Analysis of survival data / Inger Persson
Updated Nov 21 2025

SAS examples

Content

1. The DATA step

1.1 The INPUT statement (enter data directly in SAS)

```
data patients;
input patientno diagn_day treatment $ trt_start_day effect_day censored
death hospital & $ 10.;
/* $ sign after the variable name denotes a character variable */
/* & sign after the variable name tells SAS to use two whitespace
 characters (instead of the default one) to signal end of character */
/* The length of the variable is chosen automatically, but for variables
with whitespace characters included the length might need to be set
manually - here by adding “10.” After the variable name */
datalines;
1 5 A 9 16 0 0 Hospital 1
2 8 B 8 14 0 0 Hospital 1
3 12 B 14 18 0 0 Hospital 2
4 16 B 17 21 0 0 Hospital 1
5 17 A 20 23 0 0 Hospital 2
6 19 A 20 27 0 0 Hospital 2
7 23 B 24 30 0 0 Hospital 1
8 24 A 28 35 1 1 Hospital 2
9 30 A 32 40 1 0 Hospital 1
10 32 B 33 37 1 1 Hospital 2
;
run;

/* Variable lists can be used on the input statement. For example, “var1-
var4” can be used instead of typing “var1 var2 var3 var4”. */

/* The two examples below give the same result */
input name treatment dose1 dose2 dose3 dose4 dose5 dose6;
input name treatment dose1-dose6;
```

Note: If SAS needs to read an additional line to input all the variables referenced in the input statement it prints the following message on the log: ”Note: SAS went to a new line when INPUT statement reached past the end of a line.” If you see this note, check that you haven’t included one more variable than there are columns.

1.2 The INFILE statement (read .txt files into SAS)

First view the file in a simple text editor (such as Notepad/Wordpad) to know which variables are included, if they contain numbers or text, and at which lines the data begins and ends.

To find the location of the file in SAS Studio: Find the file in the folder you uploaded it to, under Server Files and Folders - Files (Home). Right click and choose Properties. Copy the path from Location (Ctrl/Command+C) and paste it in your code (Ctrl/Command+V), with quotation marks around it (single or double). An example of a path is used in the code below, replace that by your own unique path.

```
data ministers;
infile '/home/username/SAS introduction/Prime ministers w headings.txt'
   dlm='09'x                     /* Tab delimiter */
   firstobs=2          /* First row to be imported */
   obs = 10                      /* Last row number to be imported */
   encoding='ISO-8859-1';  /* Support special characters */
input year minister & $20.;      /* Set the length of column 2 */
run;
```

1.3 The SET statement (create a dataset or make changes in an existing one)

```
data patients2;                    /* Create a new dataset */
set patients;                      /* based on an existing dataset */
time_to_effect = effect_day-trt_start_day;     /* Create a new variable */
run;

data patients;                     /* Make changes in a dataset */
set patients;
time_to_effect = effect_day-trt_start_day;     /* Create a new variable */
run;
```

1.4 The WHERE statement

```
data trtA;
set patients;
where treatment = 'A';    /* keep only treatment A patients */
run;

data trtA;
set patients;
where diagn_day <= 21;    /* keep only pts diagnosed the first 3 weeks */
run;
```

1.5 The OUTPUT and DELETE statements

```
data patients2;
set patients;
if treatment = 'A' then output;
run;

/* or */
data patients2;
set patients;
if treatment ^= 'A' then delete;
run;
```

1.6 The IF, THEN and ELSE statements

```
data patients;
set patients;
if treatment='A' then trt_name = 'Treatment A';
   else trt_name = 'Treatment B';
if missing(diagn_day) then missing_diagn_day = 1;
   else missing_diagn_day = 0;
run;
```

1.7 The KEEP and DROP statements

```
data patients_subset1;
set patients;
keep patientno treatment;        /* keep only selected variables */
run;

/* or */
data patients_subset1;
set patients(keep=patientno treatment);
run;


data patients_subset2;
set patients;
drop treatment;       /* drop selected variables */
run;

/* or */
data patients_subset2;
set patients(drop=treatment);
run;
```

1.8 The MERGE statement

```
data patients3;
merge patients_subset1 patients_subset2;
by patientno;
run;
/* Both datasets must first be sorted by the by variable(s) */

data patients4;
merge patients_subset1(IN=IN_set1) patients_subset2(IN=IN_set2);
by patientno;
if IN_set1 and IN_set2;  /*Only individuals with data in both study parts*/
run;
```

1.9 Convert character variables to numeric and vice versa

```
/* Convert numeric to character */
data patients; set patients;
patientno_character=put(patientno,4.0);
run;


/* Convert character to numeric */
data patients; set patients;
patientno_numeric=input(patientno_character,best4.);
run;
```

1.10 Dummy coding

```
/* Alternative 1 */
data patients; set patients;
treatmentA=(treatment='A');
treatmentB=(treatment='B');
run;

/* Alternative 2 */
data patients; set patients;
if treatment='A' then treatmentA=1; else treatmentA = 0;
if treatment='B' then treatmentB=1; else treatmentB = 0;
run;
```

1.11 Impute missing values with the RETAIN statement

```
data mydata; set mydata;
retain impute_value .;            /* Start with a missing value */
if missing(myvariable)
   then myvariable = impute_value;
else impute_value = myvariable;
run;
/* If myvariable is not missing, impute_value will change to the value of
myvariable and this value will be retained for lower rows until myvariable
takes on a new value */
```

1.12 Arithmetic operators

Code	Operation
+	Addition
-	Subtraction
*	Multiplication
/	Division
sqrt()	Square root
**	Exponentiation
log()	Logarithm (base e)
Log10()	Logarithm (base 10)
Log10()	Logarithm (base 10)

1.13 Logical operators

Code	Operation
`= or` *`eq`*	equal to
`^=`	not equal to
`> or` *`gt`*	greater than
`>=`	greater than or equal to
`< or` *`lt`*	less than
`<=`	less than or equal to

1.14 Boolean operators (true or false)

Code	Operation
`& or` *`and`*	and
`| or` *`or`*	or
`^ or` *`not`*	negation

1.15 Other operators

Code	Operation
`><`	minimum
`<>`	Maximum
`In`	The in operator lets you test for equality to any of several constant values. “x in(1,2,3)” is the same as “x=1 or x=2 or x=3”

2. The PROC steps

2.1 Proc IMPORT

```
/* Import Excel file */
proc import
   dbms=xlsx          /* The type of file to be imported */
   datafile='/home/username/SAS introduction/Prime ministers.xlsx'
   out=survey         /* Give a name to the imported dataset */
   replace;           /* Replace any previous versions */
run;

/* Import CSV file */
proc import
   dbms=csv           /* The type of file to be imported */
   datafile='/home/username/SAS introduction/Prime ministers.csv'
   out=survey         /* Give a name to the imported dataset */
   replace;           /* Replace any previous versions */
   delimiter=';';     /* Default: comma separated */
run;
```

