Appendix A

Contents

Data

Data Sources

Three food webs were downloaded from the Dryad Digital Repository (Roopnarine & Hertog 2012a, 2012b). Another seven were available from Ecological Archives (Hechinger et al. 2011; Mouritsen et al. 2011; Thieltges et al. 2011; Zander et al. 2011; Preston et al. 2012). Fourteen webs were provided by Jennifer Dunne of the PEaCE Lab (Baird & Ulanowicz 1989; Warren 1989; Polis 1991; Hall & Raffaelli 1991; Martinez 1991; Christensen & Pauly 1992; Havens 1992; Goldwasser & Roughgarden 1993; Opitz 1996; Waide & Reagan 1996; Yodzis 1998, 2000; Christian & Luczkovich 1999; Martinez et al. 1999; Memmott et al. 2000; Link 2002) that were analyzed in (Dunne et al. 2002, 2004). The remaining 26 food webs were downloaded from the Interaction Web Database (Jaarsma et al. 1998; Townsend et al. 1998; Thompson & Townsend 1999, 2000, 2003, 2005; Thompson & Edwards 2001). We also used a subset of sixteen ecosystem networks provided by Robert Ulanowicz through his website, and used most recently in Ulanowicz et al. 2014.

References

Baird, D. and Ulanowicz, R. (1989). The seasonal dynamics of the Chesapeake Bay ecosystem. Ecol. Monogr., 59, 329:364.

Christensen, V. and Pauly, D. (1992). ECOPATH II: a software for balancing steady-state ecosystem models and calculating network characteristics. Ecol. Modell., 61, 169: 185.

Christian, R.R. and Luczkovich, J.J. (1999). Organizing and understanding a winter's seagrass foodweb network through effective trophic levels. Ecol. Modell., 117, 99:124.

Dunne, J., Williams, R. and Martinez, N. (2004). Network structure and robustness of marine food webs. Mar. Ecol. Prog. Ser., 273, 291:302.

Dunne, J.A., Williams, R.J. and Martinez, N.D. (2002). Network structure and biodiversity loss in food webs: robustness increases with connectance. Ecol. Lett., 5, 558:567.

Goldwasser, L. and Roughgarden, J. (1993). Construction and analysis of a large Carribean food web. Ecology, 74, 1216:1233.

Hall, S. and Raffaelli, D. (1991). Food:web patterns: lessons from a species:rich web. J. Anim. Ecol., 60, 823:841.

Havens, K. (1992). Scale and structure in natural food webs. Science., 257, 1107:1109.

Hechinger, R.F., Lafferty, K.D., McLaughlin, J.P., Fredensborg, B.L., Huspeni, T.C., Lorda, J., et al. (2011). Food webs including parasites, biomass, body sizes, and life stages for three California/Baja California estuaries. Ecology, 92, 791.

Jaarsma, N.G., de Boer, S.M., Townsend, C.R., Thompson, R.M. and Edwards, E.D. (1998). Characterising food-webs in two New Zealand streams. New Zeal. J. Mar. Freshw. Res., 32, 271:286.

Link, J. (2002). Does food web theory work for marine ecosystems? Mar. Ecol. Prog. Ser., 230, 1:9.

Martinez, N. (1991). Artifacts or Attributes? Effects of Resolution on the Little Rock Lake Food Web. Ecol. Monogr., 61, 367:392.

Martinez, N., Hawkins, B., Dawah, H. and Feifarek, B. (1999). Effects of sampling effort on characterization of food:web structure. Ecology, 80, 1044:1055.

Memmott, J., Martinez, N.D. and Cohen, J.E. (2000). Predators, parasitoids and pathogens: species richness, trophic generality and body sizes in a natural food web. J. Anim. Ecol., 69, 1:15.

Mouritsen, K.N., Poulin, R., McLaughlin, J.P. and Thieltges, D.W. (2011). Food web including metazoan parasites for an intertidal ecosystem in New Zealand. Ecology, 92, 2006.

Opitz, S. (1996). Trophic interactions in Caribbean coral reefs. ICALRM Tech, Makati City, Philippines.

Polis, G. (1991). Complex trophic interactions in deserts: an empirical critique of food:web theory. Am. Nat., 138, 123:155.

Preston, D.L., Orlofske, S.A., McLaughlin, J.P. and Johnson, P.T. (2012). Food web including infectious agents for a California freshwater pond. Ecology, 93, 1760.

Roopnarine, P.D. and Hertog, R. (2012a). Data from: Detailed food web networks of three Greater Antillean coral reef systems: the Cayman Islands, Cuba, and Jamaica. Dryad Digital Repository. doi: 10.5061/dryad.c213h

Roopnarine, P.D. and Hertog, R. (2012b). Detailed food web networks of three Greater Antillean coral reef systems: the Cayman Islands, Cuba, and Jamaica. Dataset Pap. Ecol., 2013, 857470. doi:10.7167/2013/857470

Thieltges, D.W., Reise, K., Mouritsen, K.N., McLaughlin, J.P. and Poulin, R. (2011). Food web including metazoan parasites for a tidal basin in Germany and Denmark. Ecology, 92, 2005.

