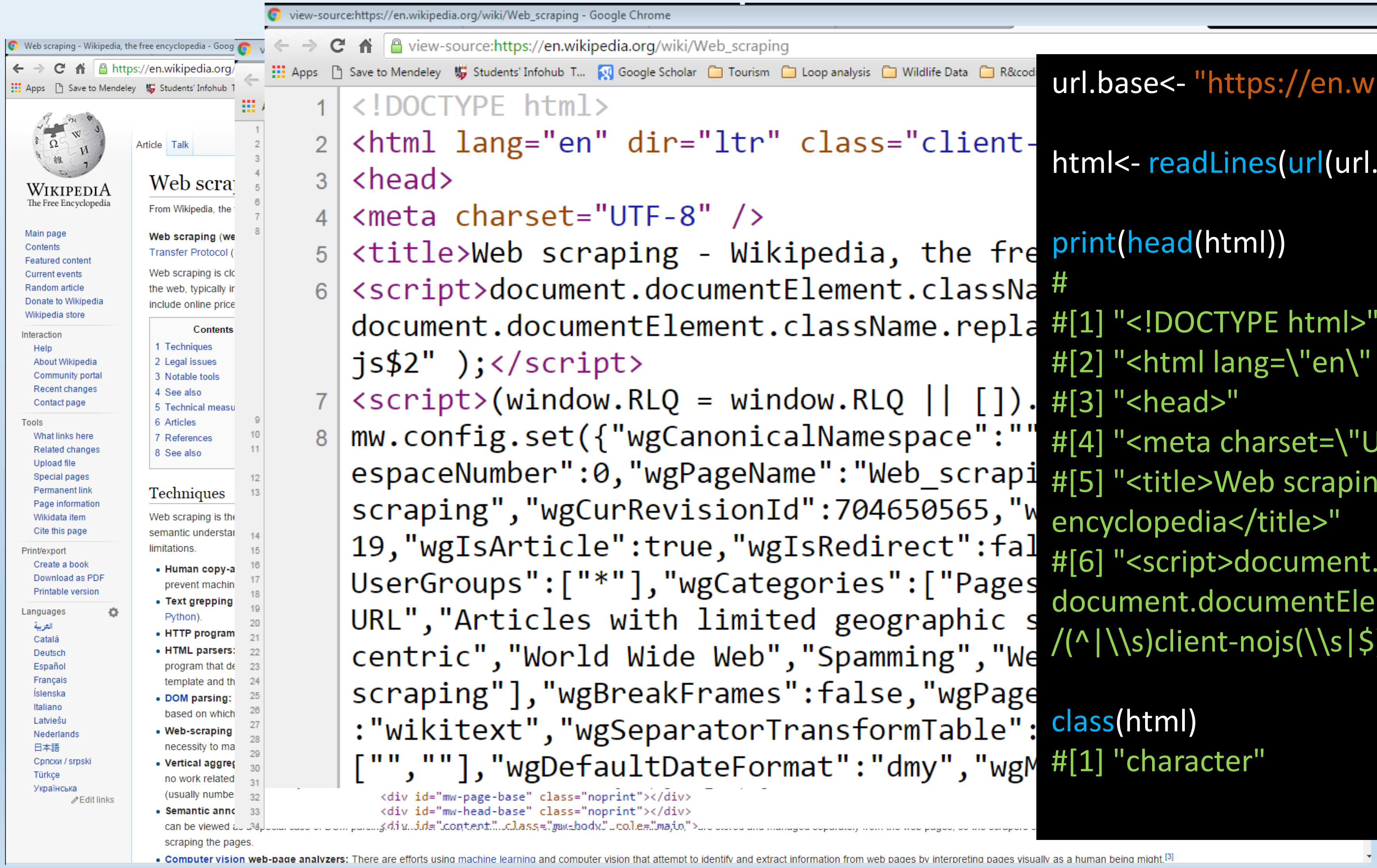
Social Media Scraping with R

What is web scraping?

From Wikipedia:

“**Web scraping** (**web harvesting** or **web data extraction**) is a computer software technique of extracting information from websites.”

How does it work? – Basic R



```
<!DOCTYPE html>
<html lang="en" dir="ltr" class="client-
<head>
<meta charset="UTF-8" />
<title>Web scraping - Wikipedia, the fre
<script>document.documentElement.classNa
document.documentElement.className.repla
js$2" );</script>
<script>(window.RLQ = window.RLQ || []).
mw.config.set({"wgCanonicalNamespace":""
espaceNumber":0,"wgPageName":"Web_scrapi
scraping","wgCurRevisionId":704650565,"w
19,"wgIsArticle":true,"wgIsRedirect":fal
UserGroups":["*"],"wgCategories":["Pages
URL","Articles with limited geographic s
centric","World Wide Web","Spamming","We
scraping"],"wgBreakFrames":false,"wgPage
:"wikitext","wgSeparatorTransformTable":
["",""],"wgDefaultDateFormat":"dmy","wgM
```

```
url.base<- "https://en.wikipedia.org/wiki/Web_scraping"

html<- readLines(url(url.base))

print(head(html))
#
#[1] "<!DOCTYPE html>"
#[2] "<html lang=\"en\" dir=\"ltr\" class=\"client-nojs\">"
#[3] "<head>"
#[4] "<meta charset=\"UTF-8\" />"
#[5] "<title>Web scraping - Wikipedia, the free
encyclopedia</title>"
#[6] "<script>document.documentElement.className =
document.documentElement.className.replace(
/(^|\\s)client-nojs(\\s|$)/, \"$1client-js$2\" );</script>"

class(html)
#[1] "character"
```

How does it work? - XML

```
library(XML)

html <- htmlTreeParse(html, useInternalNodes=TRUE)

class(html)
#[1] "HTMLInternalDocument" "HTMLInternalDocument" "XMLInternalDocument"
#[4] "XMLAbstractDocument"

html
#<!DOCTYPE html>
#<html lang="en" dir="ltr" class="client-nojs">
#<head>
#<meta charset="UTF-8">
#<title>Web scraping - Wikipedia, the free encyclopedia</title>
#<script>document.documentElement.className = document.documentElement.className.replace(
#......
```

