SWISH DataLab: A Web Interface for Data Exploration and Analysis

Tessel Bogaard[1]($\boxtimes$) , Jan Wielemaker[1,2] , Laura Hollink[1] ,
and Jacco van Ossenbruggen[1,2]

[1] Centrum Wiskunde & Informatica, Amsterdam, Netherlands
{Tessel.Bogaard,J.Wielemaker,L.Hollink,Jacco.van.Ossenbruggen}@cwi.nl
[2] Vrije Universiteit Amsterdam, Amsterdam, Netherlands

Abstract. SWISH DataLab is a single integrated collaborative environment for data processing, exploration and analysis combining Prolog and R. The web interface makes it possible to share the data, the code of all processing steps and the results among researchers; and a versioning system facilitates reproducibility of the research at any chosen point. Using search logs from the National Library of the Netherlands combined with the collection content metadata, we demonstrate how to use SWISH DataLab for all stages of data analysis, using Prolog predicates, graph visualizations, and R.

Keywords: Prolog · R · Data processing · Data mining

1 Introduction

Data is ubiquitous, and so are tools supporting data analysis. More often than not, different tools are used for different stages of the analysis. For example, the preprocessing and exploration of data is handled in one tool using one programming language and the analysis in a completely different environment, with scripts spread out in different files stored locally. This interferes with transparency, shareability and reproducibility of the research. SWISH DataLab[1] provides a web interface. It is a Wiki-like collaborative environment combining processing, exploration, and analysis of data, supporting transparency of the choices made. It blends the clarity of Prolog with the statistical computing power of R[2]. Data cleaning and creating concepts and abstractions over the data benefit from the elegance of Prolog's rule-based logic programming paradigm; the statistical analysis and visualization are the strength of R. The combination of logic programming and R has been shown in [1], and applied in the context of biomedical research, e.g., [2], and in sentiment analysis of social media, e.g., [3]. SWISH DataLab integrates SWI-Prolog and R into a single computational environment accessible through a shared web interface.

---

[1] A version of SWISH for teaching Prolog is available online: http://swish.swi-prolog.org/.

[2] https://www.r-project.org/.

Using SWISH DataLab, it is easy to quickly try out different data abstractions on a sample and evaluate the impact on the results. SWISH DataLab is currently being developed as a collaborative environment for responsible data science using data from the National Library of the Netherlands. In this case study we combine six months of search logs with the content metadata from the historical newspaper collection. We explore and process the datasets using the Prolog programming language, and use R for a statistical analysis of the data.

2 SWISH

SWISH DataLab is an instantiation of SWISH (SWI-Prolog for SHaring), geared to data analysis. In SWISH, a web interface gives access to the computational environment. The development of SWISH [4,5] is influenced by Jupyter[3] and JSFiddle.[4] With JSFiddle it shares the model of a server where people can save and share programs and documents. From Jupyter the concept of *notebooks* is taken, a mixture of text and program fragments that can be edited in a browser [6].
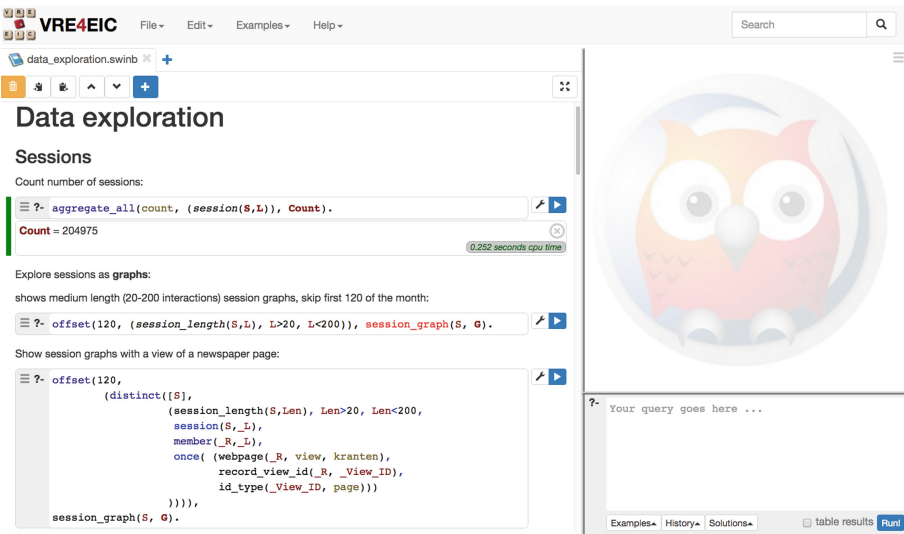

Fig. 1. The interface of SWISH DataLab, showing a Prolog notebook with some queries on the left and a query window on the right.

