Lecture overview

- Some general remarks
  - Purpose of the course
  - How to do this course?
  - ECTS, study hours and Mark on course
- General introduction to R
  - Situations for Using R
  - Installing
  - Help: searching, examples, demonstrations, books, tutorials
  - Reading and writing
  - Important Functions
  - Some Plots of data
  - Some Statistical Tests
- Notes on Chapter 1
Purpose of the course

- Applied statistics
  - only little bit of calculus
  - basic understanding of notation

- Introduction to important statistical concepts
  - understanding of concepts such as goodness of fit, maximum likelihood, applications of central limit theorem
  - being able to work with several basic distributions and densities
  - justify choice for statistical test, analyzing data

- Introduction in using R
  - understanding and programming correct code
  - make correct interpretation from output
  - make judgement of validity of statistical inference
How to do this course?

- Study the book and presentations well
- Learn from examples, exercises, assignments, and trial exam
- Print lectures from acrobat (print setup | Lay out | multiple pages), make notes
- Come up with ideas, problems, mistakes, and (sometimes) frustrations!
- Be active and explore!
- Ability to write correct R scripts determines usefulness
ECTS, study hours and Mark on course

5 ECT $\Rightarrow 5 \cdot 28 = 140$ hours of study
lectures 4+4 hours during 7 weeks $\Rightarrow 56$ hours
there remain $\frac{140-56}{7} = 12$ hours of study per week

\[ M = 0.7 \cdot E + 0.15 \cdot \max (E, A_1) + 0.15 \cdot \max (E, A_2) \]

$M$ mark of course, $E$ exam, $A_1$ Assignment 1, $A_2$ Assignment 2

```r
> E <- 5; A1 <- 7; A2 <- 7
> (M <- 0.7 * E + 0.15 * max(E,A1) + 0.15 * max(E,A2))
[1] 5.6
> E <- 8; A1 <- 7; A2 <- 7
> (M <- 0.7 * E + 0.15 * max(E,A1) + 0.15 * max(E,A2))
[1] 8
```

this holds if and only if $E \geq 5.0$
if $E < 5.0$, then $M = E$ (not passing)
General introduction to R
Situations for Using R

- Repeated similar problems
- Programming of visualizations: scalable publication ready plots
- Handle large data sets
- Desire flexibility in statistical programming
- Use of modern techniques (bootstrap, robust, Bayesian)
Reasons for Using R

▶ Widely used in statistics and applied sciences
▶ Reliable free open source
▶ Incorporated in traditional statistical programs (SPSS, MINITAB, SAS)
▶ Versatile: Matlab, MySQL, Perl, JAVA, C++, Fortran
▶ Numerous libraries with modern methods
▶ High level language with many built-in-functions (allows quick programming)

Disadvantages:
▶ Steep learning curve (frequent use helps a lot)
▶ Mainly command line, some Graphical User Interface (GUI)
▶ Use internal C code; pure C++ faster, more efficient
Install R on your personal computer

URL: http://cran.r-project.org
Choose

- operating system: Windows, Linux, Mac
- base
- html help

Type writers to for writing R code

- recommended are Notepad, WinEdt, Kate, emacs
- word processors are not recommended
- use syntax highlighting for correct use of symbols \{ [ (
installing libraries

Example of installing library:

```r
chooseCRANmirror()
install.packages(c("TeachingDemos"), repo="http://cran.r-project.org", dep=TRUE)
library(TeachingDemos)
plot(dice(12,1))
```

or use Packages at top of RGui
Install a complete bundles of libraries (not needed for current course):

```r
install.packages("ctv")
library("ctv")
install.views("Distributions")
install.views("Econometrics")
install.views("ChemPhys")
```
Searching and help

From http://cran.r-project.org

- Search, Task Views, Manuals, FAQs, The R Journal, Wiki
- → Contributed "R reference card"
- Manuals: "An Introduction to R", "The R Language Definition", "Installation and Administration", "R Data Import/Export"

Useful tutorial for beginners:
http://cran.r-project.org/doc/contrib/Verzani-SimpleR.pdf
http://www.r-project.org → choose Books
Help on examples

from commandline

- `help.start()`
- `library(), ls(package:stats), library(help="stats")`
- `help(t.test), ?sum, ??solve, apropos("if")`
- `methods(plot) plotting functions`
- `example(boxplot) examples`
- `demo(lm.glm) demonstrations of code gen. lin. model`
- `mean.default study code of function`
Help on Programming

- help(Control) : “for” and “while” loops
- help(Syntax) : syntax of operators
- help(Logic) : logical operators AND, OR, negation
- help(Arith) : on arithmetic, relational, logical operators, mathematical functions
- help(Special) : gamma function

Help on Editing: top of GUI Help | Console
Some notes on Chapter 1: R as calculator

illustrating the usual rules for addition, multiplication, etc.

