BigVAR User's Guide

Dimension Reduction Procedures for Multivariate Time Series

William Nicholson

wbn8@cornell.edu

September 25, 2015

1 Overview

The R package `BigVAR` allows for the simultaneous estimation of high-dimensional time series by applying structured penalties to the conventional vector autoregression (VAR) and vector autoregression with exogenous variables (VARX) frameworks. Our methods can be utilized in many applications that make use of time-dependent data, such as macroeconomics, finance, and internet traffic.

Our package adapts solution algorithms from the regularization literature to a multivariate time series setting, allowing for the computationally efficient estimation of high-dimensional VARs with the option to additionally incorporate unmodeled exogenous series.

The conventional VAR and VARX are heavily overparameterized, often forcing practitioners to arbitrarily specify a reduced subset of series to model. `BigVAR` allows for the incorporation of many potentially useful series by imposing zero restrictions in a data-driven manner. Moreover, if forecasts are only desired from a subset of included series, using the VARX-L framework (Nicholson et al. [2015]), `BigVAR` can effectively leverage the information from the unmodeled *exogenous* series to improve the forecasts of the modeled *endogenous* series. We additionally offer a class of Hierarchical Vector Autoregression (HVAR) procedures (Nicholson et al. [2014]) that address the issue of lag order selection in VAR models.

This document presents a brief overview of our notation, the models that comprise `BigVAR`, and the functionality of the package. For more detail, we refer you to the original papers. Section 2 provides a brief overview of the notation, the VARX-L penalties presented in Nicholson et al. [2015] and the HVAR methodology detailed in Nicholson et al. [2014], and Section 3 details practical implementation of `BigVAR` with a macroeconomic data example.

If you have questions about the package's functionality or feature requests, please feel free to contact me at wbn8@cornell.edu.

2 Notation and Overview of BigVAR Procedures

Let $\{\mathbf{y_t}\}_{t=1}^T$ denote a k dimensional vector time series and $\{\mathbf{x}_t\}_{t=1}^T$ denote an m-dimensional unmodeled *exogenous* series. A vector autoregression with exogenous variables of order (p,s) (VAR$_{k,m}(p,s)$) can be expressed as

$$\mathbf{y}_t = \boldsymbol{\nu} + \sum_{\ell=1}^{p} \boldsymbol{\Phi}^{(\ell)} \mathbf{y}_{t-\ell} + \sum_{j=1}^{s} \boldsymbol{\beta}^{(j)} \mathbf{x}_{t-j} + \mathbf{u}_t \text{ for } t = 1, \ldots, T, \tag{1}$$

in which

- $\boldsymbol{\nu}$ denotes a $k \times 1$ intercept vector
- each $\boldsymbol{\Phi}^{(\ell)}$ represents a $k \times k$ endogenous (modeled) coefficient matrix
- each $\boldsymbol{\beta}^{(j)}$ represents a $k \times m$ exogenous (unmodeled) coefficient matrix
- $\mathbf{u}_t \overset{\text{wn}}{\sim} (\mathbf{0}, \boldsymbol{\Sigma}_u)$ ($\boldsymbol{\Sigma}_u$ is nonsingular, but left unspecified)

Note the the VAR is a special case of Equation 1 in which $m = s = 0$.

Provided k, m, s, and p are small relative to T, one can estimate a VARX$_{k,m}(p,s)$ at time t by least squares via

$$\min_{\boldsymbol{\nu}, \boldsymbol{\Phi}, \boldsymbol{\beta}} \|\mathbf{y}_t - \boldsymbol{\nu} - \sum_{\ell=1}^{p} \boldsymbol{\Phi}^{(\ell)} \mathbf{y}_{t-\ell} - \sum_{j=1}^{s} \boldsymbol{\beta}^{(j)} \mathbf{x}_{t-j}\|_2^2, \tag{2}$$

in which $\boldsymbol{\Phi} = [\boldsymbol{\Phi}^{(1)}, \ldots, \boldsymbol{\Phi}^{(p)}]$, $\boldsymbol{\beta} = [\boldsymbol{\beta}^{(1)}, \ldots, \boldsymbol{\beta}^{(s)}]$.

2.1 The VARX-L Framework

To reduce the parameter space of the VARX we consider applying structured convex penalties to Equation (2), resulting in the objective

$$\min_{\boldsymbol{\nu}, \boldsymbol{\Phi}, \boldsymbol{\beta}} \sum_{t=1}^{T} \|\mathbf{y}_t - \boldsymbol{\nu} - \sum_{\ell=1}^{p} \boldsymbol{\Phi}^{(\ell)} \mathbf{y}_{t-\ell} - \sum_{j=1}^{s} \boldsymbol{\beta}^{(j)} \mathbf{x}_{t-j}\|_2^2 + \lambda \Big(\mathcal{P}_y(\boldsymbol{\Phi}) + \mathcal{P}_x(\boldsymbol{\beta}) \Big), \tag{3}$$

in which

- $\lambda \geq 0$ is a penalty parameter estimated by cross-validation
- $\mathcal{P}_y(\boldsymbol{\Phi})$ represents the group penalty structure on endogenous coefficients
- $\mathcal{P}_x(\boldsymbol{\beta})$ represents the group penalty structure on endogenous coefficients.

These penalties impose structured sparsity based upon a partition of the parameter space that takes into account the intrinsic structure of the VAR. All VARX-L penalty structures are detailed in Table 1 and plots of example sparsity patterns are depicted in Figure 1.

