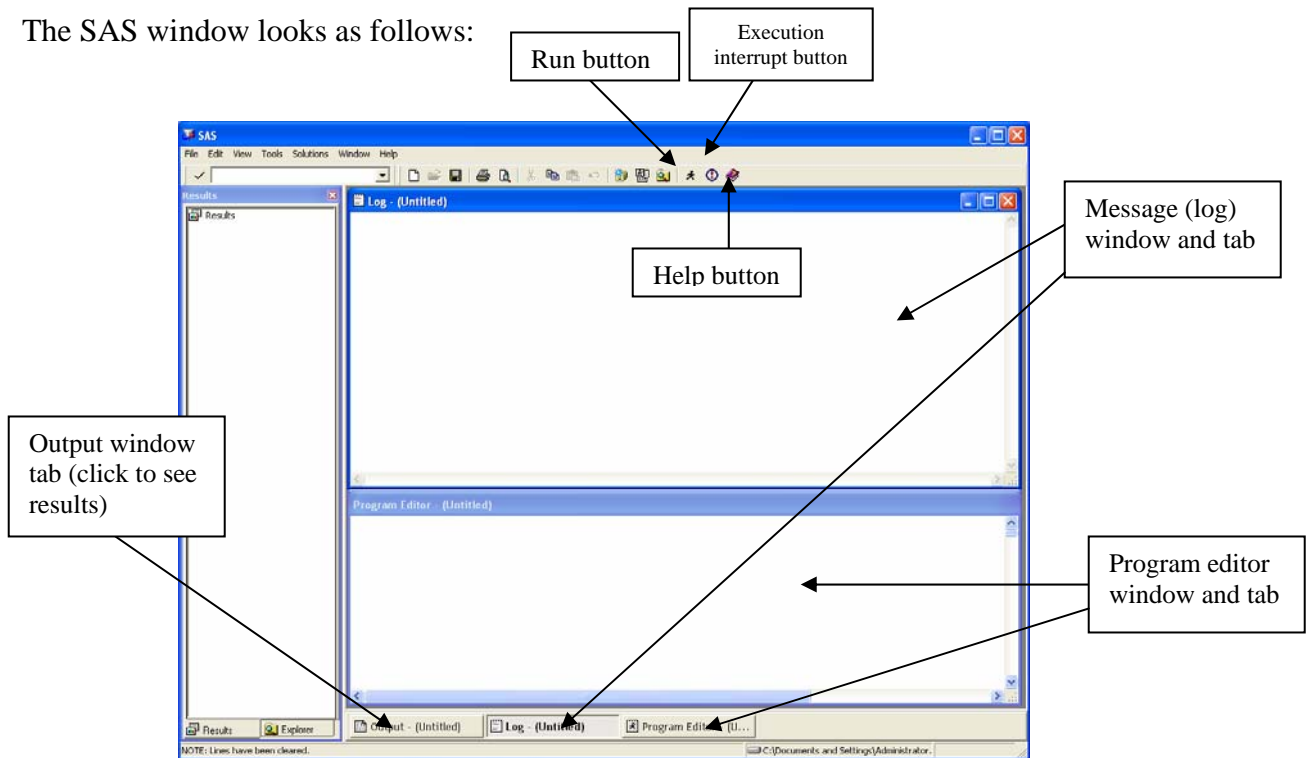


Session 1: Entering data into SAS

In this session we will “dissect” the DATA step of SAS

The SAS window looks as follows:



The DATA step is used to enter, edit or define a new data set in SAS. The syntax is as follows:

```
data dataname;          * Note that all lines in SAS end in ';' ;
```

Where *dataname* is a name for the new data set.

There are various ways to enter data into SAS. The simplest is by hand.

For example, consider entering the following data set of survival times (the sign “+” denotes a censored observation):

1, 2, 2, 2⁺ 3, 5, 6, 7⁺, 8, 16⁺, 17, 34⁺

We will enter these data into SAS manually, creating a variable (*nhltime*) for the survival time and a second (*fail*) for the censoring indicator. We will call this data set *nhodlymph*, for non-Hodskin’s lymphoma.

The SAS code is as follows:

```
data nhodlymph;
  input nhltime fail;
  datalines;
  1 1
  2 1
  2 1
  2 0
  3 1
  5 1
  6 1
  7 0
  8 1
  16 0
  17 1
  34 0
;
run;
```

Consider the components of this data step:

```
input nhltime fail;
```

This is the input statement that tells SAS that variables are to be inputted and their names.

After an input statement, there needs to be a statement declaring how the variables will be entered in the data. In our case, we will be entering them manually, so we write

```
datalines;
```

The values of the data points will follow. Note that they need to be entered in the order that were declared in the input statement. The first one will be a value for the variable `nhltime`, the second one will be associated with `fail`, the third with `nhltime`, the fourth with `fail` and so on. Unpredictable things will happen if SAS does not find something and goes looking for a value in the next line (for example, if there is no value for `fail` in the third line, the value 2 in the fourth line will become a value for `fail`, the value 0 a value for `nhltime` and so on (you don't want that!). If the values that you are entering are truly missing, then put a period (“.”) to signify that the value is missing and satisfy SAS (missing character values become more complicated, so we will not discuss them here).


Notice also that there are no semicolons (“;”) until the data entry stream is complete.

Every piece of code should end with the statement

```
run;
```

although this is not strictly necessary (as soon as SAS finds another data step or procedure it will execute the code; however it is a good habit to establish, especially if we run code

in segments as we will show later on).

Now let's try to run the data. You do that by clicking on the icon  on the tool bar. Alternatively you can press the F8 key. SAS will produce the following comments:

```
39 data nhodlymph;
40     input nhltime censor;
41     datalines;

NOTE: The data set WORK.NHODLYMPH has 12 observations and 2 variables.
NOTE: DATA statement used:
      real time           0.03 seconds
      cpu time            0.03 seconds

54 ;
55 run;
```

You do not want to see any other color in the comments (especially red!). Warnings will be in green.

Now let's print the data set. SAS accomplishes all actions through "procedures" or PROCs. These have a standard syntax. Consider the simplest of all, the printing procedure. To print the data set nhodlymph we do the following:

```
proc print data=nhodlymph;
    title "Non-hodgkin's lymphoma data set";
run;
```

Now consider the components of this procedure:

```
proc print data=nhodlymph;
```

After the statement `proc` follows the name of the procedure (here `print`). Then follows the name of the data that will be processed. Here this is `nhodlymph`. The data set is recognized as the name following the statement `data=`. It is not strictly necessary to use this. SAS will use, by default, the last data set created. However, it is a good habit to *always* write the data set name, since, eventually, you will have created many data sets and you will not know which one is the "default" one. The `proc` statement can also include several options. It is ended by a semicolon.

Notice the title statement in the second line of the previous code:

```
title "Non-hodgkin's lymphoma data set";
```

This is a title that will be attached to the output. Titles in SAS remain active until superseded by another title. You can attach subtitles by numbering them. The first title is `title1` or simply `title`. The second subtitle is `title2` and so on.

Note: Higher level (lower number) titles remove lower-level titles but do not remove higher-level titles!

The text of a title is included between single or double quotes. To include an apostrophe (right single quote) in a title, you must use double quotes around the text and the apostrophe (as shown above).

The results are as follows:

Non-hodgkin's lymphoma data set			2
			06:07 Monday, December 8, 2003
Obs	nhltime	fail	
1	1	1	
2	2	1	
3	2	1	
4	2	0	
5	3	1	
6	5	1	
7	6	1	
8	7	0	
9	8	1	
10	16	0	
11	17	1	
12	34	0	

The SAS output includes a time/date stamp and a page stamp. These can be removed, but we will not concern ourselves with this at present.

Note that, by default, SAS prints an observation counter (under Obs above). If, for some reason, you want to remove it, type the previous print procedure with the NOOBS option as follows:

```
proc print data=nhodlymph noobs;  
  title "Non-hodgkin's lymphoma data set";  
run;
```

Now let's complete the laboratory session by carrying out a simple Kaplan-Meier analysis of this data set. This is accomplished in SAS with another procedure named `lifetest`. The code is as follows:

```
proc lifetest data=nhodlymph method=pl plots=(s);  
  time nhltime*fail(0);  
  title "Kaplan-Meier analysis of the non-Hodgkin's lymphoma data set";  
run;
```

Let's analyze this procedure.

```
proc lifetest data=nhodlymph method=pl plots=(s);
```

The syntax of the procedure is standard. There are however, two options that request specific analyses. The first is `method` and the second is `plot`. The `method` option tells SAS what method of estimation of the survival distribution to use. There are two

methods: The Kaplan-Meier method (denoted by the letters PL, i.e., product-limit, or KM, Kaplan-Meier estimator). These are entered following an equal sign “=”. The requested plots can be survival plots “(s)”, minus log-survival plots (i.e., $-\log S(t)$ versus time t) “(ls)”, log-minus-log survival plots ($\log[-\log S(t)]$ versus t) “(lls)”, hazard “(h)”, probability distribution function ($F(t)$) plots “(pdf)” and a plot of the censored observations “(c)”. You can request multiple plots by entering a list of plots separated by commas. The output of the previous commands is as follows:

Kaplan-Meier analysis of the non-Hodgkin's lymphoma data set					4
					06:07 Monday, December 8, 2003
The LIFETEST Procedure					
Product-Limit Survival Estimates					
nhltime	Survival	Failure	Survival Standard Error	Number Failed	Number Left
0.0000	1.0000	0	0	0	12
1.0000	0.9167	0.0833	0.0798	1	11
2.0000	.	.	.	2	10
2.0000	0.7500	0.2500	0.1250	3	9
2.0000*	.	.	.	3	8
3.0000	0.6563	0.3438	0.1402	4	7
5.0000	0.5625	0.4375	0.1482	5	6
6.0000	0.4688	0.5313	0.1503	6	5
7.0000*	.	.	.	6	4
8.0000	0.3516	0.6484	0.1517	7	3
16.0000*	.	.	.	7	2
17.0000	0.1758	0.8242	0.1456	8	1
34.0000*	.	.	.	8	0

NOTE: The marked survival times are censored observations.

Summary Statistics for Time Variable nhltime

Quartile Estimates

Percent	Point Estimate	95% Confidence Interval [Lower Upper)	
75	17.0000	6.0000	.
50	6.0000	3.0000	17.0000
25	2.5000	2.0000	8.0000

Mean Standard Error

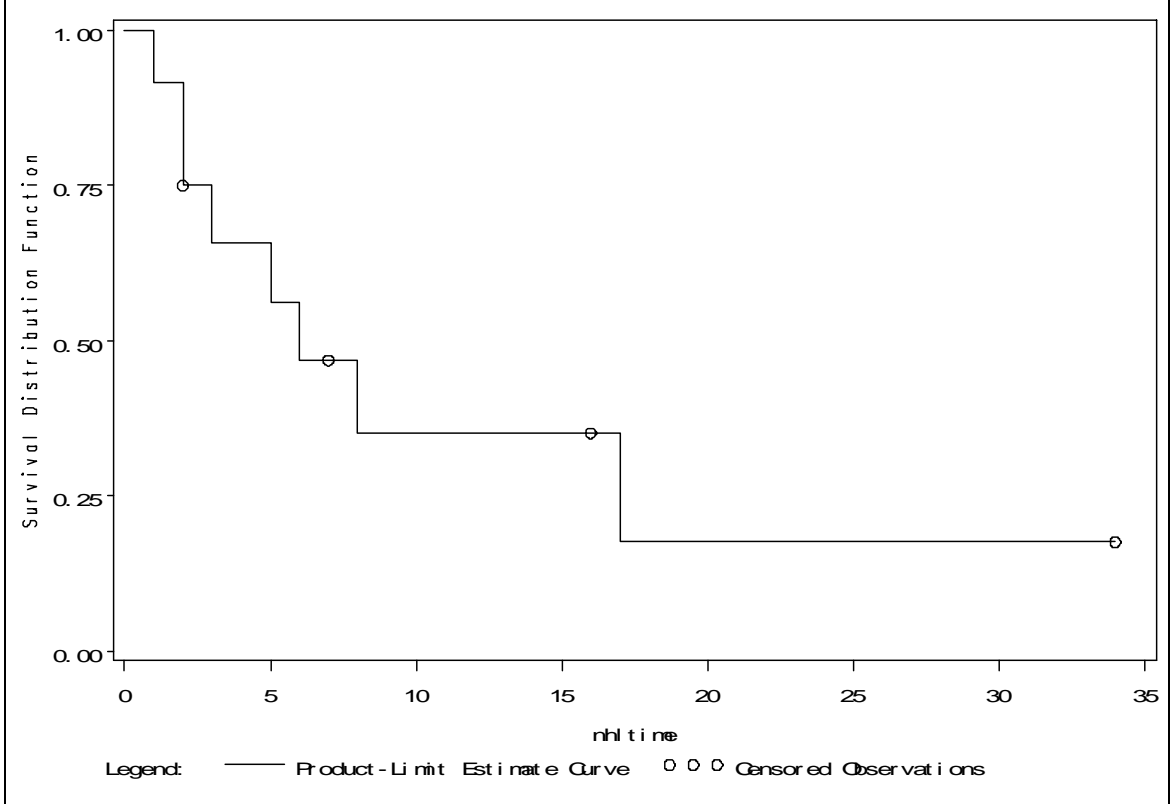
8.6432 2.1132

NOTE: The mean survival time and its standard error were underestimated because the largest observation was censored and the estimation was restricted to the largest event time.

Summary of the Number of Censored and Uncensored Values

Total	Failed	Censored	Percent Censored
12	8	4	33.33

Kaplan-Meier analysis of the non-Hodgkin's lymphoma data set



Session 2: Reading data from text files

In this session we will enter data into SAS from a text file using the `infile` statement.

Consider the leukemia data set `leukemia.dat`. This is a text file containing three variables: `trt`, `remiss` and `status`, containing information on treatment arm, remission time and status (1=event, 0=censored observation).

Before we present the data step consider how format information is generated in SAS. It is always a good idea not to use character fields in data, but rather use numerical codes and then associate these codes with a format (like the `label` command in STATA).

Formatting is done through `PROC FORMAT`. In the following code we create two formats: One for the censor indicator called `status` (it does not matter that the name of the format is the same as the name of the variable `status`) and one for the treatment group called `trtfmt` as follows:

```
proc format;
  value trtfmt 0='Control' 1='6-MP';
  value status 0='Censored' 1='Event';
run;
```

The following components of the format procedure are important. First the heading

```
proc format;
```

The heading takes no options. Then follows the word `value`, which is followed by the name of the format, which is followed in turn by the numerical fields that are equated with their character interpretations in (single or double) quotes. Note that there is no semicolon until the definition of the format is complete.

