Title: Evaluating consumptive and nonconsumptive predator effects on prey density using field times series data

Authors: J.A. Marino, Jr., S.D. Peacor, D.B. Bunnell, H.A. Vanderploeg, S.A. Pothoven, A.K. Elgin, J.R. Bence, J. Jiao, and E.L. Ionides

Journal: *Ecology*

Appendix S1

Table of Contents

Model description and code

We describe the model and pomp code for a Lake Michigan zooplankton system. The state variable is a victim, V (i.e., *Daphnia mendotae*). The predator, P (i.e., *Bythotrephes longimanus*, a large predatory zooplankter) is treated as a covariate. Dynamics are represented by the following stochastic differential equations in which Brownian motion (dW) drives stochasticity in prey growth.

$$dV = (V\beta(t)(1 - V/\kappa)(1 - \eta g(P)) - \alpha P - \mu V)\, dt + V\epsilon\, dW + \rho(t)$$

$$dW \sim \text{Normal}(0, sd = \sqrt{dt})$$

where $\beta(t)$ represents prey brith and/or somatic growth rate at low population size, κ represents density dependence for the prey population, η is a proportional reduction in *D. mendotae* birth rate caused by *B. longimanus* (i.e., a nonconsumptive effect), α is predator attack rate, μ is prey background mortality, ϵ is the standard deviations of the prey population growth rate, and $\rho(t)$ is a pulse term that initiates prey dynamics each year. Prey birth rate changes seasonally according to:

$$\beta(t) = \exp\left\{\sum_{i=1}^{N_s} \lambda_i \, s_i(t)\right\}$$

where $\{s_i(t), i = 1, \ldots, N_s\}$ is a periodic cubic B-spline basis; $\{\lambda_i, i = 1, \ldots, N_s\}$ model seasonality of the birth rate. We set $N_s = 4$.

$\rho(t)$ represents the initiation of the population each year via *D. mendotae* emergence from resting eggs. The size of this pulse is assumed to be random and log-normally distributed, with parameters ϕ and ψ representing the natural log of the mean and standard deviation of the pulse (ρ(t)), respectively:

$$\ln(\rho(t)) \sim \text{Normal}(\phi, \psi)$$

The pulse is assumed to occur 7 days prior to the first observation of *D. mendotae*. On other dates, $\rho(t)$ = 0.

We build the process model according to the above equations.

```
Daphnia_rprocess <- Csnippet("
  double births;
  double deaths;
  double betaz;
  double dw;

  if (error_count != 0.0) return;

  betaz = exp(dot_product(4,&seas_1,&log_betaz1));

  dw = rnorm(0,sqrt(dt));  // white noise for prey growth

  births = (betaz*(1-eta*(7.601+log(byth + 0.0005)))*dt)*prey*(1-prey/kappa)
+
  prey*sdbeta*dw + pulse*rlnorm(mean_rho,sigma_rho);    // prey births
  deaths = prey*(alpha*byth*0 + mu)*dt; // prey deaths due to predation

  prey += births - deaths;
  noise += dw;

  // check for violations of positivity constraints
```

```
  if (prey <= 0.0)  {
    prey = 0;
  }
  //reset population to 0 at beginning of each year
  if (nocoup < 1)  {
    prey = 0;
  }
  ")
```

The data consist of measurements of *D. mendotae* biomass density:

```
#need to import data
zoop<-read.table("GLERL_M110_Zoop_1994-2012.txt",header=TRUE)
zoop$day<-as.numeric(1+(as.Date(zoop$Date, format="%m/%d/%y")-rep(as.Date("19
94-01-01"),length(zoop$Date))))
zoopsubset<-subset(zoop,Prior_to_1st_Daphnia_obs==0) #generate subset excludi
ng early zero observations
```