How does it work? - XPath

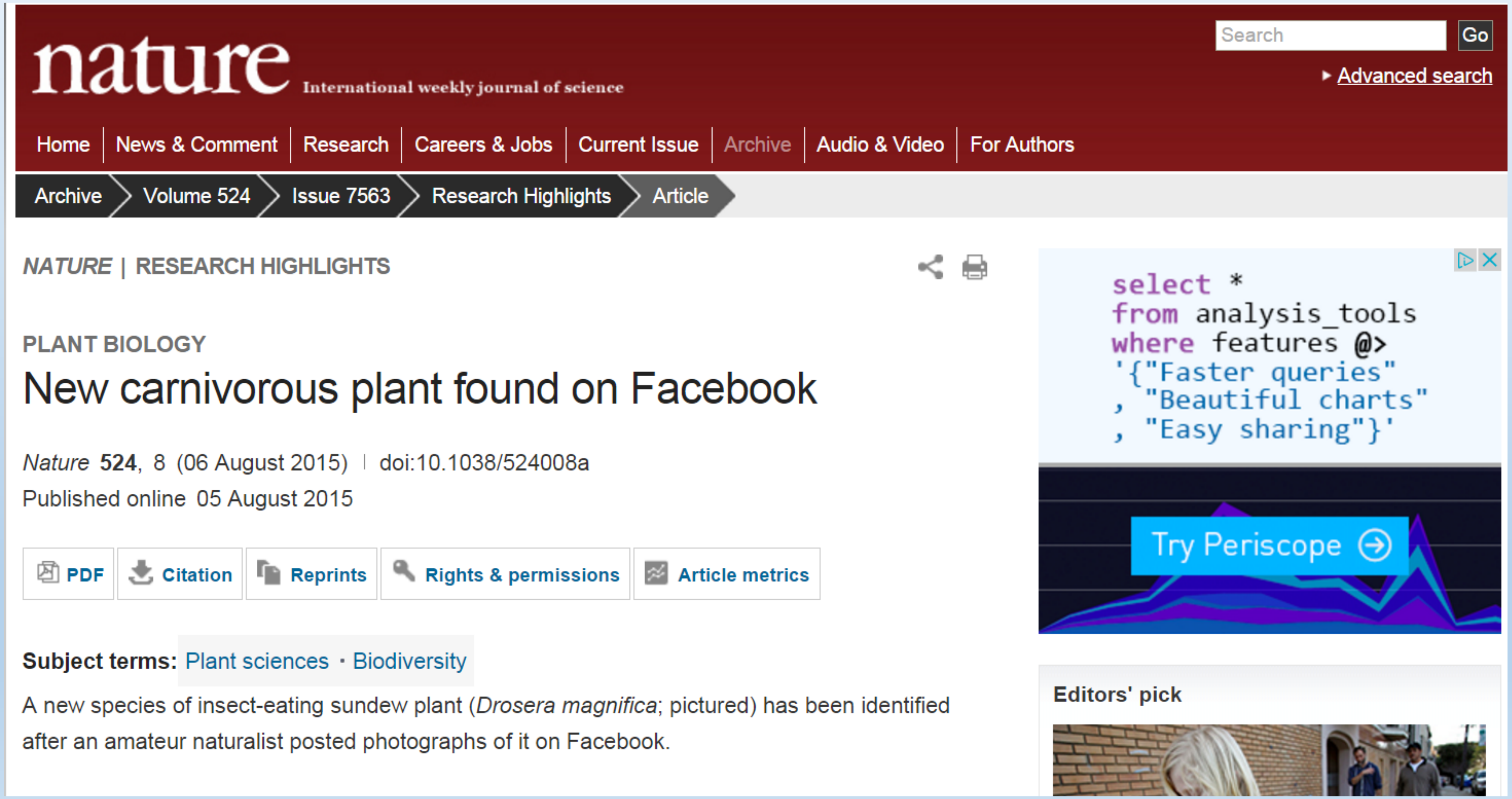
Xpath syntax*

```
headings <- xpathSApply(html, "//h2", xmlValue)

headings
#[1] "Contents"                        "Techniques[edit]"
#[3] "Legal issues[edit]"                 "Notable tools[edit]"
#...
```

* www.w3schools.com

Social Media Data: Why?



