# VAR Explanation

## Contents

**Setup** 2

Load packages & functions . . . 2

Load Data . . . 3

**Stationarity** 3

**Information Criteria** 10

**VAR Models** 11

Volatility & Tariff Mention . . . 11

**SVAR Model** 12

B matrix . . . 12

IRF . . . 13

Granger test . . . 14

# Setup

## Load packages & functions

```r
rm(list=ls())
require(tinytex) #LaTeX
require(ggplot2) #plots
require(AEC) #JP-Renne functions
require(AER) #NW formula
require(forecast) #time series stuff
require(expm) #matrix exponents
require(here) #directory finder
require(stringr) # analysis of strings, important for the detection in tweets
require(dplyr) #data management
require(lubridate) #data dates management
require(zoo) #for lagging
require(jtools) #tables
require(huxtable) #tables
require(lmtest) #reg tests
require(vroom) #for loading data
require(data.table) #for data filtering
require(sysid) #for ARMA-X modeling
require(sandwhich) #regression errors
require(stargazer) #nice reg tables
require(tidytext) #text mining
require(textstem) #lemmatization
require(quanteda) #tokenization
require(texreg) #arima tables
require(vars) #VAR models
require(xts) #time series objects
require(tseries) #includes adf test
require(quantmod)
require(TSA)
require(aTSA)
require(tibble)
require(FinTS)
require(kableExtra)
require(writexl)
require(purrr)

getwd()
#setwd("...") -> set wd at base repo folder

#load helper functions
source(here("helperfunctions/data_loaders.R"))
source(here("helperfunctions/date_selector.R"))
source(here("helperfunctions/plotters.R"))
source(here("helperfunctions/quick_arma.R"))
source(here("helperfunctions/r.vol_calculators.R"))
source(here("helperfunctions/truths_cleaning_function.R"))
source(here("helperfunctions/armax_functions.R"))
source(here("helperfunctions/var_irf.R"))
```

### Load Data

We connect R to our GitHub folder where the data are stored. We then load the dataset and select the relevant time window for our analysis.

```r
#load final dataset
source(here("helperfunctions/full_data.R"))

#select timeframe
Vdata = filter(data,between(timestamp, as.Date('2014-01-01'), as.Date('2025-05-07')))
```

