PhUSE 2016

Paper TT05

Generating Analysis Results and Metadata – report from a PhUSE CS project

Marc Andersen, StatGroup ApS, Copenhagen, Denmark
Marcelina Hungria, DIcore Group, LLC, NJ, USA
Suhas R. Sanjee, Merck & Co., Inc., Kenilworth, NJ USA

INTRODUCTION

The PhUSE CS Semantic Technology Working Group investigates how W3C semantic standards can support the clinical and non-clinical trial data life cycle. This presentation reports work from the project "Analysis Results and Metadata in RDF"[1], with the scope of development of standard models and technical standards for the storage and usage of analysis results data and metadata to support clinical and non-clinical applications. The overall idea is to store analysis results in the RDF Data Cube format[2].

The project team has developed a white paper [1], technical specification [2] and proof of concept R-package [3] providing an overall framework and tools to generate RDF Data Cubes.

In this paper, the authors present their experiences in using the overall framework and tools developed by the project team plus the SAS programs to:

- reproduce selected tables from the CSR
  - generate the results as RDF data cubes and
  - query the RDF data cubes to
  - present the results in the usual tabular format
- Generate interactive tables and figures, and
- Hyperlink results in the body of the report to the location in the table section.

This paper provides a description of the process and a summary of learnings. The framework and tools are freely available[3].

MATERIAL

For testing the approach, the updated Version of the CDISC Pilot Submission Package from 2013[4] specifically the clinical study report (CSR), ADaM datasets and DEFINE-xml were used. The PhUSE scripting group uses these datasets for development of standard scripts[5].

In addition, the PhUSE CS working group deliverables i.e. the white paper [1], technical specification [2] and proof of concept R-package [3] served as materials for this paper.

PROCESS

The process used is a three-step process consisting of generating analysis results, storing the results as RDF Data Cubes, and finally presenting the results obtained by querying the RDF data cube version of the analysis results (Figure 1). This follows the process outlined in the white paper [1], technical specification [2] and proof of concept R-package [3].

[1] www.phusewiki.org/wiki/index.php?title=Analysis_Results_Model
[2] https://www.w3.org/TR/vocab-data-cube/
[3] https://github.com/MarcJAndersen/poc-analysis-results-metadata
[4] http://www.cdisc.org/sdtmadam-pilot-project
[5] https://github.com/phuse-org/phuse-scripts/tree/master/data/adam/cdisc

