Using R package odataR for Statistics Netherlands data

Han Oostdijk

2016-05-16

Introduction

The package was made because I was interested in the information of Statistics Netherlands.
The easiest tool method to do an adhoc query is using the Statline interface in the Dutch or English. It offers the possibility to pivot the various dimensions of the information and download the result to a csv (comma separated values) file or an Excel spreadsheet. And then the csv or Excel file can be read into the R environment for further processing.
However especially the handling of the headers is laborious in this way because it is different for each table/layout combination.

Getting the information in R is made easier by the **odataR** package. Using this package avoids the intermediary csv or Excel file. The code for getting the information of a table in a *data.frame* can be as easy as `df=oadR_get_table(table_id='03759ned')`
when the identification code of the table is *03759ned* .

When you would execute this code it would result in a data.frame *df* with 3371018 rows and 22 columns. That is probably more data than information.
In this vignette I will show how to:

- indicate the location (*root*) of the OData structures. The default is set for the data of Statistics Netherlands so you don't have to do anything when you want to use that information. For other sources however you have to set the root.
- use the catalog to find out the available tables and their characteristics.
- find out the topics and dimensions of a table. There is a sub table for each dimension.
- determine with these sub tables how to do queries on the dimensions to request more or less exactly the information that is needed.

For a full example for retrieving and using the data of Statistics Netherland we use the following case study.

Case study

For this case study I want to see if the number of persons in my municipality (Amstelveen) in various age groups remains relatively constant. So I will retrieve information about this and create a plot of the relative group size over various years. For retrieval of the data we use of course the **odataR** package but to report, manipulate and plot the data we use the following packages (**knitr** is used to display the tables in this vignette):

```
library(knitr)
library(ggplot2)
library(magrittr)
library(dplyr)
```

Indicate where the OData information of Statistics Netherlands is located

The user of the package has to indicate the location (*root*) of the OData structures. Because I wrote the package with a special interest for the information of Statistics Netherlands it is not necessary to specify the *root* for the standard data of Statistics Netherlands: when no *root* is specified the software will always use the Statistics Netherlands data. But you can always check which *root* will be used with the `odataR_get_root` function:

```
library(odataR)
print(odataR_get_root())
```

```
## [1] "http://opendata.cbs.nl"
```

Indicate where OData information other than that of Statistics Netherlands is located

For other OData information the user of the package has to indicate the location (*root*) with the `odataR_set_root` function:

```
odataR_set_root("http://dataderden.cbs.nl")
print(odataR_get_root())
```

```
## [1] "http://dataderden.cbs.nl"
```

From now on the indicated url will be used as *root* until the moment that it is changed again with the `odataR_set_root` function. This function has as default argument the url of the OData structure ofStatistics Netherlands:

```
odataR_set_root()
print(odataR_get_root())
```

```
## [1] "http://opendata.cbs.nl"
```

