Faster Computation for Hierarchical Bayesian Models with Rcpp Packages

Lu Chen[*,#], Balgobin Nandram[+], Nathan B. Cruze[#]

[*]National Institute of Statistical Sciences (NISS)

[+]Worcester Polytechnic Institute

[#]United States Department of Agriculture
National Agricultural Statistics Service (NASS)

GASP II
Washington, DC
September 23, 2019

Disclaimer and Acknowledgment

The findings and conclusions in this presentation are those of the authors and should not be construed to represent any official USDA or US Government determination or policy.

- This research was supported in part by the intramural research program of the U.S. Department of Agriculture, National Agriculture Statistics Service.

Outline

Motivation

"Sometimes R code just isn't fast enough."

– Hadley Wickham

- Problem: How to combine survey and auxiliary data to improve the county-level estimates for crops?
- Application: Bayesian small area models
- Potential bottlenecks of R code: subsequent iterations, repeatedly calling functions, loops in Markov chain Monte Carlo (MCMC) algorithms
- One solution: rewriting key functions in C++ through **Rcpp packages**

Rcpp Packages

- **Rcpp** is a R package to extend R with C++ codes developed by Dirk Eddelbuettel and Romain Francois (2013) .
  - Speed
  - New Things
- **RcppArmadillo** is a Rcpp extension package that provides all the functionality of Armadillo, focusing on **matrix math**.
  - Easy-to-use
  - Further speedup

Using sourceCpp() in R

- The Rcpp::sourceCpp function parses the C++ file (.cpp) and makes C++ functions available as R functions.

Example: Calculate mean of $x_i, i = 1, \ldots, 10^5$, where $x_i \sim U(0,1)$.

cpp1.cpp

```cpp
#include <Rcpp.h>
using namespace Rcpp;

// [[Rcpp::export]]
double meanC(NumericVector x) {
  int n = x.size();
  double total = 0;

  for(int i = 0; i < n; ++i) {
    total += x[i];
  }
  return total/n;
}
```

r1.R

```r
Rcpp::sourceCpp('cpp1.cpp')
x <- runif(1e5)
microbenchmark(
        mean(x), #build-in R mean() function
        meanC(x) #C++ function
)
```

Using sourceCpp() in R

- The Rcpp::sourceCpp function parses the C++ file (.cpp) and makes C++ functions available as R functions.

Example: Calculate mean of $x_i, i = 1, \ldots, 10^5$, where $x_i \sim U(0, 1)$.

cpp1.cpp

```cpp
#include <Rcpp.h>
using namespace Rcpp;

// [[Rcpp::export]]
double meanC(NumericVector x) {
  int n = x.size();
  double total = 0;

  for(int i = 0; i < n; ++i) {
    total += x[i];
  }
  return total/n;
}
```

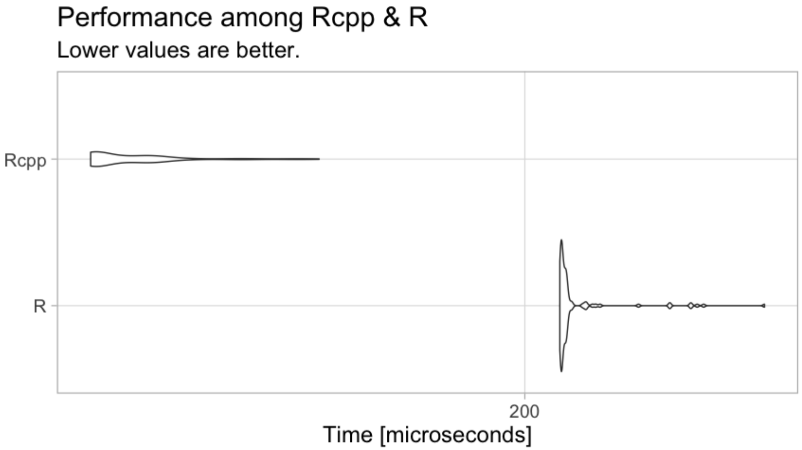
Function Name

input

r1.R

```r
Rcpp::sourceCpp('cpp1.cpp')
x <- runif(1e5)
microbenchmark(
        mean(x), #build-in R mean() function
        meanC(x) #C++ function
)
```

Comparison



name	min	mean	median	max	times
Rcpp	105.432	151.448	109.209	228.249	100
R	210.845	237.125	235.366	396.785	100

MCMC in Bayesian Computation

- MCMC is a sampling method to draw random samples from distributions.
- Each random sample is used as a stepping stone to generate the next one (chains).
- Gibbs sampler, Metropolis-Hastings sampler and many others are widely used in Bayesian inference.
- Involve **loops** and **calling functions repeatedly** within loops.

