Statistical Programming Languages – Day 1

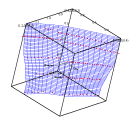
SVN-revision: 50

Uwe Ziegenhagen

Institut für Statistik and Ökonometrie
Humboldt-Universität zu Berlin
http://www.uweziegenhagen.de

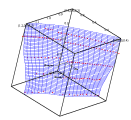
About this Course

- Uwe Ziegenhagen (ziegenhagen@wiwi.hu-berlin.de)
- consultation hours: upon agreement
- 80% Exam (90 minutes)
- 20% Presentation (15-20 minutes)
- Presentation: practical data analysis
- Moodle registration is required
- dates as announced in Moodle



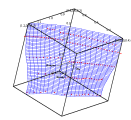
What is *R*?

- ***S*** language, developed by Becker and Chamber in 1984
- GNU implementation of ***S*** 1992 by R. Ihaka and R. Gentleman in NZ
- great variety of packages covering all fields of statistics
- versions for Win32, Linux/Unix, Mac OS



Course Overview

- Introduction
- *R* as a calculator
- Exploratory Data Analysis
- Graphics
- Regression and Testing
- Programming *R*



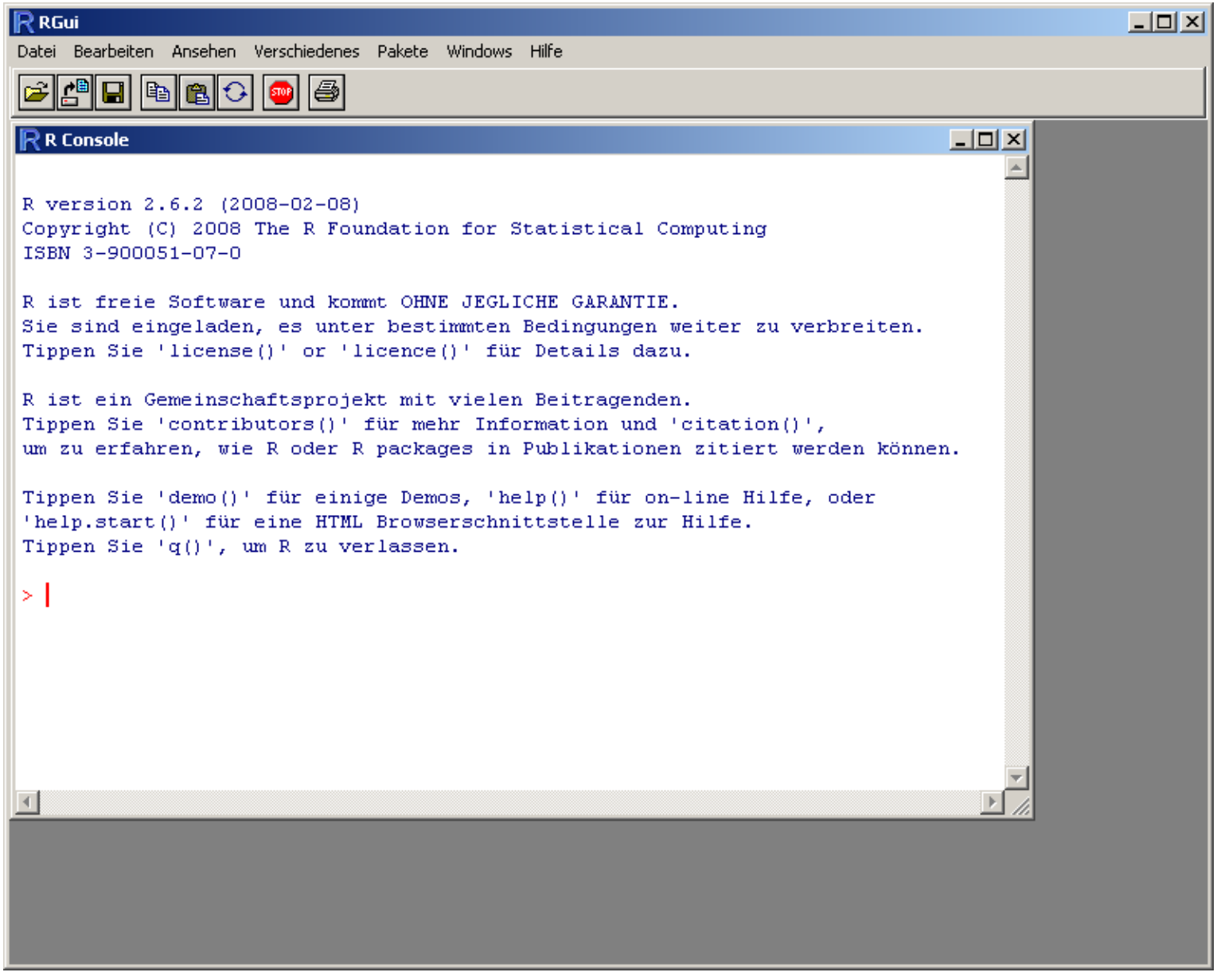


Figure 1: R Graphical User Interface (Windows)

