Analysis (Short Version)

Rajath Prabhakar

2024-12-09

```
library(here)
```

```
## here() starts at C:/Users/14049/Desktop/Reddit-Forecasting-Project
```

```
library(tidyverse)
```

```
## Warning: package 'ggplot2' was built under R version 4.3.3
```

```
## Warning: package 'dplyr' was built under R version 4.3.2
```

```
## Warning: package 'stringr' was built under R version 4.3.2
```

```
## Warning: package 'lubridate' was built under R version 4.3.2
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.4     v readr     2.1.4
## v forcats   1.0.0     v stringr   1.5.1
## v ggplot2   3.5.1     v tibble    3.2.1
## v lubridate 1.9.3     v tidyr     1.3.0
## v purrr     1.0.1
```

```
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

Background

Our collaborators at the University of Texas-Austin found and established that COVID-19 case trends inform conversation patterns on Reddit. The paper in question found that during the first phase of the pandemic, anxiety levels on Reddit spiked and positive emotion levels sunk. Shortly thereafter, amidst the onset of the lockdowns, people's thinking reflected attempts to process the subsequent uncertainty. Overall, the magnitude of shifts in people's psychological indicators corresponded highly with the shifting trends of the pandemic.

Our research aims to do the opposite: use psychological indicators to predict COVID trends. The vehicle for this study is, like the UT Austin study, to use Reddit conversation data to accomplish this.

Forecasting diseases are notoriously difficult to accomplish, particularly with a novel disease like COVID-19. This is especially true at times of dynamic increase and decrease in COVID activity. As a result, most efforts

to date have come up short, not only due to this problem, but also due to case reporting standards differing from state to state and different standards in what constitutes a COVID case vs a "probable" COVID case.

Most disease forecasting efforts have involved so-called "big data". Case in point: Google Flu Trends. This study attempted to find the "best matches among 50 million search terms to fit 1152 data points"(Lazer et al.). Akin to trying to find a needle in a haystack, this project fell victim to "big data hubris" - the "implicit assumption that big data are a substitute for, rather than a supplement to, traditional data collection and analysis (Lazer et al.). This resulted in a major overestimation of flu trends, wherein it became a predictor of winter rather than a predictor of flu specifically.

The major lesson? The size of the data isn't everything. Although big data is useful for understanding large-scale human interactions, "small data" contain information not contained in big data. This brings us back to the question of why Reddit specifically.

Reddit is a social forum used to view discussions about the news and politics, as well as any other even mildly conceivable topic. It is useful because within Reddit, there are city-level subreddits dedicated to discussion of affairs in its corresponding city. These subreddits all contain a lot of conversations, posts, and comments, and are available in near-real time if one were to sort by New. Compared to data scraped off of sites like Twitter, Reddit data is smaller. This makes it more reproducible; given that Reddit data is smaller than the data used to make Google Flu Trends, it is more feasible to reproduce the results.

To produce the conversation data analyzed here, subreddits from 29 cities across the US, including Atlanta, Austin, NYC, San Francisco, LA, Phoenix, and Seattle were scraped for posts and comments. These were then turned into psychological indicators using the LIWC software. LIWC, or Linguistic Inquiry and Word Count, is a software that connects the words people use with psychosocial constructs. It accesses each text in the dataset fed into the software and compares it with a similar word in its dictionary. From there, it calculates the percentage of total words represented in each category.

Methods

1. Start with the merged Reddit files. Then apply the user-generated function process_reddit_files() to the Reddit files. This function outputs a daily 7 day rolling average of each variable.

```
#source(here("Functions", "03 - process_reddit_files().R"))
```

For cities with multiple subreddits, namely New York City and Minneapolis, combine the data from those subreddits first, then manually apply the contents of the process_reddit_files() function to it.

Minneapolis

```
# library(data.table)
# library(tidyverse)
# library(zoo)
# min1 <- fread("Source Data/01 - Raw/merged_minneapolis.csv")
# min2 <- fread("Source Data/01 - Raw/merged_twincities.csv")
#
# min3 <- bind_rows(min1, min2) %>%
#   mutate(Date = as_datetime(created_utc, tz = "America/Chicago") %>%
#            as_date()) %>%
#   select(-created_utc) %>%
#   select(author, Date, everything()) %>%
#   mutate(author = replace(author, author %in% c("[deleted]", "AutoModerator"), NA)) %>%
#   select(-c(author, subreddit, parent_id, score, id, Segment)) %>%
#   group_by(Date) %>%
#   summarize_all(mean, na.rm = T) %>%
#   rename_with(~paste0("mean_", .), -Date) %>%
```

```
#   mutate_at(vars(starts_with("mean_")),
#             list(~rollmean(., k = 7, fill = NA, align = "right"))) %>%
#   arrange(Date) %>%
#   filter(Date >= "2019-01-01")
#
# #fwrite(min3, "Source Data/03 - Daily Data/minneapolis_daily.csv")
```

2.