Here we have created two formats

```
value trtfmt 0='Control' 1='6-MP';
value status 0='Censored' 1='Event';
```

Thus, `trtfmt` assigns the word “control” to 0 and “6-MP” to 1, while the format `status` assigns the word “censored” to 0 and “event” to 1. This is an example of a numerical format. You can easily have character formats. For example, suppose that you have data where “Y” is yes and “N” is no. Then a character format assigning “N” to “No” and “Y” to “Yes” is as follows:

```
value $yesno 'Y'='Yes' 'N'='No';
```

Notice the “\$” before the value `yesno`, denoting that this is a character format.

Running the format procedure we get the following comments in the log file:

```
1  proc format;
2     value trtfmt 0='Control' 1='6-MP';
NOTE: Format TRTFMT has been output.
3     value status 0='Censored' 1='Event';
NOTE: Format STATUS has been output.
4     run;

NOTE: PROCEDURE FORMAT used:
      real time          0.15 seconds
      cpu time           0.03 seconds
```

Be aware that, at this point, no format has been attached to any variable in any data set. Now let's enter the leukemia data set. The code is as follows:

```
data leukemia;
    infile 'leukemia.dat';
    input trt remiss status;
    format trt trtfmt. status status.;
    label remiss='Time to end of remission'
           status='Censoring indicator'
           trt='Treatment assignment';
run;
```

Concentrate on the following statement:

```
infile 'leukemia.dat';
```

This indicates to SAS which file to get the data from. There should be some path information available inside the quotes. For example, if the file leukemia.dat (which *must* be a text file) is in a diskette, you should write

```
infile 'a:\leukemia.dat';
```

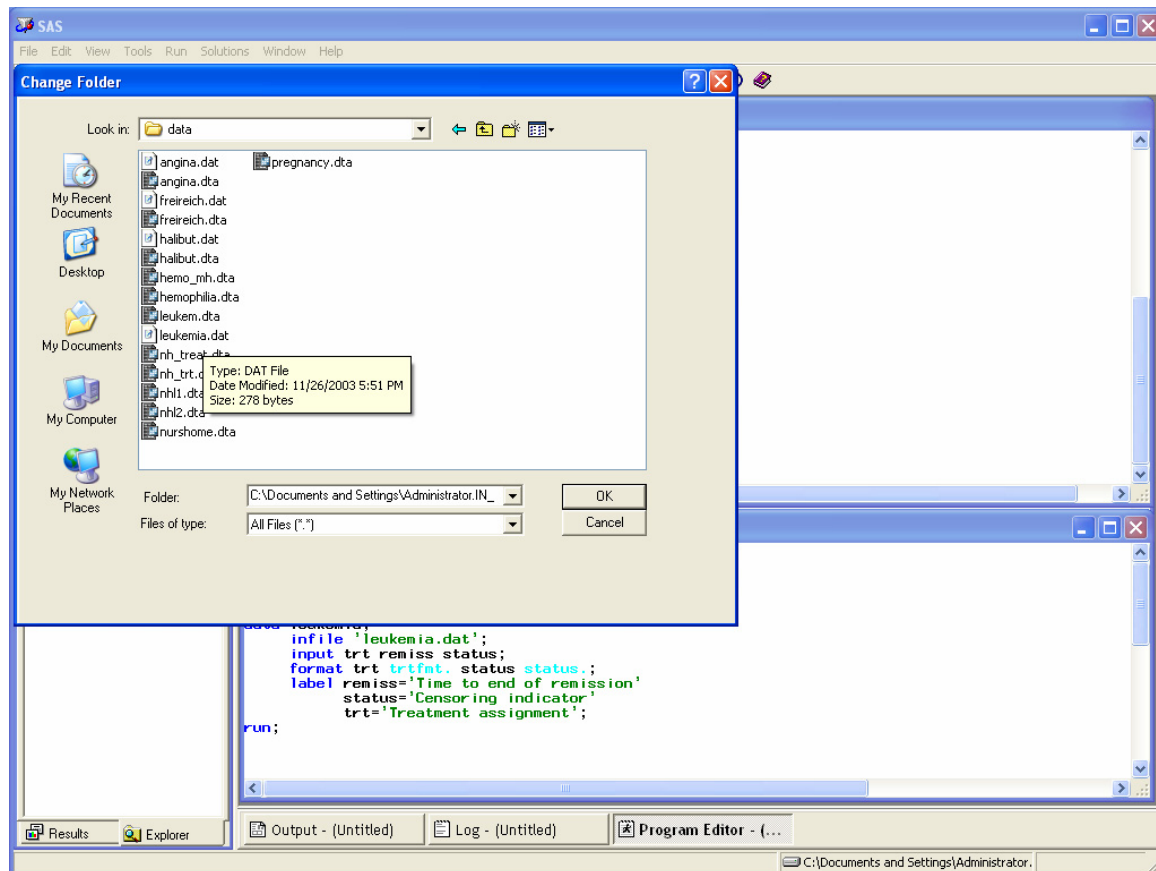
Alternatively you can make the directory where the file is located the *default* directory for SAS. The default directory is located at the bottom right corner of the SAS window.

You can change it by double-clicking the mouse on it and using the Windows change folder window to search for the appropriate directory archive (see next page). Once you find the appropriate directory you choose it by pressing the OK button.

Then you can run the data step being certain that SAS will look in the appropriate directory for the data file. Now to assign the appropriate formats to the appropriate variables you should write

```
format trt trtfmt. status status.;
```

Notice that, to indicate which word represents a format, you follow the format name with a period (“.”). If more than one variable have the same format, you can have a list of variables followed by their common format name.



The second novel part of the data step is the `label` statement.

```
label remiss='Time to end of remission'
      status='Censoring indicator'
      trt='Treatment assignment';
```

This assigns a name to the variables, which will appear in graphs and tables and will provide more information for the variable than its SAS name (and will make your output friendlier to non-statisticians).

Running the data step we have the following output in the log file:

```
6 data leukemia;
7   infile 'leukemia.dat';
8   input trt remiss status;
9   format trt trtfmt. status status.;
10  label remiss='Time to end of remission'
11        status='Censoring indicator'
12        trt='Treatment assignment';
13  run;
```

NOTE: The infile 'leukemia.dat' is:

```
File Name=C:\Documents and Settings\cyiannou\Desktop\BIO223 (Survival-
Yiannoutsos)\data\leukemia.dat, RECFM=V,LRECL=256
```

NOTE: 42 records were read from the infile 'leukemia.dat'.
The minimum record length was 5.
The maximum record length was 7.

NOTE: The data set WORK.LEUKEMIA has 42 observations and 3 variables.

NOTE: DATA statement used:

```
real time      0.74 seconds
cpu time       0.10 seconds
```

Printing the data set we get

```
proc print data=leukemia label;  
  title 'Leukemia data set';  
run;
```

Notice the option `label` in the invocation of the procedure. This will make the variables to be headed by their label rather than their SAS name. The output is as follows:

Obs	Treatment assignment	Time to end of remission	Censoring indicator
1	Control	1	Event
2	Control	1	Event
3	Control	2	Event
4	Control	2	Event
5	Control	3	Event
6	Control	4	Event
7	Control	4	Event
8	Control	5	Event
9	Control	5	Event
10	Control	8	Event
11	Control	8	Event
12	Control	8	Event
13	Control	8	Event
14	Control	11	Event
15	Control	11	Event
16	Control	12	Event
17	Control	12	Event
18	Control	15	Event
19	Control	17	Event
20	Control	22	Event
21	Control	23	Event
22	6-MP	6	Censored
23	6-MP	6	Event
24	6-MP	6	Event
25	6-MP	6	Event
26	6-MP	7	Event
27	6-MP	9	Censored
28	6-MP	10	Censored
29	6-MP	10	Event
30	6-MP	11	Censored
31	6-MP	13	Event
32	6-MP	16	Event
33	6-MP	17	Censored
34	6-MP	19	Censored
35	6-MP	20	Censored
36	6-MP	22	Event
37	6-MP	23	Event
38	6-MP	25	Censored
39	6-MP	32	Censored
40	6-MP	32	Censored
41	6-MP	34	Censored
42	6-MP	35	Censored

Now let's run the `lifetest` procedure in order to carry out the Kaplan-Meier analysis on this data set, *stratifying* it by treatment assignment and carrying out the log-rank test. The SAS code is as follows:

```
proc lifetest data=leukemia method=pl plot=(s);  
  time remiss*status(0);  
  strata trt;  
  title 'Analysis of the effect of treatment on remission from leukemia';  
run;
```

The output is as follows:

Analysis of the effect of treatment on remission from leukemia						2
						06:00 Tuesday, December 9, 2003
The LIFETEST Procedure						
Stratum 1: trt = 6-MP						
Product-Limit Survival Estimates						
remiss	Survival	Failure	Survival Standard Error	Number Failed	Number Left	
0.0000	1.0000	0	0	0	21	
6.0000	.	.	.	1	20	
6.0000	.	.	.	2	19	
6.0000	0.8571	0.1429	0.0764	3	18	
6.0000*	.	.	.	3	17	
7.0000	0.8067	0.1933	0.0869	4	16	
9.0000*	.	.	.	4	15	
10.0000	0.7529	0.2471	0.0963	5	14	
10.0000*	.	.	.	5	13	
11.0000*	.	.	.	5	12	
13.0000	0.6902	0.3098	0.1068	6	11	
16.0000	0.6275	0.3725	0.1141	7	10	
17.0000*	.	.	.	7	9	
19.0000*	.	.	.	7	8	
20.0000*	.	.	.	7	7	
22.0000	0.5378	0.4622	0.1282	8	6	
23.0000	0.4482	0.5518	0.1346	9	5	
25.0000*	.	.	.	9	4	
32.0000*	.	.	.	9	3	
32.0000*	.	.	.	9	2	
34.0000*	.	.	.	9	1	
35.0000*	.	.	.	9	0	

NOTE: The marked survival times are censored observations.

Summary Statistics for Time Variable remiss				
Quartile Estimates				
Percent	Point Estimate	95% Confidence Interval [Lower Upper)		
75	.	23.0000	.	
50	23.0000	13.0000	.	
25	13.0000	6.0000	23.0000	
Mean		Standard Error		
17.9092		1.6474		

NOTE: The mean survival time and its standard error were underestimated because the largest observation was censored and the estimation was restricted to the largest event time.

The LIFETEST Procedure

Stratum 2: trt = Control

Product-Limit Survival Estimates

remiss	Survival	Failure	Survival Standard Error	Number Failed	Number Left
0.0000	1.0000	0	0	0	21
1.0000	.	.	.	1	20
1.0000	0.9048	0.0952	0.0641	2	19
2.0000	.	.	.	3	18
2.0000	0.8095	0.1905	0.0857	4	17
3.0000	0.7619	0.2381	0.0929	5	16
4.0000	.	.	.	6	15
4.0000	0.6667	0.3333	0.1029	7	14
5.0000	.	.	.	8	13
5.0000	0.5714	0.4286	0.1080	9	12
8.0000	.	.	.	10	11
8.0000	.	.	.	11	10
8.0000	.	.	.	12	9
8.0000	0.3810	0.6190	0.1060	13	8
11.0000	.	.	.	14	7
11.0000	0.2857	0.7143	0.0986	15	6
12.0000	.	.	.	16	5
12.0000	0.1905	0.8095	0.0857	17	4
15.0000	0.1429	0.8571	0.0764	18	3
17.0000	0.0952	0.9048	0.0641	19	2
22.0000	0.0476	0.9524	0.0465	20	1
23.0000	0	1.0000	0	21	0

Summary Statistics for Time Variable remiss

Quartile Estimates

Percent	Point Estimate	95% Confidence Interval [Lower Upper)	
75	12.0000	8.0000	17.0000
50	8.0000	4.0000	11.0000
25	4.0000	2.0000	8.0000

Mean	Standard Error
8.6667	1.4114

Summary of the Number of Censored and Uncensored Values

Stratum	trt	Total	Failed	Censored	Percent Censored
1	6-MP	21	9	12	57.14
2	Control	21	21	0	0.00

Total		42	30	12	28.57

The LIFETEST Procedure

Testing Homogeneity of Survival Curves for remiss over Strata

Rank Statistics

trt	Log-Rank	Wilcoxon
6-MP	-10.251	-271.00
Control	10.251	271.00

Covariance Matrix for the Log-Rank Statistics

trt	6-MP	Control
6-MP	6.25696	-6.25696
Control	-6.25696	6.25696

Covariance Matrix for the Wilcoxon Statistics

trt	6-MP	Control
6-MP	5457.11	-5457.11
Control	-5457.11	5457.11

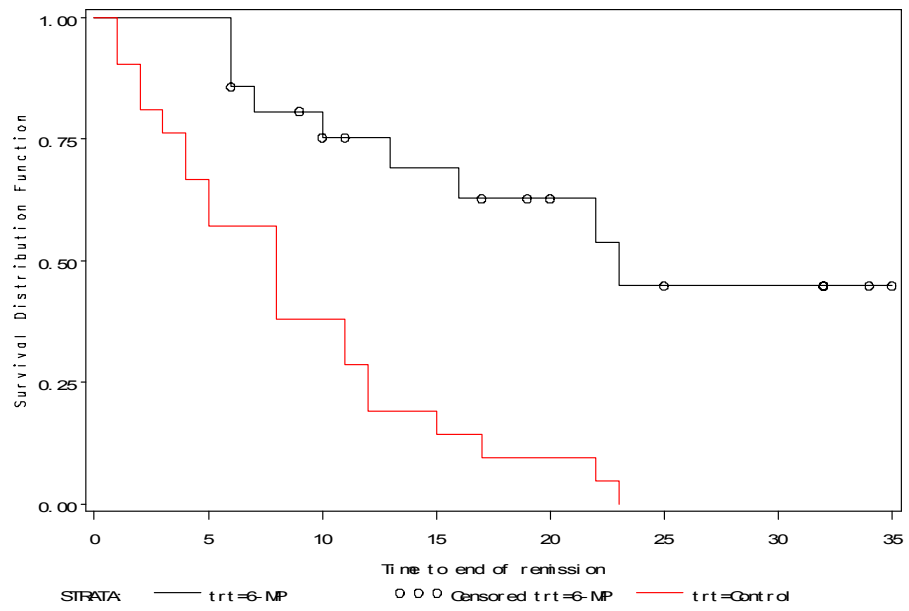
Test of Equality over Strata

Test	Chi-Square	DF	Pr > Chi-Square
Log-Rank	16.7929	1	<.0001
Wilcoxon	13.4579	1	0.0002
-2Log(LR)	16.4852	1	<.0001

The log-rank test given at the bottom of the output has test statistic 16.7929, which, compared to a chi-square distribution with one degree of freedom, is statistically significant. We conclude that there is a significant difference in survival between the two treatment groups (although we cannot tell directly from the log-rank test which treatment has the advantage).

