5-2017

Vector Autoregressions with Machine Learning

Michael Allen Varner
mvarner@bates.edu

Vector Autoregressions with Machine Learning

An Honors Thesis

Presented to

The Faculty of the Department of Economics

Bates College

In partial fulfillment of the requirements for the

Degree of Bachelor of Arts

By

Michael Allen Varner

Lewiston, Maine

April 27, 2017

Abstract

I develop three new types of vector autoregressions that use supervised machine learning models to estimate coefficients in place of ordinary least squares. I use these models to estimate the effects of monetary policy on the real economy. Overall, I find that the machine learning vector autoregressions produce impulse responses that are well behaved and similar to their ordinary least squares counterparts. In practice, the machine learning vector autoregressions produce more conservative estimates than the traditional ordinary least squares vector autoregressions. Additionally, I establish a simulation scheme to compare the relative efficiency of impulse responses generated from machine learning and ordinary least squares vector autoregressions. To calculate confidence intervals, I use a bias corrected bootstrapping method from Politis and Romano (1994) called the stationary bootstrap. In future work, I intend to compare these impulse responses using simulated data from Killian and Kim (2011).

Keywords
impulse response functions; supervised machine learning; vector autoregressions; bootstrap; monetary policy

JEL Classification
C31; C45; E58;

Acknowledgments

I have had to overcome obstacles and setbacks on the way to completing this thesis. Luckily, I did not have to do this alone. I received great support both academically and personally. I would first like to thank my advisor Nathan Tefft for guiding me through this process and for encouraging me to be ambitious. I want to thank my father Christoper Varner for his support in all of my endeavors and for making it possible to attend Bates. Throughout my time at Bates, I have made many lasting relationships. In particular, I would like to thank Amanda Zakowich, as well as other friends, for supporting me in this process. I would not have been able to complete this work if it were not for them. I want to thank Chris Orlandi and Matt Pedlow for introducing me to economics and giving me the tools I needed to succeed at Bates. Kai Evenson gave me great technical support.

Lastly, I want to dedicate this thesis to Catherine Susan Croisant Varner (1954-2010). I have always admired her grace, kindness, and dedication.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background

Vector autoregressions (VARs) have been used by economists over the past 36 years to analyze multivariate time-series data. VARs provide a convenient framework for policy analysis, forecasting, structural inference, and data description (Stock and Watson, 2001). In practice, this class of models is most commonly used in empirical macroeconomics. These models provide a framework to answer a variety of questions. Examples include, what is the effect of a 2% increase in the federal funds interest rate on unemployment in 6 months? or Is there a persistent effect on GDP growth from increases in technology?

Since VARs were introduced in the 1980s, an emerging class of estimators has been developed in the computer science and statistics literature called supervised machine learning (ML) models. Recently, this class of mod-

els has seen more use in the economics literature. These models can be highly flexible and are designed for prediction rather than for causal inference. In this thesis, I develop three VAR models, which use ML models to estimate coefficients, and apply these models to a monetary policy question. VARs have traditionally been estimated using ordinary least squares (OLS) also known as linear regression.

Using machine learning models to calculate VARs is attractive because the results may be more accurate than traditional linear econometric methods if the data generating process (DGP) is nonlinear. Furthermore, machine learning models can be highly flexible and the machine learning literature has developed model selection tools; which aim to minimize out of sample prediction error. However, one concern about these models is that many are biased, and this is because they were designed to maximize predictive accuracy rather than for causal inference.[1] A fundamental property of statistical estimators is that the total error can be decomposed into the error from bias and the error from variance. In the ML community this is called the bias-variance trade off.[2] Each of the ML models I use has some bias, but it also has lower variance compared to OLS. OLS is at the far end of this trade off, in that it is has no bias but high variance. This trade off gets to the heart of the differences between the ML models I consider and OLS. Where the 'perfect' estimator lies on this spectrum of bias and variance depends on the goals of the analysis.

Even though these ML models are biased it could be the case that they

[1] In fact, little is known about the asymptotic properties of many machine learning models.

[2] See Galit (2010) for more.

may be more efficient compared to OLS. In a world with an infinite amount of data and a linear DGP, OLS would do a better job at estimating structural parameters compared to the ML models because OLS is unbiased. However, VARs are often applied to small data sets. In macroeconomics small data sets are typical because reliable data only goes back roughly 50 years. This means that the ML models may be able to get closer to the true parameter estimates compared to OLS because the sample is finite. ML models have a comparative advantage over OLS in out of sample prediction. Therefore, there could be scenarios in which the ML models would outperform OLS. The difficult part becomes constructing a scenario where this can be shown to happen and then to argue that this scenario is applicable to data.

To demonstrate how the different ML VARs work in practice, I estimate how quantitative easing affects the unemployment rate. To do this, I reestimate the results of Shea *et al.*, (2017) using ML VARs. The model used in Shea *et al.*, (2017) is a five variable OLS VAR estimated with 45 years of monthly data. This is a common example of how VARs have been used in the past to estimate macroeconomic dynamics. Many papers in this area which employ VARs use similar types and quantities of data. This gives me confidence that the results presented in chapter (4) represent how VARs have been used in the past.

The most common way to interpret the results from a VAR is by looking at its corresponding impulse responses (IRs). As I will show in chapter (2), VARs are systems of linear equations and this makes interpretation difficult. To simplify the interpretation, people in the applied literature have use IRs as a way to characterize an estimated model's dynamics. Intuitively an IR is

just a marginal effect projected over time. We can think of them as describing what happens to one variable over time from an impulse to another variable.[3] In chapter (2) I will describe how IRs are calculated, but for intuition they are mostly dependent on regression coefficients. Confidence intervals for the IR point estimates are most commonly calculated using an asymptotic formula derived from the VAR. Confidence intervals allow for easy interpretation of the results. The IR point estimates are statistically different from zero if it is either the case that the lower confidence estimate is above zero or the upper confidence estimate in below zero.

In chapter (3) I will compare how the ML VARs defined in the previous section perform relative to the OLS VAR by looking at the associated IRs. I use simulations to determine the relative performance of the IR estimators.

1.2 Motivation

The motivation for this thesis comes from the intersection of my interests in time-series econometrics and machine learning. After taking Big Data and Economics with Professor Nathan Tefft, I became interested in ML models and their use in economics. Towards the end of the Big Data class I remembered the VAR models I learned about the previous semester and wondered if it would be possible to incorporate ML models into the structure of a VAR. This is how I came to my research topic.

I have decided to focus my attention on IRs because I have experience using them and I do not know of any ways to construct IRs which do not use

[3]There are IRs with impulses from the response variable, but these are usually less interesting.

OLS. Furthermore, ML models are becoming more widely used in economics, and I want to integrate some of these methods into time-series analysis. So far, ML models have mostly been used in microeconomics.[4] However, the machine learning community has developed a wide range of models which have been designed to tackle a variety of tasks.

There has been some economic literature that uses ML models to estimate causal treatment effects and this is relevant for my purposes because IRs are an inferential statistic.[5] In Athey and Imbens (2015) the authors propose ways in which ML models could be used to estimate average causal treatment effects. They assume a binary treatment group, maybe from a randomized control trial, and show that ML models can accurately estimate causal treatment effects. This paper was useful to me while I was developing my thesis topic because it was an example of how to use ML models for causal inference.

1.3 Relevant Literature

In working on this thesis I came across papers not only from the economics literature but from the computer science and statistics literature as well. In a forthcoming coming article in the *International Journal of Forecasting* Nicholson *et al.* (2015) lay out an optimization scheme using the LASSO model to estimate a VAR.[6] VAR models are heavily parameterized

[4]Examples include Athey and Imbens (2015), Bajari *et al.* (2015), Chernozhukov *et al.* (2015), and Belloni *et al.*, (2011).

[5]Inferential in this context refers to a statistic which does not measure model performance. Performance statistics include RSS and MSE.

[6]LASSO was introduced in Tibshirani (1996).

compared to other time-series models. For macroeconomic modeling, this usually means that researchers quickly run into degrees of freedom constraints when adding in endogenous variables. The authors address this problem by introducing a method to calculate VARs using the LASSO. They do this to be able to include more endogenous variables into the model. Their model is able to do this because the LASSO performs variable selection at the same time as estimating coefficients. In practice, the LASSO makes the VAR coefficient matrices sparse.[7] This property of the LASSO is able to reduce the dimensionality of the data and allow for more endogenous variables to be used.

Nicholson *et al.* (2015) does not address IRs using the LASSO VAR. Nicholson *et al.* seem to mostly be concerned with prediction accuracy rather than causal inference. They use U.S. macroeconomic data to illustrate how their models are able to forecast future macroeconomic conditions. They find that their models are better than the traditional OLS VAR models. It is peculiar that the authors only compare these models using a metric for predictive performance. VARs were designed for causal inference and not for prediction. What makes VARs useful for causal inference is that they have a structure. This also makes them poor for prediction because the world is most likely not as linear as the models assume. The authors do not fairly compare the OLS VARs and LASSO VARs because in a game of pure prediction the ML models should win every time. What seems to really matter is the efficiency of the models in estimating some inferential statistic.[8] This criticism comes from my previous experience with VARs. I

[7] i.e. many entries are zero.
[8] Perhaps an IR.

have only had exposure to VARs in the context of causal inference and not for prediction/forecasting.

The authors published an R package called 'BigVAR' that implements the LASSO VAR model. I would have considered using this package if I had known about it at the beginning of this project because it could have sped up the programming process.

Related to Nicholoson *et al.* (2015) a PhD dissertation in Statistics, Basu (2014), looked at the theoretical properties of using the LASSO for high dimensional VARs. Basu establishes the consistency of the LASSO VAR under high-dimensional scaling. This piece of the dissertation is not directly applicable to my thesis, but it is great to see that people are thinking about the asymptotic properties of these kinds of models. To supplement his theoretical results, Basu conducts many simulations and overall they confirm his theory. Basu applies his models to "reconstruction of gene regulatory network from time course gene expression data." I enjoyed this application because I was not aware that VARs were being used in biological gene modeling until seeing this application.

In Hsu *et al.* (2008) the authors developed a subset selection method for VARs using LASSO. Importantly, they did not consider IRs in their analyses and instead focus on using VARs for forecasting. Using simulations, they compared the LASSO to Akaike's information criterion (AIC) and the Bayesian information criterion (BIC). They find that LASSO performed better than either the BIC or AIC. For my purposes, this is an encouraging finding because variable selection is an important step in estimating a VAR. The LASSO VAR may be able to better approximate the DGP compared to

an OLS VAR.

The LASSO is one of the ML models I use in chapter (2) and the other two are called the Elastic Net and Ridge regression.[9][10] The Elastic Net is a generalized version of the LASSO and Ridge regression. It thus captures features from both models. These models are similar to OLS in that they minimize the residual sum of squares, but unlike OLS they include a constraint on the magnitude of the sum of the coefficients. I am most interested in using the LASSO model because it also performs variable selection at the same time as estimating coefficients.

While there is no literature that uses machine learning models to calculate IRs, there has been work done to calculate IRs without the need for a VAR. Jordà (2005) puts forth a method to compute IRs, using local projections, that does not rely on estimating the DGP. There has been much criticism over the assumptions required to calculate IRs from a VAR, and therefore an alternative method with fewer assumptions would be preferable. Jordà proves that his method is robust to misspecification of the DGP, and can accommodate nonlinear specifications that would be difficult to implement in a multivariate setting. Theoretically, Jordà's local projections method could be preferable to VAR IRs because it does not compound misspecification errors.

Even though the local projections method is a consistent estimator of IRs, this does not mean that it is more efficient than a VAR. Meier (2005), uses Monte Carlo simulations to test the efficiency of local projections compared to VAR in estimating IRs. On the whole, he finds that local pro-

[9]Elastic Net was first developed in Hastie and Zou (2005).
[10]Ridge regression was first developed in Hoerl and Kennard (1970).

jections do not perform better than standard VAR methods. Meier uses a New Keynesian Dynamic Stochastic General Equilibrium (DSGE) model for his DGP. He finds some evidence that local projections have more bias and greater variance compared to OLS VARs.

Another paper, by Kilian and Kim (2011), uses Monte Carlo simulations to compare local projections to VAR methods and finds mixed results. To compare relative performance, Killian and Kim look at the IR confidence intervals and not the point estimates. They are concerned with how well the estimators are able to cover the true IR. An IR confidence interval covers the true IR if the true IR lies within the upper and lower bounds at each time horizon. The authors compare the estimators using average coverage rates and average confidence interval length. Kilian and Kim use these two metrics because the ideal IR captures the true IR and has low variation. Visually, low variation corresponds to tight IR confidence intervals. Using only one of these metrics would lead to meaningless results because either the model with the highest or lowest variation would prevail depending on the metric.

