21 Recipes for Mining Twitter Data with rtweet

Bob Rudis

2018-01-03

Table of Contents

# Preface

I’m using this as way to familiarize myself with bookdown so I don’t make as many mistakes with my web scraping field guide book.

It’s based on Matthew R. Russell’s [book](https://github.com/ptwobrussell/Recipes-for-Mining-Twitter). That book is out of distribution and much of the content is in Matthew’s “Mining the Social Web” book. There will be *many* similarities between his “21 Recipes” book and this book *on purpose*. I am not claiming originality in this work, just making an R-centric version of the cookbook.

As he states in his tome, “this intentionally terse recipe collection provides you with 21 easily adaptable Twitter mining recipes”.

The recipes contained in this book use the [rtweet](http://rtweet.info/) package by [Michael W. Kearney](https://github.com/mkearney). I’ll be using the [GitHub](https://github.com/mkearney/rtweet) version of the package since it has some cutting-edge features and bug-fixes in it.

You can install the GitHub version of [rtweet] by first installing the [devtools](https://github.com/hadley/devtools) package via:

install.packages("devtools")

then installing the GitHub rtweet package via:

devtools::install\_github("mkearney/rtweet")

NOTE: If you try to run examples in this book and receive an error about a package not being found or not available, you’ll need to triage it by using one of the above methods. If any GitHub packages are used, each initial library() call will have a comment after it noting which repository/packagename to use the devtools method with.

Matthew also states that “one other thing you should consider doing up front, if you haven’t already, is quickly skimming through the official Twitter API documentation and related development documents linked on that page. Twitter has a very easy-to-use API with a lot of degrees of freedom”. Michael has documented rtweet well, but reading the official documentation will really help.

This book also makes extensive use of the tidyverse meta-package. You will need to:

install.packages("tidyverse")

if you have not used packages from it before (it may take a few minutes, especially on Linux systems).

# About the Author

Bob Rudis is a cybersecurity researcher and R afficionado presently thwartng cyber evildoers as [Master] Chief Data Scientist at Rapid7. He was formerly a Security Data Scientist & Managing Principal at Verizon, overseeing the team that produces the annual Data Breach Investigations Report. Bob is a serial tweeter (<https://twitter.com/hrbrmstr>), avid blogger (<https://rud.is/b>), author (Data-Driven Security — [<http://amzn.to/2CKvrqX>]), Stack Overflow contributor (<https://stackoverflow.com/users/1457051/hrbrmstr>), speaker, and regular contributor to the open source community (<https://github.com/hrbrmstr>). He is the author of several packages on CRAN including **ggalt**, **hrbrthemes**, **waffle**, **statebins**.

# Using OAuth to Access Twitter APIs

## Problem

You want to access your own data or another user’s data for analysis.

## Solution

Take advantage of Twitter’s OAuth implementation to gain full access to Twitter’s entire API.

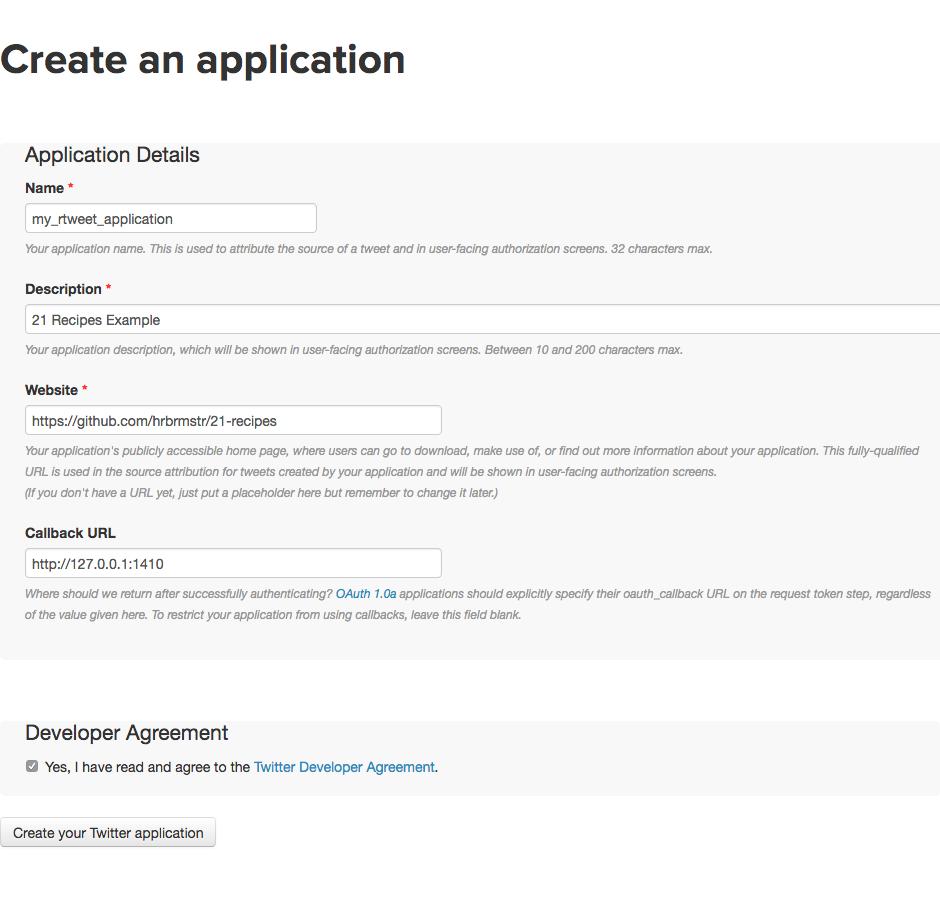
## Discussion

Twitter uses [OAuth Core 1.0 Revision A](https://oauth.net/core/1.0a/) (“OAuth 1.0a” for short & to further reduce verbosity, “oauth” from now on). A few, key purposes of oauth in the context of Twitter are:

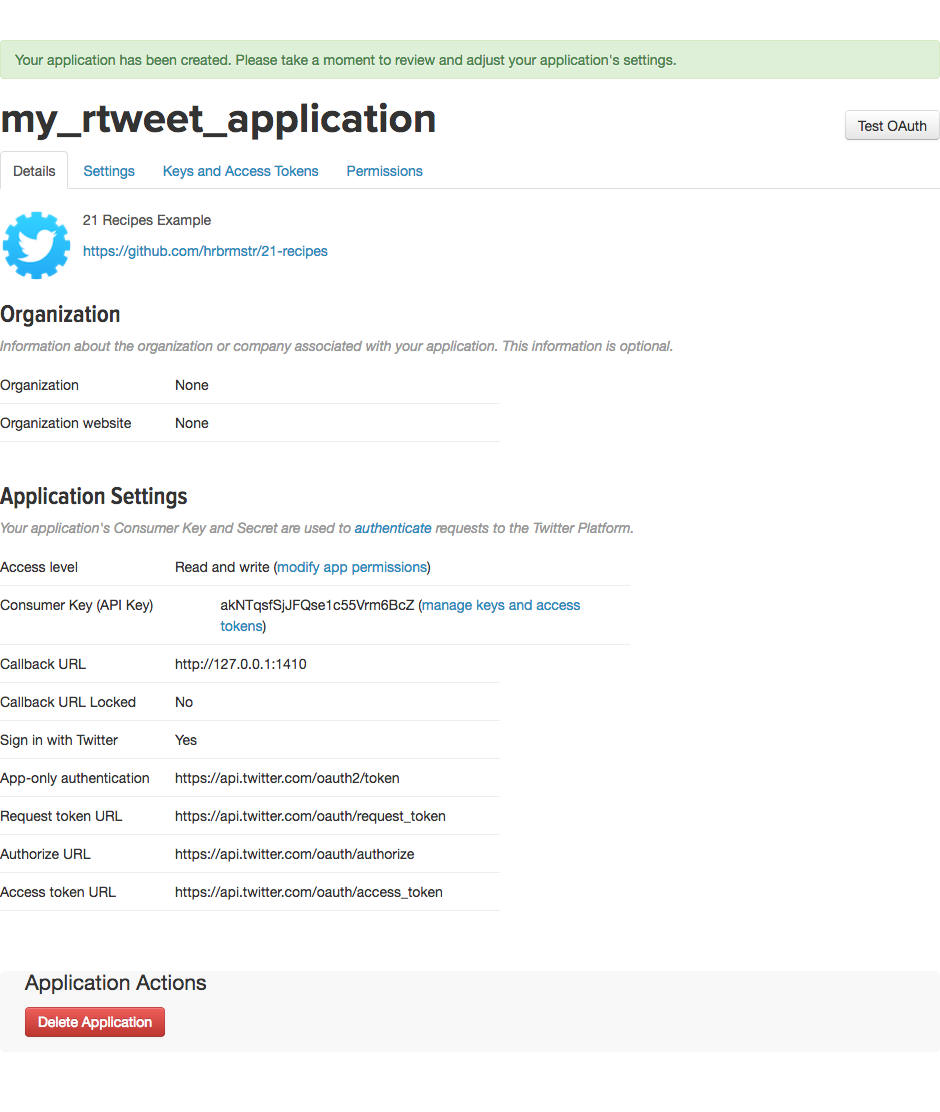
* to ensure end-users know an application is registered with Twitter, and
* know who the author(s) fo the application are;
* enable limiting what operations an application can perform with your Twitter account;
* obviate the need to share your actual Twitter username and password with a third party, which also
* enables recovation of application access to your Twitter account without resetting your password.

