

[Next](#)

Scale Plus Deviation Parameterization



This video was created using ADMB-IDE release 4.5.0-1 (July 15, 2011)
You may notice some minor differences if using a different version.

[Next](#)

Related parameters can sometimes be thought of as scale or mean parameter plus deviation from mean.

The original related parameters can be highly correlated.

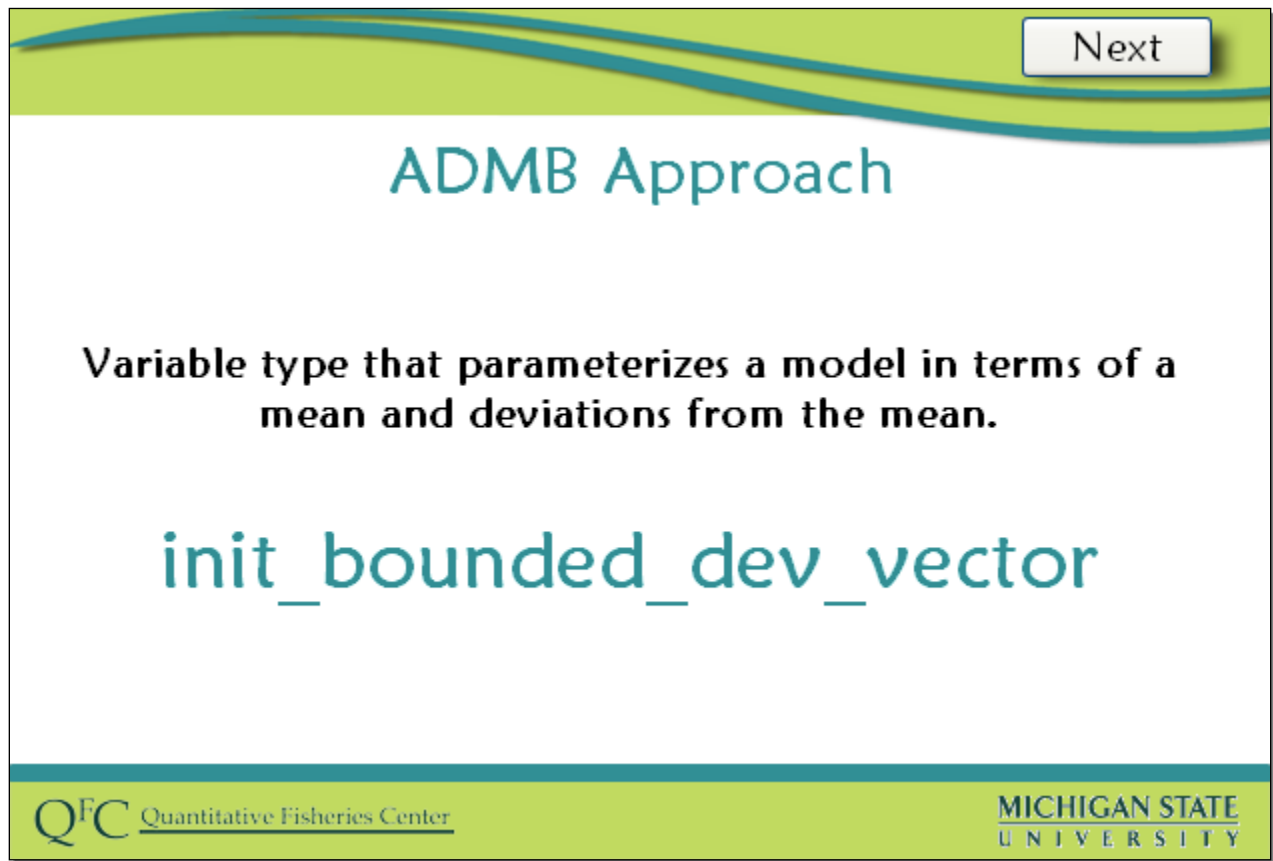
Programs searching for best solution need to "figure out" that to change scale they need to change all the parameters in the same direction.

It is not uncommon to estimate a number of related parameters, where one can think about them as having a common scale or average value. The example we worked with in a previous video introducing looping is an example of this, where we fit a growth model by estimating separate asymptotic lengths for each pond but allowed other parameters to be shared among ponds. In cases like this it is often the case that the estimates of closely related parameters are highly correlated. When parameter estimates are highly correlated, automated routines for finding the best fitting parameters can run into difficulties and sometimes fail. Also, procedures that account for uncertainty in estimates can have difficulties when the adjusted parameters are highly correlated. Part of the difficulty is that to adjust the overall scale the parameter adjustment routine has to use numerical methods to figure out that all related parameters need to be changed in the same direction.

PARAMETER_SECTION

```
init_number log_K
init_number log_sigma
init_number t0
init_vector log_Linfs(1,nponds)

number K
number sigma
vector Linfs(1,nponds)
matrix pred_lengths(fage,lage,1,nponds);
matrix resids(fage,lage,1,nponds);
```

A presentation slide with a green header and footer. The header contains a 'Next' button. The main content area has a white background with blue and green wavy borders at the top and bottom. The title 'ADMB Approach' is in large blue font. Below it, a definition is in bold black font. The variable name 'init_bounded_dev_vector' is in large blue font. The footer contains the Quantitative Fisheries Center logo and Michigan State University logo.

