SimTools: Output Analysis for Monte Carlo

Siddharth Pathak

3/24/23

Table of contents

MENTORS

Evaluating mentor : Prof. Dootika Vats (dootika@iitk.ac.in)

Co-mentor : Prof. James Flegal (jflegal@ucr.edu)

1. Introduction

The term Markov chain(usually) refers to a discrete or continuous-time stochastic process on a general state space with the Markov property: the future is independent of the past given the present state. This follows one of the two conflicting standard usages of the term "Markov chain". Monte Carlo is a cute name for learning about probability models by simulating them, Monte Carlo being the location of a famous gambling casino. A half-century of use as a technical term in statistics, probability, and numerical analysis has drained the metaphor of its original cuteness. Everybody uses "Monte Carlo" as the only technical term describing this method. We can calculate probabilities and expectations by averaging over the simulations whenever we can simulate a random process. This means we can handle any calculation we might want to. We can always use brute force computation if we can't do pencil and paper calculations deriving closed-form expressions for the quantities we want. The Monte Carlo method may not be as elegant as the pencil and paper method, and it may not give as much insight into the problem. Still, it applies to any random process we can simulate, and we shall see that we affectate almost any random process. Pencil and paper methods are excellent when they work, but they only apply to a small set of simple, computationally convenient probability models. Monte Carlo massiveness huge increase in the models we can handle.

The general notion of MCMC is to estimate probabilities or expectations by simulating a Markov chain and averaging over the simulations. The probabilities or expectations calculated are those for functionals $g(Xi)$ of the stationary chain, hence they are probabilities or expectation concerning to the invariant distribution. Thus the first task in any MCMC application is to find a Markov chain having a specified consistent distribution.

So, after this introduction, one can understand the essence of the analysis of output from Monte Carlo algorithms, particularly (Markov chain Monte Carlo) MCMC, which requires a specialized toolkit, that must precisely done this task with awareness of co-relational behavior of variables and with decent level of accuracy . The analysis of the Monte Carlo estimates from the MCMC output requires the study of Monte Carlo standard error with the following central limit theorem (CLT)

if $\sigma^2 < \infty$, so that as $N \to \infty$

$$\sqrt{N}(\hat{\theta} - \theta) \xrightarrow{d} N(0, \sigma^2)$$

Whatever results our computation returns, it must be understandable based on available theoretical principles. I am aimed to complete or nearly finish development of the R package SimTools, which is written for the purpose of analyzing output of MCMC efficiently, and relatively better than all other tools available for this task.

2. Project Details

2.1. Background

Markov Chain Monte Carlo (MCMC) is a popular statistical method for estimating complex models and performing Bayesian inference. There are many simulation & output analysis tools available for MCMC, some of which are as follows:

Name of Tool	Introduction	Major Drawback
Stan	Stan is a powerful software package for Bayesian inference using MCMC. It can be used to fit a wide range of models, including hierarchical models, generalized linear models, and time series models.	Although Stan provides a wide range of MCMC algorithms and diagnostic tools, it may not be as flexible as other software packages for some types of models or analyses.Stan, for example, inherently assumes reversibility of the Markov chain, and is thus not equipped to handle general MCMC algorithms.
PyMC3	PyMC3 is a Python library for probabilistic programming that provides a range of MCMC algorithms for fitting Bayesian models.	PyMC3 may not be fully compatible with other Python libraries or modules, which may limit its usefulness for some researchers who rely on other tools for data analysis or visualization.
JAGS	JAGS (Just Another Gibbs Sampler) is a program for Bayesian inference using MCMC. It supports a wide range of models, including linear regression, logistic regression, and mixed-effects models.	JAGS does not support some advanced statistical techniques, such as Bayesian nonparametrics or machine learning methods, which may be required in certain applications. JAGS provides limited options for model checking and diagnosing convergence problems. This can make it difficult to identify problems with models, particularly when models are complex.

Name of Tool	Introduction	Major Drawback
BUGS	BUGS (Bayesian inference Using Gibbs Sampling) is a program for Bayesian inference using MCMC that supports a wide range of models, including linear regression, logistic regression, and hierarchical models.	BUGS can be slow when running large models or when models have complex data structures. This is because BUGS uses a Gibbs sampler to estimate model parameters, which can require many iterations to converge to a stable estimate.
RStan	RStan is an R package that provides an interface to the Stan software package. It allows users to fit Bayesian models using MCMC and provides a range of diagnostic tools for evaluating model convergence.	RStan can be computationally intensive and may require a large amount of resources, particularly for complex models or large datasets. This may limit its use for some researchers or applications. Debugging RStan code can be challenging, particularly when dealing with complex models or errors that arise during MCMC simulations.

These tools provide a range of MCMC algorithms and diagnostic tools to help users evaluate the performance of their models. They are widely used in the scientific community for fitting complex models and performing Bayesian inference. On the other hand, R packages like coda and bayesplot, serve a large audience of MCMC users, but are not up to speed with the latest theoretical developments in the area.

