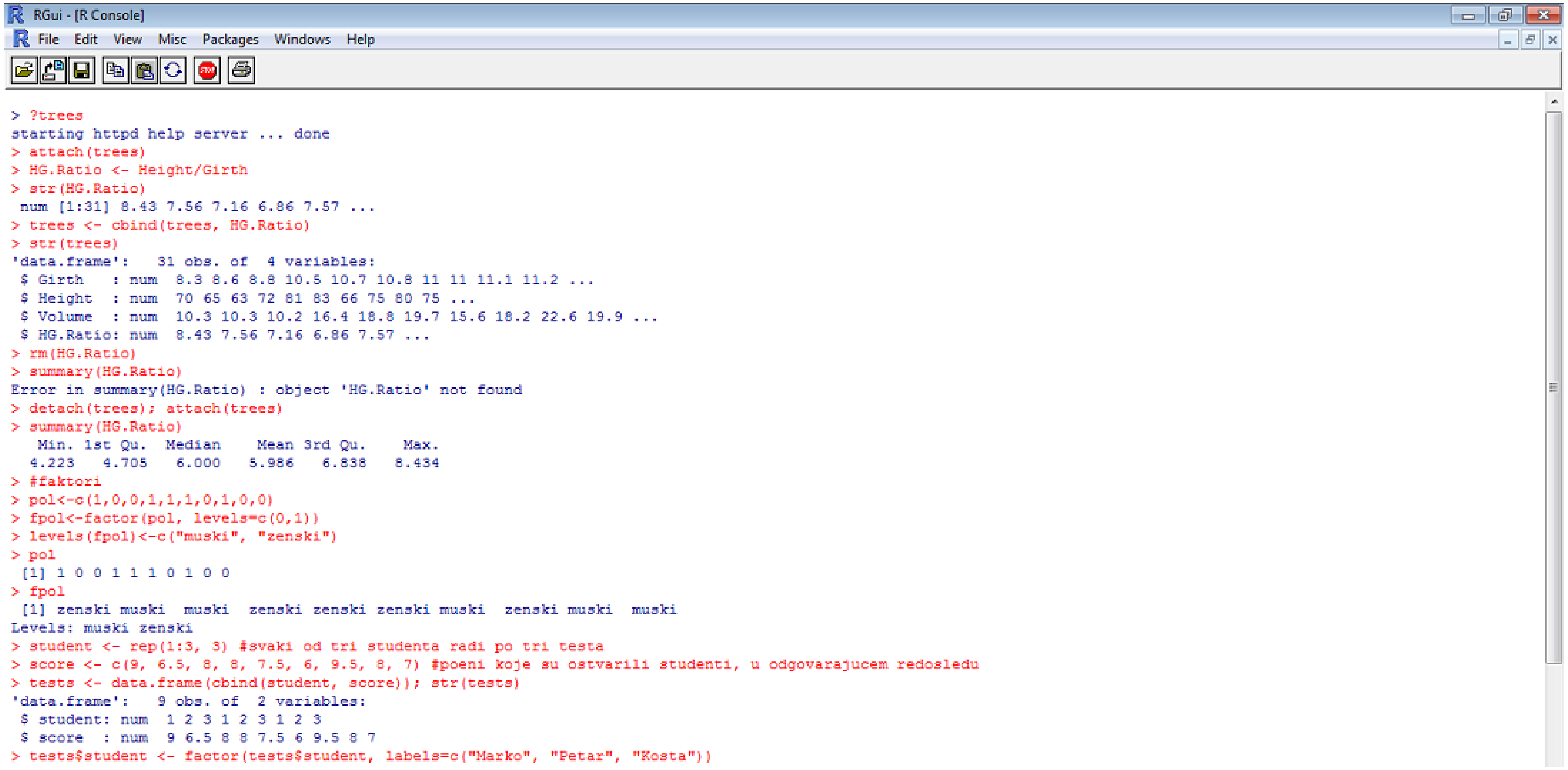
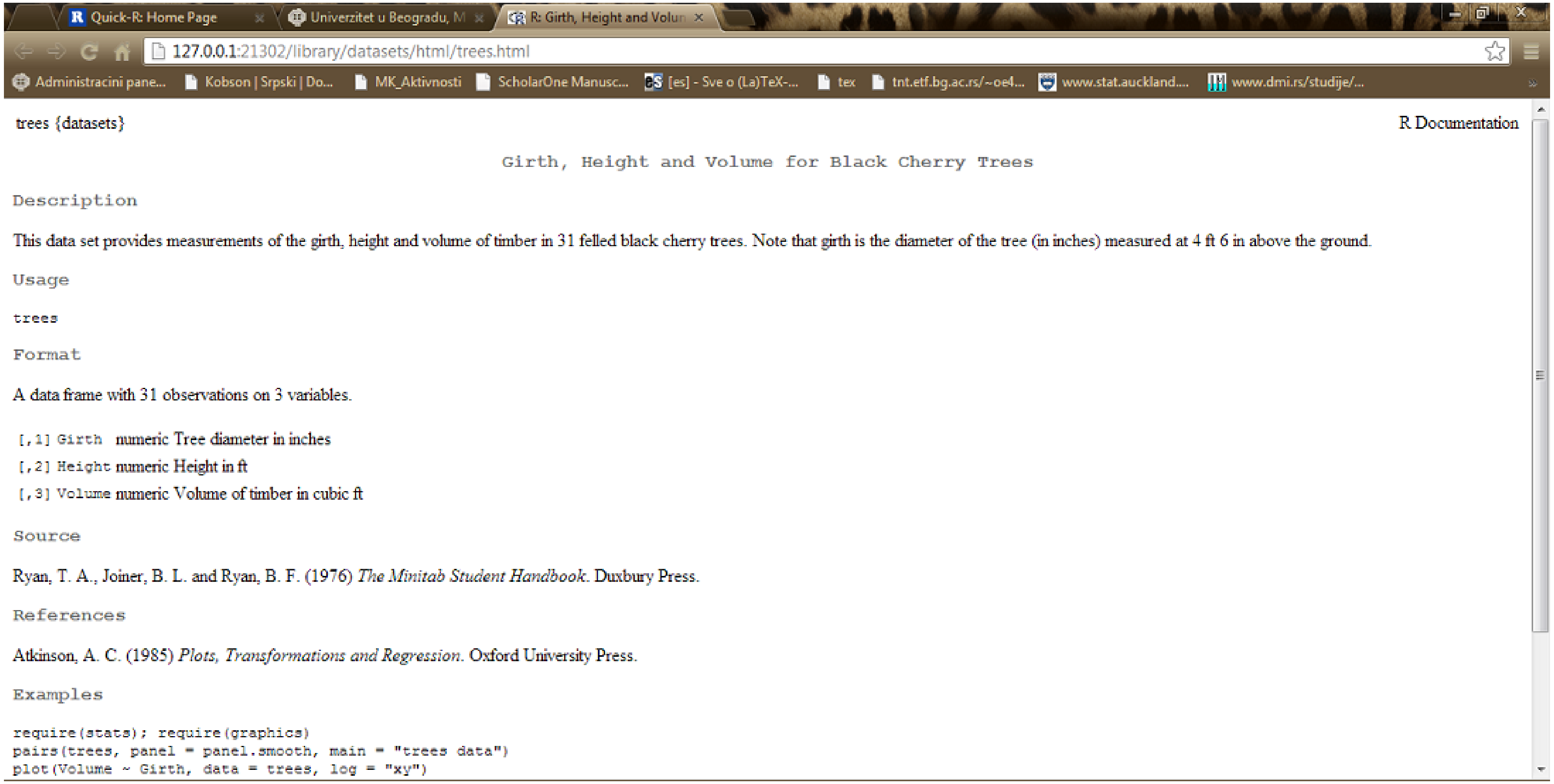


```
> ?trees
starting httpd help server ... done
> attach(trees)
> HG.Ratio <- Height/Girth
> str(HG.Ratio)
 num [1:31] 8.43 7.56 7.16 6.86 7.57 ...
> trees <- cbind(trees, HG.Ratio)
> str(trees)
'data.frame':   31 obs. of  4 variables:
 $ Girth   : num  8.3 8.6 8.8 10.5 10.7 10.8 11 11 11.1 11.2 ...
 $ Height  : num  70 65 63 72 81 83 66 75 80 75 ...
 $ Volume  : num  10.3 10.3 10.2 16.4 18.8 19.7 15.6 18.2 22.6 19.9 ...
 $ HG.Ratio: num  8.43 7.56 7.16 6.86 7.57 ...
> rm(HG.Ratio)
> summary(HG.Ratio)
Error in summary(HG.Ratio) : object 'HG.Ratio' not found
> detach(trees); attach(trees)
> summary(HG.Ratio)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  4.223   4.705   6.000   5.986   6.838   8.434
> #faktori
> pol<-c(1,0,0,1,1,1,0,1,0,0)
> fpol<-factor(pol, levels=c(0,1))
> levels(fpol)<-c("muski", "zenski")
> pol
 [1] 1 0 0 1 1 1 0 1 0 0
> fpol
 [1] zenski muski  muski  zenski zenski zenski muski  zenski muski  muski
Levels: muski zenski
> student <- rep(1:3, 3) #svaki od tri studenta radi po tri testa
> score <- c(9, 6.5, 8, 8, 7.5, 6, 9.5, 8, 7) #poeni koje su ostvarili studenti, u odgovarajucem redosledu
> tests <- data.frame(cbind(student, score)); str(tests)
'data.frame':   9 obs. of  2 variables:
 $ student: num  1 2 3 1 2 3 1 2 3
 $ score  : num  9 6.5 8 8 7.5 6 9.5 8 7
> tests$student <- factor(tests$student, labels=c("Marko", "Petar", "Kosta"))
```



