Reading OData with MATLAB

Han Oostdijk (han@hanoostdijk.nl)

May 20, 2016

Contents

Reading OData with MATLAB

In the vignette of the R package odataR (https://github.com/HanOostdijk/odataR.git) we show how the package can be used to read data of Statistics Netherlands (CBS).

In this document we show how to do this with MATLAB.

Query syntax for R package odataR

```
  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"))
```

Query syntax for MATLAB webread

```
url = 'http://opendata.cbs.nl/ODataFeed/OData/03759ned/TypedDataSet' ;
query=['?$filter=startswith(RegioS,''GM0362'') ', ...
'and (startswith(Leeftijd,''30'') or startswith(Leeftijd,''31''))&', ...
'$select=Perioden,Leeftijd,MannenEnVrouwen_1'] ;
```

Return the query information in a struct

Without the options parameter in webread the data is returned in a struct. The value field of the struct contains the data. Here we transform the data to a table. Be aware that the **odataR** function odataR_get_table has two additional features. See next section.

```
struct_data = webread([url,query]) ;
table_data      = struct2table(struct_data.value)   ;
```

show the first three rows

```
disp(table_data(1:3,:))
```

```
    Perioden      Leeftijd    MannenEnVrouwen_1
   ----------    --------    -----------------
   '1988JJ00'    '301'       6174
   '1989JJ00'    '301'       6236
   '1990JJ00'    '301'       6526
```

Additional features of `odataR_get_table`

The simple call to `webread` does not offer the same functionality as `odataR_get_table`. The next sections show how to fix the two 'deficiencies":

- the values are not decoded (see column 1 and 2 in the table_data output) and not made numeric (see column 3) where possible. See the section **Decode the table** how to resolve this.
- only 10.000 rows are returned. See section **Reading large table with $skip construct** for how this can be circumvented.

Decode the table

The `odataR_get_table` function of the **odataR** package automatically decodes the dimensions. Using the MATLAB function we have to do this manually by retrieving the translation table for the dimensions. See the package documentation for how to find out the specifications of the table and its sub tables. From the description in the package we know that the two dimension to decode are `Leeftijd` (age) and `Perioden` (periods) and we also know the urls of the sub tables. First we load these tables in cell array format

```
url = 'http://opendata.cbs.nl/ODataFeed/OData/03759ned/Leeftijd' ;
query='?$select=Key,Title' ;
struct_data = webread([url,query]) ;
Leeftijd    = struct2cell(struct_data.value)' ;
disp(size(Leeftijd))
```

```
   133     2
```

```
url = 'http://opendata.cbs.nl/ODataFeed/OData/03759ned/Perioden' ;
query='?$select=Key,Title' ;
struct_data = webread([url,query]) ;
Perioden    = struct2cell(struct_data.value)' ;
disp(size(Perioden))
```

```
    29     2
```

Now we use the translation tables to decode the table we downloaded in one of the first sections.

```
[~,i_Leeftijd] = ismember(table_data.Leeftijd,Leeftijd(:,1)) ;
[~,i_Perioden] = ismember(table_data.Perioden,Perioden(:,1)) ;
table_data.Leeftijd = Leeftijd(i_Leeftijd,2) ;
table_data.Perioden = Perioden(i_Perioden,2) ;
```

show first three rows

```
disp(table_data(1:3,:))
```

```
    Perioden            Leeftijd            MannenEnVrouwen_1

    ---------     ---------------------     -----------------
    '1988'        'Jonger dan 10 jaar'      6174
    '1989'        'Jonger dan 10 jaar'      6236
    '1990'        'Jonger dan 10 jaar'      6526
```

