Replication Note for *A Bayesian Poisson Vector Autoregression Model*[*]

Patrick T. Brandt[†]
pbrandt@utdallas.edu

Todd Sandler
tsandler@utdallas.edu
School of Economic, Political and Policy Sciences
The University of Texas, Dallas
800 W. Campbell Rd., GR 31
Richardson, TX 75080

January 17, 2012

Here we provide the replication materials for our article, "A Bayesian Poisson Vector Autoregression Model." The replication code is organized into two folders, one for each of the examples in the paper:

superpower/ Replication files for the superpower rivalry application,

targeting/ Replication files for the terrorist targeting appplication.

For both applications we have also included the dataset analyzed and the `R` and `JAGS` scripts used in the analysis, and output logs from the `R` runs. We have omitted the interim files that hold the Markov chain Monte Carlo (MCMC) output for the sake of space (these can be reproduced by running the scripts included here — random number seeds are set thoughout for reproducibility).

`R` version 2.14.0 and `JAGS` version 2.2.0 were used to conduct the analysis. The files and descriptions listed below work with these versions. Your mileage will vary with future versions. The `JAGS` code has not been tested in `BUGS` as of January 2012.

[*]This study was funded by the US Department of Homeland Security (DHS) through the Center for Risk and Economic Analysis of Terrorism Events (CREATE) at the University of Southern California, Grant 2007-ST-061-000001 and 2010-ST-061-RE0001. Brandt's research is based upon work supported by the National Science Foundation under Award Number 0921051. However, any opinions, findings, conclusions, and recommendations are solely those of the authors and do not necessarily reflect the views of DHS, CREATE or the National Science Foundation.

[†]Corresponding author.

In several places in the code we have used parallel processing with the `R snow` package. This was done for analyzing the multiple datasets with multiple MCMC chains or for other diagnostics.

There are both single and multicore versions of the code. Given the large number of models fit (e.g., for the terrorist targeting application there are 5 data subsets and 3 different lag specifications considered), some of the models are fit using the `snow R` package so that we can process different models in parallel on multiple cores. If you do not have a multicore processor and a working version of `R` 's `snow` package, you will have to run this code serially. If you have more or fewer cores, you will want to adjust the number of cores employed for your environment. Please do not contact me with questions about setting up and using `snow` in `R` , since there are ample resources available online (e.g., `rseek.org`).

In what follows, files with a `.R` file extension are `R` input scripts, with a `.Rout` file extension are `R` output files, and anything with a `.bug` extension are `JAGS` model code. All code is heavily annotated with descriptions of what is being done.

Superpower Rivalry Application

The replication of the superpower rivalry example in section 4 of the article uses the following files. These are presented in the order in which they need to be used or run, since each one generates output that is necessary for subsequent `R` scripts. The assumption in setting up these files for a run is that they are all in the same folder.

`dyadyr.dta` **and** `dyadyr.txt` The dataset, from King (1989), in Stata or text format.

`MVPLN-superpower.R` This is the simplest version of the BaP-VAR estimator. It sets up the data from the dataset files `dyadyr`, generates Figure 1, and estimates the BaP-VAR models. Examples in the file show how to set up the priors for each of the BaP-VAR(p) models. The conclusion of the script generates the DIC statistics and checks posterior convergence with Gelman-Rubin diagnostics. This file is meant to be a quick pedagogical example of how to employ and setup the model.

`MVPLN-VAR-*.bug` These are the `JAGS` model code, *specific to this application.* The number in the filename corresponds to the value of p, the lag length in the BaP-VAR(p) specification. The initial lines of the file read in the data dumped out in a proper format from the last script. The model is then defined in standard `JAGS` or `BUGS` notation. Note that the prior for the coefficients is set in the ending lines of code in the file.

