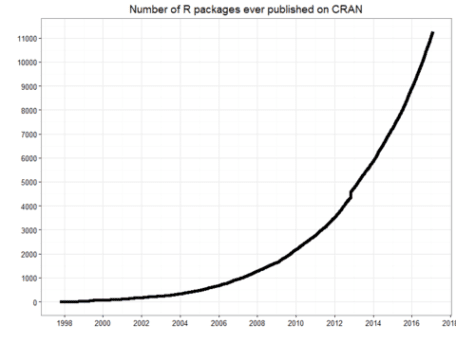
Discover one R package a day by following the [@RLangPackage](https://twitter.com/RLangPackage) account on Twitter! Inspired by One R Tip Day ([@RLangTip](https://twitter.com/RLangTip)) I created a Twitter bot that will feature one package with its description and GitHub URL each day. The R programming language (referred to as [#rstats](https://twitter.com/hashtag/rstats) on Twitter) now has over 12,600 packages. Tweeting one each day would take more than 34 years to get through them all and that doesn’t even consider the rapid growth of new packages as shown in this post.



The purpose of the [@RLangPackage](https://twitter.com/RLangPackage) Twitter account is to feature the diversity of packages that R has to offer. 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**

I was found a couple different blog posts on creating a Twitter bot with R, namely [here](https://medium.com/@randerson112358/create-a-twitter-bot-using-r-5a94f1b1b886) and [here](https://rstudio-pubs-static.s3.amazonaws.com/54352_3afcd597884a4192b3361513f2d32699.html), but they didn’t involve deploying on Heroku or another cloud service. 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. I think most users will gravitate towards the **tweepy** Python library but I found that it throws up an error when using it with Conda 3.6.

**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)