Package ‘cheddar’

October 10, 2016

Type Package
Title Analysis and Visualisation of Ecological Communities
Version 0.1-631
Date 2016-10-10
Author Lawrence Hudson with contributions from Dan Reuman and Rob Emerson
Maintainer Lawrence Hudson <quicklizard@googlemail.com>
Description Provides a flexible, extendable representation of an ecological community and a range of functions for analysis and visualisation, focusing on food web, body mass and numerical abundance data. Allows inter-web comparisons such as examining changes in community structure over environmental, temporal or spatial gradients.
License BSD_2_clause + file LICENSE
LazyLoad yes
URL https://github.com/quicklizard99/cheddar/
NeedsCompilation yes
Repository CRAN
Date/Publication 2016-10-10 13:51:44

R topics documented:

AggregateCommunities ................................................. 3
ApplyByClass ............................................................... 5
Benguela ................................................................. 6
Body mass, numerical abundance and biomass abundance ............. 7
BodyMassBins ............................................................. 8
BroadstoneStream ....................................................... 9
cheddar ................................................................. 10
ChesapeakeBay .......................................................... 11
CollectionApply ......................................................... 11
CollectionCPS .......................................................... 12
CollectionNPS .......................................................... 14
<table>
<thead>
<tr>
<th>R topics documented:</th>
</tr>
</thead>
<tbody>
<tr>
<td>CollectionTLPS</td>
</tr>
<tr>
<td>Community</td>
</tr>
<tr>
<td>Community has property?</td>
</tr>
<tr>
<td>CommunityCollection</td>
</tr>
<tr>
<td>CommunityPropertyNames</td>
</tr>
<tr>
<td>CP</td>
</tr>
<tr>
<td>CPS</td>
</tr>
<tr>
<td>Degree</td>
</tr>
<tr>
<td>DegreeDistribution</td>
</tr>
<tr>
<td>Intervality</td>
</tr>
<tr>
<td>IsCannibal</td>
</tr>
<tr>
<td>LinearRegressionByClass</td>
</tr>
<tr>
<td>LoadCollection</td>
</tr>
<tr>
<td>LoadCommunity</td>
</tr>
<tr>
<td>LumpNodes</td>
</tr>
<tr>
<td>LumpTrophicSpecies</td>
</tr>
<tr>
<td>Millstream</td>
</tr>
<tr>
<td>Node connectivity</td>
</tr>
<tr>
<td>NodeNameIndices</td>
</tr>
<tr>
<td>NodePropertyNames</td>
</tr>
<tr>
<td>NP</td>
</tr>
<tr>
<td>NPS</td>
</tr>
<tr>
<td>NumberOfNodes</td>
</tr>
<tr>
<td>NumberOfTrophicLinks</td>
</tr>
<tr>
<td>NvMConvexHull</td>
</tr>
<tr>
<td>NvMLinearRegressions</td>
</tr>
<tr>
<td>NvMTriTrophicStatistics</td>
</tr>
<tr>
<td>NvMTriTrophicTable</td>
</tr>
<tr>
<td>Omnivory</td>
</tr>
<tr>
<td>OrderCollection</td>
</tr>
<tr>
<td>OrderCommunity</td>
</tr>
<tr>
<td>pHWebs</td>
</tr>
<tr>
<td>PlotAuppervAlower</td>
</tr>
<tr>
<td>PlotCircularWeb</td>
</tr>
<tr>
<td>PlotHelpers</td>
</tr>
<tr>
<td>PlotNPS</td>
</tr>
<tr>
<td>PlotNPSDistribution</td>
</tr>
<tr>
<td>PlotRankNPS</td>
</tr>
<tr>
<td>PlotTLPS</td>
</tr>
<tr>
<td>PlotWagonWheel</td>
</tr>
<tr>
<td>PlotWebByLevel</td>
</tr>
<tr>
<td>PredationMatrix</td>
</tr>
<tr>
<td>PredationMatrixToLinks</td>
</tr>
<tr>
<td>Pyramid plots</td>
</tr>
<tr>
<td>QuantitativeDescriptors</td>
</tr>
<tr>
<td>RemoveCannibalisticLinks</td>
</tr>
<tr>
<td>RemoveIsolatedNodes</td>
</tr>
<tr>
<td>RemoveNodes</td>
</tr>
</tbody>
</table>
## AggregateCommunities

Aggregate communities

### Description

Functions that aggregate communities in a collection.

### Usage

```r
AggregateCommunities(collection, 
                      aggregate = names(collection), 
                      weight.by='N', 
                      title = NULL)
```

```r
AggregateCommunitiesBy(collection, aggregate.by, ...)
```

### Arguments

- **collection**: an object of class `CommunityCollection`.
- **aggregate**: the names of the communities to aggregate.
- **weight.by**: the name of a column by which to compute weighted mean of numeric values.
- **title**: the title of the new Community.
- **aggregate.by**: the name of a community property, either first-class or computed, over which to aggregate.
- **...**: values passed to `AggregateCommunities`.
AggregateCommunities combines communities given in aggregate into a single new Community. Nodes that appear in one or more of the communities in aggregate are combined into a single node in the returned community. The way in which numeric node properties are aggregated is governed by the weight.by parameter. If weight.by is NULL or is not the name of a node property, the arithmetic mean is computed for each numeric node property. If weight.by is the name of a node property, that property is used to compute weighted means of the other numeric node properties; the arithmetic mean of weight.by is computed. This scheme means that if a community contains both N and M, aggregation using weight.by='N' results in the arithmetic mean of N and the N-weighted mean of M for each node. Node properties that are characters or logicals are aggregated by joining unique values with a ','. Empty character strings are ignored. Species that are not present in some communities in the collection are assumed to have a value of 0 for all numeric node properties, an empty string ('') for all character node properties and a value of NA for all logical node properties. The returned community contains the union of trophic links for each node. Community properties are aggregated by computing the arithmetic mean of numeric values and joining unique character and logical values with a ','. See the ‘Aggregating communities’ section of the ‘Collections’ vignette for a more detailed explanation and examples of how properties are aggregated.

AggregateCommunitiesBy aggregates by a property of the communities, either first-class or computed. If there is more than one unique value of the property across the contained communities, a new CommunityCollection object is returned. If there is just one unique value, a single Community is returned.

Value
A new object that is either of class Community or CommunityCollection.

Author(s)
Lawrence Hudson

See Also
CommunityCollection, CollectionCPS

Examples
data(pHWebs)

# An aggregate of 3 communities
AggregateCommunities(pHWebs, c('Old Lodge', 'Afon Hafren', 'Broadstone'))

# The Duddon Pike Beck and Mosedal Beck communities share the same
# latitude and have pH values of 6.1 and 5.9 respectively.
CollectionCPS(pHWebs[c('Duddon Pike Beck', 'Mosedal Beck')])

# Aggregating by the 'lat' property therefore results in a new collection
# of nine communities.
ApplyByClass

CollectionCPS(AggregateCommunitiesBy(pHWebs, 'lat'))

# Would produce an error
## Not run: AggregateCommunities(pHWebs, c('not a community', 'Afon Hafren'))

---

**ApplyByClass**

**Apply by class**

**Description**

Apply functions to a group of values given by a node property.

**Usage**

```r
ApplyByClass(community, property, class, fn, ...)
SumMByClass(community, class, na.rm=FALSE)
SumNByClass(community, class, na.rm=FALSE)
SumBiomassByClass(community, class, na.rm=FALSE)
```

**Arguments**

- `community` : an object of class `Community`.
- `property` : the property to which `fn` is applied.
- `class` : the property over which `fn` is applied. Defaults to 'category' if the community has a node property with that name.
- `fn` : a function.
- `na.rm` : logical - if TRUE then NA values are removed.
- `...` : Other parameters to `fn`.

**Details**

ApplyByClass applies `fn` to `property` by `class`. `property` and `class` should both be names that meet the criteria of the `properties` argument of NPS.

`SumMByClass`, `SumNByClass` and `SumBiomassByClass` are convenient wrapper around `ApplyByClass`.

**Value**

A vector or list of values, named by unique values of `class`.

**Author(s)**

Lawrence Hudson

**See Also**

Community, NPS
Examples

data(TL84)

# Sum body mass by category
ApplyByClass(TL84, 'M', 'category', sum)

# A more convenient way to sum body mass by category
SumByClass(TL84)

# Sum body mass by kingdom. The 'Unclassified flagellates' node does not have a
# kingdom, so we get a value labelled '<unnamed>'
SumByClass(TL84, 'kingdom')

# Maximum body mass by category
ApplyByClass(TL84, 'M', 'category', max)

# A list of min and max M
ApplyByClass(TL84, 'M', 'category', range)

# A list of min and max M by kingdom
ApplyByClass(TL84, 'M', 'kingdom', range)

# The same values as a matrix
do.call('rbind', ApplyByClass(TL84, 'M', 'kingdom', range))

# Broadstone Stream has some nodes in every category without M so all returned
# values are NA.
data(BroadstoneStream)
SumByClass(BroadstoneStream)

# Get rid of the NA values
SumByClass(BroadstoneStream, na.rm=TRUE)

---

Benguela

Description

The food-web of the Benguela ecosystem.

Usage

Benguela

Format

Community.
Body mass, numerical abundance and biomass abundance

Source

References

Description
Convenience functions for accessing log10-transformed body mass, M, numerical abundance, N, and biomass abundance, B.

Usage
- `logQM(community)`
- `logQN(community)`
- `Biomass(community)`
- `logQBiomass(community)`
- `logQMNBiomass(community)`
- `RCMRatio(community)`
- `logRCMRatio(community)`
- `CRMRatio(community)`
- `logCRMRatio(community)`

Arguments
- `community` an object of class `Community`.

Details
- `Log10M`, `Log10N`, `Biomass`, `Log10Biomass` and each return a value per node. `Log10MN.Biomass` returns a matrix with a row per node and columns ‘Log10M’, ‘Log10N’ and ‘Log10Biomass’. These functions are all suitable for use with NPS.
- `RCMRatio` returns the ratio between the resource and consumer body mass for every trophic link. `Log10RCMRatio` returns the same data log10-transformed. `CRMRatio` and `log10CRMRatio` are analogous functions that return the ratio between the consumer and resource body mass. These functions are all suitable for use with TLPS.

Value
A vector of length `NumberOfNodes` or a vector of length `NumberOfTrophicLinks`
**BodyMassBins**

**Author(s)**
Lawrence Hudson

**See Also**
NumberofNodes, NPS, NumberOfTrophicLinks, TLPS

**Examples**
```r
data(TL84)
NPS(TL84, c('M', 'Log10M', 'N', 'Log10N', 'Biomass', 'Log10Biomass'))
NPS(TL84, 'Log10MBiomass')
TLPS(TL84, link.properties=c('Log10RCMRatio', 'Log10RCMRatio'))
```

<table>
<thead>
<tr>
<th>BodyMassBins</th>
<th>Body-mass bins</th>
</tr>
</thead>
</table>

**Description**
Function that assigns each node in a Community to a body-mass bin.

**Usage**
```r
BodyMassBins(community, lower=min(NP(community,'M'), na.rm=TRUE),
             upper=max(NP(community,'M'), na.rm=TRUE), n.bins=10)
```

**Arguments**
- **community** an object of class Community.
- **lower** lower bound of the bins.
- **upper** upper bound of the bins.
- **n.bins** the number of bins.

**Details**
Divides the range lower to upper in to n.bins equally-spaced log10(M) bins. Assigns each node in the community to one of these bins and returns the bins numbers. The returned vector has attributes bin.centres and breaks.

**Value**
A vector of length NumberofNodes.
**Author(s)**
Lawrence Hudson

**See Also**
Community, NumberOfNodes

**Examples**
data(TL84)
BodyMassBins(TL84)

---

**Description**
The community of Broadstone Stream.
Taxonomic classification provided by Guy Woodward.

**Usage**
BroadstoneStream

**Format**
Community.

**Source**

**References**
**cheddar**

*Analysis and visualisation of ecological communities*

**Description**

Cheddar provides a flexible, extendable representation of an ecological community and a range of functions for analysis and visualisation, focusing on food web, body mass and numerical abundance data. It also allows inter-web comparisons such as examining changes in community structure over environmental, temporal or spatial gradients.


**Details**

<table>
<thead>
<tr>
<th>Package:</th>
<th>cheddar</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type:</td>
<td>Package</td>
</tr>
<tr>
<td>Version:</td>
<td>0.1-630</td>
</tr>
<tr>
<td>Date:</td>
<td>2016-10-10</td>
</tr>
<tr>
<td>License:</td>
<td>BSD 2 clause</td>
</tr>
</tbody>
</table>

**Author(s)**

Maintainer: Lawrence Hudson <quicklizard@googlemail.com>

**Examples**

```r
# The dataset of Tuesday Lake sampled in 1984
data(TL84)

# Properties of the community
CPS(TL84)

# Properties of each node
head(NPS(TL84))

# Some computed node properties
head(NPS(TL84), c('Log10M', 'Log10N', 'Log10Biomass',
                   'PreyAveragedTrophicLevel', 'TS='TrophiSpecies'))

# Properties of each trophic link
head(TLPS(TL84))

# Computed properties of each node in each link
head(TLPS(TL84, node.properties=c('Log10M', 'Log10N', 'Log10Biomass',
                                   'PreyAveragedTrophicLevel')))
```
### ChesapeakeBay

A collection of 10 webs sampled across a wide pH gradient.

```r
# A collection of 10 webs sampled across a wide pH gradient
data(pHwebs)

# A data.frame of predictors and responses
CollectionCPS(pHwebs, c('pH',
  S='NumberOfNodes',
  L='NumberOfTrophicLinks',
  C='DirectedConnectance',
  Slope='NVMslope'))
```

<table>
<thead>
<tr>
<th>ChesapeakeBay</th>
<th>ChesapeakeBay</th>
</tr>
</thead>
</table>

#### Description

The community of Chesapeake Bay sampled in the years 1983 to 1986.

#### Usage

ChesapeakeBay

#### Format

Community.

#### Source


#### References


### CollectionApply

Apply a function to every Community in a CommunityCollection. Works the same as lapply but returns a CommunityCollection rather than a list.

#### Usage

CollectionApply(collection, f, ...)

<table>
<thead>
<tr>
<th>CollectionApply</th>
<th>Collection apply</th>
</tr>
</thead>
</table>

#### Description

Apply a function to every Community in a CommunityCollection. Works the same as lapply but returns a CommunityCollection rather than a list.

#### Usage

CollectionApply(collection, f, ...)
Arguments

- collection: an object of class CommunityCollection.
- f: a function to be applied to each Community.
- ...: optional arguments passed to f.

Value

A new object of class CommunityCollection.

Author(s)

Lawrence Hudson

See Also

CommunityCollection, lapply

Examples

data(pHWebs)

# Remove isolated nodes from each community
CollectionCPS(pHWebs, 'FractionIsolatedNodes')
pHWebs.no.iso <- CollectionApply(pHWebs, RemoveIsolatedNodes)
CollectionCPS(pHWebs.no.iso, 'FractionIsolatedNodes')

# Remove cannibalistic links from each community
sapply(pHWebs, function(community) length(Cannibals(community)))
pHWebs.no.can <- CollectionApply(pHWebs, RemoveCannibalisticLinks)
sapply(pHWebs.no.can, function(community) length(Cannibals(community)))

# Order the nodes each community by body mass
head(CollectionNPS(pHWebs))
pHWebs.by.M <- CollectionApply(pHWebs, OrderCommunity, 'M')
head(CollectionNPS(pHWebs.by.M))

Description

Returns a data.frame of first-class and computed properties of communities in a CommunityCollection.

Usage

CollectionCPS(collection, properties=NULL)
Arguments

- collection: an object of class CommunityCollection.
- properties: the names of the properties to be returned.

Details

This function is named CollectionCPS for Collection Community Properties.

The properties argument is a vector whose entries are either names of first-class properties or names of functions which take as single required argument a CommunityCollection and return a single value. If properties is NULL, all first-class properties are included in the returned data.frame.

Value

A data.frame.

Author(s)

Lawrence Hudson

See Also

CPS, CommunityPropertyNames, CommunityCollection

Examples

data(pHWwebs)

CollectionCPS(pHWwebs)

# pH and a computed property
CollectionCPS(pHWwebs, c('pH', 'NumberOfNodes'))

# A shorter name for the 'NumberOfNodes' column
CollectionCPS(pHWwebs, c('pH', S='NumberOfNodes'))

# A function that returns more than one value. Some pHWwebs communities contain nodes (detritus and the like) that do not have a category. These appear in # <unnamed>.
CollectionCPS(pHWwebs, 'SumBiomassByClass')

# Prefix columns with 'B'
CollectionCPS(pHWwebs, c(B='SumBiomassByClass'))

# Remove biomasses of NA
CollectionCPS(pHWwebs, list(B=list('SumBiomassByClass', na.rm=TRUE)))
CollectionNPS  

Collection node properties

Description

Returns a data.frame of first-class and computed node properties of communities in a CommunityCollection.