We generate the *B. longimanus* covariate by assuming that *B. longimanus* is absent from the water column during the first 50 days each year beginning January 1 and then linearly interpolating remaining data points from observed biomass density of *B. longimanus*. We then use those interpolated values to calculate a 45-day moving average.

```
zoop$cases<-complete.cases(zoop$D.mendotae);zoopcomplete<-subset(zoop,cases==
TRUE)
require(timeSeries)
Byth<-rep(c(rep(0,50),rep(NA,315)),19);
Byth[zoopcomplete$day]<-zoopcomplete$Bythotrephes; Byth[1]<-0;Byth[6935]<-0
Byth<-interpNA(Byth, method = "linear")
Byth_corrected<-Byth+0.0005

#calculate 45 day moving average
BythMA45<-rep(0.0005,6935)

for (i in 23:6913) {
  BythMA45[i]<-(Byth_corrected[i-22]*Byth_corrected[i-21]*
    Byth_corrected[i-20]*Byth_corrected[i-19]*Byth_corrected[i-18]*
    Byth_corrected[i-17]*Byth_corrected[i-16]*Byth_corrected[i-15]*
    Byth_corrected[i-14]*Byth_corrected[i-13]*Byth_corrected[i-12]*
    Byth_corrected[i-11]*Byth_corrected[i-10]*Byth_corrected[i-9]*
    Byth_corrected[i-8]*Byth_corrected[i-7]*Byth_corrected[i-6]*
    Byth_corrected[i-5]*Byth_corrected[i-4]*Byth_corrected[i-3]*
    Byth_corrected[i-2]*Byth_corrected[i-1]*Byth_corrected[i]*
    Byth_corrected[i+1]*Byth_corrected[i+2]*Byth_corrected[i+3]*
    Byth_corrected[i+4]*Byth_corrected[i+5]*Byth_corrected[i+6]*
    Byth_corrected[i+7]*Byth_corrected[i+8]*Byth_corrected[i+9]*
    Byth_corrected[i+10]*Byth_corrected[i+11]*Byth_corrected[i+12]*
    Byth_corrected[i+13]*Byth_corrected[i+14]*Byth_corrected[i+15]*
    Byth_corrected[i+16]*Byth_corrected[i+17]*Byth_corrected[i+18]*
    Byth_corrected[i+19]*Byth_corrected[i+20]*Byth_corrected[i+21]*
```

```
    Byth_corrected[i+22])^(1/45)
}

#need to back-transform
BythMA45<-BythMA45-0.0005;BythMA45<-round(BythMA45,10)
```

We build the periodic B-spline basis for $\beta(t)$. We also generate the covariate representing an influx of *D. mendotae* (set to occur 1 week prior to the observation of *D. mendotae* each year) to determine the timing of the pulse ($\rho(t)$).

```
#set up covariate to reset population to 0 each year
nocoup<-rep(c(rep(1,364),0),19)

#figure out first date of daphnia observation each year
#need to determine first observation of Daphnia each year
zoopcomplete$year<-year(as.Date(zoopcomplete$Date, format="%m/%d/%y"))
surveyyears<-c(1994:2003,2007:2012)
firstdaph<-rep(0,16)
firstdaphjday<-rep(0,16)
firstdaphdens<-rep(0,16)

for (i in 1:16) {
  zoopyear<-subset(zoopcomplete,year==surveyyears[i])
  zoopyearno0<-subset(zoopyear,D.mendotae>0)
  firstdaph[i]<-zoopyearno0$day[which.min(zoopyearno0$JulianDay)]
  firstdaphjday[i]<-zoopyearno0$JulianDay[which.min(zoopyearno0$JulianDay)]
  firstdaphdens[i]<-zoopyearno0$D.mendotae[which.min(zoopyearno0$JulianDay)]
}

#set start equal to 7 days before each observation
daphstart<-firstdaph-7

#create vector with 1s at time of first observation to indicate when pulse oc
curs
pulse<-rep(0,6935)
pulse[daphstart]<-1



covar.dt <- 1
t0 <- 0
nbasis <- 4
tcovar <- seq(from=t0,to=6934,by=covar.dt)
yr <- 1:365

covartable <- data.frame(
  time=tcovar,
  seas=periodic.bspline.basis(tcovar,nbasis=nbasis,degree=3,period=365),
  byth=BythMA45,
  nocoup,
```

```
  pulse
)
```

