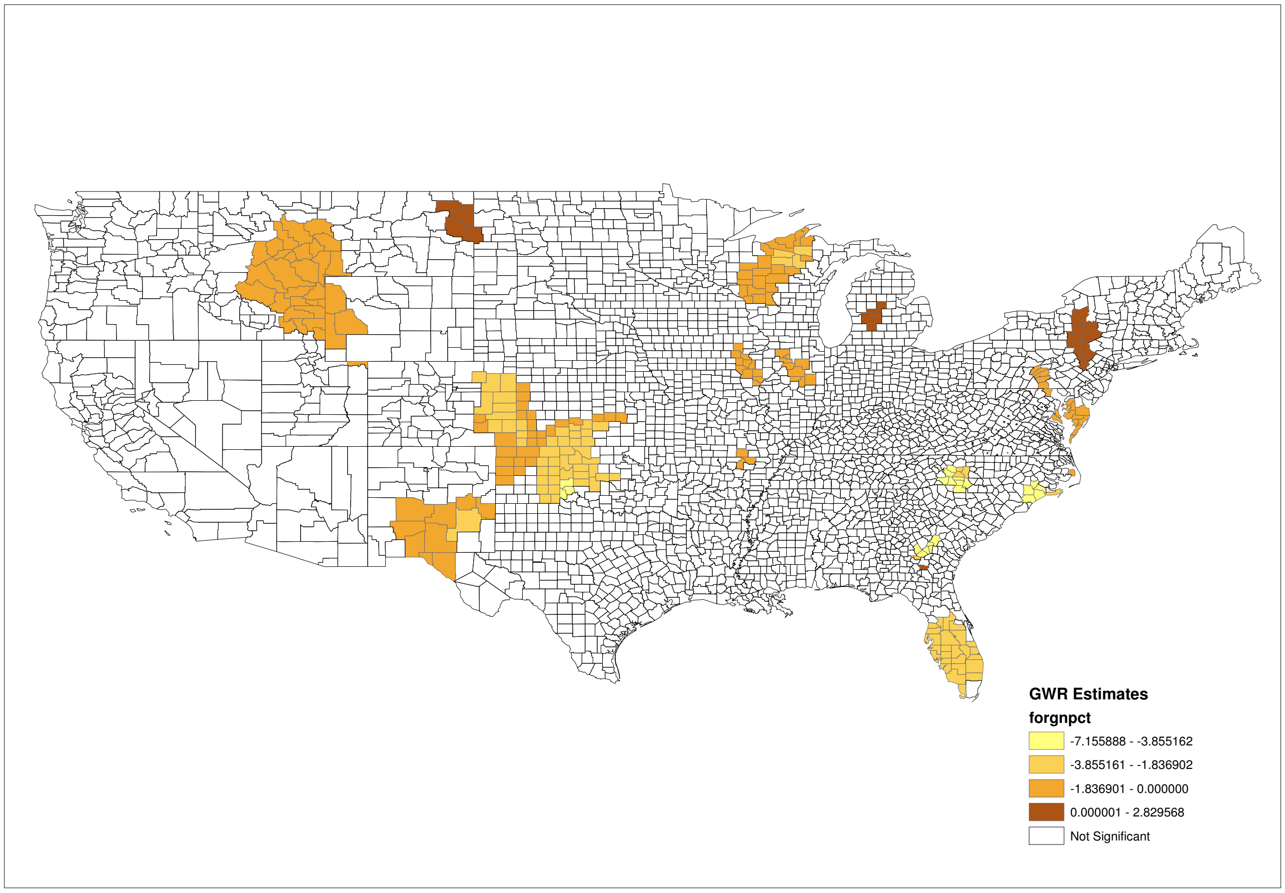
Figure 1: GWR Estimates



`superdiag`: A Comprehensive Test Suite for Markov Chain Non-Convergence

Tsung-han Tsai and Jeff Gill
Washington University in St. Louis
t.tsai@wustl.edu and *jgill@wustl.edu*

Overview

The use of Markov chain Monte Carlo to solve difficult estimation problems, both Bayesian and non-Bayesian, is now quite common in political science and related fields. This is because it is increasingly easy to set-up and run a Gibbs sampler or Metropolis-Hastings kernel to produce marginal posterior (sampling distribution) summaries for standard regression table construction. Programs like `WinBUGS` and `JAGS` automate the construction of the actual sampler from modeling statements dictated by the user. Both of these programs can also be called in multiple ways from `R`, which greatly simplifies the data-handling process. Yet we have noticed from conference papers, papers to review, and even published work, that the issue of Markov chain convergence is often poorly addressed.

