Basic MCMC Diagnostics

STAT8810, Fall 2017

M.T. Pratola

November 19, 2017

Today

Traceplots
Convergence Diagnostics
Effective Sample Size

Basic MCMC Diagnostics

- MCMC is an algorithm that generates (approximate) samples from the posterior distribution of interest.
- We would like to check, to some degree, if our samples are any good.
- This is a difficult problem. Most methods in the literature are univariate.
- Run multiple chains: do they agree with eachother?
- Run a long chain: is the chain transient or stationary?
- Insightful plots are helpful.
- Strategy: check for (obvious) ways it might have failed, rather than checking (guaranteeing) that it worked.

Traceplots

- First place to start. This is simply a plot of the sample versus its index.
- Plot should show stationary behavior - constant mean/median, constant variance, no trend.
- Often, this is called the "fat marker" test.
- Check the autocorrelation by making an ACF plot. The ACF should decay rapidly.
    - Ideally, we want independent draws from the posterior.

Example

```
source("dace.sim.r")
set.seed(88)
n=10
k=1
rhotrue=0.2
lambdaytrue=1
design=as.matrix(runif(n))
l1=list(m1=outer(design[,1],design[,1],"-"))
l.dez=list(l1=l1)
R=rhogeodacecormat(l.dez,c(rhotrue))$R
L=chol(R)
u=rnorm(nrow(R))
z=t(L)%*%u

# now set up our observed data:
y=z
l1=list(m1=outer(design[,1],design[,1],"-"))
```

Example

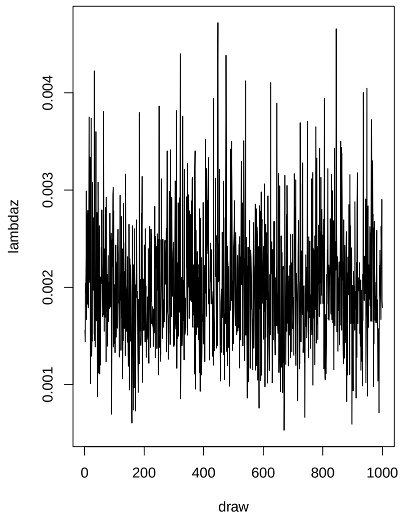
```
source("regression.r")
pi=list(az=5,bz=5,rhoa=rep(1,k),rhob=rep(5,k))

# Run MCMC using a proposal width of 1e-5 for
# the rho parameters.
mh=list(rr=1e-5)

# regress works as follows:
# Adapt for first 50% of iterations -- here that is 2500.
# Further burn-in is draws up to start of last
#       iterations -- here that is 2501--4000.
# last is number of draws to take as posterior samples.
#       Here that is 4001--5000.

fit=regress(y,l.dez,5000,pi,mh,last=1000,adapt=FALSE)
```

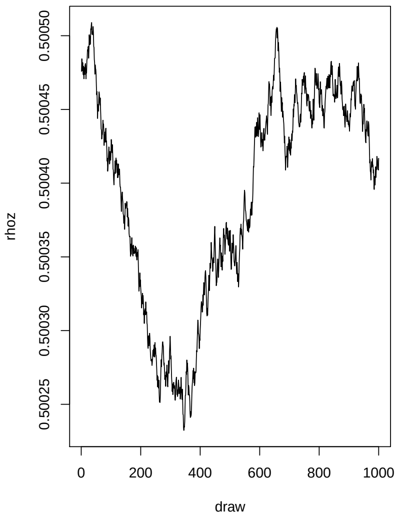
```
##
```

Example

```
par(mfrow=c(1,2))
plot(fit$lambdaz,type='l',xlab="draw",ylab="lambdaz")
abline(h=lambdaytrue)
plot(fit$rhoz,type='l',xlab="draw",ylab="rhoz")
abline(h=rhotrue)
```

Example


lambdaz
0.001
0.002
0.003
0.004
0
200
400
600
800
1000
draw


rhoz
0.50025
0.50030
0.50035
0.50040
0.50045
0.50050
0
200
400
600
800
1000
draw

Example

- 5k draws is usually considered way too small.
- Let's repeat with 20,000 iterations.
- We'll take last=5,000 iterations.
- So adapt would occur for the first 10,000 iterations and further burn-in up to the 15,000th iteration.
- Realistically I would do much greater than this, but compiling my slides then takes a long time...