To compare local projection IRs to VAR IRs the authors need to assume a DGP. This makes it difficult to compare the models because a model's performance could change depending on which DGP the authors have assumed. In their paper, Kilian and Kim use a four variable VAR(12) for their DGP. I will discuss in more detail how this DGP works in chapter (3) where I present my simulations.

Chapter 2

Machine Learning VARs

In this chapter, I describe how I create the ML VARs that will be used in the coming chapters. To show how I create these models, I will describe how the machine learning models work and the structural VAR theory. The machine learning models that I use address the classical linear regression problem described in section (2.1). This is the most substantial chapter of my thesis and an example of the Python code that I created can be viewed in figure (7.6) of the appendix.

2.1 Linear Regression

Consider the following linear regression problem. Given an outcome or dependent variable y , T observations, and n explanatory variables $\{x_1, x_2, \ldots, x_n\}$ we can approximate the true value of y by taking linear combinations of the explanatory variables. The question becomes, how should we select coefficients and an intercept?

$$\hat{y} = \hat{\beta}_0 + \hat{\beta}_1 x_1 + \hat{\beta}_2 x_2 + \ldots + \hat{\beta}_n x_n \tag{2.1}$$

To answer this question, ordinary least squares (OLS) was developed which selects values of $\beta_i \in \mathbb{R}$ to minimize the residual sum of squares (RSS). Note that the equation (2.2) uses T to indicate the sample size and not n. There is a notational conflict between the time-series and ML literature, so this is why T appears in the upper limit of the summation. The equation for RSS calculates total squared deviations from the true value of y for every observation in the sample.

$$RSS = \sum_{i=1}^{T} \left(y_i - \hat{y}_i \right)^2 = \sum_{i=1}^{T} \left(y_i - \beta_0 - \sum_{j=1}^{n} \beta_j x_j \right)^2 \tag{2.2}$$

Minimizing RSS is an intuitive approach because we want to fit a line that goes through as many observations as possible.[1] The coefficients that OLS produces can be viewed as marginal effects. In applied work, economists will often interpret these coefficients as causal effects. For example, x_2 is associated with/causes a β_2 unit change in the dependent variable y. This framework can be used to answer a wide variety of questions.

From a theoretical perspective, OLS has the great property of being unbiased if we assume away omitted variable bias. However, this unbiasedness comes at the cost of increased variance. I choose to bring up OLS because it is how VAR models are estimated and it will help in understanding the machine learning models that follow. The machine learning models that I describe in section (2.2) provide an alternative way to choose coefficients for

[1] This is true in the case of one independent variable and for more independent variables we quickly lose geometric interpretation.

equation (2.1). The most significant difference between the machine learning models and OLS is that the former are biased. However, VARs are most commonly used on small time-series data sets. This means that the machine learning models could be more efficient than OLS. In other words, the machine learning models may be able to get closer to the true values when compared to OLS in small samples. If the DGP is equation (2.1), then OLS will outperform the machine learning models, but it is highly unlikely that this is the case. This gives me hope that the machine learning models may be more efficient than OLS in small samples.

2.2 Machine Learning Models

I use three distinct supervised ML models to create three VAR models.[2] I will refer to these models as ML VARs. One of the models that I employ is called the Least Absolute Shrinkage and Selection Operator (LASSO). This model solves the following optimization problem described in equation (2.3).

$$\hat{\beta}_{\lambda}^{LASSO} = Min \left\{ RSS \right\} s.t. \sum_{j=1}^{n} |\beta_j| \leq \lambda \quad [3] \tag{2.3}$$

Notice that (2.3) is identical to OLS except that the LASSO constrains how large the βs can be. This constraint is designed to maximize out of sample predictive accuracy. Before selecting the coefficients, each variable is normalized by dividing by its standard deviation. After estimation, the regression

[2]Supervised in this context means that the dependent variable is specified within the model. Other types of machine learning include unsupervised and semi-supervised learning.

[3]Hastie *et al.* (2013)

coefficients are scaled back up. This is done to account for the fact that variables can be measured differently. The other piece of the LASSO determines how to choose λ.

The LASSO chooses λ by searching across a range of potential values and then stops by using a convergence threshold.[4] The LASSO is fitted using a variety of λs and the optimal choice is one that minimizes the k-fold cross-validated MSE. A common algorithm for how the LASSO decides to increase or decrease the size of λ is called Gradient Descent. From my own experience, the LASSO often selects small values for λ.

An interesting property of the LASSO is that it will automatically force some of the coefficients to be exactly zero. This means that the LASSO is able to perform variable selection at the same time as calculating coefficients. This is partially why the LASSO has been very popular in the applied literature because it is able to perform both of these tasks at the same time. The LASSO is computationally inexpensive compared to other ML models.

The other two machine learning models that I use are very similar to the LASSO. The Ridge regression/model is identical to the LASSO except that it squares the βs instead of taking the absolute value.[5] The Ridge regression's optimization problem is described in equation (2.4). Unlike LASSO, Ridge regression does not perform variable selection because it will not necessarily constrain some of the coefficients to be exactly zero. In practice, the Ridge regression gives coefficient estimates that are similar to that of OLS.

[4] I use the default setting of 0.0001 as my convergence threshold.

[5] The Ridge regression/model I use comes from Hoerl and Kennard (1970)

$$\hat{\beta}_{\lambda}^{ridge} = Min\Big\{RSS\Big\}\ s.t.\ \sum_{j=1}^{n}(\beta_j)^2 \leq \lambda \quad [6] \tag{2.4}$$

The last machine learning model that I use is called the Elastic Net from Hastie and Zou (2005). This model combines the coefficient penalties of the LASSO and Ridge regressions into a single optimization problem (2.5). The Elastic Net is able to perform variable selection because it is comprised of the LASSO regression. In practice, Elastic Net yields similar results compared to the LASSO.

$$\hat{\beta}_{\lambda}^{elastic\ net} = Min\Big\{RSS\Big\}\ s.t.\ \left(\phi\sum_{j=1}^{n}(\beta_j)^2 + \alpha\sum_{j=1}^{n}|\beta_j|\right) \leq \lambda \quad [7]\ ; \phi + \alpha = 1 \tag{2.5}$$

These machine learning models can at first seem complicated, but in reality each is just a variant of OLS. The basic idea of these models is that we shrink the magnitude of the coefficients, from their OLS values, just enough to maximize simulated out of sample predictive accuracy. I use the word simulated here to indicate that the out of sample prediction is not truly out of sample since these models rely on a single data set. Since the coefficients are necessarily smaller, this implies that the marginal effect of any independent variable on the dependent variable is smaller than the same effect estimated using OLS. I show examples of this in chapter (4) by comparing IRs generated from ML VARs and OLS VARs.

[6] Hastie *et al.* (2013)

[7] The values of ϕ and α are taken exogenously and need to be specified outside of the model. For my computations, I weigh the coefficient penalties equally or $\phi = \alpha = \frac{1}{2}$.

To implement these models, I have used tools developed by the computer science community. I use the programming language Python version 3.6 and the package called 'Scikit-Learn' by Pedregosa *et al.* (2011) to run the three ML models.

2.3 Validation

To understand how the ML models choose this λ, I will introduce the validation set approach and k-fold cross-validation. The validation set approach is a resampling method that simulates out of sample predictive performance. The idea is that we can hold out a subset of the sample to see how well the model can predict into the held out sample. This procedure involves partitioning the sample into two disjoint sets which in the literature are called the test and train sets. Training the model in this context means calculating optimal coefficients. Next, we create predicted values of the dependent variable y, call these estimates $\hat{y}$, using the testing set. To determine how well a candidate model performs, we compare how much the true values y differ from the estimates of $\hat{y}$. The most common way to calculate this difference is called mean squared error (MSE).[8] Lower values of MSE indicate that the model has done a better job at predicting the variable of interest y.

$$MSE = \frac{1}{T}\sum_{i=1}^{T}(y_i - \hat{y}_i)^2 \tag{2.6}$$

In selecting between different models, we should select the model with the lowest corresponding MSE if we want to maximize predictive accuracy.

[8] Hastie *et al.* (2013)

However, the validation set approach is not an ideal way to measure prediction performance because it involves randomly partitioning the data set into two disjoint sets. This means that the MSE will usually have significant variation each time we run the algorithm. This happens because the model is typically only trained on half of the data and larger samples usually lead to more accurate predictions. It could also be the case that the testing half of the data could be randomly difficult to estimate.

To account for this variation and stabilize the MSE estimates the ML literature has developed a similar resampling method called k-fold cross-validation. This method involves partitioning the data set into k disjoint sets of equal size. We then train the model on the (k-1) folds and then predict into the k^{th} fold. We then repeat this procedure k times. This allows us to get predicted values for each observation in our data set. Finally we calculate the MSEs for each fold and then take an average to get a cross-validated MSE. We want the MSE values for each model to have low variation because we will use this metric to choose among a host of candidate models. We want to be sure we are picking the best model and MSE estimates with low variation will increase our confidence. The cross-validated MSE calculation is represented below in equation (2.7).

$$CV_k = \frac{1}{k}\sum_{i=1}^{k}(MSE)_i \tag{2.7}$$

A major advantage of this method over the validation set approach is that we are able to get estimates for the whole data set. Using the validation set approach there will always be a subset of observations left out, typically 30% or 50%. Another advantage of k-fold cross-validation is that its MSE

estimates have less variation. This is because the overall MSE is averaged across each of the k folds. For the machine learning models described below, I use k = 10. This is computationally inexpensive compared to other methods and is popular in the applied literature.

K-fold cross-validation is similar to information criterion used in econometrics and statistics. These information criterion are typically used for variable selection and k-fold cross-validation is a more general version of an information criteria. Forward, backward, and mixed stagewise variable selection are popular tools in applied machine learning. Popular information criterion include, AIC and the BIC mentioned earlier. With this background information, I can describe how I have implemented these models into the VAR structure.

2.4 VARs

Conceptually, a VAR is just a system of linear equations. These equations predict a column vector of dependent variables using linear combinations of the lagged independent variables. In this context, a lagged variable is a variable that is one or more periods behind the current variable's value. Intuitively, a VAR describes how today's variables of interest, perhaps GDP or unemployment, depend on other variables as well as their own past values.

In my procedure I do not include any exogenous variables, but this can be done. I have chosen to do this because I am most interested in comparing the OLS IRs to the ML IRs and not accounting for omitted variable bias. Since I test these IRs using simulation, adding in more variables into the model would only serve to improve the legitimacy of the theoretical model.

The simulations will serve to determine relative performance and not absolute performance.

In the following notation, a VAR(p) process contains n endogenous variables and p lags. Intuitively, we can interpret (2.8) as how this period's endogenous variables X_t depend on linear combinations of previous period's endogenous variables X_{t-T}, an intercept term Υ, and an error term ϵ_t. [9] We assume ϵ_t to be normally distributed with mean zero and to have variance/covariance Ω.

It is important to note here that the coefficient matrices ϕ_i have dimension of nxn and this will be important when we convert this VAR(p) process into an AR(1) process. Equation (2.8) allows us to determine all of the parameters that we can empirically estimate. [10]

$$X_t = \begin{bmatrix} x_{1t} \\ x_{2t} \\ \vdots \\ x_{nt} \end{bmatrix} ; \Upsilon = \begin{bmatrix} \upsilon_1 \\ \upsilon_2 \\ \vdots \\ \upsilon_n \end{bmatrix} ; X_t = \Upsilon + \sum_{i=1}^{p} \phi_i X_{t-i} + \epsilon_t \; ; \; \epsilon_t \sim iid\mathcal{N}(0, \Omega) \quad (2.8)$$

After we get the empirical estimates from equation (2.8) we can recover the parameters from the structural model. By structural model, I mean the data generating process which we assume to be a VAR(p). This means that we can in principal correctly specify the model. Equation (2.9) describes the structural VAR(p) model. In order to identify all of the parameters in (2.9)

[9] The notation I use is a combination of Lütkepohl (2005), Cochrane (2005), and Hamilton (1994).