Usage

CollectionNPS(collection, properties=NULL)

Arguments

collection  
an object of class CommunityCollection.

properties  
names of the properties. These can be names of first-class properties and names of functions. Names must meet the criteria of the properties parameter of NPS.

Details

This function is named CollectionNPS for Collection Node PropertieS. If properties is NULL, all first-class node properties are included in the returned data.frame.

Value

A data.frame.

Author(s)

Lawrence Hudson

See Also

NPS, CommunityCollection

Examples

data(pHwebs)
head(CollectionNPS(pHwebs), 10)

head(CollectionNPS(pHwebs, 'M'), 10)

# Biomass is a function
head(CollectionNPS(pHwebs, 'Biomass'), 10)

head(CollectionNPS(pHwebs, c(B='Biomass')), 10)

# Several first-class and computed properties
head(CollectionNPS(pHwebs, c('M', 'N', B='Biomass', 'TrophicSpecies', TL='PreyAveragedTrophicLevel')), 10)
# Pass parameters to functions

```r
head(CollectionNPS(phwebs,
     list(TS1=‘TrophicSpecies’,
         TS2=list(‘TrophicSpecies’, include.isolated=FALSE),
         Iso=‘IsIsolatedNode’)), 10)
```

### CollectionTLPS

<table>
<thead>
<tr>
<th>CollectionTLPS</th>
<th>Collection trophic-link properties</th>
</tr>
</thead>
</table>

### Description

Returns a data.frame of first-class and computed trophic-link properties of communities in a `CommunityCollection`.

### Usage

```r
CollectionTLPS(collection, node.properties=NULL, link.properties=NULL)
```

### Arguments

- **collection**: an object of class `CommunityCollection`.
- **node.properties**: the names of the node properties to return. Should meet the criteria of the `node.properties` parameter of `TLPS`.
- **link.properties**: the names of the trophic link properties to return. Should meet the criteria of the `link.properties` parameter of `TLPS`.

### Details

This function is named `CollectionTLPS` for Collection Trophic Link Properties. It returns a data.frame containing the columns ‘resource’ and ‘consumer’ and any requested node and trophic-link properties. If `node.properties` and `link.properties` are both `NULL` then all first-class trophic-link properties are included in the returned data.frame.

### Value

A data.frame.

### Author(s)

Lawrence Hudson

### See Also

- `TLPS`
- `CommunityCollection`
Examples

```r
data(pHWebs)

# Just community, resource and consumer
head(CollectionTLPS(pHWebs), 10)

# The M of the resource and consumer in each link
head(CollectionTLPS(pHWebs, node.properties='M'), 10)
```

Description

Creates and returns a new object that represents an ecological community.

Usage

```r
Community(nodes, properties, trophic.links = NULL)

## S3 method for class 'Community'
print(x, ...)

## S3 method for class 'Community'
plot(x, ...)

## S3 method for class 'Community'
summary(object, ...)
```

Arguments

- **nodes**: a data.frame containing one row per node. A column called `node` is mandatory and must contain node names. An error is raised if any node names are duplicated. Whitespace is stripped from the beginning and end of node names. If provided, columns called `M` and/or `N` must represent mean body mass and mean numerical abundance respectively. All values in `M` and `N` must be either NA or greater than 0 and less than infinity. If the columns `M` and/or `N` are in nodes, values named `M.units` and/or `N.units` must be provided in properties.

- **properties**: a list of properties the community as a whole. All elements must be named and must be of length one.

- **trophic.links**: NULL or a data.frame or matrix of trophic link properties. If not NULL, columns called ‘resource’ and ‘consumer’ must be given and these should contain node names. An error is raised if any names in resource or consumer are not in `nodes$node`. Whitespace is stripped from the beginning and end of all names in ‘resource’ and ‘consumer’. Other columns are taken to be properties of links. An error is raised if any links appear more than once.

- **x**: an object of class `Community`.

- **object**: an object of class `Community`.

- **...**: further arguments passed to other methods.
Details

The most convenient way to import community data into Cheddar is to put data into CSV files and use the `loadcommunity` function.

Many of Cheddar’s plot and analysis functions make use of the ‘category’ node property by default, following previously-used metabolic groupings (Yodzis and Innes, 1992). The column `nodes$category` is optional but, if given, it should contain one of ‘producer’, ‘invertebrate’, ‘vert.ecto’, ‘vert.endo’ or should be an empty string.

Community supports standard generic functions `plot`, `print`, and `summary`.

Value

A new object of class `Community`.

Author(s)

Lawrence Hudson

References


See Also

`CPS`, `NPS`, `TLPS`, `LoadCommunity`, `SaveCommunity`

Examples

data(TL84)
TL84

# Node properties
NPS(TL84)

# Trophic-link properties
TLPS(TL84)

# Eyeball the data
plot(TL84)

# A different plot function
PlotWebByLevel(TL84)

# Construct a new community.
# TL84.new is an exact copy of TL84
TL84.new <- Community(
  properties=CPS(TL84),
  nodes=NPS(TL84),
  trophic.links=TLPS(TL84))

identical(TL84, TL84.new)
# A copy of TL84 without trophic links
TL84.no.links <- Community(properties=CPS(TL84),
    nodes=NPS(TL84))
NumberOfTrophicLinks(TL84.no.links)

# A community with 10 species and no properties
test <- Community(nodes=data.frame(node=paste('Species', 1:10)),
    properties=list(title='Test community'))
test
NPS(test)

Community has property?

**Description**

Functions that return whether or not a community has a particular property.

**Usage**

HasM(community)
HasN(community)
HasTrophicLinks(community)

**Arguments**

community an object of class Community.

**Value**

A logical.

**Author(s)**

Lawrence Hudson

**See Also**

Community, CPS, NPS, TLPS

**Examples**

# Tuesday Lake 1984 has all three
data(TL84)
HasM(TL84)
HasN(TL84)
HasTrophicLinks(TL84)
# Skipwith Pond has trophic links but not M or N
data(SkipwithPond)
HasM(SkipwithPond)
HasN(SkipwithPond)
HasTrophicLinks(SkipwithPond)

CommunityCollection  Collections of communities

Description
Collections of communities

Usage
CommunityCollection(communities)
## S3 method for class 'CommunityCollection'
print(x, ...)
## S3 method for class 'CommunityCollection'
plot(x, ncol=min(length(x),5), by.col=TRUE, plot.fn=plot, ...)
## S3 method for class 'CommunityCollection'
summary(object, ...)

Arguments
- **communities**: a list of Community objects.
- **x**: an object of class CommunityCollection.
- **object**: an object of class CommunityCollection.
- **ncol**: the number of columns in the plot.
- **by.col**: logical - if TRUE communities are plotted along columns.
- **plot.fn**: a plot function that accepts a Community object.
- **...**: further arguments passed to other methods.

Details
Constructs a new CommunityCollection from a list of existing Community objects. CommunityCollection is a subclass of list. CommunityCollection objects can not be modified directly.

An error is raised if any Community objects in communities share the same ‘title’ community property. An error is also raised if the Community objects in communities do not all have the same value of the community properties ‘M.units’ and ‘N.units’. CommunityCollection places no restrictions on other properties. For example, all of the ten communities with the pHWebs collection has a valid pH property but this is not enforced by CommunityCollection - it would be possible for a Community within a collection to not have a pH property, to have a pH of NA or even to have an invalid pH, for example a negative value.

CommunityCollection supports standard generic functions plot, print, subset and summary.
CommunityCollection

Value

A new object of class CommunityCollection.

Author(s)

Lawrence Hudson

See Also

Community, CollectionCPS, CollectionNPS, CollectionTLPS, OrderCollection, subset, CommunityCollection, AggregateCommunitiesBy, AggregateCommunities, pHWebs

Examples

# 10 stream webs sampled over a wide pH gradient
data(phWebs)
phWebs

# Eyeball the webs
plot(phWebs)

# Consistent axis limits
plot(phWebs, xlim=c(-14,6), ylim=c(-3,13))

# Different plot function
plot(phWebs, plot.fn=PlotWebByLevel, ylim=c(1,4.5))

# list-like operations
length(phWebs)
sapply(phWebs, 'NumberOfTrophicLinks')
phWebs[['Broadstone']]  # Access the Community

# A new CommunityCollection containing every other pH web
phWebs[seq(1, 10, by=2)]

# A new CommunityCollection containing two webs
phWebs[c('Old Lodge','Bere Stream')]

# CollectionCPS gets community properties
CollectionCPS(phWebs)  # Webs are sorted by increasing pH

# Order by decreasing pH
phWebs.decreasing.pH <- OrderCollection(phWebs, 'pH', decreasing=TRUE)
CollectionCPS(phWebs.decreasing.pH)

# Order by name
phWebs.name <- OrderCollection(phWebs, 'title')
CollectionCPS(phWebs.name, c('pH', 'NumberOfNodes'))
# The following will always be TRUE.
all(FALSE==duplicated(names(pHWebs)))

# A new collection of the two Tuesday Lake communities
data(TL84, TL86)
BothTL <- CommunityCollection(list(TL84, TL86))
CollectionCPS(BothTL)

# You can't modify CommunityCollections
## Not run: pHWebs[1] <- 'silly'

CommunityPropertyNames

Names of community properties

Description

Returns a vector of names of community properties.

Usage

CommunityPropertyNames(community)

Arguments

community an object of class Community.

Value

A vector of names of community properties.

Author(s)

Lawrence Hudson

See Also

Community, CP, CPS

Examples

data(TL84)
CommunityPropertyNames(TL84)
Description

Returns a single community property or NA if property is not in CommunityPropertyNames.

Usage

CP(community, property)

Arguments

community an object of class Community.
property the name of the community property to be returned.

Details

This function is named CP for Community Property.

Value

A single community property.

Author(s)

Lawrence Hudson

See Also

Community, CPS, CommunityPropertyNames

Examples

data(TL84)
CP(TL84, 'title')

CP(TL84, 'lat')

CP(TL84, 'M.units')

# Returns a vector of NA
CP(TL84, 'not a property')
Description

Returns a list of first-class and computed community properties.

Usage

CPS(community, properties = NULL)

Arguments

community an object of class Community.
properties the names of the properties to be returned.

Details

This function is named CPS for Community PropertieS. If properties is NULL, all properties are returned. If properties is not NULL then a list containing that subset of community properties is returned. Elements will be NA for values of property not in CommunityPropertyNames.

Value

A list.

Author(s)

Lawrence Hudson

See Also

Community, CP, CommunityPropertyNames

Examples

data(TL84)

# All properties
CPS(TL84)

# Just lat and long
CPS(TL84, c('lat', 'long'))

# lat and long and number of nodes
CPS(TL84, c('lat', 'long', 'NumberOfNodes'))

# lat and long and number of nodes, renamed
CPS(TL84, c('lat', 'long', S='NumberOfNodes'))
# 'not a property' is NA
CPS(TL84, c('lat', 'long', S='NumberOfNodes', 'not a property'))

<table>
<thead>
<tr>
<th>Degree</th>
<th>Node degree</th>
</tr>
</thead>
</table>

**Description**

The number of trophic links in to and out of nodes in a Community.

**Usage**

- Degree(community)
- InDegree(community)
- TrophicGenerality(community)
- NumberOfResources(community)
- OutDegree(community)
- TrophicVulnerability(community)
- NumberOfConsumers(community)

- NormalisedTrophicGenerality(community)
- NormalisedTrophicVulnerability(community)

**Arguments**

- community an object of class Community.

**Details**

InDegree and OutDegree return the number of trophic links in-to and out-of each node. Degree returns InDegree + OutDegree. TrophicGenerality and NumberOfResources are synonyms for InDegree. TrophicVulnerability and NumberOfResources are synonyms for OutDegree.

NormalisedTrophicGenerality and NormalisedTrophicVulnerability return the containing the number of resources and consumer of each node, normalised with respect to LinkageDensity. The mean of the values returned by both NormalisedTrophicGenerality and NormalisedTrophicVulnerability is 1, making their standard deviations comparable across different food webs.

**Value**

A vector of length NumberOfNodes.

**Author(s)**

Lawrence Hudson
References

See Also
Community, NumberOfNodes, LinkageDensity, DirectedConnectance, DegreeDistribution

Examples

data(TL84)

d <- Degree(TL84)
i <- InDegree(TL84)
o <- OutDegree(TL84)

# This equality is always TRUE for all food webs
all(d == i+o)

ntg <- NormalisedTrophicGenerality(TL84)
mean(ntg)  # Equals 1
ntv <- NormalisedTrophicVulnerability(TL84)
mean(ntv)  # Equals 1

DegreeDistribution    Node degree distribution

Description
Node degree distribution.

Usage
DegreeDistribution(community, cumulative=FALSE)

Arguments

community    an object of class Community.
cumulative    logical - if TRUE the cumulative degree distribution is returned.

Details
Returns a vector of proportions of nodes with \( \theta : \max \text{(Degree(community))} \) trophic links.

Value
A vector of numbers.
Author(s)
Lawrence Hudson

See Also
Degree, PlotDegreeDistribution

Examples

data(TL84)
DegreeDistribution(TL84)
DegreeDistribution(TL84, cumulative=TRUE)

Description
Functions for computing the sum diet/consumer gaps of each species in a Community and for minimising the sum diet/consumer gaps using a simulated annealing learning method.

Usage

SumDietGaps(community)
SumConsumerGaps(community)

MinimiseSumDietGaps(community, T.start = 10, T.stop = 0.1, c = 0.9,
swaps.per.T = 1000, trace.anneal = FALSE, n = 1,
best = TRUE)

MinimiseSumConsumerGaps(community, T.start = 10, T.stop = 0.1, c = 0.9,
swaps.per.T = 1000, trace.anneal = FALSE, n = 1,
best = TRUE)

Arguments

community an object of class Community.
T.start the temperature at which annealing starts; must be >0
T.stop annealing will stop when the system temperature drops below T.stop; must be >0 and <T/start
C cooling coefficient; must be >0 and <1.
swaps.per.T the number of predation matrix row swaps per temperature.
trace.anneal logical - if TRUE the annealing process prints feedback.
n numeric - the number of repetitions of the minimisation procedure.
best logical - if TRUE then only the result of the best of the n minimisations is returned.
Details

SumDietGaps and SumConsumerGaps return the total number of gaps in each species’ diet (Stouffer et al 2006) and each species’ consumers (Zook et al 2011) respectively.

MinimiseSumDietGaps and MinimiseSumConsumerGaps use the simulated annealing learning method described by Stouffer et al (2006) to minimise either SumDietGaps or SumConsumerGaps. Simulated annealing learning is a stochastic method so several optimisations might be required to find the global minimum. Use a value of \( n \) greater than 1 to perform several optimisations.

Value

For SumDietGaps and SumConsumerGaps, a single number. For the two minimisation functions, if \( n \) is 1 or best is TRUE, a list containing the values

- \text{sum.gaps} \quad \text{the lowest SumDietGaps or SumConsumerGaps resulting from the best ordering that was found.}
- \text{order} \quad \text{a vector of node names giving the best ordering.}
- \text{reordered} \quad \text{community reordered by the best ordering.}

If \( n \) is greater than 1 and best is FALSE then a list of \( n \) lists, each list containing the above three values, sorted by increasing \text{sum.gaps}.

Author(s)

Lawrence Hudson

References


See Also

Community, OrderCommunity, PredationMatrix, PlotPredationMatrix

Examples

data(TL84)
# Perform 5 independent optimisations
res <- MinimiseSumDietGaps(TL84, n=5)

# Compare the original, ordered by body mass and minimised predation matrices
par(mfrow=c(1,3))
PlotPredationMatrix(TL84, main=paste('Sum diet gap', SumDietGaps(TL84)))
TL84.by.M <- OrderCommunity(TL84, 'M')
PlotPredationMatrix(TL84.by.M, main=paste('Sum diet gap', SumDietGaps(TL84.by.M)))
PlotPredationMatrix(res$reordered, main=paste('Sum diet gap', res$sum.gaps))
IsCannibal

# The same comparison but retaining the original column ordering
par(mfrow=c(1,3))
PlotPredationMatrix(TL84)
PlotPredationMatrix(TL84, resource.order=NP(TL84.by.M, 'node'))
PlotPredationMatrix(TL84, resource.order=res$order)

<table>
<thead>
<tr>
<th>IsCannibal</th>
<th>Cannibalistic nodes</th>
</tr>
</thead>
</table>

Description

Nodes that consume themselves in the food web.

Usage

IsCannibal(community)
Cannibals(community)
FractionCannibalistic(community)

Arguments

community an object of class Community.

Details

IsCannibal returns a vector of logical of length NumberOfNodes; values are TRUE for nodes consume themselves. Cannibals returns the names of nodes for which IsCannibals returns TRUE. FractionCannibalistic returns the proportion of nodes for which IsCannibal returns TRUE

Value

Either a logical vector of length NumberOfNodes or a vector of names.

