enaR: Ecological Network Analysis with R

Matthew K. Lau[*]
Pawandeep Singh[†]
Stuart R. Borrett[‡]

Last updated - 12 February 2017

Abstract

Ecological Network Analysis (ENA) provides a framework for investigating the structure, function and dynamics of ecological systems, primarily ecosystem models that trace physically conserved units. This paper documents the *enaR* package, which collects synthesizes the core ENA functions, including those developed by the Ulanowicz and Patten schools. Further, the package connects users to additional network analysis tools available in **R** and beyond. This document details how to use the primary functions for the analysis of single models. In addition, we also demonstrate a key strength of this package, which is that it enables a user to perform simultaneous, synthetic analysis of multiple ecosystem models.

Introduction

Network models have provided an in-road to a variety of complex systems (Watts and Strogatz 1998; Newman 2001; Barabási 2012; Newman, Barabási, and Watts 2006; Wasserman and Faust 1994), and although the network approach has deep roots Newman, Barabási, and Watts (2006), its use has been expanding rapidly in a variety of disciplines including ecology (Borrett, Moody, and Edelmann 2014; Ings et al. 2009); and investigators are currently building a science of networks (National Research Council, Committee on Network Science for Army Applications 2006; Brandes et al. 2013). This is due in part to the flexibility of the core representation, its utility in answering relational questions, and its applicability to "Big Data" problems.

Ecosystem ecologists have developed and used network modeling and analysis for several decades (Borrett, Christian, and Ulanowicz 2012; Ulanowicz 1986; Fath and Patten 1999). The network models map transfers of thermodynamically conserved energy or matter (represented by weighted, directed graph edges) between nodes that represent species, groups of species, or non-living components (e.g., dead organic matter) of the ecosystem. These analyses, collectively known as Ecosystem Network Analysis (ENA), have been used in a variety of ways including to reveal the relative importance of indirect effects in ecosystems (Patten 1983; Higashi and Patten 1989; Salas and Borrett 2011) and their capacity to effectively transform the relations among organisms (Ulanowicz and Puccia 1990; Patten 1991; Fath and Patten 1998; Bondavalli and Ulanowicz 1999). From these applications a new theoretical understanding of ecosystems has emerged (Higashi and Burns 1991; Belgrano et al. 2005; Jørgensen et al. 2007). Recently, scientists have applied these methods to understand trophic dynamics in the Sylt-Romo Bight Baird, Asmus, and Asmus (2004),baird08_sylt, biogeochemical cycling in lakes and estuaries (Christian and Thomas 2003; Small, Sterner, and Finlay 2014; Hines et al. 2015), and urban sustainability (Zhang, Yang, and Fath 2010; Chen and Chen 2012).

Two major schools of ENA have developed (Scharler and Fath 2009). The first is based on Dr. Robert E. Ulanowicz's work with a strong focus on trophic dynamics and a use of information theory (Ulanowicz 1986, ulanowicz97, ulanowicz04). The second school has an environment focus and is built on the environ concept introduced by Dr. Bernard C. Patten (Patten et al. 1976; Patten 1978; Fath and Patten 1999). Patten's approach has been collectively referred to separately as *Network Environ Analysis*. At the core the two

[*]Harvard Forest, Harvard University
[†]Department of Biology and Marine Biology, University of North Carolina Wilmington
[‡]Department of Biology and Marine Biology, University of North Carolina Wilmington & Duke Network Analysis Center, Social Science Research Institute, Duke University

approaches are very similar; however, they make some different starting assumptions and follow independent yet braided development tracks.

Disparate software packages have been created to support ENA. Initially algorithms were developed and distributed as the DOS based NETWRK4 (Ulanowicz and Kay 1991), which is still available from www.cbl.umces.edu/~ulan/ntwk/network.html. Some of these algorithms were re-implemented in a Microsoft Excel based toolbox, WAND (Allesina and Bondavalli 2004). The popular Ecopath with Ecosim software that assists with model construction (Christensen and Walters 2004) also provides multiple ENA algorithms. The algorithms for flow analysis – one component of ENA – were collected into a stand-alone software tool (Latham II 2006). (Fath and Borrett 2006) published NEA.m that collects most of the Patten School ENA algorithms together in a single **MATLAB** (c) function. Similarly, the online tool EcoNet (Kazanci 2007) has made many of the ENA algorithms available in an easy access framework. Although these packages collectively provide access to a large set of powerful analytical tools, the fragmented distribution of the key algorithms among the software tools has inhibited the development of theory and the further implementation of important algorithms.

The *enaR* package brings together the ENA algorithms into one common software framework that is readily available and extensible. The package is written in the **R** language, which is free and open-source. Due largely to this, **R** is now one of the most widely used analytical programming languages in the biological sciences. *enaR* builds on existing **R** packages for network analysis. For example, it uses the *network* data structure developed by (Butts 2008a) and the network analysis tools built into the *network*, *sna* (social network analysis) (Butts 2008b), and *statnet* (Handcock et al. 2008) packages. While (Borrett and Lau 2014) introduced the *enaR* package, here we provide a richer documentation of the software and illustrate its use.

Getting Started

In this section we describe the data necessary for Ecological Network Analysis and show how to build the central network data object in **R** that contains the model data for subsequent analysis. To start, the current stable version can be installed from CRAN:

```
install.packages('enaR')
```

The development version can be installed from GitHub:

```
require(devtools)
install_github('SEELab/enaR',ref='develop')
```

You can now load the package:

```
require(enaR)
```

Ecosystem Network Model

ENA is applied to a network model of energy–matter exchanges among system components. The system is modeled as a set of n compartments or nodes that represent species, species-complexes (i.e., trophic guilds or functional groups), or non-living components of the system in which energy–matter is stored. Nodes are connected by L observed fluxes, termed directed edges or links. This analysis requires an estimate of the energy–matter flowing from node i to j over a given period, $\mathbf{F}_{n\times n} = [f_{ij}]$, $i, j = 1, 2, \ldots, n$. These fluxes can be generated by any process such as feeding (like a food web), excretion, and death. As ecosystems are thermodynamically open, there must also be energy or matter inputs into the system $\mathbf{z}_{1\times n} = [z_i]$, and output losses from the system $\mathbf{y}_{1\times n} = [y_i]$. While the Patten School treats all outputs the same, the Ulanowicz School

typically partitions outputs into respiration $\mathbf{r}_{1\times n} = [r_i]$ and export $\mathbf{e}_{1\times n} = [e_i]$ to account for differences in energetic quality. Note that $y_i = r_i + e_i, \forall i$. Some analyses also require the amount of energy–matter stored in each node (e.g., biomass), $\mathbf{X}_{1\times n} = [x_i]$. The final required information is a categorization of each node as living or not, which is essential for algorithms from the Ulanowicz School. For our implementation, we have created a logical vector $\mathbf{Living}_{1\times n}$ that indicates whether the i^{th} node is living (TRUE) or not (FALSE). This obviates the need to order the nodes in a specific way (i.e., living before non-living). Together, the model data $\mathcal{M}$ can be summarized as $\mathcal{M} = \{\mathbf{F}, \mathbf{z}, \mathbf{e}, \mathbf{r}, \mathbf{X}, \mathbf{Living}\}$.

Notice the row-to-column orientation of the flow matrix: **F**. This is consistent with the Ulanowicz School of network analysis, as well as the orientation commonly used in Social Network Analysis and used in the *statnet* packages. However, this is the opposite orientation typically used in the Patten School of analysis that conceptually builds from a system of differential equations and thus uses the column-to-row orientation common in this area of mathematics. Even though the difference is only a matrix transpose, this single difference may be the source of much confusion in the literature and frustration on the part of users. We have selected to use row-to-column orientation for our primary data structure, as it is the dominant form across network analytics as evidenced by it use in the *statnet* packages. The package algorithms also return the results in the row-to-column orientation by default; however, we have built in functionality with get.orient and set.orient that allows users to return output in the Patten School row-to-column orientation (see the Orientation Section for details).

Model Construction

There are multiple methods for constructing ecosystem network models and tools for assisting with this process (Fath et al. 2007). One approach is to construct a dynamic, processes-based, mathematical model of the system typically using ordinary differential equations. For example, the EcoPath with EcoSim (Christensen and Pauly 1992; Christensen 1995) software assists scientists with constructing food-web focused ecosystem models using an underlying bioenergetic approach. Alternatively, (Ulanowicz 1986) has called for a more phenomenologial approach to the model construction. This modeling process starts with a conceptual network model of the system and then the node and edge weights are estimated directly from observations. Its phenomenologial in the sense that it focuses on what the flows are, rather than the forms of the mechanistic processes that generate the flows. As this approach is essentially an inverse problem, some have developed inverse linear modeling methods to assist with inferring the network weights from data (Vézina and Platt 1988; Oevelen et al. 2010). The *limSolve* **R** pacakge can assist with this modeling approach (Soetaert, Van den Meersche, and Oevelen 2009). (Ulanowicz and Scharler 2008) also introduced two least-inference algorithms to assist with this kind of model constuction. These methods focus on constructing models to represent specific empirical systems. Algorithms also exist for constructing simulated ecosystems, including (Fath 2004) Cyber Models that use a community assembly type approach. Currently, the *enaR* software focuses on the analysis of network models and assumes that the user has a network model to be analyzed.

Network Data Class

The *enaR* package stores the model data in the **network** class defined in the *network* package (Butts 2008a). In this software, a complete ecosystem network model description includes:

- **F** is the flow matrix, oriented row-to-column
- **z** a vector of inputs
- **r** a vector of respirations
- **e** a vector of exports
- **y** a vector of outputs, which are respirations plus exports
- **X** a vector ofbiomass or storage values
- **Living** = logical vector indicating if the node is living (TRUE) or non-living (FALSE)

Building a Network Object

At present, *enaR* assumes that the user has a model constructed. Thus, the frist task is to get the model into the softward. One way to to this is to assemble the necessary data elements and then use the pack function to create the network data object. Here is an example of doing this for the generic hypothetical ecosystem model shown in the following Figure (modified from Borrett, Whipple, and Patten (2010)).

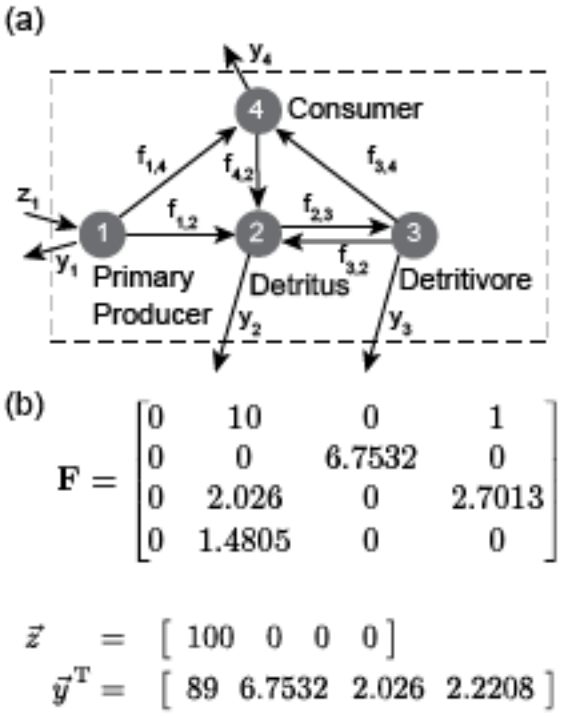


Figure 1:

```
## Generate the flow matrix
flow.mat <- matrix( c(0,0,0,0,
                      10,0,2.026,1.4805,
                      0,6.7532,0,
                      0,1,0,2.7013,0), ncol = 4)

## Name the nodes
rownames(flow.mat) <- colnames(flow.mat) <- c("Primary Producer", "Detritus",
                                              "Detritivore", "Consumer")

## Generate the inputs
inputs <- c(100, 0, 0, 0)
## Generate the exports
exports <- c(89, 6.7532, 2.026, 2.2208)
## "Pack" the model into a network object
fake.model <- pack(flow = flow.mat,
                   input = inputs,
                   export = exports,
                   living = c(TRUE,FALSE,TRUE,TRUE))
```

```
## [1] "respiration" "storage"
```

```
## Warning in pack(flow = flow.mat, input = inputs, export = exports, living =
## c(TRUE, : Missing model components: respiration, storage
```

When we pack this model, we receive a warning that reminds us that the model we have specified is missing the resipration and storage components. This is not an error, as we did not specify these components. With the information we have, we can still complete many of the analyses collected in *enaR*; however, some of them will not work without the requried information (i.e., enaStorage). Individual *enaR* functions check to ensure the required information is present in the model before they are applied.

We can take a closer look at the network data object as follows:

```
## The model network object contents
fake.model
```

```
##  Network attributes:
##   vertices = 4
##   directed = TRUE
##   hyper = FALSE
##   loops = TRUE
##   multiple = FALSE
##   bipartite = FALSE
##   balanced = TRUE
##   total edges= 6
##     missing edges= 0
##     non-missing edges= 6
##
##  Vertex attribute names:
##     export input living output respiration storage vertex.names
##
##  Edge attribute names:
##     flow
```

These results tell the user what the software has already inferred about the network from the initial data. The network data object devides these initial properties into whole network attributes, vertex attributes, and edge attributes. At the network level, the network has 4 vertices, it is a directed network, it is not a hypergraph, it can contain self-loops (set by the pack function), it is not a bipartite matrix, it is balanced (inputs = outputs), and it has 6 edges. The Vertecies (nodes) have a set of attributes including the values for export, input, living, output, respiration, storage, and vertex.names. Finally, the edge attributes are currenlty limited to the flow weights.

The individual components can be extracted from the data object using the form specified in the *network* package (network vignette).

For example, we can extract specific network attirbutes as follows:

```
# is the network directed?
fake.model%n%"directed"
```

```
## [1] TRUE
```

```
# how many nodes are in the network?
fake.model%n%"n"
```

```
## [1] 4
```

```
# alternatively, we can use a different network package function to find the number of nodes
network.size(fake.model)
```

```
## [1] 4
```

Similarily, we can pull out "vertex" (i.e. node) attributes as follows:

```
fake.model%v%'output'
```

```
## [1] 89.0000  6.7532  2.0260  2.2208
```

```
fake.model%v%'input'
```

```
## [1] 100   0   0   0
```

```
fake.model%v%'living'
```

```
## [1]  TRUE FALSE  TRUE  TRUE
```

The network flows are stored as edge weights in the network object, which lets users fully manipulate the network object with the network functions. The flow matrix can be extracted from the object with:

```
as.matrix(fake.model, attrname="flow")
```

```
##                   Primary Producer Detritus Detritivore Consumer
## Primary Producer                 0  10.0000      0.0000   1.0000
## Detritus                         0   0.0000      6.7532   0.0000
## Detritivore                      0   2.0260      0.0000   2.7013
## Consumer                         0   1.4805      0.0000   0.0000
```

There are times that it is useful to extract all of the ecosystem model data elements from the network data object. This can be accomplished using the unpack function. The unpack output is as follows:

```
unpack(fake.model)
```

```
## $F
##                   Primary Producer Detritus Detritivore Consumer
## Primary Producer                 0  10.0000      0.0000   1.0000
## Detritus                         0   0.0000      6.7532   0.0000
## Detritivore                      0   2.0260      0.0000   2.7013
## Consumer                         0   1.4805      0.0000   0.0000
##
## $z
```

```
## [1] 100   0   0   0
##
## $r
## [1] 0 0 0 0
##
## $e
## [1] 89.0000  6.7532  2.0260  2.2208
##
## $y
## [1] 89.0000  6.7532  2.0260  2.2208
##
## $X
## [1] NA NA NA NA
##
## $living
## [1]  TRUE FALSE  TRUE  TRUE
```

Since we did not specify the storage values when we used pack, the stroage values were set to NA values. In contrast, pack generated zero values for the respiration values. The function assumes that if you don't specify respiration (or export) then the values must be zero. The output values are generated by adding the respiration and export values together. Although the package is designed to help users navigate missing data issues, you should check that you are providing the appropriate input for a given function.

Lastly, once a model object is created, it might be necessary to adjust the ordering of the nodes. This can be done easily using the netOrder function:

```
netOrder(x = fake.model, order = c(1, 3, 2, 4))
```

```
##  Network attributes:
##   vertices = 4
##   directed = TRUE
##   hyper = FALSE
##   loops = TRUE
##   multiple = FALSE
##   bipartite = FALSE
##   balanced = TRUE
##   total edges= 6
##     missing edges= 0
##     non-missing edges= 6
##
##  Vertex attribute names:
##     export input living output respiration storage vertex.names
##
##  Edge attribute names:
##     flow
```

Model Library

enaR includes a library of 100 empirically-based, previously published ecosystem models that can be categorized into two general classes: trophic and biogeochemical cycling (Christian et al. 1996; Baird, Asmus, and Asmus 2008; Borrett, Whipple, and Patten 2010; Borrett, Hines, and Carter 2015). First, 58 of the models are trophically-based models with food webs at their core and 42 models focused on biogeochemical cycling in ecosystems (Network Model Information Table). In summary, these models were originally published for a

number of different types of ecosystems, though predominantly aquatic, by a number of author teams. Models in the library range in size from 4 nodes to 125 nodes with connectance values ranging from 7% to 45%.

This collection of models overlaps with other extant data sets. For example, twenty-four of the models are included in the set of forty-eight models compiled and distributed by Dr. Ulanowicz (http://www.cbl.umces.edu/~ulan/ntwk/network.html). All 50 of the models analyzed by (Borrett and Salas 2010) and (Salas and Borrett 2011) and the 45 models analyzed in (Borrett 2013) are included in this model library.

The trophic models are grouped as the troModels object and the biogeochemically-based models are available as the bgcModels object. Both data objects return a list of the model network objects. To use these models simply use the **R** *base* data function. This will load the models into the working memory as a named list of network objects:

```
## Import the model sets
data(bgcModels)
data(troModels)
## Find the names of the first few models
head(names(bgcModels))
```

```
## [1] "Hubbard Brook (Ca)(Waide)"     "Hardwood Forest, NH (Ca)"
## [3] "Duglas Fir Forest, WA (Ca)"    "Duglas Fir Forest, WA (K)"
## [5] "Puerto Rican Rain Forest (Ca)" "Puerto Rican Rain Forest (K)"
```

```
head(names(troModels))
```

```
## [1] "Marine Coprophagy (oyster)" "Lake Findley "
## [3] "Mirror Lake"                "Lake Wingra"
## [5] "Marion Lake"                "Cone Springs"
```

```
## Isolate a single model
x <- troModels[[1]]
x <- troModels$"Marine Coprophagy (oyster)"
## Check out the model
summary(x)
```

```
## Network attributes:
##   vertices = 4
##   directed = TRUE
##   hyper = FALSE
##   loops = TRUE
##   multiple = FALSE
##   bipartite = FALSE
##   balanced = TRUE
##  total edges = 4
##    missing edges = 0
##    non-missing edges = 4
##  density = 0.25
##
## Vertex attributes:
##
##  export:
##    logical valued attribute
##    attribute summary:
```

```
##    Mode    NA's
## logical       4
##
##  input:
##    numeric valued attribute
##    attribute summary:
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    0.00    0.00   62.05   94.90  157.00  255.50
##
##  living:
##    logical valued attribute
##    attribute summary:
##    Mode   FALSE    TRUE    NA's
## logical       2       2       0
##
##  output:
##    numeric valued attribute
##    attribute summary:
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    6.60   21.67   64.45   94.90  137.70  244.10
##
##  respiration:
##    numeric valued attribute
##    attribute summary:
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    6.60   21.67   64.45   94.90  137.70  244.10
##
##  storage:
##    numeric valued attribute
##    attribute summary:
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##       1       1       1       1       1       1
##   vertex.names:
##    character valued attribute
##    4 valid vertex names
##
## Edge attributes:
##
##  flow:
##    numeric valued attribute
##    attribute summary:
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   15.30   20.25   37.40   42.42   59.58   79.60
##
## Network adjacency matrix:
##                           SHRIMP BENTHIC ORGANISMS SHRIMP FECES & BACTERIA
## SHRIMP                         0                 0                       1
## BENTHIC ORGANISMS              0                 0                       0
## SHRIMP FECES & BACTERIA        0                 1                       0
## BENTHIC FECES & BACTERIA       0                 1                       0
##                           BENTHIC FECES & BACTERIA
## SHRIMP                                           0
## BENTHIC ORGANISMS                                1
## SHRIMP FECES & BACTERIA                          0
```

```
## BENTHIC FECES & BACTERIA                    0
```

Network Model Information

Table 1: Trophic (n=58) and biogeochemical ecosystem (n=42) networks included in the *enaR* model library. n is the number of nodes in the network model, $C = L/n^2$ is the model connectance when L is the number of direct links or energy–matter transfers , $Input = a\sum z_i$ is the total amount of energy–matter flowing into the system, $TST = \sum\sum f_{ij} + \sum z_i$ is the total system throughflow, and FCI is the Finn Cycling Index (Finn 1980). Flow based network statistics (*Input*, *TST*, and *FCI*) were calculated after models were balanced using the AVG2 algorithm.

Model	Type	Units	n	Reference
Marine Coprophagy (oyster)	Tro	kcal mˆ-2 yrˆ-1	4	Haven and Morales-Alamo (1966)
Lake Fndley	Tro	gC mˆ-2 yrˆ-1	4	Richey et al. (1978)
MirrorLake	Tro	gC mˆ-2 yrˆ-1	5	Richey et al. (1978)
Lake Wngra	Tro	gC mˆ-2 yrˆ-1	5	Richey et al. (1978)
MarionLake	Tro	gC mˆ-2 yrˆ-1	5	Richey et al. (1978)
Cone Srings	Tro	kcal mˆ-2 yrˆ-1	5	Tilly (1968)
SilverSprings	Tro	kcal mˆ-2 yrˆ-1	5	Odum (1957)
Englis Channel	Tro	kcal mˆ-2 yrˆ-1	6	Brylinsky (1972)
OysterReef	Tro	kcal mˆ-2 yrˆ-1	6	Dame and Patten (1981)
Baie de Somme	Tro	mgC mˆ-2 dˆ-1	9	Rybarczyk et al. (2003)
Bothnin Bay	Tro	gC mˆ-2 yrˆ-1	12	Sandberg, Elmgren, and Wulff (2000)
Bothnin Sea	Tro	gC mˆ-2 yrˆ-1	12	Sandberg, Elmgren, and Wulff (2000)
Ythan Estuary	Tro	gC mˆ-2 yrˆ-1	13	Baird and Milne (1981)
Sundarban Mangrove (virgin)	Tro	kcal mˆ-2 yrˆ-1	14	Ray (2008)
Sundarban Mangrove (reclaimed)	Tro	kcal mˆ-2 yrˆ-1	14	Ray (2008)
Baltic Sea	Tro	mg C mˆ-2 dˆ-1	15	Baird, McGlade, and Ulanowicz (1991)
Ems Estuary	Tro	mg C mˆ-2 dˆ-1	15	Baird, McGlade, and Ulanowicz (1991)
Swartkops Estuary 15	Tro	mg C mˆ-2 dˆ-1	15	Baird, McGlade, and Ulanowicz (1991)
Southern Benguela Upwelling	Tro	mg C mˆ-2 dˆ-1	16	Baird, McGlade, and Ulanowicz (1991)
Peruvian Upwelling	Tro	mg C mˆ-2 dˆ-1	16	Baird, McGlade, and Ulanowicz (1991)
Crystal River (control)	Tro	mg C mˆ-2 dˆ-1	21	Ulanowicz (1986)
Crystal River (thermal)	Tro	mg C mˆ-2 dˆ-1	21	Ulanowicz (1986)
Charca de Maspalomas Lagoon	Tro	mg C mˆ-2 dˆ-1	21	Almunia et al. (1999)
Northern Benguela Upwelling	Tro	mg C mˆ-2 dˆ-1	24	Heymans and Baird (2000)
Swartkops Estuary	Tro	mg C mˆ-2 dˆ-1	25	Scharler and Baird (2005)
Sunday Estuary	Tro	mg C mˆ-2 dˆ-1	25	Scharler and Baird (2005)
Kromme Estuary	Tro	mg C mˆ-2 dˆ-1	25	Scharler and Baird (2005)
Okefenokee Swamp	Tro	g dw mˆ-2 yˆ-1	26	Whipple and Patten (1993)
Neuse Estuary (early summer 1997)	Tro	mg C mˆ-2 dˆ-1	30	Baird et al. (2004)
Neuse Estuary (late summer 1997)	Tro	mg C mˆ-2 dˆ-1	30	Baird et al. (2004)
Neuse Estuary (early summer 1998)	Tro	mg C mˆ-2 dˆ-1	30	Baird et al. (2004)
Neuse Estuary (late summer 1998)	Tro	mg C mˆ-2 dˆ-1	30	Baird et al. (2004)
Gulf of Maine	Tro	g ww mˆ-2 yrˆ-1	31	Link et al. (2008)
Georges Bank	Tro	g ww mˆ-2 yrˆ-1	31	Link et al. (2008)
Middle Atlantic Bight	Tro	g ww mˆ-2 yrˆ-1	32	Link et al. (2008)
Narragansett Bay	Tro	mgC mˆ-2 yrˆ-1	32	Monaco and Ulanowicz (1997)
Southern New England Bight	Tro	g ww mˆ-2 yrˆ-1	33	Link et al. (2008)
Chesapeake Bay	Tro	mg C mˆ-2 yrˆ-1	36	Baird and Ulanowicz (1989)

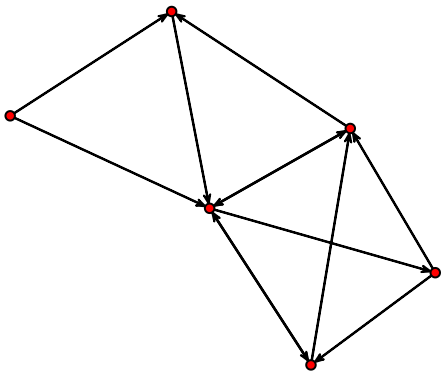
Model	Type	Units	n	Reference
Mondego Estuary (Zostera Meadows)	Tro	g AFDW mˆ2 yrˆ-1	43	Patrício and Marques (2006)
St. Marks Seagrass site 1 (Jan.)	Tro	mg C mˆ-2 dˆ-1	51	Baird, Luczkovich, and Christian (1998)
St. Marks Seagrass site 1 (Feb.)	Tro	mg C mˆ-2 dˆ-1	51	Baird, Luczkovich, and Christian (1998)
St. Marks Seagrass site 2 (Jan.)	Tro	mg C mˆ-2 dˆ-1	51	Baird, Luczkovich, and Christian (1998)
St. Marks Seagrass site 2 (Feb.)	Tro	mg C mˆ-2 dˆ-1	51	Baird, Luczkovich, and Christian (1998)
St. Marks Seagrass site 3 (Jan.)	Tro	mg C mˆ-2 dˆ-1	51	Baird, Luczkovich, and Christian (1998)
St. Marks Seagrass site 4 (Feb.)	Tro	mg C mˆ-2 dˆ-1	51	Baird, Luczkovich, and Christian (1998)
Sylt-Romo Bight	Tro	mg C mˆ-2 dˆ-1	59	Baird, Asmus, and Asmus (2004)
Graminoids (wet)	Tro	g C mˆ-2 yrˆ-1	66	Ulanowicz et al. (2000)
Graminoids (dry)	Tro	g C mˆ-2 yrˆ-1	66	Ulanowicz et al. (2000)
Cypress (wet)	Tro	g C mˆ-2 yrˆ-1	68	Ulanowicz, Bondavalli, and Egnotovich (1
Cypress (dry)	Tro	g C mˆ-2 yrˆ-1	68	Ulanowicz, Bondavalli, and Egnotovich (1
Lake Oneida (pre-ZM)	Tro	g C mˆ-2 yrˆ-1	74	Miehls, Mason, et al. (2009b)
Lake Oneida (post-ZM)	Tro	g C mˆ-2 yrˆ-1	76	Miehls, Mason, et al. (2009b)
Bay of Quinte (pre-ZM)	Tro	g C mˆ-2 yrˆ-1	74	Miehls, Mason, et al. (2009a)
Bay of Quinte (post-ZM)	Tro	g C mˆ-2 yrˆ-1	80	Miehls, Mason, et al. (2009a)
Mangroves (wet)	Tro	g C mˆ-2 yrˆ-1	94	Ulanowicz et al. (1999)
Mangroves (dry)	Tro	g C mˆ-2 yrˆ-1	94	Ulanowicz et al. (1999)
Florida Bay (wet)	Tro	mg C mˆ-2 yrˆ-1	125	Ulanowicz, Bondavalli, and Egnotovich (1
Florida Bay (dry)	Tro	mg C mˆ-2 yrˆ-1	125	Ulanowicz, Bondavalli, and Egnotovich (1
Hubbard Brook (Waide)	BGC	kg Ca Haˆ-1 yrˆ-1	4	Waide et al. (1974)
Hardwood Forest NH	BGC	kg Ca Haˆ-1 yrˆ-1	4	Jordan, Kline, and Sasscer (1972)
Douglas Fir Forest WA	BGC	kg Ca Haˆ-1 yrˆ-1	4	Jordan, Kline, and Sasscer (1972)
Douglas Fir Forest WA	BGC	kg K Haˆ-1 yrˆ-1	4	Jordan, Kline, and Sasscer (1972)
Puerto Rican Rain Forest	BGC	kg Ca Haˆ-1 yrˆ-1	4	Jordan, Kline, and Sasscer (1972)
Puerto Rican Rain Forest	BGC	kg K Haˆ-1 yrˆ-1	4	Jordan, Kline, and Sasscer (1972)
Puerto Rican Rain Forest	BGC	kg Mg Haˆ-1 yrˆ-1	4	Jordan, Kline, and Sasscer (1972)
Puerto Rican Rain Forest	BGC	kg Cu Haˆ-1 yrˆ-1	4	Jordan, Kline, and Sasscer (1972)
Puerto Rican Rain Forest	BGC	kg Fe Haˆ-1 yrˆ-1	4	Jordan, Kline, and Sasscer (1972)
Puerto Rican Rain Forest	BGC	kg Mn Haˆ-1 yrˆ-1	4	Jordan, Kline, and Sasscer (1972)
Puerto Rican Rain Forest	BGC	kg Na Haˆ-1 yrˆ-1	4	Jordan, Kline, and Sasscer (1972)
Puerto Rican Rain Forest	BGC	kg Sr Haˆ-1 yrˆ-1	4	Jordan, Kline, and Sasscer (1972)
Tropical Rain Forest	BGC	g N mˆ-2 dˆ-1	5	Edmisten (1970)
Neuse River Estuary (AVG)	BGC	mmol N mˆ-2 seasonˆ-1	7	Christian and Thomas (2003)
Neuse River Estuary (Spring 1985)	BGC	mmol N mˆ-2 seasonˆ-1	7	Christian and Thomas (2003)
Neuse River Estuary (Summer 1985)	BGC	mmol N mˆ-2 seasonˆ-1	7	Christian and Thomas (2003)
Neuse River Estuary Fall 1985)	BGC	mmol N mˆ-2 seasonˆ-1	7	Christian and Thomas (2003)
Neuse River Estuary Winter 1986)	BGC	mmol N mˆ-2 seasonˆ-1	7	Christian and Thomas (2003)
Neuse River Estuary (Spring 1986)	BGC	mmol N mˆ-2 seasonˆ-1	7	Christian and Thomas (2003)
Neuse River Estuary (Summer 1986)	BGC	mmol N mˆ-2 seasonˆ-1	7	Christian and Thomas (2003)
Neuse River Estuary (Fall 1986)	BGC	mmol N mˆ-2 seasonˆ-1	7	Christian and Thomas (2003)
Neuse River Estuary (Winter 1987)	BGC	mmol N mˆ-2 seasonˆ-1	7	Christian and Thomas (2003)
Neuse River Estuary (Spring 1987)	BGC	mmol N mˆ-2 seasonˆ-1	7	Christian and Thomas (2003)
Neuse River Estuary (Summer 1987)	BGC	mmol N mˆ-2 seasonˆ-1	7	Christian and Thomas (2003)
Neuse River Estuary (Fall 1987)	BGC	mmol N mˆ-2 seasonˆ-1	7	Christian and Thomas (2003)
Neuse River Estuary (Winter 1988)	BGC	mmol N mˆ-2 seasonˆ-1	7	Christian and Thomas (2003)
Neuse River Estuary (Spring 1988)	BGC	mmol N mˆ-2 seasonˆ-1	7	Christian and Thomas (2003)
Neuse River Estuary (Summer 1988)	BGC	mmol N mˆ-2 seasonˆ-1	7	Christian and Thomas (2003)
Neuse River Estuary (Fall 1988)	BGC	mmol N mˆ-2 seasonˆ-1	7	Christian and Thomas (2003)
Neuse River Estuary (Winter 1989)	BGC	mmol N mˆ-2 seasonˆ-1	7	Christian and Thomas (2003)
Cape Fear River Estuary (Oligohaline)	BGC	nmol N cmˆ-3 dˆ-1	8	Hines et al. (2012)
Cape Fear River Estuary (Polyhaline)	BGC	nmol N cmˆ-3 dˆ-1	8	Hines et al. (2015)

