Package ‘fluxweb’

July 7, 2025

Type Package

Title Estimate Energy Fluxes in Food Webs

Version 2.0.1

Description
Compute energy fluxes in trophic networks, from resources to their consumers, and can be applied to systems ranging from simple two-species interactions to highly complex food webs. It implements the approach described in Gauzens et al. (2017) <doi:10.1101/229450> to calculate energy fluxes, which are also used to calculate equilibrium stability.

License GPL (>= 2.0)

Depends stats

Encoding UTF-8

URL `https://www.biorxiv.org/content/10.1101/229450v4`

LazyData TRUE

RoxygenNote 7.3.2

NeedsCompilation no

Suggests testthat, R.rsp

VignetteBuilder R.rsp

Author Benoit Gauzens [cre, aut]

Maintainer Benoit Gauzens <`benoit.gauzens@gmail.com`>

Repository CRAN

Date/Publication 2025-07-07 13:20:02 UTC

Contents

`create.jacob` *Compute the Jacobian matrix*

Description

Compute the Jacobian matrix

Usage

```
create.jacob(
  val.mat,
  biomasses,
  efficiencies,
  metabolic.types,
  ef.level = "prey"
)
```

Arguments

`val.mat`	A matrix describing fluxes between species (usually a result of `fluxing` function).
`biomasses`	A vector of species biomasses.
`efficiencies`	A vector or an array of conversion efficiencies of species in the adjacency matrix.
`metabolic.types`	A vector containing information on species type (`"detritus"`, `"plant"` or `"animal"`)
`ef.level`	Set to `"prey"` if efficiencies are defined by prey, `"pred"` if they are a property of the predator, `link.specific` if they depend on both consumer and resource identity.

Value

A matrix representing the Jacobian matrix of the dynamical system associated with the fluxes from the `fluxing` function.

Author(s)

Benoit Gauzens, <`benoit.gauzens@gmail.com`>

Examples

```
# First compute species per unit biomass metabolic rates:
losses = 0.15 * groups.level$bodymasses^(-0.25)


val.mat = fluxing(groups.level$mat,
                  groups.level$biomasses,
                  losses,
                  groups.level$efficiencies,
                  bioms.pref = TRUE,
                  ef.level = "prey")

# define metabolic types

met.types = rep("animal", nrow(val.mat))
met.types[groups.level$efficiencies == 0.545] = "plant"
met.types[groups.level$efficiencies == 0.158] = "detritus"

create.jacob(val.mat,
               groups.level$biomasses,
               groups.level$efficiencies,
               met.types,
               ef.level = "prey")
```

---

fluxing *generate fluxes*

Description

Creates a valuated graph adjacency matrix from its binary version.

Usage

```
fluxing(
  mat,
  biomasses = NULL,
  losses,
  efficiencies,
  bioms.prefs = TRUE,
  bioms.losses = TRUE,
  ef.level = "prey"
)
```

Arguments

`mat`	Network adjacency matrix describing interactions among species. Interactions can be either binary or weighted.
`biomasses`	Vector of species biomasses.
`losses`	A vector or an array of species energy losses (excluding consumption).
`efficiencies`	A vector or an array of conversion efficiencies of species in the adjacency matrix. These values describe the proportion of consumed energy that is converted to biomass of the consumer.

`bioms.prefs`	Logical - if `TRUE`, consumer preferences are scaled according to species biomasses.
`bioms.losses`	Logical - if `TRUE`, losses are scaled with species biomasses.
`ef.level`	Set to `"prey"` if efficiencies are defined by prey, `"pred"` if they are a property of the predator.

Details

This function computes fluxes in food webs based on an equilibrium hypothesis: for each species, sum of ingoing fluxes (gains from predation) balances the sum of outgoing fluxes. Outgoing fluxes are defined by consumption and the `losses` argument. Usually `losses` relate to species metabolic rates and/or natural death rates. For each species `i`, sum of ingoing fluxes `F_i` is computed as:

$$F_i = \frac{1}{e_i}(L_i + \sum_j W_{ij}F_j) \quad if \quad \texttt{ef.level == "pred"}$$

$$F_i = \frac{L_i + \sum_j W_{ij}F_j}{\sum_j W_{ji}e_j} \quad if \quad \texttt{ef.level == "pred"}$$

`W` set the matrix of preferences estimated from `mat`, according to `bioms.prefs`. `L` is the vector depicting sum of losses (scaled or not by biomasses, accordingly to `bioms.losses`) and `e` is the vector of species efficiencies.

