Paper 427-2013

Creating and Customizing the Kaplan-Meier Survival Plot in PROC LIFETEST

Warren F. Kuhfeld and Ying So, SAS Institute Inc.

ABSTRACT

If you are a medical, pharmaceutical, or life sciences researcher, you have probably analyzed time-to-event data (survival data). One of several survival analysis procedures that SAS/STAT® provides, the LIFETEST procedure computes Kaplan-Meier estimates of the survivor functions and compares survival curves between groups of patients. You can use the Kaplan-Meier plot to display the number of subjects at risk, confidence limits, equal-precision bands, Hall-Wellner bands, and homogeneity test p-value. You can control the contents of the survival plot by specifying procedure options with PROC LIFETEST. When the procedure options are insufficient, you can modify the graph templates with SAS macros. This paper provides examples of survival plot modifications using procedure options, graph template modifications using macros, and style template modifications.

INTRODUCTION

Data that measure lifetime or the length of time until the occurrence of an event are called *survival* data. Survival data are often medical data; examples include the survival time for heart transplant or cancer patients. Survival time is a measure of the duration of time until a specified event (such as relapse or death) occurs. Survival data consist of survival time and possibly a set of independent variables thought to be associated with the survival time variable. The system that gives rise to the event of interest can be biological (as for most medical data) or physical (as for engineering data). Survival analysis estimates the underlying distribution of the survival time variable and assesses the dependence of the survival time variable on the independent variables.

Standard data analysis methods are not appropriate for survival data. Survival times are generally positively skewed, and it is not reasonable to assume that data of this type have a normal distribution. Furthermore, survival times are often censored. The survival time of an individual is right censored when the event of interest has not been observed for that individual. For example, a patient who is recruited for a clinical trial drops out of the trial or the event is not observed when the period of data collection ends. In either case, the observed time is less than the true survival time. Analysis of survival data must take censoring into account and correctly use both the censored observations and the uncensored observations.

The LIFETEST procedure in SAS/STAT is a nonparametric procedure for analyzing survival data. You can use PROC LIFETEST to compute the Kaplan-Meier (1958) curve, which is a nonparametric maximum likelihood estimate of the survivor function. You can display the Kaplan-Meier plot that contains step functions representing the Kaplan-Meier curves of different samples. You can also use PROC LIFETEST to compare the survivor functions of different samples by the log-rank test.

The data that are used in this paper come from 137 bone marrow transplant patients in a study by Klein and Moeschberger (1997) and are available in the BMT data set in the Sashelp library. At the time of transplant, each patient is classified in one of three risk categories: ALL (acute lymphoblastic leukemia), AML (acute myelocytic leukemia)—Low-Risk, and AML—High-Risk. The endpoint of interest is the disease-free survival time, which is the time in days until death, relapse, or the end of the study. The variable Group represents the patient's risk category, the variable T represents the disease-free survival time, and the variable Status is the censoring indicator. A status of 1 indicates an event time, and a status of 0 indicates a censored time.

All examples use the 12.1 release of SAS software from 2012. Three types of examples are provided: specifying procedure options, modifying graph templates, and modifying style templates.

CONTROLLING THE SURVIVAL PLOT BY SPECIFYING PROCEDURE OPTIONS

This section provides a series of examples that use ODS Graphics and the PLOTS= option in the PROC LIFETEST statement to control the appearance of the survival plot. You can use the following statements to enable ODS Graphics and run PROC LIFETEST:

```
ods graphics on;
ods select survivalplot(persist) failureplot(persist);

proc lifetest data=sashelp.BMT;
   time T * Status(0);
   strata Group;
run;
```

The results are displayed in Figure 1. ODS Graphics is enabled for this step and all subsequent steps until ODS Graphics is disabled. ODS Graphics remains enabled throughout the examples in this paper. The ODS SELECT statement persistently selects just the survival and failure time plot for this and subsequent steps. Each analysis produces only one of these two plots. You specify in the TIME statement that the disease-free survival time is recorded in the variable T. You can further specify that the variable Status indicates censoring and 0 indicates a censored time. Separate survivor functions are compared for each of the groups in the Group variable, which you specify in the STRATA statement. The graph in Figure 1 consists of three step functions, one for each of the three groups of patients. The graph shows that patients in the AML—Low-Risk group have longer disease-free survival than patients in the ALL and AML—High-Risk groups.

Figure 1: Default Kaplan-Meier Plot

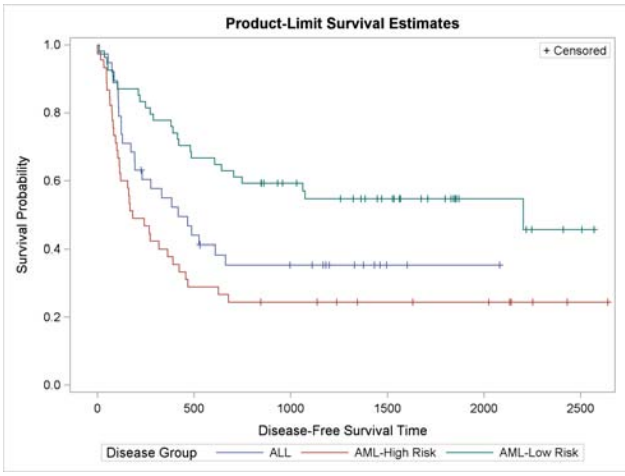


Figure 2: One of Three Individual Plots

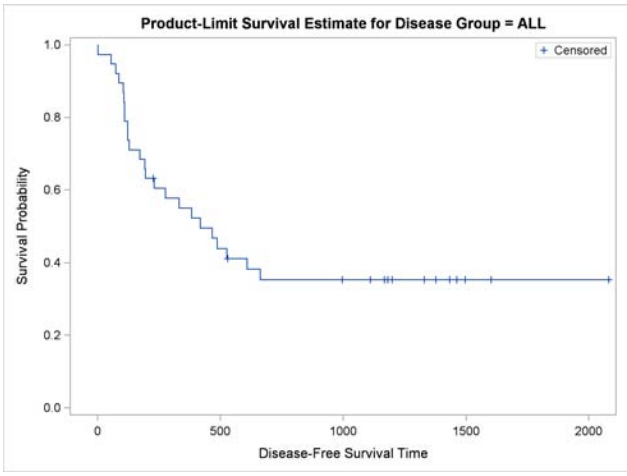


Figure 3: Individual Plots Displayed in a Panel

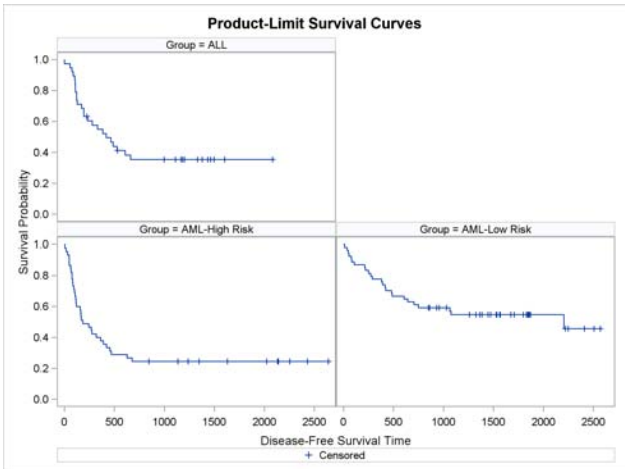
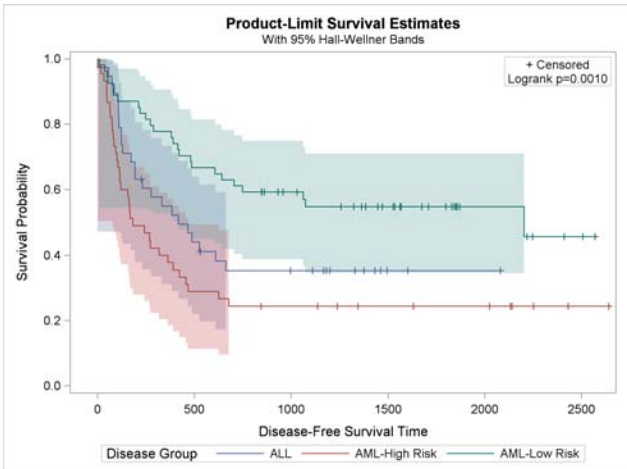


Figure 4: Confidence Bands and Homogeneity Test



The PLOTS= option enables you to control some details of the graphs. You can use it to request nondefault graphs and specify options for all graphs. For example, you can use the STRATA=INDIVIDUAL option to request individual survival plots. By default, the STRATA=OVERLAY option produces the graph of overlaid step functions displayed in Figure 1. You can run the same analysis but request the results in three separate graphs, one per patient group, as follows:

```
proc lifetest data=sashelp.BMT plots=survival(strata=individual);
   time T * Status(0);
   strata Group;
run;
```

The first of the three survival plots is displayed in Figure 2. In the interest of space, the other graphs are not displayed. You can use the STRATA=PANEL option as follows to display the results in separate panels of a single graphical display:

```
proc lifetest data=sashelp.BMT plots=survival(strata=panel);
   time T * Status(0);
   strata Group;
run;
```

The results are displayed in Figure 3.

