Food Web Structure and Dynamics

Jonathan J. Borrelli

Monday, November 02, 2015

Contents

1 Introduction

Some intro stuff about food web dynamics, structure and my past work.

1.1 Purpose

To further explore the relationship between food web structure and dynamics.

1.2 Hypothesis

In previous work I have asserted that observed food web structure emerges from the preferential loss of unstable configurations. This was supported by correlational evidence that the basic building blocks of food webs, short food chains and motifs, are also more likely to be stable than other potential building blocks of similar types (e.g., longer food chains or other three species subgraphs).

This project seeks a more mechanistic approach, by simulating the dynamics of model food webs. Based on my prior work I suggest that the structure of the food web following extinction events (from simulated dynamics) will exhibit a higher proportion of stable components via loss of species that contribute to destabilizing building blocks.

2 The Model

I have borrowed a model of multispecies predator prey dynamics based on species biomass modified from Yodzis and Innes (1992), that has been used in a number of publications. Specifically, I am using the form and parameterization suggested by Romanuk et al. (2009) and Williams and Martinez (2004).

$$\frac{dB_i(t)}{dt} = G_i(B) - x_i B_i(t) + \sum_j^n (x_i y_{ij} F_{ij}(B) B_i(t) - x_j y_{ji} F_{ji}(B) B_j(t)/e_{ji})$$

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.

$$G_i(B) = r_i B_i(t)(1 - \frac{B_i(t)}{K_i})$$

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.

This equation is represented in my code as:

```
G.i <- function(r, B, K){return(r * B * (1 - (B/K)))}
```

As in the equation `r` is the intrinsic rate of increase, `B` is biomass, and `K` is carrying capacity.

The consumption of resource j by consumer i is modeled by:

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

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.

The functional response is represented in my code as:

```
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)
}
```

The function takes in a vector of species' biomasses, the adjacency matrix defining species' interactions, the half saturation constant `B.0`, and the tuning parameter (q, here termed `xpar`). The first line of the function generates the summation in the denominator of the functional response, $\sum_k B_k^{1+q}$, and the second line gives the last part of the denominator. In the code, the denominator is a vector of length N (number of species), were each element is the denominator of the corresponding species. The third line generates the numerator by assigning the biomass of the prey species to each interaction and raising it to `1+q` and at the same time completing the function by dividing each row of the matrix by the corresponding denominator.

2.1 Functional Response

To test how the tuning parameter changes the shape of the functional response, I wrote a simple simulation to test how the biomass of prey eaten changes with the biomass of the prey species in a two species predator prey system defined by 0, 0, 1, 0.

```
K = 1
x.i = .5
yij = 6
xpar = 0
B.o = .5
A =  matrix(c(0,0,1,0), nrow = 2)
FR = Fij

prey <- seq(0, 2, .01)
pred <- .5
x <- seq(0, 5, .2)
eaten <- matrix(nrow = length(prey), ncol = length(x))
for(j in 1:length(x)){
  for(i in 1:length(prey)){
    states = c(prey[i], pred)
    eaten[i,j] <- rowSums((x.i * yij * FR(states, A, B.o, xpar = x[j]) * states))[2]
  }
}
```

I ran the simulation using the same fixed parameter values I used in the full model based on Romanuk (2009), listed in Table 1. I tested a sequence of potential `q` values from `0` (Type I) to `5` (Type III with large prey refuge). The results are in Figure 1.

```
matplot(prey, eaten, typ = "l", xlab = "Prey Biomass", ylab = "Prey Biomass Consumed")
```

In this model, as I have understood it 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 $F_j i = t(F_i j)$.

2.2 Full Model

The full model is implemented in R using a differential equation solver, `ode` in the R package `deSolve`. The function input to the solver for numerical integration is:

```
conres <- 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 =

    list(c(dB))
  })

}
```

The function takes in a series of times (`t`), intitial states (biomasses in this case, `states`), and a list of parameter values. To input into the solver, I have written a wrapper function that allows me to modify the food web structure, specific parameters and functions, and create plots.

