

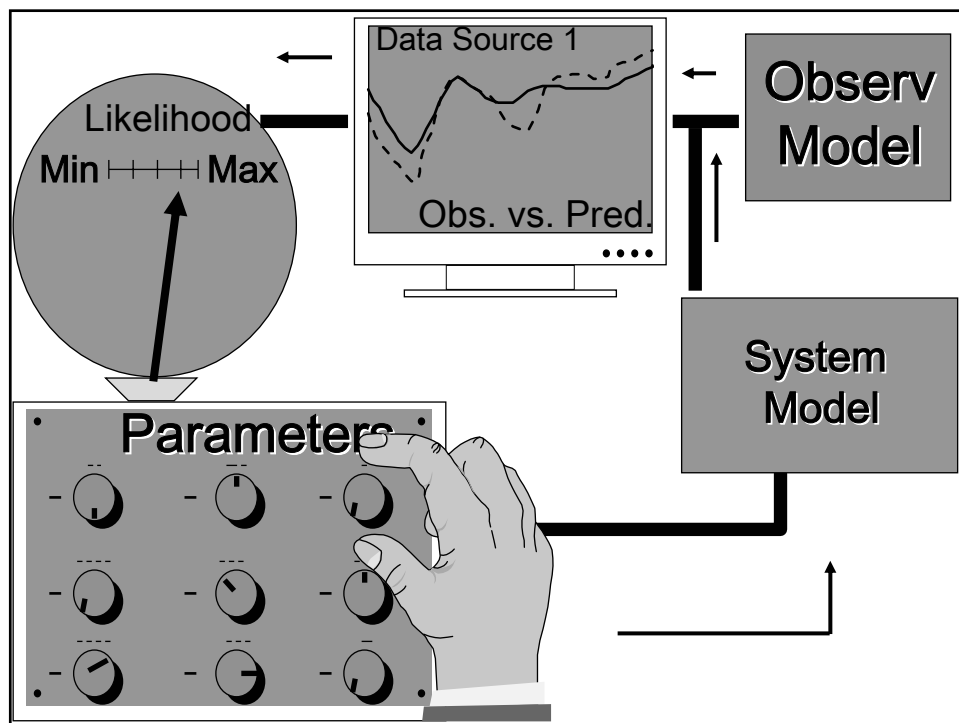
Q^FC Quantitative Fisheries Center

Welcome to our first (introductory 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 and Agency Partners

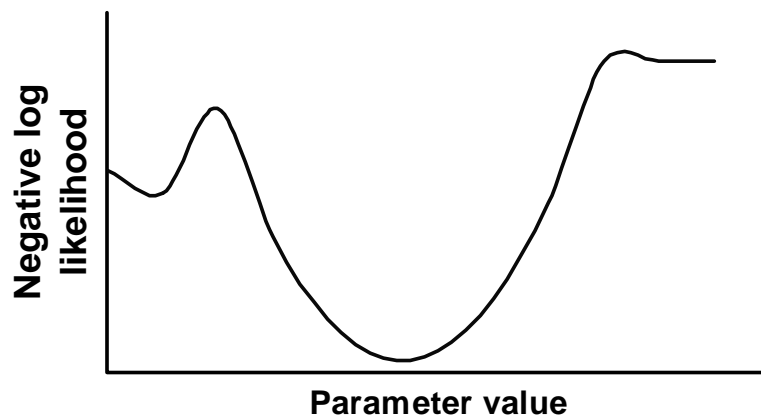
Introduction to parameter estimation using ADMB

- What is it, really, we are trying to do?
- Why use ADMB?
- Simple nonlinear regression as a learning example:
 - General framework and likelihood
 - Von Bertalanffy Growth Model
 - Initial parameter values
 - Coding in ADMB




The problem: estimate parameters of models

- Many models are nonlinear.
- Models vary from one problem to the next.
 - Biological differences
 - Data availability differs
 - Probability distributions differ
- Interested in uncertainty about parameter estimates and things calculated from parameter estimates.



Reasons to use ADMB

- Fast
 - Accurate
 - Flexible (not “canned”)
 - Designed for general maximum likelihood problems
 - Several approaches to uncertainty
- 
- Automatic differentiation*

ADMB Approach

- Create “template” (tpl file) that:
 - Reads in data (from dat file)
 - Defines model
 - Defines likelihood
- Convert to c++ code using tpl2cpp.exe
- Build (compile and link) a *.exe that:
 - Finds maximum likelihood
 - Reports results

Nonlinear regression as an Example

Observed value y_i = $g(\underline{\theta}, X_i)$ + error ε_i

Parameters (to be estimated) $\underline{\theta}$

Function (needs to be specified) g

Explanatory variable (known) X_i

$\varepsilon_i \sim N(0, \sigma^2)$

"Distributed as" ε_i

normal N

mean = 0, variance = σ^2

For nonlinear regression
minimize the objective function:
negative log concentrated
Likelihood

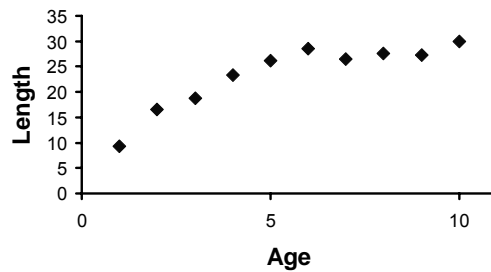
$$-Conc[LogL] = \frac{k}{2} \ln[RSS] + IC$$

Equivalent to maximizing the likelihood. Accept for now,
later we will go over where this comes from.

