NOTE: The likelihood of this recipe being added is slim, but I’ll try to keep the same format for the blog post.

**Problem**

You want to collect all the tweets in a Twitter tweet thread

**Solution**

Use a few key functions in rtweet to piece the thread elements back together.

**Discussion**

In Twitterland, a “thread” is a series of tweets by an author that are in a reply chain to each other which enables them to be displayed sequentially to form a larger & (ostensibly) more cohesive message. Even with the recent 280 character tweet-length increase, threads are still popular and used daily. They’re very easy to distinguish on Twitter but there is no Twitter API call to collect up all the pieces of these threads.

Let’s build a function — get\_thread() — that will take as input a starting thread URL or status id and return a data frame of all the tweets in the thread (in order). As a bonus, we’ll also include a way to include all first-level retweets and replies to each threaded tweet (that, too, happens quite a bit).

There are documentation snippets in the code block (below), but the essence of the function is:

* first, finding the tweet that belongs to the status id to get some metadata
* then doing a search for tweets from the author that occurred after that tweet (we do this to save on API calls and we grab a bunch of them)
* rather than do a bunch of things by hand, we make from/to pairs to feed in as vertex edges into igraph
* once that’s done, separate out the graph into unique subgraphs and find the one containing the starting status id
* since that subgraph is just a set of status ids, rebuild the data frame from it and put it in order.

There may be occasions where we want to grab the replies or RTs of any of the original thread tweets. They aren’t always useful, but when they are it’d be good to have this context. So, we’ll add an option that — if TRUE — will cause the function to go down the list of threaded tweets and pull the first-level replies and RTs (excluding the ones from the author). We’ll do this using the Twitter search API as it’ll ultimately save on API calls and it puts the filtering closer to the data (I’m generally “a fan” of putting computation as close to the data as possible for any given task). If there were any, they’ll be in a replies column which can be unnested at-will.

Here’s the complete function:

get\_thread <- function(first\_status, include\_replies=FALSE, .timeline\_history=3000) {

require(rtweet, quietly=TRUE, warn.conflicts=FALSE)

require(igraph, quietly=TRUE, warn.conflicts=FALSE)

require(tidyverse, quietly=TRUE, warn.conflicts=FALSE)

first\_status <- if (str\_detect(first\_status[1], "^http[s]://")) basename(first\_status[1]) else first\_status[1]

# get first status

orig <- rtweet::lookup\_tweets(first\_status)

# grab the author's timeline to search

author\_timeline <- rtweet::get\_timeline(orig$screen\_name, n=.timeline\_history, since\_id=first\_status)

# build a data frame containing from/to pairs (anything the author

# replied to) that also includes the `first\_status` id.

suppressWarnings(

dplyr::filter(

author\_timeline,

(status\_id == first\_status) | (reply\_to\_screen\_name == orig$screen\_name)

) %>%

dplyr::select(status\_id, reply\_to\_status\_id) %>%

igraph::graph\_from\_data\_frame() -> g

) # build a graph from this

# decompose the graph into unique subgraphs and return them to data frames

igraph::decompose(g) %>%

purrr::map(igraph::as\_data\_frame) -> threads\_dfs

# find the thread with our `first\_status` ids

thread\_df <- purrr::keep(threads\_dfs, ~any(which(unique(unlist(.x, use.names=FALSE)) == first\_status)))

# BONUS: we get them in the order we need!

thread\_order <- purrr::discard(rev(unique(unlist(thread\_df))), str\_detect, "NA")

# filter out the thread from the timeline corpus & sort it

dplyr::filter(author\_timeline, status\_id %in% pull(thread\_df[[1]], from)) %>%

dplyr::mutate(status\_id = factor(status\_id, levels=thread\_order)) %>%

dplyr::arrange(status\_id) -> tweet\_thread

if (include\_replies) {

# for each status, lookup 1st-level references to it, excluding ones from the original author

mutate(

tweet\_thread,

replies = purrr::map(

as.character(status\_id),

~rtweet::search\_tweets(sprintf("%s -from:%s", .x, orig$screen\_name[1]))

)

) -> tweet\_thread

}

class(tweet\_thread) <- c("tweet\_thread", class(tweet\_thread))

return(tweet\_thread)

