zyas19 / INSC-590.001-Data-Software-Preservation-Project Private

Branch: master INSC-590.001-Data-Software-Preservation-Project / Reproducible ReadMe.md Find file Copy path

zyas19 Update Reproducible ReadMe.md 91c614a a day ago

1 contributor

133 lines (124 sloc) 12 KB Raw Blame History

Reproducibility ReadMe for INSC 590.001 Data Software Preservation Project

This file describes how to reproduce all analyses from the ANTH 511 Twitter Analysis project conducted by Yasmin Stoss, Spring 2019

Version and System Information

platform x86_64-w64-mingw32
arch x86_64
os mingw32
system x86_64, mingw32
status
major 3
minor 5.2
year 2018
month 12
day 20
svn rev 75870
language R
version.string R version 3.5.2 (2018-12-20) nickname Eggshell Igloo

#Software (R Studio) RStudio Team (2016). RStudio: Integrated Development for R. RStudio, Inc., Boston, MA URL http://www.rstudio.com/.

Version: 1.1.463

R Packages/Libraries

Plotly, lubridate, tidytext, tidyverse, lmtest, RColorBrewer, wordcloud

R Code to install and initialize packages

- install.packages(c("lubridate", "tidytext", "tidyverse", "lmtest", "RColorBrewer", wordcloud"))
- install.packages("plotly", repos="http://cran.rstudio.com/ 229", dependencies=TRUE)
- packages <- c("lubridate", "tidytext", "tidyverse", "lmtest", "RColorBrewer", wordcloud", "plotly")
- lapply(packages, library, character.only = TRUE)

Steps and R Code for Project Analyses

1. Download "Clemson.csv" onto your local hard drive
2. Open R Studio
3. Read in csv file

- tweets <-read.csv("Clemson.csv")

4. Format dates in the " " column of the spreadsheet. This enables analyses using dates and timeframes.

- tweets_dated <-tweets %>% mutate(Date = mdy_hm(publish_date))

5. Rename the "Content" column to "Tweet". This makes things less confusing down the line

- colnames(tweets_dated)[colnames(tweets_dated)=="content"] <-"Tweet"

6. Clean data by removing URLs and other unwanted characters

- replace_reg <-"https://t.co/[A-Za-z\\d]+|http://[A-Za-z\\d]+|&amp;|&lt;|&gt;|https"
- unnest_reg <-"([^A-Za-z_\d#@']|'(?![A-Za-z_\d#@]))"
- clean_tweets<-tweets_dated%>%mutate(Tweet=str_replace_all(Tweet,replace_reg,""))

7. Sort out all tweets containing the keyword(s):

- keywords<-c("vacc", "Vacc", "vax", "Vax", "clean coal", "Clean Coal", "paris climate accord", "Paris Climate Accord", "ebola", "Ebola, "zika", "Zika, "dakota access pipeline", "Dakota Access Pipeline","pasteuriz", "Pasteuriz", "frack", "Frack")
- tweets_sci_words<-clean_tweets%>%filter(str_detect(Tweet,paste(keywords, collapse="|")))

8. Count instances of the word Vaccine and all variations of (result=821)

- sum(str_count(tweets_sci_words$Tweet, "vacc"))
- sum(str_count(tweets_sci_words$Tweet, "Vacc"))
- sum(str_count(tweets_sci_words$Tweet, "Vax"))
- sum(str_count(tweets_sci_words$Tweet, "vax"))

9. Count instances of the word Ebola (Result = 556)

- sum(str_count(tweets_sci_words$Tweet, "ebola"))
- sum(str_count(tweets_sci_words$Tweet, "Ebola"))

10. Count instances of the word Zika (result = 2125)

- sum(str_count(tweets_sci_words$Tweet, "ebola"))
- sum(str_count(tweets_sci_words$Tweet, "Ebola"))

11. Count instances of the word Paris Climate Accord (Result= 31)

- sum(str_count(tweets_sci_words$Tweet, "Paris climate accord"))
- sum(str_count(tweets_sci_words$Tweet, "paris climate accord"))
- sum(str_count(tweets_sci_words$Tweet, "Paris Climate Accord"))

12. Count instances of the word Clean Coal (Result = 27)

- sum(str_count(tweets_sci_words$Tweet, "Clean coal"))
- sum(str_count(tweets_sci_words$Tweet, "clean coal"))
- sum(str_count(tweets_sci_words$Tweet, "Clean Coal"))

13. Count instances of the word Dakota Access Pipeline (Result = 50)

- sum(str_count(tweets_sci_words$Tweet, "dakota access pipeline"))
- sum(str_count(tweets_sci_words$Tweet, "dakota access pipeline"))
- sum(str_count(tweets_sci_words$Tweet, "Dakota Access Pipeline"))

