ITS design analysis

Jae Yeon Kim

Contents

0. Setup 1
1. Importing files 2
2. Merging dataframes 2
3. Descriptive analysis 3
4. Interrupted time series design analysis 6

0. Setup

I tweaked the global option of the R Markdown to enlarge figures produced by ggplot2.

```
# Clean up the environment

# rm(list = ls())

# Import libraries (adapted from this link: https://stackoverflow.com/questions/4090169/elegant-way-to-

if (!require("pacman")) install.packages("pacman")
pacman::p_load(
        tidyverse, # for the tidyverse framework
        splitstackshape, # for stacking and reshaping datasets
        tsModel, # for specifying time series regression models
        lmtest, # for testing linear regression models
        Epi, # for statistical analysis in epidemology
        splines, # for spline regression
        vcd, # for visualizing categorical variable
        nlme, # for linear and nonlinear mixed effects models
        zoo, # S3 infrastructure for regular and irregular time series
        TTR, # for constructing technical trading rules
        ggpmisc, # for detecting peaks and valleys
        broom, # for tidying model objects
        modelr, # for modeling
        forecast, # for forecasting
        ggseas, # for seasonal and decomposition on the fly
        itsadug, # for interpreting time series and autocorrelated data
        mgcv, # for mixed GAM Computation Vehicle with Automatic Smoothness Estimation
        tseries, # for computational financial analysis
        mcp, # for regression with multiple change points
        lawstat, # for testing stats in biostat, public policy, and law
        sarima, # for simulation and prediction with seasonal ARIMA models
        car, # for Durbin Watson test
        tidyr, # for tidying messy daa
```

```
        AICcmodavg, # for computing predicted values and standard errors
        MuMIn, # for selecting models
        ggthemes, # for fancy ggplot themes
        ggpubr) # for arranging plots
```

1. Importing files

Unfortunately, I cannot share the original data because they are proprietary. Proquest holds the copyrights.

2. Merging dataframes

```
# Replace NAs with 0s in sample articles

sample$expanding[is.na(sample$expanding)] <- 0
sample$distancing[is.na(sample$distancing)] <- 0
sample$assimilating[is.na(sample$assimilating)] <- 0

# Collapsing three variables into one

sample$domestic <- ifelse(sample$expanding == 1 | sample$distancing == 1 | sample$assimilating == 1, 1,

# Create domestic variable for unlabeled and apply predicted values to the variable

unlabeled$domestic <- results$`0`

# Select key variables

sample_selected <- dplyr::select(sample,
              source,
              intervention,
              date,
              domestic,
              text)

labeled_selected <- dplyr::select(unlabeled,
              source,
              intervention,
              date,
              domestic,
              text)

# Row bind the two dataframes

df <- bind_rows(sample_selected, labeled_selected)

# Save the processed data

write.csv(df, "/home/jae/ITS-Text-Classification/processed_data/df.csv")

library(tidytext)
```

```r
token <-df %>%
  unnest_tokens(word, text)
```

3. Descriptive analysis

3.1. Data wrangling

If you want to repliace the analysis, you can start from here using `df.csv` saved in `processed_data` directory..

```r
# df <- read_csv("/home/jae/ITS-Text-Classification/processed_data.df.csv")

# Check date variable

paste("the class of date is", class(df$date))
```

