Creating Completely Automated Reports in SAS® using the ODS RTF Destination: Protect your Results from Human Error

Debra A. Goldman, MS, Memorial Sloan Kettering Cancer Center, New York, NY

ABSTRACT

Result reproducibility is an issue across disciplines, and unintentional human error is a major, but avoidable source of discrepant findings. Reporting an odds ratio of 1.9 instead of .19 can change the entire meaning of a study; having a $100,000 profit versus $1,000,000 can determine a company's solvency. Manually transferring results from SAS® Output to a report increases the probability of errors, and automating report writing is a fantastic way to prevent major errors from occurring. By combining the abilities of ODS OUTPUT, ODS RTF destination, PROC ODSTEXT, and the SAS Macro language, one can move data directly from the dataset into a complete results report, including tables and text without having to manually type out findings. This will not only prevent errors, but also save time and enhance efficiency.

This paper will demonstrate how to take results from SAS procedures, such as PROC FREQ and PROC LIFETEST, and move them into an RTF document using SAS 9.4 or later. The main focus will be to demonstrate how to insert procedure findings into report text, but I will also touch on making tables for the same numbers in PROC REPORT. The methods are applicable across disciplines, but the example will be biostatistics based. Attendees should have a basic understanding of statistical regression procedures, the ODS RTF destination, and SAS Global Macro variables.

INTRODUCTION

In an ideal world, a study sample is complete and values are locked before any analysis takes place. However, most of us do not live in this world. Patients or data points may be excluded for multiple reasons. In prospective research studies, patients may become non-evaluable as they did not take the minimum required dose or did not report back for follow up. In retrospective studies, a patient may initially meet the inclusion criteria, but additional chart review may indicate that disease status was marked incorrectly or for another reason, no longer meet the inclusion criteria. Patients may also be added as the study time period is elongated. Further, follow up data may be updated, such that all time-to-event analyses will need to be recalculated. In all of these scenarios, the report needs to be re-run. With the release of SAS 9.4, PROC ODSTEXT became available, which enables uses to write paragraphs and lists. Combining PROC ODSTEXT with the already available features of SAS Macro variables and PROC REPORT, one never needs to manually type metrics and estimates from data again. I'll cover two examples, one using descriptive statistics and another using inferential statistics.

This paper will demonstrate how to:

- View underlying tabular data from SAS procedures
- Pull data from SAS procedures
- Create macro variables from underlying tabular data
- Write in PROC ODS TEXT
- Call macro variables in PROC ODSTEXT
- Output data in PROC REPORT

EXAMPLE DATASETS

Two datasets from SASHELP will be used throughout this paper

DATASET #1: SASHELP.HEART → HEART_V2

SASHELP.HEART contains data on 5,209 patients from the Framingham Heart Study. In this paper, the data will be used to demonstrate how to pull data for a patient characteristics table, known as a Table 1, using PROC MEANS and PROC FREQ. In addition to the existing variables, height and weight were converted to meters and kilograms to calculate body mass index (BMI), and grouping of BMI into the traditional WHO categories of Underweight (<18.5), Normal (18.5-<25), Overweight (25-<30) and Obese (>30). Please see SASHELP documentation for more information.[1]

Alphabetic List of Variables and Attributes

#	Variable	Type	Len	Label
3	AgeAtStart	Num	8	Age at Start
6	BP_Status	Char	7	Blood Pressure Status
5	Chol_Status	Char	10	Cholesterol Status
4	Cholesterol	Num	8	
7	Smoking_Status	Char	17	Smoking Status
1	Status	Char	5	
10	bmi	Num	8	
11	bmi_g	Num	8	
2	gender	Char	6	
8	height_m	Num	8	
9	weight_kg	Num	8	

Output 1. PROC CONTENTS of HEART_V2

Obs	AgeAtStart	height_m	weight_kg	bmi	bmi_g	Cholesterol	Chol_Status	BP_Status
1	29	1.58750	63.5029	25.1980	Overweight	.		Normal
2	41	1.51765	87.9968	38.2053	Obese	181	Desirable	High
3	57	1.58115	59.8741	23.9493	Normal	250	High	High
4	39	1.67005	71.6675	25.6959	Overweight	242	High	Normal
5	42	1.67640	70.7604	25.1788	Overweight	281	High	Optimal

Output 2. First five data points in HEART_V2

DATASET #2: SASHELP.BMT → BMT_V2

This data comes from the paper by Klein and Moeschberger (1997)[2], which examined the difference in disease-free survival among three risk groups of leukemia patients, acute lymphoblastic leukemia (ALL), acute myelocytic leukemia- low risk (AML-Low Risk), and acute myelocytic leukemia- high risk (AML-high risk). In total, 137 were included and the set itself contains three variables: GROUP, T, and STATUS. GROUP refers to the three risk groups, T

refers to time, and status refers to the disease status at last follow up. For the purpose of this analysis, time has been converted into months for a new variable time_months. More details on this data can be found in the SASHELP guide.[1] This data will be used to illustrate PROC LIFETEST.