Thompson, R. and Edwards, E. (2001). Allocation of effort in stream food:web studies: the best compromise? Mar. Freshw. Res., 52, 339:345.

Thompson, R. and Townsend, C. (1999). The effect of seasonal variation on the community structure and food:web attributes of two streams: implications for food:web science. Oikos, 87, 75:88.

Thompson, R. and Townsend, C. (2003). Impacts on stream food webs of native and exotic forest: an intercontinental comparison. Ecology, 84, 145:161.

Thompson, R. and Townsend, C. (2005). Energy availability, spatial heterogeneity and ecosystem size predict food:web structure in streams. Oikos, 108, 137:148.

Thompson, R.M. and Townsend, C.R. (2000). Is resolution the solution?: the effect of taxonomic resolution on the calculated properties of three stream food webs. Freshw. Biol., 44, 413:422.

Townsend, C.R., Thompson, R.M., Mcintosh, A.R., Kilroy, C., Edwards, E. and Scarsbrook, M.R. (1998). Disturbance, resource supply, and food:web architecture in streams. Ecol. Lett., 1, 200:209.

Ulanowicz, R.E., Holt, R.D., Barfield, M., 2013. Limits on ecosystem trophic complexity: insights from ecological network analysis. Ecol. Lett. 17, 127:136. doi:10.1111/ele.12216

Waide, R. and Reagan, W. (1996). The food web of a tropical rainforest. Univ. Chicago. University of Chicago Press, Chicago, Illinois, U.S.A.

Warren, P. (1989). Spatial and temporal variation in the structure of a freshwater food web. Oikos, 55, 299:311.

Yodzis, P. (1998). Local trophodynamics and the interaction of marine mammals and fisheries in the Benguela ecosystem. J. Anim. Ecol., 67, 635:658.

Yodzis, P. (2000). Diffuse Effects in Food Webs. Ecology, 81, 261:266.

Zander, C.D., Josten, N., Detloff, K.C., Poulin, R., McLaughlin, J.P. and Thieltges, D.W. (2011). Food web including metazoan parasites for a brackish shallow water ecosystem in Germany and Denmark. Ecology, 92, 2007.

Interaction strength

Required libraries

```
library(igraph)
library(NetIndices)
library(data.table)
library(ggplot2)
library(reshape2)
```

Import the Ulanowicz data

```
setwd(ulanpath)

ulanEDGE <- list()
for(i in 1:length(list.files())){
  ulanEDGE[[i]] <- read.csv(list.files()[i])[,-1]
}
```

