Package ‘fluxweb’

December 1, 2017

Type Package

Title Estimate energy fluxes in food webs

Version 0.1.0

Author Benoit Gauzens

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

Description
fluxweb is a package to compute energy fluxes 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. (Hopefully published ahead of time) to calculate energy fluxes, which are also used to calculate equilibrium stability.

License GPL (>=2.0)

Depends stats

LazyData TRUE

RoxygenNote 6.0.1

NeedsCompilation no

R topics documented:

`fluxweb-package` *The fluxweb package*

Description

the new fancy package fluxweb that fluxes webs

Author(s)

Benoit Gauzens

`fluxing` *generate fluxes*

Description

create 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 predation).
`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 efficiences are defined by prey, `"pred"` if they are a property of the predator.

Details

This function compute 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 predation and `losses` argument. Ususally `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`, accordingly 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 assume no preferences.
- `losses`: express species energetic losses not related to predation. 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 unsing the following formula:

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

  If `FALSE`, a mormalisation 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 metabolised from a trophic link will be determined by prey (resp pred) identity. `"link.specific"` assumes that efficiencies are defined for each trophic interaction and ask `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.le
```

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

Description

This dataset contains the matrix describing trophic interactions between trophic groups of a deutsch soil food-web (reference) as well as some ecological information on These groups: biomasses, body masses and and species composition.

Format

: a list of 4 elements:

mat the network adjacency matrix

biomasses groups total biomasses (g)

bodymasses group mean bodyamasses of species (g)

species.tgs groups' species composition

`make.stability` *making network stability*

Description

Find the smallest multiplicator of a variable from losses insuring system stability - !Version under devlopment!

Usage

```
make.stability(val.mat, biomasses, losses, efficiencies, growth.rate,
  losses.scale = NULL, bioms.prefs = TRUE, bioms.losses = TRUE,
  ef.level = "prey", interval = c(1e-12, 1), ...)
```

Arguments

`val.mat`	A matrix describing fluxes between species (usually a result of `fluxing` function).
`biomasses`	A vector of species biomasses.
`losses`	A vector or an array of species energy losses (excluding predation).
`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.
`growth.rate`	A vector defining growth rate of basal species.
`losses.scale`	Defines a Column from `losses` mulitplicator value will apply to. (default `NULL` if multiplicator independant of losses).
`bioms.prefs`	Logical, if `TRUE` (default) preferences are scaled accordingly to species biomasses.
`bioms.losses`	Logical, if `TRUE` (default) losses are scaled with biomass.
`ef.level`	Set to `"prey"` if efficiences are defined by prey, `"pred"` if they are a property of the predator.
`interval`	Search interval for returned value.
`...`	Optional parameters for function `uniroot`

Details

The function assume a monotonous increase of stability with multiplicator value. Solution is estimated from the `uniroot` function, and stability using the `fluxing` function Thus, accordingly to `uniroot` solving criteria, if stability values at the two extremu parts of the interval are of same sign, an error is raised.

Behavior of the multiplicative term depends on the type of losses:

- `losses.scale = NULL` and `is.vector(losses)`: multiplicator will be applied to the `losses` vector.
- `losses.scale = NULL` and `is.matrix(losses)`: multiplicator will be independant of any columns from `losses`.
- `losses.scale = FALSE` : multiplicator always independant of losses.
- other values: should refer to an element of losses.

  This is an ugly part, but I don't see how to do better right now. Any ideas?

Value

A list from `uniroot` function.

See Also

`uniroot` for root estimate and `stability.value` for assessing system stability.

Examples

```
losses = 0.15 * groups.level$bodymasses^(-0.25)

# growth rates of basal sppecies
growth.rates = rep(0.5, length(groups.level$biomasses[colSums(groups.level$mat) == 0]))

val.mat = fluxing(groups.level$mat, groups.level$biomasses, losses, groups.level$efficiencies, bioms.pref = TRUE
make.stability(val.mat, groups.level$biomasses, losses, groups.level$efficiencies, growth.rates, ef.level = "pre
```

`sensitivity` *sensitivity analysis*

Description

Assess how sensitive are results from argument function 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 wich variation is applied.
`var`	Define the interval of incertainty 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 (relatively to results obtained form `fun.name` without random variation). if `full.output` is `FALSE` (default) an object of the same type as the one produced by `fun.name` is returned, containing all of variation coefficients. If `full.output` is `TRUE`, a list of size `n` with of objects containing variation coeficient is returned.

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

Value

mean coefficient of variation in comparison to non randomised inputs among all the replicates

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$mat, biomasses = species.level$
res = sensitivity(fluxing, "efficiencies", 0.01, 50, mat = species.level$mat, biomasses = species.level$biomasse

# growth rates of basal sppecies
growth.rate = rep(0.5, length(species.level$biomasses[colSums(species.level$mat) == 0]))

val.mat = fluxing(species.level$mat, species.level$biomasses, losses, species.level$efficiencies)
#sensitivity(stability.value, "efficiencies", 0.01, 50, val.mat = val.mat, biomasses = species.level$biomasses,


cvs = c()
for (var in seq(0, 0.6, 0.05)){
 cvs = c(cvs, sensitivity(stability.value, "mat", var, 50, val.mat = val.mat, biomasses = species.level$biomasses
}

plot(abs(cvs) ~ seq(0, 0.6, 0.05))
```

`species.level` *Food web of a soil network ecosystem and species general informations.*

Description

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

Format

: a list of 4 elements:

mat the network adjacency matrix

biomasses species biomasses (g)

bodymasses species bodyamasses (g)

names species names

stability.value *estimate network stability*

Description

compute resiliance of the system through jacobian eigenvalues

Usage

```
stability.value(val.mat, biomasses, losses, efficiencies, growth.rate,
  bioms.prefs = TRUE, bioms.losses = TRUE, 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.
`losses`	A vector or an array of species energy losses (excluding predation).
`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.
`growth.rate`	A vector defining growth rate of basal species.
`bioms.prefs`	Logical, if `TRUE` (default) preferences are scaled accordingly to species biomasses.
`bioms.losses`	Logical, if `TRUE` (default) losses are scaled with biomass.
`ef.level`	Set to `"prey"` if efficiences are defined by prey, `"pred"` if they are a property of the predator.
`full.output`	Logical, if `TRUE` function return supplementary informations

Details

- `losses`: express species energetic losses not related to predation. 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
- `growth.rate`: Growth rates of basal species defined in `growth.rate` should appear in the same order as in other arguments. For example the second value specified in `growth.rate` should set the groth rate of the second basal species found in `biomasses`. IS THAT CLEAR...
- `bioms.pref`: If `TRUE`, preferences w_{ij} of predator j on prey i are scaled accordingly to species biomass unsing the following formula:

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

- `bioms.losses`: If `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 metabolised from a trophic link will be determined by prey (resp pred) identity. `"link.specific"` assumes that efficiencies are defined for each trophic interaction and ask `efficiencies` parameter to be a matrix
- `full.output`: If `TRUE`, function result is a list of eigenvalues and eigenvectors of the jacobian matrix. (return also the jacobian?)

Value

maximum eigenvalue of the jacobian matrix of a lotka Voltera like system of equation 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)

# growth rates of basal sppecies
growth.rates = rep(0.5, length(groups.level$biomasses[colSums(groups.level$mat) == 0]))

val.mat = fluxing(groups.level$mat, groups.level$biomasses, losses, groups.level$efficiencies, bioms.pref = TRUE
stability.value(val.mat, groups.level$biomasses, losses, groups.level$efficiencies, growth.rates, ef.level = "pr
```

Index