The rest of this paper discusses overlaid plots such as the one displayed in Figure 1. You can use the following statements to add Hall-Wellner confidence bands (Hall and Wellner 1980) to Figure 1 and display the p-value from a test that the strata are homogeneous:

```
proc lifetest data=sashelp.BMT plots=survival(cb=hw test);
   time T * Status(0);
   strata Group;
run;
```

The results are displayed in Figure 4. The Hall-Wellner confidence bands extend to the last event. The small p-value supports rejecting the hypothesis that the groups are homogeneous.

Figure 5: At-Risk Table Inside the Plot

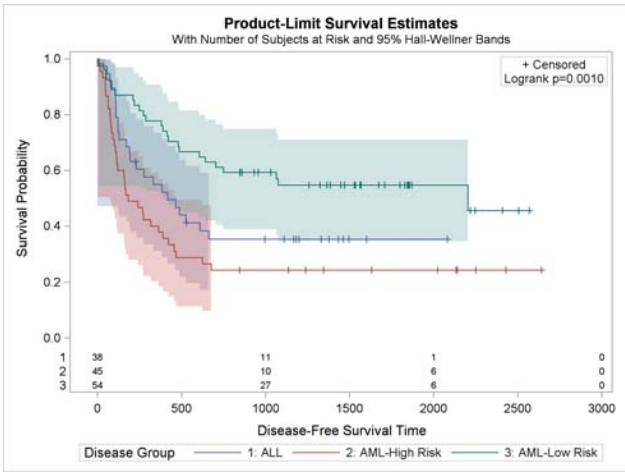
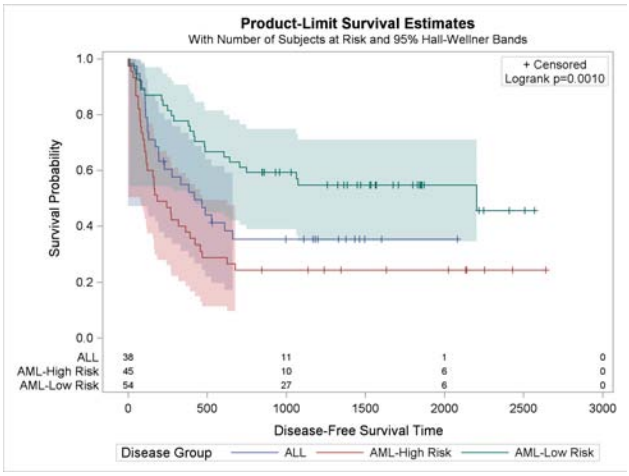


Figure 6: At-Risk Table with Labels



You can add the at-risk table as follows:

```
proc lifetest data=sashelp.BMT plots=survival(cb=hw test atrisk);
   time T * Status(0);
   strata Group;
run;
```

The results are displayed in Figure 5. By default, the at-risk table is displayed inside the body of the graph. For these data, the default survival times are 0, 1000, 2000, and 3000. You will see how to specify other values in subsequent examples. This table shows the number of patients who are at risk for each group for each of the different times.

The group labels for the at-risk table are group numbers, and these numbers appear in the legend. Numbers are used rather than the actual labels because the length of the longest label (13) is greater than the default set by the maximum label length option (MAXLEN=12). You can display labels rather than the group numbers by specifying a MAXLEN= value equal to the maximum group label length as follows:

```
proc lifetest data=sashelp.BMT plots=survival(cb=hw test atrisk(maxlen=13));
   time T * Status(0);
   strata Group;
run;
```

The results are displayed in Figure 6. The legend entries and the order of the rows in the at-risk table correspond to the sort order of the values of the Group variable. You can change the order by first mapping each value to a new value in the desired order, and then running the analysis by using a FORMAT statement to provide the original values. The following steps illustrate:

```
proc format;
   invalue bmtnum 'ALL' = 1  'AML-Low Risk' = 2  'AML-High Risk' = 3;
   value   bmtfmt 1 = 'ALL'  2 = 'AML-Low Risk'  3 = 'AML-High Risk';
run;

data BMT(drop=g);
   set sashelp.BMT(rename=(group=g));
   Group = input(g, bmtnum.);
run;

proc lifetest data=BMT plots=survival(cl test atrisk(maxlen=13));
   time T * Status(0);
   strata Group / order=internal;
   format group bmtfmt.;
run;
```

The PROC FORMAT step has two statements. The INVALUE statement creates an informat that maps the values of the original Group variable into integers that have the right order. The VALUE statement creates a format that maps the integers back to the original values. The informat is used in the DATA step to create a new integer Group variable. You specify the ORDER=INTERNAL option in the STRATA statement of PROC LIFETEST step to sort the Group values based on internal order (the order specified by the integers, which are the internal unformatted values). The FORMAT statement is used to assign the BMTFMT format to the Group variable so that the actual risk groups are displayed in the analysis. This example also illustrates the use of the CL option, which displays pointwise confidence limits for the survival curve (instead of the Hall-Wellner confidence bands). The results are displayed in Figure 7.

Figure 7: Controlling Legend Order

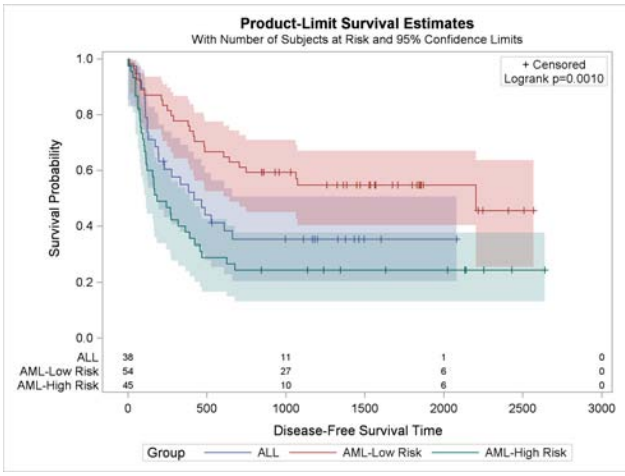


Figure 8: Failure Plot

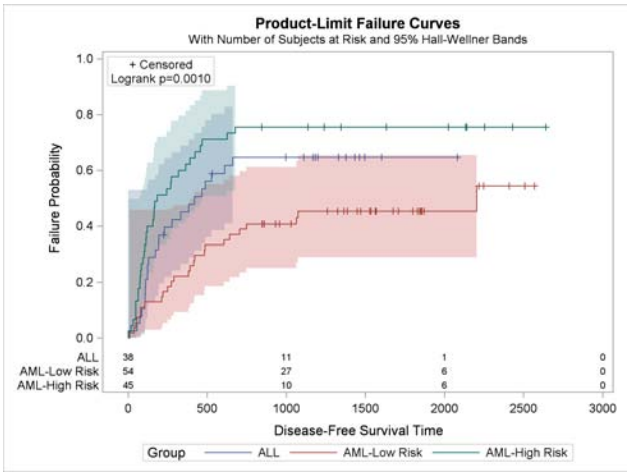


Figure 9: Censored Values Not Displayed

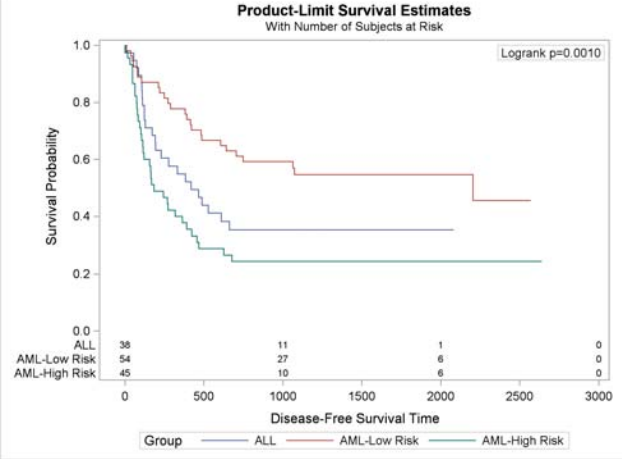
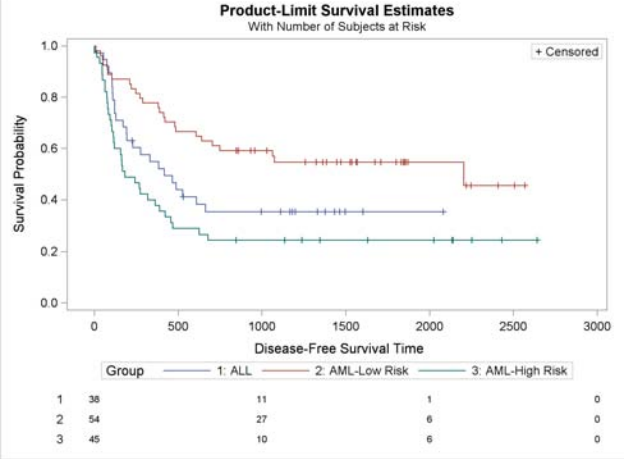


Figure 10: At-Risk Table Outside the Plot



All the discussion up to this point has been about survival plots. You can instead plot failure probabilities by using the PLOTS=SURVIVAL(FAILURE) option as follows:

```
proc lifetest data=BMT plots=survival(cb=hw failure test atrisk(maxlen=13));
   time T * Status(0);
   strata Group / order=internal;
   format group bmtfmt.;
run;
```

The results are displayed in Figure 8.