Determine the threshold flow weight above which we call "strong" interactions. Here we subset the Ulanowicz webs by taking just the top 50%, 40%, 30%, 20%, and 10%.

```
q50 <- sapply(ulanEDGE, function(x){quantile(x[,3], .5)})
q60 <- sapply(ulanEDGE, function(x){quantile(x[,3], .6)})
q70 <- sapply(ulanEDGE, function(x){quantile(x[,3], .7)})
q80 <- sapply(ulanEDGE, function(x){quantile(x[,3], .8)})
q90 <- sapply(ulanEDGE, function(x){quantile(x[,3], .9)})

uEDGE.50 <- list()
uEDGE.60 <- list()
uEDGE.70 <- list()
uEDGE.80 <- list()
uEDGE.90 <- list()
for(i in 1:length(ulanEDGE)){
  uEDGE.50[[i]] <- ulanEDGE[[i]][which(ulanEDGE[[i]][,3] >= q50[i]),]
  uEDGE.60[[i]] <- ulanEDGE[[i]][which(ulanEDGE[[i]][,3] >= q60[i]),]
  uEDGE.70[[i]] <- ulanEDGE[[i]][which(ulanEDGE[[i]][,3] >= q70[i]),]
  uEDGE.80[[i]] <- ulanEDGE[[i]][which(ulanEDGE[[i]][,3] >= q80[i]),]
```

```
  uEDGE.90[[i]] <- ulanEDGE[[i]][which(ulanEDGE[[i]][,3] >= q90[i]),]
}

uGRAPH <- lapply(uEDGE.50, function(x){graph.edgelist(as.matrix(x[,1:2]))})
uGRAPH6 <- lapply(uEDGE.60, function(x){graph.edgelist(as.matrix(x[,1:2]))})
uGRAPH7 <- lapply(uEDGE.70, function(x){graph.edgelist(as.matrix(x[,1:2]))})
uGRAPH8 <- lapply(uEDGE.80, function(x){graph.edgelist(as.matrix(x[,1:2]))})
uGRAPH9 <- lapply(uEDGE.90, function(x){graph.edgelist(as.matrix(x[,1:2]))})

uMAT <- lapply(uGRAPH, get.adjacency, sparse = F)
uMAT6 <- lapply(uGRAPH6, get.adjacency, sparse = F)
uMAT7 <- lapply(uGRAPH7, get.adjacency, sparse = F)
uMAT8 <- lapply(uGRAPH8, get.adjacency, sparse = F)
uMAT9 <- lapply(uGRAPH9, get.adjacency, sparse = F)
```

```
tind <- lapply(uMAT, TrophInd)
tind6 <- lapply(uMAT6, TrophInd)
tind7 <- lapply(uMAT7, TrophInd)
tind8 <- lapply(uMAT8, TrophInd)
tind9 <- lapply(uMAT9, TrophInd)

prop.omn <- sapply(tind, function(x){(1 - sum(x$OI == 0)/nrow(x)) * 100})
prop.omn6 <- sapply(tind6, function(x){(1 - sum(x$OI == 0)/nrow(x)) * 100})
prop.omn7 <- sapply(tind7, function(x){(1 - sum(x$OI == 0)/nrow(x)) * 100})
prop.omn8 <- sapply(tind8, function(x){(1 - sum(x$OI == 0)/nrow(x)) * 100})
prop.omn9 <- sapply(tind9, function(x){(1 - sum(x$OI == 0)/nrow(x)) * 100})

p.omn <- matrix(c(prop.omn, prop.omn6, prop.omn7, prop.omn8, prop.omn9), ncol = 5)
colnames(p.omn) <- seq(50, 90, 10)
omn.dat <- data.frame(prop = seq(50, 90, 10), omn = colMeans(p.omn),
                      upper = apply(p.omn, 2,
                                    function(x){mean(x) + 1.96 * (sd(x)/sqrt(length(x)))}),
                      lower = apply(p.omn, 2,
                                    function(x){mean(x) - 1.96 * (sd(x)/sqrt(length(x)))}))
```

```
ggplot(omn.dat, aes(x = prop, y = omn)) + geom_errorbar(aes(x = prop, ymax = upper, ymin = lower), widt
```