NATURE | RESEARCH HIGHLIGHTS

PLANT BIOLOGY

New carnivorous plant found on Facebook

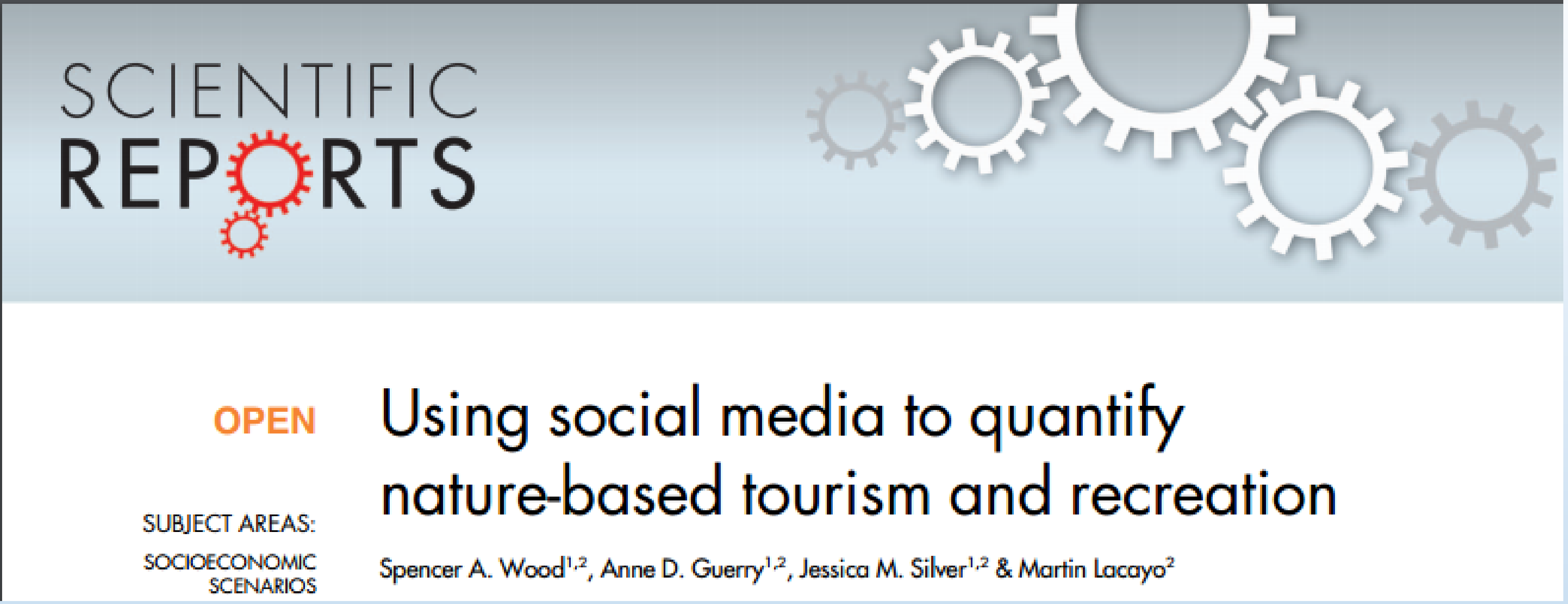
Nature **524**, 8 (06 August 2015) | doi:10.1038/524008a
Published online 05 August 2015

Subject terms: Plant sciences · Biodiversity

A new species of insect-eating sundew plant (*Drosera magnifica*; pictured) has been identified after an amateur naturalist posted photographs of it on Facebook.

```
select *
from analysis_tools
where features @>
'{"Faster queries"
, "Beautiful charts"
, "Easy sharing"}'
```

Editors' pick


SCIENTIFIC REPORTS

OPEN

Using social media to quantify nature-based tourism and recreation

SUBJECT AREAS:
SOCIOECONOMIC SCENARIOS

Spencer A. Wood[1,2], Anne D. Guerry[1,2], Jessica M. Silver[1,2] & Martin Lacayo[2]

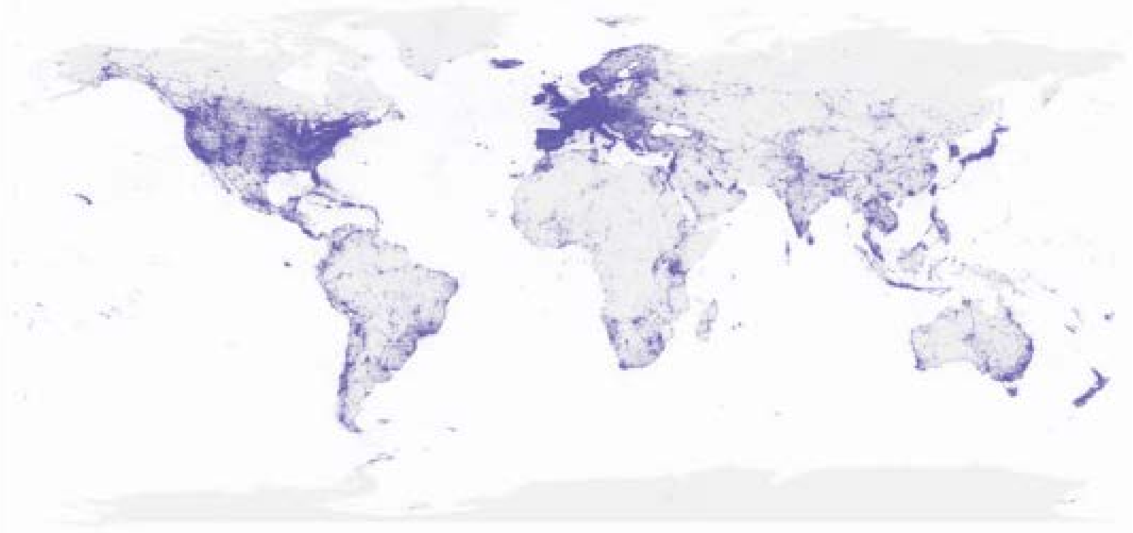
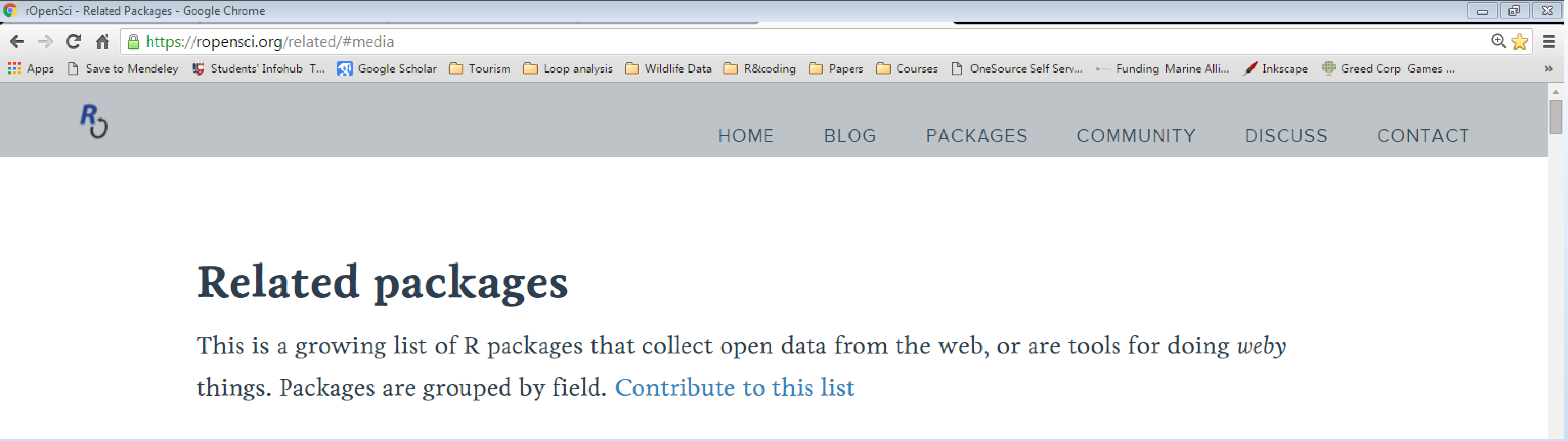


