ARMA-X Analysis

Contents

Data

Raw Data

```r
# 1. Political

#truthsocial
raw_truths <- read.csv(here("data/political_data", "truths_new.csv"))

#twitter
raw_tweets <- read.csv(here("data/political_data", "tweets.csv"))


# 2. Financial

#S&P500
data_loader(symbol="SPY")

#STOXX50
data_loader(symbol="VGK")

#CSI 300 (China)
data_loader(symbol="ASHR")
```

Tweet Cleanup & Count

```r
tweets = raw_tweets

#only keep original Tweets
tweets <- tweets %>% filter(isRetweet != "t")
tokens <- tokens(tweets$text)
dfm <- dfm(tokens)

#cleanup
tweets = as.data.table(tweets)
names(tweets)[names(tweets) == 'date'] <- 'timestamp'
tweets <- tweets[order(tweets$timestamp, decreasing=T), ]
tweets$timestamp = as.POSIXct(tweets$timestamp,format = "%Y-%m-%d %H:%M:%S")

#count by hour
tweet_count = tweets[, .N, by=.(year(timestamp), month(timestamp),
                                day(timestamp), hour(timestamp))]

#fix timestamp
tweet_count$timestamp = as.POSIXct(sprintf("%04d-%02d-%02d %02d:00:00",
                         tweet_count$year, tweet_count$month, tweet_count$day,
                         tweet_count$hour), format = "%Y-%m-%d %H:00:00")

#remove useless columns and reorder by oldest first
tweet_count = select(tweet_count, timestamp, N)
tweet_count = tweet_count[ order(tweet_count$timestamp , decreasing = F ),]
```

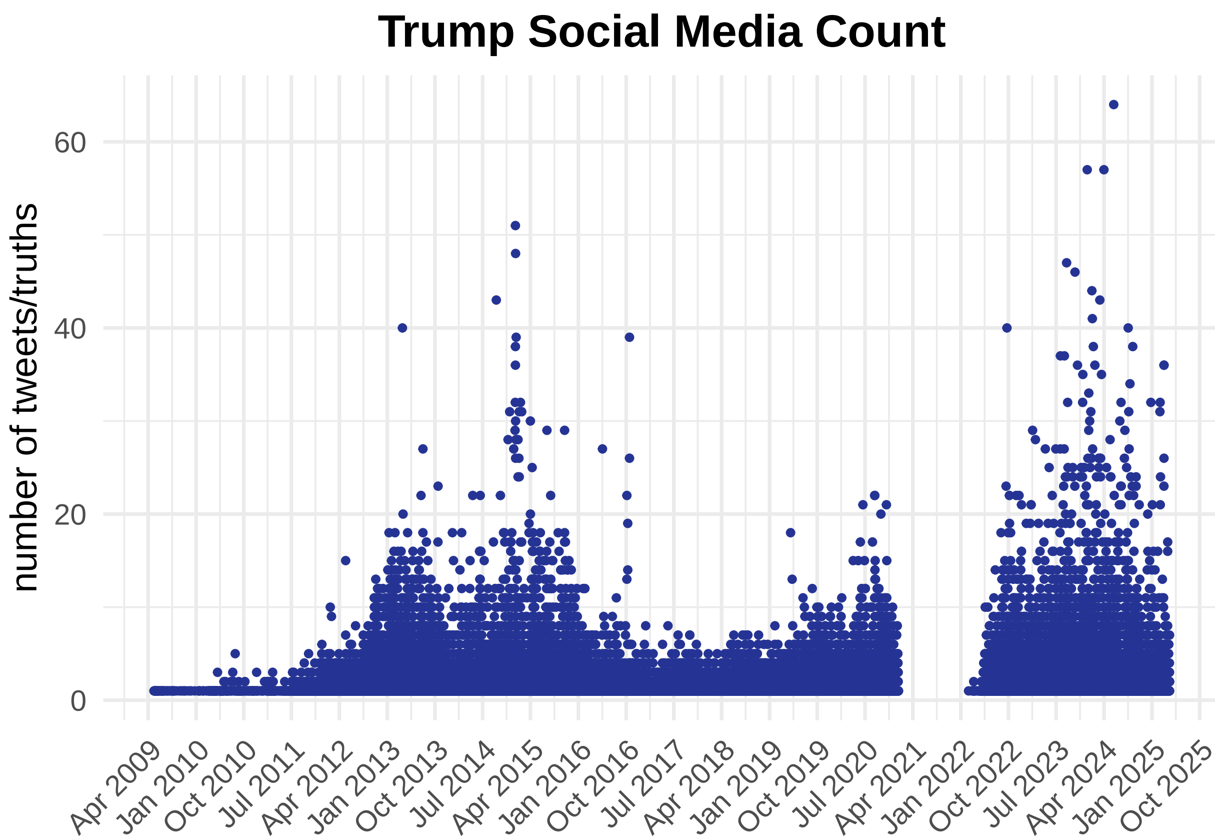
Truths Cleanup & Count

```r
truthsbackup <- truths_processer(raw_truths)

#cleanup
truths = as.data.table(truthsbackup)
names(truths)[names(truths) == 'date_time_parsed'] <- 'timestamp'
truths <- truths[order(truths$timestamp, decreasing=T), ]

#count by hour
truth_count = truths[, .N, by=.(year(timestamp), month(timestamp),
                                day(timestamp), hour(timestamp))]

#fix timestamp
truth_count$timestamp = as.POSIXct(sprintf("%04d-%02d-%02d %02d:00:00",
                         truth_count$year, truth_count$month, truth_count$day,
                         truth_count$hour), format = "%Y-%m-%d %H:00:00")

#remove useless columns and reorder by oldest first
truth_count = select(truth_count, timestamp, N)
truth_count = truth_count[ order(truth_count$timestamp , decreasing = F ),]
```

