Confronting Ecological Models with Data

Bio534: Fundamentals of Ecological Modeling

1 Learning Objectives

By the end of class, you should be able to do the following:

- Identify the three major points where data enters the modeling process;
- Understand how to use parametric methods to estimate parameter rates;
- Distinguish between model calibration and evaluation (validation); and
- Incorporate forcing data into ordinary differential equation models.

2 Introduction: Three Points of Contact

In general, modeling has many characteristics in common with *inverse problems*: we are in part deducing system properties and structures, parameter values, and underlying principles from the data. There are at least three points where data is used to guide model development.

1. **Model Construction**: Selecting processes, mathematical formulations, and identifying parameters (calibration). Model building is essentially a search for the model(s) that best {explain, predict} system behavior. This contact can occur at two levels of organization: (1) individual processes or parameters, and (2) whole model (model *calibration*).
2. **Model Evaluation**: Determine how well the model performs on independent data (model *validation*). Recall that Ellner and Guckenheimer (2006) noted that $prediction.error = model.error + parameter.error$.
3. **Forcing Data**: Exogenous data that drives or forces system dynamics.

3 Model Construction

Again there are two levels at which we can use data in model construction. We will first discuss the parameter fitting and function selection at the process level and then consider whole model calibration.

3.1 Process Level

For each process hypothesized to occur in the model:

1. Choose a mathematical function to model the process based on knowledge of domain theory, process, mathematical functions and data;
2. **Identify** parameter values
   (a) Estimate or Guess-timate
   (b) Literature Estimates
   (c) Fit to data
      i. By hand
      ii. Optimization Routine (e.g., Ordinary Least Squares)
3. Build model process by process

Ordinary Least Squares

As described in Ellner and Guckenheimer (2006), *ordinary least squares* is a common technique for estimating parameters is to fit process functions to process data. Using this approach, we first select a mathematical function and then try (as in trial and error) different parameter values. We then compare the *predicted* values to the *actual* values by calculating the sum of squared errors shown in equation (1).

$$SSE = \sum_{i=1}^{n} (a_i - p_i)^2 \tag{1}$$

We use this calculation in an iterative fashion with objective to find the parameter values that minimize SSE. Most statistical packages have optimization routines to automatically implement this. In following example I will show you how to do this with the *nls* function in R .

Connections The sum of squared error (a.k.a. sum of squared deviation) calculation is the heart of the most common method to describing variation. As presented in equation (1), the actual value a_i is the *expected* value $E(x)$ for the observations. In different applications, the expected value might be different. For example when we are calculating standard deviation (of the mean), the expected value is the mean ($\bar{X}$) value of the x_i observations (predicted), such that

$$SD = \sqrt{\frac{\sum_{i=1}^{n} \left(\bar{X} - x_i\right)}{(n-1)}}. \tag{2}$$

To complete the calculation, the modified SSE term is divided by the degrees of freedom to get a metric of variation normalized to the relative sample size, and then we take the square root of the result to get the answer in units that match the original observations. Note the similarity of the formulas for SEE and SD and the root mean squared error (RMSE) that we have used previously.

Example: Photosynthesis vs. Irradiance

Algal photosynthetic production depends on the quantity of light reaching the organisms in two ways. An increase in light intensity stimulates production until some threshold intensity is achieved and it begins to inhibit production (photoinhibition). This is seen clearly in the example laboratory data shown in Figure 1.

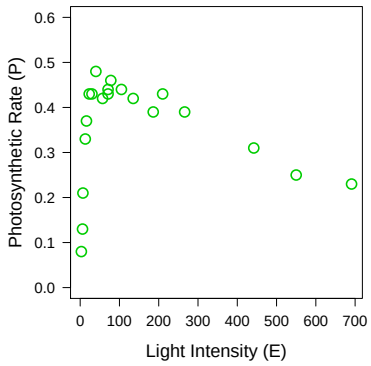

Figure 1: Example data for the relationship between photosynthesis (P) and light irradiance (E).

