A Gateway to all Postgraduate Courses

An MHRD project under its National Mission on Education through ICT (NME-ICT)

Subject: Statistics

Paper: Advanced R

Module: Convergence Diagnostics for MCMC algorithm

Development Team

Principal investigator: **Dr. Bhaswati Ganguli**,
Professor, Department of Statistics, University of Calcutta

Paper co-ordinator: **Dr. Abhra Sarkar**,
Duke University

Content writer: **Dr. Santu Ghosh**,
Lecturer, Department of Environmental Health Engineering, Sri Ramachandra University, Chennai

Content reviewer: **Dr. Bhaswati Ganguli**,
Professor, Department of Statistics, University of Calcutta

Outline

- Convergence
- Visual Inspection
- Gelman and Rubin Diagnostic
- Geweke Diagnostic
- Raftery and Lewis Diagnostic
- Heidelberg and Welch Diagnostic

A Multivariate Random Walk Metropolis-Hastings Sampler

- Let define a function that does the Random Walk Metropolis-Hastings sampling when supplied with the required arguments such as
  - (i) 'target density',
  - (ii) 'starting point(vector) of random walk',
  - (iii) 'variance-covariance matrix for the multivariate normal that is used as the proposal distribution for random-walk Metropolis-Hastings' and
  - (iv) 'burn-in length for the chain'.

```
myrwmetro <- function(target, N, x, VCOV, burnin = 0) {
    require(MASS)  #requires package MASS for normal sampling
    mysamples <- x
    for (i in 2:(burnin + N)) {
        propl <- mvrnorm(n = 1, x, VCOV)
        if (runif(1) < min(1, target(propl)/target(x)))
            x <- propl
        mysamples <- rbind(mysamples, x)
    }
    mysamples[(burnin + 1):(N + burnin), ]
}
```

