M.Sc. in ‚Transportation Systems'

Applied Statistics in Transport
Introduction R

Prof. Regine Gerike
Technische Universität München, mobil.TUM
regine.gerike@tum.de

Munich, 25/10/2011

Last Week

- We – mobil.TUM
- You – Experiences and Expectations
- Organisational Remarks

Plan for Today's First Lecture

- Questions last week
- Introduction R
- Some first exercises

R - Introduction

- R is a dialect of the S language.
- R is a language and environment for statistical computing and graphics
- R is an implementation of the S programming language created in the 1970th by John Chambers and colleagues at Bell Laboratories
- R was created by Ross Ihaka and Robert Gentleman at the University of Auckland, New Zealand
- R is now developed by the R Development Core Team, of which Chambers is a member.
- R is named partly after the first names of the first two R authors (Robert Gentleman and Ross Ihaka), and partly as a play on the name of S

Features of R

- Runs on almost any standard computing platform
- Frequent releases (April and October); bugfixes; active development
- Quite lean, as far as software goes; functionality is divided into modular packages
- Graphics capabilities very sophisticated and better than most stat packages
- Useful for interactive work, but contains a powerful programming language for developing new tools
- Very active and vibrant user community

Features of R - It’s free!

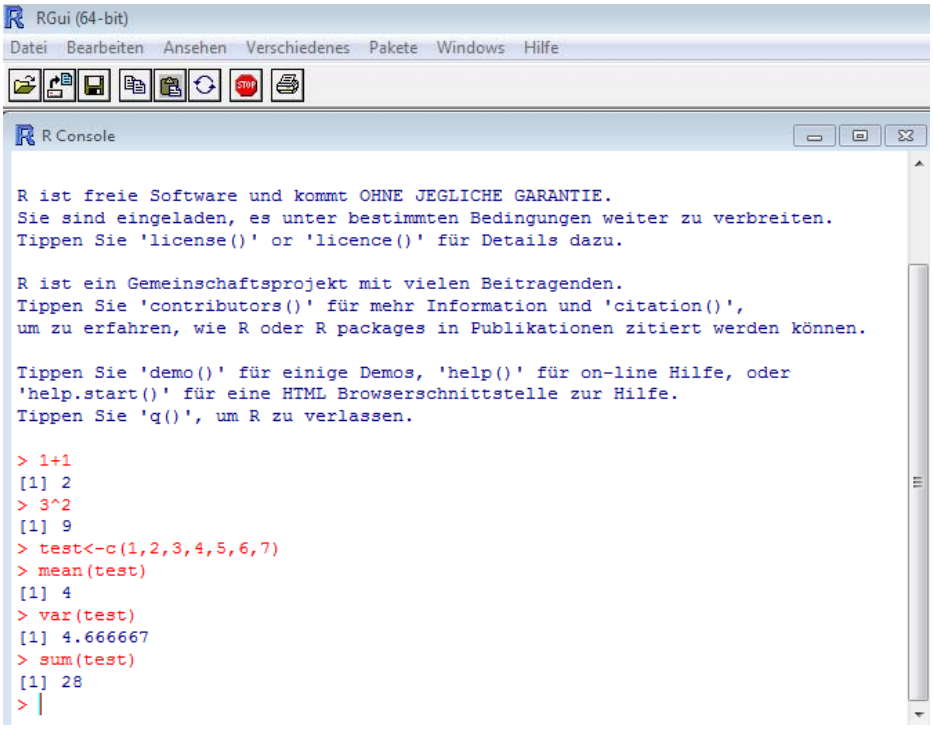
- With free software, you are granted
- The freedom to run the program, for any purpose (freedom 0).
- The freedom to study how the program works, and adapt it to your needs (freedom 1). Access to the source code is a precondition for this.
- The freedom to redistribute copies so you can help your neighbour (freedom 2).
- The freedom to improve the program, and release your improvements to the public, so that the whole community benefits (freedom 3). Access to the source code is a precondition for this.

