

Q^FC Quantitative Fisheries Center

Welcome to our second (advanced)
course on using AD Model Builder

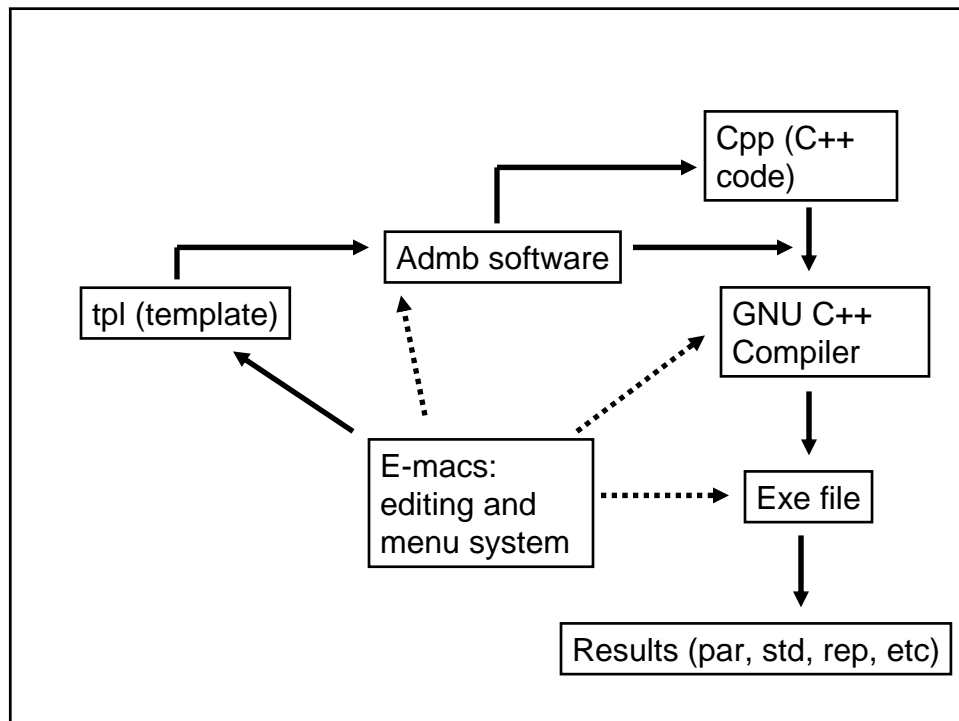
- Instructors Jim Bence and Brian Linton
- Special thanks to Travis Brenden, Kendra Porath, Mike Wilberg, and Weihai Liu
- And to MSU, GLFC, MDNR and CLC Agency Partners

Preliminary Issues

- Please wear your name tags (have pity on my poor memory...)
- Coffee etc in room 153
- Change in schedule: Tonight we plan to meet from 6:30-9:00 and offer to organize a Pizza Dinner in room 153 during the break
- Course materials (revised based on workshop) will be available within two weeks at an ftp site.
- Possibility of group purchase of admb software.
- Do you have a parking pass on your vehicle?

Course Overview

- Session 1 (this morning): Course introductory material, using the computing system, assessing uncertainty using admb
- Session 2 (this afternoon): Using functions and doing simulations
- Session 3 (tonight): Hands on development of simulation code
- Session 4 (tomorrow morning): Random effects in admb
- Session 5 (tomorrow afternoon): Misc. topics and catchup



Welcome to Emacs

- Von Bertalanffy growth model
- Navigating and editing in Emacs
- ADMB in Emacs
 - Growth model example

Von Bertalanffy Growth Model

$$\tilde{L}_a = \overset{\substack{\text{Asymptotic length} \\ \downarrow}}{L_\infty} \left(1 - e^{-\overset{\substack{\text{Age of length 0} \\ \downarrow}}{K(a - \overset{\substack{\text{Age of length 0} \\ \downarrow}}{t_0})}} \right) e^{\varepsilon_a}$$

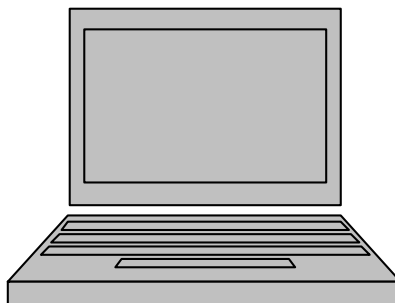
Brody growth coefficient

$$\varepsilon_a \sim N(0, \sigma_\varepsilon^2)$$

Concentrated Negative Log Likelihood

$$-conc \log(l) = \frac{n}{2} \log \sum_{a=1}^n \left[\log(\tilde{L}_a) - \log(L_a) \right]^2$$

Now to learn about Emacs
`growth.tpl`



ADMB Methods of Assessing Uncertainty

- Asymptotic standard errors
 - Of parameters (produced by default)
 - Of derived quantities
- Likelihood profile method
- MCMC (Bayesian posterior distribution)
- All admb methods require minimizing the negative log-likelihood (or something related to it)

Review on probability distributions

- Likelihood depends on assumed probability distribution for the data summarized by $f(x)$
- $f(x)$ represents either a probability density function (pdf continuous distributions) or probability mass functions (pmf) for discrete distributions

$$\int_{-\infty}^{+\infty} f(x)dx = 1.0 \text{ or } \sum f(x) = 1.0$$

- Joint pdf/pmf for multiple independent observations

$$\begin{aligned} f(\underline{x}) &= f(x_1, x_2, x_3, \dots, x_k) = \\ &f(x_1)f(x_2)f(x_3)\cdots f(x_k) = \\ &\prod_1^k f(x_i) \end{aligned}$$

More probability distribution stuff

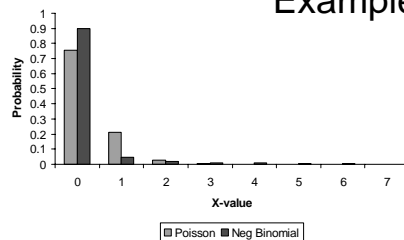
- Previous slide presumed that joint pdf/pmf was for multiple independent observations from the same probability distribution.
- Similar result when observations come from different distributions.

$$f(\underline{x}, \underline{y}) = f(x_1, \dots, x_k, y_1, \dots, y_m) =$$

$$f_X(x_1)f_X(x_2)f_X(x_3)\cdots f_X(x_k), f_Y(y_1), \dots, f_Y(y_m) =$$

$$\prod_{i=1}^k f_X(x_i) \prod_{i=1}^m f_Y(y_i) = f_{\underline{Y}}(\underline{x}) \cdot f_{\underline{Y}}(\underline{y})$$

Example PDF/PMFs

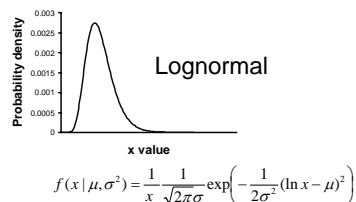
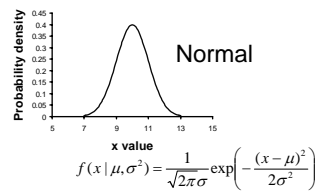


Poisson

$$f(x) = e^{-\lambda} \frac{\lambda^x}{x!}$$

Negative Binomial

$$f(x | m, s) = \frac{\Gamma(x+s)}{\Gamma(s) x!} \left(\frac{s}{m+s} \right)^s \left(\frac{m}{m+s} \right)^x$$



Mathematically, likelihood function is same as pdf/pmf – But expressed as function of parameters

- Implicitly (and sometimes explicitly) pdf/pmf is conditional on parameter values

$$f(\underline{x}) = f(\underline{x} | \underline{\theta})$$

- Sometimes likelihood function is expressed as conditional on data

$$L(\underline{\theta}) = L(\underline{\theta} | \underline{x}) = f(\underline{x} | \underline{\theta})$$

The negative log-likelihood

- Log transform the likelihood (joint pdf) and remember that $\log(a*b)=\log(a)+\log(b)$

$$-\log L(\theta) = -\log f(\underline{x}, \underline{y}) = -\log f_X(\underline{x}) - \log f_Y(\underline{y}) = -L_X - L_Y$$

$$L_X = \log f_{\underline{X}}(\underline{x}) = \log f_X(x_1) + \log f_X(x_2) + \dots + \log f_X(x_k)$$

-log Likelihood functions

Negative Binomial

$$-\log L(m, s) = -\log(\Gamma(x+s)) + \log(\Gamma(s)) + \log(x!) \\ - s \log\left(\frac{s}{m+s}\right) - x \log\left(\frac{m}{m+s}\right)$$