```
#source(here("Functions", "04 - ccf_city_case_files().R"))
```

It performs several processing and analysis tasks to analyze COVID-19 cases and deaths over time across the 29 MSA's (Metropolitan Statistical Area) studied. It does several things:

a)

It reads in several datasets: COVID-19 Cases, COVID-19 Deaths, Census Population Data, and Crosswalk Data.

b)

In the data preprocessing stage of this function, the columns are renamed for readability and the COVID-19 datasets are joined with the census data to link counties to their corresponding MSA.

c)

In the main script, several inputs are taken: folder path and file name, an optional explanatory variable, the MSA code, city and state names, an option to include CCF plots, and which lags to analyze (positive, negative, or both). Using the MSA Code parameter, the script filters the combined cases and deaths data for the given MSA. From this, the weekly cases and deaths are calculated by subtracting cumulative values from the previous week.

This data are then merged with the Reddit data on a weekly basis, and missing values are filtered.

In the actual cross-correlation analysis, the following occurs:

If an explanatory Reddit indicator is specified, the CCF between the weekly COVID Cases and the specified variable is computed. If the indicator is not specified (by default), the CCF between weekly COVID Cases and all Reddit variables are computed.

The output is a table specifying the Reddit variable(s) analyzed, the MSA labels, the maximum ACF value, the lag at which the max ACF occurs, and whether this is in the positive or negative direction of lags.

3.

```
#source(here("Results", "R codes", "Code for combined_data.R"))
```

This analyzes cross-correlations between weekly COVID-19 cases and the Reddit indicators across all 29 cities studied. These results are stored in a single table. The result is shown below.

```
combo <- read_csv("Results/CSV Files/combined_data.csv")
```

```
## Rows: 6960 Columns: 8
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (4): Variable, City, State, Sign
```

```
## dbl (4): Max_ACF, Lag, Est_Population, Population
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
knitr::kable(head(combo))
```

Variable	City	State	Max_ACF	Lag	Sign	Est_Population	Population
mean_Race_Related	Atlanta	Georgia	-0.0009229	-34	Lag < 0	6089755	6103261
mean_Race_Related	Atlanta	Georgia	0.2718514	31	Lag > 0	6089755	6103261
mean_COVID_Related	Atlanta	Georgia	0.3671760	0	Lag < 0	6089755	6103261
mean_COVID_Related	Atlanta	Georgia	0.3575372	2	Lag > 0	6089755	6103261
mean_Abortion_Related	Atlanta	Georgia	0.3976468	-18	Lag < 0	6089755	6103261
mean_Abortion_Related	Atlanta	Georgia	0.0923836	5	Lag > 0	6089755	6103261

4.

```
source(here("Results/R codes", "AverageCityRank.R"))
```

```
##
## Attaching package: 'data.table'

## The following objects are masked from 'package:lubridate':
##
##     hour, isoweek, mday, minute, month, quarter, second, wday, week,
##     yday, year

## The following objects are masked from 'package:dplyr':
##
##     between, first, last

## The following object is masked from 'package:purrr':
##
##     transpose

## Rows: 6960 Columns: 8
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (4): Variable, City, State, Sign
## dbl (4): Max_ACF, Lag, Est_Population, Population
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
## Rows: 27840 Columns: 7
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (4): Variable, City, State, Sign
## dbl (3): year, Max_ACF, Lag
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

This script processes the Max ACF for each variable in each city, ranking them based on the magnitude of their positive and negative lag values, and summarizes the rankings.

```
knitr::kable(rank_avg %>% arrange(avg_rank_pos) %>%
  head() %>%
  mutate(avg_rank_pos = round(avg_rank_pos, 2),
         avg_rank_neg = round(avg_rank_neg, 2)))
```

