

MWSUG Paper S1-12-2013
Deanna Chyn
Anca Tilea

Data Presentation 101: **An Analyst's Perspective**

Presented at Michigan SAS User's Group
February 20, 2014

Overview

1. ODS TRACE
2. ODS TAGSETS.EXCELXP
3. PROC TRANSPOSE with ID
4. CALL SYMPUT
5. SGANNO
6. DICTIONARY.COLUMNS
7. PROC SQL SELECT INTO:

1. ODS TRACE

- o The ODS TRACE command, used with any SAS® PROC, creates a list in the Log window of all data tables the procedure is able to generate.

```
ODS TRACE ON;  
PROC REG DATA = SASHELP.CLASS;  
    MODEL WEIGHT = AGE;  
RUN;  
ODS TRACE OFF;
```

Output Added:

Name: ANOVA

Label: Analysis of Variance

Template: Stat.REG.ANOVA

Path: Reg.MODEL1.Fit.Weight.ANOVA

Output Added:

Name: ParameterEstimates

Label: Parameter Estimates

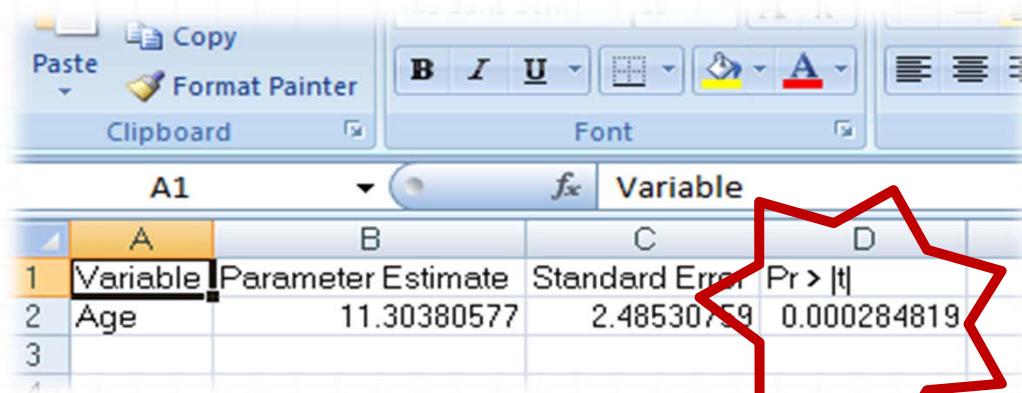
Template: Stat.REG.ParameterEstimates

Path:

Reg.MODEL1.Fit.Weight.ParameterEstimates

2. ODS TAGSETS.EXCELXP

- You want to present p-values for the coefficients from a simple linear regression as “***” or “NS” instead of their actual value.
- Using PROC FORMAT, you apply this format then create an Excel® file via PROC EXPORT



The screenshot shows the Microsoft Excel interface. The ribbon includes the Clipboard group (Paste, Copy, Format Painter) and the Font group (Bold, Italic, Underline, text color, background color). The active cell is A1, containing the text 'Variable'. The table below is a regression output with columns A, B, C, and D. Row 1 contains the headers: 'Variable', 'Parameter Estimate', 'Standard Error', and 'Pr > |t|'. Row 2 contains the data for 'Age': 'Age', '11.30380577', '2.48530759', and '0.000284819'. A red starburst shape is drawn around the p-value '0.000284819' in cell D2.

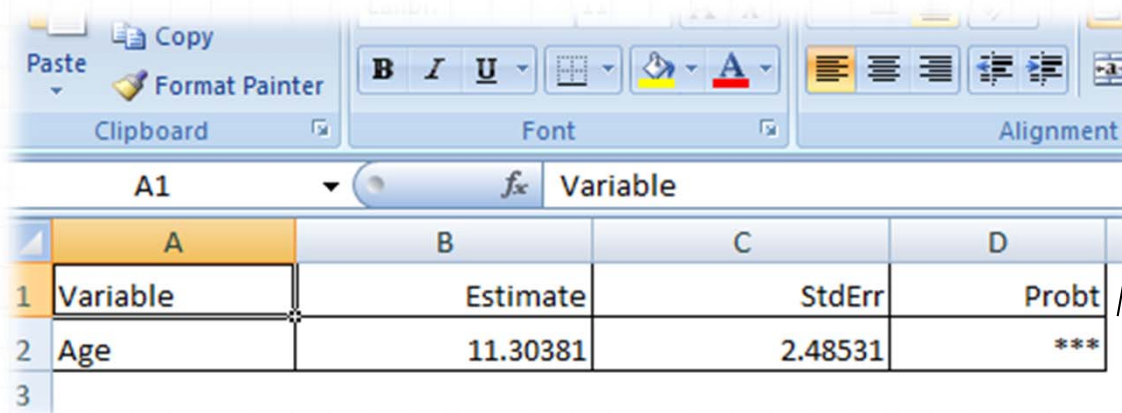
	A	B	C	D
1	Variable	Parameter Estimate	Standard Error	Pr > t
2	Age	11.30380577	2.48530759	0.000284819
3				

