**Roslyn Walker, Student number 45149631**

Learning Journal

FOAR705 Digital Humanities

Week 9 11 October 2019

Week 7 and 8.

Due on 4 October 2019. Learning Journal, Weeks 7-8, Proof of Concept work and Data Carpentry Open Refine, Due week 8 - Part 1

ASSESSMENT CRITERIA

* Acquire core technological skills needed to produce transparent and reproducible research.
* Identify existing digital tools and approaches and apply them to your research.

=======================================================

Contents

[(1) Technology deployment--testing 1](#_Toc21018792)

[(2) Data carpentry exercise notes 3](#_Toc21018793)

======================================================

## (1) Technology deployment—testing

**Objective: install TwitteR on Rstudio on my laptop**

Note: currently installed on desktop but not on laptop

Check installation across both, not operational on laptop.

Steps:

> install.packages("twitterR")

Warning in install.packages :

package ‘twitterR’ is not available (for R version 3.6.1)

Error: install R version 3.6

Steps: go back to setup datacarpentry

Follow install steps below for twitteR

**Result:** completed. Direct authentication. [1] "Using direct authentication"

> library(devtools)

Loading required package: usethis

> install\_github("geoffjentry/twitteR")

Downloading GitHub repo geoffjentry/twitteR@master

✔ checking for file ‘/private/var/folders/23/mg11wv994z38w3gz\_rybdvlh0000gn/T/RtmpCVUcTb/remotes21a25e212d1/geoffjentry-twitteR-815fecc/DESCRIPTION’ ...

─ preparing ‘twitteR’:

✔ checking DESCRIPTION meta-information ...

─ checking for LF line-endings in source and make files and shell scripts

─ checking for empty or unneeded directories

─ building ‘twitteR\_1.1.9.tar.gz’

\* installing \*source\* package ‘twitteR’ ...

\*\* using staged installation

\*\* R

\*\* inst

\*\* byte-compile and prepare package for lazy loading

\*\* help

\*\*\* installing help indices

\*\* building package indices

\*\* testing if installed package can be loaded from temporary location

\*\* testing if installed package can be loaded from final location

\*\* testing if installed package keeps a record of temporary installation path

\* DONE (twitteR)

> library(twitteR)

> setup\_twitter\_oauth(consumer\_key, consumer\_secret, access\_token, access\_secret)

[1] "Using direct authentication"

Use a local file ('.httr-oauth'), to cache OAuth access credentials between R sessions?

1: Yes

2: No

Selection: Yes

Adding .httr-oauth to .gitignore

Error in check\_twitter\_oauth() : OAuth authentication error:

This most likely means that you have incorrectly called setup\_twitter\_oauth()'

> setup\_twitter\_oauth(consumer\_key, consumer\_secret, access\_token, access\_secret)

[1] "Using direct authentication"

Error in check\_twitter\_oauth() : OAuth authentication error:

This most likely means that you have incorrectly called setup\_twitter\_oauth()'

>

**Objective: understand next steps for Proof of concept.**

Resource: Question to Brian.

Steps:

(1) using resource: <http://geoffjentry.hexdump.org/twitteR.pdf>

(2) meeting with Brian on Wednesday, AHH Level 2, room 2.606. Call x7084 in lobby and I'll wave to you

(3) notes from discussion in Slack “there are  two authentication routes, right above section 4. run sample code on page 3

Other:

* <https://blog.trello.com/github-and-trello-integrate-your-commits> - github
* <https://gitlab.com/kathrynee/ingredients-analysis> -- example work
* <https://stackoverflow.com/a/37739112/263449> -- workflow
* <https://www.nature.com/articles/d41586-019-02619-z> workflow
* <https://www.grahamcluley.com/create-robust-data-backup-plan-make-sure-works/> -- backup plans see week 7 assignment
* <http://www.lastpass.com/> password storage
* <https://lastpass.com/yubico>
* <http://gum.co/bite-size-command-line> [https://pbs.twimg.com/media/DjkXh-OX0AADQNA.jpg](https://slack-redir.net/link?url=https%3A%2F%2Fpbs.twimg.com%2Fmedia%2FDjkXh-OX0AADQNA.jpg) -- command line unix

**Objective: generate a bearer token:**

Steps: Use curl command in command line to create token.

Resource: <https://developer.twitter.com/en/docs/basics/authentication/guides/bearer-tokens>

Example:

Below is a curl command that you can run from your command line to create a bearer token:

curl -u 'API key:API secret key' \

--data 'grant\_type=client\_credentials' \

'https://api.twitter.com/oauth2/token'

Be sure to replace API key and API secret key with your alphanumeric consumer API key and API secret key. For this example the result you would get back would be as follows:

{"token\_type":"bearer","access\_token":"bearer token"}

Here is what the response would look like with a decommissioned token:

{"token\_type":"bearer","access\_token":"AAAAAAAAAAAAAAAAAAAAAMLheAAAAAAA0%2BuSeid%2BULvsea4JtiGRiSDSJSI%3DEUifiRBkKG5E2XzMDjRfl76ZC9Ub0wnz4XsNiRVBChTYbJcE3F"}

The bearer token in this case that you would use to authenticate to resources with would be:

AAAAAAAAAAAAAAAAAAAAAMLheAAAAAAA0%2BuSeid%2BULvsea4JtiGRiSDSJSI%3DEUifiRBkKG5E2XzMDjRfl76ZC9Ub0wnz4XsNiRVBChTYbJcE3F

Steps:

In terminal:

(base) Roslyns-iMac:~ Tquestudio$

(base) Roslyns-iMac:~ Tquestudio$ curl -u 'uH2z9vskA7YuRhYEfNSPVfIsX:vRtkK1yaygSfELR1hk2ajLmyhYyIDnCRiTENPROviIhLwfPJQg' \

> --data 'grant\_type=client\_credentials' \

> 'https://api.twitter.com/oauth2/token'

Result:

{"token\_type":"bearer","access\_token":"AAAAAAAAAAAAAAAAAAAAACpJAQEAAAAAdB9lz%2FPlABc3U1GofoiPROW5MM8%3DABFVmReaNWplI8xrisaA4CIAcfy9ANrSN7LxGwDZJhZy0oDHnS"}(base) Roslyns-iMac:~ Tquestudio$

Objective: Lesson: exploring Twitter, searching Twitter with R.

Resource: <http://geoffjentry.hexdump.org/twitteR.pdf>

Steps. Look at <https://developer.twitter.com/en/docs>

referred to: <https://developer.twitter.com/en/docs/tweets/search/api-reference/get-search-tweets> and search tweets: standard search API

Note: To learn how to use [Twitter Search](https://twitter.com/search) effectively, please see the [Standard search operators](https://developer.twitter.com/en/docs/tweets/search/guides/standard-operators) page for a list of available filter operators. Also, see the [Working with Timelines](https://developer.twitter.com/en/docs/tweets/timelines/guides/working-with-timelines) page to learn best practices for navigating results by since\_id and max\_id.

Resource.

Steps: https://developer.twitter.com/en/docs/tweets/search/guides/standard-operators

Url: <https://twitter.com/search?q=(%23ClimateStrike)%20(to%3Asent%20OR%20to%3Ain%20OR%20to%3Areply%20OR%20to%3Ato%20OR%20to%3AGretaThunberg)%20lang%3Aen%20until%3A2019-09-24%20since%3A2019-09-17&src=typed_query&f=live>

Replace:

Replace “https://twitter.com/search” with “https://api.twitter.com/1.1/search/tweets.json

[https://api.twitter.com/1.1/search/tweets.json?q=(%23ClimateStrike)%20(to%3Asent%20OR%20to%3Ain%20OR%20to%3Areply%20OR%20to%3Ato%20OR%20to%3AGretaThunberg)%20lang%3Aen%20until%3A2019-09-24%20since%3A2019-09-17&src=typed\_query&f=live](https://twitter.com/search?q=(%23ClimateStrike)%20(to%3Asent%20OR%20to%3Ain%20OR%20to%3Areply%20OR%20to%3Ato%20OR%20to%3AGretaThunberg)%20lang%3Aen%20until%3A2019-09-24%20since%3A2019-09-17&src=typed_query&f=live)

Error: {"errors":[{"code":215,"message":"Bad Authentication data."}]}

I don’t understand the application of this for a result.

**Objective Lesson: Towards data science: on Twitter API**

Resource: <https://towardsdatascience.com/access-data-from-twitter-api-using-r-and-or-python-b8ac342d3efe>

Notes: R If you want to use R, you can use twitteR (make sure you install first install the twitteR package). twitteR is an R package which provides access to the Twitter API. Most functionality of the API is supported, with a bias towards API calls that are more useful in data analysis as opposed to daily interaction. Read the user vignette if you want to learn more about how to use the package for your various needs. The code below uses the Twitter search API.

