

Automatically create diagrams showing the structure and performance of your SAS® code

Philip Mason, Wood Street Consultants Ltd, England

ABSTRACT

The enhanced analysis script allows users to analyse their SAS® programs, and display that program visually. It then overlays an extra level of insight on performance measures within that script, allowing the user to identify potential bottlenecks and areas for additional coding efficiency. The following document explains how this code works in more detail, and explains how to use it within your own programs.

INTRODUCTION

PROC SCAPROC has been available in SAS since SAS 9.2. It implements the SAS Code Analyzer which provides information about input tables, output tables and macro symbols in a SAS program. Its easy to read this information in with SAS, which gives you a nice view of how the program runs. Having that information means it is possible to 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
3. Defining how to produce a diagram
4. Making the diagram

HOW IT'S DONE

PROC SCAPROC

You can read all about this procedure, but I will tell you the part you need to know for this process to work. The way it works is that you start it running to record the activity of your SAS program, and then when your program has finished you use PROC SCAPROC to write that information out. So a simple use of it would be like this...

```
proc scaproc ;
    record "c:\users\phil\scaproc.txt" ;
run ;

data x ;
    set sashelp.prdsale sashelp.prdsal2 ;
run ;

proc scaproc ;
    write ;
run ;
```

and the output produced is like this ...

```
/* JOBSPLIT: JOBSTARTTIME 27JAN2017:17:07:04.57 */
/* JOBSPLIT: TASKSTARTTIME 27JAN2017:17:07:04.57 */
/* JOBSPLIT: DATASET INPUT SEQ #C00001.PRDSALE.DATA */
/* JOBSPLIT: LIBNAME #C00001 V9 'C:\Program Files\SASHome\SASFoundation\9.4\core\sashelp' */
/* JOBSPLIT: CONCATMEM #C00001 SASHELP */
/* JOBSPLIT: LIBNAME SASHELP V9 ( 'C:\Program Files\SASHome\SASFoundation\9.4\nls\en\SASCFG' 'C:\Program Files\SASHome\SASFoundation\9.4\core\sashelp' 'C:\Program Files\SASHome\SASFoundation\9.4\aacomp\sashelp' 'C:\Program Files\SASHome\SASFoundation\9.4\af\sashelp' 'C:\Program Files\SASHome\SASFoundation\9.4\cmp\sashelp' 'C:\Program Files\SASHome\SASFoundation\9.4\connect\sashelp' 'C:\Program Files\SASHome\SASFoundation\9.4\dmne\sashelp' 'C:\Program Files\SASHome\SASFoundation\9.4\ets\sashelp' 'C:\Program Files\SASHome\SASFoundation\9.4\graph\sashelp' 'C:\Program Files\SASHome\SASFoundation\9.4\iml\sashelp' 'C:\Program Files\SASHome\SASFoundation\9.4\inttech\sashelp' 'C:\Program Files\SASHome\SASFoundation\9.4\mdbserv\sashelp' 'C:\Program Files\SASHome\SASFoundation\9.4\or\sashelp' 'C:\Program Files\SASHome\SASFoundation\9.4\spdsclient\sashelp' 'C:\Program Files\SASHome\SASFoundation\9.4\stat\sashelp' )' */
/* JOBSPLIT: DATASET INPUT SEQ #C00001.PRDSAL2.DATA */
/* JOBSPLIT: LIBNAME #C00001 V9 'C:\Program Files\SASHome\SASFoundation\9.4\core\sashelp' */
/* JOBSPLIT: CONCATMEM #C00001 SASHELP */
/* JOBSPLIT: LIBNAME SASHELP V9 ( 'C:\Program Files\SASHome\SASFoundation\9.4\nls\en\SASCFG' 'C:\Program Files\SASHome\SASFoundation\9.4\core\sashelp' 'C:\Program Files\SASHome\SASFoundation\9.4\aacomp\sashelp' 'C:\Program Files\SASHome\SASFoundation\9.4\af\sashelp' 'C:\Program Files\SASHome\SASFoundation\9.4\cmp\sashelp' 'C:\Program Files\SASHome\SASFoundation\9.4\connect\sashelp' 'C:\Program Files\SASHome\SASFoundation\9.4\dmne\sashelp' 'C:\Program Files\SASHome\SASFoundation\9.4\ets\sashelp' 'C:\Program Files\SASHome\SASFoundation\9.4\graph\sashelp' 'C:\Program Files\SASHome\SASFoundation\9.4\iml\sashelp' 'C:\Program Files\SASHome\SASFoundation\9.4\inttech\sashelp' 'C:\Program Files\SASHome\SASFoundation\9.4\mdbserv\sashelp' 'C:\Program Files\SASHome\SASFoundation\9.4\or\sashelp' 'C:\Program Files\SASHome\SASFoundation\9.4\spdsclient\sashelp' 'C:\Program Files\SASHome\SASFoundation\9.4\stat\sashelp' )' */
/* JOBSPLIT: DATASET OUTPUT SEQ WORK.X.DATA */
/* JOBSPLIT: LIBNAME WORK V9 'C:\Users\phil\AppData\Local\Temp\2\SAS Temporary Files\_TD21420_D351T092_' */
/* JOBSPLIT: FILE OUTPUT c:\users\phil\scaproc.txt */
/* JOBSPLIT: ELAPSED 28 */
/* JOBSPLIT: SYSSCP WIN */
/* JOBSPLIT: PROCNAME DATASEP */
/* JOBSPLIT: STEP SOURCE FOLLOWS */

data x ;
    set sashelp.prdsale sashelp.prdsal2 ;
run ;

/* JOBSPLIT: JOBBENDTIME 27JAN2017:17:07:04.60 */
/* JOBSPLIT: END */
```

The key bits of information that we can extract from this text is as follows:

- JOBSPLIT: DATASET INPUT - tells us what tables were read in by the step
- JOBSPLIT: DATASET OUTPUT - tells us what tables were written out by the step
- JOBSPLIT: ELAPSED - tells us the elapsed time taken for the step, in milli-seconds
- JOBSPLIT: PROCNAME - tells us the type of step, e.g. DATASTEP, FREQ, SORT, etc.