In SWISH, programs can be executed from the browser and the results appear in the browser as plain answers or rendered as tables, graphs or charts (Fig. 1). The core language of SWISH is SWI-Prolog. Unlike Jupyter it does not support
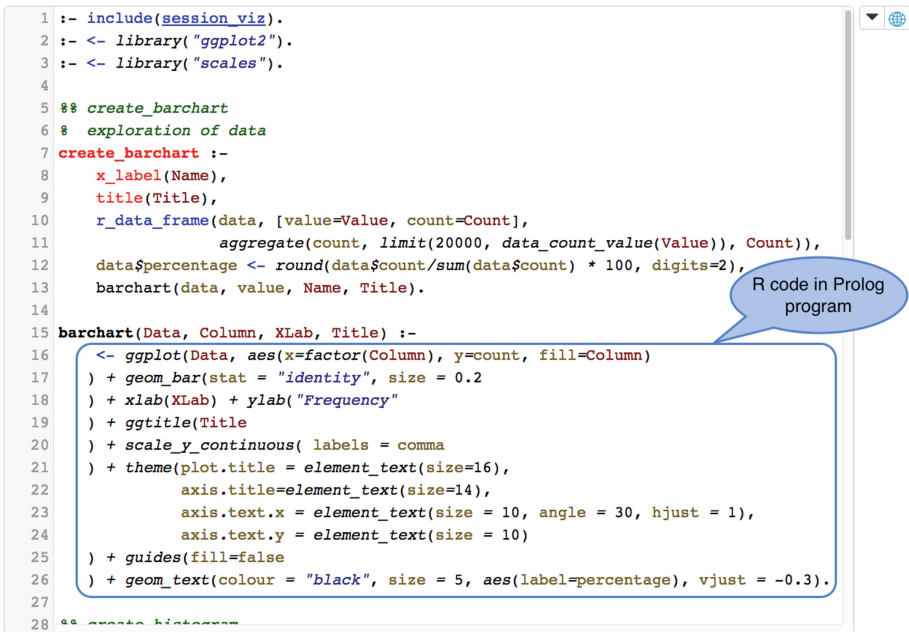
---

[3] http://jupyter.org/.
[4] https://jsfiddle.net/.

other languages directly. Instead, it allows for extending the Prolog core by binding it to other systems. On the *backend* it may be connected to external data using e.g., ODBC or SPARQL. Computational flexibility can be extended using e.g., R. The complementary power resulting from combining logic programming and R has been demonstrated in [1]. The *frontend* can be enhanced using server-side support from e.g., R or Graphviz[5] as well as client-side support using e.g., D3.js[6] or C3.js[7] as visualization methods.

2.1 SWISH and the R Programming Language

R is made available to Prolog by means of the Rserve[8] package. This implies that every SWISH user has a private instance of R, providing both isolation and concurrency.

R can be accessed from Prolog using two distinct mechanisms. The predicate *Result <- Expression* uses Prolog syntax to represent R expressions (Fig. 2). The match is close, but not 100%. For example, where '10.' is a valid R floating point number, Prolog requires writing this as '10.0' and R identifiers that start with a

```prolog
1  :- include(session_viz).
2  :- <- library("ggplot2").
3  :- <- library("scales").
4
5  %% create_barchart
6  %  exploration of data
7  create_barchart :-
8      x_label(Name),
9      title(Title),
10     r_data_frame(data, [value=Value, count=Count],
11                  aggregate(count, limit(20000, data_count_value(Value)), Count)),
12     data$percentage <- round(data$count/sum(data$count) * 100, digits=2),
13     barchart(data, value, Name, Title).
14
15  barchart(Data, Column, XLab, Title) :-
16      <- ggplot(Data, aes(x=factor(Column), y=count, fill=Column))
17      ) + geom_bar(stat = "identity", size = 0.2
18      ) + xlab(XLab) + ylab("Frequency"
19      ) + ggtitle(Title
20      ) + scale_y_continuous( labels = comma
21      ) + theme(plot.title = element_text(size=16),
22                axis.title=element_text(size=14),
23                axis.text.x = element_text(size = 10, angle = 30, hjust = 1),
24                axis.text.y = element_text(size = 10)
25      ) + guides(fill=false
26      ) + geom_text(colour = "black", size = 5, aes(label=percentage), vjust = -0.3).
27
28  %% create histogram
```

R code in Prolog program

Fig. 2. R expression in a Prolog background program in SWISH DataLab.

---

[5] http://www.graphviz.org/.

[6] https://d3js.org/.

[7] http://c3js.org/.

