Stable Community With Mullet

Richard Southwell

Mon Feb 12 21:15:45 2018

```
library(progress)
library(mizer)
```

We use the analytic solution to our trait based model, with variable egg size, to set up a community thats aggregation produces a background power law size spectrum that is self stablizing. The stablity of this fixed point is promoted by our inclusion of density dependence. We then replace the last species with mullet, and investigate stability with/without fishing.

Set grid points and characteristic sizes

```
dist_sp <- 0.2
minimum_egg <- -4
maximum_egg <- -2
no_sp <- (maximum_egg-minimum_egg)/dist_sp + 1
species <- 1:no_sp
x_min <- seq(minimum_egg, by = dist_sp, length.out = no_sp)

w_min <- 10^x_min
w_inf <- 10^(x_min+5)
w_mat <- 10^(x_min+4.4)  # This is about a quarter of w_inf
min_w <- min(w_min)
max_w <- max(w_inf)
no_w <- log10(max_w/min_w)*100+1
min_w_pp <- 1e-8 # make sure this is low enough for there to be food for min egg.
```

```
# in m^3
NS_vol <- 5.4e+13
# 2100 km^2 = 2.1*10^9 m^2,
CS_area <- 2.1*10^9
CS_depth <- 100
CS_vol <- CS_area*CS_depth
vol_scale <- CS_vol/NS_vol
```

Global Parameters

```
f0 <- 0.6
r_pp <- 1e-1
#v
r_pp <- r_pp*vol_scale
n <- 2/3
p <- n
q <- 3/4
lambda <- 2+q-n
kappa <- 7e10
```

```
#v
kappa <- kappa*vol_scale
```

Species Parameters

```
erepro <- rep(0.1,no_sp) # Will be changed later
beta <- rep(100,no_sp)
sigma <- rep(1.3,no_sp)
h <- rep(30,no_sp)
ks <- rep(4,no_sp)
alpha <- rep(0.4,no_sp)
z0 <- rep(0,no_sp)
sel_func <- rep("knife_edge",no_sp)
knife_edge_size <- rep(10^2,no_sp)
```

Input mullet Parameters

```
#params_data_NS <- read.csv("./vignettes/NS_species_params.csv")
rep_idx <- no_sp
gurn_sp <- 8

w_min[rep_idx] <- 0.001
w_inf[rep_idx] <- 251.94
w_mat[rep_idx] <- 16.48
beta[rep_idx] <- 283
sigma[rep_idx] <- 1.8
h[rep_idx] <- 47.36894
# h was computed using k_vb according to formula below
ks[rep_idx] <- 4

###############

#params_data_NS <- read.csv("./vignettes/NS_species_params.csv")
#rep_idx <- no_sp
#gurn_sp <- 8
#lam <- 1 # when lam is one the new species gets mullet parameters
#params_data_NS$w_inf[gurn_sp] <- 251.94
#params_data_NS$w_mat[gurn_sp] <- 16.48
#params_data_NS$k_vb[gurn_sp] <- 0.6
#w_min[rep_idx] <- lam*0.001+(1-lam)*w_min[rep_idx]
#w_inf[rep_idx] <- lam*params_data_NS$w_inf[gurn_sp]+(1-lam)*w_inf[rep_idx]
#w_mat[rep_idx] <- lam*params_data_NS$w_mat[gurn_sp]+(1-lam)*w_mat[rep_idx]
#beta[rep_idx] <- lam*params_data_NS$beta[gurn_sp]+(1-lam)*beta[rep_idx]
#sigma[rep_idx] <- lam*params_data_NS$sigma[gurn_sp]+(1-lam)*sigma[rep_idx]
#h_gurn <- 3*params_data_NS$k_vb[gurn_sp]*((params_data_NS$w_inf[gurn_sp])^(1/3))/(alpha[rep_idx]*f0)
#h[rep_idx] <- lam*h_gurn+(1-lam)*h[rep_idx]
#ks[rep_idx] <- lam*0.2*h_gurn+(1-lam)*ks[rep_idx] # how is ks determined by mizer params
#ks[rep_idx] <- 4


#################
```

Build Params Object

```
species_params <- data.frame(
  species = 1:no_sp,
  w_min = w_min,
  w_inf = w_inf,
  w_mat = w_mat,
  h = h,
  ks = ks,
  beta = beta,
  sigma = sigma,
  z0 = z0,
  alpha = alpha,
  erepro = erepro,
  sel_func = sel_func, # not used but required
  knife_edge_size = knife_edge_size
)

params <- MizerParams(species_params, p=p, n=n, q=q, lambda = lambda, f0 = f0,
                      kappa = kappa, min_w = min_w, max_w = max_w, no_w = no_w,
                      min_w_pp = min_w_pp, w_pp_cutoff = max_w, r_pp = r_pp,
                      chi = 0)
```

```
## Note:    No gamma column in species data frame so using f0, h, beta, sigma, lambda and kappa to calcu
```

gamma is determined by mizerparams. Note that density dependence is currently off

```
gamma <- params@species_params$gamma
w <- params@w
```

Determine analytic solution

We should expand this code so that it supports more additional species. First we get the exact solution n_exact for the 1st background species.

```
mu0 <- ((1-f0) * sqrt(2*pi) * kappa * gamma * sigma *
         (beta^(n-1)) * exp(sigma^2 * (n-1)^2 / 2))[1]
hbar <- (alpha * h * f0 - ks)[1]
pow <- (mu0/hbar/(1-n))
n_mult <- (1 - (w/w_inf[1])^(1-n))^(pow-1) * (1 - (w_mat[1]/w_inf[1])^(1-n))^(-pow)
n_mult[w < w_mat[1]] <- 1
n_mult[w >= w_inf[1]] <- 0
n_exact <- w  # Just to get array with correct dimensions and names
n_exact <- ((w_min[1]/w)^(mu0/hbar) / (hbar * w^n)) * n_mult
n_exact[w < w_min[1]] <- 0
```

use n_exact as a template to build the solution for other background species

```
initial_n <- params@psi
initial_n[,] <- 0
w_inf_idx <- w_inf
for (i in 1:no_sp) {
  w_inf_idx[i] <- length(w[w<=w_inf[i]])
  initial_n[i, params@species_params$w_min_idx[i]:
              (params@species_params$w_min_idx[i]+
                 (w_inf_idx[1]-params@species_params$w_min_idx[1]))] <-
    n_exact[params@species_params$w_min_idx[1]:
              (params@species_params$w_min_idx[1]+
                 (w_inf_idx[1]-params@species_params$w_min_idx[1]))] *
    (w_min[1]/w_min[i])^lambda
}
```

