Science For A Better Life

Macumba
State of the Art SAS GUI

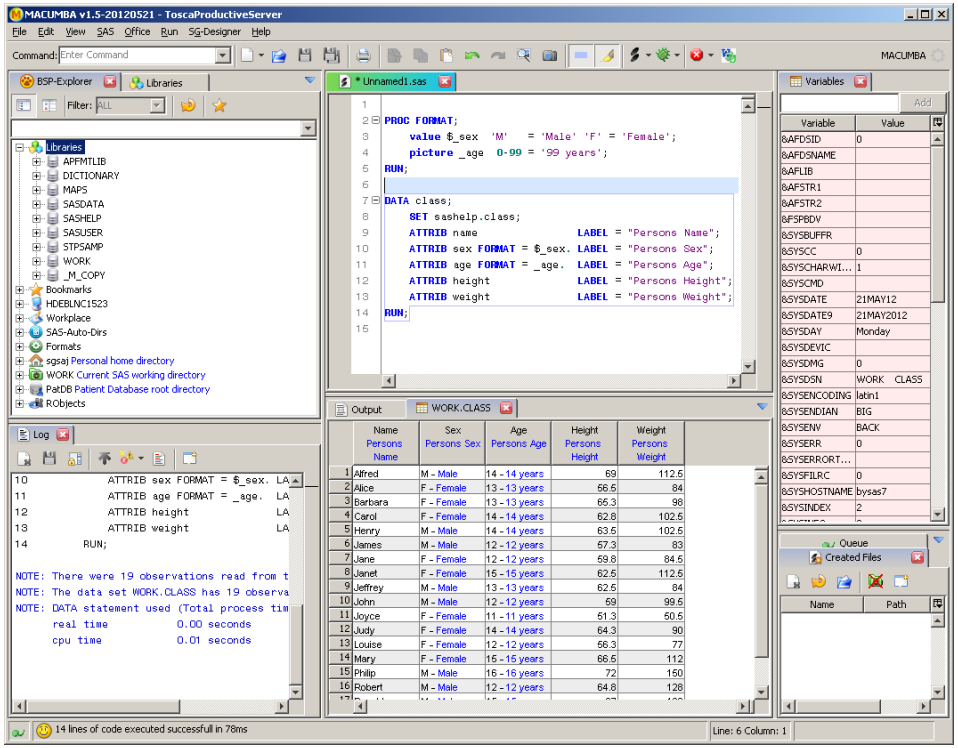
Debugging made easy

Macumba – Debugging made easy – Overview

1. What is MACUMBA
   - Short Overview of the Application
2. Special SAS Code Execution Modes
   - Run (To/From) Line, Run Step
   - Run in Template
   - Run As Macro
   - Resolve Code
3. SAS DATA-Step Debugger
   - Overview
   - Compare with Interactive SAS Debugger
4. Outlook

Macumba – What is MACUMBA?

- In-House developed SAS IDE
- “Pseudo”-Interactive code execution on Remote SAS Server
- Focus on SAS program / macro development
- SAS code editor
  - Syntax Highlight
  - Code Completion, …
- Data Set Viewer / Editor
  - Sorting, Filtering, Statistics, …
  - Display of Formats / Labels

