Testing

Contents

Data

Raw Political Data

```
#political shocks
raw_truths <- read.csv(here("data/political_data", "truths_new.csv"))
raw_tweets <- read.csv(here("data/political_data", "tweets.csv"))
```

Raw Data

```
#market prices (loads and names them automatically)
#raw_ONEQ <- read.csv(here("data/market_data", "ONEQ.csv")) #USA
#raw_SMI <- read.csv(here("data/market_data", "SMI.csv")) #CH
#raw_VTHR <- read.csv(here("data/market_data", "VTHR.csv")) #USA
#raw_VTI <- read.csv(here("data/market_data", "VTI.csv")) #USA
#raw_DAX <- read.csv(here("data/market_data", "DAX.csv")) #DE
#raw_ASHR <- read.csv(here("data/market_data", "ASHR.csv")) #CHINA
raw_SPYy <- read.csv(here("data/market_data", "Spyqyahoo.csv")) #yahoo

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

#STOXX50
data_loader(symbol="VGK")




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

Daily Data

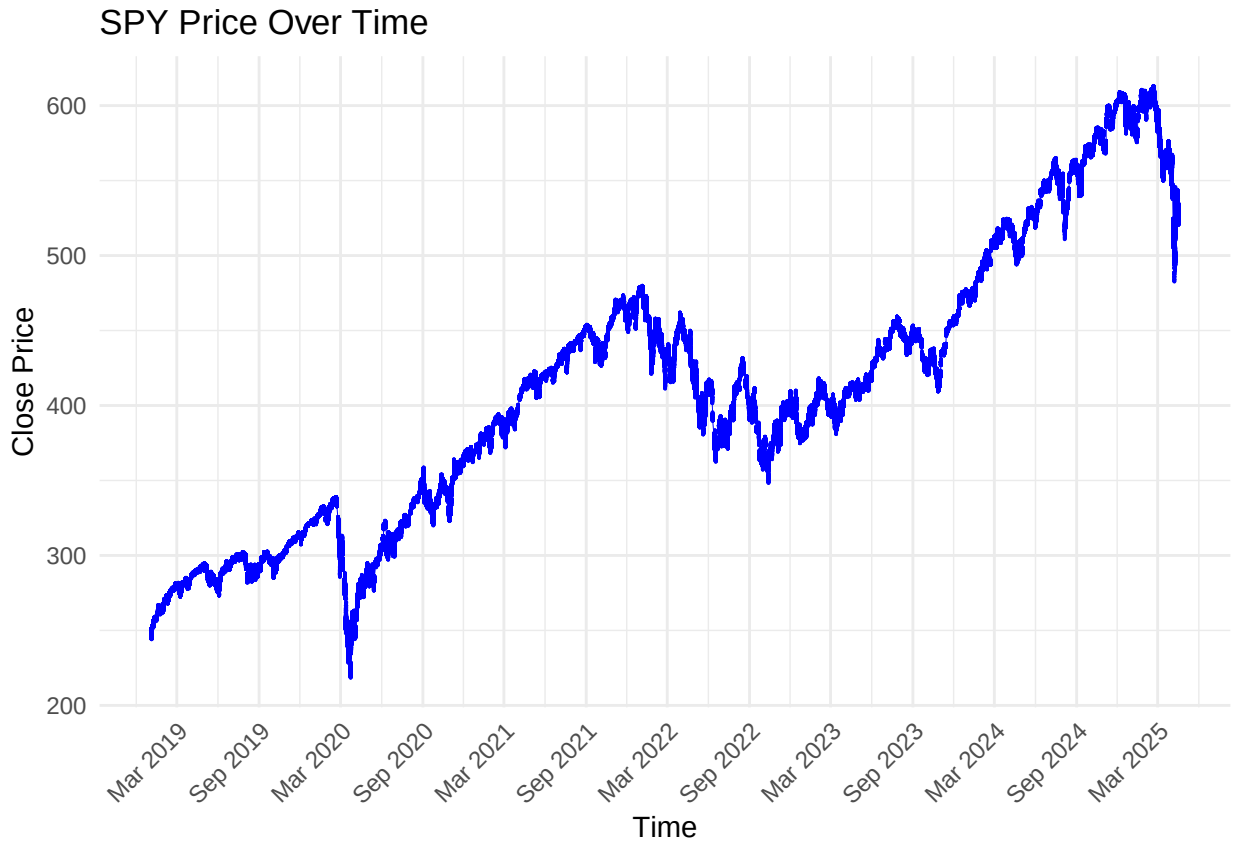
```
#political shocks


#market prices
day_SPY_0409 = filter(raw_SPY, str_detect(timestamp, "^2025-04-09")) #9th of april
day_SPY_0409$timestamp = as.POSIXct(day_SPY_0409$timestamp,
                                    format = "%Y-%m-%d %H:%M:%S", tz = "EST")

yahoo_ds0409 = filter(raw_SPYy, str_detect(Date, "^2025-04-09"))
yahoo_ds0409$Date = as.POSIXct(yahoo_ds0409$Date,
                                    format = "%Y-%m-%dT%H:%M:%S", tz = "UTC")
yahoo_ds0409$Date = with_tz(yahoo_ds0409$Date, "EST")


#extract a particular month
```

```
SPY_24_09 = month_selector(raw_SPY,2024,09) #november 2024


#extract a particular year
SPY_24 = year_selector(raw_SPY,2024) #2024
```

Plots

Total

```
#SPY
ggplot(raw_SPY, aes(x = as.POSIXct(timestamp), y = close)) +
  geom_point(color = "blue", size = 0.01) +
  geom_line(aes(group=1), color="blue", linewidth=0.05) +
  labs(title = "SPY Price Over Time",
       x = "Time",
       y = "Close Price") +
  scale_x_datetime(date_labels = "%b %Y", date_breaks = "6 month") +
      theme_minimal() +
      theme(axis.text.x = element_text(angle = 45, hjust = 1))
```



Per Day

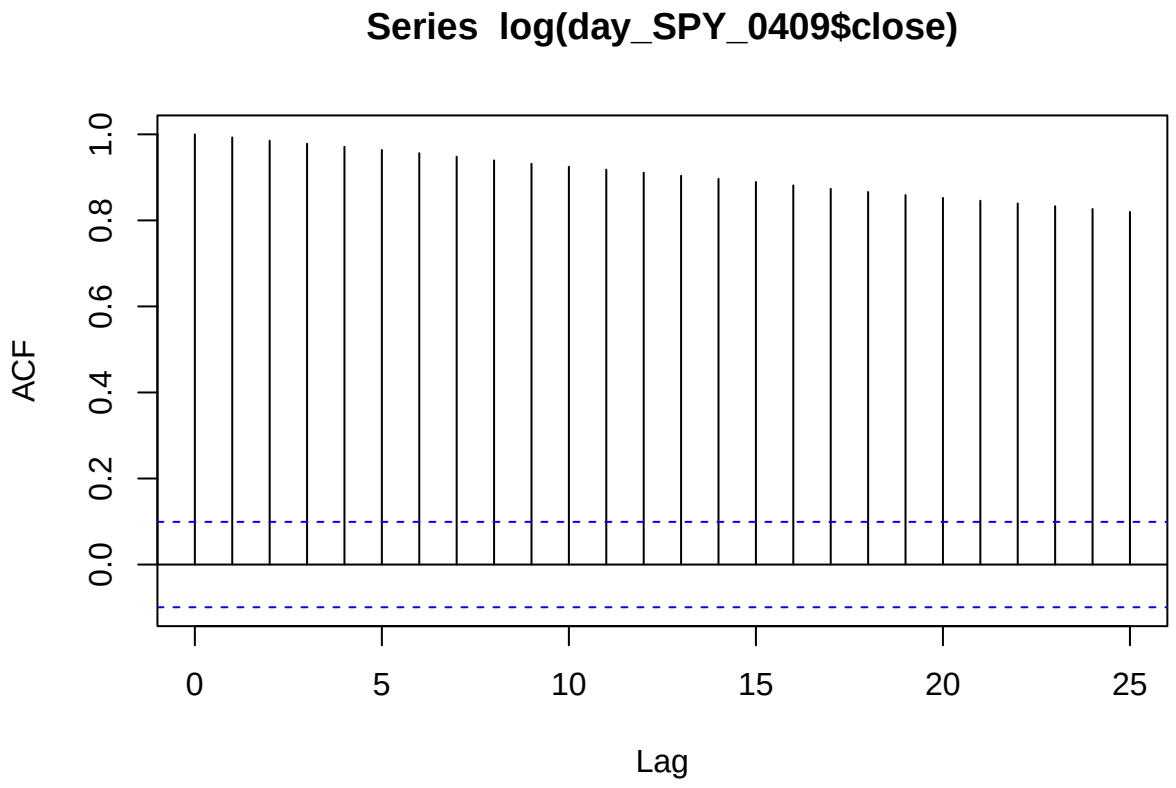
```
#SPY Source: alpha
ggplot(day_SPY_0409, aes(x = timestamp, y = close)) +
  geom_point(color = "blue", size = 0.01) +
  geom_line(aes(group=1)) +
  labs(title = "SPY alpha Price April 9th",
       x = "Time",
       y = "Close Price") +
   scale_x_datetime(date_labels = "%H:%M",
                    date_breaks = "60 min") +
   theme_minimal() +
   theme(axis.text.x = element_text(angle = 45, hjust = 1))
```


