Introduction to Maximum Likelihood Estimation and Template Model Builder

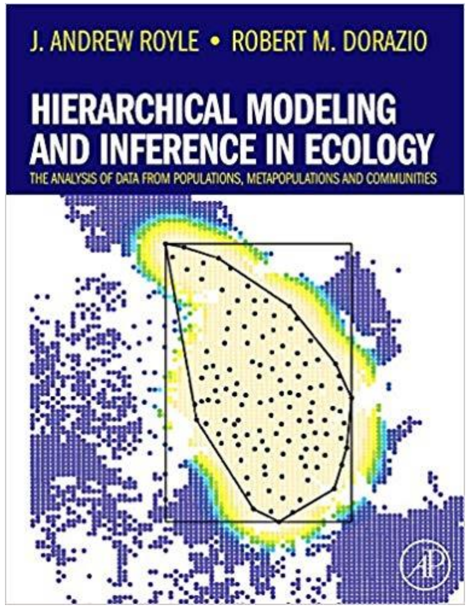
Christopher L. Cahill

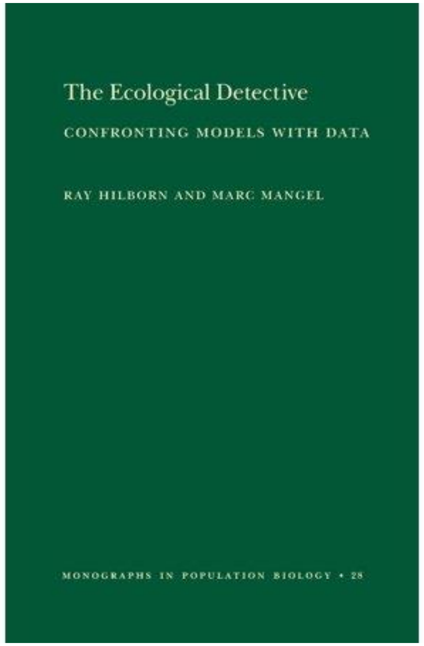
This workshop is focused on three principles of model building:

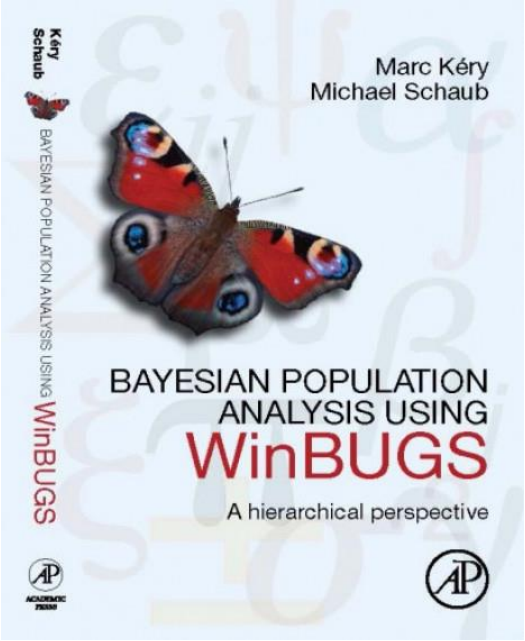
1. Formulation
    1. Building models from first principles using biologically meaningful parameters
    2. Likelihoods
2. Implementation
    1. Fitting your (or your supervisor's) model to data using Template Model Builder (TMB)—the bulk of this course focuses on this as it is perhaps the most difficult for ecologists
    2. Likelihoods
3. Evaluating your (potentially) fancy model to make sure it isn't boondoggled
    1. Residual diagnostics
    2. Cross validation
    3. Posterior predictive simulation (For mixed models)
    4. Likelihoods (via likelihood profiling techniques)

