ASHR SVAR Models

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

```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')))
```

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.

```r
y = cbind(Vdata$dummy, Vdata$ASHR_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      -0.000006 ***
##               (0.000001)
## vol.l1         0.282482 ***
##               (0.023326)
## dummy.l2      -0.000005 ***
##               (0.000001)
## vol.l2         0.072926 *
##               (0.031382)
## dummy.l3      -0.000006 ***
##               (0.000001)
## vol.l3         0.047892 ***
##               (0.008643)
## dummy.l4      -0.000004 **
##               (0.000001)
## vol.l4         0.056084 ***
##               (0.014109)
## dummy.l5      -0.000006 ***
##               (0.000001)
## vol.l5         0.059763 ***
##               (0.017855)
## dummy.l6      -0.000005 ***
##               (0.000001)
## vol.l6         0.109466 ***
##               (0.023870)
## const          0.000095 ***
##               (0.000009)
## ---------------------------
## R^2            0.270711
## Adj. R^2       0.270236
## 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      -0.092434 *** 
##               (0.003468)    
## vol.l1       -24.484296     
##              (31.437508)    
## dummy.l2      -0.084188 *** 
##               (0.003669)    
## vol.l2       -95.442834     
##              (57.815084)    
## dummy.l3      -0.077451 *** 
##               (0.004069)    
## vol.l3         5.890426     
##              (81.556094)    
## dummy.l4      -0.075739 *** 
##               (0.003714)    
## vol.l4      -153.454700 *** 
##              (41.958279)    
## dummy.l5      -0.087766 *** 
##               (0.003737)    
## vol.l5        -7.311264     
##              (32.226447)    
## dummy.l6      -0.099812 *** 
##               (0.004184)    
## vol.l6       181.502758 **  
##              (62.005728)    
## const          1.724904 *** 
##               (0.040648)    
## ----------------------------
## R^2            0.154875     
## Adj. R^2       0.154324     
## 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 = 160.42, 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)))
```

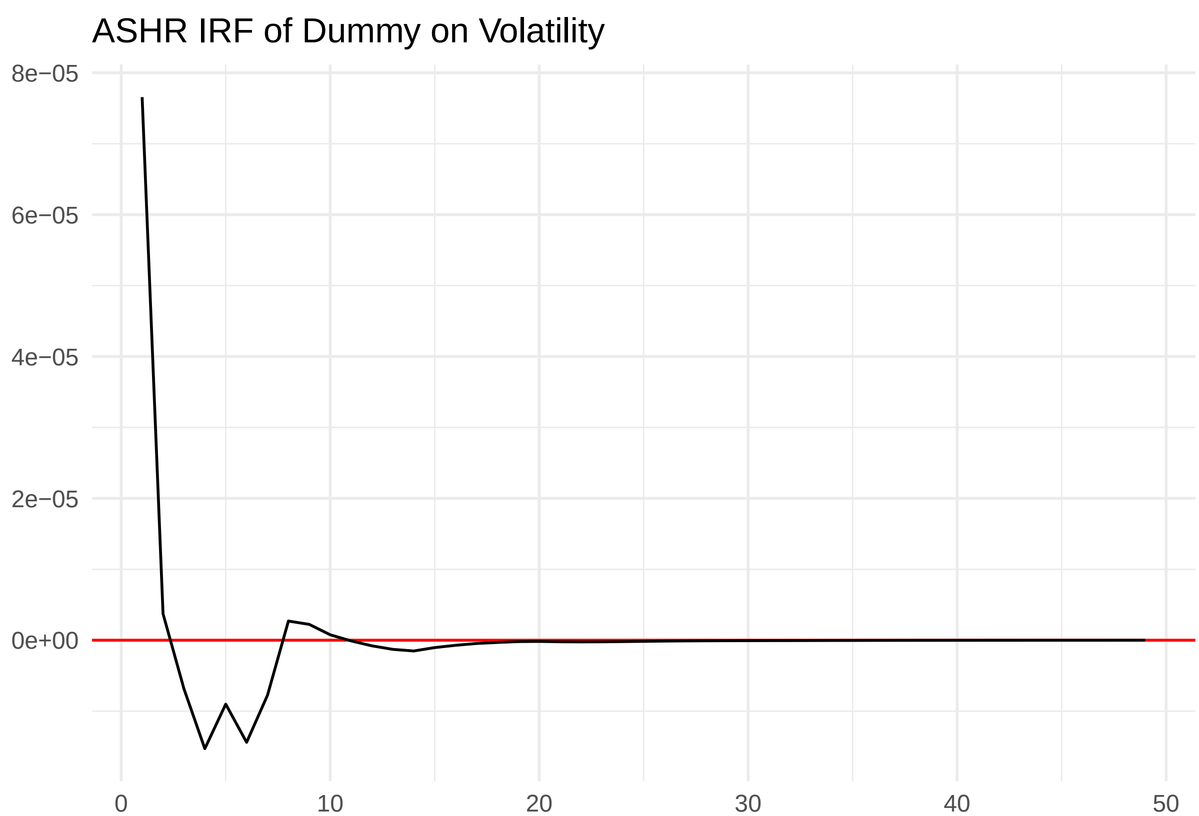
```
##              dummy          vol
## dummy 1.024704e+01 2.442181e-04 1.024704e+01 2.450982e-04
## vol   2.442181e-04 1.462361e-07 2.450982e-04 4.466506e-05
```

```r
B.hat
```

```
##              [,1]        [,2]
## [1,] 3.201099e+00 0.000000000
## [2,] 7.656688e-05 0.006682754
```

