VGK SVAR Models

Contents

Setup

Load packages & functions

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

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

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

Load Data

```
#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$dummy, Vdata$VGK_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        0.000002
##                (0.000004)
## vol.l1          0.258482 *
##                (0.108773)
## dummy.l2       -0.000006 ***
##                (0.000002)
## vol.l2          0.021449
##                (0.082689)
## dummy.l3       -0.000015 ***
##                (0.000001)
## vol.l3          0.028590
##                (0.014848)
## dummy.l4        0.000005
##                (0.000011)
## vol.l4          0.036605 *
##                (0.016773)
## dummy.l5       -0.000016 ***
##                (0.000003)
## vol.l5          0.032120 *
##                (0.015363)
## dummy.l6       -0.000010 ***
##                (0.000002)
## vol.l6          0.057830 **
##                (0.020860)
## const           0.000267 ***
##                (0.000028)
## ---------------------------
## R^2             0.136174
## Adj. R^2        0.135611
## Num. obs.   19965
## ===========================
## *** p < 0.001; ** p < 0.01; * p < 0.05
```

```r
#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      -0.092611 ***
##               (0.003383)
## vol.l1       -16.120257
##              (10.081167)
## dummy.l2      -0.085967 ***
##               (0.003444)
## vol.l2       -15.929253
##              (17.836825)
## dummy.l3      -0.077783 ***
##               (0.003512)
## vol.l3         5.310043
##              (14.834785)
## dummy.l4      -0.078681 ***
##               (0.003602)
## vol.l4       -41.242889 **
##              (13.057660)
## dummy.l5      -0.087661 ***
##               (0.003686)
## vol.l5        -0.582973
##               (4.789688)
## dummy.l6      -0.096239 ***
##               (0.003870)
## vol.l6        38.336579
##              (20.988262)
## const          1.723741 ***
##               (0.039937)
## ---------------------------
## R^2            0.154811
## Adj. R^2       0.154260
## 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 = 142.73, 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)))
```

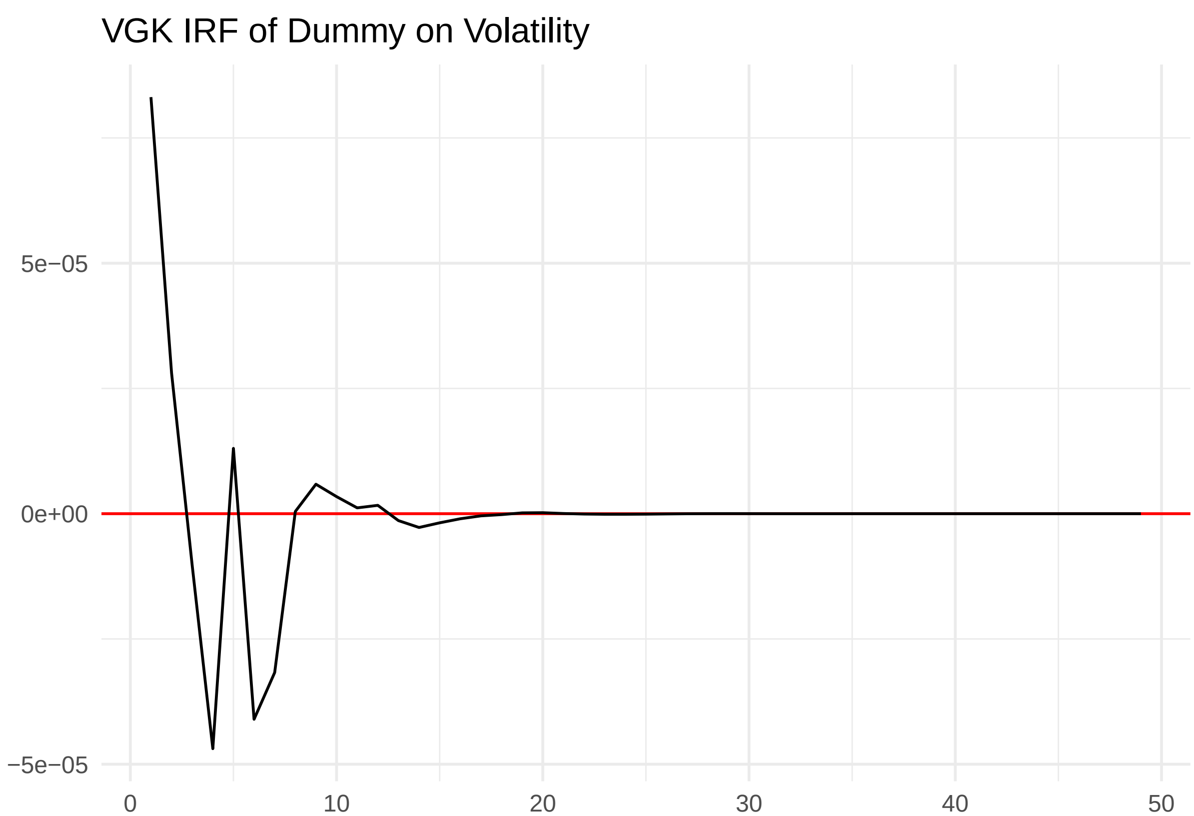
```
##              dummy          vol
## dummy 1.025400e+01 2.867427e-04 1.025400e+01 2.662416e-04
## vol   2.867427e-04 2.544169e-06 2.662416e-04 4.349722e-05
```

```r
B.hat
```

```
##              [,1]        [,2]
## [1,] 3.202186e+00 0.000000000
## [2,] 8.314371e-05 0.006594718
```