Model	Type	Units	n	Reference
Lake Lanier (AVG)	BGC	mg P m^-2 day^-1	11	Borrett and Osidele (2007)
Baltic Sea	BGC	mg N m^-3 day^-1	16	Hinrichsen and Wulff (1998)
Chesapeake Bay	BGC	mg N m^-2 yr^-1	36	Baird, Ulanowicz, and Boynton (1995)
Chesapeake Bay	BGC	mg P m^-2 yr^-1	36	Ulanowicz and Baird (1999)
Chesapeake Bay (Winter)	BGC	mg P m^-2 season^-1	36	Ulanowicz and Baird (1999)
Chesapeake Bay (Spring)	BGC	mg P m^-2 season^-1	36	Ulanowicz and Baird (1999)
Chesapeake Bay (Summer)	BGC	mg P m^-2 season^-1	36	Ulanowicz and Baird (1999)
Chesapeake Bay (Fall)	BGC	mg P m^-2 season^-1	36	Ulanowicz and Baird (1999)
Sylt-Romo Bight	BGC	mg N m^-2 yr^-1	59	Baird, Asmus, and Asmus (2008)
Sylt-Romo Bight	BGC	mg P m^-2 yr^-1	59	Baird, Asmus, and Asmus (2008)

Network Visualization

Network plots are a useful tool to visualize patterns in complex datasets. Here, we present one example of how to plot a network model using the plot tools in the *network* package. The figure scaling may need to be adjusted depending on computer and the graphics devices. Also, note that the graph only shows internal system flows.

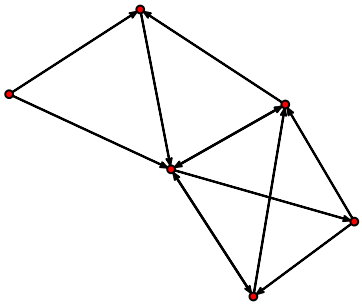
```
## Load data
data(oyster)
m <- oyster
## Set the random seed to control plot output
set.seed(2)
## Plot network data object (uses plot.network)
plot(m)
```

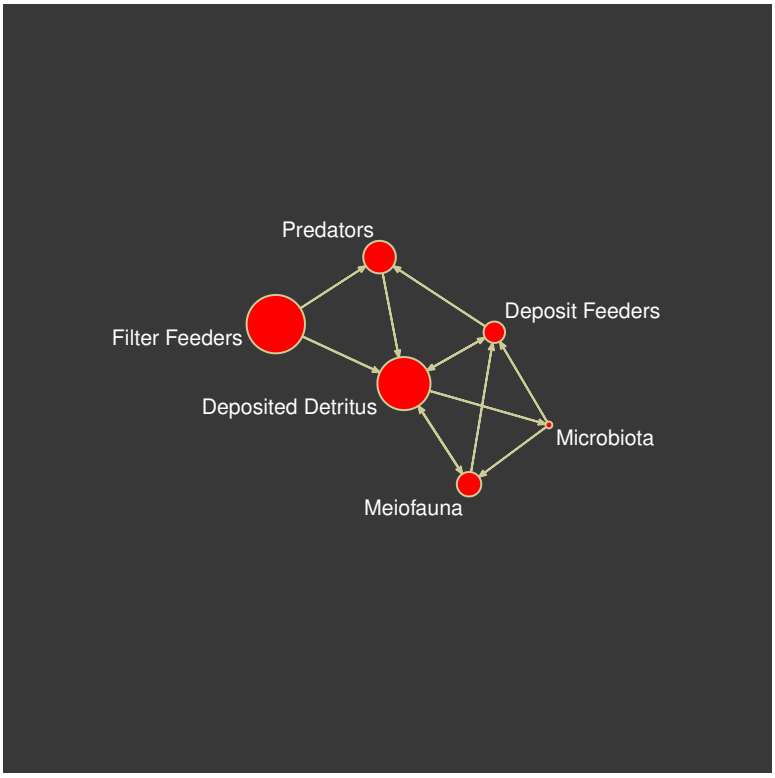


We can use the powerful graphics capabilities of **R** to make a fancier plot of the same data.

```
## Set colors to use
my.col <- c('red','yellow',rgb(204,204,153,maxColorValue=255),'grey22')
## Extract flow information for later use.
F <- as.matrix(m,attrname='flow')
## Get indices of positive flows
f <- which(F!=0, arr.ind=T)
opar <- par(las=1,bg=my.col[4],xpd=TRUE,mai=c(1.02, 0.62, 0.82, 0.42))
## Set the random seed to control plot output
set.seed(2)
plot(m,
```

```
## Scale nodes with storage
     vertex.cex=log(m%v%'storage'),
## Add node labels
     label= m%v%'vertex.names',
     boxed.labels=FALSE,
     label.cex=0.65,
## Make rounded nodes
     vertex.sides=45,
## Scale arrows to flow magnitude
     edge.lwd=log10(abs(F[f])),
     edge.col=my.col[3],
     vertex.col=my.col[1],
     label.col='white',
     vertex.border = my.col[3],
     vertex.lty = 1,
     xlim=c(-4,1),ylim=c(-2,-2))
## Lastly, remove changes to the plotting parameters
rm(opar)
```





Two networks for the Oyster Reef model (Dame and Patten 1981) showing a simple (left) and more elaborate (right) implementation of the network plotting function.

Data Input: Reading Common Data File Formats

Several software packages exist in the literature for running ENA. We have written functions to read in a few of the more common data formats used by them to help *enaR* users to import models formatted for these other packages. Example data files can be found in the data folder here: https://github.com/SEELab/enaR_development.

SCOR

The read.scor function reads in data stored in the SCOR format specified by (Ulanowicz and Kay 1991) that is the input to the NETWRK4 programs. This function can be run as follows.

```
scor.model <- readLines('../../data/oyster.dat')
m <- read.scor(scor.model,from.file=FALSE)
```

This constructs the network data object from the SCOR file that stores the ecosystem model data for an oyster reef model (Dame and Patten 1981). The individual model elements are

```
unpack(m)
```

```
## $F
##                    Filter Feeders Microbiota Meiofauna Deposit Feeders
## Filter Feeders                  0     0.0000    0.0000          0.0000
## Microbiota                      0     0.0000    1.2060          1.2060
## Meiofauna                       0     0.0000    0.0000          0.6609
## Deposit Feeders                 0     0.0000    0.0000          0.0000
## Predators                       0     0.0000    0.0000          0.0000
## Deposited Detritus              0     8.1721    7.2745          0.6431
##                    Predators Deposited Detritus
## Filter Feeders        0.5135            15.7910
## Microbiota            0.0000             0.0000
## Meiofauna             0.0000             4.2403
## Deposit Feeders       0.1721             1.9076
## Predators             0.0000             0.3262
## Deposited Detritus    0.0000             0.0000
##
## $z
## [1] 41.47  0.00  0.00  0.00  0.00  0.00
##
## $r
## [1] 25.1650  5.7600  3.5794  0.4303  0.3594  6.1759
##
## $e
## [1] 0 0 0 0 0 0
##
## $y
## [1] 25.1650  5.7600  3.5794  0.4303  0.3594  6.1759
##
```

```
## $X
## [1] 2000.0000    2.4121   24.1210   16.2740   69.2370 1000.0000
##
## $living
## [1]  TRUE  TRUE  TRUE  TRUE  TRUE FALSE
```

This same data is stored as a network data object that is distributed with this package, which can be accessed as:

```
data(oyster)
m <- oyster
```

WAND

In part to make ENA more accessible to biologists, (Allesina and Bondavalli 2004) recoded some of Ulanowicz's NETWRK4 algorithms into a Microsoft Excel based tool called WAND. For this tool, the model data is stored as a separate Excel file with two worksheets. The first contains many of the node attributes and the second contains the flow matrix. The read.wand function will create an **R** network data object from a WAND model file.

```
m <- read.wand('../../data/MDmar02_WAND.xls')
```

This code creates a network data object for *enaR* from the WAND formatted Mdloti ecosystem model data (Scharler 2012). This data is courtesy of U.M. Scharler.

NEA

For their Matlab function to perform network environ analysis (Patten School), (Fath and Borrett 2006) packaged the model flows, inputs, outputs, and storage values into what they called a system matrix

$$\mathbf{S} = \left[\begin{array}{ccc} \mathbf{F} & \vec{z} & \vec{X} \\ \vec{y} & 0 & 0 \end{array} \right]_{(n+1)\times(n+2)}. \tag{1}$$

Flows in the system matrix are oriented from column to row.

The *enaR* function read.nea reads in data with this format stored as a comma separated value file (CSV). The function write.nea() will write any network model to a CSV file with this format.

While convenient, this data format does not enable inclusion of the full range of model information included in the *enaR* network data object. This format does not partition outputs into exports and respiration values, nor does it identify the node labels or their living status. This missing information will prevent the use of some *enaR* functions.

Here is an example of using these functions:

```
data(oyster)

## Write oyster reef model to a CSV file
write.nea(oyster, file.name="oyster.csv")
```

```
##         [,1]  [,2]   [,3]   [,4]   [,5]   [,6]  [,7]      [,8]
## [1,]  0.0000 0.000 0.0000 0.0000 0.0000 0.0000 41.47 2000.0000
## [2,]  0.0000 0.000 0.0000 0.0000 0.0000 8.1721  0.00    2.4121
## [3,]  0.0000 1.206 0.0000 0.0000 0.0000 7.2745  0.00   24.1210
## [4,]  0.0000 1.206 0.6609 0.0000 0.0000 0.6431  0.00   16.2740
## [5,]  0.5135 0.000 0.0000 0.1721 0.0000 0.0000  0.00   69.2370
## [6,] 15.7910 0.000 4.2403 1.9076 0.3262 0.0000  0.00 1000.0000
## [7,] 25.1650 5.760 3.5794 0.4303 0.3594 6.1759  0.00    0.0000
```

```
## Read in oyster reef model data from NEA.m formatted CSV file
m <- read.nea("oyster.csv")
```

```
## [1] "export" "living"
```

```
## Warning in pack(flow = Flow, input = z, respiration = y, storage = X):
## Missing model components: export, living
```

```
## Again, this model object does NOT contain all
## of the information in the "oyster" data object.
```

ENAM

Another commonly used data format stores the necessary model data in a CSV or Excel formatted file. We include an example Excel file of the Mdloti estuary stored in this form ("MDMAR02.xlsx", courtesy of U. M. Scharler). This format has not been described technically in the literature nor has it been named. We refer to it as ENAM as it is the ENA model data stored primarily as a square matrix with several preliminary rows that include meta-data, the number of nodes, and number of living nodes (similar to SCOR). The data format is generally similar in concept, if not exact form, to the data system matrix used as the input to the NEA.m function (Fath and Borrett 2006). However, the ENAM format includes information on whether nodes are living and partitions output into respiration and exports.

Using an example data file, MDMAR02.xlsx, this data format can be read into the *enaR* package as:

```
  m <- read.enam('../../data/MDMAR02.xlsx')
```

The current read.enam function assumes the data are stored on the first worksheet of an Excel file. In the future, we expect to expand this function's capabilities to read the data from a CSV file.

Analyzing an Ecosystem Model

ENA is often applied to investigate the structure and function of a single ecosystem model. Here, we walk through an example of applying multiple ENA algorithms to the South Carolina oyster reef model (Dame and Patten 1981). The table below summarizes the main ENA algorithms encoded in *enaR*.

Table 2: Primary Ecosystem Network Analysis algorithms in *enaR*.

Analysis	Function.Name	School
Structure	enaStructure	foundational/Patten
Flow	enaFlow	foundational/Patten
Ascendency	enaAscendency	Ulanowicz

Analysis	Function.Name	School
Storage	enaStorage	Patten
Utility	enaUtility	Patten
Mixed Trophic Impacts	enaMTI	Ulanowicz
Control	enaControl	Patten
Environ	enaEnviron	Patten
Cycle Basis	enaCycle	Ulanowicz
Canonical Trophic Aggregation	enaTroAgg	Ulanowicz

Again, in this package results are reported in the row-to-column orientation by default – including the algorithms from the Patten school. Please see Orientation Section for how to change this default if needed.

Balancing to Steady-State

Many of the ENA functions assume that the network model is at steady-state (node inputs equal node outputs). Thus, this package has functions for (1) checking to see if the assumption is met and (2) automatically balancing the model so that input equal outputs.

To determine if the model is balanced and then balance it if necessary:

```
## Check to see if the model is balanced
ssCheck(fake.model)
```

```
## [1] TRUE
```

```
## To BALANCE a model if needed
fake.model <- balance(fake.model,method="AVG2")
```

```
## [1] BALANCED
```

The automated balancing routines include Input, Output, AVG, and AVG2 following Allesina and Bondavalli (2003). These authors compare the four alternative balancing algorithms and further discuss the implications of using automated procedures. AVG2 is the default algorithm becuase Allesina and Bondavalli (2003) found that it produced the smallest error in selected network metrics.

Unfortunatley, sometimes a single application of the balancing routines does not sufficiently balance the model. Thus, we include a function force.balance that runs the balancing algorithm selected multiple times until the flow model is balanced.

```
## To FORCE BALANCE a model if needed
fake.model <- force.balance(fake.model)
```

Caution is warranted when using these techniques, as they tend to alter all of the model flows. A more neuanced approach may be desired when the uncertainty in estimates of model fluxes are known.

Structural Network Analysis

Structural network analysis is common to many types of network analysis. The structural analyses applied here are largely based on those presented in *NEA.m* (Fath and Borrett 2006) following the Patten School. Output of the enaStructure function is summarized in following table.

Table 3: Resultant matrices and network statistics returned by th
enaStructure function in *enaR*.

Label	Description
A	$n \times n$ adjacency matrix
n	number of nodes
L	number of directed edges
C	connectance ($C = L/n^2$); the p
LD	Link Density (L/n)
ppr	estimated rate of pathway prol
lam1A	dominant eigenvalue of A
$\lambda_1(\mathbf{A})$ (i.e. the asymptotic rate of pathway proliferation) (Borrett, Fath, and Patten 2007)	
mlam1A	multiplicity of the dominant ei
rho	damping ratio (how quickly $[a_i$
R	distance of $\lambda_1(\mathbf{A})$ from the bul
d	difference between dominant ei
no.scc	number of strongly connected
no.scc.big	number of SCC with more tha
pscc	fraction of network nodes inclu

```
data(oyster)
St <- enaStructure(oyster)
```

```
## Node 1, Reach 1, Total 1
## Node 2, Reach 6, Total 7
## Node 3, Reach 6, Total 13
## Node 4, Reach 6, Total 19
## Node 5, Reach 6, Total 25
## Node 6, Reach 6, Total 31
```

```
attributes(St)
```

```
## $names
## [1] "A"  "ns"
```

```
St$ns
```

```
##      n  L         C LD      ppr    lam1A mlam1A      rho         R
## [1,] 6 12 0.3333333  2 2.147899 2.147899      1 2.147899 0.4655712
##             d no.scc no.scc.big      pscc
## [1,] 0.147899      2          1 0.8333333
```

The number of nodes, number of links, link density, and connectance (density) are common statistics used to describe networks like food webs (Martinez 1992; Dunne, Williams, and Martinez 2002; Eklöf and Ebenman 2006; Estrada 2007; Brandes and Erlebach 2005). The pathway proliferation rate quantifies if and how fast the number of pathways increases with path length in the network (Borrett and Patten 2003; Borrett, Fath, and Patten 2007). This rate is equivalent to the dominant eigenvalue of the adjacency matrix ($\lambda_1(A)$) if the network is comprised of a single strongly connected component (Borrett, Fath, and Patten 2007).

The structural network statistics for the oyster reef model shows that it has 6 nodes, a pathway proliferation rate of 2.14 (ppr), and that the model is comprised of two strongly connected components (no.scc) but that only one has more than one node (no.scc.big). Thus, 83% of the nodes are participating in a strongly connected component (pscc).

Flow Analysis

Flow analysis is one of the core ENA analyses for both the Ulanowicz and Patten Schools (Fath and Patten 1999; Latham II 2006; Fath and Borrett 2006; Schramski, Kazanci, and Tollner 2011). The *enaR* implementation enaFlow mostly follows the NEA.m function, with small updates (Borrett and Freeze 2011; Borrett, Freeze, and Salas 2011). Results returned by enaFlow are summarized in following table.

To validly apply flow analysis, the network model must meet two analytical assumptions. First, the model must trace a single, thermodynamically conserved currency, such as energy, carbon, or nitrogen. Second, the model must be at steady-state for many of the analyses.

Flow analysis has been used in a variety of ways. For example, (Finn 1980) used ENA flow analysis to compare the cycling of multiple nutrients through the Hubbard Brook Ecosystem, New Hampshire, USA, and (Oevelen et al. 2009) used the technique to show how different marine canyon conditions change the flow of carbon through the food webs in Nazar{'e} Canyon. Gattie et al. (2006) applied the analysis to characterize N cycling in the Neuse River Estuary (North Carolina, USA), and Zhang, Yang, and Fath (2010) used flow analysis to help assess the sustainability of the urban water metabolism of Beijing, China. Borrett (2013) showed that the throughflow vector T can be considered as a type of centrality measure that indicates the relative importance of each node to the generation of the total system throughflow or activity.

Table 4: Matrices and network statistics returned by the enaFlow function in *enaR*.

Code.Label	Description
Vectors & Matrices	
T	$n \times 1$ vector of node throughflows (M $L^{-2 \text{ or } -3}$ T^{-1})
G	output-oriented direct throughflow intensity matrix
GP	input-oriented direct throughflow intensity matrix
N	output-oriented integral throughflow intensity matrix
NP	input-oriented integral throughflow intensity matrix
Network Metrics	
Input	Total input boundary flow
TST	Total System ThroughFLOW
TSTp	Total System ThroughPUT
APL	Average Path Length (Finn 1976)
FCI	Finn Cycling Index (Finn 1980)
BFI	Boundary Flow Intensity (Borrett et al. 2006)
DFI	Direct Flow Intensity (Borrett et al. 2006)
IFI	Indirect Flow Intensity (Borrett et al. 2006)
ID.F	Ratio of Indirect to Direct Flow (Borrett and Freeze 2011; Borrett, Freeze, and Salas 2011)
ID.F.I	Input oriented ratio of indirect to direct flow intensity (Fath and Patten 1999)
IF.F.O	output oriented ratio of indirect to direct flow intensity (Fath and Patten 1999)
HMG.F.I	input oriented network homogenization to direct flow intensity
HMG.F.O	output oriented network homogenization to direct flow intensity
AMP.F.I	input oriented network amplification
AMP.F.O	output oriented network amplification
mode0.F	Boundary Flow (Higashi, Patten, and Burns 1993; Fath, Patten, and Choi 2001)
mode1.F	Internal First Passage Flow (Higashi, Patten, and Burns 1993; Fath, Patten, and Choi 2001)
mode2.F	Cycled Flow (Higashi, Patten, and Burns 1993; Fath, Patten, and Choi 2001)
mode3.F	Dissipative Equivalent to mode1.F (Fath, Patten, and Choi 2001)
mode4.F	Dissipative Equivalent to mode0.F (Fath, Patten, and Choi 2001)
Flow Diversity Metrics	
H	total flow diversity (entropy)
AMI	average mutual information

Code.Label	Description
Hr	residual mutual information
CAP	(CAP = ASC + OH)
ACS	
OH	
ASC.CAP	
OH.CAP	
Robustness	
ELD	
TD	
	Tetrapartite Partition of Ascendency Metrics
A.input	
A.internal	
A.export	
A.respiration	
OH.input	
OH.internal	
OH.export	
OH.respiration	
CAP.input	
CAP.internal	
CAP.export	
CAP.respiration	

Here, we extract the flow statistics and then isolate and remove the output-oriented direct flow intensity ($\mathbf{G}$) matrix. Recall that ENA is partially derived from Input–Output analysis; the input and output orientations provide different information about the system. We also show the input-oriented integral flow matrix $\mathbf{N}'$.

```
F <- enaFlow(oyster)
```

```
## Warning in enaAscendency(x): Export data is absent from the model.
```

```
attributes(F)
```

```
## $names
## [1] "T"  "G"  "GP" "N"  "NP" "ns"
```

```
F$ns
```

```
##      Boundary     TST     TSTp      APL       FCI       BFI       DFI
## [1,]    41.47 83.5833 125.0533 2.015512 0.1101686 0.4961517 0.1950689
##            IFI     ID.F   ID.F.I   ID.F.O    HMG.I    HMG.O AMP.I AMP.O
## [1,] 0.3087794 1.582925 1.716607 1.534181 2.051826 1.891638     3     1
##      mode0.F  mode1.F  mode2.F  mode3.F mode4.F        H      AMI       Hr
## [1,]   41.47 32.90504 9.208256 32.90504   41.47 3.018275 1.330211 1.688063
##           CAP      ASC       OH   ASC.CAP    OH.CAP robustness     ELD
## [1,] 377.4452 166.3473 211.0979 0.4407191 0.5592809  0.3611021 1.79506
##            TD  A.input A.internal A.export A.respiration OH.input
## [1,] 2.514395 66.03696   72.62476        0      27.68558        0
##      OH.internal OH.export OH.respiration CAP.input CAP.internal
```

```
## [1,]    103.2914         0       107.8065  66.03696     175.9162
##      CAP.export CAP.respiration
## [1,]          0         135.492
```

```
## Output-oriented direct flow matrix
F$G
```

```
##                     Filter Feeders Microbiota Meiofauna Deposit Feeders
## Filter Feeders                   0  0.0000000 0.0000000      0.00000000
## Microbiota                       0  0.0000000 0.1475753      0.14757529
## Meiofauna                        0  0.0000000 0.0000000      0.07793173
## Deposit Feeders                  0  0.0000000 0.0000000      0.00000000
## Predators                        0  0.0000000 0.0000000      0.00000000
## Deposited Detritus               0  0.3670363 0.3267221      0.02888377
##                      Predators Deposited Detritus
## Filter Feeders     0.01238245          0.3807813
## Microbiota         0.00000000          0.0000000
## Meiofauna          0.00000000          0.5000059
## Deposit Feeders    0.06856574          0.7600000
## Predators          0.00000000          0.4757876
## Deposited Detritus 0.00000000          0.0000000
```

```
## Input-oriented integral flow matrix
F$NP
```

```
##                     Filter Feeders Microbiota Meiofauna Deposit Feeders
## Filter Feeders                   1  1.0000000 1.0000000       1.0000000
## Microbiota                       0  1.1018630 0.2440716       0.6197856
## Meiofauna                        0  0.2971032 1.2971032       0.5604100
## Deposit Feeders                  0  0.1240688 0.1240688       1.1240688
## Predators                        0  0.0203426 0.0203426       0.0203426
## Deposited Detritus               0  1.3885039 1.3885039       1.3885039
##                     Predators Deposited Detritus
## Filter Feeders     1.0000000          1.0000000
## Microbiota         0.1555792          0.1018630
## Meiofauna          0.1406747          0.2971032
## Deposit Feeders    0.2821649          0.1240688
## Predators          1.0051064          0.0203426
## Deposited Detritus 0.3485436          1.3885039
```

In recognition that the Ascendency metrics (Ulanowicz 1986; Ulanowicz 1997) quantify the diversity of flow pathways in the model, the enaFlow function calls the enaAscendnecy (described in the next section). Thus, the Ascendency metrics (Ascendnecy, Overhead, Capacity, AMI, Hr, C, etc.) are included in the vector of flow-based whole-network statistics.

Ascendency

A key contribution of the Ulanowicz School to ENA is the Ascendency concept and the development of several information based network-level statistics (Ulanowicz 1986; Ulanowicz 1997). This analysis is based on all of the flows in the system and does not assume the modeled system is at steady-state. The enaAscendency function returns several of these information based measures. This is run as follows:

```
  enaAscendency(oyster)
```

```
## Warning in enaAscendency(oyster): Export data is absent from the model.
```

```
##             H      AMI       Hr      CAP      ASC       OH   ASC.CAP
## [1,] 3.018275 1.330211 1.688063 377.4452 166.3473 211.0979 0.4407191
##         OH.CAP robustness     ELD       TD  A.input A.internal A.export
## [1,] 0.5592809  0.3611021 1.79506 2.514395 66.03696   72.62476        0
##      A.respiration OH.input OH.internal OH.export OH.respiration CAP.input
## [1,]      27.68558        0    103.2914         0       107.8065  66.03696
##      CAP.internal CAP.export CAP.respiration
## [1,]     175.9162          0         135.492
```

Table 5: Graph-level network statistics returned by the *enaR* interpretations (Ulanowicz 1986; Ulanowicz 1997).

Label	Description
AMI	average mutual information (bits)
ASC	ascendency (AMI × TSTp)
OH	overhead
CAP	capacity
ASC.CAP	ascendency-to-capacity ratio (dimensionless)
OH.CAP	overhead-to-capacity ratio (dimensionless)
robustness	robustness of the network as in Fath (2014)
ELD	effective link density of the network Ulanowicz, Holt, and Barfield (2014)
TD	trophic depth of the network as in Ulanowicz, Holt, and Barfield (2014)

Storage Analysis

Storage ENA was developed in the Patten School (Barber 1978a; Barber 1978b). It is similar to flow ENA, but divides the flows by storage (e.g., biomass) instead of throughflow. Several papers provide an overview of this methodology (Fath and Patten 1999; Gattie et al. 2006; Schramski, Kazanci, and Tollner 2011). Output of this function is summarized in Table~(tab:storage). What follows is an example of applying the storage analysis to the oyster reef model.

