ARMA-X Analysis Tutorial

Contents

Setup

Load Libraries & 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(sandwich) #regression errors
require(stargazer) #nice reg tables
require(tidytext) #text mining
require(textstem) #lemmatization
require(quanteda) #tokenization
require(texreg) #arima tables
require(future.apply) #parallel computation (speed)
require(aTSA) #adf test

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"))
```

Load Data

Use the full_data script to load our final dataset. Then select the timeframe.

Stationarity

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

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

Univariate ARMA-X Models

Custom ARMA-X Specification

This first function allows to run a certain specification of an ARMA-X model. We needed to create this function due to the creation of the lags for the exogenous regressor.

```r
#armax enables a custom armax specification with p,q,r
res2 = armax(data$SPY_vol, xreg=data$dummy, nb.lags=2,
                   p=5, q=0, d=0, latex=F)
```

================================== Model 1
——————————————- ar1 0.3576 ***(0.0071)***
ar2 0.0416 (0.0075)
ar3 0.0994 ***(0.0074)***
ar4 0.1045 (0.0075)
ar5 0.0816 ***(0.0071)***
intercept 0.0199 (0.0018)
dummy_lag_0 0.0015 ***(0.0002)***
dummy_lag_1 0.0009 (0.0002)
dummy_lag_2 0.0001
(0.0002)
——————————————- AIC -44706.1942
AICc -44706.1832
BIC -44627.1749
Log Likelihood 22363.0971
Num. obs. 19969
================================== *** $p < 0.001$; ** $p < 0.01$; * $p < 0.05$

Best ARMA-X Given r

This second function uses the base auto.arima function in order to look for the best (lowest AIC) specification given a certain number of x lags. Once again it was necessary to create a custom function for this due to the creation of the lags for the exogenous regressor.

