rend: An R package for Ecological Network Dynamics

Jonathan J. Borrelli

16 March, 2016

Contents

Abstract:

The structure of ecological communities determines how they respond to environmental impacts such as global climate change and anthropogenic disturbance. Despite continually changing environments and numerous different habitat types, the structure of observed ecological communities exhibit some remarkable similarities. Food web structure is generated by the introduction of new species through invasion and speciation, species loss from extinction, and the altering of interactions through foraging decisions and population dynamics. Many packages currently exist to describe the structure (topology) of networks using the R programming languages, but there are none that implement any of the variety of models developed to understand the dynamics of these systems. In this paper I introduce a new (in-development) R-package *rend that allows users to apply a bioenergetic model of multispecies predator-prey dynamics to trophic networks. The* rend package is being developed with the goal of providing stronger links between the structure and dynamics of complex ecological systems.

1 Introduction

While there is an abundance of packages available in the R programming environment to analyze the structure of networks (e.g., igraph, foodweb, sna, enar, cheddar), there is a dearth of packages that can be used to apply dynamic models to the interacting populations. With the numerous models for multispecies predator-prey dynamics it would be possible to simply utilize the numerical integration package deSolve to implement them, but it is rarely obvious how to go from the mathematics of the model to the required code.

In this paper I present an in-development R package for the simulation and analysis of ecological network dynamics. The package is still in-development because the current version limits the user to the simulation of food web (trophic) dynamics. Future versions will incorporate additional functionality (discussed below).

2 Overview

2.1 Food Web Dynamics Simulation

2.1.1 Consumer-Resource Model

A bioenergetics consumer resource model is at the core of the food web dynamics simulator function. The model was originally developed for two species by Yodzis and Ines (CITATION), generalized to multiple species by Mcann et al. (CITATION). The version included in this version of `rend` is was first presented in Williams and Martinez (CITATION) and used again in Romanuk et al. (CITATION).

$$\frac{dB_i(t)}{dt} = G_i(B) - x_iB_i(t) + \sum_{j}^{n} (x_iy_{ij}F_{ij}(B)B_i(t) - x_jy_{ji}F_{ji}(B)B_j(t)/e_{ji}) \tag{1}$$

There are four basic parts to the model: growth, death, consumption of prey, and being consumed by predators. The first term, $G_i(B)$, is the function describing primary production of producer species in the absence of predation. Producers grow exponentially with density dependence.

$$G_i(B) = r_iB_i(t)(1 - \frac{B_i(t)}{K_i}) \tag{2}$$

In the equation, r_i is the intrinsic rate of increase, B_i is the biomass of population i, and K_i is the carrying capacity of population i. The consumption of resource j by consumer i is modeled by:

$$F_{Hij}(B) = \frac{B_j^{1+q}}{\sum_k B_k^{1+q} + B_0^{1+q}} \tag{3}$$

Here, B_j is the biomass of the consumed resource, in the denominator the summed biomass across all k resources, and B_0 is the half saturation density. The parameter q is a tuning parameter that lets the modeller shift the functional response between a type II ($q = 0$) and a type III ($q = 1$). The functional response above defines the fraction of a predator's maximal ingestion that is realized at a given time step (Figure 1).

In this model, the impact of the predator on the prey is equivalent to the impact of the prey on the predator. Thus the numerical response of the predator is the same magnitude as the functional response of the prey but of the opposite sign. This is accomplished as $F_ji = t(F_ij)$.

An alternative functional response is based on consumer interference (Beddington DeAngelis CITATION).

