Benaderen Open Data van CBS vanuit R

Han Oostdijk (www.hanoostdijk.nl)

1 april 2016

Gebruikte R libraries

```
library(curl)
library(magrittr)
library(XML)
library(dplyr)
```

Inleiding

Het is nog maar gedeeltelijk gelukt toegang te krijgen tot de CBS open data. Ik heb nog de volgende vragen:

- hoe krijg ik grote bestanden binnen (meer dan 10000 regels)?
  Sinds de eerste versie van dit document is deze vraag beantwoord: je kunt aan de url toevoegen een constructie als *?$skip=10000&$top=200* om de records 10001 t.m. 10200 te lezen.
- hoe doe ik een query?
  Toevoegen aan de url van *?$top=10&$skip=3&$filter=substring(Perioden,0,4) eq '2013'* werkt niet voor de CBS server. Een soortgelijke constructie werkt wel voor http://services.odata.org/OData/OData.svc/Products met *?$top=5&$skip=3&$filter=Price le 20*.

Hieronder laat ik zien tot hoever ik ben gekomen met het inlezen van een CBS tabel die aangeeft hoeveel mensen (gesplitst naar geslacht en leeftijd) er in bepaalde jaren gebruik maken van bepaalde geneesmiddelen (groepen). De tabel heeft als identificatie **81071NED** en wordt omschreven als *Personen met verstrekte geneesmiddelen; leeftijd en geslacht*.

Gebruikte documentatie

De file 2014handleidingcbsopendataservices.pdf bevat informatie over de CBS open data omgeving. Er wordt onderscheid gemaakt tussen de *API* en de *FEED* omgeving, maar het document omvat beide. Het document wijst ook naar de catalogus waarin je kunt vinden welke tabellen aanwezig zijn.
NB: *geef je als zoekargument '81071NED' dan wordt de informatie getoond die hoort bij '81072NED' ??*

Constanten

Voor enkele functies is soms nodig de *namespaces* aan te geven. Ik doe dat dus maar waarom soms wel en soms niet, is mij niet duidelijk. Ook gebruik ik constanten om aan te geven met welke tabellen we aan de gang gaan.

```
std_namespaces = c(ns="http://www.w3.org/2005/Atom",
  m="http://schemas.microsoft.com/ado/2007/08/dataservices/metadata",
  d="http://schemas.microsoft.com/ado/2007/08/dataservices")

myroot   = "http://opendata.cbs.nl/ODataFeed/OData"
mytable  = "/81071NED"
```

Basis lees functies

Om data van de CBS server te halen gebruiken de **get_cbs_data** functie. Na ophalen wordt de data omgevormd naar een *XMLInternalDocument* object en desgewenst locaal opgeslagen.

```
get_cbs_data <- function (root, table_name=NULL, save_file_name = NULL) {
  if (!is.null(table_name)) {
    f  = curl_escape(table_name)
    f  = paste0(root, f)
  } else{
    f  = root
  }
  r  = curl_fetch_memory(f)
  x  = rawToChar(r$content)
  doc = xmlParse(x,asText =T)
  if (!is.null(save_file_name)) {
    saveXML(doc, save_file_name)
  }
  return(doc)
}
```

De url die wordt gevormd door de identificatie aan de root vast te knopen (in dit geval "http://opendata.cbs.nl/ODataFeed/OData/81071NED") levert een xml document op met referenties naar de onderliggende tabellen. Die referenties halen we eruit met de **get_cbs_table_info** functie en stoppen we in variable **x1** wat een *named character vector* is. In Table 1 on page 2 geven we die weer in tabel vorm.

```
get_cbs_table_info <- function(doc) {
  m1    = xpathSApply(t1,"//@href/..",
    function(x) c(xmlValue(x), xmlAttrs(x)[["href"]]))
  hrefs = m1[2,]
  names(hrefs) =m1[1,]
  return(hrefs)
}

t1    = get_cbs_data(myroot,mytable)
x1    = get_cbs_table_info(t1)
```

key	href
TableInfos	http://opendata.cbs.nl/ODataFeed/OData/81071NED/TableInfos
UntypedDataSet	http://opendata.cbs.nl/ODataFeed/OData/81071NED/UntypedDataSet
TypedDataSet	http://opendata.cbs.nl/ODataFeed/OData/81071NED/TypedDataSet
DataProperties	http://opendata.cbs.nl/ODataFeed/OData/81071NED/DataProperties
Geslacht	http://opendata.cbs.nl/ODataFeed/OData/81071NED/Geslacht
Leeftijd	http://opendata.cbs.nl/ODataFeed/OData/81071NED/Leeftijd
GeneesmiddelengroepATC	http://opendata.cbs.nl/ODataFeed/OData/81071NED/GeneesmiddelengroepATC
Perioden	http://opendata.cbs.nl/ODataFeed/OData/81071NED/Perioden

Table 1: Information in http://opendata.cbs.nl/ODataFeed/OData/81071NED

We weten nu dus welke onderliggende tabellen er zijn. Zo vinden we de informatie over de *Geslacht* codering in **x1['Geslacht']** ofwel http://opendata.cbs.nl/ODataFeed/OData/81071NED/Geslacht.

