

Cheddar plots and statistics (0.1-622)

Lawrence Hudson

2013-03-19

Contents

1	Introduction	2
2	Plotting node properties	3
2.1	PlotNPS	3
2.2	Rank of properties	11
2.3	Distribution of a property	14
2.4	Colours and symbols	15
2.5	Highlights and lowlights	22
2.6	Nodes that lack properties	25
3	Plotting trophic-link properties	29
3.1	PlotTLPS	29
3.2	Colours and symbols	32
4	Values by class	34
5	$\log_{10}(N)$-versus-$\log_{10}(M)$ statistics	35
6	Tri-trophic statistics	38
7	Examples through recreating figures and tables from published articles	39
7.1	Jonsson et al. (2005)	39
7.1.1	Jonsson et al. (2005), Fig. 3 (p 30)	39
7.1.2	Jonsson et al. (2005), Fig. 4 (p 33)	40
7.1.3	Jonsson et al. (2005), Fig. 5 (p 37)	41
7.1.4	Jonsson et al. (2005), Fig. 7 (p 47)	42
7.1.5	Jonsson et al. (2005), Fig. 8 (p 49)	43
7.1.6	Jonsson et al. (2005), Fig. 10 (p 57)	44
7.1.7	Jonsson et al. (2005), Fig. 11 (p 60)	45
7.1.8	Jonsson et al. (2005), Fig. 12 (p 61)	46
7.2	Layer et al. (2010)	47
7.2.1	Layer et al. (2010), Fig. 6 (p 282)	47
7.3	Woodward et al. (2005)	48
7.3.1	Woodward et al. (2005), Fig. 4. (p 108)	48
7.4	Cohen et al. (2009)	49
7.4.1	Cohen et al. (2009), Table 1 (p 22338)	49

7.4.2	Cohen et al. (2009), Table S3 (p 8)	50
7.4.3	Cohen et al. (2009), Fig. 1, (p 22336)	51

1 Introduction

This vignette describes Cheddar’s flexible system for plotting properties of ecological communities. You should read the ‘Community’ vignette before reading this one. Cheddar’s plotting system is built upon two general-purpose functions: `PlotNPS` (for **Plot Node PropertieS**) and `PlotTLPS` (**Plot Trophic-Link PropertieS**), which allow a range of properties, first-class and computed, Cheddar-provided and user-defined, to be plotted in a variety of ways. `PlotNPS` and `PlotTLPS` are used by a large number of convenience ‘wrapper’ functions that provide different views of a community, focussing on food web data that are augmented with estimates of body mass, M , and/or numerical abundance, N , following Cohen et al. (2003) (Table 1). You can use whatever mixture of `PlotNPS`, `PlotTLPS` and the functions in Table

Trophic links	M	N	Plot function	Description
	✓		<code>PlotMvRankM</code>	$\log_{10}(M)$ versus $\text{rank}(M)$
	✓		<code>PlotMDistribution</code>	Distribution of $\log_{10}(M)$
		✓	<code>PlotNvRankN</code>	$\log_{10}(N)$ versus $\text{rank}(N)$
		✓	<code>PlotNDistribution</code>	Distribution of $\log_{10}(N)$
	✓	✓	<code>PlotBvRankB</code>	$\log_{10}(B)$ versus $\text{rank}(B)$
	✓	✓	<code>PlotBDistribution</code>	Distribution of $\log_{10}(B)$
	✓	✓	<code>PlotNSpectrum</code>	Approximate numerical abundance spectrum
	✓	✓	<code>PlotNvM *</code>	$\log_{10}(N)$ versus $\log_{10}(M)$
	✓	✓	<code>PlotBSpectrum</code>	Approximate biomass abundance spectrum
	✓	✓	<code>PlotBvM *</code>	$\log_{10}(B)$ versus $\log_{10}(M)$
✓			<code>PlotPredationMatrix</code>	Predation matrix
✓			<code>PlotWebByLevel</code>	Food web plotted vertically by trophic level
✓			<code>PlotDegreeDistribution</code>	Histogram of node degree
✓			<code>PlotCircularWeb</code>	An overview of complexity
✓	✓		<code>PlotMRvMC *</code>	$\log_{10}(M)$ of resource versus $\log_{10}(M)$ of consumer
✓		✓	<code>PlotNRvNC *</code>	$\log_{10}(N)$ of resource versus $\log_{10}(N)$ of consumer
✓		✓	<code>PlotNPyramid</code>	$\log_{10}(\sum(N))$ by trophic level
✓	✓	✓	<code>PlotBRvBC *</code>	$\log_{10}(B)$ of resource versus $\log_{10}(B)$ of consumer
✓	✓	✓	<code>PlotBPyramid</code>	$\log_{10}(\sum(B))$ by trophic level
✓	✓	✓	<code>PlotAlowervAupper</code>	Upper link-angle against lower link angle

Table 1: Cheddar’s high-level plot functions.

1 that meets your needs and fits your data. The examples below use the datasets from the two pelagic epilimnion communities of Tuesday Lake, Michigan, USA (Carpenter and Kitchell, 1996; Cohen et al., 2003; Jonsson et al., 2005) and the community of Ythan Estuary in north Scotland (Hall and Raffaelli, 1991; Emmerson and Raffaelli, 2004).

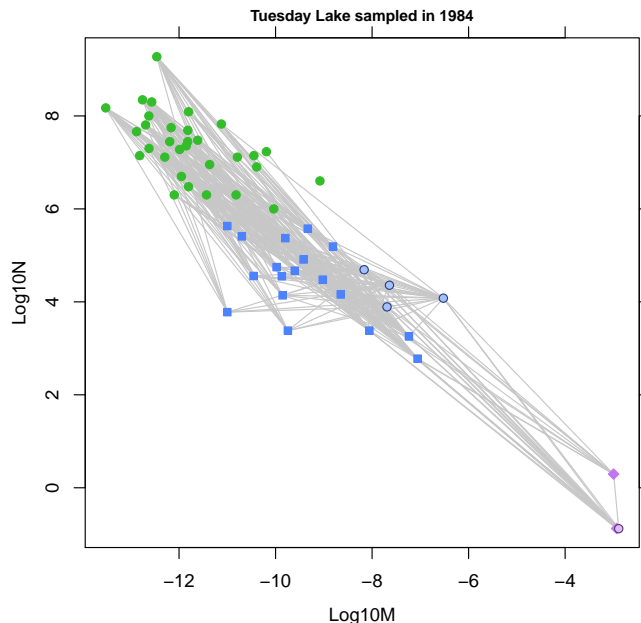
2 Plotting node properties

2.1 PlotNPS

`PlotNPS` plots one node property against another. Every Cheddar function that plots one point per node delegates the job of plotting to `PlotNPS` and all of its power and flexibility are available to all of the functions discussed in this section. `PlotNPS` function accepts the names of properties to plot on the x and y axes. These can be either ‘first-class’ properties, such as body mass, M , and numerical abundance, N , or they can be the names of functions that compute and return node properties.

The relationship between species-level log-transformed N and M has implications for community- (Reuman et al., 2008) and ecosystem-level (Brown et al., 2004) theories. We can plot $\log_{10}(N)$ against $\log_{10}(M)$ by using `PlotNPS` together with the `Log10M` and `Log10N` functions.

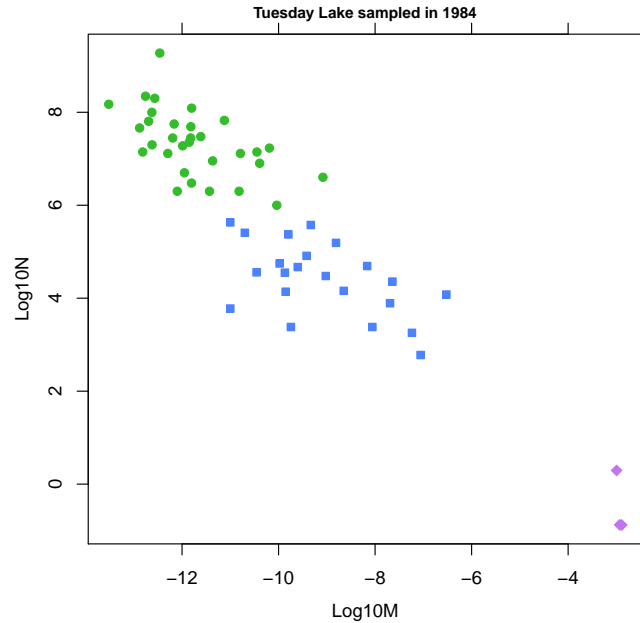
```
> data(TL84)
> PlotNPS(TL84, 'Log10M', 'Log10N')
```



`PlotNPS` uses the ‘category’ node property, if present, to decide plotting colours and symbols. Producers are shown by green circles, invertebrates by blue squares and vertebrate ectotherms by purple diamonds. Many of Cheddar’s plotting and 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. This is explored more in Section 2.4.

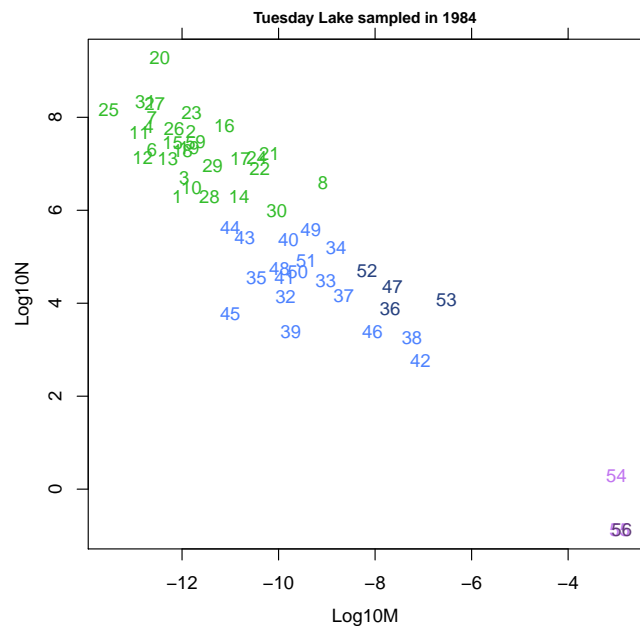
The TL84 dataset includes trophic links so `PlotNPS` will show the food web and highlights cannibals by plotting them in a light-coloured filled circle with a dark border, as shown in the previous example. We can turn off these behaviours.

```
> PlotNPS(TL84, 'Log10M', 'Log10N', show.web=FALSE, highlight.nodes=NULL)
```



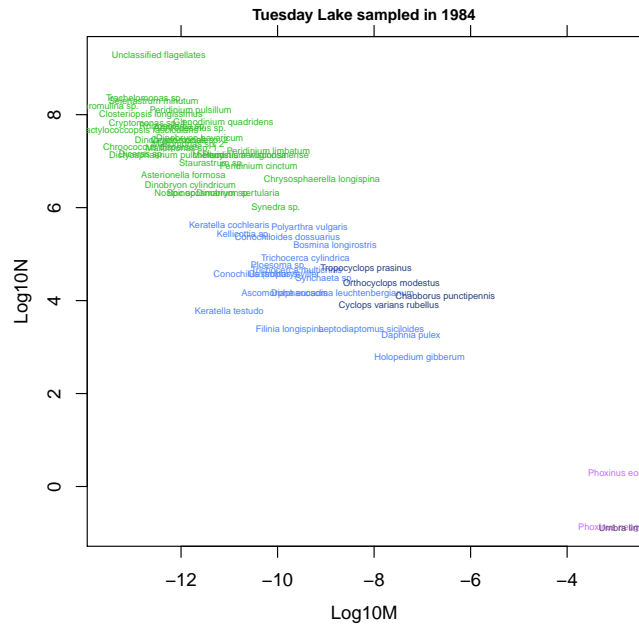
We can plots labels rather than points. When plotting labels, nodes given by 'highlight.nodes' (by default, those nodes that are cannibals) are shown in a darker colour.

```
> PlotNPS(TL84, 'Log10M', 'Log10N', show.nodes.as='labels', show.web=FALSE)
```



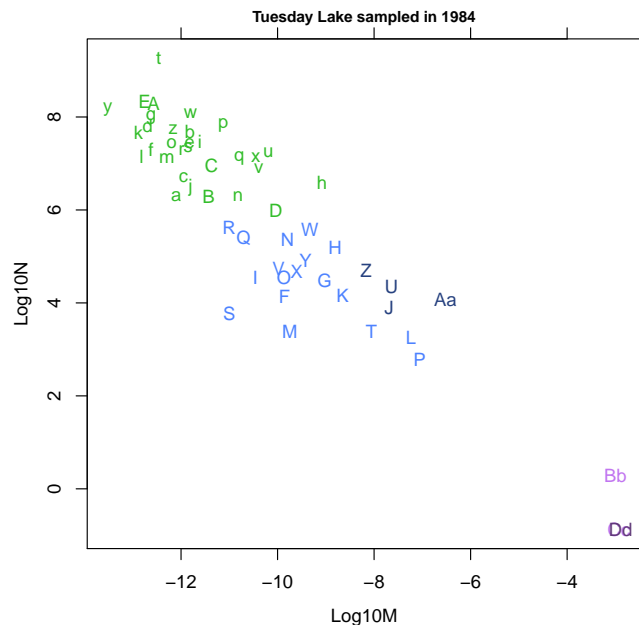
In the above example, labels are taken from the order of the nodes in the TL84 community. We can plot a different label by setting the 'node.labels' property. The example below plots node names.

```
> PlotNPS(TL84, 'Log10M', 'Log10N', show.nodes.as='labels', show.web=FALSE,
  node.labels='node', cex=0.5)
```



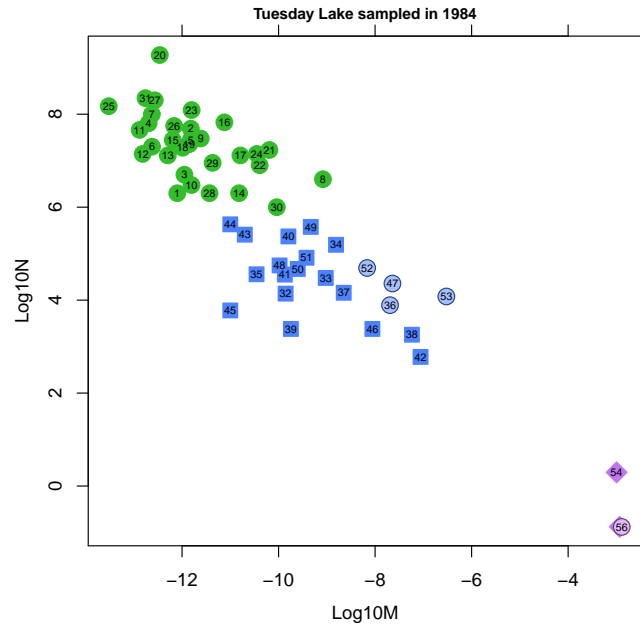
The parameter node.labels also accepts a vector of labels, so we could use letters to label the nodes.

```
> lots.of.letters <- c(letters, LETTERS, paste(LETTERS,letters,sep=''))
> PlotNPS(TL84, 'Log10M', 'Log10N', show.nodes.as='labels', show.web=FALSE,
  node.labels=lots.of.letters[1:NumberOfNodes(TL84)])
```



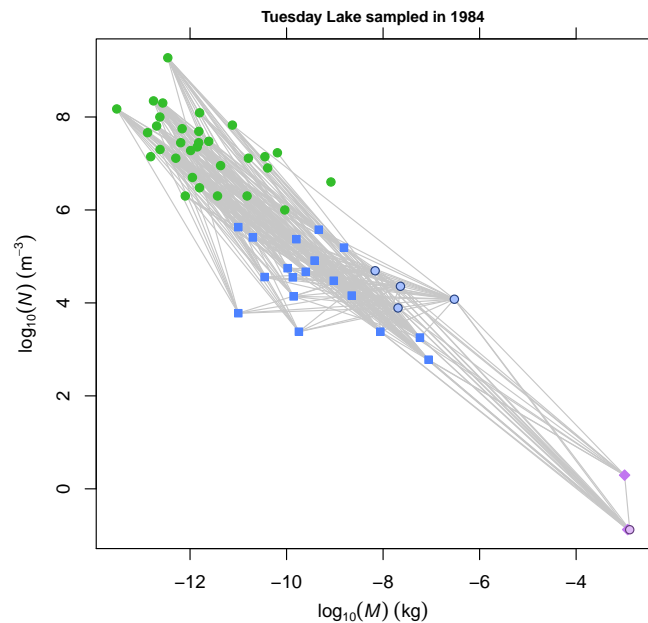
PlotNPS can plot both symbols and labels.

```
> PlotNPS(TL84, 'Log10M', 'Log10N', show.nodes.as='both', show.web=FALSE, cex=2)
```



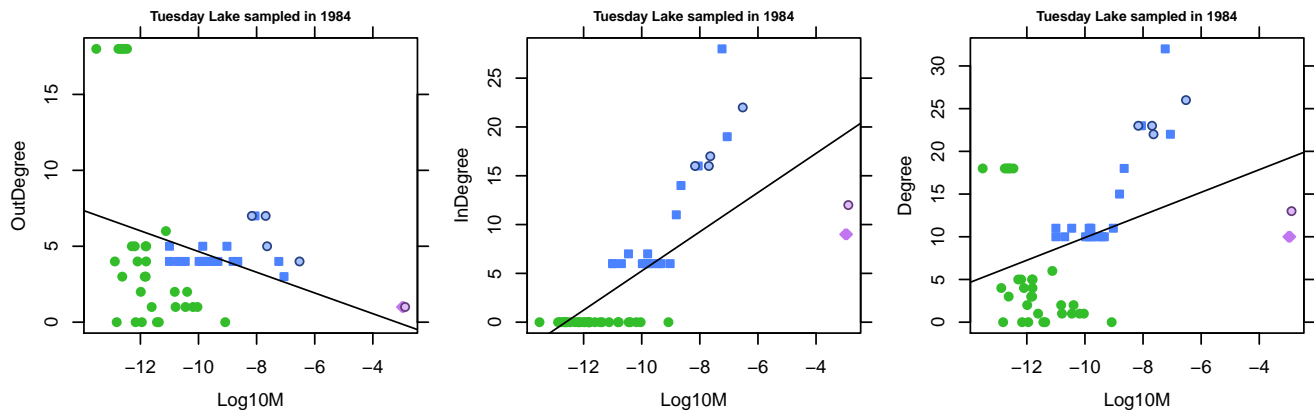
When `show.nodes.as` is 'both', the size and colours of labels are governed by `label.cex` and `label.colour`, the latter discussed further in Section 2.4. The axes titles in the above examples are taken from the names of the properties. Cheddar provides some functions for producing nicely-formatted labels, shown below.

```
> PlotNPS(TL84, 'Log10M', 'Log10N', xlab=Log10MLabel(TL84),
  ylab=Log10NLabel(TL84))
```