Author(s)

Lawrence Hudson

See Also

RemoveCannibalisticLinks, NumberOfNodes, PredationMatrix, Degree, InDegree, OutDegree, ResourcesByNode, ConsumersByNode, ResourcesOfNodes, ConsumersOfNodes

Examples

data(TL84)

IsCannibal(TL84)
Cannibals(TL84)
FractionCannibalistic(TL84)
LinearRegressionByClass

Linear regression by class

Description

Fit linear regressions to node data by class.

Usage

LinearRegressionByClass(community, X, Y, class)

Arguments

community an object of class Community.
X Independent variable. A property name that must meet the criteria of the properties parameter of NPS.
Y Dependent variable. A property name that must meet the criteria of the properties parameter of NPS.
class The property over which linear regressions are fitted.

Details

A linear model is fitted through all data points and through each subset of the data given by class. A list of lm objects is returned. The list will contain NULL if it is not possible to fit a linear regression to that class; this will happen for classes that contain just a single node or that contain all or all but one nodes where X and/or Y is NA.

Value

A list of lm objects.

Author(s)

Lawrence Hudson

See Also

Community, ApplyByClass, NPS, NvMLinearRegressions, lm

Examples

data(TL84)

# Regressions fitted to log10(Biomass) versus log10(M) data.
models <- LinearRegressionByClass(TL84, 'Log10M', 'Log10Biomass', 'category')
LoadCollection

Loading and saving CommunityCollection objects

Description

LoadCollection and SaveCollection are functions for loading and saving CommunityCollection objects to text files.

Usage

LoadCollection(dir, ...)  
SaveCollection(collection, dir, ...)

Arguments

collection  an object of class CommunityCollection.  
dir         a directory.  
...          other values to LoadCommunity or SaveCommunity.

Details

The Community objects in collection are saved to a directory named communities inside dir. The order of the collection is not saved. Any existing data in dir is ignored.

Value

LoadCollection returns a new CommunityCollection.

Author(s)

Lawrence Hudson

See Also

CommunityCollection, OrderCollection, LoadCommunity, SaveCommunity
Examples

```r
data(phWwebs)
temp.path <- tempfile()
SaveCollection(phWwebs, temp.path)
phWwebs.loaded <- LoadCollection(temp.path)
phWwebs.loaded <- OrderCollection(phWwebs.loaded, 'pH')
unlink(temp.path, recursive=TRUE)
identical(phWwebs, phWwebs.loaded)  # TRUE
```

Description

LoadCommunity and SaveCommunity are functions for loading and saving Community objects to CSV files.

Usage

```r
LoadCommunity(dir, fn='read.csv', ...)
SaveCommunity(community, dir, fn='write.csv', na='', ...)```

Arguments

- `community` an object of class Community.
- `dir` a directory.
- `fn` the name of an R function that loads/saves CSV files.
- `na` the string to use for missing values in the data; see `write.csv`.
- `...` other values to `fn`.

Details

Data are stored in CSV (Comma-Separated Value) files in `dir`. Properties of any aspect of the community (nodes, links or the whole community) can be added simply by adding columns to the relevant CSV file. The data-quality checks defined by Community are applied by LoadCommunity. The `fn` and `dots` arguments can be used to read/write files in a range of formats.

`properties.csv` defines items applicable to the community as a whole, such as sampling date, lat & long or altitude and environmental variables such as temperature or pH. This file must contain a column called ‘title’.

`nodes.csv` should contain the list of nodes and together with any associated properties such as mean body mass, mean numerical abundance and classification. This file must contain a column called ‘node’ that must contain node names. Many of Cheddar’s plot and analysis functions make use of the ‘category’ node property by default, following previously-used metabolic groupings (Yodzis & Innes, 1992). The ‘category’ column of `nodes.csv` is optional but, if given, it should contain one of ‘producer’, ‘invertebrate’, ‘vert.ecto’, ‘vert.endo’ or should be an empty string.
LumpNodes

A function that lumps together nodes in a Community.

Usage

LumpNodes(community,
    lump,
    title = NULL,
    weight.by = 'N')
**LumpNodes**

**Arguments**

- `community`: an object of class `Community`
- `lump`: a vector of length `NumberOfNodes` containing names of lumped nodes. Nodes with the same value of `lump` will be merged.
- `title`: the title of the new `Community`.
- `weight.by`: the name of a column by which to compute weighted mean of numeric values.

**Details**

If `weight.by` is not `NULL` and it is the name of a node property, it is used to compute weighted means of all the other numeric node properties. The arithmetic mean of `weight.by` is computed. If `weight.by` is `NULL` or is not the name of a node property, the arithmetic mean is computed for each numeric node property. Node properties that are characters or logicals are aggregated by joining unique values with a `,`. Empty character strings are ignored.

**Value**

A new object of class `Community`.

**Author(s)**

Lawrence Hudson

**See Also**

- `LumpTrophicSpecies`, `IsIsolatedNode`, `IsolatedNodes`, `NPS`, `weighted.mean`

**Examples**

```r
data(TL84)

# Lump together isolated nodes in TL84
length(which(IsIsolatedNode(TL84)))  # 6 isolated species
IsolatedNodes(TL84)  # Names of isolated nodes

lump <- NP(TL84, 'node')  # Existing node names

# Give isolated nodes the same lump value
lump[IsolatedNodes(TL84)] <- 'Isolated nodes lumped together'
TL84.lumped <- LumpNodes(TL84, lump)

NumberOfNodes(TL84)  # 56 nodes in unlumped web
NumberOfNodes(TL84.lumped)  # 51 nodes in lumped web

IsolatedNodes(TL84.lumped)  # A single node

# This trivial example shows that no nodes are lumped if values in lump are
# unique to each node
lump <- NP(TL84, 'node')
```
identical(TL84, LumpNodes(TL84, lump, title=CP(TL84, 'title')))

# Ythan Estuary contains two species that are split in to adult and
# juvenile forms. The example below lumps these in to single species.
data(YthanEstuary)

# The names of nodes in YthanEstuary
lump <- NP(YthanEstuary, 'node')

# European flounder:
# "Platichthys flesus" and "Platichthys flesus (juvenile)"
# Lump these in to one node
lump["Platichthys flesus (juvenile)"==lump] <- "Platichthys flesus"

# Common eider:
# "Somateria mollissima" and "Somateria mollissima (juvenile)"
# Lump these in to one node
lump["Somateria mollissima (juvenile)"==lump] <- "Somateria mollissima"
YthanEstuary.lumped <- LumpNodes(YthanEstuary, lump)

# Examine the computed means for Somateria mollissima
# Arithmetic mean of N is 2592
NP(YthanEstuary.lumped, 'N')['Somateria mollissima']
mean(NP(YthanEstuary, 'N')[[c("Somateria mollissima (juvenile)",
                           "Somateria mollissima")]])

# N-weighted mean of M is 1637.018
NP(YthanEstuary.lumped, 'M')['Somateria mollissima']
weighted.mean(NP(YthanEstuary, 'M')[[c("Somateria mollissima (juvenile)",
                                   "Somateria mollissima")]],
              NP(YthanEstuary, 'N')[[c("Somateria mollissima (juvenile)",
                                     "Somateria mollissima")]], )

# Plot the original community and the community with lumped nodes
par(mfrow=c(1,2))
plot(YthanEstuary, highlight.nodes=c("Platichthys flesus",
                                   "Platichthys flesus (juvenile)",
                                   "Somateria mollissima",
                                   "Somateria mollissima (juvenile)")
plot(YthanEstuary.lumped, highlight.nodes=c("Platichthys flesus",
                                          "Somateria mollissima"))

LumpTrophiSpecies Lump trophic species

Description
Lump trophic species.
LumpTrophicSpecies

Usage

LumpTrophicSpecies(community, include.isolated=TRUE, title=NULL, ...)

Arguments

community an object of class Community.
include.isolated if TRUE then nodes for which IsIsolatedNode is TRUE are given their own trophic species number. If FALSE the isolated species are assigned a trophic species of NA.
title the title of the new Community.
... other parameters to LumpNodes.

Details

Aggregates nodes that share identical sets of prey and predators.

Value

A Community.

Author(s)

Lawrence Hudson

References


See Also

TrophicSpecies, LumpNodes, IsIsolatedNode

Examples

data(TL84)
NumberOfNodes(TL84)

TL84.lumped <- LumpTrophicSpecies(TL84)

length(unique(TrophicSpecies(TL84))) # 22 trophic species in TL84...
NumberOfNodes(TL84.lumped) # ... and 22 nodes in the lumped web
Millstream

Description

The control and drought treatments from one of the four replicates from a long-running study of the effects of drought on community structure.

Taxonomic classification provided by Mark Ledger.

Usage

Millstream

Format

CommunityCollection.

Source


References


Node connectivity

Description

Functions that report the connectivity of nodes in a food web.
Node connectivity

Usage

IsBasalNode(community)
IsTopLevelNode(community)
IsIntermediateNode(community)
IsIsolatedNode(community)
IsConnectedNode(community)
IsNonBasalNode(community)
IsNonTopLevelNode(community)

BasalNodes(community)
TopLevelNodes(community)
IntermediateNodes(community)
IsolatedNodes(community)
ConnectedNodes(community)
NonTopLevelNodes(community)
NonBasalNodes(community)

FractionBasalNodes(community)
FractionIntermediateNodes(community)
FractionTopLevelNodes(community)
FractionIsolatedNodes(community)
FractionNonBasalNodes(community)
FractionConnectedNodes(community)
FractionNonTopLevelNodes(community)

Arguments

community an object of class Community.

Details

Each node in a community is defined as:

- isolated: No resources or consumers, other than possibly itself
- basal: No resources and one or more consumers
- top-level: One or more resources and no consumers, other than possibly itself
- intermediate: Nodes not fitting any of the above categories

These definitions allow the following additional definitions:

- connected: basal, intermediate or top-level
- non-basal: isolated, intermediate or top-level
- non-top-level: isolated, basal or intermediate

For each of the above seven definitions, ‘X’, there are three functions: IsX, X and FractionX. The first returns a vector of logical of length NumberOfNodes; values are TRUE for nodes that fit the definition of ‘X’. The second returns the names of nodes for which IsX returns TRUE. The third returns the proportion of nodes in the community that fit the definition of ‘X’.
NodeNameIndices

Value

Either a logical vector of length NumberOfNodes or a vector of names.

Author(s)

Lawrence Hudson

See Also

NumberOfNodes, Cannibals, IsCannibal, NumberOfTrophicLinks, PredationMatrix, Degree, InDegree, OutDegree, ResourcesByNode, ConsumersByNode, ResourcesOfNodes, ConsumersOfNodes

Examples

data(TL84)

# Assemble a table of node connectivity. Only one of each of the following
# four properties is TRUE for each node.
connectivity <- NPS(TL84, c('IsBasalNode', 'IsIsolatedNode',
                            'IsIntermediateNode', 'IsTopLevelNode'))

connectivity

# Each row sums to 1, confirming that exactly one of the columns in each row
# is TRUE.
all(1==rowSums(connectivity))

# These summations are 1
sum(FractionBasalNodes(TL84),
    FractionIntermediateNodes(TL84),
    FractionTopLevelNodes(TL84),
    FractionIsolatedNodes(TL84))

sum(FractionConnectedNodes(TL84),
    FractionIsolatedNodes(TL84))

sum(FractionBasalNodes(TL84),
    FractionNonBasalNodes(TL84))

sum(FractionTopLevelNodes(TL84),
    FractionNonTopLevelNodes(TL84))

NodeNameIndices

Node name indices

Description

Node name indices.
**NodePropertyNames**

Usage

NodeNameIndices(community, nodes)

Arguments

  community  an object of class Community.
  nodes      node names.

Details

Returns integer indices of names in nodes.

Value

A vector of integers

Author(s)

Lawrence Hudson

See Also

Community

Examples

data(TL84)

  NodeNameIndices(TL84, 'Umbra limi')
  NodeNameIndices(TL84, c('Nostoc sp.', 'Umbra limi'))

**Description**

Returns a vector of names of node properties.

Usage

NodePropertyNames(community)

Arguments

  community  an object of class Community.
Value
A vector of the names of node properties.

Author(s)
Lawrence Hudson

See Also
NP, NPS

Examples

```r
data(TL84)
NodePropertyNames(TL84)
```
See Also

NPS, NumberOfNodes

Examples

data(TL84)

NP(TL84, 'M')

# Returns a vector of NA
NP(TL84, 'not a property')

Description

Returns a data.frame of first-class and computed node properties.

Usage

NPS(community, properties = NULL)

Arguments

community an object of class Community.

properties the names of node properties. These can be names of first-class properties (returned by NodePropertyNames) and names of functions that take a Community object as the only parameter and return either a vector of length NumberOfNodes or a matrix or data.frame with NumberOfNodes rows.

Details

This function is named NPS for Node Properties. It returns a data.frame containing the column ‘node’ and any requested properties. If properties is NULL, all first-class node properties are included in the returned data.frame.

properties should be either a vector or a list that contains either names of first class properties, names of functions that take only a community or lists in which the first element is the name of a function that takes a community and subsequent elements are named arguments to that function. Names of properties are column names in the returned data.frame.

Value

A data.frame with NumberOfNodes rows.

Author(s)

Lawrence Hudson
See Also

np, NumberOfNodes

Examples

data(TL84)
NPS(TL84)
NPS(TL84, 'M')

# Biomass is a function
NPS(TL84, 'Biomass')
NPS(TL84, c(B='Biomass'))

# Several first-class and computed properties
NPS(TL84, c('M', 'N', B='Biomass', 'TrophicSpecies',
          TL='PreyAveragedTrophicLevel'))

# Pass parameters to functions
NPS(TL84, list(TS1='TrophicSpecies',
              TS2=list('TrophicSpecies', include.isolated=FALSE),
              Iso='IsIsolatedNode'))

NumberofNodes  Number of nodes

Description

Functions that return the number of nodes in the community.

Usage

NumberOfNodes(community)
NumberOfNodesByClass(community, class)
FractionOfNodesByClass(community, class)

Arguments

community  an object of class Community.
class  the property over which fn is applied. Defaults to 'category' if the community
       has a node property with that name.

Value

NumberOfNodes returns a single number. NumberOfNodesByClass and FractionOfNodesByClass
both return a value for each class.
**Number Of Trophic Links**

**Author(s)**
Lawrence Hudson

**See Also**
Community, NPS

**Examples**

```r
data(TL84)
NumberOfNodes(TL84)
NumberOfNodesByClass(TL84)
FractionOfNodesByClass(TL84)
```

---

**Description**

The number of trophic links in Community.

**Usage**

```r
NumberOfTrophicLinks(community)
LinkageDensity(community)
DirectedConnectance(community)
```

**Arguments**

- `community`: an object of class Community.

**Details**

- `NumberOfTrophicLinks` returns the total number of links in the web, including cannibalistic links.
- `LinkageDensity` returns the `NumberOfTrophicLinks / NumberOfNodes`, including cannibalistic links and isolated nodes.
- `DirectedConnectance` returns `NumberOfTrophicLinks / NumberOfNodes^2`, including cannibalistic links and isolated nodes.

**Value**

A single number.

**Author(s)**
Lawrence Hudson
References


See Also

NumberOfNodes

Examples

data(TL84)

NumberOfTrophicLinks(TL84)
LinkageDensity(TL84)
DirectedConnectance(TL84)

Description

Compute the convex hull around log-transformed $N$ versus $M$ data.

Usage

NvMConvexHull(community)

Arguments

community an object of class Community.

Details

Returns the points and area of the minimum convex hull (a polygon in log10-transformed numerical abundance versus log10-transformed body mass space) that bounds all the species within the community.

Value

A list containing the values

- nodes The names of the nodes that make up the convex hull.
- points A matrix containing columns ‘x’ and ‘y’ that contain the coordinates of the points that make up the convex hull.
- area The area within the convex hull.

Author(s)

Lawrence Hudson
References


See Also

Community, PlotNvM, chull

Examples

data(TL84)

# Compute convex hull
convex.hull <- NvMConvexHull(TL84)

# The nodes that form the hull
convex.hull$nodes

# The area of the hull
convex.hull$area

# Plot the hull in red around the nodes
PlotNvM(TL84)
polygon(convex.hull$points, lwd=2, border='red')

Description

Creation and analysis of linear regressions fitted to log10-transformed numerical abundance versus log10-transformed body mass.

Usage

NvMLinearRegressions(community, class)

NvMSlope(community)
NvMIntercept(community)
NvMSlopeAndIntercept(community)

NvMSlopeByClass(community, class)
NvMInterceptByClass(community, class)
NvMSlopeAndInterceptByClass(community, class)
Arguments

community an object of class Community.

class the property over which linear regressions are fitted. Defaults to 'category' if the community has a node property with that name.

Value

NvMLinearRegressions returns a list of lm objects, one for each class and one fitted to all data. The list will contain NULL if it is not possible to fit a linear regression to that class; this will happen for classes that contain just a single node or that contain all or all but one nodes where x and/or y is NA.

NvMSlope, NvMIntercept and NvMSlopeAndIntercept return the slope, intercept and both, respectively, of a single linear regression fitted to all data. NvMSlopeByClass, NvMInterceptByClass and NvMSlopeAndInterceptByClass return the slope, intercept and both, respectively, of linear regressions fitted to each class and one to all data. For classes where it is not possible to fit a linear regression (for the reasons given above), NvMSlopeByClass, NvMInterceptByClass and NvMSlopeAndInterceptByClass will return NA.

Author(s)

Lawrence Hudson

See Also

Community, CommunityCollection ApplyByClass

Examples

data(TL84)

models <- NvMLinearRegressions(TL84)

# 'all', 'producer', 'invertebrate', 'vert.ecto'
names(models)

# Extract slopes and intercepts
sapply(models, coef)

# Slopes and intercepts through all data for each web in the pHWebs
# collection
data(pHWebs)
CollectionCPS(pHWebs, properties=c('NvMSlope'))
CollectionCPS(pHWebs, properties=c('NvMIntercept'))
CollectionCPS(pHWebs, properties=c('NvMSlopeAndIntercept'))

# Slopes and intercepts through each category for each web in pHWebs
CollectionCPS(pHWebs, properties=c('NvMSlopeAndInterceptByClass'))
**NvMTriTrophicStatistics**

*N-versus-M tri-trophic statistics*

---

**Description**

Tri-trophic statistics.

**Usage**