```
#SPY Source: yahoo
ggplot(yahoo_ds0409, aes(x = Date, y = Close)) +
  geom_point(color = "blue", size = 0.01) +
  geom_line(aes(group=1)) +
  labs(title = "SPY yahoo Price April 9th",
       x = "Time",
       y = "Close Price") +
   scale_x_datetime(date_labels = "%H:%M",
                    date_breaks = "60 min") +
   theme_minimal() +
   theme(axis.text.x = element_text(angle = 45, hjust = 1))
```

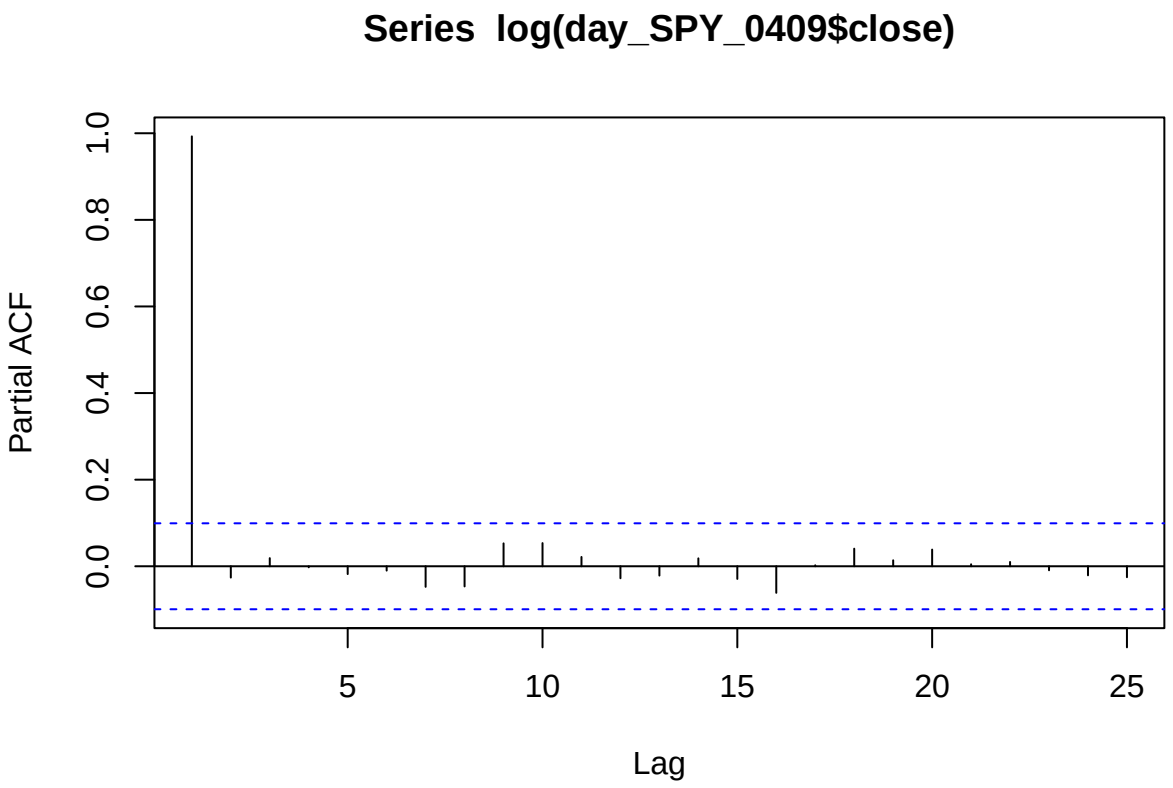
```
## Warning: Removed 3 rows containing missing values or values outside the scale range
## (`geom_point()`).
```

Time Series Analysis

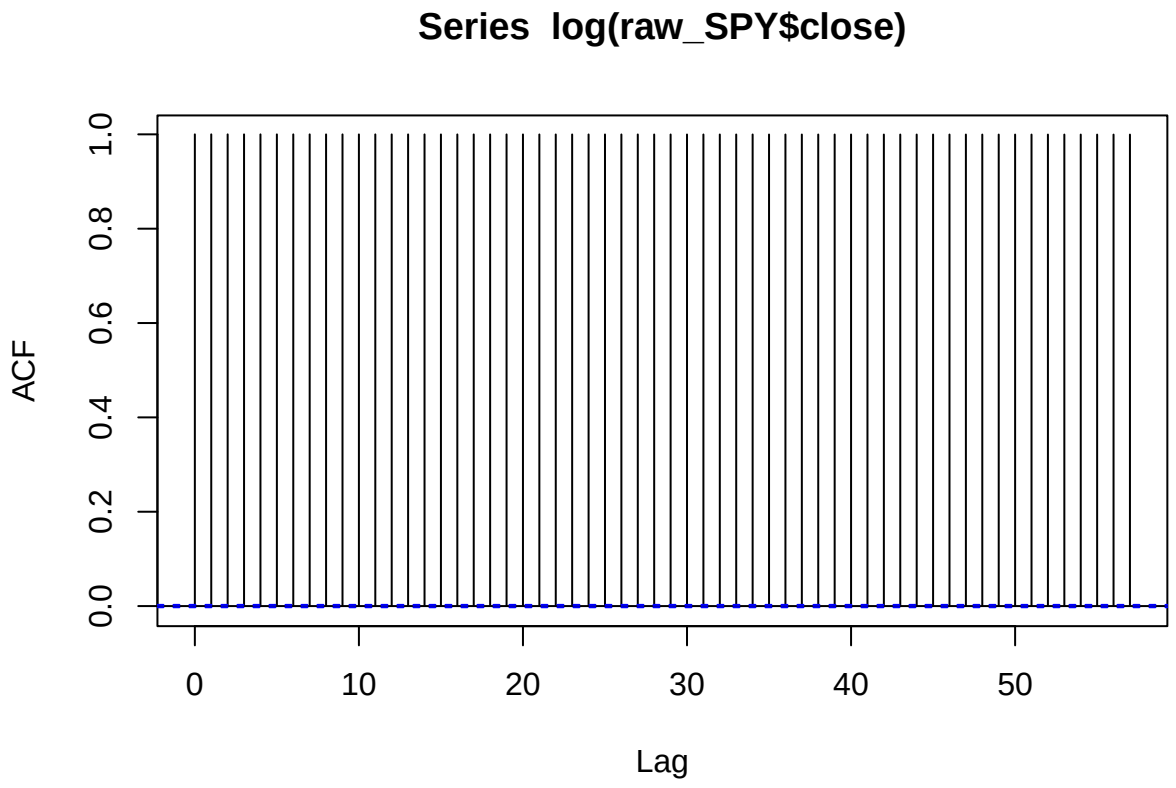
```
acf(log(day_SPY_0409$close))
```



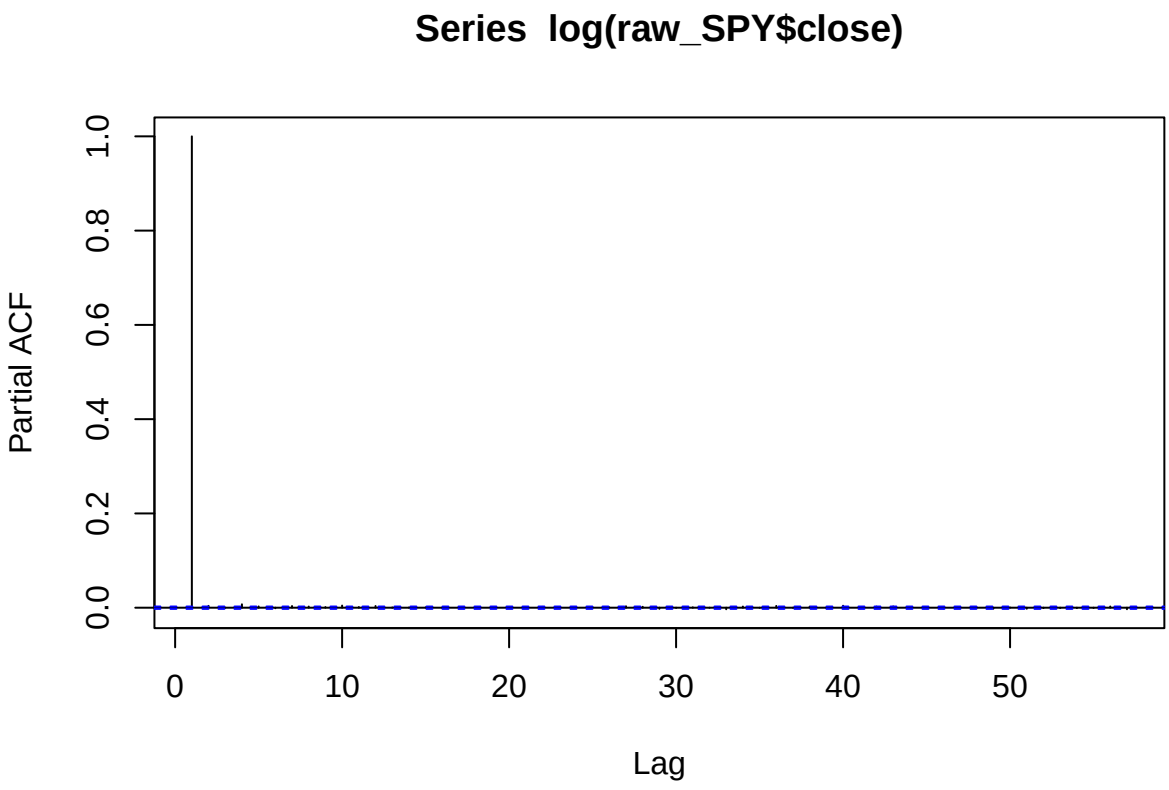

```
pacf(log(day_SPY_0409$close))
```




```
acf(log(raw_SPY$close))
```




```
pacf(log(raw_SPY$close))
```




```
AR1 = arima(day_SPY_0409$close,c(1,0,0),method="ML")
AR2 = arima(day_SPY_0409$close,c(2,0,0),method="ML")
AR3 = arima(day_SPY_0409$close,c(3,0,0),method="ML")
table1 = export_summs(AR1,AR2,AR3, model.names = c("AR1","AR2","AR3"), digits = 4)
```

```
## Warning in FUN(X[[i]], ...): tidy() does not return p values for models of
## class data.frame; significance stars not printed.
## Warning in FUN(X[[i]], ...): tidy() does not return p values for models of
## class data.frame; significance stars not printed.
## Warning in FUN(X[[i]], ...): tidy() does not return p values for models of
## class data.frame; significance stars not printed.
```

```
huxtable::caption(table1) <- "AR Estimations"
huxtable::set_width(table1, 0.8)
```

```
AR1res = as.numeric(AR1$residuals)
AR1res_lagged <- lag(AR1res, 1)
iidcheck1 = lm(AR1res ~ AR1res_lagged)
AR2res = as.numeric(AR2$residuals)
AR2res_lagged <- lag(AR2res, 1)
iidcheck2 = lm(AR2res ~ AR2res_lagged)
AR3res = as.numeric(AR3$residuals)
AR3res_lagged <- lag(AR3res, 1)
iidcheck3 = lm(AR3res ~ AR3res_lagged)
```

Table 1: AR Estimations

	AR1	AR2	AR3
ar1	0.9983	1.0884	1.0919
	(0.0020)	(0.0504)	(0.0506)
intercept	517.4887	516.8318	517.3178
	(19.5350)	(18.7930)	(19.1239)
ar2		-0.0902	-0.1336
		(0.0505)	(0.0746)
ar3			0.0399
			(0.0506)
nobs	390	390	390
sigma	1.2932	1.2880	1.2869
logLik	-656.5286	-654.9411	-654.6302
AIC	1319.0572	1317.8822	1319.2604
BIC	1330.9556	1333.7468	1339.0912
nobs.1	390.0000	390.0000	390.0000

*** p < 0.001; ** p < 0.01; * p < 0.05.

