UPPSALA
UNIVERSITET

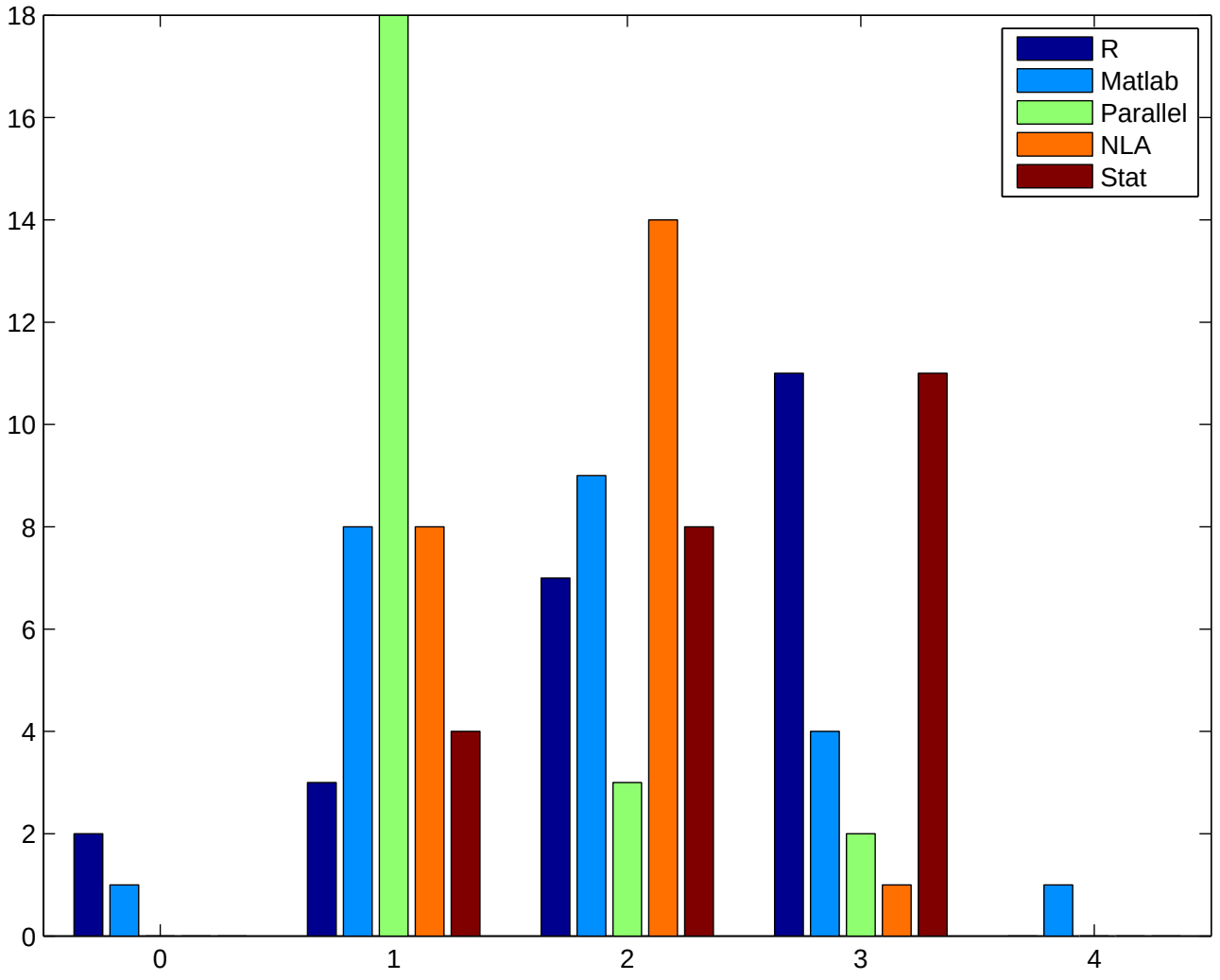
Computational Methods for Statistics with Applications

Maya Neytcheva

Department of Information Technology

Uppsala University

NGSSC
FORSKARSKOLA I TEKNISK-VETENSKAPLIGA BERÄKNINGAR
NATIONAL GRADUATE SCHOOL IN SCIENTIFIC COMPUTING


18
16
14
12
10
8
6
4
2
0
0
1
2
3
4
R
Matlab
Parallel
NLA
Stat

UPPSALA
UNIVERSITET

Introduction to $\mathbb{R}$ and Matlab

UPPSALA
UNIVERSITET

Plan of the lecture:

- What are $\mathbb{R}$ and Matlab?
  - History
  - Computation
  - Visualization
  - Programming
- $\mathbb{R}$ /Matlab basics
  - Handling data - importing, exporting, creating
  - Distributions
  - Plotting stuff
  - Package handling
  - Examples

UPPSALA
UNIVERSITET

What is $\mathbb{R}$

- An object-oriented language and programming environment for statistical computing and graphics
- Developed via the $\mathbb{R}$ project for Statistical computing
- Based on the S system, developed by Bell Labs
- Supported by a large user network (packages)
- Available as a free software

UPPSALA
UNIVERSITET

What is $\mathbb{R}$ not

- a statistical package
- very quick to learn in details
- menu-driven

UPPSALA
UNIVERSITET

The $\mathbb{R}$ environment (suite)

- efficient data handling and storage
- operates on arrays and spreadsheets
- provides a collection of tools for statistical analysis and numerical methods
- flexible graphical display tools
- a programming language
- add-on capabilities (packages/libraries) for creating, testing and distribution of software
- options to integrate with other languages (C, XML, Java)
- tools for producing documentation and training materials (vignettes)

UPPSALA UNIVERSITET

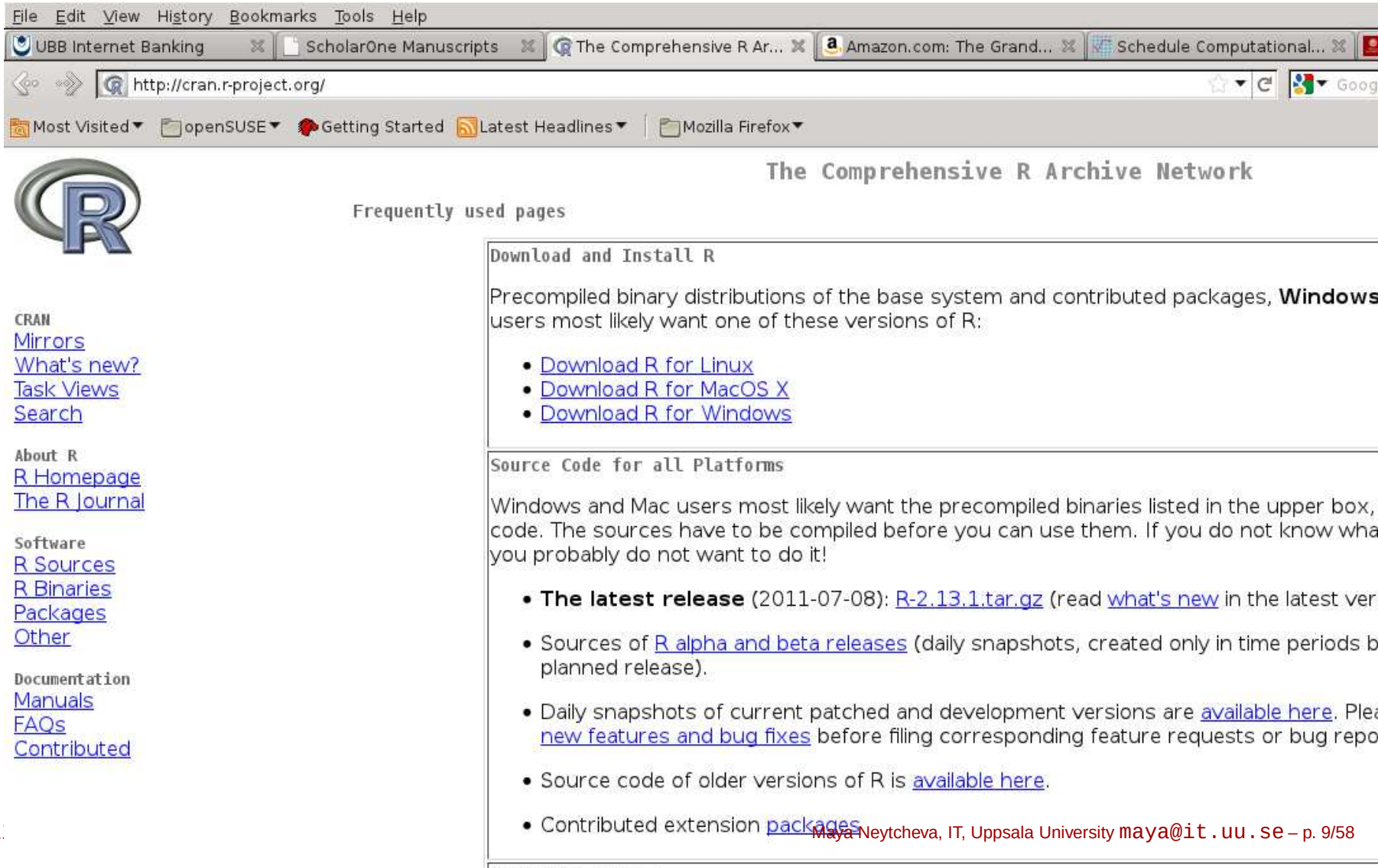
History of $\mathbb{R}$

- S: language for organizing, visualizing, and analyzing data analysis, a project for statistics at Bell Labs since 1976
- S-plus: licensed by AT&T/Lucent to Insightful Corp.
- $\mathbb{R}$: originally written and released by Ross Ihaka and Robert Gentleman at Univ. Auckland during the 90's
- Since 1997: system under the stewardship of the R Project and the R-foundation
- CRAN (*Comprehensive* $\mathbb{R}$ *Archive Network*)

```
http://www.stat.auckland.ac.nz/ ihaka/
http://www.r-project.org/
http://www.r-project.org/foundation/main.html
http://cran.r-project.org/
```

UPPSALA UNIVERSITET

ℝ on the web



UPPSALA UNIVERSITET

History of Matlab (Matrix Laboratory)

- a numerical computing environment and fourth-generation programming language