A Markov chain that is not in its stationary (target) distribution does not produce valid empirical draws for inferential purposes. Therefore it is imperative that researchers perform sufficient analysis to assure themselves and their readers that the ergodic process is complete. This should involve multiple empirical diagnostics as well as graphical approaches to look for evidence of non-convergence. Unfortunately many authors provide no discussion of such checking or perhaps use only one tool. See Gill (2008) for specific advice and a discussion of various related problems.

We provide here an easy-to-use `R` function that integrates all of the standard empirical MCMC convergence diagnostics with one simple command. Users can simultaneously run functions supplied in menu form by the `R` suites `boa` and `coda`. Furthermore, the function automatically alters parameter choices, e.g. window definitions in the Geweke diagnostic, which are important features of the tests. Our aim is to help improve convergence testing in political science by making the mechanics much easier.

The Commonly Used Convergence Diagnostics

Four tests dominate practice: Geweke, Gelman-Rubin, Raftery-Lewis, and Heidelberger-Welch, all named after their authors. The Geweke (1992) test compares some proportion of the early era of the chain after the burn-in period with some nonoverlapping proportion of the late era of the chain. A formal difference of means test, based on an asymptotic standard normal statistic, is performed with the idea that in stationarity there should not be an appreciable difference in the means of the two periods. The default uses the 0.0 to 0.1 and 0.5 to 1.0 periods. Gelman and Rubin's (1992) convergence diagnostic compares a small set of independently run chains with different starting points that are overdispersed relative to the target distribution. This is based on normal or Student's-t theory approximations to the marginal posteriors using an ANOVA-based test. The core part of the Heidelberger and Welch (1983) diagnostic uses a Brownian bridge assumption to produce a comparison of a sum from an early part of the chain to an appropriately scaled sum from the full length of the chain, both after the burn-in period. There is a second part of the test, the half-width comparison, that is less important and is often ignored. The Raftery and Lewis (1992) diagnostic gives a rough indication of convergence for a currently running chain. It takes thresholds from the user and estimates the chain length that provides a satisfactory result based on setting up a parallel (but not Markov) chain during a pilot run, where the iterations are a binary series according to whether the generated value in the primary chain at that stage is less than a chosen quantile. These four diagnostics are described in detail in Gill (2007, Chapter 11) along with other approaches.

Note that only the Gelman and Rubin diagnostic requires the user to run multiple chains for the convergence analysis. We think that there is more to be learned by running the other diagnostics on multiple chains. Furthermore, when there was controversy about one long run versus multiple short runs, computer resources were more limited (see the discussion in Kass, et al. [1998]). Running multiple chains allows the user the ability to start from different points in the multidimensional sample space, change the random seed, and alter parameters of the diagnostic. However, running multiple chains, saving/processing them, and moving up and down the menu systems of `boa` and `coda` can add to the overhead of running MCMC, thus providing a disincentive. This extra work may explain the insufficient attention to convergence issues that we have observed in recent papers. Our goal here is to provide a function that reduces this administrative burden thus motivating researchers towards a more robust practice.

Introducing `superdiag`

The `R` function `superdiag` is mostly a wrapper that calls each of the four popular formal diagnostics from the underlying `coda` routines without the `coda` menu structure. The output very closely resembles standard output from these functions and can be automatically dumped to a text file, rather than the screen, with the `sink()` function. In several instances described below we manipulate the diagnostics to exploit the provision of multiple chains. The code for `superdiag`is in the Appendix here and can also be downloaded at `http://jgill.wustl.edu/computing.html`.

Geweke Diagnostic

Our extension of the Geweke diagnostic automatically alters the window specification. There are two parameters: the first gives the proportion of the chain to use advancing upward from the starting value, and the second gives the proportion of the chain to use descending from the final value downward. The default in `boa` and `coda` uses Geweke's (1992) defaults of 0.1 and 0.5: the first 10% of the values and the last 50% of the values. If a user of `superdiag` gives only one chain to analyze, then we use these defaults. However, if multiple chains are provided, only the first uses the defaults and all other chain analyses get random nonoverlapping proportions up from the start of the chain and down from the end of the chain.

Heidelberger and Welch Diagnostic

For the Heidelberger and Welch diagnostic we manipulate two parameters. The halfwidth part of the test calculates a $(1-\alpha)\%$ credible interval around the sample mean for each parameter dimension, where the estimated asymptotic standard error is the square root of spectral density divided by the non-discarded sample size, $s(0)/n^*$. The default in `boa` and `coda` is $\alpha = 0.05$. If the mean divided by this halfwidth is lower than ϵ (defaulted to 0.1), then the halfwidth test is passed for this dimension. We sample with replacement common α values for each chain to alter the size of the credible interval that creates the halfwidth value. This adds robustness to the effects of (necessarily) arbitrary choices of the α parameter. Secondly, we modify the default value for ϵ by separately sampling uniformly in the interval $[0.01 : 0.2]$ to provide a richer range of test criteria for the ratio. If a single chain is analyzed then the defaults are used, but with multiple chains the user will see differing combinations of these parameters.