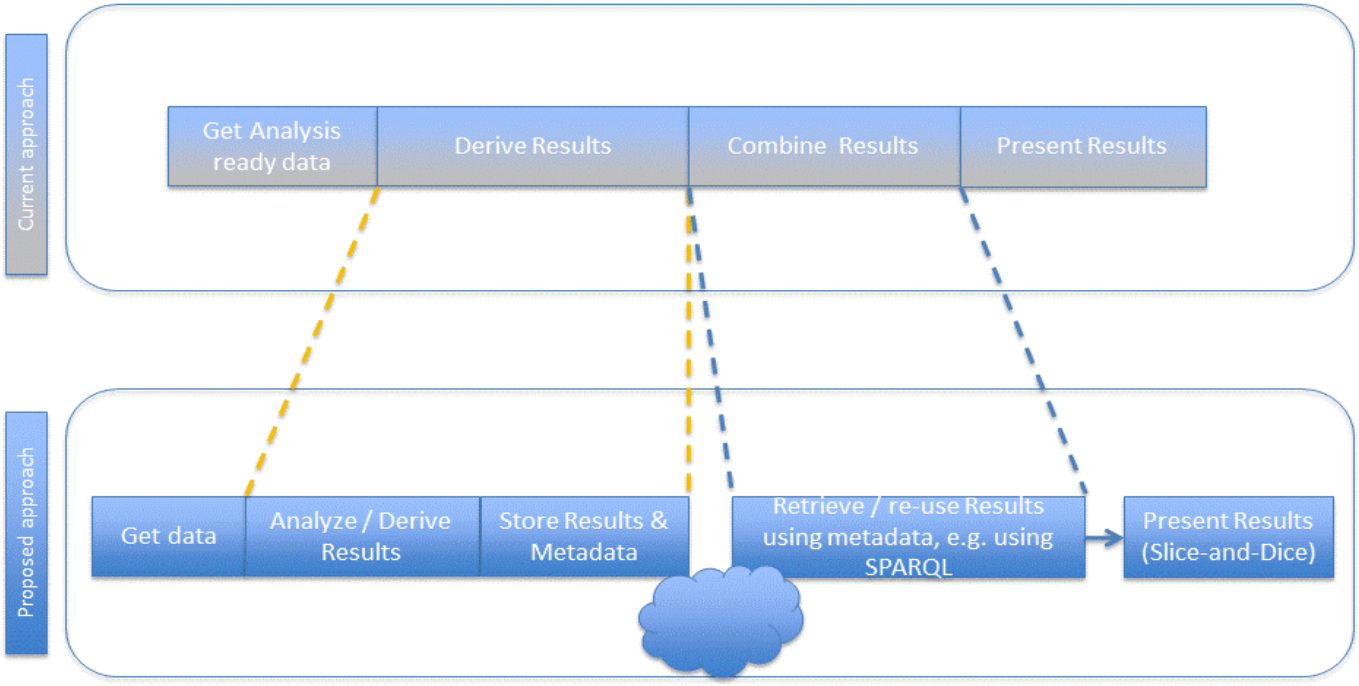


Figure 1: Process Flow of Proposed Approach

SCOPE

A subset of tables containing descriptive statistics, counts or percentages, identified from the CDISC pilot CSR and the associated Define-XML file were reproduced using the proposed process. Listing 1 shows the tables and datasets selected.

Listing 1:Tables reproduced from CDISC Pilot Project Data

Table	Title	ADaM dataset
14-1.01	Summary of Populations	ADSL
14-1.02	Summary of End of Study Data	ADSL
14-1.03	Summary of Number of Subjects by Site	ADSL
14-2.01	Summary of Disposition	ADSL
14-3.01	Primary Endpoint Analysis: ADAS Cog (11) - Change from Baseline to Week 24 - LOCF	ADQSADAS
14-5.01	Incidence of Treatment Emergent Adverse Events by Treatment Group	ADAE

POTENTIAL NEW FEATURES

During the development of the White Paper on RDF data cube potential new features and benefits were identified, including:

- Generic application for providing traceability between results as table or figure and underlying data
- Presentation of RDF data cube results would simplify programming for the presentation of results
- Validation of results using SPARQL queries
- Overview of results using SPARQL queries

These features are addressed in the following sections.

```
%let tabulateOutputDs=work.tab_14_3x01;
proc tabulate data = ADQSADAS missing;
  ods output table=&tabulateOutputDs.;
  where EFFFL='Y' and ANL01FL='Y' and AVISIT='Week 24' and PARAMCD="ACTOT";
  class trtpn sitegr1;
  class EFFFL ANL01FL AVISIT PARAMCD;
  var base chg aval;
  table
      EFFFL*ANL01FL*AVISIT*PARAMCD,
      base chg aval, trtpn*(n*f=F3.0 mean*f=f4.1 stddev*f=F5.2 median*f=f4.1 (min
max)*f=F4.0);
run;

%include "include_tabulate_to_csv.sas" /source;
```

Figure 2 SAS PROC tabulate code to generate table 14.3.01. The results from PROC tabulate is stored in the dataset given by &tabulateOutputDs. The %include statement invokes a generic program converting the PROC tabulate ODS output dataset to .csv file for subsequent converversion to RDF data cubes (see text)

GENERATION OF ANALYSIS RESULTS DATASET

The analysis results were created following the steps below:

1. Generate summary statistics in SAS using PROC TABULATE[6], see example code in Figure 2
    a. Store the analysis results using ODS output and export to a .csv file. The process of converting the ODS output is handled by a SAS program, that interprets the structure of ODS output data to generate the .csv files.
2. Convert .csv files to RDF data cubes using the R-package, which uses the RRDF package[5] that provides an interface from R to Apache Jena[9]
3. Write SPARQL select queries for retrieval of results and metadata, see example in Figure 5
4. Present results using PROC report in SAS.

The naming convention used for associated programs and outputs is as shown below in Listing 2 for generation of Table 14-2.01. The naming convention adopted simplifies the generation of scripts, and are helpful for keeping track of the various components involved.