2.2 Proc SORT

```
proc sort data=patients;
by patientno;         /* Sort by patientno */
run;

proc sort data=patients;
by descending patientno;    /* Sort by patientno in reverse order */
run;


/* Sort by two variables */
proc sort data=patients;
by treatment time_to_effect; /*First by treatment, then by time_to_effect*/
run;
```

2.3 Proc FREQ

```
proc freq data=patients;
table censored*trt;
run;

proc freq data=patients;
table censored*trt / norow nocol nopercent;  /* Exclude percentages */
run;
```

2.4 Proc MEANS (summary statistics)

```
proc means data=patients;  /* Get a summary of all variables */
run;

proc means data=patients N MEAN STD;    /* Choose specific statistics */
class treatment;    /* Present statistics by a grouping variable*/
var diagn_day time_to_effect;   /* Choose variables to present stats for */
run;

proc means data=patients;
by treatment;    /* by (instead of class) presents two separate outputs */
run;
```

2.5 Proc UNIVARIATE (summarize, visualise, and analyse)

```
proc univariate data=patients;
class treatment;
var time_to_effect;
histogram time_to_effect /endpoints = 2 4 6 8 10;
run;
```

2.6 Proc FORMAT

```
/* Format for numerical variables */
proc format;
value cens /* Creating a format named “cens” */
1='Not censored'
0='Censored'
other = 'Unknown';
run;
/* It is always wise to include 'other', which will capture all
alternatives not covered by the values above */

/* Format for character variables */
proc format;
value $trt
'A' ='Treatment A'
'B' ='Treatment B'
other = 'Unknown';
run;


/* Example of how to use a format */
proc means data=patients;
class censored;
format censored cens. Treatment $cens.;
run;
```

2.7 Proc SGPLOT

Some examples of how to use proc sgplot are included in Chapter 4. Fore more information, use the SAS online documentation.
https://documentation.sas.com/doc/en/pgmsascdc/9.4_3.5/grstatproc/n0yjdd910dh59zn1toodgupaj4v9.htm

3. Other useful commands

3.1 The LIBNAME statement (create/open permanent datasets)

```
/* First assign a name to the directory/library you want to use (here
"intro") with the libname statement. Maximum length of the name: 8 chrs.
Note that the run; command is not needed here. */

libname intro '/home/username/SAS introduction/Data';

/*Then use the name of that directory/library as a suffix to your dataset*/

data intro.ministers; /* Create a permanent dataset named ministers */
set ministers;        /* based on the working data ministers */
run;

/* Create a working copy based on a permanent dataset */
data ministers;
set intro.ministers;
run;
```

3.2 The Output Delivery System (ODS)

SAS produces a lot of output through its Output Delivery System (ods). Most of it is presented in the output window, and can be saved in datasets.

```
/* Turn on the ods trace before running a proc to see which output can be
produced from that proc */
ods trace on;

/* Run the proc */
proc univariate data=patients;
class treatment;
run;


/* Check the log window to see which output can be produced (only a part of
the output is included here) */
```

```
Output Added:
-------------
Name:       BasicMeasures
Label:      Basic Measures of Location and Variability
Template:   base.univariate.Measures
Path:       Univariate.patientno.'0'n.BasicMeasures
-------------

Output Added:
-------------
Name:       TestsForLocation
Label:      Tests For Location
Template:   base.univariate.Location
Path:       Univariate.patientno.'0'n.TestsForLocation
-------------

Output Added:
-------------
Name:       Quantiles
Label:      Quantiles
Template:   base.univariate.Quantiles
Path:       Univariate.patientno.'0'n.Quantiles
-------------
```

```
/* Turn off the ods trace */
ods trace off;
```

```
/* Run the proc again, with the ods output statement*/
proc univariate data=patients;
class treatment;
ods output BasicMeasures=patients_basics Quantiles=patients_quantiles;
run;
/* Output BasicMeasures is saved in dataset patients_basics
and Quantiles is saved in dataset patients_quantiles */
```

3.2.1 Save SAS output in a text document (.rtf) document using ODS

The Output Delivery System (ODS) can be used to send procedure output directly to several types of external files. Output sent to files with the extension .rtf can open as Word documents. Procedure output is contained in Word formatted tables that can be easily copied into homework documents and further edited.

```
options orientation=landscape    /* Choose landscape or portrait */
   rightmargin = 0.5 cm          /* Choose margins to fit your output */
   leftmargin = 1.5 cm;

title 'Title of my choice ';

ods rtf file ="/home/username/folder name/document name.rtf";

proc univariate data=patients;
class treatment;
histogram time_to_effect;
run;

proc freq data=patients;
table censored*trt;
run;

ods rtf close;

title; /* Clear title */
```

All the output from both proc univariate and proc freq above will be included in the same rtf file.

- There can be multiple procedures between the opening and closing of the .rtf file.
- If the specified .rtf file already exists, SAS will prompt for permission to overwrite it.
- The default table style of the output can be changed. Try:
  ```
  ods rtf file ="/home/username/folder name/document name.rtf";
  style=minimal;
  ```
- Other styles to try: Journal, Analysis, Statistical

3.3 Delete SAS work datasets

```
proc datasets lib=work nolist memtype=data kill;
run;
quit;
```

4. Survival analysis examples

4.1 Censoring plot

```
ods graphics / reset attrpriority=none;
/* Need to override the built-in attributes in proc sgplot */

proc sgplot data=patients;
styleattrs /* Without statement above these changes would have no effect */
   datasymbols=(plus circle)
   datacolors=(black red);
scatter x=time_to_effect y=treatment /group=censored jitter;
xaxis label= "Time to effect (days from treatment start)";
yaxis integer label="Treatment group" min=-0.2 max=1.2;
/*"Integer" ensures that no tick marks are shown between the group values*/
/*"min" and “max” adds some space below and above the symbols */
format censored cens.;
label censored="00"x; /* Remove the label “censored” from x axis legend */
run;
```

