Why & How To Use SAS Macro Language: Easy Ways To Get More Value & Power from Your SAS® Software Tools

LeRoy Bessler PhD
Bessler Consulting and Research
Strong Smart Systems™
Mequon, WI, USA
Le_Roy_Bessler@wi.rr.com

All SAS products are trademarks of SAS Institute Inc. (Cary, NC) in the USA and other countries. ® denotes USA registration.
Strong Smart Systems is a trademark of LeRoy Bessler PhD.

About These Slides – Point 1

- They use the Rockwell font
- It IS embedded in the slides file
- But I am not sure how other computers will handle its display
- If you have something that is unreadable or horribly formatted, I can change the font, but it will be much work
- The next slide is a screen image of what THIS slide show look like in Rockwell

If your slide looks like this, the Rockwell font is used.

About These Slides – Point 1

- They use the Rockwell font
- It IS embedded in the slides file
- But I am not sure how other computers will handle its display
- If you have something that is unreadable or horribly formatted, I can change the font, but it will be much work
- The next slide is a screen image of what THIS slide show look like in Rockwell

Slightly reduced screen capture of the previous slide

About These Slides – Point 1

- They use the Rockwell font
- It IS embedded in the slides file
- But I am not sure how other computers will handle its display
- If you have something that is unreadable or horribly formatted, I can change the font, but it will be much work
- The next slide is a screen image of what THIS slide show look like in Rockwell

About These Slides – Point 2

- If you have any questions about the slides that are indicated as skipped during the live presentation, ask me.

Scope of Presentation

- Why Use Macro Language
- What IS a Macro
- How To Use Macro Language
- Simple Macros

Why Use Macro Language

Some SAS Software Facilities Require Use of Macro Language

E.g., Stored Processes

With a Macro

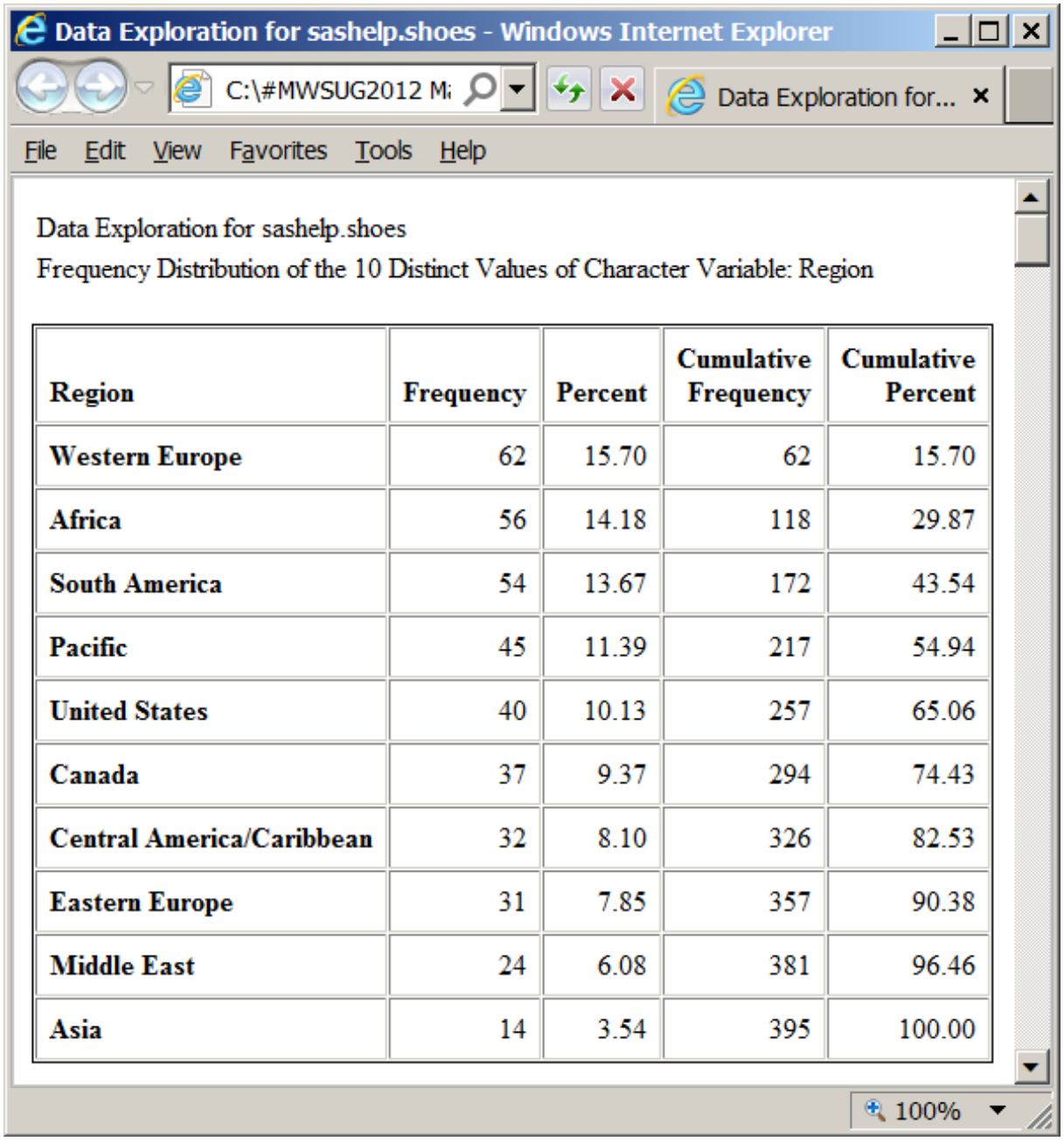
- Create a tool you can use
  to perform tasks that are largely similar,
  but differ in simple pre-definable ways
- You can share that tool with other users
- Similar to the PROC concept

