PhUSE 2015

Paper TT07

‘Dude, where’s my graph?’ RDF Data Cubes for Clinical Trials Data.

Marc Andersen, StatGroup ApS, Denmark
Tim Williams, UCB BioSciences Inc, USA

ABSTRACT

The concept of Linked Data conjures images of effortlessly traversing from one data point to the next on a voyage of semantic discovery. Contrast this with the seemingly countless number of subject→predicate→object triples and their complex relationships in a Resource Description Framework (RDF) data source and comprehension of even familiar data becomes challenging. The added value of RDF can appear elusive without tools to facilitate data exploration and display.

This paper introduces approaches to visualizing RDF as Linked Data networks and presenting summary results as interactive tables. Visualization is a powerful tool for designing, exploring, and validating data models like the RDF Data Cube. Interactive tabular display enables presentation of the data and metadata. RDF provides traceability from source to result. Alternating between visualization and tabular display allows new forms of data exploration. The suitability of RDF Data Cubes is demonstrated by representing and displaying results for disposition, efficacy and adverse event data.

This paper is presented in support of the PhUSE Analysis Results & Metadata project.

Keywords: clinical trials results, CDISC, define-xml, force network graph, data visualization, D3js, JSON, Linked Data, metadata, R, RDF, RDF data cube, Resource Description Framework, SAS, semantic web, summary results, traceability

INTRODUCTION

Use of the term Directed Graphs in the context of describing Linked Data may lead you to believe that the data is immediately accessible as interactive graphs of connected triples. Opening your first RDF file or viewing the results of your first SPARQL query may lead you to exclaim in disappointment, “Dude, where’s my graph?” Your quest for RDF visualization will identify a few commercial and open source tools, although most will require some form of customization to fit your purpose. Many visualization tools focus on ontology development and are not easily adapted to viewing other RDF data. Visual learners must find other ways to graph RDF as a means to understand data models and content.

Clinical trial results are typically present in Appendix 14.2 of the Clinical Study Report (CSR). All data contributing to results summaries are ideally available in the listings in CSR Appendix 16.1. Linked data is a natural choice to enhance traceability between results and underlying data. However, beyond the usual use of a hyperlink to click through to the data, pathways for navigation are still unchartered. Here we present approaches for navigation which will be beneficial both during programming, report writing, and subsequent report review by applicable parties.

WHY VISUALIZATION?

RDF triples are often depicted as subject node joined to an object node by directed line, known as an edge, to represent the predicate (**Figure 1 a**). Force Network (FN) graphs are a natural choice for depicting Linked Data triples. FN algorithms place nodes in two- or three-dimensional space while minimizing the crossing of edges. Forces between nodes and specification of edge lengths produce visually pleasing interactive graphs. Graphs can be highly customized with nodes of different shapes, colours, mouse-over effects, and click-through events to provide additional information. Node click-through functionality provides the opportunity to integrate FN graphs with Linked Data URIs, resulting in interactive RDF exploration.

Simple triple relations build into complex relationships. The predicates that give meaning to the data linkages are often part of ontologies (which are themselves Linked Data) whose interpretation is made easier using interactive visualizations. Visual representation also facilitates construction of SPARQL queries and interpretation of the returned results.

Visualization makes Linked Data accessible for people inexperienced with the underlying data or data models. The PhUSE Analysis Results & Metadata project found that graphical representations of the RDF Data Cube promoted understanding, improved development, identified errors, and led to informed discussion during model development. Interactive graphical and tabular displays of Linked Data benefit non-technical audiences who lack in-depth knowledge of the Linked Data principles.

GRAPH TRIPLES FROM ANY SPARQL QUERY USING R

R provides an easy entry point for working with Linked Data. Packages like `rrdf` and `rrdflibs` provide the capability to read and write RDF data files and interact with local RDF files as well as local and remote SPARQL endpoints.

SETTING UP A SPARQL ENDPOINT

Information on setting up a SPARQL endpoint is widely available. We found the "A how-to guide for creating a Linked Data site" [1] helpful. Apache Jena-Fuseki [2] and Virtuoso [3,4] are popular choices for serving RDF and either may be installed to provide a SPARQL endpoint on your local machine. The authors found Virtuoso's Faceted Browser and Text Search capabilities particularly useful. Virtuoso is well documented and helpful guides are available [5]. Due to SPARQL being a W3C standard [6] there are many endpoints to use [7].