Rcpp (C++) and RcppArmadillo are useful tools for efficient MCMC computation.

Simulated Data

Data: simulated planted acres data in Illinois (Nandram et al., 2019 and Battese et al., 1988)

- Survey estimates $\hat{\theta}_i, \ i = 1, \ldots, 102$
- Survey standard errors $\hat{\sigma}_i, \ i = 1, \ldots, 102$
- Covariates: corn and soybean planted acres from land observatory satellites (LANDSAT)

Fay-Herriot Model

Fay-Herriot Model (1979) in small area estimation:

$$\hat{\theta}_i|\theta_i \overset{ind}{\sim} N(\theta_i, \hat{\sigma}_i^2),$$

$$\theta_i|\boldsymbol{\beta}, \delta^2 \overset{ind}{\sim} N(\mathbf{x}_i'\boldsymbol{\beta}, \delta^2), \;\; i = 1, \ldots, n,$$

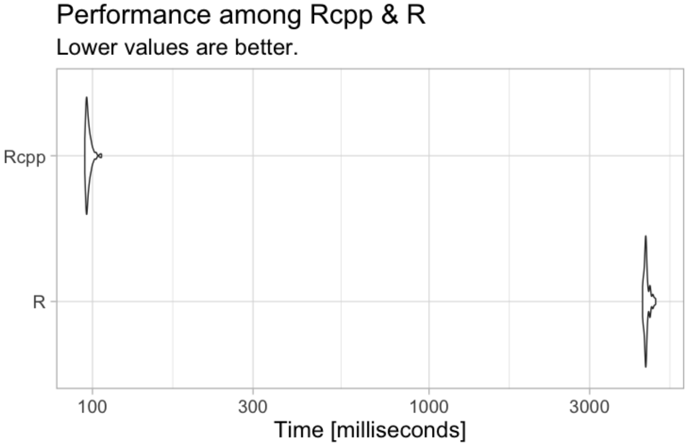
Priors for the parameters: $\pi(\boldsymbol{\beta}) \propto 1; \pi(\delta^2) \propto \frac{1}{\delta^2}$.

The full conditional distributions for Gibbs sampling are:

1. $\boldsymbol{\beta}|\delta^2 \sim MVN\Big((\sum_{i=1}^n \mathbf{x}_i\mathbf{x}_i')^{-1}(\sum_{i=1}^n \mathbf{x}_i'\boldsymbol{\beta}), \delta^2(\sum_{i=1}^n \mathbf{x}_i\mathbf{x}_i')^{-1}\Big)$,
2. $\theta_i|\boldsymbol{\beta}, \delta^2 \overset{ind}{\sim} N(\lambda_i\hat{\theta}_i + (1-\lambda_i)\mathbf{x}_i'\boldsymbol{\beta}, (1-\lambda_i)\delta^2), \;\; \lambda_i = \frac{\delta^2}{\delta^2+\hat{\sigma}_i^2}$,
3. $\delta^2|\boldsymbol{\theta}, \boldsymbol{\beta} \sim IG\Big(\frac{n-1}{2}, \frac{1}{2}\sum_{i=1}^n(\theta_i - \mathbf{x}_i'\boldsymbol{\beta})^2\Big)$.