To summarize the process

- Define the model
 - i.e., for nonlinear regression pick g
- Figure out the negative log likelihood
 - For nonlinear regression we can use negative log concentrated likelihood
- Specify initial parameter values (educated guess)
- Adjust parameters until likelihood is maximized (negative log concentrated likelihood is minimized)

Example – Modeling size versus age



Simulated data

Von Bertalanffy Model

$$L_i = g(a_i, \underline{\theta}) + \varepsilon_i$$

$$g(a_i, \underline{\theta}) = L_{\infty} (1 - e^{-K(a_i - t_0)})$$

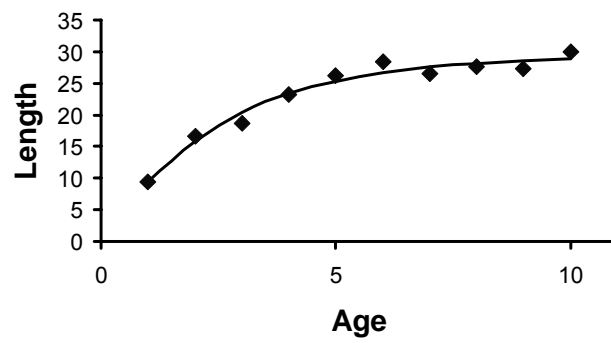
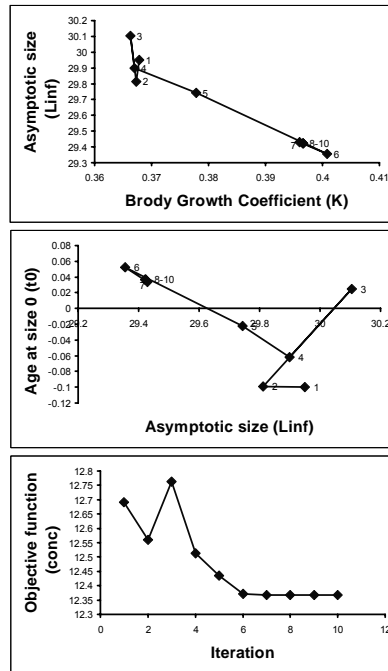
Initial values

L_{∞} Asymptotic size, Brian picked 30 (about the length at the oldest age).

t_0 Age at size zero. Brian picked 0.1, a number close to zero.

K Brody growth coefficient. Brian picked 0.35. Could base this on values from other studies or crude analysis. E.g.,

$$\frac{\Delta_L}{(L_{\infty} - L)} = (1 - e^{-K})$$



Sections of a simple ADMB Template

- DATA
- *INITIALIZATION*
- PARAMETER
- PROCEDURE
- *REPORT*

You will see...

- How to use Emacs
- How to comment out a line
- How to define data
- How to define parameters to be estimated
- Defining the objective function
- Initial values for parameters
- Some simple calculations and built in functions
- Writing some results you want out (*.rep)
- Standard admb output (*.par, *.std, *.cor)

Now to learn some coding.....



Create a Comment

- Add the following as a comment at the end of DATA_SECTION

“We must make sure the data was read in correctly”

Two Useful Commands

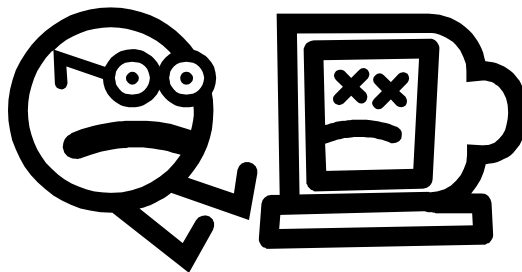
```
!!cout << length_obs << endl;  
!!exit(5);
```

- Output variable value to Run-Time Message buffer
- Exit the model at this point and output code number to Run-Time Message buffer

Create a Variable

- Declare a vector called *residuals* that goes from 1 to *nobs* in length
- Calculate the value of the variable
 - *residuals* = observed – predicted mean length-at-age
- Include *residuals* in growth.rep file

.....That's all for now



You saw...

- Comments
 - // in tpl, # in dat
- DATA_SECTION
 - init_* (int, vector – there is also number, matrix...)
- PARAMETER_SECTION
 - init_*, objective_function_value
- INITIALIZATION_SECTION
 - Set initial values for parameters

You also saw...

- PROCEDURE_SECTION
 - This is where the work gets done
 - Use semicolons here
 - +, -, *, /, exp, log, norm2
- REPORT_SECTION
 - Output defined variables, Quoted text, Calculated quantities
- Output files
 - *.rep, *.par, *.std, *.cor

Simple Diagnostics

- Types of error messages:
 - Compile
 - Run-time
- Modes of operation:
 - Safe mode
 - Optimization mode

TPL2CPP Errors

Error in line 65 while reading

r

- Line number refers to tpl file
- Need a space at start of each line of code
 - Except for comments and section headings
- Need a “return” after last line of tpl

Compile Errors

c:/.../growth.cpp:51: error: expected `;' before "rss"

c:/.../growth.cpp:44: error: `Linf' undeclared (first use this function)

- Check designated line in cpp file
- Make corrections to tpl file, not cpp file

Run-Time Errors

Error reading in dat file – no error message

- In DATA_SECTION, values made up for *init_objects* that are not assigned values from dat file
- Use “cout” command to make sure dat file reads in properly
 - Can include arbitrary test numbers at end of dat file as a check

Run-Time Errors

Var	Value	Gradient
1	10.00000	-1.#IND0e+000
Var	Value	Gradient
2	0.00000	1.#QNANe+000

- IND0: infinity or division by zero
- QNAN: not a real number
- Use “cout” command to check calculations

Run-Time Errors

Error in matrix inverse -- matrix singular in
inv(dmatrix)

- Hessian cannot be inverted to obtain asymptotic standard errors
- Use different parameter starting values
- Reparameterize model

Run-Time Errors

array bound exceeded -- index too high in
prevariable::operator[]

- Tried to assign value outside the defined range indices for vector or matrix
- Use “exit” command to locate error in tpl
- Error message only appears when in safe mode

Modes of Operation

- Safe mode: provides bounds checking on all array objects
 - ADModel > tpl2cpp > compile > link
 - ADModel > makeadms
- Optimization mode: provides faster execution
 - ADModel > makeadm

Some essential theory

What is a likelihood etc?

A likelihood is what it sounds like

- Measure of how likely a set of parameters are to have produced the data
- Can be confusing. Often written as:

$$L(\underline{\theta} | \underline{X})$$

But not always – $L(\underline{\theta}, \underline{X})$

$$L(\underline{\theta})$$

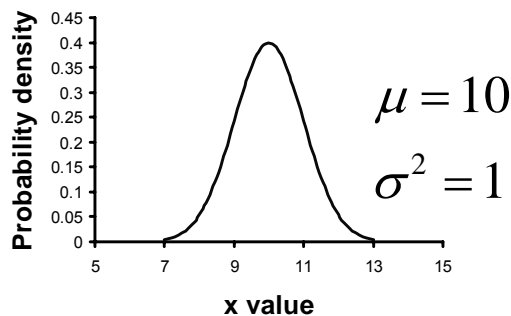
- Think of as function of parameters.
- Depends upon data.
- Requires us to specify probability distributions.

