

# SAS/IML Introduction

By the  
Analytics Group

# Agenda

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

# Importing data from .csv .txt .xls

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

# Importing a delimited raw data file

- Import is executed in a Data step.
- Importing a raw data file – example:

```
Data <data-set-name>;  
infile '<raw-data-file>' dlm='delimiter(s)';  
input variable 1 : variable 2 : variable n;  
run;
```

# Example

```
data test;  
infile 'C:\Documents and Settings\rix\Desktop\kursus.csv' dlm=';' ;  
input revenue : employee ;  
run;
```

# Importing from Excel

Importing from Excel leaves you with several options:

- *Replace* – will overwrite existing dataset
- *Sheet*='sheetname' You can define the sheet(s) that you need to import.
- *Getnames*=yes/no You can use the first cell in each column as the variable name
- *Mixed*=yes/no – specifies whether there are both character and numeric observations within certain variables.
- *Usedate* = yes/no – are there any dates in the worksheet?

# Importing from Excel

```
proc import out=work.stocks  
datafile='C:\Documents and Settings\rix\Desktop\stock.xls' replace;  
sheet='sheet1';  
Getnames=yes;  
Mixed=no;  
usedate=yes;  
run;
```

# SAS/IML

Interactive Matrix Language



# 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*** =  $\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$

# 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;  
4throw=A[4,];  
2ndcol=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.:  $4 \times 5 * 5 \times 4 = 4 \times 4$ )

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

# Elementwise multiplication

- If you need to multiply 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: //

# Other useful matrix operations (self study)

- `/* Sum down the rows */`
- `totals=a[+,:];`
- `/* Sum across the columns */`
- `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[:, < >];`
- `answer2=a[:, < >];`
- `/* Find the smallest row average (average across the columns) */`
- `means=a[:, :];`
- `answer1=means[> < , :];`
- `answer2=a[:, :][> < , :];`
- `Compute the sum of squares of the second row`
- `ASumssq=SSQ(A[2,:]);`



# Assignment 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.
- Multiply seven with three. (Try to multiply three with column and see what the log says)
- Multiply seven and three elementwise.
- Pick out the 2<sup>nd</sup> row of three and call it row. Then pick out the 3<sup>rd</sup> column of seven and call it column. Multiply column and row.
- Sum across the columns of three.
-

# Loops

- Very useful in financial context and for simulation purposes

## Example

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

produces:

a		
1	0	0
0		1
aa		
2		1
1		2
aa		
3		2
2		3
aa		
4		3
3		4
aa		
5		4
4		5
aa		
6		5
5		6

# 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.kursus;  
read all into x ;  
print x;  
quit;
```

Will create a matrix with all variable from the dataset.

# Create several matrixes (column 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.csv contains the log-returns of OMX C20 and Carlsberg.
- It is the objective to calculate the Beta of Carlsberg.
- This is done i two steps.
  - Importing the .csv file into two column vectors.
  - Creating a regression module on the basis of these vectors.

# Example – creating a regression module

## Step 1

```
data stocks;  
infile 'C:\data\stocks.csv' dlm=';' firstobs=2;  
input c20 : Carlsberg;  
run;
```

## Step 2

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

# 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.
- Instead 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\_est contains the following variables:

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

```
PROC IML;  
USE beta_est;  
READ all var _num_ into return;  
print return;
```

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

```
N=nrow(return);
```

The number of observations in the dataset

```
beta=J(5,2,0);
```

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

```
X=J(N,1,1)||return[,2];
```

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.

```
do i=1 to 5 by 1;
```

```
  y=return[,2+i];
```

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

```
  beta[i,]=(INV(X'*X)*X'*y)';
```

```
end;
```

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

```
print beta;
```

```
quit;
```

Loops when i= 1 to 5

## IML - Operators

Operator	Description
` (accent grave)	Transpose (postfix)
- ( <i>prefix</i> )	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)

# 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.

# Assignments

## 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 a SAS and create a regression module calculating beta for each of the three stocks.

## Assignment 3)

Create a module to estimate the market model (both the betas and the constant) for the three stocks. 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.