Drawbacks of R

- Essentially based on 40 year old technology.
- Little built in support for dynamic or 3-D graphics.
- Functionality is based on consumer demand and user contributions. If no one feels like implementing your favourite method, then it's your job!
- Not ideal for all possible situations (drawback of all software packages)

R – Getting started

- Information on R: http://www.r-project.org/
- Installing R: http://cran.r-project.org/



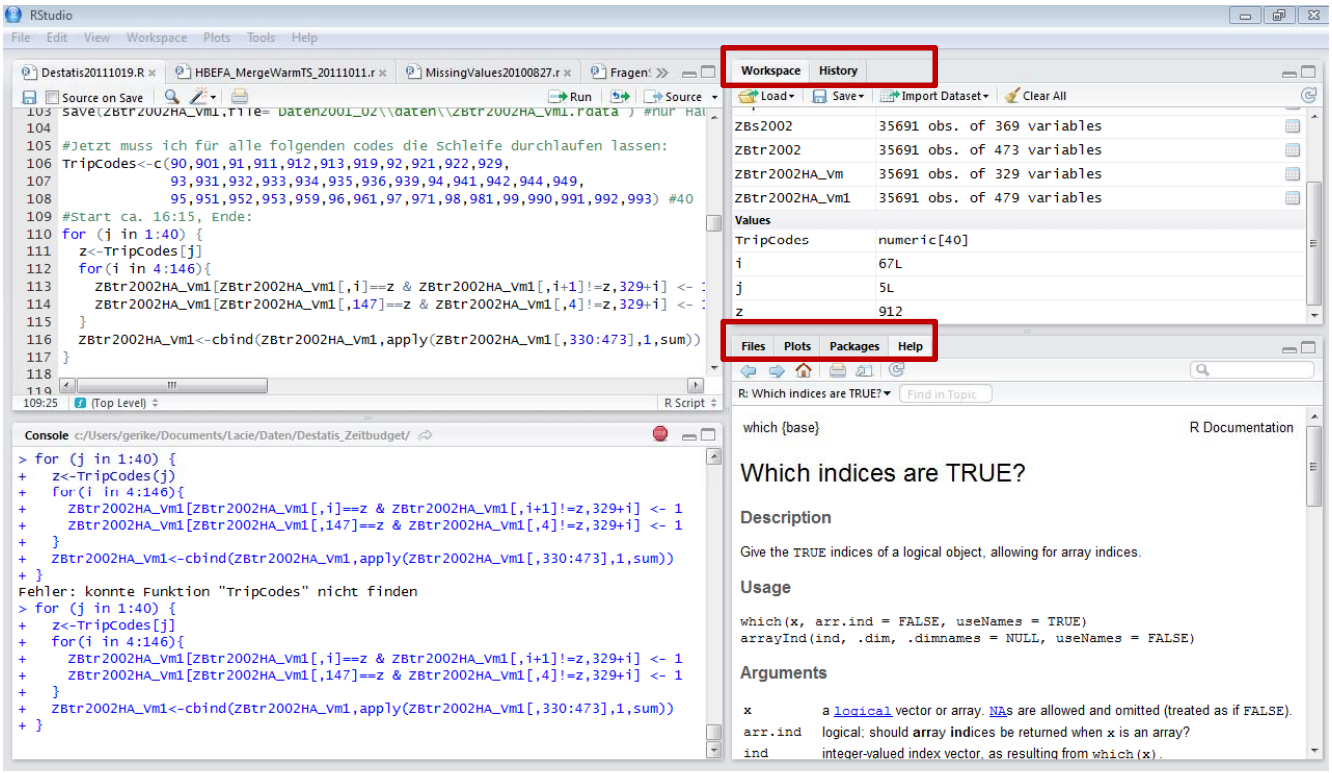
R – Getting started

- Information on R: http://www.r-project.org/
- Installing R: http://cran.r-project.org/

- Recommended for Windows: Tinn-R: http://www.sciviews.org/Tinn-R/

- Even better: R Studio: http://rstudio.org/ (not installed in the lab)

R Studio, http://rstudio.org/



We work with Tinn-R !

