H Henson (July 2023)

A Working Example CANSIM-SDMX

The document <*https://www.statcan.gc.ca/en/developers/sdmx/user-guide* recently released by Stats Can is highly significant in nature. However, it is intended for a technically inclined reader. This note provides a working example to help a reader less proficient get started. This note is based on Example 1 given on page 7 of the above cited document. This example provides a single line of http. This is a sufficient for a developer but some analysts would need additional support before they could make use of SDMX. This note is targetted on the needs of an analyst with an interest in Python.

Enter CURL COMMAND

Code can be written that is stored in a one-line text file. This script can then be executed in a fashion specific to the computation environment in use. This example was written and tested for a BASH window under Ubuntu LINUX. However, there are variants available under almost every possible computation environment. With Ubuntu LINUX the following command would be stored in a text file with and extension 'sh'. The file would be executed via the SOURCE command.

```
curl https://www150.statcan.gc.ca/t1/wds/sdmx/statcan/v1/rest/data/DF_17100005/1.1.1 > DF17100005.xml
```

Essentially the curl command is equivalent to entering the http web address in the browser window. Thus if curl is entered on a commandline the data will come back and fill the screen. That is why the URL is follow by what looks like a greater than sign but is known as a pipe. In this case the pipe command directs the output of the curl command to an xml file, which was arbitrarily named after the vector. Once this command has been executed an xml file should appear in the local drive.

Transform the XML to CVS

Most analysts would not be able to work directly with the XML that is generated. Although the xml file generated is valid from an XML perspective, it is not standard. As a result the following code in not what some might expect as the data note was located futher down the XML tree than what might be expected.

```python
import xml.etree.ElementTree as ET
import csv

tree = ET.parse("DF17100005.xml")
root = tree.getroot()

for elem in root:
 for subelem in elem:
  with open("DF17100005.csv",'w',newline='') as f_out:
   writer = csv.writer(f_out)
   writer.writerow(('Year','Population'))
   for ssubelem in subelem:
    #print(ssubelem.attrib)
    theyr = ssubelem.get('TIME_PERIOD')
    theval = ssubelem.get('OBS_VALUE')
    writer.writerow((theyr,theval))
```

This code can be used for another vector by following the following steps:

- Determine the vector number of the data of interest and replace DF1700005.xml with the label for the new vector

- 'Year' and 'Population' will have to be changed if column headings are desired. Note they are not required.
- uncomment the print statement if difficulties are encountered for diagnostic purposes

Future Work

It is not know to what extent this example is of practical value. It may be that the package available on R is more than sufficient. There are plans to expand this note along the following lines:

- test the curl command on non-Linux environments, such as Chrome and Windows.
- test with collections of vectors
- test with various periodicities

I will also be open to requests based on unique and interesting situations.