- `mat`: Either a binary or a valuated matrix can be used. A non zero value for mat[i,j] means that species i is consumed by species j. Matrix entries would assess predator preferences on its prey, thus providing a binary matrix assumes no preferences.
- `losses`: Express species energetic losses not related to consumption. Usually metabolic or death rates. When an array is provided, losses associated to each species correspond to line sums.
- `efficiencies`: Determines how efficient species are to convert energy (see `ef.level` for more details). Providing an array will assume values depending on both prey and predator identity.
- `bioms.pref`: If `TRUE`, preferences W_{ij} of predator j on prey i are scaled accordingly to species biomass using the following formula:

  $$W_{i,j} = \frac{mat[i,j] * biomasses[i]}{\sum_k mat[i,k] * biomasses[k]}$$

  If `FALSE`, a normalisation on column values is performed.
- `bioms.losses`: Set to true, function will assume that losses are defined per biomass unit. Thus, total losses will be thereafter multiplied by biomass values for each species.
- `ef.level`: If `"prey"` (resp `"pred"`), the total amount of energy that can be metabolized from a trophic link will be determined by prey (resp predator) identity. `"link.specific"` assumes that efficiencies are defined for each trophic interaction and implies `efficiencies` parameter to be a matrix.

Value

Returns an adjacency matrix where entries are the computed energy fluxes between consumer species and their respective resources.

Author(s)

Benoit gauzens, <`benoit.gauzens@gmail.com`>

Examples

```
# first compute species per unit biomass metabolic rates using the metabolic theory:
losses = 0.1 * species.level$bodymasses^(-0.25)

# call of the function:
fluxing(species.level$mat,
       species.level$biomasses,
       losses,
       species.level$efficiencies,
       bioms.pref = TRUE,
       ef.level = "prey")
```

`groups.level` *Aggregated version of the Food web of a soil network ecosystem and species general information (*`species.level`*).*

Description

This dataset contains the matrix describing trophic interactions between trophic groups of a soil food-web (Digel et al. 2014, Oikos) as well as some ecological information on these groups: biomasses, body masses and and species composition.

Format

a list of 5 elements:

mat the network adjacency matrix

biomasses groups total biomasses (g)

bodymasses group mean bodymasses of species (g)

efficiencies group species mean assimilation efficiencies

species.tgs groups' species composition `sensitivity` *sensitivity analysis*

Description

Assesses how sensitive the results from argument function are to variability of input parameter through coefficient of variation.

Usage

```
sensitivity(fun.name, param.name, var, n, full.output = FALSE, ...)
```

Arguments

`fun.name`	Function to analyse.
`param.name`	Parameter from `...` on which variation is applied.
`var`	Define the interval of uncertainty for the uniform law around `x` as `[x - x*var, x + x*var]`.
`n`	Number of replicates.
`full.output`	Logical, if `TRUE` all of `n` estimations of `fun.name` are returned. Only their mean otherwise.
`...`	Arguments to be passed to `fun.name`. Argument names must exactly match those of fun.name.

Details

At each replicate, a coefficient of variation is computed (relative to results obtained form `fun.name` without random variation). if `full.output` is `FALSE` (default) a list of two objects of the same type as the one produced by `fun.name` is returned, first element contains the mean coefficient of variation in comparison to non randomised inputs among all the replicates, second element contains the standard deviation of these coefficients of variation If `full.output` is `TRUE`, a list of size `n` with of objects containing the coefficients of variation is returned.

Argument for `...` should be passed with their names.