Next

ADMB Approach

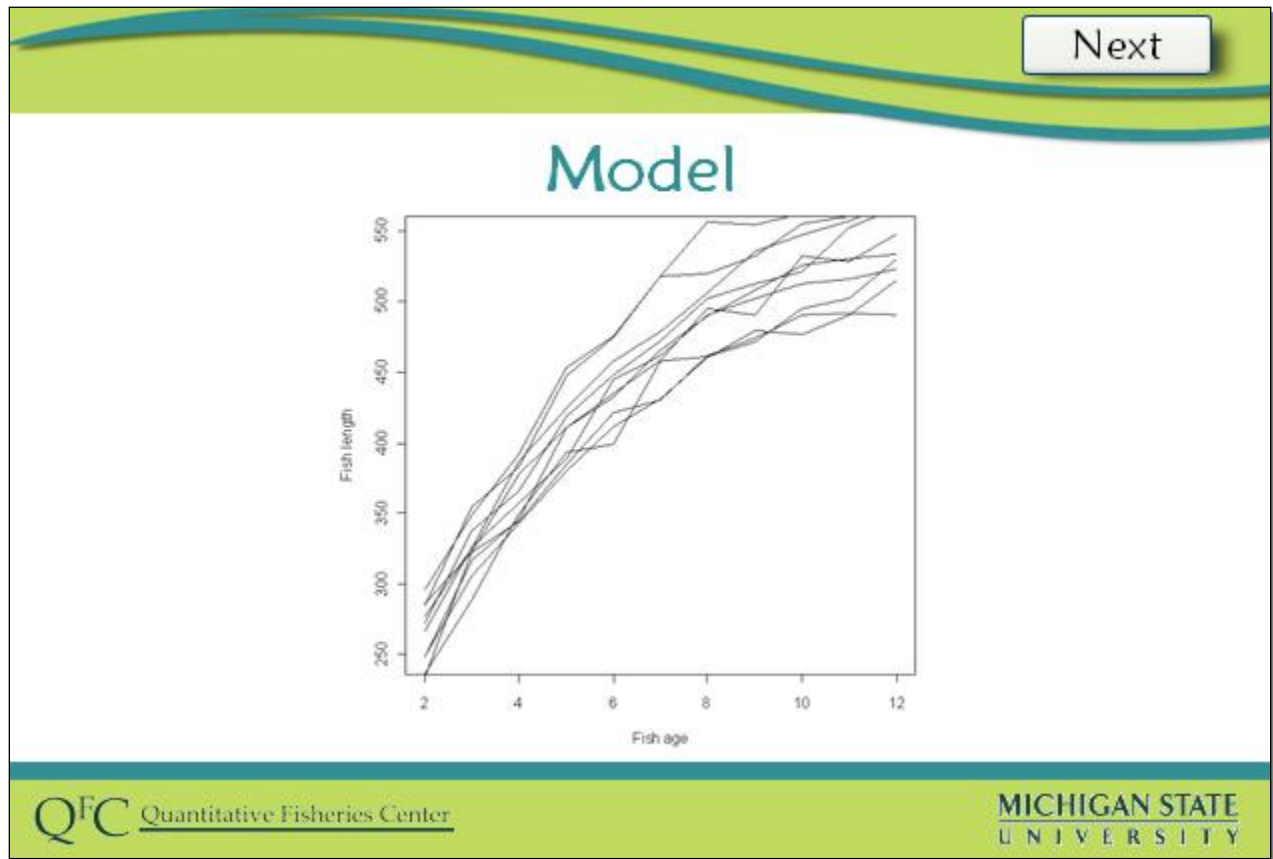
Variable type that parameterizes a model in terms of a mean and deviations from the mean.

init_bounded_dev_vector

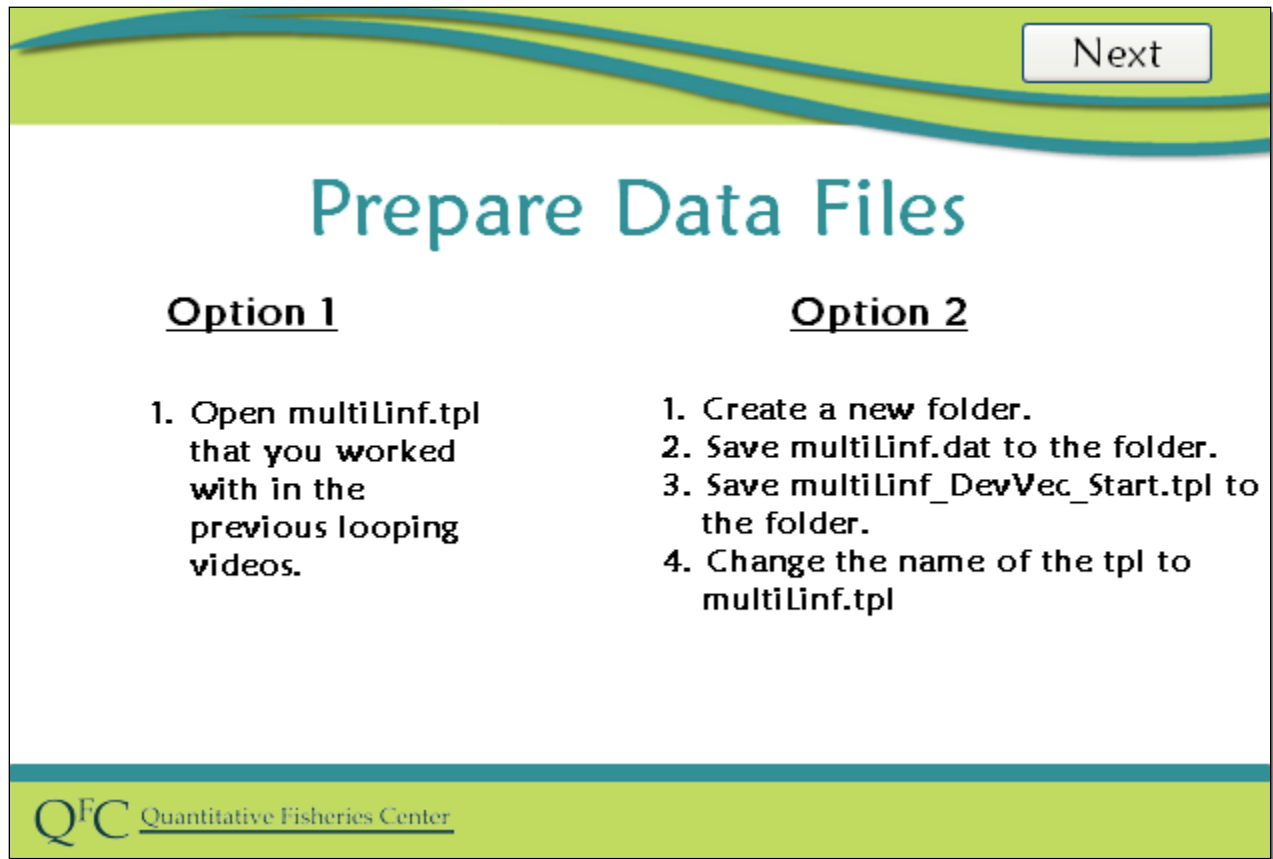
Q^{FC} Quantitative Fisheries Center

MICHIGAN STATE UNIVERSITY

AD Model builder has a built in variable type that allows us to easily parameterize a model in terms of a mean and deviations from the mean. This new variable type is specified by `init_bounded_dev_vector`. The syntax is essentially the same as using the usual `init_bounded_vector`. This particular option does not come in an unbounded version.



