Using The SAS® System For Medical Data Processing And Event Analysis: An Overview

Harald Pitz, Frankfurt University Hospital
Marcus Frenz, Frankfurt University Hospital
Hans-Peter Howaldt, Frankfurt University Hospital

Abstract
This paper shows necessary methods and appropriate SAS utilities (data and procedure steps) to perform medical data analysis. The methods described include data acquisition (getting data into the SAS System), data preparation (preparing data for presentation and analysis), data presentation (printing charts and tables) and survival analysis (comparison of survival curves and computing prognostic factors). Especially the second step is very important as it facilitates subsequent presentation and analysis.

By modularizing these steps SAS applications become clear and easy to modify and to extend. This is supported by features such as macro processing, formats and SAS functions. This paper gives an overview to novice users and demonstrates the use of these techniques to experienced users.

1. Data acquisition
The first step of data processing with the SAS System is to create a SAS dataset. How to create a dataset depends on the source of the data. Basically four different methods can be distinguished:

PROC FSEDIT
The data can directly be entered into the SAS System using PROC FSEDIT included in SAS/FSP®. First a new data set has to be created using the NEW= option. For each variable its name, data type and length are required. The SAS System provides only numeric and character data types. Processing other data types such as date values is implemented by formats for input (called informat) and output (called formats). When the dataset description phase is finished, data can be entered in full screen mode. Formats, informat, labels and variable names can be modified using PROC DATASETS. Other modifications require the creation of a new dataset. PROC FSEDIT called with options NEW= and LIKE= allows creating a dataset similar to an existing one which can then be modified.

PROC FSEDIT creates default screens for data entry. These can be modified by using the MODIFY command. The use of the SCREEN= option in the PROC FSEDIT statement is required to store the screen in a catalog. Colors, initial values, minimum, maximum and other parameters can be stored for each data field. The screen modification may become a troublesome work when using datasets with many variables, especially if the structure of the dataset has been modified. A precise preparation of the dataset creation by collecting all necessary variables and data types is recommended to save time.

/* Creating a dataset with FSEDIT */
proc fseedit new=first
    screen=funny;
run;

/* Modifying a dataset with FSEDIT */
proc fseedit new=second like=first
    screen=funny;
run;

PROC DBF / PROC DIF
If the data is already stored in a different database system, the files can be converted to SAS datasets. Base SAS software supports dBASE files (PROC DBF) and Data Interchange Format files (PROC DIF) created by many software packages. SAS can also convert datasets to these files. The correct handling of data types and missing values should be verified.
Converting a dBASE-III file

filename demodbf "\SUGIDEMO\EX.DBF";
proc dbf db3-demodbf out=itworks;
run;

SAS/ACCESS®

SAS/ACCESS is a software product which can be used to access databases under various operating systems.

Files and databases not directly supported by SAS software can be processed with the fourth method described below.

Data step / INPUT statement

This method is appropriate if data is stored in external files which cannot be processed with one of the procedures listed above. The dataset is created by a data step. The format of the external files has to be described using the input statement. The most simple method is to read input arranged in columns. Various other modes exist to read more complicated data, e.g., observations consisting of multiple input lines. These methods frequently have to be applied to read the output produced by medical devices. Formats and Informats are important features to process those data files.

/* Reading a file with columns */
data columns;
input SEX 1
BIRTHDAY 2-11 DIAGDATE 12-21;
informat BIRTHDAY DIAGDATE DDMMYY10.;
format BIRTHDAY DIAGDATE DDMMYY8.;
infile "\SUGIDEMO\DATES.DAT";
run;

PROC FSEDIT, PROC FSPRINT and PROC FSBROWSE can be used to edit or review all datasets created by any of the described methods. Customized screens should be generated if a dataset contains many variables or if one of the FSP procedures is used frequently.

2. Data Preparation

For data analysis information contained in several variables of the raw data has to be combined. This is done by generating new variables in an additional data step. The following example shows the calculation of a patient's age at the time of the diagnosis of the disease. Since SAS date values are stored as numeric values the age can be easily computed by subtracting the birth date (Variable BIRTHDAY) from the date of the diagnosis (DIAGDATE):

/* Calculation of age */
data newcol;
set columns;
AGE = DIAGDATE - BIRTHDAY;
run;

The SAS language contains a large set of functions, operators and statements to create new variables. Frequently the values of a variable have to be converted for two reasons:

a) Coded data can be converted to strings for better readability. If the sex of a patient is stored as codes 1 or 2 you may wish to read them as 'male' or 'female'.

b) Continuous variables can be projected to interval scaled variables. The main purpose for that is building groups.

