Hurricane Ida on Twitter

Joseph Holler, Andrey Cao

12/15/2023

Hurricane Ida on X/Twitter

Author: Joseph Holler, Andrey Cao Created: Fall 2019 Updated: Fall 2023

This analysis was developed with assistance from: - Casey Lilley's GEOG 323 final project available at: https://caseylilley.github.io/finalproj.html
- Leah Wasser and Carson Farmer's *Twitter Data in R Using RTweet* tutorial on EarthLab at: https://www.earthdatascience.org/courses/earth-analytics/get-data-using-apis/use-twitter-api-r/ - Michael Minn's *Basic Spatial Point Pattern Analysis in R* tutorial at: http://michaelminn.net/tutorials/r-point-analysis/ - Help from fellow student Elise Chan

Abstract

This original study conducted by Professor Joseph Holler examines the impacts of natural disasters on social media communication. Focusing on X/Twitter data during Hurricane Dorian, Holler (2021) replicated the methods of Wang et al (2016) for the case of the hurricane's landfall on the U.S. mainland during the 2019 Atlantic Hurricane season. Holler modified the methodology for normalizing tweet data by creating a normalized Tweet difference index and extended the methodology to test for spatial clustering with the local Getis-Ord statistic. This replication of Holler (2021) conducted by Cao (2023) examines X/Twitter data during Hurricane Ida in 2021. As a replication study, we aimed to successfully generate the figures and type of results produced in Holler (2021) with Hurricane Ida data. We found differences in the computational environment that made packages such as 'rtweet' not compatible with the code of Holler (2021). Additionally, the new lack of free access to X/Twitter data led us to alter the normalization of the data to use data from the Census API.

Study metadata

- `Key words`: hurricane, ida, twitter, x
- `Subject`: Social and Behavioral Sciences: Geography: Geographic Information Sciences
- `Date created`: Replication started November 16, 2023
- `Date modified`: December 13, 2023
- `Spatial Coverage`: continental United States
- `Spatial Resolution`: GPS coordinates
- `Spatial Reference System`: NAD 1983
- `Temporal Coverage`: 2021-08-29 to 2021-9-10
- `Temporal Resolution`: Tweets are measured to the second (formatted as 'day hour:minute:second')
- `Funding Name`: n/a
- `Funding Title`: n/a

- `Award info URI`: n/a
- `Award number`: n/a

Original study spatio-temporal metadata

- `Spatial Coverage`: continental United States
- `Spatial Resolution`: GPS coordinates
- `Spatial Reference System`: NAD 1983
- `Temporal Coverage`: 2019-09-05 to 2019-09-10
- `Temporal Resolution`: Tweets are measured to the second (formatted as 'day hour:minute:second')

#Study design

This is a replication of this study of Hurricane Dorian conducted by Professor Joseph Holler. The original study archetype was observational, as it was comparing X/Twitter data for the trending news surrounding #Sharpiegate and Hurricane Dorian.

Holler (2021) focused on the research question surrounding whether the trending news and social media content (#Sharpiegate) regarding a false narrative of risk would cluster along the real hurricane (Dorian) track. Using a temporal analysis graph, a content analysis graph, a map of X/Twitter activity, and a hot spot map, Holler (2021) visualizes patterns of the observed tweets.

This replication will use X/Twitter data surrounding Hurricane Ida. We attempt to successfully replicate the figures presented in the original Holler (2021) study.

Materials and procedure

Computational environment (planned deviations)

Load the R project saved in the root directory of this repository, so that the working directory is the root directory of the repository.

The following code snippet may ask you if you want to update some packages. Enter '3' (None) when prompted.

```
# Install the 'remotes' package if you haven't already
#install.packages("remotes")
# Load the 'remotes' package
library(remotes)
library(here)
```

```
## here() starts at C:/Users/ajcao/Documents/GitHub/RPl-Dorian-Ida
```

```
# installing a previous version of rtweet
# in 2021, would have been version 0.7.0.
#install.packages("devtools")
#library(devtools)
#install_version("rtweet", version = "0.7.0",
#                repos = "https://cran.r-project.org")
library(rtweet)

packages <- c(
```

```r
  "here", "svDialogs", "tidyverse",
  "tidytext", "tm", "igraph", "ggraph",
  "tidycensus", "sf", "spdep"
)

package_check <- lapply(
  packages,
  FUN = function(x) {
    if (!require(x, character.only = TRUE)) {
      library(x, character.only = TRUE)
    }
  }
)
```

```
## Loading required package: svDialogs

## Loading required package: tidyverse

## -- 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   3.4.4     v tibble    3.2.1
## v lubridate 1.9.3     v tidyr     1.3.0
## v purrr     1.0.2
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter()  masks stats::filter()
## x purrr::flatten() masks rtweet::flatten()
## x dplyr::lag()     masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
## Loading required package: tidytext
##
## Loading required package: tm
##
## Loading required package: NLP
##
##
## Attaching package: 'NLP'
##
##
## The following object is masked from 'package:ggplot2':
##
##     annotate
##
##
## Loading required package: igraph
##
##
## Attaching package: 'igraph'
##
##
## The following objects are masked from 'package:lubridate':
##
##     %--%, union
```

```
##
##
## The following objects are masked from 'package:dplyr':
##
##     as_data_frame, groups, union
##
##
## The following objects are masked from 'package:purrr':
##
##     compose, simplify
##
##
## The following object is masked from 'package:tidyr':
##
##     crossing
##
##
## The following object is masked from 'package:tibble':
##
##     as_data_frame
##
##
## The following objects are masked from 'package:stats':
##
##     decompose, spectrum
##
##
## The following object is masked from 'package:base':
##
##     union
##
##
## Loading required package: ggraph
##
## Loading required package: tidycensus
##
## Loading required package: sf
##
## Linking to GEOS 3.11.2, GDAL 3.7.2, PROJ 9.3.0; sf_use_s2() is TRUE
##
## Loading required package: spdep
##
## Loading required package: spData
##
## To access larger datasets in this package, install the spDataLarge
## package with: `install.packages('spDataLarge',
## repos='https://nowosad.github.io/drat/', type='source')`
```

```r
# save the R processing environment, including date in file name
writeLines(
  capture.output(sessionInfo()),
  here("procedure", "environment", paste0("r-environment-", Sys.Date(), ".txt"))
  )
```

Data and variables

Since the X/Twitter API is no longer accessible for free, we used older tweet files that must remain private due to Twitter privacy regulations. Contact Professor Joseph Holler if you are interested in getting access to these files.