trees {datasets}   R Documentation

Girth, Height and Volume for Black Cherry Trees

Description

This data set provides measurements of the girth, height and volume of timber in 31 felled black cherry trees. Note that girth is the diameter of the tree (in inches) measured at 4 ft 6 in above the ground.

Usage

```
trees
```

Format

A data frame with 31 observations on 3 variables.

`[,1]`	`Girth`	numeric Tree diameter in inches
`[,2]`	`Height`	numeric Height in ft
`[,3]`	`Volume`	numeric Volume of timber in cubic ft

Source

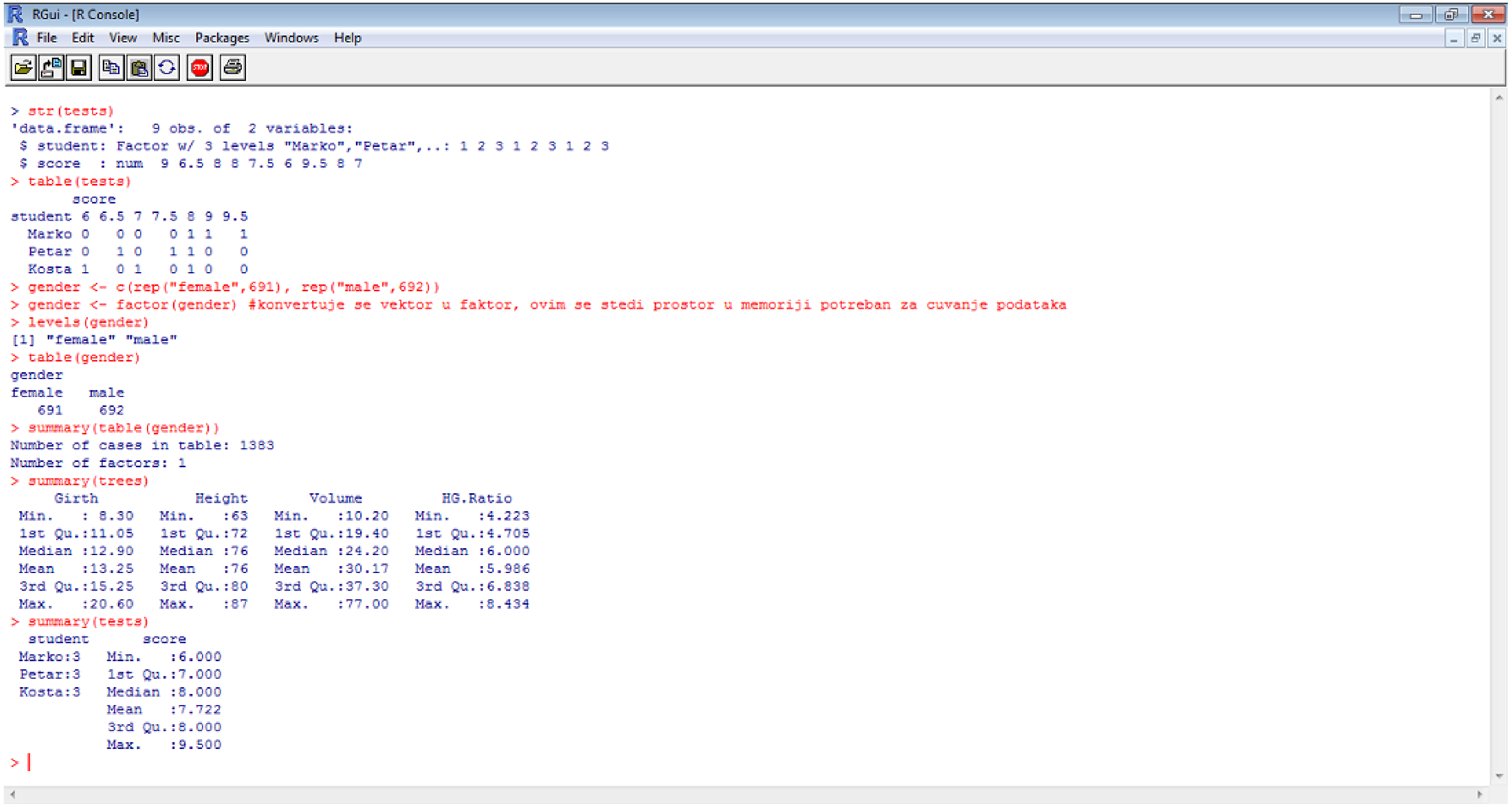
Ryan, T. A., Joiner, B. L. and Ryan, B. F. (1976) *The Minitab Student Handbook*. Duxbury Press.