Tweets & Truths Merge

```r
tt_count = rbind(tweet_count,truth_count) #tweets & truths count
tt_count = tt_count %>% mutate(dummy = if_else(N > 0, 1, 0))

ggplot(tt_count, aes(x = timestamp, y = N)) +
    geom_point(color = "#253494", size = 1) +
    scale_x_datetime(date_labels = "%b %Y", date_breaks = "9 month") +
    labs(title = "Trump Social Media Count",
         x = NULL,
         y = "number of tweets/truths") +
    theme_minimal(base_size = 14) +
    theme(axis.text.x = element_text(angle = 45, hjust = 1),
          plot.title = element_text(face = "bold", hjust = 0.5))
```

```
## Warning: Removed 1 row containing missing values or values outside the scale range
## (`geom_point()`).
```



Volatility - Daily

```
#find daily volatility
SPY_dvolatility_alltime = r.vol_daily(raw_SPY,merge=F)

#select time period
SPY_dvolatility = filter(SPY_dvolatility_alltime,
                  between(timestamp, as.Date('2019-01-01'), as.Date('2025-04-10')))
colnames(SPY_dvolatility)[1] <- "timestamp_day"
tt_count_d = filter(tt_count,
                  between(timestamp, as.Date('2019-01-01'), as.Date('2025-04-10')))
```

Volatility - Hourly

```
#find hourly volatility
#NOTE: this ignores tweets made outside trading hours!!
SPY_hvolatility_alltime = r.vol_hourly(raw_SPY,merge=F)

#select time period
SPY_hvolatility = filter(SPY_hvolatility_alltime,
                  between(timestamp, as.Date('2019-01-01'), as.Date('2025-04-10')))
colnames(SPY_hvolatility)[1] <- "timestamp_hour"
```

```r
tt_count_h = filter(tt_count,
                    between(timestamp, as.Date('2019-01-01'), as.Date('2025-04-10')))
```