Primary data (tevent_raw.RDS, tevent_raw2.RDS, tevent_raw3.RDS, tevent_raw4.RDS)

- `Title`: X/Twitter Data (Hurricane Ida)
- `Abstract`: Tweet data during Hurricane Ida (2021)
- `Spatial Coverage`: continental United States
- `Spatial Resolution`: GPS coordinates
- `Spatial Reference System`: NAD 1983
- `Temporal Coverage`: 2021-08-29 to 2021-09-10
- `Temporal Resolution`: Tweets are measured to the second (day hour:minute:second)
- `Lineage`: The data files were made using the X/Twitter API. Though the code to retrieve the data is within the study, new regulations of the API prevent that code from working.
- `Distribution`: Cannot be distributed publicly, contact Professor Joseph Holler for access
- `Constraints`: Legal constraints for privacy, X/Twitter data cannot be distributed publicly for user privacy
- `Data Quality`: n/a
- `Variables`: Here are the variables used within the study (out of a total of 90 variables)

Label	Alias	Definition	Type	Accuracy	Domain	Missing Data Value(s)	Missing Data Frequency
status_id	unique status identifier	ID number for each status	character	...	1431923022718459904 to 1436353772679204866	...	no missing data
created_at	time of tweet	post time measured as (day hour:minute:second)	character	to the second	(2021-08-29 10:13:47) to (2021-09-10 15:40:00)	...	no missing data
text	text of tweet	any characters contained within tweet	character	...	...	...	no missing data
place_type	place type of tweet location	X/Twitter geographic place type	character	based on X/Twitter categorization	city, neighborhood, poi, or NA	NA	about 94.79% of place_type missing
coords_coords	coordinates	coordinates in a (x,y) format	list of numbers	...	...	c(NA, NA)	about 99.55% of coords_coords missing

Label	Alias	Definition	Type	Accuracy	Domain	Missing Data Value(s)	Missing Data Frequency
bbox_coords	bounding box coordinates	eight coordinates that form edges of a region	list of numbers	...	...	c(NA, NA, NA, NA, NA, NA, NA, NA)	about 94.79% of bbox_coords missing

The following section showcases the method for using the X/Twitter API. This code no longer works, but it will be showcased for transparency.

Skip to load original search results

Set up Twitter Application

You will need Twitter Developer API access to run this analysis: See https://cran.r-project.org/web/packages/rtweet/vignettes/auth.html for instructions.

reference for search_tweets function: https://rtweet.info/reference/search_tweets.html - don't add any spaces in between variable name and value for your search
e.g. n=1000 is better than n = 1000 - the first parameter in quotes is the search string - `n=10000` asks for 10,000 tweets - if you want more than 18,000 tweets, change `retryonratelimit` to `TRUE` and wait 15 minutes for every batch of 18,000 - `include_rts=FALSE` excludes retweets. - `token` refers to the twitter token you defined above for access to your twitter developer account - `geocode` is equal to a string with three parts: longitude, latitude, and distance with the units mi for miles or km for kilometers

This code block will ask you for your twitter application name, key, and secret. Then it will launch a web browser and prompt you to log in to Twitter to authenticate the application.

Never save your API keys in code where it can be committed and synced to GitHub! The code below is configured to save your keys in the environment, and this Git repository is set up to ignore the R environment data file.

```
# Twitter application values
twitter_vars <- list(
  app = "enter Twitter application name",
  key = "enter Twitter API key",
  secret = "enter Twitter API secret key"
)

# if Twitter token has already been created, auto-fill dialogue with its values
if (exists("twitter_token")) {
  twitter_vars$app <- twitter_token$app$appname
  twitter_vars$key <- twitter_token$app$key
  twitter_vars$secret <- twitter_token$app$secret
}

twitter_token <- create_token(
  app = dlgInput("Twitter App Name:", twitter_vars$app)$res,
  consumer_key = dlgInput("Consumer Key:", twitter_vars$key)$res,
  consumer_secret = dlgInput("Consumer Secret:", twitter_vars$secret)$res,
  access_token = NULL,
```

```
  access_secret = NULL
)
```

Pre-processing

- Acquire Twitter data for analysis
- Filter Twitter data for good geographic information and convert to Lat/Long coordinates

Search for Hurricane Dorian tweets

get tweets for hurricane Dorian, searched on September 11, 2019 **Warning**: this code will no longer result in the same data! It is here for reference or replication work only.

```
dorian <- read_sf(here("data", "raw", "private", "covidcase080120.gpkg"))

dorian_raw <- search_tweets("dorian OR hurricane OR sharpiegate",
  n = 200000, include_rts = FALSE,
  token = twitter_token,
  geocode = "32,-78,1000mi",
  retryonratelimit = TRUE
)

# write status id's for results of the original twitter search
write.table(dorian_raw$status_id,
  here("data", "raw", "public", "dorianids.txt"),
  append = FALSE, quote = FALSE, row.names = FALSE, col.names = FALSE
)
```

Search for generic tweets after hurricane season

get tweets without any text filter for the same geographic region in November, searched on November 19, 2019 the query searches for all verified or unverified tweets, i.e. everything

Warning: this code will no longer result in the same data! It is here for reference or replication work only.

```
november_raw <- search_tweets("-filter:verified OR filter:verified",
  n = 200000, include_rts = FALSE,
  token = twitter_token,
  geocode = "32,-78,1000mi",
  retryonratelimit = TRUE
)

# write status id's for results of the original twitter search
write.table(november_raw$status_id,
  here("data", "raw", "public", "novemberids.txt"),
  append = FALSE, quote = FALSE, row.names = FALSE, col.names = FALSE
)
```

Rehydrate the twitter data

Twitter does not permit redistribution of Twitter data, with the exception of tweet status ids. For the purposes of transparency and reproducibility, researchers may include a list of status id's with their publication.

