SPY SVAR Models

Contents

Setup 2

Load packages & functions . . . 2

Load Data . . . 3

Some SVAR estimations 3

Dummy variable . . . 3

Post Counts . . . 8

Trade Mention . . . 14

China Mention 19

Split Terms 25

First mandate . . . 26

Second Mandate . . . 31

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

```
#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')))
```

Some SVAR estimations

Note that this is not an exhaustive list of our VAR estimations, you can find more by going on /modeling/VAR/VAR_SPY_FULLMODELS.rmd or VAR_ASHR_FULLMODELS.rmd or VAR_VGK_FULLMODELS.rmd).

Dummy variable

Here we use a dummy variable which equal to one if Trump has made a post or 0 otherwise, taking into account the closed hour market posts.

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

#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
```

```
#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
```

```r
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
```

```r
#H0 test whether there is NOT heteroscedasticity. if less by alpha, then there is heteroscedasticity
bptest(lm_clean)
```

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

```r
#Recreate a Robust Omega Matrix
U = residuals(est.VAR)
T = nrow(U)
L = 6 #number of lag
```

```r
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_))
  }
}


#make 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)))
```

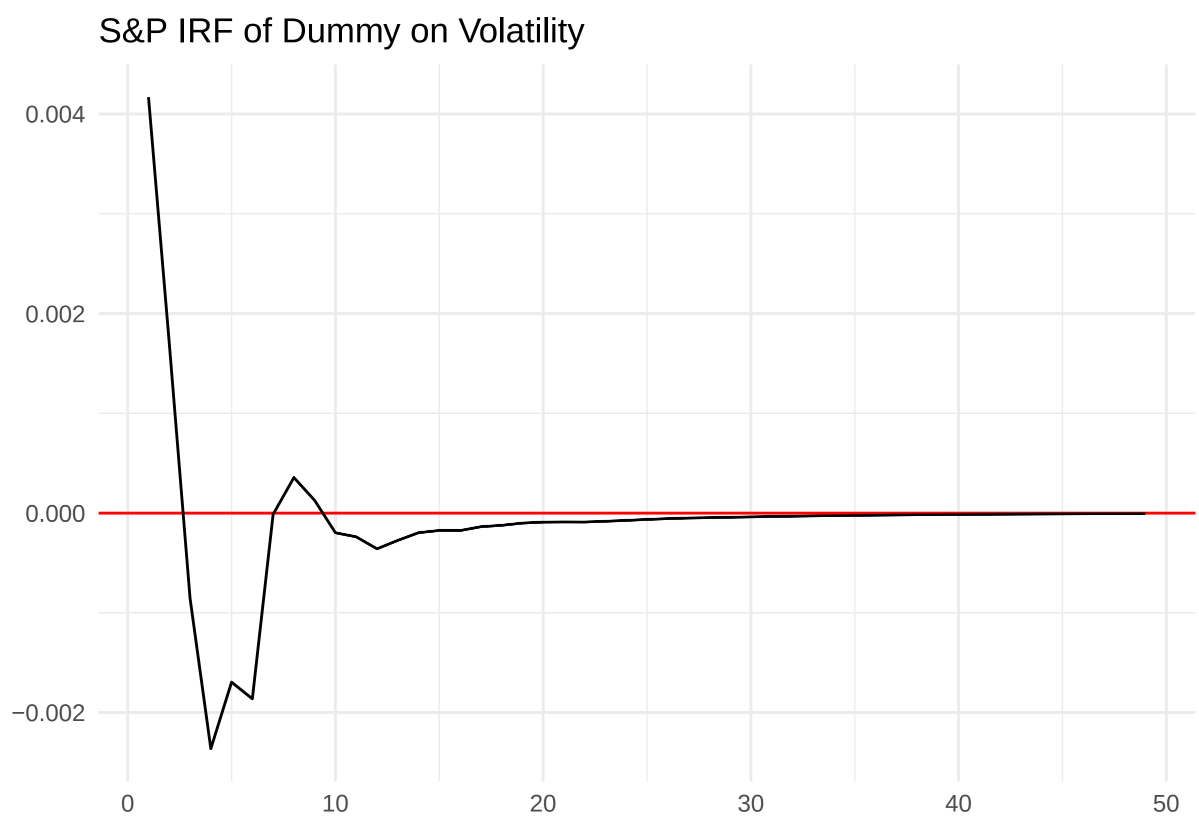
```
##             dummy         vol
## dummy 10.24415441 0.013344869 10.24415377 0.013344867
## vol    0.01334487 0.005298555  0.01334487 0.005297573
```

```r
B.hat
```

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

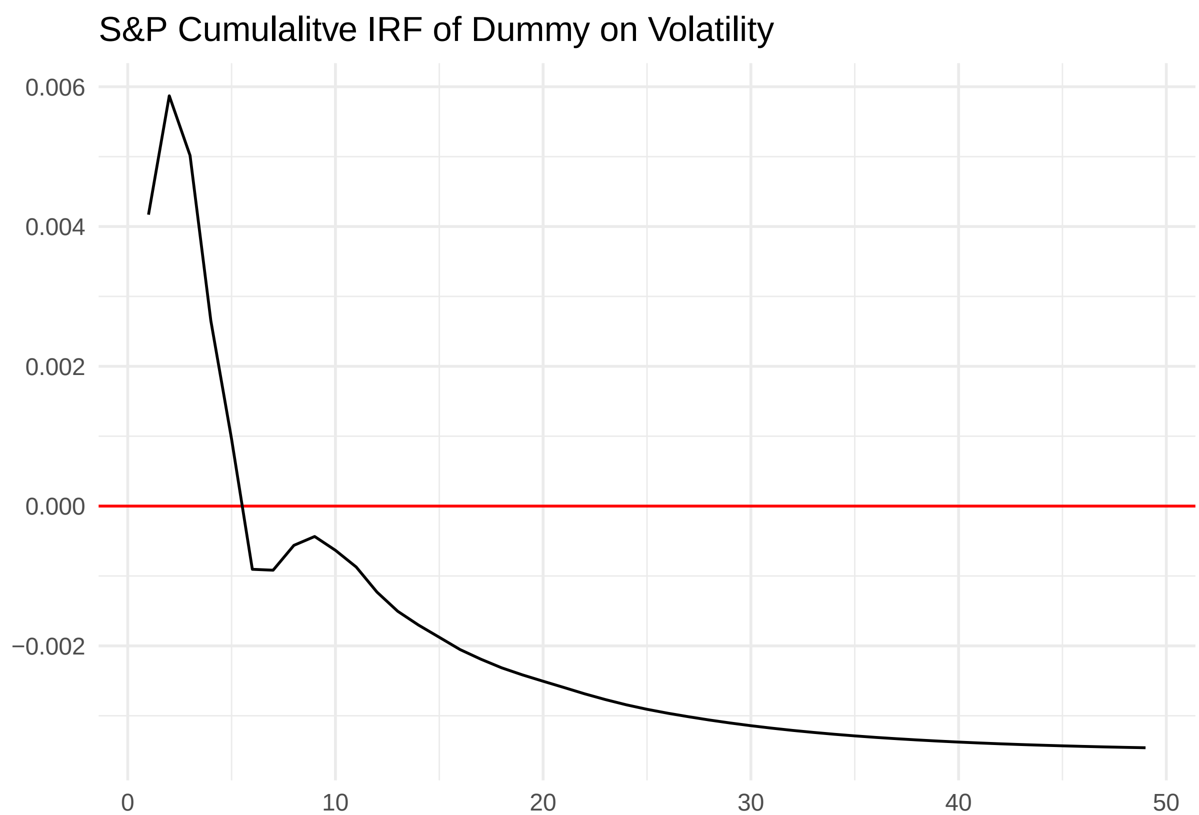
```r
nb.sim = 7*7
#get back the coefficient of est.VAR
phi <- Acoef(est.VAR)
PHI = make.PHI(phi)

#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]
```

```r
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()
```




```r
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()
```



Post Counts

```
y2 = cbind(Vdata$N, Vdata$SPY_vol)
colnames(y2)[1:2] <- c("N", "vol")
est.VAR2 <- VAR(y2,p=6)

#extract results
mod_vol2 = est.VAR2$varresult$vol
f2 = formula(mod_vol2)
d2 = model.frame(mod_vol2)
lm_clean2 = lm(f2, data= d2)

#apply Newey-West
nw_vcov2 = NeweyWest(lm_clean2, lag=6)
nw_se2 = sqrt(diag(nw_vcov2))

#t-stats
coef2 = coef(lm_clean2)
t_stat2 = coef2/nw_se2
```

```r
#recalculate p-values
robust2 = 2*(1-pt(abs(t_stat2), df = df.residual(lm_clean2)))

nw_se2       <- nw_se2[names(coef(lm_clean2))]
robust2      <- robust2[names(coef(lm_clean2))]

#table
screenreg(lm_clean2, override.se = nw_se2, override.pvalues = robust2, digits = 6)
```