This can be gleaned from studying the medians in the two groups (8 weeks in the control group versus 23 weeks in the treatment group), or by inspection of the Kaplan-Meier plot

Analysis of the effect of treatment on remission from leukemia



Session 3: The PHREG procedure

We will first learn how to manipulate SAS data libraries. These are catalogs of data sets in the SAS format.

Consider the statement

```
libname datalib 'data';
```

This defines the library `datalib`, which is located in the subdirectory `\data` of the default directory.

In this library it is located the data set `mac`, which is already in SAS format. We will create a new data set `mac`, reading from this data set and only including a small subset of the variables in the original data set.

The data step statements are as follows:

```
data newmac;  
  set datalib.mac (keep=patid macstat mactime rif clari cd4 karnof);  
  label patid='Patient ID'  
         macstat='Status of MAC infection'  
         mactime='Time until MAC infection'  
         rif='Rifabutin monotherapy'  
         clari='Clarithromycin monotherapy'  
         karnof='Karnofsky score'  
         cd4='CD4+ count';  
run;
```

Now consider the new statements. The statement that makes SAS read from a data set already created is the `set` statement, i.e.,

```
set datalib.mac
```

Notice also, how the library information is conveyed. We put the library information before the data name in the library, followed by a period. Thus, the *new dataset* `newmac` will read from the SAS data set `mac` that is located in the library `datalib`.

We also do not want to read all the variables from the original dataset, so we include a `keep` statement along with the `set` command.

```
set datalib.mac (keep=patid macstat mactime rif clari cd4 karnof);
```

Alternatively, the keep statement could have been added as a command after the set statement as follows:

```
data newmac;
  set datalib.mac;
  keep patid macstat mactime rif clari cd4 karnof;
  label patid='Patient ID'
        macstat='Status of MAC infection'
        mactime='Time until MAC infection'
        rif='Rifabutin monotherapy'
        clari='Clarithromycin monotherapy'
        karnof='Karnofsky score'
        cd4='CD4+ count';
run;
```

Notice that there is no longer an equal sign (“=”) or parentheses accompanying the keep statement. We will get the following comments in the log file

```
28 data newmac;
29 set datalib.mac (keep=patid macstat mactime rif clari cd4 karnof);
30 label patid='Patient ID'
31 macstat='Status of MAC infection'
32 mactime='Time until MAC infection'
33 rif='Rifabutin monotherapy'
34 clari='Clarithromycin monotherapy'
35 karnof='Karnofsky score'
36 cd4='CD4+ count';
37 run;
```

NOTE: There were 1177 observations read from the data set DATALIB.MAC.
NOTE: The data set WORK.NEWMAC has 1177 observations and 7 variables.
NOTE: DATA statement used:
real time 0.03 seconds
cpu time 0.03 seconds

To print the data set we write

```
options ls=80;
proc print data=newmac label;
  title 'The new mac data set';
run;
```

The output is as follows:

The new mac data set								88
								07:27 Wednesday, December 10, 2003
								Status of Time until
Obs	Patient ID	Karnofsky score	CD4+ count	MAC infection	MAC infection	Rifabutin monotherapy	Clarithromycin monotherapy	
1	1	90	8	1	560	1	0	
2	2	90	30	0	651	1	0	
3	3	100	80	0	26	1	0	
4	4	80	58	0	622	0	1	
5	5	90	59	0	643	0	1	
6	6	90	18	0	171	0	1	
7	7	90	20	1	174	0	0	
8	8	90	30	1	449	1	0	
9	9	80	30	1	377	0	0	
10	10	60	20	0	58	1	0	

Notice the option statement that limits the width of the output to a line size of 80 columns

```
options ls=80;
```

Now let's carry out a proportional hazards regression with variables rif, clari, karnof and cd4. The SAS statements are as follows:

```
proc phreg data=newmac;  
  model mactime*macstat(0)=rif clari karnof cd4;  
  title 'PH regression analysis of the MAC data set';  
run;
```

Notice the model statement of the PHREG procedure

```
model mactime*macstat(0)=rif clari karnof cd4;
```

First comes the time variable, linked with the status (censoring/failure indicator) by an asterisk (“*”). Then the explanatory variables follow in the same manner as all regression procedures. The output is as follows:

```
PH regression analysis of the MAC data set              105  
                                07:27 Wednesday, December 10, 2003  
  
The PHREG Procedure  
  
Model Information  
  
Data Set                WORK.NEWMAC  
Dependent Variable      mactime                Time until MAC infection  
Censoring Variable      macstat                Status of MAC infection  
Censoring Value(s)      0  
Ties Handling            BRESLOW  
  
Summary of the Number of Event and Censored Values  
  
Total      Event      Censored      Percent  
1177       121       1056         89.72  
  
Convergence Status  
  
Convergence criterion (GCONV=1E-8) satisfied.  
  
Model Fit Statistics  
  
Criterion      Without      With  
Covariates     Covariates  
  
-2 LOG L      1541.064     1477.325  
AIC           1541.064     1485.325  
SBC           1541.064     1496.508  
Testing Global Null Hypothesis: BETA=0  
  
Test           Chi-Square     DF      Pr > ChiSq  
  
Likelihood Ratio      63.7399      4      <.0001  
Score                 56.1915      4      <.0001  
Wald                  55.5623      4      <.0001  
  
Analysis of Maximum Likelihood Estimates
```

Variable	DF	Parameter Estimate	Standard Error	Chi-Square	Pr > ChiSq
rif	1	0.88034	0.23711	13.7846	0.0002
clari	1	0.25302	0.25835	0.9592	0.3274
karnof	1	-0.03685	0.01067	11.9405	0.0005
cd4	1	-0.01836	0.00368	24.8254	<.0001

Analysis of Maximum Likelihood Estimates

Variable	Hazard Ratio	Variable Label
rif	2.412	Rifabutin monotherapy
clari	1.288	Clarithromycin monotherapy
karnof	0.964	Karnofsky score
cd4	0.982	CD4+ count

First we obtain information about the convergence of the model, as well as information about the significance of the whole model (i.e., likelihood ratio tests, AIC, BIC factors that we use to compare between models and so on).

Wald tests (chi-square) and hazard ratios are given for all variables.

Session 4: Predicted survival

Today we will familiarize ourselves with more of the capabilities of PROC PHREG.

1. Predicted survival

Let's start by running the nursing home data set and ultimately produce the predicted survival from the PH regression model.

First we generate the nursing home data set, which we read in from a text file as follows:

```
proc format;
  value marfmt 0='Single' 1='Married';
run;

data nurshome;
  infile 'nurshome.dat';
  input los age rx gender married health fail;
  label los='Length of stay'
        rx='Treatment'
        married='Marriage status'
        health='Health index'
        fail='Censoring index';
  format married marfmt.;
run;
```

Note the format statement and the rest of the data-step statements.

The PROC PHREG statements are as follows:

```
proc phreg data=nurshome;
  model los*fail(0)=married health;
  output out=outsurv survival=predsuvr;
  title 'PH regression analysis of nursing home data';
run;
```

Note the new statement

```
output out=outsurv survival=predsuvr;
```

This statement produces a data set named `outsurv`, which includes, beyond the variables `married` and `health`, the variables `los` and `fail` and the predicted survival at each value of `los`, `presurv`.

We would like to sort the data by combination group of `married` and `health`. If we would like to maintain the original, unsorted, data set, then we output the sorted data set with a different name. This we accomplish as follows:

```
proc sort data=outsurv out=prsurvsort;
  by married health los;
run;
```

The statement that produces a new data set `prsurvsort` is `out=prsurvsort` (otherwise the sorted data set will be stored in the overwritten `outsurv` data set).

The print out of the new, sorted, data set is as follows (note that sorting is accomplished from right to left, with `los` being sorted within `health` and the latter within `married`).

```
proc print label data=prsurvsort;
  var married health los predsurv;
  title 'Printout of the predicted survival from nursing home data';
run;
```

```
Printout of the predicted survival from nursing home data
14
```

Obs	Marriage status	Health index	Length of stay	Survivor Function Estimate
1	Single	2	1	0.98961
2	Single	2	1	0.98961
3	Single	2	1	0.98961
4	Single	2	1	0.98961
5	Single	2	1	0.98961
6	Single	2	1	0.98961
7	Single	2	1	0.98961
8	Single	2	2	0.98156
9	Single	2	2	0.98156
10	Single	2	2	0.98156
.
.
.

We can also produce survival estimates for specific values of the covariates. For this we proceed as follows:

```
data cov;
  married=0; health=2;
run;
```

thus, we are creating a data set with only one observation and two variables set at the desired values for the two covariates.

Then the `PHREG` command becomes as follows:

```
proc phreg data=nurshome;
  model los*fail(0)=married health;
  output out=outsurv covariates=cov survival=predsurr/nomean;
  title 'PH regression analysis of nursing home data';
run;
```

notice the addition in the data set line

```
output out=outsurv covariates=cov survival=predsurr/nomean;
```

This specifies to SAS that you want to estimate survival at these two specific values of the covariate, that is, single (married=0) healthy (health=2) persons. This will create estimates for the survival at all event times. The option nomean excludes the generation of survival estimates for the mean of the two covariates.

```
proc print label data=outsurv;
  title 'Printout of the predicted survival from nursing home data';
  title2 'For single healthy persons';
run;
```

The printed survival dataset is as follows:

Printout of the predicted survival from nursing home data					353
For single healthy persons					
Obs	Marriage status	Health index	Length of stay	Survivor Function Estimate	
1	Single	2	0	1.00000	
2	Single	2	1	0.98961	
3	Single	2	2	0.98156	
4	Single	2	3	0.97728	
5	Single	2	4	0.96917	
.	
.	
.	
371	Single	2	653	0.29394	
372	Single	2	654	0.29269	
373	Single	2	663	0.29143	
374	Single	2	664	0.29017	
375	Single	2	665	0.28891	
376	Single	2	674	0.28761	

Calculating the median survival

There are 3 possible approaches to calculating the median:

1. Calculate the median from a specified covariate combination and perform a Kaplan-Meier analysis

The problem with this approach is that we cannot do this for combinations where the covariate combination does not exist. For example, in the nursing home data, there are no individuals with health index 0 (i.e., totally healthy). So any combination in survival involving healthy individuals cannot be performed by this approach.

2. Generate predicted survival curves for each combination of covariates and obtain the medians directly.

For example, in the nursing home data we go to the printout about and look for the observations where the predicted survival (predsurv) goes from above 50% to below 50%.

Printout of the predicted survival from nursing home data					
	Obs	married	health	los	predsurv
	163	Single	2	182	0.50167
	164	Single	2	189	0.49684

	513	Single	3	139	0.50050
	514	Single	3	142	0.49863

and so on. So, for single healthy (health=2) individuals, the median survival is 189 days, while for single a bit less healthy individuals (health=3) the median survival is 142 days.

3. We may also generate the estimate from the model itself, by the formula

$$S(M; Z) = [S_0(M)]^{e^{\beta Z_i}}$$

so that the median M satisfies

$$S_0(M) = [0.5]^{e^{-\beta Z_i}}$$

As we presented in class, suppose that we wanted to estimate the median for a single unhealthy (i.e., health=5) subject. This is

$$S_0(M) = [0.5]^{e^{-\beta Z_i}} = [0.5]^{e^{-(\beta_1 Z_1 + \beta_2 Z_2)}}$$

From the output of the PHREG procedure we know that $\beta_2=0.165$ and $Z_1=0$ (single person), so the above becomes

$$S_0(M) = [0.5]^{e^{-(\beta_1 Z_1 + \beta_2 Z_2)}} = (0.5)^{e^{-(0.165 \times 5)}} = 0.7385$$

This is the *baseline* survival function (i.e., the survival function associated with an individual with married=0 and health=0)!!!

So we must find the predicted survival for a covariate combination that does not exist in the data. To do this with SAS we proceed as follows:

```
* Set covariate at single (married=0) and healthy (health=0);
data cov0;
  married=0;
  health=0;
run;
```

i.e., we've created a data set cov0 with married=0 and single=0. Then we invoke PHREG as follows:

```
proc phreg data=nurshome;  
  model los*fail(0)=married health;  
  baseline out=outsurv survival=preds surv covariates=cov0/nomean;  
  title 'PH regression analysis of nursing home data';  
run;
```

notice the code fragment

```
baseline out=outsurv survival=preds surv covariates=cov0/nomean;
```

By printing the data `outsurv` we get the predicted *baseline* survival in the variable `predsurv` and if we look at the point where the predicted baseline survival goes from above 0.7385 to below 0.7385. We have

```
Printout of the predicted survival from nursing home data
```

Obs	married	health	los	predsurv
.
.
.
80	Single	0	80	0.73851
81	Single	0	81	0.73672
82	Single	0	82	0.73492

so that the median is about 80 days.

Session 5: Model selection

1. Univariate analyses of the halibut data set

Now we input the data set `halibut.dat`. This is done through the following SAS statements:

```
data halibut;
  infile 'halibut.dat';
  input id  survtime  censor  towdur  depth  length  handling  logcatch;
  label survtime='Length of survival'
        depth='Depth'
        handling='Handling time'
        towdur='Duration of towing'
        length='Length of fish'
        logcatch='Logarithm of the total catch';
run;
```

We would like first to produce univariate PH regressions. This is straightforward with PHREG, but, to generate a graph, we will discretize each continuous factor. One such straightforward discretization is to dichotomize the factor as below or above the median. In turn, to obtain the medians we use PROC UNIVARIATE as follows:

```
proc univariate data=halibut;
  var towdur length depth handling logcatch;
  title 'Descriptive statistics for main explanatory variables';
run;
```

From this we see that the median of `towdur` is 100. We can incorporate this directly into the statement of the PROC PHREG as follows:

```
proc phreg data=halibut noprint;
  model survtime*censor(0)=disctd;
  disctd=(towdur<100);
  id towdur;
  title 'Survival time by discretized tow duration (above vs. below median)';
  output out=outsurv survival=preds surv;
run;
```

The output from the previous statements is suppressed with the `noprint` statement (which allows us to produce the output data set `outsurv` without getting lengthy output).