References

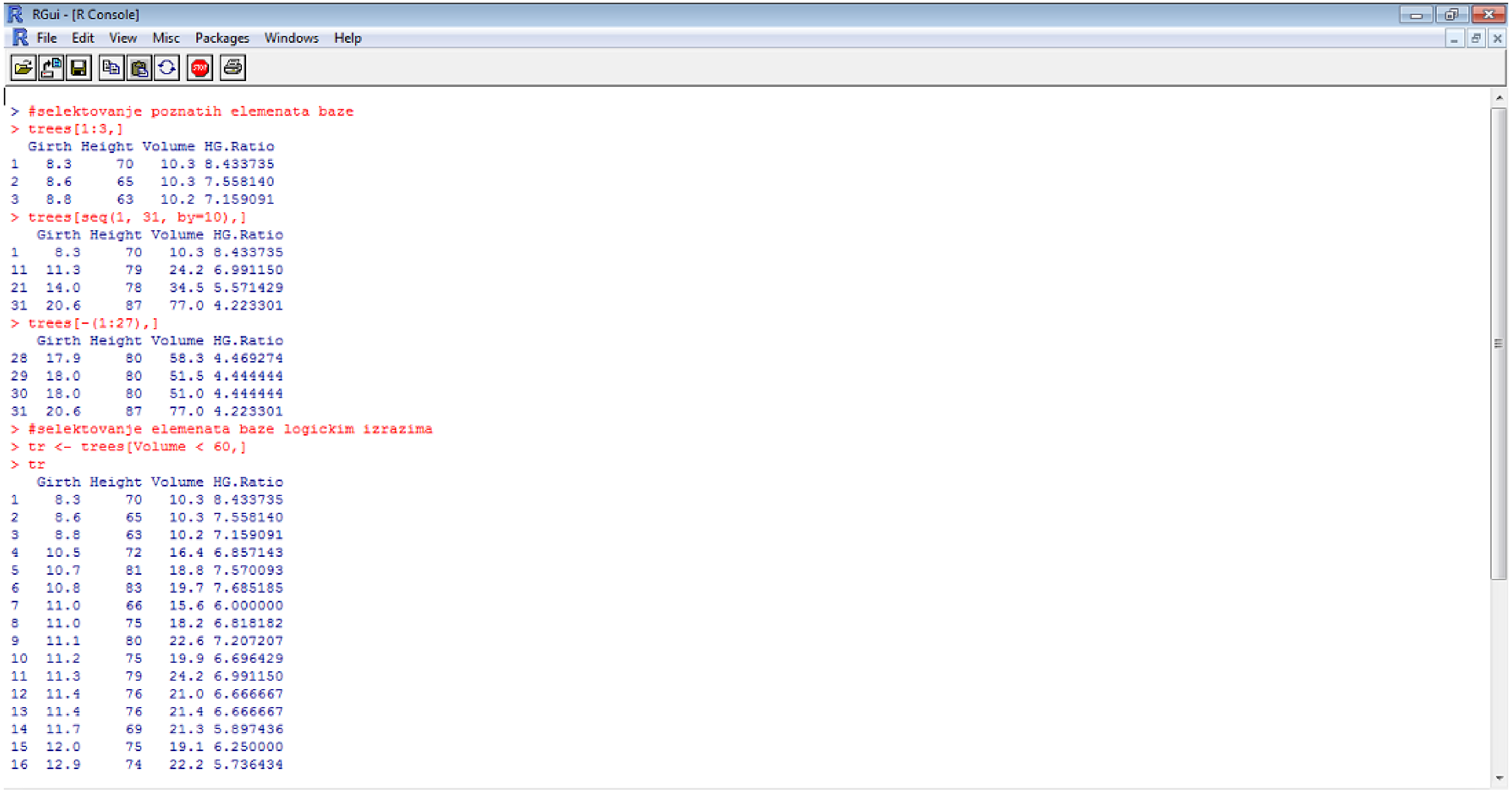
Atkinson, A. C. (1985) *Plots, Transformations and Regression*. Oxford University Press.

Examples

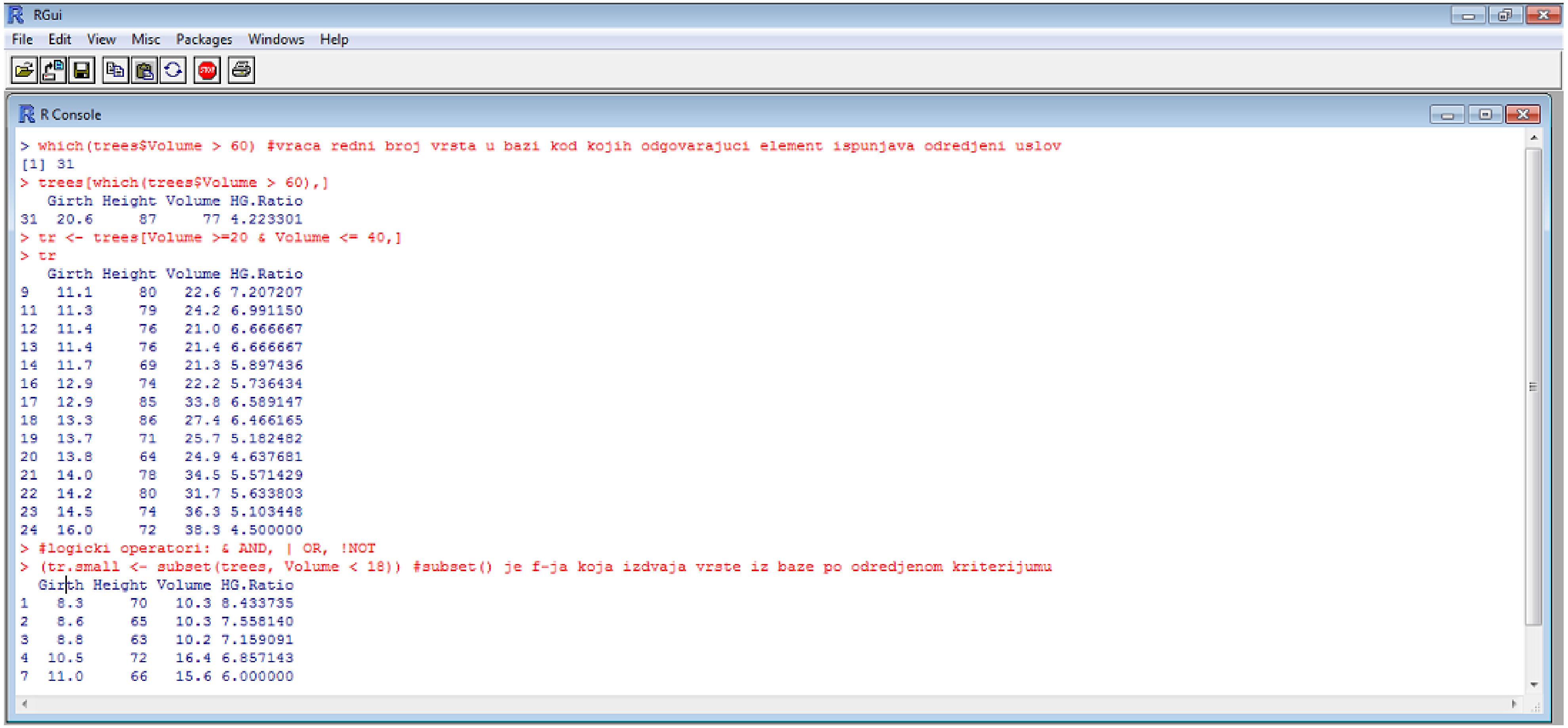
```
require(stats); require(graphics)
pairs(trees, panel = panel.smooth, main = "trees data")
plot(Volume ~ Girth, data = trees, log = "xy")
```