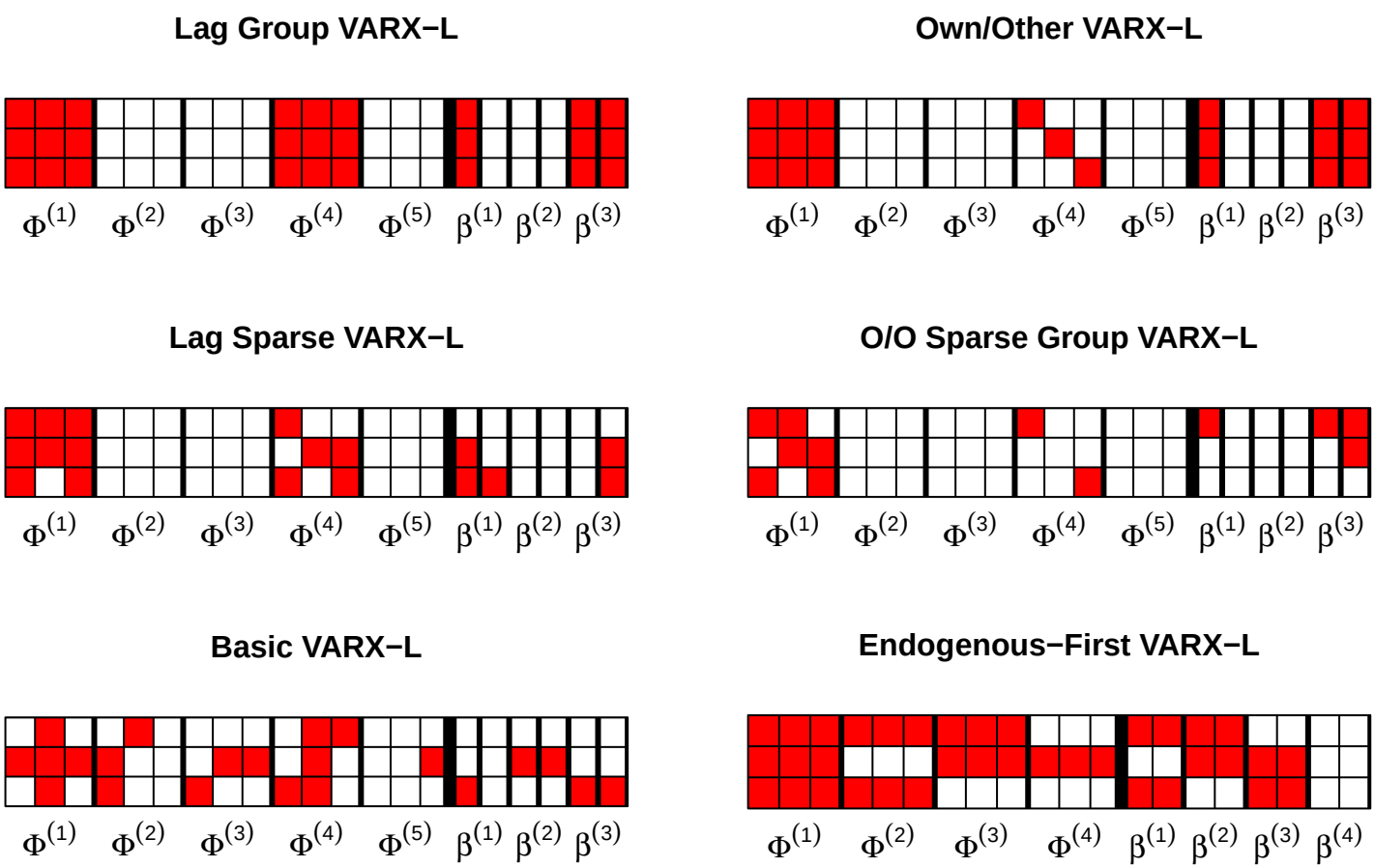


Figure 1: VARX-L Example Sparsity Patterns (k=3, p=5))

Table 1: VARX-L Penalty Functions (Reproduced from Nicholson et al. [2015]). Note that $\mathbf{\Phi}_{\text{on}}^{(\ell)}$ and $\mathbf{\Phi}_{\text{off}}^{(\ell)}$ denote the diagonal and off-diagonal elements of coefficient matrix $\mathbf{\Phi}^{(\ell)}$, respectively.

Group Name	$\mathcal{P}_y(\mathbf{\Phi})$	$\mathcal{P}_x(\boldsymbol{\beta})$
(4) Lag	$\sqrt{k^2}\sum_{\ell=1}^{p}\|\mathbf{\Phi}^{(\ell)}\|_2$	$\sqrt{k}\sum_{j=1}^{s}\sum_{i=1}^{m}\|\boldsymbol{\beta}_{\cdot,i}^{(j)}\|_2$
(5) Own/Other	$\sqrt{k}\sum_{\ell=1}^{p}\|\|\mathbf{\Phi}_{\text{on}}^{(\ell)}\|\|_2+\sqrt{k(k-1)}\sum_{\ell=1}^{p}\|\|\mathbf{\Phi}_{\text{off}}^{(\ell)}\|\|_2$	$\sqrt{k}\sum_{j=1}^{s}\sum_{i=1}^{m}\|\boldsymbol{\beta}_{\cdot,i}^{(j)}\|_2$
(6) Sparse Lag	$(1-\alpha)\sqrt{k^2}\sum_{\ell=1}^{p}\|\mathbf{\Phi}^{(\ell)}\|_2+\alpha\|\mathbf{\Phi}\|_1$	$(1-\alpha)\sqrt{k}\sum_{j=1}^{s}\sum_{i=1}^{m}\|\boldsymbol{\beta}_{\cdot,i}^{(j)}\|_2+\alpha\|\boldsymbol{\beta}\|_1$
(7) Sparse Own/Other	$(1-\alpha)(\sqrt{k}\sum_{\ell=1}^{p}\|\|\mathbf{\Phi}_{\text{on}}^{(\ell)}\|\|_2+\sqrt{k(k-1)}\sum_{\ell=1}^{p}\|\|\mathbf{\Phi}_{\text{off}}^{(\ell)}\|\|_2)+\alpha\|\mathbf{\Phi}\|_1$	$(1-\alpha)\sqrt{k}\sum_{j=1}^{s}\sum_{i=1}^{m}\|\boldsymbol{\beta}_{\cdot,i}^{(j)}\|_2+\alpha\|\boldsymbol{\beta}\|_1$
(8) Basic	$\|\mathbf{\Phi}\|_1$	$\|\boldsymbol{\beta}\|_1$
(9) Endogenous-First	$\mathcal{P}_{y,x}(\mathbf{\Phi},\boldsymbol{\beta})=\sum_{\ell=1}^{p}\sum_{j=1}^{k}\left(\|[\mathbf{\Phi}_{j,\cdot}^{(\ell)},\boldsymbol{\beta}_{j,\cdot}^{(\ell)}]\|_2+\|\boldsymbol{\beta}_{j,\cdot}^{(\ell)}\|_F\right)$	

Note that groups are weighted by their cardinality to prevent regularization favor larger groups. Each class of model is described in detail in the following sections.