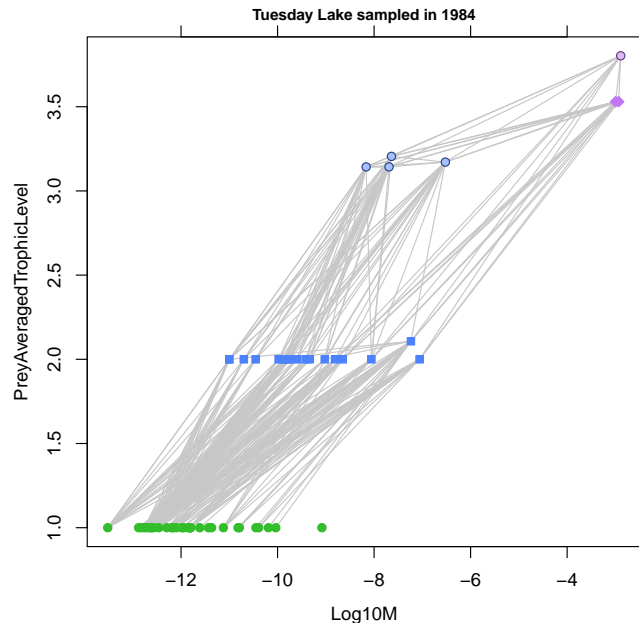
PlotNPS can plot any computed node property. For example, we can examine allometric degree distributions (Jonsson et al., 2005; Otto et al., 2007; Digel et al., 2011), which describe how species' numbers of trophic links scale with their log-transformed body masses.

```
> par(mfrow=c(1,3))
> PlotNPS(TL84, 'Log10M', 'OutDegree', show.web=FALSE)
> abline(lm(OutDegree(TL84) ~ Log10M(TL84)))
> PlotNPS(TL84, 'Log10M', 'InDegree', show.web=FALSE)
> abline(lm(InDegree(TL84) ~ Log10M(TL84)))
> PlotNPS(TL84, 'Log10M', 'Degree', show.web=FALSE)
> abline(lm(Degree(TL84) ~ Log10M(TL84)))
```



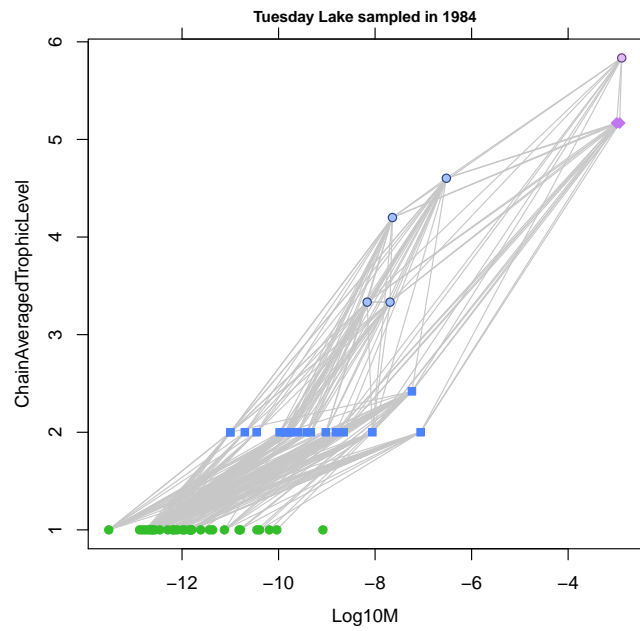
To examine the relationship between trophic structure and body mass (Cohen et al., 2003; Jonsson et al., 2005; Gilljam et al., 2011; Jacob et al., 2011; Riede et al., 2011; de Visser et al., 2011) we can plot trophic level against log-transformed body mass. The `PreyAveragedTrophicLevel` uses the matrix-inversion method of Levine (1980) to compute trophic levels.

```
> PlotNPS(TL84, 'Log10M', 'PreyAveragedTrophicLevel')
```



An alternative measure of trophic level is provided by `ChainAveragedTrophicLevel`, which computes the mean position of each node in every chain in the food web.

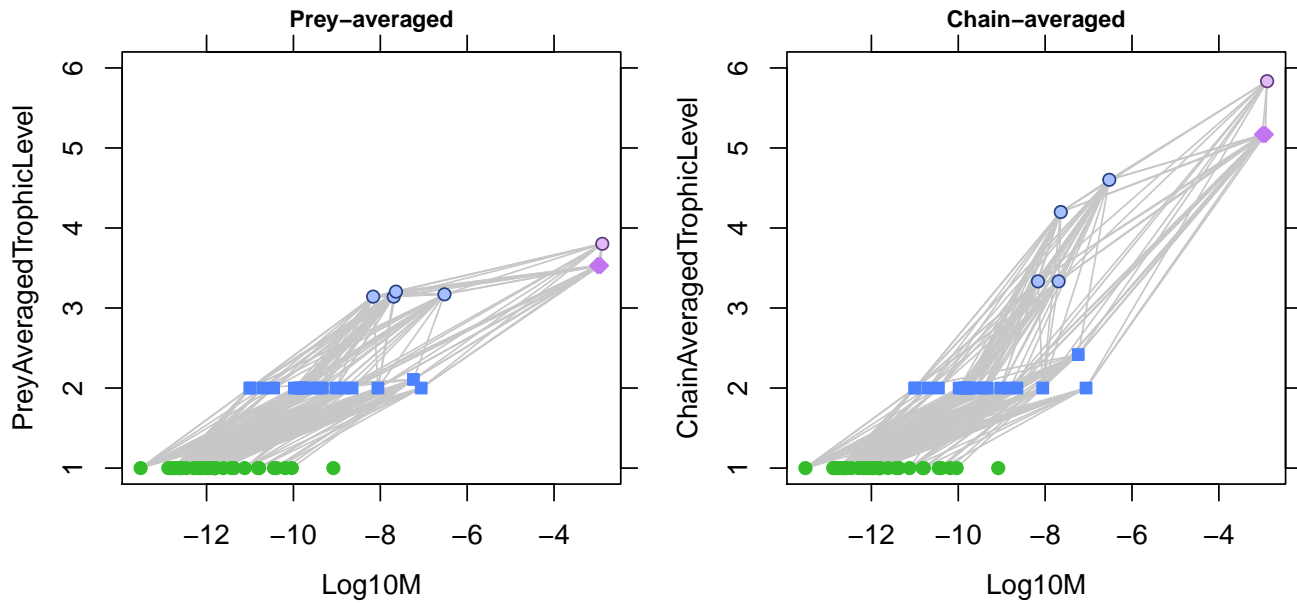
```
> PlotNPS(TL84, 'Log10M', 'ChainAveragedTrophicLevel')
```



Cheddar offers the six different measures of trophic level described by Williams and Martinez (2004); see the ‘Communities’ vignette and the help page for `TrophicLevels` for details.

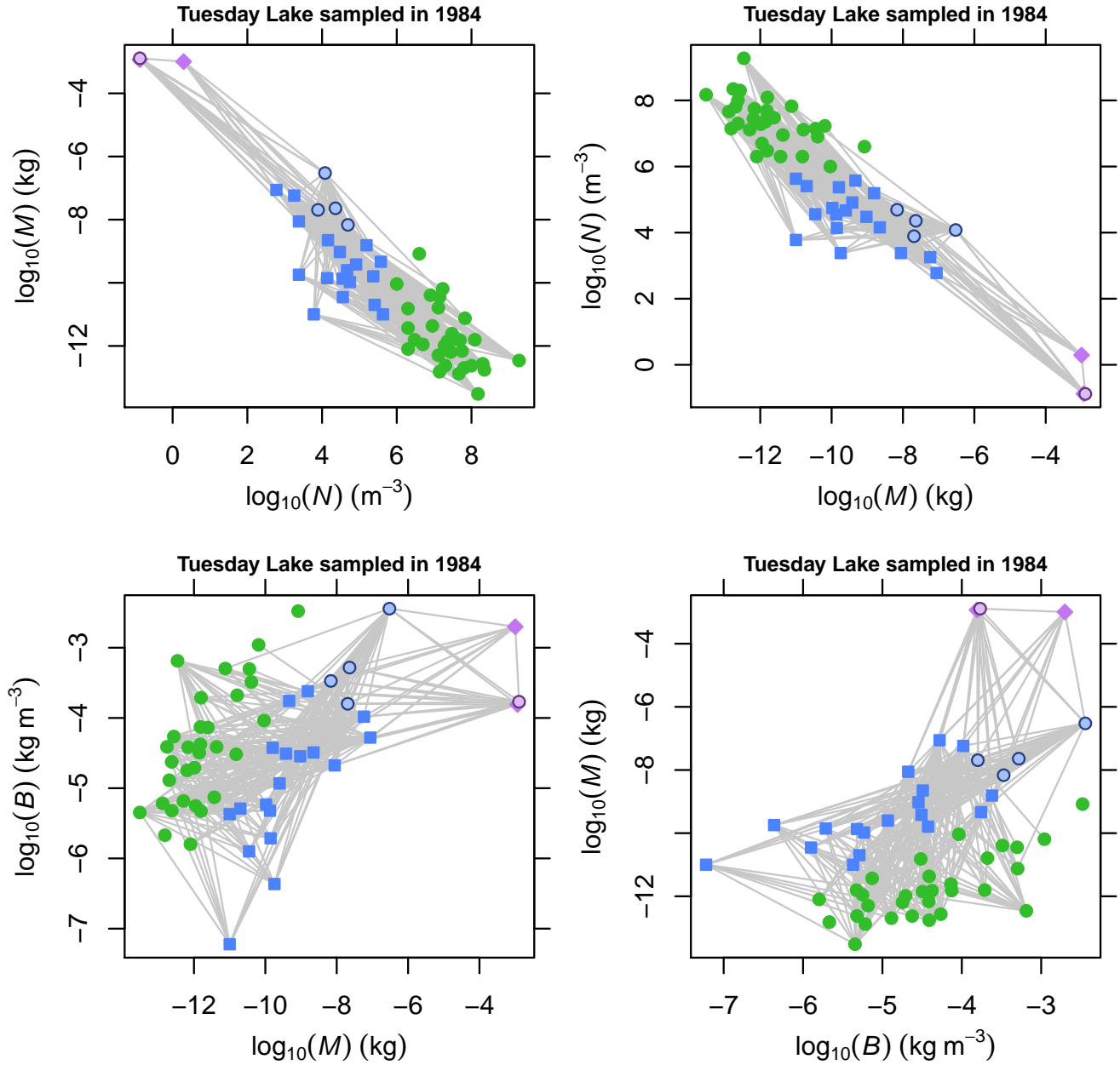
PlotNPS accepts the range of graphical parameters provided by R's plotting system, such as `ylim` and `main`. To directly compare prey-averaged and chain-averaged trophic level we can produce side-by-side plots with the same y-axis limits.

```
> par(mfrow=c(1,2))
> PlotNPS(TL84, 'Log10M', 'PreyAveragedTrophicLevel', ylim=c(1, 6),
  main='Prey-averaged')
> PlotNPS(TL84, 'Log10M', 'ChainAveragedTrophicLevel', ylim=c(1, 6),
  main='Chain-averaged')
```



It is common to plot some combination of \log_{10} -transformed body mass, M , numerical abundance, N , or biomass, B (e.g. Cohen et al., 2003; Jonsson et al., 2005; Woodward et al., 2005). The convenience functions `PlotMvN`, `PlotNvM`, `PlotBvM` and `PlotMvB` are ‘wrappers’ around `PlotNPS` that create these plots.

```
> par(mfrow=c(2,2))
> PlotMvN(TL84)
> PlotNvM(TL84)
> PlotBvM(TL84)
> PlotMvB(TL84)
```

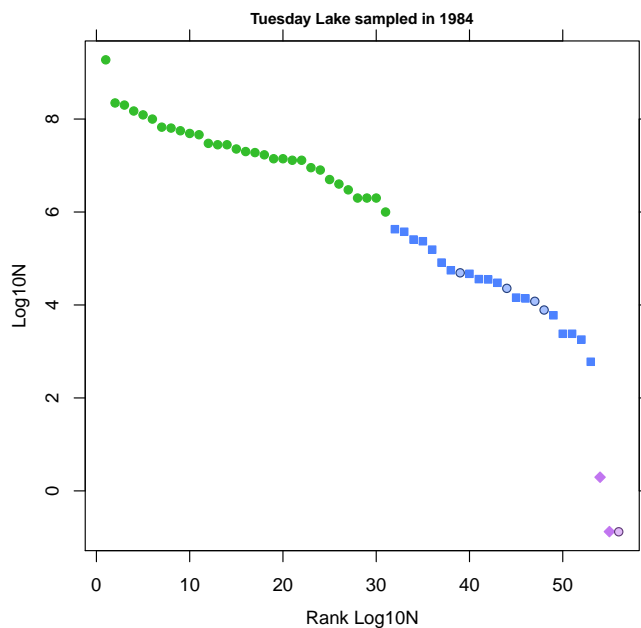


All of the modifying parameters that can be sent to `PlotNPS` to do things like changing the highlighting etc., can also be sent to to these convenience functions.

2.2 Rank of properties

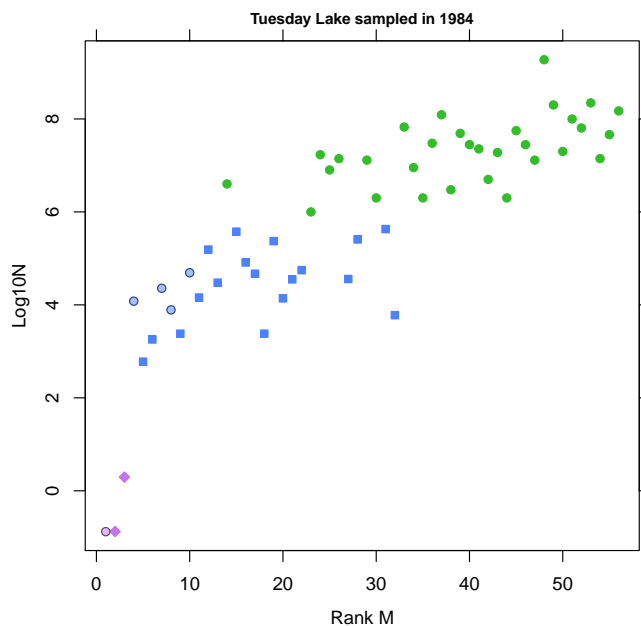
`PlotRankNPS` plots the rank of a property. As with `PlotNPS`, properties can be either first-class or computed. To plot $\log_{10}(N)$ against the rank of N (Jonsson et al., 2005):

```
> PlotRankNPS(TL84, 'Log10N')
```



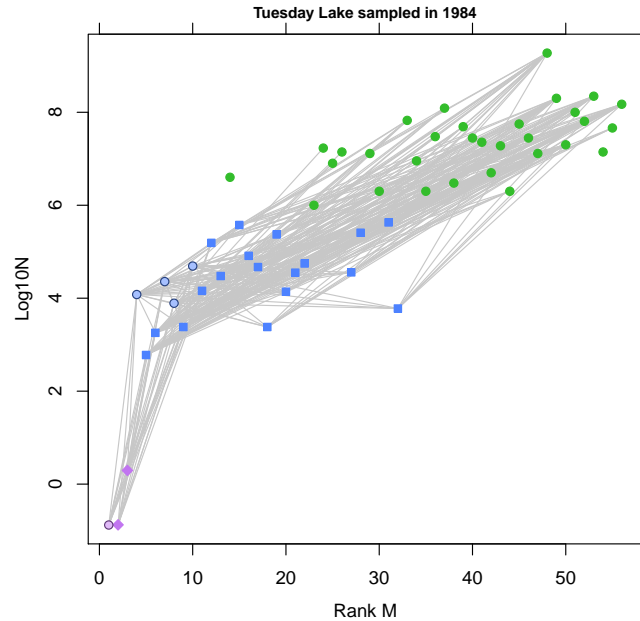
To plot $\log_{10}(N)$ against the rank of M :

```
> PlotRankNPS(TL84, 'Log10N', rank.by='M')
```



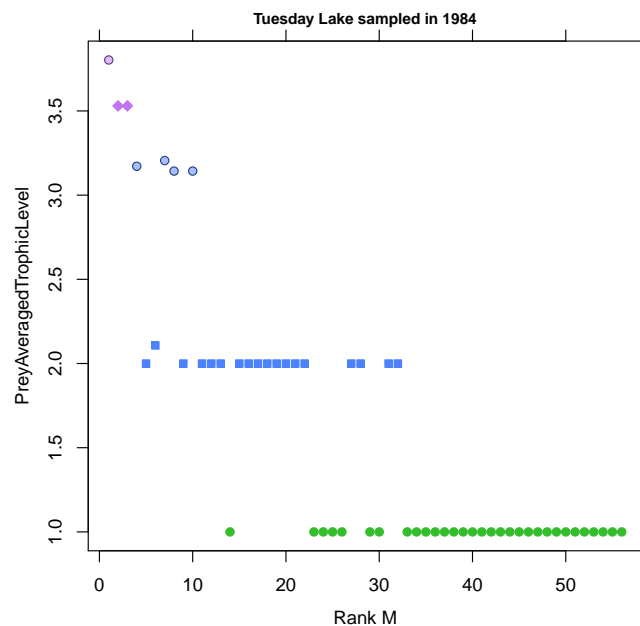
PlotRankNPS uses PlotNPS to do the actual plotting so all of PlotNPS's power and parameters are available. For example, the default behaviour of PlotRankNPS is to not show the food web. We can use PlotNPS's 'show.web' parameter to show the food web.

```
> PlotRankNPS(TL84, 'Log10N', rank.by='M', show.web=TRUE)
```



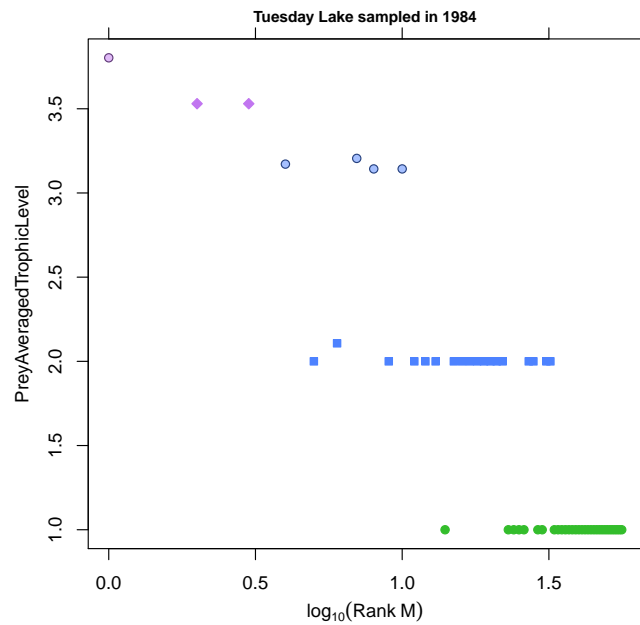
To plot trophic level against rank of M (Jonsson et al., 2005):

```
> PlotRankNPS(TL84, 'PreyAveragedTrophicLevel', rank.by='M')
```



