Autoregressive Model Estimation via LASSO

LJ Valencia*

Introduction

The use of machine learning methods has gained prominence in the past two decades. Tibshirani (1996) proposed an estimation method that provides useful features of simultaneous parameter estimation and variable selection. This method, called the Least Absolute Shrinkage Selection Operator (LASSO) has grown in popularity since its introduction by Tibshirani for its ability to yield interpretable models. LASSO can easily produce models out of very large datasets and avoid overfitting.

In this paper, I demonstrate how to develop a lagged dataset for estimation, how to implement LASSO, and how to produce a forecast from an estimated LASSO model.

LASSO

LASSO is defined as an extension of a linear regression model:

$$\min RSS + \lambda \sum_{k=1}^{K} |\beta_k|$$

where RSS is the Residual Sum of Squared Errors, K is the number of variables, $P = \sum_{k=1}^{K} |\beta_k|$ is the L1 penalty term and λ is the regularization parameter.

LASSO penalizes the absolute value of all regression coefficients in the model. The regularization parameter determines the level of penalty imposed on the coefficients. This allows some parameter estimates to be possibly equivalent to zero. This feature is very useful for variable selection. A larger penalty (larger lambda value) means more estimates are shrunk towards zero and the result is a smaller model with fewer variables. A smaller penalty (smaller lambda value) means a larger model that is closer to an unrestricted OLS regression. In this paper, LASSO is used to fit an autoregressive model which is used to forecast the United States' Industrial Production Index. Furthermore, LASSO is also utilized to determine lagged predictors relevant for forecasting.

Dataset

The Industrial Production Index is a measure of real output for all relevant establishments in the United States regardless of ownership. The index does not account for establishments located in the U.S. Territories. The data is collected from the Federal Reserve Economic Data (FRED). The time-series is seasonally-adjusted and is in monthly frequency. The series begins in January 1970 and ends in July 2019. Table 1 shows the summary statistics.

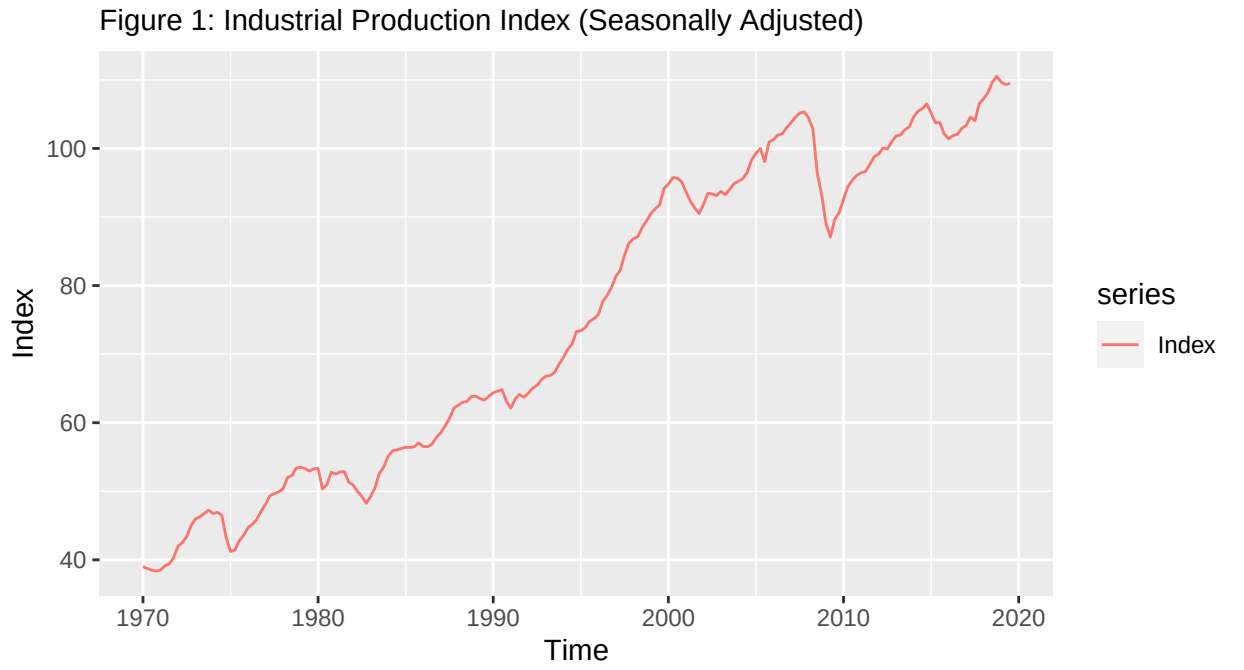
*University of Alberta, Department of Economics