Setup parameters for mullet

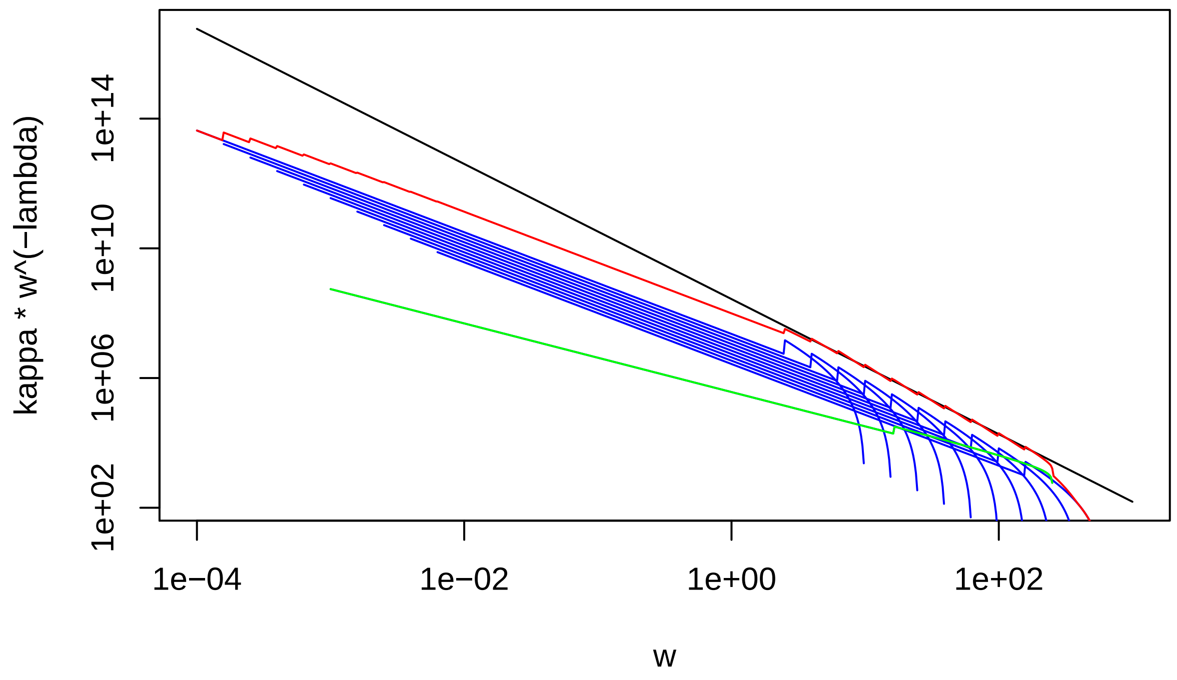
```
hbar_g <- (alpha * h * f0 - ks)[rep_idx]
pow_g <- (mu0/hbar_g/(1-n))
#w_mat_old <- (w_mat[2]/w_mat[1])*w_mat[length(w_mat)-1]
#w_mat_old_idx <- sum(w <= w_mat_old)
#target_mat_abund <- initial_n[rep_idx,w_mat_old_idx]
#v target_mat_abund <- 10*3.368058e-09
target_mat_abund <- 50*3.368058e-07
#v
target_mat_abund <- target_mat_abund*vol_scale
# set things up so this is the abundance at maturity (does this get rescaled when
# we determine n_output ?)
w_mat_g <- w_mat[rep_idx]
w_mat_g_idx <- sum(w <= w_mat_g)
n_mult_g <- (1 - (w/w_inf[rep_idx])^(1-n))^(pow_g-1) * (1 - (w_mat[rep_idx]/w_inf[rep_idx])^(1-n))^(-pow
n_mult_g[w < w_mat[rep_idx]] <- 1
n_mult_g[w >= w_inf[rep_idx]] <- 0
n_exact_g <- w  # Just to get array with correct dimensions and names
n_exact_g <- ((w_min[rep_idx]/w)^(mu0/hbar_g) / (hbar_g * w^n)) * n_mult_g
n_exact_g[w < w_min[rep_idx]] <- 0
n_exact_g <- n_exact_g*target_mat_abund/n_exact_g[w_mat_g_idx]
initial_n[rep_idx,] <- n_exact_g
```

renormalize solutions so there sum agrees with plankton powerlaw at v

```
v <- sqrt(min(w_mat)*max(w_mat))
v_idx <- length(w[w<v])
n_output <- initial_n*(kappa*w[v_idx]^(-lambda))/sum(initial_n[,v_idx])
```

Plot solution. mullet is green, sum of fish spectra is red.

```
plot(w, kappa*w^(-lambda), type="l", log="xy")
for (i in 1:no_sp) {
  lines(w, n_output[i,], col="blue")
}
lines(w, n_output[rep_idx,], col="green")
lines(w, colSums(n_output), col="red")
```



Setup plankton

```
plankton_vec <- (kappa*w^(-lambda))-colSums(n_output)
plankton_vec[plankton_vec<0] <- 0
plankton_vec[min(which(plankton_vec==0)):length(plankton_vec)] <- 0
params@cc_pp[sum(params@w_full<=w[1]):length(params@cc_pp)] <- plankton_vec
initial_n_pp <- params@cc_pp
# The cc_pp factor needs to be higher than the desired steady state in
# order to compensate for predation mortality
m2_background <- getM2Background(params, n_output, initial_n_pp)
params@cc_pp <- (1+m2_background/params@rr_pp) * initial_n_pp
##params@cc_pp <- (1+(mu0*params@w_full^(n-1))/params@rr_pp) * initial_n_pp
# using m2_background seems to work ok. using mu0*params@w_full^(n-1) seems to indice
# oscilations
```

Setup background death and steplike psi

```
m2 <- getM2(params, n_output, initial_n_pp)

for (i in 1:no_sp) {
  params@psi[i, ] <- (w/w_inf[i])^(1-n)
  params@psi[i, w < (w_mat[i]-1e-10)] <- 0
  params@psi[i, w > (w_inf[i]-1e-10)] <- 1
  params@mu_b[i, ] <- mu0 * w^(n-1) - m2[i,]
}
params@mu_b[rep_idx, ] <- mu0 * w^(n-1) - m2[rep_idx,]
```