```
##
## ===========================
##              Model 1
## ---------------------------
## N.l1           0.000045
##               (0.000037)
## vol.l1         0.345011 ***
##               (0.104492)
## N.l2          -0.000116 ***
##               (0.000023)
## vol.l2         0.023575
##               (0.043816)
## N.l3          -0.000213 ***
##               (0.000028)
## vol.l3         0.082525 ***
##               (0.008145)
## N.l4          -0.000147 ***
##               (0.000021)
## vol.l4         0.096739
##               (0.060827)
## N.l5          -0.000119 **
##               (0.000041)
## vol.l5         0.022593 **
##               (0.006952)
## N.l6           0.000000
##               (0.000028)
## vol.l6         0.164442 ***
##               (0.049763)
## const          0.007587 ***
##               (0.001578)
## ---------------------------
## R^2            0.325324
## Adj. R^2       0.324885
## Num. obs.  19965
## ===========================
## *** p < 0.001; ** p < 0.01; * p < 0.05
```

```r
#extract results
mod_post2 = est.VAR2$varresult$N
ff2 = formula(mod_post2)
dd2 = model.frame(mod_post2)
lm_clean_post2 = lm(ff2, data= dd2)

#apply Newey-West
```

```r
nw_vcov_post2 = NeweyWest(lm_clean_post2, lag=6)
nw_se_post2 = sqrt(diag(nw_vcov_post2))

#t-stats
coef_post2 = coef(lm_clean_post2)
t_stat_post2 = coef_post2/nw_se_post2

#recalculate p-values
robust_post2 = 2*(1-pt(abs(t_stat_post2), df = df.residual(lm_clean_post2)))

nw_se_post2       <- nw_se_post2[names(coef(lm_clean_post2))]
robust_post2      <- robust_post2[names(coef(lm_clean_post2))]

#table
screenreg(lm_clean_post2, override.se = nw_se_post2, override.pvalues = robust_post2, digits = 6)
```

```
##
## ===========================
##            Model 1
## ---------------------------
## N.l1          -0.043917 ***
##               (0.003632)
## vol.l1         0.898680
##               (0.753418)
## N.l2          -0.039092 ***
##               (0.004198)
## vol.l2        -0.990623
##               (0.742365)
## N.l3          -0.025847 ***
##               (0.004288)
## vol.l3        -1.241543
##               (0.872900)
## N.l4          -0.023845 ***
##               (0.004990)
## vol.l4        -1.830204 *
##               (0.745455)
## N.l5          -0.040694 ***
##               (0.003975)
## vol.l5        -0.683162
##               (0.469187)
## N.l6          -0.045066 ***
##               (0.004790)
## vol.l6         2.727809
##               (1.511411)
## const          3.531650 ***
##               (0.092709)
## ---------------------------
## R^2            0.100807
## Adj. R^2       0.100221
## Num. obs.  19965
## ===========================
## *** p < 0.001; ** p < 0.01; * p < 0.05
```

```r
#Recreate a Robust Omega Matrix
U2 = residuals(est.VAR2)
T2 = nrow(U2)
Omega2 = matrix(0, ncol(U2), ncol(U2))
for(l in 0:L) {
  weight = 1 - l/(L+1)
  Gamma_l_2 = t(U2[(l+1):T2, , drop=FALSE]) %*% U2[1:(T2-l), , drop=FALSE] /T2
  if (l == 0){
    Omega2 = Omega2 + Gamma_l_2
  } else {
    Omega2 = Omega2 + weight*(Gamma_l_2 + t(Gamma_l_2))
  }
}

#make the B matrix
loss2 <- function(param2){
  #Define the restriction
  B2 <- matrix(c(param2[1], param2[2], 0, param2[3]), ncol = 2)

  #Make BB' approximatively equal to omega
  X2 <- Omega2 - B2 %*% t(B2)

  #loss function
  loss2 <- sum(X2^2)
  return(loss2)
}

res.opt2 <- optim(c(1, 0, 1), loss2, method = "BFGS")
B.hat2 <- matrix(c(res.opt2$par[1], res.opt2$par[2], 0, res.opt2$par[3]), ncol = 2)

print(cbind(Omega2,B.hat2 %*% t(B.hat2)))
```

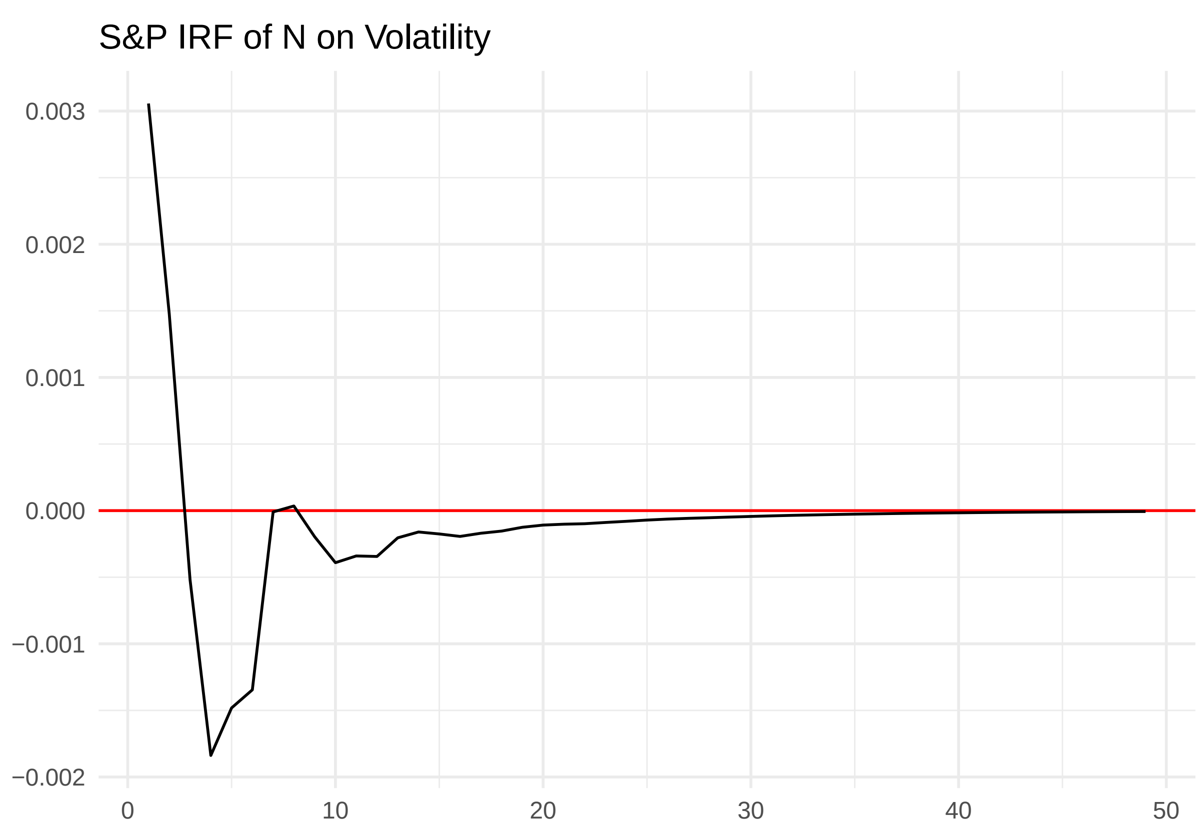
```
##               N         vol
## N   86.4367215 0.028419903 86.43672134 0.028419567
## vol  0.0284199 0.005294799  0.02841957 0.005293894
```

```r
B.hat2
```

```
##              [,1]       [,2]
## [1,] 9.297135115 0.00000000
## [2,] 0.003056809 0.07269491
```

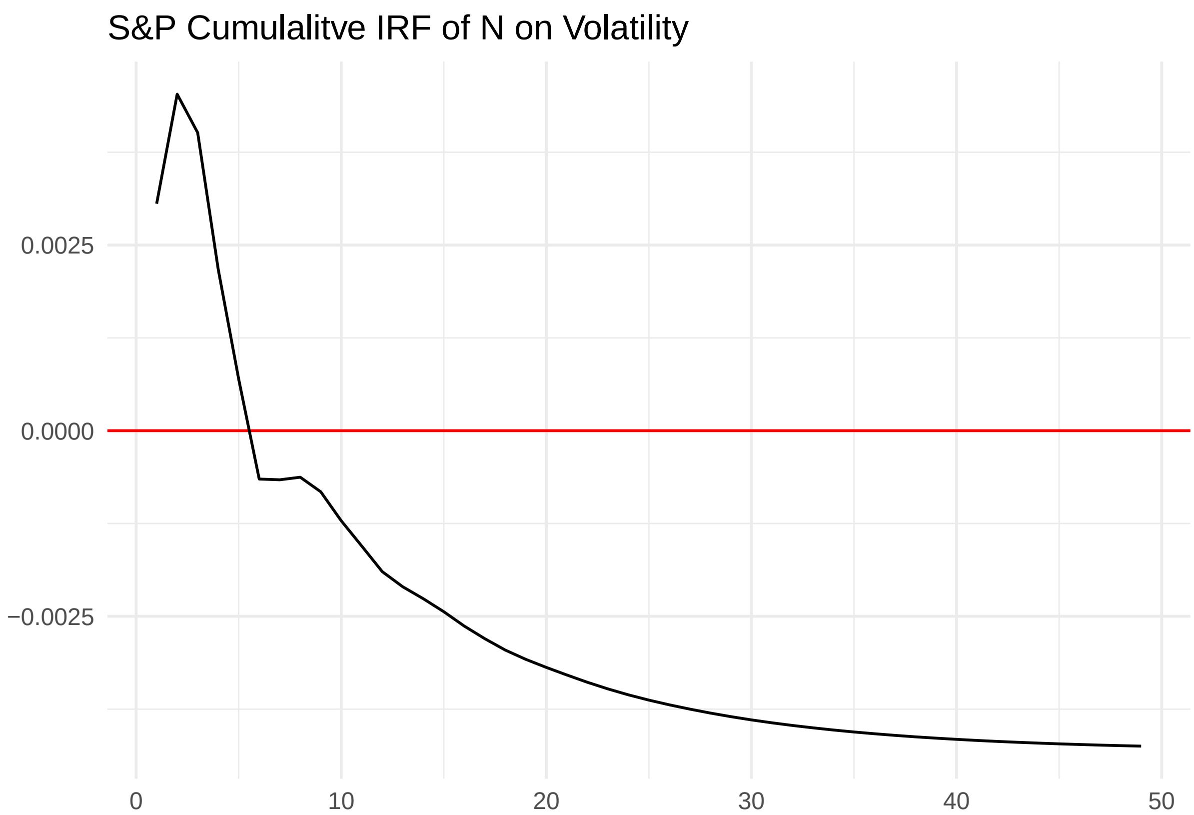
```r
#get back the coefficient of est.VAR
phi2 <- Acoef(est.VAR2)
PHI2 = make.PHI(phi2)