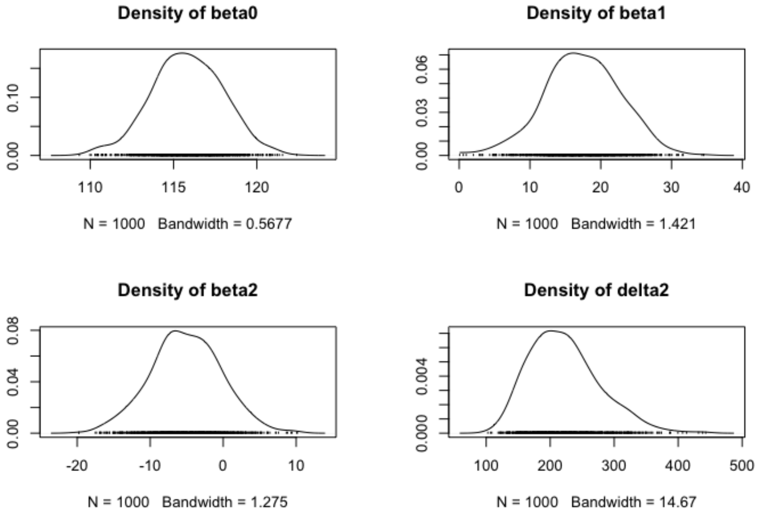
Comparison

12,000 iterations with 2,000 burn-in and pick every 10^{th} sample

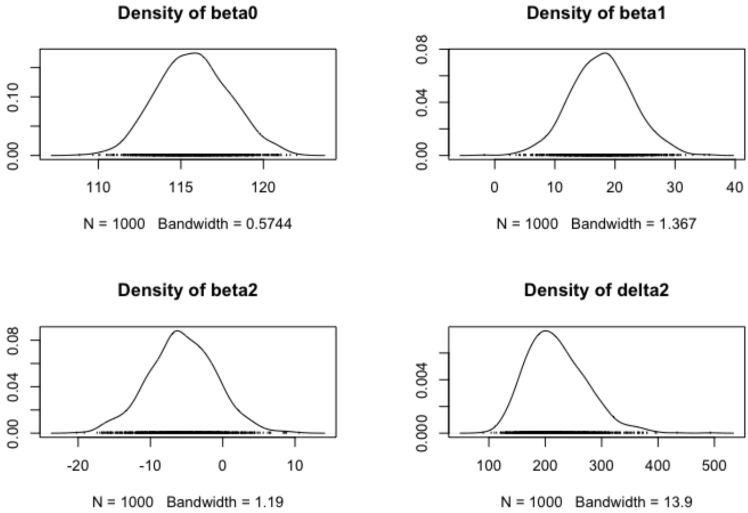


name	min	mean	median	max	times
Rcpp	94.863	97.285	96.621	106.362	100
R	4707.729	4856.196	4829.291	4912.025	100

Comparison - R density plots



Comparison - Rcpp density plots



Fay-Herriot Model with Benchmarking Constraints

In some applications, we need to incorporate benchmarking constraints into the model. For example, the county-level estimates should be summed to state target and they need to cover certain values. The model with inequality constraints:

$$\hat{\theta}_i|\theta_i \overset{ind}{\sim} N(\theta_i,\ \hat{\sigma}_i^2),\ i = 1, \ldots, n,$$

$$\theta_i|\boldsymbol{\beta}, \delta^2 \overset{ind}{\sim} N(\mathbf{x}_i'\boldsymbol{\beta}, \delta^2),\ \theta_i \geq c_i, \sum_{i=1}^{n} \theta_i \leq a,$$

where $\mathbf{C} = (c_1, \ldots, c_n)'$ are known and fixed and a is state target. The priors are $\pi(\boldsymbol{\beta}) \propto 1; \delta^2 \propto \frac{1}{(1+\delta^2)^2}$.

Joint Posterior Distribution

The posterior density is

$$\pi(\boldsymbol{\theta}, \boldsymbol{\beta}, \delta^2 | \hat{\boldsymbol{\theta}}, \hat{\boldsymbol{\sigma}}^2) = \frac{\prod_{i=1}^{n} \phi((\theta_i - \mathbf{X}'\boldsymbol{\beta})/\delta)\phi((\theta_i - \hat{\theta}_i)/\hat{\sigma}_i)}{\int_{\theta \in V} \prod_{i=1}^{n} \phi((\theta_i - \mathbf{X}'\boldsymbol{\beta})/\delta) d\theta}, \ \boldsymbol{\theta} \in V,$$

where $\phi(\cdot)$ is the standard normal density and the support of $\boldsymbol{\theta}$ is

$$V = \left\{ c_i \leq \theta_i, \sum_{i=1}^{n} \theta_i \leq a, \ i = 1, \ldots, n \right\}.$$

Awkward joint posterior distribution and intractable.

