Data Management

Contents

Setup

```
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(syuzhet) #sentiment analysis
require(purrr) #map functions
require(tidyr) #more data stuff
require(alphavantager)
data("stop_words")
stop_words_list <- stop_words$word
av_api_key(Sys.getenv("ALPHAVANTAGE_API_KEY"))

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"))
```

Raw Data

First, we load our raw data, found in the market_data and political_data folders. These are the uncleaned, unprepared, unedited original data files as downloaded originally. Note that we use a custom function to load the financial data since the files are separated by month due to restrictions while downloading. This function also then merges every month and year. The sources of the financial data is the API for AlphaVantage (find instructions on how to use at the end of this document). For the Twitter posts, we downloaded it from Kaggle. For the TruthSocial posts, we had to go and scrape the data ourselves (see the scraper function for details).
Our data ends on the 7th of May 2025 and starts at 2009 for Twitter, and 2014 for the financial data.

```
# 1. Political

#truthsocial
raw_truths <- read.csv(here("data/political_data", "truths_new.csv"))
raw_truths <- read.csv(here("data/political_data", "truths250510.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")
```

Cleanup

We then begin the long process of cleaning this data. This is obviously heavier for the social media data, as we must remove retweets, media posts, etc. We use a custom function for the processing of the TruthSocial posts. We then merge these two datasets in order to get the full social data. Note that there are no observations for 2021 to 2022 as Trump was banned from Twitter and hadn't yet created TruthSocial.
As for the financial data, nothing really is needed as it comes from a more traditional data source.

```
# 1. Tweets
tweets = raw_tweets

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

#cleanup
tweets = data.frame(tweets$date,tweets$text)
colnames(tweets) = c("timestamp","tweet_text")
tweets$timestamp = as.POSIXct(tweets$timestamp,format = "%Y-%m-%d %H:%M:%S")
second(tweets$timestamp) = 0


# 2. Truths
truthsbackup <- truths_processer(raw_truths)
truths = truthsbackup

#cleanup
truths = data.frame(truths$date_time_parsed,truths$post)
colnames(truths) = c("timestamp","truths_text")
truths$timestamp = as.POSIXct(truths$timestamp,format = "%Y-%m-%d %H:%M:%S")
second(truths$timestamp) = 0

#note: we are keeping tweets of just an image/video/link in order to have
#them in the tweet count and tweet dummy

# Merging social media data since it is not overlapping
names(truths)[names(truths) == 'truths_text'] <- 'tweet_text'
social = rbind(tweets,truths)
social <- social[order(social$timestamp, decreasing=F), ]

# Round 2 cleanup
social <- social %>%
    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]", " "))
social <- social %>%
  select(-post_lower, -tweet_clean, -tweet_text)
names(social)[names(social) == 'post_clean'] <- 'tweet_text'
social$timestamp <- as.POSIXct(social$timestamp, format = "%Y-%m-%d %H:%M:%S")
#select time period
social = filter(social,between(timestamp,
                           as.Date('2009-01-01'),
```

```
                            as.Date('2025-12-12')))

#for plot
truths_plot <- social %>%
  mutate(
    # Use POSIX 'timestamp' directly
    date_time_parsed = as.POSIXct(timestamp, format = "%Y-%m-%d %H:%M:%S"),

    # Extract date only for plot
    day = as.Date(date_time_parsed),

    # Extract time only for plot
    time = format(date_time_parsed, "%H:%M"),

    # Convert time to numeric hours & minutes as fractions
    time_numeric = hour(date_time_parsed) + minute(date_time_parsed) / 60,

    # Shift time such that y = 0 corresponds to 12 PM
    time_shifted = time_numeric - 12)


# 3. Financial

#remove index
SPY = raw_SPY[-1]
VGK = raw_VGK[-1]
ASHR = raw_ASHR[-1]
```