Use of a local endpoint for processing by client-side web applications must include enabling Cross-Origin Resource Sharing (CORS). The "Allow-Control-Allow-Origin" extension provides a browser-based solution for users of Google Chrome. CORS may also be enabled on the server side, with the advantage of not having to remember to activate a plugin or extension. CORS support for Virtuoso is easily configured within Virtuoso Conductor [8].

GRAPHING WITH R

Several R packages can be used to graph RDF triples directly within the R application (**Table 1**). Other packages make it easy to export the data in formats readily consumed by other graphing libraries or applications.

Table 1 Select R packages for network graphs

R Package	Description
d3Network	Tools for creating D3 JavaScript network, tree, dendrogram, and Sankey graphs
ggplot2	An implementation of the Grammar of Graphical Models
igraph	Network analysis and visualization
networkD3	Tools for creating D3 JavaScript force network and Sankey graphs from R
qgraph	Network-based data visualization

Graphing triples directly within R using the package networkD3 is shown in **Figure 1 (c)**. R code was used to query the DBPedia endpoint `dbpedia.org/sparql` for the subject:

```
http://dbpedia.org/page/Vienna
```

and its predicate

```
dbpedia-owl:populationTotal
```

in order to obtain the population value object. The result was then visualized as a simple network graph. The graph can be further enhanced within R by incorporating D3.js elements and use of R Shiny (not shown). When a large number of visual customizations and functionality is required, it may be advantageous to export the data as JSON for display in D3.js web pages, as shown in **Figure 1 (d)**. The latter approach is used by the AR&M Working Group for visualizing the RDF Data Cube model.

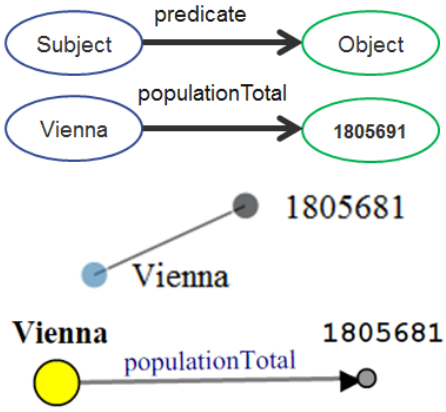


Figure 1 Triples Representing the Urban Population of Vienna
a) General triple pattern of Subject, Predicate Object
b) Graphical representation
c) Visualized using R, SPARQL and networkD3
d) Visualized using R, SPARQL and D3.js

INTRODUCING D3.JS

D3.js, also known as Data-Driven Documents or D3, is a free JavaScript library created by Mike Bostock [9] to display digital data in rich, dynamic, and interactive visualizations within web browsers. "Data-Driven" highlights the intimate connection between the visual and its underlying data. Since its release in 2011, an increasing number of resources are available to help users new to D3.js [10]. Stackoverflow.com (using hashtag #d3.js) is an excellent resource for D3.js as well as for R, R-SHINY, JavaScript, and a host of other tools.

While learning D3.js can be daunting for new users, installing D3.js on a standalone computer could not be easier. All you need in addition to the D3.js library is a web server. We recommend Python's `SimpleHTTPServer` that is part of the base installation.[1] The HTTP service is started from the command line within the folder that serves as the root for your D3 library and the pages you wish serve. MS-Windows users may wish to configure a desktop shortcut similar to the following: here, the first line changes directory into the folders where the D3 content will be served.

```
cd C:\_sandbox\d3
python -m http.server
```

After your web server is installed and running you must create a folder within the server directory to store the D3.js library. You may now start programming using the many examples on the web for inspiration and guidance, like those at http://d3js.org/. Among the online examples are several for FN graphs, although it is likely that no single one will provide all of the functionality you may wish to display for your Subject->Predicate->Object relations. Once you have mastered the basics of how to represent and label nodes and edges, you will be able to create very detailed interactive graphs for complex data models and data subsets.

GRAPHING FEDERATED QUERIES

Figure 2 shows an example of how D3.js FN graphs can be used to display and explore data returned from querying Linked Open Data. R was used to query the SPARQL endpoints DBPedia, MeSH, Drug Bank, PubMed, and Clinical Trials for information on a specific drug. SPARQL can query across diverse data sources, including both local and remote sources in the same query. These are known as federated queries [11].

