Convergence Diagnostic for Markov Chain Monte Carlo Methods

I. Abstract

Our goal is to provide researchers utilizing MCMC methods with a comprehensive guide on the practical application of diagnostic tools to ensure dependable results. To evaluate the effectiveness of these diagnostic tools, we generated two distinct chains and wrote codes for five methods, including Graphical observation, Effective sample size, Geweke, Heidelberg and Welch, and Gelman-Rubin to assess their convergence. In particular, we focused on the Geweke and ESS calculations employed in prevalent R packages such as "coda" and "mcmcse", while also including additional features such as considering batch means, spectral variance estimator method **[2]**, we also considered multivariate cases. Additionally, we modified the Geweke code to consider batch means **[2]** and multivariate cases which is also kinda an innovation.

II. Introduction

The Markov Chain Monte Carlo (MCMC) method is a widely used tool for statistical inference and simulation-based estimation. The fundamental idea of MCMC is that if simulating from a target density π is difficult so that the ordinary Monte Carlo method based on independent and identically distributed (iid) samples cannot be used for making inference on π, it may be possible to construct a Markov chain $\{Xn\}_{n\geq 0}$ with stationary density π for forming Monte Carlo estimators **[2]**.

It is particularly useful for problems where the likelihood function is intractable or computationally expensive. In such cases, MCMC methods allow us to simulate samples from the posterior distribution using only the likelihood function and prior distributions. However, MCMC simulations can suffer from slow or even failed convergence to the target distribution. This is because the algorithm works by constructing a Markov chain that produces a sequence of samples, and the convergence of the chain to the target distribution can be slow or uncertain.

To overcome this issue, it is crucial to use convergence diagnostics to assess the reliability of MCMC simulations. Convergence diagnostics are statistical tools that help to evaluate whether the MCMC chain has reached convergence to the target distribution. The diagnostic tools provide an indication of whether the MCMC simulations have produced a sufficient number of independent samples from the target distribution. The diagnostic tools can help to identify issues with the MCMC simulations and provide guidance on how to improve the simulations. This paper aims to provide a comprehensive review of commonly used diagnostic tools for evaluating the convergence of MCMC simulations. The diagnostic tools covered in this paper include the Geweke statistic, Heidelberger and Welch test, effective sample size, Gelman-Rubin diagnostic, and graphical outputs.

III. MCMC Diagnostics

Our first code is used to implement a Markov Chain Monte Carlo (MCMC) simulation using the Metropolis-Hastings (MH) algorithm. In this specific case, two chains were generated to test different target distributions. The first chain uses an exponential target distribution with $\lambda=1$ of $\pi(x) = \exp(-x)$ and a proposal distribution of $P(x'| x) = 0.5\exp(-0.5x)$. The second chain uses the same target distribution of $\pi(x) = \exp(-x)$ but with a proposal distribution of $P(x'| x) = 5\exp(-5x)$. The purpose of the simulation is to generate a sequence of random samples from the target distribution that can be used for statistical inference or simulation-based estimation. In addition, the generated samples from the two chains will be used to assess the convergence of MCMC simulations by applying the aforementioned diagnostic methods. The code includes a burn-in period to allow the chains to converge to the target distribution, and the final output is a matrix of the generated samples after discarding the burn-in period if specified.

A. Effective Sample Size

The second code contains several functions that are used to calculate effective sample size (ESS) (which is defined in **[3])** for Markov Chain Monte Carlo (MCMC) simulations. The ESS is a measure of the number of independent samples in a MCMC sample, which is important for estimating the precision of the sample mean and other summary statistics. The “min.ESS” function computes the minimum ESS required for a given dimensionality p of the parameter space, a significance level alpha, and a desired error margin epsilon.

$$\widehat{\text{mESS}} \geq \frac{2^{2/p}\pi}{(p\Gamma(p/2))^{2/p}} \frac{\chi^2_{1-\alpha,p}}{\varepsilon^2}$$

where ε is the desired level of precision for the volume of the 100(1 − α)% asymptotic confidence region, and $\chi2_{2,\,1-\alpha}$ is the (1 − α) quantile of $\chi2_{p}$.

The “get.ESS” function is used to estimate the ESS of a given MCMC chain x, using one of several methods depending on the value of the argument ESS. If the ESS = 1, the function calculates the ESS using the following formula (the most common definition (Robert and Casella 2004)):

$$ESS = \frac{n}{1 + 2\sum_{i=1}^{\infty} \text{Corr}(g(X_0), g(X_1))}$$

where g is a real valued function.

If ESS is 'coda', then it uses the mcse.multi function from the “mcmcse” package. If ESS is “batchmeans”, then it uses the mcse_batchmeans function defined in the code **[2]** (Basically this two are equivalent while method “batchmeans” will use use local “mcse_batchmeans” function). This two method also considered multivariate setting, that is, when p ≥ 1, Vats et al. (2019) define multivariate ESS (mESS) as

$$\text{mESS} = n\left(\frac{|\Lambda_g|}{|\Sigma_g|}\right)^{1/p}$$

where Λ_g is the population covariance matrix and Σ_g is defined as an estimate of the spectral density at frequency zero (introduced later).

The “mcse_batchmeans” function is used to estimate the Monte Carlo standard error (MCSE) of a given MCMC chain x using the batch means method. This method involves dividing the chain into batches, computing the mean of each batch, and then estimating the standard deviation of the batch means. The MCSE is then estimated as the standard deviation of the batch means divided by the square root of the number of batches. In the formula above, Λ_g is the square of MCSE.

The spectrum0.ar function is used to compute the spectral density **[2]** and autoregressive (AR) model order for each column of a given matrix x (Σ_g in the above formula). The function fits each column to a linear regression model and checks if its residuals are zero. If the residuals are zero, then the column is regarded as a constant sequence with a spectral density of zero and an AR model order of zero. Otherwise, an AR model is used to fit the column, and its spectral density and AR model order are computed.

Spectral estimation values are used to estimate the variances before and after a given point, and Geweke's statistics are then calculated by taking the weighted average of these two variances. The zero-frequency spectral estimate is used as the variance estimate value for Geweke's statistics calculation, as it can reflect the overall variance in time series data and remove the influence of autoregressive coefficients from spectral estimation. However, the AR method is not always more accurate than directly calculating the sample variance, especially when there are outliers or anomalies in the data that may affect the stationarity of the time series.

B. Geweke

The third code is used for testing the convergence of a Markov Chain using Geweke's test. Geweke's test calculates the standardized difference between the means of the first a_n samples (SSA) and the last b_n samples of the chain (SSB), and tests whether this value is significantly different from zero using a two-sided t-test **[1]**. Where

$$\mathrm{SSA} = \frac{1}{A-1}\sum_{t=1}^{A}(X_t - \bar{X}_{1:A})^2 \qquad \mathrm{SSB} = \frac{1}{n-B}\sum_{t=B}^{n-1}(X_t - \bar{X}_{B:n-1})^2$$

$$z = \frac{\bar{X}_{1:A} - \bar{X}_{B:n-1}}{\sqrt{\frac{\mathrm{SSA}}{A} + \frac{\mathrm{SSB}}{n-B+1}}}$$

The function “geweke_toy” implements Geweke's test and returns the Geweke statistic and p-value for a given Markov chain. The function takes in the Markov chain (chain), the proportions of the chain to use for the first set of samples (a) and the last set of samples (b), and the method to use for variance calculation (method). Two methods are available for variance calculation: normal uses the var function, while “spectral” uses a spectral variance estimator calculated by the spectrum0.ar function.