Two real macros from the Paper

Here, only invocation code and pictures of output

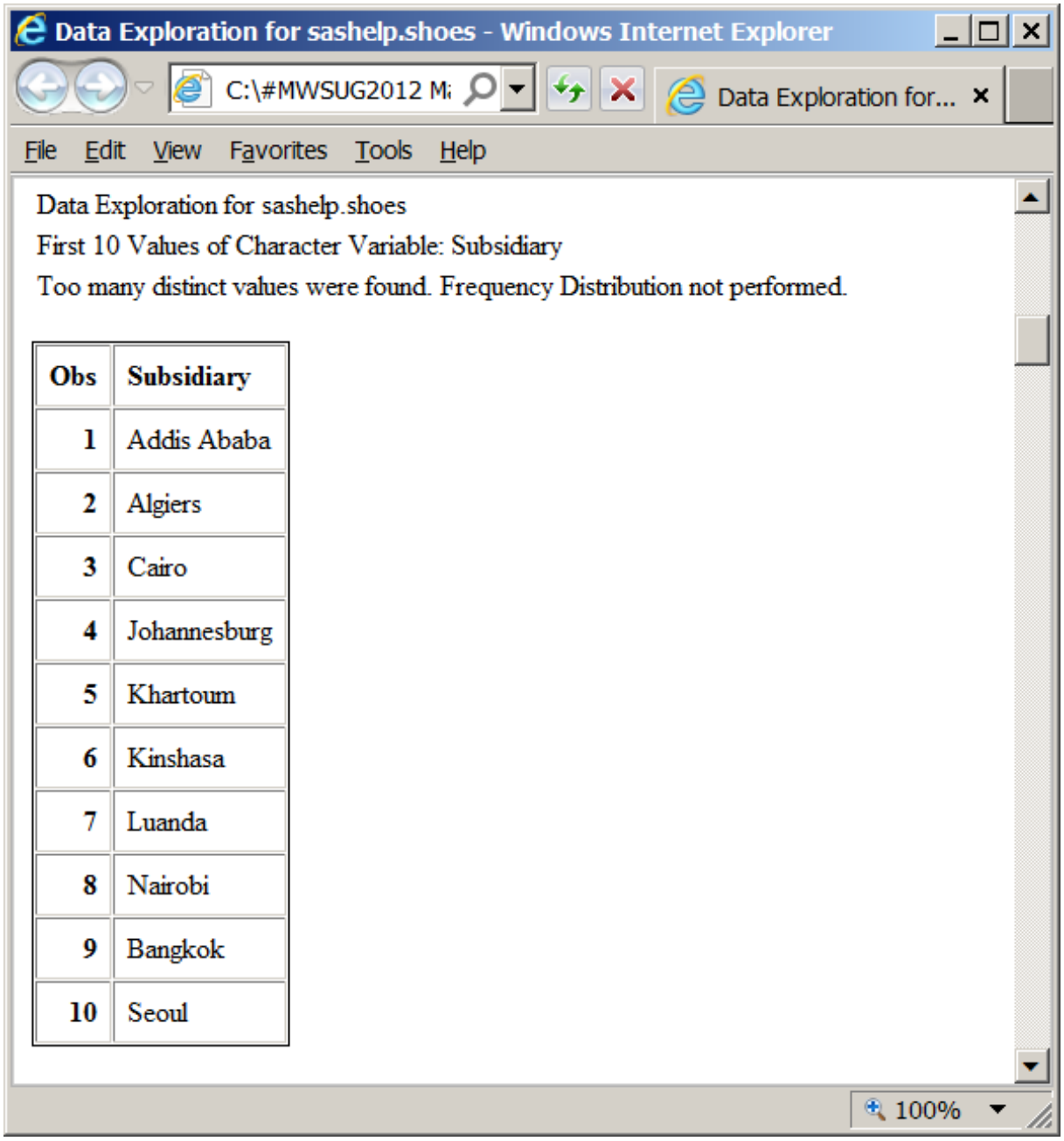
```
%DataExplorer(
libname=sashelp
,datasetname=shoes
,MaxDistinctForFreqAnalysis=10
,RptPath=C:\SomeFolderName
,RptFileName=SomeFileName);
```

