

Most users have heard of **ggplot2** and **dplyr**, but there are thousands of other great packages waiting to be discovered. In order to highlight those I took a list of all packages on CRAN and filtered it down to ones with anywhere between 5,000 and 1,000,000 CRAN downloads. I then took a subset of that with anywhere between 10 and 1,000 stars on GitHub. I decided to focus on these mildly popular packages with source code hosted on GitHub so that I can embed the link in the tweet and promote packages with code that people have already started to explore. After following those subsetting rules, one package is selected at random after narrowing to a list of packages that haven’t already been tweeted. I am hosting the bot on Heroku and using Heroku Scheduler to send out a tweet each day at 10:30am UTC or 6:30am Eastern Time. Below the credits and resources is the sloppily written Python that’s currently being hosted on Heroku and executed.

**Credits**

However, there were plenty of Python based Twitter bot tutorials, so I took full advantage of them and went the Python route. Below is the host of resources I considered while figuring out how to deploy the app, what Twitter package to use, and some basic Python syntax which, embarrassingly, I should know by now.

Make a Twitter Bot with R

# Load packages

First, we’ll need to install and load some packages. We’ll get the most recent version of twitteR from github.

devtools::install\_github("geoffjentry/twitteR")

## Downloading github repo geoffjentry/twitteR@master

## Installing twitteR

## "C:/R-3.1.2/bin/x64/R" --vanilla CMD INSTALL \

## "C:\Users\Mike\AppData\Local\Temp\RtmpgJoXqD\devtools281053ed2576\geoffjentry-twitteR-964f2d0" \

## --library="C:/Users/Mike/Documents/R/win-library/3.1" --install-tests

##

## Reloading installed twitteR

**library**(twitteR)

**library**(ROAuth)

# Set-up text bits

Now, we’ll set up some text strings with different adjectives, nouns, and hashtags. These are the raw materials for assembling a new Mexican food.

adjectives1 <- c("Cheesy","Beefy","Spicy","Santa Fe","Tex Mex", "Mexican",

"Stuffed","Crunchy","Grilled",

"Cool Ranch","Spicy Nacho","Creamy","Chipotle","Double","Triple")

adjectives2<- c("Cheesy","Beefy","Spicy","Santa Fe","Tex Mex", "Mexican",

"Stuffed","Crunchy","Grilled","Chicken","Beef",

"Cool Ranch","Spicy Nacho","Creamy","Chipotle","Double","Triple", "Steak",

"Jalapeno")

nouns <- c("Taco","Burrito","Quesadilla","Blaster","Slammer","Slider",

"Popper","Bowl","Gorditta","Nachos","Saltisa","Carnitas","Pizza","Chalupa")

hashtags <- c("#tacos","#TacoTuesday","#Tacos4Life","#tacosauce","#tacotime","#tacosrule")

# Set-up Twitter authentication

There are number of different tutorials out there for getting Twitter Authenticated with R. You will need to [create a new App](https://apps.twitter.com/) and follow the instructions [here](http://www.r-datacollection.com/blog/How-to-conduct-a-tombola-with-R/) or [here](http://thinktostart.com/twitter-authentification-with-r/) or [here](http://data-mining.business-intelligence.uoc.edu/home/data-from-twitter/r-oauth-for-twitter).

api\_key <- "Insert\_yours\_here"

api\_secret <- "Insert\_yours\_here"

access\_token <- "Insert\_yours\_here"

access\_token\_secret <- "Insert\_yours\_here"

setup\_twitter\_oauth(api\_key,api\_secret,access\_token,access\_token\_secret)

# Generate the tweet

First, we’ll randomly sample the number of nouns and adjectives to have in the tweet.

numb\_adjectives1 <- sample(seq(0,2,1),size=1)

numb\_adjectives2 <- sample(seq(0,1,1),size=1)

numb\_nouns <- sample(seq(1,2,1),size=1)

Next, we’ll go ahead and sample that number.