Table 1: Summary Statistics

	indpro
	Min. : 38.35
	1st Qu.: 53.10
	Median : 73.29
	Mean : 75.01
	3rd Qu.: 96.48
	Max. :110.55

The data is non-stationary. The first plot exhibits certain characteristics of non-stationarity in the index such as a positive time trend, cyclicality, a significant volatility in growth during 1970-1980, and a structural break caused by the Great Recession. In addition, the data is converted to quarterly frequency. This is done by taking the value of the index at every third month as a representation of each quarter in a given year. For example, the value of the index on March 2017 represents the first quarter of 2017. To address non-stationary, the series is log differenced to convert the data to a growth rate as shown in the second plot. This removes the time trend. However, the growth rate appears to show heteroskedastic features caused by volatile growth during 1970-1980 and therefore is not fully stationary. This is can compromise the predictive power of our models. To avoid this problem, the start period is modified to the first quarter of 1980 for both the ARIMA and LASSO modes. The time-series in this demonstration are plotted using the 'autoplot' function.

```
#Plot the Data
index <- ts(data$indpro, start=c(1970,1), frequency=4) # turn data into time-series object
autoplot(window(index, start=1970, frequency=4), series="Index") +
  ggtitle("Figure 1: Industrial Production Index (Seasonally Adjusted)")+
  ylab("Index") +
  xlab("Time") +
  theme(plot.title = element_text(size = 11))
```



```r
#Transform the Data:
indpro <- 100*(diff(log(data$indpro)))
#Store data in a time-series object
x0 <- ts(cbind(indpro), start=c(1970,1), frequency=4)
#plot original dataset
autoplot(x0, series = 'Growth Rate') + ylab("Log-Difference (%)") +
  xlab("Time") +
  ggtitle("Figure 2: Industrial Production Index Growth Rate") +
  theme(plot.title = element_text(size = 11))
```

Figure 2: Industrial Production Index Growth Rate

Lagged Time-Series

The next step is to create a lagged dataset. This is done by first transforming the data using log-difference. Then, the transformed data is stored in a time-series (ts) object. The code below uses lapply to create a lagged time-series. Then sapply function is used to assign each variable a name with respect to its lag.[1]

```r
#Creates Lagged Variables
data <- read.table("dataset.csv",
                   sep=",",
                   header=TRUE)
#Transform the Data:
indpro <- 100*(diff(log(data$indpro)))
#Store data in a time-series object
x0 <- ts(cbind(indpro),
         start=c(1970,1),
         frequency=4)
#Set values for lags
```

[1]I'd like to thank Dr. Max Sties for his useful resource on how to create a lagged time series, Source: http://econgrad.blogspot.com/

```
n.lag=8 # eight quarterly lags
n.var = dim(x0)[2] # take the number of variables
seq.l = seq(0,-n.lag) # generate a sequence from zero to the allowed number of lags.
# Creating Lagged Data Set
xl <- ts(do.call(cbind,lapply(1:ncol(x0),
                              function(z) lag(as.zoo(x0[,z]),seq.l))),
         start=start(x0),
         end=end(x0),
         frequency=frequency(x0))
# Name variables in new data set by adding the respective lag
colnames(xl) <- as.vector(sapply(colnames(x0),
                                 function(z) paste(z,"_l",-seq.l, sep = "")))
```

