AARHUS
UNIVERSITET
BUSINESS AND SOCIAL SCIENCES

SAS Programming – Introduction

SAS Introduction & IML

Analytics Group

See more at www.asb.dk/ag

Analytics Group

Agenda

SAS Introduction (PART 1)

- About SAS and pros/cons
- SAS 9.2 user interface
- Library references
- Proc contents / proc means
- Debugging

SAS IML (PART 2)

- Importing data files
- The Interactive Matrix Language
- Regression(s)
- Saving your matrices to datasets

Practical information

- First a basic introduction to SAS –

  You will need knowledge of basic SAS function, in order to be able to code IML.

- An *introduction* to SAS IML (Interactive Matrix Language), which will be applied later in some Problem Sets in Applied Econometrics Methods. SAS will also be incorporated in Empirical Finance, where it is a necessarily tool, also for the exam

- Duration 3 hours – approximately

What is it?

- At first: SAS - Statictical Analysis System
  - Mainly a tool for statistical analysis of large amounts of data
- Now, it is just SAS and so much more than just statistics
- Used for business intelligence and business analytics within many different industries
  - Risk management
  - Forecasting and econometrics
  - Operations research
  - Supply chain management
  - IT management
  - Management accounting
  - ...so much more

Why learn SAS

- Almost every large company in Denmark (and the rest of the world) uses SAS for their data warehousing.
- Can generate standard reports and lists to be used in the daily, weekly or monthly analysis.
- Is very good at handling large amounts of data without using much space on servers.

Why not?

- Quite expensive
- Steep learning curve

SAS 9.2 User interface

- Three windows
  - Editor: F5
  - Output: F7
  - Log: F6
- Use CTRL + E to clear any of the three windows
- F3 to submit (run) your code
- Two tabs:
  - Results
  - Explorer

Data library

- SAS library references: Highway to data
- Default library references
  - SAS User
  - Work (temporary)
- User defined library references
  - The physical location on your computer/server where you want SAS to retrieve data sets from and store them in.

Library reference

- In general

```
libname name "path";
run;
```

- Examples:

```
libname asb "C:\documents\school\class";
run;
```

```
libname mt "M:\master thesis\datasets";
run;
```

The data set *Brown*

- Consists of seven variables
- Contains a total of 120 observations with monthly sales figures from five different regions from 1948 – 1957
- How?

Proc contents

- Proc contents can be used to get general information about the data set at hand

```
proc contents data=asb.brown;
run;
```

The CONTENTS Procedure

Data Set Name	ASB.BROWN	Observations	120
Member Type	DATA	Variables	7
Engine	V9	Indexes	0
Created	22. oktober 2007 mandag 16:39:53	Observation Length	56
Last Modified	22. oktober 2007 mandag 16:39:53	Deleted Observations	0
Protection		Compressed	NO
Data Set Type		Sorted	YES
Label	Written by SAS		
Data Representation	WINDOWS_32		
Encoding	wlatin1 Western (Windows)		

Engine/Host Dependent Information

Data Set Page Size	4096
Number of Data Set Pages	3
First Data Page	1
Max Obs per Page	72
Obs in First Data Page	42
Number of Data Set Repairs	0
Filename	C:\Users\rusa\Desktop\SAS_IML_AEM\DATA\brown.sas7bdat
Release Created	9.0101M3
Host Created	XP_PRO

Alphabetic List of Variables and Attributes

#	Variable	Type	Len
6	MONTH	Num	8
1	SALES1	Num	8
2	SALES2	Num	8
3	SALES3	Num	8
4	SALES4	Num	8
5	SALES5	Num	8
7	YEAR	Num	8

Sort Information

Sortedby	YEAR
Validated	YES
Character Set	ANSI

Proc Means

- The Means procedure computes descriptive statistics about one or more variables.

```
proc means data=asb.brown;
var sales1;
run;
```

