Paper 3081-2015

Tweet-o-matic: An Automated Approach to Batch Processing of Tweets

Isabel Litton, California Polytechnic State University, San Luis Obispo, CA

Rebecca Ottesen, California Polytechnic State University, San Luis Obispo, CA

ABSTRACT

Currently, there are several methods to read JSON formatted files into SAS® which depend on the version of SAS and which products are licensed. These methods include user-defined macros, visual analytics, PROC GROOVY and more. The user defined macro %GrabTweet, in particular, provides a simple way to directly read JSON formatted tweets into Base SAS 9.3. The main limitation of %GrabTweet is that it requires the user to repeatedly run the macro in order to download large amounts of data over time. Manually downloading tweets while adhering to the Twitter rate limits may cause missing observations and is time consuming overall. Imagine having to sit by your computer the entire day to continuously grab data every 15 minutes, just to download a complete data set of tweets for a popular event. Fortunately, the %GrabTweet macro can be modified to automate of the retrieval of Twitter data based on the rate that the tweets are coming in. This paper describes the application of the %GrabTweet macro combined with batch processing to download tweets without manual intervention. Users can specify the phrase parameters they desire, run the batch processing macro, leave their computer to automatically download tweets overnight and return to a complete data set of recent Twitter activity. The batch processing implements an automated retrieval of tweets through an algorithm that assesses the rate of tweets for a specified topic in order to make downloading large amounts simpler and effortless for the user.

INTRODUCTION

Since its launch in 2006, Twitter has rapidly expanded into a social media giant with over 250 million active users. Despite the limited 140 character messages, this microblogging service has established its role as a vital marketing and information resource. Each miniature post allows business to connect with their customers, create brand awareness, and market new products. In addition Twitter users can socialize and communicate about a variety of subjects and events. Methods to obtain tweets have become an important tool, now more than ever, as businesses rush to capitalize on the information stored in each of the 500 million tweets that are sent per day. This paper explores a macro that allows Base SAS 9.3 users the ability to access Twitter's API without having to turn to other methods, which may be more restricted or costly.

%GRABTWEET MACRO

UPDATES

Originally, the %GrabTweet macro ran as two nested macros for different PROC HTTP URLs. Both URLS incorporated the search term, result type, and count parameter, but one included an additional parameter (max_id), in order to download older tweets. The updated %GrabTweet macro doesn't rely on nested macros, but instead applies conditional logic to employ three distinct PROC HTTP calls. These separate URL arguments are necessary to download: 1) a base set of recent tweets, 2) more recent tweets that occur after the first base tweets, and 3) tweets older than base tweets. The macro variables "loopvar" and "olderid" are used as switches to designate which URL should be executed. Examples of the three HTTP calls are shown below.

```
URL =
"https://api.twitter.com/1.1/search/tweets.json?q=&search_term&type&num_tweet"
```

The first URL, shown above, downloads the 100 most recent tweets (as defined by Twitter) as a base dataset. The string only contains the search term, result type, and count (labeled as num_tweet) parameter because it is first call to download data. The PROC HTTP containing this URL is only executed when "loopvar" equals "N" because the looping to download subsequent activity is not yet required. Since this is the first data pull, we're using this URL to create the base set of tweets.

```
URL =
"https://api.twitter.com/1.1/search/tweets.json?q=&search_term&recent_tweet&type&
num_tweet"
```

The second URL, shown above, downloads tweets more recent than the tweet id that was specified in the since_id parameter, which is labeled as &recent_tweet. The string contains all the parameters from the first URL, along with the since_id parameter. This since_id URL parameter returns tweets more recent (i.e. tweets with larger ids) than the specified id. The id of the first tweet downloaded from the most recent URL call is stored in a macro variable through CALL SYMPUT to be set later in the since_id parameter. This allows tweets with ids after the max id from the previous download to be captured. The PROC HTTP containing this URL is only executed when “loopvar” equals “Y” because this URL needs to repeat in order to continuously download newer tweets.

```
URL =
"https://api.twitter.com/1.1/search/tweets.json?q=&search_term&id&type&num_tweet"
```

The third and final URL, shown above, downloads tweets older than the tweet id specified in the max_id parameter, labeled as the macro variable &id. Similar to the second URL, this string contains all the parameters from the first URL, as well as the max_id parameter. The max_id URL parameter returns older tweets (tweets with smaller ids) than the specified id. The id of the last tweet downloaded from the most recent URL call is stored in a macro variable through CALL SYMPUT as well. This PROC HTTP will only be executed when “olderid” equals “Y”. This URL needs to be repeated as well, in order to download as many older tweets as allowed.

