Building a simple model

Assessment workshop

Arni Magnusson

Hafro, 16–24 Jan 2012

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

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

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

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)));
```

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

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

	when inside	
Declaration	DATA_SECTION	PARAMETER_SECTION
`int`	integer	integer
`init_int`	integer from file	–
`number`	double	dvariable
`init_number`	double from file	dvariable to estimate
`init_bounded_number`	–	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)));
```

Outline

Estimate the mean

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

Use `hello.tpl` as a template

Implement two approaches, where the objective function is:

1. Residual sum of squares
   quick and dirty
2. Negative log likelihood
   evaluate uncertainty, estimate sigma

Hints

Residual sum of squares

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

Negative log likelihood

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