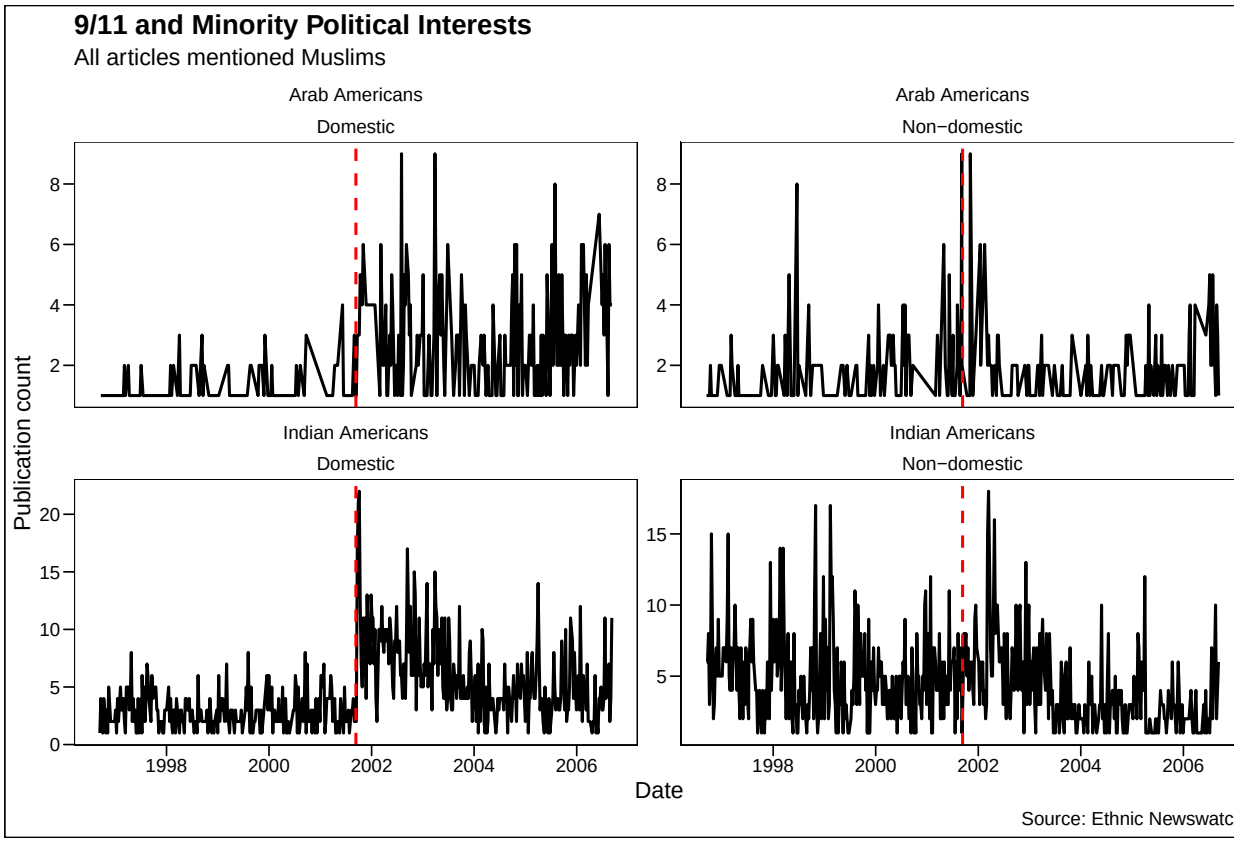
```
## [1] "the class of date is numeric"
```

```r
# Convert date into date object

df$date <- as.Date(as.character(df$date), "%Y%m%d")

# Recode values in domestic variable

df$domestic <- as.character(df$domestic)

df$domestic[df$domestic == 1] <- "Domestic"
df$domestic[df$domestic == 0] <- "Non-domestic"

# Create group variable based on newspaper names

df$group <- ifelse(str_detect(df$source, "India"), "Indian Americans", "Arab Americans")
```

3.2. Data visualization

```r

# Grouping and summarizing data

df_grouped <- df %>%
  group_by(date, domestic, group) %>% # group by
  dplyr::summarize(n = n()) %>% # summarize
  as.data.frame()

# Raw plot

df_grouped %>%
  ggplot(aes(x = date, y = n)) +
    geom_line(size = 1) + # line plot
    geom_vline(xintercept = as.Date("2001-09-11"), linetype = "dashed", size = 1, color = "red") + # ve
    scale_y_continuous(breaks= scales::pretty_breaks()) + # pretty breaks on transformed scale
    facet_wrap(group~domestic, scale = "free_y") + # facetting, y scale is free
    labs(x = "Date",
         y = "Publication count",
         col = "Issue focus",
```

```
        title = "9/11 and Minority Political Interests",
        subtitle = "All articles mentioned Muslims",
        caption = "Source: Ethnic Newswatch") + # labels
    theme_base() # theme
```