Freq Dist for Character Var



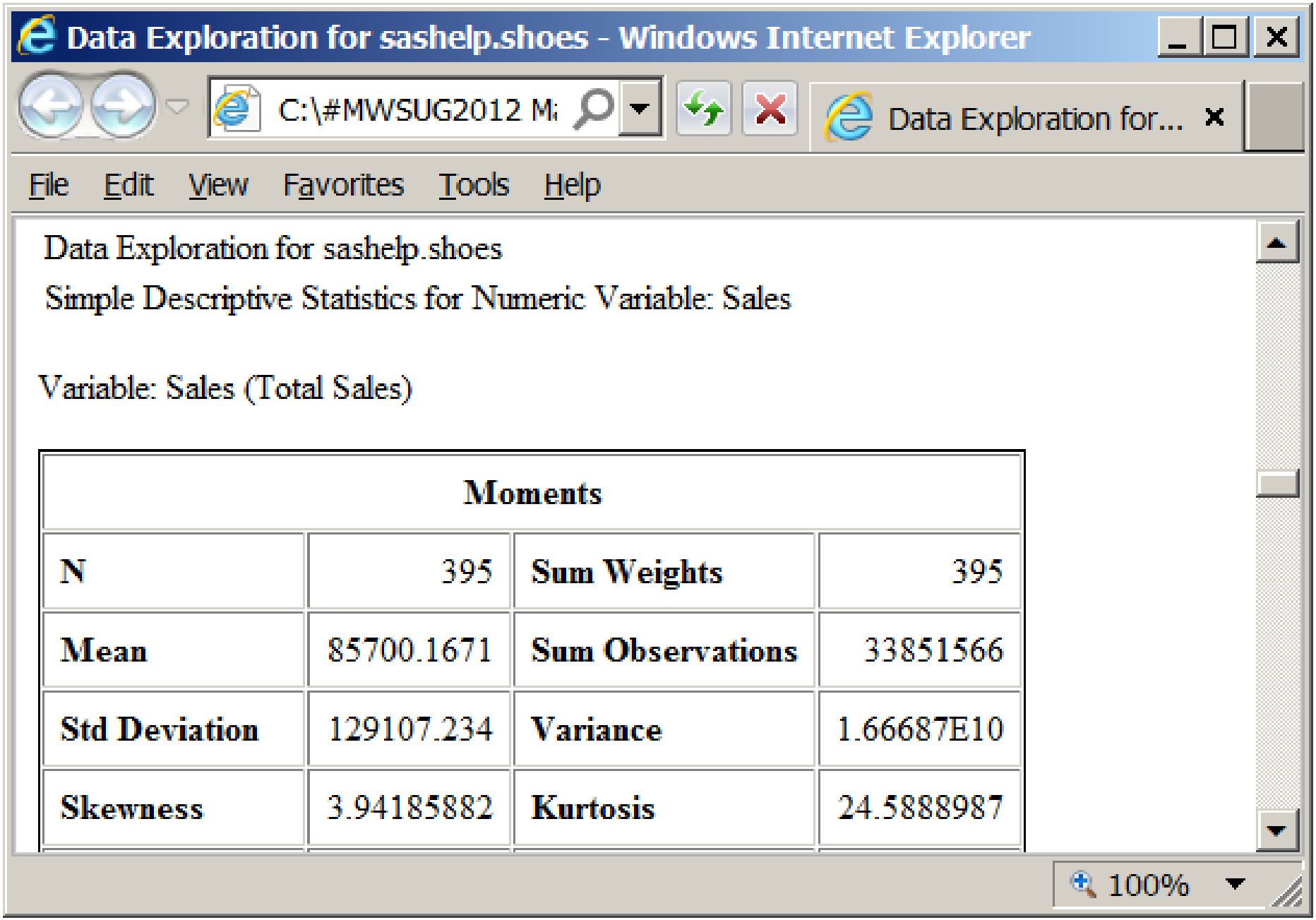
Region	Frequency	Percent	Cumulative Frequency	Cumulative Percent
Western Europe	62	15.70	62	15.70
Africa	56	14.18	118	29.87
South America	54	13.67	172	43.54
Pacific	45	11.39	217	54.94
United States	40	10.13	257	65.06
Canada	37	9.37	294	74.43
Central America/Caribbean	32	8.10	326	82.53
Eastern Europe	31	7.85	357	90.38
Middle East	24	6.08	381	96.46
Asia	14	3.54	395	100.00

Char Var with too many distinct values



Obs	Subsidiary
1	Addis Ababa
2	Algiers
3	Cairo
4	Johannesburg
5	Khartoum
6	Kinshasa
7	Luanda
8	Nairobi
9	Bangkok
10	Seoul

Some of Simple Stats for Numeric Var

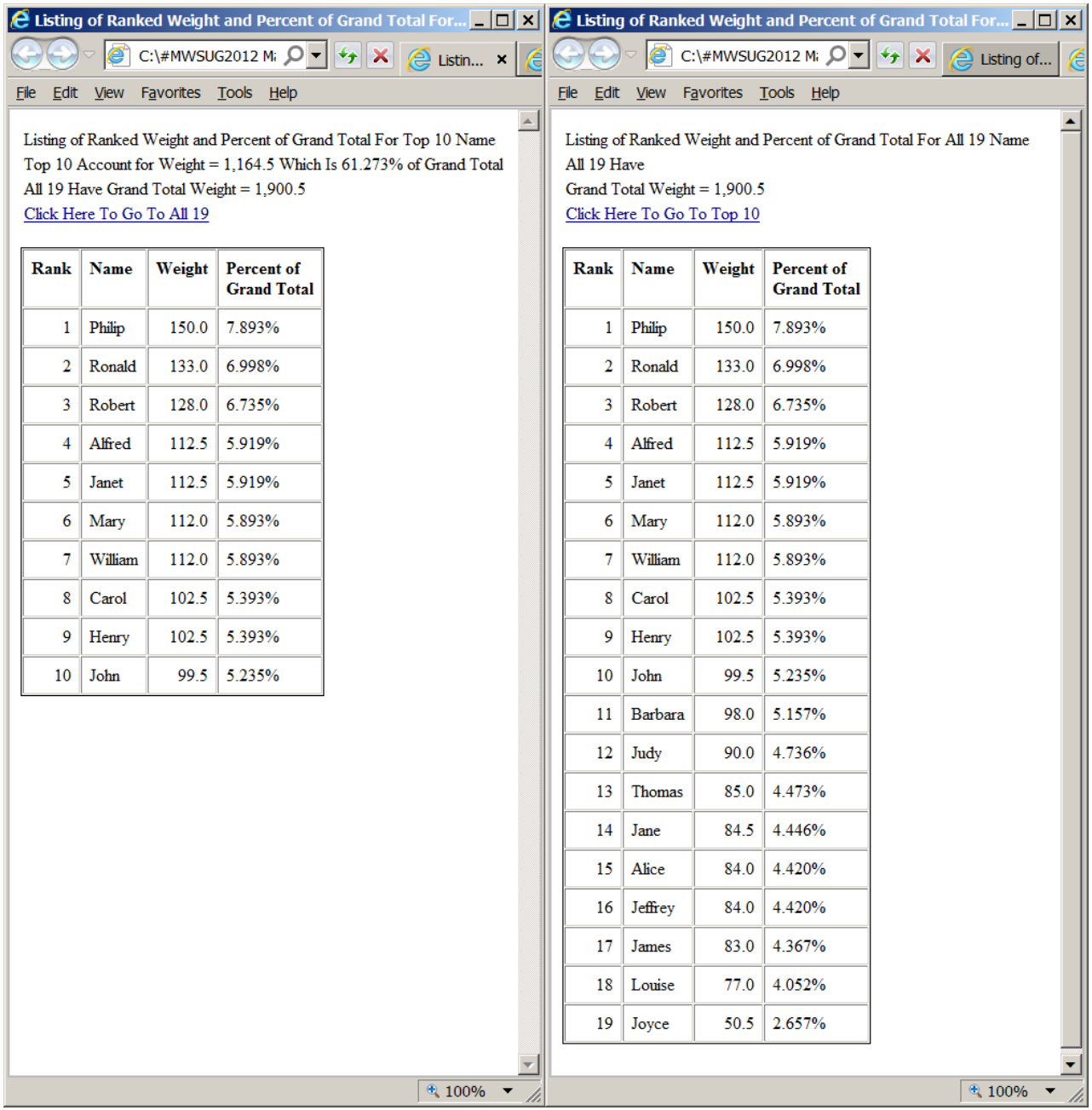


```
%SubsettedRankingReports(
data=sashelp.class
,ClassCountMax=10
,RptPath=C:\SomeFolderName
,Title4_LinkToTableWithCount=ALL
,ClassVar=Name
,ClassCountFormat=comma5.
,ClassMeasureVar=Weight
,ClassMeasureFormat=comma11.1);
```