```r
NvMTriTrophicStatistics(community)
```

**Arguments**

- `community` an object of class `Community`.

**Details**

Tri-trophic statistics as described by Cohen et al 2009 PNAS.

**Value**

A list containing

- `links` a `data.frame`
- `three.node.chains` a `data.frame`
- `trophic.chains` a `data.frame`

**Author(s)**

Lawrence Hudson

**References**


**See Also**

`TLPS`, `ThreeNodeChains`, `TrophicChains`, `PlotAuppervAlower`, `NvMTriTrophicTable`
Examples

data(TL84)
tts <- NvMTriTrophicStatistics(TL84)
nrow(tts$links)
head(tts$links)

nrow(tts$three.node.chains)
head(tts$three.node.chains)

nrow(tts$trophic.chains)
head(tts$trophic.chains)

NvMTriTrophicTable

N-versus-M tri-trophic statistics

Description
Tri-trophic statistics.

Usage
NvMTriTrophicTable(collection)

Arguments

collection an object of class CommunityCollection.

Details
Returns a data.frame that contains the same statistics presented in Table 1 on Cohen et al 2009 PNAS. The function removes nodes lacking body mass (M) and/or numerical abundance (N), canni-balistic links and isolated nodes from each community. The last eight rows of the table contain four network statistics both with and without these removals.

Value
A data.frame with a column per community and the rows
Mean link length

Mean L upper
Mean L lower
2 x mean link length

Mean 2-span
Mean L upper + L lower
NvMTrTrophicTable

2 x mean link length / mean 2-span

Mean L upper + L lower / mean 2-span

Mean count chain length

Mean count chain length x mean link length

Community span

Mean count chain length x mean link length / community span

Mean sum chain lengths

Mean chain span

Mean chain span / community span

Mean sum chain lengths / mean chain span

Mean sum chain lengths / community span

L number of trophic links after removals.

S^2 number of nodes squared after removals.

L/S^2 directed connectance links after removals.

L/S linkage density after removals.

L number of trophic links before removals.

S^2 number of nodes squared before removals.

L/S^2 directed connectance links before removals.

L/S linkage density before removals.

Author(s)

Lawrence Hudson

References


See Also

NvMTriTrophicStatistics, CommunityCollection
Examples

```r
data(TL84, TL86, YthanEstuary)
collection <- CommunityCollection(list(TL84, TL86, YthanEstuary))
table <- NvMTrivTrophicTable(collection)
print(round(table, 2))
```

---

**Omnivory**

**Description**

Nodes that consume themselves in the food web.

**Usage**

- `IsOmnivore(community, level=PreyAveragedTrophicLevel)`
- `Omnivores(community, ...)`
- `FractionOmnivorous(community, ...)`
- `Omnivory(community, ...)`

**Arguments**

- `community` an object of class `Community`.
- `level` a function that returns the trophic level of each node in `community`.
- `...` other values to `IsOmnivore`.

**Details**

Omnivores are those nodes that consume two or more species and have a non-integer trophic level (Polis 1991). `IsOmnivore` returns a vector of logical of length `NumberOfNodes`; values are TRUE for nodes that are omnivorous. `Omnivores` returns the names of nodes for which `IsOmnivore` returns TRUE. `FractionOmnivorous` and `Omnivory` both return the proportion of nodes for which `IsOmnivore` returns TRUE.

Many researchers have used chain-averaged trophic level when computing omnivory (Polis, 1991; Bersier et al 2002). Computing chain-averaged trophic level requires enumerating every unique food chain - this can be lengthy for complex food webs so the default function used by `IsOmnivore` is `PreyAveragedTrophicLevel`. Omnivory values obtained using these two methods might differ slightly.

**Value**

Either a logical vector of length `NumberOfNodes` or a vector of names.

**Author(s)**

Lawrence Hudson
References


See Also

NumberOfNodes, PreyAveragedTrophicLevel, ChainAveragedTrophicLevel

Examples

data(TL84)

IsOmnivore(TL84)
Omnivores(TL84)
Omnivory(TL84)

# Omnivory values found using PreyAveragedTrophicLevel and
# ChainAveragedTrophicLevel differ for ChesapeakeBay
data(ChesapeakeBay)
Omnivory(ChesapeakeBay)
Omnivory(ChesapeakeBay, level=ChainAveragedTrophicLevel)

OrderCollection (Order a collection of communities)

Description

Order a CommunityCollection

Usage

OrderCollection(collection, ..., decreasing=FALSE)

Arguments

collection an object of class CommunityCollection.
... the names of properties by which to order the communities.
decreasing logical.

Value

A CommunityCollection.

Author(s)

Lawrence Hudson
See Also

CommunityCollection, order, CollectionCPS

Examples

data(pHWwebs)
CollectionCPS(pHWwebs, c('pH', 'NumberOfNodes'))

# Order by name
pHWwebs.name <- OrderCollection(pHWwebs, 'title')
CollectionCPS(pHWwebs.name, c('pH', 'NumberOfNodes'))

# Order by decreasing pH
pHWwebs.decreasing.pH <- OrderCollection(pHWwebs, 'pH', decreasing=TRUE)
CollectionCPS(pHWwebs.decreasing.pH, c('pH', 'NumberOfNodes'))

# Order by increasing diversity
pHWwebs.increasing.S <- OrderCollection(pHWwebs, 'NumberOfNodes')
CollectionCPS(pHWwebs.increasing.S, c('pH', 'NumberOfNodes'))

OrderCommunity

Order a community

Description

Order a Community.

Usage

OrderCommunity(community, ..., decreasing=FALSE, na.last = TRUE,
    new.order=NULL, title=NULL)

Arguments

community an object of class Community.
... the names of properties by which to order the communities.
decreasing logical.
na.last logical.
new.order a vector of either node integer indices or node names giving the order.
title the title of the new Community.

Details

Returns a new Community object. dots can contain any name that meets the criteria of the properties parameter of NPS. If new.order is NULL then ... and optionally decreasing are used to compute the new node ordering.

Different node orders will yield different SumDietGaps and SumConsumerGaps (e.g. Stouffer et al 2006, Zook et al 2011).
Value
A Community.

Author(s)
Lawrence Hudson

References

See Also
Community, order, Intervality, CollectionNPS, PreyAveragedTrophicLevel, PlotPredationMatrix

Examples

data(TL84)
NPS(TL84)

# Order by increasing M
TL84.increasing.M <- OrderCommunity(TL84, 'M', title='Increasing M')
NPS(TL84.increasing.M)

# Order by decreasing M
TL84.decreasing.M <- OrderCommunity(TL84, 'M', decreasing=TRUE)
NPS(TL84.decreasing.M)

# Order by increasing M and N
TL84.increasing.MN <- OrderCommunity(TL84, 'M', 'N')
NPS(TL84.increasing.MN)

# Reverse existing order
TL84.reversed <- OrderCommunity(TL84, new.order=56:1)
NPS(TL84.reversed)

# Sort alphabetically by category and by increasing M within each category
TL84.category <- OrderCommunity(TL84, 'category', 'M')

# Increasing trophic level, then randomly sorted within trophic level
new.order <- order(PreyAveragedTrophicLevel(TL84), sample(1:56))
TL84.increasing.TL <- OrderCommunity(TL84, new.order=new.order, title='Increasing TL')
NPS(TL84.increasing.TL)

# Graphically show the effect of different orders
par(mfrow=c(1,2))
PlotPredationMatrix(TL84.increasing.M)
**pHWebs**

**Description**

Ten stream food webs sampled across a large pH gradient.

**Usage**

```
pHWebs
```

**Format**

```
CommunityCollection.
```

**Source**

Layer et al 2010.

**References**


---

**plotauppervalower**

**Plot upper-versus-lower link angles**

**Description**

High-level function for plotting upper-versus-lower link angles.

**Usage**

```
PlotAuppervAlower(community,
               main=CPS(community)$title,
               xlab="A[lower],
               ylab="A[upper],
               xlim=c(-180, 180),
               ylim=c(-180, 180),
               pch=19,
               ...
               )
```
Arguments

community an object of class Community.
main title of the plot
xlab title of the x axis.
ylab title of the y axis.
xlim limits of the x axis.
ylim limits of the y axis.
pch plotting symbol.
... other values to plot functions.

Author(s)

Lawrence Hudson

References


See Also

NvMTriTrophicStatistics

Examples

data(TL84)
PlotAuppervalowerTL84

PlotCircularWeb  Plot circular web

Description

High-level function for plotting nodes in a circle.

Usage

PlotCircularWeb(community,
clockwise = TRUE,
origin.degrees = 0,
proportional.radius = 1,
frame.plot = FALSE,
xlim = c(-1,1),
ylim = c(-1,1),
...
Plot Helpers

Arguments

community      an object of class Community.
clockwise      logical - if TRUE nodes are plotted in a clockwise order.
origin.degrees the angle in degrees at which the first node in community will be placed.
proportional.radius a value between 0 and 1.
frame.plot     logical.
xlim           limits of the x axis.
ylim           limits of the y axis.
...            other values to PlotNPS.

Author(s)

Lawrence Hudson

See Also

Community, PlotBSpectrum, PlotWagonWheel, PlotNPS, PlotNPSDistribution, PlotNSpectrum, PlotRankNPS, PlotTLPS, PlotWebByLevel

Examples

data(TL84)
PlotCircularWeb(TL84)

# Plot the first node at the 6 o'clock position
PlotCircularWeb(TL84, origin.degrees=180)

# Plot the first node at the 6 o'clock position and plot nodes
# counter-clockwise
PlotCircularWeb(TL84, origin.degrees=180, clockwise=FALSE)
**PlotHelpers**

**Usage**

```r
Log10BLabel(community, name = "italic(B)", units = with(CPS(community),
paste(M.units, "\~", N.units)))
Log10MLabel(community, name = "italic(M)", units = CPS(community)$M.units)
Log10NLabel(community, name = "italic(N)", units = CPS(community)$N.units)
```

DefaultCategoryColours()
DefaultCategoryLabelColours()
DefaultCategorySymbols()
DefaultLinkColour()

PlaceMissingPoints(x, xlim, y, ylim)

LMabline(model, ...)
PlotLinearModels(models, colour.spec, col, ...)
FormatLM(model, slope.95.ci = FALSE, ci.plus.minus.style = FALSE,
r = FALSE, r.squared = TRUE, model.var.names = TRUE, dp = 2)

**Arguments**

- `community` an object of class `Community`.
- `name` the name that appears in the label.
- `units` the units that appears in the label.
- `x` x values.
- `y` y values.
- `xlim` limits of the x axis.
- `ylim` limits of the y axis.
- `models` a list of `lm` objects to be plotted.
- `colour.spec` either `NULL` or a named vector that maps values of `colour.by` to plotting values; defaults to the vector returned by `DefaultCategoryColours`.
- `col` plot colours.
- `model` an `lm` object for which a textual description is assembled.
- `slope.95.ci` logical - if `TRUE` then the 95% confidence intervals are included in the description.
- `ci.plus.minus.style` logical - if `TRUE` then the 95% confidence intervals are shown by a ‘plus-minus’ sign. If `FALSE` then the confidence intervals are shown by an upper and lower bound.
- `r` logical - if `TRUE` then ‘r’ is included in the description.
- `r.squared` logical - if `TRUE` then ‘r squared’ is included in the description.
- `model.var.names` logical - if `TRUE` then the names of the dependent and independent variables fitted in the model are included in the description. If `FALSE`, the names ‘x’ and ‘y’ are used.
dp the number of decimal places to which values are presented.
...
other values passed to plotting functions.

Author(s)

Lawrence Hudson

See Also

Community, DefaultCategoryColours, NvMLinearRegressions, LinearRegressionByClass, lm

PlotNPS | Plot node properties

Description

High-level functions for plotting node properties.

Usage

PlotNPS(community,
    X,
    Y,
    main = CPS(community)$title,
    xlab,
    ylab,
    xlim = NULL,
    ylim = NULL,
    colour.by,
    colour.spec,
    col = NULL,
    symbol.by,
    symbol.spec,
    pch = NULL,
    bg.by,
    bg.spec,
    bg = NULL,
    cex.by = NULL,
    cex.spec = NULL,
    cex = NULL,
    label.colour.by = NULL,
    label.colour.spec = NULL,
    label.colour = NULL,
    link.colour.by = NULL,
    link.colour.spec = NULL,
    link.col = NULL,
    link.line.type.by = NULL,
link.line.type.spec = NULL,
link.lty = NULL,
link.lwd = NULL,
highlight.links = NULL,
highlight.nodes = Cannibals,
lowlight.nodes,
show.na = FALSE,
show.web = TRUE,
show.nodes.as = "points",
node.labels = NULL,
label.cex = 0.6,
are.values = FALSE,
frame.plot = TRUE,
...)

PlotMVN(community,
  xlab = Log10NLabel(community),
  ylab = Log10MLabel(community),
  ...
)

PlotMN(community,
  xlab = Log10MLabel(community),
  ylab = Log10NLabel(community),
  ...
)

PlotBvM(community,
  xlab = Log10NLabel(community),
  ylab = Log108Label(community),
  ...
)

PlotMVb(community,
  xlab = Log108Label(community),
  ylab = Log10MLabel(community),
  ...
)

Arguments

community an object of class Community
X the name of a property that is plotted on the x axis. Must meet the criteria of the
  properties parameter of NPS. If are.values is TRUE then X and Y should be
  vectors of length NumberOfNodes.
Y plotted on the y axis; see X.
xlab title of the x axis.
ylab title of the y axis.
main title of the plot.
xlim limits of the x axis.
ylim limits of the y axis.
colour.by  node colours property. Either NULL, a vector of length NumberOfNodes or the name of a property that meets the criteria of the properties parameter of NPS.

colour.spec  node colours specification. Either NULL or a named vector that maps values of colour.by to plotting values.

col  node colours.

symbol.by  node symbols property; must meet the criteria of colour.by.

symbol.spec  node symbols specification.

pch  node symbols.

bg.by  node background colours property; must meet the criteria of colour.by.

bg.spec  node background colours specification; must meet the criteria of colour.spec.

bg  node background colours.

cex.by  node cex values property; must meet the criteria of colour.by.

cex.spec  node cex values specification; must meet the criteria of colour.spec.

cex  node cex values.

label.colour.by  node label colours property; must meet the criteria of colour.by.

label.colour.spec  node label colours specification; must meet the criteria of colour.spec.

label.colour  node label colours.

link.colour.by  link colours; either NULL, a vector of length NumberOfTrophicLinks or the name of a property that meets the criteria of the link.properties parameter of TLPS.

link.colour.spec  link line colour specification; either NULL or a named vector that maps values of link.colour.by to plotting values.

link.col  link colours.

link.line.type.by  link link types; must meet the criteria of link.colour.by.

link.line.type.spec  link line type specification; must meet the criteria of link.colour.spec.

link.lty  link line types.

link.lwd  line line widths.

highlight.links  either NULL, a vector of length NumberOfNodes or a name that meets the criteria of the properties paremeter of NPS..

highlight.nodes  nodes to be highlighted; either NULL, a vector of node names, a vector of node indices or a function that takes a Community as its only parameter and returns a vector of either node names or indices.

lowlight.nodes  nodes to be lowlighted; must meet the criteria of highlight.nodes.

show.na  logical - if TRUE then nodes for which X and/or Y is NA will be placed at the lowest extent of the relevant axis using the PlaceMissingPoints function. If FALSE then these nodes will not be plotted.
show.web logical - if TRUE and community has trophic links then the food web is plotted using the link* and highlight.links parameters
show.nodes.as how nodes should be plotted. One of
  1. "points" for symbols,
  2. "labels" for text (see node.labels, label.cex and label.colour),
  3. "points" for symbols and text.
node.labels Either NULL, a vector of length NumberOfNodes or a name that meets the criteria of the properties parameter of NPS. If NULL node labels are 1:NumberOfNodes.
label.cex a character expansion factor; used only if show.nodes.as is equal to "points".
are.values logical - if TRUE X and Y must be vectors of values of length NumberOfNodes.
frame.plot logical - default TRUE.
... other values to plot functions.

Details
The general-purpose function PlotNPS plots one node property against another.
For colour.by, symbol.by, bg.by, cex.by and label.colour.by, if X.by is not NULL and a relevant X.spec is not given, the X.by values are converted to a factor, the levels of which are used as the plot parameter. An error is raised if X.by contains any values not present in X.spec.
If colour.by/bg.by/symbol.by is NULL and community has a node property named ‘category’ then node colours/background colours/symbols are given by ‘category’ using the colour.spec/bg.spec/symbol.spec given by DefaultCategoryColours/DefaultCategorySymbols.
label.colour.by, node.labels and label.cex are used only if show.nodes.as is equal to "points".
The convenience functions PlotMvN, PlotNmV, PlotBvM and PlotMvB are ‘wrappers’ around PlotNPS that plot log10-transformed body mass (M), numerical abundance (N) or biomass (B). All of the parameters of PlotNPS, with the exception of X, Y and are.values, can be used with these four functions.
If show.nodes.as is equal to "points" then labels are plotted using label.cex and label.colour.

Author(s)
Lawrence Hudson

See Also
Community, NPS, DefaultCategoryColours, DefaultCategorySymbols, PlotBSpectrum, PlotCircularWeb, PlotNPSDistribution, PlotNSpectrum, PlotRankNPS, PlotTLPS, PlotWebByLevel PlaceMissingPoints

Examples
data(TL84)
PlotNmV(TL84)

# Set colours and plot symbols directly
PlotNmV(TL84, col=1, pch=19, highlight.nodes=NULL)
# Plot each level of taxonomic resolution in a different colour
PlotNvM(TL84, colour.by='resolved.to', pch=19, highlight.nodes=NULL)

# Plot each level of taxonomic resolution in a specific colour
colour.spec <- c(Species='purple3', Genus='green3', 'red3')
PlotNvM(TL84, colour.by='resolved.to', colour.spec=colour.spec, pch=19,
       highlight.nodes=NULL)
legend("topright", legend=names(colour.spec), pch=19, col=colour.spec)

# Use PlotNPS to plot trophic height against log10 body mass
PlotNPS(TL84, 'Log10M', 'TrophicHeight', xlab=Log10MLabel(TL84),
       ylab='Trophic height')

# The 'POM (detritus)' node in the Ythan Estuary dataset lacks both body mass
# and numerical abundance.
par(mfrow=c(1,2))
data(YthanEstuary)
PlotNvM(YthanEstuary)
PlotNvM(YthanEstuary, show.na=TRUE)

---

PlotNPSDistribution  Plot distributions of node properties

Description

High-level functions for plotting distributions of node properties.

Usage

PlotNPSDistribution(community, 
                    property, 
                    main = CPS(community)$title, 
                    density.args = list(),
                    ...
)

PlotBDistribution(community, 
                   xlab = Log10BLabel(community),
                   ...
)

PlotMDistribution(community, 
                   xlab = Log10MLabel(community),
                   ...
)

PlotNDistribution(community, 
                   xlab = Log10NLLabel(community),
                   ...
PlotRankNPS