The “geweke” is a more complex implementation of Geweke's test that divides the Markov chain into multiple sub-sequences to increase the sample size and improve the accuracy of the test. The function takes in the Markov chain (x), the proportions of the chain to use for the first set of samples (a) and the last set of samples (b), the number of batches **[2]** to divide the chain into (num_batches), and the method to use for variance calculation (method).

The function initializes vectors to store the Geweke statistics and p-values for each batch, and then calculates the Geweke statistic and p-value for each batch using the same method as “geweke_toy”. The function returns a list of the Geweke statistics and p-values for each batch. Moreover, we calculated the proportion of p-values below 0.05 among all p-values and plotted the statistics and p-values, so a high proportion means that p-values calculated in multiple batches tend to reject the null hypothesis: the chain has converged.

We also considered a function for multi-dimensional cases using a “geweke multi” function, which performs Geweke's convergence diagnostic test on a multi-dimensional input data. The function takes in additional parameters such as num_batches and method to specify the number of batches to use and the method for calculating the test statistic, respectively. The function returns the test statistics and p-values for each dimension of the input data. Our code then generates a random multi-dimensional input data using the ‘mvrnorm’ function from the “MASS” package, and applies the “geweke_multi” function to it.

C. Heidelberger and Welch

Our fourth R code is an implementation of the Heidelberger and Welch test for checking the stationarity and adequacy of Markov chains, as described in Heidelberger and Welch's papers from 1981 and 1983. The test consists of two parts: a stationary portion test and a half-width test. The stationary portion test assesses the stationarity of a Markov chain by testing the hypothesis that the chain comes from a covariance stationary process. The half-width test **[1]** checks whether the Markov chain sample size is adequate to estimate the mean values accurately. Given $\{\theta^t\}$, set $S_0 = 0$, $S_n = \sum_{t=1}^{n} \theta^t$, and $\bar{\theta} = (1/n) \sum_{t=1}^{n} \theta^t$.

You can construct the following sequence with s coordinates on values from $\frac{1}{n}, \frac{2}{n}, \ldots, 1$:

$$B_n(s) = \frac{S_{[ns]} - [ns]\bar{\theta}}{n\hat{p}(0)}$$

where [] is the rounding operator, and $\widehat{p}(0)$ is an estimate of the spectral density at zero frequency that uses the second half of the sequence.

The statistic used in these procedures is the Cramer–von Mises statistic; that is $\int_0^1 B_n(S)^2 ds$. As n goes infinite, the statistic converges in distribution to a standard Cramer–von Mises distribution.

The “pcramer” function is a helper function that computes the probability of exceeding the given value of a test statistic, based on the Cramer-Lundberg approximation. The Cramer-Lundberg approximation is an approximation of the Cramer-von Mises test and is used to calculate the p-value of the Cramer-von Mises statistic. The form of the Cramer-Lundberg approximation is as follows:

$$p_{value} = \sum_{k=0}^{3} z_k \cdot e^{-u_k} \cdot K_{\frac{1}{4}}(u_k)$$

where $z_k = \frac{\Gamma(k+0.5) \cdot \sqrt{4k+1}}{\Gamma(k+1) \cdot \pi^{3/2} \cdot \sqrt{q}}$, $u_k = \frac{(4k+1)^2}{16q}$ and

$$K_{\frac{1}{4}}(x) = z_k \cdot e^{-x} \cdot \text{besselK}(x = u_k, \nu = \frac{1}{4})$$

This test can be performed repeatedly on the same chain, and it helps you identify a time t when the chain has reached stationarity. The whole chain, $\{\theta^t\}$, is first used to construct the Cramer–von Mises statistic. If it passes the test, you can conclude that the entire chain is stationary. If it fails the test, you drop the initial 10% of the chain and redo the test by using the remaining 90%. This process is repeated until either a time t is selected or it reaches a point where there is not enough data remaining to construct a confidence interval (the cutoff proportion is set to be 50%).

The RHW quantifies accuracy of the 1 - α level confidence interval of the mean estimate by measuring the ratio between the sample standard error of the mean and the mean itself. In other words, you can stop the Markov chain if the variability of the mean stabilizes with respect to the mean.val. The RHW for a confidence interval of level 1 - α is:

$$\mathrm{RHW} = \frac{z_{(1-\alpha/2)} \cdot (\hat{s}_n/n)^{1/2}}{\hat{\theta}}$$

The “Heidelberger” function takes as input a matrix of data x and optional parameters “eps” and “p-value.” It returns a matrix containing the test results for each column of x, including the starting position of the stationary portion, the p-value of the test, the half-width of the stationary portion, and whether the half-width test passed or failed.

D. Gelman-Rubin

The fifth code presents the implementation of the Gelman-Rubin diagnostic test to verify if parallel chains with dispersed initial values converge to the same target distribution. This method is useful for detecting multi-modal posterior distribution and to identify the need to run a longer chain. The function “gelman_toy” calculates the potential scale reduction factors (PSRF) by estimating the between-chain variance, within-chain variance, and the PSRF R.

There are two main problems with the toy model. The first small problem is that the improved version of Brooks and Gelman (1997) was not considered when calculating the PSRF statistics. This improved version takes into account the possible existence of multicollinearity between samples and considers the effective degrees of freedom at the end of the calculation. The second problem is that the function only considers chains with the same target distribution. That is, even if multiple Markov chains are input, they are considered to have the same target distribution.

Next, we consider ways to solve the above problems. In Brooks and Gelman's (1998) modified version of PSRF, d_hat represents the effective number of degrees of freedom for the model parameters, calculated as:$2\widehat{V}^2$. At the same time, when calculating the variance, we need to calculate the covariance between different variables (because the target distribution may be different at this time), so we need to use cov.wb to calculate the covariance and use if statement to judge whether it is multivariate. Specific calculation method is below **[1]**:

$$\begin{aligned} B &= \frac{n}{M-1}\sum_{m=1}^{M}(\bar{\theta}^{\cdot}_{m}-\bar{\theta}^{\cdot}_{\cdot})^2, \text{ where } \bar{\theta}^{\cdot}_{m}=\frac{1}{n}\sum_{t=1}^{n}\theta^{t}_{m}, \ \bar{\theta}^{\cdot}_{\cdot}=\frac{1}{M}\sum_{m=1}^{M}\bar{\theta}^{\cdot}_{m} \\ W &= \frac{1}{M}\sum_{m=1}^{M}s^2_m, \text{ where } s^2_m=\frac{1}{n-1}\sum_{t=1}^{n}(\theta^{t}_{m}-\bar{\theta}^{\cdot}_{m})^2 \end{aligned}$$