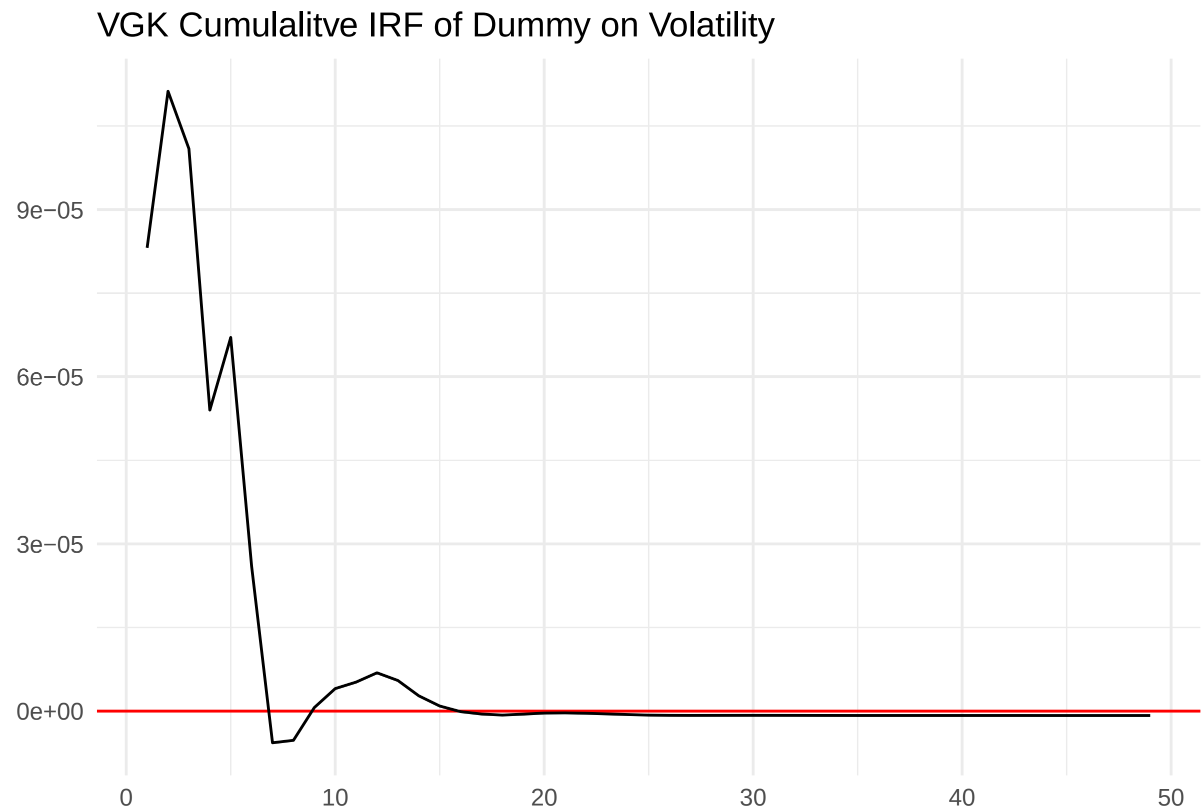
```r
nb.sim = 7*7
#get back the coefficient of est.VAR
phi <- Acoef(est.VAR)
PHI = make.PHI(phi)

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

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



Post Counts

```
y2 = cbind(Vdata$N, Vdata$VGK_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
##                (0.000001)
## vol.l1          0.257940 *
##                (0.110271)
## N.l2           -0.000002 ***
##                (0.000001)
## vol.l2          0.021295
##                (0.082209)
## N.l3           -0.000004 ***
##                (0.000000)
## vol.l3          0.028552
##                (0.014752)
## N.l4           -0.000001
##                (0.000002)
## vol.l4          0.037417 *
##                (0.016928)
## N.l5           -0.000004 ***
##                (0.000001)
## vol.l5          0.031707 *
##                (0.015054)
## N.l6           -0.000003 ***
##                (0.000000)
## vol.l6          0.057719 **
##                (0.020797)
## const           0.000257 ***
##                (0.000031)
## ---------------------------
## R^2             0.135605
## Adj. R^2        0.135041
## 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          -0.043228 ***
##               (0.003605)
## vol.l1       -40.154465
##              (32.166860)
## N.l2          -0.039217 ***
##               (0.004208)
## vol.l2        28.214645
##              (85.500685)
## N.l3          -0.026538 ***
##               (0.004254)
## vol.l3       -48.227617
##              (34.606128)
## N.l4          -0.024488 ***
##               (0.004981)
## vol.l4       -98.491114 **
##              (37.824331)
## N.l5          -0.040435 ***
##               (0.003975)
## vol.l5       -12.590645
##              (14.403769)
## N.l6          -0.044891 ***
##               (0.004744)
## vol.l6       124.796513 *
##              (56.161046)
## const          3.525550 ***
##               (0.098794)
## ----------------------------
## R^2            0.100656
## Adj. R^2       0.100070
## 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)))
```

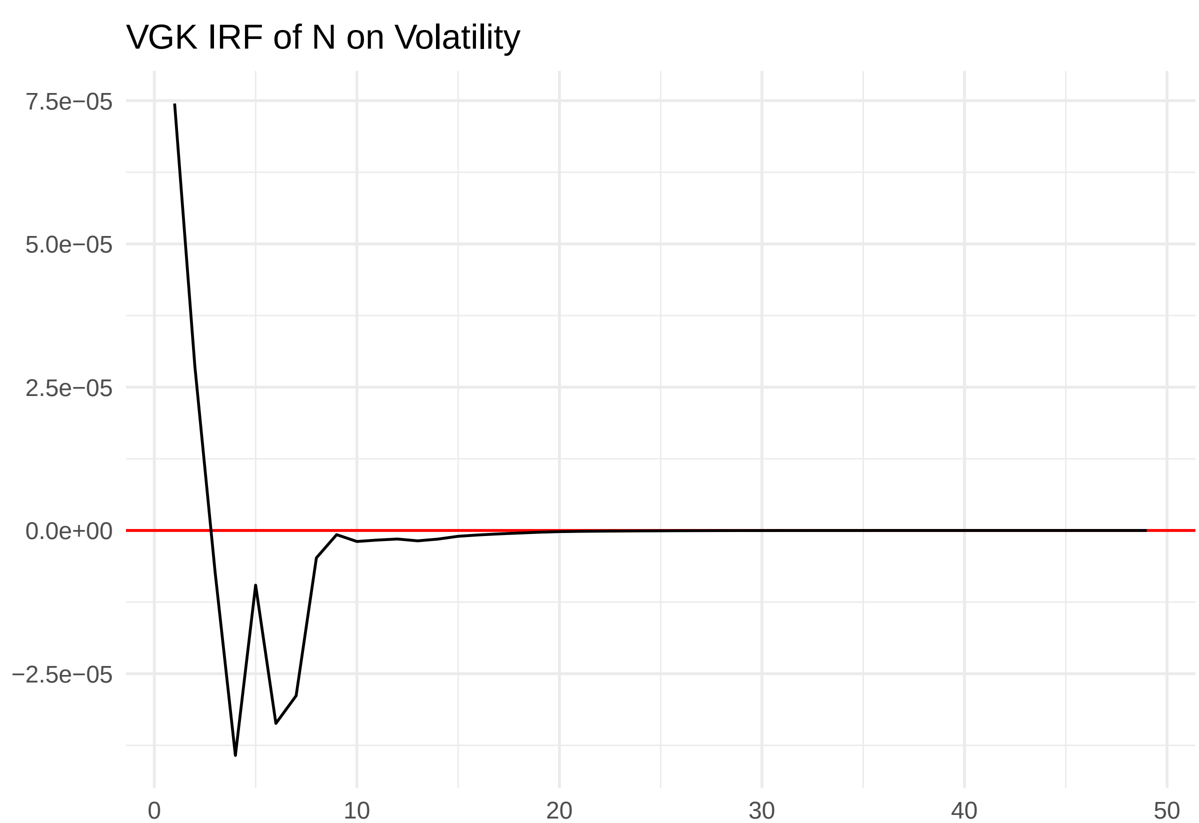
```
##                N          vol
## N   8.648668e+01 6.926668e-04 8.648668e+01 0.0006928754
## vol 6.926668e-04 2.546098e-06 6.928754e-04 0.0000154312
```