```r
PlotDegreeDistribution(community, 
    xlab = "Number of links", 
    ...) 
```

**Arguments**

- `community` an object of class Community.
- `property` the name of a property that is plotted on the y axis. Must meet the criteria of the `properties` parameter of NPS.
- `main` title of the plot.
- `density.args` arguments passed to R’s density function.
- `xlab` title of the x axis.
- `...` other values to plot functions.

**Details**

The convenience functions PlotBDistribution, PlotMDistribution and PlotNDistribution are wrappers around PlotNPSDistribution.

**Author(s)**

Lawrence Hudson

**See Also**

Community, NPS, DegreeDistribution, PlotCircularWeb, PlotNPS, PlotNPSDistribution, PlotRankNPS, PlotTLPS, PlotWebByLevel

**Examples**

```r
data(TL84)
PlotMDistribution(TL84)

# A bandwidth of 3
PlotMDistribution(TL84, density.args=list(bw=3))

PlotDegreeDistribution(TL84)
```

---

| PlotRankNPS | Plot rank of node properties |

**Description**

High-level functions for plotting value-versus-rank of node properties.
Usage

PlotRankNPS(community,
    property,
    rank.by=property,
    log10.rank = FALSE,
    xlab,
    ylab,
    show.web=FALSE,
    ...)

PlotMvRankM(community,
    log10.rank = FALSE,
    xlab,
    ylab,
    ...)

PlotNvRankN(community,
    log10.rank = FALSE,
    xlab,
    ylab,
    ...)

PlotBvRankB(community,
    log10.rank = FALSE,
    xlab,
    ylab,
    ...)

Arguments

community      an object of class Community
property        the name of a property that is plotted on the y axis. Must meet the criteria of the
                properties parameter of NPS.
rank.by         the name of a property by which points are ordered along the x axis. Must meet
                the criteria of the properties parameter of NPS.
log10.rank      logical - if TRUE the rank values plotted on the x axis are log10-transformed.
xlab            title of the x axis.
ylab            title of the y axis.
show.web        logical - if TRUE and community has trophic links then the food web is plotted
                using the link* and highlight.links parameters
...              other values to PlotNPS.

Details

The convenience functions PlotMvRankM, PlotNvRankN and PlotBvRankB are ‘wrappers’ around
PlotRankNPS that plot rank log10-transformed body mass (M), numerical abundance (N) or biomass
(B).
**PlotTLPS**

**Author(s)**

Lawrence Hudson

**See Also**

Community, NPS, PlotBSpectrum, PlotCircularWeb, PlotNPS, PlotNPSDistribution, PlotNSpectrum, PlotTLPS, PlotWebByLevel

**Examples**

data(TL84)
PlotNvRankN(TL84)

# log10(N) against log10(rank of M)
PlotRankNPS(TL84, property='Log10N', rank.by='M', log10.rank=TRUE)

# The 'POM (detritus)' node in the Ythan Estuary dataset lacks body mass.
par(mfrow=c(1,2))
data(YthanEstuary)
PlotMvRankM(YthanEstuary)
PlotMvRankM(YthanEstuary, show.na=TRUE)

**Description**

High-level functions for plotting trophic link properties.

**Usage**

PlotTLPS(community,
X,
Y,
xlab,
ylab,
axes.limits.equal = FALSE,
xlim = NULL,
ylim = NULL,
main = CPS(community)$title,
highlight.links = NULL,
lowlight.links = NULL,
colour.by =
col = NULL,
symbol.by =
symbol.spec,
pch = NULL,
bg.by,
bg.spec,
bg = NULL,
cex.by = NULL,
cex.spec = NULL,
cex = NULL,
are.values = FALSE,
...

PlotPredationMatrix(community,
   xlab='Consumer',
   ylab='Resource',
   resource.order,
   consumer.order,
   ...
)

PlotMRvMC(community,
   xlab=Log10MLabel(community, name='italic(M)[consumer]'),
   ylab=Log10MLabel(community, name='italic(M)[resource]'),
   axes.limits.equal = TRUE,
   ...
)

PlotMCvMR(community,
   xlab=Log10MLabel(community, name='italic(M)[resource]'),
   ylab=Log10MLabel(community, name='italic(M)[consumer]'),
   axes.limits.equal = TRUE,
   ...
)

PlotNRvNC(community,
   xlab=Log10NLable(community, name='italic(N)[consumer]'),
   ylab=Log10NLable(community, name='italic(N)[resource]'),
   axes.limits.equal = TRUE,
   ...
)

PlotNCvNR(community,
   xlab=Log10NLable(community, name='italic(N)[resource]'),
   ylab=Log10NLable(community, name='italic(N)[consumer]'),
   axes.limits.equal = TRUE,
   ...
)

PlotBRvBC(community,
   xlab=Log10BLabel(community, name='italic(B)[consumer]'),
   ylab=Log10BLabel(community, name='italic(B)[resource]'),
   axes.limits.equal = TRUE,
   ...
)

PlotBCvBR(community,
Arguments

- **community**: an object of class `Community`.
- **X**: the name of a node or link property to plot on the x axis. If the name begins with 'resource.' or 'consumer.', the remainder of the name is assumed to be a node property and should meet the criteria of the `node.properties` parameter of TLPS, otherwise the name is assumed to be a link property and should meet the criteria of the `link.properties` parameter of TLPS.
- **Y**: plotted on the y axis; see X.
- **xlab**: title of the x axis.
- **ylab**: title of the y axis.
- **axes.limits.equal**: logical - if TRUE and `xlim` and `ylim` are NULL then the limits of the x and y axes will be the same.
- **xlim**: limits of the x axis
- **ylim**: limits of the y axis
- **main**: title of the plot
- **highlight.links**: trophic links to be highlighted; either NULL, a vector of trophic link indices or a function that takes a Community as its only parameter and returns a data.frame containing the columns 'resource' and 'consumer', which should contain node names.
- **lowlight.links**: trophic links to be lowlighted; should meet criteria of `lowlight.links`.
- **colour.by**: trophic link colours property. Either NULL, a vector of length `NumberOfTrophicLinks` or a name. If the name begins with 'resource.' or 'consumer.', the remainder of the name is assumed to be a node property and should meet the criteria of the `node.properties` parameter of TLPS, otherwise the name is assumed to be a link property and should meet the criteria of the `link.properties` parameter of TLPS.
- **colour.spec**: trophic links colours specification. Either NULL or a named vector that maps values of `colour.by` to plotting values.
- **col**: trophic links colours.
- **symbol.by**: trophic links symbols property; must meet the criteria of `colour.by`.
- **symbol.spec**: trophic links symbols specification specification; must meet the criteria of `colour.spec`.
- **pch**: trophic links symbols.
- **bg.by**: trophic links background colours property; must meet the criteria of `colour.by`.
- **bg.spec**: trophic links background colours specification; must meet the criteria of `colour.spec`.

```r
xlab=Log10BLabel(community, name='italic(B)[resource]'),
ylab=Log10BLabel(community, name='italic(B)[consumer]'),
axes.limits.equal = TRUE,
...)
```
PlotTLPS

bg trophic links background colours.
cex.by trophic links cex property; must meet the criteria of colour.by
cex.spec cex values specification; must meet the criteria of colour.spec.
cex cex values.
are.values logical - if TRUE X and Y must be vectors of values of length NumberOfTrophicLinks.
resource.order the order in which to show resources. Either missing, which implies the native node order, a vector of length NumberOfTrophicLinks containing the integer order of resources, or the name of a property that meets the criteria of the properties parameter of NPS.
consumer.order the order in which to show consumer; requirements are the same as resource.order.
... other values to plot functions.

Details

The general-purpose function PlotTLPS plots one trophic-link property against another.
If colour.by/bg.by/symbol.by is NULL and community has a node property named ‘category’ then trophic-link colours/background colours/symbols are given by ‘resource.category’ using colour.spec/bg.spec/symbol.spec given by DefaultCategoryColours/DefaultCategorySymbols.

PlotPredationMatrix shows trophic links as a binary matrix with species shown in node order, starting at the top-left. If row.node and col.order are both missing (the default) or are the same, then a dashed diagonal line is drawn. Points on the dashed line indicate cannibalistic trophic links.

The convenience functions PlotMRvMC, PlotMCvMR, PlotNRvNC, PlotNCvNR, PlotBRvBC, PlotBCvBR are ‘wrappers’ around PlotRankNPS that plot a log10-transformed body mass, M, numerical abundance, N, or biomass abundance, B.

Author(s)

Lawrence Hudson

See Also

Community, TLPS, PlotBSpectrum, PlotCircularWeb, PlotNPS, PlotNPSDistribution, PlotNSpectrum, PlotRankNPS, PlotWebByLevel

Examples

data(TL84)

# The predation matrix
PlotPredationMatrix(TL84)

# The predation matrix with rows ordered by body mass
PlotPredationMatrix(TL84, resource.order='M')

# Colours and symbols by resource.category
PlotMCvMR(TL84)
# Colours and symbols by consumer\_category
PlotMCvMR(TL84, bg\_by='consumer\_category', symbol\_by='consumer\_category',
colour\_by='consumer\_category')

# Consumer trophic height against resource log10(M)
PlotTLPS(TL84, 'resource\_Log10M', 'consumer\_TrophicHeight')

# Log10(M of resource / M of consumer) against consumer log10(M)
PlotTLPS(TL84, 'consumer\_Log10M', 'Log10RCMRatio')

---

### PlotWagonWheel

#### Description

Plot a nodes as concentric circles around a focal node.

#### Usage

```r
PlotWagonWheel(community,
focus,
clockwise=TRUE,
origin\_degrees=0,
frame\_plot=FALSE,
main=NULL,
...)```

#### Arguments

- **community**: an object of class `Community`.
- **focus**: the node to be placed at the centre of the plot - a node name or integer index.
- **clockwise**: logical - if TRUE nodes are plotted in a clockwise order.
- **origin\_degrees**: the angle in degrees at which the first node in `community` will be placed.
- **frame\_plot**: logical.
- **main**: the plot’s title.
- **...**: other values to `PlotNPS`.

#### Details

The node given in `focus` is plotted at the centre of the wagon wheel. Other nodes in the community are plotted in concentric circles that are one, two, three etc trophic links away from the focus. Isolated nodes - those with no resources or consumers, other than possibly themselves - are not shown.

#### Author(s)

Lawrence Hudson
PlotWebByLevel

See Also

Community, Degree, IsolatedNodes, OrderCommunity, PlotCircularWeb, PlotNPS, PlotWebByLevel, rgb, ShortestPaths

Examples

data(TL84)
# Ploesoma sp. is the focal species
PlotWagonWheel(TL84, 'Ploesoma sp. ')

# Show nodes as numbers
PlotWagonWheel(TL84, 'Ploesoma sp.', show.nodes.as='labels')

# 'Daphnia pulex' is the focus, nodes ordered by degree (total number of
# trophic links), lines partially transparent
PlotWagonWheel(OrderCommunity(TL84, 'Degree'), 'Daphnia pulex',
               show.nodes.as='labels', link.col=rgb(0.8,0.8,0.8,0.5))

---

PlotWebByLevel  Plot web by level

Description

A high-level function for plotting a food-web by vertically with the lowest trophic-level nodes at the bottom.

Usage

PlotWebByLevel(community,
                level='PreyAveragedTrophicLevel',
                max.nodes.per.row=20,
                round.levels.to.nearest=0.2,
                stagger=0.1,
                x.layout='wide',
                y.layout='compress',
                show.level.labels=TRUE,
                show.level.lines=FALSE,
                xaxt='n',
                yaxt='n',
                xlab='',
                ylab='',
                frame.plot=FALSE,
                ylim=NULL,
                ...)
Arguments

- **community**: an object of class Community
- **level**: either a function, a name that meets the criteria of the properties parameter of NPS or a vector of length NumberOfNodes, which must contain numbers greater than 0.
- **max.nodes.per.row**: a number greater than 2.
- **round.levels.to.nearest**: a number greater or equal to 0 and less than 1.
- **stagger**: a number greater or equal to 0 and less than 1. Only used if y.layout is 'stagger'.
- **x.layout**: 'skinny', 'narrow' or 'wide'.
- **y.layout**: 'stagger' or 'compress'. Only has an effect if round.levels.to.nearest is greater than 0.
- **show.level.labels**: logical - if TRUE then integer values of level are shown to the left of the plot.
- **show.level.lines**: logical - if TRUE then a horizontal line is drawn for each unique value of level.
- **xaxt**: a character that specifies the type of the x axis.
- **yaxt**: a character that specifies the type of the y axis.
- **xlab**: title of the x axis.
- **ylab**: title of the y axis.
- **frame.plot**: logical - if TRUE then a border is drawn around the plot.
- **ylim**: limits of the y axis
- **...**: other values to PlotNPS.

Details

If round.levels.to.nearest is greater than 0, values in level are rounded to the nearest round.levels.to.nearest. Rounded values are used by the x.layout and y.layout engines.

If x.layout is 'skinny' then nodes are spaced one x unit apart and max.nodes.per.row is ignored. If x.layout is 'narrow', nodes are spaced one x unit apart if fewer than max.nodes.per.row on that row, otherwise nodes are squashed in to the available x space. If x.layout is 'wide', nodes are spaced widely.

If y.layout is 'compress', then nodes are always shown at the values in level. If y.layout is 'stagger' and there are more than max.nodes.per.row on a level then the plotted levels are staggered by the values in stagger.