There is other info that we can get which can be useful, depending on our requirements, however to make a diagram showing the flow of data in a program really only requires the inputs and outputs from steps. I like to also know what kind of step it was, and how long it took.

READING IN THE OUTPUT OF PROC SCAPROC

To read the information in from the PROC SCAPROC text is quite easy. You just need to scan the lines for the information you need, and to keep it in some variables. I use the scan function to break lines up into words separated by spaces.

The following code looks at words separated by spaces from lines read in. It then works out which are input tables, output tables and what step they belong to. This would then be the bare minimum needed to draw a diagram of the data flow.

```
data scaproc_parsed(keep=in out step) ;
  retain step 1 ;
  infile "c:\users\phil\scaproc.txt" ;
  input ;
  word1=scan(_infile_,1,' ') ;
  word2=scan(_infile_,2,' ') ;
  word3=scan(_infile_,3,' ') ;
  word4=scan(_infile_,4,' ') ;
  word5=scan(_infile_,5,' ') ;
  word6=scan(_infile_,6,' ') ;
  word7=scan(_infile_,7,' ') ;
  if word2='JOBSPLIT:' & word3='DATASET' & word4='INPUT' then in=word6 ;
  if word2='JOBSPLIT:' & word3='DATASET' & word4='OUTPUT' then out=word6 ;
  if word2='JOBSPLIT:' & word3='STEP' then step+1 ;
  if in>' ' or out>' ' then output ;
run ;
```

This program will read the previous SCAPROC text file and result in a SAS table like this ...

	step	in	out
1	1	#C00001.PRDSALE.DATA	
2	1	#C00001.PRDSAL2.DATA	
3	1		WORK.X.DATA

You will notice that instead of SASHELP, I have #C00001 as the libname recorded. There is another line in the SCAPROC text file that tells us what the actual libref is for this, so we can code around this issue although I haven't done so here for simplicity.

What we can see from this data is that step 1 has 2 input tables and 1 output table. With that information we can draw a simple diagram. Or we could use this information to sit down with VISIO and construct a diagram from it. However if we distill this information into a language used to make diagrams, then we can make one automatically.

GRAPHVIZ

Graphviz is open source graph visualisation software that has been around for a very long time. It uses a simple text language as input which is used to construct a diagram. This language is called the DOT language. Here is a simple world graph, followed by the DOT language for a more complex one.

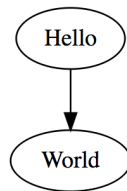
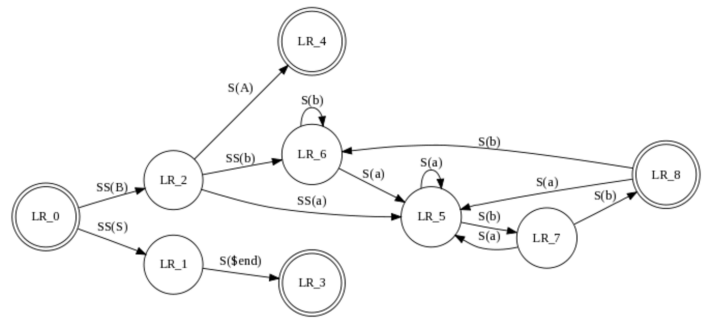
```
digraph G {Hello->World}

digraph finite_state_machine {rankdir=LR;
  size="8,5"
  node [shape = doublecircle];
      LR_0 LR_3 LR_4 LR_8;
  node [shape = circle];
```

```

LR_0 -> LR_2 [ label = "SS(B)" ];
LR_0 -> LR_1 [ label = "SS(S)" ];
LR_1 -> LR_3 [ label = "S($end)" ];
LR_2 -> LR_6 [ label = "SS(b)" ];
LR_2 -> LR_5 [ label = "SS(a)" ];
LR_2 -> LR_4 [ label = "S(A)" ];
LR_5 -> LR_7 [ label = "S(b)" ];
LR_5 -> LR_5 [ label = "S(a)" ];
LR_6 -> LR_6 [ label = "S(b)" ];
LR_6 -> LR_5 [ label = "S(a)" ];
LR_7 -> LR_8 [ label = "S(b)" ];
LR_7 -> LR_5 [ label = "S(a)" ];
LR_8 -> LR_6 [ label = "S(b)" ];
LR_8 -> LR_5 [ label = "S(a)" ];

```



DEFINING A DIAGRAM

We can use the data from our table which was produced after analyzing the PROC SCAPROC text file. That data shows us that step 1 has 2 inputs and 1 output. In DOT language this can be represented like this...

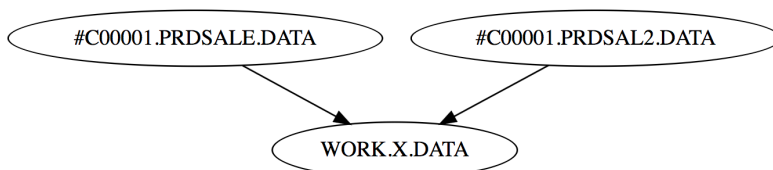
```

digraph {
  "#C00001.PRDSALE.DATA" -> "WORK.X.DATA"
  "#C00001.PRDSAL2.DATA" -> "WORK.X.DATA"
}

```

MAKING A DIAGRAM

Once we have the DOT commands to make a diagram, then we can use that with the GraphViz software to generate it. This can be done in a number of ways. You can install the software on your own machine, and run it locally on that machine. Or if you don't have the software installed you can use a web version of it, such as WebGraphViz, to generate the diagram. So if I copy and paste this DOT language into WebGraphViz, it generates the following diagram...