Figure 1 | Locations of the approximately 197 M geotagged photographs uploaded to flickr from 2005–2012. Figure created using the maps package for R.

Application Programming Interface

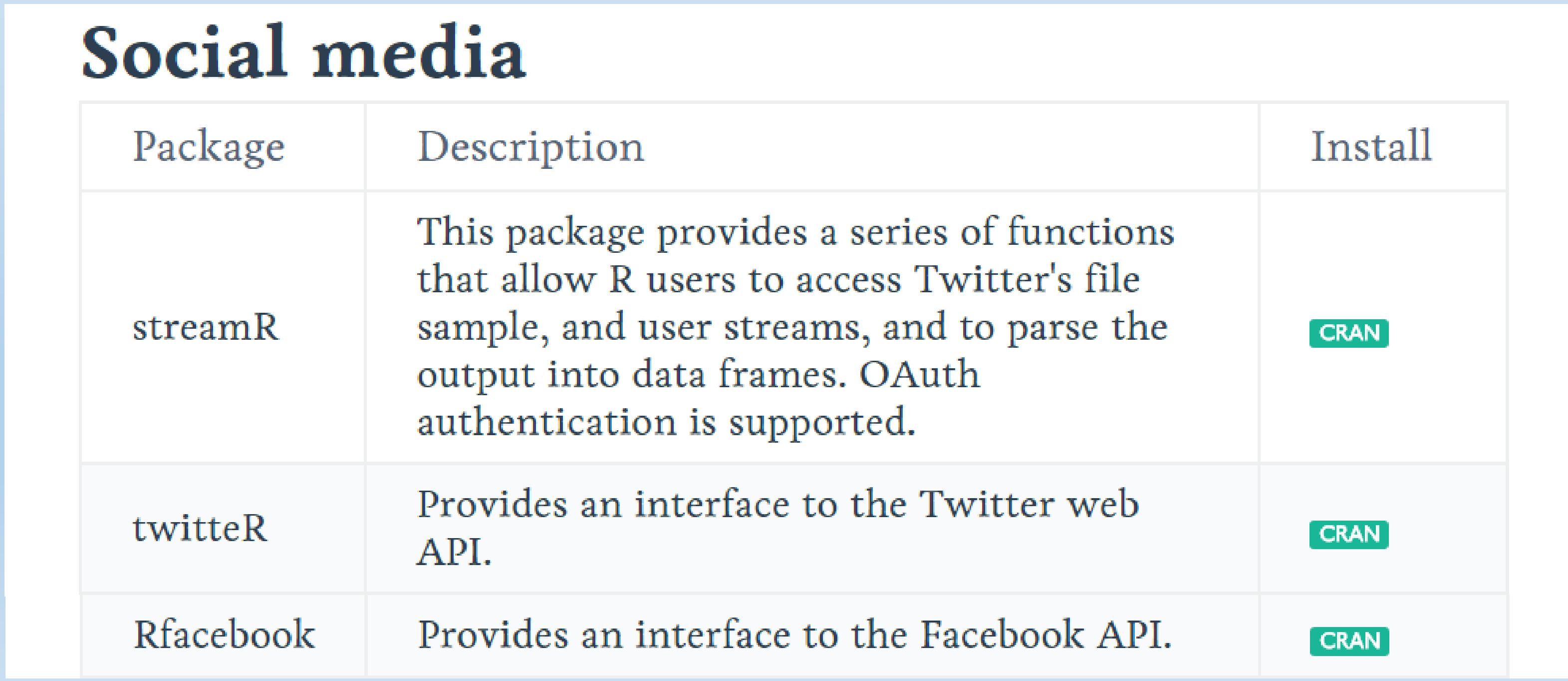
From Wikipedia:

“In computer programming, an **application programming interface** (**API**) is a set of routines, protocols, and tools for building software and applications.”…
… “A good API makes it easier to develop a program by providing all the building blocks, which are then put together by the programmer.”