4.2 Proc LIFETEST K-M survival and plot

```
/* proc lifetest creates a survival plot, K-M by default */

proc lifetest data=patients;
time time_to_effect*censored(1);  /*Time var.*event var.(censoring value)*/
run;


/* Survival by groups (strata) */
/* Log-rank and Gehan’s (=Wilcoxon) tests are presented by default */

proc lifetest data=patients;
time time_to_effect*censored(1);
strata treatment;     /* Grouping variable */
run;

proc lifetest data=patients;
time time_to_effect*censored(1);
strata diagn_day(0-10 11-20 21);  /*Strata can be defined within the
procedure */
run;

proc lifetest data=patients;
time time_to_effect*censored(1);
strata treatment/test=tarone; /* Tarone and ware's weights */
run;


/* Stratified test */

proc lifetest data=survival;
time time_to_effect*censored(1);
strata diagn_day(0-10 11-20 21) / group=treatment;
run;
```

```
/* Store survival estimates in a dataset with the "outsurv" option */
/* The option "stderr" produces standard errors */

proc lifetest data=patients outsurv=estimates stderr;
time time_to_effect*censored(1);
strata treatment;     /* Grouping variable */
run;


/* Store the plot data to make manual adjustments. */

proc lifetest data=patients;
time time_to_effect*censored(1);
strata treatment;     /* Grouping variable */
ods output SurvivalPlot=survplot;
run;

ods graphics / attrpriority=none;
/* Need to override the built-in attributes in proc sgplot */

proc sgplot data=survplot;
styleattrs datacontrastcolors=(orange purple blue)
   datalinepatterns=(solid shortdash mediumdash);
step x=time_to_effect y=censored / group=stratum markerattrs=(symbol=plus);
title "Kaplan-Meier survival estimates";
xaxis label="Time to treatment effect (days)";
yaxis label="Survival Probability";
run;
```

See SAS online documentation for a User's guide for customization of the Kaplan-Meier plot: https://support.sas.com/documentation/onlinedoc/stat/151/kaplan.pdf

4.3 Proc LIFETEST Nelson-Aalen survival and plot

```
/* Nelson-Aalen cumulative hazard */

proc lifetest data=patients nelson; /* 'nelson' option */
time time_to_effect*censored(1);
strata treatment;
run;

/* Survival estimated by the Nelson-Aalen method. */

proc lifetest data=patients nelson method=breslow;
time time_to_effect*censored(1);
strata treatment;
run;
/* 'method=Breslow' gives the Breslow estimator, which is the exponenti-
ation of the negative Nelson-Aalen estimator of the cumulative hazard
function - i.e. the survival estimated by the Nelson-Aalen method. */
/* Alternatively, store the cumhaz (as below) and calculate the survival as
exp(-CumHaz) */


/* Plot Nelson-Aalen cumulative hazard*/

/* Store the data and create a plot "manually". The 'outsurv' option only
stores the survival, not the cumulative hazard. To store the cumulative
hazard, the ods statement can be used (see chapter 3.2) */
```

```
proc lifetest data=patients method=breslow nelson;
time time_to_effect*censored(1);
strata treatment;
ods output BreslowEstimates=NAestimates;
run;

/* In order to plot the cumhaz, we need to replace the missing values of
the cumulative hazard (for censored observations) with previous existing
value */

data figure; set NAestimates;
 retain impute_value .;
 if missing(cumhaz)
   then cumhaz = impute_value;
   else impute_value = cumhaz;
run;

ods graphics / attrpriority=none;
/* Need to override the built-in attributes in proc sgplot */

proc sgplot data=figure;
styleattrs datacontrastcolors=(orange purple blue)
   datalinepatterns=(solid shortdash mediumdash);
step x=time_to_effect y=cumhaz / group=treatment;
title "Nelson-Aalen cumulative hazard";
xaxis label="Time to treatment effect (days)";
yaxis label="Cumulative hazard" grid;
format treatment $trt.;
label treatment="00"x; /*Remove the word “treatment” from x-axis label*/
run;
```

4.4 Proc LIFETEST Confidence intervals

4.4.1 Pointwise confidence intervals

```
/* Linear */

proc lifetest data=patients
   plots=survival(cl strata=panel)
   conftype=LINEAR;   /* default type is LOGLOG, i.e., log-transformed */
time time_to_effect*censored(1);
strata treatment;     /* By group */
label time_to_effect="Time to treatment effect (days)";
format treatment $trt.;
run;
/* If you exclude “strata=panel” the groups will be in the same plot, with
confidence intervals overlapping */

/* Log-transformed */
   conftype=LOGLOG    /* Can be excluded since LOGLOG is the default */

/* Arcsine-square root transformed */
   conftype= ASINSQRT

/* If you want to make adjustments to the plot, e.g. titles, not connecting
the confidence intervals over time, etc, you have to store the data and
create a plot “manually” */
```

```
proc lifetest data=patients
   plots=none
   outsurv=survCIlog;   /* Saves the results in dataset survCIlog */
time time_to_effect*censored(1);
strata treatment;
run;

/* Impute survival estimates for the censored times */

data survCI2; set survCIlog;
retain impute_value .;
if missing(survival)
/* Only S(t) for censored times, not the CI, to make
clear that the intervals are pointwise at event times */
   then survival = impute_value;
else impute_value = survival;
run;

/* Plot survival curve including censored observations at the end*/

ods graphics / reset attrpriority=none;
/* Need to override the built-in attributes in proc sgplot */

proc sgplot data=survCI2;
styleattrs datacontrastcolors=(orange purple blue)
   datalinepatterns=(solid);
step y=survival x= time_to_effect /Yerrorlower=SDF_LCL Yerrorupper=SDF_UCL
   group=treatment /* Remove group statement if there is only one group */
   markers markerattrs=(symbol=plus);
title "Pointwise 95% log-transformed confidence intervals for Kaplan-Meier
survival";
xaxis label="Time to treatment effect (days)";
yaxis label="Estimated survival (probability of not reaching an effect)";
label treatment="Treatment";
format treatment $trt.;
run;
/* Add the statement "by treatment;" to produce three separate graphs if
wanted */
```

4.4.2 Confidence bands

```
/* EQUAL PROBABILITY CONFIDENCE BANDS */

/* Arcsine-square root transformed */
proc lifetest data=patients plots=survival(cb=ep strata=panel)
    conftype=ASINSQRT;
time time_to_effect*censored(1);
strata treatment;
label time_to_effect="Time to treatment effect (days)";
format treatment $trt.;
run;
/* If you exclude "strata=panel" the groups will be in the same plot, with
confidence bands overlapping */


/* If you want to make adjustments to the plot, e.g. titles, not connecting
the confidence intervals over time, etc, you have to store the data and
create a plot "manually" */
```

```
proc lifetest data=patients plots=none confband=EP conftype=ASINSQRT
     bandmintime=2.5 bandmaxtime=7  /* tL and tU */
     outsurv=survEP stderr;  /* Saves the results (incl std) in dataset
survEP */
time time_to_effect*censored(1);
strata treatment;
run;

/* The survEP dataset will still contain the full set of data, which means
that you have to constraint the graph too */
/* Plot survival curve with EP confidence bands */

proc sgplot data=survEP;
where 2.5<=time_to_effect<=7;
step x=time_to_effect y=survival / lineattrs=(color=black pattern=1);
step x=time_to_effect y= EP_LCL / lineattrs=(color=red pattern=3);
step x=time_to_effect y= EP_UCL / lineattrs=(color=red pattern=3);
by treatment;
title "95% Arcsine-square root transformed equal probability bands";
xaxis label ="Time to treatment effect (days)";
yaxis label="Estimated Survival Function";
format treatment $trt.;
run;



/* HALL-WELLNER CONFIDENCE BANDS */

/* Arcsine-square root transformed */

proc lifetest data=patients plots=survival(cb=hw strata=panel)
    conftype=ASINSQRT;
time time_to_effect*censored(1);
strata treatment;
label time_to_effect="Time to treatment effect (days)";
format treatment $trt.;
run;

/*See code for EP bands above if you want to make adjustments to the plot*/
```