An arbitrary bi-variate function

Let assume a bi-variate target function $f(x, y)$ as

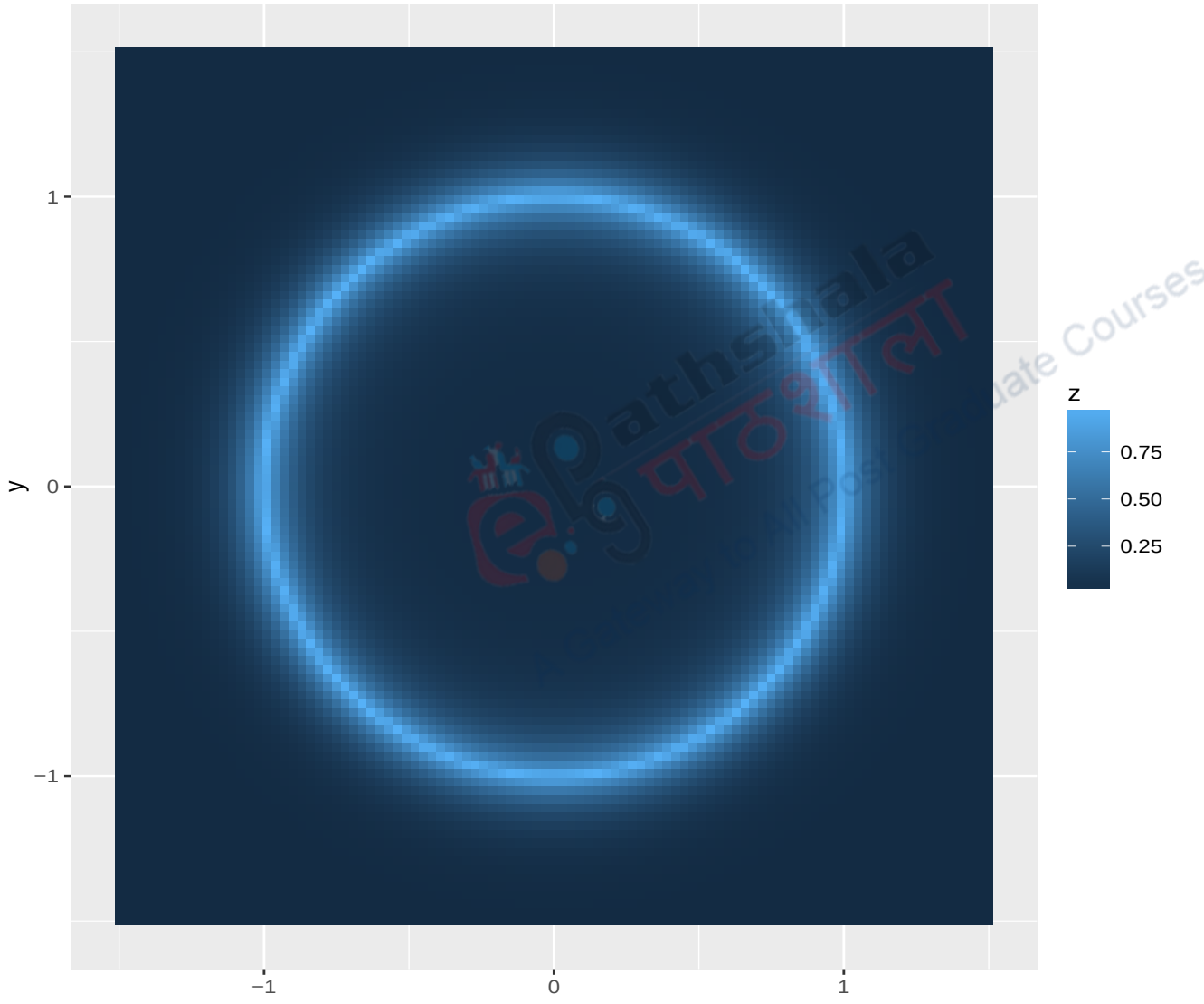
$$f(x, y) = e^{-5|x^2+Y^2-1|}.$$

This looks like a ring of radius 1 rising from the x-y plane and centered at the origin.

```
library(ggplot2)
curve3D <- function(f, from.x, to.x, from.y, to.y, n = 101) {
    x.seq <- seq(from.x, to.x, (to.x - from.x)/n)
    y.seq <- seq(from.y, to.y, (to.y - from.y)/n)
    eval.points <- expand.grid(x.seq, y.seq)
    names(eval.points) <- c("x", "y")
    eval.points <- transform(eval.points, z = apply(eval.points,
        1, function(r) {
            f(c(r["x"], r["y"]))
        }))
    p <- ggplot(eval.points, aes(x = x, y = y, fill = z)) + geom_tile()
    print(p)
}
g <- function(x) {
    exp(-5 * abs(x[1]^2 + x[2]^2 - 1))
}
```

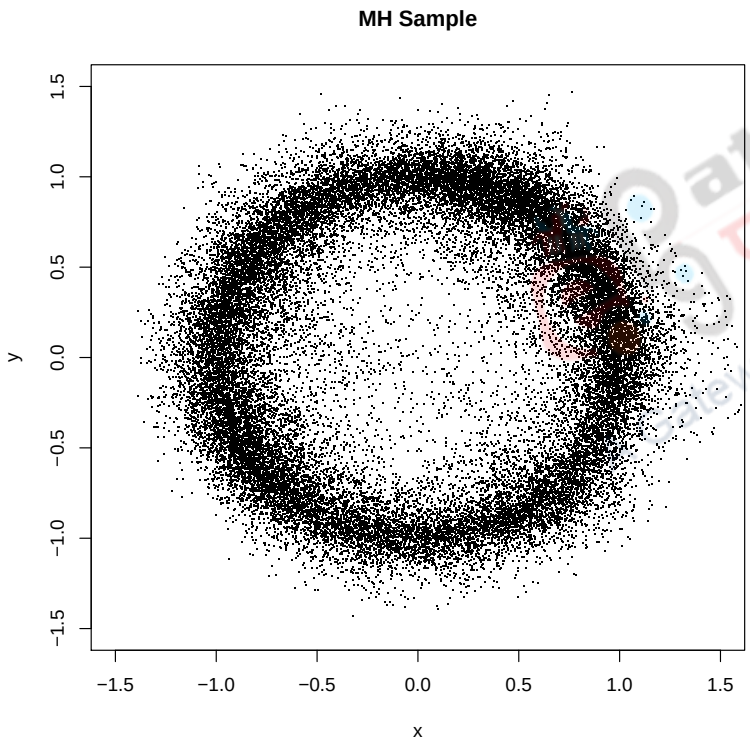
An arbitrary bi-variate function

```
curve3D(g,-1.5,1.5,-1.5,1.5,n=100)
```