Variable	avg_rank_pos	avg_rank_neg
mean_emo_sad	2.10	47.00
mean_tone_pos	4.76	32.52
mean_relig	6.34	45.93
mean_comm	8.41	84.97
mean_polite	8.41	43.76
mean_Exclam	11.07	36.14

```
knitr::kable(rank_avg %>% arrange(avg_rank_neg) %>%
  head() %>%
  mutate(avg_rank_pos = round(avg_rank_pos, 2),
         avg_rank_neg = round(avg_rank_neg, 2)))
```

Variable	avg_rank_pos	avg_rank_neg
mean_Abortion_Related	91.76	1.24
mean_achieve	28.90	4.76
mean_want	27.10	5.97
mean_substances	59.10	7.72
mean_Apostro	12.62	8.31
mean_Lifestyle	71.83	9.69

mean_emo_sad has the highest average rank for positive lags, while mean_Abortion_Related has the highest average rank for negative lags.

5.

```
source(here("Results/R codes", "Code for table_of_average_acf_lag.R"))
```

```
## Rows: 6960 Columns: 8
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (4): Variable, City, State, Sign
## dbl (4): Max_ACF, Lag, Est_Population, Population
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
## Rows: 27840 Columns: 7
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (4): Variable, City, State, Sign
```

```
## dbl (3): year, Max_ACF, Lag
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
## `summarise()` has grouped output by 'year'. You can override using the `.groups` argument.
## `summarise()` has grouped output by 'year'. You can override using the `.groups` argument.
```

```
knitr::kable(table1 %>% filter(abs(Average_Lag_negative) <= 6) %>%
               arrange(-abs(Average_Lag_negative)) %>%
               head(10) %>%
               rename(Avg_ACF_Pos = Average_ACF_positive,
                      Avg_Lag_Pos = Average_Lag_positive,
                      Avg_ACF_Neg = Average_ACF_negative,
                      Avg_Lag_Neg = Average_Lag_negative) %>%
               mutate(Avg_ACF_Pos = round(Avg_ACF_Pos, 2),
                      Avg_Lag_Pos = round(Avg_Lag_Pos, 2),
                      Avg_ACF_Neg = round(Avg_ACF_Neg, 2),
                      Avg_Lag_Neg = round(Avg_Lag_Neg, 2))
   )
```

Variable	Avg_ACF_Pos	Avg_Lag_Pos	Avg_ACF_Neg	Avg_Lag_Neg
mean_Analytic	0.12	5.88	0.32	-6.00
mean_emo_pos	0.24	3.00	0.26	-5.60
mean_number	0.17	4.00	0.24	-5.07
mean_Tone	0.29	4.75	0.24	-5.00
mean_attention	0.07	1.00	0.13	-3.78
mean_emotion	0.26	2.00	0.17	-3.33
mean_money	0.07	3.86	0.27	-3.00
mean_work	0.04	1.00	0.11	-2.81
mean_fatigue	0.15	1.20	0.12	-2.18
mean_home	0.19	2.45	0.26	-2.11

Because we are primarily concerned with negative lags (since they indicate leading indicators of Weekly COVID Cases), we look at highest optimal correlations in the negative lag direction. Therefore, mean_Analytic has the highest correlation with weekly COVID Cases (0.318) with an average lag of 6 weeks.

6.