The rtweet package takes this one step further by having *you* create an “application”, which is nothing more than you setting up some basic configuration information. To do so, you must visit [apps.twitter.com](https://apps.twitter.com/) and create a new application. You will need to provide values for the following fields:

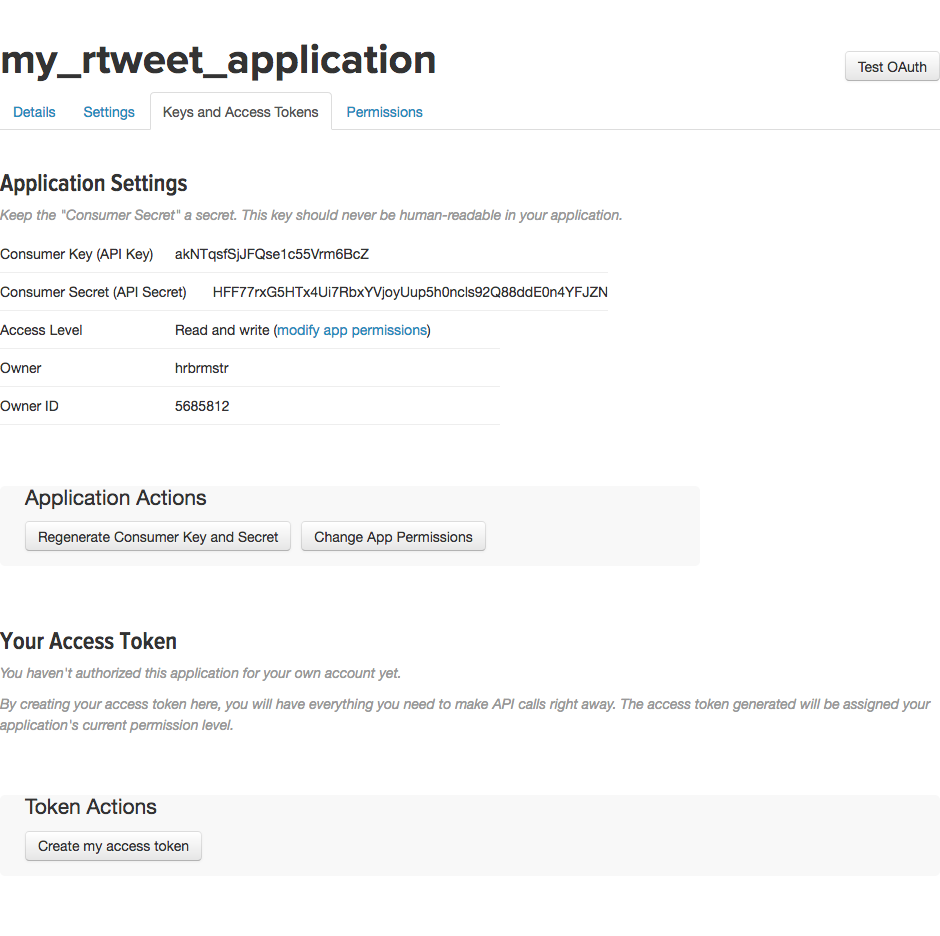
* Name : something you’ll remember
* Description : another place you can remind yourself what this is for
* Website : something that points to information you can use to associate this app when you’ve forgotten about it 5 years from now
* Callback URL : This **must** be http://127.0.0.1:1410 (we’ll see why in a moment)
* tick the agreement checkbox



Once you submit that form, you’ll see a new page:



Select the “Keys and Access Tokens” tab to see important information you’ll need:



From the previous page and this page, you’ll need the:

* Application **Name** (which is my\_rtweet\_application in this example but you need to use the one you supplied)
* **Consumer Key (API Key)** (which is akNTqsfSjJFQse1c55Vrm6BcZ in this example but you need to use your own)
* **Consumer Secret (API Secret)** (which is HFF77rxG5HTx4Ui7RbxYVjoyUup5h0ncls92Q88ddE0n4YFJZN in this example, but — again — you need to use your own)

Store both of those in your ~/.Renviron file. If you’re unfamiliar with how to do that, see [this handy section](https://csgillespie.github.io/efficientR/3-3-r-startup.html#renviron) from “Efficient R Programming”. I prefer storing these as such:

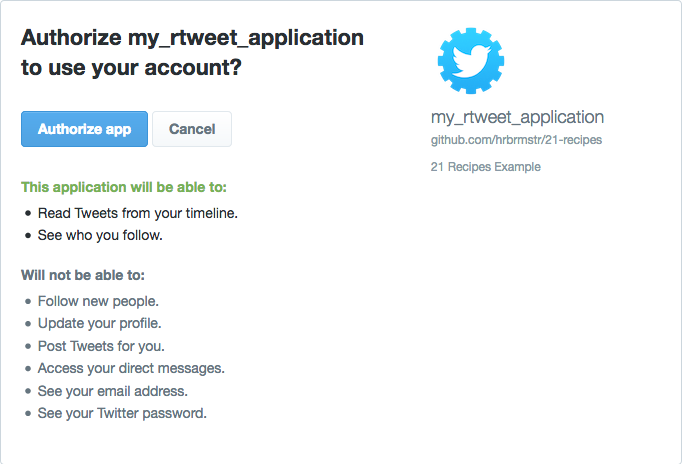
TWITTER\_APP=my\_rtweet\_application  
TWITTER\_CONSUMER\_KEY=akNTqsfSjJFQse1c55Vrm6BcZ  
TWITTER\_CONSUMER\_SECRET=HFF77rxG5HTx4Ui7RbxYVjoyUup5h0ncls92Q88ddE0n4YFJZN

By storing these values in ~/.Renviron you avoid exposing them in subdirectories or within scripts and will always be able to reference them.

Now you can enable your Twitter account with this application and create a *token*:

create\_token(  
 app = Sys.getenv("TWITTER\_APP"),  
 consumer\_key = Sys.getenv("TWITTER\_CONSUMER\_KEY"),  
 consumer\_secret = Sys.getenv("TWITTER\_CONSUMER\_SECRET")  
) -> twitter\_token

You should see a browser window appear that has an authorization form in it:



You’ll also see:

Waiting for authentication in browser...  
Press Esc/Ctrl + C to abort

in the R console.

The rtweet package used httr to send an oauth request to Twitter and then started up a local web server (this is why that weird localhost URL from before is necessary). When you authorize the application, the browser sends a response back to the web server httr spun up with some important, secret information that will make it possible for you to never have to do this oauth dance again.

If everything was successful, you’ll see:

Authentication complete. Please close this page and return to R.

in the browser window, and:

Authentication complete.

in the R console.

The next step is *very important*.

Save the secret token you just received this way:

saveRDS(twitter\_token, "~/.rtweet.rds")

then create *one more* environment variable in ~/.Renviron:

TWITTER\_PAT=~/.rtweet.rds

That last step will help ensure you never have to deal with oauth again (until you want to).

**Keep this token file safe**!! It enables anyone who has it to do virtually anything with your account. If you believe it has been exposed, go back to [apps.twitter.com](https://apps.twitter.com/) and delete the application (you can also choose to regenerate the Consumer Key and Consumer Secret, but it’s often easier to just make a new application). You should also review your [Twitter apps](https://twitter.com/settings/applications) and ensure it’s removed from there as well. Use the Revoke access button if it is:



## See Also

* The official rtweet [authentication vignette](http://rtweet.info/articles/auth.html)

# Looking Up the Trending Topics

## Problem

You want to keep track of the trending topics on Twitter over a period of time.

## Solution

Use rtweet::trends\_available() to see trend areas and rtweet::get\_trends() to pull trends, after which you can setup a task to retrieve and cache the trend data periodically.

## Discussion

Twitter has [extensive information](https://help.twitter.com/en/using-twitter/twitter-trending-faqs) on trending topics and their API enables you to see topics that are trending globally or regionally. Twitter uses [Yahoo! Where on Earth](https://developer.yahoo.com/geo/geoplanet/guide/concepts.html) identifiers (WOEIDs) for the regions which can be obtained from rtweet::trends\_available():

library(rtweet)  
library(tidyverse)

(trends\_avail <- trends\_available())

## # A tibble: 467 x 8  
## name url parentid  
## \* <chr> <chr> <int>  
## 1 Worldwide http://where.yahooapis.com/v1/place/1 0  
## 2 Winnipeg http://where.yahooapis.com/v1/place/2972 23424775  
## 3 Ottawa http://where.yahooapis.com/v1/place/3369 23424775  
## 4 Quebec http://where.yahooapis.com/v1/place/3444 23424775  
## 5 Montreal http://where.yahooapis.com/v1/place/3534 23424775  
## 6 Toronto http://where.yahooapis.com/v1/place/4118 23424775  
## 7 Edmonton http://where.yahooapis.com/v1/place/8676 23424775  
## 8 Calgary http://where.yahooapis.com/v1/place/8775 23424775  
## 9 Vancouver http://where.yahooapis.com/v1/place/9807 23424775  
## 10 Birmingham http://where.yahooapis.com/v1/place/12723 23424975  
## # ... with 457 more rows, and 5 more variables: country <chr>,  
## # woeid <int>, countryCode <chr>, code <int>, place\_type <chr>

glimpse(trends\_avail)

## Observations: 467  
## Variables: 8  
## $ name <chr> "Worldwide", "Winnipeg", "Ottawa", "Quebec", "Mont...  
## $ url <chr> "http://where.yahooapis.com/v1/place/1", "http://w...  
## $ parentid <int> 0, 23424775, 23424775, 23424775, 23424775, 2342477...  
## $ country <chr> "", "Canada", "Canada", "Canada", "Canada", "Canad...  
## $ woeid <int> 1, 2972, 3369, 3444, 3534, 4118, 8676, 8775, 9807,...  
## $ countryCode <chr> NA, "CA", "CA", "CA", "CA", "CA", "CA", "CA", "CA"...  
## $ code <int> 19, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7...  
## $ place\_type <chr> "Supername", "Town", "Town", "Town", "Town", "Town...

The Twitter API is somewhat unforgiving and unfriendly when you use it directly since it requires the use of a WOEID. Michael has made life much easier for us all by enabling the use of names or regular expressions when asking for trends from a particular place. That means we don’t even need to care about capitalization:

(us <- get\_trends("united states"))

## # A tibble: 44 x 9  
## trend  
## \* <chr>  
## 1 #WednesdayWisdom  
## 2 Steve Bannon  
## 3 #BombCyclone  
## 4 #BiggestMisconceptionAboutMe  
## 5 Tallahassee  
## 6 #BadDaysIn5Words  
## 7 #1linewed  
## 8 James Risen  
## 9 Bill Lazor  
## 10 Eric Hosmer  
## # ... with 34 more rows, and 8 more variables: url <chr>,  
## # promoted\_content <lgl>, query <chr>, tweet\_volume <int>, place <chr>,  
## # woeid <int>, as\_of <dttm>, created\_at <dttm>

glimpse(us)

## Observations: 44  
## Variables: 9  
## $ trend <chr> "#WednesdayWisdom", "Steve Bannon", "#BombCyc...  
## $ url <chr> "http://twitter.com/search?q=%23WednesdayWisd...  
## $ promoted\_content <lgl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, N...  
## $ query <chr> "%23WednesdayWisdom", "%22Steve+Bannon%22", "...  
## $ tweet\_volume <int> 46551, 53663, NA, NA, 19722, NA, NA, NA, NA, ...  
## $ place <chr> "United States", "United States", "United Sta...  
## $ woeid <int> 23424977, 23424977, 23424977, 23424977, 23424...  
## $ as\_of <dttm> 2018-01-03 16:22:42, 2018-01-03 16:22:42, 20...  
## $ created\_at <dttm> 2018-01-03 16:18:00, 2018-01-03 16:18:00, 20...