Output

The SAS System 19:54 Sunday, November 6, 2011 2

The MEANS Procedure

Analysis Variable : SALES1

N	Mean	Std Dev	Minimum	Maximum
120	100.7250000	23.4459731	35.0000000	167.0000000

Proc Means

- You can get confidence intervals for a variable by adding *cml* at the end of the Proc Means statement. The default level is 95%

```
proc means data=asb.brown clm;
var sales1;
run;
```

Output

The SAS System 19

The MEANS Procedure

Analysis Variable : SALES1

Lower 95% CL for Mean	Upper 95% CL for Mean
96.4869631	104.9630369

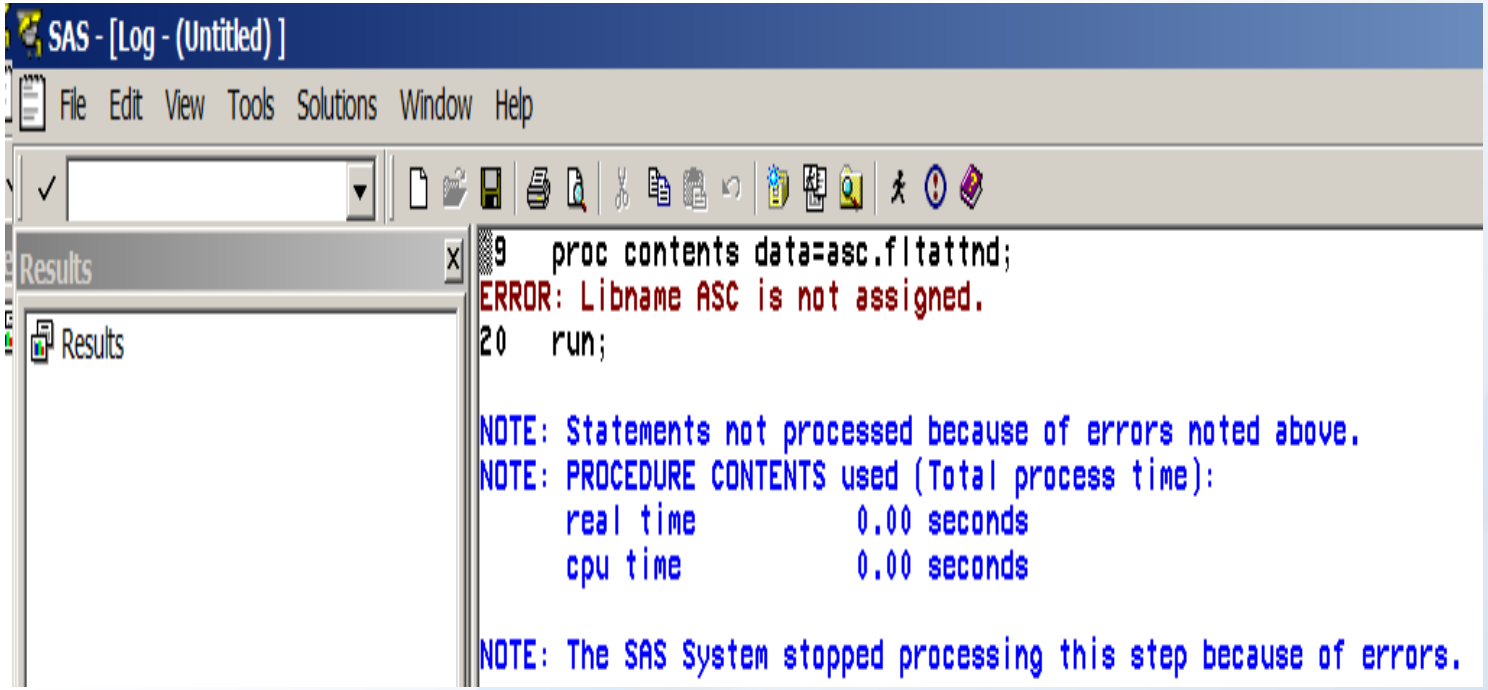
Debugging techniques

- The SAS log is very useful if problems occur when running your code.