Listing 2: Filenames of programs and outputs used to generate Table 14-2.01

build-tab2x01.cmd	Windows CMD script for generating the outputs
tab2x01.sas	SAS program generating .csv file with results and meta data
TAB2X01.csv	.csv file with the results for the RDF data cube
TAB2X01-Components.csv	.csv file with meta data for the RDF data cube
tab2x01-ttl.Rmd	R script generating RDF data cube using the .csv files
CDISC-pilot-TAB2X01.ttl	The table as RDF data cube
tab2x01-observations.rq	SPARQL SELECT query to get observations for the data cube
tab2x01.rq	SPARQL SELECT query to get table results in format suitable for presentation in SAS
get-tab2x01-with-proc-groovy.sas	SAS program querying RDF data cube and present as HTML with links (href) to cube observations
tab2x01.html	HTML representation of analysis results
File extensions: .cmd – windows cmd script, .sas - SAS system program, .csv – comma separated values, .Rmd - R markdown, .ttl – RDF turtle, .rq – SPARQL query, html – hypertext markup language	

[6] http://support.sas.com/documentation/cdl/en/proc/67327/HTML/default/viewer.htm#n1ql5xnu0k3kdtn11gwa5hc7u435.htm

The first step is to create a CSV file containing summary statistics. A snapshot of the CSV file is shown in Figure 3.

ittfl	saffl	efffl	comp24fl	compfl	trt01p	procedure	factor	unit	denomina	measure
Y	_ALL_	_ALL_	_ALL_	_ALL_	Placebo	count	quantity			86
Y	_ALL_	_ALL_	_ALL_	_ALL_	Placebo	percent	proportion		ittfl	100

Figure 3: Snapshot from csv file showing summary statistics for ITT population for placebo group

The CSV file is then used to create RDF (.ttl) file using the RRDF R package [3]. The RDF data cube is generated using a R markdown script (.Rmd), which also provides documentation of the generation. Figure 4 shows one observation from the RDF datacube.

```
ds:obs01  a                          qb:Observation ;
        rdfs:comment                 "Statistic for number of records/Statistics for factor with the dimensions XX"@en ;
        rdfs:label                   "1"^^xsd:string ;
        qb:dataSet                   ds:dataset-TAB1X01 ;
        crnd-attribute:denominator   "" ;
        crnd-attribute:unit          "NA"^^xsd:string ;
        crnd-dimension:comp24fl      code:comp24fl-_ALL_ ;
        crnd-dimension:compfl        code:compfl-_ALL_ ;
        crnd-dimension:efffl         code:efffl-_ALL_ ;
        crnd-dimension:factor        code:factor-quantity ;
        crnd-dimension:ittfl         code:ittfl-Y ;
        crnd-dimension:procedure     code:procedure-count ;
        crnd-dimension:saffl         code:saffl-_ALL_ ;
        crnd-dimension:trt01p        code:trt01p-Placebo ;
        crnd-measure:measure         "86"^^xsd:double .
```

Figure 4: Snapshot of one of the observations from the RDF (.ttl) file showing number of patients in ITT population for placebo group

PRESENTATION FROM RDF DATA CUBE

The generated RDF data cubes are queried using SPARQL[7], and the tabular output as html files is created using SAS.

The SPARQL query is performed using a SAS macro that returns the results as a SAS dataset for use with PROC REPORT to generate the table. During development, it was found that instead of restarting the SPARQL endpoint for each new version of the RDF data cube, it is simpler to have a SAS program performing the SPARQL query. PROC GROOVY was used to interface with Apache Jena[8] to load the generated RDF file (.ttl) and perform the query. Apache Jena returns the results as XML, which is processed by the macro to convert it to SAS dataset. The SPARQL query and a subset the results are shown in Figure 5 and Figure 7.

```
select
 ?ittfl ?procedureZ1 ?col1z1URI ?col1z1
where
{
?col1z1URI  a qb:Observation;
      crnd-dimension:comp24fl ?comp24fl   ;
      crnd-dimension:compfl ?compfl   ;
      crnd-dimension:efffl ?efffl     ;
      crnd-dimension:factor ?factorZ1     ;
      crnd-dimension:ittfl ?ittfl     ;
      crnd-dimension:procedure ?procedureZ1 ;
      crnd-dimension:saffl ?saffl   ;
      crnd-dimension:trt01p code:trt01p-Placebo ;
      crnd-measure:measure ?col1z1 .
  filter (?ittfl = code:ittfl-Y) }
```