Other %GrabTweet macro updates include additional output variables including latitude and longitude coordinates, and the date of tweet as a SAS date. Twitter has a geotagging feature called “Twitter with Location”, which allows the latitude and longitude coordinates to be displayed alongside other tweet attributes. Users must give their permission, however, by turning the location feature “on” in order for the latitude and longitude coordinates to be returned. Therefore the latitude and longitude data that is downloaded with %GrabTweet may often be incomplete.

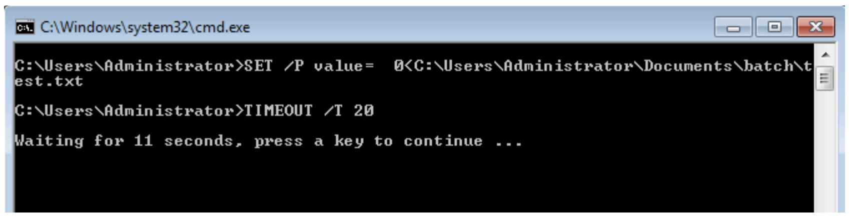
%TWEETOMATIC MACRO

BATCH PROCESSING

A key aspect to continuously download tweets is through the combination of batch processing with the %GrabTweet macro. The CALL SYSTEM statement in SAS describes the path of the batch file to be executed. The NOXWAIT system option is specified so that users do not have to type “exit” to close the command prompt window. It should be noted that the CALL SYSTEM path and NOXWAIT option are specific to the Windows operating system on a user’s machine. There are three instances in which the CALL SYSTEM routine is executed in order to submit a batch script.

```
SET /P value= <C:\Users\Administrator\Documents\batch\waittime.txt
TIMEOUT /T %value%
```

The first occurrence of batch processing is right before a DO UNTIL loop to download older tweets. The .bat file executed contains the script shown above. The first line initializes the variable “value” with the number contained in the waittime.txt file. Test.txt contains the number of seconds the command processor will delay the current processing of the macro. In this case, test.txt contains the value 20 in order to pause the SAS program for 20 seconds before the next URL call. Sometimes, if the URL calls to access the API are too close zero unwanted observations will be returned. A 20 second pause will allow sufficient spacing between PROC HTTP runs. The value in the test.txt file should be an integer and can only range from 0 to 99999. TIMEOUT /T is the actual command to pause the command processor. %value% is the batch command to use the variable declared in the first line. In summary, the batch script creates a variable, passes the amount of seconds to wait from a text file, and delays the macro processing. The command prompt will automatically close when the script is done running and return to process the subsequent code in the SAS program. Figure 1 displays the command prompt window that appears when this batch script is called from a Windows computer.



```
C:\Users\Administrator>SET /P value=  0<C:\Users\Administrator\Documents\batch\t
est.txt

C:\Users\Administrator>TIMEOUT /T 20

Waiting for 11 seconds, press a key to continue ...
```

Figure 1. Command Prompt Window To Delay Current Macro Processing

The second occurrence of batch processing is placed within a DO UNTIL loop to download the most recent tweets. The purpose and script of this .bat file is exactly the same as the .bat described above, however, the file containing the wait time value is different. Therefore, only the file name should be changed from the batch script shown above. The avgwaittime.txt file being read contains just a single integer, which is the number of seconds the program will timeout for.

```
DATA _null_;
             SET test_mean;
             avgrate = CEIL(avgtime);
             IF avgrate > 99999 THEN avgrate = 99999;
             FILE 'C:\Users\Administrator\Documents\batch\batch_test';
             PUT avgrate;
RUN;
```

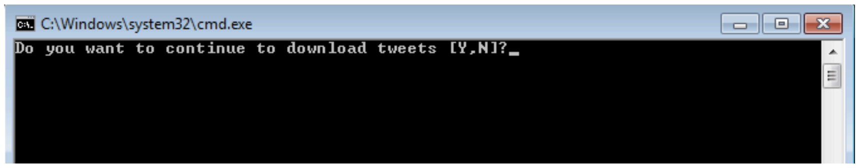
PROC MEANS with only the MEAN option was used to obtain the average time to wait. Another variable (avgrate) is created with the CEIL function to round the average time calculated from PROC MEANS to return an integer value. This additional variable is necessary because the TIMEOUT function can only be passed integer values from 0 to 99999. The average time between tweets is only calculated for the most recently downloaded tweets from the last PROC HTTP call, rather than the entire dataset. Users may calculate another statistics for the rate of the tweets, but should remember that: 1) the final value must be rounded, and 2) the text file should contain only that value and nothing else. Batch processing may be used to execute a series of commands without manual intervention, however, the last batch script called in %Tweetomatic prompts the user for input. Figure 2 displays an example of what the avgwaittime.txt file contains in order to properly illustrate what occurs when this batch script is run.

```
%IF (&i > 10 AND &recentloopcount < 5) %THEN %DO;
OPTIONS NOXWAIT;
DATA _NULL_;
      CALL SYSTEM('call C:\Users\Administrator\Documents\batch\prompt.bat');
RUN;
%END;
```