Alphabetic List of Variables and Attributes

#	Variable	Type	Len	Label
1	Group	Char	13	Disease Group
3	Status	Num	8	Event Indictor: 1=Event 0=Censored
2	T	Num	8	Disease-Free Survival Time
5	group_g	Num	8	
4	time_months	Num	8	

Output 3. PROC CONTENTS of BMT_V2

Obs	Group	T	Status	time_months	group_g
1	ALL	2081	0	68.4539	1
2	ALL	1602	0	52.6974	1
3	ALL	1496	0	49.2105	1
4	ALL	1462	0	48.0921	1
5	ALL	1433	0	47.1382	1

Output 4. First 5 data points in BMT_V2

INTRODUCTION TO PROC ODSTEXT

PROC ODSTEXT is an extremely powerful tool that can be employed in any output destination. It allows one to make paragraphs, lists and include many elements of text within reports. Although one can make many specific stylistic changes with the style= statement, double spacing text is not currently an option. If one wanted to double space, this would mean manually going through the report each time to double space the paragraphs. Fortunately, SAS allows for RTF syntax within the statement that will be recognized by an RTF destination. SAS online documentation has a more thorough review of the syntax for PROC ODSTEXT.[3,4] Below is an example of how to create a simple paragraph in PROC ODSTEXT.

```
proc odstext;
p "{\pard \sl480\slmult1 \fi720 This is a sample paragraph written in PROC
ODSTEXT that begins with an indent and has double spaced lines. \par}"
/style=[fontsize=11pt fontfamily=Arial ];
run;
```

SAS Syntax	RTF/SAS Function
P	Starts paragraph
\pard \par	Creates a paragraph in RTF
\sl480\slmult1	Double spaces lines
\fi720	Adds indent in the first line.
Style=	Allows one to make specific style changes to statement
Fontsize=	Sets the font size

Fontfamily=	Sets the font family

Table 1. Definitions for sample PROC ODSTEXT paragraph

This is a sample paragraph written in PROC ODSTEXT that begins with an indent and has double spaced lines.

Output 5. Sample paragraph output

EXAMPLE 1: TABLE 1 DESCRIPTIVE CHARACTERISTICS

In every study, it is critical to first describe the sample of patients, as it tells us the population we can generalize to. It also tells us how similar our sample is to previously published samples. Variables such as age, gender and BMI are typically included. However, during the course of a study, the patient sample may change as individuals are excluded or rendered non-evaluable. As a result, it is beneficial to set up both text and table syntax to automatically generate these values, so that these don't have to be manually updated each time the patient sample changes.

EXAMPLE 1A: PROC FREQ

GOAL 1: USE ODS OUTPUT TO STORE TABLES AND VALUES

The first step in making reproducible reports is to uncover the underlying tables within each procedure. For any procedure, ODS TRACE ON and ODS TRACE OFF can be placed before and after the syntax, respectively. The underlying table names will be printed in the SAS log.

SAS Syntax	Function
ODS TRACE ON / ODS TRACE OFF	Prints the underlying ODS tables in the SAS log

Table 2. Definitions for Step 1

STEP 1: RUN PROC WITH TRACE ON

```
ods trace on;
proc freq data=heart_v2;
tables gender /nocum plots=none missing;
run;

ods trace off;
```

gender	Frequency	Percent
Female	2873	55.15
Male	2336	44.85

Output 6. PROC FREQ

These are the details that appear in the log. It provides us with the name that we'll need to call, the label, the template the table is based off of, and where to find it. Each procedure labels its tables differently, and for PROC FREQ, the table is called "OneWayFreqs".

```
Output Added:
-------------
Name:        OneWayFreqs
Label:       One-Way Frequencies
Template:    Base.Freq.OneWayFreqs
Path:        Freq.Table1.OneWayFreqs
```

Log 1. Log file for tables underlying PROC FREQ

The next step is to call the table using ODS OUTPUT. The table can be named whatever one wants. This is similar to using an out= statement.

STEP 2: OUTPUT TABLE AND VIEW USING PROC PRINT

```
ods output OneWayFreqs=freq_v1;
proc freq data=heart_v2;
tables gender /nocum plots=none missing;
run;
```

SAS Syntax	Function
ODS OUTPUT	Outputs the underlying ODS table

Table 3. Definitions for Step 2

```
proc print data=freq_v1;
run;
```

Obs	Table	F_gender	gender	Frequency	Percent
1	Table gender	Female	Female	2873	55.15
2	Table gender	Male	Male	2336	44.85

Output 7. Data stored from the PROC FREQ Procedure

After viewing the specifics of the table, the next step is to set up the table data in a way that it can be called later. It's helpful to make this as general as possible so that it can be easily transferred to other variables or later implemented in macro code. With respect to a Table 1, this is also beneficial as continuous descriptive statistics can also be later stored in the same field. Here, the frequency count and the percent were combined into one variable, and the variable name was parsed out.

