SESUG 2025 - Paper ####

Dot my SAS program and show me the GRAPHVIZ in VS Code

Ahmed Al-Attar, AnA Data Warehousing Consulting, LLC

ABSTRACT

This paper introduces the PROC SCAPROC procedure, otherwise known as the SAS Code Analyzer procedure, that was first introduced in Release 9.2 of Base SAS® Software back in 2008. SAS Institute's R&D developers later in 2009, integrated it into SAS® Data Integration Studio 4.2 via the "SAS Code Analyzer" tool, and in 2010 they integrated it into SAS® Enterprise Guide® 4.3 via the "Analyze SAS Program" feature. Despite being introduced 16+ years ago, and multiple SAS papers showcasing the procedure, not many SAS users are aware of this procedure and its capabilities. It's time to change this.

With the rise of Visual Code Studio (VS Code) usage within the SAS ecosystem, where SAS Institute has created an extension for SAS coders back in 2022, and SAS® Viya® Workbench providing SAS source-code editor based on VS Code, this paper will illustrate how to use custom developed SAS macro programs to analyze SAS program(s), automatically generate diagram(s) based on the DOT language of the open source GRAPHVIZ visualization software, use extension within VS Code to manipulate the generated *.dot files export/save it as *.svg files that could be added to final user documentation.

INTRODUCTION

The SCAPROC procedure uses the SAS Code Analyzer, which captures information about input, output, and the use of macro symbols from a SAS program while it is running. The SAS Code Analyzer can write this information and the information in the original SAS file to a file that you specify. It's relatively easy to read this information with SAS and generate a diagram showing how that program runs. We will look at the following steps:

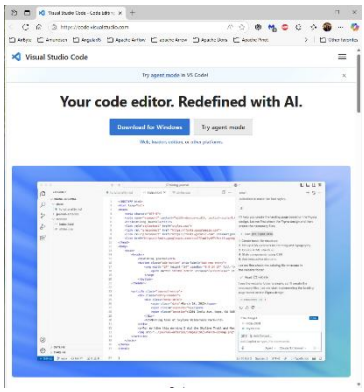
1. Generating information about the flow of a program
2. Reading that into SAS using custom developed macros
3. Defining how to produce a diagram
4. Making the diagram
5. Use Microsoft Visual Studio Code (Vs Code) to manipulate the diagram

In my latest project, I needed to produce documentation about the data used in my SAS programs and that's when I used the contents from a SAS Global Forum 2017 paper by Philip Mason titled "Automatically create diagrams showing the structure and performance of your SAS code" as the main foundation for the work being presented today.

I've added/modified the original SAS code to implement the features and capabilities presented in the code I'm sharing in my GitHub repository

INSTALLING VISUAL STUDIO CODE (VS CODE)

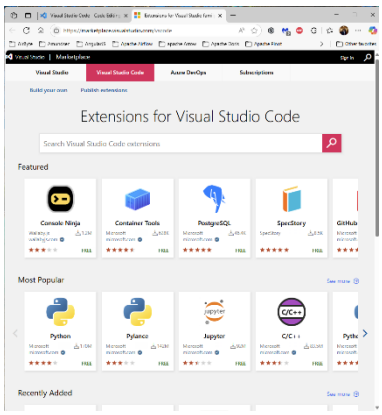
You can download it from here https://code.visualstudio.com/ or you can follow whatever official rout your organization dictates for installing open-source software.

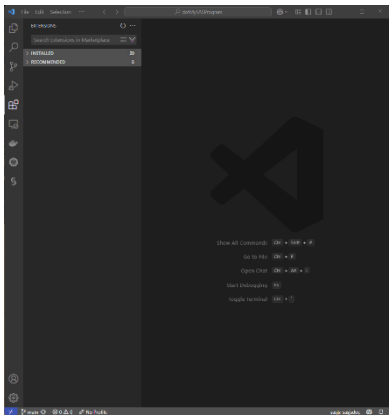


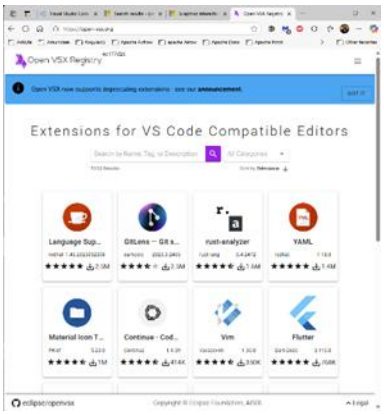
INSTALLING VS CODE EXTENSION

You can install VS Code extensions via two possible ways, Extension Marketplace (https://marketplace.visualstudio.com/vscode) Or from the Extensions viewer within VS Code.