R – Getting started

The R system:

- The “base” R system (download from CRAN)
- Everything else.

R functionality is divided into packages.

- The “base” R system contains the base package which is required to run R and the most fundamental functions.
- > 1000 packages on CRAN available
- People often make packages available on their personal websites; there is no reliable way to keep track of how many packages are available in this fashion.

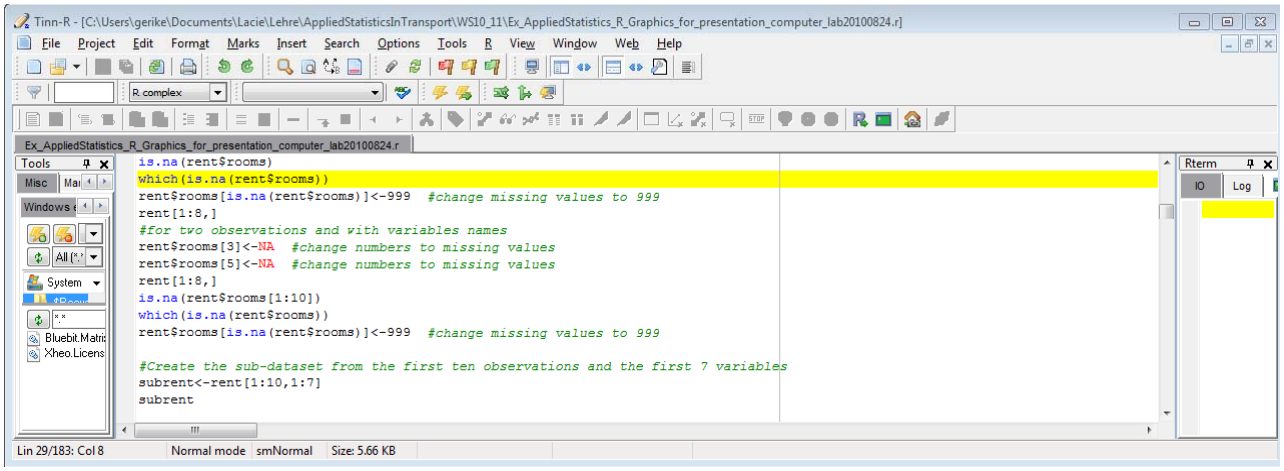
Run r-files: *source("name.r",echo=TRUE)*

Tinn-R

- Tinn is a small ASCII file editor primarily intended as a better replacement of the default Notepad under Windows. The name is the recursive acronym: Tinn is not Notepad.
- Tinn-R is an extension of Tinn that provides additional tools to control R.
- As such, Tinn-R is a feature-rich replacement of the basic script editor provided with Rgui. It provides syntax-highlighting, submission of code in whole, or line-by-line, and many other useful tools to ease writing and debugging of R code.