To equip the user with almost all type of output analysis of MCMC, SimTools is here. SimTools is an R package (not completed) that provides user enriched analysis of MCMC simulations (also iids), and also facilitates to generate Markov Chains.

SimTools provides support of two S3 Classes as following:

1. **Siid:** To Deal with iid Variables and their Simulations having functions boxplot.Siid(Box plots with simultaneous error bars around all quantiles for iid data), Plot.Siid(Density plots with simultaneous error bars around means and quantiles for iid data)

2. **Smcmc:** This class is for simulated data using Markov chain Monte Carlo, having functions addCI (Adds simultaneous confidence intervals for quantiles and means to an existing plot), boxCI (Adds simultaneous confidence intervals for quantiles to an existing box plot), getCI (Calculates simultaneous confidence intervals for means and quantiles as indicated for the desired MCMC output), plot.Smcmc (Density plots with simultaneous error bars around means and quantiles for MCMC data. The error bars account for the correlated nature of the process)

2.2. Goals

There are five major goals that I will do to complete the SimTools package before its ready for a CRAN submissions:

- **Functions in both Siid, and Smcmc are compatible to make density plots of chains having continuous state space. I will make a separate option in plot.Smcmc and plot.Siid i.e. **Discrete** , it is a Boolean value type argument, by default it FALSE but if user give its value as TRUE , the Given chains in input consider to have Discrete State Space and plot the Approximated Mass function of them in different way(that Continuous State Space Chains).
- ***Equip the SimTools package with advance multiple chain compatibility, so that the chains can be combined to obtain better estimates of statistical parameters, such as means and variances, and to assess the uncertainty of the estimates. This includes some change in the current plot.Smcmc, plot.Siid and acf function.
- *Implement a Summary functions for both Siid, and Smcmc functionmc classes, this function will take multiple chains as input and gives information about mean estimates, quantiles, standard errors, effective sample sizes, and suggestions on accuracy, about these given chains.
- **In MCMC as the simulation size increases, the autocorrelation between samples typically decreases, resulting in a more dispersed trace plot. So, I will implement a different Trace.Plt(), who shows a subset a the full trace plot, we will also gives choice to user to input the range within which he need to see trace plot.
- ***As the C++ is a compiled language, which means that the source code is translated into machine code by a compiler before it is executed. This machine code is specific to the program's hardware, which can lead to very efficient code. So, we will try to make Most of the calculation part via RCpp, in which our primary focus is onto improve confidence interval calculation efficiency.
- *Drafting the rich documentation and vignette for whole Package SimTools.

As per my understanding , I have marked stars to rate difficulty of the goals (more the stars, more difficult (It includes amount of time I need to do in that task) goal).

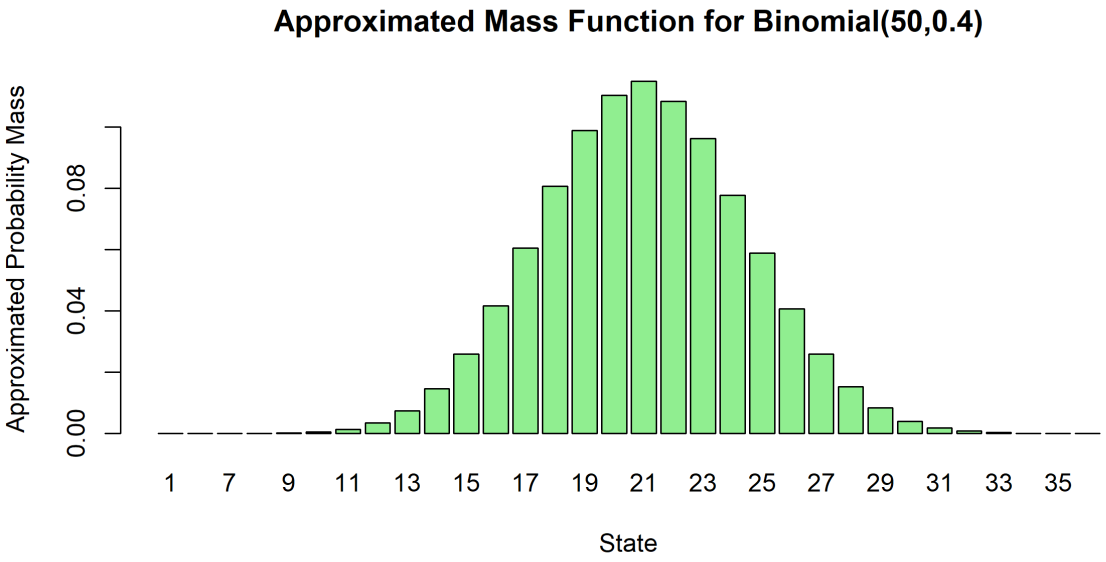
2.3. Implementation Plan

2.3.1. Discrete State Space Compatibility

I shall start the work on this during the community bonding period. This will allow me to help finish the first goal (i.e. to change the functions plot.Siid, plot. Smcmc so that it can also plot the density function of discrete state space Markov Chain) by the time the community