A very simple example

- A single observation from a normal distribution

Probability density function

$$f(x | \mu, \sigma^2) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(x - \mu)^2}{2\sigma^2}\right)$$



More on probability distributions

- Probability density functions (pdf) for continuous distributions

$$\int_{-\infty}^{\infty} f(x) dx = 1.0$$

- Probability mass functions (pmf) for discrete distributions

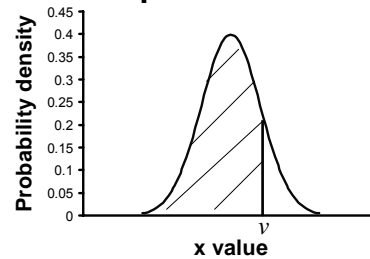
$$\sum f(x) = 1.0$$

- Joint pdf/pmf for multiple independent observations

$$f(x_1, x_2, x_3, \dots, x_k) = f(x_1)f(x_2)f(x_3)\cdots f(x_k) = \prod_{i=1}^k f(x_i)$$

Cumulative Distribution Function and probabilities from pdf

- CDF $F_X(v) = \int_{-\infty}^v f(x)dx$



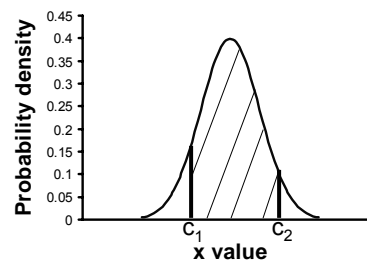
- $P(c_1 < X < c_2) = \int_{c_1}^{c_2} f(x)dx$

This slide and next altered. Here only show CDF graph since cdf and prob in Interval do not show unless Animation is active!

Cumulative Distribution Function and probabilities from pdf

- CDF $F_X(v) = \int_{-\infty}^v f(x)dx$

- $P(c_1 < X < c_2) = \int_{c_1}^{c_2} f(x)dx$



This slide and previous altered. Here only show prob in interval since cdf and prob in interval do not show unless animation is active!

The likelihood function

- Quantitatively equal to the joint probability density function

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

- But NOT a probability density for parameters

$$\int_{\text{all } \underline{\theta}} L(\underline{\theta}) d\underline{\theta} \text{ NOT necessary} = 1.0$$

Maximum likelihood estimates are values of parameters that maximize the likelihood

Properties of maximum likelihood estimates

- Invariant to transformations
- Asymptotically efficient (lowest possible variance)
- Asymptotically normally distributed
- Asymptotically unbiased (expected value of the estimated parameter equals the true value)
- If we assume independent, normally distributed errors, ML methods provide the same estimates of structural parameters as least squares.

Summary – versatile and widely used with a number of desirable properties, but not perfect!

E.g., can be biased for small n

$$\hat{\sigma}_{ML} = \frac{\sum (x_i - \bar{x})^2}{n}$$

$$\hat{\sigma}_{UB} = \frac{\sum (x_i - \bar{x})^2}{n-1}$$

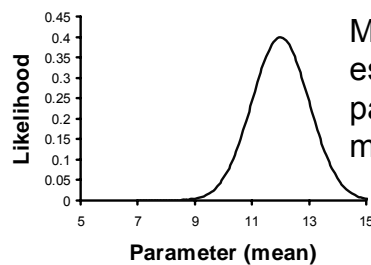
For normal distribution

Back to really, really, really simple example: Normal, one observation, variance known

- Quantitatively equal to the probability density function (also called the probability function for discrete random variables)

$$L(\mu | x, \sigma^2 = 1) = f(x | \{\mu, \sigma^2 = 1\}) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(x - \mu)^2}{2\sigma^2}\right)$$

$x = 12$



Maximum likelihood estimates are values of parameters that maximize the likelihood

Slightly more complicated example – normal sample

$$L(\mu, \sigma^2) = \prod_{i=1}^k \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{1}{2\sigma^2}(x_i - \mu)^2\right)$$

$$= \left(\frac{1}{\sqrt{2\pi}\sigma}\right)^k \exp\left(-\frac{1}{2\sigma^2} \sum_i (x_i - \mu)^2\right)$$

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

$$\log(ab) = \log(a) + \log(b)$$

$$\log(a^n) = n \log(a)$$

$$\log(a/b) = \log(a) + \log(b^{-1}) = \log(a) - \log(b)$$

$$\log(e^a) = a$$

$$a^b \cdot a^c = a^{b+c} \rightarrow \exp(a) \exp(b) = \exp(a+b)$$

More forms of almost the same thing. Ignoring constants.

- If you minimize negative log likelihood you can ignore constants because:

For $-\log L(\theta) = IC + g(\theta)$
 same θ will minimize neg log L and g()

- Example of the reduced (ignored constants dropped) negative log likelihood (for normal).
 This depends on what you estimate.

$$-\log L(\mu, \sigma^2) = k \ln \sigma + \frac{1}{2\sigma^2} \sum_i (x_i - \mu)^2 \quad \mu, \sigma^2 \text{ estimated}$$

$$-\log L(\mu, \sigma^2) = \frac{1}{2\sigma^2} \sum_i (x_i - \mu)^2 \quad \mu \text{ estimated}$$

OK lets do something beyond a simple normal sample, back to nonlinear regression

$$y_i = g(\theta, X_i) + \varepsilon_i$$

$$y_i = \mu_i + \varepsilon_i$$

$$y_i \sim N(\mu_i, \sigma^2)$$

$$-\log L(\mu, \sigma^2) = k \ln \sigma + \frac{1}{2\sigma^2} \sum_i (x_i - \mu_i)^2 + \mathbf{IC}$$

Concentrated Likelihood

In - log likelihood for normal replace σ^2 by

$$\frac{\sum_{i=1}^k (y_i - \mu_i)^2}{k} = \frac{RSS}{k} \text{ to obtain}$$

$$-Conc[LogL] = \frac{k}{2} \ln[RSS] + IC$$

Combining normal data with different variances

- Data are: $\{y_{11}, y_{12}, \dots, y_{1k1}, y_{21}, y_{22}, \dots, y_{2k2}\}$ Plus known predictors \underline{X}
- First and second set of y have different distributions (variances)

$$y_{ij} \sim N(\mu_{ij}, \sigma_i^2)$$

- $-\log L = L_1 + L_2 + IC$: $L_i = k_i \ln \sigma_i + \frac{1}{2\sigma_i^2} \sum_j (y_{ij} - \mu_{ij})^2$
- Just special case of rule for getting joint pdf for independent data

Concentrated negative log likelihood when there is more than one normal component