This final occurrence of batch processing, shown above, is found within a conditional statement at the very end of the DO UNTIL loop to download as many recent tweets as possible. The condition checks whether at least ten iterations of the loop have passed and the number of tweets downloaded for each pass are less than five. The prompt.bat prompts the user for a yes or no answer in order to determine whether to continue downloading more tweets or to exit the loop and terminate current macro processing.

```
@ECHO OFF
:start
choice /m "Do you want to continue to download tweets" /t:10 /d:Y
if ERRORLEVEL 2 goto :NO
if ERRORLEVEL 1 goto :YES
goto :start
:YES
EXIT
:NO
copy nul C:\Users\Administrator\Documents\batch\exit.txt
```

The ECHO OFF allows the user to display a message that is several lines long without showing the other commands. The next line creates the label “start” which is jumped to by the goto command. The choice command prompts the user to select either “Y” for yes or “N” for no to the question “Do you want to continue to download tweets”. The /m parameter indicates the text message to be displayed in the prompt window. The /c parameter specifies choices for user input. If the /c parameter is missing as shown above, the default choices are “Y” or “N”. The /t parameter is the timeout command used in the other batch scripts, and specifies the amount of seconds to pause before the default choice is made. The /d parameter specifies the default option after the seconds in the timeout command run out. The ERRORLEVEL will return a 1 if the user inputs “Y” and a 2 if the user inputs “N”. If the user chooses “Y” or does not answer within 10 seconds, the batch script closes and returns to download more recent tweets. If the user chooses “N”, a null file called “exit.txt” will be made in the specified pathname. This null file becomes necessary to exit the macro entirely and return the downloaded tweets. In order to terminate the current processing of %Tweetomatic without the DO UNTIL condition fulfilled, SAS checks the existence of the null file. If the null file exists, SAS will delete the file so the user does not have to be concerned about its existence for future implementations and exits the macro.



```
Do you want to continue to download tweets [Y,N]?_
```

Figure 2. Command Prompt Window To Prompt User Input

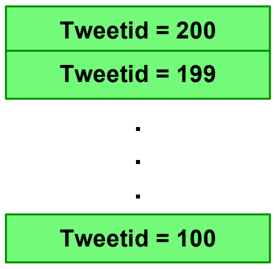
Figure 2 illustrates what the user will see from the command prompt window if the loop is iterated at least ten times and the number of recent tweets to be downloaded is less than five. This batch script is crucial to automate the retrieval of tweets because it allows the program to continuously download tweets while giving the user the option to return already processed observations if there aren't many tweets to be downloaded. On the other hand, if there's a trending topic generating a lot of tweets, the batch script will allow the user to leave the program running overnight. Although the prompt would still appear after ten iterations, without user input after ten seconds, the command prompt will default to continuing to download more data.

EXAMPLE CALL

This example demonstrates what the user should expect to see as well as what occurs behind the scenes as the macro is processed. An example macro call to download 1,000 tweets for #SASGF15 is shown below:

```
%Tweetomatic(%23sasgf15, 1000);
```

The following figures are used to illustrate the logic of how tweets are downloaded as well as actual command prompt window and SAS output (shown to the left) as well as the corresponding portions code (shown to the right).



Tweetomatic Code:

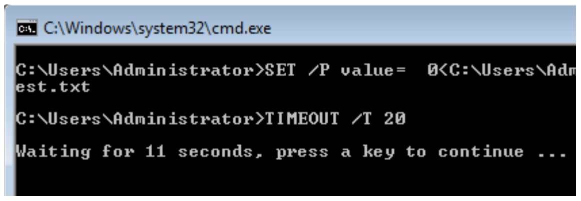
```
%LET olderid = NA;
%LET loopvar = N;
%GRABTWEET(&search_term, 100);
```

GrabTweet URL call:

```
URL =
"https://api.twitter.com/1.1/search/tweets.json?q=&%2
3sasgf15&result_type=recent&count=100"
```

Figure 3. Demonstration of Base Data Set

In the first step, the macro will download 100 tweets to create the base data as shown by Figure 3.