Raftery and Lewis Diagnostic

For each parameter (separately), the Raftery and Lewis is run as a pre-processing algorithm to determine a potential chain length to use after burn-in. First select the posterior tail threshold of interest, defaulted to $q = 0.025$, then determine a "tolerance" for this quantile, defaulted to $r = 0.0005$, followed by the desired probability of being within that tolerance, defaulted to $s = 0.95$. So generically we get a 95% probability of being in the interval $[0.0245 : 0.0255]$, which is the default here when only one chain is supplied to `superdiag`. Also, we need a convergence tolerance value ϵ, which is used to determine a stopping point based on a parallel chain process (defaulted to 0.001). The diagnostic then runs a pilot sampler whose length is determined as if there is no autocorrelation in the parallel chain, given by rounding: $n_{\text{pilot}} = \left[\Phi^{-1}\left(\frac{s+1}{2}\right)\frac{\sqrt{q(1-q)}}{r}\right]^2$, where $\Phi^{-1}()$ is the inverse of the normal CDF. The process will then return: the length of the burn-in period (M), the estimated number of post burn-in iterations required to meet these goals (N), and a specified thinning interval (k). For each of these four parameters we sample from a vector (changeable by users) of values around the defaults (larger and smaller) to provide a reasonable range of alternatives.

Example Model

This example comes from Gill (2007, Chapter 11), which is an extension of Norrander (2000), using tobit models to account for 15 states that do not have a death penalty on the books and therefore cannot express public support through a count of executions. The usual questions center on ideological, racial and religious makeup, political culture, and urbanization as causal effects of state-level executions (1993-1994 in the data here). If $\mathbf{z}$ is a latent outcome variable in this context with $\mathbf{z} = \mathbf{x}\boldsymbol{\beta} + \boldsymbol{\epsilon}$ and $z_i \sim \mathcal{N}(\mathbf{x}\boldsymbol{\beta}, \sigma^2)$, then the observed outcome variable is produced according to: $y_i = z_i$ if $z_i > 0$, and $y_i = 0$, if $z_i \leq 0$. This gives the likelihood function:

$$L(\boldsymbol{\beta}, \sigma^2|\mathbf{y}, \mathbf{X}) = \prod_{y_i=0}\left[1 - \Phi\left(\frac{x_i\boldsymbol{\beta}}{\sigma}\right)\right] \times \prod_{y_i>0}(\sigma^{-1})\exp\left[-\frac{1}{2\sigma^2}(y_i - x_i\boldsymbol{\beta})^2\right]. \tag{1}$$

A flexible parameterization for the priors is given by Gawande (1998):

$$\boldsymbol{\beta}|\sigma^2 \sim \mathcal{N}(\boldsymbol{\beta}_0, \mathbf{I}\sigma^2 B_0^{-1}), \quad \sigma^2 \sim \mathcal{IG}\left(\frac{\gamma_0}{2}, \frac{\gamma_1}{2}\right) \tag{2}$$

with vector hyperparameter $\boldsymbol{\beta}_0$, scalar hyperparameters B_0, $\gamma_0 > 2$, $\gamma_1 > 0$, and an identity matrix $\mathbf{I}$. The resulting full conditional distributions for Gibbs sampling are given for the $\boldsymbol{\beta}$ block, σ^2, and the $z_i|y_i = 0$ as:

$$\begin{aligned}\boldsymbol{\beta}|\sigma^2, \mathbf{z}, \mathbf{y}, \mathbf{X} &\sim \mathcal{N}\Big((B_0 + \mathbf{X}'\mathbf{X})^{-1})(\boldsymbol{\beta}_0 B_0 + \mathbf{X}'\mathbf{z}), \\ &\qquad (\sigma^{-2}B_0 + \sigma^{-2}\mathbf{X}'\mathbf{X})^{-1})\Big) \\ \sigma^2|\boldsymbol{\beta}, \mathbf{z}, \mathbf{y}, \mathbf{X} &\sim \mathcal{IG}\left(\frac{\gamma_0 + n}{2}, \right. \\ &\qquad \left. \frac{\gamma_1 + (\mathbf{z} - \mathbf{X}\boldsymbol{\beta})'(\mathbf{z} - \mathbf{X}\boldsymbol{\beta})}{2}\right) \\ z_i|y_i = 0, \boldsymbol{\beta}, \sigma, \mathbf{X} &\sim \mathcal{TN}(\mathbf{X}\boldsymbol{\beta}, \sigma^2)I_{(-\infty,0)},\end{aligned} \tag{3}$$

where $\mathcal{TN}()$ is the truncated normal and the indicator function $I_{(-\infty,0)}$ gives the truncation bounds. The prior parameters for the inverse gamma distribution are designed to give a diffuse form: $\gamma_0 = 300$, $\gamma_1 = 100$, $B_0 = 0.02$, and $\boldsymbol{\beta} = \mathbf{0}$.