STEP 3: SET UP DATA AND COMBINE FIELDS

```
data freq_v2;
set freq_v1;
variable1 = gender;
combo=trim(left(frequency)) || " (" || trim(left(round(Percent,0.1))) ||
"%)";
var1_name = scan(table,2);
keep variable1 frequency percent combo var1_name ;
run;
```

SAS Syntax	Function
Trim	Reduces additional white space in the variable
Left	Left aligns the data
Round	Rounds the value to the given decimal place
Scan	Pulls the nth word from the text string. In this case, it pulls the 2nd word from the field called "Table" above

Table 4a. Definitions for Step 3

If you are using a format for a numerically coded variable, a put statement can be used to apply the format permanently, creating a character variable from the numeric.

```
data freq_v2;
set freq_v1;
variable1 = put(bmi_g, bmi_gf.);
```

```
combo=trim(left(frequency)) || " (" || trim(left(round(Percent,0.1))) ||
"%)";
var1_name = scan(table,2);
keep variable1 frequency percent combo var1_name ;
run;
```

SAS Syntax	Function
put	Converts numeric value into text using specified format

Table 4b. Definitions for modified Step 3

Tips and Tricks: Use the SAS Macro language to repeat this for multiple variables and store in a central repository.

Tips and Tricks: The put *statement can also be used to take data from a central dataset and place into an annotation dataset for automatically placing text into figures.*

GOAL 2: CREATE MACRO VARIABLES FROM OUTPUT DATASETS

Using symput (or symputx/symputn), one can take the data from the PROC FREQ output and send it to a macro variable, with the name of the variable, the level, and a prefix to mark the type of data. This prefix may be helpful if one is calling the same variable for descriptive statistics and then multiple outcomes. For instance, below "ds_" is used to indicate that these are the descriptive statistic values. More details on the symput/symputx/symputn can be found in references.[5]

STEP 4: SEND VARIABLES TO GLOBAL LOCATION USING SYMPUT

```
data _null_;
set freq_v2 ;
 call symput( "ds_" || trim(left(var1_name)) || "_" || trim(left(variable1))
,trim(left(combo)));
run;
```

SAS Syntax	Function
Call symput	Outputs the underlying ODS table. First value is the variable name and the second value is the variable value

Table 5. Definitions for Step 4

GOAL 3: CALL GLOBAL MACRO VARIABLES WITHIN PROC ODSTEXT

Although one can call variables from a specific dataset in PROC ODSTEXT using var, frequently, data will be stored in multiple tables or one may want to call data for multiple outcomes. Therefore, using the underlying dynamic or var calls within PROC ODSTEXT may not be ideal. Fortunately, the call for macro variables is simple. Similar to how one would call these in a macro, one uses the double ampersand, "&&", prior to the variable name, and the variable value will appear.

STEP 5: CALL GLOBAL MACRO VARIABLES

```
proc odstext;
p "Patient and Treatment Characteristics ^n ^n" / style=[fontsize=11pt
fontstyle=italic fontfamily=Arial ];
p "{\pard \sl480\slmult1 \fi720 The sample contained &&ds_gender_female
females and &&ds_gender_male males. \par}" /style=[fontsize=11pt
fontfamily=Arial ];
```

```
run;
```

SAS Syntax	Function
&&	Calls global macro variables
^n	Inserts space

Table 6. Definitions for Step 5

Patient and Treatment Characteristics

The sample contained 2873 (55.2%) females and 2336 (44.8%) males.

Output 8. Final Product

EXAMPLE 1B: PROC MEANS

Below is the code for how to repeat these steps for PROC MEANS. Because the details for continuous variables may require a unit designation, it can be helpful to keep the measure of centrality and measure of spread as separate variables for the text component. Also, as no levels are present here, there is no need to include an additional categorical value in the variable title.

STEP 1: RUN PROC WITH TRACE ON

```
ods trace on;
proc means data=heart_v2 n  mean median min max maxdec=1 ;
var ageatstart ;
run;
ods trace off;
```

STEP 2: OUTPUT TABLE

```
ods output Summary=means_v1(rename=(ageatstart_n=count
ageatstart_median=median ageatstart_mean=mean ageatstart_min=min
ageatstart_max=max ));
proc means data=heart_v2 n mean median min max maxdec=1 ;
var ageatstart ;
run;
```

STEP 3: SET UP DATA AND COMBINE FIELDS

```
data means_v2;
set means_v1;
var1_name = "ageatstart";
variable1="Median (range)";
combo=trim(left(put(median,5.1))) || " (" || trim(left(put(min,5.1))) || "-"
|| trim(left(put(max,5.1))) || ")";
range = " (range: " || trim(left(put(min,5.1))) || "-" ||
trim(left(put(max,5.1))) || ")";
keep variable1 var1_name  combo count median range;
run;
```

STEP 4: SEND VALUES TO MACRO VARIABLES USING SYMPUT

```
data _null_;
set means_v2 ;
 call symput( "ds_" || trim(left(var1_name)),trim(left(combo)));
```

```
 call symput( "median_" || trim(left(var1_name)) ,trim(left(median)));
 call symput( "range_" || trim(left(var1_name))  ,trim(left(range)));
run;
```