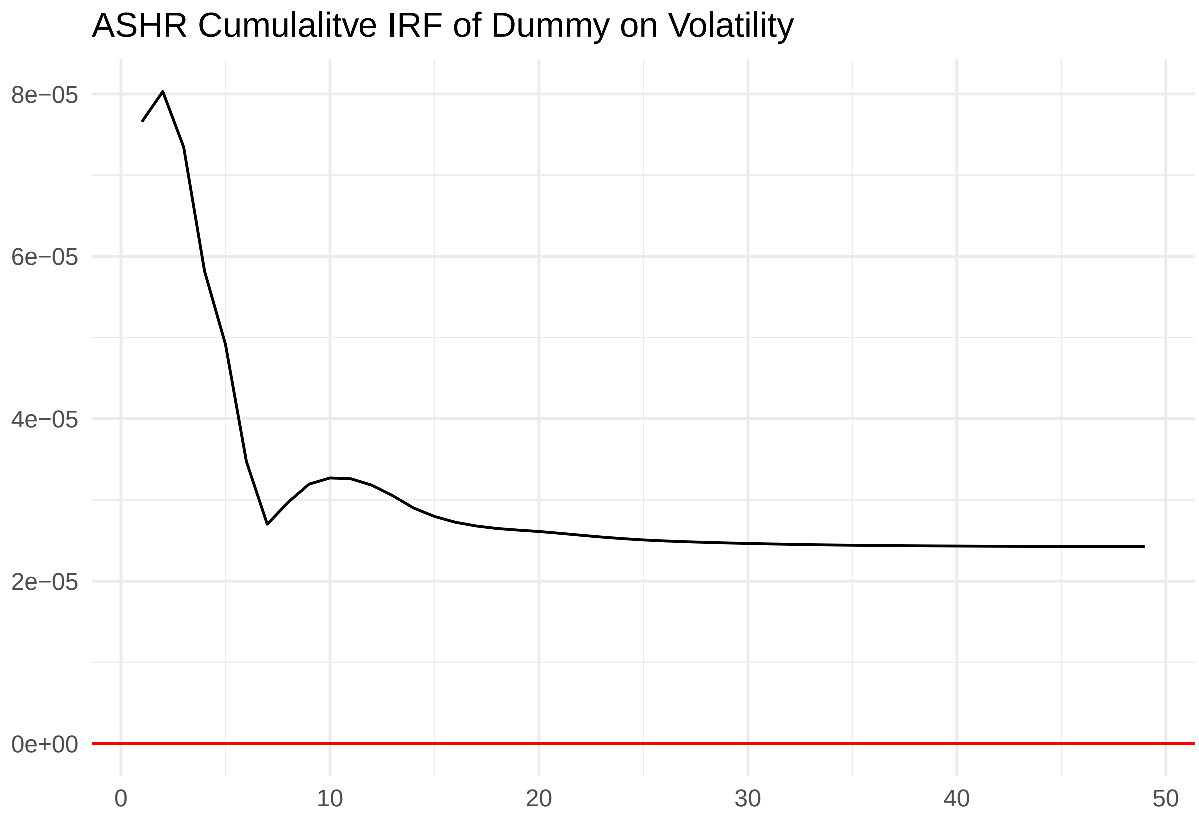
```r
nb.sim = 7*7
#get back the coefficient of est.VAR
phi <- Acoef(est.VAR)
PHI = make.PHI(phi)

#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("ASHR 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("ASHR Cumulalitve IRF of Dummy on Volatility") +
  ylab("")+
```

```
  xlab("") +
  theme_minimal()
```



Post Counts

```
y2 = cbind(Vdata$N, Vdata$ASHR_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          -0.000001 ***
##               (0.000000)
## vol.l1         0.282497 ***
##               (0.023442)
## N.l2          -0.000001 ***
##               (0.000000)
## vol.l2         0.072640 *
##               (0.031158)
## N.l3          -0.000002 ***
##               (0.000000)
## vol.l3         0.047738 ***
##               (0.008436)
## N.l4          -0.000001 *
##               (0.000000)
## vol.l4         0.056237 ***
##               (0.014097)
## N.l5          -0.000002 ***
##               (0.000000)
## vol.l5         0.059528 ***
##               (0.017745)
## N.l6          -0.000001 ***
##               (0.000000)
## vol.l6         0.109380 ***
##               (0.023887)
## const          0.000081 ***
##               (0.000009)
## ---------------------------
## R^2            0.268901
## Adj. R^2       0.268425
## 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          -0.042465 ***
##               (0.003696)
## vol.l1      -129.076390
##              (98.105219)
## N.l2          -0.038882 ***
##               (0.004629)
## vol.l2       -22.372568
##             (276.983664)
## N.l3          -0.026617 ***
##               (0.004632)
## vol.l3       -26.972939
##             (204.041589)
## N.l4          -0.021873 ***
##               (0.005161)
## vol.l4      -433.230826 **
##             (146.247539)
## N.l5          -0.040310 ***
##               (0.004097)
## vol.l5       -27.699216
##              (90.152967)
## N.l6          -0.047981 ***
##               (0.005136)
## vol.l6       548.279747 **
##             (199.229337)
## const          3.519787 ***
##               (0.095735)
## ---------------------------
## R^2            0.100612
## Adj. R^2       0.100026
## 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)))
```

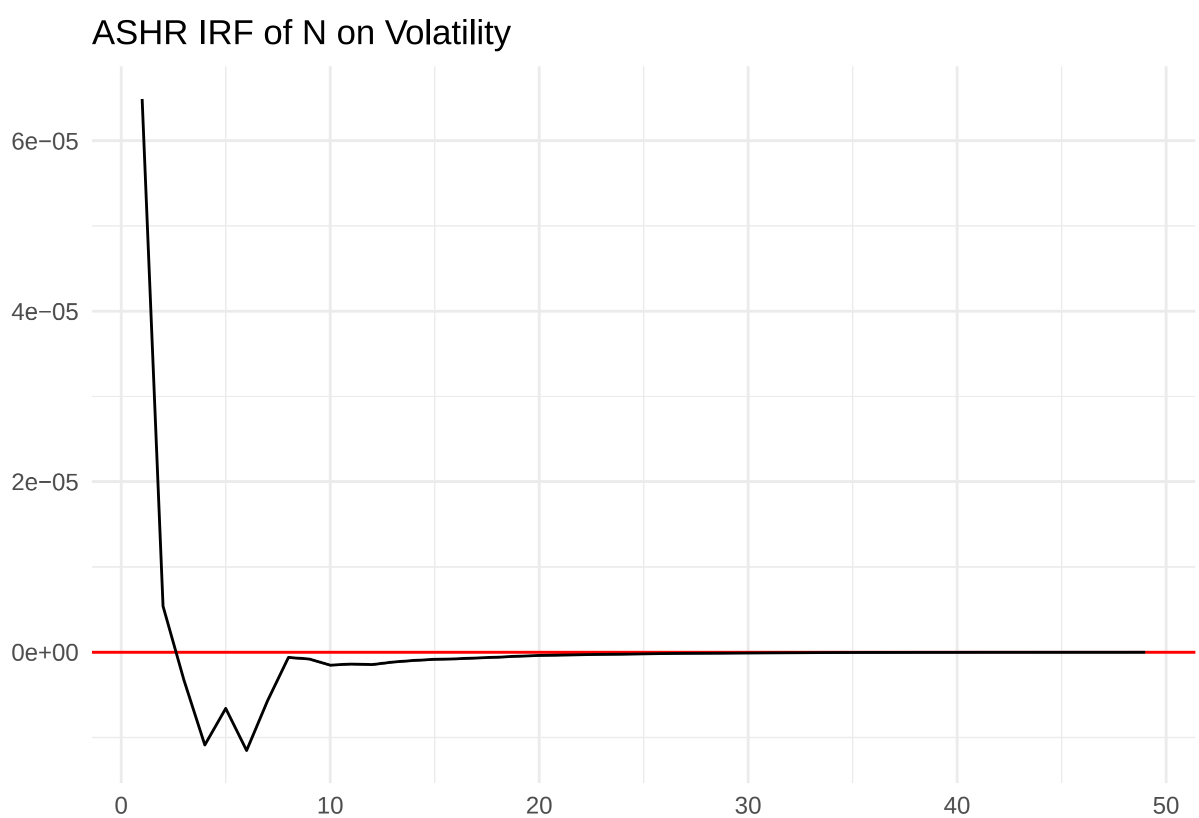
```
##                N          vol
## N   86.36287757 5.984100e-04 8.636288e+01 6.030108e-04
## vol  0.00059841 1.452053e-07 6.030108e-04 2.564515e-05
```

```r
B.hat2
```

```
##              [,1]        [,2]
## [1,] 9.293163e+00 0.000000000
## [2,] 6.488758e-05 0.005063688
```