```
table2 = export_summs(iidcheck1,iidcheck2,iidcheck3,
             model.names = c("AR1 Residuals","AR2 Residuals","AR3 Residuals"),
             digits = 4)
huxtable::caption(table2) <- "Checking Residuals"
huxtable::set_width(table2, 0.8)
```

Volatility

JPR Formula

$$v_t = \frac{1}{N}\sum_{i=1}^{N}(\Delta p_{t,i})^2$$

where Δp_t is the difference in price (open - close)
and i represents every minute

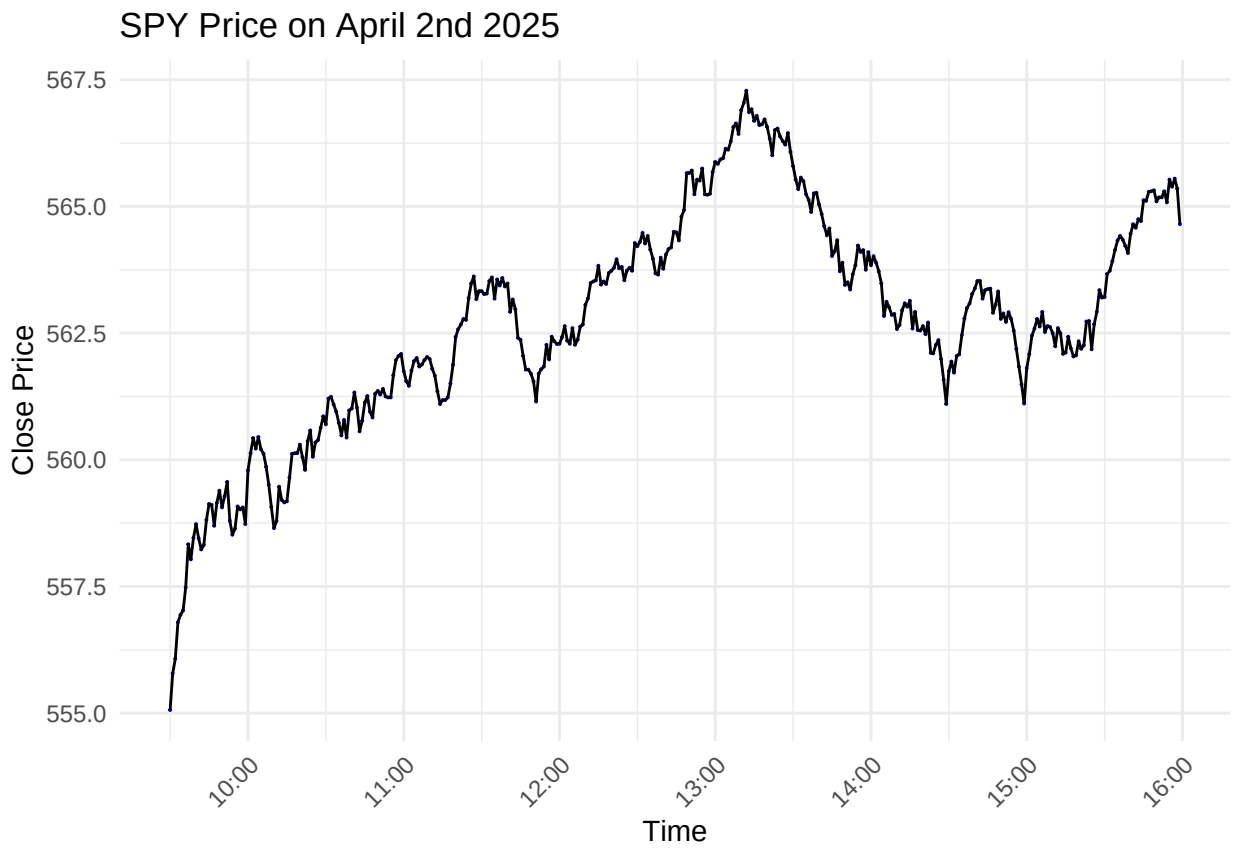
```
#extract a particular day
SPY_25_04_02 = day_selector(raw_SPY,2025,04,02) #april 2nd 2025
```

Table 2: Checking Residuals

	AR1 Residuals	AR2 Residuals	AR3 Residuals
(Intercept)	0.1102	0.1092	0.1135
	(0.0655)	(0.0655)	(0.0654)
AR1res_lagged	0.0799		
	(0.0506)		
AR2res_lagged		-0.0054	
		(0.0508)	
AR3res_lagged			-0.0078
			(0.0508)
N	389	389	389
R2	0.0064	0.0000	0.0001

*** p < 0.001; ** p < 0.01; * p < 0.05.

```
#let's plot it
price_plotter_day(SPY_25_04_02,"SPY Price on April 2nd 2025")
```




```
#quick
quickplot = function(x){price_plotter_day(day_selector(raw_SPY,2025,04,x),"SPY Price 2025")}
quickplot(9)
```


```
#realized volatility
delta_price = SPY_25_04_02$close - SPY_25_04_02$open
delta_price_sqr = delta_price^2
SPY_25_04_02 = cbind(day_selector(raw_SPY,2025,04,02),delta_price_sqr)
v_t = sum(delta_price_sqr) / length(delta_price)

#or is it like this??
#realized volatility method2
p_t = SPY_25_04_02$close
p_t_1 <- lag(p_t, 1)
delta_price2 = p_t_1 - p_t
v_t2 = sum((na.omit(delta_price2))^2) / length(na.omit(delta_price2))
```

```
#average per day (outputs scalar)
r.vol_day(SPY_25_04_02)
```

```
## [1] 0.08152862
```

```
#average per day for each day in a month (outputs vector of each day's realised volatility)
r.vol_month(SPY_24_09)
```

```
##  [1] 0.03554182 0.06306683 0.04483728 0.07865960 0.02596162 0.03080083
##  [7] 0.06853948 0.04630338 0.02524256 0.02271454 0.03173591 0.14493815
## [13] 0.03160202 0.02320854 0.01822570 0.01616798 0.01071128 0.01843709
## [19] 0.01466890 0.02055323
```

```
#for each hour in a day (outputs a vector of each hour's realised volatility)
r.vol_day_hour(SPY_25_04_02)
```

```
## [1] 0.15760939 0.08701794 0.06571201 0.06303564 0.06319524 0.08271313 0.06726031
```

```
#for each hour in a day for each day in a month (outputs a matrix)
month_hour = r.vol_month_hour(SPY_24_09)
huxtable(head(data.frame(month_hour)))
```

X5	X6	X7	X8	X9	X10	X11	X12	X13	X14	X15	X16	X
0.0296	0.0304	0.121	0.0735	0.0232	0.0419	0.0384	0.0141	0.075	0.0243	0.0624	0.0155	0.02
0.0398	0.0607	0.106	0.0779	0.0539	0.0585	0.0284	0.026	0.0428	0.0253	0.0296	0.0349	0.01
0.0256	0.0486	0.0732	0.0547	0.0178	0.0179	0.0181	0.0168	0.0319	0.0315	0.013	0.0132	0.00
0.0124	0.0302	0.0683	0.0275	0.0133	0.0199	0.0471	0.00939	0.0124	0.0112	0.0225	0.00894	0.00
0.0219	0.0189	0.0408	0.0135	0.0093	0.00948	0.0376	0.0152	0.0117	0.013	0.0111	0.00717	0.01
0.0194	0.0147	0.0452	0.0745	0.0279	0.0104	0.035	0.333	0.0253	0.0237	0.00372	0.0118	0.00

```
#avg per day for each month of any dataset
#works for datasets with more than 1 year!
vol_SPY_daily = r.vol_daily(raw_SPY,merge=F)
head(vol_SPY_daily)
```

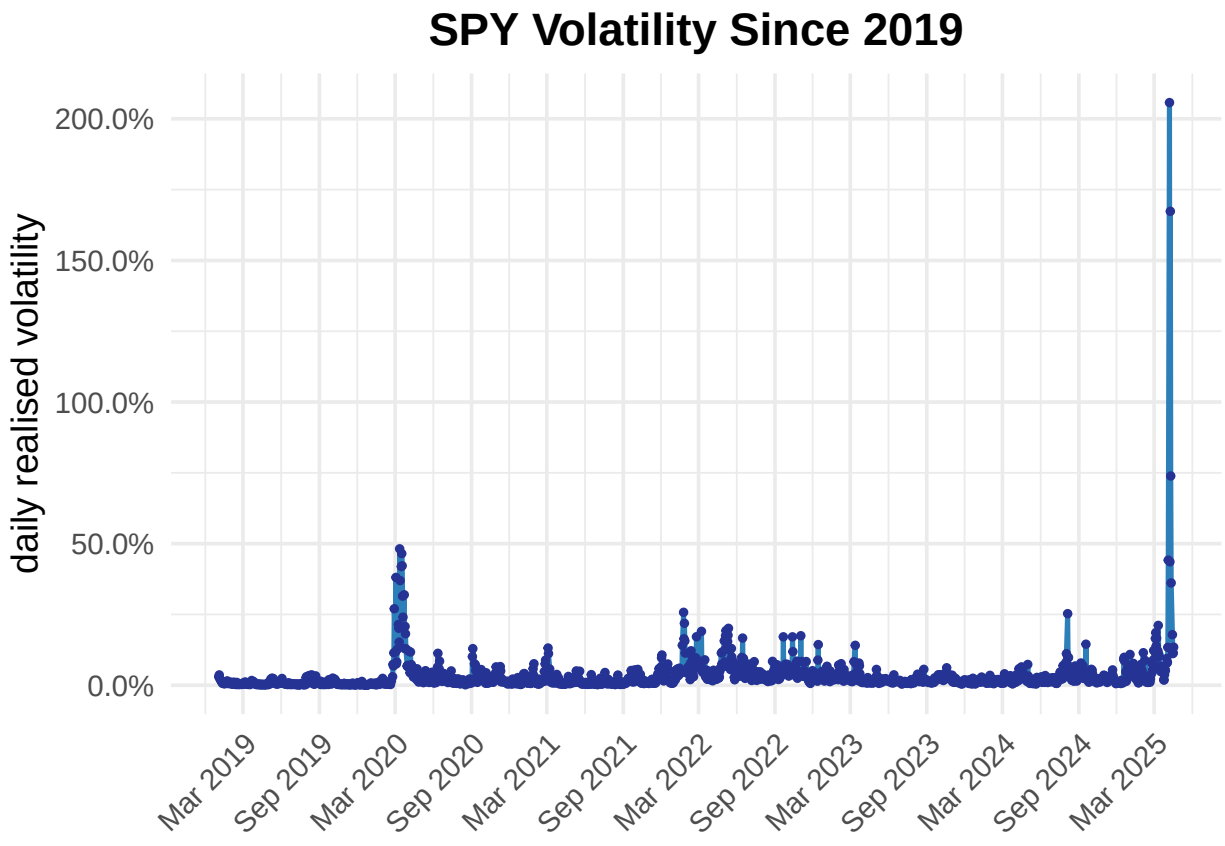
timestamp	r__vol__d
2019-01-02	0.0295
2019-01-03	0.0365
2019-01-04	0.0241
2019-01-07	0.0165
2019-01-08	0.0136
2019-01-09	0.0144