Twitter’s [documentation](https://developer.twitter.com/en/docs/trends/trends-for-location/api-reference/get-trends-place) states that trends are updated every 5 minutes, which means you should not call the API more frequently than that and their current API rate-limit (Twitter puts some restrictions on how frequently you can call certain API targets) is 75 requests per 15-minute window.

The rtweet::get\_trends() function returns a data frame. Our ultimate goal is to retrieve the trends data on a schedule and cache it. There are numerous — and usually complex – ways to schedule jobs. One cross-platform solution is to use R itself to run a task periodically. This means keeping an R console open and running at all times, so is far from an optimal solution. See the [taskscheduleR](https://github.com/bnosac/taskscheduleR) package for other ideas on how to setup more robust scheduled jobs.

In this example, we will:

* use a [SQLite](https://www.sqlite.org/) database to store the trends
* use the DBI add RSQlite packages to work with this database
* setup a never-ending loop with Sys.sleep() providing a pause between requests

library(DBI)  
library(RSQLite)  
library(rtweet) # mkearney/rtweet  
  
repeat {  
 message("Retrieveing trends...") # optional  
 us <- get\_trends("united states")  
 db\_con <- dbConnect(RSQLite::SQLite(), "data/us-trends.db")  
 dbWriteTable(db\_con, "us\_trends", us, append=TRUE) # append=TRUE will update the table vs overwrite and also create it on first run if it does not exist  
 dbDisconnect(db\_con)  
 Sys.sleep(10 \* 60) # sleep for 10 minutes  
}

Later on, we can look at this data with dplyr/dbplyr:

library(dplyr)  
  
trends\_db <- src\_sqlite("data/us-trends.db")  
us <- tbl(trends\_db, "us\_trends")  
select(us, trend)

## # Source: lazy query [?? x 1]  
## # Database: sqlite 3.19.3  
## # [/Users/hrbrmstr/Development/21-recipes/data/us-trends.db]  
## trend  
## <chr>  
## 1 #TuesdayThoughts  
## 2 #backtowork  
## 3 #SavannahHodaTODAY  
## 4 Justin Timberlake  
## 5 #MyTVShowWasCanceledBecause  
## 6 #AM2DM  
## 7 The Trump Effect  
## 8 Carrie Underwood  
## 9 Sean Ryan  
## 10 Larry Krasner  
## # ... with more rows

## See Also

* [RSQlite](https://www.r-project.org/nosvn/pandoc/RSQLite.html) quick reference
* Introduction to dbplyr : <http://dbplyr.tidyverse.org/articles/dbplyr.html>

# Extracting Tweet Entities

## Problem

You want to extract tweet entities such as @mentions, #hashtags, and short URLs from Twitter search results or other batches of tweets.

## Solution

Use rtweet::search\_tweets() or any of the *timeline* functions in rtweet.

## Discussion

Michael has provided a very powerful search interace for Twitter data mining. rtweet::search\_tweets() retrieves, parses and extracts an asounding amount of data for you to then use. Let’s search Twitter for the #rstats hashtag and see what is available:

library(rtweet)  
library(tidyverse)

(rstats <- search\_tweets("#rstats", n=300)) # pull 300 tweets that used the "#rstats" hashtag

## # A tibble: 300 x 42  
## status\_id created\_at user\_id  
## <chr> <dttm> <chr>  
## 1 948590403036569601 2018-01-03 16:22:30 280608149  
## 2 948590359524786176 2018-01-03 16:22:20 781583089667420161  
## 3 948590304223006721 2018-01-03 16:22:07 594746010  
## 4 948590234584911872 2018-01-03 16:21:50 144529492  
## 5 948589562187735040 2018-01-03 16:19:10 280455470  
## 6 948589232838332416 2018-01-03 16:17:51 18057156  
## 7 948589231332560896 2018-01-03 16:17:51 732217484972048384  
## 8 948588899265257472 2018-01-03 16:16:32 368551889  
## 9 948588777152417793 2018-01-03 16:16:03 37686255  
## 10 948588614715367424 2018-01-03 16:15:24 2865404679  
## # ... with 290 more rows, and 39 more variables: screen\_name <chr>,  
## # text <chr>, source <chr>, reply\_to\_status\_id <chr>,  
## # reply\_to\_user\_id <chr>, reply\_to\_screen\_name <chr>, is\_quote <lgl>,  
## # is\_retweet <lgl>, favorite\_count <int>, retweet\_count <int>,  
## # hashtags <list>, symbols <list>, urls\_url <list>, urls\_t.co <list>,  
## # urls\_expanded\_url <list>, media\_url <list>, media\_t.co <list>,  
## # media\_expanded\_url <list>, media\_type <list>, ext\_media\_url <list>,  
## # ext\_media\_t.co <list>, ext\_media\_expanded\_url <list>,  
## # ext\_media\_type <lgl>, mentions\_user\_id <list>,  
## # mentions\_screen\_name <list>, lang <chr>, quoted\_status\_id <chr>,  
## # quoted\_text <chr>, retweet\_status\_id <chr>, retweet\_text <chr>,  
## # place\_url <chr>, place\_name <chr>, place\_full\_name <chr>,  
## # place\_type <chr>, country <chr>, country\_code <chr>,  
## # geo\_coords <list>, coords\_coords <list>, bbox\_coords <list>

glimpse(rstats)

## Observations: 300  
## Variables: 42  
## $ status\_id <chr> "948590403036569601", "9485903595247861...  
## $ created\_at <dttm> 2018-01-03 16:22:30, 2018-01-03 16:22:...  
## $ user\_id <chr> "280608149", "781583089667420161", "594...  
## $ screen\_name <chr> "cortinah", "LynnYarmey", "thibaultdej"...  
## $ text <chr> "Airline passenger safety has clearly i...  
## $ source <chr> "TweetDeck", "Twitter for iPhone", "Twi...  
## $ reply\_to\_status\_id <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,...  
## $ reply\_to\_user\_id <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,...  
## $ reply\_to\_screen\_name <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,...  
## $ is\_quote <lgl> FALSE, FALSE, FALSE, FALSE, FALSE, FALS...  
## $ is\_retweet <lgl> FALSE, TRUE, FALSE, FALSE, FALSE, TRUE,...  
## $ favorite\_count <int> 0, 0, 0, 1, 1, 0, 0, 2, 1, 0, 0, 0, 0, ...  
## $ retweet\_count <int> 0, 11, 0, 0, 2, 9, 15, 1, 0, 15, 4, 0, ...  
## $ hashtags <list> [<"rstats", "ggplot2">, "rstats", "rst...  
## $ symbols <list> [NA, NA, NA, NA, NA, NA, NA, NA, NA, N...  
## $ urls\_url <list> [NA, NA, "tidyverse.org/articles/2017/...  
## $ urls\_t.co <list> [NA, NA, "https://t.co/xFf78u9sIX", "h...  
## $ urls\_expanded\_url <list> [NA, NA, "https://www.tidyverse.org/ar...  
## $ media\_url <list> ["http://pbs.twimg.com/media/DSoOd51Wk...  
## $ media\_t.co <list> ["https://t.co/DNifWLdiVY", NA, NA, NA...  
## $ media\_expanded\_url <list> ["https://twitter.com/cortinah/status/...  
## $ media\_type <list> ["photo", NA, NA, NA, NA, NA, "photo",...  
## $ ext\_media\_url <list> ["http://pbs.twimg.com/media/DSoOd51Wk...  
## $ ext\_media\_t.co <list> ["https://t.co/DNifWLdiVY", NA, NA, NA...  
## $ ext\_media\_expanded\_url <list> ["https://twitter.com/cortinah/status/...  
## $ ext\_media\_type <lgl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,...  
## $ mentions\_user\_id <list> [NA, <"189172089", "862680370436874240...  
## $ mentions\_screen\_name <list> [NA, <"rgfitzjohn", "vaccineimpact">, ...  
## $ lang <chr> "en", "en", "fr", "en", "en", "en", "en...  
## $ quoted\_status\_id <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,...  
## $ quoted\_text <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,...  
## $ retweet\_status\_id <chr> NA, "948501344989663233", NA, NA, NA, "...  
## $ retweet\_text <chr> NA, "Last week to apply for our #rstats...  
## $ place\_url <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,...  
## $ place\_name <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,...  
## $ place\_full\_name <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,...  
## $ place\_type <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,...  
## $ country <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,...  
## $ country\_code <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,...  
## $ geo\_coords <list> [<NA, NA>, <NA, NA>, <NA, NA>, <NA, NA...  
## $ coords\_coords <list> [<NA, NA>, <NA, NA>, <NA, NA>, <NA, NA...  
## $ bbox\_coords <list> [<NA, NA, NA, NA, NA, NA, NA, NA>, <NA...

From the output, you can see that all the URLs (short and expanded), status id’s, user id’s and other hashtags are all available and all in a [tidy](http://r4ds.had.co.nz/tidy-data.html) data frame.

What are the top 10 (with ties) other hashtags used in conjunction with #rstats (for this search group)?

select(rstats, hashtags) %>%   
 unnest() %>%   
 mutate(hashtags = tolower(hashtags)) %>%   
 count(hashtags, sort=TRUE) %>%   
 filter(hashtags != "rstats") %>%   
 top\_n(10)

## # A tibble: 10 x 2  
## hashtags n  
## <chr> <int>  
## 1 datascience 60  
## 2 bigdata 23  
## 3 ggplot2 15  
## 4 python 15  
## 5 dataviz 13  
## 6 machinelearning 11  
## 7 abdsc 10  
## 8 dbplyr 10  
## 9 dplyr 10  
## 10 statistics 10

## See Also

* Official Twitter [search API](https://developer.twitter.com/en/docs/tweets/search/guides/build-standard-query) documentation
* Twitter [entites](https://developer.twitter.com/en/docs/tweets/data-dictionary/overview/entities-object) information
* The [tidyverse](https://www.tidyverse.org/) introduction.

# Searching for Tweets

## Problem

You want to collect a sample of tweets from the public timeline for a custom query.

## Solution

Use rtweet::search\_tweets() and custom [search operators](https://developer.twitter.com/en/docs/tweets/search/guides/standard-operators).

## Discussion

The Twitter API has free and paid tiers. The free tier is what many of us use and there are a number of operators that can be added to a search query to refine the results. We saw one of those in Recipe 3 by using the #rstats hashtag in the search query. But there are far more options at our disposal.