3.1. Raw data

```
# Save plot

ggsave("/home/jae/ITS-Text-Classification/output/raw_data_plot.png", height = 10)
```

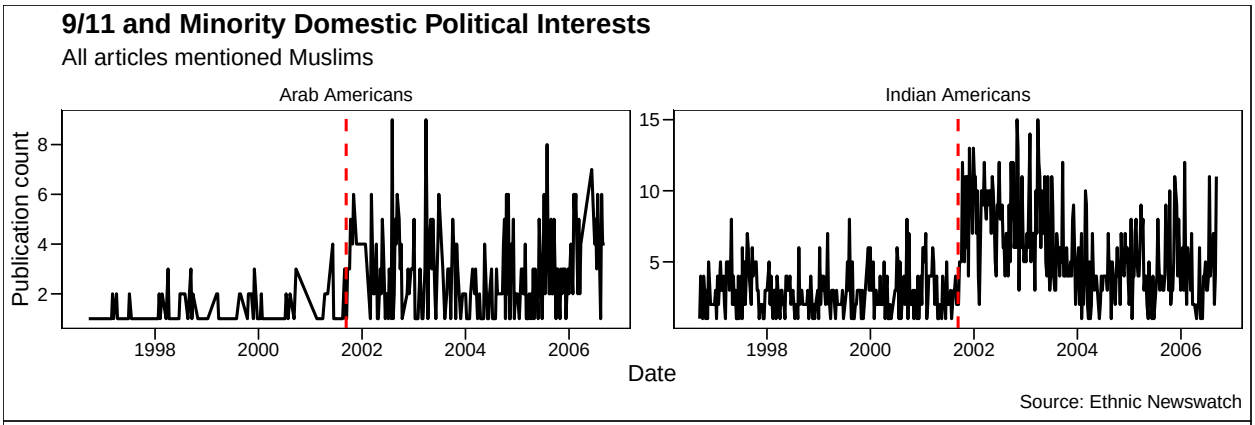
3.2. No outliers and missing values I adapted the code from this Oracle blog post to remove outliers and impute missing values. As the number of missing values in the data is 0, this process focuses on removing outliers. The plot shows that a couple of outliers disappeared in the updated data.

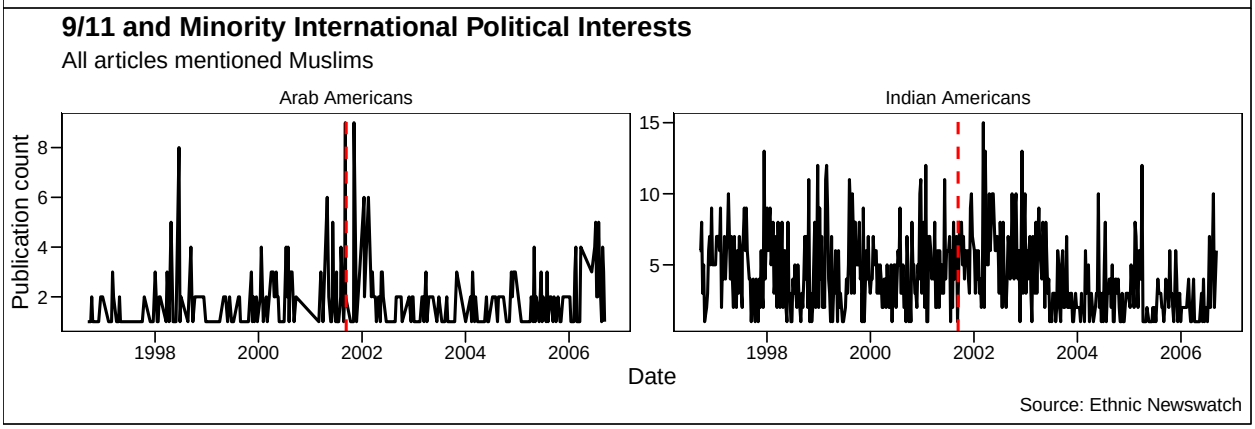
```
# checking whether there's NA
paste("The number of missing values in the data is", sum(is.na(df_grouped)))
```

```
## [1] "The number of missing values in the data is 0"
```

```
# Clean the dependent variable

count_ts <- ts(df_grouped[, c('n')])

df_grouped$count_ts <- as.integer(tsclean(count_ts))

# Get rid of n
df_grouped <- df_grouped %>%
  dplyr::select(-n)
```

```r
# Cleaned plot

cleaned_domestic_plot <- df_grouped %>%
  filter(domestic == "Domestic") %>%
  ggplot(aes(x = date, y = count_ts)) +
    geom_line(size = 1) + # line plot
    geom_vline(xintercept = as.Date("2001-09-11"), linetype = "dashed", size = 1, color = "red") + # ve
    scale_y_continuous(breaks= scales::pretty_breaks()) + # pretty breaks on transformed scale
    facet_wrap(~group, scale = "free_y") + # facetting, y scale is free
    labs(x = "Date",
         y = "Publication count",
         col = "Issue focus",
         subtitle = "All articles mentioned Muslims",
         caption = "Source: Ethnic Newswatch") + # labels
    theme_base() # theme

cleaned_nondomestic_plot <- df_grouped %>%
  filter(domestic != "Domestic") %>%
  ggplot(aes(x = date, y = count_ts)) +
    geom_line(size = 1) + # line plot
    geom_vline(xintercept = as.Date("2001-09-11"), linetype = "dashed", size = 1, color = "red") + # ve
    scale_y_continuous(breaks= scales::pretty_breaks()) + # pretty breaks on transformed scale
    facet_wrap(~group, scale = "free_y") + # facetting, y scale is free
    labs(x = "Date",
         y = "Publication count",
         col = "Issue focus",
         subtitle = "All articles mentioned Muslims",
         caption = "Source: Ethnic Newswatch") + # labels
    theme_base() # theme

# Save plot

ggarrange(cleaned_domestic_plot + ggtitle("9/11 and Minority Domestic Political Interests"),
          cleaned_nondomestic_plot + ggtitle("9/11 and Minority International Political Interests"),
          nrow = 2, ncol =1)
```