[10] We also assume that error terms are not serially correlated , $E(\epsilon_t \epsilon'_{t-T}) = 0$ for T≠0.

we need to make assumptions about some of the parameters. This happens because we have more structural parameters to estimate than parameters we can estimate (Cochrane, 2005).

$$\theta X_t = \alpha + \sum_{i=1}^{p} \psi_i X_{t-i} + e_t \tag{2.9}$$

To make (2.9) look like the empirical model (2.8) we can left multiply by θ^{-1}. This leaves us with (2.10).

$$X_t = \theta^{-1}\alpha + \sum_{i=1}^{p} \theta^{-1}\psi_i X_{t-i} + \theta^{-1} e_t \tag{2.10}$$

Now we can just match up the theoretical matrices with the empirical matrices see equation (2.11). Once we do this we have fully estimated the VAR(p).

$$\theta^{-1}\alpha \; = \; \Upsilon \; ; \; \theta^{-1}\psi_i \; = \; \phi_i \; ; \; \theta^{-1} e_t \; = \; \epsilon_t \tag{2.11}$$

Taking the estimated VAR(p) we can calculate IRs. There are a few different ways to calculate IRs and I have chosen to use the method from Sims (1980). It can be shown that any time-series can be represented as an arbitrary linear combination of IRs and we need a way of determining which combinations to analyze. Sims (1980) gives a convenient way to do this. His method involves orthogonalizing Ω which forces the endogenous variables to be uncorrelated. This method is popular in the literature because it removes the contemporaneous effects and exogenous shocks from other variables. This means that the shocks are orthogonal. This assumption is consistent with the many theoretical macroeconomic models. To do this, we impose that the off diagonals of the variance-covariance matrix Ω are zero. This leaves Ω

with the variances of the endogenous variables on the main diagonal.

To make the computations easier I have chosen to simplify (2.8) into an AR(1) process. I do this by putting the coefficient matrices next to one another (column binding) and this results in a new coefficient matrix F. I_n in (2.12) is the identity matrix of dimension nxn. This results in a new AR(1) process $\xi_t = F\xi_{t-1} + V_t$ (Hamilton, 1994). This is useful because it will force F to be square (same number of rows as columns) and this will allow us to preform certain matrix operations later. The notation used in (2.12) is called space-state notation in the literature (Hamilton, 1994).

$$F = \begin{bmatrix} \phi_1 & \phi_2 & \cdots & \phi_{p-1} & \phi_p \\ I_n & 0 & \cdots & 0 & 0 \\ 0 & I_n & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & I_n & 0 \end{bmatrix} ; \xi_t = \begin{bmatrix} x_t \\ x_{t-1} \\ \vdots \\ x_{t-p+1} \end{bmatrix} ; V_t = \begin{bmatrix} \epsilon_t \\ 0 \\ \vdots \\ 0 \end{bmatrix} ; E(V_t V_s') = \begin{bmatrix} \Omega & 0 & \cdots & 0 \\ 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \cdots & \vdots \\ 0 & 0 & \cdots & 0 \end{bmatrix} \tag{2.12}$$

After estimating the system above we can calculate IRs. There are a few different ways to do this and I have chosen the structural approach.[11] The following approach is structural because we use economic theory to figure out the contemporaneous links between the variables in the system (Stock and Watson, 2001). To make these temporal assumptions I use the method from Sims (1980). Other popular methods include Bernanke (1986) and

[11] There are two other types of VARs; recursive and reduced form. I have chosen the structural approach because I have the most experience with it compared to the other approaches.

Blanchard and Quah (1989). In principal, there are an infinite number of ways to make these identifying assumptions because all we need to do is fix the temporal relationships of some of the variables. In applied work, the identification scheme will often depend on the results of the associated theoretical literature. I choose to use the method from Sims (1980) because it is the most popular way to make identifying assumption and it is easier to implement than other methods.

To implement the method from Sims (1980) we take the Cholesky decomposition of the variance covariance matrix to specify the contemporaneous effects of the variables. This decomposition produces a matrix Q such that $QQ' = \Omega$. For this decomposition to work, we need Ω to be positive-definite and we are guaranteed this since $\Omega = \Omega'$ and Ω's eigenvalues will always be positive since its main diagonal contains strictly positive values. The main diagonal of Ω contains the variances of the endogenous variables and these are necessarily non-negative. To put this into space-state notation, I take the inverse of the Cholesky decomposition of Ω and then put that matrix into the larger Q matrix. I form the Q matrix in this way to be able to take its inverse later.

$$\Theta = \Big\{ Cholesky\_Decomposition(\Omega) \Big\}^{-1} \; ; \; Q = \begin{bmatrix} \Theta & 0 & \cdots & 0 \\ 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \cdots & \vdots \\ 0 & 0 & \cdots & 0 \end{bmatrix} \tag{2.13}$$

An interesting consequence of this matrix decomposition is that the

ordering of the variables matters since we are restricting contemporaneous effects. This can be seen in (2.14) because the example matrix has zeros on its upper diagonal. To determine how to order the variables, people usually appeal to economic theory to choose an ordering. Reestimating results using alternative orderings is a popular robustness check.

$$\begin{bmatrix} 1 & 0 & 0 \\ 2 & 4 & 0 \\ 3 & 5 & 6 \end{bmatrix} \tag{2.14}$$

To calculate IRs, we can move an exogenous shock (2.15) through the system. This shows how the system will respond to a positive one standard deviation impulse to the first variable x. Also, notice that all other shocks are set to zero. The applied literature frequently uses an impulse of one standard deviation, but this choice is arbitrary.

$$e_0 = [SD(x)\, 0 \cdots 0]' \tag{2.15}$$

To calculate the 0^{th} point estimate, also called the initial impact, we left multiply e_0 by Q'. For future point estimate, we just take the previous period's estimate and left multiply by F. This method of calculating IRs has been a major source of criticism of the VAR method because any miss-specification of F will become compounded into future periods. This happens because we keep taking powers of F, so a small misspecification in the first period could lead to larger errors in future periods. See (2.16) below. This is part of the motivation behind Jordà's local projection method, because his method does not compound misspecification errors.

$$IR_k = \{Q'e_o,\, QFQ'e_0,\, QF^2Q'e_0,\, QF^3Q'e_0, \cdots,\, QF^kQ'e_0\} \tag{2.16}$$

So far, the econometric theory that I have described is what has been used for OLS VARs for the past thirty odd years. The ML VARs and subsequent IRs are identical to the OLS VARs except for the F coefficient matrix. To get F people usually used the closed form solution $F = (X'X)^{-1}X'Y$.[12] To estimate F using ML models, I estimate a regression for each row of F. I do this n times to get the coefficients and then I stack each row on top each other. I have to build the coefficient matrix in this manner because all of the machine learning models I consider are designed for a single dependent variable instead of a vector. In future chapters, when I refer to a ML model for example, Ridge I am referring to the Ridge VAR model.

This process took a substantial amount of time to code because I have to estimate the coefficients and then place them into F in a particular order. I have put an example of code I created in the appendix figure (7.6).

The other important component of IRs that I have not mentioned yet is the confidence intervals around each IR point estimate. In OLS IRs, confidence intervals can be calculated using a closed form asymptotic calculation. However, I am not able to do this for the ML IRs because the econometric/statistical theory would be difficult to derive for each model. As an alternative approach, I calculate confidence intervals using the bootstrap resampling method explained in the next section.

[12]There are numerical ways to solve for F like Gradient Descent, but these methods are not as popular as the closed form solution.

2.5 Bootstrapping

I have chosen to calculate confidence intervals for the ML IRs using a version of the bootstrap instead of using an asymptotic method. Computing asymptotic confidence intervals for the machine learning models would be very difficult because I am unaware of any literature that addresses the consistency of these models for a VAR DGP. Additionally, the bootstrap method is relatively easy to implement.

The bootstrap from Efron (1980) is a data resampling method that can be used to calculate the variation of a particular statistic. This method involves taking random sub-samples of the data, calculating a statistic, and then calculating the variation of the statistic of interest. The idea is that we can treat the sample as the population and then take samples of the sample. This requires that we have have a representative sample of the population to begin with. Traditionally, the bootstrap has used random sampling for IID (independent and identically distributed) data. However, since the data that I use are time-series data they are not IID and I should not use the traditional bootstrap. I considered using the IID bootstrap method after correcting for nonstationarity, but even strong stationarity does not imply that the data are IID.

Luckily, the statistics literature has developed bootstrapping methods for time-series data. I use the the stationary bootstrap from Politis and Romano (1994). This is a block bootstrapping method that uses block lengths of geometrically distributed lengths. A block, in the temporal sense, is a consecutive set of observations that is used to calculate the statistic of interest. For example, a block of length 8 could be a monthly measure of CPI from

(7/1/2016) to (3/1/2017).

The statistics I calculate are the point estimates that comprise an IR. The stationary bootstrap uses block sizes which are distributed geometrically. The mean block size is taken exogenously in this method. The block size is chosen randomly and then a subset of observations which will make up the block is chosen randomly. The statistic of interest is then calculated using this block. This method will give consistent estimates of the ML IR confidence intervals and will account for serial correlation. To implement this method, I use the Python package 'Arch 4.0'.

I got the idea to use a block bootstrapping method from Killian and Kim (2011). Killian and Kim use a block bootstrapping method from Killian (1998), but their approach is very similar to mine. They use an average block size of 8 for their bootstrapping method. I use an average block size of 12, but I have tried 8 and it does not make a meaningful difference. I chose to use 12 because the data that I use in chapter (4) are monthly and one year blocks seemed like a sensible choice. Killian and Kim use 1,000 replications and I have done the same.

Ideally, I would have preferred to select the average block size in a more rigorous way, but I was not able to find software to do this. There have been papers such as Hall *et al.* (1995), Lahiri (2003), and Politis and White (2004) that address how to choose an optimal block size. In the future, it would be great if these statistical methods were coded into a user friendly package. As of this thesis, I am not aware of a Python program that implements any of these block size selection methods.

The machine learning models that I use are biased and to account

for this I do a bias correction within the confidence intervals. The Python package 'Arch 4.0' has a variety of ways to calculate confidence intervals and I use the bias corrected version. This bias correction will increase the size of the confidence intervals and this will account for the bias of underlying estimator. The bias correction is as follows. Let B be the total number of resamples, I use 1000, and θ be the true statistic of interest. $\hat{\theta_{(.)}}$ is the average of the bootstrap estimates, shown in equation (2.17).

$$\hat{\theta_{(.)}} = \frac{\sum_{b=1}^{B} \hat{\theta}_b}{B} \tag{2.17}$$

The bias of the $\hat{\theta}$ is $E[\hat{\theta}]-\theta$. We then replace, $E[\hat{\theta}]$ with $\hat{\theta_{(.)}}$ and θ with $\hat{\theta}$. This means we can express the bias of the bootstrap estimate as $Bias_{bs}[\hat{\theta}] = \hat{\theta_{(.)}}-\hat{\theta}$. Combining these pieces, the bias corrected estimate is (2.18).

$$\hat{\theta_{bs}} = 2\hat{\theta} - \hat{\theta_{(.)}} \tag{2.18}$$

I use this bias correction for every IR in the following chapters. This particular bias correction comes from Efron (1987).

Chapter 3

Simulations

3.1 Set Up

In this chapter, I describe a simulation scheme that could be used to compare the ML and OLS IRs. Simulations have been used in the past as a way to compare IR estimators. However, since IRs are inferential statistics their true values need to be known to determine relative performance. In the beginning of this project, I wanted to use an alternative performance metric, such as cross-validated MSE, to compare the IR estimators. One of the benefits of using a metric like cross-validate MSE is that it does not require a theoretical model. In the future, it would be interesting to see if there is a way to compare inferential statistic estimators only using realized data instead of using simulated data from a theoretical model.[1]

I was able to find two similar papers, Killian and Kim (2011) and Meier (2005), that compare IR estimators using simulations. In both papers,

[1] I am skeptical that such a procedure exists, but this would be worth thinking about.

the authors compare local projections to VAR models. They compare these competing models using simulations because they need to know the true IR values. To generate these true IR values the authors specify a DGP. For my simulations, I use a DGP from Killian and Kim (2011).

In their paper, Killian and Kim compare Jordà's local projections to OLS VAR models. They want to see if the local projections method for calculating IRs is closer to the DGP's IRs compared to the OLS VAR's IRs. Local projections have been thought to be better than OLS VARs because they require fewer assumptions, but this does not necessarily imply that their corresponding IRs are better than the OLS VAR's IRs. To compare the models, Killian and Kim use average coverage rate and average confidence interval length. The average coverage rate is how often the true IR point estimate lies within the estimated confidence interval. The average confidence interval length is just the average difference between the upper and lower confidence interval estimates. They need to use both metrics because otherwise the model with the highest or lowest variance would win depending on the metric. For example, using average confidence interval length the model with the lowest variance would seem like the best because it would have tight confidence intervals. Conversely, using average coverage rate the model with the highest variance would win because it would have wide confidence intervals and always capture the true IR. The optimal IR, has a high coverage rate and tight confidence intervals. This is why Killian and Kim use both metrics and I follow their approach.