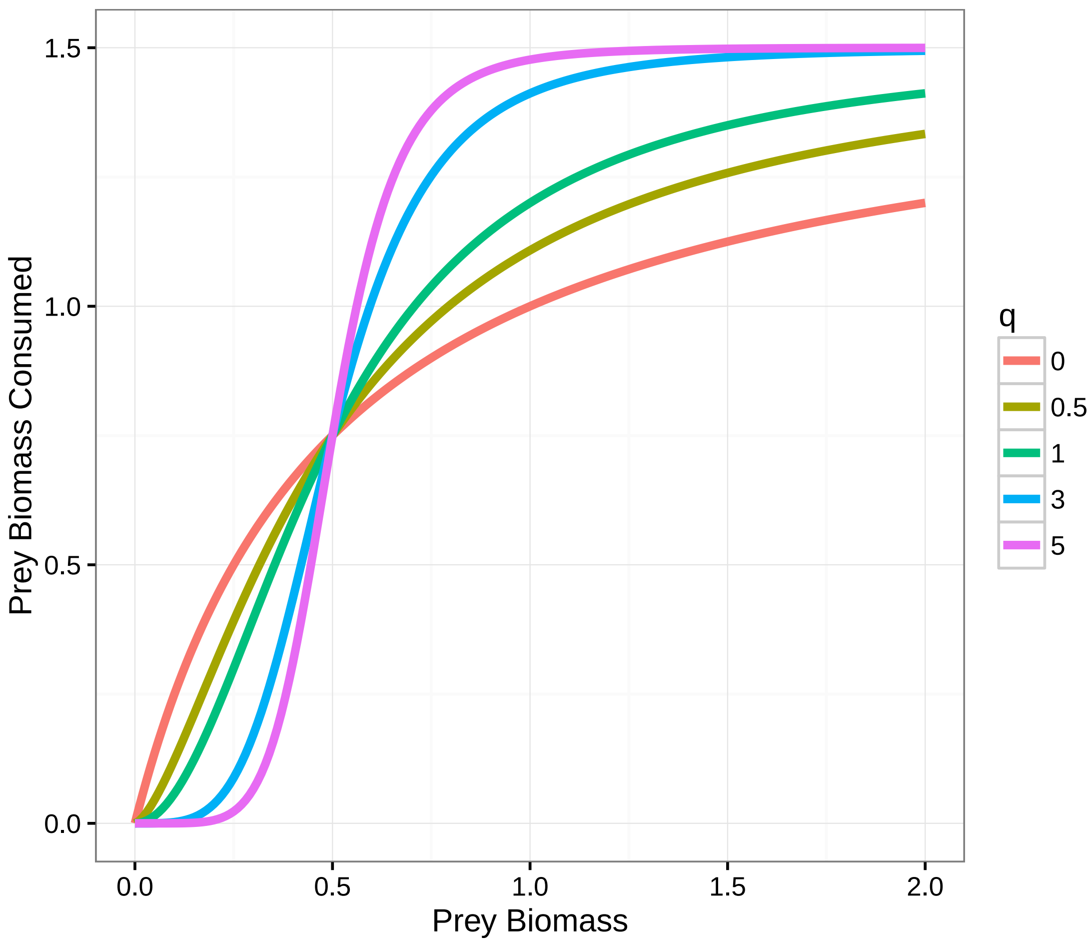

Figure 1: How the functional response of the prey changes with altered values of the parameter q

$$F_{BDij}(B) = \frac{B_j}{\sum_{k=1}^{n} \alpha_{ik} B_k(t) + (1 + c_{ij} B_i(t)) B_{0ji}} \tag{4}$$

Here the parameter c_{ij} gives the strength of the interference. Figure 2 shows how c_{ij} changes the shape of the functional response.