Here's the plot we use in our introduction, showing the endless stream of Trump's social media presence.

```
 #Create the scatter plot
ggplot(truths_plot, aes(x = day, y = time_shifted)) +
  geom_point(alpha = 0.5, color = "blue", shape = ".") +  # Transparancy to create "heatmap"
  scale_y_continuous(
    breaks = seq(-12, 12, by = 3),  # Custom Y scale
    labels = c("00:00", "03:00", "06:00", "09:00", "12:00", "15:00", "18:00", "21:00", "24:00")  # 24-h
  ) +
  labs(title = "Terminally Online: Trumps Twitter & Truth Social Posts (EDT)",
       x = "",
       y = "Time of Day") +
  theme_minimal() +


  # Customize X Axis
  scale_x_date(
    date_labels = "%b %Y",  # Format labels to show month and year
    date_breaks = "9 months"
  ) +

  theme(axis.text.x = element_text(angle = 45, hjust = 1)) +

  # Add vertical lines at 9:30 AM and 4:00 PM for stock market
  geom_hline(yintercept = (9 + 30 / 60) - 12, linetype = "longdash", color = "red") +
```

```r
  geom_hline(yintercept = 16 - 12, linetype = "dashed", color = "red") +

  # theme adjustments
  theme(
    panel.grid.minor = element_blank(),  # Remove minor gridlines
    panel.grid.major = element_line(linewidth = 0.5),  # Major gridlines
    axis.title = element_text(size = 12),
    axis.text = element_text(size = 10))
```