STEP 5: CALL GLOBAL MACRO VARIABLES

```
proc odstext;
p "Patient and Treatment Characteristics ^n ^n" / style=[fontsize=11pt
fontstyle=italic fontfamily=Arial ];
p "{\pard \sl480\slmult1 \fi720 The sample contained &&ds_gender_female
females and &&ds_gender_male males. The median age was &&median_ageatstart
years &&range_ageatstart.\par}" /style=[fontsize=11pt fontfamily=Arial ];
run;
```

Analysis Variable : AgeAtStart				
N	Mean	Median	Minimum	Maximum
5209	44.1	43.0	28.0	62.0

Output 9. PROC MEANS output for age

```
Output Added:
-------------
Name:       Summary
Label:      Summary statistics
Template:   base.summary
Path:       Means.Summary
-------------
```

Log 2. SAS Log for PROC MEANS

Patient and Treatment Characteristics

The sample contained 2873 (55.2%) females and 2336 (44.8%) males. The median age was 43 years (range: 28.0-62.0).

Output 10. Final product for gender and age text

GOAL 4: COMBINE DATA FROM PROC FREQ AND PROC MEANS TO MAKE A UNIFIED DATASET FOR TABLE 1

Although the results from PROC MEANS can be appended directly to PROC FREQ, given the differences in variable size, it is helpful to set up a central dataset with wider variable lengths to avoid truncation. Below is the code for a sample skeleton dataset. Both the results of PROC FREQ and PROC MEANS can easily be appended to this below.

STEP 6: CREATE TABLE SKELETON

```
data table1_combined;
length variable1 $ 50;
length var1_name $ 50;
length combo $ 50;
length range $ 30;
```

```
frequency=.;
percent=.;
variable1="";
combo="";
var1_name="";
count=.;
median=.;
range = "";
run;
```

STEP 7: APPEND TABLES TO SKELETON

```
proc append base = table1_combined data = freq_v2 force nowarn; run;
proc append base = table1_combined data = means_v2 force nowarn; run;
```

SAS Syntax	Function
Base	Base dataset
Data	Dataset to append
Force	Force append in some situations where it would normally fail
Nowarn	Suppress warnings

Table 6. Definitions for Step 7

Obs	variable1	var1_name	combo	range	frequency	percent	count	median
1								
2	Female	gender	2873 (55.2%)		2873	55.1545		
3	Male	gender	2336 (44.8%)		2336	44.8455		
4	Median (range)	ageatstart	43.0 (28.0-62.0)	(range: 28.0-62.0)			5209	43

Output 11. Results of append procedure

GOAL 5: IMPLEMENT PROC REPORT

Many others have written papers and books on PROC REPORT. Please see the references for more general information.[6,7] The code for creating a Table 1 is provided below. PROC REPORT requires an analysis or computed variable, which is not present when the combined text variable is used. Fortunately, a dummy variable can be used as a placeholder that does not print with the table. This dummy variable needs to be included in parentheses with the combined variable. Also, one can use either labeled variables using label= in a datastep, or a format as written below to provide labels for each one.

STEP 8: RUN PROC REPORT

```
proc format;
value $desc_variablef "ageatstart"="Age at Study Beginning, years"
"gender"="Gender" "bmi"="BMI" "bmi_g"="BMI (grouped)"
"cholesterol"="Cholesterol Level" "chol_status"="Cholesterol (grouped)"
"bp_status"="Blood Pressure (grouped)";
run;

proc report data=work.table1_combined nowindows headline  missing;
column var1_name variable1 (combo dummyvar) ;
define var1_name/group order=data '' format=$desc_variablef.;
```

```
define variable1/ group order=data '' style=[textalign=right];
define combo/display '' style=[textalign=center];
define dummyvar/computed noprint ;
compute dummyvar;
dummyvar=1;
endcomp;
where combo ne "";
run;
```

SAS Syntax	Function
Nowindows	Prevents report from opening in windowing environment
Headline	Underlines all column headings
Missing	Tells SAS to include rows with missing values
Column	Indicates variable columns to include
Define	Defines the variable for the report
Order	Tells SAS the order for which to print the variables
Group/display/computed	Indicates the type of variable
Noprint	Prevents variable from showing in table

Table 8. Definitions for Step 8

Factor		N (%)
Gender	Female	2873 (55.2%)
	Male	2336 (44.8%)
Age at Study Beginning, years	Median (range)	43.0 (28.0-62.0)

Output 12. Results of PROC REPORT

GOAL 6: ADD TITLES TO TEXT AND TABLES

In addition to PROC ODSTEXT, SAS has a function to add lines of text to a destination, which is particularly useful for titles and footnotes. Additionally, one can use the escape character function, ODS ESCAPECHAR, to change the style of each title to fit with one's goals. Below is the complete table for the descriptive characteristics in the Framingham Heart Study.

