A very brief introduction to SAS

20180828, ANYM, 02409 Multivariate Statistics

SAS is an acronym for Statistical Analysis Software, and is a programming language for statistical data analysis. It is the king of analysis in both big enterprise and the medico/pharmaceutical business.

SAS is organized around four major tasks in the analysis:

- **Data Access**
  - Locates and read the data required
  - This may be from data stored in SAS, in SQL or Oracle databases, Microsoft Excel, plain text, csv, etc.
- **Data Management**
  - Shapes the data into the form we require
- **Data Analysis**
  - Summarize, reduce, test, transform the data as required
- **Data Presentation**
  - Presents the data in tables, plots, figures, etc.

All data in SAS is row-oriented, with each row typically representing an observation. Regarding data types only two exist: numeric and character. Missing data is represented by: '.'

SAS Programs

A SAS program is a series of statements for SAS to execute. Unlike R or Matlab you cannot execute any code or part of a program, but need to run an entire statement.

There are three types of statements:

- DATA statement
  - This does something to the data. E.g. inputting, reshaping or saving data, or reading data stored in SAS.
  - Example that generates a dataset called xample, containing 2 variables (*x1* and *x2*) with 3 observations each.

```
data xample;
input x1 x2; * two variables;
datalines;
 45.9 98.4
 3 56.3
 45.3 -42
;
run;
```

- PROC statement
  - Everything else. From reading external data, to tests, plots and figures
  - Normally a SAS PROC step looks something like this:
    - `proc <procedure-name> data=<dataset-name>;`

    - If one only needs to analyse part of the variables in the SAS dataset, it is (usually) possible to add a line:
        - `var <variable list>;`
- global
    - Global variables

Nearly all statements ends with 'run;'
A very few ends with 'quit;' - that is all the statements that are interactive

Program example

A small program could look like:

```
proc print data=stat2.sundhed;
var alder vegt;
run;
```

To break it down:

proc print *tells SAS to print the data*
data=stat2.sundhed; *tells SAS which dataset to print*
var alder vegt; *tells SAS which variables in the dataset to print*
run; *tells SAS that no more inputs are coming and to execute the PROC statement.*

Notice that every line ends with a semicolon! *This is a common error and can break an entire program.* The only exceptions to the semicolon rule are formats and labels.

To put comments in your code use asterisk:

```
*comment;

/*This
is
a
multiline
comment */
```

Library names

In the small example above we used the dataset 'sundhed' contained in the library 'stat2'.

Each time you input or read data into SAS they will be stored in the library called 'WORK'. This is a temporary library and it is cleared when you close SAS.

You can create custom libraries if you want and store data permanently there, but their names have to conform to a few rules:

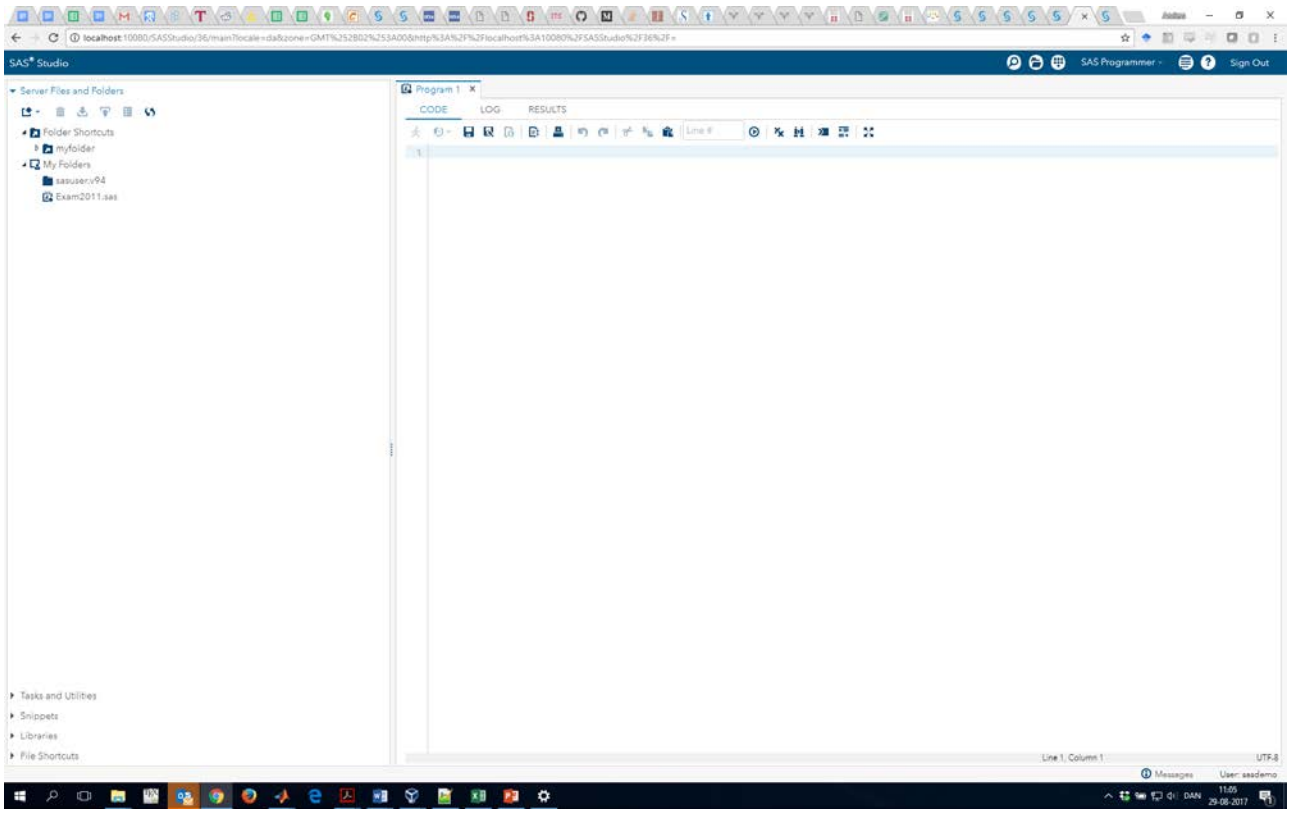
- Limited to eight characters.
- Must begin with a letter or underscore.
- Can contain only letters, numbers, or underscores. Blanks are not allowed.