#take the constant
constant2 <- sapply(est.VAR2$varresult, function(eq) coef(eq)["const"])
c2=as.matrix(constant2)

#Simulate the IRF
p2 <- length(phi2)
```

```
n2 <- dim(phi2[[1]])[1]

Y2 <- simul.VAR(c=c2, Phi = phi2, B = B.hat2, nb.sim ,y0.star=rep(0, n2*p2),
                  indic.IRF = 1, u.shock = c(1,0))


#Plot the IRF
Yd2 = data.frame(
  period = 1:nrow(Y2),
  response = Y2[,2])

ggplot(Yd2,aes(x=period, y=response)) +
  geom_hline(yintercept = 0, color="red") +
  geom_line() +
  theme_light() +
  ggtitle("S&P IRF of N on Volatility") +
  ylab("")+
  xlab("") +
  theme_minimal()
```




```
ggplot(Yd2,aes(x=period, y=cumsum(response))) +
  geom_hline(yintercept = 0, color="red") +
  geom_line() +
  theme_light() +
  ggtitle("S&P Cumulalitve IRF of N on Volatility") +
```

```r
  ylab("")+
  xlab("") +
  theme_minimal()
```




```r
grangertest(y2[,c("N", "vol")], order = 6)
```

Res.Df	Df	F	Pr(>F)
2e+04			
2e+04	-6	3.92	0.000646

```r
grangertest(y2[,c("vol", "N")], order = 6)
```

Res.Df	Df	F	Pr(>F)
2e+04			
2e+04	-6	3.89	0.000688

Trade Mention

```
y4 = cbind(Vdata$trade, Vdata$SPY_vol)
colnames(y4)[1:2] <- c("trade", "vol")
est.VAR4 <- VAR(y4,p=6)

#extract results
mod_vol4 = est.VAR4$varresult$vol
f4 = formula(mod_vol4)
d4 = model.frame(mod_vol4)
lm_clean4 = lm(f4, data= d4)

#apply Newey-West
nw_vcov4 = NeweyWest(lm_clean4, lag=6)
nw_se4 = sqrt(diag(nw_vcov4))

#t-stats
coef4 = coef(lm_clean4)
t_stat4 = coef4/nw_se4

#recalculate p-values
robust4 = 2*(1-pt(abs(t_stat4), df = df.residual(lm_clean4)))

nw_se4       <- nw_se4[names(coef(lm_clean4))]
robust4      <- robust4[names(coef(lm_clean4))]

#table
screenreg(lm_clean4, override.se = nw_se4, override.pvalues = robust4, digits = 6)
```

```
##
## ===========================
##            Model 1
## ---------------------------
## trade.l1       0.003399
##               (0.003747)
## vol.l1         0.346107 ***
##               (0.101918)
## trade.l2       0.005600
##               (0.004809)
## vol.l2         0.022949
##               (0.041538)
## trade.l3      -0.003904 *
##               (0.001726)
## vol.l3         0.081148 ***
##               (0.008258)
## trade.l4       0.000725
##               (0.003458)
## vol.l4         0.095797
##               (0.057082)
## trade.l5      -0.002363
##               (0.001901)
## vol.l5         0.023502 **
##               (0.007162)
```

```
## trade.l6       -0.001543
##                (0.001228)
## vol.l6          0.165323 ***
##                (0.049319)
## const           0.005939 ***
##                (0.001536)
## ---------------------------
## R^2             0.325134
## Adj. R^2        0.324695
## Num. obs.   19965
## ===========================
## *** p < 0.001; ** p < 0.01; * p < 0.05
```

```r
#Table for the effect of volatility on posts for variable trade
#extract results
mod_post4 = est.VAR4$varresult$trade
ff4 = formula(mod_post4)
dd4 = model.frame(mod_post4)
lm_clean_post4 = lm(ff4, data= dd4)

#apply Newey-West
nw_vcov_post4 = NeweyWest(lm_clean_post4, lag=6)
nw_se_post4 = sqrt(diag(nw_vcov_post4))

#t-stats
coef_post4 = coef(lm_clean_post4)
t_stat_post4 = coef_post4/nw_se_post4

#recalculate p-values
robust_post4 = 2*(1-pt(abs(t_stat_post4), df = df.residual(lm_clean_post4)))

nw_se_post4       <- nw_se_post4[names(coef(lm_clean_post4))]
robust_post4      <- robust_post4[names(coef(lm_clean_post4))]

#table
screenreg(lm_clean_post4, override.se = nw_se_post4, override.pvalues = robust_post4, digits = 6)
```

```
##
## ===========================
##            Model 1
## ---------------------------
## trade.l1        0.025465
##                (0.018216)
## vol.l1          0.023385
##                (0.038889)
## trade.l2        0.019892 *
##                (0.008655)
## vol.l2          0.140914 *
##                (0.057926)
## trade.l3        0.022679
##                (0.016472)
## vol.l3         -0.077822 ***
##                (0.009836)
## trade.l4        0.021687
```

```
##                (0.012853)
## vol.l4         -0.037637 **
##                (0.013028)
## trade.l5        0.019364
##                (0.010830)
## vol.l5         -0.019644
##                (0.020558)
## trade.l6        0.012565
##                (0.010365)
## vol.l6          0.107182
##                (0.070029)
## const           0.027409 ***
##                (0.002344)
## ---------------------------
## R^2             0.020264
## Adj. R^2        0.019626
## Num. obs.   19965
## ===========================
## *** p < 0.001; ** p < 0.01; * p < 0.05
```

```
#Recreate a Robust Omega Matrix
U4 = residuals(est.VAR4)
T4 = nrow(U4)
Omega4 = matrix(0, ncol(U4), ncol(U4))
for(l in 0:L) {
  weight = 1 - l/(L+1)
  Gamma_l_4 = t(U4[(l+1):T4, , drop=FALSE]) %*% U4[1:(T4-l), , drop=FALSE] /T4
  if (l == 0){
    Omega4 = Omega4 + Gamma_l_4
  } else {
    Omega4 = Omega4 + weight*(Gamma_l_4 + t(Gamma_l_4))
  }
}


#make the B matrix
loss4 <- function(param4){
  #Define the restriction
  B4 <- matrix(c(param4[1], param4[2], 0, param4[3]), ncol = 2)

  #Make BB' approximatively equal to omega
  X4 <- Omega4 - B4 %*% t(B4)

  #loss function
  loss4 <- sum(X4^2)
  return(loss4)
}

res.opt4 <- optim(c(1, 0, 1), loss4, method = "BFGS")
B.hat4 <- matrix(c(res.opt4$par[1], res.opt4$par[2], 0, res.opt4$par[3]), ncol = 2)