Set erepro to meet boundary condition

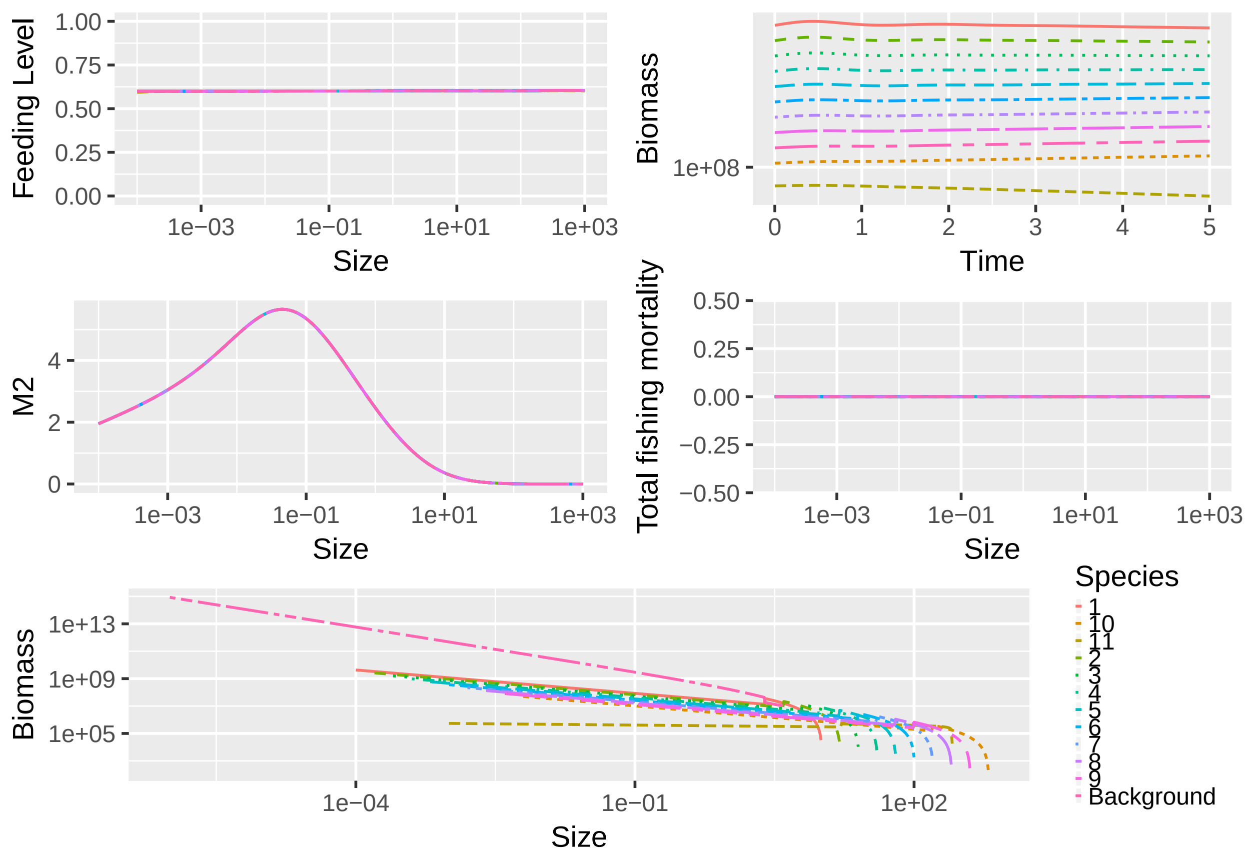
```
rdi <- getRDI(params, n_output, initial_n_pp)
gg <- getEGrowth(params, n_output, initial_n_pp)
effort <- 0
mumu <- getZ(params, n_output, initial_n_pp, effort = effort)
erepro_final <- rdi
for (i in (1:no_sp)){
  #  erepro_final[i] <- erepro*(gg[i,params@species_params$w_min_idx[i]]*n_output[i,params@species_para
  #    rdi[i]
  gg0 <- gg[i,params@species_params$w_min_idx[i]]
  mumu0 <- mumu[i,params@species_params$w_min_idx[i]]
  DW <- params@dw[params@species_params$w_min_idx[i]]
  erepro_final[i] <- erepro[i]*(n_output[i,params@species_params$w_min_idx[i]]*(gg0+DW*mumu0))/rdi[i]
}

params@species_params$erepro <- erepro_final
```

Simulate without Rmax or chi

```
params@srr <- function(rdi, species_params) {return(rdi)}
params@chi <- 0.0

# start at pertubation away from the steady state we know, and see the system reorganize.

#n_output2 <- n_output
#n_output2[no_sp,] <- n_output[no_sp,]*10
#t_max <- 15
#sim <- project(params, t_max=t_max ,dt=0.01, t_save=t_max/100, effort = 0,
#               initial_n = n_output2, initial_n_pp = initial_n_pp)
#plot(sim)
```

see that we stay close when we start at steady state see initial condition is close to steady

```
t_max <- 5
sim <- project(params, t_max=t_max ,dt=0.01, t_save=t_max/100, effort = 0,
               initial_n = n_output, initial_n_pp = initial_n_pp)
plot(sim)
```

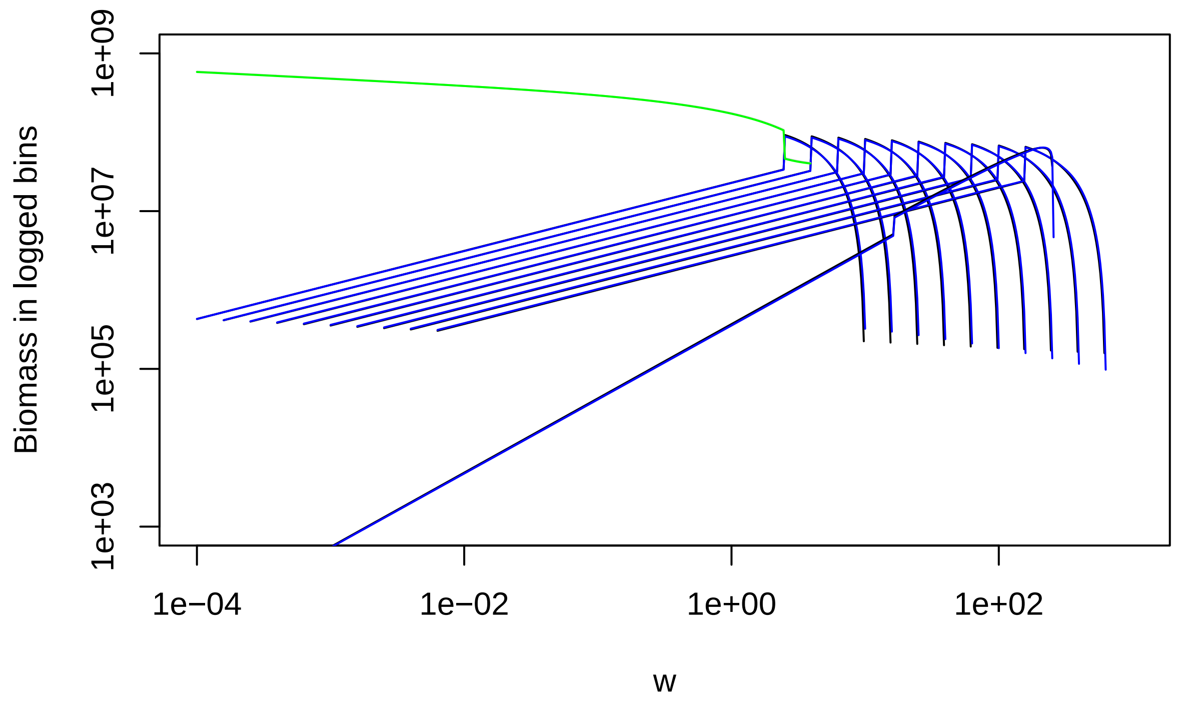


Compare the final state (in blue and green) to the initial condition (in black and grey).

```r
plot(w,w^2*sim@n[1,1,],log="xy",type="l",ylim=c(10^3,10^9),
     ylab="Biomass in logged bins")
```