Functies voor verwerken van tabellen

Uit Table 1 on page 2 kunnen we op het oog al een beetje zien welke de data tabellen en welke de coderings tabellen zijn. (Er is ook nog de *TableInfos* met een beschrijving maar die laat ik nu buiten beschouwing.) We

kunnen dat ook precies zien in de *DataProperties* tabel die ik in Table 2 on page 4 weergeef zonder de (brede) *Description* en de *ParentID* kolom. Alle tabellen worden gelezen met de functie **copy_table** die voor de data en coderings tabellen de **data_table_fun** en voor de *DataProperties* tabel de **prop_table_fun** gebruikt.

```
data_table_fun <- function(doc) {
  t1n <- xpathApply(doc,
    '//ns:entry[1]//m:properties[1]/d:*',
    xmlName,
    namespaces = std_namespaces)
  t1d  = xpathSApply(doc, '//m:properties/d:*',xmlValue)
  t1d  = as.data.frame(matrix(t1d, ncol = length(t1n), byrow = T),
    stringsAsFactors =F)
  names(t1d) = t1n
  return(t1d)
}

prop_table_fun <- function(doc) {
  m     = xpathSApply(doc, '//m:properties/d:*',
    function(x)
      c(
        xpathSApply(xmlParent(x), './d:ID', xmlValue, namespaces = std_namespaces),
        xmlName(x),
        xmlValue(x)
      ))
  # m matrix: r1 number; r2 field ; r3 value
  uf    = unique(m[2, ])
  # "ID" "Position" "ParentID" "Type" "Key" "Title" "Description" "ReleasePolicy"
  # "Datatype" "Unit" "Decimals" "Default"
  nc    = length(uf)
  nr    = 1+max(as.numeric(m[1, ]))
  m2 = matrix(rep('', nr * nc), nrow = nr, ncol = nc)
  for (i in 1:nr) {
    m3 = m[, m[1, ] == paste(i-1)] # counting origin=0
    ix = match(m3[2, ], uf)
    m2[i, ix] = m3[3, ]
  }
  colnames(m2) = uf
  rownames(m2) = 1:nr
  as.data.frame(m2,stringsAsFactors =F)
}

copy_table <- function (ti, make_table = NULL,  save_XML = NULL) {
  n1 = paste0('temp_', names(ti))
  if (is.null(save_XML)) {
    save_file_name = NULL
  } else if (nchar(save_XML) == 0) {
    save_file_name = paste0(n1, '.xml')
  } else {
    save_file_name = save_XML
  }
  t1    = get_cbs_data(ti, save_file_name = save_file_name)
  if (is.null(make_table))
    return(t1)
  t1d = make_table(t1)
}
```

```
props          = copy_table(x1['DataProperties'],prop_table_fun)
```

ID	Position	Type	Key	Title	ReleasePolicy	Datatype	Unit	Decimals	Default
0	0	Dimension	Geslacht	Geslacht					
1	1	Dimension	Leeftijd	Leeftijd					
2	2	Dimension	GeneesmiddelengroepATC	Geneesmiddelengroep (ATC)					
3	3	TimeDimension	Perioden	Perioden	true				
4	4	Topic	PersonenMetVerstrekteGeneesmiddelen_1	Personen met verstrekte geneesmiddelen		Long	aantal	0	Zero
5	5	Topic	PersonenMetGeneesmiddelenRelatief_2	Personen met geneesmiddelen, relatief		Double	%	2	Zero

Table 2: Informatie in http://opendata.cbs.nl/ODataFeed/OData/81071NED/DataProperties

Feitelijk inlezen van de data

De data (over het medicijn gebruik) bevindt zich in de *TypedDataSet* tabel die we met behulp van de genoemde functie als volgt kunnen inlezen.

```
TypedDataSet   = copy_table(x1['TypedDataSet'],data_table_fun)
sapply(TypedDataSet,class)
```

```
##                                     ID                                Geslacht
##                              "character"                              "character"
##                                Leeftijd                GeneesmiddelengroepATC
##                              "character"                              "character"
##                                Perioden PersonenMetVerstrekteGeneesmiddelen_1
##                              "character"                              "character"
##   PersonenMetGeneesmiddelenRelatief_2
##                              "character"
```

Geslacht	Leeftijd	Perioden	GeneesmiddelengroepATC	PersonenMetVerstrekteGeneesmiddelen_1	PersonenMetGeneesmiddelenRelatief_2
1100	10000	2006JJ00	100000	11241725	67.54
1100	10000	2007JJ00	100000	11320680	67.86
1100	10000	2008JJ00	100000	11704500	69.83
1100	10000	2009JJ00	100000	11803505	70.09
1100	10000	2010JJ00	100000	11859005	70.05

Table 3: Informatie in http://opendata.cbs.nl/ODataFeed/OData/81071NED/TypedDataSet

De eerste 5 regels van deze tabel vind je in Table 3 on page 4. Je ziet dat hierin alle kolommen die géén *Topic* zijn (volgens Table 2 on page 4) gecodeerd zijn. Verder zijn alle kolommen (ook de *Topic* velden) *character*.