- efficient data handling and storage
- operates on matrices (mainly)
- provides a collection of tools for numerical computations
- flexible graphical display tools
- tool-boxes: statistical computations included
- options to integrate with other languages (C, Fortran, Java)
- tools for producing documentation and training materials (publishing)

UPPSALA
UNIVERSITET

History of Matlab (Matrix Laboratory)

- Beginning, late 1970's, Cleve Moler; easy access to LINPACK and EISPACK
- Jack Little, 1984, rewrites Matlab in C
- In 2000, Matlab is rewritten again to use LAPACK

UPPSALA
UNIVERSITET

Strengths of Matlab

- MATLAB is relatively easy to learn
- MATLAB code is optimized to be relatively quick when performing matrix operations
- MATLAB may behave like a calculator or as a programming language
- MATLAB is interpreted, errors are easier to fix
- Although primarily procedural, MATLAB does have some object-oriented elements

UPPSALA
UNIVERSITET

Weaknesses of Matlab

- MATLAB is NOT a general purpose programming language
- MATLAB is an interpreted language (making it for the most part slower than a compiled language such as C++)
- MATLAB is designed for scientific computation and is not suitable for some things (such as parsing text)
- Matlab is expensive

UPPSALA UNIVERSITET

GNU Octave

http://www.gnu.org/software/octave/about.html

Octave is a high-level interactive language, primarily intended for numerical computations that is mostly compatible with Matlab.

Since 1988, John W. Eaton and many others.
Full-time development began in the Spring of 1992.
June 24, 2011, Version 3.4.2

UPPSALA
UNIVERSITET

We will go through ...

$\mathbb{R}$:

- Basics
- Computation
- Visualization
- Programming
- Miscs

UPPSALA UNIVERSITET

We will go through ...

ℝ:

- Basics
- Computation
- Visualization
- Programming
- Miscs

Matlab:

- Basics
- Computation
- Visualization
- Programming
- Miscs

Matlab-Rlink: under Windows one can call ℝ from Matlab.

UPPSALA UNIVERSITET

ℝ : Basics (syntax, conventions, ...)