Note: The Open VSX Registry (https://open-vsx.org/) provides the ability to physically download the *.vsix file to your machine to allow for “Installing form the VSIX...” within VS Code. Useful when you have VS Code installed on host without external internet connection!



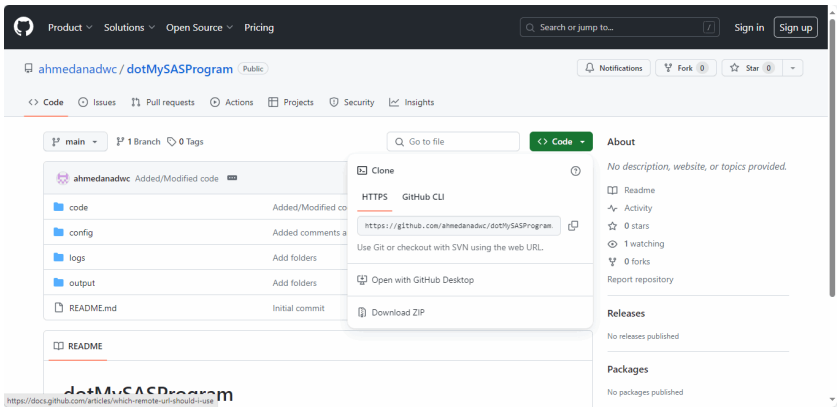




You’ll need to search and install Graphviz Interactive Preview
This extension provides syntax highlighting, snippets, and an interactive, zoom-, pan- and searchable, live preview with edge tracing for graphs in Graphviz / dot format.

DOWNLOAD THE CODE FROM GITHUB

Using your browser, navigate to this Github repository https://github.com/ahmedanadwc/dotMySASProgram



Press the Green [<> Code] button, and select the “Download ZIP” option

WORKING WITH A LOCAL COPY OF THE REPOSITORY

Having downloaded a zipped version of the repository, you'll need to unzip it into a directory/folder of your choice.

Note:

- Everything is consolidated into a single directory, with its sub-directories.
- Out of the box pre-developed custom SAS programs and SAS Macro programs to streamline data acquisition, and manipulation.

```
[drive:]/<Installation directory/folder>
│   package.json
│   README.md
├───code
│   ├───macros
│   │       util_initscaproc.sas
│   │       util_parsescaproc.sas
│   │       util_scatext2graphvizdot.sas
│   │       util_termscaproc.sas
│   └───programs
│           usage_util_parsescaproc.sas
│           usage_util_scatext2graphvizdot.sas
│           usage_util_xxxxscaproc.sas
├───config
│       setup.sas
├───docs
│   └───code
│       └───search
├───logs
│       readme.txt
├───output
│       readme.txt
└───sasjs
    │   sasjsconfig.json
    └───doxy
```

Note: Some sub-directories, such as /docs/code and /sasjs/doxy, contents were omitted/deleted to preserve visual space.

Check out the following link for SAS code documentation
https://ahmedanadwc.github.io/dotMySASProgram/docs/code/files.html

UPDATE CONFIG/SETUP.SAS

The config/setup.sas file contains various SAS statements that are supposed to be executed automatically whenever a SAS session is started, to customize the SAS session and initialize required application settings. You can integrate the contents of this file into your existing autoexec.sas file if you have one, or by simply using %include statement at the top of your existing program(s).

```
/* Get the latest list of available tables under the Census Data API */
%LET g_projRootPath = <Put your value here>;  *<---- Specify installation path. Do not include trailing slash!;
```

UPDATE YOUR PROGRAMS

Update your SAS program(s) before running/submitting (it/them) by the adding

%util_initScaproc macro call at the beginning of the code

%util_termScaproc maco call at the end of the code

```
%include "<InstallationFolder>/config/setup.sas";

/* Initialize the collection and open the output file */
%util_initScaproc(p_enableFlagName=g_enableScaProc, p_outFilePath=%str(&g_logsRoot),
p_outFileName=sca_sesug2025.txt)

proc summary data=sashelp.prdsale nway;
   class year quarter month country region division;
   var   actual predict;
   output out=work.prdsal_agg(drop=_:) sum=;
run;

/* Terminate the collection and close the output file */
%util_termScaproc(p_enableFlagName=g_enableScaProc)
```

Look at usage_util_xxxxscaproc.sas as an example program.