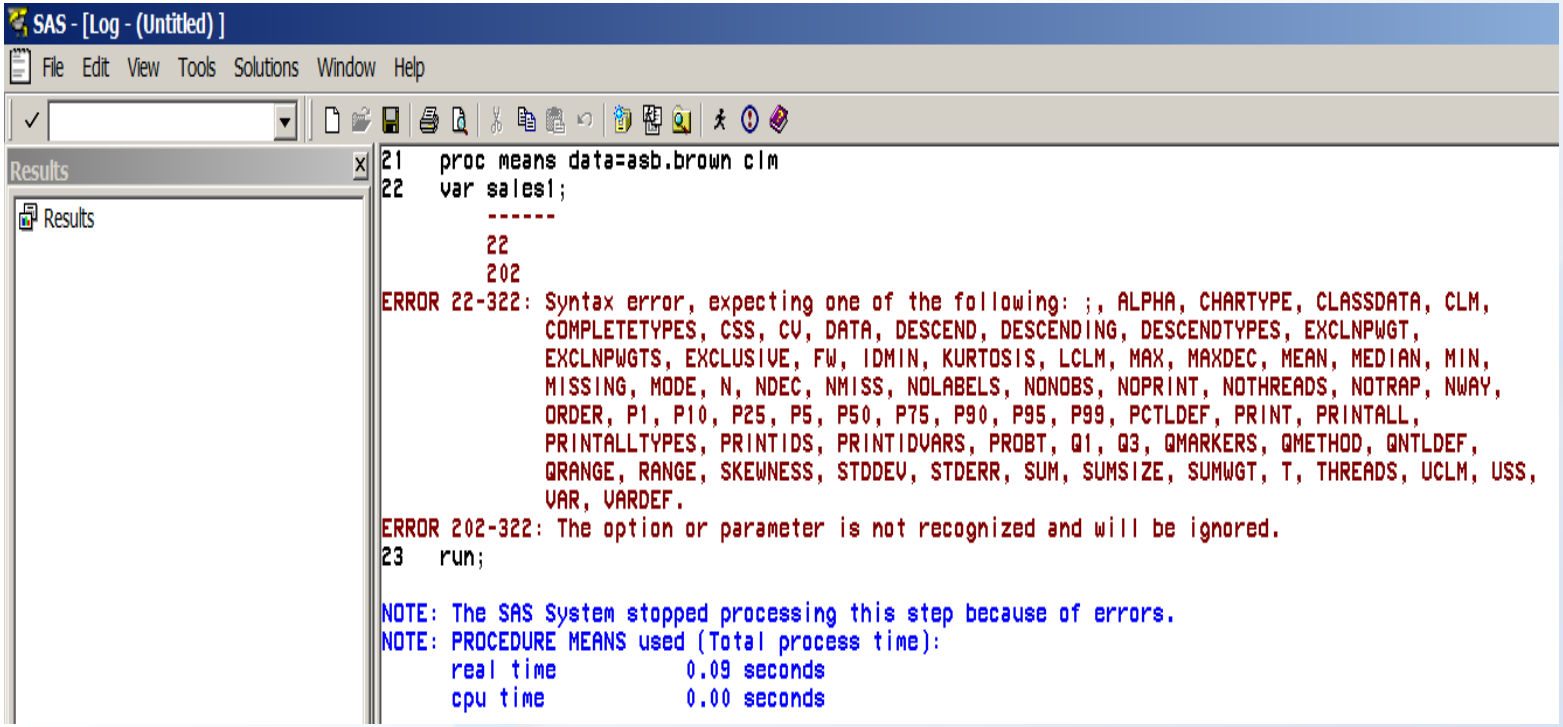
- You should ALWAYS check the log after submitting your code, and even before you check your output. Just because output has been generated does not mean your code has been submitted without errors. Also, your output results could actually be incorrect and If you do not check the log you might not notice incorrect results!