Related packages

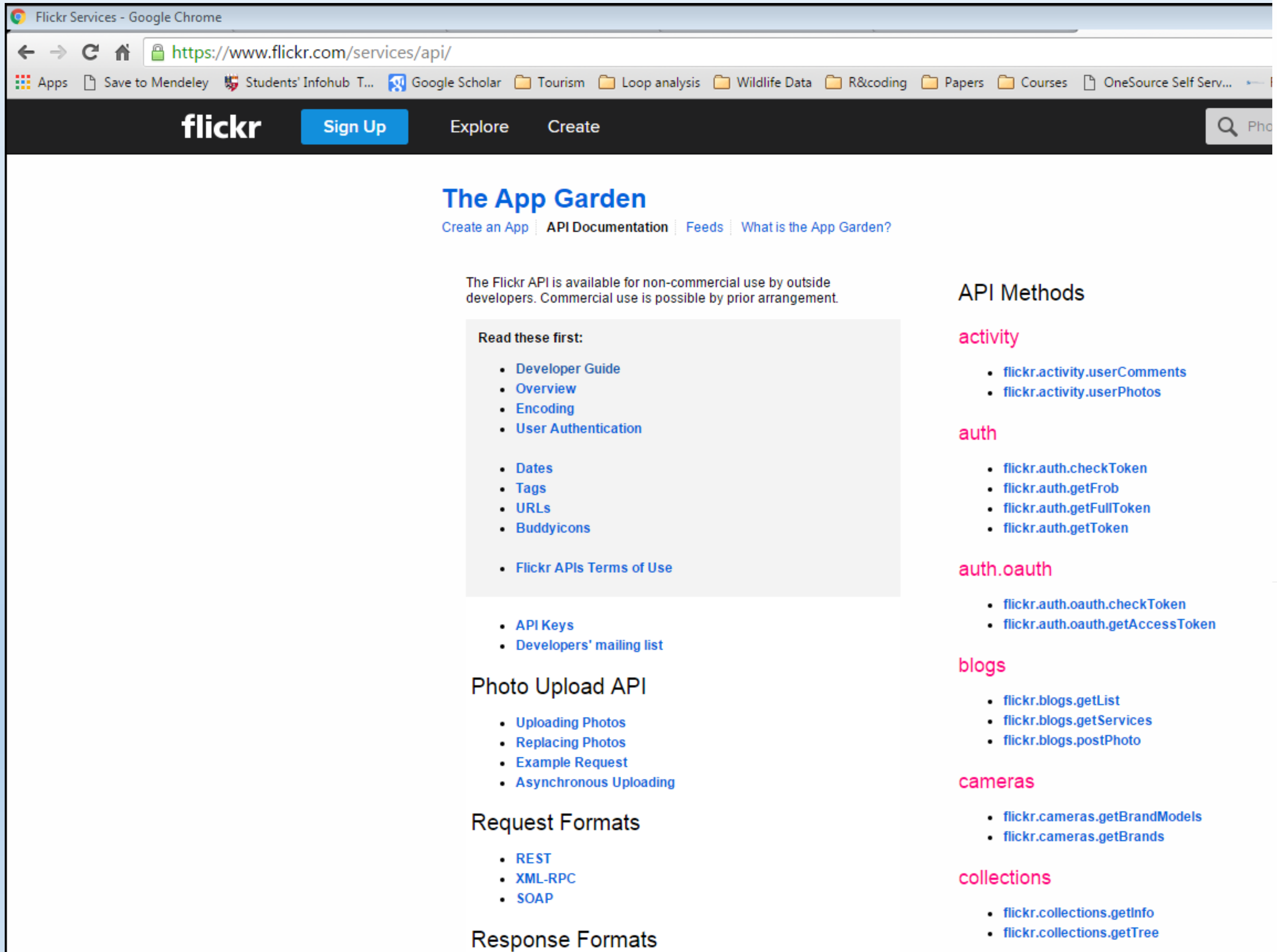
This is a growing list of R packages that collect open data from the web, or are tools for doing *weby* things. Packages are grouped by field. Contribute to this list



Social media

Package	Description	Install
streamR	This package provides a series of functions that allow R users to access Twitter's file sample, and user streams, and to parse the output into data frames. OAuth authentication is supported.	CRAN
twitteR	Provides an interface to the Twitter web API.	CRAN
Rfacebook	Provides an interface to the Facebook API.	CRAN

An example:



The App Garden

Create an App | API Documentation | Feeds | What is the App Garden?

The Flickr API is available for non-commercial use by outside developers. Commercial use is possible by prior arrangement.

Read these first:

- Developer Guide
- Overview
- Encoding
- User Authentication

- Dates
- Tags
- URLs
- Buddyicons

- Flickr APIs Terms of Use

- API Keys
- Developers' mailing list

Photo Upload API

- Uploading Photos
- Replacing Photos
- Example Request
- Asynchronous Uploading

Request Formats

- REST
- XML-RPC
- SOAP

Response Formats

API Methods

activity

- flickr.activity.userComments
- flickr.activity.userPhotos

auth

- flickr.auth.checkToken
- flickr.auth.getFrob
- flickr.auth.getFullToken
- flickr.auth.getToken