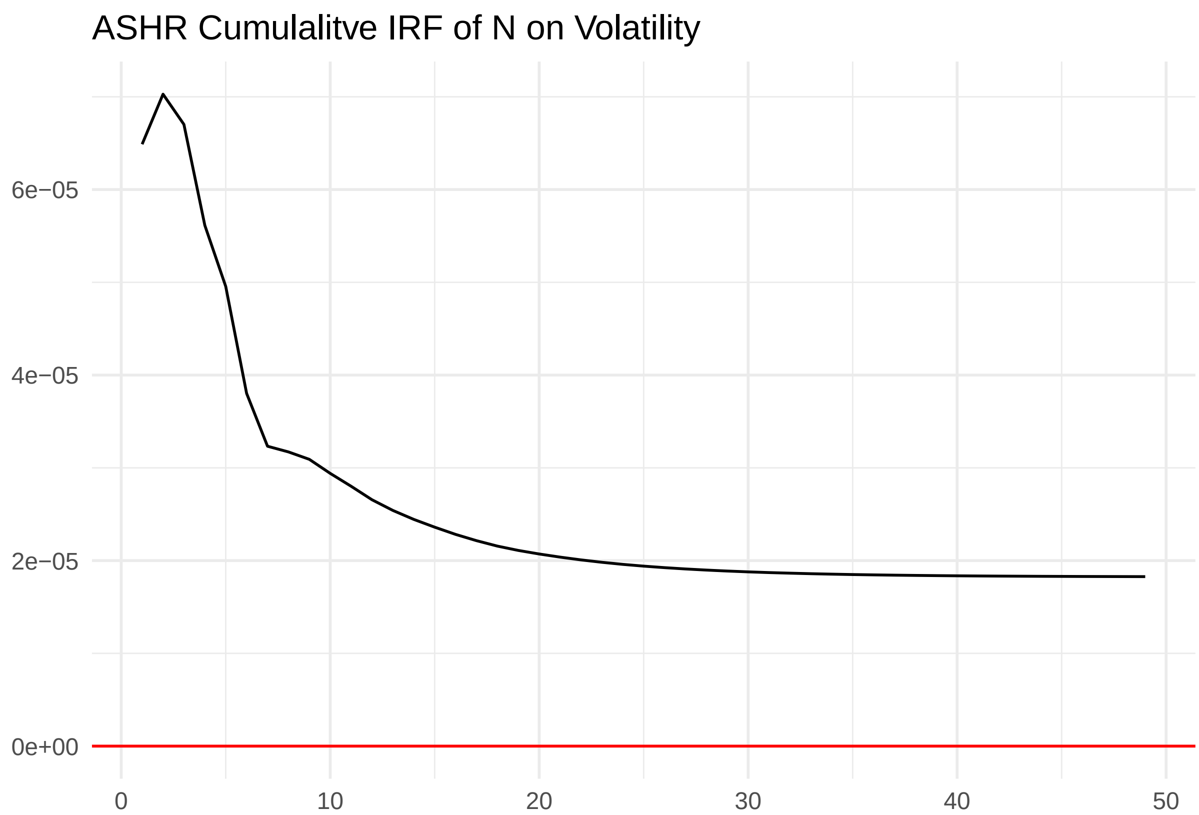
```r
#get back the coefficient of est.VAR
phi2 <- Acoef(est.VAR2)
PHI2 = make.PHI(phi2)

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

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

```r
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("ASHR IRF of N on Volatility") +
  ylab("")+
  xlab("") +
  theme_minimal()
```




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

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




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

Res.Df	Df	F	Pr(>F)
2e+04			
2e+04	-6	13.9	8.02e-16

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

Res.Df	Df	F	Pr(>F)
2e+04			
2e+04	-6	3.17	0.00413

Trade Mention

```r
y4 = cbind(Vdata$trade, Vdata$ASHR_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      -0.000025 ***
##               (0.000006)
## vol.l1         0.281371 ***
##               (0.022717)
## trade.l2       0.000012
##               (0.000019)
## vol.l2         0.071544 *
##               (0.030181)
## trade.l3      -0.000019 *
##               (0.000009)
## vol.l3         0.045746 ***
##               (0.008280)
## trade.l4      -0.000009
##               (0.000008)
## vol.l4         0.056738 ***
##               (0.013529)
## trade.l5      -0.000013 ***
##               (0.000004)
## vol.l5         0.056226 **
##               (0.017213)
```

```
## trade.l6        -0.000013 ***
##                 (0.000003)
## vol.l6           0.109845 ***
##                 (0.022503)
## const            0.000062 ***
##                 (0.000009)
## ---------------------------
## R^2              0.266464
## Adj. R^2         0.265986
## Num. obs.    19965
## ===========================
## *** p < 0.001; ** p < 0.01; * p < 0.05
```

```
#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         0.026737
##                 (0.018212)
## vol.l1          -9.226534 **
##                 (3.417063)
## trade.l2         0.020039 *
##                 (0.008617)
## vol.l2           4.799208
##                 (6.150288)
## trade.l3         0.022165
##                 (0.016504)
## vol.l3          25.623669
##                (25.460008)
## trade.l4         0.024458
```

```
##                (0.012912)
## vol.l4       -14.855768 *
##                (7.295870)
## trade.l5       0.018788
##                (0.010879)
## vol.l5        -0.433456
##                (4.696217)
## trade.l6       0.013759
##                (0.010389)
## vol.l6        -2.276427
##                (2.921722)
## const          0.029769 ***
##                (0.002707)
## ---------------------------
## R^2            0.018726
## Adj. R^2       0.018087
## Num. obs.  19965
## ===========================
## *** p < 0.001; ** p < 0.01; * p < 0.05
```

