl = NULL,
sitelist = NULL,
meas_data_sl = NULL,
calcN = FALSE,
calcNbalance = FALSE,
calcN0 = FALSE,
calcC0 = FALSE,
calcCN_fast_init = FALSE,
CTool_input_raw = FALSE,
RothC_Cin4C0 = FALSE,
RothC_dpmrpm = 1.439024,
C0_fracts = NULL,
multisite = FALSE,
pooltypes = NULL,
CN_fast_init = 40,
CN_bio = 9,
CN_spin = NULL,
CN_fast_init_sl = NULL,
CN_bio_sl = NULL,
CN_spin_sl = NULL,
init_info = FALSE,
model = "",
spinup = FALSE,
t_spin = 2,
t_spin_sl = 2)
```

Arguments

A	square matrix. Transfer matrix typical for SOC modelling. Defines number of pools, decomposition and transfer rates. n×n elements with n = number of pools. Diagonal values are decomposition rates [yr^{-1}]. Off-diagonals represent the transfer between pools . Only used when `model is NULL`
tsteps	character string indicating the type of simulation time steps. Valid options are `"annually"`, `"monthly"` (recommended) or `"weekly"`. Ensures that the rate modifying factors (passed through `xi` or used by a predefined model) are adjusted by dividing them by 1, 12, and 52 respectively. Also ensures that environment-specific information (passed through `env_in`) takes into account the time reference of the modelling.
C0	either vector with a length equal to the number of pools or scalar. If vector, initial soil organic carbon per pool [$\mathrm{tC\,ha}^{-1}$]. If scalar, initial total soil organic carbon [$\mathrm{tC\,ha}^{-1}$]. In the latter case, either `model` must be selected or `C0_fracts` must be passed. If `NULL`, filled with zeros.
N0	vector with a length equal to the number of pools. Contains initial soil organic nitrogen per pool [$\mathrm{tN\,ha}^{-1}$]. If `NULL`, filled with zeros. Only used when `calcN = TRUE` and `calcN0 = FALSE`.

Cin — either matrix with a number of columns equal to the number of pools and a number of rows corresponding to simulation time steps (if spinup = FALSE) or spin-up reference period (if spinup = TRUE), or list containing such matrices. If it is a list, each element of the list is expected to represent a stochastic repetition that covers input uncertainties. Then, the list must contain matrices of equal dimensions. Each matrix (or the one if modelling without uncertainties) must contain information about carbon input per pool and time step [$\mathrm{tC\,ha^{-1}}$]. When CTool_input_raw = TRUE, and model = "C-Tool" or model = "C-Tool-org", the matrix structure can have two columns (as described for CTool_input_raw). If NULL, filled with zeros.

Nin — either matrix with a number of columns equal to the number of pools and a number of rows corresponding to simulation time steps (if spinup = FALSE) or spin-up reference period (if spinup = TRUE), in each case in accordance with number of rows of Cin, or list containing such matrices. If it is a list, each element of the list is expected to represent a stochastic repetition that covers input uncertainties. Then, the list must contain matrices of equal dimensions. Each matrix (or the one if modelling without uncertainties) must contain information about nitrogen input per pool and time step [$\mathrm{tN\,ha^{-1}}$]. When CTool_input_raw = TRUE, and model = "C-Tool" or model = "C-Tool-org" the matrix structure can have 2 columns (as described for CTool_input_raw). If NULL, filled with zeros. Must contain entries > 0 where entries of Cin are > 0. Only used when calcN = TRUE.

Cin_wood — list of lengths of different wood diameter classes. Each list element must be in Cin format and represent a specific wood diameter. Furthermore, the list elements themselves can be lists and contain stochastic repetitions, as explained for Cin. The mean diameter per class is defined in wood_diam. Only used when model = "Yasso15" or model = "Yasso20".

Nin_wood — list of lengths of different wood diameter classes. Each list element must be in Nin format and represent a specific wood diameter. Furthermore, the list elements themselves can be lists and contain stochastic repetitions, as explained for Nin. The mean diameter per class is defined in wood_diam. Must contain entries > 0 where entries of Cin_wood are > 0. Only used when calcN = TRUE. Only used when model = "Yasso15" or model = "Yasso20".

wood_diam — vector with wood diameter [cm]. The first element corresponds to the first list element of Cin_wood and Nin_wood. If NULL, filled with zeros. Only used when Cin_wood is specified and when either model = "Yasso15" or model = "Yasso20". Must contain entries >= 0.

xi — either matrix with a number of columns equal to the number of pools and a number of rows corresponding to simulation time steps (if spinup = FALSE) or spin-up reference period (if spinup = TRUE), in each case in accordance with number of rows of Cin, or list containing such matrices. If it is a list, each element of the list is expected to represent a stochastic repetition that covers input uncertainties. Then, the list must contain matrices of equal dimensions. Each matrix (or the one if modelling without uncertainties) must contain information about time series of rate modifying factors for each model pool, built on the basis of annual decomposition rates. If NULL, filled with ones. Only used when model is NULL.

env_in — matrix with a model-specific number of columns and a number of rows corresponding to simulation time steps (if spinup = FALSE, and tsteps = "weekly" or tsteps = "monthly") or corresponding to simulation time steps multiplied by twelve (if spinup = FALSE and tsteps = "annually" or corresponding to individually chosen spin-up reference period (if spinup = TRUE). The number of rows must be in accordance with the number of rows of Cin, except when tsteps = "annually", then the number of rows must be twelve times the number of rows of Cin because monthly environmental variables still must be used to account for the annual cycle even when simulations run in annual mode. Contains environment-specific information to calculate rate modifying factors (instead of passing them with xi) and initial distributions (only RothC). When model = "RothC", it must have four columns: atmospheric temperature (T) [degrees C], precipitation (p) [mm], evapotranspiration [mm] and a vector of zeros, ones (both originally RothC) or twos (not originally RothC) describing the soil cover, where ones indicate time steps when the soil is vegetated, zeros when it is bare and twos when it is bare, but this only influences the accumulated but not the maximum topsoil moisture deficit. The latter will then be calculated as if there was soil cover. The idea behind this is that the water content should be decisive for the microorganisms as a habitat and transport medium, regardless of whether a plant is growing or not. When model = "Yasso07" or model = "Yasso15" or model = "Yasso20", it must have two columns: T [degrees C] and p [mm]. When model = "C-Tool" or model = "C-Tool-org", it has one column: T [degrees C]. If NULL, filled with ones. Only used when model is not NULL.

site — vector of model-specific length. Contains site-specific information to calculate rate modifying factors (instead of passing them with xi) and initial distributions (only RothC). and initial carbon and nitrogen distributions. When model = "RothC", it must be of length four: sample depth [mm], clay content [%], black sand status (0 or 1, 0 if unknown or if black sand method is not desired) and CN ratio (0 if unknown, but then either C0 and N0 must be passed or calcC0 = TRUE and calcN0 = TRUE, information on CN ratio given in site always takes precedence over internally calculated CN ratios). When model = "C-Tool" or model = "C-Tool-org", it must be of length one: clay content [%]. Only used when model = "RothC" or model = "C-Tool" or model = "C-Tool-org".

theta — either vector with model parameters for predefined models or matrix with rows of such parameters. If it is a matrix, each row is expected to represent a stochastic repetition that covers input uncertainties. If uncertainties are defined by another argument, e.g. Cin or Nin, these determine the number of stochastic repetitions and not theta. Then, if theta is a matrix, a parameter vector is randomly drawn for each uncertainty loop. Each vector (or row of matrix) must be of length 7 when model = "RothC", of length 10 when model = "C-Tool" or model = "C-Tool-org", of length 21 when model = "Yasso07" and of length 30 when model = "Yasso15" or model = "Yasso20". If NULL, model-specific standard parameters are used instead. Only used when model is not NULL. See model parameters table in section 'Details' for standard parameters used.

