ST 790: Advanced Computing Final Report

Brian Naughton

April 29, 2015

1 Introduction

There are many applications in statistics, econometrics and the social sciences that require sampling from the truncated normal distribution. Such applications include isotonic or order-restricted regressions, censored data models, Bayesian linear modeling subject to linear constraints, and multinomial probit models to name a few (Robert, 1995; Geweke, 1996; Yu, 2000). In particular, one of my current research topics is modeling ranking data. Observed rankings are assumed to be induced by latent variables from a multivariate (normal) distribution. The order statistics of these variables then determine the rankings. To estimate parameters of the distribution, it is useful to sample from truncated multivariate normal distributions in a Bayesian model, where the observed rankings determine the order of the variables, and thus the truncation regions.

This Final Project creates a `Julia` package called `TruncatedMVN` to sample from truncated univariate and multivariate normal distributions. There currently are no available methods for sampling from the truncated multivariate normal (TMVN) distribution in `Julia`, and there is only a crude sampler available in the `Distributions` package to sample from the truncated univariate normal (TUVN) distribution. This new package implements the recent work of Li and Ghosh (2015) which extends previous methods developed by Geweke (1991) and Robert (1995).

The format of this paper is defined as follows. In Section 2, I describe the methods developed by Li and Ghosh (2015) that are implemented by this new package. I first explain how to efficiently sample from the TUVN distribution, and then I demonstrate how to sample from the TMVN using a Gibbs sampling procedure. In Section 3, I show the superiority of this new method compared to current and available implementations in `R` and `Julia`. Primary considerations will be numerical stability of the algorithm, computational speed and good mixing of the samples from the Gibbs sampler. I end with a Discussion in Section 5 regarding future work, applications and use of the package.

All of the code required to carry out the simulations is accompanied with this paper in `*.Rmd` and `*.ipynb` files which are explained later in the paper. The `TruncatedMVN` package and associated code is publicly available through my primary Github page at `https://github.com/bnaught/TruncatedMVN.jl`. It can be installed in Julia by executing the command:

```
Pkg.clone("git://github.com/bnaught/TruncatedMVN.jl.git").
```

2 Methods

2.1 Efficient sampling from the Truncated Univariate Normal Distribution

In this section I discuss the motivation for needing an efficient sampler for the TUVN distribution, and describe the algorithm developed by Li and Ghosh (2015). The density of a TUVN random variable with mean μ and variance σ^2 restricted to the interval $[c, d]$ is given by:

$$f_W(w) = \frac{\frac{1}{\sigma}\phi(\frac{w-\mu}{\sigma})}{\Phi(\frac{d-\mu}{\sigma}) - \Phi(\frac{c-\mu}{\sigma})} I_{\{c<w<d\}}, \tag{1}$$

where c and d are the possibly infinite boundaries, ϕ is the standard normal PDF, and Φ is the CDF. To generate samples from this distribution is equivalent to sampling x from the truncated standard normal distribution ($\mu = 0, \sigma^2 = 1$) restricted to the region $[a, b]$, and convert back using the transformations $w = \sigma x + \mu$, $a = (c - \mu)/\sigma$, and $b = (d - \mu)/\sigma$. Therefore, without loss of generality we only need to be concerned with sampling from the standard TUVN distribution.

A naive approach would be to just sample $x \sim N(0, 1)$, and keep the result if it falls in the truncated region. This rejection sampling method would be very inefficient for regions away from 0, and virtually impossible for extreme regions, e.g. $x \in [35, \infty]$. Another popular approach is to use the inverse CDF method. Since we can write down the distribution function (F), and inverse CDF (F^{-1}), of the TUVN distribution, we simply need to generate a uniform(0, 1) random variable, u, and set $x = F^{-1}(u)$. However, this method is not numerically stable for extreme regions, and will be explored further in Section 3. Geweke (1991) and Robert (1995) propose rejection sampling using other envelope distributions depending on the region of truncation. This new method combines both approaches to find the optimal regions to use the different envelope densities, and is also quite intuitive.

A sketch of the so-called Ensemble Algorithm by Li and Ghosh (2015) is laid out as follows. In large regions around the mean, just use the naive rejection sampling from the normal distribution. If the region is close to the mean with $a > 0$ or $b < 0$, use half-normal rejection sampling, i.e., generate $x \sim N(0, 1)$ and keep the sample if $|x| \in [a, b]$. For relatively "flat" regions, such as small regions around the mean, use rejection sampling with a uniform envelope function. Finally for regions in the tails, use a translated-exponential density envelope function to approximate the Gaussian tails. Li and Ghosh (2015) give exact boundaries for when to use each of the four algorithms.