We can see all the #rstats tweets that aren’t retweets:

library(rtweet)  
library(tidyverse)

search\_tweets("#rstats -filter:retweets") %>%  
 select(text)

## # A tibble: 100 x 1  
## text  
## <chr>  
## 1 Airline passenger safety has clearly immeasurably improved under the presen  
## 2 J'aimerais que plus de profs d'économétrie lisent ceci https://t.co/xFf78u9  
## 3 The new version of tidycensus for @uscensusbureau data in #rstats (version   
## 4 I couldn't resist and I just signed up @DataCamp using this promo https://t  
## 5 What is the most efficient way to apply the same function to elements of a   
## 6 My team is growing again. Looking to hire a couple managers on a really fun  
## 7 I find it verrry difficult to read #rstats code where if statements have no  
## 8 "Microsoft SQL Server R Services - Internals XVII.\nUsing WinDbg to find ou  
## 9 "Adobe Illustrator swallows time. I spent the entire day creating the conce  
## 10 Rbloggers Helpful Data Science Reads https://t.co/QxVddTUaj7 #Rstats #DataS  
## # ... with 90 more rows

or, all the recent tweet-replies sent to @kearneymw:

search\_tweets("to:kearneymw") %>%  
 select(text)

## # A tibble: 100 x 1  
## text  
## <chr>  
## 1 @kearneymw @MattJStannard This is going to come in really handy for me in t  
## 2 @kearneymw @slcathena And bam!  
## 3 "@kearneymw @jhollist @dmi3k @ucfagls Could be worse: I brought an entire t  
## 4 @kearneymw Well, the algorithm's namesake, in that case.  
## 5 "@kearneymw His namesake died from an intestinal blockage. Where are the sh  
## 6 @kearneymw @BreitbartNews First thing's first: Breitbart is NOT 'news'. Na  
## 7 @kearneymw @Twitter interesting. This is what I see https://t.co/PVcuXp8u0r  
## 8 "@kearneymw @Grantimus9 Dang, he was right again.\n\nDebate world is small!  
## 9 @kearneymw Amazing. Indeed. @Grantimus9  
## 10 "@kearneymw Really good. Thanks. \n\n...Eating more fish"  
## # ... with 90 more rows

and, even all the #rstats tweets that have GitHub links in them (but no #python hashtags):

search\_tweets("#rstats url:github -#python") %>%   
 select(text)

## # A tibble: 100 x 1  
## text  
## <chr>  
## 1 The new version of tidycensus for @uscensusbureau data in #rstats (version   
## 2 RT @krlmlr: Soon on CRAN: Joint profiling of #rstats and native code, with   
## 3 RT @krlmlr: Soon on CRAN: Joint profiling of #rstats and native code, with   
## 4 RT @krlmlr: Soon on CRAN: Joint profiling of #rstats and native code, with   
## 5 RT @krlmlr: Soon on CRAN: Joint profiling of #rstats and native code, with   
## 6 RT @noamross: Had a little New Year's brainstorm for an @rOpenSci #rstats p  
## 7 RT @noamross: Had a little New Year's brainstorm for an @rOpenSci #rstats p  
## 8 RT @sbmalev: To all 4 of you doing fire history analysis in #rstats: burnr   
## 9 RT @kwbroman: I think my default should be to always include NAs in any sca  
## 10 RT @sbmalev: To all 4 of you doing fire history analysis in #rstats: burnr   
## # ... with 90 more rows

## See Also

* Twitter standard [search operators](https://developer.twitter.com/en/docs/tweets/search/guides/standard-operators)

# Extracting a Retweet’s Origins

## Problem

You want to extract the originating source from a retweet.

## Solution

If the tweet’s retweet\_count field is greater than 0, extract name out of the tweet’s user field; also parse the text of the tweet with a regular expression.

## Discussion

Twitter is *pretty darn good* about weaponizingutilizing the data on its platform. There aren’t many cases nowadays when you need to parse RT or via in hand-crafted retweets, but it’s good to have the tools in your aresenal when needed. We can pick out all the retweets from #rstats (warning: it’s a retweet-heavy hashtag) and who they refer to using the retweet\_count but also looking for a special [regular expression](https://stat.ethz.ch/R-manual/R-devel/library/base/html/regex.html) (regex) and extracting data that way.

First, the modern, API-centric way:

library(rtweet)  
library(tidyverse)

rstats <- search\_tweets("#rstats", n=500)  
  
glimpse(rstats)

## Observations: 500  
## Variables: 42  
## $ status\_id <chr> "948590403036569601", "9485903595247861...  
## $ created\_at <dttm> 2018-01-03 16:22:30, 2018-01-03 16:22:...  
## $ user\_id <chr> "280608149", "781583089667420161", "594...  
## $ screen\_name <chr> "cortinah", "LynnYarmey", "thibaultdej"...  
## $ text <chr> "Airline passenger safety has clearly i...  
## $ source <chr> "TweetDeck", "Twitter for iPhone", "Twi...  
## $ reply\_to\_status\_id <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,...  
## $ reply\_to\_user\_id <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,...  
## $ reply\_to\_screen\_name <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,...  
## $ is\_quote <lgl> FALSE, FALSE, FALSE, FALSE, FALSE, FALS...  
## $ is\_retweet <lgl> FALSE, TRUE, FALSE, FALSE, FALSE, TRUE,...  
## $ favorite\_count <int> 0, 0, 0, 1, 2, 0, 0, 2, 1, 0, 0, 0, 0, ...  
## $ retweet\_count <int> 0, 11, 0, 0, 2, 9, 15, 1, 0, 15, 4, 0, ...  
## $ hashtags <list> [<"rstats", "ggplot2">, "rstats", "rst...  
## $ symbols <list> [NA, NA, NA, NA, NA, NA, NA, NA, NA, N...  
## $ urls\_url <list> [NA, NA, "tidyverse.org/articles/2017/...  
## $ urls\_t.co <list> [NA, NA, "https://t.co/xFf78u9sIX", "h...  
## $ urls\_expanded\_url <list> [NA, NA, "https://www.tidyverse.org/ar...  
## $ media\_url <list> ["http://pbs.twimg.com/media/DSoOd51Wk...  
## $ media\_t.co <list> ["https://t.co/DNifWLdiVY", NA, NA, NA...  
## $ media\_expanded\_url <list> ["https://twitter.com/cortinah/status/...  
## $ media\_type <list> ["photo", NA, NA, NA, NA, NA, "photo",...  
## $ ext\_media\_url <list> ["http://pbs.twimg.com/media/DSoOd51Wk...  
## $ ext\_media\_t.co <list> ["https://t.co/DNifWLdiVY", NA, NA, NA...  
## $ ext\_media\_expanded\_url <list> ["https://twitter.com/cortinah/status/...  
## $ ext\_media\_type <lgl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,...  
## $ mentions\_user\_id <list> [NA, <"189172089", "862680370436874240...  
## $ mentions\_screen\_name <list> [NA, <"rgfitzjohn", "vaccineimpact">, ...  
## $ lang <chr> "en", "en", "fr", "en", "en", "en", "en...  
## $ quoted\_status\_id <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,...  
## $ quoted\_text <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,...  
## $ retweet\_status\_id <chr> NA, "948501344989663233", NA, NA, NA, "...  
## $ retweet\_text <chr> NA, "Last week to apply for our #rstats...  
## $ place\_url <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,...  
## $ place\_name <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,...  
## $ place\_full\_name <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,...  
## $ place\_type <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,...  
## $ country <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,...  
## $ country\_code <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,...  
## $ geo\_coords <list> [<NA, NA>, <NA, NA>, <NA, NA>, <NA, NA...  
## $ coords\_coords <list> [<NA, NA>, <NA, NA>, <NA, NA>, <NA, NA...  
## $ bbox\_coords <list> [<NA, NA, NA, NA, NA, NA, NA, NA>, <NA...

filter(rstats, retweet\_count > 0) %>%   
 select(text, mentions\_screen\_name, retweet\_count) %>%   
 mutate(text = substr(text, 1, 30)) %>%   
 unnest()

## # A tibble: 567 x 3  
## text retweet\_count  
## <chr> <int>  
## 1 RT @rgfitzjohn: Last week to a 11  
## 2 RT @rgfitzjohn: Last week to a 11  
## 3 I couldn't resist and I just s 2  
## 4 "RT @apreshill: \U0001f929 Snake case is" 9  
## 5 RT @JeroenBoeye: New post on d 15  
## 6 What is the most efficient way 1  
## 7 RT @JeroenBoeye: New post on d 15  
## 8 RT @GioraSimchoni: My very fir 4  
## 9 RT @GioraSimchoni: My very fir 4  
## 10 RT @GioraSimchoni: My very fir 4  
## # ... with 557 more rows, and 1 more variables: mentions\_screen\_name <chr>

The text column was pared down for display brevity. If you run that code snippet you can examine it to see that it identifies the retweets and the first screen name is usually the main reference, but you get all of the screen names from the original tweet for free.

Here’s the brute-force way. A regular expression is used that matches the vast majority of retweet formats. The patten looks for them then extracts the first found screen name:

# regex mod from https://stackoverflow.com/questions/655903/python-regular-expression-for-retweets  
filter(rstats, str\_detect(text, "(RT|via)((?:[[:blank:]:]\\W\*@\\w+)+)")) %>%   
 select(text, mentions\_screen\_name, retweet\_count) %>%   
 mutate(extracted = str\_match(text, "(RT|via)((?:[[:blank:]:]\\W\*@\\w+)+)")[,3]) %>%   
 mutate(text = substr(text, 1, 30)) %>%   
 unnest()

## # A tibble: 485 x 4  
## text retweet\_count  
## <chr> <int>  
## 1 RT @rgfitzjohn: Last week to a 11  
## 2 RT @rgfitzjohn: Last week to a 11  
## 3 "RT @apreshill: \U0001f929 Snake case is" 9  
## 4 RT @JeroenBoeye: New post on d 15  
## 5 RT @JeroenBoeye: New post on d 15  
## 6 RT @GioraSimchoni: My very fir 4  
## 7 RT @GioraSimchoni: My very fir 4  
## 8 RT @GioraSimchoni: My very fir 4  
## 9 RT @krlmlr: Soon on CRAN: Join 14  
## 10 RT @krlmlr: Soon on CRAN: Join 14  
## # ... with 475 more rows, and 2 more variables: extracted <chr>,  
## # mentions\_screen\_name <chr>

You should try the above snippets for other tags as there will be cases when the regex will pick up retweets Twitter has failed to capture.

