1 Vector Autoregression in R

We will now apply the numerous concepts learned in VAR in an actual example. In particular, we will be using a framework developed by Sims (1992) using Philippine data. In this model, we will be using four variables. These are the following:

- Overnight Reverse Repurchase Rate (RRP) which is set by the Bangko Sentral ng Pilipinas. This is, by all accounts, the main policy rate that the Philippine central bank controls.
- M1 Money Supply which can be obtained from the BSP's website
- CPI Inflation Rate which is reported monthly by the Philippine Statistics Authority and measures the relative increase in prices based on a Laspeyres price index.
- Industrial Production which measures the value of all goods in the industrial sector.

We will first estimate a standard VAR which reflects a key economic response. It is believed in theory and by Sims that shocks to the nominal interest rate represent monetary policy shocks. A shock to the policy variable affects all other variables contemporaneously. The variable is affected by all the others within the period, and is order last. Lastly, the central bank only observes non-policy variables with a lag.

1.1 Preliminaries

We start again by installing the required packages and loading them using the library() command. For this part, we need to install the "vars" package which will have a host of commands necessary for us to run the VAR and SVAR and the diagnostic tests and applications to follow. We then see the plots of each variable and judge some initial conditions such as non-stationarity.

1.1.1 Initial Steps

As said, we need to install the "vars" package. Use the install.packages("vars") command to do this. After which, we will load this package together with our standard suite of packages and libraries for us to continue on the estimation.

```
install.packages("vars")
library(urca)
library(vars)
library(mFilter)
library(tseries)
library(TSstudio)
library(forecast)
library(tidyverse)
```

After installation and loading, it is time to load our dataset. We will use the file *Sample_SVAR.csv* which contains the data on all the variables from January 2003 until February 2020. The data is monthly and is publicly available, obtained from the BSP and PSA. We use the read_csv() command to read the dataset and the file.choose() command to open up a dialogue box for us to select our data. In this case, we place the dataset under an object named "mp". Name it whatever you want, if you'd like. We then use the head() command to see the first few rows of the dataset to inspect if it loaded correctly.

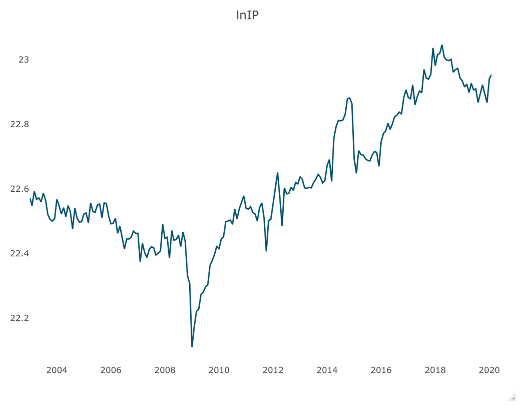
```
mp <− read_csv(file.choose())
head(mp)
```

Next, we need to declare each variable in the dataset as a time series using the ts() command. We use the $ symbol to call a variable from the dataset. All variables start in the year 2003 with the same day and month of January 1. We set the frequency to 12 since we are dealing with monthly data.

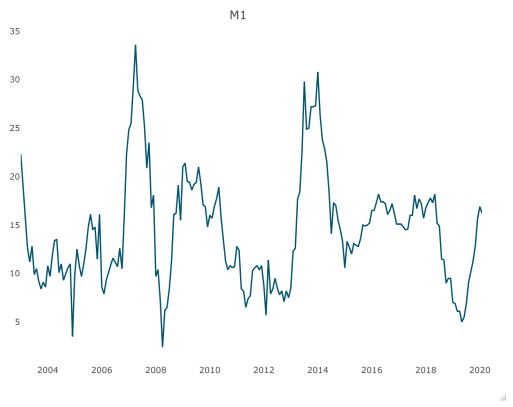
```
lnIP < − ts(mp$lnIP, start = c(2003,1,1), frequency = 12)
lnM1 < − ts(mp$lnM1, start = c(2003,1,1), frequency = 12)
M1 < − ts(mp$M1, start = c(2003,1,1), frequency = 12)
CPI < − ts(mp$CPI, start = c(2003,1,1), frequency = 12)
RRP < − ts(mp$RRP, start = c(2003,1,1), frequency = 12)
```

We can also visualize our series using the autoplot() or ts_plot() command. As before, the ts_plot() command is a more interactive version using the plot.ly package as base.