We programmed a Gibbs sampler in `R` to provide marginal distributions from the full conditional distributions, running it 50,000 times and disposing of the first 40,000 iterations. The results are summarized in the table below and the `R` code for the sampler is given at `http://jgill.wustl.edu/computing.html`.

	Mean	SE	95% HPD
Constant	-14.545	3.672	[-21.766:-7.350]
Past Rates	171.146	8.048	[155.200:186.608]
Political Culture	0.346	0.145	[0.060:0.622]
Current Opinion	3.974	1.067	[1.858:6.022]
Ideology	3.142	1.111	[0.973:5.315]
Murder Rate	0.009	0.080	[-0.157:0.159]

Running `superdiag`

As noted, `superdiag` is designed to reduce the time it takes to manipulate and test multiple chains with the standard diagnostics. The default parameter values for the four tests remain defaults for the first chain and then we systematically manipulate them for robustness. Sometimes, particularly for Raftery and Lewis, this causes an unusual combination that gives a very pessimistic view of convergence. Users are cautioned to pay attention to such combinations and possibly discount the result.

The input object definition provided to `superdiag` is flexible. The four formal functions of diagnostics in `coda` accept only an `mcmc` or `mcmc.list` object as input. Conversely, the input supplied to superdiag can be a list object, or either mcmc and mcmc.list objects, regardless of the number of chains. Therefore, researchers who generate MCMC samples on their own rather than by programs like WinBUGS or JAGS, can easily analyze data by using superdiag without necessary conversion. Since our Gibbs sampler output was created with an R function and for each chain the last 10,000 values out of 50,000 were saved to text files, we loaded the MCMC output and built objects according to:

```
tobit.list <- list()
for (i in 1:5) {
    tobit.list[[i]] <- mcmc(read.table(paste(
        "tobit.mcmc.out",i,sep=""),header=TRUE))
}
tobit.list <- as.mcmc.list(tobit.list)
```

However, chains produced by WinBUGS or JAGS can be read in directly as mcmc objects with the function read.coda. Users of R2jags, Rjags, runjags, BRugs, rbugs, R2WinBUGS, and MCMCpack are already working with appropriately formatted data objects for superdiag.