SAS University Edition

Getting started

You know have a (very) rudimentary understanding of SAS. Let us try to put it to use. If you haven't already installed SAS University Edition start by doing that. There is a guide for it under *Course Material* on DTU Inside.

When starting SAS UE, you will be greeted by a screen that looks something like this:



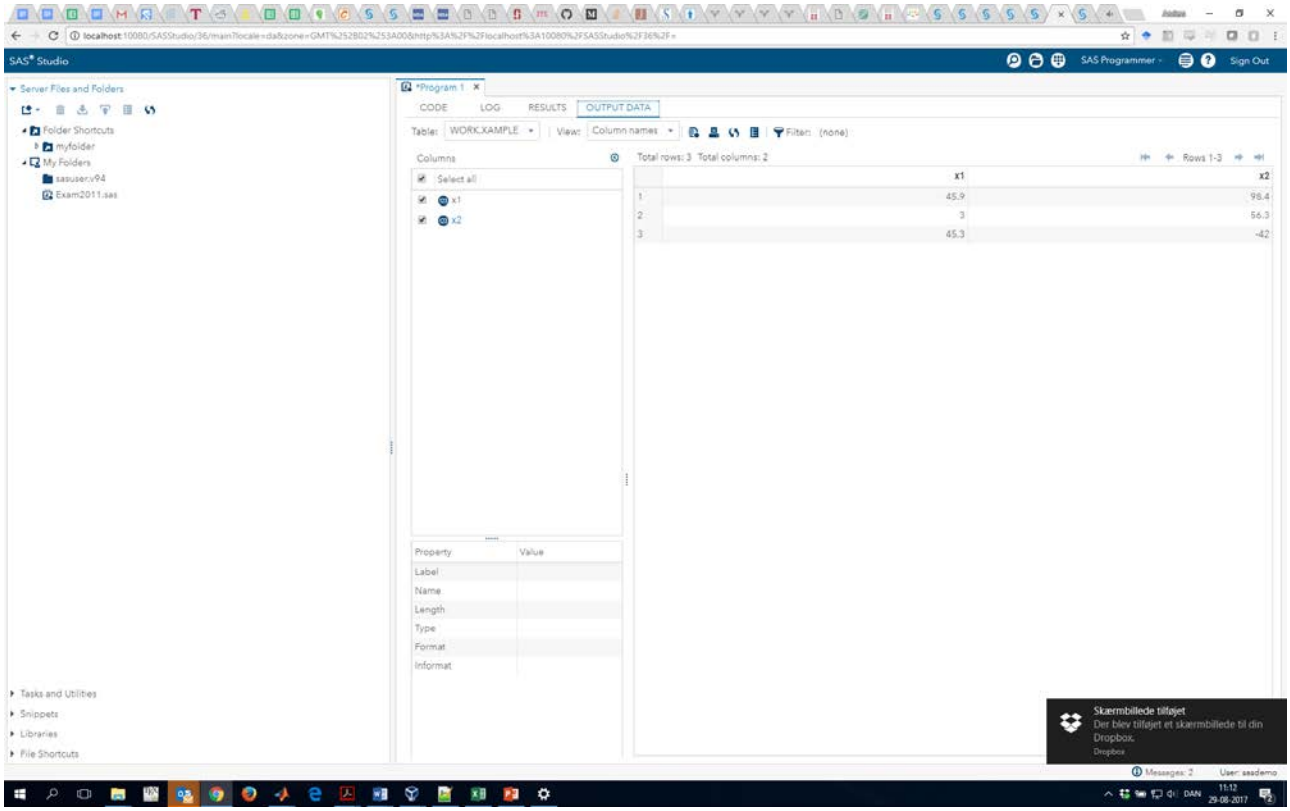
On the left you have 'Server Files and Folders' which shows the myfolder you created during installation.