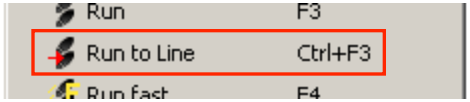


Macumba – Special SAS Code Execution Modes 1

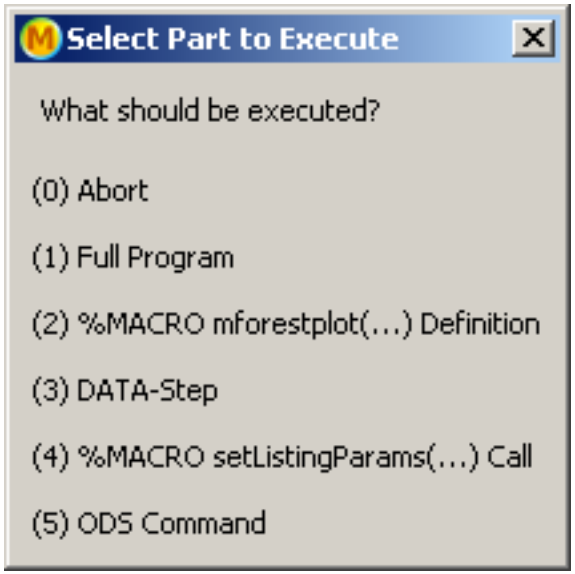
- PC-SAS provides two Execution Modes
  - Execute Selection (if code selected)
  - Execute All (if nothing selected)
- Uncomfortable when developing / debugging big programs
  - Execute from start to cursor position → scroll up / down
  - Execute from cursor position to program end → scroll down / up
  - ERROR when executing macro parts (%IF, %DO, %LOCAL, …)

Macumba – Special SAS Code Execution Modes 2

- Run To Line



  - Execute all code from start of program to cursor line
- Run From Line
  - Execute all code from cursor line to end of program
- Run Step
  - Execute current step
    - %MACRO Definition
    - DATA- or PROC-Step
    - Global Command
    - Macro Call



Macumba – Special SAS Code Execution Modes 3

- Run in Template
  - Execute the code wrapped in the specified template
    - Wrap a PROC PRINT into ODS RTF command

SAS Code

```
PROC PRINT DATA=sashelp.class;
RUN;
```

\+ RUN ODS_RTF

Template ODS_RTF

```
ODS RTF PATH="%SYSFUNC(pathname(WORK))"
      file="_temp.rtf";
${CODE}
ODS RTF CLOSE;
```

Executed Code

```
ODS RTF PATH="%SYSFUNC(pathname(WORK))"
      file="_temp.rtf";
PROC PRINT DATA=sashelp.class;
RUN;
ODS RTF CLOSE;
```

Macumba – Special SAS Code Execution Modes 4

- Run As Macro

SAS Code

```
%MACRO abc(param1=hello, param2=world, max=10);
   %LOCAL i;
   %DO i=1 %TO &max.;
      %IF %SYSFUNC(mod(&i., 2)) EQ 0
      %THEN %DO;
         %PUT &param1. &param2. (i=&i.);
      %END;
   %END;
%MEND;
%abc();
```

SAS Log

```
1          %abc();
hello world (i=2)
hello world (i=4)
hello world (i=6)
hello world (i=8)
hello world (i=10)
```

What if (for debugging) only the %DO loop should be executed?