Running the diagnostic with our newly created list is done with one simple command, producing:

```
superdiag(tobit.list,burnin=0)

Number of chains = 5
Number of iterations = 10000 per chain before discarding
                       the burn-in period
The burn-in period = 0 per chain
Sample size in total = 50000

********** The Geweke diagnostic: **********
Z-scores:
                   chain 1  chain 2  chain 3
Constant           0.19006 -1.19822 -1.97097
Past.Rates         0.43421  0.35479 -0.10703
Political.Culture  0.09149  0.23044  1.18154
Current.Opinion    1.04999 -0.26106  2.11368
Ideology          -0.70922  1.08343  1.09934
Murder.Rate        1.12295  0.41441 -0.30343
Window From Start  0.10000  0.03854  0.62622
Window From Stop   0.50000  0.92317  0.28442

                   chain 4  chain 5
Constant          -0.77899 -0.14759
Past.Rates         0.81997  0.25356
Political.Culture -1.67556 -0.27691
Current.Opinion   -2.47919 -0.15100
Ideology           1.51378  0.38307
Murder.Rate       -0.53252 -0.67923
Window From Start  0.25000  0.40163
Window From Stop   0.28153  0.12059

********** The Gelman-Rubin diagnostic: **********
Potential scale reduction factors:

                  Point est. Upper C.I.
Constant                   1          1
Past.Rates                 1          1
Political.Culture          1          1
Current.Opinion            1          1
Ideology                   1          1
Murder.Rate                1          1

Multivariate psrf

1

********** The Heidelberger-Welch diagnostic: **********

Chain 1, epsilon=0.1, alpha=0.05
                  Stationarity start     p-value
                  test         iteration
Constant          passed       1         0.275
Past.Rates        passed       1         0.303
Political.Culture passed       1         0.968
Current.Opinion   passed       1         0.267
Ideology          passed       1         0.432
Murder.Rate       passed       1         0.925

                  Halfwidth Mean     Halfwidth
                  test
Constant          passed    -14.5451 0.09711
Past.Rates        passed    171.1460 0.16613
Political.Culture passed      0.3461 0.00381
Current.Opinion   passed      3.9738 0.02737
Ideology          passed      3.1423 0.02991
Murder.Rate       failed      0.0088 0.00198

Chain 2, epsilon=0.032, alpha=0.005
                  Stationarity start     p-value
                  test         iteration
Constant          passed       1         0.871
Past.Rates        passed       1         0.660
Political.Culture passed       1         0.882
Current.Opinion   passed       1         0.616
Ideology          passed       1         0.770
Murder.Rate       passed       1         0.932

                  Halfwidth Mean      Halfwidth
                  test
Constant          passed    -14.57477 0.08874
Past.Rates        passed    171.28755 0.16882
Political.Culture passed      0.34533 0.00342
Current.Opinion   passed      3.97391 0.03253
Ideology          passed      3.14973 0.02591
Murder.Rate       failed      0.00947 0.00206

Chain 3, epsilon=0.133, alpha=0.01
                  Stationarity start     p-value
                  test         iteration
Constant          passed       1         0.1009
Past.Rates        passed       1         0.2762
Political.Culture passed       1         0.6218
Current.Opinion   passed       1         0.0683
Ideology          passed       1         0.4488
Murder.Rate       passed       1         0.8134

                  Halfwidth Mean      Halfwidth
                  test
Constant          passed    -14.58034 0.07927
Past.Rates        passed    171.09983 0.18120
Political.Culture passed      0.34438 0.00319
Current.Opinion   passed      3.96566 0.03392
Ideology          passed      3.15523 0.02413
Murder.Rate       failed      0.00937 0.00181

Chain 4, epsilon=0.152, alpha=0.025
                  Stationarity start     p-value
                  test         iteration
```

```
Constant          passed       1         0.7090
Past.Rates        passed       1         0.4077
Political.Culture passed       1         0.1044
Current.Opinion   passed       1         0.0286
Ideology          passed       1         0.2394
Murder.Rate       passed       1         0.6763

                  Halfwidth Mean     Halfwidth
                  test
Constant          passed    -14.5491 0.07792
Past.Rates        passed    171.1474 0.15642
Political.Culture passed      0.3435 0.00371
Current.Opinion   passed      3.9472 0.03155
Ideology          passed      3.1478 0.02464
Murder.Rate       failed      0.0105 0.00221

Chain 5, epsilon=0.082, alpha=0.01
                  Stationarity start     p-value
                  test         iteration
Constant          passed       1         0.181
Past.Rates        passed       1         0.592
Political.Culture passed       1         0.335
Current.Opinion   passed       1         0.409
Ideology          passed       1         0.349
Murder.Rate       passed       1         0.197

                  Halfwidth Mean     Halfwidth
                  test
Constant          passed    -14.5786 0.08361
Past.Rates        passed    171.1698 0.15530
Political.Culture passed      0.3430 0.00352
Current.Opinion   passed      3.9430 0.02890
Ideology          passed      3.1574 0.02409
Murder.Rate       failed      0.0105 0.00208

********** The Raftery-Lewis diagnostic: **********

Chain 1, converge.eps = 0.001
Quantile (q) = 0.025
Accuracy (r) = +/- 0.005
Probability (s) = 0.95

                   Burn-in  Total Lower bound  Dependence
                   (M)      (N)   (Nmin)       factor (I)
 Constant          3        4061  3746         1.08
 Past.Rates        2        3802  3746         1.01
 Political.Culture 3        4028  3746         1.08
 Current.Opinion   3        4061  3746         1.08
 Ideology          3        4061  3746         1.08
 Murder.Rate       3        4129  3746         1.10


Chain 2, converge.eps = 0.001
Quantile (q) = 0.001
Accuracy (r) = +/- 0.0025
Probability (s) = 0.99

                   Burn-in  Total Lower bound  Dependence
                   (M)      (N)   (Nmin)       factor (I)
 Constant          2        1061  1061         1.00
 Past.Rates        2        1061  1061         1.00
 Political.Culture 2        1061  1061         1.00
 Current.Opinion   2        1061  1061         1.00
 Ideology          3        1297  1061         1.22
 Murder.Rate       3        1297  1061         1.22


Chain 3, converge.eps = 2e-04
Quantile (q) = 0.001
Accuracy (r) = +/- 0.005
Probability (s) = 0.999

                   Burn-in  Total Lower bound  Dependence
                   (M)      (N)   (Nmin)       factor (I)
 Constant          2        434   433          1
 Past.Rates        2        434   433          1
 Political.Culture 2        434   433          1
 Current.Opinion   2        434   433          1
 Ideology          2        434   433          1
 Murder.Rate       2        434   433          1


Chain 4, converge.eps = 0.005
Quantile (q) = 0.1
Accuracy (r) = +/- 0.0025
Probability (s) = 0.99

You need a sample size of at least 95543 with these
values of q, r and s

Chain 5, converge.eps = 0.005
Quantile (q) = 0.001
Accuracy (r) = +/- 0.005
Probability (s) = 0.975

                   Burn-in  Total Lower bound  Dependence
                   (M)      (N)   (Nmin)       factor (I)
 Constant          1        202   201          1
 Past.Rates        1        202   201          1
 Political.Culture 1        202   201          1
 Current.Opinion   1        202   201          1
 Ideology          1        202   201          1
 Murder.Rate       1        202   201          1
```