Repeat above, but with:

```
,ClassCountMax=ALL
,Title4_LinkToTableWithCount=10
```

Tables & Titles dynamically created



Listing of Ranked Weight and Percent of Grand Total For...

C:\#MWSUG2012 M...

Listin...

File Edit View Favorites Tools Help

Listing of Ranked Weight and Percent of Grand Total For Top 10 Name
Top 10 Account for Weight = 1,164.5 Which Is 61.273% of Grand Total
All 19 Have Grand Total Weight = 1,900.5
Click Here To Go To All 19

Rank	Name	Weight	Percent of Grand Total
1	Philip	150.0	7.893%
2	Ronald	133.0	6.998%
3	Robert	128.0	6.735%
4	Alfred	112.5	5.919%
5	Janet	112.5	5.919%
6	Mary	112.0	5.893%
7	William	112.0	5.893%
8	Carol	102.5	5.393%
9	Henry	102.5	5.393%
10	John	99.5	5.235%

100%

Listing of Ranked Weight and Percent of Grand Total For...

C:\#MWSUG2012 M...

Listing of...

File Edit View Favorites Tools Help

Listing of Ranked Weight and Percent of Grand Total For All 19 Name
All 19 Have
Grand Total Weight = 1,900.5
Click Here To Go To Top 10

Rank	Name	Weight	Percent of Grand Total
1	Philip	150.0	7.893%
2	Ronald	133.0	6.998%
3	Robert	128.0	6.735%
4	Alfred	112.5	5.919%
5	Janet	112.5	5.919%
6	Mary	112.0	5.893%
7	William	112.0	5.893%
8	Carol	102.5	5.393%
9	Henry	102.5	5.393%
10	John	99.5	5.235%
11	Barbara	98.0	5.157%
12	Judy	90.0	4.736%
13	Thomas	85.0	4.473%
14	Jane	84.5	4.446%
15	Alice	84.0	4.420%
16	Jeffrey	84.0	4.420%
17	James	83.0	4.367%
18	Louise	77.0	4.052%
19	Joyce	50.5	2.657%

100%

What IS a Macro?

What IS a Macro?

- It is like a SAS procedure
  - Some of the words are predefined
  - You assign values to them

What IS a Macro?

- It is like a SAS procedure
  - You do not need to know what it does internally
  - You only care about what it does FOR you

What IS a Macro?

- It contains SAS code
  - familiar SAS language
  - SAS macro language
- When you supply values to it

  and submit it,

  SAS builds final (“resolved”) code

  and runs THAT code

What IS a Macro?

Macro "external" code:

- %MACRO statement to define macro
- %MEND statement to define its end
- %MacroName(. . .) to invoke macro as shown in examples above

What IS a Macro?

Macro internal code:

- There are parts of the code that are symbolic placeholders (&X, &Y, etc)
- There can be parts of the code that are conditional (%IF, %THEN, etc)

What is "Open Code"?

- It is SAS code outside of a macro
- NOT Open Code = macro code,
  i.e., code inside of a macro
- "inside of a macro" is code between
  %macro YourMacro Name . . . ;
  and
  %mend YourMacroName;

Macro internal code: symbolic placeholders (&X, &Y, etc)

Two ways to resolve symbols:

- Can be replaced by values assigned to parameters at macro invocation time
- Can be developed by run-time processing of the final code

Macro internal code: conditional tools (%IF, %THEN, etc)

Two ways to determine condition status:

- based on values assigned
  to parameters at macro invocation time
- from what processing discovers
  at run time based on whatever

Please, Please, Please

Just SHOW me how to CREATE a macro

How To Use Macro Language

SAS Macro Facility

SAS Language:
SAS statements, SAS functions, SAS variables

SAS Macro Language:

- macro statements
- macro functions
- macro variables (aka symbolic variables)

```
%macro MyFirstMacro(data=);

options nocenter nodate nonumber;
proc print data=&data; /* &data is to be replaced by
                          value assigned with data= */
run;

%mend  MyFirstMacro;
```

Submit:

```
options MPRINT; /* to see the code after
    replacement and what is actually run */
%MyFirstMacro(data=sashelp.class);
```

NOTE: Other related options are

MPRINTNEST – useful if macro invokes macros

MFILE – to specify a file where run-time macro-generated code can be stored, rather than only listed in the SAS log.