The authors test two types of confidence intervals asymptotic and bootstrap. They ran three rounds of comparisons using three different DGPs:

VAR(1), VAR(12), and VARMA(1,1). Overall, they found that the local projection intervals are less accurate than the VAR intervals. They found this result is robust to the VARMA(1,1) specification and different sample sizes.

While Killian and Kim do use three different DGPS, they take the VAR(12) DGP the most seriously. The other two DGPs, VAR(1) and VARMA(1,1), only contain 1 lag and they admit this is not realistic. We would expect a 'good' model to have many lags because large exogenous shocks can have long lasting effects. They use these two DGPs, and associated simulations, mainly as robustness checks. For this reason, I use their VAR(12) DGP. This DGP is calibrated to simulate monetary policy shocks and is relevant for my purposes because in chapter (4) I estimate the effects of monetary policy shocks. The model that Killian and Kim use comes from Christiano *et al.* (1999). The motivation for this model is that the Federal Reserve chooses interest rates based on previous values of inflation and the output gap. Additionally, this model assumes no contemporaneous feedback from policy decision to the output gap, inflation rate, or commodity prices. For this model, only the monetary policy shock is identified. This means that I only consider four of the possible 16 IRs. [2]

The model is calibrated using monthly data from January 1970 to December 2007. The data that Killian and Kim use includes a measure of inflation, interest rates, the output gap, and industrial commodity prices. Killian and Kim augment the model in Christiano *et al.* (1999) by including industrial commodity prices. They do this because some theoretical literature

[2]For a VAR with n endogenous variables there a n^2 possible IRs.

Table 3.1: Summary Statistics for Realized Data (Killian & Kim, 2015)

Variable	Mean	Std.Dev.	Range
CFNAI	.00438	1.008	(-5.29, 2.67)
CPI	120.24	51.28	(37.7, 211.68)
Raw Industrials Index	264.89	72.40	(104.4, 494.82)
Federal Funds rate	6.573	3.41	(.98, 19.1)

has suggested that industrial commodity prices may be a leading indicator of inflationary pressures. To measure this inflationary indicator they use the change in the Commodity Research Board's price index for raw material adjusted for inflation. For inflation Killian and Kim use the seasonally adjusted CPI for all urban consumers. The real output gap is measured using the Chicago Fed National Activity Index (CFNAI) and the Federal Funds rate measures interest rates. Summary statistics are reported in table (3.1) for the calibration data.

Killian and Kim programed their simulations in Matlab whereas my programs are written in Python. So to use their simulated data, I collected 1,000 trials of their data with each trial having 456 observations into a .csv file. This results in a data set of 456,000 observations. To get the simulated data, I ran their Matlab files and saved the data that they use to estimate the IRs. This allows me to import the data into Python and run my scripts.

Killian and Kim generate their simulated data by running randomly generated observations through the calibrated VAR(12) model. These random observations are normally distributed with variance equal to that of the realized data from table (3.1). To see more details of their DGP, see the VAR(12) Matlab file link here from Harvard's Dataverse repository. In

calibrating their DGP they test for nonstationarity and make corrections.[3] Even though they correct for nonstationarity, the simulated data that they feed into their models is substantially nonstationary.

$$y_t = \alpha + \delta t + \phi y_{t-1} + \epsilon_t \tag{3.1}$$

To test for nonstationarity, I ran augmented Dickey-Fuller tests to test for unit roots and linear time trends. Equation (3.1) illustrates the model that the augmented Dickey-Fuller test estimates.[4] This test assumes that the null hypothesis for ϕ is one. The null hypothesis for the trend t term is that δ is zero. The results of these test are summarized in table (3.2). Notice that the p-values for ϕ for CPI and Federal Funds rate are much higher than 5%.[5] First, I checked to see if using another set of simulated data with 456 observations might change the stationarity results. Surprisingly, I found that these results held for other simulated data sets. This means that the data Killian and Kim are feeding into their VAR and local projection models is significantly nonstationary.

I found this result very surprising because Killian and Kim correct for nonstationarity in calibrating the VAR(12) DGP. After carefully going over their Matlab code, I am unsure why this is happening. Something strange seems to be happening because I would not think that adding in Gaussian errors to a stationary model would not produce nonstationary data. This result is a mystery to me and since both VAR and local projection models require stationary data their results may be inaccurate. For my simulations,

[3] They take the first difference of CPI and the Federal Funds rate.
[4] StataCorp (2015)
[5] This is a popular p-value used throughout economics.

Table 3.2: Stationarity Results for Simulated VAR(12) Data

Variable	P-Value for ϕ	P-Value for δ
CPI	0.2551	0.599
1st Difference of CPI	0.0001	0.521
Raw Industrials Index	0.0047	0.872
CFNAI	0.0216	0.226
Federal Funds rate	0.2220	0.815
1st Difference of Federal Funds rate	0.0017	0.920

I correct their data for nonstationarity by differencing CPI and the Federal Funds rate.

To run the simulations, I have broken the computations into two parts and this should increase computational efficiency.[6] The first part of the simulation calculates ML and OLS IRs using the corrected Killian and Kim data. I do this by taking a single data set containing 456 observations from the main data set of 456,000 observations. Next, I calculate OLS and ML IRs using these 456 observations. Lastly, I save the IRs into a multidimensional array. To do this computationally, I loop through 456,000 observations 1,000 times taking 456 observations each time. In the second part, I calculate average coverage rates and average confidence interval lengths. I do this by looping through each of the 1,000 trials.

[6] I say should here because I have not tried combining these these two pieces into one. However, loops are computationally expensive and the less that I can do within them the better.

3.2 Results

In future work, I intend to work on implementing the simulation scheme described above. I anticipate that the programming necessary to carry out this section will be technically difficult. Additionally, these simulations will require substantial processing power because each ML model is computationally more expensive than OLS. I intend to use the Bates College High Performance Computing Cluster (Leavitt) to assist with the computations.

Chapter 4

The Effects of Quantitative Easing

In this chapter, I apply the ML VAR models developed in chapter (2) to U.S. macroeconomic data. I use this data to analyze how Quantitative Easing affects the real economy with focus on unemployment. This idea comes from Shea *et al.* (2017) and I will reestimate their results using ML VARs.

Since 2008, the Federal Reserve conducted three rounds of large asset purchases, know as Quantitative Easing, in response to the Great Recession. These asset purchases dramatically increased the Federal Reserve's balance sheet to $4.5 trillion by December 2016 (Shea *et al.*, 2017). These assets consisted mostly of mortgage backed securities and long term Treasury Bonds rather than short term Treasury Bonds. These asset purchases are a form of unconventional monetary policy. Shea *et al.* (2017) attempts to estimate the effects that these asset purchases had on the real economy. Surprisingly,

they find that these asset purchases led to small, but significant increase in unemployment and a decrease in inflation (Shea *et al.*, 2017).

The Federal Reserve indicated ways in which Quantitative Easing could reduce unemployment by stimulating aggregate demand. These include removing risky mortgage backed securities off of private firm's balance sheets, reducing long term interest rates, and increasing access to credit markets (Shea *et al.*, 2017). From these possible explanations, we would expect Quantitative Easing to decrease unemployment through increases in aggregate demand. The associated theoretical literature finds similar results. The literature suggests that Quantitative Easing should at worst have limited effects on unemployment through changes in aggregate demand (Woodford, 2003; Chen *et al.*, 2012; Wen, 2014).

To estimate the effects of Quantitative Easing, Shea *et al.* use a standard four variable VAR(4) model to estimate the impact of Quantitative Easing on important macroeconomic variables.[1] Their data consists of the Federal Reserve's balance sheet, 10 year Treasury bill rate, (U-3) unemployment rate, and CPI. They use the Fed's balance sheet to proxy for quantitative easing.[2] These data come from the Federal Reserve Bank of St. Louis's Economic Database (FRED) and were measured monthly from January 1, 1970 to October 1, 2015. The Federal Reserve's balance sheet is constructed by Gresham Law Database that extracts historical balance sheet data. They test for non-stationarity using augmented Dickey-Fuller tests and find a single unit root for each variable. Thus, they use the first difference of these

[1] They use AIC to choose a lag length of 4.

[2] Asset purchases will increase the Fed's balance sheet and makes this an appropriate proxy.

Table 4.1: U.S. Macroeconomic Data Summary Statistics

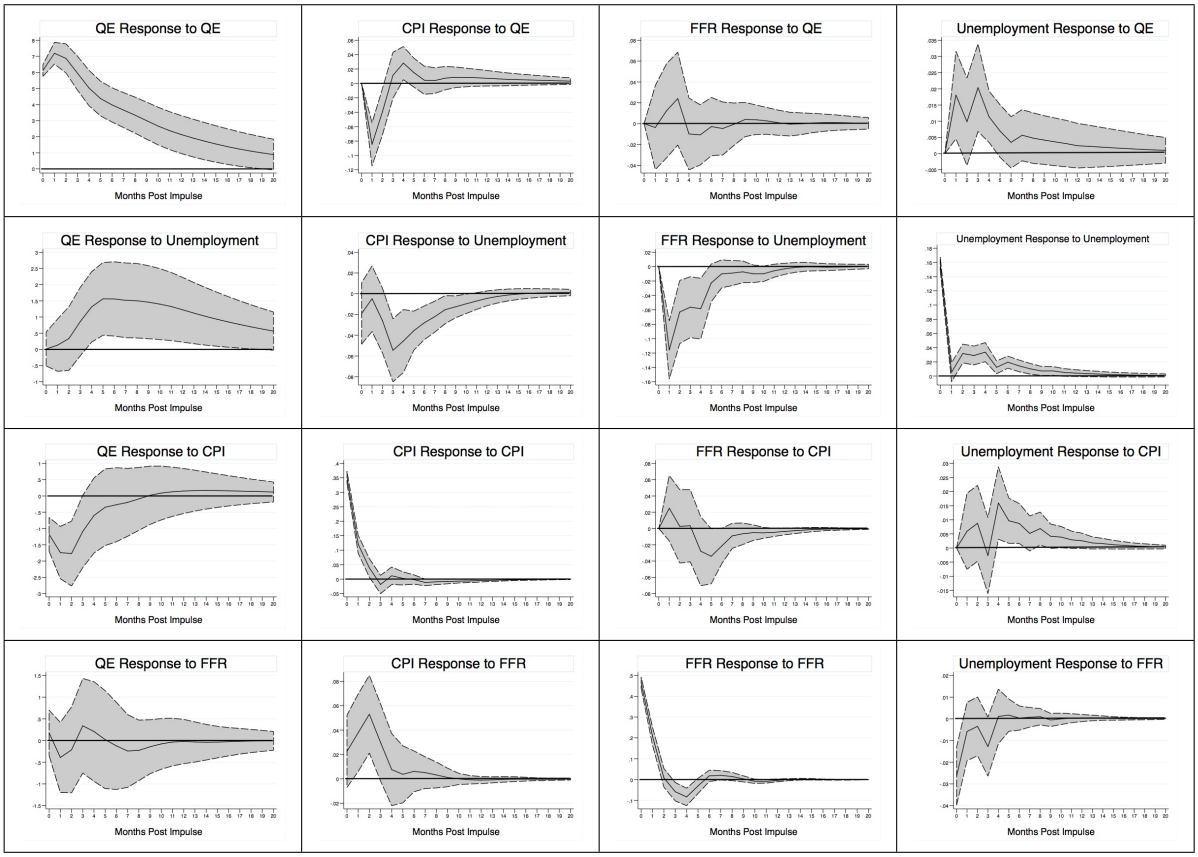
Variable	Mean	Std.Dev.	Range
CPI	4.19	3.01	(-2 - 14.6)
Unemployment	6.38	1.54	(3.8, 10.8)
Treasury Bill	6.69	2.92	(1.53, 15.32)
Federal Reserve's balance sheet	10.25	18.43	(-5. 150.9)
Federal Funds rate	5.5	3.9	(0.07, 19.1)

variables. See table (4.1) for summary statistics.

Shea *et al.* (2017) uses Stata 14 to fit an OLS VAR model to the data summarized above. The authors choose to order the endogenous variables as follows: inflation, Fed's balance sheet, unemployment, federal funds rate, and Treasury Bill rate. They try using other orderings, but find that they do not significantly change the IRs. I use this ordering in the ML VARs as well. To see how the OLS VAR behaves, we can look at its associated IRs in table (4.2). Each of the IRs converges back to zero within the first 20 periods after the impulse and the confidence intervals are not explosive. Theoretically, we would not want exogenous shocks to have infinite effects. We would expect the effect of shocks to eventually converge to zero. This is a good sign, because it indicates that the system is well behaved and not obviously nonstationary.