Running the function also generated the following warning:

```
Warning message:
In superdiag(tobit.list, burnin = 0) :
  The burn-in period is negative or zero
```

which is not material to us since we have already removed the proposed burn-in values. Even though this chain appears to be in convergence, note that there are differences resulting from parameter changes. For example, Geweke test for `Current Opinion` across the five chains returns:

```
1.04999 -1.283986  0.81387 -2.98846  0.5089598
```

gives one value typically outside of the normal range for expected convergence. It is critical, however, to keep in mind that testing at $\alpha = 0.05$ means that even in actual convergence, about 1 out of 20 values will be in the tails. Therefore it is important to look at the complete picture.

Concluding Remarks

It is important to remember that these convergence diagnostics are actually indicators of *nonconvergence* rather than evidence of convergence. Failing to find evidence of nonconvergence with these procedures is comforting but not decisive. Careful practitioners should treat positive results from one test with continued skepticism, and run multiple diagnostics on any single Markov chain, any one of which can provide sufficient evidence of failure. Our goal here is simply to encourage such caution by making the process easier, therefore improving standard practice in the literature.

References

Brooks, Stephen. P. and Gareth O. Roberts. 1998. “Convergence Assessment Techniques for Markov Chain Monte Carlo.” *Statistics and Computing* 8: 319–335.

Gawande, Kishore. 1998. “Comparing Theories of Endogenous Protection: Bayesian Comparison of Tobit Models Using Gibbs Sampling Output.” *Review of Economics and Statistics* 80(1): 128–140.

Gelman, Andrew and Donald B. Rubin. 1992. “Inference from Iterative Simulation Using Multiple Sequences.” *Statistical Science* 7(4): 457–511.

Geweke, J. 1992. “Evaluating the Accuracy of Sampling-Based Approaches to the Calculation of Posterior Moments.” In *Bayesian Statistics 4*, ed. J. M. Bernardo, A. F. M. Smith, A. P. Dawid, and J. O. Berger. Oxford, UK: Oxford University Press, 169–193.

Gill, Jeff. 2007. *Bayesian Methods for the Social and Behavioral Sciences.* Second Edition. New York, NY: Chapman & Hall.

Gill, Jeff. 2008. “Is Partial-Dimension Convergence a Problem for Inferences from MCMC Algorithms?” *Political Analysis* 16(2): 153–178.

Heidelberger, Philip and Peter D. Welch. 1983. “Simulation Run Length Control in the Presence of an Initial Transient.” *Operations Research* 31(6): 1109–1144.

Kass, Robert E., Bradley P. Carlin, Andrew Gelman, and Radford M. Neal. 1998.“Markov Chain Monte Carlo in Practice: A Roundtable Discussion.” *The American Statistician* 52(2): 93–100.

Norrander, Barbara. 2000. “The Multi-Layered Impact of Public Opinion on Capital Punishment Implementation in the American States.” *Political Research Quarterly* 53(4): 771–793.

Raftery, A. E. and Lewis, S. M. 1992. “How Many Iterations in the Gibbs Sampler?” In *Bayesian Statistics 4*, ed. J. M. Bernardo, A. F. M. Smith, A. P. Dawid, and J. O. Berger. Oxford, UK: Oxford University Press, 763–773.