You can use the PLOTS=SURVIVAL(NOCENSOR) option to suppress the display of censored observations as follows:

```
proc lifetest data=BMT plots=survival(nocensor test atrisk(maxlen=13));
   time T * Status(0);
   strata Group / order=internal;
   format group bmtfmt.;
run;
```

The results are displayed in Figure 9.

You can use the PLOTS=SURVIVAL(OUTSIDE) option to display the at-risk table outside the body of the graph. The option OUTSIDE(0.15) reserves 15% of the vertical graph window for the at-risk table. This example illustrates that the PLOTS= option has options, options nested within options, and options nested within those nested options. The following step produces the graph in Figure 10:

```
proc lifetest data=BMT plots=survival(atrisk(outside(0.15)));
   time T * Status(0);
   strata Group / order=internal;
   format group bmtfmt.;
run;
```

Figure 11: Controlling Legend Order

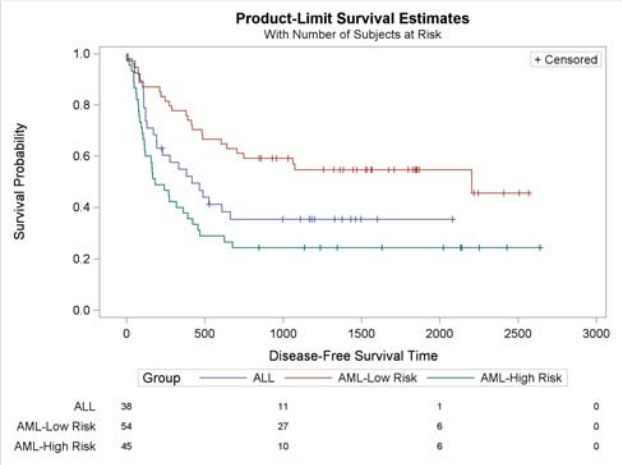


Figure 12: Specifying At-Risk Values

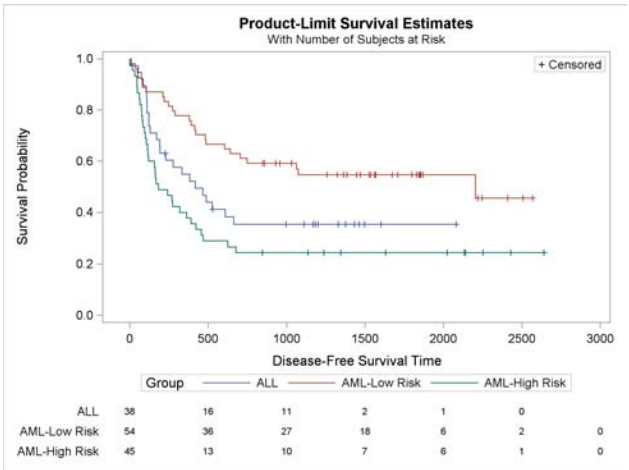
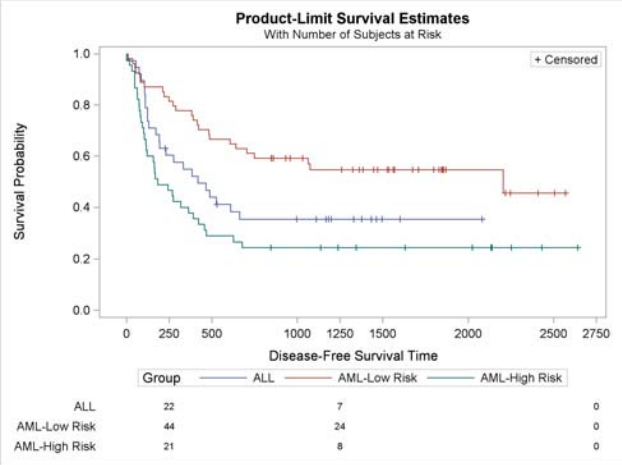


Figure 13: Controlling At-Risk Tick Marks



Because the MAXLEN= option was not specified, and because the maximum Group value length is greater than 12, integers are used to label the rows of the at-risk table, and those integers appear in the legend in Figure 10. You can specify the MAXLEN=13 option (producing the graph in Figure 11) as follows:

```
proc lifetest data=BMT plots=survival(atrisk(maxlen=13 outside(0.15)));
   time T * Status(0);
   strata Group / order=internal;
   format group bmtfmt.;
run;
```

The following step explicitly controls the time values at which the at-risk values are displayed by using the PLOTS=SURVIVAL(ATRISK=0 TO 3000 BY 500) option:

```
proc lifetest data=BMT plots=survival(atrisk(maxlen=13 outside(0.15))=0 to 3000 by 500);
   time T * Status(0);
   strata Group / order=internal;
   format group bmtfmt.;
run;
```

The results are displayed in Figure 12.

The following step uses the PLOTS=SURVIVAL(ATRISK(ATRISKTICK)) option to add tick marks to the axis that correspond to the at-risk values (250, 1250, 2750) that were specified:

```
proc lifetest data=BMT plots=survival(atrisk(atrisktick maxlen=13 outside(0.15))=250 1250 2750);
   time T * Status(0);
   strata Group / order=internal;
   format group bmtfmt.;
run;
```

The results are displayed in Figure 13.

This section illustrates the basic PLOTS= options for controlling the survival plot. If you need to make modifications that are not shown in this section, see the next section, which shows how to modify the survival plot by modifying graph templates.

CONTROLLING THE SURVIVAL PLOT BY MODIFYING GRAPH TEMPLATES

Each graph that is created when ODS Graphics is enabled is controlled by a graph template. A graph template is a SAS program, written in the Graph Template Language (GTL), that provides a detailed specification of the layout and contents of each graph. You do not need to know anything about GTL to use ODS Graphics, nor do you need to create templates. SAS provides all the templates that you need. However, you can access and modify templates to change the appearance of graphs. Typically, you ask ODS to display the template of interest, copy it into your editor, modify it, and submit it to SAS to compile. Then, when you run your procedure, it uses the new template. You see the basics of this approach next, but you do not need to use this approach with the PROC LIFETEST survival plot. The PROC LIFETEST survival plot is the only plot in SAS for which you have a different alternative available for template modification. SAS provides the survival plot templates in a series of macros and macro variables that are easier to modify than the original templates.

Because it includes the ODS TRACE ON statement, the following step displays in the SAS log the name, label, template, and path for each table and graph:

```
ods trace on;

proc lifetest data=BMT;
   time T * Status(0);
   strata Group;
run;
```

The SAS log contains the following information:

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

You can display the survival plot template by submitting the following step:

```
proc template;
   source Stat.Lifetest.Graphics.ProductLimitSurvival;
run;
```

The results of this step (not shown) consist of a 168-line SAS GTL program. This program has two major parts: one for the single-stratum case and one for the multiple-strata case. Many syntax elements appear in multiple places, and understanding and modifying this version of the template can be challenging.

The modularized version of the survival plot templates is available in the SAS sample library, which is available on the Web at http://support.sas.com/documentation/onlinedoc/stat/ex_code/121/templft2.html.[1] The following step displays the sample:

```
options ls=95;
data _null_;
   infile 'http://support.sas.com/documentation/onlinedoc/stat/ex_code/121/templft2.html'
          device=url;
   retain pre 0;
   input;
   if index(_infile_, '</pre>') then pre = 0;
   if pre then put _infile_;
   if index(_infile_, '<pre>')  then pre = 1;
run;
```

In the interest of space, the results of this step are not shown here, but all the code is displayed later. The sample consists of a single macro definition, %SurvivalTemplateRestore. This macro contains a %GLOBAL statement, 12 %LET statements, and 6 macro definitions, and it ends with a call to the %SurvivalTemplate macro, which compiles the two survival plot templates and is the first macro defined inside the %SurvivalTemplateRestore macro.[2] The %SurvivalTemplateRestore macro provides a way to restore the default macros and macro variables. **You should not modify any of the statements while they are inside the %SurvivalTemplateRestore macro.** Rather, you should

[1] This sample has been revised and no longer matches the original, which is available from http://support.sas.com/documentation/onlinedoc/stat/ex_code/121/templft.html. The revised sample simultaneously modifies two templates, one that is used when the at-risk table is inside the graph and one that is used when it is outside.

[2] You might wonder why these macros are not simply made available in the SAS autocall library. The autocall library provides macros that you can run. In this context, you do not need to simply run a macro. You need to copy it, extract parts of it, modify those parts, and submit the modified code. That is not convenient with the autocall library.

use this macro only to provide and restore all the default macros and macro variables. You should modify the individual macros and macro variables outside the context of the %SurvivalTemplateRestore macro. This will become clearer as you work through the examples. The %SurvivalTemplateRestore macro and the macros and macro variables that it provides have the following properties:

- Many options, including most of the options that are specified in multiple places in the templates, are extracted to macro variables.
- The %SurvivalTemplate macro provides the main body of the two templates, so it is easy to recompile the templates after making changes. The template **`Stat.Lifetest.Graphics.ProductLimitSurvival`** provides the survival template when the at-risk table is inside the body of the plot, and the template **`Stat.Lifetest.Graphics.ProductLimitSurvival2`** provides the survival template when the at-risk table is outside the body of the plot.[3] The templates are similar, and a macro %DO loop creates both versions.
- The table for p-values is stored in the macro %Entry_P.
- The portion of the templates for the single-stratum case is stored in the macro %SingleStratum.
- The portion of the templates for the multiple-strata case is stored in the macro %MultipleStrata.
- The macro %AtRiskLatticeStart begins the lattice that contains the graph and the table when the at-risk table is outside the body of the plot.
- The macro %AtRiskLatticeEnd ends the lattice that contains the graph and the table when the at-risk table is outside the body of plot.