The process of using those status ids to query Twitter for the full tweet data is called **rehydrating**—like going back to the analogous fire hose of big data. **Warning**: Twitter users and individual tweets can be deleted over time, therefore the results of rehydration will be similar, but not identical to, the original Twitter data used for this research.
Warning: It may take more than an hour to rehydrate the raw tweets with Twitter queries, therefore you may select to load only the derived status ids, which have filtered only the tweets with valid geographic data (approximately one tenth of the raw tweets)

Load Twitter status ids

You have a choice of loading all of the status IDs from the original search, or a list of status IDs filtered to contain only those with precise enough geographic data. Loading the original search will be a more exact replication while loading filtered data will speed up processing without changing the final result. Neither process will result in identical findings compared to the original study because Tweets can be deleted after posting.

```
# load tweet status id's for Hurricane Dorian search results

filtered <- dlgList(
  choices = c("raw", "derived"),
  title = "Which Dorian ids?")$res

dorianids <-
  data.frame(read.table(here("data", filtered, "public", "dorianids.txt"),
    numerals = "no.loss"
  ))

# load cleaned status id's for November general twitter search
filtered <- dlgList(
  choices = c("raw", "derived"),
  title = "Which November ids?")$res


novemberids <-
  data.frame(read.table(here("data", "raw", "public", "novemberids.txt"),
    numerals = "no.loss"
  ))
```

Rehydrate Twitter status ids

This operation may take over an hour to run on all of the raw tweets

```
# rehydrate dorian tweets
dorian_raw <- rehydratoR(twitter_token$app$key, twitter_token$app$secret,
  twitter_token$credentials$oauth_token,
  twitter_token$credentials$oauth_secret, dorianids,
  base_path = NULL, group_start = 1
)

# rehydrate november tweets
november_raw <- rehydratoR(twitter_token$app$key, twitter_token$app$secret,
  twitter_token$credentials$oauth_token,
```

```
  twitter_token$credentials$oauth_secret, novemberids,
  base_path = NULL, group_start = 1
)
```

Prior observations

At the time of this preanalysis, we have obtained the full X/Twitter data we are going to use for this study. We have no prior conception of the potential results that will come from this set of Hurricane Ida data except for brief research on the path of the hurricane. Our main objective of this replication is to use this data and successfully produce the figures featured in the original code.

Bias and threats to validity

The different normalization method (using Census API instead of X/Twitter data) may make it harder for us to compare certain figures to the original study. Additionally, the hot spot/cluster figure of the study may be affected by the modifiable areal unit problem, so we should be aware of the state population data we use during the normalization process.

Data transformations

Starting from the raw files, we will bind the four files (rbind function) and then sort out the duplicates (distinct function). Then we will convert geographic information into lat/long coordinates. After, we will clean the text and parse out any stop words we would like to exclude.

Later, for the normalization, we must first add the county data from the Census API. Then, we will join the tweets to the county data. Lastly, we will create the spatial weight matrix along with the Getis-Ord G* Statistic.

Load the original search results

Planned Deviation: This is where we load the Hurricane Ida data.

Students in the GEOG 323 Open Source GIScience course may download the original search results from the private course data repository: https://github.com/GIS4DEV/geog323data/raw/main/ida/tevent_raw.RDS https://github.com/GIS4DEV/geog323data/raw/main/ida/tevent_raw2.RDS https://github.com/GIS4DEV/geog323data/raw/main/ida/tevent_raw3.RDS https://github.com/GIS4DEV/geog323data/raw/main/ida/tevent_raw4.RDS

Save the four `.RDS` files to the `data/raw/private` folder.

```
tevent_raw1 <- readRDS(here("data", "raw", "private", "tevent_raw.RDS"))
tevent_raw2 <- readRDS(here("data", "raw", "private", "tevent_raw2.RDS"))
tevent_raw3 <- readRDS(here("data", "raw", "private", "tevent_raw3.RDS"))
tevent_raw4 <- readRDS(here("data", "raw", "private", "tevent_raw4.RDS"))
```

You must bind the four files using the rbind function to continue. To avoid duplicates, you can use the distinct function.

```
ida_raw_all <- rbind(tevent_raw1, tevent_raw2, tevent_raw3, tevent_raw4) #includes duplicates
ida_raw <- ida_raw_all %>% distinct(status_id, .keep_all = TRUE) #excludes duplicates
```

Process geographic data in tweets

reference for lat_lng function: https://rtweet.info/reference/lat_lng.html adds a lat and long field to the data frame, picked out of the fields that you indicate in the c() list sample function: lat_lng(x, coords = c("coords_coords", "bbox_coords"))

list and count unique place types NA results included based on profile locations, not geotagging / geocoding. If you have these, it indicates that you exhausted the more precise tweets in your search parameters and are including locations based on user profiles

```
count(ida_raw, place_type)
```

```
## # A tibble: 6 x 2
##   place_type        n
##   <chr>         <int>
## 1 admin          3541
## 2 city          15454
## 3 country          25
## 4 neighborhood     41
## 5 poi             477
## 6 <NA>         355805
```

Convert geographic information into lat/long coordinates If you have loaded filtered status ids, or you have already run this code, you will not notice a difference in `place_type` or `n` because the data has already been processed.

```
# convert GPS coordinates into lat and lng columns
# do not use geo_coords! Lat/Lng will be inverted
ida <- lat_lng(ida_raw, coords = c("coords_coords"))

# select any tweets with lat and lng columns (from GPS) or
# designated place types of your choosing
ida <- subset(
  ida,
  place_type == "city" | place_type == "neighborhood" |
    place_type == "poi" | !is.na(lat)
)

# convert bounding boxes into centroids for lat and lng columns
ida <- lat_lng(ida, coords = c("bbox_coords"))

# re-check counts of place types
count(ida, place_type)
```

```
## # A tibble: 6 x 2
##   place_type       n
##   <chr>        <int>
## 1 admin          497
## 2 city         15454
## 3 country          1
## 4 neighborhood    41
## 5 poi            477
## 6 <NA>            10
```

Save processed tweets

Optionally, Save the tweet id's to the `data\derived\public` folder as plain text.
Save the full tweet data to `data\derived\private` folder as RDS files.
Full Tweet data cannot be shared with the public, therefore it is stored in a folder ignored by Git.

```
write.table(ida$status_id,
  here("data", "derived", "public", "idaids.txt"),
  append = FALSE, quote = FALSE, row.names = FALSE, col.names = FALSE
)

saveRDS(ida, here("data", "derived", "private", "ida.RDS"))
```

Load processed tweets

Optionally, Load processed twitter data

```
ida <- readRDS(here("data", "derived", "private", "ida.RDS"))
```

