# Package ‘enaR’

February 19, 2015

**Type**  Package  
**Title**  Tools for Ecological Network Analysis (ena)  
**Version**  2.8.1  
**Date**  2015-01-14  
**Author**  M.K. Lau, S.R. Borrett, D.E. Hines, P. Singh  
**Maintainer**  Matthew K. Lau <enaR.maintainer@gmail.com>  
**Description**  Functions for the analysis of ecological networks  
**Depends**  R (>= 2.10), MASS, stringr, sna, network, gdata  
**Suggests**  codetools, igraph, R.rsp  
**VignetteBuilder**  R.rsp  
**License**  GPL-3  
**URL**  https://github.com/SEELab/enaR  
**NeedsCompilation**  no  
**Repository**  CRAN  
**Date/Publication**  2015-01-14 19:32:22

R topics documented:

<table>
<thead>
<tr>
<th>Topic</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>enaR-package</td>
<td>2</td>
</tr>
<tr>
<td>as.extended</td>
<td>3</td>
</tr>
<tr>
<td>bal</td>
<td>4</td>
</tr>
<tr>
<td>balance</td>
<td>5</td>
</tr>
<tr>
<td>bcratio</td>
<td>6</td>
</tr>
<tr>
<td>bgcModels</td>
<td>6</td>
</tr>
<tr>
<td>cycliv</td>
<td>7</td>
</tr>
<tr>
<td>eigenCentrality</td>
<td>8</td>
</tr>
<tr>
<td>enaAll</td>
<td>9</td>
</tr>
<tr>
<td>enaAscendency</td>
<td>10</td>
</tr>
<tr>
<td>enaControl</td>
<td>11</td>
</tr>
<tr>
<td>enaCycle</td>
<td>12</td>
</tr>
<tr>
<td>enaEnviron</td>
<td>14</td>
</tr>
</tbody>
</table>
### Description

This package compiles functions for the analysis of ecological networks, building on tools previously developed in the MatLab language (Fath and Borrett 2006) with multiple additions of functionality.
as.extended

Details

Package: enaR
Type: Package
Version: 2.6
Date: 2014-04-30
License: GPL-2

Author(s)

Authors: Stuart R. Borrett (UNCW) and Matthew K. Lau (NAU) Maintainer: Matthew K. Lau
<enaR.maintainer@gmail.com>

References


See Also

network

as.extended Create an Extended Format Matrix

Description

Converts a network object to the extended format of Allesina and Bondavalli (2003).

Usage

as.extended(x, zero.na = TRUE)

Arguments

x A network object.
zero.na Logical: should NA's be replaced with zeros?

Details

Used in the balance function.

Value

Returns an extended format matrix.
Author(s)

Matthew K. Lau (mkl48@nau.edu) Stuart R. Borrett (borretts@uncw.edu)

References


See Also

balance

Examples

data(troModels)
as.extended(troModels[[6]])

bal 

Subfunction for Balancing by Either Inputs or Outputs

Description

Dependency for the balance function.

Usage

bal(T.star = "matrix", method = c("input", "output"))

Arguments

T.star Extended, unbalanced matrix.
method Balance by inputs or outputs.

Value

Returns an extended matrix for balancing by inputs or outputs.

Author(s)

Matthew K. Lau (mkl48@nau.edu) Stuart R. Borrett (borretts@uncw.edu)

References


See Also

balance
balance

Balance Flow Network Models

Description

Applies the methods of Allesina and Bondavalli (2003) for balancing flow network models.

Usage

```r
balance(x, method=c('AVG2', 'AVG', 'IO', 'OI', 'I', 'O'), tol=5)
```

Arguments

- `x` A network object.
- `method` Methods for model balancing, see Allesina and Bondavalli (2003).
- `tol` Percent error tolerance used in the steady state check prior to balancing.

Value

Returns a network object with a balanced flow network model.

Author(s)

Matthew K. Lau (mkl48@nau.edu) Stuart R. Borrett (borretts@uncw.edu)

References


See Also

`bal`

Examples

```r
data(troModels)
balance(troModels[[6]])
```
bcratio

Calculates the Ratio of Positive to Negative Elements in a Network

Description
Dependent function for the enaUtility function.

Usage
bcratio(x = "matrix")

Arguments
x
A matrix of flow values.

Value
Returns the ratio of positive to negative elements in the flow matrix.

Author(s)
Stuart R. Borrett (borretts@uncw.edu)

References

See Also
enaUtility

bgcModels

Bigeochemical Cycling Models

Description
A set of 43 biogeochemical cycling models compiled by the SEE Lab at UNCW.

Usage
data(bgcModels)

References
Description

Performs the full cycle analysis on the living subset of the network based on the algorithm described in Ulanowicz (1983) and implemented in NETWRK 4.2b. It returns data.frames with details of the simple cycles and nexus, vectors of Cycle distributions and Normalized distribution and matrices of Residual Flows and Aggregated Cycles.

Usage

cycliv(x)

Arguments

x a network object. This includes all weighted flows into and out of each node. It must also include the "Living" vector that identifies the living (TRUE/FALSE) status of each node. Also, non-living nodes must be placed at the end of the node vector. The function netOrder can be used to reorder the network for this.

Value

Table.cycle data.frame that presents the details of the simple cycles in the network. It contains "CYCLE" the cycle number, "NEXUS" the nexus number corresponding to the cycle, "NODES" the nodes constituting the cycle.

Table.nexus data.frame that presents the different nexuses characterized by their corresponding weak arcs. It contains "NEXUS" the nexus number, "CYCLES" the number of simple cycles present in that Nexus, "W.arc.From" the starting node of the corresponding weak arc, "W.arc.To" the ending node of the corresponding weak arc and "W.arc.Flow" the flow through that weak arc.

CycleDist vector of the Cycle Distribution that gives the flow which is cycling in loops of different sizes.

NormDist vector of the Normalized Distribution i.e. the Cycle Distribution normalized by the Total System Throughput for the network.

ResidualFlows matrix of the straight-through (acyclic) flows in the network.