This organization makes it easy to identify the relevant parts of the templates, modify these parts, and recompile the templates. A small portion of the %SurvivalTemplateRestore macro follows:

```
%macro SurvivalTemplateRestore;

   %global TitleText0 TitleText1 TitleText2 yOptions xOptions tips
           tipl groups bandopts gridopts blockopts censored censorstr;

   %let TitleText0 = METHOD " Survival Estimate";
   %let TitleText1 = &titletext0 " for " STRATUMID;
   %let TitleText2 = &titletext0 "s";         /* plural: Survival Estimates */

   %let yOptions   = label="Survival Probability" shortlabel="Survival"
                     linearopts=(viewmin=0 viewmax=1
                                 tickvaluelist=(0 .2 .4 .6 .8 1.0));

   %let xOptions   = shortlabel=XNAME offsetmin=.05
                     linearopts=(viewmax=MAXTIME tickvaluelist=XTICKVALS
                                 tickvaluefitpolicy=XTICKVALFITPOL);
   . . .

   %macro SurvivalTemplate;   . . . %mend;
   %macro entry_p;            . . . %mend;
   %macro SingleStratum;      . . . %mend;
   %macro MultipleStrata;     . . . %mend;
   %macro AtRiskLatticeStart; . . . %mend;
   %macro AtRiskLatticeEnd;   . . . %mend;
%SurvivalTemplate
%mend;
```

Before you modify anything, you must submit the %SurvivalTemplateRestore macro definition from the sample library to SAS. You can both store the code in a temporary file and submit it to SAS by submitting the following statements:

```
data _null_;
   infile 'http://support.sas.com/documentation/onlinedoc/stat/ex_code/121/templft2.html'
          device=url;
   file 'junk.junk';
   retain pre 0;
   input;
   if index(_infile_, '</pre>') then pre = 0;
   if pre then put _infile_;
   if index(_infile_, '<pre>')  then pre = 1;
run;

%inc 'junk.junk' / nosource;
```

Submitting these statements only defines the %SurvivalTemplateRestore macro. It does not make any of its component macros and macro variables available.

[3] The macros do not affect any graph that uses graph templates other than the two that are modified here. In particular, the macros do not affect the STRATA=PANEL plot that uses the template **`Stat.Lifetest.Graphics.ProductLimitSurvivalPanel`** and the failure plot that uses the template **`Stat.Lifetest.Graphics.ProductLimitFailure`**.

You can provide the default macros and macro variables by running the following step:

```
%SurvivalTemplateRestore
```

Running this step only defines the default macros and macro variables (or restores them if you have previously submitted the %SurvivalTemplateRestore macro). It also runs the %SurvivalTemplate macro and hence replaces any compiled templates that you might have created in the past. You can recompile the templates by submitting the following step:

```
%SurvivalTemplate
```

This macro runs PROC TEMPLATE and uses all the macros and macro variables in the %SurvivalTemplateRestore macro (and any that you modified). The result is two compiled templates that are stored in a special SAS data file called an item store. For more information about SAS item stores, see the section "SAS ITEM STORES" on page 20. Assuming that you have not modified your ODS path by using an ODS PATH statement, compiled templates are stored in an item store in the Sasuser library. Files in the Sasuser library persist across SAS sessions until they are deleted. When you are done with a modified template, it is wise to clean up all remnants of it by restoring the default macros and by deleting the modified templates from the Sasuser template item store. You can restore the default templates by running the following step to delete the modified templates:

```
proc template;
   delete Stat.Lifetest.Graphics.ProductLimitSurvival  / store=sasuser.templat;
   delete Stat.Lifetest.Graphics.ProductLimitSurvival2 / store=sasuser.templat;
run;
```

This step deletes the compiled templates, but it does not change any of the macros or macro variables. Only the compiled templates affect the graph when you run PROC LIFETEST.

A simple, complete program (except for the definition of the %SurvivalTemplateRestore macro from the sample library) with setup, template modifications to change the title, and cleanup works as follows:

```
                                        /* Make the macros and macro      */
%SurvivalTemplateRestore                /* variables available            */

%let TitleText0 = "Kaplan-Meier Plot";  /* Change the title.              */
%let TitleText1 = &titletext0 " for " STRATUMID;
%let TitleText2 = &titletext0;

%SurvivalTemplate                       /* Compile the templates with     */
                                        /* the new title.                 */

proc lifetest data=BMT                  /* Perform the analysis and make  */
              plots=survival(cb=hw test); /* the graph.                   */
   time T * Status(0);
   strata Group;
   format group bmtfmt.;
run;

%SurvivalTemplateRestore                /* Restore the default macros     */
                                        /* and macro variables.           */

proc template;                          /* Restore the default templates. */
   delete Stat.Lifetest.Graphics.ProductLimitSurvival  / store=sasuser.templat;
   delete Stat.Lifetest.Graphics.ProductLimitSurvival2 / store=sasuser.templat;
run;
```

The results are displayed in Figure 14. You can see that the title is now "Kaplan-Meier Plot". There are multiple title macro variables because there are two different types of plots defined in the survival plot templates. The first macro variable, TitleText0, contains the text that is the same for both types of plots. The second macro variable, TitleText1, contains the title for the single-stratum case. The third macro variable, TitleText2, contains the title for the multiple-strata case. Both TitleText1 and TitleText2 use the common text defined in TitleText0. Both TitleText0 and TitleText2 were changed from their original definition; the definition of TitleText1 was copied from the %SurvivalTemplateRestore macro. You have to provide all relevant %LET statements when you modify TitleText0. In this case it is TitleText0 and TitleText2, but it is easy to copy all three and then just modify TitleText0.

The following statements modify the default tick value list for the Y axis from the default increment of 0.2 to have an increment of 0.25 and also change the Y-axis label to "Survival":

```
%SurvivalTemplateRestore

%let yOptions = label="Survival"
                linearopts=(viewmin=0 viewmax=1
                            tickvaluelist=(0 .25 .5 .75 1));

%SurvivalTemplate
```

Figure 14: Kaplan-Meier Plot Title Modification

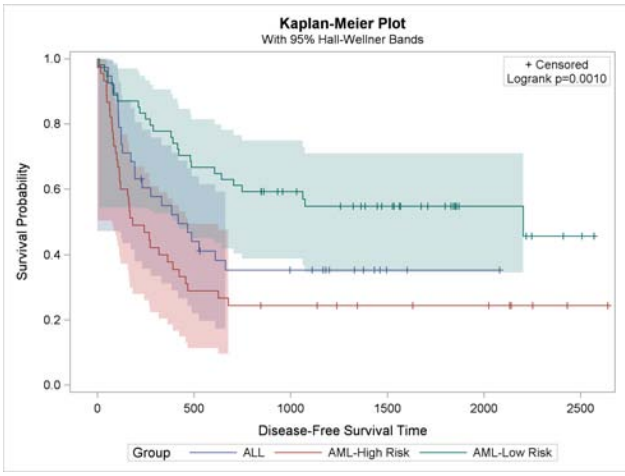


Figure 15: Y-Axis Modification

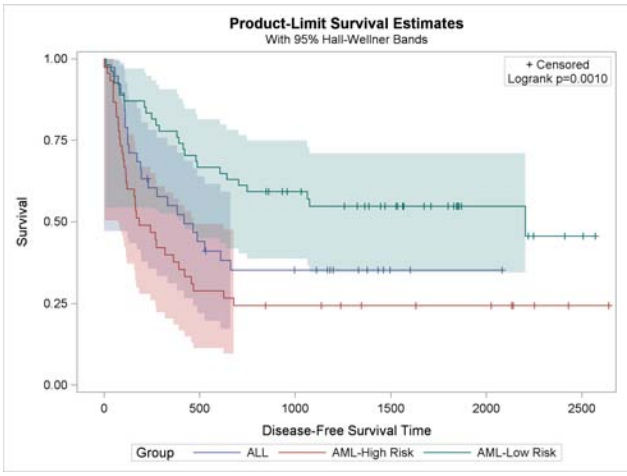


Figure 16: Y Axis, First Tick Change

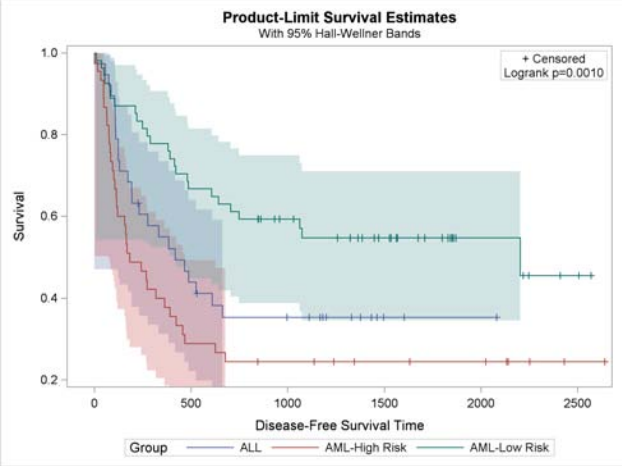
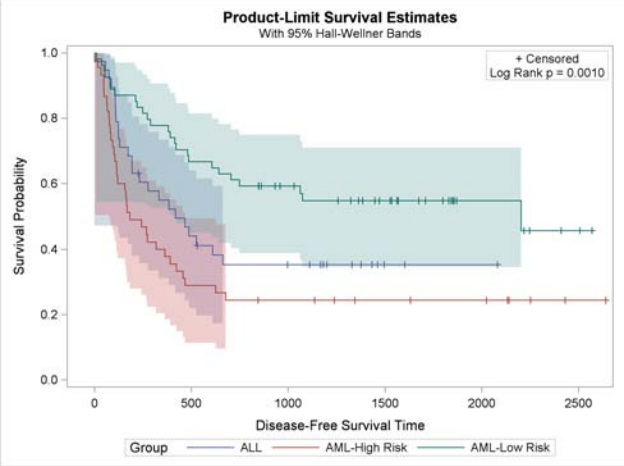


