Thesis

pderoock

2024-04-04

```
## Installing package manager
if(any("pacman" %in% rownames(installed.packages()) == FALSE)){
  install.packages("pacman", dependencies = TRUE)}

## Required packages
pacman::p_load(
  utils,
  png,
  reticulate,
  readr,
  tidyr,
  stringr,
  dplyr,
  ggplot2,
  data.table,
  nloptr,
  e1071,
  lubridate
)
```

```
## Loading in data
Datastream_DailyPrice <- read_csv("DL_DATASTREAM.csv", show_col_types = FALSE) %>%
  rename(Date = `Exchange Date`,
         LogReturn = '%Chg') %>%
  filter(!is.na(Close)) %>% # Removing result that do not have an end-of-day price
  mutate(LogReturn = as.numeric(sub("%", "", LogReturn)) / 100) %>% # Removing '%' sign and dividing by
  mutate(Date = as.Date(Date, format = "%d-%b-%Y")) %>%
  mutate(Index = n() + 1 - row_number()) %>% # Creating an index for the event study
  select(Index, Date, Close, LogReturn) # Selecting the columns from the original dataset needed
```

```
## Warning: One or more parsing issues, call `problems()` on your data frame for details,
## e.g.:
##   dat <- vroom(...)
##   problems(dat)
```

```
## Creating a sub-sample that only contains rows that have a log return
DailyLogReturn <- Datastream_DailyPrice %>%
  filter(!is.na(LogReturn))
print(head(DailyLogReturn, 10))
```

```
## # A tibble: 10 x 4
```

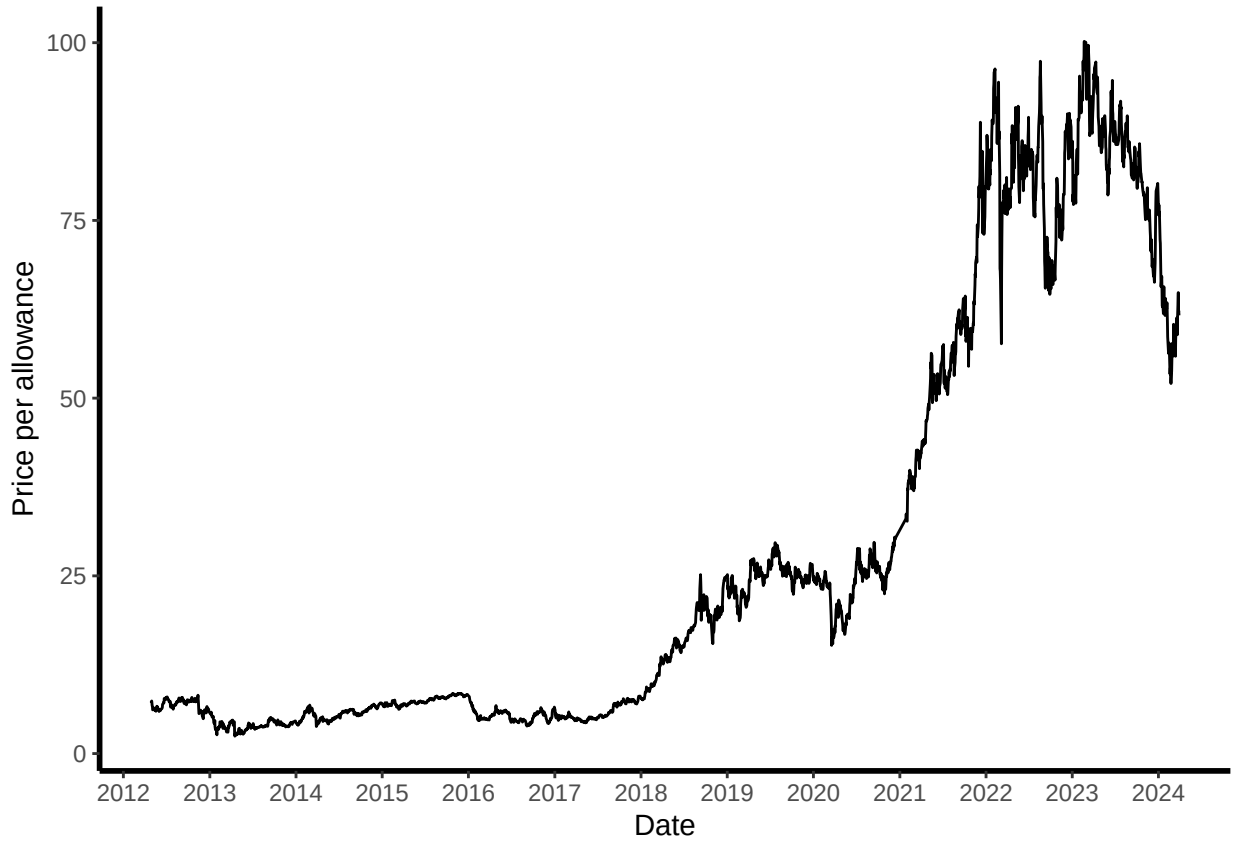
```
##    Index Date       Close LogReturn
##    <dbl> <date>     <dbl>     <dbl>
##  1  2989 2024-03-28  61.6   -0.0103
##  2  2988 2024-03-27  62.3    0.0026
##  3  2987 2024-03-26  62.1   -0.0419
##  4  2986 2024-03-25  64.8    0.0569
##  5  2985 2024-03-22  61.4    0.0414
##  6  2984 2024-03-21  58.9   -0.0256
##  7  2983 2024-03-20  60.5   -0.0036
##  8  2982 2024-03-19  60.7   -0.0096
##  9  2981 2024-03-18  61.3    0.0344
## 10  2980 2024-03-15  59.2    0.0161
```

1. Visual representation of the data

```
## Sample event data
amendment_data <- fread("amendment_data.csv", encoding = 'UTF-8') %>%
  mutate(Date = as.Date(paste(20, substr(Date, 7, 8), "-", substr(Date, 4, 5), "-", substr(Date, 1, 2),
  tidyr::separate_longer_delim(Institution, delim = ",") %>%
  mutate(Institution = stringr::str_trim(Institution))

## Creating a copy of 'DailyLogReturn' which we can safely mutate to perform the event study
event_study <- DailyLogReturn %>%
  left_join(amendment_data, by = "Date") %>%
  select(Index, Date, Close, LogReturn, Include, Single, First, Last, Year, Form, Institution, Amendmen
  mutate(EventDay = !is.na(Institution),
         ControlPeriod = NA,
         EventPeriod = NA,
         EstimationPeriod = NA
  )
```