Triples from the query were exported as JSON for display in a D3.js webpage. Clicking any node in the display will result in either the resolution of a URI value or the delivery of a web page with more information displayed as linked triples. For example, clicking on one of the nodes in the Clinical Trials subgraph will yield a web page of hyperlinked information about that trial, including sponsor, abstract, investigators, and other information. Much of the information on the resulting page is also hyperlinked URIs, so you can follow these additional links to more information.

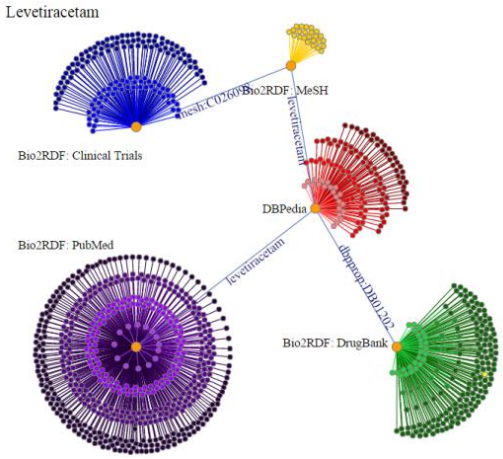


Figure 2 Clinical Trial data from remote SPARQL Endpoints. Screen shot of interactive D3.js graph.

Displays are not limited to Force Network graphs. Circle Packs, Sunbursts, or Wheel charts may group clinical trials by sponsor and phase, with each trial linked out to more information at the Clinical Trials endpoint. Choropleth maps can display trial locations by country or area, with links back to the endpoint for more information. By using Linked Data, these graphs become an interactive component in data exploration.

GRAPHING THE AR&M RDF DATA CUBES

The PhUSE Analysis Results & Metadata project is defining an RDF Data Cube [12] model and R package to convert clinical trials analysis results to RDF. The group's technical guidance document for the structure of the data cube model [13] contains FN graphs for a high-level overview of the major components (**Figure 3**), the complete model (not shown), and sub graphs like the model for an observation (**Figure 4**). These graphs have been invaluable for introducing new team members to the concepts, validating the cube structure, and creating SPARQL queries for both the model and its data. These diagrams are subject to change as cube model continues to evolve.

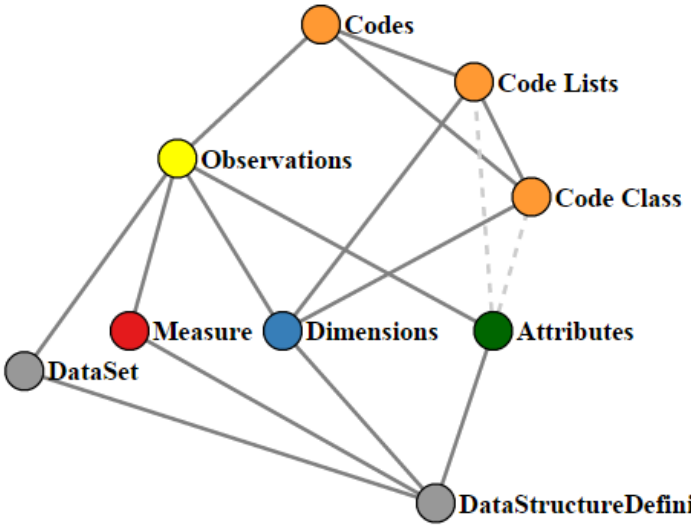


Figure 3 Major Components in the PhUSE AR&M RDF Data Cube Structure Model.