The problem with this data set is that the variable `towdur` (since it is not part of the model) is not included in `outsurv`. To insert it there we use the `id` statement, i.e.,

```
id towdur;
```

Let's see what this data set looks like:

```
proc print data=outsurv;
  title 'Predicted survival with respect to tow duration';
```

```
run;
```

Predicted survival with respect to tow duration						
49	Obs	towdur	survtime	cancel	disctd	predsurv
	1	30	209.0	1	1	0.16346
	2	30	209.0	1	1	0.16346
	3	30	209.0	1	1	0.16346
	4	30	209.0	1	1	0.16346
	5	30	38.0	1	1	0.56354
	6	30	209.0	1	1	0.16346
	7	30	140.9	1	1	0.30191
	8	30	140.9	1	1	0.30191
	9	30	140.1	1	1	0.31248
	10	30	208.0	1	1	0.19483
	11	30	140.1	1	1	0.31248

Now to produce a graph we need to format the new variable `disctd`.

```
proc format;  
  value bivarfmt 1='Below median' 0='Above median';  
run;
```

We update the `oursurv` data set as follows:

```
proc format;  
  value bivarfmt 1='Below median' 0='Above median';  
run;  
  
data outsurv;  
  set outsurv;  
  format disctd bivarfmt.;  
run;
```

Now let's produce the graph. First we must define the symbols and lines used in the graph. We have two groups, so we define two symbols as follows:

```
symbol1 c=red line=1 i=stepljs value=plus;  
symbol2 c=green line=1 i=stepljs value=plus;
```

The option `c=red` and `c=green` determine that the first plot (for `disctd==0`) will be in red color and the second (for `disctd==1`) in green.

The next option `line=1` specifies that both lines will be solid. Broken lines of varying widths can be specified by increasing the number after the "=" sign. In both cases, the symbol itself will be a "+" (plus) sign, so `value=plus`.

Because we would like to generate a plot that will look like a Kaplan-Meier plot (even though the survival estimates were derived from a Cox model) we specify that the points between the lines must be interpolated as follows

```
i=stepljs
```

Now we must define the axes (the default axes in SAS are rather unattractive)

```
axis1 label=(angle=90 height=2.0 font='arial' 'Percent surviving' )  
       value=(font='arial' height=1.5);  
axis2 label=(height=2.0 font='arial' 'Length of survival')  
       value=(font='arial' height=1.5) minor=NONE;
```

The label of the first axis (later to be defined as the y axis) has the text 'Percent surviving' which is rotated by 90° (angle=90) has size 2.0 (height=2.0) which you need to play around with since the SAS units of size are not obvious, and the PC font is arial (font='arial'). This means that this program code might not produce the expected results in a platform that does not have this font (e.g., in UNIX). You need to limit characteristics that make your code not portable as much as possible. We also can specify pretty much every aspect of the axis. Here we choose to make the markers a bit larger by specifying value=(font='arial' height=1.5). Note the syntax that, for every attribute, has all the characteristics in a parenthesis that follows an equal sign.

The second axis, axis2, (later to be defined as the x axis) is similar, except that we have removed any minor tick marks by specifying minor=NONE.

The graph is generated by PROC GLOT as follows:

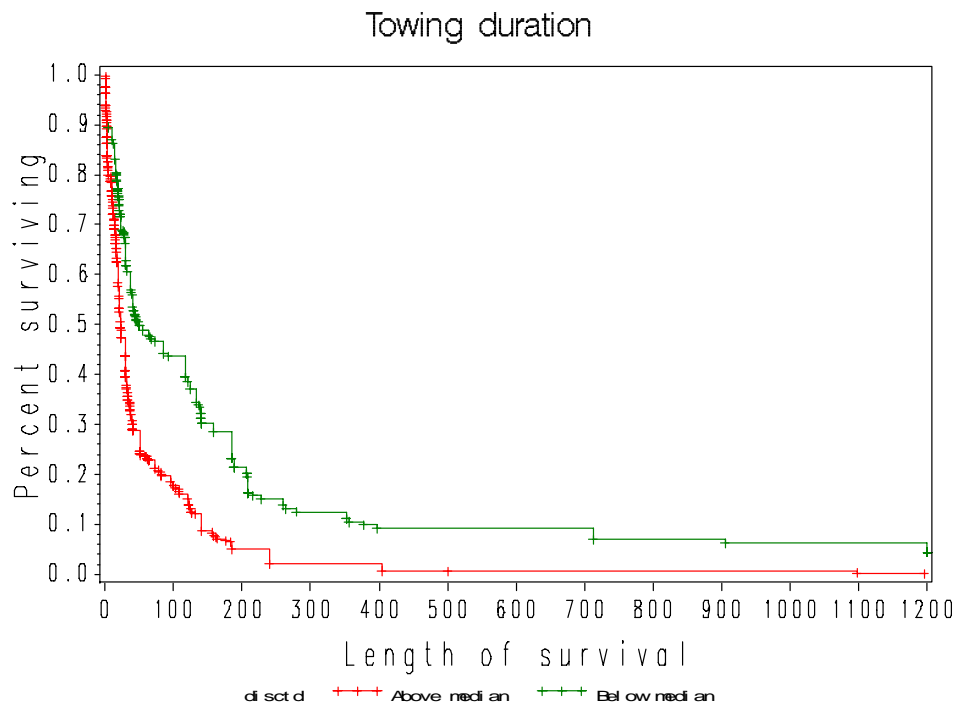
```
proc gplot data=outsurv;  
  plot predsurv*survtime=disctd/overlay vaxis=axis1 haxis=axis2;  
  title 'Towing duration';  
run;
```

The syntax of the procedure itself is familiar. The plot is generated with the statement

```
plot predsurv*survtime=disctd/vaxis=axis1  
      haxis=axis2;
```

The syntax of the plot statement is plot yvar*xvar followed by options. You can also add a categorical variable that generates as many plots as there are categories in it. Here we have included variable disctd (the discretized tow duration variable). The only option we have made define which axis is the

vertical and which is the horizontal (vaxis=axis1 and haxis=axis2 respectively). The output is as follows:



This plot suggests that longer tow duration results in shorter survival times among the halibut fish. Various plots with respect to the other continuous variables, analyzed in a univariate manner are similarly obtained.

2. Model selection

Now we describe how model selection can be automatically accomplished with PROC PHREG. We will use the stepwise method as an example understanding that the forward and backward selection methods are similar.

To carry out a model selection procedure in the halibut data set we proceed as follows:

```
proc phreg data=halibut;
  model survtime*censor(0)=towdur depth length handling
        logcatch/selection=stepwise selentry=0.2
        selstay=0.1;
  title 'Model selection of the halibut data set';
run;
```

This statement specifies that selection=stepwise and that the probability threshold for entering a variable is selentry=0.2 and that of removing one is selstay=0.1.

The output is as follows:

Model selection of the halibut data set

The PHREG Procedure

Model Information

Data Set	WORK.HALIBUT	
Dependent Variable	survtime	Length of survival
Censoring Variable	censor	
Censoring Value(s)	0	
Ties Handling	BRESLOW	

Summary of the Number of Event and Censored Values

Total	Event	Censored	Percent Censored
294	273	21	7.14

Step 1. Variable handling is entered. The model contains the following explanatory variables:

handling

Convergence Status

Convergence criterion (GCONV=1E-8) satisfied.

Model Fit Statistics

Criterion	Without Covariates	With Covariates
-2 LOG L	2599.449	2558.358
AIC	2599.449	2560.358
SBC	2599.449	2563.967

Testing Global Null Hypothesis: BETA=0

Test	Chi-Square	DF	Pr > ChiSq
Likelihood Ratio	41.0914	1	<.0001
Score	47.1417	1	<.0001
Wald	46.0330	1	<.0001

Step 2. Variable logcatch is entered. The model contains the following explanatory variables:

handling logcatch

Convergence Status

Convergence criterion (GCONV=1E-8) satisfied.

Model Fit Statistics

Criterion	Without Covariates	With Covariates
-2 LOG L	2599.449	2539.647
AIC	2599.449	2543.647
SBC	2599.449	2550.866

Testing Global Null Hypothesis: BETA=0

Test	Chi-Square	DF	Pr > ChiSq
Likelihood Ratio	59.8023	2	<.0001
Score	65.6797	2	<.0001
Wald	63.0055	2	<.0001

Model selection of the halibut data set

The PHREG Procedure

Step 3. Variable towdur is entered. The model contains the following explanatory variables:

towdur handling logcatch

Convergence Status

Convergence criterion (GCONV=1E-8) satisfied.

Model Fit Statistics

Criterion	Without Covariates	With Covariates
-2 LOG L	2599.449	2528.599
AIC	2599.449	2534.599
SBC	2599.449	2545.427

Testing Global Null Hypothesis: BETA=0

Test	Chi-Square	DF	Pr > ChiSq
Likelihood Ratio	70.8507	3	<.0001
Score	76.3454	3	<.0001
Wald	72.7407	3	<.0001

Step 4. Variable length is entered. The model contains the following explanatory variables:

towdur length handling logcatch

Convergence Status

Convergence criterion (GCONV=1E-8) satisfied.

Model Fit Statistics

Criterion	Without Covariates	With Covariates
-2 LOG L	2599.449	2515.310
AIC	2599.449	2523.310
SBC	2599.449	2537.748

Testing Global Null Hypothesis: BETA=0

Test	Chi-Square	DF	Pr > ChiSq
Likelihood Ratio	84.1397	4	<.0001
Score	94.0062	4	<.0001
Wald	90.2476	4	<.0001

Model selection of the halibut data set

55

Step 5. Variable depth is entered. The model contains the following explanatory variables:

towdur depth length handling logcatch

Convergence Status

Convergence criterion (GCONV=1E-8) satisfied.

The PHREG Procedure

Model Fit Statistics

Criterion	Without Covariates	With Covariates
-2 LOG L	2599.449	2513.769
AIC	2599.449	2523.769
SBC	2599.449	2541.817

Testing Global Null Hypothesis: BETA=0

Test	Chi-Square	DF	Pr > ChiSq
Likelihood Ratio	85.6799	5	<.0001
Score	96.1287	5	<.0001
Wald	92.0770	5	<.0001

Step 6. Variable depth is removed. The model contains the following explanatory variables:

towdur length handling logcatch

Convergence Status

Convergence criterion (GCONV=1E-8) satisfied.

Model Fit Statistics

Criterion	Without Covariates	With Covariates
-2 LOG L	2599.449	2515.310
AIC	2599.449	2523.310
SBC	2599.449	2537.748

Testing Global Null Hypothesis: BETA=0

Test	Chi-Square	DF	Pr > ChiSq
Likelihood Ratio	84.1397	4	<.0001
Score	94.0062	4	<.0001
Wald	90.2476	4	<.0001

NOTE: Model building terminates because the variable to be entered is the variable that was removed in the last step.

Analysis of Maximum Likelihood Estimates

Variable	DF	Parameter Estimate	Standard Error	Chi-Square	Pr > ChiSq	Hazard Ratio	Variable Label
towdur	1	0.00774	0.00202	14.6800	0.0001	1.008	Duration of towing
length	1	-0.03665	0.01003	13.3466	0.0003	0.964	Length of fish
handling	1	0.05490	0.00988	30.8735	<.0001	1.056	Handling time
logcatch	1	-0.18466	0.05101	13.1017	0.0003	0.831	Logarithm of the total catch

Summary of Stepwise Selection

Step	Variable Entered	Number Removed	In	Score Chi-Square	Wald Chi-Square	Pr > ChiSq	Variable Label
1	handling		1	47.1417	.	<.0001	Handling time
2	logcatch		2	18.4259	.	<.0001	Logarithm of the total catch
3	towdur		3	11.0191	.	0.0009	Duration of towing
4	length		4	13.4222	.	0.0002	Length of fish
5	depth		5	1.6661	.	0.1968	Depth
6		depth	4	.	1.6506	0.1989	Depth

Session 6: Assessing the PH Assumption

In today's lab, we are going to evaluate the assumption of proportional hazards using several graphical approaches. We will use the same example as in the lecture, the nursing home dataset (*nurshome.dat*). First we generate the nursing home data set, which we read in from a text file as follows:

```
proc format;
  value marfmt 0='Single' 1='Married';
  value sexfmt 0='Women' 1='Men';
run;

data nurshome;
  infile 'nurshome.dat';
  input los age rx gender married health fail;
  label los='Length of stay'
        rx='Treatment'
        married='Marriage status'
        health='Health index'
        fail='Censoring index';
  format married marfmt. Gender sexfmt.;
run;
```

We want to assess the proportional hazards for gender and marital status. One way is to produce the plot of $\log[-\log(S(t))]$ versus $\log(t)$. If the lines of the subgroups are parallel then the assumption is satisfied. SAS produces this as follows:

```
proc phreg data=nurshome noprint;
  model los*fail(0)=gender;
  output out=outsurv loglogs=loglogsurv;
  title 'PH regression analysis of nursing home data';
run;
```

Note here the option `noprint` in the invocation of the command, which suppresses the output (this is frequently used when we are generating data sets with a procedure only). Also note that we now are outputting in the data set `outsurv` the $\log(-\log(S(t)))$ by using the option `loglogs` .

To see what the data set looks like do the following:

```
proc print data=outsurv;run;
```

PH regression analysis of nursing home data

Obs	los	fail	gender	loglogsurv
1	665	1	0	0.37937
2	697	0	0	0.39442
3	7	1	0	-2.66415
4	217	1	0	-0.10862
.
.
.