theta_unc — either number or vector of percentage values. If it is a vector, the same model-specific lengths as described for theta must be used. When used, model parameters modified by taking from the normal distribution around given values (either

from theta or predefined values) with a standard deviation of theta_unc. This will be repeated as many times as defined in theta_n_unc or as defined by uncertainty dimensions of a carbon or nitrogen input argument (e.g. Cin) and lead to unique model results and output list elements. Only used when model is not NULL and theta is not a matrix.

theta_n_unc — number of stochastic repetitions when model parameters for predefined models should be determined from a random distribution. Only used when the number of stochastic repetitions is not defined by another argument (e.g. Cin). Only used when model is not NULL, theta_unc is not NULL and theta is not a matrix.

meas_data — matrix with a number of rows equal to the number of measurement points. The first column defines the time of measurement, the metric of which is based on simulation time steps. The second row must contain values of measured soil organic carbon stock. The third row must contain values of measured soil organic nitrogen and is only used when calcN0 = TRUE. Only used when calcC0 = TRUE.

A_sl — list with a length of number of sites to simulate. Each list element represents a site and must be in A format. Only used when multisite = TRUE and model is NULL. When multisite = TRUE, A can be passed instead of A_sl to have the same argument for all sites.

C0_sl — list with a length of number of sites to simulate. Each list element represents a site and must be in C0 format. Only used when multisite = TRUE.

N0_sl — list with a length of number of sites to simulate. Each list element represents a site and must be in N0 format. Only used when multisite = TRUE, calcN = TRUE and calcN0 = FALSE.

Cin_sl — list with a length of number of sites to simulate. Each list element represents a site and must be in Cin format, which can also contain uncertainties. Thus, Cin_sl can either be a list of different sites each containing lists of different uncertainty representations each with matrices of carbon input as described for Cin, or it can simply be a list of different sites each containing such matrices. Only used when multisite = TRUE.

Nin_sl — list with a length of number of sites to simulate. Each list element represents a site and must be in Nin format, which can also contain uncertainties. Thus, Nin_sl can either be a list of different sites each containing lists of different uncertainty representations each with matrices of carbon input as described for Nin, or it can simply be a list of different sites each containing such matrices. Only used when multisite = TRUE. Must contain entries > 0 where entries of Cin_sl are > 0.

Cin_wood_sl — list with a length of number of sites to simulate. Each list element represents a site and must be in Cin_wood format, which can also contain uncertainties. Thus, Cin_wood_sl can either be a list of different sites each containing lists of different wood diameter representations, which in turn each contain lists of different uncertainty representations, each with matrices of carbon input as described for Cin, or it can simply be a list of different sites each containing lists of different wood diameter representations each containing such matrices. Only used when multisite = TRUE and either model = "Yasso15" or model = "Yasso20".

`Nin_wood_sl` list with a length of number of sites to simulate. Each list element represents a site and must be in `Nin_wood` format, which can also contain uncertainties. Thus, `Nin_wood_sl` can either be a list of different sites each containing lists of different wood diameter representations, which in turn each contain lists of different uncertainty representations, each with matrices of carbon input as described for `Nin`, or it can simply be a list of different sites each containing lists of different wood diameter representations each containing such matrices. Only used when `multisite = TRUE` and either `model = "Yasso15"` or `model = "Yasso20"`. Must contain entries `> 0` where entries of `Cin_wood_sl` are `> 0`.

`wood_diam_sl` list with a length of number of sites to simulate. Each list element represents a site and must be in `wood_diam` format. Only used when `multisite = TRUE` and either `model = "Yasso15"` or `model = "Yasso20"`.

`xi_sl` list with a length of number of sites to simulate. Each list element represents a site and must be in `xi` format, which can also contain uncertainties. In the latter case, the site list must include uncertainty lists. Only used when `multisite = TRUE`.

`env_in_sl` list with a length of number of sites to simulate. Each list element represents a site and must be in `env_in` format. Only used when `multisite = TRUE`.

`site_sl` list with a length of number of sites to simulate. Each list element represents a site and must be in `site` format. Only used when `multisite = TRUE`.

`sitelist` list with names of sites to simulate. Only used when `multisite = TRUE`.

`meas_data_sl` list with a length of number of sites to simulate. Each list element represents a site and must be in `meas_data` format. Consequently, it is only used when `calcC0 = TRUE` and the third row of each list element is only used when `calcN0 = TRUE`. Only used when `multisite = TRUE`.

`calcN` logical indicating whether soil organic nitrogen should be modeled.

`calcNbalance` logical indicating whether the balance of nitrogen cycling should be calculated.

`calcN0` logical indicating whether `N0` should be calculated. Then, the information in `meas_data` is used to determine initial states using linear regression.

`calcC0` logical indicating whether `C0` should be calculated. Then, the information in `meas_data` is used to determine initial states using linear regression.

`calcCN_fast_init`
logical indicating whether to calculate the initial CN ratio for fast pools (using `Cin` and `Nin`) or whether `CN_fast_init` should be used.

`CTool_input_raw`
logical defining of which type `Cin` and `Nin` are when modelling with `C-Tool`. If `TRUE`, `Cin` and `Nin` can only have two columns, one for the topsoil and one for the subsoil, and `SORCERING` is applying the `C-Tool`-specific distribution to model pools. If `FALSE` (default) `Cin` and `Nin` must have six columns, one per model pool. Only used when `model = "C-Tool"` or `model = "C-Tool-org"`.

`RothC_Cin4C0` logical defining whether the SOC input should be used for the calculation of initial SOC. If `FALSE` the standard RothC ratio for agricultural soils of DPM to RPM of 0.59 to 0.41 or the ratio defined in `RothC_dpmrpm` is used. Only used when `model = "RothC"`.

RothC_dpmrpm — positive decimal number defining the initital ratio between DPM and RPM pool or vector containing such numbers. If vector, each element represents a site when multisite = TRUE. The predefined value of 1.439024 equals 0.59/0.41, the standard RothC ratio for agricultural soils. Only used when model = "RothC".

C0_fracts — numerical vector of a length equal to the number of pools. Contains initial fractions of SOC in pools, the sum of which must be 1. Only used when calcC0 = TRUE or C0 is a scalar.

multisite — logical indicating whether multiple sites should be calculated with one program call. Then, t_spin, C0, N0, Cin, Nin, Cin_wood, Nin_wood, wood_diam, env_in, site_in, xi and meas_data must be of list type and replaced with t_spin_sl, C0_sl, N0_sl, Cin_sl, Nin_sl, Cin_wood_sl, Nin_wood_sl, wood_diam_sl, env_in_sl, site_in_sl, xi_sl and meas_data_sl, respectively. A, CN_bio and CN_fast_init can be given as single variables or in list form of A_sl, CN_bio_sl and CN_fast_init_sl, respectively.

pooltypes — integer vector with a length equal to the number of pools. Contains information necessary for the calculation of N0. Allowed values are 1-6. 1: topsoil fast pool, 2: topsoil bio or humus pool, 3: topsoil chemically stable or inert pool, 4: subsoil fast pool, 5: subsoil bio or humus pool, 6: subsoil chemically stable or inert pool. Predefined values are (1,1,2,2,3) when model = "RothC", (1,2,3,4,5,6) when model = "C-Tool" or model = "C-Tool-org", (1,1,1,2,3) when model = "Yasso07" or model = "Yasso15" or model = "Yasso20". Only used when calcN = TRUE and calcN0 = TRUE.