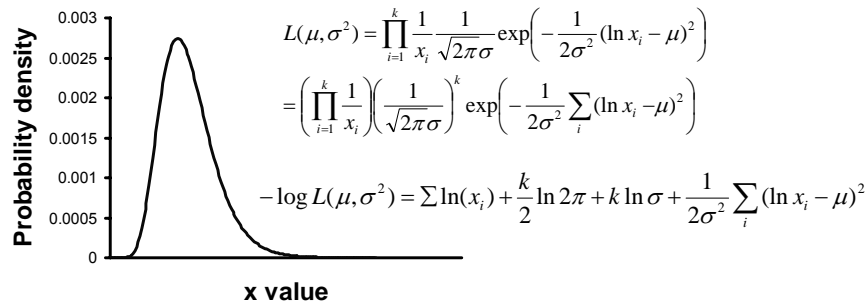
$$-Conc[LogL] = \frac{K}{2} \log[RSS] + IC$$

$$K = K_1 + K_2 + \dots K_T$$

$$RSS = RSS_1 + \lambda_2 RSS_2 + \dots \lambda_T RSS_T$$

$$\hat{\sigma}_1^2 = \frac{RSS}{K}, \hat{\sigma}_i^2 = \frac{\hat{\sigma}_1^2}{\lambda_i}, \lambda_i = \frac{\sigma_1^2}{\sigma_i^2} \text{ (assumed known)}$$

Likelihood for Lognormal Distribution



$$X \sim LN(\mu, \sigma^2) \text{ then } Y = \ln(X) \sim N(\mu, \sigma^2)$$

$$E(X) = \exp\left(\mu + \frac{\sigma^2}{2}\right)$$

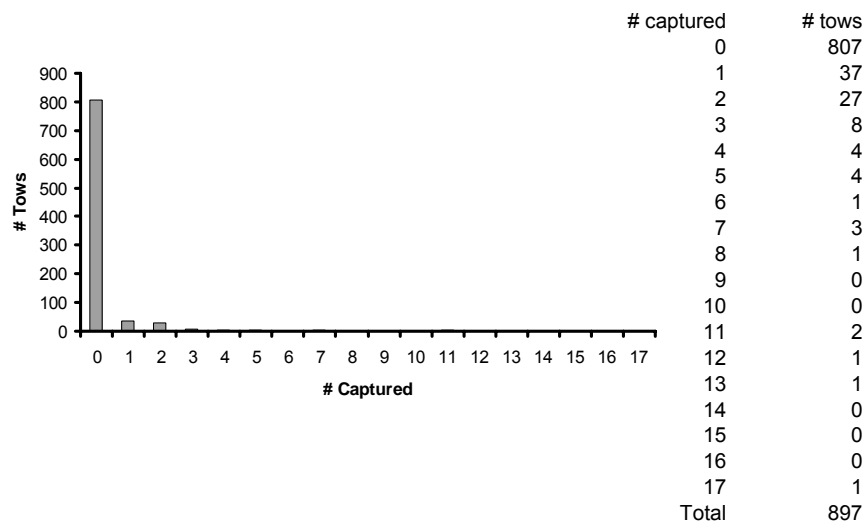
$$Var(X) = e^{2\mu+2\sigma^2} (e^{\sigma^2} + 1)$$

$$CV(X) = \sqrt{e^{\sigma^2} - 1} \cong \sigma, \sigma \text{ "small"}$$

What if the data are not
normal/lognormal?

Example of different (discrete
distribution)

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

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

$$\begin{aligned} -\log L(m, s | x) &= -\log(\Gamma(x+s)) + \log(\Gamma(s)) + \log(x!) \\ &\quad - s \log\left(\frac{s}{m+s}\right) - x \log\left(\frac{m}{m+s}\right) \end{aligned}$$

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

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

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

Lets look at some code for this albatross!



*From Royal Forest And Bird Protection
Society Website, NZ*

A challenge: change the model
so that you assume a Poisson
instead of a Negative Binomial

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

$$- \text{Log} L = \lambda - x \ln \lambda + \ln x!$$

Probability mass functions for negative binomial and Poisson and general likelihood and negative log-likelihood equations

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

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

$$L(\theta) = \prod_{all\ obs} f(x_i)$$

$$-LogL = \sum_{all\ obs} -\log(f(x_i))$$

$$P(X = x) = f(x) = \exp(\log(f(x)))$$

$$-LogL = \sum_{all\ obs} -\log(f(x)) = \sum_{x=0}^{17} \sum_i -\log(f(x_i))$$

$$\sum_i -\log(f(x_i)) = n_x \cdot (-\log(f(x)))$$

How to Manage Complex Code

- New ADMB concepts and techniques
 - Sdreport objects
 - Loops
 - Conditional statements
 - User-defined functions
- Time-varying Von Bertalanffy growth model
 - Random walk

Sdreport Objects

- Included in reports of:
 - Asymptotic standard errors (std file)
 - Correlation matrix (cor file)
- Declared in `PARAMETER_SECTION`
 - `sdreport_number`
 - `sdreport_vector`
 - `sdreport_matrix`

Loops

- Repeats code a specified number of times

```

    looping variable
for (i=m;i<=n;i++) ← 'i' goes from 'm' to 'n'
                    in increments of 1
{
    . . . . . ; ← Code that is repeated for
                  each increment of 'i'
}

```

Conditional Statements

- Runs code if conditions met

```

if (condition) ← if condition is true
{
    . . . . . ; ← then run this code
}

```

Common Conditional Statements

- $(X==Y)$ X equal to Y
- $(X!=Y)$ X not equal to Y
- $(X<Y)$ X less than Y
- $(X\leq Y)$ X less than or equal to Y
- $(X>Y)$ X greater than Y
- $(X\geq Y)$ X greater than or equal to Y

Conditional Statements

```
if (condition) ← if condition is true
{
    . . . . . ; ← then run this code
}
else ← if condition is false
{
    . . . . . ; ← then run this code
}
```


User-Defined Functions

- Organize code in
PROCEDURE_SECTION

function_name(); ← Call function for use

FUNCTION *function_name* ← Define function

.....; ← Code for function

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

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

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

Ratio of relative variances (assumed known)

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

Concentrated Negative Log Likelihood

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

Now to look at the code.....



Sdreport Object

- Change *sd_log_Lin* to an sdreport object

Some Sensitivity Checking

- What happens when we change the value of the ratio of relative variances?
 - Try *lambda* of 0.2
 - Try *lambda* of 0.3

Now for the full likelihood.....