SAS Code

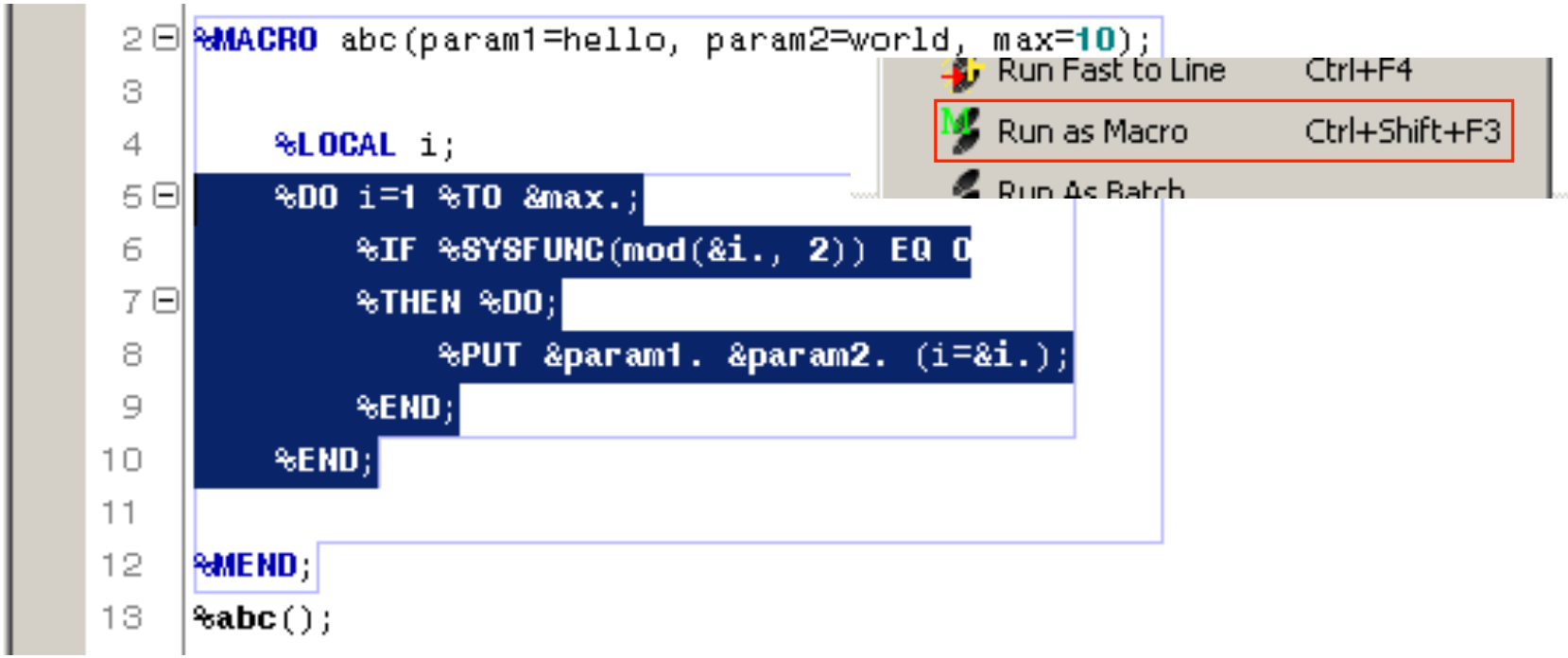
```
%DO i=1 %TO &max.;
   %IF %SYSFUNC(mod(&i., 2)) EQ 0
   %THEN %DO;
      %PUT &param1. &param2. (i=&i.);
   %END;
%END;
```

SAS Log

```
ERROR: The %DO statement is not valid in open
code.
ERROR: The %IF statement is not valid in open
code.
WARNING: Apparent symbolic reference PARAM1
not resolved.
WARNING: Apparent symbolic reference PARAM2
not resolved.
WARNING: Apparent symbolic reference I not
resolved.
```