## See Also

* Twiter [official documentation](https://developer.twitter.com/en/docs/tweets/post-and-engage/guides/tweet-availability) on what happens to retweets when origin tweets are deleted

# Creating a Graph of Retweet Relationships

## Problem

You want to construct and analyze a graph data structure of retweet relationships for a set of query results.

## Solution

Query for the topic, extract the retweet origins, and then use igraph to construct a graph to analyze.

## Discussion

Recipes 4 and 5 introduced and expanded on searching Twitter plus looking for retweets. The [igraph](http://igraph.org/r/) package can be used to capture and analyze details of relationships across retweets. We’ll focus on just examining the Twitter user pair relationships.

Let’s get a larger sample this time — 1,500 tweets in #rstats. We can use the technique from the previous recips and:

* find the retweets (using the API-provided data)
* expand out all the mentioned screen names
* create an igraph graph object
* look at some summary statistics for the graph

library(rtweet)  
library(igraph)  
library(hrbrthemes)  
library(tidyverse)

rstats <- search\_tweets("#rstats", n=1500)  
  
filter(rstats, retweet\_count > 0) %>%   
 select(screen\_name, mentions\_screen\_name) %>%  
 unnest(mentions\_screen\_name) %>%   
 filter(!is.na(mentions\_screen\_name)) %>%   
 graph\_from\_data\_frame() -> rt\_g

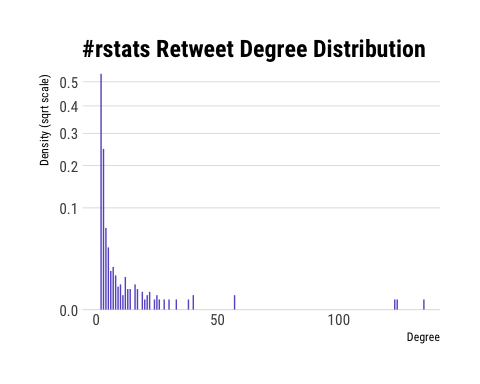
You can reference the [igraph print() and summary() functions](http://igraph.org/r/doc/print.igraph.html) for more information on the output of summary() but output from the following line shows that the graph is Directed with Named vertices and it has 959 vertices and 1,508 edges.

summary(rt\_g)

## IGRAPH ec3141a DN-- 959 1508 --   
## + attr: name (v/c)

We’ll produce more visualizations in the next recipe, but the *degree* of graph vertices is one of the most fundamental properties of a graph and it’s much nicer to see the degree distribution than stare at a wall of numbers:

ggplot(data\_frame(y=degree\_distribution(rt\_g), x=1:length(y))) +  
 geom\_segment(aes(x, y, xend=x, yend=0), color="slateblue") +  
 scale\_y\_continuous(expand=c(0,0), trans="sqrt") +  
 labs(x="Degree", y="Density (sqrt scale)", title="#rstats Retweet Degree Distribution") +  
 theme\_ipsum\_rc(grid="Y", axis="x")



## See Also

* [igraph](http://igraph.org/)

# Visualizing a Graph of Retweet Relationships

## Problem

You want to visualize a graph of retweets.

## Solution

There are a plethora of ways to visualize graph structures in R. One recent and popular one is [ggraph](https://github.com/thomasp85/ggraph).

Given the cookbook-nature of this book, we’ll cover one more visualization about retweet relationships. Let’s explore the entire retweet network and label the screen names with the most retweets over a given search term (and use #rstats again, but gather more tweets this time to truly make a spaghetti chart):

library(rtweet)  
library(igraph)  
library(hrbrthemes)  
library(ggraph)  
library(tidyverse)

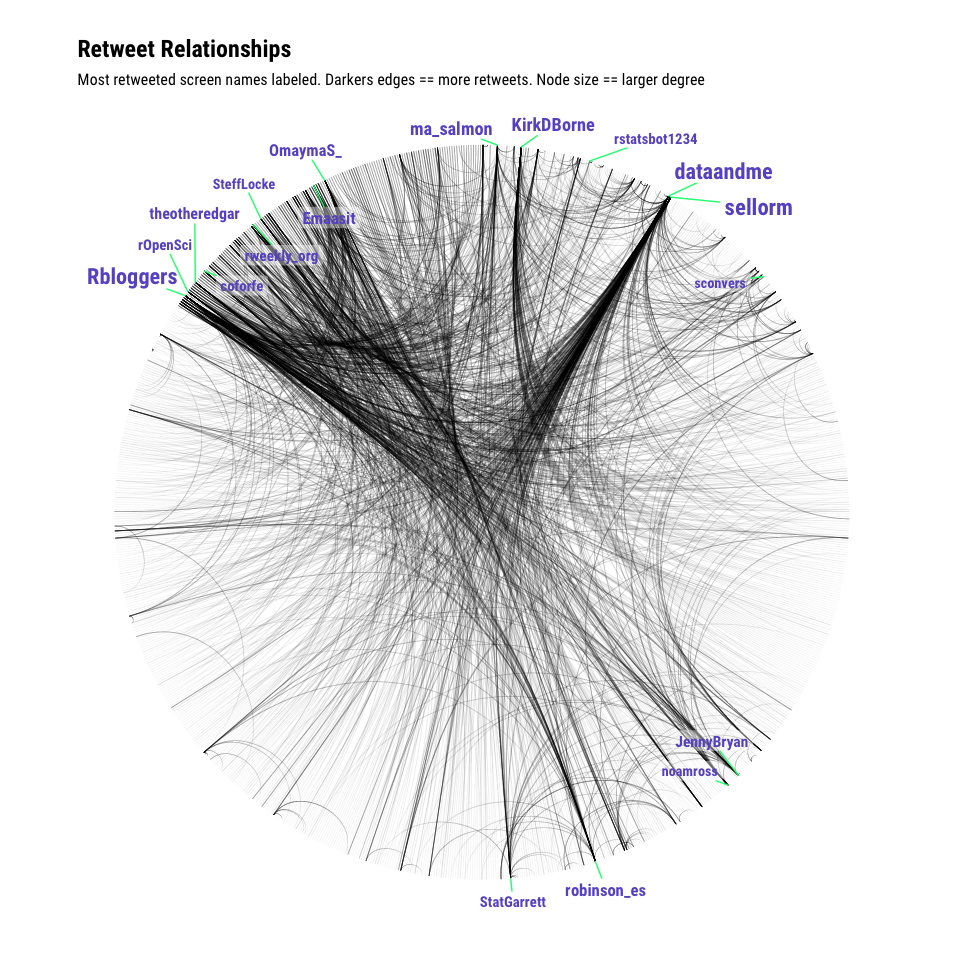
rstats <- search\_tweets("#rstats", n=1500)  
  
# same as previous recipe  
filter(rstats, retweet\_count > 0) %>%   
 select(screen\_name, mentions\_screen\_name) %>%  
 unnest(mentions\_screen\_name) %>%   
 filter(!is.na(mentions\_screen\_name)) %>%   
 graph\_from\_data\_frame() -> rt\_g

To help de-clutter the vertex labels, we’ll only add labels for nodes that have a degree of 20 or more (rough guess — you should look at the degree distribution for more formal work). We’ll also include the degree for those nodes so we can size them properly:

V(rt\_g)$node\_label <- unname(ifelse(degree(rt\_g)[V(rt\_g)] > 20, names(V(rt\_g)), ""))   
V(rt\_g)$node\_size <- unname(ifelse(degree(rt\_g)[V(rt\_g)] > 20, degree(rt\_g), 0))

Now, we’ll creatre the graph. Using ..index.. for the alpha channel will help show edge weight without too much extra effort. Note the *heavy* customization of geom\_node\_label(). Thomas made it way too easy to make beautiful network graphs with ggraph:

ggraph(rt\_g, layout = 'linear', circular = TRUE) +   
 geom\_edge\_arc(edge\_width=0.125, aes(alpha=..index..)) +  
 geom\_node\_label(aes(label=node\_label, size=node\_size),  
 label.size=0, fill="#ffffff66", segment.colour="springgreen",  
 color="slateblue", repel=TRUE, family=font\_rc, fontface="bold") +  
 coord\_fixed() +  
 scale\_size\_area(trans="sqrt") +  
 labs(title="Retweet Relationships", subtitle="Most retweeted screen names labeled. Darkers edges == more retweets. Node size == larger degree") +  
 theme\_graph(base\_family=font\_rc) +  
 theme(legend.position="none")



## See Also

* Enter twitter network analysis r into Google (seriously!). Lots of folks have worked in this space and blogged or wrote about their efforts.

# Capturing Tweets in Real-time with the Streaming API

## Problem

You want to capture a stream of public tweets in real-time, optionally filtering by select screen names or keywords in the text of the tweet.

## Solution

Use rtweet::stream\_tweets().

## Discussion

Michael has — once again — made it way too easy to work with Twitter’s API. The rtweet::stream\_tweets() function has tons of handy options to help capture tweets in real time. The primary q parameter is very versatile and has four possible capture modes:

* The default, q = "", returns a small random sample of all publicly available Twitter statuses.
* To filter by keyword, provide a comma separated character string with the desired phrase(s) and keyword(s).
* Track users by providing a comma separated list of user IDs or screen names.
* Use four latitude/longitude bounding box points to stream by geo location. This must be provided via a vector of length 4, e.g., c(-125, 26, -65, 49).

Let’s capture one minute of tweets in the good ol’ U S of A (this is one of Michael’s examples from the manual page for rtweet::stream\_tweets().

library(rtweet)  
library(tidyverse)

stream\_tweets(  
 lookup\_coords("usa"), # handy helper function in rtweet  
 verbose = FALSE,  
 timeout = (60 \* 1),  
) -> usa

##   
 Found 500 records...  
 Found 1000 records...  
 Found 1238 records...  
 Imported 1238 records. Simplifying...

A 60 second stream resulted in well over 1,000 records.