print(cbind(Omega4,B.hat4 %*% t(B.hat4)))
```

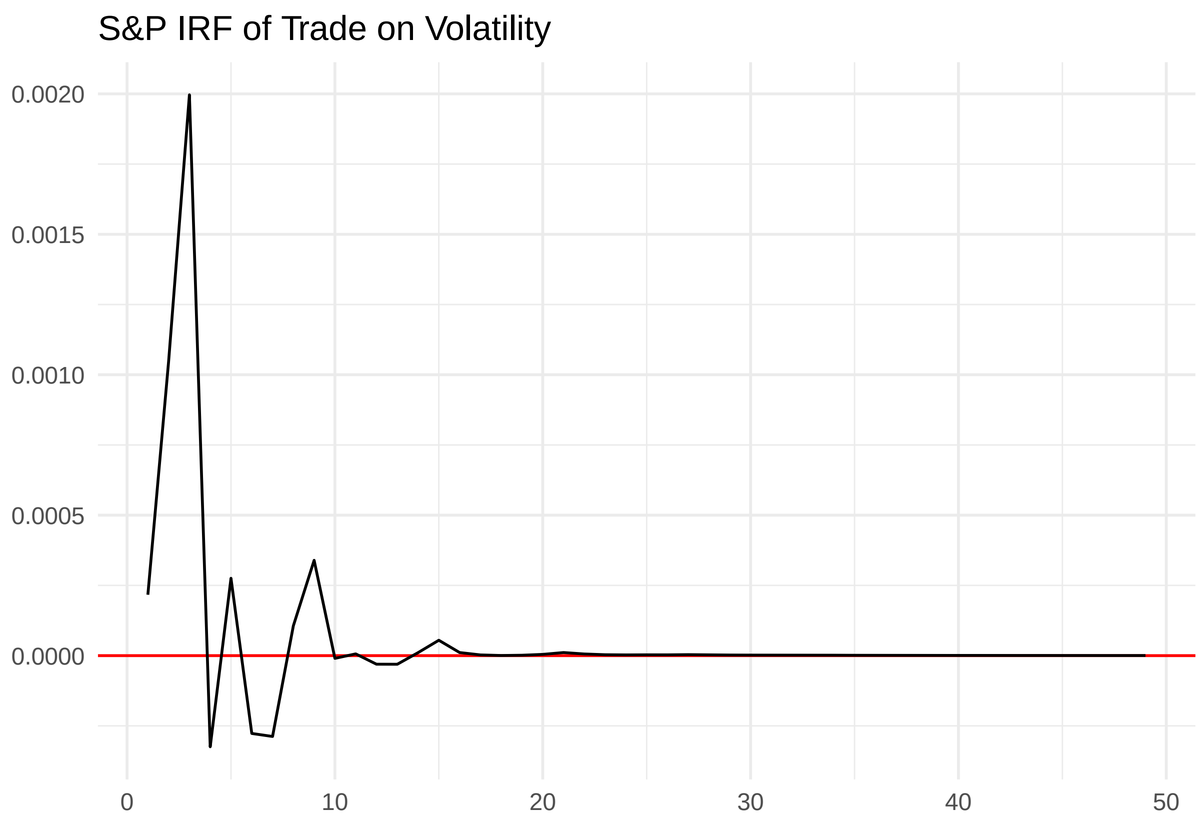
```
##               trade          vol
```

```
## trade 8.203633e-02 6.209714e-05 8.203534e-02 6.210009e-05
## vol   6.209714e-05 5.281703e-03 6.210009e-05 5.280741e-03
```

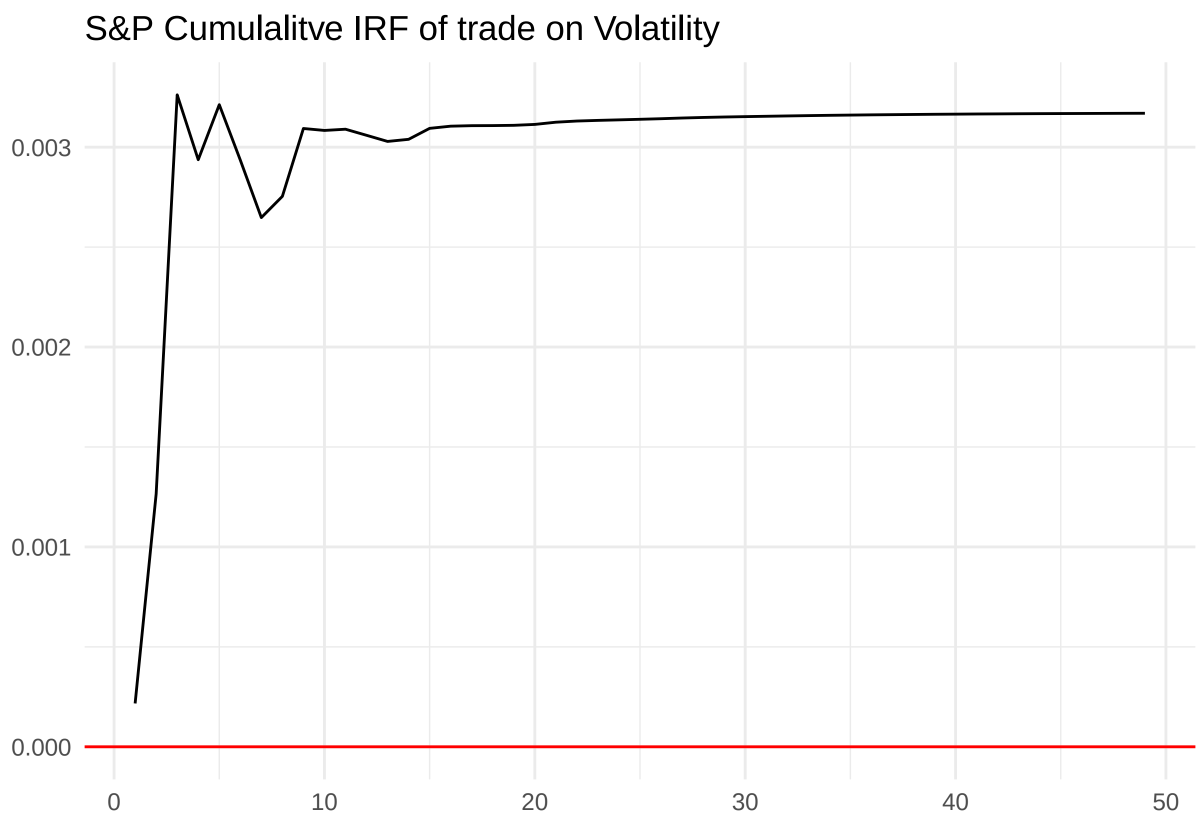
```
B.hat4
```

```
##              [,1]       [,2]
## [1,] 0.2864181131 0.00000000
## [2,] 0.0002168162 0.07266838
```

```
#get back the coefficient of est.VAR
phi4 <- Acoef(est.VAR4)
PHI4 = make.PHI(phi4)

#take the constant
constant4 <- sapply(est.VAR4$varresult, function(eq) coef(eq)["const"])
c4=as.matrix(constant4)

#Simulate the IRF
p4 <- length(phi4)
n4 <- dim(phi4[[1]])[1]

Y4 <- simul.VAR(c=c4, Phi = phi4, B = B.hat4, nb.sim ,y0.star=rep(0, n4*p4),
                  indic.IRF = 1, u.shock = c(1,0))


#Plot the IRF
Yd4 = data.frame(
  period = 1:nrow(Y4),
  response = Y4[,2])

ggplot(Yd4,aes(x=period, y=response)) +
  geom_hline(yintercept = 0, color="red") +
  geom_line() +
  theme_light() +
  ggtitle("S&P IRF of Trade on Volatility") +
  ylab("")+
  xlab("") +
  theme_minimal()
```




```
ggplot(Yd4,aes(x=period, y=cumsum(response))) +
  geom_hline(yintercept = 0, color="red") +
  geom_line() +
  theme_light() +
  ggtitle("S&P Cumulalitve IRF of trade on Volatility") +
  ylab("")+
  xlab("") +
  theme_minimal()
```




```r
grangertest(y4[,c("vol","trade")], order = 6)
```

Res.Df	Df	F	Pr(>F)
2e+04			
2e+04	-6	10.6	9.06e-12

```r
grangertest(y4[,c("trade", "vol")], order = 6)
```

Res.Df	Df	F	Pr(>F)
2e+04			
2e+04	-6	2.98	0.00655

China Mention

```r
ychina = cbind(Vdata$china, Vdata$SPY_vol)
colnames(ychina)[1:2] <- c("china", "vol")
est.VARchina <- VAR(ychina,p=6)

#extract results
mod_volchina = est.VARchina$varresult$vol
fchina = formula(mod_volchina)
dchina = model.frame(mod_volchina)
lm_cleanchina = lm(fchina, data= dchina)

#apply Newey-West
nw_vcovchina = NeweyWest(lm_cleanchina, lag=6)
nw_sechina = sqrt(diag(nw_vcovchina))

#t-stats
coefchina = coef(lm_cleanchina)
t_statchina = coefchina/nw_sechina

#recalculate p-values
robustchina = 2*(1-pt(abs(t_statchina), df = df.residual(lm_cleanchina)))

nw_sechina       <- nw_sechina[names(coef(lm_cleanchina))]
robustchina      <- robustchina[names(coef(lm_cleanchina))]

#table
screenreg(lm_cleanchina, override.se = nw_sechina, override.pvalues = robustchina, digits = 6)
```

```
##
## ===========================
##            Model 1
## ---------------------------
## china.l1       0.006729
##               (0.006694)
## vol.l1         0.344512 ***
##               (0.097994)
## china.l2       0.002778
##               (0.004067)
## vol.l2         0.024149
##               (0.043585)
## china.l3      -0.004652 *
##               (0.002066)
## vol.l3         0.081646 ***
##               (0.009192)
## china.l4      -0.002442 *
##               (0.001084)
## vol.l4         0.094919
##               (0.058821)
## china.l5      -0.000607
##               (0.000970)
## vol.l5         0.022961 **
##               (0.007678)
## china.l6       0.000596
##               (0.000981)
```

```
## vol.l6         0.166695 **
##               (0.054194)
## const          0.005857 ***
##               (0.001612)
## ---------------------------
## R^2            0.326344
## Adj. R^2       0.325905
## Num. obs.  19965
## ===========================
## *** p < 0.001; ** p < 0.01; * p < 0.05
```

```r
#Table for the effect of volatility on posts for variable china
#extract results
mod_postchina = est.VARchina$varresult$china
ffchina = formula(mod_postchina)
ddchina = model.frame(mod_postchina)
lm_clean_postchina = lm(ffchina, data= ddchina)