To visualize the data we perform the following. First we generate a new data set with a variable `logt` to hold the logarithm of time (length of stay).

```
data plotdata;
  set outsurv;
  logt=log(log);
  label logt='Log of length of stay (days)';
run;
```

Then, we sort the data by log time in order to plot appropriately (this is rather critical).

```
proc sort data=plotdata;
  by logt;
run;
```

Then we perform the preparatory steps to generate the plot. Note that we simply join the points by using the following command:

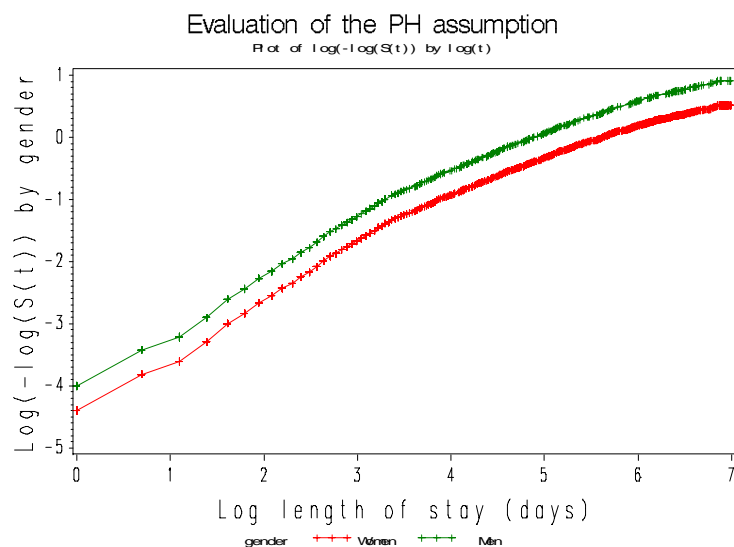
```
i=join
```

```
symbol1 c=red line=1 i=join value=plus;
symbol2 c=green line=1 i=join value=plus;
```

```
axis1 label=(angle=90 height=2.0 font='arial'
  'Log(-log(S(t))) by gender' ) value=(font='arial'
  height=1.5);
axis2 label=(height=2.0 font='arial' 'Log length of stay (days)')
  value=(font='arial' height=1.5) minor=NONE;
```

```
proc gplot data=plotdata;
  plot loglogsurv*logt=gender/overlay vaxis=axis1 haxis=axis2;
  title 'Evaluation of the PH assumption';
  title2'Plot of log(-log(S(t))) by log(t)';
run;
```

The graph looks as follows:



Recall that SAS produces the $\log(-\log(S(t)))$ (instead $-\log(-\log(S(t)))$ as in STATA). This plot is equivalent to the STATA plot using the `noneg` option.

A similar graph can be created for Marital Status:

```
proc phreg data=nurshome noprint;
  model los*fail(0)=married;
  output out=outsurv loglogs=loglogsurv;
  title 'PH regression analysis of nursing home data';
run;

proc print data=outsurv;run;
```

The data look as follows (notice that now you have marital status in the data set instead of gender).

PH regression analysis of nursing home data				
Obs	los	fail	gender	loglogsurv
1	665	1	Women	0.37937
2	697	0	Women	0.39442
3	7	1	Women	-2.66415
4	217	1	Women	-0.10862
.
.
.

```
data plotdata;
  set outsurv;
  logt=log(los);
  label logt='Log of length of stay (days)';
run;

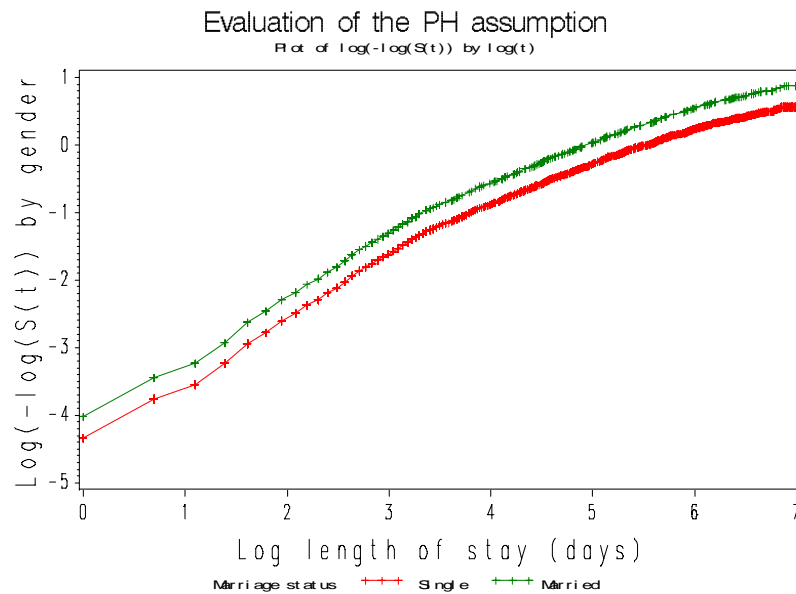
proc sort data=plotdata;
  by logt;
run;
```

```
symbol1 c=red line=1 i=join value=plus;
symbol2 c=green line=1 i=join value=plus;

axis1 label=(angle=90 height=2.0 font='arial' 'Log(-log(S(t)) by
gender' )
  value=(font='arial' height=1.5);
axis2 label=(height=2.0 font='arial' 'Log length of stay (days)')
  value=(font='arial' height=1.5) minor=NONE;

proc gplot data=plotdata;
  plot loglogsurv*logt=married/overlay vaxis=axis1 haxis=axis2;
  title 'Evaluation of the PH assumption';
  title2'Plot of log(-log(S(t)) by log(t)';
  format gender sexfmt.;
run;
```

The plot is as follows:



Now we will generate Kaplan-Meier survival curves according to gender and compare them to the Cox predicted curves for the same variable. The closer the observed values are to the predicted, the less likely the proportional hazards assumption has been violated.

There are a number of steps that need to be performed in SAS in order to accomplish this (although not really complicated they are very time consuming compared especially to the STATA command `stcoxkm`).

We generate a new data set `outsurv` that, this time, contains the estimated survival data for the PH analysis by gender.

```
proc phreg data=nurshome noprint;
  model los*fail(0)=gender;
  output out=outsurv survival=coxsurv;
  title 'PH regression analysis of nursing home data';
run;
```

This will produce a new data set named `outsurv` with the Cox prediction of survival stored in variable `coxsurv`.

Now we need to run a Kaplan-Meier analysis and save the Kaplan-Meier prediction of the same survival data. We do this using PROC LIFETEST as we've done previously.

```
proc lifetest data=nurshome outsurv=sexkmsurv method=pl noprint;
  time los*fail(0);
  strata gender;
  title 'Kaplan Meier analysis of nursing home data';
run;
```

Notice the option `outsurv=sexkmsurv` in the invocation of the procedure. This produces a data set that holds the survival estimates of the Kaplan-Meier procedure and confidence intervals.

The data set is printed and looks as follows:

```
proc print data=sexkmsurv;run;
```

Kaplan Meier analysis of nursing home data							
Obs	gender	los	_CENSOR_	SURVIVAL	SDF_LCL	SDF_UCL	STRATUM
1	Men	0	0	1.00000	1.00000	1.00000	1
2	Men	1	0	0.99282	0.98473	1.00000	1
3	Men	2	0	0.97608	0.96143	0.99073	1
4	Men	3	0	0.96172	0.94333	0.98012	1
.
.
.
238	Women	0	0	1.00000	1.00000	1.00000	2
239	Women	1	0	0.98380	0.97658	0.99103	2
240	Women	2	0	0.97528	0.96639	0.98416	2
.
.
.

Unfortunately this is a different format compared to the previous data set (where each individual has an associated survival estimate).

We will need to create a common data set that is sorted by gender and then, within each gender, by the length of stay (`los`) variable and merge them together. This will be accomplished by a new data step where the data sets `sexkmsurv` and `outsurv` will be merged *by gender los*. But before we can do this we must sort the `sexoutsurv` data set by `los`; and not by increasing order of magnitude but by decreasing (as the Kaplan-Meier data set is thus sorted), while we also must sort the data by gender (in decreasing order as well, i.e., first men and then women). This is done as follows:

```
proc sort data=outsurv;
  by descending gender los;
run;

proc print data=outsurv; run;
```

The sorted data set looks like this:

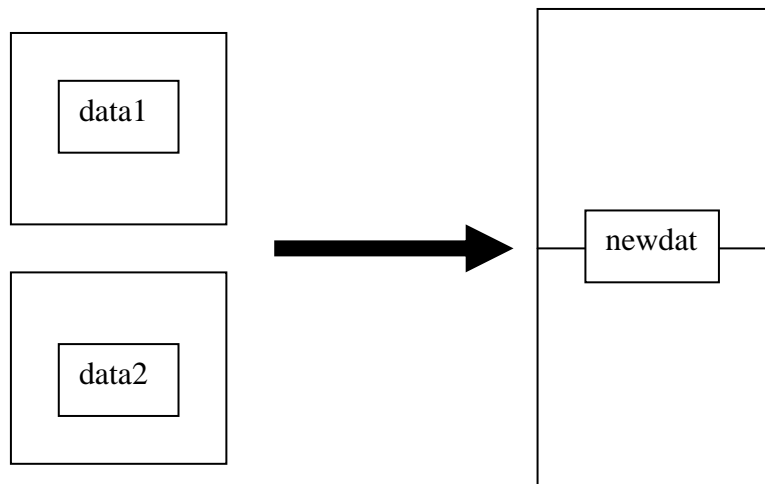
PH regression analysis of nursing home data					
Obs	los	fail	gender	coxsurv	
1	1	1	Men	0.98184	
2	1	1	Men	0.98184	
3	1	1	Men	0.98184	
.	
.	
.	
419	1	1	Women	0.98773	
420	1	1	Women	0.98773	
421	1	1	Women	0.98773	
.	
.	
.	

Now we are ready to merge the data. In SAS, there are two ways to merge two data sets: You can *set* them and you can *merge* them.

Using the set command, that is using code of the type

```
data newdata;  
  set data1 data2 ;  
  .  
  .   *More SAS statements;  
  .  
run;
```

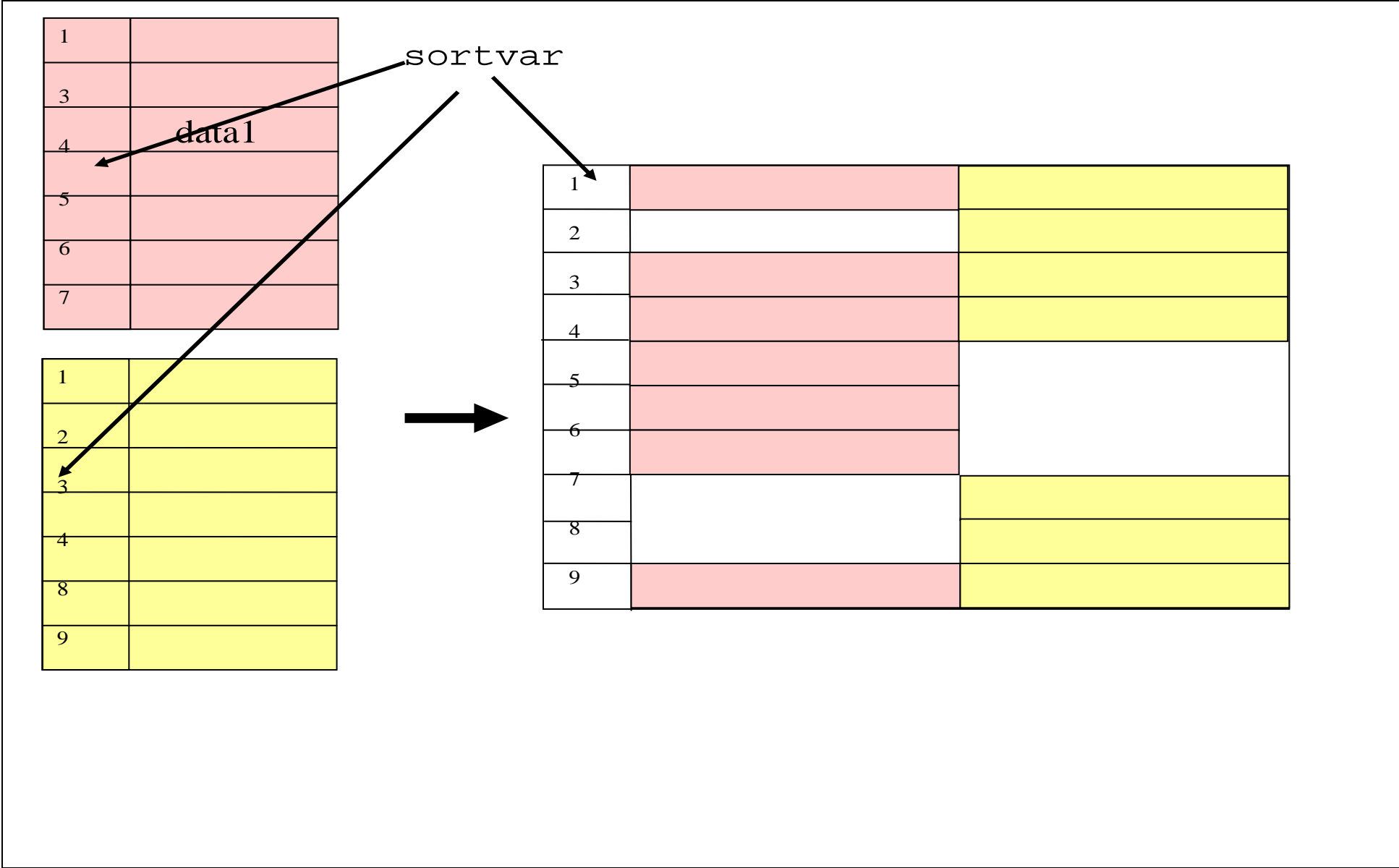
This accomplishes the following:



Merging two data sets on the other hand *by* a variable (say) `sortvar` is performed through the following SAS code (make sure that both data sets are sorted in the same order (ascending or descending) by `sortvar`):

```
data newdata;  
  merge data1 data2 ;  
  by sortvar;  
  .  
  .   *More SAS statements;  
  .  
run;
```

The results are given in the next page.



The SAS code to accomplish this is

```

data mergesurv;
  merge outsurv sexkmsurv;
  by descending gender los;
  if gender=1 then survival1=survival;
  if gender=0 then survival0=survival;
  if gender=1 then coxsurv1=coxsurv;
  if gender=0 then coxsurv0=coxsurv;
  label survival1='KM survival (males)'
        survival0='KM survival (females)'
        coxsurv1='Cox survival (males)'
        coxsurv0='Cox survival (females)';
  drop coxsurv survival _cancel_ sdf_lcl sdf_ucl stratum;
run;

proc print data=mergesurv;
  title 'Merged survival data';
run;

```

The code above has the following results:

Merged survival data							
Obs	los	fail	gender	survival1	survival0	coxsurv1	coxsurv0
1	0	.	Men	1.00000	.	.	.
2	1	1	Men	0.99282	.	0.98184	.
3	1	1	Men	0.99282	.	0.98184	.
.
.
420	0	.	Women	.	1.00000	.	.
421	1	1	Women	.	0.98380	.	0.98773
422	1	1	Women	.	0.98380	.	0.98773
.
.
.