Author(s)

Lawrence Hudson
PredationMatrix

Description

Returns a predation matrix.

Usage

PredationMatrix(community, weight=NULL)
**Arguments**

- `community` an object of class `Community`.
- `weight` either the name of a first-class link property or the name of a function that meets the specification of the `link.properties` parameter of TLPS.

**Details**

Returns a square matrix with `NumberOfNodes` rows and columns. If `weight` is `NULL` then a binary matrix, in which elements are either 0 or 1, is returned; 1 indicates a trophic link from a resource (row) to a consumer (column). If `weight` is not `NULL` then elements of the returned matrix will be set to the values given by `weight`. Row names and column names of the returned matrix are node names.

**Value**

A square matrix.

**Author(s)**

Lawrence Hudson

**See Also**

- `PlotPredationMatrix`, `TLPS`, `NumberOfNodes`, `NumberOfTrophicLinks`, `ResourcesByNode`, `ConsumersByNode`, `PredationMatrixToLinks`

**Examples**

data(TL84)

# A square matrix of `NumberOfNodes` rows and columns
dim(PredationMatrix(TL84))
NumberOfNodes(TL84)

# Should contain `NumberOfTrophicLinks` links
sum(PredationMatrix(TL84))
NumberOfTrophicLinks(TL84)

# Compare an unweighted matrix and a matrix weighted by diet fraction
data(Benguela)

PredationMatrix(Benguela)
PredationMatrix(Benguela, weight='diet.fraction')
PredationMatrixToLinks

Predation matrix to trophic links

Description

A function that converts a predation matrix to a data.frame with the columns ‘resource’ and ‘consumer’.

Usage

PredationMatrixToLinks(pm, link.property=NULL)

Arguments

pm a matrix or data.frame.
link.property either NULL or a character.

Details

Returns a data.frame of trophic links contained within pm. Non-zero and non-NA values indicate a trophic link between a resource (row) and consumer (column). pm should have both row names and column names. The returned data.frame will contain the columns ‘resource’ and ‘consumer’. If pm contains quantitative information such as diet fractions or number of observations then you can set link.property to the name of the quantity and the returned data.frame will include a column with that name, that contains link strength values extracted from pm.

If you have existing food-web data in predation-matrix form then this function can help to import your data in to Cheddar.

Value

A data.frame

Author(s)

Lawrence Hudson

See Also

Community, PredationMatrix, TLPS
Pyramid plots

Examples

data(TL84)

links <- PredationMatrixToLinks(PredationMatrix(TL84))
identical(links, TLPS(TL84))  # TRUE

# Create a Cheddar community from an existing square predation matrix
node <- c('Leaf', 'Caterpillar', 'Bluetit')
pm <- matrix( c(0, 1, 0,
               0, 0, 1,
               0, 0, 0),
              ncol=3, byrow=TRUE, dimnames=list(node, node))

community1 <- Community(nodes=data.frame(node=node),
                        trophic.links=PredationMatrixToLinks(pm),
                        properties=list(title='Test community'))
TLPS(community1)

# The same set of trophic links could be represented by a non-square predation matrix
pm <- matrix( c(1, 0,
               0, 1),
              ncol=2, byrow=TRUE, dimnames=list(node[1:2], node[2:3]))

community2 <- Community(nodes=data.frame(node=node),
                        trophic.links=PredationMatrixToLinks(pm),
                        properties=list(title='Test community'))
TLPS(community2)

all.equal(community1, community2)  # TRUE

# Extract quantitative information
node <- c('Leaf 1', 'Leaf 2', 'Caterpillar 1', 'Caterpillar 2')
pm <- matrix( c(0, 0.4, 0.8,
               0, 0.6, 0.2,
               0, 0, 0, 0,
               0, 0, 0, 0),
              ncol=4, byrow=TRUE, dimnames=list(node, node))

# A data.frame that has a column called diet.fraction
PredationMatrixToLinks(pm, link.property='diet.fraction')

Pyramid plots

Description
High-level functions that create pyramid plots.
Pyramid plots

Usage

PlotBPpyramid(community,
  level = floor(PreyAveragedTrophicLevel(community)),
  expected.levels, 
  fill.missing.levels = TRUE,
  order.by.expected = TRUE, 
  show.level.labels = TRUE, 
  xlab = Log10BLabel(community, name=expression (~sum(italic(B)))), 
  ylab = "", 
  xlim = NULL, 
  col = NULL, 
  text.col = 'black', 
  main = CPS(community)$title,
  ...)

PlotNPpyramid(community,
  level = floor(PreyAveragedTrophicLevel(community)),
  expected.levels, 
  fill.missing.levels = TRUE,
  order.by.expected = TRUE, 
  show.level.labels = TRUE, 
  xlab = Log10NLabel(community, name=expression (~sum(italic(N)))), 
  ylab = "", 
  xlim = NULL, 
  col = NULL, 
  text.col = 'black', 
  main = CPS(community)$title,
  ...)

Arguments

community an object of class Community
level levels by which values are summed. Can be either the name of a node property, in which case it must meet the criteria of the properties parameter of NPS, or a vector of length NumberOfNodes that contains the levels.
expected.levels the values that are expected to be in level; see Details for more information.
fill.missing.levels if TRUE, values in expected.levels that are not present in level are shown in the pyramid.
order.by.expected if TRUE then the levels are plotted in the order given in expected.levels.
show.level.labels logical - if TRUE then values of level are shown to the left of the pyramid.
xlab title of the x axis.
ylab title of the y axis.
Pyramid plots

xlim
limits of the x axis.
col
fill colour; either a single colour a vector containing a colour per level.
text.col
colour for the text showing the log10-transformed sums in the blocks of the pyramid; ; either a single colour a vector containing a colour per level.
main
title of the plot.
... other values to plot functions.

Details

PlotBPyramid plots log10-transformed sum biomass abundance in each level and PlotNPyramid plots log10-transformed sum numerical abundance in each level.

expected.levels provides two behaviours. First, it provides error checking: an error is raised if values are in level that are not in expected.levels. Second, it interacts with fill.missing.levels and order.by.expected which levels are drawn and how. If fill.missing.levels is TRUE then values in expected.levels that are not present in level are shown on the pyramid plot. If order.by.level is TRUE then the levels are plotted in the order given in expected.levels. If level contains numbers then expected.levels defaults to a sequence of integers floor(min(level)):ceiling(max(level)). If level is ‘category’ then expected.levels defaults to the intersection of values of ‘category’ that are present in community and the usual Cheddar default values: ‘<unnamed>’, ‘producer’, ‘invertebrate’, ‘vert.ecto’, ‘vert.endo’.

Author(s)

Lawrence Hudson

See Also

Community, SumBiomassByClass, SumNByClass, Log10BLabel, Log10NLabel, floor, ceiling

Examples

data(TL84)

# Use a large left-hand margin to show level text
reset.par <- par(mar=c(5,8,1,1))

# Using prey-averaged trophic level
PlotNPyramid(TL84)

# Using chain-averaged of trophic level
PlotNPyramid(TL84, level=floor(ChainAveragedTrophicLevel(TL84)))

# Show by category
PlotNPyramid(TL84, level='category')

# Taxonomic kingdoms as levels
PlotNPyramid(TL84, level='kingdom')

# Taxonomic kingdoms as levels, with a defined order

Usage

NodeQuantitativeDescriptors(community, weight)
QuantitativeDescriptors(community, weight, top.level.threshold=0.99)

Arguments

community an object of class Community.
weight  
the name of a tropic-link property with which quantitative descriptors should be 
computed. It can be the name of a first-class property (returned by TrophicLinkPropertyNames) or the name of a function that takes a Community object as the only parameter and a vector of length NumberOfTrophicLinks.

top.level.threshold  
TODO

Details

Quantitative food-web descriptors as described by Bersier et al 2002 Ecology.

NodeQuantitativeDescriptors computes a table of node-level quantitative descriptors, as presented in Bersier et al 2002, Table 1. It returns a matrix with columns NResources, NConsumers, bIn, bOut, nN, nP, d.prime, d, o.prime, o, g.prime, g, v.prime, v.

QuantitativeDescriptors computes values presented in Bersier et al 2002 Table 2. It returns a matrix with columns Qualitative, Unweighted and Weighted and rows Fraction top level, Fraction intermediate, Fraction basal, Ratio resources:consumers, Link density, Connectance, Fraction links top:intermediate, Fraction links top:basal, Fraction links intermediate:intermediate, Fraction links intermediate:basal, Mean chain length, Median chain length, SD chain length, Max chain length, Degree of omnivory, Generality, Vulnerability, SD standardised generality, SD standardised vulnerability.

Value

A matrix.

Author(s)

Lawrence Hudson

References


See Also

TrophicLinkPropertyNames, NumberOfTrophicLinks, NumberOfNodes

Examples

data(ChesapeakeBay)
QuantitativeDescriptors(ChesapeakeBay, 'biomass.flow')
NodeQuantitativeDescriptors(ChesapeakeBay, 'biomass.flow')
RemoveCannibalisticLinks

Remove cannibalistic trophic links

Description
Remove cannibalistic trophic links.

Usage
RemoveCannibalisticLinks(community, title)

Arguments
- community: an object of class Community.
- title: a title for the new community.

Details
Returns a new Community with any cannibalistic trophic links removed.

Value
A new object of class Community.

Author(s)
Lawrence Hudson

See Also
Community

Examples
data(TL84)
NumberOfTrophicLinks(TL84)

TL84.no.cannibal <- RemoveCannibalisticLinks(TL84)
NumberOfTrophicLinks(TL84.no.cannibal)
RemoveIsolatedNodes  Remove isolated nodes

Description
Remove isolated nodes.

Usage
RemoveIsolatedNodes(community, title)

Arguments
<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>community</td>
<td>an object of class Community.</td>
</tr>
<tr>
<td>title</td>
<td>a title for the new community.</td>
</tr>
</tbody>
</table>

Details
Returns a new Community with isolated nodes removed.

Value
A new object of class Community.

Author(s)
Lawrence Hudson

See Also
Community, IsIsolatedNode

Examples
```r
data(TL84)
IsolatedNodes(TL84)

TL84.no.isolated <- RemoveIsolatedNodes(TL84)
IsolatedNodes(TL84.no.isolated)
```
RemoveNodes

Description
Remove one or more nodes.

Usage
RemoveNodes(community, remove, title,
method=c('direct','secondary','cascade'))

Arguments
- community: an object of class Community.
- remove: a vector of either names, integer indices or logicals indicating nodes to be removed.
- title: a title for the new community.
- method: how species removals are propagated through the food web.

Details
Returns a new Community with nodes in remove removed. An error is raised if remove refers to nodes not in the community or if remove refers to all nodes in the community.

If method is 'direct', only the nodes in remove are removed. If method is 'secondary', secondarily extinct nodes - those that directly consume one or more nodes in 'remove' and that no longer have any resources (except themselves) after the removal - are also removed. If method is 'cascade', a multistep version of 'secondary' is applied. This has the effect of propogating extinctions though the community - all consumers that are ultimately dependent upon all species in 'remove', and upon no other nodes (except themselves), will be removed.

Value
A new object of class Community.

Author(s)
Lawrence Hudson

See Also
Community, BasalNodes, IsolatedNodes, NumberOfNodes
Examples

data(TL84)

# Three different ways of removing node 56 (Umbrla limi)
a <- RemoveNodes(TL84, 56)
b <- RemoveNodes(TL84, 'Umbrla limi')
c <- RemoveNodes(TL84, c(rep(FALSE, 55), TRUE))

identical(a, b)  # TRUE
identical(a, c)  # TRUE

# The behaviours of the different methods
NumberOfNodes(TL84)  # 56 nodes in total
length(BasalNodes(TL84))  # 25 basal nodes
length(IsolatedNodes(TL84))  # 6 isolated nodes

RemoveNodes(TL84, BasalNodes(TL84))  # 56 - 25 = 31 nodes remain
RemoveNodes(TL84, BasalNodes(TL84), method='secondary')  # 14 nodes remain
RemoveNodes(TL84, BasalNodes(TL84), method='cascade')  # 6 isolated nodes remain

# Results in an error
## Not run: RemoveNodes(TL84, 1:NumberOfNodes(TL84))

---

ResourceLargerThanConsumer

*Resource larger than consumer*

Description

Trophic links in which the resource has a larger body mass than the consumer.

Usage

```
ResourceLargerThanConsumer(community)
```

Arguments

- **community** an object of class `Community`

Details


Value

A `data.frame`

Author(s)

Lawrence Hudson
ResourcesByNode

See Also

Community

Examples

data(TL84)

ResourceLargerThanConsumer(TL84)

# Highlight trophic links
PlotNVM(TL84, highlight.links=ResourceLargerThanConsumer)

Description

Functions that return the resources and consumers of nodes.

Usage

ResourcesByNode(community)
ConsumersByNode(community)
ResourcesAndConsumersByNode(community)

ResourcesOfNodes(community, nodes)
ConsumersOfNodes(community, nodes)

TrophicLinksForNodes(community, nodes, node.properties=NULL,
    link.properties=NULL)

Arguments

community an object of class Community.
nodes either the names or integer indices of nodes.
node.properties passed to TLPS.
link.properties passed to TLPS.

Details

ResourcesByNode/ConsumersByNode/ ResourcesAndConsumersByNode all return a list of length NumericUpDownNodes; list elements are names of nodes that are resources/consumers/resources and/or consumers.

If nodes is of length one then ResourcesOfNodes and ConsumersOfNodes return a vector of resources / consumers. If nodes contains more than one value, then a list of vectors is returned.
TrophicLinksForNodes returns a data.frame containing the columns ‘resource’ and ‘consumer’ and a row for each trophic link in-to and out-of nodes.

Value

Either a vector, a list or a data.frame

Author(s)

Lawrence Hudson

See Also

TLPS, PredationMatrix, NumberOfNodes

Examples

data(TL84)

# A list containing a vector of resources for each node.
ResourcesByNode(TL84)

# A vector of resources of 'Umbra limi'
ResourcesOfNodes(TL84, 'Umbra limi')

# A vector of resources of 'Umbra limi'
ResourcesOfNodes(TL84, 56)

# A list containing vectors of resources for nodes 50:56
ResourcesOfNodes(TL84, 50:56)

# A data.frame containing columns resource and consumer
TrophicLinksForNodes(TL84, 'Umbra limi')

# A data.frame containing columns resource, consumer, resource.M and consumer.M
TrophicLinksForNodes(TL84, 'Umbra limi', node.properties='M')

<table>
<thead>
<tr>
<th>ShortestPaths</th>
<th>Path lengths</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Description

Functions that compute the shortest trophic paths between nodes.

Usage

ShortestPaths(community, weight.by=NULL)
CharacteristicPathLength(community)
Arguments

- **community**: an object of class `Community`.
- **weight.by**: the name of a property by which to weight paths.

Details

ShortestPaths uses Dijkstra’s algorithm to compute the number of trophic links between each pair of nodes in the food web. CharacteristicPathLength returns the mean of path lengths.

Value

A square matrix with NumberOfNodes rows and columns or a single number.

Author(s)

Lawrence Hudson

References


See Also

- `PredationMatrix`, `NPS`

Examples

```r
data(Benguela)

# Compare weighted and unweighted
ShortestPaths(Benguela)
ShortestPaths(Benguela, weight.by='diet.fraction')

CharacteristicPathLength(Benguela)
```

---

**SiteBySpeciesMatrix**

*Community collection site by species matrix*

Description

Returns a matrix with a column per community and a row per unique node within communities in the collection.

Usage

```r
SiteBySpeciesMatrix(collection, abundance=NULL, na.missing=FALSE)
```
**SiteBySpeciesMatrix**

**Arguments**

- `collection` an object of class `CommunityCollection`.
- `abundance` the name of a node property that provides abundance values. This can be the name of a first-class property or the name of a function. The name must meet the criteria of the properties parameter of NPS.
- `na.missing` if `TRUE` nodes that are absent from the community will be represented by `NA`; if `FALSE`, these nodes are represented by `0`.

**Details**

If `abundance` is `NULL`, the returned matrix indicates presence (1) or absence (0 or `NA` - see `na.missing`) of nodes. If `abundance` is given, values are the abundances of nodes, or 0 or `NA` where nodes are absent.

**Value**

A matrix.