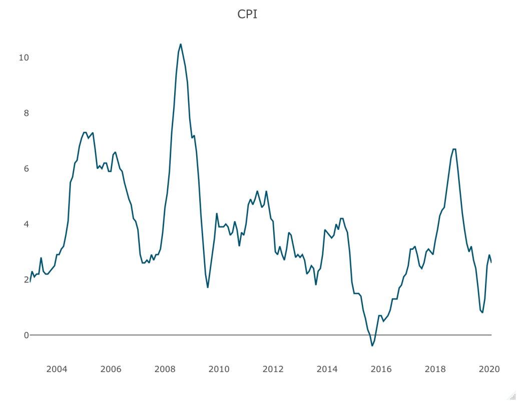
```
ts_plot(lnIP)
ts_plot(lnM1)
ts_plot(M1)
ts_plot(CPI)
ts_plot(RRP)
```



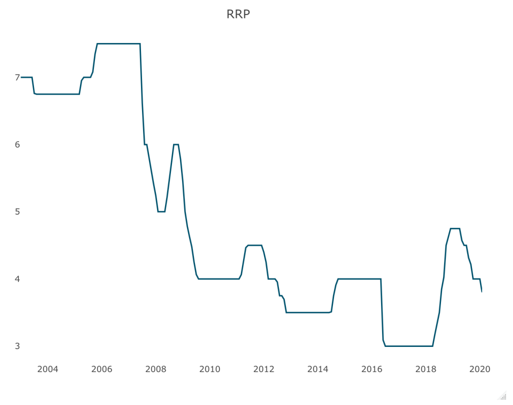
(a) Log of Industrial Production



(b) M1 Money Supply



(c) CPI Inflation Rate



(d) Overnight Reverse Repurchase Rate

Figure 1: Time Series Plots

1.1.2 Testing for Non-Stationarity

Again, it is important to assess whether the variables under study are stationary or not. As we have said, having stationary variables is of an ideal case in our VAR even if we can run it without these. As we have been accustomed to, let us use the tests we are familiar with. For simplicity, we will use the Phillips Perron so we would not need to specify the number of lags.

```
pp.test(lnIP)
pp.test(M1)
pp.test(lnM1)
pp.test(CPI)
pp.test(RRP)
```

In the estimation above, we find that all variables with the exception of lnM1 are non-stationary variables. Remember that a rejection of the null hypothesis suggests the data is stationary. Nevertheless, we can still run a VAR estimation using these level data. However, you may opt to difference the data and see if that provides better results and forecasts.

1.2 Estimation Proper

The time has come to formally estimate our VAR. We will first need to bind our VAR variables together to create the system. After which, we will select the optimal lag order behind the VAR we will be using. We will then run an unrestricted VAR estimation and see the results. Lastly, we will run some diagnostics such as tests for autocorrelation, stability, and normality.

1.2.1 Building the VAR System

The first step is to build the VAR system. This is done though the cbind() command which essentially groups our time series. We will order this in the desired order that we see fit. We will store this in an object called "v1". We will then rename the variables as the list to follow using the colnames() command.

```
v1 < − cbind(RRP, lnM1, CPI, lnIP)
colnames(v1) < − cbind("RRP","M1","CPI", "lnIP")
```

1.2.2 Lag Selection

After we bind the variables and created the VAR system, we will determine some lag order which we will use. To do this, we use VARselect() command and use the v1 object we just created. We will use a maximum lag order of 15. The command will automatically generate the preferred lag order based on the multivariate iterations of the AIC, SBIC, HQIC and the FPE.

```
lagselect < − VARselect(v1, lag.max = 15, type = "const")
lagselect$selection
```

Running the commands suggest that the lag order to be used is 2. In the study of Sims, he used 14 lags which may have been more adept in US data as the effects reflect greater persistence.