Here are some candidate parameter values:

```
params.init <- c(sigmaa=0.21907755,sigmab=0.38558873,
                 log.betaz1=-10.04763994,log.betaz2=-3.41093475,
                 log.betaz3=-1.06310127,log.betaz4=0.31538577,
                 alpha=0,mu=0.04784941,eta=0.08924933,
                 mean_rho=-3.15200907,sigma_rho=1.69402220,
                 kappa=32.54473250,sdbeta=0.25785123,
                 prey.0=0,noise.0=0,error_count.0=0)
```

The state variable V is linked to observed measures of *D. mendotae* biomass density using a measurement model. The measurement model is specified as a left-censored normal model to handle the zero observations, with demographic and environmental scale measurement noise terms (σ_a and σ_b) .

$$Vobs_{(t)} \sim \text{Normal}(V_{(t)}, \sigma)$$

$$\sigma = \sqrt{\sigma_a^2 V_{(t)} + \sigma_b^2 V_{(t)}^2}$$

if $Vobs_{(t)} > 0$. Otherwise, $Vobs_{(t)} = 0$

```
Daphnia_rmeasure <- Csnippet("
  double sigma, daphnia;
  sigma=sqrt(sigmaa*sigmaa*prey + sigmab*sigmab*prey*prey);
  if ((error_count > 0.0) || (!(R_FINITE(prey)))) {
    D_mendotae = R_NaReal;
  } else {
    daphnia = rnorm(prey,sigma);
  if (daphnia<=0) {
    D_mendotae=0;
  }
  else {
    D_mendotae=daphnia;
  }
  }
")

Daphnia_dmeasure <- Csnippet("
  double sigma, tol = 1.0e-18;
  sigma=sqrt(sigmaa*sigmaa*prey + sigmab*sigmab*prey*prey);
  double f = 0.0;
  if ((error_count > 0.0) || (!(R_FINITE(prey)))) {
    lik = tol;
  } else {
  if (D_mendotae==0) {
    f += pnorm(0,prey,sigma,1,1)+tol;
```

```
  }
  else {
    f += dnorm(D_mendotae, prey, sigma, 1)+tol;
  }
  lik = (give_log) ? f : exp(f);
  }
")
```

We also build some parameter transformations. These are only needed for optimization, so could be added later when we get to mif2 (iterated filtering).

```
Daphnia_untrans <- Csnippet("
  Tsigmaa = log(sigmaa);
  Tsigmab = log(sigmab);
  Tkappa = log(kappa);
  Tsdbeta = log(sdbeta);
  Tmu = log(mu);
  Teta = log(eta);
  Tsigma_rho = log(sigma_rho);
  ")

Daphnia_trans <- Csnippet("
                      Tsigmaa = exp(sigmaa);
                      Tsigmab = exp(sigmab);
                      Tkappa = exp(kappa);
                      Tsdbeta = exp(sdbeta);
                      Tmu = exp(mu);
                      Teta = exp(eta);
                      Tsigma_rho = exp(sigma_rho);
                      ")
```

We then construct the pomp object.

```
Daphnia_pomp <- pomp(
  data=subset(zoopsubset,select=c("D.mendotae","day")),
  times="day",
  t0=0,
  params=params.init,
  rprocess = euler.sim(step.fun = Daphnia_rprocess, delta.t=1),
  rmeasure= Daphnia_rmeasure,
  dmeasure = Daphnia_dmeasure,
  covar=covartable,
  tcovar="time",
  obsnames = c("D.mendotae"),
  zeronames = c("error_count"),
  statenames = c("prey","noise","error_count"),
  paramnames = c("sigmaa","sigmab","log.betaz1","kappa","alpha","mu","eta","s
dbeta","mean_rho","sigma_rho","prey.0","noise.0","error_count.0"),
  covarnames = c("seas.1","byth","nocoup","pulse"),
  fromEstimationScale=Daphnia_trans,
```

```
  toEstimationScale=Daphnia_untrans
)
plot(Daphnia_pomp@times,Daphnia_pomp@data[1,],type="l",xlab="Time (days)",yla
b=expression(paste(italic("D. mendotae"), " biomass density (", "mg/m"^"3", "
)")))
```

