

# Benaderen Open Data van CBS vanuit R

Han Oostdijk ([www.hanoostdijk.nl](http://www.hanoostdijk.nl))

30 maart 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)?
- hoe doe ik een query?

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 lokaal 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(myroot, 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	<a href="http://opendata.cbs.nl/ODataFeed/OData/81071NED/TableInfos">http://opendata.cbs.nl/ODataFeed/OData/81071NED/TableInfos</a>
UntypedDataSet	<a href="http://opendata.cbs.nl/ODataFeed/OData/81071NED/UntypedDataSet">http://opendata.cbs.nl/ODataFeed/OData/81071NED/UntypedDataSet</a>
TypedDataSet	<a href="http://opendata.cbs.nl/ODataFeed/OData/81071NED/TypedDataSet">http://opendata.cbs.nl/ODataFeed/OData/81071NED/TypedDataSet</a>
DataProperties	<a href="http://opendata.cbs.nl/ODataFeed/OData/81071NED/DataProperties">http://opendata.cbs.nl/ODataFeed/OData/81071NED/DataProperties</a>
Geslacht	<a href="http://opendata.cbs.nl/ODataFeed/OData/81071NED/Geslacht">http://opendata.cbs.nl/ODataFeed/OData/81071NED/Geslacht</a>
Leeftijd	<a href="http://opendata.cbs.nl/ODataFeed/OData/81071NED/Leeftijd">http://opendata.cbs.nl/ODataFeed/OData/81071NED/Leeftijd</a>
GeneesmiddelengroepATC	<a href="http://opendata.cbs.nl/ODataFeed/OData/81071NED/GeneesmiddelengroepATC">http://opendata.cbs.nl/ODataFeed/OData/81071NED/GeneesmiddelengroepATC</a>
Perioden	<a href="http://opendata.cbs.nl/ODataFeed/OData/81071NED/Perioden">http://opendata.cbs.nl/ODataFeed/OData/81071NED/Perioden</a>

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[ i, 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)
  assign(n1, t1d, envir = parent.env(environment()))
}
```

```
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	PersonenMetVerstekteGeneesmiddelen_1	Personen met verstekte geneesmiddelen		Long	aantal	0	Zero
5	5	Topic	PersonenMetGeneesmiddelenRelatief_2	Personen met geneesmiddelen, relatief		Double	%	2	Zero

(a)

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 PersonenMetVerstekteGeneesmiddelen_1
##                                "character"                        "character"
## PersonenMetGeneesmiddelenRelatief_2
##                                "character"
```

ID	Geslacht	Leeftijd	GeneesmiddelengroepATC	Perioden	PersonenMetVerstekteGeneesmiddelen_1	PersonenMetGeneesmiddelenRelatief_2
0	1100	10000	100000	2006JJ00	11241725	67.54
1	1100	10000	100000	2007JJ00	11320680	67.86
2	1100	10000	100000	2008JJ00	11704500	69.83
3	1100	10000	100000	2009JJ00	11803505	70.09
4	1100	10000	100000	2010JJ00	11859005	70.05

(a)

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 3 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	PersonenMetVerstrekkeGeneesmiddelen_1	PersonenMetGeneesmiddelenRelatief_2
Totaal mannen en vrouwen	Totaal leeftijd	2006	11241725	67.54
Totaal mannen en vrouwen	Totaal leeftijd	2007	11320680	67.86
Totaal mannen en vrouwen	Totaal leeftijd	2008	11704500	69.83
Totaal mannen en vrouwen	Totaal leeftijd	2009	11803505	70.09
Totaal mannen en vrouwen	Totaal leeftijd	2010	11859005	70.05

(a)

Table 4: Informatie in het gedecodeerde TypedDataSet data.frame