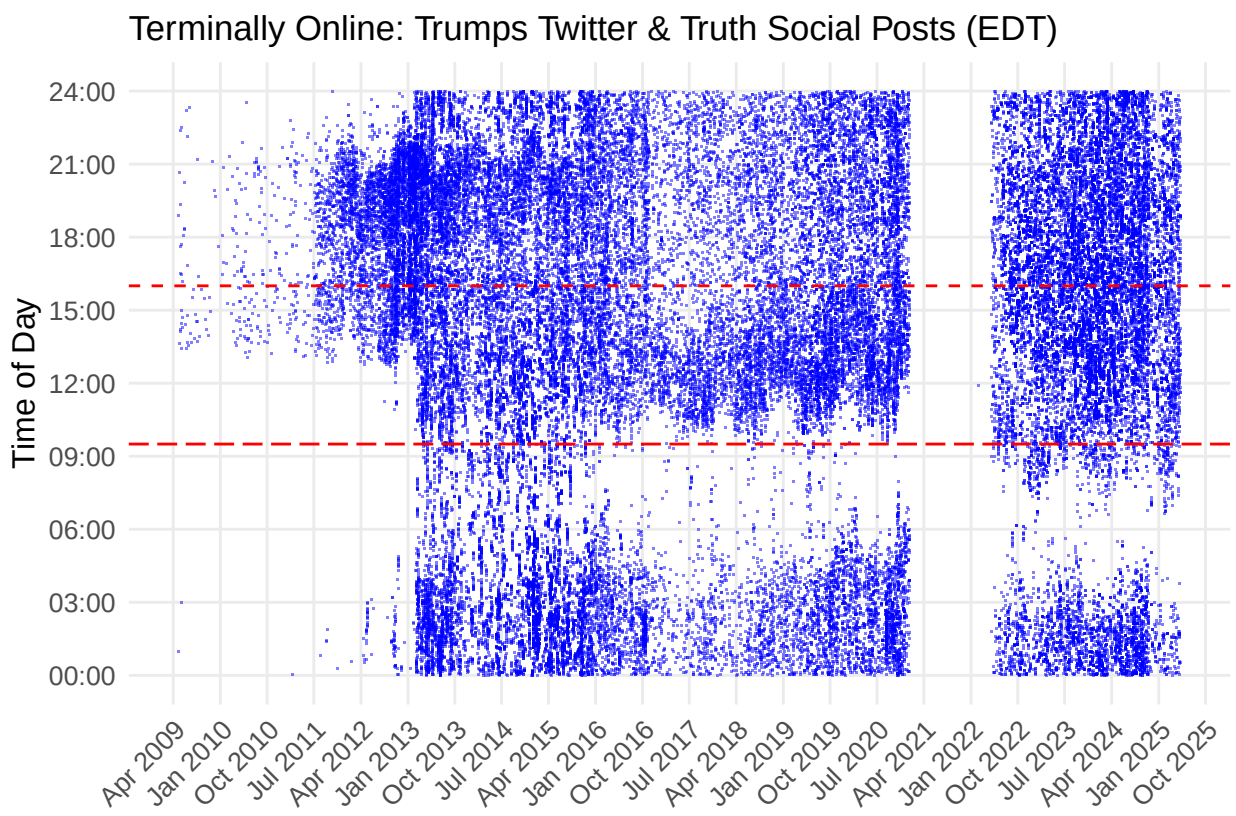

Figure 1: Terminally Online: Trump's Twitter & Truth Social Posts (EDT)

Building Additional Variables

After having cleaned the data, we can build additional variables which will be used in later analysis.

Volatility By Hour

First, we construct our main variable of observation: realised market volatility. We use a custom function that loops other custom functions in order to get the average hourly volatility for each open market hour. Note that the first hour is from 9:30 am to 10:00 am since the market open on a half-hour but closes at 4:00 pm. The formula is the following:

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

where Δp_t is the difference in price (open - close)

and i represents every minute

We do so for our three markets of interest: SPY represents the S&P500, VGK represents a broad Europe market index (FTSE Developed Europe All Cap), and ASHR does so for China as well (CSI 300 China). We store these volatilites alongside the original stock data where we have the timestamp, prices, volume, and now average hourly volatility.

```
SPY = r.vol_hourly(SPY,merge=T)
VGK = r.vol_hourly(VGK,merge=T)
ASHR = r.vol_hourly(ASHR,merge=T)
```

Social Media Post Count

We construct a variable for the number of tweets per hour.

```
#convert to datatable
social = as.data.table(social)

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

#fix timestamp by hour
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 = dplyr::select(tweet_count, timestamp, N)
tweet_count = tweet_count[ order(tweet_count$timestamp , decreasing = F ),]
```

Adding Dummy for Social Media Post

This variable is a dummy for whether there has been a Trump tweet in a certain hour.

```r
#using post count we create dummy
social_hourly = tweet_count %>% mutate(dummy = if_else(N > 0, 1, 0))
```

Sentiment Analysis

We use machine-learning tools with pre-built dictionaries in order to find sentiments for the actual content of the social media posts. We then aggregate them by hour.

```r
#sentiment analysis on post text
nrc_scores <- get_nrc_sentiment(social$tweet_text)

#add to main social dataframe
social <- bind_cols(social, nrc_scores)

#aggregate by hour
sent_hour <- social %>%
  mutate(timestamp = floor_date(timestamp, unit = "hour")) %>%
  group_by(timestamp) %>%
  summarise(across(anger:positive, sum), .groups = 'drop')
sent_hour = as.data.frame(sent_hour)

#get proportion of sentiment for each post
social <- social %>%
  mutate(total_sentiment = anger + anticipation + disgust + fear +
           joy + sadness + surprise + trust,
         total_posneg = positive + negative,
    prop_anger = anger / total_sentiment,
    prop_anticipation = anticipation / total_sentiment,
    prop_disgust = disgust / total_sentiment,
    prop_fear = fear / total_sentiment,
    prop_joy = joy / total_sentiment,
    prop_sadness = sadness / total_sentiment,
    prop_surprise = surprise / total_sentiment,
    prop_trust = trust / total_sentiment,
    prop_negative = negative / total_posneg,
    prop_positive = positive / total_posneg)
social[is.na(social)] <- 0

#same but hourly
sent_hour <- sent_hour %>%
  mutate(total_sentiment = anger + anticipation + disgust + fear +
           joy + sadness + surprise + trust,
         total_posneg = positive + negative,
    prop_anger = anger / total_sentiment,
    prop_anticipation = anticipation / total_sentiment,
    prop_disgust = disgust / total_sentiment,
    prop_fear = fear / total_sentiment,
    prop_joy = joy / total_sentiment,
    prop_sadness = sadness / total_sentiment,
    prop_surprise = surprise / total_sentiment,
    prop_trust = trust / total_sentiment,
    prop_negative = negative / total_posneg,
```

```
    prop_positive = positive / total_posneg)
sent_hour[is.na(sent_hour)] <- 0
social_hourly <- left_join(social_hourly, sent_hour, by="timestamp")
```