Figure 17: Cosmetic Inset Entry Change




```
proc lifetest data=BMT plots=survival(cb=hw test);
   time T * Status(0);
   strata Group;
   format group bmtfmt.;
run;
```

The results are displayed in Figure 15.

The following statements modify the default tick value list for the Y axis so that tick marks start at 0.2:

```
%SurvivalTemplateRestore

%let yOptions = label="Survival"
                linearopts=(viewmin=0.2 viewmax=1
                            tickvaluelist=(0 .2 .4 .6 .8 1.0));

%SurvivalTemplate

proc lifetest data=BMT plots=survival(cb=hw test);
   time T * Status(0);
   strata Group;
   format group bmtfmt.;
run;
```

You need to change only the value of the VIEWMIN= option, in this case from 0 to 0.2. The VIEWMIN= option controls the smallest value shown on the axis (not the smallest tick value from the tick value list). You do not need to modify the tick value list. The results are displayed in Figure 16.

The next example modifies the contents of the %Entry_P macro. The original %Entry_P macro definition is as follows:

```
%macro entry_p;
   if (PVALUE < .0001)
      entry TESTNAME " p " eval (PUT(PVALUE, PVALUE6.4));
   else
      entry TESTNAME " p=" eval (PUT(PVALUE, PVALUE6.4));
   endif;
%mend;
```

This example directly specifies the test name (replacing the internal name “Logrank” with “Log Rank”) and adds blank spaces around the equal sign.

```
%SurvivalTemplateRestore

%macro entry_p;
   if (PVALUE < .0001)
      entry "Log Rank p "   eval (PUT(PVALUE, PVALUE6.4));
   else
      entry "Log Rank p = " eval (PUT(PVALUE, PVALUE6.4));
   endif;
%mend;

%SurvivalTemplate

proc lifetest data=BMT plots=survival(cb=hw test);
   time T * Status(0);
   strata Group;
   format group bmtfmt.;
run;
```

The results are displayed in Figure 17 on the preceding page.

Because this template modification replaces a string that is more appropriately set by PROC LIFETEST, you should clean up afterward as follows:

```
%SurvivalTemplateRestore

proc template;
   delete Stat.Lifetest.Graphics.ProductLimitSurvival  / store=sasuser.templat;
   delete Stat.Lifetest.Graphics.ProductLimitSurvival2 / store=sasuser.templat;
run;
```

The macros and macro variables are designed so that most of the time you need to modify only the macro variables and not the macros. However, you have the full flexibility to modify both. You can modify any of the following macro variables:

```
%let TitleText0 = METHOD " Survival Estimate";
%let TitleText1 = &titletext0 " for " STRATUMID;
%let TitleText2 = &titletext0 "s";         /* plural: Survival Estimates */

%let yOptions   = label="Survival Probability" shortlabel="Survival"
                  linearopts=(viewmin=0 viewmax=1
                              tickvaluelist=(0 .2 .4 .6 .8 1.0));

%let xOptions   = shortlabel=XNAME offsetmin=.05
                  linearopts=(viewmax=MAXTIME tickvaluelist=XTICKVALS
                              tickvaluefitpolicy=XTICKVALFITPOL);

%let tips       = rolename=(_tip1= ATRISK _tip2=EVENT)
                  tiplabel=(_tip1="Number at Risk" _tip2="Observed Events")
                  tip=(x y _tip1 _tip2);
%let tipl       = tiplabel=(y="Survival Probability");

%let groups     = group=STRATUM index=STRATUMNUM;

%let bandopts   = &groups modelname="Survival";

%let gridopts   = autoalign=(TOPRIGHT BOTTOMLEFT TOP BOTTOM)
                  border=true BackgroundColor=GraphWalls:Color Opaque=true;

%let blockopts  = repeatedvalues=true display=(label values) valuehalign=start
                  valuefitpolicy=truncate labelposition=left
                  labelattrs=GRAPHVALUETEXT valueattrs=GRAPHDATATEXT(size=7pt)
                  includemissingclass=false;
```

```
%let censored   = markerattrs=(symbol=plus);
%let censorstr  = "+ Censored";
```

These macro variables specify a variety of GTL options. You should consult the *SAS/GRAPH: Graph Template Language Reference* for more details. The macro variables are briefly explained next:

TitleText0 provides the common text that is used in the title for the single-stratum and multiple-strata cases. METHOD is a dynamic variable that PROC LIFETEST sets. In these examples, the value of METHOD is "Product-Limit"; the product limit method is also known as the Kaplan-Meier (1958) method.

TitleText1 provides the text for the single-stratum title (relying on TitleText0).

TitleText2 provides the text for the multiple-strata title (relying on TitleText0).

yOptions provides the Y-axis options. The LABEL= option provides the axis label. The SHORTLABEL= option provides the axis label for small plots when the LABEL= label is too long. The LINEAROPTS= option is used to specify linear axis options. This and most other axes are linear axes. Alternatives include log-scale axes. The VIEWMIN=0 and VIEWMAX=1 options ensure that the axis goes from 0 to 1 even when the actual results have a more restricted range. The TICKVALUELIST= option provides the tick values. Standard SAS number list abbreviations like 0 TO 1 BY 0.2 are not valid in the GTL.

xOptions provides the X-axis options. The LABEL= option is not provided, so the axis label comes from the column label in the ODS data object. The SHORTLABEL= option provides the axis label for small plots when the label is too long. The short label comes from a dynamic variable provided by PROC LIFETEST. The OFFSETMIN= option ensures that there is extra space between the axis and the minimum tick mark. The LINEAROPTS= option is used to specify linear axis options. The VIEWMAX= option ensures that the axis goes to the value in the MAXTIME dynamic variable set by PROC LIFETEST. The TICKVALUELIST= option provides the tick values in a dynamic variable. The TICKVALUEFITPOLICY= option provides in a dynamic variable the approach for handling dense tick marks. Approaches include rotation, staggering, and thinning.

tips provides options for tooltips for the step plots. Tooltips are text boxes that appear in HTML output when you rest your mouse pointer over part of the plot when the IMAGEMAP=ON option is specified in the ODS GRAPHICS statement. Tooltips are provided for the X- and Y-axis columns. Additional columns that are assigned roles (and hence are eligible to use as tooltips) include the at-risk and event columns. These columns are given the tooltip labels "Number at Risk" and "Observed Events". Unless you are specifically interested in tooltips, you probably do not need to modify this macro variable.

tipl provides a label for the Y-axis tooltip. Unless you are specifically interested in tooltips, you do not need to modify this macro variable.

groups provides the name of the data object columns that provide group names and the index that provides the order of the group names. You will probably never need to modify this macro variable.

bandopts provides the group information for band plots. You will probably never need to modify this macro variable.

gridopts provides options for the inset table that provides the censored value legend and the homogeneity test p-value. The AUTOALIGN= option provides the places in the graph where the inset table can be positioned. If your preferred placement is somewhere other than the top right corner, you can modify the automatic placement list. The BORDER= option displays a border around this table. The BACKGROUNDCOLOR= option controls the table background. By default, it matches the background color for the rest of the graph by using the **GraphWalls:Color** style reference. The OPAQUE=TRUE option specifies an opaque table that hides any graphical elements that are behind the table.

blockopts provides options for the block plot, which produces the at-risk table. The REPEATEDVALUES=TRUE option enables repeated identical values to create separate blocks. The option DISPLAY=(LABEL VALUES) limits the display to labels and values. VALUEHALIGN=START center-aligns values at the starting value in the block. VALUEFITPOLICY=TRUNCATE truncates any values that do not fit. LABELPOSITION=LEFT places labels to the left of the strip of block values. LABELATTRS=GRAPHVALUETEXT specifies that label attributes come from the GraphValueText style element. VALUEATTRS=GRAPHDATATEXT(SIZE=7PT) specifies that value attributes come from the **GraphDataText** style element, but with a font size of seven points. INCLUDEMISSINGCLASS=FALSE enables the display of missing CLASS levels. You will probably never need to modify this macro variable.

censored provides the marker (a plus sign) that is displayed in the graph to indicate censored observations.

censorstr provides the string for the inset table that shows how censored observations appear in the plot.