The posterior marginal variance, **var(θ|y)**, is a weighted average of W and B. The estimate of the variance:

$$\widehat{V}=\frac{n-1}{n}W+\frac{M+1}{nM}B$$

A refined version of PSRF is calculated, as suggested by Brooks and Gelman (1997), as:

$$\widehat{\mathrm{R}}_c=\sqrt{\frac{\hat{d}+3}{\hat{d}+1}\cdot\frac{\widehat{V}}{W}}=\sqrt{\frac{\hat{d}+3}{\hat{d}+1}\left(\frac{n-1}{n}+\frac{M+1}{nM}\frac{B}{W}\right)}$$

where $\hat{d} = \dfrac{2\widehat{V}^2}{\widehat{\text{Var}}(\widehat{V})}$

where
$$\widehat{\text{Var}}(\widehat{V}) = \left(\frac{n-1}{n}\right)^2 \frac{1}{M}\widehat{\text{Var}}(s_m^2) + \left(\frac{M+1}{nM}\right)^2 \frac{2}{M-1}B^2 + 2\frac{(M+1)(n-1)}{n^2 M}\frac{n}{M}\left(\widehat{\text{cov}}(s_m^2, (\bar{\theta}_m^{\cdot})^2) - 2\bar{\theta}_{\cdot}^{\cdot}\widehat{\text{cov}}(s_m^2, \bar{\theta}_m^{\cdot})\right)$$

In particular, our refined function “gelman” solves the problems in the toy example by considering the effective number of degrees of freedom for the model parameters (using $\widehat{d}$ to modify the variance estimation bias and reflect the degree of autocorrelation of MCMC simulation results). The refined function also considers when the target distributions are different from each other.

When the autoburnin parameter is set to TRUE by default, the “gelman” function automatically removes the first half of the MCMC chain (the burn-in period). In practical experience, it has been repeatedly verified that when the option "autoburnin" is set to TRUE, non-convergent results tend to be produced by the function (the potential scale reduction factors increase, as do the point estimates and upper confidence intervals). Investigation is needed to determine whether setting autoburnin to TRUE by default is appropriate.

A large PSRF indicates that the between-chain variance is substantially greater than the within-chain variance, so that longer simulation is needed. If the PSRF is close to 1, you can conclude that each of the M chains has stabilized, and they are likely to have reached the target distribution.

It is best to choose different initial values for all M chains. The initial values should be as dispersed from each other as possible so that the Markov chains can fully explore different parts of the distribution before they converge to the target. Similar initial values can be risky because all of the chains can get stuck in a local maximum; that is something this convergence test cannot detect. If you do not supply initial values for all the different chains, the procedures generate them for you.

E. Graphical Analysis

Our sixth and final code performs a Markov Chain Monte Carlo (MCMC) simulation using the Metropolis-Hastings algorithm to estimate the posterior distribution of a target distribution with a known density function π(x). We test this on chain 1; once again, the target distribution is π(x) = exp(-x), and a proposal density function of P(x'| x) = 0.5exp(-0.5x). The MCMC chain is initialized with a starting value of 0.1 and runs for n=323,700 iterations, with a burn-in of 0. The final MCMC chain is then plotted using the ggplot2 package, and the variance and summary statistics are also computed. The "mcmcse" package is used to estimate the effective sample size of the chain, and the acf() function is used to plot the autocorrelation of the chain.

IV. Comparison

Table 1: Convergence Diagnostic in the Bayesian Procedures (MCMC)[1]

Name	Description	Interpretation of the Test
Gelman-Rubin	Uses parallel chains with dispersed initial values to test whether they all converge to the same target distribution. Failure could indicate the presence of a multi-mode posterior distribution (different chains converge to different local modes) or the need to run a longer chain (burn-in is yet to be completed).	One-sided test based on a variance ratio test statistic. Large bRc values indicate rejection.
Geweke	Tests whether the mean estimates have converged by comparing means from the early and latter part of the Markov chain.	Two-sided test based on a z-score statistic. Large absolute z values indicate rejection.
Heidelberger-Welch (stationarity test)	Tests whether the Markov chain is a covariance (or weakly) stationary	One-sided test based on a Cramer–von Mises statistic.

	process. Failure could indicate that a longer Markov chain is needed.	Small p-values indicate rejection.
Heidelberger-Welch (half-width test)	Reports whether the sample size is adequate to meet the required accuracy for the mean estimate. Failure could indicate that a longer Markov chain is needed.	If a relative half-width statistic is greater than a predetermined accuracy measure, this indicates rejection.
Effective Sample Size	Relates to autocorrelation; measures mixing of the Markov chain.	Large discrepancy between the effective sample size and the simulation sample size indicates poor mixing.

V. Results and Discussion

The first chain with the target distribution is $\pi(x) = \exp(-x)$, and the proposal density function of $P(x'| x) = 0.5\exp(-0.5x)$.

The second chain with the target distribution is $\pi(x) = \exp(-x)$, and the proposal density function of $P(x'| x) = 5\exp(-5x)$.

We run each chain 323,700 iterations and then do the convergence diagnostic.

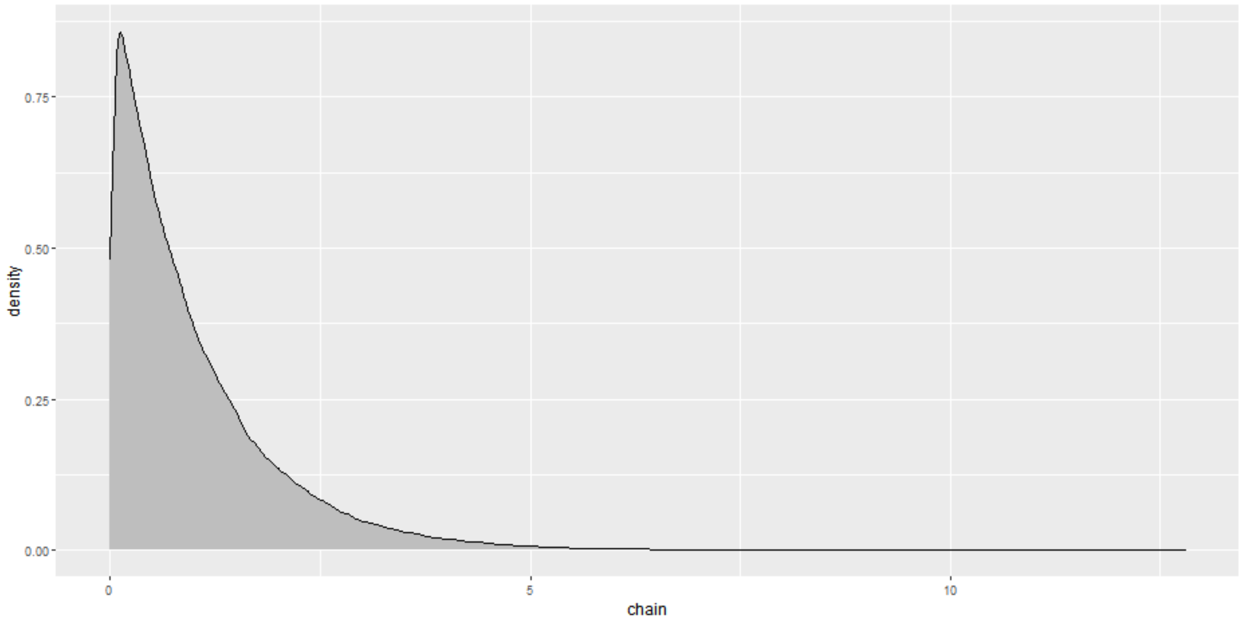
Actually, due to improper proposal distribution selection, the second Markov chain tends to not converge, which can be clearly seen from the graphical representation in figure 6, where the ACF function values remain high even at lag=50, while the first chain quickly converges.

