Package ‘soilfoodwebs’

May 9, 2023

Type Package

Title Soil Food Web Analysis

Version 1.0.2

Maintainer Robert Buchkowski <robert.buchkowski@gmail.com>

Description Analyzing soil food webs or any food web measured at equilibrium. The package calculates carbon and nitrogen fluxes and stability properties using methods described by Hunt et al. (1987) <doi:10.1007/BF00260580>, de Ruiter et al. (1995) <doi:10.1126/science.269.5228.1257>, Holtkamp et al. (2011) <doi:10.1016/j.soilbio.2010.10.004>, and Buchkowski and Lindo (2021) <doi:10.1111/1365-2435.13706>. The package can also manipulate the structure of the food web as well as simulate food webs away from equilibrium and run decomposition experiments.

License GPL-3

Encoding UTF-8

LazyData true

RoxygenNote 7.2.3

Imports stringr (>= 1.4.0), diagram (>= 1.6.5), graphics (>= 4.1.0), quadprog (>= 1.5-8), lpSolve (>= 5.6.15), rootSolve (>= 1.8.2.2), deSolve (>= 1.28)

Depends R (>= 2.10)

Suggests rmarkdown, knitr, testthat (>= 3.0.0)

Config/testthat/edition 3

VignetteBuilder knitr

NeedsCompilation no

Author Robert Buchkowski [aut, cre, cph],
Zoë Lindo [aut],
Carlos Barreto [aut]

Repository CRAN

Date/Publication 2023-05-09 16:50:02 UTC

R topics documented:

`Aijfcn` *A function to calculate the nitrogen surplus or deficit each species gets from consuming another species*

Description

A function to calculate the nitrogen surplus or deficit each species gets from consuming another species

Usage

```
Aijfcn(usin)
```

Arguments

`usin`	The community on which nitrogen calculations are made.

Value

A matrix of nitrogen surpluses or deficits.

Examples

```
Aijfcn(intro_comm)
```

`Andres2016` *The soil food webs published for grazed and ungrazed plots in the Shortgrass Steppe long-term research station.*

Description

The community contains 21 nodes in bacterial, fungal, and root energy channels.

Usage

```
Andres2016
```

Format

A list of six communities each with a feeding matrix and properties dataframe. NOTE: You must select one of the communities from the list to use the package functions. For example: comana(Andres2016$GA) carries out calculations for the GA plot.

GA The grazed plot A.

UGA The ungrazed plot A.

GB The grazed plot B.

UGB The ungrazed plot B.

GC The grazed plot C.

UGC The ungrazed plot C.

imat Within the community: The feeding matrix. Rows eat columns.

prop Within the community: The properties data frame containing node names (ID), assimilation efficiency (a), production efficiency(p), C:N ratio (CN), biomass (B), death rate (d), proportion of death cycled back to a detrital pool (DetritusRecycling), Booleans stating whether the node is detritus, plant, and can immobilize nitrogen, and a list of mutual predators. Biomass is in kilograms of carbon per hectare and turnover/death rate is in years.

Source

doi:10.1016/j.soilbio.2016.02.014

`build_foodweb` *A function to compile the food web from simple data inputs*

Description

A function to compile the food web from simple data inputs

Usage

```
build_foodweb(feeding, properties)
```

Arguments

`feeding`	A data frame listing the feeding relationships in the food web. Only needs to contain the nodes that have prey items along with their names. Three columns in this data frame must be: Predator, Prey, and Preference. The Predator column is the name of the predator, the Prey column is the name of the prey, and the Preference is the preference that the predator has for that prey item after correcting for abundance. Preference values are relative, so for each predator the preference value can be any number and all that matters is its proportion of the total of all preference values given to that predator. Preference is meaningless if a predator only has one prey item. A good default value is 1 for everything if you don't want to set preferences beyond prey abundance.
`properties`	A data frame listing the properties or parameters in the food web. Must contain the names of all of the nodes in the food web. This data frame must contain: ID, d,a,p,B, CN, Detritusrecycling, isDetritus, isPlant, and canIMM.

Value

A community that is compatible with the functions of soilfoodwebs. This is a list with the feeding matrix imat first and the properties prop second.

Examples

```
# Creating a simple three node community:

# Create a data frame of feeding relationships:
feedinglist = data.frame(
Predator = c("Pred", "Pred"),
Prey = c("Prey1", "Prey2"),
Preference = c(1,1.2))

# Create a data frame of properties for each species:
properties = data.frame(ID = c("Pred", "Prey1", "Prey2"), # Name
d = c(1,3,0.5), # Death rate
a = c(0.61,0.65,0.45), # Assimilation efficiency
p = c(0.5,0.4,0.3), # Production efficiency for carbon
B = c(0.1,8,5), # Biomass
CN = c(4.5,4.8,5), # Carbon to nitrogen ratio
DetritusRecycling = c(0,0,0), # proportion of detritus recycling
isDetritus = c(0,0,0), # Boolean: Is this pool detritus?
isPlant = c(0,0,0), # Boolean: Is this pool a plant?
canIMM = c(0,0,0)) # Boolean: Can the pool immobilize inorganic nitrogen?

# Build the food web:
build_foodweb(feedinglist, properties)
```

`calculate_inputs` *A function to calculate the inputs and outputs at equilibrium and print them for the user.*

Description

A function to calculate the inputs and outputs at equilibrium and print them for the user.