ARMA-X Models

Tweet Count on Daily Volatility

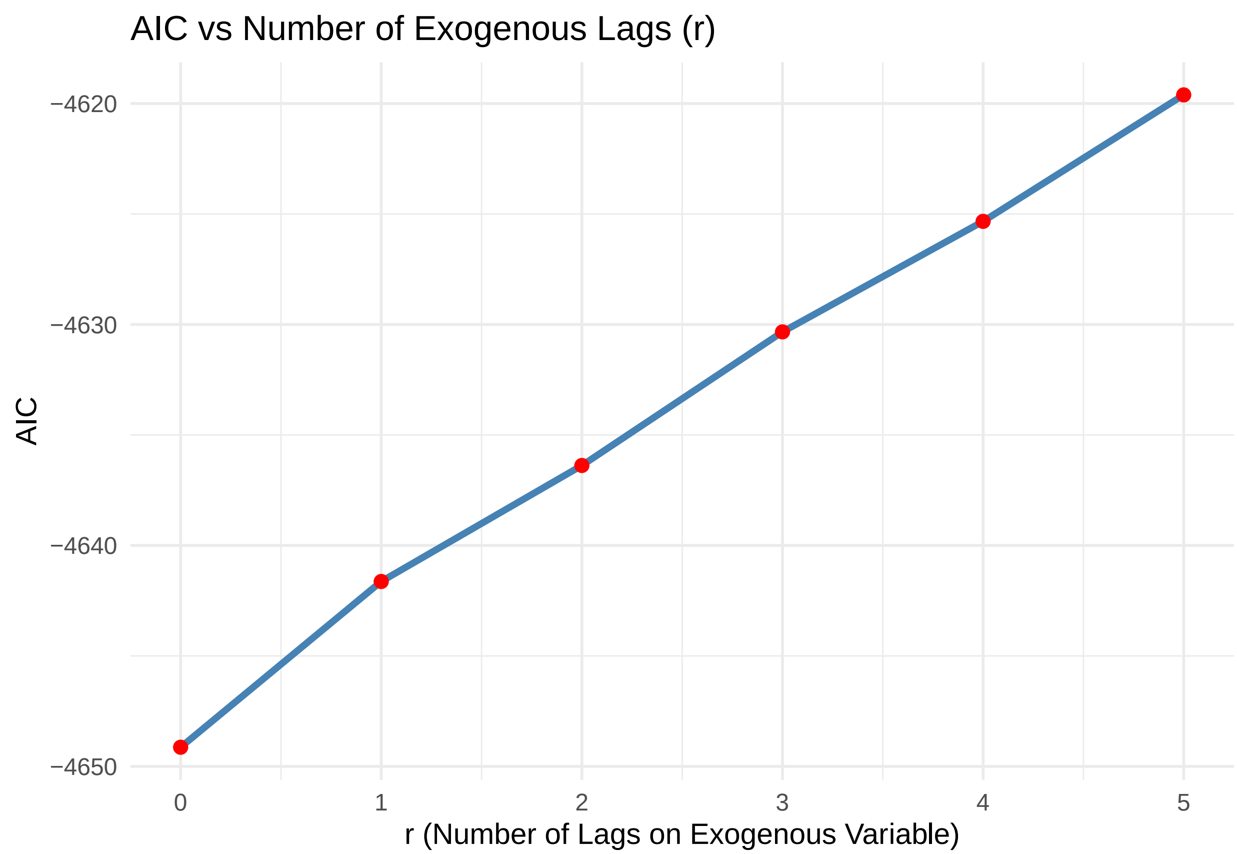
```r
#take all relevant data for armax
countvol_day = merge(SPY_dvolatility, tt_count_d, by.x = "timestamp_day",
                     by.y = "timestamp", all.x = T)

#NA tweets means no tweets
countvol_day$N[is.na(countvol_day$N)] = 0

#find best armax model and fit
armax_dayfit <- select_armax(countvol_day$r_vol_d, countvol_day$N,
                             max_p = 5, max_q = 5, max_r = 5, criterion = "AIC")
```

```
## Warning: Using `size` aesthetic for lines was deprecated in ggplot2 3.4.0.
## i Please use `linewidth` instead.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```