## Stationarity

We begin by testing whether our variables are stationary over time. We use the Augmented Dickey-Fuller test on the volatility of different markets and on variables derived from Trump post.

```r
adf.test(data$SPY_vol)
```

```
## Augmented Dickey-Fuller Test
## alternative: stationary
##
## Type 1: no drift no trend
##       lag   ADF p.value
##  [1,]   0 -85.7    0.01
##  [2,]   1 -64.6    0.01
##  [3,]   2 -49.9    0.01
##  [4,]   3 -41.0    0.01
##  [5,]   4 -36.1    0.01
##  [6,]   5 -29.5    0.01
##  [7,]   6 -26.0    0.01
##  [8,]   7 -24.0    0.01
##  [9,]   8 -23.0    0.01
## [10,]   9 -22.2    0.01
## [11,]  10 -21.1    0.01
## [12,]  11 -20.0    0.01
## [13,]  12 -18.2    0.01
## [14,]  13 -17.0    0.01
## Type 2: with drift no trend
##       lag   ADF p.value
##  [1,]   0 -88.9    0.01
##  [2,]   1 -67.5    0.01
##  [3,]   2 -52.5    0.01
##  [4,]   3 -43.2    0.01
##  [5,]   4 -38.2    0.01
##  [6,]   5 -31.3    0.01
##  [7,]   6 -27.7    0.01
##  [8,]   7 -25.6    0.01
##  [9,]   8 -24.6    0.01
## [10,]   9 -23.7    0.01
## [11,]  10 -22.5    0.01
## [12,]  11 -21.4    0.01
## [13,]  12 -19.5    0.01
```

```
## [14,]  13 -18.2    0.01
## Type 3: with drift and trend
##       lag   ADF p.value
##  [1,]   0 -91.1    0.01
##  [2,]   1 -69.6    0.01
##  [3,]   2 -54.3    0.01
##  [4,]   3 -44.9    0.01
##  [5,]   4 -39.7    0.01
##  [6,]   5 -32.6    0.01
##  [7,]   6 -28.9    0.01
##  [8,]   7 -26.7    0.01
##  [9,]   8 -25.8    0.01
## [10,]   9 -24.9    0.01
## [11,]  10 -23.7    0.01
## [12,]  11 -22.5    0.01
## [13,]  12 -20.5    0.01
## [14,]  13 -19.2    0.01
## ----
## Note: in fact, p.value = 0.01 means p.value <= 0.01
```

```r
adf.test(data$VGK_vol)
```

```
## Augmented Dickey-Fuller Test
## alternative: stationary
##
## Type 1: no drift no trend
##       lag    ADF p.value
##  [1,]   0 -106.3    0.01
##  [2,]   1  -80.7    0.01
##  [3,]   2  -66.1    0.01
##  [4,]   3  -56.0    0.01
##  [5,]   4  -49.1    0.01
##  [6,]   5  -43.2    0.01
##  [7,]   6  -38.9    0.01
##  [8,]   7  -35.9    0.01
##  [9,]   8  -33.5    0.01
## [10,]   9  -31.3    0.01
## [11,]  10  -29.4    0.01
## [12,]  11  -27.8    0.01
## [13,]  12  -26.1    0.01
## [14,]  13  -24.4    0.01
## Type 2: with drift no trend
##       lag    ADF p.value
##  [1,]   0 -110.7    0.01
##  [2,]   1  -85.2    0.01
##  [3,]   2  -70.5    0.01
##  [4,]   3  -60.4    0.01
##  [5,]   4  -53.4    0.01
##  [6,]   5  -47.3    0.01
##  [7,]   6  -43.0    0.01
##  [8,]   7  -40.0    0.01
##  [9,]   8  -37.5    0.01
## [10,]   9  -35.2    0.01
## [11,]  10  -33.3    0.01
```

```
