SVAR Models Explanation

Contents

Setup

Load packages & functions

```
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.

```
#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.

```
adf.test(data$SPY_vol)
adf.test(data$VGK_vol)
adf.test(data$ASHR_vol)

adf.test(data$dummy)
adf.test(data$N)
adf.test(data$tariff)
adf.test(data$china)
```

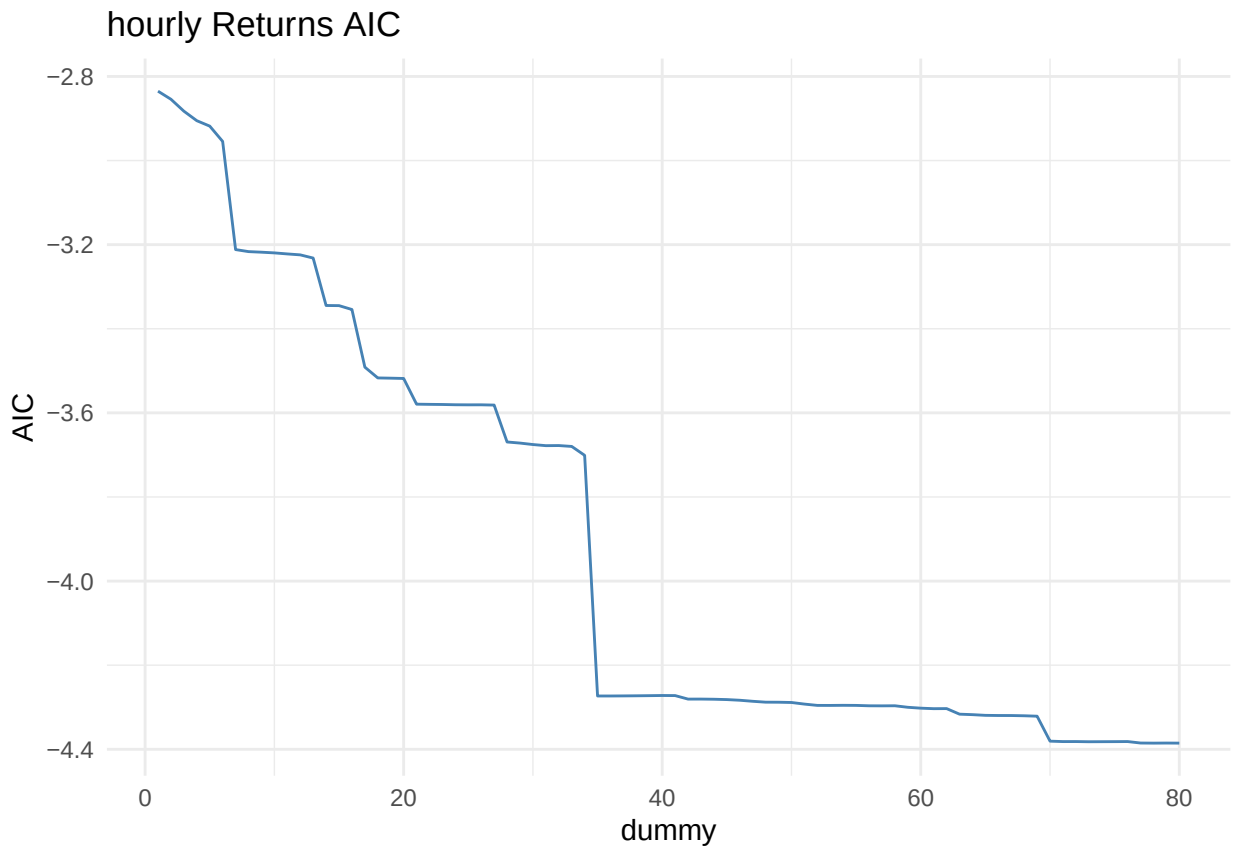
Information Criteria

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

Here is an example using the AIC

```
##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 Model

Volatility & TweetDummy

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 a post (dummy) on volatility (and not the inverse).

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

```
##
## ============================
##             Model 1
## ----------------------------
## dummy.l1       0.000083
##               (0.000188)
```

```
## vol.l1         0.344511 ***
##               (0.006992)
## dummy.l2      -0.000473 *
##               (0.000188)
## vol.l2         0.023714 **
##               (0.007402)
## dummy.l3      -0.000804 ***
##               (0.000189)
## vol.l3         0.082941 ***
##               (0.007372)
## dummy.l4      -0.000546 **
##               (0.000189)
## vol.l4         0.096948 ***
##               (0.007371)
## dummy.l5      -0.000579 **
##               (0.000188)
## vol.l5         0.022887 **
##               (0.007403)
## dummy.l6      -0.000099
##               (0.000188)
## vol.l6         0.164034 ***
##               (0.006989)
## const          0.008726 ***
##               (0.000849)
## ---------------------------
## R^2            0.285135
## Adj. R^2       0.284705
## Num. obs.  19965
## ===========================
## *** p < 0.001; ** p < 0.01; * p < 0.05
```

Serial test

We run a Portmanteau-Godfrey test in order to detect some serial correlation on the residuals. We found strong evidence of the presence of serial correlation in all estimations.