Usage

```
calculate_inputs(
  usin,
  DIETLIMTS = NA,
  diet_correct = TRUE,
  Conly = FALSE,
  verbose = TRUE,
  toround = TRUE
)
```

Arguments

`usin`	The community you want to simulate.
`DIETLIMTS`	The diet limits matrix for the stoichiometry correction (proportion of diet)?
`diet_correct`	Boolean: Does the organism correct it’s diet?

`Conly`	Boolean: Is the model meant for carbon only?
`verbose`	Boolean: Do you want to print the summary in the Console?
`toround`	Boolean: Should the answer be rounded?

Value

A list of inputs and mineralization rates.

Examples

```
# Calculate the inputs and outputs of a community. Prints a summary by default and saves a list.
calculate_inputs(intro_comm)
```

`calculate_smin` *Function used inside the calc_smin function*

Description

Function used inside the calc_smin function

Usage

```
calculate_smin(SMIN = 1, usin, isSQR = TRUE)
```

Arguments

`SMIN`	The value of smin.
`usin`	The community used in the calculation
`isSQR`	Boolean: Should the rmax value be squared?

Value

The value of rmax for the community and given SMIN

`calc_smin` *Calculate the strength of stability as smin.*

Description

Calculate the strength of stability as smin.

Usage

```
calc_smin(usin)
```

Arguments

`usin`	The community for which to calculate smin

Details

Calculates the value of smin in each web using the metric used by Moore and de Ruiter.

Value

The values of smin.

Examples

```
# Calculate the minimum values of s for the introduction community.
calc_smin(intro_comm)
```

`can_mutfeed` *A function that identifies cannibalism and mutual feeding*

Description

A function that identifies cannibalism and mutual feeding

Usage

```
can_mutfeed(usin)
```

Arguments

`usin`	The community in which to identify cannibalism and mutual feeding

Value

The community with mutual feeding added to the properties database.

Examples

```
intro_comm_mod = intro_comm
intro_comm_mod$imat["Predator", "Predator"] = 1
can_mutfeed(intro_comm_mod)
```

---

checkcomm *A function check the community for errors before it is used in calculations.*

Description

A function check the community for errors before it is used in calculations.

Usage

```
checkcomm(usin, shuffleTL = FALSE, rmzeros = TRUE, verbose = TRUE)
```

Arguments

`usin`	The community to check.
`shuffleTL`	A Boolean stating whether the community should be sorted.
`rmzeros`	A Boolean determining whether trophic species with zero biomass should be removed from the community.
`verbose`	A Boolean. Do you want to see all the warnings?

Value

The checked community.

Examples

```
checkcomm(intro_comm)
```

---

checkeqm *Check the carbon flux equilibrium output by comana.*

Description

Check the carbon flux equilibrium output by comana.

Usage

```
checkeqm(cares, eqmtolerance = NA)
```

Arguments

`cares`	The output from comana.
`eqmtolerance`	A value used to set the equilibrium tolerance for the food web verification. If NA, the default value used by the function all.equal is used, which is approximately 1.5e-8.

Value

Boolean: Is the community at equilibrium with inputs to the first trophic level?

Examples

```
checkeqm(comana(intro_comm))
```

`Cijfcn` *A utility function to calculate the consumption rate of each species on all prey assuming a type I functional response.*

Description

A utility function to calculate the consumption rate of each species on all prey assuming a type I functional response.

Usage

```
Cijfcn(usin, shuffleTL = FALSE, rmzeros = TRUE)
```

Arguments

`usin`	The community on which feeding rate calculations are made.
`shuffleTL`	A Boolean stating whether the community should be sorted before consumption rates are calculated.
`rmzeros`	A Boolean determining whether trophic species with zero biomass should be removed from the community.

Value

A matrix of consumption rates with units set by the the biomass input units in biomass and time.

Examples

```
Cijfcn(intro_comm)
```

CNsim — *A function to simulate the dynamics over time wrapping getPARAMS and foodwebode*

Description

A function to simulate the dynamics over time wrapping getPARAMS and foodwebode

Usage

```
CNsim(
  usin,
  DIETLIMTS = NA,
  diet_correct = TRUE,
  Conly = FALSE,
  userdefinedinputs = NA,
  start_mod = NA,
  TIMES = 1:100,
  keepallnitrogen = TRUE,
  has_inorganic_nitrogen = FALSE,
  densitydependence = NA,
  inorganic_nitrogen_properties = list(INN = NA, q = NA, eqmN = NA),
  DETEXPT = NA,
  DETEXPTSTART = NA,
  rtn_only_state = TRUE
)
```