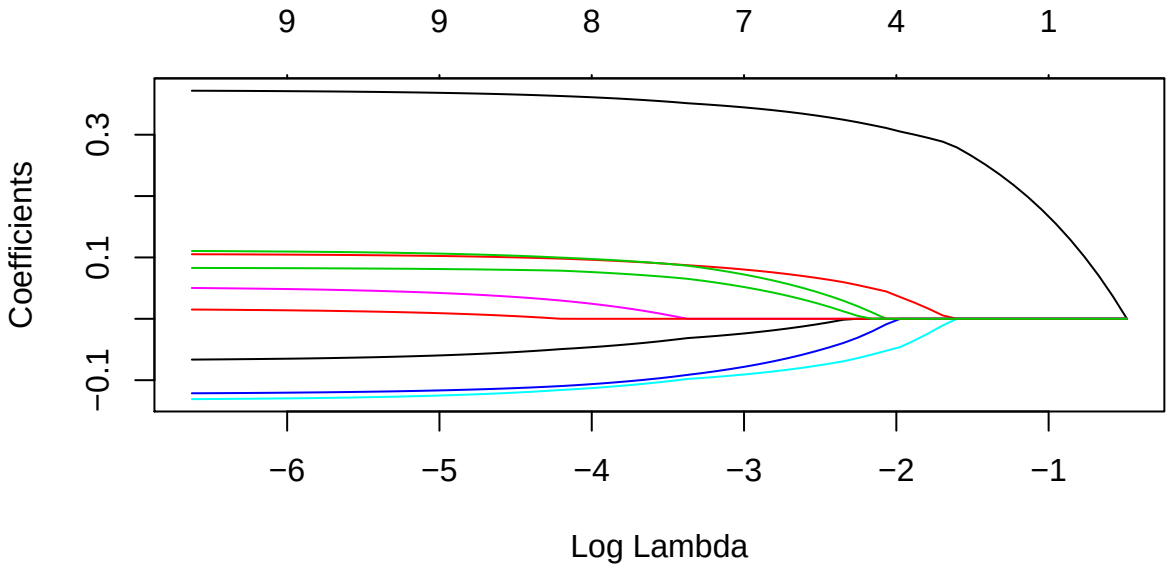
LASSO Estimation

In this section, I implement LASSO in R with the lagged dataset. The code below prepares the data inputs which is the first quarter of 1980 to the last observation of the lagged dataset.

In figure 1, The LASSO variable trace plot outlines the values of each coefficient in the model. The bottom x-axis indicate different log lambda values used for penalizing coefficients. The upper x-axis shows the number of predictors at different log lambda values. Furthermore, the trace plot highlights that as you increase the log lambda value, there are fewer variables and therefore smaller models.

```
#Prepare the data inputs for LASSO
window.start = c(1980,1)
window.end = end(xl)
xt <- window(xl, start=window.start, end=window.end)
# dim(xt)
n.obs <- nrow(xt)
#Set up the variables for LASSO
h=1
yt <- xt[,"indpro_l0"]
y <- yt[(1+h):n.obs]
X <- xt[1:(n.obs-h),]
x.new <- matrix(xt[n.obs,], nrow = 1)
#LASSO Regression
lasso <- glmnet(X,y)
#LASSO Coefficients
plot(lasso, xvar='lambda')
mtext("Figure 3: LASSO trace plot", side=3, line=2.5, cex=0.9)
```

Figure 3: LASSO trace plot



The next section of code below shows what is considered the most relevant set of lagged predictors for the Industrial Production Index which are: first, second, third, fourth, sixth, and eight lags. These values are the non-zero beta coefficients. The next value shows the intercept, followed by the lambda value.

```
#Relevant Coefficients
bb <- lasso$beta[,24]
bb[bb!=0] # nonzero beta coefficients
```

```
##   indpro_l0   indpro_l1   indpro_l2   indpro_l3   indpro_l4   indpro_l6
##  0.33460222  0.07012642  0.05108178 -0.05902013 -0.08008776 -0.01284165
##   indpro_l8
##  0.03222461
```

```
lasso$a0[24] # intercept
```

```
##      s23
## 0.339845
```