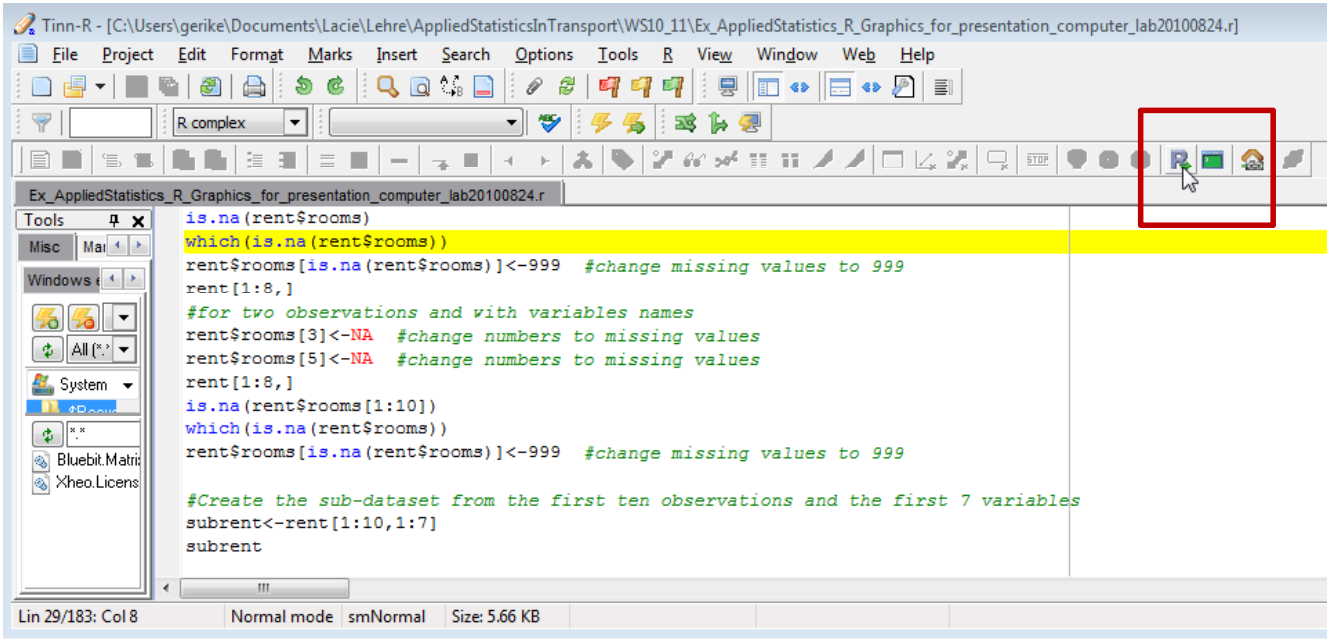
How to start R and Tinn-R on your computer!