Where are they tweeting from?

count(usa, place\_full\_name, sort=TRUE)

## # A tibble: 686 x 2  
## place\_full\_name n  
## <chr> <int>  
## 1 Manhattan, NY 28  
## 2 Houston, TX 26  
## 3 Los Angeles, CA 23  
## 4 Florida, USA 22  
## 5 Chicago, IL 20  
## 6 Georgia, USA 19  
## 7 Texas, USA 15  
## 8 Toronto, Ontario 15  
## 9 Brooklyn, NY 14  
## 10 Philadelphia, PA 13  
## # ... with 676 more rows

What are they tweeting about?

unnest(usa, hashtags) %>%   
 count(hashtags, sort=TRUE) %>%   
 filter(!is.na(hashtags))

## # A tibble: 321 x 2  
## hashtags n  
## <chr> <int>  
## 1 job 73  
## 2 CareerArc 45  
## 3 Hiring 44  
## 4 hiring 32  
## 5 Job 20  
## 6 Jobs 20  
## 7 Retail 13  
## 8 WPMOYChallenge 10  
## 9 Hospitality 4  
## 10 Boston 3  
## # ... with 311 more rows

What app are they using?

count(usa, source, sort=TRUE)

## # A tibble: 27 x 2  
## source n  
## <chr> <int>  
## 1 Twitter for iPhone 818  
## 2 Twitter for Android 175  
## 3 TweetMyJOBS 91  
## 4 Instagram 58  
## 5 Twitter Web Client 40  
## 6 Twitter for iPad 14  
## 7 SafeTweet by TweetMyJOBS 8  
## 8 Foursquare 6  
## 9 Tweetbot for Mac 5  
## 10 circlepix 3  
## # ... with 17 more rows

Michael covers the streaming topic in-depth in [a vignette](http://rtweet.info/articles/stream.html).

## See Also

* [Consuming streaming data](https://developer.twitter.com/en/docs/tutorials/consuming-streaming-data)

# Making Robust Twitter Requests

## Problem

You want to write a long-running script that harvests large amounts of data, such as the friend and follower ids for a very popular Twitterer; however, the Twitter API is inherently unreliable and imposes rate limits that require you to always expect the unexpected.

## Solution

Use rtweet.

## Discussion

No code examples and not much expository in this chaper (unlike it’s Python counterpart). Michael has taken much of the pain away by having the package abstract the rate-limit issues and API wonkiness away from your code.

Having said that, you can work on making these Twitter scripts or other scripts more robust by wrapping potentially troublesome calls in purrr::safely()and testing for the result before continuing with data operations.

## See Also

* [purrr::safely()](http://purrr.tidyverse.org/reference/safely.html)

# Harvesting Tweets

## Problem

You want to harvest and store tweets from a collection of id values, or harvest entire timelines of tweets.

## Solution

Use rtweet’s timeline and status functions.

## Discussion

Recipe 2 showed how to do this with SQLite. Unlike other API’s rtweet returns a tidy data frame which makes it easy to put data into such rectangular data stores.

Rather than repeat the example, let’s take a quick look at all of the harvesting functions in rtweet:

* get\_collections: Get collections by user or status id.
* get\_favorites: Get tweets data for statuses favorited by one or more target users.
* get\_followers: Get user IDs for accounts following target user.
* get\_friends: Get user IDs of accounts followed by target user(s).
* get\_mentions: Get mentions for the authenticating user.
* get\_retweeters: Get user IDs of users who retweeted a given status.
* get\_retweets: Get the most recent retweets of a specific Twitter status
* get\_timeline: Get one or more user timelines (tweets posted by target user(s)).
* get\_timelines: Get one or more user timelines (tweets posted by target user(s)).
* lookup\_collections: Get collections by user or status id.
* lookup\_coords: Get coordinates of specified location.
* lookup\_friendships: Lookup friendship information between two specified users.
* lookup\_statuses: Get tweets data for given statuses (status IDs).
* lookup\_tweets: Get tweets data for given statuses (status IDs).
* lookup\_users: Get Twitter users data for given users (user IDs or screen names).
* search\_tweets: Get tweets data on statuses identified via search query.
* search\_tweets2: Get tweets data on statuses identified via search query.
* search\_users: Get users data on accounts identified via search query.
* stream\_tweets: Collect a live stream of Twitter data.
* stream\_tweets2: Collect a live stream of Twitter data.

One handy method for exporting this rectangular tweet data to a file format virtually any collaborator can use is rtweet::write\_as\_csv() which saves a flattened CSV (no nested column data).

# Creating a Tag Cloud from Tweet Entities

## Problem

You want to make a meaningless word cloud.

## Solution

Use harvesting techniques shown in previous recipes and pass the cloud-destined entities to an R wordcloud package.

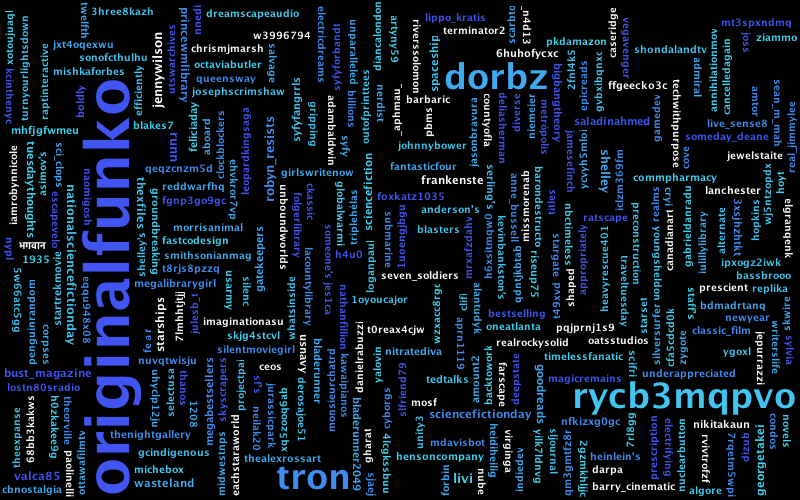
## Discussion

Word clouds are virtually devoid of meaning. Neiman Lab went to far as to call them [harmful](http://www.niemanlab.org/2011/10/word-clouds-considered-harmful/). But, this recipe is in the Python version of the book (figures, eh?) and this was desingned to be a 1:1 mapping of said book, so let’s proceed.

The folowing uses some handy text taming and word cloud packages to make a collage from #NationalScienceFictionDay tweets:

library(rtweet)  
library(tidytext)  
library(magick)  
library(kumojars) # hrbrmstr/kumojars  
library(kumo) # hrbrmstr/kumo  
library(tidyverse)

scifi <- search\_tweets("#NationalScienceFictionDay", n=1500)  
  
data\_frame(txt=str\_replace\_all(scifi$text, "#NationalScienceFictionDay", "")) %>%   
 unnest\_tokens(word, txt) %>%   
 anti\_join(stop\_words, "word") %>%   
 anti\_join(rtweet::stopwordslangs, "word") %>%   
 anti\_join(data\_frame(word=c("https", "t.co")), "word") %>% # need to make a more technical stopwords list or clean up the text better  
 filter(nchar(word)>3) %>%   
 pull(word) %>%   
 paste0(collapse=" ") -> txt  
  
cloud\_img <- word\_cloud(txt, width=800, height=500, min\_font\_size=10, max\_font\_size=60, scale="log")  
  
image\_write(cloud\_img, "data/wordcloud.png")



But, seriously, don’t make word clouds except for fun.

# Summarizing Link Targets

## Problem

You want to summarize the text of a web page that’s indicated by a short URL in a tweet.

## Solution

Extract the text from the web page, and then use a natural language processing (NLP) toolkit to help you extract the most important sentences to create a machine-generated abstract.

## Discussion

R has more than a few NLP tools to work with. We’ll work with the LSAfun package for this exercise. As the acronym-laden package name implies, it uses Latent Semantic Analysis (LSA) to determine the most important bits in a set of text.

We’ll use tweets by data journalist extraordinaire [Matt Stiles](https://twitter.com/stiles). Matt works for the Los Angeles Times and I learn a *ton* from him on a daily basis. He’s on top of *everything*. Let’s summarise some news he shared recently from the New York Times, Reuters, Washington Post, Five Thirty-Eight and his employer.

We’ll limit our exploration to the first three new links we find.

library(rtweet)  
library(LSAfun)  
library(jerichojars) # hrbrmstr/jerichojars  
library(jericho) # hrbrmstr/jericho  
library(tidyverse)

stiles <- get\_timeline("stiles")  
  
filter(stiles, str\_detect(urls\_expanded\_url, "nyti|reut|wapo|lat\\.ms|53ei")) %>% # only get tweets with news links  
 pull(urls\_expanded\_url) %>% # extract the links  
 flatten\_chr() %>% # mush them into a nice character vector  
 head(3) %>% # get the first 3  
 map\_chr(~{  
 httr::GET(.x) %>% # get the URL (I'm lazily calling "fair use" here vs check robots.txt since I'm suggesting you do this for your benefit vs profit)  
 httr::content(as="text") %>% # extract the HTML  
 jericho::html\_to\_text() %>% # strip away extraneous HTML tags  
 LSAfun::genericSummary(k=3) %>% # summarise!  
 paste0(collapse="\n\n") # easier to see  
 }) %>%  
 walk(cat)

## Continue reading the main story Advertisement Continue reading the main story The North Korea tweet near the end of the day seemed most distressing to some in Washington watching the escalating clash between the United States and a nuclear-armed North  
##   
## LEARN MORE » Sections Home Search Skip to content Skip to navigation View mobile version The New York Times Politics|Trump Says His ‘Nuclear Button’ Is ‘Much Bigger’ Than North Korea’s Search Subscribe Now Log In 0 Settings Close search Site Search Navigation Search NYTimes  
##   
## He called for an aide to Hillary Clinton to be thrown in jail, threatened to cut off aid to Pakistan and the Palestinians, assailed Democrats over immigration, claimed credit for the fact that no one died in a jet plane crash last year and announced that he would announce his own award next Monday for the most dishonest and corrupt news media We will continue to put the fairness and accuracy of everything we publish above all else — and in the inevitable moments we fall short, we will continue to own up to our mistakes, and we’ll strive to do better  
##   
## We will continue to put the fairness and accuracy of everything we publish above all else — and in the inevitable moments we fall short, we will continue to own up to our mistakes, and we’ll strive to do better  
##   
## Our report is stronger than ever, thanks to investments in new forms of journalism like interactive graphics, podcasting and digital video and even greater spending in areas like investigative, international and beat reporting Trump is the 45th president of the United States, but he has spent much of his first year in office defying the conventions and norms established by the previous 44, and transforming the presidency in ways that were once unimaginable  
##   
## Trump essentially calls it fake, making no effort to pretend to be above it all, except to boast that he is stronger, richer, smarter and more successful than anyone else  
##   
## “The hope would be that given the American people’s reaction to the way he’s handled the presidency, the people running next time will run in the opposite direction

## See Also

As noted, there are other NLP packages. Check out the [CRAN Task View](https://cran.r-project.org/web/views/NaturalLanguageProcessing.html) on NLP for more resources.

# Harvesting Friends and Followers

## Problem

You want to harvest all of the friends or followers for a particular user.

## Solution

Use rtweet::get\_followers() or rtweet::get\_friends().