Find in the catalog which tables are available

I know that the information I need is in the table with identification code *03759ned* that was used in the example above.
But assuming that we don't know this, we have to use the catalog to find out (for the case study at least) which tables contain information about persons, in regions, with certain ages. The easiest way to find out this information is using the English or Dutch visual web interface. When you have navigated through the catalog and reached the table that contains the necessary information you can read in the catalog all meta data about the table. Because you want to use the **odataR** package it is important to take note of the identifier that you will later use as *table_id*.
Apart from viewing the catalog we can also do a query on the catalog. In the next code section we show how to retrieve the whole catalog in data.frame `cbscat` and to check which describing elements (fields) it contains by looking at the first record in the catalog:

```
cbscat = odataR_list_tables()
(dimcat = dim(cbscat))
```

```
## [1] 3911   22
```

```
str(cbscat[1,])
```

```
## 'data.frame':    1 obs. of  22 variables:
##  $ ID                 : chr "0"
##  $ Identifier         : chr "82010NED"
##  $ Title              : chr "Zeggenschap bedrijven in Nederland; banen en lonen, bedrijfsgrootte"
##  $ ShortTitle         : chr "Zeggenschap bedrijven; banen, grootte"
##  $ ShortDescription   : chr "\nDeze tabel bevat informatie over banen en lonen bij bedrijven in Nede
##  $ Summary            : chr "Banen en lonen van werknemers bij bedrijven in Nederland\nnaar land van
##  $ Modified           : chr "2014-02-04T02:00:00"
##  $ MetaDataModified   : chr "2014-02-04T02:00:00"
##  $ ReasonDelivery     : chr "Actualisering"
##  $ ExplanatoryText    : chr ""
##  $ OutputStatus       : chr "Regulier"
##  $ Source             : chr ""
##  $ Language           : chr "nl"
##  $ Catalog            : chr "CBS"
##  $ Frequency          : chr "Eenmaalperjaar"
##  $ Period             : chr "2008 t/m 2011"
##  $ DefaultPresentation: chr "_la=nl&_si=&_gu=&_ed=LandVanUiteindelijkeZeggenschapUCI&_td=Perioden&gra
##  $ DefaultSelection   : chr "$filter=((LandVanUiteindelijkeZeggenschapUCI eq '11111') or (LandVanUite
##  $ GraphTypes         : chr "Table,Bar,Line"
##  $ RecordCount        : chr "32"
##  $ ColumnCount        : chr "18"
##  $ SearchPriority     : chr "2"
```

From this we see that the catalog has 3911 rows and 22 columns and that the field *ShortDescription* is probably suited to do a query on bevolking (population), leeftijd (age) and regio (region):

```
x = odataR_list_tables(query = paste0("?$filter=substringof('leeftijd',ShortDescription) ",
  "and substringof('regio',ShortDescription) ",
  "and substringof('bevolking',ShortDescription) ",
  "and substringof('Bevolking',ShortTitle)",
  "&$select=Identifier,Title,ShortTitle,RecordCount,ColumnCount"))
kable(x,caption='results query leeftijd,bevolking,regio')
```

Table 1: results query leeftijd,bevolking,regio

Identifier	Title	ShortTitle
82220NED	Regionale prognose 2014-2040; kerncijfers, regio-indeling 2013	Bevolking; kerncijfers, regio, 2014-2040
82172NED	Regionale prognose 2014-2040; bevolking, regio-indeling 2013	Bevolking; leeftijd, regio, 2014-2040
81273ned	Regionale prognose bevolkingsopbouw; 2011-2040	Bevolkingsopbouw gemeenten 2011-2040
70648ned	Bevolking op 1 januari; leeftijd, geboorteland en regio	Bevolking; geboorteland en regio
37713	Bevolking; leeftijd, herkomstgroepering, geslacht en regio, 1 januari	Bevolking; herkomstgroepering en regio
03759ned	Bevolking; geslacht, leeftijd, burgerlijke staat en regio	Bevolking; leeftijd, regio
80283ned	Regionale prognose bevolkingsopbouw;2009-2040	Bevolkingsopbouw gemeenten 2009-2040
71548ned	Regionale prognose bevolkingsopbouw; 2007-2025	Bevolkingsopbouw gemeenten 2007-2025
71188ned	Regionale prognose bevolkingsopbouw; 2005-2025	Bevolkingsopbouw gemeenten 2005-2025

The results of the query, to which also the extra condition on the field *ShortTitle* was added to limit the number of results, can be found in *Table1*. Note that I did the selection query directly on the catalog and not on `cbscat`. The mean reason for that is to show how this can be done but in general selections done on the webserver are more efficient because the amount of data to transport from webserver to client is decreased in

this way. Apart from the filter specification I also included a select statement in order to decrease the width of the table in this document. Normally such an intermediate table would be excluded from a document and in that case the select specification can be omitted. The *kable* statement serves only to get the result table included in the document. To avoid confusion I will hide further calls to *kable* in this document.
Of course the table with identification code *03759ned* is in the table.