Clean the text data

Parse the tweet data for plain language, and parse tweet text into words. Remove stop words and our search terms.

```
# remove urls, fancy formatting, etc. in other words, clean the text content
ida_text <- ida %>%
  select(text) %>%
  plain_tweets()

# parse out words from tweet text
ida_words <- ida_text %>% unnest_tokens(word, text)

# how many words do you have including the stop words?
word_count <- list(before = count(ida_words)$n)

# create list of stop words (useless words not worth analyzing)
data("stop_words")

# add "t.co" twitter links to the list of stop words
# also add the twitter search terms to the list
stop_words <- stop_words %>%
  add_row(word = "t.co", lexicon = "SMART") %>%
  add_row(word = "hurricane", lexicon = "Search") %>%
  add_row(word = "ida", lexicon = "Search") #%>%

# delete stop words from dorianWords with an anti_join
ida_words <- anti_join(ida_words, stop_words, by="word")

# how many words after removing the stop words?
word_count <- append(
  word_count,
  list(after = count(ida_words)$n)
```

```
  )
print(word_count)
```

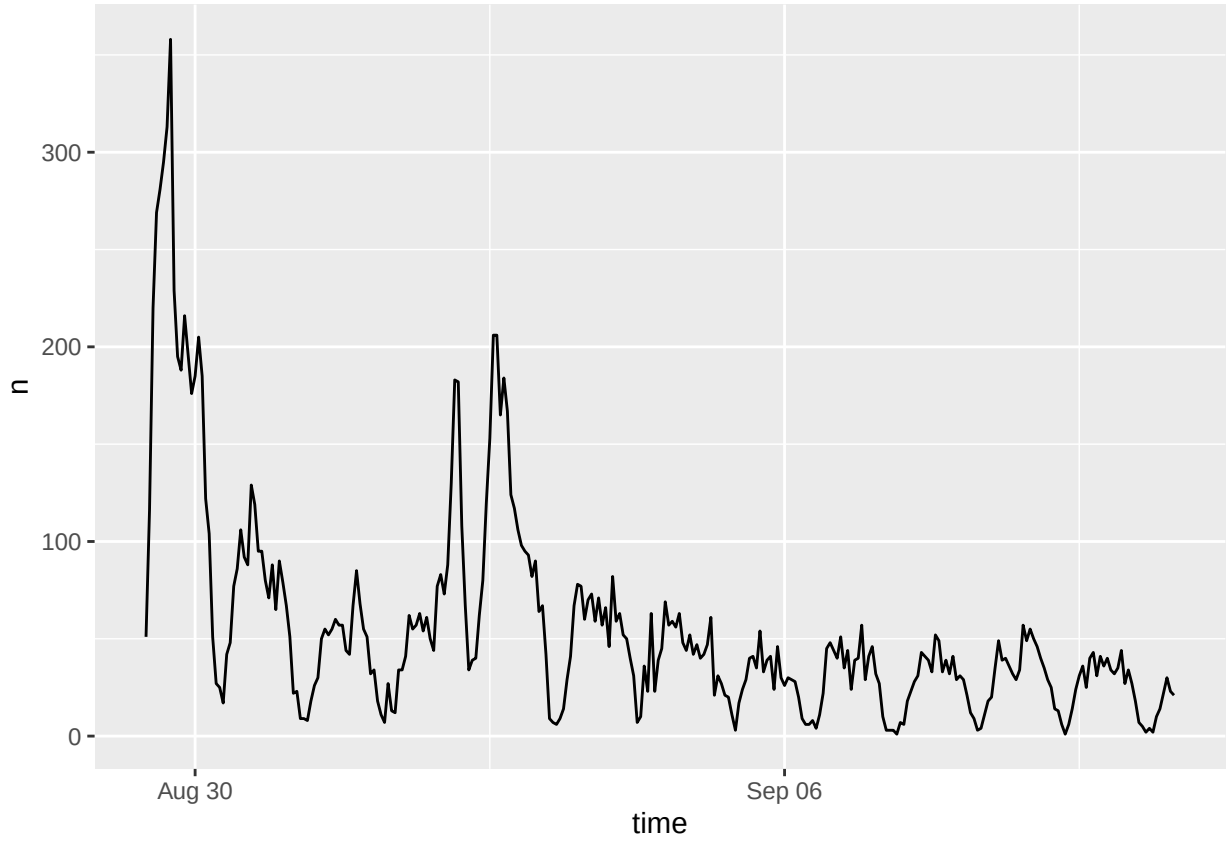
```
## $before
## [1] 398763
##
## $after
## [1] 190906
```

Analysis

Temporal Analysis

Create a temporal data frame and graph it

```
ida_tweets_by_hour <- ts_data(ida, by = "hours")
ts_plot(ida, by = "hours")
```

