Working with communities (0.1-638)

Lawrence Hudson

2022-09-01

Contents

1 Introduction

The core of the package is a flexible, extendable representation of an ecological community, described in this vignette. Cheddar's system for plotting communities and statistical analysis of communities is covered in the 'PlotsAndStats' vignette. The 'ImportExport' vignette covers getting community data in to and out of Cheddar. If you are working with collections, for example to see how community structure changes through time, across environmental gradients or resulting from experimental manipulation, read the 'Collections' vignette.

2 Datasets

Cheddar contains several published empirical food-web datasets (Table 1).

Community	Notes	References
`Benguela`	Crude estimate of M; trophic links have diet fraction	Yodzis (1998)
`BroadstoneStream`	M and N; nodes are well resolved	Woodward et al. (2005)
`TL84` and `TL86`	M and N; nodes are well resolved	Carpenter and Kitchell (1996) Cohen et al. (2003) Jonsson et al. (2005)
`SkipwithPond`	No M or N; trophic links have 'link.evidence' and 'link.life.stage' properties	Warren (1989)
`YthanEstuary`	M and N for all nodes except detritus; nodes are well-resolved at high trophic levels but poorly resolved at low trophic levels	Hall and Raffaelli (1991) Emmerson and Raffaelli (2004)

Table 1: Community data in Cheddar. M: body mass. N: numerical abundance.

3 Community representation

A Cheddar community has three aspects:

- *community properties* such as sampling date, lat & long, altitude, temperature and pH,
- *nodes*, which are the names of species together with any associated properties such as mean body mass, M, and mean numerical abundance, N, and taxonomic classification,
- the *food web*, defined as the names of each resource-consumer node pair, together with properties such as evidence for the link (e.g. empirically observed or inferred from literature).

The final aspect is optional - Cheddar communities need not contain trophic links. The `LoadCommunity` and `SaveCommunity` functions provide a standard data format, with each aspect stored in its own CSV (Comma-Separated Value) file, described further in the 'ImportExport' vignette. Cheddar allows user-defined data to be added to any of these three aspects simply by adding the data to the relevant CSV file. Any data so added will be available to Cheddar's plotting and analysis functions. Each aspect is accessed using the functions `CPS` (for **C**ommunity **P**ropertie**S**), `NPS` (for **N**ode **P**ropertie**S**) and `TLPS` (for **T**rophic **L**ink **P**ropertie**S**) (Table 2). Each of the three community aspects are discussed below. The following examples use the `TL84` dataset, which is from Tuesday Lake in Michigan, USA, sampled in 1984 (Carpenter and Kitchell, 1996; Cohen et al., 2003; Jonsson et al., 2005). The data contain estimates of body mass, M, and numerical abundance, N, for each species.

```
> data(TL84)  # Load the dataset
> print(TL84) # A description of the data

Tuesday Lake sampled in 1984 containing 56 nodes and 269 trophic links
```

Aspect	Accessor function	PlotFunction	CSV file
Properties	`CPS`	n/a	`properties.csv`
Nodes	`NPS`	`PlotNPS`	`nodes.csv`
Food web	`TLPS`	`PlotTLPS`	`trophic.links.csv`

Table 2: Aspects of a Cheddar community

3.1 Community properties

The `CommunityPropertyNames` function returns the names of the community properties.

```
> CommunityPropertyNames(TL84)

[1] "title"   "M.units" "N.units" "lat"     "long"    "habitat"
```

'title' is the only property that a community is guaranteed to contain. The function `CPS` (for **C**ommunity **P**ropertie**S**) returns a `list` of values.

```
> CPS(TL84)

$title
[1] "Tuesday Lake sampled in 1984"

$M.units
[1] "kg"

$N.units
[1] "m^-3"

$lat
[1] 46.21667

$long
[1] 89.53333

$habitat
[1] "Freshwater pelagic"
```

This shows the latitude and longitude of the lake and tells us the units for body mass, M, and numerical abundance, N, are kg and individuals per metre cubed, respectively. Many of the provided communities (Table 1) contain lat, long and habitat. Some communities have more properties. `CPS` lets you get a subset of community properties. For example, to see only the lat and long.

```
> CPS(TL84, c('lat', 'long'))

$lat
[1] 46.21667

$long
[1] 89.53333
```

CPS also accepts the names of functions that compute community properties. Two such functions are NumberOfNodes and NumberOfTrophicLinks.

```
> NumberOfNodes(TL84)

[1] 56

> NumberOfTrophicLinks(TL84)

[1] 269

> # A list containing lat, long, number of nodes and number of trophic links
> CPS(TL84, c('lat', 'long', 'NumberOfNodes', 'NumberOfTrophicLinks'))

$lat
[1] 46.21667

$long
[1] 89.53333

$NumberOfNodes
[1] 56

$NumberOfTrophicLinks
[1] 269
```

A named vector can be used to rename values.

```
> CPS(TL84, c('lat', 'long', S='NumberOfNodes', L='NumberOfTrophicLinks'))

$lat
[1] 46.21667

$long
[1] 89.53333

$S
[1] 56

$L
[1] 269
```

Names that are neither properties of the community nor function names result in NA.

```
> # Returns a list containing 'not a property'=NA
> CPS(TL84, c('not a property'))

$`not a property`
[1] NA
```

The related function CollectionCPS will be of interest if you are examining collections of communities, described in the 'Collections' vignette.

3.2 Nodes

Let's use two more Cheddar functions to get some more information about TL84's nodes.

```
> NumberOfNodes(TL84)
```

```
[1] 56
```

```
> NodePropertyNames(TL84)
```

```
 [1] "node"       "category"   "M"          "N"          "kingdom"    "phylum"
 [7] "class"      "order"      "family"     "genus"      "species"    "resolved.to"
```

The data contains 56 nodes and NodePropertyNames tells us that TL84 contains a lot of information about each node. We can get a table of these node properties by using the NPS function. To avoid printing the full table of 56 rows, the examples below use R 's head and tail functions to show just the first or last six rows of the data.frame returned by NPS.

```
> head(NPS(TL84))
```

```
                                    node category        M       N   kingdom
Nostoc sp.                     Nostoc sp. producer 7.97e-13 2.0e+06  Bacteria
Arthrodesmus sp.           Arthrodesmus sp. producer 1.52e-12 4.9e+07   Plantae
Asterionella formosa   Asterionella formosa producer 1.12e-12 5.0e+06 Chromista
Cryptomonas sp. 1         Cryptomonas sp. 1 producer 2.03e-13 6.4e+07 Chromista
Cryptomonas sp. 2         Cryptomonas sp. 2 producer 1.51e-12 2.8e+07 Chromista
Chroococcus dispersus Chroococcus dispersus producer 2.39e-13 2.0e+07  Bacteria
                            phylum             class           order           family
Nostoc sp.             Cyanobacteria      Cyanophyceae      Nostocales      Nostocaceae
Arthrodesmus sp.          Charophyta  Zygnematophyceae     Desmidiales     Desmidiaceae
Asterionella formosa      Ochrophyta Bacillariophyceae   Fragilariales   Fragilariaceae
Cryptomonas sp. 1        Cryptophyta     Cryptophyceae Cryptomonadales Cryptomonadaceae
Cryptomonas sp. 2        Cryptophyta     Cryptophyceae Cryptomonadales Cryptomonadaceae
Chroococcus dispersus Cyanobacteria      Cyanophyceae   Chroococcales   Chroococcaceae
                            genus   species resolved.to
Nostoc sp.                 Nostoc               Species
Arthrodesmus sp.      Arthrodesmus               Species
Asterionella formosa  Asterionella   formosa     Species
Cryptomonas sp. 1      Cryptomonas               Species
Cryptomonas sp. 2      Cryptomonas               Species
Chroococcus dispersus  Chroococcus dispersus     Species
```

'node' in the only node property that a community is guaranteed to contain. Many of Cheddar's plotting an analysis functions make use of the 'category' node property; this property is optional but, if included in a community, it should contain one of 'producer', 'invertebrate', 'vert.ecto', 'vert.endo' or should be empty.

```
> # Just body mass
> head(NPS(TL84, 'M'))
```

```
                            M
Nostoc sp.            7.97e-13
Arthrodesmus sp.      1.52e-12
Asterionella formosa  1.12e-12
Cryptomonas sp. 1     2.03e-13
Cryptomonas sp. 2     1.51e-12
Chroococcus dispersus 2.39e-13
```

```
> # Body mass and numerical abundance.
> head(NPS(TL84, c('M','N')))
```

```
                            M       N
Nostoc sp.            7.97e-13 2.0e+06
Arthrodesmus sp.      1.52e-12 4.9e+07
Asterionella formosa  1.12e-12 5.0e+06
Cryptomonas sp. 1     2.03e-13 6.4e+07
Cryptomonas sp. 2     1.51e-12 2.8e+07
Chroococcus dispersus 2.39e-13 2.0e+07
```

In addition to first-class node properties like M and N, you can also use `NPS` to assemble computed node properties by passing in the name(s) of function(s) 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. Cheddar contains many suitable functions and you can also write your own. For example, it is common to $\log_{10}$-transformation M and N, which we can do using the `Log10M` and `Log10N` functions.

```
> tail(NPS(TL84, c('Log10M', 'Log10N')))
```

```
                          Log10M     Log10N
Trichocerca cylindrica -9.420216  4.9116902
Tropocyclops prasinus  -8.164309  4.6919651
Chaoborus punctipennis -6.522879  4.0791812
Phoxinus eos           -2.995679  0.2944662
Phoxinus neogaeus      -2.931814 -0.8761484
Umbra limi             -2.889410 -0.8794261
```

You can provide a mix of property and function names.

```
> tail(NPS(TL84, c('Log10M', 'Log10N', 'category', 'phylum')))
```

```
                          Log10M     Log10N     category     phylum
Trichocerca cylindrica -9.420216  4.9116902 invertebrate   Rotifera
Tropocyclops prasinus  -8.164309  4.6919651 invertebrate Arthropoda
Chaoborus punctipennis -6.522879  4.0791812 invertebrate Arthropoda
Phoxinus eos           -2.995679  0.2944662    vert.ecto   Chordata
Phoxinus neogaeus      -2.931814 -0.8761484    vert.ecto   Chordata
Umbra limi             -2.889410 -0.8794261    vert.ecto   Chordata
```

The `Log10MNBiomass` function returns a `matrix` of $\log_{10}$-transformed body mass, M, numerical abundance, N, and biomass, B, and is a convenient way to get all three of these properties in to a table.

```
> tail(NPS(TL84, c('Log10MNBiomass')))
```

```
                          Log10M     Log10N Log10Biomass
Trichocerca cylindrica -9.420216  4.9116902    -4.508526
Tropocyclops prasinus  -8.164309  4.6919651    -3.472344
Chaoborus punctipennis -6.522879  4.0791812    -2.443697
Phoxinus eos           -2.995679  0.2944662    -2.701212
Phoxinus neogaeus      -2.931814 -0.8761484    -3.807962
Umbra limi             -2.889410 -0.8794261    -3.768836
```