This Ensemble Algorithm was implemented with the help of the `Distributions` package. For boundary points, a and b, the function `TruncatedNormalSampler` determines which sampling method to use and creates a `Sampleable` type object. The built-in generic function `rand()` can then be used on this object to sample from the TUVN distribution.

2.2 Efficient sampling from the Truncated Multivariate Normal Distribution

The TMVN distribution can be defined most generally as a multivariate normal distribution subject to linear constraints. If $\boldsymbol{w} \sim TMVN(\boldsymbol{\mu}, \Sigma, \tilde{\boldsymbol{R}}, \boldsymbol{c}, \boldsymbol{d})$ then $\boldsymbol{w}$ is p-dimensional multivariate normal with mean $\boldsymbol{\mu}$ and variance matrix Σ subject to the linear constraints $\boldsymbol{c} \preceq \tilde{\boldsymbol{R}}\boldsymbol{w} \preceq \boldsymbol{d}$, where "$\preceq$" means element-wise inequality, and $\tilde{\boldsymbol{R}}$ is an $m \times p$ matrix of constraints. That is, $\boldsymbol{w}$ has multivariate normal distribution restricted to a convex polyhedron.

Rejection sampling is difficult for sampling from the TMVN because of the inability to define a good envelope function for all truncation regions. Calculating the normalizing constant in the PDF can be difficult as well. The preferred approach is to use a Gibbs sampler by drawing $w_i | \boldsymbol{w}_{-i}$ iteratively to approximate the distribution of $\boldsymbol{w}$ (Geweke, 1991; Gelfand et al., 1992). It is worth noting that this sampler is not generating independent samples from this distribution. Thus, care must be taken to investigate mixing properties as well as autocorrelation of the samples generated. However, Li and Ghosh (2015) point out that as long as the initial value is a feasible point from this distribution, then all samples will be part of the convex polyhedron defining the support of $\boldsymbol{w}$.

The key component of this new method is that $\boldsymbol{w}$ is reparameterized such that the covariance matrix is the identity. That is, $\boldsymbol{x} = \Sigma^{-1/2}(\boldsymbol{w} - \mu)$, where $\Sigma^{-1/2}$ is the inverse of $\Sigma^{1/2}$, which is the lower triangular matrix of the Cholesky decomposition of Σ. This induces $\boldsymbol{x} \sim TMVN(0, \boldsymbol{I}, \boldsymbol{R}, \boldsymbol{a}, \boldsymbol{b})$, where $\boldsymbol{a} = \boldsymbol{c} - \tilde{\boldsymbol{R}}\mu$, $\boldsymbol{b} = \boldsymbol{d} - \tilde{\boldsymbol{R}}\mu$, and $\boldsymbol{R} = \tilde{\boldsymbol{R}}\Sigma^{1/2}$. Then, the conditional variance does not need to be computed for the full conditional distribution of each x_i. That is, the distribution of $x_i|\boldsymbol{x}_{-i}$ is a standard truncated univariate normal distribution, restrained to the region $[a^*(\boldsymbol{x}_{-i}), b^*(\boldsymbol{x}_{-i})]$. The truncation points a^* and b^* are determined such that $\boldsymbol{a} \preceq \boldsymbol{R}\boldsymbol{x} \preceq \boldsymbol{b}$, and Li and Ghosh (2015) give an explicit algorithm for finding them. The computation at each iteration of this algorithm is very cheap since the inversion of triangular matrix needs only to be made once, and involves only a matrix-vector multiplication at each iteration. This is in contrast to previous methods that required the conditional variance to be computed at each step using methods such as sweeping.

This algorithm is implemented in the new package for `Julia`. The user specifies the mean, covariance matrix, and bounds to the function `GibbsTMVN`, and it returns a matrix of samples from the approximated distribution. Optionally they can also supply the number of MCMC iterations(default is 10,000), and a starting value of $\boldsymbol{w}$ for the Gibbs sampler. If a starting value is not supplied, the first step is to find a feasible point in the support to start from. This is simply a feasibility problem that can be solved with a linear programming solver. The algorithm uses the SCS solver with `Convex.jl` to find a starting value.