- Red = Bad
- Green = Warnings
- Blue = Good
- Black = The lines you submitted

Libname not assigned



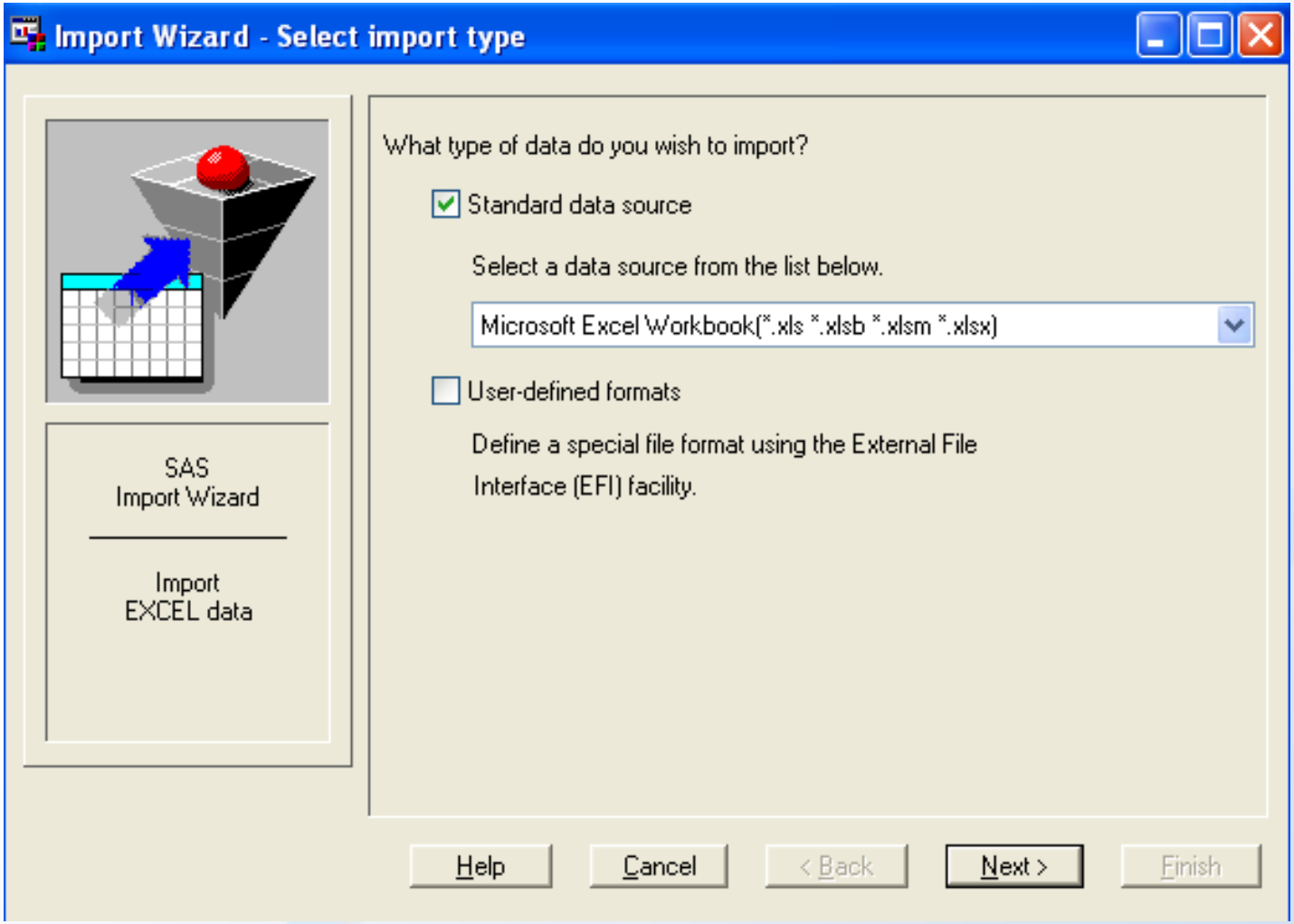
Missing ; Very common mistake