Group Lasso Penalties

The Group Lasso (Yuan and Lin [2006]) has emerged as a popular penalized regression procedure that can take into account the inherent structure of a multivariate time series. We consider two group structures for the endogenous covariates: a lag based grouping (*Lag* Group VARX-L, Equation 4) and a grouping that distinguishes between a series' *own* lags (diagonal

entries of $\mathbf{\Phi}^{(\ell)}$) and those of other series (off diagonal entries of $\mathbf{\Phi}^{(\ell)}$) (*Own/Other* Group VARX-L, Equation 5).

Though both penalties employ the same solution algorithm, since the groupings under the Lag Group VARX-L are proper submatrices, it is possible to directly solve the matrix optimization problem as opposed to performing a least squares transformation, resulting in a substantially lower computational overhead than the Own/Other Scenario.

Under both endogenous structures, we group exogenous variables by column. In our experience, we have found that assigning each exogenous covariate to its own group substantially increases computation time without an improvement in forecasts and a lag-based grouping is too general. Hence, the column-based grouping serves as a compromise; allowing for flexibility while still resulting in a computationally efficient optimization problem. In economic or financial applications, one may wish to consider *segmentized groupings* by sector or economy, akin to that proposed by Song and Bickel [2011]. We plan to enable user-specified groupings in future updates of `BigVAR` .

Sparse Group Lasso Penalties

In certain scenarios, a group penalty can be too restrictive. If a group is included, all of its coefficients are constrained to be nonzero. On the other hand, specifying a large number of groups will substantially increase computation time and, in our experience, generally does not improve forecasting performance.

As a compromise, we consider applying a *Sparse Group Lasso* penalty (Equations 6 and 7) proposed by Simon et al. [2013], which allows for "within-group" sparsity via a convex combination of L_1(unstructured sparsity) and L_2 (structured sparsity) penalties. `BigVAR` offers the Sparse Group VARX-L for both the Lag and Own/Other structured groupings.

The parameter that controls the weights of the two penalties, α, is set to $\frac{1}{k+1}$ according to a heuristic to control within-group sparsity. In future updates, we plan to enable the option to jointly estimate α and λ by cross-validation.

Basic Penalty

The Basic VARX-L (8) is the most general grouping, which can be viewed as partitioning each variable into its own group. It does not incorporate any of the structure of the VARX, but it results in a comparably simpler optimization problem, hence it can scale to much larger problems than structured approaches.

Nested Penalty Structures

The previously discussed penalty structures are disjoint groupings that form a partition of $[\mathbf{\Phi}, \boldsymbol{\beta}]$. In certain scenarios, one might wish to assign a preference to endogenous versus exogenous variables. The *Endogenous-First* Group Lasso (9) utilizes a nested penalty to prioritize endogenous series. At a given lag, an exogenous series can enter the model only if their endogenous counterpart is nonzero. Note that by construction, this penalty requires that $p \geq s$.

Solution Methods

In order to solve the optimization problems in the form of Equation (3), we employ computationally tractable algorithms designed for non-smooth convex functions. Our solution methods do not make calls to external packages or commercial convex optimization solvers. All of our solution algorithms are coded in `C++` and linked to `R` via `Rcpp` Eddelbuettel and François

[2011], `RcppArmadillo` Eddelbuettel and Sanderson [2014], and `RcppEigen` Bates et al. [2012]. The specific algorithms that we utilize for each procedure are outlined in Table 2.

Table 2: Solution Algorithms employed for each structured penalty

Algorithm	Solution Procedure	Reference
Lag Group Lasso	Block Coordinate Descent	Qin et al. [2010]
Own/Other Group Lasso	Block Coordinate Descent	.
Lag Sparse Group Lasso	Proximal Gradient Descent	Beck and Teboulle [2009]
O/O Sparse Group Lasso	Proximal Gradient Descent	.
Lasso	Coordinate Descent	Friedman et al. [2010]
Endogenous-First VARX	Fast Iterative Soft Thresholding	Jenatton et al. [2011]

Implementation details are provided in the Appendix of Nicholson et al. [2015].

2.2 Hierarchical Vector Autoregression

The VARX-L procedures remain agnostic with regard to lag order selection. Hence, as the maximum lag order increases, forecast performance may start to degrade, as each group is treated democratically despite more distant data generally tending to be less useful in forecasting. Within the VAR context, we propose the HVAR class of models which alleviates this issue by embedding lag order into *hierarchical group lasso* penalties. To allow for greater flexibility, intead of imposing a single, universal lag order, it is allowed to vary across marginal models. `BigVAR` provides three hierarchical penalties:

1. **Componentwise HVAR**

   Maximum lag order can vary across marginal models, but within a series, all components have the same maximum lag.

$$\mathcal{P}_y(\mathbf{\Phi}) = \sum_{i=1}^{k}\sum_{\ell=1}^{p} \|\mathbf{\Phi}_i^{(\ell:p)}\|_2$$

2. **Own/Other HVAR** This structure is similar to Componentwise, but imposes an additional layer of hierarchy: prioritizing "own" lags over "other" lags within a lag. This penalty incorporates a commonly-used specification in the Bayesian VAR with a Minnesota Prior (Litterman [1979]) that own lags are more informative than other lags.