Therefore, an appropriate convergence criterion would lead to accepting the null hypothesis of convergence for the first chain and rejecting the null hypothesis of convergence for the second chain.

A. Graphical Output Results

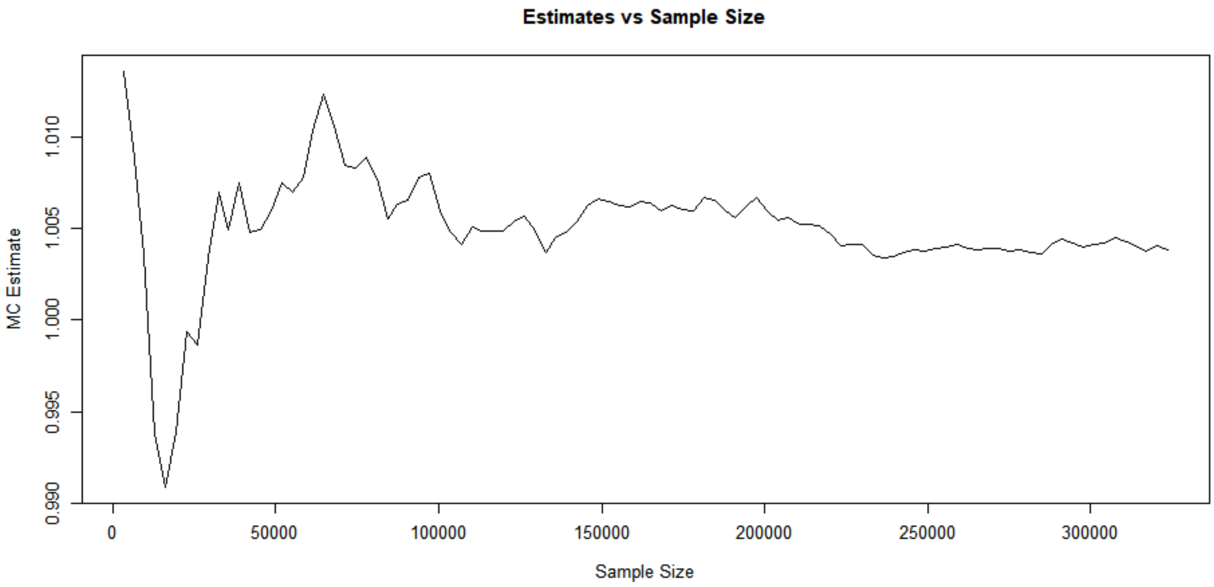
First chain:

Figure 1: Empirical pdf simulation of the first chain (π(x) = exp(-x))



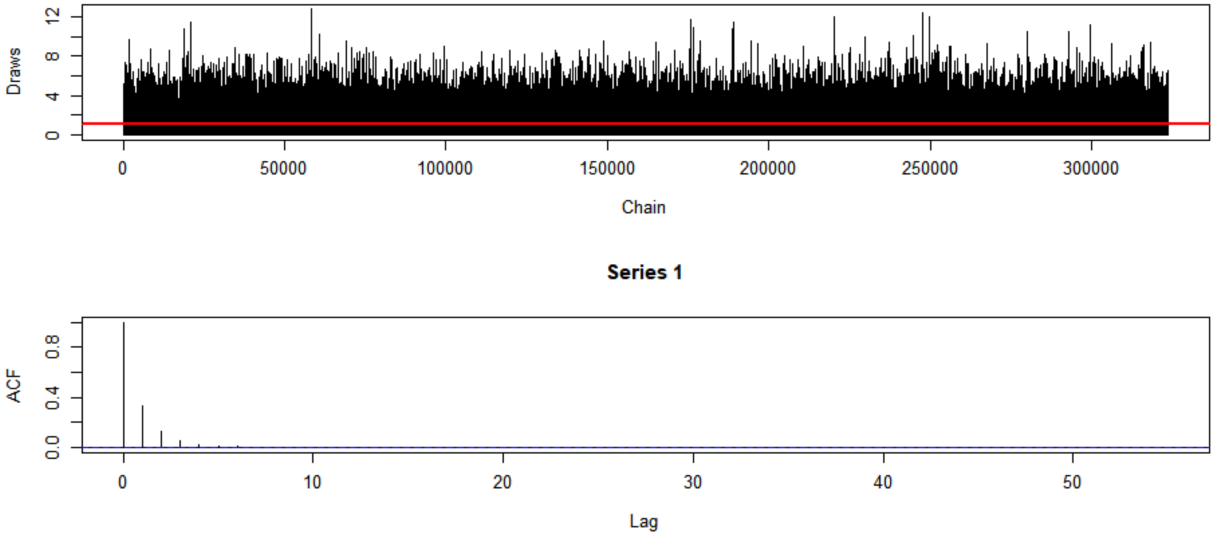
The function “estvssamp” plots the Monte Carlo estimates versus the sample size for a component of the MCMC output. This plot indicates whether the Monte Carlo estimate has stabilized:

Figure 2: “estvssamp” plots of the first chain



“ts” is used to represent a set of data arranged in chronological order, “acf” will show the autocorrelation function image:

Figure 3: Time series plot and autocorrelation function plot of the first chain



From the first figure, we can see that the empirical pdf fits very well, and the simulated histogram is very close to the EXP(1) distribution. In the second figure, the mean of the Markov chain stabilizes around 1.005 after 100,000 iterations. The time series plot in the third figure indicates that there is no apparent stagnation, indicating that the chain does not have periods of low acceptance rates. The ACF plot also shows that the autocorrelation coefficients of the chain are close to 0 after lag=5, indicating that the obtained samples are almost independent and identically distributed (i.i.d.).

Second chain:

Figure 4: Empirical pdf simulation of the second chain (π(x) = exp(-x))