ℝ	Matlab
> R	» matlab
> q()	» exit
> getwd()	» pwd
`[1] "/home/maya/Slides"`	`/home/maya/Slides`
> setwd("../")	» cd ../
> ls()	» who » whos
> help()	» help
> help("+")	» help +
> help(rand)	» help rand
> help.search("rand")	» lookfor rand
> apropos("rand")	

UPPSALA UNIVERSITET

ℝ : First examples

```
> x=2
> y<-3
> ls()
[1] "x" "y"
> help(rand)
No documentation for ’rand’ in specified packages and libraries:
you could try ’help.search("rand")’
> help.search("rand")
Help files with alias or concept or title matching ’rand’ using
regular expression matching:

Random.user(base)       User-supplied Random Number Generation
RNG(base)               Random Number Generation
r2dtable(base)          Random 2-way Tables with Given Marginals
sample(base)            Random Samples and Permutations
houseprices(DAAG)       Aranda House Prices
randu(datasets)         Random Numbers from Congruential Generator
....
```

UPPSALA UNIVERSITET

ℝ : To see the code of a function

```
> plot
function (x, y, ...)
{
    if (is.null(attr(x, "class")) && is.function(x)) {
        nms <- names(list(...))
        if (missing(y))
            y <- {
                if (!"from" %in% nms)
                  0
                else if (!"to" %in% nms)
                  1
                else if (!"xlim" %in% nms)
                  NULL
            }
        if ("ylab" %in% nms)
            plot.function(x, y, ...)
        else plot.function(x, y, ylab = paste(deparse(substitute(x))
            "(x)"), ...)
    }
```

UPPSALA
UNIVERSITET

$\mathbb{R}$: basics

- objects (variables)
- naming conventions
- assignments
- functions
- workspace
- history

UPPSALA UNIVERSITET

ℝ : basics - objects

- names
- object types: vector, factor, matrix, array, data.frame, ts, list
- attributes:
  - mode: numeric, complex, character, logical
  - length: number of elements in (each dimension)
- creating objects: assigning a value, blank objects

UPPSALA UNIVERSITET

$\mathbb{R}$: basics - names, assignments

Names:

- case-sensitive
- must start with a letter (A-Z or a-z)
- can contain letters, numbers, '.' and '_'

UPPSALA UNIVERSITET

ℝ : basics - names, assignments

ℝ	Matlab
> x=2	» x=2
> y<-4	» y=4
> z=-1:3	» z=-1:3
[1] -1 0 1 2 3	z = -1 0 1 2 3
> a_a = 5	» a_a=5
> a.a=8	» a.a=8
> a.b=-2	» a.b=-2
> ls()	» who
[1] "a.a" "a.b" "a_a"	a a_a
> 1+1:5	» 1+1:5
[1] 2 3 4 5 6	2 3 4 5 6

UPPSALA UNIVERSITET

ℝ : basics - vectors

```
> colors<-c("green", "blue", "orange", "yellow", "red")
> colors
[1] "green"  "blue"   "orange" "yellow" "red"
> x
[1]  1  3  2 10  5

> names(x)            # check if any names are attached to x
NULL
> names(x)<-colors
> x
 green   blue orange yellow    red
     1      3      2     10      5
> x["green"]          # component reference by its name
green
    1
```

UPPSALA UNIVERSITET

ℝ : basics - workspace

- during a session all objects (variables) are stored in a working (temporary) memory
- list objects: `ls()` or `ls(objname)`

  delete objects: `rm()` or `rm(objname)`

  save workspace: `save(x,y,file="MyData.Rdata")`

  load objects: `load("MyData.Rdata")`

ℝ : basics - command line history

- can be saved, loaded, displayed
- during a session one can use the arrow keys to 'walk' on the command stack

```
> savehistory(file="MY.Rhistory")
> loadhistory(file="MY.Rhistory")
> history(max.show=Inf)
```

Same functionality in Matlab (`diary`)

UPPSALA UNIVERSITET

ℝ : basics - variables (objects)

ℝ	Matlab
> a = 49	»a = 49
> sqrt(a)	» sqrt(a)
> b<- "I am happy"	» b='I am happy'
> sub("happy", "sad", b)	» changem(b,'sad','happy')
> [1] "I am sad"	>> Error using ==> changem
> d = (2==3)	» d = (2==3)
>	»

UPPSALA UNIVERSITET

$\mathbb{R}$: an example to have in mind:

```
a0 <- 11/999          # has a repeating decimal representation
(a1 <- as.character(a0))
format(a0, digits=16) # shows one more digit
a2 <- as.numeric(a1)
a2 - a0               # normally around -1e-17
as.character(a2)      # normally different from a1
print(c(a0, a2), digits = 16)

form <- y ~ a + b + c
as.character(form)  ## length 3
deparse(form)       ## like the input
```

UPPSALA
UNIVERSITET

Matlab analogy

```
>format short % default
>a=11/999
a = 0.0110

>format long
a = 0.011011011011011
```