Normal

$$-\log L(\mu, \sigma^2) = \frac{n}{2} \ln 2\pi + n \ln \sigma + \frac{1}{2\sigma^2} \sum (x_i - \mu)^2$$

Lognormal

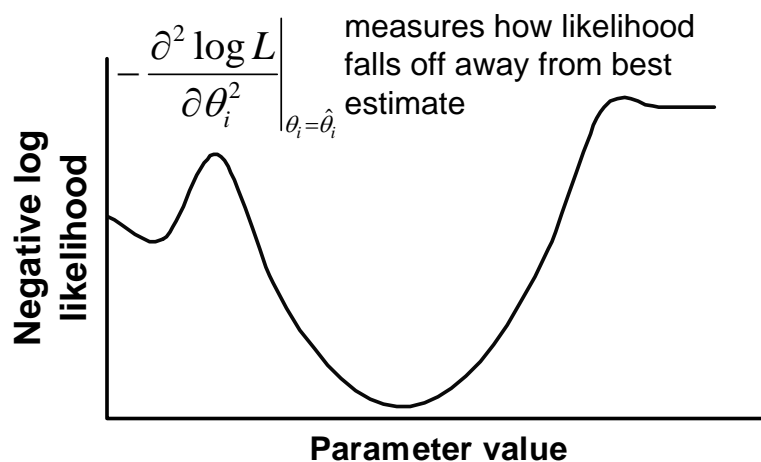
$$-\log L(\mu, \sigma^2) = \sum \ln(x_i) + \frac{n}{2} \ln 2\pi + n \ln \sigma + \\ \frac{1}{2\sigma^2} \sum (\ln x_i - \mu)^2$$

Poisson

$$-\log L(\lambda) = n\lambda - \lambda \sum x_i + \sum \ln x_i!$$

Another form of Normal

$$-\log L(\mu, \sigma^2) = \frac{n}{2} \ln \sigma^2 + \frac{1}{2\sigma^2} \sum (x_i - \mu)^2 + IC$$



Asymptotic results

- Done automatically for parameters ADMB or for any calculating quantity of type sdreport_* or likeprof_number
- Results are in *.std and *.cor
- These are based on:

$$\Sigma = -H^{-1}$$

$$h_{ij} = \frac{\partial^2 \log L(\underline{\theta})}{\partial \theta_i \partial \theta_j}$$

- The variance-covariance matrix:

$$\Sigma = \begin{bmatrix} \sigma^2_{11} & \sigma^2_{12} & \dots & \sigma^2_{1j} & \dots & \sigma^2_{1p} \\ \sigma^2_{21} & \sigma^2_{22} & \dots & \sigma^2_{2j} & \dots & \sigma^2_{2p} \\ \dots & \dots & \dots & \dots & \dots & \dots \\ \sigma^2_{i1} & \sigma^2_{i2} & \dots & \sigma^2_{ij} & \dots & \sigma^2_{ip} \\ \dots & \dots & \dots & \dots & \dots & \dots \\ \sigma^2_{p1} & \sigma^2_{p2} & \dots & \sigma^2_{pj} & \dots & \sigma^2_{pp} \end{bmatrix}$$

Diagonal elements are variances of parameter estimates, off-diagonals are covariances among parameter estimates.

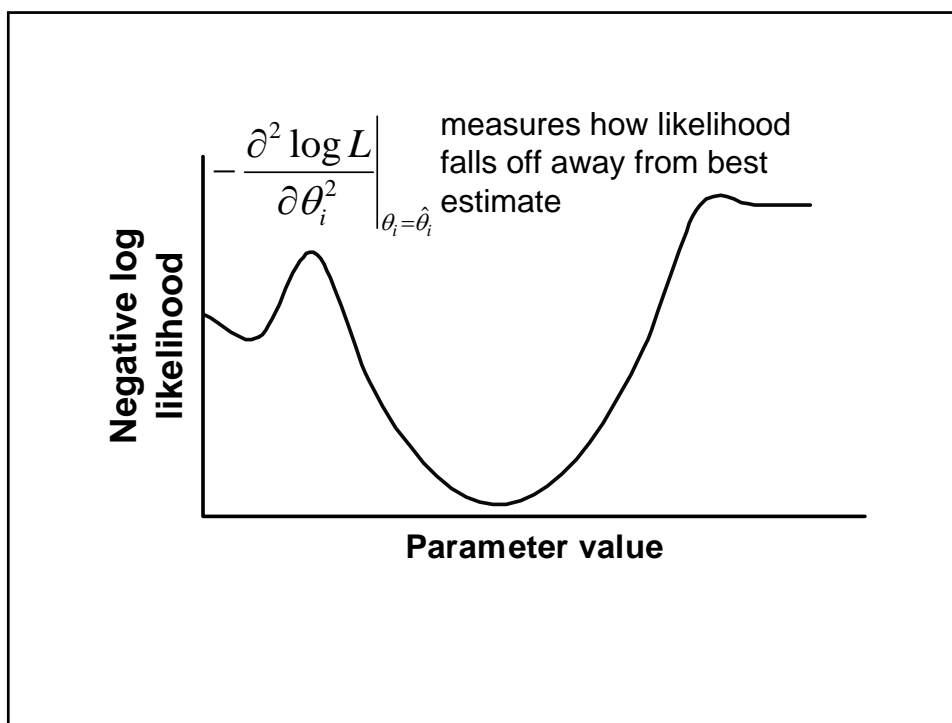
Elements of variance-covariance matrix

- The variances describe uncertainty in the parameter estimates. I.e., how variable are these estimates about their true values?
- The square-root of the variances gives the standard errors
- The covariances describe how the estimation errors for two parameters are related. When parameter “a” is over-estimated does parameter “b” also tend to be over-estimated (+ cov), tend to be under-estimated (- cov) or is there no relationship (0 cov)?

Correlation matrix

- Diagonals are 1.0
- Off diagonals are correlations among parameter estimates:

$$\rho_{ij} = \frac{\sigma^2_{ij}}{\sqrt{\sigma^2_{ii}} \sqrt{\sigma^2_{jj}}}$$



Asymptotic standard errors can produce misleading inferences

- When sample sizes are small
- and the curvature of the likelihood surface changes substantially within the range of plausible estimates – i.e., “near to the maximum likelihood estimates
- We will now explore alternative approaches to assessing uncertainty using a surplus production model fit to arrowtooth flounder data
- First we review the model and data

Overview of model and data

- There are 14 years of data consisting of yield (mass) and a biomass index.
- The biomass index is based on swept area trawl and catchability (q) is assumed known=1
- Dynamics are assumed to follow logistic model in absence of fishing and observed yield is assumed to equal true yield
- Observed biomass indices are assumed to have a lognormal distribution.

Surplus Production Model (one simple variant)

$$B_{t+1} = B_t + \Delta_B(B_t, Y_t)$$

$$\Delta_B = \frac{4m}{B_\infty} \left(1 - \frac{B_t}{B_\infty} \right) B_t - Y_t$$

Observation Stochasticity with lognormal errors

$$Bobs_t = qB_t\varepsilon_t \quad \hat{Bobs}_t = qB_t$$

$$\ln(Bobs_t) = \ln(q) + \ln(B_t) + \ln(\varepsilon_t)$$

So our objective function will be to minimize the neg log likelihood
with additive constants dropped:

$$-\log L(\mu, \sigma^2) = n \ln \sigma + \frac{1}{2\sigma^2} \sum (\ln x_i - \mu)^2$$

OK – enough already!

Lets look at the code, make sure it runs, and
then move on to implementing alternative
approaches to assessing uncertainty!