1.2.3 Estimating the VAR Model

We will now estimate a model. We estimate the VAR using the VAR() command. The *p* option refers to the number of lags used. Since we determined that 2 lags was best, we set this to 2. We let it be a typical unrestricted VAR with a constant and we will specify no exogenous variables in the system. The summary() command lists down the results.

```
Model1 < − VAR(v1, p = 2, type = "const", season = NULL, exog = NULL)
summary(Model1)
```

We do not typically interpret the coefficients of the VAR, we typically interpret the results of the applications. You will see however that we have coefficients there for each lag and each equation in the VAR. Each equation represents an equation in the VAR system.

1.2.4 Autocorrelation

One assumption is that the residuals should, as much as possible, be non-autocorrelated. This is again on our assumption that the residuals are white noise and thus uncorrelated with the previous periods. To do this, we run the serial.test() command. We store our results in an object Serial1.

```
Serial1 < − serial.test(Model1, lags.pt = 5, type = "PT.asymptotic")
Serial1
```

In this test, we see that the residuals do not show signs of autocorrelation. However, there is a chance that if we change the maximum lag order, there could be a sign of autocorrelation. As such, it is best to experiment with multiple lag orders.

1.2.5 ARCH Test

Another aspect to consider is the presence of heteroscedasticity. In time series, there is what we call *ARCH effects* which are essentially clustered volatility areas in a time series. This is common is series' such as stock prices where massive increases or decreases could be seen when an earnings call is released. In that area or window, there could be excessive volatility thereby changing the variance of the residuals, far from our assumption of a constant variance. We have models to account for these which are the conditional volatility models which we will discuss in a later section.

```
Arch1 < − arch.test(Model1, lags.multi = 15, multivariate.only = TRUE)
Arch1
```

Again, the results of the ARCH test signify no degree of hetroscedasticity as we fail to reject the null hypothesis. Therefore, we conclude that there are no ARCH effects in this model. However, like with the autocorrelation test, it is possible to register lag effects at subsequent lag orders.

1.2.6 Normality of the Residuals

A soft pre-requisite but a desirable one is the normality of the distribution of the residuals. To test for the normality of the residuals, we use the normality.test() command in R which brings in the Jarque-Bera test, the Kurtosis Test, and the Skewness test.

```
Norm1 < − normality.test(Model1, multivariate.only = TRUE)
Norm1
```

Based on all the three results, it appears that the residuals of this particular model are not normally distributed.

1.2.7 Stability Test

The stability test is some test for the presence of structural breaks. We know that if we are unable to test for structural breaks and if there happened to be one, the whole estimation may be thrown off. Fortunately, we have a simple test for this which uses a plot of the sum of recursive residuals. If at any point in the graph, the sum goes out of the red critical bounds, then a structural break at that point was seen.

```
Stability1 < − stability(Model1, type = "OLS-CUSUM")
plot(Stability1)
```

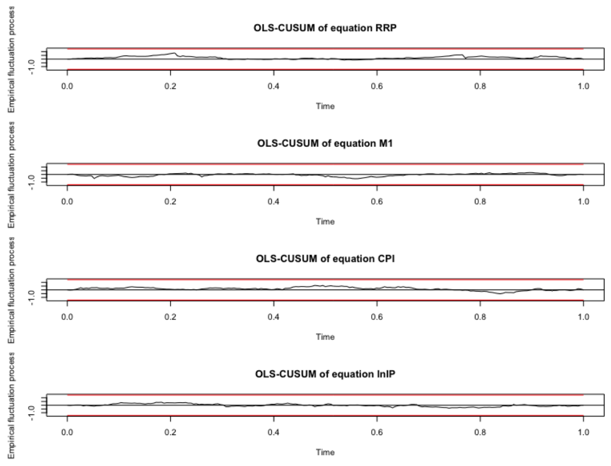


Figure 2: Stability Test in VAR

Based on the results of the test, there seems to be no structural breaks evident. As such, our model passes this particular test.

1.3 Policy Simulations

We will now move on to policy simulations in a regular VAR. We talked about three main ones, which are the Granger Causality, Forecast Error Variance Decomposition, as well as the Impulse Response Functions. While this was discussed in SVAR, this is also applicable for a regular unrestricted VAR.

1.3.1 Granger Causality

We will test for an overall granger causality testing each variable in the system against all the others. As we said, there could be a unidirectional, bidirectional, or no causality relationships between variables.