```
#can then filter out years, months, or days
vol_24d = year_selector(vol_SPY_daily,2024)
vol_24_08d = month_selector(vol_SPY_daily,2024,08)
vol_24_11_04d = day_selector(vol_SPY_daily,2024,11,04) #scalar


#avg per hour for each day of each month of any dataset
#works for datasets with more than 1 year!
vol_SPY_hourly = r.vol_hourly(raw_SPY,merge=F)
head(vol_SPY_hourly)
```

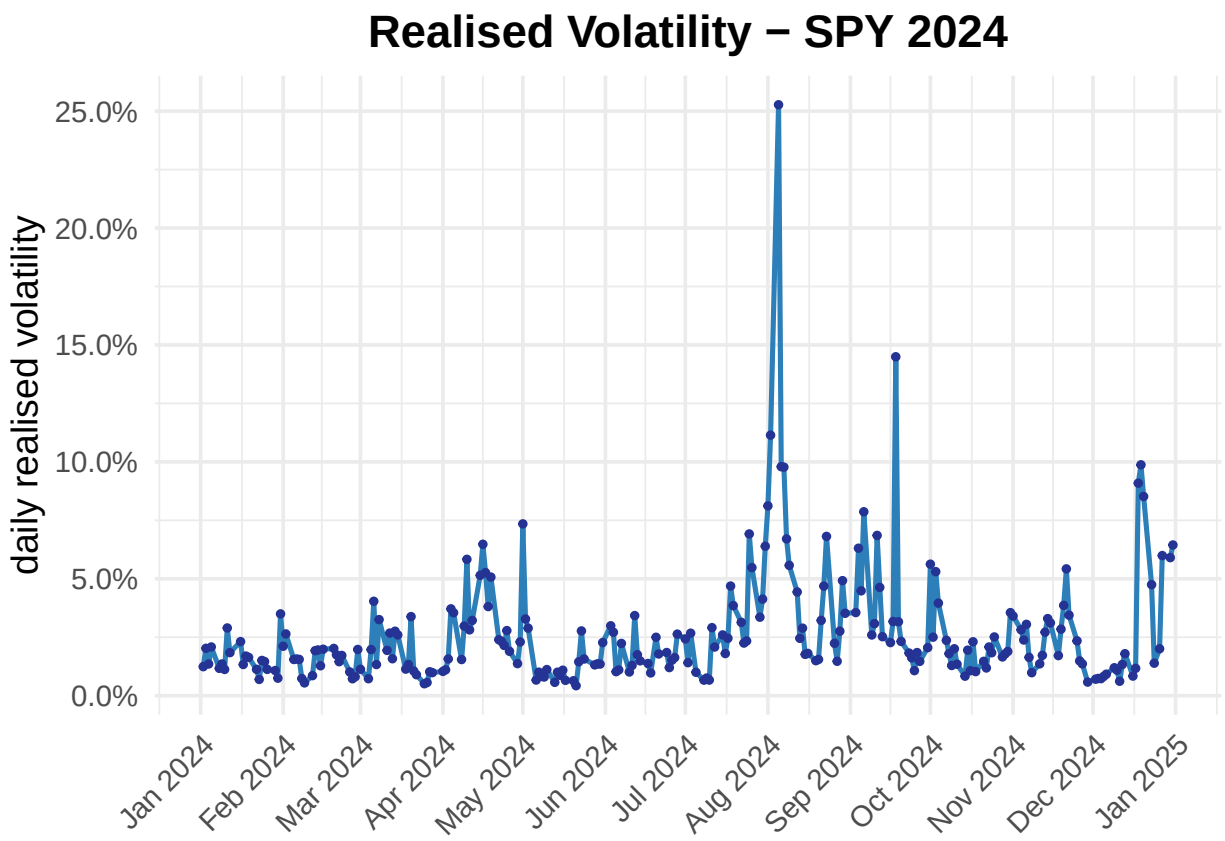
timestamp	r__vol__h
2019-01-02 09:00:00	0.034
2019-01-02 10:00:00	0.0401
2019-01-02 11:00:00	0.0363
2019-01-02 12:00:00	0.0185
2019-01-02 13:00:00	0.0185
2019-01-02 14:00:00	0.0199

```
#can then filter out years, months, or days
vol_24h = year_selector(vol_SPY_hourly,2024)
vol_24_08h = month_selector(vol_SPY_hourly,2024,08)
vol_24_11_04h = day_selector(vol_SPY_hourly,2024,11,04) #vector
```

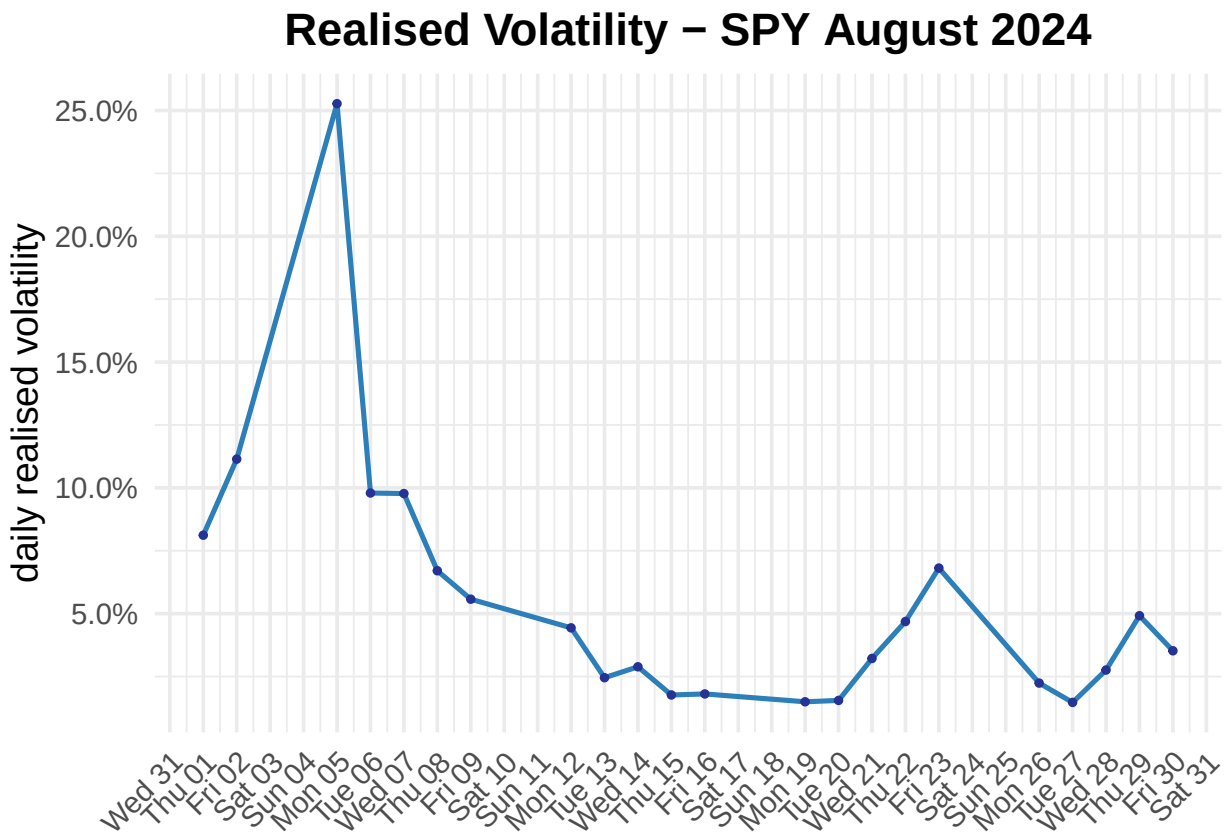
```
#avg per day volatility all time
dvol_plotter(vol_SPY_daily,breaks="yearly",
             title="SPY Volatility Since 2019")
```



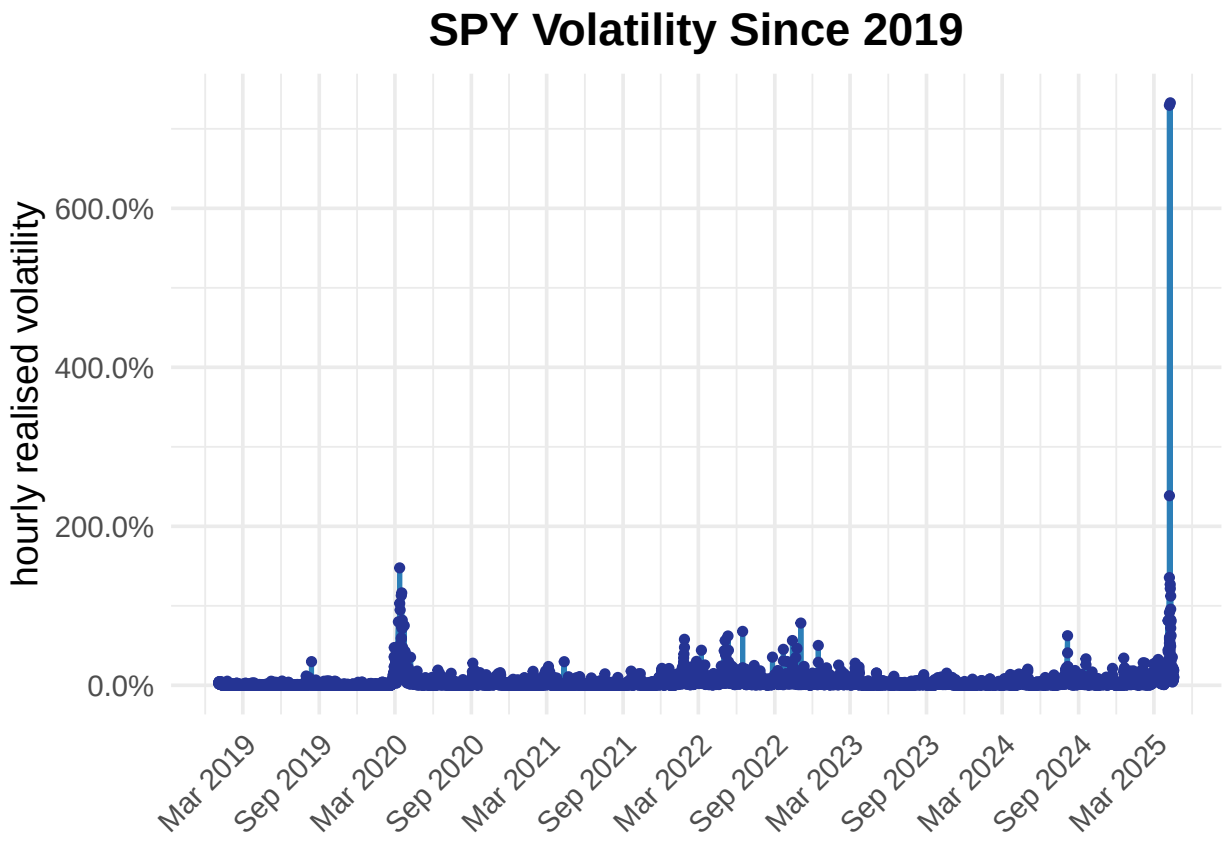
```
#avg per day volatility in a year
dvol_plotter(vol_24d,breaks="monthly",
             title="Realised Volatility - SPY 2024")
```



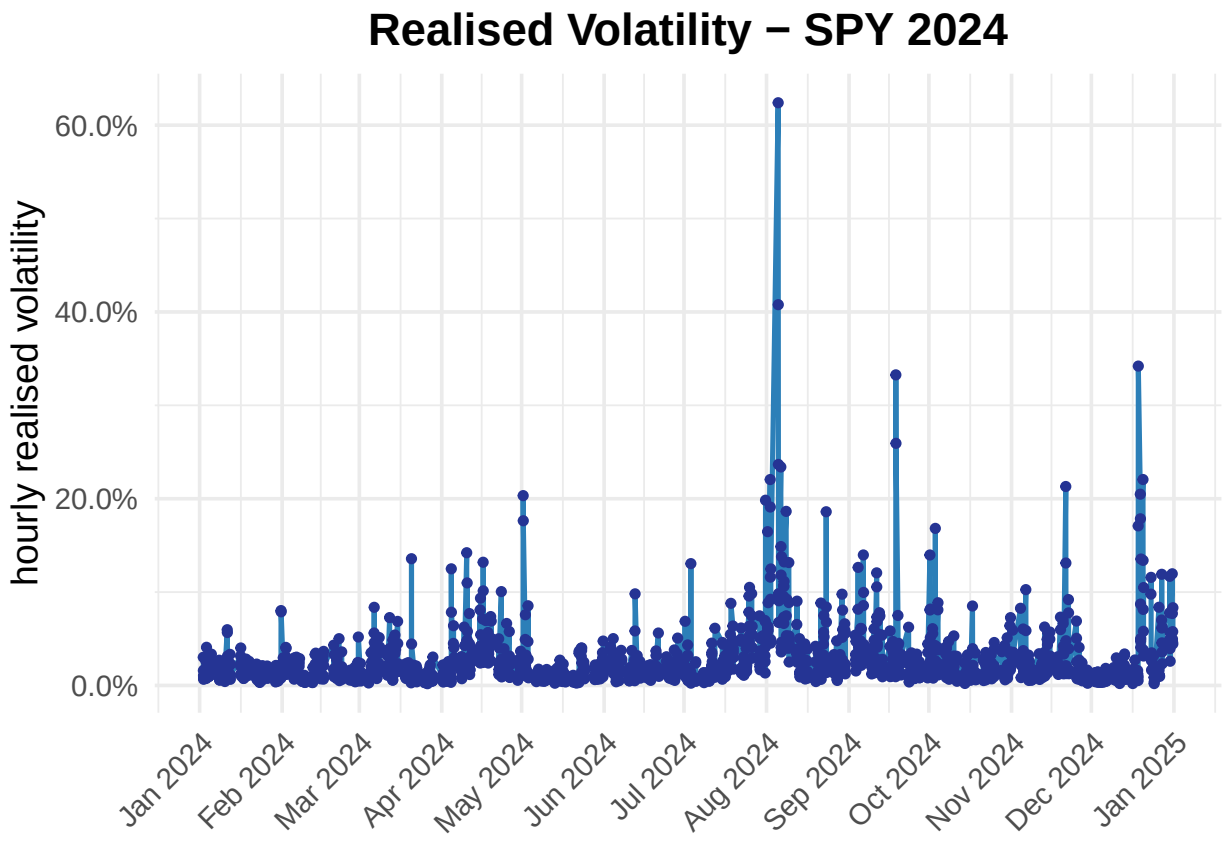