auth.oauth

- flickr.auth.oauth.checkToken
- flickr.auth.oauth.getAccessToken

blogs

- flickr.blogs.getList
- flickr.blogs.getServices
- flickr.blogs.postPhoto

cameras

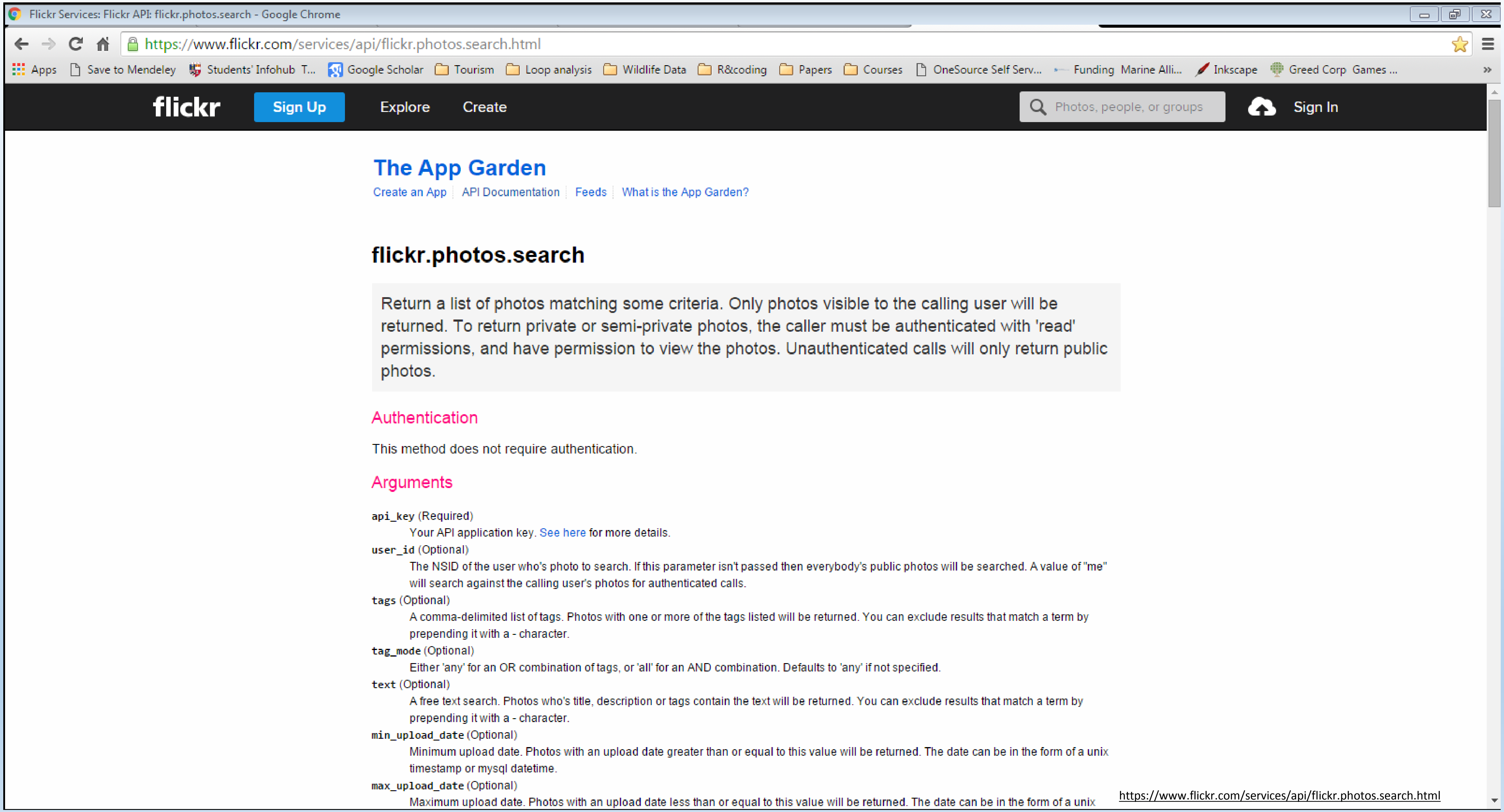
- flickr.cameras.getBrandModels
- flickr.cameras.getBrands

collections

- flickr.collections.getInfo
- flickr.collections.getTree

The Flickr API

With over 5 billion photos (many with valuable metadata such as tags, geolocation, and Exif data), the Flickr community creates wonderfully rich data. The Flickr API is how you can access that data. In fact, almost all the functionality that runs flickr.com is available through the API. And the API is completely free to use, as a service to our members as well as developers and other integrators, so they can create even more ways to interact with photos beyond flickr.com.



The App Garden

Create an App | API Documentation | Feeds | What is the App Garden?

flickr.photos.search

Return a list of photos matching some criteria. Only photos visible to the calling user will be returned. To return private or semi-private photos, the caller must be authenticated with 'read' permissions, and have permission to view the photos. Unauthenticated calls will only return public photos.

Authentication

This method does not require authentication.

Arguments

`api_key` (Required)
Your API application key. See here for more details.

`user_id` (Optional)
The NSID of the user who's photo to search. If this parameter isn't passed then everybody's public photos will be searched. A value of "me" will search against the calling user's photos for authenticated calls.

`tags` (Optional)
A comma-delimited list of tags. Photos with one or more of the tags listed will be returned. You can exclude results that match a term by prepending it with a - character.

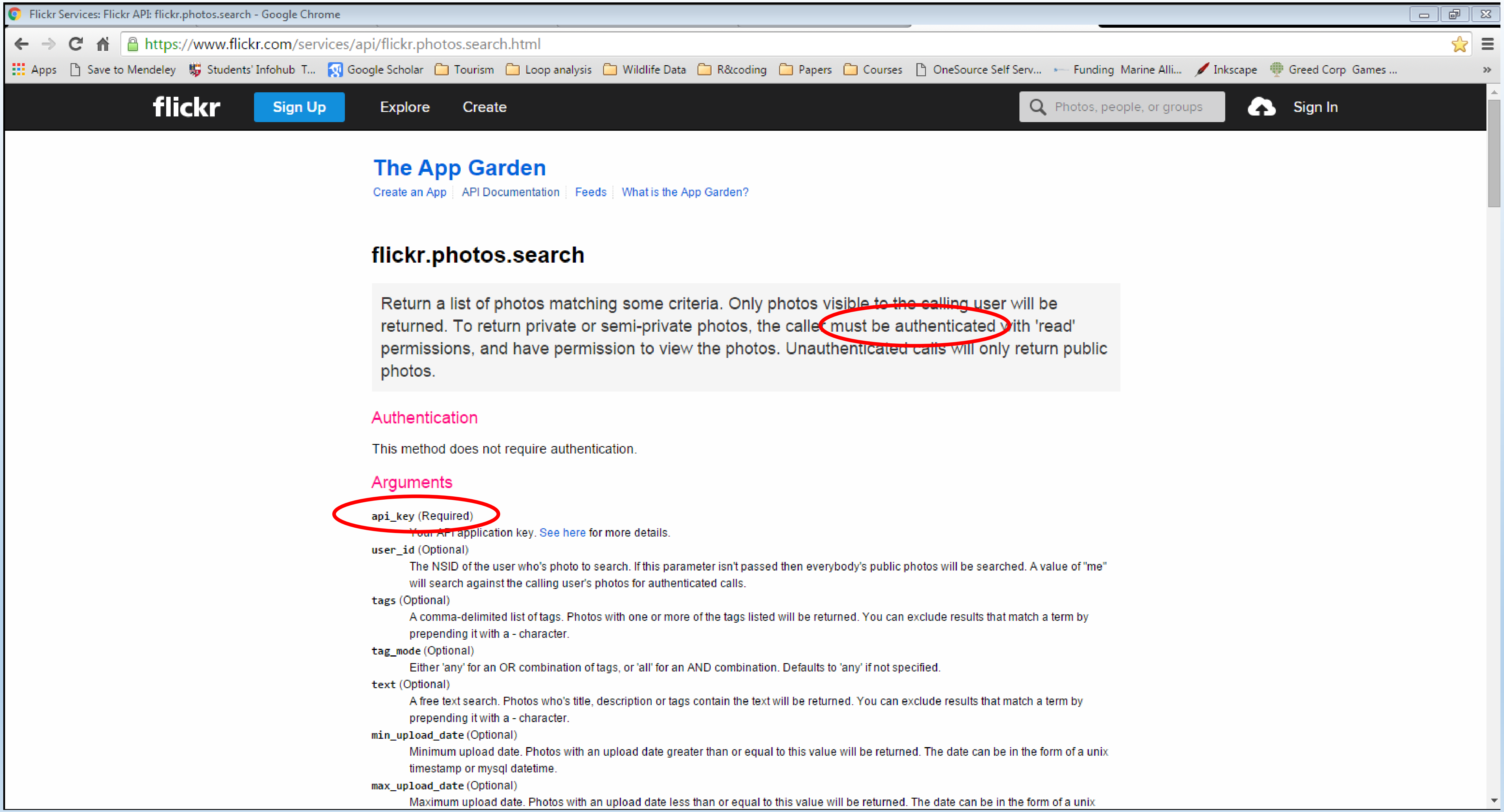
`tag_mode` (Optional)
Either 'any' for an OR combination of tags, or 'all' for an AND combination. Defaults to 'any' if not specified.

