

Building a simple model

ADMB and stock assessment

Arni Magnusson
Anders Nielsen

ICES, 21–25 Feb 2011

Outline

- 1 Modelling - data, parameters, predictions, obj fun, output
- 2 Sections - data, parameter, procedure, report
- 3 Data types - basic types, declarations
- 4 Exercise - estimate the mean

Statistical modelling

- **Read in data**

text file

- **Specify model**

relate parameters to data

- **Make predictions**

calculate fitted values to compare with observations

- **Specify objective function**

function to minimize, usually $-\log L$

- **Minimize objective function**

some algorithm searches for the “best” parameter values

- **Write out results**

text file

ADMB sections

The corresponding sections in AD Model Builder are:

DATA_SECTION

read in data

PARAMETER_SECTION

declare parameters

PROCEDURE_SECTION

relate parameters to data, make predictions, specify the objective function
[ADMB minimizes the objective function and writes out results]

(REPORT_SECTION)

write out verbose results (optional)

There are many other sections available in ADMB, and we will only use the first three sections in the first exercises

hello.tpl

DATA_SECTION

init_int n

init_vector x(1,n)

init_vector y(1,n)

PARAMETER_SECTION

init_number a

init_number b

vector yfit(1,n)

objective_function_value f

PROCEDURE_SECTION

yfit=a+b*x;

// Concentrated neglogL

f=0.5*n*log(sum(square(y-yfit)));

Three basic data types

`int` integer

`double` floating point value

`dvariable` floating point value with derivative info

Estimated parameters are dvariables

Intermediate calculations and derived quantities are also dvariables

Each data type can form a vector, matrix, 3d array, ...

Declaring objects

Declaration	when inside	
	DATA_SECTION	PARAMETER_SECTION
<code>int</code>	integer	integer
<code>init_int</code>	integer from file	–
<code>number</code>	double	dvariable
<code>init_number</code>	double from file	dvariable to estimate
<code>init_bounded_number</code>	–	dvariable to estimate with bounds

hello.tpl

DATA_SECTION

init_int n

init_vector x(1,n)

init_vector y(1,n)

PARAMETER_SECTION

init_number a

init_number b

vector yfit(1,n)

objective_function_value f

PROCEDURE_SECTION

yfit=a+b*x;

// Concentrated neglogL

f=0.5*n*log(sum(square(y-yfit)));

Estimate the mean

Create a new model `mean.tpl` that estimates the mean of a vector

Use `hello.tpl` as a template

Implement three approaches, where the objective function is:

- 1 Residual sum of squares
quick and dirty
- 2 Full likelihood
evaluate uncertainty, estimate sigma
- 3 Concentrated likelihood
evaluate uncertainty, implicit sigma

Hints

Residual sum of squares

$$\text{RSS} = \sum (y_i - \mu_i)^2$$

Full likelihood

$$-\log L = [0.5n \log(2\pi)] + n \log \sigma + \frac{\text{RSS}}{2\sigma^2}$$

Concentrated likelihood

Replace σ with $\sqrt{\frac{\text{RSS}}{n}}$:

$$-\log L = [0.5n \log(2\pi)] + 0.5n \log \text{RSS} - [0.5n \log n] + \left[\frac{n}{2} \right]$$