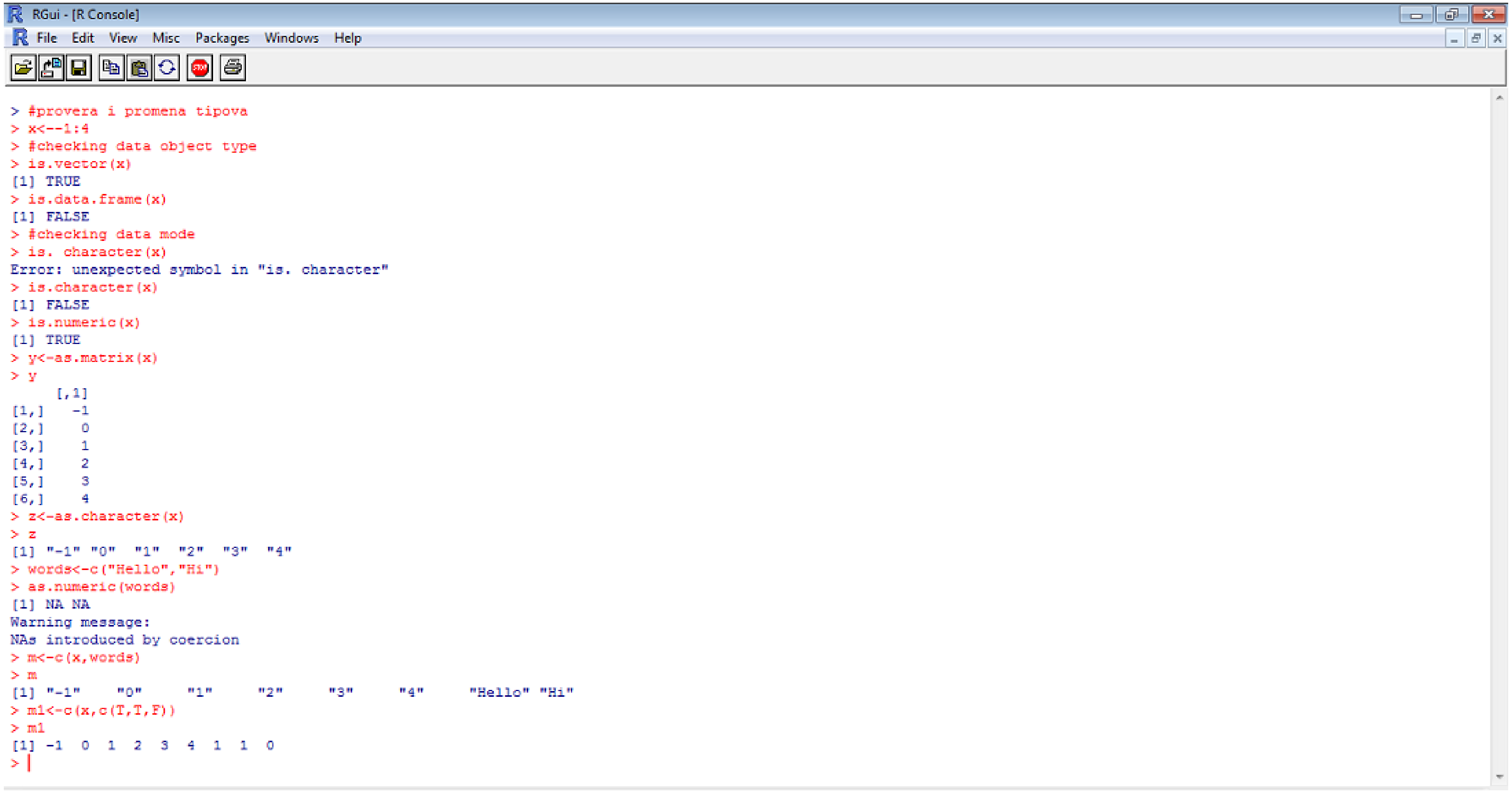
```
> str(tests)
'data.frame':   9 obs. of  2 variables:
 $ student: Factor w/ 3 levels "Marko","Petar",..: 1 2 3 1 2 3 1 2 3
 $ score  : num  9 6.5 8 8 7.5 6 9.5 8 7
> table(tests)
       score
student 6 6.5 7 7.5 8 9 9.5
  Marko 0   0 0   0 1 1   1
  Petar 0   1 0   1 1 0   0
  Kosta 1   0 1   0 1 0   0
> gender <- c(rep("female",691), rep("male",692))
> gender <- factor(gender) #konvertuje se vektor u faktor, ovim se stedi prostor u memoriji potreban za cuvanje podataka
> levels(gender)
[1] "female" "male"
> table(gender)
gender
female   male
   691    692
> summary(table(gender))
Number of cases in table: 1383
Number of factors: 1
> summary(trees)
     Girth           Height       Volume         HG.Ratio
 Min.   : 8.30   Min.   :63   Min.   :10.20   Min.   :4.223
 1st Qu.:11.05   1st Qu.:72   1st Qu.:19.40   1st Qu.:4.705
 Median :12.90   Median :76   Median :24.20   Median :6.000
 Mean   :13.25   Mean   :76   Mean   :30.17   Mean   :5.986
 3rd Qu.:15.25   3rd Qu.:80   3rd Qu.:37.30   3rd Qu.:6.838
 Max.   :20.60   Max.   :87   Max.   :77.00   Max.   :8.434
> summary(tests)
  student      score
 Marko:3   Min.   :6.000
 Petar:3   1st Qu.:7.000
 Kosta:3   Median :8.000
           Mean   :7.722
           3rd Qu.:8.000
           Max.   :9.500
> |
```



```
> #selektovanje poznatih elemenata baze
> trees[1:3,]
  Girth Height Volume HG.Ratio
1   8.3     70   10.3 8.433735
2   8.6     65   10.3 7.558140
3   8.8     63   10.2 7.159091
> trees[seq(1, 31, by=10),]
   Girth Height Volume HG.Ratio
1    8.3     70   10.3 8.433735
11  11.3     79   24.2 6.991150
21  14.0     78   34.5 5.571429
31  20.6     87   77.0 4.223301
> trees[-(1:27),]
   Girth Height Volume HG.Ratio
28  17.9     80   58.3 4.469274
29  18.0     80   51.5 4.444444
30  18.0     80   51.0 4.444444
31  20.6     87   77.0 4.223301
> #selektovanje elemenata baze logickim izrazima
> tr <- trees[Volume < 60,]
> tr
   Girth Height Volume HG.Ratio
1    8.3     70   10.3 8.433735
2    8.6     65   10.3 7.558140
3    8.8     63   10.2 7.159091
4   10.5     72   16.4 6.857143
5   10.7     81   18.8 7.570093
6   10.8     83   19.7 7.685185
7   11.0     66   15.6 6.000000
8   11.0     75   18.2 6.818182
9   11.1     80   22.6 7.207207
10  11.2     75   19.9 6.696429
11  11.3     79   24.2 6.991150
12  11.4     76   21.0 6.666667
13  11.4     76   21.4 6.666667
14  11.7     69   21.3 5.897436
15  12.0     75   19.1 6.250000
16  12.9     74   22.2 5.736434
```



```
> which(trees$Volume > 60) #vraca redni broj vrsta u bazi kod kojih odgovarajuci element ispunjava odredjeni uslov
[1] 31
> trees[which(trees$Volume > 60),]
   Girth Height Volume HG.Ratio
31  20.6     87     77 4.223301
> tr <- trees[Volume >=20 & Volume <= 40,]
> tr
   Girth Height Volume HG.Ratio
9   11.1     80   22.6 7.207207
11  11.3     79   24.2 6.991150
12  11.4     76   21.0 6.666667
13  11.4     76   21.4 6.666667
14  11.7     69   21.3 5.897436
16  12.9     74   22.2 5.736434
17  12.9     85   33.8 6.589147
18  13.3     86   27.4 6.466165
19  13.7     71   25.7 5.182482
20  13.8     64   24.9 4.637681
21  14.0     78   34.5 5.571429
22  14.2     80   31.7 5.633803
23  14.5     74   36.3 5.103448
24  16.0     72   38.3 4.500000
> #logicki operatori: & AND, | OR, !NOT
> (tr.small <- subset(trees, Volume < 18)) #subset() je f-ja koja izdvaja vrste iz baze po odredjenom kriterijumu
  Girth Height Volume HG.Ratio
1   8.3     70   10.3 8.433735
2   8.6     65   10.3 7.558140
3   8.8     63   10.2 7.159091
4  10.5     72   16.4 6.857143
7  11.0     66   15.6 6.000000
```