STEP 9: RUN PROC REPORT WITH TITLES

```
ods text="^S={fontweight=bold textdecoration=underline} Table 1. Descriptive
Characteristics";
proc report data=work.table1_all_combined nowindows headline  missing;
column var1_name variable1 (combo dummyvar) ;
define var1_name/group order=data 'Factor' format=$desc_variablef.;
define variable1/ group order=data '' style=[textalign=right];
define combo/display 'N (%)' style=[textalign=center];
define dummyvar/computed noprint ;
compute dummyvar;
dummyvar=1;
endcomp;
where combo ne "";
run;
ods text="^S={fontstyle=italic} Numbers represent frequencies with percents
in parentheses unless otherwise stated";
```

SAS Syntax	Function
ODS TEXT	Inserts a single text statement into the report
^	ODS escape character
S=	Sets style
Column	Indicates variable columns to include
Textdecoration	Provides lines under or through text
Fontstyle=italic	Sets the font style to italic

Table 9. Definitions for Step 9

Table 1. Descriptive Characteristics

Factor		N (%)
Age at Study Beginning, years	Median (range) (N=5209)	43.0 (28.0-62.0)
Gender	Female	2873 (55.2)
	Male	2336 (44.8)
BMI	Median (range) (N=5199)	25.1 (14.1-56.7)
BMI (grouped)	Unknown	10 (0.2)
	Normal	2455 (47.1)
	Overweight	1971 (37.8)
	Obese	694 (13.3)
	Underweight	79 (1.5)
Cholesterol Level	Median (range) (N=5057)	223.0 (96.0-568.0)
Cholesterol (grouped)	Unknown	152 (2.9)
	Borderline	1861 (35.7)
	High	1791 (34.4)
	Desirable	1405 (27)
Blood Pressure (grouped)	Normal	2143 (41.1)
	High	2267 (43.5)
	Optimal	799 (15.3)

Numbers represent frequencies with percents in parentheses unless otherwise stated

Output 13. Results of PROC REPORT with all variables and titles

EXAMPLE 2: SURVIVAL ESTIMATES AND COMPARISONS

The steps for storing inferential data are similar to the above steps for descriptive statistics; however, two critical differences exist. Multiple data elements may be needed and these elements may be stored in different tables. PROC LIFETEST provides the sample size, the number of events/censored, the median survival estimates, annual survival estimates, and the hypothesis test statistic and type I error. Consequently, it makes for an excellent example of how to store and call these values for macro variables and place into tables. Although the table

names and number of tables may differ, the steps for any inferential statistic PROC will be similar to PROC LIFETEST.

GOAL 1: USE ODS OUTPUT TO STORE TABLES AND VALUES

As in example 1, the first step is to use ODS TRACE to determine the tables that contain the data. In contrast to the example 1, one table that contains the confidence intervals around the annual estimates is not printed in ODS OUTPUT, and therefore, does not appear when ODS TRACE is turned on. Therefore, one needs to add the outsurv statement as listed below to obtain these additional precision estimates.

STEP 1: RUN PROC WITH TRACE ON

```
ods trace on;
proc lifetest data=bmt_v2 plots=s(atrisk(atrisktickonly maxlen=120) =(0 to
120 by 12) cl test) method=km
TIMELIST=(24) outsurv=se_tbl conftype=loglog;
time time_months*status(0);
strata group/test=logrank;
run;
ods trace off;
```

SAS Syntax	Function
OUTSURV=se_tbl	Prints the underlying ODS tables in the SAS log

Table 10. Definitions for Step 1

```
Output Added:
-------------
Name:       ProductLimitEstimates
Label:      Product-Limit Estimates
Template:   Stat.Lifetest.ProductLimitEstimates
Path:       Lifetest.Stratum1.ProductLimitEstimates
-------------

Output Added:
-------------
Name:       Quartiles
Label:      Quartiles of the Survival Distribution
Template:   Stat.Lifetest.Quartiles
Path:       Lifetest.Stratum1.TimeSummary.Quartiles
-------------

Output Added:
-------------
Name:       Means
Label:      Mean
Template:   Stat.Lifetest.Means
Path:       Lifetest.Stratum1.TimeSummary.Means
-------------

Output Added:
-------------
Name:       ProductLimitEstimates
Label:      Product-Limit Estimates
Template:   Stat.Lifetest.ProductLimitEstimates
Path:       Lifetest.Stratum2.ProductLimitEstimates
```

```
-------------

Output Added:
-------------
Name:       Quartiles
Label:      Quartiles of the Survival Distribution
Template:   Stat.Lifetest.Quartiles
Path:       Lifetest.Stratum2.TimeSummary.Quartiles
-------------

Output Added:
-------------
Name:       Means
Label:      Mean
Template:   Stat.Lifetest.Means
Path:       Lifetest.Stratum2.TimeSummary.Means
-------------

Output Added:
-------------
Name:       ProductLimitEstimates
Label:      Product-Limit Estimates
Template:   Stat.Lifetest.ProductLimitEstimates
Path:       Lifetest.Stratum3.ProductLimitEstimates
-------------

Output Added:
-------------
Name:       Quartiles
Label:      Quartiles of the Survival Distribution
Template:   Stat.Lifetest.Quartiles
Path:       Lifetest.Stratum3.TimeSummary.Quartiles
-------------

Output Added:
-------------
Name:       Means
Label:      Mean
Template:   Stat.Lifetest.Means
Path:       Lifetest.Stratum3.TimeSummary.Means
-------------

Output Added:
-------------
Name:       CensoredSummary
Label:      Censored Summary
Template:   Stat.Lifetest.CensoredSummary
Path:       Lifetest.CensoredSummary
-------------

Output Added:
-------------
Name:       HomStats
Label:      Rank Statistics
Template:   Stat.Lifetest.HomStats
Path:       Lifetest.StrataHomogeneity.HomStats
-------------
```

```
Output Added:
-------------
Name:       LogrankHomCov
Label:      Log-Rank Covariance
Template:   Stat.Lifetest.HomCov
Path:       Lifetest.StrataHomogeneity.LogrankHomCov
-------------

Output Added:
-------------
Name:       HomTests
Label:      Homogeneity Tests
Template:   Stat.Lifetest.HomTests
Path:       Lifetest.StrataHomogeneity.HomTests
-------------

Output Added:
-------------
Name:       SurvivalPlot
Label:      Survival Curves
Template:   Stat.Lifetest.Graphics.ProductLimitSurvival
Path:       Lifetest.SurvivalPlot
-------------
```