Steps:

1. in R studio, command install.packages("twitteR")

Result: The downloaded binary packages are in

/var/folders/19/t871x47j1qz\_fzpmbpksysgh0000gn/T//Rtmped3IXJ/downloaded\_packages

2. Set consumer keys.

consumer\_key <- ""

consumer\_secret <-""

access\_token <- ""

access\_secret <- ""

> library(twitteR)

Error in library(twitteR) : there is no package called ‘twitteR’

> install.packages("twitteR")

also installing the dependencies ‘bit’, ‘bit64’, ‘rjson’

trying URL 'https://cran.rstudio.com/bin/macosx/el-capitan/contrib/3.6/bit\_1.1-14.tgz'

Content type 'application/x-gzip' length 254990 bytes (249 KB)

==================================================

downloaded 249 KB

trying URL 'https://cran.rstudio.com/bin/macosx/el-capitan/contrib/3.6/bit64\_0.9-7.tgz'

Content type 'application/x-gzip' length 559670 bytes (546 KB)

==================================================

downloaded 546 KB

trying URL 'https://cran.rstudio.com/bin/macosx/el-capitan/contrib/3.6/rjson\_0.2.20.tgz'

Content type 'application/x-gzip' length 192410 bytes (187 KB)

==================================================

downloaded 187 KB

trying URL 'https://cran.rstudio.com/bin/macosx/el-capitan/contrib/3.6/twitteR\_1.1.9.tgz'

Content type 'application/x-gzip' length 537986 bytes (525 KB)

==================================================

downloaded 525 KB

The downloaded binary packages are in

/var/folders/19/t871x47j1qz\_fzpmbpksysgh0000gn/T//Rtmped3IXJ/downloaded\_packages

> consumer\_key <- " #value removed

> consumer\_secret <- "#value removed

> access\_token <- "#value removed

> access\_secret <- “ #value removed

> setup\_twitter\_oauth(consumer\_key, consumer\_secret, access\_token, access\_secret)

Error in setup\_twitter\_oauth(consumer\_key, consumer\_secret, access\_token, :

could not find function "setup\_twitter\_oauth"

Reference: http://geoffjentry.hexdump.org/twitteR.pdf

Trying another approach: resource: <https://cran.r-project.org/web/packages/twitteR/README.html>

twitteR

twitteR is an R package which provides access to the Twitter API. Most functionality of the API is supported, with a bias towards API calls that are more useful in data analysis as opposed to daily interaction.

Getting Started

* Please read the user vignette, which admittedly can get a bit out of date
* Create a Twitter application at http://dev.twitter.com. Make sure to give the app read, write and direct message authority.
* Take note of the following values from the Twitter app page: "API key", "API secret", "Access token", and "Access token secret".
* You can use the CRAN version (stable) via the standard install.packages("twitteR") or use the github version. To do the latter:

Steps:

Note x means executed.

Steps:

(1) install.packages(c("devtools", "rjson", "bit64", "httr"))

(2) restarted R

x library(devtools)

x install\_github("geoffjentry/twitteR")

At this point you should have twitteR installed and can proceed:

x library(twitteR)

~~setup\_twitter\_oauth("API key", "API secret")~~

used: > setup\_twitter\_oauth(consumer\_key, consumer\_secret, access\_token, access\_secret)

Result: [1] "Using direct authentication"

The API key and API secret are from the Twitter app page above. This will lead you through httr's OAuth authentication process. I recommend you look at the man page for Token in httr for an explanation of how it handles caching.

*You should be ready to go!*

*If you have any questions or issues, check out the mailing list*

Resource: <http://geoffjentry.hexdump.org/twitteR.pdf> reference to user vignette

Result: > setup\_twitter\_oauth(consumer\_key, consumer\_secret, access\_token, access\_secret)

[1] "Using direct authentication" #successful installation.

**Objective: lesson using R tool to improve redeployability. Resource: rstudio/renv**

renv: Project environments for R. <https://github.com/rstudio/renv>

**Objective: refine advanced search in Twitter to 7days of data**

Search for “(#ClimateStrike) (to:sent OR to:in OR to:reply OR to:to OR to:GretaThunberg) lang:en until:2019-09-24 since:2019-09-17”

Result: Url: <https://twitter.com/search?q=(%23ClimateStrike)%20(to%3Asent%20OR%20to%3Ain%20OR%20to%3Areply%20OR%20to%3Ato%20OR%20to%3AGretaThunberg)%20lang%3Aen%20until%3A2019-09-24%20since%3A2019-09-17&src=typed_query&f=live>

**Objective: capture through advanced search timeperiod from last global ClimateStrike from 3 days prior to 3 days after event.**

Steps: Date of strike, 20 September 2019. Start search on 17 September to 24 September. Locate event in timeperiod. Capturing a baseline.

Results: Advanced search saved: #ClimateStrike, replies to @GretaThunberg, timestamp, tweet text. from 17 Sept to 24 Sept 2019.

Search for “(#ClimateStrike) (to:sent OR to:in OR to:reply OR to:to OR to:GretaThunberg) lang:en until:2019-09-24 since:2019-09-17”

**Objective: Lesson using Web Scrapers, with Chrome Extension.**

Viewing steps on video: <https://youtu.be/AmMpJmYiAkQ> Extract Historical Tweet Data from Twitter using Google Chrome

Note: Install: scrapehero/twitter.json Json file: resource: <https://gist.github.com/scrapehero/d0305d8d15b0e447dcefdf548a9846e9>

Result: successful download of Json and upload to Chrome extension following steps below.

Advanced search: Search for “(#ClimateStrike) (to:sent OR to:in OR to:reply OR to:to OR to:GretaThunberg) lang:en until:2019-09-24 since:2019-09-17”

URL: <https://twitter.com/search?q=(%23ClimateStrike)%20(to%3Asent%20OR%20to%3Ain%20OR%20to%3Areply%20OR%20to%3Ato%20OR%20to%3AGretaThunberg)%20lang%3Aen%20until%3A2019-09-24%20since%3A2019-09-17&src=saved_search_click&f=live>

Steps:

Repeated steps from video.

1. open google

2. right click on page, go to ‘inspect’ and developer tools console will pop up

3. click on the tab ‘web scraper’ and go on to the ‘create new sitemap’ button and click on the ‘import sitemap’

4. now paste the JSON into the Sitemap JSON box and rename the sitemap ‘twitter-search’

5. to get the historical tweet data, use the advanced search in twitter by going to this url: <https://twitter.com/search-advanced?lang=en> and filter data to your needs.

6. copy the search result url

7. In the web scraper toolbar, click on the sitemap button and select the ‘edit metadata option. Past the url of the twitter advanced search page.

8. to start scraping, go to the sitemap and click ‘scrape’ from the drop-down menu

9. A new instance of Chrome will launch, enabling the extension to scroll and grab the data

10. You will receive a notification after the scarp is complete

11. you can click on the button ‘refresh’ to check the data: ‘error’ no data

12. click on the sitemap drop down and select ‘export data as csv’

13. Click on ‘download now and you will receive a csv file that you can open in excel: error exported sheet okay, but on headings in data sheet, no data from tweets

Error note (1) on first attempt: Error: /home:1 Refused to load the image 'https://stats.g.doubleclick.net/r/collect?v=1&aip=1&t=dc&\_r=3&tid=UA-30775-67&cid=1181354530.1570345614&jid=718151901&\_gid=1918001095.1570753196&gjid=1364598089&\_v=j79&z=982910046' because it violates the following Content Security Policy directive: "img-src 'self' blob: data: https://\*.cdn.twitter.com https://ton.twitter.com https://\*.twimg.com https://www.google-analytics.com https://www.periscope.tv https://www.pscp.tv https://media.riffsy.com https://\*.giphy.com https://\*.pscp.tv".

Error note (2) on second attempt: A cookie associated with a cross-site resource at http://google.com/ was set without the `SameSite` attribute. A future release of Chrome will only deliver cookies with cross-site requests if they are set with `SameSite=None` and `Secure`. You can review cookies in developer tools under Application>Storage>Cookies and see more details at https://www.chromestatus.com/feature/5088147346030592 and https://www.chromestatus.com/feature/5633521622188032.

local-ntp.html:1 A cookie associated with a cross-site resource at https://google.com/ was set without the `SameSite` attribute. A future release of Chrome will only deliver cookies with cross-site requests if they are set with `SameSite=None` and `Secure`. You can review cookies in developer tools under Application>Storage>Cookies and see more details at https://www.chromestatus.com/feature/5088147346030592 and <https://www.chromestatus.com/feature/5633521622188032>.