Formats and Informats

For conversions the SAS System provides formats and informats. In addition to the standard formats included in SAS additional formats can be defined using PROC FORMAT. The use of formats produces better readable programs and is much more efficient than the classic way of if-then-else programming. The following examples demonstrate the two ways to convert variables:

/* Conversion using if-then-else */
data new;
set old;
if SEX=1 then SEXCONV="Male";
else if SEX=2 then SEXCONV="Female";
else SEXCONV="Error";
run;

/* Conversion using a format */
/* by creating a new variable */
proc format;
value f sex
1="Male"
2="Female"
other="Error";
run;
data new;
set old;
SEXCONV=put(SEX,f_sex.);
run;
Various methods can be used to handle formats. Some of them modify the variable, other require creating a new variable, other modify the view of the user onto a variable. The following list shows three basic methods using formats:

a) Assigning formats temporarily

The easiest way is to use formats in a procedure step. By assigning a format to a variable only the view onto this variable (its value) is changed, not the variable in the dataset itself. The assignment is temporary and exists only during the execution of the procedure step.

```sas
/* Temporary format in a proc.step */
proc print data=old;
  format SEX f_sex.;
run;
```

b) Assigning formats permanently

If the formatted value of a variable is needed always the format can be assigned permanently using a data step. This method requires that the format exists whenever the variable is accessed.

```sas
/* Assigning a format permanently */
data new;
  set old;
  format SEX f_sex.;
run;
```

c) Creating a new variable

Instead of assigning a format to an existing variable a new variable containing the formatted value can be created. This can only be achieved with an additional data step. Both the original and the converted variable can be kept in a dataset as demonstrated in the example above. The advantage of this method is that the format only has to exist during the data step. The disadvantage is that storing the formatted value requires much more disk space than storing the original value.

```sas
/* Creating a new formatted variable */
data new;
  set old;
  SEXF = put(SEX,f_sex.);
run;
```

Formatting continuous variables

Using formats continuous variables can be easily projected to interval scaled variables. To process the age of a patient in decades the following format can be used:

```sas
/* Using formats to map ranges */
proc format;
  value decades
    low  -< 0  = "Missing"
    0   -< 10 = "0 - 10"
    10  -< 20 = "10 - 20"
    20  -< 30 = "20 - 30"
    30  -< 40 = "30 - 40"
    40  -< 50 = "40 - 50"
    50  -< 60 = "50 - 60"
    60  -< 70 = "60 - 70"
    70  -< 80 = "70 - 80"
    80  -< 90 = "80 - 90"
    high >= 90 = "Error"
  run;
format age decades.;
```

Additional methods to use formats are described in the SAS Procedures Guide (PROC FORMAT).

Storing user-defined formats in a library

If you define a format using PROC FORMAT without the LIBRARY option, the format exists only during the session where it has been defined. To define formats permanently you first have to define a libref called "LIBRARY" with the directory where the format catalog should be stored. Then you specify the option LIBRARY=LIBRARY in the PROC FORMAT statement to generate a format catalog. To use these formats in a later session you only have to specify the libref "LIBRARY".

```sas
/* creating permanent formats */
libname library "SUGI\FORMATS";
proc format library = library;
  value f_sex
    1="Male"
    2="Female"
    other="Error"
  run;
```

SAS Functions

SAS functions represent another important feature for data preparation. Many of them are used for mathematical or statistical purposes. Others facilitate programming by providing frequently used functions whose implementation with the SAS language would be very troublesome.

Multiple Observations per Patient

Multiple diagnoses, examinations or therapies of one patient often require the analysis of time series. Normally your dataset contains one observation per event (e.g., consultation) per patient. For the analysis of time series SAS/ETS® can be used.

If new variables have to be calculated requiring the information from multiple events, a
different method for analysis can be used. Variables from multiple observations can be brought together to generate one observation for every patient. Each observation then contains all the data required which makes subsequent analysis easy. The following example illustrates the problem:

For each examination of a cancer patient a PID, the date and the status have been stored. Valid values for the status are 1 (first diagnosis of disease), 2 (free of disease after treatment), 3 (recurrence) and 4 (death). Any number of observations with status 2 or 3 can appear, but only one with status 1 or 4. For analysis, one observation containing the date of the first diagnosis (DIAG_1), the time until the first recurrence (if any, otherwise missing) (TIMREC_1), the time the patient has been observed (until the last examination or until death) (LIFETIME) and his final status (FINSTAT) have to be computed.