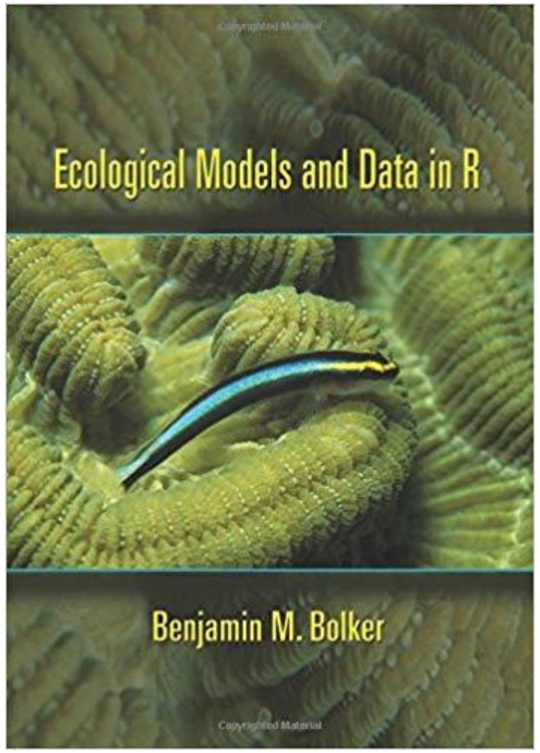
Useful Resources

I like books, particularly about statistics. Here are some good ones:









Obviously, impossible to cover all topics in detail in this class. Unfortunately, none of these books even mention Template Model Builder

What does Template Model Builder do?

- Parameters are hypotheses describing nature
- TMB finds the hypotheses that best describe your data
- TMB can find the value of the parameter vector that minimizes a (potentially very complex) nonlinear function of many variables
- It can compute measures of uncertainty via
  - Asymptotic variance-covariance matrices
  - Computing likelihood profiles for parameters and model outputs
  - Delta-method (fancy way of using Taylor Series approximations to obtain variance estimates of *any* derived quantities
  - Sampling parameter vectors from Bayesian posteriors using Hamiltonian Monte Carlo via tmbstan() –not covered in this course, but very powerful

Why use TMB?

- All the cool kids use TMB (i.e., bandwagon)
- It is fast because the minimization algorithm uses automatic differentiation (programming voodoo)
- No need to supply the derivatives of the function to be minimized with respect to the parameters—these are computed automatically (thank Our Father Below 😈)
- TMB code is basically C++ but includes:
  - Helper macros to support a variety of data structures and parameters, some R-style probability distributions, and commands to specify the function to be minimized
- TMB uses the Laplace approximation for random effect computations—it is very fast (relative to most programs)
- If you can dream the likelihood and have data, TMB can probably estimate it. The most powerful nonlinear optimizer available?

Automatic Differentiation: https://en.wikipedia.org/wiki/Automatic_differentiation

Goals for today

- Introduce maximum likelihood and some models
- Fit some models with maximum likelihood in R and in TMB
- Start the miserable TMB learnin'-journey
- Keep tears to a minimum

Maximum Likelihood

- Given a set of data and a probability model, maximum likelihood chooses values for the parameters that make the data the “most likely”

$$L(D|\theta) \approx P(D|\theta)$$

- The likelihood of the data *D* given parameter(s) θ is proportional to the probability of the data given the parameter(s)

- Probability is knowing parameters and predicting data
- Likelihood is knowing data and estimating parameters
- So we need a way to specify this miserable likelihood function

Maximum Likelihood

- If there are two data points D_1 and D_2, the total likelihood L of both data points is the product of the likelihoods l of each point:

$$L(\boldsymbol{D}|\theta) = l(D_1|\theta)l(D_2|\theta)$$

- Likelihoods are really small. i.e., 1e-4 or something silly because they represent an infinitesimally small sliver of a distribution
- How can we deal with this?

$$\log\big(L(\boldsymbol{D}|\theta)\big) = log\big(l(D_1|\theta)\big) + log(l(D_2|\theta))$$

- Typically use negative log-likelihood

Why use Maximum Likelihood?