14. Count instances of the word Pasteurization (Result= 0)

- sum(str_count(tweets_sci_words$Tweet, "Pasteuriz"))
- sum(str_count(tweets_sci_words$Tweet, "pasteuriz"))

15. Count instances of the word Fracking (Result= 174)

- sum(str_count(tweets_sci_words$Tweet, "Frack"))
- sum(str_count(tweets_sci_words$Tweet, "frack"))

16. Plot instance of each keyword in a donut plot

- Keywords_count <- read.csv("Keywords_count.csv")
- Keywords_count %>% group_by(keyword) %>% plot_ly(labels=~keyword, values=~count) %>% add_pie(hole=0.3)

17. Create a data frame with the tweets separated out into single words

- sci_words<-tweets_sci_words%>%unnest_tokens(word,Tweet,token="regex",pattern=unnest_reg)%>%filter(!word %in% stop_words$word,str_detect(word, "[a-z]"))

18. Filter out the keywords to prevent confounding. The sentiment word lists contain some of the keywords like vaccine and counting these words as part of the sentiment analysis will confound results.

- keywords2<-c("vacc", "Vacc", "vax", "Vax", "clean", "coal", "Clean", "Coal", "paris", "climate", "accord", "Paris", "Climate", "Accord", "ebola", "Ebola", "zika", "Zika", "dakota", "access", "pipeline", "Dakota", "Access", "Pipeline","pasteuriz", "Pasteuriz", "frack", "Frack")
- sci_words2<-sci_words%>%filter(!str_detect(word,paste(keywords2, collapse="|")))

19. Create a data frame of the nrc sentiments and then join with our tweets data frame (sci_words2) so that we see a sentiment assigned to each word

- nrc<-get_sentiments("nrc")
- sci_tweet_sentiment <-sci_words2 %>% inner_join(nrc) %>% count(index = Date, sentiment) %>% spread(sentiment, n, fill = 0)

20. Create a word cloud for the top 20 words for the entire data set, the positive words and the negative words

- pal <- brewer.pal(9,"RdBu")
- wordcloud(words = count_sci_word$word, freq = count_sci_word$n, min.freq = 1, max.words=20, color=pal)
- wordcloud(words = positive_all$word, freq = positive_all$n, min.freq = 1, max.words=20, color=pal)
- wordcloud(words = Negative_all$word, freq = Negative_all$n, min.freq = 1, max.words=20, color=pal)

21. Create a line plot of the cumulative sum of the negative-positive sentiment words

- sci_tweet_sentiment %>% ggplot(aes(x=as.Date(index), y=cumsum(negative-positive))) + geom_line()

22. Create a line plot of the cumulative sum of the fear-trust sentiment words

- sci_tweet_sentiment %>% ggplot(aes(x=as.Date(index), y=cumsum(fear-trust))) + geom_line()

23. Normalize the data by overall tweet counts per day. This is done because the number of positive tweets may appear to increase over time when in actuality the number of tweets is increasing overall. We divide by the number of tweets daily to remove this confounding factor.

- daily_sci <-sci_words2 %>% mutate(tweetDay= floor_date(Date, "day"))

- daily_sci_sentiment <-daily_sci %>% inner_join(nrc) %>% count(index = tweetDay, sentiment) %>% spread(sentiment, n, fill = 0)
- colnames(daily_sci_sentiment)[colnames(daily_sci_sentiment)=="index"] <-"tweetDay"
- wordsPerDay <-count(daily_sci, tweetDay)
- SentimentsPerDay <-inner_join(daily_sci_sentiment, wordsPerDay)

24. Create a line plot of the cumulative sum of the negative-positive sentiment words with normalized data. Include lines for the 2016 election day and the date of Trump's running announcement to see if changes in the plot are related to these events.

- ggplot(data = SentimentsPerDay) + geom_area(aes(x=tweetDay, y=cumsum(((negative-positive)/n))),color="#00AFBB", size=2, fill="#00AFBB")+geom_vline(xintercept = as.POSIXct(as.Date(c("2016-11-08"))), linetype="dotted",color = "#daa520", size=2)+geom_vline(xintercept = as.POSIXct(as.Date(c("2016-06-16"))), linetype="dotted",color = "#709963", size=2)

25. Create a line plot of the cumulative sum of the fear-trust sentiment words with normalized data. Include lines for the 2016 election day and the date of Trump's running announcement to see if changes in the plot are related to these events.