Next steps: Learn ‘R’ Project environments for R. <https://github.com/rstudio/renv>

**Objective: Lesson using OutWithub.**

Link from Twitter search, paste into OutWithub

<https://twitter.com/search?q=(%23contains%20OR%20%23ClimateStrike)%20(to%3Asent%20OR%20to%3Ain%20OR%20to%3Areply%20OR%20to%3Ato%20OR%20to%3AGretaThunberg)%20lang%3Aen%20until%3A2019-10-10%20since%3A2019-10-01&src=typed_query&f=live>

(#contains OR #ClimateStrike) (to:sent OR to:in OR to:reply OR to:to OR to:GretaThunberg) lang:en until:2019-10-10 since:2019-10-01

<https://twitter.com/search?f=live&q=(%23contains%20OR%20%23the%20OR%20%23hashtag%20OR%20%23ClimateStrike)%20(to%3Asent%20OR%20to%3Ain%20OR%20to%3Areply%20OR%20to%3Ato%20OR%20to%3AGretaThunberg)%20lang%3Aen%20until%3A2019-10-10%20since%3A2018-10-10&src=typed_query>

Error: non data scraped.

**Objective: setup web scrapper account on chrome, on trial ends in four days: 18 October 2019.**

Resource: Account setup in Webscraper: <https://cloud.webscraper.io/api>

Account established under Rosie Re

**Objective: Lesson on understanding application of twitter analysis of data**

Resource: http://mappingonlinepublics.net/2011/06/29/twitter-research-methods/  
<https://www.slideshare.net/Snurb> links to Axel Bruns slideshare webinars on media dynamics

<https://www.slideshare.net/Snurb>

Notes: Mapping Online Publics resource: <https://www.slideshare.net/Snurb/mapping-online-publics-part-1>

* sources
* tools (Gephi for Mac OSX) <https://gephi.org/users/download/> -- no longer useable based on java, not supported in Chrome
* baseline
* patterns events

Identification: hashtag

\*\*Capture: Texts and @replies: user name, user information, timestamp, @replies, RT @user, tweet text, embedded links

Analysis: patterns over time

* networks
* keyword phrase, mapping

Notes: This exercise provides analysis of tweet content but aggregates information is key phrase or keyword mapping, where the keyword analysis over time, and keyword co-occurrence maps.

Tools: data capture (export as csv), data processing, Excel (graphing) data visualisation Gephi (static network visualisation).

Hashtag community #ClimateStrike overall activity in time period. Mentions of the leaders. @gretathunberg, keyword co-occurrence, key themes, dynamic show @replies, #spill discussion

Objective: Examples of output using R for Twitter scraping. Sourced from resource <http://geoffjentry.hexdump.org/twitteR.pdf>. Note examples of things that can be done. could do.

* Je↵rey Stanton’s free book on data science discusses twitteR: http://ischool.syr.edu/media/documents/2012/3/DataScienceBook1\_1.pdf
* Je↵rey Breen’s sentiment analysis example: http://www.inside-r.org/howto/mining-twitter-airline-consumer-sentiment
* Mapping your followers: http://simplystatistics.org/2011/12/21/an-r-function-to-map-your-twitter-followers/
* Yangchao Zhao’s book on data mining w/ R http://www.amazon.com/Data-Mining-Examples-Case-Studies/dp/0123969638
* Gary Miner et al’s book on data mining http://www.amazon.com/Practical-Statistical-Analysis-Non-structured-Applications/dp/012386979X
* Mining Twitter with R https://sites.google.com/site/miningtwitter/home
* Organization or conversation in Twitter: A case study of chatterboxing https://www.asis.org/asist2012/proceedings/Submissions/185.pdf

Next steps: note adaptation of twitter source information scrapped through hashtags.

**Objective: Lesson on the application of R Programming Twitter Scraper (5 mins),** [**Michael Galarnyk**](https://towardsdatascience.com/@GalarnykMichael?source=post_page-----b8ac342d3efe----------------------)

* <https://www.youtube.com/watch?v=1_K01qD4Exw>
* https://www.youtube.com/watch?v=1\_K01qD4Exw
* Lesson: Access Data from Twitter API using R and Python, [Michael Galarnyk](https://towardsdatascience.com/@GalarnykMichael?source=post_page-----b8ac342d3efe----------------------)
* Resource: <https://towardsdatascience.com/access-data-from-twitter-api-using-r-and-or-python-b8ac342d3efe>
* <https://www.youtube.com/watch?v=1_K01qD4Exw>

Objective: Scrape Twitter using R and RStudio – R programming for Beginners Tutorial. Learn how to scrape Twitter with R and R Studio in just a few short steps. Then use frequency counts to build a word cloud. R programming for beginners’ tutorial.

Resource: <https://www.youtube.com/watch?v=LxOI_KQT1RU>

Steps:

<https://www.youtube.com/watch?v=WZ0iTSxz3I4>

<https://www.youtube.com/watch?v=WZ0iTSxz3I4&t=17s> -- Data Manipulation Twitter Scraper using R

Create New Twitter App: [https://apps.twitter.com/app/new](https://www.youtube.com/redirect?q=https%3A%2F%2Fapps.twitter.com%2Fapp%2Fnew&v=LxOI_KQT1RU&redir_token=zlgc89jrHKqrJLwJzZ68VNlyHJV8MTU3MTAyNjA3NUAxNTcwOTM5Njc1&event=video_description) Other videos you might find helpful which is very similar to the SkillShare Ones: [https://www.youtube.com/watch?v=1\_K01...](https://www.youtube.com/watch?v=1_K01qD4Exw) [https://www.youtube.com/watch?v=WZ0iT...](https://www.youtube.com/watch?v=WZ0iTSxz3I4)

Objective: Scrapping Twitter Data with Tweepy

Resource: <https://www.youtube.com/watch?v=Ou_floKQqd8> at 3.48 the code, source:

[https://github.com/ritvikmath/Scrapin...](https://www.youtube.com/redirect?q=https%3A%2F%2Fgithub.com%2Fritvikmath%2FScrapingData%2Fblob%2Fmaster%2FScraping%2520Twitter%2520Data.ipynb&v=Ou_floKQqd8&redir_token=MPwBItJTICcqRZFCFD1zLBh2gC58MTU3MTAyMTc4MUAxNTcwOTM1Mzgx&event=video_description)

Inputs:

INPUTS:

consumer\_key, consumer\_secret, access\_token, access\_token\_secret: codes

telling twitter that we are authorized to access this data

hashtag\_phrase: the combination of hashtags to search for

OUTPUTS:

none, simply save the tweet info to a spreadsheet

Notes: One function: search\_for\_hashtags(consumer\_key, consumer\_secret, access\_token, access\_token\_secret, hashtag\_phrase)

Contains 5 arguments—these are codes telling Twitter that we are authorized to access this data (hashtag\_phrase: the combination of hashtags to search for)

Notes: search\_for\_hashtags searches for the 5 arguments above. at 3.48

#create authentication for accessing Twitter

auth = tweepy.OAuthHandler(consumer\_key, consumer\_secret)

auth.set\_access\_token(access\_token, access\_token\_secret)

#initialize Tweepy API

api = tweepy.API(auth)

#get the name of the spreadsheet we will write to

fname = '\_'.join(re.findall(r"#(\w+)", hashtag\_phrase))

#open the spreadsheet we will write to

with open('%s.csv' % (fname), 'wb') as file:

w = csv.writer(file)

#write header row to spreadsheet

w.writerow(['timestamp', 'tweet\_text', 'username', 'all\_hashtags', 'followers\_count'])

#for each tweet matching our hashtags, write relevant info to the spreadsheet

for tweet in tweepy.Cursor(api.search, q=hashtag\_phrase+' -filter:retweets', \

lang="en", tweet\_mode='extended').items(100):

w.writerow([tweet.created\_at, tweet.full\_text.replace('\n',' ').encode('utf-8'), tweet.user.screen\_name.encode('utf-8'), [e['text'] for e in tweet.\_json['entities']['hashtags']], tweet.user.followers\_count])

consumer\_key = raw\_input(‘ConsumerKey ’)

consumer\_secret = Raw\_input(‘Consumer Secret’)

access\_token = raw\_input (‘Access Token “)

access\_token\_secret = raw\_input(‘Access Token Secret ‘)

Error: -bash: syntax error near unexpected token `('

Next steps: review implementation of “R”.

Objective: Complete Lesson. <https://www.youtube.com/watch?v=Ou_floKQqd8> HowToDataScience: Scraping Twitter Data

Resources Installed: Python and Anaconda and Twitter Access keys noted in Notepad file:

Steps:

* download a Python library
* go to terminal
* **pip install tweepy**

Result: DEPRECATION: Python 2.7 will reach the end of its life on January 1st, 2020. Please upgrade your Python as Python 2.7 won't be maintained after that date. A future version of pip will drop support for Python 2.7.