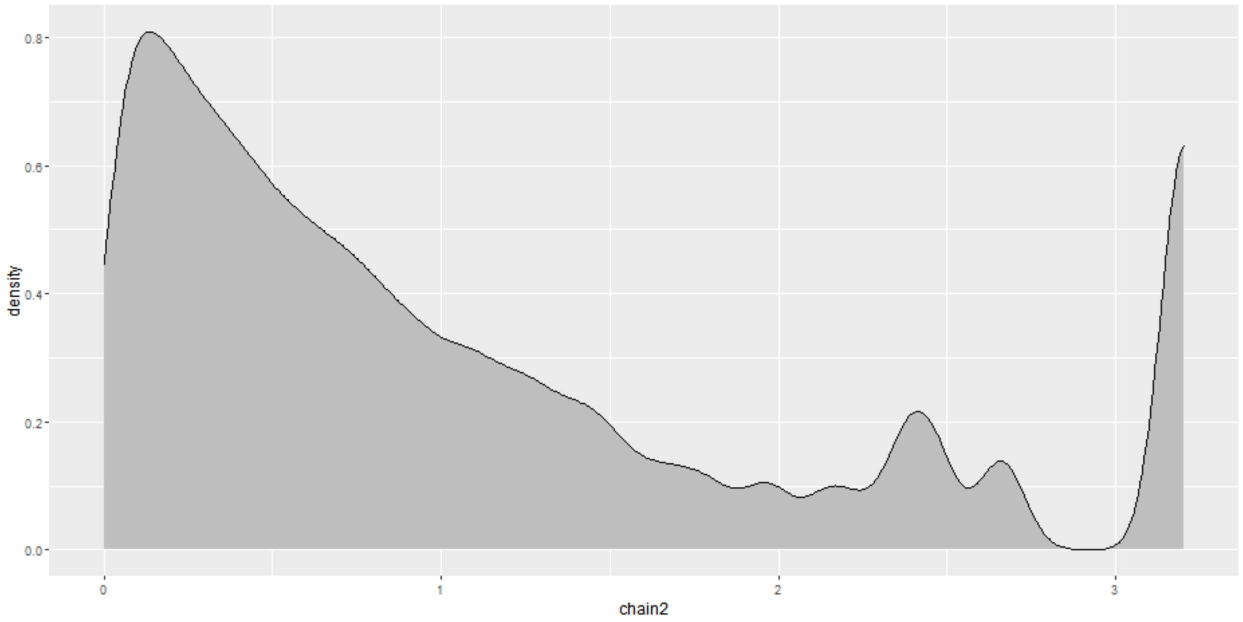


Figure 5: “estvssamp” plots of the second chain

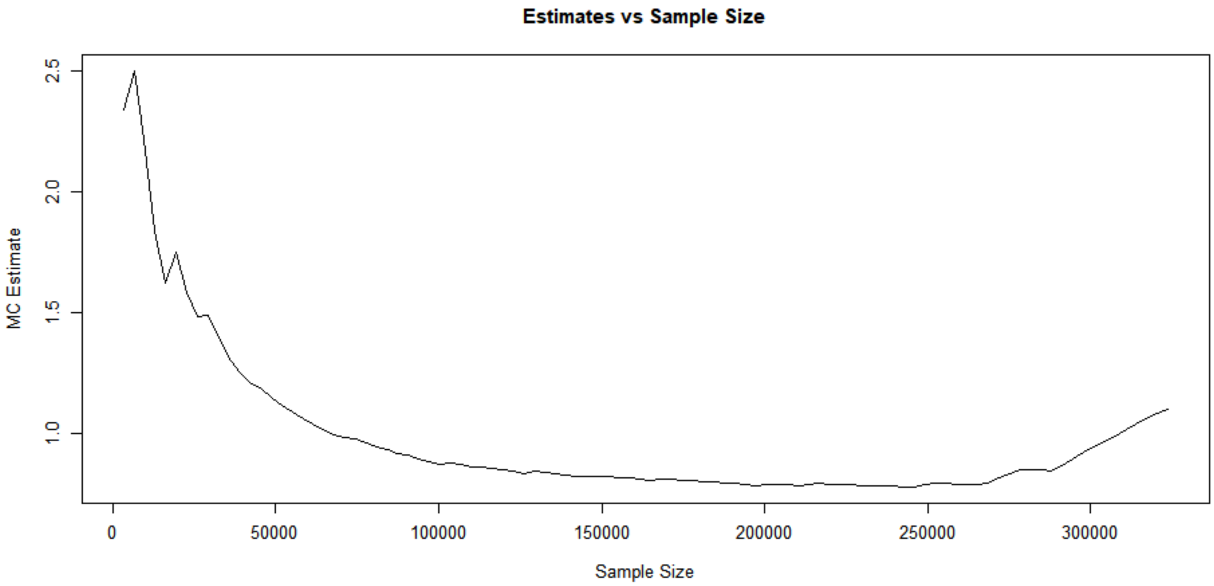
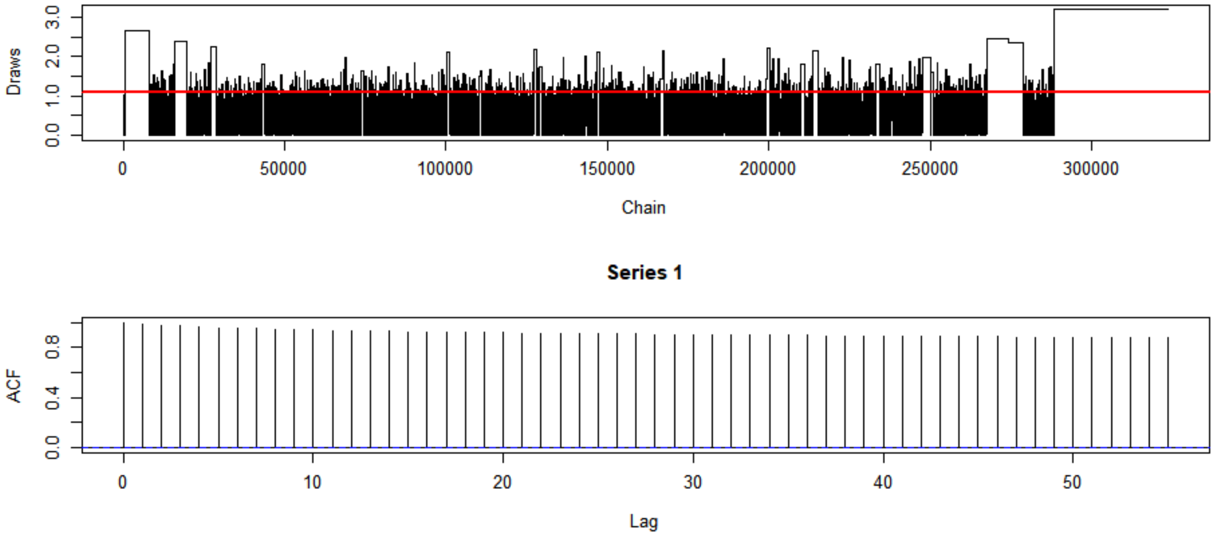


Figure 6: Time series plot and autocorrelation function plot of the second chain



From the fourth figure, we can see that the empirical pdf fits very poorly, and the simulated histogram is significantly different from the EXP(1) distribution. In fact, the simulated density even has high-density regions in the end part that violate the target function. We can see some clues from the fifth figure: the mean has a clear increasing trend before and after 300,000 iterations, the whole plot keeps changing substantially at the same time. In the time series plot of the sixth figure, we can see that the chain has many obvious stagnations, indicating that there are many periods of low acceptance rates. There are even periods of complete stagnation before and after 300,000 iterations. The ACF plot also shows that the autocorrelation coefficients of the chain are still large after lag=50, indicating that the obtained samples are not independent and identically distributed (i.i.d.).

In summary, we can conclude from the graphical analysis that the first chain has converged while the second chain has not yet converged.