Arguments

`usin`	The community you want to simulate.
`DIETLIMTS`	The diet limits matrix for the stoichiometry correction (proportion of diet)?
`diet_correct`	Boolean: Does the organism correct it's diet?
`Conly`	Boolean: Is the model meant for carbon only?
`userdefinedinputs`	Do you want to input a user defined vector of input functions? If NA the input values that keep the system at equilibrium are calculated. If not, put in a vector of input rates for each node.
`start_mod`	A vector of modifications to the starting conditions, which default to the biomass vector in the community. This vector is multiplied by the biomass vector.
`TIMES`	The vector of times that you want to run the model. Defaults to 1 to 100 by 1.
`keepallnitrogen`	Boolean: Keep all the nitrogen pools in the model output? Will be set to FALSE if you are running a stability analysis using the function stability2().
`has_inorganic_nitrogen`	Boolean: Is there an inorganic nitrogen pool?

`densitydependence`	Which nodes have density dependence? NA default means none of them do. Should be a vector of 0 (no DD) and 1 (DD) for each node.
`inorganic_nitrogen_properties`	A list of state variables for the inorganic nitrogen pool (INN = inputs, q = per capita loss of N, eqmN = equilibrium N). Must include a value for two of the three variables and has the final one as NA.
`DETEXPT`	The pool that should be used for the detritus experiment by name or position in the vector usin$prop$ID.
`DETEXPTSTART`	The start of the detritus experiment. Defaults to 100.
`rtn_only_state`	Boolean: Do you want to only return the model state variables? If FALSE then the consumption rates and production efficiency at each time step are also returned.

Details

A function that simulates the food web over the user defined times. If you do not modify the starting state using start_mod or add a detritus experiment using DETEXPT, then the result will just be a flat line if the food web is stable.

Value

The output of the simulation.

See Also

The function uses `getPARAMS` and `foodwebode` to simulate the community.

Examples

```
# Basic example with a 5% reduction in predator biomass:
CNsim(intro_comm, start_mod = c(0.95, 1,1,1,1,1,1))

# Simulate a decomposition experiment:
CNsim(intro_comm, DETEXPT = c("Detritus1"), DETEXPTSTART = 100)
```

---

comana — *A function to calculate carbon and nitrogen fluxes in the food web.*

Description

A function to calculate carbon and nitrogen fluxes in the food web.

Usage

```
comana(
  usin,
  mkplot = FALSE,
  whattoplot = c("web", "Nmin", "Cmin"),
  showCN = FALSE,
  BOX.SIZE = 0.1,
  BOX.PROP = 0.3,
  BOX.CEX = 1,
  PLOT.CEX = 1,
  edgepos = c(0.1, 0.9),
  TCK = 0.05,
  shuffleTL = FALSE,
  prettynames = NA,
  fwdlwdcust = NULL,
  arrowlog = FALSE,
  arrowsizerange = c(0.1, 30),
  rmzeros = TRUE,
  eqmtolerance = NA
)
```

Arguments

`usin`	The community that you are analyzing: contains a matrix of interactions and a data frame of properties in a list.
`mkplot`	Boolean: Should the plots be output?
`whattoplot`	A vector of what to plot. Food web typology (web), nitrogen mineralization (Nmin), and/or carbon mineralization (Cmin).
`showCN`	Boolean: # Should the food web show the C:N ratio of each trophic species next to it's name in the food web plot?
`BOX.SIZE`	Size of boxes in the food web plot
`BOX.PROP`	Proportion of box length and width in the food web plot
`BOX.CEX`	Size of box text in the food web plot
`PLOT.CEX`	Size of plot
`edgepos`	Where to put the far left and far right boxes for trophic species in the food web plot (range = 0, 1).
`TCK`	The size of the ticks on the C.min and N.min plots.
`shuffleTL`	A Boolean stating whether the community should be sorted.
`prettynames`	Alternative names in order for the food web plot trophic species. Cannot be used if showCN = T.
`fwdlwdcust`	# A matrix of arrow line widths, same dimensions at the food web for plot customization.
`arrowlog`	Boolean: Should relative arrow widths in the food web plot be on a log scale?
`arrowsizerange`	The range of arrow sizes in the food web plot.

`rmzeros`	A Boolean determining whether trophic species with zero biomass should be removed from the community before analysis.
`eqmtolerance`	A value used to set the equilibrium tolerance for the food web verification. If NA, the default value used by the function all.equal is used, which is approximately 1.5e-8.

Value

A list of consumption rates, carbon mineralization, nitrogen mineralization, carbon and nitrogen consumption rates, and the modified community if zeros where removed or sorting occurred.

Examples

```
comana(intro_comm)
```

`comtrosp` *A function to combine trophic species.*

Description

A function to combine trophic species.