*# randomly choose your first adjectives*

random\_adjectives1 <- NULL

**for**(i **in** 1:numb\_adjectives1){

random\_adjectives1 <- c(random\_adjectives1,sample(adjectives1,size=1))

}

*# randomly choose your second adjectives*

random\_adjectives2 <- NULL

**for**(i **in** 1:numb\_adjectives2){

random\_adjectives2 <- c(random\_adjectives2,sample(adjectives2,size=1))

}

*# randomly choose your nouns*

random\_nouns <- NULL

**for**(i **in** 1:numb\_nouns){

random\_nouns <- c(random\_nouns,sample(nouns,size=1))

}

*# randomly choose a hashtag*

random\_hashtag <- sample(hashtags,size=1)

Now, fix any instances where you’ve sampled the same noun twice

**if**(length(random\_nouns)==2){

**if**(random\_nouns[1]==random\_nouns[2]){

random\_nouns <- random\_nouns[1]

}

}

Finally, combine all of those elements into one string:

temp <- c(random\_adjectives1,random\_nouns,random\_hashtag)

tweettxt <- paste(temp,collapse=" ")

tweettxt

## [1] "Cheesy Blaster #tacos"

# Send the tweet

This is all it takes to send out the tweet

tweet(tweettxt)

# Create log entry

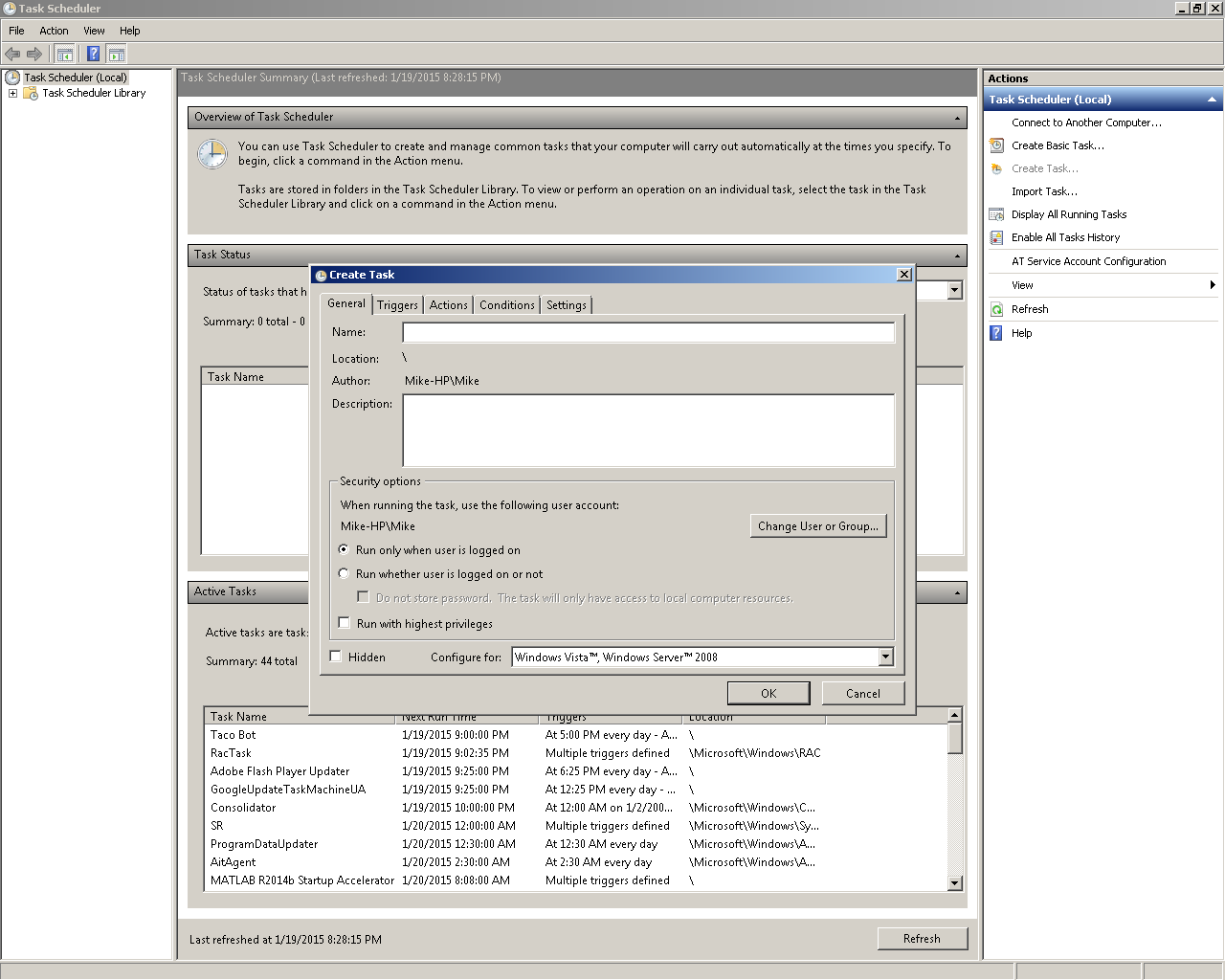
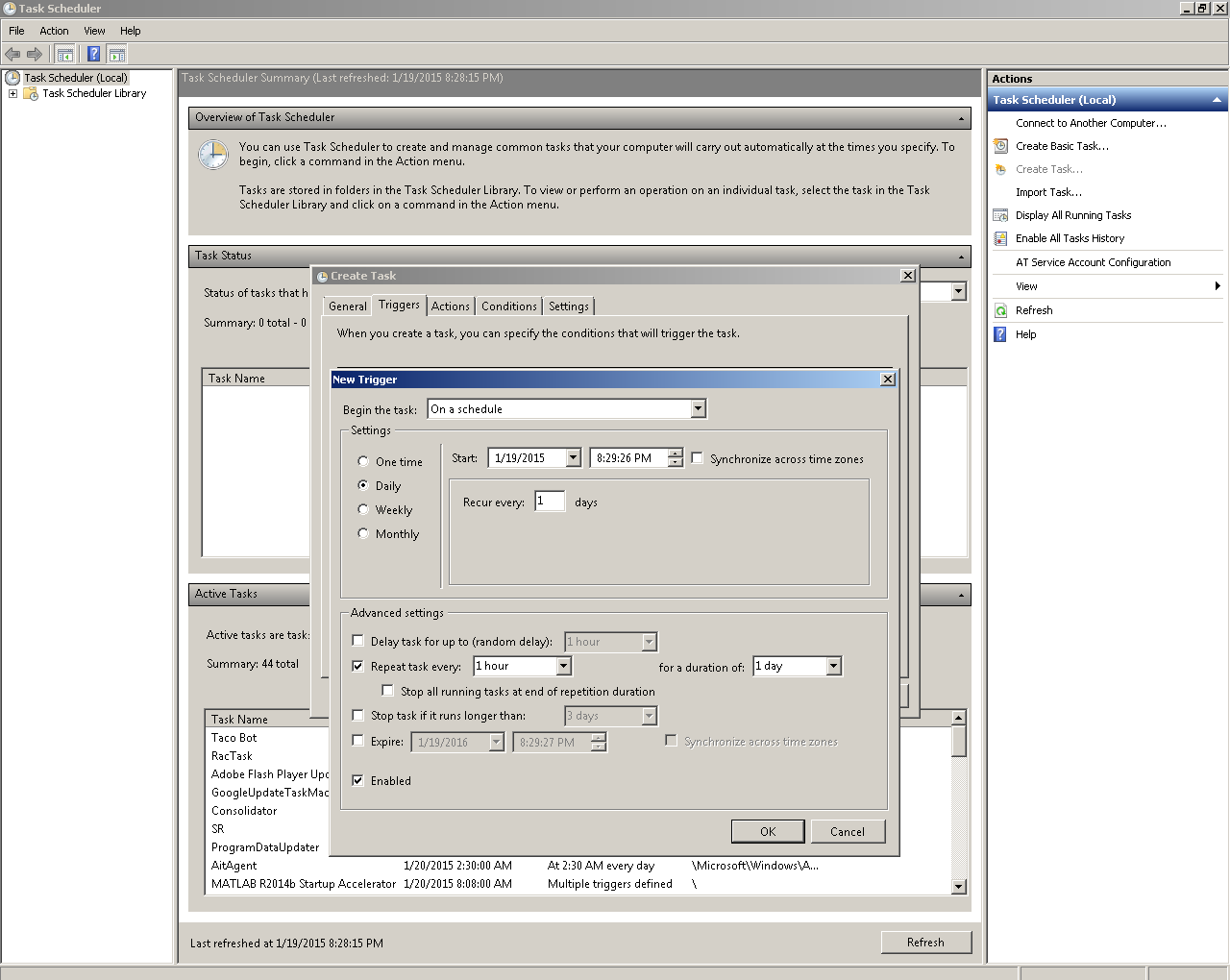
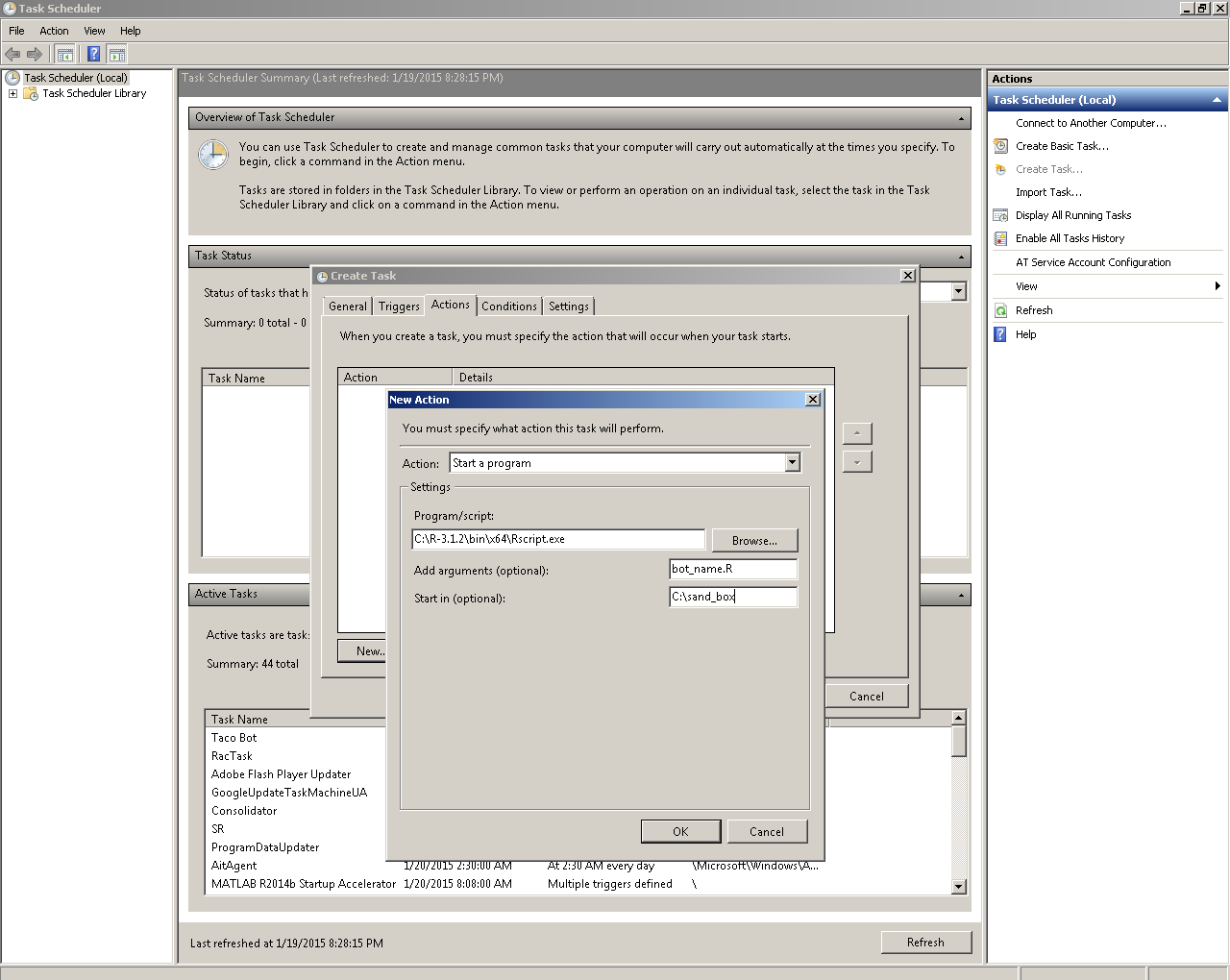
And we can even keep a record of all of our tweets