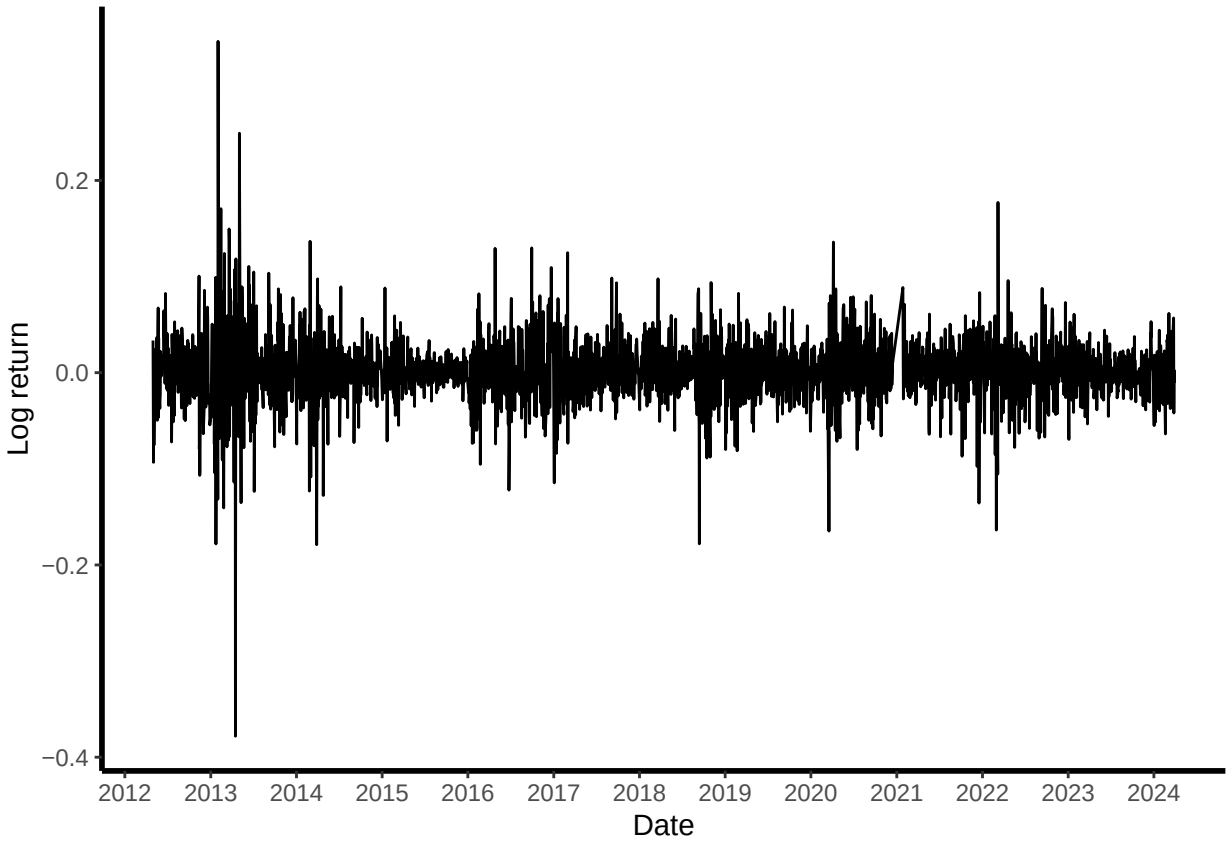
```
## EUAA Future Price -- Evolution over time
ggplot(Datastream_DailyPrice, aes(x = Date, y = Close)) +
  geom_line() +
  labs(x = "Date", y = "Price per allowance") +
  theme_bw() +
  theme(
    panel.grid.major = element_blank(),
    panel.grid.minor = element_blank(),
    panel.border = element_blank(),
    axis.line = element_line(color = "black", size = 1),
    axis.line.x = element_line(),
    axis.line.y = element_line(color = "black", size = 1)
  ) +
  scale_x_date(date_breaks = "1 year", date_labels = "%Y")
```

```
## Warning: The ‘size‘ argument of ‘element_line()‘ is deprecated as of ggplot2 3.4.0.
## i Please use the ‘linewidth‘ argument instead.
## This warning is displayed once every 8 hours.
## Call ‘lifecycle::last_lifecycle_warnings()‘ to see where this warning was
## generated.
```




```
#ggsave("EUAA_Future_End-of-Day_Price_Evolution.png", plot = last_plot(), width = 6, height = 4, dpi =
```

```
## EUAA Future log returns over time
ggplot(DailyLogReturn, aes(x = Date, y = LogReturn)) +
  geom_line() +
  labs(x = "Date", y = "Log return") +
  theme_bw() +
  theme(
    panel.grid.major = element_blank(),
    panel.grid.minor = element_blank(),
    panel.border = element_blank(),
    axis.line = element_line(color = "black", size = 1),
    axis.line.x = element_line(),
    axis.line.y = element_line(color = "black", size = 1)
  ) +
  scale_x_date(date_breaks = "1 year", date_labels = "%Y")
```




```
#ggsave("EUAA_Future_Log_Returns.png", plot = last_plot(), width = 6, height = 4, dpi = 300)
remove(amendment_data, DailyLogReturn, Datastream_DailyPrice)
```

2. Calculating the average, abnormal return on each event day

```
## First, we create a control period -- an event period around all the announcements that are relevant
base <- 0
for (i in 1:nrow(event_study)) {
  if (event_study$EventDay[i]) {
    if (event_study$Index[i] != base) {
      base <- event_study$Index[i]
      idx <- max(base - 5, 0)
      for (j in seq(idx, idx + 10)) {
        event_study$ControlPeriod[event_study$Index == j] <- 1
      }
    }
  }
}
remove(base, i, idx, j)
```

```
## Second, we create the event period for the announcements that we're using
return_ALL <- list()
```

```
## Institution
return_Commission <- list()
return_Council <- list()
return_Parliament <- list()
return_Directorate <- list()

## Type
return_Directive <- list()
return_Communication <- list()
return_Regulation <- list()

return_First <- list()
return_Last <- list()
return_Single <- list()
return_Development <- list()

## Phase
return_PhaseIII <- list()
return_PhaseIV <- list()

bases <- list()
base <- 0
for (i in 1:nrow(event_study)) {
  if (event_study$EventDay[i]) {
    if (event_study$Include[i] == 1) {
      institute <- event_study$Institution[i]
      form <- event_study$Form[i]
      first <- event_study$First[i]
      last <- event_study$Last[i]
      single <- event_study$Single[i]
      year <- event_study$Year[i]

      if (event_study$Index[i] != base) {
        base <- event_study$Index[i]

        if (!(i %in% unique(bases))) {
          return <- list()
          idx <- max(base - 5, 0)

          for (j in seq(idx, idx + 10)) {
            event_study$EventPeriod[event_study$Index == j] <- 1

            # The [1] is needed as there can be multiple rows with the same index --
            # this ensures that each index is only added once.
            return <- c(return, event_study$LogReturn[event_study$Index == j][1])
          }

          for (k in return) {
            temp_data <- data.frame(Base = base, Return = k)

            ## We now append 'temp_data' to the appropriate lists.
            return_ALL[[length(return_ALL) + 1]] <- temp_data
```

```
        ## Type of announcement
        if (form == "Directive") {
          return_Directive[[length(return_Directive) + 1]] <- temp_data
        }

        if (form == "Communication") {
          return_Communication[[length(return_Communication) + 1]] <- temp_data
        }

        if (form == "Regulation") {
          return_Regulation[[length(return_Regulation) + 1]] <- temp_data
        }

        if (first == TRUE) {
          return_First[[length(return_First) + 1]] <- temp_data
        }

        if (last == TRUE) {
          return_Last[[length(return_Last) + 1]] <- temp_data
        }

        if (single == TRUE) {
          return_Single[[length(return_Single) + 1]] <- temp_data
        }

        if (!first && !last && !single) {
          return_Development[[length(return_Development) + 1]] <- temp_data
        }

        if (year %in% c(2013, 2014, 2015, 2016, 2017, 2018, 2019, 2020)) {
          return_PhaseIII[[length(return_PhaseIII) + 1]] <- temp_data
        }

        if (year %in% c(2021, 2022, 2023, 2024)) {
          return_PhaseIV[[length(return_PhaseIV) + 1]] <- temp_data
        }

      }
      bases <- c(bases, base)
    }
  }

  for (k in return) {
    temp_data <- data.frame(Base = base, Return = k)

    ## Source of announcement
    if (institute == "European Commission") {
      return_Commission[[length(return_Commission) + 1]] <- temp_data
    }

    if (institute == "Council of the European Union") {
      return_Council[[length(return_Council) + 1]] <- temp_data
    }
```

```
        if (institute == "European Parliament") {
          return_Parliament[[length(return_Parliament) + 1]] <- temp_data
        }

        if (institute == "Directorate-General for Climate Action") {
          return_Directorate[[length(return_Directorate) + 1]] <- temp_data
        }

      }
    }
  }
}
remove(bases, base, i, institute, form, first, last, single, year, return, idx, j, k, temp_data)

## Creating the appropriate result data frames
return_ALL <- do.call(rbind, return_ALL)

return_Commission <- do.call(rbind, return_Commission)
return_Council <- do.call(rbind, return_Council)
return_Parliament <- do.call(rbind, return_Parliament)
return_Directorate <- do.call(rbind, return_Directorate)

return_Directive <- do.call(rbind, return_Directive)
return_Communication <- do.call(rbind, return_Communication)
return_Regulation <- do.call(rbind, return_Regulation)

return_First <- do.call(rbind, return_First)
return_Last <- do.call(rbind, return_Last)
return_Single <- do.call(rbind, return_Single)
return_Development <- do.call(rbind, return_Development)

return_PhaseIII <- do.call(rbind, return_PhaseIII)
return_PhaseIV <- do.call(rbind, return_PhaseIV)
```