Now we can generate the Kaplan Meier and Cox plots and check the fit of the PH model. First we define the symbols and axis labels. Notice how we have decided to join the points in the first two symbols (that will eventually correspond to the Cox survival estimates (which are smooth), while we are producing a step function for the Kaplan-Meier curve.

```

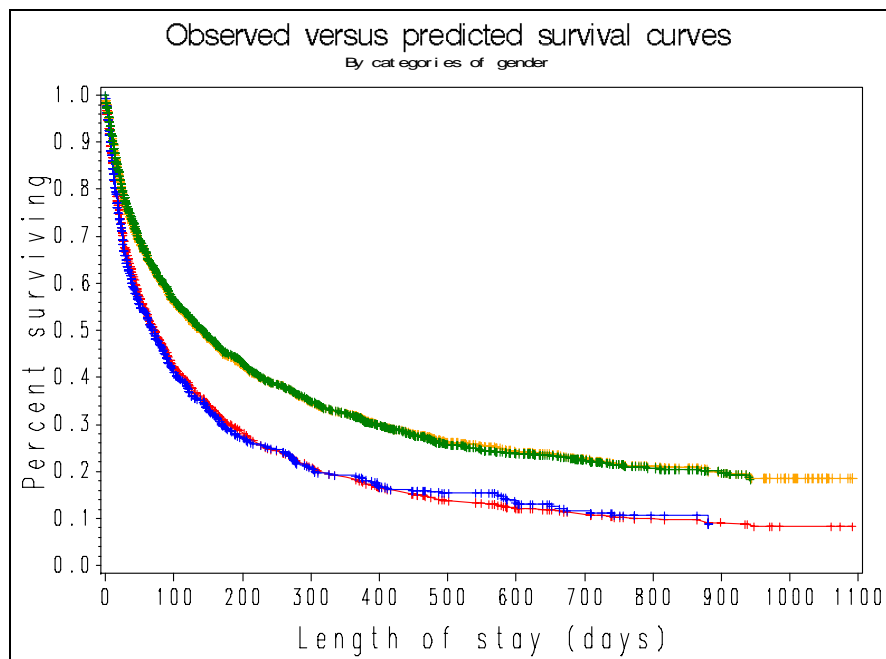
symbol1 c=red line=1 i=join value=plus;
symbol2 c=orange line=1 i=join value=plus;
symbol3 c=blue line=1 i=stepljs value=plus;
symbol4 c=green line=1 i=stepljs value=plus;
axis1 label=(angle=90 height=2.0 font='arial' 'Percent surviving' )
       value=(font='arial' height=1.5);
axis2 label=(height=2.0 font='arial' 'Length of stay (days)')
       value=(font='arial' height=1.5) minor=NONE;

```

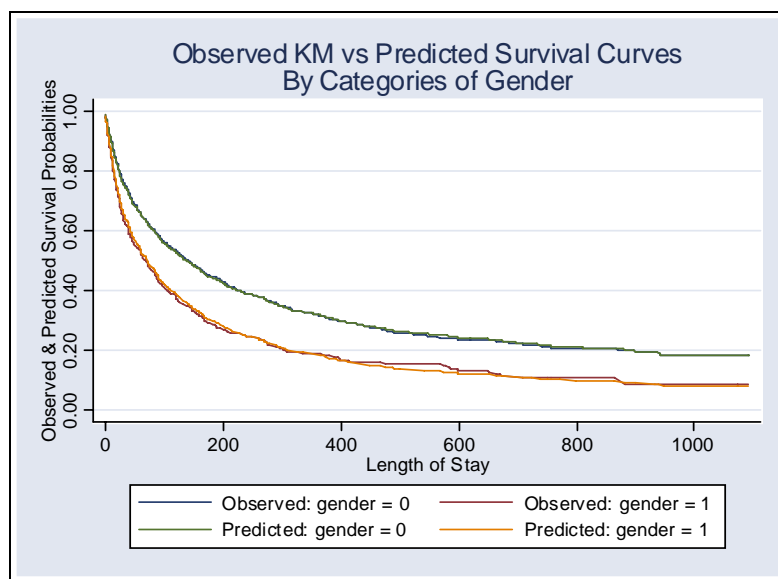

The GPLOT procedure in SAS is as follows:

```
proc gplot data=mergesurv;  
  plot coxsurv1*los=1  
      coxsurv0*los=2  
      survival1*los=3  
      survival0*los=4/overlay vaxis=axis1 haxis=axis2;  
  title 'Observed versus predicted survival curves';  
  title2 'By categories of gender';  
run;
```

resulting in the following plot:



Compare this to the STATA plot



Session 7: Time dependent covariates

We will input the Stanford heart transplant data set (stanford.dat).

The SAS code is as follows:

```
options ls=132;

data stanford;
  infile 'stanford.dat' missover;
  input patid birthm birthd birthy randm randd randy transm transd
        transy lastm lastd lasty
        dead priorsurg missnum hla_a2 miss misscore reject;
  birthy=birthy+1900;
  birthdt=mdy(birthm,birthd,birthy);
  randdt= mdy(randm, randd, randy );
  transdt=mdy(transm,transd,transy);
  lastdt= mdy(lastm, lastd, lasty );
  age=((randdt-birthdt)/365.25-48);
  wait=1+(transdt-randdt);
  if patid=38 then wait=wait-.1;
  if transdt=. then do;
    rx=0;
    start=0;
    stop=1+lastdt-randdt;
    status=dead;
    output;
  end;

  else do;
    rx=0;
    start=0;
    stop=wait;
    status=0;
    output;

    rx=1;
    start=stop;
    stop=1+lastdt-randdt;
    status=dead;
    output;
  end;
  format birthdt randdt transdt lastdt date9.;
  drop birthd birthm birthy transd transm transy randd randm
        randy lastm lastd lasty;
run;
```

A partial printout of the raw data is

	p	b	b	b	r	r	r	t	t	t	l	l	l	d	p	m	h		m	r	
O	a	i	i	i	a	a	a	r	r	r	a	a	a	e	r	i	l	m	i	r	
b	t	r	r	r	n	n	n	a	a	a	s	s	s	a	s	s	a		s	e	
s	d	m	d	y	m	d	y	m	d	y	m	d	y	d	g	m	2	s	r	e	
1	1	1	10	37	11	15	67	.	.	.	1	3	68	1	0	
2	2	3	2	16	1	2	68	.	.	.	1	7	68	1	0	
3	3	9	19	13	1	6	68	1	6	68	1	21	68	1	0	2	0	1.11	0	.	
4	4	12	23	27	3	28	68	5	2	68	5	5	68	1	0	3	0	1.66	0	.	
5	5	7	28	47	5	10	68	.	.	.	5	27	68	1	0	
6	6	11	8	13	6	13	68	.	.	.	6	15	68	1	0	
7	7	8	29	17	7	12	68	8	31	68	5	17	70	1	0	4	0	1.32	1	.	
8	8	3	27	23	8	1	68	.	.	.	9	9	68	1	0	
9	9	6	11	21	8	9	68	.	.	.	11	1	68	1	0	
10	10	2	9	26	8	11	68	8	22	68	10	7	68	1	0	2	0	0.61	1	.	
11	11	8	22	20	8	15	68	9	9	68	1	14	69	1	0	1	0	0.36	0	.	
12	12	7	9	15	9	17	68	.	.	.	9	24	68	1	0	
13	13	2	22	14	9	19	68	10	5	68	12	8	68	1	0	3	0	1.89	1	.	
14	14	9	16	14	9	20	68	10	26	68	7	7	72	1	0	1	0	0.87	1	.	
15	15	12	4	14	9	27	68	.	.	.	9	27	68	1	1	
16	16	5	16	19	10	26	68	11	22	68	8	29	69	1	0	2	0	1.12	1	.	
.
.
.

We analyze this code in sections. The first section simply inputs the data.

```
data stanford;
  infile 'stanfordch.dat' missover;
  input patid birthm birthd birthy randm randd randy transm transd
        transy lastm lastd lasty
        dead priorsurg misnum hla_a2 miss misscore reject;
```

The variables are a patient identification number, the birth day, month and year of the person, the day, month and year of randomization (i.e., study entry) the day, month and year of transplantation and the day month and year last seen at the clinic (dead or alive). There is an indicator about survival status (variable `dead`) which is 0 if the person is alive at the date last seen or 1 if the person is dead, as well as a binary indicator for prior surgery (1=Prior surgery, 0=No prior surgery), plus immune marker data (`hla_a2`), plus mismatch data (`miss`, `misscore` and `reject`).

Note also that we used the option `missover` to prevent SAS from searching for data in subsequent lines when data are not present in the line of input. This is *absolutely necessary* in order to read correctly some of the data lines that have missing data (although in this version of the data this is not necessary since all missing values are clearly marked with a period '.').

First we need to turn the day/month/year data into dates. We accomplish this as follows:

```
birthy=birthy+1900;
```

turning the year of birth (which is a two-digit number) into a year after 1900. Then we use the macro `mdy(month, day, year)` to turn the three components of the date into a single date.

```
birthdt=mdy(birthm,birthd,birthy);
randdt= mdy(randm, randd, randy );
transdt=mdy(transm,transd,transy);
lastdt= mdy(lastm, lastd, lasty );
```

We also create the variables for the wait time until a heart was found and the variable for age as a function of the randomization (study entry) date minus the birth date. To change it in years after the age of 48 (as in the original analysis) we divide by 365.25 (to take into account leap years) and subtract 48.

```
age=((randdt-birthdt)/365.25-48);
wait=1+(transdt-randdt);
```

Patient #38, received a transplant upon entry into the study, so wait time for that patient is 0. To overcome this problem, we subtract a small value (say 0.1) to its wait time (so the patient received a transplant not exactly at time zero).

```
if patid=38 then wait=wait-.1;
```

Now we need to figure out who got a transplant. If a person did not receive a heart then their transplant date would be missing. So one way to identify individuals without a transplant is to look for individuals with missing transplant date.

In addition, we will split the time of pre-transplant and post transplant. The information assigned to each individual pre and post-transplant will be as follows:

a. For individuals that did not receive a transplant

- i. Start of time is zero. Stop of time is the day last seen minus the study entry date plus one (i.e., the first day on study is day 1 not day 0).
- ii. The survival status is equivalent to the status determined by the variable `dead`.

The SAS code to accomplish this is as follows:

```
if transdt=. then do;
  rx=0;
  start=0;
  stop=1+lastdt-randdt;
  status=dead;
  output;
end;
```

Notice the command `output` in the previous code segment.

```
output;
```

This command outputs a line for that individual in the new data set. By including this line of code, SAS will not output by default but only where an output command exists.

b. For individuals that received a transplant

- i. Before transplant
 1. Start time is zero, stop (end of the interval) is the duration of the waiting time (contained in variable `wait`).
 2. Transplant status during this interval is zero
 3. Survival status (`dead`) is 0 (i.e., alive, otherwise the subject would not have received a transplant). Note that this is the source of the possible bias. The persons that received a transplant are alive *by definition* for some time until the transplant.

```
else do;
  rx=0;
  start=0;
  stop=wait;
  status=0;
  output;
```

ii. After transplant

1. Start time is equal to the stop time in the previous section.
Alternatively, we could have written

```
start=wait
```

2. Stop time is the difference between randomization and entry into the study plus one day.
3. The survival status is whatever is determined by the variable dead.

```
rx=1;  
start=stop;  
stop=1+lastdt-randdt;  
status=dead;  
output;  
end;
```

The final step is to assign formats and drop unnecessary variables.

```
format birthdt randdt transdt lastdt date9.;  
drop birthd birthm birthy transd transm transy randd randm  
randy lastm lastd lasty;
```

The new data is as follows:

```
proc print data=stanford;run;
```

	p	r	m																			
	i	o	i	b	t																	
	h	s	r	i	r	r	l															
	s	e	s	r	a	a	a															
	a	m	c	t	n	n	s	w	t	s	a											
	d	o	j	h	d	d	a	a	a	t	t											
	u	e	o	d																		
	n	r	e																			
	a	c	t																			
	r	s	r																			
	c	e	e																			
	j	o																				
1	1	1	0	.	.	.	10JAN1937	15NOV1967	.	03JAN1968	-17.1554	.	0	0	50	1						
2	2	1	0	.	.	.	02MAR1916	02JAN1968	.	07JAN1968	3.8357	.	0	0	6	1						
3	3	1	0	2	0	1.11	0	19SEP1913	06JAN1968	06JAN1968	21JAN1968	6.2971	1	0	0	1	0					
4	4	3	1	0	2	0	1.11	0	19SEP1913	06JAN1968	06JAN1968	21JAN1968	6.2971	1	1	1	16	1				
5	5	4	1	0	3	0	1.66	0	23DEC1927	28MAR1968	02MAY1968	05MAY1968	-7.7372	36	0	0	36	0				
6	6	4	1	0	3	0	1.66	0	23DEC1927	28MAR1968	02MAY1968	05MAY1968	-7.7372	36	1	36	39	1				
7	7	5	1	0	.	.	.	28JUL1947	10MAY1968	.	27MAY1968	-27.2142	.	0	0	18	1					
8	8	6	1	0	.	.	.	18NOV1913	13JUN1968	.	15JUN1968	6.5681	.	0	0	3	1					
9	9	7	1	0	4	0	1.32	1	29AUG1917	12JUL1968	31AUG1968	17MAY1970	2.8693	51	0	0	51	0				
10	10	7	1	0	4	0	1.32	1	29AUG1917	12JUL1968	31AUG1968	17MAY1970	2.8693	51	1	51	675	1				
11	11	8	1	0	.	.	.	27MAR1923	01AUG1968	.	09SEP1968	-2.6502	.	0	0	40	1					
12	12	9	1	0	.	.	.	11JUN1921	09AUG1968	.	01NOV1968	-0.8378	.	0	0	85	1					
13	13	10	1	0	2	0	0.61	1	09FEB1926	11AUG1968	22AUG1968	07OCT1968	-5.4976	12	0	0	12	0				
14	14	10	1	0	2	0	0.61	1	09FEB1926	11AUG1968	22AUG1968	07OCT1968	-5.4976	12	1	12	58	1				
15	15	11	1	0	1	0	0.36	0	22AUG1920	15AUG1968	09SEP1968	14JAN1969	-0.0192	26	0	0	26	0				
16	16	11	1	0	1	0	0.36	0	22AUG1920	15AUG1968	09SEP1968	14JAN1969	-0.0192	26	1	26	153	1				
17	17	12	1	0	.	.	.	09JUL1915	17SEP1968	.	24SEP1968	5.1937	.	0	0	8	1					
.
.
.