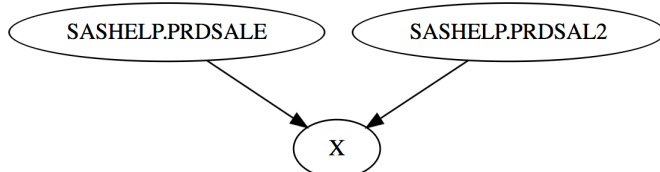


I could, of course clean up the text a bit manually, or enhance my SAS program to do it for me, thus making a diagram like this...

```

digraph {
  "SASHELP.PRDSALE" -> "X"
  "SASHELP.PRDSAL2" -> "X"
}

```



AUTOMATING THE APPROACH

I have made some macros to enable this technique to be used more easily on all kinds of SAS code. The macros are:

- Eanbegin - turns on recording with PROC SCAPROC
- Eanend - turns off recording and writes out to file
- Percentiles - calculates some percentiles based on the run time of steps, so we know which are the longest ones
- Scaproc_analyse - analyses the output from PROC SCAPROC and produces DOT language that is used by Graphviz to make a diagram.

TO USE THIS IN YOUR OWN SAS PROGRAM

- 1) You need to have the macros accessible either via an autocall library or just run them first. Just copy them from this document and run in your SAS session.
- 2) Turn on Enhanced Logging:
%let _eandebbug=scaproc ;
- 3) Put the %EANBEGIN macro before your code.
- 4) Put the %EANEND macro after your code.
- 5) Run those macros with your code.
- 6) Copy the lines from the table just created (GRAPHVIZ).
 - a) View the table.
 - b) Click on column, to select all values in it
 - c) Then control-C.
- 7) Go to <http://webgraphviz.com>
 - a) Paste the lines into the Text Area.
 - b) Press "Generate Graph!" button.
- 8) Now you will have your diagram displayed in your web browser. You can copy it into documentation, or you could save it as a SVG (small and scalable), PNG (small and not scalable) or BMP (big and not scalable). I would suggest SVG as the file size is small and you can scale it as big or small as you want to without losing quality.

EXAMPLE SHOWING HOW WE CAN MAKE A DIAGRAM

The statements highlighted in yellow are the ones which are used to create the information needed to make a flow diagram of your SAS code.

```
%* set macro variable to turn on SCAPROC and set verbose logging ;
%let _eandebbug=scaproc,verbose;

%* Start recording SCAPROC data ;
%eanbegin(Sample 1)

*****;
*** This is the sample program we will measure and then make a flow chart ;
*****;

data x ;
    set sashelp.class ;
run ;

data y ;
    set sashelp.class ;
run ;

proc summary data=x ;
    class sex ;
    var height ;
    output out=x2 mean= ;
```

```

run ;

proc summary data=y ;
  class sex ;
  var height ;
  output out=y2 mean= ;
run ;

proc sort data=x2 out=x3 ;
  by sex ;
run ;

proc sort data=y2 out=y3 ;
  by sex ;
run ;

data z ;
  merge x3 y3 ;
  by sex ;
run ;

proc print ;
run ;

proc sql ;
  create table sql_table as
  select *
  from x
  left join
  y
  on x.sex=y.sex ;
quit ;

*****;
*** finish of sample program ;
*****;

%* Finish recording SCAPROC data and write it out ;
%eanend

%* Generate the graphViz dot language to be used to make diagram ;
%scaproc_analyse

```

OUTPUT PRODUCED BY THE EXAMPLE ABOVE

The code above produces a file containing the SCAPROC data, which looks like this (not all rows shown).

```

/* JOBSPLIT: JOBSTARTTIME 18JAN2017:08:17:56.55 */
/* JOBSPLIT: TASKSTARTTIME 18JAN2017:08:17:56.59 */
/* JOBSPLIT: DATASET INPUT SEQ #C00002.CLASS.DATA */
/* JOBSPLIT: LIBNAME #C00002 V9 '/config/sas_9.4/sas_9.4/RTL/SASFoundation/9.4/sashelp' */
/* JOBSPLIT: CONCATMEM #C00002 SASHELP */
/* JOBSPLIT: LIBNAME SASHELP V9 '( '/config/sas_9.4/sas_9.4/RTL/SASFoundation/9.4/nls/en/sascfg'
'/config/sas_9.4/sas_9.4/RTL/SASFoundation/9.4/nls/en/sascfg'
'/config/sas_9.4/sas_9.4/RTL/SASFoundation/9.4/sashelp' )' */
/* JOBSPLIT: OPENTIME #C00002.CLASS.DATA DATE:18JAN2017:08:17:56.59
PHYS:/config/sas_9.4/sas_9.4/RTL/SASFoundation/9.4/sashelp/class
.sas7bdat SIZE:131072 */
/* JOBSPLIT: DATASET OUTPUT SEQ WORK.X.DATA */
/* JOBSPLIT: LIBNAME WORK V9
'/temp/work1/sas_9.4/SAS_work873E02930064_p24107prw310/SAS_work870802930064_p24107prw310' */
/* JOBSPLIT: ATTR #C00002.CLASS.DATA INPUT VARIABLE:Name TYPE:CHARACTER LENGTH:8 LABEL: FORMAT: INFORMAT: */
/* JOBSPLIT: ATTR #C00002.CLASS.DATA INPUT VARIABLE:Sex TYPE:CHARACTER LENGTH:1 LABEL: FORMAT: INFORMAT: */