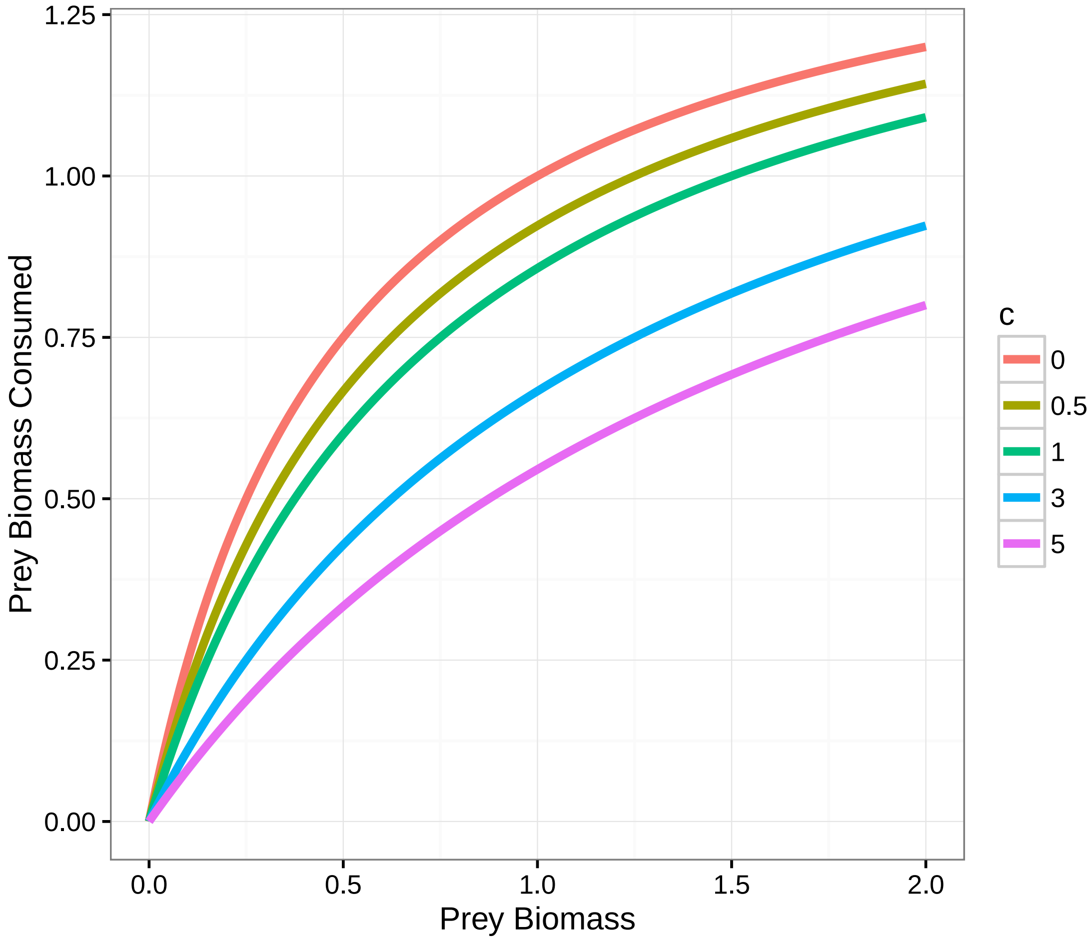

Figure 2: How the functional response of the prey changes with altered values of the parameter c

2.1.2 Implementation

The model is implemented in the code as a numerical integration using the deSolve R package (CITATION). Currently, the primary function is `CRsimulator`, which allows the user to specify all parameter values and the desired functions for the growth of basal species and the functional response. Inputs to `CRsimulator` are described in Table 1.

Table 1: Parameter definitions for the dynamic model

Parameter	Definition
Adj	Adjacency matrix
t	Sequence of time steps

Parameter	Definition
G	Function for basal resource growth
method	Function to input into the ode solver
FuncRes	Functional response
K	Carrying capacity
x.i	Mass specific metabolic rate
yij	Maximum rate at which species i assimilates specesi j per unit metabolic rate of species i
eij	Conversion efficiency
xpar	Tuning parameter either q or c depending on the functional response
B.o	Half saturation density of species j when consumed by species i
ext	Function describing extinction events during the simulation
plot	Whether or not to generate a plot of biomass against time

The `CRsimulator` function can be broken down into four main parts: growth rate, parameter collecting, simulation, and visualization.

The first part, `grow <- getR(Adj)` creates a vector of whether or not species are basal. It uses the `getR` function to assess the column sums of the adjacency matrix to determine which species are basal. If there are no basal species, the function will return a warning (*"No basal species in simulation"*). The second part gathers all required parameters of the consumer resource model into a single list. The default values for these parameters are in Table 2. By default the function for growth is `Gi`, functional response is `Fij`, extinction events is `goExtinct`, and the default method is `CRmod`. All species initially start with a random biomass drawn from a uniform distribution between 0.5 and 1.

Table 2: Fixed parameter values for the dynamic model

Parameter	Value
K	1.0
x.i	0.5
yij	6.0
eij	1.0
xpar	0.2
B.o	0.5

The third part of the `CRsimulator` function is the numerical integration using `deSolve::ode`. This integration requires the method function `CRmod`, which codes the bioenergetic model. This function interacts with the event function, which by default is `goExtinct`, and determines whether a species has gone extinct at each time step by checking whether the species' abundance has passed below the threshold level for extinction ($10^{1}0$).

```
function(t, states, par) {

    with(as.list(c(states, par)), {
        dB <- G.i(r = r.i, B = states, K = K) - x.i * states + rowSums((x.i *
            yij * FR(states, A, B.o, xpar = xpar) * states)) - rowSums((x.i *
            yij * t(FR(states, A, B.o, xpar = xpar) * states))/eij)

        list(c(dB))
    })
}
```

Currently `rend` only has two main types of functional responses: one based on Holling's Type II and Type

III (`Fij`), and a second based on consumer interference (`Fbd`).