```r
summary(armax_dayfit$model)
```

```
## Series: y_trimmed
## Regression with ARIMA(5,0,3) errors
##
## Coefficients:
##          ar1     ar2      ar3     ar4     ar5     ma1     ma2     ma3
##       0.2603  0.6186  -0.8849  0.1840  0.6797  0.0074  0.0553  0.9339
## s.e.  0.0354  0.0469   0.0550  0.0431  0.0343  0.0338  0.0352  0.0443
##       intercept   Lag_0
##          0.0466  -2e-04
## s.e.     0.0196   6e-04
##
## sigma^2 = 0.003046:  log likelihood = 2335.57
## AIC=-4649.13   AICc=-4648.96   BIC=-4590.13
##
## Training set error measures:
##                        ME       RMSE        MAE       MPE     MAPE      MASE
## Training set 0.0001424625 0.05501887 0.01655447 -49.51062 75.68609 0.9401725
##                     ACF1
## Training set 0.003902538
```

```r
armax_dayfit$ICplot
```




```
armax_dayfit$params
```

```
## $p
## [1] 5
##
## $q
## [1] 3
##
## $r
## [1] 0
```

Tweet Dummy on Daily Volatility

```
#NA tweets means no tweets
countvol_day$dummy[is.na(countvol_day$dummy)] = 0

#find best armax model and fit
armax_dayfit <- select_armax(countvol_day$r_vol_d, countvol_day$dummy,
                             max_p = 5, max_q = 5, max_r = 5, criterion = "AIC")

summary(armax_dayfit$model)
```

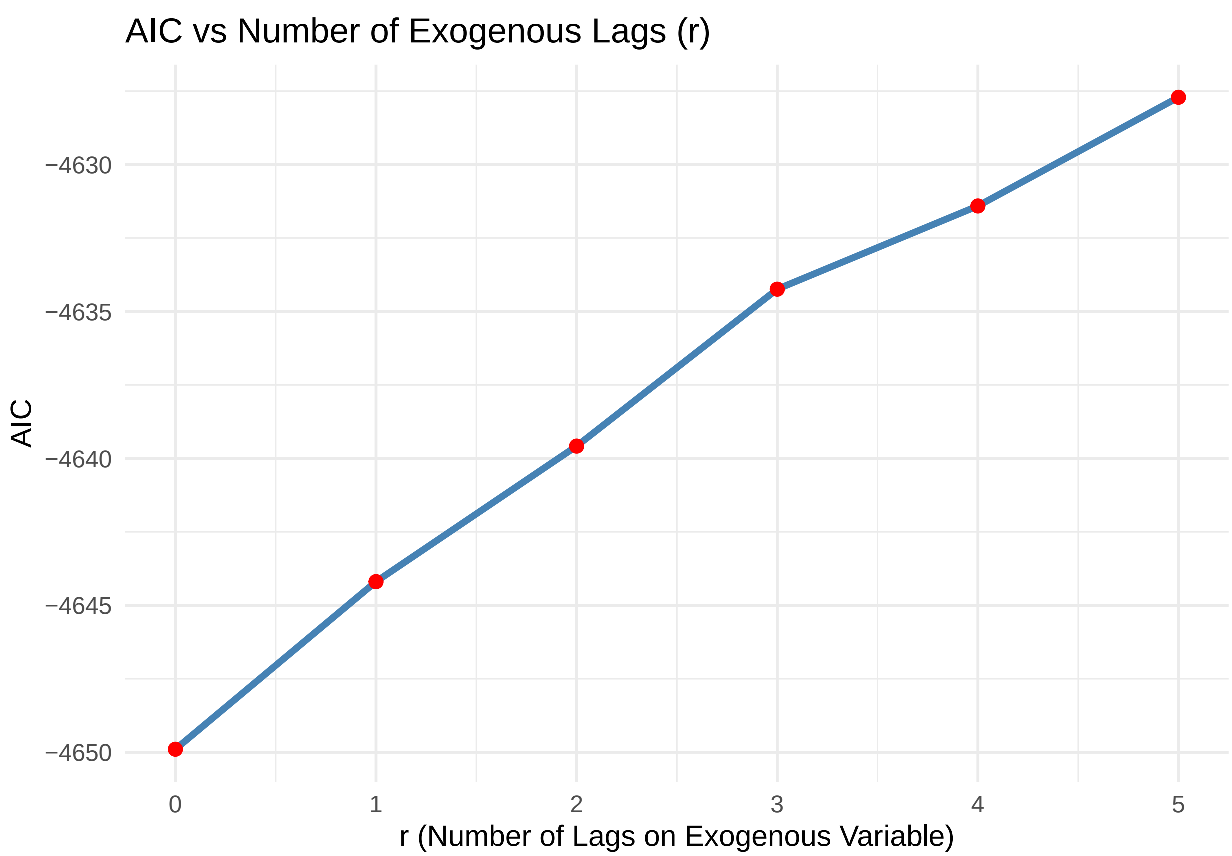
```
## Series: y_trimmed
```