Importing data from .csv .txt .xls

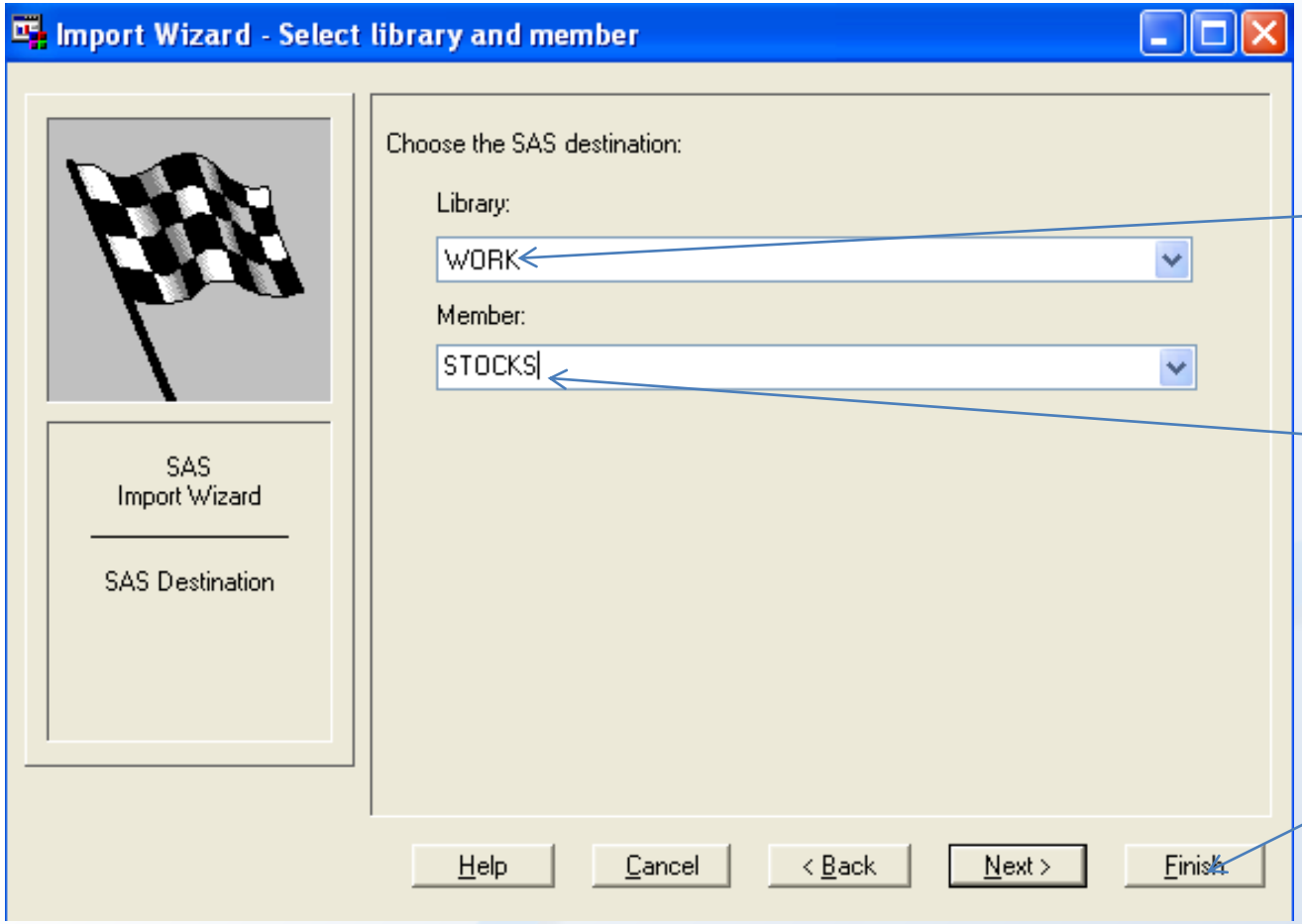
- Importing via wizard (File=> Import Data..)
- Importing via programming

Importing via wizard



… And browse to the specific file…..