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

```
## $names
##  [1] "X"   "C"   "P"   "S"   "VS"  "Q"   "CP"  "PP"  "SP"  "VSP" "QP"
## [12] "dt"  "ns"
```

```
S$ns
```

```
##           TSS       CIS         BSI         DSI       ISI     ID.S  ID.S.I
## [1,] 3112.044 0.9940252 0.003331412 0.003320932 0.9933477 299.1171 454.227
##        ID.S.O  HMG.S.O  HMG.S.I NAS NASP mode0.S  mode1.S mode2.S  mode3.S
## [1,] 294.1527 1.115985 1.464503  20   21 10.3675 8.226261 3093.45 8.226261
##      mode4.S
## [1,] 10.3675
```

Table 6: Matrices and graph-level network statistics returned by the *enaR* enaStorage function.

Label	Description
Matrices	
X	$n \times 1$ vector of storage values [M L^{-2}]
C	$n \times n$ donor-storage normalized output-oriented direct flow intensity matrix (T^{-1})
P	$n \times n$ storage-normalized output-oriented direct flow matrix (dimensionless)
S	$n \times n$ donor-storage normalized output-oriented integral flow intensity matrix (T^{-1})
Q	$n \times n$ output-oriented integral flow intensity matrix (dimensionless)
CP	$n \times n$ recipient-storage normalized input-oriented direct flow intensity matrix (T^{-1})
PP	$n \times n$ storage-normalized input-oriented direct flow matrix (dimensionless)
SP	$n \times n$ donor-storage normalized input-oriented integral flow intensity matrix (T^{-1})
QP	$n \times n$ input-oriented integral flow intensity matrix (dimensionless)
dt	discrete time step
Network Statistics	
TSS	Total System Storage
CIS	Storage Cycling Index
BSI	Boundary Storage Intensity
DSI	Direct Storage Intensity
ISI	Indirect Storage Intensity
ID.S	Ratio of Indirect-to-Direct storage (realized)
ID.S.I	storage-based input-oriented indirect-to-direct ratio (fath06)
ID.S.O	storage-based input-oriented indirect-to-direct ratio (fath06)
HMG.S.I	input-oriented storage network homogenization
HMG.S.O	output-oriented storage network homogenization
AMP.S.I	input-oriented storage network amplification
AMP.S.O	output-oriented storage network amplification
mode0.S	Storage from Boundary Flow
mode1.S	Storage from Internal First Passage Flow
mode2.S	Storage from Cycled Flow
mode3.S	Dissipative Equivalent to mode1.S
mode4.S	Dissipative Equivalent to mode0.S

This storage analysis of the oyster reef model indicates that the total energy stored in the system on an average day is 3112 kcal m^-2, and that 99.3% of this storage is generated by energy flowing over indirect pathways (ISI).

(Whipple, Patten, and Borrett 2014) provides a detailed example of applying storage analysis to characterize the dynamic organization of an ecosystem. They investigated how the storage analysis properties changed across sixteen consecutive seasonal N cycling models of the Neuse River Estuary. They found that from this storage perspective NOx was the dominant compartment, and thus a primary controller of the system dynamics. Note that this work provides an example of applying this analysis at multiple levels of analysis (Hines and Borrett 2014).

Environ Analysis

Environ Analysis finds the n unit input and output environs for the model (Patten 1978; Fath and Patten 1999). These unit environs are returned by the *environ* function as in NEA.m. They indicate the flow activity in each subnetwork generated by pulling a unit out of a node (input environs) or pushing a unit into a node (output environ). These unit environs can be converted into "realized" environs by multiplying each by the relevant observed input or output (Borrett and Freeze 2011; Whipple et al. 2007; Whipple, Patten, and Borrett 2014).

```
E <- enaEnviron(oyster)
```

```
## Warning in enaAscendency(x): Export data is absent from the model.
```

```
attributes(E)
```

```
## $names
## [1] "input"  "output"
```

```
E$output[1]
```

```
## $`Filter Feeders`
##                     Filter Feeders Microbiota   Meiofauna Deposit Feeders
## Filter Feeders                  -1  0.0000000  0.00000000      0.00000000
## Microbiota                       0 -0.1970605  0.02908126      0.02908126
## Meiofauna                        0  0.0000000 -0.20449723      0.01593682
## Deposit Feeders                  0  0.0000000  0.00000000     -0.06052568
## Predators                        0  0.0000000  0.00000000      0.00000000
## Deposited Detritus               0  0.1970605  0.17541596      0.01550760
## z                                1  0.0000000  0.00000000      0.00000000
##                       Predators Deposited Detritus           y
## Filter Feeders      0.012382445        0.380781288 0.606836267
## Microbiota          0.000000000        0.000000000 0.138897999
## Meiofauna           0.000000000        0.102249819 0.086310586
## Deposit Feeders     0.004149988        0.045999518 0.010376176
## Predators          -0.016532433        0.007865927 0.008666506
## Deposited Detritus  0.000000000       -0.536896552 0.148912467
## z                   0.000000000        0.000000000 0.000000000
```

The TET function returns vectors of the unit and realized input and output total environ throughflow. The realized total environ throughflow is an environ based partition of the total system throughflow (Whipple et al. 2007).

```
tet <- TET(oyster)
```

```
## Warning in enaAscendency(x): Export data is absent from the model.
```

```
## Warning in set.orient(user.orient): NOTE: output of functions from a
## particular analytical school will be returned in the standard orientation
## of that school.
```

```
show(tet)
```

```
## $realized.input
## [1] 25.165000 22.647638 14.582798  2.028052  1.053786 18.107007
##
## $realized.output
## [1] 83.5833  0.0000  0.0000  0.0000  0.0000  0.0000
##
## $unit.input
```

```
## [1] 1.000000 3.931882 4.074090 4.713111 2.932069 2.931882
##
## $unit.output
## [1] 2.015512 1.836089 2.540670 3.124836 2.234317 2.594261
```

The TES functions returns the both the realized and unit total environ storage for the input and output environs. Again, the realized TES is a partition of the total system storage (TSS).

```
tes <- TES(oyster)
show(tes)
```

```
## $realized.input
##     Filter Feeders         Microbiota          Meiofauna
##         2000.00000            2.41209           24.12171
##    Deposit Feeders          Predators Deposited Detritus
##           16.27440           69.23803         1000.03118
##
## $realized.output
## [1] 3112.044    0.000    0.000    0.000    0.000    0.000
##
## $unit.input
##     Filter Feeders         Microbiota          Meiofauna
##        289.3658066          0.6561948          7.3735209
##    Deposit Feeders          Predators Deposited Detritus
##         11.5308112        109.7205293        265.1036470
##
## $unit.output
##     Filter Feeders         Microbiota          Meiofauna
##           75.04326           16.06273           41.03146
##    Deposit Feeders          Predators Deposited Detritus
##           65.81279          132.44451           66.11575
```

Realized TET and TES might be considered network centrality measures that indicate the relative importance of the environs in generating the observed flow or storage, respectively.

Utility Analysis

Utility analysis describes the relationship between node pairs in the ecosystem model when considering both direct and indirect interactions. It developed in the Patten School (Patten 1991; Fath and Patten 1999) and is similar to yet distinct from the Ulanowicz School mixed trophic impacts analysis (Ulanowicz and Puccia 1990). Utility analysis can be conducted from both the flow and storage perspectives, so the "type" argument needs to be set to suit the user's needs. This is again implemented as in *NEA.m*. The following table summarizes the function output for the flow and storage versions. These analyses are executed as:

```
UF <- enaUtility(oyster, eigen.check=TRUE,type="flow")
US <- enaUtility(oyster, eigen.check=TRUE,type="storage")
attributes(UF)
```

```
## $names
## [1] "D"              "SD"             "U"              "Y"
## [5] "SY"             "Relations.Table" "ns"
```

Table 7: Matrices and graph-level network statistics returned by the *enaR* enaUtility function.

Label	Description	Co
Matrices		
$D_{n\times n}$	throughflow-normalized direct utility intensity (dimensionless)	**D**
$U_{n\times n}$	integral flow utility (dimensionless)	**U**
$Y_{n\times n}$	integral flow utility scaled by original throughflow (M $L^{-2 \text{ or } -3}$ T^{-1})	**Y**
$DS_{n\times n}$	storage-normalized direct utility intensity (dimensionless)	$\mathbf{D_S}$
$US_{n\times n}$	integral storage utility (dimensionless)	$\mathbf{U_S}$
$YS_{n\times n}$	integral storage utility scaled by original throughflow (M $L^{-2 \text{ or } -3}$ T^{-1})	$\mathbf{Y_S}$
Other Objects		
Relations.Table	a table listing the pairwise relationships derived from both the direct and integral perspective.	
Network Statistics		
lam1D	dominant eigenvalue of D; must be <1 for **D** power series to converge	λ_1(
relation.change.F	Percent of relatinoships that changed between the direct and integral flow utility analysis	
synergism.F	benefit-cost ratio or network synergism (flow)	*SY*
mutualism.F	positive to negative interaction ratio or network mutualism (flow)	*M*
lam1DS	dominant eigenvalue of DS; must be <1 for the $\mathbf{D}_S$ power series to converge	λ_1(
relation.change.S	Percent of relatinoships that changed between the direct and integral storage utility analysis	
synergism.S	benefit-cost ratio or network synergism (storage)	*SY*
mutualism.S	positive to negative interaction ratio or network mutualism (storage)	*M*

Please note the function argument *eigen.check = TRUE*. For this analysis to work, the power series of the direct utility matrices must converge, which is only TRUE if the dominant eigenvalue of the direct utility matrix is less than 1. The function default prevents the analysis from being performed if this condition is not met. Users that wish to perform the analysis anyway can set "eigen.check=FALSE". Care should be used when doing this, as the meaning of the underlying mathematics is uncertain.

While this function returns a number of results, the *Relations.Table* summarizes a number of the critical results. It shows the character of the pairwise relationships between each node combination when considering the direct and the integral relations. Thus, it shows the power of the network to transform the nature of the ecological relationships among the system components. This change is reflected in the synergism and mutualism whole-network metrics.

```
UF$Relations.Table
```

From	To	Direct	Integral	changed
Filter Feeders	Filter Feeders	(0,0)	(+,+)	*
Filter Feeders	Microbiota	(0,0)	(+,+)	*
Filter Feeders	Meiofauna	(0,0)	(+,+)	*
Filter Feeders	Deposit Feeders	(0,0)	(+,-)	*
Filter Feeders	Predators	(+,-)	(+,-)	-
Filter Feeders	Deposited Detritus	(+,-)	(+,-)	-
Microbiota	Microbiota	(0,0)	(+,+)	*
Microbiota	Meiofauna	(+,-)	(+,-)	-
Microbiota	Deposit Feeders	(+,-)	(+,-)	-
Microbiota	Predators	(0,0)	(+,+)	*
Microbiota	Deposited Detritus	(-,+)	(-,+)	-
Meiofauna	Meiofauna	(0,0)	(+,+)	*
Meiofauna	Deposit Feeders	(+,-)	(+,-)	-
Meiofauna	Predators	(0,0)	(+,+)	*

From	To	Direct	Integral	changed
Meiofauna	Deposited Detritus	(-,+)	(-,+)	-
Deposit Feeders	Deposit Feeders	(0,0)	(+,+)	*
Deposit Feeders	Predators	(+,-)	(+,-)	-
Deposit Feeders	Deposited Detritus	(+,-)	(+,+)	*
Predators	Predators	(0,0)	(+,+)	*
Predators	Deposited Detritus	(+,-)	(-,-)	*
Deposited Detritus	Deposited Detritus	(0,0)	(+,+)	*

```
UF$ns
```

```
##              lam1D relation.change.F synergism.F mutualism.F
## r.change 0.8991676              61.9    4.915298    2.272727
```

Mixed Trophic Impacts

Mixed Trophic Impacts is a popular analysis from the Ulanowicz School of ENA (Ulanowicz and Puccia 1990). The enaMTI function generates comparable results to the calculations in (Ulanowicz and Puccia 1990). These are implemented as follows:

```
mti <- enaMTI(oyster)
attributes(mti)
```

```
## $names
## [1] "G"               "FP"              "Q"               "M"
## [5] "Relations.Table"
```

```
mti$M
```

```
## [1] NA
```

In this case, the power series of the direct trophic impacts matrix does not converge (dominant eigenvalue is greater than one). Thus, the function returns NA. Like with Utility analysis, however, we can use the eigen.check argument to do the calculation despite the mathematical problem.

```
mti <- enaMTI(oyster,eigen.check=FALSE)
attributes(mti)
```

```
## $names
## [1] "G"               "FP"              "Q"               "M"
## [5] "Relations.Table"
```

```
mti$M
```

```
##                   Filter Feeders  Microbiota    Meiofauna Deposit Feeders
## Filter Feeders      -0.0250635283  0.16956382  0.431493557      0.26144106
## Microbiota          -0.0015848556 -0.30675078 -0.182458391      0.20520368
## Meiofauna           -0.0001241781 -0.47413204 -0.070959618      0.01607831
## Deposit Feeders     -0.0069255188 -0.26769125 -0.007062628     -0.10329881
```

```
## Predators           -0.0301817448  0.02000515 -0.004028911     -0.07586335
## Deposited Detritus  -0.0034657973  0.21795628  0.612654910      0.44874394
##                       Predators Deposited Detritus
## Filter Feeders      0.795834137        0.516016759
## Microbiota          0.050323410       -0.295378609
## Meiofauna           0.003942987       -0.001592286
## Deposit Feeders     0.219903765        0.177109591
## Predators          -0.041648786       -0.019939324
## Deposited Detritus  0.110048344       -0.251366300
```

Table 9: Matrices returned by the *enaR* enaMTI function, which are based on Ulanowicz and Puccia (1990).

Label	Description	Common.Symbols
Matrices		
$G_{n\times n}$	positive effect of prey on its predator; identical to the input-oriented direct flow matrix	$\mathbf{G}'$ or $\mathbf{B}'$
$FP_{n\times n}$	negative impact of the predator on its prey	$\mathbf{F}'$ or $\check{\mathbf{B}}$
$Q_{n\times n}$	direct net impact of one node on another	$\mathbf{Q}$ or $\check{\mathbf{D}}$
$M_{n\times n}$	total impact of i on j (direct and indirect)	$\mathbf{M}$ or $\check{\mathbf{U}}$

The mixed trophic impacts analysis has been usefully applied to discover interesting and sometimes unexpected ecological relationships. For example, although alligators directly eat frogs in the Florida Everglades (USA), it appears that their net relationship when considering the whole food web is actually mutualistic (Bondavalli and Ulanowicz 1999). This is in part because the alligators also eat other key predators of the frogs such as snakes.

As with enaUtility, enaMTI returns a summary table of the pairwise relationships between each node pair (*Relations.Table*). This table includes the relationship when only the direct connectsion are considered, and the relationships when the mixed or integral connections are considered.

```
mti$Relations.Table
```

From	To	Net (direct)	Mixed (integral)	changed
Filter Feeders	Filter Feeders	(0,0)	(-,-)	*
Filter Feeders	Microbiota	(0,0)	(-,+)	*
Filter Feeders	Meiofauna	(0,0)	(-,+)	*
Filter Feeders	Deposit Feeders	(0,0)	(-,+)	*
Filter Feeders	Predators	(-,+)	(-,+)	-
Filter Feeders	Deposited Detritus	(0,+)	(-,+)	*
Microbiota	Microbiota	(0,0)	(-,-)	*
Microbiota	Meiofauna	(-,+)	(-,-)	*
Microbiota	Deposit Feeders	(-,+)	(-,+)	-
Microbiota	Predators	(0,0)	(+,+)	*
Microbiota	Deposited Detritus	(+,-)	(+,-)	-
Meiofauna	Meiofauna	(0,0)	(-,-)	*
Meiofauna	Deposit Feeders	(-,+)	(-,+)	-
Meiofauna	Predators	(0,0)	(-,+)	*
Meiofauna	Deposited Detritus	(+,-)	(+,-)	-
Deposit Feeders	Deposit Feeders	(0,0)	(-,-)	*
Deposit Feeders	Predators	(-,+)	(-,+)	-
Deposit Feeders	Deposited Detritus	(+,+)	(+,+)	-

From	To	Net (direct)	Mixed (integral)	changed
Predators	Predators	(0,0)	(-,-)	*
Predators	Deposited Detritus	(0,+)	(+,-)	*
Deposited Detritus	Deposited Detritus	(0,0)	(-,-)	*

In the exemplar Oyster Reef model, we see that the *Filter Feeder* compartment has no direct relationship with the *Microbiota*. However, when the Mixed or integral relationships are considered in the MTI framework, the relationship changes such that the *Microbiota* appear to be funtionally predators of teh *Filter Feeders*.

Control Analysis

Control analysis was implemented as in the NEA.m function, but we also include recent updates to control analysis (Schramski et al. 2006; Schramski et al. 2007). In general, these analyses determine the pairwise control relationships between the nodes in the network.

```
C <- enaControl(oyster)
```

```
## Warning in enaAscendency(x): Export data is absent from the model.
```

```
attributes(C)
```

```
## $names
## [1] "CN"   "CQ"   "CD"   "CR"   "CA"   "CDep" "sc"   "psc"  "ns"
```

```
C$sc
```

```
##     Filter Feeders         Microbiota          Meiofauna
##        0.120569086       -0.063400232       -0.042706068
##    Deposit Feeders          Predators Deposited Detritus
##        0.002634493       -0.069125297        0.052028018
```

The elements of the sc vector indicate the relative control exherted by each node on the system functioning.

Table 11: Matrices returned by the *enaR* enaControl function, which are based on (Dame and Patten 1981; Patten and Auble 1981; Schramski et al. 2006; Schramski et al. 2007).

Label	Description
Matrices	
$CN_{n\times n}$	Control matrix using flow values
$CQ_{n\times n}$	Control matrix using storage values
$CR_{n\times n}$	Schramski's Control Ratio Matrix
$CD_{n\times n}$	Schramski's Control Difference Matrix
$sc_{n\times 1}$	Schramski's System Control vector

Cycle Analysis

The Cycle Analysis provides the detailed account of the cycling present in the network. It follows the algorithm by the DOS-based NETWRK 4.2b software by Ulanowicz (Ulanowicz and Kay 1991; Ulanowicz 1983) and provides results similar to NETWRK's 'Full Cycle Analysis'. Cycles in a network are grouped together into disjoint nexuses and each nexus is characterized by a weak arc. This function gives details of the individual cycles along with the disjoint nexuses present in the network.

```
  cyc <- enaCycle(oyster)
attributes(cyc)
```

```
## $names
## [1] "Table.cycle"      "Table.nexus"      "CycleDist"
## [4] "NormDist"         "ResidualFlows"    "AggregatedCycles"
## [7] "ns"
```

```
## The individual cycles
names(cyc$Table.cycle)
```

```
## [1] "CYCLE" "NEXUS" "NODES"
```

```
## The disjoint nexuses
names(cyc$Table.nexus)
```

```
## [1] "NEXUS"      "CYCLES"     "W.arc.From" "W.arc.To"   "W.arc.Flow"
```

Table 12: Data frames, matrices and graph-level network statistics returned by the *enaR* enaCycle function, which is based on Ulanowicz (1983).

Label	Description
Data frames	
Table.cycle	Data frame of cycles in the network. Up to 50 cycles are returned per nexus
Table.nexus	Data frame with details of the disjoint nexuses present in the network
Matrices	
$\text{CycleDist}_{n\times 1}$	Vector of flows cycling in loops of increasing length
$\text{NormDist}_{n\times 1}$	Vector of Cycle Distributions normalized by the total system throughput
$\text{ResidualFlows}_{n\times n}$	Matrix of straight-through flows or the underlying acyclic graph
$\text{AggregatedCycles}_{n\times n}$	Matrix of all the cycled flows or the underlying cyclic graph
Network Statistics	
NCYCS	Number of cycles detected in the network
NNEX	Number of disjoint nexuses detected in the network
CI	Cycling index of the network based on flow matrix

Trophic Analysis

The Trophic Aggregation algorithm (enaTroAgg) assumes that the network being analyzed is a food web and performs a number of trophic-based analysises. Specifically, it identifies the trophic structure of the given network based on the Lindeman's trophic concepts (Lindeman 1942). This includes identifying the effective

trophic level of each newtork node, and building the 'Linedeman Trophic Spine'.

The algorithm is implemented as in NETWRK 4.2b by Ulanowicz (Ulanowicz and Kemp 1979) and provides similar results as NETWRK's 'Lindeman Trophic Aggregations' (Ulanowicz and Kay 1991). It apportions the nodes into integer trophic levels and estimates the corresponding inputs, exports, respirations and the grazing chain and trophic spine which represent the transfers between integer trophic levels.

It is crucial for this algorithm that the cycles among the living nodes of the network (Feeding Cycles) be removed beforehand to assign trophic levels to nodes. Thus, the output for this function contains the Cycle Analysis for the Feeding Cycles in the network.

Following (Ulanowicz and Kay 1991), the non-living nodes are grouped together for this analysis and referred to as the detrital pool.

The following table summarizes the function output except the outputs for the feeding cycles which are similar to the enaCycle outputs.

```
trop <- enaTroAgg(oyster)
attributes(trop)
```

```
## $names
##  [1] "Feeding_Cycles" "A"              "ETL"            "CE"
##  [5] "CR"             "GC"             "RDP"            "LS"
##  [9] "TE"             "ns"
```

```
## Cycle analysis output for Feeding Cycles
trop$Feeding_Cycles
```

```
## $ResidualFlows
##                 Filter Feeders Microbiota Meiofauna Deposit Feeders
## Filter Feeders               0          0     0.000          0.0000
## Microbiota                   0          0     1.206          1.2060
## Meiofauna                    0          0     0.000          0.6609
## Deposit Feeders              0          0     0.000          0.0000
## Predators                    0          0     0.000          0.0000
##                 Predators
## Filter Feeders     0.5135
## Microbiota         0.0000
## Meiofauna          0.0000
## Deposit Feeders    0.1721
## Predators          0.0000
```

Table 13: Matrices and graph-level network statistics returned by the *enaR* enaTroAgg function, which are based on Ulanowicz and Kemp (1979).

Label	Description
Matrices	
$A_{nl \times nl}$	Lindeman transformation matrix that apportions nodes to integer trophic levels
$ETL_{n \times 1}$	Vector of the effective trophic levels of different nodes
$M.Flow_{nl \times 1}$	Migratory flows in living nodes (if present)
$CI_{n \times 1}$	Vector of canonical inputs to integer trophic levels (if migratory flows present)
$CE_{n \times 1}$	Canonical Exports. Vector of exports from Integer trophic levels
$CR_{n \times 1}$	Canonical Respirations. Vector of respiration from Integer trophic levels

Label	Description
$GC_{nl\times 1}$	Grazing Chain. Vector of inputs to Integer trophic levels from preceding level
$RDP_{nl\times 1}$	Vector of returns from each level to the detrital pool
$LS_{nl\times 1}$	Vector representing the Lindeman Spine
$TE_{nl\times 1}$	Vector of the trophic efficiencies for integer trophic levels
Network Statistics	
Detritivory	Flow from the detrital pool (non-living nodes) to the second trophic level
DetritalInput	Exogenous inputs to the detrital pool
DetritalCirc	internal circulation within the detrital pool
NCYCS	number of feeding cycles removed from the network
NNEX	number of disjoint nexuses detected for the feeding cycles
CI	cycling index of the living component of the network based on flow matrix

Additional Helpful Functions

There are a number of additional tools in the package. Here selected a subset of these to highlight.

Centrality

Centrality analysis is a large topic in network science (Brandes and Erlebach 2005; Wasserman and Faust 1994). In general the goal is to describe the relative importance of parts of the networks (nodes, edges, environs). Many different types of centrality measures exist in network science (Freeman 1979; Freeman, Borgatti, and White 1991; Borgatti and Everett 2006; Brandes and Erlebach 2005). Environ centrality is unique to ENA (Fann and Borrett 2012), but like eigenvector centrality, it is a degree-based centrality measure that considers the equilibrium effect of all pathways of all lengths in the system and as such can be classified as a global centrality measure. Both of these centralities can be calculated in *enaR* as follows:

```
F <- enaFlow(oyster)
```

```
## Warning in enaAscendency(x): Export data is absent from the model.
```

```
ec <- environCentrality(F$N)
show(ec)
```

```
## $ECin
##     Filter Feeders         Microbiota          Meiofauna
##          0.1404961          0.1279889          0.1771034
##    Deposit Feeders          Predators Deposited Detritus
##          0.2178241          0.1557484          0.1808391
##
## $ECout
##     Filter Feeders         Microbiota          Meiofauna
##         0.06970737         0.19108709         0.20595483
##    Deposit Feeders          Predators Deposited Detritus
##         0.12350944         0.07903903         0.33070223
##
## $AEC
##     Filter Feeders         Microbiota          Meiofauna
##          0.1051017          0.1595380          0.1915291
##    Deposit Feeders          Predators Deposited Detritus
##          0.1706668          0.1173937          0.2557707
```

```
eigenCentrality(F$G)
```

```
## $EVCin
## [1] 0.1207568 0.1093625 0.1876329 0.2518905 0.1470501 0.1833072
##
## $EVCout
## [1] 0.00000000 0.23325048 0.26566843 0.11130122 0.01286707 0.37691280
##
## $AEVC
## [1] 0.06037842 0.17130647 0.22665067 0.18159586 0.07995858 0.28011000
```

These centrality values have been normalized to sum to one. In addition, the throughflow vector from flow analysis (Borrett 2013), the total environ throughflow, and total environ storage vectors might also be considered centrality metrics (Whipple et al. 2007; Whipple, Patten, and Borrett 2014). The following code and figure demonstrates how the Averavge Environ Centrality can be quantified and visualized.

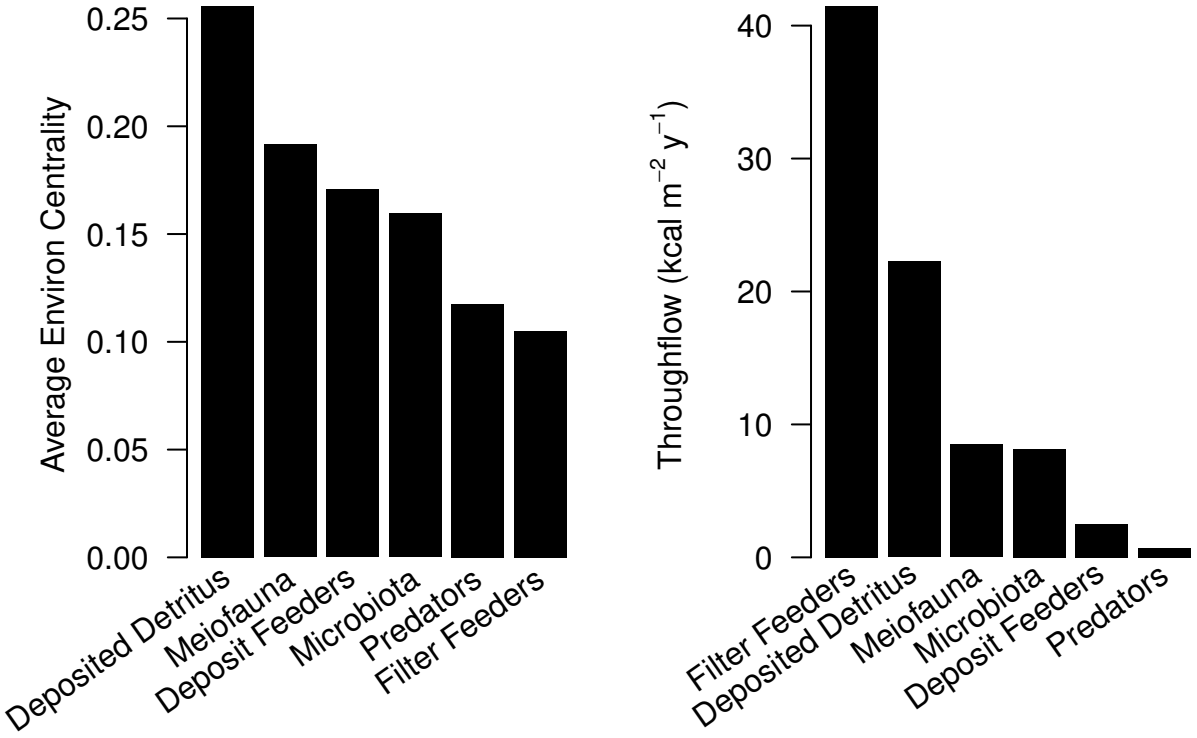
```
## Set plotting parameters
opar <- par(las=1,mfrow=c(1,2),mar=c(7,5,1,1),xpd=TRUE,bg="white")
## Find centrality order
o <- order(ec$AEC,decreasing=TRUE)

## Creating a barplot
bp <- barplot(ec$AEC[o],
              names.arg=NA,
              ylab="Average Environ Centrality",
              col="black",border=NA)
## Adding labels
text(bp,-0.008,
     labels=names(ec$AEC)[o],
     srt=35,adj=1,cex=1)

# throughflow centrality
T <- enaFlow(oyster)$T
```

```
## Warning in enaAscendency(x): Export data is absent from the model.
```

```
o <- order(T,decreasing=TRUE)
bp2 <- barplot(T[o],
               names.arg=NA,
               ylab=expression(paste("Throughflow (kcal m"^-2, " y"^-1,")")),
               col="black", border=NA)
text(bp2,-1,
     labels=names(T)[o],
     srt=35,adj=1,cex=1)
```




```
## Remove the plotting parameters
 rm(opar)
```

```
## Warning in enaAscendency(x): Export data is absent from the model.
```

Shannon Diversity

Biodiversity is a critical concept in ecology and conservation biology. For example, it is hypothesized to contribute to the stability, productivity, and broad ecosystem functioning of these complex dynamic systems (Tilman, Wedin, and Knops 1996; Hooper et al. 2005). Ecologists often use Shannon's measure of information entropy (H) as an indicator of biodiveristy because it captures the effects of richness (number of species) and the evenness of the distribution of individuals among the species (Shannon and Weaver 1949).