```

```

/* JOBSPLIT: ATTR #C00002.CLASS.DATA INPUT VARIABLE:Age TYPE:NUMERIC LENGTH:8 LABEL: FORMAT: INFORMAT: */
/* JOBSPLIT: ATTR #C00002.CLASS.DATA INPUT VARIABLE:Height TYPE:NUMERIC LENGTH:8 LABEL: FORMAT: INFORMAT: */
/* JOBSPLIT: ATTR #C00002.CLASS.DATA INPUT VARIABLE:Weight TYPE:NUMERIC LENGTH:8 LABEL: FORMAT: INFORMAT: */
/* JOBSPLIT: ATTR WORK.X.DATA OUTPUT VARIABLE:Name TYPE:CHARACTER LENGTH:8 LABEL: FORMAT: INFORMAT: */
/* JOBSPLIT: ATTR WORK.X.DATA OUTPUT VARIABLE:Sex TYPE:CHARACTER LENGTH:1 LABEL: FORMAT: INFORMAT: */
/* JOBSPLIT: ATTR WORK.X.DATA OUTPUT VARIABLE:Age TYPE:NUMERIC LENGTH:8 LABEL: FORMAT: INFORMAT: */
/* JOBSPLIT: ATTR WORK.X.DATA OUTPUT VARIABLE:Height TYPE:NUMERIC LENGTH:8 LABEL: FORMAT: INFORMAT: */
/* JOBSPLIT: ATTR WORK.X.DATA OUTPUT VARIABLE:Weight TYPE:NUMERIC LENGTH:8 LABEL: FORMAT: INFORMAT: */
/* JOBSPLIT: SYMBOL GET SCAPROC_LABEL */
/* JOBSPLIT: SYMBOL GET USE_LABEL */
/* JOBSPLIT: SYMBOL SET SCAPROC_LABEL */
/* JOBSPLIT: SYMBOL GET _EANDBUG */
/* JOBSPLIT: SYMBOL GET _I */
/* JOBSPLIT: SYMBOL SET PARM */
/* JOBSPLIT: SYMBOL GET PARM */
/* JOBSPLIT: SYMBOL SET I */
/* JOBSPLIT: ELAPSED 117 */
/* JOBSPLIT: SYSSCP AIX 64 */
/* JOBSPLIT: PROCNAME DATASTEP */
/* JOBSPLIT: STEP SOURCE FOLLOWS */

*****;
*** This is the sample program we will measure and then make a flow chart ;
*****;

data x ;
    set sashelp.class ;
run ;

/* JOBSPLIT: TASKSTARTTIME 18JAN2017:08:17:56.70 */
/* JOBSPLIT: DATASET INPUT SEQ #C00002.CLASS.DATA */
/* JOBSPLIT: LIBNAME #C00002 V9 '/config/sas_9.4/sas_9.4/RTL/SASFoundation/9.4/sashelp' */
/* JOBSPLIT: CONCATMEM #C00002 SASHELP */
/* JOBSPLIT: LIBNAME SASHELP V9 '( '/config/sas_9.4/sas_9.4/RTL/SASFoundation/9.4/nls/en/sascfg'
'/config/sas_9.4/sas_9.4/RTL/SASFoundation/9.4/nls/en/sascfg'
'/config/sas_9.4/sas_9.4/RTL/SASFoundation/9.4/sashelp' )' */
/* JOBSPLIT: OPENTIME #C00002.CLASS.DATA DATE:18JAN2017:08:17:56.71
PHYS:/config/sas_9.4/sas_9.4/RTL/SASFoundation/9.4/sashelp/class.sas7bdat SIZE:131072 */
/* JOBSPLIT: DATASET OUTPUT SEQ WORK.Y.DATA */
/* JOBSPLIT: LIBNAME WORK V9
'/temp/work1/sas_9.4/SAS_work873E02930064_p24107prw310/SAS_work870802930064_p24107prw310' */
/* JOBSPLIT: ATTR #C00002.CLASS.DATA INPUT VARIABLE:Name TYPE:CHARACTER LENGTH:8 LABEL: FORMAT: INFORMAT: */
/* JOBSPLIT: ATTR #C00002.CLASS.DATA INPUT VARIABLE:Sex TYPE:CHARACTER LENGTH:1 LABEL: FORMAT: INFORMAT: */
/* JOBSPLIT: ATTR #C00002.CLASS.DATA INPUT VARIABLE:Age TYPE:NUMERIC LENGTH:8 LABEL: FORMAT: INFORMAT: */
/* JOBSPLIT: ATTR #C00002.CLASS.DATA INPUT VARIABLE:Height TYPE:NUMERIC LENGTH:8 LABEL: FORMAT: INFORMAT: */
/* JOBSPLIT: ATTR #C00002.CLASS.DATA INPUT VARIABLE:Weight TYPE:NUMERIC LENGTH:8 LABEL: FORMAT: INFORMAT: */
/* JOBSPLIT: ATTR WORK.Y.DATA OUTPUT VARIABLE:Name TYPE:CHARACTER LENGTH:8 LABEL: FORMAT: INFORMAT: */
/* JOBSPLIT: ATTR WORK.Y.DATA OUTPUT VARIABLE:Sex TYPE:CHARACTER LENGTH:1 LABEL: FORMAT: INFORMAT: */
/* JOBSPLIT: ATTR WORK.Y.DATA OUTPUT VARIABLE:Age TYPE:NUMERIC LENGTH:8 LABEL: FORMAT: INFORMAT: */
/* JOBSPLIT: ATTR WORK.Y.DATA OUTPUT VARIABLE:Height TYPE:NUMERIC LENGTH:8 LABEL: FORMAT: INFORMAT: */
/* JOBSPLIT: ATTR WORK.Y.DATA OUTPUT VARIABLE:Weight TYPE:NUMERIC LENGTH:8 LABEL: FORMAT: INFORMAT: */
/* JOBSPLIT: ELAPSED 96 */
/* JOBSPLIT: PROCNAME DATASTEP */
/* JOBSPLIT: STEP SOURCE FOLLOWS */

data y ;
    set sashelp.class ;
run ;

/* JOBSPLIT: TASKSTARTTIME 18JAN2017:08:17:56.80 */
/* JOBSPLIT: DATASET INPUT SEQ WORK.X.DATA */
/* JOBSPLIT: LIBNAME WORK V9
'/temp/work1/sas_9.4/SAS_work873E02930064_p24107prw310/SAS_work870802930064_p24107prw310' */
/* JOBSPLIT: OPENTIME WORK.X.DATA DATE:18JAN2017:08:17:56.80
PHYS:/temp/work1/sas_9.4/SAS_work873E02930064_p24107prw310/SAS_work8708
02930064_p24107prw310/x.sas7bdat SIZE:196608 */
/* JOBSPLIT: DATASET OUTPUT SEQ WORK.X2.DATA */
/* JOBSPLIT: LIBNAME WORK V9
'/temp/work1/sas_9.4/SAS_work873E02930064_p24107prw310/SAS_work870802930064_p24107prw310' */
/* JOBSPLIT: ATTR WORK.X.DATA INPUT VARIABLE:Name TYPE:CHARACTER LENGTH:8 LABEL: FORMAT: INFORMAT: */
/* JOBSPLIT: ATTR WORK.X.DATA INPUT VARIABLE:Sex TYPE:CHARACTER LENGTH:1 LABEL: FORMAT: INFORMAT: */
/* JOBSPLIT: ATTR WORK.X.DATA INPUT VARIABLE:Age TYPE:NUMERIC LENGTH:8 LABEL: FORMAT: INFORMAT: */
/* JOBSPLIT: ATTR WORK.X.DATA INPUT VARIABLE:Height TYPE:NUMERIC LENGTH:8 LABEL: FORMAT: INFORMAT: */
/* JOBSPLIT: ATTR WORK.X.DATA INPUT VARIABLE:Weight TYPE:NUMERIC LENGTH:8 LABEL: FORMAT: INFORMAT: */
/* JOBSPLIT: ATTR WORK.X2.DATA OUTPUT VARIABLE:Sex TYPE:CHARACTER LENGTH:1 LABEL: FORMAT: INFORMAT: */
/* JOBSPLIT: ATTR WORK.X2.DATA OUTPUT VARIABLE:_TYPE TYPE:NUMERIC LENGTH:8 LABEL: FORMAT: INFORMAT: */