Now we are ready to perform the Cox regression. The simplest model involving the effect of transplant is

```
proc phreg data=stanford;  
  model (start stop)*status(0)=rx;  
  title 'Analysis of the Stanford heart transplant data';  
run;
```

Notice the new definition of the model

```
model (start stop)*status(0)=rx;
```

in terms of start and stop times (i.e., intervals of the form (t_{j-1}, t_j) instead of a single time).

Analysis of the Stanford heart transplant data

The PHREG Procedure

Model Information

Data Set	WORK.STANFORD
Dependent Variable	start
Dependent Variable	stop
Censoring Variable	status
Censoring Value(s)	0
Ties Handling	BRESLOW

Summary of the Number of Event and Censored Values

Total	Event	Censored	Percent Censored
172	75	97	56.40

Convergence Status

Convergence criterion (GCONV=1E-8) satisfied.

Model Fit Statistics

Criterion	Without Covariates	With Covariates
-2 LOG L	596.651	596.475
AIC	596.651	598.475
SBC	596.651	600.793

The PHREG Procedure						
Testing Global Null Hypothesis: BETA=0						
Test		Chi-Square	DF	Pr >	ChiSq	
Likelihood Ratio		0.1757	1	0.6751		
Score		0.1743	1	0.6763		
Wald		0.1742	1	0.6764		
Analysis of Maximum Likelihood Estimates						
Variable	DF	Parameter Estimate	Standard Error	Chi-Square	Pr > ChiSq	Hazard Ratio
rx	1	0.12567	0.30108	0.1742	0.6764	1.134

The interpretation of the model is that the overall impact of heart transplant on survival was not significant (if anything the hazard of death seems higher by 13.4% among patients having received transplants).

Naïve analysis of the Stanford heart transplant data

It is educational to try to perform the naïve analysis of the Stanford transplant data. For this analysis we are only interested in the final time from study entry to dead or last date seen alive and whether the person received a transplant or not.

We will input the previous data set and only keep the last line for each patient. This is accomplished as follows:

```
data naive;
  set stanford;
  by patid;
  if last.patid;
  keep patid stop rx status;
run;
```

A critical statement in the above data step is

```
by patid;
```

which acknowledges that the data are sorted by patient id (make sure this is the case or you'll get an error message). This also has the effect of creating two temporary SAS variables that keep track whether each observation is the first in the particular patient ID (this is called `first.patid` and is 1 if the observation is the first one within the specific patient ID and zero any other time) and `last.patid`, which is one and zero in the opposite order as `first.patid`.

The other critical statement is

```
if last.patid;
```

This is an SQL (structured query language – that is database language – statement that tells SAS to only keep those observations for which `last.patid` is one (i.e., the last observation from each patient).

We keep only the minimum amount of variables with the remaining statements. A printout of the data is as follows:

```
proc print data=naive;  
  title 'Data for naive analysis of Stanford transplant data';  
run;
```

Data for naive analysis of Stanford transplant data

Obs	patid	rx	stop	status
1	1	0	50	1
2	2	0	6	1
3	3	1	16	1
4	4	1	39	1
5	5	0	18	1
6	6	0	3	1
7	7	1	675	1
8	8	0	40	1
9	9	0	85	1
10	10	1	58	1
11	11	1	153	1
.
.
.
99	99	0	21	1
100	100	1	39	0
101	101	0	31	0
102	102	0	11	0
103	103	0	6	1

Compare this with the printout of the complete data set above. Note now that the variable `stop` is our time until death or censoring. Note also that there are now 103 lines of data (as many as the subjects) versus 172 in the previous analysis.

The analysis will be concluded by evoking the following command.

```
proc phreg data=naive;  
  model stop*status(0)=rx;  
  title 'Naive analysis of the Stanford heart transplant data';  
run;
```

The output from the naïve analysis is as follows:

```

Naive analysis of the Stanford heart transplant data

The PHREG Procedure

Model Information

Data Set                WORK.NAIVE
Dependent Variable      stop
Censoring Variable      status
Censoring Value(s)     0
Ties Handling           BRESLOW

Summary of the Number of Event and Censored Values

Total      Event      Censored      Percent
                                Censored

      103          75          28          27.18

Convergence Status

Convergence criterion (GCONV=1E-8) satisfied.

Model Fit Statistics

Criterion      Without      With
                Covariates  Covariates

-2 LOG L      596.649     570.924
AIC            596.649     572.924
SBC            596.649     575.242

Testing Global Null Hypothesis: BETA=0

Test      Chi-Square      DF      Pr > ChiSq

Likelihood Ratio      25.7251      1      <.0001
Score                  33.0137      1      <.0001
Wald                   29.1873      1      <.0001

Analysis of Maximum Likelihood Estimates

Variable DF      Parameter      Standard      Hazard
          Estimate      Error Chi-Square      Pr > ChiSq      Ratio
rx        1      -1.31832      0.24402      29.1873      <.0001      0.268

```

This shows the potentially dramatically erroneous analysis that can result from not adjusting properly for the bias inherent in this type of problem.

Session 9: Multiple failures

We will analyze the bladder data set (Wei et al., 1989). The data statement inputting the data into SAS is given below:

```
title 'Example 1: Multiple Failure Outcomes';

data bladder(keep=id tstart tstop status trt number size visit);
  retain id tstart 0;
  array tt t1-t4;
  infile cards missover;
  input trt time number size @27 t1 @31 t2 @35 t3 @39 t4;
  id + 1;
  tstart=0;
  do over tt;
    visit=_i_;
    if tt = . then do;
      tstop=time;
      status=0;
    end;
    else do;
      tstop=tt;
      status=1;
    end;
    output;
    tstart=tstop;
  end;
  if (tstart < time) then do;
    tstop= time;
    status=0;
    visit=5;
    output;
  end;
  datalines;
1      0      1      1
1      1      1      3
1      4      2      1
1      7      1      1
1     10      5      1
.      .      .      .
.      .      .      .
.      .      .      .
.      .      .      .
2     49      3      3
2     50      1      1
2     50      4      1      4      24      47
2     54      3      4
2     54      2      1      38
2     59      1      3
;
```

The data set is from a study in bladder cancer. The patients were followed for up to four recurrences (t1-t4). Some had less than four and some had none at all.

```
data bladder(keep=id tstart tstop status trt number size visit);
```

This snippet of code defines the first data set and tells SAS which variables will be ultimately kept in the resulting data set (regardless of any variables that will be created). This is very good style of programming.

```
retain id tstart 0;
```

Since we will turn data with a single line per subject into data with multiple lines per subject (as many as the recurrences plus, in some cases, the residual time of follow-up) we need to keep some variables the same as we generate line after line of code. The `retain` statement does just that. Also, `id` and `tstart` are defined (initially) equal to zero. It will be changed from one subject to another.

```
array tt t1-t4;  
infile cards missover;  
input trt time number size @27 t1 @31 t2 @35 t3 @39 t4;
```

The command `array` associates a bunch of variables in a variable list so that we can loop over them. It's the same as the command `foreach` in STATA. In this manner, we can loop over `t1`, `t2`, `t3` and `t4`.

```
id + 1;  
tstart=0;
```

For each subject, variable `id` is incremented by one and `tstart` is initialized to zero.

```
do over tt;  
  visit=_i_;  
  if tt = . then do;  
    tstop=time;  
    status=0;  
  end;  
  else do;  
    tstop=tt;  
    status=1;  
  end;  
  output;  
  tstart=tstop;  
end;
```

The above loop is done over `t1-t4`. If any of these is missing, then there is no recurrence (this was set to zero in STATA). In that case, `status=0` (censored observation for the recurrence in question) and `tstop` is set to follow-up time. When `t1`, `t2`, `t3` or `t4` are non-missing, `status` is set to 1 (recurrence event) and `tstop` to the non-missing time of recurrence. A new observation is output. After this has been done (and we are at the next line of output), `tstart` is set equal to the previous `tstop` and we loop again. In this manner four observations will be created *for all subjects*. Notice that, by using the `output` command, we create new observations (lines of data) whereas, if we had not, SAS would by default only create as many lines as the original data set.

```
if (tstart < time) then do;  
  tstop= time;  
  status=0;  
  visit=5;  
  output;  
end;
```

In a small number of cases, the last (`tstop`) observation (recall that after the output `tstart` is equal with the previous `tstop`), is still smaller than the total follow-up time. In that case there is

residual follow-up. We will create a fifth line for these subjects with `tstop` equal the total follow-up time.

A print out of the produced data set is as follows:

```
proc print data=bladder;
  by id;
  var tstart tstop status trt number size visit ;
run;
```

Example 1: Multiple Failure Outcomes								
----- id=1 -----								
Obs	tstart	tstop	status	trt	number	size	visit	
1	0	0	0	1	1	1	1	
2	0	0	0	1	1	1	2	
3	0	0	0	1	1	1	3	
4	0	0	0	1	1	1	4	
----- id=2 -----								
Obs	tstart	tstop	status	trt	number	size	visit	
5	0	1	0	1	1	3	1	
6	1	1	0	1	1	3	2	
7	1	1	0	1	1	3	3	
8	1	1	0	1	1	3	4	
----- id=3 -----								
Obs	tstart	tstop	status	trt	number	size	visit	
9	0	4	0	1	2	1	1	
10	4	4	0	1	2	1	2	
11	4	4	0	1	2	1	3	
12	4	4	0	1	2	1	4	
----- id=4 -----								
Obs	tstart	tstop	status	trt	number	size	visit	
13	0	7	0	1	1	1	1	
14	7	7	0	1	1	1	2	
15	7	7	0	1	1	1	3	
16	7	7	0	1	1	1	4	
----- id=5 -----								
Obs	tstart	tstop	status	trt	number	size	visit	
17	0	10	0	1	5	1	1	
18	10	10	0	1	5	1	2	
19	10	10	0	1	5	1	3	
20	10	10	0	1	5	1	4	
----- id=6 -----								
Obs	tstart	tstop	status	trt	number	size	visit	
21	0	6	1	1	4	1	1	
22	6	10	0	1	4	1	2	
23	10	10	0	1	4	1	3	
24	10	10	0	1	4	1	4	

Note that, for the first five subjects, there were no recurrences, so only the first line of data has a different `tstart` and `tstop` (in fact the first subject had zero follow-up and will be excluded from all analyses). By contrast, subject 6 has a recurrence at six months and is followed to 10 months. So that subject has two lines of data with different `tstart` and `tstop`.

The log output of this data step is as follows:

```
NOTE: The data set WORK.BLADDER has 356 observations and 8 variables.
NOTE: DATA statement used (Total process time):
      real time           0.03 seconds
      cpu time            0.04 seconds
```

There are four ways to analyze these data that we will show below. These are:

- The Andersen-Gill (conditional model)
- The marginal (Wei-Lin-Weisfeld or WLW model)
- The conditional Prentice-Williams-Peterson (PWP) model. This has two versions:
 - The time from start model
 - The gap-time model

All of these models have in common that they attempt to describe the risk set (i.e., which subjects are at risk for which type of failure, first, second, third or fourth) and estimating the variance.

The Andersen-Gill model

This model (Andersen & Gill, 1981), assumes that the failures are ordered and each subject is at risk for failure k only after he or she has had failure $k-1$. That is, you cannot be at risk for the second failure before you have experienced the first failure. While this is a reasonable assumption, the model also assumes that the failures are *independent* from each other, that is, the model does not account for clustering of failures within the same subject.

The analysis of the A-G model is given as follows:

```
title2 'Andersen-Gill Multiplicative Hazards Model';

proc phreg data=bladder;
  model (tstart, tstop) * status(0) = trt number size;
  where tstart < tstop;
run;
```

Notice that we specify that the model is to only be executed in cases where `tstart < tstop`. This will exclude all the superfluous observations generated by the code above. The output is as follows:

```
Example 1: Multiple Failure Outcomes                                37
Andersen-Gill Multiplicative Hazards Model                          17:49 Tuesday, January 22, 2008

                                The PHREG Procedure

                                Model Information

                                Data Set                               WORK.BLADDER
                                Dependent Variable                     tstart
                                Dependent Variable                     tstop
                                Censoring Variable                     status
                                Censoring Value(s)                     0
                                Ties Handling                           BRESLOW

                                Number of Observations Read             190
                                Number of Observations Used            190
```

Summary of the Number of Event and Censored Values

Total	Event	Censored	Percent Censored
190	112	78	41.05

Convergence Status

Convergence criterion (GCONV=1E-8) satisfied.

Model Fit Statistics

Criterion	Without Covariates	With Covariates
-2 LOG L	934.210	920.159
AIC	934.210	926.159
SBC	934.210	934.315

Testing Global Null Hypothesis: BETA=0

Test	Chi-Square	DF	Pr > ChiSq
Likelihood Ratio	14.0509	3	0.0028
Score	15.4173	3	0.0015
Wald	15.1736	3	0.0017

We note that, out of the 356 observations produced, only 190 are used in this analysis (including all 112 recurrence events).

Example 1: Multiple Failure Outcomes 38
 Andersen-Gill Multiplicative Hazards Model
 17:49 Tuesday, January 22, 2008

The PHREG Procedure

Analysis of Maximum Likelihood Estimates

Variable	DF	Parameter Estimate	Standard Error	Chi-Square	Pr > ChiSq	Hazard Ratio
trt	1	-0.40710	0.20007	4.1402	0.0419	0.666
number	1	0.16065	0.04801	11.1980	0.0008	1.174
size	1	-0.04009	0.07026	0.3256	0.5683	0.961

The result is that treatment reduces recurrences significantly. In addition, the number of tumors at baseline is predictive of subsequent recurrences.