Find flounder folder and open up founder.dat
and flounder.tpl in buffers

Things to do

- Create tpl, compile and link
- Look at results (rep file, cor file)
- Change Inblast to be of type likeprof_number
- Run with “switch” -lprof

How to use the profile method

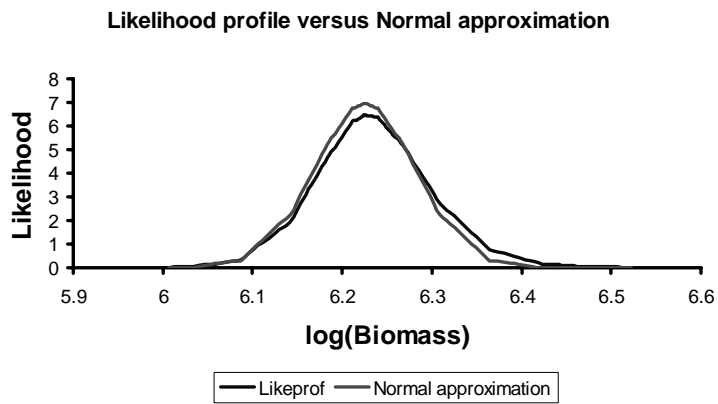
- Declare a variable you would like to profile as type likeprof_number in the parameter section, and assign it the correct value in the procedure section.
- When you run your program use the lprof switch: myprog -lprof
- Results are saved in myvar.plt where myvar is the name of your likeprof_number variable
- Your variable is varied over a “profile” of values and the best fit constrained to match each value of your variable is found

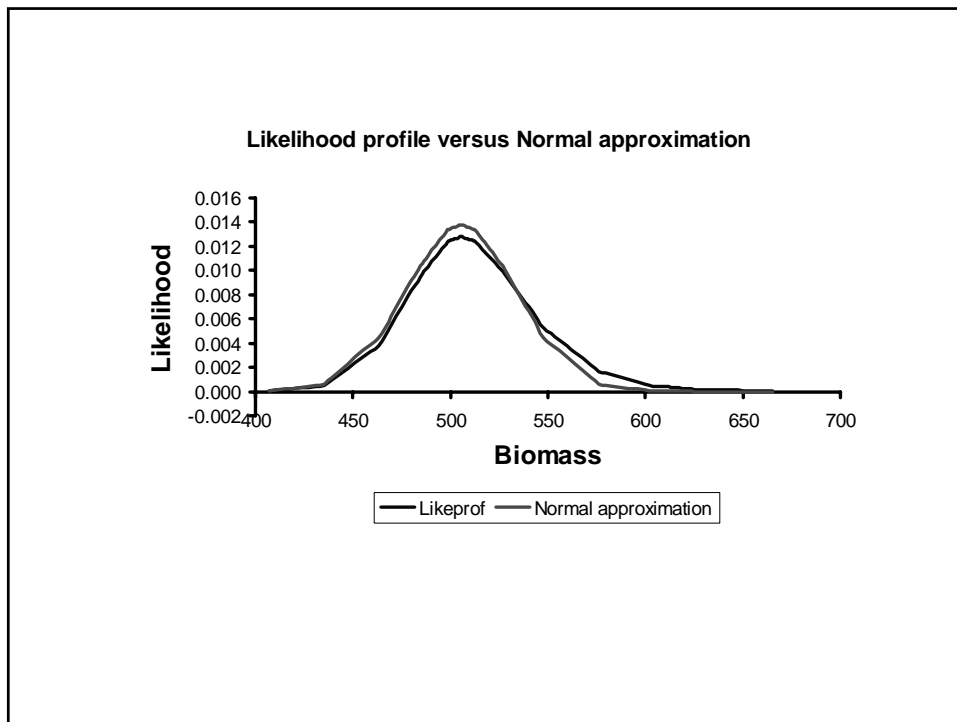
PLT File contains list of point (x,y)
x is value (say biomass)
y is associated prob density

Plot of Y vs X gives picture of prob distribution

ADMB manual says estimate probability x in in (x_r, x_s) by

$$\sum_{i=r}^s (x_{i+1} - x_i) y_i$$





Profile likelihood options

Switch

-prsave This saves the parameter values associated with each step of profile in myvar.pvl

Options set in tpl (preliminary calcs section): e.g., for lprof var myvar:

PRELIMINARY_CALCS_SECTION

```
myvar.set_stepnumber(10); // default is 8  
myvar.set_stepsize(0.2); //default is 0.5
```

Note manuals says stepsize is in estimated standard deviations but this appears to be altered adaptively during the profile

WARNING -- LOTS OF STEPS CAN TAKE LOTS OF TIME!

Profile Likelihood Method

- This is NOT inverting a likelihood ratio test in ADMB land!
- This is Bayesian in philosophy (in the same way that MCMC is). Can also be motivated by likelihood theory (support intervals)
- Idea is to use the profile for $g(\theta)$ to approximate the probability density function for g .

When $g(\theta) = \theta_1$ -- i.e, we are interested in the distribution of a parameter -- ADMB approximates the marginal distribution of θ_1 :

$$\int f(\theta_1, \dots, \theta_n) d\theta_2 d\theta_3 \dots d\theta_n$$

with

$$\lambda \max[f(\theta \mid \theta_1 = \theta_0)]$$

More generally $g(\theta)$ (say biomass) is a complex function of many parameters and we want a pdf for $g(\theta)$

- The approximation needs to be modified:
 - ✓ biomass (or other derived quantities) will change at different rates with respect to changes in parameters in different parts of the parameter space.
 - ✓ i.e., some “biomasses” might represent a small part of the parameter space and others might represent a larger part.

The modified approximation is:

$$\lambda \frac{\max[f(\theta_1, \dots, \theta_n) \mid \theta : g = g_0]}{\|\nabla g(\hat{\theta}_1, \dots, \hat{\theta}_n)\|}$$

PLT File contains list of point (x,y)
 x is value (say biomass)
 y is associated prob density

Plot of Y vs X gives picture of prob distribution

ADMB manual says estimate probability x in in (x_r, x_s) by

$$\sum_{i=r}^s (x_{i+1} - x_i) y_i$$

MCMC

- MCMC is a way to generate samples from a complex multivariate pdf.
- In practice the pdf is usually the posterior in a Bayesian analysis.
- This is useful in looking at marginal distributions of derived quantities.
- These marginal distributions are the same thing the profile likelihood method was approximating.

We need priors to do this!

The diagram shows the formula for the posterior probability density function: $p(\theta | X) = \frac{L(X, \theta)p(\theta)}{\int L(X, \theta)p(\theta)d\theta}$. Arrows point from the words 'Likelihood' and 'Prior' to the terms $L(X, \theta)$ and $p(\theta)$ in the numerator, respectively. An arrow points from the word 'Posterior' to the entire fraction.

$$p(\theta | X) = \frac{L(X, \theta)p(\theta)}{\int L(X, \theta)p(\theta)d\theta}$$

$$\ln(p(\theta | X)) = \ln(L(X | \theta)) + \ln(p(\theta)) + \mathbf{Constant}$$

ADMB presumes we are going to start by finding the parameters that maximize the posterior density (called highest posterior or modal estimates), so just minimize the log posterior. Just like a negative log-likelihood but with new terms for priors

Two examples

- If prior on M were log-normal with median of 0.2 and with sd for $\ln(M)=0.1$, then just add to your likelihood:

$$0.5 \left(\frac{\ln(M / 0.2)}{0.1} \right)^2$$

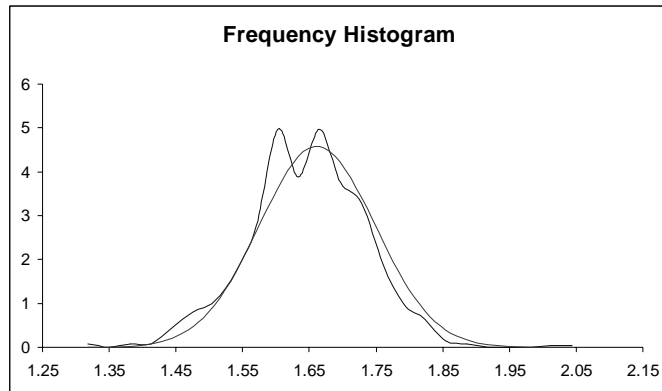
- For special case of diffuse prior $\ln(p())$ is constant inside the bounds, so a bounded diffuse prior can be specified just by setting bounds on parameters.

The Basic Idea Behind MCMC

- Transition from θ_i to θ_{i+1} is stochastic
- Transition only depends upon θ_i
- For suitable transition rule $f(\theta_i)$ approaches target pdf
- Approximate marginal pdf by output of chain after burn-in

$\theta_0 \rightarrow \theta_1 \rightarrow \theta_2 \rightarrow \dots \rightarrow \theta_{100000}$

Log of ratio of final biomass to initial biomass



For MCMC with 100,000 steps, every 100 saved,
100 of saved steps dropped as "burn-in"

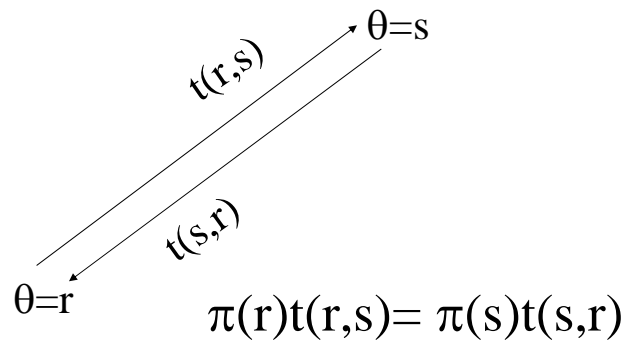
Doing an MCMC run

- Use -mcmc N switch to generate a chain of length N. Default N is 100,000.
- Summarized output for parameters, sdreport variables and likeprof_numbers is in *.hst
- This automatic summary is for the entire chain with no provision for discarding a burn-in and no built in diagnostics.
- Serious evaluation of the validity of the MCMC results requires you gain access to the chain values.

