Bayesian Methods for Data Science Using R

Christina Knudson, Ph.D

University of St. Thomas

Symposium on Data Science and Statistics 2020

Overview

Basic Bayesian Steps

1. Design experiment/study and collect data
2. Select a model and priors
3. Approximate the posterior via Markov chain Monte Carlo
4. Assess the posterior approximation
5. Use the MCMC samples to conduct inference

May need to iterate between steps 3 and 4.

Today: R packages for steps 3, 4, and 5.

Why R? Accessibility: R, RStudio, and all R packages on CRAN are free.

Step 3: Approximating the Posterior via MCMC

More Common Models

- MCMCpack
- brms, Rstan (interfacing with Stan)
- rjags, runjags, R2jags (interfacing with JAGS)
- R2Winbugs, BRugs (interfacing with BUGS)

More Customized Models

- mcmc
- BayesianTools

Step 3: Approximating the Posterior via MCMC

Baby linear regression using the MCMCpack package:

```
#fit a regression model
freq.fit  <- lm(riders_registered ~ temp_feel, data=bikes)

bayes.fit <- MCMCregress(riders_registered~temp_feel,
                         b0=freq.fit$coefficients,B0=.01,
                         c0=.1,d0=.1,
                         beta.start=freq.fit$coefficients,
                         data=bikes,
                         burnin=10^3,mcmc=10^5)
```

Step 3: Approximating the Posterior via MCMC

Baby linear regression using the MCMCpack package:

```
#fit a regression model
freq.fit  <- lm(riders_registered ~ temp_feel, data=bikes)

bayes.fit <- MCMCregress(riders_registered~temp_feel,
                         b0=freq.fit$coefficients,B0=.01,
                         c0=.1,d0=.1,
                         beta.start=freq.fit$coefficients,
                         data=bikes,
                         burnin=10^3,mcmc=10^5)
```

Baby logistic regression using the MCMCpack package:

```
titanic.mod <- MCMClogit(Survived ~ Fare,
    data = titanic.complete)
```

Step 3: Approximating the Posterior via MCMC

Baby example using the mcmc package:

```
library(mcmc)
data(logit)
out <- glm(y~x1,data=logit,family=binomial,x=TRUE)

lupost_factory <- function(x,y)function(beta){
   eta  <- as.numeric(x%*%beta)
   logp <- ifelse(eta<0,eta-log1p(exp(eta)),-log1p(exp(-eta)))
   logq <- ifelse(eta<0,-log1p(exp(eta)),-eta-log1p(exp(-eta)))
   logl <- sum(logp[y==1])+sum(logq[y==0])
   return(logl-sum(beta^2)/8)
   }
lupost <- lupost_factory(out$x,out$y)
```

Step 3: Approximating the Posterior via MCMC

Baby example using the mcmc package:

```
library(mcmc)
data(logit)
out <- glm(y~x1,data=logit,family=binomial,x=TRUE)

lupost_factory <- function(x,y)function(beta){
   eta  <- as.numeric(x%*%beta)
   logp <- ifelse(eta<0,eta-log1p(exp(eta)),-log1p(exp(-eta)))
   logq <- ifelse(eta<0,-log1p(exp(eta)),-eta-log1p(exp(-eta)))
   logl <- sum(logp[y==1])+sum(logq[y==0])
   return(logl-sum(beta^2)/8)
   }
lupost <- lupost_factory(out$x,out$y)
```

```
set.seed(317)
beta.init <- as.numeric(coefficients(out))
out <- metrop(lupost,beta.init,1e3)
names(out)
out$accept
```

Step 4: Assessing the Posterior Approximation

Visualizing the MCMC Samples

- Trace plots: make sure the chains mix well and don't stay in one place too long (better to bounce around and travel the space)
- Caterpillar plots: visualizes each parameter's credible interval and median (especially useful for many parameters)
- Density plots: shows where the parameter's posterior distribution lies and helps compare across chains (if densities differ between chains, then the chains aren't running long enough or the chains aren't mixing well)
- Autocorrelation plots: helps you understand the degree of correlation between steps in a chain (high correlation means you will need to sample longer and–in very specific situations–will want to thin)
- Running mean plots: reminds you law of large numbers exists :)
- Many diagnostic plots (PSRF, Geweke, etc)

Step 4: Assessing the Posterior Approximation

R Packages for Visualizing the MCMC Samples

- basicMCMCplots
- MCMCvis (also for summarizing and manipulating MCMC samples)
- bayesplot (for tidyverse and ggplot2 fans)
- ggmcmc (for tidyverse and ggplot2 fans)