line <- paste( as.character(Sys.time()), tweettxt,sep="\t" )

write(line, file="tweets.log", append=TRUE)

# Automating the tweets

Now, this thing needs to get automated. Since I’m running Windows, we can use **Task Scheduler**.

Open up Task Scheduler. Click on Action, Create Task and give your task a name:   
  
Then, under Triggers pick your scheduling for the tweet. For example, you can make it daily, weekly, or monthly. You can also repeat the task every few minutes or hourly.   
  
Finally, under Actions you need to locate the file Rscript.exe for your installation of R. Mine was under “C:-3.1.2 bin x64 Rscript.exe”. For Add Arguments choose the name of the R script where all of this code is saved. Finally, for Start in specify the directory for that script. 

Create the R-Program to connect to twitter

#Install and load the appropriate packages  
install.packages("twitteR")  
library("twitteR")  
  
#REPLACE '####' with the appropriate values from your twitter app  
consumerKey='####'  
consumerSecret='####'  
accessToken='####'  
accessTokenSecret= '####'  
  
#Connect to twitter  
setup\_twitter\_oauth(consumerKey,consumerSecret,accessToken,accessTokenSecret)  
  
#Extract Tweets !  
searchTwitter("nba", n=3, lang="en")  
  
#Post Tweet !  
tweet("A tweet!")

On a traditional host or when working locally, you often set environment variables in your .bashrc file. On Heroku, you use **config vars**.

## [Managing config vars](https://devcenter.heroku.com/articles/config-vars#managing-config-vars)

Config var values are persistent–they remain in place across deploys and app restarts. Unless you need to change a value, you only need to set it once.

### [**Using the Heroku CLI**](https://devcenter.heroku.com/articles/config-vars#using-the-heroku-cli)

The heroku config commands of the Heroku CLI makes it easy to manage your app’s config vars.

#### [View current config var values](https://devcenter.heroku.com/articles/config-vars#view-current-config-var-values)

$ heroku config

GITHUB\_USERNAME: joesmith

OTHER\_VAR: production

$ heroku config:get GITHUB\_USERNAME

joesmith

#### [Set a config var](https://devcenter.heroku.com/articles/config-vars#set-a-config-var)

$ heroku config:set GITHUB\_USERNAME=joesmith

Adding config vars and restarting myapp... done, v12

GITHUB\_USERNAME: joesmith

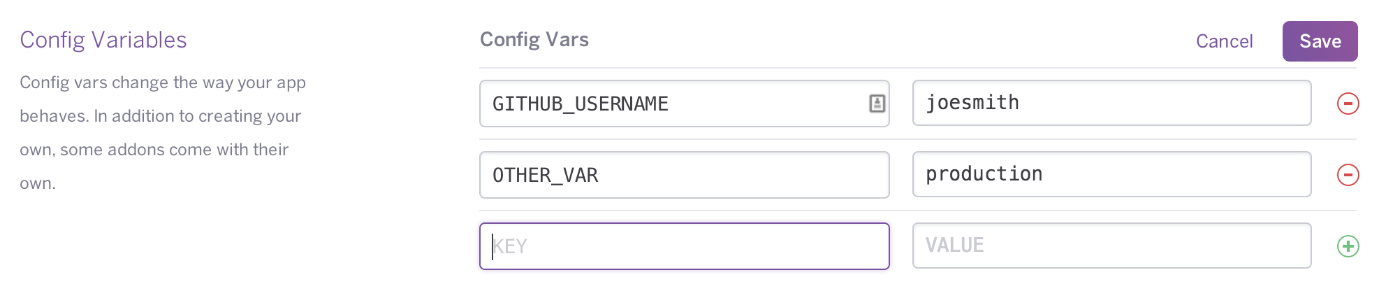
#### [Remove a config var](https://devcenter.heroku.com/articles/config-vars#remove-a-config-var)

$ heroku config:unset GITHUB\_USERNAME

Unsetting GITHUB\_USERNAME and restarting myapp... done, v13

### [**Using the Heroku Dashboard**](https://devcenter.heroku.com/articles/config-vars#using-the-heroku-dashboard)

You can also edit config vars from your app’s **Settings** tab in the Heroku Dashboard:



### [**Using the Platform API**](https://devcenter.heroku.com/articles/config-vars#using-the-platform-api)

You need a valid Heroku access token representing a user with proper permissions on the app.

## [Accessing config var values from code](https://devcenter.heroku.com/articles/config-vars#accessing-config-var-values-from-code)

Config vars are exposed to your app’s code as environment variables. For example, in Node.js you can access your app’s DATABASE\_URL config var with process.env.DATABASE\_URL.

### [**Examples**](https://devcenter.heroku.com/articles/config-vars#examples)

Add some config vars for your S3 account keys:

$ cd myapp

$ heroku config:set S3\_KEY=8N029N81 S3\_SECRET=9s83109d3+583493190

Setting config vars and restarting myapp... done, v14

S3\_KEY: 8N029N81

S3\_SECRET: 9s83109d3+583493190

Set up your code to read the vars at runtime. For example, in Ruby you access the environment variables using the ENV['KEY'] pattern - so now you can write an initializer like so:

AWS::S3::Base.establish\_connection!(

:access\_key\_id => ENV['S3\_KEY'],

:secret\_access\_key => ENV['S3\_SECRET']

)

In Node.js, use process.env to access environment variables:

const aws = require('aws-sdk');

let s3 = new aws.S3({

accessKeyId: process.env.S3\_KEY,

secretAccessKey: process.env.S3\_SECRET

});

In Java, you can access it through calls to System.getenv('key'), like so:

S3Handler = new S3Handler(System.getenv("S3\_KEY"), System.getenv("S3\_SECRET"))

In Python, using the [boto library](http://boto.cloudhackers.com/en/latest/s3_tut.html):

from boto.s3.connection import S3Connection

s3 = S3Connection(os.environ['S3\_KEY'], os.environ['S3\_SECRET'])

Now, upon deploying to Heroku, the app will use the keys set in the config.