```
#avg per day volatility in a month
dvol_plotter(vol_24_08d,breaks="daily",
             title="Realised Volatility - SPY August 2024")
```



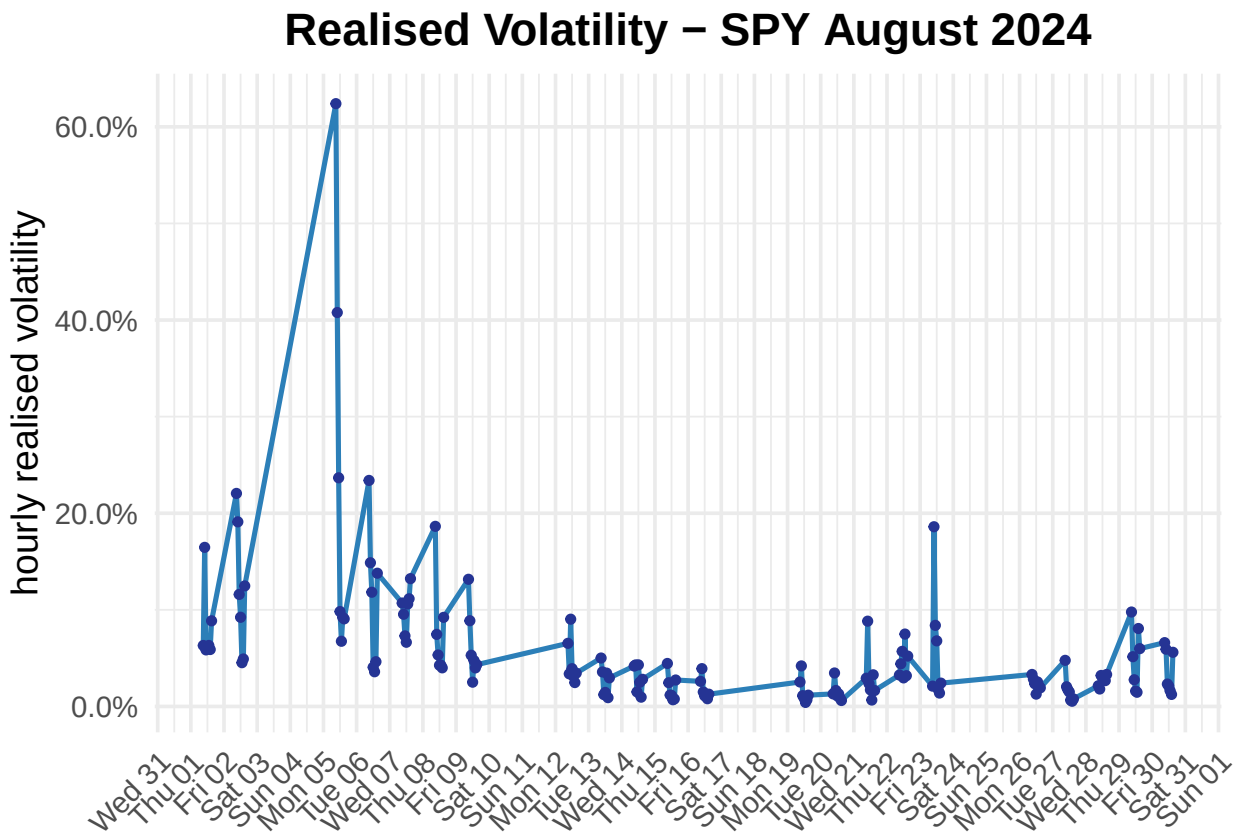

```
#hourly volatility all time
hvol_plotter(vol_SPY_hourly,breaks="yearly",
             title="SPY Volatility Since 2019")
```



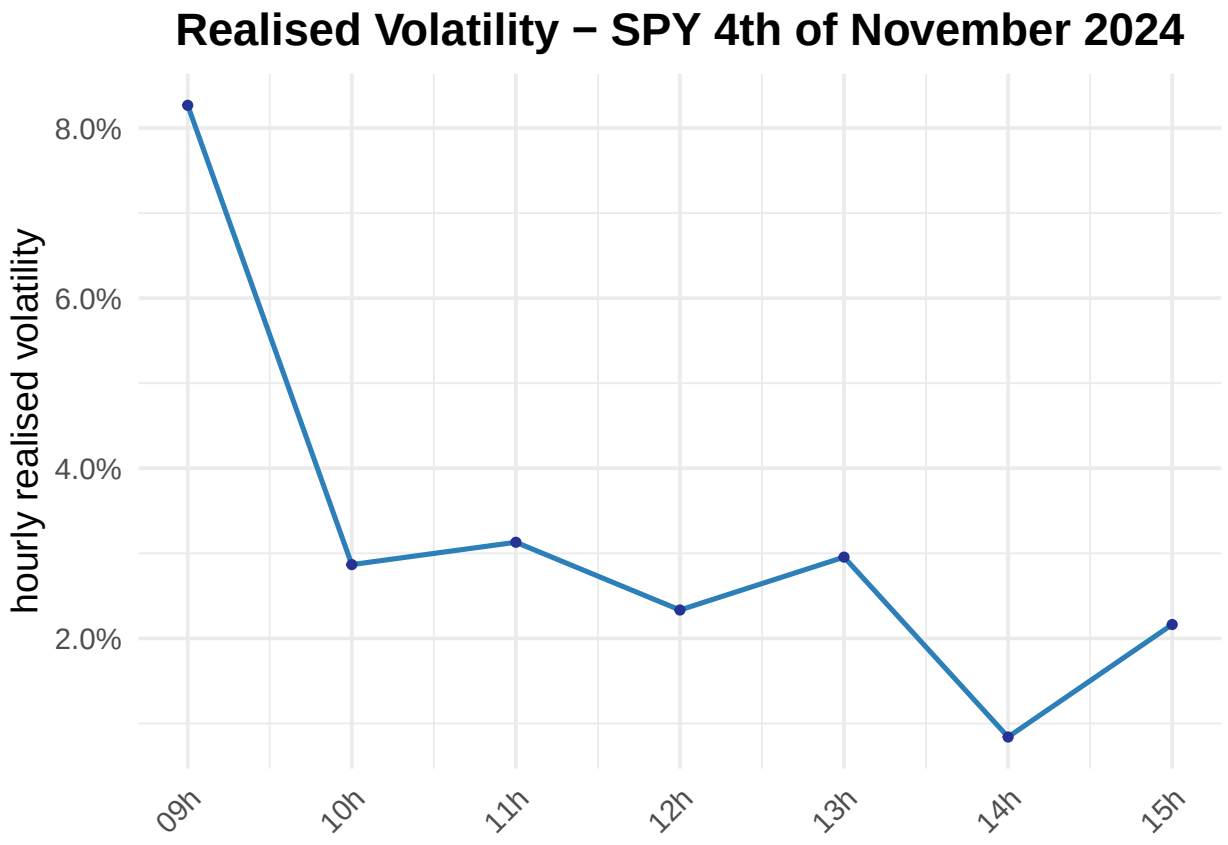