Example

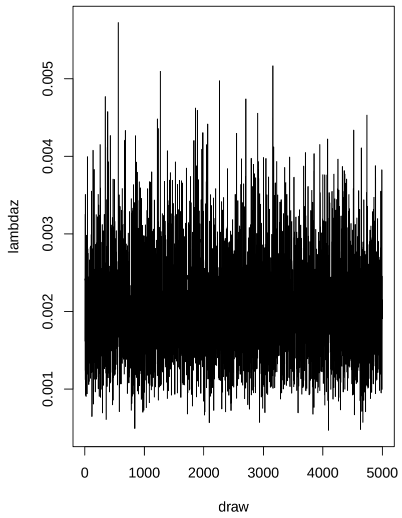
```
pi=list(az=5,bz=5,rhoa=rep(1,k),rhob=rep(5,k))

# Run MCMC using a proposal width of 1e-5 for
# the rho parameters.
mh=list(rr=1e-5)

# Run the MCMC
fit=regress(y,l.dez,20000,pi,mh,last=5000,adapt=FALSE)
```

```
##
##  Bayesian Gaussian Process Interpolation model
##  The last  4999  samples from the posterior will be repo
##  The stepwidth for uniform corr. param proposal distn is
##  Prior params:  az= 5  bz= 5
##  ---------------------------------------------------------
##
##
```

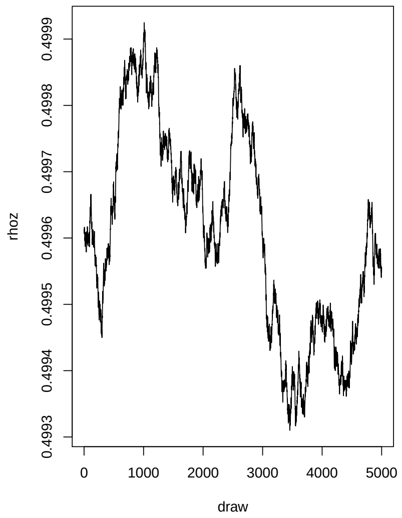
Example

```r
par(mfrow=c(1,2))
plot(fit$lambdaz,type='l',xlab="draw",ylab="lambdaz")
abline(h=lambdaytrue)
plot(fit$rhoz,type='l',xlab="draw",ylab="rhoz")
abline(h=rhotrue)
```

Example



lambdaz
0.001
0.002
0.003
0.004
0.005
0
1000
2000
3000
4000
5000
draw

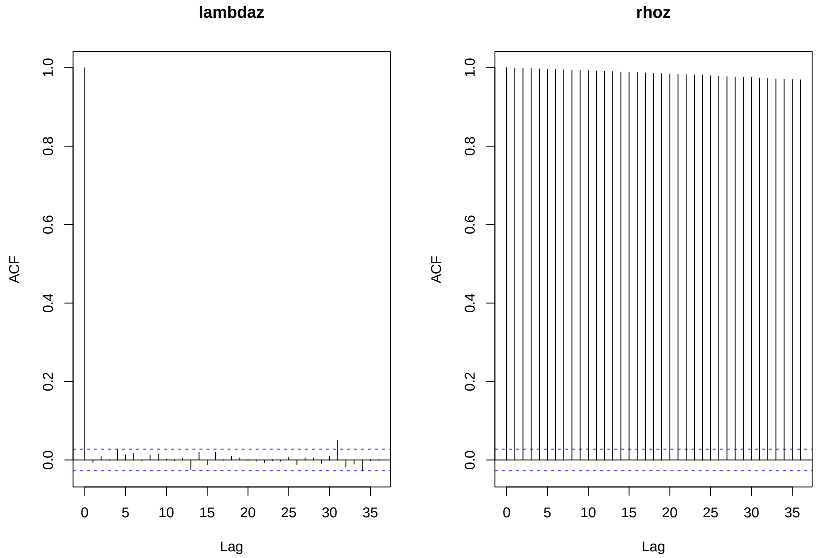


rhoz
0.4993
0.4994
0.4995
0.4996
0.4997
0.4998
0.4999
0
1000
2000
3000
4000
5000
draw