Step 4: Assessing the Posterior Approximation

Did the MCMC sampler run long enough to create a suitably-detailed approximation of the posterior? Gelman-Rubin (1992):

$$\text{psrf} = \sqrt{\frac{\text{chain length} - 1}{\text{chain length}} + \frac{\text{between-chain variance}}{\text{within-chain variance}}}$$

`psrf` decreases to 1 as chain length increases

Step 4: Assessing the Posterior Approximation

Did the MCMC sampler run long enough to create a suitably-detailed approximation of the posterior? Gelman-Rubin (1992):

$$\text{psrf} = \sqrt{\frac{\text{chain length} - 1}{\text{chain length}} + \frac{\text{between-chain variance}}{\text{within-chain variance}}}$$

`psrf` decreases to 1 as chain length increases

```
> library(stableGR)
> stable.GR(bikemod)$psrf
[1] 1.000008 1.000011 1.000006
```

Thanks to batch means variance estimation, `psrf` can be calculated whether we have one chain or multiple chains.

Step 4: Assessing the Posterior Approximation

In a multivariate setting, it's better to check the multivariate psrf.

- Assesses joint convergence rather than component-wise.
- Like `psrf`, `mpsrf` decreases to 1 as chain length increases.

```
> stable.GR(bikemod)$mpsrf
[1] 1.000006
```

Step 4: Assessing the Posterior Approximation

In a multivariate setting, it's better to check the multivariate psrf.

- Assesses joint convergence rather than component-wise.
- Like `psrf`, `mpsrf` decreases to 1 as chain length increases.

```
> stable.GR(bikemod)$mpsrf
[1] 1.000006
```

Is this low enough? Use `target.psrf` from `stableGR`.

```
> target.psrf(p=3, m=1)
$`psrf`
[1] 1.000062
```

Previously, the recommended threshold for (m)psrf was 1.1. We now know that this almost always results in premature termination of the MCMC sampler.

Step 4: Assessing the Posterior Approximation

Equivalently, `n.eff` from `stableGR` checks if sampler ran long enough.

```
> n.eff(bikemod)
$`n.eff`
[1] 8947.84
```

Effective sample size: number of uncorrelated samples that produce the same precision as the MCMC sample.

```
$converged
[1] TRUE
```

Did our sample achieve the target psrf?

```
$n.target
NULL
```

If not, `n.target` approximates target Monte Carlo sample size to hit the target psrf.

Step 5: Using the MCMC Samples

Inference

- Point estimation
- Credible intervals/regions and hypotheses testing
- Prediction
- Calculate covariance and Monte Carlo standard error
- Definitely depends on your research question

R Packages

- HDInterval
- mcmcse
- Many more!
- Definitely depends on your research question

Step 5: Using the MCMC Samples

Basic model info (estimates and Monte Carlo standard errors):

```
> library(mcmcse)
> mcse.mat(bikemod)
                          est           se
(Intercept)      -667.55493    2.57855798
temp_feel          57.88718    0.03525018
sigma2        1720954.96124 999.41040863
```

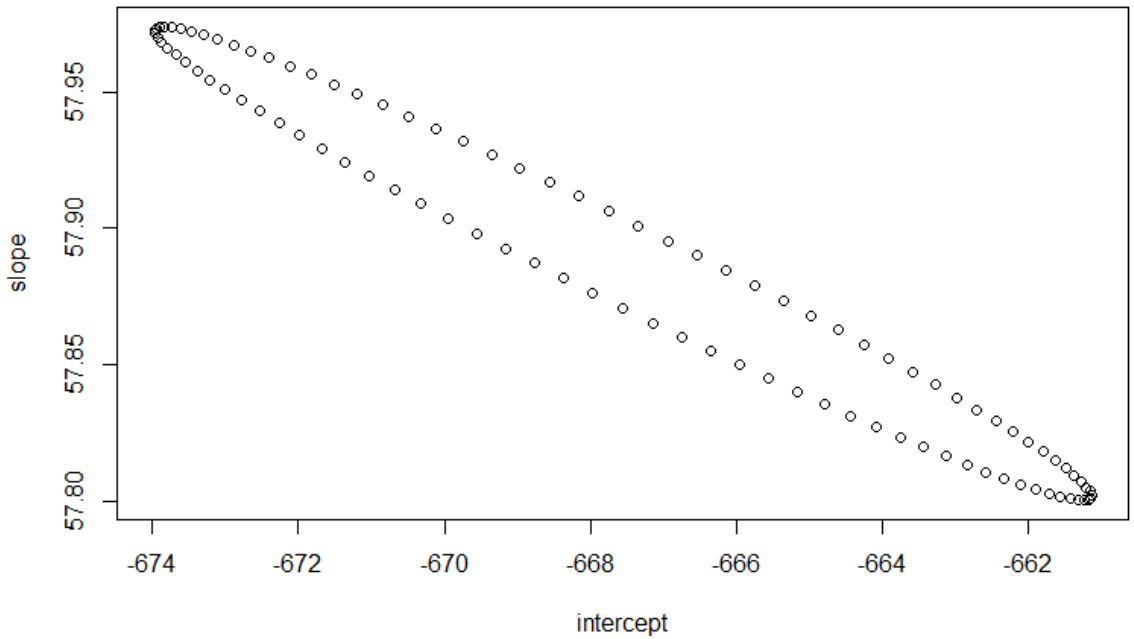
As the chain length increases, the Monte Carlo standard error decreases.