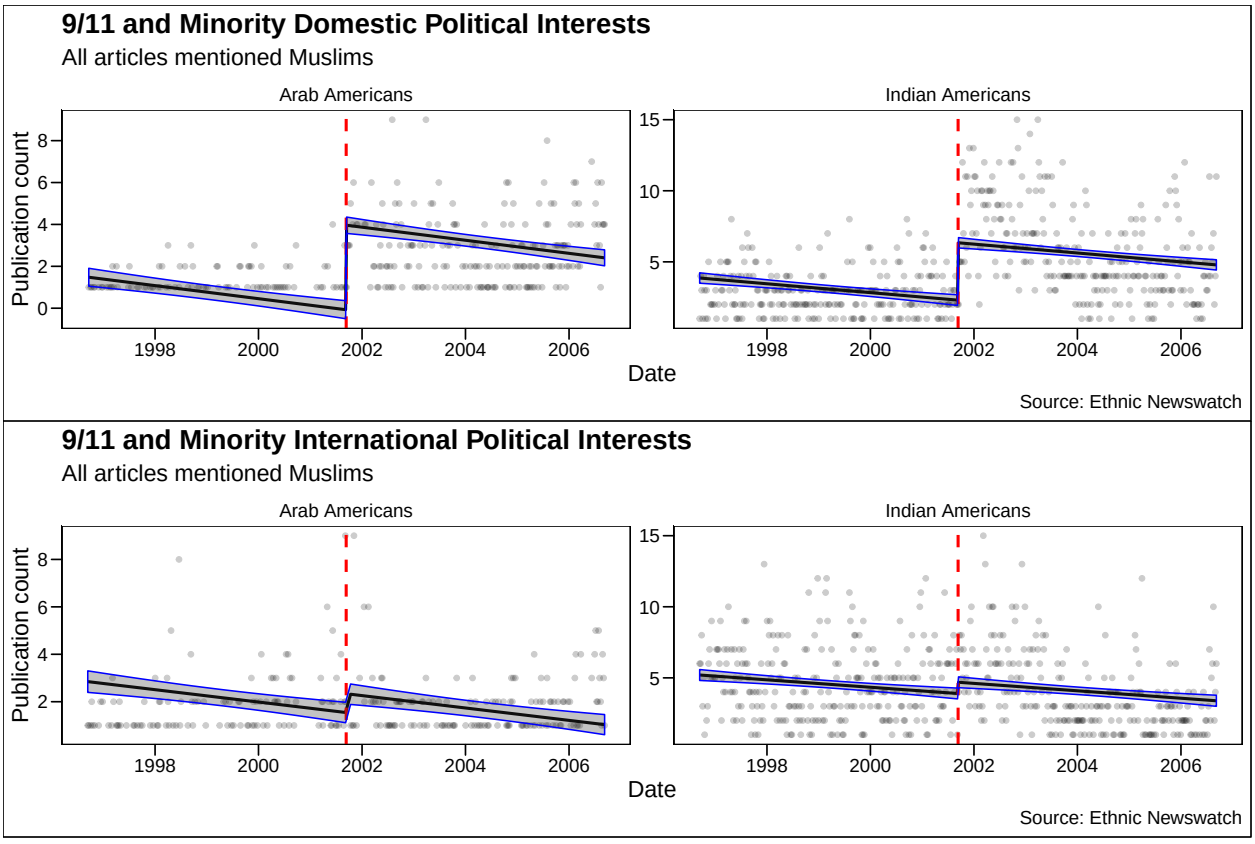

```
ggsave("/home/jae/ITS-Text-Classification/output/cleaned_data_plot.png", height = 10)
```