AggregatedCycles matrix of the Aggregated Biogeochemical Cycles in the network.

ns vector of the full cycle analysis based network statistics. These include "NCYCS" the number of simple cycles identified in the network, "NNEX" the number of the disjoint cycles or number of Nexuses detected in the network and "CI" the cycling index of the network.
**Note**

This function uses the same mechanism for analysis as used in the enaCycle function but is restricted to the living nodes only.

Also, similar to the enaCycle function, if the number of cycles in a nexus is more than 50, the "Table.cycle" has a blank line after 50 cycles followed by the cycles for the next nexus.

The analysis requires all the non-living nodes to be placed at the end in the network object.

**Author(s)**

Pawandeep Singh (pawan.s@live.co.uk)

**References**


**See Also**

enaTroAgg, enaCycle, netOrder

**Examples**

```r
data(troModels)
liv.cyc <- cycliv(troModels[[6]])
attributes(liv.cyc)
```

---

**eigenCentrality**  
*Calculates the Eigen Centrality of a Network*

**Description**

Calculates the eigen centrality of a network.

**Usage**

`eigenCentrality(x='matrix')`

**Arguments**

- `x` A matrix defining a network graph.
enaAll

Value

Returns the eigen based centrality of the network.

Author(s)

Stuart R. Borrett (borretts@uncw.edu) Matthew K. Lau (mkl48@nau.edu)

References


enaAll Conduct All Major ENA

Description

Conducts all major ENA with default settings and returns the output as a named list.

Usage

enaAll(x = "network object")

Arguments

x A network object.

Value

<table>
<thead>
<tr>
<th>ascendency</th>
<th>enaAscendency</th>
</tr>
</thead>
<tbody>
<tr>
<td>control</td>
<td>enaControl</td>
</tr>
<tr>
<td>environ</td>
<td>enaEnviron</td>
</tr>
<tr>
<td>flow</td>
<td>enaFlow</td>
</tr>
<tr>
<td>mti</td>
<td>enaMTI</td>
</tr>
<tr>
<td>storage</td>
<td>enaStorage</td>
</tr>
<tr>
<td>structure</td>
<td>enaStructure</td>
</tr>
<tr>
<td>utility</td>
<td>enaUtility with eigen.check=FALSE</td>
</tr>
</tbody>
</table>

Author(s)

Matthew K. Lau (mkl48@nau.edu) Stuart R. Borrett (borretts@uncw.edu)

See Also

enaAscendency, enaControl, enaEnviron, enaFlow, enaMTI, enaStorage, enaUtility
Examples

data(troModels)
output = enaAll(troModels[[6]])
names(output)

Description

Calculates the average mutual information (AMI), ascendency, overhead, and capacity of input-output networks. It also returns the ratios of ascendency and overhead to capacity. These metrics describe the organization of flow in an ecological network (Ulanowicz 1997).

Usage

enaAscendency(x = "network object")

Arguments

x A network object.

Value

- **AMI**: Returns the Average Mutual Information (AMI) in a network. AMI provides a measure of the constraints placed on a given piece of energy matter moving through a network (Patricio et al. 2006)
- **ASC**: Returns the ascendency of a network. Ascendency is a scaled form of AMI relative to the total system throughput (Ulanowicz 1997; 2004). Total system throughput is the sum of all activity in a network (Kay et al. 1989).
- **OH**: Returns the overhead of a network. Overhead is the proportion of the capacity in a network that is not used as ascendency (Ulanowicz 2004).
- **CAP**: Returns the capacity of a network. Capacity is defined as the sum of ascendency and overhead (Ulanowicz 2004).
- **ACS, CAP**: Returns the proportion of capacity used by ascendency.
- **OH, CAP**: Returns the proportion of capacity used by overhead.
- **robustness**: Returns the robustness of the network.
- **ELD**: Returns the Effective Link Density of the network(c) (Ulanowicz et al. 2014).
- **TD**: Returns the Trophic Depth of the network(r) (Ulanowicz et al. 2014).

Note

This and other Ulanowicz school functions require that export and respiration components of output be separately quantified.
enacControl

Author(s)

David E. Hines (deh9951@uncw.edu) Matthew K. Lau (mkl48@nau.edu)

References

Study Series. Springer-Verlag, Berlin.

Patrico, J., Ulanowicz, R.E., Pardal, M.A., Marques J.C., 2004. Ascendency as an ecological indi-

York.

complexity: insights from ecological network analysis

See Also

read.scor, read.wand, enaStorage, enaUtility

Examples

data(troModels)
enaAscendency(troModels[[6]])

---

enacControl

Control Analyses of Ecological Networks

Description

Analyses for analyzing the control amongst the nodes in ecological networks.

Usage

enacControl(x, zero.na = TRUE, balance.override = FALSE)

Arguments

x A network object.
zero.na Makes undefined (NA) values zero.
balance.override Turns off balancing and checks of network balance.
**Value**

- **CN**: Control matrix using flow values.
- **CQ**: Control matrix using storage values.
- **CR**: Schramski Control Ratio Matrix
- **CD**: Schramski Control Difference Matrix
- **sc**: Schramski System Control vector

**Author(s)**

Matthew K. Lau (mkl48@nau.edu) Stuart R. Borrett (borretts@uncw.edu) Pawandeep Singh (singhp@uncw.edu)

**References**


**See Also**

- enaStorage

**Examples**

```r
data(troModels)
enacControl(troModels[[6]])
```

---

**Description**

It performs the full cycle analysis on the network based on the algorithm described in Ulanowicz (1983) and implemented in NETWRK 4.2b. It returns data.frames with details of the simple cycles and nexus, vectors of Cycle distributions and Normalized distribution and matrices of Residual Flows and Aggregated Cycles.

**Usage**

```r
enaCycle(x)
```
**Arguments**

x  
a network object. This includes all weighted flows into and out of each node.