What mathematical function could we use to model this relationship? Here we will consider three alternative models for this process. The first model is based on the **theoretical relationship between photosynthetic rate (P) and irradiance (E)** in which the photosynthetic rate initially increases at a rate of α until the photosynthetic machinery becomes saturated and the maximum rate of photosynthesis P_{max} is achieved. This relationship is often described in the literature by the following equation:

$$P = P_{max} - e^{(-\alpha E/P_{max})} \tag{3}$$

The second model adds **photoinhibition** to the first:

$$P = P_{max} - e^{(-\alpha E/P_{max})} e^{-(\beta E/P_{max})} \tag{4}$$

where β is the slope of the decline in photosynthesis due to photoinhibition.

Alternatively, we could use a **modified Monod** function.

$$P = \underbrace{a * \frac{E}{E+k}}_{growth} * \underbrace{e^{(-E/b)}}_{photoinhibition} \tag{5}$$

where a scales the photosynthetic response, b is the light intensity where photoinhibition occurs, and k is the half-saturation constant of light intensity.

The next step is to *identify* the optimum parameter values for these functions. Specifically, our task is to find a function f that minimizes the difference between the observed and predicted values given the data (x_i, y_i) $i = 1, 2, \ldots, n$. We can do this task by hand, but it is often convenient to use optimization algorithms like least squares that minimizes the sum of squared errors

$$SSE = \sum_{i=1}^{n} (y_i - f(x_i))^2, \tag{6}$$

which is encoded in R in the *nls* function. I used the *nls* function to fit the models in equations (3), (4) and (5). The estimated parameter values are shown in Table 1 and the estimated lines are shown in Fig. 2. A word of caution. If you are going to use these powerful numerical methods, you should learn more about them. Unfortunately, this material is beyond the scope of this class.

Which function would you select to use in your model? Why? We might choose the function that generates the lowest SSE. However, we know that mathematically it usually becomes easier to fit data as we include more parameters, so our *model selection* might

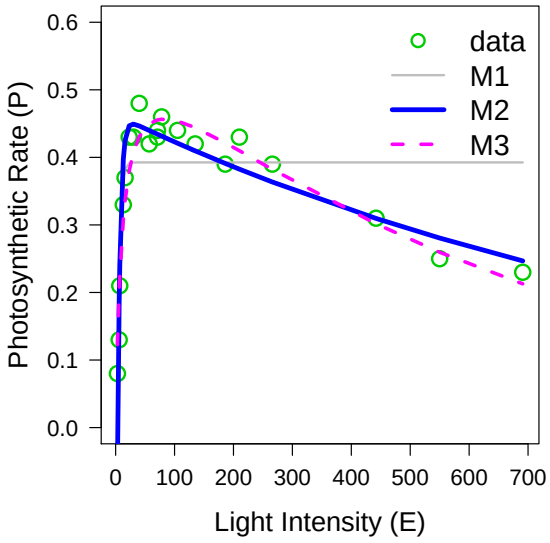

Figure 2: Example of fitting alternative process models to laboratory data for the relationship between photosynthesis (P) and light irradiance (E).

be biased toward functions with more parameters. This is problematic because we introduce more error and uncertainty into the model with each parameter we add. Thus, two widely used criteria for model selection add a penalization for adding parameters. These functions are the Akaike Information Criteria (AIC) and Bayesian Information Criterion, which are calculated as:

$$AIC = N \log(SSE/N) + 2p \quad (7)$$

$$BIC = N \log(SSE/N) + p \log N \quad (8)$$

where N is the number of data points and p is the number of parameters. Then, the model with the smallest AIC or BIC is preferred.

3.2 Model Calibration

Inevitably we are left with processes and parameters for which we have little or no data, or which may be unobservable. How do we derive parameter estimates for these? Model calibration is the process of fitting parameters from the whole model to observed data.