```

```

/* JOBSPLIT: ATTR WORK.X2.DATA OUTPUT VARIABLE: _FREQ_ TYPE:NUMERIC LENGTH:8 LABEL: FORMAT: INFORMAT: */
/* JOBSPLIT: ATTR WORK.X2.DATA OUTPUT VARIABLE:Height TYPE:NUMERIC LENGTH:8 LABEL: FORMAT: INFORMAT: */
/* JOBSPLIT: SYMBOL GET SYSSUMSTACKODS */
/* JOBSPLIT: SYMBOL GET SYSSUMTRACE */
/* JOBSPLIT: ELAPSED 58 */
/* JOBSPLIT: PROCNAME SUMMARY */
/* JOBSPLIT: STEP SOURCE FOLLOWS */
proc summary data=x ;
  class sex ;
  var height ;
  output out=x2 mean= ;
run ;

```

Several tables are produced by the `scaproc_analyse` macro, but the main one that we are interested in is called `Graphviz`. This contains the DOT language statements which are used to make the diagram.

```

// Generated by SAS for Sample 1
// Percentiles: 50:0.035 60:0.05 70:0.051 80:0.092 90:0.1 95:0.113 99:0.113
digraph test {
graph [label="\n\nSample 1\n08DEC16:08:12:49"]
node [shape=box color=lightblue style=filled]
"PRINT"[shape=ellipse color=lightgreen]
"#C00002.CLASS"->"WORK.X" [label=" datastep (0.113)" color=red penwidth=7
style=solid];
"#C00002.CLASS"->"WORK.Y" [label=" datastep (0.100)" color=red penwidth=5
style=solid];
"WORK.X"->"WORK.X2" [label=" summary (0.092)" color=red penwidth=4 style=solid];
"WORK.Y"->"WORK.Y2" [label=" summary (0.051)" color=red penwidth=3 style=solid];
"WORK.X2"->"WORK.X3" [label=" sort (0.035)" color=red style=solid];
"WORK.Y2"->"WORK.Y3" [label=" sort (0.019)" style=solid];
"WORK.X3"->"WORK.Z" [label=" datastep (0.018)" style=solid];
"WORK.Y3"->"WORK.Z" [label=" datastep (0.018)" style=solid];
"WORK.Z"->"PRINT" [label="(0.050)" color=red penwidth=2 style=dashed];
"WORK.X"->"WORK.SQL_TABLE" [label=" sql (0.035)" color=red style=solid];
"WORK.Y"->"WORK.SQL_TABLE" [label=" sql (0.035)" color=red style=solid];
}

```

You can now copy the rows from this table and paste them into `WebGraphviz`.

WebGraphviz is [Graphviz](#) in the Browser

Enter your graphviz data into the Text Area:

(Your Graphviz data is private and never harvested)

Sample 1

Sample 2

Sample 3

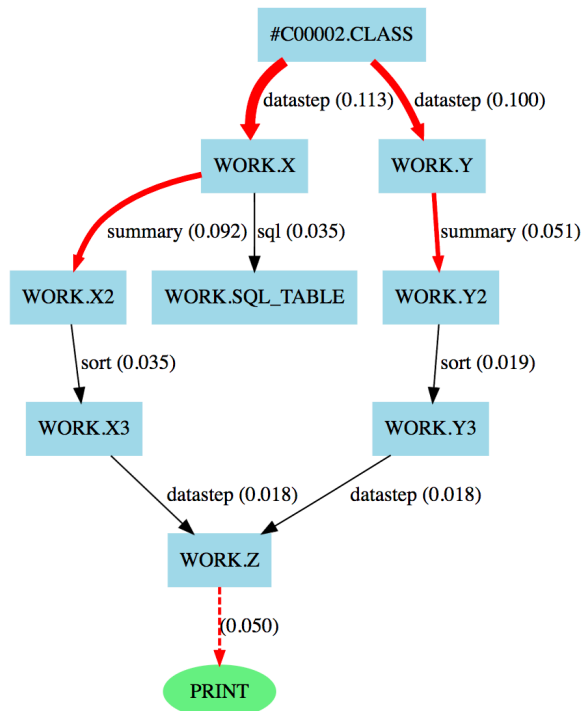
Sample 4

Sample 5

```
// Generated by SAS for Sample 1
// Percentiles: 50:0.035 60:0.05 70:0.051 80:0.092 90:0.1 95:0.113 99:0.113
digraph test {
graph [label="\n\nSample 1\n08DEC16:08:12:49"]
node [shape=box color=lightblue style=filled]
"PRINT"[shape=ellipse color=lightgreen]
"#C00002.CLASS"->"WORK.X" [label=" datastep (0.113)" color=red penwidth=7 style=solid];
"#C00002.CLASS"->"WORK.Y" [label=" datastep (0.100)" color=red penwidth=5 style=solid];
"WORK.X"->"WORK.X2" [label=" summary (0.092)" color=red penwidth=4 style=solid];
"WORK.Y"->"WORK.Y2" [label=" summary (0.051)" color=red penwidth=3 style=solid];
"WORK.X2"->"WORK.X3" [label=" sort (0.035)" color=red style=solid];
"WORK.Y2"->"WORK.Y3" [label=" sort (0.019)" style=solid];
"WORK.X3"->"WORK.Z" [label=" datastep (0.018)" style=solid];
"WORK.Y3"->"WORK.Z" [label=" datastep (0.018)" style=solid];
"WORK.Z"->"PRINT" [label="(0.050)" color=red penwidth=2 style=dashed];
"WORK.X"->"WORK.SQL_TABLE" [label=" sql (0.035)" color=red style=solid];
"WORK.Y"->"WORK.SQL_TABLE" [label=" sql (0.035)" color=red style=solid];
}
```

Generate Graph!

