gpuideal: Fast Fully Bayesian Estimation of Ideal Points with Massive Data

Jeffrey B. Lewis*
Chris Tausanovitch†

August 2018

In this white paper, we describe the R package **gpuideal**. This software package estimates two-parameter item response models, often used in political science to calculate legislator ideal points based on roll call vote matrices (see Clinton, Jackman, and Rivers, 2004). The software is much faster than existing fully Bayesian approaches (Jackman, 2017) but unlike alternative estimation approaches such as Expectation Maximization (Imai, Lo, and Olmsted, 2016) it still provides draws from the full posterior of the model. This is essential for estimating quantities such as polarization, where naive estimates based on maximum or expected a posteriori estimates can be highly misleading (Hill and Tausanovitch, 2015). The software relies on the fact that the estimation procedure used by Jackman (2017) is highly parallelizable and can take advantage of the wide availability of graphics processing units.

*Professor, Department of Political Science, UCLA, jblewis@ucla.edu.
†Assistant Professor, Department of Political Science, UCLA, ctausanovitch@ucla.edu.

1 Introduction

gpuideal is an R (R Core Team, 2018) package for fast, fully Bayesian estimation of ideal points. It is meant to improve upon the functionality of existing software such as **ideal** (Jackman, 2017) or **MCMCirtKd** (Martin, Quinn, and Park, 2011), although for the time being it only estimates one-dimensional models. In the following white paper we describe the model and algorithm underlying **gpuideal**, as well as how it is implemented. We compare its performance to existing alternatives. **gpuideal** is many times faster than other options, save for **emIRT**. However unlike **gpuideal**, **emIRT** does not produce posterior distributions for the model's parameters and other statistics of interest.

2 The Model

Take the standard one-dimensional binary item response model with normal shocks (see Clinton, Jackman, and Rivers, 2004) which yields the following choice probabilities:

$$\begin{aligned} \Pr(y_{ij} = 1) &= \Pr(\tilde{x}_{i.}\tilde{\beta}_{.j} + \epsilon_{ij} > 0) = \Phi(\tilde{x}_{i.}\tilde{\beta}_{.j}) \text{ and} \\ \Pr(y_{ij} = 0) &= \Pr(\tilde{x}_{i.}\tilde{\beta}_{.j} + \epsilon_{ij} \leq 0) = 1 - \Phi(\tilde{x}_{i.}\tilde{\beta}_{.j}) \end{aligned} \tag{1}$$

for $i = \{1, \ldots, N\}$ and $j = \{1, \ldots, M\}$ where $\tilde{x} = (1, x)$ is an $N \times 2$ matrix consisting of a vector of ones and a vector x of the N person ideal points and $\tilde{x}_{i.}$ is the row vector $(1, x_i)$ containing the ith row of x. We will refer to these people as "legislators" following the roll call voting literature, but the model can be applied to many types of units and binary outcomes. $\tilde{\beta}$ is a $2 \times M$ matrix of parameters such that $\tilde{\beta}^\top = [\alpha\ \beta]$, α and β are column vectors of length M, and $\tilde{\beta}_{.j}$ is a column vector containing the jth column of $\tilde{\beta}$. These parameters characterize each of the M outcomes, which we refer to as "roll calls." ϵ is an $N \times M$ matrix of independently and identically distributed standard normal random shocks. y is the $N \times M$ data matrix, where rows correspond to legislators and columns correspond to roll calls. Each

entry in y is coded 1 for a vote in favor and 0 for a vote against. The function Pr assigns probability and Φ is the standard normal CDF.

3 The Algorithm

Following Jackman (2009, p. 456), we estimate the model using the data augmentation approach presented by Albert and Chib (1993). Let y^* be an $N \times M$ matrix of latent propensities for each legislator to vote in favor of each bill, such that

$$y^* = \tilde{x}\tilde{\beta} + \epsilon. \tag{2}$$

Take some set of starting values for x, and call these $x^{(0)}$, denoting the full matrix $\tilde{x}^{(0)} = (1, x^{(0)})$. In addition, generate starting values for the item parameters, $\tilde{\beta}^{(0)\top} = [\alpha^{(0)}\ \beta^{(0)}]$. This can be done simply by sampling from the prior distributions, but a better approach is to find reasonable values using a fast approximation such as factor analysis or **emIRT**.