Example

```
par(mfrow=c(1,2))
acf(fit$lambdaz,main="lambdaz")
acf(fit$rhoz,main="rhoz")
```

Example



Single-chain Diagnostics

- How many samples should we draw? How many should we discard as burn-in?
- Starting rule of thumb: take total number of draws to be $N = 50,000$.
- Discard at least half as burn-in.
- More complex the model, the larger number of samples needed and the longer we need to run the algorithm to burn-in.

Raftery & Lewis†

- R&L diagnostic tells us how big to make N based on our needs, and how much burn-in to throw away.
- Used for single chains. Aims to detect non-convergence to the stationary distribution.
- Gives us:
  - N_{min}: the minimum total number of iterations that should be run assuming independent samples.
  - M: the suggested number of iterations to discard as burn-in.
  - $k = N/N_{min}$: the thinning interval. If we keep every kth sample we would have approximately independent draws.

A.E. Raftery and S. Lewis (1992): *How many iterations in the Gibbs sampler?* in Bayesian Statistics 4, eds. J.M. Bernardo, J.O. Berger, A.P. Dawid and A.F.M. Smith, Oxford University Press, pp. 763–773.

Raftery & Lewis

- To use R&L, we supply it with 4 numbers: Q, R, S, A.
  - Q: a quantile that we want to estimate.
  - R: we want to estimate Q with the precision R.
  - S: the $S\%$ probability interval associated with the precision R.
  - A: a convergence tolerance that is used in determining how much burn-in to discard.

Raftery & Lewis

- The approach is based on a 2-state Markov Chain and sample size formulas for a binomial variance.
- Basically, the Markov Chain $\theta(i), i \geq 1$ is turned into a binary sequence of indicators $B(i), i \geq 1$ that correspond to the event $\theta(i) < Q$, the chosen quantile.
- The algorithm looks for the smallest thinning interval k that makes the behavior of this sequence of indicators behave as if it came from an independent 2-state Markov Chain.

Raftery & Lewis

- Burn-in is found as the minimum number of iterations of $B(i)$ it takes for the chain $B(i)$ to approach within A of its estimated stationary distribution.
- N_{min}, the number of samples we need to estimate our quantile Q with the descried precision R at the level S is found using binomial theory on the chain $B(i)$.

Raftery & Lewis

- The defaults are:
  - the quantile $Q = 0.025$,
  - an accuracy (i.e. width of interval for the estimate of Q) of $R = 0.005$,
  - and a probability of obtaining this accuracy level of $S = 95\%$ (i.e. the interval $Q \pm R$ needs to correspond to a 95% interval for Q).
- N_{min} is the minimum number of samples you will need to achieve this.

The Raftery& Lewis diagnostic, along with others we will consider here, are available in the `R` package `CODA` available on `CRAN`. For R& L, see the function `raftery.diag()`.

Example

```
# Run the MCMC -- here I will return everything.
fit=regress(y,l.dez,20000,pi,mh,last=20000,adapt=FALSE)
```

```
##
##  Bayesian Gaussian Process Interpolation model
##  The last  19999  samples from the posterior will be rep
##  The stepwidth for uniform corr. param proposal distn is
##  Prior params:  az= 5  bz= 5
##  -----------------------------------------------------
##
##
## 0.01  percent complete
0.015  percent complete
0.02  percent complete
0.025  percent complete
0.03  percent complete
0.035  percent complete
```

Example

```r
raftery.diag(postmcmc,q=0.025)
```

```
##
## Quantile (q) = 0.025
## Accuracy (r) = +/- 0.005
## Probability (s) = 0.95
##
##  Burn-in  Total  Lower bound  Dependence
##  (M)      (N)    (Nmin)       factor (I)
##  2        3802   3746          1.01
##  312      328604 3746         87.70
```