We will illustrate its use by modifying the version of the growth model we developed for a previous video where each of ten ponds had its own asymptotic length of L infinity. Previously we estimated an asymptotic length for each pond on a log scale. We will still estimate on a log-scale, but now we estimate an average value and a deviation for each pond. By using a bounded dev vector we can force the sum of the estimated deviations to sum to zero. It may seem like we have added a parameter because previously we estimated 10 L -infinities and now we estimate an average and ten deviations. However what we are doing is equivalent because we have added a constraint, that the deviations sum to zero, which is effectively like estimating one less quantity.



Next

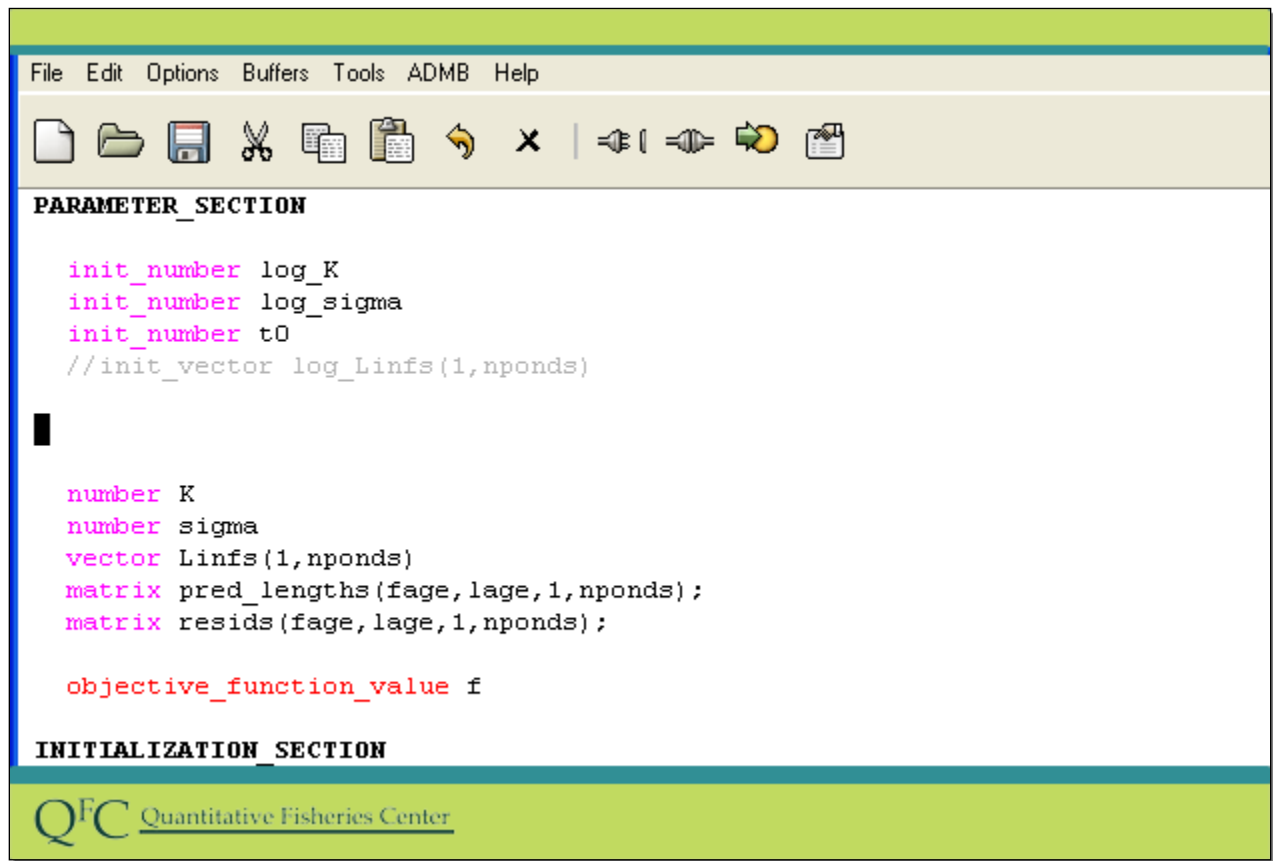
Prepare Data Files

<u>Option 1</u>	<u>Option 2</u>
<ol style="list-style-type: none">1. Open multiLinf.tpl that you worked with in the previous looping videos.	<ol style="list-style-type: none">1. Create a new folder.2. Save multiLinf.dat to the folder.3. Save multiLinf_DevVec_Start.tpl to the folder.4. Change the name of the tpl to multiLinf.tpl

QFC Quantitative Fisheries Center

Prepare and open your files. Either open multiLinf.tpl that you worked with in the previous looping videos or:

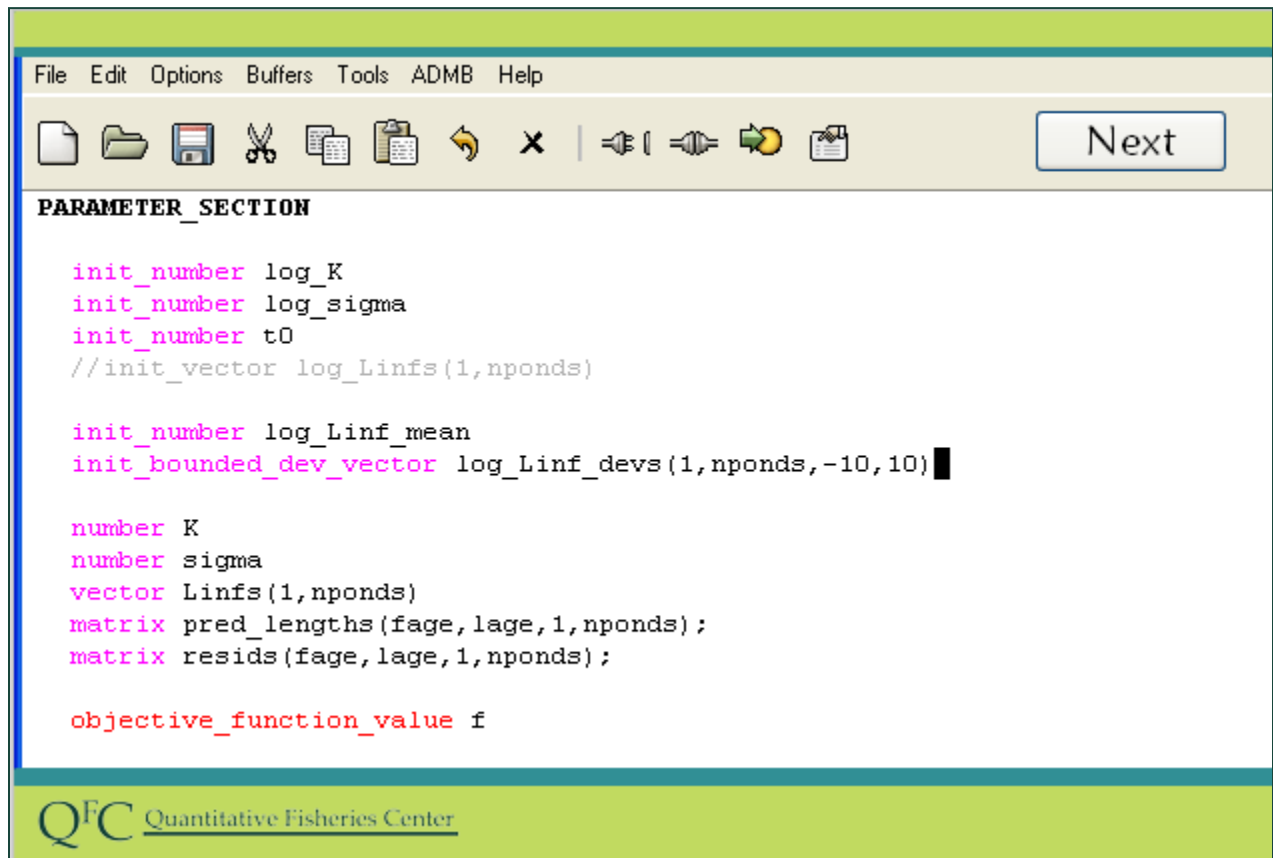
1. Create a new folder.
2. Save multiLinf.dat to the folder.
3. Save multiLinf_DevVec_Start.tpl to the folder.
4. Change the name of the tpl to multiLinf.tpl
5. Open multiLinf.tpl. You should also have access to the dat file multiLinf.dat in the same directory.
6. Click next when you are ready.



Scroll down to the parameter section where the L infinity parameters were previously defined. Comment them out.

Slide Code:

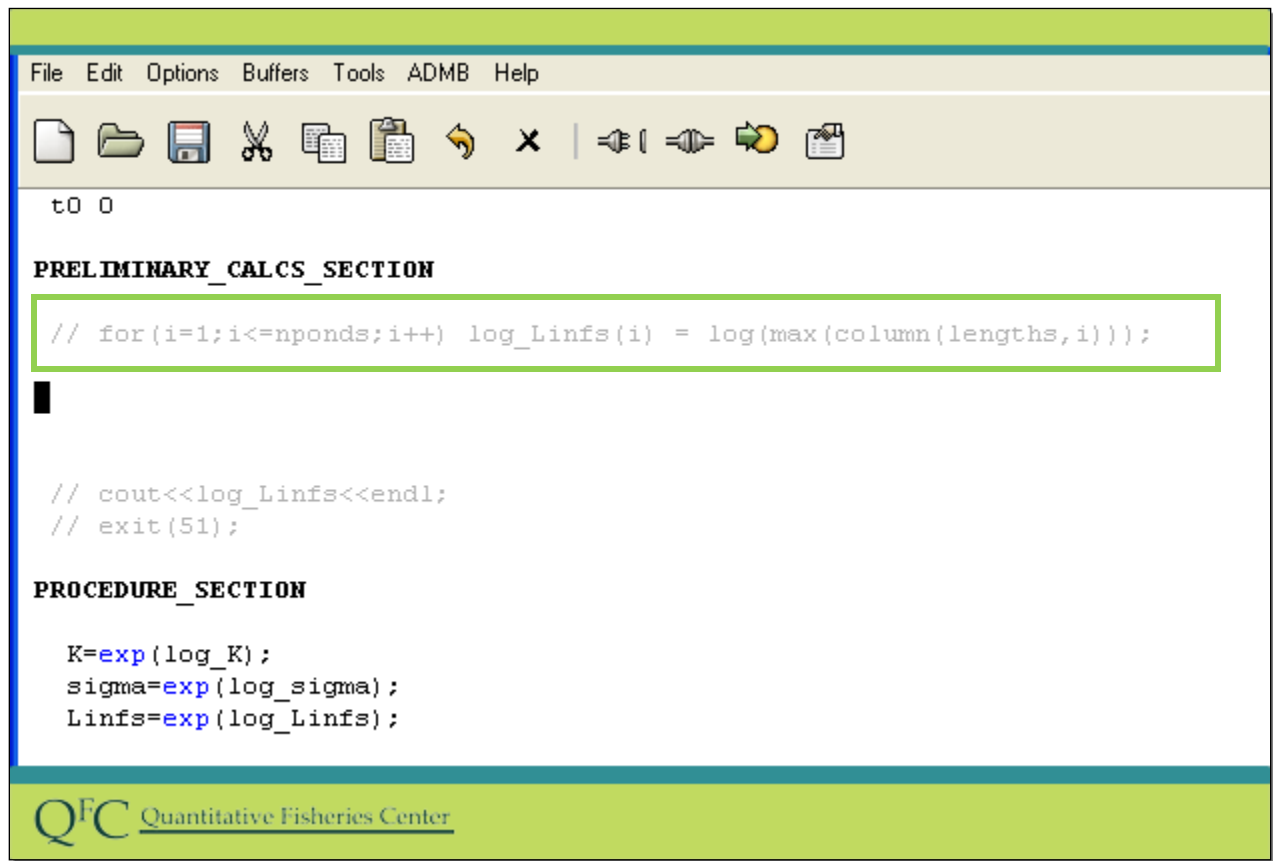
```
// init_vector log_Linfs(1,nponds)
```



Replace the previous parameterization with a new one in terms of a mean and deviations. We chose the bounds of minus 10 and 10 so that these bounds are very broad. On the backtransformed scale individual L-infinity values can range from approximately 20,000 times less to 20,000 times more than the estimated average value.