4. Interrupted time series design analysis

4.1. Base model

```
# Create assignment variable

df_grouped$intervention <- ifelse(df_grouped$date < as.Date("2001-09-11"), 0, 1)

# Divide data

df_domestic <- df_grouped %>%
  filter(domestic == "Domestic")

df_nondomestic <- df_grouped %>%
  filter(domestic != "Domestic")

# Create a function for non-adjusted models

visualize_base <- function(input){

  # Apply OLS regression

  model <- lm(count_ts ~ intervention + date + group, data = input)
```

```r
  # Make predictions

  input$pred <- predict(model, type = "response", input)

  # Create confidence intervals

  ilink <- family(model)$linkinv # Extracting the inverse link from parameter objects

  # Combined prediction outputs

  input <-predict(model, input, se.fit = TRUE)[1:2] %>%
    bind_cols(input) %>%
    mutate(
      upr = ilink(fit + (2 * se.fit)),
      lwr = ilink(fit - (2 * se.fit)))

  # Visualize the outcome

  input %>%
    ggplot(aes(x = date, y = count_ts)) +
    geom_point(alpha = 0.2) +
    facet_wrap(~group, scale = "free_y") +
    geom_line(aes(y = pred), size = 1) +
    ggthemes::theme_base() +
    geom_vline(xintercept = as.Date("2001-09-11"), linetype = "dashed", size = 1, color = "red") +
    ggthemes::theme_base() +
    labs(x = "Date",
         y = "Publication count",
         subtitle = "All articles mentioned Muslims",
         caption = "Source: Ethnic Newswatch") +
    scale_y_continuous(breaks= scales::pretty_breaks()) +
    geom_ribbon(aes(ymin = lwr, ymax = upr),
                alpha = 0.3, color = "blue")
}


# Visualize ITS analysis result for each subset of the data

ggarrange(visualize_base(df_domestic) + ggtitle("9/11 and Minority Domestic Political Interests"),
          visualize_base(df_nondomestic) + ggtitle("9/11 and Minority International Political Interests
          nrow = 2, ncol =1)
```



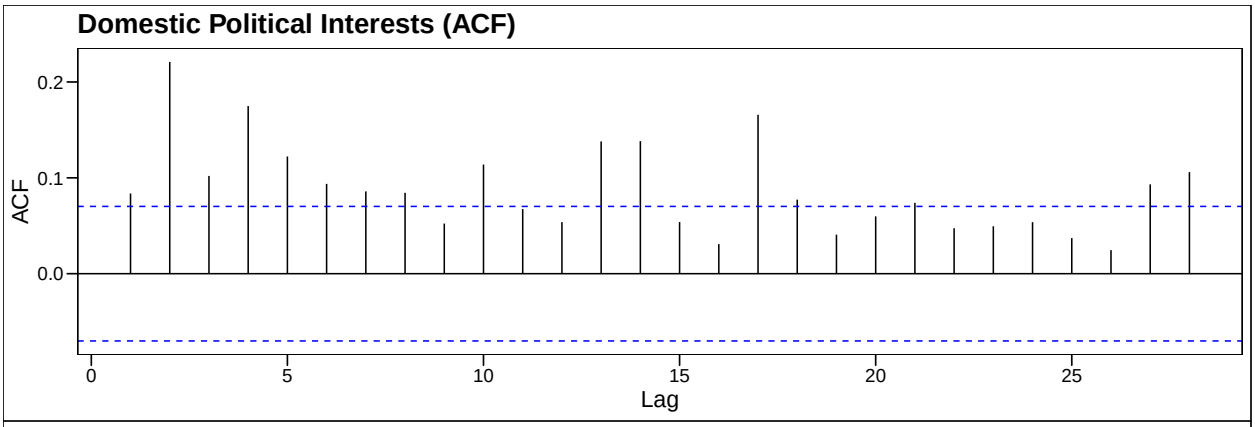

```
ggsave("/home/jae/ITS-Text-Classification/output/its_base_plot.png", height = 10)
```