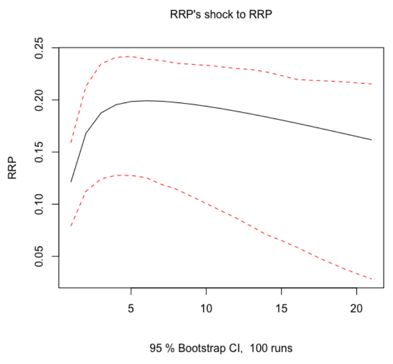
```
GrangerRRP< − causality(Model1, cause = "RRP")
GrangerRRP

GrangerM1 < − causality(Model1, cause = "M1")
GrangerM1

GrangerCPI < − causality(Model1, cause = "CPI")
GrangerCPI

GrangerlnIP < − causality(Model1, cause = "lnIP")
GrangerlnIP
```

The command used is suitable for bivariate cases but we will use it in a system of four for now. Some of you may try reducing the VAR models to two variables to see more straightforward interpretations. To perform the Granger causality test, we use the causality() command. Based on the results, we conclude that CPI Granger causes the other variables but the converse is not seen. Maybe granular variable to variable rather than variable to group relationships would prove more significant.

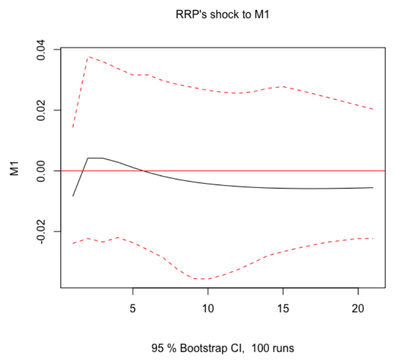
1.3.2 Impulse Response Functions

We will now turn to our impulse response functions. While we can get more impulse response functions than the ones below, we will zero in on the impact of a shock in RRP to the other variables in the system

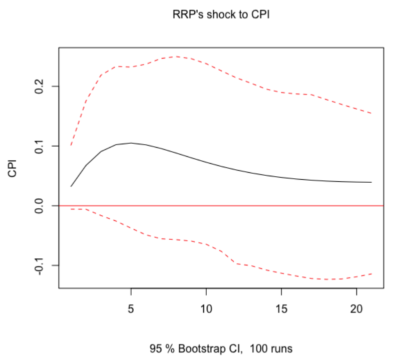
```
RRPirf < − irf(Model1, impulse = "RRP", response = "RRP", n.ahead = 20, boot = TRUE)
plot(RRPirf, ylab = "RRP", main = "RRP's shock to RRP")

M1irf < − irf(Model1, impulse = "RRP", response = "M1", n.ahead = 20, boot = TRUE)
plot(M1irf, ylab = "M1", main = "RRP's shock to M1")

CPIirf < − irf(Model1, impulse = "RRP", response = "CPI", n.ahead = 20, boot = TRUE)
plot(CPIirf, ylab = "CPI", main = "RRP's shock to CPI")

lnIPirf < − irf(Model1, impulse = "RRP", response = "lnIP", n.ahead = 20, boot = TRUE)
plot(lnIPirf, ylab = "lnIP", main = "RRP's shock to lnIP")
```



