Computer Based Statistics: Introduction in SAS

Franziska Peter

11.11.2010

What is SAS?

- SAS stands for **S**tatistical **A**nalysis **S**ystem
- Powerful and flexible tool for many purposes: data management, estimation, optimization, visualization...
- Mixture of 'easy-to-use' procedures and manual programming
- Macro language for recurring code sequences

A few general concepts of SAS

- Data steps: create data sets, create new variables, subset existing datasets,...
- Procs: analyze your data, estimation, visualization,...
- Macros: often recurring pieces of code can be written as macros, very efficient programming!!!

The SAS User Interface

1. Log window:
   - Blue is ok (in general)
   - Green is a warning (should be checked)
   - Red is an error message (**do not ignore**!)
2. Output window: Results are printed in this screen
3. Editor window: Write your program commands in here.

SAS Help

Online documentation:

`http://support.sas.com/onlinedoc/913/docMainpage.jsp`.

SAS procedures:

`http://www.technion.ac.il/docs/sas/proc/index.htm`.

Or use the help references and language guides implemented in SAS itself.

Built up your own file with commented procedures during the semesters.

Basic handling of data

To use a SAS-dataset saved on your harddisk or to create a permanent dataset, assign a library pointing to the respective folder:

`libname` *name* "*drive*:\ *folder*";

Now, you can find all datasets already in *folder* or newly created datasets in the respective library *name* in your SAS-Explorer.

***Task**: Create a new folder in your home directory for this course and assign a library to this folder.*

Working with temporary files vs. permanent files

The name of a SAS dataset consists of two parts:

library.*dataset*

Permanent dataset: choose *name* as your library, if you want to write the dataset permanently into *folder*

Temporary dataset: skip the library or choose `work` as your library to create temporary datasets.

Temporary datasets will be lost when you terminate SAS.

Getting data into your SAS system

Reading data into SAS via Data Steps

```
DATA library.dataset;
INFILE 'path.filename';
INPUT variable1 variable2 ...;
FORMAT variable format;
LABEL variable='variable label';
RUN;
```

Formats in SAS: Characters, Numeric, Dates

Getting data into your SAS system

Reading data into SAS using the import procedure

PROC IMPORT DATAFILE= *"path.filename"*
OUT =*library.dataset* DBMS=*identifier* REPLACE;
RUN;

Formats (identifier) eg: EXCEL (.xls), TAB (.txt), CSV (.csv)

→ Example in sample program.

Getting data into your SAS system

***Task**: Download the files* `abx.txt` *and* `aby.txt` *from the course homepage. Each file contains data of New York Stock Exchange (NYSE) quotes for a certain stock. Open the files in an editor and figure out the data structure. Then read them into temporary datasets in SAS. Try both ways: use a data step and the proc import procedure. Format the data and label the variables.*

Data Step

`DATA` *library.dataset*;

`SET` *library.dataset*;

...*create, modify, keep or drop variables*...;

`RUN;`

Data Step

Numeric functions that can be used within a data step eg. abs(x), exp(x), lag(x), log(x), sqrt(x)...

If conditions can be used within a datastep:

`IF` *condition* `THEN` *action*;

More than one action to execute:

`IF` *condition* `THEN DO` ;

action 1;

action 2;

`END`;

→ Example in sample program.

Data Step

Task: Use the data sets from the previous task and try some of the functions and if-conditions that can be used in a data step.

Some procedures (Procs)

- Proc SORT: Sorts the data
- Proc MEANS: calculates mean, median, standard error, confidence limits for the mean,...
- Proc CORR: calculates several correlation coefficients
- Proc FREQ: one-way frequency table, n-way contingency tables, several test statistics

Example for Proc MEANS

`Proc MEANS data=`*dataset*;

`var` *variables*;

`by` *variable*;

`output out=` *outdataset* `mean=`*meanvar* ...;

`run`;

→ Example in sample program.

***Task**: Use the SAS help sites to figure out the main syntax of the procedures listed above. Then apply them using one of the datasets you imported in SAS in the previous task.*

Macro Example

A macro can be used to keep elements (variables, libraries, dataset names ect) flexible.

When writing macros as a beginner: Start by programming without a macro first, then make the desired elements flexible and define the macro.

Macro Example

```
%macro mymacro(variable1, variable2);
```

DATA *library.outdataset;*

SET *library.inputdata;*

variablenew=&*variable1*.+&*variable2*.;

RUN;

```
%mend mymacro;
```

A macro can be called with

%mymacro(*inputforvar1, inputforvar2*);

→ Example in sample program.

***Task**: Use one of the data sets imported before and write a macro that calculates and adds the spread (ask-bid) to the existing dataset. Furthermore the macro returns another temporary data set which contains the average spread (use proc means). Keep the variables, the library and the input data set flexible. Call the macro using one of the data sets given above.*

Creating a table with SQL code

```
Proc SQL;
create table name as select
variables
from dataset;
quit;
```

Merging two tables with SQL code

```
Proc SQL;
create table name as select
a.variables
,b.variables
from dataset1 a left/right/full/inner join dataset2 b
on a.idvar1=b.idvar2;
quit;
```