Figure 5: SPARQL query that retrieves the observation shown in Figure 4

OVERVIEW OF RESULTS USING SPARQL QUERIES

By design, the RDF data cubes can store the title for the results. The SPARQL query in **Figure 6**provides the same information as in Listing 1. The expression "(REPLACE(str(?ds), "^[^-]+-","") as ?shortname)" is a work-around to extract the table name, which is at end of the URI for the dataset given in ?ds variable.

```
prefix rdfs:  <http://www.w3.org/2000/01/rdf-schema#>
prefix qb: <http://purl.org/linked-data/cube#>
prefix rrdfqbcrnd0: <http://www.example.org/rrdfqbcrnd0/>
select (REPLACE( str(?ds), "^[^-]+-","") as ?shortname) ?dslabel
?underlyingData  where {
?ds  a                  qb:DataSet ;
        rdfs:label                      ?dslabel ;
        rrdfqbcrnd0:D2RQ-DataSetName    ?underlyingData .
}
```

Figure 6: SPARQL query that retrieves Table of Contents

	ittfl	col1z1URI	col1z1	procedureZ1
1	code:ittfl-Y	<http://www.example.org/dc/tab1x01/ds/obs01>	86	code:procedure-count
2	code:ittfl-Y	<http://www.example.org/dc/tab1x01/ds/obs02>	100	code:procedure-percent

Figure 7: Results produced by the SPARQL query show in Figure 5

PROVIDING TRACEABILITY FOR RESULTS IN HTML FILES

The most obvious approach for providing reference to a result is to use the URI for the observation, e.g. for the RDF data cube observation shown in Figure 4 and Figure 7 the reference is made using

```
<a href="http://www.example.org/rdf-data-cube/obs01">86</a>
```

It is being investigated to use RDFa to represent a citation from an RDF datacube, the formatting, and the reference to RDF data cube observation. Figure 8 and 9 uses http://rdfa.info/play/ to represent the RDFa markup as a graph and as RDF.

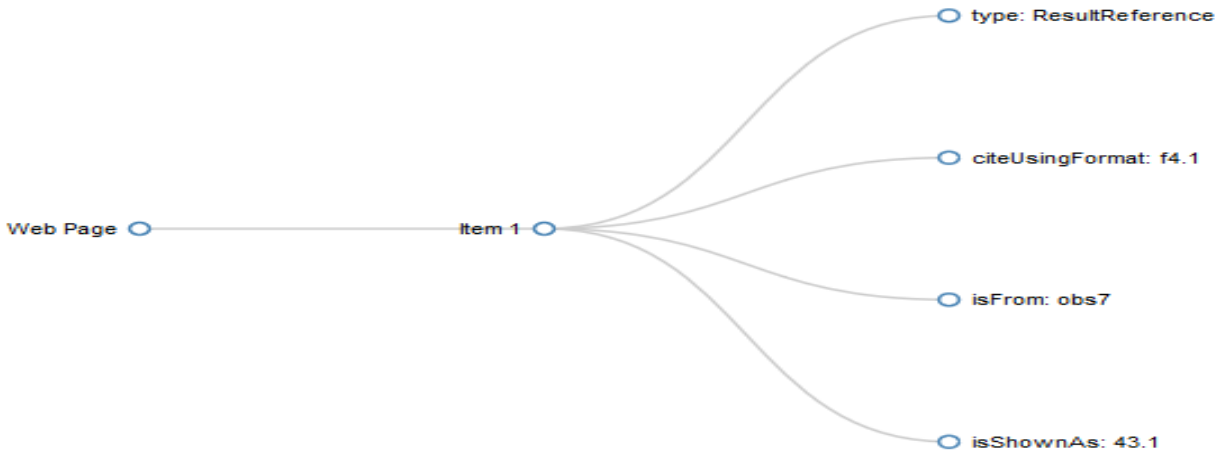


Figure 8: Visualization of RDFa

```
<span vocab="http://www.example.org/citingForCSR/" typeof="ResultReference">
<span property="citeUsingFormat" content="f4.1">
<a property="isFrom" href="http://www.example.org/rdf-data-cube/obs7">
<span property="isShownAs">43.1</span>
</a></span></span>
```