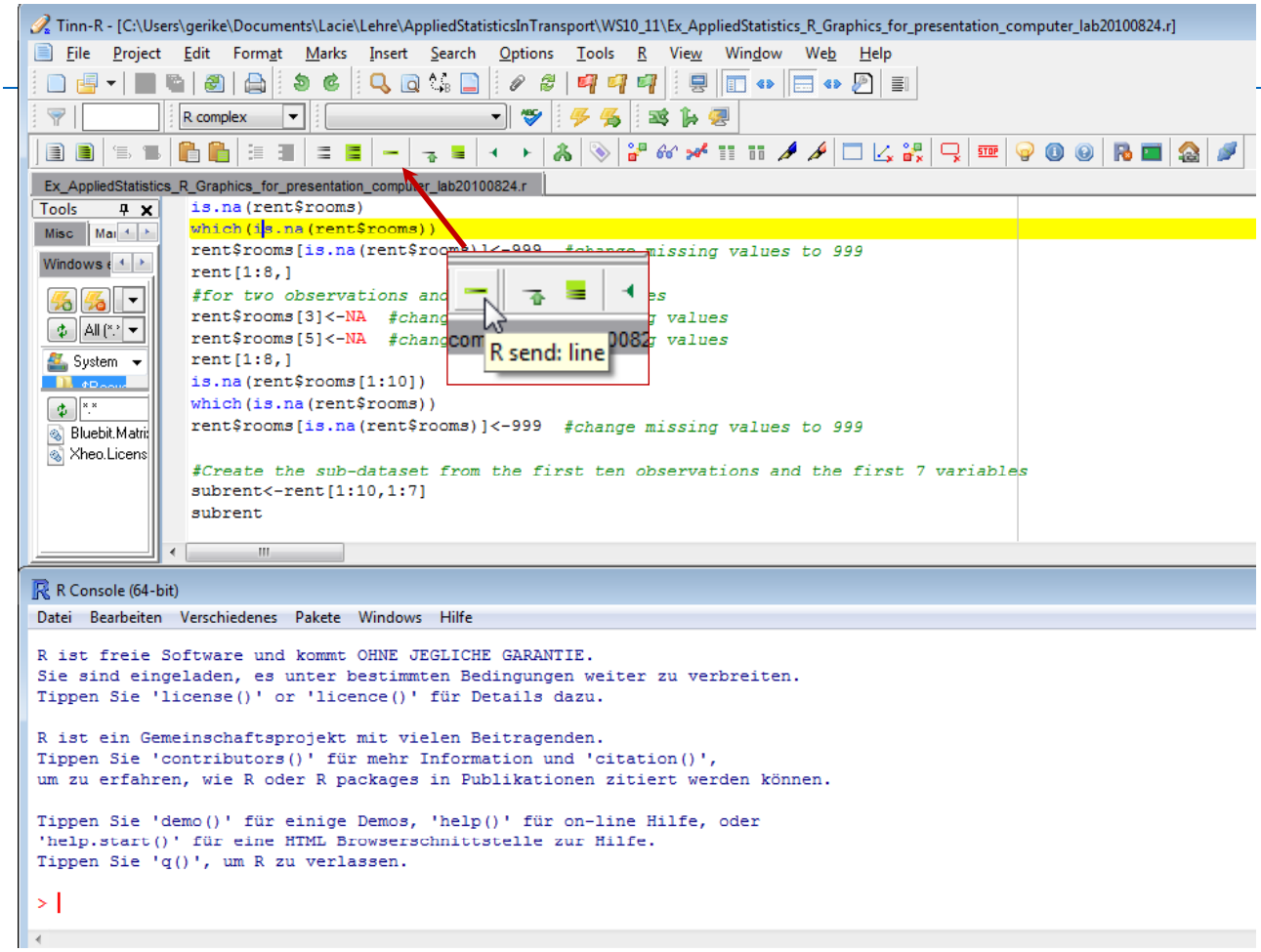
1. Start Tinn-R:
(or by starting an *.r file)



How to start R and Tinn-R on your computer

2. Start R from Tinn-R:

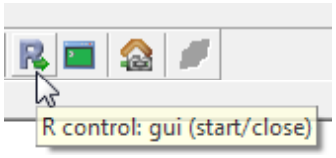




Please start R and Tinn-R on your computer!

1. Start Tinn-R:

2. Start R from Tinn-R:



R – Getting help, Installing packages

- *?mean*: description, usage, arguments, references, see also, examples
- *help.search("mean")* : help for a subject
- *find("mean")*: in what package is a specific command
- *apropos("mean")* : returns a character vector giving the names of all objects in the search list that match your (potentially partial) enquiry
- *example("mean")*
- *search()*: attached libraries and dataframes
- *data(package = .packages(all.available = TRUE)):* list the data sets in all available*packages
- *demo(graphics)*: demonstrations of R functions
- *demo(package = .packages(all.available = TRUE))*: all demos in currently available packages

Installing packages:

- *install.packages()*: You need administrator rights!
- *library()*: activate the package

Tidying up:

- *detach()*: detach dataframes
- *rm()*, *rm(list=ls())*: remove variables

R – Getting help

- Literature in the course reserve:
- Sá, Joaquim P. Marques ¬de (2007) Applied statistics using SPSS, STATISTICA, MATLAB and R
- Muenchen, Robert A. (2009) R for SAS and SPSS users
- Crawley, Michael J. (2009) The R book

- More books in the library
- A long list of books is at http://www.r-project.org/doc/bib/R-books.html
- Additional material at http://www.transportation.bv.tum.de/
- Or: just google

R as a calculator

```
1 + 1 #[1] 2
a <- 0.5
b <- a*2
exp(b) #[1] 2.718282
log(exp(b)) #[1] 1
c <- (cos(log(exp(b))) - a) / b + 1
c #[1] 1.040302
```