Verifying the validity of the return result

```
ALL <- c(return_ALL, return_Commission, return_Council, return_Parliament, return_Directorate, return_Di

## Checking whether each (unique) base has exactly 11 returns (one for each event day)
for (i in seq_along(ALL)) {
  data <- ALL[[i]]  # Accessing the vector at index 'i' in ALL
  length_data <- length(data)

  if (!(length_data %% 11 == 0)) {
    print(paste("Problem with: ", i))
  }
}
remove(ALL, data, i, length_data)
```

```r
## Truncated model
remove_extremes <- function(return_vector, truncated_percentage) {
  runs <- truncated_percentage * length(return_vector)
  run <- 0

  while (run < runs) {
    keep_index <- return_vector != max(return_vector) & return_vector != min(return_vector)
    return_vector <- return_vector[keep_index]
    run <- run + 1
  }

  return(return_vector)
}
```

```r
estimationperiod_return <- list()
estimationperiod_average_return <- list()

bases <- numeric()
verify <- list()

truncated_percentage <- 0.15

for (i in 1:nrow(event_study)) {
  if (!is.na(event_study$ControlPeriod[i])) {
    if (event_study$EventDay[i]) {
      bases <- c(bases, event_study$Index[i])
    }

    if (is.na(event_study$ControlPeriod[i + 1])) {
      index <- event_study$Index[i + 1]
      return <- numeric()

      found <- FALSE
      while (found == FALSE) {
        idx <- max(index - 19, 0)
        na_count <- 0

        for (j in seq(idx + 19, idx)) {
          if (is.na(event_study$ControlPeriod[event_study$Index == j][1])) {
            na_count <- na_count + 1
          } else {
            break
          }
        }
        if (na_count == 20) {
          na_count <- 0
          for (k in seq(idx + 19, idx)) {
            if (!(k %in% verify)) {
              na_count <- na_count + 1
            }
          }
          if (na_count == 20) {
            for (m in seq(idx + 19, idx)) {
```

```
              event_study$EstimationPeriod[event_study$Index == m] <- 1

              verify <- c(verify, m)
              return <- c(return, event_study$LogReturn[event_study$Index == m][1])
            }

            return <- remove_extremes(return, truncated_percentage)

            for (n in unique(bases)) {
              temp_data <- data.frame(Base = n, AverageReturn = mean(return))
              estimationperiod_average_return[[length(estimationperiod_average_return) + 1]] <- temp_da

              for (o in return) {
                temp_data <- data.frame(Base = n, Return = o)
                estimationperiod_return[[length(estimationperiod_return) + 1]] <- temp_data
              }
            }
            bases <- numeric()
            found <- TRUE
          }
        } else {
          break
        }
      }
    }
  }
}
remove(verify, temp_data, bases, found, i, idx, index, j, k, m, n, o, na_count, return)

estimationperiod_return <- do.call(rbind, estimationperiod_return)
estimationperiod_average_return <- do.call(rbind, estimationperiod_average_return)
```

```
temp_data <- event_study %>%
  filter(Include == 1) %>%
  select(Index) %>%
  distinct() %>%
  rename(Base = Index) %>%
  left_join(estimationperiod_return, by = "Base")
estimationperiod_return <- temp_data

temp_data <- event_study %>%
  filter(Include == 1) %>%
  select(Index) %>%
  distinct() %>%
  rename(Base = Index) %>%
  left_join(estimationperiod_average_return, by = "Base")
estimationperiod_average_return <- temp_data

remove(temp_data)
```

```
Obs <- length(estimationperiod_return$Return)
Kurt <- kurtosis(estimationperiod_return$Return, type = 1)
Skew <- skewness(estimationperiod_return$Return, type = 1)
```

```
chi_sq_stat <- Obs * (Skew ^ 2 / 6 + Kurt ^ 2 / 24)
JarqueBera.pvalue <- pchisq(chi_sq_stat, df = 2, lower.tail = FALSE)

remove(Obs, Kurt, Skew, chi_sq_stat)
```

Following the methodology of Deeney et al.

```
AbnormalReturns <- function(return_data) {
  ## Calculate Abnormal Return
  return_data <- return_data %>%
    left_join(estimationperiod_average_return, by = "Base") %>%
    mutate(AbnormalReturn = Return - AverageReturn)
  return(return_data)
}

## 1. Calculate the Abnormal Return per event day (Day -5 to Day 5)
AbnormalReturns_per_eventday <- function(return_data) {
  AbnormalReturn_per_eventday <- list()
  for (i in 1:11) {
    returns <- numeric()
    for (y in unique(return_data$Base)) {
      df <- return_data[return_data$Base == y, ]
      returns <- c(returns, df$AbnormalReturn[i])
    }
    temp_data <- data.frame(Day = i - 6, AbnormalReturn = returns)
    AbnormalReturn_per_eventday[[length(AbnormalReturn_per_eventday) + 1]] <- temp_data
  }
  AbnormalReturn_per_eventday <- do.call(rbind, AbnormalReturn_per_eventday)
  return(AbnormalReturn_per_eventday)
}

Average_AbnormalReturns_per_eventday <- function(return_data) {
  Average_AbnormalReturn_per_eventday <- list()
  for (y in unique(return_data$Day)) {
    df <- return_data[return_data$Day == y, ]
    Average_AR <- mean(df$AbnormalReturn)

    temp_data <- data.frame(Day = y, AbnormalReturn = Average_AR)
    Average_AbnormalReturn_per_eventday[[length(Average_AbnormalReturn_per_eventday) + 1]] <- temp_data
  }
  Average_AbnormalReturn_per_eventday <- do.call(rbind, Average_AbnormalReturn_per_eventday)
  return(Average_AbnormalReturn_per_eventday)
}

## 2. Calculate the CAR for each day (aka 'the rest of the fking owl')
CumulativeAbnormalReturns_per_eventday <- function(return_data, estimation_data, sample_count) {
  variance <- var(estimation_data$Return)

  CumulativeAbnormalReturn_per_eventday <- list()
  for (y in unique(return_data$Day)) {
    df <- return_data[return_data$Day <= y, ]
```

```
    CAR <- sum(df$AbnormalReturn)

    L <- y + 6
    bottom <- sqrt(variance * L)
    ttest <- CAR / bottom

    ptest <- 2 * (1 - pnorm(abs(ttest)))

    temp_data <- data.frame(
      Day = y,
      CumulativeAbnormalReturn = CAR,
      t.test = ttest,
      p.test = ptest
                            )

    CumulativeAbnormalReturn_per_eventday[[length(CumulativeAbnormalReturn_per_eventday) + 1]] <- temp_
  }
  CumulativeAbnormalReturn_per_eventday <- do.call(rbind, CumulativeAbnormalReturn_per_eventday)
  return(CumulativeAbnormalReturn_per_eventday)
}

TableFilter <- function(return_data) {
  return_data <- return_data %>%
    select(Day, p.test)
}

main <- function(return_data, estimation_data) {
  return_data <- AbnormalReturns(return_data)
  return_data <- AbnormalReturns_per_eventday(return_data)
  sample_count <- c("n", nrow(return_data) / 11)
  return_data <- Average_AbnormalReturns_per_eventday(return_data)
  return_data <- CumulativeAbnormalReturns_per_eventday(return_data, estimation_data, sample_count)
  return_data <- TableFilter(return_data)
  return_data <- rbind(return_data, sample_count)
  return(return_data)
}
```

```
estimationperiod_return <- AbnormalReturns(estimationperiod_return) %>%
  select(Base, AbnormalReturn) %>%
  rename(Return = AbnormalReturn)
```

```
return_ALL <- main(return_ALL, estimationperiod_return)

return_Commission <- main(return_Commission, estimationperiod_return)
return_Council <- main(return_Council, estimationperiod_return)
return_Parliament <- main(return_Parliament, estimationperiod_return)
return_Directorate <- main(return_Directorate, estimationperiod_return)

return_Directive <- main(return_Directive, estimationperiod_return)
return_Communication <- main(return_Communication, estimationperiod_return)
return_Regulation <- main(return_Regulation, estimationperiod_return)

return_First <- main(return_First, estimationperiod_return)
```

```
return_Last <- main(return_Last, estimationperiod_return)
return_Single <- main(return_Single, estimationperiod_return)
return_Development <- main(return_Development, estimationperiod_return)

return_PhaseIII <- main(return_PhaseIII, estimationperiod_return)
return_PhaseIV <- main(return_PhaseIV, estimationperiod_return)

#write.csv(return_ALL, file = "ttest_test.csv", row.names = FALSE)
#write.csv(estimationperiod_return, file = "estimationperiod_return.csv", row.names = FALSE)

remove(estimationperiod_average_return, estimationperiod_return)
remove(AbnormalReturns, AbnormalReturns_per_eventday, Average_AbnormalReturns_per_eventday, CumulativeAb
```