WAIT.

2. ODS TAGSETS.EXCELXP

```
ODS TAGSETS.EXCELXP  
  FILE = "C:\Users\Deanna\Desktop\MY_FORMATTED_FILE.XLS"  
  STYLE = MINIMAL;  
PROC PRINT DATA = estimates NOOBS;  
  VAR Variable Estimate StdErr Probt;  
  FORMAT Probt SIGF.;  
RUN;  
ODS TAGSETS.EXCELXP CLOSE;
```

Thanks, ODS!



	A	B	C	D
1	Variable	Estimate	StdErr	Probt
2	Age	11.30381	2.48531	***
3				

3. PROC TRANSPOSE with ID

- o You need to rearrange your dataset using PROC TRANSPOSE, but the basic syntax does not necessarily produce the most readable output.

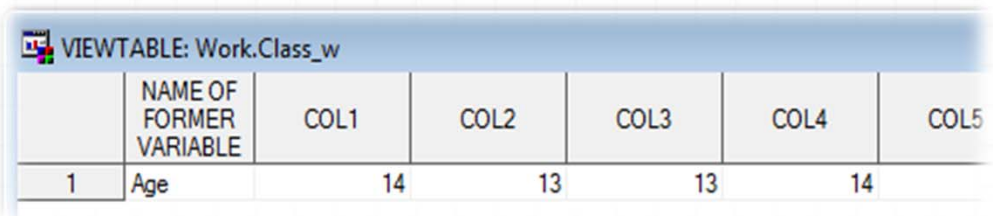
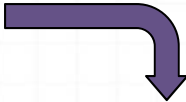


	Name	Sex	Age	Height
1	Alfred	M	14	150
2	Alice	F	13	140
3	Barbara	F	13	145
4	Carol	F	14	155
5	Henry	M	14	145
6	James	M	12	130
7	Jane	F	12	135
8	Janet	F	15	145
9	Jeffrey	M	13	140
10	John	M	12	135
11	Joyce	F	11	125
12	Judy	F	14	145
13	Louise	F	12	135
14	Mary	F	15	145
15	Philip	M	16	155
16	Robert	M	12	135
17	Ronald	M	15	145
18	Thomas	M	11	125
19	William	M	15	145



PROC TRANSPOSE

```
DATA = sashelp.class out = class_w;  
VAR Age;  
RUN;
```



	NAME OF FORMER VARIABLE	COL1	COL2	COL3	COL4	COL5
1	Age	14	13	13	14	

3. PROC TRANSPOSE with ID

- o Adding the **ID statement** to the procedure yields a more presentable table.

VIEWTABLE: Sashelp.Class (Student Data)

	Name	Sex	Age	Height
1	Alfred	M	14	160
2	Alice	F	13	145
3	Barbara	F	13	145
4	Carol	F	14	150
5	Henry	M	14	155
6	James	M	12	140
7	Jane	F	12	140
8	Janet	F	15	155
9	Jeffrey	M	13	145
10	John	M	12	140
11	Joyce	F	11	135
12	Judy	F	14	150
13	Louise	F	12	140
14	Mary	F	15	155
15	Philip	M	16	165
16	Robert	M	12	140
17	Ronald	M	15	155
18	Thomas	M	11	135
19	William	M	15	155

PROC TRANSPOSE

`DATA = sashelp.class out = class_w;`

`VAR Age;`

`ID Name;`

RUN;