Computation

Our strategy:

$$\pi(\boldsymbol{\theta}, \boldsymbol{\beta}, \delta^2 | \hat{\theta}, \hat{\sigma}^2) = \boxed{\pi(\boldsymbol{\beta}, \delta^2 | \hat{\boldsymbol{\theta}}, \hat{\boldsymbol{\sigma}}^2)} \times \pi(\boldsymbol{\theta} | \boldsymbol{\beta}, \delta^2, \hat{\boldsymbol{\theta}}, \hat{\boldsymbol{\sigma}}^2)$$

- Metropolis-Hastings Sampler

Computation

Our strategy:

$$\pi(\boldsymbol{\theta}, \boldsymbol{\beta}, \delta^2 | \hat{\theta}, \hat{\sigma}^2) = \boxed{\pi(\boldsymbol{\beta}, \delta^2 | \hat{\boldsymbol{\theta}}, \hat{\boldsymbol{\sigma}}^2)} \times \pi(\boldsymbol{\theta} | \boldsymbol{\beta}, \delta^2, \hat{\boldsymbol{\theta}}, \hat{\boldsymbol{\sigma}}^2)$$

- Metropolis-Hastings Sampler
- Gibbs Sampler

Metropolis-Hastings Sampler

We will draw $(\boldsymbol{\beta}, \delta^2)$ samples from $\pi(\boldsymbol{\beta}, \delta^2 | \hat{\boldsymbol{\theta}}, \hat{\boldsymbol{\sigma}}^2)$. The proposal density is

$$(\boldsymbol{\beta}, \log(\delta^2)) \sim MVN(\hat{\boldsymbol{\beta}}_p, \sigma^2 \hat{\Sigma}_p)$$
$$\nu/\sigma^2 \sim \Gamma(\nu/2, 1/2)$$

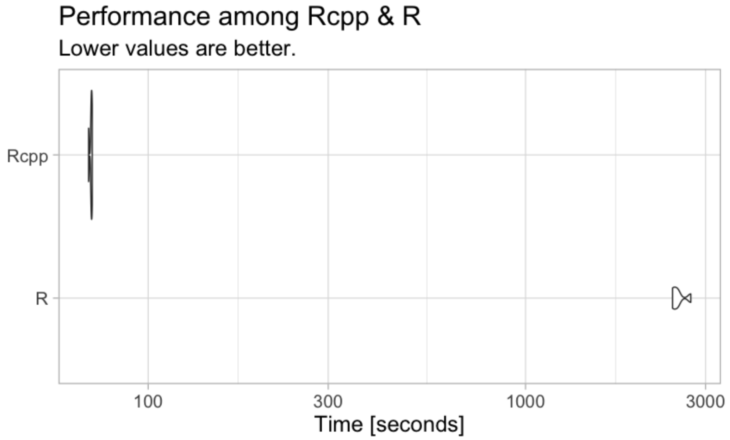
Bottleneck:

For each iteration h:

- **Generate**: Generate a candidate $(\boldsymbol{\beta}^c, \log(\delta^2)^c)$ from proposal density;
- **Calculate**: Calculate the acceptance ratio $\alpha = \pi(\boldsymbol{\beta}^c, \log(\delta^2)^c)/\pi(\boldsymbol{\beta}^{(h)}, \log(\delta^2)^{(h)})$;
- **Accept or Reject** candidate based on the comparison between α and $u \sim U(0,1)$.

Comparison

10,000 iterations with 2,000 burn-in and pick every 8^{th} sample



name	min	mean	median	max	times
Rcpp	69.527	70.554	70.711	71.073	10
R	2451.688	2540.393	2519.215	2741.561	10