Find in SAS log (beginning of log content):

```
1    %macro MyFirstMacro(data=);
2
3    options nocenter nodate nonumber;
4    proc print data=&data; /* &data is to be replaced by value
assigned with data= */
5    run;
6
7    %mend  MyFirstMacro;
8
```

Continuation in SAS log:

```
9    options MPRINT; /* to see the code after replacement and
what is actually run */
10   %MyFirstMacro(data=sashelp.class);
MPRINT(MYFIRSTMACRO):   options nocenter nodate
nonumber;
MPRINT(MYFIRSTMACRO):   proc print data=sashelp.class;
MPRINT(MYFIRSTMACRO):   run;
 NOTE: There were 19 observations read from the data set
SASHELP.CLASS.
NOTE: PROCEDURE PRINT used (Total process time):
      real time          0.06 seconds
      cpu time           0.06 seconds
```

Here is an excerpt of the top of the result:

The SAS System

Obs	Name	Sex	Age	Height	Weight
1	Alfred	M	14	69.0	112.5
2	Alice	F	13	56.5	84.0
3	Barbara	F	13	65.3	98.0
4	Carol	F	14	62.8	102.5

< excerpt cut off here >

```
%macro MySecondMacro(data=,HowMany=);

options nocenter nodate nonumber;
options obs=&HowMany;
title
"Listing of First &HowMany observations in &data";
proc print data=&data;
run;
options obs=max;

%mend  MySecondMacro;
```

Submit:

```
%MySecondMacro(
data=sashelp.class
,HowMany=3
);
```

Here is the result:

Listing of First 3 observations in sashelp.class

Obs	Name	Sex	Age	Height	Weight
1	Alfred	M	14	69.0	112.5
2	Alice	F	13	56.5	84.0
3	Barbara	F	13	65.3	98.0

Macro Variables
a.k.a. Symbolic Variables

- used inside or outside of a macro
- Global versus Local

Local Macro Variables

- Where created (via either of two ways):
  - on %MACRO statement,
    like data in %MyFirstMacro(data=);
  - INSIDE a macro
- Where their values can be retrieved:
  - ONLY inside the creating macro

Global Macro Variables

- Where created:
  - outside a macro (in open code)
  - inside a macro
- Where their values can be retrieved:
  - outside a macro (in open code)
  - inside a macro

How To Create Global Macro Variables INSIDE of a Macro

- Macro variables created inside of a macro are Local by default
- Use %GLOBAL macvarname to make MacVarName a global macro variable
- Then MacVarName can be accessed outside (and inside) of the creating macro

```
options nosource nomprint;
%macro MyThirdMacro(data=);
proc print data=&data;
run;
%let MacVar1CreatedInsideMacro =  Created inside
macro without PercentGlobal statement;
%mend  MyThirdMacro;
%MyThirdMacro(data=sashelp.class); /* invoke it */
%put MacVar1CreatedInsideMacro is
&MacVar1CreatedInsideMacro; /* to try to display
value of macro variable in the SAS log */
```

In SAS Log:

```
2    options nosource nomprint;
NOTE: There were 19 observations read from the data
set SASHELP.CLASS.
NOTE: PROCEDURE PRINT used (Total process time):
      real time          0.03 seconds
      cpu time           0.01 seconds

WARNING: Apparent symbolic reference
MACVAR1CREATEDINSIDEMACRO not resolved.

MacVar1CreatedInsideMacro is
&MacVar1CreatedInsideMacro
```

Tip

Whenever you get this message

WARNING: Apparent symbolic reference SOMEMACROVARIABLENAME not resolved.

the most likely reason is

- either you misspelled the macro variable name here
- or you simply need to make it global at the place where you created it

```
options nosource nomprint;
%macro MyFourthMacro(data=);
proc print data=&data; run;
%global MacVar2CreatedInsideMacro;
%let MacVar2CreatedInsideMacro = This macro
variable was created inside a macro after the
PercentGlobal statement for it;
%mend  MyFourthMacro;
%MyFourthMacro(data=sashelp.class); /* invoke it */
%put MacVar2CreatedInsideMacro is
&MacVar2CreatedInsideMacro; /* to display value of
macro variable in the SAS log */
```

In SAS Log:

NOTE: There were 19 observations read from the data set SASHELP.CLASS.
NOTE: PROCEDURE PRINT used (Total process time):
real time 0.00 seconds
cpu time 0.00 seconds

MacVar2CreatedInsideMacro is This macro variable was created inside a macro after the PercentGlobal statement for it

Next set of slides shows how you can use SAME macro variable name outside a macro (as Global)
and
inside a macro (as Local)
This and following slides were skipped during the live presentation.

```
options nosource nomprint;


%let MacVar3 = Macro variable MacVar3 created
before macro invocation is GLOBAL, but macro
variable name is reused as LOCAL inside of macro;


%put **********************************************;
%put Next statement runs before macro invocation;
%put MacVar3 is &MacVar3;
%put **********************************************;
```

```
%macro MyFifthMacro(data=);
proc print data=&data;
run;
%local MacVar3;
%let MacVar3 = Macro variable MacVar3 being
reused as LOCAL inside of macro;
%put **********************************************;
%put Next statement runs inside macro execution;
%put MacVar3 is &MacVar3;
%put **********************************************;
%mend  MyFifthMacro;
```

```
%MyFifthMacro(data=sashelp.class);

%put **********************************;
%put Next statement runs after macro execution;
%put MacVar3 is &MacVar3;
%put **********************************;
```

In SAS Log:

****

Next statement runs before macro invocation

MacVar3 is Macro variable MacVar3 created before macro invocation is GLOBAL, but macro variable name is reused as LOCAL inside of macro

****

NOTE: There were 19 observations read from the data set SASHELP.CLASS.

NOTE: PROCEDURE PRINT used (Total process time):

real time 0.00 seconds

cpu time 0.00 seconds

****

Next statement runs inside macro execution

MacVar3 is Macro variable MacVar3 being reused as LOCAL inside of macro

****

Next statement runs after macro execution

MacVar3 is Macro variable MacVar3 created before macro invocation is GLOBAL, but macro variable name is reused as LOCAL inside of macro

****

End of skipped slides.

If you have a question, send me email.