3 Simulations

In this section I investigate the performance of the new package compared to other available packages in `R` and `Julia`. First, I compare the numerical stability and computational speed of the new TUVN sampler to what is available in the `Distributions` package. Next, I compare the performance of the TMVN sampler to a sampler that is currently available in the `R` package `tmvtnorm`.

The `Distributions` package contains a generic method for creating truncated distributions from any standard distribution which can be used to sample from. However, it uses the inverse CDF method described previously which is not numerically stable. For example, to sample from the TUVN distribution constrained to the interval $[9, \infty]$ it first needs to evaluate $\Phi(9)$ and $\Phi(\infty)$. Clearly, $\Phi(\infty) = 1$, but due to numerical round-off errors, $\Phi(9) = 1$ as well. Although in theory this method is correct, computationally the sampler will always return ∞. However, with the new sampler, extreme values such as this are generated from a translated-exponential distribution to approximate the tail behavior, and are always finite.

The acceptance rate for the inverse CDF method is always 1, but is usually less than one in the new sampler. Therefore, the new method is necessarily slower than the current implementation. However, the timing difference is inconsequential when the sampler gives unrealistic values. Table 1 shows the timing results for 100,000 samples, and how they might change depending on the sampler required for that given region. The same simulation was timed in `R` using the `truncnorm` package, which is a wrapper for Geweke's algorithm written `C`. The simulation shows that new sampler takes 2-7 times longer to run than the generic sampler. It is still very fast though, taking about two-hundreths of a second to take 100,000 samples. It also shows the power of Julia and the efficiency of the algorithm since it still 1-3 times faster than the `C` implementation called by `R`. The timing results were recorded on a laptop running Ubuntu Linux (14.04) with an Intel Core i7-2670QM CPU @ 2.20GHz. The IJulia notebook, `CompareTUVN.ipynb`, and the Rmarkdown file, `CompareTUVN_R.Rmd`, carry out these simulations, and are accompanied with this report along with the html output files from my machine.

	Rejection sampling method			
Package	NR	HR	UR	ER
Distributions.jl	0.0032	0.0052	0.0036	0.0068
TruncatedMVN.jl	0.0080	0.0100	0.0229	0.0173
truncnorm (in `R`)	0.023	0.023	0.026	0.029

Table 1: Time benchmarks (in seconds) to sample 100,000 samples from a truncated univariate standard normal distribution using different packages. Times were compared for 4 different sampling methods: Normal rejection (NR), Half-Normal rejection (HR), Uniform rejection (UR) and translated exponential rejection (ER) sampling.

Next I investigate the performance of the TMVN Gibbs sampler to an implementation in the `R` package `tmvtnorm`. That package implements the Gibbs sampling method proposed by Geweke (1991) in two ways: an `R` version and a Fortran version. Based on Example 2 in Li and Ghosh (2015), I consider the 3-dimensional case with

$$\boldsymbol{\mu} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}, \quad \Sigma = \begin{pmatrix} 1 & 0.99 & 0.98 \\ 0.99 & 1 & 0.99 \\ 0.98 & 0.99 & 1 \end{pmatrix}, \quad \tilde{\boldsymbol{R}} = \begin{pmatrix} 1 & -2 & 0 \\ -1 & 0 & 0 \end{pmatrix}, \quad \boldsymbol{c} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}, \quad \text{and} \quad \boldsymbol{d} = \begin{pmatrix} 1 \\ 2 \end{pmatrix}.$$

The high correlation structure will help differentiate the new method from the previously available methods. Each Gibbs sampler was ran for 10,000 and 100,000 iterations. Table 2 displays the timing results for the new sampler against the two implementations available in `tmvtnorm`. The new package is about twice as slow as the Fortran implementation, but almost 20 times faster than the pure `R` implementation. Since these are not independent samples from the TMVN distribution we must also investigate the mixing and convergence properties of the sampler.