```r
B.hat2
```

```
##              [,1]       [,2]
## [1,] 9.299821e+00 0.00000000
## [2,] 7.450417e-05 0.00392755
```

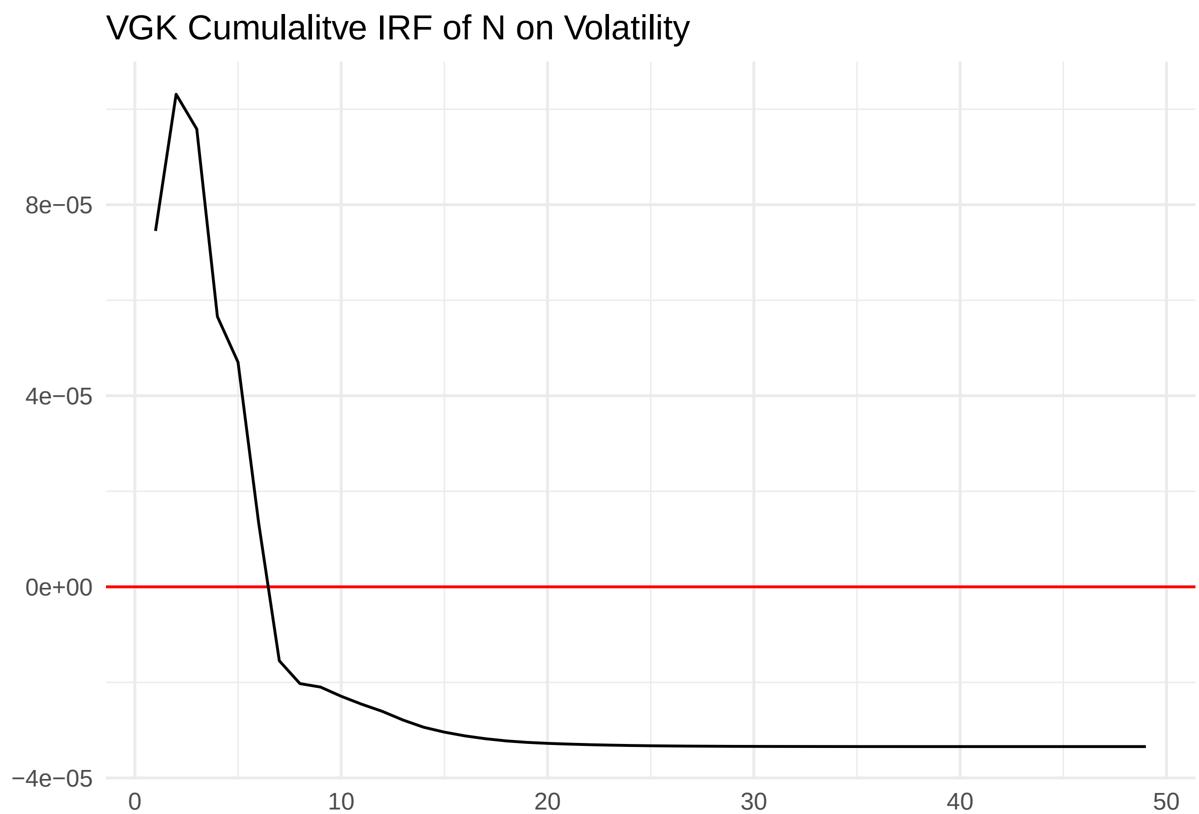
```r
#get back the coefficient of est.VAR
phi2 <- Acoef(est.VAR2)
PHI2 = make.PHI(phi2)

#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("VGK 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("VGK 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	4.34	0.000217

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

Res.Df	Df	F	Pr(>F)
2e+04			
2e+04	-6	3.33	0.00278

Trade Mention

```r
y4 = cbind(Vdata$trade, Vdata$VGK_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       0.000017
##               (0.000032)
## vol.l1         0.258921 *
##               (0.102476)
## trade.l2       0.000042
##               (0.000053)
## vol.l2         0.020907
##               (0.073217)
## trade.l3      -0.000068 ***
##               (0.000019)
## vol.l3         0.028060
##               (0.014485)
## trade.l4       0.000057
##               (0.000074)
## vol.l4         0.037660 *
##               (0.016099)
## trade.l5      -0.000061 *
##               (0.000024)
## vol.l5         0.030651 *
##               (0.015488)
```

```
## trade.l6      -0.000030 *
##               (0.000013)
## vol.l6         0.058531 **
##               (0.021326)
## const          0.000224 ***
##               (0.000030)
## ---------------------------
## R^2            0.134864
## Adj. R^2       0.134300
## 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       0.024487
##               (0.018152)
## vol.l1         5.014758
##               (3.431028)
## trade.l2       0.019537 *
##               (0.008624)
## vol.l2         4.508728
##               (3.232222)
## trade.l3       0.022995
##               (0.016467)
## vol.l3         3.804473
##               (4.626875)
## trade.l4       0.023363
```

```
##                (0.012889)
## vol.l4         -3.357488 *
##                (1.557762)
## trade.l5        0.018748
##                (0.010787)
## vol.l5         -0.710649
##                (0.720776)
## trade.l6        0.013491
##                (0.010359)
## vol.l6          1.274795
##                (1.680999)
## const           0.026319 ***
##                (0.002573)
## ---------------------------
## R^2             0.020099
## Adj. R^2        0.019460
## 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)))
```