Importing via wizard



In which library do you want to import your file?

State a name of your imported file

Click finish

Importing via wizard

Be aware that there are some issues when trying to import files if you have installed the SAS 64-bit engine. Here you would have some problems importing files using the wizard. Just import the files using coding instead.

Importing from Excel (coding)

```
proc import out= test
datafile= "C:\Documents and Settings\rusa\Desktop\test1.xls"
dbms = xls  replace;
getnames=yes;
run;
```

SAS/IML

Interactive Matrix Language

Coding technique

```
proc iml;
..
..
..
Your code
..
..
..

Quit;
```

Producing a scalar

```
proc iml;
A=2;
print A;
Quit;
```

Produces $A = 2$

Producing matrixes

```
proc iml;
X={2 3, 4 5, 6 7};
print X;
Quit;
```

Produces

$$\mathbf{x} = \begin{bmatrix} 2 & 3 \\ 4 & 5 \\ 6 & 7 \end{bmatrix}$$

Producing matrixes

- Producing an Identity matrix

```
proc iml;
Identity=I(3);
print Identity;
Quit;
```

Produces

$$Identity = \begin{matrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{matrix}$$

Produce row and column vectors

- 1x5 (row vector) with numbers from 1 to 5

```
proc iml;
row=1:5;
print row;
quit;
```

Produces: Row = [1 2 3 4 5]

Produce row and column vectors

- 6x1 (column vector) with numbers from 3 to 8:

```
proc iml;
col = t(3:8);
print col;
Quit;
```

Produces: Row =

$$\begin{bmatrix} 3 \\ 4 \\ 5 \\ 6 \\ 7 \\ 8 \end{bmatrix}$$

The J-function

- Can be used to create constant matrixes.

```
a=j(5,6,0);
```

Produces:

A

0	0	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0

Selecting a specific row

- If you wish to select the 4th row and the 2nd column from matrix A

```
Proc IML;
fourthrow=A[4,];
secondcol=A[,2];
Quit;
```

Selecting a specific element

- If you wish to select a scalar - the 4,7 th element in matrix A

```
Proc iml;
Element=A[4,7];
Quit;
```

Transpose a matrix

- If you need to transpose matrix A.

```
Proc IML;
Transpose=t(A);
Print Transpose;
Quit;

or

Proc IML;
TransposeA=A`;
Print TransposeA;
Quit;
```

Addition

- If you need to compute matrix C, which equals A+B.

```
Proc IML;
C=A+B;
Print C;
Quit;
```

Multiplication

Multiplication of matrix A and B, assuming that dimensions matches. (Ex.: 4x5 * 5x4 = 4x4)

```
Proc IML;
AB=A*B;
Print AB;
Quit;
```

Elementwise multiplication

- If you need to multiplicate Matrix A and B elementwise.

```
Proc IML;
AB=A#B;
Print AB;
Quit;
```

Squaring the elements

- If you need to raise all the elements in A to the second power

```
Proc IML;
Ap_two=A##2;
Print Ap_two;
Quit;
```

Inverse matrix

- Inverse matrix of C – must be quadratic

```
Proc iml;
Cinv= Inv(c);
Print Cinv;
Quit;
```

The horizontal concatenation operator

The concatenation operator '||' produces a new matrix by joining *matrix1* and *matrix2* . The two matrices must have the same number of rows, which is also the number of rows In the new matrix.

