Getting location information with the Google Maps API

Han Oostdijk (www.hanoostdijk.nl)

March 10, 2016

Introduction

Google Maps is well known as an interface to view maps. In this document we use the Google Maps Geocoding API to lookup addresses. For more than incidental work a license could be necessary.

We will use R to do this but it can also be done in other environments such as e.g. javascript, Python or PHP. We show how the API returns an XML-file as response to an address input. The input can be a full address or a uniquely identifying part of it. It is assumed that the XML-file contains at most one result. For more than one result a loop over the result item would be needed as shown in example 6 in the Example of use section. On special request this could be done :) .

Used libraries

```
library(xml2)
library(magrittr)
library(xtable)
```

Utility functions

Request an XML document from the API

The **read_address** function actually reads the information from the Google website. It returns an XML-document. When you want to see the contents of the document, uncomment the line with **write_xml**.

```
read_address <-function(address) {
  url <- paste0('http://maps.google.com/maps/api/geocode/xml',
                '?sensor=false&address=',url_escape(address))
  doc <- read_xml(url)
  # write_xml(doc, 'temp.xml')
  return(doc)
}
```

Get type(s) from document

The XML document contains elements that describe an attribute of the address such as the country, administrative area (in the Netherlands a province) or location. The *type* tag indicates which attribute it concerns. The **address_data** function select a list of types from the document and uses the **get_type** function for each of these. For some addresssen not all types are present: e.g. when the address is a country, the postcode is not present. The line with *gsub* handles that. *Types* is a list with in the first element the types that are collected and in the second one the names that they will be assigned.

```
get_type <-function(tt,type_nodes) {
  z=grepl(tt,sapply(type_nodes,xml_text))
  xml_text(xml_find_all(xml_parent(type_nodes[z]),"./long_name"))
}
```

```
address_data <-function(doc,types) {
  type_nodes = xml_find_all(doc, ".//type")
  w=sapply(types$i,function(x) {get_type(x,type_nodes)})
  w=as.character(w); w=gsub("character(0)","",w,fixed=T)
  names(w)=types$o
  return(w)
}
```

Get coordinates from document

The coordinates are given in **lat** (latitude or *breedte* in Dutch) and **long** (longitude or *lengte*). The document contains (in my experience) three sets of these pairs. The first set is found in the element *location* and indicates the coordinates of the centre of the address. The second set is found in the element *viewport/southwest* and indicates the coordinates of the southwest corner of the address and the third one indicates the northeast corner of the address. The **tc** parameter indicates the element and the **n** parameter the names these coordinates will get.

```
coord_data <-function(doc,tc='location',n=c('lat','lng')) {
  loc = xml_find_all(doc, paste0(".//",tc))
  lat = loc %>% xml_find_all(".//lat") %>% xml_text()
  lng = loc %>% xml_find_all(".//lng") %>% xml_text()
  w   = c(lat,lng)
  names(w)=n
  w
}
```

Print data.frame

The function **print_adress** prints a data.frame . In this example it is used to print the results of the examples. Apart from the data.frames there are parameters to indicate the LaTeX label and caption and the number of digits that will be shown.

```
def_tab <- function (label_name,label_tekst)  {
  paste0(label_tekst,"\\label{table:",label_name,"}")
}

print_adress <- function (df,lbl,cap,digits=3) {
  print(xtable(df,caption=def_tab(lbl,cap),digits=digits),
    rownames=F, table.placement='!htbp')
}
```

Get all data for an address

The **all_data** function combines the utility functions and creates a character vector with the attributes of the address and all its coordinates. When the API does not return results (maybe an invalid address was specified) the function does not return a vector but the boolean value FALSE.

```
all_data <- function(address,n=c('lat','lng')) {
  doc = read_address(address)
  if (xml_find_all(doc, ".//status") %>% xml_text() == "ZERO_RESULTS") {
    return(F)
  }
  types = list(i=c('postal_code','locality',
```

```
                      'administrative_area_level_1','country'),
               o=c('postcode','location','level2','country') )
  a1=address_data(doc,types)
  c1=coord_data(doc,n=n)
  c2=coord_data(doc,'viewport/southwest',n=paste0('sw_',n))
  c3=coord_data(doc,'viewport/northeast',n=paste0('ne_',n))
  c(a1,c1,c2,c3)
}
```

Example of use

In this example we give the results from 6 different calls to the API on page 4:

- with the full address in Dutch. Results in Table 1. NB: the country indication is in English.
- with only the numeric part of the postcode and the country indication *NL*. Results in Table 2.
- with only the location (*Amstelveen*) and the country indication *NL*. Results in Table 3.
- with only the country (*Nederland*) and the country indication *NL*. Results in Table 4.
- with (what I thought to be) an invalid address. Apparently the API links this to the country *Italy*. Results in Table 5.
- with only the country (*Nederland*) and without the country indication. In the introduction we already said that this code currently works only when there is at most one result. The result of *print(e6)* shows garbled output.

```
e1=all_data('Runmoolen 24 Amstelveen Nederland')
e2=all_data('1181 NL')
e3=all_data('Amstelveen NL')
e4=all_data('Nederland NL')
e5=all_data('X Y Z')
e6=all_data('Nederland')
print(e6)
```

```
                                    postcode
                                     "80466"
                                    location
                "c(\"Nederland\", \"Nederland\")"
                                       level2
                    "c(\"Texas\", \"Colorado\")"
                                     country
```

"c("Netherlands", "United States", "United States")" lat "52.1326330" lng "29.9743803" "39.9613759" "5.2912660" "-93.9923965" "-105.5108312" sw_lat "50.7503837" sw_lng "29.9465649" "39.9549809" "3.3316000" "-94.0419501" "-105.5233440" ne_lat "53.6756000" ne_lng "30.0189810" "39.9753149" "7.2271405" "-93.9629590" "-105.4841760"

Convert to data.frame

We convert the vectors in e1 . . . e5 to a data.frame for a better presentation.

```
e = matrix(c(e1,e2,e3,e4,e5),nrow=5,byrow=T)
e = data.frame(pc=e[,1],loc=e[,2],prov=e[,3],cntr=e[,4],
        lat=as.numeric(e[,5]),   long=as.numeric(e[,6]),
        slat=as.numeric(e[,7]), wlong=as.numeric(e[,8]),
        nlat=as.numeric(e[,9]), elong=as.numeric(e[,10])   )
```

Print the rows of the data.frame (separately)

By printing the rows separately we can give them individually a caption for better readability.

```
print_adress(e[1,],'e1',cape1,digits=2)
print_adress(e[2,],'e2',cape2,digits=2)
print_adress(e[3,],'e3',cape3,digits=2)
print_adress(e[4,],'e4',cape4,digits=2)
print_adress(e[5,],'e5',cape5,digits=2)
```

	pc	loc	prov	cntr	lat	long	slat	wlong	nlat	elong
1	1181 NZ	Amstelveen	Noord-Holland	Netherlands	52.31	4.87	52.31	4.86	52.31	4.87

Table 1: results for address 'Runmoolen 24 Amstelveen Nederland'

	pc	loc	prov	cntr	lat	long	slat	wlong	nlat	elong
2	1181	Amstelveen	Noord-Holland	Netherlands	52.31	4.86	52.30	4.85	52.32	4.87

Table 2: results for address '1181 NL'

	pc	loc	prov	cntr	lat	long	slat	wlong	nlat	elong
3		Amstelveen	Noord-Holland	Netherlands	52.31	4.87	52.24	4.79	52.33	4.91

Table 3: results for address 'Amstelveen NL'

	pc	loc	prov	cntr	lat	long	slat	wlong	nlat	elong
4				Netherlands	52.13	5.29	50.75	3.33	53.68	7.23

Table 4: results for address 'Nederland NL'

	pc	loc	prov	cntr	lat	long	slat	wlong	nlat	elong
5				Italy	41.87	12.57	35.49	6.63	47.09	18.52

Table 5: results for address 'X Y Z'

Session Info

```
sessionInfo()
```

```
## R version 3.2.0 (2015-04-16)
## Platform: x86_64-w64-mingw32/x64 (64-bit)
## Running under: Windows 8 x64 (build 9200)
##
## locale:
## [1] LC_COLLATE=English_United States.1252  LC_CTYPE=English_United States.1252
## [3] LC_MONETARY=English_United States.1252 LC_NUMERIC=C
## [5] LC_TIME=English_United States.1252
##
## attached base packages:
## [1] stats     graphics  grDevices utils     datasets  methods   base
##
## other attached packages:
## [1] knitr_1.12.3 xtable_1.8-2 magrittr_1.5 xml2_0.1.2
##
## loaded via a namespace (and not attached):
##  [1] formatR_1.2.1   tools_3.2.0     htmltools_0.2.6 curl_0.9.4      yaml_2.1.13
##  [6] Rcpp_0.12.2     stringi_1.0-1   rmarkdown_0.9.2 stringr_1.0.0   digest_0.6.8
## [11] evaluate_0.8
```