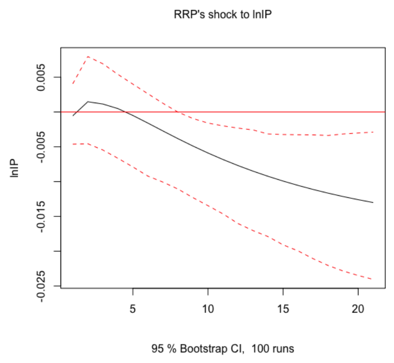
(a) Shock of RRP to RRP



(b) Shock of RRP to M1 Money Supply



(c) Shock of RRP to CPI



(d) Shock of RRP to lnIP

Figure 3: Impulse Responses in VAR

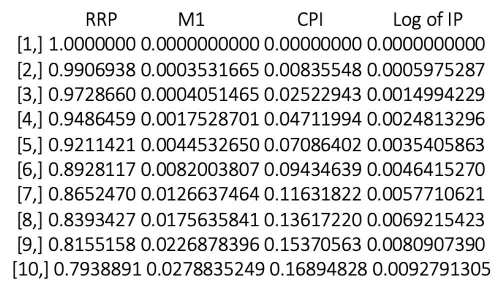
The irf() commnd generates the IRFs where we need to specify the model, what the impulse series will be, and what the response series will be. We can also specify the number of periods ahead to see how the impact or shock will progress over time. We then use the plot() command to graph this IRF. The results from the IRF are quite *puzzling*. Remember that such an increase in the RRP cannot be a reaction of the BSP to what is happening the the other variables since it was ordered first. As you can see, in (a), the shock to the RRP will of course increase RRP, this would then lead to a slight fall in the money supply (b) and a fall in output (d). What we also find is that prices increase in the short run but decrease moderately thereafter. According to Christopher Sims, this is a puzzling result. The results suggest that prices go up after an RRP hike. If a monetary contraction reduces aggregate demand (lnIP) thereby lowering output, it cannot be associated with inflation. He goes on to say that the VAR could potentially be miss-specified. For instance, there could be a leading indicator for inflation to which the BSP will reach and which was wrongly omitted from the VAR. The BSP could know that inflatioary pressures are about to arrive and counteract that by raising the interest rate.

If you recall basic macroeconomics, a contractionary monetary policy like raising the policy rate (interest rate) will lead to a fall in output. That fall in output should generally weigh inflation down and cause prices to decrease. However, what we see in this model is that prices have increased, for the most part, which is puzzling. Supposedly, exogenous movements of the policy interest rate in the previous estimation were not totally exogenos. In other words, the exogenous shocks were not properly identified.

1.3.3 Forecast Error Variance Decomposition

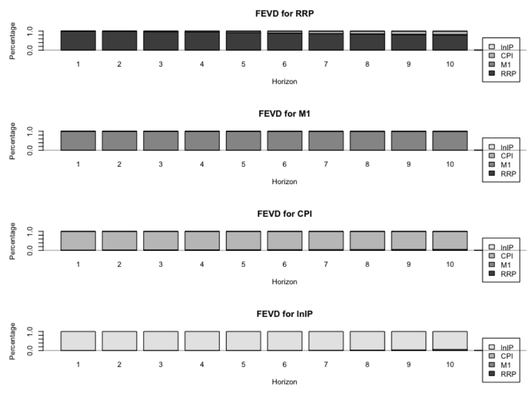
We now turn our attention to the forecast error variance decomposition. Again, we can trace the development of shocks in the our system to explaining the forecast error variances of all the variables in the system. To do this, we use the fevd() command. Like the IRF, we can also specify the number of periods ahead. For this case, let us just zero in on RRP.

```
FEVD1 < − fevd(Model1, n.ahead = 10)
FEVD1
plot(FEVD1)
```



```
           RRP          M1         CPI    Log of IP
 [1,] 1.0000000 0.0000000000 0.00000000 0.0000000000
 [2,] 0.9906938 0.0003531665 0.00835548 0.0005975287
 [3,] 0.9728660 0.0004051465 0.02522943 0.0014994229
 [4,] 0.9486459 0.0017528701 0.04711994 0.0024813296
 [5,] 0.9211421 0.0044532650 0.07086402 0.0035405863
 [6,] 0.8928117 0.0082003807 0.09434639 0.0046415270
 [7,] 0.8652470 0.0126637464 0.11631822 0.0057710621
 [8,] 0.8393427 0.0175635841 0.13617220 0.0069215423
 [9,] 0.8155158 0.0226878396 0.15370563 0.0080907390
[10,] 0.7938891 0.0278835249 0.16894828 0.0092791305
```

(a) Table Summary of the FEVD



(b) FEVD Plot

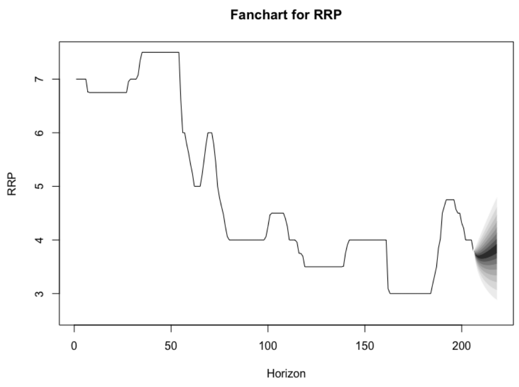
Figure 4: Forecast Error Variance Decomposition in VAR

We can clearly see in window (a) that the forecast error of the RRP (column 1) at short horizons is due to itself. This is the case because the RRP was placed first in the ordering and that no other shocks affect RRP contemporaneously. At longer horizons say 10 months, we can see that CPI now accounts for about 16 percent and that money supply accounts for about 2 percent respectively.