```
## Regression with ARIMA(5,0,3) errors
##
## Coefficients:
```

```
## Warning in sqrt(diag(x$var.coef)): NaNs produced
```

```
##          ar1     ar2      ar3     ar4     ar5     ma1     ma2     ma3
##       0.2371  0.6284  -0.8745  0.1792  0.6758  0.0281  0.0531  0.9357
## s.e.  0.0111  0.0446   0.0635  0.0456  0.0290     NaN  0.0367  0.0409
##       intercept    Lag_0
##          0.0469  -0.0033
## s.e.     0.0183   0.0027
##
## sigma^2 = 0.003045:  log likelihood = 2335.95
## AIC=-4649.9   AICc=-4649.73   BIC=-4590.89
##
## Training set error measures:
##                        ME       RMSE        MAE       MPE     MAPE     MASE
## Training set 9.927148e-05 0.05500831 0.01664516 -51.12605 77.58281 0.945323
##                     ACF1
## Training set 0.006313194
```

```
armax_dayfit$ICplot
```



```r
armax_dayfit$params
```

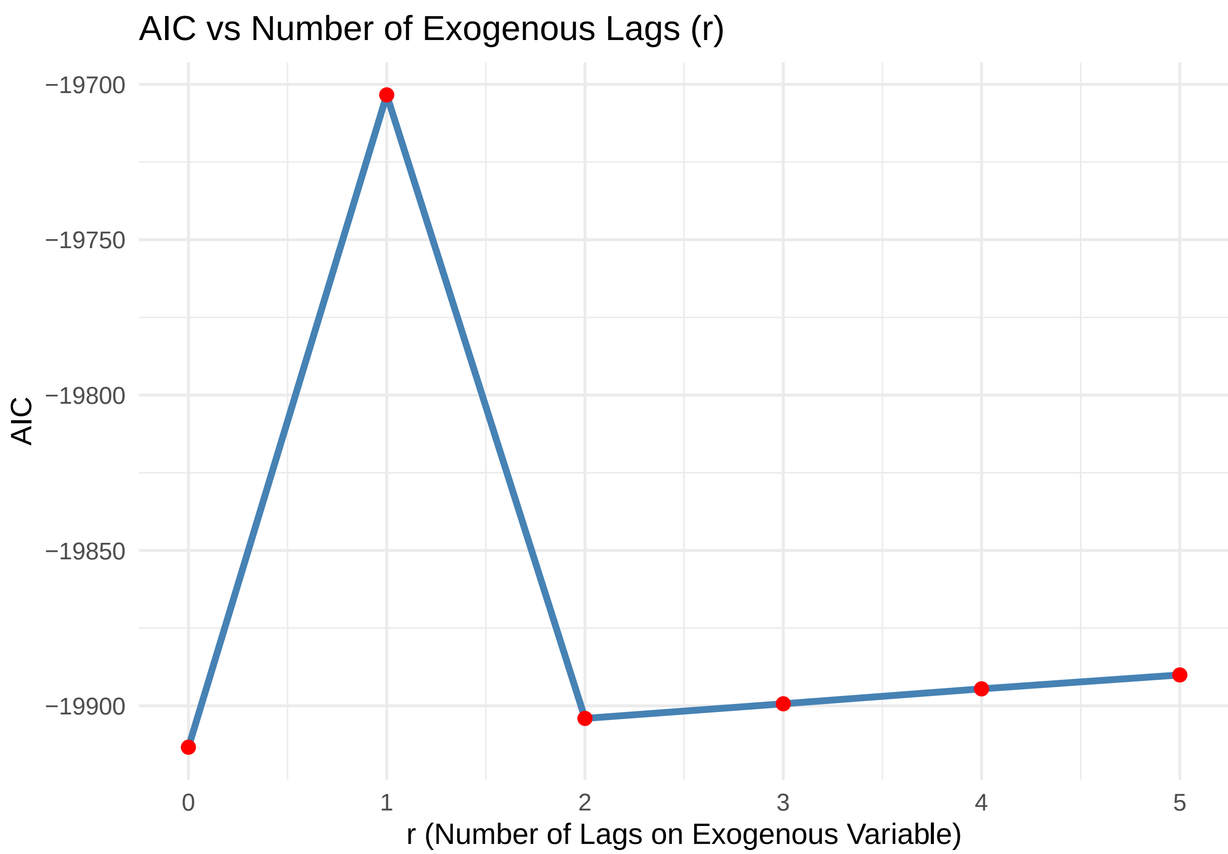
```
## $p
## [1] 5
##
## $q
## [1] 3
##
## $r
## [1] 0
```

Tweet Count on Hourly Volatility

```r
#take all relevant data for armax
countvol_hour = merge(SPY_hvolatility, tt_count_h, by.x = "timestamp_hour",