UPPSALA UNIVERSITET

ℝ /Matlab: vectors

- vector: an ordered collection of data of the same type
- In ℝ , a single number is a vector of length 1
- vector types: numbers, character strings, logical

ℝ	Matlab
> v=c("a","aa","aaa")	»v=['a','aa','aaa']
> [1] "a" "aa" "aaa"	>> v = aaaaaa
> v[2]	» v(2)
[1] "aa"	a
> w=1:10	» w=1:10

ℝ : basics - matrices and arrays

Matrix: rectangular table of data of the same type

Array: multidimensional

ℝ	Matlab
> x=matrix(data=1,nr=2,nc=3)	» x=ones(2,3)
> x=matrix(0,2,3)	» x=zeros(2,3);
> array(1:3, c(2,4))	»
> v=c(1,"a",2)	
> zz=array(v,c(2,2,2))	

```
zz=array(v,c(2,2,2))
zz
, , 1                    , , 2
     [,1] [,2]             [,1] [,2]
[1,] "1"  "2"        [1,] "a"  "1"
[2,] "a"  "1"        [2,] "2"  "a"
```

$\mathbb{R}$: basics - matrices

```
> m1<-matrix(c(1,3,2,5,-1,2,2,3,9),ncol=3,byrow=T);m2
     [,1] [,2] [,3]
[1,]    1    3    2
[2,]    5   -1    2
[3,]    2    3    9

 #submatrix of m2 with the first row and column removed
> m2[-1,-1]
     [,1] [,2]
[1,]   -1    2
[2,]    3    9
> m1=matrix(1:4, ncol=2);
> m2<-matrix(c(10,20,30,40),ncol=2)
> m1*m2                    # component-wise multiplication
> m1 %*% m2                # matrix-matrix multiply
> solve(m1)                #inverse matrix of m1
```

ℝ : basics - matrices

```
#diag() is used to construct a k by k identity matrix
> diag(3)
     [,1] [,2] [,3]
[1,]    1    0    0
[2,]    0    1    0
[3,]    0    0    1
> diag(c(2,3,3))        #as well as other diagonal matrices
     [,1] [,2] [,3]
[1,]    2    0    0
[2,]    0    3    0
[3,]    0    0    3
```

ℝ /Matlab: basics - matrices

```
> eigen(m2)
$values
[1] 53.722813 -3.722813

$vectors
           [,1]       [,2]
[1,] -0.5657675 -0.9093767
[2,] -0.8245648  0.4159736
```

```
[V,D] = eig(A);  % dense matrix
E = eigs(AS);    % sparse matrix
```

UPPSALA UNIVERSITET

Matlab: basics - matrices

```
A = rand(100);
B = A(1:20,1:20);

cn = cond(A);
cn = condest(A);

x = A\b;
```

Matrix characteristics: positive definiteness, norms, eigenvalues, confition number

What is the computational complexity of the operations we do?

UPPSALA UNIVERSITET

ℝ : basics - data frames

Data frame: rectangular table with rows and columns;
- data within each column has the same type (number, character, logical)
- different columns may have different types (as in a spreadsheet)

```
> L3 <- LETTERS[1:3]
> (d <- data.frame(cbind(x=1, y=1:10),fac=sample(L3,10,repl=TRUE)))
> data.frame(cbind(1,1:10),sample(L3, 10, repl=TRUE))

> is.data.frame(d)
> (dd <- cbind(d, char = I(letters[1:10])))
> rbind(class=sapply(dd, class), mode=sapply(dd, mode))

> stopifnot(1:10 == row.names(d))# {coercion}

> (d0  <- d[, FALSE]) # NULL data frame with 10 rows
> (d.0 <- d[FALSE, ]) # <0 rows> data frame  (3 cols)
> (d00 <- d0[FALSE,])  # NULL data frame with 0 rows
```

ℝ : basics - data frames

From
http://www.personality-project.org/r/r.anova.htm
R and Analysis of Variance

```
#tell where the data come from
> datafn="http://personality-project.org/R/datasets/R.appendix1.data
#read the data into a table
>data.ex1=read.table(datafn,header=T)
> data.ex1
   Dosage Alertness
1       a        30
2       a        38
3       a        35
4       a        41
5       a        27
6       a        24
7       b        32
8       b        26
...
```