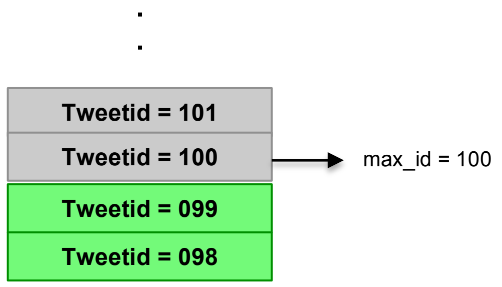
```
C:\Windows\system32\cmd.exe

C:\Users\Administrator>SET /P value=  0<C:\Users\Adm
est.txt

C:\Users\Administrator>TIMEOUT /T 20

Waiting for 11 seconds, press a key to continue ...
```

Tweetomatic Code:

```
OPTIONS NOXWAIT;
DATA _NULL_;
CALL SYSTEM('call C:\... \batch.bat');
RUN;
```

Figure 4. Command Prompt Window To Pause Macro

A batch script is run to pause before the next URL call to download older tweets and the command prompt window pops up to display a countdown time as shown by Figure 4.



Tweetomatic Code:

```
%LET loopvar = NA;
%LET olderid = Y;
%GRABTWEET(&search_term, 100);
```

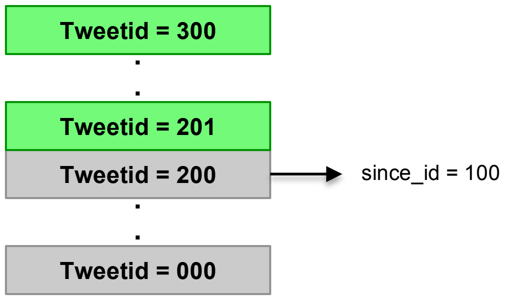
GrabTweet URL call:

```
URL =
"https://api.twitter.com/1.1/search/tweets.json?q=&%2
3sasgf15&max_id=100&result_type=recent&count=100"
```

Figure 5. Demonstration of Downloading Older Tweets

Next the id of the last processed tweet (Tweetid = 100) is stored in a macro variable called "old_id" in order to be set as the max_id URL parameter in order to return tweets older than this exact tweet as shown in Figure 5. Notice that in

Figure 5 and all subsequent figures, the color green will be used to indicate tweets currently being downloaded while the color grey will designate tweets already processed and read into SAS.

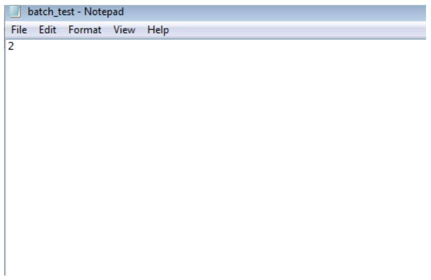


Tweetomatic Code:
```
%LET olderid = NA;
%LET loopvar = N;
%GRABTWEET(&search_term, 100);
```

GrabTweet URL call:
```
URL =
"https://api.twitter.com/1.1/search/tweets.json?q=&%2
3sasgf15&since_id=200&result_type=recent&count=100"
```

Figure 6. Demonstration of Downloading Recent Tweets

When all possible older tweets are downloaded, the id of the first tweet read (Tweetid = 200) is stored in a macro variable called "recent_id" in order to be set as the since_id URL parameter to return more recent tweets. In this case the macro would start searching at Tweetid = 201 on the next loop.

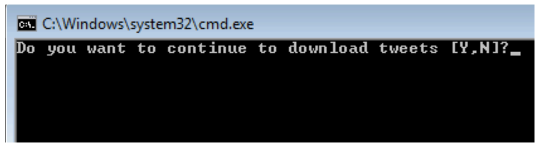


```
2
```

Tweetomatic Code:
```
PROC MEANS DATA = main_conversion MEAN;
               VAR time_diff;
               OUTPUT OUT = test_mean MEAN = avgtime;
RUN;
PROC EXPORT DATA = test_mean_int
               OUTFILE = 'C:\...\batch_test'
               DBMS = tab
               REPLACE;
RUN;
```

Figure 7. Example of Middle Man File Containing Timeout Value

A series of steps including PROC MEANS and the FILE statement are needed to calculate the amount of time to delay macro processing and write the value to a file. These steps create a middle man file to be used in a batch script to delay the next call to the URL. Next, another command prompt window will appear exactly as shown previously in Figure 4, that shows the system waiting for the number of seconds as specified by the middleman file. This process will be repeated until the number of observations downloaded is greater than the target amount specified by the user or if the user chooses to manually terminate the program.