Starting values for y^* can be drawn directly from Equation 2 subject to the restriction that $y_{ij}^{*(0)} > 0$ for $y_{ij} = 1$ and $y_{ij}^{*(0)} \leq 0$ for $y_{ij} = 0$:

$$\begin{aligned} y_{ij}^{*(0)} &\sim N(\tilde{x}_{i\cdot}^{(0)}\tilde{\beta}_{\cdot j}^{(0)}, 1) \text{ left truncated at 0 for } y_{ij} = 1 \text{ and} \\ y_{ij}^{*(0)} &\sim N(\tilde{x}_{i\cdot}^{(0)}\tilde{\beta}_{\cdot j}^{(0)}, 1) \text{ right truncated at 0 for } y_{ij} = 0 \end{aligned} \tag{3}$$

where $i \in \{1, \ldots, N\}$ indexes legislators and $j \in \{1, \ldots, M\}$ indexes roll calls.

In each iteration t of the algorithm, we draw $x^{(t)}$ conditional on $\tilde{\beta}^{(t-1)}$ and $y^{*(t-1)}$. We then draw $\tilde{\beta}^{(t)}$ conditional on $y^{*(t-1)}$ and $x^{(t)}$. Finally we draw $y^{*(t)}$ conditional on $x^{(t)}$ and $\tilde{\beta}^{(t)}$. This cycle is repeated until we achieve convergence for the posterior densities of the parameters.

Observe that Equation 2 resembles a regression model, but y^* is a matrix, not a vector. In order to draw values of $x^{(t)}$ we observe that $\tilde{x}\tilde{\beta} = \tilde{\alpha} + x\beta^\top$ where $\tilde{\alpha}$ is an $N \times M$ matrix

where every column is equal to α. Rearranging:

$$y^{*(t-1)} - \tilde{\alpha}^{(t-1)} = x^{(t)}\beta^{(t-1)\top} + \epsilon. \tag{4}$$

Consider the ith row of y^*:

$$(y_{i.}^{*(t-1)} - \tilde{\alpha}_{i.}^{(t-1)})^\top = x_i^{(t)}\beta^{(t-1)} + \epsilon_{i.}. \tag{5}$$

This has the familiar linear regression form with $(y_{i.}^{*(t-1)} - \tilde{\alpha}_{i.}^{(t-1)})^\top$ being regressed on $\beta^{(t-1)}$ where $x_i^{(t)}$ is the coefficient and the intercept is suppressed. If we give x_i a normal prior with mean 0 and variance σ_x, normal-normal conjugacy allows us to draw then from the (conditional) posterior of $x_i^{(t)}$ as follows:

$$x_i^{(t)} \sim N((\beta^{(t-1)\top}\beta^{(t-1)} + \sigma_x^{-1})^{-1}\beta^{(t-1)\top}(y_{i.}^{*(t-1)} - \alpha_{i.}^{(t-1)}), (\beta^{(t-1)\top}\beta^{(t-1)} + \sigma_x^{-1})^{-1}). \tag{6}$$

This is simply the posterior of $x_i^{(t)}$ from the Bayesian regression in Equation 5.