MH sampling

```
vcov2D <- 0.01 * diag(2)
# Call the sampling function with the arguments
rsamp <- myrwmetro(g, 50000, c(0, 0), vcov2D)
plot(rsamp[, 1], rsamp[, 2], xlim = c(-1.5, 1.5), ylim = c(-1.5,
    1.5), main = "MH Sample", xlab = "x", ylab = "y", pch = ".")
```



MCMC convergence

- From the theory of Markov chains, we expect the chains to eventually converge to the stationary distribution, which is also our target distribution.
- However, there is no guarantee that our chain has converged after M draws.
- Question is then, how do we know whether the chain has actually converged?
- We can never be sure, but there are several tests we can do, both visual and statistical, to see if the chain appears to be converged.

Convergence Diagnostics in R

- For all the convergence diagnostics, we are going to use the coda package in R.
- Before we use the diagnostics, we should turn our chains into mcmc objects.
- We can tell the mcmc() function to burn-in or drop draws with the start and end arguments.
- mcmc() also has a thin argument, which only tells it the thinning interval that was used.

```
library(coda)
mh.draws <- mcmc(rsamp)
```

Summary of mcmc object

```
summary(mh.draws)

##
## Iterations = 1:50000
## Thinning interval = 1
## Number of chains = 1
## Sample size per chain = 50000
##
## 1. Empirical mean and standard deviation for each variable,
##    plus standard error of the mean:
##
##          Mean     SD Naive SE Time-series SE
## [1,] 0.01908 0.7148 0.003197        0.06588
## [2,] 0.09037 0.6933 0.003101        0.06271
##
## 2. Quantiles for each variable:
##
##        2.5%     25%     50%    75% 97.5%
## var1 -1.075 -0.6825 0.06762 0.6925 1.083
## var2 -1.047 -0.5681 0.17540 0.7337 1.088
```

Summary of `mcmc` object

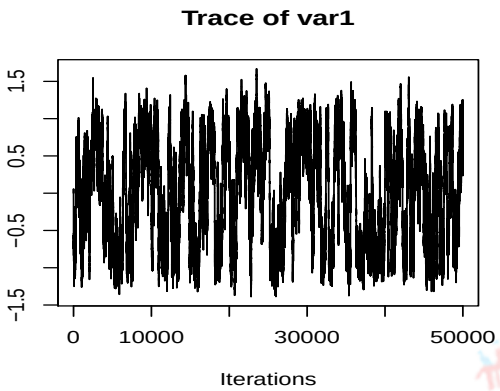
- The results give the means, standard deviations, and quantiles for each components of random variate(or posterior in case of Bayesian analysis).
- The "naive" standard error is the standard error of the mean, which captures simulation error of the mean rather than (posterior) uncertainty.
- The time-series standard error adjusts the "naive" standard error for autocorrelation.

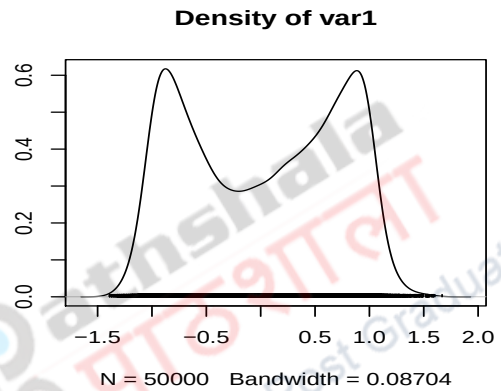
Visual Inspection

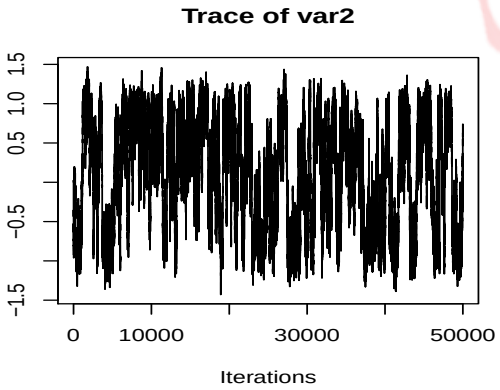
- One way to see if our chain has converged is to see how well our chain is mixing, or moving around the parameter space.
- If our chain is taking a long time to move around the parameter space, then it will take longer to converge.
- We can see how well our chain is mixing through visual inspection.
- We need to do the inspections for every parameter.
- A *traceplot* is a plot of the iteration number against the value of the draw of the parameter at each iteration.
- We can see whether our chain gets stuck in certain areas of the parameter space, which indicates bad mixing.