```
serial2 = serial.test(est.VAR, lags.pt =6, type = "PT.asymptotic")
serial2
```

```
##
##  Portmanteau Test (asymptotic)
##
## data:  Residuals of VAR object est.VAR
## Chi-squared = 245.62, df = 0, p-value < 2.2e-16
```

Newey-west estimator

Newey-West correction of Standard Error is implemented in order to correct for bias in the standard errors. We run an OLS with both regressions of the VAR framework in order to use the Newey-West function. Then we calculate the p-value with the robust estimation of the variance of the residuals in order to display correct statistical significance on our tables.

```
#extract results
mod_vol = est.VAR$varresult$vol
f = formula(mod_vol)
d = model.frame(mod_vol)
lm_clean = lm(f, data= d)

#apply Newey-West
nw_vcov = NeweyWest(lm_clean, lag=6)
nw_se = sqrt(diag(nw_vcov))

#t-stats
coef = coef(lm_clean)
t_stat = coef/nw_se

#recalculate p-values
robust = 2*(1-pt(abs(t_stat), df = df.residual(lm_clean)))

nw_se       <- nw_se[names(coef(lm_clean))]
robust      <- robust[names(coef(lm_clean))]

#table
screenreg(lm_clean, override.se = nw_se, override.pvalues = robust, digits = 6)
```

```
##
## ============================
##             Model 1
## ----------------------------
## dummy.l1       0.000083
##               (0.000201)
## vol.l1         0.344511 ***
##               (0.103790)
## dummy.l2      -0.000473 ***
##               (0.000071)
## vol.l2         0.023714
##               (0.042739)
## dummy.l3      -0.000804 ***
##               (0.000088)
## vol.l3         0.082941 ***
##               (0.007496)
## dummy.l4      -0.000546 ***
##               (0.000088)
## vol.l4         0.096948
##               (0.059298)
## dummy.l5      -0.000579 ***
##               (0.000147)
## vol.l5         0.022887 ***
##               (0.006876)
## dummy.l6      -0.000099
##               (0.000101)
## vol.l6         0.164034 ***
##               (0.047379)
## const          0.008726 ***
##               (0.001609)
```

```
## ---------------------------
## R^2            0.325745
## Adj. R^2       0.325306
## Num. obs.  19965
## ===========================
## *** p < 0.001; ** p < 0.01; * p < 0.05
```

We also look for the effect of the AHV on Trump's post, here TweetDummy, with robust standard error

```
#extract results
mod_post = est.VAR$varresult$dummy
ff = formula(mod_post)
dd = model.frame(mod_post)
lm_clean_post = lm(ff, data= dd)

#apply Newey-West
nw_vcov_post = NeweyWest(lm_clean_post, lag=6)
nw_se_post = sqrt(diag(nw_vcov_post))

#t-stats
coef_post = coef(lm_clean_post)
t_stat_post = coef_post/nw_se_post

#recalculate p-values
robust_post = 2*(1-pt(abs(t_stat_post), df = df.residual(lm_clean_post)))

nw_se_post       <- nw_se_post[names(coef(lm_clean_post))]
robust_post      <- robust_post[names(coef(lm_clean_post))]

#table
screenreg(lm_clean_post, override.se = nw_se_post, override.pvalues = robust_post, digits = 6)
```

```
##
## ============================
##            Model 1
## ----------------------------
## dummy.l1      -0.093610 ***
##               (0.003418)
## vol.l1         0.410764
##               (0.303756)
## dummy.l2      -0.085915 ***
##               (0.003447)
## vol.l2        -0.558043
##               (0.306897)
## dummy.l3      -0.076436 ***
##               (0.003627)
## vol.l3        -0.269652
##               (0.350524)
## dummy.l4      -0.077830 ***
##               (0.003667)
## vol.l4        -0.719476 *
##               (0.300555)
## dummy.l5      -0.087761 ***
```

```
##                (0.003688)
## vol.l5        -0.276823
##                (0.169338)
## dummy.l6      -0.096838 ***
##                (0.003939)
## vol.l6         0.973172
##                (0.525025)
## const          1.721194 ***
##                (0.039256)
## ---------------------------
## R^2            0.155107
## Adj. R^2       0.154556
## Num. obs.  19965
## ===========================
## *** p < 0.001; ** p < 0.01; * p < 0.05
```

Breusch-Pagan test

We also run Breusch-Pagan test for checking the heteroscedasticity. We found strong evidence of the presence of heteroscedasticity in all estimations, except for the second term.