And the generated SAS Code Analyzer output stored in the sca_sesug2025.txt would look like this

```
/* JOBSPLIT: JOBSTARTTIME 29MAY2025:22:05:53.55 */
/* JOBSPLIT: TASKSTARTTIME 29MAY2025:22:05:53.55 */
/* JOBSPLIT: DATASET INPUT SEQ #C00003.PRDSALE.DATA */
/* JOBSPLIT: LIBNAME #C00003 V9 '/pbr/sfw/sas/940/SASFoundation/9.4/sashelp' */
/* JOBSPLIT: CONCATMEM #C00003 SASHELP */
/* JOBSPLIT: LIBNAME SASHELP V9 '( '/pbr/sfw/sas/940/SASFoundation/9.4/nls/u8/sascfg' '/pbr/sfw/sas/940/SASFoundation/9.4/nls/u8/sashelp' '/pbr/sfw/sas/940/SASFoundation/9.4/nls/en/sascfg'
'/pbr/sfw/sas/940/SASFoundation/9.4/sashelp' )' */
/* JOBSPLIT: OPENTIME #C00003.PRDSALE.DATA DATE:29MAY2025:22:05:53.55 PHYS:/pbr/sfw/sas/940/SASFoundation/9.4/sashelp/prdsale.sas7bdat SIZE:262144 */
/* JOBSPLIT: DATASET OUTPUT SEQ WORK.PRDSAL_AGG.DATA */
/* JOBSPLIT: LIBNAME WORK V9 '/saswork/SAS_work815D000079EC_odaws01-usw2.oda.sas.com/SAS_work0AFA000079EC_odaws01-usw2.oda.sas.com' */
/* JOBSPLIT: CATALOG INPUT WORK.SASMAC1.UTIL_TERMSCAPROC.MACRO */
/* JOBSPLIT: LIBNAME WORK V9 '/saswork/SAS_work815D000079EC_odaws01-usw2.oda.sas.com/SAS_work0AFA000079EC_odaws01-usw2.oda.sas.com' */
/* JOBSPLIT: FILE INPUT /home/anawarehousing/dotMySASProgram/code/macros/util_termscaproc.sas */
/* JOBSPLIT: CATALOG OUTPUT WORK.SASMAC1.UTIL_TERMSCAPROC.MACRO */
/* JOBSPLIT: LIBNAME WORK V9 '/saswork/SAS_work815D000079EC_odaws01-usw2.oda.sas.com/SAS_work0AFA000079EC_odaws01-usw2.oda.sas.com' */
/* JOBSPLIT: CATALOG OUTPUT WORK.SASMAC1.SYSTEMP.MACRO */
/* JOBSPLIT: LIBNAME WORK V9 '/saswork/SAS_work815D000079EC_odaws01-usw2.oda.sas.com/SAS_work0AFA000079EC_odaws01-usw2.oda.sas.com' */
/* JOBSPLIT: CATALOG UPDATE WORK.SASMAC1.SYSTEMP.MACRO */
/* JOBSPLIT: LIBNAME WORK V9 '/saswork/SAS_work815D000079EC_odaws01-usw2.oda.sas.com/SAS_work0AFA000079EC_odaws01-usw2.oda.sas.com' */
/* JOBSPLIT: FILE OUTPUT /home/anawarehousing/dotMySASProgram/logs/sca_sesug2025.txt */
/* JOBSPLIT: ATTR #C00003.PRDSALE.DATA INPUT VARIABLE:ACTUAL TYPE:NUMERIC LENGTH:8 LABEL:Actual Sales FORMAT:DOLLAR12.2 INFORMAT: */
/* JOBSPLIT: ATTR #C00003.PRDSALE.DATA INPUT VARIABLE:PREDICT TYPE:NUMERIC LENGTH:8 LABEL:Predicted Sales FORMAT:DOLLAR12.2 INFORMAT: */
/* JOBSPLIT: ATTR #C00003.PRDSALE.DATA INPUT VARIABLE:COUNTRY TYPE:CHARACTER LENGTH:10 LABEL:Country FORMAT:$CHAR10. INFORMAT: */
/* JOBSPLIT: ATTR #C00003.PRDSALE.DATA INPUT VARIABLE:REGION TYPE:CHARACTER LENGTH:10 LABEL:Region FORMAT:$CHAR10. INFORMAT: */