## Discussion

The aforementioned rtweet functions give us all the data we need and handle pagination and rate-limits.

Let’s see who [Brooke Anderson](https://twitter.com/gbwanderson) follows and who follows her. She’s an *incredibly talented* data scientist, weather expert and educator. We’ll pull her followers and friends and work with her data a bit more in future recipes.

library(rtweet)  
library(tidyverse)

(brooke\_followers <- rtweet::get\_followers("gbwanderson"))

## # A tibble: 256 x 1  
## user\_id  
## <chr>  
## 1 913819461727195138  
## 2 25819761  
## 3 2198622000  
## 4 59655036  
## 5 769616000593428480  
## 6 2973406683  
## 7 73603242  
## 8 2790116012  
## 9 392787202  
## 10 920639877397364736  
## # ... with 246 more rows

(brooke\_friends <- rtweet::get\_friends("gbwanderson"))

## # A tibble: 103 x 2  
## user user\_id  
## <chr> <chr>  
## 1 gbwanderson 3230388598  
## 2 gbwanderson 776596392559177728  
## 3 gbwanderson 1715370056  
## 4 gbwanderson 131498466  
## 5 gbwanderson 97464922  
## 6 gbwanderson 363210621  
## 7 gbwanderson 17203405  
## 8 gbwanderson 910392773081104384  
## 9 gbwanderson 91333167  
## 10 gbwanderson 1568606814  
## # ... with 93 more rows

## See Also

* [Official Twitter API documentation](https://developer.twitter.com/en/docs/accounts-and-users/follow-search-get-users/overview) on friends and followers.

# Performing Setwise Operations on Friendship Data

## Problem

You want to operate on collections of friends and followers to answer questions such as *“Who isn’t following me back?”*, *“Who are my mutual friends?”*, and *“What friends/followers do certain users have in common?”*.

## Solution

Use R setwise operations amd rtweet::lookup\_friendships().

## Discussion

R has [set operations](https://stat.ethz.ch/R-manual/R-devel/library/base/html/sets.html) and they’ll do *just fine* for helping us cook this recipe.

If you need a refresher on set operations, check out this introductory lesson from [Khan Academy](https://www.khanacademy.org/math/statistics-probability/probability-library/basic-set-ops/v/intersection-and-union-of-sets).

library(rtweet)  
library(tidyverse)

brooke\_followers <- rtweet::get\_followers("gbwanderson")  
brooke\_friends <- rtweet::get\_friends("gbwanderson")

Now we can see the count of mutual and disperate relationships:

# common  
length(intersect(brooke\_followers$user\_id, brooke\_friends$user\_id))

## [1] 50

# diff  
length(setdiff(brooke\_followers$user\_id, brooke\_friends$user\_id))

## [1] 206

The Python counterpart to this cookbook suggests [Redis](https://redis.io/topics/data-types-intro) as a “big-ish” data solution for performing set operations at-scale. R has at least 3 packages that provide direct support for Redis, so if you need to perform these operations at-scale, cache the info you retrieve from the Twitter API into Redis and then go crazy!

## See Also

* Google (yes, seriously) redis packages r to see the impressive/diverse number of packages linking R to Redis
* [Official Twitter API documentation](https://developer.twitter.com/en/docs/accounts-and-users/follow-search-get-users/overview) on friends and followers.

# Resolving User Profile Information

## Problem

You have a collection of ids and need to resolve basic profile information (such as screen names) for these users.

## Solution

Use rtweet::lookup\_users().

## Discussion

The rtweet interface to the Twitter API makes this task very straightforward.

library(rtweet)  
library(tidyverse)

rstats <- rtweet::search\_tweets("#rstats", n=30)  
  
(recent\_rtweeters <- lookup\_users(unique(rstats$user\_id)))

## # A tibble: 27 x 20  
## user\_id name screen\_name  
## <chr> <chr> <chr>  
## 1 1217315090 Women in Statistics WomenInStat  
## 2 172176348 Heidi ideaofhappiness  
## 3 554300827 Nujcharee (เป็ด) Nujcharee  
## 4 746493721215008768 Vidhya Kalaichelvan itsmevidhya\_k  
## 5 5685812 b❆B Rudis hrbrmstr  
## 6 280608149 Hernando Cortina cortinah  
## 7 781583089667420161 Lynn Yarmey LynnYarmey  
## 8 594746010 Thibault Dejeanne thibaultdej  
## 9 144529492 Kyle Walker kyle\_e\_walker  
## 10 280455470 Tom Martens tommartens68  
## # ... with 17 more rows, and 17 more variables: location <chr>,  
## # description <chr>, url <chr>, protected <lgl>, followers\_count <int>,  
## # friends\_count <int>, listed\_count <int>, statuses\_count <int>,  
## # favourites\_count <int>, account\_created\_at <dttm>, verified <lgl>,  
## # profile\_url <chr>, profile\_expanded\_url <chr>, account\_lang <chr>,  
## # profile\_banner\_url <chr>, profile\_background\_url <chr>,  
## # profile\_image\_url <chr>

glimpse(recent\_rtweeters)

## Observations: 27  
## Variables: 20  
## $ user\_id <chr> "1217315090", "172176348", "554300827",...  
## $ name <chr> "Women in Statistics", "Heidi", "Nujcha...  
## $ screen\_name <chr> "WomenInStat", "ideaofhappiness", "Nujc...  
## $ location <chr> "USA", "North Carolina", "North Yorkshi...  
## $ description <chr> "Enticing, Elevating and Empowering the...  
## $ url <chr> "http://t.co/6ZTyTgVzrn", NA, NA, "http...  
## $ protected <lgl> FALSE, FALSE, FALSE, FALSE, FALSE, FALS...  
## $ followers\_count <int> 958, 830, 539, 48, 9139, 48, 329, 168, ...  
## $ friends\_count <int> 253, 988, 660, 98, 389, 80, 382, 151, 5...  
## $ listed\_count <int> 34, 96, 229, 30, 612, 2, 19, 8, 204, 10...  
## $ statuses\_count <int> 275, 12316, 4447, 6367, 71001, 221, 215...  
## $ favourites\_count <int> 68, 7434, 6363, 3931, 11019, 140, 3010,...  
## $ account\_created\_at <dttm> 2013-02-25 04:45:27, 2010-07-29 02:27:...  
## $ verified <lgl> FALSE, FALSE, FALSE, FALSE, TRUE, FALSE...  
## $ profile\_url <chr> "http://t.co/6ZTyTgVzrn", NA, NA, "http...  
## $ profile\_expanded\_url <chr> "http://women-in-stats.org", NA, NA, "h...  
## $ account\_lang <chr> "en", "en", "en", "en", "en", "en", "en...  
## $ profile\_banner\_url <chr> NA, "https://pbs.twimg.com/profile\_bann...  
## $ profile\_background\_url <chr> "http://abs.twimg.com/images/themes/the...  
## $ profile\_image\_url <chr> "http://pbs.twimg.com/profile\_images/33...

## See Also

* [Official Twitter API documentation](https://developer.twitter.com/en/docs/accounts-and-users/follow-search-get-users/api-reference/get-users-lookup) on users.

# Crawling Followers to Approximate Potential Influence

## Problem

You want to approximate someone’s influence based upon their popularity and the popularity of their followers.

## Solution

Use a breadth-first traversal to crawl the followers of the user to a reasonable depth, and then count the number of nodes in the graph.

## Discussion

TBD

## See Also

# Analyzing Friendship Relationships such as Friends of Friends

## Problem

You want to create a graph that facilitates the analysis of interesting relationships amongst users, such as friends of friends.

## Solution

Systematically harvest all of the friendships for users of interest, and load the data into igraph which offers native graph operations.

## Discussion

TBD

## See Also

# Analyzing Friendship Cliques

## Problem

You want to find the friendship cliques in a graph.

## Solution

Construct a graph using igraph and use its built-in functionality to for clique/community detection.

## Discussion

TBD

## See Also

# Analyzing the Authors of Tweets that Appear in Search Results

## Problem

You want to analyze user profile information as it relates to the authors of tweets that appear in search results.

## Solution

Covered in Recipe 15!

# Visualizing Geodata with a Dorling Cartogram

## Problem

You want to visualize geolocation information (for example, the location field from user profile information, included in a batch of tweets such as a search query), in order to determine if there is a correlation between location and some other criterion.

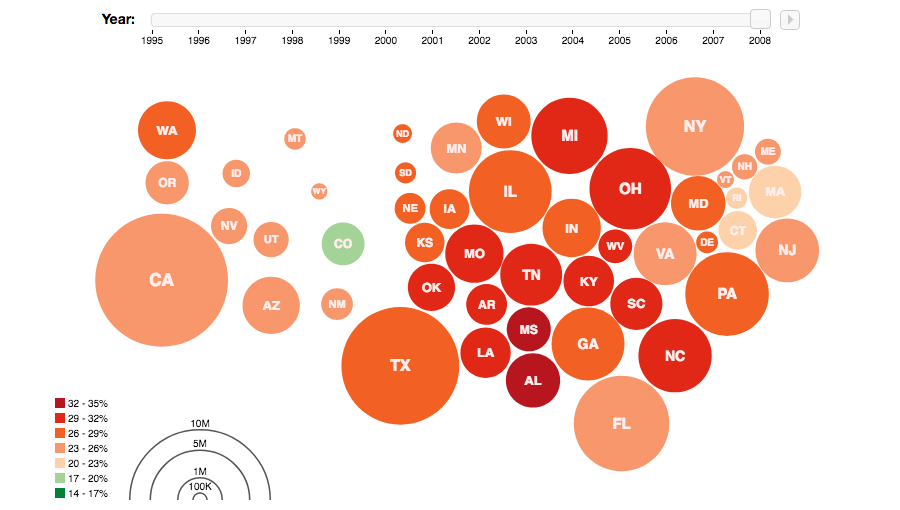
## Solution

Devise a heuristic to extract the state from the location information in user profiles and visualize it with a Dorling Cartogram.