VIEWTABLE: Work.Class_w

	NAME OF FORMER VARIABLE	Alfred	Alice	Barbara	Carol
1	Age	14	13	13	14

4. CALL SYMPUT

- CALL SYMPUT provides the ability to save values – numbers or character strings – into macro variables that later can be recalled

4. CALL SYMPUT

```
PROC MEANS DATA = SASHELP.CLASS;  
VAR AGE;  
CLASS SEX;  
OUTPUT OUT = MY_MEANS;  
RUN;  
DATA _NULL_;  
  SET MY_MEANS (WHERE = (_STAT_ = "N"));  
  IF SEX="F" THEN DO; CALL SYMPUT ("N_F", INPUT(AGE, 2.0)); END;  
  IF SEX="M" THEN DO; CALL SYMPUT ("N_M", INPUT(AGE, 2.0)); END;  
RUN;
```

VIEWTABLE: Work.My_means (SUMMARY STATISTICS)

	Sex	_TYPE_	_FREQ_	STAT_	Age
1	F	1	9	N	9
2	F	1	9	MIN	11
3	F	1	9	MAX	15
4	F	1	9	MEAN	13.22222222
5	F	1	9	STD	1.3944333776
6	M	1	10	N	10
7	M	1	10	MIN	11
8	M	1	10	MAX	16
9	M	1	10	MEAN	13.4
10	M	1	10	STD	1.6465452047

```
%PUT &N_F. ;          9  
%PUT &N_M. ;          10
```

Thank you, SAS. So... what now?

5. SGANNO

- One common way to display summary statistics is through charts and graphs.
- But, if there are no automatic options in the graphics procedure, how can we get the mean value we saved in CALL SYMPUT onto a nice-looking box plot?
- We get it there via the SGANNO= option.
 - **Note:** If you prefer not to use the SGs, the equivalent ANNO= option is available for PROC GPLOTS, GMAPs, BOXPLOTS, etc.
- Although we use SGANNO with a CALL SYMPUT in our example, the option can be used many ways.