Value

a list of two elements of the same type as `param.name`: first element contains the mean coefficient of variation in comparison to non randomised inputs among all the replicates, second element contains the standard deviation of these coefficient of variation

Examples

```
# first compute species per unit biomass metabolic rates using the metabolic theory:
losses = 0.1 * species.level$bodymasses^(-0.25)


res = sensitivity(fluxing, "mat", 0.1, 5, full.output = TRUE,
                  mat = species.level$mat,
                  biomasses = species.level$biomasses,
                  losses = losses,
                  efficiencies = species.level$efficiencies)
res = sensitivity(fluxing, "efficiencies", 0.01, 50,
                  mat = species.level$mat,
                  biomasses = species.level$biomasses,
                  losses = losses,
                  efficiencies = species.level$efficiencies)

# growth rates of basal species
growth.rates = rep(NA, dim(species.level$mat)[1])
growth.rates[colSums(species.level$mat) == 0] = 0.5

val.mat = fluxing(species.level$mat, species.level$biomasses, losses, species.level$efficiencies)
```

`simple.case` *Food web of a soil network ecosystem and species general information.*

Description

This dataset correspond to the food web of a microcosm assembled from the Chesapeake Bay estuary (Lefcheck and Duffy 2010, Ecology)

Format

a list of 4 elements:

mat the network adjacency matrix

met.rate metabolic rates of species (J.h-1)

biomasses species biomasses (g)

efficiencies species assimilation efficiencies

names species names

species.level — *Food web of a soil network ecosystem and species general information.*

Description

This dataset contains the matrix describing trophic interactions from a deutsch soil food-web (Digel et al. 2014, Oikos) as well as some ecological information on species: biomasses, body masses and and species names.

Format

a list of 5 elements:

mat the network adjacency matrix

biomasses species biomasses (g)

bodymasses species bodymasses (g)

efficiencies species assimilation efficiencies

names species names

stability.value — *Estimates network stability*

Description

Computes resilience of the system through Jacobian matrix eigenvalues.

Usage

```
stability.value(
  val.mat,
  biomasses,
  efficiencies,
  metabolic.types,
  ef.level = "prey",
  full.output = FALSE
)
```

Arguments

`val.mat`	A matrix describing fluxes between species (usually a result of `fluxing` function).
`biomasses`	A vector of species biomasses.

`efficiencies`	A vector or an array of conversion efficiencies of species in the adjacency matrix. These values describe the proportion of consumed energy that is converted to biomass of the consumer.
`metabolic.types`	A vector containing information on species type (`"detritus"`, `"plant"` or `"animal"`)
`ef.level`	Set to `"prey"` if efficiencies are defined by prey, `"pred"` if they are a property of the predator.
`full.output`	Logical, if `TRUE` function return supplementary informations.

Details

- `efficiencies`: Determines how efficient species are to convert energy (see `ef.level` for more details). Providing an array will assume values depending on both prey and predator identity.
- `ef.level`: If `"prey"` (resp `"pred"`), the total amount of energy that can be metabolized from a trophic link will be determined by prey (resp pred) identity. `"link.specific"` assumes that efficiencies are defined for each trophic interaction and implies `efficiencies` parameter to be a matrix
- `full.output`: If `TRUE`, function result is a list of eigenvalues and eigenvectors of the Jacobian matrix.

Value

Maximum eigenvalue of the Jacobian matrix of a Lotka Voltera like system of equations. If full.output, Jacobian eigenvalues and eigenvectors are returned.

Author(s)

Benoit Gauzens, <`benoit.gauzens@gmail.com`>

Examples

```
losses = 0.15 * groups.level$bodymasses^(-0.25)
# define metbolic types:
met.types = rep('animal', length(losses))
met.types[groups.level$efficiencies == 0.545] = 'plant'

val.mat = fluxing(groups.level$mat,
                  groups.level$biomasses,
                  losses,
                  groups.level$efficiencies,
                  bioms.pref = TRUE,
                  ef.level = "pred")

stability.value(val.mat,
                groups.level$biomasses,
                groups.level$efficiencies,
                metabolic.types = met.types,
                ef.level = "pred")
```

Index