Collecting tweepy

Downloading https://files.pythonhosted.org/packages/36/1b/2bd38043d22ade352fc3d3902cf30ce0e2f4bf285be3b304a2782a767aec/tweepy-3.8.0-py2.py3-none-any.whl

Requirement already satisfied: six>=1.10.0 in ./anaconda2/lib/python2.7/site-packages (from tweepy) (1.12.0)

Collecting requests-oauthlib>=0.7.0 (from tweepy)

Downloading https://files.pythonhosted.org/packages/c2/e2/9fd03d55ffb70fe51f587f20bcf407a6927eb121de86928b34d162f0b1ac/requests\_oauthlib-1.2.0-py2.py3-none-any.whl

Requirement already satisfied: PySocks>=1.5.7 in ./anaconda2/lib/python2.7/site-packages (from tweepy) (1.7.0)

Requirement already satisfied: requests>=2.11.1 in ./anaconda2/lib/python2.7/site-packages (from tweepy) (2.22.0)

Collecting oauthlib>=3.0.0 (from requests-oauthlib>=0.7.0->tweepy)

Downloading https://files.pythonhosted.org/packages/05/57/ce2e7a8fa7c0afb54a0581b14a65b56e62b5759dbc98e80627142b8a3704/oauthlib-3.1.0-py2.py3-none-any.whl (147kB)

|████████████████████████████████| 153kB 942kB/s

Requirement already satisfied: certifi>=2017.4.17 in ./anaconda2/lib/python2.7/site-packages (from requests>=2.11.1->tweepy) (2019.6.16)

Requirement already satisfied: urllib3!=1.25.0,!=1.25.1,<1.26,>=1.21.1 in ./anaconda2/lib/python2.7/site-packages (from requests>=2.11.1->tweepy) (1.24.2)

Requirement already satisfied: idna<2.9,>=2.5 in ./anaconda2/lib/python2.7/site-packages (from requests>=2.11.1->tweepy) (2.8)

Requirement already satisfied: chardet<3.1.0,>=3.0.2 in ./anaconda2/lib/python2.7/site-packages (from requests>=2.11.1->tweepy) (3.0.4)

Installing collected packages: oauthlib, requests-oauthlib, tweepy

Successfully installed oauthlib-3.1.0 requests-oauthlib-1.2.0 tweepy-3.8.0

Resource: Tweepy Documentation: [http://docs.tweepy.org/en/v3.5.0/](https://www.youtube.com/redirect?q=http%3A%2F%2Fdocs.tweepy.org%2Fen%2Fv3.5.0%2F&v=Ou_floKQqd8&redir_token=K8DtuDrCNE9R9WuX7kaC0imSstV8MTU3MDkzODU0OEAxNTcwODUyMTQ4&event=video_description)

Objective: Lesson how to data science. Information on tweets, time stamp, person who tweeted, username, follower numbers, text of the tweet

Resources: Need: Python version 2.7 and Anaconda

Steps:

* Download: <https://www.anaconda.com/distribution/> for macOS installer
* Download: <https://datacarpentry.org/python-ecology-lesson/setup.html>
* Python version 3.x. [Python 3.7.4 - July 8, 2019](https://www.python.org/downloads/release/python-374/)

Result: applications downloaded

Objective: Resolve error. The automator twitter.com Scraper is disabled.

Resource: <https://seotoolsforexcel.com/twitter-scraper/>

Points to Github: <https://github.com/nielsbosma/SeoTools-for-Excel-Connectors/blob/master/TwitterScraper.xml>

Next steps: Try application of twitter scrapper from Niels Bosma in Github resource.

**Objective: Utilise web scrapper to access #ClimateStrike search.**

Steps: Open OutWit hub.

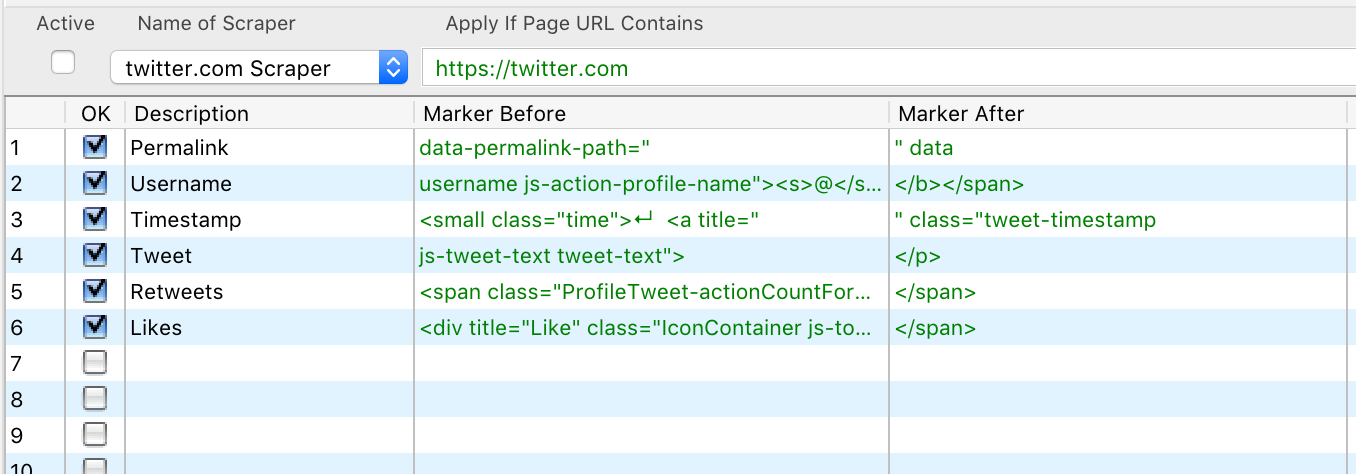
* Add url. Result from Advanced Search. (#contains OR #ClimateStrike) (to:sent OR to:in OR to:reply OR to:to OR to:GretaThunberg) lang:en until:2019-10-10 since:2019-10-01
* Scroll to bottom of page.

Use serial key ref email from 2016.

URL: <https://twitter.com/search?q=(%23contains%20OR%20%23ClimateStrike)%20(to%3Asent%20OR%20to%3Ain%20OR%20to%3Areply%20OR%20to%3Ato%20OR%20to%3AGretaThunberg)%20lang%3Aen%20until%3A2019-10-10%20since%3A2019-10-01&src=typeahead_click>

Set to dynamic.

Apply Twitter Scraper.



Result. Error. The automator twitter.com Scraper is disabled. Use it anyway? [ok]

Error. The scraper twitter.com Scraper is not destined to the current URL.

Expected URL: https://twitter.com

**Objective: Explore methodology of accessing and AI using Twitter data, in and out with csv.**

Resource: <https://github.com/ritvikmath/ScrapingData/blob/master/Scraping%20Twitter%20Data.ipynb>

<https://www.youtube.com/watch?v=Ou_floKQqd8>

Objective: Request API from Twitter completing steps outlined in Learning Journal week 6 on API for Twitter.

Steps: Developer account application completed and submitted.

Complete steps for APP to utlise API

Result: successfully created a new Twitter developer account. With this account, you now have access to new APIs, app management, and tools to facilitate and support development.

Result: Developer account operational under CarrierPigeon on Twitter. App name: Spill Network.

**Objective: Access dataset of 2016 election on Twitter: Use own twitter files as source files, then apply learning to data set at:** [**https://about.twitter.com/en\_us/values/elections-integrity.html#data**](https://about.twitter.com/en_us/values/elections-integrity.html#data)

Steps: email sent to Twitter to receive access to the dataset. Sending to [Roslyn.Walker@students.mq.edu.au](mailto:Roslyn.Walker@students.mq.edu.au)

Result: Note from Twitter:

* These datasets are of a size that a degree of capability for large dataset analysis is required, we hope to support broad analysis by making a public version of these datasets (with some account-specific information hashed) available. You can download the datasets below. No content has been redacted. Specialist researchers can request access to an unhashed version of these datasets, which will be governed by a data use agreement that will include provisions to ensure the data is used within appropriate legal and ethical parameters.

Objective: Follow Methodology: Learn applications approaches summaries in: A. Bruns and J. Burgess, 2011a. “Tools”, *Mapping Online Publics*, at http://mappingonlinepublics.net/resources/, accessed 12 October 2019.

* Results: These applications: are no longer on the web. Data gathering Twapperkeeper no longer online. Gawk available. Textual analysis Leximancer is commercial application for large text. WordStat for PC only. Visulatisation Wordle, notes below.

Result: Installed Wordle.net for Mac OS X Installer <http://www.wordle.net/> or web version: <http://www.wordle.net/create>

Notes on install: Wordle is a toy for generating “word clouds” from text that you provide. The clouds give greater prominence to words that appear more frequently in the source text. You can tweak your clouds with different fonts, layouts, and color schemes. The images you create with Wordle are yours to use however you like. You can print them out, or save them to your own desktop to use as you wish. Because the Wordle web toy no longer works for most people, you might want to try installing a desktop version of it on your Mac or Windows computer. The desktop version is exactly the same as the old web version. You'll have to work around various scary security warnings, because the app installers aren't signed.

Error: Installation “Wordle Installer” can’t be opened because it is from an unidentified developer. Your security preferences allow installation of only apps from the App Store and identified developers. Fix: open setting, security and open.