```r
#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)))
```

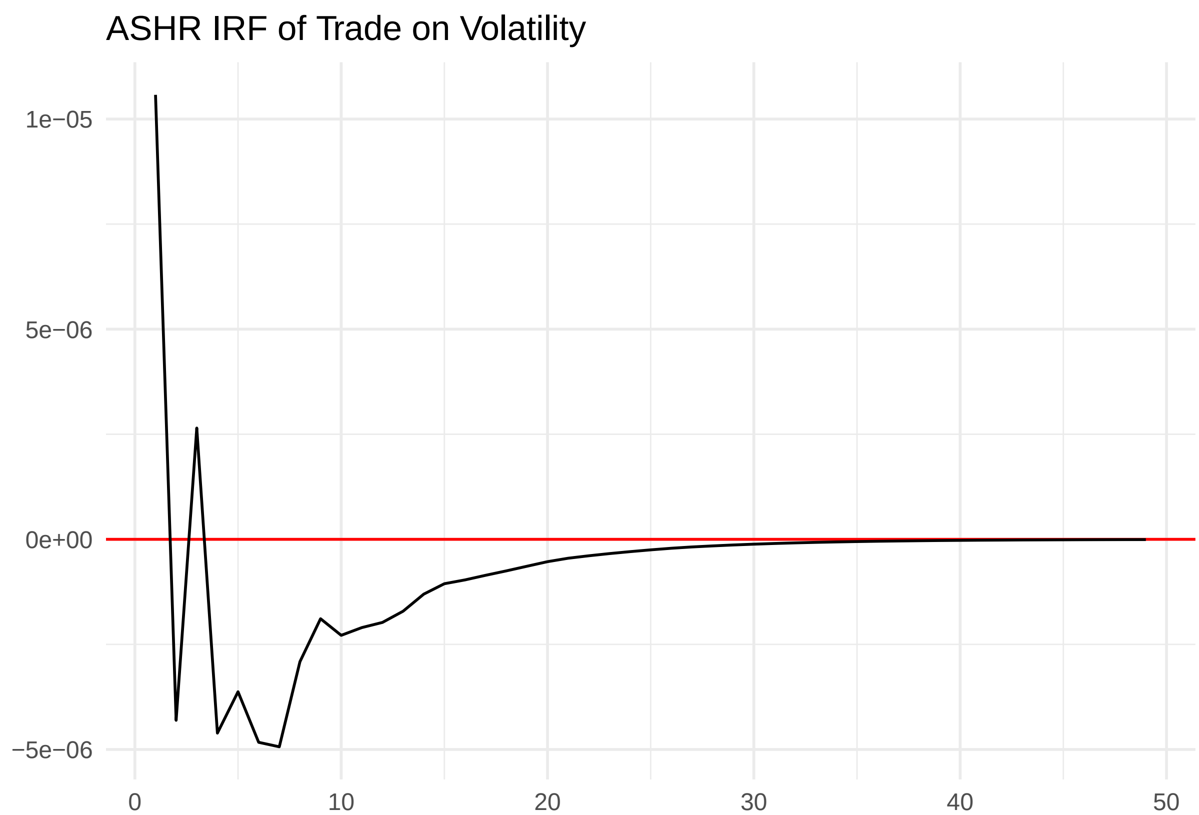
```
##               trade         vol
```

```
## trade 8.201947e-02 3.026524e-06 8.201858e-02 3.028763e-06
## vol   3.026524e-06 1.426699e-07 3.028763e-06 1.210986e-05
```

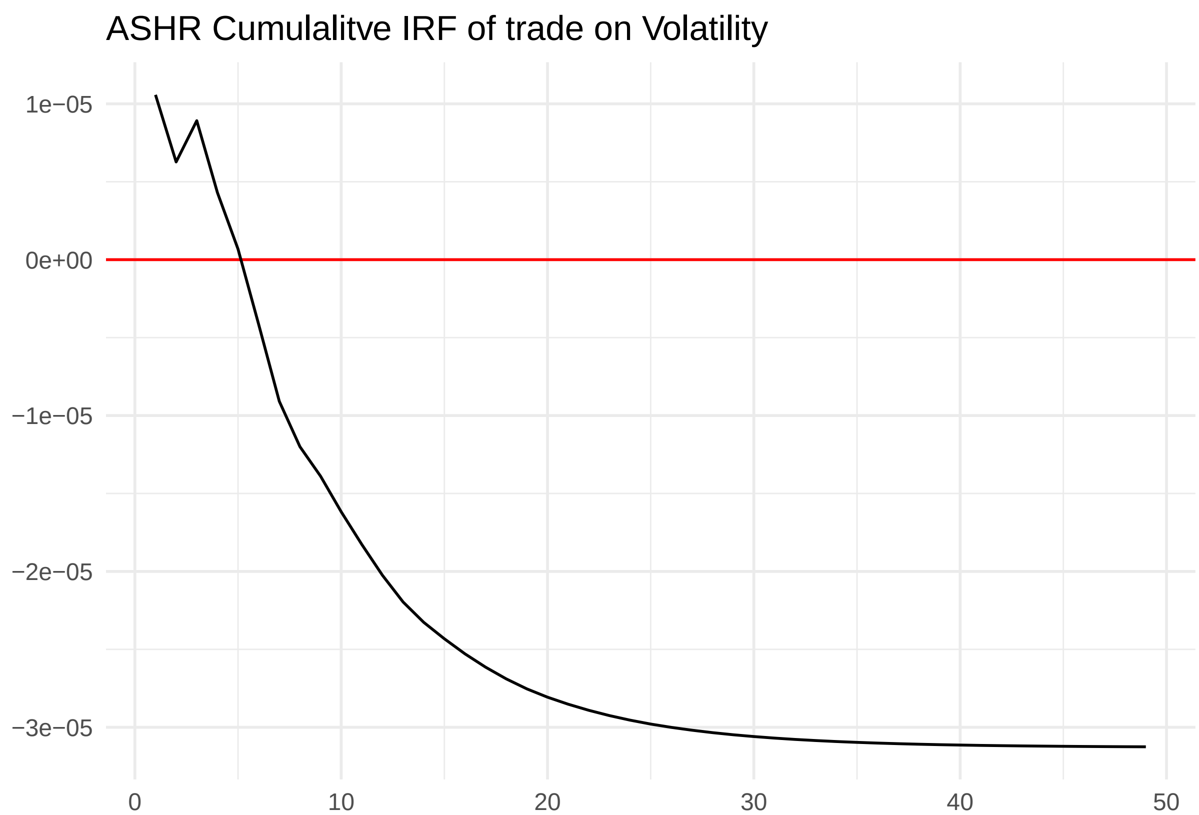
```r
B.hat4
```

```
##              [,1]        [,2]
## [1,] 0.2863888688 0.000000000
## [2,] 0.0000105757 0.003479906
```

```r
#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("ASHR IRF of Trade on Volatility") +
  ylab("")+
  xlab("") +
  theme_minimal()
```




```r
ggplot(Yd4,aes(x=period, y=cumsum(response))) +
  geom_hline(yintercept = 0, color="red") +
  geom_line() +
  theme_light() +
  ggtitle("ASHR 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	5.36	1.55e-05