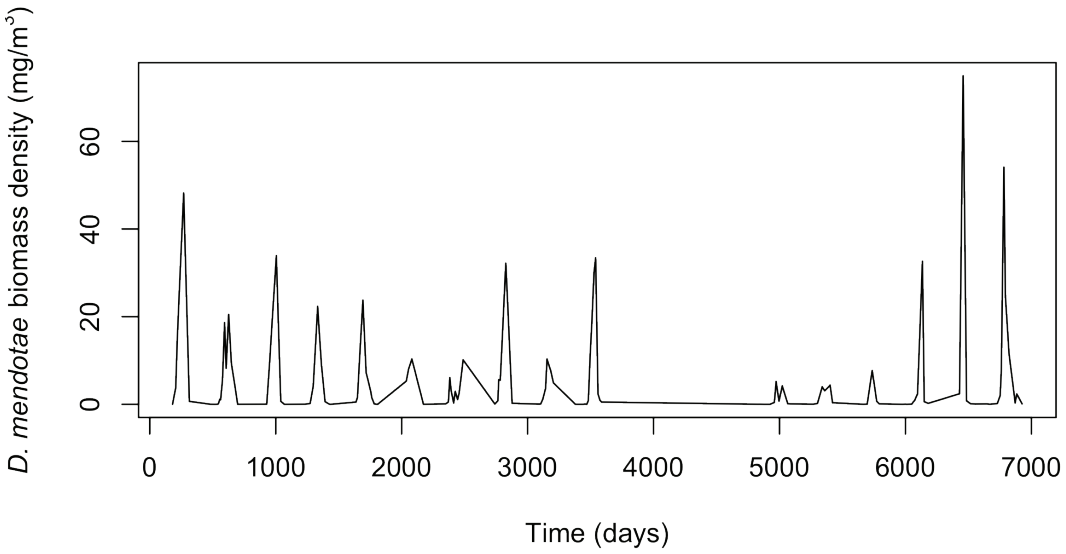

Figure S1: *D. mendotae* biomass density data.

The model can then be simulated to test the coding:

```
sim1 <- simulate(Daphnia_pomp,seed=3)
plot(sim1@times,sim1@data[1,],type="l", xlab="Time (days)", ylab=expression(p
aste("Simulated ",italic("D. mendotae"), " biomass density (", "mg/m"^"3", ")
")))
```