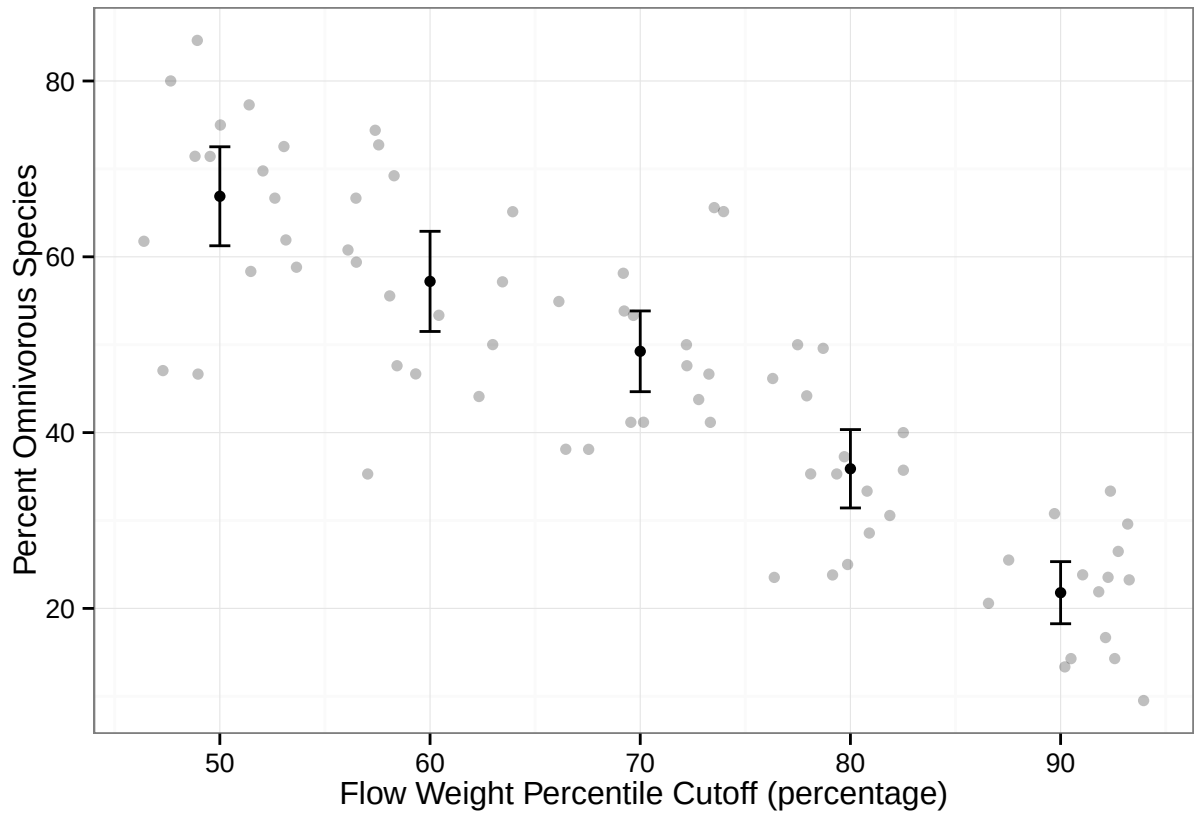

Figure A1 The proportion of omnivorous species in 15 ecosystem flow networks that have been subset by flow weight. Values along the x-axis represent removal of the bottom 50, 60, 70, 80, and 90 percentiles of flow weights. Data for making this figure is available from http://www.cbl.umces.edu/~ulan

Note that there is still at least 20% omnivory even when only looking at the top 10% of interactions. We say "at least" because this simple analysis does not remove from consideration species who are disconnected from the web (i.e. they do not have any interactions whose flow is above the threshold).

Code

All code here can also be found on GitHub

Loading required packages

```
library(RCurl)
library(igraph)
library(reshape2)
library(ggplot2)
library(grid)
library(data.table)
library(devtools)
library(NetIndices)
```

Functions

The function `analyze.eigen` randomly fills a signed matrix and calculates the eigenvalue with the largest real part. This function takes in a sign matrix (a matrix of +1s, -1s, and 0s) and replaces the a_{ij} with values randomly drawn from predefined uniform distributions. This is the function used for the simple food webs.

```
analyze.eigen<-function(m){
  for(i in 1:nrow(m)){
    for (j in 1:nrow(m)){
      ifelse(m[i,j]==1,m[i,j]<-runif(1,0,10),NA)
      ifelse(m[i,j]==-1,m[i,j]<-runif(1,-1,0),NA)
    }
  }
  for(i in 1:nrow(m)){
    if(m[i,i]<0){m[i,i]<--1}
  }
  ev<-eigen(m)$values[1]
  return(ev)
}
```

The following functions are used for the random and niche model webs.

The `ran.unif` function fills in a sign structured matrix with random values drawn from a random uniform distribution. The impact of the prey on the predator population is drawn from a distribution between 0 and `pred`, while the impact of the predator on the prey is distributed between `prey` and 0.

```
ran.unif <- function(motmat, pred = 10, prey = -1, random = F){
  newmat <- apply(motmat, c(1,2), function(x){
    if(x==1){runif(1, 0, pred)}else if(x==-1){runif(1, prey, 0)} else{0}
  })
  if(random){
    diag(newmat) <- runif(length(diag(newmat)), -1, 0)
  }else{diag(newmat) <- -1}

  return(newmat)
}
```

The `maxRE` computes the largest eigenvalue and returns the real part.

```
maxRE <- function(rmat){
  lam.max <- eigen(rmat)$values[which.max(Re(eigen(rmat)$values))]
  return(lam.max)
}
```

The `eig.analysis` function takes an imput of a list of matrices and randomly fills it according to `ran.unif` (above) `n` times and computes `maxRE` for each iteration.

```
eig.analysis <- function(n, matrices, params){
  require(data.table)
  dims <- dim(matrices[[1]])
  cols <- length(matrices)
  rows <- n
  eigenMATRIX.re <- matrix(nrow = rows, ncol = cols)
```

```
  eigenMATRIX.im <- matrix(nrow = rows, ncol = cols)
  samps <- list()
  for(i in 1:n){
    ranmat <- lapply(matrices, ran.unif, pred = params[,1],
                     prey = params[,2], random = T)
    sampvals <- matrix(nrow = length(ranmat), ncol = dims[1]^2)
    for(j in 1:length(ranmat)){
      sampvals[j,] <- ranmat[[j]]
    }
    eigs <- sapply(ranmat, maxRE)
    eigenMATRIX.re[i,] <- Re(eigs)
    eigenMATRIX.im[i,] <- Im(eigs)
    samps[[i]] <- as.data.frame(sampvals)
  }
  svals <- cbind(web = rep(1:length(matrices), n), n = rep(1:n, each = cols), rbindlist(samps))
  return(list(samples = svals, ematrix.re = eigenMATRIX.re, ematrix.im = eigenMATRIX.im))
}
```

The `conversion` function takes in an adjacency matrix and converts it into sign matrix, assuming all interactions are predator/prey (+/-).

```
conversion <- function(tm){
  for(i in 1:nrow(tm)){
    for(j in 1:ncol(tm)){
      if(tm[i,j] == 1){tm[j,i] <- -1}
    }
  }
  return(tm)
}
```