                      by.y = "timestamp", all.x = T)

#NA tweets means no tweets
countvol_hour$N[is.na(countvol_hour$N)] = 0

#find best armax model and fit
armax_hourfit <- select_armax(countvol_hour$r_vol_h, countvol_hour$N,
                              max_p = 5, max_q = 5, max_r = 5, criterion = "AIC")

summary(armax_hourfit$model)
```

```
## Series: y_trimmed
## Regression with ARIMA(4,0,5) errors
##
## Coefficients:
##          ar1     ar2     ar3      ar4     ma1      ma2      ma3     ma4     ma5
##       0.0846  1.7072  0.0152  -0.8093  0.2494  -1.7008  -0.5851  0.8500  0.2412
## s.e.  0.0184  0.0138  0.0175   0.0138  0.0215   0.0114   0.0299  0.0109  0.0152
##       intercept  Lag_0
##          0.0331  2e-04
## s.e.     0.0241  3e-04
##
## sigma^2 = 0.009625:  log likelihood = 9968.66
## AIC=-19913.32   AICc=-19913.3   BIC=-19825.61
##
## Training set error measures:
##                        ME       RMSE        MAE       MPE     MAPE     MASE
## Training set 0.0008251572 0.09805864 0.02191545 -74.75022 116.6469 1.096779
##                      ACF1
## Training set -0.004361623
```