bonding period ends. Finally, I need to implement tests with many self implanted Chains and some commonly available data sets that can work as input. I study how similar tasks in other libraries like RStan and Bayesplot, so I am comfortable with this task. To check the test result we will compare the test results with the result obtained from different platforms (I mentioned above) available to do the same task. A small Example of the idea is shown below(This is not the implementation code, that will be much different from it and much more complicated):

```
k = dbinom(0:50, size = 50, prob = 0.4)
# Define the transition matrix of the Markov chain
P <- matrix( rep(k,51), nrow = 51, byrow = TRUE)
# Define the number of steps to simulate
n <- 1000000
# Simulate the Markov chain
states <- vector(length = n)
states[1] <- 1
for (i in 2:n) {
  k = ceiling(states[i-1]/2)
  states[i] <- sample(seq(1,51,1), size = 1, prob = P[k,])
}
# Compute the approximate mass function
pmf_approx <- table(states)/n
# Plot will be expected as every bar has same height
#(see Transition Matrix I declare ) Creating the bar plot
barplot(pmf_approx, xlab = "State",
        ylab = "Approximated Probability Mass",
        main = "Approximated Mass Function for Binomial(50,0.4)",
        col = "lightgreen")
```



I am also planning to use some techniques used in bayesplot for plotting data because the some themes match our SimTool package.

2.3.2. Multiple Chain Compatibility

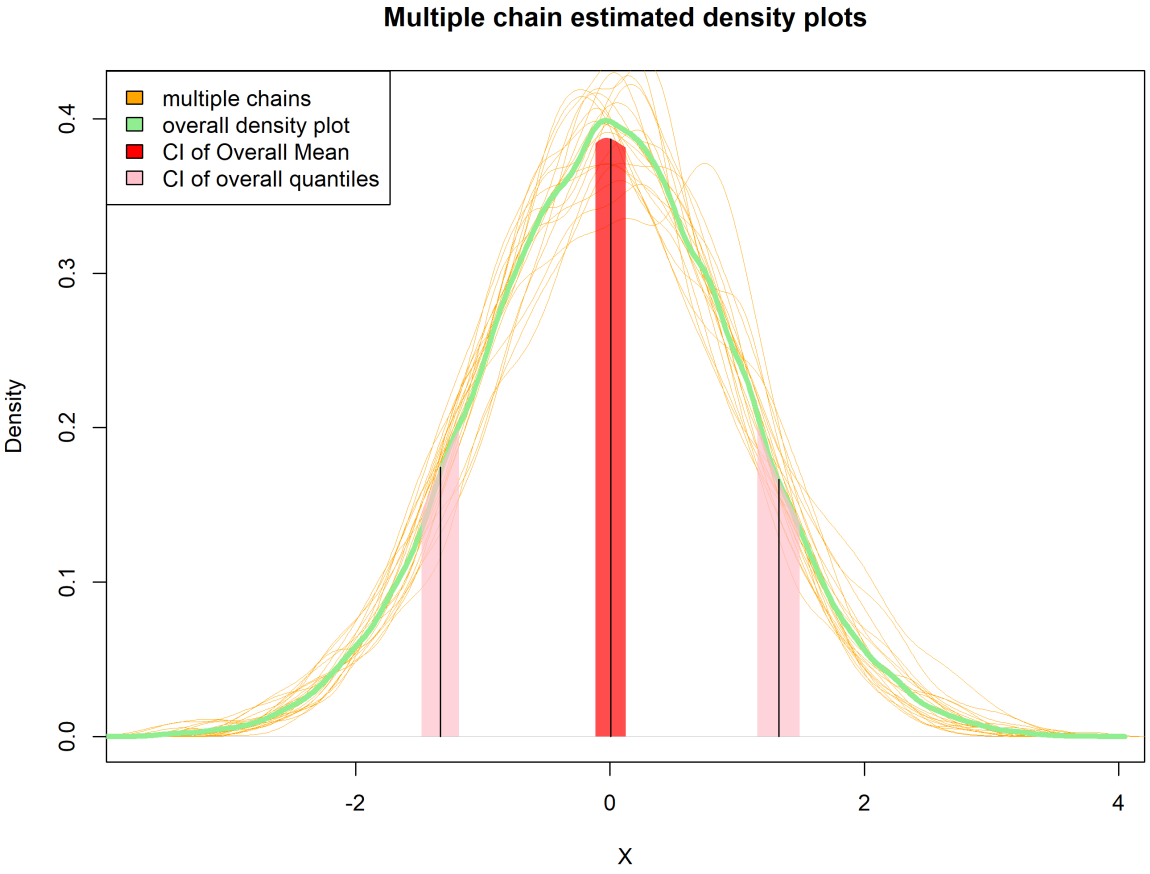
In MCMC, multiple chains are often used to explore the target distribution in different regions of the parameter space simultaneously. If the chains are not compatible, it means that they do not explore the parameter space in a way that is consistent with the target distribution. This can lead to issues such as slow convergence, bias in the estimates of the parameters, and difficulty in assessing the convergence of the chains.