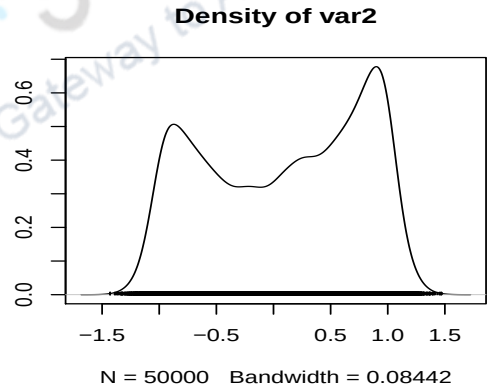
Traceplots and Density Plots

```
plot(mh.draws)
```



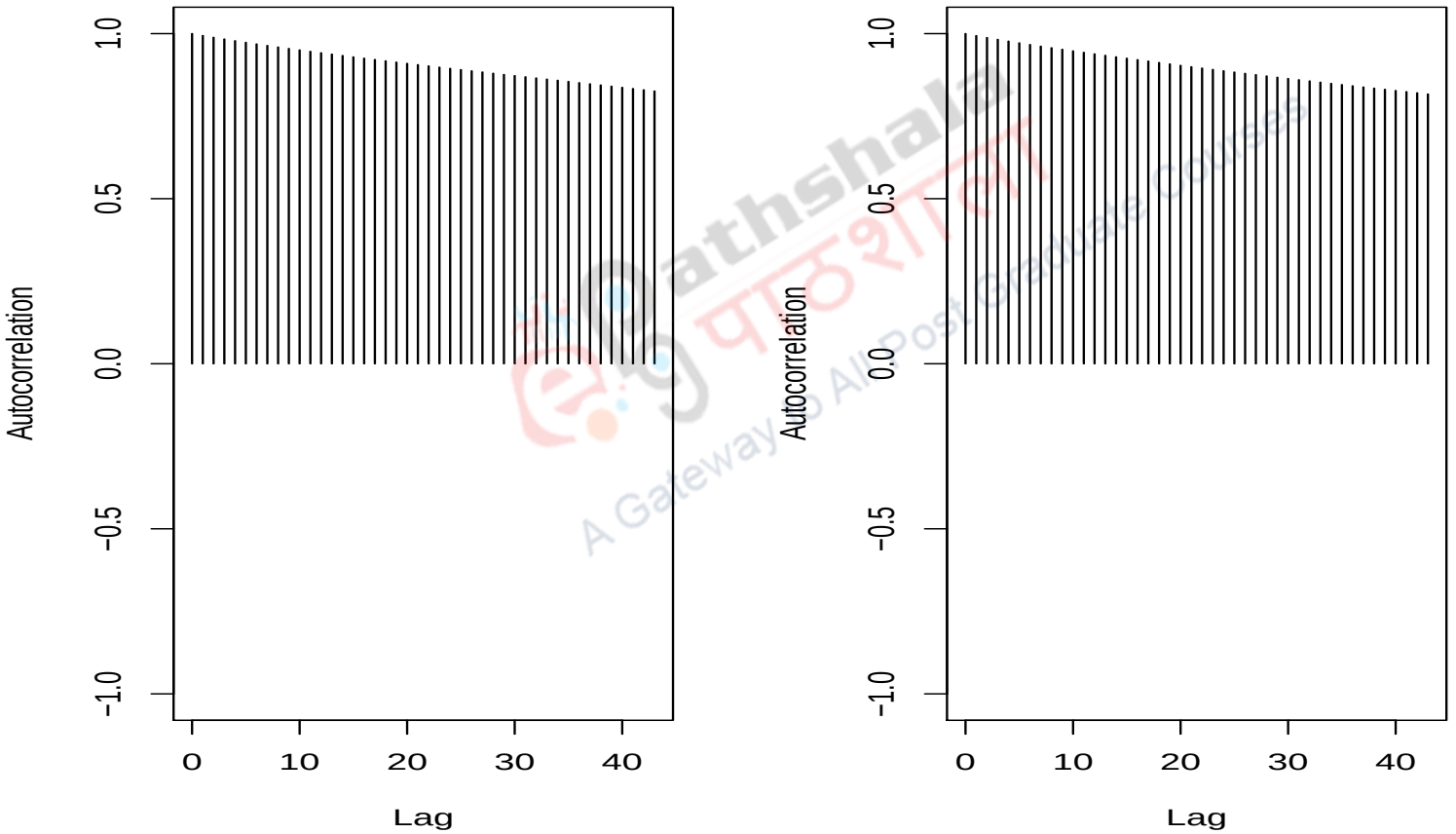






Autocorrelation Plots

```
autocorr.plot(mh.draws)
```



Rejection rate of MH algorithm

- We can also get a rejection rate for the Metropolis-Hastings algorithm using the `rejectionRate()` function.
- To get the acceptance rate, we just want 1- rejection rate.

```
rejectionRate(mh.draws)

##      var1      var2
## 0.2933059 0.2933059

acceptance.rate <- 1 - rejectionRate(mh.draws)
acceptance.rate

##      var1      var2
## 0.7066941 0.7066941
```

Gelman and Rubin Multiple Sequence Diagnostic

Steps (for each parameter):

1. Run $m \geq 2$ chains of length $2n$ from overdispersed starting values.
2. Discard the first n draws in each chain.
3. Calculate the within-chain and between-chain variance.
4. Calculate the estimated variance of the parameter as a weighted sum of the within-chain and between-chain variance.
5. Calculate the potential scale reduction factor.

Gelman and Rubin Multiple Sequence Diagnostic: Interpretation