The following concatenates two 3x2 matrices and produces a 3x4 matrix.

```
proc iml;
x={1 2, 3 4, 5 6};
y={7 8, 9 10, 11 12};
X_and_Y=X||Y;
print x_and_y;
quit;
```

$$X = \begin{bmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{bmatrix} \quad Y = \begin{bmatrix} 7 & 8 \\ 9 & 10 \\ 11 & 12 \end{bmatrix}$$

$$X\_and\_Y = \begin{bmatrix} 1 & 2 & 7 & 8 \\ 3 & 4 & 9 & 10 \\ 5 & 6 & 11 & 12 \end{bmatrix}$$

The vertical concatenation operator: //

$$X = [1 \quad 2] \quad Y = \begin{bmatrix} 4 & 5 \\ 5 & 3 \end{bmatrix}$$

$$Y \; and \; X = \begin{bmatrix} 4 & 5 \\ 5 & 3 \\ 1 & 2 \end{bmatrix}$$

Other usefull matrix operations (self study)

```
/* Sum down the rows and makes a vector with dimension 1 X columns*/
totals=a[+,];

/* Sum across the columns and makes a vector with dimension rows X 1 */
totals=a[,+];

/*Sums the second row*/
Sum_row=sum(A[2,]);

/* Multiply down the rows */
product=a[#,];

/* Find the largest column average (average down the rows)*/
means=a[:,];
answer1=means[,<>];

/* Find the smallest row average (average across the columns) */
means=a[,:];
answer1=means[><,];

Compute the sum of squares of the second row
ASumssq=SSQ(A[2,]);
```

Assigment 1

Problem 1)
In order to get experience with the different matrix operating procedures, do the following:

- Create a 3x3 matrix (name: **seven**) with all 7's.
- Create a column vector (name: **column**) with the numbers from 5 to 15
- Create a matrix (name: **three**) with the following values

$$\begin{pmatrix} 5 & 7 & 1 \\ 2 & 7 & 8 \\ 20 & 21 & 4 \end{pmatrix}$$

- Find the **inverse** matrix to **three** (name: **Inv_three**).
- **Multiply seven** with **three**. (name: **sevenxthree**) (Try to multiply three with column and see what the log says)
- **Multiply seven** and **three elementwise**. (name: **sevenxxthree**)
- Pick out the **2nd row** of **three** and call it **row**. Then pick out the **3rd column** of **seven** and call it **column**. **Multiply column and row.** (name: **rowxcolumn**)
- **Sum** across the **columns of three**. (name: **sum_col_three**)

AARHUS
UNIVERSITET
BUSINESS AND SOCIAL SCIENCES

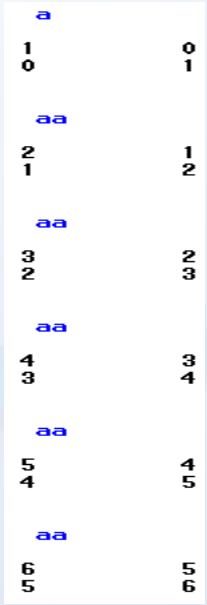
Loops

- Very useful in financial context and for simulation purposes

Example

```
proc iml;
a=I(2);
print a;
  do i=1 to 5 by 1;
  aa=a+i;
  print aa ;
End;
quit;
```

produces:



Analytics Group

Creating a regression using IML

$$\hat{\beta} = (X`X)^{-1} X'Y$$

Beta = inv(X`*X)*X`*Y

Create a matrix based on a SAS Data Set

- You can create a matrix using a SAS Data Set

```
proc iml;
use sasuser.course;
read all into x ;
print x;
quit;
```

Will create a matrix with all variable from the dataset.

Create several matrixes (colum vectors)

- If you need each variable to be created as a 1xn matrix.

```
proc iml;
use sasuser.kursus;
read all var {var1} into x ;
read all var {var2} into y;
print x;
Print y;
quit;
```

- Will produce 2 column vectors from variable 1 and 2

Example – creating a regression module

- The file stocks.xls contains, among other things, the log-returns of OMXC20 and Carlsberg.
- It is the objective to calculate the Beta of Carlsberg.
- This is done i two steps.
  - Importing the .xls file into two column vectors.
  - Creating a regression module on the basis of these vectors.