To see how Quantitative Easing affects unemployment, Shea *et al.* isolate one of the IRs from table (4.2). They find that an increase Quantitative Easing leads to a statistically significant increase in the unemployment rate. This result is illustrated in figure (4.1). This is an interesting result because it is contrary to what intuition and theory would predict. Shea *et al.* (2017) indicates that there is no formal theoretical work that shows this effect. They

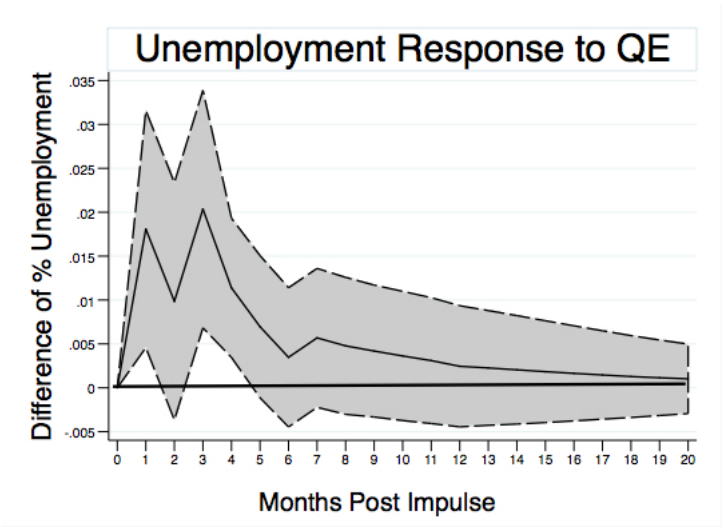
Table 4.2: OLS VAR Impulse Response Functions (Shea *et al.*, 2017)



offer the following explanation. Quantitative Easing may have increased inflationary expectations and when these expectations were not realized this gap could have resulted in decreased output. This comes from the New Keynesian idea, that a decrease in output could occur if inflation does not attain its expected value (Woodford, 2003).

In figure (4.1), notice that the lower confidence interval estimates are above zero. In the paper, Shea *et al.* test the robustness of this result to alternative specifications and find that the result holds. For the robustness

Figure 4.1: How Unemployment Responds to QE (Shea *et al.*, 2017)



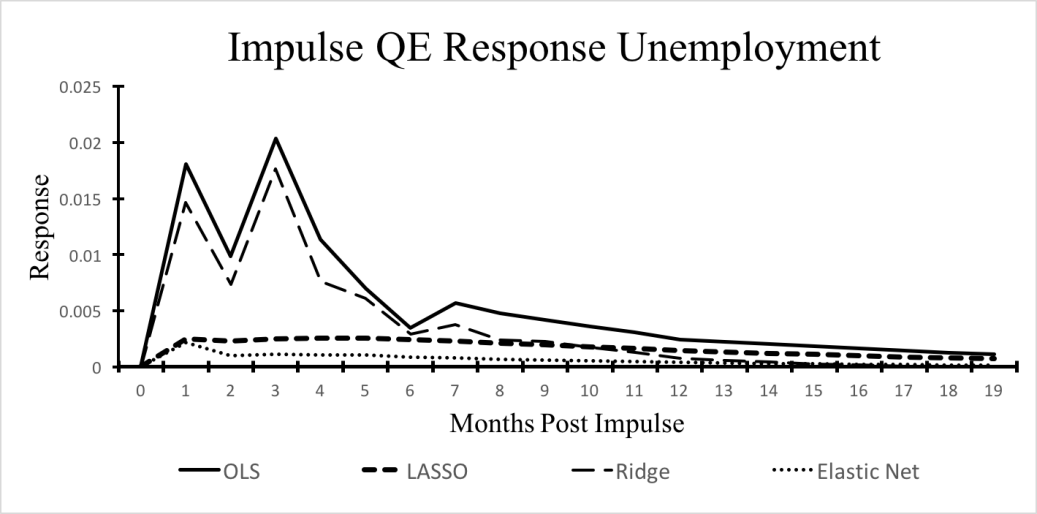
checks, they include a set of exogenous variables to test for serious omitted variable bias.

Using the ML VAR models from chapter (2) and data summarized in table (4.1), I reestimate the most important IRs.[3] To do this, I run the stationary version of the data through each ML model and calculate IRs. I fix the maximum lag length that the ML VARs can consider at 8 lags each because two of the three ML models perform variable/lag selection.

Figure (4.2) displays the point estimates for how unemployment responds to Quantitative Easing by model type. The Quantitative Easing impulse corresponds to a 1SD or 18.43% increase in the Federal Reserve's balance sheet. The first thing that jumped out to me from the figure was that the ML IRs lie strictly below the OLS IR. This is a good sign because this is theoretically what should be happening. Since each ML model is itself

[3]These being the impulse to Quantitative Easing.

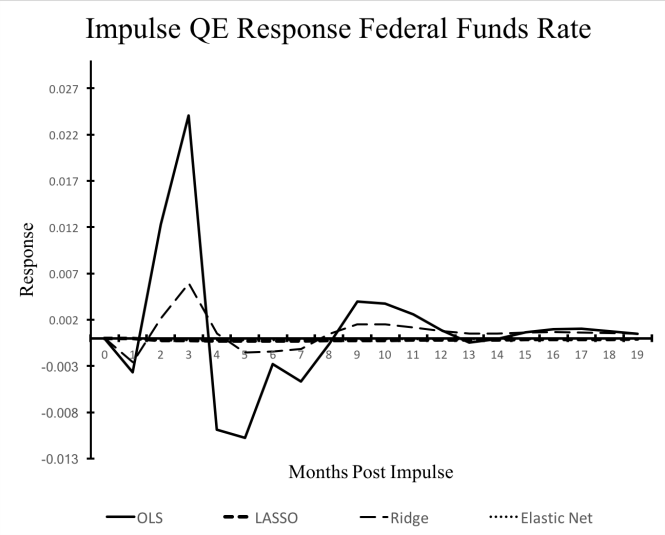
Figure 4.2: Point Estimates of How Unemployment Responds to QE



a constrained version of OLS this implies that the ML coefficients will be smaller in magnitude than their OLS counterparts. These coefficients are formed into the Q matrix from (2.14) and multiplied in (2.16) to calculate IR point estimates. Since, each ML coefficient is no larger than its OLS counterpart, it follows that the IR point estimates will be smaller in magnitude. This is great to see because it indicates that the ML VAR models are behaving as we would expect them to.

From figure (4.2) we see that the point estimates of the Ridge model are very similar to the OLS point estimates. This is unsurprising since Ridge is the most similar of the three ML models to OLS. Ridge is different from LASSO and Elastic Net in that it only performs coefficient estimation and not variable selection. The LASSO and Elastic Net point estimates are much smaller in magnitude. Compared to OLS and Ridge, the LASSO and Elastic Net point estimates are conservative because they indicate much smaller

Figure 4.3: Point Estimates of How Federal Funds Rate Responds to QE

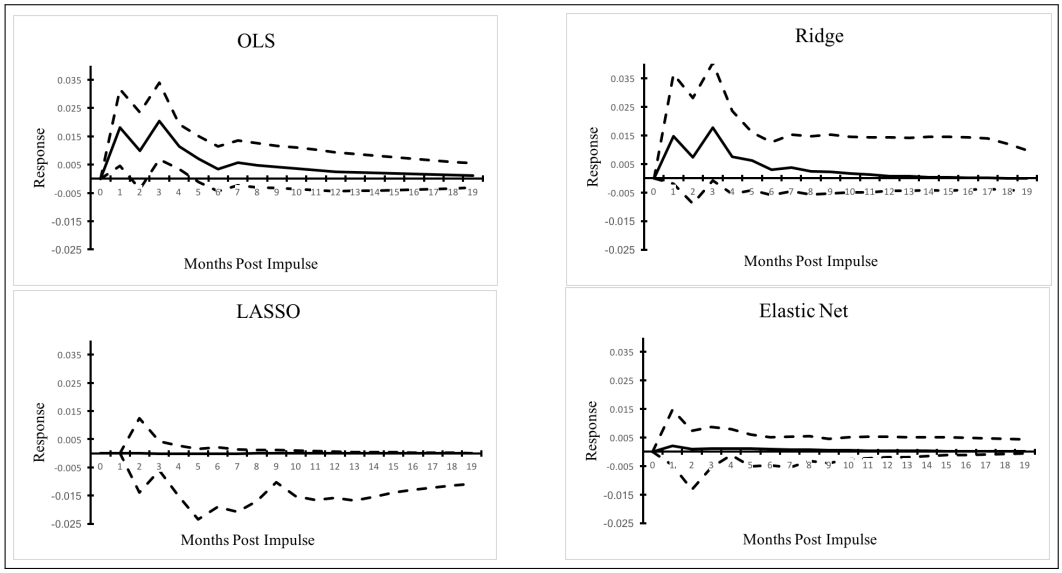


marginal effects. I think this is happening because LASSO and Elastic Net set some of the coefficients to be exactly zero and this could wipe out any effect.

Figure (4.3) is identical to figure (4.2) except that it reports how the federal funds rate responds to Quantitative Easing. The Ridge point estimates are the closest to the OLS point estimates just like in (4.2). The LASSO and Elastic point estimates are near zero and this indicates a small marginal effect. The point estimates for how Quantitative Easing responds to Quantitative Easing are displayed in figure (7.4) of the appendix. The point estimates for how the Treasury bill rate responds to Quantitative Easing are displayed in figure (7.5) of the appendix.

I chose to only plot the point estimates for figures (4.2) and (4.3) because the confidence intervals would overlap, but they are important. In figure (4.4), I display the four IRs from figure (4.2) with their confidence

Figure 4.4: IRs for How Unemployment Responds to QE by ML Type



intervals. I use the bias corrected stationary bootstrap method from chapter (2) to calculate the confidence intervals. The confidence intervals for Ridge are larger than the OLS confidence intervals. At first I found this counterintuitive because ML models have lower variance since they are biased. This means that the confidence intervals should be smaller than their OLS counterparts. However, the bias correction increases the size of the confidence intervals. The bias correction explains why the confidence intervals for the ML models could be larger than the OLS confidence intervals. If I did not do a bias correction, the confidence intervals would be smaller that the OLS estimates.[4] The other three IR point estimate graphs are reported in figures (7.1), (7.2), and (7.3) of the appendix.

To see how these IRs are generated, we can look at each model's co-

[4]I have calculated confidence intervals without a bias correction and the confidence intervals are smaller than the OLS estimates.

efficient matrix. These matricies are displayed in tables (4.3), (4.4), (4.5), and (4.6). Each matrix represent the first row of the F matrix from equation (2.12) for each model. From table (4.4) we see that the Ridge estimates are very close to the OLS estimates in table (4.3) and this is consistent with the IRs above. Since the IRs are close we would expect the coefficient estimates to look similar.

Tables (4.5) and (4.6) show the coefficient estimates for the LASSO and Elastic Net respectively. In chapter (2), I described that LASSO and Elastic Net will force some of the coefficients to be exactly zero and we can now see this in action. I have highlighted in blue the coefficients that are exactly zero. The LASSO has more zero entries compared to Elastic Net and this is what we would expect because Elastic Net is a generalized version of Ridge and LASSO.

The coefficient matrices from the LASSO and Elastic Net could be used to supplement the identifying assumptions from section (2.4). The identifying assumptions specify the temporal relationships between the variables and these coefficient matrices give empirical estimates of just that. For example, notice that the first entry in the second row of table (4.5) is zero. This implies that the Federal Reserve's balance sheet does not depend on contemporaneous effects from inflation. A researcher could use this result as an identifying restriction rather than relying solely on economic theory. This idea could be used in future work as an empirical way to make identifying assumptions.