Breakdown cross-correlation analysis by year using the following function:

```
ccf_by_year <- function(folder_path, filename, explanatory = NULL, code = NULL, city = NULL,
                        state = NULL) {
  file_path <- file.path(folder_path, filename)
  #print(file_path)
  data <- fread(here(file_path)) %>%
    separate(week, into = c("year", "month", "day"), sep = "-")

  city_msa_cases <- cases %>%
    filter(MSA_Code == code) %>%
    separate(week, into = c("year", "month", "day"), sep = "-") %>%
```

```
  group_by(year, month, day, MSA_Code, MSA_Title, FIPS, Admin2) %>%
  summarize(Cumulative_Cases = Cases,
            Est_Population = unique(estpop2020),
            Population = unique(pop2020)) %>%
  ungroup() %>%
  group_by(FIPS, Admin2,MSA_Code, MSA_Title, year, month, day) %>%
  slice_tail(n = 1) %>%
  na.omit() %>%
  ungroup() %>%
  group_by(year, month, day, MSA_Code, MSA_Title) %>%
  summarize(Cumulative_Cases = sum(Cumulative_Cases),
            Est_Population = sum(Est_Population),
            Population = sum(Population)) %>%
  ungroup()
city_msa_deaths <- deaths %>%
  filter(MSA_Code == code) %>%
  separate(week, into = c("year", "month", "day"), sep = "-") %>%
  group_by(year, month, day, MSA_Code, MSA_Title, FIPS, Admin2) %>%
  summarize(Cumulative_Deaths = Deaths,
            Est_Population = unique(estpop2020),
            Population = unique(pop2020)) %>%
  ungroup() %>%
  group_by(FIPS, Admin2, MSA_Code, MSA_Title, year, month, day) %>%
  slice_tail(n = 1) %>%
  na.omit() %>%
  ungroup() %>%
  group_by(year, month, day, MSA_Code, MSA_Title) %>%
  summarize(Cumulative_Deaths = sum(Cumulative_Deaths),
            Est_Population = sum(Est_Population),
            Population = sum(Population)) %>%
  ungroup()
city_combined <- left_join(city_msa_cases, city_msa_deaths) %>%
  ungroup() %>%
  arrange(year, month, day) %>%
  mutate(Weekly_Cases = Cumulative_Cases - lag(Cumulative_Cases, n = 1, default = 0),
         Weekly_Deaths = Cumulative_Deaths - lag(Cumulative_Deaths, n = 1, default = 0)) %>%
  left_join(data) %>%
  na.omit()

ccf_columns <- city_combined %>% select(starts_with("mean_")) %>% names()
years <- as.vector(unique(city_combined$year))

full_acf_table <- data.frame()
for (y in years) {
  city_combined1 <- city_combined %>% filter(year == y)
  for (column in ccf_columns) {

    ccf_result <- ccf(city_combined1$Weekly_Cases, city_combined1[[column]], plot = FALSE,
                      lag.max = 40)
    #print(str(ccf_result))

    acf_table <- data.frame(Lag = ccf_result$lag, ACF = ccf_result$acf, year = y) %>%
      mutate(Sign = case_when(
```

```
          Lag <= 0 ~ "Lag < 0",
          TRUE ~ "Lag > 0")) %>%
        group_by(Sign) %>%
        summarise(Max_ACF = max(ACF),
                  Lag = Lag[which.max(ACF)]) %>%
        ungroup() %>%
        mutate(Variable = column,
               City = city,
               State = state,
               Est_Population = mean(city_combined1$Est_Population),
               year = y) %>%
        select(year, Variable, City, State, Max_ACF, Lag, Sign)
      #acf_table_year <- rbind(acf_table_year, acf_table)
      full_acf_table <- rbind(full_acf_table, acf_table)
    }
  }
  return(full_acf_table)
}
```

For each year, the optimal ACF and corresponding lag were calculated for each variable in each lag direction.

```
combined_data_year <- read_csv("Results/CSV Files/combined_data_year.csv")
```

```
## Rows: 27840 Columns: 7
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (4): Variable, City, State, Sign
## dbl (3): year, Max_ACF, Lag
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
negative <- combined_data_year %>%
  filter(Lag <= 0 & Lag >= -6) %>%
  group_by(year, Variable) %>%
  summarize(Average_ACF_negative = mean(Max_ACF),
            Average_Lag_negative = mean(Lag))
```

```
## `summarise()` has grouped output by 'year'. You can override using the
## `.groups` argument.
```

```
positive <- combined_data_year %>%
  filter(Lag > 0 & Lag <= 6) %>%
  group_by(year, Variable) %>%
  summarize(Average_ACF_positive = mean(Max_ACF),
            Average_Lag_positive = mean(Lag))
```

```
## `summarise()` has grouped output by 'year'. You can override using the
## `.groups` argument.
```

```
table2 <- left_join(negative, positive, by = c("year", "Variable"))
```

Then, to find the final variables of interest, the top variables for each year were identified using the code below. The results are thus shown:

```
common <- table2 %>%
  group_by(year) %>%
  slice_max(Average_ACF_negative, n = 30) %>%
  ungroup() %>%
  group_by(Variable) %>%
  summarize(count = n(),
            ACF = mean(Average_ACF_negative),
            Lag = mean(Average_Lag_negative)) %>%
  arrange(desc(count), desc(ACF)) %>%
  filter(count == 4)
knitr::kable(common)
```

Variable	count	ACF	Lag
mean_COVID_Related	4	0.5947301	-0.5990726
mean_illness	4	0.4932069	-0.5176293
mean_health	4	0.4576412	-0.6134948
mean_Affect	4	0.4135997	-1.4768519
mean_Physical	4	0.4063716	-0.6851607
mean_family	4	0.3843147	-1.3985746

COVID_Related, illness, health, Affect, Physical, and family appear in the top 30 variables in each of the 4 years studied. Therefore, we will use these variables to forecast weekly COVID Cases.