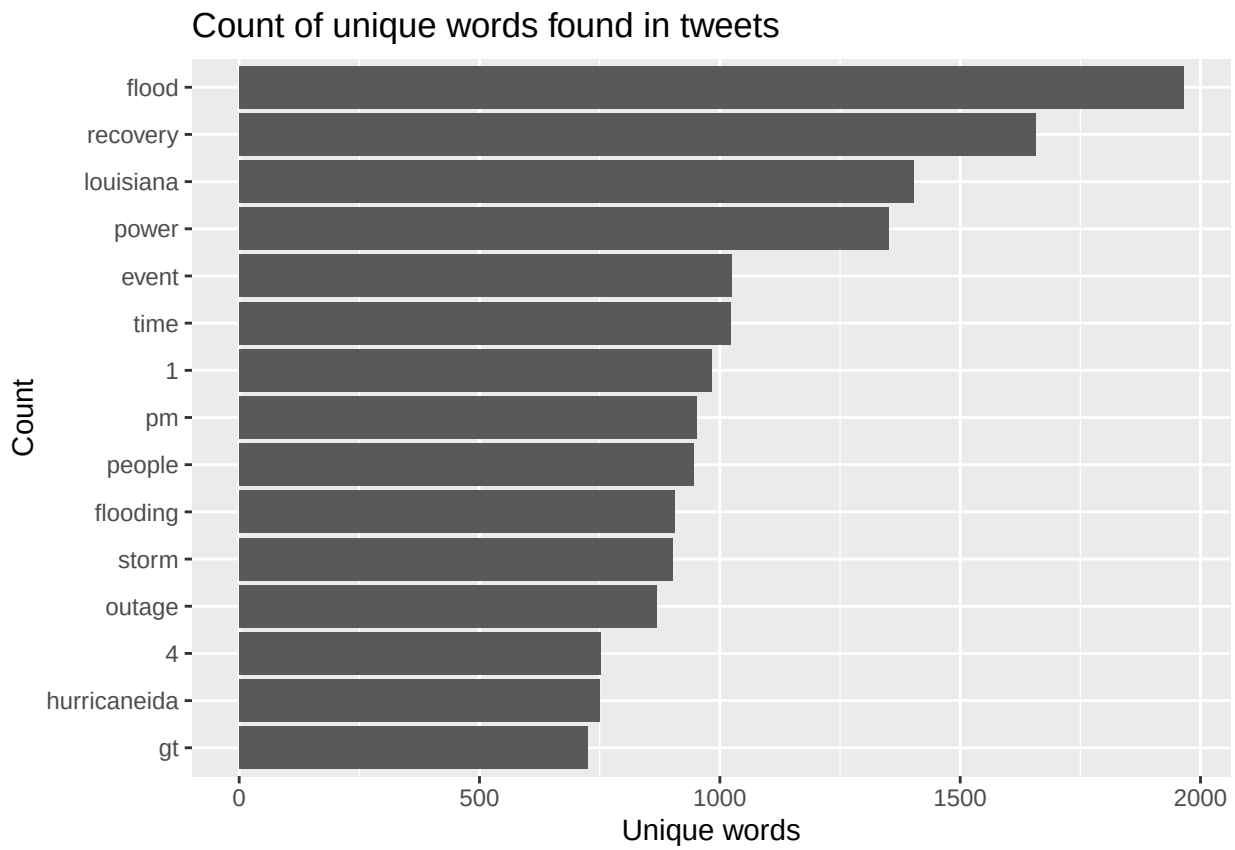


Text Analysis

Graph frequencies of words

```
ida_words %>%
  count(word, sort = TRUE) %>%
  top_n(15) %>%  #deprecated; try replacing with slice_min() or slice_max()
  mutate(word = reorder(word, n)) %>%
  ggplot(aes(x = word, y = n)) +
  geom_col() +
  xlab(NULL) +
  coord_flip() +
  labs(
    x = "Count",
    y = "Unique words",
    title = "Count of unique words found in tweets"
  )
```

```
## Selecting by n
```



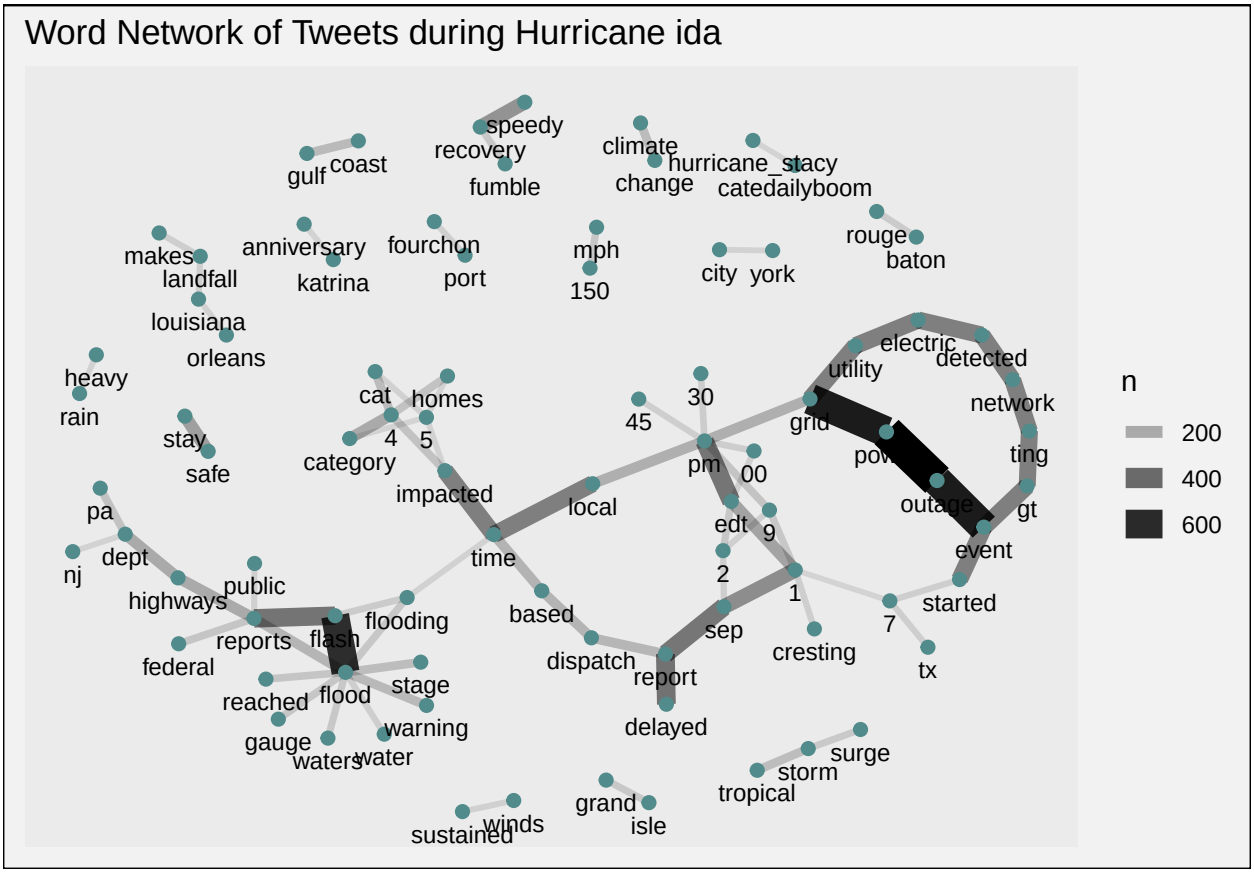
Analyze and graph word association

Unplanned deviation: We changed the filter to 50 from 25 to make the figure clearer.

```r
# separate words and count frequency of word pair occurrence in tweets
ida_word_pairs <- ida_text %>%
  mutate(text = removeWords(tolower(text), stop_words$word)) %>%
  unnest_tokens(paired_words, text, token = "ngrams", n = 2) %>%
  separate(paired_words, c("word1", "word2"), sep = " ") %>%
  count(word1, word2, sort = TRUE)

# graph a word cloud with space indicating association.
# you may change the filter to filter more or less than pairs with 25 instances
ida_word_pairs %>%
  filter(n >= 50 & !is.na(word1) & !is.na(word2)) %>% #can change
  graph_from_data_frame() %>%
  ggraph(layout = "fr") +
  geom_edge_link(aes(edge_alpha = n, edge_width = n)) +
  geom_node_point(color = "darkslategray4", size = 2) +
  geom_node_text(aes(label = name), vjust = 1.8, size = 3) +
  labs(
    title = "Word Network of Tweets during Hurricane ida",
    x = "", y = ""
  ) +
  theme(
    plot.background = element_rect(
    fill = "grey95",
    colour = "black",
    size = .5
    ),
    legend.background = element_rect(fill = "grey95")
  )
```

```
## Warning: The ‘size‘ argument of ‘element_rect()‘ 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.
```

```
## Warning: Using the ‘size‘ aesthetic in this geom was deprecated in ggplot2 3.4.0.
## i Please use ‘linewidth‘ in the ‘default_aes‘ field and elsewhere instead.
## This warning is displayed once every 8 hours.
## Call ‘lifecycle::last_lifecycle_warnings()‘ to see where this warning was
## generated.
```



