Posterior Simulator with 'saveEvery=' Option

more_on_poster_simulator.m

by Jaromir Benes

May 27, 2015

Summary

In this file, we show two features of convenience when running larger posterior simulations. First, the posterior simulator can be run with the option 'saveEvery=' to split the simulated posterior chain into smaller bits and saving them each in a separate data file. This is a way to get around possible out-of-memory problems when simulating larger models and/or longer chains. Second, a large posterior simulation can executed incrementally in smaller chunks, with the final state of one simulation being used as the initial state for the next one.

Contents

1 Clear Workspace

Clear workspace, close all graphics figures, clear command window, and check the IRIS version.

```
clear;
close all;
clc;
irisrequired 20140612;
```

2 Load Posterior Simulator Object

Load the posterior object created when maximising the posterior mode in `estimate_params`. Run `estimate_params` at least once before running this m-file.

```
load estimate_params.mat pos;
```

3 Run Posterior Simulator and Statistics

Reset the random number generator and run the posterior simulator the normal way (because this is just an illustration of how the functions work, keep the number of draws). Then compute some of the posterior statistics.

```
N = 100;

rng(0);

disp('Run the posterior simulator at once');
tic();
[theta1,logpost1,ar1,pos1] = arwm(pos,N,'progress=',true); %#ok<ASGLU>
toc();
```

```
Run the posterior simulator at once
[--IRIS poster.arwm progress-------------]
[****************************************]
Elapsed time is 2.215404 seconds.
```

4 Run Again Saving Every N Draws

Reset the random number generator again to reproduce the above numbers, and run the posterior simulator saving now every 20 draws in a separate HDF5 (hierarchical data file). Note that you must assign a valid file name through the option 'saveAs=' whenever using 'saveEvery='.

```
54 if exist('myposter.h5','file')
55     delete('myposter.h5');
56 end
57
58 rng(0);
59
60 N = 100;
61 disp('Run the posterior simulator saving every 20 draws');
62
63 tic();
64 arwm(pos,N,'progress=',true,'saveEvery=',N,'saveAs=','myposter.h5');
65 toc();
```

```
Run the posterior simulator saving every 20 draws
[--IRIS poster.arwm progress-------------]
[**************************************]
Elapsed time is 2.615668 seconds.
```

5 Compute Posterior Statistics

To compute the posterior statistics, use the function 'stats' and pass in

- either the simulated posterior chain, theta1, and posterior log densities, logpost1;
- or the file name under which the batches where saved when running arwm with the option 'saveevery=', i.e. 'myposterior' in our example.

In a real-life simulation, remember to exclude 'chain' from the list of requested outputs in stats in the latter case, i.e. add the option 'chain=' false. You use the option 'saveEvery=' to breake the simulated chain down into smaller bits because the length of the chain would be overwhelming for your computer memory; you don't therefore want the chains to be restored in full length.

```
85 tic();
86 stats1 = stats(pos,theta1,logpost1,'mode=',true,'cov=',true);
87 toc();
```

```
88
89  tic();
90  stats2 = stats(pos,'myposter.h5','mode=',true,'cov=',true);
91  toc();
92  disp(' ');
```

```
Elapsed time is 0.013980 seconds.
Elapsed time is 0.493881 seconds.
```

6 Compare Results

Display the max abs differences between the chains simulated in a plain run of the posterior simulatior, and in a run with the 'saveevery=' option.

```
100 disp('Compare the two runs of the posterior simulator')
101 disp('Max discrepancy in simulate chain, mean, and std devs');
102 maxabs(stats1.chain,stats2.chain) ...
103     & maxabs(stats1.mean,stats2.mean) ...
104     & maxabs(stats1.std,stats2.std) %#ok<NOPTS>
105
106 disp('Max discrepancy in covariance matrix');
107 maxabs(stats1.cov,stats2.cov)
```

```
Compare the two runs of the posterior simulator
Max discrepancy in simulate chain, mean, and std devs
ans =
            chi: [0 0 0]
            xiw: [0 0 0]
            xip: [0 0 0]
           rhor: [0 0 0]
         kappap: [0 0 0]
         kappan: [0 0 0]
         std_Ep: [0 0 0]
         std_Ew: [0 0 0]
         std_Ea: [0 0 0]
         std_Er: [0 0 0]
    corr_Er__Ep: [0 0 0]
Max discrepancy in covariance matrix
ans =
     0
```

7 Incremental Runs of Posterior Simulator

First, run a posterior simulation of 100 draws with 20 burn-ins. Then, run the same simulation split into two steps. Using the posterior object returned from the first to initialize the second one reproduces exactly the results of the original simulation.

The second incremental simulation, 4, is based on the posterior object pos21 returned from the first simulation, 2, the third is based on the simulation object from the second, etc. This is the way to initialize the posterior simulation by the final results obtained in the previous step.