Likewise, we can draw $\tilde{\beta_j}^{(t)}$ by considering the regression of $y_{.j}^{*(t-1)}$ on $x^{(t)}$ for each column j. If we give $\tilde{\beta}$ a multivariate normal prior distribution with mean $\mu_{\tilde{\beta}}$ and variance $\Sigma_{\tilde{\beta}}$, we can draw $\tilde{\beta}^{(t)}$ as follows:

$$\tilde{\beta_j}^{(t)} \sim N((\tilde{x}^{(t)\top}\tilde{x}^{(t)} + \Sigma_{\tilde{\beta}}^{-1})^{-1}(\tilde{x}^{(t)\top}y_{.j}^{*(t-1)} + \Sigma_{\tilde{\beta}}^{-1}\mu_{\tilde{\beta}}), (\tilde{x}^{(t)\top}\tilde{x}^{(t)} + \Sigma_{\tilde{\beta}}^{-1})^{-1}). \tag{7}$$

Finally, as in Equation 3, we draw $y_{ij}{}^{*(t)}$ conditional on $x_{i.}^{(t)}$ and $\tilde{\beta}_{.j}^{(t)}$:

$$\begin{aligned} &y_{ij}^{*(t)} \sim N(\tilde{x}_{i.}^{(t)}\tilde{\beta}_{.j}^{(t)}, 1) \text{ left truncated at 0 for } y_{ij} = 1 \text{ and} \\ &y_{ij}^{*(t)} \sim N(\tilde{x}_{i.}^{(t)}\tilde{\beta}_{.j}^{(t)}, 1) \text{ right truncated at 0 for } y_{ij} = 0. \end{aligned} \tag{8}$$

In cases in which not every legislator votes on every roll call, at each update step, the rows or columns associated with any missing y_{ij} are simply excluded from the calculation, as in list-wise deletion. For the x step, each x_i is updated using only the rows of $\tilde{\beta}$ that correspond to non-missing y_{ij}. Likewise, each β_i and α_i is updating using only the xs that correspond to non-missing y_{ij}. Finally, y^*_{ij} is only updated if the corresponding y_{ij} is non-missing.

Like existing approaches, this algorithm "augments" the data with a latent variable, $y*$ allowing Gibbs sampling from easy-to-sample-from normal and truncated normal distributions. A key feature of the estimation problem for our purposes is that the draws of $x_i^{(t)}$ are all independent of one another for all i, as are the draws of $\tilde{\beta}_j^{(t)}$ for all j and $y_{ij}^{*(t)}$ for all i and j, conditioning on the other parameters.

4 Parallelization

When a problem can be easily separated into many processes with no need for communication between them, this is what the parallel computing community calls an "embarrassingly parallel" problem (Herlihy and Shavit, 2011, p. 14). Each of the three steps in the data augmentation algorithm for this model is embarrassingly parallel. The datasets that this model is applied to are often quite large in N, M or both, with an increasing need for speed as the problem grows larger[1]. For large datasets, raw processing speed is of little advantage compared to the advantage that can be gained by adding additional processors. For this reason, this problem is an ideal application for graphical processing units.

Whereas computer processing units (CPUs) are meant for fast execution of serialized processes, graphical processing units (GPUs) are sets of many slower processors tailored for processes with high returns to parallelization. 3-dimensional graphics are the paradigmatic example, hence the name. A typical consumer graphics card has hundreds of cores, and cards designed for scientific computing can have thousands. Executing our algorithm requires

[1]Imai, Lo, and Olmsted (2016) identify 16 recent high-profile applications in political science across different substantive area, often using millions of individuals and thousands of choices.

transferring data to and from memory after each operation, which adds overhead and costs processing time. However to get a sense of the potential for higher performance, consider a problem where N=M and a GPU with N cores. If there were no overhead, the speed of each step could be improved by a factor of N, for an N times overall speed improvement over a single core. GPUs are typically much slower than CPUs, but as N grows large this difference becomes immaterial compared to the gains from parallelization.

We implement the parallelization of this algorithm using NVIDIA's proprietary CUDA platform. As a result it is currently necessary to have an NVIDIA graphics card in order to use **gpuideal**.

5 Speed

Any test of this software will be hardware dependent. For this test we used an NVIDIA GeForce GTX 550 Ti graphics card with 192 cores and an Intel Xeon E5-2620 processor. At the time that they were purchased, in October of 2012, this was a middle-of-the-road graphics card costing $124 and a server-grade CPU processor costing $423. All tests that do not involve GPUs are single-threaded for comparability, despite the fact that the CPU has 6 cores. **emIRT** allows for parallelization of the bootstrap trials on multi-core CPUs, which could lead to significant performance improvements.