```r
armax_hourfit$ICplot
```




```r
armax_hourfit$params
```

```
## $p
## [1] 4
##
## $q
## [1] 5
##
## $r
## [1] 0
```

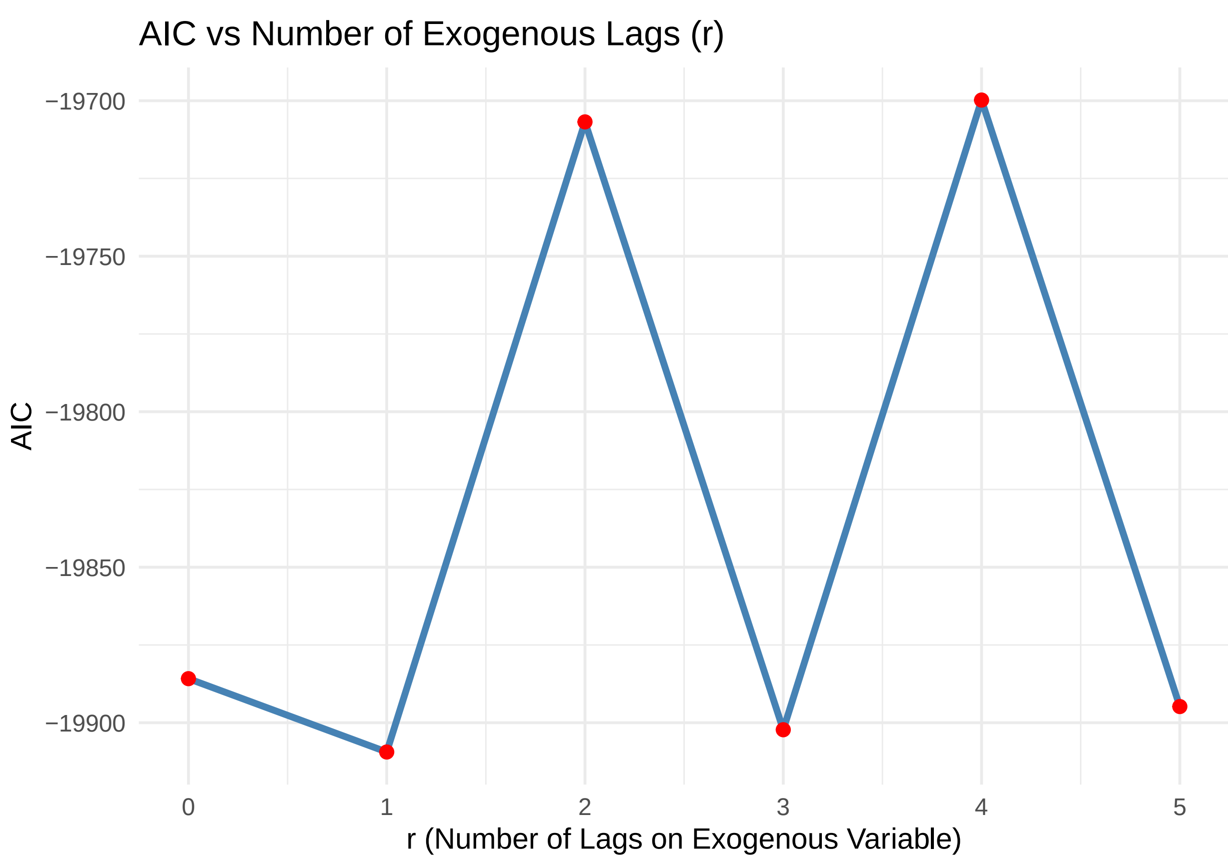
Tweet Dummy on Hourly Volatility

```r
#NA tweets means no tweets
countvol_hour$dummy[is.na(countvol_hour$dummy)] = 0