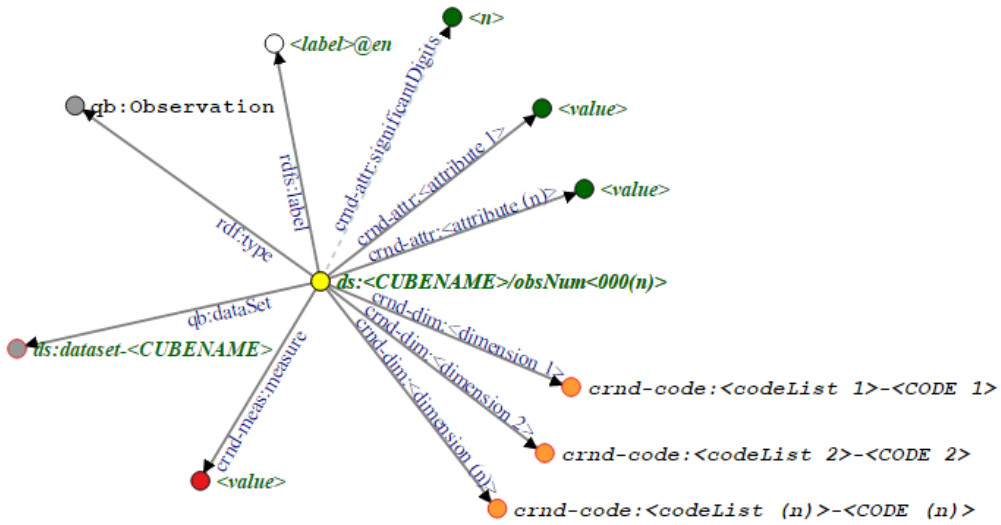


Figure 4 Subgraph for a cube observation.

Visualizations of dataset size, content, structure, and associated metadata will begin to play an important role when the model is applied to clinical trials data. The AR&M Project is investigating the use of interactive D3.js tree diagrams to explore hierarchical code lists converted to RDF from Define-XML. Nodes in the tree can link out to other data sources that in the future may include the Study Protocol, Statistical Analysis Plan, a master online code list, or other sources of linked data. **Figure 5** illustrates this concept using a link to DBPedia for further explanation of the female code "F".

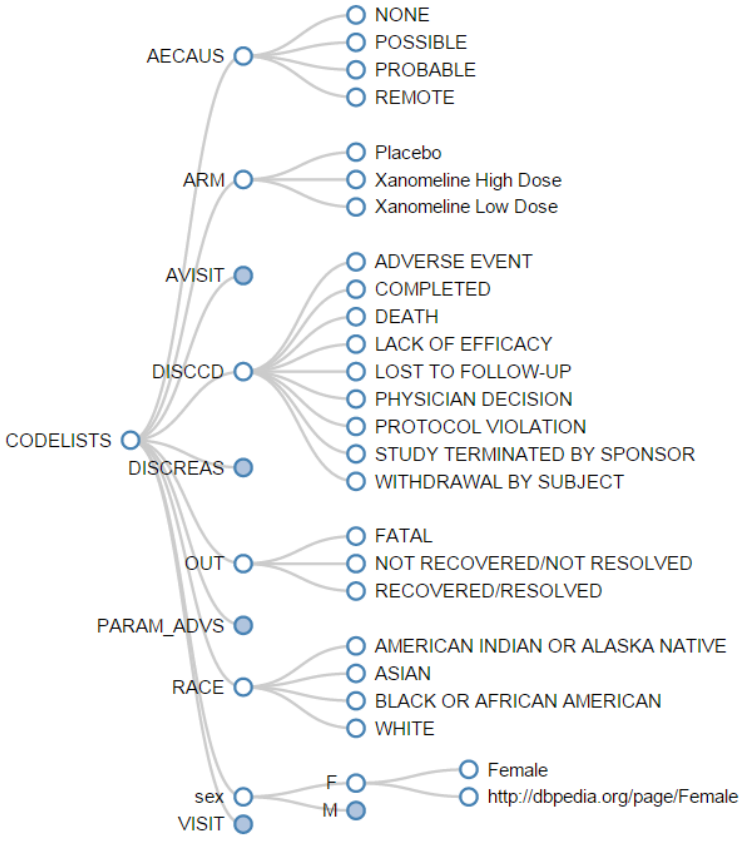
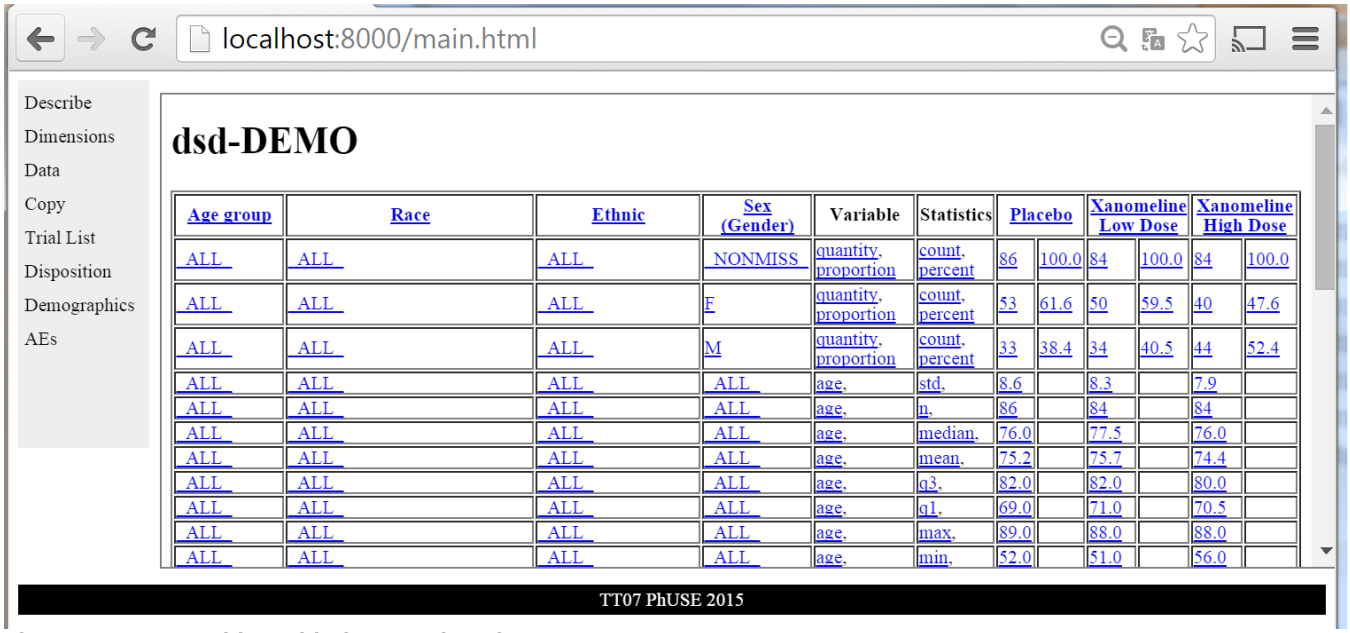