Creating one table with all results

```
event_study_results <- return_ALL %>%
  left_join(return_Single, by = "Day") %>%
  left_join(return_First, by = "Day") %>%
  left_join(return_Development, by = "Day") %>%
  left_join(return_Last, by = "Day") %>%
  ## Phases
  left_join(return_PhaseIII, by = "Day") %>%
  left_join(return_PhaseIV, by = "Day") %>%
  ## Institution
  left_join(return_Commission, by = "Day") %>%
  left_join(return_Council, by = "Day") %>%
  left_join(return_Parliament, by = "Day") %>%
  left_join(return_Directorate, by = "Day") %>%
  ## Type
  left_join(return_Communication, by = "Day") %>%
  left_join(return_Directive, by = "Day") %>%
  left_join(return_Regulation, by = "Day")
colnames(event_study_results) <- c("Day", "All", "Single", "First", "Development", "Last",
                                   "Phase III", "Phase IV",
                                   "Commission", "Council", "Parliament", "Directorate",
                                   "Communication", "Directive", "Regulation")

write.csv(event_study_results, file = "eventstudy_results.csv", row.names = FALSE)

head(event_study_results, 12)
```

```
##    Day               All            Single             First       Development
## 1   -5  0.86941022270176  0.90346095177945 0.281799695625815 0.451775345123743
## 2   -4 0.731538532343467 0.697585910955792 0.494700138135872 0.674151213725395
## 3   -3 0.688505205061295 0.821557360401664 0.213367419040339 0.594491909032646
## 4   -2 0.815306498507755 0.899621796346168 0.138657028238107 0.415887759227317
## 5   -1 0.983871694667322 0.835283182295745 0.308827432770922  0.40124770009407
## 6    0 0.917907189194063  0.70772939639748 0.465806098061601 0.511328519712274
## 7    1 0.986222140200141 0.819532655623229 0.434079812353744 0.542667395205726
## 8    2 0.940778136166927  0.94564675026765 0.458930272528317 0.676014253185797
## 9    3 0.916669842428071 0.745269270362233 0.518794837051588 0.532666533058309
```

```
## 10   4 0.910639852038317 0.907869105541572 0.567206530782223 0.565120416701596
## 11   5 0.885350691201683 0.936612360660026 0.530368058572726 0.459372319528507
## 12   n                46                10                10                15
##                 Last         Phase III          Phase IV        Commission
## 1  0.596375269731546 0.853591221954761  0.67630597321424 0.670039457029497
## 2  0.301128023547281 0.666685018783385 0.844076953913882 0.597349729721654
## 3  0.286393254883293 0.736694124777414 0.981627755155247 0.511056424446768
## 4  0.392708295781798 0.907460009055705 0.813617220214019 0.505045568339284
## 5  0.621994932332021 0.899169147163763 0.720734099748631 0.644999287431856
## 6  0.886778451255729 0.798388459290991 0.639649761040333 0.771660998408803
## 7  0.690141080647402 0.897954061449626 0.759941987106236 0.675552049026505
## 8  0.787740847012314 0.985395629253122 0.756342484650514 0.706390342960484
## 9  0.902830779263108 0.823904327080857 0.696354863516136 0.857800639263636
## 10 0.920467349425456 0.788844238572059 0.667561646237237 0.769216435797702
## 11 0.926577278084902 0.783486614369681 0.602340011691839 0.722316871556365
## 12                11                27                17                21
##              Council        Parliament       Directorate      Communication
## 1  0.375932375886531  0.89706646853425 0.842132079726883 0.0762106746038134
## 2  0.726725045235791  0.91466986478022 0.735815469249151  0.196076963956236
## 3   0.89614925665336 0.941390988948567 0.960956805771571  0.145603616728611
## 4  0.902649408698723 0.777029428541932 0.575043617845412  0.123924099166914
## 5  0.784316563793786 0.678425574587397 0.469295406804863  0.204084249647349
## 6  0.815153336224994 0.693888395590214  0.26457328703622  0.406454125754009
## 7  0.909484550593061 0.785132811804207 0.363611104692934   0.19137868690295
## 8  0.983698586208291 0.739747258773447  0.27002236858759  0.614824515168176
## 9  0.777331817083946 0.780421399658868 0.248663230275713  0.254887071943484
## 10 0.721028120970037 0.701533723152644 0.319193522950867 0.0528179692245832
## 11 0.670678412293326 0.573118122135039 0.374508804919948 0.0275948157862811
## 12                19                 7                 6                 1
##            Directive        Regulation
## 1  0.380604758710384 0.790763952380567
## 2  0.269877326171779 0.987451421582988
## 3  0.287321603852355 0.877853536462586
## 4  0.676800201399151 0.809680533245152
## 5  0.668162500533214 0.685030946910326
## 6  0.414384468445081  0.66949527456867
## 7  0.762684015661288 0.741778906173483
## 8  0.611269316816604 0.878691378520239
## 9  0.702001887003638 0.749389200192397
## 10 0.647460561424772 0.764090108991569
## 11 0.671171168964774 0.800652288379866
## 12                5                22
```

```
remove(return_ALL, return_Commission, return_Communication, return_Council, return_Directive, return_Di

remove(JarqueBera.pvalue)
```