## [12,]  11  -31.6    0.01
## [13,]  12  -29.8    0.01
## [14,]  13  -28.0    0.01
## Type 3: with drift and trend
##       lag    ADF p.value
##  [1,]   0 -111.2    0.01
##  [2,]   1  -85.7    0.01
##  [3,]   2  -71.0    0.01
##  [4,]   3  -61.0    0.01
##  [5,]   4  -53.9    0.01
##  [6,]   5  -47.9    0.01
##  [7,]   6  -43.5    0.01
##  [8,]   7  -40.5    0.01
##  [9,]   8  -38.0    0.01
## [10,]   9  -35.7    0.01
## [11,]  10  -33.8    0.01
## [12,]  11  -32.1    0.01
## [13,]  12  -30.3    0.01
## [14,]  13  -28.5    0.01
## ----
## Note: in fact, p.value = 0.01 means p.value <= 0.01
```

```r
adf.test(data$ASHR_vol)
```

```
## Augmented Dickey-Fuller Test
## alternative: stationary
##
## Type 1: no drift no trend
##       lag   ADF p.value
##  [1,]   0 -92.8    0.01
##  [2,]   1 -66.5    0.01
##  [3,]   2 -53.7    0.01
##  [4,]   3 -44.7    0.01
##  [5,]   4 -38.3    0.01
##  [6,]   5 -32.6    0.01
##  [7,]   6 -25.4    0.01
##  [8,]   7 -25.2    0.01
##  [9,]   8 -24.1    0.01
## [10,]   9 -23.3    0.01
## [11,]  10 -22.6    0.01
## [12,]  11 -21.3    0.01
## [13,]  12 -19.8    0.01
## [14,]  13 -16.5    0.01
## Type 2: with drift no trend
##       lag    ADF p.value
##  [1,]   0 -100.4    0.01
##  [2,]   1  -73.2    0.01
##  [3,]   2  -59.9    0.01
##  [4,]   3  -50.5    0.01
##  [5,]   4  -43.6    0.01
##  [6,]   5  -37.4    0.01
##  [7,]   6  -29.3    0.01
##  [8,]   7  -29.1    0.01
##  [9,]   8  -28.1    0.01
```

```
## [10,]   9  -27.2    0.01
## [11,]  10  -26.5    0.01
## [12,]  11  -25.2    0.01
## [13,]  12  -23.4    0.01
## [14,]  13  -19.6    0.01
## Type 3: with drift and trend
##       lag    ADF p.value
##  [1,]   0 -101.2    0.01
##  [2,]   1  -74.0    0.01
##  [3,]   2  -60.7    0.01
##  [4,]   3  -51.2    0.01
##  [5,]   4  -44.3    0.01
##  [6,]   5  -38.0    0.01
##  [7,]   6  -29.8    0.01
##  [8,]   7  -29.7    0.01
##  [9,]   8  -28.6    0.01
## [10,]   9  -27.8    0.01
## [11,]  10  -27.1    0.01
## [12,]  11  -25.7    0.01
## [13,]  12  -23.9    0.01
## [14,]  13  -20.0    0.01
## ----
## Note: in fact, p.value = 0.01 means p.value <= 0.01
```

```r
adf.test(data$dummy)
```

```
## Augmented Dickey-Fuller Test
## alternative: stationary
##
## Type 1: no drift no trend
##       lag    ADF p.value
##  [1,]   0 -702.2    0.01
##  [2,]   1 -134.7    0.01
##  [3,]   2  -92.8    0.01
##  [4,]   3  -73.0    0.01
##  [5,]   4  -61.3    0.01
##  [6,]   5  -53.6    0.01
##  [7,]   6  -47.4    0.01
##  [8,]   7  -24.5    0.01
##  [9,]   8  -23.9    0.01
## [10,]   9  -23.3    0.01
## [11,]  10  -22.7    0.01
## [12,]  11  -22.0    0.01
## [13,]  12  -21.4    0.01
## [14,]  13  -20.7    0.01
## Type 2: with drift no trend
##       lag    ADF p.value
##  [1,]   0 -754.9    0.01
##  [2,]   1 -155.3    0.01
##  [3,]   2 -114.6    0.01
##  [4,]   3  -96.7    0.01
##  [5,]   4  -87.0    0.01
##  [6,]   5  -81.9    0.01
##  [7,]   6  -77.7    0.01
```