Gibbs Sampler for $\boldsymbol{\theta}$

The conditional posterior density of θ_i is

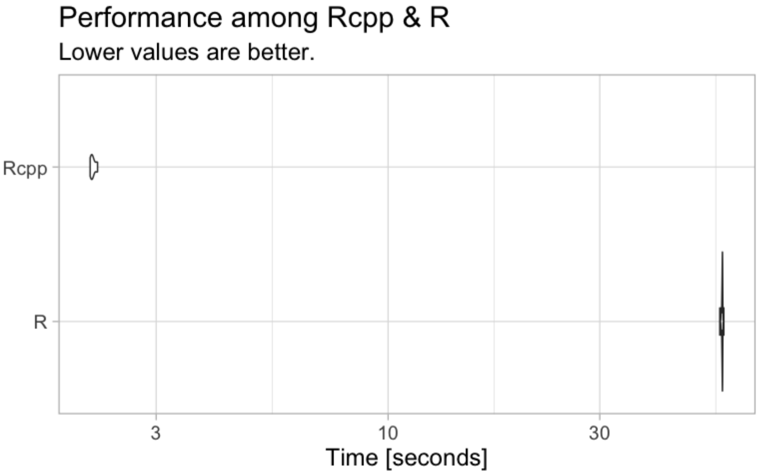
$$\theta_i | \boldsymbol{\theta}_{(i)}, \boldsymbol{\beta}, \delta^2, \hat{\boldsymbol{\theta}}, \hat{\boldsymbol{\sigma}}^2 \sim N\Big(\mu_i, \phi_i\Big), \ \theta_i \in V_i,$$

where μ_i and ϕ_i related to $\boldsymbol{\beta}$ and δ^2 and

$$V_i = \Big\{ c_i \leq \theta_i \leq a - \sum_{j=1, j \neq i}^{n-1} \theta_j \Big\}, \ i = 1, \ldots, n.$$

Bottleneck:
Each θ_i related to other $\boldsymbol{\theta}$s based on the V_i.
For one iteration, we need to loop n times.

Comparison



name	min	mean	median	max	times
Rcpp	2.130	2.162	2.153	2.213	10
R	55.934	56.615	56.696	57.134	10

Convergence Diagnostics

- M-H: 1,000 samples of $(\boldsymbol{\beta}^{(h)}, \delta^{2(h)}), h = 1, \ldots, 1000$.
- Gibbs: for each $(\boldsymbol{\beta}^{(h)}, \delta^{2(h)})$, we run 100 times Gibbs sampler and pick the last set of $\boldsymbol{\theta}$.

	pm	psd	lb	ub	gewe.pval	ess
β_0	116.22	2.01	112.43	120.06	0.45	909
β_1	16.74	4.58	7.75	25.29	0.40	870
β_2	-4.81	4.23	-13.14	3.35	0.24	968
δ^2	325.96	60.78	206.98	469.46	0.64	827

- Rcpp vs R code: **72s** vs **2576s** for 102 samples size in the constraint Bayesian model.

Conclusion

- Rcpp functions can reduce the running time by a significant factor and reasonable in further production for county-level estimates in NASS.
- Large data set or complicated hierarchical Bayesian models: Rcpp packages
  - Pros: incorporating C++ code into R workflow easily; substantially speed up MCMC computation in R
  - Cons: learning curve and long coding time
- Small data set or simple, classic Bayesian models: such as RJags and RStan
  - Pros: Easy-to-use; less coding time
  - Cons: Black-box sampler; not for non-standard problems

Reference

[1] H. Wickham, *Advanced r*. Chapman and Hall/CRC, 2014.

[2] D. Eddelbuettel and R. François, "Rcpp: Seamless R and C++ integration," *Journal of Statistical Software*, vol. 40, no. 8, pp. 1–18, 2011.

[3] D. Eddelbuettel and C. Sanderson, "Rcpparmadillo: Accelerating r with high-performance c++ linear algebra," *Computational Statistics and Data Analysis*, vol. 71, pp. 1054–1063, March 2014.

[4] B. Nandram, A. L. Erciulescu, and N. B. Cruze, "Bayesian benchmarking of the fay-herriot model using random deletion," *Survey Methodology 45-2*, vol. 45, p. 365, 2019.

[5] G. E. Battese, R. M. Harter, and W. A. Fuller, "An error-components model for prediction of county crop areas using survey and satellite data," *Journal of the American Statistical Association*, vol. 83, no. 401, pp. 28–36, 1988.

[6] R. E. Fay III and R. A. Herriot, "Estimates of income for small places: an application of james-stein procedures to census data," *Journal of the American Statistical Association*, vol. 74, no. 366a, pp. 269–277, 1979.

Thank You!

lu.chen@usda.gov