Usage

```
comtrosp(
  usin,
  selected = NA,
  deleteCOMBOcannibal = FALSE,
  allFEEDING1 = FALSE,
  newname = NA
)
```

Arguments

`usin`	The community were you are combining trophic species
`selected`	Select trophic species to combine, which ones do you want to combine (vector of names)? If left as NA, the two most similar trophic species are combined. Similarity is determined by shared feeding relationships.
`deleteCOMBOcannibal`	Boolean: Do you want to delete the cannibalism that may have been created by combining two trophic species (TRUE) or leave it in the model (FALSE)?
`allFEEDING1`	Boolean: Do you want to return all feeding preferences to 1 (TRUE), or would you like to set the feeding preferences of the newly combined trophic species as the biomass-weighted average of the old ones (FALSE)?
`newname`	The name you want to use for the combined trophic species. Default replaces combines the names of the original trophic species divided by "/".

Details

The function combines trophic species by merging them in both the community matrix (imat) and the properties database (prop). The user can select the two or more trophic species to be combined using the selected option. If this is left as the default (NA), then the two most similar trophic species are combined by comparing their feeding relationships.

Value

The new community with the seltf or most similar trophic species combined.

Examples

```
# Combine two trophic species that are the most similar.
comtrosp(intro_comm)

# Combine two selected trophic species.
comtrosp(intro_comm, selected = c("Orib1", "Predator"))

# Combine two selected trophic species
# Remove the cannibalism that is created
# Rescale all feeding preferences to 1.
comtrosp(intro_comm,
selected = c("Orib1", "Predator"),
deleteCOMBOcannibal = TRUE,
allFEEDING1 = TRUE)
```

correction_function *A function to correct stoichiometry dynamically*

Description

A function to correct stoichiometry dynamically

Usage

```
correction_function(
  biomasses,
  cij,
  CN,
  p,
  a,
  canIMM,
  dietlimits,
  diet_correct = TRUE,
  Conly = FALSE,
  Immobilizationlimit = Inf
)
```

Arguments

`biomasses`	A vector of biomasses.
`cij`	The consumption matrix from Cijfcn
`CN`	A vector of C:N ratios.
`p`	A vector of production efficiency.
`a`	A vector of assimilation efficiency.
`canIMM`	A Boolean vector of whether the nodes can immobilize nitrogen.
`dietlimits`	The diet limits matrix for the stoichiometry correction (proportion of diet)?
`diet_correct`	Boolean: Does the organism correct it's diet?
`Conly`	Boolean: Is the model meant for carbon only?
`Immobilizationlimit`	This is the limit of the amount of nitrogen the food web can immobilize nitrogen (NOT PLANTS). This will impact the calculations of inorganic nitrogen dynamics.

Details

This function takes inputs from the ODE and outputs corrected consumption rates. The key difference from 'corrstoich' is that the prey DO NOT correct their feeding rates to compensate for higher consumption from the predators, so the system can leave equilibrium if a diet shift occurs.

Value

Returns the consumption rates (FMAT) and production efficiencies (p).

`correct_diet` *A function to correct the diet of trophic species.*

Description

A function to correct the diet of trophic species.

Usage

```
correct_diet(usin, dietlimits = c(NA))
```

Arguments

`usin`	The input community in which to fix the diets.
`dietlimits`	# A matrix the same size as imat that gives the diet limits as a proportion of the total diet. All values must be between 0 and 1. Leaving it as NA sets the limits of all diet items to 1.

Value

The modified community with new diet preferences.

corrstoich *Correct stoichiometry*

Description

Correct stoichiometry

Usage

```
corrstoich(
  usin,
  forceProd = FALSE,
  dietlimits = c(NA),
  Immobilizationlimit = Inf
)
```

Arguments

usin	# Community in which to correct stoichiometry.
forceProd	Boolean: Should we force organisms to only change their production efficiency and not their diet?
dietlimits	A matrix the same size as imat that gives the diet limits as a proportion of the total diet. All values must be between 0 and 1. Leaving it as NA sets the limits of all diet items to 1.
Immobilizationlimit	Set a limit for the maximum rate of immobilization of inorganic nitrogen for the food web per time step.

Details

A function that corrects trophic species production efficiency and/or diet composition to balance carbon and nitrogen demands.

Value

The modified community with new production efficiencies and diets.

Examples

```
corrstoich(intro_comm)
# To force the correction to modify production efficiency only.

corrstoich(intro_comm, forceProd = TRUE)

# Set limits on the composition of the animal diet
DL = intro_comm$imat # copy over the feeding matrix to use for the diet limits.
# Oribatid 1 can only have up to 20% of its diet from each fungal species
DL["Orib1",] = c(0,0,0,0.2,0.2,1,1)
```

```
# Oribatid 2 can only have up to 10% of its diet from fungi 1
DL["Orib2",] = c(0,0,0,0.1,1,1,1)

# Run them with the limits:
corrstoich(intro_comm, dietlimits = DL)
```

---

decompexpt *Decomposition rates and effect of individual organisms*

Description

Decomposition rates and effect of individual organisms

Usage

```
decompexpt(usin, selected = NULL, overtime = NA)
```

Arguments

`usin`	The community in which we want to calculate decomposition rates.
`selected`	A vector of the species names for which you want to calculate direct and indirect effects on decomposition. Default NULL means all species in the food web. Useful for cases where removing certain species breaks the food web structure.
`overtime`	Do you want to return a decomposition trajectory overtime with and without all species? If so, set this to the number of time steps you want to see.