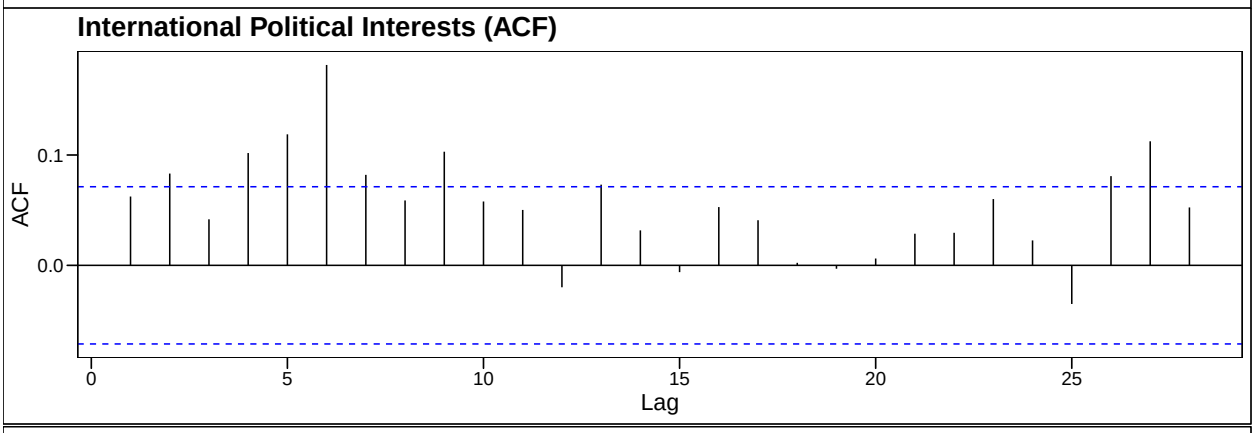
4.2. Check for autocorrelation

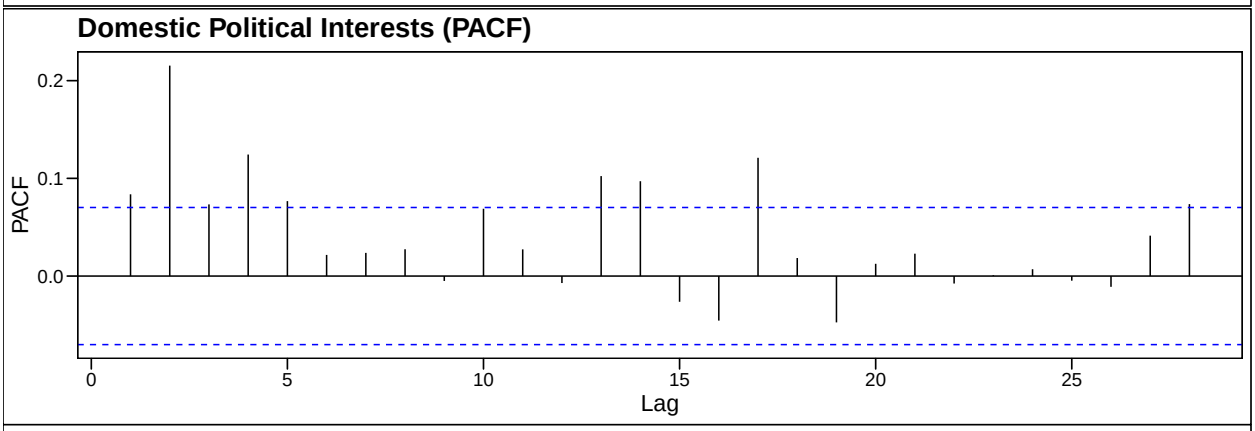
The base model is naive. To make it more robust, first, I assess the correlation between the series and its time lags (autocorrelation). In both cases, ACF (correlation a time series and its lags) test shows there exits a weak seasonal trend.