The `randomWEBS` function generates `numweb` random webs of `S` species and `total` interactions. The connectance of these webs would then be `total/S^2`. The `chain` parameter can be modified to change the maximum chain length in the web. For example, the default `chain = 9` means that the webs have at least one chain of ten species.

```
randomWEBS <- function(S = 10, numweb = 200, chain = 9, total = 14){
  require(NetIndices)
  require(igraph)
  mywebs <- list()
  for(j in 1:numweb){

    check <- 1
    while(!check == 0){
      myweb <- matrix(0, nrow = S, ncol = S)
      for(i in 1:chain){
        myweb[i,i+1] <- 1
      }
      tophalf <- which(myweb[upper.tri(myweb)] == 0)
      newones <- sample(tophalf, total-chain)
      myweb[upper.tri(myweb)][newones] <- 1
      mywebs[[j]] <- myweb

      indeg <- apply(myweb, 1, sum)
```

```
      outdeg <- apply(myweb, 2, sum)
      deg <- indeg + outdeg

      if(sum(deg == 0) >= 1){check <- 1}else{check <- 0}

    }

  }
  return(mywebs)
}
```

The two functions below are the niche model (`niche.model`), and a function that uses the niche model code to generate a list of niche model food webs with a given connectance `C` and and number of species `S` (`niche_maker`).

```
niche.model<-function(S,C){
  require(igraph)
  connected = FALSE
  while(!connected){
    new.mat<-matrix(0,nrow=S,ncol=S)
    ci<-vector()
    niche<-runif(S,0,1)
    r<-rbeta(S,1,((1/(2*C))-1))*niche

    for(i in 1:S){
      ci[i]<-runif(1,r[i]/2,niche[i])
    }

    r[which(niche==min(niche))]<-.00000001

    for(i in 1:S){

      for(j in 1:S){
        if(niche[j]>(ci[i]-(.5*r[i])) && niche[j]<(ci[i]+.5*r[i])){
          new.mat[j,i]<-1
        }
      }
    }

    new.mat<-new.mat[,order(apply(new.mat,2,sum))]

    connected <- is.connected(graph.adjacency(new.mat))
  }
  return(new.mat)
}

niche_maker <- function(n, S, C){
  niche.list <- list()
  for (i in 1:n){
    niche.list[[i]]<- niche.model(S, C)
  }
  return(niche.list)
}
```

The `randomQSS` function takes in a list of webs (as adjacency matrices) with the `mywebs` paramater, and a 1 row by 2 column matrix of parameters to feed into the `eig.analysis` function. It then outputs a list of two data frames. The first, `web.dat` contains information on quasi sign-stability, max, mean and median trophic level, standard deviation of trophic level, and the diameter (longest food chain) of the web. The second data frame, `iter.dat` contains information on each random sampling of each matrix. `iter.dat` has each sampled value for each link, and the real and imaginary parts of the largest eigenvalue.

```
randomQSS <- function(mywebs, params){
  require(NetIndices)
  require(igraph)

  mywebs1 <- lapply(mywebs, conversion)
  myweb.tl <- lapply(mywebs, TrophInd)
  emat <- eig.analysis(1000, mywebs1, params)

  qss <- apply(emat$ematrix.re, 2, function(x){sum(x<0)/1000})
  maxtl <- sapply(myweb.tl, function(x){max(x$TL)})
  meantl <- sapply(myweb.tl, function(x){mean(x$TL)})
  medtl <- sapply(myweb.tl, function(x){median(x$TL)})
  sdtl <- sapply(myweb.tl, function(x){sd(x$TL)})
  diam <- sapply(lapply(mywebs, graph.adjacency), diameter)

  web.dat <- data.frame(qss, diam, maxtl, meantl, medtl, sdtl)
  iter.dat <- cbind(par = rep(paste(params, collapse = "_"), nrow(emat$samples)),
                    emat$samples, reals = as.vector(emat$ematrix.re),
                    im = as.vector(emat$ematrix.im))

  return(list(web.dat, iter.dat))
}
```