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

Res.Df	Df	F	Pr(>F)
2e+04			
2e+04	-6	2.79	0.0103

China Mention

```r
ychina = cbind(Vdata$china, Vdata$ASHR_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      -0.000005
##               (0.000003)
## vol.l1         0.280637 ***
##               (0.022879)
## china.l2      -0.000004
##               (0.000004)
## vol.l2         0.072261 *
##               (0.030357)
## china.l3      -0.000011 ***
##               (0.000002)
## vol.l3         0.045298 ***
##               (0.008336)
## china.l4      -0.000007 *
##               (0.000003)
## vol.l4         0.056264 ***
##               (0.013530)
## china.l5      -0.000007 *
##               (0.000003)
## vol.l5         0.056857 ***
##               (0.017212)
## china.l6      -0.000010 ***
##               (0.000003)
```

```
## vol.l6          0.109272 ***
##                (0.023313)
## const           0.000063 ***
##                (0.000009)
## ---------------------------
## R^2             0.266273
## Adj. R^2        0.265795
## Num. obs.   19965
## ===========================
## *** p < 0.001; ** p < 0.01; * p < 0.05
```

```
#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, digit
```

```
##
## ===========================
##            Model 1
## ---------------------------
## china.l1        0.075875 *
##                (0.034050)
## vol.l1        -10.102292 ***
##                (2.724874)
## china.l2        0.044876 *
##                (0.021535)
## vol.l2          3.212025
##                (7.103060)
## china.l3        0.007296
##                (0.010495)
## vol.l3         10.034793
##                (7.822179)
## china.l4        0.026711 **
##                (0.010004)
## vol.l4        -14.106424 ***
```

```
##                (4.258786)
## china.l5        0.048839
##                (0.034024)
## vol.l5         -4.365003
##                (3.514978)
## china.l6        0.054840
##                (0.048399)
## vol.l6          3.977833
##                (4.376596)
## const           0.047264 ***
##                (0.006139)
## ---------------------------
## R^2             0.034788
## Adj. R^2        0.034159
## Num. obs.   19965
## ===========================
## *** p < 0.001; ** p < 0.01; * p < 0.05
```

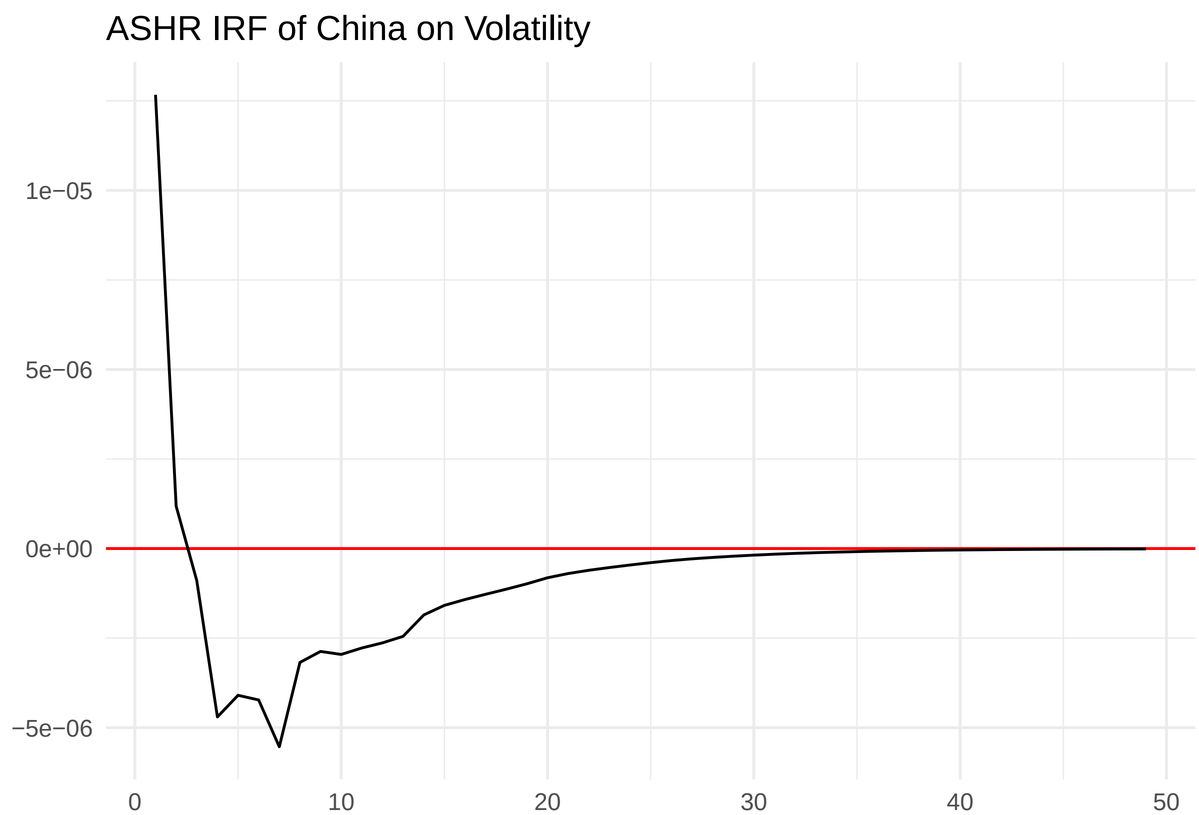
```
#Recreate a Robust Omega Matrix
Uchina = residuals(est.VARchina)
Tchina = nrow(Uchina)
Omegachina = matrix(0, ncol(Uchina), ncol(Uchina))
for(l in 0:L) {
  weight = 1 - l/(L+1)
  Gamma_l_china = t(Uchina[(l+1):Tchina, , drop=FALSE]) %*% Uchina[1:(Tchina-l), , drop=FALSE] /Tchina
  if (l == 0){
    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 1.939575e-01 5.702559e-06 1.939570e-01 5.579026e-06
## vol   5.702559e-06 1.425509e-07 5.579026e-06 1.809801e-05
```