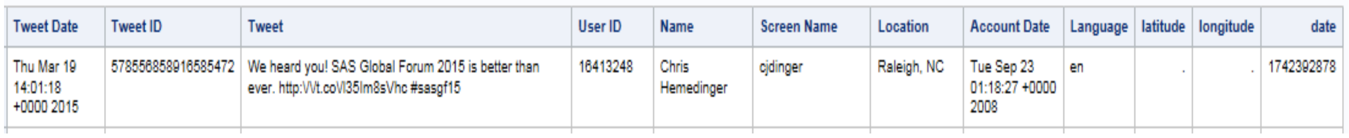


```
Do you want to continue to download tweets [Y,N]?_
```

Tweetomatic Code:
```
%IF (&i > 10 AND &recentloopcount < 5) %THEN %DO;
OPTIONS NOXWAIT;
DATA _NULL_;
CALL SYSTEM('call
C:\Users\Administrator\Documents\batch\prompt.bat');
RUN;
```

Figure 8. Command Prompt Window to Manually Stop Program

As described earlier, when the loop has iterated a certain number of times and there are few recent tweets to download, a batch script will be executed to display the command prompt window given in Figure 9. When the user wants to stop downloading tweets, the CALL SYSTEM prompt in the null file is written to the specified location and will be used to stop the macro processing.



Tweet Date	Tweet ID	Tweet	User ID	Name	Screen Name	Location	Account Date	Language	latitude	longitude	date
Thu Mar 19 14:01:18 +0000 2015	578556858916585472	We heard you! SAS Global Forum 2015 is better than ever. http:\/\/t.co\/I35lm8sVhc #sasgf15	16413248	Chris Hemedinger	cjdinger	Raleigh, NC	Tue Sep 23 01:18:27 +0000 2008	en	.	.	1742392878

Figure 9. Partial Output of %Tweetomatic(%23sasgf15, 1000);

Figure 9 illustrates the final end product, a SAS dataset containing the tweet, date, latitude and longitude coordinates, and more.

CONCLUSION

While downloading Twitter data is all the rage these days, it can be frustrating to sit in front of a computer to grab data manually in chunks. Having the option to have the program download data at a rate specific to the tweet rate is quite liberating and leads to complete data sets for analysis. The automated retrieval of Twitter data grants Base SAS 9.3 users straightforward access to the wealth of Twitter's API.

There are three issues to address with the use of the Tweetomatic macro: 1) the number of tweets returned will not exactly match the target amount, 2) the macro must be terminated manually if the user does not want to wait until the number of observations is greater than the target, 3) batch commands discussed here are specific to a Windows operating system. The number of tweets downloaded will not exactly match the amount specified by the user because of the inclusion of original tweets of retweets and the varying number of tweets accessible from the API. The user may want 1,000 tweets but there may be only 200 tweets available. Due to the fluid nature of tweets available, the macro provides the user an option to stop downloading data and return the tweets already processed. Otherwise the macro may be running for undesirable amount of time waiting for tweets that may never be posted. Batch processing was essential to execute Windows commands to force the SAS program to wait before processing the following SAS code.

Future steps for the macro include work toward spell checking text and parsing apart phrase that are concatenated together. In addition, reverse geocoding the latitude and longitude coordinates to return a location and scoring keywords based on sentiment will be explored.

REFERENCES

- "CHOICE.exe" July 19th, 2014. Available at http://ss64.com/nt/choice.html.
- "FOR /F." July 19th, 2014. Available at http://ss64.com/nt/for_f.html.
- "Geo Developer Guidelines." Available at https://dev.twitter.com/terms/geo-developer-guidelines.
- "GET search/tweets". March 7th, 2013. Available at https://dev.twitter.com/docs/api/1.1/get/search/tweets.
- "Using the Twitter Search API." Available at https://dev.twitter.com/docs/using-search.
- "Working with Timelines". September 7th, 2012. Available at https://dev.twitter.com/docs/working-with-timelines.

ACKNOWLEDGMENTS

The authors would like to thank the Cal Poly Statistics Department for the opportunity to conduct this summer research project.