7.

```
source(here("Functions/06 - forecast_reddit().R"))
```

```
##
## Attaching package: 'zoo'

## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric

## Warning: package 'ciTools' was built under R version 4.3.3

## ciTools version 0.6.1 (C) Institute for Defense Analyses

## Warning: package 'forecast' was built under R version 4.3.3

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

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

## Rows: 4920 Columns: 131

## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr    (3): MSA_Code, MSA_Title, City
## dbl  (126): Cumulative_Cases, Cumulative_Deaths, Est_Population, Population,...
## lgl    (1): mean_function
## date   (1): week
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
## [conflicted] Will prefer dplyr::select over any other package.
## [conflicted] Will prefer dplyr::filter over any other package.
## [conflicted] Will prefer dplyr::lag over any other package.
## [conflicted] Will prefer dplyr::between over any other package.
```

There are two functions of interest in this script: get_0week_forecasts() and get_1week_forecasts.

get_0week_forecasts()

This function generates nowcasts using the auto.arima() function. This function automatically selects the best fitting ARIMA model - in this case - for each week. This attempts to correct for changing trends week to week. For every week, it uses a training period up to the week before the cutoff date, and predicts the COVID case counts of the cutoff date. An example is shown below:

```
ex1 <- get_0week_forecasts(cutoff_date = "2023-02-19", city = "Atlanta", explanatory = "mean_illness",
                    case_lag = 0, reddit_lag = 0)
ex1$forecast %>% select(week, City, `Lo 95`: Weekly_Cases, mean_illness) %>% knitr::kable()
```

week	City	Lo 95	Lo 80	Point Forecast	Hi 80	Hi 95	Weekly_Cases	mean_illness
2023-02-19	Atlanta	0	0	3063.815	10670.91	14697.86	7864	0.1273618

```
ex1$model %>% knitr::kable()
```

date	ar1	ma1	ma2	mean_illness
2023-02-19	-0.4262509	0.2524146	0.5634114	7816.327

Here, the cutoff date is February 19, 2023. The training period would be all weeks in the source data frame before that date, and would generate a model based on that data. Then, using that data, it would predict cases for February 19. In this example, it is important to note that neither the cases nor the Reddit variable(s) are lagged. However, the script is capable of accounting for such terms.

The model is as follows:

$WeeklyCases = -0.426x_t + w_t + 0.252w_{t-1} + 0.563w_{t-2} + \beta z_t$

The model for 2023-02-19 is an ARIMA(1,0,2) model.

get_1week_forecasts()

This function generates 7-day forecasts, also using the auto.arima() function. For every week, it uses a training period up to and including the cutoff date, and predicts the COVID case counts of the week immediately following the cutoff date.

```
ex2 <- get_1week_forecasts(cutoff_date = "2023-02-19", city = "Atlanta", explanatory = "mean_illness",
                    case_lag = 0, reddit_lag = 0)
ex2$forecast %>% select(week, City, `Lo 95`: Weekly_Cases, mean_illness) %>% knitr::kable()
```

week	City	Lo 95	Lo 80	Point Forecast	Hi 80	Hi 95	Weekly_Cases	mean_illness
2023-02-26	Atlanta	0	0	3294.182	10901.27	14928.22	2187	0.1568344

```
ex2$model %>% knitr::kable()
```

date	ar1	ma1	ma2	mean_illness
2023-02-19	-0.4262509	0.2524146	0.5634114	7816.327

Here again, the cutoff date is February 19, 2024. The training period would be all weeks in the source data frame up to but not including that date, generating a training model based on that data. Then, using the Reddit data of February 19, it would predict COVID case counts for February 26.