CN_fast_init — number that defines the initial CN ratio for fast pools (pooltypes = 1 or 4). Only used when Nin (or Nin_sl) and Cin (or Cin_sl) do not provide enough information for the estimation of initial nitrogen. The user will be informed about it when init_info = TRUE. Only used when calcN = TRUE and calcN0 = TRUE.

CN_bio — number that defines the initial CN ratio for slow pools (pooltypes = 2 or 5). Only used when calcN = TRUE and calcN0 = TRUE.

CN_spin — vector with a length equal to the number of pools. Defines the initial CN ratios for spin-up runs. For the case where the spinup starts from bare ground without soil organic components, the CN ratios of the pools must be defined. Since the CN ratios would then only be influenced by external inputs, the CN ratios for slow target pools without input would exceptionally be defined by the CN ratios of fast source pools due to a lack of alternatives. To prevent this, it is necessary to define initial CN ratios for spin-up runs in that case. Only used when calcN = TRUE and spinup = TRUE and for elements of which that of C0 (and N0) are zero.

CN_fast_init_sl — list with a length of number of sites to simulate. Each list element represents a site and must be in CN_fast_init format. Only used when calcN = TRUE, multisite = TRUE and calcN0 = TRUE. When multisite = TRUE, CN_fast_init can be passed instead of CN_fast_init_sl to have the same argument for all sites.

CN_bio_sl — list with a length of number of sites to simulate. Each list element represents a site and must be in CN_bio format. Only used when calcN = TRUE, multisite = TRUE and calcN0 = TRUE. When multisite = TRUE, CN_bio can be passed instead of CN_bio_sl to have the same argument for all sites.

CN_spin_sl — list of vectors that defines the initial CN ratios for spin-up runs. Each list element represents a site and must be in CN_spin format. Only used when multisite = TRUE, calcN = TRUE and spinup = TRUE and for elements of which that of N0_sl are zero. When multisite = TRUE, CN_spin can be passed instead of CN_spin_sl to have the same argument for all sites.

init_info — logical indicating whether additional information about the calculation of initial carbon, initial nitrogen, and CN ratio should be printed out during the simulations. Only used when calcC0 = TRUE or calcN0 = TRUE.

model — character string specifying a predefined soil organic carbon model to use. Valid options are "Yasso07", "Yasso15", "Yasso20", "RothC", "C-Tool" or "C-Tool-org". When not NULL, xi and A are calculated by SORCERING, and env_in must be specified. Additionally, theta can be specified to not use standard model parameters. See model parameters table in section 'Details' for standard parameters used. If calcN0 = TRUE and pooltypes = NULL model-specific standard values for pooltypes are used.

spinup — logical indicating whether the simulations should run in spin-up mode. Then, from all time-depending input (Cin, Nin, xi, env_in and site list derivates) random years (consisting of one time step when tsteps = "annually", of twelfth time step when tsteps = "monthly" and of fifty-second time step when tsteps = "weekly") are taken and t_spin defines the length of the spin-up. If modelling multiple sites, t_spin_sl allows for varying spin-up times. The length of the time-depending input is independent of the spin-up length, but all input data must refer to a specific reference period.

t_spin — integer number of spin-up time steps.

t_spin_sl — list with a length of number of sites to simulate or integer number of spin-up time steps. If list, each list element represents a site and must be in t_spin format. If integer number, applied to all sites. Only used when multisite = 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; Viskari et al. 2020; Viskari et al. 2022), RothC (Coleman and Jenkinson 1996) and C-Tool (Taghizadeh-Toosi et al. 2014). Moreover, it additionally offers the possibility of modelling N immobilisation and mineralisation by enhancing given SOC models by an additional N module. SORCERING was created using the C++ interface Rcpp (Eddelbuettel et al. 2021) and can handle multiple sites and multiple stochastic representations with just one function call. This makes SORCERING a computationally efficient SOC and SON modelling tool.

In the following a description of each output value (see section 'Value') is given. Detailed mathematical descriptions of the SOC and SON calculation, the optional extensions of the SORCERING function and the predefined models used can be found in the extended R documentation at browseVignettes("sorcering").

Value C

SORCERING calculates SOC applying a given SOC model for every simulation time step defined by passing tsteps and the number of rows of Cin (or number of rows of matrix elements in Cin_sl). SOC models applied here are defined by a number of pools, each characterised by specific decomposition and turnover rates. The model-specific decomposition kinetics and SOC fluxes among pools are described by a set of partial differential equations represented by the transfer matrix A (as passed with A or provided by model). Each row and column of A represent SOC pools. Off-diagonal elements of A describe SOC fluxes and diagonal elements describe SOC decomposition. The differential equations furthermore contain the boundary condition $Cin(t)$ (as passed with Cin) and the model-specific generated rate modifying factor series $xi(t)$ (as passed with xi or calculated for a predefined model). The change of SOC concentration in time is then defined as:

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

with

$$A_e(t) = A \cdot diag(xi(t))$$

Initial conditions must be defined for every SOC pool by passing C0 or by using the capabilities of SORCERING to calculate it. A description of the numerical solution can be found in the extended pdf documentation at browseVignettes("sorcering"). For more information on the functioning and possibilities of solving first-order kinetics SOC models see Sierra et al. (2012).

Value 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. As N decomposition is proportional to C decomposition, SON is calculated based on the results of the SOC calculations and input conditions (for details see the extended pdf documentation at browseVignettes("sorcering")).

Values Nloss, Nmin, Nmin.sink<1>, ..., Nmin.sink<n>

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

$$Nloss(t) = N(t-1) + Nin(t-1) - N(t)$$

In contrast, mineralisation rates contain information about sources and sinks of SON. They are calculated based on the CN ratios in the pools and the turnover rates (for details see the extended pdf documentation at browseVignettes("sorcering")). Pool-specific N mineralisation $Nmin.sink\left\langle 1\right\rangle, ..., Nmin.sink\left\langle n\right\rangle$ and N mineralisation $Nmin$ are related as follows:

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

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.

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:

$$\sum_{p=1}^{n} Nloss_p(t) \approx \sum_{p=1}^{n} Nmin_p(t)$$

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

Value Nbalance

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

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

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

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

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

$\Delta N(t)$ is saved in the first column, $N_{bal1}(t)$ in the second and $N_{bal2}(t)$ in the third column of `"Nbalance"`.

Model parameters

If a predefined model has been specified (`model is not NULL`) the following standard parameters are used. They can be changed using `theta` within the program call.

RothC

k_dpm	10	Decomposition rate for DPM pool [yr^{-1}]
k_rpm	0.3	Decomposition rate for RPM pool [yr^{-1}]
k_bio	0.66	Decomposition rate for BIO pool [yr^{-1}]
k_hum	0.02	Decomposition rate for HUM pool [yr^{-1}]
k_iom	0	Decomposition rate for IOM pool [yr^{-1}]
R_W_max	1	Maximum rate modifying factor for soil moisture
R_W_min	0.2	Minimum rate modifying factor for soil moisture

C-Tool

	C-Tool	C-Tool-org	