Example – creating a regression module

Step 1

```
proc import out=stocks
datafile= "C:\Documents and Settings\rusa\Desktop\stocks.xls"
dbms = xls  replace;
getnames=yes;
run;
```

Step 2

```
proc iml;
use stocks;
read all var{c20} into X;
read all var{Carlsberg} into Y;
Beta= inv(X`* X)* X`* Y ;
print Beta;
quit;
```

Produces: **Beta** = 0.69796

Assignment 2

- In the workbook stocks.xls you will find the log-returns on 3 stocks and 1 index; OMX C20, Carlsberg, SAS and TDC. There are 261 returns.

- Import the data into SAS and create a regression module calculating beta for each of the three stocks. (with no use of loops)

- The results should be: Carlsberg: 0,6979644; SAS: 1,3397; TDC: 0,1012995

Estimating many betas

- If you have more than one stock it is a lot of work to estimate the betas the way you learned before.
- In stead you can use the power of do-loops.
- In this example we will estimate both the beta and the constant in the market model.
- It is still done in two steps.
  - Import the data into a matrix.
  - Calculate the estimates.

Step 1 – import the data to *ONE* matrix

Beta contains the following variables:

- Log-Market return
- The log-return from stock 1 to 5

```
PROC IML;
USE asb.beta;
READ all var _num_ into return;
print return;
```

Step 2 – Calculate the estimates of betas and the constants.

```
N=nrow(return);

beta=J(5,2,0);

X=J(N,1,1)||return[,1];

 do i=1 to 5 by 1;
      y=return[,1+i];
      beta[i,]=(INV(X`*X)*X`*y)`;
end;

print beta;
quit;
```

The number of observations in the dataset

Create a matrix with zero's to hold the estimated parameters.

Create the matrix X (100x1 with one's) and add the second column of the return matrix which produces a 100x2 matrix with one's and the market return.

Selects the 3rd to 7th column in the return (the returns of the 5 stocks as Y).

Regresses each of the 5 stock returns (Y) on the market return (X), and outputs it in the matrix beta in row i.

Loops when i= 1 to 5

IML - Operators

Operator	Description
` (back tick)	Transpose (postfix)
- ***(prefix)***	Negative prefix
[ ]	Subscript
**	Matrix exponentiation
##	Element-wise exponentiation
*	Matrix multiplication
#	Element-wise multiplication
/	Element-wise division
@	Direct (Kronecker) product
+	Addition
-	Subtraction
\| \| | Horizontal concatenation |
// | Vertical concatenation |
<> | Maximum |
>< | Minimum |
< | Less than |
<= | Less than or equal to |
= | Equal to |
^= | Not equal to |
> | Greater than |
>= | Greater than or equal to |

Creating a dataset from a matrix

```
create <data_set_name> from <Matrix_name>[colname={"col_name" "col_name"}];
append from <Matrix_name>;
```

```
Create regdata from beta[colname={"alpha" "beta"}];
Append from beta;
```

You can now use the export function in SAS to view the data in Excel. (Do not use the View function)

Assignment 3

- Create a module to estimate the market model (both the betas and the constant) for the three stocks in your imported excel file stocks.xls. This time do it using a do-loop, so you easy can alter the code to calculate the market model for 100 stocks.

- Create a dataset containing the alphas and betas from the regression in question 3) and export the data to Excel.

Where to learn more?

- Borrow *The Little SAS Book at the library.*
- F1 – very useful. Place the cursor on a statement, frase or anything in your code, and press F1.
- Try to Google your question.
- The Analytics Group are at the moment looking into making a course in advanced SAS IML for use in Empirical Finance. Courses will be available during the end of january