Table 4.3: OLS Coefficient Matrix

0.320	-0.616	-0.001	0.503	-0.161	-0.028	0.006	-0.072	-0.082	-0.263	-0.001	-0.014	-0.141	-0.269	-0.001	0.131
-0.003	0.028	0.003	-0.044	0.009	0.185	-0.002	0.005	-0.028	0.184	0.002	0.047	0.020	0.153	-0.002	0.025
-1.446	1.051	1.189	0.965	0.996	0.749	-0.281	1.402	0.854	3.086	-0.053	-0.956	-0.857	2.047	0.049	-0.131
0.001	-0.176	-0.001	0.378	0.056	0.064	0.002	-0.283	-0.048	0.098	-0.003	0.091	0.064	-0.001	0.003	-0.058

Table 4.4: Ridge Coefficient Matrix

0.324	-0.406	-0.001	0.419	-0.143	-0.048	0.005	-0.038	-0.067	-0.181	-0.001	-0.023	-0.117	-0.179	-0.001	0.098
-0.003	0.028	0.003	-0.044	0.009	0.184	-0.002	0.005	-0.028	0.183	0.002	0.046	0.020	0.152	-0.002	0.025
-1.326	0.909	1.192	0.863	0.835	0.616	-0.282	1.060	0.693	1.888	-0.053	-0.698	-0.828	1.358	0.052	-0.218
0.009	-0.111	-0.002	0.291	0.038	0.034	0.002	-0.197	-0.035	0.054	-0.003	0.040	0.053	0.001	0.003	-0.033

Table 4.5: LASSO Coefficient Matrix

0.205	-0.281	0	0.321	0	0	0	0	-0.004	0	0	0	0	0	0	0
0	0	0.001	-0.007	0	0.140	0	0	0	0.111	0	0	0	0.093	0	0
-0.349	0	0.935	0	0	0	0	0.250	0.050	1.230	-0.031	0	0	1.549	0	0
0	-0.041	0	0.251	0	0	0	-0.100	0	0	0	0	0	0	0	0

Table 4.6: Elastic Net Coefficient Matrix

0.179	-0.362	0	0.306	-0.024	-0.002	0	0	-0.043	-0.043	0	0	-0.049	-0.066	0	0
0	0.011	0.001	-0.003	0	0.104	0	0	0	0.081	0	0	0	0.072	0	0
-0.608	2.670	0.441	0.126	0	2.402	0.203	1.207	0.640	3.551	0.080	0.440	-0.124	3.913	0.045	-0.168
0	-0.056	0	0.172	0	0	0	-0.063	0	0	0	0	0.001	0	0	0

Shea *et al.* conclude their paper by indicating that the Quantitative Easing was ineffective at reducing unemployment. From a policy perspective, this is concerning because these results show that Quantitative Easing is behaving contrary to what theory and intuition would predict. Shea *et al.* do not offer any policy prescriptions and I do not know how this effect, assuming it is real, could be reduced or eliminated.

Overall, the ML IRs give more conservative estimates compared to the OLS IRs. By conservative, I mean that the ML IRs show lower marginal effects relative to OLS. Applying the ML VAR models to this U.S. data does not add any new insights to the analysis because the ML IRs are very similar to the OLS IRs. However, these results show that these ML models are well behaved. This gives me confidence in applying them to other questions and data sets.

Chapter 5

Conclusion

Vector Autoregressions (VARs) have been used in economics, finance, and biology as a popular way to analyze multivariate time-series. However, this class of models requires many restricting assumptions and there has been work done to reduce/relax these assumptions. In this thesis, I propose three new VAR models which relax the ordinary least squares (OLS) assumption. To do this, I use supervised machine learning (ML) models to estimate coefficients in place of OLS. I outline the details of the ML VAR models in chapter (2).

In macroeconomics, VARs are most commonly used on small data sets because reliable data is limited. Given this constraint, the ML VARs may be able to outperform the OLS VARs because the ML VARs could be more efficient. To test the efficiency of the ML and OLS VARs, I propose a simulation scheme in chapter (3). I hope to use this scheme in future projects. In future projects it would be interesting to use nonlinear DGPs to see if the ML VARs are able to capture the nonlinearities. An easy way to do this would

be adding in squared and or moving average terms to a VAR(p) process.

In chapter (4) I use the ML VAR models to estimate how Quantitative Easing affects the real economy. The ML VARs are well behaved and produce impulse responses (IRs) that are similar to the OLS VARs that are commonly used. Overall, the ML VARs estimate more conservative marginal effects compared to the OLS VARs and this is exactly what we would expect. This is an important contribution of this thesis. I propose three new VAR models and demonstrate that they yield sensible estimates. Other contributions of this work include using the ML VARs to make identifying assumptions and using the ML VARs to reestimate existing work. The LASSO VAR could be used as an empirical way to make identifying assumptions rather than relying soley on economic theory. Future work could be done to see if important OLS VAR results hold when using the ML VARs.

This thesis has been very rewarding and I have enjoyed the process even though it has been difficult at times. The proposal I submitted last year, September 2016, was ambitious and I was not able to complete everything I hoped I would. However, I am satisfied with the work that I completed and I look forward to testing the relative efficiencies of the ML and OLS VARs.

Chapter 6

References

Athey, Susan, and Guido W. Imbens. "Machine learning methods for estimating heterogeneous causal effects." *stat* 1050 (2015): 5.

Bajari, Patrick, Denis Nekipelov, Stephen P. Ryan, and Miaoyu Yang.*Demand estimation with machine learning and model combination.* No. w20955. National Bureau of Economic Research, 2015.

Basu, Sumanta. "Modeling and estimation of high-dimensional vector autoregressions." PhD diss., The University of Michigan, 2014.

Belloni, Alexandre, Victor Chernozhukov, and Christian Hansen. "Lasso methods for gaussian instrumental variables models." (2011).

Bernanke, Ben S. "Alternative explanations of the money-income correlation." In *Carnegie-Rochester conference series on public policy*, vol. 25, pp. 49-99. North-Holland, 1986.

Bien, J., Taylor, J. and Tibshirani, R., 2013. "A lasso for hierarchical inter actions". *Annals of statistics*, 41(3), p.1111.

Blanchard, O. and D. Quah, 1989, "The dynamic effects of aggregate demand and supply disturbances", *American Economic Review* 79, 655-673.

Bontempi, G., 2011. "Statistical foundations of machine learning". *Universit Libre de Bruxelles*.

Breiman, Leo. "Statistical modeling: The two cultures (with comments and a rejoinder by the author)." *Statistical science* 16, no. 3 (2001): 199-231.

Chen, Han, Vasco Crdia, and Andrea Ferrero. "The macroeconomic effects of largescale asset purchase programmes." *The economic journal* 122, no. 564 (2012): F289-F315.

Chernozhukov, Victor, Christian Hansen, and Martin Spindler. "Post-selection and post-regularization inference in linear models with many controls and instruments." *The American Economic Review* 105, no. 5 (2015): 486-490.

Christiano, Lawrence J., Martin Eichenbaum, and Charles L. Evans. "Monetary policy shocks: What have we learned and to what end?." *Handbook of macroeconomics* 1 (1999): 65-148.

Cochrane, John H. "Time series for macroeconomics and finance." *Manuscript, University of Chicago* (2005).

Efron, Bradley. "Nonparametric estimates of standard error: the jackknife, the bootstrap and other methods." *Biometrika* (1981): 589-599.

Efron, Bradley. "Better bootstrap confidence intervals." *Journal of the American statistical Association* 82, no. 397 (1987): 171-185.

Hall, Peter, Joel L. Horowitz, and Bing-Yi Jing. "On blocking rules for the bootstrap with dependent data." *Biometrika* 82, no. 3 (1995): 561-574.

Hamilton, J.D., 1994. *Time series analysis* (Vol. 2). Princeton: Princeton university press.

Hoerl, Arthur E., and Robert W. Kennard. "Ridge regression: Biased estimation for nonorthogonal problems." *Technometrics* 12, no. 1 (1970): 55-67.

Hsu, Nan-Jung, Hung-Lin Hung, and Ya-Mei Chang. "Subset selection for vector autoregressive processes using lasso." *Computational Statistics & Data Analysis* 52, no. 7 (2008): 3645-3657.

Jordà, Òscar. "Estimation and inference of impulse responses by local proj-

ections." *The American Economic Review* 95, no. 1 (2005): 161-182.

Kilian, Lutz. "Confidence intervals for impulse responses under departures from normality." *Econometric Reviews* 17, no. 1 (1998): 1-29.

Kilian, Lutz, and Yun Jung Kim. "How reliable are local projection estimators of impulse responses?." *Review of Economics and Statistics* 93, no. 4 (2011): 1460-1466.

Kilian, Lutz; Yun Jung Kim, 2012, "Replication data for: How Reliable are Local Projection Estimators of Impulse Responses?", hdl:1902.1/17434, *Harvard Dataverse*, V1

Lahiri, S. N. "Selecting optimal block lengths for block bootstrap methods." In *Proceedings of the 35th Symposium of the Interface*. 2003.

Lin, Jin-Lung. "Teaching notes on impulse response function and structural VAR." *Institute of Economics, Academia Sinica, Department of Economics, National Chengchi University. Source http://faculty. ndhu. edu. tw/ jlin/files/impulse. pdf (2006).*

Ljung, Lennart. "Will Machine Learning Change the Paradigm of System Identification?" *Gloverfest* 2013 Cambridge, UK (n.d.): n. pag. 2 Sept. 2013. Web. 10 Aug. 2016.

Lütkepohl, Helmut. *New introduction to multiple time series analysis*. Springer Science & Business Media, 2005.

Meier, André. "How Big is the Bias in Estimated Impulse Responses? A Horse Race between VAR and Local Projection Methods Preliminary-comments welcome." (2005).

Melnyk, Igor, and Arindam Banerjee. "Estimating structured vector autoregressive model." *arXiv preprint arXiv:1602.06606* (2016).

Nicholson, William, David Matteson, and Jacob Bien. "VARX-L: Structured regularization for large vector autoregressions with exogenous variables." *arXiv preprint arXiv:1508.07497* (2015).

Pedregosa, Fabian, Gal Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel et al. "Scikit-learn:

Machine learning in Python." *Journal of Machine Learning Research* 12, no. Oct (2011): 2825-2830.

Politis, Dimitris N., and Joseph P. Romano. "The stationary bootstrap." *Journal of the American Statistical association* 89, no. 428 (1994): 1303-1313.

Politis, Dimitris N., and Halbert White. "Automatic block-length selection for the dependent bootstrap." *Econometric Reviews* 23, no. 1 (2004): 53-70.

Python Software Foundation. Python Language Reference, version 3.6. Available at http://www.python.org

Shea, Paul, Yanying Sheng, and Michael Varner. "Estimating the Effects of Quantitative Easing on the Real Economy." *Working Paper*, http://paulshea.com/wp-content/uploads/2013/06/qe-bad.pdf, February 2017.

Sheppard, K. (2015). ARCH Toolbox for Python [Data set]. Zenodo. http://doi.org/10.5281/zenodo.15681

Shmueli, Galit. "To explain or to predict?." *Statistical science* 25, no. 3 (2010): 289-310.

Sims, Christopher A. "Macroeconomics and reality." *Econometrica: Journal of the Econometric Society* (1980): 1-48.

StataCorp. 2015. Stata 14 Base Reference Manual. College Station, TX: Stata Press.

Stock, James H., and Mark W. Watson. "Vector autoregressions." *The Journal of Economic Perspectives* 15, no. 4 (2001): 101-115.

Tibshirani, Robert. "Regression shrinkage and selection via the lasso." *Journal of the Royal Statistical Society.* Series B (*Methodological*) (1996): 267-288.

Wen, Yi. 2014. "Evaluating Unconventional Monetary Policies? Why Aren't They More Effective?" *Federal Reserve Bank of St. Louis Working Paper 2012-28.*

Woodford, Michael. 2003. *Interest and Prices*. Princeton, NJ and Oxford: Princeton University Press.

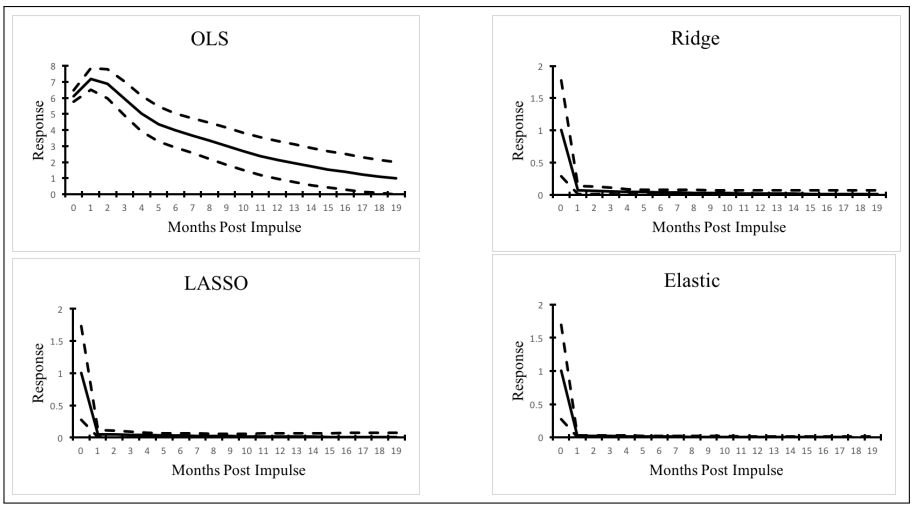
Zou, Hui, and Trevor Hastie. "Regularization and variable selection via the elastic net." *Journal of the Royal Statistical Society*: Series B (*Statistical Methodology*) 67, no. 2 (2005): 301-320.