$$\mathcal{P}_y(\mathbf{\Phi}) = \sum_{i=1}^{k}\sum_{\ell=1}^{p} \left[\|\mathbf{\Phi}_i^{(\ell:p)}\|_2 + \|(\mathbf{\Phi}_{i,-i}^{(\ell)}, \mathbf{\Phi}_i^{([\ell+1]:p)})\|_2\right]$$

3. **Elementwise HVAR**

   The most general structure; in each marginal model, each series may have its own maximum lag. Under this framework, there are k^2 possible lag orders.

$$\mathcal{P}_y(\mathbf{\Phi}) = \sum_{i=1}^{k}\sum_{j=1}^{k}\sum_{\ell=1}^{p} \|\mathbf{\Phi}_{ij}^{(\ell:p)}\|_2$$

In addition, for comparison purposes we provide a *Lag Weighted Lasso*, which consists of a Lasso penalty that increases geometrically with lag,

$$\mathcal{P}_y(\mathbf{\Phi}) = \sum_{\ell=1}^{p} \ell^{\gamma} \|\mathbf{\Phi}^{(\ell)}\|_1,$$

in which $\gamma \in [0, 1]$ is an additional tuning parameter. This is similar to the approach proposed by Song and Bickel [2011]. Though it encourages greater regularization at more distant lags, it does not explicitly force sparsity and requires the specification of an arbitrary functional form as well as an additional tuning parameter.

Example Sparsity Patterms of the HVAR Procedures and the lag-weighted lasso are depicted in Figure 2.

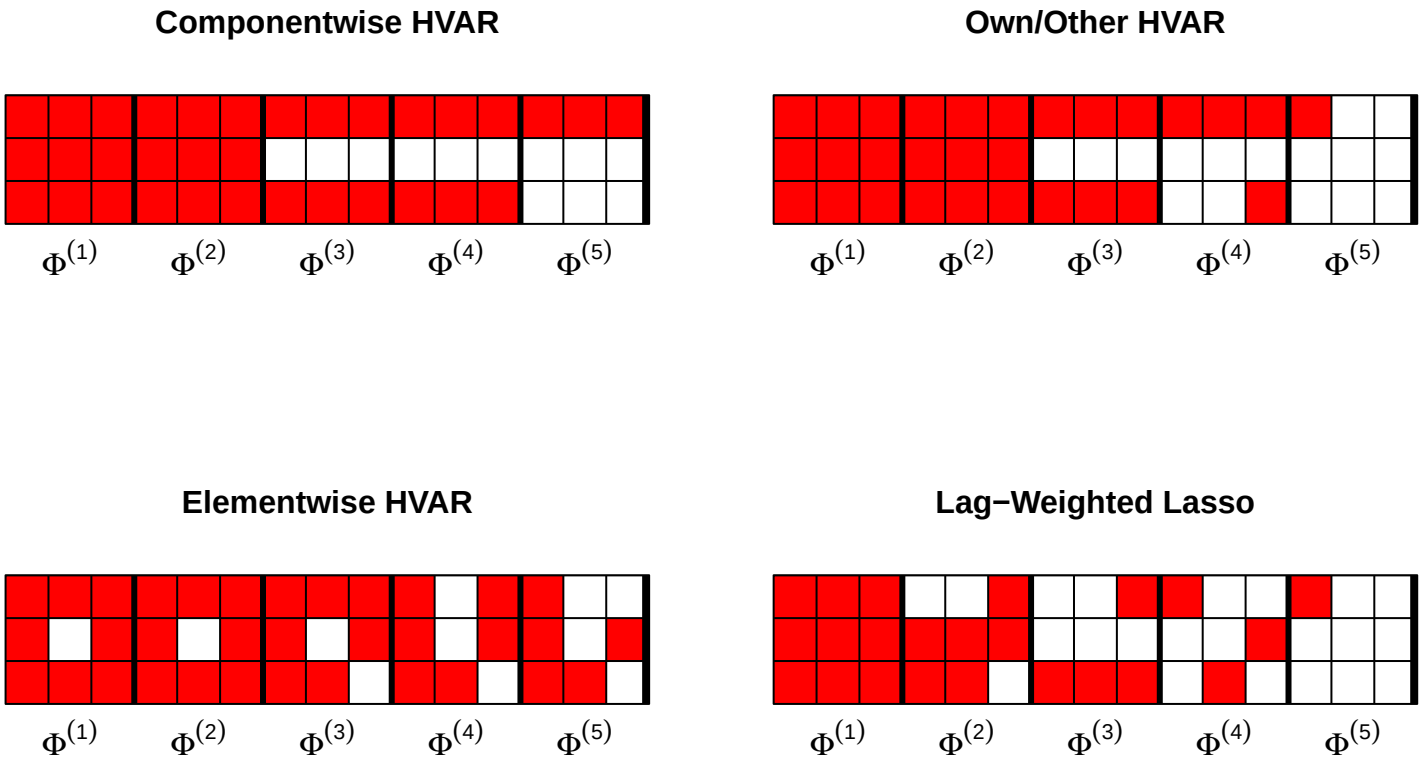


Figure 2: HVAR and Lag-Weighted Lasso Example Sparsity Patterns (k=3,p=5)

2.3 Penalty Parameter Selection

In order to account for time dependence, cross-validation is conducted in a rolling manner. We perform cross validation between times T_1 and T_2 (by default, `BigVAR` sets $T_1 = \lfloor \frac{T}{3} \rfloor, T_2 = \lfloor \frac{2T}{3} \rfloor$). $\hat{\lambda}$ is selected from a grid of values $\lambda_1, \ldots, \lambda_n$. At T_1, we forecast $\hat{\mathbf{y}}_{T_1+1}^{\lambda_i}$ for $i = 1, \ldots, n$, and sequentially add observations until time T_2. We choose $\hat{\lambda}$ as the minimizer of MSFE:

$$MSFE(\lambda_i) = \frac{1}{(T_2 - T_1 - 1)} \sum_{t=T_1}^{T_2-1} \|\hat{\mathbf{y}}_{t+1}^{\lambda_i} - \mathbf{y}_{t+1}\|_2^2.$$

$T_2 + 1$ through T is used for out of sample forecast evaluation. The process is visualized in the Figure 3.