The task requires retaining variables from one observation to another. The first observation (initialization) and the last observation (final calculation and generation of the necessary observation) play a specific role in the data step. The following program shows a solution:

```
proc sort data=old;  /* 1 */
   by PID DATE;
data new;  /* 2 */
   by PID;
   retain DIAG_1 TIMREC_1 /* 3 */
   LIFETIME FINSTAT;
   label
      DIAG_1 = "Date 1st Diag."
      TIMREC_1 = "Time 1st Rec."
      LIFETIME = "Overall Lifetime"
      FINSTAT = "Final Status";
   format DIAG_1 ddmmyy6.;  /* 4 */
   informat DIAG_1 ddmmyy6.;
   if FIRST.PID then do;  /* 5 */
      DIAG_1 = DATE;
      TIMREC_1 = .;
   end:
   LIFETIME = DATE - DIAG_1;  /* 6 */
   if STATUS=3 AND TIMREC_1=. then TIMREC_1 = LIFETIME;  /* 7 */
   if LAST.PID then do;  /* 8 */
      TIMREC_1 = .;
      FINSTAT = STATUS;  /* 9 */
      output;  /* 10 */
   end:
run;
```

Explanations and Comments:
1. Sorting by PID is required for the following data step. Sorting by DATE guaranties that the first/last observation contains the data of first diagnosis/last examination.
2. The data step is performed once for each observation in the data set. For each observation, first the variables are read from the dataset, then the statements are executed. Normally, variables not contained in the data set are set to missing at the beginning of each execution. This must be avoided if the value of a variables calculated from one observation has to be retained for the next observation. The RETAIN statement must be used to retain values of variables to next observations.
3. Generated variables should have labels.
4. Generated date variables should have formats and informats for better editing or printing.
5. FIRST.PID is true if the current observation is the first of the patient. Due to sorting by DATE, this corresponds to the primary diagnosis.
6. The date of the primary diagnosis is stored in variable DIAG_1 which is retained to the next observations.
7. Initializes (reset) TIMREC_1 to missing.
8. Sets TIMREC_1 if first recurrence.
9. LAST.PID is true if the current observation is the last of the patient.
10. Final status is saved in last observation.
11. Output observation to dataset "new".

Merging Datasets

Depending on the kind of data one or more observations exist for each patient. Usually, demographic data is entered and stored only once in one observation, while multiple examinations require multiple observations. Normally, multiple datasets are needed to store the information about a patient: one dataset for demographics, one for therapies, etc. Because analysis often requires the access of variables from multiple datasets at the same time, one dataset with one observation for each patient provides a comfortable basis for later analysis. The generation of this dataset requires two steps:

1. For each dataset containing more than one observation per patient: Create a new dataset with one observation per patient using the method described above.

147
2. Merge all necessary datasets together to a new one. This requires consistent patient PID's in all datasets. The datasets must be sorted by PID. Datasets resulting from step 1 follow this requirement. Calculate new variables which consist of data from more than one dataset, e.g., the age at the time of the diagnosis using the birth date from dataset demographics and the date of diagnosis from dataset examination.

```sas
/* Merging datasets */
data all;
merge demogra exa therapy followup;
by PID;
/* further calculations */
run;
```

The resulting dataset provides a powerful basis for all kinds of analysis. New requirements concerning the analysis may require the calculation of new variables. This shouldn't be a serious problem if the methods of data preparation described above are applied.

3. Data Presentation

For the presentation of data, tables, charts and graphs are used most frequently.

For a simple calculation of frequency tables or simple statistics such as mean and standard deviation many procedures included in Base SAS Software can be used. Frequently used procedures are PROC FREQ, PROC MEANS and PROC UNIVARIATE. Low resolution ASCII-charts can be produced using PROC CHART which includes histograms, block charts and pies. The lines and the curves of ASCII-charts are generated using a set of characters (e.g., +-*.1). The options FORMCHAR= can be used to substitute the standard characters to customize the output. For printers with the IBM character set the following statement produces good results:

```sas
OPTIONS FORMCHAR='B3C40AC2BFC3C5B4CD1D9';
```

PROC TABULATE is a very powerful tool to generate tables. With this procedure most of the descriptive analysis of data can be performed. This paper cannot demonstrate all the capabilities of PROC TABULATE. We refer to the SAS Guide to TABULATE Processing.