Important Words

We now build variables for certain key words. We also aggregate these at the hourly level, meaning we get how many times the mention of a word appears in posts of that hour.
The first word is tariff, and its variants.

```
#create dummy for mention of tariff
tariff = str_count(social$tweet_text,pattern = "tariff|tariffs")
Tariff = str_count(social$tweet_text,pattern = "Tariff|Tariffs")
tariff = tariff + Tariff
social$tariff <- tariff

#counts number of tweets mentioning tariffs per hour
tariff_hour <- social %>%
  mutate(timestamp = floor_date(timestamp, unit = "hour")) %>%
  group_by(timestamp) %>%
  summarise(total_tariff = sum(tariff), .groups = "drop")
tariff_hour = as.data.frame(tariff_hour)
social_hourly <- left_join(social_hourly, tariff_hour, by="timestamp")
```

The second word is trade.

```
#create dummy for mention of trade
trade = str_count(social$tweet_text,pattern = "trade")
Trade = str_count(social$tweet_text,pattern = "Trade")
trade = trade + Trade
social$trade <- trade

#counts number of tweets mentioning tariffs per hour
trade_hour <- social %>%
  mutate(timestamp = floor_date(timestamp, unit = "hour")) %>%
  group_by(timestamp) %>%
  summarise(total_trade = sum(trade), .groups = "drop")
trade_hour = as.data.frame(trade_hour)
social_hourly <- left_join(social_hourly, trade_hour, by="timestamp")
```

The third is the mention of China.

```
#create dummy for mention of trade
china = str_count(social$tweet_text,pattern = "china")
China = str_count(social$tweet_text,pattern = "China")
china = China + china
social$china <- china

#counts number of tweets mentioning tariffs per hour
china_hour <- social %>%
  mutate(timestamp = floor_date(timestamp, unit = "hour")) %>%
```

```
  group_by(timestamp) %>%
  summarise(total_china = sum(china), .groups = "drop")
china_hour = as.data.frame(china_hour)
social_hourly <- left_join(social_hourly, china_hour, by="timestamp")
```

Non-Market Hours

A big problem we had in our analysis was what to do with social media posts which appeared outside market hours. On weekends, holidays as well. We first decided to simply ignore them, but it turned out to remove a lot of observations, leaving us with less than 100 mentions of tariffs against thousands of data points for market volatility. We finally decided to push all the social media information outside market hours to the next open hour. This comes as an assumption. That is, if Trump tweets on Good Friday (market holiday), then the market will only react to this new information on Monday at 9:30 am. Given this, it is possible that the first effect of our variables might be biased due to increased volatility in the morning (market reacts to all other information released during the weekend). See the analysis to check whether this happens.

In order implement this approach, we first search for hours that have social data but that have no observations for financial data. Then, we dynamically search for the next open hour. Finally, we add a new column with the adjusted time.