Spatial Analysis (planned deviation)

First, you will need a Census API. You can sign up for one here: https://api.census.gov/data/key_signup.html

```
census_api_key(dlgInput(
  "Enter a Census API Key",
  Sys.getenv("CENSUS_API_KEY")
)$res,
overwrite = TRUE
)
```

```
## To install your API key for use in future sessions, run this function with ‘install = TRUE‘.
```

```
counties <- get_acs(
  "county",
  variables = "B01003_001",
  year = 2021,
  output = "wide",
  geometry = TRUE,
  keep_geo_vars = TRUE
)
```

```
## Getting data from the 2017-2021 5-year ACS
```

```
## Downloading feature geometry from the Census website.  To cache shapefiles for use in future sessions

##   |                                                                              |
```

```
counties <- filter(
  counties,
  STATEFP %in% c(
    "12", "11", "10", "09", "48", "40", "20", "19", "26", "55", "27", "38",
    "46", "31", "35", "08", "54", "51", "50", "47", "45", "44", "42", "39",
    "34", "33", "29", "28", "25", "24", "23", "22", "21", "18", "17", "13",
    "37", "36", "05", "01"

  )
) %>%
  rename(POP = "B01003_001E", MOE = "B01003_001M") %>%
  mutate(DENSITY = POP/ALAND) #find density


saveRDS(counties, here("data", "derived", "public", "counties.RDS"))
```

Load counties Optionally, load counties from the `counties.RDS` file saved with the repository

```
counties <- readRDS(here("data", "derived", "public", "counties.RDS"))
```

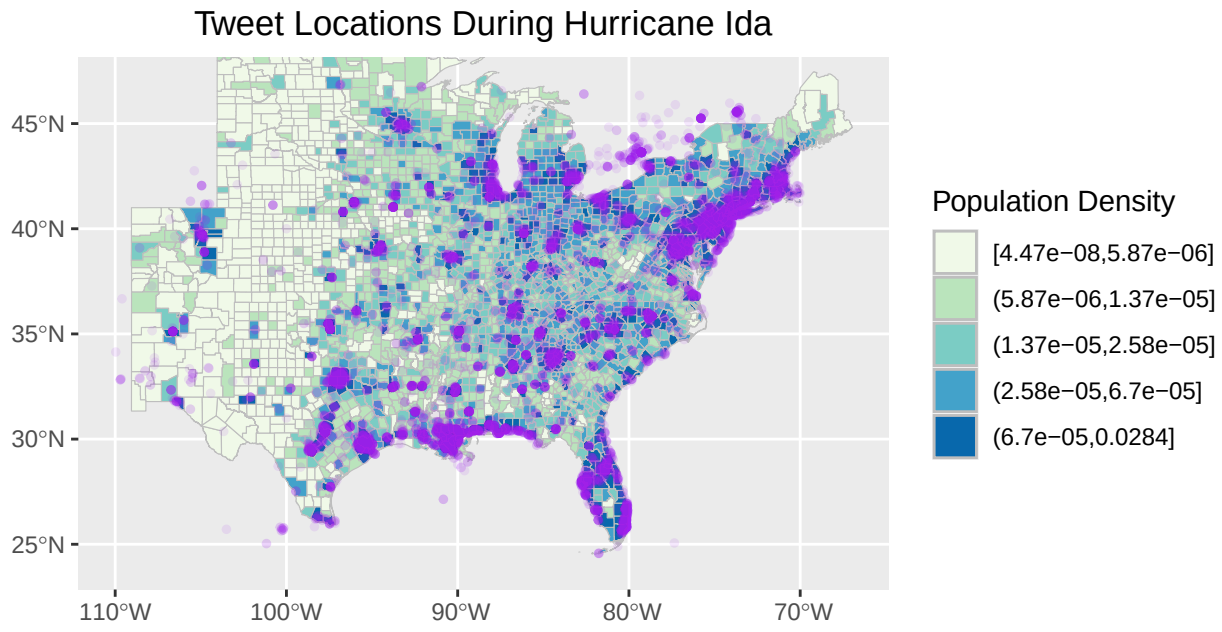
Map Population Density and Tweet Points

map results with GGPlot note: cut_interval is an equal interval classification function, while cut_number is a quantile / equal count function you can change the colors, titles, and transparency of points

```
ggplot() +
  geom_sf(data = counties,
    aes(fill = cut_number(DENSITY, 5)), color = "grey") +
  scale_fill_brewer(palette = "GnBu") +
  guides(fill = guide_legend(title = "Population Density")) +
  geom_point(
    data = ida, aes(x = lng, y = lat),
    colour = "purple", alpha = 0.1, size = 1
  ) +
  labs(title = "Tweet Locations During Hurricane Ida") +
  theme(
    plot.title = element_text(hjust = 0.5),
    axis.title.x = element_blank(),
```

```
    axis.title.y = element_blank()
  ) +
  xlim(-110, -67) + #note: changed the xlim to include more states in the visualization
  ylim(24, 47)
```

```
## Warning: Removed 12 rows containing missing values (`geom_point()`).
```



Join Tweets to Counties

Spatially join Dorian and November tweets to counties and save counties as `counties_tweet_counts.RDS`

Planned Deviation: This is where we change the normalization process to use Census data.

```
ida_sf <- ida %>%
  st_as_sf(coords = c("lng", "lat"), crs = 4326) %>% # make point geometries
  st_transform(4269) %>% # transform to NAD 1983
  st_join(select(counties, GEOID)) # spatially join counties to each tweet

ida_by_county <- ida_sf %>%
  st_drop_geometry() %>% # drop geometry / make simple table
  group_by(GEOID) %>% # group by county using GEOID
  summarise(ida = n()) # count # of tweets

counties <- counties %>%
```