Example

```
raftery.diag(postmcmc,q=0.975)
```

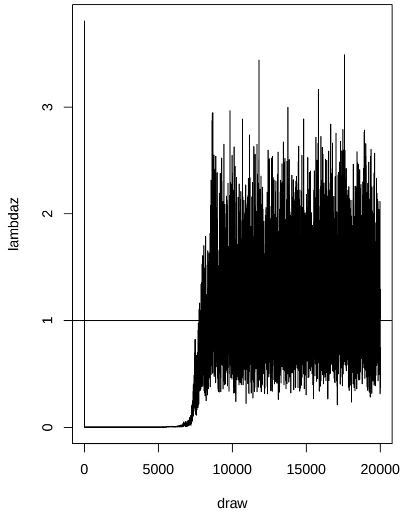
```
##
## Quantile (q) = 0.975
## Accuracy (r) = +/- 0.005
## Probability (s) = 0.95
##
##  Burn-in  Total   Lower bound  Dependence
##  (M)      (N)     (Nmin)       factor (I)
##  2        3764    3746           1
##  1095     1176480 3746         314
```

Example

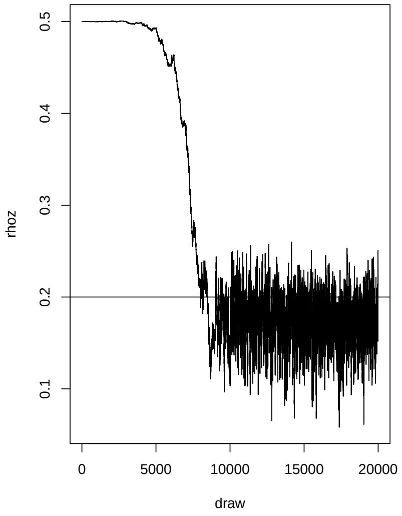
```r
# Run the MCMC, but now turn on adaptation.  Again,
# we will return everything.
fit2=regress(y,l.dez,20000,pi,mh,last=20000,adapt=TRUE)
```

```
##
##  Bayesian Gaussian Process Interpolation model
##  The last  19999  samples from the posterior will be rep
##  The stepwidth for uniform corr. param proposal distn is
##  Prior params:  az= 5  bz= 5
##  ----------------------------------------------------------
##
##
## 0.01  percent complete
0.015  percent complete
0.02  percent complete
0.025  percent complete
0.03  percent complete
```

Example

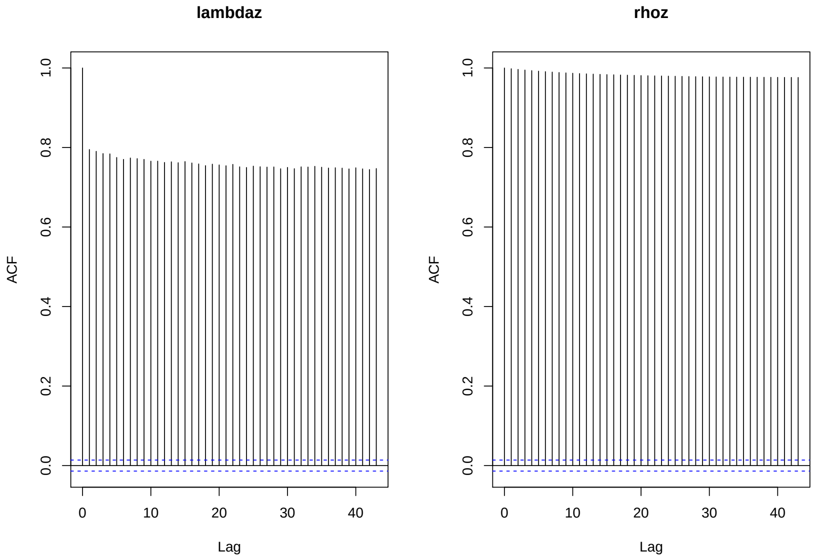


lambdaz
0
1
2
3
0
5000
10000
15000
20000
draw



rhoz
0.1
0.2
0.3
0.4
0.5
0
5000
10000
15000
20000
draw

Example



Example

```r
raftery.diag(postmcmc,q=0.025)
```

```
##
## Quantile (q) = 0.025
## Accuracy (r) = +/- 0.005
## Probability (s) = 0.95
##
##  Burn-in  Total Lower bound  Dependence
##  (M)      (N)   (Nmin)       factor (I)
##  12       16053 3746          4.29
##  36       41700 3746         11.10
```