Error: You need at least a 1.8 Java Vitural Macine to install Wordle Installer. Plase download it from <http://java.com>

Steps: downloading: java from <https://www.java.com/en/download/mac_download.jsp>

**Objective: Setup Twitter Developer Account and App.**

Steps following Resource: <https://towardsdatascience.com/access-data-from-twitter-api-using-r-and-or-python-b8ac342d3efe>

Result: Twitter Developer Account and App setup.

**Objective: Advanced search Twitter for data set.**

(#contains OR #the OR #hashtag OR #ClimateStrike) (to:sent OR to:in OR to:reply OR to:to OR to:GretaThunberg) lang:en until:2019-10-10 since:2018-10-10

* The purpose of this study is to explore how the #ClimateStrike public engaged with social media to advocate for urgent action on climate.
* Generate six weeks of social media posting starting with 1 September 2019 to 10 October 2019

Steps:

(1) a simple digital search.

(2) using the advanced search function in Twitter containing the hashtag #ClimateStrike and tweeted between 1 September to 10 October 2019, in English.

(3) load in to OutWit Hub and scraped to identify:

* the user
* the date and time of the tweet
* the tweet content
* whether the message was retweeted
* commented on or liked by other users

(4) export to an Excel format

(5) clean data cleaning with manual screening in the Excel format to eliminate tweets that did not relate to the topic.

Error: Scraped content but result not useable fields need refining to sort data result. Scraper needs refinement to fields.

Error: Scraper updated. New scraper written title Scraper for Twitter Climate.

Error: Outwit hub unresponsive. Not returning results.

Alternative path. Try another application. Trying Chrome Extension.

**Notes on result from search “twitter” on slack FOAR705**

* https://www.scrapehero.com/how-to-scrape-historical-search-data-from-twitter/
* update: https://chrome.google.com/webstore/detail/web-scraper/jnhgnonknehpejjnehehllkliplmbmhn/related

**Objective:**

* <https://towardsdatascience.com/access-data-from-twitter-api-using-r-and-or-python-b8ac342d3efe>
* Reference on Access Data from Twitter API using R and Python

**Objective: On triple builder:** <https://twitter.com/semlabteam/status/1024670710621528066>

**Reference:** <https://medium.com/@thisismattmiller/triple-builder-subsets-28152364b587>

The Tool: <https://tb.semlab.io/>

The fundamentals of [Linked Data](https://en.wikipedia.org/wiki/Linked_data) are pretty straightforward at a conceptual level. The difficultly comes in like most things with the details. What is the URI of this predicate? Can I say this literal is in another language? How does this triple look serialized? This is where the learning curve usually is, how do I actually make a [RDF statement](https://en.wikipedia.org/wiki/Resource_Description_Framework) and what does it look like?

When I started teaching metadata and linked data I made little tools for my students to use that reduced the complexity of building RDF. The first one I made I simply called Triple Builder which allowed creating collaborative RDF statements.

**Objective: Test collecting tweets methodology.**

**Using methodology from Bay. M (2018)**

Objective: Explore research methods web scrapping, collecting tweets. Tweets directed at @GretaThunberg date range 1 August 2019 to 1 October 2019

Research on Methodology: Article on source data test: Bay, M. (2018). Weaponizing the haters: The Last Jedi and the strategic politicization of pop culture through social media manipulation. First Monday, 23(11)

Steps:

1. Use Twitter’s Advanced Search Functionality

2. How to use (source: <https://help.twitter.com/en/search?q=advanced+search>)

1. Enter your search into the search bar on twitter.com.
2. Click Advanced search, located underneath Search filters on the upper right of your results page, or click More options and then click Advanced search.
3. Fill in the appropriate fields to refine your search results (see below for some helpful tips).
4. Click Search to see your results.

Steps to test advanced search in Twitter for a dataset:

1. use term: (in reply to “@GretaThunberg”)

2. use the calendar dropdown to select a “from” date, “to” date or both ((@sent OR @in OR @reply OR @to OR @“GretaThunberg”) lang:en until:2019-10-01 since:2019-08-01)

3. Scrape data using Data Miner software

4. clean data for errors, duplicates and tweets from GretaThunberg herself. Gifs and meme images unaccompanied by text removed from the dataset to retain text searchability.

Result. Successful file download saved to github filename: [Advanced search @GretaThunberg](https://github.com/Roslyn007/Proof-of-Concept/blob/master/Advanced%20search%20%40GretaThunberg)

Next steps: try data miner software on data file.

**Objective: Learn how to refine advanced search on Twitter**

(source: <https://help.twitter.com/en/using-twitter/twitter-advanced-search>) Using advanced search, you can refine your search results by using any combination of the fields below:

For words

* Tweets containing all words in any position (“Twitter” and “search”)
* Tweets containing exact phrases (“Twitter search”)
* Tweets containing any of the words (“Twitter” or “search”)
* Tweets excluding specific words (“Twitter” but not “search”)
* Tweets with a specific hashtag (#twitter)
* Tweets in a specific language (written in English)

For people

* Tweets from a specific account (Tweeted by “@TwitterComms”)
* Tweets sent as replies to a specific account (in reply to “@TwitterComms”)
* Tweets that mention a specific account (Tweet includes “@TwitterComms”)

For places

* Tweets sent from a [geographic location](https://help.twitter.com/safety-and-security/tweet-location-settings), e.g. a specific city, state, country
* Use the place dropdown to select the geographic location

For dates

* Tweets sent before a specific date, after a specific date or within a date range
* Use the calendar dropdown to select a “from” date, “to” date or both
* Search for Tweets from any date since the [first public Tweet](https://twitter.com/jack/status/20)

By combining fields in the advanced search, search results are combined in a powerful way. For example, [search for Tweets](https://twitter.com/search?q=%23WorldCup%20lang%3Aen%20near%3A%22Brazil%22%20within%3A15mi%20until%3A2014-07-31&src=typd) in English with the hashtag “#WorldCup” sent from Brazil in July 2014.

Action: Test Advanced Search with example on using reply to @GretaThunberg.

Result: Combine search successful, results used above to create a data set of Tweets.

**Objective: Create a template for documenting running the acceptance tests.**

Steps:

1. Create a template for documenting running the acceptance tests you’ve identified.

2. Make sure your project management system can track errors that you find and need to fix (or document) with these tests.

3. Make sure you have a place where it is easy to document your specific acceptance tests, both good and bad

Resources: https://multisearch.mq.edu.au/permalink/f/1lmkbbh/TN\_safari\_s9781118133156

* Example: We have used a confluence wiki for this purpose: <https://faimsproject.atlassian.net/wiki/spaces/FAIMS/pages/941293583/Copy+of+Harvard+Excavation+2017+Regression+Test+-+AW>
* Example: In a github issue, <https://guides.github.com/features/issues/>, reply to the issue with a comment which contains your acceptance test for that issue.

Alternative path: Use your learning journal in the already normal format, where expectation/intention is the task you are performing as your acceptance test.

Result: using learning journal in the interim.

Next steps follow example from github.

**Objective: Export data report from Trello**

Steps:

* Head to the Board Menu to the right of your screen in Trello and click on More.
* Click on Print and Export.
* click Print to print it, or click Export JSON to export the board data in a JSON format.

Result: PDF file of job lists filed to github and cloudstor.

Next steps: trial text export in tabular form.

**Objective: Setup example for testing proof of concept**

Steps:

* Establish test summary to test user stories
* Preconditions
* Datasets required to test

Example below

|  |  |  |  |
| --- | --- | --- | --- |
| Test steps: | Result | Expected results | Comment |
| Login |  | login successful |  |
| Click on xxx |  | go to page |  |
| Enter module name, sampling loads |  | module created |  |

Objective: Explore research methods web scrapping

Research on Methodology: Article on source data test: Bay, M. (2018). Weaponizing the haters: The Last Jedi and the strategic politicization of pop culture through social media manipulation. First Monday, 23(11)

Notes from Consultation with Brian 13 September 2019.

**Objective:** Develop an action plan for next steps to Minimum Viable Product Development

**Steps:**

(1) Action data carpentry lessons all way to the end.

(2) Get to R.

(3) Learn how to use R with js files.

(4) Use on own source files.

(5) Expand for use on twitter dataset for the 2016 elections.

By end of break use twitter dataset in R (search google to see if others have accomplished this).

**Resources:**

* Explore application and links from Brian to be helpful to project development and testing.
* <https://datacarpentry.org/r-socialsci/02-starting-with-data/index.html>
* Lessons in data carpentry all to the end <https://cran.r-project.org/web/packages/jsonlite/vignettes/json-aaquickstart.html>
* <https://datacarpentry.org/lessons/#social-science-curriculum> See especially, Data Analysis and Visualization with R for Social Scientists link on the above page.
* <https://stackoverflow.com/questions/16947643/getting-imported-json-data-into-a-data-frame/37739735>
* Use own twitter files as source files, then apply learning to data set at: <https://about.twitter.com/en_us/values/elections-integrity.html#data>
* Investigate functions and operation of the Twitter power track - search Power Track on Twitter.

Results: requires further research following resources.

Next steps: return to steps under this objective.