R – Important operators

- + - * / basic arithmetics
- ^ power
- = <- -> assignments
- : create sequences of whole numbers
- == != < > <= >= logical comparisons
- # comments

```
a <- 1:5
a #[1] 1 2 3 4 5
-a #[1] -1 -2 -3 -4 -5
a * 10 #[1] 10 20 30 40 50
a^3 #[1] 1 8 27 64 125
2^a #[1] 2 4 8 16 32
```

```
2 * 3^4 + 5 #[1] 167
(2 * (3^4)) + 5 #[1] 167
2^a < 6 #[1] TRUE TRUE FALSE FALSE FALSE
100 + 2^a < 6 #[1] FALSE FALSE FALSE FALSE FALSE
```

Variable names

- Each analysis can be saved in a variable.
- Variables can be named with any combinations of letters, numbers, dots and underscore.
- Variables names should not begin with number or symbols.
- Variables names should not contain blank spaces (rent.data, not rent data).
- R is case sensitive (x is not the same as X).

Good practice:

- Use longer, self-explanatory variable names.
- Do not calculate variables with the same name as a variables inside a dataframe.
- Always detach dataframes once you have finished working with them.
- Remove calculated variables from the work space once you are finished with them (rm()).

Functions in R

- R is a complete programming language, most of R is written in R, users can define new functions
- Writing functions in R is easy but NOT part of the exam
- Syntax: function(argument list) body

```
arithmetic.mean<-function(x) sum(x)/length(x)
x<-c(1,3,7,2,4,9,7,7,2,3,0)
arithmetic.mean(x) #[1] 4.090909
mean(x) #[1] 4.090909

med<-function(x){
odd.even<-length(x)%%2
if(odd.even==0)(sort(x)[length(x)/2]+sort(x)[1+length(x)/2])/2
else sort(x)[ceiling(length(x)/2)]
}
x<-c(1,3,7,2,4,9,7,7,2,3,0)
med(x) #[1] 3
median(x) #[1] 3
sort(x) #[1] 0 1 2 2 3 3 4 7 7 7 9
```

Vectors

- Use c() (concatenate) for creating vectors
- Also useful: seq() (sequences) und rep() (repeat)

```
v1 <- c(1, 3.14, 17)
v1 #[1] 1.00 3.14 17.00
v2 <- seq(from = -pi, to = pi, length = 5)
v2 #[1] -3.141593 -1.570796 0.000000 1.570796 3.141593
v3 <- rep(2, 5)
v3 #[1] 2 2 2 2 2
length(v3) #[1] 5
mode(v3) #[1] "numeric"
```

```
sequence(5) #[1] 1 2 3 4 5
sequence(5:1) #[1] 1 2 3 4 5 1 2 3 4 1 2 3 1 2 1
sequence(c(5,2,4)) #[1] 1 2 3 4 5 1 2 1 2 3 4
```

- Vectors of characters:

```
LETTERS  #[1] "A" "B" "C" "D" "E" "F" "G" "H" "I" "J" "K" "L" "M" "N" "O"
         #[16] "P" "Q" "R" "S" "T" "U" "V" "W" "X" "Y" "Z"
mode(LETTERS) #[1] "character"
Benutzer <- c("Hansi", "Seppl")
Benutzer #[1] "Hansi" "Seppl"
"Benutzer" #[1] "Benutzer"
Benutzer < "R" #[1] TRUE FALSE
"Benutzer" < "R" #[1] TRUE
```

Vectors

- Logical vectors:

```
x<-1:6 #[1] 1 2 3 4 5 6
x<4   #[1]  TRUE  TRUE  TRUE FALSE FALSE FALSE
all(x>0) #[1] TRUE
any(x<0) #[1] FALSE
sum(x<4) #[1] 3
```