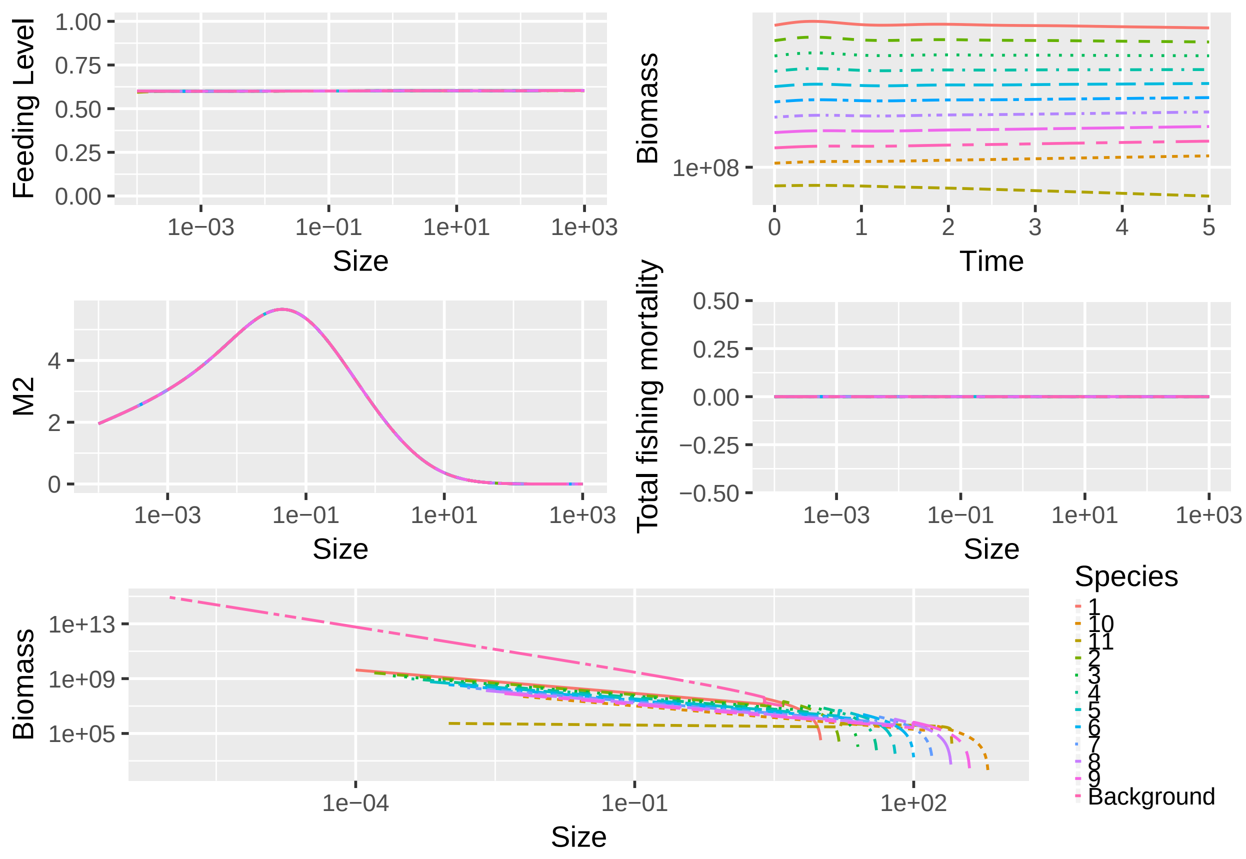
```
## Warning in xy.coords(x, y, xlabel, ylabel, log): 201 y values <= 0 omitted
## from logarithmic plot
```

```r
lines(w,w^2*sim@n[dim(sim@n)[1],1,],col="blue")
for (i in (2:no_sp)){
  lines(w,w^2*sim@n[1,i,],col="black")
  lines(w,w^2*sim@n[dim(sim@n)[1],i,],col="blue")
}
fish_indices <- (length(params@w_full)-length(w)+1):length(params@w_full)
lines(w,w^2*sim@n_pp[1, fish_indices],col="grey")
lines(w,w^2*sim@n_pp[dim(sim@n)[1], fish_indices],col="green")
```

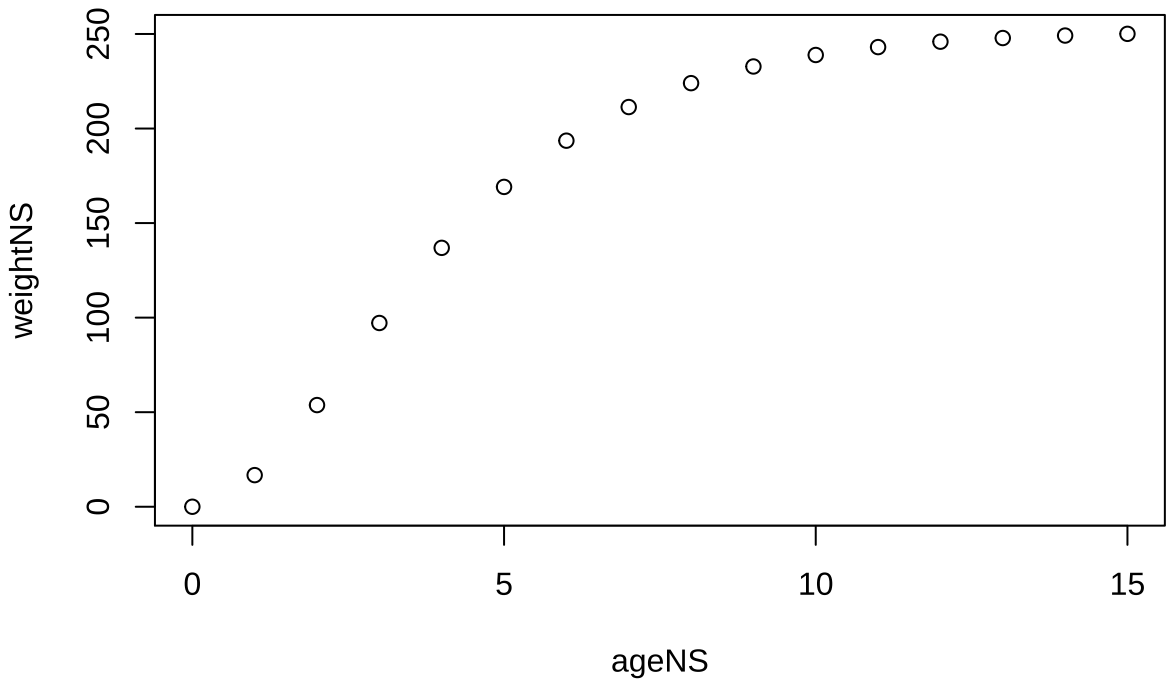


This code suggests that the background+mullet system (which now has the background death setup properly), also has many steady states (possibly arranged in something near to a convex set)

```
######################################################################################## growth


##############

inter <- params@interaction
mytimes <- seq(0.5,10,0.1)
param1 <- params
test<-sim
plot(test)
```