Gaining access to chain values

- When you do the MCMC run, add the switch -mcsave N, which saves in a binary file every Nth values from the chain.
- You can rerun your program to read in the saved results and make one run through your model for each saved set of parameters. Use the switch -mceval
- You can add code to your program to writeout results (and do special calculations) during the mceval phase.
- You can modify your program to do this even after you generate your chain, provided your change does influence the posterior density.

Reversibility

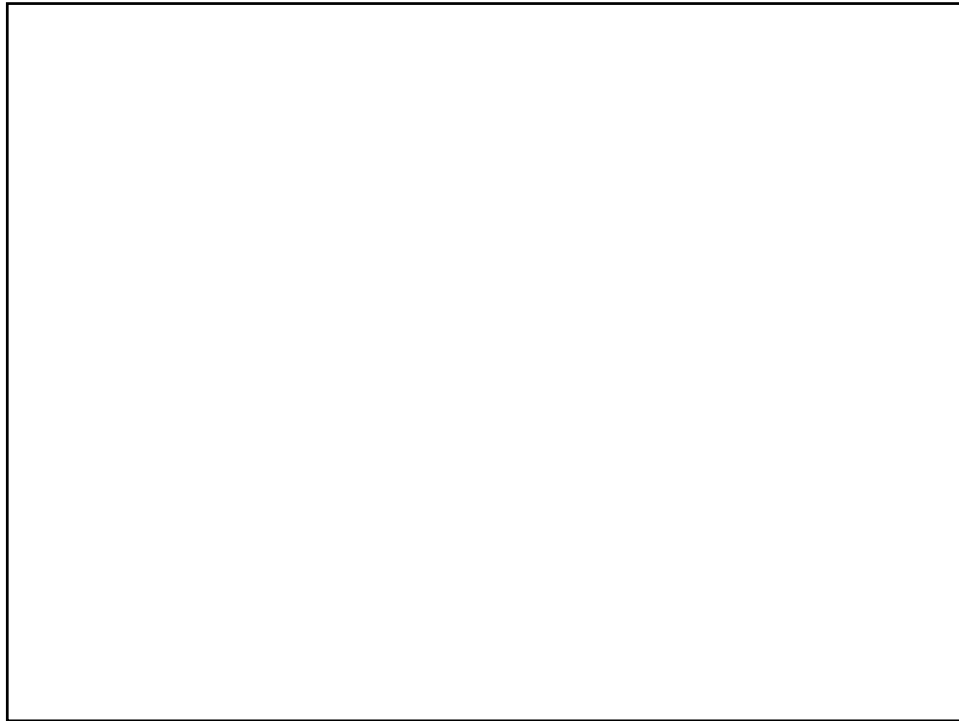


Approach to Achieve Reversibility (Metropolis algorithm)

- Generate a trial value of θ^* using “candidate” probability distribution
 - Candidate distribution is easy to simulate
 - Here we assume candidate distributions propose symmetric transitions
- Calculate $\pi(\theta_i)$ and $\pi(\theta^*)$
- If $\pi(\theta^*) > \pi(\theta_i)$ make the transition
 $\theta_{i+1} = \theta^*$
- If $\pi(\theta^*) < \pi(\theta_i)$ generate $u \sim \text{uniform}(0,1)$
 - If $u < \alpha$ make the transition
 $\theta_{i+1} = \theta^*$
 - otherwise stay put
 $\theta_{i+1} = \theta_i$
 - $\alpha = \pi(\theta^*) / \pi(\theta_i)$

ADMB Implementation of MCMC

- θ_0 = mode of posterior by default
- $\theta^* = \theta_i + \delta$, where $\delta \sim N(0, c\Sigma)$
- c is scaled so that $0.2 < \alpha < 0.4$ during first 2000 steps
- Options allow you to modify the distribution of δ and the value of θ_0



Example of code to write results out when using mceval switch

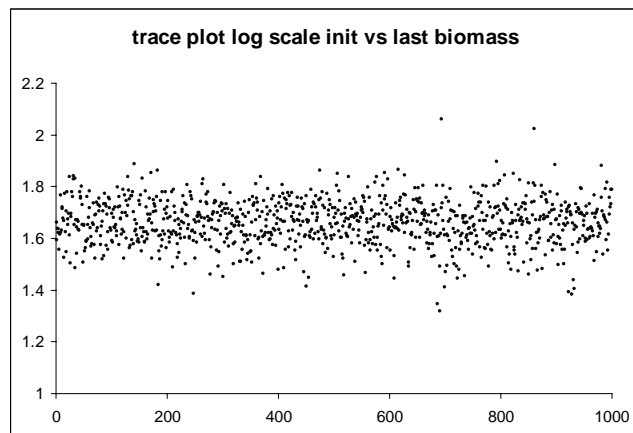
```
if (mceval_phase()) cout << negLL << " " << Blast  
    << B << endl;
```

Important caution: this writes to standard output. Better redirect this to file our millions or numbers will go scrolling by!

Some basic diagnostics

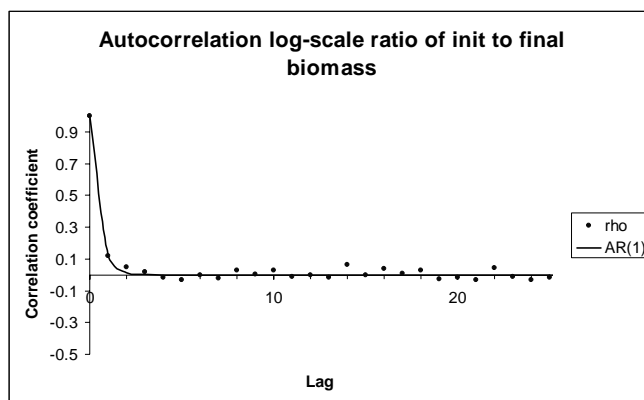
- Look at trace plot
- Look at autocorrelation function for chain
- Calculate “effective sample size”
- Compare subchain CDFs (if the first and second half differ substantially then chain may be too short)
- Lots of other diagnostics and procedures
 - E.g., parallel chains and formal comparisons

Trace plot Flounder Example

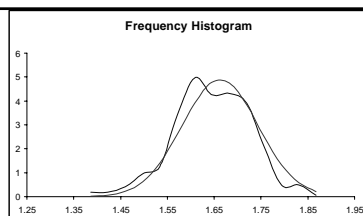


100,000 steps, sampled every 100

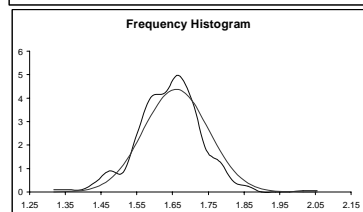
Autocorrelation for flounder example



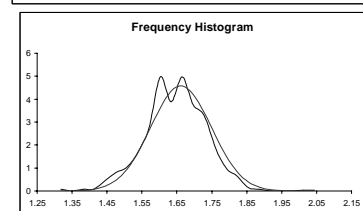
AR(1) shown for comparison (curve)



First half



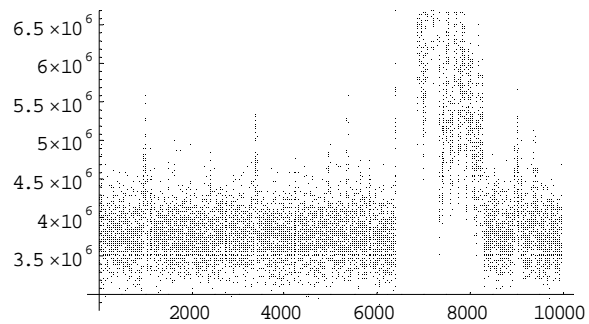
Second half



Entire chain

burn-in excluded

Trace Plot for Flack Lake



1,000,000 step chain sampled every 1000

MCMC Chain Options to make changes to transition rule

- -mcgrope p p is the proportion of “fat” tail
- -mcrb N 1 to 9, smaller = weaker correlation
- -mcdiag Hessian replaced with Identity
- -mcmult N Scaler for Hessian
- -mcnoscale No automatic adj to scaler

Starting and Restarting a Chain

- -mcr Restart from where it left off
- -mcpin fn Start chain at params in fn
 - The output obtained by running with the switches -lprof -prsave (in *.prv) can be useful for this.