4.5 Proc LIFETEST Mean and median survival time and CI

```
/* MEDIAN survival time with confidence interval (LOGLOG transformation by
default) is presented with proc lifetest */

proc lifetest data=patients plots=none;
time time_to_effect*censored(1);
strata treatment;
run;

/* Choose transformation with 'conftype' option */
/* The Nelson-Aalen estimator is also being used here */
proc lifetest data=patients nelson method=breslow conftype=linear;
time time_to_effect*censored(1);
strata treatment;
run;
```

```
/* MEAN survival time */

/* If the last observation is censored: */
proc lifetest data=patients plots=none;
time time_to_effect*censored(1);
strata treatment;
run;
/* Largest event time is used as default */

/* Efron's tail correction */
proc lifetest data=patients plots=none
   timelim=observed; /* Uses the largest observed time */
time time_to_effect*censored(1);
strata treatment;
ods output Means=mean_surv; /*Store mean and stderr in dataset mean_surv*/
run;

data mean_surv; set mean_surv;
LCL=mean-1.96*stderr;
UCL=mean+1.96*stderr;
run;
```

4.6 Proc LIFETEST Cumulative Incidence

```
/* One single indicator variable defining all competing risks is needed, to
be constructed if not already available */

/* As an example, in the patients dataset, there is one event/censoring
indicator named "censored", where the value 0 means that there is an event
(the primary event). There is also a death indicator (competing risk),
where the value 1 means that the second event (competing risk) has
occurred. */

data patients;
set patients;
if censored=0 then comprisk=1;      /* The primary event */
else if death=1 then comprisk=2;   /* The second event (competing risk) */
else comprisk=0;      /* Censoring without any competing risks */
run;

/* Note that the order might be important here! If the primary event
e.g. is relapse, and the competing risk is death, patients can first
experience a relapse and later on death – but not in the opposite
order. If we should place death before relapse in the code above, we
would miss some of the relapses. */


/* Cumulative incidence for risk/probability of treatment effect */

proc lifetest data=patients plots=cif(test); /*Cumulative incidence plot*/
time time_to_effect*comprisk(0) / eventcode=1;
           /* Censoring value (0) and event value (1)*/
strata treatment;
label time_to_effect="Time to treatment effect (days)";
label treatment="Treatment";
format treatment $trt.;
run;
```

```
/* Cumulative incidence for risk/probability of treatment related death */

proc lifetest data=patients plots=cif(test); /*Cumulative incidence plot */
time time_to_effect*comprisk(0) / eventcode=2; /* event value (2) */
strata treatment;
label time_to_effect="Time to treatment effect (days)";
label treatment="Treatment";
run;

/* To make adjustments to the plot, store the data using the ods output
statement and construct a plot with proc sgplot (below) */

proc lifetest data=patients plots=none;
time time_to_effect*comprisk(0) / eventcode=1;
strata treatment;
ods output cifPlot=cifPlot;      /* Save the data for the CIf plot */
run;


ods graphics / reset attrpriority=none;
/* Override the built-in attributes in proc sgplot */

proc sgplot data=cifPlot;
styleattrs datalinepatterns=(solid dash shortdash);
step x=Time y=CIF / group=Stratum;
title "Probability of treatment effect";
xaxis label ="Time to treatment effect (days)";
yaxis label="Cumulative Incidence"
   values=(0 to 1 by 0.2);
label Time="Time to treatment effect (days)";
label Stratum="Treatment";
run;
```

4.7 Proc PHREG Cox regression

```
/* Categorical covariates either need to be recoded to dummy variables (see
section 1.10), or defined by the class statement with chosen reference
category */

proc phreg data=patients;
model time_to_effect*censored(1)=treatmentB diagn_day; /* Dummy */
run;
/* Remember to include all dummies but one if you have more than two
categories */

proc phreg data=patients;
class treatment(ref='A'); /* Reference category for categorical variable */
model time_to_effect*censored(1)=treatment diagn_day;
run;

proc phreg data=patients;
class treatment(ref=first); /* Possible to choose ‘first’ or ‘last’
                           category as reference category */
model time_to_effect*censored(1)=treatment diagn_day;
run;
```

```
/* HANDLING TIES */

proc phreg data=patients;
model time_to_effect*censored(1)=treatmentB diagn_day/ties=discrete;
run;

proc phreg data=patients;
model time_to_effect*censored(1)=treatmentB diagn_day/ties=exact;
run;


/* HAZARD RATIO CONFIDENCE INTERVALS */

proc phreg data=patients;
model time_to_effect*censored(1)=treatmentB diagn_day/ties=exact
   risklimits;  /* Default: Wald */
run;

proc phreg data=patients;
model time_to_effect*censored(1)=treatmentB diagn_day/ties=exact
   risklimits=pl; /* Profile-likelihood */
run;


/* INVESTIGATE INTERACTIONS */

/* Can be done in two different ways:
   1) Create a new variable: interaction=var1*var2;
   2) Use the divider |

/* 1) */
proc phreg data=patients;
model time_to_effect*censored(1)=treatmentB diagn_day treatmentB*diagn_day;
run;

/* 2) */
proc phreg data=patients;
model time_to_effect*censored(1)=treatmentB|diagn_day;
run;

/* The HAZARDRATIO statement requests hazard ratios for any variable in the
model at specified values of the interacting variable */

proc phreg data=patients;
model time_to_effect*censored(1)=treatmentB|diagn_day;
hazardratio treatment/at (diagn_day=2,4); /* If no values of diagn_day is
                                 chosen, the mean value will be used. */
hazardratio diagn_day;  /* All values of treatment will be used */
run;
```