```r
#auto.armax selects the lowest AIC value given r (exogenous variable lags)
res1 = auto.armax(data$SPY_vol,xreg=data$dummy,nb.lags=2,
                latex=F,max.p = 6, max.q = 6, max.d=0)
```

================================= Model 1
—————————————- ar1 0.9828 ***(0.0017)***
ma1 -0.6786 (0.0073)
ma2 -0.2118 ***(0.0087)***
ma3 -0.0120
(0.0080)
ma4 0.0331 (0.0071)
intercept 0.0202 ***(0.0041)***
dummy_lag_0 0.0013 (0.0002)
dummy_lag_1 0.0007 ***(0.0002)***
dummy_lag_2 0.0001
(0.0002)
——————————— AIC -45719.7236
AICc -45719.7126
BIC -45640.7043
Log Likelihood 22869.8618
Num. obs. 19969
================================= p < 0.001; ** p < 0.01; * p < 0.05

Best ARMA-X

This third function is the most sophisticated. We've had to essentially custom code the auto.arima function in order to loop for p, q and r. It also provides many useful details such as a plot of the AIC value for the different number of lags. Note that the output of this function is a list, so in order get the model output, it is needed to specifically call it (see how the call works for this third model in the IRF section). The latex option in all functions determines whether the output is console-friendly, or output-friendly.

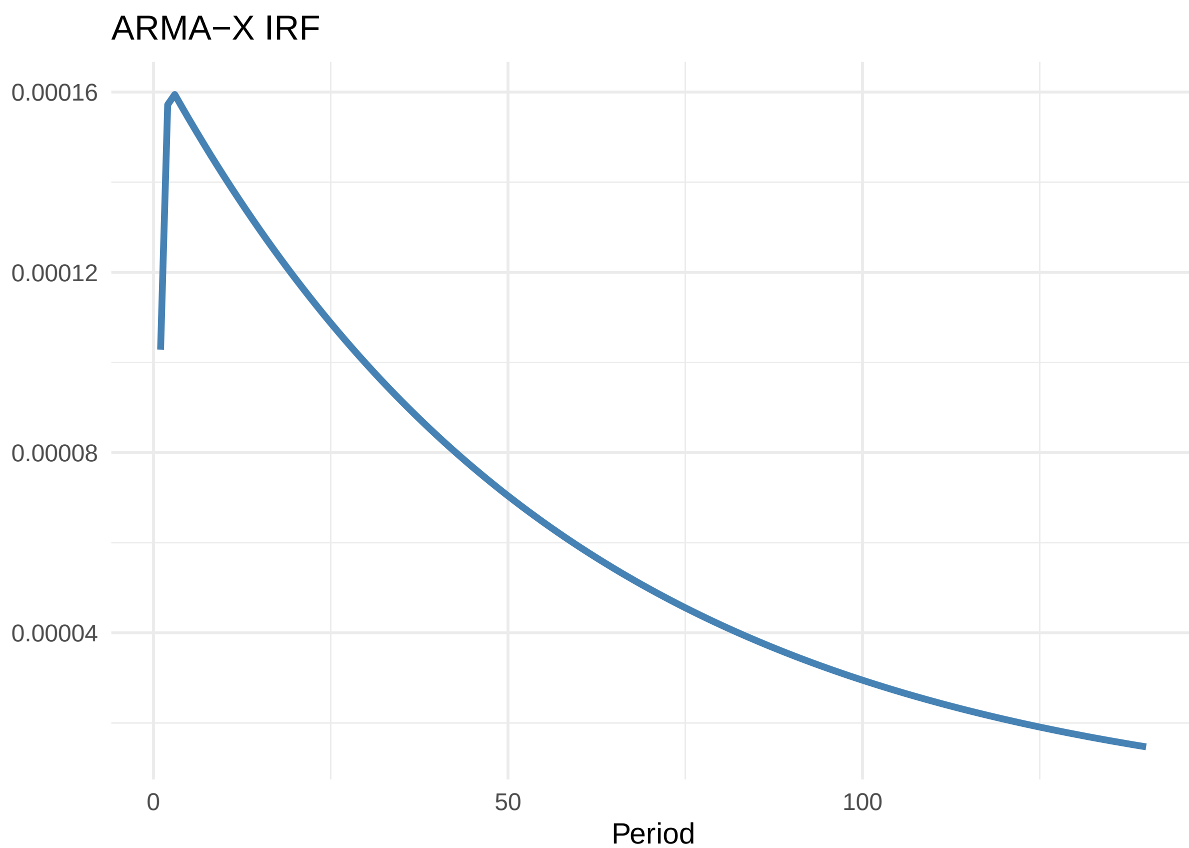
```