- Accessing elements of vectors: with squared brackets, each vector element can get a name

```
v2 <- seq(from = -pi, to = pi, length = 5)
v2 #[1] -3.141593 -1.570796 0.000000 1.570796 3.141593
v2[5] #[1] 3.141593
v2[2:4] #[1] -1.570796 0.000000 1.570796
v2[-(2:4)] #[1] -3.141593 3.141593
v2[v2<0] #[1] -3.141593 -1.570796
```

```
const <- c(e = exp(1), pi = pi, twopi = 2 * pi)
const
#e pi twopi
#2.718282 3.141593 6.283185
names(const) #[1] "e" "pi" "twopi"
names(const)[3] <- "2pi"
const
#e pi 2pi
#2.718282 3.141593 6.283185
const["2pi"]
#2pi
#6.283185
const[c("pi","2pi")]
#pi 2pi
#3.141593 6.283185
```

Factors

- Nominal and ordinal data are called factors in R, with labels for each level
- Characters can be changed to factors with factor()

```
treatment <- rep(c("control", "drug"), c(2,3))
treatment #[1] "control" "control" "drug" "drug" "drug"
summary(treatment)
#Length   Class     Mode
#   5   character character
treatment <- factor(treatment)
treatment
#[1] control control drug drug drug
#Levels: control drug
summary(treatment)
#control drug
# 2       3
```

- Ordered factors for ordinal variables:

```
rank.test <- ordered(c("good", "bad", "super", "super", "don't ask", "good"),
levels = c("don't ask", "bad", "good", "super"))
rank.test
#[1] good      bad       super     super     don't ask good
#Levels: don't ask < bad < good < super
```

Rownames, colnames

```
x<-matrix(rpois(20,1.5),nrow=4)  #rpois(n, lambda)
#random generation for the Poisson distribution with parameter lambda.
rownames(x)<-rownames(x,do.NULL=FALSE,prefix="Trial.")
x
drug.names<-c("aspirin","p","n","h","placebo")
x
colnames(x)<-drug.names
x
x<-rbind(x,apply(x,2,mean))
x
x<-cbind(x,apply(x,1,var))
x
colnames(x)[6]
rownames(x)[5]
colnames(x)[6]<-c("Mean")
rownames(x)<-c(1:4,"Variance")
#          aspirin p n   h placebo  Mean
#1             2.0 2 2 1.0       0 0.800
#2             3.0 1 1 2.0       2 0.700
#3             4.0 1 1 0.0       1 2.300
#4             1.0 0 4 3.0       1 2.700
#Variance      2.5 1 2 1.5       1 0.425
```

Dataframes

- Probably most important structure, variables in columns, observations in rows

```
data("iris")
class(iris) #[1] "data.frame"
head(iris)
#  Sepal.Length Sepal.Width Petal.Length Petal.W
#1          5.1         3.5          1.4
#2          4.9         3.0          1.4
#3          4.7         3.2          1.3
#4          4.6         3.1          1.5
#5          5.0         3.6          1.4
#6          5.4         3.9          1.7
```

```
dim(iris) #[1] 150 5
summary(iris[1:25,1:2])
summary(iris[,-(1:2)]) #without the first to variables
#Petal.Length Petal.Width Species
#Min. :1.000 Min. :0.100 setosa :50
#1st Qu.:1.600 1st Qu.:0.300 versicolor:50
#Median :4.350 Median :1.300 virginica :50
#Mean :3.758 Mean :1.199
#3rd Qu.:5.100 3rd Qu.:1.800
#Max. :6.900 Max. :2.500
iris$Petal.Length # [1] 1.4 1.4 1.3 1.5 1.4 1.7 1.4 ...
```

```
summary(iris)
#  Sepal.Length    Sepal.Width     Petal.Length    Petal.Width          Species
# Min.   :4.300   Min.   :2.000   Min.   :1.000   Min.   :0.100   setosa     :50
# 1st Qu.:5.100   1st Qu.:2.800   1st Qu.:1.600   1st Qu.:0.300   versicolor:50
# Median :5.800   Median :3.000   Median :4.350   Median :1.300   virginica :50
# Mean   :5.843   Mean   :3.057   Mean   :3.758   Mean   :1.199
# 3rd Qu.:6.400   3rd Qu.:3.300   3rd Qu.:5.100   3rd Qu.:1.800
# Max.   :7.900   Max.   :4.400   Max.   :6.900   Max.   :2.500
```

```
str(iris)
#'data.frame':   150 obs. of  5 variables:
# $ Sepal.Length: num  5.1 4.9 4.7 4.6 5 5.4 4.6 5 4.4 4.9 ...
# $ Sepal.Width : num  3.5 3 3.2 3.1 3.6 3.9 3.4 3.4 2.9 3.1 ...
# $ Petal.Length: num  1.4 1.4 1.3 1.5 1.4 1.7 1.4 1.5 1.4 1.5 ...
# $ Petal.Width : num  0.2 0.2 0.2 0.2 0.2 0.4 0.3 0.2 0.2 0.1 ...
# $ Species     : Factor w/ 3 levels "setosa","versicolor",..: 1 1 1 1 1 1 1 1 1 1 ...
```