/* JOBSPLIT: ATTR #C00003.PRDSALE.DATA INPUT VARIABLE:DIVISION TYPE:CHARACTER LENGTH:10 LABEL:Division FORMAT:$CHAR10. INFORMAT: */
/* JOBSPLIT: ATTR #C00003.PRDSALE.DATA INPUT VARIABLE:PRODTYPE TYPE:CHARACTER LENGTH:10 LABEL:Product type FORMAT:$CHAR10. INFORMAT: */
/* JOBSPLIT: ATTR #C00003.PRDSALE.DATA INPUT VARIABLE:PRODUCT TYPE:CHARACTER LENGTH:10 LABEL:Product FORMAT:$CHAR10. INFORMAT: */
/* JOBSPLIT: ATTR #C00003.PRDSALE.DATA INPUT VARIABLE:QUARTER TYPE:NUMERIC LENGTH:8 LABEL:Quarter FORMAT:8. INFORMAT: */
/* JOBSPLIT: ATTR #C00003.PRDSALE.DATA INPUT VARIABLE:YEAR TYPE:NUMERIC LENGTH:8 LABEL:Year FORMAT:4. INFORMAT: */
/* JOBSPLIT: ATTR #C00003.PRDSALE.DATA INPUT VARIABLE:MONTH TYPE:NUMERIC LENGTH:8 LABEL:Month FORMAT:MONNAME3. INFORMAT: */
/* JOBSPLIT: ATTR WORK.PRDSAL_AGG.DATA OUTPUT VARIABLE:YEAR TYPE:NUMERIC LENGTH:8 LABEL:Year FORMAT:4. INFORMAT: */
/* JOBSPLIT: ATTR WORK.PRDSAL_AGG.DATA OUTPUT VARIABLE:QUARTER TYPE:NUMERIC LENGTH:8 LABEL:Quarter FORMAT:8. INFORMAT: */
/* JOBSPLIT: ATTR WORK.PRDSAL_AGG.DATA OUTPUT VARIABLE:MONTH TYPE:NUMERIC LENGTH:8 LABEL:Month FORMAT:MONNAME3. INFORMAT: */
/* JOBSPLIT: ATTR WORK.PRDSAL_AGG.DATA OUTPUT VARIABLE:COUNTRY TYPE:CHARACTER LENGTH:10 LABEL:Country FORMAT:$CHAR10. INFORMAT: */
/* JOBSPLIT: ATTR WORK.PRDSAL_AGG.DATA OUTPUT VARIABLE:REGION TYPE:CHARACTER LENGTH:10 LABEL:Region FORMAT:$CHAR10. INFORMAT: */
/* JOBSPLIT: ATTR WORK.PRDSAL_AGG.DATA OUTPUT VARIABLE:DIVISION TYPE:CHARACTER LENGTH:10 LABEL:Division FORMAT:$CHAR10. INFORMAT: */
/* JOBSPLIT: ATTR WORK.PRDSAL_AGG.DATA OUTPUT VARIABLE:ACTUAL TYPE:NUMERIC LENGTH:8 LABEL:Actual Sales FORMAT:DOLLAR12.2 INFORMAT: */
/* JOBSPLIT: ATTR WORK.PRDSAL_AGG.DATA OUTPUT VARIABLE:PREDICT TYPE:NUMERIC LENGTH:8 LABEL:Predicted Sales FORMAT:DOLLAR12.2 INFORMAT: */
/* JOBSPLIT: SYMBOL GET SYSSUMSTACKODS */
/* JOBSPLIT: SYMBOL GET SYSSUMTRACE */
/* JOBSPLIT: SYMBOL GET SYSZSQLCAS */
/* JOBSPLIT: SYMBOL SET P_ENABLEFLAGNAME */
/* JOBSPLIT: SYMBOL GET P_ENABLEFLAGNAME */
/* JOBSPLIT: SYMBOL GET G_ENABLESCAPROC */
/* JOBSPLIT: ELAPSED 17  */
/* JOBSPLIT: SYSSCP LIN X64 */
/* JOBSPLIT: PROCNAME SUMMARY */
/* JOBSPLIT: STEP SOURCE FOLLOWS */

proc summary data=sashelp.prdsale nway;
  class year quarter month country region division;
  var   actual predict;
  output out=work.prdsal_agg(drop=_:) sum=;
run;

/* Terminate the collection and close the output file */
 * Write out the collected information to the record file;

/* JOBSPLIT: JOBENDTIME 29MAY2025:22:05:53.57 */
/* JOBSPLIT: END */
```