The examples and information up to this point have illustrated how you can make minor changes to the survival plot. It is unlikely that you will ever have to do more than that. If you need to make changes to the overall layout of the graph, then you will need to modify one of the other macros. The %SurvivalTemplate macro, which is the macro that compiles all the pieces that you modified, is as follows:

```
%macro SurvivalTemplate;
   %local outside;
   proc template;
     %do outside = 0 %to 1;
       define statgraph Stat.Lifetest.Graphics.ProductLimitSurvival%scan(2,2-&outside);
          dynamic NStrata xName plotAtRisk %if %nrbquote(&censored) ne %then plotCensored;
             plotCL plotHW plotEP labelCL labelHW labelEP maxTime xtickVals xtickValFitPol
             rowWeights method StratumID classAtRisk plotBand plotTest GroupName yMin
             Transparency SecondTitle TestName pValue _byline_ _bytitle_ _byfootnote_;
          BeginGraph;

          if (NSTRATA=1)
             if (EXISTS(STRATUMID)) entrytitle &titletext1;
             else                   entrytitle &titletext0;
             endif;

             %if not &outside %then if (PLOTATRISK);
                entrytitle "With Number of Subjects at Risk" / textattrs=GRAPHVALUETEXT;
             %if not &outside %then %do; endif; %end;

             %AtRiskLatticeStart
             layout overlay / xaxisopts=(&xoptions) yaxisopts=(&yoptions);
                %singlestratum
             endlayout;
             %AtRiskLatticeEnd

          else
             entrytitle &titletext2;
             if (EXISTS(SECONDTITLE))
                entrytitle SECONDTITLE / textattrs=GRAPHVALUETEXT;
             endif;

             %AtRiskLatticeStart
             layout overlay / xaxisopts=(&xoptions) yaxisopts=(&yoptions);
                %multiplestrata
             endlayout;
             %AtRiskLatticeEnd(%str(class=CLASSATRISK))

          endif;

          if (_BYTITLE_) entrytitle _BYLINE_ / textattrs=GRAPHVALUETEXT;
          else if (_BYFOOTNOTE_) entryfootnote halign=left _BYLINE_; endif;
          endif;
          EndGraph;
       end;
     %end;
   run;
%mend;
```

The macro %DO loop compiles two templates: Stat.Lifetest.Graphics.ProductLimitSurvival (when the macro variable Outside is 0 and %SCAN(2,2-&OUTSIDE) is null) and Stat.Lifetest.Graphics.ProductLimitSurvival2 (when Outside is 1 and %SCAN(2,2-&OUTSIDE) is 2). The primary difference between these templates is that when the macro variable Outside is 1, a layout lattice is used to place the at-risk table outside the graph. When Outside is 1, the macros %AtRiskLatticeStart and %AtRiskLatticeEnd provide the layout lattice and the layout overlay for the at-risk table. The %AtRiskLatticeStart and %AtRiskLatticeEnd macros are defined as follows:

```
%macro AtRiskLatticeStart;
   %if &outside %then %do;
      layout lattice / rows=2 rowweights=ROWWEIGHTS
                       columndatarange=union rowgutter=10;
      cell;
   %end;
%mend;
```

```
%macro AtRiskLatticeEnd(class);
   %if &outside %then %do;
      endcell;
      cell;
         layout overlay / walldisplay=none xaxisopts=(display=none);
            blockplot x=TATRISK block=ATRISK / &class &blockopts;
         endlayout;
      endcell;
   endlayout;
   %end;
%mend;
```

The %SurvivalTemplate macro relies on two other macros: %SingleStratum for the single-stratum case, and %MultipleStrata for the multiple-strata case. The %SingleStratum macro is as follows:

```
%macro SingleStratum;
   if (PLOTHW=1 AND PLOTEP=0)
      bandplot LimitUpper=HW_UCL LimitLower=HW_LCL x=TIME /
         modelname="Survival" fillattrs=GRAPHCONFIDENCE
         name="HW" legendlabel=LABELHW;
   endif;
   if (PLOTHW=0 AND PLOTEP=1)
      bandplot LimitUpper=EP_UCL LimitLower=EP_LCL x=TIME /
         modelname="Survival" fillattrs=GRAPHCONFIDENCE
         name="EP" legendlabel=LABELEP;
   endif;
   if (PLOTHW=1 AND PLOTEP=1)
      bandplot LimitUpper=HW_UCL LimitLower=HW_LCL x=TIME /
         modelname="Survival" fillattrs=GRAPHDATA1 datatransparency=.55
         name="HW" legendlabel=LABELHW;
      bandplot LimitUpper=EP_UCL LimitLower=EP_LCL x=TIME /
         modelname="Survival" fillattrs=GRAPHDATA2
         datatransparency=.55 name="EP" legendlabel=LABELEP;
   endif;
   if (PLOTCL=1)
      if (PLOTHW=1 OR PLOTEP=1)
         bandplot LimitUpper=SDF_UCL LimitLower=SDF_LCL x=TIME /
            modelname="Survival" display=(outline)
            outlineattrs=GRAPHPREDICTIONLIMITS name="CL" legendlabel=LABELCL;
      else
         bandplot LimitUpper=SDF_UCL LimitLower=SDF_LCL x=TIME /
            modelname="Survival" fillattrs=GRAPHCONFIDENCE name="CL"
            legendlabel=LABELCL;
      endif;
   endif;

   stepplot y=SURVIVAL x=TIME / name="Survival" &tips legendlabel="Survival";

   if (PLOTCENSORED=1)
      scatterplot y=CENSORED x=TIME / &censored &tipl
         name="Censored" legendlabel="Censored";
   endif;

   if (PLOTCL=1 OR PLOTHW=1 OR PLOTEP=1)
      discretelegend "Censored" "CL" "HW" "EP" / location=outside
         halign=center;
   else
      if (PLOTCENSORED=1)
         discretelegend "Censored" / location=inside
                                     autoalign=(topright bottomleft);
      endif;
   endif;
   %if not &outside %then %do;
      if (PLOTATRISK=1)
         innermargin / align=bottom;
            blockplot x=TATRISK block=ATRISK / &blockopts;
         endinnermargin;
      endif;
   %end;
%mend;
```

The %MultipleStrata macro is as follows:

```
%macro MultipleStrata;
  if (PLOTHW)
      bandplot LimitUpper=HW_UCL LimitLower=HW_LCL x=TIME / &bandopts
               datatransparency=Transparency;
   endif;
   if (PLOTEP)
      bandplot LimitUpper=EP_UCL LimitLower=EP_LCL x=TIME / &bandopts
               datatransparency=Transparency;
   endif;
   if (PLOTCL)
      if (PLOTBAND)
         bandplot LimitUpper=SDF_UCL LimitLower=SDF_LCL x=TIME / &bandopts
                  display=(outline);
      else
         bandplot LimitUpper=SDF_UCL LimitLower=SDF_LCL x=TIME / &bandopts
                  datatransparency=Transparency;
      endif;
   endif;

   stepplot y=SURVIVAL x=TIME / &groups name="Survival" &tips;

   if (PLOTCENSORED)
      scatterplot y=CENSORED x=TIME / &groups &tipl &censored;
   endif;

   %if not &outside  %then %do;
      if (PLOTATRISK=1)
         innermargin / align=bottom;
            blockplot x=TATRISK block=ATRISK / class=CLASSATRISK &blockopts;
         endinnermargin;
      endif;
      %end;

   DiscreteLegend "Survival" / title=GROUPNAME location=outside;

   if (PLOTCENSORED)
      if (PLOTTEST)
         layout gridded / rows=2 &gridopts;
            entry &censorstr;
            %entry_p
         endlayout;
      else
         layout gridded / rows=1 &gridopts;
            entry &censorstr;
         endlayout;
      endif;
   else
      if (PLOTTEST)
         layout gridded / rows=1 &gridopts;
            %entry_p
         endlayout;
      endif;
   endif;
%mend;
```

The next example shows how you can modify one of the larger macros. You can add a horizontal reference line to the survival plot by modifying either the %SurvivalTemplate or %MultipleStrata macro. In both cases, the goal is to add the following statement to the multiple-strata case:

```
referenceline y=0.5;
```