Figure 9: RDFa embedded in HTML referencing an RDF data cube observation

```
@prefix rdfa: <http://www.w3.org/ns/rdfa#> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .

<http://rdfa.info/play/> rdfa:usesVocabulary
<http://www.example.org/citingForCSR/> .

_:1  rdf:type <http://www.example.org/citingForCSR/ResultReference>;
   <http://www.example.org/citingForCSR/citeUsingFormat> "f4.1";
   <http://www.example.org/citingForCSR/isFrom>
                                  <http://www.example.org/rdf-data-cube/obs7>;
   <http://www.example.org/citingForCSR/isShownAs> "43.1" .
```

Figure 10: RDFa markup as RDF/Turtle corresponding to the RDFa embedded in HTML in Figure 8

TRACEABILITY FROM RDF DATA CUBE OBSERVATION TO DATA

An RDF data cube observation (Figure 3) provides the dimensions for the contributing data. For each dimension, the RDF data cube codelist contains the original value in the data. These values are used to select the data rows and presented together with the identification, e.g. variable USUBJID for dataset ADSL, and if applicable, the variable summarized. These values are used to select the data rows and presented together with the identifier variable, e.g. USUBJID for dataset ADSL, and if applicable, the variable being summarized. This approach was implemented in the application using SPARQL queries built from the values for the specific RDF data cube observation. When the underlying data are available, the majority of descriptive statistics can be derived using the SPARQL aggregated functions, and thereby validate the RDF data cube contents.

A more general approach is being experimented, by rephrasing the matching: the required rows in the datasets are those where zero (0) of the variables are not matching the values in the corresponding dimensions in the RDF data cube observation. This leads to a surprisingly short and generic SPARQL query.

PUTTING IT ALL TOGETHER – APPLICATION FOR PRESENTING TABLES AND SHOW TRACEABILITY

A browser based application was developed for showing the results and perform the queries [6]. To demonstrate traceability from results to data, ADaM datasets were transformed to RDF using D2RQ[9].

The application presents the SAS generated html version of the tables and shows how the linking between results and data can be implemented. The overall structure of the process and the application is shown in Figure 11, with screenshots in **Figure 12**.