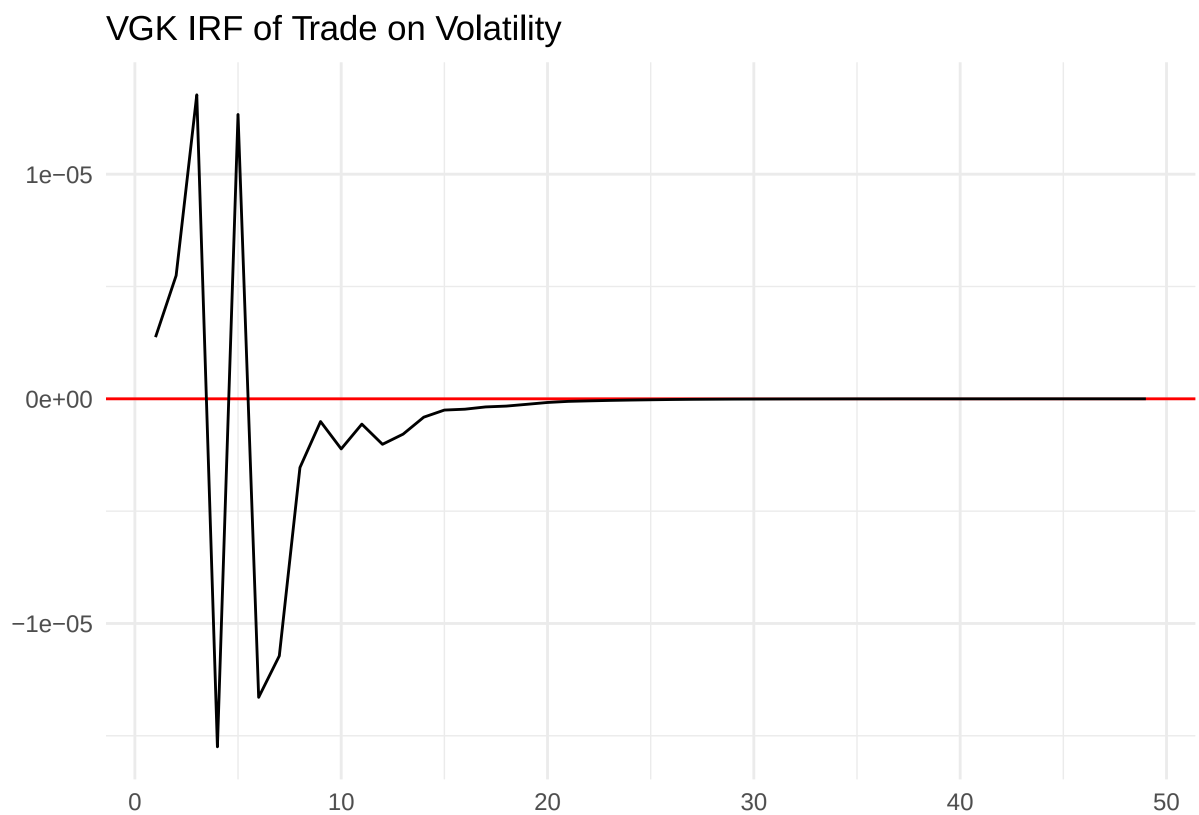
```
##                trade          vol
```

```
## trade  8.196895e-02 -5.800013e-07 8.196877e-02 7.859206e-07
## vol   -5.800013e-07  2.540930e-06 7.859206e-07 3.104957e-05
```

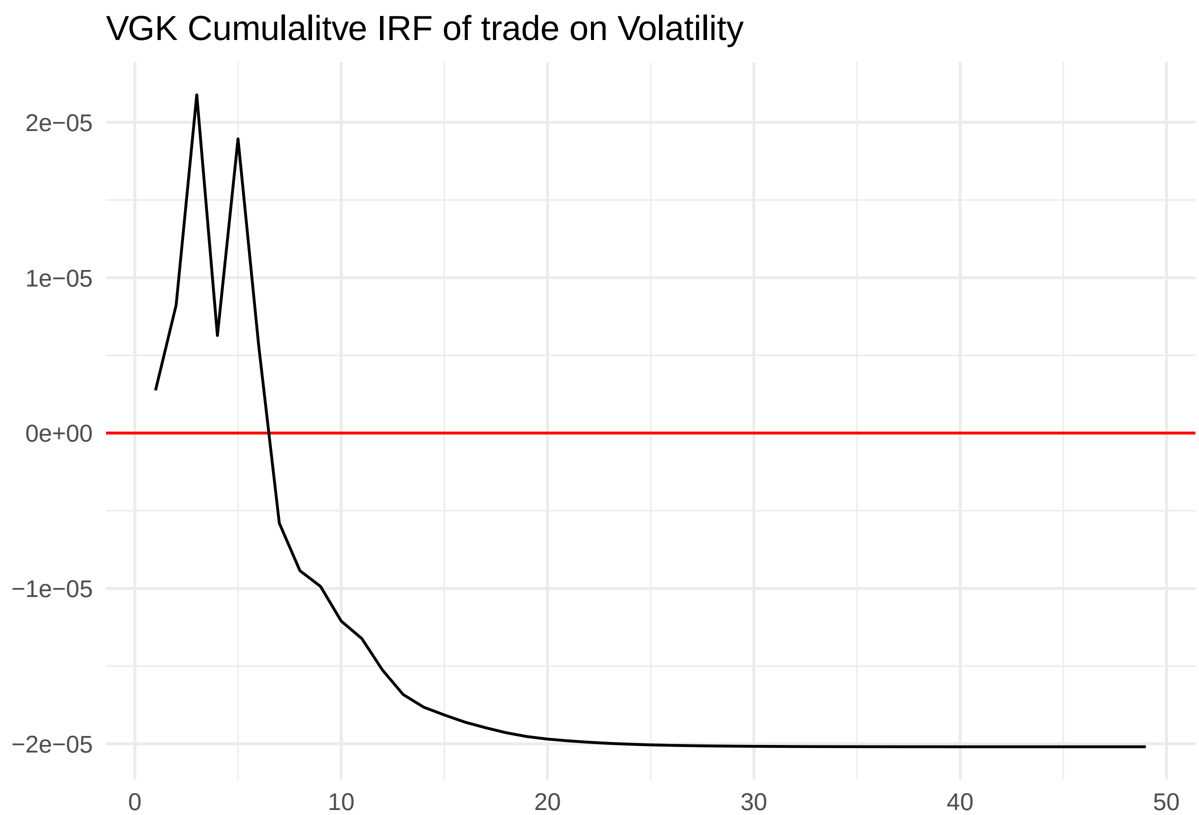
```
B.hat4
```

```
##              [,1]        [,2]
## [1,] 2.863019e-01 0.000000000
## [2,] 2.745077e-06 0.005572214
```

```
#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("VGK 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("VGK 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	4.39e-11

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

Res.Df	Df	F	Pr(>F)
2e+04			
2e+04	-6	1.49	0.176

China Mention

```r
ychina = cbind(Vdata$china, Vdata$VGK_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       0.000059
##               (0.000044)
## vol.l1         0.258298 *
##               (0.110895)
## china.l2       0.000016
##               (0.000029)
## vol.l2         0.021287
##               (0.079393)
## china.l3      -0.000040 ***
##               (0.000009)
## vol.l3         0.028374
##               (0.014559)
## china.l4      -0.000025 *
##               (0.000010)
## vol.l4         0.037333 *
##               (0.016614)
## china.l5      -0.000013
##               (0.000007)
## vol.l5         0.031020 *
##               (0.015136)
## china.l6       0.000018
##               (0.000026)
```