**Value**

*Table.cycle*  
data.frame that presents the details of the simple cycles in the network. It contains "CYCLE" the cycle number, "NEXUS" the nexus number corresponding to the cycle, "NODES" the nodes constituting the cycle

*Table.nexus*  
data.frame that presents the different nexuses characterized by their corresponding weak arcs. It contains "NEXUS" the nexus number, "CYCLES" the number of simple cycles present in that Nexus, "W.arc.From" the starting node of the corresponding weak arc, "W.arc.To" the ending node of the corresponding weak arc and "W.arc.Flow" the flow through that weak arc

*CycleDist*  
vector of the Cycle Distribution that gives the flow cycling in loops of different sizes

*NormDist*  
vector of the Normalized Distribution i.e. the Cycle Distribution normalized by the Total System Throughput of the system

*ResidualFlows*  
matrix of the straight-through (acyclic) flows in the network

*AggregatedCycles*  
matrix of the Aggregated Biogeochemical Cycles in the network

*ns*  
vector of the full cycle analysis based network statistics. These include "NCYCS" the number of simple cycles identified in the network, "NNEX" the number of the disjoint cycles or number of Nexuses detected in the network and "CI" the cycling index of the network.

**Note**

The "NODES" in "Table.cycle" are arranged such that the weak arc for the nexus is the arc between the first two nodes of the cycle.

This function uses the backtracking procedure for the identification of simple cycles, which are cycles that cross a node only once except the starting node. The backtracking process is a depth-first search algorithm.

In the data.frame "Table.cycle", if the number of cycles in a nexus is more than 50, then a blank line is displayed after 50 cycles of the nexus, followed by the cycles of the next nexus.

The results of the analysis of Feeding Cycles can be obtained as a byproduct of the enaTroAgg function that analyzes the trophic dynamics of a network.

At every multiple of 5000 cycles in a nexus, the program prints an indication for the user to know that it is still running.

**Author(s)**

Pawandeep Singh (pawan.s@live.co.uk)
References


See Also

enaTroAgg

Examples

```r
data(troModels)
cyc6 <- enaCycle(troModels[[6]])
attributes(cyc6)
```

Description

Calculates the environs for an ecological network.

Usage

```r
enaEnviron(x,input=TRUE,output=TRUE,type='unit',err.tol=1e-10,balance.override=FALSE)
```

Arguments

- `x` A network object.
- `input` Should the input environ be calculated?
- `output` Should the output environ be calculated?
- `type` Specifies the type of environs ("unit" or "realized") to be calculated.
- `err.tol` Error threshold for numerical error fluctuations in flows. Values below err.tol will be set to zero.
- `balance.override` Logical specifying whether (TRUE) or not (FALSE) the model needs to be balanced prior to calculations. If TRUE and the model is not balanced, environs will not be calculated.

Value

The function returns the input, output or both environs depending upon which were requested.
**enaFlow**

**Author(s)**

Stuart R. Borrett (borretts@uncw.edu) Matthew K. Lau (mkl48@nau.edu)

**References**


**Examples**

```r
data(troModels)
environ(troModels[[6]])
```

---

**enaFlow**

*Flow Analyses of Ecological Networks*

**Description**

Performs the primary throughflow analysis developed for input-output systems. It returns a vector of throughflows, the input and output oriented matrices for "direct flow intensities" and "integral flow intensities", and a set of flow based network statistics.

**Usage**

```r
enaFlow(x, zero.na = TRUE, balance.override = FALSE)
```

**Arguments**

- `x`: a network object. This includes all weighted flows into and out of each node.
- `zero.na`: LOGICAL: should NA values be converted to zeros.
- `balance.override`: Flow analysis assumes the network model is at steady-state (inputs = outputs). Setting balance.override = TRUE allows the function to be run on unbalanced models.

**Value**

- `T`: vector of node throughflows - total amount of energy-matter flowing into or out of each node
- `G`: matrix of the output oriented direct flow intensities
- `GP`: matrix of the input oriented direct flow intensities
- `N`: matrix of the output oriented integral (boundary+direct+indirect) flow intensities
- `NP`: matrix of the input oriented integral flow intensities
vector of flow based network statistics. These include "Boundary" the total input into or output from the system, "TST" the total system throughput, "TSTp" total system throughput, "APL" is the network aggradation TST/Boundary which is also called average path length, "FCI" (Finn Cycling Index) is a metric of the amount of cycling in a system, "BFI" is the boundary flow intensity Boundary/TST, "DFI" is the direct flow intensity Direct/TST, "IFI" is the indirect flow intensity Indirect/TST, "ID.F" is the realized indirect to direct flow intensity, "ID.F.I" is the input idealized indirect flow intensity, "id.F.O" is the output idealized indirect flow intensity, "HMG.I" is the input network homogenization, "HMG.O" is the output network homogenization, "AMP.I" is the strong measure of input network amplification, "AMP.O" is the strong measure of output network amplification, "mode0.F" is the boundary flow - flow that reaches a compartment from across the system boundary, "mode1.F" is internal first passage flow, "mode2.F" is cycled flow, "mode3.F" is the dissipative equivalent to mode2, and "mode4.F" is the dissipative equivalent of mode0.

Author(s)

Matthew K. Lau (mkl48@nau.edu) Stuart R. Borrett (borretts@uncw.edu)

References


See Also

read.scor, read.wand, enaStorage, enaUtility

Examples

data(troModels)
F = enaFlow(troModels[[6]])  # completes the full analysis
F$ns  # returns just the network statistics
**enaModelInfo**

---

**Ecosystem Model Information**

---

**Description**

Model information for the set of ecosystem models compiled by the SEE Lab at UNCW.

**Usage**

```r
data(enaModelInfo)
```

**References**


---

**enaModels**

---

**Ecosystem Models**

---

**Description**

A set of ecosystem models compiled by the SEE Lab at UNCW.

**Usage**

```r
data(enaModels)
```

**References**

enaMTI  

*Mixed Trophic Impacts (MTI) Analysis*

**Description**

Calculates the Mixed Trophic Impacts of one species on another in the given ecosystem model following the algorithm of Ulanowicz and Puccia (1990). This considers both the direct and indirect trophic impacts.

**Usage**

`enaMTI(x,eigen.check=TRUE,zero.na=TRUE, balance.override=FALSE)`

**Arguments**

- **x**: a network object. This includes all weighted flows into and out of each node. It must also include the "Living" vector that identifies the living (TRUE/FALSE) status of each node.

- **zero.na**: A logical parameter that specifies if NAs generated in the analysis should be reset to zero. The default is TRUE.