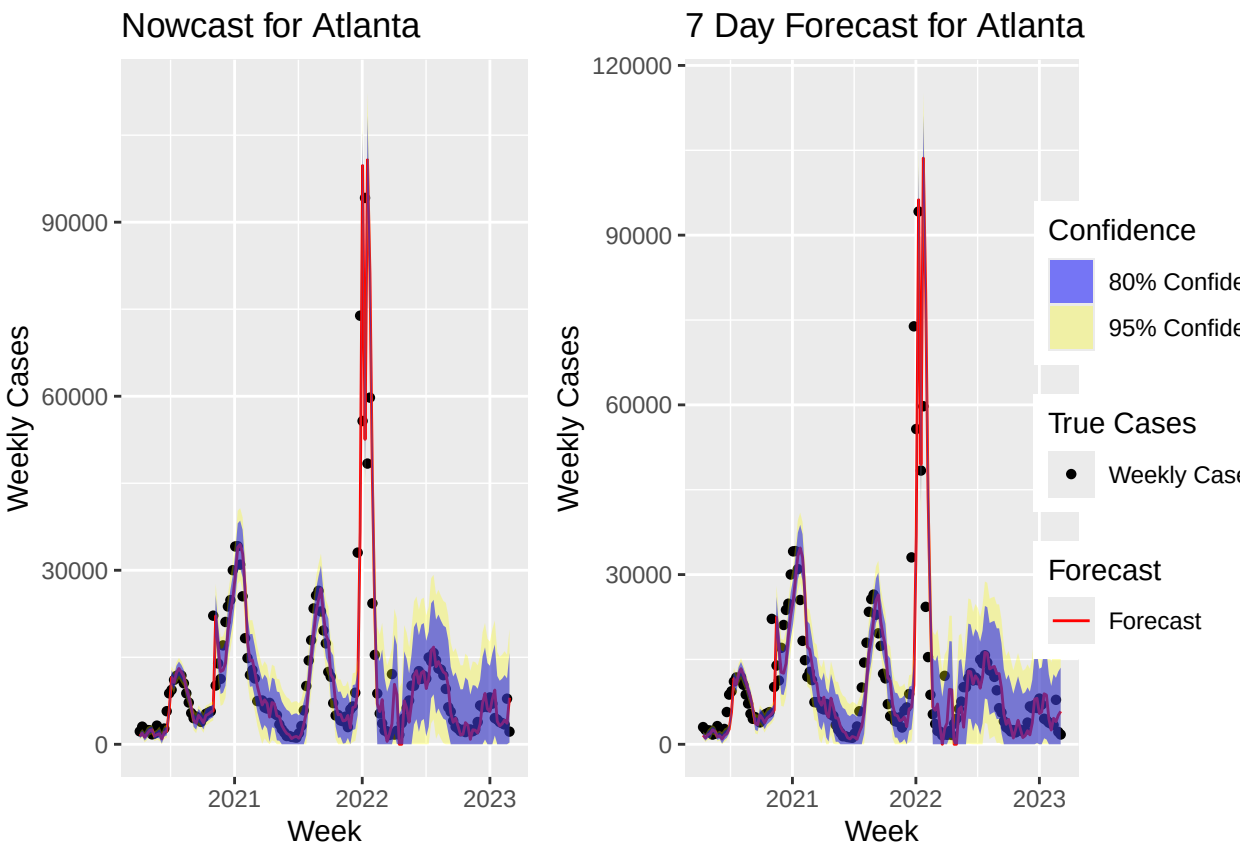
The forecast model for the same date is identical to the nowcast model.

Both these functions are capable of including lagged case and Reddit terms. They also include the models for each week along with the forecasts themselves.

8.

Thus, using the 6 common top variables (Affect, COVID_Related, family, health, illness, and physical), nowcasts and forecasts from 4/1/2020 to 3/5/2023 were generated. They were then graphed. An example is shown below:

```
load("Environments/Cumulative ARIMA/forecast_city_list_COVID_Related.RData")
forecast_city_list_COVID_Related$Atlanta$forecast_graph
```



As we can see, the predicted weekly COVID Cases follow similar trends to the actual weekly COVID case counts when using mean_COVID_Related as an external regressor. This suggests that Reddit Data does not undermine the accuracy of the predictions. But the question that needs to be answered is: does any one Reddit indicator alone significantly improve the accuracy of the predictions?