The `getQSS` function is the main function for the simulation, putting together all previous functions. The `webiter`, `maxchain`, and `totalINT` paramaters are fed into the functions that generate the food webs. The `params` parameter should be a matrix with two columns, one for the impact of the prey on the predator, and one for the impact of the predator on the prey. This function is designed to run in parallel, and will run a set of webs on each core. The output is written to file according to `filepath`. This function will generate either random webs or niche model webs according to whether `niche` is TRUE or FALSE. It calls a separate script `robustnessFUNC.R` that holds the functions described above and allows each node of the cluster to complete the analysis.

```
getQSS <- function(webiter = 100, maxchain = 9, totalINT = 14, params, filepath, niche = FALSE){
  require(doSNOW)
  require(parallel)
  require(data.table)

  #make the cluster
  cl <- makeCluster(detectCores()-1)
  registerDoSNOW(cl)

  RESULT <- foreach(i = 1:maxchain) %dopar% {
    source("C:/Users/jjborrelli/Desktop/GitHub/Food-Chain-Length/robustnessFUNC.R")

    #cat(i, "\n")
    if(!niche){
      rwebs <- randomWEBS(S = 10, numweb = webiter, chain = i, total = totalINT)
```

```
    }else{
      rwebs <- niche_maker(15, 10, totalINT)
    }

    rqss <- list()
    for(j in 1:nrow(params)){
      rqss[[j]] <- randomQSS(mywebs = rwebs, params = params[j,1:2])

      rqss[[j]][[1]] <- cbind(C = rep(totalINT, nrow(rqss[[j]][[1]])),
                                      mxch = rep(i, nrow(rqss[[j]][[1]])),
                              rqss[[j]][[1]],
                              par = rep(paste(params[j,], collapse = "/"),
                                        nrow(rqss[[j]][[1]])))

      rqss[[j]][[2]] <- cbind(C = rep(totalINT, nrow(rqss[[j]][[2]])),
                              mxch = rep(i, nrow(rqss[[j]][[2]])),
                              rqss[[j]][[2]],
                              par = rep(paste(params[j,], collapse = "/"),
                                        nrow(rqss[[j]][[2]])))
      cat("--", j, "\n")
    }
    rqss <- unlist(rqss, recursive = F)
    web.dat.ls <- rbindlist(rqss[seq(1, length(rqss), 2)])
    iter.dat.ls <- rbindlist(rqss[seq(2, length(rqss), 2)])

    return(list(web.dat.ls, iter.dat.ls))

  }

  stopCluster(cl)


  RESULT <- unlist(RESULT, recursive = F)
  chain.data <- rbindlist(RESULT[seq(1, length(RESULT), 2)])
  iter.data <- rbindlist(RESULT[seq(2, length(RESULT), 2)])
  rm(RESULT)
  write.csv(chain.data, file = paste(filepath, "/webdata-",
                                     totalINT, ".csv", sep = ""),
            row.names = F)

  #Code to generate dataframes of sampled matrices (note this can create very large files)
  #write.csv(iter.data, file = paste(filepath, "/iterdata-",
  #                                  totalINT, ".csv", sep = ""),
  #          row.names = F)

  return(chain.data)
}
```

Simulations

Simple webs

The code below will create the sign matrix structure for each perturbed chain of 2, 3, 4, 5, and 6 levels. A -1 indicates the impact of a predator on its prey (negative), while a 1 indicates the impact of the prey on the predator (positive).

```
sign2<-matrix(c(-1,-1,1,0),nrow=2,ncol=2)
diag(sign2)<--1

sign3<-matrix(c(-1,-1,-1,1,0,-1,1,1,0),nrow=3,ncol=3)
diag(sign3)<--1

sign4<-matrix(nrow=4,ncol=4)
sign4[lower.tri(sign4)]<--1
sign4[upper.tri(sign4)]<-1
diag(sign4)<--1

sign5<-matrix(nrow=5,ncol=5)
sign5[lower.tri(sign5)]<--1
sign5[upper.tri(sign5)]<-1
diag(sign5)<--1

sign6<-matrix(nrow=6,ncol=6)
sign6[lower.tri(sign6)]<--1
sign6[upper.tri(sign6)]<-1
diag(sign6)<--1
```

These matrices are combined into a list for simplicity:

```
sign.matrices<-list(sign2,sign3,sign4,sign5,sign6)
names(sign.matrices)<-c("2 sp","3 sp","4 sp","5 sp","6 sp")
sign.matrices
```

The following code applies the `analyze.eigen` function to each of the 5 sign matrices created above. It then stores the `max(Re(lambda))`, or the eigen value with the largest real part in the `eigenvalues` list. Quasi sign-stability (`qss`) can then be calculated by determining the proportion of the `max(Re(lambda))` that are negative out of the 10000 that are calculated.

```
eigenvalues<-list()
qss<-list()
for(i in 1:5){
  eigenvalues[[i]]<-replicate(10000,analyze.eigen(sign.matrices[[i]]))
  qss[[i]]<-sum(Re(eigenvalues[[i]])<0)/10000
}

names(eigenvalues)<-c("2 sp","3 sp","4 sp","5 sp","6 sp")
names(qss)<-c("2 sp","3 sp","4 sp","5 sp", "6 sp")
qss
```

Random and niche model webs

Set the parameters for the simulation, and the desired levels of connectance.

```
pars <- data.frame(pred = c(10, 10, 10, 5, 5, 5, 1, 1, 1), prey = c(-1, -5, -10, -1, -5, -10, -1, -5, -
ints <- c(12, 16, 20, 24, 28)
```

Use the getQSS function to create webs and determine quasi sign-stability for a range of parameters.