Chapter 7

Appendix

See here for the source code. This links to a Google Drive folder and contains the all files that are used to create this thesis.

Figure 7.1: IRs for How QE Responds to QE by ML Type



In figure (7.1) the Response axis for Ridge, LASSO, and Elastic is not to the same scale as the OLS graph. I do this to make the first few point estimates visible.

Figure 7.2: IRs for How FFR Responds to QE by ML Type

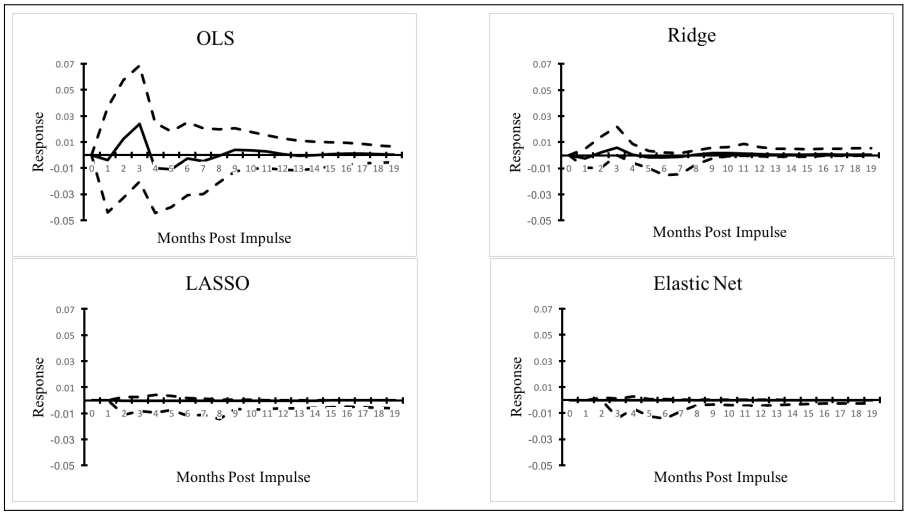


Figure 7.3: IRs for How CPI Responds to QE by ML Type

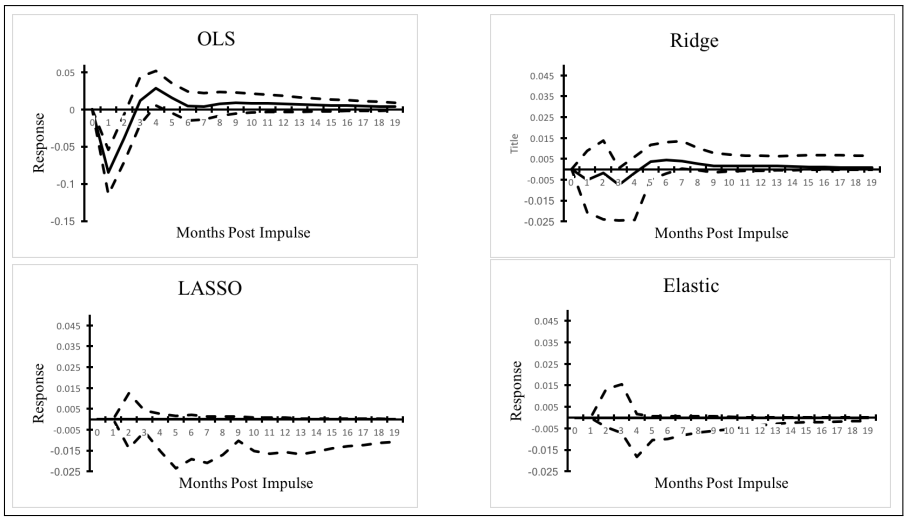


Figure 7.4: Point Estimates of How QE Responds to QE

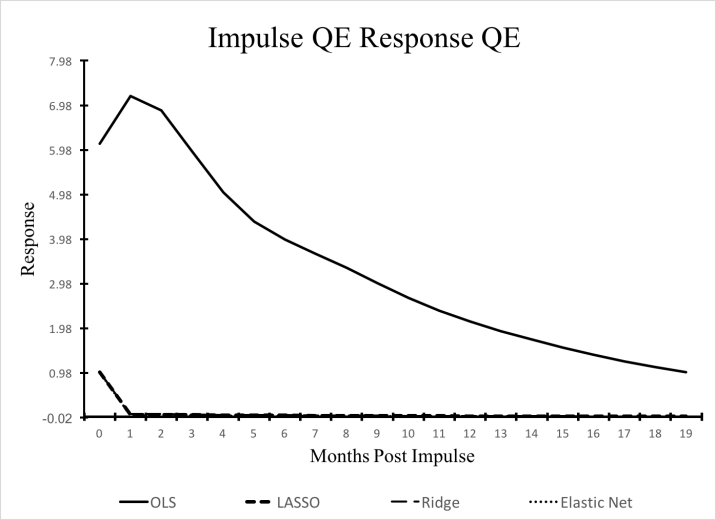


Figure 7.5: Point Estimates of How T-Bill Rate Responds to QE

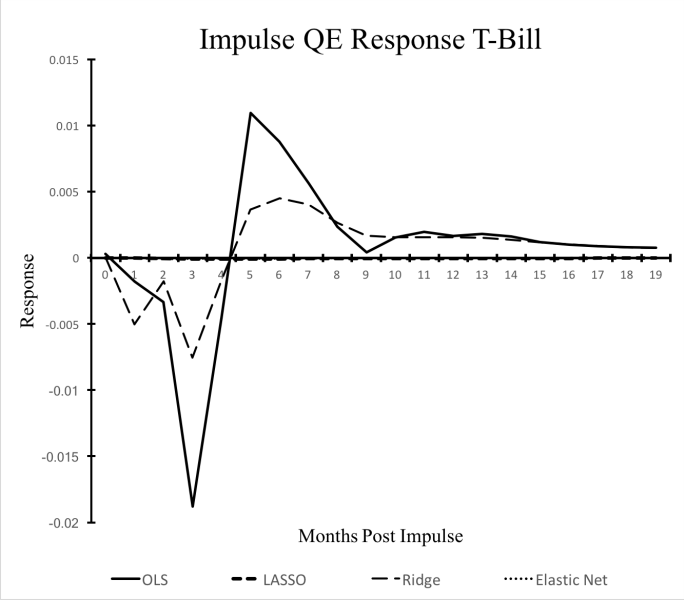


Figure 7.6: LASSO VAR Code

```
import numpy
import pandas
"""
Stage 1: Import and Clean Data
"""
#import data
qe_data = pandas.read_csv('/Users/Mike/Desktop/Bates Files/Senior Year/Thesis/QE_Results/QE_Data/qe_data.csv',
  ↪    low_memory=False)
#remove date variable
date = pandas.DataFrame(qe_data, columns=['date'])
x = qe_data.drop('date', 1)
#create list of exogeneous variable names
exogeneous_var_names = ['money_supply', 'tax_rev', 'federal_debt', 'gov_exp', 'broad_index', 'financial_stress',
  ↪    'industrial_production']
#remove exogeneous variables
endogeneous = x.drop(exogeneous_var_names, 1)
#create a list of the endogeneous variable names
endogeneous_variable_names = list(endogeneous.columns.values)
#create data without lags to use in variance/covariance calculation below
clean_data_with_out_lags = endogeneous.dropna(how='any')
#set the maximum number of lags that the model can consider for variable selection and estimation
#I have arbitrairly chose twice as many lags as endogeneous variables
max_lags = int(len(endogeneous_variable_names)+1)
lags = (range(max_lags))[1:]
#create lagged values of the endogeneous variables
for p in lags:
    for var in endogeneous_variable_names:
        endogeneous["L{0}_{1}".format(p , var)] = endogeneous[var].shift(p)
#add back in the exogeneous variables
#prepared_data = endogeneous.add(exogeneous)
#remove missing values created by creating lags
#this might not be what I want, but come back to it later
clean_data = endogeneous.dropna(how='any')
#Define the "VAR IRF" algo as a function that takes in data and outputs an IRF
def Elastic_Net_Imp_QE_Resp_Unemp(clean_data):
    #Import Packages
    import numpy
    import pandas
    from sklearn.cross_validation import KFold
    from sklearn.grid_search import GridSearchCV
    from sklearn.ensemble import RandomForestClassifier
    from sklearn.linear_model import ElasticNet
    from sklearn import linear_model
    from sklearn.linear_model import LassoCV
    from sklearn.linear_model import ElasticNetCV
    """
    Stage 2: Compute Matricies Using LASSO
    """
    ####Calculate the Y matrix (KxT) or (4x545)
    Y = clean_data[endogeneous_variable_names].transpose()
    Y.as_matrix()
    '''Calculate B using OLS row by row
    '''
    #define the model
    elastic = linear_model.LassoCV(cv=10, n_jobs=-1 , normalize=True)
    #fit a regression using all data except current value
    fed_funds_elastic = elastic.fit(clean_data.drop(endogeneous_variable_names, 1) , clean_data.fed_funds.as_matrix())
    fed_funds_elastic_coeff = fed_funds_elastic.coef_
    unemployment_elastic = elastic.fit(clean_data.drop(endogeneous_variable_names, 1) ,
      ↪    clean_data.unemployment.as_matrix())
    unemployment_elastic_coeff = unemployment_elastic.coef_
    treasury_bill_elastic = elastic.fit(clean_data.drop(endogeneous_variable_names, 1) ,
      ↪    clean_data.treasury_bill.as_matrix())
    treasury_bill_elastic_coeff = treasury_bill_elastic.coef_
    qe_elastic = elastic.fit(clean_data.drop(endogeneous_variable_names, 1),
    clean_data.qe.as_matrix())
    qe_elastic_coeff = qe_elastic.coef_
    B = treasury_bill_elastic_coeff
    #append rows of coefficients
    B = numpy.vstack([qe_elastic_coeff,B])
    B = numpy.vstack([unemployment_elastic_coeff,B])
```