Shannon's entroy based metric of diversity is

$$H = -1 \sum_{i=1}^{n} p_i \log(p_i) \tag{2}$$

where p_i is the relative abundance or quantitiy and n is the number of species (nodes in this context).

This metrics can be applied in a number of ways within the context of ENA. For example, we can find the diversity of storage (biomass) by letting $p_i = X_i/X_\bullet = X_i/\sum X_i$, or we can find the throughflow diversity by letting $p_i = T_i/T_\bullet$. In fact, this can be applied to nearly any node Centrality metrics.

For any given input vector, the maximum possible value of H is $H_{max} = \log(n)$. Thus, we can focus on the evenness component of biodiversity by calculating the relative entropy $0 \leq (H_r = H/H_{max}) \leq 1$. The closer H_r gets to 1, the more evenly distributed the stuff is among the n nodes. From this we can derive a metric of *centralization* ($H_{centraliztion} = 1 - H_r$ or how concentrated the elements are in a smaller number of nodes.

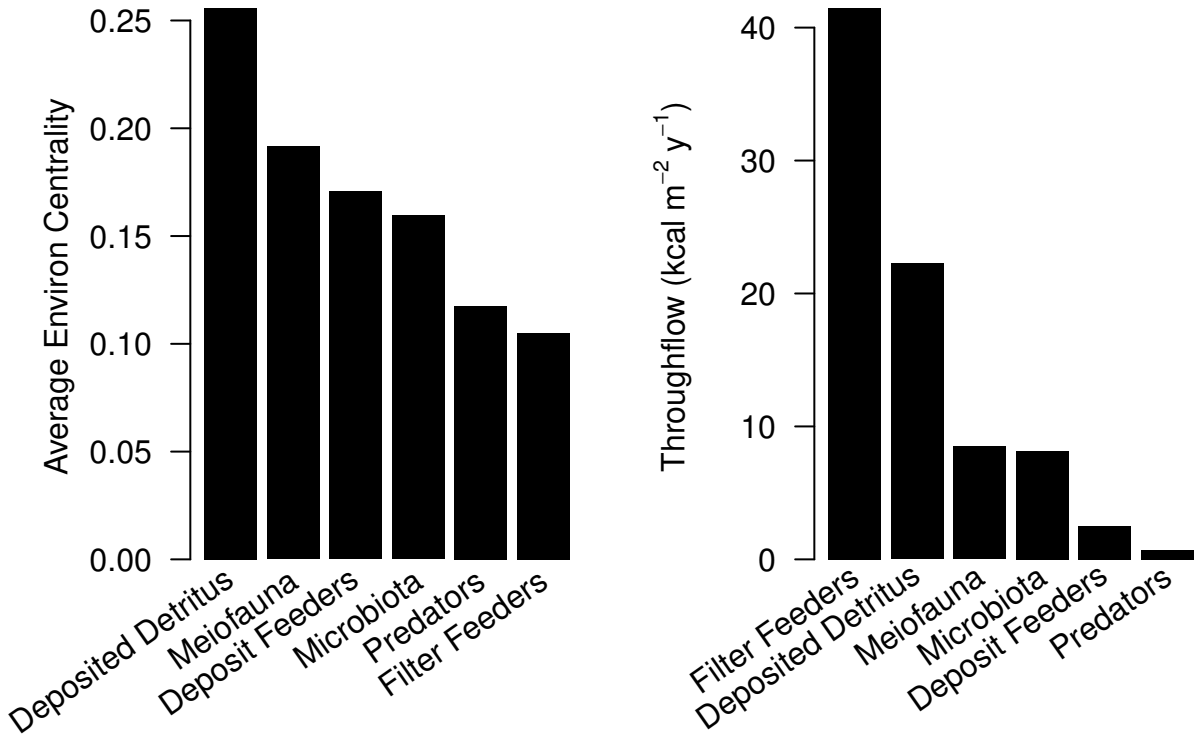


Figure 2: Bar plots of the Oyster Reef model Average Environ Centralities (left) and Throughflow Centralities).

This is a useful metic becuase it ties back to concepts in Social Network Analysis (Wasserman and Faust 1994). We can recover the effective number of nodes based on the evenness as $s = e^H$ (Ulanowicz, Holt, and Barfield 2014).

The ShannonDiversity function in *enaR* returns each of these metrics for any vector input. For example,

```
ShannonDiversity(F$T)  # throughflow diversity
```

```
##           H        Hmax          Hr    Hcentral           n effective.n
##   1.3042705   1.7917595   0.7279272   0.2720728   6.0000000   3.6849998
```

```
ShannonDiversity(S$X)  # storage diversity
```

```
##           H        Hmax          Hr    Hcentral           n effective.n
##   0.8043025   1.7917595   0.4488898   0.5511102   6.0000000   2.2351370
```

The results for the Oyster Reef model indicate that throughflow diversity is greater than the diversity from the stroage perspective. This is becuase the throughflow values are more evenly distributed (less centralized) than the stroage values. This is perhaps most clear when we determine the effective richness in the system. From the throughflow perspective there are 3.7 nodes acting, while from the storage perspective the effective number of nodes is estimated to be 2.2.

Quickly Return Multiple Analyses

There are two functions that aggregate multiple analyses and report selected results. A quick way to get a list of the global network statistics reported in Structure, Flow, Ascendency, Storage, and Utility analysis is to use the get.ns function.

```
ns <- get.ns(oyster)
```

```
## Node 1, Reach 1, Total 1
## Node 2, Reach 6, Total 7
## Node 3, Reach 6, Total 13
## Node 4, Reach 6, Total 19
## Node 5, Reach 6, Total 25
## Node 6, Reach 6, Total 31
```

```
## Warning in enaAscendency(x): Export data is absent from the model.
```

```
## Examine the structure of ns
str(ns)
```

```
## 'data.frame':    1 obs. of  81 variables:
##  $ n                : num 6
##  $ L                : num 12
##  $ C                : num 0.333
##  $ LD               : num 2
##  $ ppr              : num 2.15
##  $ lam1A            : num 2.15
##  $ mlam1A           : num 1
##  $ rho              : num 2.15
##  $ R                : num 0.466
##  $ d                : num 0.148
##  $ no.scc           : num 2
##  $ no.scc.big       : num 1
##  $ pscc             : num 0.833
##  $ Boundary         : num 41.5
##  $ TST              : num 83.6
##  $ TSTp             : num 125
##  $ APL              : num 2.02
##  $ FCI              : num 0.11
##  $ BFI              : num 0.496
##  $ DFI              : num 0.195
##  $ IFI              : num 0.309
##  $ ID.F             : num 1.58
##  $ ID.F.I           : num 1.72
##  $ ID.F.O           : num 1.53
##  $ HMG.I            : num 2.05
##  $ HMG.O            : num 1.89
##  $ AMP.I            : num 3
##  $ AMP.O            : num 1
##  $ mode0.F          : num 41.5
##  $ mode1.F          : num 32.9
##  $ mode2.F          : num 9.21
##  $ mode3.F          : num 32.9
##  $ mode4.F          : num 41.5
##  $ H                : num 3.02
##  $ AMI              : num 1.33
##  $ Hr               : num 1.69
##  $ CAP              : num 377
```

```
##  $ ASC              : num 166
##  $ OH               : num 211
##  $ ASC.CAP          : num 0.441
##  $ OH.CAP           : num 0.559
##  $ robustness       : num 0.361
##  $ ELD              : num 1.8
##  $ TD               : num 2.51
##  $ A.input          : num 66
##  $ A.internal       : num 72.6
##  $ A.export         : num 0
##  $ A.respiration    : num 27.7
##  $ OH.input         : num 0
##  $ OH.internal      : num 103
##  $ OH.export        : num 0
##  $ OH.respiration   : num 108
##  $ CAP.input        : num 66
##  $ CAP.internal     : num 176
##  $ CAP.export       : num 0
##  $ CAP.respiration  : num 135
##  $ TSS              : num 3112
##  $ CIS              : num 0.994
##  $ BSI              : num 0.00333
##  $ DSI              : num 0.00332
##  $ ISI              : num 0.993
##  $ ID.S             : num 299
##  $ ID.S.I           : num 454
##  $ ID.S.O           : num 294
##  $ HMG.S.O          : num 1.12
##  $ HMG.S.I          : num 1.46
##  $ NAS              : num 20
##  $ NASP             : num 21
##  $ mode0.S          : num 10.4
##  $ mode1.S          : num 8.23
##  $ mode2.S          : num 3093
##  $ mode3.S          : num 8.23
##  $ mode4.S          : num 10.4
##  $ lam1D            : num 0.899
##  $ relation.change.F: num 61.9
##  $ synergism.F      : num 4.92
##  $ mutualism.F      : num 2.27
##  $ lam1DS           : num 0.302
##  $ relation.change.S: num 61.9
##  $ synergism.S      : num 13.1
##  $ mutualism.S      : num 2.6
```

It is also possible to instantly return all of the main ENA output with enaAll:

```
oyster.ena <- enaAll(oyster)
```

```
## Warning in enaAscendency(x): Export data is absent from the model.

## Warning in enaAscendency(x): Export data is absent from the model.
```

```
## Warning in enaAscendency(x): Export data is absent from the model.

## Warning in enaAscendency(x): Export data is absent from the model.

## Node 1, Reach 1, Total 1
## Node 2, Reach 6, Total 7
## Node 3, Reach 6, Total 13
## Node 4, Reach 6, Total 19
## Node 5, Reach 6, Total 25
## Node 6, Reach 6, Total 31
```

```
names(oyster.ena)
```

```
## [1] "ascendency" "control"    "environ"    "flow"       "mti"
## [6] "storage"    "structure"  "utility"
```

Output Orientation

To facilitate package use by the existing ENA community, some of which use the column-to-row orientation (e.g. the Patten School), we have created orientation functions that enable the user to set the expected output orientation for functions written in a particular "school" of analysis. Thus, functions from either school will receive network models with the standard row-to-column, but will return output with flow matrices oriented in the column-to-row orientation when appropriate (i.e. Patten school functions) and return them in that same orientation.

Here is an example of how to use the model orientation functions to re-orient the output from enaFlow:

```
## Check the current orientation
get.orient()
```

```
## [1] "rc"
```

```
## enaFlow output in row-column
flow.rc <- enaFlow(oyster)$G
```

```
## Warning in enaAscendency(x): Export data is absent from the model.
```

```
## Set the global orientation to school
set.orient('school')
```

```
## Warning in set.orient("school"): NOTE: output of functions from a
## particular analytical school will be returned in the standard orientation
## of that school.
```

```
## Check that it worked
get.orient()
```

```
## [1] "school"
```

```
## enaFlow output in column-row
flow.cr <- enaFlow(oyster)$G
```

```
## Warning in enaAscendency(x): Export data is absent from the model.
```

```
## Check. Outputs should be transposed from each other.
all(flow.rc == flow.cr)
```

```
## [1] FALSE
```

```
all(flow.rc == t(flow.cr))
```

```
## [1] TRUE
```

```
## Now change back to the default orientation ('rc')
set.orient('rc')
```

Matrix powers – raising a matrix to a power is not a native operation in **R**. Thus, the *enaR* package includes a function mExp to facilitate this matrix operation commonly used in ENA.

```
mExp(F$G,2)
```

```
##                     Filter Feeders Microbiota  Meiofauna Deposit Feeders
## Filter Feeders                   0  0.1397606 0.12440966      0.01099840
## Microbiota                       0  0.0000000 0.00000000      0.01150080
## Meiofauna                        0  0.1835203 0.16336297      0.01444205
## Deposit Feeders                  0  0.2789476 0.24830879      0.02195166
## Predators                        0  0.1746313 0.15545033      0.01374254
## Deposited Detritus               0  0.0000000 0.05416549      0.07962750
##                       Predators Deposited Detritus
## Filter Feeders     0.000000000        0.005891414
## Microbiota         0.010118608        0.185945731
## Meiofauna          0.005343446        0.059228112
## Deposit Feeders    0.000000000        0.032622730
## Predators          0.000000000        0.000000000
## Deposited Detritus 0.001980437        0.185314635
```

netOrder

Sometimes it is helpful to reorder the nodes in a network. While a simple re-ordering should not change the linear algebra based *enaR* results, it can be helpful to present results or for the construction of some algorithms. Thus, the netOrder function lets the user reorder the nodes in a network to any specified vector.

```
troModels[[6]]%v%'vertex.names' # original node name order
```

```
## [1] "PLANTS"           "BACTERIA"         "DETRITUS FEEDERS"
## [4] "CARNIVORES"       "DETRITUS"
```

```
new.network <- netOrder(troModels[[6]], c(1, 3, 2, 5, 4))
# new.network is the rearranged network with nodes in the desired order.

new.network%v%'vertex.names' # new node name order
```

```
## [1] "PLANTS"           "DETRITUS FEEDERS" "BACTERIA"
## [4] "DETRITUS"         "CARNIVORES"
```

```
as.matrix(new.network, attr="flow")
```

```
##                   PLANTS DETRITUS FEEDERS BACTERIA DETRITUS CARNIVORES
## PLANTS                 0                0        0      200          0
## DETRITUS FEEDERS       0                0        0      167       8881
## BACTERIA               0             2309        0     1600          0
## DETRITUS               0             5205       75        0          0
## CARNIVORES             0                0        0      370          0
```

Note that this will also change the order for the model flows, as the whole network data object has been reordered.

as.bipartite

as.extended

findPathLength

relationalChange

ssCheck

The ssCheck function is applied to an ecological network model to determine if the model is at steady state. Specifically, it compares T_i^{input} to T_i^{output} for all i. For practical reasons, this function returns the logical value $TRUE$ if $(|T_i^{input} - T_j^{output}|)/T_i^{output} * 100 \leq 5\%, quad \forall i$.

```
ssCheck(oyster)
```

```
## [1] TRUE
```

```
ssCheck(enaModels[[11]])
```

```
## [1] FALSE
```

While users and other *enaR* functions most often simply need to know if the system is at steady state, the function can also return the input and output throughflow vectors and a vector of the percent differences.

```
ssCheck(enaModels[[11]], more = TRUE)
```

```
## $ss
## [1] FALSE
##
## $Tin
## Pelagic Producers          Bacteria   Microzooplakton    Mesozooplakton
##             27.90              8.00              6.00             12.00
##   Inv. Carnivores      Pelagic Fish Benthis Producers         Dem. Fish
##              1.70              1.57              3.90              0.22
##        Macrofauna         Meiofauna          Sedim. C               DOM
##              2.02              4.32             14.57             31.35
##
## $Tout
## Pelagic Producers          Bacteria   Microzooplakton    Mesozooplakton
##             24.90              8.00              6.00             12.00
##   Inv. Carnivores      Pelagic Fish Benthis Producers         Dem. Fish
##              1.70              1.57              3.90              0.21
##        Macrofauna         Meiofauna          Sedim. C               DOM
##              2.02              4.32              5.47              9.05
##
## $perror
## Pelagic Producers          Bacteria   Microzooplakton    Mesozooplakton
##         12.048193          0.000000          0.000000          0.000000
##   Inv. Carnivores      Pelagic Fish Benthis Producers         Dem. Fish
##          0.000000          0.000000          0.000000          4.761905
##        Macrofauna         Meiofauna          Sedim. C               DOM
##          0.000000          0.000000        166.361974        246.408840
```

Knowing which nodes are not at steady state and how far off they are can help model construction and manual balancing steps. It is also a handy tool to ensure that models are imported correctly into the package.

Multi-Model Analyses (Batch Processing)

While many investigators analyze single models, much of ENA is used to compare ecosystem models (Baird, McGlade, and Ulanowicz 1991; Oevelen et al. 2006; Christian and Thomas 2003; Niquil et al. 2012; Hines et al. 2015). Investigators have also xanalyzed large sets of models to determine the generality of hypothesized ecosystem properties (Christensen 1995; Borrett and Salas 2010; Salas and Borrett 2011). For both of these applications, investigators need to analyze multiple models. One advantage of the *enaR* **R** package is that it simplifies this batch processing. Here we illustrate how to batch analyze a selection of models.

Our first step is to build an **R** list data object with ecosystem network models to batch analyze as the elements of the list. To illustrate batch processing, we will use a subset of the trophic models distributed with *enaR*, which are already stored as a list.

```
data(troModels)
```

Now that we have the models loaded, we can start to manipulate them. Once we have balanced the models, we can run the flow analysis on them. We are using the lapply function to iterate the analysis across the list of models stored in model.list. This approach is more compact and computationally efficient than a using for-loop.

```
# balance models as necessary
m.list <- lapply(troModels[1:10],balance)
```

```
## [1] BALANCED
## [1] BALANCED
## [1] BALANCED
## [1] BALANCED
## [1] BALANCED
## [1] BALANCED
## [1] BALANCED
## [1] BALANCED
## [1] BALANCED
## [1] BALANCED
```

```
# check that models are balanced
unlist(lapply(m.list,ssCheck))
```

```
## Marine Coprophagy (oyster)              Lake Findley
##                       TRUE                      TRUE
##                Mirror Lake                Lake Wingra
##                       TRUE                      TRUE
##                Marion Lake               Cone Springs
##                       TRUE                      TRUE
##             Silver Springs            English Channel
##                       TRUE                      TRUE
##               Oyster Reef               Baie de Somme
##                       TRUE                      TRUE
```

```
## If balancing fails, you can use force.balance
## to repeatedly apply the balancing procedure
## although this is not the case with our model set

m.list <- lapply(m.list,force.balance)
## Check that all the models are balanced
all(unlist(lapply(m.list,ssCheck)))
```

```
## [1] TRUE
```

```
## Example Flow Analysis
F.list <- lapply(m.list, enaFlow)
```

```
## Warning in enaAscendency(x): Export data is absent from the model.

## Warning in enaAscendency(x): Export data is absent from the model.

## Warning in enaAscendency(x): Export data is absent from the model.
```

```
## The full results of the flow analysis is now stored in the elements
## of the F.list.  To get the results for just the first model:
F.list[[1]]
```

```
## $T
##                    SHRIMP        BENTHIC ORGANISMS  SHRIMP FECES & BACTERIA
##                     124.1                    323.7                      21.9
```

```
## BENTHIC FECES & BACTERIA
##                     79.6
##
## $G
##                           SHRIMP BENTHIC ORGANISMS SHRIMP FECES & BACTERIA
## SHRIMP                         0         0.0000000               0.1764706
## BENTHIC ORGANISMS              0         0.0000000               0.0000000
## SHRIMP FECES & BACTERIA        0         0.6986301               0.0000000
## BENTHIC FECES & BACTERIA       0         0.6645729               0.0000000
##                           BENTHIC FECES & BACTERIA
## SHRIMP                                   0.0000000
## BENTHIC ORGANISMS                        0.2459067
## SHRIMP FECES & BACTERIA                  0.0000000
## BENTHIC FECES & BACTERIA                 0.0000000
##
## $GP
##                           SHRIMP BENTHIC ORGANISMS SHRIMP FECES & BACTERIA
## SHRIMP                         0        0.00000000                       1
## BENTHIC ORGANISMS              0        0.00000000                       0
## SHRIMP FECES & BACTERIA        0        0.04726599                       0
## BENTHIC FECES & BACTERIA       0        0.16342292                       0
##                           BENTHIC FECES & BACTERIA
## SHRIMP                                           0
## BENTHIC ORGANISMS                                1
## SHRIMP FECES & BACTERIA                          0
## BENTHIC FECES & BACTERIA                         0
##
## $N
##                           SHRIMP BENTHIC ORGANISMS SHRIMP FECES & BACTERIA
## SHRIMP                         1         0.1473716               0.1764706
## BENTHIC ORGANISMS              0         1.1953471               0.0000000
## SHRIMP FECES & BACTERIA        0         0.8351055               1.0000000
## BENTHIC FECES & BACTERIA       0         0.7943953               0.0000000
##                           BENTHIC FECES & BACTERIA
## SHRIMP                                 0.03623966
## BENTHIC ORGANISMS                      0.29394387
## SHRIMP FECES & BACTERIA                0.20535805
## BENTHIC FECES & BACTERIA               1.19534712
##
## $NP
##                           SHRIMP BENTHIC ORGANISMS SHRIMP FECES & BACTERIA
## SHRIMP                         1        0.05649926                       1
## BENTHIC ORGANISMS              0        1.19534712                       0
## SHRIMP FECES & BACTERIA        0        0.05649926                       1
## BENTHIC FECES & BACTERIA       0        0.19534712                       0
##                           BENTHIC FECES & BACTERIA
## SHRIMP                                 0.05649926
## BENTHIC ORGANISMS                      1.19534712
## SHRIMP FECES & BACTERIA                0.05649926
## BENTHIC FECES & BACTERIA               1.19534712
##
## $ns
##      Boundary   TST  TSTp     APL       FCI       BFI       DFI       IFI
## [1,]    379.6 549.3 928.9 1.44705 0.1199863 0.6910614 0.1542493 0.1546893
```

```
##          ID.F    ID.F.I    ID.F.O    HMG.I    HMG.O AMP.I AMP.O mode0.F
## [1,] 1.002852 0.3603839 0.6126851 2.014161 1.891504     1     0   379.6
##       mode1.F  mode2.F  mode3.F mode4.F        H      AMI       Hr
## [1,] 103.7915 65.90846 103.7915   379.6 2.719296 1.034247 1.685049
##           CAP      ASC       OH   ASC.CAP    OH.CAP robustness      ELD
## [1,] 2525.954 960.7117 1565.242 0.3803362 0.6196638  0.3676709 1.793185
##            TD  A.input A.internal A.export A.respiration OH.input
## [1,] 2.048044 402.8692   249.2812        0      308.5612 433.3118
##      OH.internal OH.export OH.respiration CAP.input CAP.internal
## [1,]    460.6126         0       671.3179   836.181     709.8939
##      CAP.export CAP.respiration
## [1,]          0        979.8791
```

We can use the same technique to extract specific information, like just the ratio of Indirect-to-Direct flow for each model.

```
## Example of extracting just specific information - Indirect Effects Ratio
IDs <- unlist(lapply(m.list, function(x) enaFlow(x)$ns[9]))
```

```
## Warning in enaAscendency(x): Export data is absent from the model.

## Warning in enaAscendency(x): Export data is absent from the model.

## Warning in enaAscendency(x): Export data is absent from the model.

## Warning in enaAscendency(x): Export data is absent from the model.

## Warning in enaAscendency(x): Respiration data is absent from the model.

## Warning in enaAscendency(x): Respiration data is absent from the model.

## Warning in enaAscendency(x): Respiration data is absent from the model.

## Warning in enaAscendency(x): Respiration data is absent from the model.

## Warning in enaAscendency(x): Respiration data is absent from the model.

## Warning in enaAscendency(x): Respiration data is absent from the model.

## Warning in enaAscendency(x): Respiration data is absent from the model.

## Warning in enaAscendency(x): Respiration data is absent from the model.

## Warning in enaAscendency(x): Respiration data is absent from the model.

## Warning in enaAscendency(x): Respiration data is absent from the model.

## Warning in enaAscendency(x): Respiration data is absent from the model.

## Warning in enaAscendency(x): Export data is absent from the model.
```

```
## Warning in enaAscendency(x): Respiration data is absent from the model.

## Warning in enaAscendency(x): Respiration data is absent from the model.

## Warning in enaAscendency(x): Respiration data is absent from the model.

## Warning in enaAscendency(x): Respiration data is absent from the model.
```

```
## Look at the first few ID's
head(IDs)
```

```
## Marine Coprophagy (oyster)              Lake Findley
##                   1.002852                  1.723221
##                Mirror Lake                Lake Wingra
##                   1.861121                  1.861719
##                Marion Lake               Cone Springs
##                   2.175878                  1.023016
```

We can also collect the set of output-oriented integral flow matrices.

```
## Here is a list containing only the
## output-oriented integral flow matrices
N.list <- lapply(m.list,function(x) enaFlow(x)$N)
```

```
## Warning in enaAscendency(x): Export data is absent from the model.

## Warning in enaAscendency(x): Export data is absent from the model.

## Warning in enaAscendency(x): Export data is absent from the model.

## Warning in enaAscendency(x): Export data is absent from the model.

## Warning in enaAscendency(x): Respiration data is absent from the model.

## Warning in enaAscendency(x): Respiration data is absent from the model.

## Warning in enaAscendency(x): Respiration data is absent from the model.

## Warning in enaAscendency(x): Respiration data is absent from the model.

## Warning in enaAscendency(x): Respiration data is absent from the model.

## Warning in enaAscendency(x): Respiration data is absent from the model.

## Warning in enaAscendency(x): Respiration data is absent from the model.

## Warning in enaAscendency(x): Respiration data is absent from the model.

## Warning in enaAscendency(x): Respiration data is absent from the model.

## Warning in enaAscendency(x): Respiration data is absent from the model.
```

```
## Warning in enaAscendency(x): Respiration data is absent from the model.

## Warning in enaAscendency(x): Respiration data is absent from the model.

## Warning in enaAscendency(x): Export data is absent from the model.

## Warning in enaAscendency(x): Respiration data is absent from the model.

## Warning in enaAscendency(x): Respiration data is absent from the model.

## Warning in enaAscendency(x): Respiration data is absent from the model.

## Warning in enaAscendency(x): Respiration data is absent from the model.
```