`text` (Optional)
A free text search. Photos who's title, description or tags contain the text will be returned. You can exclude results that match a term by prepending it with a - character.

`min_upload_date` (Optional)
Minimum upload date. Photos with an upload date greater than or equal to this value will be returned. The date can be in the form of a unix timestamp or mysql datetime.

`max_upload_date` (Optional)
Maximum upload date. Photos with an upload date less than or equal to this value will be returned. The date can be in the form of a unix

https://www.flickr.com/services/api/flickr.photos.search.html



The App Garden

Create an App | API Documentation | Feeds | What is the App Garden?

flickr.photos.search

Return a list of photos matching some criteria. Only photos visible to the calling user will be returned. To return private or semi-private photos, the caller must be authenticated with 'read' permissions, and have permission to view the photos. Unauthenticated calls will only return public photos.

Authentication

This method does not require authentication.

Arguments

`api_key` (Required)
: Your API application key. See here for more details.

`user_id` (Optional)
: The NSID of the user who's photo to search. If this parameter isn't passed then everybody's public photos will be searched. A value of "me" will search against the calling user's photos for authenticated calls.

`tags` (Optional)
: A comma-delimited list of tags. Photos with one or more of the tags listed will be returned. You can exclude results that match a term by prepending it with a - character.

`tag_mode` (Optional)
: Either 'any' for an OR combination of tags, or 'all' for an AND combination. Defaults to 'any' if not specified.

`text` (Optional)
: A free text search. Photos who's title, description or tags contain the text will be returned. You can exclude results that match a term by prepending it with a - character.

`min_upload_date` (Optional)
: Minimum upload date. Photos with an upload date greater than or equal to this value will be returned. The date can be in the form of a unix timestamp or mysql datetime.

`max_upload_date` (Optional)
: Maximum upload date. Photos with an upload date less than or equal to this value will be returned. The date can be in the form of a unix

Flickr API and R - Authentication

```
library(httr)
library(RCurl)
library(XML)
```

```
myapp <- oauth_app("flickr", key= “your_api_key”, secret= “your_secret”)
#creates the app passing the key and secret

ep <- oauth_endpoint(request=“https://www.flickr.com/services/oauth/request_token”,
                     authorize="https://www.flickr.com/services/oauth/authorize",
                     access="https://www.flickr.com/services/oauth/access_token")
#urls to get authentication credentials from the API

sig <- oauth1.0_token(ep, myapp, cache=FALSE)           #gets authentication credentials

fl_sig <- sign_oauth1.0(myapp, sig)

baseURL <- paste("https://api.flickr.com/services/rest/?method=flickr.photos.search&api_key=",
          api_key, sep="")
```

Flickr API and R – The Request

```
getPhotos <- paste(baseURL, ,"&tags=bird“, “&format=rest”, sep=“”)

gURL<-getURL(getPhotos, ssl.verifypeer=FALSE, useragent = "flickr")

gURL
#[1] "<?xml version=\"1.0\" encoding=\"utf-8\" ?>\n<rsp stat=\"ok\">\n<photos page=\"4\"
#pages=\"8\" perpage=\"250\" total=\"1896\">
#\n\t<photo id=\"423916810\" owner=\"31625633@N00\" secret=\"7354f644c9\" server=\"153\"
#farm=\"1\" title=\"Bird Display\" ispublic=\"1\
#" isfriend=\"0\" isfamily=\"0\" datetaken=\"2006-09-12 13:59:14\" datetakengranularity=\"0\"
#datetakenunknown=\"0\"
#tags=\"bird animal museum stuffed education edinburgh university gallery naturalhistory taxidermy
#research stuffedanimals labs
#teaching edinburghuniversity biology naturalhistorymuseum specimens taxidermist
#...

class(gURL)
#[1] "character"
```

Flickr API and R – Parsing

```
parsed_data <- xmlRoot(xmlTreeParse(gURL, useInternalNodes = TRUE ))
#parses the data and extracts the root node

parsed_data

#<rsp stat="ok">
#<photos page="4" pages="8" perpage="250" total="1896">
# <photo id="423916810" owner="31625633@N00" secret="7354f644c9" server="153"
#farm="1" title="Bird Display" ispublic="1" isfriend="0" isfamily="0" datetaken="2006-09-12
#13:59:14" datetakengranularity="0" datetakenunknown="0" tags="bird animal museum
#stuffed education edinburgh university gallery naturalhistory taxidermy research
#stuffedanimals labs teaching edinburghuniversity biology naturalhistorymuseum specimens
#taxidermist universityofedinburgh ashworth kingsbuildings birddisplay ashworthlabs"
#latitude="55.924140" longitude="-3.173182" accuracy="15" context="0"
#place_id="ohOIsflVUby_lg" woeid="43668" geo_is_family="0" geo_is_friend="0"
#geo_is_contact="0" geo_is_public="1"/>
 #...
```

