Public and Media Attention

CuriosityBits Data Lab
Dec 12, 2016 · 4 min read

Say I am interested in how much public attention and media interest Smithsonian Institute has garnered for the past year. I can track the search volume using Google Trends, and count the available news articles in Google News. I am writing this post to show how I use the available Google data points to score organizations in R.

The measurement I've implemented takes in three data points, each tapping into a separate facet of public and media attention.

1. **The Google search volume from Google Trends—**this is a direct gauge of how much interest the public has in an organization.
2. **The number of results from the Google search**—this shows the visibility of an organization on the web.
3. **The number of results in the Google news search**—this indicates how much news coverage an organization has received.

Mining the Google Trends Data

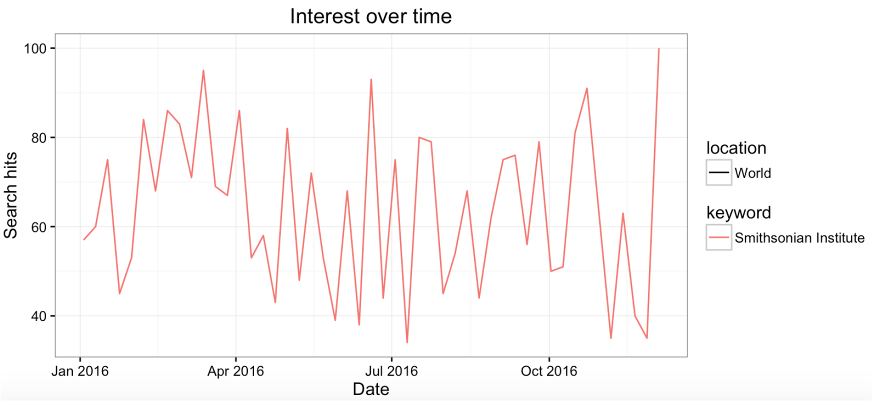
gtrendsR is the R library for Google Trends data.

```
install.packages(“gtrendR”)
library(gtrendsR)

#use your google account to authenticate the access.

user <- “xxxx@gmail.com” #your google account username
psw <- “xxxxx” #enter your password
gconnect(user, psw)
```

After setting up gtrendsR, use the following two lines to generate a trend plot for Smithsonian Institute.

```
lang_trend <– gtrends(“Smithsonian
Institute”,start_date=”2016–01–01")
```

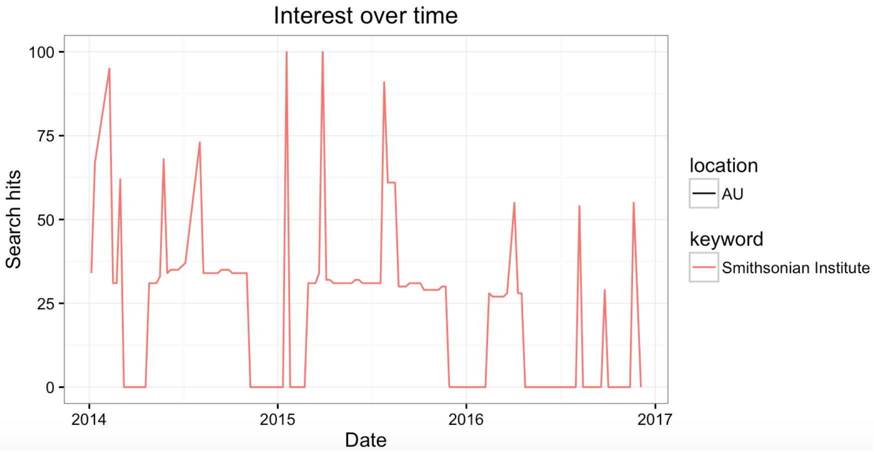
```
plot(lang_trend)
```



gtrendsR allows us to specify the search by geographic locations and time frames. For instance, I can tweak the code slightly to get a plot for the search volume in **Australia** since **2014–01–01**. (note: a list of country code)

```
lang_trend <– gtrends(“Smithsonian Institute”,geo = “AU”,
start_date=”2014–01–01")
```

```
plot(lang_trend)
```



We then calculate the sum, mean and median of the search volume.

```
sum(lang_trend$trend$hits)
mean(lang_trend$trend$hits)
median(lang_trend$trend$hits)
```

Getting the Count of Google Search Results

We will use two libraries: RCurl and XML.