```
#hourly volatility in a year
hvol_plotter(vol_24h,breaks="monthly",
             title="Realised Volatility - SPY 2024")
```




```
#hourly volatility in a month
hvol_plotter(vol_24_08h,breaks="daily",
             title="Realised Volatility - SPY August 2024")
```




```
#hourly volatility in a day
hvol_plotter(vol_24_11_04h,breaks="hourly",
             title="Realised Volatility - SPY 4th of November 2024")
```



Garman and Klass (1980) Formula

Note that this formula uses open-high-low-close information. \ This model is based on the assumption that price returns follow a Wiener process with zero drift and constant infinitesimal variance. It's constructed by minimizing the variance of a quadratic estimator subject to the constraints of price and time symmetry and scale invariance of volatility. Source: https://assets.bbhub.io/professional/sites/10/intraday_volatility-3.pdf

$$V_{ohlc} = 0.5[log(H) - log(L)]^2 - [2log(2) - 1][log(C) - log(O)]^2$$

```
#extract a particular day
day_SPY_0402 = day_selector(raw_SPY,2025,04,02) #april 2nd 2025

#variables
C = day_SPY_0402$close
O = day_SPY_0402$open
H = day_SPY_0402$high
L = day_SPY_0402$low

#realized volatility
V_ohlc = 0.5*(log(H)-log(L))^2 - (2*log(2)-1)*(log(C)-log(O))^2
v_ohlc = sqrt(V_ohlc)
day_SPY_0402 = cbind(day_selector(raw_SPY,2025,04,02),V_ohlc)
avg = sum(V_ohlc) / length(V_ohlc)
```

Tweets & Truths

Tweets

```
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), ]

#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 ),]
head(tweet_count)
```

timestamp	N
2009-05-04	1
2009-05-05	1
2009-05-08	2
2009-05-12	2
2009-05-13	1
2009-05-14	1

Truths

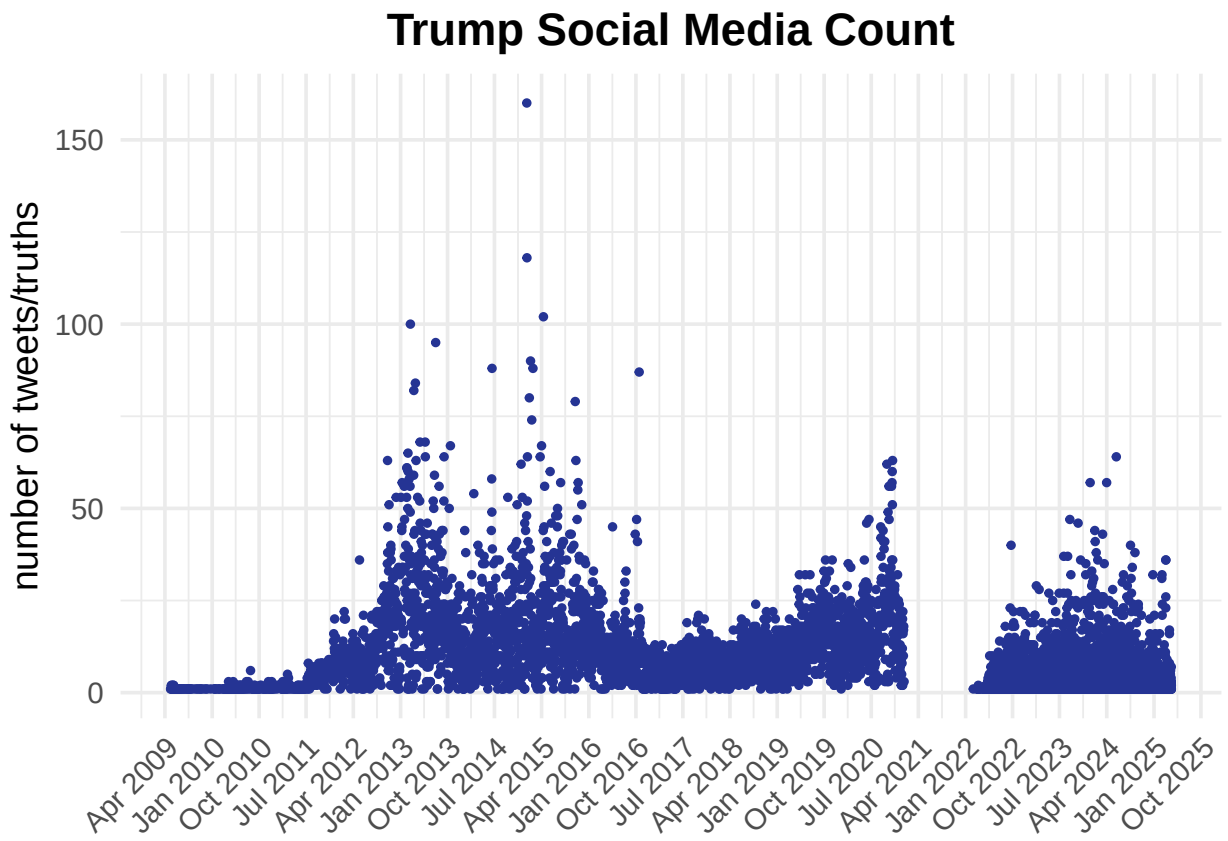
```
truthsbackup <- truths_processer(raw_truths)

#cleanup
truths = as.data.table(truthsbackup)
```

```r
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 ),]
head(truth_count)
```

timestamp	N
2022-02-14 10:00:00	1
2022-03-04 09:00:00	1
2022-03-12 20:00:00	1
2022-03-14 13:00:00	1
2022-03-14 15:00:00	1
2022-03-14 22:00:00	1

Merge

```r
tt_count = rbind(tweet_count,truth_count) #tweets & truths

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))
```



ARMA-X

Hourly

```
#merge by matching the hours
#SPY_prices = select(raw_SPY, timestamp, close, volume)
#SPY_prices$timestamp <- as.POSIXct(raw_SPY$timestamp)
#SPY_prices$timestamp_hour <- as.POSIXct(format(SPY_prices$timestamp,
#                                               "%Y-%m-%d %H:00:00"))
#prices dont make sense cuz not hourly
#armax_price = merge(SPY_prices, truth_count, by.x = "timestamp_hour",
#                    by.y = "timestamp", all.x = TRUE)


#merge by matching the hours
#NOTE: this ignores tweets made outside trading hours!!
SPY_volatility = r.vol_hourly(raw_SPY,merge=F)
#since tweet database not full yet (combine tweets & truths)
SPY_volatility = filter(SPY_volatility,
                        year(SPY_volatility$timestamp) >= 2022 &
                        month(SPY_volatility$timestamp) >= 03)
colnames(SPY_volatility)[1] <- "timestamp_hour"
```

```r
#take all relevant for armax
armax_vol = merge(SPY_volatility, truth_count, by.x = "timestamp_hour",
                  by.y = "timestamp", all.x = T)
#NA tweets means no tweets
armax_vol$N[is.na(armax_vol$N)] = 0

#common names
armax_vol$r_vol = armax_vol$r_vol_h
```

Daily

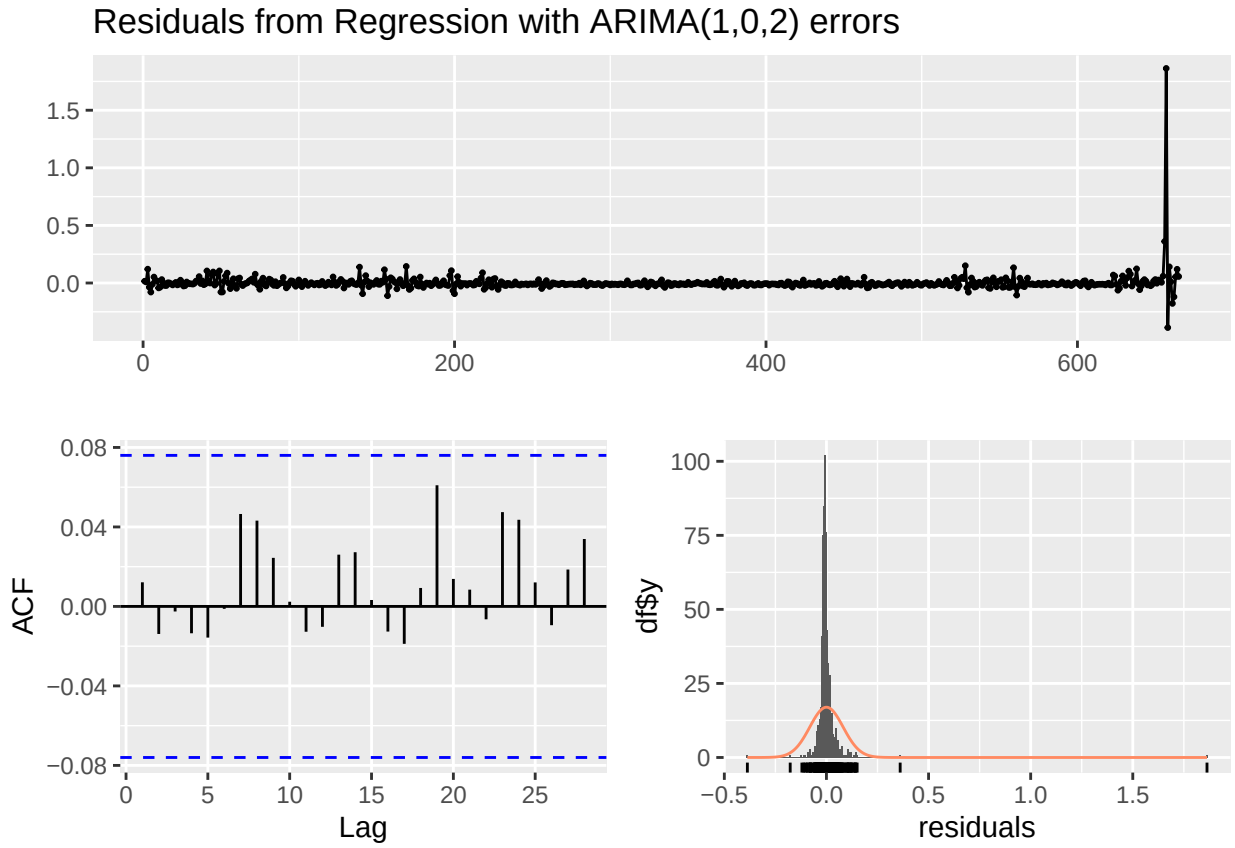
```r
#merge by matching the days