By ensuring that the multiple chains are compatible, it is possible to increase the efficiency and accuracy of the MCMC algorithm. This can be achieved through a process called "burn-in" where the chains are run for a certain number of iterations to ensure they have explored the parameter space sufficiently before their samples are combined. The compatibility of the chains can also be assessed by measures such as the Gelman-Rubin diagnostic, which compares the variance within the chains to the variance between the chains.

For the next two weeks, I will work to implement multiple-chain compatibility in this package. It's mean, plot.Smcmc and plot.Siid function can take multiple chains as input and can plot estimated PDF/PMF, for all the given chains in the same plot.**Since, SimTools have a decent level of multiple chain compatibility but, the plots are not that much interactive and requires a lot of correction in terms of level of utility. I am thinking to implement the plots from bayesplot library from R to make the plots much more cleaner and informative, for this it took an effort to learn about internal structure of Bayesplot and its classes.** I will keep an option of "Isolated" which is by default FALSE, but if user sets it as accurate in the function call, the function will print all the plots separately. An example of idea how I would implement to plot multiple chain estimated density in a Single Plot is as follows(This is not the implementation code, that will be much different from it and much more complicated):

```
library(SimTools)
MakeChain <- function(p = 4, n = 1e3, h = .5)
{chain <- matrix(0, nrow = n,ncol = p)
 for (i in 2:n)
  {prop <- chain[i-1, ] + rnorm(p, mean = 0, sd = h)
   log.ratio <- sum(dnorm(prop, log = TRUE) - dnorm(chain[i-1, ], log = TRUE))
   if(log(runif(1)) < log.ratio)
    {chain[i, ] <- prop}
   else
    {chain[i, ] <- chain[i - 1, ]}
   }
```

```
  colnames(chain) <- c("Comp 1", "Comp 2", "Comp 3", "Comp 4")
  return(chain)
}
chain1 <- MakeChain()
plot(density(chain1),main="Multiple chain estimated density plots",
     xlab = "X",ylab = "Density", lwd = 0.5, col = "orange")
chain = chain1
for(i in 1:20)
{
 chain1 <- MakeChain()
 chain = rbind(chain,chain1)
 lines(density(chain1), lwd = 0.5, col = "orange")
}
lines(density(chain),lwd = 4, col = "lightgreen")
addCI(as.Smcmc(chain),getCI(as.Smcmc(chain)),component = 1,
      mean.color = 'red',quan.color = 'pink')
legend("topleft",fill = c("orange","lightgreen","red","pink"),
       legend = c("multiple chains","overall density plot",
"CI of Overall Mean", "CI of overall quantiles"))
```



Confidence Interval calculation will be implemented after discussing the theoretically with the mentors during the coding period of the GSoC-2023

I will need to try something more to make the plot more readable and need to work on improving the clarity of the plot as the CIs are overlapping and making the plot difficult to readable. This problem needs some theoretical encounter of statistical concepts that I will do in the starting of the project.

2.3.3. Summary() Function

After Multiple Chain compatibility my Focus will be on Summary() Function, for 2 weeks. This function will take multiple chains as input for both Siid and Smcmc class. I will try to write it in such a way so that it will return following information:

1. **Name of Class**
2. **Mean**
3. **Standard error**
4. **median**
5. **Variance/SD**
6. **Quantiles (0.25 & 0.75)**
7. **Effective Sample Sizes**
8. **Suggestions on Accuracy**

On the last two things I will work during the project. Although to write the accurate and perfect Summary() will take 1 week for me, but I can show some implementation how I will write the summary function(This is not the implementation code, that will be much different from it and much more complicated):

```
Summary <- function (object)
{
  object.class <- class(object)
  Smcmc_output <- object$chains[[1]]
  columns = ncol(Smcmc_output)
  stacked = object$stacked
  b.size = object$b.size
  mean = vector(length = columns )
  median = vector(length = columns )
  variance = vector(length = columns )
  range = matrix(0,ncol = 2, nrow = columns)
```

```
    for (i in 1:columns)
    {
      mean[i] = mean(Smcmc_output[,i])
      median[i] = median(Smcmc_output[,i])
      variance[i] = var(Smcmc_output[,i])
      range[i, ] = range(Smcmc_output[,i])
    }
    npar <- colnames(Smcmc_output)
    summary_list <- list(class = object.class,
                         b.size = b.size,
                         samples = nrow(Smcmc_output),
                         range = range,
                         mean = mean,
                         median = median,
                         Variance = variance
                         )
    return(summary_list)
  }
  chain = MakeChain()
  out.one <- Smcmc(chain)
  ## values are in order col1, col2, col3, col4
  Summary(out.one)
```

```
$class
[1] "Smcmc"

$b.size
[1] 56

$samples
[1] 1000

$range
          [,1]     [,2]
[1,] -2.605717 3.057877
[2,] -2.591977 3.709921
[3,] -2.983751 3.070938
[4,] -2.493753 2.705188

$mean
[1]  0.05908121 -0.32986807 -0.22673603  0.02575697
```

```
$median
[1]  0.17971679 -0.29866091 -0.26663161  0.03441095

$Variance
[1] 0.9729992 1.0889762 0.9074441 0.9354703
```