PARSE THE SCA_XXXX.TXT FILE(S)

Execute the %util_parseScaproc macro to analyze/parse the SAS Code Analyzer output stored in the **sca_sesug2025.txt** file

```
/* Illustrate how to use the util_parseScaproc macro */
%util_parseScaproc(p_inScaprocFileName=%str(&g_logsRoot/sca_sesug2025.txt)
, p_outDotFilePath=%str(&g_outputRoot), p_outDotFileName=sesug2025.dot
, p_outDsName=work.sesug2025, p_includeAttrs_yn=Y, p_includeSteps_yn=Y, p_rankDir=LR)
```

The generated **sesug2025.dot** file will contain the diagram definition expressed in the DOT language. Opening this file with VS Code would automatically activate the Graphviz DOT extension and provide an interactive preview of the diagram.

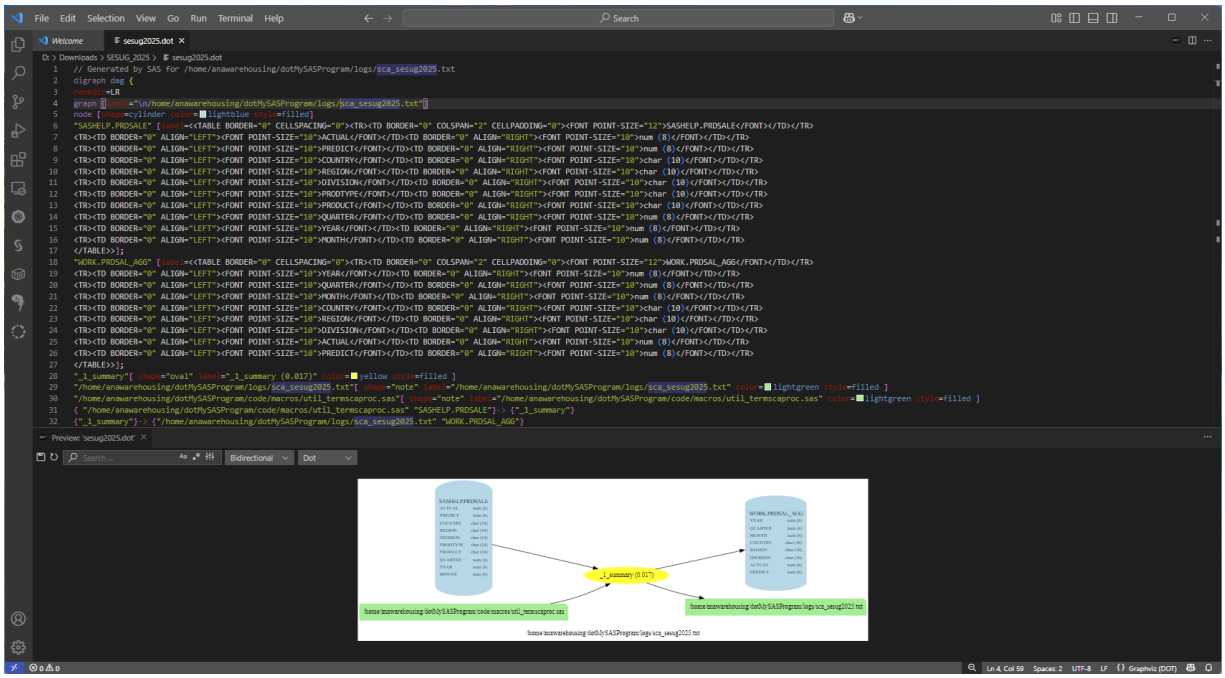


Figure 1. How to render the diagram by using the DOT extension within VS Code.