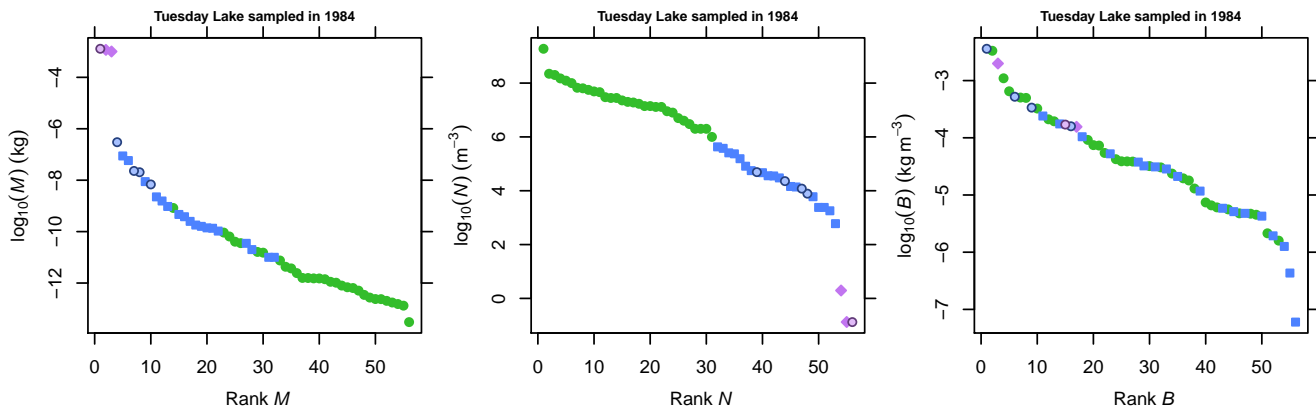
The `PlotRankNPS` function can log-transform the rank values shown on the x axis (Jonsson et al., 2005):

```
> PlotRankNPS(TL84, 'PreyAveragedTrophicLevel', rank.by='M', log10.rank=TRUE)
```



The convenience functions `PlotMvRankM`, `PlotNvRankN` and `PlotBvRankB` are ‘wrappers’ around `PlotRankNPS` that plot rank \log_{10} -transformed body mass, M , numerical abundance, N , or biomass, B (Jonsson et al., 2005).

```
> par(mfrow=c(1,3))
> PlotMvRankM(TL84)
> PlotNvRankN(TL84)
> PlotBvRankB(TL84)
```

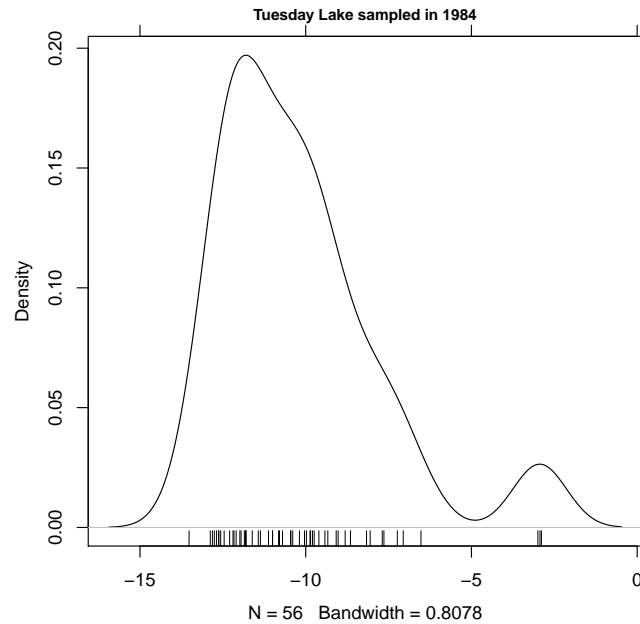


All of the modifying parameters that can be sent to `PlotNPS` to do things like changing the highlighting etc., can also be sent to these convenience functions.

2.3 Distribution of a property

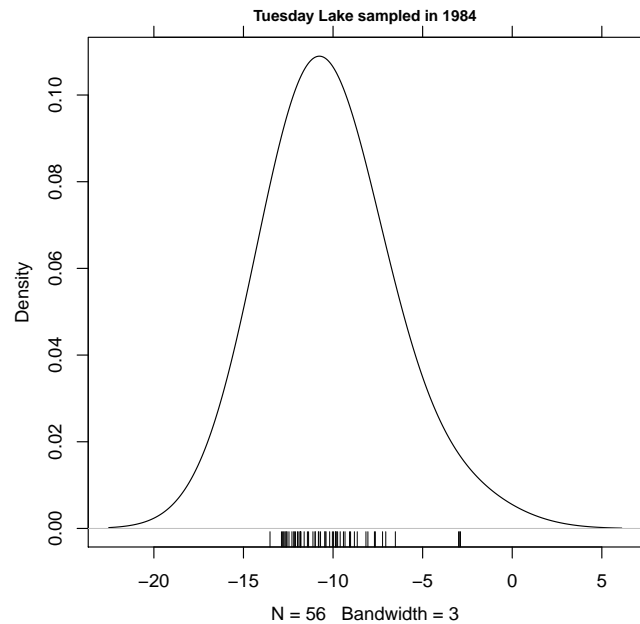
The `PlotNPSDistribution` function plots the distribution of a node property.

```
> PlotNPSDistribution(TL84, 'Log10M')
```



You can pass arguments to R's `density` function using `density.args`, to change the bandwidth, for example:

```
> PlotNPSDistribution(TL84, 'Log10M', density.args=list(bw=3))
```

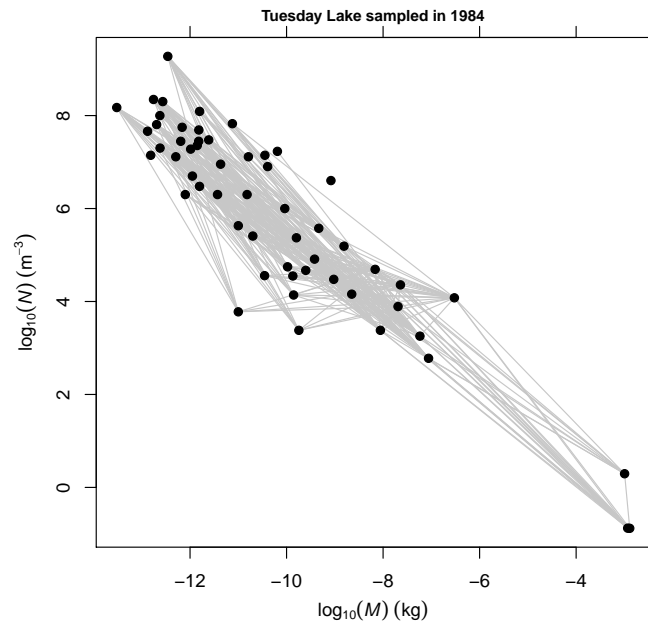


The convenience 'wrapper' functions `PlotMDistribution`, `PlotNDistribution` and `PlotBDistribution` plot \log_{10} -transformed body mass, numerical abundance and biomass abundance respectively.

2.4 Colours and symbols

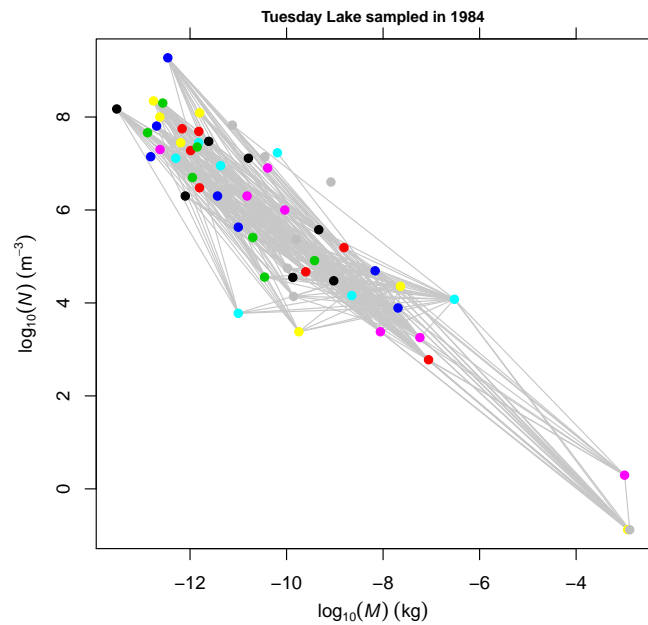
Cheddar builds on R's highly flexible plotting mechanisms. Symbol, outline colour, fill colour and symbol size can be controlled either by a first-class property, such as 'category' or taxonomic resolution, or by a computed property such as trophic species numbers. As with any R plot, you can directly set colours, symbols, a graph title etc. If the community being plotted has a node property called 'category' the function uses 'category' to decide plotting colours and symbols using the specifications given in `DefaultCategoryColours` and `DefaultCategorySymbols`.

```
> PlotNmM(TL84, col=1, pch=19, highlight.nodes=NULL)
```



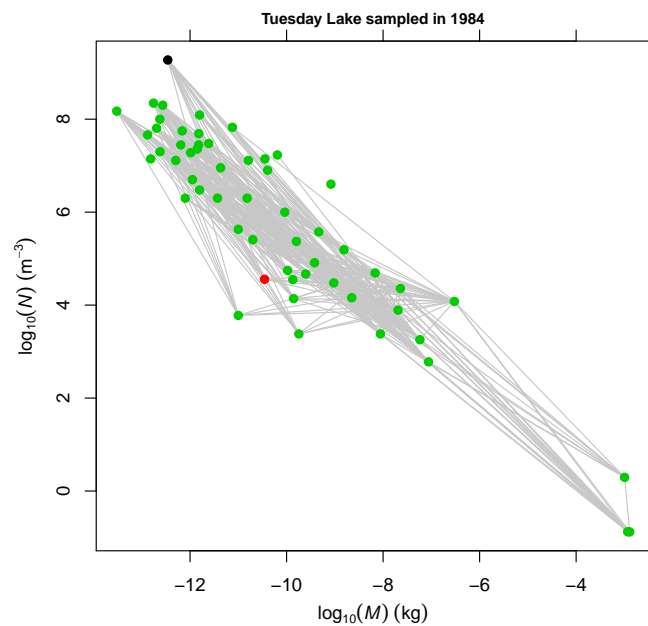
Plot each node in different colour by providing a vector of colours.

```
> PlotNmM(TL84, col=1:56, pch=19, highlight.nodes=NULL)
```



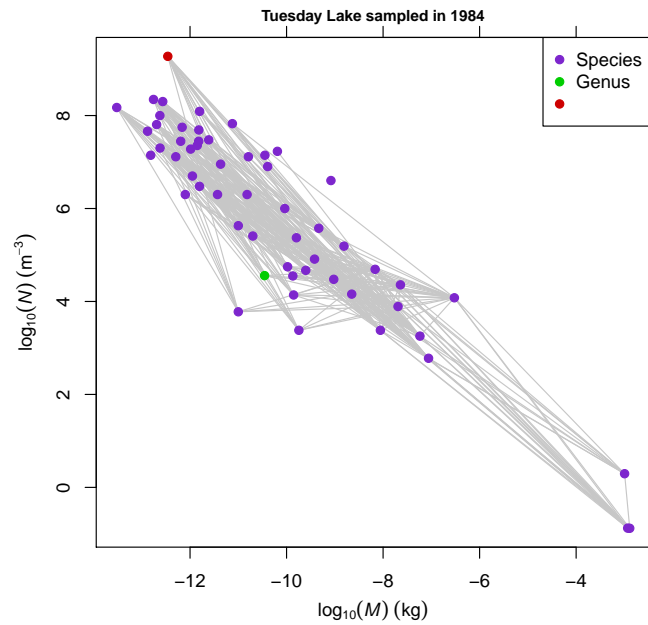
Plot each level of taxonomic resolution in a different colour. In this example, a value of 'colour.spec' is not given so PlotNPS converts the values of the first-class node property 'resolved.to' to a factor and uses the factor levels to colour each node. This is a 'quick and dirty' way to set node colours.

```
> PlotNmM(TL84, colour.by='resolved.to', pch=19, highlight.nodes=NULL)
```



We can improve on this by showing each level of taxonomic resolution in a specific colour by providing the 'colour.spec' parameter.

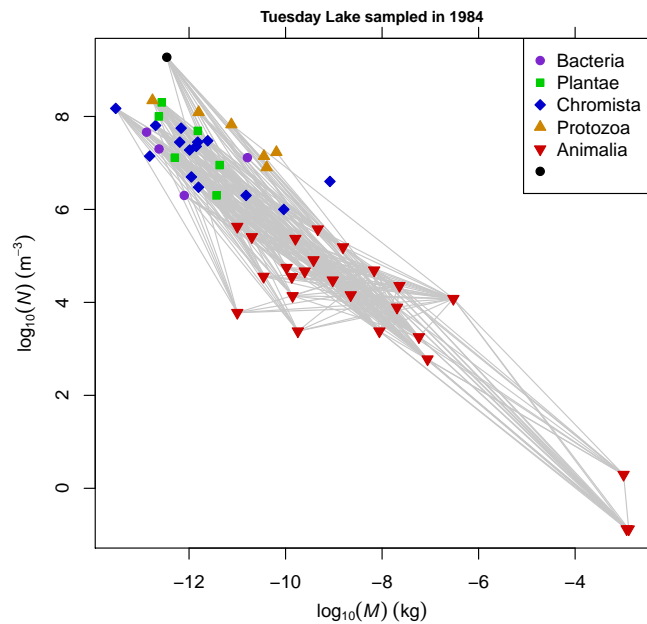
```
> colour.spec <- c(Species='purple3', Genus='green3', 'red3')
> PlotNmM(TL84, colour.by='resolved.to', colour.spec=colour.spec, pch=19,
  highlight.nodes=NULL)
> legend("topright", legend=names(colour.spec), pch=19, col=colour.spec)
```



The 'Unclassified flagellates' node does not have a taxonomic resolution. This node is matched by the unnamed value in colour.spec and so is shown in red.

The example below uses the ‘kingdom’ level of classification to determine node colours and symbols.

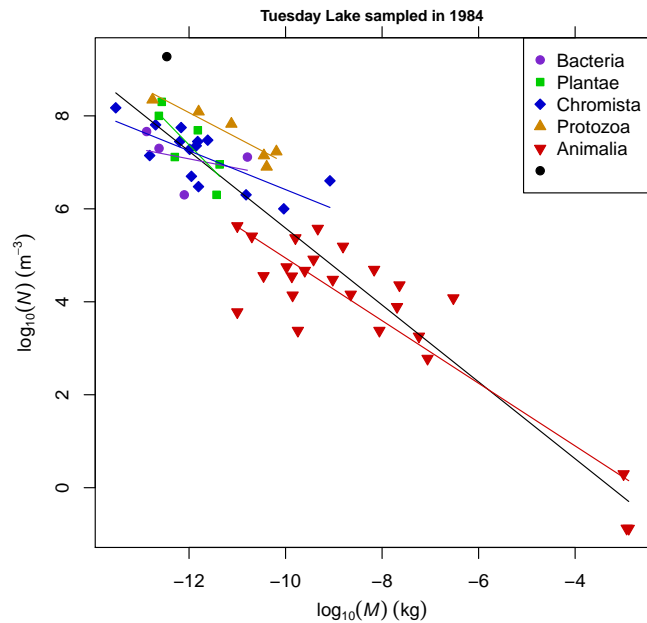
```
> symbol.spec = c(Bacteria=21, Plantae=22, Chromista=23,
                  Protozoa=24, Animalia=25, 19)
> colour.spec = c(Bacteria='purple3', Plantae='green3',
                  Chromista='blue3', Protozoa='orange3',
                  Animalia='red3', 'black')
> PlotNvM(TL84,
          symbol.by='kingdom', symbol.spec=symbol.spec,
          bg.by='kingdom', bg.spec=colour.spec,
          colour.by='kingdom', colour.spec=colour.spec,
          highlight.nodes=NULL)
> legend("topright", legend=names(colour.spec), pch=symbol.spec,
        col=colour.spec, pt.bg=colour.spec)
```



The ‘Unclassified flagellates’ node does not have a classification so is matched by the unnamed values in `symbol.spec` and `colour.spec` and is shown by a black circle.

We can fit and plot linear regressions for each kingdom by using the `NvMLinearRegressions` and `PlotLinearModels` functions. The food web is not shown in this plot to make the regression lines clearer.

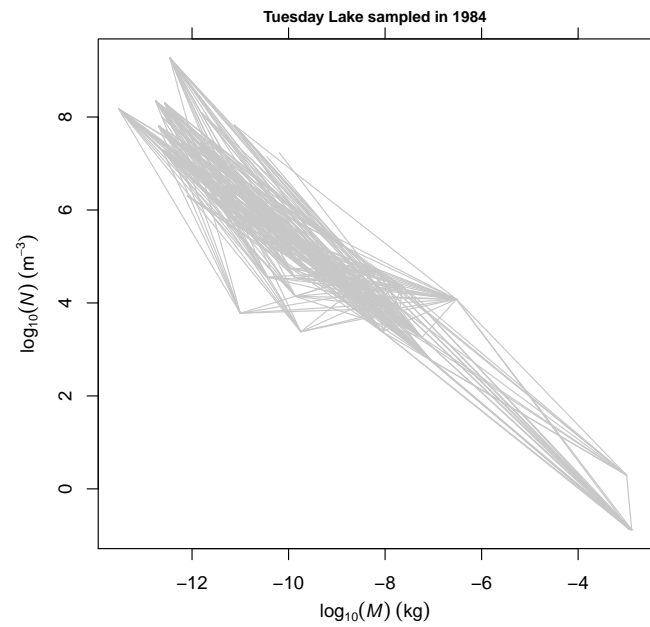
```
> symbol.spec = c(Bacteria=21, Plantae=22, Chromista=23,
                  Protozoa=24, Animalia=25, 19)
> colour.spec = c(Bacteria='purple3', Plantae='green3',
                  Chromista='blue3', Protozoa='orange3',
                  Animalia='red3', 'black')
> PlotNvM(TL84,
          symbol.by='kingdom', symbol.spec=symbol.spec,
          bg.by='kingdom', bg.spec=colour.spec,
          colour.by='kingdom', colour.spec=colour.spec,
          highlight.nodes=NULL, show.web=FALSE)
> legend("topright", legend=names(colour.spec), pch=symbol.spec,
        col=colour.spec, pt.bg=colour.spec)
> models <- NvMLinearRegressions(TL84, class='kingdom')
> colours <- PlotLinearModels(models, colour.spec=colour.spec)
```



The model fitted through all data points is matched by the empty value in `colour.spec` so is drawn in black. `NvMLinearRegressions` is described in more detail in Section 5.

You can use `pch=NA` to turn off symbols and just show the food web.