Fij

```
function(B, A, B.0, xpar) {
    sum.bk <- rowSums(sapply(1:nrow(A), function(x) {
        B[x] * A[x, ]
    }))^(1 + xpar)
    denom <- sum.bk + B.0^(1 + xpar)

    F1 <- sapply(1:nrow(A), function(x) {
        (B[x] * A[x, ])^(1 + xpar)
    })/denom

    return(F1)
}
```

Fbd

```
function(B, A, B.0, xpar) {
    sum.bk <- rowSums(sapply(1:nrow(A), function(x) {
        B[x] * A[x, ]
    }))
    denom <- sum.bk + (1 + (xpar * B)) * B.0

    F1 <- sapply(1:nrow(A), function(x) {
        (B[x] * A[x, ])
    })/denom

    return(F1)
}
```

Both functions take the same input parameters: `B` is the vector of biomasses, `A` is the adjacency matrix, `B.0` is the half saturation constant, and `xpar` is either the q parameter of the Holling functional response or the consumer interference parameter c. `Fij` and `Fbd` also both return a matrix reflecting the impact of species i on species j.

The last part of `CRsimulator` will plot the output of the integration (when `plot = TRUE`).

2.2 Food Web Dynamics Visualization

In order to create a visualization of the dynamics of the food web through time, the rend package relies on the animation package. With the current version of rend, the user is able to take the output of the `CRsimulator` function and generate a video representation of the dynamics of the time series. Both biomass and interaction dynamics are visualized. Visualization is done by the `netHTML` function, which serves as a wrapper for the `animation` package's `saveHTML`. The function output is an HTML video of the food web at each time step.

```
function(mat, dyn, path1 = getwd()) {
    require(animation)

    lay <- matrix(c(layout.sphere(graph.adjacency(mat))[, 1], TrophInd(mat)$TL),
        ncol = 2)
    s <- matrix(0, nrow = nrow(dyn), ncol = ncol(mat))
```

```
    ani.options(interval = 0.25)
    saveHTML({
        for (i in 1:50) {
            fr <- Fij(dyn[i, -1], mat, 0.5, 0.2)
            strength <- melt(fr)[, 3][melt(fr)[, 3] > 0]
            fr[fr > 0] <- 1

            g.new <- graph.adjacency(t(fr))
            E(g.new)$weight <- strength/max(strength) * 10
            s[i, c(which(dyn[i, -1] > 0))] <- log(dyn[i, c(which(dyn[i, ] >
                0)[-1])]) + abs(min(log(dyn[i, c(which(dyn[i, ] > 0)[-1])])))

            plot.igraph(g.new, vertex.size = s[i, ], edge.width = E(g.new)$weight,
                layout = lay)
        }
    }, img.name = paste(path1, "fwdyn", sep = ""), htmlfile = paste(path1, "fwdyn.html",
        sep = ""), interval = 0.25, nmax = 500, ani.width = 500, ani.height = 500,
        outdir = path1)
}
```

3 Examples

3.1 Two-Species Dynamics (Multiple Functional Response Types)

A simple example of this package is with two species, a basal resource and a consumer. The adjacency matrix is two rows and two columns.

```
     [,1] [,2]
[1,]    0    1
[2,]    0    0
```

All that is required is to feed this adjacency matrix into the `CRsimulator` function, and choose whether to alter any of the default parameter settings. The first six rows of the output are shown in Table 3. Column one is the time step, column two is the biomass species 1 (the basal species), and column three is the biomass of species 2 (the consumer). The resulting biomass dynamics are shown in Figure 3 below. The two species reach equilibrium rather quickly, within 100 time steps, following damped oscillations.

```
my_sim <- CRsimulator(matrix(c(0, 0, 1, 0), nrow = 2, ncol = 2))
```

Table 3: First six rows of the output of the CRsimulator function

time	1	2
1	0.7221	0.7355
2	0.0304	0.9557
3	0.0056	0.6017
4	0.0031	0.3683
5	0.0028	0.2248
6	0.0031	0.1372