- If we have more than one parameter, then we need to calculate the potential scale reduction factor for each parameter.
- We should run our chains out long enough so that all the potential scale reduction factors are small enough(perhaps less than 1.1 or 1.2)
- We can then combine the mn total draws from our chains to produce one chain from the stationary distribution.

Gelman & Rubin Diagnostic in R

- First, we run M chains at different (should be overdispersed) starting values ($M = 5$ here) and convert them to mcmc objects.
- We then put the M chains together into an `mcmc.list`.
- Then run the diagnostic with the `gelman.diag()` function.

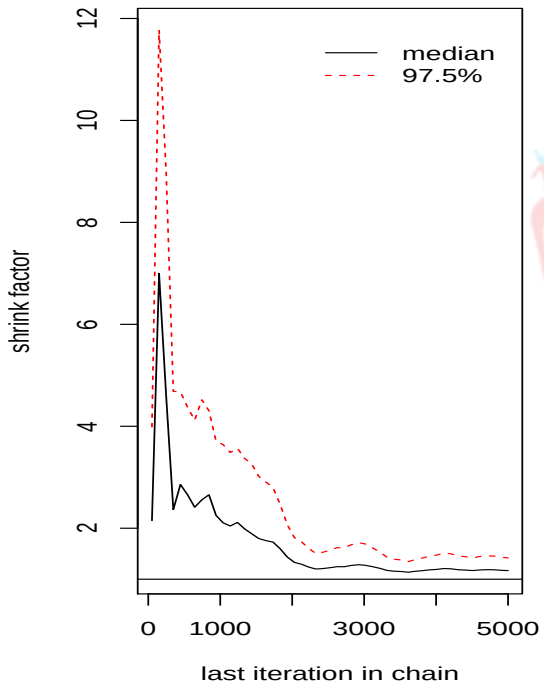
```
mh.draws1 <- mcmc(myrwmetro(g, 5000, c(0, 0), vcov2D))
mh.draws2 <- mcmc(myrwmetro(g, 5000, c(0, 0), vcov2D))
mh.draws3 <- mcmc(myrwmetro(g, 5000, c(0, 0), vcov2D))
mh.draws4 <- mcmc(myrwmetro(g, 5000, c(0, 0), vcov2D))
mh.draws5 <- mcmc(myrwmetro(g, 5000, c(0, 0), vcov2D))
mh.list <- mcmc.list(list(mh.draws1, mh.draws2, mh.draws3, mh.draws4,
    mh.draws5))
gelman.diag(mh.list)
```