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

To simply describe the process: each Google search has an URL to it. For instance, if we search "Smithsonian Institute", the URL is: https://www.google.com/#q=%22Smithsonian+Institute%22 . Notice that the search term is enclosed with quotation marks "" so that Google returns those pages that match your search term exactly. in the URL, the quotation marks are represented by %22. Using the URL we can retrieve the XML content of the returned search page. We then parse through the XML and get the search result count. You can run the following block of code to get the count for "Smithsonian Institute".

```
n<- “Smithsonian Institute”

url<-paste0(“http://www.google.com/search?q=%22",gsub("
“,”+”,n),”%22")

search.html<-getURL(url)

parse.search<-htmlTreeParse(search.html,useInternalNodes =
TRUE)

search.nodes<-
getNodeSet(parse.search,”//div[@id=’resultStats’]”)

search.value<-strsplit(xmlValue(search.nodes[[1]]),”
“,fixed=TRUE)[[1]][2]

count <- as.numeric(gsub(“,”,””,search.value,fixed=TRUE))

count
```

To get the research count from Google News, the tweak is incredibly easy. We just need to add **&tbm=nws** to the end of a URL. That is:

```
url<-paste0(“http://www.google.com/search?q=%22",gsub("
“,”+”,n),”%22&tbm=nws")
```

Building R functions

Let’s start with the R function for Google Result Counts. We name the function **google.counts**.

```
google.counts<-function(n){

 url1<-paste0(“http://www.google.com/search?q=%22",gsub("
“,”+”,n),”%22")
 search.html1<-getURL(url1)
 parse.search1<-htmlTreeParse(search.html1,useInternalNodes
= TRUE)
 search.nodes1<-
getNodeSet(parse.search1,”//div[@id=’resultStats’]”)
 search.value1<-strsplit(xmlValue(search.nodes1[[1]]),”
“,fixed=TRUE)[[1]][2]
 exact.count <-
as.numeric(gsub(“,”,””,search.value1,fixed=TRUE))

 url2<-paste0(“http://www.google.com/search?q=%22",gsub("
“,”+”,n),”%22&tbm=nws”)
 search.html2<-getURL(url2)
 parse.search2<-htmlTreeParse(search.html2,useInternalNodes
= TRUE)
 search.nodes2<-
getNodeSet(parse.search2,”//div[@id=’resultStats’]”)
 search.value2<-strsplit(xmlValue(search.nodes2[[1]]),”
“,fixed=TRUE)[[1]][2]
 news.count <-
as.numeric(gsub(“,”,””,search.value2,fixed=TRUE))
 result <- c(exact.count, news.count)


 return(result)
}
```

We can apply this function to a search term. As you can see, the function returns two values. The first is the Google Search count and the second is the count from Google News.

```
google.counts(“Smithsonian Institute”)
```

Let’s apply the google.count function to a data frame containing a list of organizations. In my example, the data frame is named ***sheet***. the names of the organizations are in the first column named ***OrganizationNames*** (see below). Here is how I manipulate the R code to score each organization.

```
sheet$GoogleSearchCount <- “NA”
sheet$GoogleNewsCount <- “NA”
```

```
#score from the 1st to the 100th organization.


for (name in sheet$OrganizationName[1:100]){
 print(paste0(“getting the data for:”,name))
 Sys.sleep(3)
 result <- google.counts(name)
 sheet[sheet$OrganizationName==name,]$GoogleSearchCount <-
result[1]
 sheet[sheet$OrganizationName==name,]$GoogleSearchCount <-
result[2]
}
```

Next, let’s build a R function for Google Trend data. We call this function **trend.counts**.

```
trend.counts<-function(n,geo,start_date,end_date){

output <- data.frame()
 lang_trend <-
gtrends(n,geo=geo,start_date=start_date,end_date=end_date)
 sum <- sum(lang_trend$trend$hits)
 mean <- mean(lang_trend$trend$hits)
 median <- median(lang_trend$trend$hits)
 output <- c(sum,mean,median)
 return(output)
}
```

This function returns three values: sum, mean, and median, respectively. You can specify geographic location, start date and end date in the function. Say you want to calculate the **worldwide** search volume for Smithsonian Institute from 2016–03–01 to 2016–06–01. Try:

```
trend.counts(“Smithsonian Institute”,””,”2016-03-01",”2016-
06-01")
```

If you are interested in the search volume in UK. Try:

```
trend.counts(“Smithsonian Institute”,”GB”,”2016-03-
01",”2016-06-01")
```

Applying this function to a list of organizations is simple as well.

```
sheet$trend_sum <- “1”
sheet$trend_mean <- “1”
```

```
sheet$trend_median <- “1”


for (name in sheet$OrganizationName[1:50]){
 print(paste0(“getting the data for:”,name))
 Sys.sleep(3)
 try(result <- trend.counts(name,””,”2014-03-01",”2016-06-
01")) #change parameters to suit your search need
 sheet[sheet$OrganizationName==name,]$trend_sum <- result[1]
 sheet[sheet$OrganizationName==name,]$trend_mean <-
result[2]
 sheet[sheet$OrganizationName==name,]$trend_median <-
result[3]
}
```