ℝ : basics - data frames

```
#do the analysis of variance
> aov.ex1 = aov(Alertness~Dosage,data=data.ex1)

#show the summary table
> summary(aov.ex1)
            Df Sum Sq Mean Sq F value   Pr(>F)
Dosage       2 426.25  213.13  8.7887 0.002977 **
Residuals   15 363.75   24.25
---
Signif. codes:  0 ?***? 0.001 ?**? 0.01 ?*? 0.05 ?.? 0.1 ? ? 1
```

UPPSALA UNIVERSITET

ℝ /Matlab: basics - choosing data elements

select only one element:	x[2
select range of elements:	x[1
select all but one element:	x[-
slicing:	x[c
select ion based on logical operators:	x[x

A(:,1:5)

spy(A>0)

B=A(p,q);

UPPSALA UNIVERSITET

ℝ : importing and exporting data

There exist various possibilities for getting data in and out. Most common case is to deal with rectangular tables in the form of +tttab-delimited text files.
Examples:

- `read.table`, `read.csv`, `scan`
- `read.delim("filename.txt")`
- `write.table`, `write`, `write.matrix`,

'csv' - comma-separated values

$\mathbb{R}$: programming - functions

What is a function in $\mathbb{R}$?
Functions perform operations with data.
Input: function arguments (many of those) and options
Output: function result (only one)
The functions themselves are objects
Example:

```
add = function(a,b){
      result=a+b
      return(result)
}
```

UPPSALA UNIVERSITET

ℝ : functions

```
> y.fun<-function (x)
+ {y<-(log(x))^2-x*exp(-x^3)
+ }
> rr=y.fun(7)
> rr
[1] 3.786566
```

```
> source("my_code.R")
> my_code(3,4,5)
[1] 2
```

UPPSALA
UNIVERSITET

Matlab: functions

```
>> y = inline('log(x).^2-a*x.*exp(x.^3)')
y = Inline function:
     y(a,x) = log(x).^2-a*x.*exp(x.^3)

x = 1:0.1:2;
plot(y(2,x)),hold
plot(y(5,x),'r')
plot(y(0,x),'k')
```

UPPSALA UNIVERSITET

ℝ : frequently used operators

<- =	assign	|	or
+	sum	&	and
-	difference	>	greater
*	multiplication	<	less
/	division	>=	greater or equal
∧	exponent	<=	less or equal
%%	mod	!	not
%*%	dot product	!=	not equal
%/%	integer division	==	is equal
%in%	subset		

UPPSALA UNIVERSITET

ℝ : frequently used functions

c	concatenate	table	counts occurrence of values
cbind	concatenate	summary	generic stats
rbind	vectors	sort, order	sort, orde
min	minimum	rank	rank a vector
max	maximum	cat	print as char
length	# of elements	print	show value
dim	% rows, cols	paste	c ( ) as char
floor	max integer	apply	repeat over rows cols
ceiling	min integer	which	TRUE indices
round, trunk			

UPPSALA UNIVERSITET

ℝ : Statistical functions

rnorm, dnorm, pnorm, qnorm	Normal distribution sample, density, cumulative distribution function (cdf), quantiles
lm, glm, anova	Model fitting
loess, lowess	Smooth curve fitting
sample	Resampling (bootstrap, permutation)
.Random.seed	Random number generation
mean, median	Location statistics
var, covar, cov, mad, range	Scale statistics
svd, qr, chol, eigen	Linear Algebra

UPPSALA UNIVERSITET

ℝ : Graphical functions

plot	generic plot
points	add points
lines, abline	add lines
text, mtext	add text
legend	add a legend
axis	add axis
box	add box around all axes
par	plotting parameters
colors, palette	use colors

UPPSALA UNIVERSITET

ℝ : programming stuff - branching

```
if (logical expression) {
    statements
}
else {
    alternative statements
}
ifelse(ligical expression, yes-statement,no-statement)
```

`else` is optional

UPPSALA UNIVERSITET

ℝ : programming stuff - loops

Performing similar tasks multiple times: for all elements of a list, for all columns of an array etc.

```
for (i in 1:10) {
    print(i*i)
}
i=1
while(i<=10) {
    print(i*i)
    i=i+sqrt(i)
}
```

See also `repeat, break, next`

ℝ : Missing values

Variables of each data type (numeric, character, logical) may also take the value `NA` (’not available)