```
## vol.l6         0.058227 **
##               (0.021086)
## const          0.000221 ***
##               (0.000029)
## ---------------------------
## R^2            0.134904
## Adj. R^2       0.134340
## 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       0.075378 *
##               (0.034296)
## vol.l1         0.447042
##               (1.081793)
## china.l2       0.044727 *
##               (0.021565)
## vol.l2         5.167661
##               (4.406098)
## china.l3       0.007296
##               (0.010586)
## vol.l3         1.639544
##               (3.949289)
## china.l4       0.026235 **
##               (0.009875)
## vol.l4        -2.805233
```

```
##                (1.587109)
## china.l5        0.049036
##                (0.033968)
## vol.l5         -0.738524
##                (0.769105)
## china.l6        0.055222
##                (0.048484)
## vol.l6          1.939535
##                (1.325907)
## const           0.043275 ***
##                (0.005225)
## ---------------------------
## R^2             0.035057
## Adj. R^2        0.034428
## Num. obs.   19965
## ===========================
## *** p < 0.001; ** p < 0.01; * p < 0.05
```

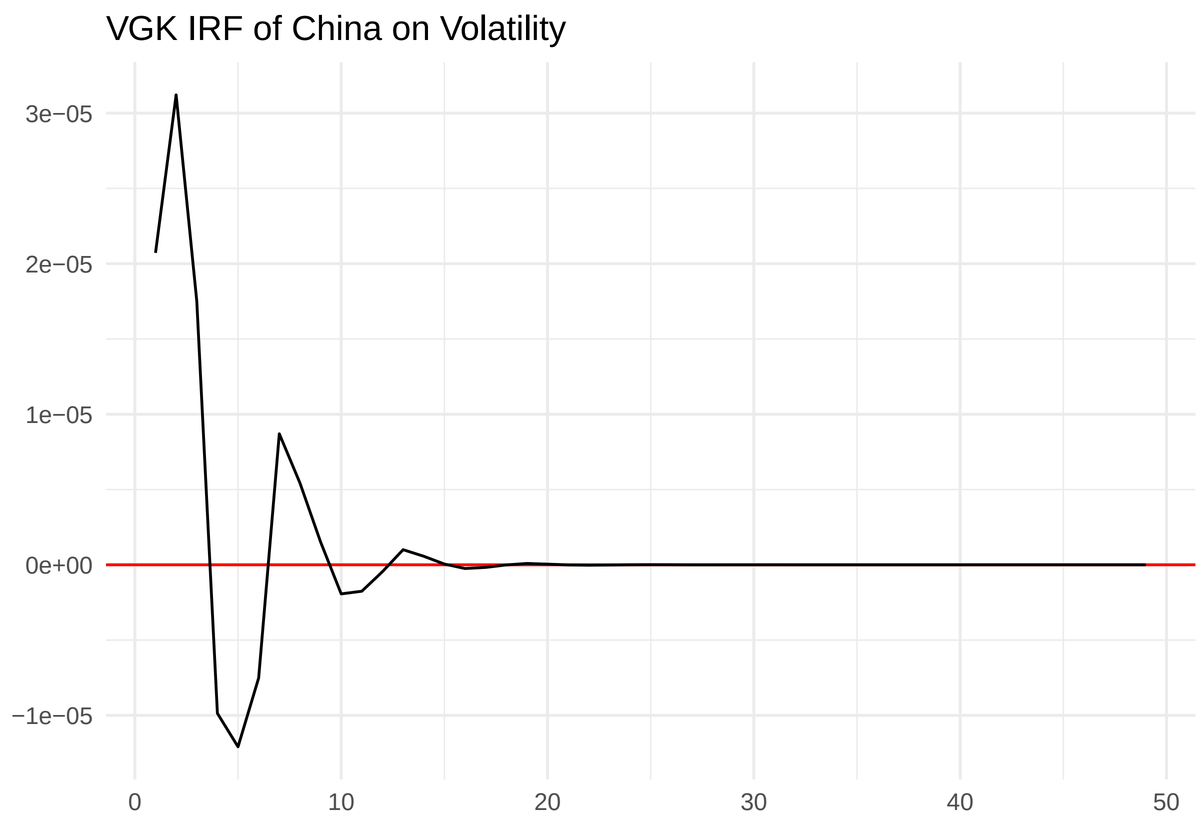
```r
#Recreate a Robust Omega Matrix
Uchina = residuals(est.VARchina)
Tchina = nrow(Uchina)
Omegachina = matrix(0, ncol(Uchina), ncol(Uchina))
for(l in 0:L) {
  weight = 1 - l/(L+1)
  Gamma_l_china = t(Uchina[(l+1):Tchina, , drop=FALSE]) %*% Uchina[1:(Tchina-l), , drop=FALSE] /Tchina
  if (l == 0){
    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 1.938785e-01 9.188264e-06 1.938778e-01 9.121855e-06
## vol   9.188264e-06 2.540775e-06 9.121855e-06 2.099700e-05
```

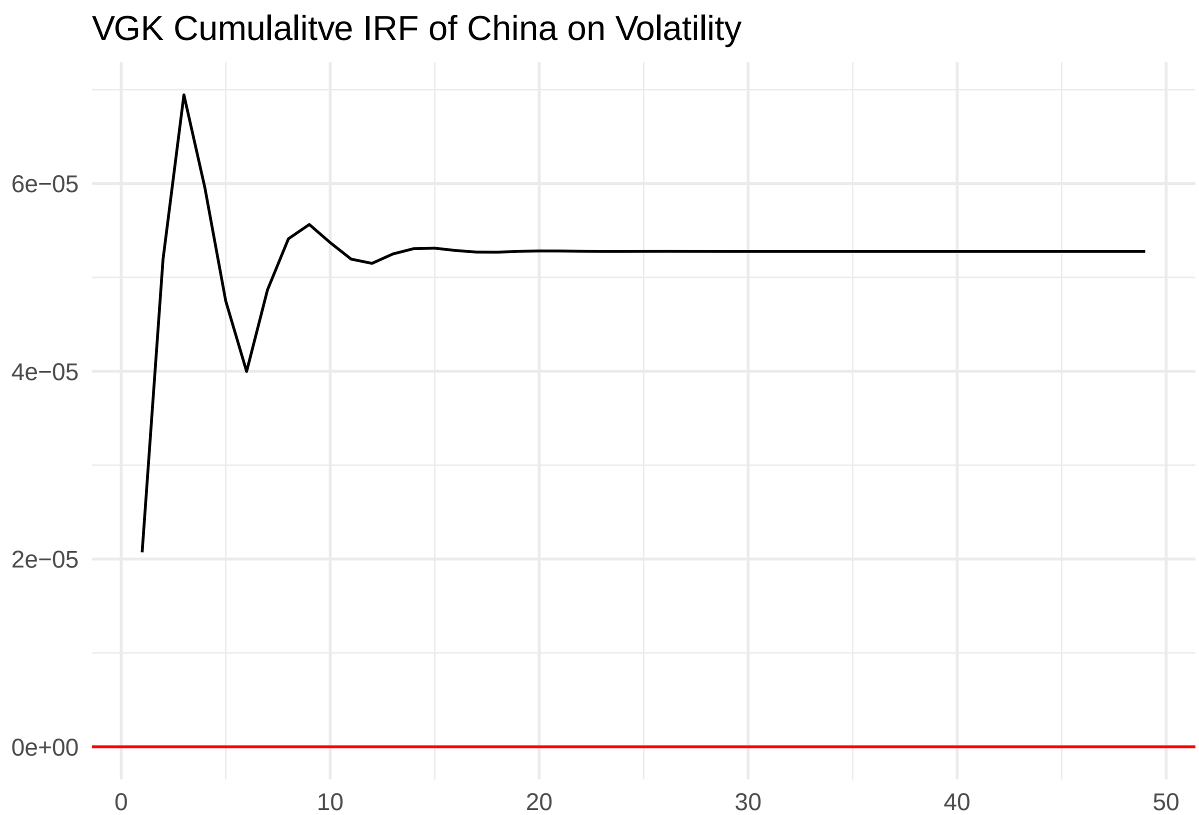
```
B.hatchina
```