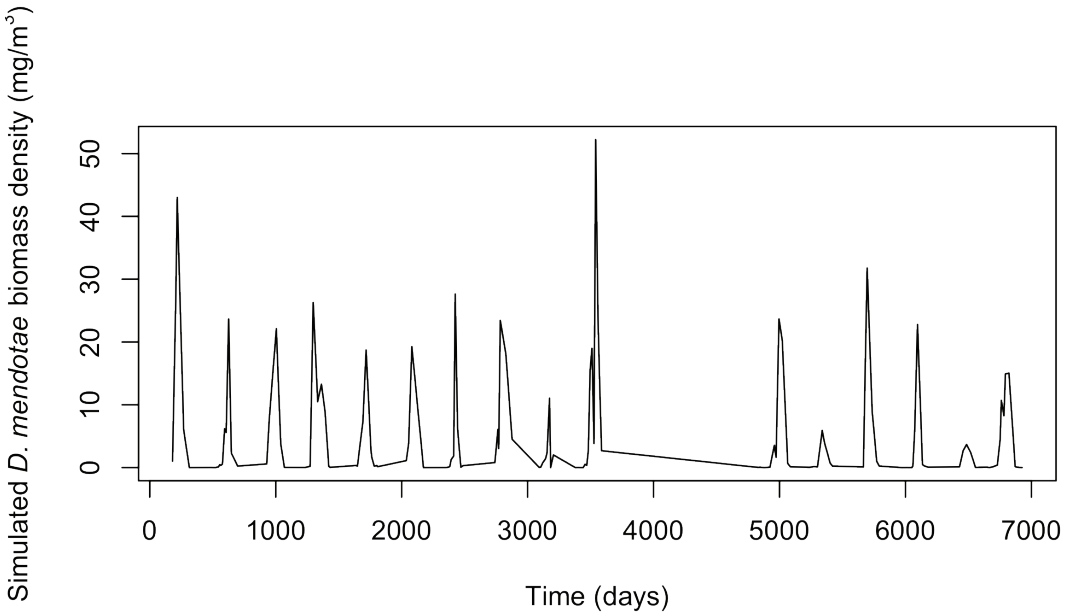

Figure S2: Simulation of *D. mendotae* data from pomp object.

Comparison with benchmark models

A reasonable mechanistic model should outperform some simple benchmarks. First, we compared the mechanistic model to a model assuming that *D. mendoate* biomass density is independently and identically distributed around a seasonal (monthly) average.

```
zoopsubset$month<-month(as.Date(zoopsubset$Date, format="%m/%d/%y"))


#create function for monthly mean (11 parameters because no Jan observations)
and two sigmas
truncnormiid<- function (x) {
  f<-rep(0,134);
  for (i in 1:134) {
    prey<-exp(x[zoopsubset$month[i]-1]);
    D_mendotae<-zoopsubset$D.mendotae[i]; #
    sigma<-sqrt(x[12]^2*prey + x[13]^2*prey^2); #two parameters control error
    if (D_mendotae==0) {
      f[i]<-pnorm(0,prey,sigma,1,1); }
    else {
      f[i]<- dnorm(D_mendotae, prey, sigma, 1);
    }
  }
  -sum(f)
}

testparams<-c(rep(0.0001,11),1,1)
maxf<-optim(par=testparams,fn=truncnormiid,method="BFGS")
```

As an alternative benchmark, we can use an AR (2) model represented as a state-space model with measurement error, using the same measurement model as the mechanistic model.

```
AR_rprocess_one <- Csnippet("
       double Ez;

       if (error_count != 0.0) return;

       Ez=rnorm(prey,epsilonz);
       prey = alphaz*prey + betaz*preylag + Ez;
       preylag = prey;

       // check for violations of positivity constraints
       // nonzero error_count variable signals violation
       if (prey <= 0.0)  {
       prey = 0.00000001;
       }
        if (preylag <= 0.0)  {
        preylag = 0.0;
         error_count += 1;
         }
         ")
```

Both benchmark models fit substantially worse than our mechanistic models (Table 1), strengthening our confidence that our mechanistic models provide a reasonable fit.

Anomaly analysis

To disentangle the contribution of seasonal and inter-annual variation in *D. mendotae* biomass to our results, we performed a loess regression of *B. longimanus* biomass density vs. Julian day. The fitted points represented a seasonal average and the residuals represented the deviations from that average (i.e., an anomaly).

```
#calculate anomaly
jday<-rep(1:365,19)
lo.B<-loess(BythMA45 ~ jday,span=0.25)
bythseas<-lo.B$fitted
bythanom<-lo.B$residuals
#plot(1:6935,bythanom,type="l",xlab="day")
plot(60:365,bythanom[60:365],xlab="Julian day", ylab=expression(paste(italic(
"B. longimanus"), " anomaly (", "mg/m"^"3", ")")),type="l",
     xlim=c(150,365),ylim=c(min(bythanom),max(bythanom)))
legend("topleft",legend=c("annomaly","mean anomaly"),lty=c(1,1),col=c("black"
,"red"))
sampleyear.no<-c(2:10,14:19)
for (i in 1:15) {
  lines(60:365,bythanom[60:365+365*(sampleyear.no[i]-1)])
}
anoavg<-rep(0,365)
```

```
for (i in 1:16) {
  anomavg<-bythanom[1:365 + 365*(i-1)]
}
lines(1:365,anomavg,col="red")
```