- **balance.override**: Mixed Trophic Impacts analysis builds on flow analysis and thus assumes the network model is at steady-state (inputs = outputs). Setting `balance.override = TRUE` allows the function to be run on unbalanced models, though this is unadvised.

- **eigen.check**: LOGICAL: should the dominant eigenvalue be checked? By default, the function will not return utility values if the eigenvalue is larger than one; however, if `eigen.check` is set to FALSE, then the function will be applied regardless of the mathematic concern.

**Value**

- **G**: output-oriented direct flow intensity matrix as in enaFlow, except oriented from row to column.

- **FP**: input-oriented direct flow intensity matrix similar to enaFlow; however, the calculation exclude respiration losses from the throughflow in the denominator to focus on NET production. Also, if the receiver compartment is not living, the flux intensity is set to zero.

- **Q**: direct net trophic impacts (G-t(FP)).

- **M**: Total (direct and indirect) tropic impacts of compartment i on j.
Note

This and other Ulanowicz school functions require that export and respiration components of output be separately quantified.

This analysis is similar in concept to the ENA Utility analysis.

With regard to the eigen.check argument, like enaFlow, enaStorage and enaUtility, this analysis considers the trophic impact propagated over path lengths ranging for zero to infinity. For the analysis to work properly, the path sequence must converge. This function checks to see if the path sequence is convergent by finding the dominant eigenvalue of the direct matrix. If this eigenvalue is less than 1, the sequence is convergent and the analysis can be applied; if the dominant eigenvalue is greater than one, then the anlysis cannot be applied.

Author(s)

Stuart R. Borrett (borrett@uncw.edu) Matthew K. Lau (mkl48@nau.edu)

References


See Also

enaFlow, enaUtility

Examples

data(troModels)
mti <- enaMTI(troModels[[6]])
attributes(mti)

Description

Calculates storage-based Ecological Network Analyses.

Usage

enaStorage(x, balance.override = FALSE)

Arguments

x  A network object. This includes all weighted flows into and out of each vertex as well as the amount of energy–matter stored at each vertex.

balance.override  LOGICAL: should an imbalanced model be analyzed? If FALSE, the functions checks to make sure the network model provided is at steady-state. If TRUE, then the function will run without ensuring that the model meets the steady-state assumption.
Value

- X: The storage values themselves.
- C: Output or donor-storage normalized output-oriented direct flow intensity matrix (Jacobian community matrix).
- S: Dimensionalized integral output community matrix.
- Q: Integral output storage matrix - non-dimensional.
- CP: Input or recipient-storage normalized oriented flow intensity matrix (Jacobian community matrix).
- SP: Dimensionalized integral input community matrix.
- QP: Integral input storage matrix - non-dimensional.
- dt: Selected time step to create P, PP, Q and QP - smallest whole number to make diag(C) nonnegative.
- ns: Vector of the storage based whole system network statistics. These statistics include total system storage (TSS), storage cycling index (CIS), Boundary storage intensity (BSI), Direct storage intensity (DSI), Indirect storage intensity (ISI), realized ratio of indirect-to-direct storage (IDS), unit input-oriented ratio of indirect-to-direct storage intensities (IDS.I), unit output ratio of indirect-to-direct storage intensities (IDS.O), input-oriented storage-based network homogenization (HMG.S.I), output-oriented storage-based network homogenization (HMG.S.O), input-oriented storage-based network amplification (AMP.S.I), output-oriented storage-based network amplification (AMP.S.O), Storage from Boundary flow (mode0.S), storage from internal first passage flow (mode1.S), storage from cycled flow (mode2.S), dissipative equivalent to mode1.S (mode3.S), dissipative equivalent to mode0.S (mode4.S).

Author(s)

Matthew K. Lau (mkl48@nau.edu) Stuart R. Borrett (borretts@uncw.edu)

References


See Also

read.scor, read.wand, enaFlow, enaUtility
**Examples**

```r
data(oyster)
S<-enaStorage(oyster)
attributes(S)
```

**Description**

Analysis of the structure of an ecological flow network.

**Usage**

```r
enaStructure(x = "network object")
```

**Arguments**

- `x` A network object.

**Value**

- `A` A vector of structure based network statistics. These include `n` = number of nodes, `L` = number of edges, `C` = connectivity, `LD` = link density, `ppr` = pathway proliferation rate, `lam1A` = dominant eigenvalue, `mlam1A` = multiplicity of dominant eigenvalue, `rho` = damping ratio, `R` = distance of the dominant eigenvalue from the eigen spectra, `d` = difference between dominant eigenvalue and link density, `no.scc` = number of strongly connected components, `no.scc.big` = number of strongly connected components with more than one node, `pscc` = percent of nodes in strongly connected components.

**Author(s)**

Matthew K. Lau (mkl48@nau.edu) Stuart R. Borrett (borretts@uncw.edu)

**References**


**See Also**

- `structure.statistics`

**Examples**

```r
data(troModels)
enaStructure(troModels[[6]])
```
*Trophic Aggregations (TroAgg) Analysis*

**Description**

It returns the data quantifying the underlying trophic structure of a given model based on the interaction of the living and non-living nodes. It is based on the Trophic Aggregations suggested by Lindeman (1942) and follows the algorithm by Ulanowicz and Kemp (1979) implemented in NETWRK 4.2b. It removes the Feeding cycles in the network beforehand to provide accurate results.

**Usage**

`enaTroAgg(x)`

**Arguments**

- **x**: a network object. This includes all weighted flows into and out of each node. It should include separate respiration and export values for the Canonical Exports and Canonical Respirations results respectively. It must also include the "Living" vector that identifies the living (TRUE/FALSE) status of each node. It must contain the non-living nodes at the end of the node vector, the function `netOrder` can be used for the same.