`NA` is NOT `0, "", FALSE, MULL`

Operations that involve `NA` may or may not produce `NA` as a result.

```
> NA==1
[1] NA
> NA+1
[1] NA
> max(c(NA,4,9))
[1] NA
> max(c(NA,4,9),na.rm=T)
[1] 9
> NA|TRUE
[1] TRUE
> NA&TRUE
[1] NA
```

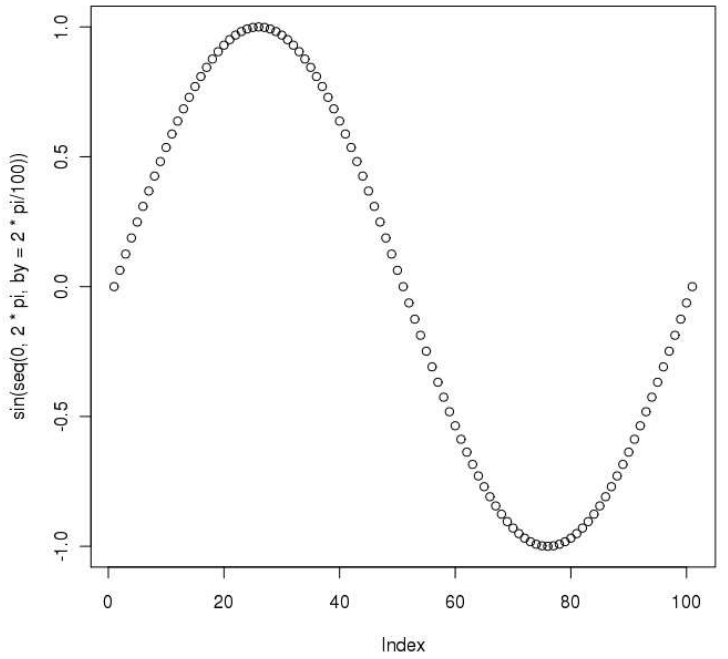
UPPSALA UNIVERSITET

ℝ : What type is my data?

class	object class (vector, matrix, function, logical, list,...)
mode	numeric, logical, character
storage.mode, typeof	double, integer,character,logical
is.function	logical (TRUE if function)
is.na	logical (TRUE if NA)
names	names, associated with the object
dimnames	names, associated with each dimension of array
attributes	names, class, etc

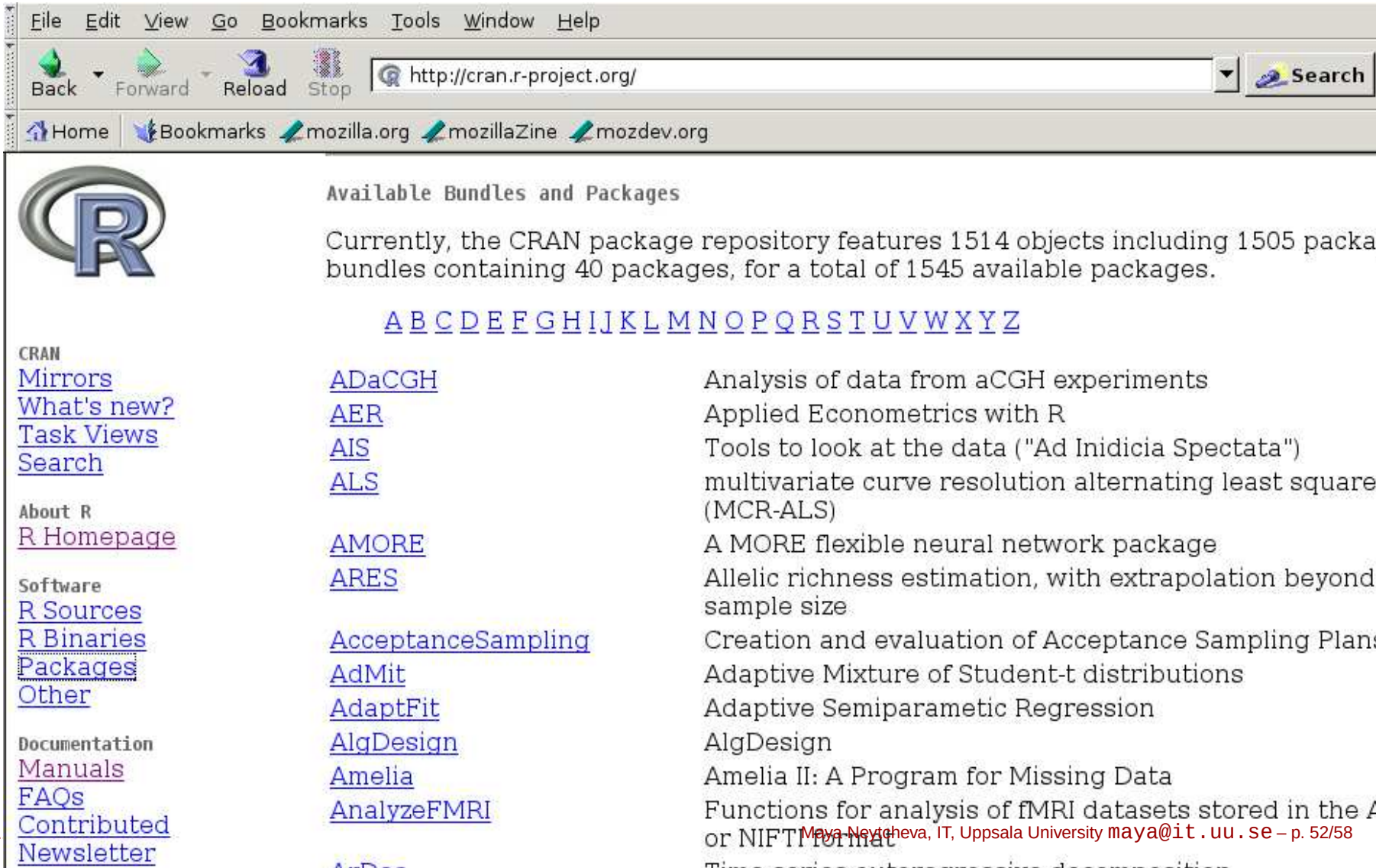
ℝ : plotting stuff

```
plot(sin(seq(0,2*pi,by=2*pi/100)))
```