We can also apply the get.ns function to extract all of the network statistics for each model. We then use the do.call function to reshape the network statistics into a single data frame.

```
## Collecting and combining all network statistics
ns.list <- lapply(m.list,get.ns) # returns as list
```

```
## Node 1, Reach 1, Total 1
## Node 2, Reach 4, Total 5
## Node 3, Reach 2, Total 7
## Node 4, Reach 4, Total 11

## Warning in enaAscendency(x): Export data is absent from the model.

## Node 1, Reach 2, Total 2
## Node 2, Reach 4, Total 6
## Node 3, Reach 1, Total 7
## Node 4, Reach 4, Total 11
## Node 1, Reach 2, Total 2
## Node 2, Reach 5, Total 7
## Node 3, Reach 5, Total 12
## Node 4, Reach 1, Total 13
## Node 5, Reach 5, Total 18
## Node 1, Reach 5, Total 5
## Node 2, Reach 5, Total 10
## Node 3, Reach 5, Total 15
## Node 4, Reach 5, Total 20
## Node 5, Reach 5, Total 25
## Node 1, Reach 2, Total 2
## Node 2, Reach 5, Total 7
## Node 3, Reach 5, Total 12
## Node 4, Reach 1, Total 13
## Node 5, Reach 5, Total 18
## Node 1, Reach 1, Total 1
## Node 2, Reach 5, Total 6
## Node 3, Reach 5, Total 11
## Node 4, Reach 5, Total 16
## Node 5, Reach 5, Total 21
## Node 1, Reach 1, Total 1
```

```
## Node 2, Reach 2, Total 3
## Node 3, Reach 3, Total 6
## Node 4, Reach 4, Total 10
## Node 5, Reach 5, Total 15
## Node 1, Reach 1, Total 1
## Node 2, Reach 2, Total 3
## Node 3, Reach 3, Total 6
## Node 4, Reach 3, Total 9
## Node 5, Reach 4, Total 13
## Node 6, Reach 6, Total 19

## Warning in enaAscendency(x): Export data is absent from the model.

## Node 1, Reach 1, Total 1
## Node 2, Reach 6, Total 7
## Node 3, Reach 6, Total 13
## Node 4, Reach 6, Total 19
## Node 5, Reach 6, Total 25
## Node 6, Reach 6, Total 31

## Warning in enaAscendency(x): Export data is absent from the model.

## Node 1, Reach 1, Total 1
## Node 2, Reach 1, Total 2
## Node 3, Reach 9, Total 11
## Node 4, Reach 9, Total 20
## Node 5, Reach 9, Total 29
## Node 6, Reach 9, Total 38
## Node 7, Reach 9, Total 47
## Node 8, Reach 9, Total 56
## Node 9, Reach 9, Total 65
## Node 1, Reach 13, Total 13
## Node 2, Reach 13, Total 26
## Node 3, Reach 13, Total 39
## Node 4, Reach 13, Total 52
## Node 5, Reach 13, Total 65
## Node 6, Reach 13, Total 78
## Node 7, Reach 13, Total 91
## Node 8, Reach 13, Total 104
## Node 9, Reach 13, Total 117
## Node 10, Reach 13, Total 130
## Node 11, Reach 13, Total 143
## Node 12, Reach 13, Total 156
## Node 13, Reach 13, Total 169
## Node 1, Reach 1, Total 1
## Node 2, Reach 1, Total 2
## Node 3, Reach 7, Total 9
## Node 4, Reach 7, Total 16
## Node 5, Reach 7, Total 23
## Node 6, Reach 7, Total 30
## Node 7, Reach 8, Total 38
## Node 8, Reach 15, Total 53
## Node 9, Reach 15, Total 68
```

```
## Node 10, Reach 15, Total 83
## Node 11, Reach 15, Total 98
## Node 12, Reach 15, Total 113
## Node 13, Reach 2, Total 115
## Node 14, Reach 7, Total 122
## Node 15, Reach 15, Total 137
## Node 1, Reach 1, Total 1
## Node 2, Reach 1, Total 2
## Node 3, Reach 3, Total 5
## Node 4, Reach 4, Total 9
## Node 5, Reach 15, Total 24
## Node 6, Reach 15, Total 39
## Node 7, Reach 15, Total 54
## Node 8, Reach 15, Total 69
## Node 9, Reach 15, Total 84
## Node 10, Reach 15, Total 99
## Node 11, Reach 15, Total 114
## Node 12, Reach 15, Total 129
## Node 13, Reach 2, Total 131
## Node 14, Reach 15, Total 146
## Node 15, Reach 15, Total 161
## Node 1, Reach 1, Total 1
## Node 2, Reach 1, Total 2
## Node 3, Reach 21, Total 23
## Node 4, Reach 21, Total 44
## Node 5, Reach 21, Total 65
## Node 6, Reach 21, Total 86
## Node 7, Reach 21, Total 107
## Node 8, Reach 21, Total 128
## Node 9, Reach 21, Total 149
## Node 10, Reach 21, Total 170
## Node 11, Reach 21, Total 191
## Node 12, Reach 21, Total 212
## Node 13, Reach 21, Total 233
## Node 14, Reach 21, Total 254
## Node 15, Reach 21, Total 275
## Node 16, Reach 21, Total 296
## Node 17, Reach 21, Total 317
## Node 18, Reach 21, Total 338
## Node 19, Reach 21, Total 359
## Node 20, Reach 21, Total 380
## Node 21, Reach 21, Total 401
## Node 1, Reach 1, Total 1
## Node 2, Reach 1, Total 2
## Node 3, Reach 21, Total 23
## Node 4, Reach 21, Total 44
## Node 5, Reach 21, Total 65
## Node 6, Reach 21, Total 86
## Node 7, Reach 21, Total 107
## Node 8, Reach 21, Total 128
## Node 9, Reach 21, Total 149
## Node 10, Reach 21, Total 170
## Node 11, Reach 21, Total 191
## Node 12, Reach 21, Total 212
```

```
## Node 13, Reach 2, Total 214
## Node 14, Reach 21, Total 235
## Node 15, Reach 21, Total 256
## Node 16, Reach 21, Total 277
## Node 17, Reach 21, Total 298
## Node 18, Reach 21, Total 319
## Node 19, Reach 21, Total 340
## Node 20, Reach 21, Total 361
## Node 21, Reach 21, Total 382
## Node 1, Reach 1, Total 1
## Node 2, Reach 1, Total 2
## Node 3, Reach 1, Total 3
## Node 4, Reach 1, Total 4
## Node 5, Reach 1, Total 5
## Node 6, Reach 1, Total 6
## Node 7, Reach 7, Total 13
## Node 8, Reach 7, Total 20
## Node 9, Reach 7, Total 27
## Node 10, Reach 8, Total 35
## Node 11, Reach 21, Total 56
## Node 12, Reach 8, Total 64
## Node 13, Reach 21, Total 85
## Node 14, Reach 21, Total 106
## Node 15, Reach 21, Total 127
## Node 16, Reach 3, Total 130
## Node 17, Reach 21, Total 151
## Node 18, Reach 21, Total 172
## Node 19, Reach 7, Total 179
## Node 20, Reach 7, Total 186
## Node 21, Reach 21, Total 207
## Node 1, Reach 26, Total 26
## Node 2, Reach 26, Total 52
## Node 3, Reach 26, Total 78
## Node 4, Reach 26, Total 104
## Node 5, Reach 26, Total 130
## Node 6, Reach 26, Total 156
## Node 7, Reach 26, Total 182
## Node 8, Reach 26, Total 208
## Node 9, Reach 26, Total 234
## Node 10, Reach 26, Total 260
## Node 11, Reach 26, Total 286
## Node 12, Reach 26, Total 312
## Node 13, Reach 26, Total 338
## Node 14, Reach 26, Total 364
## Node 15, Reach 26, Total 390
## Node 16, Reach 26, Total 416
## Node 17, Reach 26, Total 442
## Node 18, Reach 26, Total 468
## Node 19, Reach 26, Total 494
## Node 20, Reach 26, Total 520
## Node 21, Reach 26, Total 546
## Node 22, Reach 26, Total 572
## Node 23, Reach 26, Total 598
## Node 24, Reach 26, Total 624
```

```
## Node 25, Reach 26, Total 650
## Node 26, Reach 26, Total 676

## Warning in enaAscendency(x): Export data is absent from the model.

## Node 1, Reach 1, Total 1
## Node 2, Reach 24, Total 25
## Node 3, Reach 1, Total 26
## Node 4, Reach 24, Total 50
## Node 5, Reach 24, Total 74
## Node 6, Reach 1, Total 75
## Node 7, Reach 1, Total 76
## Node 8, Reach 24, Total 100
## Node 9, Reach 24, Total 124
## Node 10, Reach 24, Total 148
## Node 11, Reach 1, Total 149
## Node 12, Reach 24, Total 173
## Node 13, Reach 24, Total 197
## Node 14, Reach 24, Total 221
## Node 15, Reach 24, Total 245
## Node 16, Reach 24, Total 269
## Node 17, Reach 24, Total 293
## Node 18, Reach 24, Total 317
## Node 19, Reach 1, Total 318
## Node 20, Reach 24, Total 342
## Node 21, Reach 24, Total 366
## Node 22, Reach 24, Total 390
## Node 23, Reach 24, Total 414
## Node 24, Reach 24, Total 438
## Node 25, Reach 24, Total 462
## Node 26, Reach 24, Total 486
## Node 27, Reach 24, Total 510
## Node 28, Reach 24, Total 534
## Node 29, Reach 1, Total 535
## Node 30, Reach 1, Total 536
## Node 1, Reach 1, Total 1
## Node 2, Reach 31, Total 32
## Node 3, Reach 31, Total 63
## Node 4, Reach 31, Total 94
## Node 5, Reach 31, Total 125
## Node 6, Reach 31, Total 156
## Node 7, Reach 31, Total 187
## Node 8, Reach 31, Total 218
## Node 9, Reach 31, Total 249
## Node 10, Reach 31, Total 280
## Node 11, Reach 31, Total 311
## Node 12, Reach 31, Total 342
## Node 13, Reach 31, Total 373
## Node 14, Reach 31, Total 404
## Node 15, Reach 31, Total 435
## Node 16, Reach 31, Total 466
## Node 17, Reach 31, Total 497
## Node 18, Reach 31, Total 528
## Node 19, Reach 31, Total 559
```

```
## Node 20, Reach 31, Total 590
## Node 21, Reach 31, Total 621
## Node 22, Reach 31, Total 652
## Node 23, Reach 31, Total 683
## Node 24, Reach 31, Total 714
## Node 25, Reach 31, Total 745
## Node 26, Reach 31, Total 776
## Node 27, Reach 31, Total 807
## Node 28, Reach 31, Total 838
## Node 29, Reach 31, Total 869
## Node 30, Reach 31, Total 900
## Node 31, Reach 31, Total 931
## Node 1, Reach 1, Total 1
## Node 2, Reach 31, Total 32
## Node 3, Reach 31, Total 63
## Node 4, Reach 31, Total 94
## Node 5, Reach 31, Total 125
## Node 6, Reach 31, Total 156
## Node 7, Reach 31, Total 187
## Node 8, Reach 31, Total 218
## Node 9, Reach 31, Total 249
## Node 10, Reach 31, Total 280
## Node 11, Reach 31, Total 311
## Node 12, Reach 31, Total 342
## Node 13, Reach 31, Total 373
## Node 14, Reach 31, Total 404
## Node 15, Reach 31, Total 435
## Node 16, Reach 31, Total 466
## Node 17, Reach 31, Total 497
## Node 18, Reach 31, Total 528
## Node 19, Reach 31, Total 559
## Node 20, Reach 31, Total 590
## Node 21, Reach 31, Total 621
## Node 22, Reach 31, Total 652
## Node 23, Reach 31, Total 683
## Node 24, Reach 31, Total 714
## Node 25, Reach 31, Total 745
## Node 26, Reach 31, Total 776
## Node 27, Reach 31, Total 807
## Node 28, Reach 31, Total 838
## Node 29, Reach 31, Total 869
## Node 30, Reach 31, Total 900
## Node 31, Reach 31, Total 931
## Node 1, Reach 1, Total 1
## Node 2, Reach 32, Total 33
## Node 3, Reach 32, Total 65
## Node 4, Reach 32, Total 97
## Node 5, Reach 32, Total 129
## Node 6, Reach 32, Total 161
## Node 7, Reach 32, Total 193
## Node 8, Reach 32, Total 225
## Node 9, Reach 32, Total 257
## Node 10, Reach 32, Total 289
## Node 11, Reach 32, Total 321
```

```
## Node 12, Reach 32, Total 353
## Node 13, Reach 32, Total 385
## Node 14, Reach 32, Total 417
## Node 15, Reach 32, Total 449
## Node 16, Reach 32, Total 481
## Node 17, Reach 32, Total 513
## Node 18, Reach 32, Total 545
## Node 19, Reach 32, Total 577
## Node 20, Reach 32, Total 609
## Node 21, Reach 32, Total 641
## Node 22, Reach 32, Total 673
## Node 23, Reach 32, Total 705
## Node 24, Reach 32, Total 737
## Node 25, Reach 32, Total 769
## Node 26, Reach 32, Total 801
## Node 27, Reach 32, Total 833
## Node 28, Reach 32, Total 865
## Node 29, Reach 32, Total 897
## Node 30, Reach 32, Total 929
## Node 31, Reach 32, Total 961
## Node 32, Reach 32, Total 993
## Node 1, Reach 32, Total 32
## Node 2, Reach 32, Total 64
## Node 3, Reach 32, Total 96
## Node 4, Reach 32, Total 128
## Node 5, Reach 32, Total 160
## Node 6, Reach 32, Total 192
## Node 7, Reach 32, Total 224
## Node 8, Reach 32, Total 256
## Node 9, Reach 32, Total 288
## Node 10, Reach 32, Total 320
## Node 11, Reach 32, Total 352
## Node 12, Reach 32, Total 384
## Node 13, Reach 32, Total 416
## Node 14, Reach 32, Total 448
## Node 15, Reach 32, Total 480
## Node 16, Reach 32, Total 512
## Node 17, Reach 32, Total 544
## Node 18, Reach 32, Total 576
## Node 19, Reach 32, Total 608
## Node 20, Reach 32, Total 640
## Node 21, Reach 32, Total 672
## Node 22, Reach 32, Total 704
## Node 23, Reach 32, Total 736
## Node 24, Reach 32, Total 768
## Node 25, Reach 32, Total 800
## Node 26, Reach 32, Total 832
## Node 27, Reach 32, Total 864
## Node 28, Reach 32, Total 896
## Node 29, Reach 32, Total 928
## Node 30, Reach 1, Total 929
## Node 31, Reach 1, Total 930
## Node 32, Reach 32, Total 962
## Node 1, Reach 1, Total 1
```

```
## Node 2, Reach 33, Total 34
## Node 3, Reach 33, Total 67
## Node 4, Reach 33, Total 100
## Node 5, Reach 33, Total 133
## Node 6, Reach 33, Total 166
## Node 7, Reach 33, Total 199
## Node 8, Reach 33, Total 232
## Node 9, Reach 33, Total 265
## Node 10, Reach 33, Total 298
## Node 11, Reach 33, Total 331
## Node 12, Reach 33, Total 364
## Node 13, Reach 33, Total 397
## Node 14, Reach 33, Total 430
## Node 15, Reach 33, Total 463
## Node 16, Reach 33, Total 496
## Node 17, Reach 33, Total 529
## Node 18, Reach 33, Total 562
## Node 19, Reach 33, Total 595
## Node 20, Reach 33, Total 628
## Node 21, Reach 33, Total 661
## Node 22, Reach 33, Total 694
## Node 23, Reach 33, Total 727
## Node 24, Reach 33, Total 760
## Node 25, Reach 33, Total 793
## Node 26, Reach 33, Total 826
## Node 27, Reach 33, Total 859
## Node 28, Reach 33, Total 892
## Node 29, Reach 33, Total 925
## Node 30, Reach 33, Total 958
## Node 31, Reach 33, Total 991
## Node 32, Reach 33, Total 1024
## Node 33, Reach 33, Total 1057
## Node 1, Reach 1, Total 1
## Node 2, Reach 10, Total 11
## Node 3, Reach 36, Total 47
## Node 4, Reach 1, Total 48
## Node 5, Reach 3, Total 51
## Node 6, Reach 4, Total 55
## Node 7, Reach 10, Total 65
## Node 8, Reach 10, Total 75
## Node 9, Reach 10, Total 85
## Node 10, Reach 10, Total 95
## Node 11, Reach 11, Total 106
## Node 12, Reach 11, Total 117
## Node 13, Reach 11, Total 128
## Node 14, Reach 36, Total 164
## Node 15, Reach 36, Total 200
## Node 16, Reach 36, Total 236
## Node 17, Reach 36, Total 272
## Node 18, Reach 36, Total 308
## Node 19, Reach 36, Total 344
## Node 20, Reach 11, Total 355
## Node 21, Reach 11, Total 366
## Node 22, Reach 11, Total 377
```

```
## Node 23, Reach 11, Total 388
## Node 24, Reach 11, Total 399
## Node 25, Reach 36, Total 435
## Node 26, Reach 36, Total 471
## Node 27, Reach 36, Total 507
## Node 28, Reach 36, Total 543
## Node 29, Reach 36, Total 579
## Node 30, Reach 36, Total 615
## Node 31, Reach 12, Total 627
## Node 32, Reach 36, Total 663
## Node 33, Reach 36, Total 699
## Node 34, Reach 2, Total 701
## Node 35, Reach 10, Total 711
## Node 36, Reach 36, Total 747
## Node 1, Reach 1, Total 1
## Node 2, Reach 1, Total 2
## Node 3, Reach 1, Total 3
## Node 4, Reach 1, Total 4
## Node 5, Reach 1, Total 5
## Node 6, Reach 1, Total 6
## Node 7, Reach 2, Total 8
## Node 8, Reach 43, Total 51
## Node 9, Reach 43, Total 94
## Node 10, Reach 43, Total 137
## Node 11, Reach 43, Total 180
## Node 12, Reach 43, Total 223
## Node 13, Reach 43, Total 266
## Node 14, Reach 43, Total 309
## Node 15, Reach 43, Total 352
## Node 16, Reach 43, Total 395
## Node 17, Reach 43, Total 438
## Node 18, Reach 43, Total 481
## Node 19, Reach 43, Total 524
## Node 20, Reach 43, Total 567
## Node 21, Reach 43, Total 610
## Node 22, Reach 43, Total 653
## Node 23, Reach 43, Total 696
## Node 24, Reach 43, Total 739
## Node 25, Reach 43, Total 782
## Node 26, Reach 43, Total 825
## Node 27, Reach 43, Total 868
## Node 28, Reach 43, Total 911
## Node 29, Reach 43, Total 954
## Node 30, Reach 43, Total 997
## Node 31, Reach 43, Total 1040
## Node 32, Reach 3, Total 1043
## Node 33, Reach 43, Total 1086
## Node 34, Reach 43, Total 1129
## Node 35, Reach 43, Total 1172
## Node 36, Reach 43, Total 1215
## Node 37, Reach 43, Total 1258
## Node 38, Reach 43, Total 1301
## Node 39, Reach 43, Total 1344
## Node 40, Reach 43, Total 1387
```

```
## Node 41, Reach 43, Total 1430
## Node 42, Reach 43, Total 1473
## Node 43, Reach 43, Total 1516
## Node 1, Reach 1, Total 1
## Node 2, Reach 1, Total 2
## Node 3, Reach 1, Total 3
## Node 4, Reach 1, Total 4
## Node 5, Reach 1, Total 5
## Node 6, Reach 7, Total 12
## Node 7, Reach 38, Total 50
## Node 8, Reach 38, Total 88
## Node 9, Reach 38, Total 126
## Node 10, Reach 38, Total 164
## Node 11, Reach 38, Total 202
## Node 12, Reach 38, Total 240
## Node 13, Reach 38, Total 278
## Node 14, Reach 38, Total 316
## Node 15, Reach 38, Total 354
## Node 16, Reach 38, Total 392
## Node 17, Reach 38, Total 430
## Node 18, Reach 1, Total 431
## Node 19, Reach 38, Total 469
## Node 20, Reach 38, Total 507
## Node 21, Reach 4, Total 511
## Node 22, Reach 38, Total 549
## Node 23, Reach 3, Total 552
## Node 24, Reach 38, Total 590
## Node 25, Reach 38, Total 628
## Node 26, Reach 38, Total 666
## Node 27, Reach 38, Total 704
## Node 28, Reach 38, Total 742
## Node 29, Reach 2, Total 744
## Node 30, Reach 2, Total 746
## Node 31, Reach 1, Total 747
## Node 32, Reach 1, Total 748
## Node 33, Reach 1, Total 749
## Node 34, Reach 38, Total 787
## Node 35, Reach 1, Total 788
## Node 36, Reach 1, Total 789
## Node 37, Reach 38, Total 827
## Node 38, Reach 38, Total 865
## Node 39, Reach 38, Total 903
## Node 40, Reach 38, Total 941
## Node 41, Reach 1, Total 942
## Node 42, Reach 38, Total 980
## Node 43, Reach 1, Total 981
## Node 44, Reach 39, Total 1020
## Node 45, Reach 39, Total 1059
## Node 46, Reach 39, Total 1098
## Node 47, Reach 39, Total 1137
## Node 48, Reach 39, Total 1176
## Node 49, Reach 6, Total 1182
## Node 50, Reach 38, Total 1220
## Node 51, Reach 38, Total 1258
```

```
## Node 1, Reach 1, Total 1
## Node 2, Reach 1, Total 2
## Node 3, Reach 1, Total 3
## Node 4, Reach 1, Total 4
## Node 5, Reach 1, Total 5
## Node 6, Reach 7, Total 12
## Node 7, Reach 42, Total 54
## Node 8, Reach 42, Total 96
## Node 9, Reach 42, Total 138
## Node 10, Reach 42, Total 180
## Node 11, Reach 42, Total 222
## Node 12, Reach 42, Total 264
## Node 13, Reach 42, Total 306
## Node 14, Reach 42, Total 348
## Node 15, Reach 42, Total 390
## Node 16, Reach 42, Total 432
## Node 17, Reach 42, Total 474
## Node 18, Reach 4, Total 478
## Node 19, Reach 42, Total 520
## Node 20, Reach 42, Total 562
## Node 21, Reach 3, Total 565
## Node 22, Reach 42, Total 607
## Node 23, Reach 2, Total 609
## Node 24, Reach 42, Total 651
## Node 25, Reach 42, Total 693
## Node 26, Reach 42, Total 735
## Node 27, Reach 42, Total 777
## Node 28, Reach 42, Total 819
## Node 29, Reach 2, Total 821
## Node 30, Reach 2, Total 823
## Node 31, Reach 42, Total 865
## Node 32, Reach 42, Total 907
## Node 33, Reach 42, Total 949
## Node 34, Reach 42, Total 991
## Node 35, Reach 42, Total 1033
## Node 36, Reach 42, Total 1075
## Node 37, Reach 42, Total 1117
## Node 38, Reach 42, Total 1159
## Node 39, Reach 42, Total 1201
## Node 40, Reach 1, Total 1202
## Node 41, Reach 1, Total 1203
## Node 42, Reach 1, Total 1204
## Node 43, Reach 1, Total 1205
## Node 44, Reach 43, Total 1248
## Node 45, Reach 43, Total 1291
## Node 46, Reach 1, Total 1292
## Node 47, Reach 43, Total 1335
## Node 48, Reach 43, Total 1378
## Node 49, Reach 6, Total 1384
## Node 50, Reach 42, Total 1426
## Node 51, Reach 42, Total 1468
## Node 1, Reach 1, Total 1
## Node 2, Reach 1, Total 2
## Node 3, Reach 1, Total 3
```

```
## Node 4, Reach 1, Total 4
## Node 5, Reach 1, Total 5
## Node 6, Reach 7, Total 12
## Node 7, Reach 35, Total 47
## Node 8, Reach 35, Total 82
## Node 9, Reach 35, Total 117
## Node 10, Reach 35, Total 152
## Node 11, Reach 35, Total 187
## Node 12, Reach 35, Total 222
## Node 13, Reach 35, Total 257
## Node 14, Reach 35, Total 292
## Node 15, Reach 35, Total 327
## Node 16, Reach 35, Total 362
## Node 17, Reach 35, Total 397
## Node 18, Reach 1, Total 398
## Node 19, Reach 35, Total 433
## Node 20, Reach 35, Total 468
## Node 21, Reach 3, Total 471
## Node 22, Reach 35, Total 506
## Node 23, Reach 3, Total 509
## Node 24, Reach 35, Total 544
## Node 25, Reach 35, Total 579
## Node 26, Reach 35, Total 614
## Node 27, Reach 35, Total 649
## Node 28, Reach 35, Total 684
## Node 29, Reach 1, Total 685
## Node 30, Reach 2, Total 687
## Node 31, Reach 1, Total 688
## Node 32, Reach 1, Total 689
## Node 33, Reach 1, Total 690
## Node 34, Reach 1, Total 691
## Node 35, Reach 1, Total 692
## Node 36, Reach 1, Total 693
## Node 37, Reach 1, Total 694
## Node 38, Reach 35, Total 729
## Node 39, Reach 1, Total 730
## Node 40, Reach 35, Total 765
## Node 41, Reach 35, Total 800
## Node 42, Reach 35, Total 835
## Node 43, Reach 1, Total 836
## Node 44, Reach 36, Total 872
## Node 45, Reach 36, Total 908
## Node 46, Reach 36, Total 944
## Node 47, Reach 36, Total 980
## Node 48, Reach 36, Total 1016
## Node 49, Reach 6, Total 1022
## Node 50, Reach 35, Total 1057
## Node 51, Reach 35, Total 1092
## Node 1, Reach 1, Total 1
## Node 2, Reach 1, Total 2
## Node 3, Reach 1, Total 3
## Node 4, Reach 1, Total 4
## Node 5, Reach 1, Total 5
## Node 6, Reach 7, Total 12
```

```
## Node 7, Reach 40, Total 52
## Node 8, Reach 40, Total 92
## Node 9, Reach 40, Total 132
## Node 10, Reach 40, Total 172
## Node 11, Reach 40, Total 212
## Node 12, Reach 40, Total 252
## Node 13, Reach 40, Total 292
## Node 14, Reach 40, Total 332
## Node 15, Reach 40, Total 372
## Node 16, Reach 40, Total 412
## Node 17, Reach 40, Total 452
## Node 18, Reach 4, Total 456
## Node 19, Reach 40, Total 496
## Node 20, Reach 40, Total 536
## Node 21, Reach 3, Total 539
## Node 22, Reach 40, Total 579
## Node 23, Reach 2, Total 581
## Node 24, Reach 40, Total 621
## Node 25, Reach 40, Total 661
## Node 26, Reach 40, Total 701
## Node 27, Reach 40, Total 741
## Node 28, Reach 40, Total 781
## Node 29, Reach 1, Total 782
## Node 30, Reach 2, Total 784
## Node 31, Reach 1, Total 785
## Node 32, Reach 40, Total 825
## Node 33, Reach 40, Total 865
## Node 34, Reach 1, Total 866
## Node 35, Reach 40, Total 906
## Node 36, Reach 1, Total 907
## Node 37, Reach 40, Total 947
## Node 38, Reach 40, Total 987
## Node 39, Reach 40, Total 1027
## Node 40, Reach 1, Total 1028
## Node 41, Reach 1, Total 1029
## Node 42, Reach 1, Total 1030
## Node 43, Reach 41, Total 1071
## Node 44, Reach 41, Total 1112
## Node 45, Reach 41, Total 1153
## Node 46, Reach 41, Total 1194
## Node 47, Reach 41, Total 1235
## Node 48, Reach 41, Total 1276
## Node 49, Reach 6, Total 1282
## Node 50, Reach 40, Total 1322
## Node 51, Reach 40, Total 1362
## Node 1, Reach 1, Total 1
## Node 2, Reach 1, Total 2
## Node 3, Reach 1, Total 3
## Node 4, Reach 1, Total 4
## Node 5, Reach 1, Total 5
## Node 6, Reach 7, Total 12
## Node 7, Reach 30, Total 42
## Node 8, Reach 30, Total 72
## Node 9, Reach 30, Total 102
```

```
## Node 10, Reach 1, Total 103
## Node 11, Reach 1, Total 104
## Node 12, Reach 30, Total 134
## Node 13, Reach 30, Total 164
## Node 14, Reach 30, Total 194
## Node 15, Reach 30, Total 224
## Node 16, Reach 30, Total 254
## Node 17, Reach 1, Total 255
## Node 18, Reach 1, Total 256
## Node 19, Reach 30, Total 286
## Node 20, Reach 1, Total 287
## Node 21, Reach 3, Total 290
## Node 22, Reach 30, Total 320
## Node 23, Reach 1, Total 321
## Node 24, Reach 30, Total 351
## Node 25, Reach 30, Total 381
## Node 26, Reach 1, Total 382
## Node 27, Reach 30, Total 412
## Node 28, Reach 30, Total 442
## Node 29, Reach 2, Total 444
## Node 30, Reach 1, Total 445
## Node 31, Reach 1, Total 446
## Node 32, Reach 1, Total 447
## Node 33, Reach 1, Total 448
## Node 34, Reach 30, Total 478
## Node 35, Reach 1, Total 479
## Node 36, Reach 1, Total 480
## Node 37, Reach 30, Total 510
## Node 38, Reach 30, Total 540
## Node 39, Reach 1, Total 541
## Node 40, Reach 30, Total 571
## Node 41, Reach 1, Total 572
## Node 42, Reach 30, Total 602
## Node 43, Reach 1, Total 603
## Node 44, Reach 1, Total 604
## Node 45, Reach 31, Total 635
## Node 46, Reach 1, Total 636
## Node 47, Reach 31, Total 667
## Node 48, Reach 1, Total 668
## Node 49, Reach 6, Total 674
## Node 50, Reach 30, Total 704
## Node 51, Reach 30, Total 734
## Node 1, Reach 1, Total 1
## Node 2, Reach 1, Total 2
## Node 3, Reach 1, Total 3
## Node 4, Reach 1, Total 4
## Node 5, Reach 1, Total 5
## Node 6, Reach 7, Total 12
## Node 7, Reach 42, Total 54
## Node 8, Reach 42, Total 96
## Node 9, Reach 42, Total 138
## Node 10, Reach 42, Total 180
## Node 11, Reach 42, Total 222
## Node 12, Reach 42, Total 264
```

```
## Node 13, Reach 42, Total 306
## Node 14, Reach 42, Total 348
## Node 15, Reach 42, Total 390
## Node 16, Reach 42, Total 432
## Node 17, Reach 42, Total 474
## Node 18, Reach 4, Total 478
## Node 19, Reach 42, Total 520
## Node 20, Reach 42, Total 562
## Node 21, Reach 3, Total 565
## Node 22, Reach 42, Total 607
## Node 23, Reach 1, Total 608
## Node 24, Reach 42, Total 650
## Node 25, Reach 42, Total 692
## Node 26, Reach 42, Total 734
## Node 27, Reach 42, Total 776
## Node 28, Reach 42, Total 818
## Node 29, Reach 2, Total 820
## Node 30, Reach 2, Total 822
## Node 31, Reach 1, Total 823
## Node 32, Reach 42, Total 865
## Node 33, Reach 1, Total 866
## Node 34, Reach 42, Total 908
## Node 35, Reach 42, Total 950
## Node 36, Reach 42, Total 992
## Node 37, Reach 42, Total 1034
## Node 38, Reach 42, Total 1076
## Node 39, Reach 42, Total 1118
## Node 40, Reach 1, Total 1119
## Node 41, Reach 1, Total 1120
## Node 42, Reach 1, Total 1121
## Node 43, Reach 43, Total 1164
## Node 44, Reach 43, Total 1207
## Node 45, Reach 43, Total 1250
## Node 46, Reach 1, Total 1251
## Node 47, Reach 43, Total 1294
## Node 48, Reach 1, Total 1295
## Node 49, Reach 6, Total 1301
## Node 50, Reach 42, Total 1343
## Node 51, Reach 42, Total 1385
## Node 1, Reach 66, Total 66
## Node 2, Reach 66, Total 132
## Node 3, Reach 1, Total 133
## Node 4, Reach 1, Total 134
## Node 5, Reach 66, Total 200
## Node 6, Reach 1, Total 201
## Node 7, Reach 66, Total 267
## Node 8, Reach 66, Total 333
## Node 9, Reach 66, Total 399
## Node 10, Reach 66, Total 465
## Node 11, Reach 66, Total 531
## Node 12, Reach 66, Total 597
## Node 13, Reach 66, Total 663
## Node 14, Reach 66, Total 729
## Node 15, Reach 66, Total 795
```

```
## Node 16, Reach 66, Total 861
## Node 17, Reach 66, Total 927
## Node 18, Reach 66, Total 993
## Node 19, Reach 66, Total 1059
## Node 20, Reach 66, Total 1125
## Node 21, Reach 66, Total 1191
## Node 22, Reach 66, Total 1257
## Node 23, Reach 66, Total 1323
## Node 24, Reach 66, Total 1389
## Node 25, Reach 66, Total 1455
## Node 26, Reach 66, Total 1521
## Node 27, Reach 66, Total 1587
## Node 28, Reach 66, Total 1653
## Node 29, Reach 66, Total 1719
## Node 30, Reach 66, Total 1785
## Node 31, Reach 66, Total 1851
## Node 32, Reach 66, Total 1917
## Node 33, Reach 66, Total 1983
## Node 34, Reach 66, Total 2049
## Node 35, Reach 66, Total 2115
## Node 36, Reach 66, Total 2181
## Node 37, Reach 66, Total 2247
## Node 38, Reach 66, Total 2313
## Node 39, Reach 66, Total 2379
## Node 40, Reach 66, Total 2445
## Node 41, Reach 66, Total 2511
## Node 42, Reach 66, Total 2577
## Node 43, Reach 66, Total 2643
## Node 44, Reach 66, Total 2709
## Node 45, Reach 66, Total 2775
## Node 46, Reach 3, Total 2778
## Node 47, Reach 66, Total 2844
## Node 48, Reach 2, Total 2846
## Node 49, Reach 66, Total 2912
## Node 50, Reach 66, Total 2978
## Node 51, Reach 66, Total 3044
## Node 52, Reach 66, Total 3110
## Node 53, Reach 3, Total 3113
## Node 54, Reach 66, Total 3179
## Node 55, Reach 66, Total 3245
## Node 56, Reach 66, Total 3311
## Node 57, Reach 66, Total 3377
## Node 58, Reach 66, Total 3443
## Node 59, Reach 66, Total 3509
## Node 60, Reach 66, Total 3575
## Node 61, Reach 66, Total 3641
## Node 62, Reach 66, Total 3707
## Node 63, Reach 66, Total 3773
## Node 64, Reach 66, Total 3839
## Node 65, Reach 66, Total 3905
## Node 66, Reach 66, Total 3971
## Node 1, Reach 66, Total 66
## Node 2, Reach 66, Total 132
## Node 3, Reach 1, Total 133
```

```
## Node 4, Reach 1, Total 134
## Node 5, Reach 66, Total 200
## Node 6, Reach 1, Total 201
## Node 7, Reach 66, Total 267
## Node 8, Reach 66, Total 333
## Node 9, Reach 66, Total 399
## Node 10, Reach 66, Total 465
## Node 11, Reach 66, Total 531
## Node 12, Reach 66, Total 597
## Node 13, Reach 66, Total 663
## Node 14, Reach 66, Total 729
## Node 15, Reach 66, Total 795
## Node 16, Reach 66, Total 861
## Node 17, Reach 66, Total 927
## Node 18, Reach 66, Total 993
## Node 19, Reach 66, Total 1059
## Node 20, Reach 66, Total 1125
## Node 21, Reach 66, Total 1191
## Node 22, Reach 66, Total 1257
## Node 23, Reach 66, Total 1323
## Node 24, Reach 66, Total 1389
## Node 25, Reach 66, Total 1455
## Node 26, Reach 66, Total 1521
## Node 27, Reach 66, Total 1587
## Node 28, Reach 66, Total 1653
## Node 29, Reach 66, Total 1719
## Node 30, Reach 66, Total 1785
## Node 31, Reach 66, Total 1851
## Node 32, Reach 66, Total 1917
## Node 33, Reach 66, Total 1983
## Node 34, Reach 66, Total 2049
## Node 35, Reach 66, Total 2115
## Node 36, Reach 66, Total 2181
## Node 37, Reach 66, Total 2247
## Node 38, Reach 66, Total 2313
## Node 39, Reach 66, Total 2379
## Node 40, Reach 66, Total 2445
## Node 41, Reach 66, Total 2511
## Node 42, Reach 66, Total 2577
## Node 43, Reach 66, Total 2643
## Node 44, Reach 66, Total 2709
## Node 45, Reach 66, Total 2775
## Node 46, Reach 3, Total 2778
## Node 47, Reach 66, Total 2844
## Node 48, Reach 2, Total 2846
## Node 49, Reach 66, Total 2912
## Node 50, Reach 66, Total 2978
## Node 51, Reach 66, Total 3044
## Node 52, Reach 66, Total 3110
## Node 53, Reach 3, Total 3113
## Node 54, Reach 66, Total 3179
## Node 55, Reach 66, Total 3245
## Node 56, Reach 66, Total 3311
## Node 57, Reach 66, Total 3377
```

```
## Node 58, Reach 66, Total 3443
## Node 59, Reach 66, Total 3509
## Node 60, Reach 66, Total 3575
## Node 61, Reach 66, Total 3641
## Node 62, Reach 66, Total 3707
## Node 63, Reach 66, Total 3773
## Node 64, Reach 66, Total 3839
## Node 65, Reach 66, Total 3905
## Node 66, Reach 66, Total 3971
## Node 1, Reach 68, Total 68
## Node 2, Reach 68, Total 136
## Node 3, Reach 1, Total 137
## Node 4, Reach 1, Total 138
## Node 5, Reach 1, Total 139
## Node 6, Reach 1, Total 140
## Node 7, Reach 1, Total 141
## Node 8, Reach 1, Total 142
## Node 9, Reach 1, Total 143
## Node 10, Reach 1, Total 144
## Node 11, Reach 1, Total 145
## Node 12, Reach 1, Total 146
## Node 13, Reach 1, Total 147
## Node 14, Reach 1, Total 148
## Node 15, Reach 68, Total 216
## Node 16, Reach 68, Total 284
## Node 17, Reach 68, Total 352
## Node 18, Reach 68, Total 420
## Node 19, Reach 68, Total 488
## Node 20, Reach 68, Total 556
## Node 21, Reach 68, Total 624
## Node 22, Reach 68, Total 692
## Node 23, Reach 68, Total 760
## Node 24, Reach 68, Total 828
## Node 25, Reach 68, Total 896
## Node 26, Reach 68, Total 964
## Node 27, Reach 68, Total 1032
## Node 28, Reach 68, Total 1100
## Node 29, Reach 68, Total 1168
## Node 30, Reach 68, Total 1236
## Node 31, Reach 68, Total 1304
## Node 32, Reach 68, Total 1372
## Node 33, Reach 68, Total 1440
## Node 34, Reach 68, Total 1508
## Node 35, Reach 68, Total 1576
## Node 36, Reach 68, Total 1644
## Node 37, Reach 68, Total 1712
## Node 38, Reach 68, Total 1780
## Node 39, Reach 68, Total 1848
## Node 40, Reach 68, Total 1916
## Node 41, Reach 68, Total 1984
## Node 42, Reach 68, Total 2052
## Node 43, Reach 68, Total 2120
## Node 44, Reach 68, Total 2188
## Node 45, Reach 68, Total 2256
```

```
## Node 46, Reach 68, Total 2324
## Node 47, Reach 68, Total 2392
## Node 48, Reach 68, Total 2460
## Node 49, Reach 68, Total 2528
## Node 50, Reach 68, Total 2596
## Node 51, Reach 68, Total 2664
## Node 52, Reach 68, Total 2732
## Node 53, Reach 68, Total 2800
## Node 54, Reach 68, Total 2868
## Node 55, Reach 68, Total 2936
## Node 56, Reach 68, Total 3004
## Node 57, Reach 68, Total 3072
## Node 58, Reach 68, Total 3140
## Node 59, Reach 68, Total 3208
## Node 60, Reach 5, Total 3213
## Node 61, Reach 68, Total 3281
## Node 62, Reach 3, Total 3284
## Node 63, Reach 3, Total 3287
## Node 64, Reach 68, Total 3355
## Node 65, Reach 68, Total 3423
## Node 66, Reach 68, Total 3491
## Node 67, Reach 68, Total 3559
## Node 68, Reach 68, Total 3627
## Node 1, Reach 1, Total 1
## Node 2, Reach 1, Total 2
## Node 3, Reach 1, Total 3
## Node 4, Reach 1, Total 4
## Node 5, Reach 1, Total 5
## Node 6, Reach 94, Total 99
## Node 7, Reach 94, Total 193
## Node 8, Reach 94, Total 287
## Node 9, Reach 94, Total 381
## Node 10, Reach 94, Total 475
## Node 11, Reach 94, Total 569
## Node 12, Reach 94, Total 663
## Node 13, Reach 94, Total 757
## Node 14, Reach 94, Total 851
## Node 15, Reach 94, Total 945
## Node 16, Reach 94, Total 1039
## Node 17, Reach 94, Total 1133
## Node 18, Reach 94, Total 1227
## Node 19, Reach 94, Total 1321
## Node 20, Reach 94, Total 1415
## Node 21, Reach 94, Total 1509
## Node 22, Reach 94, Total 1603
## Node 23, Reach 94, Total 1697
## Node 24, Reach 94, Total 1791
## Node 25, Reach 94, Total 1885
## Node 26, Reach 94, Total 1979
## Node 27, Reach 94, Total 2073
## Node 28, Reach 94, Total 2167
## Node 29, Reach 94, Total 2261
## Node 30, Reach 94, Total 2355
## Node 31, Reach 94, Total 2449
```

```
## Node 32, Reach 94, Total 2543
## Node 33, Reach 94, Total 2637
## Node 34, Reach 94, Total 2731
## Node 35, Reach 94, Total 2825
## Node 36, Reach 94, Total 2919
## Node 37, Reach 94, Total 3013
## Node 38, Reach 94, Total 3107
## Node 39, Reach 94, Total 3201
## Node 40, Reach 94, Total 3295
## Node 41, Reach 94, Total 3389
## Node 42, Reach 94, Total 3483
## Node 43, Reach 94, Total 3577
## Node 44, Reach 94, Total 3671
## Node 45, Reach 94, Total 3765
## Node 46, Reach 94, Total 3859
## Node 47, Reach 94, Total 3953
## Node 48, Reach 94, Total 4047
## Node 49, Reach 94, Total 4141
## Node 50, Reach 94, Total 4235
## Node 51, Reach 94, Total 4329
## Node 52, Reach 94, Total 4423
## Node 53, Reach 94, Total 4517
## Node 54, Reach 94, Total 4611
## Node 55, Reach 94, Total 4705
## Node 56, Reach 94, Total 4799
## Node 57, Reach 94, Total 4893
## Node 58, Reach 94, Total 4987
## Node 59, Reach 94, Total 5081
## Node 60, Reach 94, Total 5175
## Node 61, Reach 94, Total 5269
## Node 62, Reach 94, Total 5363
## Node 63, Reach 94, Total 5457
## Node 64, Reach 94, Total 5551
## Node 65, Reach 94, Total 5645
## Node 66, Reach 94, Total 5739
## Node 67, Reach 94, Total 5833
## Node 68, Reach 94, Total 5927
## Node 69, Reach 94, Total 6021
## Node 70, Reach 94, Total 6115
## Node 71, Reach 94, Total 6209
## Node 72, Reach 94, Total 6303
## Node 73, Reach 94, Total 6397
## Node 74, Reach 94, Total 6491
## Node 75, Reach 94, Total 6585
## Node 76, Reach 94, Total 6679
## Node 77, Reach 94, Total 6773
## Node 78, Reach 94, Total 6867
## Node 79, Reach 94, Total 6961
## Node 80, Reach 94, Total 7055
## Node 81, Reach 2, Total 7057
## Node 82, Reach 3, Total 7060
## Node 83, Reach 94, Total 7154
## Node 84, Reach 94, Total 7248
## Node 85, Reach 94, Total 7342
```

```
## Node 86, Reach 94, Total 7436
## Node 87, Reach 94, Total 7530
## Node 88, Reach 94, Total 7624
## Node 89, Reach 3, Total 7627
## Node 90, Reach 94, Total 7721
## Node 91, Reach 94, Total 7815
## Node 92, Reach 94, Total 7909
## Node 93, Reach 94, Total 8003
## Node 94, Reach 94, Total 8097
## Node 1, Reach 1, Total 1
## Node 2, Reach 1, Total 2
## Node 3, Reach 1, Total 3
## Node 4, Reach 1, Total 4
## Node 5, Reach 1, Total 5
## Node 6, Reach 94, Total 99
## Node 7, Reach 94, Total 193
## Node 8, Reach 94, Total 287
## Node 9, Reach 94, Total 381
## Node 10, Reach 94, Total 475
## Node 11, Reach 94, Total 569
## Node 12, Reach 94, Total 663
## Node 13, Reach 94, Total 757
## Node 14, Reach 94, Total 851
## Node 15, Reach 94, Total 945
## Node 16, Reach 94, Total 1039
## Node 17, Reach 94, Total 1133
## Node 18, Reach 94, Total 1227
## Node 19, Reach 94, Total 1321
## Node 20, Reach 94, Total 1415
## Node 21, Reach 94, Total 1509
## Node 22, Reach 94, Total 1603
## Node 23, Reach 94, Total 1697
## Node 24, Reach 94, Total 1791
## Node 25, Reach 94, Total 1885
## Node 26, Reach 94, Total 1979
## Node 27, Reach 94, Total 2073
## Node 28, Reach 94, Total 2167
## Node 29, Reach 94, Total 2261
## Node 30, Reach 94, Total 2355
## Node 31, Reach 94, Total 2449
## Node 32, Reach 94, Total 2543
## Node 33, Reach 94, Total 2637
## Node 34, Reach 94, Total 2731
## Node 35, Reach 94, Total 2825
## Node 36, Reach 94, Total 2919
## Node 37, Reach 94, Total 3013
## Node 38, Reach 94, Total 3107
## Node 39, Reach 94, Total 3201
## Node 40, Reach 94, Total 3295
## Node 41, Reach 94, Total 3389
## Node 42, Reach 94, Total 3483
## Node 43, Reach 94, Total 3577
## Node 44, Reach 94, Total 3671
## Node 45, Reach 94, Total 3765
```

```
## Node 46, Reach 94, Total 3859
## Node 47, Reach 94, Total 3953
## Node 48, Reach 94, Total 4047
## Node 49, Reach 94, Total 4141
## Node 50, Reach 94, Total 4235
## Node 51, Reach 94, Total 4329
## Node 52, Reach 94, Total 4423
## Node 53, Reach 94, Total 4517
## Node 54, Reach 94, Total 4611
## Node 55, Reach 94, Total 4705
## Node 56, Reach 94, Total 4799
## Node 57, Reach 94, Total 4893
## Node 58, Reach 94, Total 4987
## Node 59, Reach 94, Total 5081
## Node 60, Reach 94, Total 5175
## Node 61, Reach 94, Total 5269
## Node 62, Reach 94, Total 5363
## Node 63, Reach 94, Total 5457
## Node 64, Reach 94, Total 5551
## Node 65, Reach 94, Total 5645
## Node 66, Reach 94, Total 5739
## Node 67, Reach 94, Total 5833
## Node 68, Reach 94, Total 5927
## Node 69, Reach 94, Total 6021
## Node 70, Reach 94, Total 6115
## Node 71, Reach 94, Total 6209
## Node 72, Reach 94, Total 6303
## Node 73, Reach 94, Total 6397
## Node 74, Reach 94, Total 6491
## Node 75, Reach 94, Total 6585
## Node 76, Reach 94, Total 6679
## Node 77, Reach 94, Total 6773
## Node 78, Reach 94, Total 6867
## Node 79, Reach 94, Total 6961
## Node 80, Reach 94, Total 7055
## Node 81, Reach 2, Total 7057
## Node 82, Reach 3, Total 7060
## Node 83, Reach 94, Total 7154
## Node 84, Reach 94, Total 7248
## Node 85, Reach 94, Total 7342
## Node 86, Reach 94, Total 7436
## Node 87, Reach 94, Total 7530
## Node 88, Reach 94, Total 7624
## Node 89, Reach 3, Total 7627
## Node 90, Reach 94, Total 7721
## Node 91, Reach 94, Total 7815
## Node 92, Reach 94, Total 7909
## Node 93, Reach 94, Total 8003
## Node 94, Reach 94, Total 8097
## Node 1, Reach 1, Total 1
## Node 2, Reach 1, Total 2
## Node 3, Reach 1, Total 3
## Node 4, Reach 1, Total 4
## Node 5, Reach 1, Total 5
```

```
## Node 6, Reach 1, Total 6
## Node 7, Reach 1, Total 7
## Node 8, Reach 1, Total 8
## Node 9, Reach 1, Total 9
## Node 10, Reach 1, Total 10
## Node 11, Reach 1, Total 11
## Node 12, Reach 1, Total 12
## Node 13, Reach 1, Total 13
## Node 14, Reach 1, Total 14
## Node 15, Reach 14, Total 28
## Node 16, Reach 125, Total 153
## Node 17, Reach 125, Total 278
## Node 18, Reach 125, Total 403
## Node 19, Reach 125, Total 528
## Node 20, Reach 125, Total 653
## Node 21, Reach 125, Total 778
## Node 22, Reach 125, Total 903
## Node 23, Reach 125, Total 1028
## Node 24, Reach 125, Total 1153
## Node 25, Reach 125, Total 1278
## Node 26, Reach 125, Total 1403
## Node 27, Reach 125, Total 1528
## Node 28, Reach 125, Total 1653
## Node 29, Reach 125, Total 1778
## Node 30, Reach 125, Total 1903
## Node 31, Reach 125, Total 2028
## Node 32, Reach 125, Total 2153
## Node 33, Reach 2, Total 2155
## Node 34, Reach 125, Total 2280
## Node 35, Reach 125, Total 2405
## Node 36, Reach 125, Total 2530
## Node 37, Reach 125, Total 2655
## Node 38, Reach 125, Total 2780
## Node 39, Reach 125, Total 2905
## Node 40, Reach 125, Total 3030
## Node 41, Reach 125, Total 3155
## Node 42, Reach 5, Total 3160
## Node 43, Reach 3, Total 3163
## Node 44, Reach 125, Total 3288
## Node 45, Reach 125, Total 3413
## Node 46, Reach 2, Total 3415
## Node 47, Reach 125, Total 3540
## Node 48, Reach 125, Total 3665
## Node 49, Reach 125, Total 3790
## Node 50, Reach 125, Total 3915
## Node 51, Reach 125, Total 4040
## Node 52, Reach 125, Total 4165
## Node 53, Reach 125, Total 4290
## Node 54, Reach 125, Total 4415
## Node 55, Reach 125, Total 4540
## Node 56, Reach 125, Total 4665
## Node 57, Reach 125, Total 4790
## Node 58, Reach 125, Total 4915
## Node 59, Reach 125, Total 5040
```

```
## Node 60, Reach 125, Total 5165
## Node 61, Reach 125, Total 5290
## Node 62, Reach 125, Total 5415
## Node 63, Reach 125, Total 5540
## Node 64, Reach 125, Total 5665
## Node 65, Reach 125, Total 5790
## Node 66, Reach 125, Total 5915
## Node 67, Reach 125, Total 6040
## Node 68, Reach 125, Total 6165
## Node 69, Reach 125, Total 6290
## Node 70, Reach 125, Total 6415
## Node 71, Reach 10, Total 6425
## Node 72, Reach 125, Total 6550
## Node 73, Reach 125, Total 6675
## Node 74, Reach 125, Total 6800
## Node 75, Reach 125, Total 6925
## Node 76, Reach 125, Total 7050
## Node 77, Reach 125, Total 7175
## Node 78, Reach 125, Total 7300
## Node 79, Reach 125, Total 7425
## Node 80, Reach 125, Total 7550
## Node 81, Reach 125, Total 7675
## Node 82, Reach 125, Total 7800
## Node 83, Reach 125, Total 7925
## Node 84, Reach 125, Total 8050
## Node 85, Reach 125, Total 8175
## Node 86, Reach 125, Total 8300
## Node 87, Reach 125, Total 8425
## Node 88, Reach 125, Total 8550
## Node 89, Reach 125, Total 8675
## Node 90, Reach 125, Total 8800
## Node 91, Reach 125, Total 8925
## Node 92, Reach 125, Total 9050
## Node 93, Reach 125, Total 9175
## Node 94, Reach 125, Total 9300
## Node 95, Reach 125, Total 9425
## Node 96, Reach 125, Total 9550
## Node 97, Reach 125, Total 9675
## Node 98, Reach 125, Total 9800
## Node 99, Reach 125, Total 9925
## Node 100, Reach 125, Total 10050
## Node 101, Reach 125, Total 10175
## Node 102, Reach 125, Total 10300
## Node 103, Reach 125, Total 10425
## Node 104, Reach 125, Total 10550
## Node 105, Reach 125, Total 10675
## Node 106, Reach 125, Total 10800
## Node 107, Reach 125, Total 10925
## Node 108, Reach 125, Total 11050
## Node 109, Reach 125, Total 11175
## Node 110, Reach 125, Total 11300
## Node 111, Reach 125, Total 11425
## Node 112, Reach 125, Total 11550
## Node 113, Reach 125, Total 11675
```

```
## Node 114, Reach 125, Total 11800
## Node 115, Reach 125, Total 11925
## Node 116, Reach 125, Total 12050
## Node 117, Reach 125, Total 12175
## Node 118, Reach 125, Total 12300
## Node 119, Reach 3, Total 12303
## Node 120, Reach 125, Total 12428
## Node 121, Reach 125, Total 12553
## Node 122, Reach 125, Total 12678
## Node 123, Reach 125, Total 12803
## Node 124, Reach 125, Total 12928
## Node 125, Reach 13, Total 12941
## Node 1, Reach 1, Total 1
## Node 2, Reach 1, Total 2
## Node 3, Reach 1, Total 3
## Node 4, Reach 1, Total 4
## Node 5, Reach 1, Total 5
## Node 6, Reach 1, Total 6
## Node 7, Reach 1, Total 7
## Node 8, Reach 1, Total 8
## Node 9, Reach 1, Total 9
## Node 10, Reach 1, Total 10
## Node 11, Reach 1, Total 11
## Node 12, Reach 1, Total 12
## Node 13, Reach 1, Total 13
## Node 14, Reach 1, Total 14
## Node 15, Reach 14, Total 28
## Node 16, Reach 125, Total 153
## Node 17, Reach 125, Total 278
## Node 18, Reach 125, Total 403
## Node 19, Reach 125, Total 528
## Node 20, Reach 125, Total 653
## Node 21, Reach 125, Total 778
## Node 22, Reach 125, Total 903
## Node 23, Reach 125, Total 1028
## Node 24, Reach 125, Total 1153
## Node 25, Reach 125, Total 1278
## Node 26, Reach 125, Total 1403
## Node 27, Reach 125, Total 1528
## Node 28, Reach 125, Total 1653
## Node 29, Reach 125, Total 1778
## Node 30, Reach 125, Total 1903
## Node 31, Reach 125, Total 2028
## Node 32, Reach 125, Total 2153
## Node 33, Reach 2, Total 2155
## Node 34, Reach 125, Total 2280
## Node 35, Reach 125, Total 2405
## Node 36, Reach 125, Total 2530
## Node 37, Reach 125, Total 2655
## Node 38, Reach 125, Total 2780
## Node 39, Reach 125, Total 2905
## Node 40, Reach 125, Total 3030
## Node 41, Reach 125, Total 3155
## Node 42, Reach 5, Total 3160
```

```
## Node 43, Reach 3, Total 3163
## Node 44, Reach 125, Total 3288
## Node 45, Reach 125, Total 3413
## Node 46, Reach 2, Total 3415
## Node 47, Reach 125, Total 3540
## Node 48, Reach 125, Total 3665
## Node 49, Reach 125, Total 3790
## Node 50, Reach 125, Total 3915
## Node 51, Reach 125, Total 4040
## Node 52, Reach 125, Total 4165
## Node 53, Reach 125, Total 4290
## Node 54, Reach 125, Total 4415
## Node 55, Reach 125, Total 4540
## Node 56, Reach 125, Total 4665
## Node 57, Reach 125, Total 4790
## Node 58, Reach 125, Total 4915
## Node 59, Reach 125, Total 5040
## Node 60, Reach 125, Total 5165
## Node 61, Reach 125, Total 5290
## Node 62, Reach 125, Total 5415
## Node 63, Reach 125, Total 5540
## Node 64, Reach 125, Total 5665
## Node 65, Reach 125, Total 5790
## Node 66, Reach 125, Total 5915
## Node 67, Reach 125, Total 6040
## Node 68, Reach 125, Total 6165
## Node 69, Reach 125, Total 6290
## Node 70, Reach 125, Total 6415
## Node 71, Reach 10, Total 6425
## Node 72, Reach 125, Total 6550
## Node 73, Reach 125, Total 6675
## Node 74, Reach 125, Total 6800
## Node 75, Reach 125, Total 6925
## Node 76, Reach 125, Total 7050
## Node 77, Reach 125, Total 7175
## Node 78, Reach 125, Total 7300
## Node 79, Reach 125, Total 7425
## Node 80, Reach 125, Total 7550
## Node 81, Reach 125, Total 7675
## Node 82, Reach 125, Total 7800
## Node 83, Reach 125, Total 7925
## Node 84, Reach 125, Total 8050
## Node 85, Reach 125, Total 8175
## Node 86, Reach 125, Total 8300
## Node 87, Reach 125, Total 8425
## Node 88, Reach 125, Total 8550
## Node 89, Reach 125, Total 8675
## Node 90, Reach 125, Total 8800
## Node 91, Reach 125, Total 8925
## Node 92, Reach 125, Total 9050
## Node 93, Reach 125, Total 9175
## Node 94, Reach 125, Total 9300
## Node 95, Reach 125, Total 9425
## Node 96, Reach 125, Total 9550
```

```
## Node 97, Reach 125, Total 9675
## Node 98, Reach 125, Total 9800
## Node 99, Reach 125, Total 9925
## Node 100, Reach 125, Total 10050
## Node 101, Reach 125, Total 10175
## Node 102, Reach 125, Total 10300
## Node 103, Reach 125, Total 10425
## Node 104, Reach 125, Total 10550
## Node 105, Reach 125, Total 10675
## Node 106, Reach 125, Total 10800
## Node 107, Reach 125, Total 10925
## Node 108, Reach 125, Total 11050
## Node 109, Reach 125, Total 11175
## Node 110, Reach 125, Total 11300
## Node 111, Reach 125, Total 11425
## Node 112, Reach 125, Total 11550
## Node 113, Reach 125, Total 11675
## Node 114, Reach 125, Total 11800
## Node 115, Reach 125, Total 11925
## Node 116, Reach 125, Total 12050
## Node 117, Reach 125, Total 12175
## Node 118, Reach 125, Total 12300
## Node 119, Reach 3, Total 12303
## Node 120, Reach 125, Total 12428
## Node 121, Reach 125, Total 12553
## Node 122, Reach 125, Total 12678
## Node 123, Reach 125, Total 12803
## Node 124, Reach 125, Total 12928
## Node 125, Reach 13, Total 12941
## Node 1, Reach 4, Total 4
## Node 2, Reach 4, Total 8
## Node 3, Reach 4, Total 12
## Node 4, Reach 1, Total 13

## Warning in enaAscendency(x): Respiration data is absent from the model.

## Node 1, Reach 4, Total 4
## Node 2, Reach 4, Total 8
## Node 3, Reach 4, Total 12
## Node 4, Reach 4, Total 16

## Warning in enaAscendency(x): Respiration data is absent from the model.

## Node 1, Reach 4, Total 4
## Node 2, Reach 4, Total 8
## Node 3, Reach 4, Total 12
## Node 4, Reach 4, Total 16

## Warning in enaAscendency(x): Respiration data is absent from the model.

## Node 1, Reach 4, Total 4
## Node 2, Reach 4, Total 8
## Node 3, Reach 4, Total 12
## Node 4, Reach 4, Total 16
```

```
## Warning in enaAscendency(x): Respiration data is absent from the model.

## Node 1, Reach 4, Total 4
## Node 2, Reach 4, Total 8
## Node 3, Reach 4, Total 12
## Node 4, Reach 4, Total 16

## Warning in enaAscendency(x): Respiration data is absent from the model.

## Node 1, Reach 4, Total 4
## Node 2, Reach 4, Total 8
## Node 3, Reach 4, Total 12
## Node 4, Reach 4, Total 16

## Warning in enaAscendency(x): Respiration data is absent from the model.

## Node 1, Reach 4, Total 4
## Node 2, Reach 4, Total 8
## Node 3, Reach 4, Total 12
## Node 4, Reach 4, Total 16

## Warning in enaAscendency(x): Respiration data is absent from the model.

## Node 1, Reach 4, Total 4
## Node 2, Reach 4, Total 8
## Node 3, Reach 4, Total 12
## Node 4, Reach 4, Total 16

## Warning in enaAscendency(x): Respiration data is absent from the model.

## Node 1, Reach 4, Total 4
## Node 2, Reach 4, Total 8
## Node 3, Reach 4, Total 12
## Node 4, Reach 4, Total 16

## Warning in enaAscendency(x): Respiration data is absent from the model.

## Node 1, Reach 4, Total 4
## Node 2, Reach 4, Total 8
## Node 3, Reach 4, Total 12
## Node 4, Reach 4, Total 16

## Warning in enaAscendency(x): Respiration data is absent from the model.

## Node 1, Reach 4, Total 4
## Node 2, Reach 4, Total 8
## Node 3, Reach 4, Total 12
## Node 4, Reach 4, Total 16

## Warning in enaAscendency(x): Respiration data is absent from the model.
```

```
## Node 1, Reach 4, Total 4
## Node 2, Reach 4, Total 8
## Node 3, Reach 4, Total 12
## Node 4, Reach 4, Total 16

## Warning in enaAscendency(x): Respiration data is absent from the model.

## Node 1, Reach 5, Total 5
## Node 2, Reach 5, Total 10
## Node 3, Reach 5, Total 15
## Node 4, Reach 5, Total 20
## Node 5, Reach 5, Total 25

## Warning in enaAscendency(x): Export data is absent from the model.

## Node 1, Reach 7, Total 7
## Node 2, Reach 7, Total 14
## Node 3, Reach 7, Total 21
## Node 4, Reach 7, Total 28
## Node 5, Reach 7, Total 35
## Node 6, Reach 7, Total 42
## Node 7, Reach 7, Total 49
## Node 1, Reach 7, Total 7
## Node 2, Reach 7, Total 14
## Node 3, Reach 7, Total 21
## Node 4, Reach 7, Total 28
## Node 5, Reach 7, Total 35
## Node 6, Reach 7, Total 42
## Node 7, Reach 7, Total 49
## Node 1, Reach 7, Total 7
## Node 2, Reach 7, Total 14
## Node 3, Reach 7, Total 21
## Node 4, Reach 7, Total 28
## Node 5, Reach 7, Total 35
## Node 6, Reach 7, Total 42
## Node 7, Reach 7, Total 49
## Node 1, Reach 7, Total 7
## Node 2, Reach 7, Total 14
## Node 3, Reach 7, Total 21
## Node 4, Reach 7, Total 28
## Node 5, Reach 7, Total 35
## Node 6, Reach 7, Total 42
## Node 7, Reach 7, Total 49
## Node 1, Reach 7, Total 7
## Node 2, Reach 7, Total 14
## Node 3, Reach 7, Total 21
## Node 4, Reach 7, Total 28
## Node 5, Reach 7, Total 35
## Node 6, Reach 7, Total 42
## Node 7, Reach 7, Total 49
## Node 1, Reach 7, Total 7
## Node 2, Reach 7, Total 14
## Node 3, Reach 7, Total 21
```

```
## Node 4, Reach 7, Total 28
## Node 5, Reach 7, Total 35
## Node 6, Reach 7, Total 42
## Node 7, Reach 7, Total 49
## Node 1, Reach 7, Total 7
## Node 2, Reach 7, Total 14
## Node 3, Reach 7, Total 21
## Node 4, Reach 7, Total 28
## Node 5, Reach 7, Total 35
## Node 6, Reach 7, Total 42
## Node 7, Reach 7, Total 49
## Node 1, Reach 7, Total 7
## Node 2, Reach 7, Total 14
## Node 3, Reach 7, Total 21
## Node 4, Reach 7, Total 28
## Node 5, Reach 7, Total 35
## Node 6, Reach 7, Total 42
## Node 7, Reach 7, Total 49
## Node 1, Reach 7, Total 7
## Node 2, Reach 7, Total 14
## Node 3, Reach 7, Total 21
## Node 4, Reach 7, Total 28
## Node 5, Reach 7, Total 35
## Node 6, Reach 7, Total 42
## Node 7, Reach 7, Total 49
## Node 1, Reach 7, Total 7
## Node 2, Reach 7, Total 14
## Node 3, Reach 7, Total 21
## Node 4, Reach 7, Total 28
## Node 5, Reach 7, Total 35
## Node 6, Reach 7, Total 42
## Node 7, Reach 7, Total 49
## Node 1, Reach 7, Total 7
## Node 2, Reach 7, Total 14
## Node 3, Reach 7, Total 21
## Node 4, Reach 7, Total 28
## Node 5, Reach 7, Total 35
## Node 6, Reach 7, Total 42
## Node 7, Reach 7, Total 49
## Node 1, Reach 7, Total 7
## Node 2, Reach 7, Total 14
## Node 3, Reach 7, Total 21
## Node 4, Reach 7, Total 28
## Node 5, Reach 7, Total 35
## Node 6, Reach 7, Total 42
## Node 7, Reach 7, Total 49
## Node 1, Reach 7, Total 7
## Node 2, Reach 7, Total 14
## Node 3, Reach 7, Total 21
## Node 4, Reach 7, Total 28
## Node 5, Reach 7, Total 35
## Node 6, Reach 7, Total 42
## Node 7, Reach 7, Total 49
## Node 1, Reach 7, Total 7
```

```
## Node 2, Reach 7, Total 14
## Node 3, Reach 7, Total 21
## Node 4, Reach 7, Total 28
## Node 5, Reach 7, Total 35
## Node 6, Reach 7, Total 42
## Node 7, Reach 7, Total 49
## Node 1, Reach 7, Total 7
## Node 2, Reach 7, Total 14
## Node 3, Reach 7, Total 21
## Node 4, Reach 7, Total 28
## Node 5, Reach 7, Total 35
## Node 6, Reach 7, Total 42
## Node 7, Reach 7, Total 49
## Node 1, Reach 7, Total 7
## Node 2, Reach 7, Total 14
## Node 3, Reach 7, Total 21
## Node 4, Reach 7, Total 28
## Node 5, Reach 7, Total 35
## Node 6, Reach 7, Total 42
## Node 7, Reach 7, Total 49
## Node 1, Reach 7, Total 7
## Node 2, Reach 7, Total 14
## Node 3, Reach 7, Total 21
## Node 4, Reach 7, Total 28
## Node 5, Reach 7, Total 35
## Node 6, Reach 7, Total 42
## Node 7, Reach 7, Total 49
## Node 1, Reach 8, Total 8
## Node 2, Reach 8, Total 16
## Node 3, Reach 8, Total 24
## Node 4, Reach 8, Total 32
## Node 5, Reach 8, Total 40
## Node 6, Reach 8, Total 48
## Node 7, Reach 8, Total 56
## Node 8, Reach 8, Total 64
```

```
## Warning in enaAscendency(x): Respiration data is absent from the model.
```

```
## Node 1, Reach 8, Total 8
## Node 2, Reach 8, Total 16
## Node 3, Reach 8, Total 24
## Node 4, Reach 8, Total 32
## Node 5, Reach 8, Total 40
## Node 6, Reach 8, Total 48
## Node 7, Reach 8, Total 56
## Node 8, Reach 8, Total 64
```

```
## Warning in enaAscendency(x): Respiration data is absent from the model.
```

```
## Node 1, Reach 11, Total 11
## Node 2, Reach 11, Total 22
## Node 3, Reach 11, Total 33
## Node 4, Reach 11, Total 44
```

```
## Node 5, Reach 11, Total 55
## Node 6, Reach 11, Total 66
## Node 7, Reach 11, Total 77
## Node 8, Reach 11, Total 88
## Node 9, Reach 11, Total 99
## Node 10, Reach 11, Total 110
## Node 11, Reach 11, Total 121

## Warning in enaAscendency(x): Respiration data is absent from the model.

## Node 1, Reach 16, Total 16
## Node 2, Reach 16, Total 32
## Node 3, Reach 16, Total 48
## Node 4, Reach 16, Total 64
## Node 5, Reach 16, Total 80
## Node 6, Reach 16, Total 96
## Node 7, Reach 16, Total 112
## Node 8, Reach 16, Total 128
## Node 9, Reach 16, Total 144
## Node 10, Reach 16, Total 160
## Node 11, Reach 16, Total 176
## Node 12, Reach 16, Total 192
## Node 13, Reach 16, Total 208
## Node 14, Reach 16, Total 224
## Node 15, Reach 16, Total 240
## Node 16, Reach 16, Total 256

## Warning in enaAscendency(x): Respiration data is absent from the model.

## Node 1, Reach 36, Total 36
## Node 2, Reach 36, Total 72
## Node 3, Reach 36, Total 108
## Node 4, Reach 36, Total 144
## Node 5, Reach 36, Total 180
## Node 6, Reach 36, Total 216
## Node 7, Reach 36, Total 252
## Node 8, Reach 36, Total 288
## Node 9, Reach 36, Total 324
## Node 10, Reach 36, Total 360
## Node 11, Reach 36, Total 396
## Node 12, Reach 36, Total 432
## Node 13, Reach 36, Total 468
## Node 14, Reach 36, Total 504
## Node 15, Reach 36, Total 540
## Node 16, Reach 36, Total 576
## Node 17, Reach 36, Total 612
## Node 18, Reach 36, Total 648
## Node 19, Reach 36, Total 684
## Node 20, Reach 36, Total 720
## Node 21, Reach 36, Total 756
## Node 22, Reach 36, Total 792
## Node 23, Reach 36, Total 828
## Node 24, Reach 36, Total 864
```

```
## Node 25, Reach 36, Total 900
## Node 26, Reach 36, Total 936
## Node 27, Reach 36, Total 972
## Node 28, Reach 36, Total 1008
## Node 29, Reach 36, Total 1044
## Node 30, Reach 36, Total 1080
## Node 31, Reach 36, Total 1116
## Node 32, Reach 36, Total 1152
## Node 33, Reach 36, Total 1188
## Node 34, Reach 36, Total 1224
## Node 35, Reach 36, Total 1260
## Node 36, Reach 36, Total 1296
## Node 1, Reach 59, Total 59
## Node 2, Reach 59, Total 118
## Node 3, Reach 59, Total 177
## Node 4, Reach 59, Total 236
## Node 5, Reach 59, Total 295
## Node 6, Reach 59, Total 354
## Node 7, Reach 59, Total 413
## Node 8, Reach 59, Total 472
## Node 9, Reach 59, Total 531
## Node 10, Reach 59, Total 590
## Node 11, Reach 59, Total 649
## Node 12, Reach 59, Total 708
## Node 13, Reach 59, Total 767
## Node 14, Reach 59, Total 826
## Node 15, Reach 59, Total 885
## Node 16, Reach 59, Total 944
## Node 17, Reach 59, Total 1003
## Node 18, Reach 59, Total 1062
## Node 19, Reach 59, Total 1121
## Node 20, Reach 59, Total 1180
## Node 21, Reach 59, Total 1239
## Node 22, Reach 59, Total 1298
## Node 23, Reach 59, Total 1357
## Node 24, Reach 59, Total 1416
## Node 25, Reach 59, Total 1475
## Node 26, Reach 59, Total 1534
## Node 27, Reach 59, Total 1593
## Node 28, Reach 59, Total 1652
## Node 29, Reach 59, Total 1711
## Node 30, Reach 59, Total 1770
## Node 31, Reach 59, Total 1829
## Node 32, Reach 59, Total 1888
## Node 33, Reach 59, Total 1947
## Node 34, Reach 59, Total 2006
## Node 35, Reach 59, Total 2065
## Node 36, Reach 59, Total 2124
## Node 37, Reach 59, Total 2183
## Node 38, Reach 59, Total 2242
## Node 39, Reach 59, Total 2301
## Node 40, Reach 59, Total 2360
## Node 41, Reach 59, Total 2419
## Node 42, Reach 59, Total 2478
```

```
## Node 43, Reach 59, Total 2537
## Node 44, Reach 59, Total 2596
## Node 45, Reach 59, Total 2655
## Node 46, Reach 59, Total 2714
## Node 47, Reach 59, Total 2773
## Node 48, Reach 59, Total 2832
## Node 49, Reach 59, Total 2891
## Node 50, Reach 59, Total 2950
## Node 51, Reach 59, Total 3009
## Node 52, Reach 59, Total 3068
## Node 53, Reach 59, Total 3127
## Node 54, Reach 59, Total 3186
## Node 55, Reach 59, Total 3245
## Node 56, Reach 59, Total 3304
## Node 57, Reach 59, Total 3363
## Node 58, Reach 59, Total 3422
## Node 59, Reach 59, Total 3481
```

```
ns <- do.call(rbind,ns.list)  # ns as a data.frame
## Let's take a quick look at some of the output
colnames(ns)    # return network statistic names.
```

```
##  [1] "n"                 "L"                 "C"
##  [4] "LD"                "ppr"               "lam1A"
##  [7] "mlam1A"            "rho"               "R"
## [10] "d"                 "no.scc"            "no.scc.big"
## [13] "pscc"              "Boundary"          "TST"
## [16] "TSTp"              "APL"               "FCI"
## [19] "BFI"               "DFI"               "IFI"
## [22] "ID.F"              "ID.F.I"            "ID.F.O"
## [25] "HMG.I"             "HMG.O"             "AMP.I"
## [28] "AMP.O"             "mode0.F"           "mode1.F"
## [31] "mode2.F"           "mode3.F"           "mode4.F"
## [34] "H"                 "AMI"               "Hr"
## [37] "CAP"               "ASC"               "OH"
## [40] "ASC.CAP"           "OH.CAP"            "robustness"
## [43] "ELD"               "TD"                "A.input"
## [46] "A.internal"        "A.export"          "A.respiration"
## [49] "OH.input"          "OH.internal"       "OH.export"
## [52] "OH.respiration"    "CAP.input"         "CAP.internal"
## [55] "CAP.export"        "CAP.respiration"   "TSS"
## [58] "CIS"               "BSI"               "DSI"
## [61] "ISI"               "ID.S"              "ID.S.I"
## [64] "ID.S.O"            "HMG.S.O"           "HMG.S.I"
## [67] "NAS"               "NASP"              "mode0.S"
## [70] "mode1.S"           "mode2.S"           "mode3.S"
## [73] "mode4.S"           "lam1D"             "relation.change.F"
## [76] "synergism.F"       "mutualism.F"       "lam1DS"
## [79] "relation.change.S" "synergism.S"       "mutualism.S"
```

```
dim(ns)         # show dimensions of ns matrix
```