Statements are drawn in the order in which they appear in the template. Hence, you want the reference line drawn first so that when it coincides with other graphical elements, the other elements are displayed. You can add the REFERENCELINE statement either to the top of the %MultipleStrata macro or to the %SurvivalTemplate macro just before the %MultipleStrata macro is called. You can add the REFERENCELINE statement to both the single-stratum and multiple-strata cases as follows (look for the long arrow comments):

```
%SurvivalTemplateRestore

%macro SurvivalTemplate;
   %local outside;
   proc template;
     %do outside = 0 %to 1;
       define statgraph Stat.Lifetest.Graphics.ProductLimitSurvival%scan(2,2-&outside);
          dynamic NStrata xName plotAtRisk %if %nrbquote(&censored) ne %then plotCensored;
             plotCL plotHW plotEP labelCL labelHW labelEP maxTime xtickVals xtickValFitPol
             rowWeights method StratumID classAtRisk plotBand plotTest GroupName yMin
             Transparency SecondTitle TestName pValue _byline_ _bytitle_ _byfootnote_;
          BeginGraph;

          if (NSTRATA=1)
             if (EXISTS(STRATUMID)) entrytitle &titletext1;
             else                   entrytitle &titletext0;
             endif;

             %if not &outside %then if (PLOTATRISK);
                entrytitle "With Number of Subjects at Risk" / textattrs=GRAPHVALUETEXT;
             %if not &outside %then %do; endif; %end;

             %AtRiskLatticeStart
             layout overlay / xaxisopts=(&xoptions) yaxisopts=(&yoptions);
                referenceline y=0.5; *<==============================================<<<<<<<<<<<<<;
                %singlestratum
             endlayout;
             %AtRiskLatticeEnd

          else
             entrytitle &titletext2;
             if (EXISTS(SECONDTITLE))
                entrytitle SECONDTITLE / textattrs=GRAPHVALUETEXT;
             endif;

             %AtRiskLatticeStart
             layout overlay / xaxisopts=(&xoptions) yaxisopts=(&yoptions);
                referenceline y=0.5; *<==============================================<<<<<<<<<<<<<;
                %multiplestrata
             endlayout;
             %AtRiskLatticeEnd(%str(class=CLASSATRISK))

          endif;

          if (_BYTITLE_) entrytitle _BYLINE_ / textattrs=GRAPHVALUETEXT;
          else if (_BYFOOTNOTE_) entryfootnote halign=left _BYLINE_; endif;
          endif;
          EndGraph;
       end;
     %end;
   run;
%mend;
```

Figure 18: Horizontal Reference Line

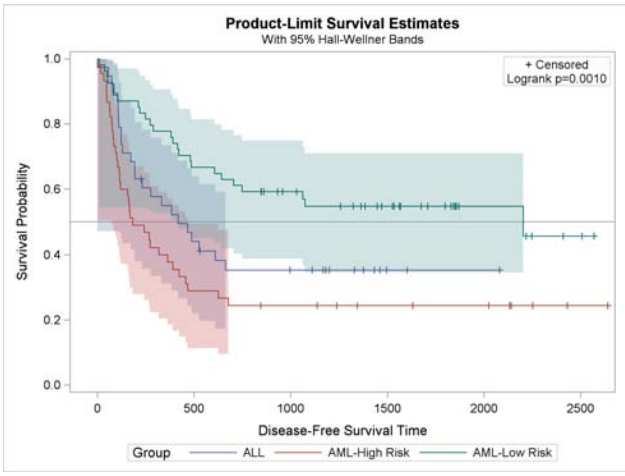
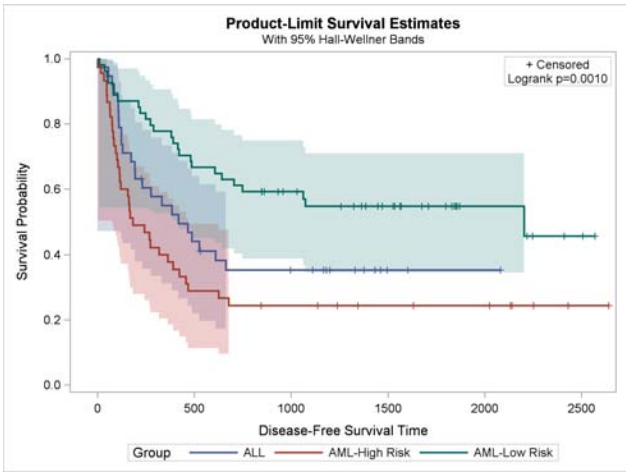


Figure 19: Changing Line Thickness



The following step creates the graph in Figure 18:

```
%SurvivalTemplate

proc lifetest data=BMT plots=survival(cb=hw test);
   time T * Status(0);
   strata Group;
   format group bmtfmt.;
run;
```

The next steps modify the line thickness for the step functions in the survival plot. The STEPPLOT statements, in their original forms, are as follows:

```
stepplot y=SURVIVAL x=TIME / name="Survival" &tips legendlabel="Survival";
stepplot y=SURVIVAL x=TIME / &groups name="Survival" &tips;
```

You can modify these statements in two different ways. You can add the LINEATTRS=(THICKNESS=2.5) option to the STEPPLOT statement (after the slash) in both the %SingleStratum and %MultipleStrata macros. Or instead you can add the option to the Tips (step plot tips) macro variable, which both statements use, as follows:

```
%SurvivalTemplateRestore

%let tips = rolename=(_tip1= ATRISK _tip2=EVENT)
            tiplabel=(_tip1="Number at Risk" _tip2="Observed Events")
            tip=(x y _tip1 _tip2) lineattrs=(thickness=2.5);

%SurvivalTemplate

proc lifetest data=BMT plots=survival(cb=hw test);
   time T * Status(0);
   strata Group;
   format group bmtfmt.;
run;
```

The results are displayed in Figure 19.

The modified templates are deleted by using the following step:

```
proc template;
   delete Stat.Lifetest.Graphics.ProductLimitSurvival  / store=sasuser.templat;
   delete Stat.Lifetest.Graphics.ProductLimitSurvival2 / store=sasuser.templat;
run;
```

Figure 20: Line and Color Group Differentiation

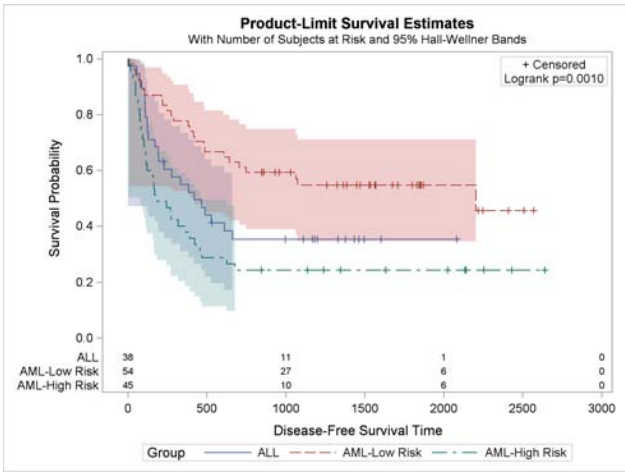
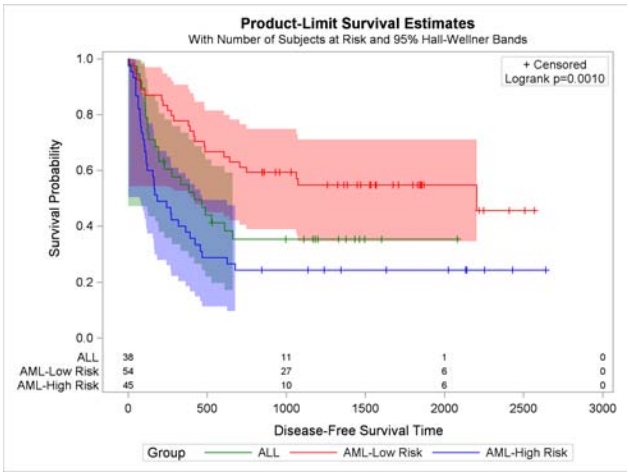


Figure 21: Named Color Specifications



CONTROLLING THE SURVIVAL PLOT BY MODIFYING STYLE TEMPLATES

Graphs that are produced by ODS Graphics are controlled by the data object (the matrix of information that is graphed), the graph template (the program that controls how a specific graph is constructed), and a style template (a program that controls the overall appearance of graphs including colors, line and marker styles, sizes, fonts, and so on). You can use different styles or change styles to change the appearance of the survival plot and all other graphs. The graphs that have been displayed up to this point were all produced using the HTMLBlue style, which is the default style for the HTML destination in the SAS windowing environment. This is an all color style; it does not rely on line style or marker changes to differentiate groups. You can switch to a style that varies colors, markers, and lines by specifying the STYLE= option in an ODS destination statement. The following step uses the HTMLBlueCML style to make a graph where line styles differ:

```
ods html style=htmlbluecml image_dpi=300;
proc lifetest data=BMT plots=survival(cb=hw test atrisk(maxlen=13));
   time T * Status(0);
   strata Group / order=internal;
   format group bmtfmt.;
run;
ods html close;
```

The results are displayed in Figure 20. This example also illustrates specifying the IMAGE_DPI= option to control the resolution (measured in DPI, or dots per inch) of the image. All images in this paper are created at 300 DPI. The default setting for the HTML destination is 100 DPI. Images created at 300 DPI are cleaner than images created at 100 DPI, but they require roughly nine times as much disk space.

You can modify a style to take more detailed control of the graph. The style elements **GraphData1**, **GraphData2**, ..., **GraphData12** control the appearance of groups of observations, such as the survival step plots in the Kaplan-Meier plot. The %ModStyle autocall macro provides a convenient way to modify these style elements. By default, the colors for the first three groups in the HMTLBlue style are shades of blue, red, and green. You can change them to green, red, and blue as follows:

```
%modstyle(name=mystyle, parent=htmlblue,
          colors=green red blue, fillcolors=green red blue)

ods html style=mystyle image_dpi=300;
proc lifetest data=BMT plots=survival(cb=hw test atrisk(maxlen=13));
   time T * Status(0);
   strata Group / order=internal;
   format group bmtfmt.;
run;
ods html close;
```

The results are displayed in Figure 21. The COLORS= option specifies the contrast colors, which are for markers and lines. The FILLCOLORS= option specifies the colors, which are for filled areas. The original colors (as shown in Figure 20) are more subtle than those shown in Figure 21. If you want to change the order of the original colors, you need to know what they are so that you can specify them.