Currently, there are no other package that calculate multivariate effective sample size (except mcmcse) of the Markov chain. There are a few other packages in R that do uni variate effective sample size calculations, the most popular of which is coda (Plummer et al. (2008)) . coda uses inconsistent parametric estimators in comparison to theoretically consistent estimators in mcmcse. Adequate Sample Size & Suggestions on Accuracy needs some discussion with mentors. So, I did not have much idea about that. This will need to change. **This functions return a "List" as output. During the implementation of the project, I will make the returning value more structural and similar to usual summary functions in R. Contents will print in the form of a table or in some other structured way but these things will be decided and implement after they are discussed with mentors.**

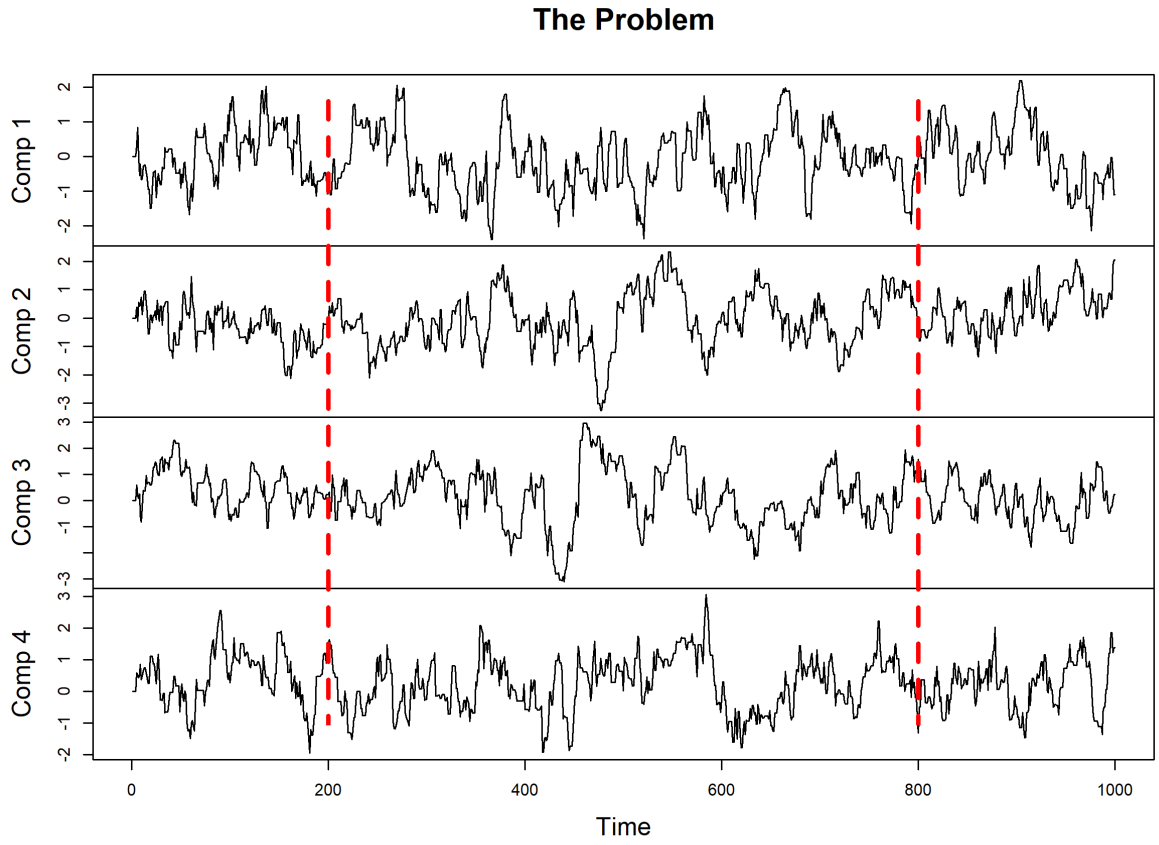
2.3.4. Efficient Trace Plot

A trace plot is a graph that shows the progression of a parameter or variable over time or across iterations in a statistical model. It is an important tool for analyzing the behavior of a model and can provide insights into its performance and behavior. Here are a few reasons why trace plots are important:

1. **Identifying convergence**
2. **Diagnosing problems**
3. **Assessing model fit**
4. **Visualizing uncertainty**

In the next two weeks we need to make an efficient trace plot function, that shows the most significant part or User wanted part of the trace plot. As the present plot.ts() function is not able to do this. we will need to write a separate functions for this task. I am representing the problem and the expected solution in the following plots.