RECOMMENDED READING

- Twitter Developers Documentation for Rest API v1.1
- %GrabTweet: A SAS Macro to Read JSON Formatted Tweets

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:

Name:	Isabel Litton
Enterprise:	California Polytechnic State University
E-mail:	isabel.litton@gmail.com
Web:	https://github.com/ilitton

Name:	Rebecca Ottesen
Address:	1 Grand Avenue, Statistics Dept
City, State ZIP:	San Luis Obispo, CA, 93407-0405
Work Phone:	805-756-2709
Fax:	805-756-2700
E-mail:	rottesen@calpoly.edu
Web:	http://statweb.calpoly.edu/rottesen/

APPENDIX

```
/*************************************************************************************
MACRO: Tweetomatic
PARAMETERS:
       Search_Term = Word or phrase you want tweets to contain. Represent hashtags (#)
       with %23.
       Target = Number of tweets you want returned.
REQUIREMENTS:
       Bearer token from Twitter Developers' Page to authorize requests to Twitter API
       (Not provided)
       %GrabTweet (Provided)
OUTPUT VARIABLES:
       created_at1 = Date of tweet
       tweet_id = Tweet ID
       text = Tweet
       user_id = User ID
       name = Name of account
       screen_name = Screen Name
       location = Location specified by user
       created_at2 = Date account was created
       lang = Language
       latitude = Latitude coordinate (Only available if user opts-in to use Tweeting
       With Location Feature on)
       longitude = Longitude coordinate (Only available if user opts-in to use
       Tweeting With Location Feature on)
       date = date of tweet as SAS date value
WARNINGS:
If original tweets are included, the macro will return more observations than the
specified target amount.
Macro processing must be terminated manually if user wants to stop downloading before
the amount of tweets already downloaded is more than specified.
*************************************************************************************/
%MACRO Tweetomatic(search_term, target);
%LET oldloopcount = ;
%LET olderid = NA;
%LET loopvar = N;

/*Downloads 100 most recent tweets to create base file*/
%GRABTWEET(&search_term, 100);

/*Create a macro variable to count the number of observations*/
PROC SQL NOPRINT;
       SELECT COUNT(*)
       INTO :loopcount
       FROM test;
QUIT;

DATA first100;
SET test;
RUN;

/*Forces program to wait before next URL call. The path in the CALL SYSTEM command is
specific for a Windows computer. Path may need to be changed depending on operating
system.*/
OPTIONS NOXWAIT;
DATA _NULL_;
CALL SYSTEM('call C:\Users\Administrator\Documents\batch\batch_run2.bat');
RUN;

/*Downloads as many older tweets as possible than the last tweet downloaded above*/
%DO %UNTIL(&oldloopcount = 0 OR &oldloopcount = 1);
```

```
/*In order to grab more tweets and return older tweets (Tweets with IDs lower than
specified tweet_id), we need to store the id of the oldest tweet downloaded in a macro
variable. The stored value of &old_id will be used as the max_id parameter in the GET
search/tweets url.*/
              DATA last;
              SET first100 (FIRSTOBS = &loopcount);
              CALL SYMPUT("old_id", tweet_id);
              RUN;

              %LET loopvar = NA;
              %LET olderid = Y;

              %GRABTWEET(&search_term, 100);

              PROC APPEND BASE = first100 DATA = test FORCE;

              PROC SQL NOPRINT;
                     SELECT COUNT(*)
                     INTO :oldloopcount
                     FROM test;
              QUIT;
              %LET loopcount = %EVAL(&loopcount+&oldloopcount);
%END;
%LET i = 0;

/*In order to grab more tweets and return more recent tweets (Tweets with IDs higher
than specified tweet_id), we need to store the id of the most recent tweet in a macro
variable. The stored value of &recent_id will be used as the since_id parameter in the
GET search/tweets url.*/
DATA first;
       SET first100 (OBS = 1);
       CALL SYMPUT("recent_id", tweet_id);
RUN;

/*Downloads as many recent tweets as possible until number of tweets downloaded is
more than amount specified by user.*/
%DO %UNTIL(&loopcount > &target);
       %LET i=%EVAL(&i+1);
       %LET olderid = NA;
       %LET loopvar = Y;

       %GRABTWEET(&search_term, 100);

       PROC APPEND BASE = first100 DATA = test FORCE;

       PROC SQL NOPRINT;
              SELECT COUNT(*)
              INTO :recentloopcount