Data input and output

- Easiest with text files: one column per variable, one row per observation, use clear seperators
- *scan()* for directly entering data
- *read.table(), read.csv()*, create dataframes
- *sep=„\t“* for separators
- *header=T/N*
- Set working directory
- *setwd("c:\\Documents\\AppliedStatisticsInTransport\\WS10_11\\")*
- *setwd("c:/Documents/AppliedStatisticsInTransport/WS10_11/")*
- *getwd()*
- *rent<-read.table("rent.asc", header=T, sep="\t")*
- Browsing for files:
- *data<-read.table(file.choose(),header=T)*
- Build in data:
- *data()*: Data sets in package ‘datasets’
- *data(package = .packages(all.available = TRUE))*: list the data sets in all available packages

Data output, saving your work

- *save(test, file=„test.Rdata")*
- *load(„test.Rdata")* – load the data in the next session
- or as text file:
- *write.table(), write.csv()*
- *writeClipboard:* writing data from R to the clipboard

Vector functions

- *min(x), max(x), mean(x), sum(x), median(x), range(x)*
- *var(x), cor(x,y), sort(x), order(x)*
- *quantile(x)* vector containing the minimum, lower quartile, median, upper quartile, maximum of x
- *cumsum(x)* vector containing the sum of all the elements up to that point
- *cumprod(x)* vector containing the product of all the elements up to that point
- *cummax(x)* vector of non-decreasing numbers which are the cumulative maxima of the values in x up to that point
- *cummin(x)* vector of non-increasing numbers which are the cumulative minima of the values in x up to that point
- *colMeans(x)* column means of dataframe or matrix x
- *colSums(x)* column totals of dataframe or matrix x
- *rowMeans(x)* row means of dataframe or matrix x
- *rowSums(x)* row totals of dataframe or matrix x

```
x<-c(1,5,4,2,8,7,5,3)
sort(x) #[1] 1 2 3 4 5 5 7 8
order(x) #[1] 1 4 8 3 2 7 6 5
cummax(x) #[1] 1 5 5 5 8 8 8 8
cummin(x) #[1] 1 1 1 1 1 1 1 1
```

Mathematical functions

- *abs(x)*: absolute value of x (ignore the minus sign if there is one)
- *sqrt(x)*: square root of x
- *round(x, digits=0), floor(x), ceiling(x)*: rounding numbers
- *sum(x), prod(x)*: sum and product
- *factorial(x)*: x!
- *log(x), log(x,n), log10(x)*: Logarithms
- *exp(x)*: antilog of x, exponential function
- *sin(x), cos(x), tan(x)*: trigonometric functions

- *pi*: the number pi
- *Inf*: infinity
- *NaN*: Not a Number, e.g. 0/0, Inf-Inf, Inf/Inf
- *NA*: Not available, missing values
- *NULL*: empty set