```
##  [8,]   7  -41.2    0.01
##  [9,]   8  -41.2    0.01
## [10,]   9  -41.2    0.01
## [11,]  10  -41.2    0.01
## [12,]  11  -41.1    0.01
## [13,]  12  -41.0    0.01
## [14,]  13  -40.6    0.01
## Type 3: with drift and trend
##       lag    ADF p.value
##  [1,]   0 -754.9    0.01
##  [2,]   1 -155.3    0.01
##  [3,]   2 -114.7    0.01
##  [4,]   3  -96.8    0.01
##  [5,]   4  -87.1    0.01
##  [6,]   5  -81.9    0.01
##  [7,]   6  -77.8    0.01
##  [8,]   7  -41.2    0.01
##  [9,]   8  -41.3    0.01
## [10,]   9  -41.2    0.01
## [11,]  10  -41.3    0.01
## [12,]  11  -41.2    0.01
## [13,]  12  -41.0    0.01
## [14,]  13  -40.7    0.01
## ----
## Note: in fact, p.value = 0.01 means p.value <= 0.01
```

```r
adf.test(data$N)
```

```
## Augmented Dickey-Fuller Test
## alternative: stationary
##
## Type 1: no drift no trend
##       lag    ADF p.value
##  [1,]   0 -482.3    0.01
##  [2,]   1 -132.8    0.01
##  [3,]   2  -93.1    0.01
##  [4,]   3  -73.8    0.01
##  [5,]   4  -62.1    0.01
##  [6,]   5  -54.7    0.01
##  [7,]   6  -47.5    0.01
##  [8,]   7  -27.6    0.01
##  [9,]   8  -26.8    0.01
## [10,]   9  -26.1    0.01
## [11,]  10  -25.4    0.01
## [12,]  11  -24.6    0.01
## [13,]  12  -23.8    0.01
## [14,]  13  -22.8    0.01
## Type 2: with drift no trend
##       lag    ADF p.value
##  [1,]   0 -509.5    0.01
##  [2,]   1 -147.4    0.01
##  [3,]   2 -108.4    0.01
##  [4,]   3  -90.0    0.01
##  [5,]   4  -79.3    0.01
```

```
## [6,]   5  -73.1    0.01
## [7,]   6  -66.1    0.01
## [8,]   7  -39.2    0.01
## [9,]   8  -38.7    0.01
## [10,]   9  -38.3    0.01
## [11,]  10  -37.9    0.01
## [12,]  11  -37.4    0.01
## [13,]  12  -36.8    0.01
## [14,]  13  -35.8    0.01
## Type 3: with drift and trend
##       lag    ADF p.value
##  [1,]   0 -509.5    0.01
##  [2,]   1 -147.4    0.01
##  [3,]   2 -108.4    0.01
##  [4,]   3  -90.0    0.01
##  [5,]   4  -79.3    0.01
##  [6,]   5  -73.1    0.01
##  [7,]   6  -66.1    0.01
##  [8,]   7  -39.2    0.01
##  [9,]   8  -38.7    0.01
## [10,]   9  -38.3    0.01
## [11,]  10  -37.9    0.01
## [12,]  11  -37.4    0.01
## [13,]  12  -36.8    0.01
## [14,]  13  -35.8    0.01
## ----
## Note: in fact, p.value = 0.01 means p.value <= 0.01
```

```r
adf.test(data$tariff)
```

```
## Augmented Dickey-Fuller Test
## alternative: stationary
##
## Type 1: no drift no trend
##       lag    ADF p.value
##  [1,]   0 -139.4    0.01
##  [2,]   1  -92.0    0.01
##  [3,]   2  -77.4    0.01
##  [4,]   3  -65.3    0.01
##  [5,]   4  -56.7    0.01
##  [6,]   5  -49.6    0.01
##  [7,]   6  -40.6    0.01
##  [8,]   7  -36.9    0.01
##  [9,]   8  -35.1    0.01
## [10,]   9  -34.1    0.01
## [11,]  10  -32.7    0.01
## [12,]  11  -31.2    0.01
## [13,]  12  -30.4    0.01
## [14,]  13  -28.4    0.01
## Type 2: with drift no trend
##       lag    ADF p.value
##  [1,]   0 -140.0    0.01
##  [2,]   1  -92.5    0.01
##  [3,]   2  -78.0    0.01
```