Build your first admb tpl from “scratch”

- This example is a very simple surplus production stock assessment model fit to arrowtooth flounder data
- I will go over the model and things to worry about as you code
- And I will make some suggestions about coding strategy

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 (simple learning example)

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

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

Observation Stochasticity with lognormal errors

$$Bobs_t = qB_t\varepsilon_t \quad \hat{B}obs_t = qB_t$$

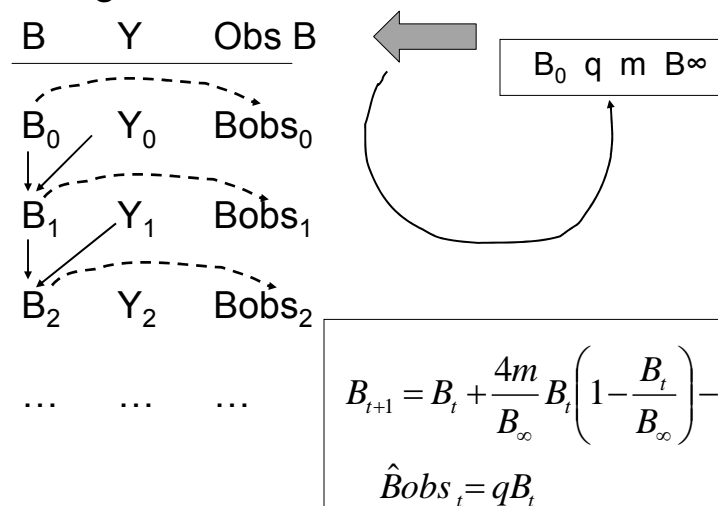
$$\ln(Bobs_t) = \ln(q) + \ln(B_t) + \ln(\varepsilon_t)$$

So our objective function will be to minimize the neg conc log likelihood:

$$conc = \frac{n}{2} \log(RSS)$$

$$RSS = \sum \ln(Bobs_t / \hat{B}obs_t)^2$$

Schematic of forward simulation approach Lognormal observation error



But finding a solution with the discrete time logistic can be tricky

- During fitting process catch yield can exceed biomass and you get negative B in next step (and you are done, program will bail!)
- Avoid this by creating an extra “true yield” that equals observed yield except when this is more than X% (say 50%) of current biomass, in which case just set true yield to 50% of biomass. Add a “penalty sum of squares” to RSS (don’t worry about adjusting n).
- Also Biomass could go negative if it starts too high over carrying capacity. Just to be safe make Biomass equal to a small positive value (say 10) if the pop model has it going lower than that.
- These ad hoc fixes are to get to the solution but should not be influencing things at solution or this is of concern.

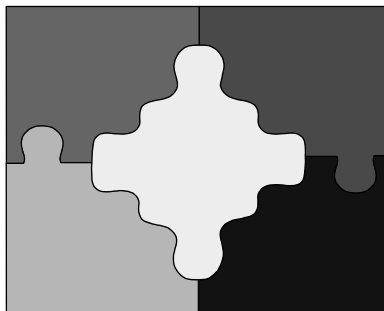
Strategy for building your tpl from scratch

- There is no one way but this is how I do it.
- I start with a working file from another problem, as memory aid on coding syntax etc (section names, define variables, loops...).
- I first create a minimal program which consists of the required data, parameter, and procedure sections, has an objective function variable and an estimable parameter.
- I first just get the program to read the data in correctly (use cout and exit)
- I then sequentially add calculations to the procedure section and check they work using cout and exit. I find it much easier to check things as I build them up rather than trying to find where the errors are after writing lots of code.
- Use small steps and do not worry about efficiency too much at this stage.
- Before I get too much invested in cool stuff in the rep file, or defining lots of derived variables of interest not needed for estimation, I try to make sure estimation is actually working.
 - First think about initial values for parameters

A few other details

- Remember estimate parameters that must be positive on log scale and backtransform them at top of procedure section.
- Try to avoid hard coding stuff so you can change your mind. E.g, rather than leaving q out of all the code because we are assuming it is known, keep it in code, define $\ln q$ as parameter, and set the parameter phase to -1 and give $\ln q$ an initial value of zero.

Now lets build some code....



What are these asymptotic standard errors anyway?

An overview on inferences

- By inference I mean going beyond point estimates and saying something about the quality of the estimates. How likely is it that the estimate is close to the true value?
- Topics related to inference
 - Estimates of standard errors
 - Confidence intervals
 - Hypothesis tests
 - Bayesian probability intervals

- Inferences depend upon the variance-covariance matrix:

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

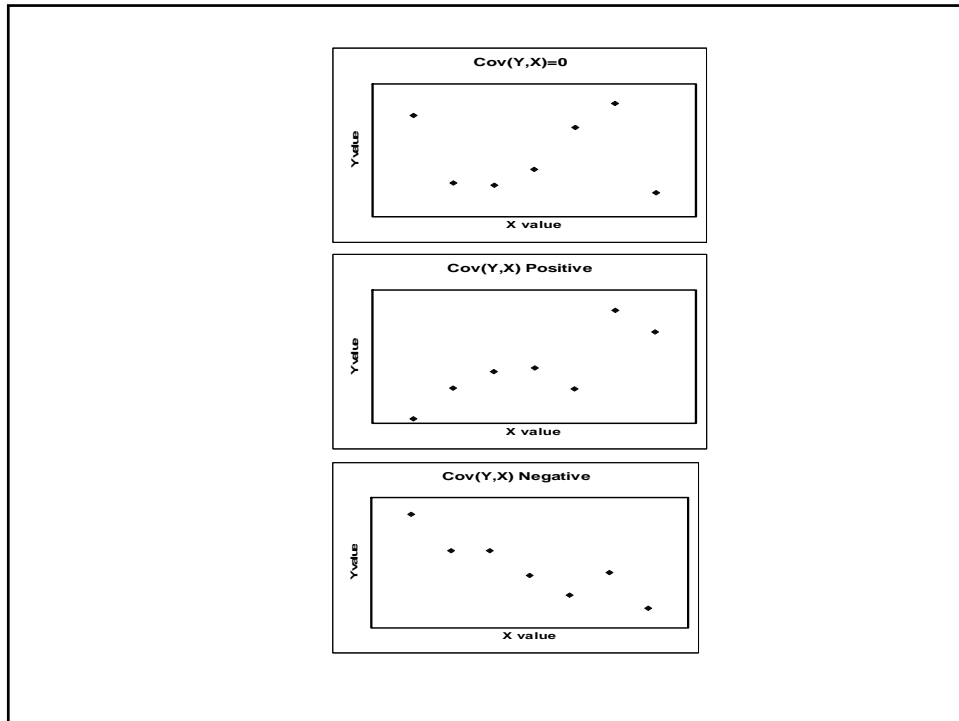
Diagonal elements are variances of parameter estimates, off-diagonals are covariances among parameter estimates.