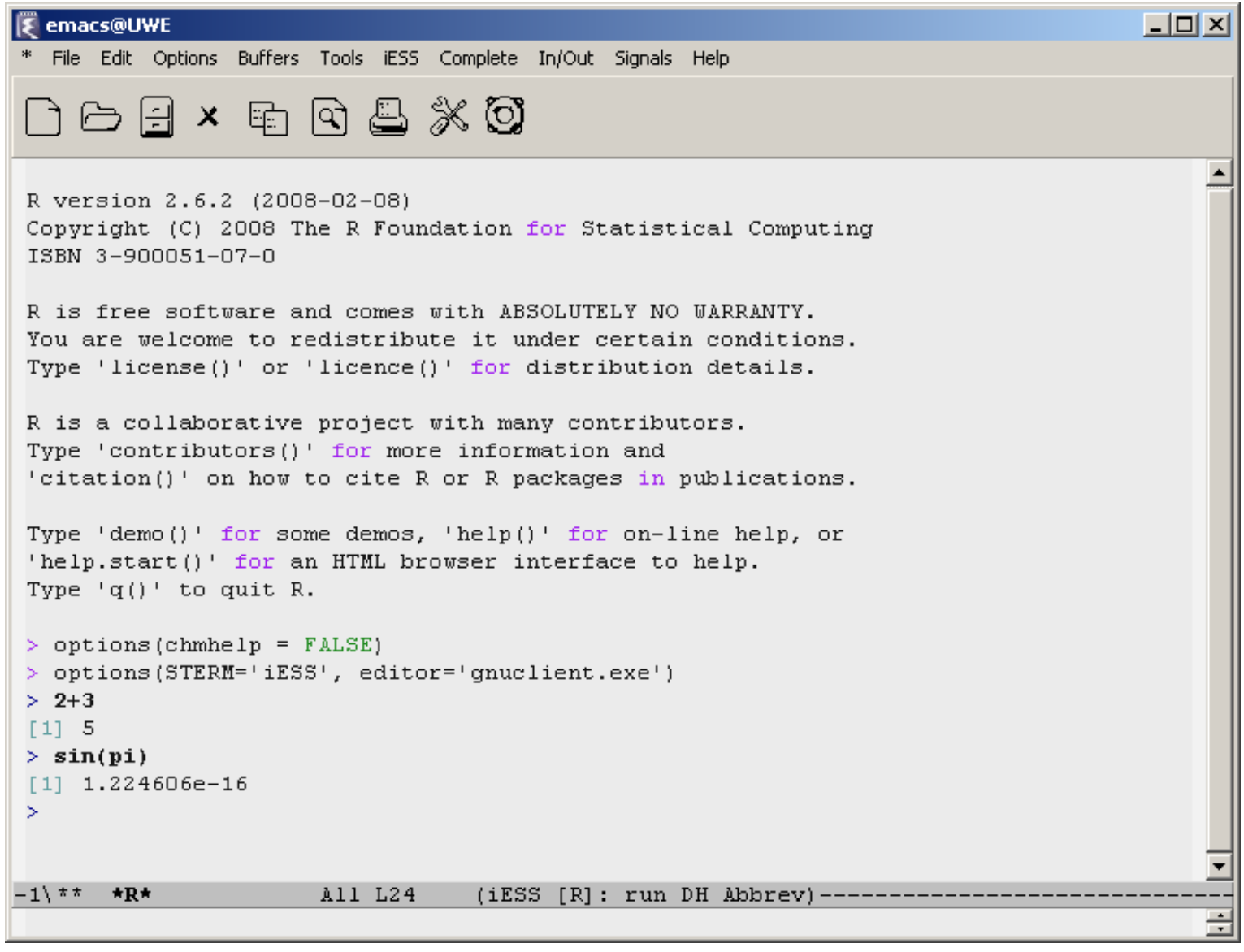
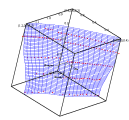


Figure 2: Emacs with ESS (Windows)