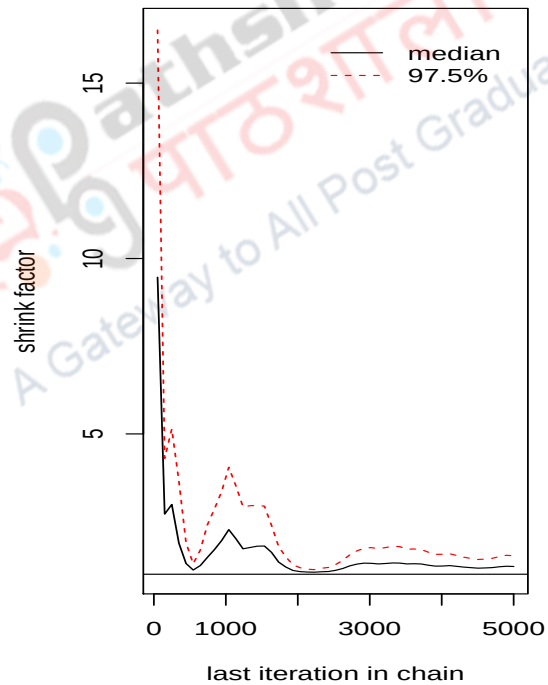
```
## Potential scale reduction factors:
##
##      Point est. Upper C.I.
## [1,]       1.17       1.42
## [2,]       1.22       1.53
##
## Multivariate psrf
##
## 1.24
```

Gelman & Rubin Diagnostic in R

- We can see how the potential scale reduction factor changes through the iterations using the `gelman.plot()` function.

```
gelman.plot(mh.list)
```





Geweke's Diagnostic

- Can we treat, say, the first 10,000 iterations as burn-in?
- In a long enough chain whose trace plot suggests convergence to the target distribution, we assume the second half of the chain has converged to the target distribution, and we test if the first 10% can be treated as burn-in.
- Idea: mimic the simple two sample test of means. If the mean of the first 10% is not significantly different from the last 50%, then we conclude the target distribution converged somewhere in the first 10% of the chain.
- Geweke's diagnostics use spectral densities to estimate the sample variances (Geweke, 1992).

Bivariate Geweke's Diagnostic

- Originally, Geweke's diagnostic was only designed for the univariate scenario. One way to expand it to a bivariate scenario is to also consider the sample covariance between the two variables X and Y.
- Suppose, a bivariate chain (X, Y) is considered.
- After performing Geweke's diagnostic for each chain,calculate $\gamma_i = (x_i - \bar{x})(y_i - \bar{y})$ for each draw of X and Y, and then perform Geweke's diagnostic for the sample of . Burn-in is chosen as the early proportion that passes all three diagnostics.

Univariate:MH RW sampler

```
MH.Gamma = function(NN = 10000, start = 0.1, sd.norm = 0.3) {
    X = length(NN)
    X[1] = start
    for (i in 2:NN) {
        Xcandi = rnorm(1, mean = X[i - 1], sd = sd.norm)
        if ((dexp(Xcandi, 3)/dexp(X[i - 1], 3)) >= runif(1)) {
            X[i] = Xcandi
        } else {
            X[i] = X[i - 1]
        }
    }
    X
}
library(coda)
set.seed(100)
object = MH.Gamma()
gd <- geweke.diag(mcmc(object))
gd
```

```
##
## Fraction in 1st window = 0.1
## Fraction in 2nd window = 0.5
##
##   var1
## 0.5701
```

```
pnorm(abs(gd$z), lower.tail = FALSE) * 2  # P-value
```

```
##      var1
## 0.5686305
```

Bivariate Normal Gibbs sampler

```
Gibbs.bnorm = function(NN = 10000, rho = 0.5) {
    X = Y = length(NN)
    X[1] = Y[1] = 0
    for (i in 2:NN) {
        X[i] = rnorm(1, rho * Y[i - 1], sqrt(1 - rho^2))
        Y[i] = rnorm(1, rho * X[i], sqrt(1 - rho^2))
    }
    return(list(X = X, Y = Y))
}
set.seed(100)
object = Gibbs.bnorm()
# P-value for X
pnorm(abs(geweke.diag(mcmc(object$X))$z), lower.tail = FALSE) * 2
```

```
##      var1
## 0.4069461
```

```
# P-value for Y
pnorm(abs(geweke.diag(mcmc(object$Y))$z), lower.tail = FALSE) * 2
```

```
##      var1
## 0.3292853
```

```
# P-value for sample covariance
gamma = (object$X - mean(object$X)) * (object$Y - mean(object$Y))
pnorm(abs(geweke.diag(mcmc(gamma))$z), lower.tail = FALSE) * 2
```

```
##      var1
## 0.1977521
```