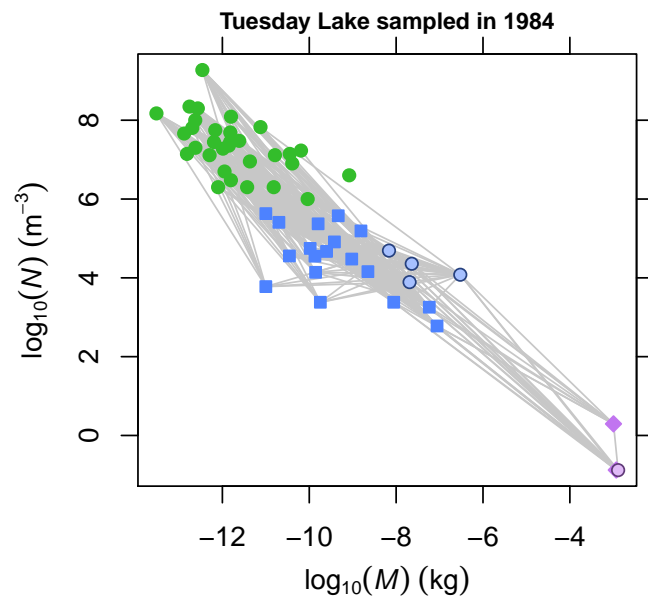
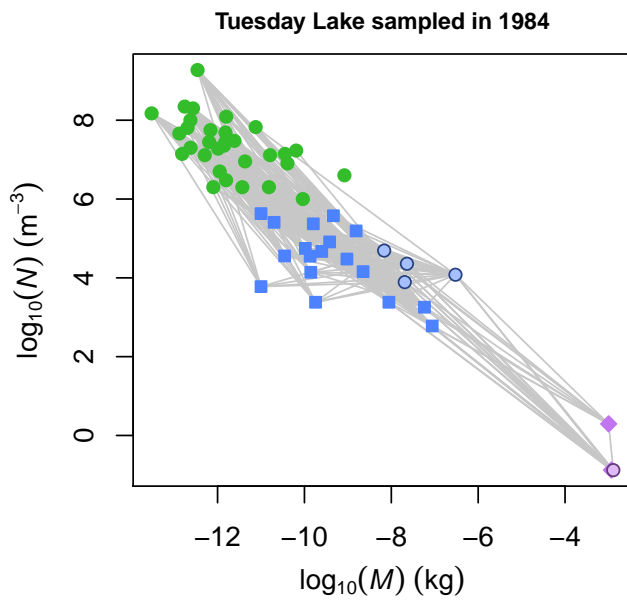
```
> PlotNmM(TL84, pch=NA, highlight.nodes=NULL)
```



See the help page for the R `par` function for more information about colours and symbols.

All of Cheddar's plot functions add axis ticks to the top and right of the plot by default. This can be turned off by using the `cheddarTopAndRightTicks` option.

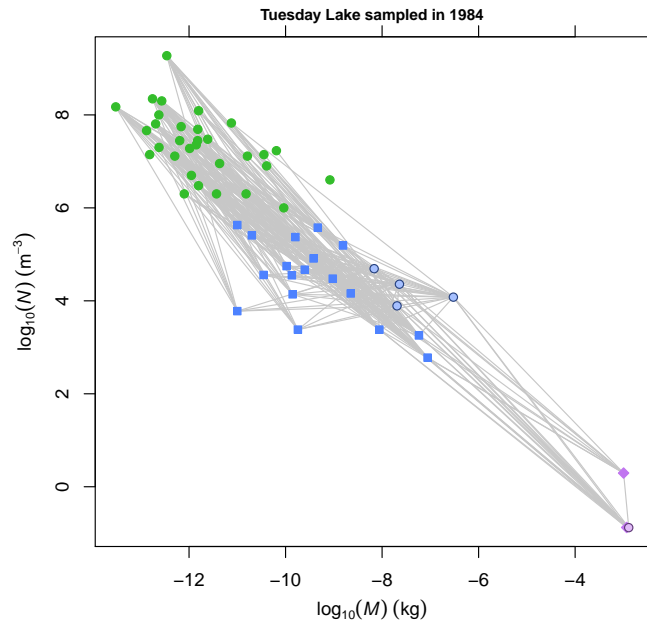
```
> par(mfrow=c(1,2))
> # Don't add ticks
> options(cheddarTopAndRightTicks=FALSE)
> PlotNvM(TL84)
> # Add ticks
> options(cheddarTopAndRightTicks=TRUE)
> PlotNvM(TL84)
```



2.5 Highlights and lowlights

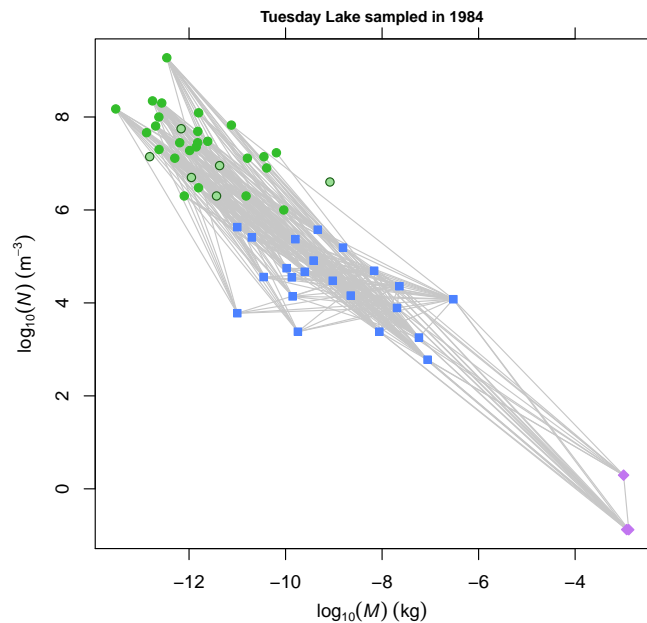
`PlotNPS` (and by extension all of the other node plotting functions) allow individual nodes to be highlighted or lowlighted using the `highlight.nodes` and `lowlight.nodes` parameters. These parameters accept functions. Functions should take a single `Community` object and return node names. Cheddar provides many such functions, such as `Cannibals`, which returns the names of all nodes that consume themselves.

```
> PlotNm(TL84, highlight.nodes=Cannibals)
```



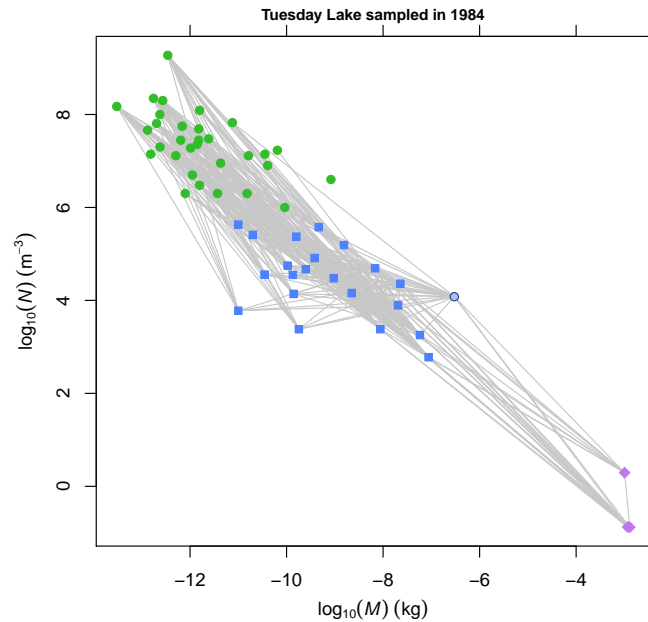
To highlight nodes without trophic-links to any other nodes we can use the `IsolatedNodes` function.

```
> PlotNm(TL84, highlight.nodes=IsolatedNodes)
```



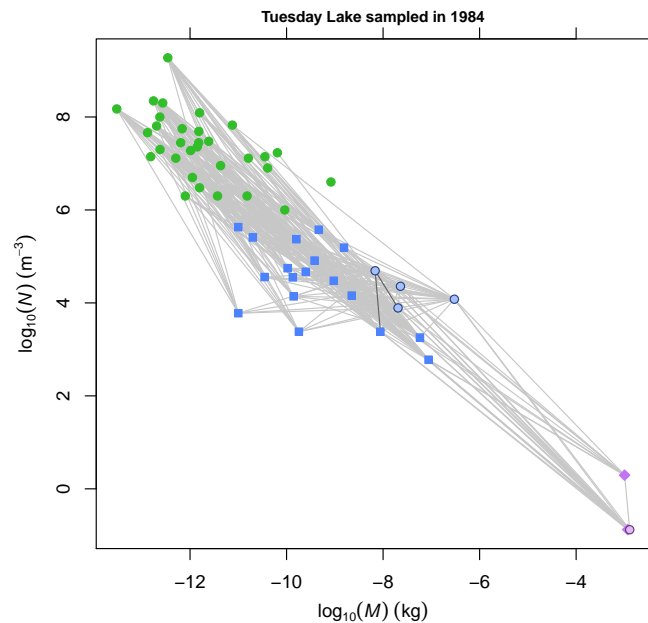
These parameters also accept the names or IDs of nodes. We can highlight an individual node.

```
> PlotNmM(TL84, highlight.nodes='Chaoborus punctipennis')
```



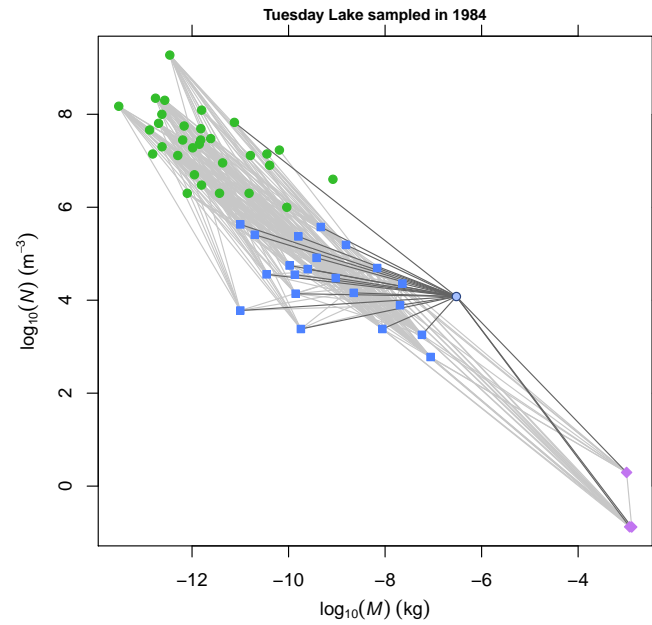
PlotNPS also allows trophic links to be highlighted. The `highlight.links` parameter accepts either a `data.frame` with the columns 'resource' and 'consumer' or a function that accepts a single community as its only parameters and returns such a `data.frame`. One such function is `ResourceLargerThanConsumer`, which returns a `data.frame` containing trophic links in which the resource has a larger body mass, M , than its consumer. To highlight these links we would simply pass this function as the `highlight.links` parameter.

```
> PlotNmM(TL84, highlight.links=ResourceLargerThanConsumer)
```



The example below uses the helper function `TrophicLinksForNodes` to highlight all links in to and out of *Chaoborus punctipennis*.

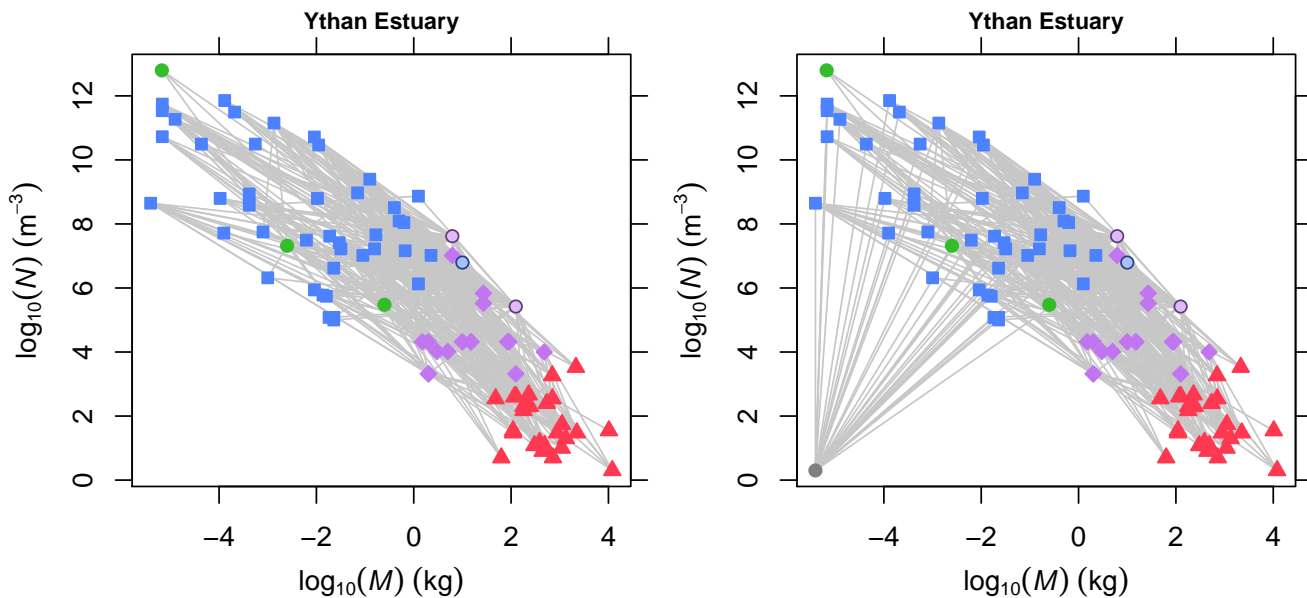
```
> PlotNmM(TL84, highlight.nodes='Chaoborus punctipennis',  
  highlight.links=TrophicLinksForNodes(TL84, 'Chaoborus punctipennis'))
```



2.6 Nodes that lack properties

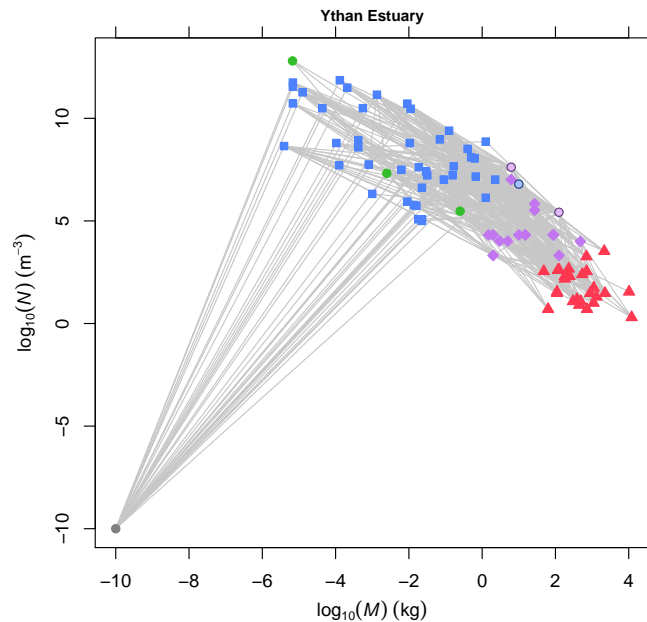
Communities are never completely sampled and some species may be represented by too few individuals for properties such as M and N to be accurately estimated. Other resources such as detritus, commonly eaten by primary consumers in many communities, have no clearly-defined M or N . Many datasets will therefore contain food-web nodes that lack one or more properties, for example the Ythan Estuary dataset has a large number of primary consumers that depend upon detritus, which has no M or N . `PlotNPS`, and hence the higher-level functions that use it, will place nodes without a property at the lowest extent of the relevant axis if the 'show.na' parameter is `TRUE`. By default, 'show.na' is `FALSE`. The second of the $\log_{10}(N)$ -versus- $\log_{10}(M)$ plots below shows detritus at the bottom-left of the plot.

```
> data(YthanEstuary)
> par(mfrow=c(1,2))
> PlotNvM(YthanEstuary)
> PlotNvM(YthanEstuary, show.na=TRUE)
```



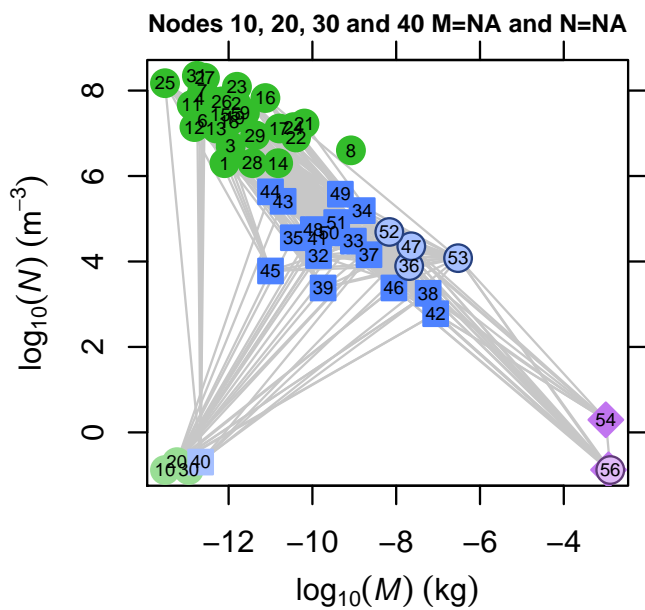
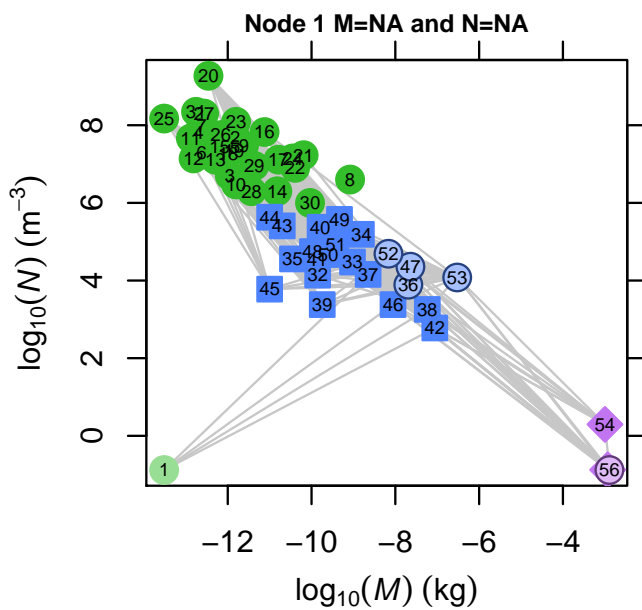
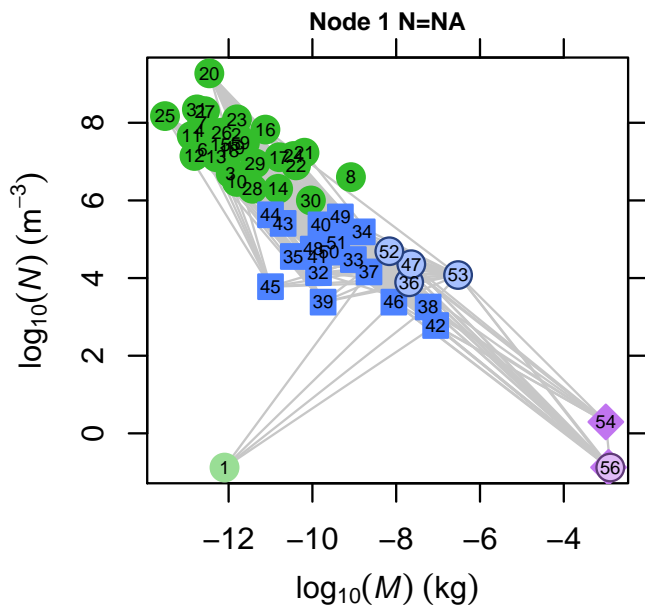
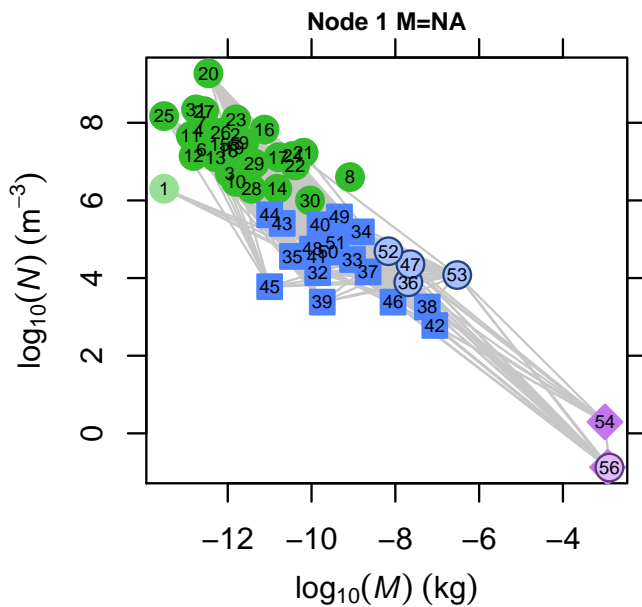
The location of the detritus node is determined by the extent of the data, or by the axes limits, if given.

```
> PlotNvM(YthanEstuary, xlim=c(-10, 4), ylim=c(-10, 13), show.na=TRUE)
```