Following Imai, Lo, and Olmsted (2016), we test the speed of the software using roll call voting data from the 102nd through the 112th House of Representatives. We drop legislators who voted on fewer than 25 non-unanimous bills, as they do. We also follow their procedures in terms of the number of iterations and convergence criteria to be used for each estimation method. For our MCMC-based method we run and 120,000 iterations and discard the first 20,000. For **emIRT** we use the default setting of requiring each iteration to be correlated with the previous iteration at $1 - 10^{-6}$ for convergence. For the bootstrapped version we run 100 bootstrap replicates.

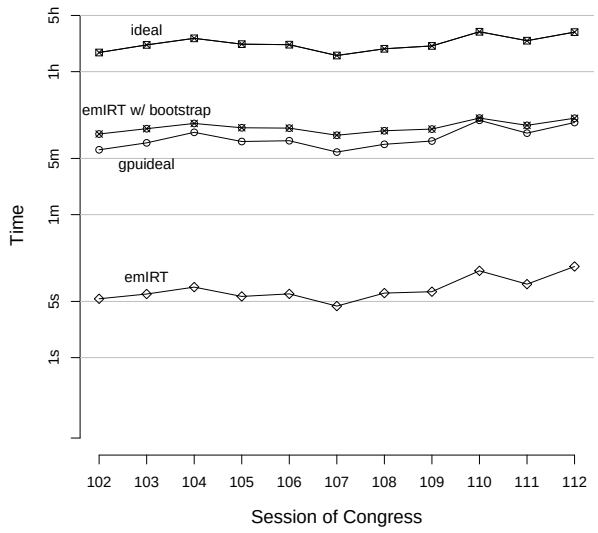


Figure 1: Speed comparison of software for estimating ideal points.

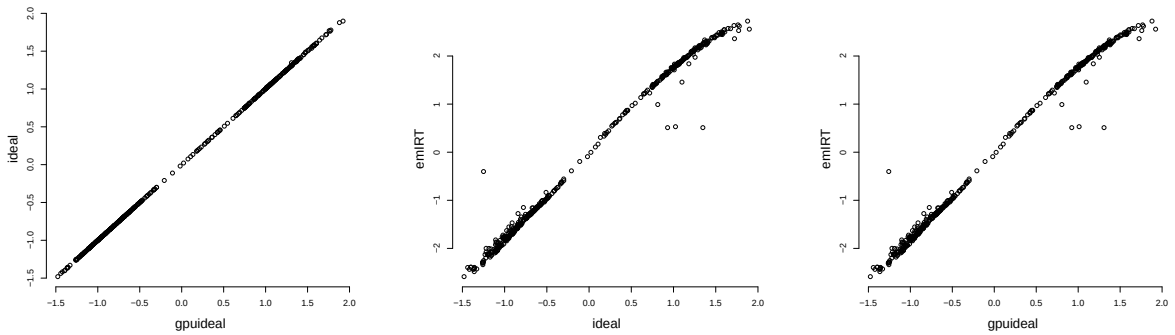
Figure 1 shows the test results, closely mirroring the left panel of Figure 1 from Imai, Lo, and Olmsted (2016). The y-axis shows the time elapsed for each analysis on the log scale, with some points of reference labelled. Each point on the x-axis is a session of the House of Representatives, and estimates are based on roll call voting data for that session. There is some variation in the number of bills in each session, leading to some variation in estimation time.

emIRT is by far the fastest method in terms of performance, often running in almost as little as 5 seconds. However if any inference is to be performed, the bootstrap version is necessary. Without any parallelization, this scales roughly linearly, with at least 500 seconds needed for 100 bootstrap replicates. **gpuideal** is slightly faster in every case. It is dramatically faster than **ideal**, which typically takes several hours, while retaining all of the advantages in terms of inference.