**Value**

- **Feeding_Cycles**: List that gives the details of the Feeding Cycles in the network. The output being according to the `enaCycle` function applied to the Living components in the network.
- **A**: matrix that distributes the species in integer Trophic Levels (Lindeman Transformation Matrix). The dimension of A is (NL X NL) where NL is the number of Living nodes.
- **ETL**: vector of the Effective Trophic Level of each species.
- **M_flow**: vector of the Migratory flows, if present, in the network.
- **CI**: vector of Canonical Inputs to the integer trophic levels. Displayed if the Migratory flows are present.
- **CE**: vector of Canonical exports or the exports from the integer trophic levels.
- **CR**: vector of the Canonical Respirations or the respiration values for integer trophic levels.
- **GC**: vector of the input flow to a trophic level from the preceeding trophic level. It represents the Grazing Chain for the network.
- **RDP**: vector of the Returns to Detrital Pool from each trophic level.
- **LS**: vector of the Lindeman trophic spine. It combines the Detrital pool with the autotrophs and forms a monotonically decreasing sequence of flows from one trophic level to the next, starting with the said combination.
vector of the trophic efficiencies i.e. the ratio of input to a trophic level to the amount of flow that is passed on the next level from it.

ns vector of trophic aggregations based network statistics. These include "Detritivory" the flow from the detrital pool to the second trophic level, "DetritalInput" the exogenous inputs to the detrital pool, "DetritalCirc" the circulation within the detrital pool, "NCYCS" the number of feeding cycles removed, "NNEX" the number of feeding cycle Nexuses removed and "CI" the Cycling Index for the Feeding Cycles.

Note
This and other Ulanowicz school functions require that export and respiration components of output be separately quantified.
This analysis involves the ENA Cycle analysis for removal of the Feeding Cycles in the network. These are cycles amongst only the living nodes and cause error in the trophic aggregations.
The analysis requires all the non-living nodes to be placed at the end in the network object.

Author(s)
Pawandeep Singh (pawan.s@live.co.uk)

References

See Also
enaCycle, netOrder

Examples
data(troModels)
tro6 <- enaTroAgg(troModels[[6]])
attributes(tro6)
Description

Performs the flow and storage based utility analysis developed for input-output network models of ecosystems. It returns a set of matrices for the direct and integral utilities as well as a set of utility based network statistics.

Usage

```
enaUtility(x,type=c("flow","storage"),eigen.check=TRUE,balance.override=FALSE,tol=10)
```

Arguments

- **x**: a network object. This includes all weighted flows into and out of each node. For the storage utility analysis this must also include the amount of energy–matter stored at each node (biomass).
- **type**: Determines whether the flow or storage utility analysis is returned.
- **eigen.check**: LOGICAL: should the dominant eigenvalue be checked. Like enaFlow and enaStorage analyses, enaUtility analysis considers the utility propagated over path lengths ranging for zero to infinity. For utility analysis to work properly, the path sequence must converge. enaUtility checks to see if the utility path sequence is convergent by finding the dominant eigenvalue of the direct utility matrix. If this eigenvalue is less than 1, the sequence is convergent and the analysis can be applied; if the dominant eigenvalue is greater than one, then the analysis cannot be applied. By default, the function will not return utility values if the eigenvalue is larger than one; however, if eigen.check is set to FALSE, then the function will be applied regardless of the mathematic validity.
- **balance.override**: LOGICAL: should model balancing be ignored. enaUtility assumes that the network model is at steady-state. The default setting will not allow the function to be applied to models not at steady-state. However, when balance.override is set to TRUE, then the function will work regardless.
- **tol**: The integral utility matrix is rounded to the number of digits specified in tol. This approximation eliminates very small numbers introduced due to numerical error in the ginv function. It does not eliminate the small numerical error introduced in larger values, but does truncate the numbers.

Value

- **D**: Direct flow utility intensity matrix. \((f_{ij}-f_{ji})/T_i\) for \(i,j=1:n\)
- **U**: Nondimensional integral flow utility
- **Y**: Dimensional integral flow utility
If type is set to 'flow', this is a list of flow utility network statistics including: the dominant eigenvalue of D (lambda_1D), flow based network synergism (synergism.F), and flow based network mutualism (mutualism.F).

**Author(s)**

Matthew K. Lau (mkl48@nau.edu) Stuart R. Borrett (borretts@uncw.edu)

**References**


**See Also**

enaFlow, enaStorage, enaMTI

**Examples**

```r
data(troModels)
U <- enaUtility(troModels[[6]], type="flow", eigen.check=FALSE)
attributes(U)
US <- enaUtility(troModels[[6]], type="storage", eigen.check=FALSE)
```

**Description**

This function calculates the input, output, and average environ centrality of the nodes in the network (Fath and Borret, 2012). This is a type of weighted degree centrality that indicates the relative importance of the nodes in the flow activity in the network.

**Usage**

```r
environCentrality(x='matrix')```
Arguments

x  A square matrix. Usually the integral flow matrix from enaFlow. The assumption is that the flows are oriented column to row.

Value

ECin  input oriented environ centrality
ECout  output oriented environ centrality
AEC  average environ centrality (average of input and output)

Author(s)

Matthew K. Lau (mkl48@nau.edu) Stuart R. Borrett (borretts@uncw.edu)

References


See Also

enaFlow

Examples

data(troModels)
F<-enaFlow(troModels[[6]])
ec<-environCentrality(F$N)
attributes(ec)
barplot(sort(ec$AEC,decreasing=TRUE),col=4,
ylab="Average Environ Centrality",
ylim=c(0,0.4))

findPathLength  Cumulative Flow over a Range of Path Lengths

Description

Calculates the flow throughout the entire network over a given path length.

Usage

findPathLength(x, maxPath = 100, plot.sw = FALSE)
force.balance

Arguments

x
Network model object.

maxPath
The maximum path length to calculate total flow.

plot.sw
LOGICAL: should a plot be generated showing flow accumulation?

Value

thresholds
thresholds indicating the development of throughflow as path length increases: the path length at which indirect flow exceeds direct flow (mID), path length at which 50%, 90%, and 95% of total system throughflow is achieved (m50, m90, and m95, respectively)

tf
total flow across paths from length 0 (Boundary inputs) to maxPath

ctf
cumulative total flow from path length 0 to maxPath

Author(s)

Matthew K. Lau (mkl48@nau.edu) Stuart R. Borrett (borretts@uncw.edu)

References


See Also

enaFlow

Examples

data(troModels)
pl10 <- findPathLength(troModels[[6]], plot.sw=TRUE,maxPath=10)
names(pl10)
pl10$thresholds

force.balance

Repeated Application the Balance Function

Description

This function repeatedly balances a model, sequentially with the output being passed back to the balance function, until it is within tolerance or the maximum number of iterations is reached.

Usage

force.balance(x,tol=5,max.itr=10,method='AVG2')
Arguments

- `x`: A network object.
- `tol`: Percent error tolerance for difference between inputs and outputs.
- `max.itr`: Maximum number iterations.
- `method`: The balancing method to use, see `balance`. DEFAULT = AVG2.

Value

Returns a balanced network model.

Author(s)

Matthew K. Lau (mkl48@nau.edu) Stuart R. Borrett (borretts@uncw.edu)

References


See Also

- `balance`

Examples