We can use `NPS` to assemble a table showing node degree: the number of trophic links in to and out of that node. Cheddar contains three functions that compute a different aspect of node degree.

```
> nps <- NPS(TL84, c('InDegree','OutDegree','Degree'))
> head(nps)

                     InDegree OutDegree Degree
Nostoc sp.                  0         4      4
Arthrodesmus sp.            0         3      3
Asterionella formosa        0         0      0
Cryptomonas sp. 1           0        18     18
Cryptomonas sp. 2           0         5      5
Chroococcus dispersus       0        18     18

> # This is always true for all nodes
> all(nps$Degree == nps$InDegree+nps$OutDegree)

[1] TRUE
```

Some readers will be more familiar with the terms 'trophic vulnerability' and 'trophic generality'; the functions `TrophicVulnerability` and `TrophicGenerality` are synonyms for `OutDegree` and `InDegree` respectively. Cannibalistic links count twice towards `Degree` - one link going out and one going in. The cannibalistic fish *Umbra limi* in `TL84` has no consumers other than itself so it has an `OutDegree` of one.

```
> IsCannibal(TL84)['Umbra limi']

Umbra limi
      TRUE

> InDegree(TL84)["Umbra limi"]

Umbra limi
        12

> OutDegree(TL84)["Umbra limi"]

Umbra limi
         1

> Degree(TL84)["Umbra limi"]

Umbra limi
        13
```

We can combine some of these functions to investigate allometric degree distribution (Jonsson et al., 2005; Otto et al., 2007; Digel et al., 2011; Jacob et al., 2011), which describe how species' numbers of trophic links scale with their log-transformed body masses.

```
> tail(NPS(TL84, c('Log10M', 'OutDegree', 'InDegree', 'Degree')))
```

```
                         Log10M OutDegree InDegree Degree
Trichocerca cylindrica -9.420216         4        6     10
Tropocyclops prasinus  -8.164309         7       16     23
Chaoborus punctipennis -6.522879         4       22     26
Phoxinus eos           -2.995679         1        9     10
Phoxinus neogaeus      -2.931814         1        9     10
Umbra limi             -2.889410         1       12     13
```

Some authors have been interested in how trophic level varies with body mass (Jacob et al., 2011). Two more functions suitable for use with `NPS` are `PreyAveragedTrophicLevel` and `ChainAveragedTrophicLevel`, which give different measures of each node's trophic level in the food web; these two functions, and others related to trophic level, are discussed further in section 3.3.

```
> tail(NPS(TL84, c('Log10M', 'PreyAveragedTrophicLevel',
                   'ChainAveragedTrophicLevel')))
```

```
                         Log10M PreyAveragedTrophicLevel ChainAveragedTrophicLevel
Trichocerca cylindrica -9.420216                2.000000                  2.000000
Tropocyclops prasinus  -8.164309                3.142857                  3.333333
Chaoborus punctipennis -6.522879                3.171344                  4.602527
Phoxinus eos           -2.995679                3.529951                  5.168337
Phoxinus neogaeus      -2.931814                3.529951                  5.168337
Umbra limi             -2.889410                3.802678                  5.835003
```

The column titles for the trophic-level measures are very long. We can use a named vector to get shortened column titles.

```
> tail(NPS(TL84, c('Log10M', PATL='PreyAveragedTrophicLevel',
                   CATL='ChainAveragedTrophicLevel')))
```

```
                         Log10M     PATL     CATL
Trichocerca cylindrica -9.420216 2.000000 2.000000
Tropocyclops prasinus  -8.164309 3.142857 3.333333
Chaoborus punctipennis -6.522879 3.171344 4.602527
Phoxinus eos           -2.995679 3.529951 5.168337
Phoxinus neogaeus      -2.931814 3.529951 5.168337
Umbra limi             -2.889410 3.802678 5.835003
```

`NPS` also allows parameters to be passed to functions. This is demonstrated using the `TrophicSpecies` function: in order to account for different levels of taxonomic resolution and other biases, researchers often lump biological species together. Species in the food web that have the same resources and consumers are the same 'trophic species' (Briand and Cohen, 1984; Pimm et al., 1991; Williams and Martinez, 2000). The `TrophicSpecies` function computes these IDs.

```
> tail(TrophicSpecies(TL84))
```

```
Trichocerca cylindrica  Tropocyclops prasinus Chaoborus punctipennis
                    16                     13                    20
          Phoxinus eos      Phoxinus neogaeus             Umbra limi
                    21                     21                    22
```

Some analyses (e.g. Jonsson et al., 2005) exclude isolated species when computing trophic species numbers. Isolated species are those nodes that consume no others and have no consumers (Section 3.3.4). To compare the effect of including or excluding isolated species we can pass the function to `NPS` twice, once setting the 'include.isolated' parameter.

```
> head(NPS(TL84, list(TS.iso='TrophicSpecies',
                      TS.no.iso=list('TrophicSpecies', include.isolated=FALSE))))

                      TS.iso TS.no.iso
Nostoc sp.                 1         1
Arthrodesmus sp.           2         2
Asterionella formosa       3        NA
Cryptomonas sp. 1          4         3
Cryptomonas sp. 2          5         4
Chroococcus dispersus      4         3
```

Asterionella formosa is an isolated species so has been given a trophic species of `NA` in the 'TS.no.iso' column. The `LumpTrophicSpecies` function lumps nodes together using these IDs (Section 4.5). The second argument to `NPS` can therefore be defined as a list containing 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 the list are column names in the returned `data.frame`.

`NPS` therefore makes it very easy to assemble tables of properties either for analysis or for presentation in a manuscript. The example below recreates the first ten rows of Jonsson et al. (2005), Appendix 1A (p74–75).

```
> head(NPS(TL84, list('category', BM='M', 'NA'='N',
                      TS=list('TrophicSpecies', include.isolated=FALSE),
                      TH=list('TrophicHeight', include.isolated=FALSE))),
      10)

                            category       BM      NA TS TH
Nostoc sp.                  producer 7.97e-13 2.0e+06  1  1
Arthrodesmus sp.            producer 1.52e-12 4.9e+07  2  1
Asterionella formosa        producer 1.12e-12 5.0e+06 NA NA
Cryptomonas sp. 1           producer 2.03e-13 6.4e+07  3  1
Cryptomonas sp. 2           producer 1.51e-12 2.8e+07  4  1
Chroococcus dispersus       producer 2.39e-13 2.0e+07  3  1
Closteriopsis longissimus   producer 2.37e-13 1.0e+08  5  1
Chrysosphaerella longispina producer 8.31e-10 4.0e+06 NA NA
Dinobryon bavaricum         producer 2.44e-12 3.0e+07  6  1
Dinobryon cylindricum       producer 1.57e-12 3.0e+06  1  1
```

Some values in this table are different to those presented by Jonsson et al. (2005) in their Appendix 1A. Firstly, the numerical abundance values for zooplankton are different. Values in their table "...should

be multiplied by 6 to convert them to concentrations in the epilimnion" (Jonsson et al., 2005), and our data incorporate that conversion. Secondly, the values of trophic height presented are slightly different for species at higher trophic levels because of the different methods used to break cycles (see the help for Cheddar's `TrophicSpecies` function). It is not clear from the text of Jonsson et al. (2005) exactly how they broke cycles. Because Cheddar is open source, users can refer readers to the function and version used to completely specify the algorithm used.

`NPS` returns `NA` for any names that are neither a first-class property of the community nor the name of a function.

```
> head(NPS(TL84, c('Not a property or function')))
```

```
                     Not a property or function
Nostoc sp.                                   NA
Arthrodesmus sp.                             NA
Asterionella formosa                         NA
Cryptomonas sp. 1                            NA
Cryptomonas sp. 2                            NA
Chroococcus dispersus                        NA
```

3.3 Food web

`NumberOfTrophicLinks` returns the number of trophic links that the community contains.

```
> NumberOfTrophicLinks(TL84)
```

```
[1] 269
```

Cheddar communities need not contain trophic links so this function might return zero. The following sections describe some different ways in which to view and analyse food webs in cheddar.

3.3.1 Resource-consumer pairs

`TLPS` (for **T**rophic **L**ink **P**ropertie**S**) returns a `data.frame` of trophic links pairs (or `NULL` if the community has no food web). The `data.frame` always contains the columns 'resource' and 'consumer'.

```
> head(TLPS(TL84))
```

```
                  resource           consumer
1        Cryptomonas sp. 1 Ascomorpha eucadis
2    Chroococcus dispersus Ascomorpha eucadis
3 Unclassified flagellates Ascomorpha eucadis
4           Chromulina sp. Ascomorpha eucadis
5      Selenastrum minutum Ascomorpha eucadis
6        Trachelomonas sp. Ascomorpha eucadis
```

`TLPS` takes a parameter `node.properties`, which should be a vector of names suitable for passing to `NPS`. You can therefore use functions and names and can pass parameters to functions, just as in the `NPS` examples above.

```
> head(TLPS(TL84, node.properties='M'))
```

```
                                      resource           consumer resource.M
Cryptomonas sp. 1                Cryptomonas sp. 1 Ascomorpha eucadis   2.03e-13
Chroococcus dispersus       Chroococcus dispersus Ascomorpha eucadis   2.39e-13
Unclassified flagellates Unclassified flagellates Ascomorpha eucadis   3.46e-13
Chromulina sp.                      Chromulina sp. Ascomorpha eucadis   3.03e-14
Selenastrum minutum           Selenastrum minutum Ascomorpha eucadis   2.72e-13
Trachelomonas sp.                Trachelomonas sp. Ascomorpha eucadis   1.75e-13
                         consumer.M
Cryptomonas sp. 1           1.4e-10
Chroococcus dispersus       1.4e-10
Unclassified flagellates    1.4e-10
Chromulina sp.              1.4e-10
Selenastrum minutum         1.4e-10
Trachelomonas sp.           1.4e-10
```

> head(TLPS(TL84, node.properties=c('M','Biomass')))

```
                                      resource           consumer resource.M
Cryptomonas sp. 1                Cryptomonas sp. 1 Ascomorpha eucadis   2.03e-13
Chroococcus dispersus       Chroococcus dispersus Ascomorpha eucadis   2.39e-13
Unclassified flagellates Unclassified flagellates Ascomorpha eucadis   3.46e-13
Chromulina sp.                      Chromulina sp. Ascomorpha eucadis   3.03e-14
Selenastrum minutum           Selenastrum minutum Ascomorpha eucadis   2.72e-13
Trachelomonas sp.                Trachelomonas sp. Ascomorpha eucadis   1.75e-13
                         resource.Biomass consumer.M consumer.Biomass
Cryptomonas sp. 1              1.2992e-05    1.4e-10        1.932e-06
Chroococcus dispersus          4.7800e-06    1.4e-10        1.932e-06
Unclassified flagellates       6.5048e-04    1.4e-10        1.932e-06
Chromulina sp.                 4.5147e-06    1.4e-10        1.932e-06
Selenastrum minutum            5.4400e-05    1.4e-10        1.932e-06
Trachelomonas sp.              3.8850e-05    1.4e-10        1.932e-06
```

> head(TLPS(TL84, node.properties=c('M', B='Biomass')))

```
                                      resource           consumer resource.M
Cryptomonas sp. 1                Cryptomonas sp. 1 Ascomorpha eucadis   2.03e-13
Chroococcus dispersus       Chroococcus dispersus Ascomorpha eucadis   2.39e-13
Unclassified flagellates Unclassified flagellates Ascomorpha eucadis   3.46e-13
Chromulina sp.                      Chromulina sp. Ascomorpha eucadis   3.03e-14
Selenastrum minutum           Selenastrum minutum Ascomorpha eucadis   2.72e-13
Trachelomonas sp.                Trachelomonas sp. Ascomorpha eucadis   1.75e-13
                         resource.B consumer.M consumer.B
Cryptomonas sp. 1        1.2992e-05    1.4e-10  1.932e-06
Chroococcus dispersus    4.7800e-06    1.4e-10  1.932e-06
Unclassified flagellates 6.5048e-04    1.4e-10  1.932e-06
Chromulina sp.           4.5147e-06    1.4e-10  1.932e-06
Selenastrum minutum      5.4400e-05    1.4e-10  1.932e-06
Trachelomonas sp.        3.8850e-05    1.4e-10  1.932e-06
```

TLPS takes a parameter link.properties, which should be a vector of names that are either first-class trophic-link properties or are functions. Functions should take a community as the first parameter and a second parameter that is a data.frame containing the columns 'resource' and 'consumer'. They should return either a vector of length NumberOfTrophicLinks or a matrix or data.frame with NumberOfTrophicLinks rows.