The generated diagram may not be exactly what you want, but it could be 80% or more of the final desired diagram. Well, that's where VS Code and the DOT extension come handy. You can edit and refine the diagram definition till you get what you are looking for.

In the above diagram, I'm only interested in keeping the Input & Output data set names and the proc summary statement. I want to rename the "_1_summary" step to "Proc Summary" and remove the path of the source sca_sesug2025.txt file. Figure 2 below shows what the sesug2025.dot would look like.

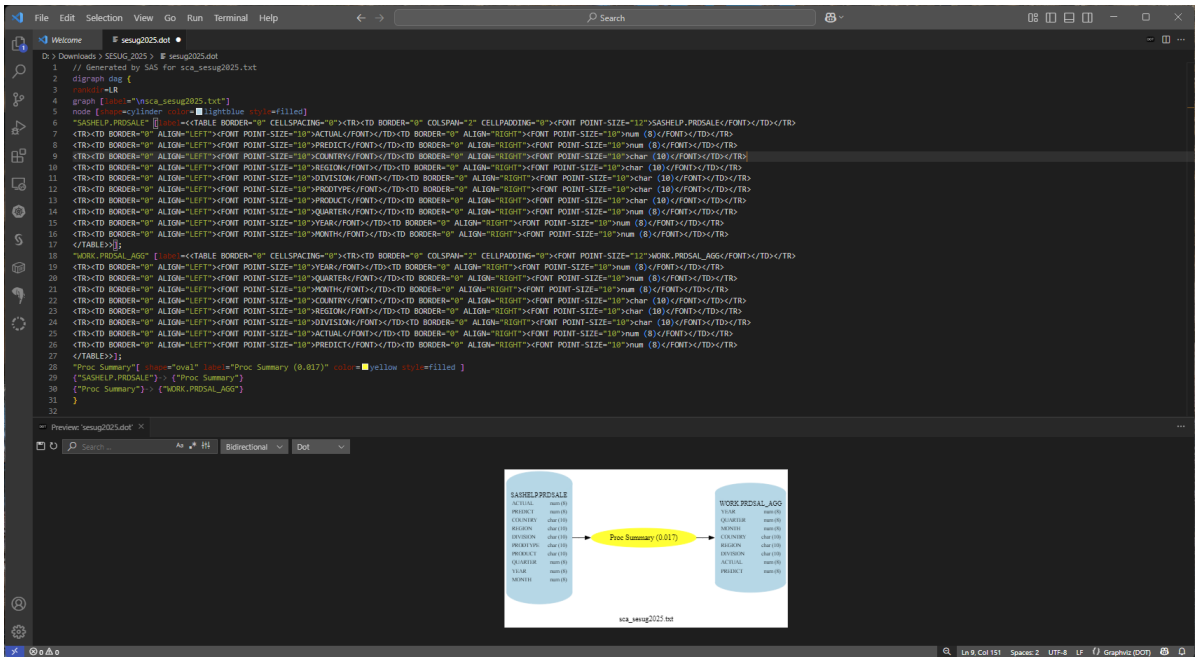


Figure 2. Edited/Refined sesug2025.dot file

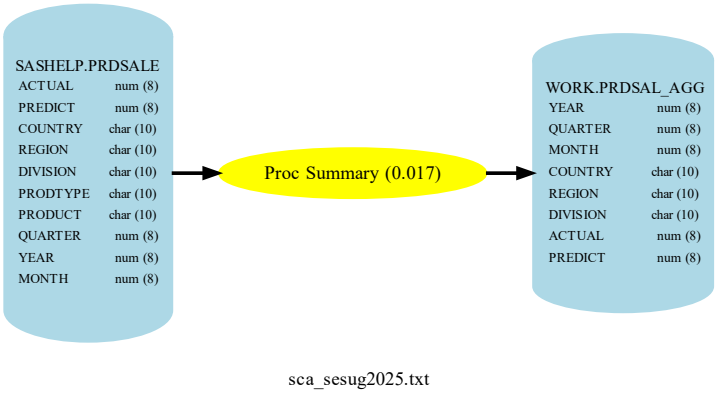


Figure 3. Exported SVG (Scalable Vector Graphics) image by the DOT extension

Specifying "**p_includeAttrs_yn=Y**" in the macro call, causes the diagram to include the data set's columns/variables definitions. But it you specify "**p_includeAttrs_yn=N**", it will only display the data set's name(s)

```
/* Illustrate how to use the util_parseScaproc macro */
%util_parseScaproc(p_inScaprocFileName=%str(&g_logsRoot/sca_sesug2025.txt)
, p_outDotFilePath=%str(&g_outputRoot) , p_outDotFileName=sesug2025_wo.dot
, p_outDsName=work.sesug2025, p_includeAttrs_yn=N, p_includeSteps_yn=Y, p_rankDir=LR)
```