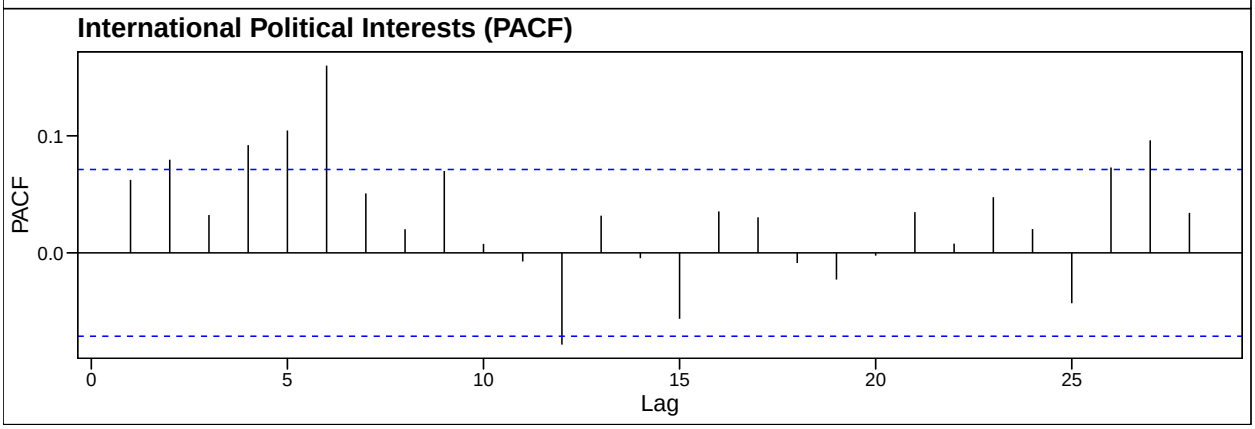
```
# Create the full date

all_days <- seq(min(df_grouped$date), max(df_grouped$date), by ="+1 day")

# Turn it into a dataframe

all_days <- data.frame(date = all_days)

# Filling the missing days

df_grouped <- all_days %>%
  merge(df_grouped, by = "date", all.x = TRUE)

# Divide data

df_domestic <- df_grouped %>%
  filter(domestic == "Domestic")

df_nondomestic <- df_grouped %>%
```

```r
  filter(domestic != "Domestic")

# ACF

acf_plot <- function(input){

  # Model

  model <- lm(count_ts ~ intervention + date + group, data = input)

  # Autocorrelation functions

  pacf <- ggAcf(resid(model)) +
    theme_base()
}

# PACF

pacf_plot <- function(input){

  # Model

  model <- lm(count_ts ~ intervention + date + group, data = input)

  # Autocorrelation functions

  pacf <- ggPacf(resid(model)) +
    theme_base()
}
```


Domestic Political Interests (ACF)
ACF
0.2
0.1
0.0
0
5
10
15
20
25
Lag


International Political Interests (ACF)
ACF
0.1
0.0
0
5
10
15
20
25
Lag


Domestic Political Interests (PACF)
PACF
0.2
0.1
0.0
0
5
10
15
20
25
Lag


International Political Interests (PACF)
PACF
0.1
0.0
0
5
10
15
20
25
Lag

4.3. Parameterizing correlation

The new model parameterizes autocorrelation by accounting for the correlation between the time series data and its time lags. `corARMA() function` defines correlation structure. `p` argument specifies the autoregressive order, and `q` argument specifies the moving average order of the ARMA structure. I ran for loop to find the combination of `p` and `q` that yields the minimum Akaike Information Criterion (AIC).

```
# Create a function

correct_ac <- function(a, b, input){

  model <- gls(count_ts ~ intervention + date + group,
               data = input,
               correlation = corARMA(p = a, q = b, form = ~ date | group),
               na.action = na.omit,
               method = "ML",
               verbose = TRUE)

  results <- data.frame('P' = a,
                        'Q' = b,
                        'logLik' = logLik(model),
                        'AIC' = AIC(model)) # data.frame is better than rbind to keep heterogeneous dat

  return(results)
}
```

```
# Nested for loop

final_list <- list() # list
counter = 1

# I adapted the code from this link: https://www.tylermw.com/theres-no-need-to-apply-yourself/

for (j in c(1,3)){
  for (i in c(1,2,3)){
   final_list[[counter]] <- correct_ac(i, j, df_domestic) # computation
   counter = counter + 1
  }
}



# Put together

final_list <- do.call(rbind, final_list)

final_list %>%
  arrange(AIC)
```

```
##   P Q    logLik      AIC
## 1 3 1 -1643.408 3304.816
## 2 3 3 -1643.122 3308.243
## 3 1 3 -1645.811 3309.621
## 4 2 3 -1648.709 3317.419
## 5 1 1 -1708.324 3430.649
```

```
## 6 2 1 -1708.325 3432.649
```

```
# P

final_list$P[final_list$AIC == min(final_list$AIC)]
```

```
## [1] 3
```