3.3.2 Trophic-link properties

Food web data in Cheddar communities can be augmented with extra information. The dataset of SkipwithPond (Warren, 1989) contains two such properties: 'link.evidence' and 'link.life.stage'.

```
> data(SkipwithPond)
> head(TLPS(SkipwithPond))
```

```
                                     resource         consumer link.evidence
1 Small oligochaetes (principally Enchytraeidae) Polycelis tenuis      Inferred
2                         Lumbriculus variegatus Polycelis tenuis      Inferred
3                          Procladius sagittalis Polycelis tenuis      Inferred
4                         Corynoneura scutellata Polycelis tenuis      Inferred
5                            Chironomus dorsalis Polycelis tenuis         Known
6                         Glyptotendipes pallens Polycelis tenuis         Known
  link.life.stage
1 All life stages
2 All life stages
3 All life stages
4 All life stages
5 All life stages
6 All life stages
```

TrophicLinkPropertyNames returns the names of the trophic-link properties in a community.

```
> TrophicLinkPropertyNames(SkipwithPond)
```

```
[1] "resource"        "consumer"        "link.evidence"   "link.life.stage"
```

TLPS accepts a 'link.properties' parameter. You can use this to get a subset of the first-class link properties. The columns 'resource' and 'consumer' are always returned.

```
> head(TLPS(SkipwithPond, link.properties='link.evidence'))
```

```
                                     resource         consumer link.evidence
1 Small oligochaetes (principally Enchytraeidae) Polycelis tenuis      Inferred
2                         Lumbriculus variegatus Polycelis tenuis      Inferred
3                          Procladius sagittalis Polycelis tenuis      Inferred
4                         Corynoneura scutellata Polycelis tenuis      Inferred
5                            Chironomus dorsalis Polycelis tenuis         Known
6                         Glyptotendipes pallens Polycelis tenuis         Known
```

TLPS takes a parameter link.properties, which should be a vector of names that are either first-class trophic-link properties or are functions. Functions should take a community as the only parameter.

They should return either a vector of length `NumberOfTrophicLinks` or a matrix or data.frame with `NumberOfTrophicLinks` rows. This is illustrated by the code fragment below, which uses the `Log10RCMRatio` function to get the $\log_{10}$-transformed ratio between body mass of the resource and consumer in each trophic link in `TL84`.

```
> head(TLPS(TL84, link.properties='Log10RCMRatio'))
```

```
                  resource           consumer Log10RCMRatio
1        Cryptomonas sp. 1 Ascomorpha eucadis     -2.838632
2    Chroococcus dispersus Ascomorpha eucadis     -2.767730
3 Unclassified flagellates Ascomorpha eucadis     -2.607052
4           Chromulina sp. Ascomorpha eucadis     -3.664685
5      Selenastrum minutum Ascomorpha eucadis     -2.711559
6        Trachelomonas sp. Ascomorpha eucadis     -2.903090
```

You can combine `node.properties` and `link.properties`.

```
> head(TLPS(TL84, node.properties='Log10M', link.properties='Log10RCMRatio'))
```

```
                  resource           consumer Log10RCMRatio resource.Log10M
1        Cryptomonas sp. 1 Ascomorpha eucadis     -2.838632       -12.69250
2    Chroococcus dispersus Ascomorpha eucadis     -2.767730       -12.62160
3 Unclassified flagellates Ascomorpha eucadis     -2.607052       -12.46092
4           Chromulina sp. Ascomorpha eucadis     -3.664685       -13.51856
5      Selenastrum minutum Ascomorpha eucadis     -2.711559       -12.56543
6        Trachelomonas sp. Ascomorpha eucadis     -2.903090       -12.75696
  consumer.Log10M
1       -9.853872
2       -9.853872
3       -9.853872
4       -9.853872
5       -9.853872
6       -9.853872
```

3.3.3 Predation matrix

The `PredationMatrix` function returns an R `matrix` object. The matrix returned by the code fragment below is 56 x 56 and so is not shown for brevity.

```
> pm <- PredationMatrix(TL84)
```

In the example above, all entries in 'pm' are either 0 or 1. This summation below computes the number of 1s in the matrix, which is the same as the number of trophic links in the community.

```
> sum(pm)
```

```
[1] 269
```

```
> NumberOfTrophicLinks(TL84)
```

```
[1] 269
```

Data that contain estimates of link strength can be used to construct a weighted predation matrix, such as the `Benguela` dataset, which contains the 'diet.fraction' node property (Yodzis, 1998).

```
> data(Benguela)
> pm <- PredationMatrix(Benguela, weight='diet.fraction')
```

More information about link strengths is in Section 3.3.8.

3.3.4 Node connectivity

A node in a community can be defined as falling in to one of four categories (Table 3). A node will

Category	Description
Isolated	No resources or consumers
Basal	No resources and one or more consumers
Top-level	One or more resources and no consumers
Intermediate	Nodes not fitting any of the above categories

Table 3: Node connectivity. Cannibalistic links are disregarded.

satisfy only one of the above four definitions. These definitions allow three additional definitions (Table 4). For each of the seven definitions (Tables 3 and 4), 'X', there are three functions: `IsXNode`, `XNodes`

Category	Description
Connected	Basal, Intermediate or Top-level
Non-basal	Isolated, Intermediate or Top-level
Non-top-level	Isolated, Basal or Intermediate

Table 4: Additional node connectivity

and `FractionXNodes`. The first returns a vector of type `logical` of length `NumberOfNodes`; values are `TRUE` for nodes that fit the definition of 'X'. The second returns the names of nodes for which `IsXNode` returns `TRUE`. The third returns the proportion of nodes in the community that fit the definition of 'X'. For example, a community's isolated species can be accessed by using `IsolatedNodes`.

```
> IsolatedNodes(TL84)

[1] "Asterionella formosa"        "Chrysosphaerella longispina"
[3] "Diceras sp."                 "Rhizosolenia sp."
[5] "Spinocosmarium sp."          "Staurastrum sp."
```

We can use the `IsXNode` functions together with `NPS` to see a table of connectivity for the whole community.

```
> connectivity <- NPS(TL84, c(Basal='IsBasalNode',
                              Isolated='IsIsolatedNode',
                              Intermediate='IsIntermediateNode',
                              TopLevel='IsTopLevelNode'))
> connectivity
```

	Basal	Isolated	Intermediate	TopLevel
Nostoc sp.	TRUE	FALSE	FALSE	FALSE
Arthrodesmus sp.	TRUE	FALSE	FALSE	FALSE
Asterionella formosa	FALSE	TRUE	FALSE	FALSE
Cryptomonas sp. 1	TRUE	FALSE	FALSE	FALSE
Cryptomonas sp. 2	TRUE	FALSE	FALSE	FALSE
Chroococcus dispersus	TRUE	FALSE	FALSE	FALSE
Closteriopsis longissimus	TRUE	FALSE	FALSE	FALSE
Chrysosphaerella longispina	FALSE	TRUE	FALSE	FALSE
Dinobryon bavaricum	TRUE	FALSE	FALSE	FALSE
Dinobryon cylindricum	TRUE	FALSE	FALSE	FALSE
Dactylococcopsis fascicularis	TRUE	FALSE	FALSE	FALSE
Diceras sp.	FALSE	TRUE	FALSE	FALSE
Dictyosphaerium pulchellum	TRUE	FALSE	FALSE	FALSE
Dinobryon sertularia	TRUE	FALSE	FALSE	FALSE
Dinobryon sociale	TRUE	FALSE	FALSE	FALSE
Glenodinium quadridens	TRUE	FALSE	FALSE	FALSE
Microcystis aeruginosa	TRUE	FALSE	FALSE	FALSE
Mallomonas sp. 1	TRUE	FALSE	FALSE	FALSE
Mallomonas sp. 2	TRUE	FALSE	FALSE	FALSE
Unclassified flagellates	TRUE	FALSE	FALSE	FALSE
Peridinium limbatum	TRUE	FALSE	FALSE	FALSE
Peridinium cinctum	TRUE	FALSE	FALSE	FALSE
Peridinium pulsillum	TRUE	FALSE	FALSE	FALSE
Peridinium wisconsinense	TRUE	FALSE	FALSE	FALSE
Chromulina sp.	TRUE	FALSE	FALSE	FALSE
Rhizosolenia sp.	FALSE	TRUE	FALSE	FALSE
Selenastrum minutum	TRUE	FALSE	FALSE	FALSE
Spinocosmarium sp.	FALSE	TRUE	FALSE	FALSE
Staurastrum sp.	FALSE	TRUE	FALSE	FALSE
Synedra sp.	TRUE	FALSE	FALSE	FALSE
Trachelomonas sp.	TRUE	FALSE	FALSE	FALSE
Ascomorpha eucadis	FALSE	FALSE	TRUE	FALSE
Synchaeta sp.	FALSE	FALSE	TRUE	FALSE
Bosmina longirostris	FALSE	FALSE	TRUE	FALSE
Conochilus (solitary)	FALSE	FALSE	TRUE	FALSE
Cyclops varians rubellus	FALSE	FALSE	TRUE	FALSE
Diaphanosoma leuchtenbergianum	FALSE	FALSE	TRUE	FALSE
Daphnia pulex	FALSE	FALSE	TRUE	FALSE
Filinia longispina	FALSE	FALSE	TRUE	FALSE
Conochiloides dossuarius	FALSE	FALSE	TRUE	FALSE
Gastropus stylifer	FALSE	FALSE	TRUE	FALSE
Holopedium gibberum	FALSE	FALSE	TRUE	FALSE
Kellicottia sp.	FALSE	FALSE	TRUE	FALSE
Keratella cochlearis	FALSE	FALSE	TRUE	FALSE
Keratella testudo	FALSE	FALSE	TRUE	FALSE
Leptodiaptomus siciloides	FALSE	FALSE	TRUE	FALSE
Orthocyclops modestus	FALSE	FALSE	TRUE	FALSE

```
Ploesoma sp.                  FALSE    FALSE         TRUE    FALSE
Polyarthra vulgaris           FALSE    FALSE         TRUE    FALSE
Trichocerca multicrinis       FALSE    FALSE         TRUE    FALSE
Trichocerca cylindrica        FALSE    FALSE         TRUE    FALSE
Tropocyclops prasinus         FALSE    FALSE         TRUE    FALSE
Chaoborus punctipennis        FALSE    FALSE         TRUE    FALSE
Phoxinus eos                  FALSE    FALSE         TRUE    FALSE
Phoxinus neogaeus             FALSE    FALSE         TRUE    FALSE
Umbra limi                    FALSE    FALSE        FALSE     TRUE
```