**Author(s)**

Lawrence Hudson

**See Also**

`CommunityCollection`, `NPS`, `CollectionCPS`, `Biomass`, `Log10Biomass`, `matrix`

**Examples**

data(pHWebs)

# If abundance is NULL, you get a presence/absence matrix:
SiteBySpeciesMatrix(pHWebs)

# Numerical abundance
SiteBySpeciesMatrix(pHWebs, 'N')

# Biomass abundance
SiteBySpeciesMatrix(pHWebs, 'Biomass')

# Log10 biomass abundance
SiteBySpeciesMatrix(pHWebs, 'Log10Biomass')

# Example showing how to model biomass in term of pH using vegan's rda function
m <- SiteBySpeciesMatrix(pHWebs, 'Biomass')

# Some nodes (e.g. CPOM) do not have a biomass. The rows in m for these nodes
# will contain all NA. Rows containing all NA will upset vegan's rda function
# so these rows must be removed.
m <- m[apply(m, 1, function(row) all(!is.na(row))),]
# Tranpose to get row per site - the format required by vegan’s rda function
m <- t(m)

# Matrix (with a row per site) of variables on the right hand side of the
# model equation
variables <- CollectionCPS(pHWebs)

## Not run: library(vegan)
## Not run: res <- rda(m~pH,variables)

Description
The food-web of Skipwith Pond.
Taxonomic classification provided by Guy Woodward.

Usage
SkipwithPond

Format
Community.

Source

References

Description
High-level functions that plot the sum numerical abundance (N) or biomass abundance (B) in equally-spaced log10 body-mass bins.
Usage

PlotBSpectrum(community,
        lower = min(NP(community, "M"), na.rm = TRUE),
        upper = max(NP(community, "M"), na.rm = TRUE),
        n.bins = 10,
        main = CPS(community)$title,
        xlab = Log10MLabel(community),
        ylab = Log10Label(community),
        xlim = NULL,
        ylim = NULL,
        pch = 19,
        show.bin.limits = TRUE,
        show.bin.centres = FALSE,
        ...
)

PlotNSpectrum(community,
        lower = min(NP(community, "M"), na.rm = TRUE),
        upper = max(NP(community, "M"), na.rm = TRUE),
        n.bins = 10,
        main = CPS(community)$title,
        xlab = Log10MLabel(community),
        ylab = Log10Label(community),
        xlim = NULL,
        ylim = NULL,
        pch = 19,
        show.bin.limits = TRUE,
        show.bin.centres = FALSE,
        ...
)

Arguments

  community   an object of class Community.
  lower       lower bound of the bins.
  upper       upper bound of the bins.
  n.bins      the number of bins.
  main        title of the plot
  xlab        title of the x axis.
  ylab        title of the y axis.
  xlim        limits of the x axis.
  ylim        limits of the y axis.
  pch         plotting symbol.
  show.bin.limits
               logical - if TRUE the centres of the bins are marked with a line.
  show.bin.centres
               logical - if TRUE the centres of the bins are marked with a line.
  ...          other values to plot functions.
subset.CommunityCollection

Value

A list:

- bins: value returned by the BodyMassBins function.
- lm: a linear regression fitted through the data.

Author(s)

Lawrence Hudson

See Also

Community, BodyMassBins, PlotCircularWeb, PlotNPS, PlotNPSDistribution, PlotRankNPS, PlotTLPS, PlotWebByLevel

Examples

data(TL84)
PlotNSpectrum(TL84)
PlotBSpectrum(TL84)

Description

A subset of a collection of communities.

Usage

```r
## S3 method for class 'CommunityCollection'
subset(x, subset, properties=NULL, ...)
```

Arguments

- `x`: An object of class CommunityCollection
- `subset`: logical expression indicating communities to keep.
- `properties`: The names of properties passed to CollectionCPS.
- `...`: further arguments passed to other methods.

Details

CollectionCPS is used to gather properties. Properties should contain the names of properties required to evaluate subset. If properties is NULL, all first-class properties are available to the subset expression. Returns a new CommunityCollection or NULL if no communities in x meet the criteria in subset.
**ThreeNodeChains**

**Value**

A new object of class `CommunityCollection` or `NULL`.

**Author(s)**

Lawrence Hudson

**See Also**

`CommunityCollection`, `CollectionCPS`, `subset`

**Examples**

```r
data(pHwebs)

# Two communities have pH>7
subset(pHwebs, pH>7)

# No communities have pH>10 so this returns NULL
subset(pHwebs, pH>7)

# Get a subset based on a computed property
subset(pHwebs, S>50, properties=c(S='NumberOfNodes'))

# X is not a property so this raises an error
## Not run: subset(pHwebs, X==1)
```

**Description**

Enumerates every three-node chain in a food web.

**Usage**

```r
ThreeNodeChains(community, exclude.loops=FALSE, node.properties=NULL, chain.properties=NULL)
```

**Arguments**

- **community**: an object of class `Community`.
- **exclude.loops**: logical - should loops A -> B -> A be included?
- **node.properties**: the names of the node properties to return. Should meet the criteria of the `properties` parameter of `NPS`.
- **chain.properties**: the names of chain properties to return.
Details

Enumerates every three-node chain in the food-web and returns a data.frame containing the columns bottom, intermediate and top and any requested node and trophic-link columns.

Value

A data.frame.

Author(s)

Lawrence Hudson

See Also

TLPS, TrophicChains

Examples

data(TL84)

nrow(ThreeNodeChains(TL84))
nrow(ThreeNodeChains(TL84, exclude.loops=TRUE))

# bottom, intermediate and top
head(ThreeNodeChains(TL84))

# bottom, intermediate, top, bottom.M, intermediate.M and top.M
head(ThreeNodeChains(TL84, node.properties='M'))

# As above with the addition of bottom.N, intermediate.N and top.N
head(ThreeNodeChains(TL84, node.properties=c('M','N')))

---

TL84  Tuesday Lake datasets

Description

The communities of Tuesday Lake, Michigan, USA sampled in 1984 and 1986.

Usage

TL84
TL86

Format

Community objects.
Source

References

TLP

A single trophic-link property

Description
Returns a single trophic-link property.

Usage
TLP(community, property)

Arguments

<table>
<thead>
<tr>
<th>community</th>
<th>an object of class Community.</th>
</tr>
</thead>
<tbody>
<tr>
<td>property</td>
<td>the name of the property to return.</td>
</tr>
</tbody>
</table>

Details
This function is named TLP for Trophic Link Property. It returns a vector containing the value of property for every trophic link. The returned vector is all NA if there is no trophic-link property with that name.

Value
A vector of length NumberOfTrophicLinks.
Author(s)

Lawrence Hudson

See Also

TrophicLinkPropertyNames, TLPS, NumberOfTrophicLinks

Examples

# Skipwith Pond has a first-class property called link.evidence
data(SkipwithPond)
TLP(SkipwithPond, 'link.evidence')

# Benguela has a first-class property called diet.fraction
data(Benguela)
TLP(Benguela, 'diet.fraction')

# All NA
TLP(SkipwithPond, 'not a property')

---

TLPS  Trophic-link properties

Description

Returns a data.frame of first-class and computed trophic-link properties.

Usage

TLPS(community, node.properties=NULL, link.properties=NULL)

Arguments

- **community**: an object of class Community.
- **node.properties**: the names of the node properties to return. Should meet the criteria of the properties parameter of NPS.
- **link.properties**: the names of link properties. These can be names of first-class properties (returned by TrophicLinkPropertyNames) and names of functions that take a Community object as the only parameter and return either a vector of length NumberOfTrophicLinks or a matrix or data.frame with NumberOfTrophicLinks rows.

Details

This function is named TLPS for Trophic Link Properties. It returns a data.frame containing the columns 'resource' and 'consumer' and any requested properties.
**TrophicChains**

**Value**

A data.frame with NumberOfTrophicLinks rows.

**Author(s)**

Lawrence Hudson

**See Also**

TrophicLinkPropertyNames, TLP, NumberOfTrophicLinks, NPS, Log10RCMRatio, ThreeNodeChains, TrophicChains

**Examples**

data(TL84)

# Just resource and consumer
head(TLPS(TL84))

# resource, consumer, resource.M and consumer.M
head(TLPS(TL84, node.properties="M"))

# Log10RCMRatio returns log10-transformed resource.M / consumer.M
head(TLPS(TL84, node.properties="M", link.properties='Log10RCMRatio'))

# Skipwith Pond has link.evidence and link.life.stage first-class properties
data(SkipwithPond)
head(TLPS(SkipwithPond))

# resource, consumer and link.evidence
head(TLPS(SkipwithPond, link.properties='link.evidence'))

# Skipwith Pond has diet.fraction first-class property
data(Benguela)
head(TLPS(Benguela))

---

**TrophicChains**  |  **Trophic chains**

**Description**

Enumerates every trophic chain in a food web.

**Usage**

TrophicChains(community, node.properties = NULL, chain.properties = NULL)
Arguments

community an object of class Community.
node.properties the names of the node properties to return. Should meet the criteria of the properties parameter of nps.
chain.properties the names of chain properties to return.

Details

Enumerates every trophic chain in the food-web and returns a data.frame containing any requested node and trophic-link columns.

Some network properties and analyses require knowledge of every unique path - ‘trophic chain’ - through the food-web. A trophic chain starts at a basal node (BasalNodes) and ends when it is not possible to add nodes that are not already in the chain, so loops and cannibalism are ignored. For communities that have one or more top-level nodes (TopLevelNodes) each trophic chain will end with a top-level node.

If your analysis requires only simple statistics about trophic chains, the TrophicChainsStats function is more suitable as it is much faster and requires less memory than TrophicChains. This is particularly true for communities that contain a large number of trophic chains, such as the SkipwithPond dataset, which has more than $10^5$ unique chains.

It will not be possible to compute, within reasonable time and available system memory, trophic chains for food webs with a large number of nodes and/or trophic links. TrophicChains will raise an error ‘Unable to compute paths’ for these food webs. The ‘Large numbers of trophic chains’ section of the ‘Community’ vignette explains this in more detail.

Value

A data.frame.

Author(s)

Lawrence Hudson

See Also

BasalNodes, TopLevelNodes, TLPS, ThreeNodeChains, TrophicChainsStats, SkipwithPond

Examples

data(TL84)
tc <- TrophicChains(TL84)

# Every chain starts with a basal node
BasalNodes(TL84)
first <- tc[,1]
all(IsBasalNode(TL84)[unique(first)])
# TL84 has a single top-level consumer - every trophic chain ends with this
# consumer
TrophicChainsStats(TL84)

# Get the last node in each chain
last <- apply(tc, 1, function(row) row[which("!"=row)])
unique(last)

# M of nodes
head(TrophicChains(TL84, node.properties='M'))

# M and N of nodes
head(TrophicChains(TL84, node.properties=c('M','N')))

# Skipwith Pond has more than 10e5 unique chains
data(SkipwithPond)

# Not all systems will be able to allocate the memory required to hold the
# chains
## Not run: dim(TrophicChains(SkipwithPond))

TrophicChainsStats  Trophic chains statistics

Description

Computes simple statistics about every trophic chain in a food web.

Usage

TrophicChainsStats(community)

Arguments

community an object of class Community.

Details

Enumerates every trophic chain in the food-web and returns a list object containing some simple statistics. If your analysis requires only simple statistics about trophic chains then this function is more suitable than TrophicChains because it is faster and requires less memory.

Value

A list containing:

chain.lengths The number of nodes in each trophic chain.
node.pos.counts A matrix of NumberOfNodes rows and 1+max(chain.lengths) columns. Elements are the number of chains in which a node appear in that position.
Trophic Levels

Author(s)
Lawrence Hudson

See Also
TrophicChains, NumberOfNodes, IsolatedNodes, BasalNodes, IntermediateNodes, TopLevelNodes

Examples
data(TL84)
chain.stats <- TrophicChainsStats(TL84)

# The length of every chain
length(chain.stats$chain.lengths)  # 5,988 chains
summary(chain.stats$chain.lengths)

# The number of chains in which a node appears in that position in a chain
chain.stats$node.pos.counts

# Basal nodes only have counts in the first column.
chain.stats$node.pos.counts[BasalNodes(TL84),]

# Consumers only have counts in columns two and up.
chain.stats$node.pos.counts[c(IntermediateNodes(TL84), TopLevelNodes(TL84)),]

# All counts are zero for isolated nodes IsolatedNodes.
chain.stats$node.pos.counts[IsolatedNodes(TL84),]

TrophicLevels

Trophic levels

Description
Functions that compute different measures of trophic level.

Usage
PreyAveragedTrophicLevel(community, include.isolated=TRUE)
FlowBasedTrophicLevel(community, weight.by, include.isolated=TRUE)
ShortestTrophicLevel(community, include.isolated=TRUE)
ShortWeightedTrophicLevel(community, include.isolated=TRUE)
LongestTrophicLevel(community, include.isolated=TRUE)
LongWeightedTrophicLevel(community, include.isolated=TRUE)
ChainAveragedTrophicLevel(community, include.isolated=TRUE)
TrophicHeight(community, include.isolated=TRUE)

TrophicLevels(community, weight.by=NULL, include.isolated=TRUE)
Trophic Levels

Arguments

- `community` an object of class `Community`.
- `include.isolated` if `TRUE` then nodes for which `IsIsolatedNode` is `TRUE` are given a trophic level of 1; if `FALSE` then isolated nodes are given a trophic level of `NA`.
- `weight.by` the name of a node property, either first-class or computed, by which to weight flow-based trophic level. Must satisfy the criteria of the properties parameters of `NPS`.

Details

Trophic level is a measure of a node’s ‘distance’ from the primary producers in the community and hence indicates how many steps matter, and hence energy, has been through to reach that node. Each function (with the exception of `TrophicLevels`) returns a vector containing a different measure of trophic level. These functions follow the definitions of Williams and Martinez (2004).

`PreyAveragedTrophicLevel` returns 1 plus the mean trophic level of the node’s resources, using the matrix inversion method of Levine (1980) that is very fast and accounts for flow through loops. If this matrix inversion fails then there is an important problem with the network topology and the function will return a vector containing all `NA`s. For a food web to be energetically feasible, every node must be connected to a basal node. When the inversion fails it is because there is at least one node that has no connection to a basal node. `FlowBasedTrophicLevel` also implements the matrix inversion technique and uses the `weight.by` node property to provide an estimate of energy flow through each trophic link.

`ShortestTrophicLevel`, `ShortWeightedTrophicLevel`, `LongestTrophicLevel`, `LongWeightedTrophicLevel` and `ChainAveragedTrophicLevel` compute trophic level by examining the position of each node in every food chain in which it appears. `ShortestTrophicLevel` returns 1 plus the shortest chain length from a node to a basal species. `ShortWeightedTrophicLevel` returns the average of `ShortestTrophicLevel` and `PreyAveragedTrophicLevel`. `LongestTrophicLevel` is the longest chain length from each node to a basal species. `LongWeightedTrophicLevel` is the average of `LongestTrophicLevel` and `PreyAveragedTrophicLevel`. `ChainAveragedTrophicLevel` is 1 plus the average chain length of all paths from each node to a basal species. These five functions each enumerate every unique food chain (using `TrophicChainsStats`), which can be lengthy for complex food webs. If more than one of these five measures of trophic level is required, it will be faster to use the `TrophicLevels` convenience function, which enumerates unique food chains only once and returns a matrix containing every measure of trophic level in columns ‘ShortestTL’, ‘ShortWeightedTL’, ‘LongestTL’, ‘LongWeightedTL’, ‘ChainAveragedTL’, ‘PreyAveragedTL’ and, if `weight.by` is given, ‘FlowBasedTL’.

Jonsson et al (2005) defined ‘trophic height’ to be the same as Williams and Martinez (2004) chain-averaged trophic level so `TrophicHeight` is a synonym for `ChainAveragedTrophicLevel`.

All methods will return a vector containing all `NA`s if no nodes are basal i.e., all nodes are cannibalistic or ‘community’ has no trophic links.

Value

Either a vector of length `NumberOfNodes` or a matrix with `NumberOfNodes` rows.
Author(s)
Lawrence Hudson and Rich Williams

References

See Also
IsIsolatedNode, IsBasalNode, IsCannibal, NPS, TrophicChains, NumberOfNodes, NumberOfTrophicLinks, TrophicChainsStats, PredationMatrix

Examples

data(TL84)

# Six different measures of trophic level
TrophicLevels(TL84)

# The Benguela data contains diet.fraction
data(Benguela)

# Compare prey-averaged and flow-based
cbind(pa=PreyAveragedTrophicLevel(Benguela),
   fb=FlowBasedTrophicLevel(Benguela, weight.by='diet.fraction'))

TrophicLinkPropertyNames

Trophic link property names

Description
Returns the names of the first-class trophic link properties in a community.

Usage
TrophicLinkPropertyNames(community)

Arguments
community an object of class Community.
Details

The names 'resource' and 'consumer' are always returned.

Value

Two or more characters.

Author(s)

Lawrence Hudson

See Also

Community, TLP, TLPS

Examples

data(TL84, SkipwithPond)

# Just 'resource' and 'consumer'
TrophicLinkPropertyNames(TL84)

# Just 'resource', 'consumer', 'link.evidence' and 'link.life.stage'
TrophicLinkPropertyNames(SkipwithPond)

| TrophicSimilarity | Trophic similarity |

Description

A measure of trophic overlap between nodes in a community.

Usage

TrophicSimilarity(community)
MeanMaximumTrophicSimilarity(community)

Arguments

community     an object of class Community.