To display your current macro variables and values in SAS log

```
%put _local_;
%put _global_;
%put _user_;
%put _automatic_; /* discussed later */
```

How To Assign Values To Macro Variables From Anywhere

Two Macro Statements That Can Be Used Anywhere: i.e., inside or outside macro

- %LET statement to assign values

```
%LET MyFirstName = LeRoy;
```

- %PUT to display values in SAS Log

```
%PUT My First Name is &MyFirstName;
```

In SAS Log:

My First Name is LeRoy

%LET statement: the value assigned is first non-blank through last non-blank before semi-colon

- Three assignments:
  - %LET MacVar = ABC;
  - %LET MacVar = ABC;
  - %LET MacVar = ABC ;
- All have same effect if you submit:
  - %put Result is LLL&MacVar.RRR;
  - In SAS Log:
  - Result is LLLABCRRR

%LET statement:

How to assign a value

with leading or trailing blanks

```
%LET MacVarA = %str(XYZ          );
%LET MacVarB = %str(          XYZ);
%put MacVarA is LLL&MacVarA.RRR;
%put MacVarB is LLL&MacVarB.RRR;
```

In SAS Log:

```
MacVarA is LLLXYZ          RRR
MacVarB is LLL          XYZRRR
```

More about that red dot . later

How To Assign Values To Macro Variables From DATA Step or PROC SQL

Assign Values to Macro Variables: Using CALL SYMPUT in DATA Step

```
data _null_;
length TallestStudent $ 8;
retain TallestStudent ' ' MaxHeight 0;
set sashelp.class end=LastOne;
if height GT MaxHeight then do;
  MaxHeight = height;
  TallestStudent = Name;
end;
if LastOne;
call symput('MaxHgt',trim(left(MaxHeight)));
call symput('Tallest',trim(left(TallestStudent)));
run;
```

Using Macro Variables Assigned By CALL SYMPUT in DATA Step

```
title "&Tallest is Tallest Student with Height = &MaxHgt inches";
proc print data=sashelp.class;
where height EQ &MaxHgt;
run;
```

In Output Window:

```
Philip is Tallest Student with Height = 72 inches
Obs     Name      Sex    Age    Height    Weight
 15    Philip      M      16      72        150
```

Using Macro Variables Assigned Without trim(left(. . .))

If in DATA _NULL_ step, you use:

```
call symput('MaxHgt',MaxHeight);
call symput('Tallest',TallestStudent);
```

In Output Window:

```
The title
Philip is Tallest Student with Height = 72 inches
INSTEAD becomes
Philip   is Tallest Student with Height =
72 inches
```

Assign Values to Macro Variables: Using PROC SQL and Select INTO

```
proc sql noprint;
select max(height) into: MaxHgt
from sashelp.class;
quit;

proc sql noprint;
select name into: Tallest
from sashelp.class
where height EQ &MaxHgt;
quit;
```

Using Macro Variables Assigned By PROC SQL and Select INTO

```
options nocenter nodate nonumber;
title "&Tallest is Tallest Student with Height = &MaxHgt
inches";
proc print data=sashelp.class;
where height EQ &MaxHgt;
run;
```

In Output Window:

```
Philip   is Tallest Student with Height =
72 inches
Obs      Name      Sex     Age     Height     Weight
 15     Philip      M       16       72          150
```

Bad title like CALL SYMPUT without trim(left(. . .))

Fix Macro Variables Assigned With %trim(%left(. . .))

If after PROC SQL and before PROC PRINT you use:

```
%let Tallest = %trim(%left(&Tallest));
%let MaxHgt  = %trim(%left(&MaxHgt));
```

In Output Window:

```
The title
Philip   is Tallest Student with Height =
72 inches
INSTEAD becomes
Philip is Tallest Student with Height = 72 inches
```

Better Way to Prepare Macro Variables: Using PROC SQL and Select INTO with TRIMMED option

```
proc sql noprint;
select max(height) into: MaxHgt trimmed
from sashelp.class; quit;

proc sql noprint;
select name into: Tallest trimmed
from sashelp.class
where height EQ &MaxHgt; quit;
```

NOTE: If you don't use the trimmed keyword, the macro variables get created with trailing blanks if character or leading blanks if numeric. Then you need to do trimming with a post-creation step as above.

How To Retrieve Values Of Macro Variables

Use & as prefix
to macro variable name
to retrieve it

Use OPTION SYMBOLGEN;
to get a message in SAS Log
about substitution of value
at point of resolution.

Use double quotes to retrieve a macro variable in a quoted string (e.g., in a TITLE statement)

%let ReportCreator = Macro Programmer X;

TITLE “Report By: &ReportCreator”;

In SAS Log:

TITLE “Report By: Macro Programmer X”;

If instead you use

TITLE ‘Report By: &ReportCreator’;

In SAS Log:

TITLE ‘Report By: &ReportCreator’;

When concatenating to end of a macro variable, use . to identify the end of the macro variable

```
%macro GetData(LibRef=,DataSet=);
data work.Got; set &LibRef..&DataSet; run;
%mend GetData;
options mprint;
%GetData(LibRef=SASHELP,DataSet=CLASS);
```

In SAS Log:

```
MPRINT(GETDATA):   data work.Got;
MPRINT(GETDATA):   set SASHELP.CLASS;
MPRINT(GETDATA):   run;
```

Macro Functions

SAS Functions perform operations on SAS variables.

Macro Functions perform operations on strings of text that result from resolution of macro variables.

Many Macro Functions are analogues of SAS Functions

%INDEX, %LENGTH, %SCAN, %SUBSTR, %UPCASE, %LEFT, %TRIM

are analogues of

INDEX, LENGTH, SCAN, SUBSTR, UPCASE, LEFT, TRIM

- Not a complete list
- These can be used anywhere:
  - inside macros
  - in open code

When the Needed Macro Function Does Not Exist: Use %SYSFUNC

```
%let LongVersionOfMyName = Le Roy;
%let ShortVersionOfMyName =
      %sysfunc(compress(&LongVersionOfMyName));
%put LongName=&LongVersionOfMyName;
%put ShortName=&ShortVersionOfMyName;
```

In SAS Log:

```
LongName=Le Roy
ShortName=LeRoy
```

Conditional Processing with Macro Language

Conditional Processing with Macro Language

Analogous to SAS Language structure

IF <some condition> THEN DO;

< some statements >

END;

ELSE DO;

<some other statements >

END;

SAS Macro Language uses

%IF, %THEN, %DO, %END, %ELSE

Macro Language conditional structures can also be used WITHIN a SAS Language statement that is inside of a macro.