On the right you have an empty program. Lets us create a small dataset using the example from before:

```
data xample;
input x1 x2; * two variables;
datalines;
45.9 98.4
3 56.3
45.3 -42
;
run;

proc print data=xample;
run;
```

Paste the example into program one and press the little running man just below where it says 'CODE'. The program executes and you a greeted by a new tab called 'OUTPUT DATA', which shows the data created by the program.



The tab 'LOG' will show if there has been any errors during the execution of the program.

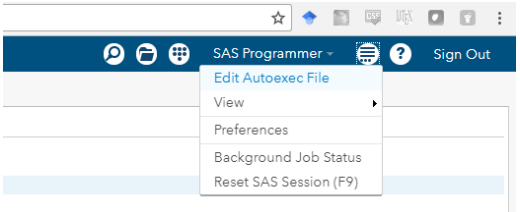
The tab 'RESULTS' will show the output of the program. This is what is created by the 'PROC PRINT' statement in this case.

EXERCISE:

1) Try to modify the 'PROC PRINT' statement so it will only print *x2* (hint see the example on p. 2)
2) Try to delete the last semicolon in the program and run it. Inspect the log to see what happens.
3) Right click on 'print' and select 'Syntax help'. Click on print and see what happens.

Linking libraries

In the course we will not spend a lot of time on reading data into SAS. Instead we will use data in the library stat2. Download the library from DTU Inside, extract it, and put it in your 'myfolder' created during installation.



In the left panel go to 'Server Files and Folders', right click the stat2data folder and select 'Properties' to see the path to the folder.

In the upper right corner click 'Edit Autoexec File' and paste the following lines:

```
libname stat2 '/folders/myfolders/stat2data';          * where stat2's data resides;
libname home '/folders/myfolders';          * for SAS University Edition;'
```

Check that the path matches with one you saw when you selected 'properties'. If not, then correct the paths and save the changes.

The first line adds the stat2data as the library called stat2
The second line creates a new library called home, that you can use for storing data permanently

The autoexec file is run on each startup. This way you do not need to link to the library every time you need it. The same procedure as used in the autoexec file can of course be use in a program to make a library available.

EXERCISE:

1) Restart SAS UE (F9) and test whether you have linked stat2 correctly by

```
proc print data=stat2.sundhed;
var alder vegt;
run;
```

2) Modify **data** xample; to **data** home.xample; Check in the left hand side under 'LIBRARIES' that 'xample' has been written to the library called 'HOME', and under 'SERVER FILES AND FOLDERS' that you can see it as a 'xample.sas7bdat' SAS datafile.

Reading data into SAS

SAS can read basically all data, but it may not be very easy. As mentioned it is not something we will spend a lot of time on in course. You can familiarize yourself with several ways to do it, in the note on DTU Inside under *Course Material* called **SAS Input and Output.pdf**

Some often used procedures.

`proc print` Lists data, nicely formatted

`proc means` Computes mean, variance etc.

`proc univariate` As means but more descriptive. Eg:

```
proc univariate data=stat2.sundhed plot freq normal;
var vegt;
```

`proc sgplot` Plots in a nice graphical format. MANY options. Eg:

```
proc sgplot data=stat2.sundhed;
scatter x=vegt y=alder;
```

`proc corr` Computes correlation matrices, and as an option also the variance-covariance-matrix. Eg:

```
proc corr data=stat2.sundhed cov;
```

`proc princomp` Among other things computes eigenvalues and eigenvectors. Eg.:

```
proc princomp data=xample cov;
```

analyses the covariance matrix based on the observations in the temporary dataset xample. COV indicates we want to analyse the covariance matrix. Exclusion of "COV" means we analyse the correlation matrix.

EXERCISE

1) Play around with the different PROC statements and the data contained in stat2. There is an extensive – and sometimes overwhelming – documentation online. Try to look up some of the different statements to see what they can do.
2) There are several tutorials online, that you can try: https://support.sas.com/en/software/university-edition.html#freetutorials
3) Under 'Tasks and Utilities' in the left hand side, go to 'Tasks'. There you will find a collection of routines that will help you get a quick overview of the data and other tasks. Try to apply some of them to the stat2 datasets.
4) In the pen-and-paper exercise, you calculated the eigenvectors for a dispersion matrix. Use SAS to calculate the dispersion matrix, the eigenvalues and eigenvectors for the `stat2.sundhed;`

```
proc princomp data=stat2.sundhed cov;
     var vegt alder;
run;
```

vegt is the weight, and alder is the age.

5) Plot the data in a scatter plot using SGPLOT.

```
proc sgplot data=stat2.sundhed ASPECT=1;
   scatter x=vegt y=alder;
   xaxis min=55 max=95;
   yaxis min=30 max=70;
run;
```

In your head determine the major and minor axes of the contour ellipse. Compare your intuition to the eigenvectors and values.