**30 September 2019**

**Objective: Understand how a project brief is formed. Acquire core technological skills needed to produce transparent and reproducible research. Design and develop a digital tool to support your masters research.**

Steps:

(1) Proof-of-concept technology deployment

* Design: Scoping Exercise (wk 2, 3)
* Design: Elaboration Plan (wk 4)
* Implementation: Elaboration Results (wk 6)
* Design: Proof of concept design (wk 7)
* Implementation: Proof of concept due (wk 7)

Result: Set of documentation tracking the use of core technological skills to produce transparent and reproducible research. Using new skills to design and develop a digital tool to support masters research.

## (2) Data carpentry exercise notes

Week 8: Start working through the first episode (<https://datacarpentry.org/r-socialsci/01-intro-to-r/index.html>) in Data Carpentry, R. Learning Journal for this lesson (all episodes up to Introducing dplyr and tidyr) due before class Week 13.

(1) <https://datacarpentry.org/r-socialsci/01-intro-to-r/index.html> (50mins)

(2) <https://datacarpentry.org/r-socialsci/02-starting-with-data/index.html> (50mins)

(3) <https://datacarpentry.org/r-socialsci/03-dplyr-tidyr/index.html> (50mins)

Objective overview lesson for Introduction to R.

(1) Questions

* What data types are available in R?
* What is an object?
* How can values be initially assigned to variables of different data types?
* What arithmetic and logical operators can be used?
* How can subsets be extracted from vectors and data frames?
* How does R treat missing values?
* How can we deal with missing values in R?

Objectives

* Define the following terms as they relate to R: object, assign, call, function, arguments, options.
* Assign values to objects in R.
* Learn how to name objects.
* Use comments to inform script.
* Solve simple arithmetic operations in R.
* Call functions and use arguments to change their default options.
* Inspect the content of vectors and manipulate their content.
* Subset and extract values from vectors.
* Analyze vectors with missing data.

Objective Setup. R Studio: <https://datacarpentry.org/r-socialsci/setup.html>

Steps: download RStudio: 1.2.5001 macOS 10.12 on desktop computer. Added to applications.

install.packages(c("tidyverse"))

After installing R and RStudio, you need to install the tidyverse package. After starting RStudio, at the console type: install.packages(c("tidyverse"))

Result: Installed. no error messages.

Objective: lesson creating objects in R.

* <- is the assignment operator. It assigns values on the right to objects on the left. So, after executing x <- 3, the value of x is 3. The arrow can be read as 3 goes into x. For historical reasons, you can also use = for assignments, but not in every context. Because of the slight differences in syntax, it is good practice to always use <- for assignments.
* In RStudio, typing Alt + - (push Alt at the same time as the - key) will write <- in a single keystroke in a PC, while typing Option + - (push Option at the same time as the - key) does the same in a Mac.
* Objects can be given any name such as x, current\_temperature, or subject\_id. You want your object names to be explicit and not too long. They cannot start with a number (2x is not valid, but x2 is). R is case sensitive (e.g., age is different from Age). There are some names that cannot be used because they are the names of fundamental functions in R (e.g., if, else, for, see here for a complete list). In general, even if it’s allowed, it’s best to not use other function names (e.g., c, T, mean, data, df, weights). If in doubt, check the help to see if the name is already in use. It’s also best to avoid dots (.) within an object name as in my.dataset. There are many functions in R with dots in their names for historical reasons, but because dots have a special meaning in R (for methods) and other programming languages, it’s best to avoid them. It is also recommended to use nouns for object names, and verbs for function names. It’s important to be consistent in the styling of your code (where you put spaces, how you name objects, etc.). Using a consistent coding style makes your code clearer to read for your future self and your collaborators. In R, three popular style guides are Google’s, Jean Fan’s and the tidyverse’s. The tidyverse’s is very comprehensive and may seem overwhelming at first. You can install the lintr package to automatically check for issues in the styling of your code.

Objective: Lesson objects vs. variables

Note: What are known as objects in R are known as variables in many other programming languages. Depending on the context, object and variable can have drastically different meanings. However, in this lesson, the two words are used synonymously. For more information see: https://cran.r-project.org/doc/manuals/r-release/R-lang.html#Objects

Steps:

add area\_hectares as object with value 1.0   
area\_hectares <- 1.0

Note: #allows comments in code.

area\_hectares <- 1.0 *# land area in hectares*

area\_acres <- area\_hectares \* 2.47 *# convert to acres*

area\_acres *# print land area in acres.*

RStudio makes it easy to comment or uncomment a paragraph: after selecting the lines you want to comment, press at the same time on your keyboard Ctrl + Shift + C. If you only want to comment out one line, you can put the cursor at any location of that line (i.e. no need to select the whole line), then press Ctrl + Shift + C.

**Exercise:**

Create two variables length and width and assign them values.

R Studio might add “()” after width and if you leave the parentheses you will get unexpected results.

This is why you might see other programmers abbreviate common words.

Create a third variable area and give it a value based on the current values of length and width. Show that changing the values of either length and width does not affect the value of area.

Result:

> area\_hectares <- 1.0

> area\_hectares

[1] 1

> 2.47 \* area\_hectares

[1] 2.47

> area\_hectares <- 2.5

> 2.47 \* area\_hectares

[1] 6.175

> area\_hectares <- 2.47 \* area\_hectares

> area\_acres <- 2.47 \* area\_hectares

> area\_acres <- 2.5

> area\_acres <- 2.47

> area\_hectares <- 2.5

> area\_acres <- 50

> area\_acres <- 2.47 x area\_hectares

Error: unexpected symbol in "area\_acres <- 2.47 x"

> area\_acres <- 2.47 \* area\_hectares

> length<- 2.5

> length <- 2.5

> width <- 3.2

> area <- length \* width

> area

[1] 8

> length <- 7.0

> width <- 6.5

> area

[1] 8

>

Objective complete Lesson: Functions and their arguments.

Note: Functions are “canned scripts” that automate more complicated sets of commands including operations assignments, etc. Many functions are predefined, or can be made available by importing R packages (more on that later). A function usually gets one or more inputs called arguments. Functions often (but not always) return a value. A typical example would be the function sqrt(). The input (the argument) must be a number, and the return value (in fact, the output) is the square root of that number. Executing a function (‘running it’) is called calling the function. An example of a function call is:

Note: It’s good practice to put the non-optional arguments (like the number you’re rounding) first in your function call, and to specify the names of all optional arguments. If you don’t, someone reading your code might have to look up the definition of a function with unfamiliar arguments to understand what you’re doing.

Steps: Type in ?round at the console and then look at the output in the Help pane. What other functions exist that are similar to round? How do you use the digits parameter in the round function?

Result: Notes from Help on Round through command ?round: Result:

|  |  |
| --- | --- |
| digits | integer indicating the number of decimal places (round) or significant digits (signif) to be used. Negative values are allowed |

**Objective: Lesson vectors and data types:**

Note: A vector is the most common and basic data type in R, and is pretty much the workhorse of R. A vector is composed by a series of values, which can be either numbers or characters. We can assign a series of values to a vector using the c() function. For example we can create a vector of household members for the households we’ve interviewed and assign it to a new object hh\_members:

A vector can also contain characters. For example, we can have a vector of the building material used to construct our interview respondents’ walls (respondent\_wall\_type):

Note: The quotes around “muddaub”, etc. are essential here. Without the quotes R will assume there are objects called muddaub, burntbricks and sunbricks. As these objects don’t exist in R’s memory, there will be an error message.

Result:

[1] 8

> round(3.14159)

[1] 3

> args(round)

function (x, digits = 0)

NULL

> ?round

> digits=2

> round(3.14159,2)

[1] 3.14

> round(digits = 2, x = 3.14159)

[1] 3.14

> ?round

> ?round

> hh\_members <- c(3,7,10,6)

> hh\_members

[1] 3 7 10 6

> respondent\_wall\_type <- c("muddaub", "burntbrisk", "sunbricks")

> respondent\_wall\_type

[1] "muddaub" "burntbrisk" "sunbricks"

There are many functions that allow you to inspect the content of a vector. length() tells you how many elements are in a particular vector:

Result:

> length (hh\_members)

[1] 4

> lenght(respondent\_wall\_type)

Error in lenght(respondent\_wall\_type) : could not find function "lenght"

> length(respondent\_wall\_type)

[1] 3

The function class() indicates the class (the type of element) of an object:

Result:

> class(hh\_members)

[1] "numeric"

> class(respondent\_wall\_type)

[1] "character

The function str() provides an overview of the structure of an object and its elements. It is a useful function when working with large and complex objects:

> str(hh\_members)

num [1:4] 3 7 10 6

> str(respondent\_wall\_type)

chr [1:3] "muddaub" "burntbrisk" "sunbricks"

You can use the c() function to add other elements to your vector:

> possessions <- c(possessions, "mobile\_phone") #add to the end of the vector

> possessions <- c("car", possessions) #add to the beginning of the vector

> possessions

[1] "car" "bicycle" "radio" "television" "mobile\_phone"

>

An atomic vector is the simplest R data type and is a linear vector of a single type. Above, we saw 2 of the 6 main atomic vector types that R uses: "character" and "numeric" (or "double"). These are the basic building blocks that all R objects are built from. The other 4 atomic vector types are:

"logical" for TRUE and FALSE (the boolean data type)

"integer" for integer numbers (e.g., 2L, the L indicates to R that it’s an integer)

"complex" to represent complex numbers with real and imaginary parts (e.g., 1 + 4i) and that’s all we’re going to say about them

"raw" for bitstreams that we won’t discuss further

You can check the type of your vector using the typeof() function and inputting your vector as the argument.