5. SGANNO

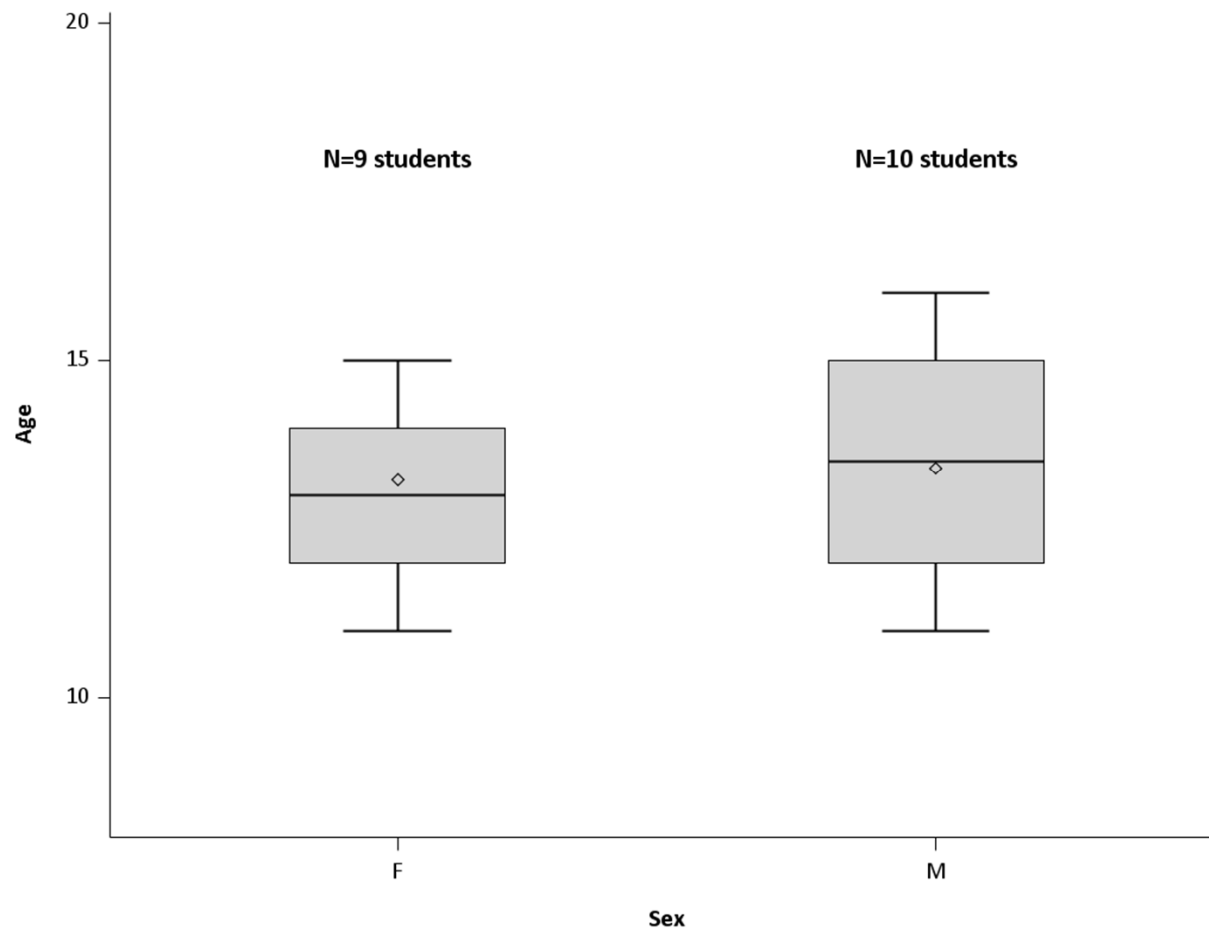
- 0 First, create a set of instructions, in the form of a SAS data set, that tells the SAS procedure where and how the data will be displayed.

```
DATA SGANNO_DSET;  
  LENGTH LABEL 30;  
  RETAIN WHEN "A" X1SPACE "DATAVALUE" Y1SPACE "DATAVALUE" TEXTFONT  
    "CALIBRI" WIDTH 40 JUSTIFY "CENTER" FUNCTION "TEXT" TEXTCOLOR  
    "BLACK" TEXTWEIGHT "BOLD" TEXTSIZE 14 Y1 18;  
  X1=0; LABEL=CAT("N=" || TRIM(LEFT(&N_F.)) || " students"); OUTPUT;  
  X1=1; LABEL=CAT("N=" || trim(left(&N_M.)) || " students"); OUTPUT;  
RUN;
```

- 0 Then, feed those instructions into the procedure.

```
PROC SGPLOT DATA=SASHELP.CLASS SGANNO=SGANNO_DSET;  
  VBOX AGE/CATEGORY=SEX ;  
  YAXIS MIN=8 MAX=20 INTEGER;  
RUN;QUIT;
```

5. SGANNO



6. DICTIONARY.COLUMNS

- During a SAS session, **DICTIONARY** tables can be used to retrieve information related to currently defined
 - libnames,
 - table names,
 - column names and attributes,
 - formats and...
 - much more.

DICTIONARY Tables and Associated SASHELP Views

DICTIONARY Table
CATALOGS
CHECK_CONSTRAINTS
COLUMNS
CONSTRAINT_COLUMN_USAGE
CONSTRAINT_TABLE_USAGE
DATAITEMS
DESTINATIONS
DICTIONARIES
ENGINES
EXTFILES
FILTERS
FORMATS
FUNCTIONS
GOPTIONS
INDEXES
INFOMAPS
LIBNAMES

6. DICTIONARY.COLUMNS

- o DICTIONARY.COLUMNS table is useful when you want to find **specific columns** to include in your analysis
- o Similar to PROC CONTENTS

Column vs. variable

VIEWTABLE: Work.Dict

	Member Name	Column Name	Column Type	Column Length	Column Position
1	CLASS	Name	char	8	24
2	CLASS	Sex	char	1	32
3	CLASS	Age	num	8	0
4	CLASS	Height	num	8	8
5	CLASS	Weight	num	8	16

mIsug.sas *

```
50 PROC SQL;
51   CREATE TABLE DICT AS
52   SELECT *
53   FROM DICTIONARY.COLUMNS
54   WHERE UPCASE(LIBNAME) = "SASHELP" &
55         UPCASE(MEMNAME) = "CLASS";
56 QUIT;
```