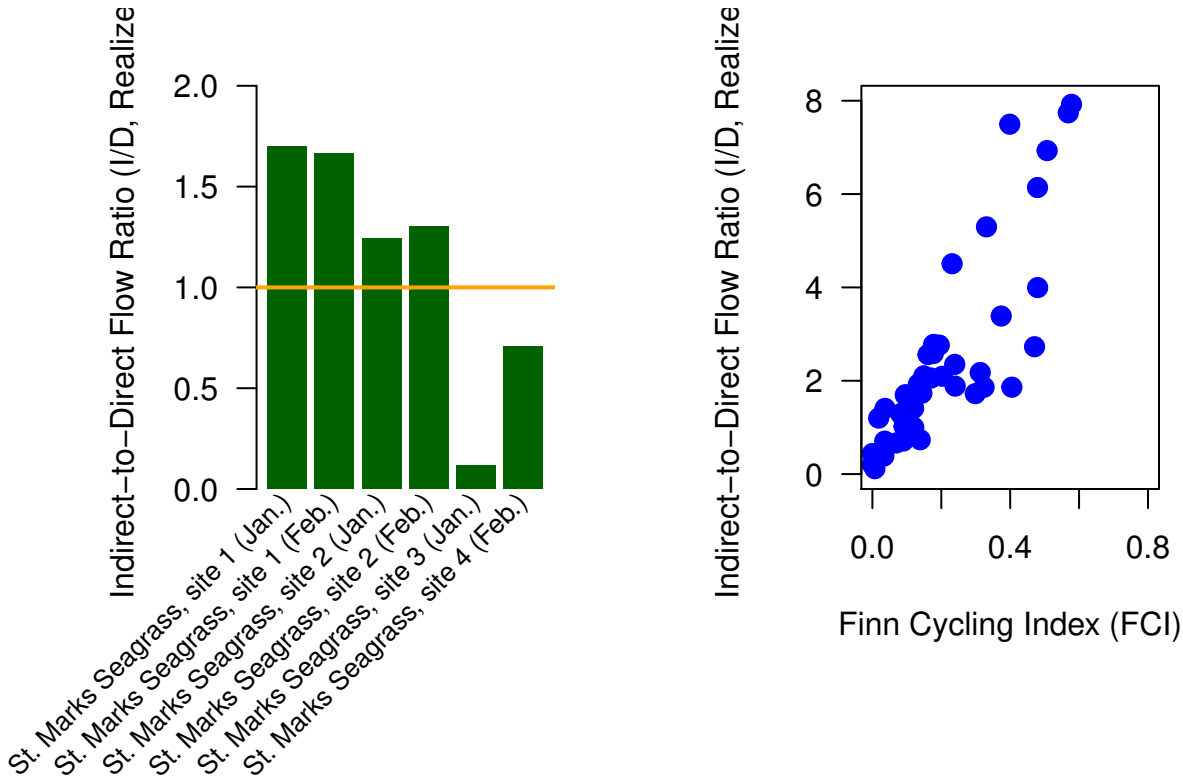
```
## [1] 74 81
```

```
ns[1:5,1:5]      # show selected results
```