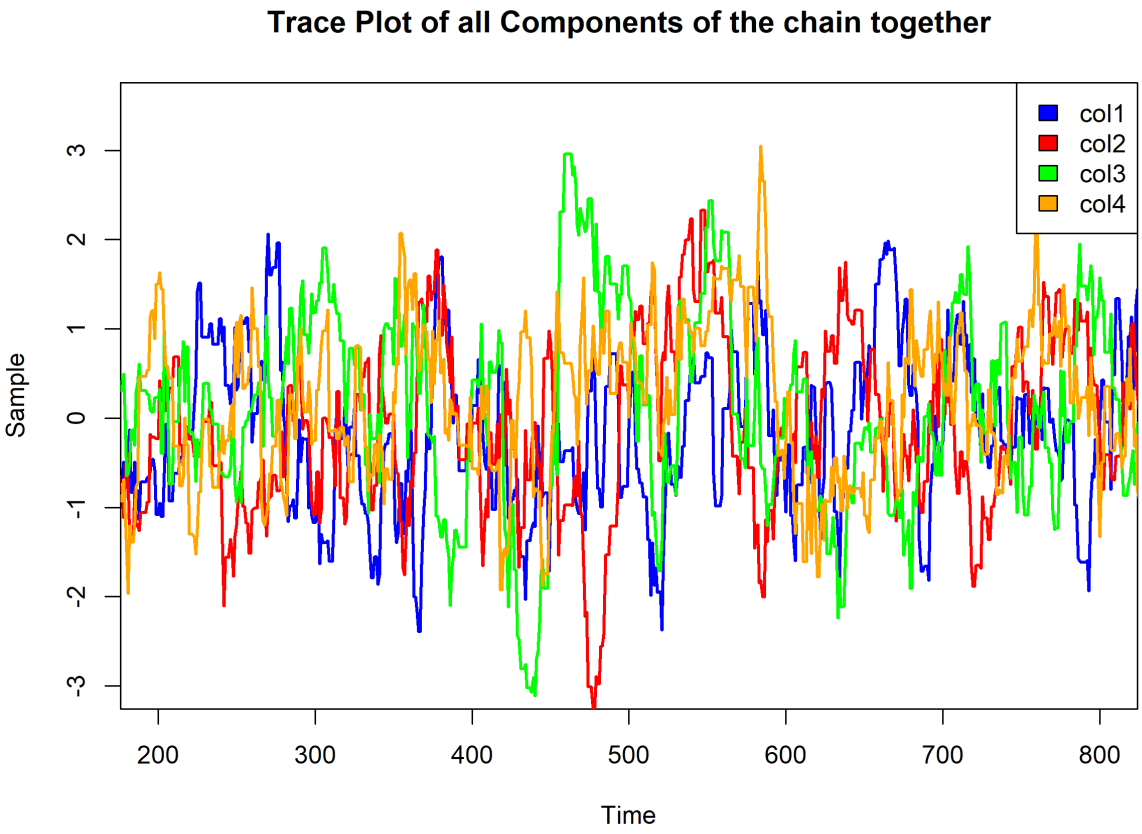
```
chain = MakeChain()
t = ts(chain)
plot.ts(t,xlim = c(200,800),main="The Problem")
abline(v = c(200, 800), col="red", lwd=3, lty=2)
```



We can do this by make some appropriate changes in plot.ts() function. The work will not be done by adding some amount of code chunk, it will require more things to learn. Moreover, which part of the plots function will show by default , will be decide by mentors. It include the application of some theoretical concepts of Statistics which will be first discussed with mentors and then implement it.

I am presenting an expected solution of this problem. I must work on this more during the coding period to implement this successfully. This is not the implementation code(just to give glimpse), that will be much different from it and much more complicated:

```
### Problem can be solved the following implementation
x = 1:1000
plot(x = x,y = chain[,1],xlim = c(200,800),ylim = c(-3,3.5),"l",col ="blue",
main="Trace Plot of all Components of the chain together",
ylab = "Sample",xlab = "Time",lwd = 2)
lines(x = x,y = chain[,2],xlim = c(200,800),"l",lwd =2,col = "red")
lines(x = x,y = chain[,3],xlim = c(200,800),"l",col = "green",lwd =2)
lines(x = x,y = chain[,4],xlim = c(200,800),"l",col = "orange",lwd =2)
legend("topright",fill = c("blue","red","green","orange"),
       legend = c("col1","col2","col3","col4"))
```



2.3.5. Documentation

Documentation is one of the most important aspects of good code. Without it, users won't know how to use your package, and are unlikely to do so. Documentation is also useful for us in the future (so you remember what the heck you were thinking!), and for other developers working on your package. I am planning to use the package roxygen2 to maintain and update the documentation.