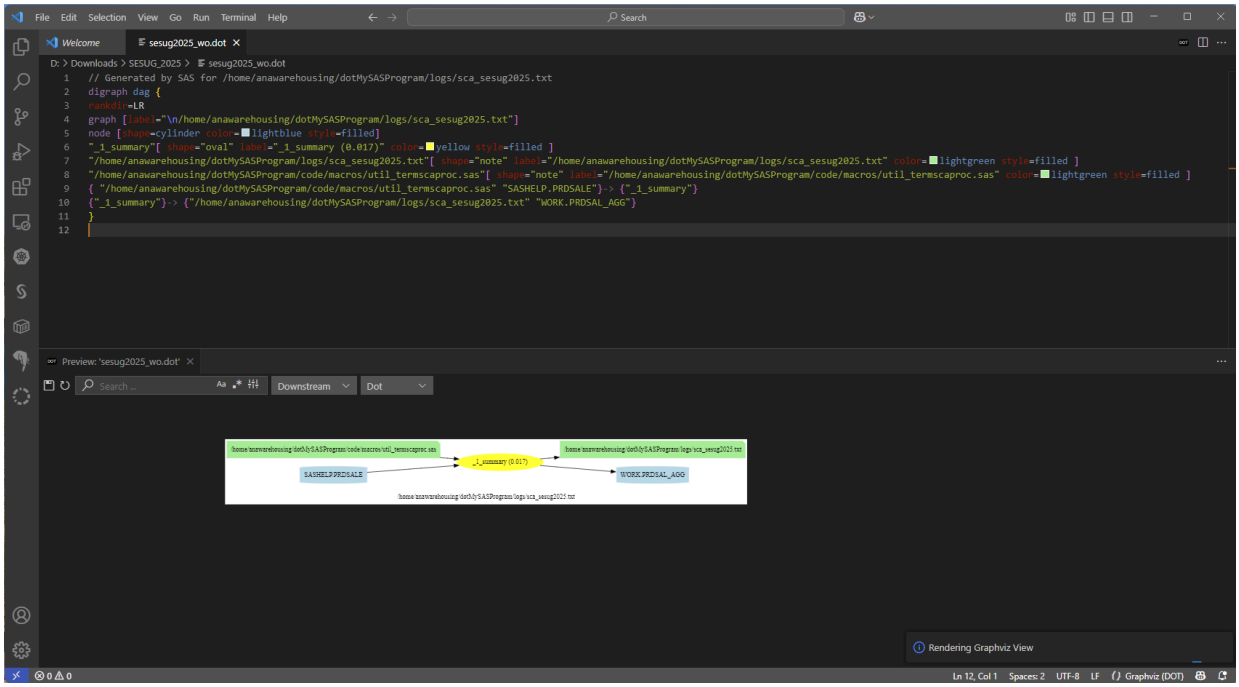


Figure 4. Diagram without data columns in VS Code.