Figure 7.7: LASSO VAR Code Continued

```
    B = numpy.vstack([fed_funds_elastic_coeff,B])
    B = pandas.DataFrame(B)
    #####PUT INTO SPACE STATE NOTATION#####
    #create a new coefficient matrix which is (npxnp) or (16x16)
    I_4 = numpy.identity(12)
    zero_columns = numpy.zeros((12,4))
    add_on = numpy.column_stack((I_4,zero_columns))
    #append add_on to to B to get B_Space_State
    B_Space_State = pandas.DataFrame(numpy.append(B,add_on, axis=0))
    #calculate the variance/covariance matrix using numpy
    Sigma = pandas.DataFrame(numpy.cov(clean_data_with_out_lags,rowvar=False))
    #orthogonalize Sigma by imposing off diagonals are 0
    diag_sigma = pandas.DataFrame(numpy.diag(numpy.diag(Sigma)))
    #calculate cholesky decomposistion of sigma
    chol_of_sigma = pandas.DataFrame(numpy.linalg.cholesky(diag_sigma))
    #define the inverse of the cholesky decomposistion of sigma to use below
    inverse_chol = numpy.linalg.inv(chol_of_sigma)
    ####Calculate Space-State version of the Q matrix
    zero_columns = numpy.zeros((12,4))
    zeros_16x12 = numpy.zeros((16,12))
    #append add_on to to Sigma to get Q_Space_State
    Q_Space_State = pandas.DataFrame(numpy.append(inverse_chol,zero_columns, axis=0))
    Q_Space_State = pandas.DataFrame(numpy.column_stack((Q_Space_State,zeros_16x12)))
    #Calculate IRs
    #create impulse column vector (TX1)
    e_feddunds = pandas.DataFrame(numpy.vstack((1,numpy.zeros(((15),1)))))
    #create impulse column vector (TX1) for "qe"
    sd_of_qe = clean_data["qe"].std()
    e_qe = pandas.DataFrame(numpy.vstack((numpy.vstack((numpy.zeros((2, 1)),sd_of_qe)),numpy.zeros((13,1)))))
    e_0 = e_qe
    #run shock e_0 through system
    Q_e = pandas.DataFrame(numpy.matmul(Q_Space_State,e_0))
    #create the point estimates (QBQE_0)
    for p in range(1,21):
        globals()['point_estimate_{}'.format(p)] = pandas.DataFrame(numpy.matmul(Q_Space_State,
        ↪    numpy.matmul(numpy.linalg.matrix_power(B_Space_State, p),Q_e)))
    #append point estimates into a matrix
    impulse_qe_var = pandas.DataFrame(numpy.column_stack((Q_e,point_estimate_1)))
    #for p in range(2,20):
    #    impulse_qe_var = numpy.column_stack((impulse_qe_var,"point_estimate_"+str(p)))
    impulse_qe_var = pandas.DataFrame(numpy.column_stack((impulse_qe_var,point_estimate_2)))
    impulse_qe_var = pandas.DataFrame(numpy.column_stack((impulse_qe_var,point_estimate_3)))
    impulse_qe_var = pandas.DataFrame(numpy.column_stack((impulse_qe_var,point_estimate_4)))
    impulse_qe_var = pandas.DataFrame(numpy.column_stack((impulse_qe_var,point_estimate_5)))
    impulse_qe_var = pandas.DataFrame(numpy.column_stack((impulse_qe_var,point_estimate_6)))
    impulse_qe_var = pandas.DataFrame(numpy.column_stack((impulse_qe_var,point_estimate_7)))
    impulse_qe_var = pandas.DataFrame(numpy.column_stack((impulse_qe_var,point_estimate_8)))
    impulse_qe_var = pandas.DataFrame(numpy.column_stack((impulse_qe_var,point_estimate_9)))
    impulse_qe_var = pandas.DataFrame(numpy.column_stack((impulse_qe_var,point_estimate_10)))
    impulse_qe_var = pandas.DataFrame(numpy.column_stack((impulse_qe_var,point_estimate_11)))
    impulse_qe_var = pandas.DataFrame(numpy.column_stack((impulse_qe_var,point_estimate_12)))
    impulse_qe_var = pandas.DataFrame(numpy.column_stack((impulse_qe_var,point_estimate_13)))
    impulse_qe_var = pandas.DataFrame(numpy.column_stack((impulse_qe_var,point_estimate_14)))
    impulse_qe_var = pandas.DataFrame(numpy.column_stack((impulse_qe_var,point_estimate_15)))
    impulse_qe_var = pandas.DataFrame(numpy.column_stack((impulse_qe_var,point_estimate_16)))
    impulse_qe_var = pandas.DataFrame(numpy.column_stack((impulse_qe_var,point_estimate_17)))
    impulse_qe_var = pandas.DataFrame(numpy.column_stack((impulse_qe_var,point_estimate_18)))
    impulse_qe_var = pandas.DataFrame(numpy.column_stack((impulse_qe_var,point_estimate_19)))
    #pull out the 3rd row to get response of unemployment
    imp_qe_response_unemp = pandas.DataFrame(impulse_qe_var.loc[1:1].transpose())
    #remove the 0th impact estimate, becuase it has no variation
    imp_qe_response_unemp = imp_qe_response_unemp.drop(imp_qe_response_unemp.index[[0]])
    #convert back to numpy array
    imp_qe_response_unemp = imp_qe_response_unemp.as_matrix()
    #put into Pandas Series
        #imp_qe_response_unemp = pandas.Series(imp_qe_response_unemp[:,0])
    imp_qe_response_unemp = imp_qe_response_unemp[:,0]
    indexing_numbers = [1 , 2 , 3 , 4 , 5 , 6 , 7 , 8 , 9 , 10 , 11 , 12 , 13 , 14 , 15 , 16 , 17 , 18 , 19 ]
    final_output = pandas.Series(imp_qe_response_unemp, index=indexing_numbers)
    return final_output
#calculate point estimates using the whole data set
point_estimates = Elastic_Net_Imp_QE_Resp_Unemp(clean_data)
point_estimates = point_estimates.to_frame()
#rename the column as Point_Estimates
```

Figure 7.8: LASSO VAR Code Continued

```
point_estimates = point_estimates.rename(columns = {0 : 'Point_Estimates'})
#Define the Stationary Bootstrap using the Cleaned Data
from arch.bootstrap import StationaryBootstrap
bs_block = StationaryBootstrap(12, clean_data)
#compute confidence intervals using bias-correction at 95%
confidence_interval_1 = bs_block.conf_int(Elastic_Net_Imp_QE_Resp_Unemp, 1000, method='bc', size=.95, tail='two',
 ↪    sampling='nonparametric')
confidence_interval_2 = pandas.DataFrame(confidence_interval_1, index=['Lower','Upper'])
confidence_interval_3 = confidence_interval_2.transpose()
#combine the point estimates and the confidence intervals
#REMEMBER THAT 0 PERIOD IS EXACTLY 0 WITH NO VARIANCE!! WE HAD TO REMOVE IT TO BOOTSTRAP
#ADD THE 0th ESTIMATE BACK IN
final_ir = pandas.concat([point_estimates.reset_index(drop=True), confidence_interval_3], axis=1)
#display final_ir
print(final_ir)
#copy final_ir to clipboard, to paste into excel file
final_ir.to_clipboard(excel=True)
```

Figure 7.9: Simulation Loop

```
import numpy
import pandas
#tell Python where to look for custom functions
import sys
sys.path.append('C:/Users/mvarner/Desktop/Bates_Files_(2-8-2017)/Senior
  ↪    Year/Thesis/Simulations/Simulation_Scripts/ML_Scripts/')
#tell Python where to find the Arch package
sys.path.append('C:/ProgramData/Anaconda3/Lib/site-packages/')
#import the OLS IR script
from OLS_Imp_FFR import OLS_Imp_FFR
#import the ML IR scripts
from Elastic_Resp_FFR import Elastic_Resp_FFR
from Elastic_Resp_Raw import Elastic_Resp_Raw
from Elastic_Resp_CPI import Elastic_Resp_CPI
from Elastic_Resp_CFNAI import Elastic_Resp_CFNAI
from LASSO_Resp_FFR import LASSO_Resp_FFR
from LASSO_Resp_Raw import LASSO_Resp_Raw
from LASSO_Resp_CPI import LASSO_Resp_CPI
from LASSO_Resp_CFNAI import LASSO_Resp_CFNAI
from Ridge_Resp_FFR import Ridge_Resp_FFR
from Ridge_Resp_Raw import Ridge_Resp_Raw
from Ridge_Resp_CPI import Ridge_Resp_CPI
from Ridge_Resp_CFNAI import Ridge_Resp_CFNAI
#Import the confidence interval function
from Confidence_Interval import CI
"""
Stage 1: Import and Clean Data
"""
#create column names
endogeneous_variable_names = ['cfnai', 'cpi', 'raw', 'ffr']
#import data
all_data = pandas.read_csv('C:/Users/mvarner/Desktop/Bates_Files_(2-8-2017)/Senior
  ↪    Year/Thesis/Simulations/Simulation_Data/kk_code/All_Simulation_Data_1000_Runs', names =
  ↪    endogeneous_variable_names, low_memory=False)
#this function will run until the ml_model actually works
def Find_Error(ml_model):
    while True:
        try:
            ml_model = CI(ml_model, clean_data)
            print('it worked')
            return ml_model
        except ValueError:
            print('errored_out')
'''Stage 2: Loop Through all of the data
'''
for i in range(0,100):
    #subset the larger data set
    first_row = int(i*455)
    last_row = int((i+1)*455)
    endogeneous = all_data.loc[first_row:last_row]
    #create lags and clean the data,
    clean_data = endogeneous
    #difference CPI and FFR to make stationary: they each have 1 unit root
    endogeneous.cpi.diff()
    endogeneous.ffr.diff()
    endogeneous.dropna
    #choose max lags to be 12. this is because the DGP is a VAR(12)
    max_lags = int(12)
    lags = (range(max_lags+1))[1:]
    #create lagged values of the endogeneous variables
    for p in lags:
        for var in endogeneous_variable_names:
            endogeneous["L{0}_{1}".format(p , var)] = endogeneous[var].shift(p)
        #remove missing values created by creating lags
    clean_data = endogeneous.dropna(how='any')
    #create data without lags to use in variance/covariance calculation below
    clean_data_with_out_lags = clean_data[endogeneous_variable_names]
    '''Elastic IR Calculations
    '''
    #create the confidence intervals
    elastic_ffr = Find_Error(Elastic_Resp_FFR)
    elastic_raw = Find_Error(Elastic_Resp_Raw)
    elastic_cpi = Find_Error(Elastic_Resp_CPI)
    elastic_cfnai = Find_Error(Elastic_Resp_CFNAI)
    #add back in the 0th estimate which is zero, so confidence intervals are exactly zero
```

Figure 7.10: Simulation Loop Continued

```
    zeros = numpy.zeros((2,1))
    elastic_raw = numpy.concatenate((zeros, elastic_raw), axis=1)
    elastic_cpi = numpy.concatenate((zeros, elastic_cpi), axis=1)
    elastic_cfnai = numpy.concatenate((zeros, elastic_cfnai), axis=1)
    #create results matrix for elastic net (2x24x4)
    elastic_results = elastic_ffr[:,:, numpy.newaxis]
    elastic_results = numpy.dstack((elastic_results,elastic_raw))
    elastic_results = numpy.dstack((elastic_results,elastic_cpi))
    elastic_results = numpy.dstack((elastic_results,elastic_cfnai))
    '''LASSO IR Calculations
    '''
    lasso_ffr = Find_Error(LASSO_Resp_FFR)
    lasso_raw = Find_Error(LASSO_Resp_Raw)
    lasso_cpi = Find_Error(LASSO_Resp_CPI)
    lasso_cfnai = Find_Error(LASSO_Resp_CFNAI)
    zeros = numpy.zeros((2,1))
    lasso_raw = numpy.concatenate((zeros, lasso_raw), axis=1)
    lasso_cpi = numpy.concatenate((zeros, lasso_cpi), axis=1)
    lasso_cfnai = numpy.concatenate((zeros, lasso_cfnai), axis=1)
    lasso_results = lasso_ffr[:,:, numpy.newaxis]
    lasso_results = numpy.dstack((lasso_results,lasso_raw))
    lasso_results = numpy.dstack((lasso_results,lasso_cpi))
    lasso_results = numpy.dstack((lasso_results,lasso_cfnai))
    '''Ridge IR Calculations
    '''
    ridge_ffr = Find_Error(Ridge_Resp_FFR)
    ridge_raw = Find_Error(Ridge_Resp_Raw)
    ridge_cpi = Find_Error(Ridge_Resp_CPI)
    ridge_cfnai = Find_Error(Ridge_Resp_CFNAI)
    zeros = numpy.zeros((2,1))
    ridge_raw = numpy.concatenate((zeros, ridge_raw), axis=1)
    ridge_cpi = numpy.concatenate((zeros, ridge_cpi), axis=1)
    ridge_cfnai = numpy.concatenate((zeros, ridge_cfnai), axis=1)
    ridge_results = ridge_ffr[:,:, numpy.newaxis]
    ridge_results = numpy.dstack((ridge_results,ridge_raw))
    ridge_results = numpy.dstack((ridge_results,ridge_cpi))
    ridge_results = numpy.dstack((ridge_results,ridge_cfnai))
    '''OLS IR Calculations
    '''
    ols_results = OLS_Imp_FFR(clean_data)
    '''Put results into Matricies
    '''
    #create all results matrix using 1st iteration of the loop
    if i==0:
        all_elastic_results = elastic_results[:,:,:, numpy.newaxis]
        all_lasso_results = lasso_results[:,:,:, numpy.newaxis]
        all_ridge_results = ridge_results[:,:,:, numpy.newaxis]
        all_ols_results = ols_results[:,:,:, numpy.newaxis]
    #add in the new results after 1st interation of the loop
    if i!=0:
        all_elastic_results = numpy.concatenate((all_elastic_results,elastic_results[:,:,:,numpy.newaxis]),axis=3)
        all_lasso_results = numpy.concatenate((all_lasso_results,lasso_results[:,:,:,numpy.newaxis]),axis=3)
        all_ridge_results = numpy.concatenate((all_ridge_results,ridge_results[:,:,:,numpy.newaxis]),axis=3)
        all_ols_results = numpy.concatenate((all_ols_results,ols_results[:,:,:,numpy.newaxis]),axis=3)
#change the working directory
os.chdir('C:/Users/mvarner/Desktop/Bates_Files_(2-8-2017)/Senior
  ↪    Year/Thesis/Simulations/Simulation_Scripts/Output_Matricies/')
#save the output files
numpy.save('elastic_results.npy', all_elastic_results)
numpy.save('lasso_results.npy', all_lasso_results)
numpy.save('ridge_results.npy', all_ridge_results)
numpy.save('ols_results.npy', all_ols_results)
```