Now click on the "Generate Graph!" button and it will make your diagram.



Sample 1
08DEC16:08:12:49

Future Enhancements

There are an almost unlimited number of enhancements that can be made using these basic techniques outlined here.

1. Install Graphviz in a place accessible from SAS server so that we can then use it directly from a SAS program. This will enable the graphics to be produced automatically thereby removing any manual steps. This has already been tested on another server and works well.
2. Integrate output from SAS log so that we can choose to display various numbers on arrows and scale and colour them accordingly, such as Elapsed time, CPU time, records in/out, etc.
3. Fix “#” references from PROC SCAPROC with info from code and/or log to show what library they actually come from.
4. Integrate tool into a Stored Process, to allow running an entire analysis by filling in some parameters and browsing to your code.
5. Make simple version of diagram based only on SAS code.
6. Make diagram that can be generated from a SAS log, enabling us to create diagrams on logs run in the past.
7. Create framework to store graph DOT commands and then enable comparison with others previously stored, so that we could compare performance between multiple runs, highlight changes in the structure of complex code diagrammatically, etc.
8. Provide parameters to allow changing of line attributes, box attributes, etc.
9. Add support for showing diagrams in a web page, which then will enable use of tooltips and links which will enable a lot more functionality such as popping up a box showing code used in a step directly from diagram.
10. Enhance code to use the grid analysis capabilities. This will suggest how to grid enable your SAS code, and how much time could be saved.

APPENDIX

SAS Macros

The following SAS macros are the version of these at the time of creating this document. They show the technique although the code is currently under development so it might not handle the analysis of all kinds of SAS code, and won't have features currently being added in development. Feel free to make your own enhancements and share the code with others.

- EANBEGIN – “Enhanced Analysis” begin, which is used before some SAS code that you want to analyse. This follows the SAS naming scheme along the lines of STPBEGIN/STPEND. Parameters are:
 - use_label is a positional parameter which can optionally specify a label to use in .DOT output that is produced. If you don't specify a value then one is created for you.
 - where is a keyword parameter which can optionally specify a location for PROC SCAPROC to write its output to, will default to a text file in your own home directory.
 - _options is a keyword parameter which can optionally specify options to use when using PROC SCAPROC, they are just passed through to the procedure itself.
- EANEND – “Enhanced Analysis” end, which is used at the end of SAS code which is being analysed. This causes the analysis to be written out and is then ready to be analysed. This macro has no parameters.
- PERCENTILES – Creates a range of global macro variables in the form PCTn, where n is the percentile. E.g. PCT50 is the 50th percentile. There is a default list of them created although you can override which ones are created. e.g. `%percentiles(sashelp.air,air,pctlpts=25 50 75)`
 - Dset is a positional parameter which specifies the dataset to use
 - var is a positional parameter which specifies the variable to use from dataset for generating %tiles
 - pctlpts is a keyword parameter which can optionally specify a list of percentiles to generate, separated by spaces. A global macro variable will be created for each one. They are prefixed with PCT. e.g. 10 would produce `&pct10`
- SCAPROC_ANALYSE – Reads in a text file containing the output from PROC SCAPROC and produces a table containing DOT directives. These are a set of directives used with a program called GraphViz which enable

graphs & diagrams to be produced. There are some manual steps that must then be taken which are outlined in the description below.

```

/*****
* Macro:                EANBEGIN
*
* Author:                Philip Mason
*
* Date Created:         01/08/2016
*
* Description:          Enhanced Analysis - begins enhanced analysis
*                      Macro variable _EANDEBUB defines what kind of enhanced
*                      analysis is carried out according to the following values:
*                      - scaproc = uses Proc SCAPROC
*                      - verbose = turns on extra verbose logging
*                      - on = has the effect of scaproc,verbose
*                      - 1 = has the effect of scaproc,verbose
*                      values for _EANDEBUB should be comma separated
*
* Parameters:           use_label = optionally specify a label to use in .DOT
*                      output that is produced.
*                      where = optionally specify a location for PROC SCAPROC
*                      to write its output to, will default to a text
*                      file in your own work directory
*                      _options = optionally specify options to use when using
*                      PROC SCAPROC
*
* Data Sources:         n/a
*
* Data Output:          &where defines where to save SCAPROC output
*                      default puts it in users Work directory
*
* Auxiliary Files:      n/a
*
* -----
* Modification History
* Date      By      Details
* 01/08/2016 PM    Original Coding
*****/
%macro eanbegin(use_label,
               where=%sysfunc(pathname(WORK))/scaproc.txt,
               _options=attr opentimes expandmacros) ;
  %* if _eandebub macro exists and is not set to 0 or off, then continue ;
  %if %symexist(_eandebub)=0 %then %return ;
  %if &_eandebub=0 or %upcase(&_eandebub)=OFF %then %return ;
  %if %superq(use_label)= %then %let use_label=Program run by &sysuserid ;

  %* process parameters ;
  %let n_parms=%eval(%sysfunc(count(%superq(_eandebub),%str(,)))+1) ;
  %do i=1 %to &n_parms ;
    %let parm=%upcase(%scan(%superq(_eandebub),&i,%str(,))) ;
    %put INFO: EANBEGIN macro invoked: &parm ;
    %if &parm=SCAPROC or &parm=ON or &parm=1 %then %do ;
      * Start recording info to a text file ;
      proc scaproc ;
        record "&where" &_options ;
      run ;
      %global scaproc_label ;
    %end ;
  %end ;
%mend eanbegin ;