Appendix

This appendix gives our `R` code for the `superdiag` function.

```
## CREATE THE DIAGNOSTIC FUNCTION, TSUNG-HAN TSAI AND JEFF GILL
superdiag <- function(mcmcoutput, burnin=10000, confidence.gr=0.95,
             frac1.gw=0.1, frac2.gw=0.5, eps.hw=0.1,
             pvalue.hw=0.05, q.rl=0.025, r.rl=0.005, s.rl=0.95,
             eps.rl=0.001) {

# mcmcoutput: input chains from jags, bugs, etc.
# confidence.gr: 1-alpha for testing with the Gelman and Rubin test
# frac1.gw: frac1 for the Geweke Test
# frac2.gw: frac2 for the Geweke Test
# eps.hw: epsilon for the Heidelberger and Welch test
# pvalue.hw: p-value for the Heidelberger and Welch test
# q.rl: q-parameter for the Raftery and Lewis Test
# r.rl: r-parameter for the Raftery and Lewis Test
# s.rl: s-parameter for the Raftery and Lewis Test
# eps.rl: convergence epsilon for the Raftery and Lewis Test

# CREATE A FUNCTION TO DISCARD THE BURN-IN PERIOD
burn <- function(input.matrix, burnin) {
out <- input.matrix[-(1:burnin),];
return(out);
}

# THE INPUT SHOULD BE AN "mcmc", "mcmc.list", OR "list" OBJECT
if (class(mcmcoutput) != "mcmc" & class(mcmcoutput) !=
             "mcmc.list" & class(mcmcoutput) != "list")
stop("The inputs have to be mcmc, mcmc.list, or list objects.");
# CONVERT "mcmc" INTO "mcmc.list"
if (class(mcmcoutput) == "mcmc") {
mcmcoutput <- as.mcmc.list(mcmcoutput);
}

para.names <- dimnames(mcmcoutput[[1]])[[2]];  # PARAMETERS NAMES
n.chains <- length(mcmcoutput);  # THE NUMBER OF CHAINS
dim.chain <- sapply(mcmcoutput, dim);  # THE DIMENSION OF EACH CHAIN
        # THE NUMBER OF ITERATIONS BEFORE DELETING THE BURN-IN PERIOD
t.iter <- dim.chain[1];
diff.dim <- dim.chain - dim.chain;
if (sum(diff.dim != 0) != 0) stop("The number of iterations or
              variables is not equal for all chains.");

# DISCARD THE BURN-IN PERIOD
if (burnin <= 0) {
warning("The burn-in period is negative or zero");
mcmcburnin <- mcmcoutput;
}
else {
mcmcburnin <- lapply(mcmcoutput, burn, burnin=burnin);
}

# SAVE THE SAMPLES AS A MCMC LIST AFTER DISCARDING THE BURN-IN PERIOD
mcmcburnin.list <- vector("list", n.chains);
for (i in 1:n.chains) {
mcmcburnin.list[[i]] <- as.mcmc(mcmcburnin[[i]]);
}
mcmcburnin.mcmclist <- as.mcmc.list(mcmcburnin.list);

# THE TOTAL SAMPLES FOR ALL CHAINS
t.samples <- as.matrix(mcmcburnin.mcmclist);

# REARRANGE THE PRINTED RESULTS
geweke.chains <- matrix(NA, nrow=n.chains, ncol=dim.chain[2]);
heidel.list <- vector("list", n.chains);
raftery.list <- vector("list", n.chains);

# SETUP DIFFERENT WINDOW SPECIFICATIONS FOR GEWEKE
geweke.windows <- matrix(c(frac1.gw,frac2.gw),ncol=2)
for (i in 2:n.chains) {
win1 <- runif(1,0,0.99); win2 <- 1-runif(1,win1,1)
geweke.windows <- rbind(geweke.windows,c(win1,win2))
}

# SETUP DIFFERENT PARAMETER SPECIFICATIONS FOR HEIDELBERGER AND WELCH
heidel.params <- matrix(c(eps.hw,pvalue.hw),ncol=2)
pvals <- c(0.1,0.05,0.025,0.01,0.005)
for (i in 2:n.chains) {
param1 <- runif(1,0.01,0.2); param2 <- sample(x=pvals,size=1)
heidel.params <- rbind(heidel.params,c(param1,param2))
}

# SETUP DIFFERENT PARAMETER SPECIFICATIONS FOR RAFTERY AND LEWIS
raft.params <- matrix(c(q.rl, r.rl, s.rl, eps.rl),ncol=4)
qvals <-c(0.25,0.1,0.05,0.01,0.001)
rvals <- c(0.001,0.0025,0.0005,0.001,0.005)
svals <- c(0.9,0.95,0.975,0.99,0.999)
evals <- c(0.005,0.0025,0.001,0.0005,0.0002)
for (i in 2:n.chains) {
param1 <- sample(x=qvals,size=1); param2 <- sample(x=rvals,size=1)
param3 <- sample(x=svals,size=1); param4 <- sample(x=evals,size=1)
raft.params <- rbind(raft.params,c(param1,param2,param3,param4))
}
```

```
# RUN DIAGNOSTICS BY CHAIN IN ORDER TO AVOID ONE CHAIN RUINS ALL CHAINS
for (i in 1:n.chains) {
geweke <- suppressWarnings(try(geweke.diag(mcmcburnin.mcmclist[[i]],
                      geweke.windows[i,1], geweke.windows[i,2]),
                      silent=TRUE));
if (class(geweke) == "geweke.diag")
                      geweke.chains[i,] <- t(geweke[1]$z);
heidel.list[[i]] <- suppressWarnings(try(heidel.diag(
                      mcmcburnin.mcmclist[[i]], heidel.params[i,1],
                      heidel.params[i,2]), silent=TRUE));
raftery.list[[i]] <- suppressWarnings(try(raftery.diag(
                      mcmcburnin.mcmclist[[i]], q=raft.params[i,1],
                      r=raft.params[i,2], s=raft.params[i,3],
                      converge.eps=raft.params[i,4]), silent=TRUE));
}
colnames(geweke.chains) <- para.names;

# PROVIDE THE BASIC INFORMATION OF MCMC SAMPLES
cat(paste("Number of chains =", n.chains, "\n"));
cat(paste("Number of iterations =", t.iter, "per chain before
                   discarding the burn-in period\n"));
cat(paste("The burn-in period =", burnin, "per chain\n"))
cat(paste("Sample size in total =", dim(t.samples)[1], "\n"))
cat("\n");

# REPORT RESULTS OF DIAGNOSTICS OF CONVERGENCE
if (n.chains < 2) {
chain.name <- "chain 1";
rownames(geweke.chains) <- chain.name;
cat("The Gelman-Rubin diagnostic is not reported since
                     the number of chains is less than 2.\n");
cat("\n");
cat("The Geweke diagnostic:\n");
cat(paste("Fraction in 1st window =", frac1.gw, "\n"));
cat(paste("Fraction in 2nd window =", frac2.gw, "\n"));
cat(paste("Z-scores:\n"));
print(t(geweke.chains));
cat("\n");
cat("The Heidelberger-Welch diagnostic:\n");
cat("\n");
print(heidel.list);
cat("\n");
cat("The Raftery-Lewis diagnostic:\n");
cat("\n");
print(raftery.list);
cat("\n");
}
else {
chain.name <- "chain1";
for (i in 2:n.chains) {
namei <- paste("chain ", i, sep="");
chain.name <- c(chain.name, namei);
}
rownames(geweke.chains) <- chain.name;

cat("********** The Geweke diagnostic: **********\n");
dimnames(geweke.windows)[[2]] <- c("Window From Start","Window From Stop")
cat(paste("Z-scores:\n"));
print(rbind( t(geweke.chains), t(round(geweke.windows,5))));
cat("\n");
cat("********** The Gelman-Rubin diagnostic: **********\n");
print(gelman.diag(mcmcburnin.mcmclist, confidence.gr));
cat("\n");
cat("********** The Heidelberger-Welch diagnostic: **********\n");
cat("\n");
for (i in 1:n.chains)  {
cat(paste("Chain ",i,", epsilon=",round(heidel.params
                                  [i,1],3),",
                                  alpha=",round(heidel.params
                                  [i,2],3),sep=""))
print(heidel.list[[i]]);
cat("\n");
}
cat("********** The Raftery-Lewis diagnostic: **********\n");
cat("\n");
for (i in 1:n.chains)  {
cat(paste("Chain ",i,", converge.eps = ",round(raft.params[i,4],4),sep=""))
print(raftery.list[[i]]);
cat("\n");
}
cat("\n");
}
# RETURN THE MCMC SAMPLES WITH THE BURN-IN DISCARDED
    superdiag.chains <- list(mcmc.samples = mcmcburnin.mcmclist)
```