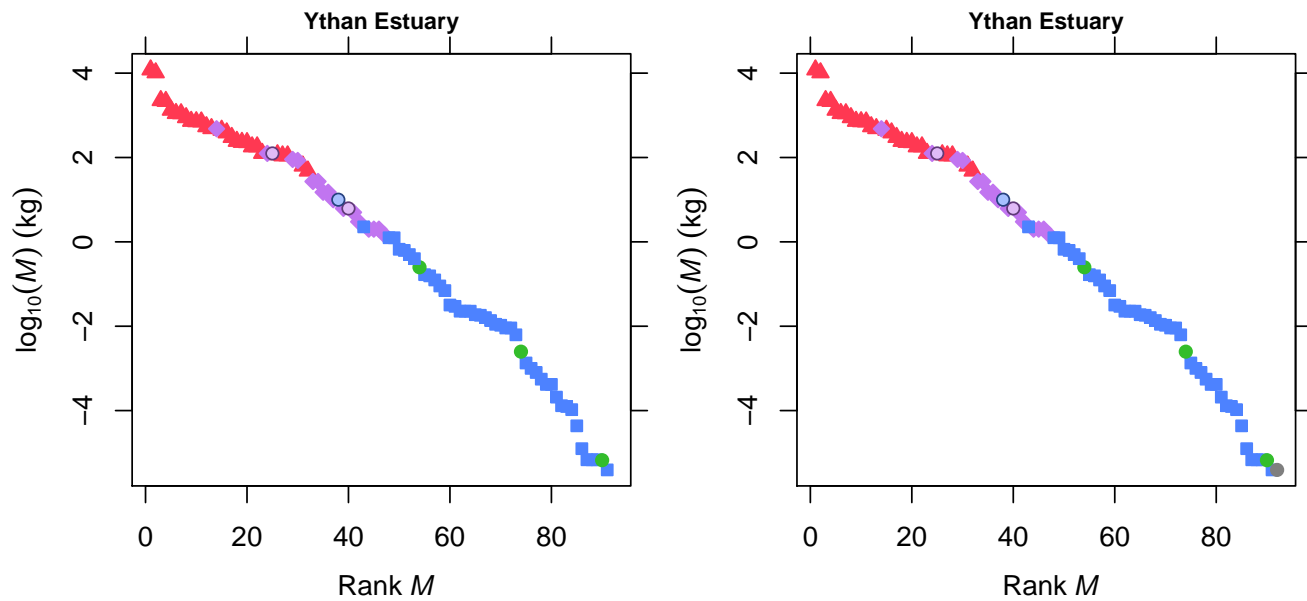
The example below shows how `PlotNPS` behaves in four different cases - node lacks x property, node lacks y property, node lacks both x and y properties and many nodes lack both x and y properties. `lowlight.nodes` defaults to nodes that lack either one of the properties being plotted, so the plots below lowlight the node(s) lacking *M* and/or *N*.

```
> par(mfrow=c(2,2))
> np <- NPS(TL84)
> np[1,'M'] <- NA
> PlotNvM(Community(nodes=np, trophic.links=TLPS(TL84), properties=CPS(TL84)),
  main='Node 1 M=NA', show.nodes.as='both', cex=2, show.na=TRUE)
> np <- NPS(TL84)
> np[1,'N'] <- NA
> PlotNvM(Community(nodes=np, trophic.links=TLPS(TL84), properties=CPS(TL84)),
  main='Node 1 N=NA', show.nodes.as='both', cex=2, show.na=TRUE)
> np <- NPS(TL84)
> np[1,'M'] <- NA
> np[1,'N'] <- NA
> PlotNvM(Community(nodes=np, trophic.links=TLPS(TL84), properties=CPS(TL84)),
  main='Node 1 M=NA and N=NA', show.nodes.as='both', cex=2, show.na=TRUE)
> np <- NPS(TL84)
> np[c(10, 20, 30, 40),'M'] <- NA
> np[c(10, 20, 30, 40),'N'] <- NA
> PlotNvM(Community(nodes=np, trophic.links=TLPS(TL84), properties=CPS(TL84)),
  main='Nodes 10, 20, 30 and 40 M=NA and N=NA', show.nodes.as='both',
  cex=2, show.na=TRUE)
```



The `PlotMvRankM` function delegates work to `PlotNPS`. The behaviour of ‘show.na’ for `PlotMvRankM` is shown below. The left-hand plot does not show the detritus node whereas the right-hand plot shows it as a grey circle.

```
> data(YthanEstuary)
> par(mfrow=c(1,2))
> PlotMvRankM(YthanEstuary)
> PlotMvRankM(YthanEstuary, show.na=TRUE)
```



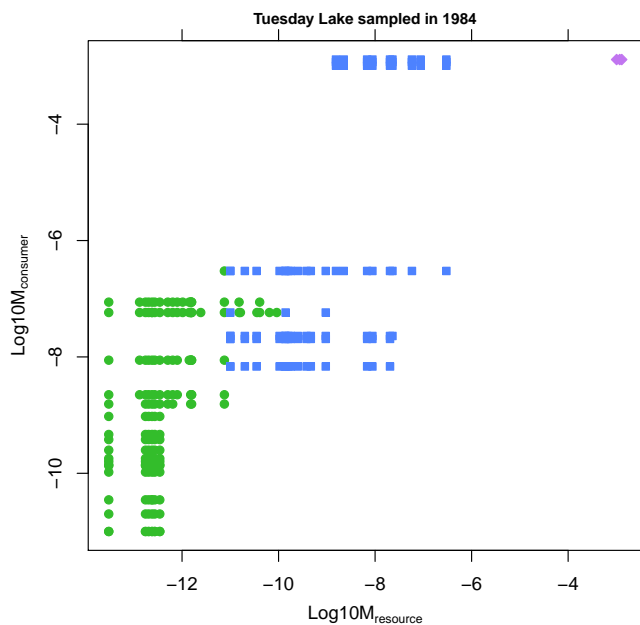
3 Plotting trophic-link properties

3.1 PlotTLPS

PlotTLPS (for **Plot Trophic-Link PropertieS**) plots one trophic-link property against another. Every Cheddar function that plots a point per trophic link delegates the job of plotting to PlotTLPS and all of its power and flexibility are available to all of the functions discussed in this section.

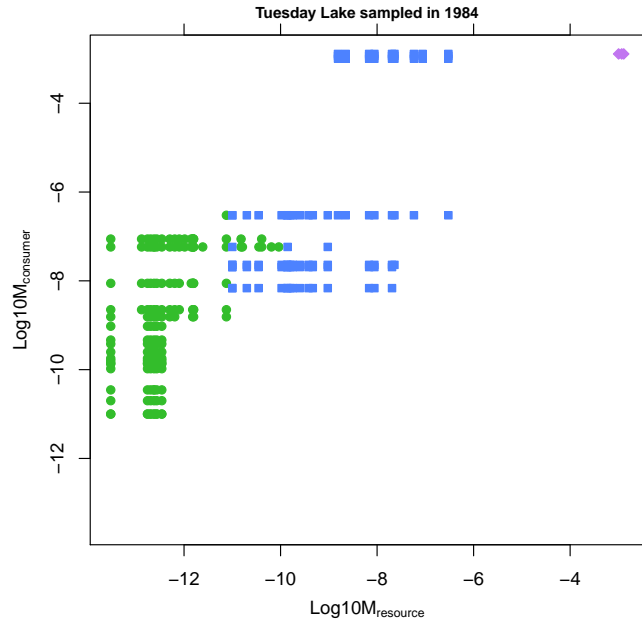
PlotTLPS accepts the names of properties to plot on the x and y axes. These can be both ‘first-class’ and computed properties of trophic links or nodes (where node properties are here viewed as properties of the trophic link of which the node is a part). If a property name begins with either ‘resource.’ or ‘consumer.’ then the remainder of the name is assumed to be a node property. For example, to plot $\log_{10}(M)$ of consumers against $\log_{10}(M)$ of consumers we can use the helper function Log10M.

```
> PlotTLPS(TL84, 'resource.Log10M', 'consumer.Log10M')
```



We can make the plot more informative if the the limits of both the x and y axes are the same. Setting the ‘axes.limits.equal’ parameter to TRUE instructs PlotTLPS to do just that, as shown below.

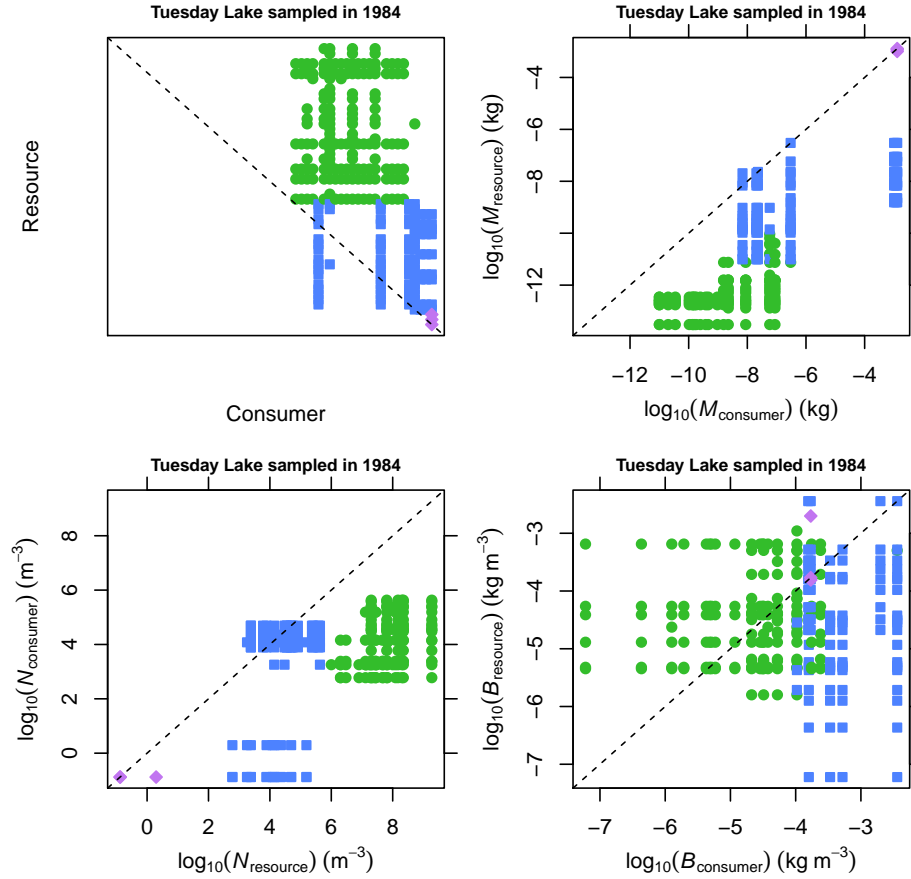
```
> PlotTLPS(TL84, 'resource.Log10M', 'consumer.Log10M', axes.limits.equal=TRUE)
```



PlotTLPS uses the TLPS function to assemble data for plotting. PlotTLPS uses the `category` node property of resources, if present, to decide plotting colours and symbols. Resources that are producers are shown by green circles, invertebrates by blue squares and vertebrate ectotherms by purple diamonds. This is explored more in Section 2.4.

The convenience functions `PlotPredationMatrix`, `PlotMRvMC`, `PlotMCvMR`, `PlotNRvNC`, `PlotNCvNR`, `PlotBRvBC`, `PlotBCvBR` are ‘wrappers’ around `PlotTLPS` that plot a predation matrix (a binary matrix with species shown in node order, starting at the top-left, points on the dashed line are cannibals) or \log_{10} -transformed body mass, M , numerical abundance, N , or biomass, B .

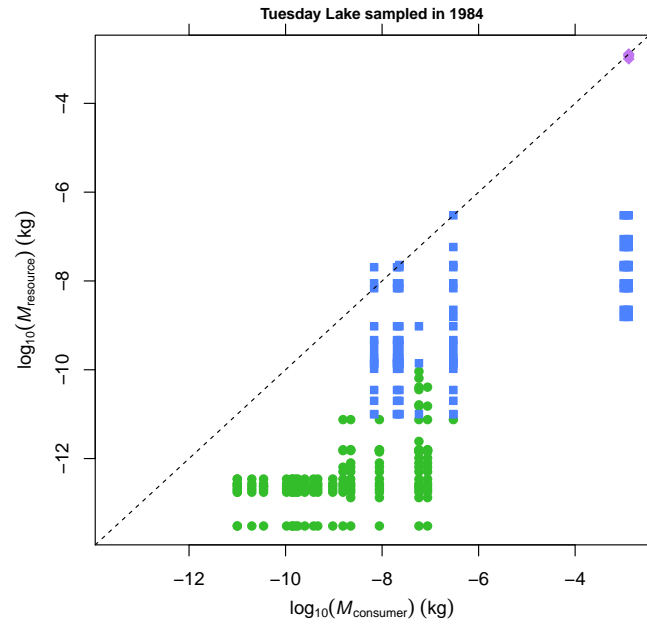
```
> par(mfrow=c(2,2))
> PlotPredationMatrix(TL84)
> PlotMRvMC(TL84)
> PlotNCvNR(TL84)
> PlotBRvBC(TL84)
```



3.2 Colours and symbols

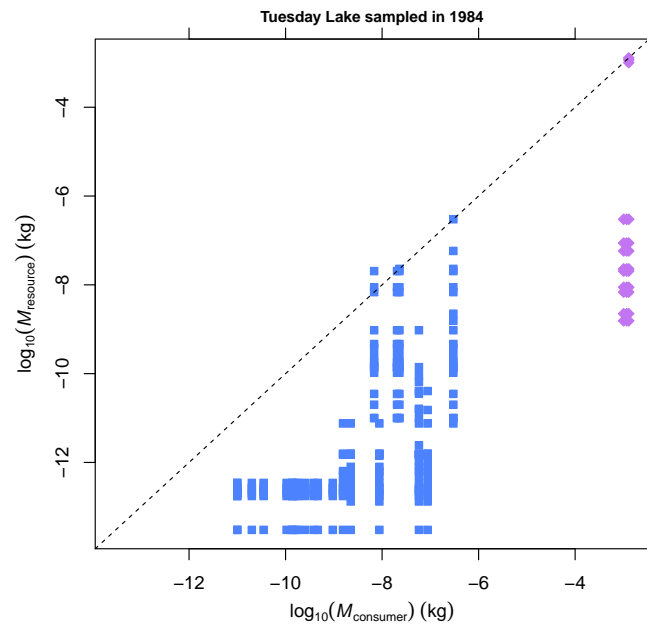
PlotTLPS uses a similar mechanism to PlotNPS. If the community being plotted has a node property called ‘category’ the function uses ‘resource.category’ to decide plotting colours and symbols using the specifications given in `DefaultCategoryColours` and `DefaultCategorySymbols`, as shown by this plot of consumer-versus-resource \log_{10} -transformed body mass.

```
> PlotMRvMC(TL84)
```



To plot the same data using the consumer's category to decide colours and symbols.

```
> PlotMRvMC(TL84, colour.by='consumer.category', bg.by='consumer.category',  
  symbol.by='consumer.category')
```



The `bg.by` parameter is used to determine the background colour of each point. See the section for ‘bg’ in the help page for the R `par` function for more information.

4 Values by class

Cheddar provides functions that compute values over classes of nodes. For example, we can examine the total M , N and biomass in each category.

```
> SumMByClass(TL84)
```

```
invertebrate    producer    vert.ecto
5.110800e-07 1.127248e-09 3.470000e-03
```

```
> SumNByClass(TL84)
```

```
invertebrate    producer    vert.ecto
1.8678e+06 3.2107e+09 2.2350e+00
```

```
> SumBiomassByClass(TL84)
```

```
invertebrate    producer    vert.ecto
0.005376146 0.007438267 0.002315590
```

These functions can show the sums by any property of the community so we could look at the same totals by kingdom.

```
> SumMByClass(TL84, 'kingdom')
```

```
<unnamed>    Animalia    Bacteria    Chromista    Plantae    Protozoa
3.460000e-13 3.470511e-03 1.736800e-11 9.488933e-10 1.054600e-11 1.500950e-10
```

```
> SumNByClass(TL84, 'kingdom')
```

```
<unnamed>    Animalia    Bacteria    Chromista    Plantae    Protozoa
1880000000 1867802 81000000 425700000 373000000 451000000
```

```
> SumBiomassByClass(TL84, 'kingdom')
```

```
<unnamed>    Animalia    Bacteria    Chromista    Plantae    Protozoa
0.000650480 0.007691736 0.000223046 0.003699680 0.000205291 0.002659770
```

The ‘Unclassified flagellates’ node does not have a kingdom, and the resulting value is labelled ‘<unnamed>’. These convenience functions are ‘wrappers’ around the more general-purpose `ApplyByClass` function. The following two calls produce the same result.

```
> SumBiomassByClass(TL84)
```

```
invertebrate    producer    vert.ecto
0.005376146 0.007438267 0.002315590
```

```
> ApplyByClass(TL84, 'Biomass', 'category', sum)
```

```
invertebrate    producer    vert.ecto
0.005376146 0.007438267 0.002315590
```

See the help page for `ApplyByClass` for more examples.