Reading large table with $skip construct

Until now the result of the query consisted of 290 rows. Without the filter construct the data contains more than 3 million rows. The OData interface allows no more than 10.000 rows to return in one request. Therefore the query below only returns the first 10.000 rows in the struct. NB To avoid a timeout for the connection after 5 seconds the `Timeout` value is set to 25 (seconds).

```
url = 'http://opendata.cbs.nl/ODataFeed/OData/03759ned/TypedDataSet' ;
query='?$select=Perioden,Leeftijd,MannenEnVrouwen_1' ;
options      = weboptions('Timeout',25);
struct_data = webread([url,query],options) ;
cell_data1    = struct2cell(struct_data.value)' ;
disp(size(cell_data1))
```

```
       10000           3
```

show the first three rows of the first block

```
disp(cell_data1(1:3,:))
```

```
    '1988JJ00'    ' 99'    [14714948]
    '1989JJ00'    ' 99'    [14805240]
    '1990JJ00'    ' 99'    [14892574]
```

Indeed we see that no more than 10.000 rows are retrieved. The next 10.000 rows (and the next ...) can be retrieved by using the **$skip** construct:

```
query='?$select=Perioden,Leeftijd,MannenEnVrouwen_1&$skip=10000' ;
options      = weboptions('Timeout',25);
struct_data = webread([url,query],options) ;
cell_data2    = struct2cell(struct_data.value)' ;
disp(size(cell_data2))
```

```
       10000           3
```

show the first three rows of the second block

```
disp(cell_data2(1:3,:))
```

```
    '2012JJ00'    ' 99'    []
    '2013JJ00'    ' 99'    []
    '2014JJ00'    ' 99'    []
```

When you are only interested in getting a part of a table (e.g. when checking results or taken a sample) you can use the **$top** construct. Here we show the first three rows of the third block with **$top** instead of loading 10.000 rows and displaying only three.

```
query='?$select=Perioden,Leeftijd,MannenEnVrouwen_1&$skip=20000&$top=3'
    ;
options      = weboptions('Timeout',25);
struct_data = webread([url,query],options) ;
cell_data3    = struct2cell(struct_data.value)' ;
disp(size(cell_data3))
```

```
     3     3
```

show all rows of the third block

```
disp(cell_data3)
```

```
    '2007JJ00'    ' 99'    [54956]
    '2008JJ00'    ' 99'    [54962]
    '2009JJ00'    ' 99'    [55201]
```

Return data as an XML string

webread without an options parameter retrieves the data in a struct. This is probably the most convenient method when only the information in the table is needed. In special (more general) cases it could be advantageous to retrieve all data in XML format. This can be done by setting ContentType to text:

```
query=['?$filter=startswith(RegioS,''GM0362'') ', ...
'and (startswith(Leeftijd,''30'') or startswith(Leeftijd,''31''))&', ...
'$select=Perioden,Leeftijd,MannenEnVrouwen_1'] ;
options      = weboptions('ContentType','text');
xmlstring    = webread([url,query],options);
disp(class(xmlstring))
```

```
char
```

We can now view the data but XML data is not easily readable. We show the first 600 characters:

```
disp(reshape(xmlstring(1:600),60,10)')
```

```
<?xml version="1.0" encoding="utf-8"?>
<feed xml:base="http
://opendata.cbs.nl/ODataFeed/OData/03759ned" xmlns="http://w
ww.w3.org/2005/Atom" xmlns:d="http://schemas.microsoft.com/a
do/2007/08/dataservices" xmlns:m="http://schemas.microsoft.c
om/ado/2007/08/dataservices/metadata" xmlns:georss="http://w
ww.georss.org/georss" xmlns:gml="http://www.opengis.net/gml"
>
  <id>http://opendata.cbs.nl/ODataFeed/OData/03759ned/Typ
edDataSet</id>
  <title type="text">TypedDataSet</title>

 <updated>2016-05-12T02:00:00+02:00</updated>
  <link rel="
self" title="TypedDataSet" href="http://opendata.cbs.nl/ODat
```

In such cases it is better to copy the string to a file and view the file. The websave function can be used to directly copy the information to a file:

```
xmlfile      = 'table_03759ned.xml' ;
websave(xmlfile,[url,query],options);
```