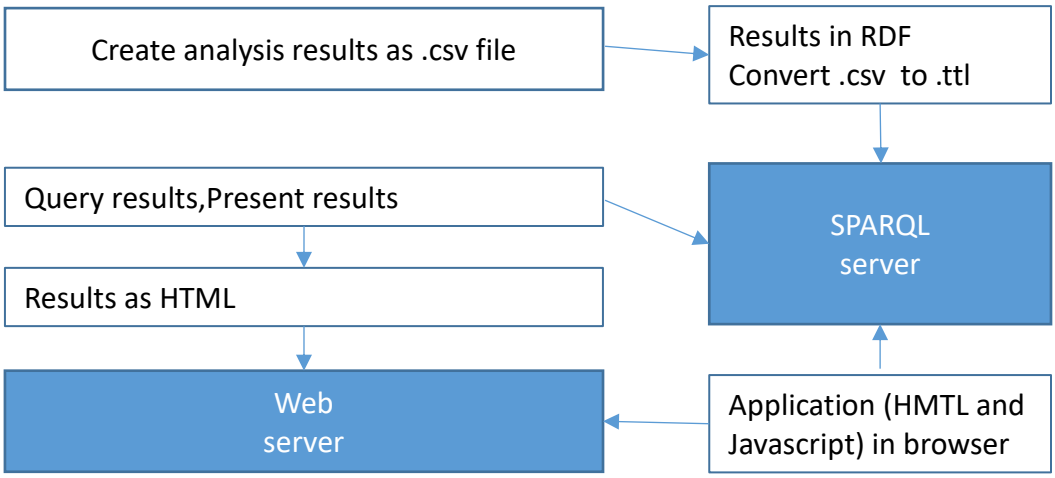
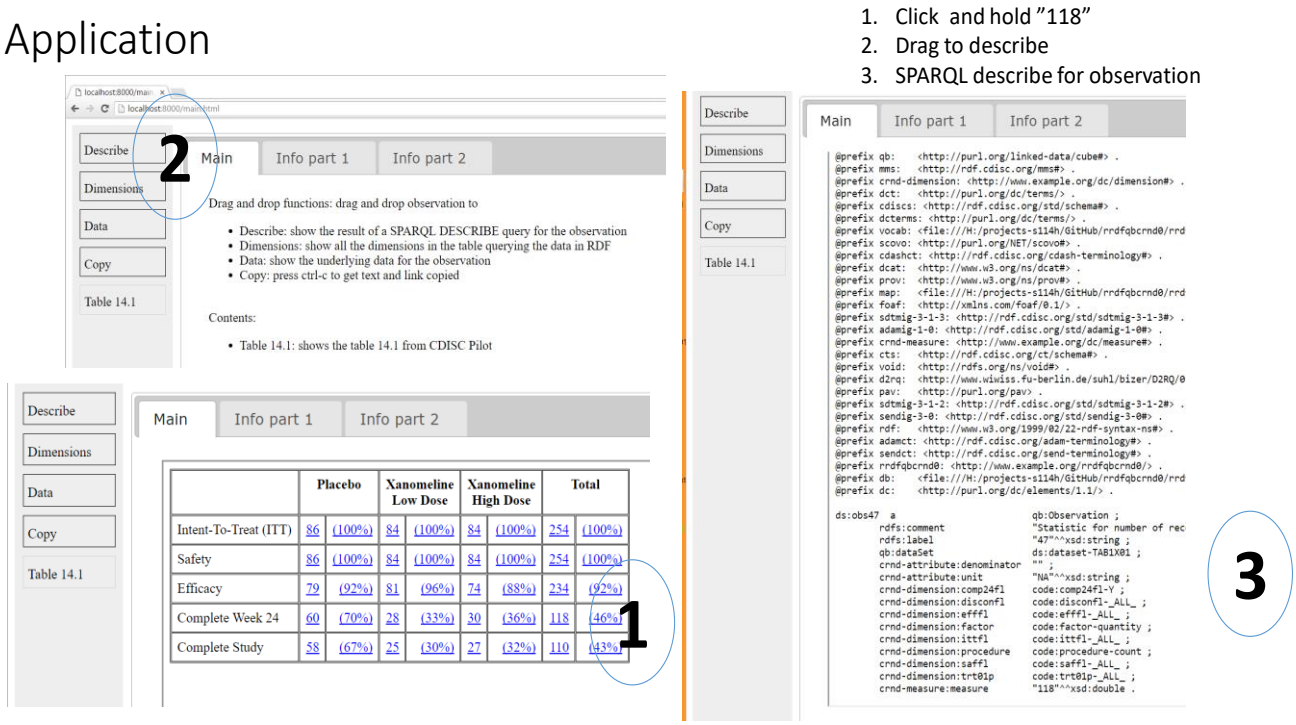


Figure 11: Block diagram showing different components of the application

[9] http://d2rq.org/



http://www.phusewiki.org/docs/Conference%202015%20TT%20Papers/TT07.pdf
http://www.phusewiki.org/docs/Conference%202015%20TT%20Presentations/TT07_Dude_Wheres_My_Graph.pptx

Figure 12: Screenshots showing views of the application

EVALUATION

GENERATION OF RESULTS

It is feasible to use SAS PROC TABULATE to generate results. However, the usual approach for presenting results often involves creation of new presentation variables. For the RDF data cubes only variables in the ADaM datasets can be used. Therefore some of the logic used for presentation is not represented in the RDF data cubes, but implemented in the presentation program. The specification of presentation should also be part of the metadata.

Handling of missing data is not straightforward in PROC TABULATE for derivation of percentages, however several approaches exist[10]. It was found that the AR&M specifications of using an attribute to specify the denominator could be handled better by having the factor property specify the denominator. For generating AE tables, a workaround was used by defining indicator variables DISTINCT_USUBJID that is 1 for the first value and 0 for the combination. This would be simplified if PROC TABULATE had a summary statistic counting distinct values of a class variables, like “count(distinct USUBJID)” in PROC SQL.

It was initially anticipated to show definition of new RDF data cubes with the sub-population added as a dimension. However, when using PROC TABULATE, it was realized that this is easily done by extending the table statement with the variable and defining the variable as a class variable.

GENERATION OF RDF DATA CUBES FROM CSV FILES

The R-package was usable, but the installation of the R-package proved to be time consuming. The generation of SPARQL queries in the package proved to be helpful.

