THE POWER TO KNOW.

SAS/STAT® 15.1 User's Guide

Customizing the Kaplan-Meier Survival Plot

This document is an individual chapter from *SAS/STAT® 15.1 User's Guide*.

The correct bibliographic citation for this manual is as follows: SAS Institute Inc. 2018. *SAS/STAT® 15.1 User's Guide*. Cary, NC: SAS Institute Inc.

SAS/STAT® 15.1 User's Guide

SAS Institute Inc., SAS Campus Drive, Cary, NC 27513-2414

November 2018

Chapter 23
Customizing the Kaplan-Meier Survival Plot

Contents

Overview

The LIFETEST procedure is a nonparametric procedure for analyzing survival data. You can use PROC LIFETEST to compute the Kaplan-Meier curve (1958), which is a nonparametric maximum likelihood estimate of the survivor function. The Kaplan-Meier plot (also called the product-limit survival plot) is a popular tool in medical, pharmaceutical, and life sciences research. The Kaplan-Meier plot contains step functions that represent the Kaplan-Meier curves of different samples (strata). The Kaplan-Meier plot has many other features that you can add or change through procedure options, graph templates, and style templates. This chapter explores these features in detail but does not explain how to interpret the graphs or the underlying analysis. For more information about PROC LIFETEST and the Kaplan-Meier plot, see Chapter 74, "The LIFETEST Procedure."

This chapter shows you how to modify the Kaplan-Meier plot through a series of examples. It discusses four types of examples: specifying procedure options, modifying graph templates by using macro variables, modifying graph templates by using macros, and changing styles. Most examples do not go into detail about the tools that underlie the template changes. Each example is designed to be small, simple, self-contained, and easy to copy and use "as is" or with minor modifications. Subsequent sections provide more details about the macro variables and macros that are used to modify the graph templates. You can use the simple examples to make a wide variety of changes without reading or understanding the detailed descriptions at the end of this chapter.

Statistical procedures produce tables by using the Output Delivery System (ODS) and produce graphs by using ODS Graphics. Procedures produce graphs as automatically as they produce tables, and graphs and tables are integrated in the ODS output. Graphs that are produced by ODS Graphics are controlled by options, the data object (the matrix of information that is graphed), a style template, and a graph template. A style template is a SAS program that controls the overall appearance of graphs, including colors, line and marker styles, sizes, fonts, and so on. 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. Each graph that is created when ODS Graphics is enabled is controlled by a graph template.[1]

If you want to modify a graph template, you usually use the TEMPLATE procedure to display the template of interest, and then you 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. The PROC LIFETEST survival plot is the only plot in SAS for which you have another alternative available for template modification. SAS provides the survival plot templates in a series of macros and macro variables that are modular and easier to modify than the original templates. This chapter provides numerous examples of using these macros and macro variables.

The data that are used in this chapter 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.

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. Other examples use formats and the ORDER= option to control the order of the groups.

Enabling ODS Graphics and the Default Kaplan-Meier Plot

You can use the following statements to enable ODS Graphics and run PROC LIFETEST:

```
ods graphics on;

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

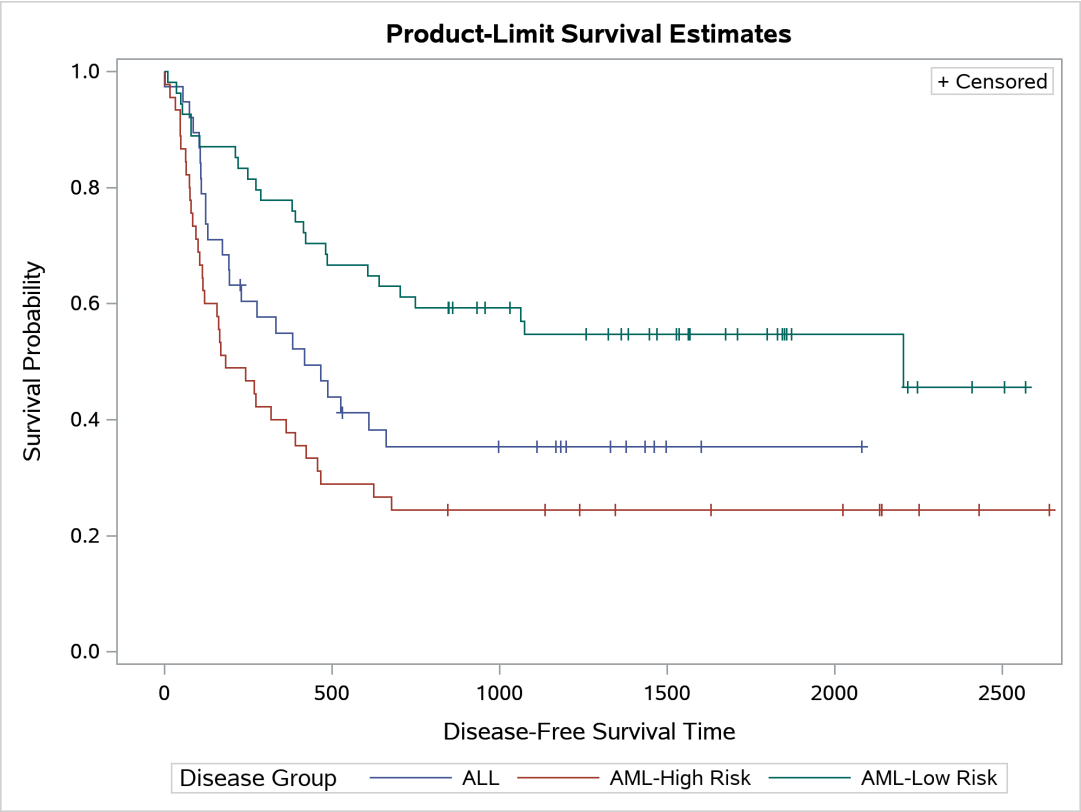
[1] ODS Graphics might or might not be enabled by default. ODS Graphics is usually enabled by default in the SAS windowing environment and disabled when you invoke SAS in other ways. However, these defaults can be changed in a number of ways. ODS Graphics is enabled in the first example in this chapter by the ODS GRAPHICS ON statement and remains enabled throughout the chapter.

ODS Graphics is enabled for this step and all subsequent steps until it is disabled. ODS Graphics remains enabled throughout the examples in this chapter.

You specify in the TIME statement that the disease-free survival time is recorded in the variable T. You further specify that the variable Status indicates censoring and 0 indicates a censored time. Separate survivor functions are displayed for each group in the Group variable, which you specify in the STRATA statement.

The plot in Figure 23.1 consists of three step functions, one for each of the three groups of patients. The plot 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 23.1 Default Kaplan-Meier Plot



The following step, which explicitly specifies the default PLOTS=SURVIVAL option, is equivalent to the preceding step:

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

The PLOTS= option enables you to control the graphs that a procedure produces. You can use it to request nondefault graphs and specify options for some graphs. You can specify graph names (PLOTS=SURVIVAL), graph options (PLOTS=SURVIVAL(ATRISK OUTSIDE)), and suboptions (PLOTS=SURVIVAL(ATRISK OUTSIDE(0.15))). The PLOTS= option is described in the section "PROC LIFETEST Statement" on page 5576 in Chapter 74, "The LIFETEST Procedure."

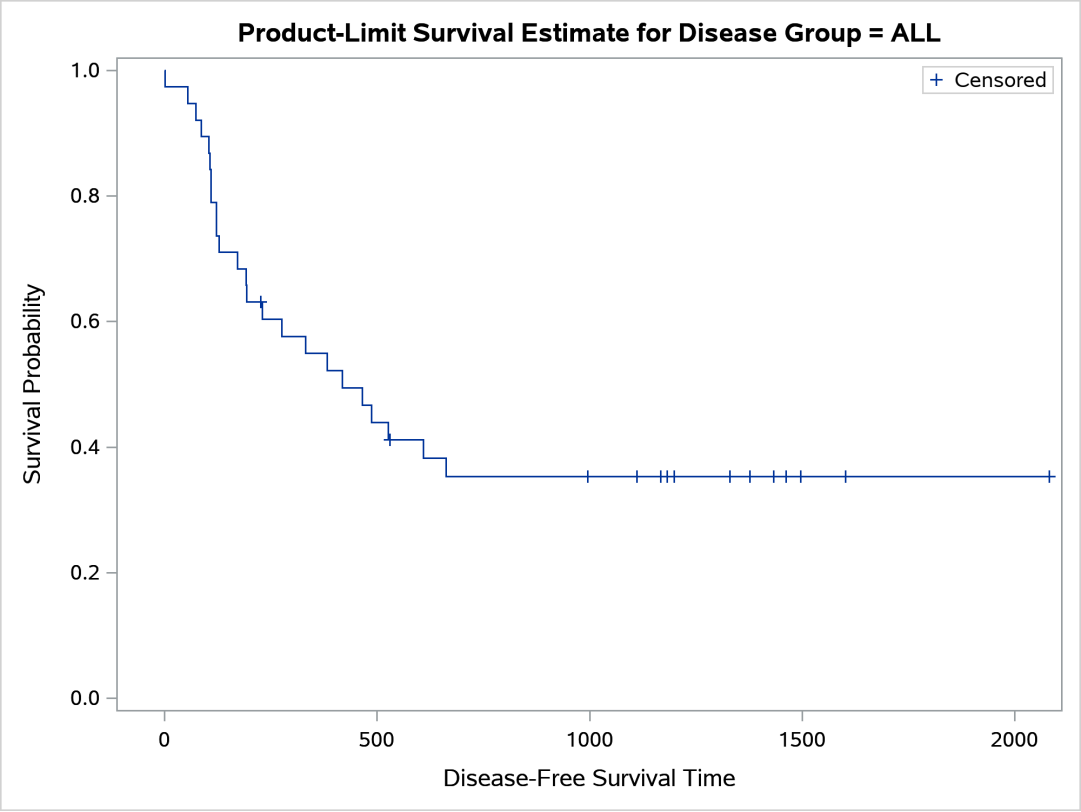
Individual Survival Plots

You can use the STRATA=INDIVIDUAL option to request individual survival plots. By default, the STRATA=OVERLAY option produces the plot of overlaid step functions displayed in Figure 23.1. 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 23.2. To conserve space, the other graphs are not displayed.

Figure 23.2 One of Three Individual Plots

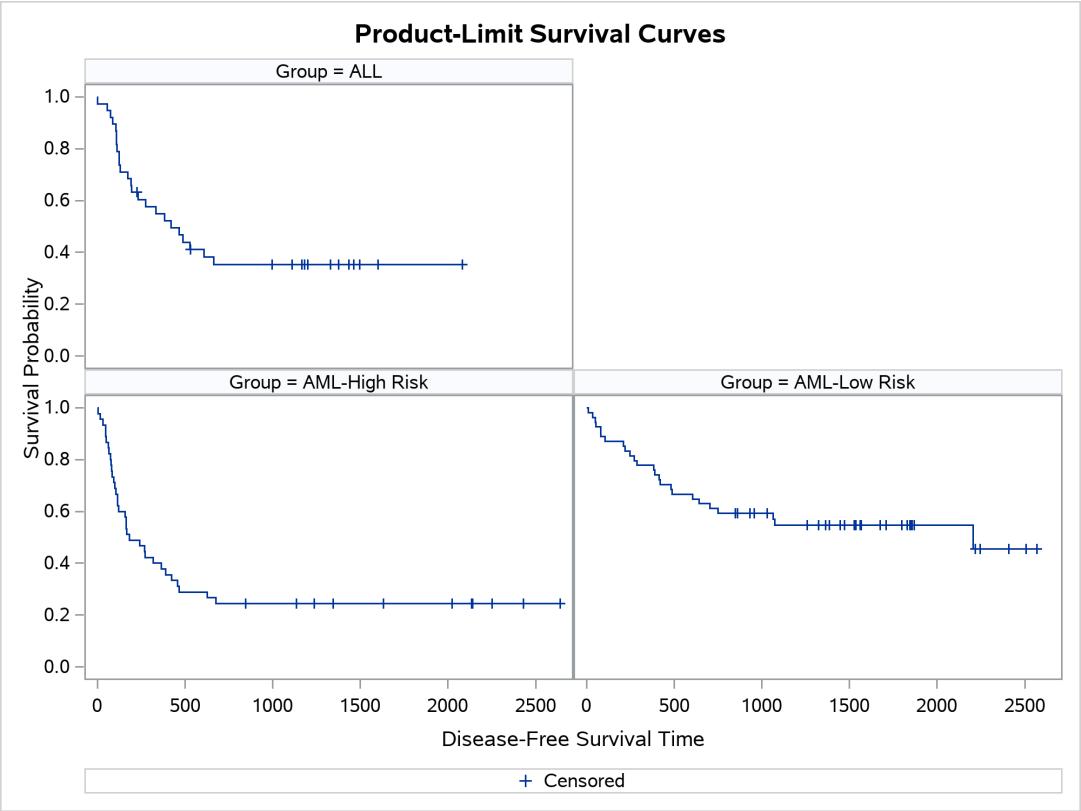


You can use the STRATA=PANEL option as follows to display the results in separate panels of a single graphical display:

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

The results are displayed in Figure 23.3.

Figure 23.3 Individual Plots Displayed in a Panel



The rest of this chapter discusses overlaid plots such as the one displayed in Figure 23.1.

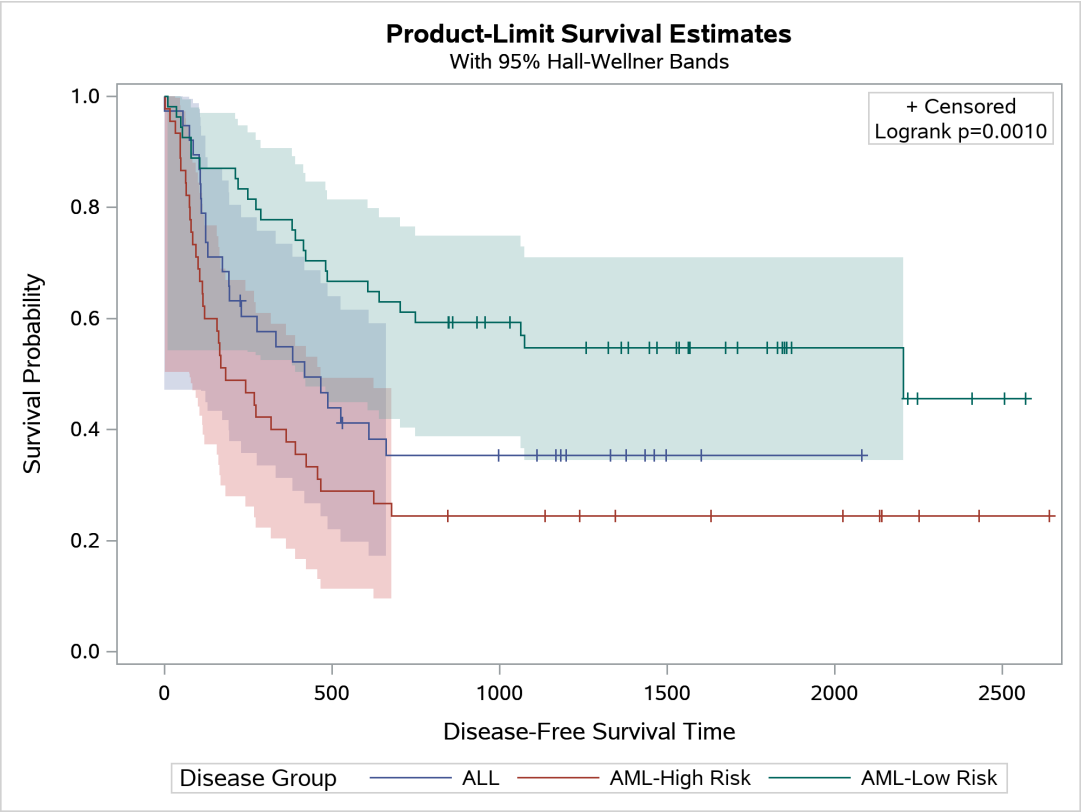
Hall-Wellner Confidence Bands and Homogeneity Test

You can use the following statements to add Hall-Wellner confidence bands (Hall and Wellner 1980) to Figure 23.1 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 23.4. The Hall-Wellner confidence bands extend to the last event times. The small *p*-value supports rejecting the hypothesis that the groups are homogeneous.

Figure 23.4 Confidence Bands and Homogeneity Test



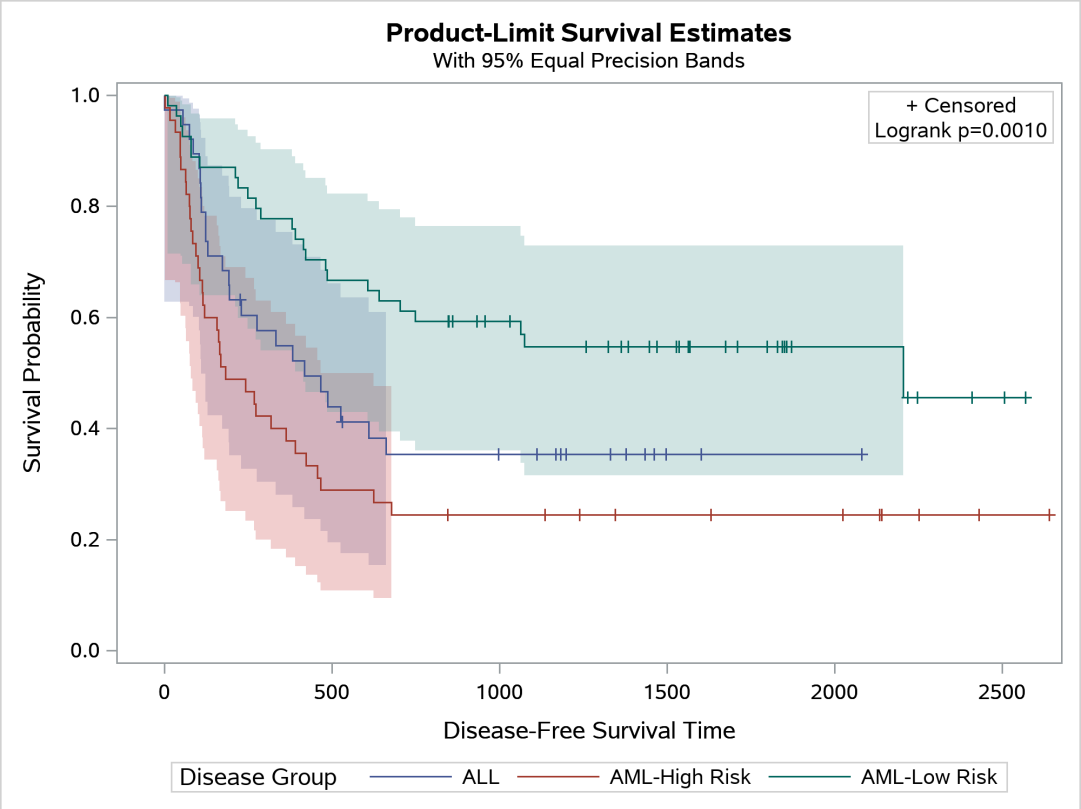
Equal-Precision Bands

You can use the following statements to add equal-precision bands to the plot:

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

The results are displayed in Figure 23.5.

Figure 23.5 Equal-Precision Bands

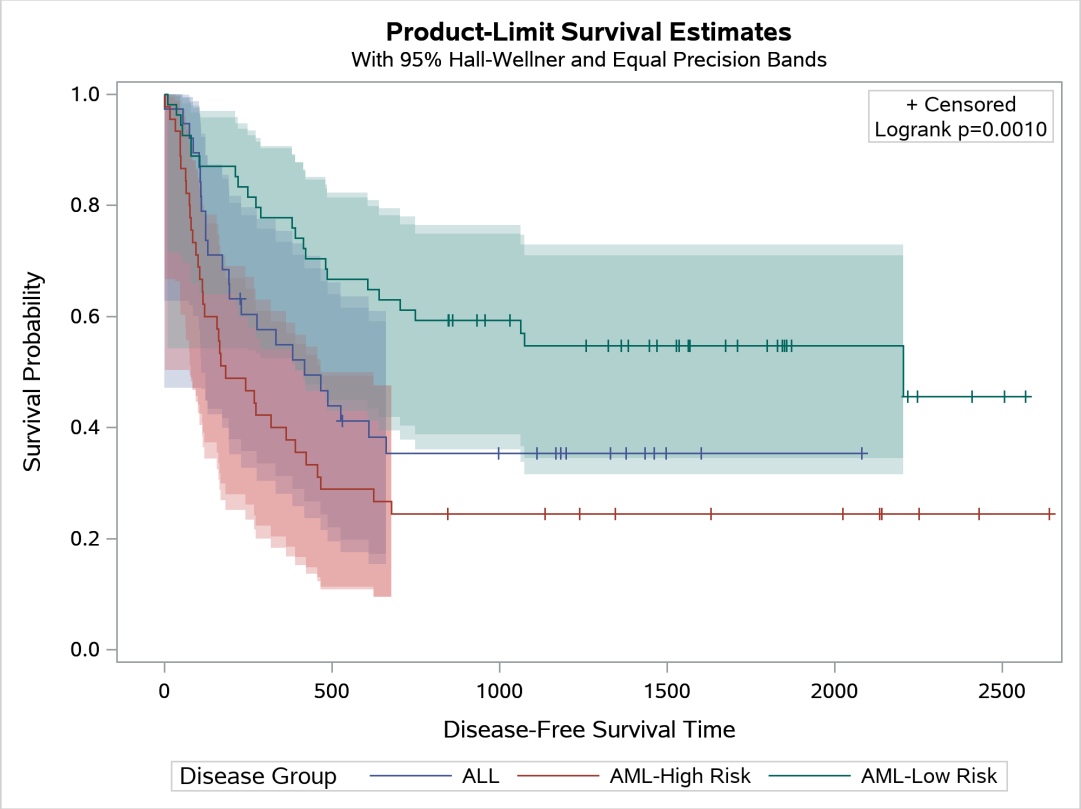


You can use the following statements to add both Hall-Wellner and equal-precision bands to the plot:

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

The results are displayed in Figure 23.6.

Figure 23.6 Hall-Wellner and Equal-Precision Bands



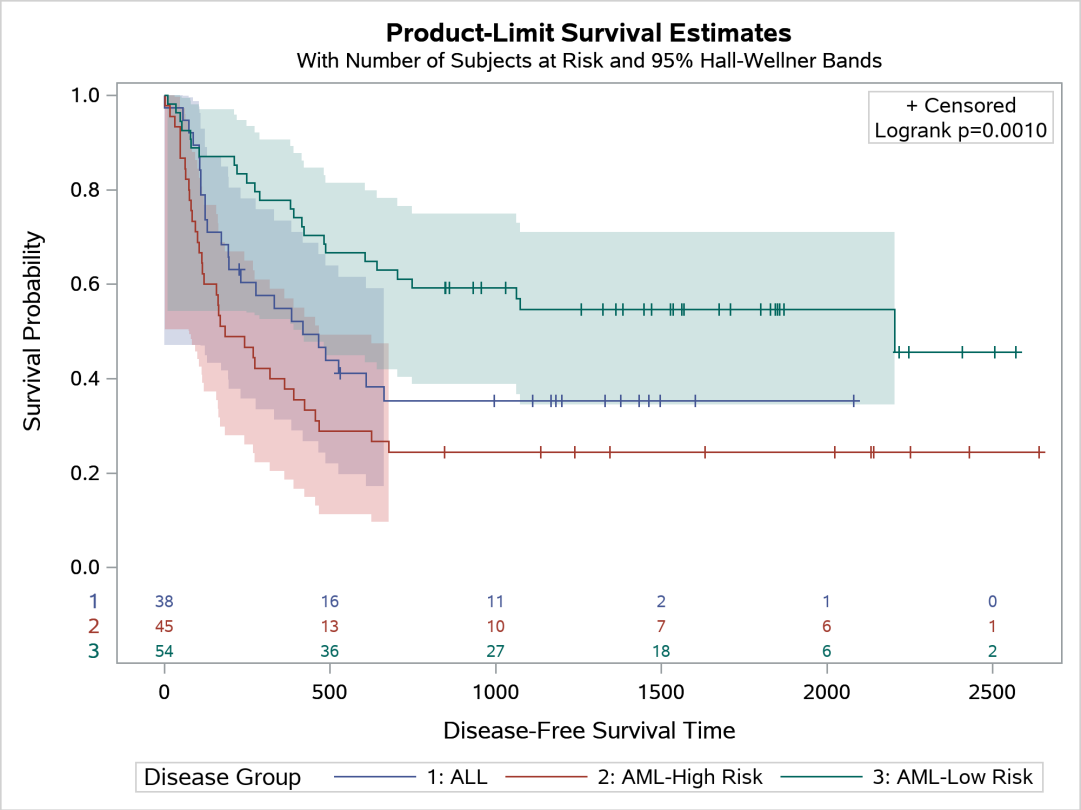
Displaying the Patients-at-Risk Table inside the Plot

You can add the patients-at-risk table to the Kaplan-Meier plot 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 23.7. By default, the at-risk table is displayed inside the body of the plot. This table shows the number of patients who are at risk for each group for each of the different times. For these data, the default survival times at which at-risk values are displayed are 0 to 2500 by 500. You will see how to specify other values in subsequent examples.

Figure 23.7 At-Risk Table inside the Plot

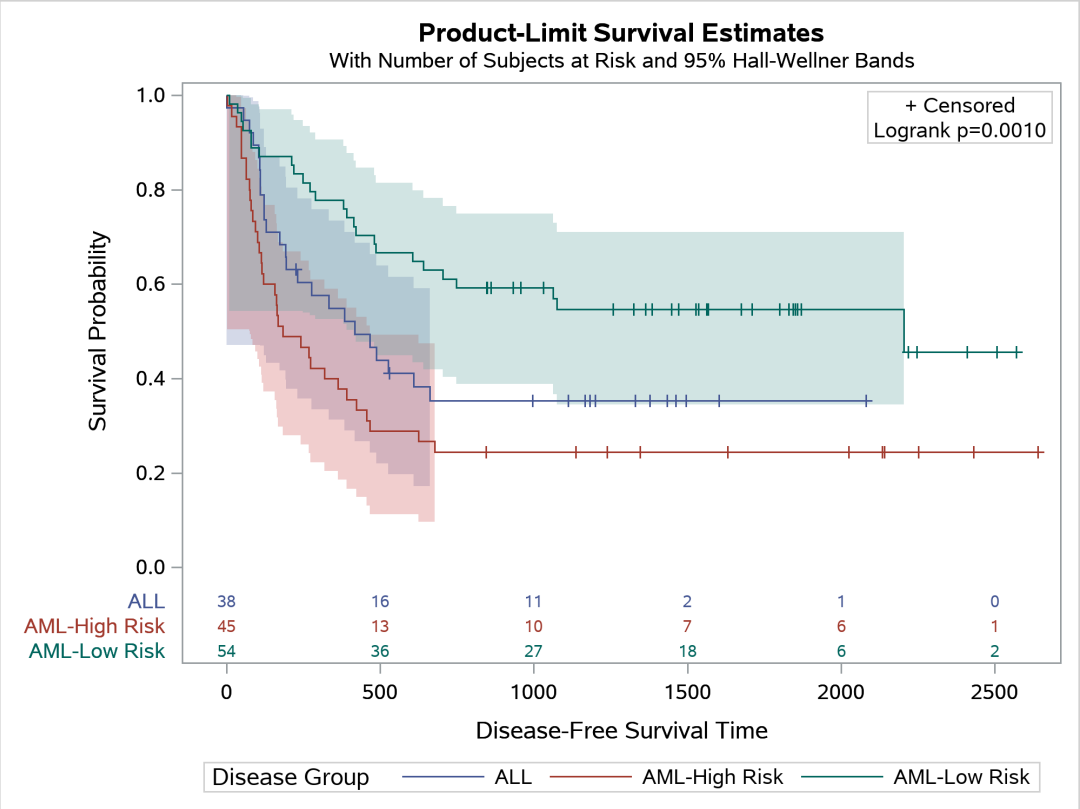


The group labels for the at-risk table are group numbers, and these numbers appear in the legend. Numbers are used rather than the actual labels because the length of the longest label (13) is greater than the default that is set by the maximum label length option (MAXLEN=12). You can display labels 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 23.8. 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.

Figure 23.8 At-Risk Table with Labels

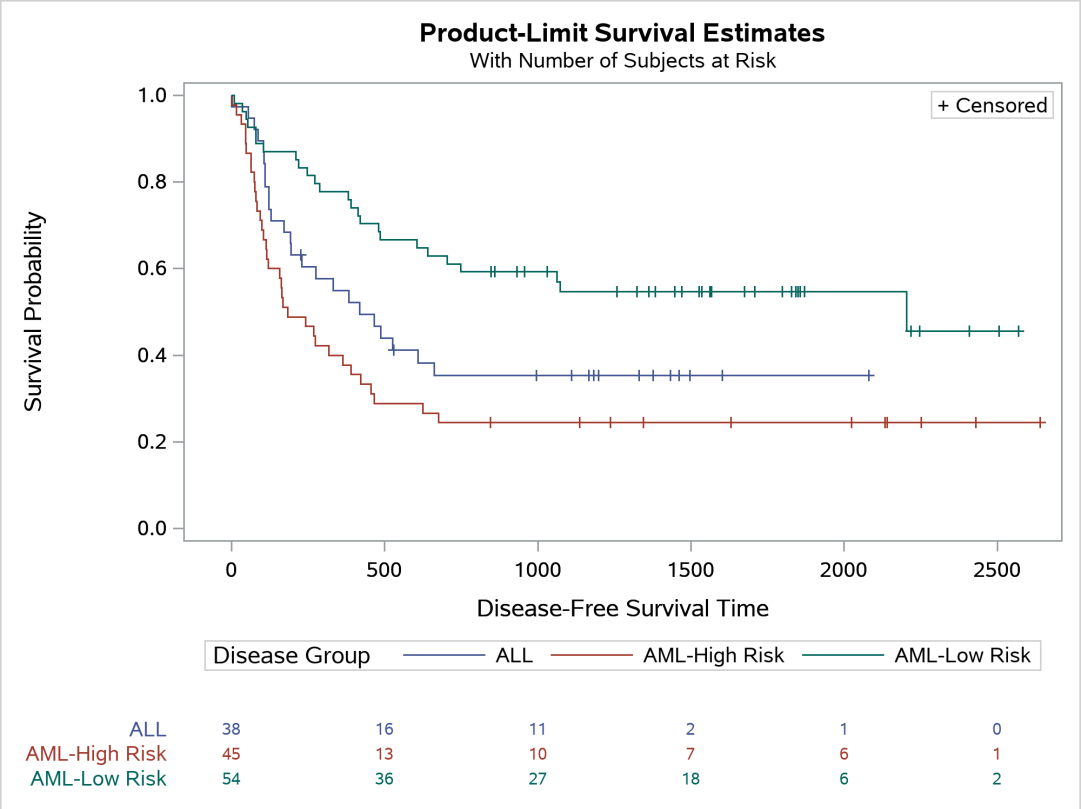


Displaying the Patients-at-Risk Table outside the Plot

You can use the PLOTS=SURVIVAL(OUTSIDE) option to display the at-risk table outside the body of the plot. 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 nested within options and options nested within those nested options. The following step produces the plot in Figure 23.9:

```
proc lifetest data=sashelp.BMT
              plots=survival(atrisk(maxlen=13 outside(0.15)));
   time T * Status(0);
   strata Group;
run;
```

Figure 23.9 Moving the At-Risk Table outside the Plot



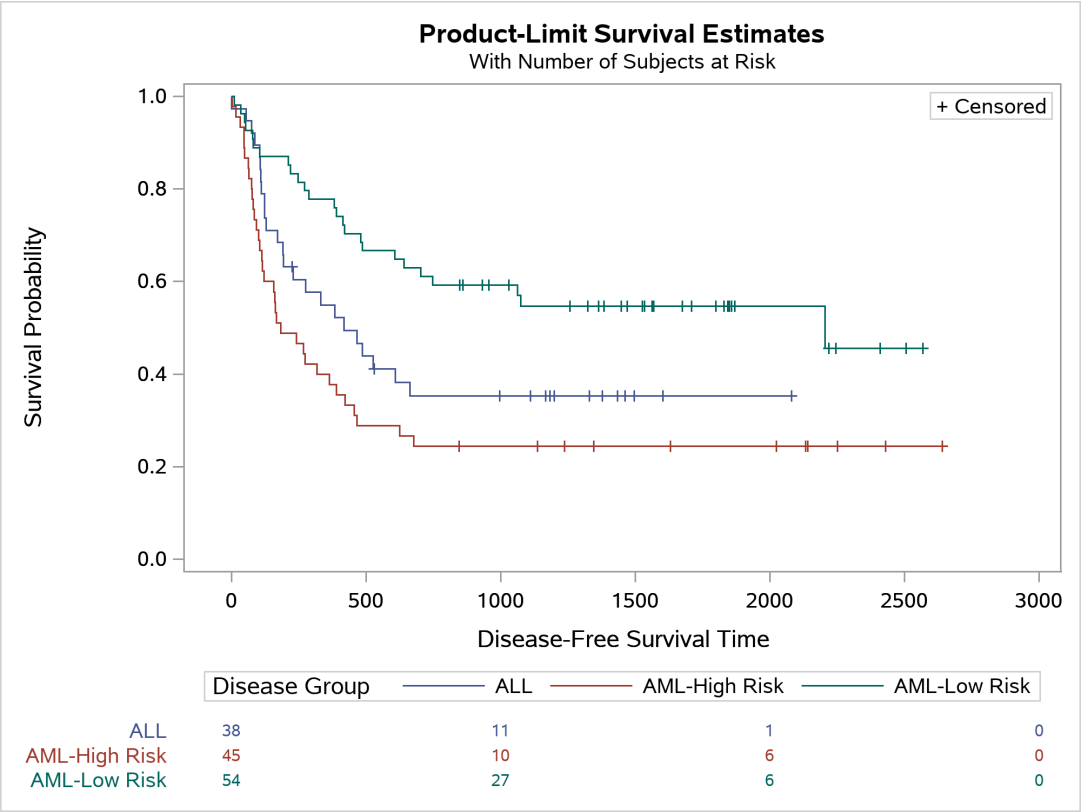
Modifying At-Risk Table Times

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 1000) option:

```
proc lifetest data=sashelp.BMT
              plots=survival(atrisk(maxlen=13 outside)=0 to 3000 by 1000);
   time T * Status(0);
   strata Group;
run;
```

The results are displayed in Figure 23.10.

Figure 23.10 Specifying At-Risk Values

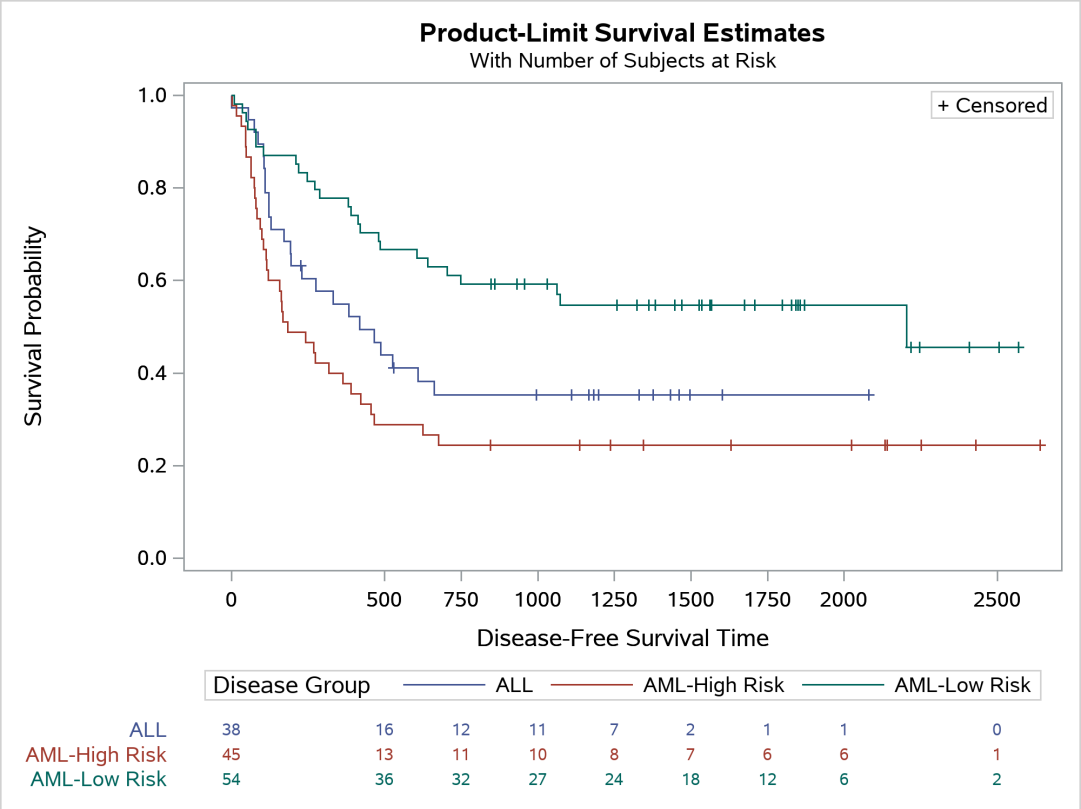


You can specify at-risk values that do not correspond to the original time axis tick marks. You can use the PLOTS=SURVIVAL(ATRISK(ATRISKTICK)) option to add tick marks that correspond to the specified at-risk values:

```
proc lifetest data=sashelp.BMT plots=survival(atrisk
   (atrisktick maxlen=13 outside)=0 500 750 1000 1250 1500 1750 2000 2500);
   time T * Status(0);
   strata Group;
run;
```

The results are displayed in Figure 23.11.

Figure 23.11 Controlling At-Risk Tick Marks

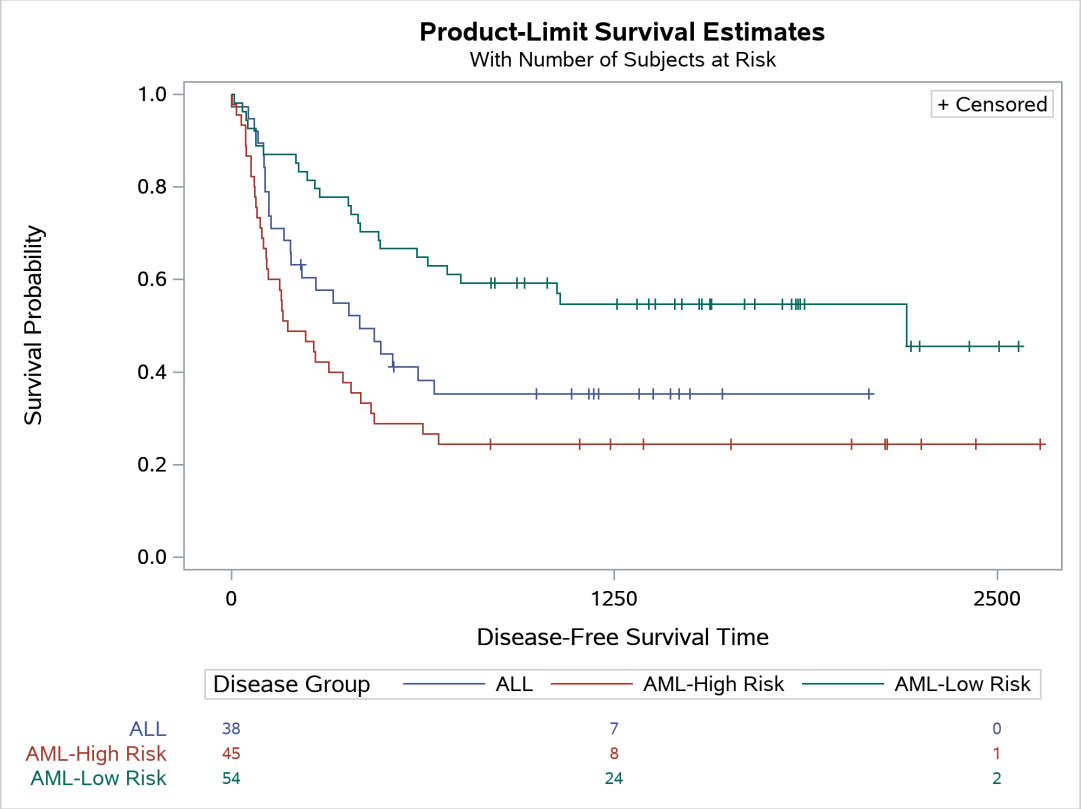


You can display tick values only at those times that are given in the ATRISK= list:

```
proc lifetest data=sashelp.BMT plots=survival(atrisk
   (atrisktickonly maxlen=13 outside)=0 1250 2500);
   time T * Status(0);
   strata Group;
run;
```

The results are displayed in Figure 23.12.

Figure 23.12 Controlling At-Risk Tick Marks



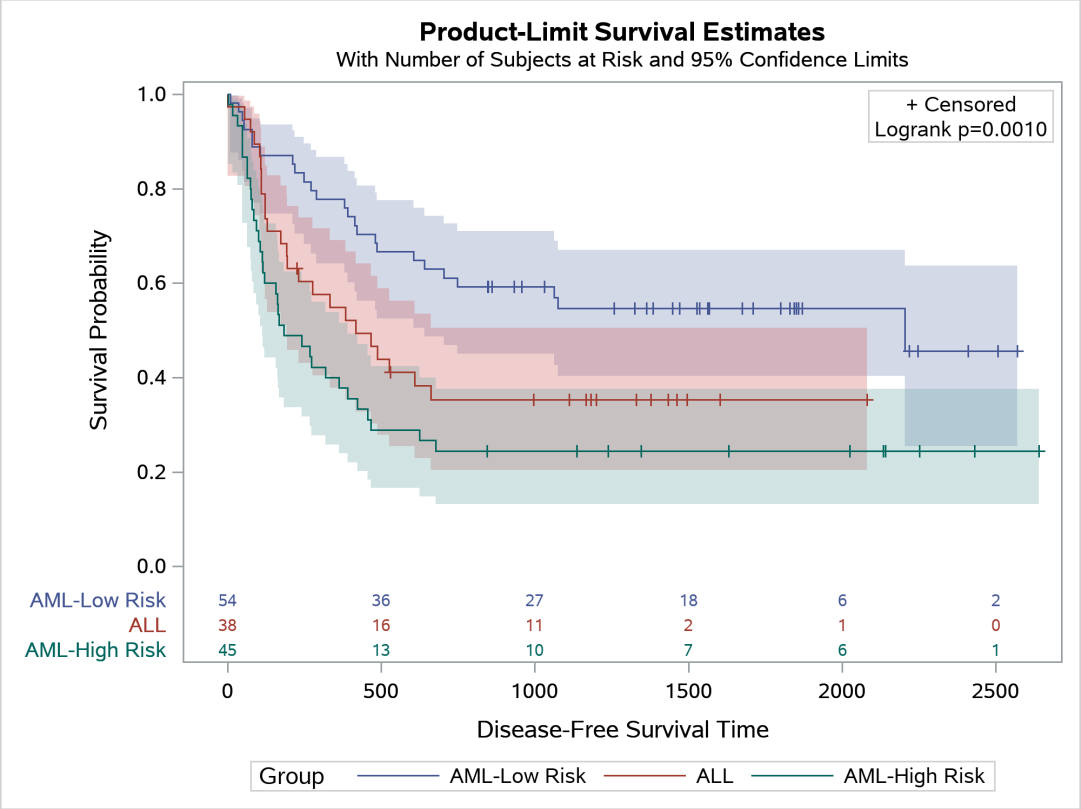
Reordering the Groups

You can change the order of the legend entries by first changing each original group value to a new value in the desired order and then running the analysis with a FORMAT statement to provide the original values. In this example, the order is changed to AML–Low Risk (the top function), followed by ALL (the middle function), followed by AML–High Risk. With this ordering, there is a clearer correspondence between the functions, the at-risk table, and the legend. The following steps illustrate this reordering:

```
proc format;
   invalue bmtnum   'AML-Low Risk' = 1  'ALL' = 2  'AML-High Risk' = 3;
   value   bmtfmt   1 = 'AML-Low Risk'  2 = 'ALL'  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 correct order. The VALUE statement creates a format that maps the integers back to the original values. The informat is used with the INPUT function in the DATA step to create a new integer Group variable. The FORMAT statement assigns the BMTFMT format to the Group variable so that the actual risk groups are displayed in the analysis. You specify the ORDER=INTERNAL option in the STRATA statement to sort the Group values based on internal order (the order specified by the integers, which are the internal unformatted values). This example also illustrates 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 23.13.

Figure 23.13 Controlling Legend Order

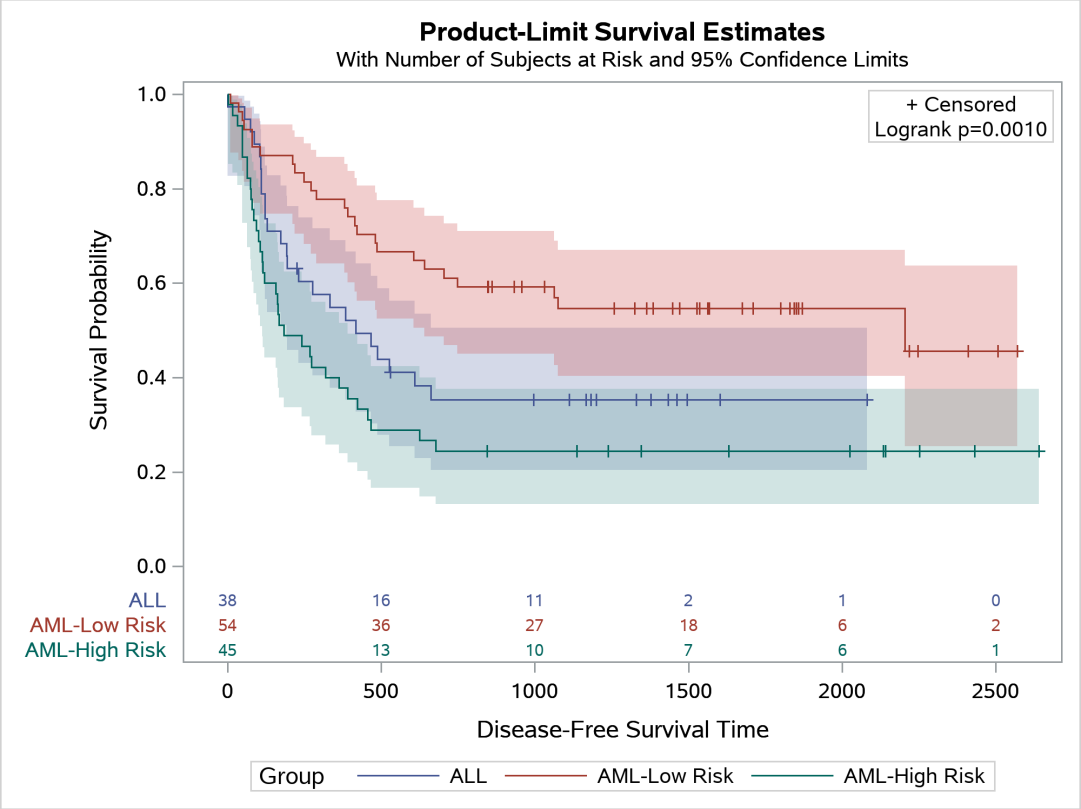


You can submit the following steps to display ALL first, followed by AML–Low Risk and then AML–High Risk:

```
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 results are displayed in Figure 23.14.

Figure 23.14 Controlling Legend Order



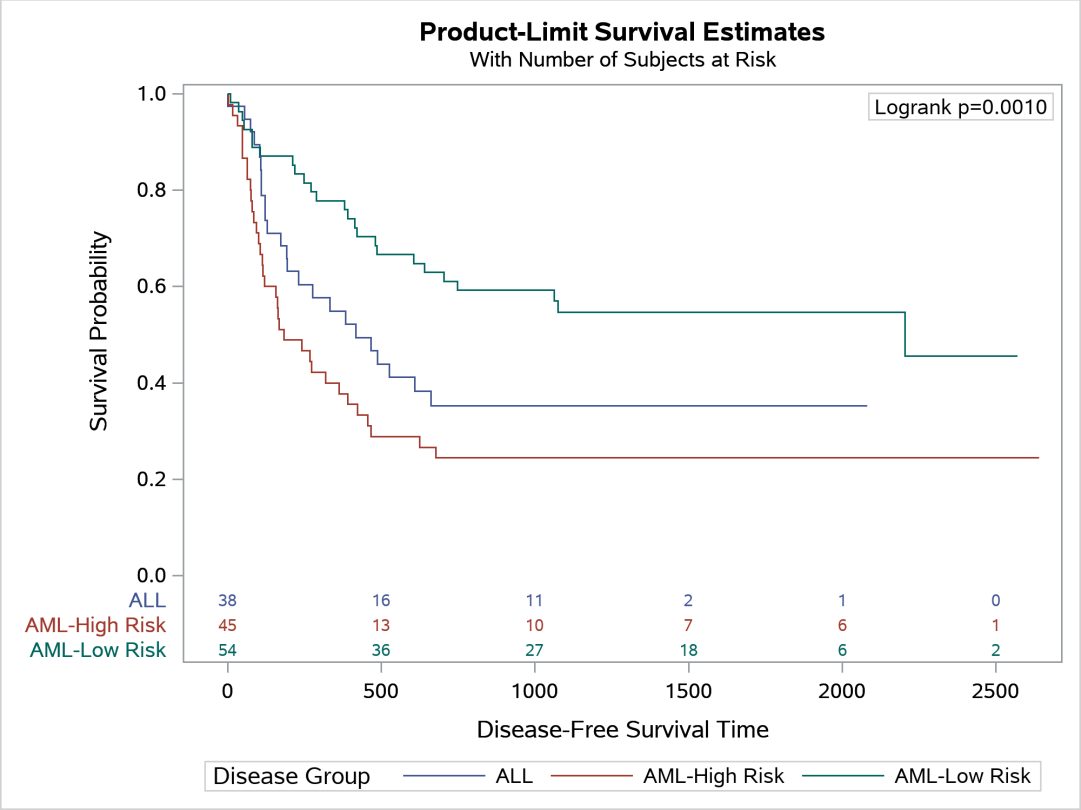
Suppressing the Censored Observations

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

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

The results are displayed in Figure 23.15.

Figure 23.15 Censored Values Not Displayed



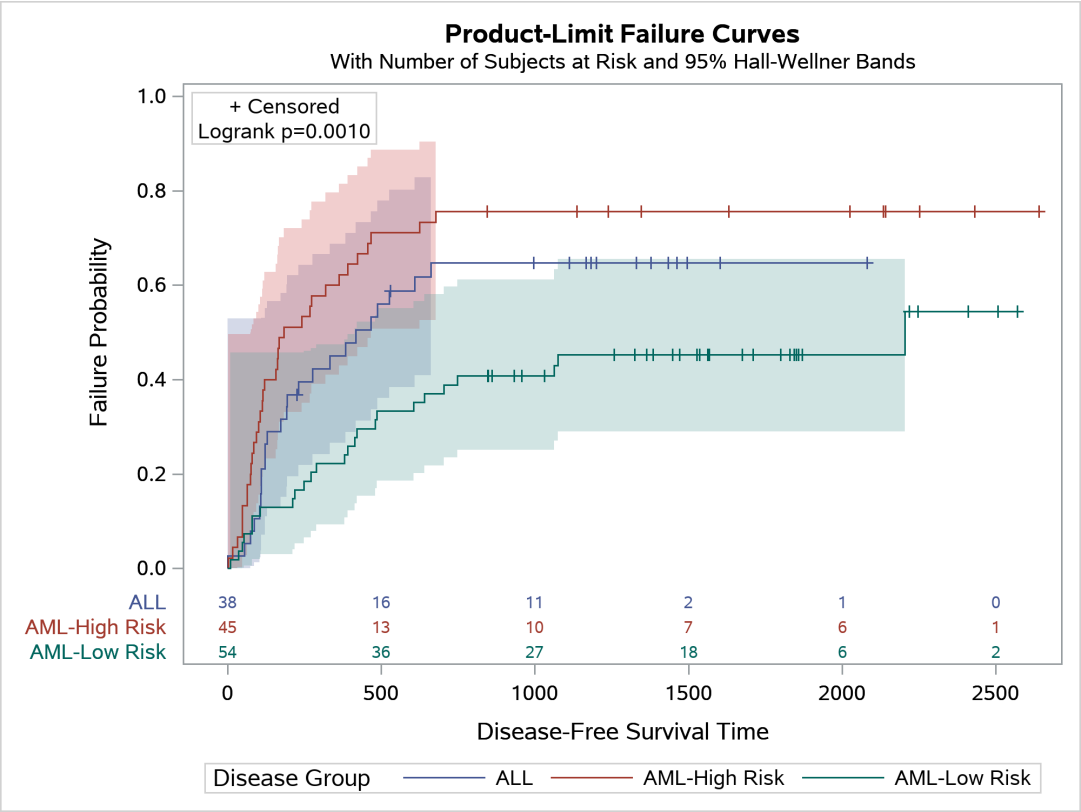
Failure Plots

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

The results are displayed in Figure 23.16.

Figure 23.16 Failure Plot



The section "Controlling the Survival Plot by Modifying Graph Templates" on page 894 provides macros for customizing the survival plot. That is the only plot in SAS that has customization macros. If you want to customize the failure plot or other plots, you might need to directly modify the graph template. The rest of this example shows how to programmatically modify the template for the failure plot by using a DATA step. This technique of writing a DATA step to modify a template was introduced by Kuhfeld (2015).

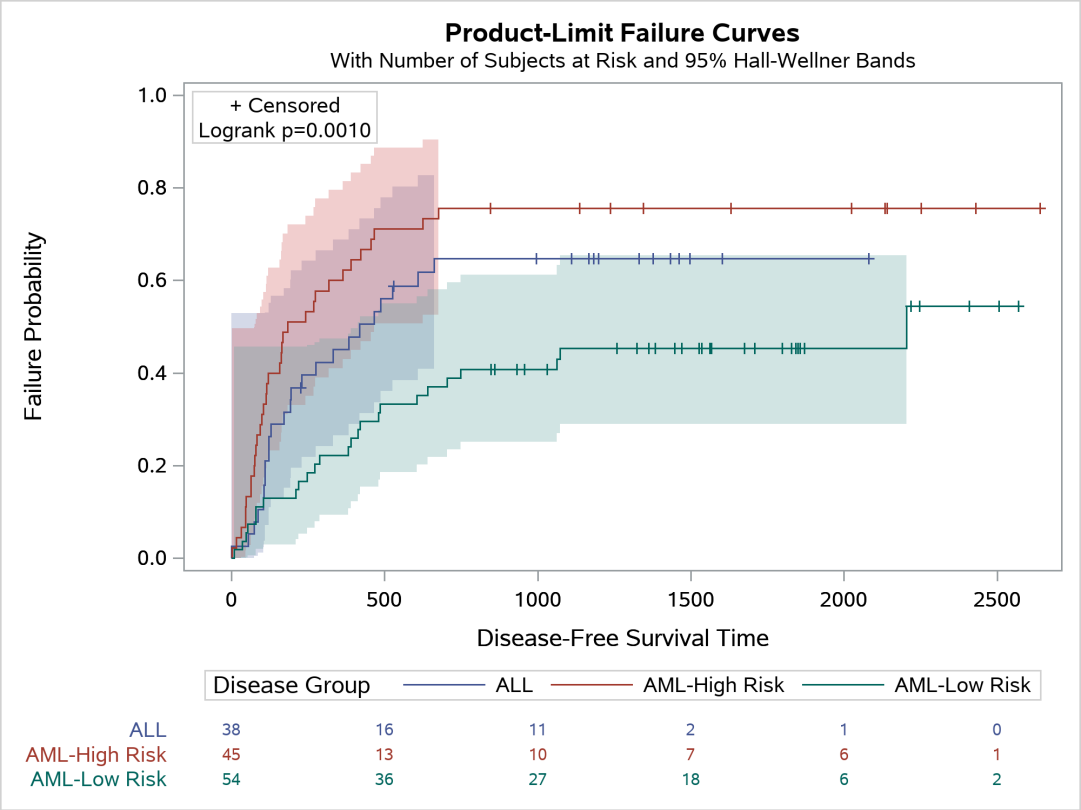
Customizing the Failure Plot Y Axis

This example shows how to display a Y axis that ranges from 0 to 0.8 instead of the customary 0 to 1. You begin by enabling ODS TRACE output and running the procedure. This step, unlike the preceding step, displays the at-risk table outside the graph.

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

The graph is displayed in Figure 23.17. It has the full Y-axis range of 0 to 1.

Figure 23.17 Failure Plot with At-Risk Table Outside



The output from the ODS TRACE ON statement (not displayed here) shows that the name of the graph template is **Stat.Lifetest.Graphics.ProductLimitFailure2**. You can write to a file the template that SAS provides for the failure plot by submitting the following step:

```
proc template;
   delete Stat.Lifetest.Graphics.ProductLimitFailure2;
   source Stat.Lifetest.Graphics.ProductLimitFailure2 / file='tpl.tpl';
quit;
```

The DELETE statement deletes any compiled templates that might be left from previous template modifications. The SOURCE statement writes the template to the file *tpl.tpl*. You might need to name your files differently, depending on your operating system and how you run SAS. You can use an editor to read the template, and then you can use a DATA step to modify it and change the Y-axis maximum. The option that you need to change is the VIEWMAX= option. You also need to submit a PROC TEMPLATE statement before submitting the template definition. Finally, you can submit a QUIT statement, although this is not required. You can do all this by submitting the following step:

```
data _null_;                                          /* Standard boilerplate   */
   infile 'tpl.tpl' end=eof;                          /* Standard boilerplate   */
   input;                                             /* Standard boilerplate   */
   if _n_ eq 1 then call execute('proc template;'); /* Standard boilerplate   */
   _infile_ = tranwrd(_infile_, 'viewmax=1',         /* Customization: var =   */
                              'viewmax=0.8');        /* tranwrd(var, from, to);*/
   call execute(_infile_);                            /* Standard boilerplate   */
   if eof then call execute('quit;');                 /* Standard boilerplate   */
run;                                                  /* Standard boilerplate   */
```

You are not required to specify variables in the INPUT statement. When you are processing text files, it is often convenient to instead use the automatic input buffer variable, _Infile_. The TRANWRD function does the translation and replaces the VIEWMAX= option. You must look at the original template to know how to write translation assignment statements. The CALL EXECUTE statements submit all the template code to a code buffer. SAS runs the program that is stored in the code buffer when the DATA step finishes. If you want to make different template modifications, you need to replace only the assignment statement. The rest of the DATA step remains the same. You can view the modified template by submitting the following statements:

```
proc template;
   source Stat.Lifetest.Graphics.ProductLimitFailure2 / store=sasuser.templat;
quit;
```

The SOURCE statement explicitly references the compiled template that is stored in the templat item store in the Sasuser library. Alternatively, you can write the modified template to a file, as in the following code, and then use a text comparison utility to see the changes between *tpl.tpl* and *tplmod.tpl*:

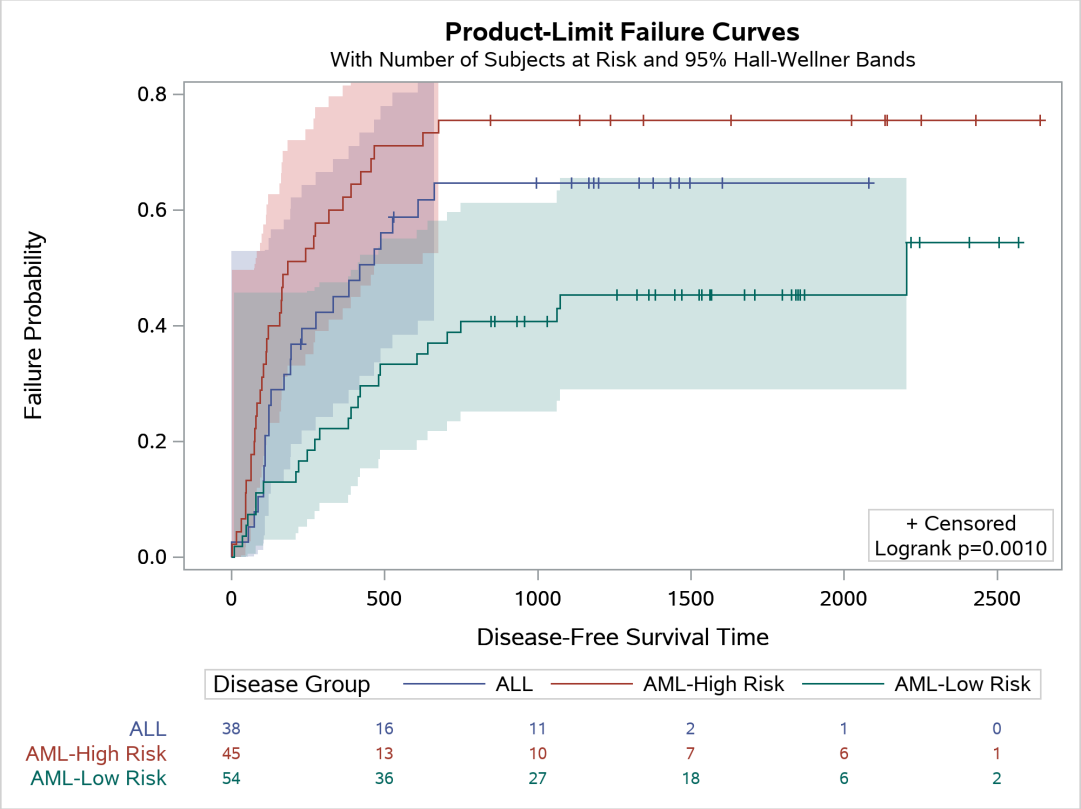
```
proc template;
   source Stat.Lifetest.Graphics.ProductLimitFailure2 /
     store=sasuser.templat file='tplmod.tpl';
quit;
```

The following step runs PROC LIFETEST, which uses the modified template:

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

The graph is displayed in Figure 23.18.

Figure 23.18 Restricted Axis Range



Converting the Y-Axis Scale to Percentages

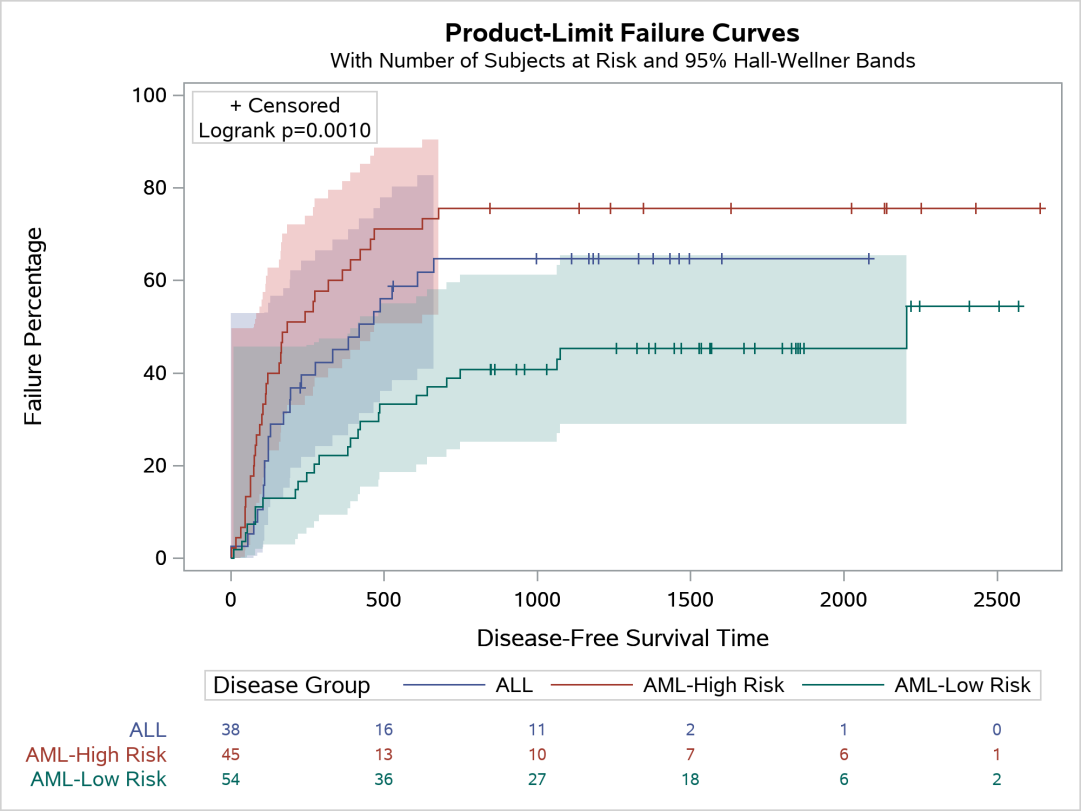
The following steps continue using the *tpl.tpl* file, but this time they convert the Y-axis scale to percentages:

```
data _null_;
   infile 'tpl.tpl' end=eof;
   input;
   if _n_ eq 1 then call execute('proc template;');
   _infile_ = tranwrd(_infile_, 'viewmax=1', 'viewmax=100');
   _infile_ = tranwrd(_infile_, 'tickvaluelist=(0 .2 .4 .6 .8 1.0)',
                  'tickvaluelist=(0 20 40 60 80 100)');
   _infile_ = tranwrd(_infile_, '1-', '100-100*');
   _infile_ = tranwrd(_infile_, 'Failure Probability', 'Failure Percentage');
   call execute(_infile_);
   if eof then call execute('quit;');
run;

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

The *tpl.tpl* file was not changed by the previous customization, so you can process it again to perform a new template modification. The DATA step changes the option VIEWMAX=1 to VIEWMAX=100; changes the

tick values to range from 0 to 100; and changes the expression for many Y-axis options, such as Y=EVAL(1–SURVIVAL) in the STEPPLOT statement, to Y=EVAL(100–100*SURVIVAL). The output data object variables for the Y axis are on a survival scale of 0 to 1. Subtracting the Y value from 1 transforms the scale to a failure scale. Subtracting 100 times the survival value from 100 converts the scale to a percentage scale. Finally, the DATA step changes the Y-axis label. Again, you cannot write code like this without first looking at the template that is stored in the *tpl.tpl* file. For example, you must correctly specify the case and spacing when you specify the string that is to be translated. The graph is displayed in Figure 23.19.

Figure 23.19 Percentages on the Y Axis



Controlling the Survival Plot by Modifying Graph Templates

The preceding section illustrates the PLOTS= options for controlling the survival plot. If you need to make modifications that are not shown in that section, this section shows how to modify the survival plot by using macros and macro variables to modify graph templates.

The Modularized Templates

SAS provides the two templates that are used to make the survival plot in a modularized form.[2] The modularized version of the two survival plot templates is available in the SAS sample library and is on the Web at http://support.sas.com/documentation/onlinedoc/stat/ex_code/151/templft.html.

The file defines the macro %ProvideSurvivalMacros, which defines a series of macros and macro variables. The %ProvideSurvivalMacros macro contains a %GLOBAL statement, a series of %LET statements, and several macro definitions. It ends with a call to the %CompileSurvivalTemplates macro (which is defined inside the %ProvideSurvivalMacros macro), which compiles the two survival plot templates.[3] By using these macros and macro variables, you can easily specify single changes that modify both templates. All the statements in this file are displayed and explained in more detail in the section "Graph Templates, Macros, and Macro Variables" on page 924.

The %ProvideSurvivalMacros macro provides a way to provide (and in subsequent steps restore) the default macros and macro variables. The macros and macro variables are designed so that you can make most changes by submitting just a few lines of SAS code. Hence, you should not modify any of the statements while they are inside the %ProvideSurvivalMacros macro. Rather, you should use this macro only to provide all the default macros and macro variables. You should modify the individual macros and macro variables outside the context of the %ProvideSurvivalMacros macro.[4] The reasons for this will become clearer as you work through the examples. Before you modify anything, you must submit the %ProvideSurvivalMacros macro definition from the sample library to SAS. You can both store the macros in a temporary file and submit them to SAS by submitting the following statements:

```
data _null_;
   %let url = //support.sas.com/documentation/onlinedoc/stat/ex_code/151;
   infile "http:&url/templft.html" device=url;

   file 'macros.tmp';
   retain pre 0;
   input;
   _infile_ = tranwrd(_infile_, '&amp;', '&');
   _infile_ = tranwrd(_infile_, '&lt;' , '<');
   if index(_infile_, '</pre>') then pre = 0;
   if pre then put _infile_;
   if index(_infile_, '<pre>')  then pre = 1;
run;

%inc 'macros.tmp' / nosource;
```

[2] The two templates that PROC LIFETEST uses are named **Stat.Lifetest.Graphics.ProductLimitSurvival** and **Stat.Lifetest.Graphics.ProductLimitSurvival2**.

[3] 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 statements. That is not convenient with the autocall library.

[4] However, there might be something that you *always* want to change. For example, if you always want the survival plot to be entitled 'Kaplan-Meier Plot', then you can modify the title once inside the %ProvideSurvivalMacros macro. This is not illustrated in this chapter. All examples illustrate ad hoc changes that are made outside the context of the %ProvideSurvivalMacros macro.

You might need to change the file name here and elsewhere in this chapter, depending on your default working directory, your operating system, and the method you use to run SAS. For more information about specifying file names and directories, see the *SAS Companion* documentation for your operating system.

The TRANWRD function calls change the HTML coding of the ampersand (`'&'`) and the less than sign `'<'`) to the actual characters (`'&'` and `'<'`).

Submitting these statements only defines the %ProvideSurvivalMacros macro. It does not make any of its component macros and macro variables available. The URL macro variable is used to avoid an overly long INFILE statement.

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

```
%ProvideSurvivalMacros
```

Running this macro provides the default macros and macro variables (or restores them if you have previously submitted the %ProvideSurvivalMacros macro).[5] The %ProvideSurvivalMacrosmacro also runs the %CompileSurvivalTemplates macro and hence replaces any compiled survival plot templates that you might have created in the past. You can recompile the templates by submitting the following macro:

```
%CompileSurvivalTemplates
```

This macro runs PROC TEMPLATE and compiles the templates from all the macros and macro variables in the %ProvideSurvivalMacros macro along with any that you modified. Running this macro produces 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 962. 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 delete the modified templates (so that SAS can only find the original templates) by running the following step:

```
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 from the item store sasuser.templat. You can omit the STORE= option if you are using the default ODS path, but it is good practice to explicitly control which templates are deleted. Deleting the compiled templates does not change any of the macros or macro variables. Only the compiled templates (not the macros or macro variables) affect the graph when you run PROC LIFETEST. For more information about compiled templates, item stores, and cleanup, see the section "SAS Item Stores" on page 962.

[5] Semicolons are not needed after a macro call like this one, so they are not used in these examples.

Changing the Plot Title

Begin by including the macros that you copied in the section "The Modularized Templates" on page 895.

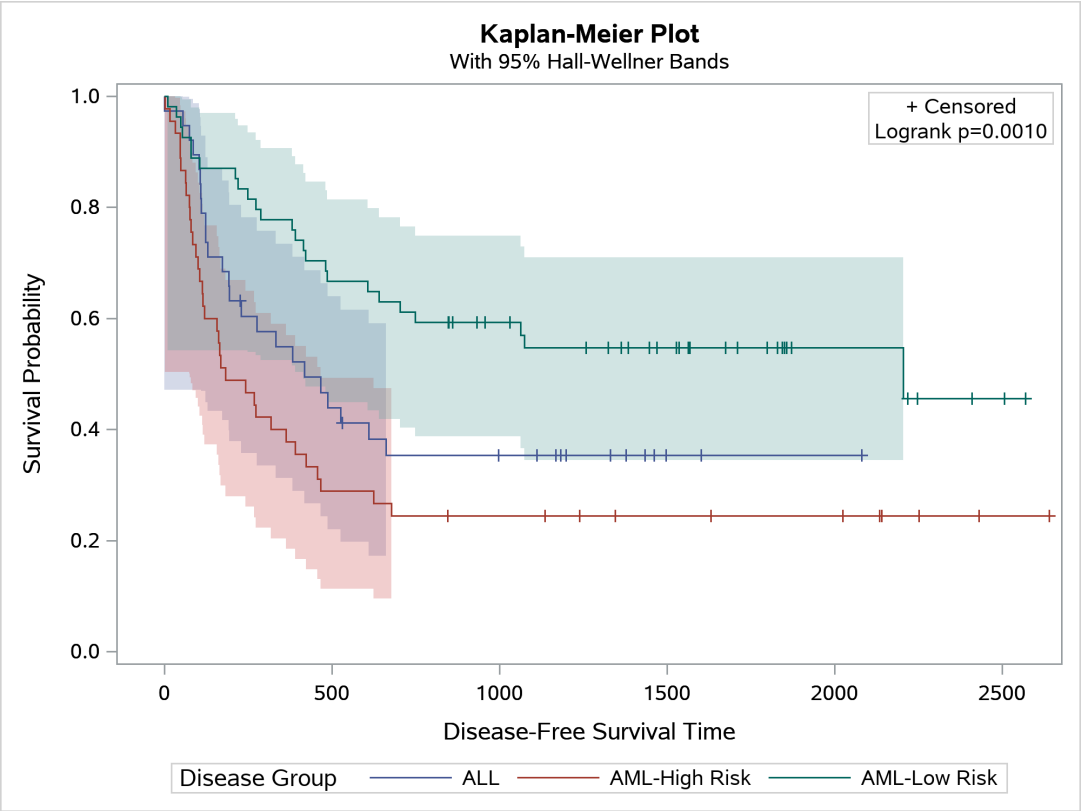
Here is a simple, complete program (except for retrieving the %ProvideSurvivalMacros macro from the sample library) with setup, macro variable modifications to change the title, and cleanup:

```
/*-- Original Macro Variable Definitions ---------------------------------
%let TitleText0 = METHOD " Survival Estimate";
%let TitleText1 = &titletext0 " for " STRATUMID;
%let TitleText2 = &titletext0 "s";
-------------------------------------------------------------------------*/

                                        /* Make the macros and macro      */
%ProvideSurvivalMacros                  /* variables available.           */

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

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

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

%ProvideSurvivalMacros                  /* Optionally restore the default */
                                        /* macros and macro variables.    */

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

The results are displayed in Figure 23.20. You can see that the graph title is now 'Kaplan-Meier Plot'.

There are multiple title macro variables because two different types of plots are 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 %ProvideSurvivalMacros macro. You must 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 what you need. Alternatively, when you know the number of strata, you can modify only TitleText1 or TitleText2.

Figure 23.20 Kaplan-Meier Plot Title Modification



Modifying the Y Axis

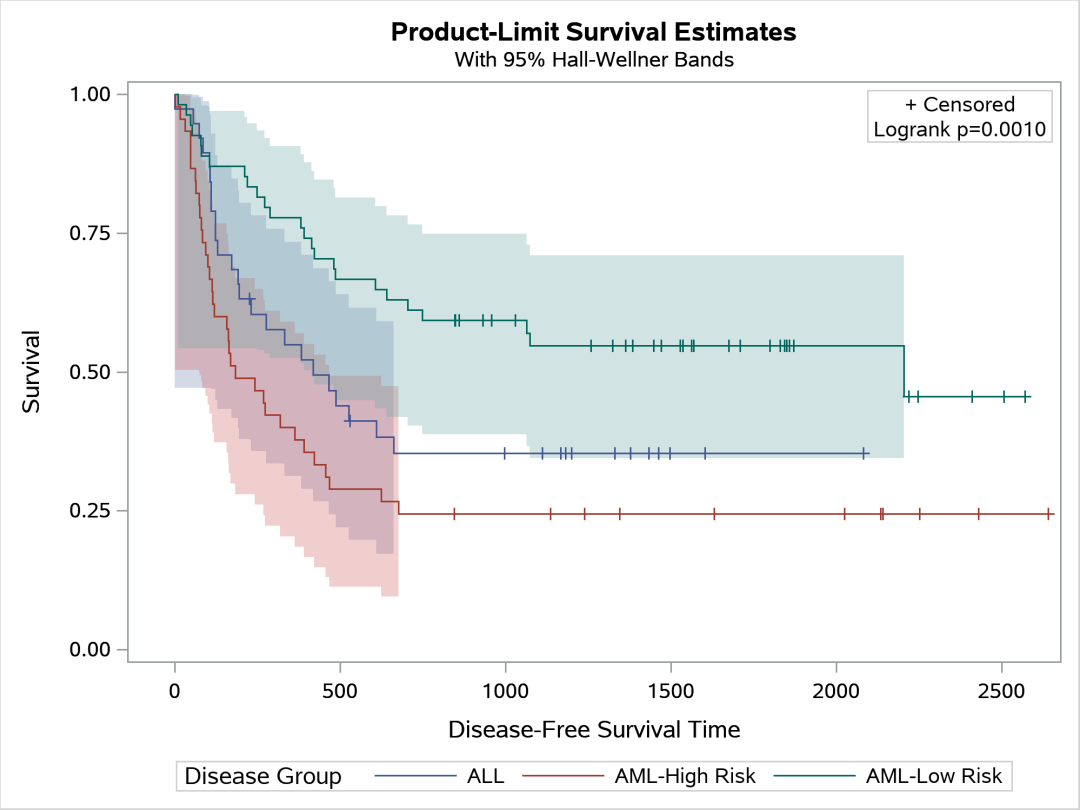
Begin by including the macros that you copied in the section "The Modularized Templates" on page 895.

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':

```
/*-- Original Macro Variable Definitions ---------------------------------
%let yOptions    = label="Survival Probability" shortlabel="Survival"
                   linearopts=(viewmin=0 viewmax=1
                               tickvaluelist=(0 .2 .4 .6 .8 1.0));
-------------------------------------------------------------------------*/

%ProvideSurvivalMacros

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

%CompileSurvivalTemplates

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

The results are displayed in Figure 23.21.

Figure 23.21 Y-Axis Modification

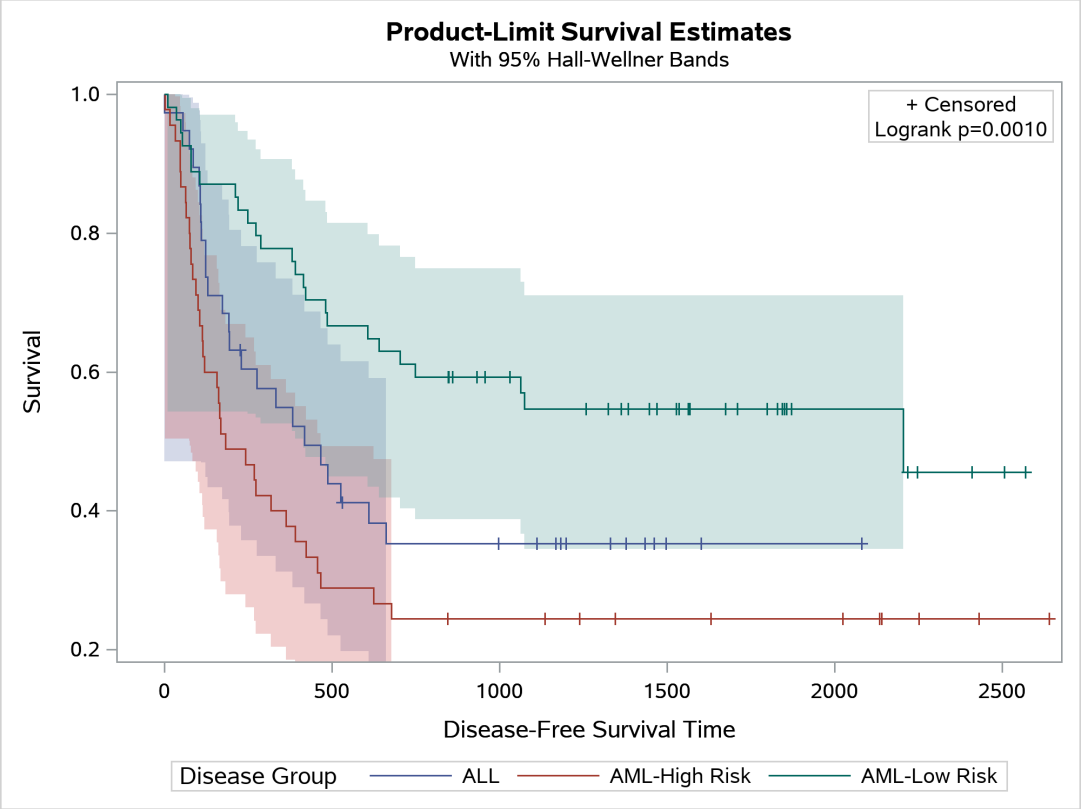


The following statements modify the Y axis so that tick marks start at 0.2:

```
%ProvideSurvivalMacros

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

%CompileSurvivalTemplates

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

You only need to change the value of the VIEWMIN= option, in this case from 0 to 0.2. You do not need to modify the tick value list. The VIEWMIN= option (not the tick value list) controls the smallest value shown on the axis. The results are displayed in Figure 23.22.

Figure 23.22 Y Axis, First Tick Change



Changing the Line Thickness

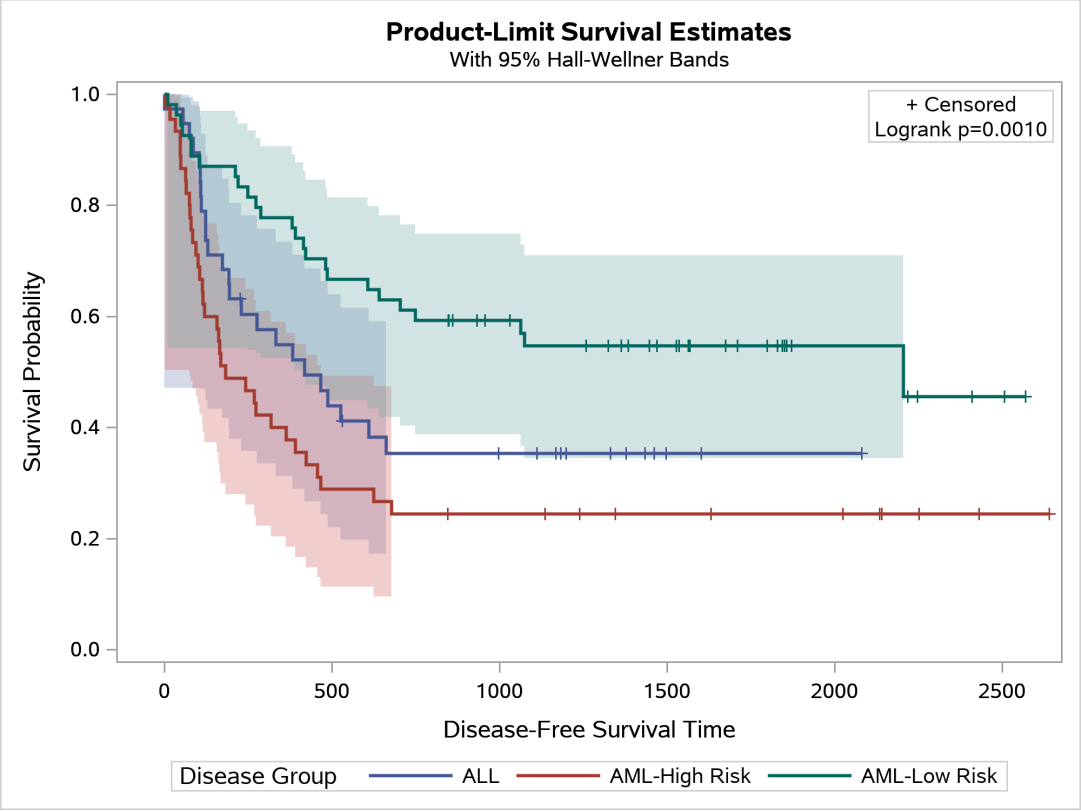
Begin by including the macros that you copied in the section "The Modularized Templates" on page 895.

The following steps modify the line thickness for the step functions in the survival plot:

```
%ProvideSurvivalMacros

%let StepOpts = lineattrs=(thickness=2.5);

%CompileSurvivalTemplates

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

The results are displayed in Figure 23.23.

Figure 23.23 Changing Line Thickness



By default, the StepOpts macro variable is null.

Changing the Group Color

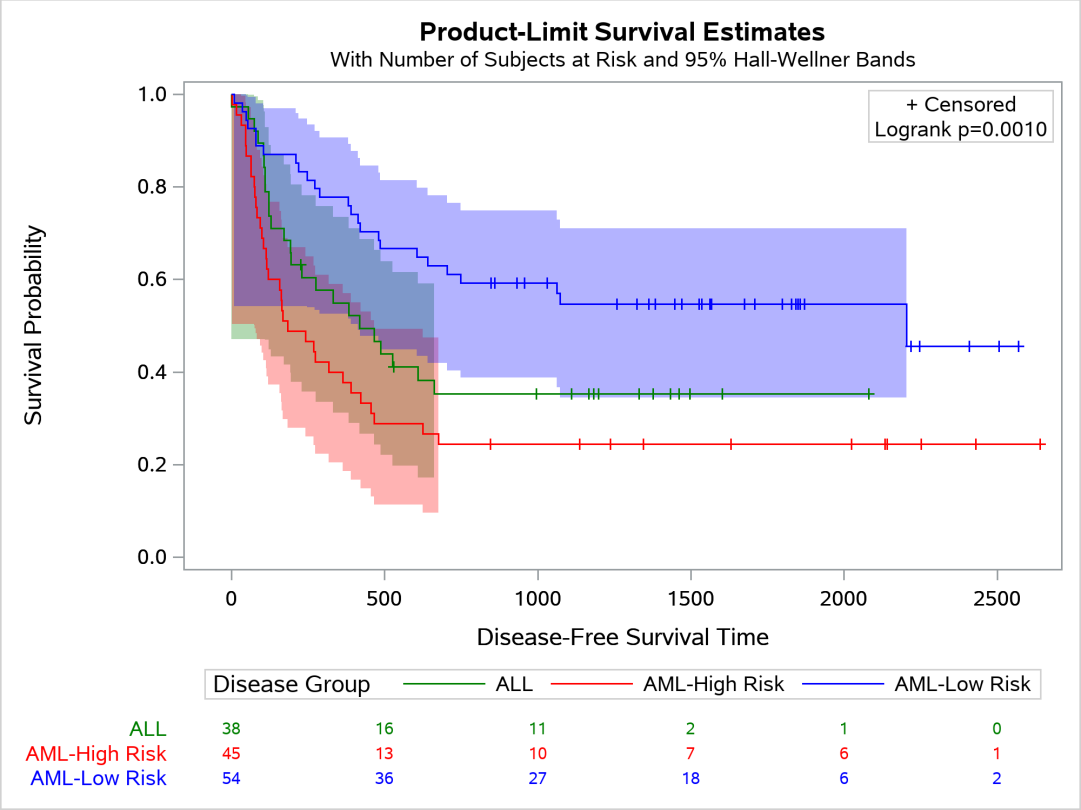
Begin by including the macros that you copied in the section "The Modularized Templates" on page 895.

SAS styles control the colors displayed in graphs. The style elements **GraphData1**, **GraphData2**, ..., **GraphData12** control the appearance of groups of observations such as the survival step plots in the Kaplan-Meier plot. You can override these colors by using the GraphOpts macro variable (which is null by default). 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 a pure green, red, and blue as follows:

```
%ProvideSurvivalMacros

%let GraphOpts = DataContrastColors=(green red blue)
                 DataColors=(green red blue);

%CompileSurvivalTemplates

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

The results are displayed in Figure 23.24. The DATACONTRASTCOLORS= option specifies the contrast colors, which are used for markers and lines. The DATACOLORS= option specifies the colors, which are used for shaded areas such as confidence bands.