#apply Newey-West
nw_vcov_postchina = NeweyWest(lm_clean_postchina, lag=6)
nw_se_postchina = sqrt(diag(nw_vcov_postchina))

#t-stats
coef_postchina = coef(lm_clean_postchina)
t_stat_postchina = coef_postchina/nw_se_postchina

#recalculate p-values
robust_postchina = 2*(1-pt(abs(t_stat_postchina), df = df.residual(lm_clean_postchina)))

nw_se_postchina       <- nw_se_postchina[names(coef(lm_clean_postchina))]
robust_postchina      <- robust_postchina[names(coef(lm_clean_postchina))]

#table
screenreg(lm_clean_postchina, override.se = nw_se_postchina, override.pvalues = robust_postchina, digits
```

```
##
## ===========================
##            Model 1
## ---------------------------
## china.l1       0.074471 *
##               (0.034456)
## vol.l1         0.028311
##               (0.039275)
## china.l2       0.044301 *
##               (0.021635)
## vol.l2         0.138351
##               (0.165247)
## china.l3       0.005765
##               (0.010638)
## vol.l3         0.140252
##               (0.202759)
## china.l4       0.024636 **
##               (0.009414)
## vol.l4        -0.108103
```

```
##                (0.064202)
## china.l5        0.048992
##                (0.034173)
## vol.l5         -0.061820 **
##                (0.022604)
## china.l6        0.055998
##                (0.048651)
## vol.l6          0.044227
##                (0.048698)
## const           0.041625 ***
##                (0.004912)
## ---------------------------
## R^2             0.036628
## Adj. R^2        0.036000
## Num. obs.   19965
## ===========================
## *** p < 0.001; ** p < 0.01; * p < 0.05
```

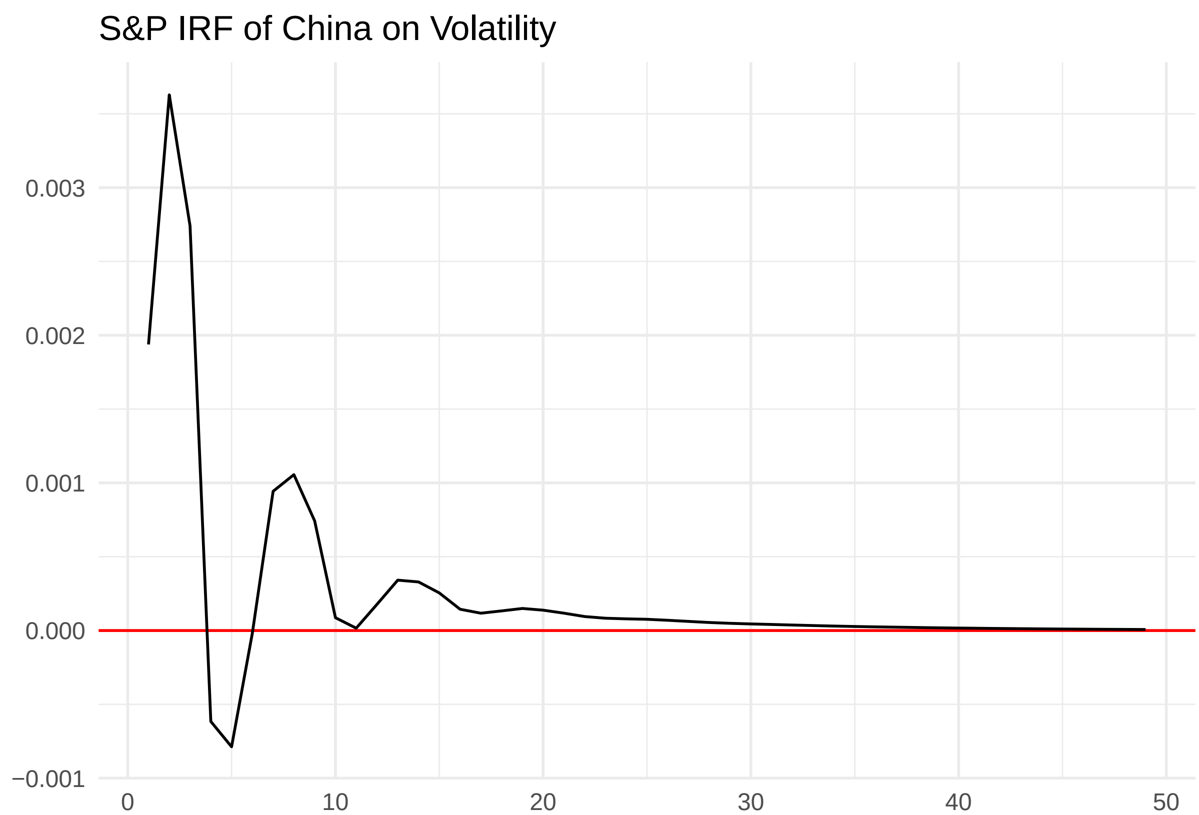
```
#Recreate a Robust Omega Matrix
Uchina = residuals(est.VARchina)
Tchina = nrow(Uchina)
Omegachina = matrix(0, ncol(Uchina), ncol(Uchina))
for(l in 0:L) {
  weight = 1 - l/(L+1)
  Gamma_l_china = t(Uchina[(l+1):Tchina, , drop=FALSE]) %*% Uchina[1:(Tchina-l), , drop=FALSE] /Tchina
  if (l == 0){
    Omegachina = Omegachina + Gamma_l_china
  } else {
    Omegachina = Omegachina + weight*(Gamma_l_china + t(Gamma_l_china))
  }
}


#make the B matrix
losschina <- function(paramchina){
  #Define the restriction
  Bchina <- matrix(c(paramchina[1], paramchina[2], 0, paramchina[3]), ncol = 2)

  #Make BB' approximatively equal to omega
  Xchina <- Omegachina - Bchina %*% t(Bchina)

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

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

print(cbind(Omegachina,B.hatchina %*% t(B.hatchina)))
```

```
##              china          vol
## china 0.1936352468 0.0008527492 0.1936342473 0.0008527453
## vol   0.0008527492 0.0052690830 0.0008527453 0.0052683426
```

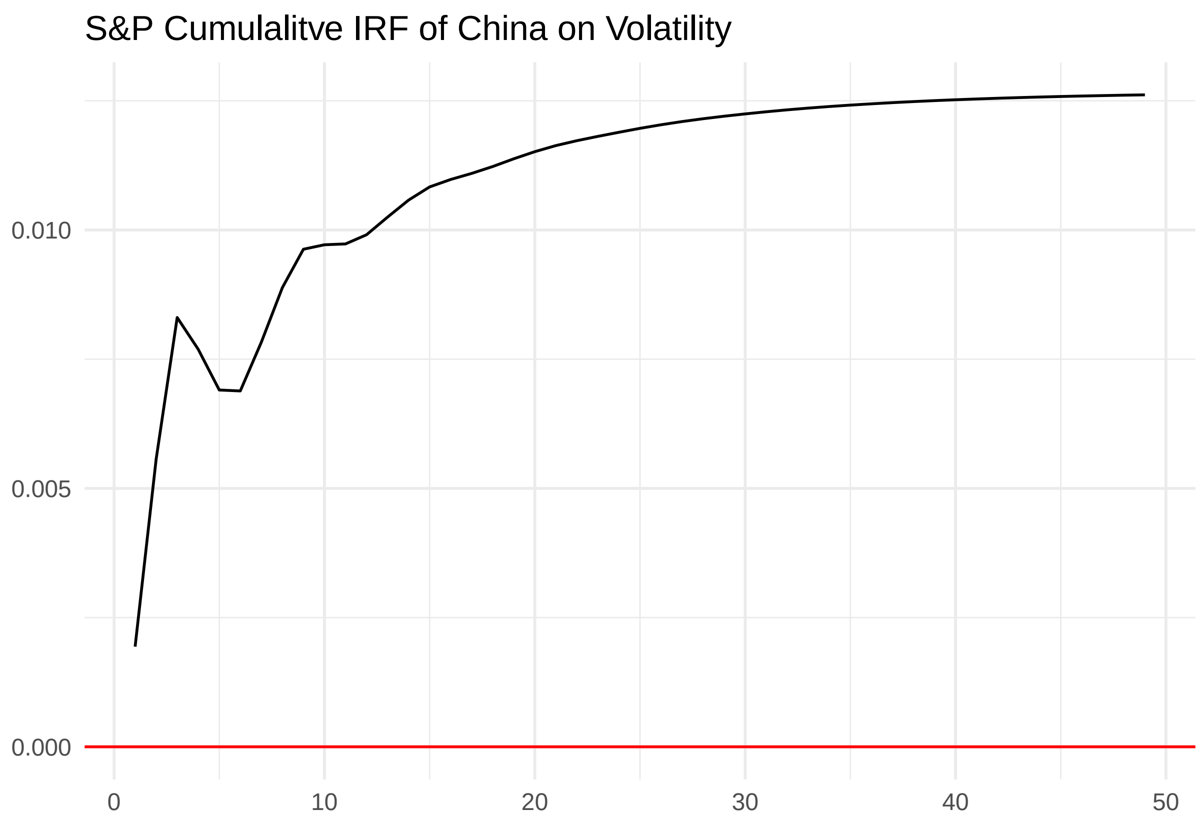
```r
B.hatchina
```