Figure 1 shows the traceplots of the samples of w_1 from both packages. There does not appear to be a distinct pattern, which shows that the samples are well mixed in both packages. Figure 2 shows the estimated densities via histograms of those same samples. These show that both packages are giving similar results for the first variable, and are consistent with the original bounds of the problem, i.e. $w_i \in [-2, 0]$. The traceplots for w_2 and w_3 also showed similarities between the two methods. Finally, we must investigate how correlated the samples are. Figure 3 shows autocorrelation plots for the samples of w_1, and there is a striking difference between the two methods. Geweke's method implemented in `tmvtnorm` produces highly correlated samples – the autocorrelation is above 0.1 until lag 10. On the other hand, the autocorrelation drops below 0.1 by lag 2 when using Li and Ghosh's method. This is most likely due to the reparameterization to work with uncorrelated variables. The autocorrelation plots for w_2 and w_3 showed similar behavior. Although the Geweke's method implemented in Fortran is faster, it is possible that this new method produces independent samples faster.

The `R` and `Julia` code used to carry out these simulations and create the plots is available in the associated `CompareTMVN_R.Rmd` and `CompareTMVN.ipynb` files respectively. The html output from this Rmarkdown file and notebook is accompanied with this report as well.

	Iterations	
Package	10,000	100,000
TruncatedMVN.jl	0.020	0.204
tmvtnorm (in `R`)	0.485	4.774
tmvtnorm (in Fortran)	0.015	0.097

Table 2: Time benchmarks (in seconds) to sample from the truncated multivariate normal distribution using a Gibbs sampler in different packages.

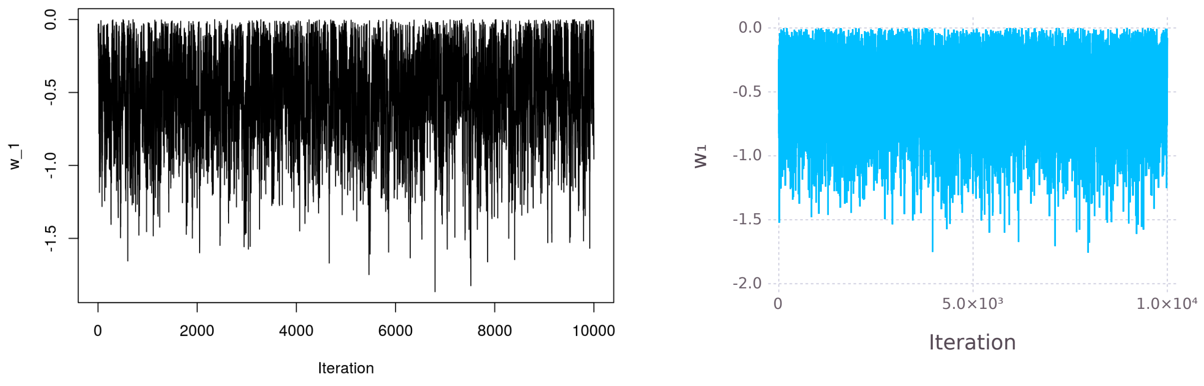

Figure 1: Traceplots of the 10,000 samples of w_1 from the tmvtnorm package (left) and TruncatedMVN.jl (right).

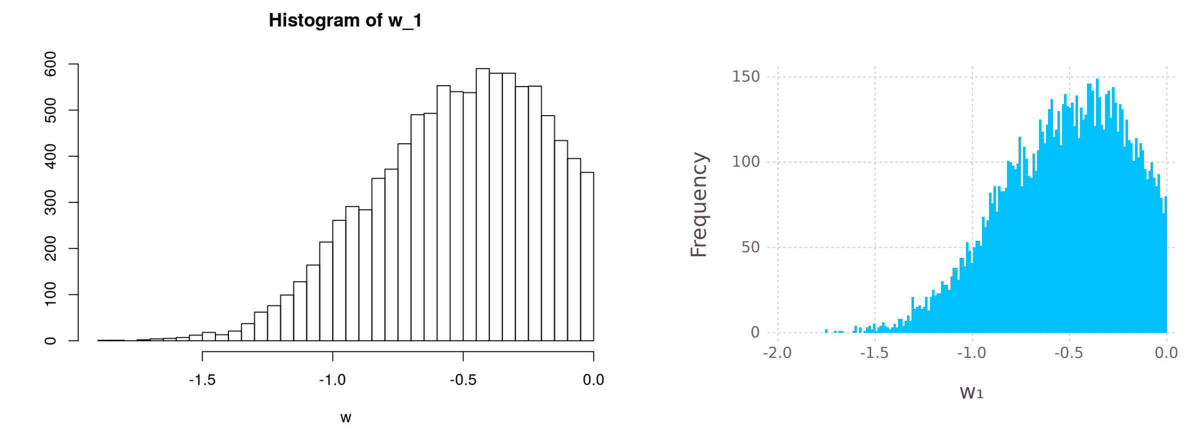

Figure 2: Histograms of the 10,000 samples of w_1 from the tmvtnorm package (left) and TruncatedMVN.jl (right).