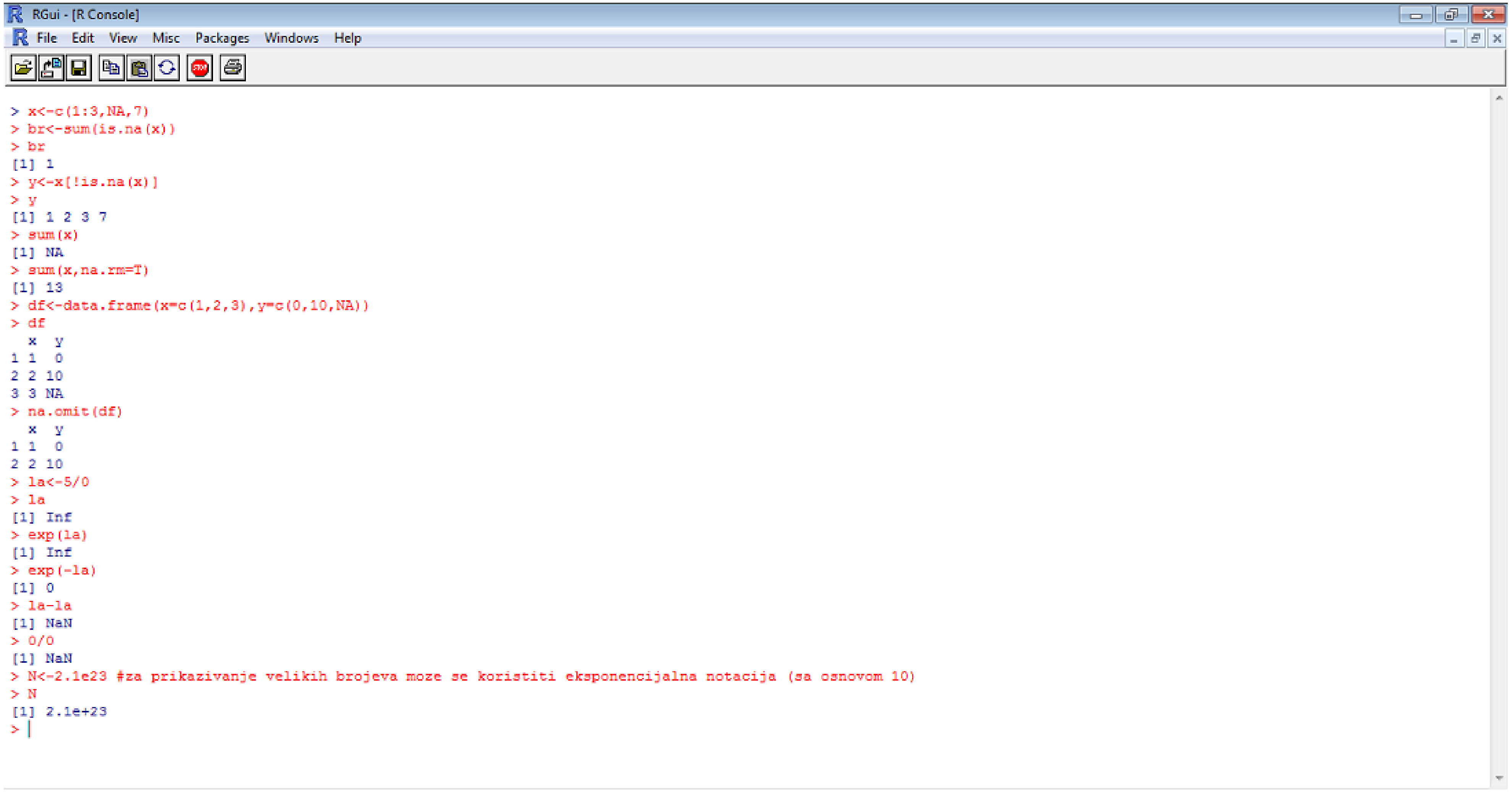
```
> #provera i promena tipova
> x<--1:4
> #checking data object type
> is.vector(x)
[1] TRUE
> is.data.frame(x)
[1] FALSE
> #checking data mode
> is. character(x)
Error: unexpected symbol in "is. character"
> is.character(x)
[1] FALSE
> is.numeric(x)
[1] TRUE
> y<-as.matrix(x)
> y
     [,1]
[1,]   -1
[2,]    0
[3,]    1
[4,]    2
[5,]    3
[6,]    4
> z<-as.character(x)
> z
[1] "-1" "0"  "1"  "2"  "3"  "4"
> words<-c("Hello","Hi")
> as.numeric(words)
[1] NA NA
Warning message:
NAs introduced by coercion
> m<-c(x,words)
> m
[1] "-1"    "0"     "1"     "2"     "3"     "4"     "Hello" "Hi"
> m1<-c(x,c(T,T,F))
> m1
[1] -1  0  1  2  3  4  1  1  0
> 
```

The following table gives an overview of the type of objects representing data.