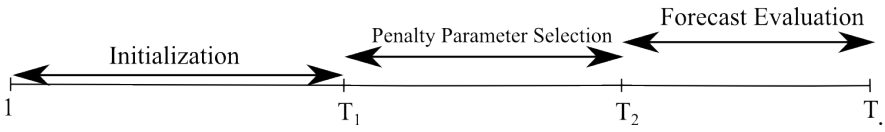


Figure 3: (Reproduced from Nicholson et al. [2015]) Illustration of Rolling Cross-Validation

3 Estimating VAR models with `BigVAR`

In this section, we demonstrate how to utilize `BigVAR` to forecast a set of quarterly macroeconomic indicators procured from Quandl. We consider forecasting three US macroeconomic series: Consumer Price Index, Federal Funds Rate, and Gross Domestic Product. We first download the data using the API provided in the `Quandl` package and then transform each series to stationarity, via the transformation codes provided by Stock and Watson [2005]

```
library(devtools)
if (!require(Quandl)) {
    install_github("quandl/R-package")
}
# to install BigVAR from Github
if (!require(BigVAR)) {
    install_github("wbnicholson/BigVAR/BigVAR")
}
suppressPackageStartupMessages(library(Quandl))
# Gross Domestic Product (Relative to 2000)
GDP = Quandl("FRED/GDP", type = "xts")
GDP <- GDP/mean(GDP["2000"]) * 100
# Transformation Code: First Difference of Logged Variables
GDP <- diff(log(GDP))
# Federal Funds Rate
FFR = Quandl("FRED/FEDFUNDS", type = "xts", collapse = "quarterly")
# Transformation Code: First Difference
FFR <- diff(FFR)
# CPI ALL URBAN CONSUMERS, relative to 1983
CPI = Quandl("FRED/CPIAUCSL", type = "xts", collapse = "quarterly")
CPI <- CPI/mean(CPI["1983"]) * 100
# Transformation code: Second difference of logged variables
CPI <- diff(log(CPI), 2)
```

GDP and CPI start in Quarter 1 of 1947, but since the Federal Funds Rate was not officially published until 1954, we discard all realizations of GDP and CPI before Quarter 4 of 1954. The data ranges until Quarter 1 of 2015, resulting in $T = 243$. As is standard in the regularization framework, before estimation we standardize each series to have zero mean and unit variance.

```
k = 3
Y <- cbind(CPI, FFR, GDP)
Y <- na.omit(Y)
# Demean
Y <- Y - (c(rep(1, nrow(Y)))) %*% t(c(apply(Y, 2, mean)))
# Standarize Variance
for (i in 1:k) {
    Y[, i] <- Y[, i]/apply(Y, 2, sd)[i]
}
# Plot
```

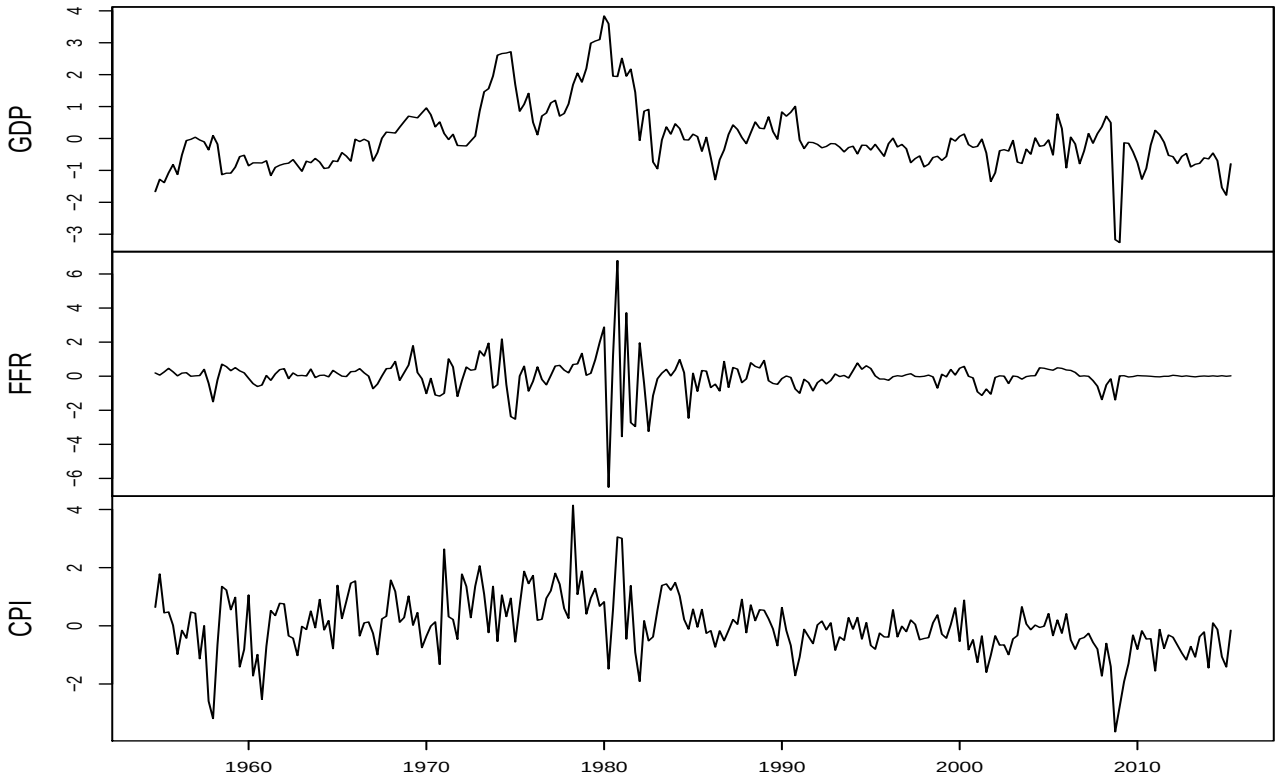


Figure 4: Plots of Standardized Quarterly GDP, Federal Funds Rate, and CPI

```
plot(as.zoo(Y), main = "", xlab = "", ylab = c("GDP", "FFR", "CPI"))
```