`BAPVAR-superpower-multicore.R` This is the main `R` script used to generate the posterior output reported in section 4 of the paper. This script does all of the data setup from the `MVPLN-superpower.R` script. In this file, however, the models are fit using multiple processors or cores. Different models (values of p) are deployed on different cores. Each core does the MCMC simulation for a given posterior, saves the results, and estimates the DIC for that specification. The setup in this file uses three cores, on each for $p = 1, 2, 3$. The computational cluster is set up using the load-balancing functions in `snow` for optimal parallel computation. The only output beyond saving the posterior samples is to return the posterior DIC statistics.

`BAPVAR-superpower-diagnostics.R` Uses the posterior sample output from the previous file to generate summary statistics for the posterior parameters and the Gelman-Rubin diagnostics (Brooks and Gelman, 1998) on convergence.

`BAPVAR-superpower-dynamics.R` This file computes the dynamic responses or impulse response functions (IRFs) for the models. The file contains a number of special, model specific functions to compute and map moments. Some of these are model specific and will need to be modified for other applications. These are documented at the beginning of the script. This file produces the output in Figure 2 and the BaP-VAR results in Table 2 of the paper. The plotting method for the impulse responses is the same one employed in the `MSBVAR R` package. The code here employs several versions of error bands for impulse responses, as documented in Brandt and Freeman (2006).

`BAPVAR-superpower-multiplers.R` Computes the dynamic or impact multiplers for changes in the *exogenous* covariates, as reported in Figure 3 of the article. The `BaPVAR.multiplier` function in this file is generate and can be used in other conforming applications. Note that this file used parallel processing across the number of covariates, since for a fixed number of k covariates for a given posterior sample of size N, this is a parallel processing problem. The plot at the end of the code generates Figure 3.

`VAR-superpower.R R` script that generates the Gaussian VAR results in Table 2 and footnote 9 of the article using the `MSBVAR R` package.

Terrorist Targeting Application

Replication of the terrorist targeting example in section 5 of the article uses the following files. These are presented in the order in which they need to be used or run, since each one generates output that is necessary for subsequent `R` scripts. The assumption in setting up these files for a run is that they are all in the same folder.

`MonthlySeries1968-2008.RData` The raw data in `R` format used in the analysis (if you are using MS-Windows, rename the file with the extension `.rda` so your OS recognizes it — but this will not work necessarily with the rest of the code). Since ITERATE (Mickolus et al., 2009) is a proprietary database, we cannot provide the original, raw, event data used to construct the time series. Here we have aggregated the monthly totals from 1968–2008 as `zoo R` package objects of time series for each target type. The included `setup.R` script illustrates how to get the aggregated data from a CSV version of the original ITERATE data using R.

`MVPLN-VAR-monthly.R R` script that does the main posterior simulation of the analysis of the monthly terrorist targeting data. This script does a series of things:

- Subsets the data into the five sub-samples, `y1,...,y5`.
- Sets up the initial conditions for each sample's posterior simulation and dumps these for use in each `JAGS` run / model.

- Defines a function that will estimate a model for each sub-sample, lag length, and number of chains. This is how `JAGS` is called from `R` using the `rjags` package. The `MVPLNmodel` function in this file writes out the main results and only returns the model fit, or DIC. As interim output, this code outputs the results for each sub-sample and lag length specifcation (see the `MVPLNmodel` function and its calls for more details).

`MVPLN-VAR-*.bug` These are the files where the * differentiates *only* a number. These are the `JAGS` model code, *specific to this application for the separate analyses, based on lag length*. The number in the filename corresponds to the value of p, the lag length in the BaP-VAR(p) specification. The initial lines of the file read in the data dumped out in a proper format from the last script. The model is then defined in standard `JAGS` or `BUGS` notation. Note that the prior for the coefficients is set in the final lines of code in the file.

`MVPLN-VAR-monthly-DIC.R` Computes the DIC's for each of the fitted models for this application and reports them in a table like Table 3 of the article.