k_fom_t	1.44	1.44	Decomposition rate for FOM pool (topsoil) [yr^{-1}]
k_hum_t	0.0336	0.0336	Decomposition rate for HUM pool (topsoil) [yr^{-1}]
k_rom_t	0.000463	0	Decomposition rate for ROM pool (topsoil) [yr^{-1}]
k_fom_s	1.44	1.44	Decomposition rate for FOM pool (subsoil) [yr^{-1}]
k_hum_s	0.0336	0.0336	Decomposition rate for HUM pool (subsoil) [yr^{-1}]
k_rom_s	0.000463	0	Decomposition rate for ROM pool (subsoil) [yr^{-1}]
tf	0.03	0	Fraction going to downward transport
f_co2	0.628	0.628	Fraction of CO_2 released
f_rom	0.012	0	Fraction of fresh organic matter going to ROM pool
f_hum	0	0.358	Fraction of input going to HUM pool

Yasso

	Yasso07	Yasso15	Yasso20	
kA	0.66	0.49	0.51	Base decomposition rate for pool A [yr^{-1}]
kW	4.3	4.9	5.19	Base decomposition rate for pool W [yr^{-1}]
kE	0.35	0.25	0.13	Base decomposition rate for pool E [yr^{-1}]
kN	0.22	0.095	0.1	Base decomposition rate for pool N [yr^{-1}]
kH	0.0033	0.0013	0.0015	Base decomposition rate for pool H [yr^{-1}]
p1	0.32	0.44	0.5	Transference fraction from pool A to pool W
p2	0.01	0.25	0	Transference fraction from pool A to pool E
p3	0.93	0.92	1	Transference fraction from pool A to pool N
p4	0.34	0.99	1	Transference fraction from pool W to pool A
p5	0	0.084	0.99	Transference fraction from pool W to pool E
p6	0	0.011	0	Transference fraction from pool W to pool N
p7	0	0.00061	0	Transference fraction from pool E to pool A
p8	0	0.00048	0	Transference fraction from pool E to pool W
p9	0.01	0.066	0	Transference fraction from pool E to pool N
p10	0	0.00077	0	Transference fraction from pool N to pool A
p11	0	0.1	0.163	Transference fraction from pool N to pool W
p12	0.02	0.65	0	Transference fraction from pool N to pool E
pH	0.04	0.0046	0.0015	Transference fraction from AWEN pools to pool H
beta_1	0.076	0.091	0.158	1^{st}-order temperature parameter for AWE pools [$\mathrm{degrees\,C}^{-1}$]
beta_2	-0.00089	-0.00021	-0.002	2^{nd}-order temperature parameter for AWE pools [$\mathrm{degrees\,C}^{-2}$]
beta_N1	-	0.049	0.17	1^{st}-order temperature parameter for N pool [$\mathrm{degrees\,C}^{-1}$]
beta_N2	-	-0.000079	-0.005	2^{nd}-order temperature parameter for N pool [$\mathrm{degrees\,C}^{-2}$]
beta_H1	-	0.035	0.067	1^{st}-order temperature parameter for H pool [$\mathrm{degrees\,C}^{-1}$]
beta_H2	-	-0.00021	0	2^{nd}-order temperature parameter for H pool [$\mathrm{degrees\,C}^{-2}$]
gamma	-1.27	-1.8	-1.44	Precipitation impact parameter for AWE pools [$\mathrm{yr\,mm}^{-1}$]
gamma_N	-	-1.2	-2	Precipitation impact parameter for N pool [$\mathrm{yr\,mm}^{-1}$]
gamma_H	-	-13	-6.9	Precipitation impact parameter for H pool [$\mathrm{yr\,mm}^{-1}$]
theta_1	-	-0.44	-2.55	1^{st}-order impact parameter for wood size [cm^{-1}]
theta_2	-	1.3	1.24	2^{nd}-order impact parameter for wood size [cm^{-2}]
r	-	0.26	0.25	Exponent parameter for wood size

Value

SORCERING returns either a list of carbon and nitrogen output values or, when multisite = TRUE, a list broken down by site with result lists for each site. When modelling uncertainties (as can be defined by passing e.g. Cin, Nin, xi or theta), the output is even extended to include another list dimension that covers these uncertainties. The lowest output list-level contains the following components:

C — matrix with a number of rows corresponding to simulation time steps (number of rows of Cin or number of rows of matrix elements in Cin_sl, when spinup = FALSE), or to t_spin resp. t_spin_sl, when spinup = TRUE) and a number of columns equal to the number of pools. Contains soil organic carbon [$\mathrm{tC\,ha^{-1}}$].

N — matrix with a number of rows corresponding to simulation time steps (number of rows of Cin or number of rows of matrix elements in Cin_sl, when spinup = FALSE), or to t_spin resp. t_spin_sl, when spinup = TRUE) and a number of columns equal to the number of pools. Contains soil organic nitrogen [$\mathrm{tN\,ha^{-1}}$]. Only generated when calcN = TRUE.

Nloss — matrix with a number of rows corresponding to simulation time steps (number of rows of Cin or number of rows of matrix elements in Cin_sl, when spinup = FALSE), or to t_spin resp. t_spin_sl, when spinup = TRUE) and a number of columns equal to the number of pools. Contains nitrogen losses [$\mathrm{tN\,ha^{-1}}$]. Positive values indicate that nitrogen was lost in the pools between this and the previous time steps (taking nitrogen decomposition and input into account). Only generated when calcN = TRUE.

Nmin — matrix with a number of rows corresponding to simulation time steps (number of rows of Cin or number of rows of matrix elements in Cin_sl, when spinup = FALSE), or to t_spin resp. t_spin_sl, when spinup = TRUE) and a number of columns equal to the number of pools. Contains nitrogen mineralisation [$\mathrm{tN\,ha^{-1}}$]. If values are negative, nitrogen immobilisation exceeds mineralisation. Only generated when calcN = TRUE.

Nmin.sink.1, ..., Nmin.sink.n — matrices with a number of rows corresponding to simulation time steps (number of rows of Cin or number of rows of matrix elements in Cin_sl, when spinup = FALSE), or to t_spin resp. t_spin_sl, when spinup = TRUE) and a number of columns equal to the number of pools n. Contain pool-specific nitrogen mineralisation sinks [$\mathrm{tN\,ha^{-1}}$] (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 when calcN = TRUE.

Nbalance — matrix with a number of rows corresponding to simulation time steps (number of rows of Cin or number of rows of matrix elements in Cin_sl, when spinup = FALSE), or to t_spin resp. t_spin_sl, when spinup = TRUE) 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) [$\mathrm{tN\,ha^{-1}}$]. Only generated when calcN = TRUE and calcNbalance = TRUE.

Package Building Information

The SORCERING code was written in C++ using the R packages Rcpp (Eddelbuettel et al. 2021) and RcppArmadillo (Eddelbuettel et al. 2021). This documentation was built with the help of the R packages mathjaxr (Viechtbauer 2021) and Rdpack (Boshnakov 2021).

Author(s)

Marc Scherstjanoi <marc.scherstjanoi@thuenen.de>, Rene Dechow

References

Boshnakov GN (2021). *Rdpack: Update and Manipulate Rd Documentation Objects*. R package version 2.1.1, https://CRAN.R-project.org/package=Rdpack.

Coleman K, Jenkinson DS (1996). "RothC-26.3 - A Model for the turnover of carbon in soil." In Powlson DS, Smith P, Smith JU (eds.), *Evaluation of Soil Organic Matter Models*, 237–246. ISBN 978-3-642-61094-3.

Eddelbuettel D, Francois R, Allaire JJ, Ushey K, Kou Q, Russell N, Bates D, Chambers J (2021). *Rcpp: Seamless R and C++ Integration*. R package version 1.0.6, https://CRAN.R-project.org/package=Rcpp.

Eddelbuettel D, Francois R, Bates D, Ni B (2021). *RcppArmadillo: 'Rcpp' Integration for the 'Armadillo' Templated Linear Algebra Library*. R package version 0.10.4.0.0, https://CRAN.R-project.org/package=RcppArmadillo.

Sierra CA, Mueller M, Trumbore SE (2012). "Models of soil organic matter decomposition: the SoilR package, version 1.0." *Geoscientific Model Development*, **5**(4), 1045–1060. doi:10.5194/gmd510452012, https://gmd.copernicus.org/articles/5/1045/2012/.

Taghizadeh-Toosi A, Christensen BT, Hutchings NJ, Vejlin J, Kaetterer T, Glendining M, Olesen JE (2014). "C-TOOL: A simple model for simulating whole-profile carbon storage in temperate agricultural soils." *Ecological Modelling*, **292**, 11 - 25. ISSN 0304-3800, doi:10.1016/j.ecolmodel.2014.08.016.

Tuomi M, Thum T, Jaervinen H, Fronzek S, Berg B, Harmon M, Trofymow JA, Sevanto S, Liski J (2009). "Leaf litter decomposition-Estimates of global variability based on Yasso07 model." *Ecological Modelling*, **220**(23), 3362 - 3371. ISSN 0304-3800, doi:10.1016/j.ecolmodel.2009.05.016.

Viechtbauer W (2021). *mathjaxr: Using 'Mathjax' in Rd Files*. R package version 1.4-0, https://CRAN.R-project.org/package=mathjaxr.

Viskari T, Laine M, Kulmala L, Maekelae J, Fer I, Liski J (2020). "Improving Yasso15 soil carbon model estimates with ensemble adjustment Kalman filter state data assimilation." *Geoscientific Model Development*, **13**(12), 5959–5971. doi:10.5194/gmd1359592020, https://gmd.copernicus.org/articles/13/5959/2020/.

Viskari T, Pusa J, Fer I, Repo A, Vira J, Liski J (2022). "Calibrating the soil organic carbon

model Yasso20 with multiple datasets." *Geoscientific Model Development*, **15**(4), 1735–1752. doi:10.5194/gmd1517352022, https://gmd.copernicus.org/articles/15/1735/2022/.

Examples