```r
data(troModels)
ssCheck(troModels[[1]])
fb.model=force.balance(troModels[[2]]) #produces a balanced model
```

---

get.ns

Quick Calculation of a Range of Network Statistics.

Description

This is a high level function for calculated the main network analyses (Ascendancy, Flow, Structure, Storage and Utility) on an ecological network.

Usage

```r
get.ns(x, balance.override=FALSE)
```

Arguments

- `x`: A network object.
- `balance.override`: Turns off balancing and balance checking.
get.orient

Value

Returns the network statistics (ns) of all of the major ENA functions: enaStructure, enaFlow, enaAscendency, enaStorage and enaUtility (both flow and storage).

Author(s)

Matthew K. Lau (mkl48@nau.edu) Stuart R. Borrett (borretts@uncw.edu) David E. Hines (deh9951@uncw.edu)

References


See Also

enaStructure, enaFlow, enaAscendency, enaUtility

Examples

data(troModels)
get.ns(troModels[[6]])

get.orient

Get the Current Global Matrix Orientation Setting

Description

Returns the current setting for the expected orientation of all matrices, which is either 'rc' (DEFAULT) or 'school' (output orientation as expected for the school of analysis for a given function).

Usage

get.orient()

Details

This function is intended to provide increase flexibility for users of both the Patten and Ulanowicz schools of ENA.

Author(s)

Matthew K. Lau (mkl48@nau.edu) Stuart R. Borrett (borretts@uncw.edu)
### m.list

**Sub-set of the Larger Ecosystem Models**

**Description**

A set of ecosystem models compiled by the SEE Lab at UNCW.

**Usage**

```r
data(enaModels)
```

**References**


---

### mExp

**Calculates the Exponent of a Matrix**

**Description**

Function for calculating the pathway proliferation of flows in a network model through matrix exponentiation.

**Usage**

```r
mExp(x='matrix', n=2)
```

**Arguments**

- `x`: A matrix.
- `n`: Desired exponent (i.e. the path length).

**Value**

Returns an exponentiated flow matrix.

**Author(s)**


**References**

This function was originally designed by Alberto Monteiro in the following R help thread: https://stat.ethz.ch/pipermail/r-help/2007-May/131330.html.
Description

Reorders nodes in a network either through a user defined node order vector or by default places the non-living nodes to the end of the node vector, minimizing the order change for other nodes.

Usage

\[
\text{netOrder}(x, \text{order})
\]

Arguments

- \(x\): A network object. This includes all weighted flows into and out of each node.
- \(\text{order}\): An integer vector of length \(N\), where \(N\) is number of nodes in \(x\), specifying the new order of the nodes (by default \(\text{order} = 0\), which indicates moving non-living nodes to the end).

Value

Returns a network object with nodes ordered as per the node order vector or without the node order vector, by default moves the non-living nodes to the end of the node vector, minimizing the order change for other nodes.

Note

The node order vector "order" must be of length equal to the number of nodes in \(x\) (i.e. \(N\)) and must contain all integers from 1 to \(N\).

This function can be used with default conditions (i.e. without "order" vector) to reorder the nodes of a network which does not have non-living nodes placed at the end so that the Trophic Aggregations analysis (enaTroAgg) can be run on the reordered model.

Author(s)

Pawandeep Singh (pawan.s@live.co.uk)

See Also

enaTroAgg

data(troModels)

new.network <- netOrder(troModels[[6]], c(1, 3, 2, 5, 4))

# new.network is the required rearranged network with nodes in the desired order.
Intertidal Oyster Reef Ecosystem Model

Description

Intertidal oyster reef ecosystem model created by Dame and Patten (1981). Data were taken from Patten (1985). Model flows are in kcal m^{-2} day^{-1}; storage data is kcal m^{-2}.

Usage

data(oyster)

References


Compile Network Information into a Network Class

Description

This function provides a flexible framework for importing flow network information into a network class object for analyses.

Usage

pack(flow, input=NA, respiration=NA, export=NA, output=NA, storage=NA, living=NA)

Arguments

flow The flow matrix.
input The inputs into the system.
respiration The quantities respired from the system.
export The exports from the system.
output The output (i.e. exports + respiration) from the system.
storage The quantities stored in compartments within the system.
living A logical vector indicating whether a node is either 'living' (= TRUE) or 'dead' (=FALSE).
Value

Returns a network object for the supplied model.

Author(s)

Matthew K. Lau (mkl48@nau.edu) Stuart R. Borrett (borretts@uncw.edu)

See Also

unpack

read.enam

Read ENA Model from an Mdloti Formatted Excel File

Description

This function reads network data from an excel file commonly used by Ursula Sharler. The file has three header lines (name/source, number of compartments, number of living nodes) and then a n+2 x n+2 matrix of flows. This is the flow matrix with an additional row for imports and biomass each and additional columns for exports and respirations.

Usage

read.enam(file='file path and name')

Arguments

file The name and path for the data file. This function assumes the data are stored on the first sheet of an Microsoft Excel formatted. NOTE: this function depends on the read.xlsx function from the xlsx package, which requires that the entire path be specified from the root directory (i.e. the absolute path).

Value

Returns the network object.

Author(s)

Stuart R. Borrett (borretts@uncw.edu)

References


See Also

read.scor
read.nea  Read NEA Formatted Network Model

Description
This function reads in and creates a network object from a NEA formatted data file (Fath and Borrett 2006).

Usage
read.nea(file="file name", sep=',', warn=TRUE)

Arguments
- file: The name and path for the data file.
- sep: The separation character used to delimit data values.
- warn: LOGICAL: should pack warnings be reported?

Value
Returns the network object.

Author(s)
Stuart R. Borrett (borretts@uncw.edu)

References

See Also
write.nea

---

read.scor  Read SCOR Formatted Model

Description
Read in network model data files that are in the SCOR format (REFERENCE).

Usage
read.scor(file, from.file=TRUE, warn = FALSE)
**Arguments**

- **file**: File path or plain text.
- **from.file**: States whether the file argument input should be treated as a file path (TRUE) or plain text (FALSE).
- **warn**: Turn on (TRUE) or off (FALSE) warnings.

**Details**

The SCOR file must be formatted properly. In particular, the number of nodes on the second line must have the first three characters dedicated to the total number of nodes and the next three characters should contain the number of living nodes. That is, the second line of the file should be formatted as 'xxxxyy' where x and y are the characters for the total number of nodes and the number of living nodes, respectively. Thus, if the total number of nodes is 10 and the number of living nodes is 1, then the second line should read, " 10 1."

**Value**

Returns the network model in one of several formats. The default format is a network object used by the statnet package (type="network"). Three other options are the network environ analysis format (type="nea") as defined by (Fath and Borrett 2006), a list format (type="list") and an edge list (type="edge.list").

**Author(s)**

Matthew K. Lau (mkl48@nau.edu) Stuart R. Borrett (borretts@uncw.edu)

**References**


---

**Description**

Reads WAND formatted network models.

**Usage**