Condition Wraps SAS Statements

```
%macro UseWhereStatement(Where=);
data work.FromWhereStatement;
set SASHELP.CLASS;
%if %length(&Where) NE 0 %then %do;
where &Where ;
%end;
%mend  UseWhereStatement; options mprint;
%UseWhereStatement(Where=Age EQ 13);
In SAS Log:
data work.FromWhereStatement;
set SASHELP.CLASS;
where Age EQ 13 ;
run;
```

Condition Inside SAS Statement

```
%macro UseWhereClause(Where=);
data work.FromWhereClause;
set SASHELP.CLASS
%if %length(&Where) NE 0 %then %do;
                          (where = (&Where))
%end;
  ; /* Note location of this semi-colon */
run;
%mend  UseWhereClause; options mprint;
%UseWhereClause(Where=Age EQ 13);
```

In SAS Log:

```
data work.FromWhereClause;
set SASHELP.CLASS (where = (Age EQ 13)) ;
run;
```

List-Driven Macro Processing

List-Driven Macro Processing

- Want report by Age group in sashelp.class
- Could use BY processing in PROC PRINT
- Gives one report chain of sub-reports
- But I want separate web page for each age
- List-Driven Macro Processing is the solution

List-Driven Macro Processing: The Controls

```
%DO k = 1 %TO UpperLimit %BY 1;
< SAS statements here >
%END;
```

UpperLimit is a hardcoded integer
or
a macro variable such as &LimitCount
set to an integer by predecessor processing

List-Driven Macro Processing: The List Elements & Their Use

%DO k = 1 %TO UpperLimit %BY 1;
< some SAS statements
a SAS statement that involves &&MacVar&k
more SAS statements >
%END;

Step-Wise Use of &&MacVar&k:

1. produces &MacVar1, &MacVar2, etc.
2. resolves each above into its respective value
3. resulting code is then run

Generating The List Elements: DATA Step Method

```
proc sort data=sashelp.class out=Ages nodupkey; by Age; run;
data _null_;
set Ages end=LastOne;
set sashelp.class end=LastOne;
call symput('Age'||trim(left(_N_)),trim(left(Age)));
if LastOne;
call symput('CountOfAgeValues',trim(left(_N_)));
run; %put _user_; /* trim(left( . . .)) above is NOT essential */
```

In SAS Log:

```
GLOBAL COUNTOFAGEVALUES 6
GLOBAL AGE1 11
GLOBAL AGE2 12
GLOBAL AGE3 13
Etc.
```

Generating The List Elements: PROC SQL Method

```
proc sql noprint;
select distinct(Age) , Count(Distinct(Age))
into :Age1-:Age999  , :CountOfAgeValues
from sashelp.class;
quit; /* excessively high list size 999 above is OK */
%put _user_;
```

In SAS Log:

```
GLOBAL COUNTOFAGEVALUES 6
GLOBAL AGE1 11
GLOBAL AGE2 12
GLOBAL AGE3 13
. . .
GLOBAL AGE6 16 /* Age7 to Age999 were not created */
```

List-Driven Macro Processing: Macro Structure

```
%macro WebPageByAge;
%do k = 1 %to &CountOfAgeValues %by 1;
ods html
  path="C:\#MWSUG2012 Macros\Results" (url=none)
  body="StudentInfoForAge_&&Age&k...html"
  (title="Student Info For Age &&Age&k")
  style=Styles.Minimal;
 < PROC PRINT code here >
ods html;
%end;
%mend  WebPageByAge;
```

/* Notice the three dots … in body= */

List-Driven Macro Processing: PROC PRINT Code

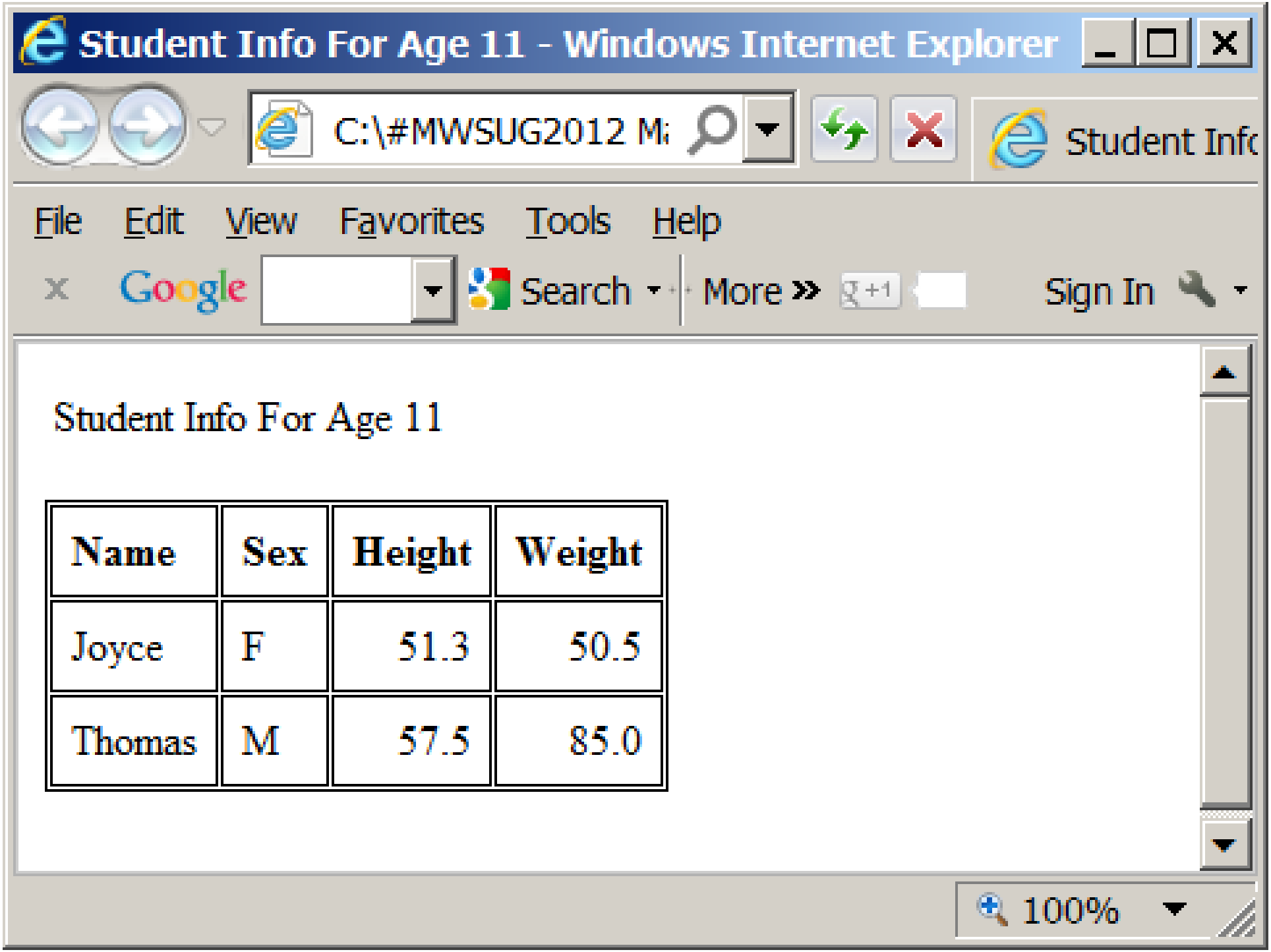
```
title "Student Info For Age &&Age&k";
proc print data=sashelp.class noobs;
where Age EQ &&Age&k;
var Name Sex Height Weight;
run;
```