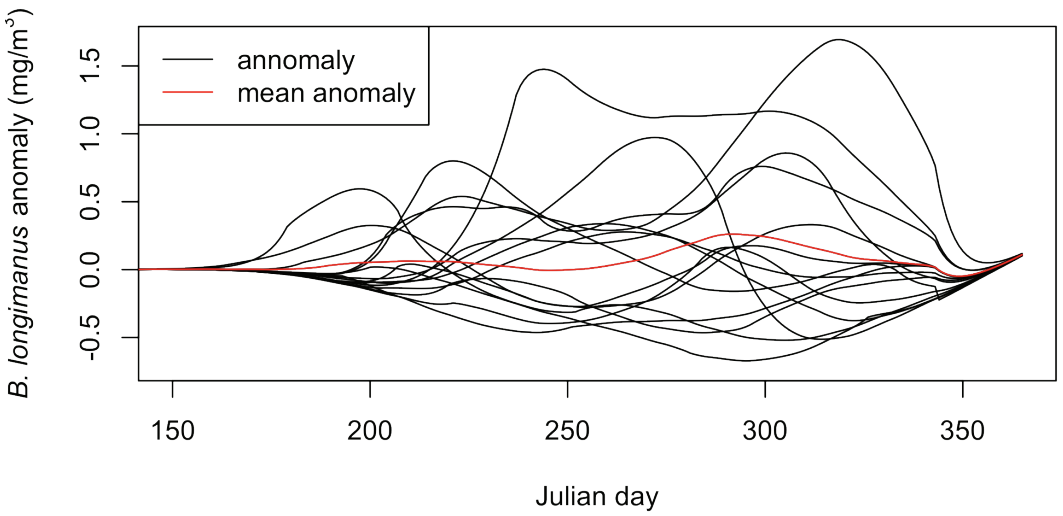

Figure S3: *B. longimanus* anomaly for different years (black line) and averaged across years (red line).

We then can substitute the sum of the seasonal average and anomaly term for total *B. longimanus* biomass density as the predator covariate and test the consequences for model fit.

```
Daphnia_rprocess_anom <- Csnippet("
       double births;
       double deaths;
       double betaz;
       double dw;

       if (error_count != 0.0) return;

       betaz = exp(dot_product(4,&seas_1,&log_betaz1));

       dw = rnorm(0,sqrt(dt));  // white noise for prey birth

       births = (betaz*(1-eta*(0.163 + log(bythanom + 0.85)))*dt)*prey*(1-pre
y/kappa) + prey*sdbeta*dw + pulse*rlnorm(mean_rho,sigma_rho);  // prey births
       deaths = prey*mu*dt; // prey deaths due to predation

       prey += births - deaths;
       noise += dw;
```

```
      // check for violations of positivity constraints
      // nonzero error_count variable signals violation
      if (prey <= 0.0)  {
      prey = 0;
      }
      if (nocoup < 1)  {
      prey = 0;
       }
      ")

covartable_anom <- data.frame(
  time=tcovar,
  seas=periodic.bspline.basis(tcovar,nbasis=nbasis,degree=3,period=365),
  bythseas=bythseas,
  bythanom=bythanom,
  nocoup,
  pulse
)

Daphnia_pomp_anom <- pomp(
  data=subset(zoopsubset,select=c("D.mendotae","day")),
  times="day",
  t0=0,
  params=params.init,
  rprocess = euler.sim(step.fun = Daphnia_rprocess_anom, delta.t=1),
  rmeasure= Daphnia_rmeasure,
  dmeasure = Daphnia_dmeasure,
  covar=covartable_anom,
  tcovar="time",
  obsnames = c("D.mendotae"),
  zeronames = c("error_count"),
  statenames = c("prey","noise","error_count"),
  paramnames = c("sigmaa","sigmab","log.betaz1","kappa","alpha","mu","eta","s
dbeta","mean_rho","sigma_rho","prey.0"),
  covarnames = c("seas.1","bythseas","bythanom","nocoup","pulse"),
  all.state.names=c("prey","noise", "error_count"),
  comp.names=c("prey"),
  comp.ic.names=c("prey.0"),
  fromEstimationScale=Daphnia_trans,
  toEstimationScale=Daphnia_untrans,
  initializer = function (params, t0, comp.ic.names, comp.names, all.state.na
mes, ...) {
    states <- numeric(length(all.state.names))
    names(states) <- all.state.names
    frac <- params[comp.ic.names]
    states[comp.names] <- frac
    states
  }
)
```

The model including the *B. longimanus* anomaly performed substantially better than the null model ($\Delta AIC > 4$), suggesting that interaannual variation contributes to the observed *B. longimanus* effect.