Because nodes can fit only one of the definitions in Table 3, each row in the `connectivity data.frame` should have one, and only one, value of `TRUE`. We can verify this by summing each row using R 's `apply` function.

```
> all(1==apply(connectivity, 1, sum))

[1] TRUE
```

The following summations are also 1.

```
> sum(FractionBasalNodes(TL84),
     FractionIntermediateNodes(TL84),
     FractionTopLevelNodes(TL84),
     FractionIsolatedNodes(TL84))

[1] 1

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

[1] 1

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

[1] 1

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

[1] 1
```

3.3.5 Trophic chains

Some network properties and analyses require knowledge of every unique path - 'trophic chain' - through the food-web. A trophic chain starts with a basal node (Section 3.3.4) and ends when it is not possible to add nodes that are not already in the chain. Loops and cannibalism are therefore ignored when computing trophic chains. For communities that have one or more top-level nodes each trophic chain will end with a top-level node. The 'length' of a chain is defined as the number of links that it contains, i.e. the number of nodes in the chain minus one.

Cheddar provides two functions for examining trophic chains. The `TrophicChains` function returns a `data.frame` of trophic chains.

```
> tc <- TrophicChains(TL84)
> dim(tc)
```

```
[1] 5988    8
```

There are 5988 unique chains in the food web and the longest chains contain 8 nodes. Let's look at the first 20 chains.

```
> head(tc, 20)
```

```
       Node.1                          Node.2                  Node.3            Node.4
1  Nostoc sp. Diaphanosoma leuchtenbergianum              Umbra limi
2  Nostoc sp.                   Daphnia pulex              Umbra limi
3  Nostoc sp.            Holopedium gibberum              Umbra limi
4  Nostoc sp.      Leptodiaptomus siciloides              Umbra limi
5  Nostoc sp. Diaphanosoma leuchtenbergianum   Chaoborus punctipennis        Umbra limi
6  Nostoc sp. Diaphanosoma leuchtenbergianum            Phoxinus eos        Umbra limi
7  Nostoc sp. Diaphanosoma leuchtenbergianum        Phoxinus neogaeus        Umbra limi
8  Nostoc sp.                   Daphnia pulex   Chaoborus punctipennis        Umbra limi
9  Nostoc sp.                   Daphnia pulex            Phoxinus eos        Umbra limi
10 Nostoc sp.                   Daphnia pulex        Phoxinus neogaeus        Umbra limi
11 Nostoc sp.            Holopedium gibberum            Phoxinus eos        Umbra limi
12 Nostoc sp.            Holopedium gibberum        Phoxinus neogaeus        Umbra limi
13 Nostoc sp.      Leptodiaptomus siciloides Cyclops varians rubellus        Umbra limi
14 Nostoc sp.      Leptodiaptomus siciloides    Orthocyclops modestus        Umbra limi
15 Nostoc sp.      Leptodiaptomus siciloides    Tropocyclops prasinus        Umbra limi
16 Nostoc sp.      Leptodiaptomus siciloides   Chaoborus punctipennis        Umbra limi
17 Nostoc sp.      Leptodiaptomus siciloides            Phoxinus eos        Umbra limi
18 Nostoc sp.      Leptodiaptomus siciloides        Phoxinus neogaeus        Umbra limi
19 Nostoc sp. Diaphanosoma leuchtenbergianum   Chaoborus punctipennis      Phoxinus eos
20 Nostoc sp. Diaphanosoma leuchtenbergianum   Chaoborus punctipennis Phoxinus neogaeus
       Node.5 Node.6 Node.7 Node.8
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
```

```
18
19 Umbra limi
20 Umbra limi
```

Every chain starts with a basal node.

```
> BasalNodes(TL84)
```

```
 [1] "Nostoc sp."                    "Arthrodesmus sp."
 [3] "Cryptomonas sp. 1"             "Cryptomonas sp. 2"
 [5] "Chroococcus dispersus"         "Closteriopsis longissimus"
 [7] "Dinobryon bavaricum"           "Dinobryon cylindricum"
 [9] "Dactylococcopsis fascicularis" "Dictyosphaerium pulchellum"
[11] "Dinobryon sertularia"          "Dinobryon sociale"
[13] "Glenodinium quadridens"        "Microcystis aeruginosa"
[15] "Mallomonas sp. 1"              "Mallomonas sp. 2"
[17] "Unclassified flagellates"      "Peridinium limbatum"
[19] "Peridinium cinctum"            "Peridinium pulsillum"
[21] "Peridinium wisconsinense"      "Chromulina sp."
[23] "Selenastrum minutum"           "Synedra sp."
[25] "Trachelomonas sp."
```

```
> # The first node in each chain
> first <- tc[,1]
> all(unique(first) %in% BasalNodes(TL84))  # TRUE
```

```
[1] TRUE
```

Tuesday Lake 1984 has a single top-level consumer so every trophic chain ends with this consumer.

```
> TopLevelNodes(TL84)
```

```
[1] "Umbra limi"
```

```
> # The last node in each chain
> last <- apply(tc, 1, function(row) row[max(which(""!=row))])
> unique(last)
```

```
[1] "Umbra limi"
```

Just as with `TLPS`, `TrophicChains` accepts a 'node.properties' parameter that you can use to add node properties to the returned. For example, to get a table containing the $\log_{10}$-transformed body mass of each node in every chain.

```
> tc.with.log10M <- TrophicChains(TL84, node.properties='Log10M')
```

The food web of the Bengula marine ecosystem (Yodzis, 1998) does not have any top-level nodes. All chains start with a basal node (by definition) but, for this community, all chains end with an intermediate node.

```
> TopLevelNodes(Benguela)
```

```
character(0)
```

```
> tc <- TrophicChains(Benguela)
> last <- apply(tc, 1, function(row) row[max(which(""!=row))])
> unique(last)
```

```
[1] "Sharks" "Seals"  "Birds"
```

```
> IsIntermediateNode(Benguela)[unique(last)]
```

```
Sharks  Seals  Birds
  TRUE   TRUE   TRUE
```

The second function - `TrophicChainsStats` - returns a list of simple statistics about trophic chains.

```
> chain.stats <- TrophicChainsStats(TL84)
```

The 'chain.lengths' item contains the length of every unique food chain.

```
> length(chain.stats$chain.lengths)    # 5,988 chains
```

```
[1] 5988
```

```
> summary(chain.stats$chain.lengths)
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  2.000   4.000   5.000   4.835   6.000   7.000
```

The 'node.pos.counts' item is 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.

```
> dim(chain.stats$node.pos.counts)    # 56 nodes. Longest chain contains 8 nodes
```

```
[1] 56  8
```

Basal nodes only have counts in the first column.

```
> chain.stats$node.pos.counts[BasalNodes(TL84),]
```

```
                            [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8]
Nostoc sp.                    75    0    0    0    0    0    0    0
Arthrodesmus sp.              69    0    0    0    0    0    0    0
Cryptomonas sp. 1            840    0    0    0    0    0    0    0
Cryptomonas sp. 2             81    0    0    0    0    0    0    0
Chroococcus dispersus        840    0    0    0    0    0    0    0
Closteriopsis longissimus    120    0    0    0    0    0    0    0
Dinobryon bavaricum            6    0    0    0    0    0    0    0
Dinobryon cylindricum         75    0    0    0    0    0    0    0
Dactylococcopsis fascicularis 75    0    0    0    0    0    0    0
Dictyosphaerium pulchellum    81    0    0    0    0    0    0    0
Dinobryon sertularia           9    0    0    0    0    0    0    0
Dinobryon sociale             81    0    0    0    0    0    0    0
```

Glenodinium quadridens	84	0	0	0	0	0	0	0
Microcystis aeruginosa	6	0	0	0	0	0	0	0
Mallomonas sp. 1	9	0	0	0	0	0	0	0
Mallomonas sp. 2	69	0	0	0	0	0	0	0
Unclassified flagellates	840	0	0	0	0	0	0	0
Peridinium limbatum	6	0	0	0	0	0	0	0
Peridinium cinctum	9	0	0	0	0	0	0	0
Peridinium pulsillum	81	0	0	0	0	0	0	0
Peridinium wisconsinense	6	0	0	0	0	0	0	0
Chromulina sp.	840	0	0	0	0	0	0	0
Selenastrum minutum	840	0	0	0	0	0	0	0
Synedra sp.	6	0	0	0	0	0	0	0
Trachelomonas sp.	840	0	0	0	0	0	0	0

Consumers only have counts in columns two and higher.

```
> chain.stats$node.pos.counts[c(IntermediateNodes(TL84),TopLevelNodes(TL84)),]
```

	[,1]	[,2]	[,3]	[,4]	[,5]	[,6]	[,7]	[,8]
Ascomorpha eucadis	0	378	0	0	0	0	0	0
Synchaeta sp.	0	378	0	0	0	0	0	0
Bosmina longirostris	0	66	0	0	0	0	0	0
Conochilus (solitary)	0	399	0	0	0	0	0	0
Cyclops varians rubellus	0	0	2304	1152	0	0	0	0
Diaphanosoma leuchtenbergianum	0	84	0	0	0	0	0	0
Daphnia pulex	0	150	108	0	0	0	0	0
Filinia longispina	0	342	0	0	0	0	0	0
Conochiloides dossuarius	0	399	0	0	0	0	0	0
Gastropus stylifer	0	342	0	0	0	0	0	0
Holopedium gibberum	0	57	0	0	0	0	0	0
Kellicottia sp.	0	342	0	0	0	0	0	0
Keratella cochlearis	0	378	0	0	0	0	0	0
Keratella testudo	0	342	0	0	0	0	0	0
Leptodiaptomus siciloides	0	960	0	0	0	0	0	0
Orthocyclops modestus	0	0	576	1152	1152	0	0	0
Ploesoma sp.	0	342	0	0	0	0	0	0
Polyarthra vulgaris	0	342	0	0	0	0	0	0
Trichocerca multicrinis	0	342	0	0	0	0	0	0
Trichocerca cylindrica	0	342	0	0	0	0	0	0
Tropocyclops prasinus	0	0	2304	1152	0	0	0	0
Chaoborus punctipennis	0	3	438	918	1152	576	0	0
Phoxinus eos	0	0	86	452	690	576	192	0
Phoxinus neogaeus	0	0	86	452	690	576	192	0
Umbra limi	0	0	86	624	1594	1956	1344	384

All counts are zero for isolated nodes.

```
> chain.stats$node.pos.counts[IsolatedNodes(TL84),]
```

```
                          [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8]
Asterionella formosa         0    0    0    0    0    0    0    0
Chrysosphaerella longispina  0    0    0    0    0    0    0    0
Diceras sp.                  0    0    0    0    0    0    0    0
Rhizosolenia sp.             0    0    0    0    0    0    0    0
Spinocosmarium sp.           0    0    0    0    0    0    0    0
Staurastrum sp.              0    0    0    0    0    0    0    0
```

If your analysis requires only simple statistics about trophic chains, the `TrophicChainsStats` function is more suitable than `TrophicChains` because it is faster and requires less memory.

```
> system.time(tc <- TrophicChains(TL84))

   user  system elapsed
  0.013   0.000   0.013

> system.time(stats <- TrophicChainsStats(TL84))

   user  system elapsed
  0.011   0.000   0.011
```

The difference in speed will be greater for communities that contain a large number of nodes and trophic chains such as the Skipwith Pond dataset, which has more than 10^5 unique chains.