Convert XML file to a text file or text string

After viewing the XML file we can transform it with the function `xslt` and a suitable XSL file. Here we want to extract only the table information and convert to a text file. We created the `table_03759ned.xsl` for this purpose :

```
<!-- table_035759ned.xsl -->
<xsl:stylesheet version="2.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
xmlns:d="http://schemas.microsoft.com/ado/2007/08/dataservices"
xmlns:m="http://schemas.microsoft.com/ado/2007/08/dataservices/metadata
   ">
<xsl:output method="text" omit-xml-declaration="yes" indent="no"/>

<xsl:template match="m:properties" >
 <xsl:value-of select="d:Perioden" />, <xsl:value-of select="d:Leeftijd
    " />,  <xsl:value-of select="d:MannenEnVrouwen_1" /> <xsl:text>&#xa
    ;</xsl:text>
 <!-- for readability these fields are repeated (as a comment) on
    separate lines:
 <xsl:value-of select="d:Perioden" />,
 <xsl:value-of select="d:Leeftijd" />,
 <xsl:value-of select="d:MannenEnVrouwen_1" />
 <xsl:text>&#xa;</xsl:text>
   -->
</xsl:template>

<xsl:template match="text()" >
</xsl:template>

<xsl:template match="/ | *" >
                <xsl:apply-templates/>
</xsl:template>

</xsl:stylesheet>
```

The `xslt` function can create a file or a string (depending on the third parameter) from the XML input data set:

```
xslt(xmlfile,'table_03759ned.xsl','table_03759ned.txt') ;
txtstring   = xslt(xmlfile,'table_03759ned.xsl','-tostring') ;
% display the first part of the transformed XML:
disp(txtstring(1:120))
```

```
1988JJ00, 301,  6174
1989JJ00, 301,  6236
1990JJ00, 301,  6526
1991JJ00, 301,  6726
1992JJ00, 301,  7060
1993JJ00, 301,
```

Return data as an XML object

By setting `ContentType` to `xmldom` a `DeferredDocumentImpl` object is returned. We can not view the XML in this way. It is however possible to 'translate' it to a text file (or text string) as was done in the previous section

```
options      = weboptions('ContentType','xmldom');
xml_data     = webread([url,query],options);
disp(class(xml_data))
```

```
org.apache.xerces.dom.DeferredDocumentImpl
```

```
xslt(xml_data,'table_03759ned.xsl','table_03759ned.txt') ;
txtstring    = xslt(xml_data,'table_03759ned.xsl','-tostring') ;
disp(txtstring(1:120))
```

```
1988JJ00, 301,  6174
1989JJ00, 301,  6236
1990JJ00, 301,  6526
1991JJ00, 301,  6726
1992JJ00, 301,  7060
1993JJ00, 301,
```

Return data as a json string

webread without an options parameter retrieves the data in a struct. By setting the format to json in the query string and setting the ContentType to text the data is returned as a character string with json contents.

```
query=['?$format=json&$filter=startswith(RegioS,''GM0362'') ', ...
'and (startswith(Leeftijd,''30'') or startswith(Leeftijd,''31''))&', ...
'$select=Perioden,Leeftijd,MannenEnVrouwen_1'] ;
options      = weboptions('ContentType','text');
jsonstring    = webread([url,query],options);
```

display the first 600 character of the json string

```
disp(jsonstring(1:600))
```

```
{
  "odata.metadata":"http://opendata.cbs.nl/ODataFeed/OData/03759ned/$metadata#Cbs.OData.WebAPI.Typed
    {
      "Perioden":"1988JJ00","Leeftijd":"301","MannenEnVrouwen_1":6174.0
    },{
      "Perioden":"1989JJ00","Leeftijd":"301","MannenEnVrouwen_1":6236.0
    },{
      "Perioden":"1990JJ00","Leeftijd":"301","MannenEnVrouwen_1":6526.0
    },{
      "Perioden":"1991JJ00","Leeftijd":"301","MannenEnVrouwen_1":6726.0
    },{
      "Perioden":"1992JJ00","Leeftijd":"301","MannenEnVrouwen_1":7060.0
    },{
      "Perioden"
```