Slide Code:

```
init_number log_Linf_mean
init_bounded_dev_vector log_Linf_devs(1,nponds,-10,10)
```



```
t0 0

PRELIMINARY_CALCS_SECTION

// for(i=1;i<=nponds;i++) log_Linfs(i) = log(max(column(lengths,i)));

// cout<<log_Linfs<<endl;
// exit(51;

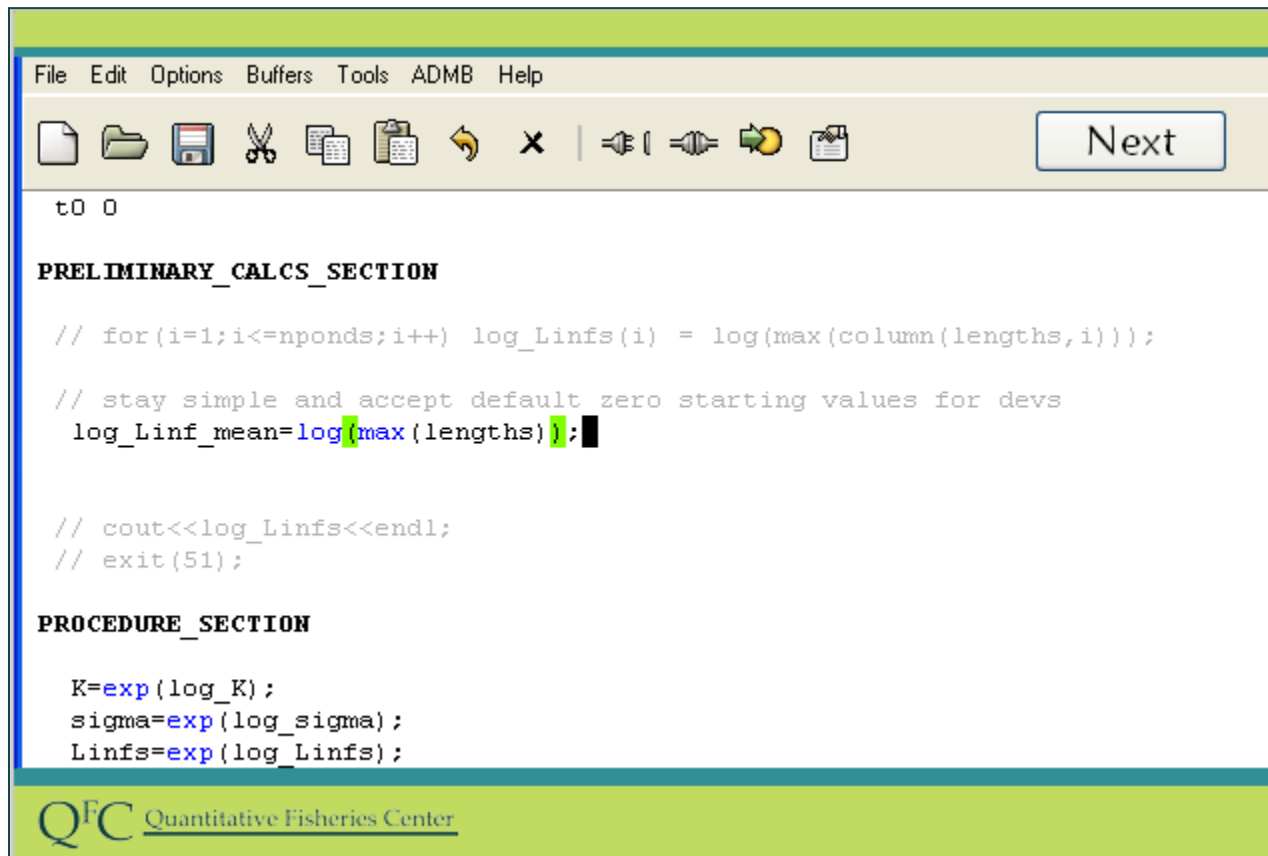
PROCEDURE_SECTION

K=exp(log_K);
sigma=exp(log_sigma);
Linfs=exp(log_Linfs);
```

Now go down to the PRELIMINARY_CALCS_SECTION and comment out the loop. Previous versions of code here were coming up different starting values for L-infinity for each pond based on the observed maximum length observed in each pond.

Slide Code:

```
// for(i=1;i<=nponds;i++) log_Linfs(i) = log(max(column(lengths,i)));
```

```
File Edit Options Buffers Tools ADMB Help

to 0

PRELIMINARY_CALCS_SECTION

// for(i=1;i<=nponds;i++) log_Linf(i) = log(max(column(lengths,i)));

// stay simple and accept default zero starting values for devs
log_Linf_mean=log(max(lengths));

// cout<<log_Linf<<endl;
// exit(51);

PROCEDURE_SECTION

K=exp(log_K);
sigma=exp(log_sigma);
Linf=exp(log_Linf);

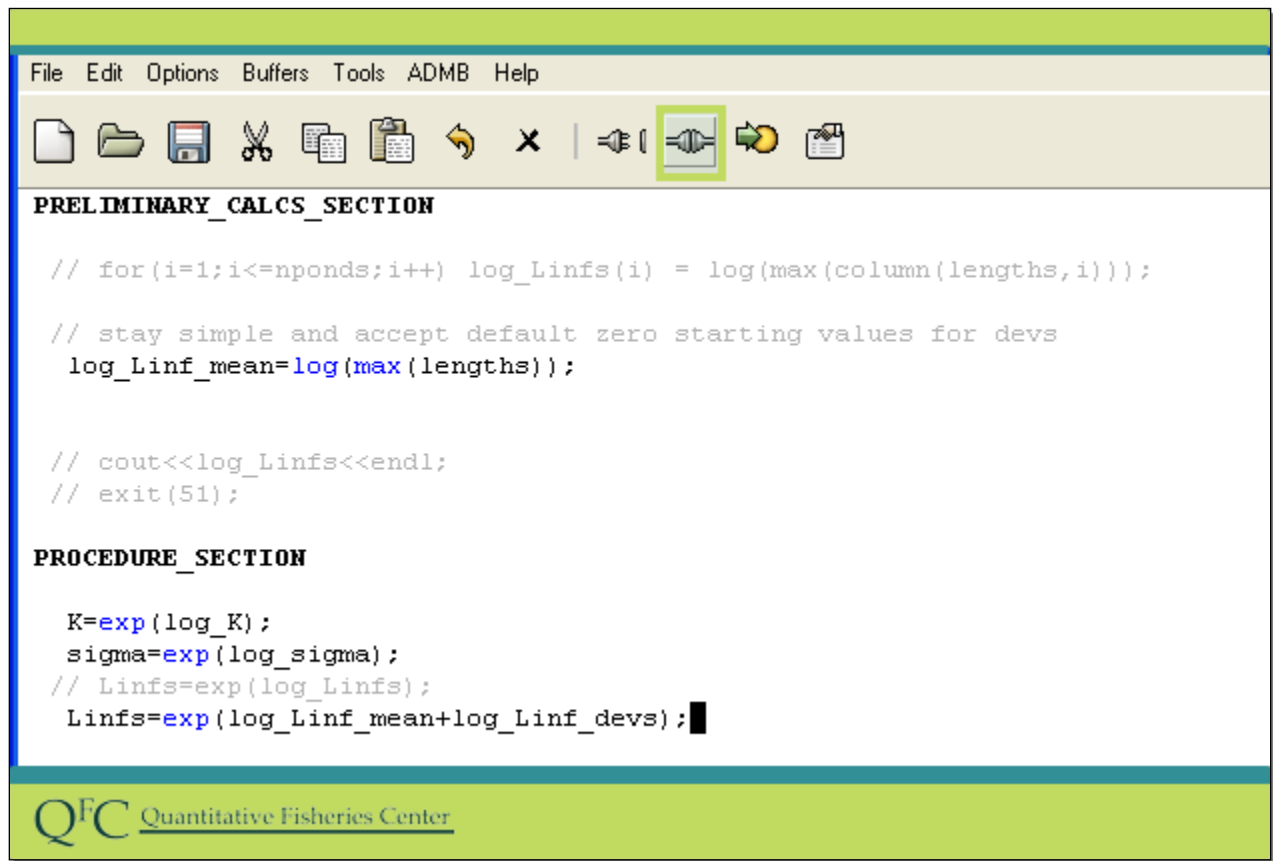
QFC Quantitative Fisheries Center
```