Flickr API and R – Extracting Info

```
id <- xpathSApply(parsed_data, "//photo", xmlGetAttr, "id")

owner <- xpathSApply(parsed_data, "//photo", xmlGetAttr, "owner")

datetaken <- xpathSApply(parsed_data, "//photo", xmlGetAttr, "datetaken")

tags <- xpathSApply(parsed_data, "//photo", xmlGetAttr, "tags")

latitude <- xpathSApply(parsed_data, "//photo", xmlGetAttr, "latitude")

longitude <- xpathSApply(parsed_data, "//photo", xmlGetAttr, "longitude")
```

Finally a dataframe!

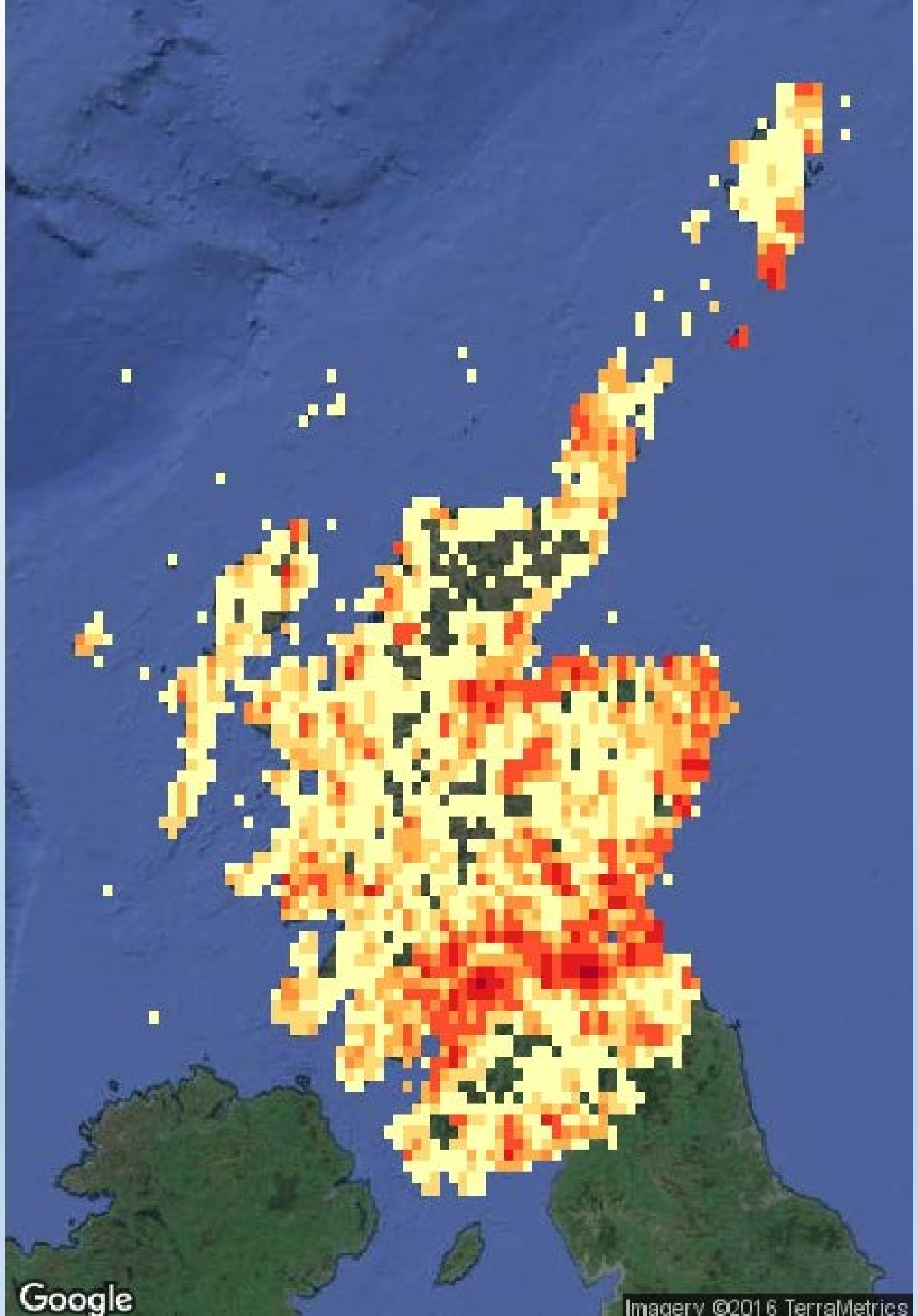
```
df <- data.frame(cbind(id, owner, datetaken, tags, latitude,longitude), stringsAsFactors=FALSE)

str(df)

#'data.frame':   249 obs. of  6 variables:
# $ id       : chr  "423916810" "423916760" "423916709" "423450425" ...
# $ owner    : chr  "31625633@N00" "31625633@N00" "31625633@N00" "34277201@N00" ...
# $ datetaken: chr  "2006-09-12 13:59:14" "2006-09-12 13:59:34" "2006-09-12 13:50:12" "2006-10-05...
# $ tags     : chr  "bird animal museum stuffed education edinburgh university gallery naturalhistory...
# $ latitude : chr  "55.924140" "55.924140" "55.924140" "55.928432" ...
# $ longitude: chr  "-3.173182" "-3.173182" "-3.173182" "-4.324235" ...
```

Lessons learnt

- 1000+ ways to do it
- Read the API documentation!
- Explore the HTML/XML code
- Don’t give in to frustration!


Google
Imagery ©2016 TerraMetrics