

Using ARQ to investigate RDF data cube

mja@statgroup.dk

2016-04-29

Contents

SPARQL scripts for the demographics cube (DC-DEMO-sample.ttl)	1
Get all member of qb:ComponentProperty	1
How to run this .Rmd file	4

SPARQL scripts for the demographics cube (DC-DEMO-sample.ttl)

The examples below uses `arq` from Apache Jena (<http://jena.apache.org>). To install `arq` - download and unpack the latest version of `apache-jena` from (<http://jena.apache.org/download/index.cgi>). Then you need some way of invoking `arq`; I use a not-so-clever-approach: `cd ~/bin; ln -s /opt/apache-jena-2.13.0/bin/arq`.

Given a SPARQL query and RDF data, `arq` returns the result of the query. So this is the command line way of making a SPARQL query.

The use of `arq` is described in many places, see for example (<http://www.learningsparql.com/>).

All `arq` commands below are to be run in the directory with the sample files, which is `inst/extdata/CUBE-standards-rdf` directory or `extdata/CUBE-standards-rdf` depending on the whether the development version or the installed version of the package is used.

The `cd` below in each code block is included because I could not find a quick way to get the code chunk executed in that directory. `knitr` is flexible enough to do it, I have not yet found the right way to do it. So, ignore the repeated `cd ..`

For making the SPARQL queries I used a simple trick - copy the turtle definition from the `cube.ttl`, and do a replace regexp in `emacs` using pattern `+\([^:]+\):\[^\]+\) .*$` and replacement `\1:\2 ?\2 ;`.

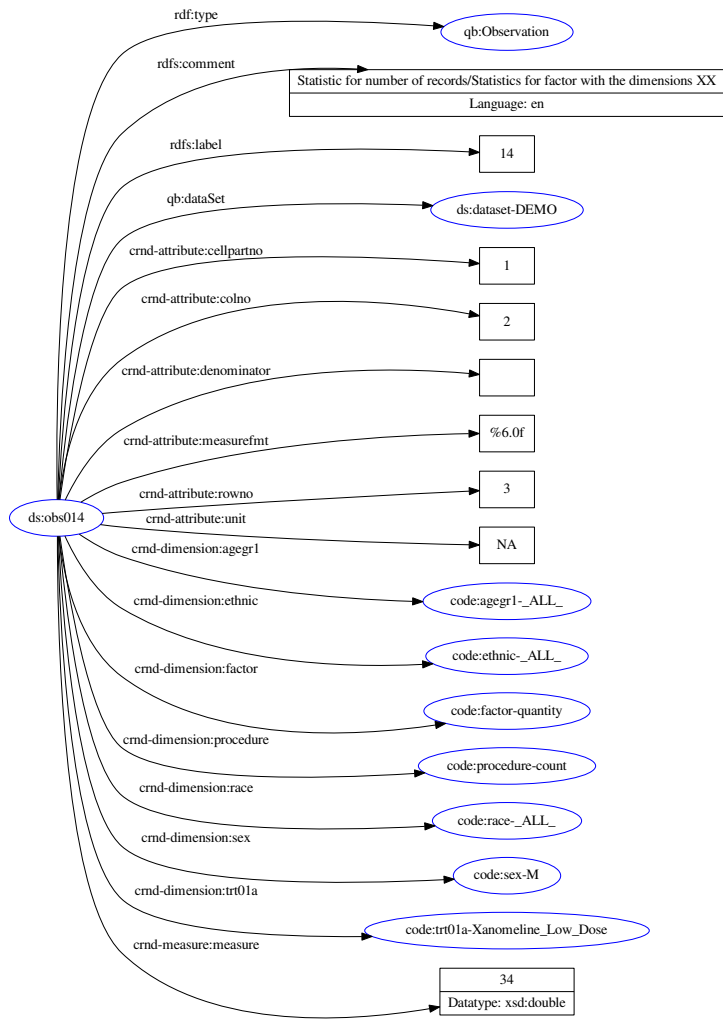
Get all member of qb:ComponentProperty

```
cd ../extdata/sample-rdf
arq --results ttl --data ../../../../rrdfqb/inst/extdata/cube-vocabulary-rdf/cube.ttl --query qb-constr
rapper -i turtle -o dot fordot.ttl > fordot.dot
dot -x -Tpng -ograph.png fordot.dot
rm -f fordot.dot
```

```
## rapper: Parsing URI file:///home/ma/projects/rrdfqbcrnd0/rrdfqbcrndex/inst/extdata/sample-rdf/fordot
## rapper: Serializing with serializer dot
## rapper: Parsing returned 29 triples
```

ToDo(MJA): location for `cube.ttl` should be generated by the program - not using a directory reference

```
knitr::include_graphics("../extdata/sample-rdf/graph.png")
```



Model:
(Unknown)

Namespaces:

dccc: <http://www.example.org/dc/demo/dccc/>
 code: <http://www.example.org/dc/code/>
 sdms-1-3: <http://rdf.cdsc.org/sdm-1-3/schema#>
 adam-2-1: <http://rdf.cdsc.org/std/adam-2-1#>
 owl: <http://www.w3.org/2002/07/owl#>
 xsd: <http://www.w3.org/2001/XMLSchema#>
 sdm-1-3: <http://rdf.cdsc.org/std/sdm-1-3#>
 cdash-1-1: <http://rdf.cdsc.org/std/cdash-1-1#>
 skos: <http://www.w3.org/2004/02/skos/core#>
 rdfs: <http://www.w3.org/2000/01/rdf-schema#>
 adamvr-1-2: <http://rdf.cdsc.org/std/adamvr-1-2#>
 crmd-attribute: <http://www.example.org/dc/attribute#>
 sdm-1-2: <http://rdf.cdsc.org/std/sdm-1-2#>
 ds: <http://www.example.org/dc/demo/ds/>
 sdmct: <http://rdf.cdsc.org/sdm-terminology#>
 qb: <http://purl.org/linked-data/cube#>
 mms: <http://rdf.cdsc.org/mms#>
 crmd-dimension: <http://www.example.org/dc/dimension#>
 dct: <http://purl.org/dc/terms/>
 cdscs: <http://rdf.cdsc.org/std/schema#>
 cdashct: <http://rdf.cdsc.org/cdash-terminology#>
 dcat: <http://www.w3.org/ns/dcat#>
 prov: <http://www.w3.org/ns/prov#>
 sdmig-3-1-3: <http://rdf.cdsc.org/std/sdmig-3-1-3#>
 crmd-measure: <http://www.example.org/dc/measure#>
 adamig-1-0: <http://rdf.cdsc.org/std/adamig-1-0#>
 cts: <http://rdf.cdsc.org/cts/schema#>
 pav: <http://purl.org/pav/>
 sdmig-3-1-2: <http://rdf.cdsc.org/std/sdmig-3-1-2#>
 sendig-3-0: <http://rdf.cdsc.org/std/sendig-3-0#>
 adamct: <http://rdf.cdsc.org/adam-terminology#>
 rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
 sendct: <http://rdf.cdsc.org/send-terminology#>
 rrdqbcrnd0: <http://www.example.org/rrdqbcrnd0/>
 dc: <http://purl.org/dc/elements/1.1/>

The file is needed for rendering, so no clean up.

```
cd ../extdata/sample-rdf  
rm -f graph.png
```

How to run this .Rmd file

.. add text ..