Model fitting

For each model, we performed 100 searches using mif2 function in the pomp R package in which a search through parameter space was initiated using a random set of starting values for each parameter. Starting values were generated from a uniform distribution bounded by plausible values for each parameter.

```
param.tab <- read.table("params.csv",sep=",",row.names=1,header=TRUE)
kable(t(param.tab[1:13]),caption = "Parameter ranges used to generate random starting point for each search")
```

Parameter ranges used to generate random starting point for each search

	lower.bound	upper.bound
sigmadem	0.0010	2.0
sigmaenv	0.0010	2.0
log.betaz1	-11.0000	11.0
log.betaz2	-11.0000	11.0
log.betaz3	-11.0000	11.0
log.betaz4	-11.0000	11.0
alpha	-0.1000	0.1
eta	0.0001	1.0
mean_rho	-10.0000	0.0
sigma_rho	0.0010	10.0
mu	0.0010	1.0
kappa	5.0000	1000.0
sdbeta	0.0010	1.0

```
CORES<-10 ##update for flux
JOBS<-100 ##update for flux

require(doParallel)
registerDoParallel(CORES)

tic <- Sys.time()
mpar <- foreach(
  i=1:JOBS,
  .packages=c('pomp'),
  .inorder=FALSE) %dopar% {
    Sys.sleep(i*.1)
    NMIF<-200 ##update for flux
    NP<-10000 ##update for flux
```

```
    METHOD="mif2"
    param.tab <- read.table("params.csv",sep=",",row.names=1,header=TRUE)
    LV.pars <- c("sigmaa","sigmab","log.betaz1","log.betaz2",
                 "log.betaz3","log.betaz4","alpha", "mu","kappa",
                 "eta","sdbeta","mean_rho","sigma_rho")
    LV.ivps <- c("prey.0")
    LV.rw.sd<- rw.sd(sigmaa=0.02, sigmab=0.02,log.betaz1=0.02,
                     log.betaz2=0.02,log.betaz3=0.02,log.betaz4=0.02,
                     alpha=0.02, mu=0.02,kappa=0.02,eta=0.02,sdbeta=0.02,
                     mean_rho=0.02,sigma_rho=0.02)

    LV.hyperparams <-
      list(min=unlist(param.tab["lower.bound",]),max=unlist(param.tab["upper.
bound",]))

    LV.rprior <- function(hyperparams, ...)
    {
      r<-runif(length(hyperparams$min),min=hyperparams$min,max=hyperparams$ma
x)
      names(r) <- names(hyperparams$min)
      return(r)
    }
    set.seed(8100+i)
    Sys.sleep(i*0.1)
    th.draw <-LV.rprior(LV.hyperparams)
    m<-try(mif2(Daphnia_pomp,
                Nmif=NMIF,
                start=th.draw, # we will initialize
                rw.sd=LV.rw.sd,
                Np=NP,
                cooling.type='geometric',
                cooling.fraction= 0.3,
                max.fail=200,
                transform=TRUE
    ))
    list(pomp=m,start=th.draw)
  }
m.out <- rbind(
      pf.lik = sapply(mpar,function(x){
        if(class(x$pomp)=="mif2d.pomp") logLik(x$pomp) else NA
      }),
      sapply(mpar,function(x) {
        if(class(x$pomp)=="mif2d.pomp") coef(x$pomp) else rep(NA,length(coef(
Daphnia_pomp)))
      }),
      sapply(mpar,function(x)x$start)
    )
toc <- Sys.time()
print(toc-tic)
print(m.out[1,])
```

