

MCMCthin (and Rcode script)

MCMC thinning utility for use with IM surface files (*.surf).

Code written by C. R. Young

Last updated: 4 Oct 2005

Monitoring the convergence of Markov chain Monte Carlo (MCMC) algorithms is essential. This thinning utility was written to aid users in monitoring the convergence of IM by making surface files a manageable size. The companion script (in "Rcode.txt") is a collection of R functions that aid in monitoring convergence and understanding the results of the analysis. It can be cut and pasted in full into the R console. The script will create several files containing useful graphics, and will perform a few formal diagnostics. R is a free software environment for statistical computing and graphics, and can be downloaded from: <http://www.r-project.org/>

MCMCthin

MCMCthin expects *at least* two files to be present in the same directory as the executable. The first file, "filelist.txt" is a text file containing a list of the names of all of the surface files that you wish to thin. *This file needs to end with a new line character.* All of the surface files listed in "filelist.txt" need to be present in the directory also. This utility can be executed while IM is running. In this case, run MCMCthin from the IM directory containing the surface files.

Input files (in same directory as MCMCthin):

"filelist.txt"	Contains a list of the names of the files that you want to read
"outfile.txt_0000.surf"	All of the surface files listed in "filelist.txt"
"outfile.txt_0001.surf"	
...	

Output file:

"MCMCdata.out"	Tab delimited file that contains the thinned dataset
----------------	--

Usage:

```
MCMCtest <burnin> <samplefreq> <Print iteration number? 1=yes, 0=no>
```

Options:

<burnin>	integer >= 0.	Ignores this many samples at the beginning of the run
<samplefreq>	integer > 0.	Subsample every <samplefreq> samples from file.
<Print iteration number? 1=yes, 0=no>		numbers thinned samples for plotting traces

Example of usage for 500,000 iteration burnin and sampling every 5000 iterations:

```
MCMCthin 500000 5000 1
```

NOTE: The surface files generated by IM do not contain the records from the burnin period specified during the IM run. However, if you find that the chain has not yet reached stationarity, the thinning option in MCMCthin can be used to increase the burnin. The Rcode script can then be used to analyze the MCMC dataset.

Rcode script: "Rcode.txt"

You will need to download and install R to use this script: <http://cran.cnr.berkeley.edu/>

This file contains a set of R functions that I find useful in diagnosing convergence of IM runs and in understanding the fitted model parameters, both by visualizing them and by examining numerical output. This script assumes that the file "MCMCdata.out" is present in the working directory (root by default) of R. This file is the output file generated by MCMCthin. I suggest using MCMCthin to generate an output file on the order of 5000 lines. It is also not necessary to have MCMCthin print the iteration number when using the output with this script.

The Rcode script can be copied and pasted (in full) into the R console. The script file contains some detail about the functions that are used. Numerical output will be generated, and when the script is finished running, you will need to scroll up to get to it. Look for the blue text. In addition, several plots (*.pdf) are saved to the root directory of R. Most of the plotting functions are fairly automatic, and the parameters of the plotting functions should be reasonable for most data. However, you may need to adjust the xlim and ylim in some plots. This script is meant to provide a starting point for analysis of IM runs, but you might need to tailor it to your particular data.

Output of this script include:

- Parameter traces over the length of the chain
- Cumulative quantile plots (0.025%, 50%, and 97.5%)
- Marginal kernel density estimates (KDEs)
- Joint KDEs
- 2D contour plots of joint KDEs
- Summary statistics of marginal distributions (including cross correlations and autocorrelations)
- Effective sample size estimates
- Geweke's convergence diagnostic
- Raftery and Lewis's diagnostic
- Gelman-Rubin diagnostic
- Heidelberger-Welch diagnostic

Passing MCMC diagnostics does NOT guarantee that a chain is stationary!!!!!!

You will need to make sure that the CODA and KernSmooth packages are installed in R before attempting to use this script.

To install CODA:

1. In the menu, go to Packages, then install packages
2. A list of servers will appear
3. Pick one of the servers
4. A list of packages will appear
5. Select the coda package
6. R will download a zip file and unpack it.

Repeat these steps for the KernSmooth package

Further instructions for installing packages in R can be found in the html help documentation of R.

Convergence Diagnostics:

Geweke

Geweke's diagnostic (1992) tries to identify whether a chain is stationary. The idea here is that if the chain is stationary, then the means of the first, say, 10% of the chain and the last, say, 50% of the chain should be the same. The test calculates a **z-score** for the difference in means between these two parts of the chain.

Heidelberger-Welch

The Heidelberger and Welch diagnostic (1983) uses the **Cramer-von-Mises** statistic to test the null hypothesis of a stationary distribution. The test is first applied to the whole chain, then, discarding values at the beginning of the chain, reapplied to 90% of the chain, 80%, ... , etc. until either the test passes, or 50% of the chain has been discarded – an overall failure. This test uses the portion of the chain that passed H-W stationary to compute 95% confidence intervals on parameter means. Half of the width of this interval is then compared to the mean. If the ratio between the **half-width** and the mean is lower than the specified “eps”, then the test passes. Note: the default setting is not very stringent.

Gelman-Rubin

The Gelman-Rubin diagnostic amounts to an analysis of variance among two or (preferably) more chains started from different (overdispersed) initial values. The idea here is to search for multimodality in the parameter space. If you start the chain from different places in the state space, you may find a local peak, and you may not be able to get off of that peak. Hopefully, by running multiple chains, you will be able to tell if you're getting stuck. Using the mean of the empirical variance within each chain and the empirical variance for all chains combined, The Gelman-Rubin diagnostic calculates a **shrink factor**. Values near one indicate convergence, and values **substantially above one** indicate a problem. It is also possible that the Gelman-Rubin diagnostic can fail if the shrink factor is close to one by random chance. CODA can plot the shrink factors as the chain progresses in an attempt to see if the shrink factor is still fluctuating.

NOTE: requires files "MCMCdata_chain1.out" and "MCMCdata_chain2.out" in the working directory of R.

You'll get an error for the Gelman-Rubin diagnostic unless you have two MCMC datasets named: "MCMCdata_chain1.out" and "MCMCdata_chain2.out". You'll have to use MCMCthin twice on the surface files of two different runs, then rename "MCMCdata.out" from run1 "MCMCdata_chain1.out" and rename "MCMCdata.out" from run2 "MCMCdata_chain2.out". Then, put both of these files in R's root directory.

Modifications to script are necessary if more chains are tested. Also note that the summary statistics that Rcode computes include data from both chains, so the script can be used as a brute force way to **parallelize** IM. You can run two (or more) instances of the same analysis, using the same parameters, on multiple machines, then use Rcode to assess convergence and then combine the results.

Raferty-Lewis

We can use Raftery and Lewis's diagnostic to estimate how long our chain needs to run in order to estimate quantiles within a specified accuracy with some specified probability. The **total length estimate** is the suggested length to run the chain for the desired level of accuracy. The **lower bound** tells you how many samples that you'd need if the chain were IID instead of autocorrelated. The **dependence factor** tells you if you have a particularly bad problem with mixing. Values above about 5 or so are problematic, and values around 1 or so indicate good mixing. The diagnostic also attempts to **suggest a burn-in**. This diagnostic will return an error if you haven't sampled enough for it to compute the N's. It will tell you, though, the minimum sample size that you need to run the diagnostic if this happens.

Further information on these and other functions can be obtained from the html documentation in R.