High resolution graphics can be produced using the procedures provided by SAS/GRAPH®. PROC GCHART contains functions similar to PROC CHART. PROC GPLOT can be used to plot curves. The Customization of graphics is very troublesome. A large set of statements such as TITLE, FOOTNOTE, NOTE, AXIS, LABEL, PATTERN, SYMBOL and LEGEND with a lot of options are necessary. In addition the ANNOTATE facility can be used for further enhancements, e.g., marking of censored observations in a survival plot which is demonstrated in [Pitz91].

4. Event Analysis

One important step in medical data analysis is the analysis of events. Frequently used events in the analysis tumor data are the death of a patient (survival analysis) or the relapse of a tumor. Event analysis can be used to compare the success of different therapies or to calculate prognostic factors.

Basically two different methods are available: PROC LIFETEST containing the Kaplan-Meier method is used for creating survival plots and performing univariate analysis, PROC LIFEREG and PROC PHREG provide multivariate test statistics.

Kaplan-Meier Survival Plots

The Kaplan-Meier method contained in PROC LIFETEST is used to estimate a survival function. The plot of this function provides an optical presentation of data. If a stratum variable is specified, subgroups are built from the levels of the stratum variable, e.g., representing different therapies in a clinical study. In addition the log-rank and Wilcoxon statistic are calculated.

PROC LIFETEST produces an ASCII plot if the PLOTS= option is specified. High resolution plots can be obtained by specifying an output dataset to be plotted with PROC GPLOT.

```sas
/* PROC LIFETEST with PROC GPLOT */
PROC LIFETEST data=demo method=km
noprint outs=tmp;
time survtime*death(0);
strata therapy;
run;

symbol1 c=red i=step j=1 l=1;
symbol2 c=green i=step j=33;
axis1 order = 0 to 1825 by 365;
axis2 order = 0 to 1 by 0.1;
PROC GPLOT data=tmp;
plot survival*survtime=therapy / haxis = axis1 vaxis = axis2;
run;
```

148
Multivariate Analysis - Calculation of prognostic factors

The univariate analysis provided by PROC LIFETEST reflects the clinical view of the physician but is not sufficient to discover complex relations between multiple prognostic variables. Correct estimates of the prognostic influence for the different factors can only be obtained if the main relevant variables are known because unobserved heterogeneity may cause incorrect results. Furthermore the relevance of variables can often only be established in multivariate analysis.

Thus we perform multivariate analysis. Two approaches of multivariate event analysis are available: semiparametric and full parametric. In social science the exact distribution of the baseline rate (required for full parametric analysis) is often unknown. This leads to a preference for semiparametric analysis with the COX-model, which is implemented in PROC PHREG.

In contrast to PROC LIFETEST which directly uses variables with multinominal or ordinal data as stratum, multivariate analysis requires replacing these variables by \( n-1 \) dummy variables with 0/1 coding, if the original variable has \( n \) categories. For use in PROC LIFEREG the only way to generate the new variables is a time consuming DATA step. PROC PHREG is more flexible because the main SAS statements known from the data step are supported. In addition time-dependent covariates can be analysed. This is also useful for mapping metric scaled variables to interval scaled dummies to avoid inherent linear assumptions in the model.

```sas
/* Example of PROC PHREG */
PROC PHREG data=sugi.demo;
model survtime*death(O) = age_mid age_old
  therapy tu size grading;
  age_mid = (50 <= age < 70);
  age_old = (70 <= age);
  /* age < 50 is residual category */
run;
```

Conclusion

The data preparation is one of the most important steps to perform medical data processing. Within this step all data required for later analysis is provided by transforming the raw patient data and calculating new variables. As an important feature for this transformation SAS provides so-called formats which make programs easy to read and to modify.

Normally your patient's database consists of multiple relational tables. If possible they should be converted to one table containing one observation for every patient. Complex data steps are necessary to perform this conversion.

References


Author:

Harald Pitz
Department of Medical Informatics
J.W. Goethe University Hospital
Theodor-Stern-Kai 7
D-6000 Frankfurt/Main
Germany

Tel.: (49) 69 6301-6642
Fax.: (49) 69 6301-6301