The Wei-Lin-Weisfeld marginal model

The WLW model assumes that each tumor is a separate tumor type. Thus, the first tumor recurrence is a failure of type 1, the second of type 2 and so on. In addition, each subject is eligible for all recurrences (since they are simply failures of different types) *simultaneously*. While this is a mathematical approach (it is not logical in our setting of ordered failures) it makes sense in that, by setting the data in this manner, the approach allows construction of the correct matrices for calculation of the standard errors of the point estimates of the regression coefficients. The WLW approach uses a “sandwich estimator” of the variance of the type

$$V = I^{-1}U'UI^{-1} = D'D$$

where $I = \partial^2 \log L(\beta) / \partial \beta \partial \beta'$ is the usual information matrix and U is an $n \times p$ matrix of the score residuals. Matrix $D = UI^{-1}$ (is the matrix of leverage residuals – also called *dfbeta* by some

packages) with elements d_{ij} that are the differences in the estimate of $\hat{\beta}_j$ if observation i is removed from the dataset. The WLW data set is constructed from the original bladder data set as follows:

```
data bladder2;
  set bladder;
  if visit < 5;
  trt1= trt * (visit=1);
  trt2= trt * (visit=2);
  trt3= trt * (visit=3);
  trt4= trt * (visit=4);
  number1= number * (visit=1);
  number2= number * (visit=2);
  number3= number * (visit=3);
  number4= number * (visit=4);
  size1= size * (visit=1);
  size2= size * (visit=2);
  size3= size * (visit=3);
  size4= size * (visit=4);
run;
```

The above code simply excludes all observations past the fourth failure as subjects that have experienced all four failures cannot be at risk for anything else. The log output is as follows:

```
NOTE: There were 356 observations read from the data set WORK.BLADDER.
NOTE: The data set WORK.BLADDER2 has 344 observations and 20 variables.
NOTE: DATA statement used (Total process time):
      real time           0.00 seconds
      cpu time            0.00 seconds
```

Since there are 86 subjects with four observations each, there is a total of 344 observations in the bladder2 data set. In addition, the data step has created four interaction terms between trt, size and number with visit.

The WLW model is fit as follows:

```
title2 'Marginal Proportional Hazards Models';

proc phreg data=bladder2 covs(aggregate);
  model tstop*status(0)=trt1-trt4 number1-number4 size1-size4;
  strata visit;
  id ID;
  TREATMENT: test trt1, trt2, trt3, trt4/average e;
run;
```

Note some new features of the PHREG procedure. The WLW model essentially fits four separate models, one each for the first, second, third and fourth failure. Then it literally aggregates the matrices of the score residuals from the four models in a new matrix of the form

$$D'D = \begin{pmatrix} D_{11} & D_{12} & D_{13} & D_{14} \\ D_{21} & D_{22} & D_{23} & D_{24} \\ D_{31} & D_{32} & D_{33} & D_{34} \\ D_{41} & D_{42} & D_{43} & D_{44} \end{pmatrix}$$

where each $D_{ij} = I_i^{-1}G_i'G_jI_j^{-1}$, i.e., the information matrix for each model i and j and G is the $g \times p$ score residual matrix ($g=4$). The strata command performs a stratification over each visit (type of failure).

In addition, an overall test for treatment with an estimate of an average effect is requested. The analysis of the WLW model with stata is as follows:

Example 1: Multiple Failure Outcomes

Marginal Proportional Hazards Models

The PHREG Procedure

Model Information

Data Set WORK.BLADDER2
 Dependent Variable tstop
 Censoring Variable status
 Censoring Value(s) 0
 Ties Handling BRESLOW

Number of Observations Read 344
 Number of Observations Used 344

Summary of the Number of Event and Censored Values

Stratum	visit	Total	Event	Censored	Percent Censored
1	1	86	47	39	45.35
2	2	86	29	57	66.28
3	3	86	22	64	74.42
4	4	86	14	72	83.72
Total		344	112	232	67.44

Convergence Status

Convergence criterion (GCONV=1E-8) satisfied.

Model Fit Statistics

Criterion	Without Covariates	With Covariates
-2 LOG L	880.828	851.435
AIC	880.828	875.435
SBC	880.828	908.057

Testing Global Null Hypothesis: BETA=0

Test	Chi-Square	DF	Pr > ChiSq
Likelihood Ratio	29.3932	12	0.0034
Score (Model-Based)	33.0747	12	0.0009
Score (Sandwich)	17.7990	12	0.1219
Wald (Model-Based)	31.0544	12	0.0019

Example 1: Multiple Failure Outcomes

Marginal Proportional Hazards Models

The PHREG Procedure

Testing Global Null Hypothesis: BETA=0

Test	Chi-Square	DF	Pr > ChiSq
Wald (Sandwich)	34.8311	12	0.0005

Analysis of Maximum Likelihood Estimates

Variable	DF	Parameter Estimate	Standard Error	StdErr Ratio	Chi-Square	Pr > ChiSq	Hazard Ratio
trt1	1	-0.51762	0.30750	0.974	2.8336	0.0923	0.596
trt2	1	-0.61944	0.36391	0.926	2.8975	0.0887	0.538
trt3	1	-0.69988	0.41516	0.903	2.8419	0.0918	0.497
trt4	1	-0.65079	0.48971	0.848	1.7661	0.1839	0.522
number1	1	0.23599	0.07208	0.947	10.7204	0.0011	1.266
number2	1	0.13756	0.08690	0.946	2.5059	0.1134	1.147
number3	1	0.16984	0.10356	0.984	2.6896	0.1010	1.185
number4	1	0.32880	0.11382	0.909	8.3453	0.0039	1.389
size1	1	0.06789	0.08529	0.842	0.6336	0.4260	1.070
size2	1	-0.07612	0.11812	0.881	0.4153	0.5193	0.927
size3	1	-0.21131	0.17198	0.943	1.5097	0.2192	0.810
size4	1	-0.20317	0.19106	0.830	1.1308	0.2876	0.816

Example 1: Multiple Failure Outcomes

Marginal Proportional Hazards Models

The PHREG Procedure

Linear Coefficients for Test TREATMENT

Parameter	Row1	Row2	Row3	Row4	Average Effect
trt1	1	0	0	0	0.67684
trt2	0	1	0	0	0.25723
trt3	0	0	1	0	-0.07547
trt4	0	0	0	1	0.14140
number1	0	0	0	0	0.00000
number2	0	0	0	0	0.00000
number3	0	0	0	0	0.00000
number4	0	0	0	0	0.00000
size1	0	0	0	0	0.00000
size2	0	0	0	0	0.00000
size3	0	0	0	0	0.00000
size4	0	0	0	0	0.00000
CONSTANT	0	0	0	0	0.00000

Test TREATMENT Results

Wald Chi-Square	DF	Pr > ChiSq
3.9668	4	0.4105

Average Effect for Test TREATMENT

Estimate	Standard Error	z-Score	Pr > z
-0.5489	0.2853	-1.9240	0.0543

The overall average treatment effect has $\hat{\beta} = -0.5489$, which corresponds to a hazard rate of 0.578, that is, the treatment is associated with about half of the risk for recurrence compared to the control.

To understand the workings of this model, fit a separate model for the first failure. This is done as follows:

```
proc phreg data=bladder2;
  model tstop*status(0)=trt1 number1 size1;
  where visit=1;
  title3 'Model for first visit';
run;
```

The relevant output is as follows:

Example 1: Multiple Failure Outcomes						
Marginal Proportional Hazards Models						
Model for first visit						
The PHREG Procedure						
Analysis of Maximum Likelihood Estimates						
Variable	DF	Parameter Estimate	Standard Error	Chi-Square	Pr > ChiSq	Hazard Ratio
trt1	1	-0.51757	0.31576	2.6868	0.1012	0.596
number1	1	0.23605	0.07607	9.6287	0.0019	1.266
size1	1	0.06790	0.10125	0.4498	0.5024	1.070

We notice that the point estimate for $\hat{\beta} = -0.518$ is almost identical to the point estimate produced above for `trt1`. The standard error estimate is slightly different because, in the previous case, all standard errors were produced by a sandwich estimator, while the standard error in this case follows the naïve approach.

The Prentice-Williams-Peterson model

There are two types of PWP models: The gap time model and the total time model. In both cases, the setup of the data set is identical to the A-G model, with the exception that time of observation past the last failure is not considered (i.e., once the fourth failure has occurred the patient is not considered further).

a) The gap time model

In this case, the PWP approach is a version of the A-G conditional model where each subject is considered at risk for each failure *conditional* on having experienced the previous failure. The differentiation of the model is in the fact that the variance estimation proceeds by a stratified analysis according to each failure (i.e., just as in the WLW model, the first failure is considered as failure of type 1, the second of type 2 and so on). In the gap-time model the length of the interval (i.e., $(t_{start}, t_{stop}]$) is considered, where the start of the interval, just as in the A-G case, is past the occurrence of the previous failure (i.e., the subject cannot be eligible to experience a subsequent failure prior to having experienced all previous failures).

The setup of the data are similar to the A-G model, but the clock starts from the occurrence of the previous model.

We will define variable $gap = t_{stop} - t_{start}$ and we will set up the data as follows:

```
* Fitting the models of Prentice, Williams and Peterson;
data bladder3(drop=lstatus);
  retain lstatus;
  set bladder2;
  by id;
  if first.id then lstatus=1;
  if (status=0 and lstatus=0) then delete;
  lstatus=status;
  gaptime=tstop-tstart;
run;
```

In this formulation, all observations following a censored observation, other than the first observation when `lstatus` is set to 1, are discarded from the model. This in effect discards all duplicate censored visits but keeps the last censored visit (since the subject would be at risk for a subsequent failure then). A print out of the data is as follows:

```
title2 'PWP Total Time Model with Noncommon Effects';
proc print data=bladder3;
  by id;
  var tstart tstop status trt number size visit gaptime ;
run;
```

Example 1: Multiple Failure Outcomes PWP Total Time Model with Noncommon Effects									
----- id=1 -----									
Obs	tstart	tstop	status	trt	number	size	visit	gaptime	
1	0	0	0	1	1	1	1	0	
----- id=2 -----									
Obs	tstart	tstop	status	trt	number	size	visit	gaptime	
2	0	1	0	1	1	3	1	1	
----- id=3 -----									
Obs	tstart	tstop	status	trt	number	size	visit	gaptime	
3	0	4	0	1	2	1	1	4	
----- id=4 -----									
Obs	tstart	tstop	status	trt	number	size	visit	gaptime	
4	0	7	0	1	1	1	1	7	
----- id=5 -----									
Obs	tstart	tstop	status	trt	number	size	visit	gaptime	
5	0	10	0	1	5	1	1	10	
----- id=6 -----									
Obs	tstart	tstop	status	trt	number	size	visit	gaptime	
6	0	6	1	1	4	1	1	6	
7	6	10	0	1	4	1	2	4	
----- id=7 -----									
Obs	tstart	tstop	status	trt	number	size	visit	gaptime	
8	0	14	0	1	1	1	1	14	

The analysis proceeds as in the case of single-observation per subject data.

```

title2 'PWP Gap Time Model with Common Effects';
proc phreg data=bladder3;
  model gaptime * status(0) = trt number size;
  strata visit;
run;

```

Example 1: Multiple Failure Outcomes

PWP Gap Time Model with Common Effects

The PHREG Procedure

Model Information

```

Data Set                WORK.BLADDER3
Dependent Variable      gaptime
Censoring Variable      status
Censoring Value(s)     0
Ties Handling           BRESLOW

```

```

Number of Observations Read    184
Number of Observations Used    184

```

Summary of the Number of Event and Censored Values

Stratum	visit	Total	Event	Censored	Percent Censored
1	1	86	47	39	45.35
2	2	47	29	18	38.30
3	3	29	22	7	24.14
4	4	22	14	8	36.36
Total		184	112	72	39.13

Convergence Status

Convergence criterion (GCONV=1E-8) satisfied.

Model Fit Statistics

Criterion	Without Covariates	With Covariates
-2 LOG L	735.076	726.320
AIC	735.076	732.320
SBC	735.076	740.476

Testing Global Null Hypothesis: BETA=0

Test	Chi-Square	DF	Pr > ChiSq
Likelihood Ratio	8.7559	3	0.0327
Score	9.5977	3	0.0223
Wald	9.4570	3	0.0238

Example 1: Multiple Failure Outcomes

PWP Gap Time Model with Common Effects

The PHREG Procedure

Analysis of Maximum Likelihood Estimates

Variable	DF	Parameter Estimate	Standard Error	Chi-Square	Pr > ChiSq	Hazard Ratio
trt	1	-0.26952	0.20766	1.6845	0.1943	0.764
number	1	0.15353	0.05211	8.6823	0.0032	1.166
size	1	0.00684	0.07001	0.0095	0.9222	1.007

Note that this is equivalent to a model with $t_{start}=0$ and $t_{stop}=gaptime$.

b) The total time conditional model

In this model, t_{start} is set to zero, i.e., the time at risk for each failure is the total time from entry until the occurrence of the failure. The analysis of the PWP model proceeds as follows:

```

title2 'PWP Total Time Model with Common Effects';
proc phreg data=bladder3;
  model tstop * status(0) = trt number size;
  strata visit;
run;

```

The analysis by the Cox model is given by the following output:

Example 1: Multiple Failure Outcomes

PWP Total Time Model with Common Effects

The PHREG Procedure

Model Information

Data Set	WORK.BLADDER3
Dependent Variable	tstop
Censoring Variable	status
Censoring Value(s)	0
Ties Handling	BRESLOW

Number of Observations Read	184
Number of Observations Used	184

Summary of the Number of Event and Censored Values

Stratum	visit	Total	Event	Censored	Percent Censored
1	1	86	47	39	45.35
2	2	47	29	18	38.30
3	3	29	22	7	24.14
4	4	22	14	8	36.36

Total		184	112	72	39.13

Convergence Status

Convergence criterion (GCONV=1E-8) satisfied.

Model Fit Statistics

Criterion	Without Covariates	With Covariates
-2 LOG L	743.098	734.347
AIC	743.098	740.347
SBC	743.098	748.502

Testing Global Null Hypothesis: BETA=0

Test	Chi-Square	DF	Pr > ChiSq
Likelihood Ratio	8.7512	3	0.0328
Score	8.8795	3	0.0309
Wald	8.7957	3	0.0321

Example 1: Multiple Failure Outcomes

PWP Total Time Model with Common Effects

The PHREG Procedure

Analysis of Maximum Likelihood Estimates

Variable	DF	Parameter Estimate	Standard Error	Chi-Square	Pr > ChiSq	Hazard Ratio
trt	1	-0.48972	0.20925	5.4775	0.0193	0.613
number	1	0.11027	0.05105	4.6659	0.0308	1.117
size	1	-0.03773	0.06754	0.3121	0.5764	0.963