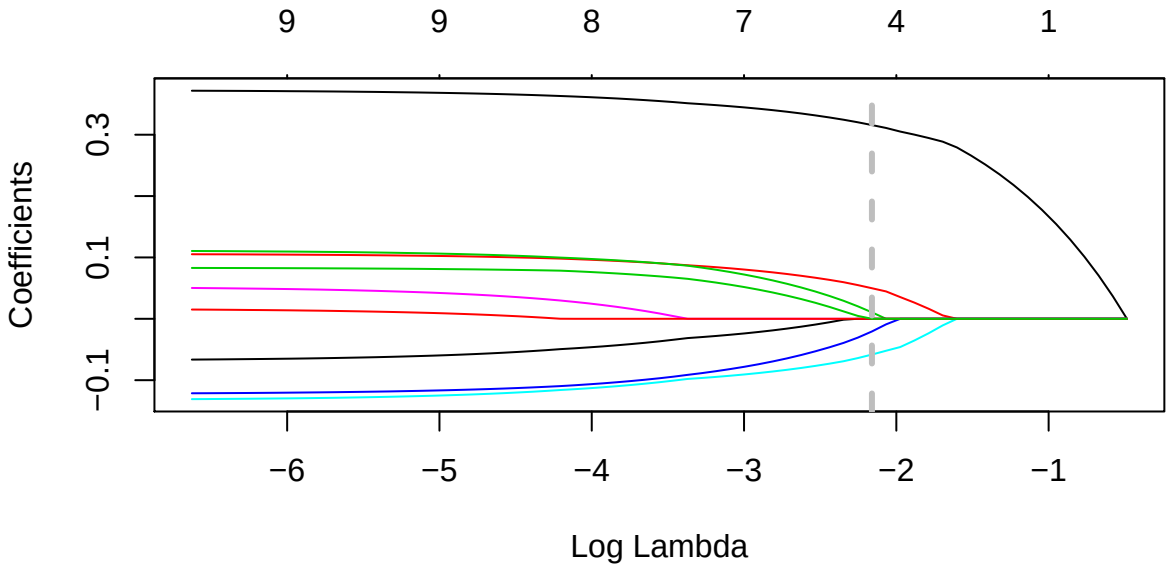
```
lasso$lambda[24] # lambda value
```

```
## [1] 0.07240868
```

Forecast

In this section, the variables are set for forecasting. These variables are stored in a new matrix. The maximum permitted number of variables for selection is set to five. The code below shows a LASSO trace plot. The grey line indicates which lambda value yields the amount of penalization the maximum allowed number of restricted variables is five.

Figure 3: LASSO trace plot

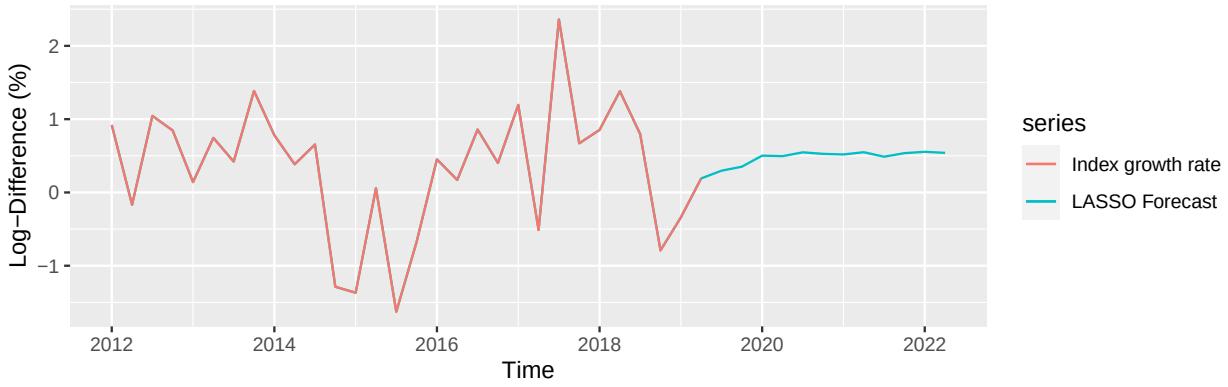


Next, the sections of code below shows how a forecast is produced. A for loop is used to create an h-step ahead forecast. The forecast horizon is set to twelve quarters. The 'rep' function is used to produce a vector where the forecast results are stored. The forecasts are then converted to a time-series object which is then plotted.

```
#Loop for creating a h-step ahead forecast
n.fc=12  #Forecast Horizon
fcast.lasso <- rep(NA,n.fc)
for (h in 1:n.fc){
  y <- yt[(1+h):n.obs]
  X <- xt[1:(n.obs-h),]
  x.new <- matrix(xt[n.obs,],nrow=1)
  lasso <- glmnet(X,y)
  #Stores forecast values
  fcast.lasso[h] <- predict(lasso,x.new,s=lambda)
}
```

```
#Plot Forecast
yt.fc.lasso <- ts(c(yt,fcast.lasso), start=window.start, freq=4)
autoplot(window(yt.fc.lasso,start=c(2012,1)), series='LASSO Forecast') +
  autolayer(window(yt, start=c(2012,1)), series='Index growth rate') +
  ylab("Log-Difference (%)") +
  xlab("Time") +
  ggtitle("Figure 4: Industrial Production Index Growth Rate") +
  theme(plot.title = element_text(size = 11))
```