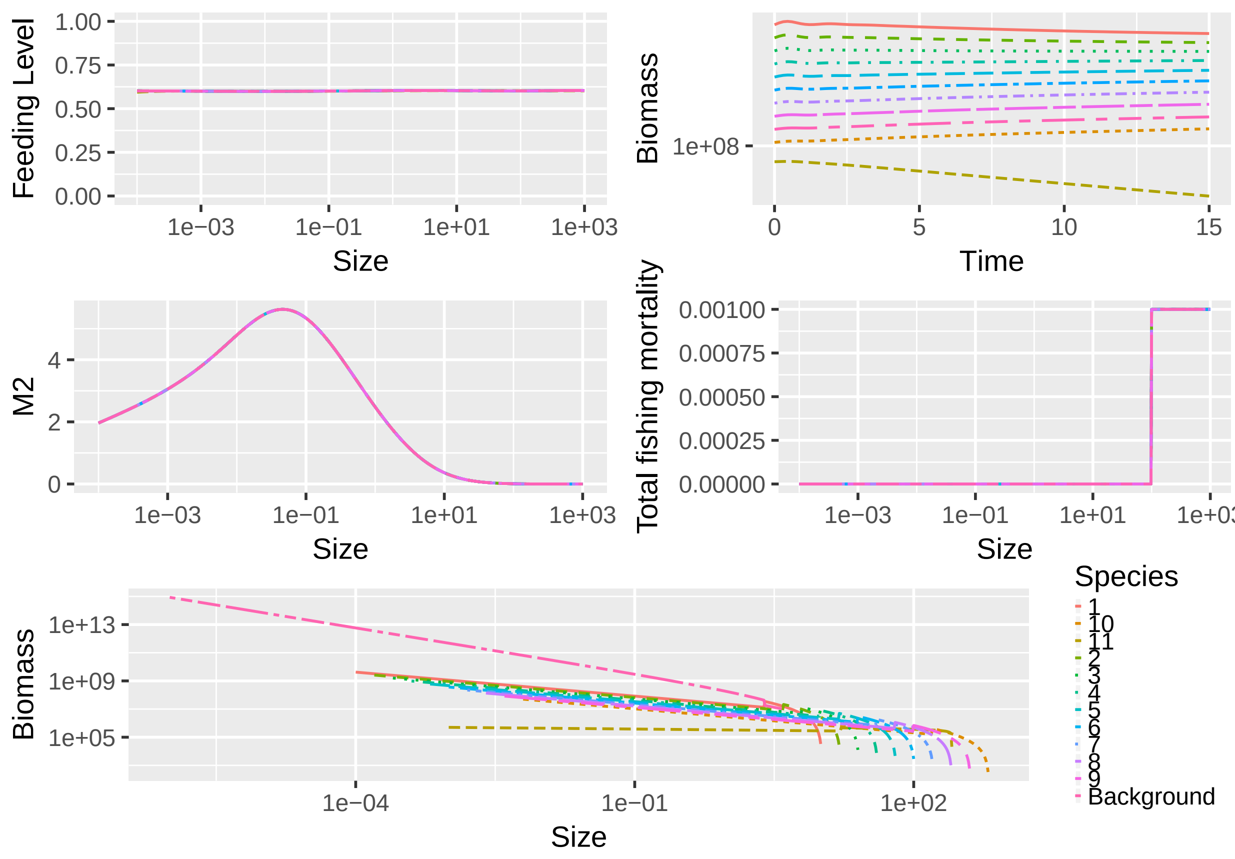
```
##############
my_sp <- no_sp
gNS <- getEGrowth(param1, test@n[dim(test@n)[1], , ], test@n_pp[dim(test@n_pp)[1], ])[my_sp,]

g_fnNS <- approxfun(param1@w, gNS)
myodefunNS <- function(t, state, parameters){
  return(list(g_fnNS(state)))
}
ageNS <- (0:15)
library(deSolve)
weightNS <- ode(y = param1@species_params$w_min[my_sp], times = ageNS, func = myodefunNS, parms = 1)[,2
plot(ageNS,weightNS)
```



```
################################################################################
```

Add Fishing

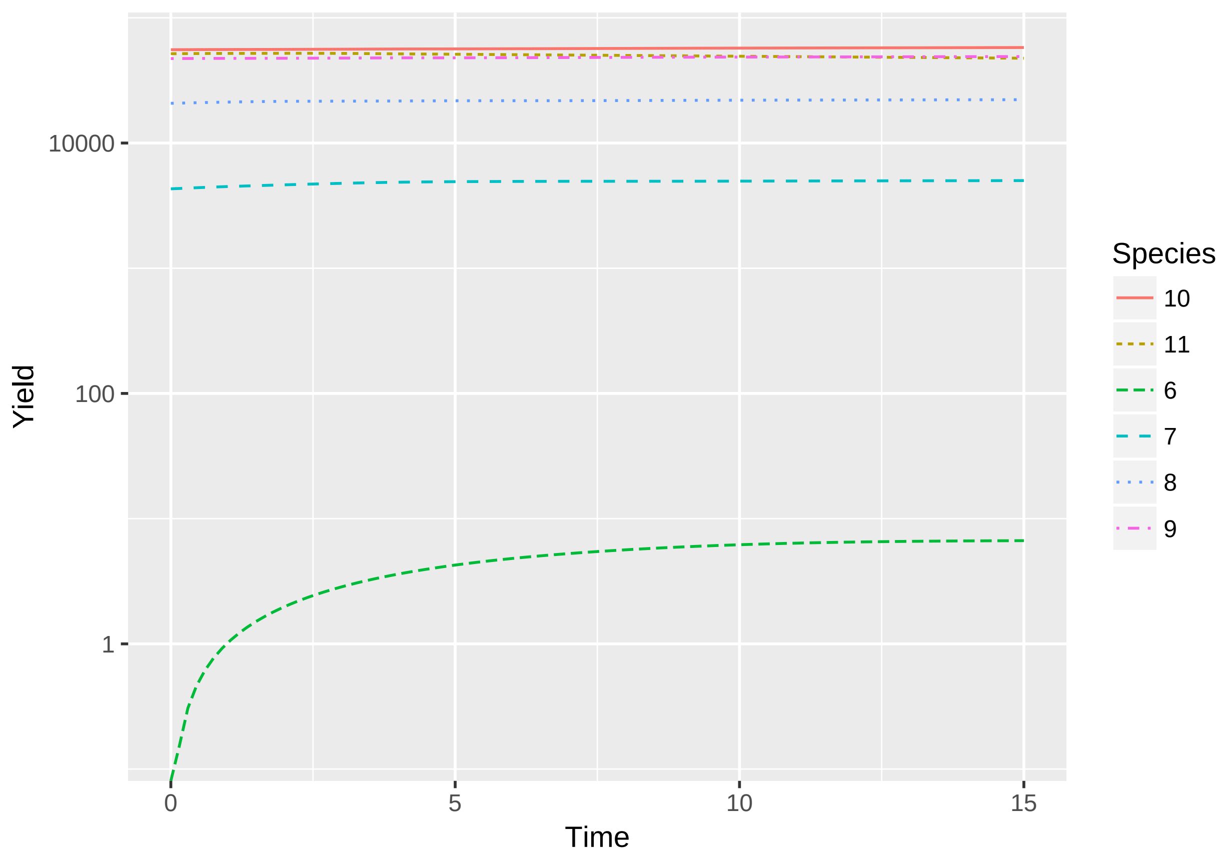
```
eff <- 10^(-3)

params@srr <- function(rdi, species_params) {return(rdi)}
params@chi <- 0.0
t_max <- 15
sim <- project(params, t_max=t_max ,dt=0.01, t_save=t_max/100, effort = eff,
               initial_n = n_output, initial_n_pp = initial_n_pp)
plot(sim)
```