Emacs and ESS

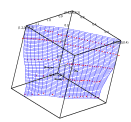
- **If:** you have no idea what Emacs is, skip this...
- **Else:** download ESS, extract to EMACS home directory
- add lisp code (below) to `.emacs`
- Run ***R*** by `M-x R`

```
1 (load "c:/emacs-22.1/ess-5.3.7/lisp/ess-site")
2 (setq inferior-R-program-name "c:/Programme/R/R
    -2.6.2/bin/Rterm.exe")
```

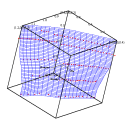
Getting Help...

```
help() #
help(sin) # help for sin()
?sin # help for sin
help.start() # HTML help
library() # show installed libraries
library(help="<package>")
help.search("sin")
```

Basic *R* stuff

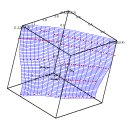
```
setwd("x:/myR") # set working directory
getwd() # get working directory
save.image() # saves workspace to .Rdata
savehistory() # save command history
load(".Rdata")) # loads workspace
loadhistory() # as the name says
source("myfile.r") # read commands from file
sink("output.txt") # write output to file
sink() # output to screen
q() # quits R
```

Installing new packages

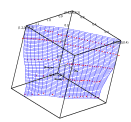
(may require administrator rights)

```
# download and install
install.packages("multtest")
# update local packages
update.packages()
# show installed libraries
libraries()
# load library
library(name,lib.loc=[location])
# load library in functions
require() # return TRUE or FALSE
```

Customizing *R*

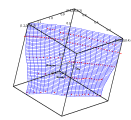
```
# show global options
options()
# set option
options(prompt=":-)")
# get spec. option
getOption(prompt)
```

Customizing *R*

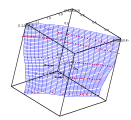
- Unix/Linux: local `.Rprofile`
- Windows: global `Rprofile.site`

Look & Feel changes can be made in `Rconsole`. Unix/Linux allows different .Rprofile files.



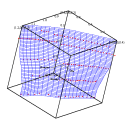
Basic Calculations with *R*

```
1+2
1*2
1/2
1-2
5 %/% 2 # 2; int division
5 %% 2  # 1; modulo division
```

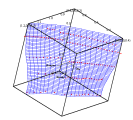
Basic Calculations with *R*

```
1  <   # smaller
2  <=  # smaller or equal
3  >   # bigger
4  >   # bigger or equal
5  !=  # unequal
6  ==  # equal
7  &   # logical AND (vector)
8  |   # logical OR (vector)
9  &&  # logical AND (no vector)
10 ||  # logical OR (no vector)
```

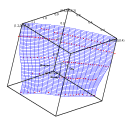
Basic Calculations with *R*

```
2^2
sqrt(2)
sin(pi) # cos, tan
acos(0) # asin, atan atan2
```

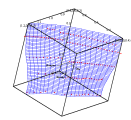
Basic Calculations with *R*

```
c <- 1:3 * pi
c # [1] 3.141593 6.283185 9.424778
floor(c)   # [1] 3 6 9
ceiling(c) # [1] 4 7 10
trunc(pi)  # 3
trunc(-pi) # -3
floor(-p)  # -4
round(pi)  # 3
```

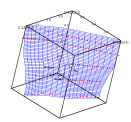
Variable names

- sequence always alphabetic => numeric
- strings of alphabetic characters: a, b2, abc.de, a1, a1_23
- names are case sensitive 'a123' is not equal to 'A123'
- `pi` is a constant, cannot be used as variable name
- `print(x)` prints content of x



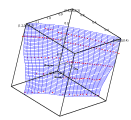
Overview of Objects I

```
1 # show loaded packages and data
2 search()
3 ls(2) # show functions for spec. package
4 ls() # list objects
5 objects() # like ls()
6 rm(name) # removes name from workspace
7 rm(list=ls()) # removes all objects
```

R main structures

vectors	just vectors of length m, one type
matrices	$m \times n$ arrays, one type
dataframes	usually read from files, different types

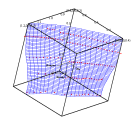


R classes for vectors

use `class(`*`object`*`)` for the type

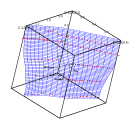
character	vector of strings
numeric	vector of real numbers
integer	vector of signed integer
logical	vector of TRUE or FALSE*
complex	vector of complex numbers
list	vector of ***R*** objects
factor	sets of labelled observations, pre-defined set of labels
NA	not available, missing value

*Remark: Short forms T and F exist, but do not use them!