Details

The output list indicates the baseline decomposition rate (k; basedecomp) and then a table showing the effect on decomposition rates when the direct and indirect effects of individual trophic species are removed (decompeffects). Removing direct effects means setting consumption of that species to zero, while indirect effects occur when the community is recalculated without that species before calculating the decompostion rate (k). A positive number means the species increases decomposition. The column ID contains the node having the ffect, while DetritusID indicates the detritus pool whose rate is being reported. Direct effects are calculated as the decomposition rate in the full community minus decomposition rate without the species consumption of detritus, divided by the decomposition rate in the full community. Indirect effects are calculated as the difference in decomposition rate of the full community minus decomposition rate without the species and minus, divided by the decomposition rate with the species and minus the direct effect. A negative value here means that removing the species reduces decomposition more than any direct consumption effects it has on detritus. If ‘overtime‘ is a number, a table of the proportion of detritus over time is returned with and without all species. Column "Original" is the original community, while all successive columns show the removal of each trophic species.

Value

A list. The first item is the basic decomposition constants. The second is a table of decomposition constants for each detritus pool. The third only runs if asked for by ‘overtime‘ and is the predicted trajectory.

Examples

```
# Basic example for the introductory community:
decompexpt(intro_comm)
```

`deRuiter1994` *The soil food webs published for conventional (CON) and integrated (INT) management at Lovinkhoeve experimental farm.*

Description

The community contains 18 nodes in bacterial and fungal energy channels.

Usage

```
deRuiter1994
```

Format

A list of four communities each with a feeding matrix and properties dataframe. NOTE: You must select one of the communities from the list to use the package functions. For example: comana(deRuiter1994$INT) carries out calculations for the integrated field.

INT The field with integrated management, 0 to 10-cm.

INT10 The field with integrated management, 10 to 25-cm.

CON The field with conventional management, 0 to 10-cm.

CON10 The field with conventional management, 10 to 25-cm.

imat Within the community: The feeding matrix. Rows eat columns.

prop Within the community: The properties data frame containing node names (ID), assimilation efficiency (a), production efficiency(p), C:N ratio (CN), biomass (B), death rate (d), proportion of death cycled back to a detrital pool (DetritusRecycling), Booleans stating whether the node is detritus, plant, and can immobilize nitrogen, and a list of mutual predators. Biomass is in kilograms of carbon per hectare in the depth range noted above and turnover/death rate is in years.

Source

doi:10.1016/01678809(94)900442

`foodwebode` *A function to simulation the food webs away from equilibrium.*

Description

A function to simulation the food webs away from equilibrium.

Usage

```
foodwebode(t, y, pars)
```

Arguments

`t`	The ODE time.
`y`	The ODE simulation start.
`pars`	The ODE parameters.

Details

The food web model ode for simulating the model over time: requires `y` inputs and `pars` parameters from the `getPARAMS` function. The function can handle a detritus decomposition experiment as set up by either `getPARAMS` or `CNsim`. The only pools with nitrogen values are the detritus, because all others have fixed C:N ratios.

Value

The changes in each node biomass along with parameters and mineralization rates.

See Also

Use this function inside `CNsim`, `stability2`, and `calculate_inputs`. See the documentation for these functions.

`getPARAMS` *A function to get the parameters for a food web model.*

Description

A function to get the parameters for a food web model.

Usage

```
getPARAMS(
  usin,
  DIETLIMTS = NA,
  diet_correct = TRUE,
  Conly = FALSE,
  userdefinedinputs = NA,
  returnCNmin = FALSE,
  has_inorganic_nitrogen = FALSE,
  densitydependence = NA,
  inorganic_nitrogen_properties = list(INN = NA, q = NA, eqmN = NA),
  verbose = TRUE,
  Immbolizationlimit = Inf
)
```

Arguments

`usin`	The community you want to simulate.
`DIETLIMTS`	The diet limits matrix for the stoichiometry correction (proportion of diet)?
`diet_correct`	Boolean: Does the organism correct it's diet?
`Conly`	Boolean: Is the model meant for carbon only?
`userdefinedinputs`	Do you want to input a user defined vector of input functions? If NA the input values that keep the system at equilibrium are calculated. If not, put in a vector of input rates for each node.
`returnCNmin`	Boolean: Do you want to add Cmin and Nmin to the list of the results? Used internally in the package in `calculate_inputs`.
`has_inorganic_nitrogen`	Boolean: Is there an inorganic nitrogen pool?
`densitydependence`	Which nodes have density dependence? NA default means none of them do. Should be a vector of 0 (no DD) and 1 (DD) for each node.
`inorganic_nitrogen_properties`	A list of state variables for the inorganic nitrogen pool (INN = inputs, q = per capita loss of N, eqmN = equilibrium N). Must include a value for two of the three variables and has the final one as NA.
`verbose`	Boolean: Do you want extra warnings updates?
`Immbolizationlimit`	This is the limit of the amount of nitrogen the food web can immobilize nitrogen (NOT PLANTS). This will impact the calculations of inorganic nitrogen dynamics.