What is a variance and covariance?

- Recall definition of expected value

$$E(X) = \int xf(x)dx$$

- $Var(X) = E((x - E[X])^2)$
- $Cov(X, Y) = E[(x - E[X])(y - E[Y])]$



Covariances and parameter estimates

- 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:

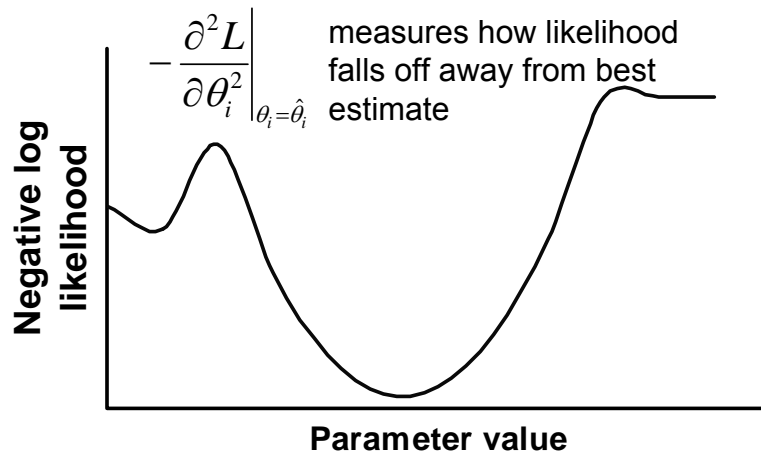
$$\rho_{ij} = \frac{\sigma^2_{ij}}{\sqrt{\sigma^2_{ii}} \sqrt{\sigma^2_{jj}}}$$

Asymptotic results for parameters

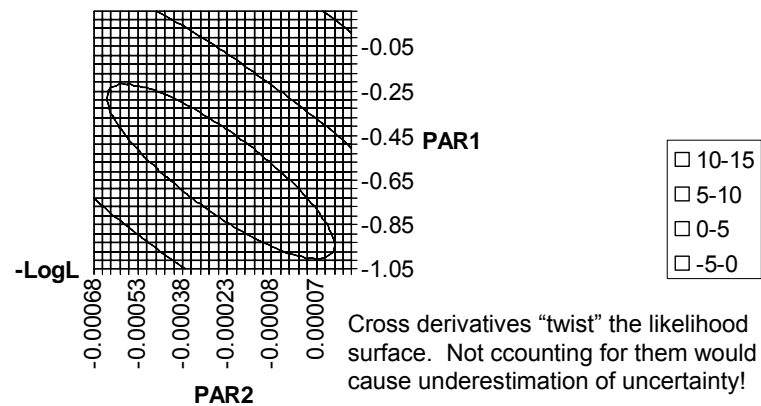
- Done automatically in ADMB (i.e., you don't have to code anything)
- Results are in *.std and *.cor
- These are based on:

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

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



Neg log likelihood for stock recruitment model



Example *.std output

index	name	value	std dev
1	log_q	-1.6219e+000	2.7145e+000
2	log_popscale	7.4954e+000	1.6715e-001
3	log_sel_par	-6.0105e+000	2.7178e+000
4	log_sel_par	-3.1105e+000	2.7089e+000
5	log_sel_par	-1.3544e+000	2.7038e+000
6	log_sel_par	-1.4792e-001	2.6779e+000
7	log_sel_par	-4.7468e-002	2.5159e+000
8	log_sel_par	-7.7288e-001	2.0588e+000
9	log_relpop	8.1995e-001	1.7816e-001
10	log_relpop	1.5404e+000	1.7094e-001
11	log_relpop	1.2639e+000	1.7262e-001
...

Example of *.cor file

index	name	value	std dev	1	2	3	4	5
1	log_q	-1.6219e+0002.7145e+000	1.0000					
2	log_popscale	7.4954e+0001.6715e-001	-0.6779	1.0000				
3	log_sel_par	-6.0105e+0002.7178e+000	-0.9971	0.6695	1.0000			
4	log_sel_par	-3.1105e+0002.7089e+000	-0.9997	0.6763	0.9970	1.0000		
5	log_sel_par	-1.3544e+0002.7038e+000	-0.9999	0.6771	0.9971	0.9997	1.0000	
6	log_sel_pa	...						

52
	...							

Standard error estimates for derived quantities

- Often (well ... almost always) we want to assess the uncertainty of derived quantities that are not formally parameters
 - E.g., biomass in last year of assessment, MSY for a logistic surplus production model, ratio of abundance in 2002 to abundance in 1995, SSBR based on recent mortality schedule,...
- This can be done almost automatically in ADMB based on delta method

Standard error estimates for derived quantities (continued)

- This is done for any variable of types:
 - sdreport_number, sdreport_vector, sdreport_matrix, likeprof_number

- The results are based on:

$$Cov[g(\underline{\theta}), h(\underline{\theta})] \cong \sum_i \sum_j Cov[\theta_i, \theta_j] \frac{\partial g}{\partial \theta_i} \frac{\partial h}{\partial \theta_j}$$

- Results are included in *.std and *.cor files

What happens if we use RSS instead of neg logL or neg logConc?