Huge literature on MCMC and Diagnostics

- Gelman et al. *Bayesian Data Analysis*
good general source on all things Bayesian
- I Like Cowles and Carlin, Markov Chain Monte Carlo convergence diagnostics: a comparative review. JASA 91:833-904

Do mcmc with growth model

- Modify growth model to
 - Include at least one `sdreport_*` variable
 - Cout things you care about during `mcmceval_phase`
 - Bound all parameters
 - Change to neg log like from conc
- Tpl2cpp, compile and link your program
- Run... `-mcmc 1000000 -mcsave 100`
- Run... `-mceval>mymcmcfile.dat`

What to expect this afternoon

- Improving model efficiency
- Creating functions that take arguments
- Using control programs that automate model fitting
 - Introduction to time-varying growth model
- Simulating data to test model
- Combining control and data simulation programs in a simulation study
 - Introduction to catch-at-age model

Improving Efficiency

- You do not need to worry about model efficiency in most cases
- In general, it is only important when:
 - Your model is very complex
 - You are running your model many times (e.g., mcmc, simulation study)

Rule number 1 – calculate something only once if you can!

- Quantities that do not change but are needed during estimation should be calculated in PRELIMINARY_CALCS_SECTION
- Quantities that are not needed for estimation but only for reporting should be calculated in REPORT_SECTION or if uncertainty estimates are needed conditional on phase too:
- If (sd_phase())
 {...
 }

Rule number 2 – avoid unneeded loops

- Use admb built in functions (e.g., sum, rowsum, element by element multiplication and division, etc)
- Combine loops over the same index

Functions that take arguments

- Functions that do not take arguments can be used to organize code
`get_catch_at_age();`
- Functions that take arguments can simplify calculations
`rss=norm2(residuals);`
- Beware of functions that take parameters as arguments

Functions that take arguments

`w=my_function(my_argument);` ← Call function

← Returns object

FUNCTION `double my_function (double x)`

↑ Takes object as argument

`double y;`

..... ← Declare local variables

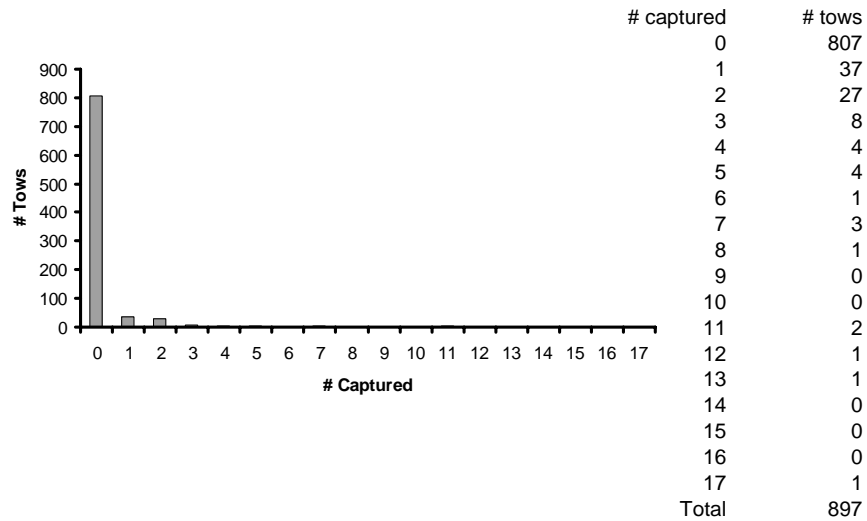
`z=x+y;`

..... ← Carry out calculations

`return(z);`

← Output variable

Albatross bycatch in New Zealand squid trawl fishery
 From Chapter 4 of Ecological Detective (Hilborn and Mangel 1997),
 originally published by Bartle (1991, cited in H&M)



Negative binomial PMF and neg log likelihood for a single observation

$$f(x | m, s) = \frac{\Gamma(x + s)}{\Gamma(s) x!} \left(\frac{s}{m + s} \right)^s \left(\frac{m}{m + s} \right)^x$$

$$-\log L(m, s | x) = -\log(\Gamma(x + s)) + \log(\Gamma(s)) + \log(x!)$$

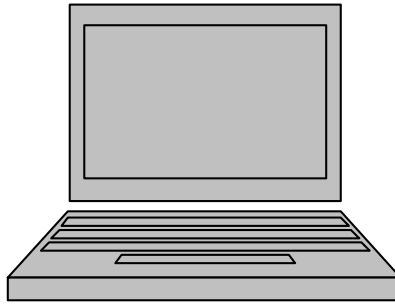
$$-s \log \left(\frac{s}{m + s} \right) - x \log \left(\frac{m}{m + s} \right)$$

**Do NOT worry about stuff below
 (included for completeness)**

$$\Gamma(x) = \int_0^{\infty} e^{-u} u^{x-1} du$$

$$\Gamma(i) = (i - 1)! \quad i \text{ an integer}$$

Now for factorial function example
albatross.tpl



Control Programs

- A control program can be used to automate repeated running of models
 - Fitting same model to multiple dat files
 - Fitting multiple models to same dat file
 - Fitting multiple models to multiple dat files
- Avoid having to move tpl and dat files around
- Manage output from multiple model runs

Tpl file must still contain:

- Data section
- Parameter section
- Procedure section
- `objective_function_value`
- One active parameter

Control Programs

- Most of work is done in preliminary calcs section or using `local_calcs` command
 - Operations involved only need to be run once
- Use “Run –est” to run control program
 - No parameters or asymptotic standard errors to estimate

Time-Varying Von Bertalanffy Growth Model

- Asymptotic length (L_∞) varies over time
- Mean length at age-1 (L_1) and Brody growth coefficient (K) are constant over time

$$\tilde{L}_{y+1,a+1} = \left[L_{y,a} + (L_{\infty,y} - L_{y,a})(1 - e^{-K}) \right] e^{\varepsilon_{y,a}}$$

$$\varepsilon_{y,a} \sim N(0, \sigma_\varepsilon^2)$$

Random Walk

- Model time-varying asymptotic length

$$L_{\infty,y+1} = L_{\infty,y} e^{\omega_y}$$

$$\log(L_{\infty,y+1}) = \log(L_{\infty,y}) + \omega_y$$

$$\omega_y \sim N(0, \sigma_\omega^2)$$

Model Parameters

$$L_{\infty,0} \quad K \quad L_1$$

$$\omega_2, \dots, \omega_{m-1} \quad L_{1,2}, \dots, L_{1,n-1}$$

Ratio of relative variances (assumed known)

$$\lambda = \frac{\sigma_{\varepsilon}^2}{\sigma_{\omega}^2}$$

Concentrated Negative Log Likelihood

$$-conc \log(L) = \frac{mn + m - 2}{2} \log \left[\sum_{i=1}^{mn} \log \left(\frac{\tilde{L}_i}{L_i} \right)^2 + \lambda \sum_{j=1}^{m-2} \omega_j^2 \right]$$

New ADMB Stuff

- Redirecting output
- Char objects
- Character string commands
- System commands

Redirecting Output

New output command
↓
ofstream ofs("myfile.txt",ios::app);
↑
File to receive output

Append new output to file

ofs << variable << endl;

Char Objects

- Object that holds a character string

GLOBALS_SECTION

```
char x[5];
```

↖ Maximum string size

- Object should be large enough to hold desired string
- Keeping track of large strings can tax memory

Character String Commands

- Convert numerical variable (y) to char object (x)

```
sprintf(x, "%i", y);
```

- Copy string "text" to char object (x)

```
strcpy(x, "text");
```

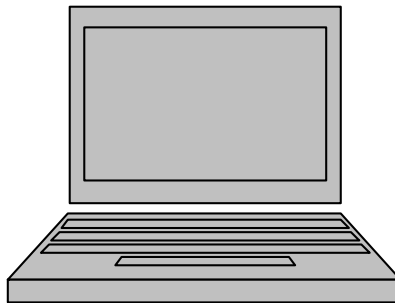
- Concatenate string "text" to char object (x)

```
strcat(x, "text");
```


System Commands

- Send commands to operating system (e.g., Windows, Linux)
- E.g., tell OS to run mymodel.exe with `-est` run-time switch
`system("mymodel -est");`