4.7.1 Survival curves estimated from Cox regression

```
/* Define some values of the covariates in the dataset inrisks below, for
which the estimated survival curves will be plotted. Time-dependent
covariates cannot be included. */

/* Survival curves will be plotted for each of the two treatment groups A
and B, and for the value 18 of the variable diagn_day (the median day of
diagnosis is 18) */
```

```
data inrisks;
input treatment $ diagn_day label & $ 25.;
datalines;
A 18 Treatment A, diagn_day=18
B 18 Treatment B, diagn_day=18
;
run;

/* Estimate model using covariate values from dataset inrisks, using the
baseline statement */
proc phreg data=patients plots(overlay)=survival;
class treatment(ref='A');
model time_to_effect*censored(1)=treatment diagn_day/ties=exact;
baseline covariates=inrisks / rowid=label;
run;


/* PLOT ESTIMATED SURVIVAL FOR A STRATIFIED MODEL */

/* Estimated survival plots can be provided for the different strata of a
stratification covariate. Define values of the other covariates in the
dataset inrisks2 as below. */

data inrisks2;
input diagn_day label $ 12.;
datalines;
18 diagn_day=18
;
run;

proc phreg data=patients plots(overlay)=survival;
class treatment(ref='A');
model time_to_effect*censored(1)= diagn_day/ties=exact;
strata treatment;
baseline covariates=inrisks2 / rowid=label;
ods output SurvivalPlot=survivalplot; /* Store plot values */
run;


/* There will be no information about the stratification covariate included
in the survivalplot data, relevant information is added below. The survival
estimates are provided in the same order as the strata are sorted,
numerically or alphabetically (check the output from proc phreg above)*/

data survivalplot; set survivalplot;
row=_n_;
if row <=3 then hospital="Hospital 1";
else hospital="Hospital 2";  /* Add more groups here if needed */
run;


/* Produce the plot */
ods graphics / reset attrpriority=none;

proc sgplot data=survivalplot;
styleattrs datacontrastcolors=(blue red) /* Add more colors if >2 groups */
      datalinepatterns=(solid shortdash);/*Same no. of patterns as colors*/
step x=time y=survival/group=hospital;
label hospital='Hospital';
label time='Time to effect (days)';
title 'Estimated survival';
run;
```

4.7.2 Cox regression Model selection

```
/* FORWARD, BACKWARD, AND STEPWISE SELECTION */

proc phreg data=mydata;
model mytime*myevent(0)=myvar1 myvar2 myvar3 myvar4 myvar5 /ties=exact
    selection=forward;
run;
/* Backward selection: selection=backward */
/* Stepwise selection: selection=stepwise */


/* Force a covariate into the model */

proc phreg data=mydata;
model mytime*myevent(0)=myvar1 myvar2 myvar3 myvar4 myvar5 /ties=exact
    selection=forward include=1;
run;


/* Choose the level of entry/exit with stepwise selection: */

proc phreg data=mydata;
model mytime*myevent(0)=myvar1 myvar2 myvar3 myvar4 myvar5 /ties=exact
    selection=stepwise slentry=0.1 slstay=0.1;
run;


/* Best subset approach */

proc phreg data=mydata;
model mytime*myevent(0)=myvar1 myvar2 myvar3 myvar4 myvar5 /ties=exact
    selection=score best=3; /* "best" denotes the number of models to be
                        displayed for each number of explanatory variables */
run;


/* All possible model selection */

Read more at:
```

http://www2.sas.com/proceedings/forum2008/375-2008.pdf

4.7.3 Cox regression Local tests

```
/* Local tests using the wald test */

proc phreg data=mydata;
model mytime*myevent(0)=myvar1 myvar2 myvar3 myvar4 /ties=exact
    test myvar2=myvar4;
run;

proc phreg data=mydata;
model mytime*myevent(0)=myvar1 myvar2 myvar3 myvar4 /ties=exact
    test myvar2, myvar3, myvar4;
run;
/* the local Ho myvar2=0 AND myvar3=0 AND myvar4=0 is tested */
```

```
proc phreg data=mydata;
model mytime*myevent(0)=myvar1 myvar2 myvar3 myvar4 /ties=exact
   test myvar2=0, myvar3=0, myvar4=0;
run;
/* Same test as the one above */


/* To obtain the likelihood test for the local test above (var2=0 AND
var3=0 AND var4=0), we need to obtain the likelihood for both the model
with var1 as the only covariate and the model with all variables as
covariates. These can be obtained by running each model with proc phreg. */

proc phreg data=mydata outest=out1_to_4 noprint;
                                  /* "noprint" suppresses the output */
model mytime*myevent(0)=myvar1 myvar2 myvar3 myvar4 /ties=exact
run;

proc phreg data=mydata outest=out1 noprint;
model mytime*myevent(0)=myvar1 /ties=exact
run;

proc sql;
select -2*( out1._lnlike_- out1_to_4._lnlike_ ) as lratio
  from out1, out1_to_4;
quit;
/* This provides the chi-square value, with q degrees of freedom (q=no. of
betas in Ho, 3 in this example)*/
/* Note that the run; statement is not needed with proc sql, each row is
executed instantly. Instead, the quit; statement shows when the execution
mode is ended. */


/* Estimating the variance-covariance matrix */

proc phreg data=mydata;
model mytime*myevent(0)=myvar1 /ties=exact covb;
run;
```

4.7.4 Cox regression Time dependent covariates

```
/* Assume dataset mydata contains the following variables:
timeZ1 = time for the time dependent covariate
indicator = indicator for the time dependent covariate
myvar1, myvar2 = covariates (not time dependent) */

/* Time dependent covariate Zt is constructed in proc phreg below */
proc phreg data=mydata;
model mytime*myevent(0)=myvar1 myvar2 Zt/ties=exact;
if timeZ1<=mytime and indicator=1 then Zt=1; else Zt=0;
run;


/* TESTING THE PH ASSUMPTION BY USING TIME-DEPENDENT COVARIATE(S) */

proc phreg data=mydata;
model mytime*myevent(0)= myvar1 myvar2 lnt_myvar1/ties=exact;
lnt_myvar=log(mytime)*myvar1;
run;
```