```

```

        %let scaproc_label=&use_label ;
    %end ; /* scaproc */
%else %if &parm=VERBOSE or &parm=ON or &parm=1 %then %do ;
    * Turn on various extra logging ;
    options symbolgen
           mlogic mlogicnest
           mprint mprintnest
           mautolocdisplay mautocomploc
           msglevel=i ;
    %* look at macro variables before process ;
    %put _all_ ;
    %end ; /* verbose */
%end ; /* do */
%mend eanbegin ;

/*****
* Macro:                EANEND
*
* Author:               Philip Mason
*
* Date Created:         01/08/2016
*
* Description:          Enhanced Analysis - ends enhanced analysis
*
* Parameters:           n/a
*
* Data Sources:         n/a
*
* Data Output:          as defined by the invocation of macro EANBEGIN
*
* Auxiliary Files:      n/a
*
* -----
* Modification History
* Date      By      Details
* 01/08/2016 PM    Original Coding
*****/
%macro eanend ;
    %* if _eandebg macro exists and is not set to 0 or off, then continue ;
    %if %symexist(_eandebg)=0 %then %return ;
    %if &_eandebg=0 or %upcase(&_eandebg)=OFF %then %return ;

    %* process parameters ;
    %let n_parms=%eval(%sysfunc(count(%superq(_eandebg),%str(,))))+1) ;
    %do i=1 %to &n_parms ;
        %let parm=%upcase(%scan(%superq(_eandebg),&i,%str(,))) ;
        %put INFO: EANBEGIN macro invoked: &parm ;
        %if &parm=SCAPROC or &parm=ON or &parm=1 %then %do ;
            * write out the recorded info ;
            proc scaproc ;
                write ;
            run ;
            %end ; /* scaproc */
        %else %if &parm=VERBOSE or &parm=ON or &parm=1 %then %do ;
            %* look at macro variables after process ;
            %put _all_ ;
            %end ; /* verbose */
        %end ; /* do */
    %mend eanend ;

```

```

/*****
* Macro: PERCENTILES
*
* Author: Philip Mason
*
* Date Created: 01/08/2016
*
* Description: Generate percentiles using a variable from a table and
*              then write them into a series of macro variables
*
* Parameters:  dset = dataset to use
*              var = variable to use from dataset for generating %tiles
*              pctlpts = list of percentiles to generate, separated by
*                      spaces. A global macro variable will be created
*                      for each one. e.g. 10 would produce &pctl10
*
* Data Sources: &dset
*
* Data Output:  _stats
*
* Auxiliary Files: n/a
*
*-----
* Modification History
* Date      By      Details
* 01/08/2016 PM    Original Coding
*****/
%macro percentiles(dset, /* dataset to use */
                  var, /* variable to use */
                  pctlpts=50 60 70 80 90 95 99, /* percentiles that we want */
                  clear_first=0 /* 1 = clear macro variables named 'PCT%' first */
                  ) ;
    * work out some percentiles ;
    proc univariate data=&dset noprint ;
        var &var ;
        output out=_stats
            pctlpts = &pctlpts
            pctlpre = pc
            pctlname =
            %do i=1 %to %eval(%sysfunc(count(&pctlpts,%str( )))+1) ;
                %let item=%scan(%superq(pctlpts),&i,%str( )) ;
                t&item
            %end ;
    ;
run ;

%if &clear_first %then %do ;
    %let pct_vars=;
    proc sql noprint ;
        select name into :pct_vars separated by ' '
            from dictionary.macros where name like 'PCT%' ;
    %symdel &pct_vars / nowarn ;
%end ;

* put the percentiles into global macro variables ;
data _null_ ;
    length varname $ 32 ;
    dsid=open("_stats") ;
    call set(dsid) ;
    rc=fetch(dsid) ;

```

```

do i=1 to attrn(dsid,'nvars') ;
    varname=varname(dsid,i) ;
    value=getvarn(dsid,i) ;
    call execute('%global '||varname||';') ;
    call symputx(varname,value) ;
    put varname '=' value ;
end ;
dsid=close(dsid) ;
run ;
%mend percentiles ;

```

```

/*****
* Program:          SCAPROC_ANALYSE
*
* Author:           Philip Mason
*
* Date Created:     01/08/2016
*
* Description:      Analyse output from PROC SCAPROC, which has previously
*                   been written to a text file.
*                   After running this ...
*                   Now there are some manual steps. These could be automated,
*                   but would need to install some software or use the Stored
*                   Process Web App.
*                   1 - Copy the lines from the table just created (GRAPHVIZ).
*                   View the table in EG.
*                   Click on column, to select all values in it
*                   Then control-C.
*                   2 - Go to http://webgraphviz.com/ .
*                   3 - Paste the lines into the Text Area.
*                   4 - Press "Generate Graph!" button.
*
* Parameters:       scaproc_dir - directory where the proc scaproc output is
*                   scaproc_file - file name of the proc scaproc output
*
* Data Sources:     %sysfunc(pathname(WORK))/scaproc.txt
*
* Data Output:      WORK.GRAPHVIZ
*
* Auxiliary Files:  n/a
*
* -----
* Modification History
* Date      By      Details
* 01/08/2016 PM    Original Coding
*****/

```

```

*****
*** You need to have already produced a file to analyse by using the ***
*** eanbegin and eanend macros put around the code you want to analyse. ***
*****;

```

```

%macro scaproc_analyse(scaproc_dir=%sysfunc(pathname(WORK)),scaproc_file=scaproc.txt) ;
* list the directory to see the file created ;
data _null_ ;
    infile "&scaproc_dir./&scaproc_file" ;
    input ;
    put _infile_ ;