If we wanted to get fancy we could figure out how to convert these original starting values into a mean and deviations.

To keep things simple we will simply set the starting value for the scale parameter `log_Linf_mean` to the log of the maximum observed length. We will not explicitly set the starting values of the deviations parameters. By default these take starting values of zero. The usual starting value for parameters with bounds is half way between the bounds. But here because they are constrained to sum to zero they cannot all take the same non-zero value.

Slide Code:

```
// stay simple and accept default zero starting values for devs
log_Linf_mean = log(max(lengths));
```

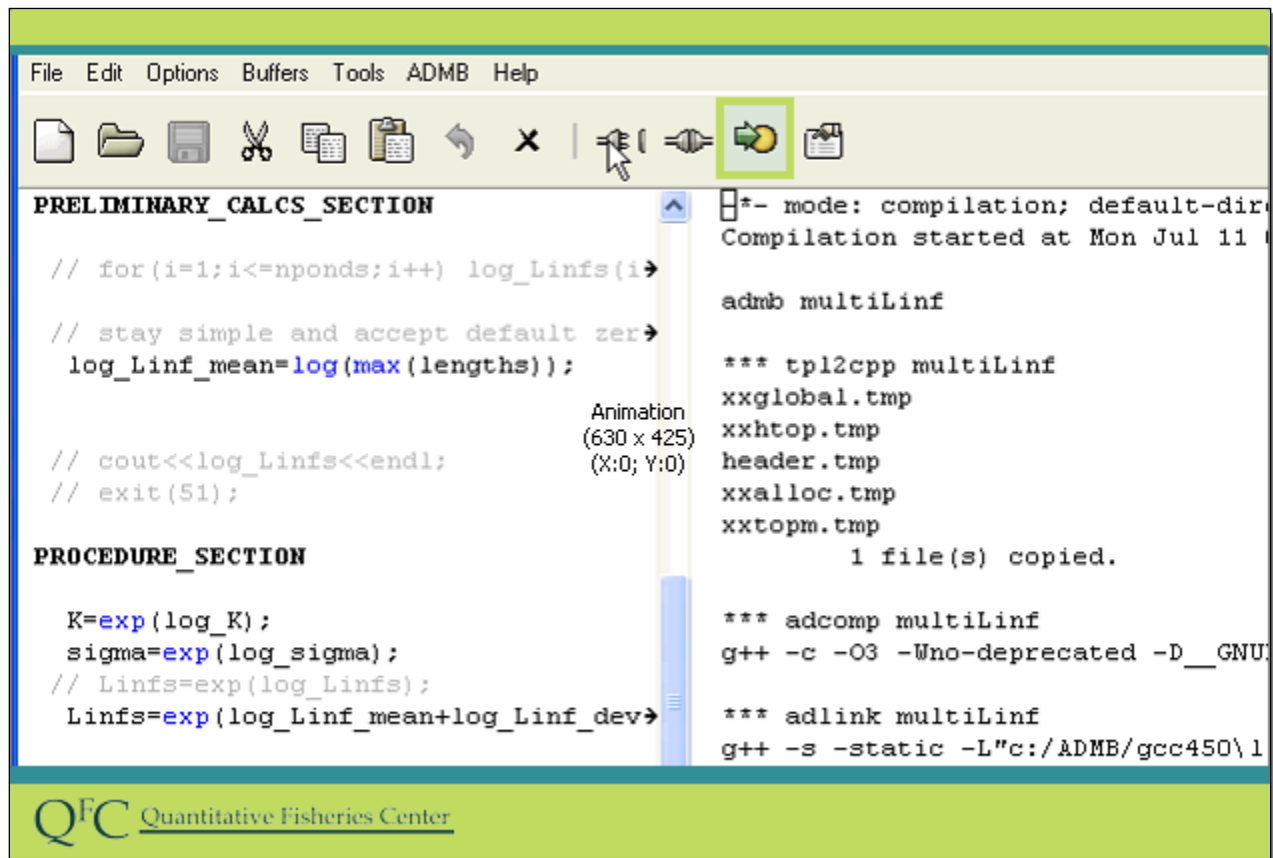


1. We now scroll down to the procedure section and comment out the previous line that calculated the back transformed L-infinities from the log-scale ones and
2. add code to calculate the back-transformed L-infinities from the new parameterization
3. Now we will build the revised growth model.