```
##              [,1]       [,2]
## [1,] 0.440038916 0.00000000
## [2,] 0.001937886 0.07255748
```

```r
#get back the coefficient of est.VAR
phichina <- Acoef(est.VARchina)
PHIchina = make.PHI(phichina)

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

#Simulate the IRF
pchina <- length(phichina)
nchina <- dim(phichina[[1]])[1]

Ychina <- simul.VAR(c=cchina, Phi = phichina, B = B.hatchina, nb.sim ,y0.star=rep(0, nchina*pchina),
                  indic.IRF = 1, u.shock = c(1,0))


#Plot the IRF
Ydchina = data.frame(
  period = 1:nrow(Ychina),
  response = Ychina[,2])

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




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




```r
grangertest(ychina[,c("vol", "china")], order = 6)
```

Res.Df	Df	F	Pr(>F)
2e+04			
2e+04	-6	7.43	5.74e-08

```r
grangertest(ychina[,c("china", "vol")], order = 6)
```

Res.Df	Df	F	Pr(>F)
2e+04			
2e+04	-6	8.96	8.55e-10

Split Terms

Here we look for the first and second mandate effect of posts. We will use the tariff variable as a proxy for the posts.

```r
# First and Second Mandate

#first term
Vdata_f = filter(data,between(timestamp, as.Date('2017-01-20'), as.Date('2021-01-20')))

#second term
Vdata_s = filter(data,between(timestamp, as.Date('2025-01-20'), as.Date('2025-05-07')))
```

First mandate

```r
y_f_d = cbind(Vdata_f$dummy, Vdata_f$SPY_vol)
colnames(y_f_d)[1:2] <- c("dummy", "vol")
est.VAR_f_d <- VAR(y_f_d,p=6)

#extract results
mod_vol_f_d = est.VAR_f_d$varresult$vol
f_f_d = formula(mod_vol_f_d)
d_f_d = model.frame(mod_vol_f_d)
lm_clean_f_d = lm(f_f_d, data= d_f_d)

#apply Newey-West
nw_vcov_f_d = NeweyWest(lm_clean_f_d, lag=6)
nw_se_f_d = sqrt(diag(nw_vcov_f_d))

#t-stats
coef_f_d = coef(lm_clean_f_d)
t_stat_f_d = coef_f_d/nw_se_f_d

#recalculate p-values
robust_f_d = 2*(1-pt(abs(t_stat_f_d), df = df.residual(lm_clean_f_d)))

nw_se_f_d       <- nw_se_f_d[names(coef(lm_clean_f_d))]
robust_f_d      <- robust_f_d[names(coef(lm_clean_f_d))]

#table
screenreg(lm_clean_f_d, override.se = nw_se_f_d, override.pvalues = robust_f_d, digits = 6)
```

========================== Model 1
———————————— dummy.l1 -0.000478 ***(0.000133)***
vol.l1 0.541944 (0.074062)
dummy.l2 -0.000184 ** (0.000069)
vol.l2 -0.113920 ** (0.038762)
dummy.l3 -0.000693 ***(0.000160)***
vol.l3 0.058050
(0.027504)
dummy.l4 -0.000564 *(0.000166)*
vol.l4 0.188383
(0.132562)
dummy.l5 -0.000435 **(0.000113)**
vol.l5 -0.088758
(0.091453)

dummy.l6 0.000118
(0.000118)
vol.l6 0.336662 *(0.049019)*
const 0.004020 **(0.000669)**
——————————— **R^2 0.687909**
Adj. R^2 0.687331
Num. obs. 7036
========================== * p < 0.001; ** p < 0.01; * p < 0.05

```r
#Table for the effect of volatility on posts for variable dummy
#extract results
mod_post_f_d = est.VAR_f_d$varresult$dummy
ff_f_d = formula(mod_post_f_d)
dd_f_d = model.frame(mod_post_f_d)
lm_clean_post_f_d = lm(ff_f_d, data= dd_f_d)

#apply Newey-West
nw_vcov_post_f_d = NeweyWest(lm_clean_post_f_d, lag=6)
nw_se_post_f_d = sqrt(diag(nw_vcov_post_f_d))

#t-stats
coef_post_f_d = coef(lm_clean_post_f_d)
t_stat_post_f_d = coef_post_f_d/nw_se_post_f_d

#recalculate p-values
robust_post_f_d = 2*(1-pt(abs(t_stat_post_f_d), df = df.residual(lm_clean_post_f_d)))

nw_se_post_f_d       <- nw_se_post_f_d[names(coef(lm_clean_post_f_d))]
robust_post_f_d      <- robust_post_f_d[names(coef(lm_clean_post_f_d))]


#table
screenreg(lm_clean_post_f_d, override.se = nw_se_post_f_d, override.pvalues = robust_post_f_d, digits =
```

```
##
## ==========================
##            Model 1
## --------------------------
## dummy.l1     -0.136890 ***
##              (0.006701)
## vol.l1        6.918718 **
##              (2.346819)
## dummy.l2     -0.105079 ***
##              (0.005805)
## vol.l2       -4.399079 **
##              (1.426496)
## dummy.l3     -0.089466 ***
##              (0.006149)
## vol.l3       -1.329415
##              (0.681253)
## dummy.l4     -0.087428 ***
##              (0.006320)
## vol.l4       -2.004975 **
##              (0.762777)
```

```
## dummy.l5     -0.110469 ***
##              (0.006633)
## vol.l5       -0.823577
##              (0.894915)
## dummy.l6     -0.135443 ***
##              (0.007492)
## vol.l6        3.883219 **
##              (1.365831)
## const         1.941260 ***
##              (0.060551)
## --------------------------
## R^2           0.190525
## Adj. R^2      0.189027
## Num. obs.  7036
## ==========================
## *** p < 0.001; ** p < 0.01; * p < 0.05
```

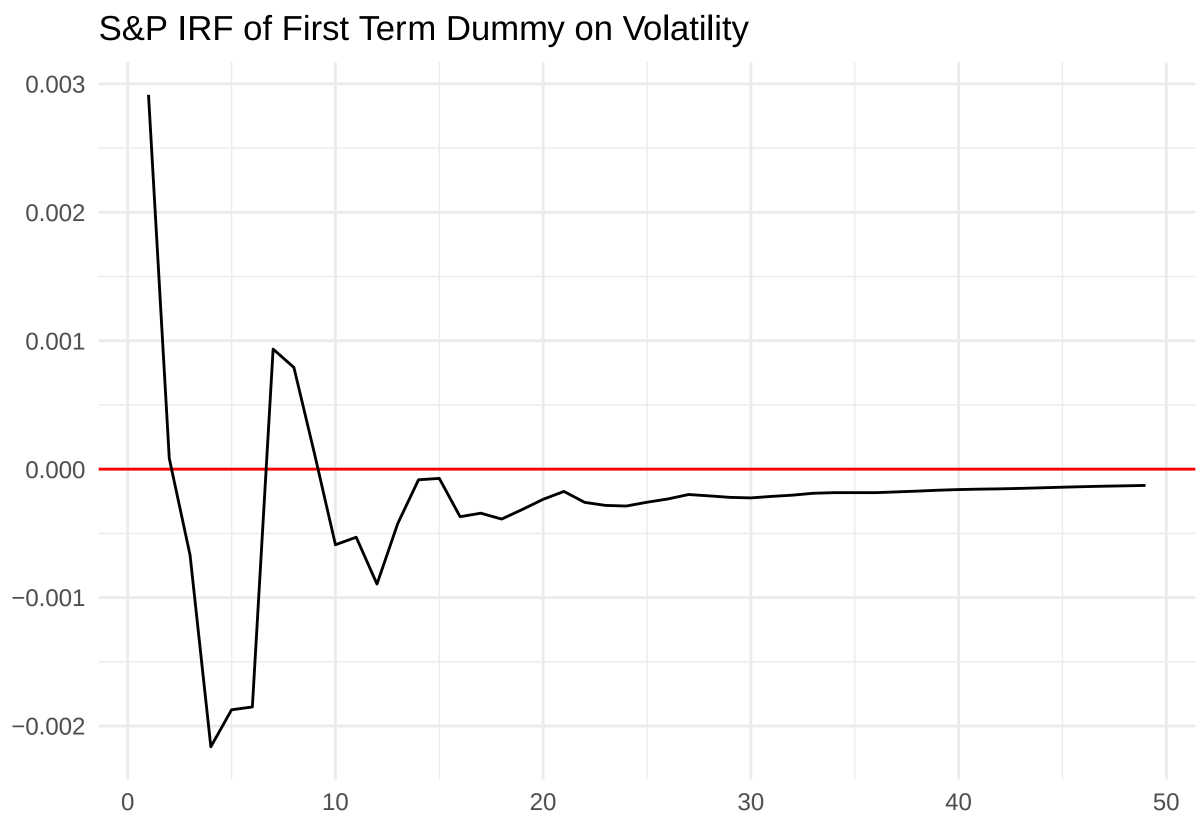
```r
#Construct the Robust Omega Matrix
U_f_d = residuals(est.VAR_f_d)
T_f_d = nrow(U_f_d)
Omega_f_d = matrix(0, ncol(U_f_d), ncol(U_f_d))
for(l in 0:L) {
  weight = 1 - l/(L+1)
  Gamma_l__f_d = t(U_f_d[(l+1):T_f_d, , drop=FALSE]) %*% U_f_d[1:(T_f_d-l), , drop=FALSE] /T_f_d
  if (l == 0){
    Omega_f_d = Omega_f_d + Gamma_l__f_d
  } else {
    Omega_f_d = Omega_f_d + weight*(Gamma_l__f_d + t(Gamma_l__f_d))
  }
}


#make the B matrix
loss_f_d <- function(param_f_d){
  #Define the restriction
  B_f_d <- matrix(c(param_f_d[1], param_f_d[2], 0, param_f_d[3]), ncol = 2)

  #Make BB' approximatively equal to omega
  X_f_d <- Omega_f_d - B_f_d %*% t(B_f_d)

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

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

print(cbind(Omega_f_d,B.hat_f_d %*% t(B.hat_f_d)))
```

```
##            dummy          vol
## dummy 9.80314631 0.0091248204 9.803145998 0.0091245935
## vol   0.00912482 0.0007597912 0.009124594 0.0007587832
```