```r
read.wand(file = "file name with path")
```

**Arguments**

- **file**: File path to WAND formatted data file.
Value

Returns a network object from a WAND formatted data file.

Note

IMPORTANT: this function depends on the read.xlsx function from the xlsx package, which requires that the entire path be specified from the root directory (i.e. the absolute path).

Author(s)

Matthew K. Lau (mkl48@nau.edu) Stuart R. Borrett (borretts@uncw.edu)

References


---

`scc`  
*Find the Strongly Connected Component (SCC) in a Graph*

Description

This function finds the strongly connected components (SCCs) of an adjacency matrix `A` and returns a number of derived network statistics.

Usage

`scc(A = "adjacency")`

Arguments

`A`  
an n x n adjacency matrix.

Value

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>sp</code></td>
<td>a list of structural properties including: the number of SCCs (&quot;no.scc&quot;), the number of SCCs with more than 1 node (&quot;no.scc.big&quot;), and the fraction of the network nodes participating in a large SCC (&quot;pscc&quot;)</td>
</tr>
<tr>
<td><code>membership</code></td>
<td>numeric vector giving the cluster id to which each node belongs (as in igraph:clusters)</td>
</tr>
<tr>
<td><code>scc.id</code></td>
<td>numeric vector of the numeric identity in &quot;membership&quot; of SCCs with more than 1 node</td>
</tr>
</tbody>
</table>

Note

Input matrix is assumed to be oriented from columns to rows.
scifix

Author(s)
Matthew K. Lau (mkl48@nau.edu) Stuart R. Borrett (borretts@uncw.edu)

References

See Also
enaStructure

Examples
data(troModels)
A<-enaStructure(troModels[[6]])$A
scc(A)

scifix

Standardizes Scientific Notation from SCOR Formatted Files

Description
This is a support function that corrects the scientific notation in SCOR formatted data files.

Usage
scifix(x)

Arguments

x A numeric or character scalar.

Value
Returns a numeric scalar in appropriate scientific notation.

Author(s)
Matthew K. Lau (mkl48@nau.edu)

See Also
read.scor
set.orient  

Globally Set the Output Matrix Orientation

Description

Changes the orientation of output matrices.

Usage

set.orient(x = c('rc','school'))

Arguments

x  
Orientation setting. If "rc" (DEFAULT), all matrix output will be returned in row (=input) to column (=output) orientation, regardless of school. If "school", then output matrices from functions from particular ENA schools will be oriented as expected in that school (i.e. Patten = column-row or Ulanowicz = row-column). Note, that all functions in the enaR package expect input matrices to be oriented row-column.

Details

The enaR package as a whole, and the broader network analysis community, assumes a row to column orientation; thus, the default orientation for the package is row to column (DEFAULT = 'rc'). However, functions from the Patten school were originally developed to conduct calculations and produce output in the column to row orientation. In order to facilitate the use of these functions, we also provide the option for users to return output in the orientation of the "school" (i.e. Patten results will be column to row oriented) by setting the global orientation to "school" using this function.

Author(s)

Matthew K. Lau (mkl48@nau.edu) Stuart R. Borrett (borretts@uncw.edu)

See Also

get.orient

Examples

original.orientation = get.orient()
original.orientation
set.orient('school')
get.orient()
set.orient('rc')
get.orient()
set.orient(original.orientation)
ssCheck

*Checks the Balance of Inputs and Outputs from a Network*

**Description**

This function supports the balancing process by checking if the inputs and outputs of a given network model are within acceptable limits.

**Usage**

```r
ssCheck(x, tol = 5, more = FALSE, zero.na = TRUE)
```

**Arguments**

- **x**: A network object.
- **tol**: The threshold for balance in percent difference between input and outputs.
- **more**: LOGICAL: should more detailed results be returned?
- **zero.na**: LOGICAL: should NA values be changed to zeros?

**Value**

Returns a logical value stating if the model is within acceptable limits of balance (TRUE) or if it is not (FALSE).

**Author(s)**

Matthew K. Lau (mkl48@nau.edu) Stuart R. Borrett (borretts@uncw.edu)

**References**


**See Also**

`balance`

**Examples**

```r
data(troModels)
ssCheck(troModels[[2]])
ssCheck(troModels[[6]])
```
**structure.statistics**  
*Structural Statistics of an Ecological Network*

**Description**

This function returns several network statistics that describe a network.

**Usage**

