PhUSE 2017

Paper DV08

Looking at SAS code as data

Christof Binder, F. Hoffmann-La Roche, Basel, Switzerland

ABSTRACT

Working with a large code base can be or a complex macro suite can be a daunting task. In this paper methods are presented to look at such a code base from a different angle. Interpreting the code as data offers advantages over the traditional 'comprehend by looking over code' approach. Tools to visualise data have been developed over the last years and they can be applied to a code base. This makes highly connected modules stand out and shows the overall structure of the code in a much more approachable manor. This helps identify on the one hand key macros that are called from a lot of other macros, on the other hand macros that are not called at all and may be redundant.

INTRODUCTION

Maintaining and enhancing a large macro system in SAS is a difficult task. Traditionally doing so was depending on documentation provided by the original developers such as design documents, functional specification, technical documentation, information in the macro header and comments in the macro code itself. While this information is essential for a large system it can be a) non-existing or b) not kept up-to-date. So puring over the actual code is a task that most developers are very familiar with.

The paper presents a different approach, by treating the code base of a fictional tool as data. We can analyse the code-data as any other data, but also can use more interesting an interactive approaches to show connectivity, hot-spots and orphan parts in the code. This can guide the developer to areas in the code that are worth a closer look.

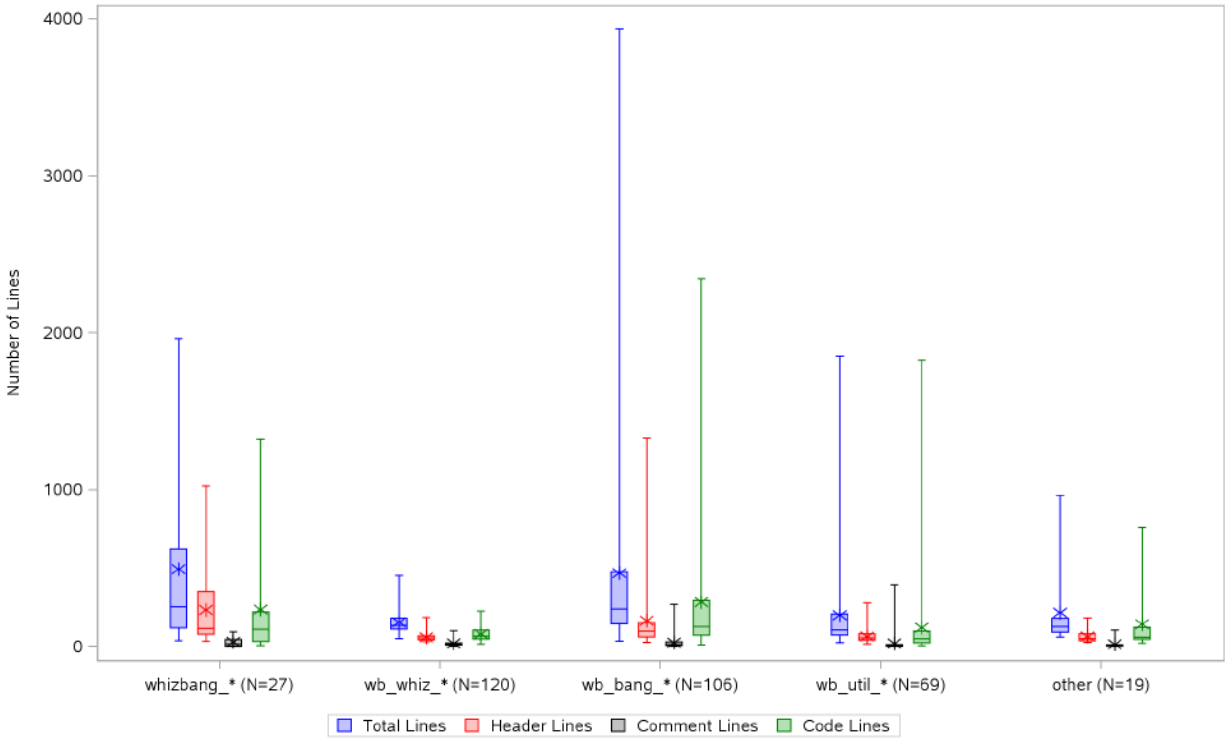
THE SYSTEM

The Whiz-bang system used in this paper is a fictional one, but the complexity of it is real. The tool consists of two main parts. The whiz part and the bang part. There is a naming convention that is used in the code base:

Macro Name	Meaning
whizbang_*	End-user macro
wb_whiz_*	Macros relating to the whiz-part
wb_bang_*	Macros relating to the bang-part
wb_util_*	Utility macros

BASIC STATISTICS

Looking at the code base and extracting some information from it lead to these basic statistics:



A FEW POINTS TO NOTICE.