#find best armax model and fit
armax_hourfit <- select_armax(countvol_hour$r_vol_h, countvol_hour$dummy,
                              max_p = 5, max_q = 5, max_r = 5, criterion = "AIC")

summary(armax_hourfit$model)
```

```
## Series: y_trimmed
```

```
## Regression with ARIMA(4,0,5) errors
##
## Coefficients:
##          ar1     ar2     ar3      ar4     ma1      ma2      ma3     ma4     ma5
##       0.0796  1.7068  0.0198  -0.8085  0.2544  -1.6989  -0.5909  0.8479  0.2433
## s.e.  0.0185  0.0138  0.0175   0.0139  0.0215   0.0117   0.0298  0.0112  0.0152
##       intercept   Lag_0   Lag_1
##          0.0325  0.0016  0.0017
## s.e.     0.0240  0.0019  0.0019
##
## sigma^2 = 0.009626:  log likelihood = 9967.71
## AIC=-19909.42   AICc=-19909.39   BIC=-19814.4
##
## Training set error measures:
##                       ME       RMSE        MAE       MPE     MAPE     MASE
## Training set 0.000812867 0.09805908 0.02194342 -76.01837 118.0597 1.098109
##                      ACF1
## Training set -0.004408657
```

```
armax_hourfit$ICplot
```




```
armax_hourfit$params
```

```
## $p
## [1] 4
```

```
##
## $q
## [1] 5
##
## $r
## [1] 1
```

```r
nb.lags <- 3 #r
count_lags <- embed(countvol_day$N, nb.lags + 1)
dummy_lags <- embed(countvol_day$dummy, nb.lags + 1)
colnames(count_lags) <- paste0("Lag_", 0:nb.lags)

#align volatility to match count rows (for lag)
vol_aligned <- tail(countvol_day$r_vol_d, nrow(count_lags))

#choosing how many lags
# fit an ARMA(0,0,0) model with lm (with r set above)
eq <- lm(vol_aligned ~ count_lags)
eq2 <- lm(vol_aligned ~ dummy_lags)

#compute Newey-West HAC standard errors
var.cov.mat <- NeweyWest(eq, lag = 7, prewhite = FALSE)
robust_se <- sqrt(diag(var.cov.mat))
#for both
var.cov.matD <- NeweyWest(eq2, lag = 7, prewhite = FALSE)
robust_seD <- sqrt(diag(var.cov.matD))

#output table; significant lags are how many we choose
stargazer(eq, eq, type = "text",
          column.labels = c("(no HAC)", "(HAC)"), keep.stat = "n",
          se = list(NULL, robust_se), no.space = TRUE)
```

```
##
## ============================================
##                     Dependent variable:
##                 ----------------------------
##                         vol_aligned
##                    (no HAC)       (HAC)
##                      (1)           (2)
## --------------------------------------------
## count_lagsLag_0     -0.001       -0.001*
##                    (0.001)       (0.001)
## count_lagsLag_1     -0.001       -0.001
##                    (0.001)       (0.001)
## count_lagsLag_2     -0.001       -0.001**
##                    (0.001)       (0.001)
## count_lagsLag_3     -0.001       -0.001**
##                    (0.001)      (0.0005)
## Constant           0.036***     0.036***
##                    (0.002)       (0.005)
## --------------------------------------------
## Observations        1,575         1,575
## ============================================
## Note:            *p<0.1; **p<0.05; ***p<0.01
```

```r
#output table; significant lags are how many we choose
stargazer(eq2, eq2, type = "text",
          column.labels = c("(no HAC)", "(HAC)"), keep.stat = "n",
          se = list(NULL, robust_se), no.space = TRUE)
```

```
##
## =========================================
##                  Dependent variable:
##              ----------------------------
##                      vol_aligned
##                 (no HAC)        (HAC)
##                   (1)            (2)
## -----------------------------------------
## dummy_lags1      -0.007        -0.007
##                 (0.005)
## dummy_lags2      -0.007        -0.007
##                 (0.005)
## dummy_lags3     -0.009*        -0.009
##                 (0.005)
## dummy_lags4      -0.003        -0.003
##                 (0.005)
## Constant        0.040***      0.040***
##                 (0.003)        (0.005)
## -----------------------------------------
## Observations     1,575          1,575
## =========================================
## Note:         *p<0.1; **p<0.05; ***p<0.01
```