ATNmcmc Documentation

Contents:

Introduction

This document is designed to be a brief guide to the code I wrote to enable me to produce the C++ scripts I needed for my MSc thesis on the off chance that somebody wants to use it again. It would be useful for anybody seeking to fit lotka-volterrra type models of complex food webs using MCMC that need the speed of C++. It is highly flexible, but in order to do much science the C++ scripts produced will need to be adjusted to some extent. Basic adjustments do not need a great understanding of C++ however.

If anything is particularly unclear, get in contact through github. I have relatively limited experience with c++, but would be very happy to clear up any ambiguity with the structure of how the model is implemented.

The model type is the ATN model, see http://rspb.royalsocietypublishing.org/content/280/1770/20131901.long for an open-access paper showing the model.

The script here itself runs in R, and produces a script that runs in C++.

When I was writing the function I made a fair bit of effort to make it as flexible as possible but there are a large number of possible setting permuatations and I could not test them all. I have tried to list below which settings are incompatible with each other, but this list is unlikely to be complete. If other settings are needed it should not be too hard to edit the function I have made to suit your purposes.

This guide is written for somebody completely new to C++, as I was when I started this project.

Windows Setup

1. Visual Studio Express 2010 is free and an easy platform on which to run C++, which Filzbach (the MCMC package) is designed to work with. Download it from the Microsoft website http://www.microsoft.com/visualstudio/en-us/products/2010-editions/visual-cpp-express
2. Download and unzip the Filzbach package from www.research.microsoft.com/en-us/um/cambridge/groups/science/tools/filzbach/filzbach.htm . This folder will form the core of the system.
3. The documentation included with Filzbach is extremely good and explains how it works and the outputs. It is well worth going through.
4. There is a strange bug that stops these things working nicely. You need to set: Project → Properties -> Configuration Properties -> Linker (General) -> Enable Incremental Linking -> "No (/INCREMENTAL:NO)"
5. Install boost package to access the odeint package. (http://www.boost.org/) and put it in the include/ folder.

The scriptmaker function itself

When called the function will create a .txt file to put into visual studio. While it running it will output its inputs and processes to the screen to enable checking that everything is going ok,

The arguments (with defaults if specficed in [ ])are:

Title

Title of the analysis. This will be the first part of the file names produced by filzbach

Matrix

A predation matrix, derived from a .csv file, with row names in first column, and column names in first row. 1's indicate predation, 0's indicate no predation. See example. Only Predators need have their own column

Species

A .csv file with information on the body masses of the species, see template and example.

Years

Number of years/seasons of data.

Timesteps

Number of sample points per year

TimeUnit

Unit of time as an interval between each sample point.

Data

A list of tab deliminated .txt files, with species in the header, containing the population dynamic data. Not needed if using Fake Data. 0's are not allowed due to log-normal measurement model – replace with a suitable small number

GlobalParameters

A .csv of the global parameter means and bounds

AssimEf [=2]

Inverse Assimilation efficiency inserted into formula.

MakeFake [= F]

T/F. If set to True, will produce and fit to an artificial data set created using the parameters in the species file. Incompatible with multiyear

SeperateParams [= T]

If set to false, consumers and producers share a single aT, aJ and aR n[ie are completely allometrically constrained), If you want to make fake data the values wanted will need to be specified maually.

AbsoluteError [=1.0e-12]

Allowed error of the ODEsolver stepper function

RelativeError [=1.0e-6]

Allowed error of the ODEsolver stepper function

Pset [=-1]

Set global control of dynamic parameters as per Filzbach: 0=randomly choose start between bounds, -1=fix start at what is in table, 1= all parameters fixed (can then go and manually loosen a few)

STARTTYPE [=1]

1= will start all the start points at the first data value, bounds set by table (unlikely to be valid for multiyear!)

2= will start all the start points set in the parameter file (unlikely to be valid for multiyear!)

3= will start all the start points at the first data value, bounds given % leeway either side.

StartBounds [=5]

% leeway for bounds when using start type = 3.

StartFixed [=1]

Are start points fixed or flexible, following filzbach settings

RejectNegs [= F]

Set to 'True' to include a part of the likelihood that will cause immediate rejection of any parameters that result in negative populations or unrealistically large populations. This is most commonly caused by solver steps being too small leading to intergration errors, so look for this first and be careful!

MaxForReject [= 1000]

Level if above the population goes above the likelihood is stopped (if RejectNegs = T)

MCMC [=c(1000,1000,0,0,1)]

List of number of iterations for each of Filzbach stages (Burn-in, Bayesian Sampling, Maximum Likelihood adjustments, Maximum Likelihood search, Number of Chains)

Example use:

```
scriptmaker(
  Title='ATNmcmcExample',
  Matrix= as.matrix(read.csv('ExamplePredationMatrix.csv', header =T, row.names=1)),
  Species= read.csv('ExampleSpeciesTable.csv',header=T),
  Years=1,
  Timesteps =14,
  TimeUnit = 0.1,
  Data ='ExampleTimeSeries.txt',
  GlobalParameters=read.csv('ExampleGlobalParameters.csv', header=T) ,
  AssimEf=2,
  MakeFake = F,
  SeperateParams = T,
  AbsoluteError=1.0e-12,
  RelativeError=1.0e-6,
  Pset=-1,
  STARTTYPE=1,
  StartBounds=5,
  StartFixed=1,
  RejectNegs = F,
  MaxForReject= 1000,
  MCMC=c(10000,10000,0,0,1)
)
```

The input and output files of this example are included in the repository. There is also a much longer example which was one of the runs from my thesis which includes 17 species, 5 years of data and prior information.

Additional tweaks that may be useful

- To add priors use the “parameter_addprior()” function directly after the parameter_create section following the filzbach manual
- It is often advantageous to run a short run with perhaps only a couple of thousand interations first before editting the runmcmc() function at the end of the C++ script to do a longer run.
-

Workflow to run model fitting.

1. Create the C++ script with the R script
2. Open up the Filzbach Example.sln with visual studio.
3. File>New>File>Visual C++>C++File Open
4. Paste in the C++ script
5. File> Move Source1.cpp into Filzbach and save as whatever you want.
6. Edit examples.h to include the run title, (remember to have only one uncommented at a time)
7. Start Debug [Green arrow]
8. The program will then compile and run. It asks to
9. Outputs will be produced in the workspace folder. The easiest way to analyse them is just to drag them into Excel. See the filzbach documentation for interpetation.

Hope this is useful!

Chris