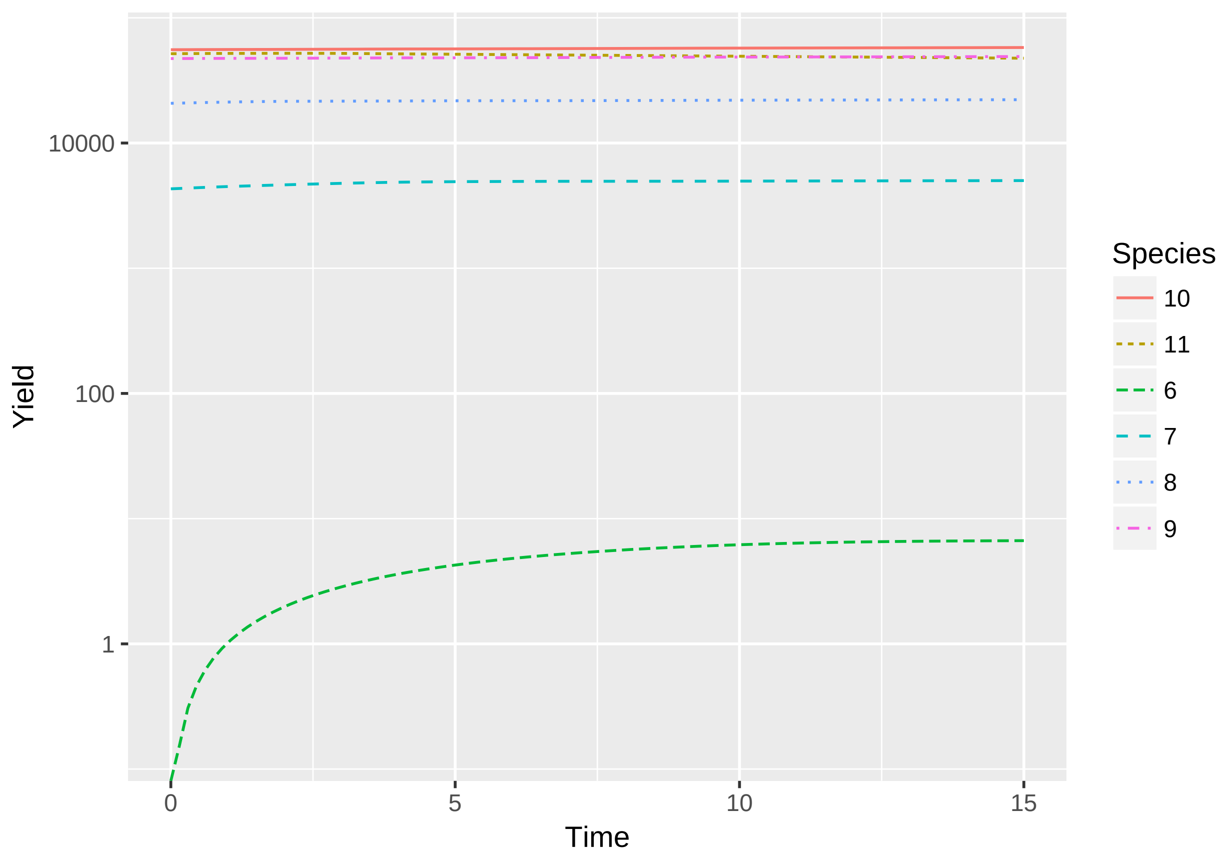
plot yields

```r
plotYield(sim)
```

```
## Warning: Transformation introduced infinite values in continuous y-axis
```