58 PROC CONTENTS DATA = SASHELP.CLASS OUT = I

VIEWTABLE: Work.Dict_2

	Library Member Name	Variable Name	Variable Type	Variable Length	Variable Number
1	CLASS	Age	1	8	3
2	CLASS	Height	1	8	4
3	CLASS	Name	2	8	1
4	CLASS	Sex	2	1	2
5	CLASS	Weight	1	8	5

7. PROC SQL SELECT INTO:

- How do we use DICTIONARY.COLUMNS?
- Let's look at an example...
 - What we **have**: 19 students' sex, age, height and weight over 3 years (modified *SASHELP.CLASS*)
 - What we **want**: Multiple years of data in a wide data set
 - What to **do**: Merge and rename

VIEWTABLE: Work.Class_year1						
	Name	Sex	Age	Height	Weight	
1	Alfred	M	14	69	112.5	
2	Alice	F	13	56.5	84	
3	Barbara	F	13	65.3	98	
4	Carol	F	14	62.8	102.5	
5	Henry	M	14	63.5	102.5	
6	James	M	13	60	92	
7	Jane	F	14	62	89	
8	Janet	F	15	64	85	
9	Jeffrey	M	14	60	85	
10	John	F	14	69	101	
11	Joyce	F	15	64	106	
12	Judy	M	15	64	103	
13	Louis	M	13	60	92	
14	Marjorie	F	14	62	89	

VIEWTABLE: Work.Class_year2						
	Name	Sex	Age	Height	Weight	
1	Alfred	M	15	69	118	
2	Alice	F	14	60	85	
3	Barbara	F	14	69	101	
4	Carol	F	15	64	106	
5	Henry	M	15	64	103	
6	James	M	13	60	92	
7	Jane	F	14	62	89	
8	Janet	F	15	64	85	
9	Jeffrey	M	14	60	85	
10	John	F	14	69	101	
11	Joyce	F	15	64	106	
12	Judy	M	15	64	103	
13	Louis	M	13	60	92	
14	Marjorie	F	14	62	89	

VIEWTABLE: Work.Class_year3						
	Name	Sex	Age	Height	Weight	
1	Alfred	M	16	76	121	
2	Alice	F	15	64	87	
3	Barbara	F	15	73	105	
4	Carol	F	16	71	103	
5	Henry	M	16	67	107	
6	James	M	14	59	85	
7	Jane	F	14	62	89	
8	Janet	F	15	64	85	
9	Jeffrey	M	14	60	85	
10	John	F	14	69	101	
11	Joyce	F	15	64	106	
12	Judy	M	15	64	103	
13	Louis	M	13	60	92	
14	Marjorie	F	14	62	89	

7. PROC SQL SELECT INTO:

SAS, put together: **each value of the variable
"NAME"
[from DICTIONARY.COLUMNS]...** **...followed by an
equals sign...** **...followed by the
new value for
"NAME," which
has the suffix _Y2.**

```
PROC SQL;  
SELECT CATT(NAME, "=", CATT(NAME, "_Y2"))  
INTO : VARS_Y2 SEPARATED BY " "  
FROM DICTIONARY.COLUMNS  
WHERE LIBNAME = "WORK" & MEMNAME = "CLASS_YEAR2";  
QUIT;
```

(repeat for Y3)

7. PROC SQL SELECT INTO:

- o The data step below renames the variables in the data set to their corresponding labels:

```
DATA STEP;  
  MERGE CLASS_YEAR1  
        CLASS_YEAR2(RENAME = (&VARS_Y2.))  
        CLASS_YEAR3(RENAME = (&VARS_Y3.));  
BY NAME;  
RUN;
```

- o Voila. Done. Renamed. Need more evidence? Enter a %PUT statement to see the actual values (printed to the Log window) stored in your macro variable:

```
%PUT &VARS_Y2.;
```

```
Macro variable VARS_Y2 resolves to  
Age=Age_Y2 Height=Height_Y2  
Weight=Weight_Y2
```

COMMUNITIES.SAS.COM

The screenshot displays the SAS Communities website interface. At the top, a blue navigation bar contains links for Home, a search icon, a download icon, a checkmark icon, a 'Browse' dropdown menu, and a 'Create' dropdown menu. Below this is the SAS logo with the tagline 'THE POWER TO KNOW.' and a large, faint 'Communities' watermark in the background.