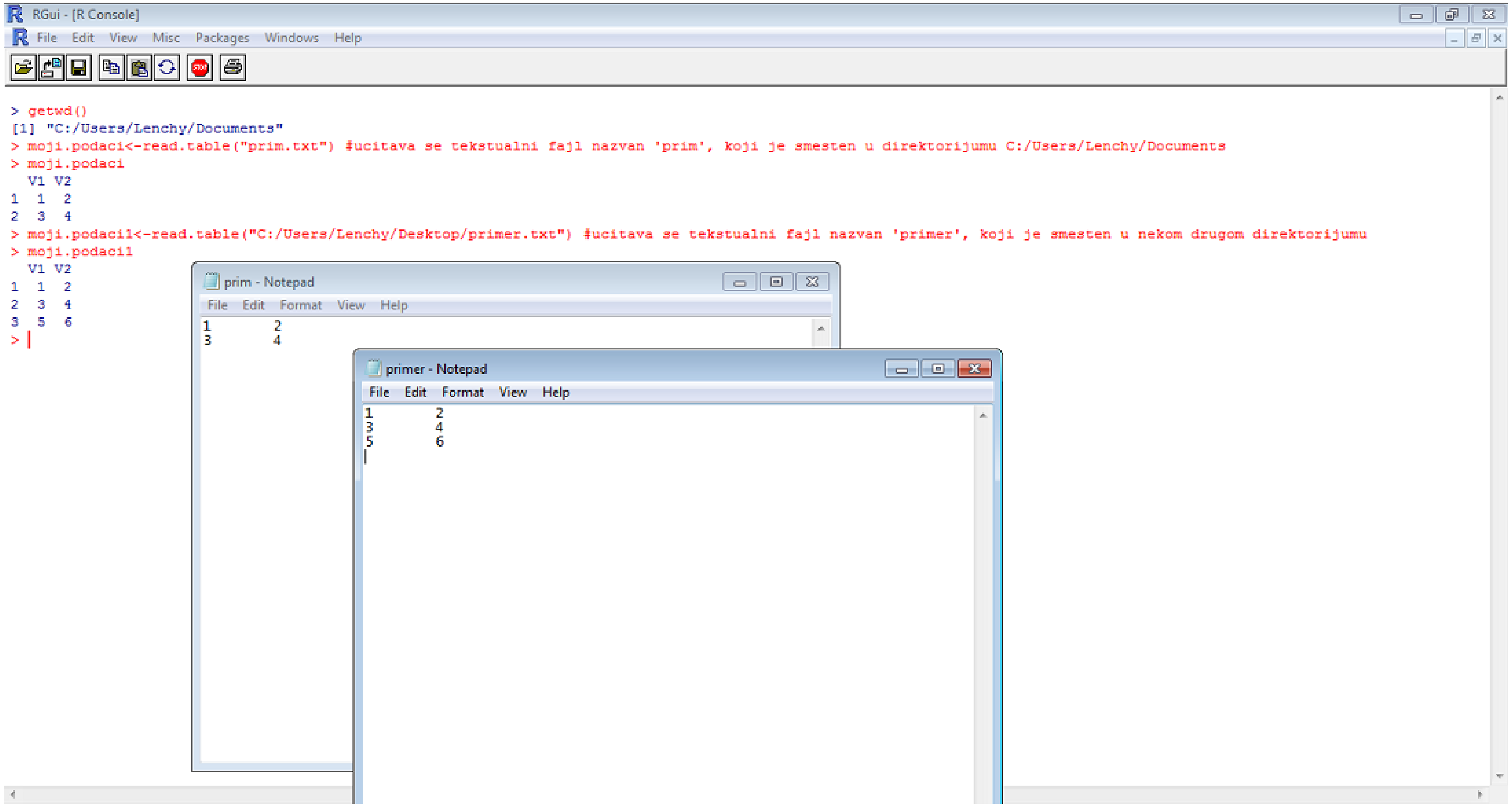
object	**modes**	**several modes possible in the same object?**
vector	numeric, character, complex *or* logical	No
factor	numeric *or* character	No
array	numeric, character, complex *or* logical	No
matrix	numeric, character, complex *or* logical	No
data frame	numeric, character, complex *or* logical	Yes
ts	numeric, character, complex *or* logical	No
list	numeric, character, complex, logical, function, expression, . . .	Yes

A vector is a variable in the commonly admitted meaning. A factor is a categorical variable. An array is a table with k dimensions, a matrix being a particular case of array with $k = 2$. Note that the elements of an array or of a matrix are all of the same mode. A data frame is a table composed with one or several vectors and/or factors all of the same length but possibly of different modes. A 'ts' is a time series data set and so contains additional attributes such as frequency and dates. Finally, a list can contain any type of object, included lists!

For a vector, its mode and length are sufficient to describe the data. For other objects, other information is necessary and it is given by *non-intrinsic* attributes. Among these attributes, we can cite *dim* which corresponds to the dimensions of an object. For example, a matrix with 2 lines and 2 columns has for `dim` the pair of values [2, 2], but its length is 4.



```
> x<-c(1:3,NA,7)
> br<-sum(is.na(x))
> br
[1] 1
> y<-x[!is.na(x)]
> y
[1] 1 2 3 7
> sum(x)
[1] NA
> sum(x,na.rm=T)
[1] 13
> df<-data.frame(x=c(1,2,3),y=c(0,10,NA))
> df
  x  y
1 1  0
2 2 10
3 3 NA
> na.omit(df)
  x  y
1 1  0
2 2 10
> la<-5/0
> la
[1] Inf
> exp(la)
[1] Inf
> exp(-la)
[1] 0
> la-la
[1] NaN
> 0/0
[1] NaN
> N<-2.1e23 #za prikazivanje velikih brojeva moze se koristiti eksponencijalna notacija (sa osnovom 10)
> N
[1] 2.1e+23
> 
```



```
> getwd()
[1] "C:/Users/Lenchy/Documents"
> moji.podaci<-read.table("prim.txt") #ucitava se tekstualni fajl nazvan 'prim', koji je smesten u direktorijumu C:/Users/Lenchy/Documents
> moji.podaci
  V1 V2
1  1  2
2  3  4
> moji.podaci1<-read.table("C:/Users/Lenchy/Desktop/primer.txt") #ucitava se tekstualni fajl nazvan 'primer', koji je smesten u nekom drugom direktorijumu
> moji.podaci1
  V1 V2
1  1  2
2  3  4
3  5  6
> 
```