3.3.6 Large numbers of trophic chains

The number of possible trophic chains rapidy increases with the number of nodes and links. Let's examine the relationship between number of trophic chains and number of nodes communities in which every consumer eats every other node.

```
> HighlyConnected <- function(n)
 {
     # Returns a community containing a single producer and n consumers, all
     # of whom eat everything else
     consumers <- paste('Consumer', 1:n)
     tl <- data.frame(resource=c(rep('Producer', n), rep(consumers, each=n)),
                      consumer=consumers)
     return (Community(nodes=data.frame(node=c('Producer',consumers)),
                       trophic.links=tl,
                       properties=list(title='test')))
 }
> # A list of communities of between 1 and 8 consumers
> n <- 8
> communities <- lapply(1:n, HighlyConnected)
> # A list of statistics about each community
> stats <- lapply(communities, TrophicChainsStats)
> # Extract the chain lengths
> cl <- lapply(stats, '[[', 'chain.lengths')
> # The number of chains
> n.chains <- sapply(cl, length)
> # The number of chains in each community
> cbind(n.consumers=1:n, longest.chain=sapply(cl, max), n.chains=n.chains)
```

```
     n.consumers longest.chain n.chains
[1,]           1             1        1
[2,]           2             2        2
[3,]           3             3        6
[4,]           4             4       24
[5,]           5             5      120
[6,]           6             6      720
[7,]           7             7     5040
[8,]           8             8    40320
```

So with one consumer there is one chain, two consumers - two chains, three consumers - six chains, four consumers - 24 chains and so on. You may recognise this sequence as factorial.

```
> cbind(n.consumers=1:n, longest.chain=sapply(cl, max), n.chains=n.chains,
       factorial.n=factorial(1:n))
```

```
     n.consumers longest.chain n.chains factorial.n
[1,]           1             1        1           1
[2,]           2             2        2           2
[3,]           3             3        6           6
[4,]           4             4       24          24
[5,]           5             5      120         120
[6,]           6             6      720         720
[7,]           7             7     5040        5040
[8,]           8             8    40320       40320
```

What happens if we have even more consumers?

```
> n <- 20
> cbind(n.consumers=1:n, longest.chain=1:n, factorial.n=factorial(1:n))
```

```
      n.consumers longest.chain  factorial.n
 [1,]           1             1 1.000000e+00
 [2,]           2             2 2.000000e+00
 [3,]           3             3 6.000000e+00
 [4,]           4             4 2.400000e+01
 [5,]           5             5 1.200000e+02
 [6,]           6             6 7.200000e+02
 [7,]           7             7 5.040000e+03
 [8,]           8             8 4.032000e+04
 [9,]           9             9 3.628800e+05
[10,]          10            10 3.628800e+06
[11,]          11            11 3.991680e+07
[12,]          12            12 4.790016e+08
[13,]          13            13 6.227021e+09
[14,]          14            14 8.717829e+10
[15,]          15            15 1.307674e+12
[16,]          16            16 2.092279e+13
[17,]          17            17 3.556874e+14
```

```
[18,]          18            18 6.402374e+15
[19,]          19            19 1.216451e+17
[20,]          20            20 2.432902e+18
```

The number of trophic chains quickly becomes too large to compute within practical time and within available memory, and this for communities with just a single producer. When computing chains, cheddar allocates memory as required, with a safety limit to prevent too much of the system's memory from being consumed. If you see an error message 'Unable to compute paths' then this safety limit has been reached. The limit can be altered by setting the 'cheddarMaxQueue' option.

```
> # Set to a low number to illustrate the error
> options(cheddarMaxQueue=10)
> tryCatch(TrophicChains(TL84), error=print)

<simpleError in .TrophicChainsSize(community): Unexpected error>

> # Default value
> options(cheddarMaxQueue=NULL)
> chains <- TrophicChains(TL84)
```

If you encounter this error message you can increase the value of 'cheddarMaxQueue' (from it's default of $1e7$), but it is likely that the food-web is so complex that it will not be possible to compute all paths. Setting 'cheddarMaxQueue' to 0 disables the safety limit.

3.3.7 Trophic level

Several different measures of trophic level are used (e.g. Williams and Martinez, 2004; Jonsson et al., 2005; Zook et al., 2011.) The `PreyAveragedTrophicLevel` function uses the matrix-inversion method of Levine (1980) to compute trophic levels (Williams and Martinez, 2004). This method is very fast and accounts for flow through loops. A different measure of trophic level is offered by the `ChainAveragedTrophicLevel` function, which enumerates every unique food chain in the web (using `TrophicChainsStats`) and computes the mean position of each node in every chain (Williams and Martinez, 2004). The method of `ChainAveragedTrophicLevel` is the same as that described as 'trophic height' by Jonsson et al. (2005) and the name `TrophicHeight` is a synonym for `ChainAveragedTrophicLevel`. `ChainAveragedTrophicLevel` might be noticeably slower than `PreyAveragedTrophicLevel` for very large and/or highly connected food webs.

```
> tail(NPS(TL84, c('PreyAveragedTrophicLevel', 'ChainAveragedTrophicLevel')), 10)

                       PreyAveragedTrophicLevel ChainAveragedTrophicLevel
Orthocyclops modestus                  3.205357                 4.200000
Ploesoma sp.                           2.000000                 2.000000
Polyarthra vulgaris                    2.000000                 2.000000
Trichocerca multicrinis                2.000000                 2.000000
Trichocerca cylindrica                 2.000000                 2.000000
Tropocyclops prasinus                  3.142857                 3.333333
Chaoborus punctipennis                 3.171344                 4.602527
Phoxinus eos                           3.529951                 5.168337
Phoxinus neogaeus                      3.529951                 5.168337
Umbra limi                             3.802678                 5.835003
```

Cheddar offers the six different measures of trophic level described by Williams and Martinez (2004). A function is provided for each one. The `TrophicLevels` convenience function returns a matrix containing all six.

```
> tail(TrophicLevels(TL84), 10)
```

```
                        ShortestTL ShortWeightedTL LongestTL LongWeightedTL
Orthocyclops modestus            3        3.102679         5       4.102679
Ploesoma sp.                     2        2.000000         2       2.000000
Polyarthra vulgaris              2        2.000000         2       2.000000
Trichocerca multicrinis          2        2.000000         2       2.000000
Trichocerca cylindrica           2        2.000000         2       2.000000
Tropocyclops prasinus            3        3.071429         4       3.571429
Chaoborus punctipennis           2        2.585672         6       4.585672
Phoxinus eos                     3        3.264975         7       5.264975
Phoxinus neogaeus                3        3.264975         7       5.264975
Umbra limi                       3        3.401339         8       5.901339
                        ChainAveragedTL PreyAveragedTL
Orthocyclops modestus          4.200000       3.205357
Ploesoma sp.                   2.000000       2.000000
Polyarthra vulgaris            2.000000       2.000000
Trichocerca multicrinis        2.000000       2.000000
Trichocerca cylindrica         2.000000       2.000000
Tropocyclops prasinus          3.333333       3.142857
Chaoborus punctipennis         4.602527       3.171344
Phoxinus eos                   5.168337       3.529951
Phoxinus neogaeus              5.168337       3.529951
Umbra limi                     5.835003       3.802678
```

See the help page for `TrophicLevels` for more information on these different measures.

3.3.8 Link strengths

Cheddar's data format allows zero, one or many measures of link strength to be defined simply by adding additional columns to the trophic.links.csv file (Section 3). It is also possible to define and use theoretical measures of link strength. Cheddar's `PredationMatrix` and `FlowBasedTrophicLevel` functions allow empirical and/or theoretical link strengths to be used.

The `Benguela` dataset contains empirical estimates of diet fraction for each trophic link (Yodzis, 1998), available as the 'diet.fraction' property.

```
> head(TLPS(Benguela))
```

```
                resource           consumer diet.fraction
1          Phytoplankton           Bacteria          98.8
2 Benthic filter feeders Benthic carnivores         100.0
3          Phytoplankton   Microzooplankton          30.6
4               Bacteria   Microzooplankton          29.7
5       Microzooplankton   Microzooplankton          29.7
6          Phytoplankton    Mesozooplankton          45.0
```

A binary predation matrix contains just '0' and '1'.

```
> pm <- PredationMatrix(Benguela)
```

We can weight the predation matrix by empirical diet fractions.

```
> pm <- PredationMatrix(Benguela, weight='diet.fraction')
```

These matrices are 29 x 29 and so are not shown for brevity.

The `FlowBasedTrophicLevel` function uses the same matrix inversion technique as `PreyAveragedTrophicLevel` and uses the 'weight.by' node property to provide an estimate of energy flow through each trophic link. We can easily compare these two different ways of computing trophic level.

```
> cbind(PreyAveragedTrophicLevel(Benguela),
       FlowBasedTrophicLevel(Benguela, weight.by='diet.fraction'))
```

```
                          [,1]     [,2]
Phytoplankton          1.000000 1.000000
Benthic filter feeders 1.000000 1.000000
Bacteria               2.000000 2.000000
Benthic carnivores     2.000000 2.000000
Microzooplankton       3.000000 2.985075
Mesozooplankton        3.000000 2.992537
Macrozooplankton       3.000000 2.797015
Gelatinous zoolplankton 3.250000 3.244403
Anchovy                3.500000 3.826134
Pilchard               3.333333 2.651672
Round herring          4.000000 3.914328
Lightfish              4.000000 3.875224
Lanternfish            4.000000 3.875224
Goby                   3.500000 3.238903
Other pelagics         3.812500 3.952869
Horse mackerel         4.000000 3.908589
Chub mackerel          4.250000 4.165017
Other groundfish       4.776994 4.144872
Hakes                  4.930366 4.767544
Squid                  4.804069 4.609870
Tunas                  4.844189 4.421241
Snoek                  4.733434 4.713249
Kob                    4.909476 4.658580
Yellowtail             4.897690 4.719394
Geelbek                4.865726 4.526048
Whales and dolphins    4.888027 4.769149
Birds                  4.985516 4.737059
Seals                  5.113082 4.829057
Sharks                 5.183100 4.933519
```