Now a control program example
`controlpro.tpl`



Control Program Exercise

- Modify controlpro.tpl to save the growthtimeC.par file from each model run to a unique file
 - E.g., growthtimeC1.par, growthtimeC2.par,...
- Use character strings to create this unique file name (as with *filename* variable) and the system command to save the file (as with *runcommand* variable)
`system("cat growthtimeC.par >> your_filename");`

Simulating Data

- Simulated data is useful for testing models
- How well does model perform when processes underlying “reality” are known?
 - The “true” values of parameters and variables can be compared to model estimates
- Make sure model works before using real world data

Simulating Data

- “Parameter” values are read in from dat file
- “Parameter” values used in estimation model equations to calculate true data
- Random number generator creates random errors
- Adding random error to true data gives observed data

How to simulate data for
time-varying growth model

Simulated data input “parameters”

$$L_{\infty,0} \quad K \quad L_I$$

$$\sigma_{\varepsilon}^2 \quad \sigma_{\omega}^2$$

Calculate ratio of relative variances

$$\lambda = \frac{\sigma_{\varepsilon}^2}{\sigma_{\omega}^2}$$

Simulate time-varying Linf

- Deviations randomly drawn from distribution

$$\omega_y \sim N(0, \sigma_{\omega}^2)$$

$$\log(L_{\infty,y+1}) = \log(L_{\infty,y}) + \omega_y$$

$$L_{\infty,y+1} = L_{\infty,y} e^{\omega_y}$$

Simulate observed mean length-at-age data

- Observation errors randomly drawn from distribution

$$\tilde{L}_{1,a+1} = \left[L_{1,a} + (L_{\infty,0} - L_{1,a})(1 - e^{-K}) \right] e^{\varepsilon_{1,a}}$$

$$\tilde{L}_{y+1,a+1} = \left[L_{y,a} + (L_{\infty,y} - L_{y,a})(1 - e^{-K}) \right] e^{\varepsilon_{y,a}}$$

$$\varepsilon_{y,a} \sim N(0, \sigma_{\varepsilon}^2)$$

New ADMB Stuff Random Number Generator

- Initialize random number generator (x)

random_number_generator x(seed);

Random number seed

- Fill object (y) with random numbers

y.fill_randn(x); // $y_i \sim \text{Normal}(0,1)$

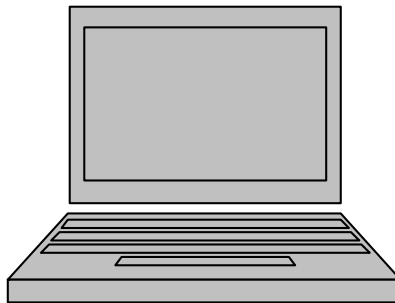
y.fill_randu(x); // $y_i \sim \text{Uniform}(0,1)$

New ADMB Stuff

Random Number Generator

- Random number generator produces pseudo-random numbers
- Pseudo-random numbers are generated from an algorithm which is a function of the random number seed
- The same random number seed will always produce the same string of numbers

Now an example of simulating data
datasim.tpl



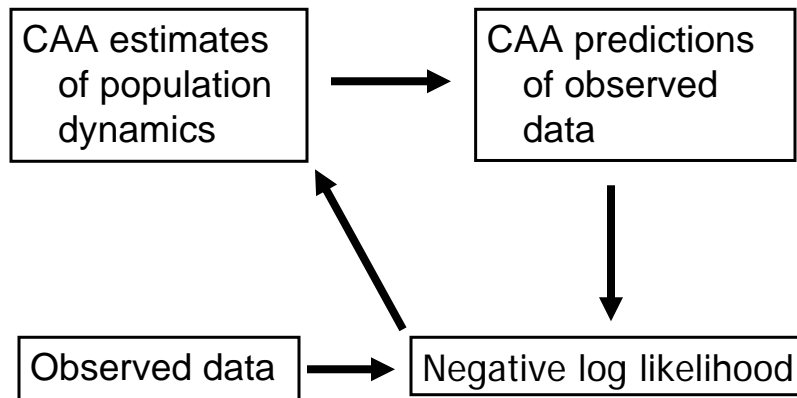
Simulation Study

- Simulation study combines a data generating model with a control program to repeatedly fit an estimating model to many simulated data sets
- This provides replicate model runs to better evaluate an estimating model's performance
 - Only one replicate normally is available in the real world

Simulation Study

- Can evaluate how a model performs vs. different underlying “reality”
 - E.g., with different levels of observation error
- Can evaluate how well different models can fit the same data sets
 - E.g., fit Ricker and Beverton-Holt stock-recruitment models to same data sets
- Or can use a combination of the two approaches

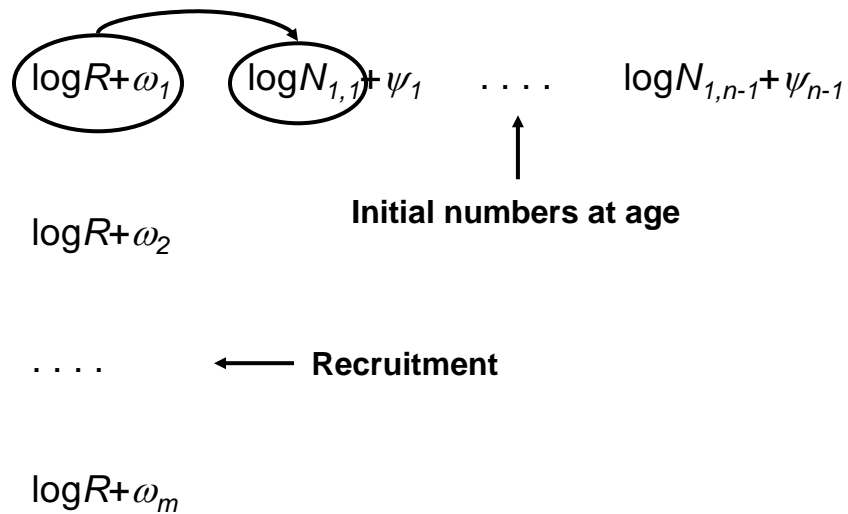
Overview of Catch-At-Age



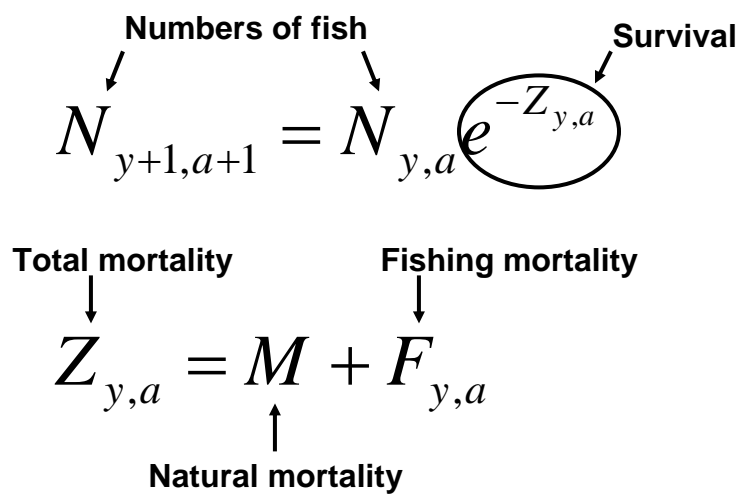
Observed Data

- Total annual fishery catch
- Proportion of catch-at-age
- Auxiliary data
 - Fishing effort

Population Submodel



Population Submodel



Population Submodel

$$F_{y,a} = s_a q E_y e^{\zeta_y}$$

Diagram illustrating the Population Submodel equation:

- Selectivity points to s_a
- Catchability points to q
- Effort points to E_y
- Effort error points to ζ_y

$$\zeta_y \sim N(0, \sigma_\zeta^2)$$

Observation Submodel

- Baranov's catch equation

$$C_{y,a} = \frac{F_{y,a}}{Z_{y,a}} \left(1 - e^{-Z_{y,a}} \right) N_{y,a}$$

Observation Submodel

Total catch

$$C_y = \left[\sum_a C_{y,a} \right]$$

Observed total catch

$$\tilde{C}_y = C_y e^{\varepsilon_y} \leftarrow \text{Observation error}$$

$$\varepsilon_y \sim N(0, \sigma_\varepsilon^2)$$

Observation Submodel

Proportion of catch-at-age

$$P_{y,a} = \frac{C_{y,a}}{C_y}$$

Numbers sampled at age

Proportions

$$\underline{n}_y = \{N_E P_{y,a}\} \sim MNOM(\underline{p} | N_E)$$

Effective sample size

$$\tilde{P}_{y,a} = \frac{n_{y,a}}{N_E}$$

Obs. proportion of catch-at-age

Negative Log Likelihood for Multinomial

$$\begin{aligned}
 -\log(L) = & -\sum_y \log\left(\frac{N_{E,y}!}{n_{y,1}!n_{y,2}!\dots n_{y,k}!}\right) \\
 & -\sum_y N_{E,y} \sum_a [\tilde{P}_{y,a} \log(p_{y,a})]
 \end{aligned}$$