Figure 4: Industrial Production Index Growth Rate



Multiple Forecasts

Although the previous section showed how to produce a forecast, there are multiple lambda values. This means that I have yet to account for other alternative models. The code below shows how to create multiple forecasts under different lambda values. The manner in which the forecasts are produced is similar to the previous section.The first component shows the maximum number of lambda values from the LASSO implementation. This means that there are 67 possible models and 67 different forecasts.

```
h=1
yt <- xt[,"indpro_l0"]
y <- yt[(1+h):n.obs]
X <- xt[1:(n.obs-h),]
lasso <- glmnet(X,y)
lambda.seq <- lasso$lambda # store lambdas as another variable
length(lambda.seq) # shows the maximum number of lambda values
```

```
## [1] 67
```

This section of code shows how to produce the forecasts using a for loop. The forecasts are stored in a matrix.

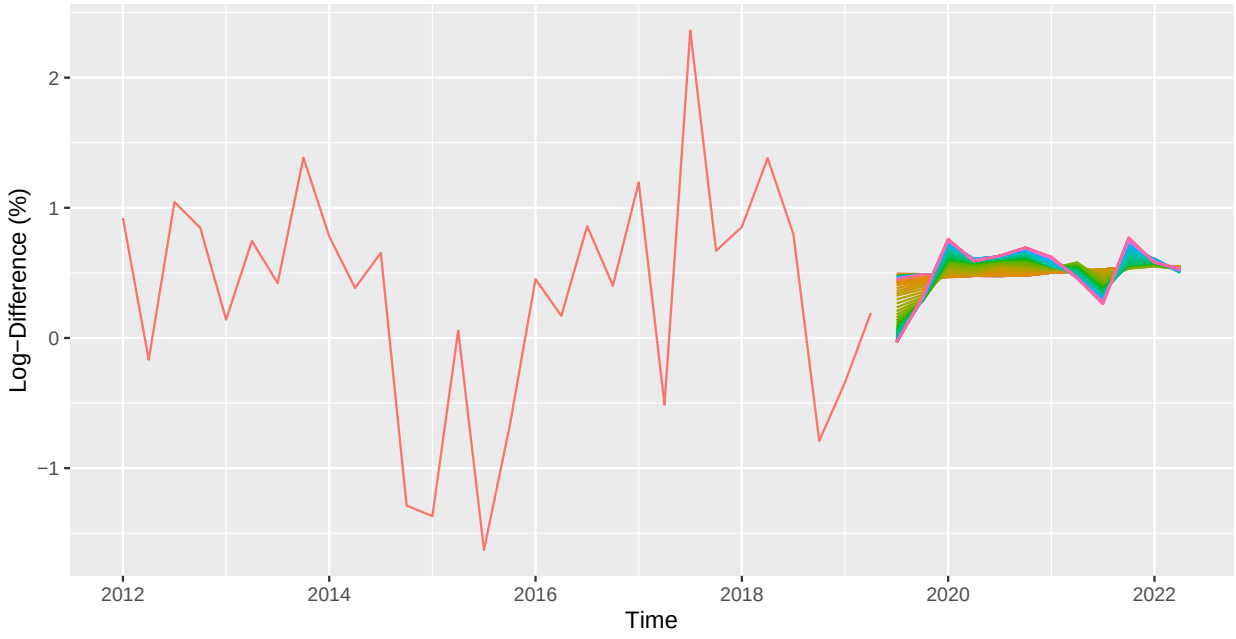
```
forecast.horizon = 12
n.fc <- forecast.horizon
lasso.multi.fcast <- matrix(NA, nrow=n.fc, ncol=length(lambda.seq))

for (h in 1:n.fc){
  y <- yt[(1+h):n.obs]
  X <- xt[1:(n.obs-h),]
  x.new <- matrix(xt[n.obs,],nrow=1)
  lasso <- glmnet(X,y)
  lasso.multi.fcast[h,] <- predict(lasso, x.new, type='response', s=lambda.seq)
}
```

The multiple alternative forecasts are plotted using the ‘autoplot’ function.

```
lasso.multi.fcast <- ts(lasso.multi.fcast,
                        start=c(end(yt)[1],end(yt)[2]+1),
                        freq=4) # start period is one period after the end of the variable
autoplot(window(yt, start=c(2012,1)), series='Index growth rate') +
  autolayer(lasso.multi.fcast) + theme(legend.position="none") +
  ylab("Log-Difference (%)") +
  xlab("Time") +
  ggtitle("Figure 5: Multiple Forecasts") +
  theme(plot.title = element_text(size = 11))
```