Profile likelihood

Confidence intervals for parameters of interest were generated via profile likelihood, in which the likelihood is maximized across a fixed range of values for the parameter of interest while estimating all other parameters (Hilborn and Mangel 1997). Code to generate the profile for η is below.

```
CORES<-14 ##update for flux
require(doParallel)
registerDoParallel(CORES)

source("Daphnia.R")

estpars <- setdiff(names(params.init),c("eta"))

theta.t <- partrans(Daphnia_pomp,params.init,"toEstimationScale")

theta.t.hi <- theta.t.lo <- theta.t
theta.t.lo[estpars] <- theta.t[estpars]-log(2)
theta.t.hi[estpars] <- theta.t[estpars]+log(2)

profileDesign(
  eta=seq(from=-3,to=-2,length=50),
  lower=theta.t.lo,upper=theta.t.hi,nprof=100
) -> pd

dim(pd)

pd <- as.data.frame(t(partrans(Daphnia_pomp,t(pd),"fromEstimationScale")))

bake("eta-profile1.rds",{

  foreach (p=iter(pd,"row"),
           .combine=rbind,
           .errorhandling="remove",
           .inorder=FALSE,
           .options.mpi=list(chunkSize=1,seed=1598260027L,info=TRUE)
  ) %dopar% {

    tic <- Sys.time()

    require(magrittr)
    require(plyr)
    require(reshape2)
    require(pomp)

    options(stringsAsFactors=FALSE)
    dat<-subset(zoopcomplete,select=c("D.mendotae","day"))
```

```
    dat %>%
      pomp(
  times="day",
  t0=0,
  params=params.init,
  rprocess = euler.sim(step.fun = Daphnia_rprocess, delta.t=1),
  rmeasure= Daphnia_rmeasure,
  dmeasure = Daphnia_dmeasure,
  covar=covartable,
  tcovar="time",
  obsnames = c("D.mendotae"),
  zeronames = c("error_count"),
  statenames = c("prey","noise","error_count"),
  paramnames = c("sigmaa","sigmab","log.betaz1","kappa","alpha","mu","eta","s
dbeta","mean_rho","sigma_rho","prey.0","noise.0","error_count.0"),
  covarnames = c("seas.1","byth","nocoup","pulse"),
  fromEstimationScale=Daphnia_trans,
  toEstimationScale=Daphnia_untrans
        ) %>%
      mif2(start = unlist(p),
           Nmif = 75,
           rw.sd = rw.sd(sigmaa=0.02,sigmab=0.02,
                         log.betaz1=0.02,log.betaz2=0.02,
                         log.betaz3=0.02,log.betaz4=0.02,
                         mu=0.02,mean_rho=0.02,
                         sigma_rho=0.02,alpha=0,
                         kappa=0.02,sdbeta=0.02,
                         prey.0=ivp(0)),
           Np = 2000,
           cooling.type = "geometric",
           cooling.fraction.50 = 0.1,
           max.fail=200,
           transform = TRUE) %>%
      mif2() -> mf

    ## Runs 10 particle filters to assess Monte Carlo error in likelihood
    pf <- replicate(10, pfilter(mf, Np = 2000))
    ll <- sapply(pf,logLik)
    ll <- logmeanexp(ll, se = TRUE)
    nfail <- sapply(pf,getElement,"nfail")

    toc <- Sys.time()
    etime <- toc-tic
    units(etime) <- "hours"

    data.frame(as.list(coef(mf)),
               loglik = ll[1],
               loglik.se = ll[2],
```

```
                nfail.min = min(nfail),
                nfail.max = max(nfail),
                etime = as.numeric(etime))
  }
}) -> eta_prof
```

Note on reproducibility

To enhance the reproducibility of this work, this appendix was generated using Rmarkdown (https://rmarkdown.rstudio.com/). An advantage of Rmarkdown is that it shows the development of the model combining the mathematical model specification, model code and data analysis code in a single, reproducible document.

References

Hilborn, R., and M. Mangel. 1997. *The Ecological Detective: Confronting Models with Data*. Princeton University Press, Princeton, New Jersey, USA.