For random webs

```
for(con in 1:length(ints)){
  getQSS(25, maxchain =7, totalINT = ints[con], params = pars,
         filepath = myfilepath1, niche = F)
  cat(con/length(ints)*100, "%", "\n")
}
```

For niche model webs

```
for(con in 1:length(ints)){
  getQSS(25, maxchain =7, totalINT = ints[con]/100, params = pars,
         filepath = myfilepath2, niche = T)
  cat(con/length(ints)*100, "%", "\n")
}
```

Figures

Figure 1

```
path <- getURL("https://raw.githubusercontent.com/jjborrelli/Food-Chain-Length/master/Tables/webDiameter
               ssl.verifypeer=0L, followlocation=1L)

web.diameters <- read.csv(text = path, row.names = 1)

diam.plot <- ggplot(web.diameters, aes(x = Diameter + 1))
diam.plot <- diam.plot + geom_histogram(binwidth = 1)
diam.plot <- diam.plot + scale_x_continuous(name = "Longest Chain Length", breaks = 0:9) + ylab("Freque
```

Figure 1a

```
path2 <- getURL("https://raw.githubusercontent.com/jjborrelli/Food-Chain-Length/master/Tables/NodePrope
trophic.properties <- read.csv(text = path2, row.names = 1)

consumers <- which(round(trophic.properties$TL, 6) >= 2)

# ggplot of distribution of trophic positions equal or higher than 2
tc.plot <- ggplot(trophic.properties[consumers,], aes(x = TL)) + theme_bw()
tc.plot <- tc.plot + geom_histogram(binwidth = 1) + xlab("Trophic Position") + ylab("Frequency")
tc.plot <- tc.plot + scale_x_continuous(limits = c(2,6),name = "Trophic Position") #+ scale_y_continuou
tc.plot
```

Figure 1b

Figure 2

The simple web matrices can be visualized with the following code (note: requires the igraph library):

But first the sign matrices need to be converted to graph objects

```
graph.chains<-lapply(sign.matrices, graph.adjacency)
```

The layout is defined for each node of each chain:

```
twospec2<-matrix(c(1,1,
                   2,2),nrow=2,ncol=2,byrow=T)
threespec2<-matrix(c(1,1,
                     3,1,
                     2,2),nrow=3,ncol=2,byrow=T)
fourspec2<-matrix(c(1,1,
                    2,2,
                    0,2,
                    1,3),nrow=4,ncol=2,byrow=T)
fivespec2<-matrix(c(2,1,
                    3,2,
                    1,2,
                    3,3,
                    1,3),nrow=5,ncol=2,byrow=T)
sixspec2<-matrix(c(2,1,
                   3,2,
                   1,2,
                   3,3,
                   1,3,
                   2,4),nrow=6,ncol=2,byrow=T)

layouts<-list(twospec2,threespec2,fourspec2,fivespec2,sixspec2)
```

Setting the plotting options to highlight the longest chain in each web:

```
for(i in 1:5){
  E(graph.chains[[i]])$color = "darkslategray4"
  E(graph.chains[[i]], path = c(1:(i+1)))$color = "black"
}
```

Create the plot

```
par(mfrow=c(5, 1),mar=c(.1, .1, .1, .1))
for(i in 1:5){
  plot.igraph(graph.chains[[i]], layout = layouts[[i]],
              vertex.size = 40,
              vertex.color = "white",
              vertex.label.color = "black",
              vertex.label.cex = .8,
              edge.width = 1,
              edge.arrow.size = .35,
              frame = T,
              margin = 0)
}
```

Figure 3

```
qss.plot <- qplot(2:6, unlist(qss), xlab = "Longest Chain Length", ylab = "Quasi Sign-Stability", margi
qss.plot <- qss.plot + geom_point(size = 2)
qss.plot <- qss.plot + geom_line() + theme_bw()
qss.plot
```

Figure 4

First import the data from the random webs

```
web.files <- list.files(path = filepath1, pattern = "web")
temp.ls <- list()
for(i in 1:length(web.files)){
  temp.ls[[i]]<- fread(paste(filepath1,
                             web.files[i], sep = ""))
}
webdata <- rbindlist(temp.ls)
webdata$C <- factor(webdata$C)

# Formulas for upper and lower confidence limits
sem.l <- function(x){mean(x) - 1.96*sqrt(var(x)/length(x))}
sem.u <- function(x){mean(x) + 1.96*sqrt(var(x)/length(x))}

sub1 <- subset(webdata, par == "1/-10" | par == "10/-1" | par == "1/-5" | par == "5/-1")
sub2 <- subset(webdata, par == "5/-10" | par == "10/-5")
sub3 <- subset(webdata, par == "1/-1" | par == "10/-10" | par == "5/-5")
```