Log 3. Log file for PROC LIFETEST

As illustrated, many more tables are present with PROC LIFETEST, some of which are repeated due to the multiple strata. Fortunately, these only need to be called once. In PROC LIFETEST, the number of patients/events is stored in the censored summary, the comparison between the strata is stored in the homogeneity tests, the annual estimates are stored both in the product-limit table and the outsurv dataset noted above, and the quartile/median estimates are stored in the quartiles table. Each of these will need to be called and formatted. As in PROC FREQ, the variable that contains the categories is stored with the name of the variable. In a single variable situation, this isn't much of a problem, but if one wants to concatenate multiple variables into one dataset, it's helpful to have this as a generic name. Within the ODS OUTPUT, the variable name has been renamed "levels". To note, within a macro, one can simply refer to the macro variable name.

STEP 2: OUTPUT DATASETS

```
ods output CensoredSummary=cs_v1(rename=(failed=events group=levels))
HomTests=ht_v1
ProductLimitEstimates=ple_tbl(rename=(group=levels))
Quartiles=quart_v1(rename=(group=levels estimate=median_est
lowerlimit=median_lcl upperlimit=median_ucl));
proc lifetest data=bmt_v2 plots=s(atrisk(atrisktickonly maxlen=120) =(0 to
120 by 12) cl test) method=km
TIMELIST=(24) outsurv=se_tbl conftype=loglog;
time time_months*status(0);
strata group/test=logrank;
run;
```

STEP 3: FORMAT DATASETS

For the censored summary, create a combination of total number of patients with events for tabular display, and remove the total sample numbers.

```
data cs_v2;
set work.cs_v1;
total_events = total || " (" || trim(left(events)) || ")";
keep events total levels total_events;
where levels ne .;
run;
```

For the quartiles, keep the median estimate and create a combination variable, containing the median estimate and confidence interval. One can use if / else statements to add indicators, such as “NR” for not reached, for when the upper or lower bounds have not been reached.

```
data quart_v2;
set quart_v1;
where percent=50;
if median_lcl ne . and median_ucl ne . then median_combined =
trim(left(round(median_est,0.1))) || " months (95%CI: " ||
trim(left(round(median_lcl,0.1))) || "-" || trim(left(round(median_ucl,0.1)))
|| ")";
else if median_lcl ne . and median_ucl = . then  median_combined =
trim(left(round(median_est,0.1))) || " months (95%CI: " ||
trim(left(round(median_lcl,0.1))) || "- NR)";
else median_combined = trim(left(round(median_est,0.1))) || " months
(95%CI:NR)";
keep median_est median_lcl median_ucl median_combined levels stratum;
run;
```

The remaining data set up is more complicated, as it involves the merger of the product-limit table with the survival estimate table. In the earlier PROC LIFETEST statement, the timelist= variable provided the survival estimate closest to the given time value. For instance, specifying 24 gives the 24 month (2 year) survival estimate. Additionally, the product-limit table provides the number at risk for that given time. One can use PROC SQL to return the estimate and confidence levels for the given strata. The inner join should be on both the time and the stratum values. If unfamiliar with PROC SQL, one can use a merge statement in a datastep as an alternative.

```
proc sql;
create table seple_tbl_v1 as
select se_tbl.time_months, se_tbl.sdf_lcl, se_tbl.sdf_ucl, se_tbl.survival as
sdf_est,
ple_tbl.levels, ple_tbl.timelist, ple_tbl.failed, ple_tbl.left
from se_tbl inner join ple_tbl on ((se_tbl.time_months = ple_tbl.time_months)
and
(se_tbl.stratum=ple_tbl.stratum))
where sdf_lcl ne .;
quit;
```