[8] https://www.rforge.net/Rserve/.

capital letter need single quotes to avoid misinterpretation as a Prolog variable. Complex R objects such as functions cannot be expressed using the Prolog syntax. This problem is resolved using *quasi quotations* [7]. Quasi quotations allow for embedding external languages verbatim, while interpolating values from Prolog. The basic syntax is {|lang(param...)||code|}. For example, we can write $? - \{|\texttt{r}||\texttt{plot}(\texttt{c}(1,2,3))|\}.$ to realize a simple R plot. We can combine this with Prolog as shown:

```
?- numlist(1, 25, Data),
   {|r(Data)||plot(Data)|}.
```

Quasi quotations allow for reusing long snippets of R code verbatim, while the Prolog syntax is more natural for relatively simple R calls and allows for building R calls dynamically.

3   Case Study: Analysis of Online User Search Behavior

The goal of this use case is to understand user search behavior. Under strictest confidentiality agreement we have received six months of server logs from the full text search platform[9], spanning a period from October 2015 to March 2016. This search platform provides access to combined collections from the National Library of the Netherlands and other national heritage and research institutes. These collections are–as is the case for other digital libraries and archives– characterized by bibliographic data describing the content (e.g. publication date, type of document, origin of document). These metadata values are reflected in the search interface in *facets* that can help filter the results (Fig. 3). Over 90% of user requests accesses the historical newspaper collection. For this reason we focus on this collection, that contains over 100M documents across four centuries. Using SWISH DataLab, we have linked the content metadata values to the clicked and downloaded documents, enabling a comparison between facet-use in search and the metadata of clicks and downloads.

3.1   Iterative Data Exploration Through Abstraction

Constructing a vocabulary. The goal to describe user search behavior has motivated an exploration of the server logs based on user interactions within sessions. The sessions are defined based on IP address and a 30 min timeout. In order to recognize usage patterns, we have visualized these sessions in graphs with Graphviz for the rendering (see Fig. 4). As an added benefit, the graphs help to conceptualize the data.

Visualization of abstractions. The graphs make visible that users visit the same search engine results page often in a session, leading to a Prolog rule where we abstract this to a single node with multiple incoming arrows. This abstraction

---

Fig. 3. Search interface of the Delpher platform, with facets on the left and results to the right.

produces a different count of how many search interactions a user has engaged in, a count where revisiting the same results page in the same session is not seen as a new search (Fig. 5).

Removing reloads. We also removed repeated visits of the same web page right after each other, as this is likely a reload of the web browser and not a new interaction by the user. This has resulted in a more practical count of the number of clicks and a clearer definition of the dwell time on a document (as we time this from the first load of the page until a new interaction and not as separate shorter dwell times).

Return to any previous state. These incrementally written rules to clean the data and visualize the graphs can be fully traced in the web environment, where all intermediate saved versions of both data and code have been saved and can be retrieved. Being able to return to an exact previous state improves the reproducibility of the research, making it possible to save the state of the project at the time of publication and to rerun the exact steps on the same (or possibly updated) data.

Evaluate concepts and abstractions. We can inspect the statistical effect of different symbolic definitions of the data, such as shown in Fig. 5, where two abstractions over the data are set aside each other in a single notebook. The environment supports this type of transparency in code and results, making it possible to measure the impact of these abstractions on the results.
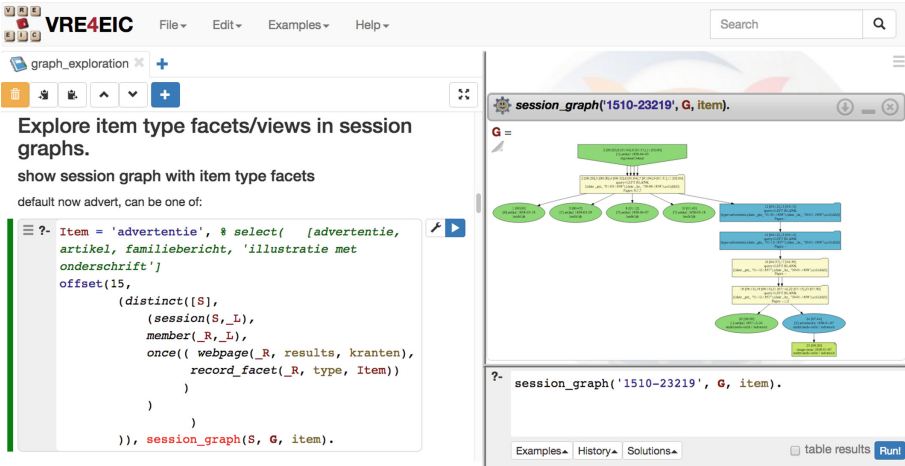
Fig. 4. Visualizing user sessions in graphs. Query code on the left, session graph to the right.
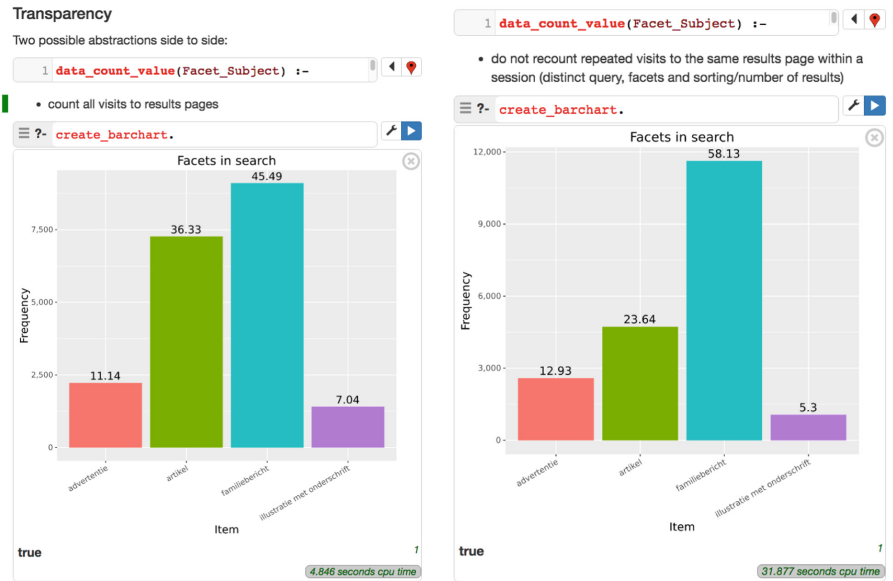