```
##  [4,]   3  -65.8    0.01
##  [5,]   4  -57.2    0.01
##  [6,]   5  -50.1    0.01
##  [7,]   6  -41.1    0.01
##  [8,]   7  -37.3    0.01
##  [9,]   8  -35.6    0.01
## [10,]   9  -34.6    0.01
## [11,]  10  -33.2    0.01
## [12,]  11  -31.6    0.01
## [13,]  12  -30.9    0.01
## [14,]  13  -28.9    0.01
## Type 3: with drift and trend
##       lag    ADF p.value
##  [1,]   0 -140.4    0.01
##  [2,]   1  -92.9    0.01
##  [3,]   2  -78.4    0.01
##  [4,]   3  -66.2    0.01
##  [5,]   4  -57.6    0.01
##  [6,]   5  -50.6    0.01
##  [7,]   6  -41.5    0.01
##  [8,]   7  -37.7    0.01
##  [9,]   8  -35.9    0.01
## [10,]   9  -34.9    0.01
## [11,]  10  -33.6    0.01
## [12,]  11  -32.0    0.01
## [13,]  12  -31.3    0.01
## [14,]  13  -29.3    0.01
## ----
## Note: in fact, p.value = 0.01 means p.value <= 0.01
```

```
adf.test(data$china)
```

```
## Augmented Dickey-Fuller Test
## alternative: stationary
##
## Type 1: no drift no trend
##       lag    ADF p.value
##  [1,]   0 -586.8    0.01
##  [2,]   1 -130.3    0.01
##  [3,]   2  -92.2    0.01
##  [4,]   3  -76.0    0.01
##  [5,]   4  -64.7    0.01
##  [6,]   5  -55.8    0.01
##  [7,]   6  -48.8    0.01
##  [8,]   7  -40.1    0.01
##  [9,]   8  -38.5    0.01
## [10,]   9  -36.8    0.01
## [11,]  10  -35.0    0.01
## [12,]  11  -32.7    0.01
## [13,]  12  -31.8    0.01
## [14,]  13  -30.7    0.01
## Type 2: with drift no trend
##       lag    ADF p.value
##  [1,]   0 -592.5    0.01
```

```
##   [2,]   1 -132.6    0.01
##   [3,]   2  -94.5    0.01
##   [4,]   3  -78.4    0.01
##   [5,]   4  -67.1    0.01
##   [6,]   5  -58.2    0.01
##   [7,]   6  -51.1    0.01
##   [8,]   7  -42.2    0.01
##   [9,]   8  -40.7    0.01
##  [10,]   9  -39.1    0.01
##  [11,]  10  -37.3    0.01
##  [12,]  11  -34.9    0.01
##  [13,]  12  -34.1    0.01
##  [14,]  13  -33.0    0.01
## Type 3: with drift and trend
##        lag    ADF p.value
##   [1,]   0 -592.9    0.01
##   [2,]   1 -132.8    0.01
##   [3,]   2  -94.6    0.01
##   [4,]   3  -78.6    0.01
##   [5,]   4  -67.3    0.01
##   [6,]   5  -58.4    0.01
##   [7,]   6  -51.3    0.01
##   [8,]   7  -42.4    0.01
##   [9,]   8  -40.8    0.01
##  [10,]   9  -39.3    0.01
##  [11,]  10  -37.4    0.01
##  [12,]  11  -35.1    0.01
##  [13,]  12  -34.3    0.01
##  [14,]  13  -33.1    0.01
## ----
## Note: in fact, p.value = 0.01 means p.value <= 0.01
```