```
##              [,1]        [,2]
## [1,] 4.403156e-01 0.000000000
## [2,] 2.071663e-05 0.004582201
```

```
#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("VGK 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("VGK 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	2.01	0.0609

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

Res.Df	Df	F	Pr(>F)
2e+04			
2e+04	-6	1.65	0.13

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.

```
# 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

```
y_f_d = cbind(Vdata_f$dummy, Vdata_f$VGK_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 0.000005
(0.000003)
vol.l1 0.098328 *
(0.049538)
dummy.l2 0.000003
(0.000003)
vol.l2 0.062420 ** (0.022012)
dummy.l3 -0.000011 ***(0.000003)***
vol.l3 0.072057 *(0.024898)*
dummy.l4 0.000026
(0.000034)
vol.l4 0.076567
(0.031207)

dummy.l5 -0.000012 ***(0.000003)***
vol.l5 0.071082 *(0.022446)*
dummy.l6 -0.000003
(0.000003)
vol.l6 0.078200 ***(0.026858)***
const 0.000174 (0.000023)
——————— R^2 0.097187
Adj. R^2 0.095515
Num. obs. 7036
============================ *** p < 0.001; ** p < 0.01; * p < 0.05

```
#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     -0.128418 ***
##              (0.005863)
## vol.l1       12.992441
##             (10.995170)
## dummy.l2     -0.111603 ***
##              (0.005704)
## vol.l2       -7.474749
##             (16.399095)
## dummy.l3     -0.098654 ***
##              (0.006073)
## vol.l3       31.638078
##             (24.675166)
## dummy.l4     -0.095999 ***
##              (0.006159)
```

```
## vol.l4       -18.773938
##               (9.861811)
## dummy.l5      -0.112249 ***
##               (0.006464)
## vol.l5        24.121246
##              (21.402557)
## dummy.l6      -0.129537 ***
##               (0.006945)
## vol.l6        35.169932
##              (28.858610)
## const          1.968056 ***
##               (0.059379)
## --------------------------
## R^2            0.182533
## Adj. R^2       0.181020
## 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)))
```

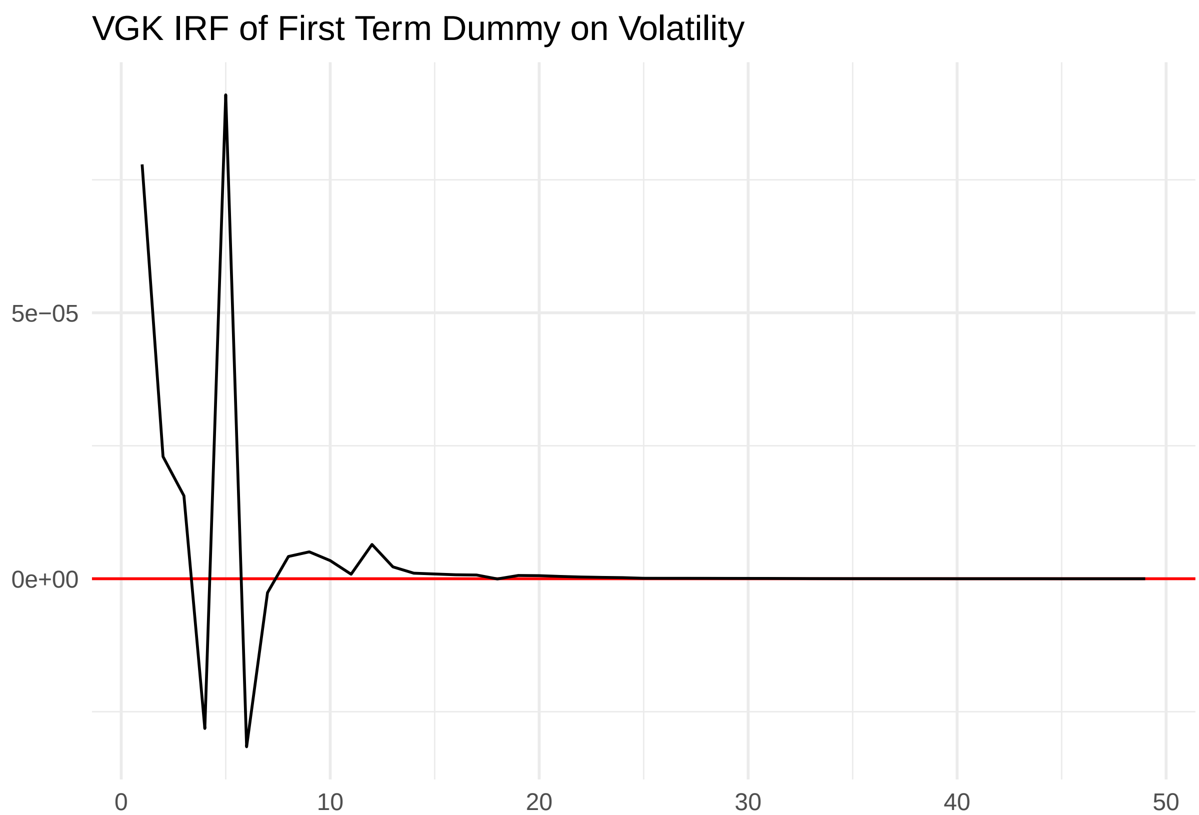
```
##             dummy          vol
## dummy 9.965798044 2.459490e-04 9.9657972053 2.459287e-04
```

```
## vol   0.000245949 2.476707e-06 0.0002459287 3.159322e-05
```