News proxy

```
## Importing the relevant articles
articles <- read_csv("EU ETS Aviation relevant articles.csv", show_col_types = FALSE)
```

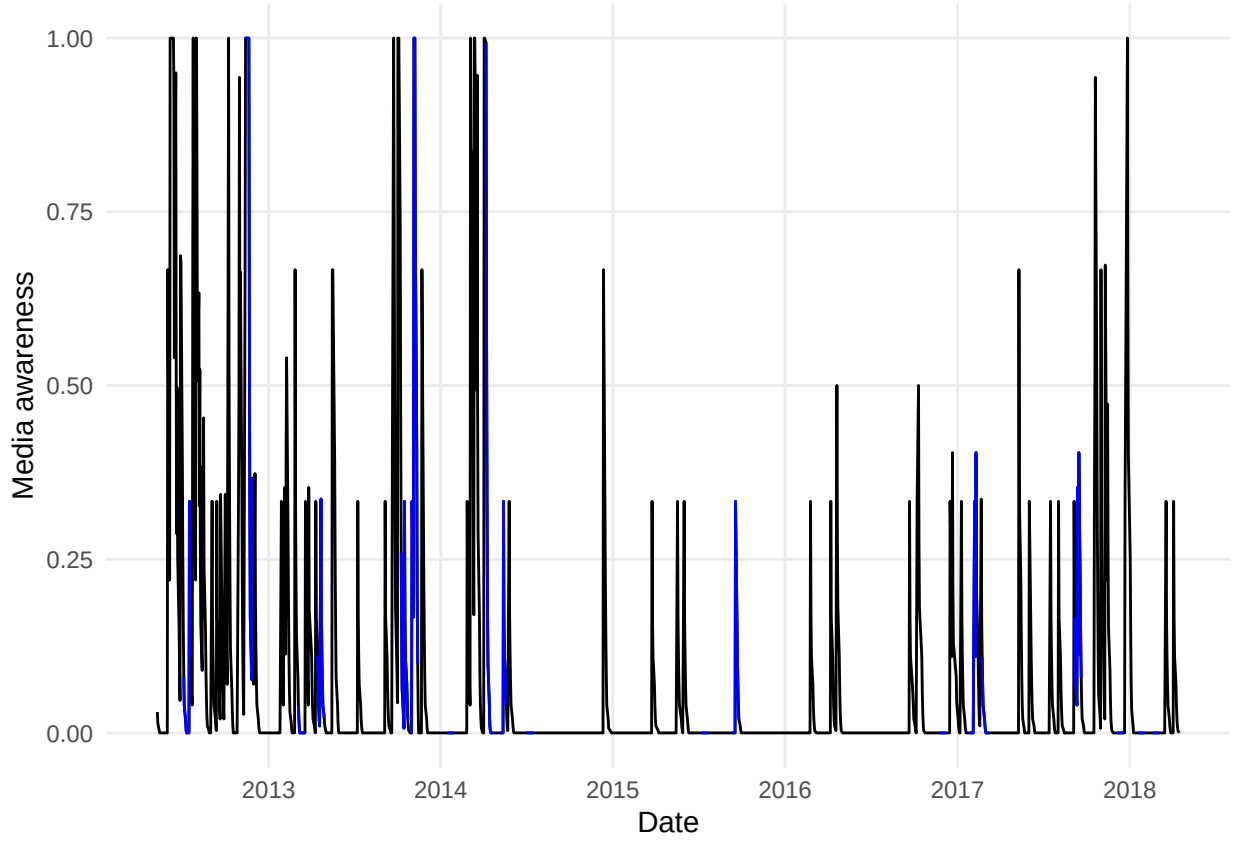
```r
## Formatting the date in the same structure as 'event_study'
articles <- articles[articles$Date != "Unknown", ] %>%
  mutate(Date = mdy(Date),
         Date = format(Date, "%Y-%m-%d"),
         Date = as.Date(Date))

## New df with the number of articles on each date
article_freq <- articles %>%
  count(Date)

article_mean <- round(mean(article_freq$n), digits = 1)
```

```r
## Modifying 'event_study' to only take the EventPeriod
news_proxy <- event_study %>%
  select(Index, Date, EstimationPeriod, EventPeriod) %>%
  distinct(Index, .keep_all = TRUE) %>%
  arrange(Index) %>%
  ## Merging it with the article frequency
  left_join(article_freq, by = "Date") %>%
  mutate(n = case_when(is.na(n) ~ 0, TRUE ~ n)) %>%
  mutate(IntAwareness = if_else(row_number() <= 7, n, NA)) %>%
  mutate(EventPeriodAwareness = NA)

## Assuming that the AWARENESS caused by an article dissipates over 7 days in these proportions
for (i in 8:nrow(news_proxy)) {
  news_proxy$IntAwareness[i] <- news_proxy$n[i] +
                                0.5 * news_proxy$n[i-1] +
                                0.33 * news_proxy$n[i-2] +
                                0.21 * news_proxy$n[i-3] +
                                0.12 * news_proxy$n[i-4] +
                                0.06 * news_proxy$n[i-5] +
                                0.03 * news_proxy$n[i-6] +
                                0.01 * news_proxy$n[i-7]
}

news_proxy <- news_proxy %>%
  mutate(Awareness = case_when(IntAwareness > (2 * article_mean) ~ 1,
                               TRUE ~ IntAwareness / (2 * article_mean)))

## Removing the first 7 rows as they don't follow the the above logic
news_proxy <- news_proxy[-(1:7), ]

remove(article_freq, article_mean, i)
```

```r
## A little hack to properly graph the event period results
news_proxy <- news_proxy %>%
  mutate(EventPeriodAwareness = if_else(!is.na(EventPeriod) & EventPeriod == 1, Awareness, -1))

temp <- news_proxy
news_proxy <- news_proxy[1:(nrow(news_proxy)/2), ]

## Graph
p <- ggplot(news_proxy, aes(x = Date, y = Awareness)) +
```

```
  geom_line(color = "black") +
  labs(x = "Date",
       y = "Media awareness") +
  theme_minimal() +
  theme(panel.grid.minor = element_blank()) +
  scale_x_date(date_breaks = "1 year", date_labels = "%Y")

p + geom_line(data = news_proxy, aes(x = Date, y = EventPeriodAwareness), color = "blue") +
  scale_y_continuous(limits = c(0, 1))
```

```
## Warning: Removed 66 rows containing missing values or values outside the scale range
## (`geom_line()`).
```