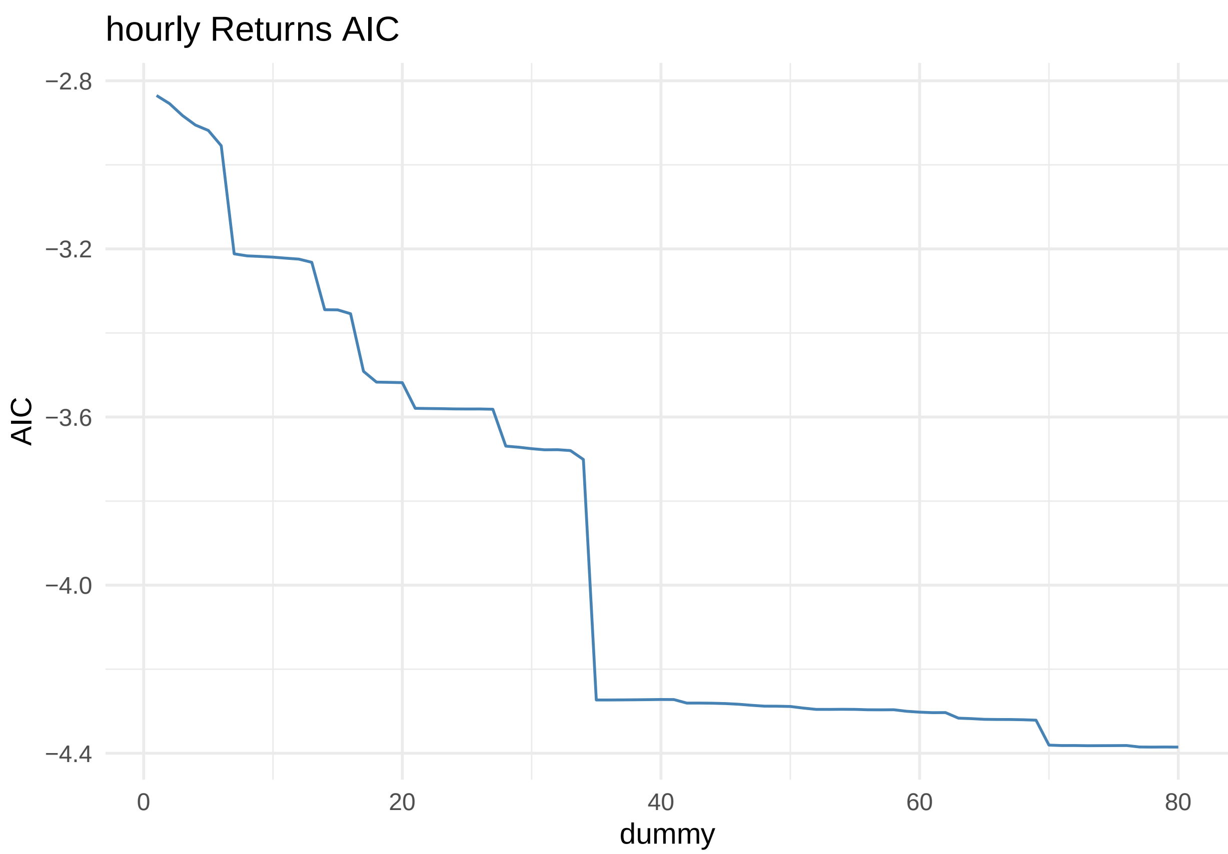
## Information Criteria

we determine the optimal number of lags using several Information Criteria (AIC, SC, HQ, FPE).

Here is an example using the AIC

```r
##with dummy
y = cbind(Vdata$dummy, Vdata$SPY_vol)

y_lag = VARselect(y, lag.max = 80)
y_list = list(y_lag)


##AIC
resultats1 <- lapply(y_list, function(x) x$criteria["AIC(n)", ])

resultats1 = as.data.frame(resultats1)

resultats1 = resultats1 %>%
  rename(name = 1) %>%
```

```
  mutate(
    n = c(1:length(name))
  )

ggplot (resultats1, aes(x=n, y= name)) +
  geom_line(color = "steelblue") +
  labs(title = "hourly Returns AIC",x="dummy" , y = "AIC") +
  theme_minimal()
```