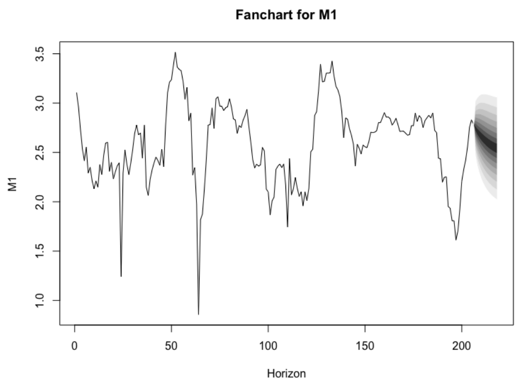
1.3.4 Forecasting using a VAR

As we have said, we can also forecast using VAR. To do this, we use the predict() command and generate something called a *fanchart* which is commonly used in identifying the confidence level of forecasts in a graphical manner. We will set the forecast horizon to 12 months ahead or a full year forecast.

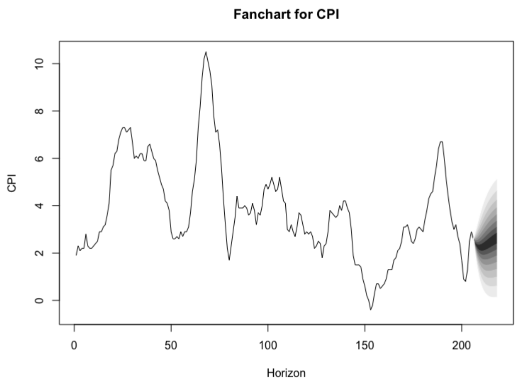
```
forecast < − predict(Model1, n.ahead = 12, ci = 0.95)
fanchart(forecast, names = "RRP", main = "Fanchart for RRP", xlab = "Horizon", ylab = "RRP")
fanchart(forecast, names = "M1", main = "Fanchart for M1", xlab = "Horizon", ylab = "M1")
fanchart(forecast, names = "CPI", main = "Fanchart for CPI", xlab = "Horizon", ylab = "CPI")
fanchart(forecast, names = "lnIP", main = "Fanchart for lnIP", xlab = "Horizon", ylab = "lnIP")
forecast
```



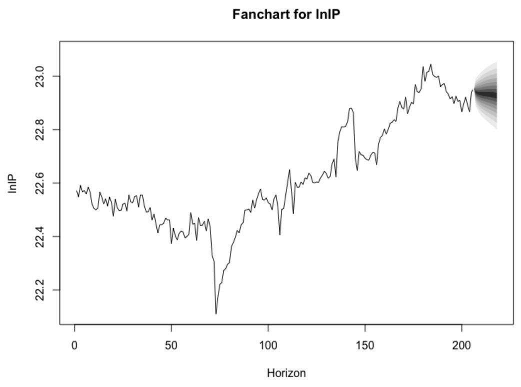
(a) Fanchart for RRP



(b) Fanchart for M1 Money Supply



(c) Fanchart for CPI



(d) Fanchart for lnIP

Figure 5: Fancharts in VAR

The figure shows that RRP is expected to decrease slightly then increase, M1 is expected to decrease and so is IP. CPI is expected to decrease slightly in the first few months then rebound moderately. Take note that this is different from the IRFs simply because we are using VAR as a forecasting tool instead of a policy tool. All in all, I hope with this example you are able to see the many use cases of the VAR methodology and why many economists continue to use it for its flexibility.

1.4 Notes

- This section is taken from my lecture notes in time series econometrics. For a better introduction to time series, consider reading Hamilton or Brooks.