B. ESS Results

Using the minESS function, **the minimum ESS of chain 1 and chain 2 is 153,658.4** for given dimensionality 1 of the parameter space, significance level alpha = 0.05, and the desired error margin epsilon = 0.05**.**

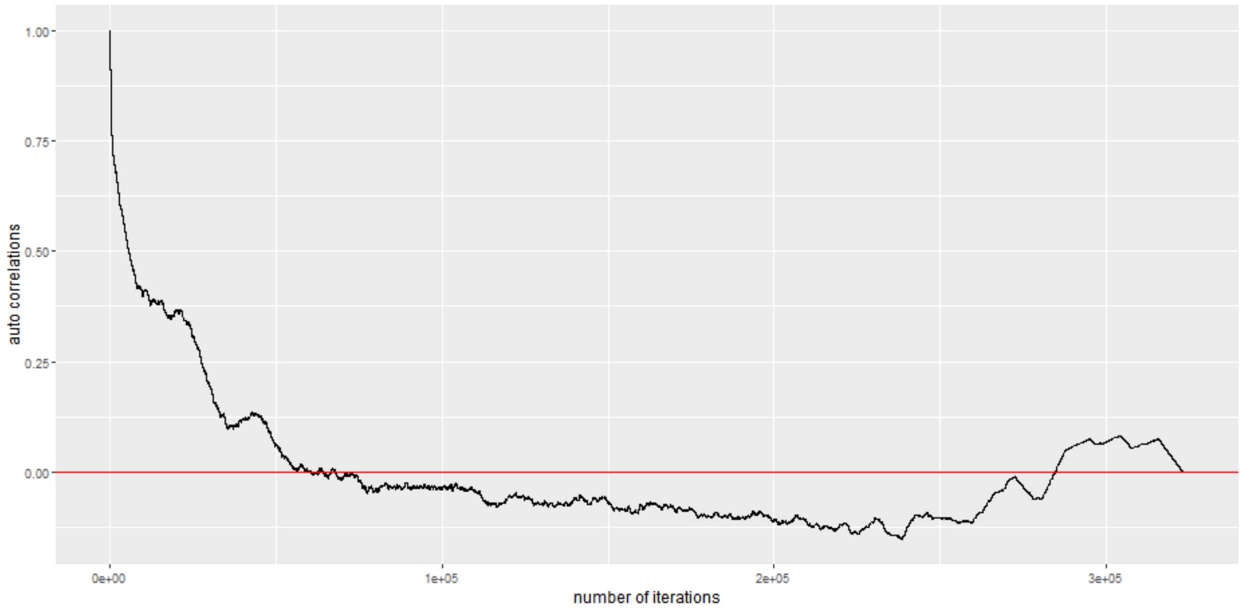
Table 2: Result of different chains using different ESS calculation methods

ESS Method	Calculation	Description	ESS of the first chain	ESS of the second chain
Default	M / (1 + 2 * sum(rho))	Basic definition	161,849.3	161,849
Coda	M*(lamda^2/sigma^2)	This method is the default in Coda	154,088.2	84.15389
Batch Means	M*(lamda^2/sigma^2)	Same as above but with local batch_means function	155,330.8	3,557.791
Regularized	sigma.sq = lambda.sq + 2 * sum(rho)		162,173	162,220.6

We marked the unreasonable results in red. It can be seen that the ESS values calculated by the four methods for the first Markov chain are consistent with the fact that the chain has converged, but the ESS values calculated by the first and fourth methods for the second chain do not match the actual situation. Why is that?

Due to the high autocorrelation of the second chain, which causes the ESS values calculated by the first and fourth methods to be inflated. This is also evident from the ACF plot, which shows that the ACF value of the second chain is still very high at lag = 50, indicating strong autocorrelation. When calculating the sum of autocorrelation coefficients, we are supposed to have obtained a large value, but in fact we obtained 0.5 as a result (323700/(1+2*0.5)=161,850) the result obtained by the first method), indicating a much lower autocorrelation. The plot below may provide some clues as to why this is the case.

Figure 7: Autocorrelation verses numbers of iterations of the second chain



We can see from figure above that the autocorrelation coefficients of the second chain remained high during the first tens of thousands of iterations, indicating poor convergence (high autocorrelation means that adjacent samples are highly correlated, indicating that the samples obtained are not i.i.d.). In fact, we can obtain a sum of 14,526.14 for the first 50,000 terms. However, as the number of iterations increases, the autocorrelation coefficients gradually change from positive to negative. Surprisingly, these negative autocorrelation coefficients cancel out the initial positive ones. The cumulative value of these coefficients at iteration 323,299 is 0.5, leading to an erroneous conclusion.

Therefore, these two methods may be unreliable in cases where the autocorrelation coefficient is too high.

C. Geweke Results

<u>The results of the Geweke:</u>

Firstly, we present the results of the toy function, and we found that in the case where we know the first chain has converged while the second chain has not yet converged, both methods for computing the variance give incorrect conclusions for the toy function:

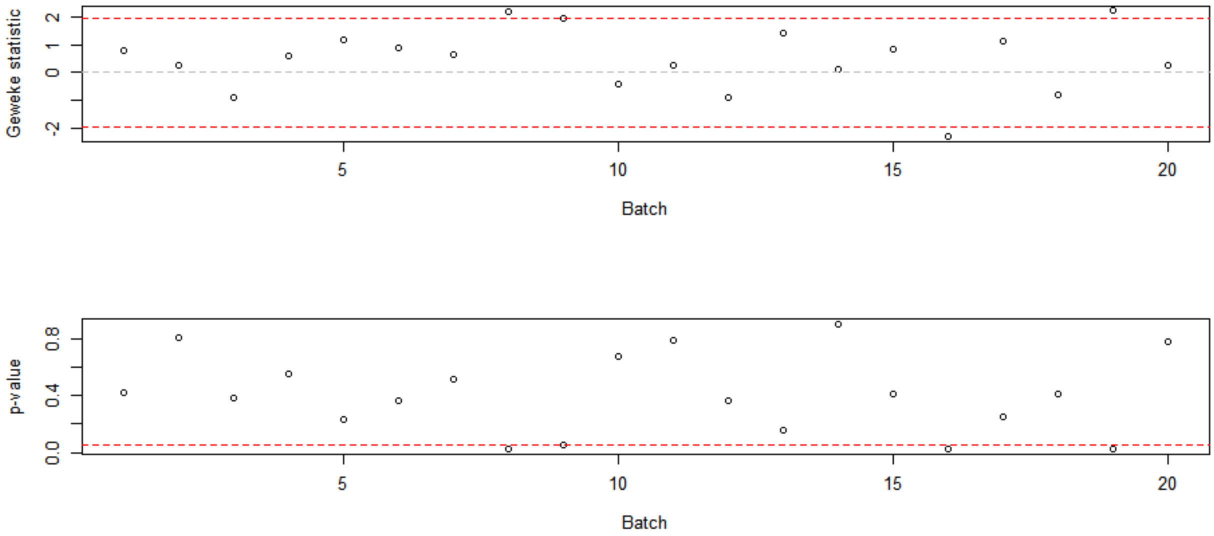
Table 3: Result using different variance calculation methods with the toy function

Geweke Method	Calculation	Description	P-value of the first chain	P-value of the second chain
Normal	M / (1 + 2 * sum(rho))	Calculate the variance directly using the var() function	0.338618	0.455918
Spectral	M*(lamda^2/sigma^2)	Calculate the variance using the spectral variance estimator	0.5085427	0.9845621

Now, we will demonstrate the improved Geweke function to see if it has improved. Of course, since the refined version returns results for multiple batches, the two functions cannot be directly compared side by side.