```
#1 RothC application with fictional input for a single site

#1.1 Input

data(RothC_Cin_ex, RothC_Nin_ex, RothC_N0_ex, RothC_C0_ex, RothC_xi_ex,
  RothC_site_ex, RothC_env_in_ex) #fictional data

#1.2 Simulations

#In the following two methods are presented, one with a RothC as a predefined
#model (1.2.1), one where the RothC rate modifying factors must be calculated
#beforehand (1.2.2). Both methods lead to the same results.

#1.2.1 Simulation with predefined model

out_rothC <- sorcering( model="RothC", site=RothC_site_ex, env_in=RothC_env_in_ex,
  Cin=RothC_Cin_ex, Nin=RothC_Nin_ex, N0=RothC_N0_ex, C0=RothC_C0_ex,
  calcN=TRUE, tsteps="monthly")

#1.2.2 Simulation with own model definition and rate modifying factor definition

A_RothC <- fget_A_RothC(clay=30) #create transfer matrix for RothC
out_rothC_own <- sorcering(A=A_RothC , xi=RothC_xi_ex, Cin=RothC_Cin_ex,
  Nin=RothC_Nin_ex, N0=RothC_N0_ex, C0=RothC_C0_ex, calcN=TRUE, tsteps="monthly")
#Note that RothC_xi_ex contains site and model specific rate modifying factors that
#are only valid in this specific example. Generally, xi must be calculated by the
#user for different environmental conditions and SOC models used.