}

Now, the function will return the following:

xdf <- get\_thread("https://twitter.com/petersagal/status/952910499825451009")

glimpse(select(xdf, 1:5))

## Observations: 10

## Variables: 5

## $ status\_id 952910499825451009, 952910695804305408, 952911012990193664, 952911632077852679, 9529121...

## $ created\_at 2018-01-15 14:29:02, 2018-01-15 14:29:48, 2018-01-15 14:31:04, 2018-01-15 14:33:31, 201...

## $ user\_id "14985228", "14985228", "14985228", "14985228", "14985228", "14985228", "14985228", "149...

## $ screen\_name "petersagal", "petersagal", "petersagal", "petersagal", "petersagal", "petersagal", "pet...

## $ text "Funny you mention that. I talked to Minniejean (Brown) Trickey, one of the Little Rock ...

purrr::map(xdf$text, strwrap) %>%

purrr::map\_chr(paste0, collapse="\n") %>%

cat(sep="\n\n")

## Funny you mention that. I talked to Minniejean (Brown) Trickey, one of the Little Rock Nine, about

## that very day in front of CHS for my documentary, "Constitution USA." https://t.co/MRwtlfZtvp

##

## You would think that of all people, she would be satisfied with the government's response to racism

## and hate. Ike sent the 101st Airborne to escort her to class!

##

## But what I didn't know is that after the 101st left, CHS expelled her on a trumped up charge of

## assault after she spilled some chili on a white student.

##

## She spilled some chili. After being tripped by another white kid. "We got rid of one of them!" the

## teachers bragged.

##

## Then, of course, rather than continue to allow black students to attend CHS, the governor of Alabama

## closed the schools. https://t.co/2DfBEI0OTL"The\_Lost\_Year"

##

## Ms Brown looked around the country post high school. She saw Jim Crow, firehoses turned on Blacks,

## the murder of the Birmingham Four and the Mississippi Three. She moved to Canada.

##

## As of 2012, she found herself coming back to Little Rock, a place she told me she never wanted to

## see again. But she had family. And the National Historic Site center was there. She liked to drop

## by, talk to the kids about what happened.

##

## Now she lives in Little Rock full time. She doesn't care that her name is inscribed on a bench in

## front of the school. She doesn't care that your dad welcomed her back in '99. She spends time at

## the Center, telling people what really happened. You should go talk to her.

##

## (Sorry: Arkansas, obviously. Typing too quickly.)

##

## Here's me, talking to Ms Trickey and Marty Sammon, who served with the 101st at Little Rock. Buddy

## Squiers on camera. CHS is off to the left. https://t.co/ft4LUBf3sr

##

## https://t.co/EHLbe1finj

The replies data frame looks much the same as the thread data frame — it’s essentially just another rtweet data frame, so we won’t waste electrons showing it.

While that map/map/cat sequence isn’t bad to type, it’d be more convenient if we had a print() method for this structure (this is one reason we added a class to it). It’d be even spiffier if this print() method made it easier to distinguish the main thread from the RT’s/replies — but still show those extra bits of info. We’ll use the crayon package for added emphasis:

print.tweet\_thread <- function(x, ...) {

cat(crayon::cyan(sprintf("@%s - %s\n\n", x$screen\_name[1], x$created\_at[1])))

if (!("replies" %in% colnames(x))) x$replies <- purrr::map(1:nrow(x), ~list())

purrr::walk2(x$text, x$replies, ~{

cat(crayon::green(paste0(strwrap(.x), collapse="\n")), "\n\n", sep="")

if (length(.y) > 0) {

purrr::walk2(.y$screen\_name, .y$text, ~{

sprintf("@%s\n%s", .x, .y) %>%

strwrap(indent=8, exdent=8) %>%

paste0(collapse="\n") %>%

crayon::silver$italic() %>%

cat("\n\n", sep="")

})