SPY_volatility = r.vol_daily(raw_SPY,merge=F)
#since tweet database not full yet (combine tweets & truths)
SPY_volatility = filter(SPY_volatility,
                        year(SPY_volatility$timestamp) >= 2022 &
                        month(SPY_volatility$timestamp) >= 03)
colnames(SPY_volatility)[1] <- "timestamp_day"

#take all relevant for armax
armax_vol = merge(SPY_volatility, truth_count, by.x = "timestamp_day",
                  by.y = "timestamp", all.x = T)
#NA tweets means no tweets
armax_vol$N[is.na(armax_vol$N)] = 0

#common names
armax_vol$r_vol = armax_vol$r_vol_d
```

```r
T <- length(armax_vol$r_vol_h)

# Create lagged exogenous variables (3 lags of N)
nb.lags <- 3
count_lags <- embed(armax_vol$N, nb.lags + 1)
colnames(count_lags) <- paste0("Lag_", 0:nb.lags)

# Align volatility to match count rows
vol_aligned <- tail(armax_vol$r_vol, nrow(count_lags))
# Fit ARMAX(0,0) with 3 exogenous lags
fit_armax <- Arima(vol_aligned, order = c(1,0,2), xreg = count_lags)
fit_armax2 <- Arima(armax_vol$r_vol, order = c(1,0,2), xreg = armax_vol$N)
fit_armax3 <- Arima(armax_vol$r_vol, order = c(1,0,2))

summary(fit_armax)
```

```
## Series: vol_aligned
## Regression with ARIMA(1,0,2) errors
##
## Coefficients:
##          ar1      ma1     ma2  intercept    Lag_0   Lag_1    Lag_2    Lag_3
##       0.5928  -0.3398  0.6032     0.0460  -0.0005  0.0000  -0.0004  -0.0003
```

```
## s.e.  0.0411    0.0385  0.0324     0.0102    0.0014  0.0014    0.0014    0.0014
##
## sigma^2 = 0.006706:  log likelihood = 723.79
## AIC=-1429.59   AICc=-1429.31   BIC=-1389.09
##
## Training set error measures:
##                       ME       RMSE        MAE       MPE     MAPE      MASE
## Training set 1.91058e-05 0.08139779 0.02451252 -53.17103 79.45613 0.9901503
##                     ACF1
## Training set 0.01213776
```

```
checkresiduals(fit_armax)
```



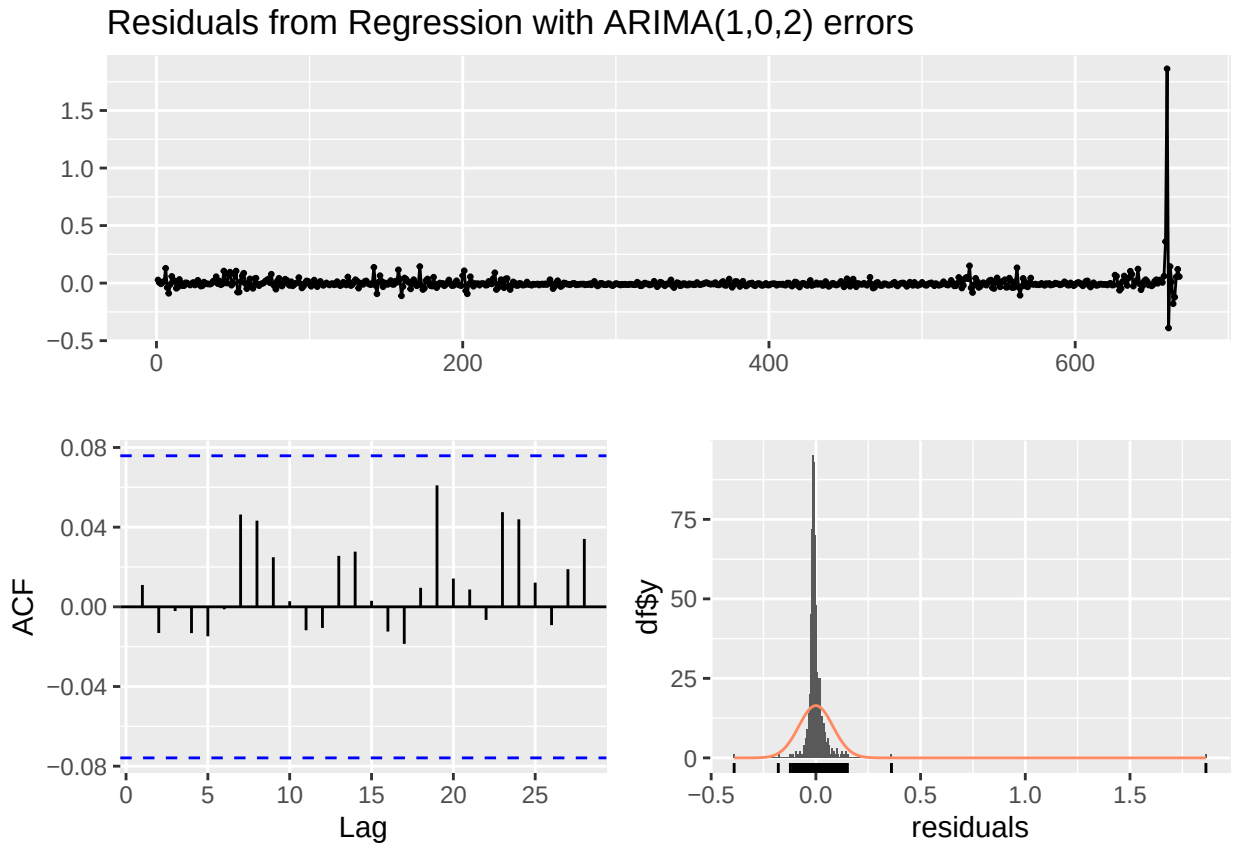

```
##
##  Ljung-Box test
##
## data:  Residuals from Regression with ARIMA(1,0,2) errors
## Q* = 3.6451, df = 7, p-value = 0.8196
##
## Model df: 3.   Total lags used: 10
```

```
summary(fit_armax2)
```

```
## Series: armax_vol$r_vol
```

```
## Regression with ARIMA(1,0,2) errors
##
## Coefficients:
##          ar1      ma1     ma2  intercept    xreg
##       0.5936  -0.3386  0.6010     0.0458  -2e-04
## s.e.  0.0410   0.0384  0.0324     0.0098   1e-03
##
## sigma^2 = 0.006656:  log likelihood = 728.04
## AIC=-1444.08   AICc=-1443.95   BIC=-1417.05
##
## Training set error measures:
##                        ME       RMSE        MAE       MPE    MAPE      MASE
## Training set -0.000101056 0.08127921 0.02449869 -53.79234 79.4115 0.9924542
##                     ACF1
## Training set 0.01096865
```

```
checkresiduals(fit_armax2)
```

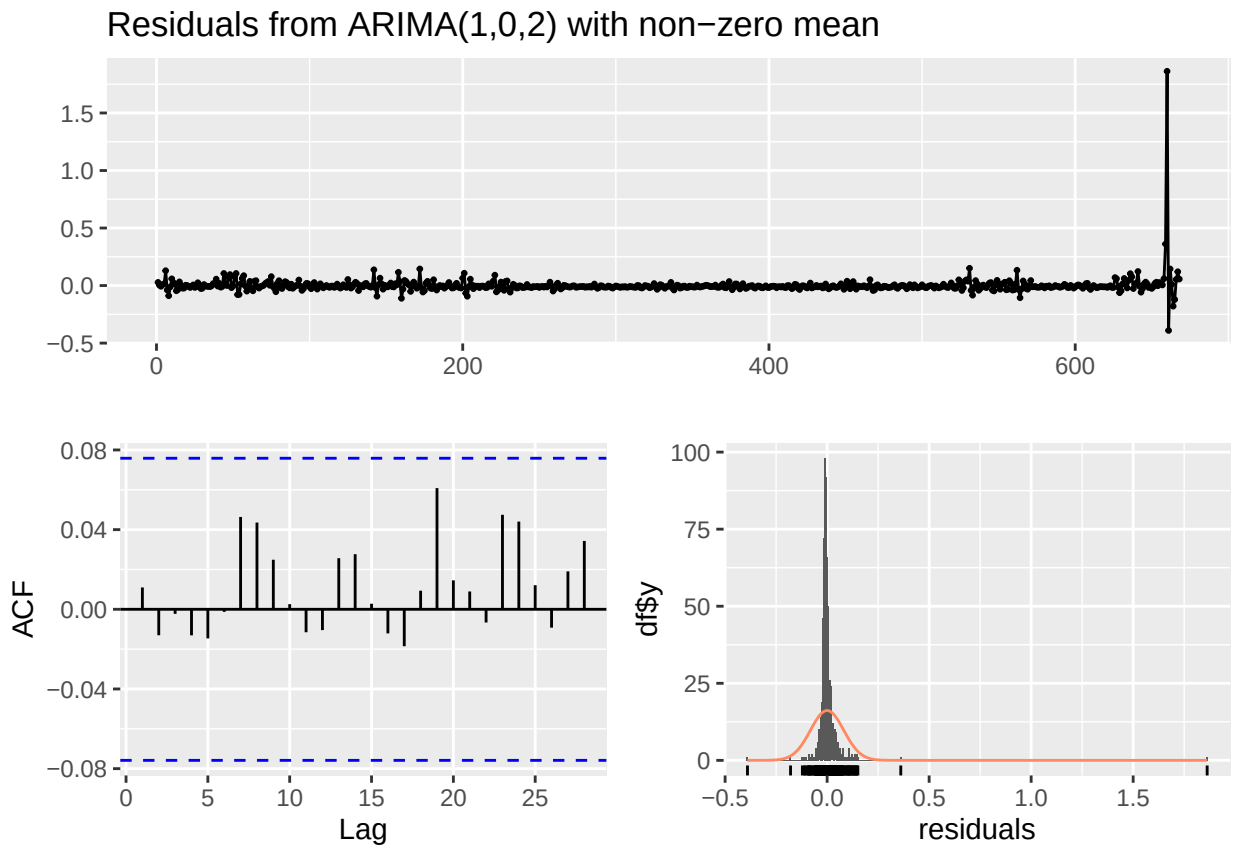



```
##
##  Ljung-Box test
##
## data:  Residuals from Regression with ARIMA(1,0,2) errors
## Q* = 3.615, df = 7, p-value = 0.8229
##
## Model df: 3.   Total lags used: 10
```