Het koppelen van de coderings tabellen en maken van selecties

Omdat de *(Time)Dimension* kolommen gecodeerd zijn moeten we ook de tabellen voor deze kolommen ophalen. Eerst bepalen we (om in een later stadium dit proces zo veel mogelijk te automatiseren) welke de *Topic* en *(Time)Dimension* variabelen zijn. Dan halen we de tabellen op waarbij we alleen de *Key* en *Title* kolommen bewaren en de laatste de *(Time)Dimension* naam geven. Eventuele selecties kunnen hier al gedaan worden: voor GeneesmiddelengroepATC worden alleen de hoofdgroepen (naam begint met hoofdletter en spatie) en het totaal meegenomen.

```
topic_vars     = props %>%
    filter(Type=='Topic') %>%
    select(Key)
dim_vars       = props %>%
  filter(Type %in% c('Dimension','TimeDimension')) %>%
    select(Key)
Geslacht       = copy_table(x1['Geslacht'],data_table_fun) %>%
    select(Key,Title) %>% rename(Geslacht=Title)
Leeftijd       = copy_table(x1['Leeftijd'],data_table_fun) %>%
    select(Key,Title) %>% rename(Leeftijd=Title)
Perioden       = copy_table(x1['Perioden'],data_table_fun) %>%
```

```
    select(Key,Title) %>% rename(Perioden=Title)
GeneesmiddelengroepATC =
  copy_table(x1['GeneesmiddelengroepATC'],data_table_fun) %>%
    select(Key,Title) %>% rename(GeneesmiddelengroepATC=Title) %>%
    filter(grepl('^[[:upper:]]{1} |^Totaal', GeneesmiddelengroepATC))
```

Het feitelijke koppelen van de coderings tabellen aan *TypedDataSet* gebeurt hieronder, nadat de *Topic* kolommen numeriek zijn gemaakt. Voor elk van de dimensie namen wordt de dimensie tabel opgepakt (in de code in **tab1**) en die wordt met een inner join gekoppeld aan de hoofd tabel **tt**. Dan wordt de oorspronkelijke dimensie naam verwijderd (deze wees naar de gecodeerde informatie) en opnieuw gebruikt voor de gedecodeerde informatie.

```
tt = TypedDataSet %>% mutate_each_(funs(as.numeric),topic_vars$Key)
for (dim in dim_vars$Key) {
  tab1 = eval(parse(text=dim))
  by1  = c('Key') ; names(by1) = dim
  tt = tt %>%
    inner_join(tab1, by=by1) %>%
    select_(.dots = setdiff(names(.),dim)) %>%
    rename_(.dots = setNames(paste0(dim,'.y'), dim))
}
```

De eerste 5 regels van de gedecodeerde tabel vind je in Table 4 on page 5.

Geslacht	Leeftijd	Perioden	GeneesmiddelengroepATC	PersonenMetVerstrekteGeneesmiddelen_1	PersonenMetGeneesmiddelenRelatief_2
Totaal mannen en vrouwen	Totaal leeftijd	2006	Totaal	11241725	67.54
Totaal mannen en vrouwen	Totaal leeftijd	2007	Totaal	11320680	67.86
Totaal mannen en vrouwen	Totaal leeftijd	2008	Totaal	11704500	69.83
Totaal mannen en vrouwen	Totaal leeftijd	2009	Totaal	11803505	70.09
Totaal mannen en vrouwen	Totaal leeftijd	2010	Totaal	11859005	70.05

Table 4: Informatie in het gedecodeerde TypedDataSet data.frame

Session info

```
sessionInfo()
```

```
## R version 3.2.4 (2016-03-10)
## Platform: x86_64-w64-mingw32/x64 (64-bit)
## Running under: Windows 10 x64 (build 10586)
##
## 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] xtable_1.8-2 dplyr_0.4.3  XML_3.98-1.4 magrittr_1.5 curl_0.9.6   knitr_1.12.3
##
## loaded via a namespace (and not attached):
##  [1] Rcpp_0.12.3     digest_0.6.9    assertthat_0.1  R6_2.1.2        DBI_0.3.1
##  [6] formatR_1.3     evaluate_0.8.3  stringi_1.0-1   lazyeval_0.1.10 rmarkdown_0.9.5
## [11] tools_3.2.4     stringr_1.0.0   parallel_3.2.4  yaml_2.1.13     htmltools_0.3
```