}

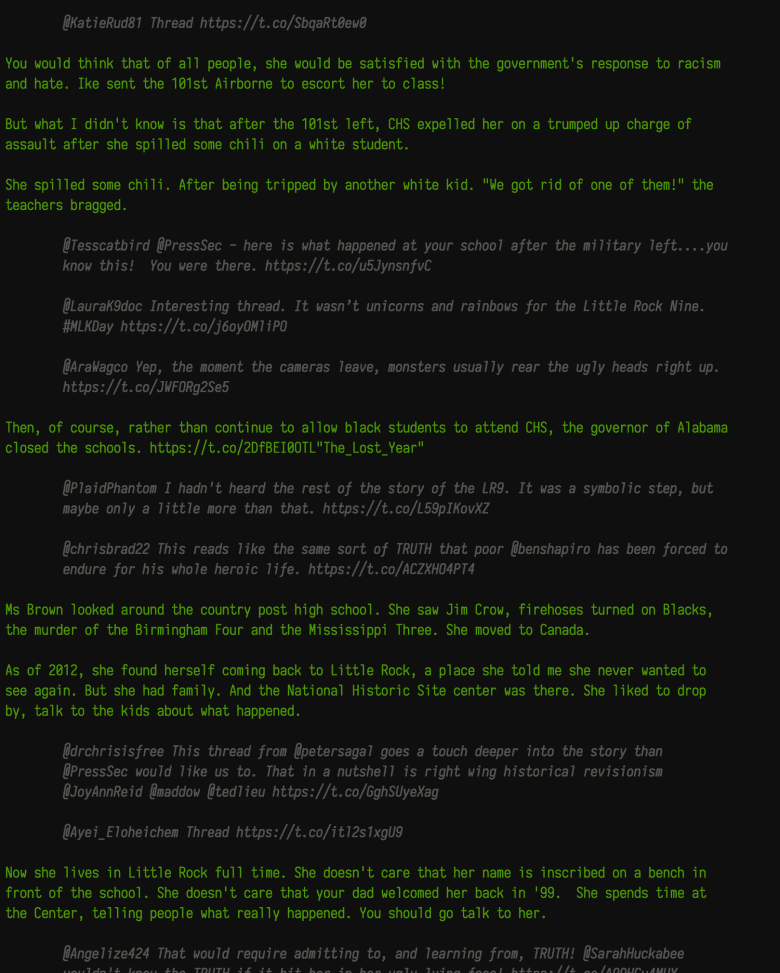
})

}

Let’s re-capture the tweet thread but also include replies this time and print it out:

ydf <- get\_thread("https://twitter.com/petersagal/status/952910499825451009", include\_replies=TRUE)

ydf

[](https://i1.wp.com/rud.is/b/wp-content/uploads/2018/01/Screen-Shot-2018-01-15-at-3.01.13-PM.png?ssl=1)

**See Also**

I’ve git-chatted with Sir Kearney to see where to best put this function. I mention that as there are some upcoming posts that kick the aforeblogged tweet\_shot() up a notch or two and all of this may work better in a tweetview package.

Regardless, drop a note in the comments if there are other bits of functionality or function options you think belong in get\_thread().

Code Chunks to Grab the tweet

The ultimate goal is to make a function that builds a tweet using only R and magick. This will have to do until the new year.

tweet\_shot <- function(statusid\_or\_url, zoom=3) {

require(glue, quietly=TRUE)

require(rtweet, quietly=TRUE)

require(magick, quietly=TRUE)

require(webshot, quietly=TRUE)

x <- statusid\_or\_url[1]

is\_url <- grepl("^http[s]://", x)

if (is\_url) {

is\_twitter <- grepl("twitter", x)

stopifnot(is\_twitter)

is\_status <- grepl("status", x)

stopifnot(is\_status)

already\_mobile <- grepl("://mobile\\.", x)

if (!already\_mobile) x <- sub("://twi", "://mobile.twi", x)

} else {

x <- rtweet::lookup\_tweets(x)

stopifnot(nrow(x) > 0)

x <- glue\_data(x, "https://mobile.twitter.com/{screen\_name}/status/{status\_id}")

}

tf <- tempfile(fileext = ".png")

on.exit(unlink(tf), add=TRUE)

webshot(url=x, file=tf, zoom=zoom)

img <- image\_read(tf)

img <- image\_trim(img)

if (zoom > 1) img <- image\_scale(img, scales::percent(1/zoom))

img

}

Now just do one of these:

tweet\_shot("947082036019388416")

tweet\_shot("https://twitter.com/jhollist/status/947082036019388416")