Vectors are one of the many data structures that R uses. Other important ones are lists (list), matrices (matrix), data frames (data.frame), factors (factor) and arrays (array).

Objective: Lesson subsetting vectors.

to extract one or several values from a vector, we must provide one or several indices in square brackets

Objective: Lesson Conditional subsetting

Another common way of subsetting is by using a logical vector. TRUE will select the element with the same index, while FALSE will not:

You can combine multiple tests using & (both conditions are true, AND) or | (at least one of the conditions is true, OR):

Here, < stands for “less than”, > for “greater than”, >= for “greater than or equal to”, and == for “equal to”. The double equal sign == is a test for numerical equality between the left and right hand sides, and should not be confused with the single = sign, which performs variable assignment (similar to <-).

A common task is to search for certain strings in a vector. One could use the “or” operator | to test for equality to multiple values, but this can quickly become tedious. The function %in% allows you to test if any of the elements of a search vector are found:

Result:

> respondent\_wall\_type <- c

> respondent\_wall\_type <- c("muddarb", "burntbricks", "sunbricks")

> respondent\_wall\_type <- c("muddaub", "burntbricks", "sunbricks")

> respondent\_wall\_type[2]

[1] "burntbricks"

> respondent\_wall\_type[c(3, 2)]

[1] "sunbricks" "burntbricks"

> more\_respondent\_wall\_type <- respondent\_wall\_type[c(1, 2, 3, 2, 1, 3)]

> more\_respondent\_wall\_type

[1] "muddaub" "burntbricks" "sunbricks" "burntbricks" "muddaub" "sunbricks"

> hh\_members <- c(3, 7, 10, 6)

> hh\_members[c(TRUE, FALSE, TRUE, TRUE)]

[1] 3 10 6

> hh\_members > 5

[1] FALSE TRUE TRUE TRUE

> hh\_members[hh\_members > 5]

[1] 7 10 6

> hh\_members[hh\_members < 3 | hh\_members > 5]

[1] 7 10 6

> hh\_members[hh\_members >= 7 & hh\_members == 3]

numeric(0)

> possessions <- c("car", "bicycle", "radio", "television", "mobile\_phone")

> possessions[possessions == "car" | possessions == "bicycle"] # returns both car and bicycle

[1] "car" "bicycle"

> possessions %in% c("car", "bicycle", "motorcycle", "truck", "boat")

[1] TRUE TRUE FALSE FALSE FALSE

> possessions[possessions %in% c("car", "bicycle", "motorcycle", "truck", "boat")]

[1] "car" "bicycle"

Objective missing data

R was designed to analyze datasets, it includes the concept of missing data (which is uncommon in other programming languages). Missing data are represented in vectors as NA.

When doing operations on numbers, most functions will return NA if the data you are working with include missing values. This feature makes it harder to overlook the cases where you are dealing with missing data. You can add the argument na.rm=TRUE to calculate the result while ignoring the missing values.

use the typeof() function to find the type of your atomic vector.

Use the function median() to calculate the median of the rooms vector.

Use R to figure out how many households in the set use more than 2 rooms for sleeping.

Summary:

* Access individual values by location using [].
* Access arbitrary sets of data using [c(...)].
* Use logical operations and logical vectors to access subsets of data.

**(2) Objective: Lesson Starting with Data** [**https://datacarpentry.org/r-socialsci/02-starting-with-data/index.html**](https://datacarpentry.org/r-socialsci/02-starting-with-data/index.html)

Notes:

Questions

What is a data.frame?

* How can I read a complete csv file into R?
* How can I get basic summary information about my dataset?
* How can I change the way R treats strings in my dataset?
* Why would I want strings to be treated differently?
* How are dates represented in R and how can I change the format?

Objectives

* Describe what a data frame is.
* Load external data from a .csv file into a data frame.
* Summarize the contents of a data frame.
* Describe the difference between a factor and a string.
* Convert between strings and factors.
* Reorder and rename factors.
* Change how character strings are handled in a data frame.
* Examine and change date formats.

**Objective lesson of presentation of the SAFI data**

Notes: <https://datacarpentry.org/socialsci-workshop/data/> SAFI teaching database

Steps:

Load data

Error: locating directory where SAFI\_clean.csv is --- is looking for "/Users/Tquestudio/ that doesn’t exist

Fix: Moved file to download and changed path to: "/Users/Tquestudio/download" > getwd()

[1] "/Users/Tquestudio"

> setwd("/Users/Tquestudio/downloads")

> getwd()

[1] "/Users/Tquestudio/Downloads"

Steps: to load csv file to downloads

> interviews <- read\_csv("/Users/Tquestudio/downloads/SAFI\_clean.csv", na = "NULL")

> view(interviews)

> head(interviews) #lists data in file by headers

read\_csv() assumes that fields are delineated by commas, however, in several countries, the comma is used as a decimal separator and the semicolon (;) is used as a field delineator. If you want to read in this type of files in R, you can use the read\_csv2 function. It behaves exactly like read\_csv but uses different parameters for the decimal and the field separators. If you are working with another format, they can be both specified by the user. Check out the help for read\_csv() by typing ?read\_csv to learn more. There is also the read\_tsv() for tab-separated data files, and read\_delim() allows you to specify more details about the structure of your file.

Objective: Lesson on data frames and tibbles.

Notes: Data frames are the *de facto* data structure for tabular data in R, and what we use for data processing, statistics, and plotting.

A data frame is the representation of data in the format of a table where the columns are vectors that all have the same length. Because columns are vectors, each column must contain a single type of data (e.g., characters, integers, factors). For example, here is a figure depicting a data frame comprising a numeric, a character, and a logical vector.

Notes: A data frame can be created by hand, but most commonly they are generated by the functions read\_csv() or read\_table(); in other words, when importing spreadsheets from your hard drive (or the web).

A tibble is an extension of R data frames used by the tidyverse. When the data is read using read\_csv(), it is stored in an object of class tbl\_df, tbl, and data.frame. You can see the class of an object with

Notes: When calling a tbl\_df object (like interviews here), there is already a lot of information about our data frame being displayed such as the number of rows, the number of columns, the names of the columns, and as we just saw the class of data stored in each column. However, there are functions to extract this information from data frames. Here is a non-exhaustive list of some of these functions. Let’s try them out!

Objective: inspect data frames:

Size:

* dim(interviews) - returns a vector with the number of rows in the first element, and the number of columns as the second element (the dimensions of the object)
* nrow(interviews) - returns the number of rows
* ncol(interviews) - returns the number of columns

Content:

* head(interviews) - shows the first 6 rows
* tail(interviews) - shows the last 6 rows

Names:

* names(interviews) - returns the column names (synonym of colnames() for data.frame objects)

Summary:

* str(interviews) - structure of the object and information about the class, length and content of each column
* summary(interviews) - summary statistics for each column

Objective: Lesson Indexing and subsetting data frames

Steps: to extract some specific data from it, we need to specify the “coordinates”

>>>>>>>>>>>>>>>>>>>>>

Notes from in class workshop: Datacarpentry 4 October

Command: getwd()

Result: [1] "/Users/roslynwalker"

New project

New directory

~ home directory

dir.create("data\_output")

Git message

Create project structure

\* Made gitignore, data-carpentry project, script.R

\* script.R is empty

For rstudion download <https://rstudio.com/products/rstudio/download/#download>

Ls -al

Show files

Find my password

Tools.version control>commit

Push commit

Link local to git

Tools version control push Datacarpentry 4 October

Command: getwd()

Result: [1] "/Users/roslynwalker"

New project

New directory

~ home directory

dir.create("data\_output")

Git message

Create project structure

\* Made gitignore, data-carpentry project, script.R

\* script.R is empty

For rstudion download <https://rstudio.com/products/rstudio/download/#download>

Ls -al

Show files

Find my password

Tools.version control>commit

Push commit

Link local to git

Tools version control push

>>>>>>>>>>>>

Week 7 exercises: Data Carpentry [OpenRefine for Social Science Data](https://datacarpentry.org/openrefine-socialsci/), 13 September 2019

Data Cleaning with OpenRefine. Examining Numbers, Using Scripts, Exporting, Other Resources: <https://datacarpentry.org/openrefine-socialsci/>

>>>>>>>>>

Open Refine notes from class exercise: 13 September 2019

**Objective Lesson 4. Examining numbers** [**https://datacarpentry.org/openrefine-socialsci/04-numbers/index.html**](https://datacarpentry.org/openrefine-socialsci/04-numbers/index.html)

Steps: To transform cells in the years\_farm column to numbers, click the down arrow for that column, then Edit cells > Common transforms… > To number. You will notice the years\_farm values change from left-justified to right-justified, and black to green in color.