Table 1: Parameter estimates for the OLS fit of equations 3–5 to laboratory data

Parameter	Model 1	Model 2	Model 3
P_{max}	0.3925	0.4633138	
α	0.1117	0.0994733	
β		0.000422	
a			0.58
k			11.21
b			695.66
SSE	0.1006	0.03931	0.01874
AIC	-43.08	-59.88	-74.69

We can use the same type of approach as we used for process level function selection and parameter fitting, but we are now asking: How well does the whole model prediction fit the observed data (often time series data). We then *tune* the parameters to optimize the model fit to the observed data. Notice that we can use our confidence in various parameter estimates to choose which parameters to systematically alter (changing the ones we know the least about).

4 Model Evaluation

A second place we use data is to evaluate the quality of our models by comparing their predictions to empirical observations. For a true evaluation, we must use a data set that is *independent* from the data we used to initially estimate our parameters and do model calibration. Sometimes this modeling step is called model *validation*, but this has led to a number of interesting arguments about the possibility of validating (confirming) a model. We will discuss more about this next week.

5 Forcing Data

Up to this point, we have largely considered models comprised of *autonomous* ordinary differential equations in which the model dynamics were **not** forced by exogenous variables. For many ecological models this is unrealistic because the dynamics are forced by variables operating outside of the modeled system. The model of phytoplankton production discussed in Section 3.1 provides a clear example. Light irradiance is a time dependent variable driving the phytoplankton production.

There are two ways to incorporate exogenous variables of this type into our model. We can either

1. explicitly model the variable as a function of time, or
2. we can use empirical observations of light intensity.

In the first case, we must select another function and estimate its relevant parameters and then encode this as a time function. Jørgensen and Bendoricchio (2001; Fundamentals of Ecological Modeling) consider how to do this in depth. The second case presents a different kind of challenge. When solving our differential equations, we usually integrate our models using a time step that is different than the sampling frequency, but we need a value at each step. The solution is to interpolate the forcing data, effectively making them a function of time. I illustrate the second approach in the following example.

Example: Producer(algae)-Consumer(zooplankton)

For this example, consider a system composed of a zooplankton species that feeds upon an algal species that we will model with the following equations:

$$\frac{dP}{dt} = \mu * P - \delta_1 * P - \Phi(P, Z) \quad (9)$$

$$\frac{dZ}{dt} = -\Phi(P, Z) - \delta_2 * Z \quad (10)$$

Where μ is the realized growth rate of P, δ_1 and δ_2 are natural mortality rates, and $\Phi(P, Z)$ is the predation function. In this model, we will let μ be a function of both light intensity (E) and temperature (T).

$$\mu = \mu_{max} * f(E) * f(T) \tag{11}$$

where

$$f(E) = 1 - e^{(-\alpha E/P_{max})} * e^{(-\beta E/P_{max})} \tag{12}$$

$$f(T) = e^{0.06933*T} \tag{13}$$

and the parameters are defined as before.

E and T are forcing variables for which we will use empirical measurements. To do this, we will need to *interpolate* the variables so that we can treat them as functions of time. We do this in R using the *approxfun* or *splinefun* functions. The first methods performs a linear interpolation between the data points. The latter method fits a cubic spline–piecewise cubic polynomial–to the data. I will demonstrate this in class and a copy of my code is in the appendix.