5 $\log_{10}(N)$ -versus- $\log_{10}(M)$ statistics

Cheddar contains some helper functions that assist in the commonly-performed $\log_{10}(N)$ -versus- $\log_{10}(M)$ statistics. `NvMLLinearRegressions` returns a list of linear regressions through $\log_{10}(N)$ -versus- $\log_{10}(M)$ node data. By default it fits a regression through all of the nodes (called ‘all’) and a separate regression through each node category.

```
> models <- NvMLLinearRegressions(TL84)
> names(models)

[1] "all"          "producer"      "invertebrate" "vert.ecto"
```

You can extract the slopes and intercepts for each of these regression models.

```
> sapply(models, 'coef')

              all      producer invertebrate vert.ecto
(Intercept) -2.6862753  2.5583364    1.4656062 -34.66097
x            -0.8271145 -0.4071509   -0.3243168 -11.62787
```

The ‘class’ parameter defines the sets of nodes to which `NvMLLinearRegressions` fits regressions. This defaults to ‘category’, if this is present in the community. You can set the ‘class’ parameter to fit regressions through different sets of nodes, such as through each phylum as shown below.

```
> models <- NvMLLinearRegressions(TL84, class='phylum')
> names(models)

[1] "all"          "Cyanobacteria" "Charophyta"    "Ochrophyta"    "Cryptophyta"
[6] "Chlorophyta"  "Myzozoa"       "<unnamed>"     "Euglenozoa"    "Rotifera"
[11] "Arthropoda"   "Chordata"
```

The ‘<unnamed>’ label refers to a model fitted to nodes that do not have a phylum - just the ‘unclassified flagellates’ node in this case. It is not possible to fit a linear regression to a single data point and `NvMLLinearRegressions` returns NULL for classes that contain just a single node. The data contain one node in phylum Euglenozoa and one node without a phylum, so these entries are NULL.

```
> sapply(models, is.null)

              all Cyanobacteria   Charophyta   Ochrophyta   Cryptophyta   Chlorophyta
FALSE          FALSE           FALSE        FALSE        FALSE           FALSE
Myzozoa        <unnamed>      Euglenozoa    Rotifera    Arthropoda    Chordata
FALSE          TRUE            TRUE          FALSE        FALSE           FALSE
```

`NvMLLinearRegressions` also returns NULL for classes where all or all bar one of the nodes have M and/or M of NA. The Broadstone stream dataset contains detrital and algal nodes that all lack both M and N ; these nodes are in the <unnamed> category.

```
> data(BroadstoneStream)
> models <- NvMLLinearRegressions(BroadstoneStream)
> sapply(models, is.null)
```

all	invertebrate	<unnamed>	producer
FALSE	FALSE	TRUE	TRUE

Three helper functions return the slope and/or intercept of an ordinary linear regression through all data.

```
> NvMSlope(TL84)
```

```
[1] -0.8271145
```

```
> NvMIntercept(TL84)
```

```
[1] -2.686275
```

```
> NvMSlopeAndIntercept(TL84)
```

slope	intercept
-0.8271145	-2.6862753

Three more helper functions extract these coefficients for each class.

```
> NvMSlopeByClass(TL84)
```

slope.all	slope.producer	slope.invertebrate	slope.vert.ecto
-0.8271145	-0.4071509	-0.3243168	-11.6278709

```
> NvMInterceptByClass(TL84)
```

intercept.all	intercept.producer	intercept.invertebrate
-2.686275	2.558336	1.465606
intercept.vert.ecto		
-34.660973		

```
> NvMSlopeAndInterceptByClass(TL84)
```

slope.all	slope.producer	slope.invertebrate
-0.8271145	-0.4071509	-0.3243168
slope.vert.ecto	intercept.all	intercept.producer
-11.6278709	-2.6862753	2.5583364
intercept.invertebrate	intercept.vert.ecto	
1.4656062	-34.6609728	

You can fit regressions to whatever class you like.

```
> NvMSlopeByClass(TL84, class='kingdom')
```

slope.all	slope.Bacteria	slope.Plantae	slope.Chromista	slope.Protozoa
-0.8271145	-0.2028766	-1.0654368	-0.4177200	-0.5434582
slope.<unnamed>	slope.Animalia			
NA	-0.6737747			

```
> NvMInterceptByClass(TL84, class='kingdom')
```

intercept.all	intercept.Bacteria	intercept.Plantae	intercept.Chromista
-2.686275	4.640388	-5.409768	2.236353
intercept.Protozoa	intercept.<unnamed>	intercept.Animalia	
1.547918	NA	-1.796438	

> *NvMSlopeAndInterceptByClass*(TL84, class='kingdom')

slope.all	slope.Bacteria	slope.Plantae	slope.Chromista
-0.8271145	-0.2028766	-1.0654368	-0.4177200
slope.Protozoa	slope.<unnamed>	slope.Animalia	intercept.all
-0.5434582	NA	-0.6737747	-2.6862753
intercept.Bacteria	intercept.Plantae	intercept.Chromista	intercept.Protozoa
4.6403880	-5.4097680	2.2363526	1.5479185
intercept.<unnamed>	intercept.Animalia		
NA	-1.7964381		

6 Tri-trophic statistics

Cohen et al. (2009) proposed a novel analysis that simultaneously considers body mass, M , numerical abundance, N , and the community's food web. The `NvMTriTrophicStatistics` function provides this analysis. It takes a single community as its only parameter. Cannibalistic links and all nodes with an N or M of NA are removed before statistics are computed. It returns a list containing three matrices: 'links', 'three.node.chains' and 'trophic.chains'. The examples in Section 7.4 show how to use this function and the three matrices that it returns to reconstruct the important figures and tables from Cohen et al. (2009).

7 Examples through recreating figures and tables from published articles

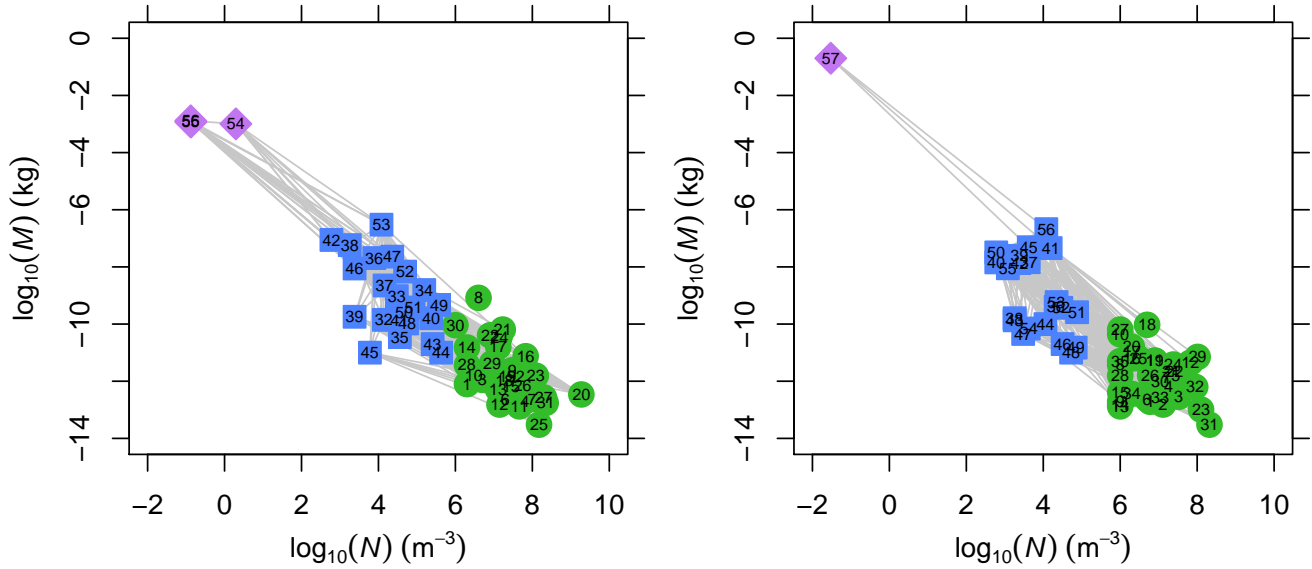
The following sections show how to use Cheddar to recreate some published figures.

7.1 Jonsson et al. (2005)

7.1.1 Jonsson et al. (2005), Fig. 3 (p 30)

```
> data(TL84, TL86)
> par(mfrow=c(1,2))
> PlotMvN(TL84, show.nodes.as='both', cex=2, xlim=c(-2, 10), ylim=c(-14, 0),
  highlight.nodes=NULL, highlight.links=NULL, main='')
> PlotMvN(TL86, show.nodes.as='both', cex=2, xlim=c(-2, 10), ylim=c(-14, 0),
  highlight.nodes=NULL, highlight.links=NULL, main='')
> title(main='Jonsson et al. (2005) AER, Fig. 3 (p 30)', outer=TRUE)
```

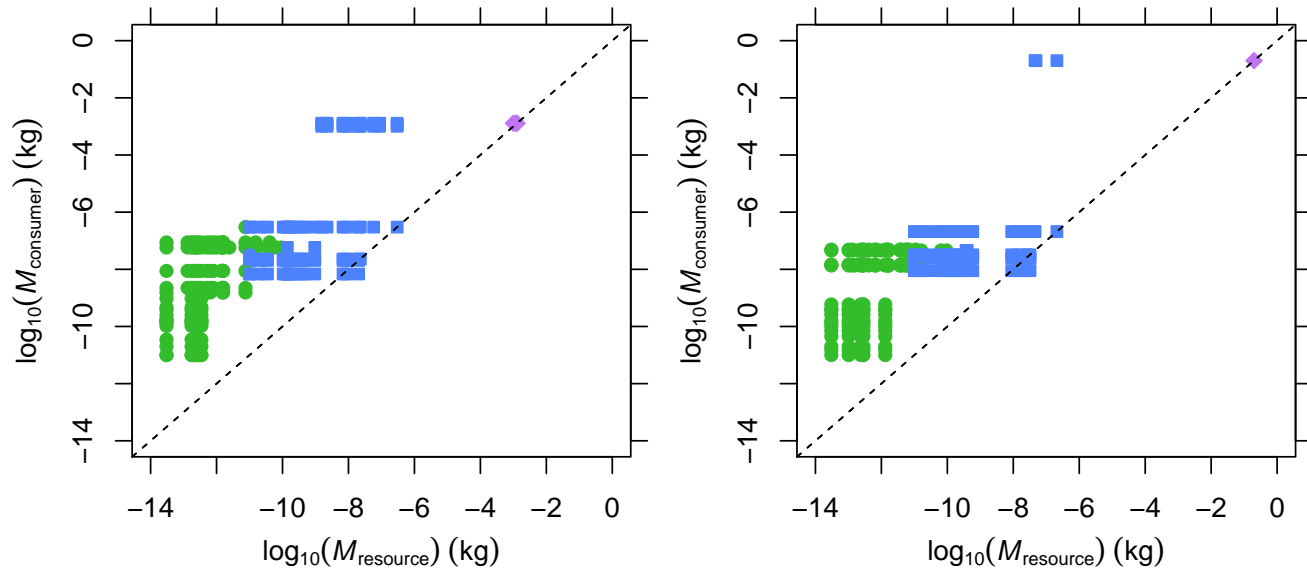
Jonsson et al. (2005) AER, Fig. 3 (p 30)



7.1.2 Jonsson et al. (2005), Fig. 4 (p 33)

```
> par(mfrow=c(1,2))
> PlotMCvMR(TL84, xlim=c(-14, 0), ylim=c(-14, 0), main='')
> abline(a=0, b=1, lty=2)
> PlotMCvMR(TL86, xlim=c(-14, 0), ylim=c(-14, 0), main='')
> abline(a=0, b=1, lty=2)
> title(main='Jonsson et al. (2005) AER, Fig. 4 (p 33)', outer=TRUE)
```

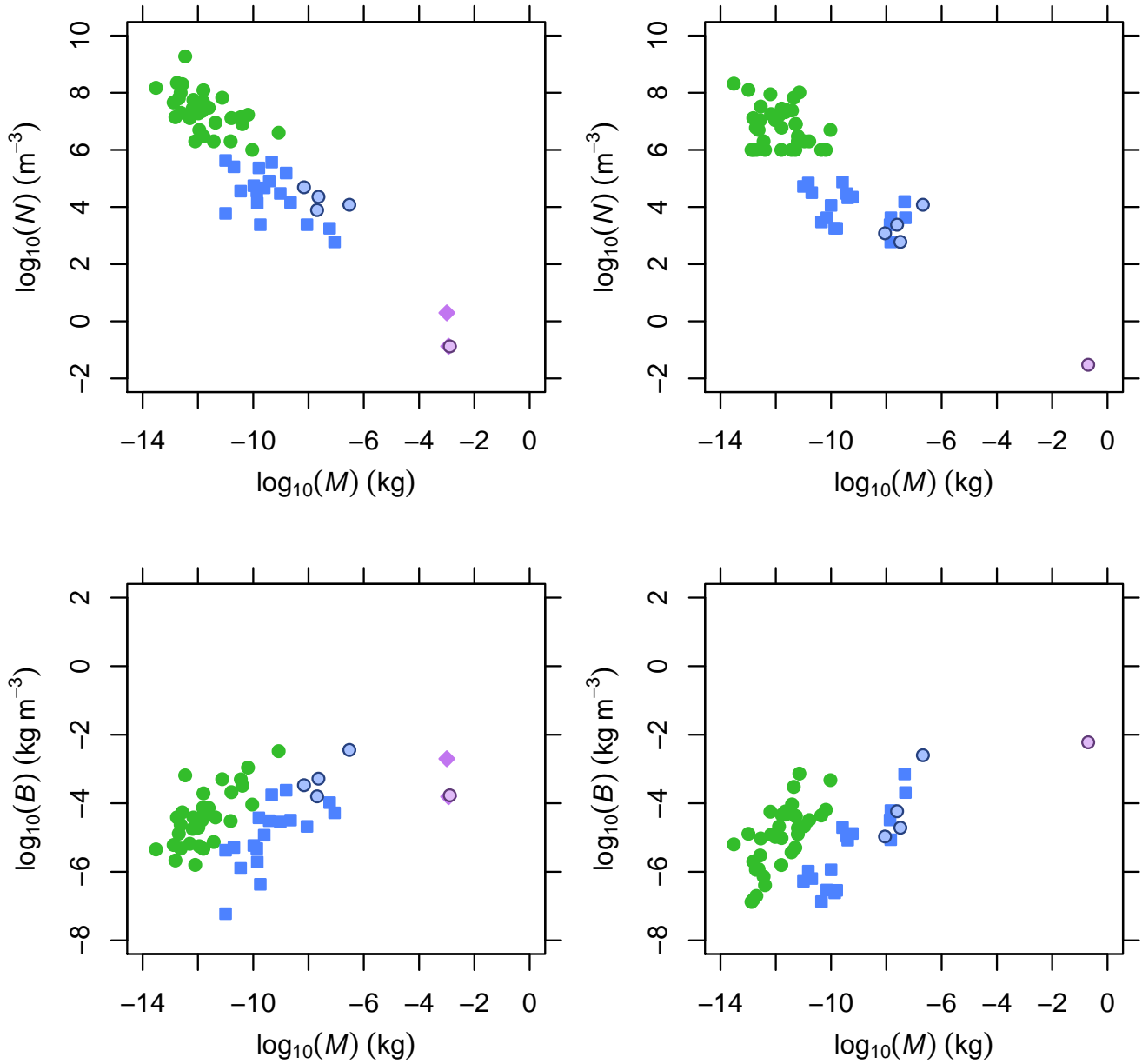
Jonsson et al. (2005) AER, Fig. 4 (p 33)



7.1.3 Jonsson et al. (2005), Fig. 5 (p 37)

```
> par(mfrow=c(2,2))
> PlotNmM(TL84, xlim=c(-14, 0), ylim=c(-2,10), show.web=FALSE, main='')
> PlotNmM(TL86, xlim=c(-14, 0), ylim=c(-2,10), show.web=FALSE, main='')
> PlotBvM(TL84, xlim=c(-14, 0), ylim=c(-8,2), show.web=FALSE, main='')
> PlotBvM(TL86, xlim=c(-14, 0), ylim=c(-8,2), show.web=FALSE, main='')
> title(main='Jonsson et al. (2005) AER, Fig. 5 (p 37)', outer=TRUE)
```

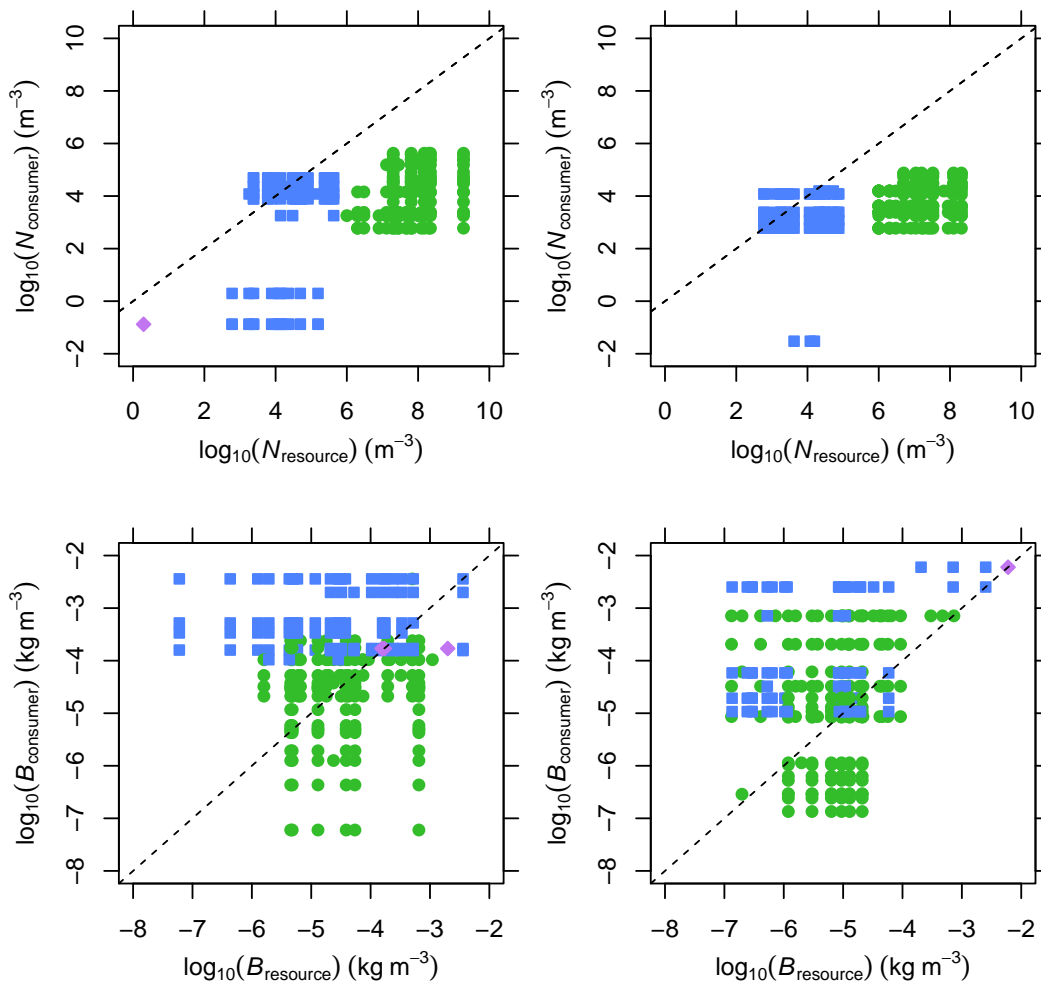
Jonsson et al. (2005) AER, Fig. 5 (p 37)