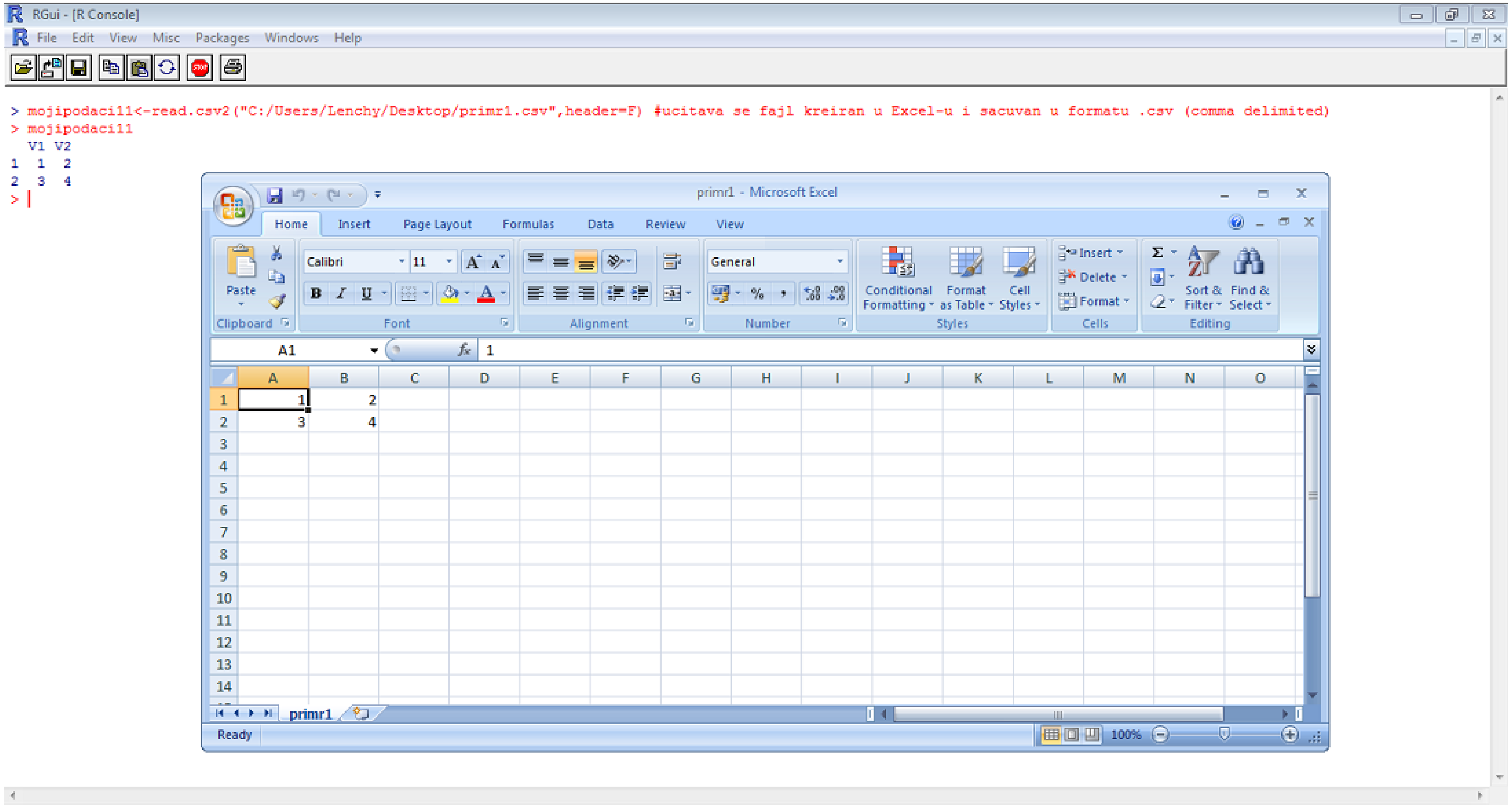
prim - Notepad

```
1	2
3	4
```

primer - Notepad

```
1	2
3	4
5	6
```

RGui - [R Console]
File Edit View Misc Packages Windows Help
File Edit Format View Help



```
> mojipodaci11<-read.csv2("C:/Users/Lenchy/Desktop/primr1.csv",header=F) #ucitava se fajl kreiran u Excel-u i sacuvan u formatu .csv (comma delimited)
> mojipodaci11
  V1 V2
1  1  2
2  3  4
> 
```

	A	B
1	1	2
2	3	4

RGui - [R Console]
File Edit View Misc Packages Windows Help
primr1 - Microsoft Excel
Home Insert Page Layout Formulas Data Review View
A1
primr1
Ready

```
read.table(file, header = FALSE, sep = "", quote = "\"'", dec = "

           row.names, col.names, as.is = FALSE, na.strings = "NA",
           colClasses = NA, nrows = -1,
           skip = 0, check.names = TRUE, fill = !blank.lines.skip,
           strip.white = FALSE, blank.lines.skip = TRUE,
           comment.char = "#")
```

`file`	the name of the file (within `""` or a variable of mode character), possibly with its path (the symbol \ is not allowed and must be replaced by /, even under Windows), or a remote access to a file of type URL (http://...)
`header`	a logical (`FALSE` or `TRUE`) indicating if the file contains the names of the variables on its first line
`sep`	the field separator used in the file, for instance `sep="\t"` if it is a tabulation
`quote`	the characters used to cite the variables of mode character
`dec`	the character used for the decimal point
`row.names`	a vector with the names of the lines which can be either a vector of mode character, or the number (or the name) of a variable of the file (by default: `1`, `2`, `3`, . . .)
`col.names`	a vector with the names of the variables (by default: `V1`, `V2`, `V3`, . . .)
`as.is`	controls the conversion of character variables as factors (if `FALSE`) or keeps them as characters (`TRUE`); `as.is` can be a logical, numeric or character vector specifying the variables to be kept as character
`na.strings`	the value given to missing data (converted as `NA`)
`colClasses`	a vector of mode character giving the classes to attribute to the columns
`nrows`	the maximum number of lines to read (negative values are ignored)
`skip`	the number of lines to be skipped before reading the data
`check.names`	if `TRUE`, checks that the variable names are valid for R
`fill`	if `TRUE` and all lines do not have the same number of variables, "blanks" are added
`strip.white`	(conditional to `sep`) if `TRUE`, deletes extra spaces before and after the character variables
`blank.lines.skip`	if `TRUE`, ignores "blank" lines
`comment.char`	a character defining comments in the data file, the rest of the line after this character is ignored (to disable this argument, use `comment.char = ""`)

The variants of `read.table` are useful since they have different default values:

```
read.csv(file, header = TRUE, sep = ",", quote="\"", dec=".",
         fill = TRUE, ...)
read.csv2(file, header = TRUE, sep = ";", quote="\"", dec=",",
          fill = TRUE, ...)
```

Неке важне функције:

The functions available in R for manipulating data are too many to be listed here. One can find all the basic mathematical functions (`log`, `exp`, `log10`, `log2`, `sin`, `cos`, `tan`, `asin`, `acos`, `atan`, `abs`, `sqrt`, ...), special functions (`gamma`, `digamma`, `beta`, `besselI`, ...), as well as diverse functions useful in statistics. Some of these functions are listed in the following table.