Note that this is still not the final step for our data. We then have to aggregate these data points for each adjusted hour. We do so in the "full_data" script found in our helper functions. This is done last in order to enable us to still do the ignore method if required.

```
#find hourly volatility (to get open hours)
SPY_volatility = dplyr::select(SPY,timestamp,r_vol_h)

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

#just to make sure (often causes problems)
tz = "EST"
social_hourly$timestamp = as.POSIXct(social_hourly$timestamp,
                                     format = "%Y-%m-%d %H:%M:%S")
social_hourly$timestamp <- force_tz(social_hourly$timestamp, tzone = tz)
SPY_volatility$timestamp = as.POSIXct(SPY_volatility$timestamp,
                                      format = "%Y-%m-%d %H:%M:%S")
SPY_volatility$timestamp <- force_tz(SPY_volatility$timestamp, tzone = tz)

#this searches for the times where there are tweets and no value for r.vol
valid_market_hours <- sort(unique(SPY_volatility$timestamp))

# Function to get next market hour for each timestamp
find_next_market_hour <- function(ts) {
  # Find the next market hour that is >= the current timestamp
  next_valid <- valid_market_hours[valid_market_hours >= ts][1]

  # Convert it to POSIXct to maintain correct date-time format
  return(as.POSIXct(next_valid, tz = "EST"))
}

# Use purrr::map to apply the function and ensure POSIXct output
social_hourly <- social_hourly %>%
  mutate(adjusted_time = map(timestamp, find_next_market_hour)) %>%
  unnest(adjusted_time)  # Unnest the list column created by map
```

```r
#aggregate by adjusted time
aggregated <- social_hourly %>%
  group_by(adjusted_time) %>%
  summarise(across(where(is.numeric), sum, .names = "{.col}"), .groups = "drop") %>%
  rename(timestamp = adjusted_time)

#reorder and clean columns
social_hourly <- social_hourly %>%
  select(
    timestamp,
    adjusted_time,
    everything())
```

Data Save

We now save this data, in multple files as otherwise the file size is too large to be uploaded to GitHub. Keep in mind that this is not the final data set used for analysis, and that there a few more transformations occurring in our full_data script you can see below.

```
#financial
write.csv(SPY, here("data/mothership/SPY.csv"), row.names=F)
write.csv(VGK, here("data/mothership/VGK.csv"), row.names=F)
write.csv(ASHR, here("data/mothership/ASHR.csv"), row.names=F)

#social media
write.csv(social, here("data/mothership/social.csv"), row.names=F)
write.csv(social_hourly, here("data/mothership/socialhourly.csv"), row.names=F)
```

Full Data Function (see helperfunctions)

Here is a more readable version of the final full_data script we use to load the actual final (beautiful) dataset. Note that there might be a few tiny differences, and if you want the actual script we use, please find it in the helperfunctions folder.

Load Base Data

We first load the close-to-final data.

```
# 1. Load Political Social Media

#contains posts from Twitter & TruthSocial
social <- read.csv(here("data/mothership", "social.csv"))

social_hourly <- read.csv(here("data/mothership", "socialhourly.csv"))


# 2. Load Financial

#S&P500
SPY <- read.csv(here("data/mothership", "SPY.csv"))

#STOXX50
VGK <- read.csv(here("data/mothership", "VGK.csv"))

#CSI 300 (China)
ASHR <- read.csv(here("data/mothership", "ASHR.CSV"))
```

We make sure of the time format.

```
#make posixct
SPY$timestamp = as.POSIXct(SPY$timestamp,format = "%Y-%m-%d %H:%M:%S")
VGK$timestamp = as.POSIXct(VGK$timestamp,format = "%Y-%m-%d %H:%M:%S")
ASHR$timestamp = as.POSIXct(ASHR$timestamp,format = "%Y-%m-%d %H:%M:%S")
social$timestamp = as.POSIXct(social$timestamp,format = "%Y-%m-%d %H:%M:%S")
social_hourly$timestamp = as.POSIXct(social_hourly$timestamp,format = "%Y-%m-%d %H:%M:%S")
social_hourly$adjusted_time = as.POSIXct(social_hourly$adjusted_time,format = "%Y-%m-%d %H:%M:%S")
```