\[ 2 + 3 \times (5+1) \]
\[ \text{[1]} \ 20 \]

\[ \text{ln}(10) \]
Error: could not find function "ln"

\[ \text{log}(10) \quad \# \ \text{log} = \ln \]
\[ \text{[1]} \ 2.302585 \]

\[ \text{log10}(10) \]
\[ \text{[1]} \ 1 \]

\[ 10^2 \]
\[ \text{[1]} \ 100 \]

\[ e^1 \]
Error: object 'e' not found

\[ \text{exp}(1) \]
\[ \text{[1]} \ 2.718282 \]
Assignment to objects

> x <- 5
> x
[1] 5
> 6 -> z ; z
[1] 6
> (y <- pi)
[1] 3.141593
Executing functions

```r
> x <- 5
> print(x)  # object x is recognized and printed
[1] 5
> ls()
[1] "x"
> rm(list = ls())
> ls()
character(0)
> dir()
[1] "Figure1.2.jpg"          "Figure12.jpg"
[3] "Lecture1.aux"           "Lecture1.bbl"
```
Vectors

> x <- 1:5
> x[2]
[1] 2
> x[c(2,4)]
[1] 2 4
> x1 <- seq(1, 5, by = 1)
> x2 <- seq(1, 5, length = 5)
> x1==x2
[1] TRUE TRUE TRUE TRUE TRUE
Mode and Class

Some modes: "logical", "integer", "double", "complex", "raw", "character", "list", "expression", "name", "symbol" and "function"

```r
> (x <- 1:5)
[1] 1 2 3 4 5
> mode(x)
[1] "numeric"
> class(x)
[1] "integer"
> str(x)
int [1:5] 1 2 3 4 5
> (z <- x==3)
[1] FALSE FALSE TRUE FALSE FALSE
> mode(z)
[1] "logical"
> as.character(x)
[1] "1" "2" "3" "4" "5"
> mode(as.character(x))
[1] "character"
```
Vectors

create vector e.g. before for loop

```r
> (v1 <- vector())
logictical(0)
> (v2 <- integer(5))
[1] 0 0 0 0 0
> (v3 <- character(5))
[1] "" "" "" ""
> (v4 <- double(5))
[1] 0 0 0 0 0
> mode(v4); str(v4); typeof(v4)
[1] "numeric"
  num [1:5] 0 0 0 0 0
[1] "double"
```
Functions on vectors

> x <- letters[1:5]
> x == c("b")
[1] FALSE TRUE FALSE FALSE FALSE
> sum(x == c("b"))
[1] 1
> x <- 1:5
> length(x)
[1] 5
> sum(x)
[1] 15
> sum(x)/length(x)
[1] 3
Arithmetic on vectors

```r
x <- 1:5
> x + 1
[1] 2 3 4 5 6
> x/2
[1] 0.5 1.0 1.5 2.0 2.5
> x + x
[1] 2 4 6 8 10
> x[1:4]
[1] 1 2 3 4
[1] 3 5 7 9
> sqrt(x)
[1] 1.000000 1.414214 1.732051 2.000000 2.236068
> log10(x)
[1] 0.0000000 0.3010300 0.4771213 0.6020600 0.6989700
> 10^log10(x)
[1] 1 2 3 4 5
```
Dealing with Not Available (NA); indexing

```r
> (x <- c(10,20,NA,4,NA,2))
[1] 10 20 NA 4 NA 2
> mean(x)
[1] NA
> x[c(1,2,4,6)]
[1] 10 20 4 2
> (i <- is.na(x))
[1] FALSE FALSE TRUE FALSE TRUE FALSE
> x[i]
[1] NA NA
> !i
[1] TRUE TRUE FALSE TRUE FALSE TRUE FALSE TRUE FALSE TRUE
> x[!i]
[1] 10 20 4 2
> mean(x[!i])
[1] 9
> mean(x,na.rm = TRUE)
[1] 9
```
Fill vector with random numbers

**draw random numbers from uniform distribution**

```r
> x <- runif(100000)
> round(x[1:5],3)
[1] 0.708 0.292 0.123 0.408 0.339
> mean(x)
[1] 0.4997755
> mean(x[x>.5])
[1] 0.7499488
```
Logical operators

> 5 > 4
[1] TRUE
> 5 >= 4
[1] TRUE
> 5 == 4
[1] FALSE
> 5 != 4
[1] TRUE
> 5 >= 4 & 4 <= 5  # AND
[1] TRUE
> 5 >= 4 | 4 == 5  # OR
[1] TRUE