4.7.5 Checking the proportional hazards assumption graphically

```
/* PLOTS OF LOG CUMULATIVE BASELINE HAZARDS */

proc phreg data=patients noprint;
model time_to_effect*censored(1) = ; /* Fit a Cox model without covariates,
                                 to provide baseline cumulative hazards */
strata treatment;
output out = logcumhaz logsurv = ls; /* -logsurv = cumulative hazard
                                 (baseline since there are no covariates)*/
run;


data logcumhaz; set logcumhaz;
logcumhaz=log(-ls);
if treatment = 'A' then logh1 = logcumhaz; /* Create separate variables for
                                 the log cumhaz in the different strata,
                                 for later plots*/
if treatment = 'B' then logh2 = logcumhaz;
run;


proc sort data=logcumhaz;
by time_to_effect;
run;


data logcumhazplot;
set logcumhaz;
retain L1 L2 . ;      /* Create an imputation variable, to fill the blanks
                      for logh1 and
                      logh2. If missing for the first times, we use either
                      . or a "large" value for log cumhaz. At time 0, the
                      cumhaz is 0, and ln(0) is undefined. "retain" means
                      that for a missing value, the value on the previous
                      line will be used */
if logh1 ne . then L1 = logh1;   /* When logh1 has values, the imputation
                      variable gets the same values */
if logh2 ne . then L2 = logh2;
diff = L1 - L2;       /* Difference in log cumulative baseline hazard
                      between stratum 1 (poverty) and stratum 2 (not
                      poverty) */
run;


ods graphics / reset attrpriority=none;
/* Needed to override the built-in attributes in proc sgplot */

proc sgplot data=logcumhazplot;
step x=time_to_effect y=diff /lineattrs=(Color=blue);
refline 0 / axis = y;
yaxis label = "Difference in log Cumulative Hazard Rates";
title 'Difference in log cumulative baseline hazard rates';
title2 '(Treatment A - B)';
run;
```

```
/* ANDERSEN PLOT OF CUMULATIVE BASELINE HAZARD */

proc phreg data=patients noprint;
model time_to_effect*censored(1) =; /* Fit a Cox model without covariates,
                                 to provide baseline cumulative hazards */
strata treatment;
output out = cumhaz logsurv = ls;  /* -logsurv = cumulative hazard(baseline
                                      since there are no covariates) */
run;


data cumhaz; set cumhaz;
cumhaz=-ls;
if treatment = 'A' then h1 = cumhaz;        /* Create separate variables for
                                 the log cumhaz in the different strata,
                                 for later plots */
if treatment = 'B' then h2 = cumhaz;
run;


proc sort data=cumhaz;
by time_to_effect;
run;


data andersen;
set cumhaz;
retain haz1 haz2 0 ;/* Create an imputation variable, to fill the blanks
                      for h1 and h2. If missing for the first times, we
                      use cumhaz=0 (at time 0, the cumhaz is 0) "retain"
                      means that for a missing value, the value on the
                      previous line will be used */
if h1 ne . then haz1 = h1; /* When logh1 has values, the imputation
                      variable gets the same values */
if h2 ne . then haz2 = h2;
run;


proc sgplot data=andersen noautolegend;
step x=haz1 y=haz2;
series x=haz1 y=haz1/lineattrs=(color=black);
yaxis label = "Treatment A";
xaxis label = "Treatment A";
title 'Andersen plot to check the PH assumption of the treatment
covariate';
run;
/* “noautolegend” removes the legend at the bottom of the plot */
/* ARJAS PLOT */
```

Note: To investigate the PH assumption for a continuous covariate you have to categorize the variable.

```
/* Investigate the PH assumption for a covariate with two categories */

/* Estimate the hazard by fitting a Cox model without the covariate for
which we want to check the PH assumption */

proc phreg data=mydata;
model mytime*mycens(1) = myvar1 myvar2 /ties=exact;
output out = hazarjas logsurv=ls;
run;

data hazarjas; set hazarjas;
cumhaz=-ls;                      /* Calculate cumulative hazard */
run;

/* Count the number of 'times' for each category of the investigated
variable, and create two macro variables containing those numbers */

proc sql noprint;
select count(mytime) into :t1-:t2  /* macro variables t1 and t2 */
from mydata           /* dataset to be used */
group by myvar;       /* covariate to be investigated */
quit;
/* Note that the run; statement is not needed with proc sql, each row is
executed instantly. Instead, the quit; statement shows when the execution
mode is ended. */

/* Print all macro variables, including the values of t1 and t2 above.
Check that the numbers are reasonable and note the largest one – to be used
in the plot later on */
%put _all_;

proc sort data=hazarjas;
by cumhaz;
run;

/* Calculate “total time on test” */

data arjas; set hazarjas;
myevent = 1-mycens;   /* Create an event variable if not existing */
retain n1 n2 h1 h2 c1 c2 0;      /* h=logsurv (estimated above) */
if cumhaz ne . then do;
   if myvar = 1 then do ;        /* Use first value of myvar */
                         /* Calculations are being made separately
                         for each stratum of the covariate to be checked */
   c1 = c1 + 1;          /* Starts at 0, increases by 1 for every
                         observation in this stratum */
   n1 = n1 + myevent;    /* Counts the no. of events */
   h1 = cumhaz + h1;    /* Sums the cumulative hazard up to each time */
   end;
else if myvar = 0 then do;       /* Use second value of myvar */
   c2 = c2 + 1;
   n2 = n2 + myevent;
   h2 = cumhaz + h2;
   end;
end;
tot1 = h1 + cumhaz*(&t1-c1); /* Total time on test within the stratum */
tot2 = h2 + cumhaz*(&t2-c2);
run;
```

```
proc sgplot data=arjas noautolegend;
series x=n1 y=tot1;
series x=n2 y=tot2;
series x=n2 y=n2; /* Pick the stratum with the largest n to plot the 45
degree line from. Can restrict the axes to the largest n as well (below) */
xaxis label='Number of events' min=0 max=800; /* Use largest of T1 or T2 */
yaxis label='Estimated Cumulative Hazard Rates' min=0 max=800;
title 'Arjas plot myvar';
run;



/* Investigate the PH assumption for a covariate with three categories
(or more) */

/* Estimate the hazard without the covariate for which we want to check the
PH assumption */

proc phreg data=mydata;
model mytime*mycens(1) = myvar1 myvar2 myvar3/ties=exact;
output out = hazarjas logsurv=ls;
run;

data hazarjas; set hazarjas;
cumhaz=-ls;                      /* Calculate cumulative hazard */
run;

/* Count the number of 'times' for each category of the investigated
variable, and create macro variables containing those numbers */

proc sql noprint;
select count(time) into :t1-:t3  /* macro variables t1, t2, and t2 */
from mydata           /* dataset to be used */
group by myvar;       /* covariate to be investigated */
quit;

/* Print all macro variables. Check that the numbers of t1, t2, t3, are
reasonable and note the largest one – to be used in the plot later on */

%put _all_;

proc sort data=hazarjas;
by cumhaz;
run;
```

```
/* Calculate “total time on test” */

data arjas; set hazarjas;
myevent = 1-mycens;   /* Create an event variable if not existing */
retain n1 n2 n3 h1 h2 h3 c1 c2 c3 0; /*Add here if more than 3 categories*/
if cumhaz ne . then do;
   if myvar = 1 then do ;        /* Use first value of myvar */
   c1 = c1 + 1;
   n1 = n1 + myevent;
   h1 = cumhaz + h1;
   end;
else if myvar = 2 then do;       /* Use second value of myvar */
   c2 = c2 + 1;
   n2 = n2 + myevent;
   h2 = cumhaz + h2;
   end;
else if myvar = 3 then do;       /* Use third value of myvar */
   c3 = c3 + 1;
   n3 = n3 + myevent;
   h3 = cumhaz + h3;
   end;               /* Add here if more than 3 categories */
end;
tot1 = h1 + cumhaz*(&t1-c1); /* Total time on test within the stratum */
tot2 = h2 + cumhaz*(&t2-c2);
tot3 = h3 + cumhaz*(&t3-c3); /* Add here if more than 3 categories */
run;


proc sgplot data=arjas noautolegend;
series x=n1 y=tot1;
series x=n2 y=tot2;
series x=n3 y=tot3;
series x=n2 y=n2;  /*Use largest of n1 to n3*/
xaxis label='Number of Failures' min=0 max=440; /*Use largest of T1 to T3*/
yaxis label='Estimated Cumulative Hazard Rates' min=0 max=440;
title 'Arjas plot for myvar';
run;




/* STANDARDIZED SCORE RESIDUALS */

proc phreg data=mydata;
model mytime*mycens(1)= myvar1 myvar2 myvar3 myvar4 /ties=exact;
assess ph npaths=0;   /* The observed score process only, without any
           estimated variants (npaths=0) */
run;
/* This provides standardized score residual plots for all covariates, one
at a time */

proc phreg data=mydata;
model mytime*mycens(1)= myvar1 myvar2 myvar3 myvar4 /ties=exact;
assess ph /resample;  /* "resample" performs a test of the PH assumption,
                                 based on simulations */
run;
```

```
/* SMOOTHED PLOT OF SCALED SCHOENFELDT RESIDUALS */

proc phreg data=mydata;
model mytime*mycens(1)= myvar1 myvar2 myvar3 myvar4 /ties=exact;
output out = schoenf ressch = sch_myvar1 sch_myvar2 sch_myvar3 sch_myvar4;
run;
/* This provides standardized Schoenfeldt residuals for each of the
covariates */

/* Check for non-linear relationships with time by e.g. calculate the log
of time (alternatively use the rank of time)*/
data schoenf; set schoenf;
logtime = log(time);
run;

proc loess data = schoenf;
model sch_myvar1 = logtime / smooth=(0.2 0.4 0.6 0.8) direct;
run;
/* “smooth=” gives the fraction of the data to be used as “nearby points”.
“direct” fits a separate OLS for each local neighborhood. The default
option fits one OLS for a representative sample of points and interpolates
from that (less computationally demanding). */
```