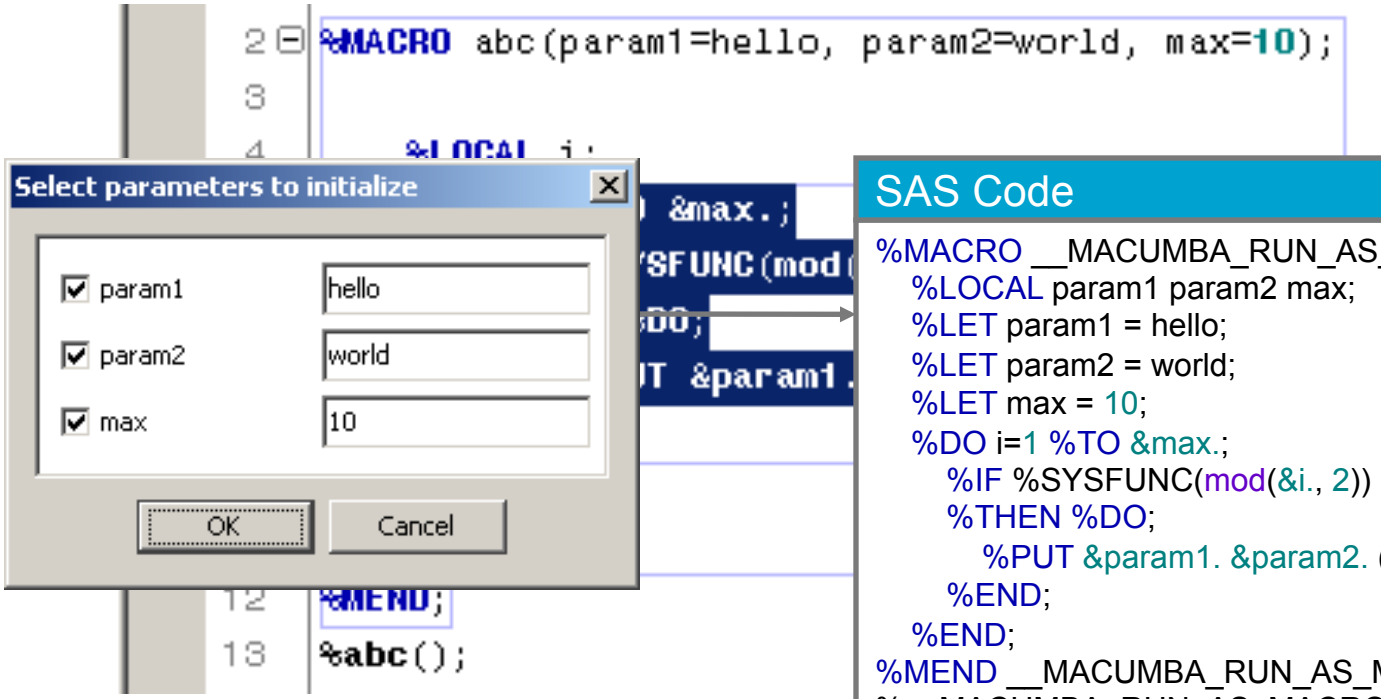
Macumba – Special SAS Code Execution Modes 4

- Run As Macro
  - MACUMBA action „Run As Macro“



Macumba – Special SAS Code Execution Modes 4

- Run As Macro
  - MACUMBA action „Run As Macro“



SAS Code

```
%MACRO __MACUMBA_RUN_AS_MACRO;
   %LOCAL param1 param2 max;
   %LET param1 = hello;
   %LET param2 = world;
   %LET max = 10;
   %DO i=1 %TO &max.;
      %IF %SYSFUNC(mod(&i., 2)) EQ 0
      %THEN %DO;
         %PUT &param1. &param2. (i=&i.);
      %END;
   %END;
%MEND __MACUMBA_RUN_AS_MACRO;
%__MACUMBA_RUN_AS_MACRO;
```

Macumba – Special SAS Code Execution Modes 5

- Resolve Code
  - What SAS code is executed for:

SAS Code

```
%doStudyEvaluation(studyId=0815, projectId=4711);
```

  - Execute:

SAS Code

```
OPTIONS MPRINT;
%doStudyEvaluation(studyId=0815, projectId=4711);
OPTIONS NOMPRINT;
```

  - Check your log

SAS Log

```
1          OPTIONS MPRINT;
2          %doStudyEvaluation(studyId=0815, projectId=4711);
Will create study formats for 4711 / 0815!
MPRINT(DOSTUDYEVALUATION):   PROC FORMAT;
MPRINT(DOSTUDYEVALUATION):   PICTURE _age low-high = '99 Years';
NOTE: Format _AGE is already on the library.
NOTE: Format _AGE has been output.
MPRINT(DOSTUDYEVALUATION):   RUN;

NOTE: PROCEDURE FORMAT used (Total process time):
      real time           0.00 seconds
```

Macumba – Special SAS Code Execution Modes 5

- Resolve Code
  - Execute:

SAS Code

```
FILENAME mprint "%SYSFUNC(pathname(WORK))/resolved.sas";
OPTIONS MPRINT MFILE;
%doStudyEvaluation(studyId=0815, projectId=4711);
OPTIONS NOMPRINT NOMFILE;
FILENAME mprint;
```

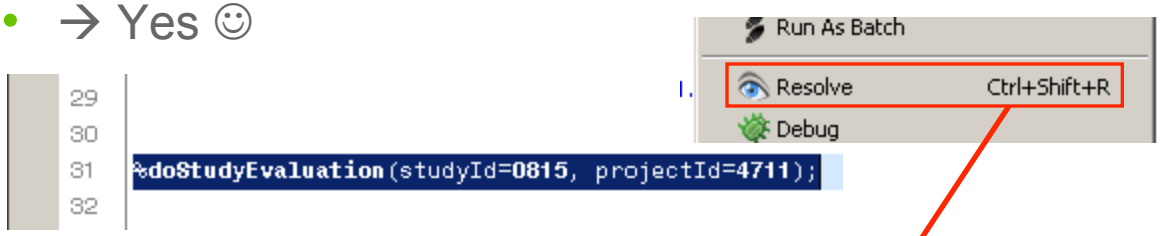
  - Check resolved.sas in work directory

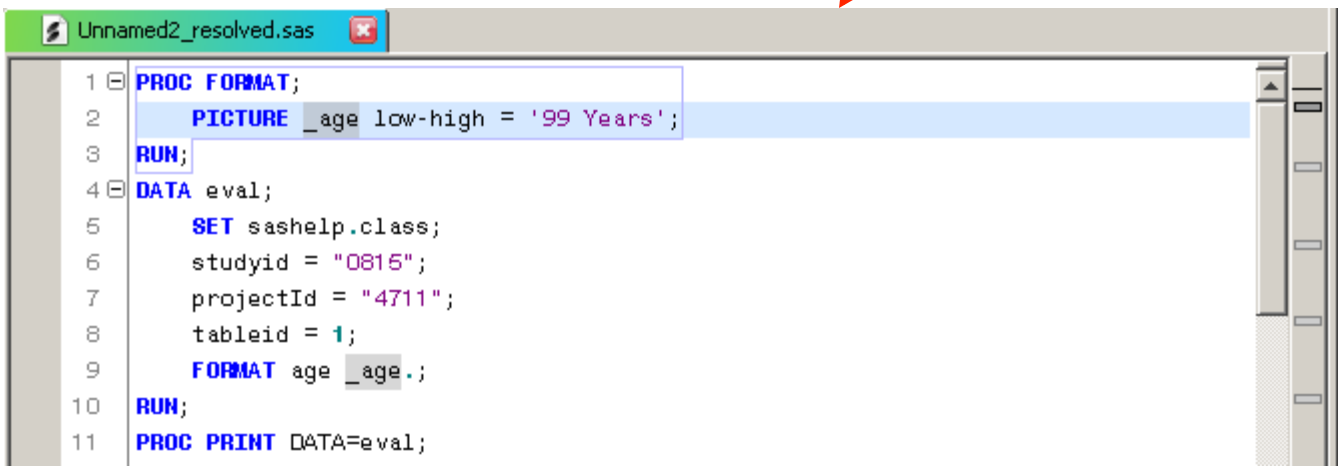
SAS Code (resolved.sas)

```
PROC FORMAT;
PICTURE _age low-high = '99 Years';
RUN;
DATA eval;
SET sashelp.class;
studyid = "0815";
projectId = "4711";
tableid = 1;
FORMAT age _age.;
RUN;
PROC PRINT DATA=eval;
RUN;
DATA eval;
SET sashelp.class;
```