Find information about table *03759ned*

In the CBS database a table name (table_id) points to a set of sub tables that together provide the information. One (or rather two but I will use only one) 'main' sub table contains the topic data with dimensions in coded form indicating where the topic data relates to. The other sub tables convey the meaning of the coded dimensions. E.g. a topic field could be the number of married male persons and the dimensions could be region and period.

We see from *Table1* that table *03759ned* has 3371018 rows and 22 columns. So it is worthwhile to create a query that exactly selects the information that is needed.

DataProperties sub table

First we use function `odataR_get_subtables` to obtain the url of each of the sub tables and then use the url of sub table *DataProperties* to see which are the topics and dimensions of the main table.

```
subtabs = odataR_get_subtables(table_id='03759ned')
props = odataR_get_subtable(subtabs['DataProperties'],mt='prop')
x= props %>% select(Position,ParentID,Type,Key,Title)
```

Table 2: properties of table 03759ned (first fields only)

Position	ParentID	Type	Key	Title
0		Dimension	Leeftijd	Leeftijd
1		GeoDimension	RegioS	Regio's
2		TimeDimension	Perioden	Perioden
		TopicGroup		Bevolking naar geslacht
3	3	Topic	MannenEnVrouwen_1	Mannen en vrouwen
4	3	Topic	Mannen_2	Mannen
5	3	Topic	Vrouwen_3	Vrouwen
		TopicGroup		Bevolking: ongehuwd naar geslacht
6	7	Topic	MannenEnVrouwen_4	Mannen en vrouwen
7	7	Topic	Mannen_5	Mannen
8	7	Topic	Vrouwen_6	Vrouwen
		TopicGroup		Bevolking: gehuwd naar geslacht
9	11	Topic	MannenEnVrouwen_7	Mannen en vrouwen
10	11	Topic	Mannen_8	Mannen
11	11	Topic	Vrouwen_9	Vrouwen
		TopicGroup		Bevolking: verweduwd naar geslacht
12	15	Topic	MannenEnVrouwen_10	Mannen en vrouwen
13	15	Topic	Mannen_11	Mannen
14	15	Topic	Vrouwen_12	Vrouwen
		TopicGroup		Bevolking: gescheiden naar geslacht
15	19	Topic	MannenEnVrouwen_13	Mannen en vrouwen
16	19	Topic	Mannen_14	Mannen
17	19	Topic	Vrouwen_15	Vrouwen

Position	ParentID	Type	Key	Title
		TopicGroup		Regionale coderingen
18	23	Topic	Gemeente_16	Gemeente
19	23	Topic	Landsdeel_17	Landsdeel
20	23	Topic	Provincie_18	Provincie
21	23	Topic	COROPGebied_19	COROP-gebied

In *Table2* we see that there are three dimensions (in position 0, 1 and 2) and 18 topics (for the other positions) for which we will (in the case study) consider only the first topic:

- Key *Leeftijd*: dimension for ages and agegroups
- Key *RegioS*: dimension for regions in the Netherlands
- Key *Perioden*: dimension for periods
- Key *MannenEnVrouwen_1* : the topic of the total number of persons (without distinction by gender) per combination of the dimensions. Other topics do makes this distinction and/or only consider (un)married or divorced or widowed persons.

Because the case study wants to use the number of persons in certain age groups in Amstelveen I first have to check the sub tables for the dimensions to see how I can recognize these persons. Remember that I can easily code to retrieve the whole table with `df=oadR_get_table(table_id='03759ned')` but this would result in a *data.frame* of 3371018 rows and 22 columns.