`MVPLN-VAR-monthly-posterior.R` Uses the posterior sample output from the previous file to generate summary statistics for the posterior parameters and the Gelman-Rubin diagnostics on convergence.

`MVPLN-VAR-monthly-dynamics.R` This file computes the dynamic responses or impulse response functions (IRFs) for the models. The file contains a number of special, model specific functions to compute and map moments. Some of these are model specific and will need to be modified for other applications. These are documented at the beginning of the script. This file produces the output in Figures 4–7, those omitted after Figure 7 for the last sub-sample, and the BaP-VAR results in Table 4 of the paper (these are in the output, but hand-edited to make the reported results). The plotting method for the impulse responses is the same one employed in the `MSBVAR R` package. The code here employs several versions of error bands for impulse responses, as documented in Brandt and Freeman (2006).

`MVPLN-VAR-dfev.R` Uses the MCMC results of the last script to compute the decompositions of the forecast error variance (DFEV) for the BaP-VAR posterior for this application. This generates the results for Figure 8 in the paper, based on the last script and the `MSBVAR R` package. Some of this code is specific to the application, but can be generalized.

`MVPLN-VAR-full.R` What if you analyze the complete monthly 1968–2008 sample? Included here. This script does the analysis noted above for the full sample. Does the same analysis as above with similar results. This generates the results for the claims in footnote 15.

`MVPLN-VAR-full-*.bug` Same as the earlier `*.bug` files, but adapted for the full sample, rather than the 5 subsets of the dataset.

`MVPLN-VAR-full-dynamics.R` Generates the IRFs for the full sample analysis using the BaP-VAR(1) results from the last two scripts.

`MVPLN-VAR-full-posterior.R` Uses the posterior sample output from the `MVPLN-VAR-full.R` script to generate summary statistics for the posterior parameters and the Gelman-Rubin diagnostics on convergence.

`MVPLN-Gaussian-VAR.R` This is the Gaussian-VAR analysis of the data in this application. It estimates the Gaussian VARs using the `MSBVAR` `R` package.

Final Notes

A few final comments if you are trying to implement your own BaP-VAR(p) model:

1. Start small. Replicate the superpower rivalry example first to make sure you have all of the software (`R` , `JAGS` , `rjags`, and related packages) working properly.

2. Tips on setting the prior for the BaP-VAR coefficients. Remember, the coefficients are in the log-normal space and the outcomes of interest are in the observable count space. So you need to map your priors for the intercept from the observable space into the log-normal space. The easiest way to do this is compute the sample means and sample covariance of your data and then take the natural logs as an approximation.

3. Setting the priors on the AR coefficients. Remember, the suggestion in the paper is to use a Sims-Zha prior (Sims and Zha, 1998) where $A_1 = I$, an identity matrix and $A_\ell = 0$ for $\ell = 2, \ldots p$. The precisions should shrink these coefficients toward zero as the lag length increases. For example, if you set the prior precision for the A_1 coefficients to be 10, then to have harmonic lag decay the prior precision for A_2 should be 20, for A_3, 30, etc.

References

Brandt, P. T. and J. R. Freeman (2006). Advances in Bayesian time series modeling and the study of politics: Theory testing, forecasting, and policy analysis. *Political Analysis 14* (1), 1–36.

Brooks, S. and A. Gelman (1998). General methods for monitoring convergence of iterative simulations. *Journal of Computational and Graphical Statistics 7*, 434–455.

King, G. (1989). A seemingly unrelated Poisson regression model. *Sociological Methods & Research 17*(3), 235–255.

Mickolus, E. F., T. Sandler, J. M. Murdock, and P. Flemming (2009). *International Terrorism: Attributes of Terrorist Events, 1968-2008 (ITERATE)*. Dunn Loring, VA: Vinyard Software.

Sims, C. A. and T. A. Zha (1998). Bayesian methods for dynamic multivariate models. *International Economic Review 39*(4), 949–968.