A secondary navigation bar below the logo offers options: 'Browse:', 'Content' (selected), 'People', 'Places', and 'Bookmarks'. On the left side, a vertical menu lists user activity categories: 'Drafts', 'Authored', 'Participated', 'Following', 'Recently Viewed', 'Trending...', and 'All' (which is highlighted with a white arrow).

The main content area features a filter bar with tabs for 'All Content', 'Articles', 'Discussions', 'Polls', and 'Ideas'. Below these tabs is a search input field with the placeholder text 'Type to filter by text', a 'Filter by tag' button, and a 'Sort by latest activity (descending)' dropdown menu. The content list below has a 'Title' header and shows several articles, each with a question mark icon, a title, and a sub-category:

- [Import data that is long format and convert to wide format](#)
in SAS Procedures
- [Monotonic Function on Sybase Database](#)
in SAS Support Communities
- [compare two groups](#)
in SAS Procedures
- [Insert into table from a data set](#)
in SAS Support Communities
- [ODBC connections are gone after upgrade to SAS EG 5.1](#)
in SAS Enterprise Guide

COMMUNITIES.SAS.COM



question about an input statement

This question has been **Answered**.

```
"data voter;  
input Age Party : $1. (Ques1-Ques4) ($1. +1);  
datalines;  
23 D 1 1 2 2  
45 R 5 5 4 1  
67 D 2 4 3 3  
39 R 4 4 4 4  
19 D 2 1 2 1  
75 D 3 3 2 3  
57 R 4 3 4 4  
;"
```

Above is a part of code I quoted. What I cannot understand is the "+1" following "\$1." in second line. What is its function? If without it, what will be wrong?



Correct Answer

by [Ksharp](#) on Sep 29, 2011 8:11 AM

Party : \$1.

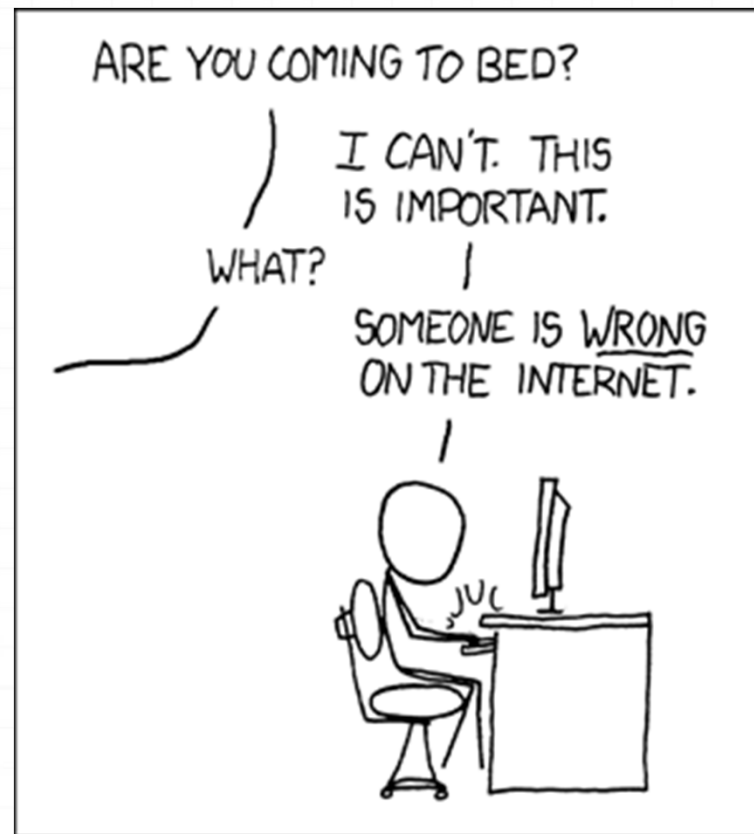
Means to eat a character, colon means to stop read when encounter a delimiter blank ,
Because it is list input method, so point will move a unit forward.

```
(Ques1-Ques4) ($1. +1);
```

is equal to Ques1 \$1. +1 Ques2 \$1. +1 Ques3 \$1. +1 Ques4 \$1. +1

Ksharp

Thanks for your time and attention



Source: www.xkcd.com