4.7.6 Cox regression diagnostics (residuals)

```
/* COX-SNELL RESIDUALS TO CHECK THE MODEL FIT */

/* Note: Cox-Snell residuals cannot be calculated for a model with time-
dependent covariates, the model fit thus has to be investigated without any
time-dependent covariates */

proc phreg data=mydata noprint;
model mytime*mycens(1)= myvar1 myvar2 myvar3 myvar4 myvar5 /ties=exact;
output out = coxsnell LOGSURV = h;
run;

/* Calculate cumulative hazards based on the Cox model, i.e. adjusting for
covariate values */
data coxsnell; set coxsnell;
r=-h;
run;

/*  Nelson-Aalen estimates based on r */
proc lifetest data=coxsnell nelson plots=none;
time r*mycens(1);
ods output ProductLimitEstimates=figure;
run;


/* Alternatively: Fitting a Cox model without covariates also gives Nelson-
Aalen estimates */
proc phreg data = coxsnell noprint;
model r*mycens(1)=;
output out = figure LOGSURV = ls;  /*-logsurv gives the Nelson-Aalen
                          estimates of the Cox-Snell residuals */
run;

data figure; set figure;
cumhaz=-ls;
run;
```

```
proc sort data=figure;
by r;
run;

proc sgplot data = figure noautolegend;
step x=r y=cumhaz;
series x=r y=r;
xaxis label = 'Residual';
yaxis label = 'Estimated Cumulative Hazard Rates';
title "Cox-Snell residuals to check the overall fit of the model";
run;


/* MARTINGALE RESIDUALS TO DETERMINE THE FUNCTIONAL FORM OF A COV. */

/* Fit the model without the covariate of interest */

proc phreg data=mydata noprint;
model mytime*mycens(1)= myvar1 myvar2 myvar3 myvar4 myvar5 /ties=exact;
output out = martingale resmart=mgale;
run;

proc loess data=martingale;
model mgale = myvar / smooth=0.7 direct; /* myvar = covariate of interest*/
label myvar = "Martingale residuals of myvar";
ods output OutputStatistics=martplot;
run;
/* “smooth=” gives the fraction of the data to be used as “nearby points”.
Default for a model with at least 2 covariates: 0.5 */
/* “direct” fits a separate OLS for each local neighborhood. The default
option fits one OLS for a representative sample of points and interpolates
from that (less computationally demanding). */



/* DEVIANCE RESIDUALS TO EXAMINE OUTLIERS */

/* Create id variable (if not already existing), to be able to identify
outliers. */
data mydata;
set mydata;
id=_n_;    /* Row number used as id */
run;

/* Calculate risk scores and deviance residuals */
proc phreg data=mydata noprint;
model mytime*mycens(1)= myvar1 myvar2 myvar3 myvar4 myvar5 /ties=exact;
output out=devres xbeta=Risk_score resdev=Deviance_res;
run;

proc sgplot data=devres;
scatter x=Risk_score y=Deviance_res;
refline 0 / axis=y;
yaxis grid;
run;
/* 'refline' produces a reference line at value 0 for the chosen axis*/

/* Sort the data to find the largest values of the residuals, i.e. to
investigate possible outliers */
proc sort data=devres out=x;
by Deviance_res Risk_score;
run;
```

```
proc print data=devres noobs;
var id myvar mytime mycens Risk_score Deviance_res;
run;


/* SCHOENFELD RESIDUALS TO CHECK THE INFLUENCE OF INDIVIDUAL OBSERVATIONS
*/

proc phreg data=mydata noprint;
model mytime*mycens(1)= myvar1 myvar2 myvar3 myvar4 myvar5 /ties=exact;
output out=schoenfres ressch=sch_myvar1 sch_myvar2 sch_myvar3 sch_myvar4
   sch_myvar5;
run;

proc sort data=schoenfres;
by mytime;
run;

data schoenfres; set schoenfres;
id=_n_;    /* Row number used as id */
run;

proc sgplot data=schoenfres;
scatter x=id y=sch_myvar1;
refline 0 / axis=y;
yaxis grid;
label id='Observation number';
run;
```

4.7.7 Investigating multicollinearity

```
/* The default of PROC CORR produces Pearson’s correlation */
proc corr data=mydata noprob;  /* ‘noprob’ excludes p-values */
var myvar1 myvar2 myvar3;
run;

/* Add the statement ‘spearman’ to get Spearman’s correlation */
proc corr data=mydata spearman noprob;
var myvar1 myvar2 myvar3;
run;

/* VIF (Variance Inflation Factor)
(VIF is based on linear regression. How this translates to Cox regression
is yet unknown, we’ll use the VIF until more research is available.) */

/* Create a fake variable to use as dependent variable in linear
regression, where all observations have the same value (no variation) */

data temp;
set mydata;
fake=1;
run;

proc reg data=temp;
model fake=myvar1 myvar2 myvar3 /vif;
run;

/* There are many rules of thumb to be found in the literature, most of
them say that VIF>3 or 4 warrants further investigation, and VIF>10 is a
sign of severe multi-collinearity */
```