Macumba – Special SAS Code Execution Modes 5

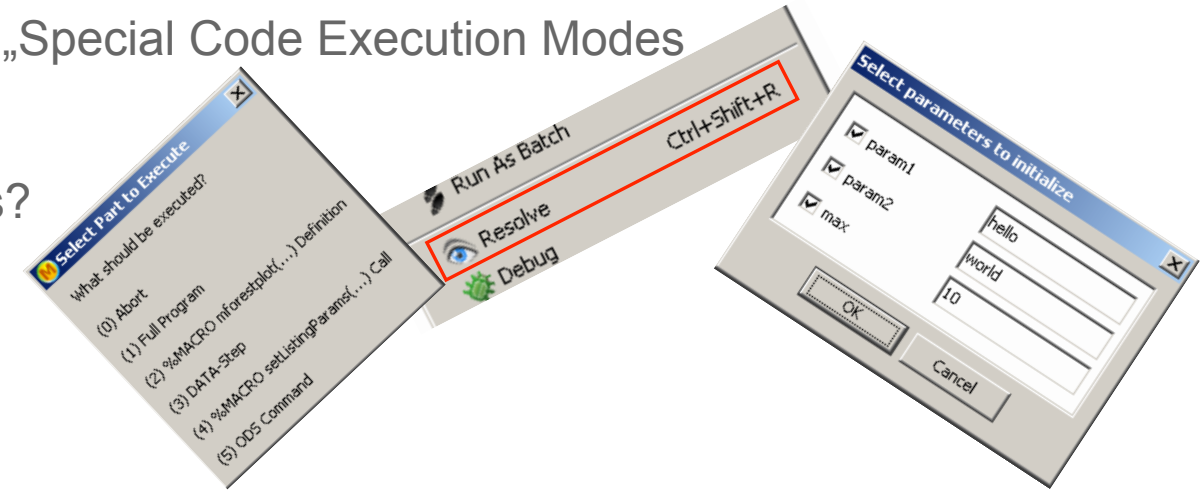
- Resolve Code
  - Can this be done automatically?
    - → Yes ☺



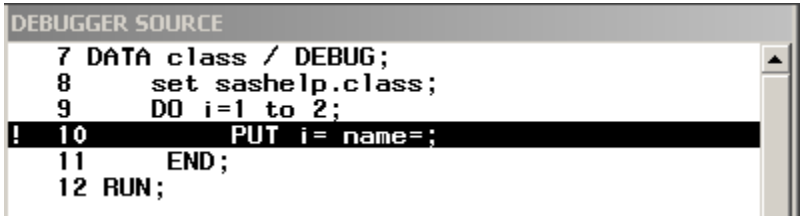


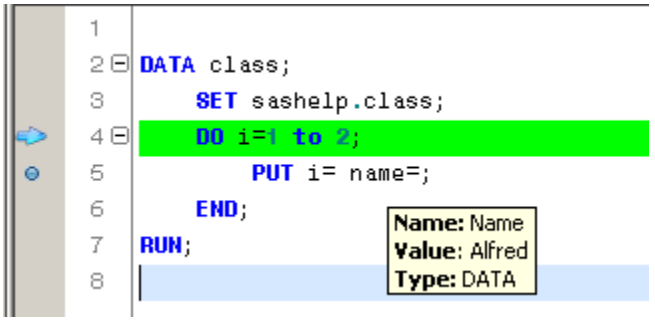
Macumba – Debugging made easy

- That‘s It For „Special Code Execution Modes
  - Questions?



- Next is “DATA-Step Debugger”