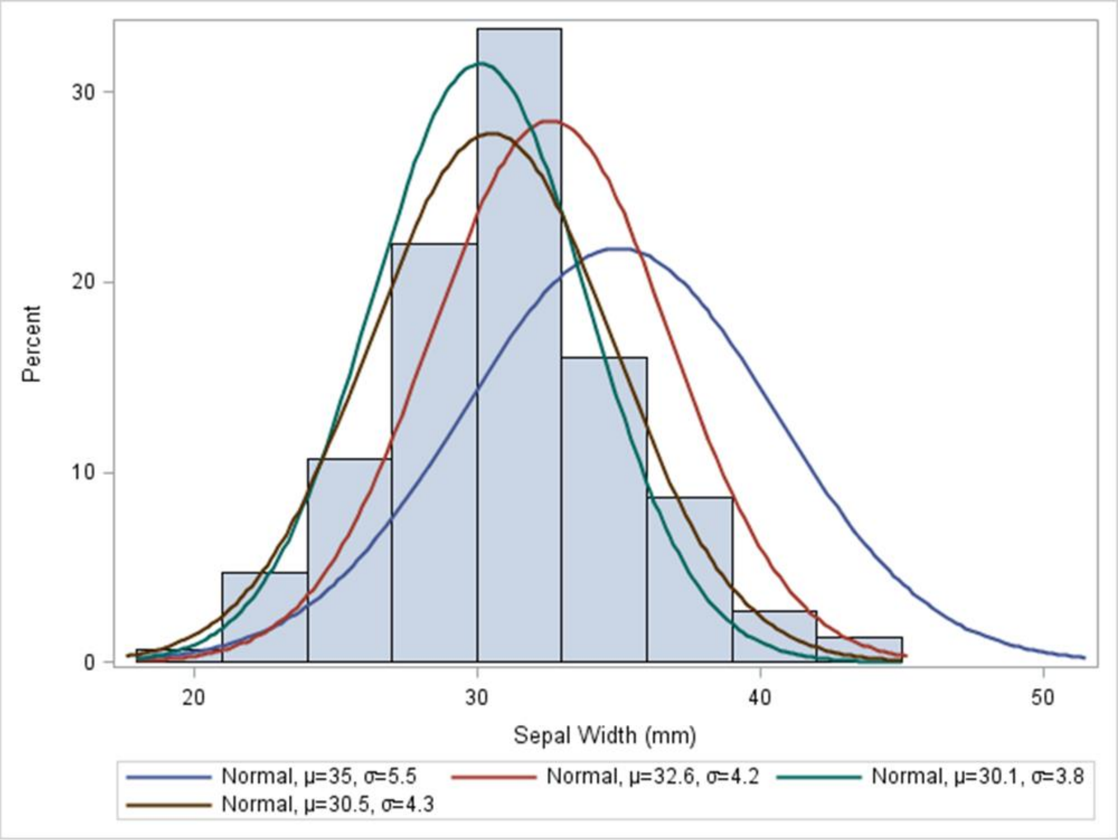
$\hat{\boldsymbol{\theta}} = \text{argmax}_{\boldsymbol{\theta}}(\log(L(\boldsymbol{\theta}; \mathbf{y})))$

$\mathbf{y}$ is a vector of data points

$\hat{\boldsymbol{\theta}}$ is the maximum likelihood estimate of unknown parameters $\boldsymbol{\theta}$