Volatility

We then proceed to select the volatility values we found for each market. We can also plot them using our helpful custom hourly volatility plot functions!

```
#find hourly volatility
SPY_volatility = dplyr::select(SPY,timestamp,r_vol_h)

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

```
#plot
hvol_plotter(SPY_volatility,breaks="3 month",
             title="Realised Volatility - SPY")
```

```
#find hourly volatility
VGK_volatility = dplyr::select(VGK,timestamp,r_vol_h)

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

```
#find hourly volatility
ASHR_volatility = dplyr::select(ASHR,timestamp,r_vol_h)

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

Number of Posts

We then load all our built social variables. We can also plot the number of times Mr. Trump posts on social media. It is quite a lot.

```
#find count
tweetcount = dplyr::select(social_hourly,timestamp,adjusted_time,N)

#for taking count of closed market hours
tweetcount2 <- tweetcount %>%
  group_by(adjusted_time) %>%
  summarise(N = sum(N))

#plot
ggplot(tweetcount, aes(x = timestamp, y = N)) +
    geom_point(color = "#253494", size = 1) +
    scale_x_datetime(date_labels = "%b %Y", date_breaks = "3 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))
```

Dummy for Social Media Post

```
#find dummy
tweetdummy = dplyr::select(social_hourly,timestamp,adjusted_time,dummy)
```

```
#for taking count of closed market hours
tweetdummy2 <- tweetdummy %>%
  group_by(adjusted_time) %>%
  summarise(dummy = sum(dummy))
#peculiar interpretation for dummy: if dummy>1 it means that there were x
#out-hours which had tweets in them
```

Number of Tweets Mentioning Tariffs

```
#find count
tariff = dplyr::select(social_hourly,timestamp,adjusted_time,total_tariff)

#for taking count of closed market hours
tariff2 <- tariff %>%
  group_by(adjusted_time) %>%
  summarise(tariff = sum(total_tariff))
```

Number of Tweets Mentioning Trade

```
#find count
trade = dplyr::select(social_hourly,timestamp,adjusted_time,total_trade)

#for taking count of closed market hours
trade2 <- trade %>%
  group_by(adjusted_time) %>%
  summarise(trade = sum(total_trade))
```

Number of Tweets Mentioning China

```
#find count
china = dplyr::select(social_hourly,timestamp,adjusted_time,total_china)

#for taking count of closed market hours
china2 <- china %>%
  group_by(adjusted_time) %>%
  summarise(china = sum(total_china))
```

Sentiments

```
#find counts
anger = dplyr::select(social_hourly,timestamp,adjusted_time,anger)
anticipation = dplyr::select(social_hourly,timestamp,adjusted_time,anticipation)
disgust = dplyr::select(social_hourly,timestamp,adjusted_time,disgust)
fear = dplyr::select(social_hourly,timestamp,adjusted_time,fear)
joy = dplyr::select(social_hourly,timestamp,adjusted_time,joy)
```

```
sadness = dplyr::select(social_hourly,timestamp,adjusted_time,sadness)
surprise = dplyr::select(social_hourly,timestamp,adjusted_time,surprise)
trust = dplyr::select(social_hourly,timestamp,adjusted_time,trust)
total_sentiment = dplyr::select(social_hourly,timestamp,adjusted_time,total_sentiment)

anger2 <- anger %>%
  group_by(adjusted_time) %>%
  summarise(anger = sum(anger))
anticipation2 <- anticipation %>%
  group_by(adjusted_time) %>%
  summarise(anticipation= sum(anticipation))
disgust2 <- disgust %>%
  group_by(adjusted_time) %>%
  summarise(disgust = sum(disgust))
fear2 <- fear %>%
  group_by(adjusted_time) %>%
  summarise(fear = sum(fear))
joy2 <- joy %>%
  group_by(adjusted_time) %>%
  summarise(joy = sum(joy))
sadness2 <- sadness %>%
  group_by(adjusted_time) %>%
  summarise(sadness = sum(sadness))
surprise2 <- surprise %>%
  group_by(adjusted_time) %>%
  summarise(surprise = sum(surprise))
trust2 <- trust %>%
  group_by(adjusted_time) %>%
  summarise(trust = sum(trust))
total_sentiment2 <- total_sentiment %>%
  group_by(adjusted_time) %>%
  summarise(total_sentiment = sum(total_sentiment))
```

Positive/Negative