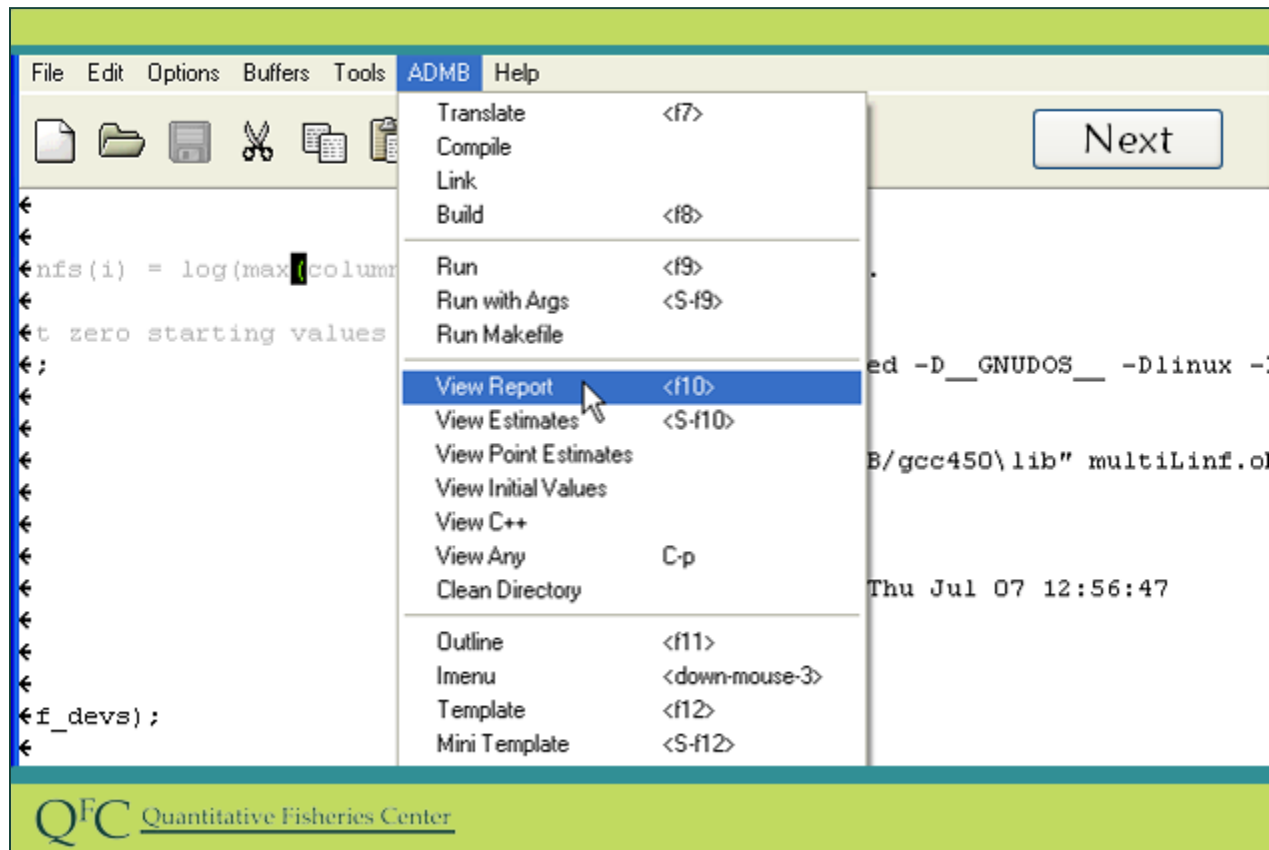
Slide Code:

```
// Linfs=exp(log_Linfs);
Linfs=exp(log_Linf_mean+log_Linf_devs);
```

And rebuild!



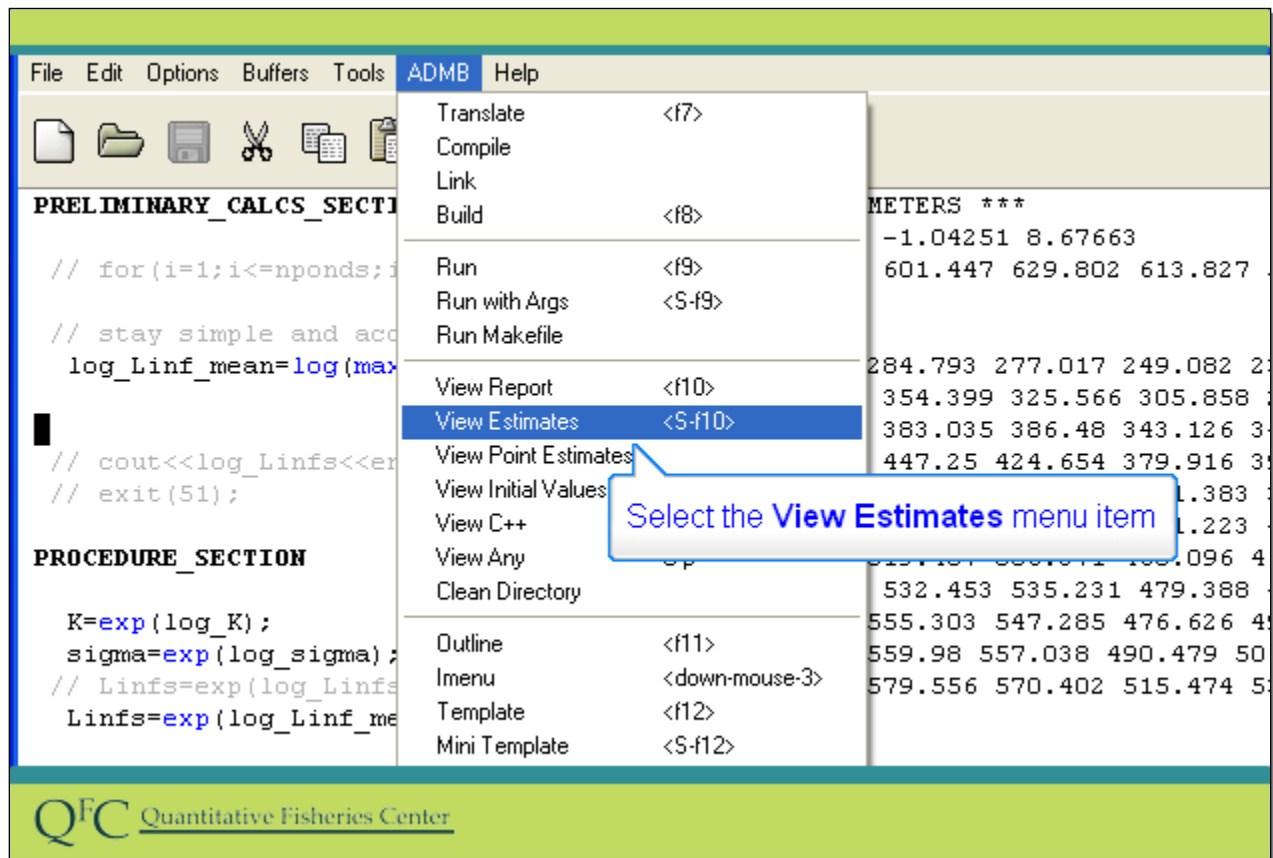
It appears to have been built successfully. Now we run the program



We quickly look at the report file. What we see here is essentially the same as before. Our program was already working and all we did was reparameterize the model into a more robust but equivalent form. So all the point estimates are unchanged and produce the same quantities such as expected lengths at age.