Example

```r
raftery.diag(postmcmc,q=0.975)
```

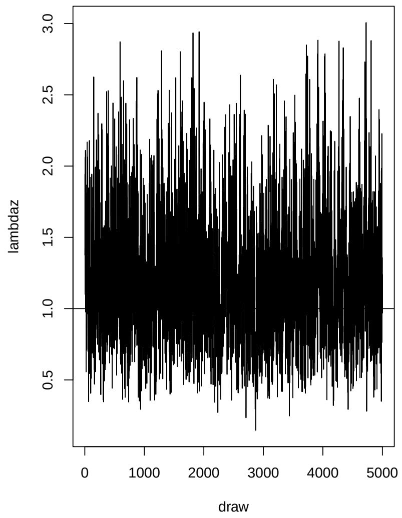
```
##
## Quantile (q) = 0.975
## Accuracy (r) = +/- 0.005
## Probability (s) = 0.95
##
##  Burn-in  Total  Lower bound  Dependence
##  (M)      (N)    (Nmin)       factor (I)
##  9        10977  3746          2.93
##  322      323841 3746         86.40
```

Example

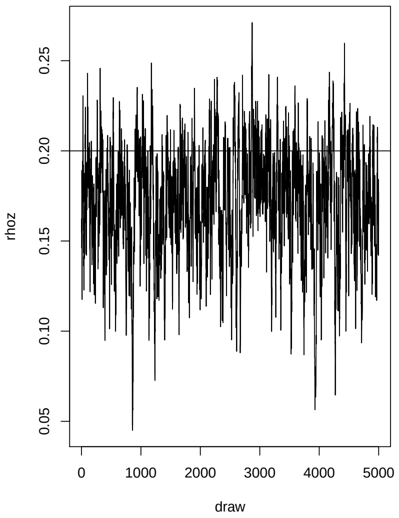
```
# Run the MCMC, but now turn on adaptation.
# Now just return the last 5000 since it looks like
# we burn-in by then.
fit3=regress(y,l.dez,20000,pi,mh,last=5000,adapt=TRUE)
```

```
##
##  Bayesian Gaussian Process Interpolation model
##  The last  4999  samples from the posterior will be repo
##  The stepwidth for uniform corr. param proposal distn is
##  Prior params:  az= 5  bz= 5
##  -----------------------------------------------------------
##
##
## 0.01  percent complete
0.015  percent complete
0.02  percent complete
0.025  percent complete
```