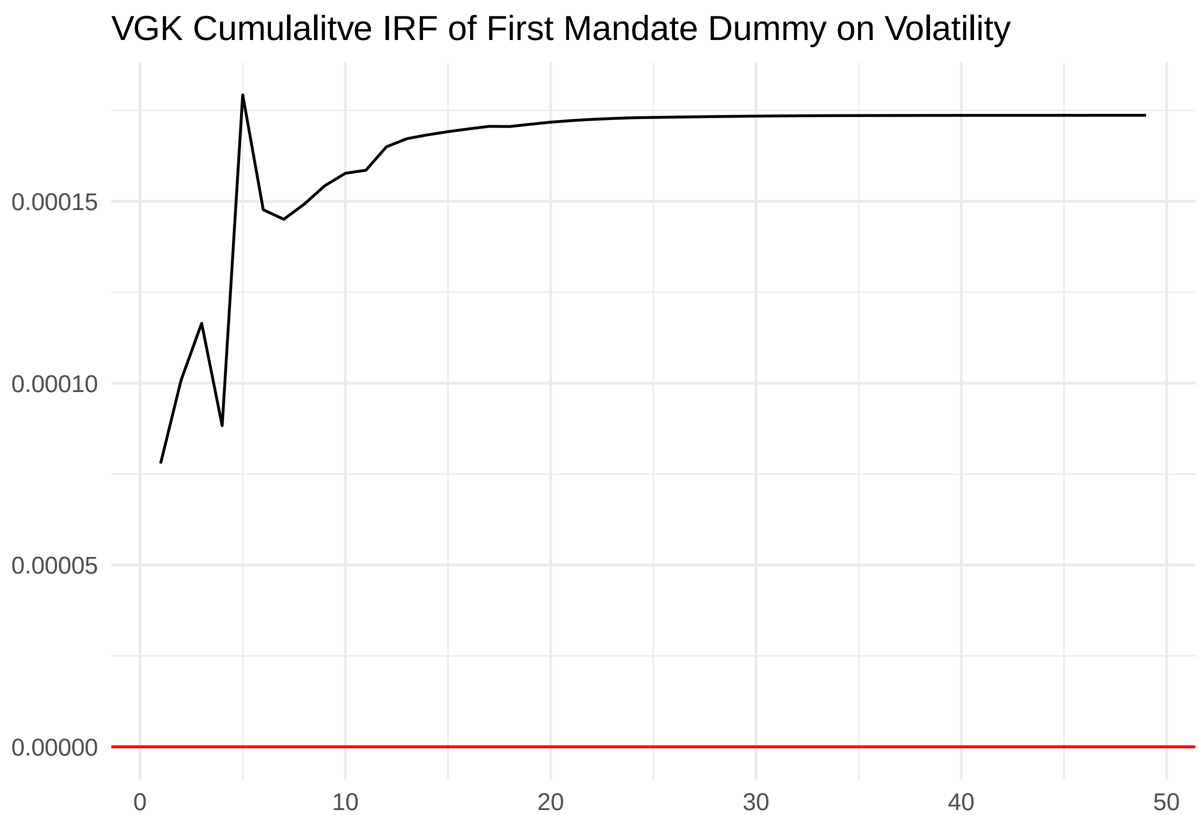
```r
B.hat_f_d
```

```
##               [,1]        [,2]
## [1,] 3.156865e+00 0.000000000
## [2,] 7.790281e-05 0.005620244
```

```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("VGK 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("VGK 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	1.48	0.182

```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	4.13	0.000377

Second Mandate

```r
y_s_d = cbind(Vdata_s$dummy, Vdata_s$VGK_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 0.000043
(0.000075)
vol.l1 0.191496 ** (0.062051)
dummy.l2 -0.000027
(0.000015)
vol.l2 0.046385 ** (0.015184)
dummy.l3 -0.000045 ** (0.000016)
vol.l3 0.052099 (0.011029)
dummy.l4 0.000040
(0.000063)
vol.l4 0.055336
(0.022999)
dummy.l5 -0.000068 *
(0.000032)
vol.l5 0.055025 (0.016348)
dummy.l6 -0.000056
(0.000032)
vol.l6 0.116787 (0.039389)
const 0.000694 *
(0.000271)
———————- R^2 0.153676
Adj. R^2 0.131627
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    -0.216558 ***
##             (0.020948)
## vol.l1     -16.008615
##             (9.280998)
## dummy.l2    -0.206873 ***
##             (0.020761)
## vol.l2     -16.232847
##             (9.132715)
## dummy.l3    -0.205327 ***
##             (0.021366)
## vol.l3      35.821715 ***
##             (9.195995)
## dummy.l4    -0.218116 ***
##             (0.021928)
## vol.l4     -14.060199 **
##             (5.242811)
## dummy.l5    -0.214649 ***
##             (0.021628)
## vol.l5      -8.193010
##             (6.221849)
## dummy.l6    -0.199750 ***
##             (0.022321)
## vol.l6      10.778416
##            (12.753512)
## const        3.143910 ***
```

```
##             (0.253068)
## -------------------------
## R^2          0.298415
## Adj. R^2     0.280137
## Num. obs.  512
## =========================
## *** p < 0.001; ** p < 0.01; * p < 0.05
```