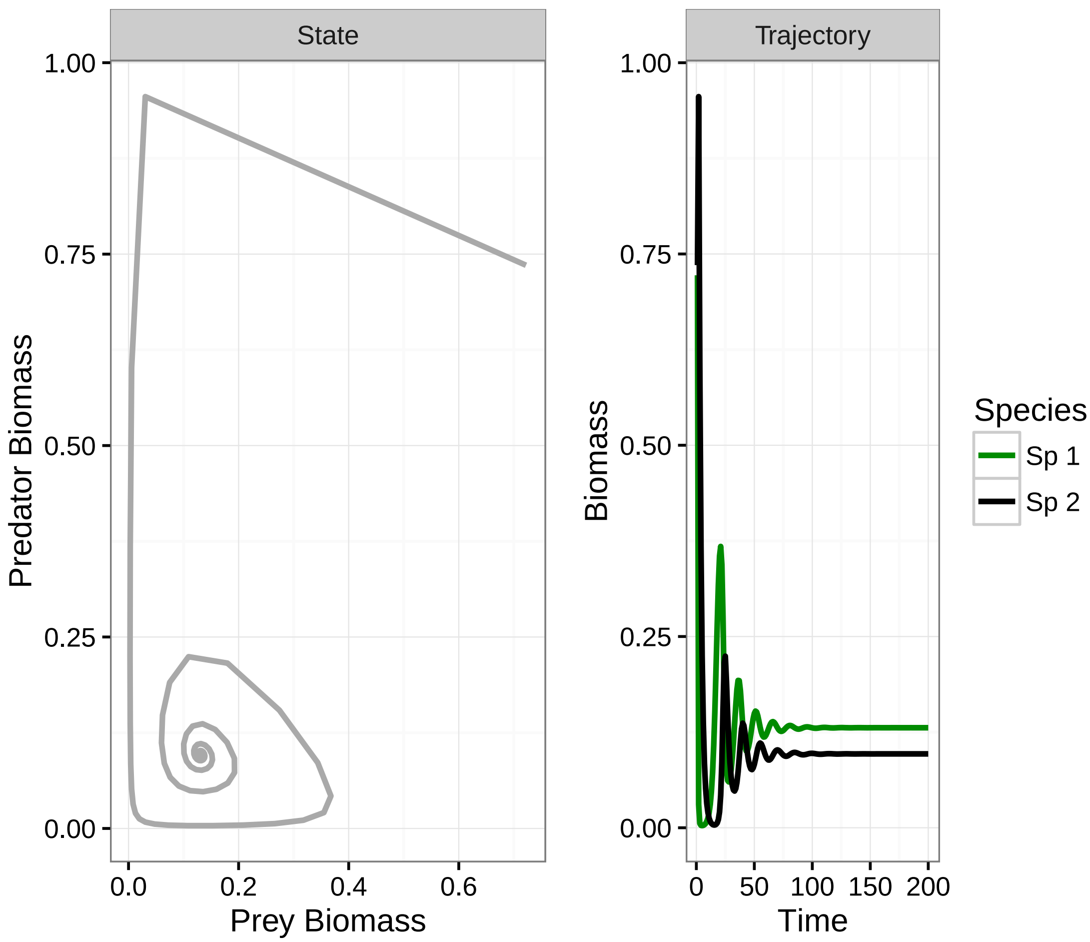

Figure 3: The dynamics of consumer and resource given default settings in CRsimulator

The previous example in Figure 3 simply used the default settings for the `CRsimulator` function. An important setting for the user to select is the form of the functional response. Depending on whether the functional response is Lotka-Volterra or based on consumer intereference, you would expect to observe differences in the dynamics of the two species. The result of altering the functional response, and preserving the other default settings is shown in Figure 4. Looking at the difference between Lotka-Volterra (`Fij`) and consumer interference (`Fbd`) it appears as if the dynamics of the two species with Lotka-Volterra dynamics equilibrate faster. The pair of species with consumer interference are still exhibiting damped oscillations at time step 200, while the other pair are no longer oscillating.

```
my_simbd <- CRsimulator(matrix(c(0, 0, 1, 0), nrow = 2, ncol = 2), FuncRes = Fbd)
```