- ggplot(data = SentimentsPerDay) + geom_area(aes(x=tweetDay, y=cumsum(((fear-trust)/n))),color="#00AFBB", size=2, fill="#00AFBB")+geom_vline(xintercept = as.POSIXct(as.Date(c("2016-11-08"))), linetype="dotted",color = "#daa520", size=2)+geom_vline(xintercept = as.POSIXct(as.Date(c("2016-06-16"))), linetype="dotted",color = "#709963", size=2)

26. Determine the top 10 tweeters by count of tweets

- count(sci_words2, Twitter_Name, sort=TRUE)

27. Do a ribbon plot of the top 10 tweeters

- sci_words2 %>% mutate(Twitter_Name = fct_lump(Twitter_Name, 10)) %>% count(month = round_date(Date, "month"), Twitter_Name) %>% complete(month, Twitter_Name, fill = list(n = 0)) %>% mutate(Twitter_Name = reorder(Twitter_Name, -n, sum)) %>% group_by(month) %>%mutate(percent = n /sum(n), maximum = cumsum(percent), minimum = lag(maximum, 1, 0)) %>% ggplot(aes(month, ymin = minimum, ymax = maximum, fill = Twitter_Name)) + geom_ribbon()+theme(legend.position="bottom", legend.title = element_blank())+geom_vline(xintercept = as.POSIXct(as.Date(c("2016-11-08"))), linetype="dotted",color = "black", size=1.5)

28. Filter out the top 10 tweeters, construct a list with those handles, and do a ribbon plot showing the number of tweets over time. Include line for the 2016 election day to see if changes in the plot are related to this event.

- top10_tweeters<c("NEWSPEAKDAILY","TODAYPITTSBURGH","SCREAMYMONKEY","SEATTLE_POST","ROOMOFRUMOR","MILWAUKEEVOICE","KANSASDAILYNEWS","WASHINGTONLINE","DAILYSANJOSE","DAILYSANFRAN","TODAYBOSTONMA")
- Top10_sci_words<-tweets_sci_words%>%filter(str_detect(Twitter_Name,paste(top10_tweeters, collapse="|")))
- Top10_sci_words %>% mutate(Twitter_Name = fct_lump(Twitter_Name, 10)) %>% count(month = round_date(Date, "month"), Twitter_Name) %>% complete(month, Twitter_Name, fill = list(n = 0)) %>% mutate(Twitter_Name = reorder(Twitter_Name, -n, sum)) %>% group_by(month) %>%mutate(percent = n /sum(n), maximum = cumsum(percent), minimum = lag(maximum, 1, 0)) %>% ggplot(aes(month, ymin = minimum, ymax = maximum, fill = Twitter_Name)) + geom_ribbon()+theme(legend.position="bottom", legend.title = element_blank())+geom_vline(xintercept = as.POSIXct(as.Date(c("2016-11-08"))), linetype="dotted",color = "black", size=1.5)

29. Determine the top 3 tweeting accounts and create a list containing those twitter handles.

- count(sci_words2, Twitter_Name, sort=TRUE)
- Top3_tweeters<-c("NEWSPEAKDAILY", "SCREAMYMONKEY","TODAYPITTSBURGH")

30. Do a Granger Causality test between the top 3 tweeters and the rest of the data set

- all_timeSeries<-sci_words2%>%group_by(week=round_date(Date,"week"))%>%summarize(tweets=n(), account=sum(str_detect(Twitter_Name, paste(top10_tweeters, collapse="|"))), percent=account/tweets)
- newspeakdaily_timeSeries<-sci_words2%>%group_by(week=round_date(Date,"week"))%>%summarize(tweets=n(), account=sum(str_detect(Twitter_Name, "NEWSPEAKDAILY")), percent=account/tweets)
- screamymonkey_timeSeries<-sci_words2%>%group_by(week=round_date(Date,"week"))%>%summarize(tweets=n(), account=sum(str_detect(Twitter_Name, "SCREAMYMONKEY")), percent=account/tweets)
- todaypittsburgh_timeSeries<-sci_words2%>%group_by(week=round_date(Date,"week"))%>%summarize(tweets=n(), account=sum(str_detect(Twitter_Name, "TODAYPITTSBURGH")), percent=account/tweets)
- grangertest(newspeakdaily_timeSeries$percent,all_timeSeries$percent, order=1)
- grangertest(todaypittsburgh_timeSeries$percent,all_timeSeries$percent, order=1)
- grangertest(screamymonkey_timeSeries$percent,all_timeSeries$percent, order=1)

31. Visualize the time series for the granger causality tests with significant p values (only SCREAMYMONKEY was significant)

- ggplot()+geom_area(data=all_timeSeries, aes(x=as.Date(week), y=percent), color="#00AFBB", size=2, fill = "#00AFBB")+geom_area(data=screamymonkey_timeSeries, aes(x=as.Date(week), y=percent), color="#FC4E07", size=2, fill="#FC4E07")+xlab("Date")+ylab("% of tweets")+geom_vline(aes(xintercept=as.numeric(as.Date("2016-11-13"))), linetype=4, color="black")