## VAR Models

### Volatility & Tariff Mention

We first extract and bind the relevant variables, estimate a VAR model using the VAR function, and then display the results in a table for the volatility equation. Our main interest is in the effect of tariff mentions on volatility (and not the inverse).

```
y3 = cbind(Vdata$tariff, Vdata$SPY_vol)
colnames(y3)[1:2] <- c("tariff", "vol")
est.VAR3 <- VAR(y3,p=6)
mod_vol <- est.VAR3$varresult$vol
screenreg(mod_vol, digits = 6)
```

```
##
## ===========================
##              Model 1
## ---------------------------
## tariff.l1      0.019718 ***
##               (0.001454)
## vol.l1         0.342081 ***
##               (0.006981)
## tariff.l2      0.005269 ***
##               (0.001460)
## vol.l2         0.027464 ***
##               (0.007388)
## tariff.l3     -0.007797 ***
##               (0.001464)
## vol.l3         0.075380 ***
##               (0.007374)
## tariff.l4      0.002275
##               (0.001463)
## vol.l4         0.088777 ***
##               (0.007383)
## tariff.l5     -0.001145
##               (0.001456)
## vol.l5         0.026049 ***
##               (0.007407)
## tariff.l6     -0.002750
##               (0.001457)
## vol.l6         0.167546 ***
##               (0.006969)
## const          0.005770 ***
##               (0.000585)
## ---------------------------
## R^2            0.291693
## Adj. R^2       0.291267
## Num. obs.  19965
## ===========================
## *** p < 0.001; ** p < 0.01; * p < 0.05
```

# SVAR Model

## B matrix

In order to implement our SVAR framework with a short run restriction, we need to reconstruct the Variance-Covariance Omega Matrix with said restriction. Here, as we have 2 outcome variables, and as the Omega Matrix is 2x2, we only need n(n-1)/2 restrictions, which is one restriction. Our assumption is that while the market reacts instantly to Trump posts, Trump does not react contemporaneously with changes in market volatility. For constructing our BB' matrix we define a matrix in a function and find the square distance between the True Omega matrix and the constructed BB matrix. We then use a optimization function in order to find the elements of B matrix (B.hat) that minimize the distance with the true Variance-Covariance matrix.

```
Omega3 <- var(residuals(est.VAR3))
```

```r
#create the B matrix
loss3 <- function(param3){
  #define the restriction
  B3 <- matrix(c(param3[1], param3[2], 0, param3[3]), ncol = 2)

  #make BB' approximatively equal to omega
  X3 <- Omega3 - B3 %*% t(B3)

  #loss function
  loss3 <- sum(X3^2)
  return(loss3)
}

res.opt3 <- optim(c(1, 0, 1), loss3, method = "BFGS")
B.hat3 <- matrix(c(res.opt3$par[1], res.opt3$par[2], 0, res.opt3$par[3]), ncol = 2)

print(cbind(Omega3,B.hat3 %*% t(B.hat3)))
```

```
##              tariff          vol
## tariff 0.1422112842 0.0007678487 0.1422102848 0.0007678501
## vol    0.0007678487 0.0060214190 0.0007678501 0.0060204238
```