              FROM test;
       QUIT;
/*Calculates the time between observations for the most recently downloaded tweets*/
       DATA main_conversion;
              SET test;
/*temp ~=2  removes original tweets of retweets for calculating the rate of tweets*/
              IF temp ~= 2 THEN time_diff = abs(dif(date));
       RUN;

       PROC MEANS DATA = main_conversion MEAN;
              VAR time_diff;
              OUTPUT OUT = test_mean MEAN = avgtime;
       RUN;

/*Rounds average rate of tweets because the TIMEOUT batch command only accepts
```

```
integers. Next, it writes this integer to a text file, which will be used in the batch
script. The text file should only contain a single integer representing the amount of
time the program will wait before downloading more data*/
       DATA _null_;
              SET test_mean;
              avgrate = CEIL(avgtime);
              IF avgrate > 99999 THEN avgrate = 99999;
              FILE 'C:\Users\Administrator\Documents\batch\batch_test';
              PUT avgrate;
       RUN;
/*Executes batch processing that forces the program to wait the calculated amount of
time. The path in the CALL SYSTEM command is specific for a Windows computer. Path may
need to be changed depending on operating system.*/
       OPTIONS NOXWAIT;
       DATA _NULL_;
       CALL SYSTEM('call C:\Users\Administrator\Documents\batch\batch_run.bat');
       RUN;

       %LET loopcount = %EVAL(&loopcount+&recentloopcount);

       PROC SORT DATA = first100 NODUPKEY;
              BY DESCENDING tweet_id;
       RUN;

       PROC SQL NOPRINT;
              SELECT COUNT(*)
              INTO :loopcount
              FROM first100;
       QUIT;

/*Sometimes older tweets have a larger tweet_id of length 17. Due to the PROC SORT,
these tweets are at the top of the data set and will be incorrectly used as the id for
the since_id URL parameter. A subsetting IF checking whether the length of the tweet is
exactly 18 avoids the problem of downloading recent tweets based on an older date.*/

       DATA testing;
              SET first100;
              lengthid = length(tweet_id);
              IF lengthid = 18;
       RUN;

/*Sets id of most recent tweet into a macro variable to be used for the since_id URL
parameter*/
       PROC SQL NOPRINT;
              SELECT MAX(tweet_id)
              INTO :recent_id
              FROM testing;
       QUIT;

/*Executes batch processing that prompts the user whether or not to download more data
if there are more than 10 iterations and if the number of tweets most recently
downloaded are less than 5. If the user types "Y", the program will continue until the
number of observations downloaded is more than the target specified. If the user types
"N", batch script creates a file to be used in a condition to terminate current
processing of the macro.If the user does not respond within 10 seconds, the default
response is "Y". The path in the CALL SYSTEM command is specific for a Windows
computer. Path may need to be changed depending on operating system.
*/
%IF (&i > 10 AND &recentloopcount < 5) %THEN %DO;
OPTIONS NOXWAIT;
DATA _NULL_;
CALL SYSTEM('call C:\Users\Administrator\Documents\batch\prompt.bat');
RUN;
```

```
%END;

/*This code will execute if the user types "N" in the prompt from the batch script. It
will create a fileref for the file written by batch and checks for the file's
existence. If the file exists, it will delete the fileref and terminate current macro
processing toreturn downloaded tweets. */

%LET filrf = exitfile;
%LET rc = %SYSFUNC(FILENAME(filrf,
C:\Users\Administrator\Documents\batch\exittest.txt));
%IF %SYSFUNC(FEXIST(&filrf)) %THEN %DO;
       %SYSFUNC(FDELETE(&filrf));
       %LET rc = %SYSFUNC(FILENAME(&filrf));
       %RETURN;
%END;
%END;
%MEND;

%MACRO grabtweet(search_term, target_amount);
/*Specifies the input token file for authorization to access API 1.1*/
filename auth "C:\Users\Administrator\Documents\token.txt";

/*Specifies the output JSON file that will contain the tweets*/
filename twtOut "C:\Users\Administrator\Documents\data\Test";

/*Sets the following parameters to use in the GET search/tweets url:
       COUNT = number of tweets to grab
       RESULT_TYPE= what type of search tweets to grab. Options include the following:
              --> Popular = returns only the most popular tweets with regard to search