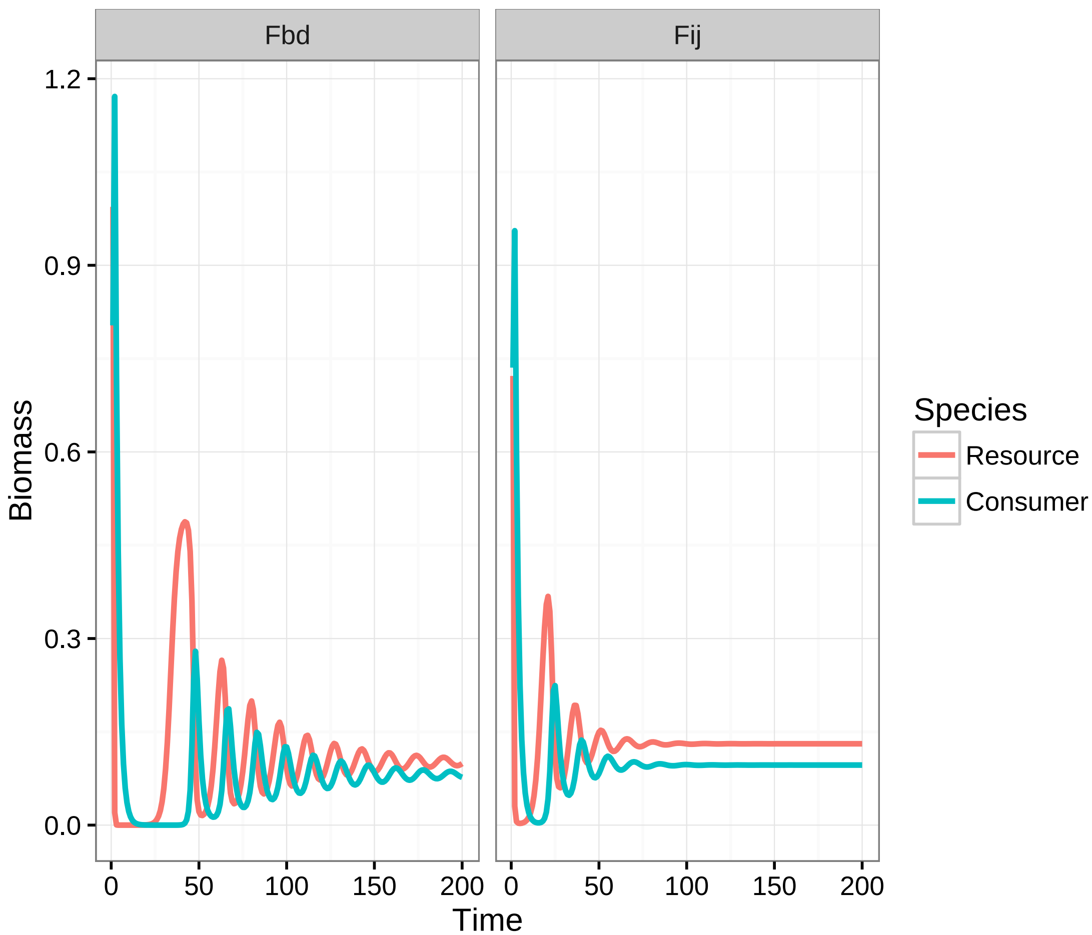

Figure 4: The dynamics of consumer and resource with differing functional responses given default settings in CRsimulator

One would also expect that differences in the tuning parameter (q in the Holling functional response or c for consumer interference) will also alter the dynamics of the interacting species. By systematically altering the tuning parameter (`xpar` in `CRsimulator`) I can show how the dynamics associated with each functional response type change. The results of the two species `CRsimulator` with default settings except for `xpar` equal to 0, 0.2, 1, and 5 for each functional response type are shown in Figure 5.

Williams and Martinez (CITATION) note that for the Holling functional response q of 0 corresponds to a Type II functional response and q of 1 corresponds to a Type III functional response. When q is greater than one, the size of the prey "refugia" is larger as demonstrated by the larger lag in the functional response shown in Figure 1. For a consumer interference functional response (`Fbd`) the parameter c reflects the strength of the interference among consumers.

As the parameter q increases the two species reach equilibrium faster and the equilibrium biomass of the prey (Species A) is higher. With increasing consumer interference the two species also reach equilibrium faster, but there is a smaller difference in equilibrium biomasses. At the highest level of consumer interference ($c = 5$), however, the equilibrium abundances of both species are higher than at lower levels of interference.

```
my_sim2 <- CRsimulator(matrix(c(0, 0, 1, 0), nrow = 2, ncol = 2), xpar = 0)
my_sim3 <- CRsimulator(matrix(c(0, 0, 1, 0), nrow = 2, ncol = 2), xpar = 1)
my_sim4 <- CRsimulator(matrix(c(0, 0, 1, 0), nrow = 2, ncol = 2), xpar = 5)


my_sim2bd <- CRsimulator(matrix(c(0, 0, 1, 0), nrow = 2, ncol = 2), FuncRes = Fbd,
    xpar = 0)
my_sim3bd <- CRsimulator(matrix(c(0, 0, 1, 0), nrow = 2, ncol = 2), FuncRes = Fbd,
    xpar = 1)
my_sim4bd <- CRsimulator(matrix(c(0, 0, 1, 0), nrow = 2, ncol = 2), FuncRes = Fbd,
    xpar = 5)
```