List-Driven Macro Processing: Macro Invocation

```
options nocenter;
ods noresults; /* do not open output in SAS session */
ods listing close;
%WebPageByAge;
ods listing;
```

List-Driven Macro Processing SAS Log one of six parts (ages are 11 to 16)

```
ods html path="C:\#MWSUG2012 Macros\Results" (url=none)
body="StudentInfoForAge_11.html"
(title="Student Info For Age 11") style=Styles.Minimal;
NOTE: Writing HTML Body file: StudentInfoForAge_11.html
title "Student Info For Age 11";
proc print data=sashelp.class noobs;
where Age EQ 11;
var Name Sex Height Weight;
run;
NOTE: There were 2 observations read from the data set
SASHELP.CLASS.
     WHERE Age=11;
NOTE: PROCEDURE PRINT used (Total process time):
     real time          0.03 seconds
     cpu time           0.00 seconds
```


Student Info For Age 11 - Windows Internet Explorer
C:\#MWSUG2012 M
Student Info
File Edit View Favorites Tools Help
Google
Search
More »
Sign In
Student Info For Age 11
Name
Sex
Height
Weight
Joyce
F
51.3
50.5
Thomas
M
57.5
85.0
100%

Macro Functions for Arithmetic

Macro Functions for Arithmetic

Integer Arithmetic

```
%let ONE = 1;
%let TWO = 2;
%let THREE = %eval(&ONE + &TWO);
%put macro variable THREE = &THREE;
```

In SAS Log:

```
macro variable THREE = 3
```

Macro Functions for Arithmetic

Non-Integer Arithmetic

```
%let THREE = 3;
%let AreaOfCircleWithRadiusOfThree =
    %sysevalf(3.14159 * &THREE.**2);
    /* Note the dot . after &THREE */

%put AreaOfCircleWithRadiusOfThree =
        &AreaOfCircleWithRadiusOfThree;
```

In SAS Log:

macro variable
AreaOfCircleWithRadiusOfThree = 28.27431

Sharing Macros with a Macro Library

SASAUTOS:
AutoCall Macro Library

Put macro MacroName in any folder

where other users have READ access,

they can use

%INCLUDE “Folder\MacroName.sas”;

OR

OPTIONS SASAUTOS=(“Folder” SASAUTOS);

BEFORE first invocation of the macro

with %MacroName

Automatic Macro Variables

Automatic Macro Variables: SAS Assigns their Values

To get list, with current values if already set:

%put _automatic_;

In SAS Log (date & time):

AUTOMATIC SYSDATE 09SEP12

AUTOMATIC SYSDATE9 09SEP2012

AUTOMATIC SYSDAY Sunday

AUTOMATIC SYSTIME 15:19

However, these are date and time of start of SAS session or batch job, not current time.

More Automatic Macro Variables

In SAS Log (environmental information):

AUTOMATIC SYSVER 9.2

AUTOMATIC SYSSCP WIN
AUTOMATIC SYSSCPL X64_VSPRO

AUTOMATIC SYSENV FORE
(if batch, value will be BACK)

You might want your program to do things differently depending on the environment.

More Automatic Macro Variables

In SAS Log (environmental information):

AUTOMATIC SYSJOBID 7152

On Windows, Unix, Linux,
SYSJOBID is Process ID (a.k.a. "PID").

If you run SAS on a server, as a batch job or with Enterprise Guide, and are unable to stop a problem program, the server administrator will need to know your PID to kill it IF you have multiple sessions or jobs running at same time.

More Automatic Macro Variables

In SAS Log (Are things OK?):

AUTOMATIC SYSERR 0
AUTOMATIC SYSRC 0
AUTOMATIC SYSLIBRC 0
AUTOMATIC SYSFILRC 0
AUTOMATIC SYSLCKRC 0

More Automatic Macro Variables

SYSUSERID – who is running the code

SYSHOSTNAME – on which computer

Your questions, comments, and ideas about macro language are always welcome:

LeRoy Bessler PhD
Bessler Consulting and Research
Strong Smart Systems™
Visual Data Insights™
Mequon, Wisconsin, USA
Le_Roy_Bessler@wi.rr.com

Strong Smart Systems is a trademark of LeRoy Bessler PhD.