[10] https://support.sas.com/resources/papers/proceedings13/134-2013.pdf

PRESENTATION FROM RDF DATA CUBE

SPARQL queries were used to get a dataset suitable for presentation and PROC REPORT was used to present the results. SPARQL queries returning columns corresponding to the columns in the table were used initially. While the generation of the SPARQL query is automated to some extent in the R script it proved tedious to adapt, especially for tables where a cell is the combination of three numbers (median, min and max). For presentation in SAS the authors found that a more simple approach could be to query the observations and then do the re-arrangement into row and columns using SAS.

OTHER ISSUES

The setup of SPARQL server and web server and the many parts in the R-package makes it quite complex and time consuming to set-up.

CONCLUSION

It is feasible to generate and store analysis results as RDF data cubes. A formal specification of presentation layout would be beneficial and could be used to store presentation information as RDF. As demonstrated, the traceability from result to underlying data is straightforward using linked data approach. More experience working with RDF and SPARQL query language will help to simplify the process.

Overall, the potential of using the proposed approach has been demonstrated, but more work is needed to make the approach usable for production.

The following topics could be investigated further

1. Use the R tables package[11] to generate results and RDF data cubes in one strep.
2. Generate and store metadata for the script using the approach from the PhUSE scripting group[12]
3. The direct generation of RDF from SAS or R as text files with either SPARQL INSERT or SPARQL CONSTRUCT or as turtle is still under development
4. Use XSL transformation of RDF/XML for subsequent presentation, Alternatively, there are other tools for presenting RDF development (for example Dokeieli[13])
5. Use proposed approach to create figures
6. Hyperlink results in CSR body of text to the table
7. Suggest format for analysis results as a CDISC standard (like the ADaM specification)– essentially data cubes – matching RDF data cubes
8. Suggest standard for representing DEFINE-xml as RDF
9. RDF representation of ADaM datasets connecting to CDISC standards in RDF

ACKNOWLEDGMENTS

We thank all participants in the PhUSE CS Semantic Technology Working Group, Analysis Results & Metadata Project for discussion and comments.

REFERENCES

1. PhUSE CS Semantic Technology Working Group, Analysis Results & Metadata Project. ”Improving the Analysis Results Creation and Use Process: Modeling Analysis Results & Metadata as Linked Data”. [Draft White paper, publication pending on PhUSE Wiki]
2. PhUSE CS Semantic Technology Working Group, Analysis Results & Metadata Project. “Clinical Research and Development (CRND) RDF Data Cube Structure Technical Guidance.” [Draft White paper, publication pending on PhUSE Wiki]
3. Marc Andersen, Generating R-RDF Data Cube for Clinical Research & Development, work from a subgroup of PhUSE Semantic Technology Project, https://github.com/phuse-org/rrdfqbcrnd
4. Brega, John, Colins, Linda. Beyond OpenCDISC: Using Define.xml Metadata to Ensure End-to-End Submission Integrity. PharmaSUG SDE 2015 In Gilead offices, Foster City, CA

[11] https://cran.r-project.org/web/packages/tables/vignettes/tables.pdf

[12] https://github.com/phuse-org/phuse-scripts/blob/master/MetaData_template.yml

[13] https://github.com/linkeddata/dokieli

5. Egon Willighagen. Accessing biological data in R with semantic web technologies. 2014; Available from: https://doi.org/10.7287/peerj.preprints.185v3
6. Tim Williams, Marc Andersen: Dude, where's my graph?' RDF Data Cubes for Clinical Trials Data. Presented at 2015 PhUSE Annual Conference, Vienna
7. Marcelina Hungria: Delivering Statistical Results as an RDF Data Cube : A Simple Use Case to Illustrate the Process of an RDF Data Cube Creation and the Link to the RDF Representation of the CDISC Standards. In North Bethseda, MD; 2014
8. Tim Williams: A Primer on Converting Analysis Results Data to RDF Data Cubes using Free and Open Source Tools, presented at 2014 PhUSE annual conference, London, United Kingdom
9. Apache Jena, http://jena.apache.org/

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the authors at:

Marc Andersen
StatGroup ApS
Fruebjergvej 3
DK-2100 Copenhagen OE
Email: mja@statgroup.dk

Suhas R. Sanjee
Merck & Co., Inc.
351 N. Sumneytown Pike
North Wales PA 19454
Email: suhas_sanjee@merck.com

Marcelina Hungria
DIcore Group, LLC
NJ, USA
Email: mhungria@dicoregroup.com