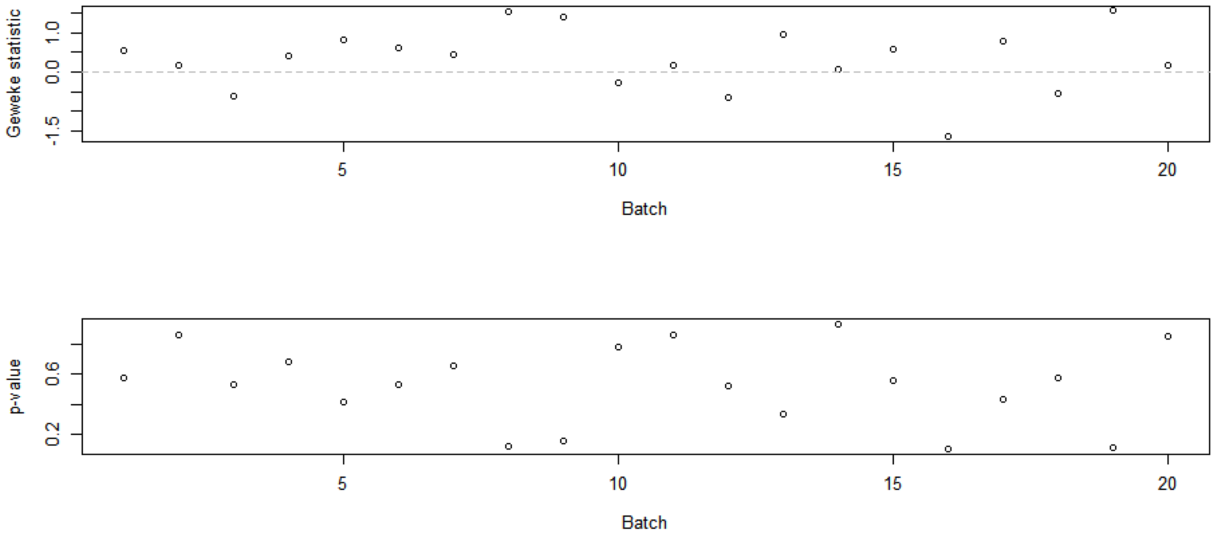
First chain:

Figure 8: Geweke Statistic and corresponding P-value using normal variance function of the first chain



Proportion of p_value < 0.05 = 20%

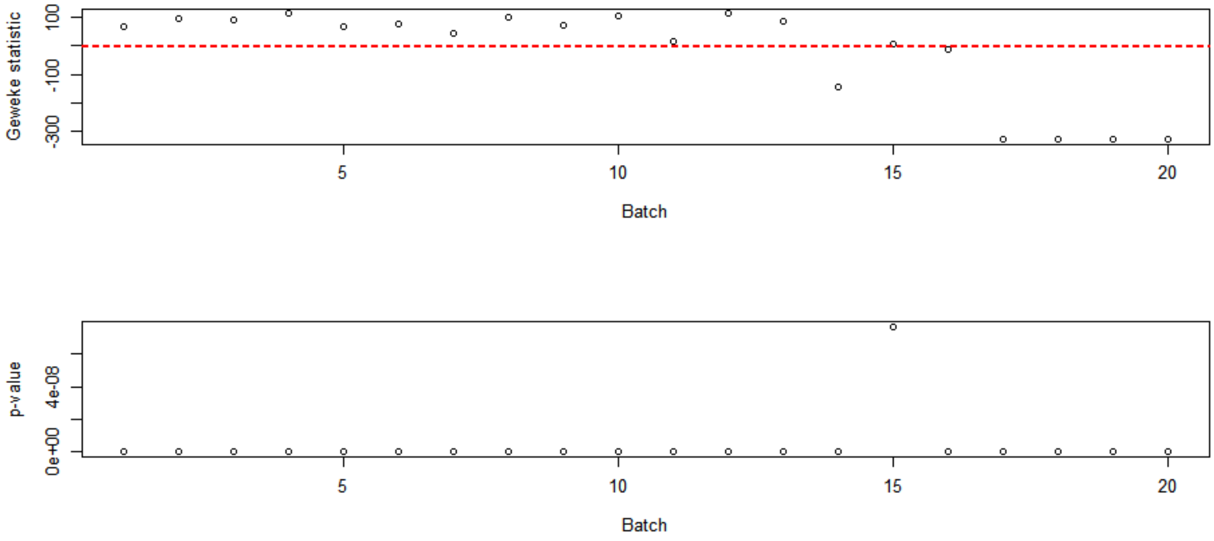
Figure 9: Geweke Statistic and corresponding P-value using spectral variance estimator of the first chain



Proportion of p_value < 0.05 = 0%

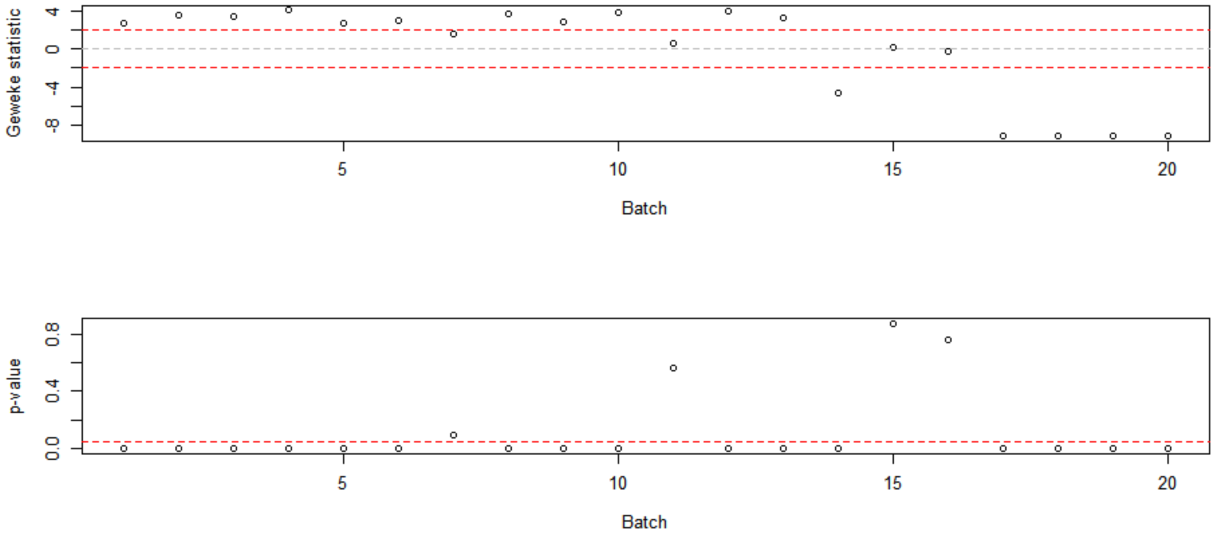
Second chain:

Figure 10: Geweke Statistic and corresponding P-value using normal variance function of the second chain



Proportion of p_value < 0.05 = 100%

Figure 11: Geweke Statistic and corresponding P-value using spectral variance estimator of the second chain



Proportion of p_value < 0.05 = 80%

For the first chain, both methods for computing the variance give results that are consistent with the truth, which is consistent with the conclusion mentioned above.