Similar to the median table, a combination variable of the annual estimate with confidence intervals can be made to provide the annual estimate as a combined variable.

```
data seple_tbl_v2;
set work.seple_tbl_v1;
if sdf_lcl ne . and sdf_ucl ne . then sdf_combined =
trim(left(round(sdf_est,0.01))) || " (95%CI: " ||
trim(left(round(sdf_lcl,0.01))) || "-" || trim(left(round(sdf_ucl,0.01))) ||
")";
```

```
else if sdf_lcl ne . and sdf_ucl = . then  sdf_combined =
trim(left(round(sdf_est,0.01))) || " (95%CI: " ||
trim(left(round(sdf_lcl,0.01))) || "- NR)";
else sdf_combined = trim(left(round(sdf_est,0.01))) || " months (95%CI:NR)";

keep timelist sdf_est sdf_lcl sdf_ucl levels sdf_combined;

RUN;
```

STEP 4: COMBINE DATASETS

As the censored summary, median estimates and annual estimates are all provided by category, these can be merged using the level variable. In a second step, the p-value from the log-rank test can be added to the first variable in the set. If one wants it to display on the line for a particular level, using PROC SORT to place that particular value first will enable this.

```
proc sort data=work.cs_v2;
by levels; run;
proc sort data=work.seple_tbl_v2;
by levels; run;
proc sort data=work.quart_v2;
by levels;run;
data km_merge_v1;
merge cs_v2 seple_tbl_v2 quart_v2;
by levels ;
var1_name = "group";
run;
data km_merge_v2;
merge km_merge_v1 ht_v1;
run;
```

Obs	levels	total_events	SDF_LCL	SDF_UCL	sdf_est	Timelist	sdf_combined	STRATUM
1	ALL	38 (24)	0.20413	0.50553	0.35306	24.0000	0.35 (95% CI: 0.2-0.51)	1
2	AML-High Risk	45 (34)	0.13152	0.37594	0.24444	24.0000	0.24 (95% CI: 0.13-0.38)	2
3	AML-Low Risk	54 (25)	0.46832	0.72636	0.61111	24.0000	0.61 (95% CI: 0.47-0.73)	3

Obs	median_est	median_lcl	median_ucl	median_combined	var1_name	ProbChiSq
1	13.7500	6.3158		13.8 months (95% CI: 6.3- NR)	group	0.0010
2	6.0197	3.7171	12.8289	6 months (95% CI: 3.7-12.8)	group	
3	72.5000	21.0855		72.5 months (95% CI: 21.1- NR)	group	

Output14. Merged KM dataset

GOAL 2: CREATE MACRO VARIABLES FROM OUTPUT DATASETS

STEP 5: SEND VALUES TO MACRO VARIABLES USING SYMPUT

As the values in the variable, group, from SASHELP.BMT contained special characters that could not be appended, including spaces and dashes, the values cannot be used to create the variable name. Luckily, SAS stores a numeric stratum for each level and this can be used instead. Rather than have to look back to see which stratum was given for which category, one

can make a separate variable to indicate this, where the categorical value is stored under the stratum number. In one's own dataset and for other procedures where numeric strata are not created separately, it may be more efficient to store variables as numeric, and add in the category labels later using a put statement along with a format. On another note, the p-value only exists for the global comparison of the three levels; therefore, only one macro variable needs to be made.

```
data null;
set km_merge_v2;
call symput("level_" || trim(left(var1_name)) || trim(left(stratum)),
trim(left(levels)));
call symput("total_" || trim(left(var1_name)) || trim(left(stratum)),
trim(left(total)));
call symput("sdf_km_" || trim(left(var1_name)) || trim(left(stratum)),
trim(left(sdf_combined)));
call symput("median_km_" || trim(left(var1_name)) || trim(left(stratum)),
trim(left(median_combined)));
if probchisq ne . then call symput("pval_" ||trim(left(var1_name)),
trim(left(put(probchisq,pval_32p.))));

RUN;
```

GOAL 3: CALL GLOBAL MACRO VARIABLES WITHIN PROC ODSTEXT

STEP 7: CALL GLOBAL MACRO VARIABLES

Just as in example 1 for descriptive statistics, one uses the && to call each of the variables. Of course, if the significance or direction of the findings changes, the semantic text surrounding the values will need to be re-written.

```
proc odstext;
p "Example 2. Survival ^n ^n" / style=[fontsize=11pt fontstyle=italic
fontfamily=Arial ];
p "{\pard \sl480\slmult1 \fi720 In this study, &&total_group1 patients with
&&level_group1, &&total_group2 patients with &&level_group2,
 and &&total_group3 patients with &&level_group3 were included. Leukemia type
was associated with survival, p=&&pval_group,
 with &&level_group2 having the shortest estimated 2 year survival
&&sdf_km_group2, followed by &&sdf_km_group1 for &&level_group1
and &&sdf_km_group3 for &&level_group3. \par}" /style=[fontsize=11pt
fontfamily=Arial ];
run;
```