3.1 Constructing an object of class `BigVAR`

In an effort to streamline functionality, `BigVAR` utilizes `R`'s s4 object class system. In order to fit a model, the user constructs an object of class `BigVAR` that contains the data as well as model specifications. A `BigVAR` object can be created through the wrapper function `constructModel`, which encompasses both HVAR and VARX-L implementation. For example, if we wish to construct an Own/Other Group Lasso $\text{VAR}_3(4)$

```
Model1 = constructModel(as.matrix(Y), p = 4, struct = "Diag", gran = c(25, 10),
    verbose = FALSE, VARX = list())
```

The required arguments for `constructModel` are

- `Y`: $T \times k$ multivariate time series (in matrix form)
- `p`: predetermined maximum lag order
- `struct`: Structured Penalty
- `gran`: Two arguments that characterize the grid of penalty parameters. The first denotes the depth of the grid and the second the number of gridpoints.

The choices for the argument `struct` are outlined in Table 2. In the `BigVAR` framework, `gran` denotes the only "hyperparameter" that must be set by the end user. Following Friedman et al. [2010], the grid of penalty values is constructed by starting

with the smallest value in which all coefficients will be zero, then decrementing in log linear increments. The grid ends at a fraction of this maximum value (as dictated by the first argument in `gran`). These bounds are detailed in the Appendix of Nicholson et al. [2015]. In practice these bounds can be coarse. Therefore, we implement a variant of Algorithm 3 in Lou et al. [2014], which uses a bisection approach in order to find a tighter data-driven bound. In practice, we find that the best choices for grid depth tend to be between 10 and 50, depending on the number of series included.

The number of penalty parameters is also left to user input. The package `glmnet` calls for 100 penalty parameters by default, but we have found no substantial forecasting improvement in considering any more than 10. If the user wishes to provide their own penalty parameters, they can do so through `gran`, but they must also set the optional argument `ownlambdas` to `TRUE`.

The additional optional arguments are

- `RVAR`: Relaxed VAR indicator (provides the option to refit the support recovered by least squares) (default: `FALSE`)
- `h`: forecast horizon (default 1)
- `crossval`: Cross-Validation Procedure (default "rolling")
- `MN`: Option for the "Minnesota VAR(X)," which shrinks toward a vector random walk.
- `verbose`: Indicator for progress bar (default TRUE)
- `IC`: Indicator to return AIC and BIC benchmarks (default TRUE)
- `VARX`: List of VARX specifications
- `T1`: Start of cross validation period (default $\lfloor \frac{T}{3} \rfloor$)
- `T2`: Start of forecast evaluation period (default $\lfloor \frac{2T}{3} \rfloor$)
- `ONESE`: Indicator for "One Standard Error" heuristic described in Hastie et al. [2009] that returns the most sparse solution within one standard error of the best performing result (default FALSE)

More details on some of the optional arguments are provided below

Relaxed VAR

Since the Lasso and its structured variants are known to shrink nonzero coefficients, they are often used for variable selection, with the resulting model being refit via least squares (Meinshausen [2007]). This procedure is known as the *Relaxed* VAR.

We offer this capability in `BigVAR`, and to ensure that the resulting coefficient matrices are numerically stable, we utilize a variant of the least squares procedure developed by Neumaier and Schneider [2001], which adds a small ridge penalty (on the order of $\epsilon_{\text{Machine}}$).

We have found that refitting generally does not improve forecasts. This could be due in part to the observation by Lütkepohl [2005], that in the presence of parameter restrictions, the Ordinary Least Squares and Generalized Least Squares estimators no longer coincide.

Cross-Validation Choices

By default, `BigVAR` implements rolling cross-validation. An alternative procedure, H-block cross-validation (Burman et al. [1994]), also respects time dependence and could be considered in future updates of `BigVAR` . In our context, H-block cross validation essentially involves, at each timepoint, removing from the training set $\max(p, s)$ observations preceding and following the "test" observation. In our applications, we have found that the use of h-block cross validation does not improve forecasts, but is substantially more computationally intensive than the rolling approach.

The Minnesota Lasso

As opposed to shrinking every coefficient toward zero, all of the procedures in `BigVAR` can be modified to instead shrink toward a vector random walk (i.e. $\Phi^{(1)} = I_k$, all other coefficient matrices are still set to zero). Such a modification is akin to the Bayesian VAR with Minnesota Prior of Litterman [1979]. This approach can be useful in scenarios exhibiting evidence of nonstationarity, which is commonplace in macroeconomic data. For more information about this approach, see Nicholson et al. [2015]

Information Criterion Benchmarks

By default, we compare our methods to the more conventional approach of selecting from a sequentially increasing lag order as chosen by AIC or BIC and fitting the resulting VAR by least squares. Due to poor numerical stability as well as substantial computational overhead, we don't recommend including this benchmark when working in high dimensions (i.e. $kp \approx T$), hence we offer the option to disable it.

3.2 Implementation

In order to estimate a model with `BigVAR` using rolling cross validation, we simply need to execute the command `cv.BigVAR` on an object of class `BigVAR` as detailed below.