For the second chain, both methods give similar conclusions that the chain has not yet converged. From Figure 10 (using the var() function to estimate the variance), we can see that the p-values of 100% of the batches are "very" significant, with values much lower than 0.05. Figure 11 (using the spectral variance estimator to estimate the variance) shows that 80% of the batches have p-values lower than 0.05. We can essentially reject the null hypothesis of convergence.

From the above figures, it seems that using the var() function directly to compute the variance gives a "more accurate" conclusion, but is this really the case?

In fact, we can see from Figure 10 that the absolute values of most Geweke statistics are very large, to a degree that seems to be beyond common sense. Further investigation revealed that directly calculating the variance using the var() function results in a very small variance (in fact, the second chain did not exhibit a large variance in the mean trajectory plot), but using the Spectral Density Estimate at Zero Frequency to estimate the variance yields a very large estimate (sometimes greater than 1000).

Therefore, although the absolute values of the computed test statistics may all be greater than 1.96, the absolute value of the test statistic will be large due to the small variance (a part of the denominator) computed using the var() function, while the Spectral Density Estimator will be relatively small (but still greater than 1.96, and the two-sided p-value will be less than 0.05).

We need to investigate this issue further in future research. However, in terms of the conclusion of this research, our improved version of Geweke has better generality and more robustness compared to Geweke_toy.

D. Heidelberger and Welch Results

The results of the Heidelberger and Welch:

Table 3: Result using of the Heidelberger and Welch

Chain	Stationarity test	Starting point	P-value	Halfwidth test	Mean	Halfwidth
First chain	passed	1	0.641	passed	1.004	0.00498
Second chain	failed	NA	5.85e-06	NA	NA	NA

Firstly, based on the results, the running result of the Heidelberger and Welch function is consistent with the actual situation.

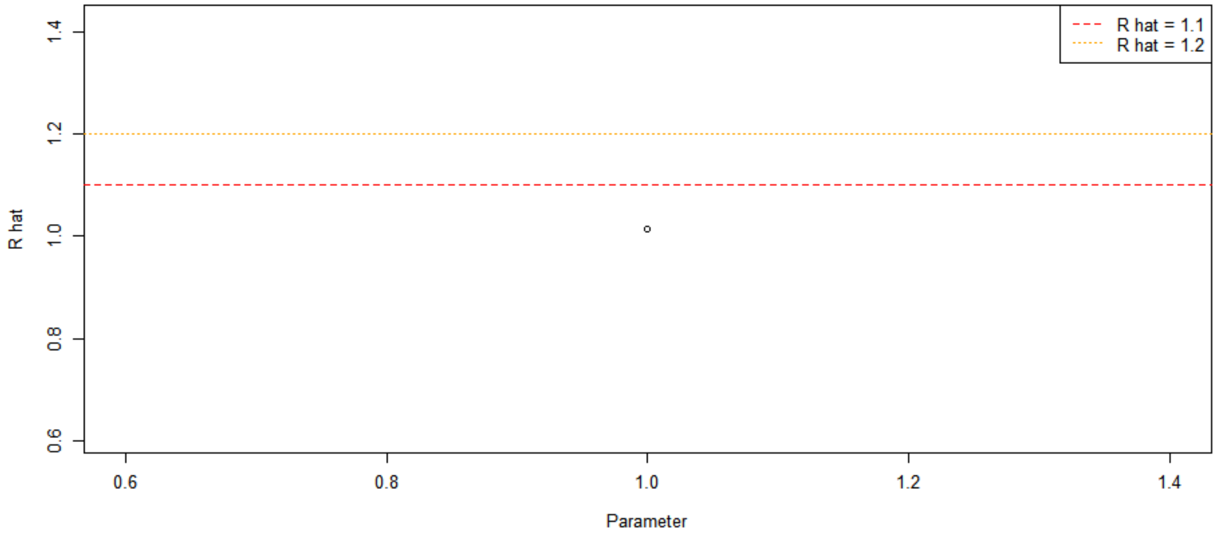
It was found that the first chain converged while the second chain did not converge. The starting point of the first chain is 1, which means that we can directly use the entire chain for testing to obtain a conclusion of convergence (without the need for burn-in, so that we can obtain more independent samples). Additionally, the first chain also passed the half-width test. In contrast, the second chain obtained a completely opposite conclusion.

Therefore, the performance of Heidelberger and Welch is good in our example, and there is no need for further improvement.

E. Gelman-Rubin Results

Firstly, we merged the first and second chains and ran the gelman_toy function, obtaining a Potential Scale Reduction Factor of 1.014203. We can also see the result from the following figure:

Figure 12: Result of the gelman_toy function



As we said earlier in the third chapter, there are two main problems with the toy model: the improved version of PSRF and different target distribution cases, so let's solve them.

Table 3: Result using of the Gelman-Rubin

Chain	Point est.	Upper C.I.
PSRF	1.065043	1.249431
MPSRF	NULL	NULL

From the table above, we can see that the improved PSRF has increased, indicating that the function takes into account the high autocorrelation of the second chain and provides some penalty for it. However, due to the sufficient convergence of the first chain, the PSRF is still below the threshold of 1.1. The improved function also provides an upper C.I. value that exceeds the threshold of 1.2, indicating that the improved function is more effective than the toy function since we know the truth that the second chain has not converged yet.

However, since this function can only provide a conclusion of rejecting convergence for all chains or not rejecting convergence for all chains, our two chains cannot perfectly demonstrate the value of Gelman-Rubin. Additionally, since both chains have the same target function, MPSRF does not exist. But we have used other examples to demonstrate the reliability of our function.

Meanwhile, as we described in Chapter 3, the initial values should be as dispersed from each other as possible so that the Markov chains can fully explore different parts of the distribution before they converge to the target.

VI. Acknowledgements

First of all, I would like to express my gratitude to Professor Guanyang Wang for giving me the opportunity to conduct this research and for introducing me to many relevant papers. Throughout the research process, Professor Wang provided me with continuous guidance and helped me deepen my understanding of MCMC convergence diagnostics.

Secondly, I would like to extend my gratitude to Professor Jack Mardekian for his detailed lectures on Bayesian analysis and MCMC convergence diagnostics. We had multiple discussions, and Professor Mardekian provided me with numerous references to authors who have made significant contributions to this field.

Finally, I would like to express my gratitude to the authors of the R packages “coda” and “mcmcse”, as well as the authors of the references cited in this research. Their unique perspectives have been crucial to this study, as they have made the convergence diagnostics of MCMC possible and introduced innovative techniques, such as spectral variance estimator, to improve upon existing diagnostic methods.

VII. References

[1]Convergence diagnostics for Markov chain Monte Carlo by Vivekananda Roy
SAS/STAT® 14.2 User’s Guide Introduction to Bayesian Analysis Procedures: Chapter 7
(Heidelberger and Welch 1981, 1983)
[2]BATCH MEANS AND SPECTRAL VARIANCE ESTIMATORS IN MCMC

(James M. Flegal and Galin L. Jones, 2010)

[3]Convergence diagnostics for Markov chain Monte Carlo

(Vivekananda Roy, 2011)

[4]The cutoff phenomenon in finite Markov chains

(Persi Diaconis, 1995)

VIII. Appendix

https://github.com/QiruPan/Convergence-diagnostics-for-MCMC