Figure 5: Multiple Forecasts



Time-Series Validation

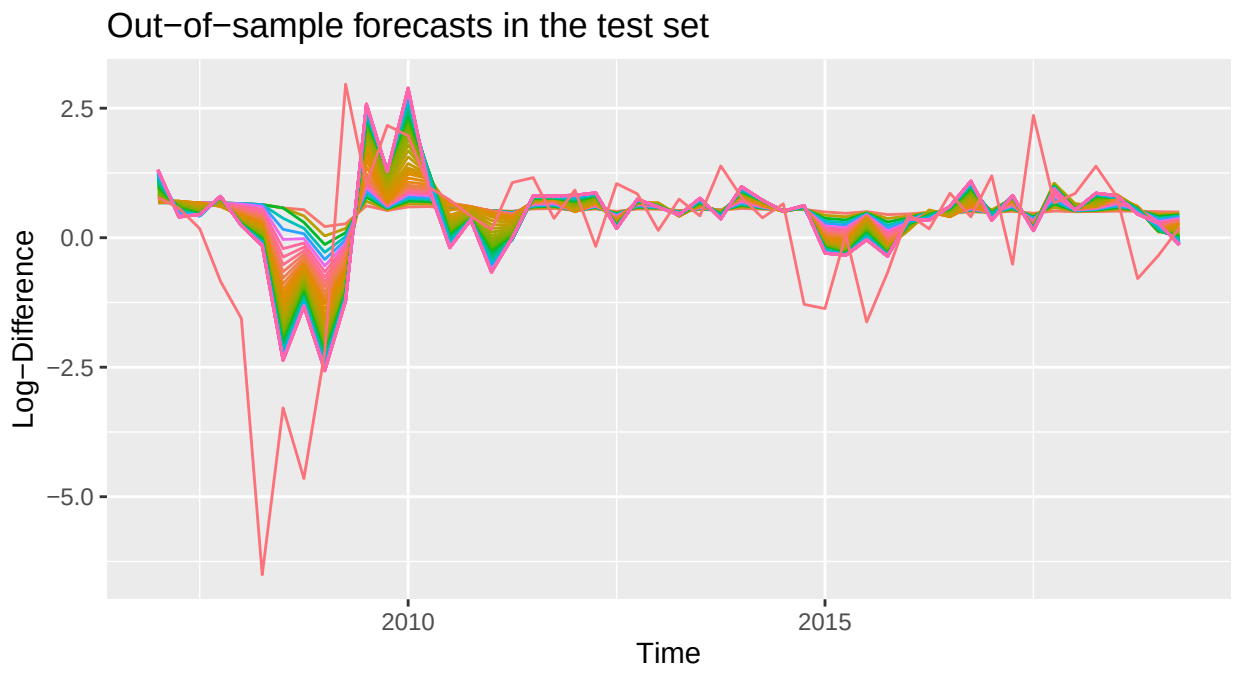
Is it possible to improve the forecasting power of the LASSO model? So far, the results from the prior implementation is based on an arbitrarily-set lambda value. Since it is optimal to make the forecasts as precise as possible, it is important to account for the other lambda values. In this portion, I utilize time-series cross validation to find the model most useful for forecasting. To allow for comparison between different models, I use the Root Mean-Squared Error (RMSE). The lambda value and the resulting model that yields the lowest RMSE is the best possible forecasting model.

The section of code below shows

To conduct the cross-validation exercise, a training set and a test set is necessary. The data for the training set begins from 1980-Q1 to 2006-Q4. The test set starts from 2007-Q1 until 2019-Q2. The LASSO regression is ran in the training set. Afterwards, different models are implemented on the test set.

```
#setup variables for plotting
yt.oos <- ts(yt.oos, start = c(2007,1), end=c(2019,2), freq=4)
pred.oos.h1 <- ts(lasso.cv, start = c(2007,1), end=c(2019,2), freq=4)
#Plot
autoplot(pred.oos.h1) + theme(legend.position="none")+
```

```
  autolayer(yt.oos) + ylab("Log-Difference") + xlab("Time") +
  ggtitle("Out-of-sample forecasts in the test set")
```