Details

A function to get the parameters of a food web model for simulation purposes. It does not correct stoichiometry, so the user must do this beforehand if they want.

Value

A list with two elements: (1) a vector of parameters to run the model away from equilibrium and (2) a vector of starting biomasses.

See Also

`CNsim`, `stability2`, and `calculate_inputs` for the application of the function and use of it's options.

Examples

```
# Basic call.
getPARAMS(intro_comm)
```

`Holtkamp2011` *The soil food webs published along a chronosequence in the Netherlands.*

Description

The community contains 21 nodes in bacterial, fungal, and root energy channels.

Usage

```
Holtkamp2011
```

Format

A list of four communities each with a feeding matrix and properties dataframe. NOTE: You must select one of the communities from the list to use the package functions. For example: comana(Holtkamp2011$Young) carries out calculations for the Young field.

Young The Young field.

Mid The mid-aged field.

Old The old field.

Heathland The heathland.

imat Within the community: The feeding matrix. Rows eat columns.

prop Within the community: The properties data frame containing node names (ID), assimilation efficiency (a), production efficiency(p), C:N ratio (CN), biomass (B), death rate (d), proportion of death cycled back to a detrital pool (DetritusRecycling), Booleans stating whether the node is detritus, plant, and can immobilize nitrogen, and a list of mutual predators. Biomass is in kilograms of carbon per hectare to 10-cm depth and turnover/death rate is in years.

Source

doi:10.1016/j.soilbio.2010.10.004

`Hunt1987` *The soil food web published for CPER*

Description

The community contains 17 nodes in bacterial, fungal, and root energy channels.

Usage

```
Hunt1987
```

Format

A community with a feeding matrix and properties dataframe:

imat The feeding matrix. Rows eat columns.

prop The properties data frame containing node names (ID), assimilation efficiency (a), production efficiency(p), C:N ratio (CN), biomass (B), death rate (d), proportion of death cycled back to a detrital pool (DetritusRecycling), Booleans stating whether the node is detritus, plant, and can immobilize nitrogen, and a list of mutual predators. Biomass is in kilograms of carbon per hectare and turnover/death rate is in years.

Source

doi:10.1007/BF00260580

`intro_comm` *A baseline community for examples*

Description

The community contains seven nodes generally meant to represent oribatid mites feeding on microorganisms and detritus.

Usage

```
intro_comm
```

Format

A community with a feeding matrix and properties dataframe:

imat The feeding matrix. Rows eat columns.

prop The properties data frame containing node names (ID), assimilation efficiency (a), production efficiency(p), C:N ratio (CN), biomass (B), death rate (d), proportion of death cycled back to a detrital pool (DetritusRecycling), Booleans stating whether the node is detritus, plant, and can immobilize nitrogen, and a list of mutual predators.

Source

Example not based on real empirical data.

`Jacobsindex` *Calculate Jacob's index.*

Description

Calculate Jacob's index.

Usage

```
Jacobsindex(usin)
```

Arguments

`usin`	The community for which to calculate Jacob's index

Details

Jacob's index is calculated for all predators with more than one prey item. A value of -1 indicates maximum avoidance, 0 indicates no preference, and 1 indicates maximum preference. This index is a more user friendly output than the values in imat, which can be difficult to interpret.

Value

A data frame with columns for predator, prey, and the value of Jacob's index.

Examples

```
# Calculate the minimum values of s for the introduction community.
Jacobsindex(intro_comm)
```

`Koltz2018` *The soil food web published for an Arctic Tundra site*

Description

The community contains 41 nodes in bacterial, fungal, and root energy channels. One node has zero biomass and is removed by the check_comm() function.

Usage

```
Koltz2018
```

Format

A community with a feeding matrix and properties dataframe:

imat The feeding matrix. Rows eat columns.

prop The properties data frame containing node names (ID), assimilation efficiency (a), production efficiency(p), C:N ratio (CN), biomass (B), death rate (d), proportion of death cycled back to a detrital pool (DetritusRecycling), Booleans stating whether the node is detritus, plant, and can immobilize nitrogen, and a list of mutual predators. Biomass is in milligrams of carbon per square meter and turnover/death rate is in years.

Source

doi:10.1007/s0030001722015

newnode *Add node to the community*

Description

Add node to the community

Usage

```
newnode(COMM, newname, prey = NA, predator = NA, newprops)
```

Arguments

`COMM`	The community to which to add nodes.
`newname`	The new node ID.
`prey`	A vector of prey preferences with names.
`predator`	A vector of predators and their preferences with name.
`newprops`	A vector of the new properties with the appropriate names.

Value

The community with the new node.