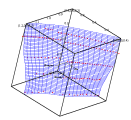
R main structures

- matrix('vector') converts to matrix of $m \times 1$
- matrix('vector', ncol=1) does the same
- matrix('vector', nrow=1) converts to matrix of $1 \times n$
- as.data.frame('matrix') converts to data frame
- as.matrix('data frame') converts to matrix
- as.vector('matrix') converts to vector, if matrix has only one row or column



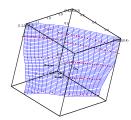
Variables, Vectors and Matrices

```
1 a <- 2 # double number a = 2
2 b <- 1:3 # integer vector b = [1 2 3]
3 c <- 1:pi # integer vector c = [1 2 3]
4 d <- c(1,2,3,4) # double vector [1] 1 2 3 4
5 t(d) # returns d as row vector (transposes d)
```

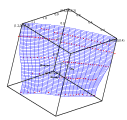
Variables, Vectors and Matrices

```
a = 1:3
b = 2:4
c(a,b) # [1] 1 2 3 2 3 4
c(1,1:3) # [1] 1 1 2 3
seq(1,3) # [1] 1 2 3
seq(3) # [1] 1 2 3
seq(1,2,by=0.1) [1] 1.1 1.2 1.3 1.4 1.5 ...
seq(1,3,0.5) # [1] 1.0 1.5 2.0 2.5 3
rep(1:4,2) # [1] 1 2 3 4 1 2 3 4
```

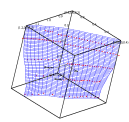
Variables, Vectors and Matrices

```
1 a <- letters[1:3]
2 a
3 b <- LETTERS[1:3]
4 b
5 c <- month.abb[1:6]
6 c
7 d<- month.name[1:12]
8 d
```

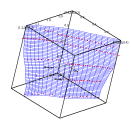
Variables, Vectors and Matrices

```
> matrix(1:12, nrow=3)
     [,1] [,2] [,3] [,4]
[1,]    1    4    7   10
[2,]    2    5    8   11
[3,]    3    6    9   12
> matrix(1:12, nrow=3, byrow = T)
     [,1] [,2] [,3] [,4]
[1,]    1    2    3    4
[2,]    5    6    7    8
[3,]    9   10   11   12
> matrix (1, nrow=2, ncol=2)
     [,1] [,2]
[1,]    1    1
[2,]    1    1
```

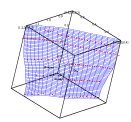
Variables, Vectors and Matrices

```
> matrix(1:12, 3,4)
     [,1] [,2] [,3] [,4]
[1,]    1    4    7   10
[2,]    2    5    8   11
[3,]    3    6    9   12
```

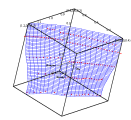
Variables, Vectors and Matrices

```
> matrix(0, nrow = 5, ncol = 5)
     [,1] [,2] [,3] [,4] [,5]
[1,]    0    0    0    0    0
[2,]    0    0    0    0    0
[3,]    0    0    0    0    0
[4,]    0    0    0    0    0
[5,]    0    0    0    0    0
```

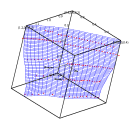
Variables, Vectors and Matrices

```
# Concatenation
> x = 1:3
> y = 4:6
> rbind(x,y)
  [,1] [,2] [,3]
x    1    2    3
y    4    5    6
> cbind(x,y)
     x y
[1,] 1 4
[2,] 2 5
[3,] 3 6
```

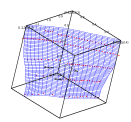
Variables, Vectors and Matrices

```
1 x <- matrix(1:12, 3, 4)
2 # extract the diagonal of a matrix
3 x[row(x) == col(x)]
```

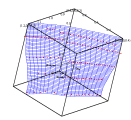
Variables, Vectors and Matrices

```
> k=matrix(1:10,2,5)
> k
     [,1] [,2] [,3] [,4] [,5]
[1,]    1    3    5    7    9
[2,]    2    4    6    8   10
> k[1:2,3:4]
     [,1] [,2]
[1,]    5    7
[2,]    6    8
```

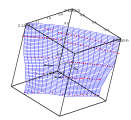
Variables, Vectors and Matrices

```
1 diag(5) # diagonal 5x5 matrix of '1'
2 diag(5,7,8) # 7x8 matrix with 5 on diag.
```

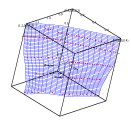
dimension names (matrices and array)

```
> b <- matrix(1:20,4,5)
> dimnames(b)<- list(letters[1:4],letters[1:5])
> b["b","b"]
[1] 6
```