Details

TrophicSimilarity computes 'trophic similarity' (I) as defined by Martinez (1991). For each pair of nodes, I = c/(a+b+c), where a is the number of resources and consumers unique to one node, b is number of resources and consumers unique to the other node and c is the number of resources and consumers common to both nodes. Where two nodes have exactly the same set of resources and consumers, I = 1. Where two nodes have no resources or consumers in common, I = 0.

Williams and Martinez (2000) defined the mean maximum trophic similarity as the sum of the largest value in each column of I (excluding the diagonal), divided by the number of nodes.
Value

TrophicSimilarity returns a matrix with NumberOfNodes rows and columns. MeanMaximumSimilarity returns a number.

Author(s)

Lawrence Hudson

References


See Also

PredationMatrix, NumberOfNodes, TrophicSpecies

Examples

data(TL84)
I <- TrophicSimilarity(TL84)
I
MeanMaximumTrophicSimilarity(TL84)

TrophicSpecies  Trophic species

Description

A function that computes trophic species numbers.

Usage

TrophicSpecies(community, include.isolated=TRUE)

Arguments

community  an object of class Community.
include.isolated  if TRUE then nodes for which IsIsolatedNode is TRUE are given their own trophic species number. If FALSE the isolated species are assigned a trophic species of NA.
TrophicSpecies

Details

Returns a vector containing the trophic species number of each node in the community. Nodes with identical sets of prey and predators are given the same trophic species number.

Value

A vector of length NumberOfNodes.

Author(s)

Lawrence Hudson

References


See Also

Community, IsIsolatedNode, NPS, LumpTrophicSpecies, NumberOfNodes

Examples

data(TL84)

# Isolated nodes assigned their own trophic species number
TrophicSpecies(TL84)

# Isolated nodes assigned a trophic species of NA
TrophicSpecies(TL84, include.isolated=FALSE)

# Compare including and excluding isolated nodes
NPS(TL84, list(TS1='TrophicSpecies',
    TS2=list('TrophicSpecies', include.isolated=FALSE),
    Iso='IsIsolatedNode'))
Description

The community of Ythan Estuary.


Usage

YthanEstuary

Format

Community.

Source


References


Index

* Topic datasets
  - Benguela, 6
  - BroadstoneStream, 9
  - ChesapeakeBay, 11
  - Millstream, 36
  - pHWebs, 54
  - SkipwithPond, 88
  - TL84, 92
  - YthanEstuary, 104

* Topic hplot
  - PlotAupperValower, 54
  - PlotCircularWeb, 55
  - PlotHelpers, 56
  - PlotNPS, 58
  - PlotNPSDistribution, 62
  - PlotRankNPS, 63
  - PlotWagonWheel, 69
  - PlotWebByLevel, 70
  - Pyramid plots, 75
  - Spectrum plots, 88

* Topic package
  - cheddar, 10

* Topic utilities
  - AggregateCommunities, 3
  - ApplyByClass, 5
  - Body mass, numerical abundance and biomass abundance, 7
  - BodyMassBins, 8
  - CollectionApply, 11
  - CollectionCPS, 12
  - CollectionNPS, 14
  - CollectionTLPS, 15
  - Community, 16
  - Community has property?, 18
  - CommunityCollection, 19
  - CommunityPropertyNames, 21
  - CP, 22
  - CPS, 23
  - Degree, 24
  - DegreeDistribution, 25
  - Intervality, 26
  - IsCannibal, 28
  - LinearRegressionByClass, 29
  - LoadCollection, 30
  - LoadCommunity, 31
  - LumpNodes, 32
  - LumpTrophicSpecies, 34
  - Node connectivity, 36
  - NodeNameIndices, 38
  - NodePropertyNames, 39
  - NP, 40
  - NPS, 41
  - NumberOfNodes, 42
  - NumberOfTrophicLinks, 43
  - NvMConvexHull, 44
  - NvMLinearRegressions, 45
  - NvMTriTrophicStatistics, 47
  - NvMTriTrophicTable, 48
  - Omnivory, 50
  - OrderCollection, 51
  - OrderCommunity, 52
  - PredationMatrix, 72
  - PredationMatrixToLinks, 74
  - QuantitativeDescriptors, 78
  - RemoveCannibalisticLinks, 80
  - RemoveIsolatedNodes, 81
  - RemoveNodes, 82
  - ResourceLargerThanConsumer, 83
  - ResourcesByNode, 84
  - ShortestPaths, 85
  - SiteBySpeciesMatrix, 86
  - subset.CommunityCollection, 90
  - ThreeNodeChains, 91
  - TLP, 93
  - TLPS, 94
  - TrophicChains, 95
  - TrophicChainsStats, 97
TrophicLevels, 98
TrophicLinkPropertyNames, 100
TrophicSimilarity, 101
TrophicSpecies, 102
[.CommunityCollection
  (CommunityCollection), 19
[->[.Community (Community), 16
[->[.CommunityCollection
  (CommunityCollection), 19
[[->[.Community (Community), 16
[[->[.CommunityCollection
  (CommunityCollection), 19
$<-[.Community (Community), 16
$<-[.CommunityCollection
  (CommunityCollection), 19
AggregateCommunities, 3, 20
AggregateCommunitiesBy, 20
AggregateCommunitiesBy
  (AggregateCommunities), 3
ApplyByClass, 5, 29, 46
BasalNodes, 82, 96, 98
BasalNodes (Node connectivity), 36
Benguela, 6
Biomass, 87
Biomass (Body mass, numerical abundance and biomass abundance), 7
Body mass, numerical abundance and biomass abundance, 7
BodyMassBins, 8, 90
BroadstoneStream, 9
Cannibals, 38
Cannibals (IsCannibal), 28
ceiling, 77
ChainAveragedTrophicLevel, 51
ChainAveragedTrophicLevel
  (TrophicLevels), 98
CharacteristicPathLength
  (ShortestPaths), 85
cheddar, 10
cheddar-package (cheddar), 10
ChesapeakeBay, 11
chull, 45
CollectionApply, 11
CollectionCPS, 4, 12, 20, 52, 87, 91
CollectionNPS, 14, 20, 53
CollectionTLPS, 15, 20
Community, 5, 9, 16, 18, 20–23, 25, 27, 29, 32, 39, 43, 45, 46, 53, 56, 58, 61, 63, 65, 68, 70, 72, 74, 77, 80–82, 84, 90, 101, 103
Community has property?, 18
CommunityCollection, 4, 12–15, 19, 30, 46, 49, 52, 87, 91
CommunityPropertyNames, 13, 21, 22, 23
ConnectedNodes (Node connectivity), 36
ConsumersByNode, 28, 38, 73
ConsumersByNode (ResourcesByNode), 84
ConsumersOfNodes, 28, 38
ConsumersOfNodes (ResourcesByNode), 84
CP, 21, 22, 23
CPS, 13, 17, 18, 21, 22, 23
CRMRatio (Body mass, numerical abundance and biomass abundance), 7
DefaultCategoryColours, 58, 61
DefaultCategoryColours (PlotHelpers), 56
DefaultCategoryLabelColours
  (PlotHelpers), 56
DefaultCategorySymbols, 61
DefaultCategorySymbols (PlotHelpers), 56
DefaultLinkColour (PlotHelpers), 56
Degree, 24, 26, 28, 38, 70
DegreeDistribution, 25, 25, 63
dim<-[.Community (Community), 16
dim<-[.CommunityCollection
  (CommunityCollection), 19
DirectedConnectance, 25
DirectedConnectance
  (NumberOfTrophicLinks), 43
floor, 77
FlowBasedTrophicLevel (TrophicLevels), 98
FormatLM (PlotHelpers), 56
FractionBasalNodes (Node connectivity), 36
FractionCanibalistic (IsCannibal), 28
FractionConnectedNodes (Node connectivity), 36
FractionIntermediateNodes (Node connectivity), 36
FractionIsolatedNodes (Node connectivity), 36
FractionNonBasalNodes (Node connectivity), 36
FractionNonConnectedNodes (Node connectivity), 36
FractionNonTopLevelNodes (Node connectivity), 36
FractionOfNodesByClass (NumberOfNodes), 42
FractionOmnivorous (Omnivory), 50
FractionTopLevelNodes (Node connectivity), 36
HasM (Community has property?), 18
HasN (Community has property?), 18
HasTrophicLinks (Community has property?), 18
InDegree, 28, 38
InDegree (Degree), 24
IntermediateNodes, 98
IntermediateNodes (Node connectivity), 36
Intervality, 26, 53
is.Community (Community), 16
is.CommunityCollection (CommunityCollection), 19
IsBasalNode, 100
IsBasalNode (Node connectivity), 36
IsCannibal, 28, 38, 100
IsConnectedNode (Node connectivity), 36
IsIntermediateNode (Node connectivity), 36
IsIsolatedNode, 33, 35, 81, 100, 103
IsIsolatedNode (Node connectivity), 36
IsNonBasalNode (Node connectivity), 36
IsNonTopLevelNode (Node connectivity), 36
IsolatedNodes, 33, 70, 82, 98
IsolatedNodes (Node connectivity), 36
IsOmnivore (Omnivory), 50
IsTopLevelNode (Node connectivity), 36
lapply, 12
length<- .Community (Community), 16
length<- .CommunityCollection (CommunityCollection), 19
levels<- .Community (Community), 16
levels<- .CommunityCollection (CommunityCollection), 19
LinearRegressionByClass, 29, 58
LinkageDensity, 25
LinkageDensity (NumberOfTrophicLinks), 43
lm, 29, 58
LMabline (PlotHelpers), 56
LoadCollection, 30
LoadCommunity, 17, 30, 31
Log10Biomass, 87
Log10Biomass (Body mass, numerical abundance and biomass abundance), 7
Log10BLabel, 77
Log10BLabel (PlotHelpers), 56
Log10CRMRatio (Body mass, numerical abundance and biomass abundance), 7
Log10M (Body mass, numerical abundance and biomass abundance), 7
Log10MLabel (PlotHelpers), 56
Log10MBiomass (Body mass, numerical abundance and biomass abundance), 7
Log10N (Body mass, numerical abundance and biomass abundance), 7
Log10NLabel, 77
Log10NLabel (PlotHelpers), 56
Log10RCMRatio (Body mass, numerical abundance and biomass abundance), 7
LongestTrophicLevel (TrophicLevels), 98
LongWeightedTrophicLevel (TrophicLevels), 98
LumpNodes, 32, 35
LumpTrophicSpecies, 33, 34, 103
matrix, 87
MeanMaximumTrophicSimilarity (TrophicSimilarity), 101
Millstream, 36
MinimiseSumConsumerGaps (Intervality), 26
MinimiseSumDietGaps (Intervality), 26
names<- .Community (Community), 16
names<- .CommunityCollection (CommunityCollection), 19
Node connectivity, 36
NodeNameIndices, 38
NodePropertyNames, 39
NodeQuantitativeDescriptors (QuantitativeDescriptors), 78
NonBasalNodes (Node connectivity), 36
NonTopLevelNodes (Node connectivity), 36
NormalisedTrophicGenerality (Degree), 24
NormalisedTrophicVulnerability (Degree), 24
NP, 40, 40, 42
NPS, 5, 8, 14, 17, 18, 29, 33, 40, 41, 43, 61, 63, 65, 86, 87, 95, 100, 103
NumberOfConsumers (Degree), 24
NumberOfNodes (NumberOfNodes), 42
NumberOfNodesByClass (NumberOfNodes), 42
NumberOfResources (Degree), 24
NumberOfTrophicLinks, 8, 38, 43, 73, 79, 94, 95, 100
NvMConvexHull, 44
NvMIntercept (NvMLinearRegression), 45
NvMInterceptByClass (NvMLinearRegression), 45
NvMLinearRegressions, 29, 45, 58
NvMSlope (NvMLinearRegression), 45
NvMSlopeAndIntercept (NvMLinearRegression), 45
NvMSlopeAndInterceptByClass (NvMLinearRegression), 45
NvMSlopeByClass (NvMLinearRegression), 45
NvMTriTrophicStatistics, 47, 49, 55
NvMTriTrophicTable, 47, 48
Omnivores (Omnivory), 50
Omnivory, 50
order, 52, 53
OrderCollection, 20, 30, 51
OrderCommunity, 27, 52, 70
OutDegree, 28, 38
OutDegree (Degree), 24
pHWebs, 19, 20, 54
PlaceMissingPoints, 61
PlaceMissingPoints (PlotHelpers), 56
plotCommunity (Community), 16
plotCommunityCollection (CommunityCollection), 19

PlotAupperVlower, 47, 54
PlotBCvBR (PlotTLS), 65
PlotBDistribution (PlotNPSDistribution), 62
PlotBPYramid (Pyramid plots), 75
PlotBRvBC (PlotTLS), 65
PlotBSpectrum, 56, 61, 65, 68, 72
PlotBSpectrum (Spectrum plots), 88
PlotBvM (PlotNPS), 58
PlotBvRankB (PlotRankNPS), 63
PlotCircularWeb, 55, 61, 63, 65, 68, 70, 72, 90
PlotDegreeDistribution, 26
PlotDegreeDistribution (PlotNPSDistribution), 62
PlotHelpers, 56
PlotLinearModels (PlotHelpers), 56
PlotMCvNR (PlotTLS), 65
PlotMDistribution (PlotNPSDistribution), 62
PlotMRvMC (PlotTLS), 65
PlotMvB (PlotNPS), 58
PlotMvN (PlotNPS), 58
PlotMvRankM (PlotRankNPS), 63
PlotNcVNR (PlotTLS), 65
PlotNDistribution (PlotNPSDistribution), 62
PlotNPS, 56, 58, 63, 65, 68, 70, 72, 90
PlotNPSdistribution, 56, 61, 62, 63, 65, 68, 72, 90
PlotNPYramid (Pyramid plots), 75
PlotNRvMC (PlotTLS), 65
PlotNRvNC (PlotTLS), 65
PlotNPSpectrum, 56, 61, 65, 68, 72
PlotNPSpectrum (Spectrum plots), 88
PlotNvM, 45
PlotNvM (PlotNPS), 58
PlotNvRankN (PlotRankNPS), 63
PlotPredationMatrix, 27, 53, 73
PlotPredationMatrix (PlotTLS), 65
PlotRankNPS, 56, 61, 63, 65, 68, 72, 90
PlotTLS, 56, 61, 63, 65, 68, 72, 90
PlotWagonWheel, 56, 69
PlotWebByLevel, 56, 61, 63, 65, 68, 70, 70, 90
PredationMatrix, 27, 28, 38, 72, 74, 85, 86, 100, 102
PredationMatrixToLinks, 73, 74
PreyAveragedTrophicLevel, 51, 53
PreyAveragedTrophicLevel
(TrophicLevels), 98
print.Community (Community), 16
print.CommunityCollection (CommunityCollection), 19
Pyramid plots, 75
QuantitativeDescriptors, 78
RCMRatio (Body mass, numerical abundance and biomass abundance), 7
read.csv, 32
RemoveCannibalisticLinks, 28, 80
RemoveIsolatedNodes, 81
RemoveNodes, 82
ResourceLargerThanConsumer, 83
ResourcesAndConsumersByNode (ResourcesByNode), 84
ResourcesByNode, 28, 38, 73, 84
ResourcesOfNodes, 28, 38
ResourcesOfNodes (ResourcesByNode), 84
rgb, 70
SaveCollection (LoadCollection), 30
SaveCommunity, 17, 30
SaveCommunity (LoadCommunity), 31
ShortestPaths, 70, 85
ShortestTrophicLevel (TrophicLevels), 98
ShortWeightedTrophicLevel (TrophicLevels), 98
SiteBySpeciesMatrix, 86
SkipwithPond, 88, 96
Spectrum plots, 88
subset, 91
subset.CommunityCollection, 20, 90
SumBiomassByClass, 77
SumBiomassByClass (ApplyByClass), 5
SumConsumerGaps (Intervality), 26
SumDietGaps (Intervality), 26
summary.Community (Community), 16
summary.CommunityCollection (CommunityCollection), 19
SumMB byClass (ApplyByClass), 5
SumNByClass, 77
SumNByClass (ApplyByClass), 5
ThreeNodeChains, 47, 91, 95, 96
TL84, 92
TL86 (TL84), 92
TLPS, 8, 15, 17, 18, 47, 68, 73, 74, 85, 92, 94, 96, 101
TopLevelNodes, 96, 98
TopLevelNodes (Node connectivity), 36
TrophicChains, 47, 92, 95, 98, 100
TrophicChainsStats, 96, 97, 100
TrophicGenerality (Degree), 24
TrophicHeight (TrophicLevels), 98
TrophicLevels, 98
TrophicLinkPropertyNames, 79, 94, 95, 100
TrophicLinksForNodes (ResourcesByNode), 84
TrophicSimilarity, 101
TrophicSpecies, 35, 102, 102
TrophicVulnerability (Degree), 24
weighted.mean, 33
write.csv, 32
YthanEstuary, 104