```
#positive
positive = dplyr::select(social_hourly,timestamp,adjusted_time,positive)

positive2 <- positive %>%
  group_by(adjusted_time) %>%
  summarise(positive = sum(positive))

#negative
negative = dplyr::select(social_hourly,timestamp,adjusted_time,negative)

negative2 <- negative %>%
  group_by(adjusted_time) %>%
  summarise(negative = sum(negative))

#total
total_posneg = dplyr::select(social_hourly,timestamp,adjusted_time,total_posneg)
```

```
total_posneg2 <- total_posneg %>%
  group_by(adjusted_time) %>%
  summarise(total_posneg = sum(total_posneg))
```

Merge

This part is a bit tough. The first few lines were used when we were using the "ignore" method for the non-market hours. All the rest of the code chunk is our current method of pushing the data to the next open hour. It requires quite a few adjustements, but this is the true final step in getting the data we actually use in our analysis.

```
#merge our dependant and independant vars

#case 1: ignore tweets outside trading hours
armax_data = left_join(SPY_volatility, VGK_volatility, by="timestamp")
armax_data = left_join(armax_data, ASHR_volatility, by="timestamp")
armax_data = left_join(armax_data, select(tweetdummy, -adjusted_time), by="timestamp")
armax_data = left_join(armax_data, select(tweetcount, -adjusted_time), by="timestamp")
armax_data = left_join(armax_data, select(tariff, -adjusted_time), by="timestamp")
armax_data = left_join(armax_data, select(trade, -adjusted_time), by="timestamp")
armax_data = left_join(armax_data, select(china, -adjusted_time), by="timestamp")

rm(armax_data)
#case 2: push tweets made outside market hours to the next open hour
armax_data = left_join(SPY_volatility, VGK_volatility, by="timestamp")
armax_data = left_join(armax_data, ASHR_volatility, by="timestamp")
armax_data <- armax_data %>%
  left_join(tweetdummy2, by = c("timestamp" = "adjusted_time"))
armax_data <- armax_data %>%
  left_join(tweetcount2, by = c("timestamp" = "adjusted_time"))
armax_data <- armax_data %>%
  left_join(tariff2, by = c("timestamp" = "adjusted_time"))
armax_data <- armax_data %>%
  left_join(trade2, by = c("timestamp" = "adjusted_time"))
armax_data <- armax_data %>%
  left_join(china2, by = c("timestamp" = "adjusted_time"))

#load sentiment counts
armax_data <- armax_data %>%
  left_join(positive2, by = c("timestamp" = "adjusted_time"))
armax_data <- armax_data %>%
  left_join(negative2, by = c("timestamp" = "adjusted_time"))
armax_data <- armax_data %>%
  left_join(total_posneg2, by = c("timestamp" = "adjusted_time"))
armax_data <- armax_data %>%
  left_join(anger2, by = c("timestamp" = "adjusted_time"))
armax_data <- armax_data %>%
  left_join(anticipation2, by = c("timestamp" = "adjusted_time"))
armax_data <- armax_data %>%
  left_join(disgust2, by = c("timestamp" = "adjusted_time"))
armax_data <- armax_data %>%
  left_join(fear2, by = c("timestamp" = "adjusted_time"))
```

```r
armax_data <- armax_data %>%
  left_join(joy2, by = c("timestamp" = "adjusted_time"))
armax_data <- armax_data %>%
  left_join(sadness2, by = c("timestamp" = "adjusted_time"))
armax_data <- armax_data %>%
  left_join(surprise2, by = c("timestamp" = "adjusted_time"))
armax_data <- armax_data %>%
  left_join(trust2, by = c("timestamp" = "adjusted_time"))
armax_data <- armax_data %>%
  left_join(total_sentiment2, by = c("timestamp" = "adjusted_time"))