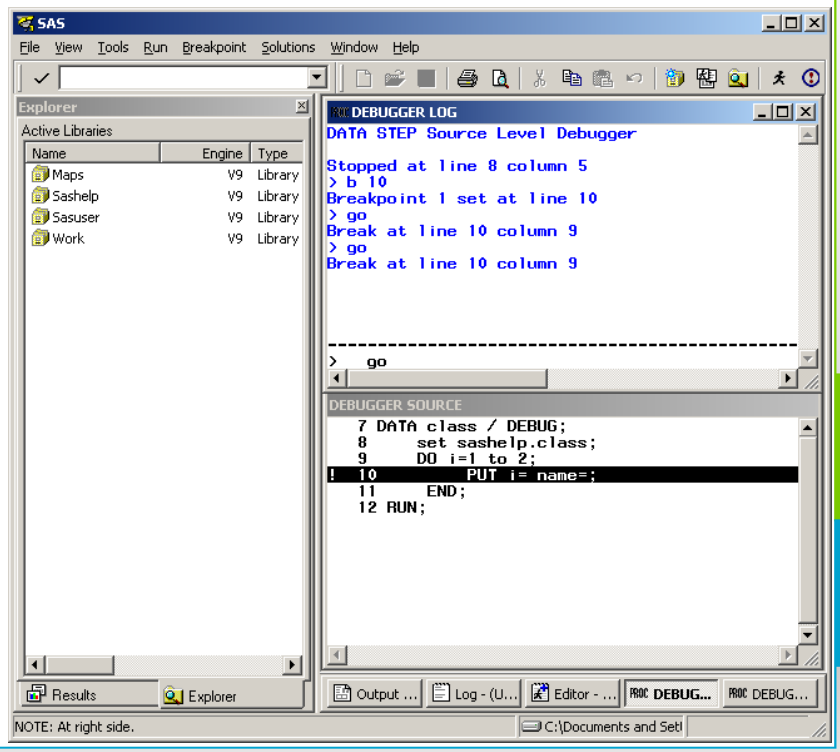
Macumba – DATA-Step Debugger – Overview - 1

- Interactive SAS DATA-Step Debugger
  - Change Code (add / DEBUG parameter to DATA command)

SAS Code

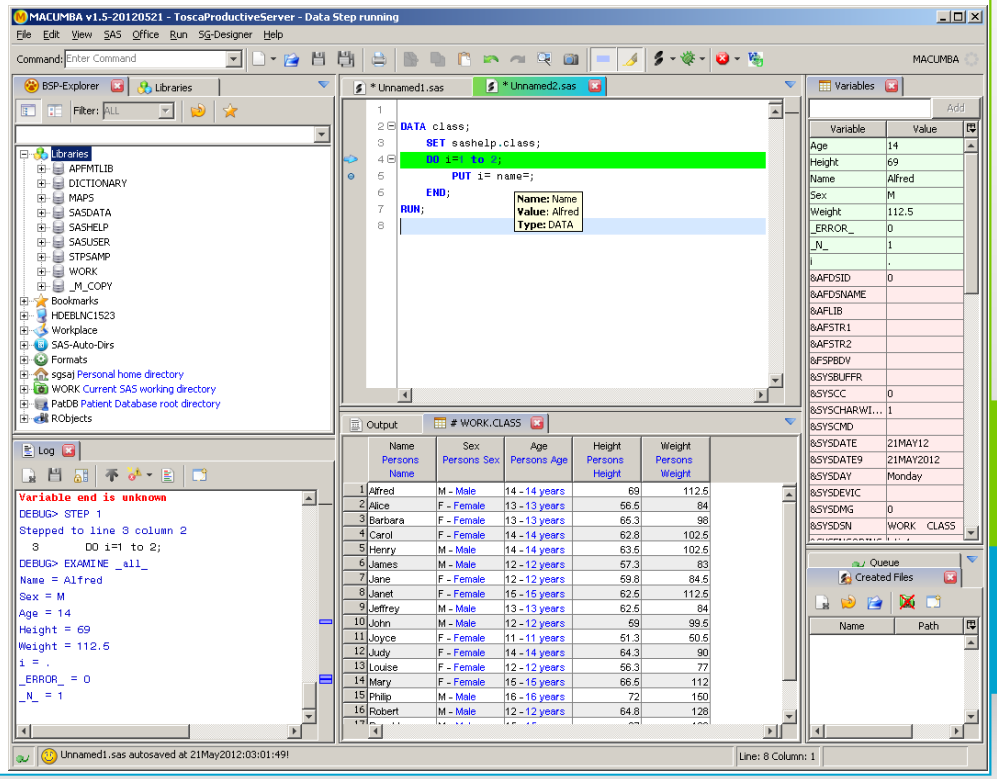
```
DATA class / DEBUG;
   SET sashelp.class;
   DO i=1 to 2;
      PUT i= name=;
   END;
RUN;
```

