Exploratory_Hourly

```
rm(list=ls())
require(tinytex) #LaTeX
```

```
## Lade nötiges Paket: tinytex
```

```
require(ggplot2) #plots
```

```
## Lade nötiges Paket: ggplot2
```

```
require(AEC) #JP-Renne functions
```

```
## Lade nötiges Paket: AEC
```

```
require(AER) #NW formula
```

```
## Lade nötiges Paket: AER

## Warning: Paket 'AER' wurde unter R Version 4.4.3 erstellt

## Lade nötiges Paket: car

## Warning: Paket 'car' wurde unter R Version 4.4.3 erstellt

## Lade nötiges Paket: carData

## Warning: Paket 'carData' wurde unter R Version 4.4.3 erstellt

## Lade nötiges Paket: lmtest

## Lade nötiges Paket: zoo

##
## Attache Paket: 'zoo'

## Die folgenden Objekte sind maskiert von 'package:base':
##
##     as.Date, as.Date.numeric

## Lade nötiges Paket: sandwich

## Lade nötiges Paket: survival
```

```r
require(forecast) #time series stuff
```

```
## Lade nötiges Paket: forecast

## Registered S3 method overwritten by 'quantmod':
##   method            from
##   as.zoo.data.frame zoo
```

```r
require(expm) #matrix exponents
```

```
## Lade nötiges Paket: expm

## Lade nötiges Paket: Matrix

##
## Attache Paket: 'expm'

## Das folgende Objekt ist maskiert 'package:Matrix':
##
##     expm
```

```r
require(here) #directory finder
```

```
## Lade nötiges Paket: here

## here() starts at C:/Users/jonas/Desktop/repos/mmetricsproject
```

```r
require(stringr) # analysis of strings, important for the detection in tweets
```

```
## Lade nötiges Paket: stringr
```

```r
require(dplyr) #data management
```

```
## Lade nötiges Paket: dplyr

##
## Attache Paket: 'dplyr'

## Das folgende Objekt ist maskiert 'package:car':
##
##     recode

## Die folgenden Objekte sind maskiert von 'package:stats':
##
##     filter, lag

## Die folgenden Objekte sind maskiert von 'package:base':
##
##     intersect, setdiff, setequal, union
```

```r
require(lubridate) #data dates management
```

```
## Lade nötiges Paket: lubridate
```

```
##
## Attache Paket: 'lubridate'
```

```
## Die folgenden Objekte sind maskiert von 'package:base':
##
##     date, intersect, setdiff, union
```

```r
require(zoo) #for lagging
require(jtools) #tables
```

```
## Lade nötiges Paket: jtools
```

```r
require(huxtable) #tables
```

```
## Lade nötiges Paket: huxtable
```

```
##
## Attache Paket: 'huxtable'
```

```
## Das folgende Objekt ist maskiert 'package:dplyr':
##
##     add_rownames
```

```
## Das folgende Objekt ist maskiert 'package:ggplot2':
##
##     theme_grey
```

```r
require(lmtest) #reg tests
require(vroom) #for loading data
```

```
## Lade nötiges Paket: vroom
```

```r
require(data.table) #for data filtering
```

```
## Lade nötiges Paket: data.table
```

```
##
## Attache Paket: 'data.table'
```

```
## Die folgenden Objekte sind maskiert von 'package:lubridate':
##
##     hour, isoweek, mday, minute, month, quarter, second, wday, week,
##     yday, year
```

```
## Die folgenden Objekte sind maskiert von 'package:dplyr':
##
##     between, first, last

## Die folgenden Objekte sind maskiert von 'package:zoo':
##
##     yearmon, yearqtr
```

```
require(sysid) #for ARMA-X modeling
```

```
## Lade nötiges Paket: sysid

##
## Attache Paket: 'sysid'

## Das folgende Objekt ist maskiert 'package:AEC':
##
##     g

## Die folgenden Objekte sind maskiert von 'package:stats':
##
##     deltat, frequency, step, time
```

```
require(sandwhich) #regression errors
```

```
## Lade nötiges Paket: sandwhich

## Warning in library(package, lib.loc = lib.loc, character.only = TRUE,
## logical.return = TRUE, : es gibt kein Paket namens 'sandwhich'
```

```
require(stargazer) #nice reg tables
```

```
## Lade nötiges Paket: stargazer

##
## Please cite as:

##  Hlavac, Marek (2022). stargazer: Well-Formatted Regression and Summary Statistics Tables.

##  R package version 5.2.3. https://CRAN.R-project.org/package=stargazer
```

```
require(tidytext) #text mining
```

```
## Lade nötiges Paket: tidytext
```

```
require(textstem) #lemmatization
```

```
## Lade nötiges Paket: textstem
```

```
## Lade nötiges Paket: koRpus.lang.en

## Lade nötiges Paket: koRpus

## Lade nötiges Paket: sylly

## For information on available language packages for 'koRpus', run
##
##   available.koRpus.lang()
##
## and see ?install.koRpus.lang()
```

```
require(quanteda) #tokenization
```

```
## Lade nötiges Paket: quanteda

## Package version: 4.2.0
## Unicode version: 15.1
## ICU version: 74.1