6 Accuracy

Each of these models is practically identical in terms of the estimates produced, with one caveat. Expectation Maximization is meant to produce the maximum likelihood estimate, which is analogous to the posterior mode or Maximum A Posteriori (MAP) in a Bayesian setting. In contrast, many Bayesian analyses report the Expected A Posteriori (EAP), which is the posterior mean. Ideal point estimation can produce very skewed posterior distributions when the informativeness of the items varies substantially across the latent space. This means that is some cases the MAP and the EAP can be substantially different. In such cases a simple standard error will not capture the nature of the uncertainty in the posterior distribution.

Figure 2 compares the estimates produced by each method for the 112th House of Representatives. In every case, the results produced by each method are very highly correlated with the results produced by the others. The correlation between the results of **ideal** and **gpuideal** is 0.999995, reflected in the almost perfect line in the leftmost panel of the Figure. The estimates produced by **emIRT** are also very highly correlated with **ideal**, but behind the .995 correlation there are perhaps a half dozen legislators whose MAP and EAPs are substantially different. The same is true of the comparison between **emIRT** and **gpuideal**. Other sessions of Congress produce very similar results.



gpuideal v. ideal ideal v. emIRT gpuideal v. emIRT

Figure 2: Comparison of ideal point estimates produced by **ideal**, **emIRT** and **gpuideal.**

7 Implementation

gpuideal is easily installed and run as a library in the **R** programming language. It is available for download at `https://github.com/JeffreyBLewis/gpuideal` and can be installed using the `install_github` function from the **devtools** package as shown in the example below. As noted earlier, **gpuideal** requires an NVIDIA graphics card. For researchers who do not have access to one, a simple solution is use Amazon's EC2 service to rent a GPU-enabled server instance. For step-by-step instructions, see the **gpuideal** README file.

The following code snippet demonstrates how to run **gpuideal** on the 112th House of Representatives and summarize the results:

```
# Install package
devtools::install_github("jeffreyblewis/gpuideal")

# load libraries
library(gpuideal)
library(pscl)

# read in the 112th congress
data <- readKH("https://voteview.com/static/data/out/votes/H112_votes.ord")

# run gpuideal
results <- gpuideal(data,thin=10,samples=120000)
```

```
13
14 # post process using party to establish direction
15 rescaled_results <- rescaleIdeal(results,dir=data$legis.data$partyCode)
16
17 # summarize the results
18 results_summary <- summary(rescaled_results)
19
20 # histogram of the estimated ideal points
21 hist(results_summary$statistics[1:data$n,1])
22
23 # histogram of the posterior distribution of Paul Ryan's ideal point
24 hist(rescaled_results[20000:120000,438])
```

References

Albert, James H, and Siddhartha Chib. 1993. "Bayesian analysis of binary and polychotomous response data." Journal of the American statistical Association 88(422): 669–679.

Clinton, Joshua, Simon Jackman, and Douglas Rivers. 2004. "The Statistical Analysis of Roll Call Data." American Political Science Review 98(2): 355?370.

Herlihy, Maurice, and Nir Shavit. 2011. The art of multiprocessor programming. Morgan Kaufmann.

Hill, Seth J, and Chris Tausanovitch. 2015. "A disconnect in representation? Comparison of trends in congressional and public polarization." The Journal of Politics 77(4): 1058–1075.

Imai, Kosuke, James Lo, and Jonathan Olmsted. 2016. "Fast estimation of ideal points with massive data." American Political Science Review 110(4): 631–656.

Jackman, Simon. 2009. Bayesian Analysis for the Social Sciences. John Wiley & Sons.

Jackman, Simon. 2017. "pscl: Classes and methods for R. Developed in the Political Science Computational Laboratory, Stanford University. Department of Political Science, Stanford University, Stanford, CA. R package version 1.5.2." http://github.com/atahk/pscl .

Martin, Andrew D, Kevin M Quinn, and Jong Hee Park. 2011. "MCMCpack: Markov Chain Monte Carlo in R." Journal of Statistical Software 42(9).

R Core Team. 2018. R: A Language and Environment for Statistical Computing. Vienna, Austria: R Foundation for Statistical Computing.