# ISSR Short Course

Gregory J. Matthews [1]

[1]Department of Mathematics and Statistics
Loyola University Chicago

June 2015

# Outline

- API stands for application programming interface.
- APIs can be used in many ways.
- One use of APIs is to access databases
- We'll look at two examples:
  - U.S. Census
  - Twitter

# Census API

- First we'll look at the Census API.
- We're not going to be scraping any text here, but this is a simple example of how an API works.
- First thing we need is a key to interact with the API.
- We can request a Census API key here: http://api.census.gov/data/key_signup.html

- Once we have a Census key, we can now access the Census API.
- First let's find data table that we are interested in.
- A list of available data tables can be found here: http://api.census.gov/data.html

- We need the root URI for 2010 Decennial Census summary file 1.
- We can modify this URI in ways that allow us to generate the specific table that we are interested in.
- Census codes can be found here: http://starr.tamu.edu/files/2013/01/Census-Codes.pdf
- Total Population is table P0010001.
- Let's access the API using R.

```r
#Request a key URL: http://api.census.gov/data/key_signup.html
#Root URl: http://api.census.gov/data/2010/sf1
#Need a key for this to work:
#http://api.census.gov/data/2012/acs5/profile?get=DP02_0001PE&for=state:*
#First, specify my key
key<-"e818359af387a7b903f44810d34e9d7a3c7c07f7"
#Specify the table I am interested in.
#This is the total population table.
get<-"P0010001"
#Define the URL
url<-paste("http://api.census.gov/data/2010/sf1?key=",
           key,"&get=",get,",NAME&for=state:*",sep="")
#Read in the contents of the URL.
aaa<-readLines(url)
#Remove the [].
do.call(rbind,strsplit(aaa,'\",\"'))[1:5,]
```

```
#Total Population table
get<-"P0010001"
#Specify only Massachusetts (state=25)
url<-paste("http://api.census.gov/data/2010/sf1?key=",
           key,"&get=",get,",NAME&for=place:*&in=state:25"
aaa<-readLines(url)
check<-do.call(rbind,strsplit(aaa,'\",\"'))[1:5,]
```

- Now we're going to access Twitter's API which will allow us to collect tweets meeting some criteria.
- As with the Census API, we will need a key to access the Twitter API.
- For Twitter, we actually need a consumer key and a consumer secret. Twitter requires both of these to access their API.
- Information on obtaining a Twitter API key can be found here: https://dev.twitter.com/docs/faq#7447

```
library(tm)
library(twitteR)
library(RCurl)
library(ROAuth)
consumerKey<-"4pIbFQLAtkY2U2QckpkogLTaY"
consumerSecret<-"Gae7HMTmuVtgZDRyU6BVrdFwwcWQfKew8TTzRUIi9Fth44Lg8O"
accessToken<-"24096463-Dl4PGRWKSXQEysjSSSXhWjMT8evTEBraxNKLSAN4Q"
accessSecret<-"sHz6hJPw1AZMAWNoCylOOOXOy5kYHlaqcvpdlg8Xxw9eV"

setup_twitter_oauth(consumerKey, consumerSecret,
      access_token=accessToken,access_secret=accessSecret)
```

```
a<-getUser('statsinthewild')
a$statusesCount
a$followersCount
a$friendsCount
a$description
#for multiple user lookups
a<-lookupUsers(c('statsinthewild','statsclass'))
a$statsinthewild$description
a$statsclass$description
```

```r
#Now let's search twitter
#Get the last 50 Tweets with the word "summer" in them
stuff<-searchTwitter("summer",n=50)
#Create a data frame
stuffDF<-do.call(rbind,lapply(stuff,as.data.frame))
#Keep only words and hashtags.
stuffDF$text<-gsub("[^A-z #?.!,]","",stuffDF$text)
```

```
myCorpus <- Corpus(VectorSource(stuffDF$text))
myCorpus<-tm_map(myCorpus,content_transformer(tolower))
myCorpus<-tm_map(myCorpus,removeWords,
                 stopwords("english"))
#This will remove hash tags.  May not be what we want.

myCorpus<-tm_map(myCorpus,removePunctuation)
myCorpus<-tm_map(myCorpus,removeNumbers)
myCorpus<-tm_map(myCorpus,stripWhitespace)
```

```r
#We can define our own functions to do what we want.
#This will remove any thing in [!.,?]
#However, this will keep hashtags intact.
removePunctuationTwitter<-function(x){
  out<-gsub('[!.,?]',"",x)
  out
}
```

```
myCorpus <- Corpus(VectorSource(stuffDF$text))
myCorpus<-tm_map(myCorpus,content_transformer(tolower))
myCorpus<-tm_map(myCorpus,removeWords,
                 stopwords("english"))
#Everything the same, but now I am
#calling a new function that I defined
myCorpus<-tm_map(myCorpus,removePunctuationTwitter)
myCorpus<-tm_map(myCorpus,removeNumbers)
myCorpus<-tm_map(myCorpus,stripWhitespace)
#Important final step
myCorpus<-tm_map(myCorpus, PlainTextDocument)
```

```r
#Get term frequency
sort(termFreq(PlainTextDocument(myCorpus)))
#Create a TermDocumentMatrix
twitterTDM<-TermDocumentMatrix(myCorpus)
#library(proxy)
#Check disimilarity between Tweets
proxy::dist(as.matrix(t(twitterTDM)),method="eJaccard")
#Find Associated words
findAssocs(twitterTDM,terms=c("summer"),corlimit=0.1)
```

Let's do it:

- If you don't have one already:
  - Get a Twitter account
  - Get a Twitter key
- Access the Twitter API
- Pick a key word and scrape 100 Tweets containing that word.