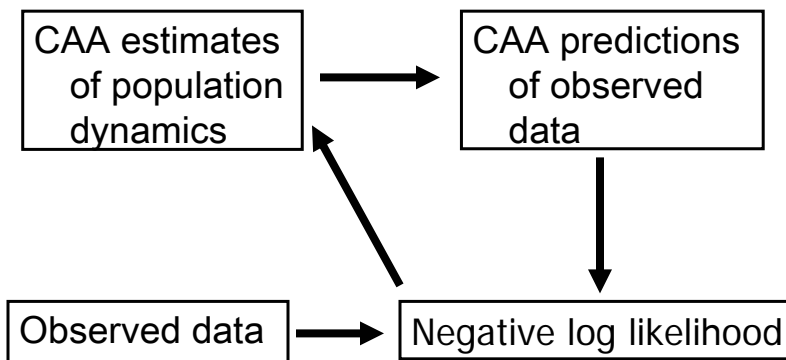
```
Made this change to growth.tpl
// conc=(nobs/2.0)*log(rss); //concentrated likelihood
//changed obj function to just be RSS for illustrative purposes
//DO NOT DO THIS
conc=rss;
```

	Neg LogL or Conc		RSS	
name	value	std	value	std
log_Linf	3.382	0.030	3.382	0.019
log_K	-0.925	0.152	-0.925	0.098
t0	0.037	0.216	0.037	0.140

Catch-At-Age Analysis

- Overview of catch-at-age model
 - Population submodel
 - Observation submodel
 - Negative log likelihood
- New ADMB concepts and techniques
 - Array and matrix functions
 - Phases
 - Some parameterization issues

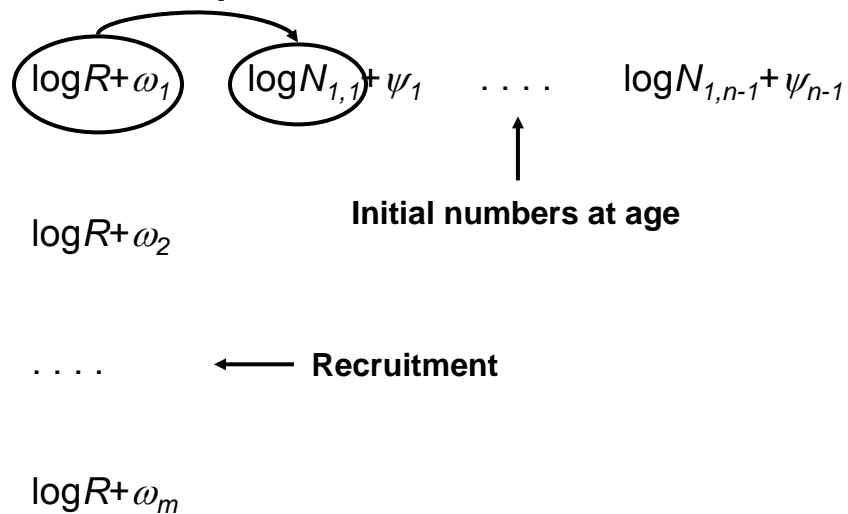
Overview of Catch-At-Age



Observed Data

- Total annual fishery catch
- Proportion of catch-at-age
- Auxiliary data
 - Fishing effort
 - Survey index of relative abundance

Population Submodel



Population Submodel

The diagram illustrates the Population Submodel. At the top, the title "Population Submodel" is centered. Below it, the equation $N_{y+1,a+1} = N_{y,a} e^{-Z_{y,a}}$ is shown. Arrows point from "Numbers of fish" to $N_{y,a}$ and from "Survival" to the exponential term $e^{-Z_{y,a}}$, which is circled. Below the equation, "Total mortality" points down to $Z_{y,a}$ in the equation $Z_{y,a} = M + F_{y,a}$. "Fishing mortality" points down to $F_{y,a}$. "Natural mortality" points up to M .

$$N_{y+1,a+1} = N_{y,a} e^{-Z_{y,a}}$$

$$Z_{y,a} = M + F_{y,a}$$

Population Submodel

The diagram illustrates the Population Submodel. At the top, the title "Population Submodel" is centered. Below it, the equation $F_{y,a} = s_a q E_y e^{\zeta_y}$ is shown. Arrows point from "Selectivity" to s_a , from "Effort" to E_y , from "Catchability" to q , and from "Effort error" to ζ_y . Below the equation, the distribution $\zeta_y \sim N(0, \sigma_\zeta^2)$ is shown.

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

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

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

$$\downarrow$$

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

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

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

Model Parameters

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

$$q \quad s_1, \dots, s_{n-1}$$

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

Ratio of relative variances (assumed known)

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

Negative Log Likelihood (ignoring constants)

$$-\log(L) = m \log(\sigma_\varepsilon) + \frac{1}{2\sigma_\varepsilon^2} \sum_y \left[\log \left(\frac{\tilde{C}_y}{C_y} \right) \right]^2$$

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

$$+ m \log(\sigma_\zeta) + \frac{1}{2\sigma_\zeta^2} \sum_y \zeta_y^2$$

Array and Matrix Functions

- The operator `*` provides matrix multiplication
- For vector objects `x` and `y`, and number `z`
`z=x*y;` // returns $z=x_1y_1 + \dots + x_ny_n$
- For matrix objects `x`, `y` and `z`
`z=x*y;` //returns $z_{i,j}=x_{i,1}y_{1,j} + \dots + x_{i,n}y_{n,j}$

Array and Matrix Functions

- Functions *elem_prod* and *elem_div* provide elementwise multiplication and division
- For vector objects `x`, `y` and `z`
`z=elem_prod(x,y);` //returns $z_i=x_iy_i$
`z=elem_div(x,y);` //returns $z_i=x_i/y_i$
- For matrix objects `x`, `y` and `z`
`z=elem_prod(x,y);` //returns $z_{i,j}=x_{i,j}y_{i,j}$
`z=elem_div(x,y);` //returns $z_{i,j}=x_{i,j}/y_{i,j}$

Phases

- Minimization of objective function can be carried out in phases
- Parameter remains fixed at starting value until its phase is reached, then it become active
- Allows difficult parameters to be estimated when other parameters are “almost” estimated

Phases

- Specified in `PARAMETER_SECTION`

```
init_number x           //estimate in phase 1
init_number x(1)        //estimate in phase 1
init_number x(-1)       //remains fixed
init_vector x(1,n,2)    //estimate in phase 2
init_matrix x(1,n,1,m,3) //estimate in phase 3
```

Parameterization Issues

- How do you estimate highly correlated parameters?
 - Catchabilities for multiple fisheries
 - Annual recruitments
- Difference method
- Dev vector method

Difference Method

- Estimate n free parameters:

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

- Then

$$\log N_1 = \log R$$

$$\log N_2 = \log N_1 + \psi_1$$

.....

$$\log N_n = \log N_{n-1} + \psi_{n-1}$$

Dev Vector Method

- Estimate one free parameter

$$R$$

- Estimate m parameters as
bounded_dev_vector

$$\omega_1, \dots, \omega_m$$

- Then

$$\log R_1 = \log R + \omega_1$$

....

$$\log R_n = \log R + \omega_n$$

bounded_dev_vector

- Specified in PARAMETER_SECTION

init_bounded_dev_vector x(1,m,-10,10)

- Each element must take value between
lower and upper bounds

$$-10 < x_i < 10$$

- All elements must sum to zero

$$\sum_{i=1}^m x_i = 0$$

Now to look at the code.....