Figure 5 Interactive code list from define.xml (excerpt).

INTERACTIVE SUMMARY TABLES

To illustrate the potential uses in a familiar setting we made a browser-based prototype showing the possibilities for combining traditional rendition of tables with linked data concepts. The left column shows the possible actions. The main part of the screen to the right shows the selected results, a table, or results of a query. Dragging and dropping a hyperlinked item over an action, invokes the action with the hyperlink as parameter. Clicking on an action either shows a table, for example the "List of Trials", or provides a description of the action. Admittedly, this is not the ideal user interface; we choose the drag-and-drop approach as HTML5 makes it possible to use with relative small programming effort. In addition, we find the drag-and-drop is quite cool.



Age group	Race	Ethnic	Sex (Gender)	Variable	Statistics	Placebo		Xanomeline Low Dose		Xanomeline High Dose	
ALL	ALL	ALL	NONMISS	quantity, proportion	count, percent	86	100.0	84	100.0	84	100.0
ALL	ALL	ALL	F	quantity, proportion	count, percent	53	61.6	50	59.5	40	47.6
ALL	ALL	ALL	M	quantity, proportion	count, percent	33	38.4	34	40.5	44	52.4
ALL	ALL	ALL	ALL	age,	std,	8.6		8.3		7.9	
ALL	ALL	ALL	ALL	age,	n,	86		84		84	
ALL	ALL	ALL	ALL	age,	median,	76.0		77.5		76.0	
ALL	ALL	ALL	ALL	age,	mean,	75.2		75.7		74.4	
ALL	ALL	ALL	ALL	age,	q3,	82.0		82.0		80.0	
ALL	ALL	ALL	ALL	age,	q1,	69.0		71.0		70.5	
ALL	ALL	ALL	ALL	age,	max,	89.0		88.0		88.0	
ALL	ALL	ALL	ALL	age,	min,	52.0		51.0		56.0	

Figure 6 Demographics table (screen dump).

The "Describe" action shows the information related to the item by use of the Virtuoso faceted browser.

The "Dimensions" action shows the dimensions together with the code list value for the item, and is the output of a SPARQL query.