#convert NA to zeroes
armax_data[is.na(armax_data)] <- 0

#get proportions
armax_data <- armax_data %>%
  mutate(
    prop_anger = anger / total_sentiment,
    prop_anticipation = anticipation / total_sentiment,
    prop_disgust = disgust / total_sentiment,
    prop_fear = fear / total_sentiment,
    prop_joy = joy / total_sentiment,
    prop_sadness = sadness / total_sentiment,
    prop_surprise = surprise / total_sentiment,
    prop_trust = trust / total_sentiment,
    prop_negative = negative / total_posneg,
    prop_positive = positive / total_posneg)

#convert NA to zeroes again (cause divided by zeroes)
armax_data[is.na(armax_data)] <- 0

#remove non-proportion sentiments
armax_data <- armax_data %>%
  select(-anger, -anticipation, -disgust, -fear, -joy, -sadness, -surprise, -trust,
         -total_sentiment, -positive, -negative, -total_posneg)

#rename volatility columns
names(armax_data)[2] <- "SPY_vol"
names(armax_data)[3] <- "VGK_vol"
names(armax_data)[4] <- "ASHR_vol"

#rename data for general analysis
data = armax_data

#keep relevant data & remove everything else
rm(list = setdiff(ls(), "data"))
```

Using Alpha Vantage API

Here is some documentation for how to use the AlphaVantage API in order to get the intraday minute-by-minute financial data we use in our project. Each call can download a maximum of a whole month of data. We had to create a loop function to easily loop the calls in order to get a full year's worth of data in one go. Note that we had a limit for how many calls we could make a day (maximum 25 months of data per day), so the "daily API call" briefly became part of our morning routines for a few weeks.

```
library(alphavantager)

av_api_key(Sys.getenv("ALPHAVANTAGE_API_KEY"))

#for past month
data=av_get("ASHR", av_fun = "TIME_SERIES_INTRADAY", interval = "1min",
              adjusted="false", extended_hours="false", outputsize = "full")

#for a particular month
data2=av_get("SPY", av_fun = "TIME_SERIES_INTRADAY", interval = "1min",
              adjusted="false", extended_hours="false",
              month="2025-04", outputsize = "full") #create loop for more



write.csv(data,"~/ASHR.csv", row.names = T) #saves to documents
write.csv(data2,"~/SPY-2025-04.csv", row.names = T)
```

```
library(alphavantager)

av_api_key(Sys.getenv("ALPHAVANTAGE_API_KEY"))

year = "2022"
months = c("01","02","03","04","05","06","07","08","09","10","11","12")
market = "SPY"

for (t in 1:length(months)) {
    date = paste(year, months[t], sep="-")
dataloop = av_get(market, av_fun="TIME_SERIES_INTRADAY",interval="1min",
                  adjusted="false", extended_hours="false",
                  month=date, outputsize="full")
filename = paste(market,date,sep="-")
filename = paste("~/", filename, sep="")
filename = paste(filename,".csv",sep="")
write.csv(dataloop,filename)
}
```

Tutorials

Manual S&P500: https://cafim.sssup.it/~giulio/other/alpha_vantage/index.html#orgaaf54ef

AlphaVantageR Tutorial: https://github.com/business-science/alphavantager/blob/master/man/av_get.Rd

Intra-Day Analysis: https://arxiv.org/html/2406.17198v1

Symbols Explanation

- ONEQ = NASDAQ Composite
- SPY = S&P500
- SMI = Swiss Market Index
- VTHR = Russell 3000 (US)
- VTI = CRSP US Total Market Index
- VGK = FTSE Developed Europe All Cap Index
- ASHR = CSI 300 China