See Also

`removenodes`

Examples

```
# Add a node to the introductory community:
newnode(intro_comm, "NewNode",
prey = c(Detritus1 = 1),
predator = c(Predator = 2, Orib1 = 0.1),
newprops = c(d = 1, a = 0.1, p = 0.1,
B = 10, CN = 10, DetritusRecycling = 0,
isDetritus = 0, isPlant = 0, canIMM = 0))
```

`parameter_uncertainty` *Parameter uncertainty returns community with new parameters drawn from a distribution of choice*

Description

Parameter uncertainty returns community with new parameters drawn from a distribution of choice

Usage

```
parameter_uncertainty(
  usin,
  parameters = c("B"),
  replacetiny = 1e-06,
  distribution = "gamma",
  errormeasure = 0.2,
  errortype = "CV",
  fcntorun = "comana",
  replicates = 100,
  returnprops = FALSE,
  returnresults = TRUE,
  rejectnegconsump = TRUE,
  correctstoich = TRUE,
  verbose = TRUE
)
```

Arguments

`usin`	The community in which we want to calculate mineralization rates.
`parameters`	A vector of parameters you want to vary.
`replacetiny`	A number. All parameter draws less than this value are replaced with it to avoid numerical errors in the calculations. Set to zero if you want all values to be left as drawn. Default is 0.000001.
`distribution`	A single string or matrix for the distribution from which to draw the parameters. If it is a matrix is has rownames of web nodes matching usin and column names matching parameters. The acceptable options are gamma, normal, uniform.

`errormeasure`	A single value or matrix following the format of distribution recording the error. Value depends on errortype.
`errortype`	A single value or matrix following the format of distribution recording the error type. This can be "CV" for coefficient of variation, "Variance" for the variance, and "Min" for the minimum value. The latter can only be used when the distribution is uniform.
`fcntorun`	The function you want to run on the resulting communities and the result you want to return. Current options are comana, whomineralizes, CNsim, decompexpt. You can also include any of the outputs of comana or decompexpt to automatically subset the results to the vector of interest. For example, Cmin only returns carbon mineralization.
`replicates`	The number of replicate communities you want to create and analyze.
`returnprops`	Boolean. Do you want to return the communities with parameter values or just the results of the function? Only used if returnresults is TRUE.
`returnresults`	Boolean. Do you want to return the results of the function? If this is FALSE, the fcntorun is ignored and a list of communities with parameter draws is returned.
`rejectnegconsump`	Boolean. Should the draws reject communities with negative consumption rates?
`correctstoich`	Boolean. Do you want to correct the stoichiometry of the community before running the fcntorun? This does NOT correct the community stoichiometry returned in communitylist, so the user can see the original result without the correction applied.
`verbose`	Boolean. Do you want warning messages about the functionality?

Details

The results are always in a list. If returnprops = T, then the top lay is a list of length 2 with resultslist and communitylist attributes, otherwise only resultslist is returned. The communitylist has the communities with parameter draws in order. The resultslist has the results of the function indicated in fcntorun.

Value

A list of the results. See details.

Examples

```
# Basic example for the introductory community:
parameter_uncertainty(intro_comm)
```

`productionadj` *A function to fix production efficiency.*

Description

A function to fix production efficiency.

Usage

```
productionadj(usin, immobilizationlimit = Inf)
```

Arguments

`usin`	The input community to fix the production efficiency.
`immobilizationlimit`	Set a limit for the maximum rate of immobilization of inorganic nitrogen for the food web per time step.

Value

The modified community with new production efficiency.

`removenodes` *Remove nodes from community.*

Description

Remove nodes from community.

Usage

```
removenodes(COMM, toremove)
```

Arguments

`COMM`	The community from which to remove nodes.
`toremove`	A vector of nodes to remove using their names.

Value

The community without the removed nodes.

Examples

```
removenodes(intro_comm, c("Predator"))
```

`renamenode` *Rename a node in a community.*

Description

Rename a node in a community.

Usage

```
renamenode(COMM, oldname, newname)
```

Arguments

`COMM`	The community from which to remove nodes.
`oldname`	The node's old name
`newname`	The node's new name

Value

The community with the new name.

Examples

```
renamenode(intro_comm, oldname = "Predator", newname = "NewPredator")
```

`RESCALE` *A function to rescale a vector.*

Description

A function to rescale a vector.

Usage

```
RESCALE(invec, a = 0, b = 1)
```

Arguments

`invec`	The vector to re-scale.
`a`	The lower limit of the new scale.
`b`	The upper limit of the new scale.

Value

The scaled vector.

stability *Calculates the stability of the food web*

Description

Calculates the stability of the food web

Usage

```
stability(
  usin,
  correctstoich = TRUE,
  forceProd = FALSE,
  smin = 1,
  method = "Jacobian"
)
```

Arguments

usin	The community to calculate stability.
correctstoich	Boolean: Should stability be calculated after the stoichiometry is corrected with corrstoich?
forceProd	Boolean: Should production efficiency be the only way to correct stoichiometry?
smin	The value of smin in the Moorecobian.
method	One of two options: Jacobian or Moorecobian. The later uses the value of smin and adds density-dependence to the calculation at the strength of smin.