```r
structure.statistics(A = "adjacency matrix")
```

**Arguments**

- **A**  
  An adjacency matrix.

**Value**

- **n**  
  Number of nodes in A.
- **L**  
  Number of direct connections in A.
- **C**  
  Connectivity of A.
- **LD**  
  Link density.
- **lam1A**  
  First dominant eigenvalue of A.
- **nlam1A**  
  Multiplicity of the dominant eigenvalue.
- **lam2A**  
  Magnitude of the second largest eigenvalue.
- **rho**  
  Damping ratio (see Caswell 2001).
- **R**  
  Distance of lam1A from the bulk of the eigen spectrum.
- **d**  
  Difference between the dominant eigenvalue and the link density.
- **no.scc**  
  Number of strongly connected components.
- **no.scc.big**  
  Number of strongly connected components greater than 1.
- **pscc**  
  Precent of nodes participating in a strongly connected component.

**Author(s)**

Matthew K. Lau (mkl48@nau.edu) Stuart R. Borrett (borretts@uncw.edu)

**References**


**See Also**

`enaStructure.scc`
Calculate the Total Environ Storage

Description

Calculates the total storage in each n input and output environ. This function calculates the storage for both the unit input (output) and the realized input (output) environ. Realized uses the observed inputs (outputs) rather than an assumed unit input (output) to each node.

Usage

TES(x, balance.override=FALSE)

Arguments

x A network object.
balance.override

LOGICAL: should balancing being ignored.

Value

realized.input input oriented, realized storage in each environ.
realized.output output oriented, realized storage in each environ.
unit.input input oriented, unit storage in each environ.
unit.output input oriented, unit storage in each environ.

Author(s)

Matthew K. Lau (mkl48@nau.edu) Stuart R. Borrett (borretts@uncw.edu) David E. Hines (deh9951@uncw.edu)

References


See Also

enaStorage, enaEnviron

Examples

data(troModels)
tes <- TES(troModels[[6]])
tes
TET  \hspace{1cm} Calculates the Total Environ Throughflow for a Ecosystem Network Model

Description

Determines the total environ throughflow (TET) for each of the 2 x n environs of the selected network model. It returns both the TET calculated from a unit input (output) vector and from the observed or realized input (output) vector.

Usage

TET(x, balance.override = FALSE)

Arguments

x  \hspace{1cm} A network object.
balance.override  \hspace{1cm} Logical: should the function work if the model is not at steady-state?

Value

realized.input  \hspace{1cm} vector of the n realized total environ throughflows for the n input oriented environs.
realzied.output  \hspace{1cm} vector of the n realized total environ throughflows for the n output oriented environs.
unit.input  \hspace{1cm} vector of the n unit total environ throughflows for the n input oriented environs.
unit.output  \hspace{1cm} vector of the n unit total environ throughflows for the n output oriented environs.

Author(s)

Matthew K. Lau (mkl48@nau.edu) Stuart R. Borrett (borretts@uncw.edu)

References


troModels

See Also
enaEnviron

Examples

data(troModels)
tet <- TET(troModels[[6]])
tet

troModels  Trophic Models  

Description
A set of 58 trophic models compiled by the SEE Lab at UNCW.

Usage
data(troModels)

References

unpack  "Unpacks" the Network Object into Separate Objects

Description
Separates the components of a network object into separate components within a list. This includes inputs, exports, respirations, outputs (exports + respirations), storage, and internal flows.

Usage
unpack(x = "network object")

Arguments
x  A network object. This includes all weighted flows into and out of each node.
write.nea

Value

- **F**: matrix of flows from each node to each node oriented row to column.
- **z**: Node boundary inputs.
- **r**: Node boundary loss from respiration.
- **e**: Node boundary loss due to exportation.
- **y**: Node boundary loss; summation of **r** and **e**.
- **X**: Node storage or biomass.
- **Living**: Logical vector indicating whether each node is living or not.

Note

Flows are oriented from row to column.

Author(s)
Matthew K. Lau (mkl48@nau.edu) Stuart R. Borrett (borretts@uncw.edu)

See Also
pack, read.scor

Examples

data(troModels)
unpack(troModels[[6]])

---

write.nea  
Write a Network Object to File Using the NEA Data Format

Description

This function writes a network object to a NEA formatted data file (Fath and Borrett 2006).

Usage

write.nea(x, file.name, sep=',')

Arguments

- **x**: Network object.
- **file.name**: The file name or path. If a simple file name is given, this function uses the current working directory by default.
- **sep**: The separation character used to delimit data values.
write.nea

Value

   Writes a network object to a NEA formatted file and returns the output composite matrix.

Author(s)

   Stuart R. Borrett (borretts@uncw.edu)

References


See Also

   read.nea
Index

*Topic **datasets**
  bgcModels, 6
  enaModelInfo, 17
  enaModels, 17
  m.list, 30
  oyster, 32
  tromodels, 43

*Topic **enaFlow**
  enaStorage, 19

*Topic **package**
  enaR-package, 2

*Topic **read.scor**
  enaStorage, 19

as.extended, 3

bal, 4, 5
balance, 4, 5, 28, 39
bcratio, 6
bgcModels, 6

cycliv, 7

eigenCentrality, 8
enaAll, 9
enaAscendancy, 9, 10, 29
enaControl, 9, 11
enaCycle, 8, 12, 23
enaEnviron, 9, 14, 41, 43
enaFlow, 9, 15, 19, 20, 25–27, 29
enaModelInfo, 17
enaModels, 17
enaMTI, 9, 18, 25
enaR (enaR-package), 2
enaR-package, 2
enaStorage, 9, 11, 12, 16, 19, 25, 41
enaStructure, 21, 29, 37, 40
enaTroAgg, 8, 14, 22, 31
enaUtility, 6, 9, 11, 16, 19, 20, 24, 29
environCentrality, 25

findPathLength, 26, 31
force.balance, 27

gt.n, 28
get.orient, 29, 38

m.list, 30
mExp, 30

netOrder, 8, 22, 23, 31
network, 3

oyster, 32

pack, 32, 44

read.enam, 33
read.nea, 34, 45
read.scor, 11, 16, 20, 33, 34, 37, 44
read.wand, 11, 16, 20, 35

scc, 36, 40
scifix, 37
set.orient, 38
ssCheck, 39
structure.statistics, 21, 40

TES, 41
TET, 42
tromodels, 43

unpack, 33, 43

write.nea, 34, 44

46