#auto.armax.r selects the lowest AIC checking all 3 p,q,r values
res3 = auto.armax.r(data$SPY_vol, x=data$dummy,
                max_p = 3, max_q = 3, max_r = 3, criterion = "AIC", latex=F)
```

================================= Model 1
—————————————- ar1 0.0300
(0.0510)
ar2 0.7229 ***(0.0397)***
ar3 0.2110 (0.0287)
ma1 0.2751 ***(0.0496)***
ma2 -0.6445 (0.0284)
ma3 -0.3527 ***(0.0256)***
intercept 0.0202 (0.0042)
dummy_lag_0 0.0014 ***(0.0002)***
dummy_lag_1 0.0008 (0.0002)
—————————————- AIC -45761.2161
AICc -45761.2051
BIC -45682.1963
Log Likelihood 22890.6081
Num. obs. 19970
================================= *** p < 0.001; ** p < 0.01; * p < 0.05

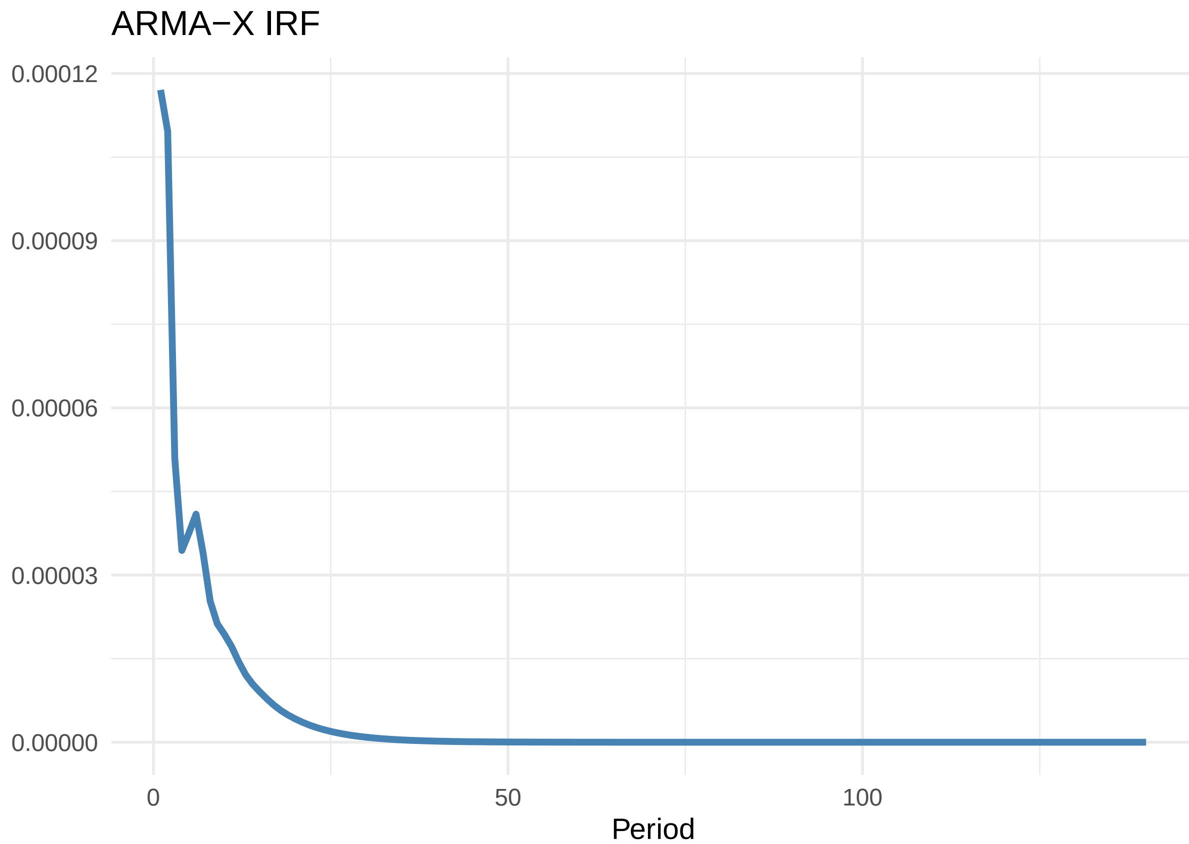
IRF

Finally, we have our personal twist on the IRF plot function from the AEC package. It enables automatic extraction of the phi's and beta's of a fitted ARMA-X model. It then uses the provided sim.arma function where it inputs the beta's as theta's. Finally, it automatically plots it, using a fancy ggplot.

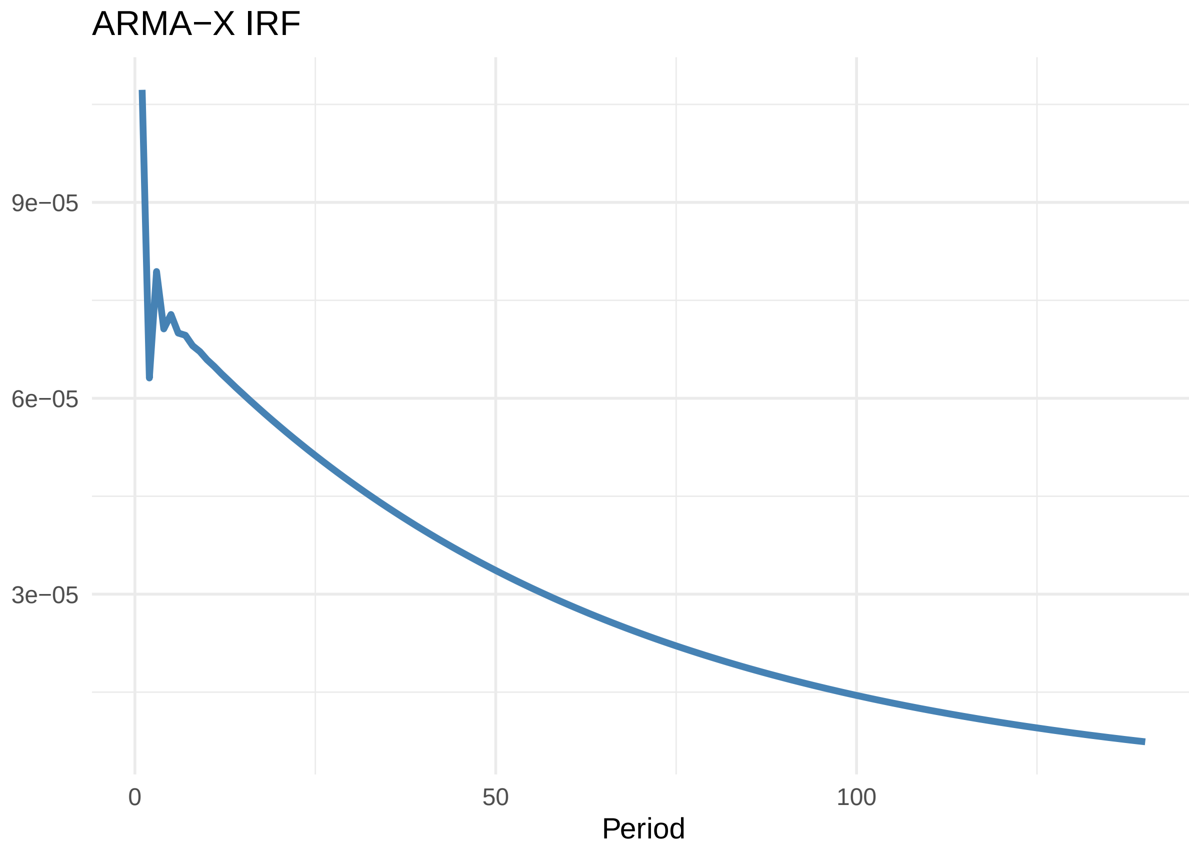
```
#we want to plot the IRFs of these models
nb.periods = 7 * 20
irf.plot(res1,nb.periods)
```




```
irf.plot(res2,nb.periods)
```




```
irf.plot(res3$model,nb.periods)
```


ARMA−X IRF
9e−05
6e−05
3e−05
0
50
100
Period