The premise of roxygen2 is simple: describe your functions in comments next to their definitions and roxygen2 will process your source code and comments to automatically generate .Rd files in man/, NAMESPACE, and, if needed, the Collate field in DESCRIPTION. I will start the documentation work in 9th week. I will make useful documents, vignette for the users that will include, all the relevant information about the respective function form the perspective of users and developers. It will take one or one and a half week. The following functions need to be documented(first time/ again):

1. **plot.ts.SimTool()**
2. **plot.Smcmc()**
3. **plot.Siid()**
4. **getCI()**
5. **add.CI()**

After ending the coding work(almost), the package requires roxygen comments on .R/.cpp files of these functions.

A demo for the package will serve to show users how to perform output analysis for MCMC by weaving together the functions of the package. I propose to create a demo that will help the existing vignette of the package and assist the users uninitiated with the theory and common practices in output analysis for MCMC. I plan to setup the code to be able to identify the demo created for the package and use the demo function in the to call the demonstration R scripts.

2.3.6. Export to C++

If the Time allows(I got only 1-1.5 week remain), I will export calculative function in C++. C++ is the language in which I am most proficient. I used to do all coding contests with C++, I have learned to calculate confidence interval in my academic curriculum, so I am implementing an example from there to show how can we implement CI calculations in C++.

I have learned to produce confidence interval by t-distribution method. The algorithm for the same is as following(This is not the implementation code, that will be much different from it and more complicated):

2.3.6.1. Algorithm:

- **Define the sample size n and the sample mean x.**
- **Calculate the sample standard deviation s .**
- **Calculate the t-value, where $t\alpha/2$ is the $(1-\alpha/2)th$ percentile of the $t-distribution$ with $n-degrees\ of\ freedom$.**
- **Calculate the standard error of the mean $se = s/\sqrt{n}$.**
- **Calculate the lower and upper bounds of the confidence interval using the formula:**

lower bound $= x - (t\ \alpha/2\ )/\ se$

upper bound $= x + (t\ \alpha/2\ )/\ se$

- **The confidence interval using the t-distribution method is [lower bound, upper bound].**

```
#include <Rcpp.h>
using namespace Rcpp;
// [[Rcpp::export]]
NumericVector gen_ci(NumericVector x, double alpha) {
```

```
  int n = x.size();
  double mean = mean(x);
  double sd = sd(x);
  double t = qt(1 - alpha/2, n - 1);
  double se = sd/sqrt(n);
  double lower = mean - t * se;
  double upper = mean + t * se;

  NumericVector result(2);
  result[0] = lower;
  result[1] = upper;
  return result;
}
```

The function "gen-ci takes two arguments: x (a NumericVector of sample observations i.e. markov chain) and α(the significance level). The function first calculates the sample mean *mean* and the sample standard deviation *sd* using the *mean* and *sd* functions in Rcpp. It then calculates the t-value t using the *qt* function, which returns the $(1-\alpha/2)$th percentile of the **t-distribution** with n-1 degrees of freedom. The function then calculates the standard error of the mean *se* using the formula $se = sd/\sqrt{n}$. Finally, the function calculates the lower and upper bounds of the confidence interval using the formula described in the algorithm and returns a Numeric Vector containing the lower and upper bounds of the confidence interval.

2.4. Users Targeted

This project shall affect both beginners and well experienced users. More examples on how to write a complicated simulation and analysis methods will help beginners and they will be able to directly use the models in their projects. Being able to import simulated chains from other libraries will help the experienced users perform transfer learning on their Simulated Markov Chains.

3. Timeline

S.No.	Dates	Planned Work
3.1.	**Community Bonding Period (May 4 - May 28)**	Get myself familiar the code base of SimTools. This will allow me to implement more efficient and same theme codes. Also read the papers/Notes mentioned in the references section. Work on the Discrete State Space Compatibility. Try to finish this work in this period.

S.No.	Dates	Planned Work
3.2.	**First Week (May 29 - June 4)**	Read about the available documentation of acf, plot.Siid, plot.Smcmc functions. Understand how much they are compatible with multiple chains and think of scenarios where users find it difficult to use them. Also, Find ways to make the plot more interactive and discuss the way to make clear CI presentation in the plot with so many chains.
3.3.	**Second Week (June 5 - June 11)**	Warm up in the R script files to implement the accurate functions(acf, plot.Siid(), plot.Smcmc) to plot the multiple chains in the same plot by default , but can also plot them in different plots if user wants. Also this includes the testing of these functions on different simulated chains.
3.4.	**Third Week (June 12 - June 18)**	My aim this week to discuss the furthur improvement that can be made in the functions. Finish the work of Multiple Chain Compatibility, and import them into the package by allign them with syntax of S3 classes used in the package.
3.5.	**Fourth Week (June 19 - June 25)**	Read significantly about how classes works with summary and plot function in R (In Application Process Period). So, to start the task to implement the Summary() function, I will first read and discuss 2 results (Effective Sample Size, Standard Errors & Suggestions on Accuracy) that are returned by this function with mentors. The reason is that I did not have sufficient knowledge for these three parameters return by function.
3.6.	**Fifth Week (June 26 - July 2)**	Again warm up on R script files,and implement decent Summary() functions for our simulated Markov Chains. It can take multiple chains as input and return the essential statistics about them.
3.7.	**Sixth Week (July 3 - July 9)**	Finish the work of the Summary() Function. Discuss the improvement of the written function in the previous week with the mentor, and implement it. Finalize the structure and export it in the package.
3.8.	**Seventh Week (July 10 - July 16)**	Revisit to the internal code for plot.ts() (as I already done it in Application Period) function in R. Discuss the statistics(if any) need to apply to find the subset of time over which we show the trace plot by default.