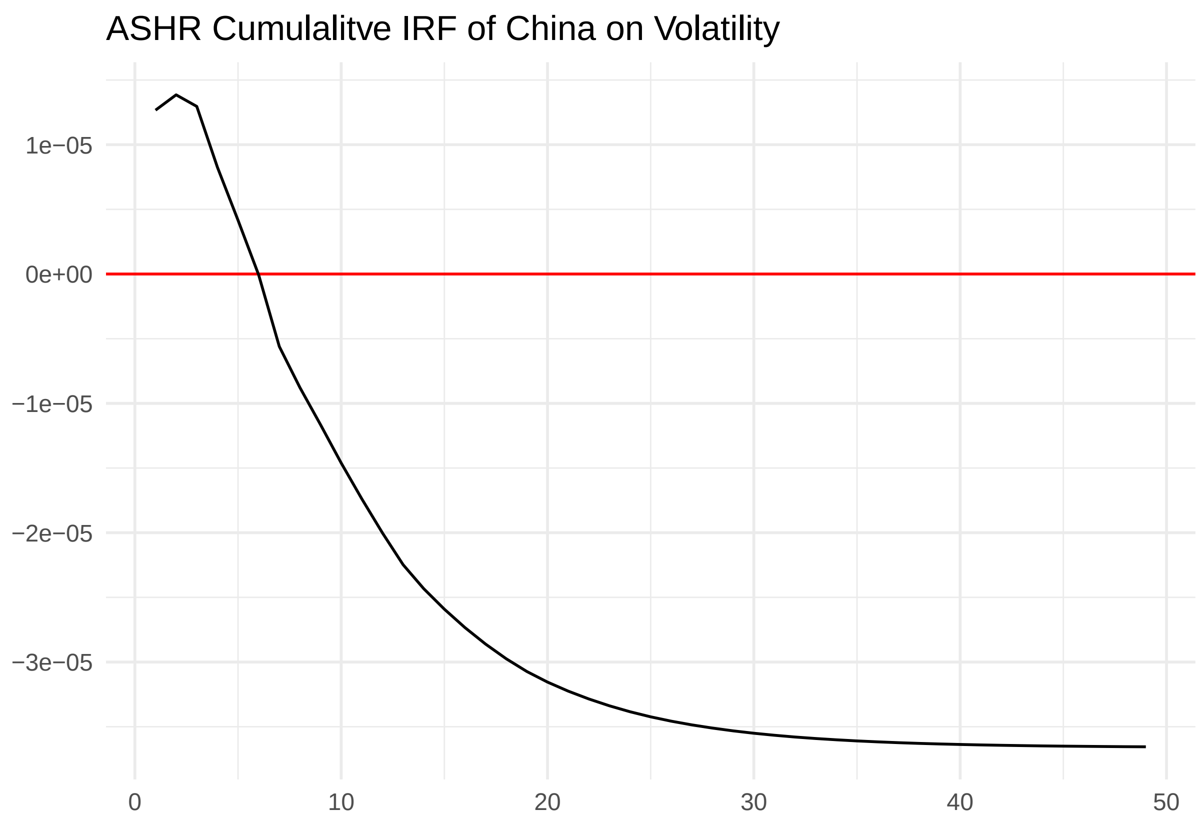
```r
B.hatchina
```

```
##              [,1]        [,2]
## [1,] 4.404055e-01 0.000000000
## [2,] 1.266793e-05 0.004254156
```

```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("ASHR 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("ASHR 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	1.08	0.372

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

Res.Df	Df	F	Pr(>F)
2e+04			
2e+04	-6	1.92	0.0733

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$dummy, Vdata_f$ASHR_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 -0.000003 ***(0.000001)***
vol.l1 0.241914 (0.041504)
dummy.l2 -0.000003 ***(0.000001)***
vol.l2 0.073400 (0.019082)
dummy.l3 -0.000005 ***(0.000001)***
vol.l3 0.065194 *(0.023639)*
dummy.l4 -0.000004 **(0.000001)**
vol.l4 0.064584 (0.020922)
dummy.l5 -0.000005 ***(0.000001)***
vol.l5 0.090550 (0.022232)
dummy.l6 -0.000005 ***(0.000001)***
vol.l6 0.176515 *(0.059500)*
const 0.000061 **(0.000009)**

—————————— **R^2 0.350208**
Adj. R^2 0.349005
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     -0.130197 ***
##              (0.006089)
## vol.l1       68.970633
##            (134.059501)
## dummy.l2     -0.106720 ***
##              (0.005931)
## vol.l2     -251.984638
##            (131.858784)
## dummy.l3     -0.090644 ***
##              (0.006611)
## vol.l3     -316.430144 *
##            (153.259332)
## dummy.l4     -0.091087 ***
##              (0.006437)
## vol.l4     -244.070250
##            (131.284080)
## dummy.l5     -0.115207 ***
##              (0.007073)
## vol.l5      303.933274
##            (233.516958)
```

```
## dummy.l6     -0.138102 ***
##              (0.007732)
## vol.l6      515.688924 *
##            (223.552268)
## const         1.980841 ***
##              (0.063785)
## --------------------------
## R^2           0.184354
## Adj. R^2      0.182844
## Num. obs.  7036
## ==========================
## *** p < 0.001; ** p < 0.01; * p < 0.05
```

```
#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)))
```

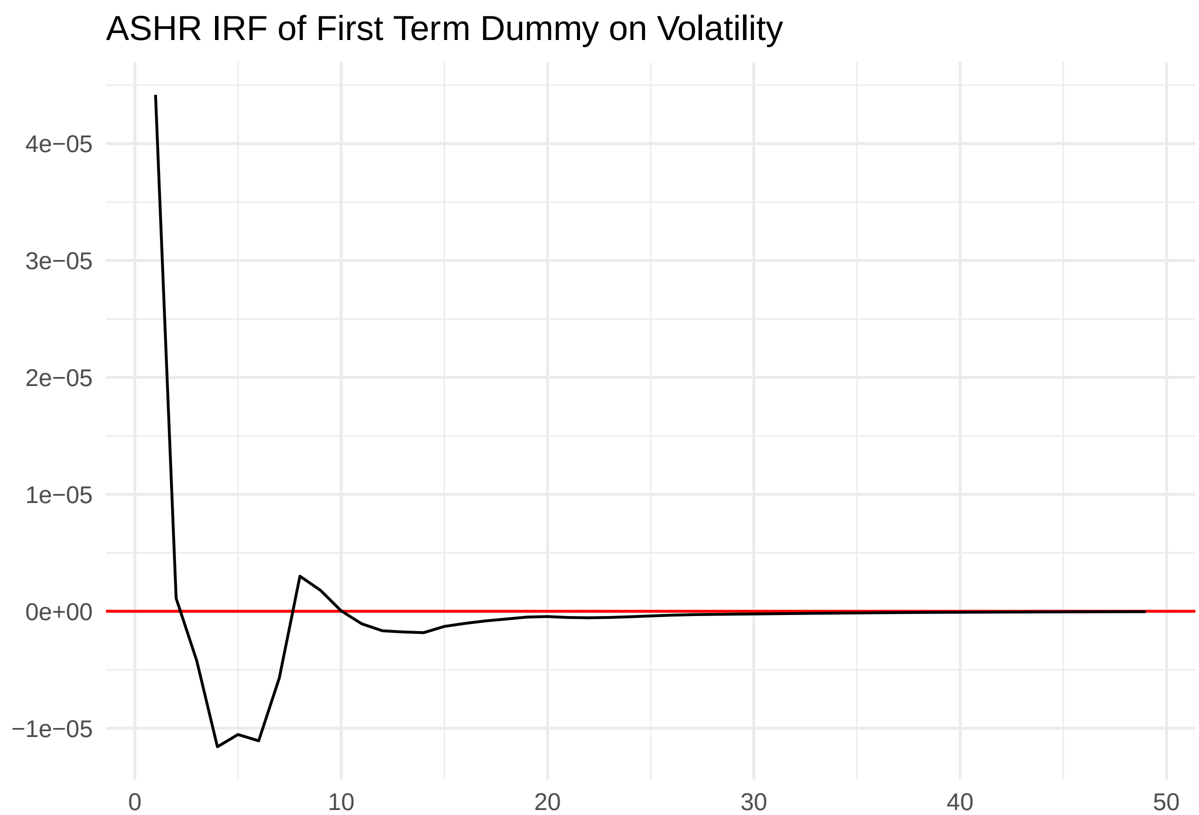
```
##              dummy          vol
## dummy 9.9149258293 1.389427e-04 9.9149251733 1.390914e-04
## vol   0.0001389427 4.024686e-08 0.0001390914 2.429784e-06
```

```
B.hat_f_d
```

