XploRe Course - Day 1

Uwe Ziegenhagen
Sigbert Klinke

Institut für Statistik and Ökonometrie
Humboldt-Universität zu Berlin
http://ise.wiwi.hu-berlin.de

HUMBOLDT-UNIVERSITÄT ZU BERLIN

C.A.S.E.
CENTER FOR APPLIED STATISTICS AND ECONOMICS

Outline of the Course

- ⊡ Day 1 (Uwe Ziegenhagen)
  - ▸ Introduction
  - ▸ Matrices and Operators
- ⊡ Day 2 (Sigbert Klinke)
  - ▸ Descriptive Statistics
  - ▸ Graphics
- ⊡ Day 3 (Sigbert Klinke)
  - ▸ Graphics

Outline of the Course

⊡ Day 4 (Uwe Ziegenhagen)

  ▶ Programming

⊡ Day 5 (Sigbert Klinke)

  ▶ Data Analysis

Introduction

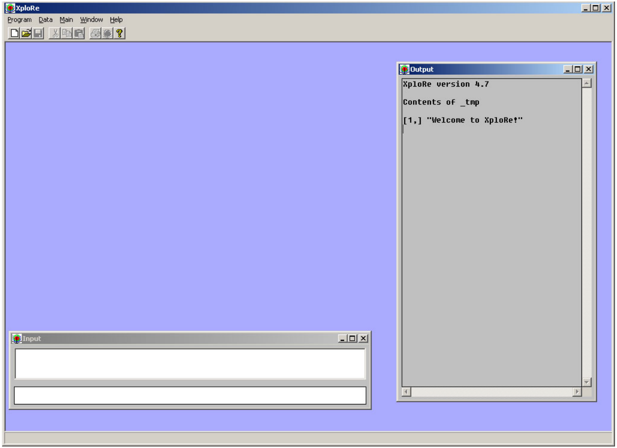
XploRe

- is a computational environment for data analysis and statistics
- has large and extendable set of statistical methods
- is a procedural language, the user writes procedures or functions
- allows Dynamic link calls (DLL)
- is available for Windows, Linux and Solaris
- and for JAVA enabled browsers

XploRe Structure

- XploRe is an interpreted procedural programming language
- built-in commands of XploRe are referred as (internal) functions
- all numbers are floats, there are no integers in XploRe
- program source is structured into procedures, called quantlets
- a quantlet is a sequence of commands with assigned name and a defined interface
- quantlets are organized in quantlibs, loaded by `library` command
- example: `library("plot")`

Graphical User Interface



```
XploRe version 4.7

Contents of _tmp

[1,] "Welcome to XploRe!"
```

Graphical User Interface

Program opens a new or existing quantlet with Program ⇒ New or Program ⇒ Open Data, loads data sets with Data ⇒ Open

Main gives information on objects, functions and quantlets

Window arranges or activates windows

Help starts the Auto Pilot Support System (APSS)

Menus are sensitive to the selected window!

Editor Window

Edit	undo, copy & paste, complete line, insert path
Search	search and replace text in current file
Execute	run current file (Alt-e)
Tools	format source and insert APSS templates

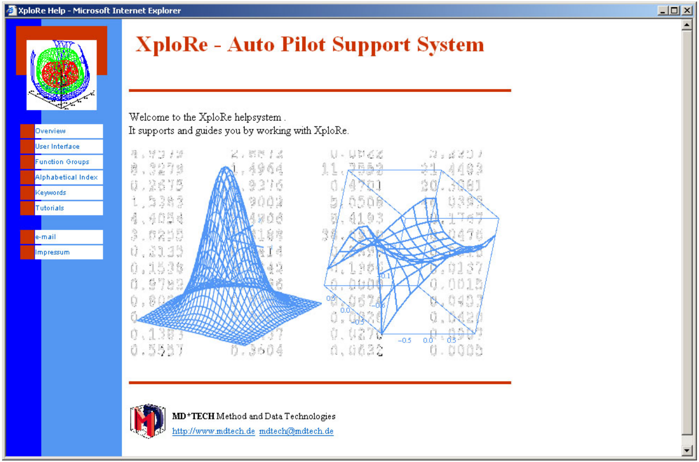
XploRe Directory Structure

data	variety of datasets, see www.quantlet.org/mdbase
dll	dynamic link libraries, connectors to C/C++
examples	examples from the different books
help	APSS
lib	all quantlets
tutorials	tutorials on selected topics

The getenv() command

```
[ 1,] "system" "i686-pc-cygwin32"
[ 2,] "os" "windows"
[ 3,] "build" "88"
[ 4,] "builddate" "Apr 27 2005"
[ 5,] "buildtype" "standalone"
[ 6,] "outheadline" "\r\nContents of %s\r\n\r\n"
[ 7,] "outlayerline" "[,,%li,%li,%li,%li,%li,%li]\r\n"
[ 8,] "outlineno" "[%*li,] "
[ 9,] "outmaxdata" "2048"
[10,] "outputformat" "% 8.5g"
[11,] "outputstringformat" ""%s""
[12,] "startup" "C:\\Programme\\MDTech\\XploRe\\startup.xpl"
[13,] "logfile" "C:\\Programme\\MDTech\\XploRe\\xplore.log"
[14,] "machineeps" "2.220446049250313e-16"
[15,] "statusmessage" "on"
```

The APSS Help System



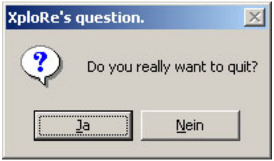
XploRe Help - Microsoft Internet Explorer

XploRe - Auto Pilot Support System

Welcome to the XploRe helpsystem .
It supports and guides you by working with XploRe.