S.No.	Dates	Planned Work
3.9.	**Eighth Week (July 17 - July 23)**	Done the Implementation of plot.ts() . test it for different simulated chains, and available datasets. Discuss the furthur improvements with mentors and fix it in the package appropriately.
3.10.	**Ninth Week (July 24 - July 30)**	Finish the work of trace plot (if it didn't end in the previous week), Also at this pint of time , I will start the Documentation whole the whole package, update the readme file in github page of the package, write "Vignette" for the package.
3.11.	**Tenth Week (July 31 - August 6)**	Finish the remaining work of Documentation of the SimTools package. Start to learn how getCI() function work in this library (As there are many methods to get confidence Interval). Try to implement the exact method in C++ via RCpp.
3.12.	**Eleventh Week (August 7 - August 13)**	Finish the code to calculate the CI with appropriate method in RCpp. Discuss the improvement with the mentor. Test the getCI() on different chains and on different Datasets.
3.13.	**Twelfth Week (August 14 - August 21)**	Export the getCI() function written in RCpp in the package. Check the functioning of library(package) as whole. Fix the bugs if found. Submit the project for Endterm Evaluation.

NOTE: There can be some minor changes in the timeline during the real implementation of the respective task, but I will try my best to keep in tone with this timeline.

4. About Me

4.1. Academic Experience

I am currently enrolled in the B.S. Undergraduate Programme of Indian Institute of Technology, Kanpur(India) in Statistics & Data Science. Currently I am doing my Academic work in Statistical Computing, Data Science, Theory of Statistics and Programming Algorithms.

4.2. Coding Experience

I have coding since my 11th grade. After having explored domains like competitive programming and web development basics I decided to stick to the domain of machine learning because it is my academically choose field. I have strong coding skills in HTML,CSS, Python, C, R, C++

(in order of proficiency). I have been actively studying about Statistical Computing, Stochastic Processes and Markov Chains and Optimization algorithms for the last 4 months. My primary interest is in Probabilistic Algorithms.

4.3. Mathematical and Scientific Background

Currently doing a course on Statistical Computing as a part of my College Course curriculum. Also Pursuing online courses in the field of Supervised machine learning. I currently studying theoretical Statistical concepts in the course Theory of Statistics. Moreover, have decent knowledge about Object Oriented Programming .

4.4. Why choose me?

I am quite proficient in R. Over the past few months I have explored the usage of several R libraries and even used quite a few of them myself. I am also enthusiastic about reading new researches in Statistics and Machine Learning and try regularly reading them to keep myself updated about this field. I am quite familiar with Git and Github.

4.5. Why contribute to R?

I have written code in R over the last 7-8 months. It has hence become clear to me how powerful this language it. As a result, I always prefer to use R for any project involving intense numerical computation. MCMC is one such field; more people are choosing R for their Statistical Tasks. Their job will become far more manageable if there are some excellent tutorials and examples they can refer to(in the Documentation or Vignette). Also by being a part of this project I also be able to contribute to this community. Also, I want to work with Statistician to learn how they see, observe, imagine and explore different phenomenon in this field.

4.6. Contact Information

- Github: Siddharth Pathak
- Email ID: siddharthp21@iitk.ac.in , siddharthpathak102002@gmail.com
- Phone No: (+91) 798-505-1222
- LinkedIn: Siddharth Pathak

5. Schedule Conflicts

I have been admitted into the BS Statistics and Data Science program in Indian Institute of Technology, Kanpur. Although I will have regular semester starting in July 31 (Will still work and communicate (except class time) as per my availability), I will be able to work on finishing the project in this time(till week 12), and if suitable for the mentors, can compensate for more hours before July 31st.

References

[1] Lecture notes on MCMC by Cornell University.

[2] Markov Chain Monte Carlo Lecture Notes Charles J. Geyer Copyright 1998, 2005 by Charles J. Geyer

[3] Classes in R Programming

[4] First Course in Stochastic models by Henk C. Tijms

[5] Modeling High-Dimensional Time Series

[6] CRAN - Package mcmcse (r-project.org)

[7] CRAN - Package Rstan (r-project.org)

[8] CRAN - Package bayesplot (r-project.org)