The "Data" action shows the underlying data for a result, and is the output of SPARQL query accessing the data. The transformation from SAS export files to RDF was done using R, sqlite and D2RQ [14], where the later transforms a relational database into RDF.

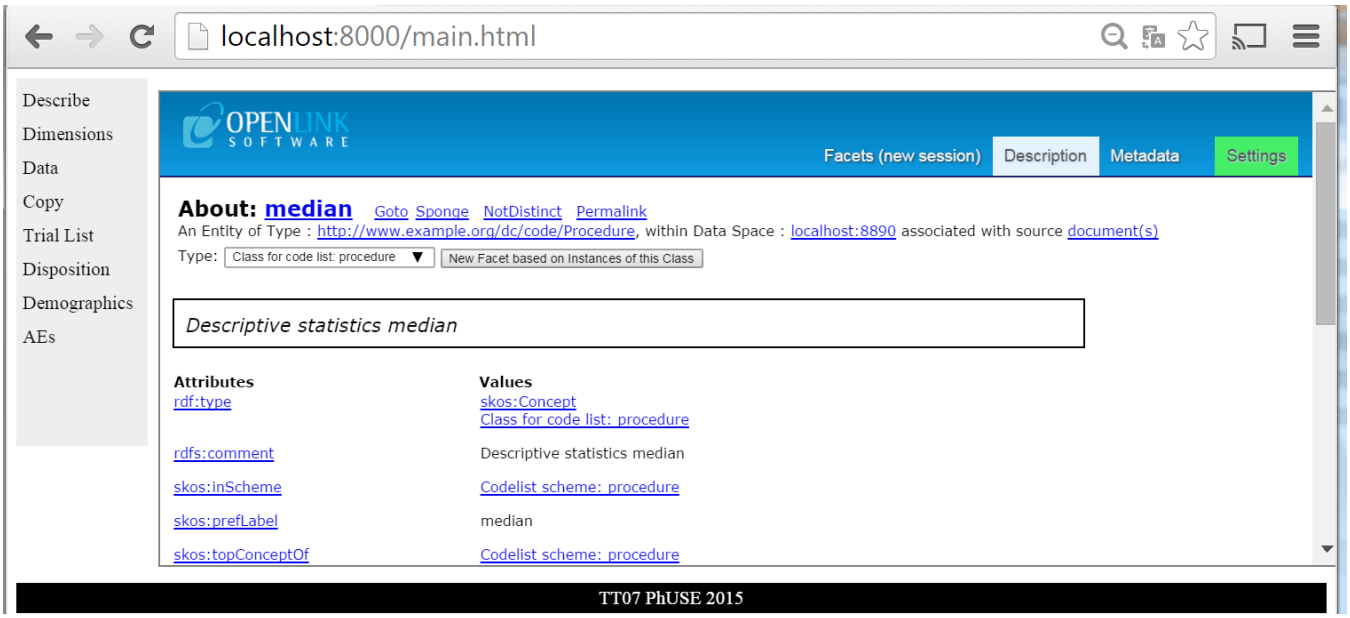


Figure 7 Faceted view for median from demographics table.

The data presentations like “Disposition” shows results in layout corresponding the table appendix in the CSR. We created HTML renditions of the sample RDF data cubes in the AR&M project with URIs referring to the underlying observations, code lists, and labels in the RDF data cube. For simplicity, we choose only to use traditional links with HREF instead of using RDFa for embedding RDF into HTML.

When dragging and dropping a hyperlinked item on the “Copy” action the text and URI is copied to the clipboard, which then can be pasted into another document, providing traceability for the copy-paste operation as the URI is included.

Further actions present the result of SPARQL queries, e.g. “Number of patients in FAS population in a treatment arm” in all tables. Finally, D3js generated graphical displays can also be integrated in this example.

INTEGRATION ALONG THE CLINICAL TRIALS DATA LIFECYCLE

The focus here has been on results and underlying data and methods for describing the structure of the data. However the approach can be extended to the whole clinical trials data. This is already done by the rdf.cdisc.org implementation of CDISC in RDF [15], thereby providing the link from data to results. The possibility of copying text the associated URI representing a result shows that the linked data techniques may be used for report writing, improve traceability, and automate QC for clinical study reports. Markup of the text with URIs for results and references to concepts in more general ontologies have the potential of bringing the vision of linked data to the clinical study report.