Slide Action:

View report file by clicking on the ADMB menu and selecting View Report



The observable difference is in the level of correlation among the estimated parameters. Select ADMB then view estimates

Slide Action:

View file by clicking on the ADMB menu and selecting View Estimates

Tools Help

Next

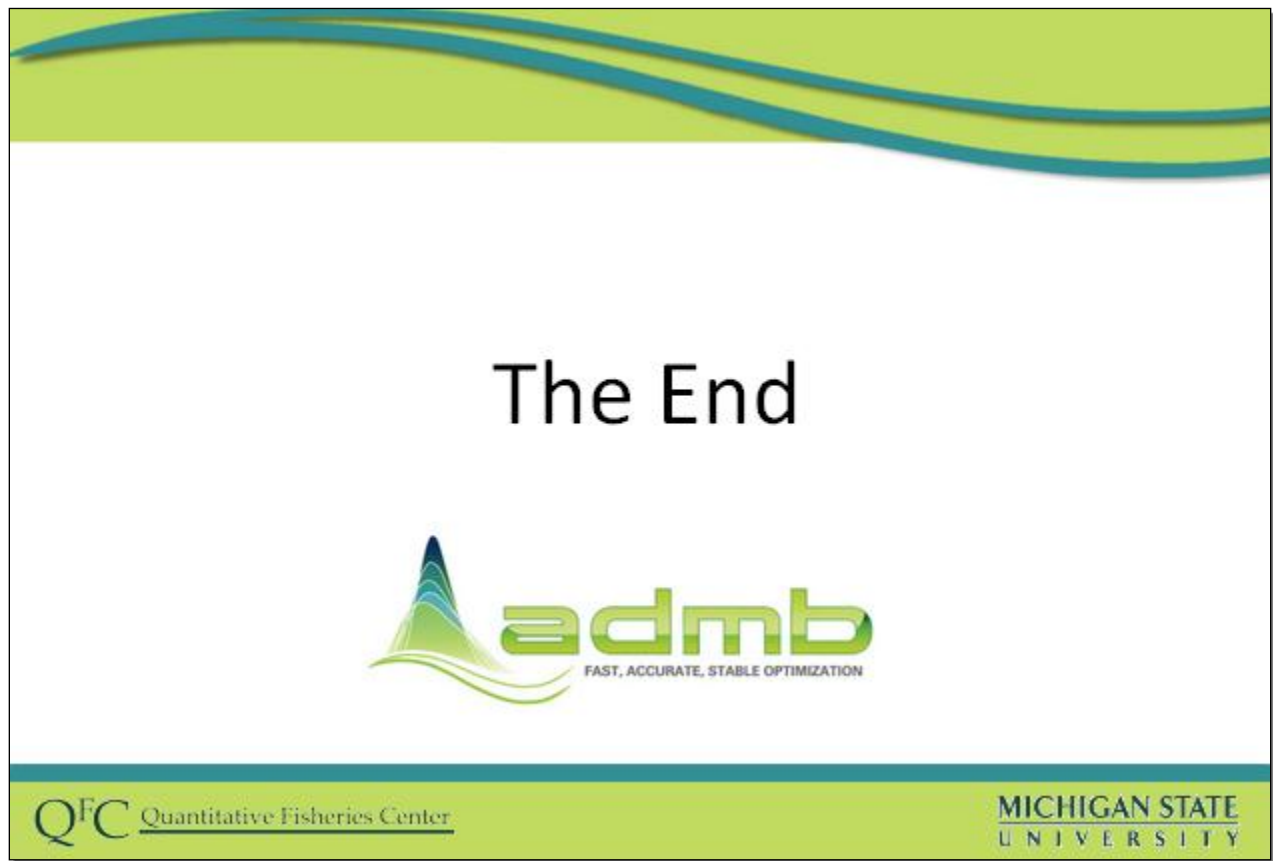
arithmetic of the determinant of the hessian = 131.422

name	value	std dev	1	2	3	4	5
log_K	-1.6305e+000	3.8181e-002	1.0000				
log_sigma	2.1606e+000	6.7420e-002	0.0000	1.0000			
t0	-1.0425e+000	1.0806e-001	0.9400	0.0000	1.0000		
log_Linf_mean	6.3786e+000	9.2451e-003	-0.9550	-0.0000	-0.8227	1.0000	
log_Linf_devs	-9.2889e-003	8.9897e-003	-0.0004	-0.0000	0.0005	0.0015	1.0000
log_Linf_devs	-1.7153e-002	9.0138e-003	0.0006	-0.0000	0.0001	0.0003	0.573
log_Linf_devs	2.0785e-002	8.9001e-003	-0.0016	0.0000	-0.0015	-0.0004	0.5840
log_Linf_devs	5.6852e-002	8.7715e-003	0.0011	0.0000	0.0006	-0.0068	0.5962
log_Linf_devs	4.1160e-002	8.8420e-003	-0.0009	0.0000	-0.0006	-0.0024	0.5895
log_Linf_devs	-6.9284e-002	9.1821e-003	-0.0005	0.0000	-0.0005	0.0060	0.558
log_Linf_devs	-5.6845e-002	9.1406e-003	0.0001	-0.0000	0.0010	0.0051	0.562
log_Linf_devs	3.1611e-004	8.9591e-003	-0.0010	-0.0000	-0.0019	0.0001	0.5785
log_Linf_devs	-6.9388e-002	9.1825e-003	0.0027	-0.0000	0.0021	0.0029	0.558
log_Linf_devs	9.2346e-002	8.7045e-003	-0.0001	-0.0000	-0.0000	-0.0071	0.6027

QFC Quantitative Fisheries Center

Here is the correlation matrix. These are the correlations between log_Linf_devs and log_Linf_mean.

Now look at the correlations between one deviation and another. These tend to be generally below point six. If you were to change the program back and rerun it in its original form the correlations among the original log scale asymptotic lengths were typically zero point 7 or larger. The reduction in correlations this time were not huge but sometimes it can be much larger and sometimes it can make the difference between even being able to reliably converge on the maximum likelihood estimates of the parameters.



You have now completed the video on default convergence criteria.