Figure 23.24 Named Color Specifications



The original colors (as shown in Figure 23.47) are more subtle than those shown in Figure 23.24. If you want to change the order of the original colors by using this approach, then you need to know what they are so that you can specify them. The graph colors for the HTMLBlue and Statistical styles are extracted from the style in the section "Displaying a Style and Extracting Color Lists" on page 951 and displayed in Figure 23.50. The section "Modifying Color Lists" on page 954 shows you how to change the graph template to specify the original colors in a different order. The section "Swapping Colors among Style Elements" on page 955 shows you how to use a macro to change a style template to specify the original colors in a different order (without having to extract and specify the color names).

Changing the Line Pattern

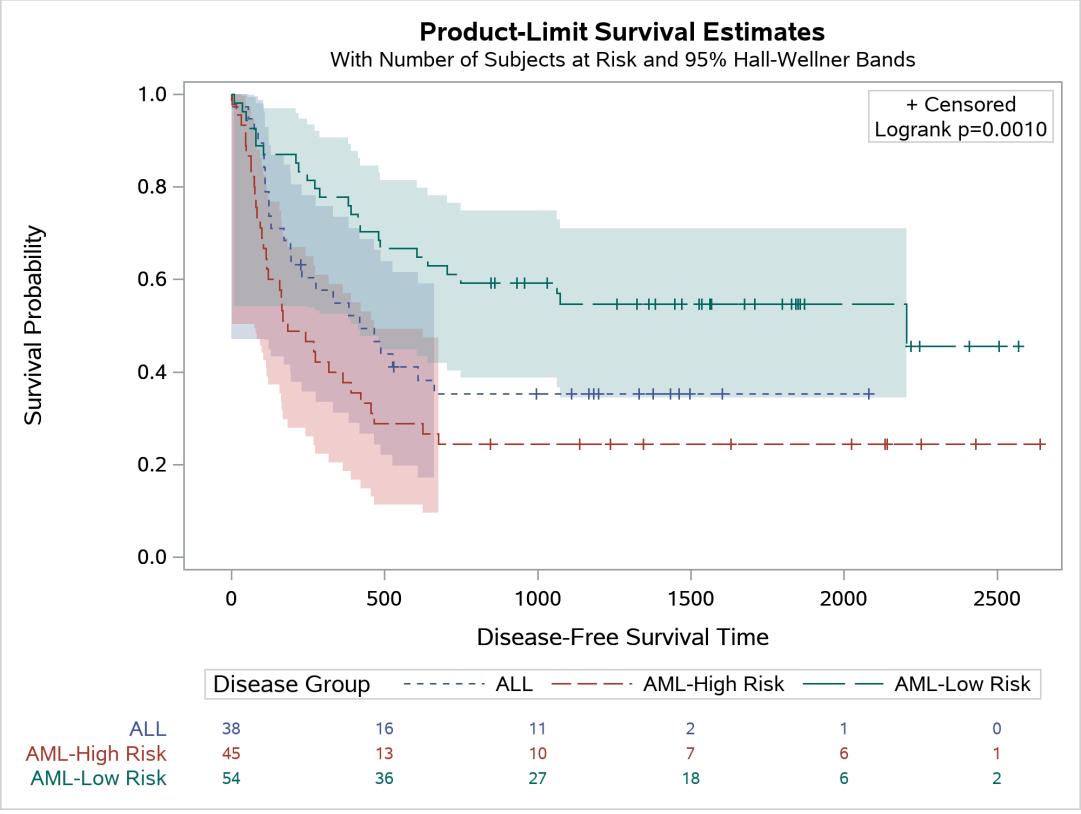
Begin by including the macros that you copied in the section "The Modularized Templates" on page 895.

You can change the line patterns as follows:

```
%ProvideSurvivalMacros

%let GraphOpts = attrpriority=none
                 DataLinePatterns=(ShortDash MediumDash LongDash);

%CompileSurvivalTemplates

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

The results are displayed in Figure 23.25.

Figure 23.25 Changing the Line Patterns



Other values for the DATALINEPATTERNS= option are provided in the section "The Macro Variables" on page 926. You must use the option ATTRPRIORITY=NONE when you want to have varying line patterns in an ATTRPRIORITY=COLOR style like HTMLBlue or Pearl. In an ATTRPRIORITY=COLOR style, groups are not distinguished by line patterns, and the line patterns for second and subsequent groups match the line pattern for the first group.

Changing the Font

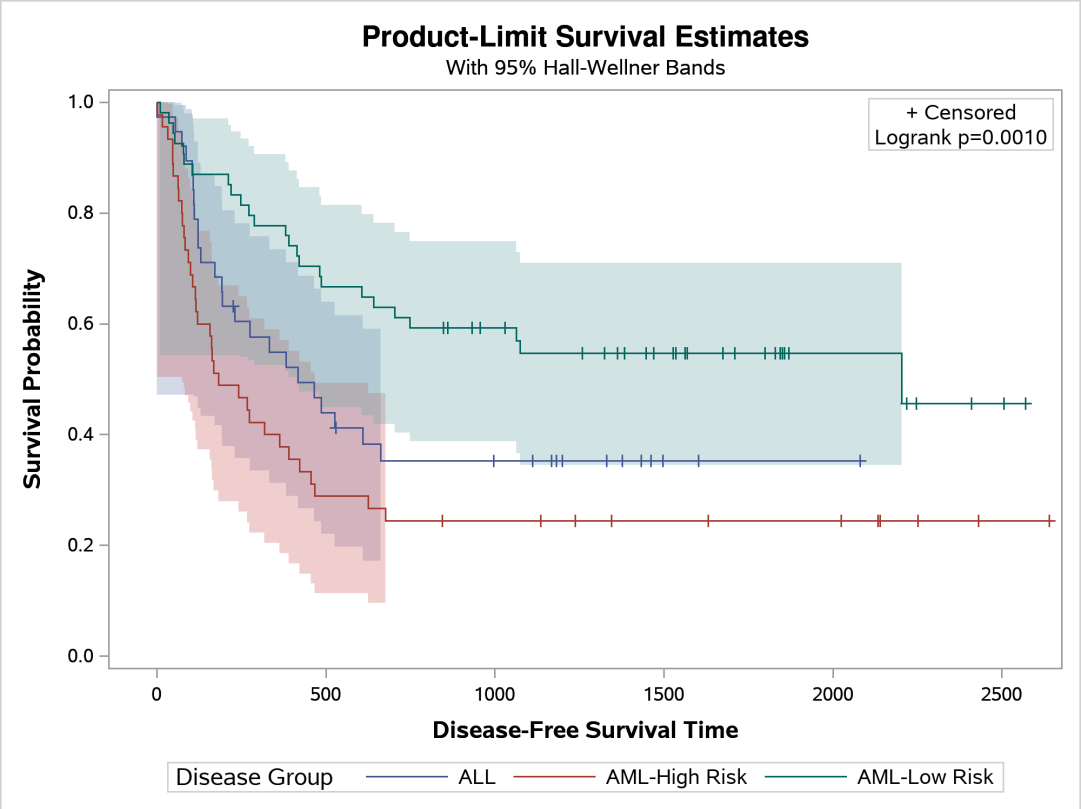
Begin by including the macros that you copied in the section "The Modularized Templates" on page 895.

You can change the Y-axis, X-axis, and title fonts as follows:

```
%ProvideSurvivalMacros

/*-- Original Macro Variable Definitions ---------------------------------
%let TitleText0 = METHOD " Survival Estimate";
%let TitleText1 = &titletext0 " for " STRATUMID;
%let TitleText2 = &titletext0 "s";
%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 tatters    = textattrs=(size=12pt weight=bold family='arial');
%let TitleText0 = METHOD " Survival Estimate";
%let TitleText1 = &titletext0 " for " STRATUMID / &tatters;
%let TitleText2 = &titletext0 "s" / &tatters;

%let yOptions   = label="Survival Probability"
                  shortlabel="Survival"
                  labelattrs=(size=10pt weight=bold)
                  tickvalueattrs=(size=8pt)
                  linearopts=(viewmin=0 viewmax=1
                              tickvaluelist=(0 .2 .4 .6 .8 1.0));

%let xOptions   = shortlabel=XNAME
                  offsetmin=.05
                  labelattrs=(size=10pt weight=bold)
                  tickvalueattrs=(size=8pt)
                  linearopts=(viewmax=MAXTIME tickvaluelist=XTICKVALS
                              tickvaluefitpolicy=XTICKVALFITPOL);

%CompileSurvivalTemplates

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

The results are displayed in Figure 23.26.

Figure 23.26 Changing the Fonts



Font options include the following:

COLOR=*style-reference* | *color*
FAMILY=*style-reference* | *'string'*
SIZE=*style-reference* | *dimension*
STYLE=*style-reference* | **NORMAL** | **ITALIC**
WEIGHT=*style-reference* | **NORMAL** | **BOLD**

Fonts vary from installation to installation. Sample font strings include: 'Times New Roman', 'Courier New', 'Arial', and 'Calibri'. For more information about text and label attribute options, see *SAS Graph Template Language: Reference*. For information about changing fonts in ODS styles, see the section "Displaying a Style and Extracting Font Information" on page 957. ODS Graphics can use a single style element in more than one place in a graph; this example shows how to change individual graph components.

Changing the Legend and Inset Position

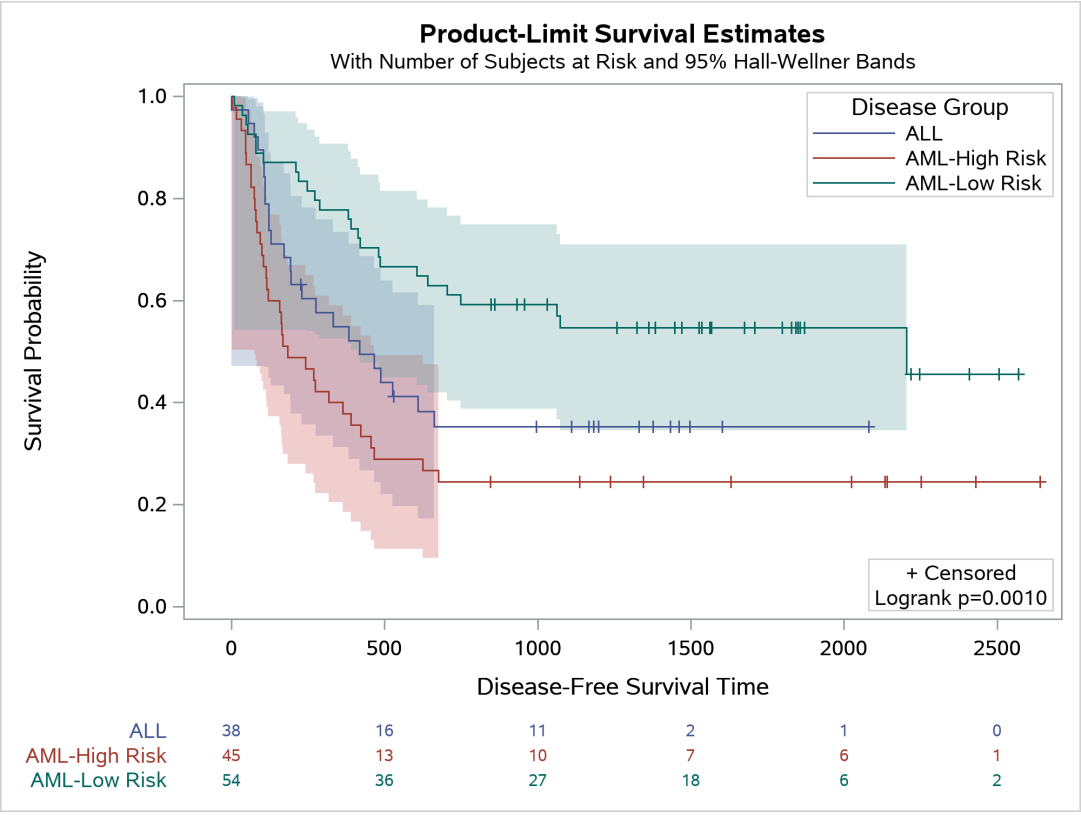
Begin by including the macros that you copied in the section "The Modularized Templates" on page 895.

This example shows you how to move the legend inside the plot (to the top right) and move the homogeneity test and censored value legend to the bottom right of the plot:

```
%ProvideSurvivalMacros
/*-- Original Macro Variable Definitions ---------------------------------
%let InsetOpts  = autoalign=(TOPRIGHT BOTTOMLEFT TOP BOTTOM)
                  border=true BackgroundColor=GraphWalls:Color Opaque=true;
%let LegendOpts = title=GROUPNAME location=outside;
-------------------------------------------------------------------------*/

%let InsetOpts  = autoalign=(BottomRight)
                  border=true BackgroundColor=GraphWalls:Color Opaque=true;
%let LegendOpts = title=GROUPNAME location=inside across=1 autoalign=(TopRight);
%CompileSurvivalTemplates

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

This example shows you how to replace the AUTOALIGN=(TOPRIGHT BOTTOMLEFT TOP BOTTOM) option in the macro variable InsetOpts with AUTOALIGN=(BOTTOMRIGHT) and add the AUTOALIGN=(TOPRIGHT) option to the LegendOpts macro variable. You can also add the option ACROSS=1 to the LegendOpts macro variable to stack all legend entries vertically (with just one element in each row). The results are displayed in Figure 23.27.

Figure 23.27 Controlling Legend Placement



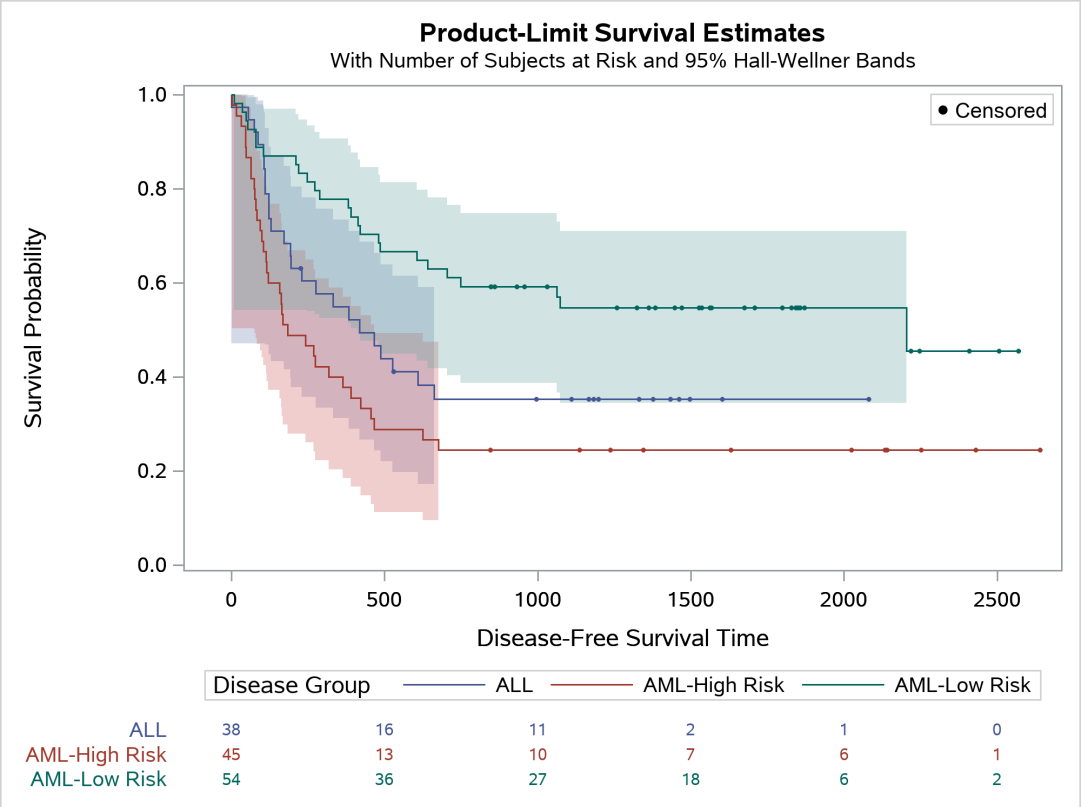
Changing How the Censored Points Are Displayed

Begin by including the macros that you copied in the section "The Modularized Templates" on page 895.

By default, PROC LIFETEST displays a plus sign to indicate censoring. This example illustrates how to change the plus sign to a small filled circle in both the step plots and the inset box. The following steps change the template and create Output 23.28:

```
/*-- Original Macro Variable Definitions ---------------------------------
%let Censored   = markerattrs=(symbol=plus);
%let CensorStr  = "+ Censored";
-------------------------------------------------------------------------*/

%ProvideSurvivalMacros

%let censored  = markerattrs=(symbol=circlefilled size=3px);
%let censorstr = "(*ESC*){Unicode '25cf'x} Censored"
                 / textattrs=GraphValueText(family=GraphUnicodeText:FontFamily);

%CompileSurvivalTemplates

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

Figure 23.28 Survival Plot with a Modified Display of Censoring



The Unicode Consortium (`http://unicode.org/`) provides a list of character codes. Also see Figure 22.2.7 in Chapter 22, "ODS Graphics Template Modification," for information about the Unicode specification for other markers. Although some Unicode characters are supported in some fonts, you should always specify a Unicode font when using special characters.

Adding a Y-Axis Reference Line

Begin by including the macros that you copied in the section "The Modularized Templates" on page 895.

You can add a horizontal reference line to the survival plot by adding the following statement to the template:

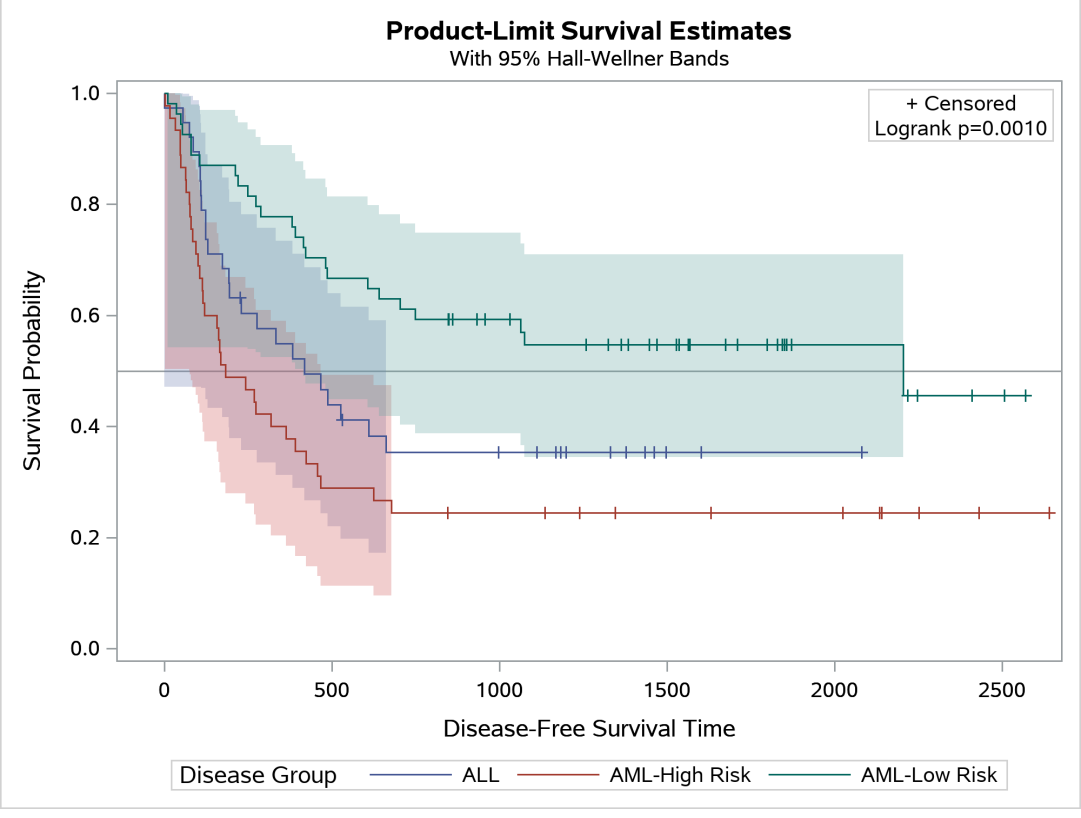
```
referenceline y=0.5;
```

You can do this by using the %StmtsTop macro. By default, this macro is empty. You can use the %StmtsTop macro to add new statements to the beginning of the block of statements that define the appearance of the graph. In contrast, you can use the %StmtsBottom macro to provide statements at the end of the statement block. ODS Graphics draws statements in the order in which they appear; therefore, reference lines should be drawn first so they do not obscure other parts of the graph.

The following step creates the plot in Figure 23.29:

```
%ProvideSurvivalMacros
%macro StmtsTop;
   referenceline y=0.5;
%mend;
%CompileSurvivalTemplates

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

Figure 23.29 Horizontal Reference Line



Adding Y-Axis Grid Lines

Begin by including the macros that you copied in the section "The Modularized Templates" on page 895.

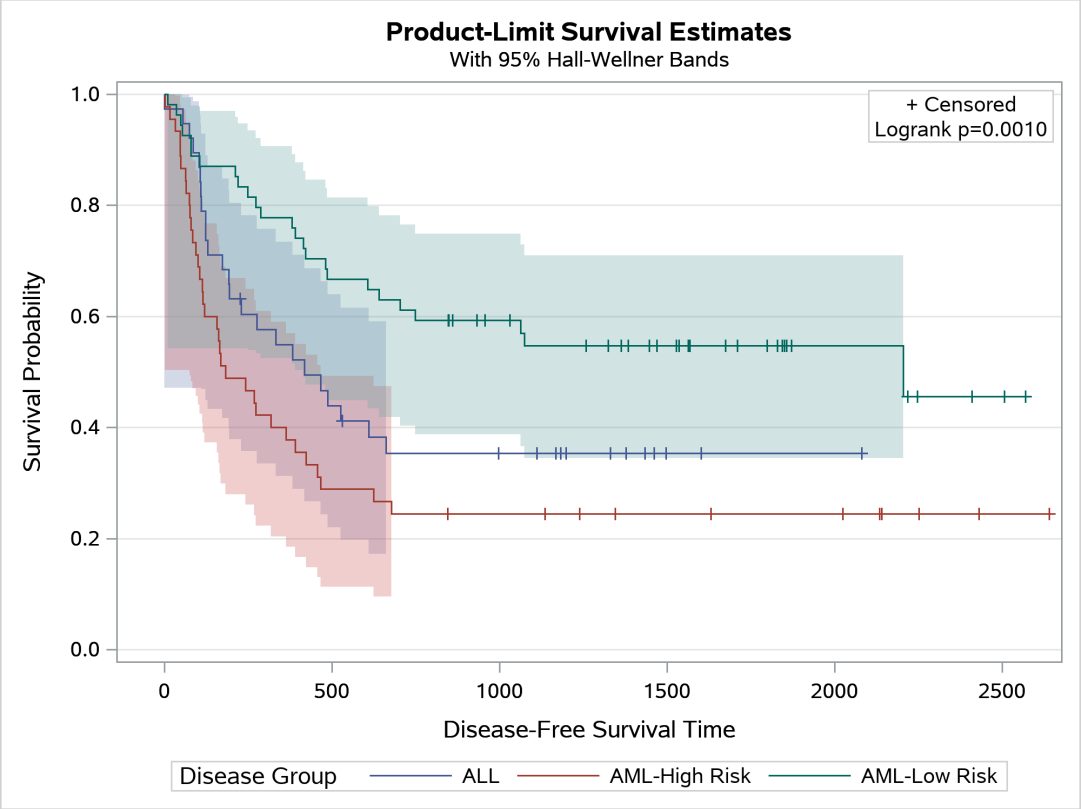
You can add horizontal grid lines to the survival plot by adding the following option to the Y-axis options:

```
griddisplay=on;
```

You can do this by modifying the yOptions macro variable. The following steps create the plot in Figure 23.30:

```
%ProvideSurvivalMacros

%let yoptions = &yoptions griddisplay=on;

%CompileSurvivalTemplates

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

Figure 23.30 Horizontal Grid Lines



Changing the Homogeneity Test Inset

Begin by including the macros that you copied in the section "The Modularized Templates" on page 895.