Fig. 5. Side by side analysis of facet use in search interactions, comparing a count of all search interactions versus not recounting revisits within a session.

4   Conclusions and Future Work

With SWISH DataLab we are moving closer to a single environment for responsible data science shared between researchers. The use of visualizations in combination with concepts and abstractions defined as rules makes data cleaning

more transparent and more thorough, and patterns in user interactions more insightful.

Future work on the use case will include a better definition of a session based on the graphs and machine learning (using for example the machine learning algorithms available in R) for prediction of behavior.

SWISH DataLab will be extended with facilities to improve collaboration such as sending change notifications, shared editing and a commenting/chat service. We also plan to facilitate generating a permanent link that captures a result (e.g., a chart or table) and all programs and data needed to reproduce this result reliably.

References

1. Angelopoulos, N., Santos Costa, V., Azevedo, J., Wielemaker, J., Camacho, R., Wessels, L.: Integrative functional statistics in logic programming. In: Sagonas, K. (ed.) PADL 2013. LNCS, vol. 7752, pp. 190–205. Springer, Heidelberg (2013). doi:10. 1007/978-3-642-45284-0_13
2. MacIntyre, D.A., Chandiramani, M., Lee, Y.S., Kindinger, L., Smith, A., Angelopoulos, N., Lehne, B., Arulkumaran, S., Brown, R., Teoh, T.G., Holmes, E., Nicoholson, J.K., Marchesi, J.R., Bennett, P.R.: The vaginal microbiome during pregnancy and the postpartum period in a European population. Sci. Rep. **5**, 8988 (2015). EP
3. Andreasen, T., Christiansen, H., Have, C.T.: Querying sentiment development over time. In: Larsen, H.L., Martin-Bautista, M.J., Vila, M.A., Andreasen, T., Christiansen, H. (eds.) FQAS 2013. LNCS (LNAI), vol. 8132, pp. 613–624. Springer, Heidelberg (2013). doi:10.1007/978-3-642-40769-7_53
4. Wielemaker, J., Lager, T., Riguzzi, F.: SWISH: SWI-Prolog for sharing. CoRR abs/1511.00915 (2015)
5. Beek, W., Wielemaker, J.: SWISH: an integrated semantic web notebook. In: Kawamura, T., Paulheim, H., (eds.) Proceedings of the ISWC 2016 Posters & Demonstrations Track co-located with 15th International Semantic Web Conference (ISWC 2016). CEUR Workshop Proceedings, Kobe, Japan, 19 October 2016, vol. 1690. CEUR-WS.org (2016)
6. Ragan-Kelley, M., Perez, F., Granger, B., Kluyver, T., Ivanov, P., Frederic, J., Bussonier, M.: The Jupyter/iPython architecture: a unied view of computational research, from interactive exploration to communication and publication. In: AGU Fall Meeting Abstracts, vol. 1, p. 07 (2014)
7. Wielemaker, J., Hendricks, M.: Why it's nice to be quoted: Quasiquoting for Prolog. CoRR abs/1308.3941 (2013)