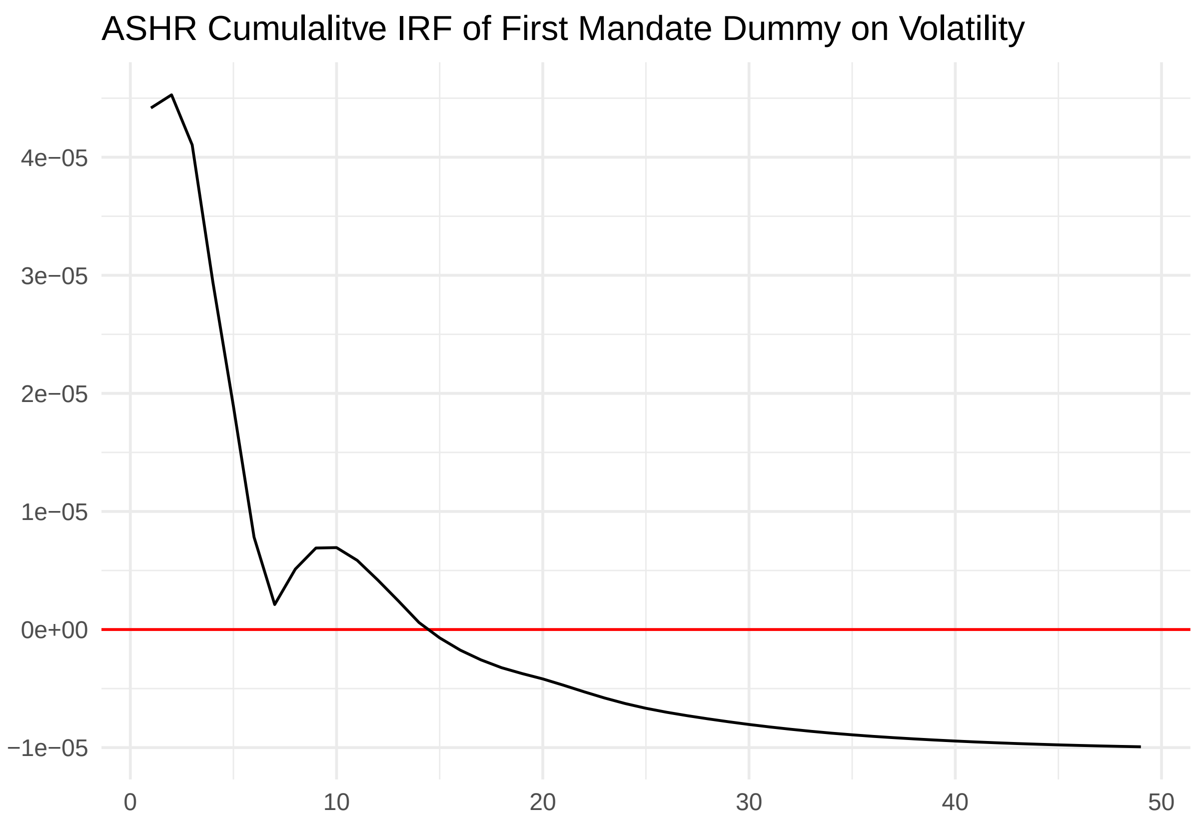
```
##               [,1]       [,2]
```

```
## [1,] 3.148797e+00 0.00000000
## [2,] 4.417287e-05 0.00155815
```

```
#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("ASHR 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("ASHR 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	4.09	0.000418

```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	15.1	2.67e-17

Second Mandate

```r
y_s_d = cbind(Vdata_s$dummy, Vdata_s$ASHR_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 -0.000002
(0.000003)
vol.l1 0.453713 ** (0.139092)
dummy.l2 -0.000003
(0.000002)
vol.l2 0.073292
(0.094132)
dummy.l3 -0.000005 ***(0.000002)***
vol.l3 -0.063668
(0.032167)
dummy.l4 -0.000002
(0.000002)
vol.l4 0.082677
(0.089570)
dummy.l5 -0.000006 (0.000002)
vol.l5 -0.003739
(0.037991)
dummy.l6 -0.000006 ** (0.000002)
vol.l6 0.111940
(0.073132)
const 0.000065 ***(0.000016)***
———————- ***R^2 0.385822***
Adj. R^2 0.369822
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      -0.215778 ***
##               (0.021070)
## vol.l1      -212.214190
##             (241.235993)
## dummy.l2      -0.204705 ***
##               (0.020905)
## vol.l2      -377.569865
##             (221.982942)
## dummy.l3      -0.208529 ***
##               (0.022295)
## vol.l3       527.218353
##             (421.359653)
## dummy.l4      -0.213563 ***
##               (0.022187)
## vol.l4      -450.277869 *
##             (193.512623)
## dummy.l5      -0.214362 ***
##               (0.021926)
## vol.l5      -151.169780
##             (262.979738)
## dummy.l6      -0.200965 ***
##               (0.022394)
## vol.l6       222.174258
##             (306.804126)
## const          3.172941 ***
```

```
##               (0.258307)
## --------------------------
## R^2           0.297405
## Adj. R^2      0.279101
## Num. obs.   512
## ==========================
## *** p < 0.001; ** p < 0.01; * p < 0.05
```