MD*TECH Method and Data Technologies
http://www.mdtech.de mdtech@mdtech.de

Important!



XploRe's question.

Do you really want to quit?

XploRe asks only once, if files are not saved they are lost!

Types of Variables

Variables can be define as numbers and character sequences with the following dimensions:

1. scalars
2. vectors (one-dimensional objects)
3. matrices and arrays
4. lists of objects

Basic Operators

+	addition
-	substraction
*	multiplication
/	division
^	exponentiation

Precedence rules:

1. ^
2. * and /
3. + and -

Comments

;	one line comment
//	one line comment
/**/	multi-line comment

Boolean Operators

< is smaller

<= is smaller or equal

> is bigger

> is bigger or equal

<> is unequal

== is equal

&& elementwise logical AND

|| elementwise logical OR

!x elementwise logical NOT

Mathematical Functions

abs computes the absolute values of the elements of an array.

rint gives the next nearest integer value of the elements of an array.

ceil returns the smallest integer value greater or equal to each element of an array.

floor gives the next smaller integer value of the elements of an array.

sqrt computes the square root of the elements of an array.

plus various trigonometric functions: sin, cos, tan, etc.

Variables

- results of numeric computations are lost if not assigned to a variable
- assignment operator '$=$'
- assignment by value, not by reference

by value

```
a=2
b=a
a=3
b ; result is 2
```

by reference

```
a=2
b=a
a=3
b ; result is 3
```

Variable Names

- strings of alphabetic characters: a, b abc, a1, a123
- sequence always alphabetic => numeric
- not allowed: _ and ␣
- names are case sensitive 'a123' is not equal to 'A123'
- `pi` and `eh` are constants, cannot be used as variable names

Vectors - Column Vectors

```
x = #(1,2,3)
```

generates a column vector $\begin{bmatrix}1\\2\\3\end{bmatrix}$

Vectors II - Row Vectors

```
x = #(1,2,3)'
```

transposes the column vector to $\begin{bmatrix}1 & 2 & 3\end{bmatrix}$

Vectors III - Columnwise Concatenation

```
a = #(1,2,3)
b = #(4,5,6)
x=a~b
x
```

```
Contents of x

[1,]        1        4
[2,]        2        5
[3,]        3        6
```

Vectors III - Rowwise Concatenation

```
a = #(1,2,3)'
b = #(4,5,6)'
x=a|b
x
```

```
Contents of x

[1,]        1        2        3
[2,]        4        5        6
```

Vectors IV - Alternatives

```
a = #(1,2,3)
b = 1|2|3
```

both generate the column vector $\begin{bmatrix}1\\2\\3\end{bmatrix}$

aseq(start,length,step computes an additive sequence

mseq(start,length,step computes a multiplicative sequence

```
aseq(2,4,0.25)
```

Matrices

```
m = #(1,2,3)~#(4,5,6)~#(7,8,9)
m
```

$$\begin{bmatrix} 1 & 4 & 7 \\ 2 & 5 & 8 \\ 3 & 6 & 9 \end{bmatrix}$$

```
textmat = #("aa","BB")~#("CC","dd")
textmat
```

$$\begin{bmatrix} "aa" & "CC" \\ "BB" & "dd" \end{bmatrix}$$

Numeric and text matrices cannot be mixed!

Matrix Generating Functions

unit(d) generates a $d \times d$ matrix with 1 on the diagonals

diag(start:end) generates a $d \times d$ matrix with d = end-start

matrix(row,col) generates a row × colum matrix of ones

zeros(row,col) generates a row × colum matrix of zeros

Arrays

Arrays can have up to eight dimensions (rarely used)

```
z = matrix(2,2,2)
z
```

```
[,,1,1,1,1,1,1]
[1,]          1          1
[2,]          1          1

[,,2,1,1,1,1,1]
[1,]          1          1
[2,]          1          1
```

Stacking Arrays

```
x=#(1:4)~#(5:8)
y=#(11:14)~#(15:18)
stack(x,y)
```

```
Contents of z

[,,1,1,1,1,1,1]
[1,]          1         5
[2,]          2         6
[3,]          3         7
[4,]          4         8

[,,2,1,1,1,1,1]
[1,]         11        15
[2,]         12        16
[3,]         13        17
[4,]         14        18
```

Matrix Functions

dim(x)	shows the dimension of an array x
rows(x)	shows the number of rows
cols(x)	shows the number of columns