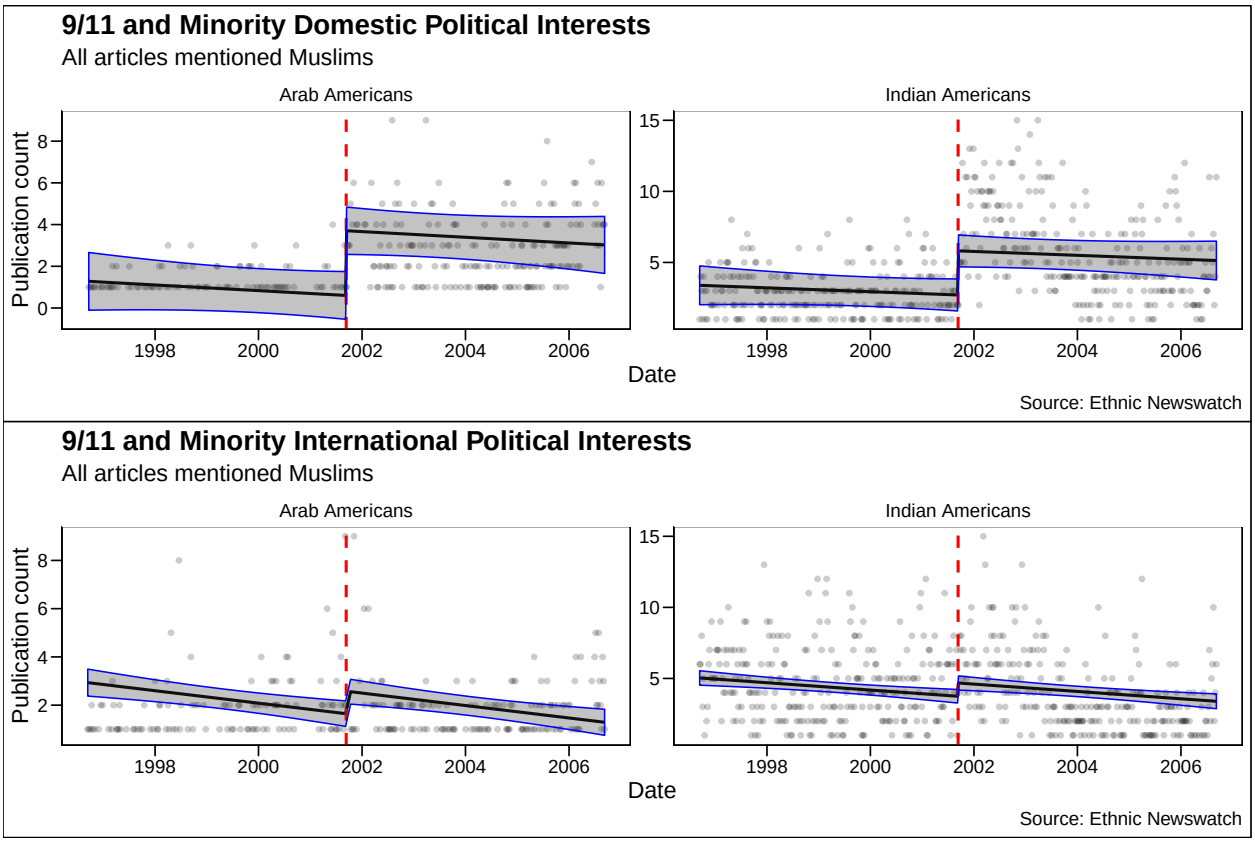
```
# Q

final_list$Q[final_list$AIC == min(final_list$AIC)]
```

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

4.4. Visualization

```
# A new function

visualize_adj <- function(input){

  model <- gls(count_ts ~ intervention + date + group,
               data = input,
               correlation = corARMA(p= 3, q = 1, form = ~ date | group),
               na.action = na.omit)

  # Make predictions

  input$pred <- predict(model, type = "response", input)

  # Combined prediction outputs

  input <- predictSE.gls(model, input, se.fit = TRUE) %>%
    bind_cols(input) %>%
    mutate(
      upr = fit + (2 * se.fit),
      lwr = fit - (2 * se.fit))

  # Visualize the outcome

  input %>%
    ggplot(aes(x = date, y = count_ts)) +
    geom_point(alpha = 0.2) +
    facet_wrap(~group, scale = "free_y") +
    geom_line(aes(y = pred), size = 1) +
    ggthemes::theme_base() +
    geom_vline(xintercept = as.Date("2001-09-11"), linetype = "dashed", size = 1, color = "red") +
    ggthemes::theme_base() +
    labs(x = "Date",
         y = "Publication count",
         subtitle = "All articles mentioned Muslims",
         caption = "Source: Ethnic Newswatch") +
    scale_y_continuous(breaks= scales::pretty_breaks()) +
    geom_ribbon(aes(ymin = lwr, ymax = upr),
                alpha = 0.3, color = "blue")
```

```
}

# Visualize each result in the scatted plot

ggarrange(visualize_adj(df_domestic) + ggtitle("9/11 and Minority Domestic Political Interests"),
          visualize_adj(df_nondomestic) + ggtitle("9/11 and Minority International Political Interests"),
           nrow = 2, ncol =1)
```




```
ggsave("/home/jae/ITS-Text-Classification/output/its_adjusted_plot.png", height = 10)
```