Forecast Results

```
files <- list.files("Environments/Cumulative ARIMA", full.names = T)
forecasts <- list()
for (file in files) {
  explanatory_variable <- gsub(".*_(.*)\\.RData", "\\1", file)
  forecast0_complete <- list()
  forecast7_complete <- list()
  #env <- new.env()  # Create a new environment for loading
  f <- get(load(file))
  for (city in names(f)) {
    forecast0_complete[[city]] <- f[[city]]$forecast0
    forecast7_complete[[city]] <- f[[city]]$forecast7
  }
  forecast0_complete <- bind_rows(forecast0_complete) %>%
    select(week, City, everything()) %>%
    mutate(Coverage_95 = ifelse(Weekly_Cases >= Lo_95 & Weekly_Cases <= Hi_95, "yes", "no"),
           Abs_Error = abs(Weekly_Cases - Point_Forecast),
           Abs_Pct_Error = (abs(Weekly_Cases - Point_Forecast)/abs(Weekly_Cases))*100) %>%
    select(week, City, MSA_Code, MSA_Title, Lo_95, Lo_80, Point_Forecast, Hi_80, Hi_95, Weekly_Cases,
           Coverage_95, Abs_Error, Abs_Pct_Error, everything())
  forecast7_complete <- bind_rows(forecast7_complete) %>%
```

```
    select(week, City, everything()) %>%
    mutate(Coverage_95 = ifelse(Weekly_Cases >= Lo_95 & Weekly_Cases <= Hi_95, "yes", "no"),
           Abs_Error = abs(Weekly_Cases - Point_Forecast),
           Abs_Pct_Error = (abs(Weekly_Cases - Point_Forecast)/abs(Weekly_Cases))*100) %>%
    select(week, City, MSA_Code, MSA_Title, Lo_95, Lo_80, Point_Forecast, Hi_80, Hi_95, Weekly_Cases,
           Coverage_95, Abs_Error, Abs_Pct_Error, everything())
  forecast0_stats <- forecast0_complete %>%
    group_by(City) %>%
    summarize(Coverage_95 = sum(Coverage_95 == "yes")/n(),
              MAE = mean(Abs_Error),
              MAPE = mean(Abs_Pct_Error)) %>%
    mutate(proj_type = "Nowcast",
           explanatory = explanatory_variable)
  forecast7_stats <- forecast7_complete %>%
    group_by(City) %>%
    summarize(Coverage_95 = sum(Coverage_95 == "yes")/n(),
              MAE = mean(Abs_Error),
              MAPE = mean(Abs_Pct_Error)) %>%
    mutate(proj_type = "Forecast",
           explanatory = explanatory_variable)
  forecast_stats <- bind_rows(forecast0_stats, forecast7_stats) %>%
    mutate(proj_type = as.factor(proj_type))
  forecasts[[explanatory_variable]] <- forecast_stats
  #print(explanatory_variable)

}
```

```
forecast_stats_complete <- bind_rows(forecasts) %>%
  mutate(explanatory = as.factor(explanatory))
```

```
aggregate_nowcast <- forecast_stats_complete %>%
  filter(MAPE != Inf & !is.nan(MAPE)) %>%
  group_by(explanatory, proj_type) %>%
  summarize(MAE = mean(MAE),
            MAPE = mean(MAPE),
            Coverage_95 = mean(Coverage_95)) %>%
  filter(proj_type == "Nowcast") %>%
  ungroup() %>%
  mutate(Pct_Improvement_MAE = if_else(explanatory != "Baseline",
                                       ((MAE - MAE[explanatory == "Baseline"]) / MAE[explanatory == "Baseli
         Pct_Improvement_MAPE = if_else(explanatory != "Baseline",
                                       ((MAPE - MAPE[explanatory == "Baseline"]) / MAPE[explanatory == "Bas
         Pct_Improvement_Coverage = if_else(explanatory != "Baseline",
                                       ((Coverage_95 - Coverage_95[explanatory == "Baseline"]) / Coverage_9
                                       0)) %>%
  select(explanatory, Pct_Improvement_MAE, Pct_Improvement_MAPE, Pct_Improvement_Coverage) %>%
  mutate(Pct_Improvement_MAE = round(Pct_Improvement_MAE, 2),
         Pct_Improvement_MAPE = round(Pct_Improvement_MAPE, 2),
         Pct_Improvement_Coverage = round(Pct_Improvement_Coverage, 2))
```

```
## `summarise()` has grouped output by 'explanatory'. You can override using the
## `.groups` argument.
```

```
aggregate_forecast <- forecast_stats_complete %>%
  filter(MAPE != Inf & !is.nan(MAPE)) %>%
  group_by(explanatory, proj_type) %>%
  summarize(MAE = mean(MAE),
            MAPE = mean(MAPE),
            Coverage_95 = mean(Coverage_95)) %>%
  filter(proj_type == "Forecast") %>%
  ungroup() %>%
  mutate(Pct_Improvement_MAE = if_else(explanatory != "Baseline",
                                       ((MAE - MAE[explanatory == "Baseline"]) / MAE[explanatory == "Baseli
         Pct_Improvement_MAPE = if_else(explanatory != "Baseline",
                                       ((MAPE - MAPE[explanatory == "Baseline"]) / MAPE[explanatory == "Base
         Pct_Improvement_Coverage = if_else(explanatory != "Baseline",
                                       ((Coverage_95 - Coverage_95[explanatory == "Baseline"]) / Coverage_9
                                       0)) %>%
  select(explanatory, Pct_Improvement_MAE, Pct_Improvement_MAPE, Pct_Improvement_Coverage) %>%
  mutate(Pct_Improvement_MAE = round(Pct_Improvement_MAE, 2),
         Pct_Improvement_MAPE = round(Pct_Improvement_MAPE, 2),
         Pct_Improvement_Coverage = round(Pct_Improvement_Coverage, 2))
```

```
## `summarise()` has grouped output by 'explanatory'. You can override using the
## `.groups` argument.
```

```
knitr::kable(aggregate_nowcast)
```