6 Appendix

6.1 Parameter Estimation

```
#---------------------------------------
# P vs E Parameter Estimation
# Stuart R. Borrett
# 22 Oct. 07
#---------------------------------------
# Load and Plot Laboratory Data
t <- 1:20; # time
E <- c(3,6,7,13,16,23,30,40,57,71,71,78,
       105,135,186,210,266,442,550,691); # Irradiance
P <- c(0.08,0.13,0.21,0.33,0.37,0.43,0.43,
       0.48,0.42,0.44,0.43,0.46,0.44,0.42,
       0.39,0.43,0.39,0.31,0.25,0.23); # Photosynthesis
dat <- data.frame(t,E,P);

sw.pdf=0  # turn on/off pdf file
sw.plot=1 # turn on/off plot of model fits.


if(sw.pdf == 1) pdf(file="../results/peCurves.pdf", width=5, height=5);
#if(sw.pdf == 1) pdf(file="../results/PECurve1.pdf", width=5, height=5);

opar<-par(las=1,mar=c(5,5,1,1),oma=c(0,0,0,0),cex.axis=1.2, cex.lab=1.4)

plot(dat$E,dat$P,pch=1,col=3,
  ylab="Photosynthetic Rate (P)",xlab="Light Intensity (E)",
  ylim=c(0,0.6),cex=1.75,lwd=2)

# exponential light model
#P<-(a-exp(-E/Emax))*exp(-1*E/b);

# -- FIT Non-linear Models using nls ---
m0 <- nls((P~(pmax-exp(-1*alpha*E/pmax))),
          data=dat, start=list(pmax=0.5,alpha=0.3))

m1 <- nls((P~(pmax-exp(-1*alpha*E/pmax)) * exp(-1 * beta * E / pmax)),
          data=dat, start=list(pmax=0.5,alpha=0.3,beta=0.001))

m2 <- nls( (P~(a * E/(E+k))*exp(-1*E/b)), data=dat, start=list(a=1,k=20,b=200))

# extract coefficients
p0 <- coefficients(m0)
p1 <- coefficients(m1)
p2 <- coefficients(m2)


if(sw.plot == 1){
# -- Create Plot ---
points(E,(p0[1]-exp(-p0[2]*E/p0[1])),
       type="l",lwd=2,lty=1,col=8)
```

```
points(E,(p1[1]-exp(-p1[2]*E/p1[1]))* exp(-1*E*p1[3]/p1[1]),
       type="l",lwd=4,lty=1,col=4)

points(E,(p2[1] * E/(E+p2[2]))* exp(-1*E/p2[3]),
       type="l", lwd=3, lty=2, col=6)

legend("topright",legend=c("data","M1","M2","M3"),lty=c(NA,1,1,2),
       lwd=c(2,2,4,3),col=c(3,8,4,6),pch=c(1,NA,NA,NA),bty="n",cex=1.5)
}

rm(opar)
if(sw.pdf == 1) dev.off()
```

6.2 Simple Producer(algae)-Consumer(zooplankton) model

Model

```
# ------------------------------------------------------------------------
# PZ1 -- model
# Borrett Bio534
# it illustrates the use of exogenous data (i.e., Diff.Eq. are non-autonomus)
# ------------------------------------------------------------------------

model<-function(t,state,parameters){
  with(as.list(c(state,parameters)),{

# forcing data
    temp<-rsp.temp(t);             # temperature, calls function
    E<-rsp.PURe(t);                # incident light, calls function

# parameters


# auxilary equations
    Tcf <- exp(0.06933 * temp);                  # temperature control function
    Lcf <- (1 - exp(-1 * alpha1 * E/Pmax)) * exp(-1 * beta1 * E/Pmax);   # light cf

    u <- umax * Tcf  *Lcf;   # realized growth rate
    phi1<- gmax * X1 * X2; # predation functional response

# equations (ODE)

    dX1 = X1*(u - delta1) - phi1; # phyto
    dX2 = gamma1 * phi1 - delta2 * X2; # zoo
    return(list(c(dX1,dX2),c(Tcf,Lcf)))

  })
}
```