Theoretical per capita interaction strengths can be computed from body-mass ratios raised to a power, typically taken to be 2/3 or 3/4 (Emmerson and Raffaelli, 2004; Reuman and Cohen, 2005; Layer et al., 2010). We can define a function to compute these theoretical interaction strengths and use it in computation of trophic levels.

```
> InteractionStrength <- function(community)
 {
     tlps <- TLPS(community, node.properties='M')
     return ((tlps$consumer.M / tlps$resource.M)^3/4)
 }
> # The InteractionStrength() function can be used together with TLPS() to
> # compute the theoretical interaction strength between each resource-consumer pair
> head(TLPS(Benguela, link.properties='InteractionStrength'))

                resource           consumer InteractionStrength
1          Phytoplankton           Bacteria             2.5e-13
2 Benthic filter feeders Benthic carnivores             2.5e-01
3          Phytoplankton   Microzooplankton             2.5e-01
4               Bacteria   Microzooplankton             2.5e+11
5       Microzooplankton   Microzooplankton             2.5e-01
6          Phytoplankton    Mesozooplankton             2.5e+05
```

We can use this measure of interaction strength to compute flow-based trophic level.

```
> cbind(PreyAveragedTrophicLevel(Benguela),
       FlowBasedTrophicLevel(Benguela, weight.by='diet.fraction'),
       FlowBasedTrophicLevel(Benguela, weight.by='InteractionStrength'))

                           [,1]     [,2]     [,3]
Phytoplankton           1.000000 1.000000 1.000000
Benthic filter feeders  1.000000 1.000000 1.000000
Bacteria                2.000000 2.000000 2.000000
Benthic carnivores      2.000000 2.000000 2.000000
Microzooplankton        3.000000 2.985075 3.000000
Mesozooplankton         3.000000 2.992537 3.000000
Macrozooplankton        3.000000 2.797015 2.000002
Gelatinous zoolplankton 3.250000 3.244403 3.000000
Anchovy                 3.500000 3.826134 3.000000
Pilchard                3.333333 2.651672 2.000002
Round herring           4.000000 3.914328 3.999999
Lightfish               4.000000 3.875224 3.999999
Lanternfish             4.000000 3.875224 3.999999
Goby                    3.500000 3.238903 3.000000
Other pelagics          3.812500 3.952869 3.999999
Horse mackerel          4.000000 3.908589 3.999999
Chub mackerel           4.250000 4.165017 3.999999
Other groundfish        4.776994 4.144872 3.999999
Hakes                   4.930366 4.767544 3.999999
Squid                   4.804069 4.609870 3.018712
Tunas                   4.844189 4.421241 4.796740
Snoek                   4.733434 4.713249 3.024660
Kob                     4.909476 4.658580 3.000829
Yellowtail              4.897690 4.719394 3.000830
Geelbek                 4.865726 4.526048 3.453351
```

```
Whales and dolphins     4.888027 4.769149 3.006892
Birds                   4.985516 4.737059 3.999999
Seals                   5.113082 4.829057 4.796740
Sharks                  5.183100 4.933519 3.999999
```

The weighting functionality offered by the `PredationMatrix` and `FlowBasedTrophicLevel` functions make it possible to use multiple empirical and/or theoretical link strengths and interaction strengths in analyses.

4 Community manipulations

4.1 Node order

The ordering of nodes within a community can be important both for presentation and analysis. Cheddar's `OrderCommunity` function reorders nodes and returns a new community object. `OrderCommunity` accepts names that meets the criteria of the `properties` parameter of the `NPS` function. This includes the names of 'first-class' properties, such as M, and the names of functions that take a single community and return a value for each node, such as `Degree`, which returns the number of trophic links for each node. The following examples order `TL84` by increasing body mass and by increasing degree.

```
> TL84.increasing.M <- OrderCommunity(TL84, 'M', title='Increasing M')
> head(NPS(TL84.increasing.M, c('M', 'Degree')))
```

```
                                  M Degree
Chromulina sp.                3.03e-14     18
Dactylococcopsis fascicularis 1.32e-13      4
Diceras sp.                   1.53e-13      0
Trachelomonas sp.             1.75e-13     18
Cryptomonas sp. 1             2.03e-13     18
Closteriopsis longissimus     2.37e-13      3
```

```
> TL84.increasing.degree <- OrderCommunity(TL84, 'Degree',
                                           title='Increasing degree')
> head(NPS(TL84.increasing.degree, c('M', 'Degree')))
```

```
                                M Degree
Asterionella formosa        1.12e-12      0
Chrysosphaerella longispina 8.31e-10      0
Diceras sp.                 1.53e-13      0
Rhizosolenia sp.            6.86e-13      0
Spinocosmarium sp.          3.71e-12      0
Staurastrum sp.             4.30e-12      0
```

Similar to R 's `order` function, `OrderCommunity` can sort by more than one name with subsequent names used to break ties. We can use this to sort alphabetically by category and then by increasing M within each category.

```
> TL84.category.then.M <- OrderCommunity(TL84, 'category', 'M')
> head(NPS(TL84.category.then.M, c('category', 'M')))
```

```
                         category        M
Keratella cochlearis  invertebrate 1.00e-11
Keratella testudo     invertebrate 1.00e-11
Kellicottia sp.       invertebrate 2.00e-11
Conochilus (solitary) invertebrate 3.50e-11
Ploesoma sp.          invertebrate 1.05e-10
Gastropus stylifer    invertebrate 1.35e-10
```

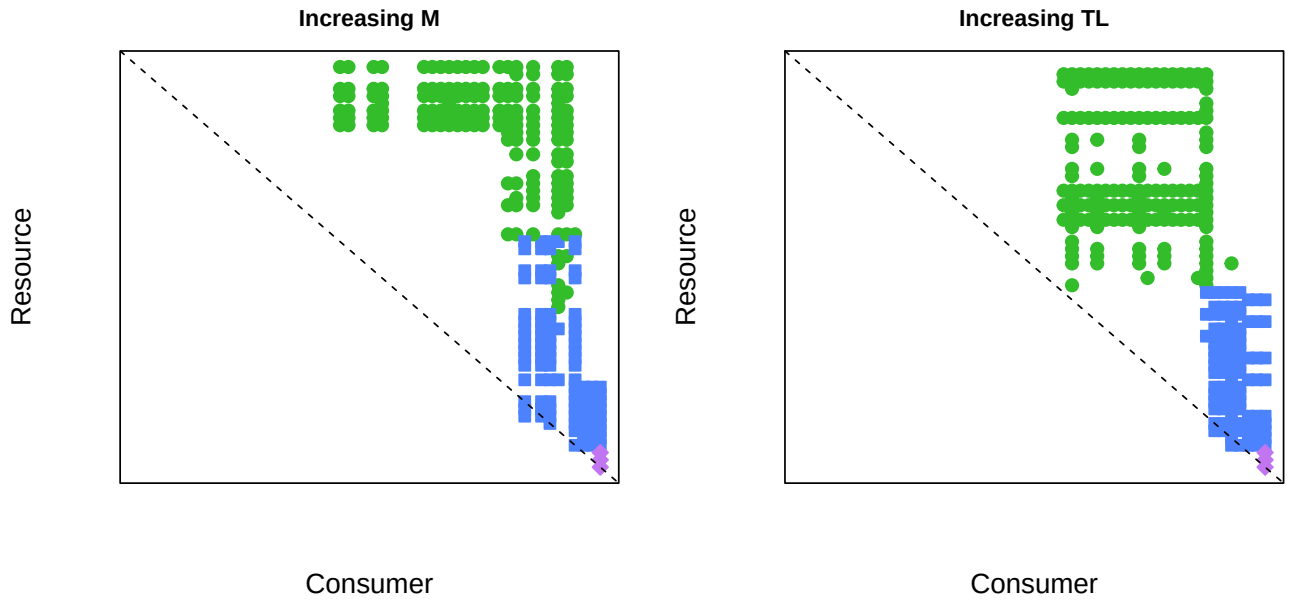
4.2 Node order and intervality

Visualising the food web as a predation matrix is central to many analyses and theories. There has been much recent interest in the relationship between food web structure and species' niches, in particular the role of body size on determining a species' position in a food web and the effect on intervality - a measure of the adjacency of resources and consumers in the food web (Williams and Martinez, 2000; Stouffer et al., 2006; Zook et al., 2011). We can use `OrderCommunity` to explore the effect ordering species along different niche axes. The code fragment below creates two new orderings of `TL84`, one by increasing body mass and the other by increasing trophic level, with random ordering within ties for trophic level (Zook et al., 2011).

```
> # Increasing M
> TL84.increasing.M <- OrderCommunity(TL84, 'M', title='Increasing M')
> new.order <- order(PreyAveragedTrophicLevel(TL84), sample(1:56))
> TL84.increasing.TL <- OrderCommunity(TL84, new.order=new.order,
                                       title='Increasing TL')
```

We could use any of Cheddar's different measure of trophic level (Section 3.3.7). The `PlotPredationMatrix` function allows us to graphically compare the effect of these different orderings.

```
> par(mfrow=c(1,2))
> PlotPredationMatrix(TL84.increasing.M)
> PlotPredationMatrix(TL84.increasing.TL)
```



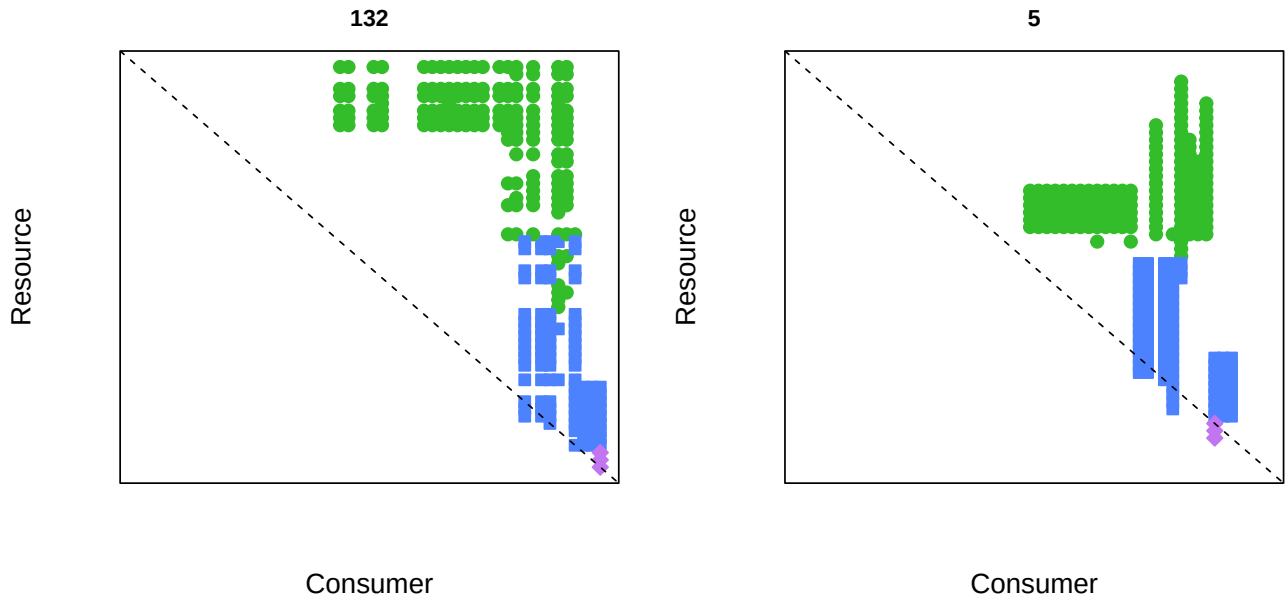
The total number of gaps in diets (columns) and consumers (rows) (Stouffer et al., 2011; Zook et al., 2011):

```
> SumDietGaps(TL84.increasing.M)

[1] 132

> SumDietGaps(TL84.increasing.TL)
```

```
[1] 322

> SumConsumerGaps(TL84.increasing.M)

[1] 154

> SumConsumerGaps(TL84.increasing.TL)

[1] 183
```