	n	L	C	LD	ppr
Marine Coprophagy (oyster)	4	4	0.250	1.0	1.000000
Lake Findley	4	6	0.375	1.5	1.004975
Mirror Lake	5	9	0.360	1.8	1.324718
Lake Wingra	5	10	0.400	2.0	2.000000
Marion Lake	5	9	0.360	1.8	1.324718

Given this data frame of network statistics, we can construct interesting plots for further analysis. Here we focus on results of the St. Marks Seagrass ecosystem (Baird, Luczkovich, and Christian 1998).

```
opar <- par(las=1,mar=c(9,7,2,1),xpd=TRUE,mfrow=c(1,2),oma=c(1,1,0,0))
## Number of models
x=dim(ns)[1]
m.select <- 26:31
bp=barplot(ns$ID.F[m.select],ylab="Indirect-to-Direct Flow Ratio (I/D, Realized)",
        col="darkgreen",border=NA,ylim=c(0,2))
## Add labels
text(bp,-0.05,
     labels=rownames(ns)[m.select],
       srt=45,adj=1,cex=0.85)
opar <- par(xpd=FALSE)
abline(h=1,col="orange",lwd=2)
#
plot(ns$FCI,ns$ID.F,pch=20,col="blue",cex=2,
     ylab="Indirect-to-Direct Flow Ratio (I/D, Realized)",
     xlab="Finn Cycling Index (FCI)",
     xlim=c(0,0.8),ylim=c(0,8))
```




```
## Remove the plotting parameters
rm(opar)
```

```
{r, fig=TRUE,echo=FALSE,fig.cap="Ratio of Indirect-to-Direct Flow   for six ecosystem
models (left) and relationship between the Finn   Cycling Index and the ratio of Indirect-to-Direct
flow in the 74   ecosysetm models."} <<idf>>
```

A strength of this software is the ease with which users can apply ENA to multiple models. We expect that this will simplify users' analytic workflows and reduce the time required to conduct the work.

Connecting to Other Useful Software

Another advantage of building the *enaR* package in **R** is that it lets ecologists take advantage of other types of network analysis and statistical tools that already exist in **R**. We highlight three examples here.

network

enaR uses the network data object introduced in the *network* package (Butts 2008a). One advantage of using this data object is that analysts can then use the tools for network construction and manipulation that are part of the *network* package. For example, *network* can import network models from Pajek project files, which is another widely used network modeling and analysis software (Batagelj and Mrvar 2007). The package also includes functions to seamlessly add and delete nodes (edges). It also provides the capability to visualize the network shown previously.

sna: Social Network Analysis

The *sna* package for Social Network Analysis is bundled in the *statnet* package and uses the same network data object defined in *network*. Thus, the design decision to use the network data object gives users direct access to *sna* tools.

As an example, the *sna* package provides a way of calculating several common centrality measures. Thus, ecologists can now use the *sna* algorithms to determine different types of centrality for their models. This includes betweenness and closeness centrality as follows:

```
sna::betweenness(oyster)
```

```
## [1] 0.0 0.0 0.5 3.5 0.0 9.0
```

```
sna::closeness(oyster)
```

```
## [1] 0.625 0.000 0.000 0.000 0.000 0.000
```

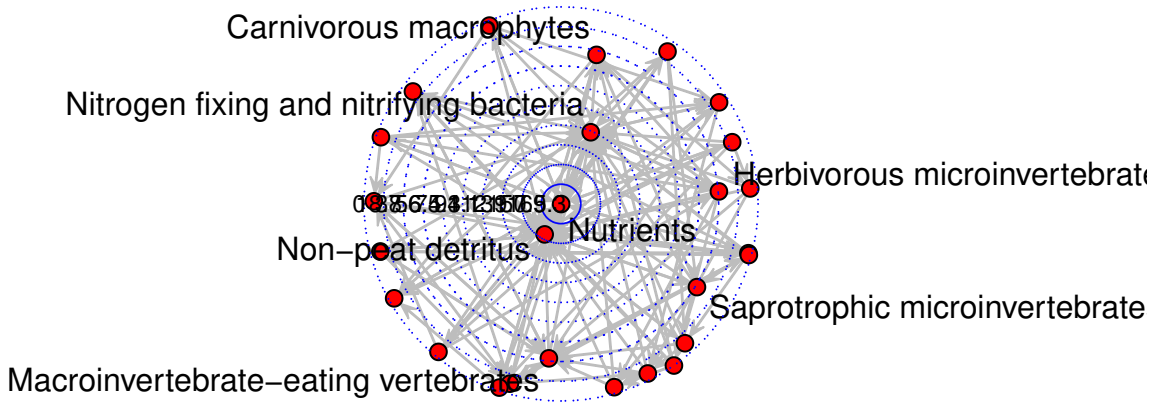
The *sna* package introduced new graphical capabilities as well. For example, it will create a target diagram to visualize the centralities.

```
m <- m.list[[17]]  # Okefenokee Food Web
## Calculate betweenness centrality
b <- sna::betweenness(m)
## Get vertex names
nms <- m%v%'vertex.names'
show(nms)
```

```
##  [1] "Peat decomposers"
##  [2] "Detritus decomposers"
##  [3] "Nitrogen fixing and nitrifying bacteria"
##  [4] "Autotrophic macrophytes"
##  [5] "Carnivorous macrophytes"
##  [6] "Phytoplankton"
##  [7] "Periphyton"
##  [8] "Filamentous algae"
##  [9] "Herbivorous microinvertebrates"
## [10] "Predaceaous microinvertebrates"
## [11] "Saprotrophic microinvertebrates"
## [12] "Algae-eating macroinvertebrates"
## [13] "Macrophyte-eating macroinvertebrates"
## [14] "Microinvertebrate-eating macroinvertebrates"
## [15] "Macroinvertebrate-eating macroinvertebrates"
## [16] "Vertebrate-eating macroinvertebrates"
## [17] "Saprotrophic macroinvertebrates"
## [18] "Algae-eating vertebrates"
## [19] "Macrophyte-eating vertebrates"
## [20] "Microinvertebrate-eating vertebrates"
## [21] "Macroinvertebrate-eating vertebrates"
## [22] "Vertebrate-eating vertebrates"
## [23] "Saprotrophic vertebrates"
## [24] "Superficial peat"
## [25] "Non-peat detritus"
## [26] "Nutrients"
```

```
## Exclude less central node names
nms[b<=(0.1*max(b))] <- NA

set.seed(2)
opar <- par(xpd=TRUE,mfrow=c(1,1))
## Create target plot showing only
## labels of most central nodes
gplot.target(m,b,
             edge.col="grey",
             label=nms)
```




```
## Remove plot settings
rm(opar)
```

```
{r, fig=TRUE,echo=FALSE,eval=TRUE,fig.cap="Target plot of node   betweenness centrality
for the Okefenokee Swamp trophic model."} <<target>>
```

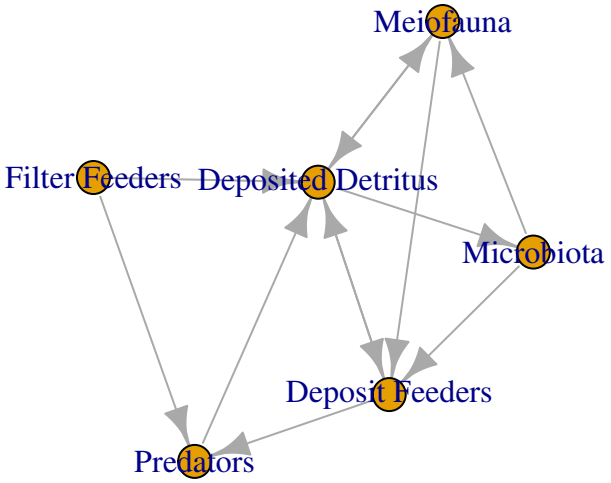
In addition to the node-level measures, *sna* includes graph-level indices.

iGraph

The *iGraph* package can also be useful for analyzing network data. Here are a few examples of using the package. Note that some functions in *iGraph* conflict with other functions already defined, so care is required when using *iGraph*.

```
## The adjacency matrix
A <- St$A

## Creating an iGraph graph
g <- igraph::graph.adjacency(A)
plot(g)
```



iGraph has a different set of visualization tools and generates a different looking plot.

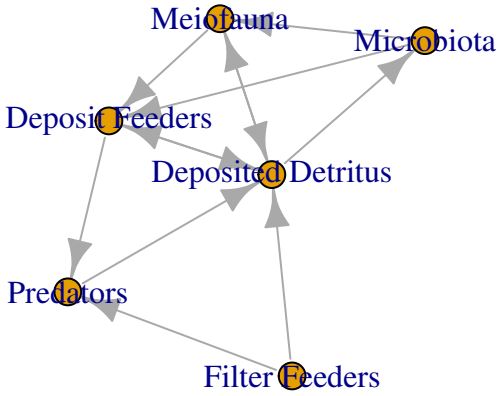


Figure 3: Plot of Oyster reef model using iGraph

```
## Betweenness centrality (calculated by iGraph and sna)
igraph::betweenness(g)
```

```
##     Filter Feeders         Microbiota          Meiofauna
##                0.0                0.0                0.5
##    Deposit Feeders          Predators Deposited Detritus
##                3.5                0.0                9.0
```

```
## Shortest path between any two nodes
igraph::shortest.paths(g)
```

```
##                    Filter Feeders Microbiota Meiofauna Deposit Feeders
## Filter Feeders                  0          2         2               2
## Microbiota                      2          0         1               1
## Meiofauna                       2          1         0               1
## Deposit Feeders                 2          1         1               0
## Predators                       1          2         2               1
## Deposited Detritus              1          1         1               1
##                    Predators Deposited Detritus
```

```
## Filter Feeders             1                 1
## Microbiota                 2                 1
## Meiofauna                  2                 1
## Deposit Feeders            1                 1
## Predators                  0                 1
## Deposited Detritus         1                 0
```

```
## Average path length in the network (graph theory sense)
igraph::average.path.length(g,directed=TRUE)
```

```
## [1] 1.52
```

```
## Diameter of the graph
igraph::diameter(g)
```

```
## [1] 2
```

```
## Connectivity of the group and sub-components
igraph::vertex.connectivity(g)  # connectivity of a graph (group cohesion)
```

```
## [1] 0
```

```
igraph::subcomponent(g,1,'in')  # subcomponent reachable from 1 along inputs
```

```
## + 1/6 vertex, named:
## [1] Filter Feeders
```

```
igraph::subcomponent(g,2,'in')  # subcomponent reachable from 2 along inputs
```

```
## + 6/6 vertices, named:
## [1] Microbiota         Deposited Detritus Filter Feeders
## [4] Meiofauna          Deposit Feeders    Predators
```

```
igraph::subcomponent(g,1,'out') # subcomponent reachable from 1 along outputs
```

```
## + 6/6 vertices, named:
## [1] Filter Feeders     Predators          Deposited Detritus
## [4] Microbiota         Meiofauna          Deposit Feeders
```

```
igraph::subcomponent(g,2,'out') # subcomponent reachable from 2 along output
```

```
## + 5/6 vertices, named:
## [1] Microbiota         Meiofauna          Deposit Feeders
## [4] Deposited Detritus Predators
```

```
igraph::edge.connectivity(g)
```

```
## [1] 0
```

bipartite

The bipartite package provides as set of functions largely developed directly from community ecology for the analysis of two-mode networks (e.g. plant-pollintor, plant-disperser, predator-prey). To facilitate analysis of ecosystem networks using the bipartite toolbox, we created a simple function for converting ecosystem models in the network format to a bipartite matrix. Here's a quick example using the Oyster Reef model (Dame and Patten 1981) where we create a vector of "membership" to divide the ecosystem compartments to create a bipartite network.

```
as.bipartite(x = oyster, y = gl(2, 3))
```

```
##                Deposit Feeders Predators Deposited Detritus
## Filter Feeders          0.0000    0.5135             15.7910
## Microbiota              1.2060    0.0000              0.0000
## Meiofauna               0.6609    0.0000              4.2403
```

EcoNet

The *EcoNet* software is an online, web-interface that provides a tool box for dynamic modeling and ENA analytics (Kazanci 2007). We have provided a write function that enables *enaR* users to output models for easy input into the *EcoNet* interface. The *EcoNet* package and details on the model input syntax can be found at http://eco.engr.uga.edu. Here is an example of how to use the write.EcoNet function in *enaR* in your current working directory:

```
data(oyster)
write.EcoNet(oyster, file = 'oyster.txt', mn = 'oyster_model')
oyster <- read.EcoNet(file = 'oyster.txt')
```

Models can also be read from the set hosted on the *EcoNet* website. If you know the name of the model that you want, you can request it directly. If not, you can leave the input empty to receive a prompt detailing the list of models:

```
EcoNetWeb(model.name = "Intertidal Oyster Reef Ecosystem Model")
EcoNetWeb()
```

Conclusion

These examples show how to use the key features of the *enaR* package that enables scientists to perform Ecosystem Network Analysis in **R**. The vision for this package is that it provides access to ENA algorithms from both the Ulanowicz and Patten Schools to facilitate theoretical synthesis and broader application. In its current form it replicates, updates, and extends the functionality of the NEA.m function (Fath and Borrett 2006) and replicates much of the main analyses in NETWRK (Ulanowicz and Kay 1991). Through the connections that *enaR* provides to other **R** packages users can connect to other network analyses provided by packages, such as *sna* and *iGraph*. There are other **R** packages that have graph and network analysis tools, like *Bioconductor*, *WGCNA*, *tnet* and *rmangal*, that might also be useful for ecologists. Our aim is for *enaR* to serve as a nexus for the introduction of analyses from the broader field of network theory into ecology. In addition, we would like to invite users to connect, collaborate and contribute to development of ENA theory and *enaR*. Programmers that are interested can visit https://github.com/SEELab/enaR_development for more information on how to contribute to development of the *enaR* package.