#1.3 Results

#output structure summary
summary(out_rothC)

#show that results of 1.2.1 and 1.2.2 differ negligibly
all( abs(out_rothC$C-out_rothC_own$C) < 1e-14)
all( abs(out_rothC$N-out_rothC_own$N) < 1e-14)

#example plot
  oldpar <- par(no.readonly=TRUE) #save old par
par(mfrow=c(1,1),mar=c(4,4,1,4))
plot(rowSums(out_rothC$N),axes=FALSE, col=1, cex.lab=2,xlab="",ylab="",ylim=c(0,9),
  pch=20)
par(new=TRUE)
plot(rowSums(RothC_Cin_ex)/rowSums(RothC_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=0.7, 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=expression("total N  [t ha"^-1*"]"), line=2, cex.lab=2)
mtext("C input / N input", side=4, line=2, cex=2,col=2)
title(xlab="time", line=2, cex.lab=2)
par(oldpar) #back to old par

#2 RothC application with fictional input for a multiple site application

#2.1 Input

data(RothC_Cin_ex_sl, RothC_Nin_ex_sl, RothC_N0_ex, RothC_C0_ex, RothC_site_ex,
  RothC_env_in_ex) #fictional data

#2.2. Simulation

out_multi_rothC <- sorcering( model="RothC", site_sl=rep(list(RothC_site_ex),3),
  env_in_sl=rep(list(RothC_env_in_ex),3), Cin_sl=RothC_Cin_ex_sl,
 Nin_sl=RothC_Nin_ex_sl, N0_sl=rep(list(RothC_N0_ex),3),C0_sl=rep(list(RothC_C0_ex),3),
  calcN=TRUE, tsteps="monthly", multisite=TRUE,
  sitelist=list("normal","half_input","double_Cin"))

#2.3 Results

#output structure summary
summary(out_multi_rothC$normal)
summary(out_multi_rothC$half_input)
summary(out_multi_rothC$double_Cin)

#example plot
oldpar <- par(no.readonly=TRUE) #save old par
par(mfrow=c(1,1),mar=c(4,4,1,4))
for (listelement in c(1:3))
{
  lwidth<-1
  if (listelement==2)lwidth<-3
 plot(rowSums(out_multi_rothC[[listelement]]$N),axes=FALSE, col=1,type="l", lwd=lwidth,
      lty=listelement+2,cex.lab=2,xlab="",ylab="",ylim=c(0,18))
  par(new=TRUE)
 plot(rowSums(RothC_Cin_ex_sl[[listelement]])/rowSums(RothC_Nin_ex_sl[[listelement]]),
      type="l", lwd=lwidth, lty=listelement+2,axes=FALSE,col=2, cex.lab=2,xlab="",
      ylab="",ylim=c(0,120))
  par(new=TRUE)
}
axis(side=2, pos=0,
  labels=(0:6)*3, at=(0:6)*20, hadj=0.7, padj=0.5, cex.axis=2,las=1,col.axis=1)
axis(side=4, pos=60,
  labels=(0:6)*20, at=(0:6)*20, 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=expression("total N  [t ha"^-1*"]"), line=2, cex.lab=2)
```

```
mtext("C input / N input", side=4, line=2, cex=2,col=2)
title(xlab="time", line=2, cex.lab=2)
legend(x=40,y=100,legend=c("normal","half_input","double_Cin"),lty=c(3,4,5),
 lwd=c(1,3,1))
par(oldpar) #back to old par

#3 RothC application with fictional input
#and fictional measurement data to calculate C0 and N0

#3.1 Input

#fictional data
data(RothC_Cin_ex_sl, RothC_Nin_ex_sl, RothC_site_ex, RothC_env_in_ex, meas_data_ex)

#3.2. Simulation

out_rothC_C0<-sorcering( model="RothC", site=RothC_site_ex, env_in=RothC_env_in_ex,
  Cin=RothC_Cin_ex, Nin=RothC_Nin_ex, calcC0=TRUE, calcN=TRUE, calcN0=TRUE,
  tsteps="monthly", meas_data=meas_data_ex)

#3.3 Results

#output structure summary
summary(out_rothC_C0)

#example plot
oldpar <- par(no.readonly=TRUE) #save old par
par(mfrow=c(1,1),mar=c(4,4,1,4))
plot(rowSums(out_rothC_C0$N),axes=FALSE, col=1, cex.lab=2,xlab="",ylab="",ylim=c(0,9),
  type="l",lwd=1)
par(new=TRUE)
plot(rowSums(out_rothC_C0$C),axes=FALSE, col=2, cex.lab=2,xlab="",ylab="",ylim=c(0,90),
  type="l",lwd=1)
par(new=TRUE)
plot(x=meas_data_ex[,1],y=meas_data_ex[,3],axes=FALSE, col=1, cex.lab=2,xlab="",ylab="",
  xlim=c(0,length(rowSums(out_rothC_C0$N))),ylim=c(0,9),pch=4,cex=3)
par(new=TRUE)
plot(x=meas_data_ex[,1],y=meas_data_ex[,2],axes=FALSE, col=2, cex.lab=2,xlab="",ylab="",
  xlim=c(0,length(rowSums(out_rothC_C0$N))),ylim=c(0,90),pch=4,cex=3)
par(new=TRUE)
axis(side=2, pos=0,
  labels=(0:8)*1, at=(0:8)*10, hadj=1, padj=0.5, cex.axis=2,las=1,col.axis=1)
axis(side=4, pos=60,
  labels=(0:8)*10, at=(0:8)*10, hadj=0, padj=0.5, cex.axis=2, las=1,col.axis=2)
axis(side=1, pos=0,
  labels= (0:8)*10 , at=(0:8)*10, hadj=0.5, padj=0, cex.axis=2)
title(ylab=expression("SON [t ha"^-1*"]"), line=2, cex.lab=2)
mtext(expression("SOC [t ha"^-1*"]"), side=4, line=3, cex=2,col=2)
title(xlab="time", line=2, cex.lab=2)
legend(x=30,y=30,legend=c("model result","measurement"),lwd=c(1,0))
legend(x=31,y=30,legend=c("",""),pch=4,pt.cex=c(0,3),bty="n")
par(oldpar) #back to old par
```

```
#4 Yasso15 application using multiple sites and
#input values of different wood diameters which take uncertainties into account

#4.1 Input

data(Yasso_Cin_ex_wood_u_sl, Yasso_Nin_ex_wood_u_sl, Yasso_C0_ex_sl, Yasso_N0_ex_sl,
  RothC_env_in_ex) #fictional data