Example

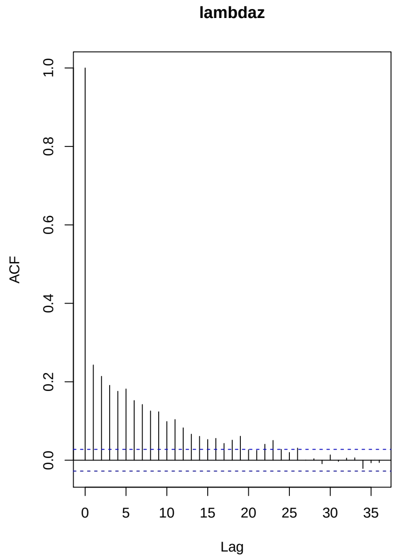


lambdaz
0.5
1.0
1.5
2.0
2.5
3.0
0
1000
2000
3000
4000
5000
draw

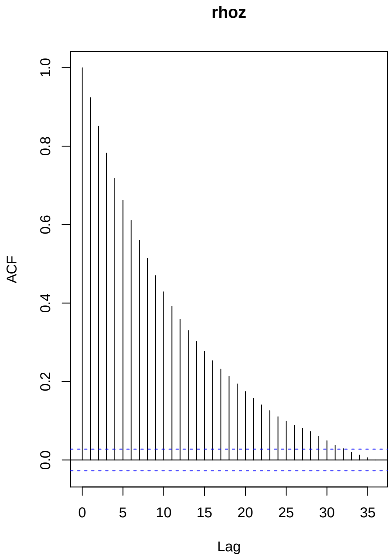


rhoz
0.05
0.10
0.15
0.20
0.25
0
1000
2000
3000
4000
5000
draw

Example



lambdaz
ACF
1.0
0.8
0.6
0.4
0.2
0.0
0
5
10
15
20
25
30
35
Lag



rhoz
ACF
1.0
0.8
0.6
0.4
0.2
0.0
0
5
10
15
20
25
30
35
Lag

Raftery & Lewis

```
library(coda)
post=as.matrix(cbind(fit3$lambdaz,fit3$rhoz))
postmcmc=as.mcmc(post)
```

Example

```r
raftery.diag(postmcmc,q=0.025)
```

```
##
## Quantile (q) = 0.025
## Accuracy (r) = +/- 0.005
## Probability (s) = 0.95
##
##  Burn-in  Total Lower bound  Dependence
##  (M)      (N)   (Nmin)       factor (I)
##  3        4558  3746         1.22
##  30       32808 3746         8.76
```

Example

```r
raftery.diag(postmcmc,q=0.975)
```

```
##
## Quantile (q) = 0.975
## Accuracy (r) = +/- 0.005
## Probability (s) = 0.95
##
##  Burn-in  Total Lower bound  Dependence
##  (M)      (N)   (Nmin)       factor (I)
##  2        3803  3746         1.02
##  24       27446 3746         7.33
```

Geweke Diagnostic†

- Idea is to look at the Markov Chain as a time-series in order to check for stationarity.
- They look at comparing the mean of θ or some function $g(\theta)$ from two disjoint segments of the posterior samples drawn using the Gibbs Sampler and compare their means to asses convergence.
- They divide the chain into the first $p_1\%$ and the last $p_2\%$ where $p_1 + p_2 < 1$.
- Regard the $g(\theta_i)$s as a time-series and assume that the MCMC process and the function $g(\cdot)$ imply the existence of a spectral density $S_g(\omega)$ that has no discontinuities at frequency $\omega = 0$.

† J. Geweke (1992): *Evaluating the Accuracy of Sampling-Based Approaches to the Calculation of Posterior Moments.* in Bayesian Statistics 4, eds. J.M. Bernardo, J. Berger, A.P. Dawid and A.F.M. Smith, Oxford University Press, pp. 169–193.\

M.K. Cowles and B.P. Carlin (1996): *Markov Chain Monte Carlo Convergence Diagnostics: A Comparative Review,* Journal of the American Statistical Association, vol.91, pp.883–904.

Geweke Diagnostic

- Under these assumptions, the estimator of $E[g(\theta)]$ based on n iterations of the Gibbs sampler,

$$\bar{g}_n = \frac{\sum_{i=1}^{n} g(\theta_i)}{n},$$

  has asymptotic variance

$$\frac{S_g(0)}{n}.$$

- The square-root of $S_g(0)/n$ can be used to estimate the standard error of the mean.
- Geweke calls this estimate the *numeric standard error*, or NSE,

Geweke Diagnostic

- Geweke statistic compares the means of $g(\theta)$ using the two separate parts of the chain p_1 and p_2 of size n_1, n_2, say

  $$\bar{g}_1(\theta), \bar{g}_2(\theta)$$

  normalized by our s.e. estimates,

  $$\sqrt{\hat{S}_1(0)/n}, \sqrt{\hat{S}_2(0)/n}$$

  where $\hat{S}_g$ are estimates of the spectral density based on a periodogram estimator.
- If the ratios p_1, p_2 are held fixed and $p_1 + p_2 < 1$ then by CLT they show the distribution of this diagnostic approaches $N(0, 1)$ as $n \to \infty$.

Geweke Diagnostic

- Suggestion is to use $n_1 = 0.1n$ and $n_2 = 0.5n$ (i.e. $p_1 = 10\%$ and $p_2 = 50\%$.)
- If we get a p-value of ≤ 0.05 then we reject the hypothesis that the first $p_1\%$ and last $p_2\%$ of the sample have the same mean.
- We can discard the first $p_1\%$ as burn-in and try again. . .