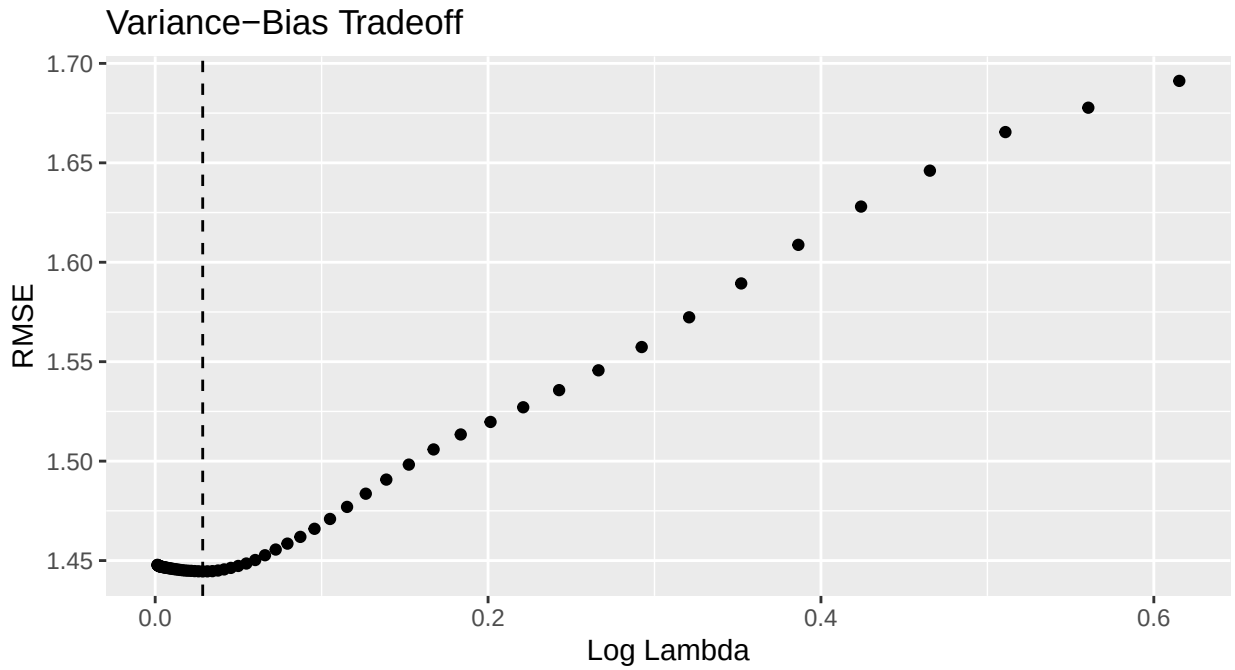



```
# compute rmse
yt.oos <- window(yt.oos,
                 start=c(2007,1),
                 end=c(2019,2),
                 frequency=4)
rmse.lasso <- rep(NA,length(lambda.seq))
for(m in 1:length(lambda.seq)){rmse.lasso[m] <- sqrt(mean((yt.oos-lasso.cv[,m])^2))}
rmse.min <- min(rmse.lasso)
lambda.cv <- data.frame(cbind(rmse.lasso,lambda.seq))
# Selects column value of lambda that gives min rmse
lambda.cv.min <- which(grepl(rmse.min,
                             lambda.cv$rmse))
lambda.opt <- lambda.seq[lambda.cv.min]
```

The variance vs. bias tradeoff in Figure 17 does not behave exactly like a quadratic function. However, this is an indication that higher penalties for LASSO produces less accurate models. This makes sense given that larger penalties yield smaller models that converge or behave similarly to the mean growth rate of industrial production. Nevertheless, the chart shows that there is a lambda value that yields the lowest RMSE.

```
lambda.cv <- data.frame(cbind(rmse.lasso, lambda.seq))
ggplot(lambda.cv,
       aes(x=lambda.seq, y=rmse)) +
  geom_point(aes(y=rmse.lasso)) +
  geom_vline(xintercept = lambda.opt,
             linetype='dashed',
             color= 'black') +
```

```
  ylab("RMSE") +
  xlab("Log Lambda") +
  ggtitle("Variance-Bias Tradeoff")
```



Tibshirani, R. 1996. "Regression Shrinkage and Selection via the Lasso." *Journal of the Royal Statistical Society (Series B)* 58: 267–88.