```
#HO test whether there is NOT heteroscedasticity.
bptest(lm_clean)
```

```
##
##  studentized Breusch-Pagan test
##
## data:  lm_clean
## BP = 386.02, df = 12, p-value < 2.2e-16
```

B matrix

Then we use the Newey-West formula in order to construct our Robust covariance matrix using the residuals of our estimation

```
#Recreate a Robust Omega Matrix
U = residuals(est.VAR)
T = nrow(U)
L = 6 #number of lag
Omega = matrix(0, ncol(U), ncol(U))
for(l in 0:L) {
  weight = 1 - l/(L+1)
  Gamma_l_ = t(U[(l+1):T, , drop=FALSE]) %*% U[1:(T-l), , drop=FALSE] /T
  if (l == 0){
    Omega = Omega + Gamma_l_
  } else {
    Omega = Omega + weight*(Gamma_l_ + t(Gamma_l_))
  }
}
```

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.

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

  #make BB' approximatively equal to omega
  X <- Omega - B %*% t(B)

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

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

print(cbind(Omega,B.hat %*% t(B.hat)))
```

```
##             dummy         vol
## dummy 10.24415441 0.013344869 10.24415377 0.013344867
## vol    0.01334487 0.005298555  0.01334487 0.005297573
```

```
#shock by dummy
B.hat
```

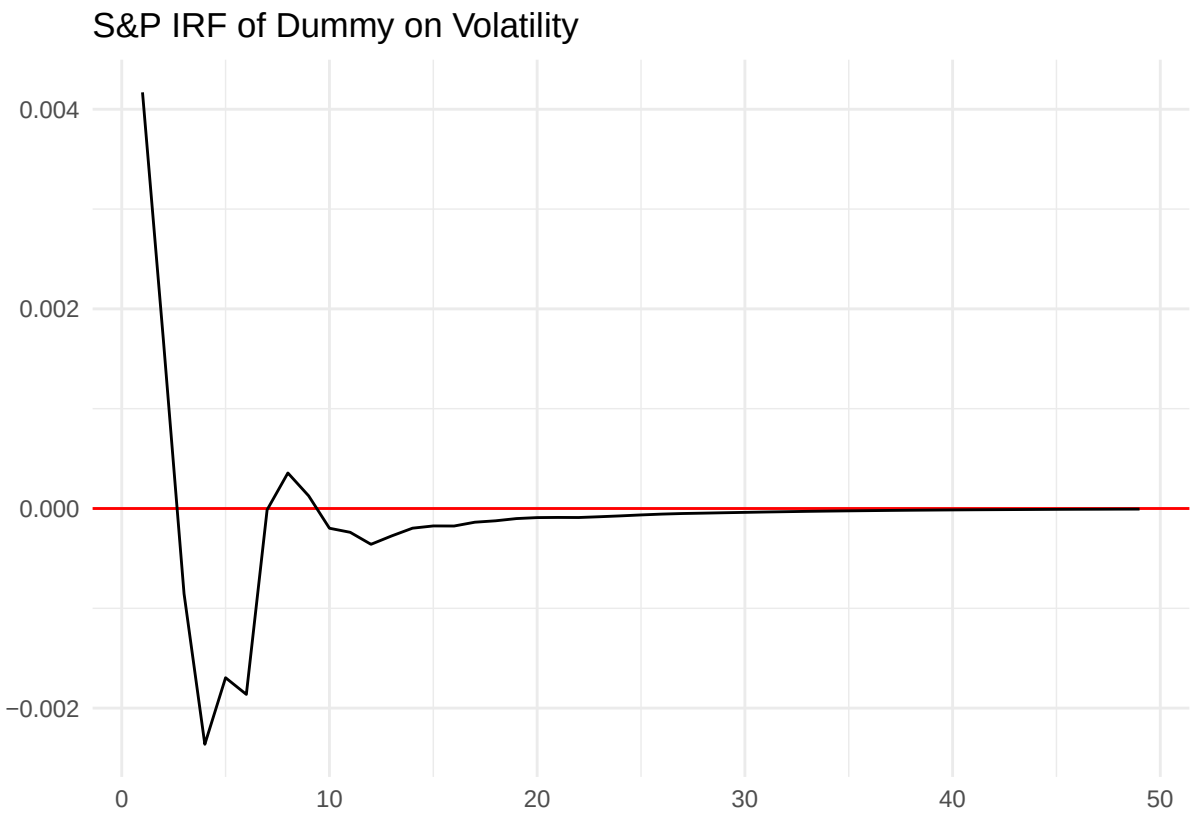
```
##             [,1]        [,2]
## [1,] 3.200648961  0.00000000
## [2,] 0.004169425 -0.07266491
```