$L(\boldsymbol{\theta}; \mathbf{y})$ is your specified probability distribution

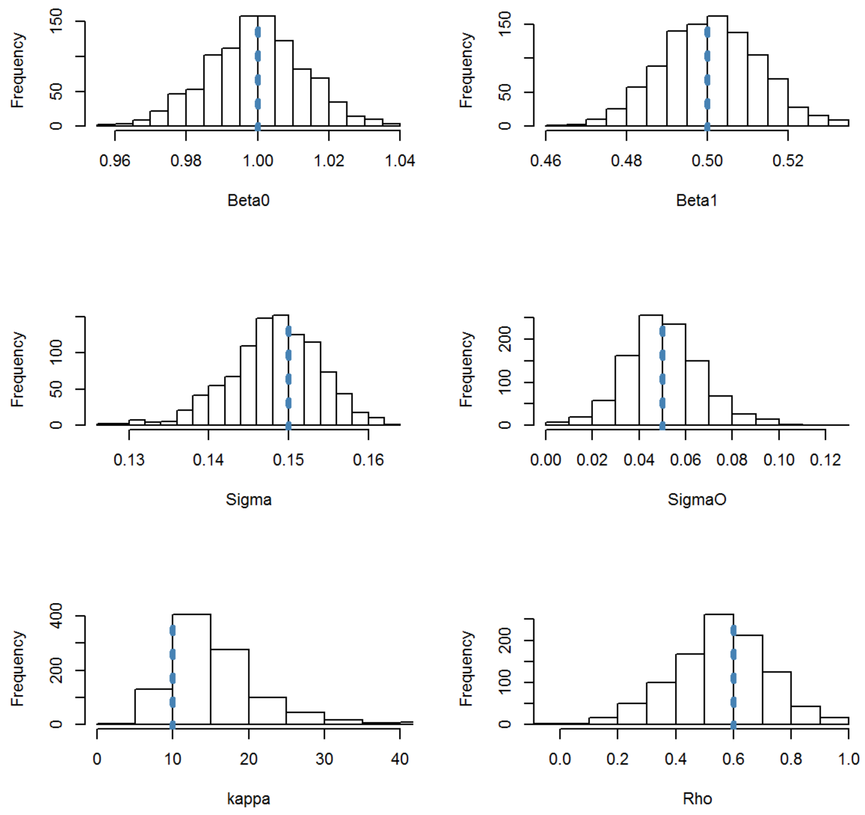
- Consistency (correct model)
  $\hat{\boldsymbol{\theta}} \rightarrow \boldsymbol{\theta}$ as $n \rightarrow \infty$
- Consistency (incorrect model)
  $\hat{\boldsymbol{\theta}} \rightarrow \boldsymbol{\theta}_{\text{optimal}}$ as $n \rightarrow \infty$
- Asymptotic normality
  - Allows one to calculate confidence intervals
  - If you repeat your experiment many times, true value of $\boldsymbol{\theta}$ will fall within your confidence intervals x% of the time (coverage)
- Invariance to reparameterization



See section 2.3.2 in Royle and Dorazio 2008

Implications

- If you have a simulation and the model used to simulate data is identical to the model used to estimate parameters
  - Estimated parameters will be perfect on average with large sample sizes
  - Total error will go to zero with large sample sizes
- If you have a simulation and your estimation model doesn't match the simulation model
  - Estimated parameters will converge on values with large sample sizes
  - Total error will decrease to an asymptote
- Gives a pretty solid way to test whether your code is behaving as expected



Distribution of MLEs vs. Truth from Simulation

Even More Likelihood Considerations

- To compute the likelihood for a given set of parameters we need to specify:
  - The deterministic (model) relationship between the input variables (i.e., covariates or parameters) and the expected (data):

$$y_i = \beta_0 + \beta_1 x_1 + \epsilon_i$$

  - And how the data relate to the model prediction—the "sampling distribution" for the data:

where $\epsilon_i \sim \mathrm{N}(0, \sigma)$

  - Linear regression with normally distributed data
  - Data vs. parameters
- Maybe we are interested in this how this terrible beast's adult weight varies as a function of some environmental parameter

Show the videos of the BFGS vs. Nelder-Mead (with TMB) nonlinear optimizer searches that took a really long time to create

How's this TMB thing work

- Write a .cpp template file that specifies your model given some parameters and data
- Compile the model & pray it doesn't explode
- Dynamically link the TMB template file to R
- Construct an objective function with derivatives based on the C++ file and pass this to a nonlinear optimizer in R
    - nlminb()
- Estimate the model
- I do this all in R-Studio

Some tips

- C++ indexing starts at 0 and not 1
- Vectors, matrices and arrays are **not** zero-initialized in C++ (= misery)

	R	TMB	
Comments	#	//	// Comment symbol
Constants	3.4	Type(3.4);	// Explicit casting recommended in TMB
Scalar	x = 5.2	Type x = Type(5.2);	// Variables must have type
Arrays	x = numeric(10)	vector<Type> x(10);	// C++ code here does NOT initialize to 0
Indexing	x[1]+x[10]	x(0)+x(9);	// C++ indexing is zero-based
Loops	for(i in 1:10)	for(int i=1;i<=10;i++)	// Integer i must be declared in C++
Increments	x[1] = x[1] + 3	x(0) += 3.0;	// += -= *= /= incremental operators in C++

- See Kasper Kristensen's intro to TMB https://kaskr.github.io/adcomp/_book/Introduction.html