Note that the number of burn-ins must be set to the original number (i.e. 50) 1 in the very first simulation 3, and to zero in all subsequent simulations 5.

Simulate 300 draws with 50 burn-ins at the beginning.

```
130 rng(1);
131 [theta1,logpost1,ar1] = arwm(pos,300, ...
132     'progress=',true, ...
133     'burnin=',50); 1
```

```
[--IRIS poster.arwm progress-------------]
[****************************************]
```

Simulate 300 draw incrementally (by 100 in each simulation).

```
139 rng(1);
140 [theta21,logpost21,ar21,pos21] = arwm(pos,100, ... 2
141     'progress=',true, ...
142     'burnin=',50); 3
143
144 [theta22,logpost22,ar22,pos22] = arwm(pos21,100, ... 4
145     'progress=',true, ...
146     'burnin=',0); 5
147
148 [theta23,logpost23,ar23,pos23] = arwm(pos22,100, ...
149     'progress=',true, ...
150     'burnin=',0); 5
```

```
[--IRIS poster.arwm progress-------------]
[****************************************]
[--IRIS poster.arwm progress-------------]
[****************************************]
[--IRIS poster.arwm progress-------------]
[****************************************]
```

Combine the three simulation results.

```
theta2 = [theta21,theta22,theta23];
logpost2 = [logpost21,logpost22,logpost23];
ar2 = [ar21,ar22,ar23];
```

Verify that the original and the incremental simulation results are identical (up to rounding errors).

```
disp('Max discrepancy in simulated chain');
maxabs(theta1,theta2)

disp('Max discrepancy in log posterior density');
maxabs(logpost1,logpost2)

disp('Max discrepancy in cumulative acceptance ratios');
maxabs(ar1,ar2)
```

```
Max discrepancy in simulated chain
ans =
     0
Max discrepancy in log posterior density
ans =
     0
Max discrepancy in cumulative acceptance ratios
ans =
     0
```

Look into the posterior objects as they are updated throughout the incremental simulations.

```
pos %#ok<NOPTS>
pos21 %#ok<NOPTS>
pos22 %#ok<NOPTS>
pos23 %#ok<NOPTS>
```

```
pos =
  poster with properties:

              ParamList: {1x11 cell}
       MinusLogPostFunc: @objfunc
    MinusLogPostFuncArgs: {1x5 cell}
         MinusLogLikFunc: []
     MinusLogLikFuncArgs: {}
           LogPriorFunc: {[]  []  []  []  []  []  []  []  []  []  []}
```

```
             InitLogPost: -315.2187
               InitParam: [1x11 double]
         InitProposalCov: [11x11 double]
        InitProposalChol: []
               InitScale: 0.3333
               InitCount: [0 0 0]
             LowerBounds: [1x11 double]
             UpperBounds: [1x11 double]
pos21 =
  poster with properties:

               ParamList: {1x11 cell}
        MinusLogPostFunc: @objfunc
    MinusLogPostFuncArgs: {1x5 cell}
         MinusLogLikFunc: []
     MinusLogLikFuncArgs: {}
            LogPriorFunc: {[]  []  []  []  []  []  []  []  []  []  []}
             InitLogPost: -320.8918
               InitParam: [1x11 double]
         InitProposalCov: [11x11 double]
        InitProposalChol: [11x11 double]
               InitScale: 0.6229
               InitCount: [150 22 50]
             LowerBounds: [1x11 double]
             UpperBounds: [1x11 double]
pos22 =
  poster with properties:

               ParamList: {1x11 cell}
        MinusLogPostFunc: @objfunc
    MinusLogPostFuncArgs: {1x5 cell}
         MinusLogLikFunc: []
     MinusLogLikFuncArgs: {}
            LogPriorFunc: {[]  []  []  []  []  []  []  []  []  []  []}
             InitLogPost: -326.1488
               InitParam: [1x11 double]
         InitProposalCov: [11x11 double]
        InitProposalChol: [11x11 double]
               InitScale: 0.6438
               InitCount: [250 51 50]
             LowerBounds: [1x11 double]
             UpperBounds: [1x11 double]
pos23 =
  poster with properties:

               ParamList: {1x11 cell}
```

```
        MinusLogPostFunc: @objfunc
    MinusLogPostFuncArgs: {1x5 cell}
         MinusLogLikFunc: []
     MinusLogLikFuncArgs: {}
            LogPriorFunc: {[]  []  []  []  []  []  []  []  []  []  []}
             InitLogPost: -321.5874
               InitParam: [1x11 double]
         InitProposalCov: [11x11 double]
        InitProposalChol: [11x11 double]
               InitScale: 0.7191
               InitCount: [350 85 50]
             LowerBounds: [1x11 double]
             UpperBounds: [1x11 double]
```

8 Help on IRIS Functions Used in This File

Use either `help` to display help in the command window, or `idoc` to display help in an HTML browser window.

```
help poster/arwm
help poster/stats
help maxabs
```