## Parallel computing: 12 of 12 threads used.

## See https://quanteda.io for tutorials and examples.

##
## Attache Paket: 'quanteda'

## Die folgenden Objekte sind maskiert von 'package:koRpus':
##
##     tokens, types

## Das folgende Objekt ist maskiert 'package:zoo':
##
##     index
```

```
require(texreg) #arima tables
```

```
## Lade nötiges Paket: texreg

## Version:  1.39.4
## Date:     2024-07-23
## Author:   Philip Leifeld (University of Manchester)
##
## Consider submitting praise using the praise or praise_interactive functions.
## Please cite the JSS article in your publications -- see citation("texreg").
```

```
getwd()
```

```
## [1] "C:/Users/jonas/Desktop/repos/mmetricsproject/exploratoryanalysis"
```

```r
#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"))

data("stop_words")
stop_words_list <- stop_words$word

SPY <- read.csv(here("data/mothership", "SPY.csv"))
```

```r
posts_raw <- read.csv(here("data/mothership", "social.csv"))
```

```r
posts <- posts_raw %>% select(timestamp, tweet_text)
```

```r
posts <- posts %>%
    mutate(
      tweet_clean = str_replace_all(tweet_text, "(http[s]?://|www\\.)\\S+", ""),  # Remove URLs

      post_lower = str_to_lower(tweet_clean),  # New column with post converted to lowercase
      post_clean = str_replace_all(post_lower, "[^a-z\\s]", " ")
    )
```

```r
posts <- posts %>%
  select(-post_lower, -tweet_clean, -tweet_text)

posts$timestamp <- as.POSIXct(posts$timestamp, format = "%Y-%m-%d %H:%M:%S", tz = "EST")
```

```r
posts <- posts %>%
  mutate(hour_timestamp = floor_date(timestamp, unit = "hour")) %>%   # Round to hour
  group_by(hour_timestamp) %>%
  summarise(
    combined_text = str_c(post_clean, collapse = " "),   # Concatenate texts
    .groups = "drop"
  )

posts <- posts %>%
  rename(timestamp = hour_timestamp)
```

```r
posts2 <- posts %>%
  mutate(
    # --- Step 1: Adjust based on time ---
    hour = hour(timestamp),
    date = as.Date(timestamp),

    move_before_9 = hour < 9,
```

```
    move_after_16 = hour >= 16,

    adjusted_timestamp = case_when(
      move_before_9 ~ as.POSIXct(paste(date, "09:00:00"), format = "%Y-%m-%d %H:%M:%S", tz = tz(timestar
      move_after_16 ~ as.POSIXct(paste(date + 1, "09:00:00"), format = "%Y-%m-%d %H:%M:%S", tz = tz(time
      TRUE ~ timestamp
    ),

    # --- Step 2: Adjust for weekends on adjusted_timestamp ---
    weekday = wday(adjusted_timestamp),  # <- use adjusted_timestamp here
    final_timestamp = case_when(
      weekday == 7 ~ as.POSIXct(paste(as.Date(adjusted_timestamp) + 2, "09:00:00"), format = "%Y-%m-%d
      weekday == 1 ~ as.POSIXct(paste(as.Date(adjusted_timestamp) + 1, "09:00:00"), format = "%Y-%m-%d
      TRUE ~ adjusted_timestamp
    )
  ) %>%
  select(final_timestamp, combined_text) %>%
  rename(timestamp = final_timestamp)

posts2 <- posts2 %>%
  group_by(timestamp) %>%
  summarise(
    text = str_c(combined_text, collapse = " "),   # Concatenate texts
    .groups = "drop"
  )
```

```
start_date <- '2025-01-01'
end_date <- '2025-04-10'
```

Market Data

```
SPY$timestamp = as.POSIXct(SPY$timestamp,format = "%Y-%m-%d %H:%M:%S", tz="EST")

#find hourly volatility
#NOTE: this ignores tweets made outside trading hours!!