## [Config var policies](https://devcenter.heroku.com/articles/config-vars#config-var-policies)

* Config var keys should use only alphanumeric characters and the underscore character (\_) to ensure that they are accessible from all programming languages. Config var keys should not include the hyphen character.
* Config var data (the combination of all keys and values) cannot exceed 32kb for each app.
* Config var keys should not begin with a double underscore (\_\_).
* A config var’s key should not begin with HEROKU\_ unless it is set by the Heroku platform itself.

**Full Script**

# script.py

from os import environ

from os.path import join, dirname

from dotenv import load\_dotenv

from re import sub

import pandas

from TwitterAPI import TwitterAPI, TwitterPager

# create .env file path

dotenv\_path = join(dirname(\_\_file\_\_), '.env')

# load file from the path

load\_dotenv(dotenv\_path)

if \_\_name\_\_ == "\_\_main\_\_":

# connect to api

api = TwitterAPI(consumer\_key=environ['TWITTER\_CONSUMER\_KEY'],

consumer\_secret=environ['TWITTER\_CONSUMER\_SECRET'],

access\_token\_key=environ['TWITTER\_ACCESS\_TOKEN'],

access\_token\_secret=environ['TWITTER\_ACCESS\_TOKEN\_SECRET'])

# scrape all prior tweets to check which packages I've already tweeted about

SCREEN\_NAME = 'RLangPackage'

pager = TwitterPager(api,

'statuses/user\_timeline',

{'screen\_name': SCREEN\_NAME, 'count': 100})

# parse out the package name that occurs before the hyphen at the beginning

previous\_pks = []

for item in pager.get\_iterator(wait=3.5):

if 'text' in item:

this\_pkg = sub("^(\w+) - (.\*)", "\\1", item['text'])

previous\_pks.append(this\_pkg)

# convert the package names to a dataframe

prev\_df = pandas.DataFrame({'name': previous\_pks})

prev\_df.set\_index('name')

# load the data I've compiled on R packages

url = "https://raw.githubusercontent.com/StevenMMortimer/one-r-package-a-day/master/r-package-star-download-data.csv"

all\_df = pandas.read\_csv(url)

all\_df.set\_index('name')

# do an "anti join" to throw away previously tweeted rows

all\_df = pandas.merge(all\_df, prev\_df, how='outer', indicator=True)

all\_df = all\_df[all\_df['\_merge'] == 'left\_only']

# focus on packages in middle ground of downloads and stars

filtered\_df = all\_df[all\_df['stars'].notnull()]

filtered\_df = filtered\_df[filtered\_df['stars'].between(10,1000)]

filtered\_df = filtered\_df[filtered\_df['downloads'].notnull()]

filtered\_df = filtered\_df[filtered\_df['downloads'].between(5000, 1000000)]

# randomly select one of the remaining rows

selected\_pkg = filtered\_df.sample(1)

# pull out the name and description to see if we need to

# truncate because of Twitters 280 character limit

prepped\_name = selected\_pkg.iloc[0]['name']

prepped\_desc = sub('\s+', ' ', selected\_pkg.iloc[0]['description'])

name\_len = len(prepped\_name)

desc\_len = len(prepped\_desc)

# 280 minus 3 for " - ", then minus 23 because links are counted as such,

# then minus 9 for the " #rstats " hashtag

if desc\_len <= (280-3-23-9-name\_len):

prepped\_desc = prepped\_desc[0:desc\_len]

else:

prepped\_desc = prepped\_desc[0:(280-6-23-9-name\_len)] + "..."

# cobble together the tweet text

TWEET\_TEXT = prepped\_name + " - " + prepped\_desc + " #rstats " + selected\_pkg.iloc[0]['github\_url']

print(TWEET\_TEXT)

# tweet it out to the world!

r = api.request('statuses/update', {'status': TWEET\_TEXT})

print('SUCCESS' if r.status\_code == 200 else 'PROBLEM: ' + r.text)