Plots of the different subsets of data. The final plot is what appears in the paper.

```
ggplot(sub1, aes(x = factor(diam+1), y = qss)) +
  geom_point(alpha = .25, position = position_jitter(w=0.2), col = "grey58") +
  stat_summary(fun.y="mean", geom="point") +
  stat_summary(fun.ymin = sem.l, fun.y = "mean", fun.ymax = sem.u,
               geom="errorbar", width = .2) +
  facet_grid(par~C) + theme_bw() +
  xlab("Longest Food Chain Length") + ylab("Quasi sign-stability")


ggplot(sub2, aes(x = factor(diam+1), y = qss)) +
  geom_point(alpha = .25, position = position_jitter(w=0.2), col = "grey58") +
  stat_summary(fun.y="mean", geom="point") +
  stat_summary(fun.ymin = sem.l, fun.y = "mean", fun.ymax = sem.u,
               geom="errorbar", width = .2) +
  facet_grid(par~C) + theme_bw() +
  xlab("Longest Food Chain Length") + ylab("Quasi sign-stability")


ggplot(sub3, aes(x = factor(diam+1), y = qss)) +
  geom_point(alpha = .25, position = position_jitter(w=0.2), col = "grey58") +
  stat_summary(fun.y="mean", geom="point") +
```

```
  stat_summary(fun.ymin = sem.l, fun.y = "mean", fun.ymax = sem.u,
               geom="errorbar", width = .2) +
  facet_grid(par~C) + theme_bw() +
  xlab("Longest Food Chain Length") + ylab("Quasi sign-stability")


ggplot(webdata, aes(x = factor(diam+1), y = qss)) +
  geom_point(alpha = .25, position = position_jitter(w=0.2), col = "grey58") +
  stat_summary(fun.y="mean", geom="point") +
  stat_summary(fun.ymin = sem.l, fun.y = "mean", fun.ymax = sem.u,
               geom="errorbar", width = .2) +
  facet_grid(par~C) + theme_bw() +
  xlab("Longest Food Chain Length") + ylab("Quasi sign-stability")
```

Figure 4

First import the data from the niche model webs

```
web.files <- list.files(path = filepath2, pattern = "web")
temp.ls <- list()
for(i in 1:length(web.files)){
  temp.ls[[i]]<- fread(paste(filepath2,
                             web.files[i], sep = ""))
}
webdata <- rbindlist(temp.ls)
webdata$C <- factor(webdata$C)

# Formulas for upper and lower confidence limits
sem.l <- function(x){mean(x) - 1.96*sqrt(var(x)/length(x))}
sem.u <- function(x){mean(x) + 1.96*sqrt(var(x)/length(x))}

sub1 <- subset(webdata, par == "1/-10" | par == "10/-1" | par == "1/-5" | par == "5/-1")
sub2 <- subset(webdata, par == "5/-10" | par == "10/-5")
sub3 <- subset(webdata, par == "1/-1" | par == "10/-10" | par == "5/-5")
```

Plots of the different subsets of data. The final plot is what appears in the paper.

```
ggplot(sub1, aes(x = factor(diam+1), y = qss)) +
  geom_point(alpha = .25, position = position_jitter(w=0.2), col = "grey58") +
  stat_summary(fun.y="mean", geom="point") +
  stat_summary(fun.ymin = sem.l, fun.y = "mean", fun.ymax = sem.u,
               geom="errorbar", width = .2) +
  facet_grid(par~C) + theme_bw() +
  xlab("Longest Food Chain Length") + ylab("Quasi sign-stability")


ggplot(sub2, aes(x = factor(diam+1), y = qss)) +
  geom_point(alpha = .25, position = position_jitter(w=0.2), col = "grey58") +
  stat_summary(fun.y="mean", geom="point") +
  stat_summary(fun.ymin = sem.l, fun.y = "mean", fun.ymax = sem.u,
               geom="errorbar", width = .2) +
  facet_grid(par~C) + theme_bw() +
```

```
  xlab("Longest Food Chain Length") + ylab("Quasi sign-stability")


ggplot(sub3, aes(x = factor(diam+1), y = qss)) +
  geom_point(alpha = .25, position = position_jitter(w=0.2), col = "grey58") +
  stat_summary(fun.y="mean", geom="point") +
  stat_summary(fun.ymin = sem.l, fun.y = "mean", fun.ymax = sem.u,
               geom="errorbar", width = .2) +
  facet_grid(par~C) + theme_bw() +
  xlab("Longest Food Chain Length") + ylab("Quasi sign-stability")


ggplot(webdata, aes(x = factor(diam+1), y = qss)) +
  geom_point(alpha = .25, position = position_jitter(w=0.2), col = "grey58") +
  stat_summary(fun.y="mean", geom="point") +
  stat_summary(fun.ymin = sem.l, fun.y = "mean", fun.ymax = sem.u,
               geom="errorbar", width = .2) +
  facet_grid(par~C) + theme_bw() +
  xlab("Longest Food Chain Length") + ylab("Quasi sign-stability")
```