#show last entries of C input for 3rd site, 2nd wood layer, 4th uncertainty layer
tail(Yasso_Cin_ex_wood_u_sl[[3]][[2]][[4]])

#diameter of wood input: 2 classes of 0 cm and 10 cm for each of the 3 sites
wood_diam_ex_sl<-list(c(0,10),c(0,10),c(0,10))

#environmental variables
Yasso_env_in_ex<-RothC_env_in_ex[,1:2]

#4.2 Simulation

out_multi_yasso_wood_unc <- sorcering( model="Yasso15", C0_sl=Yasso_C0_ex_sl,
  env_in_sl=rep(list(Yasso_env_in_ex),3), wood_diam_sl=wood_diam_ex_sl,
  Cin_wood_sl=Yasso_Cin_ex_wood_u_sl,Nin_wood_sl=Yasso_Nin_ex_wood_u_sl,
  N0_sl=Yasso_N0_ex_sl, calcN=TRUE, tsteps="monthly", multisite=TRUE,
  sitelist=list("a","b","c"))

#4.3 Results

#show the last C results for 3rd site, 4th uncertainty layer
tail(out_multi_yasso_wood_unc[[3]][[4]]$C)

#5 RothC application using stochastically varying parameters
#and multiple sites

#5.1 fictional data
data(RothC_Cin_ex_sl, RothC_Nin_ex_sl, RothC_C0_ex, RothC_N0_ex,
  RothC_site_ex, RothC_env_in_ex)

#standard deviations [%] used for each of the 7 RothC theta parameters
RothC_theta_unc <- c(0,0,1,1,1,1,2)

#5.2 Simulation

out_sl <- sorcering( model="RothC", site_sl=rep(list(RothC_site_ex),3),
  env_in_sl=rep(list(RothC_env_in_ex),3), Cin_sl=RothC_Cin_ex_sl,
  Nin_sl=RothC_Nin_ex_sl, C0_sl=rep(list(RothC_C0_ex),3),
  N0_sl=rep(list(RothC_N0_ex),3),calcN=TRUE,theta_n_unc=10,
  theta_unc=RothC_theta_unc, multisite=TRUE,
  sitelist=list("normal","half_input","double_Cin"))

#5.3 Means and standard deviation

#60 time steps, 5 pools, 9 output types, 10 theta_n_unc, 3 sites
out_sl_arr <- array(unlist(out_sl),c(60,5,9,10,3))
```

```
out_sl_arr_N <- out_sl_arr[,,2,,] #only output type 2: N
#mean over all uncerts
out_sl_arr_N_mean <- apply( out_sl_arr_N , c(1,2,4), na.rm=TRUE, FUN=mean )

#standard deviation
out_sl_arr_N_sd<-
array(0, dim=c(dim(out_sl_arr_N)[1],dim(out_sl_arr_N)[2],dim(out_sl_arr_N)[4]))
for (dim3 in c(1:dim(out_sl_arr_N)[4]))
  out_sl_arr_N_sd[,,dim3]<-apply(out_sl_arr_N[,,,dim3],c(1:2),sd)

#5.4 Results

#show the last N means for stand 1
tail(out_sl_arr_N_mean[,,1])

#show the last N standard deviations for stand 1
tail(out_sl_arr_N_sd[,,1])

#6 How to create input lists for a RothC application using stochastically
#varying inputs and input scenarios

#6.1 Input

#fictional data
data(RothC_Cin_ex_sl, RothC_C0_ex, RothC_site_ex, RothC_env_in_ex)

#create input list of 3 scenarios, 100 uncertainties each
set.seed(17) #to make 'random' results reproducible
f1<-1
for (no in c(1:3)) #loop over 3 input scenarios
{
    #normal, half and double input
    Cin <- switch (no, RothC_Cin_ex, RothC_Cin_ex/2, RothC_Cin_ex*2)
    f2 <- 1
    #create fictional uncertainties
    for (unc in c(1:100)) #loop over 100 uncertainties
    {
        randnum<-max(0,rnorm(1,1,0.5)) #out of normal dist. with 50% sd.
        if (f2==1) Cin_u <- list(Cin*randnum) else
        Cin_u[[length(Cin_u)+1]] <- Cin*randnum
        f2 <- 0
    }
    if (f1==1) Cin_u_sl <- list(Cin_u) else
    Cin_u_sl[[length(Cin_u_sl)+1]] <- Cin_u
    f1 <- 0
}

#show input of scenario 3, uncertainty 51
head(Cin_u_sl[[3]][[51]])

#6.2 Simulation
out_sl <- sorcering( model="RothC", site_sl=rep(list(RothC_site_ex),3),
  env_in_sl=rep(list(RothC_env_in_ex),3),Cin_sl=Cin_u_sl,
```

```
  C0_sl=list(RothC_C0_ex,RothC_C0_ex,RothC_C0_ex), tsteps="monthly",
  multisite=TRUE, sitelist=list("normal","half_input","double_Cin"))

#6.3 Means and standard deviation

#60 time steps, 5 pools, 1000 uncertainties, 3 sites
out_sl_arr <- array(unlist(out_sl),c(60,5,100,3))

#means
out_sl_arr_mean <- apply( out_sl_arr , c(1,2,4), na.rm=TRUE, FUN=mean )

#standard deviation
out_sl_arr_sd<-
  array(0, dim=c(dim(out_sl_arr)[1],dim(out_sl_arr)[2],dim(out_sl_arr)[4]))
for (dim3 in c(1:dim(out_sl_arr)[4]))
  out_sl_arr_sd[,,dim3]<-apply(out_sl_arr[,,,dim3],c(1:2),sd)

#6.4 Results

#C-pool sums of means for the 3 scenarios
totalC_m1<-rowSums(out_sl_arr_mean[,,1])
totalC_m2<-rowSums(out_sl_arr_mean[,,2])
totalC_m3<-rowSums(out_sl_arr_mean[,,3])

#C-pool sums of standard deviations for the 3 scenarios
totalC_s1<-rowSums(out_sl_arr_sd[,,1])
totalC_s2<-rowSums(out_sl_arr_sd[,,2])
totalC_s3<-rowSums(out_sl_arr_sd[,,3])