`sum(x)`	sum of the elements of `x`
`prod(x)`	product of the elements of `x`
`max(x)`	maximum of the elements of `x`
`min(x)`	minimum of the elements of `x`
`which.max(x)`	returns the index of the greatest element of `x`
`which.min(x)`	returns the index of the smallest element of `x`
`range(x)`	id. than `c(min(x), max(x))`
`length(x)`	number of elements in `x`
`mean(x)`	mean of the elements of `x`
`median(x)`	median of the elements of `x`
`var(x)` or `cov(x)`	variance of the elements of `x` (calculated on $n-1$); if `x` is a matrix or a data frame, the variance-covariance matrix is calculated
`cor(x)`	correlation matrix of `x` if it is a matrix or a data frame (1 if `x` is a vector)
`var(x, y)` or `cov(x, y)`	covariance between `x` and `y`, or between the columns of `x` and those of `y` if they are matrices or data frames
`cor(x, y)`	linear correlation between `x` and `y`, or correlation matrix if they are matrices or data frames

These functions return a single value (thus a vector of length one), except `range` which returns a vector of length two, and `var`, `cov`, and `cor` which may return a matrix. The following functions return more complex results.

`round(x, n)`	rounds the elements of `x` to `n` decimals
`rev(x)`	reverses the elements of `x`
`sort(x)`	sorts the elements of `x` in increasing order; to sort in decreasing order: `rev(sort(x))`
`rank(x)`	ranks of the elements of `x`

`log(x, base)`	computes the logarithm of `x` with base `base`
`scale(x)`	if `x` is a matrix, centers and reduces the data; to center only use the option `center=FALSE`, to reduce only `scale=FALSE` (by default `center=TRUE, scale=TRUE`)
`pmin(x,y,...)`	a vector which ith element is the minimum of `x[i]`, `y[i]`, ...
`pmax(x,y,...)`	id. for the maximum
`cumsum(x)`	a vector which ith element is the sum from `x[1]` to `x[i]`
`cumprod(x)`	id. for the product
`cummin(x)`	id. for the minimum
`cummax(x)`	id. for the maximum
`match(x, y)`	returns a vector of the same length than `x` with the elements of `x` which are in `y` (`NA` otherwise)
`which(x == a)`	returns a vector of the indices of `x` if the comparison operation is true (`TRUE`), in this example the values of `i` for which `x[i] == a` (the argument of this function must be a variable of mode logical)
`choose(n, k)`	computes the combinations of k events among n repetitions $= n!/[(n-k)!k!]$
`na.omit(x)`	suppresses the observations with missing data (`NA`) (suppresses the corresponding line if `x` is a matrix or a data frame)
`na.fail(x)`	returns an error message if `x` contains at least one `NA`
`unique(x)`	if `x` is a vector or a data frame, returns a similar object but with the duplicate elements suppressed
`table(x)`	returns a table with the numbers of the differents values of `x` (typically for integers or factors)
`table(x, y)`	contingency table of `x` and `y`
`subset(x, ...)`	returns a selection of `x` with respect to criteria (`...`, typically comparisons: `x$V1 < 10`); if `x` is a data frame, the option `select` gives the variables to be kept (or dropped using a minus sign)
`sample(x, size)`	resample randomly and without replacement `size` elements in the vector `x`, the option `replace = TRUE` allows to resample with replacement

3.4.2 *Random sequences*

law	function
Gaussian (normal)	`rnorm(n, mean=0, sd=1)`
exponential	`rexp(n, rate=1)`
gamma	`rgamma(n, shape, scale=1)`
Poisson	`rpois(n, lambda)`
Weibull	`rweibull(n, shape, scale=1)`
Cauchy	`rcauchy(n, location=0, scale=1)`
beta	`rbeta(n, shape1, shape2)`
'Student' (t)	`rt(n, df)`
Fisher–Snedecor (F)	`rf(n, df1, df2)`
Pearson (χ^2)	`rchisq(n, df)`
binomial	`rbinom(n, size, prob)`
multinomial	`rmultinom(n, size, prob)`
geometric	`rgeom(n, prob)`
hypergeometric	`rhyper(nn, m, n, k)`
logistic	`rlogis(n, location=0, scale=1)`
lognormal	`rlnorm(n, meanlog=0, sdlog=1)`
negative binomial	`rnbinom(n, size, prob)`
uniform	`runif(n, min=0, max=1)`
Wilcoxon's statistics	`rwilcox(nn, m, n)`, `rsignrank(nn, n)`

It is useful in statistics to be able to generate random data, and R ca
do it for a large number of probability density functions. These functions a
of the form `r`*`func`*`(n, p1, p2, ...)`, where *`func`* indicates the probabili
distribution, `n` the number of data generated, and `p1`, `p2`, ... are the values
the parameters of the distribution. The above table gives the details for ea
distribution, and the possible default values (if none default value is indicate
this means that the parameter must be specified by the user).

Most of these functions have counterparts obtained by replacing the lett
`r` with `d`, `p` or `q` to get, respectively, the probability density (`d`*`func`*`(x, ...)`

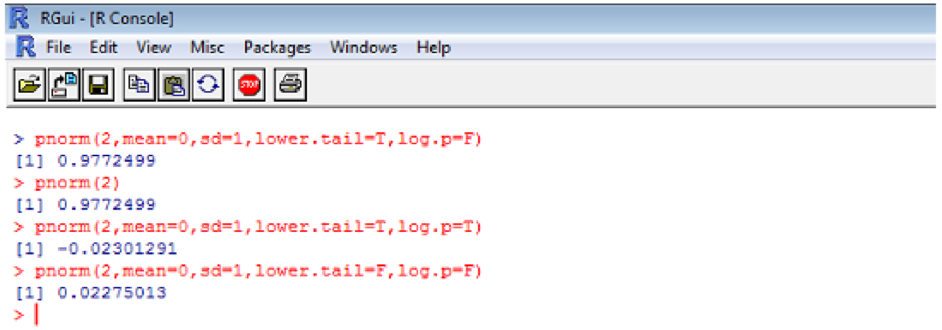
the cumulative probability density (`p`*`func`*`(x, ...)`), and the value of quantile (`q`*`func`*`(p, ...)`, with $0 <$ `p` < 1). The last two series of functions can be used to find critical values or P-values of statistical tests. For instance, the critical values for a two-tailed test following a normal distribution at the 5% threshold are:

```
> qnorm(0.025)
[1] -1.959964
> qnorm(0.975)
[1] 1.959964
```

For the one-tailed version of the same test, either `qnorm(0.05)` or `1 - qnorm(0.95)` will be used depending on the form of the alternative hypothesis.

The P-value of a test, say $\chi^2 = 3.84$ with $df = 1$, is:

```
> 1 - pchisq(3.84, 1)
[1] 0.05004352
```



```
> pnorm(2,mean=0,sd=1,lower.tail=T,log.p=F)
[1] 0.9772499
> pnorm(2)
[1] 0.9772499
> pnorm(2,mean=0,sd=1,lower.tail=T,log.p=T)
[1] -0.02301291
> pnorm(2,mean=0,sd=1,lower.tail=F,log.p=F)
[1] 0.02275013
> |
```