Model Parameters

$$R \quad \omega_1, \dots, \omega_m \quad \psi_1, \dots, \psi_{n-1}$$

$$q \quad b_1, b_2, b_3, b_4$$

$$\zeta_1, \dots, \zeta_m \quad \sigma_\varepsilon$$

Ratio of relative variances (assumed known)

$$\lambda = \frac{\sigma_\varepsilon^2}{\sigma_\zeta^2}$$

Negative Log Likelihood (ignoring constants)

$$\begin{aligned}
 -\log(L) = & m \log(\sigma_\varepsilon) + \frac{1}{2\sigma_\varepsilon^2} \sum_y \left[\log\left(\frac{\tilde{C}_y}{C_y}\right) \right]^2 \\
 & - N_E \sum_y \sum_a [\tilde{P}_{y,a} \log(P_{y,a})] \\
 & + m \log(\sigma_\zeta) + \frac{1}{2\sigma_\zeta^2} \sum_y \zeta_y^2
 \end{aligned}$$

Data Generating Model

- Recruitments generated from Ricker stock-recruitment function

$$N_{y,1} = \overset{\substack{\text{Input "parameters"} \\ \downarrow}}{\alpha} \overset{\substack{\uparrow \\ \text{Number of spawners}}}{S_{y-1}} e^{-\beta S_{y-1}} e^{\omega_y}$$

$$\omega_y \sim N(0, \sigma_\omega^2)$$

Data Generating Model

- Numbers at age in first year came from applying mortality to randomly generated recruitments

$$N_{1,a} = N_{1-a,1} e^{-\sum_{j=1}^a Z_{0,j}}$$

$$N_{1-a,1} \sim LN(\mu_N, \sigma_N^2)$$

Data Generating Model

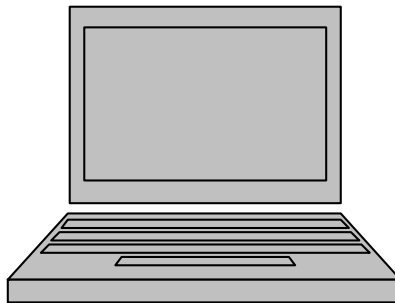
- Two treatments are examined in this simulation study
 - Low and high levels of total catch observation error
- Two values for σ_ε are specified in dat file for data generating model and one value is used to generate observed data based on the current treatment

New ADMB Stuff

Random Number Generator

- A specific random number seed will always generate the same pseudo-random numbers
- Therefore the random number seed must be changed each time create a new dat file
- Otherwise all the dat files for a given treatment will be the same

Now a simulation study example
catchsim.tpl



Simulation Study Practicum

- You have seen:
 - Control program for time-varying growth model
 - Data generating model for time-varying growth
 - Simulation study for catch-at-age analysis
- Now you need to put the pieces together to create a time-varying growth model simulation study

Simulation Study Practicum

- Your simulation study will look at the effects of process and observation error on performance of a time-varying growth model
 - Two levels of observation error for mean length-at-age
 - Two levels of process error for time-varying asymptotic length

Input “Parameters”

$$L_{\infty,0}$$

$$K$$

$$L_1$$

$$\sigma_{\varepsilon}^2$$

$$\sigma_{\omega}^2$$

Calculate ratio of relative variances

$$\lambda = \frac{\sigma_{\varepsilon}^2}{\sigma_{\omega}^2}$$

Simulate time-varying Linf as random walk

- Deviations randomly drawn from distribution

$$\omega_y \sim N(0, \sigma_{\omega}^2)$$

$$\log(L_{\infty,y+1}) = \log(L_{\infty,y}) + \omega_y$$

$$L_{\infty,y+1} = L_{\infty,y} e^{\omega_y}$$

Simulate observed mean length-at-age data

- Observation errors randomly drawn from distribution

$$\tilde{L}_{1,a+1} = \left[L_{1,a} + (L_{\infty,0} - L_{1,a})(1 - e^{-K}) \right] e^{\varepsilon_{1,a}}$$

$$\tilde{L}_{y+1,a+1} = \left[L_{y,a} + (L_{\infty,y} - L_{y,a})(1 - e^{-K}) \right] e^{\varepsilon_{y,a}}$$

$$\varepsilon_{y,a} \sim N(0, \sigma_{\varepsilon}^2)$$

Suggested “parameter” values for fullsim.dat

- | | |
|-----------------|-----------------------------------|
| • 1 to 10 years | • Low σ_{ε} 0.05 |
| • 1 to 10 ages | • High σ_{ε} 0.1 |
| • Linf 30 | • Low σ_{ω} 0.1 |
| • K 0.35 | • High σ_{ω} 0.2 |
| • L_1 9 | |

Simulation Output

- Try running 5 replicates for 20 runs total
- Output final parameter values, objective function value, and maximum gradient component for each run
 - E.g., you can use ofstream command
- Include replicate and error level numbers in output to identify model runs

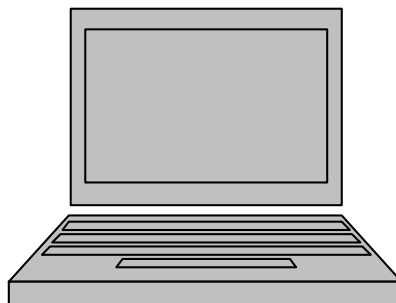
Recommendations on how to tackle this challenge

- Create data generating model (fullsim.tpl) so that it produces a dat file with same format as example.dat
- Modify growthtimeF.tpl to read in dat file correctly and make sure it runs properly
- Incorporate a control program into fullsim.tpl to automate data generation and loop over different levels of process and observation error

Recommendations on how to tackle this challenge

- Generate dat files without running growthtimeF.tpl and make sure values look reasonable
 - Remember to change random number seed for each new dat file you generate
- Try running the full simulation and check run-time message buffer to help evaluate convergence

Have fun with simulation study
fullsim.tpl



Thinking about random effects

In context of admB

Consider a model where some of the “parameters” are assumed to be random (from a distribution)

$$f(\underline{x} | \underline{\tilde{\theta}}) \quad \underline{\tilde{\theta}} = \{\underline{\theta}_1, \underline{\phi}\}$$

fixed
random

$$\underline{\phi} \sim D(\underline{\theta}_2) \quad f_{\phi}(\underline{\phi} | \underline{\theta}_2)$$

Particularly when random parameters are in blocks so that

$$\underline{\phi}_B = \{\phi_{1B}, \dots, \phi_{rB}\} \quad \phi_i \stackrel{\text{iid}}{\sim} D$$
$$f(\underline{x} | \underline{\tilde{\theta}}) \quad \underline{\tilde{\theta}} = \{\overset{\text{fixed}}{\underline{\theta}_1}, \overset{\text{random}}{\underline{\phi}}\}$$

$$\underline{\phi} \sim D(\underline{\theta}_2) \quad f_{\underline{\phi}}(\underline{\phi} | \underline{\theta}_2)$$

Particularly when random parameters are in blocks so that

$$\phi_{-B} = \{\phi_{1B}, \dots, \phi_{rB}\} \quad \text{iid} \quad \phi_i \sim D$$

Time-Varying Von Bertalanffy Growth Model

$$\tilde{L}_{y+1,a+1} = \left[L_{y,a} + (L_{\infty,y} - L_{y,a})(1 - e^{-K}) \right] e^{\varepsilon_{y,a}}$$

$$\varepsilon_{y,a} \sim N(0, \sigma_{\varepsilon}^2)$$

$$\log(L_{\infty,y+1}) = \log(L_{\infty,y}) + \omega_y$$

$$\omega_y \sim N(0, \sigma_{\omega}^2)$$

Random Walk

- Model time-varying asymptotic length

$$L_{\infty,y+1} = L_{\infty,y} e^{\omega_y}$$

$$\log(L_{\infty,y+1}) = \log(L_{\infty,y}) + \omega_y$$

$$\omega_y \sim N(0, \sigma_{\omega}^2)$$

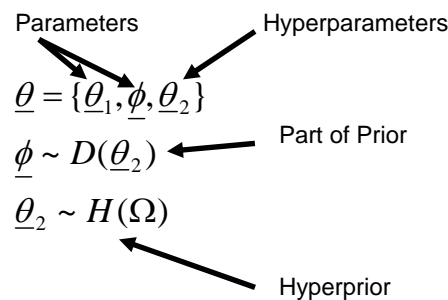
How does this connect?

