An Introduction to Bayesian Statistics (Part II)

Christina Knudson, Ph.D.

noRth

August 16, 2019

Recall from Part I of Intro to Bayes

- Nuts and bolts:
  - Prior represents information known about parameters before the study
  - Likelihood represents the data collected
  - Posterior (distribution of parameters given the data) combines the prior and the likelihood

Recall from Part I of Intro to Bayes

- Nuts and bolts:
  - Prior represents information known about parameters before the study
  - Likelihood represents the data collected
  - Posterior (distribution of parameters given the data) combines the prior and the likelihood
- Bayesian inference primarily focuses on the posterior (e.g. posterior mean, median, variance)

Recall from Part I of Intro to Bayes

- Nuts and bolts:
  - Prior represents information known about parameters before the study
  - Likelihood represents the data collected
  - Posterior (distribution of parameters given the data) combines the prior and the likelihood
- Bayesian inference primarily focuses on the posterior (e.g. posterior mean, median, variance)
- "Conjugate" priors are convenient because they combine with the likelihood to produce a well-known posterior (e.g. normal prior & normal likelihood $\longrightarrow$ normal posterior)

Moving Away From a Conjugate Prior

What do we do without a conjugate prior?

Posterior probably won't be a recognizable distribution, so you will need to work a little harder to conduct inference.

Basic Bayesian Steps

1. Select a model and priors
2. Approximate the posterior via Markov chain Monte Carlo
3. Check the posterior approximation (e.g. sufficient samples)
4. Use the MCMC samples to conduct inference

Either code it yourself or use one of the MANY packages on CRAN.

Approximating the Posterior via MCMC

Approximate the posterior by using Markov chain Monte Carlo (MCMC) to sample from the posterior distribution.

Approximating the Posterior via MCMC

Approximate the posterior by using Markov chain Monte Carlo (MCMC) to sample from the posterior distribution.

How does MCMC sampling generally work?

1. Select a starting value for each parameter
2. Iterate between the following two steps:
    1. Propose new values based on the current parameter values
    2. Move to the proposed values with some probability, or stay at the current position with the complementary probability

The exact method of selecting proposed values and calculating the probability of moving depends on the exact MCMC sampler.

Approximating the Posterior via MCMC

Some packages (such as `MCMCpack`) contain functions to perform **specific methods of Bayesian inference**.

`MCMCpack` does MCMC in the context of specific statistical models.

Some packages (such as `mcmc`) focus on the MCMC (independent of the model/context) and are therefore **more general**.

`mcmc` simulates using a user-inputted log unnormalized posterior density.

Bike Regression: Create Model with MCMCpack

To fit a Bayesian linear model with MCMCpack, use MCMCregress.

Creates a single chain using a Gibbs sampler with priors of:

- multivariate normal for regression coefficients
- inverse gamma for the variance of the residuals

```
> library(MCMCpack)
> bikemod <- MCMCregress(riders_registered ~ temp_feel,
+                   data = bikes)
>
> bikemod2 <- MCMCregress(riders_registered ~
+                   temp_feel + humidity, data = bikes)
```

Bike Regression: Create Model with MCMCpack

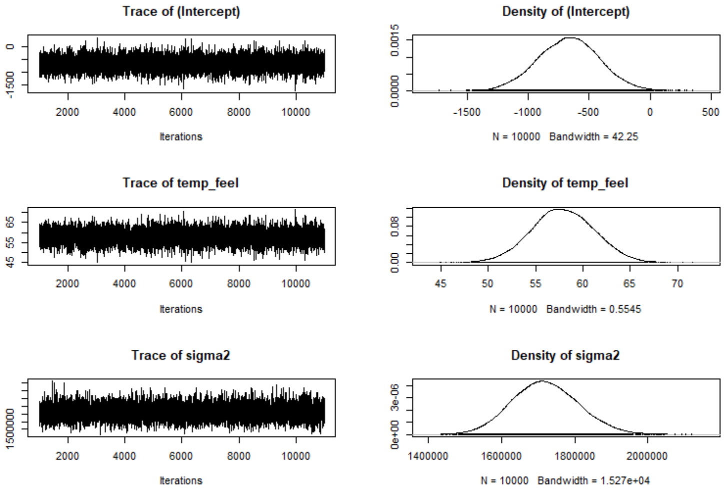
```
> library(MCMCpack)
> bikemod <- MCMCregress(riders_registered ~ temp_feel,
+                                data = bikes)
```

Some MCMCregress defaults:

- mcmc = 10000: Monte Carlo sample size (after burnin)
- burnin = 1000: discard first 1000 samples
- thin = 1: no thinning; retain all the samples
- beta.start = NA: uses OLS estimates for initial values
- b0 = 0: prior mean of 0 for the regression coefficients
- B0 = 0: precision of 0 (infinite variance) for the prior

Bike Regression: Plot Approximation with MCMCpack

```
> plot(bikemod)
```



Bike Regression: Assess Approximation with stableGR

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

$mpsrf
[1] 1.000006

$means
  (Intercept)      temp_feel         sigma2
   -667.55493       57.88718 1720954.96124

$n.eff
[1] 8947.84
```

Bike Regression: Assess Approximation with `stableGR`

Did the sampler run long enough? 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.

Bike Regression: Assess Approximation with `stableGR`

Did the sampler run long enough? 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.

```
> 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!

Bike Regression: Assess Approximation with stableGR

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
```

Bike Regression: Assess Approximation with stableGR

Equivalently, `n.eff` from `stableGR` checks whether 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.

Bike Regression: Assess Approximation with `mcmcse`

Get 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.

Bike Regression: Multivariate Analysis with `mcmcse`

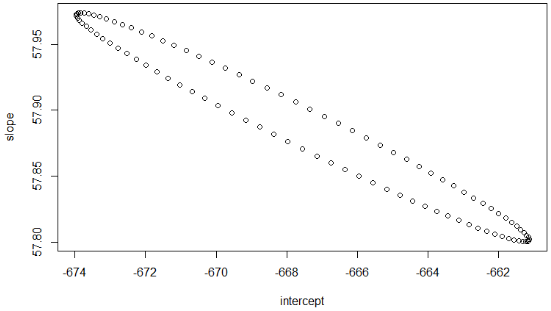
Inspect the covariance between the parameters:

```
> 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
```

Bike Regression: Multivariate Analysis with `mcmcse`

Plot joint confidence regions for a pair of parameters:

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



Bike Regression: Quantiles and Credible Intervals

Calculate quantiles and credible intervals using `quantile`:

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

Bike Regression: Quantiles and Credible Intervals

Calculate quantiles and credible intervals using `quantile`:

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

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
```

Bike Regression: Quantiles and Credible Intervals

Monte Carlo standard errors are always a good idea!.

Quantiles and their Monte Carlo standard errors (`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
```

Wrapping Up

Basic Bayesian Steps

1. Select a model and priors
2. Approximate the posterior via Markov chain Monte Carlo
3. Check the posterior approximation (e.g. sufficient samples)
4. Use the MCMC samples to conduct inference

Wrapping Up

Basic Bayesian Steps

1. Select a model and priors
2. Approximate the posterior via Markov chain Monte Carlo
3. Check the posterior approximation (e.g. sufficient samples)
4. Use the MCMC samples to conduct inference

Some R Packages

- `MCMCpack` for making specific Bayesian models
- `mcmc` for general MCMC sampling
- `stableGR` for assessing the posterior approximation with Gelman-Rubin diagnostic and effective sample size
- `mcmcse` for conducting multivariate analyses on the posterior

More Information

Questions?

`cknudson.com`

knud8583@stthomas.edu

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.

Vats, D., Flegal, J. M, and Jones, G. L. (2019). Multivariate output analysis for Markov chain Monte Carlo. *Biometrika*, 106(2):321-337.

Vats, D. and Knudson, C. Revisiting the Gelman-Rubin diagnostic, arXiv:1812.09384 (under review).

Additional Concerns: MCMC Convergence

If the MCMC sampler runs long enough, it will produce an adequate approximation of the posterior distribution.

Two types of convergence in MCMC. Run long enough to:

- Cover all the plausible parameter values
- Produce a sufficiently detailed approximation

This is a big area of research!

Sorry that we can't cover convergence in this introductory talk.

Stopping the MCMC Sampler

How can we decide to stop sampling?

We could terminate after:

- A fixed number of steps
- The Monte Carlo standard error becomes sufficiently small
- A diagnostic (e.g. Gelman-Rubin) hits a threshold

Additional Concerns: Number of Chains

Clearly, it's better to have more samples.

But how should we get those samples? From one long chain or several shorter chains?

Asking this question is a good way to start a fight amongst Bayesians!

Commands for Installing R Package stableGR from Github

```
  #get some required packages
install.packages("Rcpp")
install.packages("RcppArmadillo")
install.packages("devtools")

  #install mcmcse from github  (rather than CRAN)
library(devtools)
install_github("dvats/mcmcse")

  #install stableGR package
install_github("knudson1/stableGR/stableGR")
library(stableGR)
```

Will be available on CRAN in a couple months.