This example modifies the contents of the %pValue macro. The original %pValue macro definition is as follows:

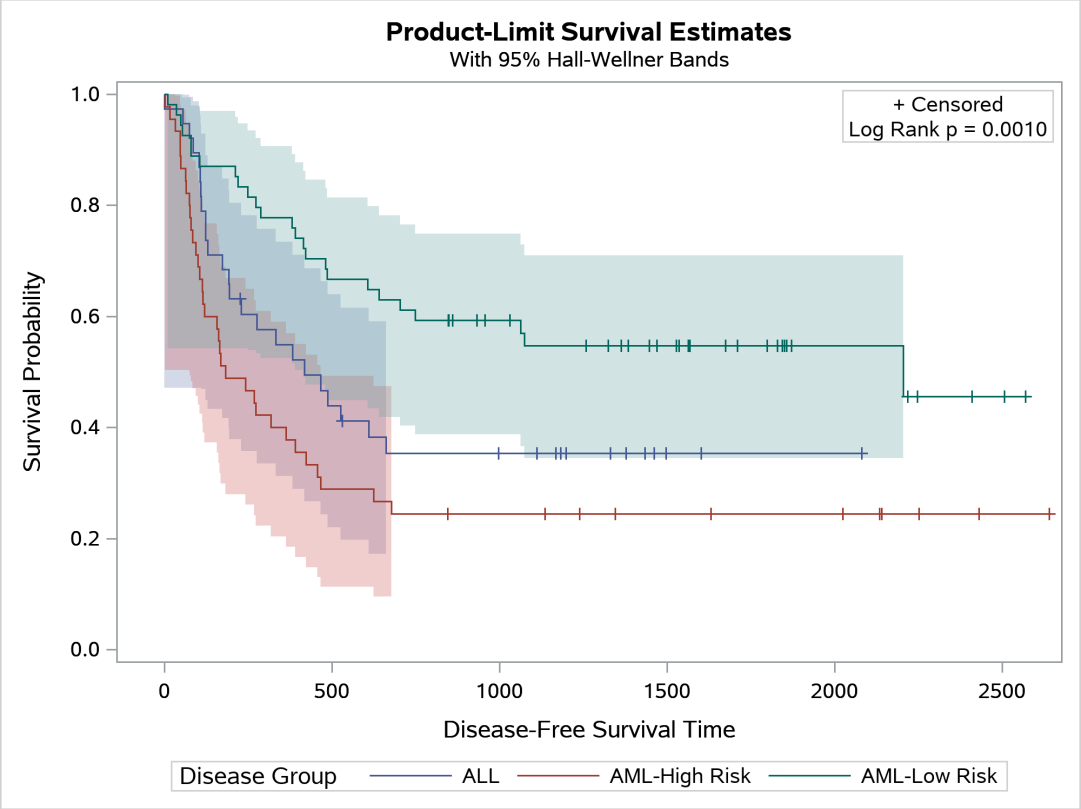
```
%macro pValue;
   if (PVALUE < .0001)
      entry TESTNAME " p " eval (PUT(PVALUE, PVALUE6.4));
   else
      entry TESTNAME " p=" eval (PUT(PVALUE, PVALUE6.4));
   endif;
%mend;
```

The following example directly specifies the test name (replacing the internal name 'Logrank' with 'Log Rank') and adds blank spaces around the equal sign:

```
%ProvideSurvivalMacros

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

%CompileSurvivalTemplates

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

The results are displayed in Figure 23.31.

Figure 23.31 Cosmetic Inset Entry Change



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

```
%ProvideSurvivalMacros

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

Changing the Second Title and Adding a Footnote

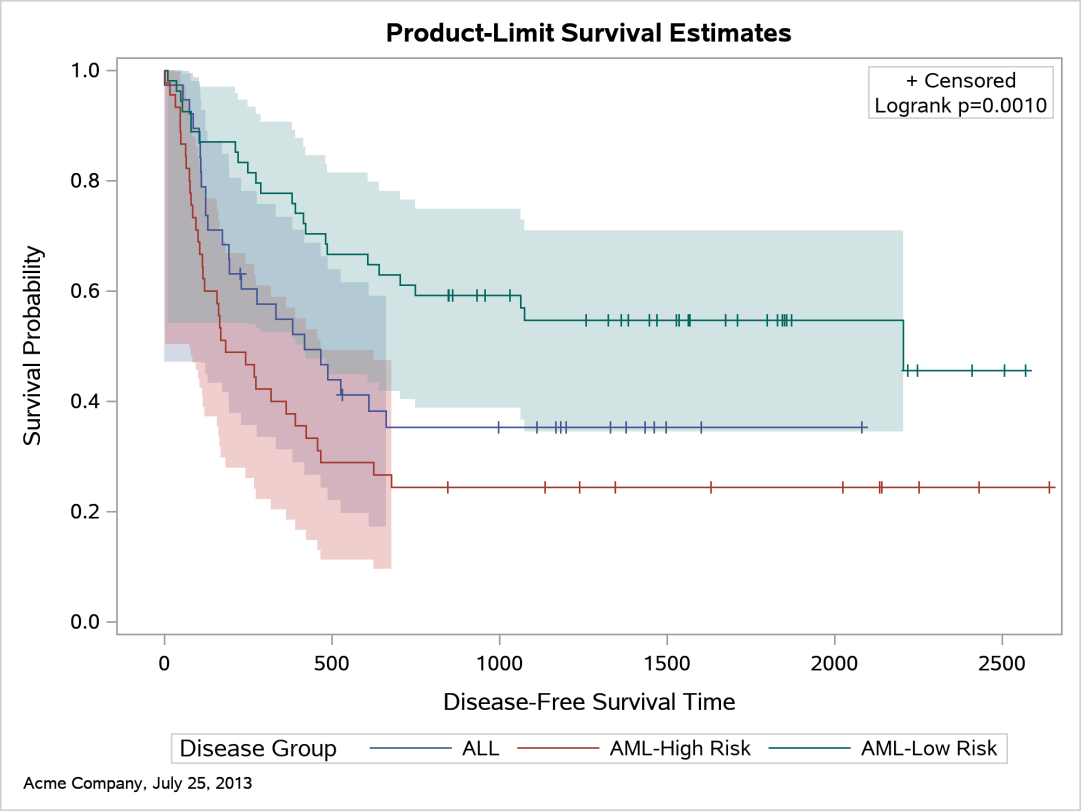
Begin by including the macros that you copied in the section "The Modularized Templates" on page 895.

The following steps add an ENTRYFOOTNOTE statement to the %StmtsBeginGraph macro and suppress the second title:

```
%ProvideSurvivalMacros

%let ntitles = 1;
%macro StmtsBeginGraph;
   entryfootnote halign=left "Acme Company, %sysfunc(date(),worddate.)" /
                 textattrs=GraphDataText;
%mend;

%CompileSurvivalTemplates

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

The results are displayed in Figure 23.32.

Figure 23.32 Footnote but No Second Title



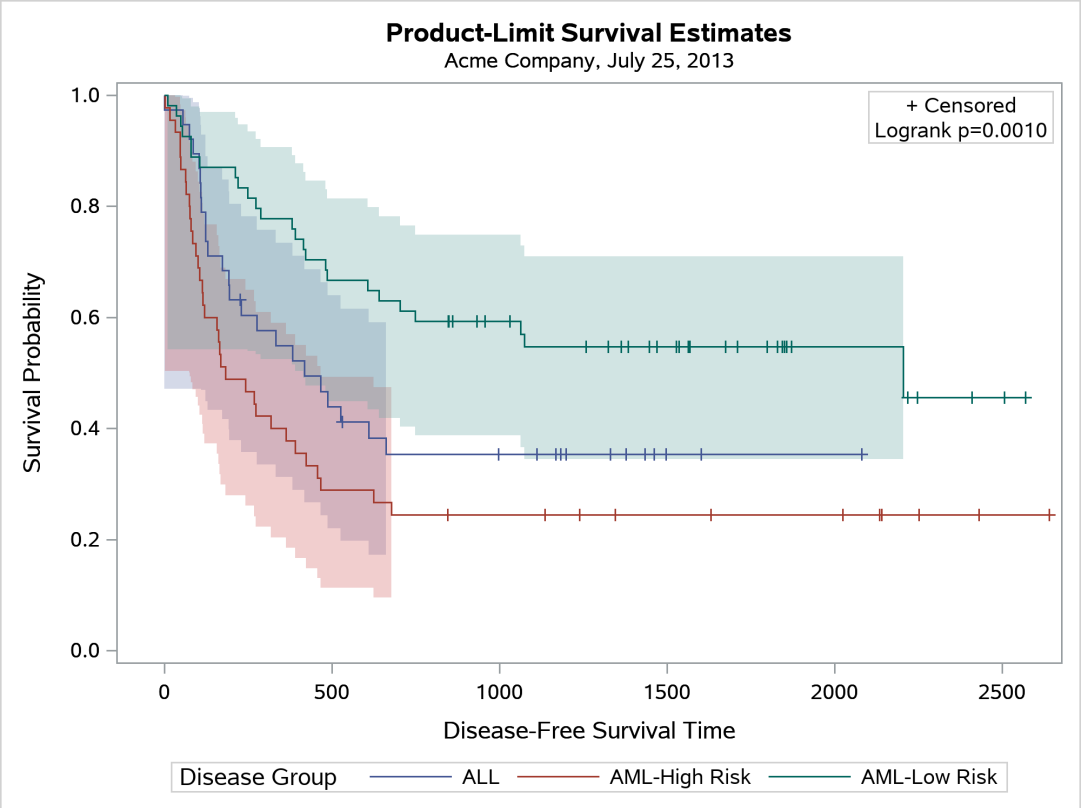
By default, the value of the nTitles macro variable is 2, and all titles are displayed. Setting nTitles to 1 suppresses the second title. You can add titles or footnotes to the plot by adding them to the %StmtsBeginGraph macro. This example adds a footnote that consists of a company name followed by the current date, formatted by using the WORDDATE format. The **GraphDataText** style element is used; it has a smaller font than the default style element, **GraphFootnoteText**.

The following steps replace the second title rather than adding a footnote:

```
%ProvideSurvivalMacros

%let ntitles = 1;

%macro StmtsBeginGraph;
entrytitle "Acme Company, %sysfunc(date(),worddate.)" /textattrs=GraphValueText;
%mend;

%CompileSurvivalTemplates

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

The results are displayed in Figure 23.33. You can find the original definition of the second title in the %CompileSurvivalTemplates macro in the section "The Larger Macros" on page 929.

Figure 23.33 Replacing the Second Title



Adding a Custom Header above the At-Risk Table

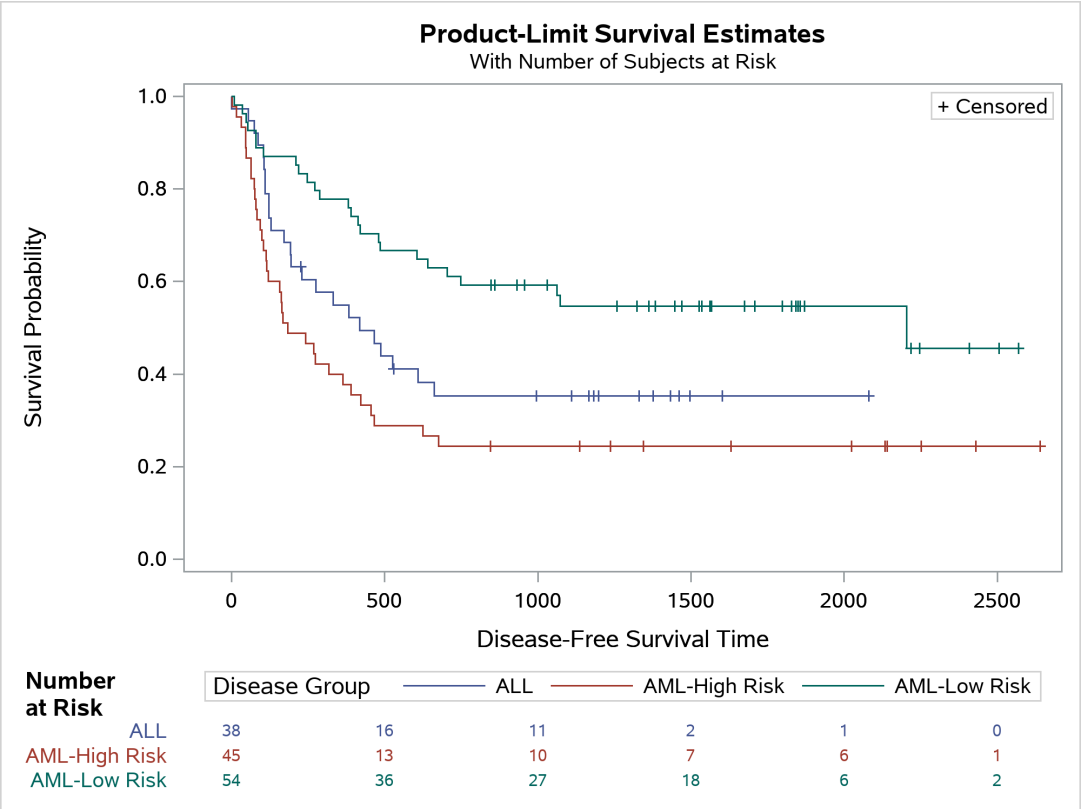
Begin by including the macros that you copied in the section "The Modularized Templates" on page 895.

The following steps add a header to the at-risk table by using the %StmtsBeginGraph macro and a DRAWTEXT statement:

```
%ProvideSurvivalMacros

%macro StmtsBeginGraph;
  * Coordinates are ad hoc, and there are many available coordinate systems.
    See the documentation for more information.;
  drawtext textattrs=(weight=Bold) 'Number at Risk' / x=5 y=14 width=9;
%mend;

%CompileSurvivalTemplates

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

The %StmtsBeginGraph macro adds statements near the top of the template. The DRAWTEXT statement adds text to the graph. The width of 9% was chosen so that the header would break between `'Number'` and `'at Risk'`. The other statements in the DRAW family include BEGINPOLYGON, BEGINPOLYLINE, DRAWARROW, DRAWIMAGE, DRAWLINE, DRAWOVAL, and DRAWRECTANGLE. They provide the same capabilities in graph templates that SG annotation provides in the SG procedures. For more information, see *SAS Graph Template Language: User's Guide*.

Figure 23.34 At-Risk Table Header



Adding a Small Inset Table with Event Information

Begin by including the macros that you copied in the section "The Modularized Templates" on page 895.

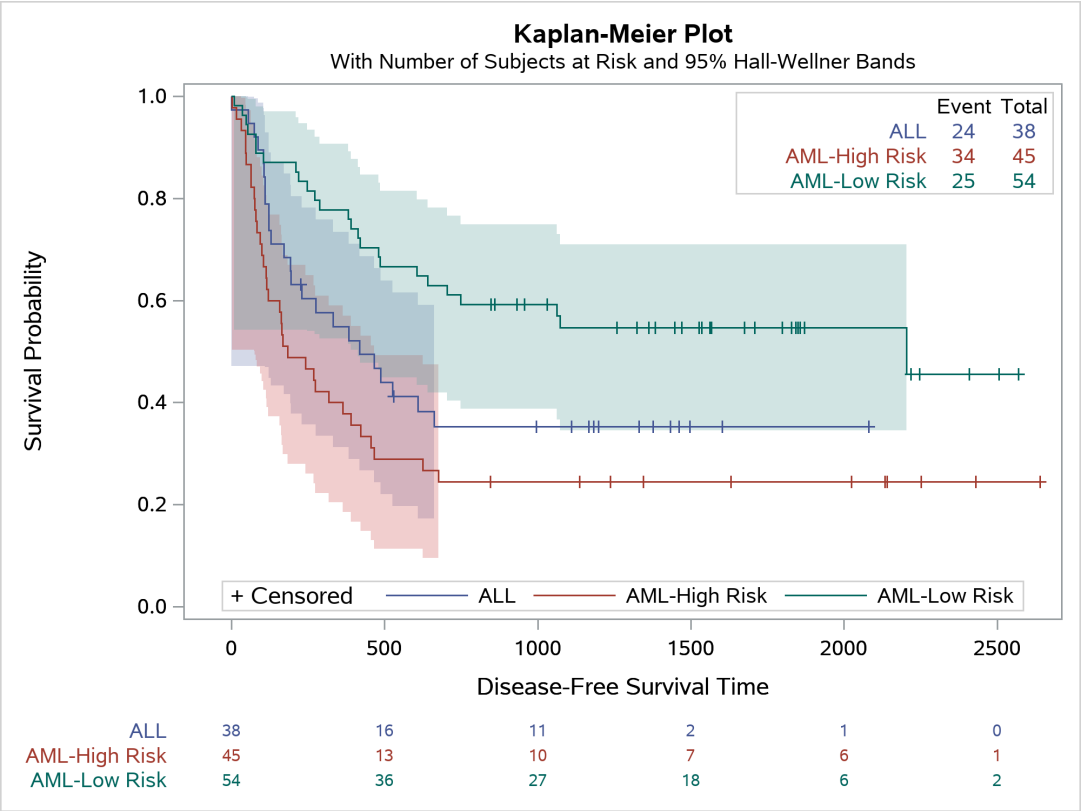
This example shows you how to modify the template to produce the plot displayed in Output 23.35. This new plot has an inset table in the top right corner that shows the number of observations and the number of events in each stratum. The legend has been moved inside the plot and combined with the old inset table that shows the marker for censored observations.[6] Also, the title is changed to 'Kaplan-Meier Plot'.

```
%ProvideSurvivalMacros

%let TitleText2 = "Kaplan-Meier Plot";
%let LegendOpts = title="+ Censored"
                  location=inside autoalign=(Bottom);
%let InsetOpts  = ;

%macro StmtsBottom;
   dynamic %do i = 1 %to 3; StrVal&i NObs&i NEvent&i %end;;
   layout gridded / columns=3 border=TRUE autoalign=(TopRight);
      entry ""; entry "Event"; entry "Total";
      %do i = 1 %to 3;
         %let t = / textattrs=GraphData&i;
         entry halign=right Strval&i &t; entry NEvent&i &t; entry NObs&i &t;
      %end;
   endlayout;
%mend;

%CompileSurvivalTemplates

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

The results are displayed in Figure 23.35.

[6] This legend is wide and might not be displayed if your graph is small. If the legend is not displayed, try increasing the size of the graph by specifying the WIDTH= or HEIGHT= option in the ODS GRAPHICS statement.

Figure 23.35 New Inset Table with Event Information



The macro variable TitleText2, which controls the title for the multiple-strata plot, is changed. You can change all three title macro variables, as is done in the construction of Figure 23.20, or you can change only TitleText2 when you have multiple overlaid strata, as in this example. The LegendOpts macro variable value was changed from TITLE=GROUPNAME LOCATION=OUTSIDE to display the censored value legend in place of the legend title and to display the legend inside the bottom of the plot. When the InsetOpts macro variable is null, the usual inset that contains the censored value and *p*-value is not displayed.

The %StmtsBottom macro (null by default) is replaced with a macro that creates the new inset table. This macro adds statements to the bottom of the templates. If you ignore for a moment most of the options, the core of the generated statements is as follows:

```
dynamic StrVal1 NObs1 NEvent1 StrVal2 NObs2 NEvent2 StrVal3 NObs3 NEvent3;
layout gridded / columns=3;
   entry "";         entry "Event";   entry "Total";
   entry Strval1;    entry NEvent1;   entry NObs1;
   entry Strval2;    entry NEvent2;   entry NObs2;
   entry Strval3;    entry NEvent3;   entry NObs3;
endlayout;
```

The macro first constructs a DYNAMIC statement that includes the names of the dynamic variables that contain some of the results. PROC LIFETEST creates these dynamic variables and sets them to values, but you must declare them in your template before using them. For more information about these dynamic variables, see the section "Additional Dynamic Variables" on page 937. The macro then constructs a 4 × 3 grid that contains a table consisting of a title line and a row for each stratum (which consists of the stratum

label, the number of events, and the total number of subjects). The full layout that the %StmtsBottom macro generates, with all the options, is as follows:

```
dynamic StrVal1 NObs1 NEvent1 StrVal2 NObs2 NEvent2 StrVal3 NObs3 NEvent3;
layout gridded / columns=3 border=TRUE autoalign=(TopRight);
   entry "";
   entry "Event";
   entry "Total";
   entry halign=right Strval1 / textattrs=GraphData1;
   entry NEvent1 / textattrs=GraphData1;
   entry NObs1 / textattrs=GraphData1;
   entry halign=right Strval2 / textattrs=GraphData2;
   entry NEvent2 / textattrs=GraphData2;
   entry NObs2 / textattrs=GraphData2;
   entry halign=right Strval3 / textattrs=GraphData3;
   entry NEvent3 / textattrs=GraphData3;
   entry NObs3 / textattrs=GraphData3;
endlayout;
```

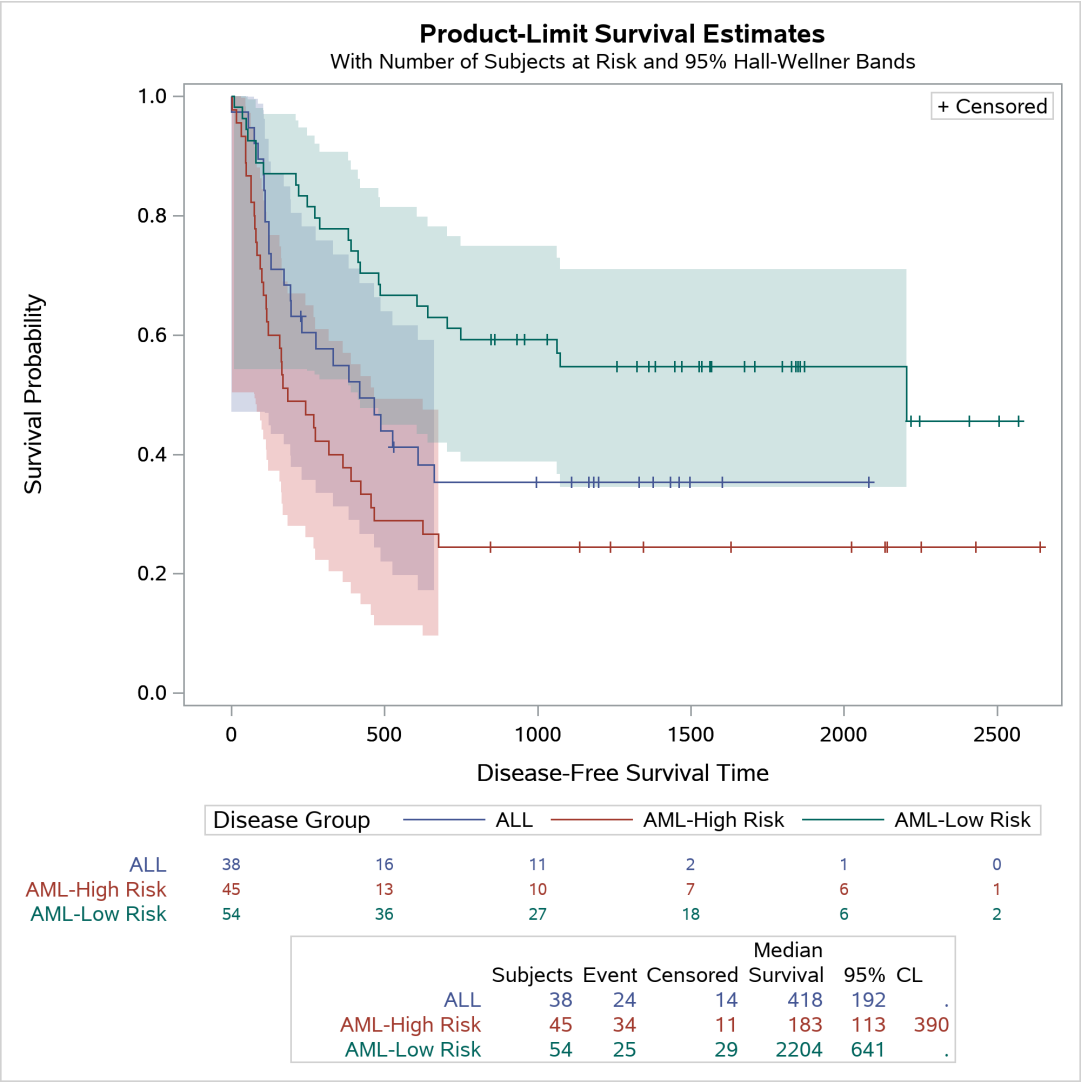
Adding an External Table with Event Information

Begin by including the macros that you copied in the section "The Modularized Templates" on page 895.

This example adds a table to the plot that displays a summary of event information. The following statements create Figure 23.36:

```
%ProvideSurvivalMacros

%let GraphOpts = DesignHeight=DefaultDesignWidth;

%SurvivalSummaryTable

%CompileSurvivalTemplates

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

Figure 23.36 External Event Table



The GraphOpts macro variable specifies the option DESIGNHEIGHT=DEFAULTDESIGNWIDTH. At the default graph size (the size at which the graph is designed), this option sets the graph height to the default graph width of 640 pixels. The macro %SurvivalSummaryTable adds new statements to the graph templates that display the number of subjects, number of events, number of censored observations, median survival time, and 95% confidence limits for the median survival time. For more information about the %SurvivalSummaryTable macro, see the section "Event Table Macros" on page 934.

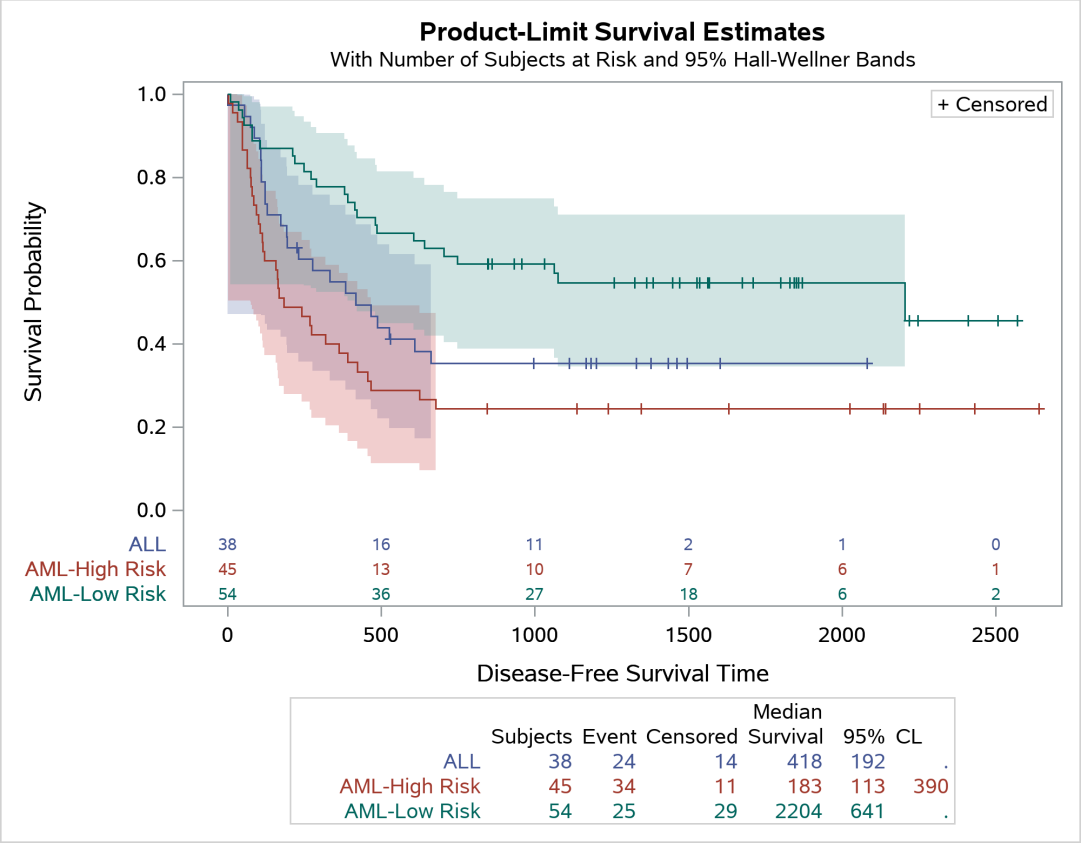
Suppressing the Legend

Begin by including the macros that you copied in the section "The Modularized Templates" on page 895.

The plot in Figure 23.36 has a legend. However, the plot displays values in the tables by using colors that match the colors of the step functions, so you do not need the legend. The next statements show how to remove the legend:

```
%ProvideSurvivalMacros

%let GraphOpts  = DesignHeight=500px;
%let LegendOpts = ;

%SurvivalSummaryTable

%CompileSurvivalTemplates

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

Figure 23.37 Plot with Legend Removed



The legend is suppressed when the LegendOpts macro variable is null. This example also illustrates changing the design height to 500 pixels and moving the at-risk table back inside the body of the plot.

Kaplan-Meier Plot with Event Table and Other Customizations

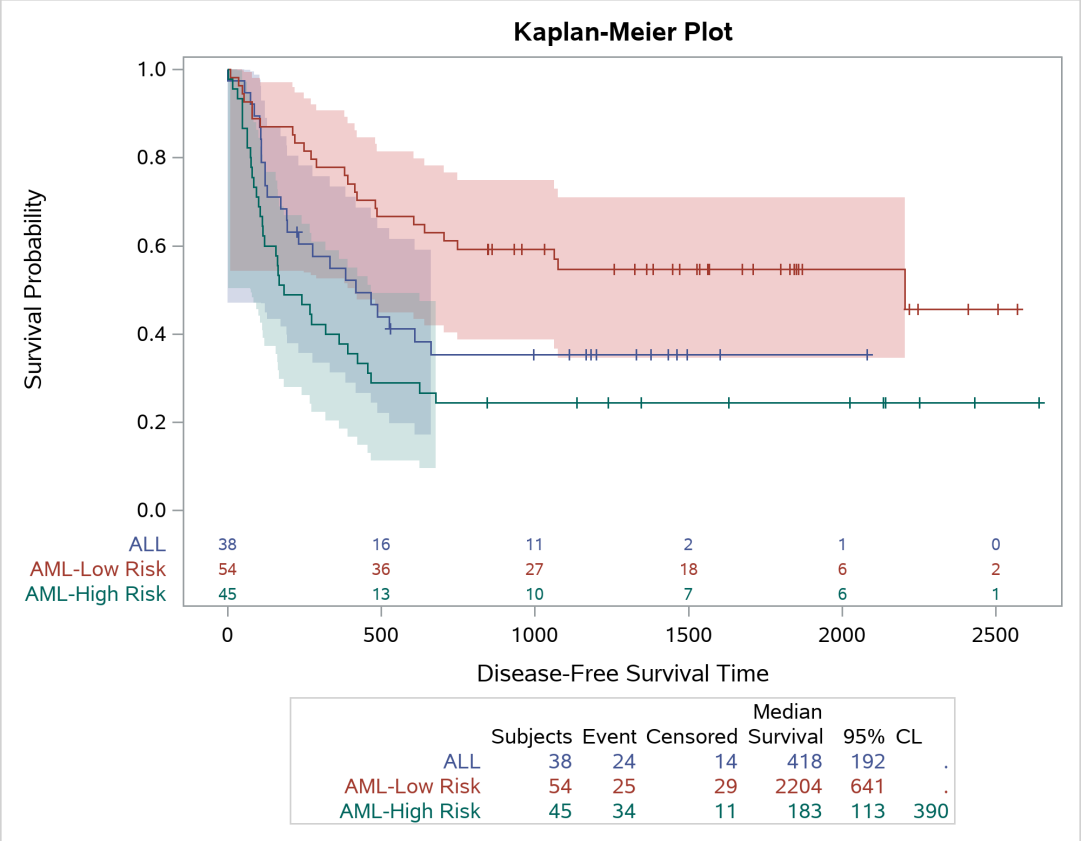
Begin by including the macros that you copied in the section "The Modularized Templates" on page 895.