7.1.4 Jonsson et al. (2005), Fig. 7 (p 47)

```
> par(mfrow=c(2,2))
> PlotNCvNR(TL84, xlim=c(0, 10), ylim=c(-2,10), main='')
> abline(a=0, b=1, lty=2)
> PlotNCvNR(TL86, xlim=c(0, 10), ylim=c(-2,10), main='')
> abline(a=0, b=1, lty=2)
> PlotBCvBR(TL84, xlim=c(-8, -2), ylim=c(-8, -2), main='')
> abline(a=0, b=1, lty=2)
> PlotBCvBR(TL86, xlim=c(-8, -2), ylim=c(-8, -2), main='')
> abline(a=0, b=1, lty=2)
> title(main='Jonsson et al. (2005) AER, Fig. 7 (p 47)', outer=TRUE)
```

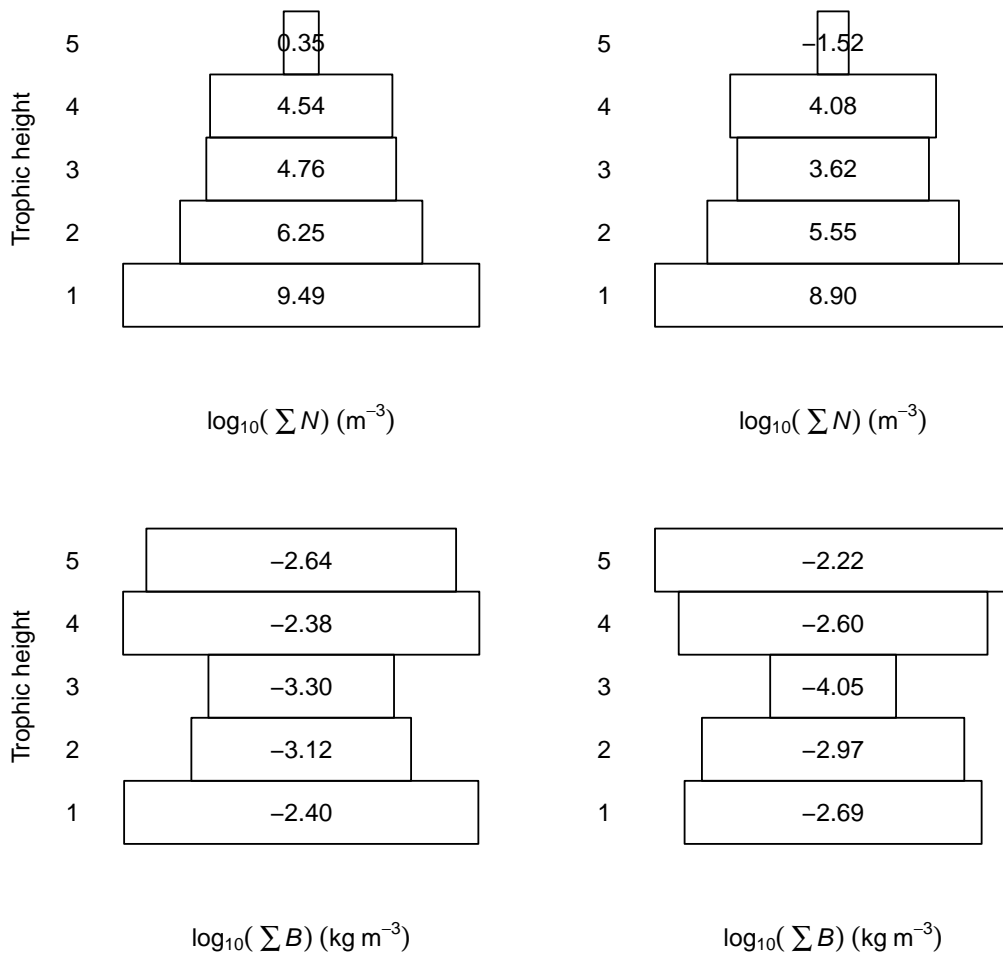
Jonsson et al. (2005) AER, Fig. 7 (p 47)



7.1.5 Jonsson et al. (2005), Fig. 8 (p 49)

```
> par(mfrow=c(2,2))
> TL84.no.iso <- RemoveIsolatedNodes(TL84)
> TL86.no.iso <- RemoveIsolatedNodes(TL86)
> tl84.levels <- floor(TrophicHeight(TL84.no.iso))
> tl86.levels <- floor(TrophicHeight(TL86.no.iso))
> PlotNPyramid(TL84.no.iso, level=tl84.levels, main='', ylab='Trophic height')
> PlotNPyramid(TL86.no.iso, level=tl86.levels, main='')
> PlotBPyramid(TL84.no.iso, level=tl84.levels, main='', ylab='Trophic height')
> PlotBPyramid(TL86.no.iso, level=tl86.levels, main='')
> title(main='Jonsson et al. (2005) AER, Fig. 8 (p 49)', outer=TRUE)
```

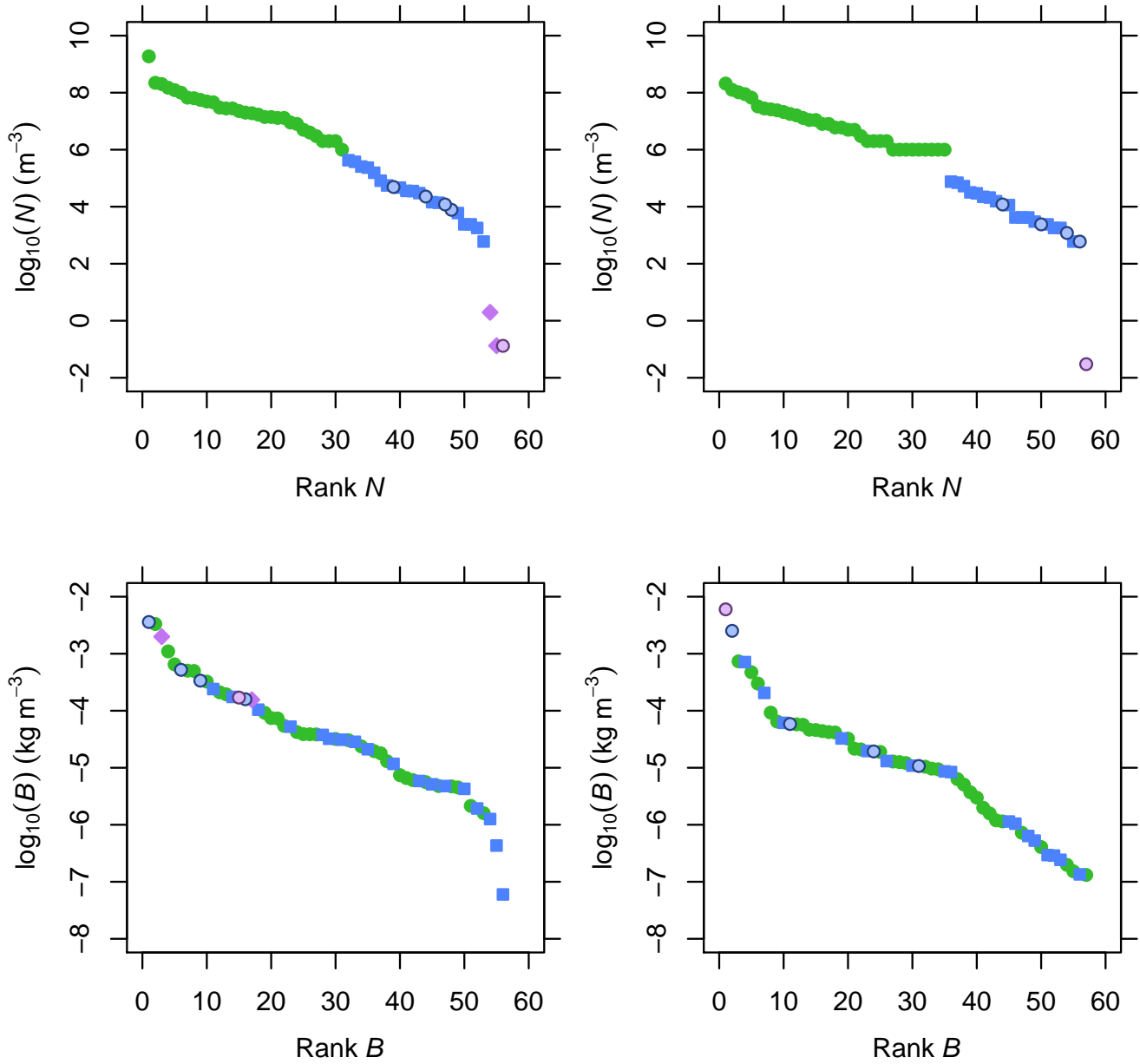
Jonsson et al. (2005) AER, Fig. 8 (p 49)



7.1.6 Jonsson et al. (2005), Fig. 10 (p 57)

```
> par(mfrow=c(2,2))
> PlotNvRankN(TL84, xlim=c(0,60), ylim=c(-2, 10), main='')
> PlotNvRankN(TL86, xlim=c(0,60), ylim=c(-2, 10), main='')
> PlotBvRankB(TL84, xlim=c(0,60), ylim=c(-8, -2), main='')
> PlotBvRankB(TL86, xlim=c(0,60), ylim=c(-8, -2), main='')
> title(main='Jonsson et al. (2005) AER, Fig. 10 (p 57)', outer=TRUE)
```

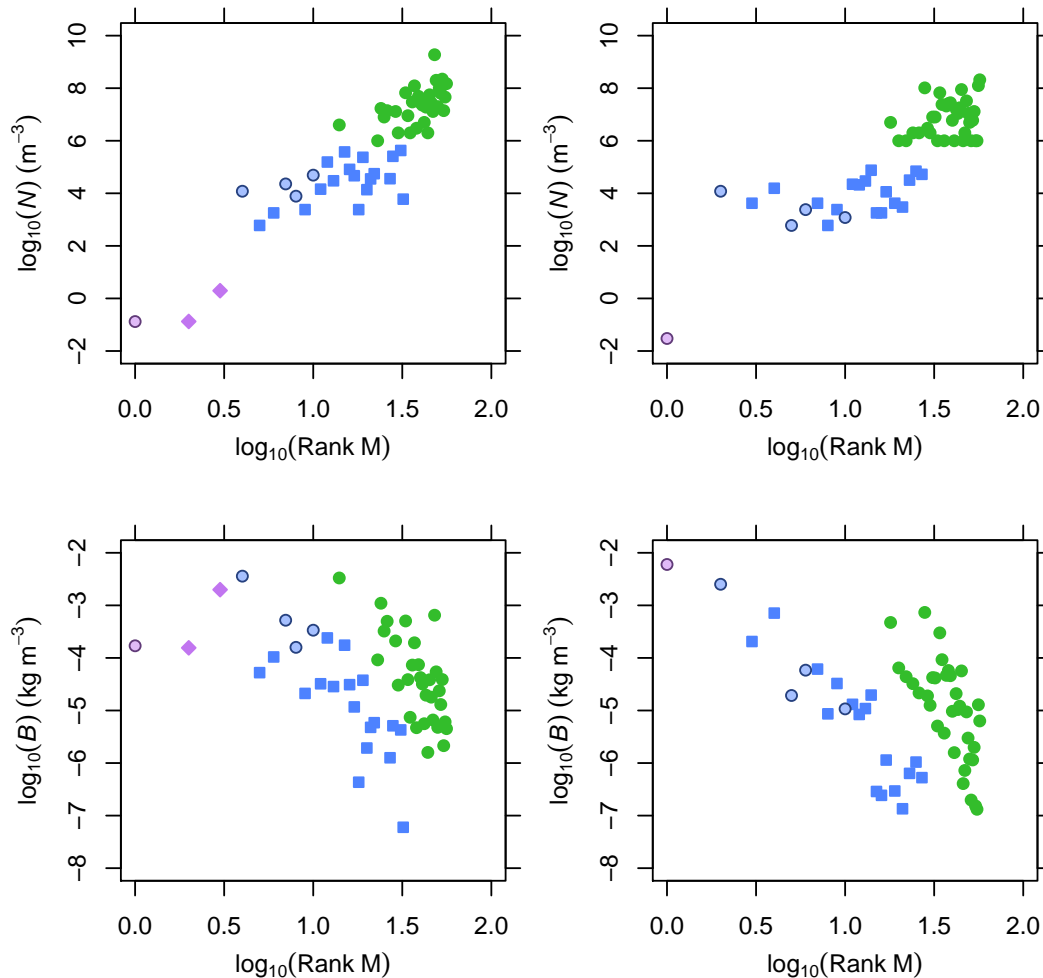
Jonsson et al. (2005) AER, Fig. 10 (p 57)



7.1.7 Jonsson et al. (2005), Fig. 11 (p 60)

```
> par(mfrow=c(2,2))
> PlotRankNPS(TL84, property='Log10N', rank.by='M', log10.rank=TRUE,
  xlim=c(0,2), ylim=c(-2, 10), ylab=Log10NLabel(TL84), main='')
> PlotRankNPS(TL86, property='Log10N', rank.by='M', log10.rank=TRUE,
  xlim=c(0,2), ylim=c(-2, 10), ylab=Log10NLabel(TL84), main='')
> PlotRankNPS(TL84, property='Log10Biomass', rank.by='M',
  log10.rank=TRUE, xlim=c(0,2), ylim=c(-8, -2),
  ylab=Log10BLabel(TL84), main='')
> PlotRankNPS(TL86, property='Log10Biomass', rank.by='M',
  log10.rank=TRUE, xlim=c(0,2), ylim=c(-8, -2),
  ylab=Log10BLabel(TL84), main='')
> title(main='Jonsson et al. (2005) AER, Fig. 11 (p 60)', outer=TRUE)
```

Jonsson et al. (2005) AER, Fig. 11 (p 60)



7.1.8 Jonsson et al. (2005), Fig. 12 (p 61)

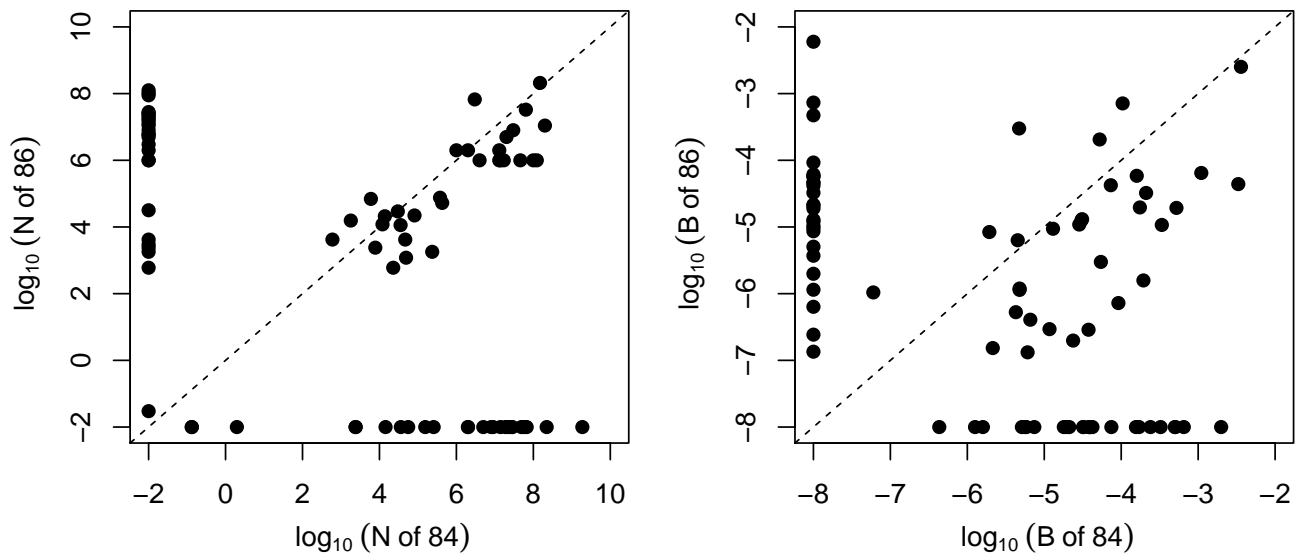
In this example, we write a small function that plots one community against another.

```
> PlotCommunityVCommunity <- function(a, b, property, xlim=NULL, ylim=NULL, ...)
{
  a.nodes <- NP(a, 'node')
  b.nodes <- NP(b, 'node')
  all.nodes <- union(a.nodes, b.nodes)

  a.values <- NPS(a, property)[,property]
  names(a.values) <- a.nodes
  b.values <- NPS(b, property)[,property]
  names(b.values) <- b.nodes
  points <- PlaceMissingPoints(a.values[all.nodes], xlim,
                              b.values[all.nodes], ylim)
  plot(points[,1], points[,2], xlim=xlim, ylim=ylim, ...)

  abline(a=0, b=1, lty=2)
}
> par(mfrow=c(1,2))
> PlotCommunityVCommunity(TL84, TL86, 'Log10N', xlim=c(-2,10), ylim=c(-2,10),
  xlab=~log[10]~(N~of~84), ylab=~log[10]~(N~of~86),pch=19)
> PlotCommunityVCommunity(TL84, TL86, 'Log10Biomass',
  xlim=c(-8,-2), ylim=c(-8,-2),
  xlab=~log[10]~(B~of~84), ylab=~log[10]~(B~of~86),pch=19)
> title(main='Jonsson et al. (2005) AER, Fig. 12 (p 61)', outer=TRUE)
```

Jonsson et al. (2005) AER, Fig. 12 (p 61)