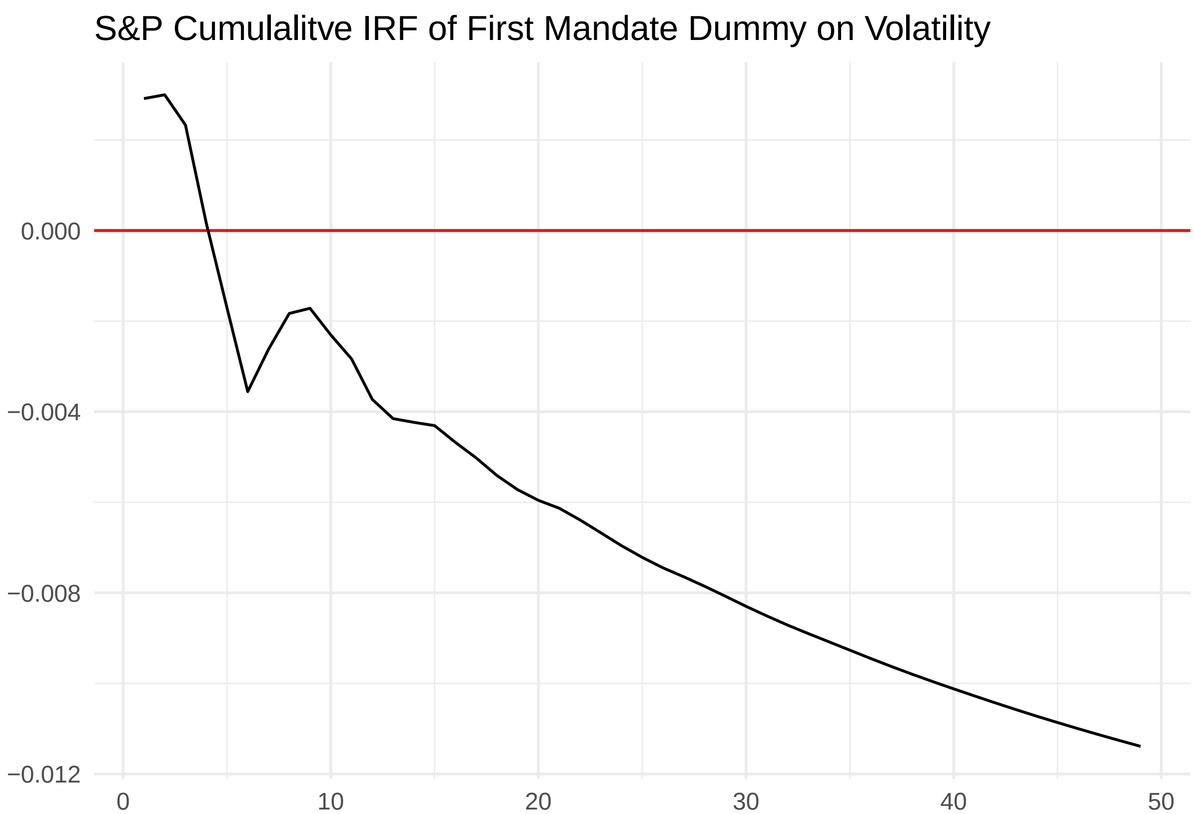
```r
B.hat_f_d
```

```
##             [,1]       [,2]
## [1,] 3.130997604 0.00000000
## [2,] 0.002914277 0.02739143
```

```r
#get back the coefficient of est.VAR
phi_f_d <- Acoef(est.VAR_f_d)
PHI_f_d = make.PHI(phi_f_d)

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

#Simulate the IRF
p_f_d <- length(phi_f_d)
n_f_d <- dim(phi_f_d[[1]])[1]

Y_f_d <- simul.VAR(c=c_f_d, Phi = phi_f_d, B = B.hat_f_d, nb.sim ,y0.star=rep(0, n_f_d*p_f_d),
                  indic.IRF = 1, u.shock = c(1,0))


#Plot the IRF
Yd_f_d = data.frame(
  period = 1:nrow(Y_f_d),
  response = Y_f_d[,2])

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




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




```r
#does vol granger cause dummy
grangertest(y_f_d[,c("vol","dummy")], order =6)
```

Res.Df	Df	F	Pr(>F)
7.02e+03			
7.03e+03	-6	13	9.77e-15

```r
#does dummy granger cause vol
grangertest(y_f_d[,c("dummy", "vol")], order =6)
```

Res.Df	Df	F	Pr(>F)
7.02e+03			
7.03e+03	-6	9.87	7.15e-11

Second Mandate

```r
y_s_d = cbind(Vdata_s$dummy, Vdata_s$SPY_vol)
colnames(y_s_d)[1:2] <- c("dummy", "vol")
est.VAR_s_d <- VAR(y_s_d,p=6)

#extract results
mod_vol_s_d = est.VAR_s_d$varresult$vol
f_s_d = formula(mod_vol_s_d)
d_s_d = model.frame(mod_vol_s_d)
lm_clean_s_d = lm(f_s_d, data= d_s_d)

#apply Newey-West
nw_vcov_s_d = NeweyWest(lm_clean_s_d, lag=6)
nw_se_s_d = sqrt(diag(nw_vcov_s_d))

#t-stats
coef_s_d = coef(lm_clean_s_d)
t_stat_s_d = coef_s_d/nw_se_s_d

#recalculate p-values
robust_s_d = 2*(1-pt(abs(t_stat_s_d), df = df.residual(lm_clean_s_d)))

nw_se_s_d       <- nw_se_s_d[names(coef(lm_clean_s_d))]
robust_s_d      <- robust_s_d[names(coef(lm_clean_s_d))]

#table
screenreg(lm_clean_s_d, override.se = nw_se_s_d, override.pvalues = robust_s_d, digits = 6)
```

========================== Model 1
————————- dummy.l1 0.006569
(0.010145)
vol.l1 0.299398 ** (0.112369)
dummy.l2 -0.003222 ** (0.001039)
vol.l2 0.015406
(0.043748)
dummy.l3 -0.005538 ** (0.001707)
vol.l3 0.076169 ***(0.008464)***
dummy.l4 0.002474
(0.004957)
vol.l4 0.084229
(0.067860)
dummy.l5 -0.008527
(0.003989)
vol.l5 0.013424 (0.005076)
dummy.l6 -0.003594
(0.003191)
vol.l6 0.126612 *
(0.050031)
const 0.072524 ** (0.024156)
————————- R^2 0.244117
Adj. R^2 0.224424
Num. obs. 512
========================== *** $p < 0.001$; ** $p < 0.01$; * $p < 0.05$

```r
#Table for the effect of volatility on posts for variable dummy
#extract results
mod_post_s_d = est.VAR_s_d$varresult$dummy
ff_s_d = formula(mod_post_s_d)
dd_s_d = model.frame(mod_post_s_d)
lm_clean_post_s_d = lm(ff_s_d, data= dd_s_d)

#apply Newey-West
nw_vcov_post_s_d = NeweyWest(lm_clean_post_s_d, lag=6)
nw_se_post_s_d = sqrt(diag(nw_vcov_post_s_d))

#t-stats
coef_post_s_d = coef(lm_clean_post_s_d)
t_stat_post_s_d = coef_post_s_d/nw_se_post_s_d

#recalculate p-values
robust_post_s_d = 2*(1-pt(abs(t_stat_post_s_d), df = df.residual(lm_clean_post_s_d)))

nw_se_post_s_d       <- nw_se_post_s_d[names(coef(lm_clean_post_s_d))]
robust_post_s_d      <- robust_post_s_d[names(coef(lm_clean_post_s_d))]

#table
screenreg(lm_clean_post_s_d, override.se = nw_se_post_s_d, override.pvalues = robust_post_s_d, digits =
```

```
##
## =========================
##             Model 1
## -------------------------
## dummy.l1    -0.216984 ***
##             (0.020971)
## vol.l1      -0.042631
##             (0.082498)
## dummy.l2    -0.208185 ***
##             (0.020850)
## vol.l2      -0.156762
##             (0.090144)
## dummy.l3    -0.205129 ***
##             (0.021613)
## vol.l3       0.273732 *
##             (0.124059)
## dummy.l4    -0.217831 ***
##             (0.022085)
## vol.l4      -0.204545 ***
##             (0.058272)
## dummy.l5    -0.214630 ***
##             (0.021698)
## vol.l5      -0.103301
##             (0.057821)
## dummy.l6    -0.199256 ***
##             (0.022328)
## vol.l6       0.059322
##             (0.076999)
## const        3.161434 ***
```