Monte Carlo SE measures the variability from run to run.

Step 5: Using the MCMC Samples

Plotting joint confidence regions visualizes the dependence between a pair of parameters:

```
> library(mcmcse)
> plot(confRegion(mcse.multi(bikemod)),
+      xlab = "intercept", ylab = "slope")
```



Step 5: Using the MCMC Samples

Calculate quantiles and credible intervals using `quantile`:

```
> quantile(bikemod[,2], c(.025, .975))
    2.5%    97.5%
51.39411 64.37282
```

The 95% credible interval for the slope does not contain 0. There is a significant linear relationship between the number of bikers and the perceived temperature.

Step 5: Using the MCMC Samples

Calculate quantiles and credible intervals using `quantile`:

```
> quantile(bikemod[,2], c(.025, .975))
    2.5%    97.5%
51.39411 64.37282
```

The 95% credible interval for the slope does not contain 0. There is a significant linear relationship between the number of bikers and the perceived temperature.

Or find the shortest (highest posterior density) credible intervals:

```
> library(HDInterval)
> hdi(bikemod)
      (Intercept) temp_feel  sigma2
lower  -1174.6978  51.37127 1542111
upper   -188.8293  64.29288 1896111
attr(,"credMass")
[1] 0.95
```

Step 5: Using the MCMC Samples

Monte Carlo standard errors for quantiles:

```
> library(mcmcse)
> mcse.q.mat(bikemod, method = "bm", q=.025)
                        est           se
(Intercept)   -1160.45971 5.456536e+00
temp_feel        51.39404 9.531083e-02
sigma2      1550503.18969 1.889143e+03
```

Step 5: Using the MCMC Samples

Covariance between the parameters:

```
> library(mcmcse)
> mcse.multi(bikemod)
$`cov`
                [,1]           [,2]           [,3]
[1,]    65872.4975    -874.52630     4282471.32
[2,]     -874.5263      12.04724      -71047.43
[3,] 4282471.3190 -71047.42549 9311035810.16
```

Thank You!

Additional Resources

- Giant list of Bayesian R packages: "CRAN Task View: Bayesian Inference" (`CRAN.R-project.org/view=Bayesian`)
- Book: *Statistical Rethinking: A Bayesian Course with Examples in R and Stan* (`xcelab.net/rm/statistical-rethinking/`)
- Bookdown: *Statistical Rethinking with brms, ggplot2, and tidyverse* (`bookdown.org/ajkurz/Statistical_Rethinking_recoded/`)
- Book: *Handbook of Markov Chain Monte Carlo* (`www.mcmchandbook.net`)

Questions?

Email me at knud8583@stthomas.edu

Find these slides at `cknudson.com`

References

James M. Flegal, John Hughes, Dootika Vats, and Ning Dai. (2018). mcmcse: Monte Carlo Standard Errors for MCMC. R package version 1.3-3. Riverside, CA, Denver, CO, Coventry, UK, and Minneapolis, MN.

Gelman, A. and Rubin, D. B. (1992). Inference from iterative simulation using multiple sequences (with discussion). *Statistical Science*, 7:457-472.

Charles J. Geyer and Leif T. Johnson (2019). mcmc: Markov Chain Monte Carlo. R package version 0.9-6. `https://CRAN.R-project.org/package=mcmc`

Christina P. Knudson and Dootika Vats (2019). stableGR: A Stable Gelman-Rubin Diagnostic for Markov Chain Monte Carlo. R package version 1.0. `https://CRAN.R-project.org/package=stableGR`.

Andrew D. Martin, Kevin M. Quinn, Jong Hee Park (2011). MCMCpack: Markov Chain Monte Carlo in R. Journal of Stat Software. 42(9): 1-21.