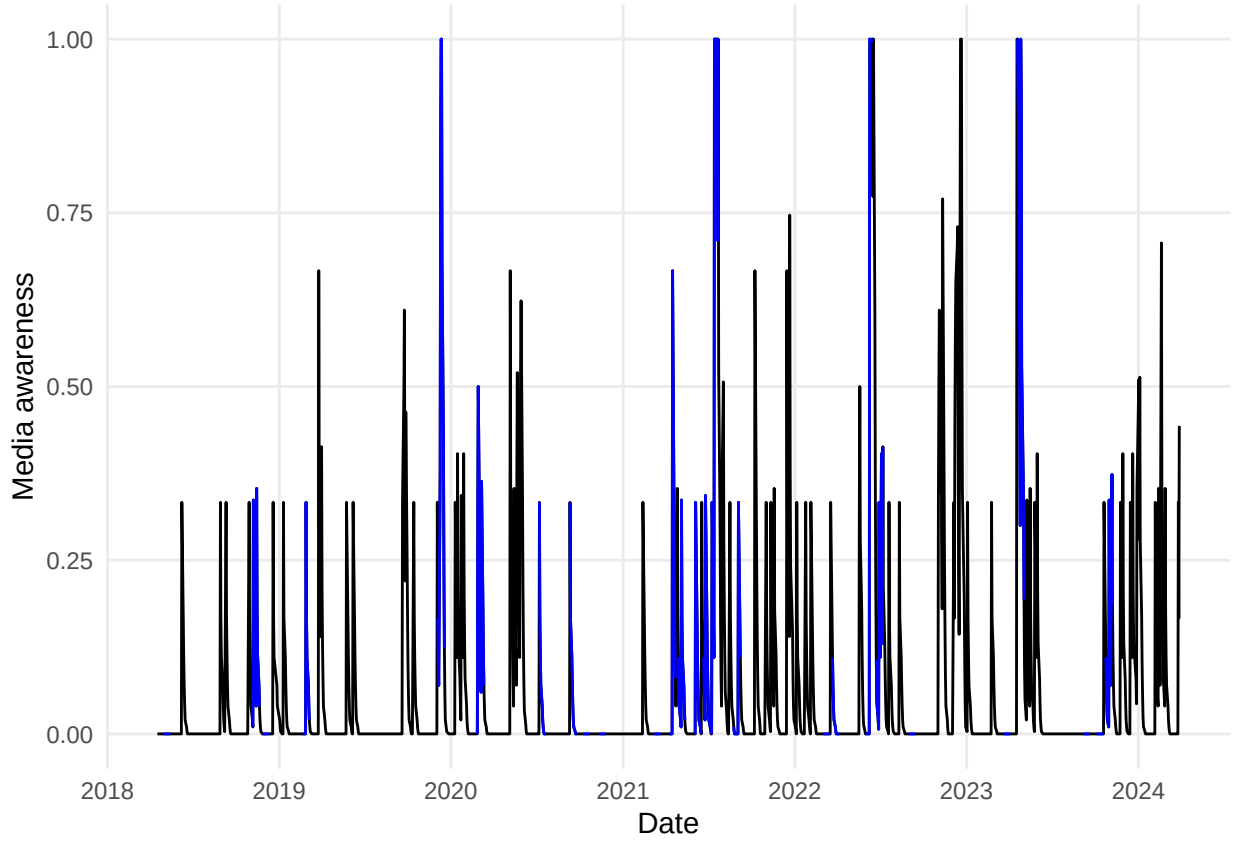



```
#ggsave("media_awareness_plot1.png", plot = last_plot(), width = 8, height = 3, dpi = 300)
```

```
news_proxy <- temp
news_proxy <- news_proxy[-(1:(nrow(news_proxy)/2)), ]
## Graph
p <- ggplot(news_proxy, aes(x = Date, y = Awareness)) +
  geom_line(color = "black") +
  labs(x = "Date",
       y = "Media awareness") +
  theme_minimal() +
  theme(panel.grid.minor = element_blank()) +
```

```
  scale_x_date(date_breaks = "1 year", date_labels = "%Y")

p + geom_line(data = news_proxy, aes(x = Date, y = EventPeriodAwareness), color = "blue") +
  scale_y_continuous(limits = c(0, 1))
```

```
## Warning: Removed 108 rows containing missing values or values outside the scale range
## (`geom_line()`).
```




```
#ggsave("media_awareness_plot2.png", plot = last_plot(), width = 8, height = 3, dpi = 300)

news_proxy <- temp
remove(p, temp)
```

Percentage media awareness

```
## Denominator
testing_main <- news_proxy %>%
  filter(EventPeriod == 1)
testing_main2 <- news_proxy %>%
  filter(EstimationPeriod == 1)

## Numerator
```

```r
testing_EventPeriod <- news_proxy %>%
  filter(Awareness > 0.00000000000000000000000001) %>%
  filter(EventPeriod == 1)

testing_EstimationPeriod <- news_proxy %>%
  filter(Awareness > 0.00000000000000000000000001) %>%
  filter(EstimationPeriod == 1)

## Statistics
print(mean(testing_EventPeriod$Awareness))
```

```
## [1] 0.2571646
```

```r
print(median(testing_EventPeriod$Awareness))
```

```
## [1] 0.11
```

```r
print(mean(testing_EstimationPeriod$Awareness))
```

```
## [1] 0.2393301
```

```r
print(median(testing_EstimationPeriod$Awareness))
```

```
## [1] 0.11
```

```r
209/540
```

```
## [1] 0.387037
```

```r
241/506
```

```
## [1] 0.4762846
```

```r
269/540
```

```
## [1] 0.4981481
```

```r
298/506
```

```
## [1] 0.5889328
```

```r
remove(testing_main, testing_main2, testing_EstimationPeriod, testing_EventPeriod)
```