Now, let us apply the same edits and changes as before, see the results

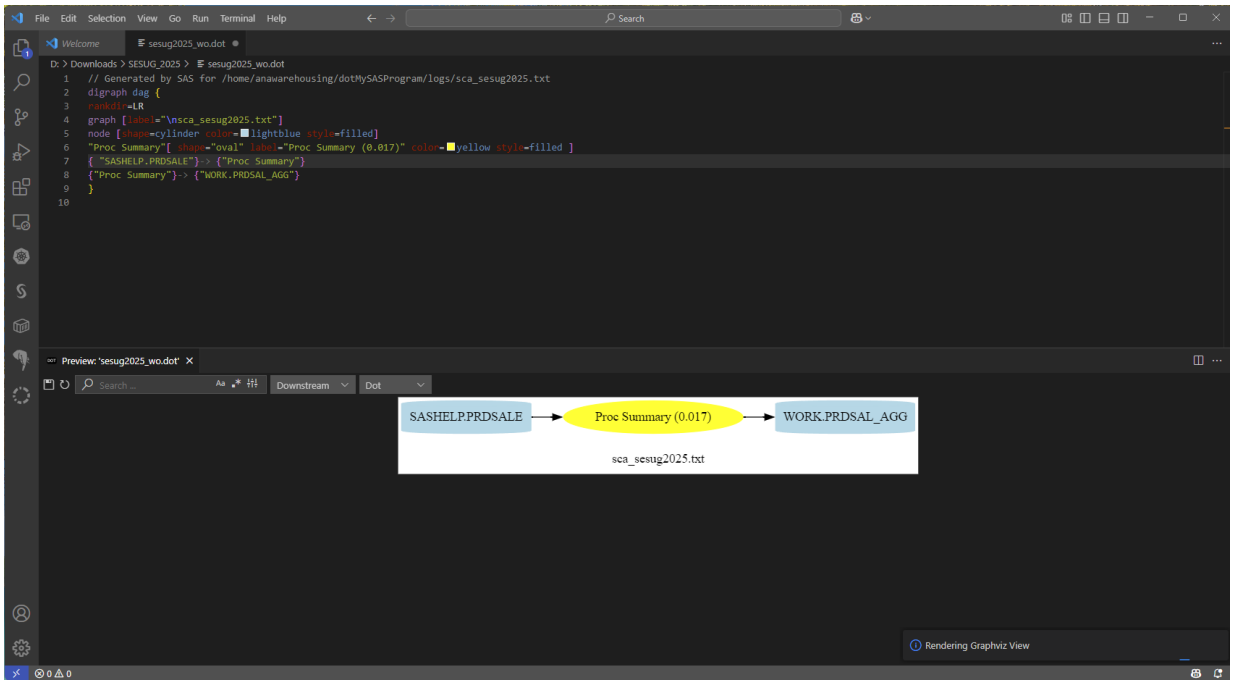


Figure 5. Refined/Edited diagram without data columns in VS Code.

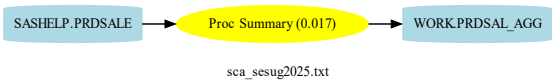


Figure 6. Exported SVG (Scalable Vector Graphics) image by the DOT extension

Having two representations of the same flow/process might be handy for documentation, where you give the reader the option to switch between the two views/representations without cluttering the documentation. This can easily be achieved using the dot language by adding **href="<SomeLink>"** attribute to the **graph** definition within the *.dot file.

```
graph [label="\nsca_sesug2025.txt\nClick here to see tables columns" href="<SomeLink>" ]
```

The GitHub Repository contains a sample program that can parse all the files generated by the SAS Code Analyzer and stored in single directory and generate parsed *.dot files in a destination directory.

This sample program can be found here
https://ahmedanadwc.github.io/dotMySASProgram/docs/code/usage__util__scatext2graphvizdot_8sas.html

CONCLUSION

Combining open-source software and tools (VS Code, Graphviz with its DOT Language and extension) with SAS gives you the ability to enhance/increase your productivity with very little programming and time on multiple operating system platforms.

These custom-developed macros provide primitive/simple diagrams the combines data flow and data lineage information driven from analyzing existing SAS programs running under the classic SAS 9 foundation deployment.

REFERENCES

Al-Attar, Ahmed. “dotMySASProgram”. Available at https://github.com/ahmedanadwc/dotMySASProgram.git

Mason, Philip. 2017 “Automatically create diagrams showing the structure and performance of your SAS code“. SAS Global Forum 2017. Wood Street Consultants Ltd, England

Available at https://support.sas.com/resources/papers/proceedings17/1104-2017.pdf

Mason, Philip. 2023 “Documenting your SAS programs with Doxygen and automatically generated diagrams.” PharmaSUG 2023. Wood Street Consultants Ltd, England

Available at Documenting your SAS programs with Doxygen and automatically generated diagrams.

Nawrocki, Bruce. 2024 “Using Visual Studio Code to Write SAS Code - A Primer”. SESUG 2024.

NC Dept of Health and Human Services.

Available at https://www.lexjansen.com/sesug/2024/109_Final_PDF.pdf