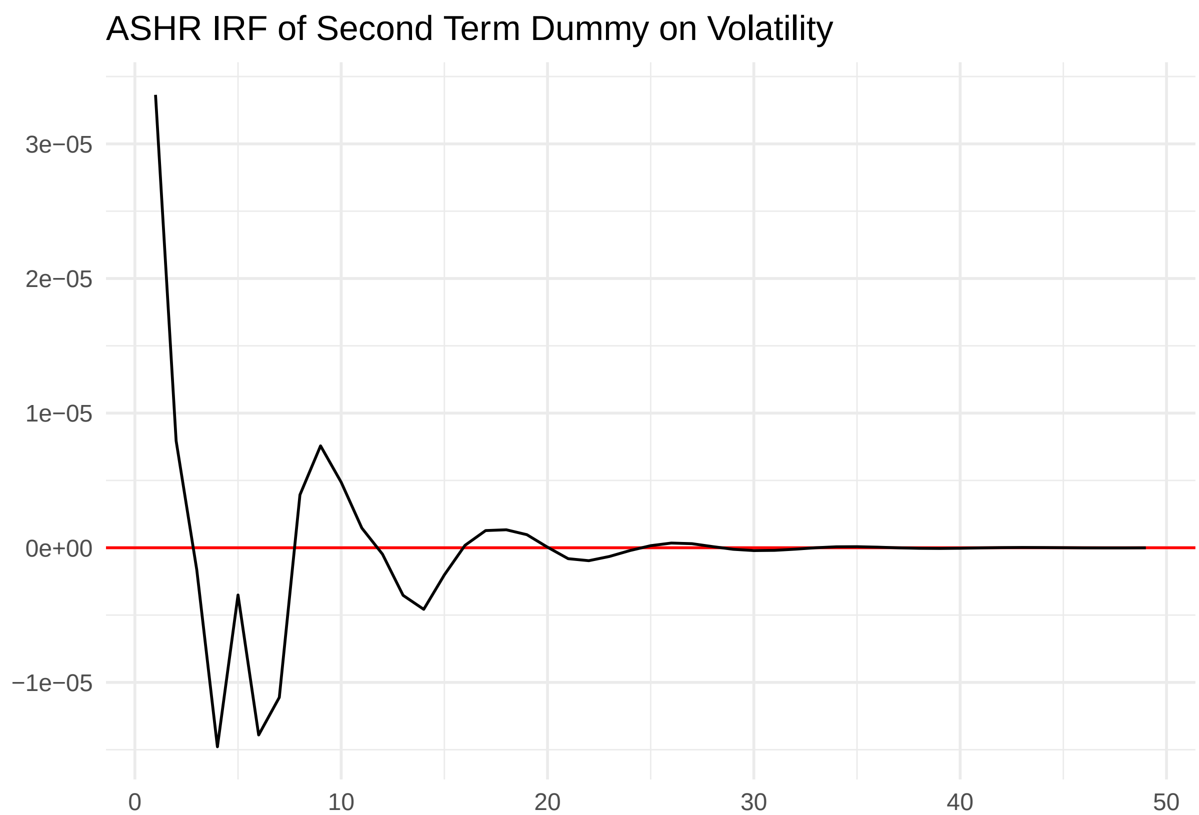
```
#Construct the Robust Omega Matrix
U_s_d = residuals(est.VAR_s_d)
T_s_d = nrow(U_s_d)
Omega_s_d = matrix(0, ncol(U_s_d), ncol(U_s_d))
for(l in 0:L) {
  weight = 1 - l/(L+1)
  Gamma_l__s_d = t(U_s_d[(l+1):T_s_d, , drop=FALSE]) %*% U_s_d[1:(T_s_d-l), , drop=FALSE] /T_s_d
  if (l == 0){
    Omega_s_d = 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 9.8009866129 1.052217e-04 9.800984276 1.05317e-04
## vol   0.0001052217 3.718029e-08 0.000105317 3.20576e-05
```

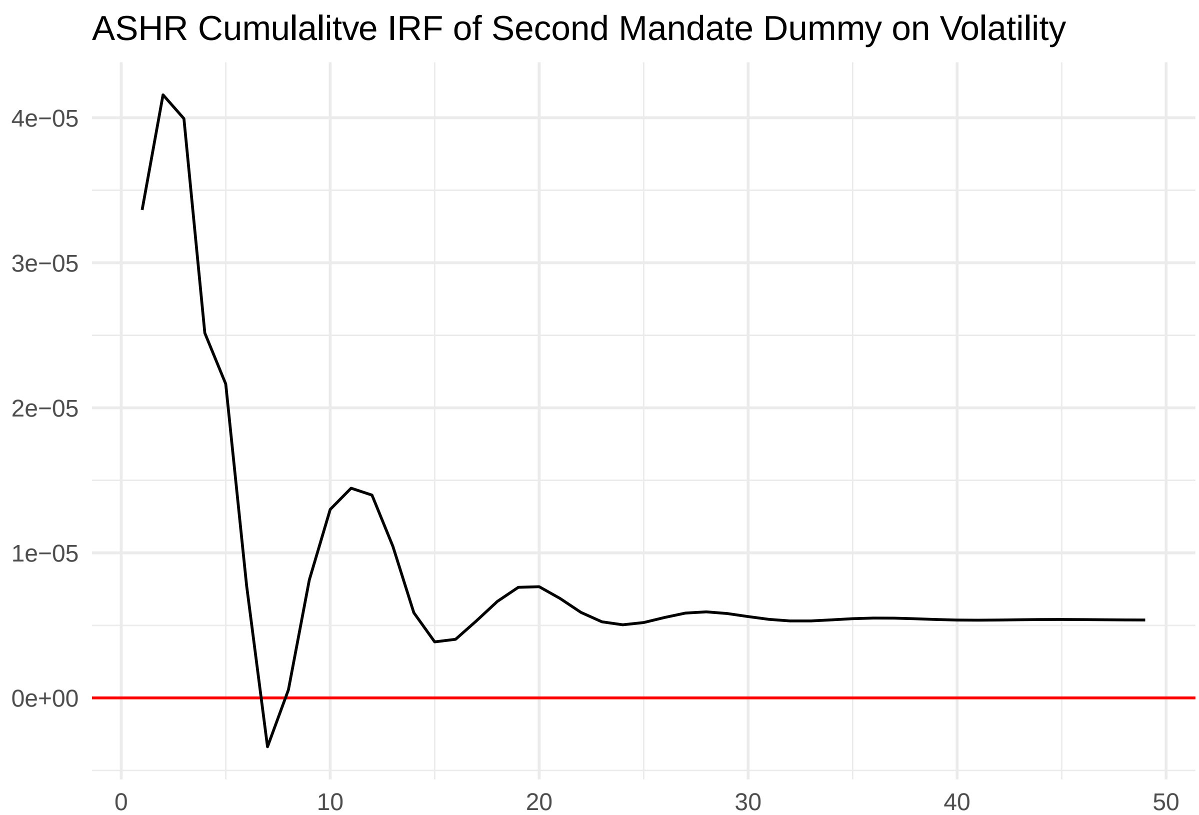
```
B.hat_s_d
```

```
##              [,1]        [,2]
## [1,] 3.130652e+00 0.000000000
## [2,] 3.364058e-05 0.005661844
```

```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("ASHR 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("ASHR 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	0.26	0.955

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

Res.Df	Df	F	Pr(>F)
499			
505	-6	1.01	0.418