STEP 8: RUN PROC REPORT

Either the combined text variable or the individual values can be placed here.

```
proc format;
picture lcl (round) low-<0='0009.99 -' (prefix='[-' ) 0-high='0009.99 -'
(prefix='[' );
picture ucl (round) low-<0='0009.99]' (prefix=' -' ) 0-high='0009.99]';
value pval_fw 0-<0.05='bold' 0.05-high='light';
value pval_32p low-<0.001="<.001" .001-<0.06=[5.3] 0.06<-0.95=[5.2] 0.95<-
high=">0.95";
run;

ods text="^S={fontweight=bold} Table 2. Kaplan Meier Estimates and Log-Rank
Comparison";
proc report data=work.km_merge_v2 nowindows headline spanrows missing;
```

```
column var1_name levels ("2 year KM Est[95%CI]" sdf_est sdf_lcl sdf_ucl)
("Median Est [95%CI]" median_est median_lcl median_ucl)("p-value" probchisq);
define var1_name/ group order=external '' format = $desc_variablef.;
define levels/ group order=data '';
define sdf_est /analysis "" style=[just=r borderrightstyle=hidden
textalign=center] f=4.2;
define sdf_lcl /analysis "" style=[just=r borderleftstyle=hidden
textalign=right] f=lcl.;
define sdf_ucl /analysis "" style=[just=l borderleftstyle=hidden
textalign=left] f=ucl.;
define median_est /analysis "" style=[just=r borderrightstyle=hidden
textalign=center] f=4.2;
define median_lcl /analysis "" style=[just=r borderleftstyle=hidden
textalign=right] f=lcl.;
define median_ucl /analysis "" style=[just=l borderleftstyle=hidden
textalign=left] f=ucl.;
define probchisq / analysis "" f=pval_32p. style=[font_weight=pval_fw.
cellwidth=95 textalign=center];
where timelist ne . ;
run;
```

SAS Syntax	Function
Picture	Adds characters to a value, such as parentheses around confidence intervals
Round	Rounds the value provided within the format
value pval_fw 0-<0.05='bold' 0.05-high='light';	Creates a font weight based on specific values. Here it makes text bold when p-value is less than 0.05
Spanrows	Merges rows in table that have the same value. Here "Leukemia type" is all one cell
Cellwidth	Defines width of cell
Textalign	Aligns text
Just	Justifies text
Border(left/right)style=hidden	Hides border between cells

Table 11. Definitions for Step 8

Tips and Tricks: Variables can be titled either within the define statement or within the column statement. Placing the title in parentheses within the column statement will place the title over all variables in the parentheses

Table 2. Kaplan Meier Estimates and Log-Rank Comparison

		2 year KM Est[95%CI]			Median Est [95%CI]			p-value
Leukemia Type	ALL	0.35	[0.20 -	0.51]	13.8	[6.32 -		**0.001**
	AML-High Risk	0.24	[0.13 -	0.38]	6.02	[3.72 -	12.83]	
	AML-Low Risk	0.61	[0.47 -	0.73]	72.5	[21.09 -		

Output15. PROC REPORT output for Kaplan Meier analysis

CONCLUSION

This paper demonstrated how to take results from SAS procedures and automatically print the findings in PROC ODSTEXT and PROC REPORT within the ODS RTF definition. Multiple additional uses exist for displaying these findings, including placing these into graphs with annotation or putting into ODS POWERPOINT slides.

REFERENCES

1. SAS Institute Inc. 2016. *Sashelp Data Sets*. Cary, NC: SAS Institute Inc.
2. Klein, J. P., and Moeschberger, M. L. (1997). *Survival Analysis: Techniques for Censored and Truncated Data*. New York: Springer-Verlag.
3. SAS Institute Inc. 2016. *SAS® 9.4 Output Delivery System: Procedures Guide, Third Edition*. Cary, NC: SAS Institute Inc
4. SAS Institute Inc. 2011. SAS®*9 ODS List and Text Block Tip Sheet*: Available at: https://support.sas.com/rnd/base/ods/Tipsheet_ListTextBlks.pdf
5. SAS Institute Inc. 2014. SAS® *Macro Language 1: Essentials*. Cary, NC: SAS Institute Inc
6. Burlew ,M.M. SAS® *Guide to Report Writing: Examples, Second Edition*. Cary, NC: SAS Institute Inc.
7. SAS Institute Inc. 2016. Base *SAS® 9.4 Procedures Guide, Seventh Edition*. Cary, NC: SAS Institute Inc

ACKNOWLEDGMENTS

Thank you to Kaitlin Woo, MS and Anne S. Reiner, MPH for providing feedback for the paper

CONTACT INFORMATION

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

Debra A. Goldman
Memorial Sloan Kettering Cancer Center
485 Lexington Avenue 2nd Floor
New York, NY 10022
646-888-8331
goldmand@mskcc.org
https://www.mskcc.org/profile/debra-goldman