References

Allesina, S., and C. Bondavalli. 2003. "Steady State of Ecosystem Flow Networks: A Comparison Between Balancing Procedures." *Ecol. Model.* 165: 221–29.

———. 2004. "WAND: An Ecological Network Analysis User-Friendly Tool." *Environ. Model. Softw.* 19: 337–40.

Almunia, J., G. Basterretxea, J. Aistegui, and R.E. Ulanowicz. 1999. "Benthic–pelagic Switching in a Coastal Subtropical Lagoon." *Estuar. Coast. Shelf Sci.* 49: 221–32.

Baird, D., and H. Milne. 1981. "Energy Flow in the Ythan Estuary, Aberdeenshire, Scotland." *Estuar. Coast. Shelf Sci.* 13: 455–72.

Baird, D., and R. E. Ulanowicz. 1989. "The Seasonal Dynamics of the Chesapeake Bay Ecosystem." *Ecol. Monogr.* 59: 329–64.

Baird, D., H. Asmus, and R. Asmus. 2004. "Energy Flow of a Boreal Intertidal Ecosystem, the Sylt-Rømø Bight." *Mar. Ecol. Prog. Ser.* 279: 45–61.

———. 2008. "Nutrient Dynamics in the Sylt-Rømø Bight Ecosystem, German Wadden Sea: An Ecological Network Analysis Approach." *Estuar. Coast. Shelf Sci.* 80: 339–56.

Baird, D., R. R. Christian, C. H. Peterson, and G. A. Johnson. 2004. "Consequences of Hypoxia on Estuarine Ecosystem Function: Energy Diversion from Consumers to Microbes." *Ecol. Appl.* 14: 805–22.

Baird, D., J. Luczkovich, and R. R. Christian. 1998. "Assessment of Spatial and Temporal Variability in Ecosystem Attributes of the St Marks National Wildlife Refuge, Apalachee Bay, Florida." *Estuar. Coast. Shelf Sci.* 47: 329–49.

Baird, D., J. M. McGlade, and R. E. Ulanowicz. 1991. "The Comparative Ecology of Six Marine Ecosystems." *Philos. Trans. R. Soc. Lond. B* 333: 15–29.

Baird, D., R. E. Ulanowicz, and W. R. Boynton. 1995. "Seasonal Nitrogen Dynamics in Chesapeake Bay—a Network Approach." *Estuar. Coast. Shelf Sci.* 41 (2): 137–62.

Barabási, Albert-László. 2012. "The Network Takeover." *Nat. Phys.* 8 (1): 14–16.

Barber, M. C. 1978a. "A Retrospective Markovian Model for Ecosystem Resource Flow." *Ecol. Model.* 5 (2). Elsevier: 125–35.

———. 1978b. "A Markovian Model for Ecosystem Flow Analysis." *Ecol. Model.* 5 (3). Elsevier: 193–206.

Batagelj, V., and A. Mrvar. 2007. "Pajek: Package for Large Network Analysis." University of Ljubljana, Slovenia. http://mrvar.fdv.uni-lj.si/pajek/.

Belgrano, A., U.M. Scharler, J. Dunne, and R.E. Ulanowicz. 2005. *Aquatic Food Webs: An Ecosystem Approach*. New York, NY.: Oxford University Press.

Bondavalli, C., and R. E. Ulanowicz. 1999. "Unexpected Effects of Predators Upon Their Prey: The Case of the American Alligator." *Ecosystems* 2: 49–63.

Borgatti, S.P., and M.G. Everett. 2006. "A Graph-Theoretic Perspective on Centrality." *Soc. Networks* 28: 466–84.

Borrett, S. R. 2013. "Throughflow Centrality Is a Global Indicator of the Functional Importance of Species in Ecosystems." *Ecol. Indic.* 32: 182–96. doi:10.1016/j.ecolind.2013.03.014.

Borrett, S. R., and M. K. Lau. 2014. "enaR: An R Package for Ecological Network Analysis." *Methods. Ecol. Evol.* 11: 1206–13.

Borrett, S. R., and O. O. Osidele. 2007. "Environ Indicator Sensitivity to Flux Uncertainty in a Phosphorus Model of Lake Sidney Lanier, USA." *Ecol. Model.* 200: 371–83.

Borrett, S. R., and B. C. Patten. 2003. "Structure of Pathways in Ecological Networks: Relationships Between Length and Number." *Ecol. Model.* 170: 173–84.

Borrett, S. R., B. D. Fath, and B. C. Patten. 2007. "Functional Integration of Ecological Networks Through Pathway Proliferation." *J. Theor. Biol.* 245: 98–111.

Borrett, S. R., D. E. Hines, and M. Carter. 2015. "Six General Ecosystem Properties Are More Intense in Biogeochemical Cycling Networks Than Food Webs." *Still Seeking a Home*.

Borrett, S. R., J. Moody, and A. Edelmann. 2014. "The Rise of Network Ecology: Maps of the Topic Diversity and Scientific Collaboration." *Ecol. Model.* 294: 111–27. doi:10.1016/j.ecolmodel.2014.02.019.

Borrett, S. R., S. J. Whipple, B. C. Patten, and R. R. Christian. 2006. "Indirect Effects and Distributed Control in Ecosystems 3. Temporal Variability of Indirect Effects in a Seven-Compartment Model of Nitrogen Flow in the Neuse River Estuary (USA)—Time Series Analysis." *Ecol. Model.* 194: 178–88.

Borrett, S.R., and M.A. Freeze. 2011. "Reconnecting Environs to Their Environment." *Ecol. Model.* 222: 2393–2403. doi:10.1016/j.ecolmodel.2010.10.015.

Borrett, S.R., and A.K. Salas. 2010. "Evidence for Resource Homogenization in 50 Trophic Ecosystem Networks." *Ecol. Model.* 221: 1710–16. doi:10.1016/j.ecolmodel.2010.04.004.

Borrett, S.R., R.R. Christian, and R.E. Ulanowicz. 2012. "Network Ecology." In *Encyclopedia of Environmetrics*, edited by A.H. El-Shaarawi and W.W. Piegorsch, 2nd ed., 1767–72. John Wiley & Sons. doi:doi:10.1002/9780470057339.van011.pub2.

Borrett, S.R., M.A. Freeze, and A.K. Salas. 2011. "Equivalence of the Realized Input and Output Oriented Indirect Effects Metrics in Ecological Network Analysis." *Ecol. Model.* 222: 2142–48. doi:10.1016/j.ecolmodel.2011.04.003.

Borrett, S.R., S.J. Whipple, and B.C. Patten. 2010. "Rapid Development of Indirect Effects in Ecological Networks." *Oikos* 119: 1136–48.

Brandes, U., and T. Erlebach, eds. 2005. *Network Analysis: Methodological Foundations*. Berlin, Heidelberg: Springer-Verlag.

Brandes, U., G. Robins, A. McCranie, and S. Wasserman. 2013. "What Is Network Science?" *Network Sci.* 1 (01). Cambridge Univ Press: 1–15.

Brylinsky, M. 1972. "Steady-State Sensitivity Analysis of Energy Flow in a Marine Ecosystem." In *Systems Analysis and Simulation in Ecology*, edited by B.C. Patten, 2:81–101. Academic Press.

Butts, C.T. 2008a. "Network: A Package for Managing Relational Data in R." *J. Stat. Softw.* 24.

———. 2008b. "Social Network Analysis with Sna." *J. Stat. Softw.* 24 (6): 1–51.

Caswell, H. 2001. *Matrix Population Models: Construction, Analysis, and Interpretation*. 2nd ed. Sunderland, Mass.: Sinauer Associates.

Chen, S., and B. Chen. 2012. "Network Environ Perspective for Urban Metabolism and Carbon Emissions: A Case Study of Vienna, Austria." *Environ. Sci. Tech.* 46 (8): 4498–4506.

Christensen, V. 1995. "Ecosystem Maturity—Towards Quantification." *Ecol. Model.* 77: 3–32.

Christensen, V., and D. Pauly. 1992. "Ecopath-II—A Software for Balancing Steady-State Ecosystem Models and Calculating Network Characteristics." *Ecol. Model.* 61: 169–85.

Christensen, V., and C. J. Walters. 2004. "Ecopath with Ecosim: Methods, Capabilities and Limitations." *Ecol. Model.* 172: 109–39.

Christian, R. R., and C. R. Thomas. 2003. "Network Analysis of Nitrogen Inputs and Cycling in the Neuse River Estuary, North Carolina, USA." *Estuaries* 26: 815–28.

Christian, R. R., E. Fores, F. Comin, P. Viaroli, M. Naldi, and I. Ferrari. 1996. "Nitrogen Cycling Networks of Coastal Ecosystems: Influence of Trophic Status and Primary Producer Form." *Ecol. Model.* 87: 111–29.

Dame, R. F., and B. C. Patten. 1981. "Analysis of Energy Flows in an Intertidal Oyster Reef." *Mar. Ecol. Prog. Ser.* 5: 115–24.

Dunne, J.A., R.J. Williams, and N.D. Martinez. 2002. "Food-Web Structure and Network Theory: The Role of Connectance and Size." *Proc. Nat. Acad. Sci. USA* 99: 12917.

Edmisten, J. 1970. "Preliminary Studies of the Nitrogen Budget of a Tropical Rain Forest." In *A Tropical Rain Forest*, edited by H.T. Odum and R.F. Pigeon, 211–15. TID-24270. USAEC Technical Information Center.

Eklöf, A., and B. Ebenman. 2006. "Species loss and secondary extinctions in simple and complex model communities." *Journal of Animal Ecology* 75 (1): 239–46.

Estrada, E. 2007. "Food Webs Robustness to Biodiversity Loss: The Roles of Connectance, Expansibility and Degree Distribution." *J. Theor. Biol.* 244 (2): 296–307. doi:10.1016/j.jtbi.2006.08.002.

Fann, S.L., and S.R. Borrett. 2012. "Environ Centrality Reveals the Tendency of Indirect Effects to Homogenize the Functional Importance of Species in Ecosystems." *J. Theor. Biol.* 294: 74–86.

Farkas, I.J., I. Derenyi, A.L. Barabasi, and T. Vicsek. 2001. "Spectra of 'Real-World' Graphs: Beyond the Semicircle Law." *Physical Review E* 64 (2): 026704.

Fath, B. D. 2004. "Network Analysis Applied to Large-Scale Cyber-Ecosystems." *Ecol. Model.* 171: 329–37.

Fath, B. D., and S. R. Borrett. 2006. "A Matlab© Function for Network Environ Analysis." *Environ. Model. Softw.* 21: 375–405.

Fath, B. D., and B. C. Patten. 1998. "Network Synergism: Emergence of Positive Relations in Ecological Systems." *Ecol. Model.* 107: 127–43.

———. 1999. "Review of the Foundations of Network Environ Analysis." *Ecosystems* 2: 167–79.

Fath, B. D., B. C. Patten, and J. S. Choi. 2001. "Complementarity of Ecological Goal Functions." *J. Theor. Biol.* 208: 493–506.

Fath, B. D., U. M. Scharler, R. E. Ulanowicz, and B. Hannon. 2007. "Ecological Network Analysis: Network Construction." *Ecol. Model.* 208: 49–55.

Fath, Brian D. 2014. "Quantifying Economic and Ecological Sustainability." *Ocean & Coastal Management*. Elsevier.

Finn, J. T. 1976. "Measures of Ecosystem Structure and Function Derived from Analysis of Flows." *J. Theor. Biol.* 56: 363–80.

———. 1980. "Flow Analysis of Models of the Hubbard Brook Ecosystem." *Ecology* 61: 562–71.

Freeman, L.C. 1979. "Centrality in Networks. I. Conceptual Clarificaiton." *Soc. Networks* 1: 215–39.

Freeman, L.C., S.P. Borgatti, and D.R. White. 1991. "Centrality in Valued Graphs: A Measure of Betweenness Based on Network Flow." *Social Networks* 13 (2): 141–54.

Gattie, D. K., J. R. Schramski, S. R. Borrett, B. C. Patten, S. A. Bata, and S. J. Whipple. 2006. "Indirect Effects and Distributed Control in Ecosystems: Network Environ Analysis of a Seven-Compartment Model of Nitrogen Flow in the Neuse River Estuary, USA—Steady-State Analysis." *Ecol. Model.* 194: 162–77.

Handcock, M.S., D.R. Hunter, C.T. Butts, S.M. Goodreau, and M. Morris. 2008. "Statnet: Software Tools for the Representation, Visualization, Analysis and Simulation of Network Data." *J. Stat. Softw.* 24 (1). NIH Public Access: 1548.

Haven, D.S., and R. Morales-Alamo. 1966. "Aspects of Biodeposition by Oysters and Other Invertebrate Filter Feeders." *Limnol. Oceanogr.* 11. JSTOR: 487–98.

Heymans, J. J., and D. Baird. 2000. "A Carbon Flow Model and Network Analysis of the Northern Benguela Upwelling System, Namibia." *Ecol. Model.* 126: 9–32.

Higashi, M., and T. P. Burns. 1991. *Theoretical Studies of Ecosystems: The Network Perspective*. Cambridge: Cambridge University Press.

Higashi, M., and B. C. Patten. 1989. "Dominance of Indirect Causality in Ecosystems." *Am. Nat.* 133: 288–302.

Higashi, M., B. C. Patten, and T. P. Burns. 1993. "Network Trophic Dynamics: The Modes of Energy Utilization in Ecosystems." *Ecol. Model.* 66 (1). Elsevier: 1–42.

Hines, D.E., and S.R. Borrett. 2014. "A Comparison of Network, Neighborhood, and Node Levels of Analyses in Two Models of Nitrogen Cycling in the Cape Fear River Estuary." *Ecological Modelling* 293. Elsevier: 210–20.

Hines, D.E., J.A. Lisa, B. Song, C.R. Tobias, and S.R. Borrett. 2012. "A Network Model Shows the Importance of Coupled Processes in the Microbial N Cycle in the Cape Fear River Estuary." *Estuar. Coast. Shelf Sci.* 106: 45–57. doi:10.1016/j.ecss.2012.04.018.

———. 2015. "Estimating the Effects of Seawater Intrusion on an Estuarine Nitrogen Cycle by Comparative Network Analysis." *Mar. Ecol. Prog. Ser.* 524: 137–54.

Hinrichsen, U., and F. Wulff. 1998. "Biogeochemical and Physical Controls of Nitrogen Fluxes in a Highly Dynamic Marine Ecosystem—model and Network Flow Analysis of the Baltic Sea." *Ecol. Model.* 109: 165–91.

Hooper, D.U., F.S. Chapin III, J.J. Ewel, A. Hector, P. Inchausti, S. Lavorel, J.H. Lawton, et al. 2005. "Effects of Biodiversity on Ecosystem Functioning: A Consensus of Current Knowledge." *Ecol. Monogr.* 75 (1). Eco Soc America: 3–35.

Ings, T. C., J. M. Montoya, J. Bascompte, N. Blüthgen, L. Brown, C. F Dormann, F. Edwards, et al. 2009. "Ecological networks–beyond food webs." *The Journal of Animal Ecology* 78 (1): 253–69. doi:10.1111/j.1365-2656.2008.01460.x.

Jordan, C.F., J.R. Kline, and D.S. Sasscer. 1972. "Relative Stability of Mineral Cycles in Forest Ecosystems." *Am. Nat.* 106. JSTOR: 237–53.

Jørgensen, S. E., B. D. Fath, S. Bastianoni, J. C. Marques, F. Müller, S. Nielsen, B. C. Patten, E. Tiezzi, and R. E. Ulanowicz. 2007. *A New Ecology: Systems Perspective*. Amsterdam: Elsevier.

Kazanci, C. 2007. "EcoNet: A New Software for Ecological Modeling, Simulation and Network Analysis." *Ecol. Model.* 208: 3–8.

Latham II, L. G. 2006. "Network Flow Analysis Algorithms." *Ecol. Model.* 192 (3). Elsevier: 586–600.

Lindeman, R. L. 1942. "The Trophic-Dynamic Aspect of Ecology." *Ecology* 23: 399–418.

Link, J., W. Overholtz, J. O'Reilly, J. Green, D. Dow, D. Palka, C. Legault, et al. 2008. "The Northeast US Continental Shelf Energy Modeling and Analysis Exercise (EMAX): Ecological Network Model Development and Basic Ecosystem Metrics." *J. Mar. Syst.* 74: 453–74.

Martinez, N.D. 1992. "Constant Connectance in Community Food Webs." *Am. Nat.* JSTOR, 1208–18.

Miehls, A.L.J., D. M. Mason, K.A. Frank, A. E. Krause, S.D. Peacor, and W.W. Taylor. 2009a. "Invasive Species Impacts on Ecosystem Structure and Function: A Comparison of the Bay of Quinte, Canada, and Oneida Lake, USA, Before and After Zebra Mussel Invasion." *Ecol. Model.* 220: 3182–93.

Miehls, A.L.J., D.M. Mason, K.A. Frank, A.E. Krause, S.D. Peacor, and W.W. Taylor. 2009b. "Invasive Species Impacts on Ecosystem Structure and Function: A Comparison of Oneida Lake, New York, USA, Before and After Zebra Mussel Invasion." *Ecol. Model.* 220 (22): 3194–3209.

Monaco, M.E., and R.E. Ulanowicz. 1997. "Comparative Ecosystem Trophic Structure of Three US Mid-Atlantic Estuaries." *Mar. Ecol. Prog. Ser.* 161: 239–54.

National Research Council, Committee on Network Science for Army Applications. 2006. *Network Science*. Washington, DC: The National Academies Press.

Newman, M. E. J. 2001. "Scientific Collaboration Networks. I. Network Construction and Fundamental Results." *Phys. Rev. E* 64 (1). APS: 016131.

Newman, M., A.-L. Barabási, and D. J. Watts. 2006. *The Structure and Dynamics of Networks*. Princeton, NJ: Princeton University Press.

Niquil, N., E. Chaumillon, G.A. Johnson, X. Bertin, B. Grami, V. David, C. Bacher, H. Asmus, D. Baird, and R. Asmus. 2012. "The Effect of Physical Drivers on Ecosystem Indices Derived from Ecological Network Analysis: Comparison Across Estuarine Ecosystems." *Estuar. Coast. Shelf Sci.* 108: 132–43. doi:{10.1016/j.ecss.2011.12.031}.

Odum, H.T. 1957. "Trophic Structure and Productivity of Silver Springs, Florida." *Ecol. Monogr.* 27: 55–112.

Oevelen, D. van, G. Duineveld, M. Lavaleye, F. Mienis, K. Soetaert, and C. H. R. Heip. 2009. "The Cold-Water Coral Community as a Hot Spot for Carbon Cycling on Continental Margins: A Food-Web Analysis from Rockall Bank (northeast Atlantic)." *Limnology and Oceanography* 54 (6): 1829.

Oevelen, D. van, K. Soetaert, J. J. Middelburg, P. M. J. Herman, L. Moodley, I. Hamels, T. Moens, and C. H. R. Heip. 2006. "Carbon Flows Through a Benthic Food Web: Integrating Biomass, Isotope and Tracer Data." *Journal of Marine Research* 64 (3). Sears Foundation for Marine Research: 453–82.

Oevelen, D. van, K. Van den Meersche, F. J. R. Meysman, K. Soetaert, J. J. Middelburg, and A. F. Vézina. 2010. "Quantifying Food Web Flows Using Linear Inverse Models." *Ecosystems* 13 (1). Springer: 32–45.

Patrício, J., and J. C. Marques. 2006. "Mass Balanced Models of the Food Web in Three Areas Along a Gradient of Eutrophication Symptoms in the South Arm of the Mondego Estuary (Portugal)." *Ecol. Model.* 197 (1). Elsevier: 21–34.

Patten, B. C. 1978. "Systems Approach to the Concept of Environment." *Ohio J. Sci.* 78: 206–22.

———. 1991. "Network Ecology: Indirect Determination of the Life–environment Relationship in Ecosystems." In *Theoretical Studies of Ecosystems: The Network Perspective*, edited by M. Higashi and T. Burns, 288–351. New York: Cambridge University Press.

Patten, B. C., and G. T. Auble. 1981. "System Theory of the Ecological Niche." *Am. Nat.* 117: 893–922.

Patten, B. C., R. W. Bosserman, J. T. Finn, and W. G Cale. 1976. "Propagation of Cause in Ecosystems." In *Systems Analysis and Simulation in Ecology, Vol. IV*, edited by B. C. Patten, 457–579. New York: Academic Press.

Patten, B.C. 1983. "On the Quantitative Dominance of Indirect Effects in Ecosystems." In *Analysis of Ecological Systems: State-of-the-Art in Ecological Modelling*, edited by W. K. Lauenroth, G. V. Skogerboe, and M. Flug, 27–37. Amsterdam: Elsevier.

Ray, S. 2008. "Comparative Study of Virgin and Reclaimed Islands of Sundarban Mangrove Ecosystem Through Network Analysis." *Ecol. Model.* 215: 207–16. doi:10.1016/j.ecolmodel.2008.02.021.

Richey, J.E., R.C. Wissmar, A.H. Devol, G.E. Likens, J.S. Eaton, R.G. Wetzel, W.E. Odum, et al. 1978. "Carbon Flow in Four Lake Ecosystems: A Structural Approach." *Science* 202: 1183–86.

Rybarczyk, H., B. Elkaim, L. Ochs, and N. Loquet. 2003. "Analysis of the Trophic Network of a Macrotidal Ecosystem: The Bay of Somme (Eastern Channel)." *Estuar. Coast. Shelf Sci.* 58: 405–21.

Salas, A.K., and S.R. Borrett. 2011. "Evidence for Dominance of Indirect Effects in 50 Trophic Ecosystem Networks." *Ecol. Model.* 222 (5): 1192–1204. doi:DOI: 10.1016/j.ecolmodel.2010.12.002.

Sandberg, J., R. Elmgren, and F. Wulff. 2000. "Carbon Flows in Baltic Sea Food Webs — a Re-Evaluation Using a Mass Balance Approach." *J. Mar. Syst.* 25: 249–60.

Scharler, U.M., and D. Baird. 2005. "A Comparison of Selected Ecosystem Attributes of Three South African Estuaries with Different Freshwater Inflow Regimes, Using Network Analysis." *J. Mar. Syst.* 56: 283–308.

Scharler, U.M., and B.D. Fath. 2009. "Comparing Network Analysis Methodologies for Consumer–resource Relations at Species and Ecosystems Scales." *Ecol. Model.* 220 (22). Elsevier: 3210–18.

Scharler, UM. 2012. "Ecosystem Development During Open and Closed Phases of Temporarily Open/closed Estuaries on the Subtropical East Coast of South Africa." *Estuar. Coast. Shelf Sci.* 108. Elsevier: 119–31.

Schramski, J. R., C. Kazanci, and E. W. Tollner. 2011. "Network Environ Theory, Simulation and EcoNet© 2.0." *Environ. Model. Softw.* 26: 419–28. doi:10.1016/j.envsoft.2010.10.003.

Schramski, J.R., D.K. Gattie, B.C. Patten, S.R. Borrett, B.D. Fath, C.R. Thomas, and S.J. Whipple. 2006. "Indirect Effects and Distributed Control in Ecosystems: Distributed Control in the Environ Networks of a Seven-Compartment Model of Nitrogen Flow in the Neuse River Estuary, USA—Steady-State Analysis." *Ecol. Model.* 194: 189–201.

Schramski, J.R., D.K. Gattie, B.C. Patten, S.R. Borrett, B.D. Fath, and S.J. Whipple. 2007. "Indirect Effects and Distributed Control in Ecosystems: Distributed Control in the Environ Networks of a Seven-Compartment Model of Nitrogen Flow in the Neuse River Estuary, USA—Time Series Analysis." *Ecol. Model.* 206: 18–30.

Shannon, C. E., and W. Weaver. 1949. *The Mathematical Theory of Information*. Urbana, Illionois: University of Illinois Press.

Small, G. E., R. W. Sterner, and J. C. Finlay. 2014. "An Ecological Network Analysis of Nitrogen Cycling in the Laurentian Great Lakes." *Ecol. Model.* 293. Elsevier: 150–60.

Soetaert, K., K. Van den Meersche, and D. van Oevelen. 2009. *limSolve: Solving Linear Inverse Models*. Comprehensive R Archival Network.

Tilly, L.J. 1968. "The Structure and Dynamics of Cone Spring." *Ecol. Monogr.* 38: 169–97.

Tilman, D, D Wedin, and J Knops. 1996. "Productivity and Sustainability Influenced by Biodiversity in Grassland Ecosystems." *Nature* 379 (6567): 718–20.

Ulanowicz, R. E. 1983. "Identifying the Structure of Cycling in Ecosystems." *Math. Biosci.* 65: 219–37.

———. 1986. *Growth and Development: Ecosystems Phenomenology*. New York: Springer–Verlag.

———. 1997. *Ecology, the Ascendent Perspective*. New York: Columbia University Press.

Ulanowicz, R. E., and D. Baird. 1999. "Nutrient Controls on Ecosystem Dynamics: The Chesapeake Mesohaline Community." *J. Mar. Syst.* 19: 159–72.

Ulanowicz, R. E., and J.J. Kay. 1991. "A Package for the Analysis of Ecosystem Flow Networks." *Environmental Software* 6: 131–42.

Ulanowicz, R. E., and W. M. Kemp. 1979. "Toward Canonical Trophic Aggregations." *Am. Nat.* JSTOR, 871–83.

Ulanowicz, R. E., and C. J. Puccia. 1990. "Mixed Trophic Impacts in Ecosystems." *Coenoses* 5: 7–16.

Ulanowicz, R. E., C. Bondavalli, and M. S. Egnotovich. 1998. *Network Analysis of Trophic Dynamics in South Florida Ecosystem, FY 97: The Florida Bay Ecosystem*. Annual Report to the United States Geological Service Biological Resources Division Ref. No. [UMCES]CBL 98-123. Chesapeake Biological Laboratory, University of Maryland.

Ulanowicz, R. E., C. Bondavalli, J. J. Heymans, and M. S. Egnotovich. 1999. *Network Analysis of Trophic Dynamics in South Florida Ecosystem, FY 98: The Mangrove Ecosystem*. Annual Report to the United States Geological Service Biological Resources Division Ref. No.[UMCES] CBL 99-0073; Technical Report Series No. TS-191-99. Chesapeake Biological Laboratory, University of Maryland.

———. 2000. *Network Analysis of Trophic Dynamics in South Florida Ecosystem, FY 99: The Graminoid Ecosystem*. Annual Report to the United States Geological Service Biological Resources Division Ref. No. [UMCES] CBL 00-0176. Chesapeake Biological Laboratory, University of Maryland.

Ulanowicz, R. E., R. D. Holt, and M. Barfield. 2014. "Limits on Ecosystem Trophic Complexity: Insights from Ecological Network Analysis." *Ecol. Lett.* 17 (2): 127–36.

Ulanowicz, R.E., and U.M. Scharler. 2008. “Least-Inference Methods for Constructing Networks of Trophic Flows.” *Ecol. Model.* 210 (3): 278–86.

Ulanowicz, R.E., C. Bondavalli, and M.S. Egnotovich. 1997. *Network Analysis of Trophic Dynamics in South Florida Ecosystem, FY 96: The Cypress Wetland Ecosystem.* Annual Report to the United States Geological Service Biological Resources Division Ref. No. [UMCES]CBL 97-075. Chesapeake Biological Laboratory, University of Maryland.

Vézina, A. F., and T. Platt. 1988. “Food Web Dynamics in the Ocean. 1. Best-Estimates of Flow Networks Using Inverse Methods.” *Mar. Ecol. Prog. Ser.* 42 (3): 269–87.

Waide, J.B., J.E. Krebs, S.P. Clarkson, and E.M. Setzler. 1974. “A Linear System Analysis of the Calcium Cycle in a Forested Watershed Ecosystem.” *Prog. Theor. Biol.* 3: 261–345.

Wasserman, S., and K. Faust. 1994. *Social Network Analysis: Methods and Applications.* Cambridge; New York: Cambridge University Press.

Watts, D.J., and S.H. Strogatz. 1998. “Collective Dynamics of ‘Small-World’ Networks.” *Nature* 393 (6684): 440–42.

Whipple, S. J., and B. C. Patten. 1993. “The Problem of Nontrophic Processes in Trophic Ecology: Toward a Network Unfolding Solution.” *J. Theor. Biol.* 163: 393–411.

Whipple, S. J., S. R. Borrett, B. C. Patten, D. K. Gattie, J. R. Schramski, and S. A. Bata. 2007. “Indirect Effects and Distributed Control in Ecosystems: Comparative Network Environ Analysis of a Seven-Compartment Model of Nitrogen Flow in the Neuse River Estuary, USA—Time Series Analysis.” *Ecol. Model.* 206: 1–17.

Whipple, S. J., B. C. Patten, and S. R. Borrett. 2014. “Indirect Effects and Distributed Control in Ecosystems: Comparative Network Environ Analysis of a Seven-Compartment Model of Nitrogen Storage in the Neuse River Estuary, USA: Time Series Analysis.” *Ecol. Model.* 293. Elsevier: 161–86.

Zhang, Y., Z. Yang, and B.D. Fath. 2010. “Ecological Network Analysis of an Urban Water Metabolic System: Model Development, and a Case Study for Beijing.” *Sci. Total Env.* 408: 4702–11.