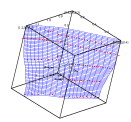
Matrix Size

```
dim(x) # size of matrix x
x <- matrix(1:10,2,5)
col(x) # column indices of ALL elements
row(x) # row indices of ALL elements
x[<i>,<j>] # extract i-th row and j-th column
```

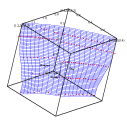
Inf, NaN

```
1 x = 1:3
2 is.finite(x)
3 is.infinite(x)
4 Inf
5 NaN
6 is.nan(x)
```

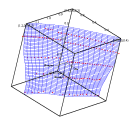
Sums and Products

```
> x = matrix(1:20,4,5)
> sum(x)
[1] 210
> prod(x)
[1] 2.432902e+18
matrix(1:10, nrow=2) -> a
colSums(a)
rowSums(a)
```

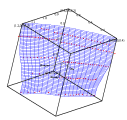
Sums and Products

```
> cumsum(1:10)
 [1]  1  3  6 10 15 21 28 36 45 55
> cumprod(1:5)
 [1]   1   2   6  24 120
[10] 3628800
> cummin(c(3:1, 2:0, 4:2))
[1] 3 2 1 1 1 0 0 0 0
> cummax(c(3:1, 2:0, 4:2))
[1] 3 3 3 3 3 3 4 4 4
```

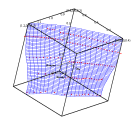
Sums and Products

```
> a=c(3:1, 2:0, 4:2)
> a
[1] 3 2 1 2 1 0 4 3 2
> cummin(a)
[1] 3 2 1 1 1 0 0 0 0
> cummax(a)
[1] 3 3 3 3 3 3 4 4 4
```

Replacing values

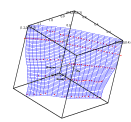
```
> replace(x,x<2,3)
 [1]  3  2  3  4  5  6  7  8  9 10
> x = 1:10
> x
 [1]  1  2  3  4  5  6  7  8  9 10
> replace(x,x<2,3)
 [1]  3  2  3  4  5  6  7  8  9 10
```

Matrix Calculation

- if x, y are $n \times m$ matrices
- $x + y = x[i,j] + y[i,j]$
- $x - y = x[i,j] - y[i,j]$
- if x is $n \times m$ and y is $m \times p$ then

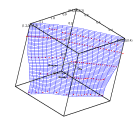
$$x\%*\%y = \sum_{j=1}^{m} x[i,j] \cdot y[j,k]$$



Matrix Calculation

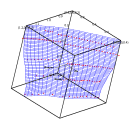
In expressions involving matrix and vector, the vector is interpreted such that the multiplication works.

- If x is vector of length m and y is an $m \times p$ matrix, $x\%*\%y$ is a vector of length p.
- If x is an $n \times m$ matrix and y is a vector of length m, $x\%*\%y$ is a vector of length m.
- If x and y are vectors of length m, $x\%*\%y$ is a scalar (i.e. vector of length 1), representing the inner product $\sum_{i=1}^{m} x[i] * y[i]$.



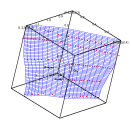
Matrix Inversion

```
1 # if a is nxn matrix
2 solve(a) # inverse of a
3 a^-1 # elementwise inverse
```

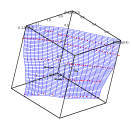
Lists in *R*

```
> a<-c(3,2,1)
> b<-c(6,5,4)
> f<-list(a,b)
> f[1]
[[1]]
[1] 3 2 1
> f[a]
[1] 3 2 1
```

Lists in *R*

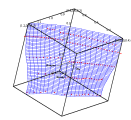
```
> a<-c(3,2,1)
> b<-c(6,5,4)
> f<-list(a,b)
> d<-c("a","b")
> e<-list(a,b,d)
> unlist(f)
> unlist(e)
```

Finding Help

- Google (or your other favorite search engine)
- R-help mailing list
- R-announce, R-package, R-devel (for developer-specific topics)
- R-sig-* special interest groups (r-sig-finance)

See `http://r-project.org/mail.html` for details



For Further Reading

- *An Introduction to R*, R-intro.pdf
- *R Data Import/Export*, R-data.pdf
- *The R Reference Index*, fullrefman.pdf
- *Introductory Statistics with R*, Peter Dalgaard
- *Data Analysis and Graphics Using R*, Maindonald/Braun
- *R Graphics*, Murrell
- *Programmieren mit R*, Uwe Ligges