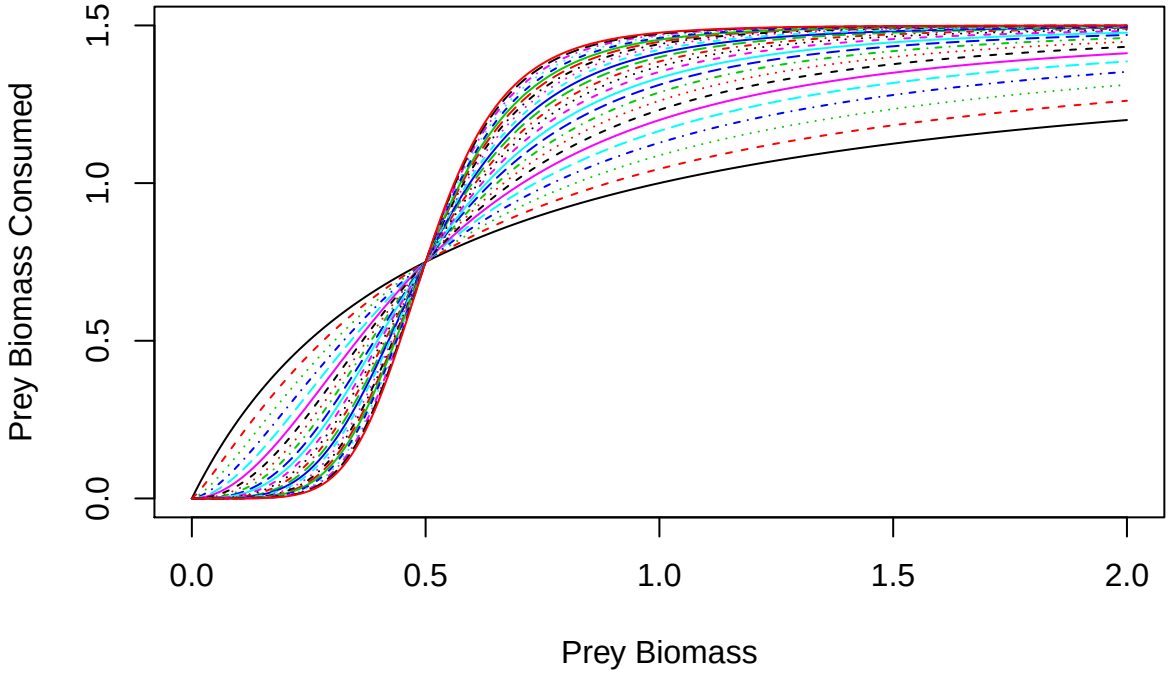

Figure 1: How the functional response of the prey changes with altered values of the parameter q

```
Crmod <- function(Adj, t = 1:200, G = G.i, method = conres, FuncRes = Fij, K = 1, x.i = .5, yij = 6, ei
  require(deSolve)

  grow <- get.r(Adj)

  par <- list(
    K = K,
    x.i = x.i,
    yij = yij,
    eij = 1,
    xpar = xpar,
    B.o = B.o,
    r.i = grow,
    A = Adj,
    G.i = G,
    FR = FuncRes
  )

  states <- runif(nrow(Adj), .5, 1)

  out <- ode(y=states, times=t, func=method, parms=par, events = list(func = eventfun, time = t))

  if(plot) print(matplot(out[,-1], typ = "l", lwd = 2))

  return(out)
}
```

2.3 Parameter Values

Work by Williams and Martinez (2004) and Romanuk et al (2009) suggest that fixed values for a number of the required model parameters give similar dynamic results to drawing these parameter values from distributions with specified means and standard deviations. The suggested values are specified in the function above, and in Table 1 below.

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

Table 1: Fixed parameter values for the dynamic model

3 Analysis

I want to know how population dynamics based extinctions alters the structure of the food web. To do that I need to be able to compare the initial and final food webs.

3.1 Functions

The `web_props` function below takes the adjacency matrix and computes several network properties: the number of species (`N`), the connectance (`C`, $\frac{L}{N*(N-1)}$), number of links (`Ltot`), link density (`LD`), clustering coefficient (`clust`), modularity (`mod`), average path length (average food chain length, `apl`), diameter (longest shortest food chain, `diam`), number of basal species (`bas`), and number of top predators (`top`).

```
web_props <- function(mat){
  require(NetIndices)
  require(modMax)

  N <- nrow(mat)
  C <- sum(mat)/(nrow(mat)*(nrow(mat)-1))

  genind <- GenInd(mat)
  Ltot <- genind$Ltot
  LD <- genind$LD

  g <- graph.adjacency(mat)

  clust <- transitivity(g)
  apl <- average.path.length(g)
  diam <- diameter(g)

  bas <- sum(degree(g, mode = "in") == 0)
  top <- sum(degree(g, mode = "out") == 0)

  mod <- simulatedAnnealing(mat, fixed = 1000)

  df <- data.frame(N, C, Ltot, LD, clust, mod = mod$modularity, apl, diam, bas, top, mot)
  return(df)
}
```

I also want to know how the motif structure of the network changes after extinctions. The `motif_counter` function (same that I used in Borrelli (2015)), gives the number of each of the three species subgraphs in the network.

```
motif_counter <- function(graph.lists){
  require(igraph)

  if(!is.list(graph.lists)){
    stop("The input should be a list of graph objects")
  }

  triad.count <- lapply(graph.lists, triad.census)
  triad.matrix <- matrix(unlist(triad.count), nrow = length(graph.lists), ncol = 16, byrow = T)
  colnames(triad.matrix) <- c("empty", "single", "mutual", "s5", "s4", "s1", "d4",
```

```
                              "d3", "s2", "s3","d8", "d2", "d1", "d5", "d7", "d6")

  triad.df <- as.data.frame(triad.matrix)

  motif.data.frame <- data.frame(s1 = triad.df$s1, s2 = triad.df$s2, s3 = triad.df$s3, s4 = triad.df$s4
                                 s5 = triad.df$s5, d1 = triad.df$d1, d2 = triad.df$d2, d3 = triad.df$d3
                                 d5 = triad.df$d5, d6 = triad.df$d6, d7 = triad.df$d7, d8 = triad.df$d8

  return(motif.data.frame)
}
```

Because it is not worthwhile to compare the counts of motifs in initial and final networks (the difference in number of species and connectance precludes direct comparison), I wrote a function to compare the motif count in the final network to networks of the same size created by assuming random extinctions. For example, if the final network has 30 species, and the initial food web had 100, I create n random adjacency matrices with 30 of the initial 100 species chosen at random.

```
rel.rand <- function(mat, n, iter = 10000){
  require(igraph)
  rands <- list()
  for(i in 1:iter){
    spp <- sample(1:nrow(mat), n)
    rands[[i]] <- mat[spp, spp]
  }
  rands.g <- lapply(rands, graph.adjacency)
  mot <- motif_counter(rands.g)
  return(mot)
}

rel.rand.z <- function(mat, dyn, iter = 10000){
  m1 <- motif_counter(list(graph.adjacency(mat[which(tail(dyn, 1)[-1] > 0),which(tail(dyn, 1)[-1] > 0)]
  n <- sum(tail(dyn, 1)[-1] > 0)
  m2 <- rel.rand(mat = mat, n = n, iter = iter)

  zscore <- (m1 - colMeans(m2))/apply(m2, 2, sd)

  return(zscore)
}
```