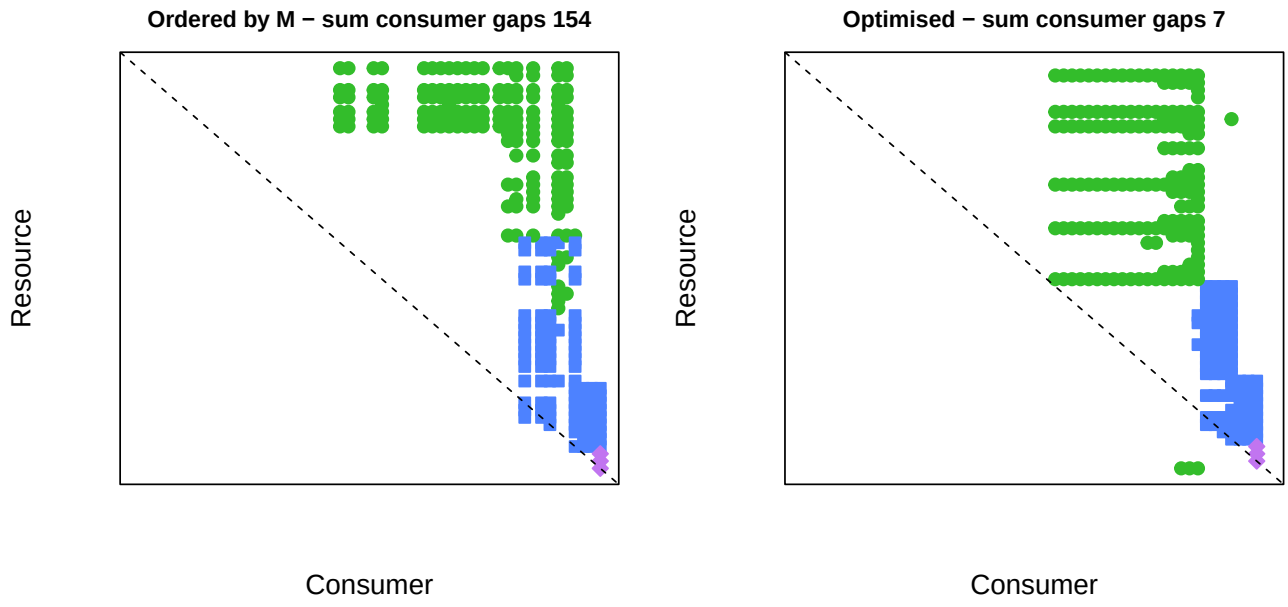
The `MinimiseSumDietGaps` function implements simulated annealing learning to minimise `SumDietGaps`, as described by Stouffer et al. (2006). Simulated annealing learning is a stochastic method so several optimisations might be required to find the global minimum; this is done by setting the 'n' parameter greater than 1:

```
> par(mfrow=c(1,2))
> PlotPredationMatrix(TL84.increasing.M, main=SumDietGaps(TL84.increasing.M))
> res <- MinimiseSumDietGaps(TL84, n=10)
> PlotPredationMatrix(res$reordered, main=SumDietGaps(res$reordered))
```



`MinimiseSumConsumerGaps` uses the same method to minimise the gaps in each species' consumers (Zook et al., 2011).

```
> par(mfrow=c(1,2))
> PlotPredationMatrix(TL84.increasing.M,
                      main=paste('Ordered by M - sum consumer gaps',
                                 SumConsumerGaps(TL84.increasing.M)))
> res <- MinimiseSumConsumerGaps(TL84, n=10)
> PlotPredationMatrix(res$reordered,
                      main=paste('Optimised - sum consumer gaps',
                                 SumConsumerGaps(res$reordered)))
```



4.3 Removing nodes

Isolated nodes are often removed from food-web analyses (e.g. Jonsson et al., 2005). `RemoveIsolatedNodes` is a convenience function that returns a new `Community` with isolated nodes removed.

```
> NumberOfNodes(TL84)

[1] 56

> IsolatedNodes(TL84)

[1] "Asterionella formosa"        "Chrysosphaerella longispina"
[3] "Diceras sp."                 "Rhizosolenia sp."
[5] "Spinocosmarium sp."          "Staurastrum sp."

> NumberOfTrophicLinks(TL84)

[1] 269
```

```
> TL84.no.isolated <- RemoveIsolatedNodes(TL84)
> NumberOfNodes(TL84.no.isolated)         # Six fewer species

[1] 50

> IsolatedNodes(TL84.no.isolated)         # No isolated species

character(0)

> NumberOfTrophicLinks(TL84.no.isolated)  # Number of trophic links unchanged

[1] 269
```

The general-purpose `RemoveNodes` function returns a new `Community` object with one or more nodes removed.

```
> NumberOfNodes(TL84)

[1] 56

> NumberOfTrophicLinks(TL84)

[1] 269

> # Remove the first ten nodes
> TL84.r <- RemoveNodes(TL84, 1:10)
> NumberOfNodes(TL84.r)

[1] 46

> NumberOfTrophicLinks(TL84.r)

[1] 213

> # Remove producers
> TL84.r <- RemoveNodes(TL84, 'producer'==NP(TL84, 'category'))
> NumberOfNodes(TL84.r)

[1] 25

> NumberOfTrophicLinks(TL84.r)

[1] 103

> # Remove species by name
> to.remove <- c("Cryptomonas sp. 1", "Chroococcus dispersus",
                 "Unclassified flagellates", "Chromulina sp.",
                 "Selenastrum minutum", "Trachelomonas sp.")
> TL84.r <- RemoveNodes(TL84, to.remove)
> NumberOfNodes(TL84.r)

[1] 50
```

```
> NumberOfTrophicLinks(TL84.r)

[1] 161

> # Three different ways of removing node 56 (Umbra limi)
> TL84.ra <- RemoveNodes(TL84, 56)
> TL84.rb <- RemoveNodes(TL84, 'Umbra limi')
> TL84.rc <- RemoveNodes(TL84, c(rep(FALSE,55), TRUE))
> identical(TL84.ra, TL84.rb)  # TRUE

[1] TRUE

> identical(TL84.ra, TL84.rc)  # TRUE

[1] TRUE
```

The `RemoveNodes` function takes an argument called 'method', which indicates how removals should be propogated through the food web. If 'method' is 'direct' (the default), 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.

```
> # The behaviours of the different methods
> NumberOfNodes(TL84)         # 56 nodes in total

[1] 56

> length(BasalNodes(TL84))    # 25 basal nodes

[1] 25

> length(IsolatedNodes(TL84)) #  6 isolated nodes

[1] 6

> RemoveNodes(TL84, BasalNodes(TL84)) # 56 - 25 = 31 nodes remain

Tuesday Lake sampled in 1984 (25 nodes directly removed) containing 31 nodes and 103 trophic links

> RemoveNodes(TL84, BasalNodes(TL84), method='secondary') # 14 nodes remain

Tuesday Lake sampled in 1984 (25 nodes directly removed) containing 14 nodes and 30 trophic links

> RemoveNodes(TL84, BasalNodes(TL84), method='cascade')   # The 6 isolated nodes remain

Tuesday Lake sampled in 1984 (25 nodes directly removed) containing 6 nodes
```

4.4 Removing cannibalistic links

`RemoveCannibalisticLinks` returns a new `Community` without those trophic links in which a node consumes itself.

```
> NumberOfNodes(TL84)

[1] 56

> Cannibals(TL84)         # 5 species

[1] "Cyclops varians rubellus" "Orthocyclops modestus"    "Tropocyclops prasinus"
[4] "Chaoborus punctipennis"   "Umbra limi"

> NumberOfTrophicLinks(TL84)

[1] 269

> TL84.no.cannibals <- RemoveCannibalisticLinks(TL84)
> NumberOfNodes(TL84.no.cannibals)         # Number of nodes unchanged

[1] 56

> Cannibals(TL84.no.cannibals)             # No species

character(0)

> NumberOfTrophicLinks(TL84.no.cannibals)  # 5 fewer trophic links

[1] 264
```

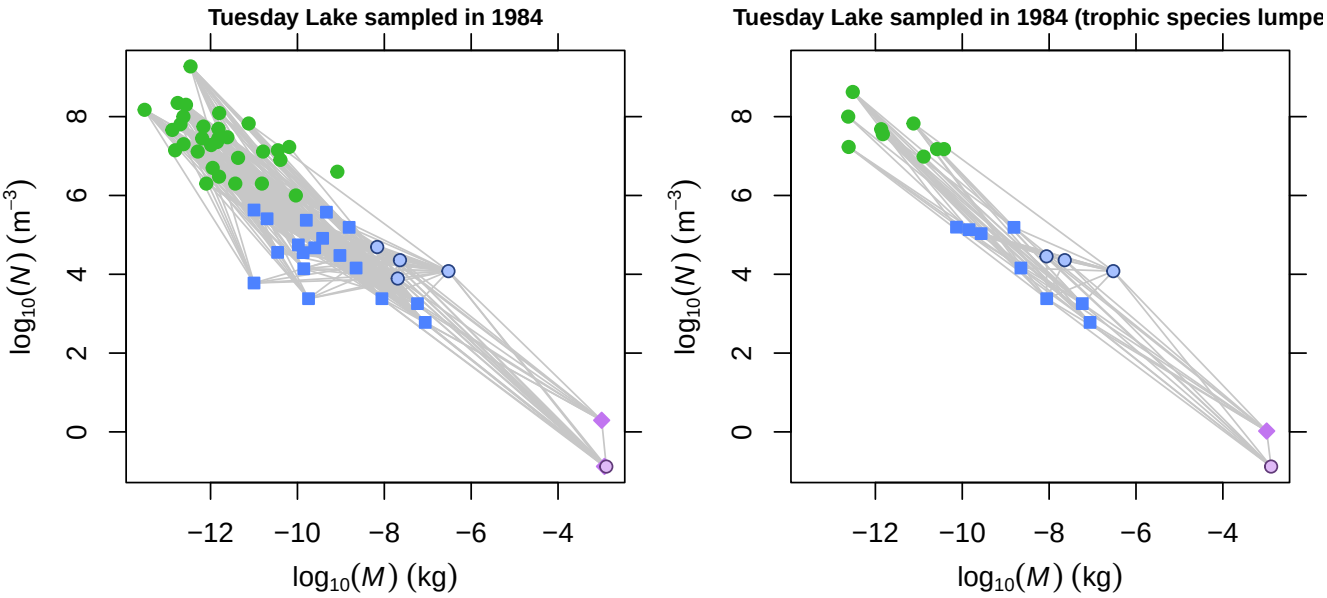
4.5 Lumping nodes

Certain analyses call for food-web nodes to be merged. In order to reduce biases, nodes that share the same resources and consumers, so-called 'trophic species', are commonly lumped together (Briand and Cohen, 1984; Pimm et al., 1991; Williams and Martinez, 2000). The `LumpTrophicSpecies` performs this task and returns a new `Community` object.

```
> NumberOfNodes(TL84)

[1] 56

> TL84.lumped <- LumpTrophicSpecies(TL84)
> length(unique(TrophicSpecies(TL84)))    # 22 trophic species in TL84...

[1] 22

> NumberOfNodes(TL84.lumped)              # ... and 22 nodes in the lumped web

[1] 22
```