Check the *Leeftijd* dimension (ages and agegroups)

By executing `x = odataR_get_subtable(subtabs['Leeftijd'])` we see that we get the ten-year age groups when we select the keys starting with a '3'. *Table3* shows these keys. Note that key 399 is the total for all the age groups

Table 3: keys for ten-year age groups

Key	Title	Description
399	Totaal leeftijden	
301	Jonger dan 10 jaar	
302	10 tot 20 jaar	
303	20 tot 30 jaar	
304	30 tot 40 jaar	
305	40 tot 50 jaar	
306	50 tot 60 jaar	
307	60 tot 70 jaar	
308	70 tot 80 jaar	
309	80 tot 90 jaar	
310	90 jaar of ouder	

Check the *RegioS* dimension (regions)

By executing `x = odataR_get_subtable(subtabs[Regios'])` we see that we get the municipality Amstelveen when we select key 'GM0362'. *Table4* shows this row

Table 4: key(s) for region Amstelveen

Key	Title
GM0362	Amstelveen

Check the *Perioden* dimension (periods)

By executing `x = odataR_get_subtable(subtabs[Perioden'])` we see in the results (*Table5*) that we get all available years without any grouping. So no filtering has to take place

Table 5: keys for periods

Key	Title	Description	Status
1988JJ00	1988		Definitief
1989JJ00	1989		Definitief
1990JJ00	1990		Definitief
1991JJ00	1991		Definitief
1992JJ00	1992		Definitief
1993JJ00	1993		Definitief
1994JJ00	1994		Definitief
1995JJ00	1995		Definitief
1996JJ00	1996		Definitief
1997JJ00	1997		Definitief
1998JJ00	1998		Definitief
1999JJ00	1999		Definitief
2000JJ00	2000		Definitief
2001JJ00	2001		Definitief
2002JJ00	2002		Definitief
2003JJ00	2003		Definitief
2004JJ00	2004		Definitief
2005JJ00	2005		Definitief
2006JJ00	2006		Definitief
2007JJ00	2007		Definitief
2008JJ00	2008		Definitief
2009JJ00	2009		Definitief
2010JJ00	2010		Definitief
2011JJ00	2011		Definitief
2012JJ00	2012		Definitief
2013JJ00	2013		Definitief
2014JJ00	2014		Definitief
2015JJ00	2015		Definitief
2016JJ00	2016		Definitief

Compose the query to get the required data

So we now have the information to do a precise query for the information we need:

```
Aveen= odataR_get_table(table_id='03759ned',
     query=paste0("?$filter=startswith(RegioS,'GM0362') ",
       "and (startswith(Leeftijd,'30') or startswith(Leeftijd,'31'))&",
       "$select=Perioden,Leeftijd,MannenEnVrouwen_1"))
```

Data.frame *Aveen* has 290 rows and 3 columns. Remember the dimensions of the full table: 3371018 rows and 22 columns.

The remainder of the vignette shows how this information can be plotted.