```

```

run ;

* read in the info and parse into a SAS table ;
filename scaproc "&scaproc_dir./&scaproc_file" ;
data scaproc ;
    length word1-word6 $ 46 ;
    retain step 1 ;
    infile scaproc ;
    input ;
    put _infile_ ;
    if _infile_ =: '/* JOBSPLIT: ' ;
    word1=scan(_infile_,2,' ') ;
    word2=scan(_infile_,3,' ') ;
    word3=scan(_infile_,4,' ') ;
    word4=scan(_infile_,5,' ') ;
    word5=scan(_infile_,6,' ') ;
    word6=scan(_infile_,7,' ') ;
    if word2='DATASET' & word3='INPUT' then
in=strip(word4)||'~'||scan(word5,1,'.')||'.'||scan(word5,2,'.') ;
    if word2='DATASET' & word3='OUTPUT' then
out=strip(word4)||'~'||scan(word5,1,'.')||'.'||scan(word5,2,'.') ;
    if word2='DATASET' & word3='UPDATE' then
out=strip(word4)||'~'||scan(word5,1,'.')||'.'||scan(word5,2,'.') ;
    if word2='PROCNAME' then procname=word3 ;
    if word2='ELAPSED' then elapsed=input(word3,8.3) ;
    output ;
    if index(_infile_,'STEP SOURCE FOLLOWS') then step+1 ;
run ;

* merge the data into one record for each step ;
proc sql noprint ;
create table flow as
    select coalesce(a.step,b.step,c.step) as step
        ,a.procname
        ,coalesce(scan(b.in,1,'~'),scan(c.in,1,'~')) as in_access
        ,coalesce(scan(b.out,1,'~'),scan(c.out,1,'~')) as out_access
        ,coalesce(scan(b.in,2,'~'),scan(c.in,2,'~')) as in
        ,coalesce(scan(b.out,2,'~'),scan(c.out,2,'~')) as out
        ,d.elapsed
    from
        scaproc(where=(procname>'')) as a
    full join
        scaproc(where=(in>'')) as b
        on a.step=b.step
    full join
        scaproc(where=(out>'')) as c
        on a.step=c.step
    left join
        scaproc(where=(elapsed>0)) as d
        on a.step=d.step
    order by calculated step
    ;
create table procnames as
    select distinct procname
    from flow
    where procname is not missing and (missing(in) or missing(out)) ;
quit ;

%* create percentiles for use in making diagram ;
%percentiles(flow, elapsed)
%put _user_ ;

```

```

* Create .DOT directives to make a diagram ;
data graphviz(keep=line) ;
  length line $ 140
  p $ 32
  color penwidth $ 12 ;
if _n_=1 then do ;
  line="// Generated by SAS for %superq(scaproc_label)" ;
  output ;
  line="// Percentiles: 50:&pct50 60:&pct60 70:&pct70 80:&pct80 90:&pct90 95:&pct95
99:&pct99" ;
  output ;
  line='digraph test {' ;
  output ;
/*      line='rankdir=LR' ;*/
/*      output ;*/

line="graph
[label=""\n\n%superq(scaproc_label)\n%sysfunc(datetime(),datetime.)"""] ;
  output ;
  line='node [shape=box color=lightblue style=filled]' ;
  output ;
  dsid=open('procnames') ;
  do while(fetch(dsid)=0) ;
    p=getvarc(dsid,1) ;
    line=quote(strip(p))||'[shape=ellipse color=lightgreen]' ;
    output ;
  end ;
  dsid=close(dsid) ;
  end ;
set flow end=end ;
in=quote(strip(in)) ;
out=quote(strip(out)) ;
procname=quote(strip(procname)) ;
if elapsed>=&pct50 then color='color=red' ;
      else color='' ;
if elapsed>=&pct99 then penwidth='penwidth=7' ; else
if elapsed>=&pct95 then penwidth='penwidth=6' ; else
if elapsed>=&pct90 then penwidth='penwidth=5' ; else
if elapsed>=&pct80 then penwidth='penwidth=4' ; else
if elapsed>=&pct70 then penwidth='penwidth=3' ; else
if elapsed>=&pct60 then penwidth='penwidth=2' ;
      else penwidth='' ;
if in_access='MULTI' or out_access='MULTI' then style='style=dashed' ;
      else style='style=solid' ;
if compress(in,'')>' ' & compress(out,'')>' ' then
  line=strip(in)||'->'||strip(out)||
  ' [label=" '||lowercase(strip(dequote(procname)))||
  ' ('||strip(put(elapsed,8.3))||
  ')" '||strip(color)||' '||strip(penwidth)||
' ||strip(style)||'];' ;
  else if compress(in,'')>' ' & compress(out,'')='' then
    line=strip(in)||'->'||strip(procname)||
    ' [label="('||strip(put(elapsed,8.3))||
    ')" '||strip(color)||' '||strip(penwidth)||
' ||strip(style)||'];' ;
  else if compress(in,'')='' & compress(out,'')>' ' then
    line=strip(procname)||'->'||strip(out)||
    ' [label="('||strip(put(elapsed,8.3))||
    ')" '||strip(color)||' '||strip(penwidth)||
' ||strip(style)||'];' ;
  else line="// '||strip(procname)||' ('||strip(put(elapsed,8.3))||')' ;
  output ;
if end then do ;

```

```
        line='}' ;  
        output ;  
    end ;  
run ;  
%mend scaproc_analyse ;
```

CONCLUSION

Combining SAS and GraphViz gives you the tools to build amazing diagrams. You can make simple diagrams with very little programming or put all kinds of extra effort into creating far more elaborate and explanatory diagrams.

RECOMMENDED READING

- PROC SCAPROC documentation
- <http://support.sas.com/documentation/cdl/en/proc/68954/HTML/default/p0sf63lx4fs2m5n14qv1bn8p863v.htm>
- Documentation for GraphViz, which enables Graph Visualisation via directives provided in text form
- <http://www.graphviz.org/>
- DOT language info & links - [https://en.wikipedia.org/wiki/DOT_\(graph_description_language\)](https://en.wikipedia.org/wiki/DOT_(graph_description_language))
- Web Version of GraphViz, which enables us to use GraphViz without installing it - <http://webgraphviz.com/>

CONTACT INFORMATION

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

Philip Mason
Wood Street Consultants
+44 1491 200635
phil@woodstreet.org.uk

SAS and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. ® indicates USA registration.

Other brand and product names are trademarks of their respective companies.