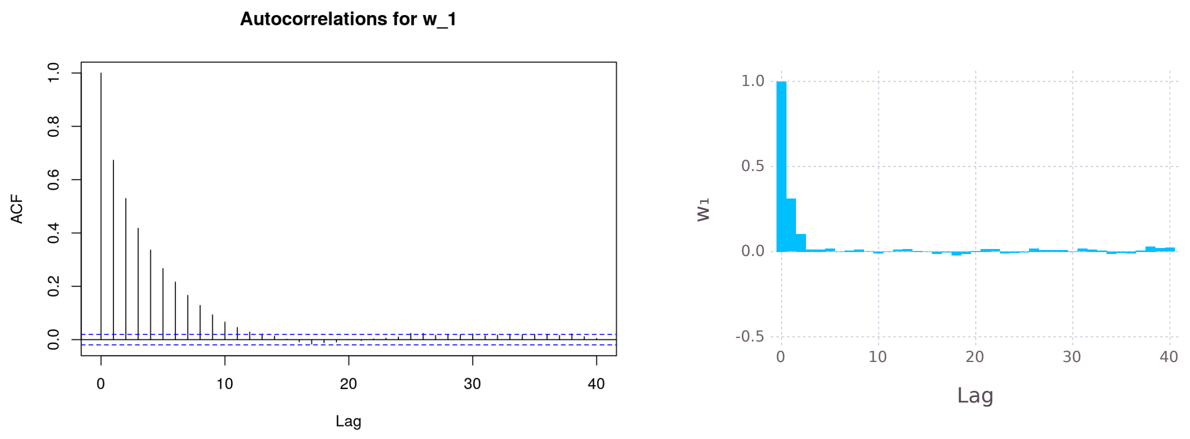

Figure 3: Autocorrelation plots of the 10,000 samples of w_1 from the tmvtnorm package (left) and TruncatedMVN.jl (right).

4 Discussion

For this final project I created a Julia package to efficiently sample from the TUNV distribution, and sample from the TMVN distribution via Gibbs sampling. I have showed that the new implementation rivals previously available methods in terms of the quality of samples generated and computational speed. It gets close to matching the speed of algorithms written in lower level `C` or Fortran code. In cases where it is slower, the samples are more reliable, or less correlated in the Gibbs sampling case.

For future work on this problem, I first intend to fork the `Distributions` package, and implement the new truncated univariate sampler. My goal is to convince the maintainers to incorporate my changes into their package so that the sampler is more reliable for extreme regions. There is also more work to be done in the univariate case in terms of making the code more efficient, so that it can run almost as fast as the current sampler. Next, I plan to continue testing the package and implement a truncated multivariate Students-t distribution that Li and Ghosh (2015) develop methods for as well. I will develop some suitable tests to ensure the package is working as it is intended, and then distribute the package so that it is available directly through Julia. Since this sampler is much more specialized and is actually a Gibbs sampler, it may be misleading to make available through the `Distributions` package.

References

Alan E Gelfand, Adrian FM Smith, and Tai-Ming Lee. Bayesian analysis of constrained parameter and truncated data problems using gibbs sampling. *Journal of the American Statistical Association*, 87(418):523–532, 1992.

John Geweke. Efficient simulation from the multivariate normal and student-t distributions subject to linear constraints and the evaluation of constraint probabilities. In *Computing science and statistics: Proceedings of the 23rd symposium on the interface*, pages 571–578. Citeseer, 1991.

John F Geweke. Bayesian inference for linear models subject to linear inequality constraints. In *Modelling and Prediction Honoring Seymour Geisser*, pages 248–263. Springer, 1996.

Yifang Li and Sujit Ghosh. Efficient sampling methods for truncated multivariate normal and student-t distributions subject to linear inequality constraints. *Journal of Statistical Theory and Practice*, 2015.

Christian P Robert. Simulation of truncated normal variables. *Statistics and computing*, 5(2):121–125, 1995.

Philip L.H. Yu. Bayesian analysis of order-statistics models for ranking data. *Psychometrika*, 65(3):281–299, 2000.