The plot below shows the lumped and unlumped webs.

```
> par(mfrow=c(1,2))
> plot(TL84)
> plot(TL84.lumped, xlim=range(Log10M(TL84)), ylim=range(Log10N(TL84)))
```



The `LumpNodes` function is a more general-purpose function that allows any nodes in a community to be lumped together. It takes a parameter 'lump', which should be a vector of length `NumberOfNodes`. Nodes with the same value of 'lump' will be merged together. This example lumps together isolated species in TL84.

```
> length(which(IsIsolatedNode(TL84)))  # 6 isolated species

[1] 6

> IsolatedNodes(TL84)                  # Names of isolated nodes

[1] "Asterionella formosa"        "Chrysosphaerella longispina"
[3] "Diceras sp."                 "Rhizosolenia sp."
[5] "Spinocosmarium sp."          "Staurastrum sp."

> 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

[1] 56

> NumberOfNodes(TL84.lumped)  # 51 nodes in lumped web
```

```
[1] 51
```

```
> IsolatedNodes(TL84.lumped)  # A single node
```

```
[1] "Isolated nodes lumped together"
```

By default, numeric values are weighted by numerical abundance, N.

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')))
```

```
[1] FALSE
```

The Ythan Estuary dataset contains two species that are split by life stage: *Platichthys flesus* (european flounder) and *Somateria mollissima* (common eider). The code fragment below shows how to lump these in to a single node for each 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)
> NumberOfNodes(YthanEstuary)         # 92
```

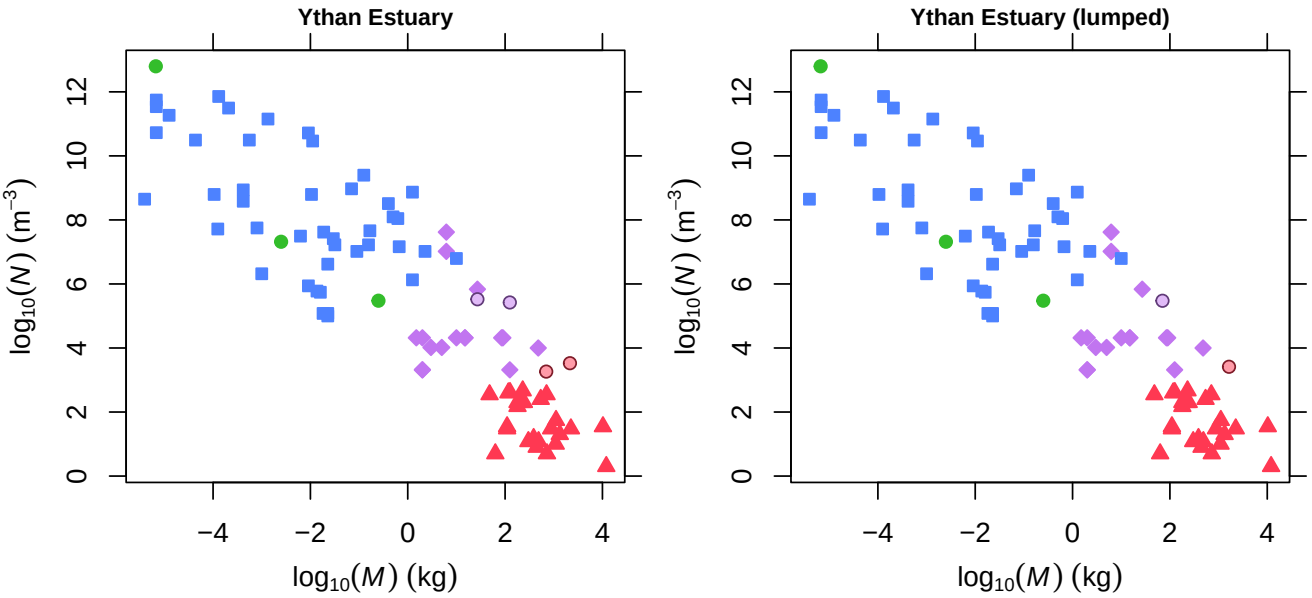
```
[1] 92
```

```
> NumberOfNodes(YthanEstuary.lumped)  # 90
```

```
[1] 90
```

Graphically compare the two communities.

```
> # Plot the original and lumped communities
> par(mfrow=c(1,2))
> plot(YthanEstuary, highlight.nodes=c("Platichthys flesus",
                                       "Platichthys flesus (juvenile)",
                                       "Somateria mollissima",
                                       "Somateria mollissima (juvenile)"),
      show.web=FALSE)
> plot(YthanEstuary.lumped, highlight.nodes=c("Platichthys flesus",
                                              "Somateria mollissima"),
      show.web=FALSE)
```



The default behaviour of `LumpNodes` and `LumpTrophicSpecies` is to aggregate numeric node properties by computing the N-weighted mean.

```
> NPS(YthanEstuary.lumped)["Platichthys flesus", c('M','N')]
```

```
                          M      N
Platichthys flesus 70.36585 298480
```

```
> # These values were computed as follows
> nps <- NPS(YthanEstuary)
> M <- nps[c("Platichthys flesus", "Platichthys flesus (juvenile)"), 'M']
> N <- nps[c("Platichthys flesus", "Platichthys flesus (juvenile)"), 'N']
> # Arithmetic mean of N
> mean(N)
```

```
[1] 298480
```

```
> # N-weighted mean of M
> weighted.mean(M, N)

[1] 70.36585
```

The 'weight.by' parameter controls this behaviour:

```
> YthanEstuary.lumped2 <- LumpNodes(YthanEstuary, lump, weight.by=NULL)
> NPS(YthanEstuary.lumped2)["Platichthys flesus", c('M','N')]

                     M      N
Platichthys flesus 76 298480

> # Computed as the arithmetic means of M and N
> mean(M)

[1] 76

> mean(N)

[1] 298480

>
```

References

F. Briand and J.E. Cohen. Community food webs have scale-invariant structure. *Nature*, 307(5948): 264–267, 1984. doi: 10.1038/307264a0.

S.R. Carpenter and J.F. Kitchell. *The trophic cascade in lakes.* Cambridge University Press, 1996.

J.E. Cohen, T. Jonsson, and S.R. Carpenter. Ecological community description using the food web, species abundance, and body size. *Proceedings of the National Academy of Sciences of the United States of America*, 100(4):1781–1786, 2003. doi: 10.1073/pnas.232715699.

C. Digel, J.O. Riede, and U. Brose. Body sizes, cumulative and allometric degree distributions across natural food webs. *Oikos*, 120(4):22335–22340, 2011. doi: 10.1111/j.1600-0706.2010.18862.x.

M.C. Emmerson and D. Raffaelli. Predator-prey body size, interaction strength and the stability of a real food web. *Journal of Animal Ecology*, 73(3):399–409, 2004. doi: 10.1111/j.0021-8790.2004.00818.x.

S.J. Hall and D. Raffaelli. Food-web patterns: lessons from a species-rich web. *Journal of Animal Ecology*, 60(3):823–841, 1991. doi: 10.2307/5416.

U. Jacob, A. Thierry, U. Brose, W.E. Arntz, S. Berg, T. Brey, I. Fetzer, T. Jonsson, K. Mintenbeck, C. Möllmann, et al. The role of body size in complex food webs: A cold case. *Advances In Ecological Research*, 45:181–223, 2011. doi: 10.1016/B978-0-12-386475-8.00005-8.

T. Jonsson, J.E. Cohen, and S.R. Carpenter. Food webs, body size, and species abundance in ecological community description. *Advances In Ecological Research*, 36:1–84, 2005. doi: 10.1016/S0065-2504(05)36001-6.

K. Layer, J.O. Riede, A.G. Hildrew, and G. Woodward. Food web structure and stability in 20 streams across a wide ph gradient. *Advances In Ecological Research*, 42:265–299, 2010. doi: 10.1016/S0065-2504(10)42005-X.

S. Levine. Several measures of trophic structure applicable to complex food webs. *Journal of Theoretical Biology*, 83(2):195–207, 1980. doi: 10.1016/0022-5193(80)90288-X.

S.B. Otto, B.C. Rall, and U. Brose. Allometric degree distributions facilitate food-web stability. *Nature*, 450(7173):1226–1229, 2007. doi: 10.1038/nature06359.

S.L. Pimm, J.H. Lawton, and J.E. Cohen. Food web patterns and their consequences. *Nature*, 350(6320): 669–674, 1991. doi: 10.1038/350669a0.

D.C. Reuman and J.E. Cohen. Estimating relative energy fluxes using the food web, species abundance, and body size. *Advances In Ecological Research*, 36:137–182, 2005.

D.B. Stouffer, J. Camacho, and L.A.N. Amaral. A robust measure of food web intervality. *Proceedings of the National Academy of Sciences of the United States of America*, 103(50):19015–19020, 2006. doi: 10.1073/pnas.0603844103.

D.B. Stouffer, E.L. Rezende, and L.A.N. Amaral. The role of body mass in diet contiguity and food-web structure. *Journal of Animal Ecology*, 80(3):632–639, 2011. doi: 10.1111/j.1365-2656.2011.01812.x.

P.H. Warren. Spatial and temporal variation in the structure of a freshwater food web. *Oikos*, 55(3): 299–311, 1989. doi: 10.2307/3565588.

R.J. Williams and N.D. Martinez. Simple rules yield complex food webs. *Nature*, 404(6774):180–183, 2000. doi: 10.1038/35004572.

R.J. Williams and N.D. Martinez. Limits to trophic levels and omnivory in complex food webs: theory and data. *The American Naturalist*, 163(3):458–468, 2004. doi: 10.1086/381964.

G. Woodward, D.C. Speirs, and A.G. Hildrew. Quantification and resolution of a complex, size-structured food web. *Advances In Ecological Research*, 36:85–135, 2005. doi: 10.1016/S0065-2504(05)36002-8.

P. Yodzis. Local trophodynamics and the interaction of marine mammals and fisheries in the benguela ecosystem. *Journal of Animal Ecology*, 67(4):635–658, 1998. doi: 10.1046/j.1365-2656.1998.00224.x.

A.E. Zook, A. Eklof, U. Jacob, and S. Allesina. Food webs: Ordering species according to body size yields high degree of intervality. *Journal of Theoretical Biology*, 271(1):106–113, 2011. doi: 10.1016/j.jtbi.2010.11.045.