GARCH

```r
GARCH <- event_study %>%
  select(Index, Date, Include, EstimationPeriod, EventPeriod, EventDay, LogReturn) %>%
  distinct(Index, .keep_all = TRUE) %>%
  filter(EstimationPeriod == 1 | EventPeriod == 1)

#write.csv(GARCH, "solver.csv", row.names = FALSE)

CALC_STATS <- data.frame(
  Constant = 0,
  `Unconditional variance` = 0,
  ARCH = 0,
  GARCH = 0,
  `Long-run volatility` = 0
)
# Estimated using (Excel's) solver
CALC_STATS <- data.frame(
  Constant = 0.00109799708601911,
  `Unconditional variance` = 0.0000391568178133135,
  ARCH = 0.17027464378627,
  GARCH = 0.819129324526429,
  `Long-run volatility` = 0.0607899898714179
)
```

```r
PARAMETER_DATA <- GARCH %>%
  arrange(Index) %>%
  mutate(Residual = LogReturn - CALC_STATS$Constant) %>%
  mutate(SqResidual = Residual * Residual) %>%
  mutate(LagSqResidual = lag(SqResidual)) %>%
  mutate(CondVariance = ifelse(Index == 27, CALC_STATS$Unconditional.variance / (1 - CALC_STATS$ARCH -

temp_data <- PARAMETER_DATA$CondVariance[[1]]
for (i in 2:nrow(PARAMETER_DATA)) {
  temp_data <- CALC_STATS$Unconditional.variance + CALC_STATS$ARCH * PARAMETER_DATA$LagSqResidual[i] +
  PARAMETER_DATA$CondVariance[i] <- temp_data
}
remove(temp_data, i)

PARAMETER_DATA <- PARAMETER_DATA %>%
  mutate(LogLikelihood = log(1/sqrt(2*pi*CondVariance)) + (-SqResidual/(2*CondVariance))) %>%
  mutate(RealisedVolatility = sqrt(SqResidual)) %>%
  mutate(GARCH = sqrt(CondVariance)) %>%
  arrange(desc(Index)) %>%
  select(Index, Date, Include, EstimationPeriod, EventPeriod, EventDay, LogReturn, RealisedVolatility,
remove(CALC_STATS, GARCH)
```

```r
## Estimation period
estimationperiod_average_GARCH <- list()
estimationperiod_GARCH <- list()

bases <- numeric()
return <- numeric()

for (i in 1:nrow(PARAMETER_DATA)) {
```

```
  if (PARAMETER_DATA$EventDay[i] == TRUE) {
    if (PARAMETER_DATA$Include[i] == 1) {
      bases <- c(bases, PARAMETER_DATA$Index[i])
    }
  }

  if (is.na(PARAMETER_DATA$EstimationPeriod[i])) {
    if (is.na(PARAMETER_DATA$EventPeriod[i+1])) {
      index <- PARAMETER_DATA$Index[i+1]
      idx <- max(index - 19, 0)

      for (j in seq(idx, idx + 19)) {
        return <- c(return, PARAMETER_DATA$GARCH[PARAMETER_DATA$Index == j])
      }

      return <- remove_extremes(return, truncated_percentage)

      for (k in unique(bases)) {
        temp_data <- data.frame(Base = k, AverageReturn = mean(return))
        estimationperiod_average_GARCH[[length(estimationperiod_average_GARCH) + 1]] <- temp_data

        for (n in return) {
          temp_data <- data.frame(Base = k, Return = n)
          estimationperiod_GARCH[[length(estimationperiod_GARCH) + 1]] <- temp_data
        }
      }
      bases <- numeric()
    }
  }
  return <- numeric()
}
remove(bases, return, i, index, idx, j, k, temp_data, n)

estimationperiod_average_GARCH <- do.call(rbind, estimationperiod_average_GARCH)
estimationperiod_GARCH <- do.call(rbind, estimationperiod_GARCH)

remove(truncated_percentage, remove_extremes)
```

```
GARCH_Commission_bases <- numeric()
GARCH_Communication_bases <- numeric()
GARCH_Council_bases <- numeric()
GARCH_Development_bases <- numeric()
GARCH_Directive_bases <- numeric()
GARCH_Directorate_bases <- numeric()
GARCH_First_bases <- numeric()
GARCH_Last_bases <- numeric()
GARCH_Parliament_bases <- numeric()
GARCH_PhaseIII_bases <- numeric()
GARCH_PhaseIV_bases <- numeric()
GARCH_Regulation_bases <- numeric()
GARCH_Single_bases <- numeric()

mall <- event_study %>%
```

```
  filter(Include == 1)

index <- numeric()
for (i in 1:nrow(mall)) {
  if (!(mall$Index[i] %in% index)) {
    index <- c(index, mall$Index[i])
    ## Type
    if (mall$Form[i] == "Directive") {
      GARCH_Directive_bases <- c(GARCH_Directive_bases, mall$Index[i])
    }

    if (mall$Form[i] == "Regulation") {
      GARCH_Regulation_bases <- c(GARCH_Regulation_bases, mall$Index[i])
    }

    if (mall$Form[i] == "Communication") {
      GARCH_Communication_bases <- c(GARCH_Communication_bases, mall$Index[i])
    }

    ## Phase
    if (mall$Year[i] < 2021 & mall$Year[i] > 2012) {
      GARCH_PhaseIII_bases <- c(GARCH_PhaseIII_bases, mall$Index[i])
    }

    if (mall$Year[i] > 2020) {
      GARCH_PhaseIV_bases <- c(GARCH_PhaseIV_bases, mall$Index[i])
    }

    ## Multiple announcements
    if (mall$First[i] == TRUE) {
      GARCH_First_bases <- c(GARCH_First_bases, mall$Index[i])
    }

    if (mall$Last[i] == TRUE) {
      GARCH_Last_bases <- c(GARCH_Last_bases, mall$Index[i])
    }

    if (mall$First[i] == FALSE & mall$Last[i] == FALSE & mall$Single[i] == FALSE) {
      GARCH_Development_bases <- c(GARCH_Development_bases, mall$Index[i])
    }

    ## Etc.
    if (mall$Single[i] == TRUE) {
      GARCH_Single_bases <- c(GARCH_Single_bases, mall$Index[i])
    }
  }
  ## Institutions
  if (mall$Institution[i] == "European Commission") {
    GARCH_Commission_bases <- c(GARCH_Commission_bases, mall$Index[i])
  }

  if (mall$Institution[i] == "Council of the European Union") {
    GARCH_Council_bases <- c(GARCH_Council_bases, mall$Index[i])
```

```
  }

  if (mall$Institution[i] == "European Parliament") {
    GARCH_Parliament_bases <- c(GARCH_Parliament_bases, mall$Index[i])
  }

  if (mall$Institution[i] == "Directorate-General for Climate Action") {
    GARCH_Directorate_bases <- c(GARCH_Directorate_bases, mall$Index[i])
  }
}
remove(mall, index, i)
```

```
## Event period
GARCH_ALL <- list()

GARCH_Commission <- list()
GARCH_Council <- list()
GARCH_Parliament <- list()
GARCH_Directorate <- list()

GARCH_Directive <- list()
GARCH_Regulation <- list()
GARCH_Communication <- list()

GARCH_PhaseIII <- list()
GARCH_PhaseIV <- list()

GARCH_Single <- list()

GARCH_First <- list()
GARCH_Last <- list()
GARCH_Development <- list()

bases <- numeric()
return <- numeric()

for (i in 1:nrow(PARAMETER_DATA)) {
  if (PARAMETER_DATA$EventDay[i] == TRUE) {
    if (PARAMETER_DATA$Include[i] == 1) {
      bases <- c(bases, PARAMETER_DATA$Index[i])
    }
  }
}

for (i in bases) {
  base <- i

  for (j in seq(base - 5, base + 5)) {
    return <- c(return, PARAMETER_DATA$GARCH[PARAMETER_DATA$Index == j])
  }

  for (k in return) {
    temp_data <- data.frame(Base = base, Return = k)
```

```
  GARCH_ALL[[length(GARCH_ALL) + 1]] <- temp_data

  ## Institution
  if (base %in% unique(GARCH_Commission_bases)) {
    GARCH_Commission[[length(GARCH_Commission) + 1]] <- temp_data
  }

  if (base %in% unique(GARCH_Council_bases)) {
    GARCH_Council[[length(GARCH_Council) + 1]] <- temp_data
  }

  if (base %in% unique(GARCH_Parliament_bases)) {
    GARCH_Parliament[[length(GARCH_Parliament) + 1]] <- temp_data
  }

  if (base %in% unique(GARCH_Directorate_bases)) {
    GARCH_Directorate[[length(GARCH_Directorate) + 1]] <- temp_data
  }

  ## Type
  if (base %in% unique(GARCH_Directive_bases)) {
    GARCH_Directive[[length(GARCH_Directive) + 1]] <- temp_data
  }

  if (base %in% unique(GARCH_Regulation_bases)) {
    GARCH_Regulation[[length(GARCH_Regulation) + 1]] <- temp_data
  }

  if (base %in% unique(GARCH_Communication_bases)) {
    GARCH_Communication[[length(GARCH_Communication) + 1]] <- temp_data
  }

  ## Phase
  if (base %in% unique(GARCH_PhaseIII_bases)) {
    GARCH_PhaseIII[[length(GARCH_PhaseIII) + 1]] <- temp_data