## Discussion

(As this cookbook’s Python counterpart notes): *“A* [*Dorling Cartogram*](http://mbostock.github.io/protovis/ex/cartogram.html) *is essentially a bubble chart where each bubble corresponds to a geographic area such as a state, and each bubble is situated as close as possible to its actual location on a map without overlapping with any other bubbles. Since the size and/or color of each bubble can be used to represent meaningful things, a Dorling Cartogram can give you a very intuitive view of data as it relates to geographic boundaries or regions of a larger land mass. The Protovis toolkit comes with some machinery for creating Dorling Cartograms for locations in the United States, and one interesting example of something that you could do builds upon Recipe 19, which demonstrated an approach that you could use to analyze the users who authored tweets from a targeted query.”*

One seminal, modern example for Dorling catograms is Mike Bostock’s Protovis (a pre-cursor to D3) [version](http://mbostock.github.io/protovis/ex/cartogram.html)



I have a love/hate relationship with cartograms. On the one hand, they capture attention by preying on the weakness most of us have to maps. On the other hand, most adults in my home country barely know where their state is on a map when they can see its shape, so most cartograms require extensive labeling and the shape distortions can initially disorient the viewer. Many cartograms also degrade the ability to readily discern the proportions or precision of the underlying data. But, we’re working with Twitter data and the goal for this recipe is to pull a sample of tweets and see where the (geographic) action is at, so using this paticular cartogram style for it is not a terrible choice.

NOTE: This recipe and the next (and, final!) recipe both cover Twitter and geographic data. Any reasonable person should disable the association of location information to Tweets for a whole host of reasons which include security and safety. They should also make the location information in their profile human discernable but difficult for machines to process (better still — use fake location data in your profile). However, there are still a cadre of Twitter users who gleefully provide this information — or, disinformation — so we all can process it.

Rather than rely on the geolocation data of a tweet, let’s pull down some more #rstats tweets and try to limit the locations of the tweets to just the U.S. (despite disabling assocation of location data, Twitter can still internally, generally figure out where you’re tweeting from, especially if you’re on a mobile device).

After that, we’ll lookup the users of the tweets and extract the location information from their user profile.

Once we have that location information, we’ll filter it a bit to discern which state it came from. We’ll cover the visualization component after we deal with the data:

library(rtweet)  
library(broom)  
library(eechidna)  
library(cartogram) # chxy/cartogram  
library(hrbrthemes)  
library(tidyverse)

# search twitter for tweets  
rstats\_us <- search\_tweets("#rstats", 3000, geocode = "2.877742,-97.380979,3000mi") # geocode request isn't perfect but helps narrow down  
  
# lookup each user (uniquely) so we can grab location information  
user\_info <- lookup\_users(unique(rstats\_us$user\_id))   
  
discard(user\_info$location, `==`, "") %>% # ignore blank data  
 str\_match(sprintf("(%s)", paste0(state.abb, collapse="|"))) %>% # try to match U.S. state abbreviations  
 .[,2] %>% # the previous step creates a matrix with column 2 being the extracted information (if any)  
 discard(is.na) %>% # if no state match was found the value is NA so discard this one  
 table() %>% # some habits are hard to break  
 broom::tidy() %>% # but we can tidy them!  
 set\_names(c("state", "n")) %>% # these are more representative names  
 tbl\_df() %>% # not really necessary but I was printing this when testing  
 arrange(desc(n)) %>% # same as ^^  
 left\_join(  
 as\_data\_frame(maps::state.carto.center) %>% # join state cartographic center data  
 mutate(state=state.abb)  
 ) %>%   
 # the GitHub-only cartogram package nas a data structure which holds state adjacency information  
 # by specifying that here, it will help make the force-directed cartogram circle positioning more precise (and pretty)  
 filter(state %in% names(cartogram::statenbrs)) -> for\_dor   
  
glimpse(for\_dor)

## Observations: 37  
## Variables: 4  
## $ state <chr> "NY", "MA", "PA", "CA", "IL", "NC", "TX", "MD", "FL", "W...  
## $ n <int> 33, 28, 16, 13, 11, 11, 10, 9, 8, 8, 6, 6, 6, 6, 6, 5, 4...  
## $ x <dbl> 8.041600, 9.025060, 7.222050, 1.410693, 4.823002, 6.9985...  
## $ y <dbl> 8.746802, 8.684993, 8.029811, 7.337543, 7.757849, 6.5711...

The visualization component needs some explanation since it’s a bit hack-ish. Xiaoyue Cheng has had n R [cartogram](https://github.com/chxy/cartogram) package on GitHub for just over five years. It’s not feature complete but has a (mostly) working Dorling cartogram generator. [Carson Sievert](https://github.com/cpsievert) incorporated and enhanced some of the cartogram functions for the [rOpenSci](https://ropensci.org/) [eechidna](https://github.com/ropenscilabs/eechidna) package, but uses it internally. Finally, cartogram package has some data sets that make Dorling U.S. state cartogramsa bit more visually appealing.

What that all means is we’ll be calling an unexported function from eechidna and using some data from cartogram. This is far from an optimal situation, especially since it also means we won’t be using ggplot2 for the final visualization. But, it works!

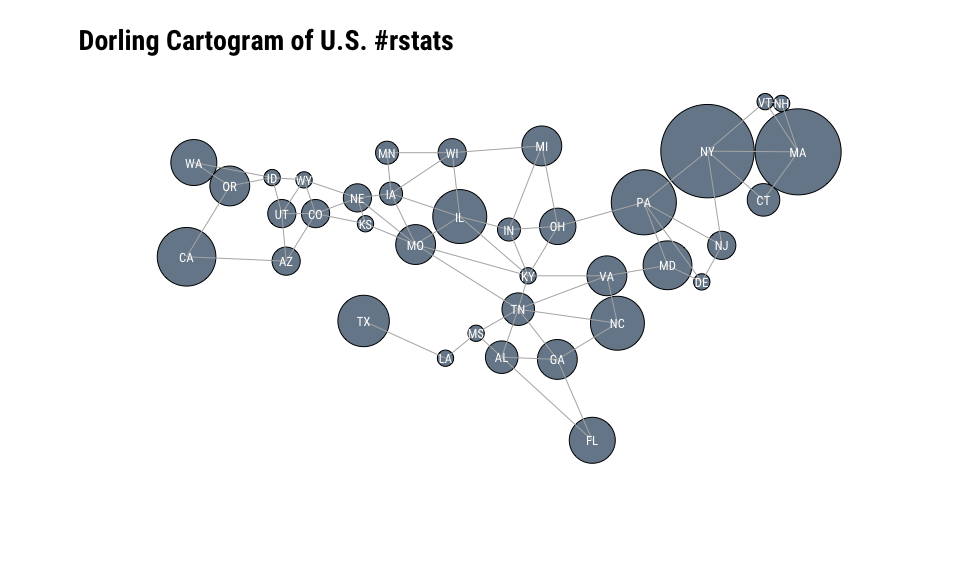
The code block below starts by setting up some base R plotting parameters, one of which — col="white" — is going to cause you some grief if you don’t remember to change it to black since it impacts the default color for base R plots. This is also an opportunity to set the font family for the text labels since there is no way to pass text label aesthetics in the dorling() function.

We pass in the:

* state label
* state center
* a scaled value for the tweet count
* state neighbor information

and tweak some aesthetics to produce the final plot. This example opted to “connect the dots” to more explicitly show neighbor information.

par(family=font\_rc, col="white")  
  
eechidna:::dorling(  
 for\_dor$state, for\_dor$x, for\_dor$y, sqrt(for\_dor$n), nbr=cartogram::statenbrs,   
 animation = FALSE, nbredge = TRUE, iteration=100, name.text=TRUE, dist.ratio=1.2,  
 main="Dorling Cartogram of U.S. #rstats", xlab='', ylab='', col="lightslategray",  
 frame=FALSE, asp=1, family=font\_rc, cex.main=1.75, adj=0  
) -> dor



# Geocoding Locations from Profiles (or Elsewhere)

## Problem

You want to geocode information in tweets for situations beyond what the Twitter API provides and not just focus on U.S. states as Ecipe 20 did.

## Solution

Use a geocoding service/package to translate location strings into more precise geographic information.

## Discussion

Recipe 20 focused on extracting U.S. state information from user profiles. But, Twitter is a global service with millions of active users in many countries. Let’s use the Google geocoding API function from the ggmaps package to try to translate user profile location strings into location data.

NOTE: Google’s API has a limit of 2,500 calls per day for free, so you’ll need to pay-up or work in daily batches if you have a large amount of Tweet location data to lookup.

library(rtweet)  
library(ggmap)  
library(tidyverse)

rstats\_us <- search\_tweets("#rstats", 3000)  
  
user\_info <- lookup\_users(unique(rstats\_us$user\_id))  
  
discard(user\_info$location, `==`, "") %>%   
 ggmap::geocode() -> coded  
  
coded$location <- discard(user\_info$location, `==`, "")  
  
user\_info <- left\_join(user\_info, coded, "location")

## # A tibble: 503 x 3  
## location lat lon  
## <chr> <dbl> <dbl>  
## 1 Peru -9.189967 -75.015152  
## 2 Richmond, B.C., Canada 49.166590 -123.133569  
## 3 Massachusetts 42.407211 -71.382437  
## 4 Frederick, MD 39.414269 -77.410541  
## 5 Japan 36.204824 138.252924  
## 6 FMU 34.190425 -79.651985  
## 7 Chicago, IL 41.878114 -87.629798  
## 8 日本 36.204824 138.252924  
## 9 北大・環境科学 43.072764 141.346112  
## 10 Stuttgart, Germany 48.775846 9.182932  
## 11 New York, NY 40.712775 -74.005973  
## 12 Asbury Park, NJ 40.220391 -74.012082  
## 13 Ann Arbor, MI 42.280826 -83.743038  
## 14 Ithaca, NY 42.443961 -76.501881  
## 15 ÜT: 36.1573208,-95.9526115 40.532392 -112.298984  
## 16 Houston, TX 29.760427 -95.369803  
## 17 Rome, NY 43.212847 -75.455730  
## 18 Perth, Australia -31.950527 115.860457  
## 19 Santiago, CL -33.448890 -70.669265  
## 20 Johnston, IA 41.670983 -93.713049  
## 21 Fort Collins, CO 40.585260 -105.084423  
## 22 Hyderabad, India 17.385044 78.486671  
## 23 Nashville, TN 36.162664 -86.781602  
## 24 Canton, CHN 23.129110 113.264385  
## 25 Bogotá 4.710989 -74.072092  
## 26 3052, Australia -37.786236 144.947418  
## 27 Charlottesville, VA 38.029306 -78.476678  
## 28 Hobart, Tasmania -42.882138 147.327195  
## 29 moon 40.516977 -80.221348  
## 30 Toronto, Ontario 43.653226 -79.383184  
## # ... with 473 more rows

## See Also

Google’s API is far from perfect, but they have also been collecting gnarly input data for map locations for over a decade, which makes them a good first-choice. You can find more R geocoding packages in the CRAN [Web Technologies](https://cran.r-project.org/web/views/WebTechnologies.html) Task View.