- To a frequentist $\underline{\phi}$ are not parameters. Parameters are fixed.
- When we have encountered this we have been “closet Bayesians.” We estimated $\tilde{\underline{\theta}}$ and assumed that $\underline{\theta}_1$ could be determined from $\underline{\theta}_2$. This is the Highest Posterior Density approach.
- Often not possible to estimate variance of random effects doing this
- Frequentist mixed model or hierarchical Bayesian approaches are alternatives

$$f(\underline{x}|\tilde{\underline{\theta}}) \quad \tilde{\underline{\theta}} = \{\underline{\theta}_1, \underline{\phi}\}$$

$$\underline{\phi} \sim D(\underline{\theta}_2) \quad f_{\phi}(\underline{\phi}|\underline{\theta}_2)$$

Hierarchical Bayesian



$$f(\underline{x}|\tilde{\underline{\theta}}) \quad \tilde{\underline{\theta}} = \{\underline{\theta}_1, \underline{\phi}\}$$

$$\underline{\phi} \sim D(\underline{\theta}_2) \quad f_{\phi}(\underline{\phi}|\underline{\theta}_2)$$

Frequentist mixed model

$\underline{\theta} = \{\underline{\theta}_1, \underline{\theta}_2\}$ **parameters**

$\underline{\phi} \sim D(\underline{\theta}_2)$ **model for random effects**

$$f(\underline{x} | \underline{\theta}) \propto \int_{\text{all } \underline{\phi}} f(\underline{x} | \underline{\theta}_1, \underline{\phi}) p(\underline{\phi} | \underline{\theta}_2) d\underline{\phi}$$

ADMB-RE

- What is old
- What is new
 - Random effects objects
 - Objective function
 - Correlated random effects
- Tips for estimating random effects

What's Old

- In tpl file, you still need:
 - Data section
 - Parameter section
 - Procedure section
 - Declare objective_function_value
 - One active parameter
- Most (but not all) ADMB functions are also available in ADMB-RE

What's New Random Effect Objects

- Need to declare random effects and associated variance in Parameter section

init_number sigma_x

random_effects_vector x(1,nobs)

init_number sigma_y

random_effects_matrix y(1,nrow,1,ncol)

- Random effects must be declared after all the other parameters (i.e., after all the init_objects)

What's New Objective Function

- Distribution of random effects should be included in objective function
- Objective function must be the negative log likelihood or sum of negative log likelihoods
- Laplace approximation used to integrate negative log likelihood with respect to random effects
 - Approximation is less accurate when random effects are not normally distributed

What's New Correlated Random Effects

- Can estimate unstructured covariance matrix for random effects
`init_matrix cov(1,nobs,1,nobs)`
`random_effects_vector x(1,nobs)`
- ADMB-RE manual covers how to parameterize cov matrix using Cholesky factor
 - Reduces number of parameters
 - Ensures matrix is positive definite

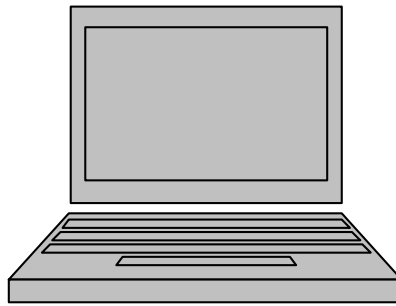
What's New Correlated Random Effects

- ADMB-RE manual covers how to `cholesky_decomp()` function to specify structured cov matrix

Tips for estimating random effects

- Estimate random effects and associated variances in later phase
 - i.e., after fixed effects parameters are well estimated
- Random effects and associated variances should be estimated in same phase
- Try estimating multiple random effects (i.e., multiple random effects objects) in different phases

Let's try out ADMB-RE
growthtimeRE.tpl



Miscellaneous Topics and Tricks

- Ragged arrays
- Missing data
- Advanced ADMB functions
- Using dat file for flexibility

Ragged Arrays

- Ragged matrix is matrix whose rows are vectors with varying valid indices
- Ragged array is an array of matrices of different sizes
 - Matrices themselves may or may not be ragged
- You can learn more about ragged arrays in the AutoDif manual

Ragged Arrays

```
int min=0; // minimum valid row index
int max=4; // maximum valid row index
ivector minind(0,4); //minimum valid index of vector
                      forming each row of matrix
ivector maxind(0,4); //maximum valid index of vector
                      forming each row of matrix
//Read in values to minind and maxind
.....
dmatrix M(min,max,minind,maxind);
```

Ragged Array

- For example, if:
 `minind(4)=-1,`
 `maxind(4)=5,`
- Then row 4 of matrix M can be thought of as:
 vector `m(-1,5)`

Missing Data

- It is not uncommon to have missing years of data in a time series of observed data
- One solution is to interpolate the missing years of data outside the model fitting process by some ad hoc method
 - E.g., averaging data from the adjacent years
- A better solution is to allow the model to predict values for the missing data
 - This takes advantage of all the available data

Missing Data Implementation

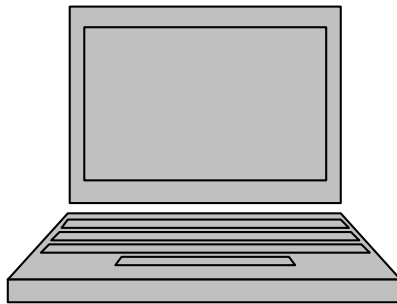
- Use special value to denote missing data in dat file
 - E.g., a value you wouldn't normally see in real data like -1
- Use loops and conditional statements to exclude missing data values from objective function value
- Otherwise, model will try to match predicted values to the missing data values

Missing Data Multinomial Case

- Replace missing data with 0 and it will not contribute to negative log likelihood value

$$-\log(L) = -N_E \sum_y \sum_a [\tilde{P}_{y,a} \log(P_{y,a})]$$

Let's look at an example
catchMD.tpl



Advanced Functions

- Filling objects
- Obtaining shape information
- Extracting subobjects
- Sorting vectors and matrices
- Cumulative density functions

Filling Objects

```
v.fill("{1,2,3,6}"); // v=[1,2,3,6]
```

```
v.fill_seqadd(1,0.5); // v=[1,1.5,2,2.5]
```

```
m.rowfill_seqadd(3,1,0.5); // fill row 3 with sequence
```

```
m.colfill_seqadd(2,1,0.5); // fill column 2 with sequence
```

```
m.rowfill(3,v); // fill row 3 with vector v
```

```
m.colfill(2,v); // fill column 2 with vector v
```

Obtaining Shape Information

```
i=v.indexmax(); // returns maximum index
```

```
i=v.indexmin(); // returns minimum index
```

```
i=m.rowmax(); // returns maximum row index
```

```
i=m.rowmin(); // returns minimum row index
```

```
i=m.colmax(); // returns maximum column index
```

```
i=m.colmin(); // returns minimum column index
```

Extracting Subobjects

```
v=column(m,2); // extract column 2 of m  
v=extract_row(m,3); // extract row 3 of m  
v=extract_diagonal(m); // extract diagonal elements of m
```

```
vector u(1,20)
```

```
vector v(1,19)
```

```
u(1,19)=v; // assign values of v to elements 1-19 of u
```

```
--u(2,20)=v; // assign values of v to elements 2-20 of u
```

```
u(2,20)=++v; // assign values of v to elements 2-20 of u
```

```
u.shift(5); // new min is 5 new max is 24
```

Sorting Objects

- **Sorting vectors**

```
w=sort(v); // sort elements of v in ascending order
```

- **Sorting matrices**

```
x=sort(m,3); // sort columns of m, with column 3 in  
              ascending order
```

Cumulative Density Functions

- For standard normal distribution

`x=cumdnorm(z); // x=p(Z<=z), Z~N(0,1)`

- Also have CDF for Cauchy distribution

`cumdcdf()`

Flexible dat files

- You can use dat file to prevent having to modify and recompile tpl file
- Quantities you can read in from dat file:
 - Object indices
 - Parameter starting values
 - Parameter bounds
 - Parameter phases
 - Switches turn code on/off

Let's look at an example
catchS.tpl