IRF

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

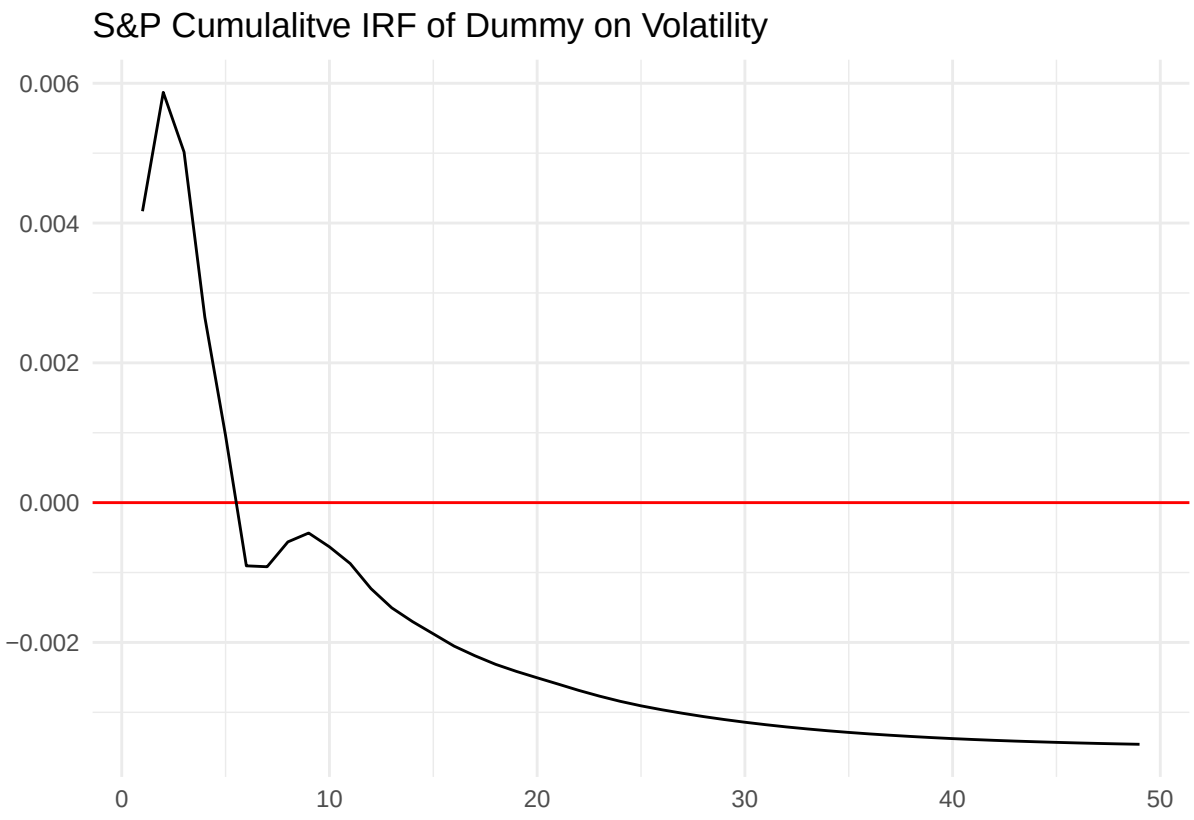
```
nb.sim = 7*7
#get back the coefficient of est.VAR
phi <- Acoef(est.VAR)
```

```r
PHI = make.PHI(phi)

#take the constant
constant <- sapply(est.VAR$varresult, function(eq) coef(eq)["const"])
c=as.matrix(constant)

#Simulate the IRF
p <- length(phi)
n <- dim(phi[[1]])[1]


Y <- simul.VAR(c=c, Phi = phi, B = B.hat, nb.sim ,y0.star=rep(0, n*p),
                  indic.IRF = 1, u.shock = c(1,0))



Yd = data.frame(
  period = 1:nrow(Y),
  response = Y[,2])

ggplot(Yd,aes(x=period, y=response)) +
  geom_hline(yintercept = 0, color="red") +
  geom_line() +
  theme_light() +
  ggtitle("S&P IRF of Dummy on Volatility")+
  ylab("")+
  xlab("") +
  theme_minimal()
```



We also graph cumulative IRF as well.

```
ggplot(Yd,aes(x=period, y=cumsum(response))) +
  geom_hline(yintercept = 0, color="red") +
  geom_line() +
  theme_light() +
  ggtitle("S&P Cumulalitve IRF of Dummy on Volatility") +
  ylab("")+
  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 dummy mentions and vice versa.

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

Res.Df	Df	F	Pr(>F)
2e+04			
2e+04	-6	4.98	4.19e-05

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

Res.Df	Df	F	Pr(>F)
2e+04			
2e+04	-6	5.99	2.83e-06