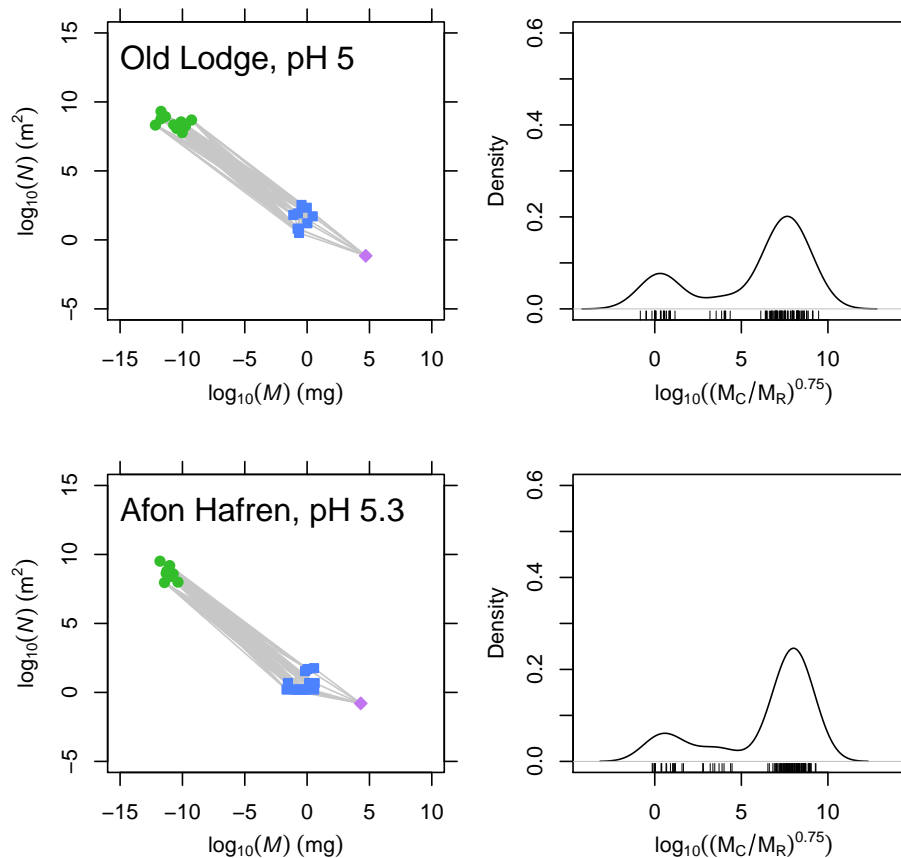
7.2 Layer et al. (2010)

7.2.1 Layer et al. (2010), Fig. 6 (p 282)

Four of the panels from the published figure. This example uses the ‘pHWebs’ community collection dataset; see the ‘Collections’ vignette for more information about community collections.

```
> data(pHWebs)
> par(mfrow=c(2,2))
> for(community in pHWebs[1:2])
{
  PlotNmM(community, xlim=c(-15, 10), ylim=c(-5,15), main='',
    highlight.nodes=NULL)
  text(-15, 13, with(CPS(community), paste(title, ', pH ', pH, sep='')),
    adj=0, cex=1.5)
  tpls <- TLPS(community, node.properties='M')
  tpls <- tpls[!is.na(tpls$resource.M) & !is.na(tpls$consumer.M),]
  interaction.strength <- log10( (tpls$consumer.M / tpls$resource.M)^0.75 )
  plot(density(interaction.strength), xlim=c(-4,14), ylim=c(0,0.6),
    main='', xlab=~log[10]((M[C]/M[R])^0.75))
  rug(interaction.strength)
}
> title(main='Layer et al. (2010) AER, Fig. 6 (p 282)', outer=TRUE)
```

Layer et al. (2010) AER, Fig. 6 (p 282)



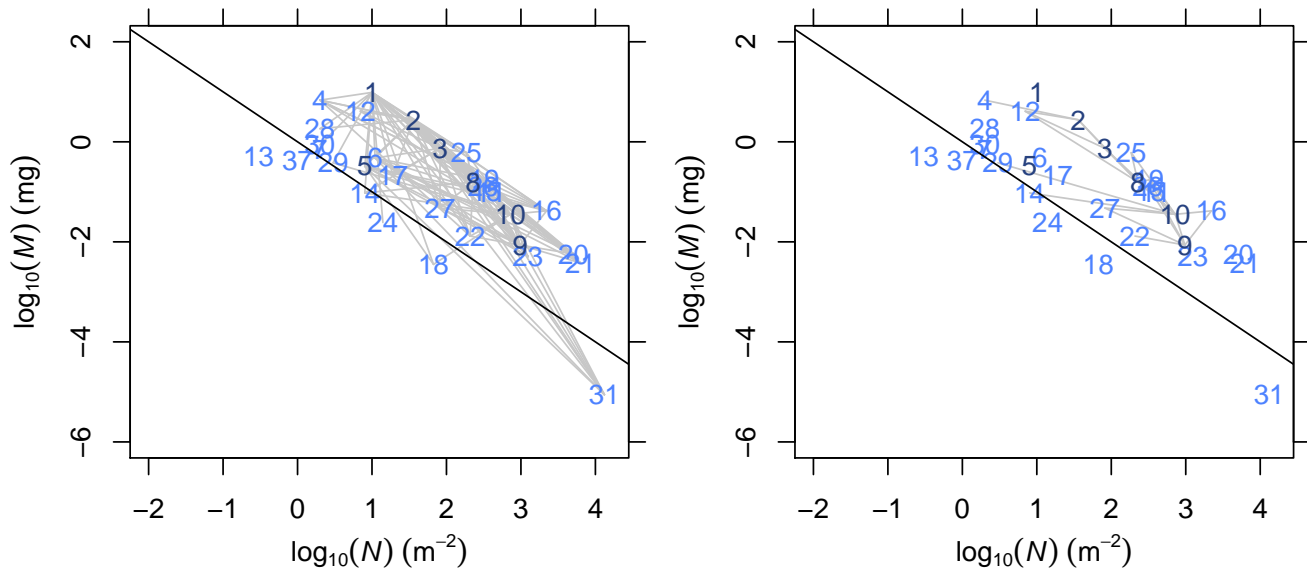
7.3 Woodward et al. (2005)

7.3.1 Woodward et al. (2005), Fig. 4. (p 108)

The left panel shows all trophic links. The right panel shows just links for which the resource has a larger body mass than the consumer.

```
> data(BroadstoneStream)
> par(mfrow=c(1,2))
> PlotMvN(BroadstoneStream, show.nodes.as='labels', label.cex=0.8,
  xlim=c(-2, 4.2), ylim=c(-6,2), main='', show.na=FALSE,
  highlight.links=NULL)
> abline(a=0, b=-1)
> ttps <- TLPS(BroadstoneStream, node.properties='M')
> lty <- rep(0, NumberOfTrophicLinks(BroadstoneStream))
> lty[ttps$resource.M > ttps$consumer.M] <- 1
> PlotMvN(BroadstoneStream, show.nodes.as='labels', label.cex=0.8,
  xlim=c(-2, 4.2), ylim=c(-6,2), main='', show.na=FALSE,
  highlight.links=NULL, link.lty=lty)
> abline(a=0, b=-1)
> title(main='Woodward et al. (2005) AER, Fig. 4 (p 108)', outer=TRUE)
```

Woodward et al. (2005) AER, Fig. 4 (p 108)



7.4 Cohen et al. (2009)

7.4.1 Cohen et al. (2009), Table 1 (p 22338)

This code removes isolated nodes from communities for consistency with the paper. `NvMTriTrophicStatistics` does this for us anyway so the $M - N$ tritrophic statistics are not affected but the network statistics in the last four rows of the table would be different to those in the paper.

```
> data(TL84, TL86, YthanEstuary)
> res <- lapply(list(TL84, TL86, YthanEstuary), function(community)
{
  community <- RemoveNodes(community, remove=with(NPS(community), node[is.na(M) | is.na(N)]))
  community <- RemoveCannibalisticLinks(community)
  community <- RemoveIsolatedNodes(community)

  tts<-NvMTriTrophicStatistics(community)
  lp <- tts[['links']]
  tncp <- tts[['three.node.chains']]
  tcp <- tts[['trophic.chains']]

  community.span <- diff(range(Log10M(community))) +
    diff(range(Log10N(community)))

  wiggling <- mean(tcp$sum.chain.length) / mean(tcp$chain.span)

  UnsafeMean <- function(x)
  {
    ifelse(is.null(x), NA, mean(x))
  }

  return (c(
    'Mean link length'=mean(lp$length),
    'Mean L upper'=UnsafeMean(tncp$Lupper),
    'Mean L lower'=UnsafeMean(tncp$Llower),
    '2 x mean link length'=2*mean(lp$length),
    'Mean 2-span'=UnsafeMean(tncp$two.span),
    'Mean L upper + L lower'=UnsafeMean(tncp$Lupper + tncp$Llower),
    '2 x mean link length / mean 2-span'=
      2 * mean(lp$length) / UnsafeMean(tncp$two.span),
    'Mean L upper + L lower/ mean 2-span'=
      UnsafeMean(tncp$Lupper + tncp$Llower) / UnsafeMean(tncp$two.span),
    'Mean count chain length'=mean(tcp$count.chain.length),
    'Mean count chain length x mean link length'=
      mean(tcp$count.chain.length)*mean(lp$length),
    'Community span'=community.span,
    'Mean count chain length x mean link length / community span'=
      mean(tcp$count.chain.length) * mean(lp$length)/community.span,
    'Mean sum chain lengths'=mean(tcp$sum.chain.length),
```

```

'Mean chain span'=mean(tcp$chain.span),
'Mean chain span / community span'=mean(tcp$chain.span) / community.span,
'Mean sum chain lengths / mean chain span'=
  mean(tcp$sum.chain.length) / mean(tcp$chain.span),
'Mean sum chain lengths / community span'=
  mean(tcp$sum.chain.length) / community.span,
'L'=NumberOfTrophicLinks(community),
'S^2'=NumberOfNodes(community)^2,
'L/S^2'=DirectedConnectance(community),
'L/S'=LinkageDensity(community)))
})
> res <- do.call('cbind', res)
> colnames(res) <- c('TL84', 'TL86', 'Ythan Estuary')
> print(round(res,2))

```

	TL84	TL86	Ythan Estuary
Mean link length	6.33	5.90	7.29
Mean L upper	5.41	3.43	5.06
Mean L lower	5.99	5.69	6.15
2 x mean link length	12.67	11.79	14.57
Mean 2-span	11.02	8.65	10.51
Mean L upper + L lower	11.40	9.12	11.20
2 x mean link length / mean 2-span	1.15	1.36	1.39
Mean L upper + L lower/ mean 2-span	1.03	1.05	1.07
Mean count chain length	4.84	4.84	4.43
Mean count chain length x mean link length	30.62	28.56	32.31
Community span	20.78	22.66	21.98
Mean count chain length x mean link length / community span	1.47	1.26	1.47
Mean sum chain lengths	19.96	23.33	16.88
Mean chain span	18.71	20.63	13.18
Mean chain span / community span	0.90	0.91	0.60
Mean sum chain lengths / mean chain span	1.07	1.13	1.28
Mean sum chain lengths / community span	0.96	1.03	0.77
L	264.00	236.00	379.00
S^2	2500.00	2601.00	8281.00
L/S^2	0.11	0.09	0.05
L/S	5.28	4.63	4.16

7.4.2 Cohen et al. (2009), Table S3 (p 8)

```

> res <- lapply(list(TL84, TL86, YthanEstuary), function(community)
{
  community <- RemoveNodes(community, remove=with(NPS(community), node[is.na(M) | is.na(N)]))
  community <- RemoveCannibalisticLinks(community)
  community <- RemoveIsolatedNodes(community)

  chains <- ThreeNodeChains(community, node.properties='M')
  MR <- chains$bottom.M
  MI <- chains$intermediate.M

```

```

MC <- chains$top.M

lp <- TLPS(community, node.properties='M')

return (c('MR<=MI<=MC'=sum(MR<=MI & MI<=MC),
          'MR<=MC<MI'=sum(MR<=MC & MC<MI),
          'MI<MR<=MC'=sum(MI<MR & MR<=MC),
          'MI<=MC<MR'=sum(MI<=MC & MC<MR),
          'MC<MR<MI'=sum(MC<MR & MR<MI),
          'MC<MI<MR'=sum(MC<MI & MI<MR),
          'All 2-chains'=nrow(chains),
          'MR<MC'=sum(lp$resource.M<lp$consumer.M),
          'MR=MC'=sum(lp$resource.M==lp$consumer.M),
          'MR>MC'=sum(lp$resource.M>lp$consumer.M),
          'All links'=nrow(lp)))
})
> res <- do.call('cbind', res)
> colnames(res) <- c('TL84', 'TL86', 'Ythan Estuary')
> print(round(res,2))

```

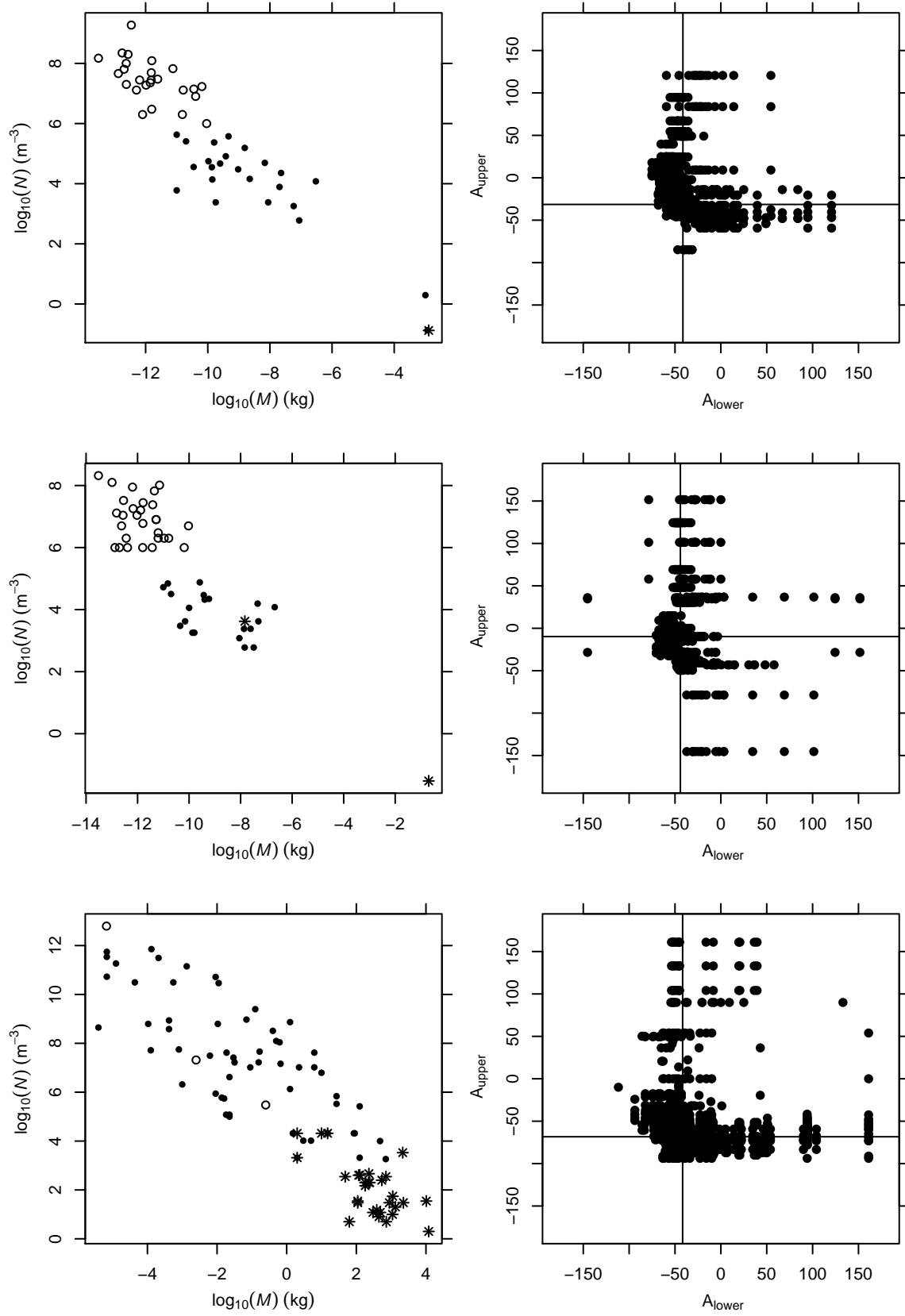
	TL84	TL86	Ythan Estuary
MR<=MI<=MC	1001	577	1232
MR<=MC<MI	30	59	65
MI<MR<=MC	12	10	68
MI<=MC<MR	0	1	3
MC<MR<MI	1	3	0
MC<MI<MR	0	1	3
All 2-chains	1044	651	1371
MR<MC	262	232	368
MR=MC	0	0	2
MR>MC	2	4	9
All links	264	236	379

7.4.3 Cohen et al. (2009), Fig. 1, (p 22336)

```

> data(TL84, TL86, YthanEstuary)
> par(mfrow=c(3,2))
> for(community in list(TL84, TL86, YthanEstuary))
{
  community <- RemoveIsolatedNodes(community)
  pch <- rep(1, NumberOfNodes(community))
  pch[IsIntermediateNode(community)] <- 20
  pch[IsTopLevelNode(community)] <- 8
  PlotNmM(community, col=1, highlight.nodes=NULL, show.web=FALSE,
          main='', pch=pch)
  PlotAupperVAlower(community, main='')
}
> title(main='Cohen et al. (2009) PNAS, Fig. 1 (p 22336)', outer=TRUE)

```



References

- J.H. Brown, J.F. Gillooly, A.P. Allen, V.M. Savage, and G.B. West. Toward a metabolic theory of ecology. *Ecology*, 85(7):1771–1789, 2004. doi: 10.1890/03-9000.
- 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.
- J.E. Cohen, D.N. Schittler, D.G. Raffaelli, and D.C. Reuman. Food webs are more than the sum of their tritrophic parts. *Proceedings of the National Academy of Sciences of the United States of America*, 106(52):22335–22340, 2009.
- S.N. de Visser, B.P. Freymann, and H. Olff. The serengeti food web: empirical quantification and analysis of topological changes under increasing human impact. *Journal of Animal Ecology*, 80(2):484–494, 2011. doi: 10.1111/j.1365-2656.2010.01787.x.
- 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.
- D. Gilljam, A. Thierry, F.K. Edwards, D. Figueroa, A.T. Ibbotson, J. Iwan Jones, R.B. Lauridsen, O.L. Petchey, G. Woodward, and B. Ebenman. Seeing double: size-based and taxonomic views of food web structure. *Advances In Ecological Research*, 45:67–133, 2011. doi: 10.1016/B978-0-12-386475-8.00003-4.
- 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.
- D.C. Reuman, C. Mulder, D. Raffaelli, and J.E. Cohen. Three allometric relations of population density to body mass: theoretical integration and empirical tests in 149 food webs. *Ecology Letters*, 11(11):1216–1228, 2008. doi: 10.1111/j.1461-0248.2008.01236.x.

- J.O. Riede, U. Brose, B. Ebenman, U. Jacob, R. Thompson, C.R. Townsend, and T. Jonsson. Stepping in elton's footprints: a general scaling model for body masses and trophic levels across ecosystems. *Ecology Letters*, 14:169–178, 2011. doi: 10.1111/j.1461-0248.2010.01568.x.
- 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.