This example combines a number of features from previous examples. The order of the strata levels in the tables is ALL, AML–Low Risk, and AML–High Risk (see the section "Reordering the Groups" on page 886). The title is set to 'Kaplan-Meier Plot' (see the section "Changing the Plot Title" on page 897). The second title line is suppressed (see the section "Changing the Second Title and Adding a Footnote" on page 912). The graph height is set to 500 pixels (see the section "Suppressing the Legend" on page 919). The legend and the inset box that contains the legend for censored values are both suppressed (see the sections "Suppressing the Legend" on page 919 and "Adding a Small Inset Table with Event Information" on page 915). The event table is displayed outside the plot (see the section "Adding an External Table with Event Information" on page 917) and the at-risk table is displayed inside the plot (see the section "Displaying the Patients-at-Risk Table inside the Plot" on page 880).

```
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;

%ProvideSurvivalMacros

%let TitleText2 = "Kaplan-Meier Plot";
%let nTitles    = 1;
%let GraphOpts  = DesignHeight=500px;
%let LegendOpts = ;
%let InsetOpts  = ;

%SurvivalSummaryTable

%CompileSurvivalTemplates

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

The results are displayed in Figure 23.38.

Figure 23.38 Kaplan-Meier Plot with Extensive Customizations



Displaying Percentages on the Y Axis

Begin by including the macros that you copied in the section "The Modularized Templates" on page 895.

There might be times when you want to use the customization macros *and* perform a subsequent template modification. If you want to make all the changes that appear in Figure 23.38 and display percentages on the Y axis, then you can modify the already-modified template. If you want to display only percentages, then you can omit the first round of template modifications. Begin by making the following template modifications, as in the section "Kaplan-Meier Plot with Event Table and Other Customizations" on page 920:

```
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;

%ProvideSurvivalMacros
```

```
%let TitleText2 = "Kaplan-Meier Plot";
%let nTitles    = 1;
%let GraphOpts  = DesignHeight=500px;
%let LegendOpts = ;
%let InsetOpts  = ;

%SurvivalSummaryTable

%CompileSurvivalTemplates
```

Next, write the modified template to a file:

```
proc template;
   source Stat.Lifetest.Graphics.ProductLimitSurvival / file='temp.tmp';
quit;
```

A subsequent step reads the file, inserts a PROC TEMPLATE statement at the beginning, and inserts a QUIT statement at the end. Then it modifies several statements in between. Before seeing how it makes those modifications, you should look at the file *temp.tmp* and find the code that needs to be modified. The following are samples of the statement fragments that need to be modified:

```
yaxisopts=(label="Survival Probability"
linearopts=(viewmin=0 viewmax=1
tickvaluelist=(0 .2 .4 .6 .8 1.0)));
bandplot LimitUpper=EP_UCL LimitLower=EP_LCL x=TIME
bandplot LimitUpper=HW_UCL LimitLower=HW_LCL x=TIME
scatterplot y=CENSORED x=TIME
stepplot y=SURVIVAL x=TIME
```

Each of these statement fragments occurs in more than one place. There are ten BANDPLOT statements and two each of the SCATTERPLOT and STEPPLOT statements. You need to change `'Survival Probability'` to `'Survival Percentage'`, VIEWMAX=1 to VIEWMAX=100, and the tick value list from proportions to percentages. Finally, you need to change all the Y-axis variables (which are specified in many statements) from proportions to percentages by specifying an expression. For example, you need to change **`LimitUpper=EP_UCL LimitLower=EP_LCL`** to **`LimitUpper=eval(100*EP_UCL) LimitLower=eval(100*EP_LCL)`**. The following step makes all these changes:

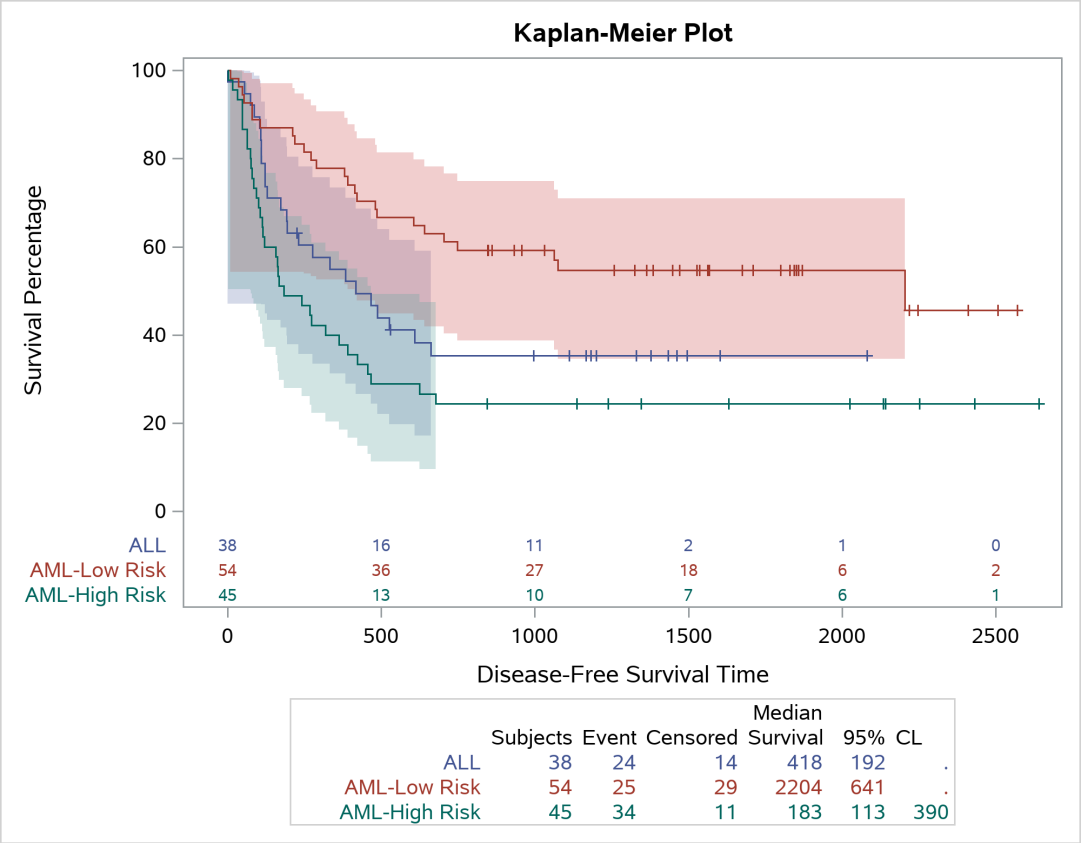
```
data _null_;
   length var stmt $ 32;
   infile 'temp.tmp' end=eof;
   input;
   if _n_ eq 1 then call execute('proc template;');
   stmt = scan(_infile_, 1);
   if stmt in ('scatterplot', 'stepplot', 'bandplot') then do;
      var = scan(scan(_infile_, 2, '='), 1);
      _infile_ = tranwrd(_infile_, trim(var), cats('eval(100*',var,')'));
      if stmt = 'bandplot' then do;
         var = scan(scan(_infile_, 3, '='), 1);
         _infile_ = tranwrd(_infile_, trim(var), cats('eval(100*',var,')'));
      end;
   end;
   _infile_ = tranwrd(_infile_, 'Survival Probability', 'Survival Percentage');
   _infile_ = tranwrd(_infile_, '0 .2 .4 .6 .8 1.0', '0 20 40 60 80 100');
   _infile_ = tranwrd(_infile_, 'viewmax=1', 'viewmax=100');
   call execute(_infile_);
   if eof then call execute('quit;');
run;
```

Statements in this DATA step begin by doing the following: read the template from the file, submit a PROC TEMPLATE statement to a buffer, find the name of each statement, and find the name of the first variable (which for this template is always a Y-axis variable). The DATA step scans for the second string by using an equal sign as a delimiter and then scans that result for the first string by using the standard delimiters (which include blanks). This step extracts only the variable name that follows the equal sign. The TRANWRD function changes the variable name to an expression. Band plots have two Y-axis specifications (for the upper and lower limits), so the next code block makes the same change to the second specification. The final TRANWRD functions change the Y-axis label, the tick values, and the maximum Y-axis value. The second-last CALL EXECUTE statement submits to a buffer all the template statements, including the modified statements. At the end, the DATA step submits a QUIT statement to the buffer. When the DATA step finishes, SAS submits all the code in the buffer, which modifies the template. The following step uses the modified template and creates a plot that has percentages on the Y axis:

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

The graph is displayed in Figure 23.39.

Figure 23.39 Survival Percentages



Compiled Template Cleanup

The following step restores all the default macros and macro variables and deletes the modified templates:

```
%ProvideSurvivalMacros

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

For more information about deleting compiled templates, see the section "SAS Item Stores" on page 962.

Graph Templates, Macros, and Macro Variables

The %ProvideSurvivalMacros 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 %CompileSurvivalTemplates macro provides the main body of the two templates. You can call it to compile 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.
  - The template **Stat.Lifetest.Graphics.ProductLimitSurvival2** provides the survival template when the at-risk table is outside the body of the plot.[7]

  The two templates share many statements, and a macro %DO loop creates both versions.
- The portion of the templates for the table for the *p*-values is stored in the macro %pValue.
- 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 two-cell lattice that contains the plot above the table when the at-risk table is outside the body of the plot.
- The macro %AtRiskLatticeEnd ends the two-cell lattice that contains the plot and the table when the at-risk table is outside the body of the plot.

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

- Some empty macros (%StmtsBeginGraph, %StmtsTop, and %StmtsBottom) are provided to enable you to add statements and options to strategic places in the templates.
- The %SurvTabHeader, %SurvivalTable, and %SurvivalSummaryTable macros enable you to easily add more GTL statements to the Kaplan-Meier plot templates to display event information for each stratum.

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 %ProvideSurvivalMacros macro follows:

```
%macro ProvideSurvivalMacros;

   %global atriskopts bandopts censored censorstr classopts
           graphopts groups insetopts legendopts ntitles stepopts tiplabel
           tips titletext0 titletext1 titletext2 xoptions yoptions;

   %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 CompileSurvivalTemplates; . . . %mend;
   %macro pValue;                   . . . %mend;
   %macro SingleStratum;            . . . %mend;
   %macro MultipleStrata;           . . . %mend;
   . . .
%CompileSurvivalTemplates
%mend;
```

The Macro Variables

The macros and macro variables are designed so that most of the time you need to modify only the macro variables and not the larger 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 nTitles    = 2;

%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 TipLabel   = tiplabel=(y="Survival Probability");
%let StepOpts   = ;

%let Groups     = group=STRATUM index=STRATUMNUM;

%let BandOpts   = displayTail=false &groups modelname="Survival";

%let InsetOpts  = autoalign=(TOPRIGHT BOTTOMLEFT TOP BOTTOM)
                  border=true BackgroundColor=GraphWalls:Color Opaque=true;
%let LegendOpts = title=GROUPNAME location=outside;

%let AtRiskOpts = display=(label) valueattrs=(size=7pt);
%let ClassOpts  = class=CLASSATRISK colorgroup=CLASSATRISK;

%let Censored   = markerattrs=(symbol=plus);
%let CensorStr  = "+ Censored";

%let GraphOpts  = ;
```

The %ProvideSurvivalMacros macro declares that these macro variables are global in scope, so you can assign values to them in your programs and have them affect the internal macros. These macro variables specify a variety of GTL options; for more information, see *SAS Graph Template Language: Reference*. The macro variables are as follows.

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

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

nTitles specifies the number of titles. Set the macro variable nTitles to 1 to suppress the second title line or 0 to suppress all title lines. You can add titles to the plot by adding ENTRYTITLE statements to the top of the %StmtsBeginGraph macro even when you suppress the usual titles by setting the nTitles macro variable to 0 or 1. By default, nTitles equals 2.

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= option label is too long. The LINEAROPTS= option specifies 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. You can add a LABEL= option or other axis options if you want. The SHORTLABEL= option provides the axis label for small plots when the label is too long. The short label comes from a dynamic variable that PROC LIFETEST provides. The OFFSETMIN= option ensures that there is extra space between the axis and the minimum tick mark. The LINEAROPTS= option specifies 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.

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

StepOpts provides options for the step functions. This macro variable is null by default. You can use this option to control the line thickness (for example, LINEATTRS=(THICKNESS=2.5)) and other aspects of the step functions.

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.

InsetOpts provides options for the inset table that provides the censored value legend and the homogeneity test *p*-value. The AUTOALIGN= option specifies the places in the plot 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 plot 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. You can set the InsetOpts macro variable to null to suppress the usual inset that contains the censored value and *p*-value.

LegendOpts provides options for the external legend that identifies the strata. The title comes from a dynamic variable GroupName that the procedure sets. By default, the legend is outside the plot. Specify LOCATION=INSIDE and an AUTOALIGN= option such as the one provided in the InsetOpts macro variable if you want the legend to appear inside the plot. You can set the LegendOpts macro variable to null to suppress the legend.

AtRiskOpts provides options for the at-risk table. The option DISPLAY=(LABEL) limits the display to labels. VALUEATTRS=(SIZE=7PT) specifies a font size of seven points.

ClassOpts provides the options that are used in the at-risk table to distinguish groups of observations.

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

CensorStr provides the character for the inset table that shows how censored observations appear in the plot.

GraphOpts provides options for the template BEGINGRAPH statement. By default, the GraphOpts macro variable is null. The following options are particularly useful:

- **ATTRPRIORITY=AUTO | NONE | COLOR** specifies the priority for varying the attributes that distinguish groups of observations. AUTO honors the setting that is otherwise in effect. COLOR varies only the color attribute. NONE simultaneously varies colors, markers, and lines. Styles such as HMTLBlue and Pearl are ATTRPRIORITY=COLOR styles, whereas styles such as DEFAULT, Statistical, Listing, and RTF are ATTRPRIORITY=NONE styles.
- **DATACOLORS=(***color-list***)** specifies the list of colors (which control confidence bands) to replace the graph data colors from the GraphData1–GraphData*N* style elements.
- **DATACONTRASTCOLORS=(***color-list***)** specifies the list of contrast colors (which control markers and lines) to replace the graph data contrast colors from the GraphData1–GraphData*N* style elements.
- **DATALINEPATTERNS=(***line-pattern-list***)** specifies the list of line patterns to replace the graph data line patterns from the GraphData1–GraphData*N* style elements. There are 46 line patterns, and you can specify each pattern by using an integer in the range 1 to 46. Some patterns have names associated with them. You can specify either the name or the number for the following number/name pairs: 1 Solid, 2 ShortDash, 4

MediumDash, 5 LongDash, 8 MediumDashShortDash, 14 DashDashDot, 15 DashDotDot, 20 Dash, 26 LongDashShortDash, 34 Dot, 35 ThinDot, 41 ShortDashDot, and 42 MediumDashDotDot.

- **DESIGNHEIGHT=***height* sets the graph height. You can set the graph height to the default graph width of 640 pixels by specifying the option DESIGNHEIGHT=DEFAULTDESIGNWIDTH. Or you can specify a size in pixels, such as DESIGNHEIGHT=500PX. Although the graph is designed at the specified height, you can resize it for the actual display by using the WIDTH= and HEIGHT= options in the ODS GRAPHICS statement. By default, DESIGNHEIGHT=480PX.
- **DESIGNWIDTH=***width* sets the graph width. You can set the graph width to the default graph height of 480 pixels by specifying the option DESIGNWIDTH=DEFAULTDESIGNHEIGHT. Or you can specify a size in pixels, such as DESIGNWIDTH=600PX. Although the graph is designed at the specified width, you can resize it for the actual display by using the WIDTH= and HEIGHT= options in the ODS GRAPHICS statement. By default, DESIGNWIDTH=640PX.

The Smaller Macros

The %ProvideSurvivalMacros macro provides four small macros that are easy for you to modify:

```
%macro StmtsBeginGraph; %mend;
%macro StmtsTop;        %mend;
%macro StmtsBottom;     %mend;

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

By default, the %StmtsBeginGraph, %StmtsTop, and %StmtsBottom macros are empty. You can use them to add new statements to the BEGINGRAPH block or to the beginning or end of the block of statements that define the appearance of the graph.

The %pValue macro is used to control the display of the *p*-value from the homogeneity test.

The Larger Macros

The examples and information up to this point have illustrated how you can make simple 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 must modify one of the other macros. The %CompileSurvivalTemplates macro, which is the macro that compiles all the pieces that you modified, is as follows:

```
%macro CompileSurvivalTemplates;
   %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
               plotTest GroupName Transparency SecondTitle TestName pValue
               _byline_ _bytitle_ _byfootnote_;
            BeginGraph %if %nrbquote(&graphopts) ne %then / &graphopts;;

            if (NSTRATA=1)
               %if &ntitles %then %do;
                  if (EXISTS(STRATUMID)) entrytitle &titletext1;
                  else                   entrytitle &titletext0;
                  endif;
               %end;

               %if &ntitles gt 1 %then %do;
                  %if not &outside %then if (PLOTATRISK=1);
                     entrytitle "With Number of Subjects at Risk" /
                                textattrs=GRAPHVALUETEXT;
                  %if not &outside %then %do; endif; %end;
               %end;

               %StmtsBeginGraph
               %AtRiskLatticeStart
               layout overlay / xaxisopts=(&xoptions) yaxisopts=(&yoptions);
                  %StmtsTop
                  %SingleStratum
                  %StmtsBottom
               endlayout;
               %AtRiskLatticeEnd

            else
               %if &ntitles %then %do; entrytitle &titletext2; %end;
               %if &ntitles gt 1 %then %do;
                  if (EXISTS(SECONDTITLE))
                     entrytitle SECONDTITLE / textattrs=GRAPHVALUETEXT;
                  endif;
               %end;

               %StmtsBeginGraph
               %AtRiskLatticeStart
               layout overlay / xaxisopts=(&xoptions) yaxisopts=(&yoptions);
                  %StmtsTop
                  %MultipleStrata
                  %StmtsBottom
               endlayout;
               %AtRiskLatticeEnd(class)

            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 the following two templates:

- **Stat.Lifetest.Graphics.ProductLimitSurvival** when the macro variable Outside is 0 and %SCAN(2,2-&OUTSIDE) is null
- **Stat.Lifetest.Graphics.ProductLimitSurvival2** when the macro variable 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 statement block is used to place the at-risk table outside the graph. When Outside is 1, the macros %AtRiskLatticeStart and %AtRiskLatticeEnd provide the LAYOUT LATTICE statement block (two cells, plot above and at-risk table below) and the LAYOUT OVERLAY statement block 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(useclassopts);
   %if &outside %then %do;
      endcell;
      cell;
         layout overlay / walldisplay=none xaxisopts=(display=none);
            axistable x=TATRISK value=ATRISK / &atriskopts
                     %if &useclassopts ne %then &classopts;;
         endlayout;
      endcell;
   endlayout;
   %end;
%mend;
```

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

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

   if (PLOTCENSORED=1)
      scatterplot y=CENSORED x=TIME / &censored &tiplabel
         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;
            axistable x=TATRISK value=ATRISK / &atriskopts;
         endinnermargin;
      endif;
   %end;
%mend;
```

The %MultipleStrata macro is as follows:

```
%macro MultipleStrata;
  if (PLOTHW=1)
     bandplot LimitUpper=HW_UCL LimitLower=HW_LCL x=TIME / &bandopts
              datatransparency=Transparency;
   endif;
   if (PLOTEP=1)
      bandplot LimitUpper=EP_UCL LimitLower=EP_LCL x=TIME / &bandopts
               datatransparency=Transparency;
   endif;
   if (PLOTCL=1)
      if (PLOTHW=1 OR PLOTEP=1)
         bandplot LimitUpper=SDF_UCL LimitLower=SDF_LCL x=TIME / &bandopts
                  display=(outline) outlineattrs=(pattern=ShortDash);
      else
         bandplot LimitUpper=SDF_UCL LimitLower=SDF_LCL x=TIME / &bandopts
                  datatransparency=Transparency;
      endif;
   endif;

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

   if (PLOTCENSORED=1)
      scatterplot y=CENSORED x=TIME / &groups &tiplabel &censored;
   endif;

   %if not &outside %then %do;
      if (PLOTATRISK=1)
         innermargin / align=bottom;
            axistable x=TATRISK value=ATRISK / &atriskopts &classopts;
         endinnermargin;
      endif;
   %end;

   %if %nrbquote(&legendopts) ne %then %do;
      DiscreteLegend "Survival" / &legendopts;
   %end;

   %if %nrbquote(&insetopts) ne %then %do;
      if (PLOTCENSORED=1)
         if (PLOTTEST=1)
            layout gridded / rows=2 &insetopts;
               entry &censorstr;
               %pValue
            endlayout;
         else
            layout gridded / rows=1 &insetopts;
               entry &censorstr;
            endlayout;
         endif;
      else
```

```
            if (PLOTTEST=1)
               layout gridded / rows=1 &insetopts;
                  %pValue
               endlayout;
            endif;
         endif;
      %end;

   %mend;
```

Event Table Macros

All the macros and macro variables that have been described up to this point are used in defining the two survival plot graph templates. Some macros (%StmtsTop and %StmtsBottom) and macro variables (StepOpts and GraphOpts) are null and do not affect the generated template code, but all are resolved somewhere in the process of producing the templates. In contrast, the macros %SurvTabHeader, %SurvivalTable, and %SurvivalSummaryTable are not used by default. They are available for you to use to add more statements to the templates.

The %SurvTabHeader macro provides the headings for the event table:

```
%macro SurvTabHeader(multiple);
   %if &multiple %then %do; entry ""; %end;
   entry "";
   entry &r "Median";
   entry "";

   %if &multiple %then %do; entry ""; %end;
   entry &r "Subjects";
   entry &r "Event";
   entry &r "Censored";
   entry &r "Survival";
   entry &r PctMedianConfid;
   entry halign=left  "CL";
%mend;
```

This table is not displayed by default. There are two types of headings: one for multiple strata and one for a single stratum.

The %SurvivalTable macro provides the body of the event table:

```
%macro SurvivalTable;
   %local fmt r i t;
   %let fmt = bestd6.;
   %let r = halign = right;
   columnheaders;
      layout overlay / pad=(top=5);
         if(NSTRATA=1)
            layout gridded / columns=6 border=TRUE;
               dynamic PctMedianConfid NObs NEvent Median
                       LowerMedian UpperMedian;
               %SurvTabHeader(0)
```

```
            entry &r NObs;
            entry &r NEvent;
            entry &r eval(NObs-NEvent);
            entry &r eval(put(Median,&fmt));
            entry &r eval(put(LowerMedian,&fmt));
            entry &r eval(put(UpperMedian,&fmt));
         endlayout;
      else
         layout gridded / columns=7 border=TRUE;
            dynamic PctMedianConfid;
            %SurvTabHeader(1)
            %do i = 1 %to 10;
               %let t = / textattrs=GraphData&i;
               dynamic StrVal&i NObs&i NEvent&i Median&i
                       LowerMedian&i UpperMedian&i;
               if (&i <= nstrata)
                  entry &r StrVal&i &t;
                  entry &r NObs&i &t;
                  entry &r NEvent&i &t;
                  entry &r eval(NObs&i-NEvent&i) &t;
                  entry &r eval(put(Median&i,&fmt)) &t;
                  entry &r eval(put(LowerMedian&i,&fmt)) &t;
                  entry &r eval(put(UpperMedian&i,&fmt)) &t;
               endif;
            %end;
         endlayout;
      endif;
   endlayout;
endcolumnheaders;
%mend;
```

The %SurvivalSummaryTable macro redefines the %AtRiskLatticeStart and %AtRiskLatticeEnd macros so that they provide the body of the event table:

```
%macro SurvivalSummaryTable;
   %macro AtRiskLatticeStart;
      layout lattice / columndatarange=union rowgutter=10
         rows=%if &outside %then 2 rowweights=ROWWEIGHTS;
              %else              1;;
      %if &outside %then %do; cell; %end;
   %mend;

   %macro AtRiskLatticeEnd(useclassopts);
      %if &outside %then %do;
         endcell;
         cell;
            layout overlay / walldisplay=none xaxisopts=(display=none);
               axistable x=TATRISK value=ATRISK / &atriskopts
                         %if &useclassopts ne %then &classopts;;
            endlayout;
         endcell;
      %end;
      %SurvivalTable
      endlayout;
   %mend;
%mend;
```

If you want to create an event table like the one displayed in Figure 23.36, you only need to call the %SurvivalSummaryTable macro. If you want to modify the table, then you need to modify the %SurvTabHeader and %SurvivalTable macros.

Dynamic Variables

Graph templates consist of instructions, written by SAS developers, in conjunction with SAS procedure code. However, SAS developers cannot fully provide some instructions when the template is written, because some elements of some graphs cannot be known until the procedure runs. For example, the legend title in a graph that has multiple strata corresponds to the label or name of the stratification variable, and the procedure calculates the *p*-value for the homogeneity test. SAS procedures create dynamic variables to provide some run-time information to graphs.[8] Some dynamic variables are set by the procedure and are declared in the template. Other dynamic variables are also set by the procedure, but you must declare them directly or through the template modification macros before you can use them.

Dynamic Variables That Are Automatically Declared

The primary dynamic variables are as follows:

ByFootNote	is a binary variable that, when true, displays the BY-group BY line as a footnote.
ByLine	is a character variable that provides the BY-group BY line.
ByTitle	is a binary variable that, when true, displays the BY-group BY line as a title.
ClassAtRisk	is a character variable that names the data object column that contains the classification (stratification) values.
GroupName	is a character variable that contains the stratification legend title.
LabelCL	is a character variable that contains the label for the confidence limits legend entry (including the percent sign).
LabelEP	is a character variable that contains the label for the equal-precision band legend entry (including the percent sign).
LabelHW	is a character variable that contains the label for the Hall-Wellner band legend entry (including the percent sign).
MaxTime	is a numeric variable that contains the maximum value to display on the X axis.
Method	is a character variable that contains the method for the plot title.
NStrata	is an integer variable that contains the number of strata.
PValue	is a numeric variable that contains the *p*-value for the homogeneity test.
PlotAtRisk	is a binary variable that, when true, is used to display the at-risk table.
PlotCensored	is a binary variable that, when true, displays the censored values on the step functions.

[8] Axis labels can be set directly in the template or at run time through dynamic variables or through data object column labels.

PlotCL	is a binary variable that, when true, displays the pointwise confidence limits.
PlotEP	is a binary variable that, when true, displays the equal-precision band.
PlotHW	is a binary variable that, when true, displays the Hall-Wellner confidence band.
PlotTest	is a binary variable that, when true, displays the *p*-value for the homogeneity test.
RowWeights	is a pair of relative heights of the plot and the external at-risk table.
SecondTitle	is a character variable that provides the second title line.
StratumID	is a character variable that provides the value of the stratification variable for the single stratum case.
TestName	is a character variable that provides the name of the homogeneity test (for example, 'logrank').
Transparency	is a numeric variable that provides the transparency for the confidence bands in the multiple strata case.
XName	is a character variable that contains a short label for the X axis, which might be used in place of the ordinary X-axis label when the ordinary label is long or the plot is small.
XtickValFitPol	is a character variable that contains the option for handling dense tick values on the X axis.
XtickVals	is a list of X-axis tick values.

Additional Dynamic Variables

PROC LIFETEST passes to the survival plots a number of dynamic variables that contain summary statistics. Some of these dynamic variables are used when you call the %SurvivalSummaryTable macro. Table 23.1 and Table 23.2 list these additional dynamic variables for the Kaplan-Meier curves and the life-table curves, respectively. These dynamic variables are not declared in the templates for the survival curves, but you can declare them and use them to enhance the default plots.[9] The names of the dynamic variables depend on the STRATA= suboption of the PLOTS=SURVIVAL option: STRATA=INDIVIDUAL produces a separate plot for each stratum, and STRATA=OVERALL produces one plot that has overlaid curves.

Table 23.1 Additional Dynamic Variables for `Stat.Graphics.ProductLimitSurvival`

STRATA=	Dynamic	Description
OVERLAY	StrVal*j*	Label for the *j*th stratum
	NObs*j*	Number of observations in the *j*th stratum
	NEvent*j*	Number of events in the *j*th stratum
	Median*j*	Median survival time of the *j*th stratum
	LowerMedian*j*	Lower median survival time of the *j*th stratum
	UpperMedian*j*	Upper median survival time of the *j*th stratum
	PctMedianConfid	Confidence of the median intervals, in percentage

[9] Because the number of dynamic variables is a function of the number of strata, the template definition cannot automatically contain the correct number of dynamic variables.

Table 23.1 *continued*

STRATA=	Dynamic	Description
INDIVIDUAL	NObs	Number of observations
	NEvent	Number of events
	Median	Median survival time
	LowerMedian	Lower median survival time
	UpperMedian	Upper median survival time
	PctMedianConfid	Confidence of the median intervals, in percentage

Table 23.2 Additional Dynamic Variables for `Stat.Graphics.LifetableSurvival`

STRATA=	Dynamic	Description
OVERLAY	StrVal*j*	Label for the *j*th stratum
	NObs*j*	Number of observations in the *j*th stratum
	NEvent*j*	Number of events in the *j*th stratum
INDIVIDUAL	NObs	Number of observations
	NEvent	Number of events

Highly Customized Graphs: Unicode in the STRATA Statement Variable

This example shows you how to display Unicode characters in the survival plot legend. In this case, the legend displays the values of the STRATA statement variable. This example does not use the template customization macros. Instead, it takes a different approach to customization and shows you how to modify the survival plot data object and create the graph after PROC LIFETEST finishes. By using this technique, you can display in the survival plot Unicode characters that otherwise could not be displayed.

Unicode Values Require Alternative Processing

The following steps attempt to replace the ordinary hyphen with a longer dash (that is, they attempt to replace '-' with '—') and run PROC LIFETEST:

```
proc format;
   value $dfmt "AML-Low Risk"  = "AML(*ESC*){Unicode '2014'x}Low Risk"
               "AML-High Risk" = "AML(*ESC*){Unicode '2014'x}High Risk";
run;

ods graphics on;
proc lifetest data=sashelp.BMT plots=survival(cb=hw atrisk(outside maxlen=13));
   time T * Status(0);
   strata Group;
   format group $dfmt.;
run;
```

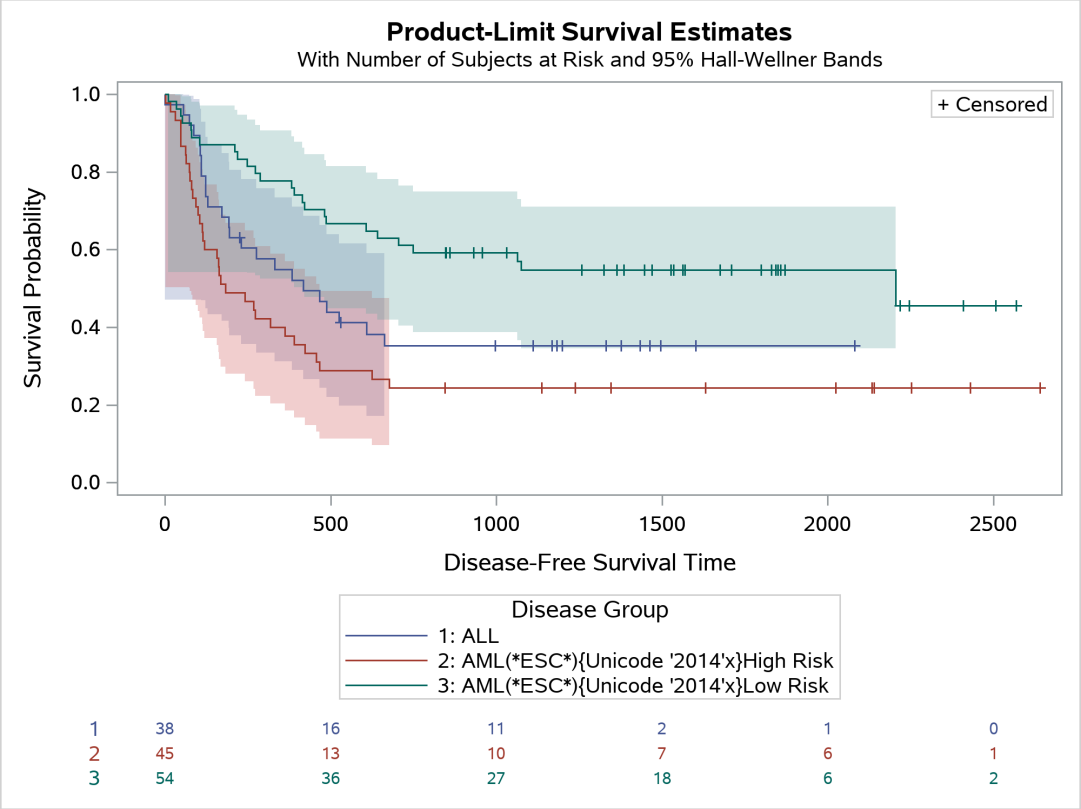
The table and graph that are displayed in Output 23.40 show a difference between how ODS handles Unicode characters in tables and how ODS Graphics handles Unicode characters in graphs. When ODS creates tables and processes an escape sequence and a Unicode character in the value of a column, it substitutes the special character in the output. (This discussion does not apply to the LISTING destination.) When ODS Graphics creates graphs and processes an escape sequence and a Unicode character in the *formatted value* of a column, it substitutes the special character in the output. In other words, in ODS Graphics, you can use Unicode only with formats. This might seem puzzling, because this example does in fact use a format, but it does not display the formatted value. It does not work as expected, because PROC LIFETEST does not send unformatted values and the format to ODS Graphics.

Figure 23.40 Unicode Displays Automatically in Tables but Not Graphs

The LIFETEST Procedure

Summary of the Number of Censored and Uncensored Values

Stratum	Group	Total	Failed	Censored	Percent Censored
1	ALL	38	24	14	36.84
2	AML—High Risk	45	34	11	24.44
3	AML—Low Risk	54	25	29	53.70
Total		137	83	54	39.42



PROC LIFETEST and most other analytical procedures apply the formats at the time that the data are read and mostly process strings that contain the formatted values, not the raw values. If you want to see the Unicode characters in the graph, you must circumvent this behavior and send the raw values and the format to ODS Graphics. There are two ways to do this. The first uses the GROUP= option, and the second modifies the data object.

Using the GROUP= Option

You can display the GROUP= variable in the graph. The following step illustrates:

```
proc lifetest data=sashelp.BMT plots=survival(cb=hw atrisk(outside maxlen=13));
   time T * Status(0);
   strata / group=Group;
   format group $dfmt.;
run;
```

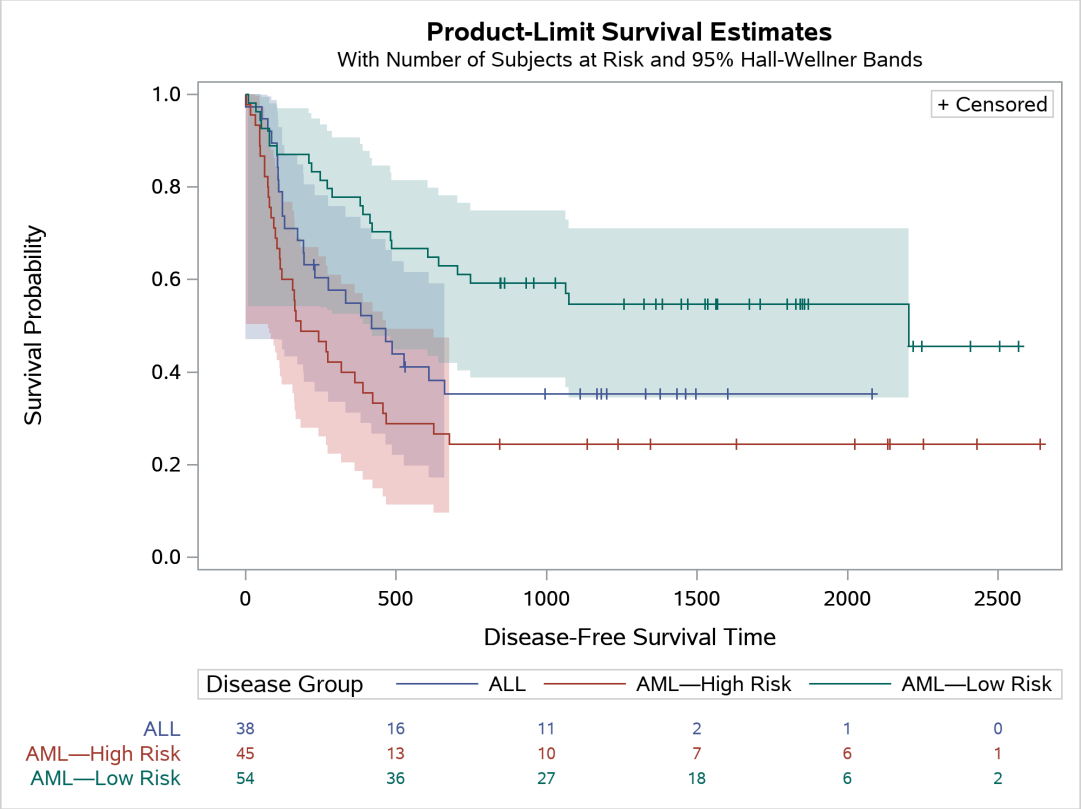
Some of the results are displayed in Output 23.41.

Figure 23.41 Select Output from Both PROC LIFETEST Runs

The LIFETEST Procedure

Summary of the Number of Censored and Uncensored Values

Stratum	Group	Total	Failed	Censored	Percent Censored
1	ALL	38	24	14	36.84
1	AML—High Risk	45	34	11	24.44
1	AML—Low Risk	54	25	29	53.70
Total		137	83	54	39.42



Modifying the Data Object

The second way to approach this problem requires you to modify the data object and use the ODS document. The first things you must do are to open an ODS document, save the results of the PROC LIFETEST analysis, save the data object that underlies the graph into a SAS data set, and close the ODS document. The following code illustrates:

```
ods document name=MyDoc (write);
proc lifetest data=sashelp.BMT plots=survival(cb=hw atrisk(outside maxlen=13));
   ods output survivalplot=sp;
   time T * Status(0);
   strata Group;
run;
ods document close;
```

The ODS document contains all the tables, graphs, titles, notes, and other components of the output. The following step lists the contents of the ODS document:

```
proc document name=MyDoc;
   list / levels=all;
quit;
```

The contents are displayed in Output 23.42.

Figure 23.42 Contents of the ODS Document

Listing of: \Work.Mydoc
Order by: Insertion
Number of levels: All

Obs	Path	Type
1	\Lifetest#1	Dir
2	\Lifetest#1\Stratum1#1	Dir
3	\Lifetest#1\Stratum1#1\ProductLimitEstimates#1	Table
4	\Lifetest#1\Stratum1#1\TimeSummary#1	Dir
5	\Lifetest#1\Stratum1#1\TimeSummary#1\SummaryNote#1	Note
6	\Lifetest#1\Stratum1#1\TimeSummary#1\Quartiles#1	Table
7	\Lifetest#1\Stratum1#1\TimeSummary#1\Means#1	Table
8	\Lifetest#1\Stratum2#1	Dir
9	\Lifetest#1\Stratum2#1\ProductLimitEstimates#1	Table
10	\Lifetest#1\Stratum2#1\TimeSummary#1	Dir
11	\Lifetest#1\Stratum2#1\TimeSummary#1\SummaryNote#1	Note
12	\Lifetest#1\Stratum2#1\TimeSummary#1\Quartiles#1	Table
13	\Lifetest#1\Stratum2#1\TimeSummary#1\Means#1	Table
14	\Lifetest#1\Stratum3#1	Dir
15	\Lifetest#1\Stratum3#1\ProductLimitEstimates#1	Table
16	\Lifetest#1\Stratum3#1\TimeSummary#1	Dir
17	\Lifetest#1\Stratum3#1\TimeSummary#1\SummaryNote#1	Note
18	\Lifetest#1\Stratum3#1\TimeSummary#1\Quartiles#1	Table
19	\Lifetest#1\Stratum3#1\TimeSummary#1\Means#1	Table
20	\Lifetest#1\CensoredSummary#1	Table
21	\Lifetest#1\StrataHomogeneity#1	Dir
22	\Lifetest#1\StrataHomogeneity#1\HomogeneityNote#1	Note
23	\Lifetest#1\StrataHomogeneity#1\HomStats#1	Table
24	\Lifetest#1\StrataHomogeneity#1\LogrankHomCov#1	Table
25	\Lifetest#1\StrataHomogeneity#1\WilcoxonHomCov#1	Table
26	\Lifetest#1\StrataHomogeneity#1\HomTests#1	Table
27	\Lifetest#1\SurvivalPlot#1	Graph

In the following step, you copy and paste the ODS document path for the survival plot into the OBDYNAM statement to create an ODS output data set that contains all the dynamic variables for the survival plot:

```
proc document name=MyDoc;
   ods output dynamics=dynamics;
   obdynam \Lifetest#1\SurvivalPlot#1;
quit;
```

The dynamic variables and their values are displayed in Output 23.43.[10]

Figure 23.43 Dynamic Variables

Dynamics for: \Work.Mydoc\Lifetest#1\SurvivalPlot#1

Name	Value	Type	Namespace
NSTRATA	3	Data	
PLOTHW	1	Data	
PLOTEP	0	Data	
PLOTCL	0	Data	
PLOTCENSORED	1	Data	
Transparency	0.7	Data	
PLOTTEST	0	Data	
BYTITLE		Data	
BYLINE		Data	
BYFOOTNOTE		Data	
METHOD	Product-Limit	Data	
ROWWEIGHTS	PREFERRED	Data	
XNAME	T	Data	
LABELHW	95% Hall-Wellner Band	Data	
SECONDTITLE	With Number of Subjects at Risk and 95% Hall-Wellner Bands	Data	
GROUPNAME	Disease Group	Data	
___NOBS___	156	Data	
___NOBS___	156	Column	HW_UCL
___NOBS___	156	Column	HW_LCL
___NOBS___	156	Column	Time
___NOBS___	156	Column	Survival
___NOBS___	156	Column	AtRisk
___NOBS___	156	Column	Event
___NOBS___	156	Column	Censored
___NOBS___	156	Column	tAtRisk
___NOBS___	156	Column	Stratum
CLASSATRISK	Stratum	Column	Stratum
___NOBS___	156	Column	StratumNum

You can use a DATA step as follows to generate a PROC SGRENDER statement; specify the ODS output data set, the template, and the format; and specify the names and values of all the dynamic variables:

[10] Some dynamic variables have numeric values, and some have character values. The ODS output data set (not shown) stores the character and formatted numeric values in the variable cValue1 and the unformatted numeric values in nValue1.

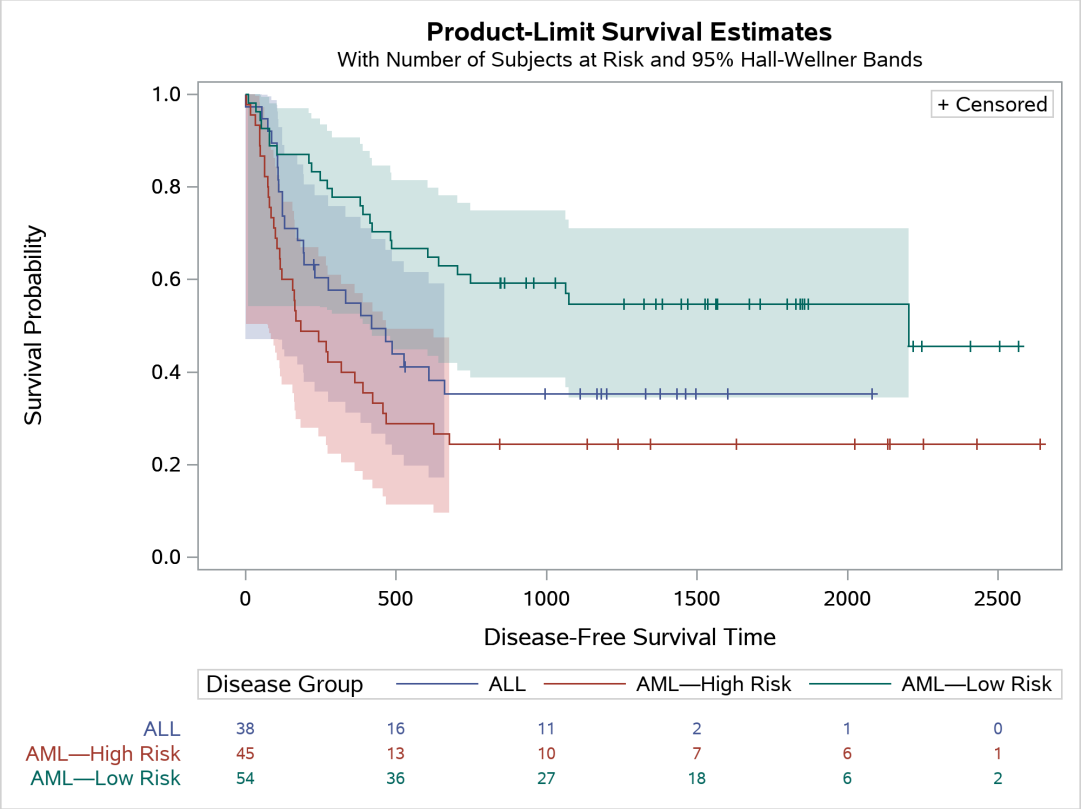
```
data _null_;
   set dynamics(where=(label1 ne '___NOBS___')) end=eof;
   if _n_ = 1 then
      call execute('proc sgrender data=sp
                    template=Stat.Lifetest.Graphics.ProductLimitSurvival2;
                    format stratum $dfmt.;
                    dynamic');
   if cvalue1 ne ' ' then
      call execute(catx(' ', label1, '=',
                   ifc(n(nvalue1), cvalue1, quote(trim(cvalue1)))));
   if eof then call execute('; run;');
run;
```

This DATA step generates and executes a PROC SGRENDER step that creates the graph. The DATA step uses a series of CALL EXECUTE statements to populate the list of dynamic variables. The DATA step generates and submits to SAS the following code (which has been reformatted for display here):

```
proc sgrender data=sp
   template=Stat.Lifetest.Graphics.ProductLimitSurvival2;
   format stratum $dfmt.;
   dynamic
      NSTRATA = 3
      PLOTHW = 1
      PLOTEP = 0
      PLOTCL = 0
      PLOTCENSORED = 1
      Transparency = 0.7
      PLOTTEST = 0
      METHOD = "Product-Limit"
      ROWWEIGHTS = "PREFERRED"
      XNAME = "T"
      LABELHW = "95% Hall-Wellner Band"
      SECONDTITLE = "With Number of Subjects at Risk and 95% Hall-Wellner Bands"
      GROUPNAME = "Disease Group"
      CLASSATRISK = "Stratum";
run;
```

The survival plot, which now contains the long dashes, is displayed in Output 23.44.

Figure 23.44 Unicode in the Legend



This example shows that you can save the data object and dynamic variables into SAS data sets. Then you can apply a format to the Stratum variable as you create the graph from those data sets. The advantage of this approach is that it provides full flexibility. For example, you can use it with multiple strata variables.

Multiple Strata Variables and Reformatting the Legend

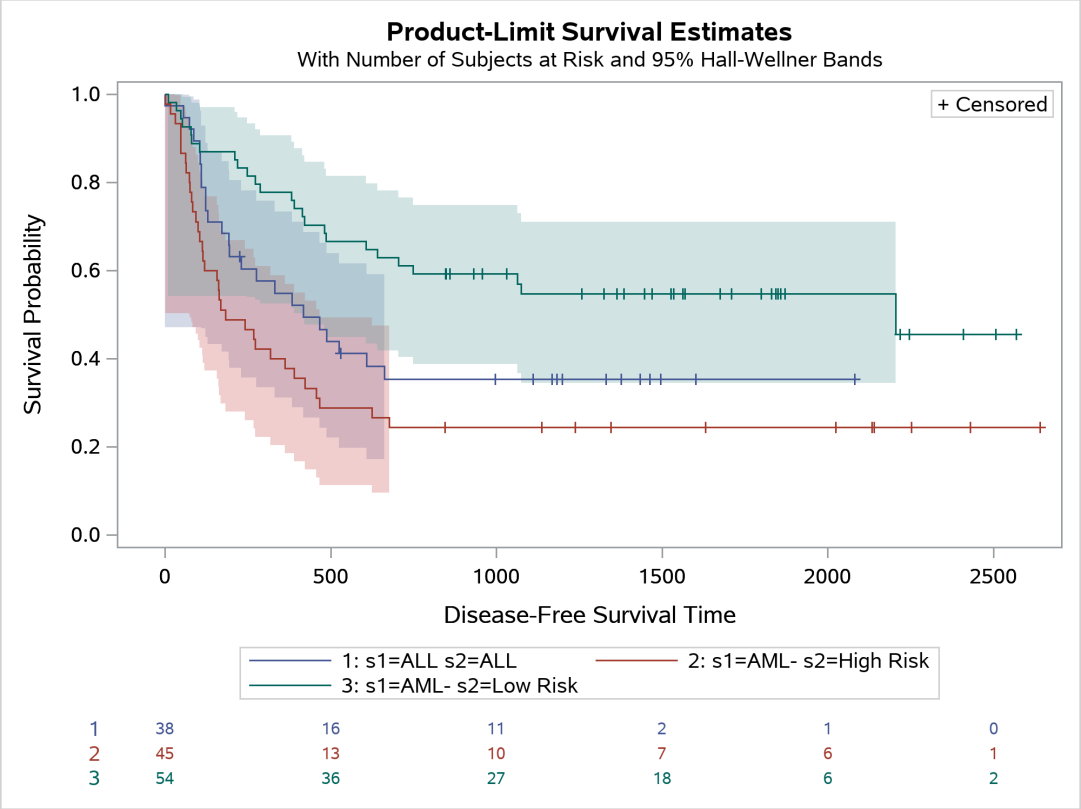
This example reformats the Group variable from the Sashelp.BMT data set as follows to produce the same analysis from two strata variables:

```
data bmt;
   set sashelp.bmt;
   s1 = scan(group, 1, '-');
   s2 = scan(group, 2, '-');
   if s2 ne ' ' then s1 = cats(s1,'-');
   else             s2 = 'ALL';
run;

proc lifetest data=BMT plots=survival(cb=hw atrisk(outside maxlen=13));
   ods output survivalplot=sp;
   time T * Status(0);
   strata s1 s2;
run;
```

In practice, you would never create two strata variables from one. This is done here simply to illustrate the techniques for dealing with multiple strata variables in the context of a familiar example.

The graph in Output 23.45 contains variable names and equal signs. You can reformat the values in any way that you choose. For example, if there were Unicode specifications in each variable, you could handle it this way.

Figure 23.45 Unicode in the Legend



The next step creates a new format that contains two Unicode specifications:

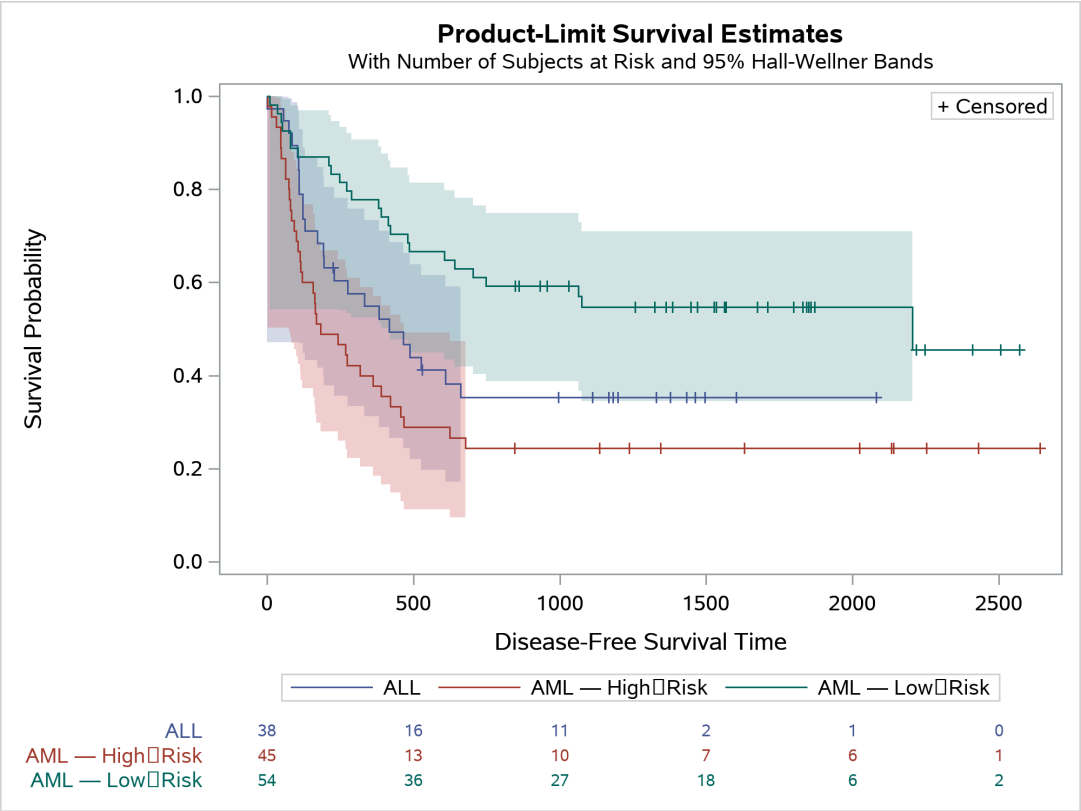
```
proc format;
   value $d2fmt '1: s1=ALL s2=ALL'= 'ALL'
                "2: s1=AML- s2=High Risk" =
                "AML (*ESC*){Unicode '2014'x} High(*ESC*){Unicode '2012'x}Risk"
                "3: s1=AML- s2=Low Risk"  =
                "AML (*ESC*){Unicode '2014'x} Low(*ESC*){Unicode '2012'x}Risk";

run;
```

The rest of this example proceeds as the previous example did. You begin by capturing the output data object and dynamic variables. Then you create the graph by using the new format. The only difference is that you need to change the value of the dynamic variable ClassAtRisk so that the formatted value is displayed to the left of the subjects-at-risk table in place of the stratum number.