see also help(Logic)
Some more logic

```r
> x <- sample(c(TRUE, FALSE), 5, replace=TRUE, prob=c(1/2, 1/2))
> x
[1]  TRUE FALSE FALSE FALSE  TRUE
> is.logical(x); is.numeric(x); is.integer(x);
[1]  TRUE
[1]  FALSE
[1]  FALSE
> sum(x); sum(as.integer(x))
[1] 2
[1] 2
```
Conditional execution

```r
if (test) {execute some} else {execute some else}

> bmi <- 22
> goodhealth <- logical()
> if (bmi<25) {goodhealth <- TRUE} else {
    goodhealth <- FALSE
> goodhealth
[1] TRUE
> bmi <- c(22,28,35,19)  # bmi of 5 persons
> goodhealth <- logical(length(mbi))
> for (i in 1:length(bmi)) if (bmi[i] < 25)
    {goodhealth[i] <- TRUE} else {
    goodhealth[i] <- FALSE}
> goodhealth
[1] TRUE FALSE FALSE TRUE
> rm(goodhealth)
> ifelse(bmi<25, goodhealth <- TRUE,  
    goodhealth <- FALSE)
[1] TRUE FALSE FALSE FALSE TRUE
```
> x <- letters[1:5]
> paste(x[1],313:317,sep='-'bla-')
[1] "a-bla-313" "a-bla-314" "a-bla-315" "a-bla-316"
   "a-bla-317"
> paste("AY34X",seq(313,317,by=2),sep = "")
[1] "AY34X313" "AY34X315" "AY34X317"
paste("Today is", date())
a <- dir()
Functions: standard normal

standard normal distribution (bell-shape)

\[ f(x) = \frac{1}{\sqrt{2\pi}} e^{-x^2/2} = \exp\left\{-\frac{x^2}{2}\right\} / \sqrt{2\pi} \]

define the function in R and numerical integration

\[
\begin{align*}
> f & \leftarrow \text{function}(x) \{ \exp(-x^2/2)/\sqrt{2*\pi}\} \\
> f(1) & \\
[1] 0.2419707 \\
> f(1) == \text{dnorm}(1) \ # \ built-in-function \\
[1] \text{TRUE} \\
> \text{class}(f) \\
[1] "function" \\
> \text{integrate}(f,-1,1) \\
0.6826895 \text{ with absolute error < 7.6e-15} \\
> \text{integrate}(f,-2,2) \\
0.9544997 \text{ with absolute error < 1.8e-11} \\
\end{align*}
\]

plot(f,xlim=c(-3,3),col="blue")
bell shaped normal density function \( f(x) = e^{-x^2/2}/\sqrt{2\pi} \)
running code from a file

Use your favorite type writer to save the code

```r
setwd("C:/work/StatisticsForAINF/book/ch1")
x <- seq(-3, 3, length = 101)
y <- dnorm(x)
plot(x, y, type = 'l')
```
in a file called e.g. "acodechunckinR.R"
use `dir()` to verify that your file is in de working directory
in the R console type

```r
source("acodechunckinR.R")
```
to run the code
Save output in a file

let's save names of months by years using a keen idea in mathematics: the outer product

```r
sink("sink-examp.txt")  # Write text to file
outer(month.abb, 2008:2010, FUN = "paste")
sink()                    # close file
```

opening the file `sink-examp.txt` in e.g. Kladblok

```r
> outer(month.abb, 2008:2010, FUN = "paste")

[,1]       [,2]       [,3]
[1,] "Jan 2008" "Jan 2009" "Jan 2010"
[2,] "Feb 2008" "Feb 2009" "Feb 2010"
[3,] "Mar 2008" "Mar 2009" "Mar 2010"
[4,] "Apr 2008" "Apr 2009" "Apr 2010"
[5,] "May 2008" "May 2009" "May 2010"
[6,] "Jun 2008" "Jun 2009" "Jun 2010"
[7,] "Jul 2008" "Jul 2009" "Jul 2010"

...etc
```
Brief overview of important functions

<table>
<thead>
<tr>
<th>R function</th>
<th>purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>rm(list=ls())</code></td>
<td>remove objects</td>
</tr>
<tr>
<td><code>q()</code></td>
<td>quit</td>
</tr>
<tr>
<td><code>history()</code></td>
<td>list of previous input lines</td>
</tr>
<tr>
<td><code>ls()</code>, <code>objects()</code></td>
<td>list of loaded objects</td>
</tr>
<tr>
<td><code>getwd</code>, <code>setwd</code></td>
<td>getting, setting working directory</td>
</tr>
<tr>
<td><code>dir()</code></td>
<td>list of file names working directory</td>
</tr>
<tr>
<td><code>function</code></td>
<td>define real to real function</td>
</tr>
<tr>
<td><code>sample</code></td>
<td>simulated discrete sampling</td>
</tr>
<tr>
<td><code>runif</code></td>
<td>random sampling from uniform distribution</td>
</tr>
<tr>
<td><code>grep</code>, <code>regexpr</code></td>
<td>regular expressions</td>
</tr>
</tbody>
</table>
Warning on some functions used throughout the book

End of Chapter 1 there are functions defined to save space in the book `open`, `save`, `h`, `ndq`

- need to be defined before usage
- these are not “very spectacular”

You should now be able to do assignments of chapter 1