CONCLUSION

Visualization and interactive summary tables are only a small part of the capabilities provided by Linked Data. Web Ontology Language, logic, and reasoning open up additional value for clinical trials data that can be linked into summary tables and visualized using these and other approaches. Traceability from results to data is readily available. Putting the pieces together is the next challenge.

REFERENCES

1. Sarven Capadisli. *A how-to guide for creating a Linked Data site.* <http://csarven.ca/how-to-create-a-linked-data-site> (Retrieved: 01-Sep-2015).
2. The Apache Software Foundation. *Apache Jena Fuseki Documentation / Fuseki2*. <http://jena.apache.org/documentation/fuseki2/index.html> (Retrieved: 01-Sep-2015).
3. OpenLink Software. *Virtuoso Universal Server*. <http://virtuoso.openlinksw.com/> (Retrieved: 01-Sep-2015).
4. OpenLink Software . *Using Virtuoso Open-Source Edition on Windows.* <http://virtuoso.openlinksw.com/dataspace/doc/dav/wiki/Main/VOSUsageWindows> (Retrieved: 01-Sep-2015).
5. Marco Fossati. *The complete tutorial for RDF data ingestion in Virtuoso*. <https://confluence.deri.ie:8443/display/webstar/The+complete+tutorial+for+RDF+data+ingestion+in+Virtuoso >

(Retrieved: 01-Sep-2015).

6. W3C. *SPARQL 1.1 Query Language - W3C Recommendation 21 March 2013*. <http://www.w3.org/TR/sparql11-query/> (Retrieved: 01-Sep-2015).
7. Wikipidia. *List of SPARQL implementations*. <https://en.wikipedia.org/wiki/List_of_SPARQL_implementations> (Retrieved: 01-Sep-2015).
8. Monsur Houssain and Michael Hausenblas. CORS on Virtuoso. <http://enable-cors.org/server_virtuoso.html> (Retrieved: 01-Sep-2015).
9. Mike Bostock. *Mike Bostock's D3.js examples*. <http://bost.ocks.org./mike/> (Retrieved: 01-Sep-2015).
10. The Big List of D3.js Resources . *Dashing D3.js.* [Online] [Cited: 08 10, 2015.] https://www.dashingd3js.com/d3-resources/the-big-list-of-d3-resources (Retrieved: 01-Sep-2015).
11. SPARQL 1.1 Federated Query. W3C Recommendation. *W3.ORG.* [Online] W3C, 2013. [Cited: 08 14, 2015.] http://www.w3.org/TR/sparql11-federated-query/ (Retrieved: 01-Sep-2015).
12. R. Cygniak, D. Reynolds and J. Tennison . *The RDF Data Cube Vocabulary. W3C Recommendation 16 January 2014*. <http://www.w3.org/TR/vocab-data-cube> (Retrieved: 01-Sep-2015).
13. 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]*.
14. The D2RQ Platform. *Accessing Relational Databases as Virtual RDF Graphs.* <http://d2rq.org/> (Retrieved: 01-Sep-2015).
15. CDISC. *CDISC Standards in RDF*. <http://www.cdisc.org/rdf> (Retrieved: 01-Sep-2015).

ACKNOWLEDGEMENTS

The authors are indebted to the following people and organizations:

- Ian Fleming for suggesting D3 as a visualization tool
- Mike Bostock for D3
- PhUSE Analysis Results & Metadata Project team members
- Apache foundation for Jena Fuseki and OpenLink Software for Virtuoso
- R-project for providing R, Egon Willighagen for providing the RRDF R package for handling RDF data.

This paper is based on free, open-source software and the efforts of volunteers in PhUSE working groups. Please support those who donate their time and expertise through your own collaboration, participation, and promotion of these activities.

CONTACT INFORMATION

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

Marc Andersen
StatGroup ApS
Copenhagen, Denmark
mja@statgroup.dk
https://www.linkedin.com/in/marcjandersen

Tim Williams
UCB BioSciences, Inc
Raleigh, NC
tim.williams@ucb.com
https://www.linkedin.com/in/timpwilliams

Interactive summary tables

Graphing triples with R and D3.js

ENDNOTES

[1] Another good alternative for MS-Windows is the WAMP server (http://www.wampserver.com), with additional options available for Mac (MAMP) and Linux (LAMP). Apache Server (http://httpd.apache.org/) is another alternative.