PresidentParser: Automated Mark-Up Utility

McKendon Lafleur and John Beieler
Louisiana State University
McKendon@acm.org and jbeiel1@lsu.edu.

PresidentParser

As any researcher dealing with text-based data knows, the task of manipulating the text into a useable form is often the most time consuming aspect of the research process. This holds true especially for the study of leader psychology in international relations. This task is what the `PresidentParser` program aims to alleviate.

In short, `PresidentParser` parses speech acts and prepares them for processing in the Profiler Plus environment[1]. Since research into the role of elite psychology in international relations is focused on at-a-distance analysis of speech acts to derive quantitative measures of that leader's psychology, a researcher must often modify those speech acts so that only the verbal material, and thus psychology, of the leader is captured. For instance, many studies of leader psychology rely on leader responses to question and answer sessions. One does not, however, wish to include the verbal material of a reporter in the psychological profile of a leader. Luckily, Profiler Plus allows for an easy way to remove this unwanted verbal material while still retaining the integrity and logic of a speech act. Profiler Plus includes the ability to insert the XML tags `<ignore></ignore>` into a speech act, which, as the name implies, tells the program to ignore the lines surrounded by the tags. Traditionally, this task has been performed by hand, and anyone who has been involved in a research project of this type can attest that it is often a long, monotonous task. Additionally, with the advent of automated webscraping scripts, the number of speech acts to be processed has begun to grow exponentially, often into the tens of thousands.

[1] Profiler Plus can be obtained from Social Science Automation