                            term
              --> Recent = returns only the most recent tweets with regard to search
                            term
              --> Mixed = popular and real time tweets
       SINCE_ID = returns tweets more recent than specified tweet
       MAX_ID = returns tweets older than specified tweet
*/

%IF &target_amount < 100 %THEN %LET num_tweet = %NRSTR(&count=)&target_amount;
%ELSE %LET num_tweet = %NRSTR(&count=100);
%LET type = %NRSTR(&result_type=recent);
%LET recent_tweet = %NRSTR(&since_id=)&recent_id;
%LET id = %NRSTR(&max_id=)&old_id;

/*Issues GET search/tweet URL to download the most recent 100 tweets for a search term
specified by the user*/
%IF &loopvar = N %THEN %DO;
PROC HTTP
HEADERIN = auth
METHOD = "get"
URL = "https://api.twitter.com/1.1/search/tweets.json?q=&search_term&type&num_tweet"
OUT = twtOut;
RUN;
%END;

/*Issues GET search/tweet URL to download 100 tweets more recent than the tweet id
specified by since_id parameter*/
%IF &loopvar = Y %THEN %DO;
PROC HTTP
HEADERIN = auth
METHOD = "get"
URL =
"https://api.twitter.com/1.1/search/tweets.json?q=&search_term&recent_tweet&type&num_t
weet"
```

```
OUT = twtOut;
RUN;
%END;

/*Issues GET search/tweet URL to download 100 tweets older than the tweet id specified
by max_id parameter*/
%IF &olderid = Y %THEN %DO;
PROC HTTP
HEADERIN = auth
METHOD = "get"
URL =
"https://api.twitter.com/1.1/search/tweets.json?q=&search_term&id&type&num_tweet"
OUT = twtOut;
RUN;
%END;

DATA test (DROP= text_start text_end userid_end name_end retweeted original_tweet
latitude_c longitude_c);
INFILE "C:\Users\Administrator\Documents\data\test" LRECL = 1000000 TRUNCOVER SCANOVER
dlm=',' dsd;
informat created_at1 $30. tweet_id $18. text $140. user_id $15. name $185. screen_name
$185. location $185. created_at2 $30.
        lang $2. retweeted $18. latitude_c $25. longitude_c $25. latitude 11.8
longitude 11.8;
INPUT @'"created_at":' created_at1
     @'"id":' tweet_id
     @'"text":' text
     @'"user":{"id":' user_id
     @'"name":' name
     @'"screen_name":' screen_name
     @'"location":' location
     @'"created_at":' created_at2
     @'"lang":' lang
     @'"coordinates":' latitude_c @;
             IF latitude_c = 'null' THEN INPUT  @'butors":null,' retweeted @@;
             ELSE INPUT longitude_c @'butors":null,' retweeted @@;

IF latitude_c = 'null' THEN latitude_c = .;
ELSE latitude_c = SUBSTR(latitude_c, 2, 11);
latitude = INPUT(latitude_c, 11.8);
IF longitude_c = '' THEN longitude_c = .;
ELSE longitude_c = SUBSTR(longitude_c, 1, 11);
IF FIND(longitude_c, ']') ~= 0 THEN longitude_c = SUBSTR(longitude_c, 1,
INDEX(longitude_c, ']')-1);
IF FIND(longitude_c, '}') ~= 0 THEN longitude_c = SUBSTR(longitude_c, 1,
INDEX(longitude_c, '}')-1);
longitude = INPUT(longitude_c, 11.8);
text_start = INDEX(text, '"')+1;
text_end = INDEX(text,'","');
IF text_end ~= 0 THEN text = SUBSTR(text,text_start,text_end-1);
IF text_end = 0 THEN text = SUBSTR(text, text_start);
userid_end = INDEX(user_id, ',"');
IF userid_end ~= 0 THEN user_id = SUBSTR(user_id,1,userid_end-1);
name_end = INDEX(name, '","s');
IF name_end = 1 THEN DO;
name_end = 2;
       name = 'NA';
END;
IF name_end ~IN(0,1) THEN name = SUBSTR(name, 1, abs(name_end-1));

/*The macro is suppose to return only the most recent tweets, but the file contains
the original tweets of retweets, so we must distinguish between original tweets and
retweets using the variable Original_tweet. Retweets will have the
```

```
string "retweeted_status" as a member while non-tweets will have a different
string.Since original_tweet will be a nonzero integer for retweets instead of the
original tweets, the variable must be lagged so that "retweeted_status" will be part
of the observation representing original tweets. Now that original_tweet is a nonzero
integer for original tweets, these observations can be flagged, and regular
tweets(whose value of temp will be zero) can be used in the CALL SYMPUT function to
grab the tweet_id for the max_id URL parameter.*/
       original_tweet = FIND(retweeted, "retweeted_status");
       temp = LAG(original_tweet);

/*Converts date of tweet to SAS date*/
date = input(cat(substr(created_at1, 9, 2), substr(created_at1, 5, 3),
substr(created_at1, 29, 2), ':', substr(created_at1, 12, 8)), DATETIME16.);
RUN;

%MEND;
```