```
summary(fit_armax3)
```

```
## Series: armax_vol$r_vol
## ARIMA(1,0,2) with non-zero mean
##
## Coefficients:
##          ar1      ma1     ma2    mean
##       0.5938  -0.3386  0.6010  0.0456
## s.e.  0.0410   0.0384  0.0324  0.0097
##
## sigma^2 = 0.006647:  log likelihood = 728.01
## AIC=-1446.02   AICc=-1445.93   BIC=-1423.5
##
## Training set error measures:
##                         ME       RMSE        MAE       MPE     MAPE      MASE
## Training set -0.0001041813 0.08128258 0.02447341 -53.95056 79.43743 0.9914299
##                    ACF1
## Training set 0.0109513
```

```
checkresiduals(fit_armax3)
```




```
##
##  Ljung-Box test
##
```

```
## data:  Residuals from ARIMA(1,0,2) with non-zero mean
## Q* = 3.6257, df = 7, p-value = 0.8217
##
## Model df: 3.   Total lags used: 10
```

```
# Refit using lm for robust SEs
eq <- lm(vol_aligned ~ count_lags)

# Compute Newey-West HAC standard errors
var.cov.mat <- NeweyWest(eq, lag = 7, prewhite = FALSE)
robust_se <- sqrt(diag(var.cov.mat))

# Stargazer output
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       -0.002        -0.002**
##                      (0.002)         (0.001)
## count_lagsLag_1       -0.002         -0.002
##                      (0.002)         (0.001)
## count_lagsLag_2       -0.002         -0.002*
##                      (0.002)         (0.001)
## count_lagsLag_3       -0.002        -0.002**
##                      (0.002)         (0.001)
## Constant             0.051***       0.051***
##                      (0.005)         (0.011)
## -----------------------------------------------
## Observations           665             665
## ===============================================
## Note:              *p<0.1; **p<0.05; ***p<0.01
```

```
# Choosing the best model
best_model <- auto.arima(vol_aligned, xreg = count_lags, seasonal = FALSE,
                         max.p = 5, max.q = 5, max.d = 0,
                         stepwise = T, approximation = FALSE, trace = TRUE)
```

```
##
##  Regression with ARIMA(0,0,0) errors : -918.5866
##  Regression with ARIMA(0,0,0) errors : -1008.586
##  Regression with ARIMA(0,0,1) errors : -1037.055
##  Regression with ARIMA(0,0,1) errors : -1099.894
##  Regression with ARIMA(0,0,2) errors : -1265.033
##  Regression with ARIMA(0,0,2) errors : -1313.548
##  Regression with ARIMA(0,0,3) errors : -1364.167
```

```
##  Regression with ARIMA(0,0,3) errors : -1398.411
##  Regression with ARIMA(0,0,4) errors : -1394.093
##  Regression with ARIMA(0,0,4) errors : -1419.88
##  Regression with ARIMA(0,0,5) errors : -1404.402
##  Regression with ARIMA(0,0,5) errors : -1425.312
##  Regression with ARIMA(1,0,0) errors : -1169.352
##  Regression with ARIMA(1,0,0) errors : -1196.03
##  Regression with ARIMA(1,0,1) errors : -1267.25
##  Regression with ARIMA(1,0,1) errors : -1276.798
##  Regression with ARIMA(1,0,2) errors : -1413.636
##  Regression with ARIMA(1,0,2) errors : -1429.314
##  Regression with ARIMA(1,0,3) errors : -1411.641
##  Regression with ARIMA(1,0,3) errors : -1427.991
##  ARIMA(1,0,4) with zero mean     : Inf
##  Regression with ARIMA(1,0,4) errors : -1426.007
##  Regression with ARIMA(2,0,0) errors : -1343.673
##  Regression with ARIMA(2,0,0) errors : -1352.575
##  Regression with ARIMA(2,0,1) errors : -1342.213
##  Regression with ARIMA(2,0,1) errors : -1351.552
##  Regression with ARIMA(2,0,2) errors : -1411.632
##  Regression with ARIMA(2,0,2) errors : -1428.015
##  ARIMA(2,0,3) with zero mean     : Inf
##  Regression with ARIMA(2,0,3) errors : -1425.937
##  Regression with ARIMA(3,0,0) errors : -1342.782
##  Regression with ARIMA(3,0,0) errors : -1352.679
##  Regression with ARIMA(3,0,1) errors : -1348.302
##  Regression with ARIMA(3,0,1) errors : -1362.879
##  Regression with ARIMA(3,0,2) errors : -1410
##  Regression with ARIMA(3,0,2) errors : -1425.95
##  Regression with ARIMA(4,0,0) errors : -1377.856
##  Regression with ARIMA(4,0,0) errors : -1394.578
##  Regression with ARIMA(4,0,1) errors : -1375.797
##  Regression with ARIMA(4,0,1) errors : -1392.696
##  Regression with ARIMA(5,0,0) errors : -1375.8
##  Regression with ARIMA(5,0,0) errors : -1392.934
##
##
##
##  Best model: Regression with ARIMA(1,0,2) errors
```

```
summary(best_model)
```

```
## Series: vol_aligned
## Regression with ARIMA(1,0,2) errors
##
## Coefficients:
##          ar1      ma1     ma2  intercept    Lag_0   Lag_1    Lag_2    Lag_3
##       0.5928  -0.3398  0.6032     0.0460  -0.0005  0.0000  -0.0004  -0.0003
## s.e.  0.0411   0.0385  0.0324     0.0102   0.0014  0.0014   0.0014   0.0014
##
## sigma^2 = 0.006706:  log likelihood = 723.79
## AIC=-1429.59   AICc=-1429.31   BIC=-1389.09
##
## Training set error measures:
```

```
##                       ME       RMSE        MAE       MPE     MAPE      MASE
## Training set 1.91058e-05 0.08139779 0.02451252 -53.17103 79.45613 0.9901503
##                     ACF1
## Training set 0.01213776
```

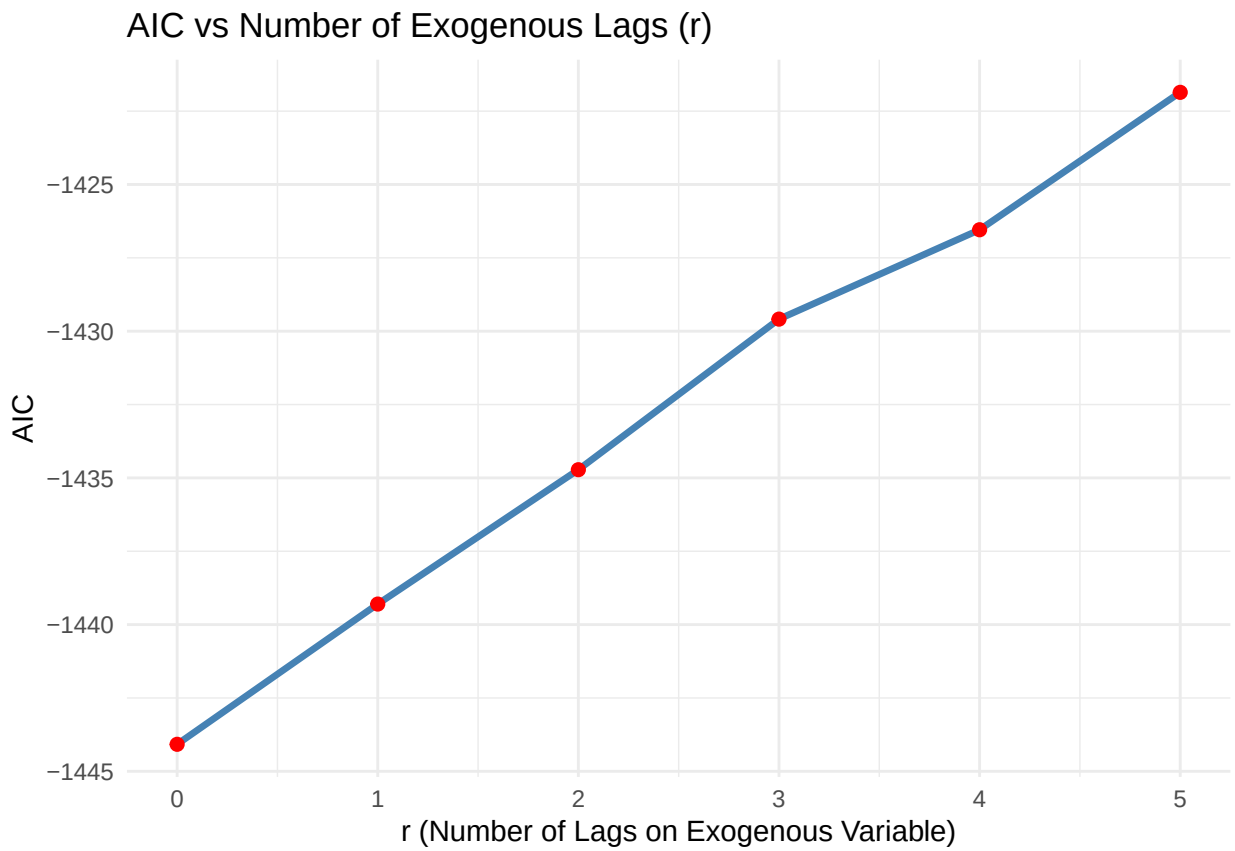
```
result <- select_armax(armax_vol$r_vol, armax_vol$N,
                       max_p = 3, max_q = 3, max_r = 5, criterion = "AIC")
```

```
## 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.
```

```
summary(result$model)
```

```
## Series: y_trimmed
## Regression with ARIMA(1,0,2) errors
##
## Coefficients:
##          ar1      ma1     ma2  intercept   Lag_0
##       0.5936  -0.3386  0.6010     0.0458  -2e-04
## s.e.  0.0410   0.0384  0.0324     0.0098   1e-03
##
## sigma^2 = 0.006656:  log likelihood = 728.04
## AIC=-1444.08   AICc=-1443.95   BIC=-1417.05
##
## Training set error measures:
##                        ME       RMSE        MAE       MPE    MAPE      MASE
## Training set -0.000101056 0.08127921 0.02449869 -53.79234 79.4115 0.9924542
##                     ACF1
## Training set 0.01096865
```

```
result$AICplot
```


AIC vs Number of Exogenous Lags (r)
AIC
−1425
−1430
−1435
−1440
−1445
0
1
2
3
4
5
r (Number of Lags on Exogenous Variable)