6.2.1 Run Code

```
# ------------------------------------------------------------------------
# PZ1 -- Run File
```

```
# Borrett, 22 Oct. 2007
# updated: Oct. 2015
#---------------------------------------------------------------------
rm(list=ls())
library(deSolve)

## === INPUT ===

#source("PZ1.r") # loads model function

# --- Model Function - Phytoplankton & Zooplankton ----

model<-function(t, state, parameters){
  with(as.list(c(state, parameters)),{

# forcing data
    temp <- rsp.temp(t)             # temperature, calls function
    E <- rsp.PURe(t)                # incident light, calls function

# auxilary equations
    Tcf <- exp(0.06933 * temp)                  # temperature control function
    Lcf <- (1 - exp(-1 * alpha1 * E/Pmax)) * exp(-1 * beta1 * E/Pmax)   # light cf

    u <- umax * Tcf  * Lcf   # realized growth rate
    phi1 <- gmax * X1 * X2  # predation functional response

# equations (ODE)

    dX1 = X1*(u - delta1) - phi1 # phyto
    dX2 = gamma1 * phi1 - delta2 * X2 # zoo
    return(list(c(dX1,dX2),c(Tcf,Lcf)))

  })
}

# Set Initial Values and Simulation Time
states = c(X1 = 10.8,
           X2 = 0.1)               # initial values
times = seq(92, 479, by = 0.05)    # vector of times for which you want a solution

# Assign Parameter Values

parameters = c(umax = 0.1,
    alpha1 = 0.4633138,
    Pmax =   0.4633138 ,
    beta1 =  0.000422 ,
    delta1 = 0.03  ,
    gmax  =  0.01   ,
    gamma1 = 0.1  ,
    delta2 = 0.05)

# --- FORCING DATA ---
# load forcing data and convert it into time functions.  This is the key element
# of this example.  The objective is to show how to convert discrete time
```

```
# measurements into a continuous time function.

dat2 <- read.table("../data/rsp-yr1.data", header=T)  # import data
rsp.day <- dat2$d                            # slice-out day information



## === ACTION ===

# Create Functions to Interpolate Data Observations as Needed
rsp.temp <- splinefun(rsp.day, dat2$temp)    # create spline function for data
rsp.PURe <- splinefun(rsp.day, dat2$PURe)    # create spline function for data

# Numerically Solve the Equation(s)
out = ode(states, times, func = model, parms = parameters, method="rk4")


## === OUTPUT ===
## PLOT

## -- plot forcing data
sw.plot=1                   # switch to control plotting behavior: 1 = plot, 0 = not

if(sw.plot == 1){
    # this creates a plot of the data with 2 different y-axes.
    d <- dat2$d
    PUR <- dat2$PURe
    t1 <- dat2$temp
    opar <- par(las = 1,
                oma = c(2,2,1,2),
                mar = c(2,2,1,2),
                mfrow = c(2,1))
    plot(d, PUR,
         type = "p", col = 4, lwd = 4,
         axes = F,          # prevents the plotting of the axes
         xlab = "",ylab = "",
         xlim = c(min(d), 500))
    axis(1) # adds x-axis
    axis(2) # adds y-axis
    box()   # adds plot box
                                        #
    opar <- par(new = T)  # lets us plot ontop of the existing plot on a different scale
    plot(d,t1,
         type = "p", col = "purple", lty = 2, lwd = 4,
         axes = F,                # suppresses axis creation
         xlab = "", ylab = "")
    axis(4)  # adds second y-axis on the right-hand side
    legend("topright",c("PUR","Temperature"),col=c(4,"purple"),
         lwd=4,lty=c(1,2),bty="n")
    # add lables
    mtext("days", side=1,outer=T,line=0)
    mtext("PUR (umol photos m-2 s-1)",side=2,outer=F,line=3,las=0)
    mtext("Temperature(Celsius)",side=4,outer=F,line=2,las=0)
    rm(opar)
```

```
}

## -- Plot Model Results --
plot(out[,1],out[,2],
     type = "l",
     xlab = "days", ylab = "concentration", lwd = 3,
        col = "green", ylim = c(0, 1.1*max(out[,2:3])),
        )
points(out[,1],out[,3],
       type = "l", lwd = 3, col = "red")
legend("topright",c("Phyto", "Zoo"),
       col = c("green","red"),
       lwd = c(3, 3), bty = "n", cex = 1)
```