Negative Log Likelihood
(ignoring constants)

$$-\log(L) = m \log(\sigma_\varepsilon) + \frac{1}{2\sigma_\varepsilon^2} \sum_y \left[\log \left(\frac{\tilde{C}_y}{C_y} \right) \right]^2$$

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

$$+ m \log(\sigma_\zeta) + \frac{1}{2\sigma_\zeta^2} \sum_y \zeta_y^2$$

Selectivity Double Logistic Function

- Double logistic function requires four parameters:
 - b1 First inflection point
 - b2 First slope
 - b3 Second inflection point
 - b4 Second slope

Selectivity Double Logistic Function

$$s_a = \frac{1}{1 + e^{-b_2(a-b_1)}} \left[1 - \frac{1}{1 + e^{-b_4(a-b_3)}} \right]$$

Standardize selectivity to age-5

$$s'_a = \frac{s_a}{s_5}$$

Selectivity

Double Logistic Function

- Suggested starting values for double logistic parameters (log scale)
 - b1 1.39
 - b2 0.34
 - b3 1.25
 - b4 -0.69

Now to play with the code.....



Sensitivity to Parameter Starting Values

- Why do we care about sensitivity to parameter starting values?
- Methods for specifying starting values
 - Default values
 - In tpl file
 - In dat file
 - In pin file
- Precedence between the methods

Why do we care about sensitivity to starting values?

- Avoiding local minimums in the likelihood surface
 - If different starting values lead to solution with lower obj. function value, then you were at a local minimum
- Identifying sensitive parameters
 - If small change to parameter starting value causes large change in results, then you may want to reparameterize model

Default Starting Values

- Parameter with unspecified starting value has default starting value of zero
- Bounded parameter has default starting value which is midway between lower and upper bounds

Specify starting values in tpl file

INITIALIZATION_SECTION

log_q -1.0

- Must recompile tpl file everytime starting values are changed

Specify starting values in dat file

DATA_SECTION

init_number start_log_q

PRELIMINARY_CALCS_SECTION

log_q = start_log_q;

- Can change starting values without recompiling tpl file

Specify starting values in pin file

#Example pin file for model with 15 parameters

0 0 0 -1 0 0 0 0 0 0 0 0 0 0 0

- Can change starting values without recompiling tpl file
- Must specify a starting value for each parameter

Precedence Between Methods

- Specifying starting values in dat file takes precedence over pin file and INITIALIZATION_SECTION
- Specifying starting values in pin file takes precedence over INITIALIZATION_SECTION

Now for an example.....



Making your code more efficient

- Really sometimes it matters!

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

Lets look at some example code

- New and improved albatross example
 - Uses built-in admb functions to avoid loop
 - Calculates quantities not needed during estimation only during sd_phase and when reporting
- New and improved growthtime example
 - Combines three loops over years into one
 - Avoids two unneeded loops using explicit and implicit elementwise operations

Controlling where data are read from and results written too

- You have already seen several ways to control where initial values for parameters come from.
- We will consider using `ad_comm` method in `tpl` to control where data are read from and parameters estimates are written to.

Code fragment examples for changing the default dat or par file:

```
DATA_SECTION
// will read data from file catchdat.dat
!! ad_comm::change_datafile_name("catchdat.dat");
init_int nyrs
init_int nages
...
PARAMETER_SECTION
// will write parameters to file catch.par
!! ad_comm::change_parfile_name("catch.par");
...
```

Examples based on admb manual

Changing where you write results to

- No `add_comm` for this but you can control this with “`cout`”
- Method 1: redirect `cout` (that by default goes to screen) to a file with a “pipe”
- Method 2: redirect `cout` to a file with code in the `tpl`

Method 1: You have compiled and linked “`myprog`”

Run `myprog >junk.dat`

(in e-macs select “Run...”,
then add `>junk`)

Method 2: // next bit of code to write results to specific file via an output file stream

```
ofstream ofs("test.dat",ios::app);  
ofs<<"Linf and K "<<endl;  
ofs<<"Linf<<" "<<K<<endl;
```

What if you want to read some data from one file and other data from another?

```
// read effort data from effort.dat and save current
// position in catchdat.dat in the object tmp
!! streampos tmp =
ad_comm::change_datafile_name("effort.dat");
  init_vector effort(1,nyrs)
// now read the rest of data from catchdat.dat
// including ioption argument tmp will reset the file to previous
position
!! ad_comm::change_datafile_name("catchdat.dat",tmp);
  init_number M
```

Convergence Issues

- Convergence criteria
- Diagnosing convergence problems
 - Convergence messages
 - Self diagnostics
- Fixing convergence problems
 - Convergence criteria problems
 - Code problems

Convergence Criteria

- Gradients close to zero
 - Maximum $|\text{gradient}| < 1 \times 10^{-4}$
- Obj. function value fails to decrease
 - Change $< 1 \times 10^{-6}$ for 10 iterations in a row
- Obj. function evaluated too many times
 - Maximum evaluations = 1,000
- Line search fails to find parameters with lower objective function value
 - Step size adjusted 30 times

Convergence Messages

ic > imax in fminim is answer attained ?

Function minimizer not making progress ... is
minimum attained?

Minimprove criterion = 0.0000e+000

- Run-time messages indicating convergence problems

Self Diagnostics

- Compare smallest and largest eigenvalues of Hessian in eva file
- Is logarithm of determinant of Hessian small in cor file?
- Are correlations large in cor file?
- Are standard errors large compared to parameter value in std file?
- Examine trajectory of iterations including objective function and key parameters

Convergence Criteria Problems

- Is convergence criteria too strict or too loose?
 - Does objective function value change substantially as gradients approach convergence criterion?
 - Are results sensitive to changes in convergence criterion?
 - Try different parameter starting values

Changing Convergence Criteria

- In tpl file

RUNTIME_SECTION

convergence_criteria 0.01,0.01,0.001

maximum_function_evaluations 20,20,500

- With runtime switches

-crit 0.01,0.01,0.001 -maxfn 20,20,500

- Switch to restart (after rescaling) if function not improving but gradients not near zero

-rs

Code Problems

- Do predictions respond to parameter values?
 - If not possibly need to use a different function
 - Or parameterize the current function differently

Now for an example.....