explanatory	Pct_Improvement_MAE	Pct_Improvement_MAPE	Pct_Improvement_Coverage
Affect	-0.98	4.45	-0.04
Baseline	0.00	0.00	0.00
family	1.39	7.14	-0.25
health	-0.97	2.00	-0.32
illness	-1.22	1.98	-0.71
Physical	2.36	6.34	-0.35
Related	3.60	2.25	-1.20

```
knitr::kable(aggregate_forecast)
```

explanatory	Pct_Improvement_MAE	Pct_Improvement_MAPE	Pct_Improvement_Coverage
Affect	-0.37	3.52	-0.74
Baseline	0.00	0.00	0.00
family	1.17	5.12	-0.91
health	-1.42	-0.17	-0.33
illness	-1.34	-0.66	-0.83
Physical	0.55	4.05	-0.50
Related	1.27	-1.06	-0.95

For each measure, the model using the Reddit data was compared to the Baseline model that did not use any Reddit data as predictors. For nowcasts, models using Affect, health, and illness indicators showed slight improvements in accuracy compared to the baseline model using Mean Absolute Error. However, this

improvement wasn't considered significant. To be considered significant, we want a 10% or greater improvement. However, using Mean Absolute Percentage Error, all the non-baseline models were measureably worse than the baseline model, shown by their consistently positive values.

For forecasts, the results were similar when MAE was considered. However, when MAPE was considered, models using health, illness, or COVID-Related indicators as predictors showed improvements in accuracy over the baseline model. However, these improvements were not considered significant for the same reason.

Next Steps

We have established that Reddit conversation data do not significantly hurt the accuracy of either COVID 7-day forecasts or COVID nowcasts. However, no one Reddit indicator is able to significantly improve the accuracy of these forecasts. It would be useful to explore several options:

1. Examine performance of Reddit models vs baseline models by city- it is possible that there is improvement in some cities but not others
2. Run same analysis with multiple of the Reddit variables as external regressors. No one of the variables significantly improves the accuracy of the forecasts and nowcasts, but a combination of 2 or more of the regressors might.
3. Explore time-varying covariance. This would account for the idea that one or more of the variables are not constant throughout the 3 years studied, whether that is the COVID case trends themselves or the Reddit conversation indicator trends.

Sources

https://www.science.org/doi/10.1126/sciadv.abg7843

https://gking.harvard.edu/sites/scholar.harvard.edu/files/gking/files/0314policyforumff.pdf

https://journal.r-project.org/articles/RJ-2022-002/

https://www.liwc.app/static/documents/LIWC-22%20Manual%20-%20Development%20and%20Psychometrics.pdf