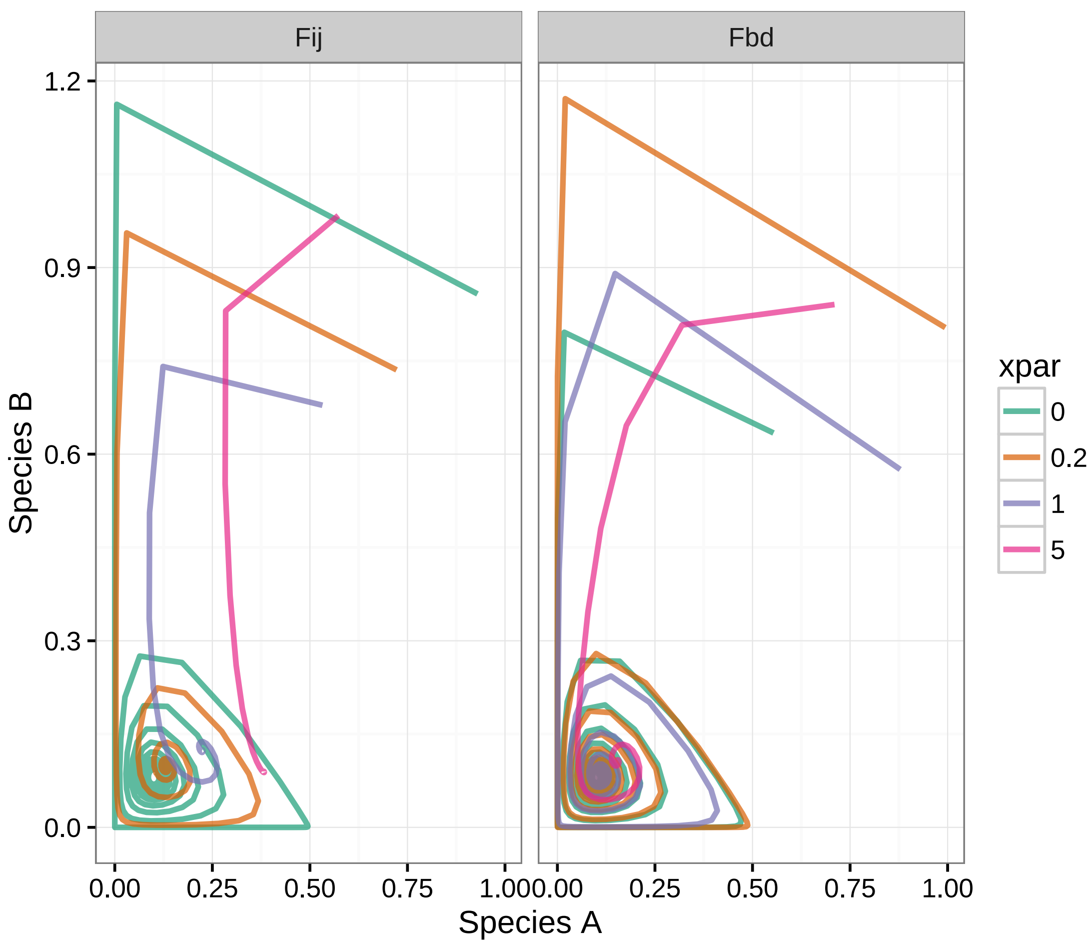

Figure 5: The dynamics of consumer and resource given varying xpar settings in CRsimulator

3.2 Sample Run of Dynamics on Model Food Webs

Most models of food webs generate a binary adjacency matrix (Figure 6) where a one indicates that species i is consumed by species j. This allows for convenient pairing with the `rend` package. The `CRsimulator` function can take the output of these models as input and simulate the dynamics of the participating species. In this example I demonstrate how the package can be used to simulate the dynamics of a niche model food web. The niche model is a food web model that arranges S species along some hypothetical niche axis. Each species consumes other species based on a randomly sampled feeding range with mean c_i sampled from a uniform distribution from 0 to the species' niche value. The range is dependent on the user specified connectance C.