  }

  if (base %in% unique(GARCH_PhaseIV_bases)) {
    GARCH_PhaseIV[[length(GARCH_PhaseIV) + 1]] <- temp_data
  }

  ## Single
  if (base %in% unique(GARCH_Single_bases)) {
    GARCH_Single[[length(GARCH_Single) + 1]] <- temp_data
  }

  ## Multiple announcements
  if (base %in% unique(GARCH_First_bases)) {
    GARCH_First[[length(GARCH_First) + 1]] <- temp_data
  }

  if (base %in% unique(GARCH_Last_bases)) {
    GARCH_Last[[length(GARCH_Last) + 1]] <- temp_data
```

```
    }

    if (base %in% unique(GARCH_Development_bases)) {
      GARCH_Development[[length(GARCH_Development) + 1]] <- temp_data
    }
  }

  return <- numeric()
}
remove(base, bases, i, j, k, temp_data, return)

remove(GARCH_Commission_bases, GARCH_Communication_bases, GARCH_Council_bases, GARCH_Development_bases,

GARCH_ALL <- do.call(rbind, GARCH_ALL)

GARCH_Commission <- do.call(rbind, GARCH_Commission)
GARCH_Council <- do.call(rbind, GARCH_Council)
GARCH_Parliament <- do.call(rbind, GARCH_Parliament)
GARCH_Directorate <- do.call(rbind, GARCH_Directorate)

GARCH_Directive <- do.call(rbind, GARCH_Directive)
GARCH_Regulation <- do.call(rbind, GARCH_Regulation)
GARCH_Communication <- do.call(rbind, GARCH_Communication)

GARCH_PhaseIII <- do.call(rbind, GARCH_PhaseIII)
GARCH_PhaseIV <- do.call(rbind, GARCH_PhaseIV)

GARCH_Single <- do.call(rbind, GARCH_Single)

GARCH_First <- do.call(rbind, GARCH_First)
GARCH_Last <- do.call(rbind, GARCH_Last)
GARCH_Development <- do.call(rbind, GARCH_Development)
```

```
temp_data <- PARAMETER_DATA %>%
  filter(Include == 1) %>%
  select(Index) %>%
  distinct() %>%
  rename(Base = Index) %>%
  left_join(estimationperiod_GARCH, by = "Base")
estimationperiod_GARCH <- temp_data

temp_data <- PARAMETER_DATA %>%
  filter(Include == 1) %>%
  select(Index) %>%
  distinct() %>%
  rename(Base = Index) %>%
  left_join(estimationperiod_average_GARCH, by = "Base")
estimationperiod_average_GARCH <- temp_data

remove(temp_data)
```

```
AbnormalReturns <- function(return_data) {
  ## Calculate Abnormal Return
```

```r
  return_data <- return_data %>%
    left_join(estimationperiod_average_GARCH, by = "Base") %>%
    mutate(AbnormalReturn = Return - AverageReturn)
  return(return_data)
}

## 1. Calculate the Abnormal Return per event day (Day -5 to Day 5)
AbnormalReturns_per_eventday <- function(return_data) {
  AbnormalReturn_per_eventday <- list()
  for (i in 1:11) {
    returns <- numeric()
    for (y in unique(return_data$Base)) {
      df <- return_data[return_data$Base == y, ]
      returns <- c(returns, df$AbnormalReturn[i])
    }
    temp_data <- data.frame(Day = i - 6, AbnormalReturn = returns)
    AbnormalReturn_per_eventday[[length(AbnormalReturn_per_eventday) + 1]] <- temp_data
  }
  AbnormalReturn_per_eventday <- do.call(rbind, AbnormalReturn_per_eventday)
  return(AbnormalReturn_per_eventday)
}

Average_AbnormalReturns_per_eventday <- function(return_data) {
  Average_AbnormalReturn_per_eventday <- list()
  for (y in unique(return_data$Day)) {
    df <- return_data[return_data$Day == y, ]
    Average_AR <- mean(df$AbnormalReturn)

    temp_data <- data.frame(Day = y, AbnormalReturn = Average_AR)
    Average_AbnormalReturn_per_eventday[[length(Average_AbnormalReturn_per_eventday) + 1]] <- temp_data
  }
  Average_AbnormalReturn_per_eventday <- do.call(rbind, Average_AbnormalReturn_per_eventday)
  return(Average_AbnormalReturn_per_eventday)
}

## 2. Calculate the CAR for each day (aka 'the rest of the fking owl')
CumulativeAbnormalReturns_per_eventday <- function(return_data, estimation_data, sample_count) {
  variance <- var(estimation_data$Return)

  CumulativeAbnormalReturn_per_eventday <- list()
  for (y in unique(return_data$Day)) {
    df <- return_data[return_data$Day <= y, ]
    CAR <- sum(df$AbnormalReturn)

    ## Temp
    L <- y + 6
    bottom <- sqrt(variance * L)
    ttest <- CAR / bottom

    ptest <- 2 * (1 - pnorm(abs(ttest)))

    temp_data <- data.frame(
      Day = y,
```

```
      CumulativeAbnormalReturn = CAR,
      t.test = ttest,
      p.test = ptest
                              )

    CumulativeAbnormalReturn_per_eventday[[length(CumulativeAbnormalReturn_per_eventday) + 1]] <- temp_
  }
  CumulativeAbnormalReturn_per_eventday <- do.call(rbind, CumulativeAbnormalReturn_per_eventday)
  return(CumulativeAbnormalReturn_per_eventday)
}

TableFilter <- function(return_data) {
  return_data <- return_data %>%
    select(Day, p.test)
}

main <- function(return_data, estimation_data) {
  return_data <- AbnormalReturns(return_data)
  return_data <- AbnormalReturns_per_eventday(return_data)
  sample_count <- c("n", nrow(return_data) / 11)
  return_data <- Average_AbnormalReturns_per_eventday(return_data)
  return_data <- CumulativeAbnormalReturns_per_eventday(return_data, estimation_data, sample_count)
  return_data <- TableFilter(return_data)
  return_data <- rbind(return_data, sample_count)
  return(return_data)
}
```

```
GARCH_ALL <- main(GARCH_ALL, estimationperiod_GARCH)

GARCH_Commission <- main(GARCH_Commission, estimationperiod_GARCH)
GARCH_Council <- main(GARCH_Council, estimationperiod_GARCH)
GARCH_Parliament <- main(GARCH_Parliament, estimationperiod_GARCH)
GARCH_Directorate <- main(GARCH_Directorate, estimationperiod_GARCH)

GARCH_Directive <- main(GARCH_Directive, estimationperiod_GARCH)
GARCH_Regulation <- main(GARCH_Regulation, estimationperiod_GARCH)
GARCH_Communication <- main(GARCH_Communication, estimationperiod_GARCH)

GARCH_PhaseIII <- main(GARCH_PhaseIII, estimationperiod_GARCH)
GARCH_PhaseIV <- main(GARCH_PhaseIV, estimationperiod_GARCH)

GARCH_Single <- main(GARCH_Single, estimationperiod_GARCH)

GARCH_First <- main(GARCH_First, estimationperiod_GARCH)
GARCH_Last <- main(GARCH_Last, estimationperiod_GARCH)
GARCH_Development <- main(GARCH_Development, estimationperiod_GARCH)
```

```
GARCH_results <- GARCH_ALL %>%
  left_join(GARCH_Single, by = "Day") %>%
  ## Multiple announcements
  left_join(GARCH_First, by = "Day") %>%
  left_join(GARCH_Development, by = "Day") %>%
  left_join(GARCH_Last, by = "Day") %>%
```

```
  ## Phases
  left_join(GARCH_PhaseIII, by = "Day") %>%
  left_join(GARCH_PhaseIV, by = "Day") %>%
  ## Institution
  left_join(GARCH_Commission, by = "Day") %>%
  left_join(GARCH_Council, by = "Day") %>%
  left_join(GARCH_Parliament, by = "Day") %>%
  left_join(GARCH_Directorate, by = "Day") %>%
  ## Type
  left_join(GARCH_Communication, by = "Day") %>%
  left_join(GARCH_Directive, by = "Day") %>%
  left_join(GARCH_Regulation, by = "Day")
colnames(GARCH_results) <- c("Day", "All", "Single", "First", "Development", "Last",
                             "Phase III", "Phase IV",
                             "Commission", "Council", "Parliament", "Directorate",
                             "Communication", "Directive", "Regulation")

write.csv(GARCH_results, file = "GARCH_results.csv", row.names = FALSE)

remove(GARCH_ALL, GARCH_Single, GARCH_First, GARCH_Development, GARCH_Last, GARCH_PhaseIII, GARCH_Phase
```

Clean up

```
GARCH <- PARAMETER_DATA
remove(PARAMETER_DATA)

remove(articles)

remove(estimationperiod_average_GARCH, estimationperiod_GARCH)
remove(AbnormalReturns, AbnormalReturns_per_eventday, Average_AbnormalReturns_per_eventday, CumulativeA
```