To fit an Own/Other Group $\text{VAR}_3(4)$, we simply run

```
Model1 = constructModel(as.matrix(Y), p = 4, struct = "Diag", gran = c(25, 10),
    verbose = F, VARX = list())
Model1Results = cv.BigVAR(Model1)
Model1Results
```

```
## *** BIGVAR MODEL Results ***
## Structure
## [1] "Diag"
## Maximum Lag Order
## [1] 4
## Optimal Lambda
## [1] 16.9702
## Grid Depth
## [1] 25
## Index of Optimal Lambda
## [1] 6
## In-Sample MSFE
## [1] 3.826
```

```
## BigVAR Out of Sample MSFE
## [1] 1.085
## *** Benchmark Results ***
## Conditional Mean Out of Sample MSFE
## [1] 1.824
## AIC Out of Sample MSFE
## [1] 1.106
## BIC Out of Sample MSFE
## [1] 1.128
## RW Out of Sample MSFE
## [1] 1.14
```

An object of class `BigVAR.Results` is returned. By default, the output displays model characteristics, such as the penalty structure, maximum lag order, the λ chosen, and both in sample and out of sample MSFE. For comparison purposes, the out of sample MSFE from benchmarks including sample mean, random walk, AIC, and BIC are also returned.

Additional information available in `Model1Results` includes

```
# Coefficient matrix at end of evaluation period
Model1Results@betaPred
# Residuals at end of evaluation period
Model1Results@resids
# Lagged Values at end of evaluation period
Model1Results@Zvals
```

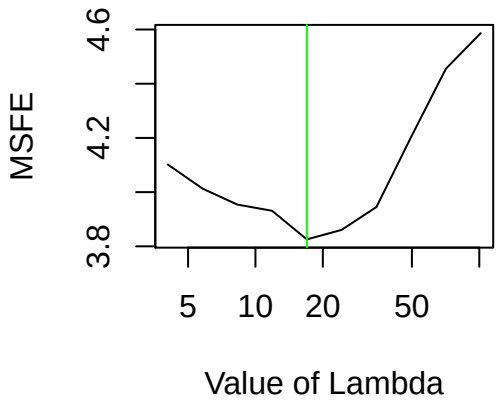
Diagnostics and Additional Features

As most of `BigVAR` is automated, there is generally little in the way of diagnostics is required. This section details the features of BigVAR that both tailor to the specific forecasting scenarios of the end-user and ensure that the most accurate possible forecasts are delivered.

3.2.1 Penalty Grid Position

`plot.BigVAR.results` allows us to visualize the position of $\hat{\lambda}$. Ideally, we want it to be near the middle of the grid; if it is at the boundary, increasing the depth of the grid may lead to improved forecasting performance.

```
plot(Model1Results)
```



In this application, we find that our $\hat{\lambda}$ is at the center of the grid as desired. If it were not, we could simply increase the first parameter in the "gran" argument in `ConstructModel`.

3.2.2 Alternative Structures

Since Figure 4 shows that all series tend to exhibit a substantial degree of persistence, the Minnesota VAR might be a more appropriate choice than shrinking toward zero.

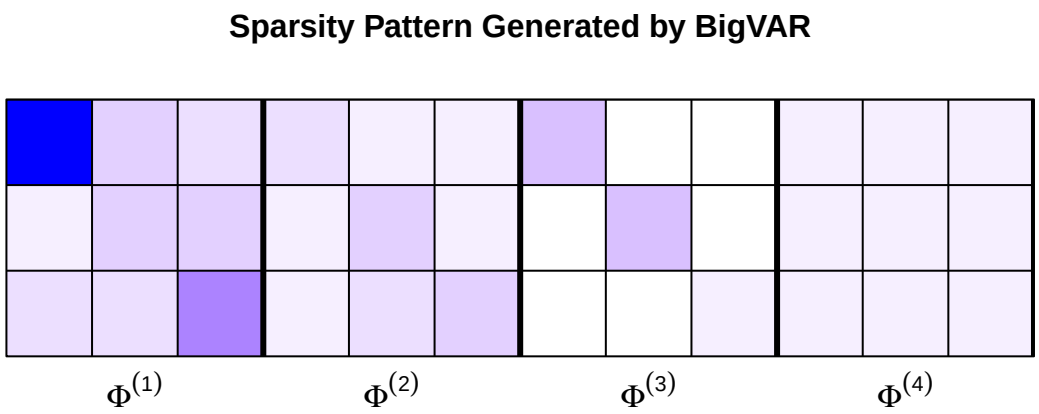
```
Model1@Minnesota = TRUE
mean(cv.BigVAR(Model1)@OOSMSFE)

## [1] 1.024424
```

Out of sample forecast performance is substantially improved by imposing the "Minnesota VAR" penalty.

3.2.3 Visualizing Sparsity Patterns

We also allow the ability to view the sparsity pattern of the final coefficient matrix in the evaluation period.

```
SparsityPlot.BigVAR.results(Model1Results)
```



3.2.4 N-step ahead prediction

Finally, predictions can be obtained via the `predict` function. While n-step ahead predictions are available for VAR models, we currently only allow 1-step ahead predictions for VARX models.

```
predict(Model1Results,1)

##              [,1]
## [1,] -0.75937293
## [2,] -0.01849378
## [3,] -0.31100812
```

Estimation with fixed λ

In certain applications, one may initially estimate $\hat{\lambda}$ by rolling cross-validation and continue to use that value as new data becomes available. In such a scenario, it would not be desirable to fit a model with `cv.BigVAR`. We provide an alternative

function `BigVAR.est` which requires an object of class `BigVAR` and will provide a single estimate using all available data for either a fixed grid of λ values or via a grid determined by the data (as is done in `cv.BigVAR`).