Value

Returns the Jacobian (or Moorecobian), eigenvalues, and the maximum eignvalue as a list.

See Also

calc_smin for the full use of the Moorecobian and stability2 for the estimation of stability using the functions from jacobian.full

Examples

```
# Basic stability calculation
stability(intro_comm)
```

stability2 — *A function to run the stability analysis using the numerical simulation of the Jacobain matrix.*

Description

A function to run the stability analysis using the numerical simulation of the Jacobain matrix.

Usage

```
stability2(
  usin,
  DIETLIMTS = NA,
  diet_correct = TRUE,
  Conly = FALSE,
  userdefinedinputs = NA,
  has_inorganic_nitrogen = FALSE,
  inorganic_nitrogen_properties = list(INN = NA, q = NA, eqmN = NA),
  forstabilityonly = TRUE,
  densitydependence = NA
)
```

Arguments

`usin`	The community for which to calculate stability.
`DIETLIMTS`	The diet limits matrix for the stoichiometry correction (proportion of diet)?
`diet_correct`	Boolean: Does the organism correct it's diet?
`Conly`	Boolean: Is the model meant for carbon only?
`userdefinedinputs`	Do you want to input a user defined vector of input functions? If NA the input values that keep the system at equilibrium are calculated. If not, put in a vector of input rates for each node.
`has_inorganic_nitrogen`	Boolean: Is there an inorganic nitrogen pool?
`inorganic_nitrogen_properties`	A list of state variables for the inorganic nitrogen pool (INN = inputs, q = per capita loss of N, eqmN = equilibrium N). Must include a value for two of the three variables and has the final one as NA.
`forstabilityonly`	Boolean: Do you only want to check the stability of the carbon equations and inorganic nitrogen? If FALSE then changes in detritus nitrogen are also included.
`densitydependence`	Which nodes have density dependence? NA default means none of them do. Should be a vector of 0 (no DD) and 1 (DD) for each node.

Details

The stability as defined by the Jacobian is estimated using the ordinary differential equations. Consequenlty, the parameters are retrieved using `getPARAMS` and then added to the function `jacobian.full` using the ODE defined by `foodwebode` This stability function allows the user to define density-dependence by node as present or absent. If you want to apply a uniform level of density-dependence use the function 'stability' and choose the option Moorecobain.

Value

The stability calculated by the Jacobian as a list of the Jacobian, eigenvalues, and the maximum eigenvalue.

See Also

`getPARAMS` and `foodwebode` for functions which are called internally and `jacobian.full` for the method used.

Examples

```
# Basic stability calculation:
stability2(intro_comm)

# Stability calculation with density-dependence for the animals:
stability2(intro_comm, densitydependence = c(1, 1, 1, 0, 0, 0, 0))
```

`TLcheddar` *Calculates the trophic level for each tropospecies*

Description

Calculates the trophic level for each tropospecies

Usage

```
TLcheddar(W)
```

Arguments

`W`	A matrix of trophic species. Rows eat columns.

Details

This function is a subset of code provided in the package Cheddar written by Lawrence Hudson, Dan Reuman and Rob Emerson. It is licensed under a BSD_2_clause, the text of which is provided as a comment in the function code. The original package can be found on CRAN or at https://github.com/quicklizard99/cheddar/

Value

A vector of trophic level assignments. The base of the food chain is 0.

Examples

```
TLcheddar(intro_comm$imat)
```

`TLsort` *Sorts the trophic levels from lowest to highest*

Description

Sorts the trophic levels from lowest to highest

Usage

```
TLsort(usin)
```

Arguments

`usin`	The community to be sorted.

Value

The community returned after sorting

Examples

```
TLsort(intro_comm)
```

`whomineralizes` *Direct and indirect contributions to mineralizations*

Description

Direct and indirect contributions to mineralizations

Usage

```
whomineralizes(usin, selected = NULL)
```

Arguments

`usin`	The community in which we want to calculate mineralization rates.
`selected`	A vector of names for which you want to calculate the direct and indirect effects. Default NULL means all of them. Useful for excluding nodes whose removal breaks the community (i.e., basal nodes)

Details

The results are labeled as follows with direct contributions calculated from the full food web and indirect contributions calculated from the food web without that node. Indirect contributions do not include the direct contribution (i.e., it is subtracted).

DirectC The direct contribution to carbon mineralization.

DirectN The direct contribution to nitrogen mineralization.

IndirectC The indirect contribution to carbon mineralization.

IndirectN The indirect contribution to nitrogen mineralization.

The indirect contributions are calculated as the total mineralization of the community with the trophic species minus the trophic species direct mineralization minus the total mineralization without the trophic species all divided by the total mineralizaiton with the trophic species.

Value

A table of node effects on mineralization rates.

Examples

```
# Basic example for the introductory community:
whomineralizes(intro_comm)
```

Index