UPPSALA UNIVERSITET

ℝ : package handling



UPPSALA UNIVERSITET

ℝ : package handling

UPPSALA UNIVERSITET

ℝ : package handling

```
> library(lpridge}
Error in library(lpridge) : there is no package called 'lpridge'
> library(MASS)
> data()
Data sets in package 'datasets':

AirPassengers           Monthly Airline Passenger Numbers 1949-1960
BOD                     Biochemical Oxygen Demand
CO2                     Carbon Dioxide uptake in grass plants
ChickWeight             Weight versus age of chicks on different die
DNase                   Elisa assay of DNase
...
HairEyeColor            Hair and Eye Color of Statistics Students
Harman23.cor            Harman Example 2.3
Harman74.cor            Harman Example 7.4
Indometh                Pharmacokinetics of Indomethicin
...
```

ℝ : package handling ...

```
> HairEyeColor
, , Sex = Male

       Eye
Hair    Brown Blue Hazel Green
  Black    32   11    10     3
  Brown    38   50    25    15
  Red      10   10     7     7
  Blond     3   30     5     8

, , Sex = Female

       Eye
Hair    Brown Blue Hazel Green
  Black    36    9     5     2
  Brown    81   34    29    14
  Red      16    7     7     7
  Blond     4   64     5     8
```

UPPSALA UNIVERSITET

ℝ : package handling

```
> summary(HairEyeColor)
Number of cases in table: 592
Number of factors: 3
Test for independence of all factors:
        Chisq = 171.81, df = 24, p-value = 2.647e-24
        Chi-squared approximation may be incorrect
```

UPPSALA UNIVERSITET

ℝ : Linux: Rcmdr

```
sudo R
install.packages('Rcmdr')
q()
...
R
library(Rcmdr)
```

UPPSALA UNIVERSITET

ℝ : lapply

```
>?lapply
Apply a Function over a List or Vector

Description:

     'lapply' returns a list of the same length as 'X', each element
     which is the result of applying 'FUN' to the corresponding elem
     of 'X'.

     'sapply' is a user-friendly version and wrapper of 'lapply' by
     default returning a vector, matrix or, if 'simplify="array"', a
     array if appropriate, by applying 'simplify2array()'.  'sapply(
     f, simplify=FALSE, USE.NAMES=FALSE)' is the same as 'lapply(x,f

     'vapply' is similar to 'sapply', but has a pre-specified type o
     return value, so it can be safer (and sometimes faster) to use.

     'replicate' is a wrapper for the common use of 'sapply' for
     repeated evaluation of an expression (which will usually involv
```