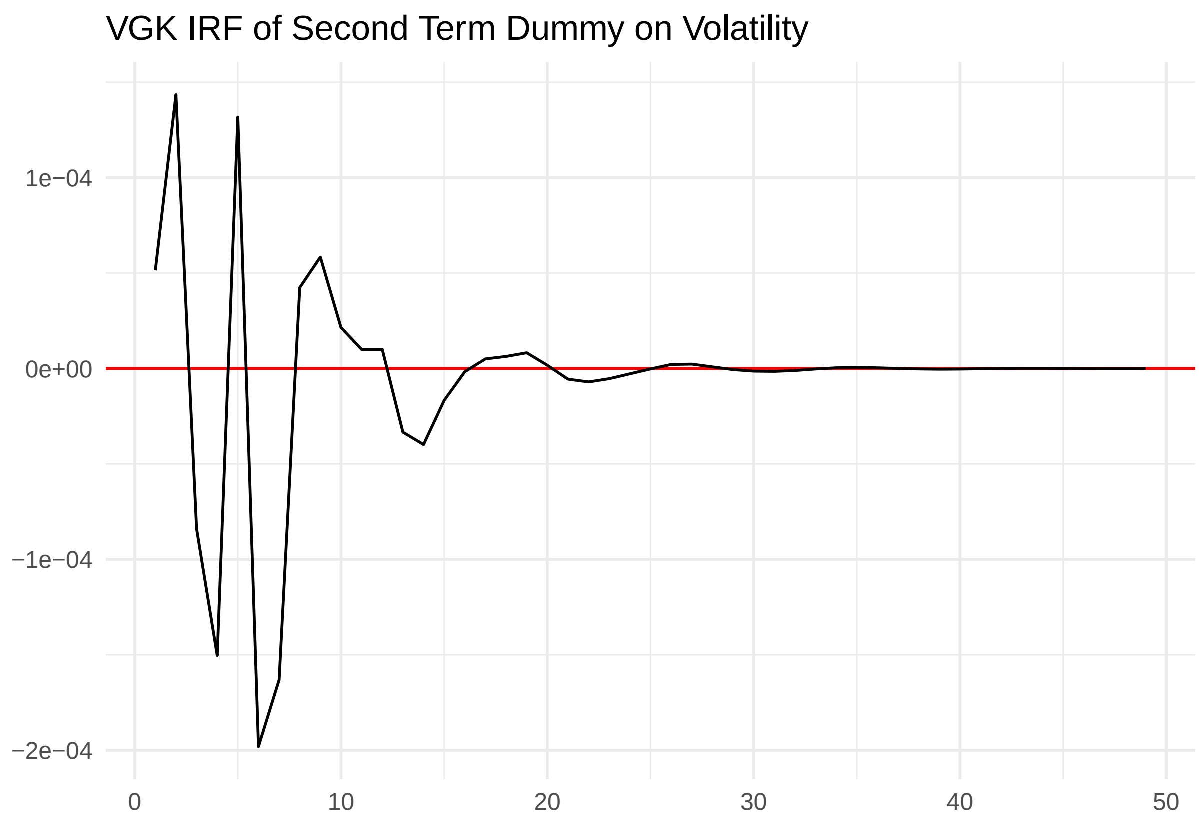
```
#Construct the Robust Omega Matrix
U_s_d = residuals(est.VAR_s_d)
T_s_d = nrow(U_s_d)
Omega_s_d = matrix(0, ncol(U_s_d), ncol(U_s_d))
for(l in 0:L) {
  weight = 1 - l/(L+1)
  Gamma_l__s_d = t(U_s_d[(l+1):T_s_d, , drop=FALSE]) %*% U_s_d[1:(T_s_d-l), , drop=FALSE] /T_s_d
  if (l == 0){
    Omega_s_d = 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 9.7948153938 1.576117e-04 9.7948166075 1.609906e-04
## vol   0.0001576117 1.588877e-05 0.0001609906 4.399124e-05
```

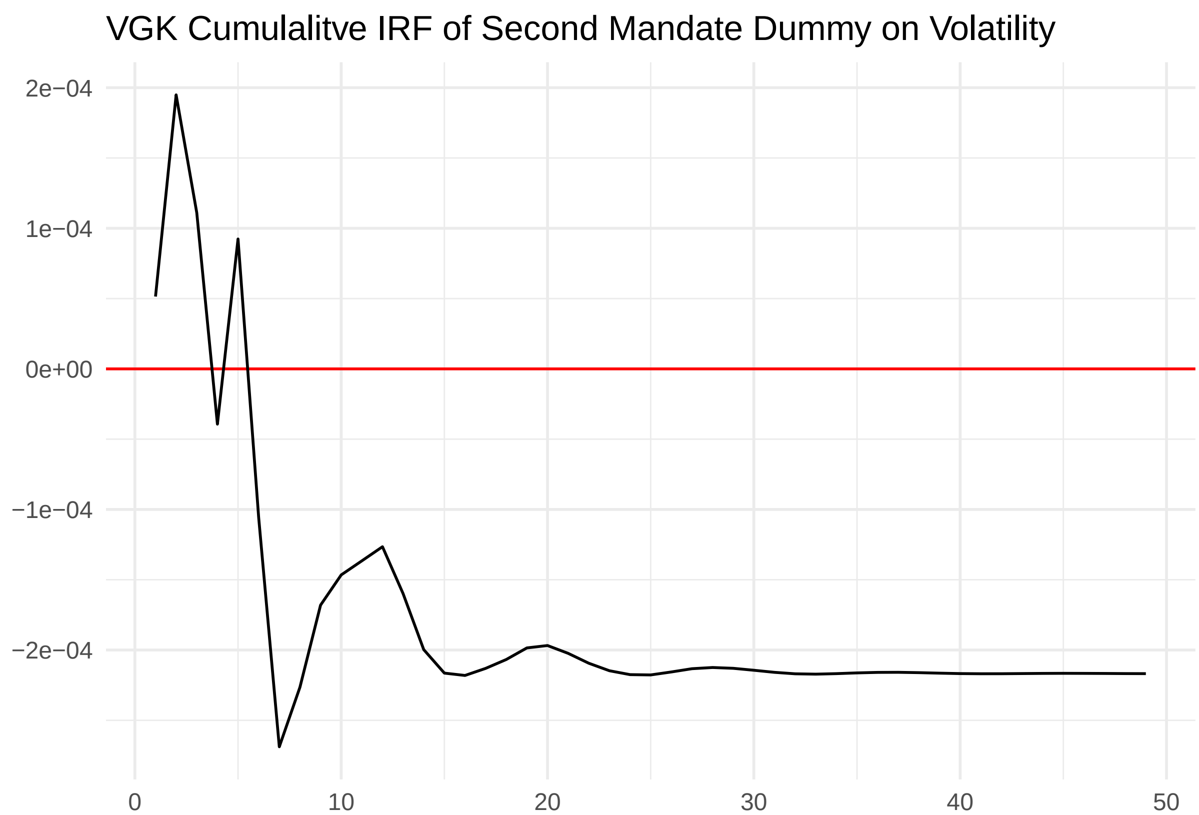
```
B.hat_s_d
```

```
##              [,1]       [,2]
## [1,] 3.129667e+00 0.00000000
## [2,] 5.144017e-05 0.00663239
```

```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("VGK IRF of Second Term Dummy on Volatility") +
  ylab("")+
  xlab("") +
  theme_minimal()
```




```r
ggplot(Yd_s_d,aes(x=period, y=cumsum(response))) +
  geom_hline(yintercept = 0, color="red") +
  geom_line() +
  theme_light() +
  ggtitle("VGK 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	0.38	0.892

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

Res.Df	Df	F	Pr(>F)
499			
505	-6	0.572	0.752