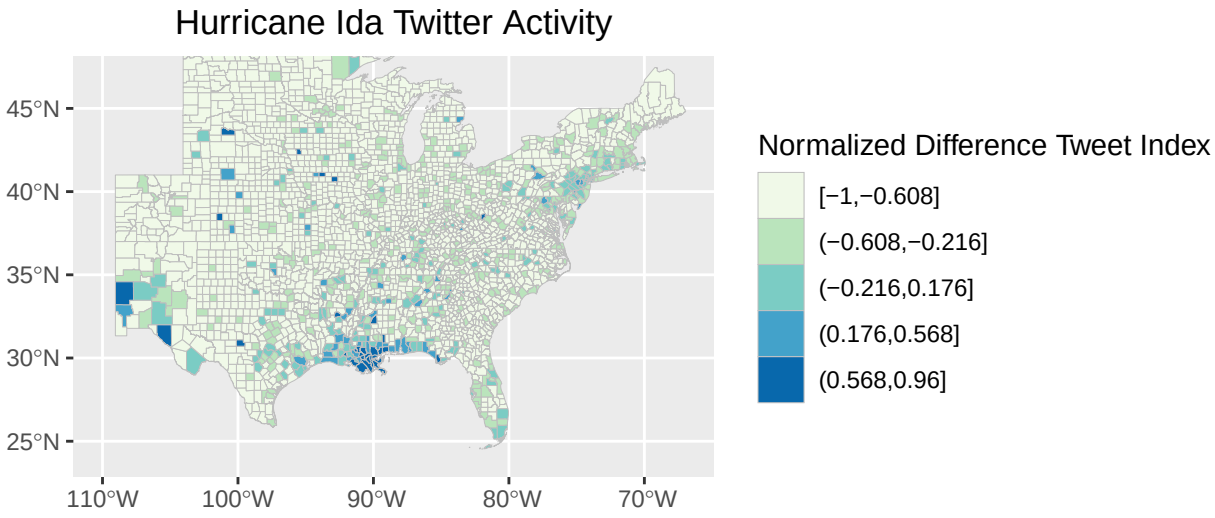
```
  left_join(ida_by_county, by = "GEOID") %>% # join count of tweets to counties
  mutate(ida = replace_na(ida, 0), # replace nulls with 0's
         popestimate = POP/10000, #using census population to mimic tweets per 10,000
         idarate = ida/(popestimate))

#normalization

counties <- counties %>%
  mutate(ndti = (ida - popestimate)/(ida + popestimate)) %>%
  mutate(ndti = replace_na(ndti, 0)) # replace NULLs with 0's

# save counties geographic data with derived tweet rates
saveRDS(counties, here("data", "derived", "public", "counties_tweet_counts.RDS"))
```

Optionally, begin here by loading counties with Twitter data

```
counties <- readRDS(here("data", "derived", "public", "counties_tweet_counts.RDS"))
```

Note: The original code used "dorrate," but we replaced it with "ndti."

```
ggplot() +
  geom_sf(data = counties,
    aes(fill = cut_interval(ndti, 5)), color = "grey", lwd=0.01) +
  scale_fill_brewer(palette = "GnBu") +
  guides(fill = guide_legend(title = "Normalized Difference Tweet Index")) +
  labs(title = "Hurricane Ida Twitter Activity") +
  theme(
    plot.title = element_text(hjust = 0.5),
    axis.title.x = element_blank(),
    axis.title.y = element_blank()
  ) +
  xlim(-110, -67) +
  ylim(24, 47)
```



Spatial Cluster Analysis

Create Spatial Weight Matrix

Use 110km Euclidean distance and include self in the weight matrix

```
county_coords <- counties %>%
  st_centroid() %>% # convert polygons to centroid points
  st_coordinates() # convert to simple x,y coordinates to play with stdep

thresdist <- county_coords %>%
  dnearneigh(0, 110, longlat = TRUE) %>% # use geodesic distance of 110km
  # distance should be long enough for every feature to have >= one neighbor
  include.self() # include a county in its own neighborhood (for G*)

thresdist # view statistical summary of the nearest neighbors
```

```
## Neighbour list object:
## Number of regions: 2791
## Number of nonzero links: 67689
## Percentage nonzero weights: 0.8689573
## Average number of links: 24.2526
## 2 disjoint connected subgraphs
```

Optionally, plot the spatial weight matrix results This should result in a very dense graph, because each county is connected to all other counties within 110 km.

```
plot(thresdist, county_coords, lwd=0.01) # plot nearest neighbor ties
```

Calculate Getis-Ord G* Statistic

```
# Create weight matrix from the neighbor objects
dwm <- nb2listw(thresdist, zero.policy = T)

######## Local G* Hotspot Analysis ########
# Get Ord G* statistic for hot and cold spots
counties$locG <- as.vector(localG(counties$idarate,
  listw = dwm,
  zero.policy = TRUE
))

# may be same as:
# counties$dorrate %>% localG(listw = dwm, zero.policy = TRUE) %>% as.vector()

# check summary statistics of the local G score
summary(counties$locG)
```

```
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
```

```
## -1.21221 -0.70274 -0.49452 -0.03238 -0.14489 21.69767
```