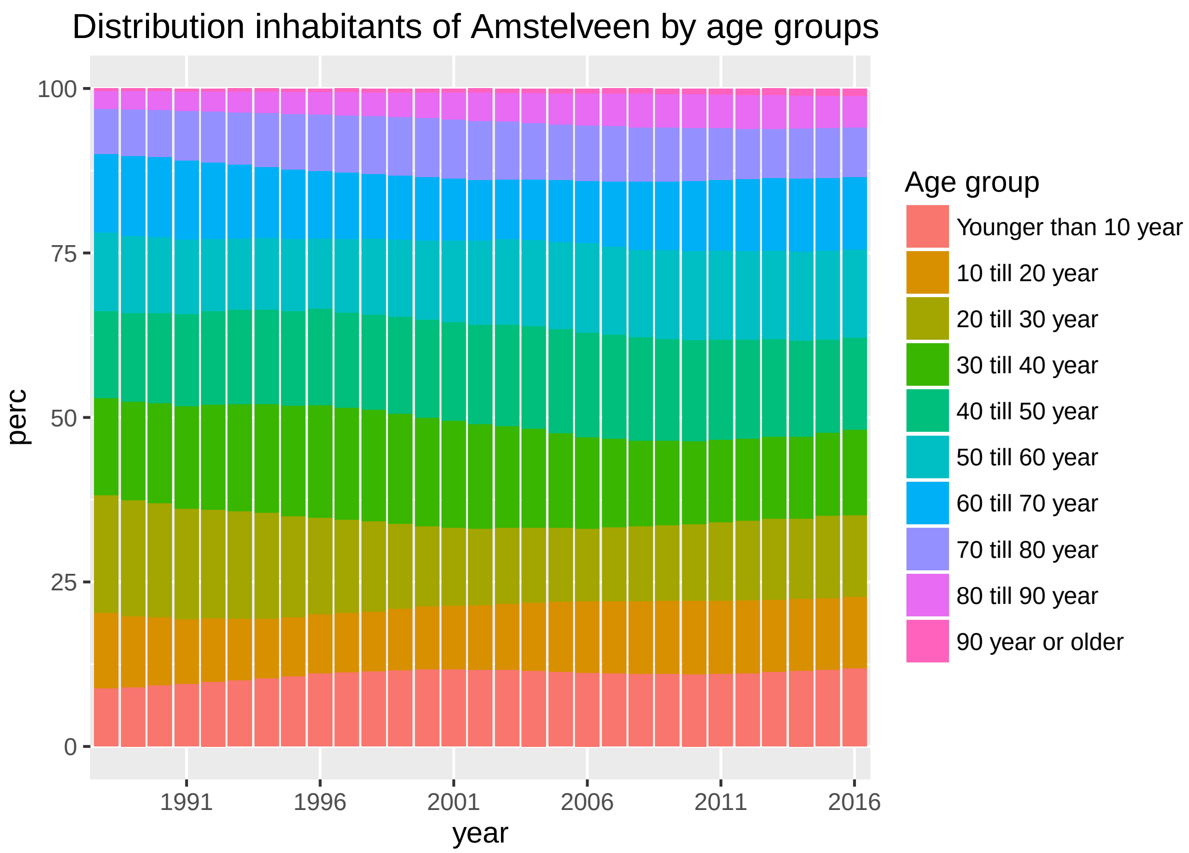
Determine the distribution per period over age groups

Per year we want to show the relative size of each age group. First we will translate the Dutch terms to English. In the second step we calculate the total number of persons in a year and in the third step we merge this total to the first table so that we can calculate percentages. In the last step we also ensure the correct sorting order by using the `factor` function.

```
av1 = Aveen %>%
  select(period=Perioden,ageg=Leeftijd,persons=MannenEnVrouwen_1) %>%
  mutate(ageg = gsub('jaar','year',ageg,fixed=T)) %>%
  mutate(ageg = gsub('Jonger dan','Younger than',ageg,fixed=T)) %>%
  mutate(ageg = gsub('of ouder','or older',ageg,fixed=T)) %>%
  mutate(ageg = gsub('tot','till',ageg,fixed=T))
av2 = av1 %>%
  group_by(period) %>%
  summarise(tot = sum(persons))
grps =  c("Younger than 10 year", "10 till 20 year", "20 till 30 year",
         "30 till 40 year", "40 till 50 year", "50 till 60 year",
         "60 till 70 year", "70 till 80 year", "80 till 90 year",
         "90 year or older")
av3 = av1 %>%
  inner_join(av2,by=c('period'='period')) %>%
  mutate(perc = 100*persons/tot,ageg = factor(ageg,levels=grps))
```

Create barchart

```
ggplot(data = av3, aes(x = period, y = perc, fill = ageg)) +
  geom_bar(stat = "identity") +
  scale_fill_discrete(guide = guide_legend(title = 'Age group')) +
  scale_x_discrete("year",breaks=seq(1986,2016,5)) +
  ggtitle('Distribution inhabitants of Amstelveen by age groups ')
```


Distribution inhabitants of Amstelveen by age groups
perc
100
75
50
25
0
1991
1996
2001
2006
2011
2016
year
Age group
Younger than 10 year
10 till 20 year
20 till 30 year
30 till 40 year
40 till 50 year
50 till 60 year
60 till 70 year
70 till 80 year
80 till 90 year
90 year or older