Matrix Extraction Functions

```
x[i,j] ; extracts the i-th row and j-th
; column of a matrix

x[1,] ; extracts the 1st row and all columns
x[,1] ; extracts the 1st column and all rows

x[1:3,1:3] ; extracts the 1st, 2nd
;and 3rd row and columns
```

Matrix Extraction Functions

```
; create a 10x10 matrix
; extract the 1st, 3rd, 5th, 7th and
; 9th row and column
data=matrix(10,10)

r=aseq(1,5,2) ; or r=1|3|5|7|9
c=r ;

data[r,c] ; or data[r,r]
; equivalent: data[aseq(1,5,2),aseq(1,5,2)]
```

Various Matrix Functions

isInf(x) determines whether elements of x are infinite values

isNaN(x) determines whether elements of x are missing values

paf(x,i) deletes all rows in x where corresponding elements in i equal 0

countNaN(x) counts missing values in array x

isNumber(x) determines whether elements of x are regular numbers

Matrix Extraction Functions

```
x=normal(10,10)
paf(x, x[,1]<0) ; deletes all rows
; where the corresponding element in the
; first column is larger than 0
```

```
data=normal(10,10) ; create data
data
data=paf(data,data[,1]<0) ; kill all rows of data
; where data[,1]>0
data
paf(data,data[,2]<0)
; kill rows where data[,2]>0
```

Various Matrix Functions

countNotNumber(x) counts missing and infinite values

replace(haystack,needle,replace) replaces in 'haystack' all 'needles' with 'replace'

sort(x,c) sorts x according to column c in ascending, with -c in descending order

inv(x) computes the inverse of a matrix x

sum(x) computes the sum of the elements of an array

cumsum(x) cumsum computes the cumulative sum of the elements of an array

Lists

Lists are containers for other object, e.g. three matrices can be put into one list.

list(x1,x2,x3) generates lists from given objects

names(L) gives the names of all components of a list L

append (L,x) append object x to list L

delete(L,pos) deletes element nr. pos in list L

insert(L,pos,x) insert object x at position pos in list L

L{i} gives the i-th element in list L

Matrix Extraction Functions

```
a = normal(10,10); generate some objects
b = normal(12,6);
c = uniform(5,5);
L = list(a,b,c); create a list with 3 elements
names(L) ; give name vector of all elements in L
L.a ; returns a
delete(L,1) ; delete 1st element in L
```

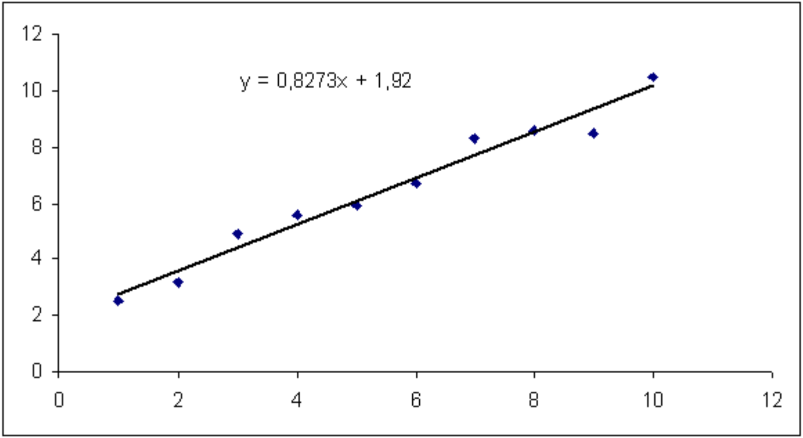
Linear Regression

$$\beta_1 = \frac{\mathrm{Cov}(x, y)}{\mathrm{Var}(x)}$$

$$\beta_0 = \bar{y} - b\bar{x}$$

x	1	2	3	4	5	6	7	8	9	10
y	2.5	3.2	4.9	5.6	5.9	6.7	8.3	8.6	8.5	10.5

Linear Regression



Linear Regression

$$\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix} \beta_0 + \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} \beta_1 + \begin{bmatrix} e_1 \\ e_2 \\ \vdots \\ e_n \end{bmatrix}$$

$$X = \begin{bmatrix} 1 & x_1 \\ 1 & x_2 \\ \vdots & \vdots \\ 1 & x_n \end{bmatrix} \quad \text{and} \quad \beta = \begin{bmatrix} \beta_0 \\ \beta_1 \end{bmatrix}$$

Linear Regression

$$\widehat{\beta} = (X'X)^{-1} * X'y$$

x	1	2	3	4	5	6	7	8	9	10
y	2.5	3.2	4.9	5.6	5.9	6.7	8.3	8.6	8.5	10.5

For Further Reading

- W. Härdle, S. Klinke and M. Müller
  XploRe Learning Guide
  Springer, 2000

- P. Cizek and S. Klinke
  XploRe Introductory Course
  www.quantlet.com/mdstat/scripts/xic/java

- W. Härdle, Z. Hlavka and S. Klinke
  XploRe Applications Guide
  Springer, 2000