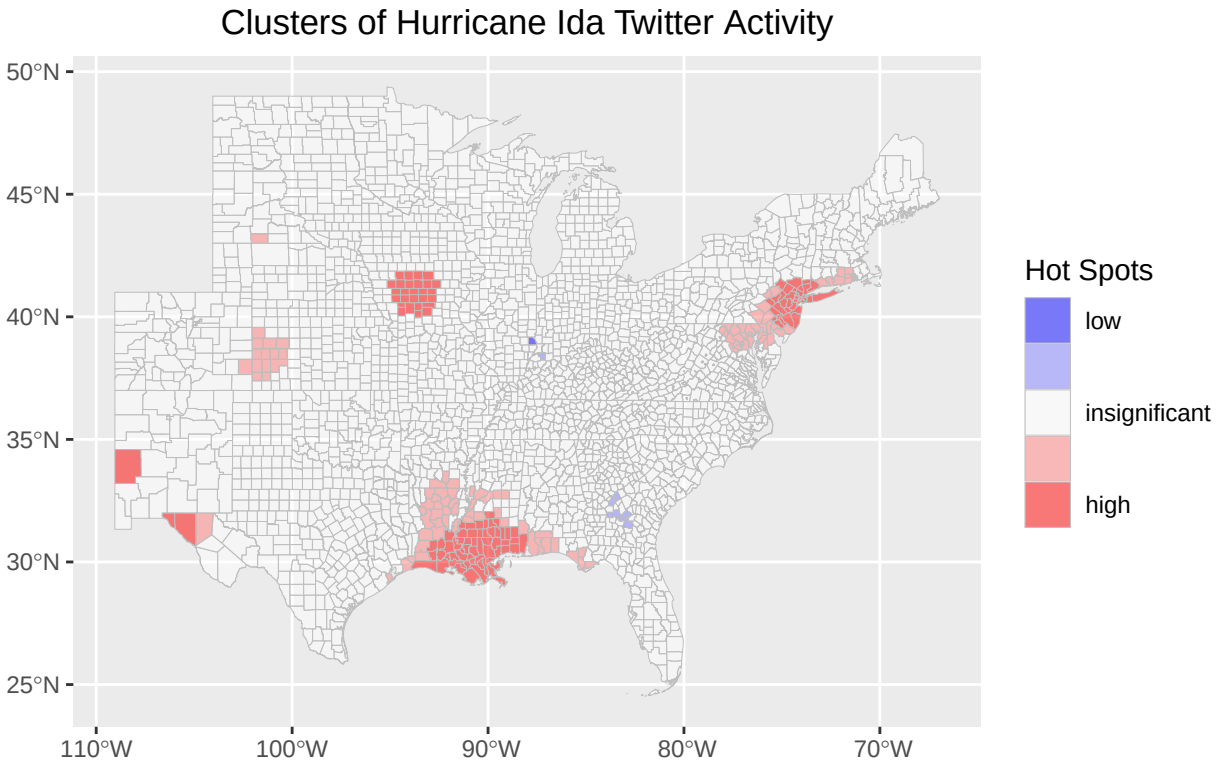
Map Hotpots

classify G scores by significance values typical of Z-scores where 1.15 is at the 0.125 confidence level, and 1.95 is at the 0.05 confidence level for two tailed z-scores based on Getis and Ord (1995) Doi: 10.1111/j.1538-4632.1992.tb00261.x to find other critical values, use the qnorm() function as shown here: https://methodenlehre.github.io/SGSCLM-R-course/statistical-distributions.html Getis Ord also suggest applying a Bonferroni correction

breaks and colors from http://michaelminn.net/tutorials/r-point-analysis/ based on 1.96 as the 95% confidence interval for z-scores if your results don't have values in each of the 5 categories, you may need to change the values & labels accordingly.

Unplanned Deviation: I changed the line color from white to gray to help see county boundaries.

```r
# classify by significance levels
siglevel <- c(1.15, 1.95)
counties <- counties %>%
  mutate(sig = cut(locG, c(
    min(counties$locG),
    siglevel[2] * -1,
    siglevel[1] * -1,
    siglevel[1],
    siglevel[2],
    max(counties$locG)
  )))
rm(siglevel)

# map results!
ggplot() +
  geom_sf(data = counties, aes(fill = sig), color = "gray", lwd = 0.1) +
  scale_fill_manual(
    values = c("#0000FF80", "#8080FF80", "#FFFFFF80", "#FF808080", "#FF000080"),
    labels = c("low", "", "insignificant", "", "high"),
    aesthetics = "fill"
  ) +
  guides(fill = guide_legend(title = "Hot Spots")) +
  labs(title = "Clusters of Hurricane Ida Twitter Activity") +
  theme(
    plot.title = element_text(hjust = 0.5),
    axis.title.x = element_blank(),
    axis.title.y = element_blank()
  )
```



Discussion

Temporal and Frequency Analysis

The figure showcasing the temporal trends of the Hurricane Ida X/Twitter data highlights peaks of social media activity when the hurricane touched Louisiana and Northeast. The word frequency analysis showcased a high frequency of words such as "flood," "recovery," and "Louisiana." Both of these figures help to visualize trends of X/Twitter activity and are especially helpful in getting a general understanding of the data.

Word Association Analysis

The word association analysis showcases two major clusters of topics: one surrounding power outages and one focused more on flooding. Other topics that appear include tweets about delayed reporting and climate change. This figure gives more context on the most common topics within the data set, and it can also showcase other topics of interest that aren't directly connected to the weather reports of the specific Hurricane.

Tweet Location and Activity Analysis

The figures presenting the tweet locations showcase higher activity along the coasts of Louisiana and Florida. Additionally, there is also very high activity along the Northeast/the Mid-Atlantic. After the normalization using population data from the Census, the tweet activity figure showcases greater activity (normalized by

country population) in Louisiana. The results of these two figures help us to see how population information affects our analysis of Hurricane Ida X/Twitter data.

Tweet Cluster Analysis

The Tweet Cluster analysis showcases the highest/largest hot spots within the Southern coast of Louisiana and Mississippi and along the Northeast/Mid-Atlantic coast. However, there are also more sparse/smaller hot spots in Texas and the Midwest. These unexpected hot spots could be observed in future reproductions using this data set.

Conclusion

Our adjustments to the computational environment were successful, as the code worked with the Hurricane Ida with only few adjustments when using the "remote" and "rtweet" packages. The new normalization showcased results that were weighted using Census population data. This resulted in helpful figures that showcased X/Twitter activity, locations, and clusters. The inclusion of the discussion and conclusion sections help to explain how the results pertain to Hurricane Ida data.

The replication of the Holler (2021) study was successfully as were able to replicate all of the original figures. Though the X/Twitter API has more barriers to tweet information now, this replication indicates the general procedure of this modified Holler (2021) study can be used to examine X/Twitter trends of other events/natural disasters.

Integrity Statement

Include an integrity statement - The authors of this preregistration state that they completed this preregistration to the best of their knowledge and that no other preregistration exists pertaining to the same hypotheses and research. If a prior registration *does* exist, explain the rationale for revising the registration here.

Acknowledgements

This report is based upon the template for Reproducible and Replicable Research in Human-Environment and Geographical Sciences, DOI:%5B10.17605/OSF.IO/W29MQ](https://doi.org/10.17605/OSF.IO/W29MQ)

References

This study is a replication of this study conducted by Professor Joseph Holler