As an example, consider obtaining estimate

```
lambda <- Model1Results@OptimalLambda
Model1@ownlambdas = TRUE
Model1@Granularity <- lambda
BigVAR.est(Model1)

## $B
## , , 1
##
##               [,1]        [,2]         [,3]       [,4]         [,5]
## [1,]  0.004212265  0.84945375  0.071498255 0.02375179 -0.027991946
## [2,] -0.002332942 -0.02553335 -0.008273462 0.08047007 -0.004736684
## [3,] -0.010033462  0.03383733  0.038584472 0.46494833  0.025839640
##               [,6]        [,7]       [,8]      [,9]      [,10]       [,11]
## [1,]  0.003736765 0.002798721 0.05107469 0.0000000 0.00000000 0.002129053
## [2,] -0.077192712 0.027693084 0.00000000 0.1178978 0.00000000 0.000000000
## [3,] -0.067544357 0.053245261 0.00000000 0.0000000 0.01520941 0.000000000
##              [,12]       [,13]
## [1,]  0.000000000 0.000000000
## [2,] -0.008513333 0.000000000
## [3,]  0.000000000 0.006299605
##
##
## $lambdas
## [1] 16.97018
```

VARX-L Estimation

If the user wishes to estimate a VARX-L model, first the series should be arranged such that the first k columns are endogenous series and the remaining are exogenous series. After doing so, a list of VARX specifications needs to be passed to `constructModel`. The list must contain two elements: k denotes the number of endogenous series and s the maximum lag order for exogenous series. For example, if we want to forecast GDP and the Federal Funds Rate using CPI as an exogenous series (with $s = 4$), we simply need to specify:

```
VARX = list()
VARX$k = 2  # 2 endogenous series
VARX$s = 4  # maximum lag order of 4 for exogenous series
Model1 = constructModel(as.matrix(Y), p = 4, struct = "None", MN = F, gran = c(25,
    10), verbose = F, VARX = VARX)
Model1Results = cv.BigVAR(Model1)
```

Univariate Estimation

Univariate estimation (i.e. $k = 1$) is possible, however, many of the structures break down (e.g. Own/Other has no context in this setting). Hence, the only structures that offer univariate support are Basic VARX-L, Lag Group VARX-L, and Componentwise HVAR.

Table 3: Arguments for "Struct" in `constructModel`

Struct Argument	Penalty	VAR Support	VARX Support	Univariate Support
"Lag"	Lag Group	X	X	X
"Diag"	Own/Other Group	X	X	.
"SparseLag"	Lag Sparse Group	X	X	.
"SparseDiag"	O/O Sparse Group	X	X	.
"None"	Basic	X	X	X
"EF"	Endogenous-First	.	X	.
"HVARC"	Componentwise Hierarchical	X	.	X
"HVAROO"	Own/Other Hierarchical	X	.	.
"HVARELEM"	Elementwise Hierarchical	X	.	.
"Tapered"	Lag Weighted Lasso	X	.	.

References

Douglas Bates, Romain Francois, and Dirk Eddelbuettel. Rcppeigen: Rcpp integration for the eigen templated linear algebra library. *R package version 0.3*, 1, 2012.

Amir Beck and Marc Teboulle. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM Journal on Imaging Sciences*, 2(1):183–202, 2009.

Prabir Burman, Edmond Chow, and Deborah Nolan. A cross-validatory method for dependent data. *Biometrika*, 81(2): 351–358, 1994.

Dirk Eddelbuettel and Romain François. Rcpp: Seamless R and C++ integration. *Journal of Statistical Software*, 40(8): 1–18, 2011. URL `http://www.jstatsoft.org/v40/i08/`.

Dirk Eddelbuettel and Conrad Sanderson. Rcpparmadillo: Accelerating r with high-performance c++ linear algebra. *Computational Statistics and Data Analysis*, 71:1054–1063, March 2014. URL `http://dx.doi.org/10.1016/j.csda.2013.02.005`.

Jerome Friedman, Trevor Hastie, and Rob Tibshirani. Regularization paths for generalized linear models via coordinate descent. *Journal of statistical software*, 33(1):1, 2010.

Trevor Hastie, Robert Tibshirani, Jerome Friedman, T Hastie, J Friedman, and R Tibshirani. *The elements of statistical learning*, volume 2. Springer, 2009.

Rodolphe Jenatton, Julien Mairal, Guillaume Obozinski, and Francis Bach. Proximal methods for hierarchical sparse coding. *The Journal of Machine Learning Research*, 12:2297–2334, 2011.

Robert B. Litterman. Techniques of forecasting using vector autoregressions. Working papers, Federal Reserve Bank of Minneapolis, 1979.

Yin Lou, Jacob Bien, Rich Caruana, and Johannes Gehrke. Sparse partially linear additive models. *arXiv preprint arXiv:1407.4729*, 2014.

Helmut Lütkepohl. New introduction to multiple time series analysis. 2005.

Nicolai Meinshausen. Relaxed lasso. *Computational Statistics & Data Analysis*, 52(1):374–393, 2007.

Arnold Neumaier and Tapio Schneider. Estimation of parameters and eigenmodes of multivariate autoregressive models. *ACM Transactions on Mathematical Software (TOMS)*, 27(1):27–57, 2001.

William Nicholson, David Matteson, and Jacob Bien. VARX-L: Structured Regularization for Large Vector Autoregression with Exogenous Variables. arXiv preprint arXiv:1508.07497, 2015.

William B Nicholson, Jacob Bien, and David S Matteson. Hierarchical vector autoregression. *arXiv preprint arXiv:1412.5250*, 2014.

Zhiwei Qin, Katya Scheinberg, and Donald Goldfarb. Efficient block-coordinate descent algorithms for the group lasso. *Mathematical Programming Computation*, pages 1–27, 2010.

Noah Simon, Jerome Friedman, Trevor Hastie, and Robert Tibshirani. A sparse-group lasso. *Journal of Computational and Graphical Statistics*, 22(2):231–245, 2013.

Song Song and Peter Bickel. Large vector auto regressions. 2011. journal: arXiv preprint arXiv:1106.3915.

James H Stock and Mark W Watson. An empirical comparison of methods for forecasting using many predictors. *Manuscript, Princeton University*, 2005.

Ming Yuan and Yi Lin. Model selection and estimation in regression with grouped variables. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 68(1):49–67, 2006.