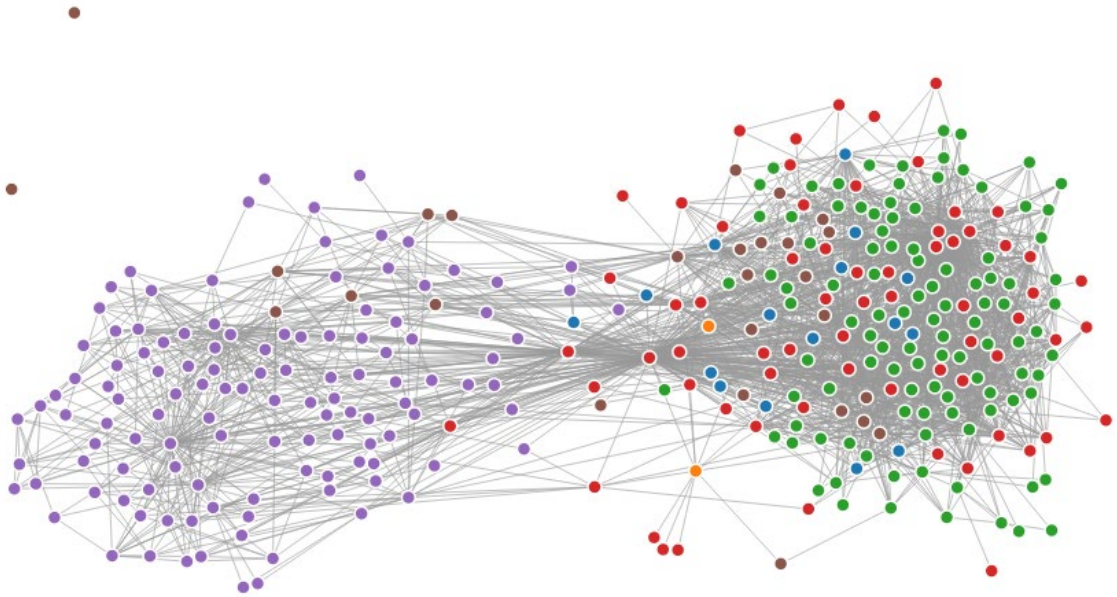
The header of the macros is roughly taking up a third of all the macro. So the header documentation seems to be fairly elaborate. Some of the macros have a very short header, so a look to see why that is so might be worthwhile.

The whiz-part is generally consisting of shorter macros. This might be due to a more modular approach or a smaller complexity of the task at hand

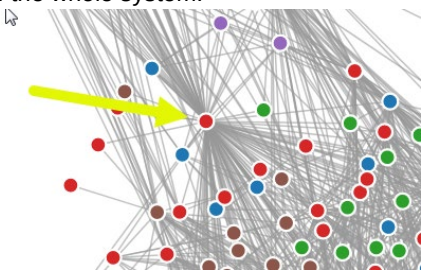
There are macros that have no additional comment in the body of the macro at all. That sounds a bit sketchy and certainly worth a closer look.

CONNECTIVITY

Looking at static data only tells you so much. Fortunately there have been tool developed to show connected data in a visually interesting manner. One of these tool is the JavaScript library D3 developed by Mike Bostock. With a script that extracts the chain of macro calls and creates a json file this data can be looked at on a web-page.



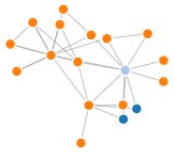
In the interactive version it is easy to hover over a dot that represents a macro to find out it's name and you can also pull a dot out to find all the connections this particular macro has.
This makes it easy to see that a macro called by a lot of other macros and therefore is central for the understanding of the whole system.



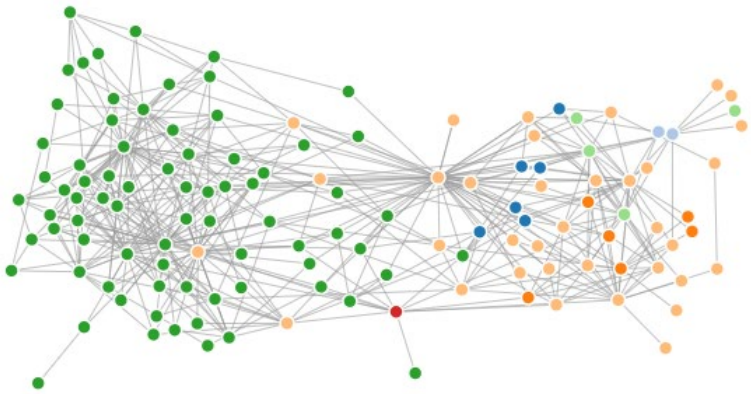
Also the two completely unlinked macros are either obsolete or serve a very specific purpose that is not related to the normal functioning of the system. Like clean-up task or setting up of metadata

ZOOMING IN

The whole graph is rather big and complicated. To get a better picture of what is happening in one special point of interest the graph can be reduced to only show the macros that are reached from one starting macro.
This can be a small utility macro resulting in a small graph



Or one of the main macros resulting in a bigger one.



CONCLUSION

Looking at a code base with different tools than just a text editor can help getting an overview and focus the attention to the most relevant part of the code base. The usage of different tools can be very beneficial, esp. when inheriting a big code base.
Identifying possible hot spots in the code can save a lot of time looking for them in a big code base, and the opposite can be true too: spending a lot of time trying to understand a macro that is not used at all might not be the best investment of time.
The tools presented can not replace looking and comprehending the SAS code in detail, but they can help to understand and visualize the relationship and complexity of the task at hand.

REFERENCES

The D3.js library can be found at https://d3js.org/

CONTACT INFORMATION

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

Christof Binder
F. Hoffmann-La Roche, Ldt
email: christof.binder@roche.com