```
ods document name=MyDoc (write);
proc lifetest data=BMT plots=survival(cb=hw atrisk(outside maxlen=16));
   ods output survivalplot=sp;
   time T * Status(0);
   strata s1 s2;
run;
ods document close;

proc document name=MyDoc;
   ods output dynamics=dynamics;
   obdynam \Lifetest#1\SurvivalPlot#1;
quit;

data _null_;
   set dynamics(where=(label1 ne '___NOBS___')) end=eof;
   if _n_ = 1 then
      call execute('proc sgrender data=sp
                    template=Stat.Lifetest.Graphics.ProductLimitSurvival2;
                    format stratum $d2fmt.;
                    dynamic');
   if label1 eq 'CLASSATRISK' then cvalue1 = 'Stratum';
   if cvalue1 ne ' ' then
      call execute(catx(' ', label1, '=',
                   ifc(n(nvalue1), cvalue1, quote(trim(cvalue1)))));
   if eof then call execute('; run;');
run;
```

The graph is displayed in Output 23.46.

Figure 23.46 Reformatted Legend



You can extend this example by specifying additional procedure options or by modifying the template. You can even add SG annotation.

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). Although it is rarely necessary, you can use different styles or modify styles to change the appearance of all graphs, including the survival plot. In the past, you could make certain Kaplan-Meier plot modifications only through style modifications. However, with the addition of the DATACOLORS=, DATACONTRASTCOLORS=, and DATALINEPATTERNS= options in the GTL, you no longer have to modify styles in order to modify how groups of observations are displayed. This section shows you how to change styles, extract group color and other information from styles, and modify styles.

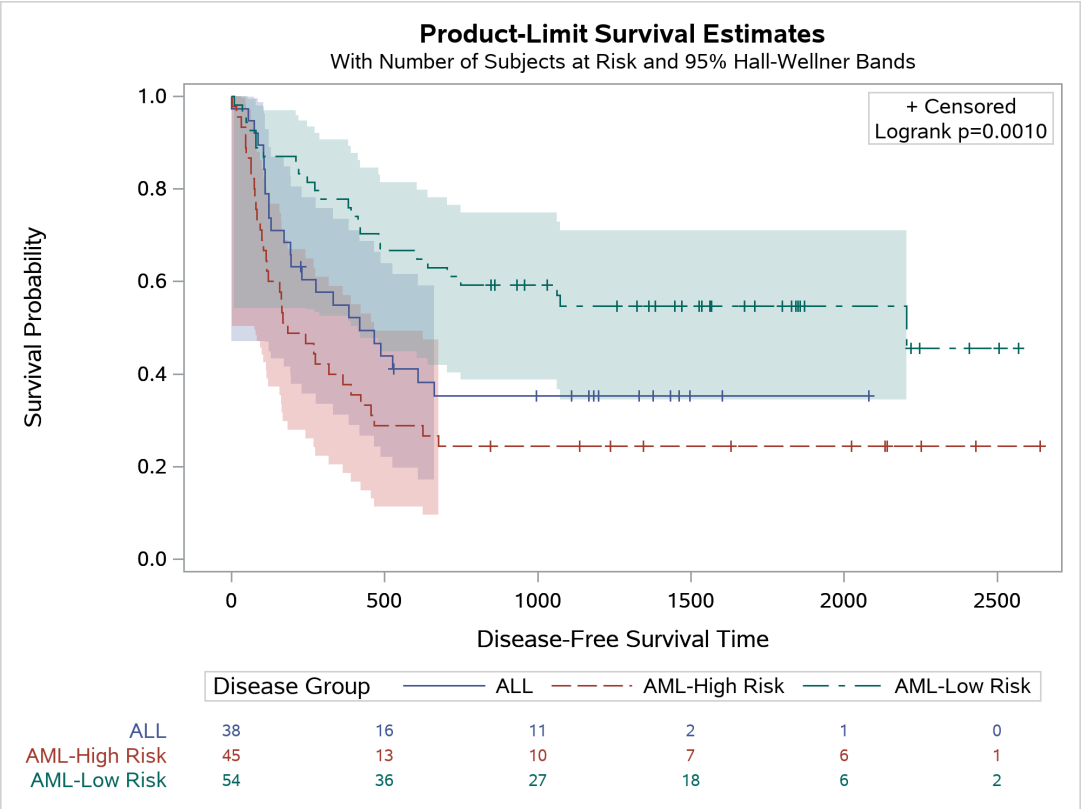
Changing the Style

The graphs that have been displayed up to this point were all produced by using the HTMLBlue style, which is the default style for the HTML destination in the SAS windowing environment. This is an all-color style (because of the ATTRPRIORITY='Color' option); 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. You can use the HTMLBlueCML style as follows to make a graph whose line patterns differ:

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

The results are displayed in Figure 23.47. This example also illustrates specifying the IMAGE_DPI= option to control the resolution (measured in dots per inch, or DPI) of the image. All images in this chapter are created at 300 DPI. The default setting for the HTML destination is 100 DPI. Images that are created at 300 DPI are clearer than images created at 100 DPI, but they require about nine times as much disk space.

Figure 23.47 Line and Color Group Differentiation



Color Priority Styles

You can use the HTMLBlue or Pearl style when you want to distinguish groups only by color. Alternatively, you can easily modify any other style to be an all-color style like HTMLBlue or Pearl by using the ATTRPRIORITY='Color' option:[11]

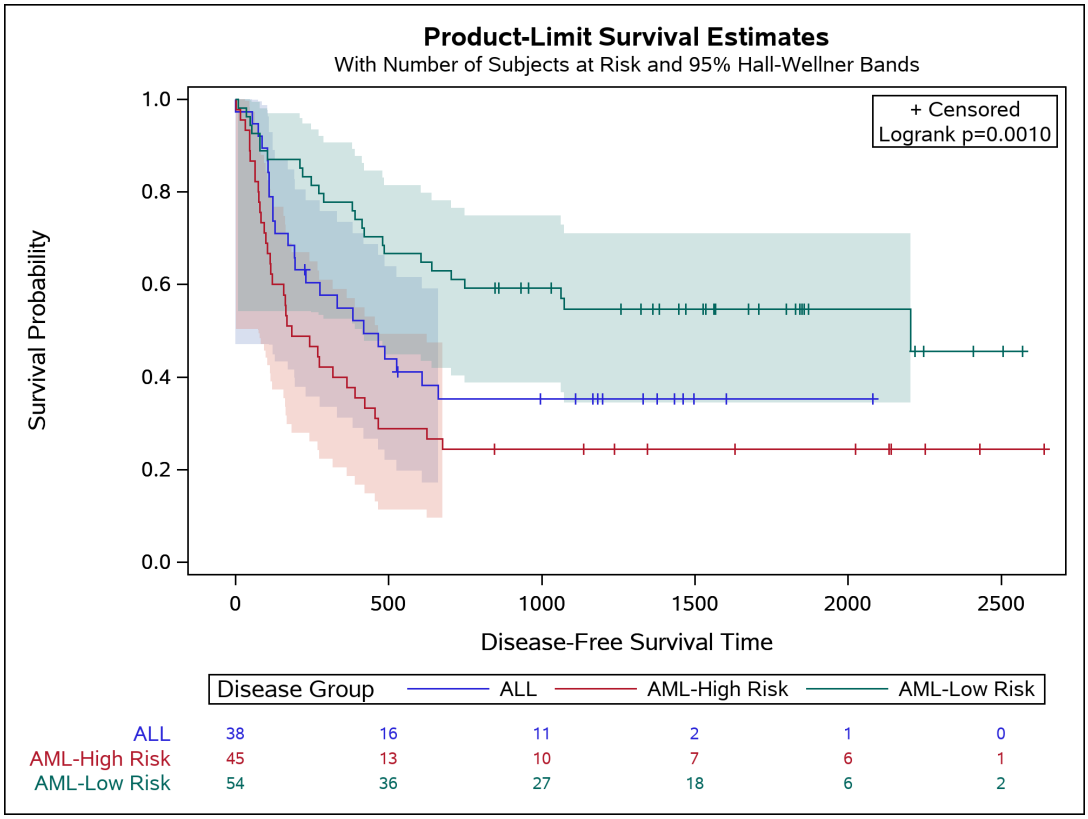
```
proc template;
   define style styles.ListingColor;
      parent = styles.Listing;
      style Graph from Graph / attrpriority = "Color";
   end;
run;
```

You need to specify the new style name in an ODS destination statement, as in the following:

```
ods html style=ListingColor image_dpi=300;
proc lifetest data=sashelp.BMT
              plots=survival(cb=hw test atrisk(outside maxlen=13));
   time T * Status(0);
   strata Group;
run;
ods html close;
```

The results are displayed in Figure 23.48.

Figure 23.48 ATTRPRIORITY='Color' Style



[11] The style option is ATTRPRIORITY=*quoted-string*, whereas the GTL option is ATTRPRIORITY=*keyword*.

Displaying a Style and Extracting Color Lists

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

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

The results of this step, which are not shown, 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 included. 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='style.tmp';
run;

data colors;
   length element Color $ 20;
   infile 'style.tmp';
   input;
   if index(_infile_, 'data') then do;
      element = scan(_infile_, 1, ' ');
      Color = scan(_infile_, 3, ' ;');
      Type = ifc(index(element, 'gc'), 'Line', 'Fill') || ' Colors';
      i = input(compress(element, 'gcdat';'), ?? 2.);
      if i then output;
   end;
run;

proc sort; by descending type i; run;

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

The results are displayed in Figure 23.49. All colors are specified in values of the form CX*rrggbb*, where the last six characters specify RGB (red, green, blue) values on the hexadecimal scale of 00 to FF (0 to 255, base 10).

Figure 23.49 HMTLBlue Style Colors List

Type=Line Colors

Type	Color
Line Colors	cx445694
	cxA23A2E
	cx01665E
	cx543005
	cx9D3CDB
	cx7F8E1F
	cx2597FA
	cxB26084
	cxD17800
	cx47A82A
	cxB38EF3
	cxF9DA04

Type	Color
Fill Colors	cx6F7EB3
	cxD05B5B
	cx66A5A0
	cxA9865B
	cxB689CD
	cxBABC5C
	cx94BDE1
	cxCD7BA1
	cxCF974B
	cx87C873
	cxB7AEF1
	cxDDD17E

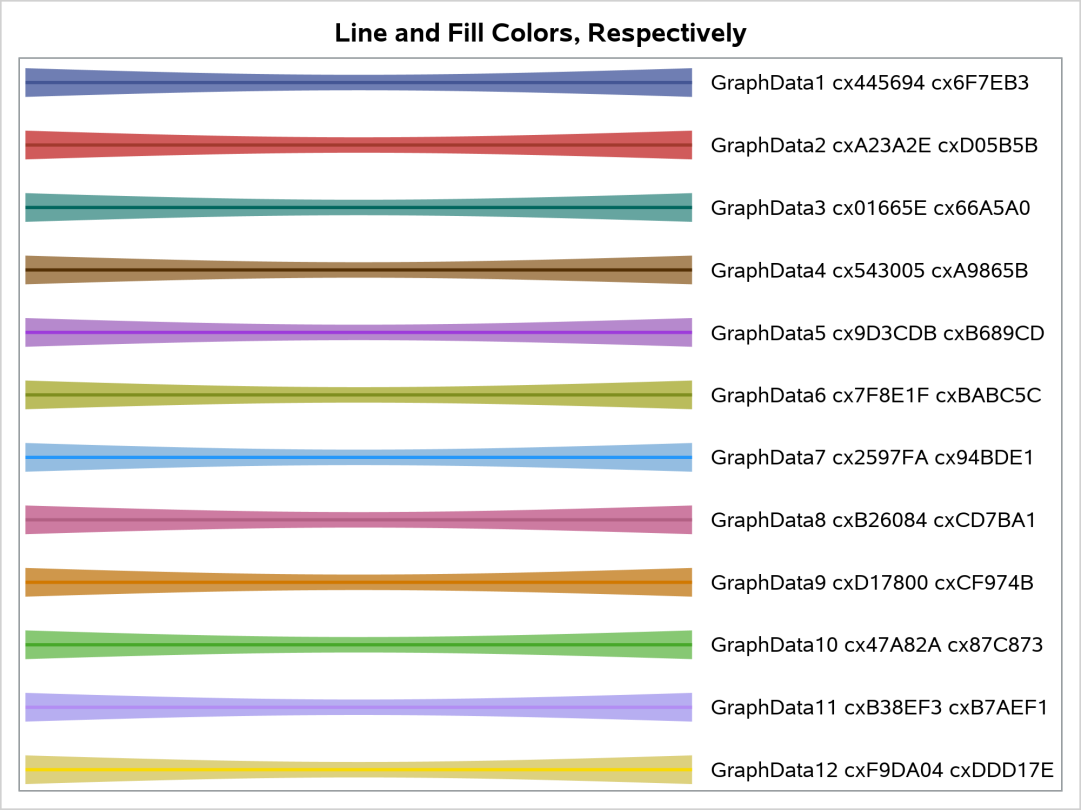
You can use the following steps to display the **GraphData1** – **GraphData12** line and fill colors (contrast colors and colors, respectively):

```
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;
run;

title;
```

The results are displayed in Figure 23.50. The colors in Figure 23.50 are richer than the colors in the bands in the survival plots because of the DATATRANSPARENCY= options in the BANDPLOT statements.

Figure 23.50 HTMLBlue Style Colors Display



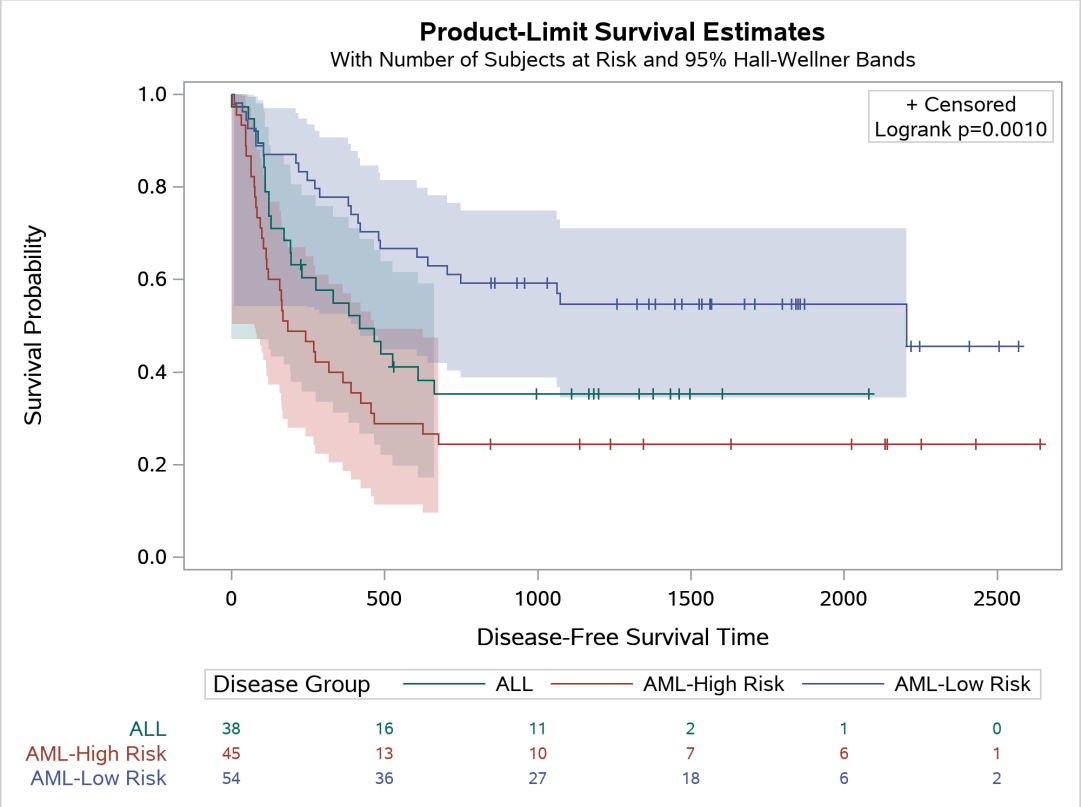
Modifying Color Lists

You can use the information in Figure 23.50 to specify the desired colors in the graph template. You can copy the third, second, and first colors from each list and switch the colors as follows:

```
%ProvideSurvivalMacros

%let GraphOpts = DataContrastColors=(cx01665E cxA23A2E cx445694)
                 DataColors=(cx66A5A0 cxD05B5B cx6F7EB3);

%CompileSurvivalTemplates

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

The results are displayed in Figure 23.51. The familiar colors are used, but they are now in a different order. The next section shows you how to modify a style template to change the color order without having to extract the original color names.

Figure 23.51 Swapping Colors from the Style



You can use the information in Figure 23.50 to modify the style template, but the next example shows an easier way.

Swapping Colors among Style Elements

You can modify the colors in a style as follows:

```
%macro reorder(from, to, list);

   proc template;
      define style styles.&to;
         parent=styles.&from;
         %do i = 1 %to 12;
            %let s = %scan(&list, &i);
            %if &s ne %then %do;
               style GraphData&i from GraphData&i /
                     contrastcolor = GraphColors("gcdata&s")
                     color = GraphColors("gdata&s");
            %end;
         %end;
      end;
   run;

%mend;

%reorder(htmlblue,   /* Parent style.                              */
         MyStyle,    /* New style to create.  Specify it in an ODS  */
                     /* destination statement.                     */
         3 2 1)      /* Replace the first few GraphData colors     */
                     /* (1 2 3) with the colors from the specified  */
                     /* GraphData style elements (3 2 1).          */
                     /* You can specify up to 12 integers in the   */
                     /* range 1 - 12.                              */
```

The %Reorder macro creates a new style called MyStyle that inherits most of its attributes from the HTMLBlue style. However, in the new style, the colors for groups 1, 2, and 3 have been replaced by the colors for groups 3, 2, and 1. In other words, the colors for **GraphData1** and **GraphData3** have been switched.

The following step creates the plot:

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

The survival plot is not shown, but it matches the plot in Figure 23.51.

The rest of this section is optional. It explains how you can directly modify colors in a style template when the %Reorder macro or the technique illustrated in the section "Changing the Group Color" on page 902 is not sufficient.

The source code for the MyStyle style (as generated by the %Reorder macro) is as follows:

```
proc template;
   define style Styles.MyStyle;
      parent = styles.htmlblue;
      style GraphData1 from GraphData1 /
         color = GraphColors('gdata3')
         contrastcolor = GraphColors('gcdata3');
      style GraphData2 from GraphData2 /
         color = GraphColors('gdata2')
         contrastcolor = GraphColors('gcdata2');
      style GraphData3 from GraphData3 /
         color = GraphColors('gdata1')
         contrastcolor = GraphColors('gcdata1');
   end;
run;
```

You can create a modified style that has direct color specifications by using the colors in Figure 23.49 as follows:

```
proc template;
   define style Styles.MyStyle;
      parent = styles.htmlblue;
      style GraphData1 from GraphData1 /
         color = cx66A5A0
         contrastcolor = cx01665E;
      style GraphData2 from GraphData2 /
         color = cxD05B5B
         contrastcolor = cxA23A2E;
      style GraphData3 from GraphData3 /
         color = cx6F7EB3
         contrastcolor = cx445694;
   end;
run;
```

You can define additional GraphData*N* style elements as well. For more information about how to define style elements, see the section "Displaying Other Style Elements" on page 959.

You can delete the new style template as follows:

```
proc template;
   delete Styles.MyStyle / store=sasuser.templat;
run;
```

Displaying a Style and Extracting Font Information

You can use PROC TEMPLATE with the SOURCE statement to display a style and its parent styles in the SAS log:

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

The results of this step are long and are not shown. You can write a copy of the style templates to a file as follows:

```
proc template;
   source styles.htmlblue / expand file='style.tmp';
run;
```

The EXPAND option writes the specified style (HTMLBlue), followed by its parent (Statistical), and followed by the parent's parent (DEFAULT) to the file. The following step extracts and displays the first place in the file that the graph fonts are defined (which make the final decision in the style template):

```
data _null_;
   infile 'style.tmp' pad;
   input line $char80.;
   file print;
   if index(lowcase(line), ' graphfonts ') then y + 1;
   if y then put line $char80.;
   if y and index(line, ';') then stop;
run;
```

The results are displayed in Figure 23.52.

Figure 23.52 Graph Font Definition

```
class GraphFonts /
   'NodeDetailFont' = ("<sans-serif>, <MTsans-serif>",7pt)
   'NodeInputLabelFont' = ("<sans-serif>, <MTsans-serif>",9pt)
   'NodeLabelFont' = ("<sans-serif>, <MTsans-serif>",9pt)
   'NodeTitleFont' = ("<sans-serif>, <MTsans-serif>",9pt)
   'GraphDataFont' = ("<sans-serif>, <MTsans-serif>",7pt)
   'GraphUnicodeFont' = ("<MTsans-serif-unicode>",9pt)
   'GraphValueFont' = ("<sans-serif>, <MTsans-serif>",9pt)
   'GraphLabel2Font' = ("<sans-serif>, <MTsans-serif>",10pt)
   'GraphLabelFont' = ("<sans-serif>, <MTsans-serif>",10pt)
   'GraphFootnoteFont' = ("<sans-serif>, <MTsans-serif>",10pt)
   'GraphTitleFont' = ("<sans-serif>, <MTsans-serif>",11pt,bold)
   'GraphTitle1Font' = ("<sans-serif>, <MTsans-serif>",14pt,bold)
   'GraphAnnoFont' = ("<sans-serif>, <MTsans-serif>",10pt);
```

If the **GraphFonts** style element is defined in the HTMLBlue style, then it will appear first in the file, followed by the definitions from the Statistical style and then the DEFAULT style. In this case, the **GraphFonts** style element is defined in the DEFAULT style (last in the file), which is overridden by a definition in the Statistical style (closer to the top of the file); that is the definition that is inherited by the HTMLBlue style and displayed in Figure 23.52.

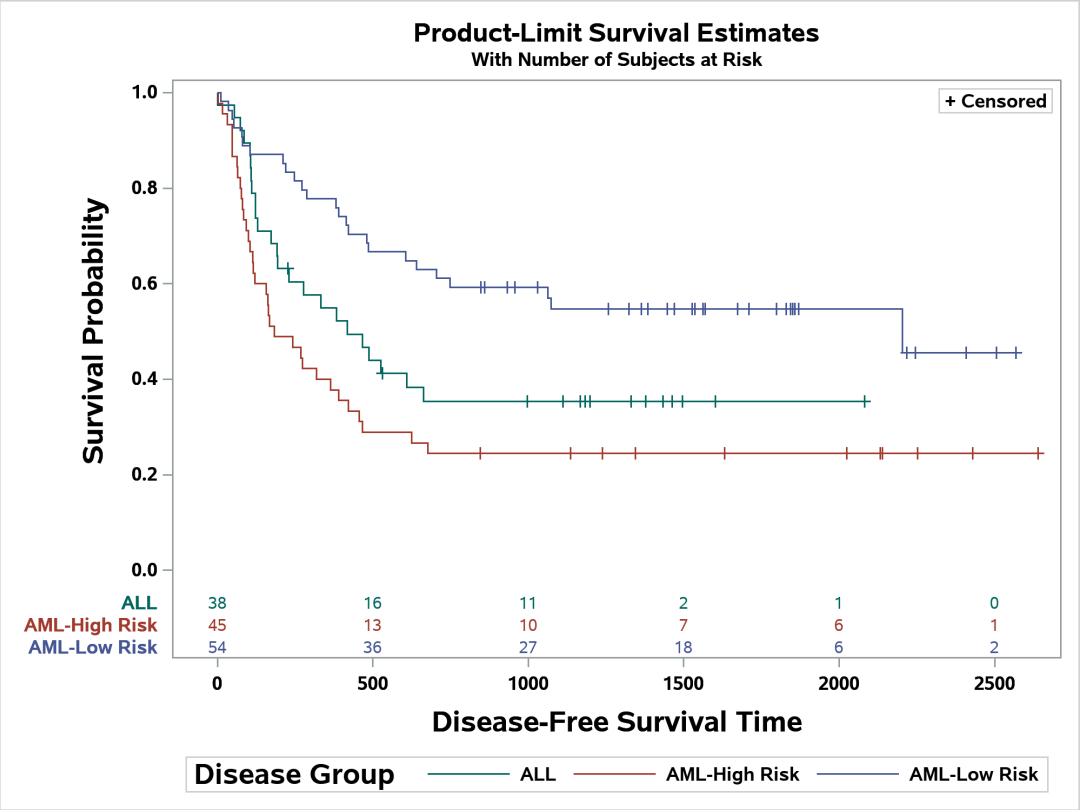
The following step creates a new style, BigFont, that changes the **GraphLabelFont** style element from a regular 10-point font to a bold 12-point font and changes the **GraphValueFont** style element from a regular 9-point font to a bold 8-point font:

```
proc template;
   define style Styles.BigFont;
      parent = Styles.HTMLBlue;
      style graphfonts from graphfonts /
         'GraphLabelFont' = ("<sans-serif>, <MTsans-serif>",12pt,bold)
         'GraphValueFont' = ("<sans-serif>, <MTsans-serif>",8pt,bold);
   end;
run;
```

The following step creates the plot that is displayed in Output 23.53:

```
ods html style=BigFont;
proc lifetest data=sashelp.BMT plots=survival(maxlen=13 atrisk);
   time T * Status(0);
   strata Group;
run;
ods html close;
```

Figure 23.53 Font Modifications



You can delete the new style template as follows:

```
proc template;
   delete Styles.BigFont / store=sasuser.templat;
run;
```

For information about making ad hoc font changes in the graph templates rather than making more global font changes in style templates, see the section "Changing the Font" on page 904.

Displaying Other Style Elements

You can use the approach from the previous example to display other style elements:

```
proc template;
   source styles.htmlblue / expand file='style.tmp';
run;

data _null_;
   infile 'style.tmp' pad;
   input line $char80.;
   file print;
   if index(lowcase(line), ' graphdata1 ') then y + 1;
   if y then put line $char80.;
   if y and index(line, ';') then stop;
run;
```

This example displays the **GraphData1** style element. The results are displayed in Figure 23.54.

Figure 23.54 GraphData1 Style Element

```
class GraphData1 /
   fillpattern = "L1"
   markersymbol = "circle"
   linestyle = 1
   contrastcolor = GraphColors('gcdata1')
   color = GraphColors('gdata1');
```

The following steps display all the **GraphFonts** style elements from all the styles:

```
proc template;
   source styles / file='style.tmp';
run;

data _null_;
   infile 'style.tmp' pad;
   length style $ 80;
   retain style;
   input line $char80.;
   file print;
   if index(lowcase(line), 'define style') then style = line;
   if index(lowcase(line), ' graphfonts ') then do;
      y + 1;
      put style $char80.;
   end;
   if y then put line $char80.;
   if index(line, ';') then y = 0;;
run;
```

The results of this step are not displayed. You can use this approach to help you better understand the options that are available for modifying styles. The SOURCE statement specifies a single-level value of STYLES rather than a specific style name such as STYLES.HTMLBLUE, so all templates that begin with STYLES as the first level (all style templates) are written to the file. The DATA step displays all definitions of **GraphFonts** and the names of all styles that define the **GraphFonts** style element.

You can insert the name of another style element (in lowercase with a leading and trailing blank) in the preceding programs in place of 'graphdata1' or 'graphfonts'. After you display a style element, you can modify the definition and create a new style that uses the modified definition, as in the example in the section "Displaying a Style and Extracting Font Information" on page 957. Some of the style elements that you might want to display and modify are listed in Table 23.3.

Table 23.3 Style Elements

AfterCaption	GraphAnnoText	GraphHeaderBackground	List3
Batch	GraphAxisLines	GraphHistogram	ListItem
Body	GraphBackground	GraphInitial	Note
BodyDate	GraphBand	GraphLabel2Text	NoteBanner
ByContentFolder	GraphBar	GraphLabelText	NoteContentFixed
Byline	GraphBlock	GraphLegendBackground	Output
BylineContainer	GraphBlockHeader	GraphMissing	PageNo
Caption	GraphBorderLines	GraphOther	Pages
Color_list	GraphBox	GraphOutlier	PagesProcLabel
Colors	GraphBoxMean	GraphOutlines	PagesTitle
Container	GraphBoxMedian	GraphOverflow	Paragraph
ContentFolder	GraphBoxWhisker	GraphPhaseBox	Parskip
ContentProcLabel	GraphClipping	GraphPrediction	PrePage
ContentTitle	GraphColors	GraphPredictionLimits	ProcTitle
Contents	GraphConfidence	GraphReference	ProcTitleFixed
Continued	GraphConfidence2	GraphRunTest	RowFooter
Data	GraphConnectLine	GraphSelection	RowFooterEmphasis
DataEmphasis	GraphContour	GraphStars	RowFooterEmphasisFixed
DataEmphasisFixed	GraphControlLimits	GraphTitle1Text	RowFooterFixed
DataFixed	GraphData1	GraphTitleText	RowFooterStrong
DataStrong	GraphData2	GraphUnderflow	RowFooterStrongFixed
DataStrongFixed	GraphData3	GraphUnicodeText	RowHeader
Date	GraphData4	GraphValueText	RowHeaderEmphasis
Document	GraphData5	GraphWalls	RowHeaderEmphasisFixed
DropShadowStyle	GraphData6	GraphZoneA	RowHeaderFixed
ErrorBanner	GraphData7	GraphZoneB	RowHeaderStrong
ErrorContentFixed	GraphData8	GraphZoneC	RowHeaderStrongFixed
ExtendedPage	GraphData9	Header	SysTitleAndFooterContainer
FatalBanner	GraphData10	HeaderEmphasis	SystemFooter
FatalContentFixed	GraphData11	HeaderEmphasisFixed	SystemTitle
Fonts	GraphData12	HeaderFixed	Table
Footer	GraphDataDefault	HeaderStrong	ThreeColorAltRamp
FooterEmphasis	GraphDataText	HeaderStrongFixed	ThreeColorRamp
FooterEmphasisFixed	GraphEllipse	HeadersAndFooters	TitleAndNoteContainer
FooterFixed	GraphError	Index	TitlesAndFooters
FooterStrong	GraphFinal	IndexItem	TwoColorAltRamp
FooterStrongFixed	GraphFit	IndexProcName	TwoColorRamp
Frame	GraphFit2	IndexTitle	UserText
Graph	GraphFloor	LayoutContainer	WarnBanner
GraphAltBlock	GraphFonts	LineContent	WarnContentFixed
GraphAnnoLine	GraphFootnoteText	List	
GraphAnnoShape	GraphGridLines	List2	

SAS Item Stores

In other sections of this chapter, you submit PROC TEMPLATE statements (either directly or through macros) to compile and save graph and style templates. Compiled templates are stored in special SAS files called item stores. Assuming that you have not modified your ODS path with an ODS PATH statement, the templates that you compile 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. **CAUTION:** 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 you delete them. You can run the following step to delete the entire Sasuser item store (including all compiled graph and style templates that you added or modified) so that ODS uses only the templates the SAS System supplies:

```
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 Chapter 21, "Statistical Graphics Using ODS."

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.

Kuhfeld, W. F. (2015). *Advanced ODS Graphics Examples*. Cary, NC: SAS Institute Inc. http://support.sas.com/documentation/prod-p/grstat/9.4/en/PDF/odsadvg.pdf.

Index