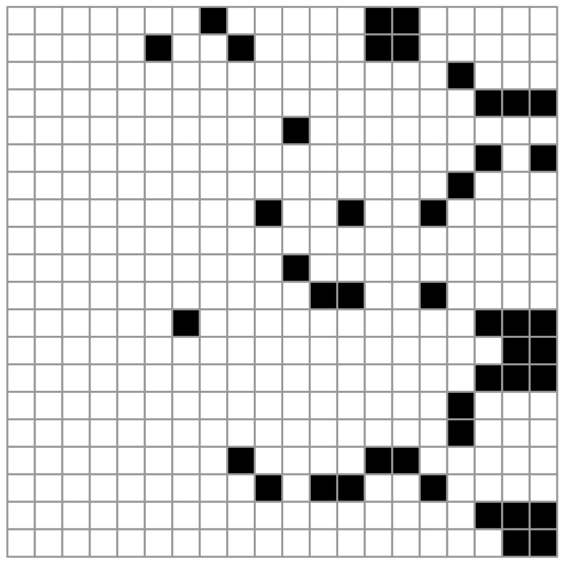

Figure 6: Adjacency matrix of a niche model food web with 20 species and connectance of 0.15

Following the dynamics of predator prey interactions in this example, of the initial 20 species in the food web 4 have gone extinct.

3.3 Food Web Dynamics Explorer via Shiny Web App

4 Conclusions

5 Future Directions

I can see a number of different directions this package could take in terms of increasing performance and functionality. Below I highlight several lines I plan to take in further developing the rend package.

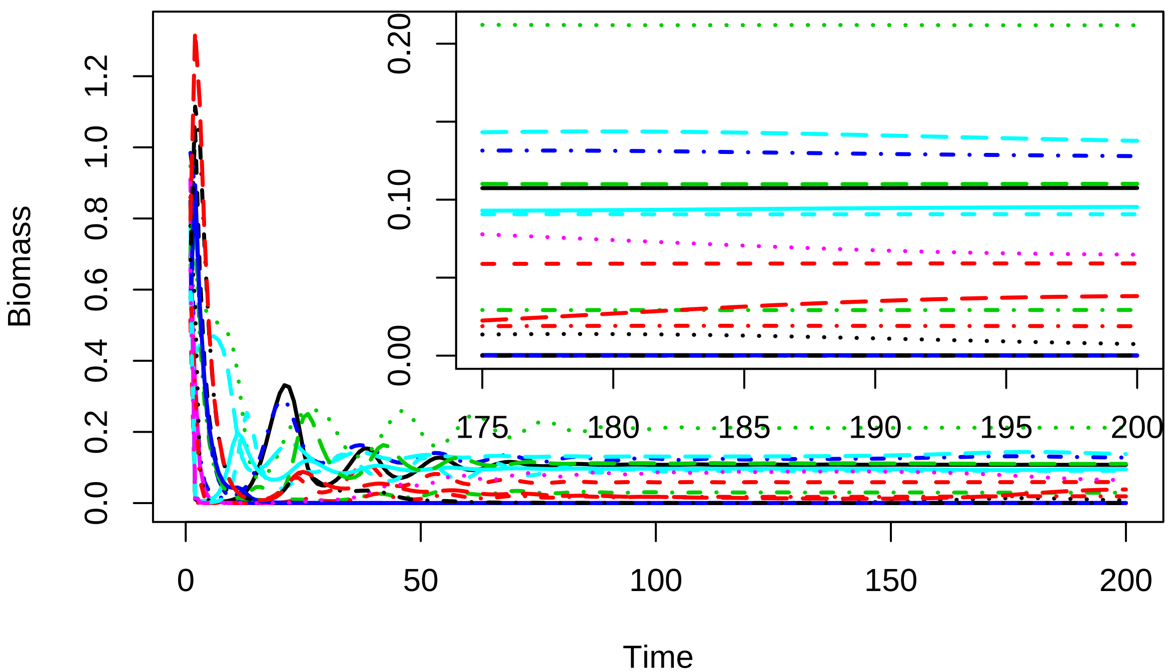

Figure 7: Biomass dynamics of species in a niche model food web, inset shows dynamics of last 25 time steps

5.1 Add internal functions for output assessment

5.1.1 Structural changes over time

5.2 Increasing options for trophic dynamic models

5.2.1 Ratio-dependent functional responses

5.2.2 Non-Bioenergetics models

5.3 Adding models for alternative interaction types

5.3.1 Mutualisms

5.3.2 Competition

5.4 Incorporate Stochasticity

5.4.1 sde package

5.4.2 Risk-based assessment