You can use PROC TEMPLATE with the SOURCE statement to display the style as follows:

```
proc template;
   source styles.htmlblue;
run;
```

The results of this step are not shown. The results include the option PARENT=STYLES.STATISTICAL and do not include definitions of the colors (**gData1**, **gData2**, ..., **gData12**) and contrast colors (**gcData1**, **gcData2**, ..., **gcData12**). These are the color definitions that are used in the style elements **GraphData1**, **GraphData2**, ..., **GraphData12**. You can examine the parent Statistical style as follows:

```
proc template;
   source styles.statistical;
run;
```

The results of this step are not shown because they are hard to interpret in their raw form, but the desired color definitions are there. You can submit the following statements to display the colors for the Statistical (and hence HTMLBlue) style in a more understandable form:

```
proc template;
   source styles.statistical / file='junk.junk';
run;

data colors;
   infile 'junk.junk';
   input;
   if index(_infile_, 'data') then do;
      element = scan(_infile_, 1, ' ');
      Color = scan(_infile_, 3, ' ;');
      Type = scan('Fill Colors,Line Colors', 1 + (index(element, 'gc') gt 0), ',');
      i = input(compress(element, 'gcdata'';'), ?? 2.);
      if i then output;
   end;
run;

proc sort; by descending type i; run;

proc print; ods select print; id type; by descending type; var color; run;
```

The results are displayed in Figure 22.

Figure 22 HMTLBlue Style Colors List

Type	Color
Line Colors	cx445694
	cxA23A2E
	cx01665E
	cx543005
	cx9D3CDB
	cx7F8E1F
	cx2597FA
	cxB26084
	cxD17800
	cx47A82A
	cxB38EF3
	cxF9DA04
Fill Colors	cx6F7EB3
	cxD05B5B
	cx66A5A0
	cxA9865B
	cxB689CD
	cxBABC5C
	cx94BDE1
	cxCD7BA1
	cxCF974B
	cx87C873
	cxB7AEF1
	cxDDD17E

Figure 23: HTMLBlue Style Colors Display

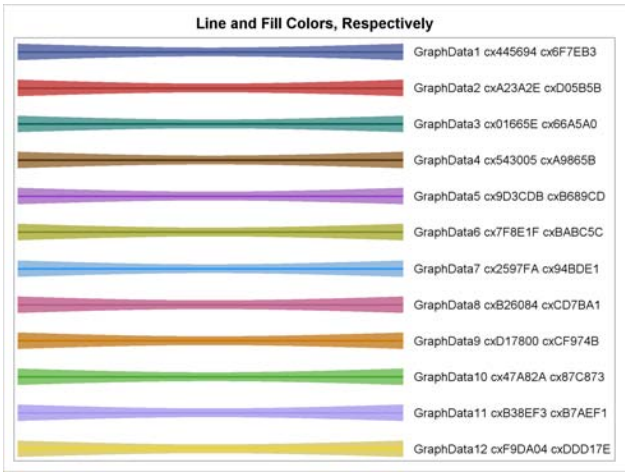
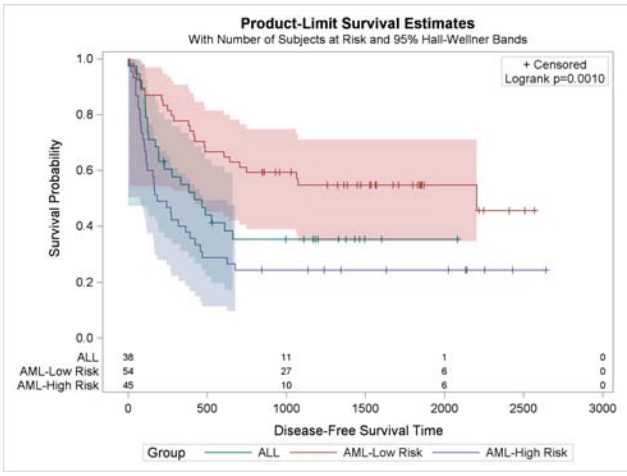


Figure 24: Swapping Colors from the Style



You can use the following steps to display the **GraphData1** – **GraphData12** line and fill colors (contrast colors and colors, respectively):

```
data display;
   array y[12] y1 - y12;
   do i = 1 to 12; y[i] = i;       end;
   do x = 1 to 10; output;         end;
   do i = 1 to 12; y[i] = i + .5; end;
   do x = 1 to 10; output;         end;
run;

data _null_;
   set colors;
   call symputx(compress(type || put(i, 2.)), color);
run;

proc sgplot noautolegend data=display;
   %macro reg;
      title 'Line and Fill Colors, Respectively';
      %do i = 1 %to 12;
         reg y=y%eval(13-&i) x=x / lineattrs=GraphData&i clmattrs=GraphData&i
                                   nomarkers clm  curvelabelpos=max
                                   curvelabel="  GraphData&i &&LineColors&i &&FillColors&i";
      %end;
   %mend;
   %reg
   xaxis display=none;
   yaxis display=none;
   ods select sgplot;
run;
```

The results are displayed in Figure 23. The colors in Figure 23 are richer than the colors in the bands in the survival plots because of the DATATRANSPARENCY= options in the BANDPLOT statements.

You can copy the third, second, and first color from each list and use the %ModStyle macro to switch the colors as follows:

```
%modstyle(name=mystyle, parent=htmlblue,
          colors=cx01665E cxA23A2E cx445694, fillcolors=cx66A5A0 cxD05B5B cx6F7EB3)

ods html style=mystyle image_dpi=300;
proc lifetest data=BMT plots=survival(cb=hw test atrisk(maxlen=13));
   time T * Status(0);
   strata Group / order=internal;
   format group bmtfmt.;
run;
ods html close;
```

The results are displayed in Figure 24. The familiar colors are used, but they are now in a different order.

SAS ITEM STORES

In other sections of this paper, you submit PROC TEMPLATE code (either directly or through macros) to compile and save graph and style templates. Assuming that you have not modified your ODS path with an ODS PATH statement, compiled templates are stored in an item store in the Sasuser library. By default, all templates that SAS provides are stored in an item store in the Sashelp library. By default, the Sashelp item store has read access only; you cannot write to it. By default, the Sasuser item store has update access; you can both read and write to it. **Never set the Sashelp item store to update or write access.** If you do, and if you have administrator privileges on your computer, you could corrupt the Sashelp item store.

Assuming that the default ODS path is used, ODS first looks for a template in the Sasuser item store. If it does not find the template there, ODS next looks for the template in the Sashelp item store. Files in the Sasuser library persist across SAS sessions until they are deleted. You can run the following step to delete the entire Sasuser item store so that ODS uses only the templates supplied by the SAS System:

```
ods path sashelp.tmplmst(read);
proc datasets library=sasuser nolist;
   delete templat(memtype=itemstor);
run;
ods path reset;
```

For more information about the ODS path and SAS item stores, see the chapter “Statistical Graphics Using ODS” in *SAS/STAT User’s Guide*.

CONCLUSIONS

You have many methods available to you for modifying the PROC LIFETEST survival plot. You can use the PLOTS= option. Some of the options discussed in this paper are new in SAS/STAT 12.1. If you find that PLOTS= option modifications are not sufficient, you can modify the graph templates through the macros. You can control colors and overall appearance by using a different SAS output style or by modifying a style. You can find more information about modifying the PROC LIFETEST survival plot in the *SAS/STAT User’s Guide*, in the procedure chapter on PROC LIFETEST and the introductory chapter on ODS Graphics template modification. You can find both on this support.sas.com documentation page: `http://support.sas.com/documentation/onlinedoc/stat/index.html`. However, the documentation discusses modifying the templates only in the case where the at-risk table is inside the graph.

SOFTWARE CREDITS

The LIFETEST procedure was designed and programmed by Ying So, Principal Research Statistician at SAS.

ACKNOWLEDGMENTS

The authors are grateful to Paul Savarese, Ed Huddleston, Judi Rourke, Tim Arnold, and Chelsea Liu of SAS Institute Inc. for their valuable assistance in the preparation of this paper.

CONTACT INFORMATION

Warren F. Kuhfeld
SAS Institute Inc.
S6018 SAS Campus Drive
Cary, NC, 27513
(919) 531-7922
Warren.Kuhfeld@sas.com

Ying So
SAS Institute Inc.
S6048 SAS Campus Drive
Cary, NC, 27513
(919) 531-0359
Ying.So@sas.com

REFERENCES

Hall, W. J. and Wellner, J. A. (1980), “Confidence Bands for a Survival Curve from Censored Data,” *Biometrika*, 67, 133–143.

Kaplan, E. L. and Meier, P. (1958), “Nonparametric Estimation from Incomplete Observations,” *Journal of the American Statistical Association*, 53, 457–481.

Klein, J. P. and Moeschberger, M. L. (1997), *Survival Analysis: Techniques for Censored and Truncated Data*, New York: Springer-Verlag.