```
##              (0.255375)
## -------------------------
## R^2           0.297771
## Adj. R^2      0.279477
## Num. obs.   512
## =========================
## *** p < 0.001; ** p < 0.01; * p < 0.05
```

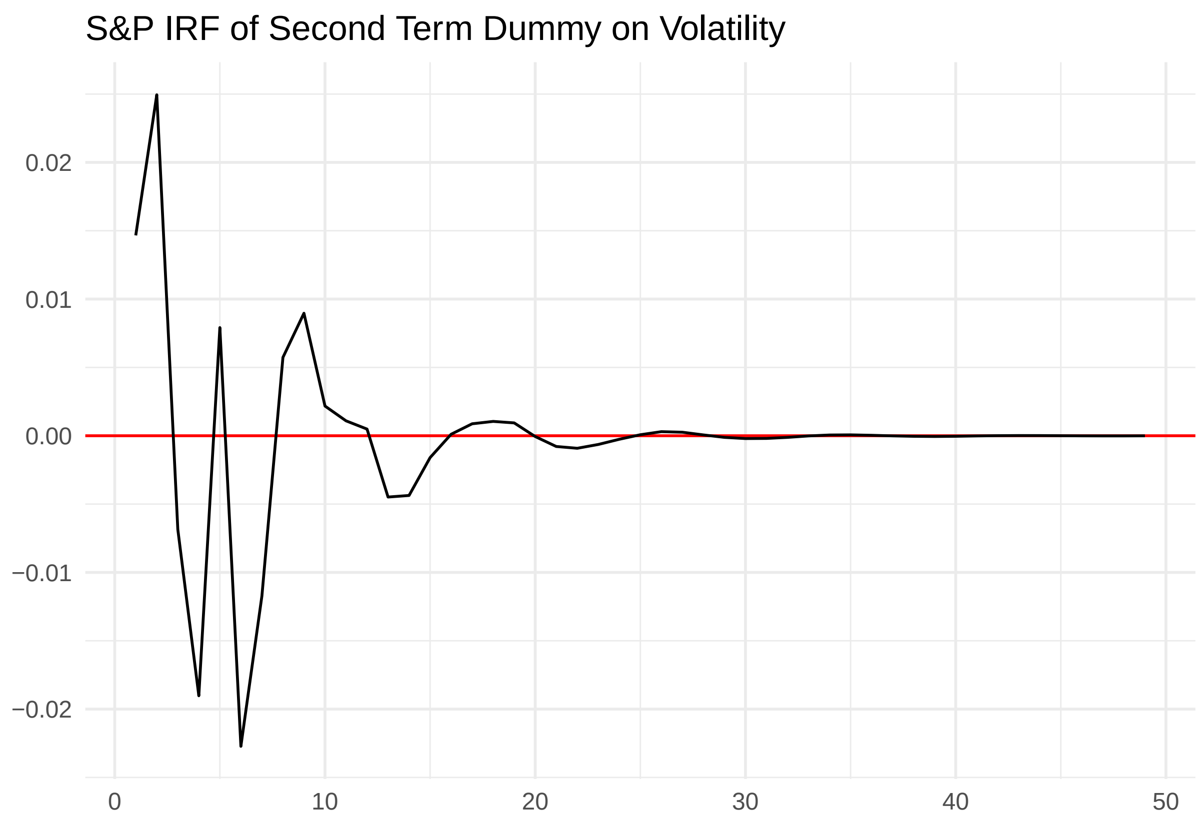
```
#Construct the Robust Omega Matrix
U_s_d = residuals(est.VAR_s_d)
T_s_d = nrow(U_s_d)
Omega_s_d = matrix(0, ncol(U_s_d), ncol(U_s_d))
for(l in 0:L) {
  weight = 1 - l/(L+1)
  Gamma_l__s_d = t(U_s_d[(l+1):T_s_d, , drop=FALSE]) %*% U_s_d[1:(T_s_d-l), , drop=FALSE] /T_s_d
  if (l == 0){
    Omega_s_d = Omega_s_d + Gamma_l__s_d
  } else {
    Omega_s_d = Omega_s_d + weight*(Gamma_l__s_d + t(Gamma_l__s_d))
  }
}


#make the B matrix
loss_s_d <- function(param_s_d){
  #Define the restriction
  B_s_d <- matrix(c(param_s_d[1], param_s_d[2], 0, param_s_d[3]), ncol = 2)

  #Make BB' approximatively equal to omega
  X_s_d <- Omega_s_d - B_s_d %*% t(B_s_d)

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

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

print(cbind(Omega_s_d,B.hat_s_d %*% t(B.hat_s_d)))
```

```
##           dummy       vol
## dummy 9.7947444 0.0458797 9.79474361 0.04587955
## vol   0.0458797 0.1852225 0.04587955 0.18522157
```

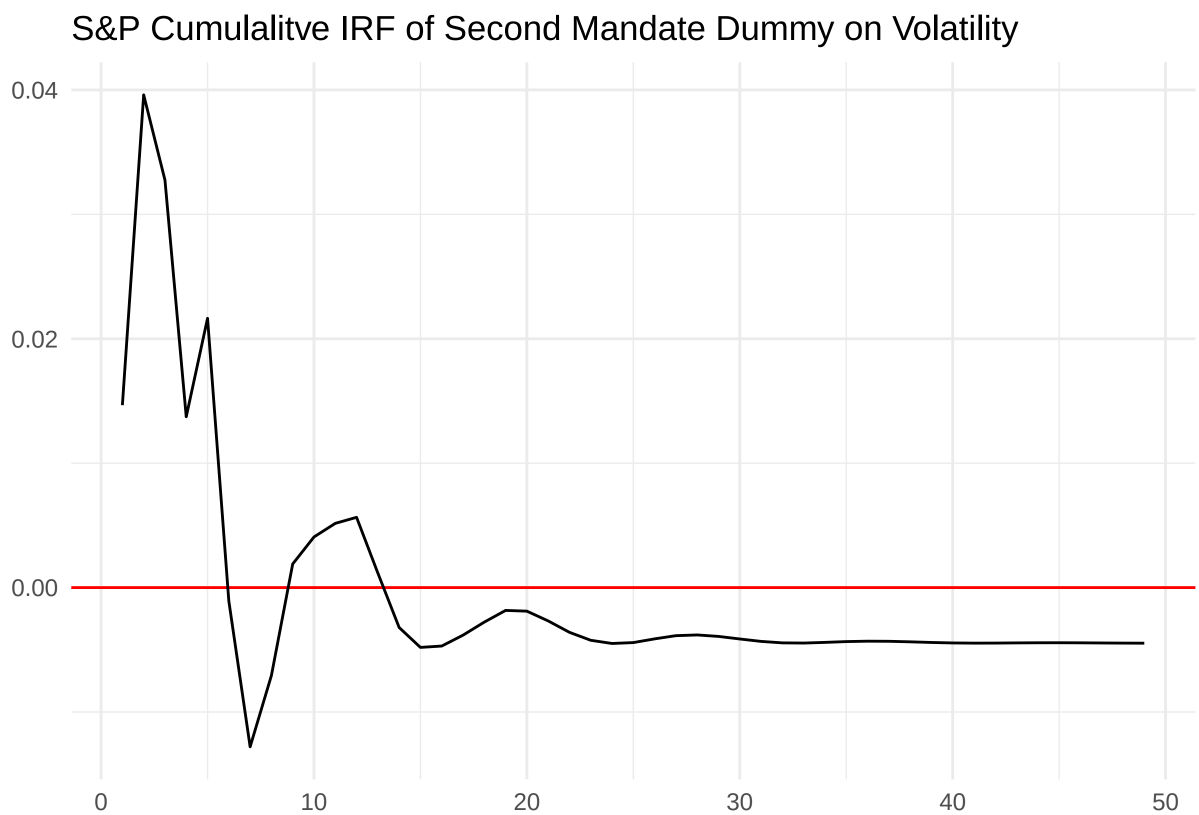
```
B.hat_s_d
```

```
##             [,1]      [,2]
## [1,] 3.12965551  0.000000
## [2,] 0.01465962 -0.430124
```

```r
#get back the coefficient of est.VAR
phi_s_d <- Acoef(est.VAR_s_d)
PHI_s_d = make.PHI(phi_s_d)

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

#Simulate the IRF
p_s_d <- length(phi_s_d)
n_s_d <- dim(phi_s_d[[1]])[1]

Y_s_d <- simul.VAR(c=c_s_d, Phi = phi_s_d, B = B.hat_s_d, nb.sim ,y0.star=rep(0, n_s_d*p_s_d),
                  indic.IRF = 1, u.shock = c(1,0))


#Plot the IRF
Yd_s_d = data.frame(
  period = 1:nrow(Y_s_d),
  response = Y_s_d[,2])

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




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




```r
#does vol granger cause dummy
grangertest(y_s_d[,c("vol","dummy")], order =6)
```

Res.Df	Df	F	Pr(>F)
499			
505	-6	0.303	0.935

```r
#does dummy granger cause vol
grangertest(y_s_d[,c("dummy", "vol")], order =6)
```

Res.Df	Df	F	Pr(>F)
499			
505	-6	0.621	0.713