Raftery and Lewis Diagnostic

- Suppose we want to measure some posterior quantile of interest q.
- If we define some acceptable tolerance r for q and a probability s of being within that tolerance, the Raftery and Lewis diagnostic will calculate the number of iterations N and the number of burn-ins M necessary to satisfy the specified conditions.
- The diagnostic was designed to test the number of iterations and burn-in needed by first running and testing shorter pilot chain.
- In practice, we can also just test our normal chain to see if it satisfies the results that the diagnostic suggests.

Raftery and Lewis Diagnostic:Input

- Select a posterior quantile of interest q (for example, the 0.025 quantile).
- Select an acceptable tolerance r for this quantile (for example, if $r = 0.005$, then that means we want to measure the 0.025 quantile with an accuracy of ± 0.005).
- Select a probability s, which is the desired probability of being within $(q - r, q + r)$.
- Run a "pilot" sampler to generate a Markov chain of minimum length given by rounding up

$$n_{min} = \left[\Phi^{-1}\left(\frac{s+1}{2}\right) \frac{\sqrt{q(1-q)}}{r} \right]^2$$

where $\Phi^{-1}(.)$ is the inverse of the normal CDF.

Raftery and Lewis Diagnostic
Result from Bivariate RW sampler

```
raftery.diag(mh.draws, q = 0.025, r = 0.005, s = 0.95)

##
## Quantile (q) = 0.025
## Accuracy (r) = +/- 0.005
## Probability (s) = 0.95
##
##  Burn-in  Total Lower bound  Dependence
##  (M)      (N)   (Nmin)       factor (I)
##  42       44184 3746         11.8
##  60       72888 3746         19.5
```

- M: number of burn-ins necessary
- N: number of iterations necessary in the Markov chain
- N_{min}: minimum number of iterations for the "pilot" sampler
- I: dependence factor, interpreted as the proportional increase in the number of iterations attributable to serial dependence.
- High dependence factors (> 5) are worrisome and may be due to influential starting values, high correlations between coefficients, or poor mixing.

Notes

- The Raftery-Lewis diagnostic will differ depending on which quantile q you choose.
- Estimates tend to be conservative in that it will suggest more iterations than necessary.
- It only tests marginal convergence on each parameter.
- Nevertheless, it often works well with simple models.

Heidelberg and Welch Diagnostic

- The Heidelberg and Welch diagnostic calculates a test statistic (based on the Cramer-von Mises test statistic) to accept or reject the null hypothesis that the Markov chain is from a stationary distribution.
- The diagnostic consists of two parts.

Part-1

1. Generate a chain of N iterations and define an α level.
2. Calculate the test statistic on the whole chain. Accept or reject null hypothesis that the chain is from a stationary distribution.
3. If null hypothesis is rejected, discard the first 10% of the chain. Calculate the test statistic and accept or reject null.
4. If null hypothesis is rejected, discard the next 10% and calculate the test statistic.
5. Repeat until null hypothesis is accepted or 50% of the chain is discarded. If test still rejects null hypothesis, then the chain fails the test and needs to be run longer.

Heidelberg and Welch Diagnostic

Part-2

1. If the chain passes the first part of the diagnostic, then it takes the part of the chain not discarded from the first part to test the second part.
2. The halfwidth test calculates half the width of the $(1-\alpha)\%$ credible interval around the mean.
3. If the ratio of the halfwidth and the mean is lower than some ϵ, then the chain passes the test. Otherwise, the chain must be run out longer.

```
heidel.diag(mh.draws)


##
##      Stationarity start     p-value
##      test         iteration
## var1 passed       1         0.5192
## var2 passed       1         0.0994
##
##      Halfwidth Mean   Halfwidth
##      test
## var1 failed    0.0191 0.129
## var2 failed    0.0904 0.123
```

Reference:

1. Chirstian P. Robert & George Casella(2010). Introducing Monte Carlo Methods with R. Springer.
2. http://math.arizona.edu/~piegorsch/675/GewekeDiagnostics.pdf
3. http://www.people.fas.harvard.edu/~plam/teaching/methods/convergence/convergence_print.pdf

Thank You