4.8 Proc LIFEREG Parametric regression

Note: It is not possible to define reference category with the class statement in proc lifereg, thus dummy variables must be used for categorical variables.

```
/* If the distribution of time to event is unknown, proc severity can be
used to find possible distributions for event times */
proc severity data=mydata crit=aicc;        /* aicc = corrected AIC */
where myevent=1;      /* Fit distribution on event times only */
loss mytime;          /* The variable to be fitted */
dist _predefined_;    /* Fits all predefined distributions */
run;
/* EDF = empirical distribution function, i.e. the observed data*/



/* WEIBULL DISTRIBUTION */

/* Estimates of gamma (for Y=ln X) */
ods output ParameterEstimates = estimates covB = covB;
proc lifereg data=mydata;
model mytime*myevent(0) = myvar1 myvar2 myvar3 myvar4 / covb
   distribution=weibull;
run;


/* Covariance matrixes above are used to transform the results in terms of
beta estimates (for X) */
proc iml; /* Interactive Matrix Language */
use CovB;
read all var {Intercept myvar1 myvar2 myvar3 myvar4 Scale};
x = Intercept || myvar1 || myvar2 || myvar3 || myvar4 || Scale ;

start grd(g); /* creating the gradient matrix corresponding to
    the transform */
gdim = nrow(g);
grdn = j(gdim, gdim, 0);
grdn[1,1] = - exp(-g[1]/g[gdim])/g[gdim];
grdn[1,gdim] = (exp(-g[1]/g[gdim])*g[1])/(g[gdim]**2);
do i = 2 to gdim - 1;
   grdn[i, gdim]=g[i]/(g[gdim]**2);
   grdn[i,i] = - 1/g[gdim];
   end;
grdn[gdim,gdim] = -1/(g[gdim]**2);
return (grdn) ;
finish grd;

use estimates;
read all var {parameter estimate};
est = estimate;
r = nrow(est);
g = est[1:r-1, 1];
parm = parameter;
parms = parm[1:r-1,1];
mytet = grd(g);
newvar = sqrt(diag(mytet*x*t(mytet))); /*delta method to
calculate the variance*/
gnew = j(r-1, 1, 0);
gstd = j(r-1, 1, 0);
expd = j(r-1, 1, 0);
```

```
do i = 1 to r-1; /*pull out the standard error*/
   gstd[i]=newvar[i,i];
   end;
gnew[1] = exp(-g[1]/g[r-1]); /*doing the transformation from Y=ln(X) to X*/
gnew[r-1] = 1/g[r-1];
do i = 2 to r-2;
   gnew[i] = - g[i]/g[r-1];
   end;
do i = 1 to r-1;
   expd[i] =exp(gnew[i]);       /* Exponentiating the beta coefficients =>
                                 Acceleration factor */
   end;

print "Transformed Parameter Estimate";
print gnew[rowname = parms colname = "Parameter Estimate"
   label=" " format=8.3]
   gstd[colname = " Standard Error" label = " " format=8.3]
   expd[colname = "Exp Estimate" label = " " format=8.3];
quit;



/* LOG LOGISTIC */
/* Estimates of gamma (for Y=ln X) */
ods output ParameterEstimates = llestimates covB = llcovB;
proc lifereg data=mydata;
model mytime*myevent(0) = myvar1 myvar2 myvar3 myvar4 / covb
   distribution=llogistic;
run;

/* Covariance matrixes above are used to transform the results in terms of
beta estimates (for X) */

proc iml;
use llcovB;
read all var {Intercept myvar1 myvar2 myvar3 myvar4 Scale};
x = Intercept || myvar1 || myvar2 || myvar3 || myvar4 || Scale ;

start grd(g); /*creating the gradient matrix corresponding to the
     transform*/
gdim = nrow(g);
grdn = j(gdim, gdim, 0);
grdn[1,1] = - exp(-g[1]/g[gdim])/g[gdim];
grdn[1,gdim] = (exp(-g[1]/g[gdim])*g[1])/(g[gdim]**2);

do i = 2 to gdim - 1;
   grdn[i, gdim]=g[i]/(g[gdim]**2);          grdn[i,i] = - 1/g[gdim];
   end;
grdn[gdim,gdim] = -1/(g[gdim]**2);
return (grdn) ;
finish grd;

use llestimates;
read all var {parameter estimate};
est = estimate;
parm = parameter;
r = nrow(est);

mytet = grd(est);
newvar = sqrt(diag(mytet*x*t(mytet))); /*delta method to calculate
           the variance*/
gnew = j(r, 1, 0);
gstd = j(r, 1, 0);
```

```
expd = j(r, 1, 0);
propodds = j(r, 1, 0);

do i = 1 to r; /*pull out the standard error*/
   gstd[i]=newvar[i,i];
   end;
gnew[1] = exp(-est[1]/est[r]); /*doing the transformation*/
gnew[r] = 1/est[r];
do i = 2 to r-1;
   gnew[i] = - est[i]/est[r];
   end;
do i = 1 to r;
   expd[i] =exp(gnew[i]);        /* Exponentiating the beta
                      coefficients => Acceleration factor */
   end;
do i = 1 to r;
   propodds[i] =exp(-1*gnew[i]);/* Exponentiating the negative
                      beta coefficients => Proportional odds */
   end;

print "Transformed Parameter Estimate";
print gnew[rowname = parm colname = "Parameter Estimate" label=" "
   format=8.3]
   gstd[colname = "  Standard Error" label = " " format=8.3]
   expd[colname = "Exp Estimate" label = " " format=8.3]
   propodds[colname = "Exp(-Estimate)" label = " " format=9.3];
quit;



/* Model fit can be evaluated by Cox-Snell residuals also for an
accelerated failure time model */

proc lifereg data = mydata noprint;
model mytime*myevent(0)= myvar1 myvar2 myvar3 /
   distribution=weibull;  /* Use the distr. that fits your data */
output out=residuals cdf=cumdistr;  /* Store cum. distr. estimates*/
run;

/* Cumulative hazards based on the Cox model, i.e. adjusting for covariate
values */
data residuals;
set residuals;
coxsnell = -log(1-cumdistr);
run;

/*  Nelson-Aalen estimates based on the residuals */
ods output ProductLimitEstimates=figure;
proc lifetest data=residuals nelson plots=none;
time mytime*myevent(0);
run;

proc sort data=figure;
by coxsnell;
run;

proc sgplot data = figure noautolegend;
step x=coxsnell y=cumhaz;
series x=coxsnell y=coxsnell;
yaxis label = "Estimated Cumulative Hazard Rates";
xaxis label = "Model residuals (Cox-Snell)";
run;
```