SPY_volatility_alltime = dplyr::select(SPY,timestamp,r_vol_h)

#aggregating per hour
SPY_volatility_alltime = SPY_volatility_alltime %>%
          mutate(timestamp = floor_date(timestamp, unit = "hour")) %>%
          distinct(timestamp, .keep_all = TRUE)

#select time period
SPY_volatility = filter(SPY_volatility_alltime,
                  between(timestamp,
                          as.Date(start_date),
                          as.Date(end_date)))
```

```
posts_token <- posts2 %>%
  mutate(
```

```
    tokens = text %>%
        str_replace_all("[^a-z\\s]", " ") %>%
        str_split("\\s+") %>%
        lapply(function(words) {
          words <- words[words != ""]  # Remove empty strings
          words <- setdiff(words, stop_words_list)  # Remove stopwords
          words <- lemmatize_words(words) # Reduces words to their base form -> is becomes be
        }

  )
  )
```

sum(wday(posts2$timestamp) %in% c(1,7))

```
# Define the tokens you want to check
specific_tokens <- c("tariff")

# Count the occurrences of specific tokens
token_count <- posts_token %>%
  pull(tokens) %>%  # Extract the 'tokens' column as a list of token vectors
  unlist() %>%      # Flatten the list into a single vector of tokens
  .[ . %in% specific_tokens ] %>%  # Filter only the tokens that match the specific ones
  length()  # Get the total count

# Print the result
print(token_count)
```

```
## [1] 281
```

```
afinn <- tidytext::get_sentiments("afinn")
```

```
## Registered S3 methods overwritten by 'readr':
##   method                     from
##   as.data.frame.spec_tbl_df vroom
##   as_tibble.spec_tbl_df     vroom
##   format.col_spec           vroom
##   print.col_spec            vroom
##   print.collector           vroom
##   print.date_names          vroom
##   print.locale              vroom
##   str.col_spec              vroom
```

```
posts_sentiment <- posts_token %>%
  unnest_tokens(word, text) %>%  # Tokenize the text
  inner_join(afinn, by = "word") %>%  # Join with the AFINN lexicon to get sentiment scores
  group_by(timestamp) %>%  # Group by timestamp to aggregate by post
  summarise(sentiment_score = sum(value, na.rm = TRUE))  # Sum up the sentiment scores for each post

# Merge the sentiment scores back into the original posts_token dataframe
posts_token_with_sentiment <- posts_token %>%
  left_join(posts_sentiment, by = "timestamp")
```

#Join analysis data

```
posts_armax = filter(posts_token_with_sentiment,
                  between(timestamp,
                          as.Date(start_date),
                          as.Date(end_date)))
```

```
analysis_data <- full_join(posts_armax, SPY_volatility, by = "timestamp")
analysis_data <- analysis_data %>% arrange(timestamp)

analysis_data <- analysis_data %>% mutate(dummy_post = if_else(!is.na(text), 1, 0))

analysis_data <- analysis_data %>%
  mutate(
    sentiment_score = ifelse(is.na(sentiment_score), 0, sentiment_score),
    tariff = ifelse(str_count(text,pattern = "tariff") >= 1, 1,0),
    tariff = if_else(is.na(tariff), 0, tariff)
  )

analysis_data_fin <- analysis_data %>% filter(r_vol_h != 0)
```

```
lag_selector(y=analysis_data_fin$r_vol_h, xreg=analysis_data_fin$tariff,
             nb.lags=8, type="text")
```

```
##
## ========================================
##                  Dependent variable:
##              ---------------------------
##                           y
##                  (no HAC)        (HAC)
##                    (1)             (2)
## ----------------------------------------
## tariff_lag_0      0.084         0.084***
##                  (0.071)         (0.028)
## tariff_lag_1     0.173**          0.173
##                  (0.072)         (0.135)
## tariff_lag_2     0.366***         0.366*
##                  (0.071)         (0.204)
## tariff_lag_3      0.110           0.110**
##                  (0.071)         (0.055)
## tariff_lag_4     0.323***         0.323*
##                  (0.071)         (0.173)
## tariff_lag_5      0.073           0.073**
##                  (0.071)         (0.037)
## tariff_lag_6      -0.009          -0.009
##                  (0.072)         (0.048)
## tariff_lag_7      0.007           0.007
##                  (0.073)         (0.045)
## tariff_lag_8      0.085           0.085
##                  (0.072)         (0.082)
## Constant          -0.011          -0.011
##                  (0.031)         (0.032)
## ----------------------------------------
```

```
## Observations      461            461
## ========================================
## Note:         *p<0.1; **p<0.05; ***p<0.01
```

```
armax(analysis_data_fin$r_vol_h,xreg=analysis_data_fin$tariff,nb.lags=4,latex=F)
```

```
##
## ============================
##                  Model 1
## ----------------------------
## ar1                0.2968 ***
##                   (0.0464)
## ar2               -0.0100
##                   (0.0493)
## ar3                0.0351
##                   (0.0686)
## ar4                0.1173
##                   (0.0707)
## ar5                0.0236
##                   (0.0677)
## intercept          0.0316
##                   (0.0470)
## tariff_lag_0       0.0601
##                   (0.0677)
## tariff_lag_1       0.1512 *
##                   (0.0725)
## tariff_lag_2       0.3326 ***
##                   (0.0726)
## tariff_lag_3       0.0758
##                   (0.0738)
## tariff_lag_4       0.2707 ***
##                   (0.0679)
## ----------------------------
## AIC               601.4179
## AICc              602.1081
## BIC               651.1223
## Log Likelihood   -288.7089
## Num. obs.         465
## ============================
## *** p < 0.001; ** p < 0.01; * p < 0.05
```