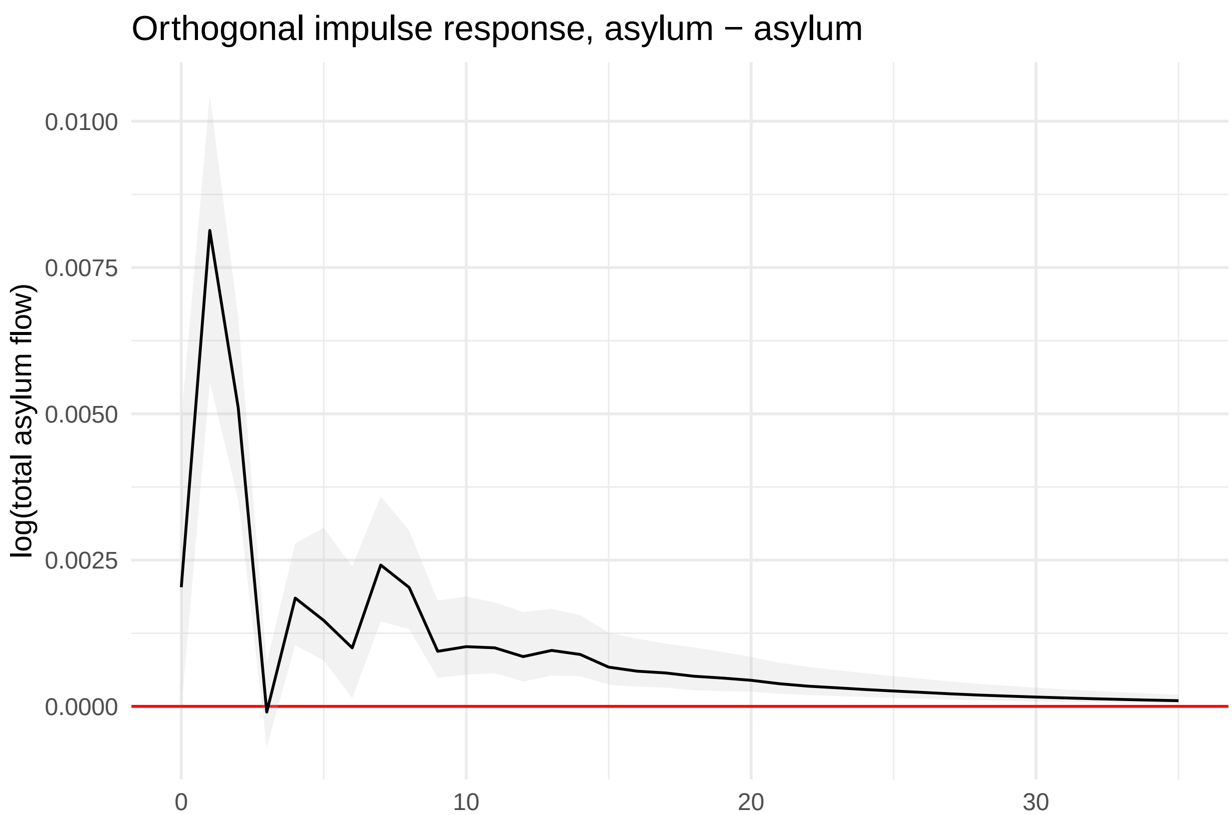
```r
#shock by tariff
B.hat3
```

```
##             [,1]       [,2]
## [1,] 0.377107789 0.00000000
## [2,] 0.002036156 0.07756467
```

## IRF

We then use the irf() function of the vars packages in order to graph the effect of a choc of Trump post (here tariff mentions) on market volatility using the coefficient of our est.VAR function and using the choc coefficient of our estimated B matrix with our restriction.

```r
#irf creation
irf_res <- irf(est.VAR3, impulse = "tariff", response = "vol",
                  bmat=b.hat3, n.ahead = 7 * 5, boot = TRUE, ci = 0.95)

#function to extract relevant objects for plotting
single_varirf <- extract_varirf(irf_res)

#the plot
single_varirf %>%
  ggplot(aes(x=period, y=irf_tariff_vol, ymin=lower_tariff_vol, ymax=upper_tariff_vol)) +
  geom_hline(yintercept = 0, color="red") +
  geom_ribbon(fill="grey", alpha=0.2) +
  geom_line() +
  theme_light() +
  ggtitle("Orthogonal impulse response, asylum - asylum")+
  ylab("log(total asylum flow)")+
  xlab("") +
  theme_minimal()
```




### Granger test

Finally, we use a Granger causality test to evalutate the robustness of the correlation we've found. We look for Granger causalities in both directions, i.e. whether volatility Granger-causes tariff mentions and vice versa.

```
#does volatility Granger cause tariff mentions
grangertest(y3[,c("vol","tariff")], order = 6)
```

| Res.Df | Df | F | Pr(>F) |
|---|---|---|---|
| 2e+04 | | | |
| 2e+04 | -6 | 37.5 | 1.54e-45 |

```
#does tariff mentions Granger cause volatility
grangertest(y3[,c("tariff", "vol")], order = 6)
```

| Res.Df | Df | F | Pr(>F) |
|---|---|---|---|
| 2e+04 | | | |
| 2e+04 | -6 | 36.8 | 1.12e-44 |