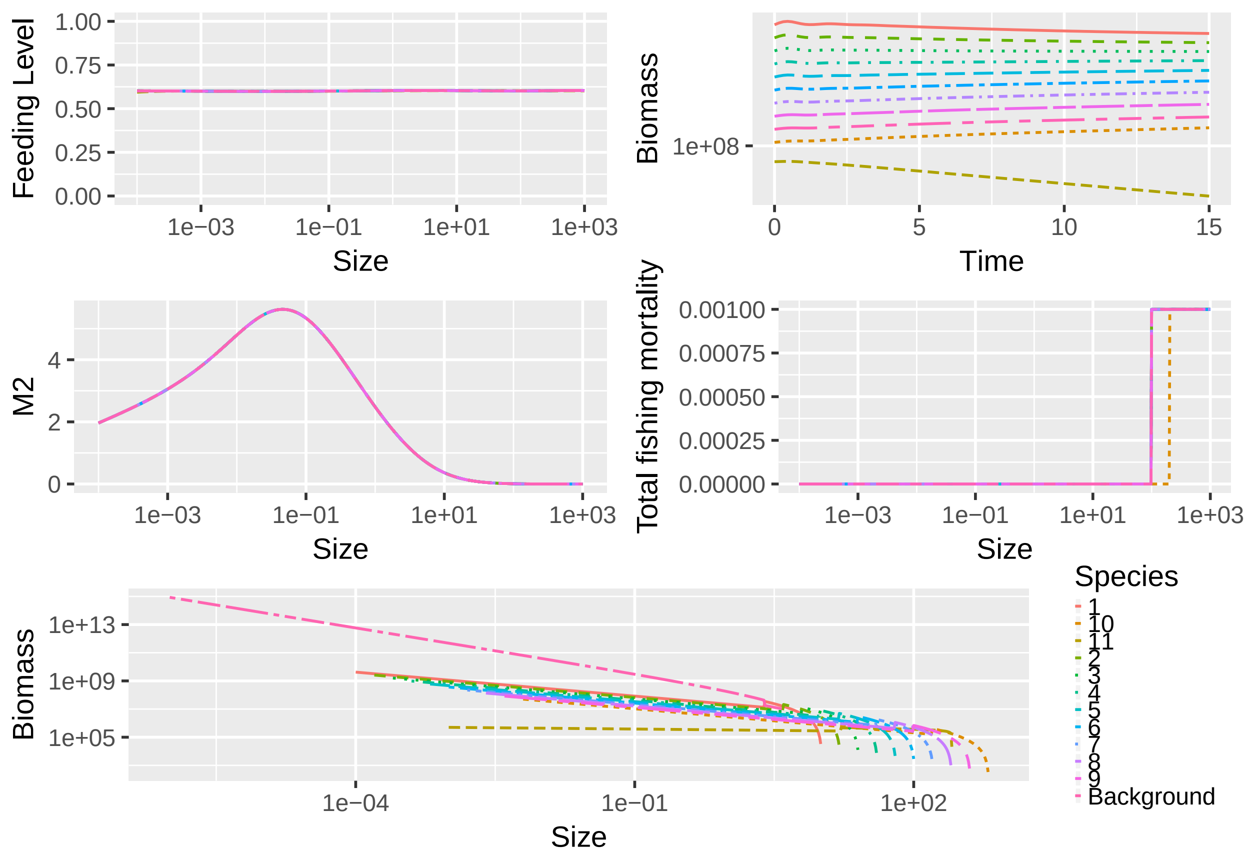
```
## Warning: Transformation introduced infinite values in continuous y-axis
```



```
gy100 <- getYield(sim)
```

change gear for mullet to 200g knife edge, and plot yields again

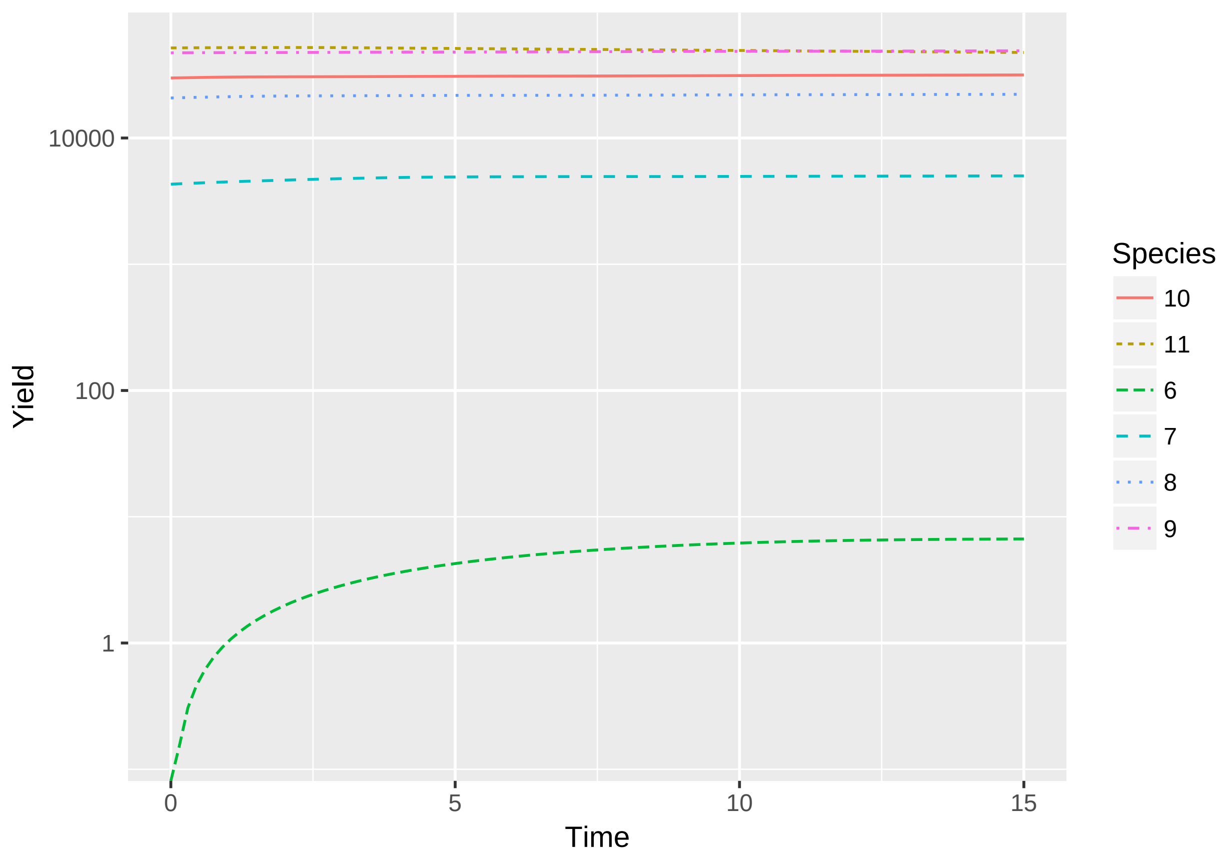
```
rep_idx2 <- no_sp-1
params@selectivity[rep_idx2,rep_idx2,] <- 0
params@selectivity[rep_idx2,rep_idx2,w>=200] <- 1
t_max <- 15
sim200 <- project(params, t_max=t_max ,dt=0.01, t_save=t_max/100, effort = eff,
               initial_n = n_output, initial_n_pp = initial_n_pp)
plot(sim200)
```