Transform three columns, no\_members, yrs\_liv, and buildings\_in\_compound, from text to numbers.

Result: only numerals (0-9) can be transformed to numbers.

**Objective Lesson 5. Using scripts** [**https://datacarpentry.org/openrefine-socialsci/05-scripts/index.html**](https://datacarpentry.org/openrefine-socialsci/05-scripts/index.html)

Steps: data-cleaning steps applied to the data

(1) In the Undo / Redo section, click Extract..., and select the steps to apply to other datasets by clicking the check boxes.

(2) Copy the code from the right hand panel and paste it into a text editor. Saves as plain text file a .txt file.

Result: All changes are tracked in OpenRefine, this information can be used for scripts for reproducing an analysis. Script saved to Github as text file.

**Objective Lesson 6. Exporting and saving data from OpenRefine**

Steps: Exporting--to share raw data and cleaning steps

(1) Click the Export button in the top right and select Export project.

A tar.gz file downloaded to default Download directory.

The tar.gz extension is a compressed file.

Result. downloaded tar.gz file, contains a collection of zip files. With a change.txt file. These change.txt files are the records of each individual transformation that you did to your data. A data.zip file. includes a file data.txt and a copy of r raw data.

Result: saved to Github as text file.

**Objective Lesson 7. Other online resources to get help with OpenRefine**

* OpenRefine web site and documentation
* [OpenRefine web site](http://openrefine.org/) <http://openrefine.org/>
* [OpenRefine Documentation for Users](https://github.com/OpenRefine/OpenRefine/wiki/Documentation-For-Users) <https://github.com/OpenRefine/OpenRefine/wiki/Documentation-For-Users>
* [OpenRefine documentation Wiki site](https://github.com/OpenRefine/OpenRefine/wiki/Documentation-For-Users) <https://github.com/OpenRefine/OpenRefine/wiki/Documentation-For-Users>
* [Using OpenRefine](http://www.worldcat.org/title/using-openrefine-the-essential-openrefine-guide-that-takes-you-from-data-analysis-and-error-fixing-to-linking-your-dataset-to-the-web/oclc/889271264) book <https://www.worldcat.org/title/using-openrefine-the-essential-openrefine-guide-that-takes-you-from-data-analysis-and-error-fixing-to-linking-your-dataset-to-the-web/oclc/889271264>

>>>>>>>>>>>>>>>

Class catch up. 1 October 2019. Learning Journal submission, 2pm. 4 October 2019

Week 6 exercises: Data Carpentry--Data Cleaning with OpenRefine - https://datacarpentry.org/openrefine-socialsci/

Lesson on Introduction, Working with Open Refine, Filtering and Sorting

**Objective: Setup for lesson on OpenRefine**

Steps: download <http://openrefine.org/> for Mac.

* Chrome browser is installed and set as default browser.
* OpenRefine runs in your default browser. It will not run correctly in Internet Explorer.
* Download software from [http://openrefine.org](http://openrefine.org/) -- OpenRefine 3.2
* Unzip the downloaded file into a directory by double-clicking it. Name that directory something like OpenRefine.
* Go to your newly created OpenRefine directory.
* Launch OpenRefine
* Drag icon into Applications folder, and Ctrl-click/Open it.
* Note: If using a different browser, or OpenRefine does not automatically open for you, point your browser at http://127.0.0.1:3333/ or http://localhost:3333 to launch the program.

Error message: “OpenRefine” can’t be opened because it is from an unidentified developer.

Solution: <https://github.com/OpenRefine/OpenRefine/wiki/Installation-Instructions>

macOS: If you get a message saying "Open Refine can't be opened because it is from an unidentified developer"

* open System Preferences and go to
* "Security and Privacy" and the
* General tab. Here you will see a message indicating that "OpenRefine was blocked from opening because it is not from an identified developer".
* Click the "Open Anyway" button to complete the OpenRefine installation.

**Objective. Complete lesson on Introduction to OpenRefine for Social Science Data. What is OpenRefine useful for?** <https://datacarpentry.org/openrefine-socialsci/01-introduction/index.html>

Teaching: 10 mins.

Noted: Background to application. An open source tool to clean up messy data and transform its’ format. OpenRefine cleans and formats data and tracks the changes made. Data workflow is used in preparing the data for analysis such as data cleaning. Data cleaning is where errors are found and corrected, and the format can be made consistent. This step forms reproducibility as the analysis. Differentiate data cleaning from data organization.

Note use and application of OpenRefine.

* actions can be reversed in OpenRefine.
* when using save it becomes a new file--OpenRefine copies the data and does not modify the original dataset.
* data cleaning steps often need repeating with multiple files. OpenRefine keeps track of all of actions taken and allows them to be applied to different datasets. Save as txt file.
* concepts such as clustering algorithms are quite complex, but OpenRefine makes it easy to introduce them and use them.

**Features of OpenRefine**

* Open source is on GitHub: <https://github.com/OpenRefine/OpenRefine>
* Community of practice ready to help.
* Works with datasets (100,000 rows) and adjusts memory allocation to accommodate larger datasets.

Objective. Complete lesson on Working with Open Refine to OpenRefine for Social Science Data. <https://datacarpentry.org/openrefine-socialsci/02-working-with-openrefine/index.html>

Teaching 15 mins: Exercises 20 mins.

* How can we bring our data into OpenRefine?
* How can we sort and summarize our data?
* How can we find and correct errors in our raw data?

Steps: Create a new OpenRefine project

1. Downloaded data obtained from interviews of farmers in two countries in eastern sub-Saharan Africa (Mozambique and Tanzania). Filename SAFI\_openrefine.csv Open file in OpenRefine.

2. Create new project.

3. Get data from. This computer.

Result: opened SAFI\_openrefine csv

**Objective: Use facets—explore data by applying multiple filters**

Steps for exercise:

Use faceting look for potential errors in data entry in the village column.

1. In the village column. Click the down arrow and choose Facet > Text facet.
2. sorting this facet by name and by count.
3. Hover the mouse over names for edit function
4. fix errors:

* Change Chirdozo to Chirodzo.
* Ruca to Ruaca.
* Correct Ruaca - Nhamuenda and Ruaca-Nhamuenda
* 49  an error fix unsure

Exercise:

1. interview\_date values = 19 unique

2. change format through edit cells> common transforms>to date

3. Face> timeline facet

Most of the data was collected in November of 2016.

**Objective using clustering to detect errors**

Steps:

* In the village Text Facet we created in the step above, click the Cluster button.
* In the resulting pop-up window, you can change the Method and the Keying Function. Try different combinations to see what different mergers of values are suggested.
* Select the key collision method and metaphone3 keying function. It should identify two clusters.
* Click the Merge? box beside each cluster, then click Merge Selected and Recluster to apply the corrections to the dataset.
* Try selecting different Methods and Keying Functions again, to see what new merges are suggested.
* You should find no more clusters are found. None of the available methods offered to cluster Ruaca-Nhamuenda with Ruaca or Chirdozo with Chirodzo. To merge these values we need to hover over them in the village text facet, select edit, and manually change the names.

Results:

* Changed Chirdozo to Chirodzo and Ruaca-Nhamuenda to Ruaca. You should now have four clusters: Chirodzo, God, Ruaca and 49.

**Objective: Lesson Transforming data.**

Steps:

Click the down arrow at the top of the items\_owned column. Choose Edit Cells >

This opens a window, type a GREL expression. GREL stands for General Refine Expression Language. First we will remove all of the left square brackets ([). In the Expression box type value.replace("[", "") and click OK. What the expression means is this: Take the value in each cell in the selected column and replace all of the “[” with “” (i.e. nothing - delete).

Click OK. See in the items\_owned column there are no left square brackets.

Repeat to remove the single quote marks ('), the right square brackets (]), and spaces from the items\_owned column.

Result

* value.replace("'", "")
* value.replace("]", "")
* value.replace(" ", "") You should now have a list of items separated by semi-colons (;).

Steps:

* Click the down arrow at the top of the items\_owned column. Choose Facet > Custom text facet...
* In the Expression box, type value.split(";").

Click OK.

Result: most commonly owned are mobile phone and radio, least commonly owned are cars and computers.

Exercise: Clean up steps and customized text faceting for the months\_lack\_food column. Which month(s) were farmers more likely to lack food?

Result: All four cleaning steps can be performed by combining .replace statements. The command is: value.replace("[", "").replace("]", "").replace(" ", "").replace("'", "") This can also be done in four separate steps if preferred. November was the most common month for respondents to lack food.