#example plot
oldpar <- par(no.readonly=TRUE) #save old par
par(mfrow=c(1,1),mar=c(4,4,1,4))
plot(totalC_m1,axes=FALSE, col=2, cex.lab=2,xlab="",ylab="",ylim=c(0,100),
  type="l",lwd=1)
par(new=TRUE)
plot(totalC_m2,axes=FALSE, col=3, cex.lab=2,xlab="",ylab="",ylim=c(0,100),
  type="l",lwd=1)
par(new=TRUE)
plot(totalC_m3,axes=FALSE, col=4, cex.lab=2,xlab="",ylab="",ylim=c(0,100),
  type="l",lwd=1)
par(new=TRUE)
polygon(c(1:60,60:1),c(totalC_m1+totalC_s1, rev(totalC_m1-totalC_s1)),
  border=NA,col=rgb(1,0,0,0.27),density=40,angle=180,xlab="",ylab="")
par(new=TRUE)
polygon(c(1:60,60:1),c(totalC_m2+totalC_s2, rev(totalC_m2-totalC_s2)),
  border=NA,col=rgb(0,1,0,0.27),density=30,xlab="",ylab="")
par(new=TRUE)
polygon(c(1:60,60:1),c(totalC_m3+totalC_s3, rev(totalC_m3-totalC_s3)),
  border=NA,col=rgb(0,0,1,0.27),density=25,angle=90,xlab="",ylab="")
par(new=TRUE)
axis(side=2, pos=0,
    labels=(0:10)*1, at=(0:10)*10, hadj=1, padj=0.5, cex.axis=2,las=1,col.axis=1)
axis(side=1, pos=0,
```

```
    labels= (0:6)*10 , at=(0:6)*10, hadj=0.5, padj=0, cex.axis=2)
title(ylab=expression("SOC [t ha"^-1*"]"), line=2, cex.lab=2)
title(xlab="time", line=2, cex.lab=2)
legend(x=20,y=30,fill=c(0,0,0,4,2,3),density=c(0,0,0,25,40,30),angle=c(0,0,0,90,0,45),
  border=c(0,0,0,1,1,1),legend=c("mean double input scenario",
  "mean regular input scenario", "mean half input scneario",
 "uncertainty range double input scenario", "uncertainty range regular input scenario",
  "uncertainty range half input scenario"))
legend(x=20,y=30,lty=c(1,1,1,0,0,0),seg.len=c(1,1,1,0,0,0), col=c(4,2,3,0,0,0),
  legend=c("","","","","",""),bty="n")
par(oldpar) #back to old par

#7 RothC application with fictional input for a spin-up application

#7.1 Input

#fictional data
data(RothC_Cin_ex_sl_spin, RothC_Nin_ex_sl_spin, RothC_site_ex, RothC_env_in_ex)

#7.2. Simulation

out_multi_rothC <- sorcering( model="RothC", site_sl=rep(list(RothC_site_ex),3),
  env_in_sl=rep(list(RothC_env_in_ex[1:12,]),3), Cin_sl=RothC_Cin_ex_sl_spin,
  Nin_sl=RothC_Nin_ex_sl_spin, calcN=TRUE, tsteps="monthly", multisite=TRUE,
  sitelist=list("normal","half_input","double_Cin"), spinup=TRUE, t_spin_sl=36000,
  C0=c(0,0,0,0,20), N0=c(0,0,0,0,2), CN_spin=c(100,100,50,50,10))

#7.3 Results

#example plot
oldpar <- par(no.readonly=TRUE) #save old par
par(mfrow=c(1,1),mar=c(4,4,1,4))
for (listelement in c(1:3))
{
  lwidth<-1
  if (listelement==2)lwidth<-3
  printN<-rowSums(out_multi_rothC[[listelement]]$N)
  printseq<-seq.int(1L,length(printN),100L)
  printC<-rowSums(out_multi_rothC[[listelement]]$C)
  plot(printN[printseq],axes=FALSE, col=1,type="l", lwd=lwidth,
      lty=listelement+2,cex.lab=2,xlab="",ylab="",ylim=c(0,30))
  par(new=TRUE)
  plot(printC[printseq],axes=FALSE, col=2,type="l", lwd=lwidth,
      lty=listelement+2,cex.lab=2,xlab="",ylab="",ylim=c(0,180))
  par(new=TRUE)
}
axis(side=2, pos=0,
  labels=(0:6)*5, at=(0:6)*30, hadj=0.7, padj=0.5, cex.axis=2,las=1,col.axis=1)
axis(side=4, pos=360,
  labels=(0:6)*30, at=(0:6)*30, hadj=0, padj=0.5, cex.axis=2, las=1,col.axis=2)
axis(side=1, pos=0,
  labels= (0:6)*6000 , at=(0:6)*60, hadj=0.5, padj=0, cex.axis=2)
title(ylab=expression("total N  [t ha"^-1*"]"), line=2, cex.lab=2)
```

```
mtext(expression("total C [t ha"^-1*"]"), side=4, line=2, cex=2,col=2)
title(xlab="time", line=2, cex.lab=2)
legend(x=120,y=140,legend=c("normal","half_input","double_Cin"),lty=c(3,4,5),
  lwd=c(1,3,1))
par(oldpar) #back to old par
```

`Yasso_C0_ex_sl` *Initial Soil Organic Carbon Data for Yasso using Multiple Sites*

Description

Fictional initial soil organic carbon for the Yasso SOC model (any Yasso version). The initial data constists of a list of 3 vectors, each containing five numeric entries, one for each model pool.

Usage

```
Yasso_C0_ex_sl
```

Format

A list of 3 vectors, each containing five numeric entries

See Also

sorcering.

`Yasso_Cin_ex_wood_u_sl`
Carbon Input Data for Yasso using Multiple Sites

Description

Fictional carbon input for the Yasso SOC model (Yasso15 or Yasso20). The input data constists of a list of 3 lists, each representing a site and containing 2 sublists. Each sublist represents a wood input layer and contains 4 matrices. Each matrix represents an uncertainty repetition and has 5 columns and 60 rows. Columns stand for pools and rows for simulation time steps.

Usage

```
Yasso_Cin_ex_wood_u_sl
```

Format

A list of 3 lists with 2 sublists each, each sublist containing 4 matrices with 5 columns and 60 rows

See Also

sorcering.

Yasso_N0_ex_sl *Initial Soil Organic Nitrogen Data for Yasso using Multiple Sites*

Description

Fictional initial soil organic nitrogen for the Yasso SOC model (any Yasso version). The initial data constists of a list of 3 vectors, each containing five numeric entries, one for each model pool.

Usage

```
Yasso_N0_ex_sl
```

Format

A list of 3 vectors, each containing five numeric entries

See Also

sorcering.

Yasso_Nin_ex_wood_u_sl *Nitrogen Input Data for Yasso using Multiple Sites*

Description

Fictional nitrogen input for the Yasso SOC model (Yasso15 or Yasso20). The input data constists of a list of 3 lists, each representing a site and containing 2 sublists. Each sublist represents a wood input layer and contains 4 matrices. Each matrix represents an uncertainty repetition and has 5 columns and 60 rows. Columns stand for pools and rows for simulation time steps.

Usage

```
Yasso_Nin_ex_wood_u_sl
```

Format

A list of 3 lists with 2 sublists each, each sublist containing 4 matrices with 5 columns and 60 rows

See Also

sorcering.

Index