Example

```r
post=as.matrix(cbind(fit$lambdaz,fit$rhoz))
postmcmc=as.mcmc(post)
geweke.diag(post)
```

```
##
## Fraction in 1st window = 0.1
## Fraction in 2nd window = 0.5
##
##   var1   var2
##  1.018 -5.580
```

Example

```r
post=as.matrix(cbind(fit2$lambdaz,fit2$rhoz))
postmcmc=as.mcmc(post)
geweke.diag(post)
```

```
##
## Fraction in 1st window = 0.1
## Fraction in 2nd window = 0.5
##
##   var1   var2
## -117.8  238.5
```

Example

```r
post=as.matrix(cbind(fit3$lambdaz,fit3$rhoz))
postmcmc=as.mcmc(post)
geweke.diag(post)
```

```
##
## Fraction in 1st window = 0.1
## Fraction in 2nd window = 0.5
##
##    var1    var2
##  0.6674 -0.2512
```

Gelman-Rubin† Diagnostic

- A multi-chain diagnostic – if we start our MCMC from m different starting points, do they all converge to the same place?
- Approach is to consider m independent, separate MCMC runs. Often $m = 10$.
- We start these runs at extremes of the prior or from a overdispersed prior.
- The G& R idea is to decompose the variance of all the chains into a within-chain variance and between-chain variance and see if there is a significant difference.

† A. Gelman and D. Rubing (1992): *Inference from Iterative Simulation Using Multiple Sequences.* Statistical Science, vol. 7, pp.457–511.\

M.K. Cowles and B.P. Carlin (1996): *Markov Chain Monte Carlo Convergence Diagnostics: A Comparative Review,* Journal of the American Statistical Association, vol.91, pp.883–904.

Gelman-Rubin Diagnostic

- After discarding burn-in, first compute

$$\bar{\theta}_j = \frac{1}{n}\sum_{i=1}^{n}\theta_{ji}$$

  for each of the $j = 1, \ldots, m$ MCMC runs.

- Next calculate average within-chain variance as

$$W = \frac{1}{m}\sum_{j=1}^{m}\left[\frac{1}{n-1}\sum_{i=1}^{n}(\theta_{ji} - \bar{\theta}_j)^2\right]$$

- Finally calculate the between-chain variance as

$$B = \frac{n}{m-1}\sum_{j=1}^{m}(\bar{\theta}_j - \bar{\theta})^2$$

  where $\bar{\theta} = \frac{1}{m}\sum_{j=1}^{m}\bar{\theta}_j$.

Gelman-Rubin Diagnostic

- The total estimated variance is

$$\widehat{Var(\theta)} = \left(1 - \frac{1}{n}\right) W + \frac{1}{m} B.$$

- And the Gelman-Rubin statistic is

$$R = \frac{\widehat{Var(\theta)}}{W}.$$

- We want R to be close to 1. They suggest $R > 1.05$ indicates possible problems.
- Univariate, but a multivariate extension exists†.

† S. Brooks and A. Gelman (1998): *General methods for monitoring convergence of iterative simulations.* Journal of Computational and Graphical Statistics, vol7, pp.434–455.

Gelman-Rubin Diagnostic

```
postdraws=vector("list",6)
for(i in 1:6) postdraws[[i]]=fit[[i]]$rhoz
for(i in 1:6) postdraws[[i]]=as.mcmc(postrows[[i]])
postmulti=as.mcmc.list(postrows)
gelman.diag(postmulti,autoburnin=FALSE)
```

Effective Sample Size

- Calculates how many samples do you effectively have adjusting for the autocorrelation in your MCMC samples.
- If your sampler truly was i.i.d., then you would have N samples.
- But since there is usually dependence between samples, effectively you have $< N$ samples.

Example

```
post=as.matrix(cbind(fit2$lambdaz,fit2$rhoz))
postmcmc=as.mcmc(post)
effectiveSize(postmcmc)
```

```
##      var1      var2
## 14.218939  2.878563
```

Example

```r
post=as.matrix(cbind(fit3$lambdaz,fit3$rhoz))
postmcmc=as.mcmc(post)
effectiveSize(postmcmc)
```

```
##      var1     var2
## 873.7204 198.3363
```