  - Command Driven
    - STEP, GO, BREAK, EXAMINE
  - Separate Window (No Syntax Highlight)



Macumba – DATA-Step Debugger – Overview - 2

- MACUMBA DATA-Step Debugger
  - Use Button to start
  - Shortcut / Mouse based
  - Similar to Eclipse / VBA
  - In SAS Code Editor
    - Same Syntax Highlight

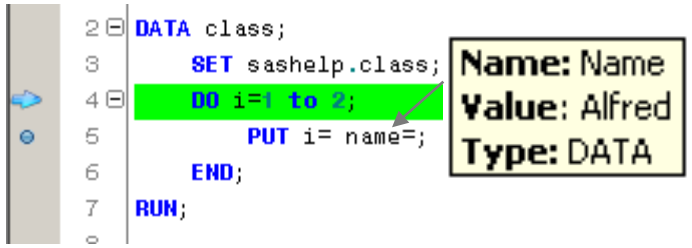


Macumba – DATA-Step Debugger – Implementation 1

- LDEBUG parameter added to DATA Command (internally)

SAS Code

```
DATA class / LDEBUG;
   SET sashelp.class;
   DO i=1 to 2;
      PUT i= name=;
   END;
RUN;
```



- Auto EXAMINE variable for Tool Tip Text
- Execute related SAS command in Background
  - Move Debug line → JUMP
  - Breakpoint Add / Delete (DELETE) BREAKPOINT <line>
  - …

Macumba – DATA-Step Debugger – Implementation 2

- Issues in Implementation

1. Q.: How to Find DATA-Step End?
   A1.: Search for line with RUN;

SAS Code

```
DATA tmp;
   a = step +
      run;
   PUT a=;
RUN;
```

SAS Code

```
DATA tmp;
   a = step + run;
   PUT a=; RUN;
```

Macumba – DATA-Step Debugger – Implementation 2

- Issues in Implementation
  1. Q.: How to Find DATA-Step End?
     A2.: <Semicolon> RUN <Semicolon> (ignore whitespaces)

SAS Code

```
DATA tmp;
   SET sashelp.class;
   /* simple copy of sashelp.class*/
RUN;
```

A3.: <Semicolon> RUN <Semicolon> (ignore whitespaces and comments)

SAS Code

```
DATA tmp;
   %calc_macro(p1=abc, p2=def)
RUN;
```

Macumba – DATA-Step Debugger – Implementation 2

- Issues in Implementation
  1. Q.: How to Find DATA-Step End?
     What about the following code?

SAS Code

```
DATA class;
   SET sashelp.class;
   DO i=1 to 2;
      PUT i= name=;
   END;

PROC SORT DATA=class;
   BY sex age name;
RUN;
```

SAS Code

```
DATA _NULL_;
   a=1;
   %PUT %STR(%);RUN;);
   PUT a=;
RUN;
```

A4.: Parse complete code, create code groups → use DATA group

Macumba – DATA-Step Debugger – Implementation 2

- Issues in Implementation
  - Q.: How to Determine if debugging ended?
    A1.: Debugging ended after the user pressed the Quit Button

    A2.: Scan log for "The DATA STEP program has completed execution"
    → issue QUIT command automatically afterwards
  - Q.: How to debug the following code:

SAS Code

```
DATA tmp;
   %calc_macro(p1=abc, p2=def)
RUN;
```

A.: Use Resolve Code before and debug resolved code ☺

Macumba – Outlook

- SAS Macro Debugger
  - Single Step debugging for SAS Macros (PhUSE 2012)