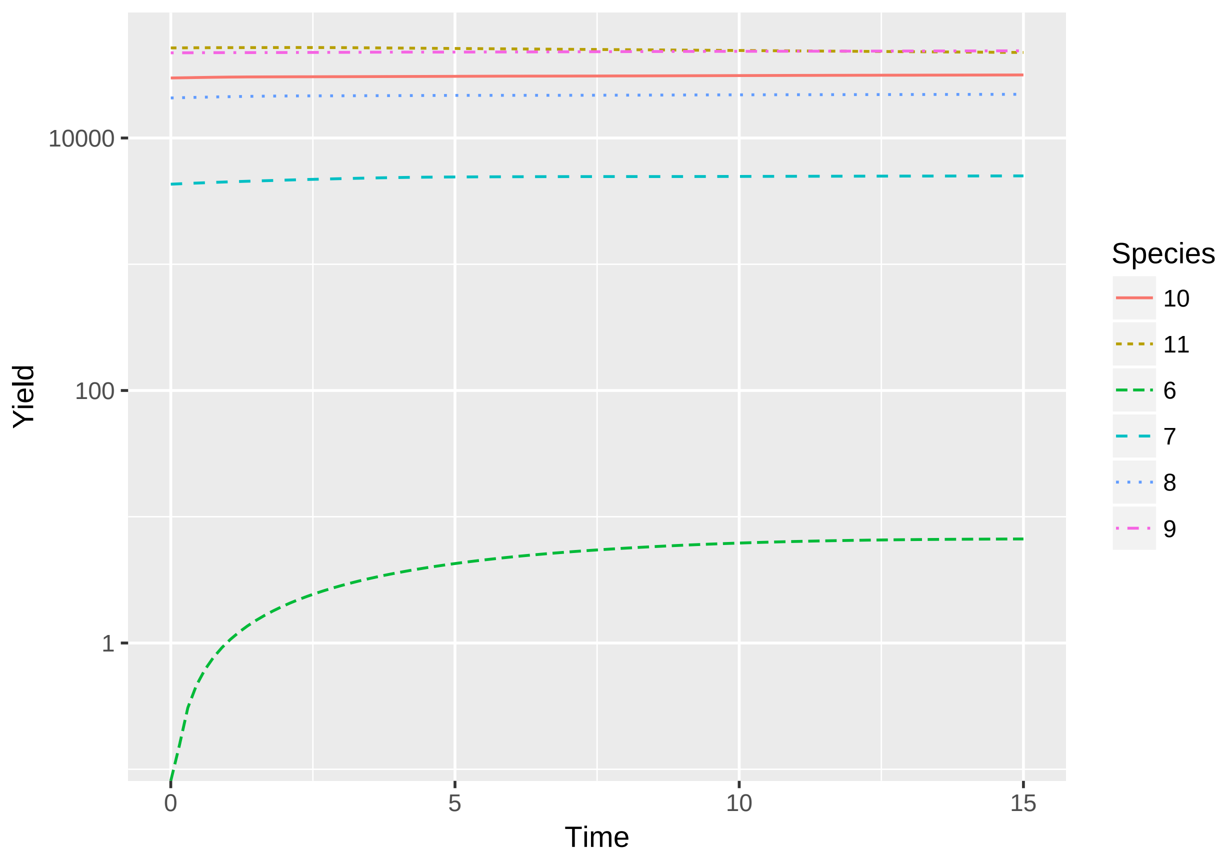
plot yields

```
plotYield(sim200)
```

```
## Warning: Transformation introduced infinite values in continuous y-axis
```



```
## Warning: Transformation introduced infinite values in continuous y-axis
```



```
gy200 <- getYield(sim200)

#################
dim(gy100)
```

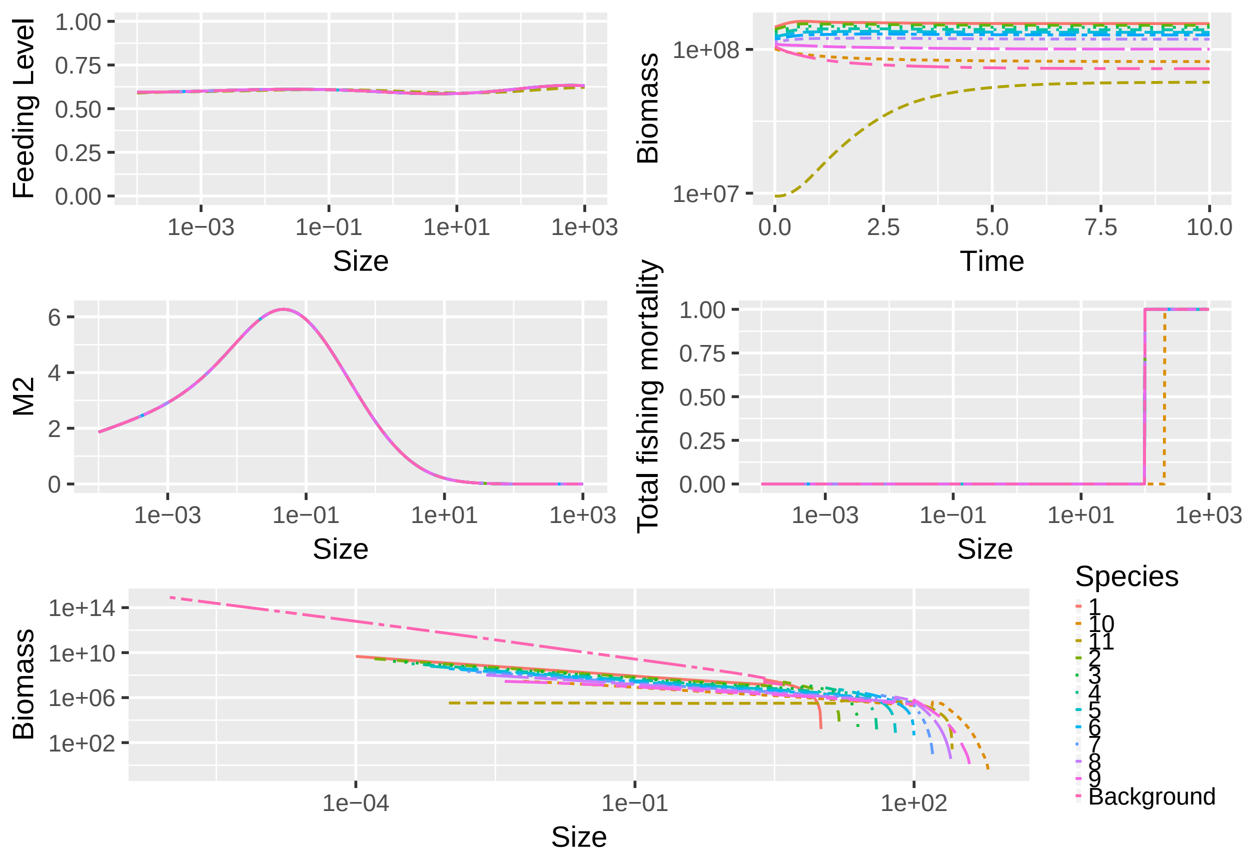
```
## [1] 101  11
```

```
gy200[dim(gy200)[1],]/gy100[dim(gy100)[1],]
```

```
##         1         2         3         4         5         6         7
##       NaN       NaN       NaN       NaN       NaN 1.0000434 0.9999460
##         8         9        10        11
## 0.9998615 0.9998000 0.5454638 0.9999491
```

Add Density Dependence

```
nn <- n_output
nn[nn==0] <- 1
params@chi <- 0.5
params@ddd <- nn^(params@chi)
n_output2 <- n_output
n_output2[no_sp,] <- 0.1*n_output[no_sp,]
t_max <- 10
sim <- project(params, t_max=t_max ,dt=0.01, t_save=t_max/100, effort = 1,
               initial_n = n_output2, initial_n_pp = initial_n_pp)
plot(sim)
```



```
# code using mullus baratus parameters, plus background. Had to use a ks value different from 0.2*h to
```